>>> from pyspark.sql import \*

>>> spark=SparkSession.builder.master("local[2]").appName("Hackathon").enableHiveSupport().getOrCreate()

>>> sc=spark.sparkContext

>>> from pyspark.sql.types import \*

>>> custom\_schema=StructType ([StructField("IssuerId",IntegerType(),True) , StructField("IssuerId2",IntegerType(),True) , StructField("BusinessDate",DateType(),True) , StructField("StateCode",StringType(),True) , StructField("SourceName",StringType(),True) , StructField("NetworkName",StringType(),True) , StructField("NetworkURL",StringType(),True) , StructField("custnum",StringType(),True) , StructField("MarketCoverage",StringType(),True) , StructField("DentalOnlyPlan",StringType(),True)])

>>> data\_frame\_1=spark.read.csv(path="/user/hduser/Desktop/Hackathon/insuranceinfo1.csv" , header=True , schema=custom\_schema )

Traceback (most recent call last):

File "<stdin>", line 1, in <module>

File "/usr/local/spark/python/pyspark/sql/readwriter.py", line 410, in csv

return self.\_df(self.\_jreader.csv(self.\_spark.\_sc.\_jvm.PythonUtils.toSeq(path)))

File "/usr/local/spark/python/lib/py4j-0.10.9.3-src.zip/py4j/java\_gateway.py", line 1322, in \_\_call\_\_

File "/usr/local/spark/python/pyspark/sql/utils.py", line 117, in deco

raise converted from None

pyspark.sql.utils.AnalysisException: Path does not exist: hdfs://localhost:54310/user/hduser/Desktop/Hackathon/insuranceinfo1.csv

>>> data\_frame\_1=spark.read.csv(path="/home/hduser/Desktop/Hackathon/insuranceinfo1.csv" , header=True , schema=custom\_schema )

Traceback (most recent call last):

File "<stdin>", line 1, in <module>

File "/usr/local/spark/python/pyspark/sql/readwriter.py", line 410, in csv

return self.\_df(self.\_jreader.csv(self.\_spark.\_sc.\_jvm.PythonUtils.toSeq(path)))

File "/usr/local/spark/python/lib/py4j-0.10.9.3-src.zip/py4j/java\_gateway.py", line 1322, in \_\_call\_\_

File "/usr/local/spark/python/pyspark/sql/utils.py", line 117, in deco

raise converted from None

pyspark.sql.utils.AnalysisException: Path does not exist: hdfs://localhost:54310/home/hduser/Desktop/Hackathon/insuranceinfo1.csv

>>> data\_frame\_1=spark.read.csv(path="/user/hduser/Hackathon/insuranceinfo1.csv" , header=True , schema=custom\_schema )

>>> custom\_schema=StructType ([StructField("IssuerId",IntegerType(),True) , StructField("IssuerId2",IntegerType(),True) , StructField("BusinessDate",DateType(),True) , StructField("StateCode",StringType(),True) , StructField("SourceName",StringType(),True) , StructField("NetworkName",StringType(),True) , StructField("NetworkURL",StringType(),True) , StructField("custnum",StringType(),True) , StructField("MarketCoverage",StringType(),True) , StructField("DentalOnlyPlan",StringType(),True)])

>>>

>>> data\_frame\_2=spark.read.csv(path="/user/hduser/Hackathon/insuranceinfo2.csv" , header=True , schema=custom\_schema )

>>> custom\_schema=StructType ([StructField("IssuerId",IntegerType(),True) , StructField("IssuerId2",IntegerType(),True) , StructField("BusinessDate",DateType(),True) , StructField("StateCode",StringType(),True) , StructField("SourceName",StringType(),True) , StructField("NetworkName",StringType(),True) , StructField("NetworkURL",StringType(),True) , StructField("custnum",StringType(),True) , StructField("MarketCoverage",StringType(),True) , StructField("DentalOnlyPlan",StringType(),True) , StructField("RejectRows",StringType(),True) ])

>>> data\_frame\_3=spark.read.csv(path="dbfs:/FileStore/Hackathon/insuranceinfo2.csv" , header=True , schema=custom\_schema , columnNameOfCorruptRecord="RejectRows" )

24/07/18 20:02:00 WARN streaming.FileStreamSink: Assume no metadata directory. Error while looking for metadata directory in the path: dbfs:/FileStore/Hackathon/insuranceinfo2.csv.

java.io.IOException: No FileSystem for scheme: dbfs

at org.apache.hadoop.fs.FileSystem.getFileSystemClass(FileSystem.java:2660)

at org.apache.hadoop.fs.FileSystem.createFileSystem(FileSystem.java:2667)

at org.apache.hadoop.fs.FileSystem.access$200(FileSystem.java:94)

at org.apache.hadoop.fs.FileSystem$Cache.getInternal(FileSystem.java:2703)

at org.apache.hadoop.fs.FileSystem$Cache.get(FileSystem.java:2685)

at org.apache.hadoop.fs.FileSystem.get(FileSystem.java:373)

at org.apache.hadoop.fs.Path.getFileSystem(Path.java:295)

at org.apache.spark.sql.execution.streaming.FileStreamSink$.hasMetadata(FileStreamSink.scala:53)

at org.apache.spark.sql.execution.datasources.DataSource.resolveRelation(DataSource.scala:370)

at org.apache.spark.sql.DataFrameReader.loadV1Source(DataFrameReader.scala:274)

at org.apache.spark.sql.DataFrameReader.$anonfun$load$3(DataFrameReader.scala:245)

at scala.Option.getOrElse(Option.scala:189)

at org.apache.spark.sql.DataFrameReader.load(DataFrameReader.scala:245)

at org.apache.spark.sql.DataFrameReader.csv(DataFrameReader.scala:571)

at sun.reflect.NativeMethodAccessorImpl.invoke0(Native Method)

at sun.reflect.NativeMethodAccessorImpl.invoke(NativeMethodAccessorImpl.java:62)

at sun.reflect.DelegatingMethodAccessorImpl.invoke(DelegatingMethodAccessorImpl.java:43)

at java.lang.reflect.Method.invoke(Method.java:498)

at py4j.reflection.MethodInvoker.invoke(MethodInvoker.java:244)

at py4j.reflection.ReflectionEngine.invoke(ReflectionEngine.java:357)

at py4j.Gateway.invoke(Gateway.java:282)

at py4j.commands.AbstractCommand.invokeMethod(AbstractCommand.java:132)

at py4j.commands.CallCommand.execute(CallCommand.java:79)

at py4j.ClientServerConnection.waitForCommands(ClientServerConnection.java:182)

at py4j.ClientServerConnection.run(ClientServerConnection.java:106)

at java.lang.Thread.run(Thread.java:750)

Traceback (most recent call last):

File "<stdin>", line 1, in <module>

File "/usr/local/spark/python/pyspark/sql/readwriter.py", line 410, in csv

return self.\_df(self.\_jreader.csv(self.\_spark.\_sc.\_jvm.PythonUtils.toSeq(path)))

File "/usr/local/spark/python/lib/py4j-0.10.9.3-src.zip/py4j/java\_gateway.py", line 1322, in \_\_call\_\_

File "/usr/local/spark/python/pyspark/sql/utils.py", line 111, in deco

return f(\*a, \*\*kw)

File "/usr/local/spark/python/lib/py4j-0.10.9.3-src.zip/py4j/protocol.py", line 328, in get\_return\_value

py4j.protocol.Py4JJavaError: An error occurred while calling o102.csv.

: java.io.IOException: No FileSystem for scheme: dbfs

at org.apache.hadoop.fs.FileSystem.getFileSystemClass(FileSystem.java:2660)

at org.apache.hadoop.fs.FileSystem.createFileSystem(FileSystem.java:2667)

at org.apache.hadoop.fs.FileSystem.access$200(FileSystem.java:94)

at org.apache.hadoop.fs.FileSystem$Cache.getInternal(FileSystem.java:2703)

at org.apache.hadoop.fs.FileSystem$Cache.get(FileSystem.java:2685)

at org.apache.hadoop.fs.FileSystem.get(FileSystem.java:373)

at org.apache.hadoop.fs.Path.getFileSystem(Path.java:295)

at org.apache.spark.sql.execution.datasources.DataSource$.$anonfun$checkAndGlobPathIfNecessary$1(DataSource.scala:747)

at scala.collection.immutable.List.map(List.scala:293)

at org.apache.spark.sql.execution.datasources.DataSource$.checkAndGlobPathIfNecessary(DataSource.scala:745)

at org.apache.spark.sql.execution.datasources.DataSource.checkAndGlobPathIfNecessary(DataSource.scala:577)

at org.apache.spark.sql.execution.datasources.DataSource.resolveRelation(DataSource.scala:408)

at org.apache.spark.sql.DataFrameReader.loadV1Source(DataFrameReader.scala:274)

at org.apache.spark.sql.DataFrameReader.$anonfun$load$3(DataFrameReader.scala:245)

at scala.Option.getOrElse(Option.scala:189)

at org.apache.spark.sql.DataFrameReader.load(DataFrameReader.scala:245)

at org.apache.spark.sql.DataFrameReader.csv(DataFrameReader.scala:571)

at sun.reflect.NativeMethodAccessorImpl.invoke0(Native Method)

at sun.reflect.NativeMethodAccessorImpl.invoke(NativeMethodAccessorImpl.java:62)

at sun.reflect.DelegatingMethodAccessorImpl.invoke(DelegatingMethodAccessorImpl.java:43)

at java.lang.reflect.Method.invoke(Method.java:498)

at py4j.reflection.MethodInvoker.invoke(MethodInvoker.java:244)

at py4j.reflection.ReflectionEngine.invoke(ReflectionEngine.java:357)

at py4j.Gateway.invoke(Gateway.java:282)

at py4j.commands.AbstractCommand.invokeMethod(AbstractCommand.java:132)

at py4j.commands.CallCommand.execute(CallCommand.java:79)

at py4j.ClientServerConnection.waitForCommands(ClientServerConnection.java:182)

at py4j.ClientServerConnection.run(ClientServerConnection.java:106)

at java.lang.Thread.run(Thread.java:750)

>>> data\_frame\_3=spark.read.csv(path="/user/hduser/Hackathon/insuranceinfo2.csv" , header=True , schema=custom\_schema , columnNameOfCorruptRecord="RejectRows" )

>>> data\_frame\_1.show(3)

+--------+---------+------------+---------+----------+------------+--------------------+-------+--------------+--------------+

|IssuerId|IssuerId2|BusinessDate|StateCode|SourceName| NetworkName| NetworkURL|custnum|MarketCoverage|DentalOnlyPlan|

+--------+---------+------------+---------+----------+------------+--------------------+-------+--------------+--------------+

| 21989| 21989| 2019-10-01| AK| HIOS| ODS Premier|https://www.modah...| 13| null| null|

| 219| 219| 2019-10-01| AK| HIOS| ODS Premier| null| null| null| null|

| 38344| 38344| 2019-10-01| AK| HIOS|HeritagePlus|https://www.preme...| 13| null| null|

+--------+---------+------------+---------+----------+------------+--------------------+-------+--------------+--------------+

only showing top 3 rows

>>> data\_frame\_2.show(3)

24/07/18 20:09:04 ERROR executor.Executor: Exception in task 0.0 in stage 1.0 (TID 1)

org.apache.spark.SparkUpgradeException: You may get a different result due to the upgrading of Spark 3.0: Fail to parse '01-10-2019' in the new parser. You can set spark.sql.legacy.timeParserPolicy to LEGACY to restore the behavior before Spark 3.0, or set to CORRECTED and treat it as an invalid datetime string.

at org.apache.spark.sql.errors.QueryExecutionErrors$.failToParseDateTimeInNewParserError(QueryExecutionErrors.scala:918)

at org.apache.spark.sql.catalyst.util.DateTimeFormatterHelper$$anonfun$checkParsedDiff$1.applyOrElse(DateTimeFormatterHelper.scala:148)

at org.apache.spark.sql.catalyst.util.DateTimeFormatterHelper$$anonfun$checkParsedDiff$1.applyOrElse(DateTimeFormatterHelper.scala:141)

at scala.runtime.AbstractPartialFunction.apply(AbstractPartialFunction.scala:38)

at org.apache.spark.sql.catalyst.util.Iso8601DateFormatter.parse(DateFormatter.scala:57)

at org.apache.spark.sql.catalyst.csv.UnivocityParser.$anonfun$makeConverter$21(UnivocityParser.scala:202)

at org.apache.spark.sql.catalyst.csv.UnivocityParser.nullSafeDatum(UnivocityParser.scala:248)

at org.apache.spark.sql.catalyst.csv.UnivocityParser.$anonfun$makeConverter$20(UnivocityParser.scala:200)

at org.apache.spark.sql.catalyst.csv.UnivocityParser.org$apache$spark$sql$catalyst$csv$UnivocityParser$$convert(UnivocityParser.scala:301)

at org.apache.spark.sql.catalyst.csv.UnivocityParser.$anonfun$parse$2(UnivocityParser.scala:264)

at org.apache.spark.sql.catalyst.csv.UnivocityParser$.$anonfun$parseIterator$1(UnivocityParser.scala:406)

at org.apache.spark.sql.catalyst.util.FailureSafeParser.parse(FailureSafeParser.scala:60)

at org.apache.spark.sql.catalyst.csv.UnivocityParser$.$anonfun$parseIterator$2(UnivocityParser.scala:410)

at scala.collection.Iterator$$anon$11.nextCur(Iterator.scala:486)

at scala.collection.Iterator$$anon$11.hasNext(Iterator.scala:492)

at scala.collection.Iterator$$anon$10.hasNext(Iterator.scala:460)

at org.apache.spark.sql.execution.datasources.FileScanRDD$$anon$1.hasNext(FileScanRDD.scala:104)

at org.apache.spark.sql.execution.datasources.FileScanRDD$$anon$1.nextIterator(FileScanRDD.scala:191)

at org.apache.spark.sql.execution.datasources.FileScanRDD$$anon$1.hasNext(FileScanRDD.scala:104)

at scala.collection.Iterator$$anon$10.hasNext(Iterator.scala:460)

at org.apache.spark.sql.catalyst.expressions.GeneratedClass$GeneratedIteratorForCodegenStage1.processNext(Unknown Source)

at org.apache.spark.sql.execution.BufferedRowIterator.hasNext(BufferedRowIterator.java:43)

at org.apache.spark.sql.execution.WholeStageCodegenExec$$anon$1.hasNext(WholeStageCodegenExec.scala:759)

at org.apache.spark.sql.execution.SparkPlan.$anonfun$getByteArrayRdd$1(SparkPlan.scala:349)

at org.apache.spark.rdd.RDD.$anonfun$mapPartitionsInternal$2(RDD.scala:898)

at org.apache.spark.rdd.RDD.$anonfun$mapPartitionsInternal$2$adapted(RDD.scala:898)

at org.apache.spark.rdd.MapPartitionsRDD.compute(MapPartitionsRDD.scala:52)

at org.apache.spark.rdd.RDD.computeOrReadCheckpoint(RDD.scala:373)

at org.apache.spark.rdd.RDD.iterator(RDD.scala:337)

at org.apache.spark.scheduler.ResultTask.runTask(ResultTask.scala:90)

at org.apache.spark.scheduler.Task.run(Task.scala:131)

at org.apache.spark.executor.Executor$TaskRunner.$anonfun$run$3(Executor.scala:506)

at org.apache.spark.util.Utils$.tryWithSafeFinally(Utils.scala:1462)

at org.apache.spark.executor.Executor$TaskRunner.run(Executor.scala:509)

at java.util.concurrent.ThreadPoolExecutor.runWorker(ThreadPoolExecutor.java:1149)

at java.util.concurrent.ThreadPoolExecutor$Worker.run(ThreadPoolExecutor.java:624)

at java.lang.Thread.run(Thread.java:750)

Caused by: java.time.format.DateTimeParseException: Text '01-10-2019' could not be parsed at index 0

at java.time.format.DateTimeFormatter.parseResolved0(DateTimeFormatter.java:1949)

at java.time.format.DateTimeFormatter.parse(DateTimeFormatter.java:1777)

at org.apache.spark.sql.catalyst.util.Iso8601DateFormatter.parse(DateFormatter.scala:55)

... 32 more

24/07/18 20:09:05 WARN scheduler.TaskSetManager: Lost task 0.0 in stage 1.0 (TID 1) (192.168.112.129 executor driver): org.apache.spark.SparkUpgradeException: You may get a different result due to the upgrading of Spark 3.0: Fail to parse '01-10-2019' in the new parser. You can set spark.sql.legacy.timeParserPolicy to LEGACY to restore the behavior before Spark 3.0, or set to CORRECTED and treat it as an invalid datetime string.

at org.apache.spark.sql.errors.QueryExecutionErrors$.failToParseDateTimeInNewParserError(QueryExecutionErrors.scala:918)

at org.apache.spark.sql.catalyst.util.DateTimeFormatterHelper$$anonfun$checkParsedDiff$1.applyOrElse(DateTimeFormatterHelper.scala:148)

at org.apache.spark.sql.catalyst.util.DateTimeFormatterHelper$$anonfun$checkParsedDiff$1.applyOrElse(DateTimeFormatterHelper.scala:141)

at scala.runtime.AbstractPartialFunction.apply(AbstractPartialFunction.scala:38)

at org.apache.spark.sql.catalyst.util.Iso8601DateFormatter.parse(DateFormatter.scala:57)

at org.apache.spark.sql.catalyst.csv.UnivocityParser.$anonfun$makeConverter$21(UnivocityParser.scala:202)

at org.apache.spark.sql.catalyst.csv.UnivocityParser.nullSafeDatum(UnivocityParser.scala:248)

at org.apache.spark.sql.catalyst.csv.UnivocityParser.$anonfun$makeConverter$20(UnivocityParser.scala:200)

at org.apache.spark.sql.catalyst.csv.UnivocityParser.org$apache$spark$sql$catalyst$csv$UnivocityParser$$convert(UnivocityParser.scala:301)

at org.apache.spark.sql.catalyst.csv.UnivocityParser.$anonfun$parse$2(UnivocityParser.scala:264)

at org.apache.spark.sql.catalyst.csv.UnivocityParser$.$anonfun$parseIterator$1(UnivocityParser.scala:406)

at org.apache.spark.sql.catalyst.util.FailureSafeParser.parse(FailureSafeParser.scala:60)

at org.apache.spark.sql.catalyst.csv.UnivocityParser$.$anonfun$parseIterator$2(UnivocityParser.scala:410)

at scala.collection.Iterator$$anon$11.nextCur(Iterator.scala:486)

at scala.collection.Iterator$$anon$11.hasNext(Iterator.scala:492)

at scala.collection.Iterator$$anon$10.hasNext(Iterator.scala:460)

at org.apache.spark.sql.execution.datasources.FileScanRDD$$anon$1.hasNext(FileScanRDD.scala:104)

at org.apache.spark.sql.execution.datasources.FileScanRDD$$anon$1.nextIterator(FileScanRDD.scala:191)

at org.apache.spark.sql.execution.datasources.FileScanRDD$$anon$1.hasNext(FileScanRDD.scala:104)

at scala.collection.Iterator$$anon$10.hasNext(Iterator.scala:460)

at org.apache.spark.sql.catalyst.expressions.GeneratedClass$GeneratedIteratorForCodegenStage1.processNext(Unknown Source)

at org.apache.spark.sql.execution.BufferedRowIterator.hasNext(BufferedRowIterator.java:43)

at org.apache.spark.sql.execution.WholeStageCodegenExec$$anon$1.hasNext(WholeStageCodegenExec.scala:759)

at org.apache.spark.sql.execution.SparkPlan.$anonfun$getByteArrayRdd$1(SparkPlan.scala:349)

at org.apache.spark.rdd.RDD.$anonfun$mapPartitionsInternal$2(RDD.scala:898)

at org.apache.spark.rdd.RDD.$anonfun$mapPartitionsInternal$2$adapted(RDD.scala:898)

at org.apache.spark.rdd.MapPartitionsRDD.compute(MapPartitionsRDD.scala:52)

at org.apache.spark.rdd.RDD.computeOrReadCheckpoint(RDD.scala:373)

at org.apache.spark.rdd.RDD.iterator(RDD.scala:337)

at org.apache.spark.scheduler.ResultTask.runTask(ResultTask.scala:90)

at org.apache.spark.scheduler.Task.run(Task.scala:131)

at org.apache.spark.executor.Executor$TaskRunner.$anonfun$run$3(Executor.scala:506)

at org.apache.spark.util.Utils$.tryWithSafeFinally(Utils.scala:1462)

at org.apache.spark.executor.Executor$TaskRunner.run(Executor.scala:509)

at java.util.concurrent.ThreadPoolExecutor.runWorker(ThreadPoolExecutor.java:1149)

at java.util.concurrent.ThreadPoolExecutor$Worker.run(ThreadPoolExecutor.java:624)

at java.lang.Thread.run(Thread.java:750)

Caused by: java.time.format.DateTimeParseException: Text '01-10-2019' could not be parsed at index 0

at java.time.format.DateTimeFormatter.parseResolved0(DateTimeFormatter.java:1949)

at java.time.format.DateTimeFormatter.parse(DateTimeFormatter.java:1777)

at org.apache.spark.sql.catalyst.util.Iso8601DateFormatter.parse(DateFormatter.scala:55)

... 32 more

24/07/18 20:09:05 ERROR scheduler.TaskSetManager: Task 0 in stage 1.0 failed 1 times; aborting job

Traceback (most recent call last):

File "<stdin>", line 1, in <module>

File "/usr/local/spark/python/pyspark/sql/dataframe.py", line 494, in show

print(self.\_jdf.showString(n, 20, vertical))

File "/usr/local/spark/python/lib/py4j-0.10.9.3-src.zip/py4j/java\_gateway.py", line 1322, in \_\_call\_\_

File "/usr/local/spark/python/pyspark/sql/utils.py", line 111, in deco

return f(\*a, \*\*kw)

File "/usr/local/spark/python/lib/py4j-0.10.9.3-src.zip/py4j/protocol.py", line 328, in get\_return\_value

py4j.protocol.Py4JJavaError: An error occurred while calling o91.showString.

: org.apache.spark.SparkException: Job aborted due to stage failure: Task 0 in stage 1.0 failed 1 times, most recent failure: Lost task 0.0 in stage 1.0 (TID 1) (192.168.112.129 executor driver): org.apache.spark.SparkUpgradeException: You may get a different result due to the upgrading of Spark 3.0: Fail to parse '01-10-2019' in the new parser. You can set spark.sql.legacy.timeParserPolicy to LEGACY to restore the behavior before Spark 3.0, or set to CORRECTED and treat it as an invalid datetime string.

at org.apache.spark.sql.errors.QueryExecutionErrors$.failToParseDateTimeInNewParserError(QueryExecutionErrors.scala:918)

at org.apache.spark.sql.catalyst.util.DateTimeFormatterHelper$$anonfun$checkParsedDiff$1.applyOrElse(DateTimeFormatterHelper.scala:148)

at org.apache.spark.sql.catalyst.util.DateTimeFormatterHelper$$anonfun$checkParsedDiff$1.applyOrElse(DateTimeFormatterHelper.scala:141)

at scala.runtime.AbstractPartialFunction.apply(AbstractPartialFunction.scala:38)

at org.apache.spark.sql.catalyst.util.Iso8601DateFormatter.parse(DateFormatter.scala:57)

at org.apache.spark.sql.catalyst.csv.UnivocityParser.$anonfun$makeConverter$21(UnivocityParser.scala:202)

at org.apache.spark.sql.catalyst.csv.UnivocityParser.nullSafeDatum(UnivocityParser.scala:248)

at org.apache.spark.sql.catalyst.csv.UnivocityParser.$anonfun$makeConverter$20(UnivocityParser.scala:200)

at org.apache.spark.sql.catalyst.csv.UnivocityParser.org$apache$spark$sql$catalyst$csv$UnivocityParser$$convert(UnivocityParser.scala:301)

at org.apache.spark.sql.catalyst.csv.UnivocityParser.$anonfun$parse$2(UnivocityParser.scala:264)

at org.apache.spark.sql.catalyst.csv.UnivocityParser$.$anonfun$parseIterator$1(UnivocityParser.scala:406)

at org.apache.spark.sql.catalyst.util.FailureSafeParser.parse(FailureSafeParser.scala:60)

at org.apache.spark.sql.catalyst.csv.UnivocityParser$.$anonfun$parseIterator$2(UnivocityParser.scala:410)

at scala.collection.Iterator$$anon$11.nextCur(Iterator.scala:486)

at scala.collection.Iterator$$anon$11.hasNext(Iterator.scala:492)

at scala.collection.Iterator$$anon$10.hasNext(Iterator.scala:460)

at org.apache.spark.sql.execution.datasources.FileScanRDD$$anon$1.hasNext(FileScanRDD.scala:104)

at org.apache.spark.sql.execution.datasources.FileScanRDD$$anon$1.nextIterator(FileScanRDD.scala:191)

at org.apache.spark.sql.execution.datasources.FileScanRDD$$anon$1.hasNext(FileScanRDD.scala:104)

at scala.collection.Iterator$$anon$10.hasNext(Iterator.scala:460)

at org.apache.spark.sql.catalyst.expressions.GeneratedClass$GeneratedIteratorForCodegenStage1.processNext(Unknown Source)

at org.apache.spark.sql.execution.BufferedRowIterator.hasNext(BufferedRowIterator.java:43)

at org.apache.spark.sql.execution.WholeStageCodegenExec$$anon$1.hasNext(WholeStageCodegenExec.scala:759)

at org.apache.spark.sql.execution.SparkPlan.$anonfun$getByteArrayRdd$1(SparkPlan.scala:349)

at org.apache.spark.rdd.RDD.$anonfun$mapPartitionsInternal$2(RDD.scala:898)

at org.apache.spark.rdd.RDD.$anonfun$mapPartitionsInternal$2$adapted(RDD.scala:898)

at org.apache.spark.rdd.MapPartitionsRDD.compute(MapPartitionsRDD.scala:52)

at org.apache.spark.rdd.RDD.computeOrReadCheckpoint(RDD.scala:373)

at org.apache.spark.rdd.RDD.iterator(RDD.scala:337)

at org.apache.spark.scheduler.ResultTask.runTask(ResultTask.scala:90)

at org.apache.spark.scheduler.Task.run(Task.scala:131)

at org.apache.spark.executor.Executor$TaskRunner.$anonfun$run$3(Executor.scala:506)

at org.apache.spark.util.Utils$.tryWithSafeFinally(Utils.scala:1462)

at org.apache.spark.executor.Executor$TaskRunner.run(Executor.scala:509)

at java.util.concurrent.ThreadPoolExecutor.runWorker(ThreadPoolExecutor.java:1149)

at java.util.concurrent.ThreadPoolExecutor$Worker.run(ThreadPoolExecutor.java:624)

at java.lang.Thread.run(Thread.java:750)

Caused by: java.time.format.DateTimeParseException: Text '01-10-2019' could not be parsed at index 0

at java.time.format.DateTimeFormatter.parseResolved0(DateTimeFormatter.java:1949)

at java.time.format.DateTimeFormatter.parse(DateTimeFormatter.java:1777)

at org.apache.spark.sql.catalyst.util.Iso8601DateFormatter.parse(DateFormatter.scala:55)

... 32 more

Driver stacktrace:

at org.apache.spark.scheduler.DAGScheduler.failJobAndIndependentStages(DAGScheduler.scala:2454)

at org.apache.spark.scheduler.DAGScheduler.$anonfun$abortStage$2(DAGScheduler.scala:2403)

at org.apache.spark.scheduler.DAGScheduler.$anonfun$abortStage$2$adapted(DAGScheduler.scala:2402)

at scala.collection.mutable.ResizableArray.foreach(ResizableArray.scala:62)

at scala.collection.mutable.ResizableArray.foreach$(ResizableArray.scala:55)

at scala.collection.mutable.ArrayBuffer.foreach(ArrayBuffer.scala:49)

at org.apache.spark.scheduler.DAGScheduler.abortStage(DAGScheduler.scala:2402)

at org.apache.spark.scheduler.DAGScheduler.$anonfun$handleTaskSetFailed$1(DAGScheduler.scala:1160)

at org.apache.spark.scheduler.DAGScheduler.$anonfun$handleTaskSetFailed$1$adapted(DAGScheduler.scala:1160)

at scala.Option.foreach(Option.scala:407)

at org.apache.spark.scheduler.DAGScheduler.handleTaskSetFailed(DAGScheduler.scala:1160)

at org.apache.spark.scheduler.DAGSchedulerEventProcessLoop.doOnReceive(DAGScheduler.scala:2642)

at org.apache.spark.scheduler.DAGSchedulerEventProcessLoop.onReceive(DAGScheduler.scala:2584)

at org.apache.spark.scheduler.DAGSchedulerEventProcessLoop.onReceive(DAGScheduler.scala:2573)

at org.apache.spark.util.EventLoop$$anon$1.run(EventLoop.scala:49)

at org.apache.spark.scheduler.DAGScheduler.runJob(DAGScheduler.scala:938)

at org.apache.spark.SparkContext.runJob(SparkContext.scala:2214)

at org.apache.spark.SparkContext.runJob(SparkContext.scala:2235)

at org.apache.spark.SparkContext.runJob(SparkContext.scala:2254)

at org.apache.spark.sql.execution.SparkPlan.executeTake(SparkPlan.scala:476)

at org.apache.spark.sql.execution.SparkPlan.executeTake(SparkPlan.scala:429)

at org.apache.spark.sql.execution.CollectLimitExec.executeCollect(limit.scala:48)

at org.apache.spark.sql.Dataset.collectFromPlan(Dataset.scala:3715)

at org.apache.spark.sql.Dataset.$anonfun$head$1(Dataset.scala:2728)

at org.apache.spark.sql.Dataset.$anonfun$withAction$1(Dataset.scala:3706)

at org.apache.spark.sql.execution.SQLExecution$.$anonfun$withNewExecutionId$5(SQLExecution.scala:103)

at org.apache.spark.sql.execution.SQLExecution$.withSQLConfPropagated(SQLExecution.scala:163)

at org.apache.spark.sql.execution.SQLExecution$.$anonfun$withNewExecutionId$1(SQLExecution.scala:90)

at org.apache.spark.sql.SparkSession.withActive(SparkSession.scala:775)

at org.apache.spark.sql.execution.SQLExecution$.withNewExecutionId(SQLExecution.scala:64)

at org.apache.spark.sql.Dataset.withAction(Dataset.scala:3704)

at org.apache.spark.sql.Dataset.head(Dataset.scala:2728)

at org.apache.spark.sql.Dataset.take(Dataset.scala:2935)

at org.apache.spark.sql.Dataset.getRows(Dataset.scala:287)

at org.apache.spark.sql.Dataset.showString(Dataset.scala:326)

at sun.reflect.NativeMethodAccessorImpl.invoke0(Native Method)

at sun.reflect.NativeMethodAccessorImpl.invoke(NativeMethodAccessorImpl.java:62)

at sun.reflect.DelegatingMethodAccessorImpl.invoke(DelegatingMethodAccessorImpl.java:43)

at java.lang.reflect.Method.invoke(Method.java:498)

at py4j.reflection.MethodInvoker.invoke(MethodInvoker.java:244)

at py4j.reflection.ReflectionEngine.invoke(ReflectionEngine.java:357)

at py4j.Gateway.invoke(Gateway.java:282)

at py4j.commands.AbstractCommand.invokeMethod(AbstractCommand.java:132)

at py4j.commands.CallCommand.execute(CallCommand.java:79)

at py4j.ClientServerConnection.waitForCommands(ClientServerConnection.java:182)

at py4j.ClientServerConnection.run(ClientServerConnection.java:106)

at java.lang.Thread.run(Thread.java:750)

Caused by: org.apache.spark.SparkUpgradeException: You may get a different result due to the upgrading of Spark 3.0: Fail to parse '01-10-2019' in the new parser. You can set spark.sql.legacy.timeParserPolicy to LEGACY to restore the behavior before Spark 3.0, or set to CORRECTED and treat it as an invalid datetime string.

at org.apache.spark.sql.errors.QueryExecutionErrors$.failToParseDateTimeInNewParserError(QueryExecutionErrors.scala:918)

at org.apache.spark.sql.catalyst.util.DateTimeFormatterHelper$$anonfun$checkParsedDiff$1.applyOrElse(DateTimeFormatterHelper.scala:148)

at org.apache.spark.sql.catalyst.util.DateTimeFormatterHelper$$anonfun$checkParsedDiff$1.applyOrElse(DateTimeFormatterHelper.scala:141)

at scala.runtime.AbstractPartialFunction.apply(AbstractPartialFunction.scala:38)

at org.apache.spark.sql.catalyst.util.Iso8601DateFormatter.parse(DateFormatter.scala:57)

at org.apache.spark.sql.catalyst.csv.UnivocityParser.$anonfun$makeConverter$21(UnivocityParser.scala:202)

at org.apache.spark.sql.catalyst.csv.UnivocityParser.nullSafeDatum(UnivocityParser.scala:248)

at org.apache.spark.sql.catalyst.csv.UnivocityParser.$anonfun$makeConverter$20(UnivocityParser.scala:200)

at org.apache.spark.sql.catalyst.csv.UnivocityParser.org$apache$spark$sql$catalyst$csv$UnivocityParser$$convert(UnivocityParser.scala:301)

at org.apache.spark.sql.catalyst.csv.UnivocityParser.$anonfun$parse$2(UnivocityParser.scala:264)

at org.apache.spark.sql.catalyst.csv.UnivocityParser$.$anonfun$parseIterator$1(UnivocityParser.scala:406)

at org.apache.spark.sql.catalyst.util.FailureSafeParser.parse(FailureSafeParser.scala:60)

at org.apache.spark.sql.catalyst.csv.UnivocityParser$.$anonfun$parseIterator$2(UnivocityParser.scala:410)

at scala.collection.Iterator$$anon$11.nextCur(Iterator.scala:486)

at scala.collection.Iterator$$anon$11.hasNext(Iterator.scala:492)

at scala.collection.Iterator$$anon$10.hasNext(Iterator.scala:460)

at org.apache.spark.sql.execution.datasources.FileScanRDD$$anon$1.hasNext(FileScanRDD.scala:104)

at org.apache.spark.sql.execution.datasources.FileScanRDD$$anon$1.nextIterator(FileScanRDD.scala:191)

at org.apache.spark.sql.execution.datasources.FileScanRDD$$anon$1.hasNext(FileScanRDD.scala:104)

at scala.collection.Iterator$$anon$10.hasNext(Iterator.scala:460)

at org.apache.spark.sql.catalyst.expressions.GeneratedClass$GeneratedIteratorForCodegenStage1.processNext(Unknown Source)

at org.apache.spark.sql.execution.BufferedRowIterator.hasNext(BufferedRowIterator.java:43)

at org.apache.spark.sql.execution.WholeStageCodegenExec$$anon$1.hasNext(WholeStageCodegenExec.scala:759)

at org.apache.spark.sql.execution.SparkPlan.$anonfun$getByteArrayRdd$1(SparkPlan.scala:349)

at org.apache.spark.rdd.RDD.$anonfun$mapPartitionsInternal$2(RDD.scala:898)

at org.apache.spark.rdd.RDD.$anonfun$mapPartitionsInternal$2$adapted(RDD.scala:898)

at org.apache.spark.rdd.MapPartitionsRDD.compute(MapPartitionsRDD.scala:52)

at org.apache.spark.rdd.RDD.computeOrReadCheckpoint(RDD.scala:373)

at org.apache.spark.rdd.RDD.iterator(RDD.scala:337)

at org.apache.spark.scheduler.ResultTask.runTask(ResultTask.scala:90)

at org.apache.spark.scheduler.Task.run(Task.scala:131)

at org.apache.spark.executor.Executor$TaskRunner.$anonfun$run$3(Executor.scala:506)

at org.apache.spark.util.Utils$.tryWithSafeFinally(Utils.scala:1462)

at org.apache.spark.executor.Executor$TaskRunner.run(Executor.scala:509)

at java.util.concurrent.ThreadPoolExecutor.runWorker(ThreadPoolExecutor.java:1149)

at java.util.concurrent.ThreadPoolExecutor$Worker.run(ThreadPoolExecutor.java:624)

... 1 more

Caused by: java.time.format.DateTimeParseException: Text '01-10-2019' could not be parsed at index 0

at java.time.format.DateTimeFormatter.parseResolved0(DateTimeFormatter.java:1949)

at java.time.format.DateTimeFormatter.parse(DateTimeFormatter.java:1777)

at org.apache.spark.sql.catalyst.util.Iso8601DateFormatter.parse(DateFormatter.scala:55)

... 32 more

>>> data\_frame\_1.show(3)

+--------+---------+------------+---------+----------+------------+--------------------+-------+--------------+--------------+

|IssuerId|IssuerId2|BusinessDate|StateCode|SourceName| NetworkName| NetworkURL|custnum|MarketCoverage|DentalOnlyPlan|

+--------+---------+------------+---------+----------+------------+--------------------+-------+--------------+--------------+

| 21989| 21989| 2019-10-01| AK| HIOS| ODS Premier|https://www.modah...| 13| null| null|

| 219| 219| 2019-10-01| AK| HIOS| ODS Premier| null| null| null| null|

| 38344| 38344| 2019-10-01| AK| HIOS|HeritagePlus|https://www.preme...| 13| null| null|

+--------+---------+------------+---------+----------+------------+--------------------+-------+--------------+--------------+

only showing top 3 rows

>>> data\_frame\_2=spark.read.csv(path="/user/hduser/Hackathon/insuranceinfo2.csv" , header=True , schema=custom\_schema )

>>> data\_frame\_2.show(3)

24/07/18 20:09:43 WARN csv.CSVHeaderChecker: Number of column in CSV header is not equal to number of fields in the schema:

Header length: 10, schema size: 11

CSV file: hdfs://localhost:54310/user/hduser/Hackathon/insuranceinfo2.csv

24/07/18 20:09:43 ERROR executor.Executor: Exception in task 0.0 in stage 3.0 (TID 3)

org.apache.spark.SparkUpgradeException: You may get a different result due to the upgrading of Spark 3.0: Fail to parse '01-10-2019' in the new parser. You can set spark.sql.legacy.timeParserPolicy to LEGACY to restore the behavior before Spark 3.0, or set to CORRECTED and treat it as an invalid datetime string.

at org.apache.spark.sql.errors.QueryExecutionErrors$.failToParseDateTimeInNewParserError(QueryExecutionErrors.scala:918)

at org.apache.spark.sql.catalyst.util.DateTimeFormatterHelper$$anonfun$checkParsedDiff$1.applyOrElse(DateTimeFormatterHelper.scala:148)

at org.apache.spark.sql.catalyst.util.DateTimeFormatterHelper$$anonfun$checkParsedDiff$1.applyOrElse(DateTimeFormatterHelper.scala:141)

at scala.runtime.AbstractPartialFunction.apply(AbstractPartialFunction.scala:38)

at org.apache.spark.sql.catalyst.util.Iso8601DateFormatter.parse(DateFormatter.scala:57)

at org.apache.spark.sql.catalyst.csv.UnivocityParser.$anonfun$makeConverter$21(UnivocityParser.scala:202)

at org.apache.spark.sql.catalyst.csv.UnivocityParser.nullSafeDatum(UnivocityParser.scala:248)

at org.apache.spark.sql.catalyst.csv.UnivocityParser.$anonfun$makeConverter$20(UnivocityParser.scala:200)

at org.apache.spark.sql.catalyst.csv.UnivocityParser.org$apache$spark$sql$catalyst$csv$UnivocityParser$$convert(UnivocityParser.scala:301)

at org.apache.spark.sql.catalyst.csv.UnivocityParser.$anonfun$parse$2(UnivocityParser.scala:264)

at org.apache.spark.sql.catalyst.csv.UnivocityParser$.$anonfun$parseIterator$1(UnivocityParser.scala:406)

at org.apache.spark.sql.catalyst.util.FailureSafeParser.parse(FailureSafeParser.scala:60)

at org.apache.spark.sql.catalyst.csv.UnivocityParser$.$anonfun$parseIterator$2(UnivocityParser.scala:410)

at scala.collection.Iterator$$anon$11.nextCur(Iterator.scala:486)

at scala.collection.Iterator$$anon$11.hasNext(Iterator.scala:492)

at scala.collection.Iterator$$anon$10.hasNext(Iterator.scala:460)

at org.apache.spark.sql.execution.datasources.FileScanRDD$$anon$1.hasNext(FileScanRDD.scala:104)

at org.apache.spark.sql.execution.datasources.FileScanRDD$$anon$1.nextIterator(FileScanRDD.scala:191)

at org.apache.spark.sql.execution.datasources.FileScanRDD$$anon$1.hasNext(FileScanRDD.scala:104)

at scala.collection.Iterator$$anon$10.hasNext(Iterator.scala:460)

at org.apache.spark.sql.catalyst.expressions.GeneratedClass$GeneratedIteratorForCodegenStage1.processNext(Unknown Source)

at org.apache.spark.sql.execution.BufferedRowIterator.hasNext(BufferedRowIterator.java:43)

at org.apache.spark.sql.execution.WholeStageCodegenExec$$anon$1.hasNext(WholeStageCodegenExec.scala:759)

at org.apache.spark.sql.execution.SparkPlan.$anonfun$getByteArrayRdd$1(SparkPlan.scala:349)

at org.apache.spark.rdd.RDD.$anonfun$mapPartitionsInternal$2(RDD.scala:898)

at org.apache.spark.rdd.RDD.$anonfun$mapPartitionsInternal$2$adapted(RDD.scala:898)

at org.apache.spark.rdd.MapPartitionsRDD.compute(MapPartitionsRDD.scala:52)

at org.apache.spark.rdd.RDD.computeOrReadCheckpoint(RDD.scala:373)

at org.apache.spark.rdd.RDD.iterator(RDD.scala:337)

at org.apache.spark.scheduler.ResultTask.runTask(ResultTask.scala:90)

at org.apache.spark.scheduler.Task.run(Task.scala:131)

at org.apache.spark.executor.Executor$TaskRunner.$anonfun$run$3(Executor.scala:506)

at org.apache.spark.util.Utils$.tryWithSafeFinally(Utils.scala:1462)

at org.apache.spark.executor.Executor$TaskRunner.run(Executor.scala:509)

at java.util.concurrent.ThreadPoolExecutor.runWorker(ThreadPoolExecutor.java:1149)

at java.util.concurrent.ThreadPoolExecutor$Worker.run(ThreadPoolExecutor.java:624)

at java.lang.Thread.run(Thread.java:750)

Caused by: java.time.format.DateTimeParseException: Text '01-10-2019' could not be parsed at index 0

at java.time.format.DateTimeFormatter.parseResolved0(DateTimeFormatter.java:1949)

at java.time.format.DateTimeFormatter.parse(DateTimeFormatter.java:1777)

at org.apache.spark.sql.catalyst.util.Iso8601DateFormatter.parse(DateFormatter.scala:55)

... 32 more

24/07/18 20:09:43 WARN scheduler.TaskSetManager: Lost task 0.0 in stage 3.0 (TID 3) (192.168.112.129 executor driver): org.apache.spark.SparkUpgradeException: You may get a different result due to the upgrading of Spark 3.0: Fail to parse '01-10-2019' in the new parser. You can set spark.sql.legacy.timeParserPolicy to LEGACY to restore the behavior before Spark 3.0, or set to CORRECTED and treat it as an invalid datetime string.

at org.apache.spark.sql.errors.QueryExecutionErrors$.failToParseDateTimeInNewParserError(QueryExecutionErrors.scala:918)

at org.apache.spark.sql.catalyst.util.DateTimeFormatterHelper$$anonfun$checkParsedDiff$1.applyOrElse(DateTimeFormatterHelper.scala:148)

at org.apache.spark.sql.catalyst.util.DateTimeFormatterHelper$$anonfun$checkParsedDiff$1.applyOrElse(DateTimeFormatterHelper.scala:141)

at scala.runtime.AbstractPartialFunction.apply(AbstractPartialFunction.scala:38)

at org.apache.spark.sql.catalyst.util.Iso8601DateFormatter.parse(DateFormatter.scala:57)

at org.apache.spark.sql.catalyst.csv.UnivocityParser.$anonfun$makeConverter$21(UnivocityParser.scala:202)

at org.apache.spark.sql.catalyst.csv.UnivocityParser.nullSafeDatum(UnivocityParser.scala:248)

at org.apache.spark.sql.catalyst.csv.UnivocityParser.$anonfun$makeConverter$20(UnivocityParser.scala:200)

at org.apache.spark.sql.catalyst.csv.UnivocityParser.org$apache$spark$sql$catalyst$csv$UnivocityParser$$convert(UnivocityParser.scala:301)

at org.apache.spark.sql.catalyst.csv.UnivocityParser.$anonfun$parse$2(UnivocityParser.scala:264)

at org.apache.spark.sql.catalyst.csv.UnivocityParser$.$anonfun$parseIterator$1(UnivocityParser.scala:406)

at org.apache.spark.sql.catalyst.util.FailureSafeParser.parse(FailureSafeParser.scala:60)

at org.apache.spark.sql.catalyst.csv.UnivocityParser$.$anonfun$parseIterator$2(UnivocityParser.scala:410)

at scala.collection.Iterator$$anon$11.nextCur(Iterator.scala:486)

at scala.collection.Iterator$$anon$11.hasNext(Iterator.scala:492)

at scala.collection.Iterator$$anon$10.hasNext(Iterator.scala:460)

at org.apache.spark.sql.execution.datasources.FileScanRDD$$anon$1.hasNext(FileScanRDD.scala:104)

at org.apache.spark.sql.execution.datasources.FileScanRDD$$anon$1.nextIterator(FileScanRDD.scala:191)

at org.apache.spark.sql.execution.datasources.FileScanRDD$$anon$1.hasNext(FileScanRDD.scala:104)

at scala.collection.Iterator$$anon$10.hasNext(Iterator.scala:460)

at org.apache.spark.sql.catalyst.expressions.GeneratedClass$GeneratedIteratorForCodegenStage1.processNext(Unknown Source)

at org.apache.spark.sql.execution.BufferedRowIterator.hasNext(BufferedRowIterator.java:43)

at org.apache.spark.sql.execution.WholeStageCodegenExec$$anon$1.hasNext(WholeStageCodegenExec.scala:759)

at org.apache.spark.sql.execution.SparkPlan.$anonfun$getByteArrayRdd$1(SparkPlan.scala:349)

at org.apache.spark.rdd.RDD.$anonfun$mapPartitionsInternal$2(RDD.scala:898)

at org.apache.spark.rdd.RDD.$anonfun$mapPartitionsInternal$2$adapted(RDD.scala:898)

at org.apache.spark.rdd.MapPartitionsRDD.compute(MapPartitionsRDD.scala:52)

at org.apache.spark.rdd.RDD.computeOrReadCheckpoint(RDD.scala:373)

at org.apache.spark.rdd.RDD.iterator(RDD.scala:337)

at org.apache.spark.scheduler.ResultTask.runTask(ResultTask.scala:90)

at org.apache.spark.scheduler.Task.run(Task.scala:131)

at org.apache.spark.executor.Executor$TaskRunner.$anonfun$run$3(Executor.scala:506)

at org.apache.spark.util.Utils$.tryWithSafeFinally(Utils.scala:1462)

at org.apache.spark.executor.Executor$TaskRunner.run(Executor.scala:509)

at java.util.concurrent.ThreadPoolExecutor.runWorker(ThreadPoolExecutor.java:1149)

at java.util.concurrent.ThreadPoolExecutor$Worker.run(ThreadPoolExecutor.java:624)

at java.lang.Thread.run(Thread.java:750)

Caused by: java.time.format.DateTimeParseException: Text '01-10-2019' could not be parsed at index 0

at java.time.format.DateTimeFormatter.parseResolved0(DateTimeFormatter.java:1949)

at java.time.format.DateTimeFormatter.parse(DateTimeFormatter.java:1777)

at org.apache.spark.sql.catalyst.util.Iso8601DateFormatter.parse(DateFormatter.scala:55)

... 32 more

24/07/18 20:09:43 ERROR scheduler.TaskSetManager: Task 0 in stage 3.0 failed 1 times; aborting job

Traceback (most recent call last):

File "<stdin>", line 1, in <module>

File "/usr/local/spark/python/pyspark/sql/dataframe.py", line 494, in show

print(self.\_jdf.showString(n, 20, vertical))

File "/usr/local/spark/python/lib/py4j-0.10.9.3-src.zip/py4j/java\_gateway.py", line 1322, in \_\_call\_\_

File "/usr/local/spark/python/pyspark/sql/utils.py", line 111, in deco

return f(\*a, \*\*kw)

File "/usr/local/spark/python/lib/py4j-0.10.9.3-src.zip/py4j/protocol.py", line 328, in get\_return\_value

py4j.protocol.Py4JJavaError: An error occurred while calling o135.showString.

: org.apache.spark.SparkException: Job aborted due to stage failure: Task 0 in stage 3.0 failed 1 times, most recent failure: Lost task 0.0 in stage 3.0 (TID 3) (192.168.112.129 executor driver): org.apache.spark.SparkUpgradeException: You may get a different result due to the upgrading of Spark 3.0: Fail to parse '01-10-2019' in the new parser. You can set spark.sql.legacy.timeParserPolicy to LEGACY to restore the behavior before Spark 3.0, or set to CORRECTED and treat it as an invalid datetime string.

at org.apache.spark.sql.errors.QueryExecutionErrors$.failToParseDateTimeInNewParserError(QueryExecutionErrors.scala:918)

at org.apache.spark.sql.catalyst.util.DateTimeFormatterHelper$$anonfun$checkParsedDiff$1.applyOrElse(DateTimeFormatterHelper.scala:148)

at org.apache.spark.sql.catalyst.util.DateTimeFormatterHelper$$anonfun$checkParsedDiff$1.applyOrElse(DateTimeFormatterHelper.scala:141)

at scala.runtime.AbstractPartialFunction.apply(AbstractPartialFunction.scala:38)

at org.apache.spark.sql.catalyst.util.Iso8601DateFormatter.parse(DateFormatter.scala:57)

at org.apache.spark.sql.catalyst.csv.UnivocityParser.$anonfun$makeConverter$21(UnivocityParser.scala:202)

at org.apache.spark.sql.catalyst.csv.UnivocityParser.nullSafeDatum(UnivocityParser.scala:248)

at org.apache.spark.sql.catalyst.csv.UnivocityParser.$anonfun$makeConverter$20(UnivocityParser.scala:200)

at org.apache.spark.sql.catalyst.csv.UnivocityParser.org$apache$spark$sql$catalyst$csv$UnivocityParser$$convert(UnivocityParser.scala:301)

at org.apache.spark.sql.catalyst.csv.UnivocityParser.$anonfun$parse$2(UnivocityParser.scala:264)

at org.apache.spark.sql.catalyst.csv.UnivocityParser$.$anonfun$parseIterator$1(UnivocityParser.scala:406)

at org.apache.spark.sql.catalyst.util.FailureSafeParser.parse(FailureSafeParser.scala:60)

at org.apache.spark.sql.catalyst.csv.UnivocityParser$.$anonfun$parseIterator$2(UnivocityParser.scala:410)

at scala.collection.Iterator$$anon$11.nextCur(Iterator.scala:486)

at scala.collection.Iterator$$anon$11.hasNext(Iterator.scala:492)

at scala.collection.Iterator$$anon$10.hasNext(Iterator.scala:460)

at org.apache.spark.sql.execution.datasources.FileScanRDD$$anon$1.hasNext(FileScanRDD.scala:104)

at org.apache.spark.sql.execution.datasources.FileScanRDD$$anon$1.nextIterator(FileScanRDD.scala:191)

at org.apache.spark.sql.execution.datasources.FileScanRDD$$anon$1.hasNext(FileScanRDD.scala:104)

at scala.collection.Iterator$$anon$10.hasNext(Iterator.scala:460)

at org.apache.spark.sql.catalyst.expressions.GeneratedClass$GeneratedIteratorForCodegenStage1.processNext(Unknown Source)

at org.apache.spark.sql.execution.BufferedRowIterator.hasNext(BufferedRowIterator.java:43)

at org.apache.spark.sql.execution.WholeStageCodegenExec$$anon$1.hasNext(WholeStageCodegenExec.scala:759)

at org.apache.spark.sql.execution.SparkPlan.$anonfun$getByteArrayRdd$1(SparkPlan.scala:349)

at org.apache.spark.rdd.RDD.$anonfun$mapPartitionsInternal$2(RDD.scala:898)

at org.apache.spark.rdd.RDD.$anonfun$mapPartitionsInternal$2$adapted(RDD.scala:898)

at org.apache.spark.rdd.MapPartitionsRDD.compute(MapPartitionsRDD.scala:52)

at org.apache.spark.rdd.RDD.computeOrReadCheckpoint(RDD.scala:373)

at org.apache.spark.rdd.RDD.iterator(RDD.scala:337)

at org.apache.spark.scheduler.ResultTask.runTask(ResultTask.scala:90)

at org.apache.spark.scheduler.Task.run(Task.scala:131)

at org.apache.spark.executor.Executor$TaskRunner.$anonfun$run$3(Executor.scala:506)

at org.apache.spark.util.Utils$.tryWithSafeFinally(Utils.scala:1462)

at org.apache.spark.executor.Executor$TaskRunner.run(Executor.scala:509)

at java.util.concurrent.ThreadPoolExecutor.runWorker(ThreadPoolExecutor.java:1149)

at java.util.concurrent.ThreadPoolExecutor$Worker.run(ThreadPoolExecutor.java:624)

at java.lang.Thread.run(Thread.java:750)

Caused by: java.time.format.DateTimeParseException: Text '01-10-2019' could not be parsed at index 0

at java.time.format.DateTimeFormatter.parseResolved0(DateTimeFormatter.java:1949)

at java.time.format.DateTimeFormatter.parse(DateTimeFormatter.java:1777)

at org.apache.spark.sql.catalyst.util.Iso8601DateFormatter.parse(DateFormatter.scala:55)

... 32 more

Driver stacktrace:

at org.apache.spark.scheduler.DAGScheduler.failJobAndIndependentStages(DAGScheduler.scala:2454)

at org.apache.spark.scheduler.DAGScheduler.$anonfun$abortStage$2(DAGScheduler.scala:2403)

at org.apache.spark.scheduler.DAGScheduler.$anonfun$abortStage$2$adapted(DAGScheduler.scala:2402)

at scala.collection.mutable.ResizableArray.foreach(ResizableArray.scala:62)

at scala.collection.mutable.ResizableArray.foreach$(ResizableArray.scala:55)

at scala.collection.mutable.ArrayBuffer.foreach(ArrayBuffer.scala:49)

at org.apache.spark.scheduler.DAGScheduler.abortStage(DAGScheduler.scala:2402)

at org.apache.spark.scheduler.DAGScheduler.$anonfun$handleTaskSetFailed$1(DAGScheduler.scala:1160)

at org.apache.spark.scheduler.DAGScheduler.$anonfun$handleTaskSetFailed$1$adapted(DAGScheduler.scala:1160)

at scala.Option.foreach(Option.scala:407)

at org.apache.spark.scheduler.DAGScheduler.handleTaskSetFailed(DAGScheduler.scala:1160)

at org.apache.spark.scheduler.DAGSchedulerEventProcessLoop.doOnReceive(DAGScheduler.scala:2642)

at org.apache.spark.scheduler.DAGSchedulerEventProcessLoop.onReceive(DAGScheduler.scala:2584)

at org.apache.spark.scheduler.DAGSchedulerEventProcessLoop.onReceive(DAGScheduler.scala:2573)

at org.apache.spark.util.EventLoop$$anon$1.run(EventLoop.scala:49)

at org.apache.spark.scheduler.DAGScheduler.runJob(DAGScheduler.scala:938)

at org.apache.spark.SparkContext.runJob(SparkContext.scala:2214)

at org.apache.spark.SparkContext.runJob(SparkContext.scala:2235)

at org.apache.spark.SparkContext.runJob(SparkContext.scala:2254)

at org.apache.spark.sql.execution.SparkPlan.executeTake(SparkPlan.scala:476)

at org.apache.spark.sql.execution.SparkPlan.executeTake(SparkPlan.scala:429)

at org.apache.spark.sql.execution.CollectLimitExec.executeCollect(limit.scala:48)

at org.apache.spark.sql.Dataset.collectFromPlan(Dataset.scala:3715)

at org.apache.spark.sql.Dataset.$anonfun$head$1(Dataset.scala:2728)

at org.apache.spark.sql.Dataset.$anonfun$withAction$1(Dataset.scala:3706)

at org.apache.spark.sql.execution.SQLExecution$.$anonfun$withNewExecutionId$5(SQLExecution.scala:103)

at org.apache.spark.sql.execution.SQLExecution$.withSQLConfPropagated(SQLExecution.scala:163)

at org.apache.spark.sql.execution.SQLExecution$.$anonfun$withNewExecutionId$1(SQLExecution.scala:90)

at org.apache.spark.sql.SparkSession.withActive(SparkSession.scala:775)

at org.apache.spark.sql.execution.SQLExecution$.withNewExecutionId(SQLExecution.scala:64)

at org.apache.spark.sql.Dataset.withAction(Dataset.scala:3704)

at org.apache.spark.sql.Dataset.head(Dataset.scala:2728)

at org.apache.spark.sql.Dataset.take(Dataset.scala:2935)

at org.apache.spark.sql.Dataset.getRows(Dataset.scala:287)

at org.apache.spark.sql.Dataset.showString(Dataset.scala:326)

at sun.reflect.NativeMethodAccessorImpl.invoke0(Native Method)

at sun.reflect.NativeMethodAccessorImpl.invoke(NativeMethodAccessorImpl.java:62)

at sun.reflect.DelegatingMethodAccessorImpl.invoke(DelegatingMethodAccessorImpl.java:43)

at java.lang.reflect.Method.invoke(Method.java:498)

at py4j.reflection.MethodInvoker.invoke(MethodInvoker.java:244)

at py4j.reflection.ReflectionEngine.invoke(ReflectionEngine.java:357)

at py4j.Gateway.invoke(Gateway.java:282)

at py4j.commands.AbstractCommand.invokeMethod(AbstractCommand.java:132)

at py4j.commands.CallCommand.execute(CallCommand.java:79)

at py4j.ClientServerConnection.waitForCommands(ClientServerConnection.java:182)

at py4j.ClientServerConnection.run(ClientServerConnection.java:106)

at java.lang.Thread.run(Thread.java:750)

Caused by: org.apache.spark.SparkUpgradeException: You may get a different result due to the upgrading of Spark 3.0: Fail to parse '01-10-2019' in the new parser. You can set spark.sql.legacy.timeParserPolicy to LEGACY to restore the behavior before Spark 3.0, or set to CORRECTED and treat it as an invalid datetime string.

at org.apache.spark.sql.errors.QueryExecutionErrors$.failToParseDateTimeInNewParserError(QueryExecutionErrors.scala:918)

at org.apache.spark.sql.catalyst.util.DateTimeFormatterHelper$$anonfun$checkParsedDiff$1.applyOrElse(DateTimeFormatterHelper.scala:148)

at org.apache.spark.sql.catalyst.util.DateTimeFormatterHelper$$anonfun$checkParsedDiff$1.applyOrElse(DateTimeFormatterHelper.scala:141)

at scala.runtime.AbstractPartialFunction.apply(AbstractPartialFunction.scala:38)

at org.apache.spark.sql.catalyst.util.Iso8601DateFormatter.parse(DateFormatter.scala:57)

at org.apache.spark.sql.catalyst.csv.UnivocityParser.$anonfun$makeConverter$21(UnivocityParser.scala:202)

at org.apache.spark.sql.catalyst.csv.UnivocityParser.nullSafeDatum(UnivocityParser.scala:248)

at org.apache.spark.sql.catalyst.csv.UnivocityParser.$anonfun$makeConverter$20(UnivocityParser.scala:200)

at org.apache.spark.sql.catalyst.csv.UnivocityParser.org$apache$spark$sql$catalyst$csv$UnivocityParser$$convert(UnivocityParser.scala:301)

at org.apache.spark.sql.catalyst.csv.UnivocityParser.$anonfun$parse$2(UnivocityParser.scala:264)

at org.apache.spark.sql.catalyst.csv.UnivocityParser$.$anonfun$parseIterator$1(UnivocityParser.scala:406)

at org.apache.spark.sql.catalyst.util.FailureSafeParser.parse(FailureSafeParser.scala:60)

at org.apache.spark.sql.catalyst.csv.UnivocityParser$.$anonfun$parseIterator$2(UnivocityParser.scala:410)

at scala.collection.Iterator$$anon$11.nextCur(Iterator.scala:486)

at scala.collection.Iterator$$anon$11.hasNext(Iterator.scala:492)

at scala.collection.Iterator$$anon$10.hasNext(Iterator.scala:460)

at org.apache.spark.sql.execution.datasources.FileScanRDD$$anon$1.hasNext(FileScanRDD.scala:104)

at org.apache.spark.sql.execution.datasources.FileScanRDD$$anon$1.nextIterator(FileScanRDD.scala:191)

at org.apache.spark.sql.execution.datasources.FileScanRDD$$anon$1.hasNext(FileScanRDD.scala:104)

at scala.collection.Iterator$$anon$10.hasNext(Iterator.scala:460)

at org.apache.spark.sql.catalyst.expressions.GeneratedClass$GeneratedIteratorForCodegenStage1.processNext(Unknown Source)

at org.apache.spark.sql.execution.BufferedRowIterator.hasNext(BufferedRowIterator.java:43)

at org.apache.spark.sql.execution.WholeStageCodegenExec$$anon$1.hasNext(WholeStageCodegenExec.scala:759)

at org.apache.spark.sql.execution.SparkPlan.$anonfun$getByteArrayRdd$1(SparkPlan.scala:349)

at org.apache.spark.rdd.RDD.$anonfun$mapPartitionsInternal$2(RDD.scala:898)

at org.apache.spark.rdd.RDD.$anonfun$mapPartitionsInternal$2$adapted(RDD.scala:898)

at org.apache.spark.rdd.MapPartitionsRDD.compute(MapPartitionsRDD.scala:52)

at org.apache.spark.rdd.RDD.computeOrReadCheckpoint(RDD.scala:373)

at org.apache.spark.rdd.RDD.iterator(RDD.scala:337)

at org.apache.spark.scheduler.ResultTask.runTask(ResultTask.scala:90)

at org.apache.spark.scheduler.Task.run(Task.scala:131)

at org.apache.spark.executor.Executor$TaskRunner.$anonfun$run$3(Executor.scala:506)

at org.apache.spark.util.Utils$.tryWithSafeFinally(Utils.scala:1462)

at org.apache.spark.executor.Executor$TaskRunner.run(Executor.scala:509)

at java.util.concurrent.ThreadPoolExecutor.runWorker(ThreadPoolExecutor.java:1149)

at java.util.concurrent.ThreadPoolExecutor$Worker.run(ThreadPoolExecutor.java:624)

... 1 more

Caused by: java.time.format.DateTimeParseException: Text '01-10-2019' could not be parsed at index 0

at java.time.format.DateTimeFormatter.parseResolved0(DateTimeFormatter.java:1949)

at java.time.format.DateTimeFormatter.parse(DateTimeFormatter.java:1777)

at org.apache.spark.sql.catalyst.util.Iso8601DateFormatter.parse(DateFormatter.scala:55)

... 32 more

>>> custom\_schema=StructType ([StructField("IssuerId",IntegerType(),True) , StructField("IssuerId2",IntegerType(),True) , StructField("BusinessDate",DateType(),True) , StructField("StateCode",StringType(),True) , StructField("SourceName",StringType(),True) , StructField("NetworkName",StringType(),True) , StructField("NetworkURL",StringType(),True) , StructField("custnum",StringType(),True) , StructField("MarketCoverage",StringType(),True) , StructField("DentalOnlyPlan",StringType(),True)])

>>> data\_frame\_2.show(3)

24/07/18 20:10:04 WARN csv.CSVHeaderChecker: Number of column in CSV header is not equal to number of fields in the schema:

Header length: 10, schema size: 11

CSV file: hdfs://localhost:54310/user/hduser/Hackathon/insuranceinfo2.csv

24/07/18 20:10:04 ERROR executor.Executor: Exception in task 0.0 in stage 4.0 (TID 4)

org.apache.spark.SparkUpgradeException: You may get a different result due to the upgrading of Spark 3.0: Fail to parse '01-10-2019' in the new parser. You can set spark.sql.legacy.timeParserPolicy to LEGACY to restore the behavior before Spark 3.0, or set to CORRECTED and treat it as an invalid datetime string.

at org.apache.spark.sql.errors.QueryExecutionErrors$.failToParseDateTimeInNewParserError(QueryExecutionErrors.scala:918)

at org.apache.spark.sql.catalyst.util.DateTimeFormatterHelper$$anonfun$checkParsedDiff$1.applyOrElse(DateTimeFormatterHelper.scala:148)

at org.apache.spark.sql.catalyst.util.DateTimeFormatterHelper$$anonfun$checkParsedDiff$1.applyOrElse(DateTimeFormatterHelper.scala:141)

at scala.runtime.AbstractPartialFunction.apply(AbstractPartialFunction.scala:38)

at org.apache.spark.sql.catalyst.util.Iso8601DateFormatter.parse(DateFormatter.scala:57)

at org.apache.spark.sql.catalyst.csv.UnivocityParser.$anonfun$makeConverter$21(UnivocityParser.scala:202)

at org.apache.spark.sql.catalyst.csv.UnivocityParser.nullSafeDatum(UnivocityParser.scala:248)

at org.apache.spark.sql.catalyst.csv.UnivocityParser.$anonfun$makeConverter$20(UnivocityParser.scala:200)

at org.apache.spark.sql.catalyst.csv.UnivocityParser.org$apache$spark$sql$catalyst$csv$UnivocityParser$$convert(UnivocityParser.scala:301)

at org.apache.spark.sql.catalyst.csv.UnivocityParser.$anonfun$parse$2(UnivocityParser.scala:264)

at org.apache.spark.sql.catalyst.csv.UnivocityParser$.$anonfun$parseIterator$1(UnivocityParser.scala:406)

at org.apache.spark.sql.catalyst.util.FailureSafeParser.parse(FailureSafeParser.scala:60)

at org.apache.spark.sql.catalyst.csv.UnivocityParser$.$anonfun$parseIterator$2(UnivocityParser.scala:410)

at scala.collection.Iterator$$anon$11.nextCur(Iterator.scala:486)

at scala.collection.Iterator$$anon$11.hasNext(Iterator.scala:492)

at scala.collection.Iterator$$anon$10.hasNext(Iterator.scala:460)

at org.apache.spark.sql.execution.datasources.FileScanRDD$$anon$1.hasNext(FileScanRDD.scala:104)

at org.apache.spark.sql.execution.datasources.FileScanRDD$$anon$1.nextIterator(FileScanRDD.scala:191)

at org.apache.spark.sql.execution.datasources.FileScanRDD$$anon$1.hasNext(FileScanRDD.scala:104)

at scala.collection.Iterator$$anon$10.hasNext(Iterator.scala:460)

at org.apache.spark.sql.catalyst.expressions.GeneratedClass$GeneratedIteratorForCodegenStage1.processNext(Unknown Source)

at org.apache.spark.sql.execution.BufferedRowIterator.hasNext(BufferedRowIterator.java:43)

at org.apache.spark.sql.execution.WholeStageCodegenExec$$anon$1.hasNext(WholeStageCodegenExec.scala:759)

at org.apache.spark.sql.execution.SparkPlan.$anonfun$getByteArrayRdd$1(SparkPlan.scala:349)

at org.apache.spark.rdd.RDD.$anonfun$mapPartitionsInternal$2(RDD.scala:898)

at org.apache.spark.rdd.RDD.$anonfun$mapPartitionsInternal$2$adapted(RDD.scala:898)

at org.apache.spark.rdd.MapPartitionsRDD.compute(MapPartitionsRDD.scala:52)

at org.apache.spark.rdd.RDD.computeOrReadCheckpoint(RDD.scala:373)

at org.apache.spark.rdd.RDD.iterator(RDD.scala:337)

at org.apache.spark.scheduler.ResultTask.runTask(ResultTask.scala:90)

at org.apache.spark.scheduler.Task.run(Task.scala:131)

at org.apache.spark.executor.Executor$TaskRunner.$anonfun$run$3(Executor.scala:506)

at org.apache.spark.util.Utils$.tryWithSafeFinally(Utils.scala:1462)

at org.apache.spark.executor.Executor$TaskRunner.run(Executor.scala:509)

at java.util.concurrent.ThreadPoolExecutor.runWorker(ThreadPoolExecutor.java:1149)

at java.util.concurrent.ThreadPoolExecutor$Worker.run(ThreadPoolExecutor.java:624)

at java.lang.Thread.run(Thread.java:750)

Caused by: java.time.format.DateTimeParseException: Text '01-10-2019' could not be parsed at index 0

at java.time.format.DateTimeFormatter.parseResolved0(DateTimeFormatter.java:1949)

at java.time.format.DateTimeFormatter.parse(DateTimeFormatter.java:1777)

at org.apache.spark.sql.catalyst.util.Iso8601DateFormatter.parse(DateFormatter.scala:55)

... 32 more

24/07/18 20:10:04 WARN scheduler.TaskSetManager: Lost task 0.0 in stage 4.0 (TID 4) (192.168.112.129 executor driver): org.apache.spark.SparkUpgradeException: You may get a different result due to the upgrading of Spark 3.0: Fail to parse '01-10-2019' in the new parser. You can set spark.sql.legacy.timeParserPolicy to LEGACY to restore the behavior before Spark 3.0, or set to CORRECTED and treat it as an invalid datetime string.

at org.apache.spark.sql.errors.QueryExecutionErrors$.failToParseDateTimeInNewParserError(QueryExecutionErrors.scala:918)

at org.apache.spark.sql.catalyst.util.DateTimeFormatterHelper$$anonfun$checkParsedDiff$1.applyOrElse(DateTimeFormatterHelper.scala:148)

at org.apache.spark.sql.catalyst.util.DateTimeFormatterHelper$$anonfun$checkParsedDiff$1.applyOrElse(DateTimeFormatterHelper.scala:141)

at scala.runtime.AbstractPartialFunction.apply(AbstractPartialFunction.scala:38)

at org.apache.spark.sql.catalyst.util.Iso8601DateFormatter.parse(DateFormatter.scala:57)

at org.apache.spark.sql.catalyst.csv.UnivocityParser.$anonfun$makeConverter$21(UnivocityParser.scala:202)

at org.apache.spark.sql.catalyst.csv.UnivocityParser.nullSafeDatum(UnivocityParser.scala:248)

at org.apache.spark.sql.catalyst.csv.UnivocityParser.$anonfun$makeConverter$20(UnivocityParser.scala:200)

at org.apache.spark.sql.catalyst.csv.UnivocityParser.org$apache$spark$sql$catalyst$csv$UnivocityParser$$convert(UnivocityParser.scala:301)

at org.apache.spark.sql.catalyst.csv.UnivocityParser.$anonfun$parse$2(UnivocityParser.scala:264)

at org.apache.spark.sql.catalyst.csv.UnivocityParser$.$anonfun$parseIterator$1(UnivocityParser.scala:406)

at org.apache.spark.sql.catalyst.util.FailureSafeParser.parse(FailureSafeParser.scala:60)

at org.apache.spark.sql.catalyst.csv.UnivocityParser$.$anonfun$parseIterator$2(UnivocityParser.scala:410)

at scala.collection.Iterator$$anon$11.nextCur(Iterator.scala:486)

at scala.collection.Iterator$$anon$11.hasNext(Iterator.scala:492)

at scala.collection.Iterator$$anon$10.hasNext(Iterator.scala:460)

at org.apache.spark.sql.execution.datasources.FileScanRDD$$anon$1.hasNext(FileScanRDD.scala:104)

at org.apache.spark.sql.execution.datasources.FileScanRDD$$anon$1.nextIterator(FileScanRDD.scala:191)

at org.apache.spark.sql.execution.datasources.FileScanRDD$$anon$1.hasNext(FileScanRDD.scala:104)

at scala.collection.Iterator$$anon$10.hasNext(Iterator.scala:460)

at org.apache.spark.sql.catalyst.expressions.GeneratedClass$GeneratedIteratorForCodegenStage1.processNext(Unknown Source)

at org.apache.spark.sql.execution.BufferedRowIterator.hasNext(BufferedRowIterator.java:43)

at org.apache.spark.sql.execution.WholeStageCodegenExec$$anon$1.hasNext(WholeStageCodegenExec.scala:759)

at org.apache.spark.sql.execution.SparkPlan.$anonfun$getByteArrayRdd$1(SparkPlan.scala:349)

at org.apache.spark.rdd.RDD.$anonfun$mapPartitionsInternal$2(RDD.scala:898)

at org.apache.spark.rdd.RDD.$anonfun$mapPartitionsInternal$2$adapted(RDD.scala:898)

at org.apache.spark.rdd.MapPartitionsRDD.compute(MapPartitionsRDD.scala:52)

at org.apache.spark.rdd.RDD.computeOrReadCheckpoint(RDD.scala:373)

at org.apache.spark.rdd.RDD.iterator(RDD.scala:337)

at org.apache.spark.scheduler.ResultTask.runTask(ResultTask.scala:90)

at org.apache.spark.scheduler.Task.run(Task.scala:131)

at org.apache.spark.executor.Executor$TaskRunner.$anonfun$run$3(Executor.scala:506)

at org.apache.spark.util.Utils$.tryWithSafeFinally(Utils.scala:1462)

at org.apache.spark.executor.Executor$TaskRunner.run(Executor.scala:509)

at java.util.concurrent.ThreadPoolExecutor.runWorker(ThreadPoolExecutor.java:1149)

at java.util.concurrent.ThreadPoolExecutor$Worker.run(ThreadPoolExecutor.java:624)

at java.lang.Thread.run(Thread.java:750)

Caused by: java.time.format.DateTimeParseException: Text '01-10-2019' could not be parsed at index 0

at java.time.format.DateTimeFormatter.parseResolved0(DateTimeFormatter.java:1949)

at java.time.format.DateTimeFormatter.parse(DateTimeFormatter.java:1777)

at org.apache.spark.sql.catalyst.util.Iso8601DateFormatter.parse(DateFormatter.scala:55)

... 32 more

24/07/18 20:10:04 ERROR scheduler.TaskSetManager: Task 0 in stage 4.0 failed 1 times; aborting job

Traceback (most recent call last):

File "<stdin>", line 1, in <module>

File "/usr/local/spark/python/pyspark/sql/dataframe.py", line 494, in show

print(self.\_jdf.showString(n, 20, vertical))

File "/usr/local/spark/python/lib/py4j-0.10.9.3-src.zip/py4j/java\_gateway.py", line 1322, in \_\_call\_\_

File "/usr/local/spark/python/pyspark/sql/utils.py", line 111, in deco

return f(\*a, \*\*kw)

File "/usr/local/spark/python/lib/py4j-0.10.9.3-src.zip/py4j/protocol.py", line 328, in get\_return\_value

py4j.protocol.Py4JJavaError: An error occurred while calling o135.showString.

: org.apache.spark.SparkException: Job aborted due to stage failure: Task 0 in stage 4.0 failed 1 times, most recent failure: Lost task 0.0 in stage 4.0 (TID 4) (192.168.112.129 executor driver): org.apache.spark.SparkUpgradeException: You may get a different result due to the upgrading of Spark 3.0: Fail to parse '01-10-2019' in the new parser. You can set spark.sql.legacy.timeParserPolicy to LEGACY to restore the behavior before Spark 3.0, or set to CORRECTED and treat it as an invalid datetime string.

at org.apache.spark.sql.errors.QueryExecutionErrors$.failToParseDateTimeInNewParserError(QueryExecutionErrors.scala:918)

at org.apache.spark.sql.catalyst.util.DateTimeFormatterHelper$$anonfun$checkParsedDiff$1.applyOrElse(DateTimeFormatterHelper.scala:148)

at org.apache.spark.sql.catalyst.util.DateTimeFormatterHelper$$anonfun$checkParsedDiff$1.applyOrElse(DateTimeFormatterHelper.scala:141)

at scala.runtime.AbstractPartialFunction.apply(AbstractPartialFunction.scala:38)

at org.apache.spark.sql.catalyst.util.Iso8601DateFormatter.parse(DateFormatter.scala:57)

at org.apache.spark.sql.catalyst.csv.UnivocityParser.$anonfun$makeConverter$21(UnivocityParser.scala:202)

at org.apache.spark.sql.catalyst.csv.UnivocityParser.nullSafeDatum(UnivocityParser.scala:248)

at org.apache.spark.sql.catalyst.csv.UnivocityParser.$anonfun$makeConverter$20(UnivocityParser.scala:200)

at org.apache.spark.sql.catalyst.csv.UnivocityParser.org$apache$spark$sql$catalyst$csv$UnivocityParser$$convert(UnivocityParser.scala:301)

at org.apache.spark.sql.catalyst.csv.UnivocityParser.$anonfun$parse$2(UnivocityParser.scala:264)

at org.apache.spark.sql.catalyst.csv.UnivocityParser$.$anonfun$parseIterator$1(UnivocityParser.scala:406)

at org.apache.spark.sql.catalyst.util.FailureSafeParser.parse(FailureSafeParser.scala:60)

at org.apache.spark.sql.catalyst.csv.UnivocityParser$.$anonfun$parseIterator$2(UnivocityParser.scala:410)

at scala.collection.Iterator$$anon$11.nextCur(Iterator.scala:486)

at scala.collection.Iterator$$anon$11.hasNext(Iterator.scala:492)

at scala.collection.Iterator$$anon$10.hasNext(Iterator.scala:460)

at org.apache.spark.sql.execution.datasources.FileScanRDD$$anon$1.hasNext(FileScanRDD.scala:104)

at org.apache.spark.sql.execution.datasources.FileScanRDD$$anon$1.nextIterator(FileScanRDD.scala:191)

at org.apache.spark.sql.execution.datasources.FileScanRDD$$anon$1.hasNext(FileScanRDD.scala:104)

at scala.collection.Iterator$$anon$10.hasNext(Iterator.scala:460)

at org.apache.spark.sql.catalyst.expressions.GeneratedClass$GeneratedIteratorForCodegenStage1.processNext(Unknown Source)

at org.apache.spark.sql.execution.BufferedRowIterator.hasNext(BufferedRowIterator.java:43)

at org.apache.spark.sql.execution.WholeStageCodegenExec$$anon$1.hasNext(WholeStageCodegenExec.scala:759)

at org.apache.spark.sql.execution.SparkPlan.$anonfun$getByteArrayRdd$1(SparkPlan.scala:349)

at org.apache.spark.rdd.RDD.$anonfun$mapPartitionsInternal$2(RDD.scala:898)

at org.apache.spark.rdd.RDD.$anonfun$mapPartitionsInternal$2$adapted(RDD.scala:898)

at org.apache.spark.rdd.MapPartitionsRDD.compute(MapPartitionsRDD.scala:52)

at org.apache.spark.rdd.RDD.computeOrReadCheckpoint(RDD.scala:373)

at org.apache.spark.rdd.RDD.iterator(RDD.scala:337)

at org.apache.spark.scheduler.ResultTask.runTask(ResultTask.scala:90)

at org.apache.spark.scheduler.Task.run(Task.scala:131)

at org.apache.spark.executor.Executor$TaskRunner.$anonfun$run$3(Executor.scala:506)

at org.apache.spark.util.Utils$.tryWithSafeFinally(Utils.scala:1462)

at org.apache.spark.executor.Executor$TaskRunner.run(Executor.scala:509)

at java.util.concurrent.ThreadPoolExecutor.runWorker(ThreadPoolExecutor.java:1149)

at java.util.concurrent.ThreadPoolExecutor$Worker.run(ThreadPoolExecutor.java:624)

at java.lang.Thread.run(Thread.java:750)

Caused by: java.time.format.DateTimeParseException: Text '01-10-2019' could not be parsed at index 0

at java.time.format.DateTimeFormatter.parseResolved0(DateTimeFormatter.java:1949)

at java.time.format.DateTimeFormatter.parse(DateTimeFormatter.java:1777)

at org.apache.spark.sql.catalyst.util.Iso8601DateFormatter.parse(DateFormatter.scala:55)

... 32 more

Driver stacktrace:

at org.apache.spark.scheduler.DAGScheduler.failJobAndIndependentStages(DAGScheduler.scala:2454)

at org.apache.spark.scheduler.DAGScheduler.$anonfun$abortStage$2(DAGScheduler.scala:2403)

at org.apache.spark.scheduler.DAGScheduler.$anonfun$abortStage$2$adapted(DAGScheduler.scala:2402)

at scala.collection.mutable.ResizableArray.foreach(ResizableArray.scala:62)

at scala.collection.mutable.ResizableArray.foreach$(ResizableArray.scala:55)

at scala.collection.mutable.ArrayBuffer.foreach(ArrayBuffer.scala:49)

at org.apache.spark.scheduler.DAGScheduler.abortStage(DAGScheduler.scala:2402)

at org.apache.spark.scheduler.DAGScheduler.$anonfun$handleTaskSetFailed$1(DAGScheduler.scala:1160)

at org.apache.spark.scheduler.DAGScheduler.$anonfun$handleTaskSetFailed$1$adapted(DAGScheduler.scala:1160)

at scala.Option.foreach(Option.scala:407)

at org.apache.spark.scheduler.DAGScheduler.handleTaskSetFailed(DAGScheduler.scala:1160)

at org.apache.spark.scheduler.DAGSchedulerEventProcessLoop.doOnReceive(DAGScheduler.scala:2642)

at org.apache.spark.scheduler.DAGSchedulerEventProcessLoop.onReceive(DAGScheduler.scala:2584)

at org.apache.spark.scheduler.DAGSchedulerEventProcessLoop.onReceive(DAGScheduler.scala:2573)

at org.apache.spark.util.EventLoop$$anon$1.run(EventLoop.scala:49)

at org.apache.spark.scheduler.DAGScheduler.runJob(DAGScheduler.scala:938)

at org.apache.spark.SparkContext.runJob(SparkContext.scala:2214)

at org.apache.spark.SparkContext.runJob(SparkContext.scala:2235)

at org.apache.spark.SparkContext.runJob(SparkContext.scala:2254)

at org.apache.spark.sql.execution.SparkPlan.executeTake(SparkPlan.scala:476)

at org.apache.spark.sql.execution.SparkPlan.executeTake(SparkPlan.scala:429)

at org.apache.spark.sql.execution.CollectLimitExec.executeCollect(limit.scala:48)

at org.apache.spark.sql.Dataset.collectFromPlan(Dataset.scala:3715)

at org.apache.spark.sql.Dataset.$anonfun$head$1(Dataset.scala:2728)

at org.apache.spark.sql.Dataset.$anonfun$withAction$1(Dataset.scala:3706)

at org.apache.spark.sql.execution.SQLExecution$.$anonfun$withNewExecutionId$5(SQLExecution.scala:103)

at org.apache.spark.sql.execution.SQLExecution$.withSQLConfPropagated(SQLExecution.scala:163)

at org.apache.spark.sql.execution.SQLExecution$.$anonfun$withNewExecutionId$1(SQLExecution.scala:90)

at org.apache.spark.sql.SparkSession.withActive(SparkSession.scala:775)

at org.apache.spark.sql.execution.SQLExecution$.withNewExecutionId(SQLExecution.scala:64)

at org.apache.spark.sql.Dataset.withAction(Dataset.scala:3704)

at org.apache.spark.sql.Dataset.head(Dataset.scala:2728)

at org.apache.spark.sql.Dataset.take(Dataset.scala:2935)

at org.apache.spark.sql.Dataset.getRows(Dataset.scala:287)

at org.apache.spark.sql.Dataset.showString(Dataset.scala:326)

at sun.reflect.NativeMethodAccessorImpl.invoke0(Native Method)

at sun.reflect.NativeMethodAccessorImpl.invoke(NativeMethodAccessorImpl.java:62)

at sun.reflect.DelegatingMethodAccessorImpl.invoke(DelegatingMethodAccessorImpl.java:43)

at java.lang.reflect.Method.invoke(Method.java:498)

at py4j.reflection.MethodInvoker.invoke(MethodInvoker.java:244)

at py4j.reflection.ReflectionEngine.invoke(ReflectionEngine.java:357)

at py4j.Gateway.invoke(Gateway.java:282)

at py4j.commands.AbstractCommand.invokeMethod(AbstractCommand.java:132)

at py4j.commands.CallCommand.execute(CallCommand.java:79)

at py4j.ClientServerConnection.waitForCommands(ClientServerConnection.java:182)

at py4j.ClientServerConnection.run(ClientServerConnection.java:106)

at java.lang.Thread.run(Thread.java:750)

Caused by: org.apache.spark.SparkUpgradeException: You may get a different result due to the upgrading of Spark 3.0: Fail to parse '01-10-2019' in the new parser. You can set spark.sql.legacy.timeParserPolicy to LEGACY to restore the behavior before Spark 3.0, or set to CORRECTED and treat it as an invalid datetime string.

at org.apache.spark.sql.errors.QueryExecutionErrors$.failToParseDateTimeInNewParserError(QueryExecutionErrors.scala:918)

at org.apache.spark.sql.catalyst.util.DateTimeFormatterHelper$$anonfun$checkParsedDiff$1.applyOrElse(DateTimeFormatterHelper.scala:148)

at org.apache.spark.sql.catalyst.util.DateTimeFormatterHelper$$anonfun$checkParsedDiff$1.applyOrElse(DateTimeFormatterHelper.scala:141)

at scala.runtime.AbstractPartialFunction.apply(AbstractPartialFunction.scala:38)

at org.apache.spark.sql.catalyst.util.Iso8601DateFormatter.parse(DateFormatter.scala:57)

at org.apache.spark.sql.catalyst.csv.UnivocityParser.$anonfun$makeConverter$21(UnivocityParser.scala:202)

at org.apache.spark.sql.catalyst.csv.UnivocityParser.nullSafeDatum(UnivocityParser.scala:248)

at org.apache.spark.sql.catalyst.csv.UnivocityParser.$anonfun$makeConverter$20(UnivocityParser.scala:200)

at org.apache.spark.sql.catalyst.csv.UnivocityParser.org$apache$spark$sql$catalyst$csv$UnivocityParser$$convert(UnivocityParser.scala:301)

at org.apache.spark.sql.catalyst.csv.UnivocityParser.$anonfun$parse$2(UnivocityParser.scala:264)

at org.apache.spark.sql.catalyst.csv.UnivocityParser$.$anonfun$parseIterator$1(UnivocityParser.scala:406)

at org.apache.spark.sql.catalyst.util.FailureSafeParser.parse(FailureSafeParser.scala:60)

at org.apache.spark.sql.catalyst.csv.UnivocityParser$.$anonfun$parseIterator$2(UnivocityParser.scala:410)

at scala.collection.Iterator$$anon$11.nextCur(Iterator.scala:486)

at scala.collection.Iterator$$anon$11.hasNext(Iterator.scala:492)

at scala.collection.Iterator$$anon$10.hasNext(Iterator.scala:460)

at org.apache.spark.sql.execution.datasources.FileScanRDD$$anon$1.hasNext(FileScanRDD.scala:104)

at org.apache.spark.sql.execution.datasources.FileScanRDD$$anon$1.nextIterator(FileScanRDD.scala:191)

at org.apache.spark.sql.execution.datasources.FileScanRDD$$anon$1.hasNext(FileScanRDD.scala:104)

at scala.collection.Iterator$$anon$10.hasNext(Iterator.scala:460)

at org.apache.spark.sql.catalyst.expressions.GeneratedClass$GeneratedIteratorForCodegenStage1.processNext(Unknown Source)

at org.apache.spark.sql.execution.BufferedRowIterator.hasNext(BufferedRowIterator.java:43)

at org.apache.spark.sql.execution.WholeStageCodegenExec$$anon$1.hasNext(WholeStageCodegenExec.scala:759)

at org.apache.spark.sql.execution.SparkPlan.$anonfun$getByteArrayRdd$1(SparkPlan.scala:349)

at org.apache.spark.rdd.RDD.$anonfun$mapPartitionsInternal$2(RDD.scala:898)

at org.apache.spark.rdd.RDD.$anonfun$mapPartitionsInternal$2$adapted(RDD.scala:898)

at org.apache.spark.rdd.MapPartitionsRDD.compute(MapPartitionsRDD.scala:52)

at org.apache.spark.rdd.RDD.computeOrReadCheckpoint(RDD.scala:373)

at org.apache.spark.rdd.RDD.iterator(RDD.scala:337)

at org.apache.spark.scheduler.ResultTask.runTask(ResultTask.scala:90)

at org.apache.spark.scheduler.Task.run(Task.scala:131)

at org.apache.spark.executor.Executor$TaskRunner.$anonfun$run$3(Executor.scala:506)

at org.apache.spark.util.Utils$.tryWithSafeFinally(Utils.scala:1462)

at org.apache.spark.executor.Executor$TaskRunner.run(Executor.scala:509)

at java.util.concurrent.ThreadPoolExecutor.runWorker(ThreadPoolExecutor.java:1149)

at java.util.concurrent.ThreadPoolExecutor$Worker.run(ThreadPoolExecutor.java:624)

... 1 more

Caused by: java.time.format.DateTimeParseException: Text '01-10-2019' could not be parsed at index 0

at java.time.format.DateTimeFormatter.parseResolved0(DateTimeFormatter.java:1949)

at java.time.format.DateTimeFormatter.parse(DateTimeFormatter.java:1777)

at org.apache.spark.sql.catalyst.util.Iso8601DateFormatter.parse(DateFormatter.scala:55)

... 32 more

>>> custom\_schema=StructType ([StructField("IssuerId",IntegerType(),True) , StructField("IssuerId2",IntegerType(),True) , StructField("BusinessDate",DateType(),True) , StructField("StateCode",StringType(),True) , StructField("SourceName",StringType(),True) , StructField("NetworkName",StringType(),True) , StructField("NetworkURL",StringType(),True) , StructField("custnum",StringType(),True) , StructField("MarketCoverage",StringType(),True) , StructField("DentalOnlyPlan",StringType(),True)])

>>> data\_frame\_2=spark.read.csv(path="/user/hduser/Hackathon/insuranceinfo2.csv" , header=True , schema=custom\_schema )

>>> data\_frame\_2.show(3)

24/07/18 20:12:37 ERROR executor.Executor: Exception in task 0.0 in stage 5.0 (TID 5)

org.apache.spark.SparkUpgradeException: You may get a different result due to the upgrading of Spark 3.0: Fail to parse '01-10-2019' in the new parser. You can set spark.sql.legacy.timeParserPolicy to LEGACY to restore the behavior before Spark 3.0, or set to CORRECTED and treat it as an invalid datetime string.

at org.apache.spark.sql.errors.QueryExecutionErrors$.failToParseDateTimeInNewParserError(QueryExecutionErrors.scala:918)

at org.apache.spark.sql.catalyst.util.DateTimeFormatterHelper$$anonfun$checkParsedDiff$1.applyOrElse(DateTimeFormatterHelper.scala:148)

at org.apache.spark.sql.catalyst.util.DateTimeFormatterHelper$$anonfun$checkParsedDiff$1.applyOrElse(DateTimeFormatterHelper.scala:141)

at scala.runtime.AbstractPartialFunction.apply(AbstractPartialFunction.scala:38)

at org.apache.spark.sql.catalyst.util.Iso8601DateFormatter.parse(DateFormatter.scala:57)

at org.apache.spark.sql.catalyst.csv.UnivocityParser.$anonfun$makeConverter$21(UnivocityParser.scala:202)

at org.apache.spark.sql.catalyst.csv.UnivocityParser.nullSafeDatum(UnivocityParser.scala:248)

at org.apache.spark.sql.catalyst.csv.UnivocityParser.$anonfun$makeConverter$20(UnivocityParser.scala:200)

at org.apache.spark.sql.catalyst.csv.UnivocityParser.org$apache$spark$sql$catalyst$csv$UnivocityParser$$convert(UnivocityParser.scala:301)

at org.apache.spark.sql.catalyst.csv.UnivocityParser.$anonfun$parse$2(UnivocityParser.scala:264)

at org.apache.spark.sql.catalyst.csv.UnivocityParser$.$anonfun$parseIterator$1(UnivocityParser.scala:406)

at org.apache.spark.sql.catalyst.util.FailureSafeParser.parse(FailureSafeParser.scala:60)

at org.apache.spark.sql.catalyst.csv.UnivocityParser$.$anonfun$parseIterator$2(UnivocityParser.scala:410)

at scala.collection.Iterator$$anon$11.nextCur(Iterator.scala:486)

at scala.collection.Iterator$$anon$11.hasNext(Iterator.scala:492)

at scala.collection.Iterator$$anon$10.hasNext(Iterator.scala:460)

at org.apache.spark.sql.execution.datasources.FileScanRDD$$anon$1.hasNext(FileScanRDD.scala:104)

at org.apache.spark.sql.execution.datasources.FileScanRDD$$anon$1.nextIterator(FileScanRDD.scala:191)

at org.apache.spark.sql.execution.datasources.FileScanRDD$$anon$1.hasNext(FileScanRDD.scala:104)

at scala.collection.Iterator$$anon$10.hasNext(Iterator.scala:460)

at org.apache.spark.sql.catalyst.expressions.GeneratedClass$GeneratedIteratorForCodegenStage1.processNext(Unknown Source)

at org.apache.spark.sql.execution.BufferedRowIterator.hasNext(BufferedRowIterator.java:43)

at org.apache.spark.sql.execution.WholeStageCodegenExec$$anon$1.hasNext(WholeStageCodegenExec.scala:759)

at org.apache.spark.sql.execution.SparkPlan.$anonfun$getByteArrayRdd$1(SparkPlan.scala:349)

at org.apache.spark.rdd.RDD.$anonfun$mapPartitionsInternal$2(RDD.scala:898)

at org.apache.spark.rdd.RDD.$anonfun$mapPartitionsInternal$2$adapted(RDD.scala:898)

at org.apache.spark.rdd.MapPartitionsRDD.compute(MapPartitionsRDD.scala:52)

at org.apache.spark.rdd.RDD.computeOrReadCheckpoint(RDD.scala:373)

at org.apache.spark.rdd.RDD.iterator(RDD.scala:337)

at org.apache.spark.scheduler.ResultTask.runTask(ResultTask.scala:90)

at org.apache.spark.scheduler.Task.run(Task.scala:131)

at org.apache.spark.executor.Executor$TaskRunner.$anonfun$run$3(Executor.scala:506)

at org.apache.spark.util.Utils$.tryWithSafeFinally(Utils.scala:1462)

at org.apache.spark.executor.Executor$TaskRunner.run(Executor.scala:509)

at java.util.concurrent.ThreadPoolExecutor.runWorker(ThreadPoolExecutor.java:1149)

at java.util.concurrent.ThreadPoolExecutor$Worker.run(ThreadPoolExecutor.java:624)

at java.lang.Thread.run(Thread.java:750)

Caused by: java.time.format.DateTimeParseException: Text '01-10-2019' could not be parsed at index 0

at java.time.format.DateTimeFormatter.parseResolved0(DateTimeFormatter.java:1949)

at java.time.format.DateTimeFormatter.parse(DateTimeFormatter.java:1777)

at org.apache.spark.sql.catalyst.util.Iso8601DateFormatter.parse(DateFormatter.scala:55)

... 32 more

24/07/18 20:12:37 WARN scheduler.TaskSetManager: Lost task 0.0 in stage 5.0 (TID 5) (192.168.112.129 executor driver): org.apache.spark.SparkUpgradeException: You may get a different result due to the upgrading of Spark 3.0: Fail to parse '01-10-2019' in the new parser. You can set spark.sql.legacy.timeParserPolicy to LEGACY to restore the behavior before Spark 3.0, or set to CORRECTED and treat it as an invalid datetime string.

at org.apache.spark.sql.errors.QueryExecutionErrors$.failToParseDateTimeInNewParserError(QueryExecutionErrors.scala:918)

at org.apache.spark.sql.catalyst.util.DateTimeFormatterHelper$$anonfun$checkParsedDiff$1.applyOrElse(DateTimeFormatterHelper.scala:148)

at org.apache.spark.sql.catalyst.util.DateTimeFormatterHelper$$anonfun$checkParsedDiff$1.applyOrElse(DateTimeFormatterHelper.scala:141)

at scala.runtime.AbstractPartialFunction.apply(AbstractPartialFunction.scala:38)

at org.apache.spark.sql.catalyst.util.Iso8601DateFormatter.parse(DateFormatter.scala:57)

at org.apache.spark.sql.catalyst.csv.UnivocityParser.$anonfun$makeConverter$21(UnivocityParser.scala:202)

at org.apache.spark.sql.catalyst.csv.UnivocityParser.nullSafeDatum(UnivocityParser.scala:248)

at org.apache.spark.sql.catalyst.csv.UnivocityParser.$anonfun$makeConverter$20(UnivocityParser.scala:200)

at org.apache.spark.sql.catalyst.csv.UnivocityParser.org$apache$spark$sql$catalyst$csv$UnivocityParser$$convert(UnivocityParser.scala:301)

at org.apache.spark.sql.catalyst.csv.UnivocityParser.$anonfun$parse$2(UnivocityParser.scala:264)

at org.apache.spark.sql.catalyst.csv.UnivocityParser$.$anonfun$parseIterator$1(UnivocityParser.scala:406)

at org.apache.spark.sql.catalyst.util.FailureSafeParser.parse(FailureSafeParser.scala:60)

at org.apache.spark.sql.catalyst.csv.UnivocityParser$.$anonfun$parseIterator$2(UnivocityParser.scala:410)

at scala.collection.Iterator$$anon$11.nextCur(Iterator.scala:486)

at scala.collection.Iterator$$anon$11.hasNext(Iterator.scala:492)

at scala.collection.Iterator$$anon$10.hasNext(Iterator.scala:460)

at org.apache.spark.sql.execution.datasources.FileScanRDD$$anon$1.hasNext(FileScanRDD.scala:104)

at org.apache.spark.sql.execution.datasources.FileScanRDD$$anon$1.nextIterator(FileScanRDD.scala:191)

at org.apache.spark.sql.execution.datasources.FileScanRDD$$anon$1.hasNext(FileScanRDD.scala:104)

at scala.collection.Iterator$$anon$10.hasNext(Iterator.scala:460)

at org.apache.spark.sql.catalyst.expressions.GeneratedClass$GeneratedIteratorForCodegenStage1.processNext(Unknown Source)

at org.apache.spark.sql.execution.BufferedRowIterator.hasNext(BufferedRowIterator.java:43)

at org.apache.spark.sql.execution.WholeStageCodegenExec$$anon$1.hasNext(WholeStageCodegenExec.scala:759)

at org.apache.spark.sql.execution.SparkPlan.$anonfun$getByteArrayRdd$1(SparkPlan.scala:349)

at org.apache.spark.rdd.RDD.$anonfun$mapPartitionsInternal$2(RDD.scala:898)

at org.apache.spark.rdd.RDD.$anonfun$mapPartitionsInternal$2$adapted(RDD.scala:898)

at org.apache.spark.rdd.MapPartitionsRDD.compute(MapPartitionsRDD.scala:52)

at org.apache.spark.rdd.RDD.computeOrReadCheckpoint(RDD.scala:373)

at org.apache.spark.rdd.RDD.iterator(RDD.scala:337)

at org.apache.spark.scheduler.ResultTask.runTask(ResultTask.scala:90)

at org.apache.spark.scheduler.Task.run(Task.scala:131)

at org.apache.spark.executor.Executor$TaskRunner.$anonfun$run$3(Executor.scala:506)

at org.apache.spark.util.Utils$.tryWithSafeFinally(Utils.scala:1462)

at org.apache.spark.executor.Executor$TaskRunner.run(Executor.scala:509)

at java.util.concurrent.ThreadPoolExecutor.runWorker(ThreadPoolExecutor.java:1149)

at java.util.concurrent.ThreadPoolExecutor$Worker.run(ThreadPoolExecutor.java:624)

at java.lang.Thread.run(Thread.java:750)

Caused by: java.time.format.DateTimeParseException: Text '01-10-2019' could not be parsed at index 0

at java.time.format.DateTimeFormatter.parseResolved0(DateTimeFormatter.java:1949)

at java.time.format.DateTimeFormatter.parse(DateTimeFormatter.java:1777)

at org.apache.spark.sql.catalyst.util.Iso8601DateFormatter.parse(DateFormatter.scala:55)

... 32 more

24/07/18 20:12:37 ERROR scheduler.TaskSetManager: Task 0 in stage 5.0 failed 1 times; aborting job

Traceback (most recent call last):

File "<stdin>", line 1, in <module>

File "/usr/local/spark/python/pyspark/sql/dataframe.py", line 494, in show

print(self.\_jdf.showString(n, 20, vertical))

File "/usr/local/spark/python/lib/py4j-0.10.9.3-src.zip/py4j/java\_gateway.py", line 1322, in \_\_call\_\_

File "/usr/local/spark/python/pyspark/sql/utils.py", line 111, in deco

return f(\*a, \*\*kw)

File "/usr/local/spark/python/lib/py4j-0.10.9.3-src.zip/py4j/protocol.py", line 328, in get\_return\_value

py4j.protocol.Py4JJavaError: An error occurred while calling o154.showString.

: org.apache.spark.SparkException: Job aborted due to stage failure: Task 0 in stage 5.0 failed 1 times, most recent failure: Lost task 0.0 in stage 5.0 (TID 5) (192.168.112.129 executor driver): org.apache.spark.SparkUpgradeException: You may get a different result due to the upgrading of Spark 3.0: Fail to parse '01-10-2019' in the new parser. You can set spark.sql.legacy.timeParserPolicy to LEGACY to restore the behavior before Spark 3.0, or set to CORRECTED and treat it as an invalid datetime string.

at org.apache.spark.sql.errors.QueryExecutionErrors$.failToParseDateTimeInNewParserError(QueryExecutionErrors.scala:918)

at org.apache.spark.sql.catalyst.util.DateTimeFormatterHelper$$anonfun$checkParsedDiff$1.applyOrElse(DateTimeFormatterHelper.scala:148)

at org.apache.spark.sql.catalyst.util.DateTimeFormatterHelper$$anonfun$checkParsedDiff$1.applyOrElse(DateTimeFormatterHelper.scala:141)

at scala.runtime.AbstractPartialFunction.apply(AbstractPartialFunction.scala:38)

at org.apache.spark.sql.catalyst.util.Iso8601DateFormatter.parse(DateFormatter.scala:57)

at org.apache.spark.sql.catalyst.csv.UnivocityParser.$anonfun$makeConverter$21(UnivocityParser.scala:202)

at org.apache.spark.sql.catalyst.csv.UnivocityParser.nullSafeDatum(UnivocityParser.scala:248)

at org.apache.spark.sql.catalyst.csv.UnivocityParser.$anonfun$makeConverter$20(UnivocityParser.scala:200)

at org.apache.spark.sql.catalyst.csv.UnivocityParser.org$apache$spark$sql$catalyst$csv$UnivocityParser$$convert(UnivocityParser.scala:301)

at org.apache.spark.sql.catalyst.csv.UnivocityParser.$anonfun$parse$2(UnivocityParser.scala:264)

at org.apache.spark.sql.catalyst.csv.UnivocityParser$.$anonfun$parseIterator$1(UnivocityParser.scala:406)

at org.apache.spark.sql.catalyst.util.FailureSafeParser.parse(FailureSafeParser.scala:60)

at org.apache.spark.sql.catalyst.csv.UnivocityParser$.$anonfun$parseIterator$2(UnivocityParser.scala:410)

at scala.collection.Iterator$$anon$11.nextCur(Iterator.scala:486)

at scala.collection.Iterator$$anon$11.hasNext(Iterator.scala:492)

at scala.collection.Iterator$$anon$10.hasNext(Iterator.scala:460)

at org.apache.spark.sql.execution.datasources.FileScanRDD$$anon$1.hasNext(FileScanRDD.scala:104)

at org.apache.spark.sql.execution.datasources.FileScanRDD$$anon$1.nextIterator(FileScanRDD.scala:191)

at org.apache.spark.sql.execution.datasources.FileScanRDD$$anon$1.hasNext(FileScanRDD.scala:104)

at scala.collection.Iterator$$anon$10.hasNext(Iterator.scala:460)

at org.apache.spark.sql.catalyst.expressions.GeneratedClass$GeneratedIteratorForCodegenStage1.processNext(Unknown Source)

at org.apache.spark.sql.execution.BufferedRowIterator.hasNext(BufferedRowIterator.java:43)

at org.apache.spark.sql.execution.WholeStageCodegenExec$$anon$1.hasNext(WholeStageCodegenExec.scala:759)

at org.apache.spark.sql.execution.SparkPlan.$anonfun$getByteArrayRdd$1(SparkPlan.scala:349)

at org.apache.spark.rdd.RDD.$anonfun$mapPartitionsInternal$2(RDD.scala:898)

at org.apache.spark.rdd.RDD.$anonfun$mapPartitionsInternal$2$adapted(RDD.scala:898)

at org.apache.spark.rdd.MapPartitionsRDD.compute(MapPartitionsRDD.scala:52)

at org.apache.spark.rdd.RDD.computeOrReadCheckpoint(RDD.scala:373)

at org.apache.spark.rdd.RDD.iterator(RDD.scala:337)

at org.apache.spark.scheduler.ResultTask.runTask(ResultTask.scala:90)

at org.apache.spark.scheduler.Task.run(Task.scala:131)

at org.apache.spark.executor.Executor$TaskRunner.$anonfun$run$3(Executor.scala:506)

at org.apache.spark.util.Utils$.tryWithSafeFinally(Utils.scala:1462)

at org.apache.spark.executor.Executor$TaskRunner.run(Executor.scala:509)

at java.util.concurrent.ThreadPoolExecutor.runWorker(ThreadPoolExecutor.java:1149)

at java.util.concurrent.ThreadPoolExecutor$Worker.run(ThreadPoolExecutor.java:624)

at java.lang.Thread.run(Thread.java:750)

Caused by: java.time.format.DateTimeParseException: Text '01-10-2019' could not be parsed at index 0

at java.time.format.DateTimeFormatter.parseResolved0(DateTimeFormatter.java:1949)

at java.time.format.DateTimeFormatter.parse(DateTimeFormatter.java:1777)

at org.apache.spark.sql.catalyst.util.Iso8601DateFormatter.parse(DateFormatter.scala:55)

... 32 more

Driver stacktrace:

at org.apache.spark.scheduler.DAGScheduler.failJobAndIndependentStages(DAGScheduler.scala:2454)

at org.apache.spark.scheduler.DAGScheduler.$anonfun$abortStage$2(DAGScheduler.scala:2403)

at org.apache.spark.scheduler.DAGScheduler.$anonfun$abortStage$2$adapted(DAGScheduler.scala:2402)

at scala.collection.mutable.ResizableArray.foreach(ResizableArray.scala:62)

at scala.collection.mutable.ResizableArray.foreach$(ResizableArray.scala:55)

at scala.collection.mutable.ArrayBuffer.foreach(ArrayBuffer.scala:49)

at org.apache.spark.scheduler.DAGScheduler.abortStage(DAGScheduler.scala:2402)

at org.apache.spark.scheduler.DAGScheduler.$anonfun$handleTaskSetFailed$1(DAGScheduler.scala:1160)

at org.apache.spark.scheduler.DAGScheduler.$anonfun$handleTaskSetFailed$1$adapted(DAGScheduler.scala:1160)

at scala.Option.foreach(Option.scala:407)

at org.apache.spark.scheduler.DAGScheduler.handleTaskSetFailed(DAGScheduler.scala:1160)

at org.apache.spark.scheduler.DAGSchedulerEventProcessLoop.doOnReceive(DAGScheduler.scala:2642)

at org.apache.spark.scheduler.DAGSchedulerEventProcessLoop.onReceive(DAGScheduler.scala:2584)

at org.apache.spark.scheduler.DAGSchedulerEventProcessLoop.onReceive(DAGScheduler.scala:2573)

at org.apache.spark.util.EventLoop$$anon$1.run(EventLoop.scala:49)

at org.apache.spark.scheduler.DAGScheduler.runJob(DAGScheduler.scala:938)

at org.apache.spark.SparkContext.runJob(SparkContext.scala:2214)

at org.apache.spark.SparkContext.runJob(SparkContext.scala:2235)

at org.apache.spark.SparkContext.runJob(SparkContext.scala:2254)

at org.apache.spark.sql.execution.SparkPlan.executeTake(SparkPlan.scala:476)

at org.apache.spark.sql.execution.SparkPlan.executeTake(SparkPlan.scala:429)

at org.apache.spark.sql.execution.CollectLimitExec.executeCollect(limit.scala:48)

at org.apache.spark.sql.Dataset.collectFromPlan(Dataset.scala:3715)

at org.apache.spark.sql.Dataset.$anonfun$head$1(Dataset.scala:2728)

at org.apache.spark.sql.Dataset.$anonfun$withAction$1(Dataset.scala:3706)

at org.apache.spark.sql.execution.SQLExecution$.$anonfun$withNewExecutionId$5(SQLExecution.scala:103)

at org.apache.spark.sql.execution.SQLExecution$.withSQLConfPropagated(SQLExecution.scala:163)

at org.apache.spark.sql.execution.SQLExecution$.$anonfun$withNewExecutionId$1(SQLExecution.scala:90)

at org.apache.spark.sql.SparkSession.withActive(SparkSession.scala:775)

at org.apache.spark.sql.execution.SQLExecution$.withNewExecutionId(SQLExecution.scala:64)

at org.apache.spark.sql.Dataset.withAction(Dataset.scala:3704)

at org.apache.spark.sql.Dataset.head(Dataset.scala:2728)

at org.apache.spark.sql.Dataset.take(Dataset.scala:2935)

at org.apache.spark.sql.Dataset.getRows(Dataset.scala:287)

at org.apache.spark.sql.Dataset.showString(Dataset.scala:326)

at sun.reflect.NativeMethodAccessorImpl.invoke0(Native Method)

at sun.reflect.NativeMethodAccessorImpl.invoke(NativeMethodAccessorImpl.java:62)

at sun.reflect.DelegatingMethodAccessorImpl.invoke(DelegatingMethodAccessorImpl.java:43)

at java.lang.reflect.Method.invoke(Method.java:498)

at py4j.reflection.MethodInvoker.invoke(MethodInvoker.java:244)

at py4j.reflection.ReflectionEngine.invoke(ReflectionEngine.java:357)

at py4j.Gateway.invoke(Gateway.java:282)

at py4j.commands.AbstractCommand.invokeMethod(AbstractCommand.java:132)

at py4j.commands.CallCommand.execute(CallCommand.java:79)

at py4j.ClientServerConnection.waitForCommands(ClientServerConnection.java:182)

at py4j.ClientServerConnection.run(ClientServerConnection.java:106)

at java.lang.Thread.run(Thread.java:750)

Caused by: org.apache.spark.SparkUpgradeException: You may get a different result due to the upgrading of Spark 3.0: Fail to parse '01-10-2019' in the new parser. You can set spark.sql.legacy.timeParserPolicy to LEGACY to restore the behavior before Spark 3.0, or set to CORRECTED and treat it as an invalid datetime string.

at org.apache.spark.sql.errors.QueryExecutionErrors$.failToParseDateTimeInNewParserError(QueryExecutionErrors.scala:918)

at org.apache.spark.sql.catalyst.util.DateTimeFormatterHelper$$anonfun$checkParsedDiff$1.applyOrElse(DateTimeFormatterHelper.scala:148)

at org.apache.spark.sql.catalyst.util.DateTimeFormatterHelper$$anonfun$checkParsedDiff$1.applyOrElse(DateTimeFormatterHelper.scala:141)

at scala.runtime.AbstractPartialFunction.apply(AbstractPartialFunction.scala:38)

at org.apache.spark.sql.catalyst.util.Iso8601DateFormatter.parse(DateFormatter.scala:57)

at org.apache.spark.sql.catalyst.csv.UnivocityParser.$anonfun$makeConverter$21(UnivocityParser.scala:202)

at org.apache.spark.sql.catalyst.csv.UnivocityParser.nullSafeDatum(UnivocityParser.scala:248)

at org.apache.spark.sql.catalyst.csv.UnivocityParser.$anonfun$makeConverter$20(UnivocityParser.scala:200)

at org.apache.spark.sql.catalyst.csv.UnivocityParser.org$apache$spark$sql$catalyst$csv$UnivocityParser$$convert(UnivocityParser.scala:301)

at org.apache.spark.sql.catalyst.csv.UnivocityParser.$anonfun$parse$2(UnivocityParser.scala:264)

at org.apache.spark.sql.catalyst.csv.UnivocityParser$.$anonfun$parseIterator$1(UnivocityParser.scala:406)

at org.apache.spark.sql.catalyst.util.FailureSafeParser.parse(FailureSafeParser.scala:60)

at org.apache.spark.sql.catalyst.csv.UnivocityParser$.$anonfun$parseIterator$2(UnivocityParser.scala:410)

at scala.collection.Iterator$$anon$11.nextCur(Iterator.scala:486)

at scala.collection.Iterator$$anon$11.hasNext(Iterator.scala:492)

at scala.collection.Iterator$$anon$10.hasNext(Iterator.scala:460)

at org.apache.spark.sql.execution.datasources.FileScanRDD$$anon$1.hasNext(FileScanRDD.scala:104)

at org.apache.spark.sql.execution.datasources.FileScanRDD$$anon$1.nextIterator(FileScanRDD.scala:191)

at org.apache.spark.sql.execution.datasources.FileScanRDD$$anon$1.hasNext(FileScanRDD.scala:104)

at scala.collection.Iterator$$anon$10.hasNext(Iterator.scala:460)

at org.apache.spark.sql.catalyst.expressions.GeneratedClass$GeneratedIteratorForCodegenStage1.processNext(Unknown Source)

at org.apache.spark.sql.execution.BufferedRowIterator.hasNext(BufferedRowIterator.java:43)

at org.apache.spark.sql.execution.WholeStageCodegenExec$$anon$1.hasNext(WholeStageCodegenExec.scala:759)

at org.apache.spark.sql.execution.SparkPlan.$anonfun$getByteArrayRdd$1(SparkPlan.scala:349)

at org.apache.spark.rdd.RDD.$anonfun$mapPartitionsInternal$2(RDD.scala:898)

at org.apache.spark.rdd.RDD.$anonfun$mapPartitionsInternal$2$adapted(RDD.scala:898)

at org.apache.spark.rdd.MapPartitionsRDD.compute(MapPartitionsRDD.scala:52)

at org.apache.spark.rdd.RDD.computeOrReadCheckpoint(RDD.scala:373)

at org.apache.spark.rdd.RDD.iterator(RDD.scala:337)

at org.apache.spark.scheduler.ResultTask.runTask(ResultTask.scala:90)

at org.apache.spark.scheduler.Task.run(Task.scala:131)

at org.apache.spark.executor.Executor$TaskRunner.$anonfun$run$3(Executor.scala:506)

at org.apache.spark.util.Utils$.tryWithSafeFinally(Utils.scala:1462)

at org.apache.spark.executor.Executor$TaskRunner.run(Executor.scala:509)

at java.util.concurrent.ThreadPoolExecutor.runWorker(ThreadPoolExecutor.java:1149)

at java.util.concurrent.ThreadPoolExecutor$Worker.run(ThreadPoolExecutor.java:624)

... 1 more

Caused by: java.time.format.DateTimeParseException: Text '01-10-2019' could not be parsed at index 0

at java.time.format.DateTimeFormatter.parseResolved0(DateTimeFormatter.java:1949)

at java.time.format.DateTimeFormatter.parse(DateTimeFormatter.java:1777)

at org.apache.spark.sql.catalyst.util.Iso8601DateFormatter.parse(DateFormatter.scala:55)

... 32 more

>>> data\_frame\_2=spark.read.csv(path="/user/hduser/Hackathon/insuranceinfo2.csv" , header=True , schema=custom\_schema ,dateFormat="MM-dd-yyyy")

>>> data\_frame\_2.show(3)

+--------+---------+------------+---------+----------+--------------------+--------------------+-------+--------------+--------------+

|IssuerId|IssuerId2|BusinessDate|StateCode|SourceName| NetworkName| NetworkURL|custnum|MarketCoverage|DentalOnlyPlan|

+--------+---------+------------+---------+----------+--------------------+--------------------+-------+--------------+--------------+

| 31195| 31195| 2019-01-10| SD| SERFF|Sanford Health Pl...|https://www.viiad...| 14| null| null|

| 60536| 60536| 2019-01-10| SD| SERFF| Avera Health Plans|www.Avera.org/Pro...| 13| null| null|

| 61214| 61214| 2019-01-10| SD| SERFF| DenteMax|http://www2.dente...| 14| null| null|

+--------+---------+------------+---------+----------+--------------------+--------------------+-------+--------------+--------------+

only showing top 3 rows

>>> custom\_schema=StructType ([StructField("IssuerId",IntegerType(),True) , StructField("IssuerId2",IntegerType(),True) , StructField("BusinessDate",DateType(),True) , StructField("StateCode",StringType(),True) , StructField("SourceName",StringType(),True) , StructField("NetworkName",StringType(),True) , StructField("NetworkURL",StringType(),True) , StructField("custnum",StringType(),True) , StructField("MarketCoverage",StringType(),True) , StructField("DentalOnlyPlan",StringType(),True) , StructField("RejectRows",StringType(),True) ])

>>> data\_frame\_3=spark.read.csv(path="/user/hduser/Hackathon/insuranceinfo2.csv" , header=True , schema=custom\_schema , columnNameOfCorruptRecord="RejectRows" ,dateFormat="MM-dd-yyyy")

>>> data\_frame\_3.show(3)

+--------+---------+------------+---------+----------+--------------------+--------------------+-------+--------------+--------------+----------+

|IssuerId|IssuerId2|BusinessDate|StateCode|SourceName| NetworkName| NetworkURL|custnum|MarketCoverage|DentalOnlyPlan|RejectRows|

+--------+---------+------------+---------+----------+--------------------+--------------------+-------+--------------+--------------+----------+

| 31195| 31195| 2019-01-10| SD| SERFF|Sanford Health Pl...|https://www.viiad...| 14| null| null| null|

| 60536| 60536| 2019-01-10| SD| SERFF| Avera Health Plans|www.Avera.org/Pro...| 13| null| null| null|

| 61214| 61214| 2019-01-10| SD| SERFF| DenteMax|http://www2.dente...| 14| null| null| null|

+--------+---------+------------+---------+----------+--------------------+--------------------+-------+--------------+--------------+----------+

only showing top 3 rows

>>> data\_frame\_3.where("RejectRows is not NULL ").show(3)

24/07/18 20:18:04 ERROR executor.Executor: Exception in task 0.0 in stage 8.0 (TID 8)

java.lang.IllegalArgumentException: RejectRows does not exist. Available: IssuerId, IssuerId2, BusinessDate, StateCode, SourceName, NetworkName, NetworkURL, custnum, MarketCoverage, DentalOnlyPlan

at org.apache.spark.sql.types.StructType.$anonfun$fieldIndex$1(StructType.scala:306)

at scala.collection.immutable.HashMap$HashTrieMap.getOrElse0(HashMap.scala:596)

at scala.collection.immutable.HashMap.getOrElse(HashMap.scala:73)

at org.apache.spark.sql.types.StructType.fieldIndex(StructType.scala:305)

at org.apache.spark.sql.catalyst.OrderedFilters.$anonfun$predicates$4(OrderedFilters.scala:61)

at org.apache.spark.sql.catalyst.OrderedFilters.$anonfun$predicates$4$adapted(OrderedFilters.scala:61)

at scala.collection.TraversableLike.$anonfun$map$1(TraversableLike.scala:286)

at scala.collection.IndexedSeqOptimized.foreach(IndexedSeqOptimized.scala:36)

at scala.collection.IndexedSeqOptimized.foreach$(IndexedSeqOptimized.scala:33)

at scala.collection.mutable.ArrayOps$ofRef.foreach(ArrayOps.scala:198)

at scala.collection.TraversableLike.map(TraversableLike.scala:286)

at scala.collection.TraversableLike.map$(TraversableLike.scala:279)

at scala.collection.mutable.ArrayOps$ofRef.map(ArrayOps.scala:198)

at org.apache.spark.sql.catalyst.OrderedFilters.$anonfun$predicates$3(OrderedFilters.scala:61)

at org.apache.spark.sql.catalyst.OrderedFilters.$anonfun$predicates$3$adapted(OrderedFilters.scala:50)

at scala.collection.immutable.List.foreach(List.scala:431)

at org.apache.spark.sql.catalyst.OrderedFilters.<init>(OrderedFilters.scala:50)

at org.apache.spark.sql.catalyst.csv.UnivocityParser.<init>(UnivocityParser.scala:103)

at org.apache.spark.sql.execution.datasources.csv.CSVFileFormat.$anonfun$buildReader$1(CSVFileFormat.scala:130)

at org.apache.spark.sql.execution.datasources.FileFormat$$anon$1.apply(FileFormat.scala:148)

at org.apache.spark.sql.execution.datasources.FileFormat$$anon$1.apply(FileFormat.scala:133)

at org.apache.spark.sql.execution.datasources.FileScanRDD$$anon$1.org$apache$spark$sql$execution$datasources$FileScanRDD$$anon$$readCurrentFile(FileScanRDD.scala:127)

at org.apache.spark.sql.execution.datasources.FileScanRDD$$anon$1.nextIterator(FileScanRDD.scala:187)

at org.apache.spark.sql.execution.datasources.FileScanRDD$$anon$1.hasNext(FileScanRDD.scala:104)

at scala.collection.Iterator$$anon$10.hasNext(Iterator.scala:460)

at org.apache.spark.sql.catalyst.expressions.GeneratedClass$GeneratedIteratorForCodegenStage1.processNext(Unknown Source)

at org.apache.spark.sql.execution.BufferedRowIterator.hasNext(BufferedRowIterator.java:43)

at org.apache.spark.sql.execution.WholeStageCodegenExec$$anon$1.hasNext(WholeStageCodegenExec.scala:759)

at org.apache.spark.sql.execution.SparkPlan.$anonfun$getByteArrayRdd$1(SparkPlan.scala:349)

at org.apache.spark.rdd.RDD.$anonfun$mapPartitionsInternal$2(RDD.scala:898)

at org.apache.spark.rdd.RDD.$anonfun$mapPartitionsInternal$2$adapted(RDD.scala:898)

at org.apache.spark.rdd.MapPartitionsRDD.compute(MapPartitionsRDD.scala:52)

at org.apache.spark.rdd.RDD.computeOrReadCheckpoint(RDD.scala:373)

at org.apache.spark.rdd.RDD.iterator(RDD.scala:337)

at org.apache.spark.scheduler.ResultTask.runTask(ResultTask.scala:90)

at org.apache.spark.scheduler.Task.run(Task.scala:131)

at org.apache.spark.executor.Executor$TaskRunner.$anonfun$run$3(Executor.scala:506)

at org.apache.spark.util.Utils$.tryWithSafeFinally(Utils.scala:1462)

at org.apache.spark.executor.Executor$TaskRunner.run(Executor.scala:509)

at java.util.concurrent.ThreadPoolExecutor.runWorker(ThreadPoolExecutor.java:1149)

at java.util.concurrent.ThreadPoolExecutor$Worker.run(ThreadPoolExecutor.java:624)

at java.lang.Thread.run(Thread.java:750)

24/07/18 20:18:04 WARN scheduler.TaskSetManager: Lost task 0.0 in stage 8.0 (TID 8) (192.168.112.129 executor driver): java.lang.IllegalArgumentException: RejectRows does not exist. Available: IssuerId, IssuerId2, BusinessDate, StateCode, SourceName, NetworkName, NetworkURL, custnum, MarketCoverage, DentalOnlyPlan

at org.apache.spark.sql.types.StructType.$anonfun$fieldIndex$1(StructType.scala:306)

at scala.collection.immutable.HashMap$HashTrieMap.getOrElse0(HashMap.scala:596)

at scala.collection.immutable.HashMap.getOrElse(HashMap.scala:73)

at org.apache.spark.sql.types.StructType.fieldIndex(StructType.scala:305)

at org.apache.spark.sql.catalyst.OrderedFilters.$anonfun$predicates$4(OrderedFilters.scala:61)

at org.apache.spark.sql.catalyst.OrderedFilters.$anonfun$predicates$4$adapted(OrderedFilters.scala:61)

at scala.collection.TraversableLike.$anonfun$map$1(TraversableLike.scala:286)

at scala.collection.IndexedSeqOptimized.foreach(IndexedSeqOptimized.scala:36)

at scala.collection.IndexedSeqOptimized.foreach$(IndexedSeqOptimized.scala:33)

at scala.collection.mutable.ArrayOps$ofRef.foreach(ArrayOps.scala:198)

at scala.collection.TraversableLike.map(TraversableLike.scala:286)

at scala.collection.TraversableLike.map$(TraversableLike.scala:279)

at scala.collection.mutable.ArrayOps$ofRef.map(ArrayOps.scala:198)

at org.apache.spark.sql.catalyst.OrderedFilters.$anonfun$predicates$3(OrderedFilters.scala:61)

at org.apache.spark.sql.catalyst.OrderedFilters.$anonfun$predicates$3$adapted(OrderedFilters.scala:50)

at scala.collection.immutable.List.foreach(List.scala:431)

at org.apache.spark.sql.catalyst.OrderedFilters.<init>(OrderedFilters.scala:50)

at org.apache.spark.sql.catalyst.csv.UnivocityParser.<init>(UnivocityParser.scala:103)

at org.apache.spark.sql.execution.datasources.csv.CSVFileFormat.$anonfun$buildReader$1(CSVFileFormat.scala:130)

at org.apache.spark.sql.execution.datasources.FileFormat$$anon$1.apply(FileFormat.scala:148)

at org.apache.spark.sql.execution.datasources.FileFormat$$anon$1.apply(FileFormat.scala:133)

at org.apache.spark.sql.execution.datasources.FileScanRDD$$anon$1.org$apache$spark$sql$execution$datasources$FileScanRDD$$anon$$readCurrentFile(FileScanRDD.scala:127)

at org.apache.spark.sql.execution.datasources.FileScanRDD$$anon$1.nextIterator(FileScanRDD.scala:187)

at org.apache.spark.sql.execution.datasources.FileScanRDD$$anon$1.hasNext(FileScanRDD.scala:104)

at scala.collection.Iterator$$anon$10.hasNext(Iterator.scala:460)

at org.apache.spark.sql.catalyst.expressions.GeneratedClass$GeneratedIteratorForCodegenStage1.processNext(Unknown Source)

at org.apache.spark.sql.execution.BufferedRowIterator.hasNext(BufferedRowIterator.java:43)

at org.apache.spark.sql.execution.WholeStageCodegenExec$$anon$1.hasNext(WholeStageCodegenExec.scala:759)

at org.apache.spark.sql.execution.SparkPlan.$anonfun$getByteArrayRdd$1(SparkPlan.scala:349)

at org.apache.spark.rdd.RDD.$anonfun$mapPartitionsInternal$2(RDD.scala:898)

at org.apache.spark.rdd.RDD.$anonfun$mapPartitionsInternal$2$adapted(RDD.scala:898)

at org.apache.spark.rdd.MapPartitionsRDD.compute(MapPartitionsRDD.scala:52)

at org.apache.spark.rdd.RDD.computeOrReadCheckpoint(RDD.scala:373)

at org.apache.spark.rdd.RDD.iterator(RDD.scala:337)

at org.apache.spark.scheduler.ResultTask.runTask(ResultTask.scala:90)

at org.apache.spark.scheduler.Task.run(Task.scala:131)

at org.apache.spark.executor.Executor$TaskRunner.$anonfun$run$3(Executor.scala:506)

at org.apache.spark.util.Utils$.tryWithSafeFinally(Utils.scala:1462)

at org.apache.spark.executor.Executor$TaskRunner.run(Executor.scala:509)

at java.util.concurrent.ThreadPoolExecutor.runWorker(ThreadPoolExecutor.java:1149)

at java.util.concurrent.ThreadPoolExecutor$Worker.run(ThreadPoolExecutor.java:624)

at java.lang.Thread.run(Thread.java:750)

24/07/18 20:18:04 ERROR scheduler.TaskSetManager: Task 0 in stage 8.0 failed 1 times; aborting job

Traceback (most recent call last):

File "<stdin>", line 1, in <module>

File "/usr/local/spark/python/pyspark/sql/dataframe.py", line 494, in show

print(self.\_jdf.showString(n, 20, vertical))

File "/usr/local/spark/python/lib/py4j-0.10.9.3-src.zip/py4j/java\_gateway.py", line 1322, in \_\_call\_\_

File "/usr/local/spark/python/pyspark/sql/utils.py", line 111, in deco

return f(\*a, \*\*kw)

File "/usr/local/spark/python/lib/py4j-0.10.9.3-src.zip/py4j/protocol.py", line 328, in get\_return\_value

py4j.protocol.Py4JJavaError: An error occurred while calling o187.showString.

: org.apache.spark.SparkException: Job aborted due to stage failure: Task 0 in stage 8.0 failed 1 times, most recent failure: Lost task 0.0 in stage 8.0 (TID 8) (192.168.112.129 executor driver): java.lang.IllegalArgumentException: RejectRows does not exist. Available: IssuerId, IssuerId2, BusinessDate, StateCode, SourceName, NetworkName, NetworkURL, custnum, MarketCoverage, DentalOnlyPlan

at org.apache.spark.sql.types.StructType.$anonfun$fieldIndex$1(StructType.scala:306)

at scala.collection.immutable.HashMap$HashTrieMap.getOrElse0(HashMap.scala:596)

at scala.collection.immutable.HashMap.getOrElse(HashMap.scala:73)

at org.apache.spark.sql.types.StructType.fieldIndex(StructType.scala:305)

at org.apache.spark.sql.catalyst.OrderedFilters.$anonfun$predicates$4(OrderedFilters.scala:61)

at org.apache.spark.sql.catalyst.OrderedFilters.$anonfun$predicates$4$adapted(OrderedFilters.scala:61)

at scala.collection.TraversableLike.$anonfun$map$1(TraversableLike.scala:286)

at scala.collection.IndexedSeqOptimized.foreach(IndexedSeqOptimized.scala:36)

at scala.collection.IndexedSeqOptimized.foreach$(IndexedSeqOptimized.scala:33)

at scala.collection.mutable.ArrayOps$ofRef.foreach(ArrayOps.scala:198)

at scala.collection.TraversableLike.map(TraversableLike.scala:286)

at scala.collection.TraversableLike.map$(TraversableLike.scala:279)

at scala.collection.mutable.ArrayOps$ofRef.map(ArrayOps.scala:198)

at org.apache.spark.sql.catalyst.OrderedFilters.$anonfun$predicates$3(OrderedFilters.scala:61)

at org.apache.spark.sql.catalyst.OrderedFilters.$anonfun$predicates$3$adapted(OrderedFilters.scala:50)

at scala.collection.immutable.List.foreach(List.scala:431)

at org.apache.spark.sql.catalyst.OrderedFilters.<init>(OrderedFilters.scala:50)

at org.apache.spark.sql.catalyst.csv.UnivocityParser.<init>(UnivocityParser.scala:103)

at org.apache.spark.sql.execution.datasources.csv.CSVFileFormat.$anonfun$buildReader$1(CSVFileFormat.scala:130)

at org.apache.spark.sql.execution.datasources.FileFormat$$anon$1.apply(FileFormat.scala:148)

at org.apache.spark.sql.execution.datasources.FileFormat$$anon$1.apply(FileFormat.scala:133)

at org.apache.spark.sql.execution.datasources.FileScanRDD$$anon$1.org$apache$spark$sql$execution$datasources$FileScanRDD$$anon$$readCurrentFile(FileScanRDD.scala:127)

at org.apache.spark.sql.execution.datasources.FileScanRDD$$anon$1.nextIterator(FileScanRDD.scala:187)

at org.apache.spark.sql.execution.datasources.FileScanRDD$$anon$1.hasNext(FileScanRDD.scala:104)

at scala.collection.Iterator$$anon$10.hasNext(Iterator.scala:460)

at org.apache.spark.sql.catalyst.expressions.GeneratedClass$GeneratedIteratorForCodegenStage1.processNext(Unknown Source)

at org.apache.spark.sql.execution.BufferedRowIterator.hasNext(BufferedRowIterator.java:43)

at org.apache.spark.sql.execution.WholeStageCodegenExec$$anon$1.hasNext(WholeStageCodegenExec.scala:759)

at org.apache.spark.sql.execution.SparkPlan.$anonfun$getByteArrayRdd$1(SparkPlan.scala:349)

at org.apache.spark.rdd.RDD.$anonfun$mapPartitionsInternal$2(RDD.scala:898)

at org.apache.spark.rdd.RDD.$anonfun$mapPartitionsInternal$2$adapted(RDD.scala:898)

at org.apache.spark.rdd.MapPartitionsRDD.compute(MapPartitionsRDD.scala:52)

at org.apache.spark.rdd.RDD.computeOrReadCheckpoint(RDD.scala:373)

at org.apache.spark.rdd.RDD.iterator(RDD.scala:337)

at org.apache.spark.scheduler.ResultTask.runTask(ResultTask.scala:90)

at org.apache.spark.scheduler.Task.run(Task.scala:131)

at org.apache.spark.executor.Executor$TaskRunner.$anonfun$run$3(Executor.scala:506)

at org.apache.spark.util.Utils$.tryWithSafeFinally(Utils.scala:1462)

at org.apache.spark.executor.Executor$TaskRunner.run(Executor.scala:509)

at java.util.concurrent.ThreadPoolExecutor.runWorker(ThreadPoolExecutor.java:1149)

at java.util.concurrent.ThreadPoolExecutor$Worker.run(ThreadPoolExecutor.java:624)

at java.lang.Thread.run(Thread.java:750)

Driver stacktrace:

at org.apache.spark.scheduler.DAGScheduler.failJobAndIndependentStages(DAGScheduler.scala:2454)

at org.apache.spark.scheduler.DAGScheduler.$anonfun$abortStage$2(DAGScheduler.scala:2403)

at org.apache.spark.scheduler.DAGScheduler.$anonfun$abortStage$2$adapted(DAGScheduler.scala:2402)

at scala.collection.mutable.ResizableArray.foreach(ResizableArray.scala:62)

at scala.collection.mutable.ResizableArray.foreach$(ResizableArray.scala:55)

at scala.collection.mutable.ArrayBuffer.foreach(ArrayBuffer.scala:49)

at org.apache.spark.scheduler.DAGScheduler.abortStage(DAGScheduler.scala:2402)

at org.apache.spark.scheduler.DAGScheduler.$anonfun$handleTaskSetFailed$1(DAGScheduler.scala:1160)

at org.apache.spark.scheduler.DAGScheduler.$anonfun$handleTaskSetFailed$1$adapted(DAGScheduler.scala:1160)

at scala.Option.foreach(Option.scala:407)

at org.apache.spark.scheduler.DAGScheduler.handleTaskSetFailed(DAGScheduler.scala:1160)

at org.apache.spark.scheduler.DAGSchedulerEventProcessLoop.doOnReceive(DAGScheduler.scala:2642)

at org.apache.spark.scheduler.DAGSchedulerEventProcessLoop.onReceive(DAGScheduler.scala:2584)

at org.apache.spark.scheduler.DAGSchedulerEventProcessLoop.onReceive(DAGScheduler.scala:2573)

at org.apache.spark.util.EventLoop$$anon$1.run(EventLoop.scala:49)

at org.apache.spark.scheduler.DAGScheduler.runJob(DAGScheduler.scala:938)

at org.apache.spark.SparkContext.runJob(SparkContext.scala:2214)

at org.apache.spark.SparkContext.runJob(SparkContext.scala:2235)

at org.apache.spark.SparkContext.runJob(SparkContext.scala:2254)

at org.apache.spark.sql.execution.SparkPlan.executeTake(SparkPlan.scala:476)

at org.apache.spark.sql.execution.SparkPlan.executeTake(SparkPlan.scala:429)

at org.apache.spark.sql.execution.CollectLimitExec.executeCollect(limit.scala:48)

at org.apache.spark.sql.Dataset.collectFromPlan(Dataset.scala:3715)

at org.apache.spark.sql.Dataset.$anonfun$head$1(Dataset.scala:2728)

at org.apache.spark.sql.Dataset.$anonfun$withAction$1(Dataset.scala:3706)

at org.apache.spark.sql.execution.SQLExecution$.$anonfun$withNewExecutionId$5(SQLExecution.scala:103)

at org.apache.spark.sql.execution.SQLExecution$.withSQLConfPropagated(SQLExecution.scala:163)

at org.apache.spark.sql.execution.SQLExecution$.$anonfun$withNewExecutionId$1(SQLExecution.scala:90)

at org.apache.spark.sql.SparkSession.withActive(SparkSession.scala:775)

at org.apache.spark.sql.execution.SQLExecution$.withNewExecutionId(SQLExecution.scala:64)

at org.apache.spark.sql.Dataset.withAction(Dataset.scala:3704)

at org.apache.spark.sql.Dataset.head(Dataset.scala:2728)

at org.apache.spark.sql.Dataset.take(Dataset.scala:2935)

at org.apache.spark.sql.Dataset.getRows(Dataset.scala:287)

at org.apache.spark.sql.Dataset.showString(Dataset.scala:326)

at sun.reflect.NativeMethodAccessorImpl.invoke0(Native Method)

at sun.reflect.NativeMethodAccessorImpl.invoke(NativeMethodAccessorImpl.java:62)

at sun.reflect.DelegatingMethodAccessorImpl.invoke(DelegatingMethodAccessorImpl.java:43)

at java.lang.reflect.Method.invoke(Method.java:498)

at py4j.reflection.MethodInvoker.invoke(MethodInvoker.java:244)

at py4j.reflection.ReflectionEngine.invoke(ReflectionEngine.java:357)

at py4j.Gateway.invoke(Gateway.java:282)

at py4j.commands.AbstractCommand.invokeMethod(AbstractCommand.java:132)

at py4j.commands.CallCommand.execute(CallCommand.java:79)

at py4j.ClientServerConnection.waitForCommands(ClientServerConnection.java:182)

at py4j.ClientServerConnection.run(ClientServerConnection.java:106)

at java.lang.Thread.run(Thread.java:750)

Caused by: java.lang.IllegalArgumentException: RejectRows does not exist. Available: IssuerId, IssuerId2, BusinessDate, StateCode, SourceName, NetworkName, NetworkURL, custnum, MarketCoverage, DentalOnlyPlan

at org.apache.spark.sql.types.StructType.$anonfun$fieldIndex$1(StructType.scala:306)

at scala.collection.immutable.HashMap$HashTrieMap.getOrElse0(HashMap.scala:596)

at scala.collection.immutable.HashMap.getOrElse(HashMap.scala:73)

at org.apache.spark.sql.types.StructType.fieldIndex(StructType.scala:305)

at org.apache.spark.sql.catalyst.OrderedFilters.$anonfun$predicates$4(OrderedFilters.scala:61)

at org.apache.spark.sql.catalyst.OrderedFilters.$anonfun$predicates$4$adapted(OrderedFilters.scala:61)

at scala.collection.TraversableLike.$anonfun$map$1(TraversableLike.scala:286)

at scala.collection.IndexedSeqOptimized.foreach(IndexedSeqOptimized.scala:36)

at scala.collection.IndexedSeqOptimized.foreach$(IndexedSeqOptimized.scala:33)

at scala.collection.mutable.ArrayOps$ofRef.foreach(ArrayOps.scala:198)

at scala.collection.TraversableLike.map(TraversableLike.scala:286)

at scala.collection.TraversableLike.map$(TraversableLike.scala:279)

at scala.collection.mutable.ArrayOps$ofRef.map(ArrayOps.scala:198)

at org.apache.spark.sql.catalyst.OrderedFilters.$anonfun$predicates$3(OrderedFilters.scala:61)

at org.apache.spark.sql.catalyst.OrderedFilters.$anonfun$predicates$3$adapted(OrderedFilters.scala:50)

at scala.collection.immutable.List.foreach(List.scala:431)

at org.apache.spark.sql.catalyst.OrderedFilters.<init>(OrderedFilters.scala:50)

at org.apache.spark.sql.catalyst.csv.UnivocityParser.<init>(UnivocityParser.scala:103)

at org.apache.spark.sql.execution.datasources.csv.CSVFileFormat.$anonfun$buildReader$1(CSVFileFormat.scala:130)

at org.apache.spark.sql.execution.datasources.FileFormat$$anon$1.apply(FileFormat.scala:148)

at org.apache.spark.sql.execution.datasources.FileFormat$$anon$1.apply(FileFormat.scala:133)

at org.apache.spark.sql.execution.datasources.FileScanRDD$$anon$1.org$apache$spark$sql$execution$datasources$FileScanRDD$$anon$$readCurrentFile(FileScanRDD.scala:127)

at org.apache.spark.sql.execution.datasources.FileScanRDD$$anon$1.nextIterator(FileScanRDD.scala:187)

at org.apache.spark.sql.execution.datasources.FileScanRDD$$anon$1.hasNext(FileScanRDD.scala:104)

at scala.collection.Iterator$$anon$10.hasNext(Iterator.scala:460)

at org.apache.spark.sql.catalyst.expressions.GeneratedClass$GeneratedIteratorForCodegenStage1.processNext(Unknown Source)

at org.apache.spark.sql.execution.BufferedRowIterator.hasNext(BufferedRowIterator.java:43)

at org.apache.spark.sql.execution.WholeStageCodegenExec$$anon$1.hasNext(WholeStageCodegenExec.scala:759)

at org.apache.spark.sql.execution.SparkPlan.$anonfun$getByteArrayRdd$1(SparkPlan.scala:349)

at org.apache.spark.rdd.RDD.$anonfun$mapPartitionsInternal$2(RDD.scala:898)

at org.apache.spark.rdd.RDD.$anonfun$mapPartitionsInternal$2$adapted(RDD.scala:898)

at org.apache.spark.rdd.MapPartitionsRDD.compute(MapPartitionsRDD.scala:52)

at org.apache.spark.rdd.RDD.computeOrReadCheckpoint(RDD.scala:373)

at org.apache.spark.rdd.RDD.iterator(RDD.scala:337)

at org.apache.spark.scheduler.ResultTask.runTask(ResultTask.scala:90)

at org.apache.spark.scheduler.Task.run(Task.scala:131)

at org.apache.spark.executor.Executor$TaskRunner.$anonfun$run$3(Executor.scala:506)

at org.apache.spark.util.Utils$.tryWithSafeFinally(Utils.scala:1462)

at org.apache.spark.executor.Executor$TaskRunner.run(Executor.scala:509)

at java.util.concurrent.ThreadPoolExecutor.runWorker(ThreadPoolExecutor.java:1149)

at java.util.concurrent.ThreadPoolExecutor$Worker.run(ThreadPoolExecutor.java:624)

... 1 more

>>> data\_frame\_3.where("RejectRows not NULL ").show(3)

Traceback (most recent call last):

File "<stdin>", line 1, in <module>

File "/usr/local/spark/python/pyspark/sql/dataframe.py", line 1731, in filter

jdf = self.\_jdf.filter(condition)

File "/usr/local/spark/python/lib/py4j-0.10.9.3-src.zip/py4j/java\_gateway.py", line 1322, in \_\_call\_\_

File "/usr/local/spark/python/pyspark/sql/utils.py", line 117, in deco

raise converted from None

pyspark.sql.utils.ParseException:

extraneous input 'NULL' expecting <EOF>(line 1, pos 15)

== SQL ==

RejectRows not NULL

---------------^^^

>>> data\_frame\_3.where("RejectRows != NULL ").show(3)

+--------+---------+------------+---------+----------+-----------+----------+-------+--------------+--------------+----------+

|IssuerId|IssuerId2|BusinessDate|StateCode|SourceName|NetworkName|NetworkURL|custnum|MarketCoverage|DentalOnlyPlan|RejectRows|

+--------+---------+------------+---------+----------+-----------+----------+-------+--------------+--------------+----------+

+--------+---------+------------+---------+----------+-----------+----------+-------+--------------+--------------+----------+

>>> data\_frame\_1=spark.read.csv(path="/user/hduser/Hackathon/insuranceinfo1.csv" , header=True , schema=custom\_schema , columnNameOfCorruptRecord="RejectRows" ,dateFormat="MM-dd-yyyy")

>>> data\_frame\_1=spark.read.csv(path="/user/hduser/Hackathon/insuranceinfo1.csv" , header=True , schema=custom\_schema , columnNameOfCorruptRecord="RejectRows" ,dateFormat="MM-dd-yyyy")

>>> data\_frame\_1.show(3)

+--------+---------+------------+---------+----------+------------+--------------------+-------+--------------+--------------+--------------------+

|IssuerId|IssuerId2|BusinessDate|StateCode|SourceName| NetworkName| NetworkURL|custnum|MarketCoverage|DentalOnlyPlan| RejectRows|

+--------+---------+------------+---------+----------+------------+--------------------+-------+--------------+--------------+--------------------+

| 21989| 21989| 2019-10-01| AK| HIOS| ODS Premier|https://www.modah...| 13| null| null| null|

| 219| 219| 2019-10-01| AK| HIOS| ODS Premier| null| null| null| null|219,219,2019-10-0...|

| 38344| 38344| 2019-10-01| AK| HIOS|HeritagePlus|https://www.preme...| 13| null| null| null|

+--------+---------+------------+---------+----------+------------+--------------------+-------+--------------+--------------+--------------------+

only showing top 3 rows

>>> data\_frame\_1.where("RejectRows != NULL ").show(3)

+--------+---------+------------+---------+----------+-----------+----------+-------+--------------+--------------+----------+

|IssuerId|IssuerId2|BusinessDate|StateCode|SourceName|NetworkName|NetworkURL|custnum|MarketCoverage|DentalOnlyPlan|RejectRows|

+--------+---------+------------+---------+----------+-----------+----------+-------+--------------+--------------+----------+

+--------+---------+------------+---------+----------+-----------+----------+-------+--------------+--------------+----------+

>>> data\_frame\_1.show(3)

+--------+---------+------------+---------+----------+------------+--------------------+-------+--------------+--------------+--------------------+

|IssuerId|IssuerId2|BusinessDate|StateCode|SourceName| NetworkName| NetworkURL|custnum|MarketCoverage|DentalOnlyPlan| RejectRows|

+--------+---------+------------+---------+----------+------------+--------------------+-------+--------------+--------------+--------------------+

| 21989| 21989| 2019-10-01| AK| HIOS| ODS Premier|https://www.modah...| 13| null| null| null|

| 219| 219| 2019-10-01| AK| HIOS| ODS Premier| null| null| null| null|219,219,2019-10-0...|

| 38344| 38344| 2019-10-01| AK| HIOS|HeritagePlus|https://www.preme...| 13| null| null| null|

+--------+---------+------------+---------+----------+------------+--------------------+-------+--------------+--------------+--------------------+

only showing top 3 rows

>>> data\_frame\_1.where("RejectRows is not null").show(3)

24/07/18 20:34:54 ERROR executor.Executor: Exception in task 0.0 in stage 11.0 (TID 11)

java.lang.IllegalArgumentException: RejectRows does not exist. Available: IssuerId, IssuerId2, BusinessDate, StateCode, SourceName, NetworkName, NetworkURL, custnum, MarketCoverage, DentalOnlyPlan

at org.apache.spark.sql.types.StructType.$anonfun$fieldIndex$1(StructType.scala:306)

at scala.collection.immutable.HashMap$HashTrieMap.getOrElse0(HashMap.scala:596)

at scala.collection.immutable.HashMap.getOrElse(HashMap.scala:73)

at org.apache.spark.sql.types.StructType.fieldIndex(StructType.scala:305)

at org.apache.spark.sql.catalyst.OrderedFilters.$anonfun$predicates$4(OrderedFilters.scala:61)

at org.apache.spark.sql.catalyst.OrderedFilters.$anonfun$predicates$4$adapted(OrderedFilters.scala:61)

at scala.collection.TraversableLike.$anonfun$map$1(TraversableLike.scala:286)

at scala.collection.IndexedSeqOptimized.foreach(IndexedSeqOptimized.scala:36)

at scala.collection.IndexedSeqOptimized.foreach$(IndexedSeqOptimized.scala:33)

at scala.collection.mutable.ArrayOps$ofRef.foreach(ArrayOps.scala:198)

at scala.collection.TraversableLike.map(TraversableLike.scala:286)

at scala.collection.TraversableLike.map$(TraversableLike.scala:279)

at scala.collection.mutable.ArrayOps$ofRef.map(ArrayOps.scala:198)

at org.apache.spark.sql.catalyst.OrderedFilters.$anonfun$predicates$3(OrderedFilters.scala:61)

at org.apache.spark.sql.catalyst.OrderedFilters.$anonfun$predicates$3$adapted(OrderedFilters.scala:50)

at scala.collection.immutable.List.foreach(List.scala:431)

at org.apache.spark.sql.catalyst.OrderedFilters.<init>(OrderedFilters.scala:50)

at org.apache.spark.sql.catalyst.csv.UnivocityParser.<init>(UnivocityParser.scala:103)

at org.apache.spark.sql.execution.datasources.csv.CSVFileFormat.$anonfun$buildReader$1(CSVFileFormat.scala:130)

at org.apache.spark.sql.execution.datasources.FileFormat$$anon$1.apply(FileFormat.scala:148)

at org.apache.spark.sql.execution.datasources.FileFormat$$anon$1.apply(FileFormat.scala:133)

at org.apache.spark.sql.execution.datasources.FileScanRDD$$anon$1.org$apache$spark$sql$execution$datasources$FileScanRDD$$anon$$readCurrentFile(FileScanRDD.scala:127)

at org.apache.spark.sql.execution.datasources.FileScanRDD$$anon$1.nextIterator(FileScanRDD.scala:187)

at org.apache.spark.sql.execution.datasources.FileScanRDD$$anon$1.hasNext(FileScanRDD.scala:104)

at scala.collection.Iterator$$anon$10.hasNext(Iterator.scala:460)

at org.apache.spark.sql.catalyst.expressions.GeneratedClass$GeneratedIteratorForCodegenStage1.processNext(Unknown Source)

at org.apache.spark.sql.execution.BufferedRowIterator.hasNext(BufferedRowIterator.java:43)

at org.apache.spark.sql.execution.WholeStageCodegenExec$$anon$1.hasNext(WholeStageCodegenExec.scala:759)

at org.apache.spark.sql.execution.SparkPlan.$anonfun$getByteArrayRdd$1(SparkPlan.scala:349)

at org.apache.spark.rdd.RDD.$anonfun$mapPartitionsInternal$2(RDD.scala:898)

at org.apache.spark.rdd.RDD.$anonfun$mapPartitionsInternal$2$adapted(RDD.scala:898)

at org.apache.spark.rdd.MapPartitionsRDD.compute(MapPartitionsRDD.scala:52)

at org.apache.spark.rdd.RDD.computeOrReadCheckpoint(RDD.scala:373)

at org.apache.spark.rdd.RDD.iterator(RDD.scala:337)

at org.apache.spark.scheduler.ResultTask.runTask(ResultTask.scala:90)

at org.apache.spark.scheduler.Task.run(Task.scala:131)

at org.apache.spark.executor.Executor$TaskRunner.$anonfun$run$3(Executor.scala:506)

at org.apache.spark.util.Utils$.tryWithSafeFinally(Utils.scala:1462)

at org.apache.spark.executor.Executor$TaskRunner.run(Executor.scala:509)

at java.util.concurrent.ThreadPoolExecutor.runWorker(ThreadPoolExecutor.java:1149)

at java.util.concurrent.ThreadPoolExecutor$Worker.run(ThreadPoolExecutor.java:624)

at java.lang.Thread.run(Thread.java:750)

24/07/18 20:34:54 WARN scheduler.TaskSetManager: Lost task 0.0 in stage 11.0 (TID 11) (192.168.112.129 executor driver): java.lang.IllegalArgumentException: RejectRows does not exist. Available: IssuerId, IssuerId2, BusinessDate, StateCode, SourceName, NetworkName, NetworkURL, custnum, MarketCoverage, DentalOnlyPlan

at org.apache.spark.sql.types.StructType.$anonfun$fieldIndex$1(StructType.scala:306)

at scala.collection.immutable.HashMap$HashTrieMap.getOrElse0(HashMap.scala:596)

at scala.collection.immutable.HashMap.getOrElse(HashMap.scala:73)

at org.apache.spark.sql.types.StructType.fieldIndex(StructType.scala:305)

at org.apache.spark.sql.catalyst.OrderedFilters.$anonfun$predicates$4(OrderedFilters.scala:61)

at org.apache.spark.sql.catalyst.OrderedFilters.$anonfun$predicates$4$adapted(OrderedFilters.scala:61)

at scala.collection.TraversableLike.$anonfun$map$1(TraversableLike.scala:286)

at scala.collection.IndexedSeqOptimized.foreach(IndexedSeqOptimized.scala:36)

at scala.collection.IndexedSeqOptimized.foreach$(IndexedSeqOptimized.scala:33)

at scala.collection.mutable.ArrayOps$ofRef.foreach(ArrayOps.scala:198)

at scala.collection.TraversableLike.map(TraversableLike.scala:286)

at scala.collection.TraversableLike.map$(TraversableLike.scala:279)

at scala.collection.mutable.ArrayOps$ofRef.map(ArrayOps.scala:198)

at org.apache.spark.sql.catalyst.OrderedFilters.$anonfun$predicates$3(OrderedFilters.scala:61)

at org.apache.spark.sql.catalyst.OrderedFilters.$anonfun$predicates$3$adapted(OrderedFilters.scala:50)

at scala.collection.immutable.List.foreach(List.scala:431)

at org.apache.spark.sql.catalyst.OrderedFilters.<init>(OrderedFilters.scala:50)

at org.apache.spark.sql.catalyst.csv.UnivocityParser.<init>(UnivocityParser.scala:103)

at org.apache.spark.sql.execution.datasources.csv.CSVFileFormat.$anonfun$buildReader$1(CSVFileFormat.scala:130)

at org.apache.spark.sql.execution.datasources.FileFormat$$anon$1.apply(FileFormat.scala:148)

at org.apache.spark.sql.execution.datasources.FileFormat$$anon$1.apply(FileFormat.scala:133)

at org.apache.spark.sql.execution.datasources.FileScanRDD$$anon$1.org$apache$spark$sql$execution$datasources$FileScanRDD$$anon$$readCurrentFile(FileScanRDD.scala:127)

at org.apache.spark.sql.execution.datasources.FileScanRDD$$anon$1.nextIterator(FileScanRDD.scala:187)

at org.apache.spark.sql.execution.datasources.FileScanRDD$$anon$1.hasNext(FileScanRDD.scala:104)

at scala.collection.Iterator$$anon$10.hasNext(Iterator.scala:460)

at org.apache.spark.sql.catalyst.expressions.GeneratedClass$GeneratedIteratorForCodegenStage1.processNext(Unknown Source)

at org.apache.spark.sql.execution.BufferedRowIterator.hasNext(BufferedRowIterator.java:43)

at org.apache.spark.sql.execution.WholeStageCodegenExec$$anon$1.hasNext(WholeStageCodegenExec.scala:759)

at org.apache.spark.sql.execution.SparkPlan.$anonfun$getByteArrayRdd$1(SparkPlan.scala:349)

at org.apache.spark.rdd.RDD.$anonfun$mapPartitionsInternal$2(RDD.scala:898)

at org.apache.spark.rdd.RDD.$anonfun$mapPartitionsInternal$2$adapted(RDD.scala:898)

at org.apache.spark.rdd.MapPartitionsRDD.compute(MapPartitionsRDD.scala:52)

at org.apache.spark.rdd.RDD.computeOrReadCheckpoint(RDD.scala:373)

at org.apache.spark.rdd.RDD.iterator(RDD.scala:337)

at org.apache.spark.scheduler.ResultTask.runTask(ResultTask.scala:90)

at org.apache.spark.scheduler.Task.run(Task.scala:131)

at org.apache.spark.executor.Executor$TaskRunner.$anonfun$run$3(Executor.scala:506)

at org.apache.spark.util.Utils$.tryWithSafeFinally(Utils.scala:1462)

at org.apache.spark.executor.Executor$TaskRunner.run(Executor.scala:509)

at java.util.concurrent.ThreadPoolExecutor.runWorker(ThreadPoolExecutor.java:1149)

at java.util.concurrent.ThreadPoolExecutor$Worker.run(ThreadPoolExecutor.java:624)

at java.lang.Thread.run(Thread.java:750)

24/07/18 20:34:54 ERROR scheduler.TaskSetManager: Task 0 in stage 11.0 failed 1 times; aborting job

Traceback (most recent call last):

File "<stdin>", line 1, in <module>

File "/usr/local/spark/python/pyspark/sql/dataframe.py", line 494, in show

print(self.\_jdf.showString(n, 20, vertical))

File "/usr/local/spark/python/lib/py4j-0.10.9.3-src.zip/py4j/java\_gateway.py", line 1322, in \_\_call\_\_

File "/usr/local/spark/python/pyspark/sql/utils.py", line 111, in deco

return f(\*a, \*\*kw)

File "/usr/local/spark/python/lib/py4j-0.10.9.3-src.zip/py4j/protocol.py", line 328, in get\_return\_value

py4j.protocol.Py4JJavaError: An error occurred while calling o209.showString.

: org.apache.spark.SparkException: Job aborted due to stage failure: Task 0 in stage 11.0 failed 1 times, most recent failure: Lost task 0.0 in stage 11.0 (TID 11) (192.168.112.129 executor driver): java.lang.IllegalArgumentException: RejectRows does not exist. Available: IssuerId, IssuerId2, BusinessDate, StateCode, SourceName, NetworkName, NetworkURL, custnum, MarketCoverage, DentalOnlyPlan

at org.apache.spark.sql.types.StructType.$anonfun$fieldIndex$1(StructType.scala:306)

at scala.collection.immutable.HashMap$HashTrieMap.getOrElse0(HashMap.scala:596)

at scala.collection.immutable.HashMap.getOrElse(HashMap.scala:73)

at org.apache.spark.sql.types.StructType.fieldIndex(StructType.scala:305)

at org.apache.spark.sql.catalyst.OrderedFilters.$anonfun$predicates$4(OrderedFilters.scala:61)

at org.apache.spark.sql.catalyst.OrderedFilters.$anonfun$predicates$4$adapted(OrderedFilters.scala:61)

at scala.collection.TraversableLike.$anonfun$map$1(TraversableLike.scala:286)

at scala.collection.IndexedSeqOptimized.foreach(IndexedSeqOptimized.scala:36)

at scala.collection.IndexedSeqOptimized.foreach$(IndexedSeqOptimized.scala:33)

at scala.collection.mutable.ArrayOps$ofRef.foreach(ArrayOps.scala:198)

at scala.collection.TraversableLike.map(TraversableLike.scala:286)

at scala.collection.TraversableLike.map$(TraversableLike.scala:279)

at scala.collection.mutable.ArrayOps$ofRef.map(ArrayOps.scala:198)

at org.apache.spark.sql.catalyst.OrderedFilters.$anonfun$predicates$3(OrderedFilters.scala:61)

at org.apache.spark.sql.catalyst.OrderedFilters.$anonfun$predicates$3$adapted(OrderedFilters.scala:50)

at scala.collection.immutable.List.foreach(List.scala:431)

at org.apache.spark.sql.catalyst.OrderedFilters.<init>(OrderedFilters.scala:50)

at org.apache.spark.sql.catalyst.csv.UnivocityParser.<init>(UnivocityParser.scala:103)

at org.apache.spark.sql.execution.datasources.csv.CSVFileFormat.$anonfun$buildReader$1(CSVFileFormat.scala:130)

at org.apache.spark.sql.execution.datasources.FileFormat$$anon$1.apply(FileFormat.scala:148)

at org.apache.spark.sql.execution.datasources.FileFormat$$anon$1.apply(FileFormat.scala:133)

at org.apache.spark.sql.execution.datasources.FileScanRDD$$anon$1.org$apache$spark$sql$execution$datasources$FileScanRDD$$anon$$readCurrentFile(FileScanRDD.scala:127)

at org.apache.spark.sql.execution.datasources.FileScanRDD$$anon$1.nextIterator(FileScanRDD.scala:187)

at org.apache.spark.sql.execution.datasources.FileScanRDD$$anon$1.hasNext(FileScanRDD.scala:104)

at scala.collection.Iterator$$anon$10.hasNext(Iterator.scala:460)

at org.apache.spark.sql.catalyst.expressions.GeneratedClass$GeneratedIteratorForCodegenStage1.processNext(Unknown Source)

at org.apache.spark.sql.execution.BufferedRowIterator.hasNext(BufferedRowIterator.java:43)

at org.apache.spark.sql.execution.WholeStageCodegenExec$$anon$1.hasNext(WholeStageCodegenExec.scala:759)

at org.apache.spark.sql.execution.SparkPlan.$anonfun$getByteArrayRdd$1(SparkPlan.scala:349)

at org.apache.spark.rdd.RDD.$anonfun$mapPartitionsInternal$2(RDD.scala:898)

at org.apache.spark.rdd.RDD.$anonfun$mapPartitionsInternal$2$adapted(RDD.scala:898)

at org.apache.spark.rdd.MapPartitionsRDD.compute(MapPartitionsRDD.scala:52)

at org.apache.spark.rdd.RDD.computeOrReadCheckpoint(RDD.scala:373)

at org.apache.spark.rdd.RDD.iterator(RDD.scala:337)

at org.apache.spark.scheduler.ResultTask.runTask(ResultTask.scala:90)

at org.apache.spark.scheduler.Task.run(Task.scala:131)

at org.apache.spark.executor.Executor$TaskRunner.$anonfun$run$3(Executor.scala:506)

at org.apache.spark.util.Utils$.tryWithSafeFinally(Utils.scala:1462)

at org.apache.spark.executor.Executor$TaskRunner.run(Executor.scala:509)

at java.util.concurrent.ThreadPoolExecutor.runWorker(ThreadPoolExecutor.java:1149)

at java.util.concurrent.ThreadPoolExecutor$Worker.run(ThreadPoolExecutor.java:624)

at java.lang.Thread.run(Thread.java:750)

Driver stacktrace:

at org.apache.spark.scheduler.DAGScheduler.failJobAndIndependentStages(DAGScheduler.scala:2454)

at org.apache.spark.scheduler.DAGScheduler.$anonfun$abortStage$2(DAGScheduler.scala:2403)

at org.apache.spark.scheduler.DAGScheduler.$anonfun$abortStage$2$adapted(DAGScheduler.scala:2402)

at scala.collection.mutable.ResizableArray.foreach(ResizableArray.scala:62)

at scala.collection.mutable.ResizableArray.foreach$(ResizableArray.scala:55)

at scala.collection.mutable.ArrayBuffer.foreach(ArrayBuffer.scala:49)

at org.apache.spark.scheduler.DAGScheduler.abortStage(DAGScheduler.scala:2402)

at org.apache.spark.scheduler.DAGScheduler.$anonfun$handleTaskSetFailed$1(DAGScheduler.scala:1160)

at org.apache.spark.scheduler.DAGScheduler.$anonfun$handleTaskSetFailed$1$adapted(DAGScheduler.scala:1160)

at scala.Option.foreach(Option.scala:407)

at org.apache.spark.scheduler.DAGScheduler.handleTaskSetFailed(DAGScheduler.scala:1160)

at org.apache.spark.scheduler.DAGSchedulerEventProcessLoop.doOnReceive(DAGScheduler.scala:2642)

at org.apache.spark.scheduler.DAGSchedulerEventProcessLoop.onReceive(DAGScheduler.scala:2584)

at org.apache.spark.scheduler.DAGSchedulerEventProcessLoop.onReceive(DAGScheduler.scala:2573)

at org.apache.spark.util.EventLoop$$anon$1.run(EventLoop.scala:49)

at org.apache.spark.scheduler.DAGScheduler.runJob(DAGScheduler.scala:938)

at org.apache.spark.SparkContext.runJob(SparkContext.scala:2214)

at org.apache.spark.SparkContext.runJob(SparkContext.scala:2235)

at org.apache.spark.SparkContext.runJob(SparkContext.scala:2254)

at org.apache.spark.sql.execution.SparkPlan.executeTake(SparkPlan.scala:476)

at org.apache.spark.sql.execution.SparkPlan.executeTake(SparkPlan.scala:429)

at org.apache.spark.sql.execution.CollectLimitExec.executeCollect(limit.scala:48)

at org.apache.spark.sql.Dataset.collectFromPlan(Dataset.scala:3715)

at org.apache.spark.sql.Dataset.$anonfun$head$1(Dataset.scala:2728)

at org.apache.spark.sql.Dataset.$anonfun$withAction$1(Dataset.scala:3706)

at org.apache.spark.sql.execution.SQLExecution$.$anonfun$withNewExecutionId$5(SQLExecution.scala:103)

at org.apache.spark.sql.execution.SQLExecution$.withSQLConfPropagated(SQLExecution.scala:163)

at org.apache.spark.sql.execution.SQLExecution$.$anonfun$withNewExecutionId$1(SQLExecution.scala:90)

at org.apache.spark.sql.SparkSession.withActive(SparkSession.scala:775)

at org.apache.spark.sql.execution.SQLExecution$.withNewExecutionId(SQLExecution.scala:64)

at org.apache.spark.sql.Dataset.withAction(Dataset.scala:3704)

at org.apache.spark.sql.Dataset.head(Dataset.scala:2728)

at org.apache.spark.sql.Dataset.take(Dataset.scala:2935)

at org.apache.spark.sql.Dataset.getRows(Dataset.scala:287)

at org.apache.spark.sql.Dataset.showString(Dataset.scala:326)

at sun.reflect.NativeMethodAccessorImpl.invoke0(Native Method)

at sun.reflect.NativeMethodAccessorImpl.invoke(NativeMethodAccessorImpl.java:62)

at sun.reflect.DelegatingMethodAccessorImpl.invoke(DelegatingMethodAccessorImpl.java:43)

at java.lang.reflect.Method.invoke(Method.java:498)

at py4j.reflection.MethodInvoker.invoke(MethodInvoker.java:244)

at py4j.reflection.ReflectionEngine.invoke(ReflectionEngine.java:357)

at py4j.Gateway.invoke(Gateway.java:282)

at py4j.commands.AbstractCommand.invokeMethod(AbstractCommand.java:132)

at py4j.commands.CallCommand.execute(CallCommand.java:79)

at py4j.ClientServerConnection.waitForCommands(ClientServerConnection.java:182)

at py4j.ClientServerConnection.run(ClientServerConnection.java:106)

at java.lang.Thread.run(Thread.java:750)

Caused by: java.lang.IllegalArgumentException: RejectRows does not exist. Available: IssuerId, IssuerId2, BusinessDate, StateCode, SourceName, NetworkName, NetworkURL, custnum, MarketCoverage, DentalOnlyPlan

at org.apache.spark.sql.types.StructType.$anonfun$fieldIndex$1(StructType.scala:306)

at scala.collection.immutable.HashMap$HashTrieMap.getOrElse0(HashMap.scala:596)

at scala.collection.immutable.HashMap.getOrElse(HashMap.scala:73)

at org.apache.spark.sql.types.StructType.fieldIndex(StructType.scala:305)

at org.apache.spark.sql.catalyst.OrderedFilters.$anonfun$predicates$4(OrderedFilters.scala:61)

at org.apache.spark.sql.catalyst.OrderedFilters.$anonfun$predicates$4$adapted(OrderedFilters.scala:61)

at scala.collection.TraversableLike.$anonfun$map$1(TraversableLike.scala:286)

at scala.collection.IndexedSeqOptimized.foreach(IndexedSeqOptimized.scala:36)

at scala.collection.IndexedSeqOptimized.foreach$(IndexedSeqOptimized.scala:33)

at scala.collection.mutable.ArrayOps$ofRef.foreach(ArrayOps.scala:198)

at scala.collection.TraversableLike.map(TraversableLike.scala:286)

at scala.collection.TraversableLike.map$(TraversableLike.scala:279)

at scala.collection.mutable.ArrayOps$ofRef.map(ArrayOps.scala:198)

at org.apache.spark.sql.catalyst.OrderedFilters.$anonfun$predicates$3(OrderedFilters.scala:61)

at org.apache.spark.sql.catalyst.OrderedFilters.$anonfun$predicates$3$adapted(OrderedFilters.scala:50)

at scala.collection.immutable.List.foreach(List.scala:431)

at org.apache.spark.sql.catalyst.OrderedFilters.<init>(OrderedFilters.scala:50)

at org.apache.spark.sql.catalyst.csv.UnivocityParser.<init>(UnivocityParser.scala:103)

at org.apache.spark.sql.execution.datasources.csv.CSVFileFormat.$anonfun$buildReader$1(CSVFileFormat.scala:130)

at org.apache.spark.sql.execution.datasources.FileFormat$$anon$1.apply(FileFormat.scala:148)

at org.apache.spark.sql.execution.datasources.FileFormat$$anon$1.apply(FileFormat.scala:133)

at org.apache.spark.sql.execution.datasources.FileScanRDD$$anon$1.org$apache$spark$sql$execution$datasources$FileScanRDD$$anon$$readCurrentFile(FileScanRDD.scala:127)

at org.apache.spark.sql.execution.datasources.FileScanRDD$$anon$1.nextIterator(FileScanRDD.scala:187)

at org.apache.spark.sql.execution.datasources.FileScanRDD$$anon$1.hasNext(FileScanRDD.scala:104)

at scala.collection.Iterator$$anon$10.hasNext(Iterator.scala:460)

at org.apache.spark.sql.catalyst.expressions.GeneratedClass$GeneratedIteratorForCodegenStage1.processNext(Unknown Source)

at org.apache.spark.sql.execution.BufferedRowIterator.hasNext(BufferedRowIterator.java:43)

at org.apache.spark.sql.execution.WholeStageCodegenExec$$anon$1.hasNext(WholeStageCodegenExec.scala:759)

at org.apache.spark.sql.execution.SparkPlan.$anonfun$getByteArrayRdd$1(SparkPlan.scala:349)

at org.apache.spark.rdd.RDD.$anonfun$mapPartitionsInternal$2(RDD.scala:898)

at org.apache.spark.rdd.RDD.$anonfun$mapPartitionsInternal$2$adapted(RDD.scala:898)

at org.apache.spark.rdd.MapPartitionsRDD.compute(MapPartitionsRDD.scala:52)

at org.apache.spark.rdd.RDD.computeOrReadCheckpoint(RDD.scala:373)

at org.apache.spark.rdd.RDD.iterator(RDD.scala:337)

at org.apache.spark.scheduler.ResultTask.runTask(ResultTask.scala:90)

at org.apache.spark.scheduler.Task.run(Task.scala:131)

at org.apache.spark.executor.Executor$TaskRunner.$anonfun$run$3(Executor.scala:506)

at org.apache.spark.util.Utils$.tryWithSafeFinally(Utils.scala:1462)

at org.apache.spark.executor.Executor$TaskRunner.run(Executor.scala:509)

at java.util.concurrent.ThreadPoolExecutor.runWorker(ThreadPoolExecutor.java:1149)

at java.util.concurrent.ThreadPoolExecutor$Worker.run(ThreadPoolExecutor.java:624)

... 1 more

>>> data\_frame\_1.where("RejectRows == null").show(3)

+--------+---------+------------+---------+----------+-----------+----------+-------+--------------+--------------+----------+

|IssuerId|IssuerId2|BusinessDate|StateCode|SourceName|NetworkName|NetworkURL|custnum|MarketCoverage|DentalOnlyPlan|RejectRows|

+--------+---------+------------+---------+----------+-----------+----------+-------+--------------+--------------+----------+

+--------+---------+------------+---------+----------+-----------+----------+-------+--------------+--------------+----------+

>>> data\_frame\_1.where("RejectRows== null").show(3)

+--------+---------+------------+---------+----------+-----------+----------+-------+--------------+--------------+----------+

|IssuerId|IssuerId2|BusinessDate|StateCode|SourceName|NetworkName|NetworkURL|custnum|MarketCoverage|DentalOnlyPlan|RejectRows|

+--------+---------+------------+---------+----------+-----------+----------+-------+--------------+--------------+----------+

+--------+---------+------------+---------+----------+-----------+----------+-------+--------------+--------------+----------+

>>> data\_frame\_1.where("RejectRows==null").show(3)

+--------+---------+------------+---------+----------+-----------+----------+-------+--------------+--------------+----------+

|IssuerId|IssuerId2|BusinessDate|StateCode|SourceName|NetworkName|NetworkURL|custnum|MarketCoverage|DentalOnlyPlan|RejectRows|

+--------+---------+------------+---------+----------+-----------+----------+-------+--------------+--------------+----------+

+--------+---------+------------+---------+----------+-----------+----------+-------+--------------+--------------+----------+

>>> data\_frame\_1.where("RejectRows=='null'").show(3)

24/07/18 20:35:26 ERROR executor.Executor: Exception in task 0.0 in stage 12.0 (TID 12)

java.lang.IllegalArgumentException: RejectRows does not exist. Available: IssuerId, IssuerId2, BusinessDate, StateCode, SourceName, NetworkName, NetworkURL, custnum, MarketCoverage, DentalOnlyPlan

at org.apache.spark.sql.types.StructType.$anonfun$fieldIndex$1(StructType.scala:306)

at scala.collection.immutable.HashMap$HashTrieMap.getOrElse0(HashMap.scala:596)

at scala.collection.immutable.HashMap.getOrElse(HashMap.scala:73)

at org.apache.spark.sql.types.StructType.fieldIndex(StructType.scala:305)

at org.apache.spark.sql.catalyst.OrderedFilters.$anonfun$predicates$4(OrderedFilters.scala:61)

at org.apache.spark.sql.catalyst.OrderedFilters.$anonfun$predicates$4$adapted(OrderedFilters.scala:61)

at scala.collection.TraversableLike.$anonfun$map$1(TraversableLike.scala:286)

at scala.collection.IndexedSeqOptimized.foreach(IndexedSeqOptimized.scala:36)

at scala.collection.IndexedSeqOptimized.foreach$(IndexedSeqOptimized.scala:33)

at scala.collection.mutable.ArrayOps$ofRef.foreach(ArrayOps.scala:198)

at scala.collection.TraversableLike.map(TraversableLike.scala:286)

at scala.collection.TraversableLike.map$(TraversableLike.scala:279)

at scala.collection.mutable.ArrayOps$ofRef.map(ArrayOps.scala:198)

at org.apache.spark.sql.catalyst.OrderedFilters.$anonfun$predicates$3(OrderedFilters.scala:61)

at org.apache.spark.sql.catalyst.OrderedFilters.$anonfun$predicates$3$adapted(OrderedFilters.scala:50)

at scala.collection.immutable.List.foreach(List.scala:431)

at org.apache.spark.sql.catalyst.OrderedFilters.<init>(OrderedFilters.scala:50)

at org.apache.spark.sql.catalyst.csv.UnivocityParser.<init>(UnivocityParser.scala:103)

at org.apache.spark.sql.execution.datasources.csv.CSVFileFormat.$anonfun$buildReader$1(CSVFileFormat.scala:130)

at org.apache.spark.sql.execution.datasources.FileFormat$$anon$1.apply(FileFormat.scala:148)

at org.apache.spark.sql.execution.datasources.FileFormat$$anon$1.apply(FileFormat.scala:133)

at org.apache.spark.sql.execution.datasources.FileScanRDD$$anon$1.org$apache$spark$sql$execution$datasources$FileScanRDD$$anon$$readCurrentFile(FileScanRDD.scala:127)

at org.apache.spark.sql.execution.datasources.FileScanRDD$$anon$1.nextIterator(FileScanRDD.scala:187)

at org.apache.spark.sql.execution.datasources.FileScanRDD$$anon$1.hasNext(FileScanRDD.scala:104)

at scala.collection.Iterator$$anon$10.hasNext(Iterator.scala:460)

at org.apache.spark.sql.catalyst.expressions.GeneratedClass$GeneratedIteratorForCodegenStage1.processNext(Unknown Source)

at org.apache.spark.sql.execution.BufferedRowIterator.hasNext(BufferedRowIterator.java:43)

at org.apache.spark.sql.execution.WholeStageCodegenExec$$anon$1.hasNext(WholeStageCodegenExec.scala:759)

at org.apache.spark.sql.execution.SparkPlan.$anonfun$getByteArrayRdd$1(SparkPlan.scala:349)

at org.apache.spark.rdd.RDD.$anonfun$mapPartitionsInternal$2(RDD.scala:898)

at org.apache.spark.rdd.RDD.$anonfun$mapPartitionsInternal$2$adapted(RDD.scala:898)

at org.apache.spark.rdd.MapPartitionsRDD.compute(MapPartitionsRDD.scala:52)

at org.apache.spark.rdd.RDD.computeOrReadCheckpoint(RDD.scala:373)

at org.apache.spark.rdd.RDD.iterator(RDD.scala:337)

at org.apache.spark.scheduler.ResultTask.runTask(ResultTask.scala:90)

at org.apache.spark.scheduler.Task.run(Task.scala:131)

at org.apache.spark.executor.Executor$TaskRunner.$anonfun$run$3(Executor.scala:506)

at org.apache.spark.util.Utils$.tryWithSafeFinally(Utils.scala:1462)

at org.apache.spark.executor.Executor$TaskRunner.run(Executor.scala:509)

at java.util.concurrent.ThreadPoolExecutor.runWorker(ThreadPoolExecutor.java:1149)

at java.util.concurrent.ThreadPoolExecutor$Worker.run(ThreadPoolExecutor.java:624)

at java.lang.Thread.run(Thread.java:750)

24/07/18 20:35:26 WARN scheduler.TaskSetManager: Lost task 0.0 in stage 12.0 (TID 12) (192.168.112.129 executor driver): java.lang.IllegalArgumentException: RejectRows does not exist. Available: IssuerId, IssuerId2, BusinessDate, StateCode, SourceName, NetworkName, NetworkURL, custnum, MarketCoverage, DentalOnlyPlan

at org.apache.spark.sql.types.StructType.$anonfun$fieldIndex$1(StructType.scala:306)

at scala.collection.immutable.HashMap$HashTrieMap.getOrElse0(HashMap.scala:596)

at scala.collection.immutable.HashMap.getOrElse(HashMap.scala:73)

at org.apache.spark.sql.types.StructType.fieldIndex(StructType.scala:305)

at org.apache.spark.sql.catalyst.OrderedFilters.$anonfun$predicates$4(OrderedFilters.scala:61)

at org.apache.spark.sql.catalyst.OrderedFilters.$anonfun$predicates$4$adapted(OrderedFilters.scala:61)

at scala.collection.TraversableLike.$anonfun$map$1(TraversableLike.scala:286)

at scala.collection.IndexedSeqOptimized.foreach(IndexedSeqOptimized.scala:36)

at scala.collection.IndexedSeqOptimized.foreach$(IndexedSeqOptimized.scala:33)

at scala.collection.mutable.ArrayOps$ofRef.foreach(ArrayOps.scala:198)

at scala.collection.TraversableLike.map(TraversableLike.scala:286)

at scala.collection.TraversableLike.map$(TraversableLike.scala:279)

at scala.collection.mutable.ArrayOps$ofRef.map(ArrayOps.scala:198)

at org.apache.spark.sql.catalyst.OrderedFilters.$anonfun$predicates$3(OrderedFilters.scala:61)

at org.apache.spark.sql.catalyst.OrderedFilters.$anonfun$predicates$3$adapted(OrderedFilters.scala:50)

at scala.collection.immutable.List.foreach(List.scala:431)

at org.apache.spark.sql.catalyst.OrderedFilters.<init>(OrderedFilters.scala:50)

at org.apache.spark.sql.catalyst.csv.UnivocityParser.<init>(UnivocityParser.scala:103)

at org.apache.spark.sql.execution.datasources.csv.CSVFileFormat.$anonfun$buildReader$1(CSVFileFormat.scala:130)

at org.apache.spark.sql.execution.datasources.FileFormat$$anon$1.apply(FileFormat.scala:148)

at org.apache.spark.sql.execution.datasources.FileFormat$$anon$1.apply(FileFormat.scala:133)

at org.apache.spark.sql.execution.datasources.FileScanRDD$$anon$1.org$apache$spark$sql$execution$datasources$FileScanRDD$$anon$$readCurrentFile(FileScanRDD.scala:127)

at org.apache.spark.sql.execution.datasources.FileScanRDD$$anon$1.nextIterator(FileScanRDD.scala:187)

at org.apache.spark.sql.execution.datasources.FileScanRDD$$anon$1.hasNext(FileScanRDD.scala:104)

at scala.collection.Iterator$$anon$10.hasNext(Iterator.scala:460)

at org.apache.spark.sql.catalyst.expressions.GeneratedClass$GeneratedIteratorForCodegenStage1.processNext(Unknown Source)

at org.apache.spark.sql.execution.BufferedRowIterator.hasNext(BufferedRowIterator.java:43)

at org.apache.spark.sql.execution.WholeStageCodegenExec$$anon$1.hasNext(WholeStageCodegenExec.scala:759)

at org.apache.spark.sql.execution.SparkPlan.$anonfun$getByteArrayRdd$1(SparkPlan.scala:349)

at org.apache.spark.rdd.RDD.$anonfun$mapPartitionsInternal$2(RDD.scala:898)

at org.apache.spark.rdd.RDD.$anonfun$mapPartitionsInternal$2$adapted(RDD.scala:898)

at org.apache.spark.rdd.MapPartitionsRDD.compute(MapPartitionsRDD.scala:52)

at org.apache.spark.rdd.RDD.computeOrReadCheckpoint(RDD.scala:373)

at org.apache.spark.rdd.RDD.iterator(RDD.scala:337)

at org.apache.spark.scheduler.ResultTask.runTask(ResultTask.scala:90)

at org.apache.spark.scheduler.Task.run(Task.scala:131)

at org.apache.spark.executor.Executor$TaskRunner.$anonfun$run$3(Executor.scala:506)

at org.apache.spark.util.Utils$.tryWithSafeFinally(Utils.scala:1462)

at org.apache.spark.executor.Executor$TaskRunner.run(Executor.scala:509)

at java.util.concurrent.ThreadPoolExecutor.runWorker(ThreadPoolExecutor.java:1149)

at java.util.concurrent.ThreadPoolExecutor$Worker.run(ThreadPoolExecutor.java:624)

at java.lang.Thread.run(Thread.java:750)

24/07/18 20:35:26 ERROR scheduler.TaskSetManager: Task 0 in stage 12.0 failed 1 times; aborting job

Traceback (most recent call last):

File "<stdin>", line 1, in <module>

File "/usr/local/spark/python/pyspark/sql/dataframe.py", line 494, in show

print(self.\_jdf.showString(n, 20, vertical))

File "/usr/local/spark/python/lib/py4j-0.10.9.3-src.zip/py4j/java\_gateway.py", line 1322, in \_\_call\_\_

File "/usr/local/spark/python/pyspark/sql/utils.py", line 111, in deco

return f(\*a, \*\*kw)

File "/usr/local/spark/python/lib/py4j-0.10.9.3-src.zip/py4j/protocol.py", line 328, in get\_return\_value

py4j.protocol.Py4JJavaError: An error occurred while calling o215.showString.

: org.apache.spark.SparkException: Job aborted due to stage failure: Task 0 in stage 12.0 failed 1 times, most recent failure: Lost task 0.0 in stage 12.0 (TID 12) (192.168.112.129 executor driver): java.lang.IllegalArgumentException: RejectRows does not exist. Available: IssuerId, IssuerId2, BusinessDate, StateCode, SourceName, NetworkName, NetworkURL, custnum, MarketCoverage, DentalOnlyPlan

at org.apache.spark.sql.types.StructType.$anonfun$fieldIndex$1(StructType.scala:306)

at scala.collection.immutable.HashMap$HashTrieMap.getOrElse0(HashMap.scala:596)

at scala.collection.immutable.HashMap.getOrElse(HashMap.scala:73)

at org.apache.spark.sql.types.StructType.fieldIndex(StructType.scala:305)

at org.apache.spark.sql.catalyst.OrderedFilters.$anonfun$predicates$4(OrderedFilters.scala:61)

at org.apache.spark.sql.catalyst.OrderedFilters.$anonfun$predicates$4$adapted(OrderedFilters.scala:61)

at scala.collection.TraversableLike.$anonfun$map$1(TraversableLike.scala:286)

at scala.collection.IndexedSeqOptimized.foreach(IndexedSeqOptimized.scala:36)

at scala.collection.IndexedSeqOptimized.foreach$(IndexedSeqOptimized.scala:33)

at scala.collection.mutable.ArrayOps$ofRef.foreach(ArrayOps.scala:198)

at scala.collection.TraversableLike.map(TraversableLike.scala:286)

at scala.collection.TraversableLike.map$(TraversableLike.scala:279)

at scala.collection.mutable.ArrayOps$ofRef.map(ArrayOps.scala:198)

at org.apache.spark.sql.catalyst.OrderedFilters.$anonfun$predicates$3(OrderedFilters.scala:61)

at org.apache.spark.sql.catalyst.OrderedFilters.$anonfun$predicates$3$adapted(OrderedFilters.scala:50)

at scala.collection.immutable.List.foreach(List.scala:431)

at org.apache.spark.sql.catalyst.OrderedFilters.<init>(OrderedFilters.scala:50)

at org.apache.spark.sql.catalyst.csv.UnivocityParser.<init>(UnivocityParser.scala:103)

at org.apache.spark.sql.execution.datasources.csv.CSVFileFormat.$anonfun$buildReader$1(CSVFileFormat.scala:130)

at org.apache.spark.sql.execution.datasources.FileFormat$$anon$1.apply(FileFormat.scala:148)

at org.apache.spark.sql.execution.datasources.FileFormat$$anon$1.apply(FileFormat.scala:133)

at org.apache.spark.sql.execution.datasources.FileScanRDD$$anon$1.org$apache$spark$sql$execution$datasources$FileScanRDD$$anon$$readCurrentFile(FileScanRDD.scala:127)

at org.apache.spark.sql.execution.datasources.FileScanRDD$$anon$1.nextIterator(FileScanRDD.scala:187)

at org.apache.spark.sql.execution.datasources.FileScanRDD$$anon$1.hasNext(FileScanRDD.scala:104)

at scala.collection.Iterator$$anon$10.hasNext(Iterator.scala:460)

at org.apache.spark.sql.catalyst.expressions.GeneratedClass$GeneratedIteratorForCodegenStage1.processNext(Unknown Source)

at org.apache.spark.sql.execution.BufferedRowIterator.hasNext(BufferedRowIterator.java:43)

at org.apache.spark.sql.execution.WholeStageCodegenExec$$anon$1.hasNext(WholeStageCodegenExec.scala:759)

at org.apache.spark.sql.execution.SparkPlan.$anonfun$getByteArrayRdd$1(SparkPlan.scala:349)

at org.apache.spark.rdd.RDD.$anonfun$mapPartitionsInternal$2(RDD.scala:898)

at org.apache.spark.rdd.RDD.$anonfun$mapPartitionsInternal$2$adapted(RDD.scala:898)

at org.apache.spark.rdd.MapPartitionsRDD.compute(MapPartitionsRDD.scala:52)

at org.apache.spark.rdd.RDD.computeOrReadCheckpoint(RDD.scala:373)

at org.apache.spark.rdd.RDD.iterator(RDD.scala:337)

at org.apache.spark.scheduler.ResultTask.runTask(ResultTask.scala:90)

at org.apache.spark.scheduler.Task.run(Task.scala:131)

at org.apache.spark.executor.Executor$TaskRunner.$anonfun$run$3(Executor.scala:506)

at org.apache.spark.util.Utils$.tryWithSafeFinally(Utils.scala:1462)

at org.apache.spark.executor.Executor$TaskRunner.run(Executor.scala:509)

at java.util.concurrent.ThreadPoolExecutor.runWorker(ThreadPoolExecutor.java:1149)

at java.util.concurrent.ThreadPoolExecutor$Worker.run(ThreadPoolExecutor.java:624)

at java.lang.Thread.run(Thread.java:750)

Driver stacktrace:

at org.apache.spark.scheduler.DAGScheduler.failJobAndIndependentStages(DAGScheduler.scala:2454)

at org.apache.spark.scheduler.DAGScheduler.$anonfun$abortStage$2(DAGScheduler.scala:2403)

at org.apache.spark.scheduler.DAGScheduler.$anonfun$abortStage$2$adapted(DAGScheduler.scala:2402)

at scala.collection.mutable.ResizableArray.foreach(ResizableArray.scala:62)

at scala.collection.mutable.ResizableArray.foreach$(ResizableArray.scala:55)

at scala.collection.mutable.ArrayBuffer.foreach(ArrayBuffer.scala:49)

at org.apache.spark.scheduler.DAGScheduler.abortStage(DAGScheduler.scala:2402)

at org.apache.spark.scheduler.DAGScheduler.$anonfun$handleTaskSetFailed$1(DAGScheduler.scala:1160)

at org.apache.spark.scheduler.DAGScheduler.$anonfun$handleTaskSetFailed$1$adapted(DAGScheduler.scala:1160)

at scala.Option.foreach(Option.scala:407)

at org.apache.spark.scheduler.DAGScheduler.handleTaskSetFailed(DAGScheduler.scala:1160)

at org.apache.spark.scheduler.DAGSchedulerEventProcessLoop.doOnReceive(DAGScheduler.scala:2642)

at org.apache.spark.scheduler.DAGSchedulerEventProcessLoop.onReceive(DAGScheduler.scala:2584)

at org.apache.spark.scheduler.DAGSchedulerEventProcessLoop.onReceive(DAGScheduler.scala:2573)

at org.apache.spark.util.EventLoop$$anon$1.run(EventLoop.scala:49)

at org.apache.spark.scheduler.DAGScheduler.runJob(DAGScheduler.scala:938)

at org.apache.spark.SparkContext.runJob(SparkContext.scala:2214)

at org.apache.spark.SparkContext.runJob(SparkContext.scala:2235)

at org.apache.spark.SparkContext.runJob(SparkContext.scala:2254)

at org.apache.spark.sql.execution.SparkPlan.executeTake(SparkPlan.scala:476)

at org.apache.spark.sql.execution.SparkPlan.executeTake(SparkPlan.scala:429)

at org.apache.spark.sql.execution.CollectLimitExec.executeCollect(limit.scala:48)

at org.apache.spark.sql.Dataset.collectFromPlan(Dataset.scala:3715)

at org.apache.spark.sql.Dataset.$anonfun$head$1(Dataset.scala:2728)

at org.apache.spark.sql.Dataset.$anonfun$withAction$1(Dataset.scala:3706)

at org.apache.spark.sql.execution.SQLExecution$.$anonfun$withNewExecutionId$5(SQLExecution.scala:103)

at org.apache.spark.sql.execution.SQLExecution$.withSQLConfPropagated(SQLExecution.scala:163)

at org.apache.spark.sql.execution.SQLExecution$.$anonfun$withNewExecutionId$1(SQLExecution.scala:90)

at org.apache.spark.sql.SparkSession.withActive(SparkSession.scala:775)

at org.apache.spark.sql.execution.SQLExecution$.withNewExecutionId(SQLExecution.scala:64)

at org.apache.spark.sql.Dataset.withAction(Dataset.scala:3704)

at org.apache.spark.sql.Dataset.head(Dataset.scala:2728)

at org.apache.spark.sql.Dataset.take(Dataset.scala:2935)

at org.apache.spark.sql.Dataset.getRows(Dataset.scala:287)

at org.apache.spark.sql.Dataset.showString(Dataset.scala:326)

at sun.reflect.GeneratedMethodAccessor41.invoke(Unknown Source)

at sun.reflect.DelegatingMethodAccessorImpl.invoke(DelegatingMethodAccessorImpl.java:43)

at java.lang.reflect.Method.invoke(Method.java:498)

at py4j.reflection.MethodInvoker.invoke(MethodInvoker.java:244)

at py4j.reflection.ReflectionEngine.invoke(ReflectionEngine.java:357)

at py4j.Gateway.invoke(Gateway.java:282)

at py4j.commands.AbstractCommand.invokeMethod(AbstractCommand.java:132)

at py4j.commands.CallCommand.execute(CallCommand.java:79)

at py4j.ClientServerConnection.waitForCommands(ClientServerConnection.java:182)

at py4j.ClientServerConnection.run(ClientServerConnection.java:106)

at java.lang.Thread.run(Thread.java:750)

Caused by: java.lang.IllegalArgumentException: RejectRows does not exist. Available: IssuerId, IssuerId2, BusinessDate, StateCode, SourceName, NetworkName, NetworkURL, custnum, MarketCoverage, DentalOnlyPlan

at org.apache.spark.sql.types.StructType.$anonfun$fieldIndex$1(StructType.scala:306)

at scala.collection.immutable.HashMap$HashTrieMap.getOrElse0(HashMap.scala:596)

at scala.collection.immutable.HashMap.getOrElse(HashMap.scala:73)

at org.apache.spark.sql.types.StructType.fieldIndex(StructType.scala:305)

at org.apache.spark.sql.catalyst.OrderedFilters.$anonfun$predicates$4(OrderedFilters.scala:61)

at org.apache.spark.sql.catalyst.OrderedFilters.$anonfun$predicates$4$adapted(OrderedFilters.scala:61)

at scala.collection.TraversableLike.$anonfun$map$1(TraversableLike.scala:286)

at scala.collection.IndexedSeqOptimized.foreach(IndexedSeqOptimized.scala:36)

at scala.collection.IndexedSeqOptimized.foreach$(IndexedSeqOptimized.scala:33)

at scala.collection.mutable.ArrayOps$ofRef.foreach(ArrayOps.scala:198)

at scala.collection.TraversableLike.map(TraversableLike.scala:286)

at scala.collection.TraversableLike.map$(TraversableLike.scala:279)

at scala.collection.mutable.ArrayOps$ofRef.map(ArrayOps.scala:198)

at org.apache.spark.sql.catalyst.OrderedFilters.$anonfun$predicates$3(OrderedFilters.scala:61)

at org.apache.spark.sql.catalyst.OrderedFilters.$anonfun$predicates$3$adapted(OrderedFilters.scala:50)

at scala.collection.immutable.List.foreach(List.scala:431)

at org.apache.spark.sql.catalyst.OrderedFilters.<init>(OrderedFilters.scala:50)

at org.apache.spark.sql.catalyst.csv.UnivocityParser.<init>(UnivocityParser.scala:103)

at org.apache.spark.sql.execution.datasources.csv.CSVFileFormat.$anonfun$buildReader$1(CSVFileFormat.scala:130)

at org.apache.spark.sql.execution.datasources.FileFormat$$anon$1.apply(FileFormat.scala:148)

at org.apache.spark.sql.execution.datasources.FileFormat$$anon$1.apply(FileFormat.scala:133)

at org.apache.spark.sql.execution.datasources.FileScanRDD$$anon$1.org$apache$spark$sql$execution$datasources$FileScanRDD$$anon$$readCurrentFile(FileScanRDD.scala:127)

at org.apache.spark.sql.execution.datasources.FileScanRDD$$anon$1.nextIterator(FileScanRDD.scala:187)

at org.apache.spark.sql.execution.datasources.FileScanRDD$$anon$1.hasNext(FileScanRDD.scala:104)

at scala.collection.Iterator$$anon$10.hasNext(Iterator.scala:460)

at org.apache.spark.sql.catalyst.expressions.GeneratedClass$GeneratedIteratorForCodegenStage1.processNext(Unknown Source)

at org.apache.spark.sql.execution.BufferedRowIterator.hasNext(BufferedRowIterator.java:43)

at org.apache.spark.sql.execution.WholeStageCodegenExec$$anon$1.hasNext(WholeStageCodegenExec.scala:759)

at org.apache.spark.sql.execution.SparkPlan.$anonfun$getByteArrayRdd$1(SparkPlan.scala:349)

at org.apache.spark.rdd.RDD.$anonfun$mapPartitionsInternal$2(RDD.scala:898)

at org.apache.spark.rdd.RDD.$anonfun$mapPartitionsInternal$2$adapted(RDD.scala:898)

at org.apache.spark.rdd.MapPartitionsRDD.compute(MapPartitionsRDD.scala:52)

at org.apache.spark.rdd.RDD.computeOrReadCheckpoint(RDD.scala:373)

at org.apache.spark.rdd.RDD.iterator(RDD.scala:337)

at org.apache.spark.scheduler.ResultTask.runTask(ResultTask.scala:90)

at org.apache.spark.scheduler.Task.run(Task.scala:131)

at org.apache.spark.executor.Executor$TaskRunner.$anonfun$run$3(Executor.scala:506)

at org.apache.spark.util.Utils$.tryWithSafeFinally(Utils.scala:1462)

at org.apache.spark.executor.Executor$TaskRunner.run(Executor.scala:509)

at java.util.concurrent.ThreadPoolExecutor.runWorker(ThreadPoolExecutor.java:1149)

at java.util.concurrent.ThreadPoolExecutor$Worker.run(ThreadPoolExecutor.java:624)

... 1 more

>>> data\_frame\_1.where("RejectRows='null'").show(3)

24/07/18 20:35:31 ERROR executor.Executor: Exception in task 0.0 in stage 13.0 (TID 13)

java.lang.IllegalArgumentException: RejectRows does not exist. Available: IssuerId, IssuerId2, BusinessDate, StateCode, SourceName, NetworkName, NetworkURL, custnum, MarketCoverage, DentalOnlyPlan

at org.apache.spark.sql.types.StructType.$anonfun$fieldIndex$1(StructType.scala:306)

at scala.collection.immutable.HashMap$HashTrieMap.getOrElse0(HashMap.scala:596)

at scala.collection.immutable.HashMap.getOrElse(HashMap.scala:73)

at org.apache.spark.sql.types.StructType.fieldIndex(StructType.scala:305)

at org.apache.spark.sql.catalyst.OrderedFilters.$anonfun$predicates$4(OrderedFilters.scala:61)

at org.apache.spark.sql.catalyst.OrderedFilters.$anonfun$predicates$4$adapted(OrderedFilters.scala:61)

at scala.collection.TraversableLike.$anonfun$map$1(TraversableLike.scala:286)

at scala.collection.IndexedSeqOptimized.foreach(IndexedSeqOptimized.scala:36)

at scala.collection.IndexedSeqOptimized.foreach$(IndexedSeqOptimized.scala:33)

at scala.collection.mutable.ArrayOps$ofRef.foreach(ArrayOps.scala:198)

at scala.collection.TraversableLike.map(TraversableLike.scala:286)

at scala.collection.TraversableLike.map$(TraversableLike.scala:279)

at scala.collection.mutable.ArrayOps$ofRef.map(ArrayOps.scala:198)

at org.apache.spark.sql.catalyst.OrderedFilters.$anonfun$predicates$3(OrderedFilters.scala:61)

at org.apache.spark.sql.catalyst.OrderedFilters.$anonfun$predicates$3$adapted(OrderedFilters.scala:50)

at scala.collection.immutable.List.foreach(List.scala:431)

at org.apache.spark.sql.catalyst.OrderedFilters.<init>(OrderedFilters.scala:50)

at org.apache.spark.sql.catalyst.csv.UnivocityParser.<init>(UnivocityParser.scala:103)

at org.apache.spark.sql.execution.datasources.csv.CSVFileFormat.$anonfun$buildReader$1(CSVFileFormat.scala:130)

at org.apache.spark.sql.execution.datasources.FileFormat$$anon$1.apply(FileFormat.scala:148)

at org.apache.spark.sql.execution.datasources.FileFormat$$anon$1.apply(FileFormat.scala:133)

at org.apache.spark.sql.execution.datasources.FileScanRDD$$anon$1.org$apache$spark$sql$execution$datasources$FileScanRDD$$anon$$readCurrentFile(FileScanRDD.scala:127)

at org.apache.spark.sql.execution.datasources.FileScanRDD$$anon$1.nextIterator(FileScanRDD.scala:187)

at org.apache.spark.sql.execution.datasources.FileScanRDD$$anon$1.hasNext(FileScanRDD.scala:104)

at scala.collection.Iterator$$anon$10.hasNext(Iterator.scala:460)

at org.apache.spark.sql.catalyst.expressions.GeneratedClass$GeneratedIteratorForCodegenStage1.processNext(Unknown Source)

at org.apache.spark.sql.execution.BufferedRowIterator.hasNext(BufferedRowIterator.java:43)

at org.apache.spark.sql.execution.WholeStageCodegenExec$$anon$1.hasNext(WholeStageCodegenExec.scala:759)

at org.apache.spark.sql.execution.SparkPlan.$anonfun$getByteArrayRdd$1(SparkPlan.scala:349)

at org.apache.spark.rdd.RDD.$anonfun$mapPartitionsInternal$2(RDD.scala:898)

at org.apache.spark.rdd.RDD.$anonfun$mapPartitionsInternal$2$adapted(RDD.scala:898)

at org.apache.spark.rdd.MapPartitionsRDD.compute(MapPartitionsRDD.scala:52)

at org.apache.spark.rdd.RDD.computeOrReadCheckpoint(RDD.scala:373)

at org.apache.spark.rdd.RDD.iterator(RDD.scala:337)

at org.apache.spark.scheduler.ResultTask.runTask(ResultTask.scala:90)

at org.apache.spark.scheduler.Task.run(Task.scala:131)

at org.apache.spark.executor.Executor$TaskRunner.$anonfun$run$3(Executor.scala:506)

at org.apache.spark.util.Utils$.tryWithSafeFinally(Utils.scala:1462)

at org.apache.spark.executor.Executor$TaskRunner.run(Executor.scala:509)

at java.util.concurrent.ThreadPoolExecutor.runWorker(ThreadPoolExecutor.java:1149)

at java.util.concurrent.ThreadPoolExecutor$Worker.run(ThreadPoolExecutor.java:624)

at java.lang.Thread.run(Thread.java:750)

24/07/18 20:35:31 WARN scheduler.TaskSetManager: Lost task 0.0 in stage 13.0 (TID 13) (192.168.112.129 executor driver): java.lang.IllegalArgumentException: RejectRows does not exist. Available: IssuerId, IssuerId2, BusinessDate, StateCode, SourceName, NetworkName, NetworkURL, custnum, MarketCoverage, DentalOnlyPlan

at org.apache.spark.sql.types.StructType.$anonfun$fieldIndex$1(StructType.scala:306)

at scala.collection.immutable.HashMap$HashTrieMap.getOrElse0(HashMap.scala:596)

at scala.collection.immutable.HashMap.getOrElse(HashMap.scala:73)

at org.apache.spark.sql.types.StructType.fieldIndex(StructType.scala:305)

at org.apache.spark.sql.catalyst.OrderedFilters.$anonfun$predicates$4(OrderedFilters.scala:61)

at org.apache.spark.sql.catalyst.OrderedFilters.$anonfun$predicates$4$adapted(OrderedFilters.scala:61)

at scala.collection.TraversableLike.$anonfun$map$1(TraversableLike.scala:286)

at scala.collection.IndexedSeqOptimized.foreach(IndexedSeqOptimized.scala:36)

at scala.collection.IndexedSeqOptimized.foreach$(IndexedSeqOptimized.scala:33)

at scala.collection.mutable.ArrayOps$ofRef.foreach(ArrayOps.scala:198)

at scala.collection.TraversableLike.map(TraversableLike.scala:286)

at scala.collection.TraversableLike.map$(TraversableLike.scala:279)

at scala.collection.mutable.ArrayOps$ofRef.map(ArrayOps.scala:198)

at org.apache.spark.sql.catalyst.OrderedFilters.$anonfun$predicates$3(OrderedFilters.scala:61)

at org.apache.spark.sql.catalyst.OrderedFilters.$anonfun$predicates$3$adapted(OrderedFilters.scala:50)

at scala.collection.immutable.List.foreach(List.scala:431)

at org.apache.spark.sql.catalyst.OrderedFilters.<init>(OrderedFilters.scala:50)

at org.apache.spark.sql.catalyst.csv.UnivocityParser.<init>(UnivocityParser.scala:103)

at org.apache.spark.sql.execution.datasources.csv.CSVFileFormat.$anonfun$buildReader$1(CSVFileFormat.scala:130)

at org.apache.spark.sql.execution.datasources.FileFormat$$anon$1.apply(FileFormat.scala:148)

at org.apache.spark.sql.execution.datasources.FileFormat$$anon$1.apply(FileFormat.scala:133)

at org.apache.spark.sql.execution.datasources.FileScanRDD$$anon$1.org$apache$spark$sql$execution$datasources$FileScanRDD$$anon$$readCurrentFile(FileScanRDD.scala:127)

at org.apache.spark.sql.execution.datasources.FileScanRDD$$anon$1.nextIterator(FileScanRDD.scala:187)

at org.apache.spark.sql.execution.datasources.FileScanRDD$$anon$1.hasNext(FileScanRDD.scala:104)

at scala.collection.Iterator$$anon$10.hasNext(Iterator.scala:460)

at org.apache.spark.sql.catalyst.expressions.GeneratedClass$GeneratedIteratorForCodegenStage1.processNext(Unknown Source)

at org.apache.spark.sql.execution.BufferedRowIterator.hasNext(BufferedRowIterator.java:43)

at org.apache.spark.sql.execution.WholeStageCodegenExec$$anon$1.hasNext(WholeStageCodegenExec.scala:759)

at org.apache.spark.sql.execution.SparkPlan.$anonfun$getByteArrayRdd$1(SparkPlan.scala:349)

at org.apache.spark.rdd.RDD.$anonfun$mapPartitionsInternal$2(RDD.scala:898)

at org.apache.spark.rdd.RDD.$anonfun$mapPartitionsInternal$2$adapted(RDD.scala:898)

at org.apache.spark.rdd.MapPartitionsRDD.compute(MapPartitionsRDD.scala:52)

at org.apache.spark.rdd.RDD.computeOrReadCheckpoint(RDD.scala:373)

at org.apache.spark.rdd.RDD.iterator(RDD.scala:337)

at org.apache.spark.scheduler.ResultTask.runTask(ResultTask.scala:90)

at org.apache.spark.scheduler.Task.run(Task.scala:131)

at org.apache.spark.executor.Executor$TaskRunner.$anonfun$run$3(Executor.scala:506)

at org.apache.spark.util.Utils$.tryWithSafeFinally(Utils.scala:1462)

at org.apache.spark.executor.Executor$TaskRunner.run(Executor.scala:509)

at java.util.concurrent.ThreadPoolExecutor.runWorker(ThreadPoolExecutor.java:1149)

at java.util.concurrent.ThreadPoolExecutor$Worker.run(ThreadPoolExecutor.java:624)

at java.lang.Thread.run(Thread.java:750)

24/07/18 20:35:31 ERROR scheduler.TaskSetManager: Task 0 in stage 13.0 failed 1 times; aborting job

Traceback (most recent call last):

File "<stdin>", line 1, in <module>

File "/usr/local/spark/python/pyspark/sql/dataframe.py", line 494, in show

print(self.\_jdf.showString(n, 20, vertical))

File "/usr/local/spark/python/lib/py4j-0.10.9.3-src.zip/py4j/java\_gateway.py", line 1322, in \_\_call\_\_

File "/usr/local/spark/python/pyspark/sql/utils.py", line 111, in deco

return f(\*a, \*\*kw)

File "/usr/local/spark/python/lib/py4j-0.10.9.3-src.zip/py4j/protocol.py", line 328, in get\_return\_value

py4j.protocol.Py4JJavaError: An error occurred while calling o218.showString.

: org.apache.spark.SparkException: Job aborted due to stage failure: Task 0 in stage 13.0 failed 1 times, most recent failure: Lost task 0.0 in stage 13.0 (TID 13) (192.168.112.129 executor driver): java.lang.IllegalArgumentException: RejectRows does not exist. Available: IssuerId, IssuerId2, BusinessDate, StateCode, SourceName, NetworkName, NetworkURL, custnum, MarketCoverage, DentalOnlyPlan

at org.apache.spark.sql.types.StructType.$anonfun$fieldIndex$1(StructType.scala:306)

at scala.collection.immutable.HashMap$HashTrieMap.getOrElse0(HashMap.scala:596)

at scala.collection.immutable.HashMap.getOrElse(HashMap.scala:73)

at org.apache.spark.sql.types.StructType.fieldIndex(StructType.scala:305)

at org.apache.spark.sql.catalyst.OrderedFilters.$anonfun$predicates$4(OrderedFilters.scala:61)

at org.apache.spark.sql.catalyst.OrderedFilters.$anonfun$predicates$4$adapted(OrderedFilters.scala:61)

at scala.collection.TraversableLike.$anonfun$map$1(TraversableLike.scala:286)

at scala.collection.IndexedSeqOptimized.foreach(IndexedSeqOptimized.scala:36)

at scala.collection.IndexedSeqOptimized.foreach$(IndexedSeqOptimized.scala:33)

at scala.collection.mutable.ArrayOps$ofRef.foreach(ArrayOps.scala:198)

at scala.collection.TraversableLike.map(TraversableLike.scala:286)

at scala.collection.TraversableLike.map$(TraversableLike.scala:279)

at scala.collection.mutable.ArrayOps$ofRef.map(ArrayOps.scala:198)

at org.apache.spark.sql.catalyst.OrderedFilters.$anonfun$predicates$3(OrderedFilters.scala:61)

at org.apache.spark.sql.catalyst.OrderedFilters.$anonfun$predicates$3$adapted(OrderedFilters.scala:50)

at scala.collection.immutable.List.foreach(List.scala:431)

at org.apache.spark.sql.catalyst.OrderedFilters.<init>(OrderedFilters.scala:50)

at org.apache.spark.sql.catalyst.csv.UnivocityParser.<init>(UnivocityParser.scala:103)

at org.apache.spark.sql.execution.datasources.csv.CSVFileFormat.$anonfun$buildReader$1(CSVFileFormat.scala:130)

at org.apache.spark.sql.execution.datasources.FileFormat$$anon$1.apply(FileFormat.scala:148)

at org.apache.spark.sql.execution.datasources.FileFormat$$anon$1.apply(FileFormat.scala:133)

at org.apache.spark.sql.execution.datasources.FileScanRDD$$anon$1.org$apache$spark$sql$execution$datasources$FileScanRDD$$anon$$readCurrentFile(FileScanRDD.scala:127)

at org.apache.spark.sql.execution.datasources.FileScanRDD$$anon$1.nextIterator(FileScanRDD.scala:187)

at org.apache.spark.sql.execution.datasources.FileScanRDD$$anon$1.hasNext(FileScanRDD.scala:104)

at scala.collection.Iterator$$anon$10.hasNext(Iterator.scala:460)

at org.apache.spark.sql.catalyst.expressions.GeneratedClass$GeneratedIteratorForCodegenStage1.processNext(Unknown Source)

at org.apache.spark.sql.execution.BufferedRowIterator.hasNext(BufferedRowIterator.java:43)

at org.apache.spark.sql.execution.WholeStageCodegenExec$$anon$1.hasNext(WholeStageCodegenExec.scala:759)

at org.apache.spark.sql.execution.SparkPlan.$anonfun$getByteArrayRdd$1(SparkPlan.scala:349)

at org.apache.spark.rdd.RDD.$anonfun$mapPartitionsInternal$2(RDD.scala:898)

at org.apache.spark.rdd.RDD.$anonfun$mapPartitionsInternal$2$adapted(RDD.scala:898)

at org.apache.spark.rdd.MapPartitionsRDD.compute(MapPartitionsRDD.scala:52)

at org.apache.spark.rdd.RDD.computeOrReadCheckpoint(RDD.scala:373)

at org.apache.spark.rdd.RDD.iterator(RDD.scala:337)

at org.apache.spark.scheduler.ResultTask.runTask(ResultTask.scala:90)

at org.apache.spark.scheduler.Task.run(Task.scala:131)

at org.apache.spark.executor.Executor$TaskRunner.$anonfun$run$3(Executor.scala:506)

at org.apache.spark.util.Utils$.tryWithSafeFinally(Utils.scala:1462)

at org.apache.spark.executor.Executor$TaskRunner.run(Executor.scala:509)

at java.util.concurrent.ThreadPoolExecutor.runWorker(ThreadPoolExecutor.java:1149)

at java.util.concurrent.ThreadPoolExecutor$Worker.run(ThreadPoolExecutor.java:624)

at java.lang.Thread.run(Thread.java:750)

Driver stacktrace:

at org.apache.spark.scheduler.DAGScheduler.failJobAndIndependentStages(DAGScheduler.scala:2454)

at org.apache.spark.scheduler.DAGScheduler.$anonfun$abortStage$2(DAGScheduler.scala:2403)

at org.apache.spark.scheduler.DAGScheduler.$anonfun$abortStage$2$adapted(DAGScheduler.scala:2402)

at scala.collection.mutable.ResizableArray.foreach(ResizableArray.scala:62)

at scala.collection.mutable.ResizableArray.foreach$(ResizableArray.scala:55)

at scala.collection.mutable.ArrayBuffer.foreach(ArrayBuffer.scala:49)

at org.apache.spark.scheduler.DAGScheduler.abortStage(DAGScheduler.scala:2402)

at org.apache.spark.scheduler.DAGScheduler.$anonfun$handleTaskSetFailed$1(DAGScheduler.scala:1160)

at org.apache.spark.scheduler.DAGScheduler.$anonfun$handleTaskSetFailed$1$adapted(DAGScheduler.scala:1160)

at scala.Option.foreach(Option.scala:407)

at org.apache.spark.scheduler.DAGScheduler.handleTaskSetFailed(DAGScheduler.scala:1160)

at org.apache.spark.scheduler.DAGSchedulerEventProcessLoop.doOnReceive(DAGScheduler.scala:2642)

at org.apache.spark.scheduler.DAGSchedulerEventProcessLoop.onReceive(DAGScheduler.scala:2584)

at org.apache.spark.scheduler.DAGSchedulerEventProcessLoop.onReceive(DAGScheduler.scala:2573)

at org.apache.spark.util.EventLoop$$anon$1.run(EventLoop.scala:49)

at org.apache.spark.scheduler.DAGScheduler.runJob(DAGScheduler.scala:938)

at org.apache.spark.SparkContext.runJob(SparkContext.scala:2214)

at org.apache.spark.SparkContext.runJob(SparkContext.scala:2235)

at org.apache.spark.SparkContext.runJob(SparkContext.scala:2254)

at org.apache.spark.sql.execution.SparkPlan.executeTake(SparkPlan.scala:476)

at org.apache.spark.sql.execution.SparkPlan.executeTake(SparkPlan.scala:429)

at org.apache.spark.sql.execution.CollectLimitExec.executeCollect(limit.scala:48)

at org.apache.spark.sql.Dataset.collectFromPlan(Dataset.scala:3715)

at org.apache.spark.sql.Dataset.$anonfun$head$1(Dataset.scala:2728)

at org.apache.spark.sql.Dataset.$anonfun$withAction$1(Dataset.scala:3706)

at org.apache.spark.sql.execution.SQLExecution$.$anonfun$withNewExecutionId$5(SQLExecution.scala:103)

at org.apache.spark.sql.execution.SQLExecution$.withSQLConfPropagated(SQLExecution.scala:163)

at org.apache.spark.sql.execution.SQLExecution$.$anonfun$withNewExecutionId$1(SQLExecution.scala:90)

at org.apache.spark.sql.SparkSession.withActive(SparkSession.scala:775)

at org.apache.spark.sql.execution.SQLExecution$.withNewExecutionId(SQLExecution.scala:64)

at org.apache.spark.sql.Dataset.withAction(Dataset.scala:3704)

at org.apache.spark.sql.Dataset.head(Dataset.scala:2728)

at org.apache.spark.sql.Dataset.take(Dataset.scala:2935)

at org.apache.spark.sql.Dataset.getRows(Dataset.scala:287)

at org.apache.spark.sql.Dataset.showString(Dataset.scala:326)

at sun.reflect.GeneratedMethodAccessor41.invoke(Unknown Source)

at sun.reflect.DelegatingMethodAccessorImpl.invoke(DelegatingMethodAccessorImpl.java:43)

at java.lang.reflect.Method.invoke(Method.java:498)

at py4j.reflection.MethodInvoker.invoke(MethodInvoker.java:244)

at py4j.reflection.ReflectionEngine.invoke(ReflectionEngine.java:357)

at py4j.Gateway.invoke(Gateway.java:282)

at py4j.commands.AbstractCommand.invokeMethod(AbstractCommand.java:132)

at py4j.commands.CallCommand.execute(CallCommand.java:79)

at py4j.ClientServerConnection.waitForCommands(ClientServerConnection.java:182)

at py4j.ClientServerConnection.run(ClientServerConnection.java:106)

at java.lang.Thread.run(Thread.java:750)

Caused by: java.lang.IllegalArgumentException: RejectRows does not exist. Available: IssuerId, IssuerId2, BusinessDate, StateCode, SourceName, NetworkName, NetworkURL, custnum, MarketCoverage, DentalOnlyPlan

at org.apache.spark.sql.types.StructType.$anonfun$fieldIndex$1(StructType.scala:306)

at scala.collection.immutable.HashMap$HashTrieMap.getOrElse0(HashMap.scala:596)

at scala.collection.immutable.HashMap.getOrElse(HashMap.scala:73)

at org.apache.spark.sql.types.StructType.fieldIndex(StructType.scala:305)

at org.apache.spark.sql.catalyst.OrderedFilters.$anonfun$predicates$4(OrderedFilters.scala:61)

at org.apache.spark.sql.catalyst.OrderedFilters.$anonfun$predicates$4$adapted(OrderedFilters.scala:61)

at scala.collection.TraversableLike.$anonfun$map$1(TraversableLike.scala:286)

at scala.collection.IndexedSeqOptimized.foreach(IndexedSeqOptimized.scala:36)

at scala.collection.IndexedSeqOptimized.foreach$(IndexedSeqOptimized.scala:33)

at scala.collection.mutable.ArrayOps$ofRef.foreach(ArrayOps.scala:198)

at scala.collection.TraversableLike.map(TraversableLike.scala:286)

at scala.collection.TraversableLike.map$(TraversableLike.scala:279)

at scala.collection.mutable.ArrayOps$ofRef.map(ArrayOps.scala:198)

at org.apache.spark.sql.catalyst.OrderedFilters.$anonfun$predicates$3(OrderedFilters.scala:61)

at org.apache.spark.sql.catalyst.OrderedFilters.$anonfun$predicates$3$adapted(OrderedFilters.scala:50)

at scala.collection.immutable.List.foreach(List.scala:431)

at org.apache.spark.sql.catalyst.OrderedFilters.<init>(OrderedFilters.scala:50)

at org.apache.spark.sql.catalyst.csv.UnivocityParser.<init>(UnivocityParser.scala:103)

at org.apache.spark.sql.execution.datasources.csv.CSVFileFormat.$anonfun$buildReader$1(CSVFileFormat.scala:130)

at org.apache.spark.sql.execution.datasources.FileFormat$$anon$1.apply(FileFormat.scala:148)

at org.apache.spark.sql.execution.datasources.FileFormat$$anon$1.apply(FileFormat.scala:133)

at org.apache.spark.sql.execution.datasources.FileScanRDD$$anon$1.org$apache$spark$sql$execution$datasources$FileScanRDD$$anon$$readCurrentFile(FileScanRDD.scala:127)

at org.apache.spark.sql.execution.datasources.FileScanRDD$$anon$1.nextIterator(FileScanRDD.scala:187)

at org.apache.spark.sql.execution.datasources.FileScanRDD$$anon$1.hasNext(FileScanRDD.scala:104)

at scala.collection.Iterator$$anon$10.hasNext(Iterator.scala:460)

at org.apache.spark.sql.catalyst.expressions.GeneratedClass$GeneratedIteratorForCodegenStage1.processNext(Unknown Source)

at org.apache.spark.sql.execution.BufferedRowIterator.hasNext(BufferedRowIterator.java:43)

at org.apache.spark.sql.execution.WholeStageCodegenExec$$anon$1.hasNext(WholeStageCodegenExec.scala:759)

at org.apache.spark.sql.execution.SparkPlan.$anonfun$getByteArrayRdd$1(SparkPlan.scala:349)

at org.apache.spark.rdd.RDD.$anonfun$mapPartitionsInternal$2(RDD.scala:898)

at org.apache.spark.rdd.RDD.$anonfun$mapPartitionsInternal$2$adapted(RDD.scala:898)

at org.apache.spark.rdd.MapPartitionsRDD.compute(MapPartitionsRDD.scala:52)

at org.apache.spark.rdd.RDD.computeOrReadCheckpoint(RDD.scala:373)

at org.apache.spark.rdd.RDD.iterator(RDD.scala:337)

at org.apache.spark.scheduler.ResultTask.runTask(ResultTask.scala:90)

at org.apache.spark.scheduler.Task.run(Task.scala:131)

at org.apache.spark.executor.Executor$TaskRunner.$anonfun$run$3(Executor.scala:506)

at org.apache.spark.util.Utils$.tryWithSafeFinally(Utils.scala:1462)

at org.apache.spark.executor.Executor$TaskRunner.run(Executor.scala:509)

at java.util.concurrent.ThreadPoolExecutor.runWorker(ThreadPoolExecutor.java:1149)

at java.util.concurrent.ThreadPoolExecutor$Worker.run(ThreadPoolExecutor.java:624)

... 1 more

>>> data\_frame\_1.where("RejectRows!='null'").show(3)

24/07/18 20:35:37 ERROR executor.Executor: Exception in task 0.0 in stage 14.0 (TID 14)

java.lang.IllegalArgumentException: RejectRows does not exist. Available: IssuerId, IssuerId2, BusinessDate, StateCode, SourceName, NetworkName, NetworkURL, custnum, MarketCoverage, DentalOnlyPlan

at org.apache.spark.sql.types.StructType.$anonfun$fieldIndex$1(StructType.scala:306)

at scala.collection.immutable.HashMap$HashTrieMap.getOrElse0(HashMap.scala:596)

at scala.collection.immutable.HashMap.getOrElse(HashMap.scala:73)

at org.apache.spark.sql.types.StructType.fieldIndex(StructType.scala:305)

at org.apache.spark.sql.catalyst.OrderedFilters.$anonfun$predicates$4(OrderedFilters.scala:61)

at org.apache.spark.sql.catalyst.OrderedFilters.$anonfun$predicates$4$adapted(OrderedFilters.scala:61)

at scala.collection.TraversableLike.$anonfun$map$1(TraversableLike.scala:286)

at scala.collection.IndexedSeqOptimized.foreach(IndexedSeqOptimized.scala:36)

at scala.collection.IndexedSeqOptimized.foreach$(IndexedSeqOptimized.scala:33)

at scala.collection.mutable.ArrayOps$ofRef.foreach(ArrayOps.scala:198)

at scala.collection.TraversableLike.map(TraversableLike.scala:286)

at scala.collection.TraversableLike.map$(TraversableLike.scala:279)

at scala.collection.mutable.ArrayOps$ofRef.map(ArrayOps.scala:198)

at org.apache.spark.sql.catalyst.OrderedFilters.$anonfun$predicates$3(OrderedFilters.scala:61)

at org.apache.spark.sql.catalyst.OrderedFilters.$anonfun$predicates$3$adapted(OrderedFilters.scala:50)

at scala.collection.immutable.List.foreach(List.scala:431)

at org.apache.spark.sql.catalyst.OrderedFilters.<init>(OrderedFilters.scala:50)

at org.apache.spark.sql.catalyst.csv.UnivocityParser.<init>(UnivocityParser.scala:103)

at org.apache.spark.sql.execution.datasources.csv.CSVFileFormat.$anonfun$buildReader$1(CSVFileFormat.scala:130)

at org.apache.spark.sql.execution.datasources.FileFormat$$anon$1.apply(FileFormat.scala:148)

at org.apache.spark.sql.execution.datasources.FileFormat$$anon$1.apply(FileFormat.scala:133)

at org.apache.spark.sql.execution.datasources.FileScanRDD$$anon$1.org$apache$spark$sql$execution$datasources$FileScanRDD$$anon$$readCurrentFile(FileScanRDD.scala:127)

at org.apache.spark.sql.execution.datasources.FileScanRDD$$anon$1.nextIterator(FileScanRDD.scala:187)

at org.apache.spark.sql.execution.datasources.FileScanRDD$$anon$1.hasNext(FileScanRDD.scala:104)

at scala.collection.Iterator$$anon$10.hasNext(Iterator.scala:460)

at org.apache.spark.sql.catalyst.expressions.GeneratedClass$GeneratedIteratorForCodegenStage1.processNext(Unknown Source)

at org.apache.spark.sql.execution.BufferedRowIterator.hasNext(BufferedRowIterator.java:43)

at org.apache.spark.sql.execution.WholeStageCodegenExec$$anon$1.hasNext(WholeStageCodegenExec.scala:759)

at org.apache.spark.sql.execution.SparkPlan.$anonfun$getByteArrayRdd$1(SparkPlan.scala:349)

at org.apache.spark.rdd.RDD.$anonfun$mapPartitionsInternal$2(RDD.scala:898)

at org.apache.spark.rdd.RDD.$anonfun$mapPartitionsInternal$2$adapted(RDD.scala:898)

at org.apache.spark.rdd.MapPartitionsRDD.compute(MapPartitionsRDD.scala:52)

at org.apache.spark.rdd.RDD.computeOrReadCheckpoint(RDD.scala:373)

at org.apache.spark.rdd.RDD.iterator(RDD.scala:337)

at org.apache.spark.scheduler.ResultTask.runTask(ResultTask.scala:90)

at org.apache.spark.scheduler.Task.run(Task.scala:131)

at org.apache.spark.executor.Executor$TaskRunner.$anonfun$run$3(Executor.scala:506)

at org.apache.spark.util.Utils$.tryWithSafeFinally(Utils.scala:1462)

at org.apache.spark.executor.Executor$TaskRunner.run(Executor.scala:509)

at java.util.concurrent.ThreadPoolExecutor.runWorker(ThreadPoolExecutor.java:1149)

at java.util.concurrent.ThreadPoolExecutor$Worker.run(ThreadPoolExecutor.java:624)

at java.lang.Thread.run(Thread.java:750)

24/07/18 20:35:37 WARN scheduler.TaskSetManager: Lost task 0.0 in stage 14.0 (TID 14) (192.168.112.129 executor driver): java.lang.IllegalArgumentException: RejectRows does not exist. Available: IssuerId, IssuerId2, BusinessDate, StateCode, SourceName, NetworkName, NetworkURL, custnum, MarketCoverage, DentalOnlyPlan

at org.apache.spark.sql.types.StructType.$anonfun$fieldIndex$1(StructType.scala:306)

at scala.collection.immutable.HashMap$HashTrieMap.getOrElse0(HashMap.scala:596)

at scala.collection.immutable.HashMap.getOrElse(HashMap.scala:73)

at org.apache.spark.sql.types.StructType.fieldIndex(StructType.scala:305)

at org.apache.spark.sql.catalyst.OrderedFilters.$anonfun$predicates$4(OrderedFilters.scala:61)

at org.apache.spark.sql.catalyst.OrderedFilters.$anonfun$predicates$4$adapted(OrderedFilters.scala:61)

at scala.collection.TraversableLike.$anonfun$map$1(TraversableLike.scala:286)

at scala.collection.IndexedSeqOptimized.foreach(IndexedSeqOptimized.scala:36)

at scala.collection.IndexedSeqOptimized.foreach$(IndexedSeqOptimized.scala:33)

at scala.collection.mutable.ArrayOps$ofRef.foreach(ArrayOps.scala:198)

at scala.collection.TraversableLike.map(TraversableLike.scala:286)

at scala.collection.TraversableLike.map$(TraversableLike.scala:279)

at scala.collection.mutable.ArrayOps$ofRef.map(ArrayOps.scala:198)

at org.apache.spark.sql.catalyst.OrderedFilters.$anonfun$predicates$3(OrderedFilters.scala:61)

at org.apache.spark.sql.catalyst.OrderedFilters.$anonfun$predicates$3$adapted(OrderedFilters.scala:50)

at scala.collection.immutable.List.foreach(List.scala:431)

at org.apache.spark.sql.catalyst.OrderedFilters.<init>(OrderedFilters.scala:50)

at org.apache.spark.sql.catalyst.csv.UnivocityParser.<init>(UnivocityParser.scala:103)

at org.apache.spark.sql.execution.datasources.csv.CSVFileFormat.$anonfun$buildReader$1(CSVFileFormat.scala:130)

at org.apache.spark.sql.execution.datasources.FileFormat$$anon$1.apply(FileFormat.scala:148)

at org.apache.spark.sql.execution.datasources.FileFormat$$anon$1.apply(FileFormat.scala:133)

at org.apache.spark.sql.execution.datasources.FileScanRDD$$anon$1.org$apache$spark$sql$execution$datasources$FileScanRDD$$anon$$readCurrentFile(FileScanRDD.scala:127)

at org.apache.spark.sql.execution.datasources.FileScanRDD$$anon$1.nextIterator(FileScanRDD.scala:187)

at org.apache.spark.sql.execution.datasources.FileScanRDD$$anon$1.hasNext(FileScanRDD.scala:104)

at scala.collection.Iterator$$anon$10.hasNext(Iterator.scala:460)

at org.apache.spark.sql.catalyst.expressions.GeneratedClass$GeneratedIteratorForCodegenStage1.processNext(Unknown Source)

at org.apache.spark.sql.execution.BufferedRowIterator.hasNext(BufferedRowIterator.java:43)

at org.apache.spark.sql.execution.WholeStageCodegenExec$$anon$1.hasNext(WholeStageCodegenExec.scala:759)

at org.apache.spark.sql.execution.SparkPlan.$anonfun$getByteArrayRdd$1(SparkPlan.scala:349)

at org.apache.spark.rdd.RDD.$anonfun$mapPartitionsInternal$2(RDD.scala:898)

at org.apache.spark.rdd.RDD.$anonfun$mapPartitionsInternal$2$adapted(RDD.scala:898)

at org.apache.spark.rdd.MapPartitionsRDD.compute(MapPartitionsRDD.scala:52)

at org.apache.spark.rdd.RDD.computeOrReadCheckpoint(RDD.scala:373)

at org.apache.spark.rdd.RDD.iterator(RDD.scala:337)

at org.apache.spark.scheduler.ResultTask.runTask(ResultTask.scala:90)

at org.apache.spark.scheduler.Task.run(Task.scala:131)

at org.apache.spark.executor.Executor$TaskRunner.$anonfun$run$3(Executor.scala:506)

at org.apache.spark.util.Utils$.tryWithSafeFinally(Utils.scala:1462)

at org.apache.spark.executor.Executor$TaskRunner.run(Executor.scala:509)

at java.util.concurrent.ThreadPoolExecutor.runWorker(ThreadPoolExecutor.java:1149)

at java.util.concurrent.ThreadPoolExecutor$Worker.run(ThreadPoolExecutor.java:624)

at java.lang.Thread.run(Thread.java:750)

24/07/18 20:35:37 ERROR scheduler.TaskSetManager: Task 0 in stage 14.0 failed 1 times; aborting job

Traceback (most recent call last):

File "<stdin>", line 1, in <module>

File "/usr/local/spark/python/pyspark/sql/dataframe.py", line 494, in show

print(self.\_jdf.showString(n, 20, vertical))

File "/usr/local/spark/python/lib/py4j-0.10.9.3-src.zip/py4j/java\_gateway.py", line 1322, in \_\_call\_\_

File "/usr/local/spark/python/pyspark/sql/utils.py", line 111, in deco

return f(\*a, \*\*kw)

File "/usr/local/spark/python/lib/py4j-0.10.9.3-src.zip/py4j/protocol.py", line 328, in get\_return\_value

py4j.protocol.Py4JJavaError: An error occurred while calling o221.showString.

: org.apache.spark.SparkException: Job aborted due to stage failure: Task 0 in stage 14.0 failed 1 times, most recent failure: Lost task 0.0 in stage 14.0 (TID 14) (192.168.112.129 executor driver): java.lang.IllegalArgumentException: RejectRows does not exist. Available: IssuerId, IssuerId2, BusinessDate, StateCode, SourceName, NetworkName, NetworkURL, custnum, MarketCoverage, DentalOnlyPlan

at org.apache.spark.sql.types.StructType.$anonfun$fieldIndex$1(StructType.scala:306)

at scala.collection.immutable.HashMap$HashTrieMap.getOrElse0(HashMap.scala:596)

at scala.collection.immutable.HashMap.getOrElse(HashMap.scala:73)

at org.apache.spark.sql.types.StructType.fieldIndex(StructType.scala:305)

at org.apache.spark.sql.catalyst.OrderedFilters.$anonfun$predicates$4(OrderedFilters.scala:61)

at org.apache.spark.sql.catalyst.OrderedFilters.$anonfun$predicates$4$adapted(OrderedFilters.scala:61)

at scala.collection.TraversableLike.$anonfun$map$1(TraversableLike.scala:286)

at scala.collection.IndexedSeqOptimized.foreach(IndexedSeqOptimized.scala:36)

at scala.collection.IndexedSeqOptimized.foreach$(IndexedSeqOptimized.scala:33)

at scala.collection.mutable.ArrayOps$ofRef.foreach(ArrayOps.scala:198)

at scala.collection.TraversableLike.map(TraversableLike.scala:286)

at scala.collection.TraversableLike.map$(TraversableLike.scala:279)

at scala.collection.mutable.ArrayOps$ofRef.map(ArrayOps.scala:198)

at org.apache.spark.sql.catalyst.OrderedFilters.$anonfun$predicates$3(OrderedFilters.scala:61)

at org.apache.spark.sql.catalyst.OrderedFilters.$anonfun$predicates$3$adapted(OrderedFilters.scala:50)

at scala.collection.immutable.List.foreach(List.scala:431)

at org.apache.spark.sql.catalyst.OrderedFilters.<init>(OrderedFilters.scala:50)

at org.apache.spark.sql.catalyst.csv.UnivocityParser.<init>(UnivocityParser.scala:103)

at org.apache.spark.sql.execution.datasources.csv.CSVFileFormat.$anonfun$buildReader$1(CSVFileFormat.scala:130)

at org.apache.spark.sql.execution.datasources.FileFormat$$anon$1.apply(FileFormat.scala:148)

at org.apache.spark.sql.execution.datasources.FileFormat$$anon$1.apply(FileFormat.scala:133)

at org.apache.spark.sql.execution.datasources.FileScanRDD$$anon$1.org$apache$spark$sql$execution$datasources$FileScanRDD$$anon$$readCurrentFile(FileScanRDD.scala:127)

at org.apache.spark.sql.execution.datasources.FileScanRDD$$anon$1.nextIterator(FileScanRDD.scala:187)

at org.apache.spark.sql.execution.datasources.FileScanRDD$$anon$1.hasNext(FileScanRDD.scala:104)

at scala.collection.Iterator$$anon$10.hasNext(Iterator.scala:460)

at org.apache.spark.sql.catalyst.expressions.GeneratedClass$GeneratedIteratorForCodegenStage1.processNext(Unknown Source)

at org.apache.spark.sql.execution.BufferedRowIterator.hasNext(BufferedRowIterator.java:43)

at org.apache.spark.sql.execution.WholeStageCodegenExec$$anon$1.hasNext(WholeStageCodegenExec.scala:759)

at org.apache.spark.sql.execution.SparkPlan.$anonfun$getByteArrayRdd$1(SparkPlan.scala:349)

at org.apache.spark.rdd.RDD.$anonfun$mapPartitionsInternal$2(RDD.scala:898)

at org.apache.spark.rdd.RDD.$anonfun$mapPartitionsInternal$2$adapted(RDD.scala:898)

at org.apache.spark.rdd.MapPartitionsRDD.compute(MapPartitionsRDD.scala:52)

at org.apache.spark.rdd.RDD.computeOrReadCheckpoint(RDD.scala:373)

at org.apache.spark.rdd.RDD.iterator(RDD.scala:337)

at org.apache.spark.scheduler.ResultTask.runTask(ResultTask.scala:90)

at org.apache.spark.scheduler.Task.run(Task.scala:131)

at org.apache.spark.executor.Executor$TaskRunner.$anonfun$run$3(Executor.scala:506)

at org.apache.spark.util.Utils$.tryWithSafeFinally(Utils.scala:1462)

at org.apache.spark.executor.Executor$TaskRunner.run(Executor.scala:509)

at java.util.concurrent.ThreadPoolExecutor.runWorker(ThreadPoolExecutor.java:1149)

at java.util.concurrent.ThreadPoolExecutor$Worker.run(ThreadPoolExecutor.java:624)

at java.lang.Thread.run(Thread.java:750)

Driver stacktrace:

at org.apache.spark.scheduler.DAGScheduler.failJobAndIndependentStages(DAGScheduler.scala:2454)

at org.apache.spark.scheduler.DAGScheduler.$anonfun$abortStage$2(DAGScheduler.scala:2403)

at org.apache.spark.scheduler.DAGScheduler.$anonfun$abortStage$2$adapted(DAGScheduler.scala:2402)

at scala.collection.mutable.ResizableArray.foreach(ResizableArray.scala:62)

at scala.collection.mutable.ResizableArray.foreach$(ResizableArray.scala:55)

at scala.collection.mutable.ArrayBuffer.foreach(ArrayBuffer.scala:49)

at org.apache.spark.scheduler.DAGScheduler.abortStage(DAGScheduler.scala:2402)

at org.apache.spark.scheduler.DAGScheduler.$anonfun$handleTaskSetFailed$1(DAGScheduler.scala:1160)

at org.apache.spark.scheduler.DAGScheduler.$anonfun$handleTaskSetFailed$1$adapted(DAGScheduler.scala:1160)

at scala.Option.foreach(Option.scala:407)

at org.apache.spark.scheduler.DAGScheduler.handleTaskSetFailed(DAGScheduler.scala:1160)

at org.apache.spark.scheduler.DAGSchedulerEventProcessLoop.doOnReceive(DAGScheduler.scala:2642)

at org.apache.spark.scheduler.DAGSchedulerEventProcessLoop.onReceive(DAGScheduler.scala:2584)

at org.apache.spark.scheduler.DAGSchedulerEventProcessLoop.onReceive(DAGScheduler.scala:2573)

at org.apache.spark.util.EventLoop$$anon$1.run(EventLoop.scala:49)

at org.apache.spark.scheduler.DAGScheduler.runJob(DAGScheduler.scala:938)

at org.apache.spark.SparkContext.runJob(SparkContext.scala:2214)

at org.apache.spark.SparkContext.runJob(SparkContext.scala:2235)

at org.apache.spark.SparkContext.runJob(SparkContext.scala:2254)

at org.apache.spark.sql.execution.SparkPlan.executeTake(SparkPlan.scala:476)

at org.apache.spark.sql.execution.SparkPlan.executeTake(SparkPlan.scala:429)

at org.apache.spark.sql.execution.CollectLimitExec.executeCollect(limit.scala:48)

at org.apache.spark.sql.Dataset.collectFromPlan(Dataset.scala:3715)

at org.apache.spark.sql.Dataset.$anonfun$head$1(Dataset.scala:2728)

at org.apache.spark.sql.Dataset.$anonfun$withAction$1(Dataset.scala:3706)

at org.apache.spark.sql.execution.SQLExecution$.$anonfun$withNewExecutionId$5(SQLExecution.scala:103)

at org.apache.spark.sql.execution.SQLExecution$.withSQLConfPropagated(SQLExecution.scala:163)

at org.apache.spark.sql.execution.SQLExecution$.$anonfun$withNewExecutionId$1(SQLExecution.scala:90)

at org.apache.spark.sql.SparkSession.withActive(SparkSession.scala:775)

at org.apache.spark.sql.execution.SQLExecution$.withNewExecutionId(SQLExecution.scala:64)

at org.apache.spark.sql.Dataset.withAction(Dataset.scala:3704)

at org.apache.spark.sql.Dataset.head(Dataset.scala:2728)

at org.apache.spark.sql.Dataset.take(Dataset.scala:2935)

at org.apache.spark.sql.Dataset.getRows(Dataset.scala:287)

at org.apache.spark.sql.Dataset.showString(Dataset.scala:326)

at sun.reflect.GeneratedMethodAccessor41.invoke(Unknown Source)

at sun.reflect.DelegatingMethodAccessorImpl.invoke(DelegatingMethodAccessorImpl.java:43)

at java.lang.reflect.Method.invoke(Method.java:498)

at py4j.reflection.MethodInvoker.invoke(MethodInvoker.java:244)

at py4j.reflection.ReflectionEngine.invoke(ReflectionEngine.java:357)

at py4j.Gateway.invoke(Gateway.java:282)

at py4j.commands.AbstractCommand.invokeMethod(AbstractCommand.java:132)

at py4j.commands.CallCommand.execute(CallCommand.java:79)

at py4j.ClientServerConnection.waitForCommands(ClientServerConnection.java:182)

at py4j.ClientServerConnection.run(ClientServerConnection.java:106)

at java.lang.Thread.run(Thread.java:750)

Caused by: java.lang.IllegalArgumentException: RejectRows does not exist. Available: IssuerId, IssuerId2, BusinessDate, StateCode, SourceName, NetworkName, NetworkURL, custnum, MarketCoverage, DentalOnlyPlan

at org.apache.spark.sql.types.StructType.$anonfun$fieldIndex$1(StructType.scala:306)

at scala.collection.immutable.HashMap$HashTrieMap.getOrElse0(HashMap.scala:596)

at scala.collection.immutable.HashMap.getOrElse(HashMap.scala:73)

at org.apache.spark.sql.types.StructType.fieldIndex(StructType.scala:305)

at org.apache.spark.sql.catalyst.OrderedFilters.$anonfun$predicates$4(OrderedFilters.scala:61)

at org.apache.spark.sql.catalyst.OrderedFilters.$anonfun$predicates$4$adapted(OrderedFilters.scala:61)

at scala.collection.TraversableLike.$anonfun$map$1(TraversableLike.scala:286)

at scala.collection.IndexedSeqOptimized.foreach(IndexedSeqOptimized.scala:36)

at scala.collection.IndexedSeqOptimized.foreach$(IndexedSeqOptimized.scala:33)

at scala.collection.mutable.ArrayOps$ofRef.foreach(ArrayOps.scala:198)

at scala.collection.TraversableLike.map(TraversableLike.scala:286)

at scala.collection.TraversableLike.map$(TraversableLike.scala:279)

at scala.collection.mutable.ArrayOps$ofRef.map(ArrayOps.scala:198)

at org.apache.spark.sql.catalyst.OrderedFilters.$anonfun$predicates$3(OrderedFilters.scala:61)

at org.apache.spark.sql.catalyst.OrderedFilters.$anonfun$predicates$3$adapted(OrderedFilters.scala:50)

at scala.collection.immutable.List.foreach(List.scala:431)

at org.apache.spark.sql.catalyst.OrderedFilters.<init>(OrderedFilters.scala:50)

at org.apache.spark.sql.catalyst.csv.UnivocityParser.<init>(UnivocityParser.scala:103)

at org.apache.spark.sql.execution.datasources.csv.CSVFileFormat.$anonfun$buildReader$1(CSVFileFormat.scala:130)

at org.apache.spark.sql.execution.datasources.FileFormat$$anon$1.apply(FileFormat.scala:148)

at org.apache.spark.sql.execution.datasources.FileFormat$$anon$1.apply(FileFormat.scala:133)

at org.apache.spark.sql.execution.datasources.FileScanRDD$$anon$1.org$apache$spark$sql$execution$datasources$FileScanRDD$$anon$$readCurrentFile(FileScanRDD.scala:127)

at org.apache.spark.sql.execution.datasources.FileScanRDD$$anon$1.nextIterator(FileScanRDD.scala:187)

at org.apache.spark.sql.execution.datasources.FileScanRDD$$anon$1.hasNext(FileScanRDD.scala:104)

at scala.collection.Iterator$$anon$10.hasNext(Iterator.scala:460)

at org.apache.spark.sql.catalyst.expressions.GeneratedClass$GeneratedIteratorForCodegenStage1.processNext(Unknown Source)

at org.apache.spark.sql.execution.BufferedRowIterator.hasNext(BufferedRowIterator.java:43)

at org.apache.spark.sql.execution.WholeStageCodegenExec$$anon$1.hasNext(WholeStageCodegenExec.scala:759)

at org.apache.spark.sql.execution.SparkPlan.$anonfun$getByteArrayRdd$1(SparkPlan.scala:349)

at org.apache.spark.rdd.RDD.$anonfun$mapPartitionsInternal$2(RDD.scala:898)

at org.apache.spark.rdd.RDD.$anonfun$mapPartitionsInternal$2$adapted(RDD.scala:898)

at org.apache.spark.rdd.MapPartitionsRDD.compute(MapPartitionsRDD.scala:52)

at org.apache.spark.rdd.RDD.computeOrReadCheckpoint(RDD.scala:373)

at org.apache.spark.rdd.RDD.iterator(RDD.scala:337)

at org.apache.spark.scheduler.ResultTask.runTask(ResultTask.scala:90)

at org.apache.spark.scheduler.Task.run(Task.scala:131)

at org.apache.spark.executor.Executor$TaskRunner.$anonfun$run$3(Executor.scala:506)

at org.apache.spark.util.Utils$.tryWithSafeFinally(Utils.scala:1462)

at org.apache.spark.executor.Executor$TaskRunner.run(Executor.scala:509)

at java.util.concurrent.ThreadPoolExecutor.runWorker(ThreadPoolExecutor.java:1149)

at java.util.concurrent.ThreadPoolExecutor$Worker.run(ThreadPoolExecutor.java:624)

... 1 more

>>> data\_frame\_1.desc

Traceback (most recent call last):

File "<stdin>", line 1, in <module>

File "/usr/local/spark/python/pyspark/sql/dataframe.py", line 1660, in \_\_getattr\_\_

"'%s' object has no attribute '%s'" % (self.\_\_class\_\_.\_\_name\_\_, name))

AttributeError: 'DataFrame' object has no attribute 'desc'

>>> data\_frame\_1.describe

<bound method DataFrame.describe of DataFrame[IssuerId: int, IssuerId2: int, BusinessDate: date, StateCode: string, SourceName: string, NetworkName: string, NetworkURL: string, custnum: string, MarketCoverage: string, DentalOnlyPlan: string, RejectRows: string]>

>>> data\_frame\_1.where("RejectRows!='null'").show(3)

24/07/18 20:37:47 ERROR executor.Executor: Exception in task 0.0 in stage 15.0 (TID 15)

java.lang.IllegalArgumentException: RejectRows does not exist. Available: IssuerId, IssuerId2, BusinessDate, StateCode, SourceName, NetworkName, NetworkURL, custnum, MarketCoverage, DentalOnlyPlan

at org.apache.spark.sql.types.StructType.$anonfun$fieldIndex$1(StructType.scala:306)

at scala.collection.immutable.HashMap$HashTrieMap.getOrElse0(HashMap.scala:596)

at scala.collection.immutable.HashMap.getOrElse(HashMap.scala:73)

at org.apache.spark.sql.types.StructType.fieldIndex(StructType.scala:305)

at org.apache.spark.sql.catalyst.OrderedFilters.$anonfun$predicates$4(OrderedFilters.scala:61)

at org.apache.spark.sql.catalyst.OrderedFilters.$anonfun$predicates$4$adapted(OrderedFilters.scala:61)

at scala.collection.TraversableLike.$anonfun$map$1(TraversableLike.scala:286)

at scala.collection.IndexedSeqOptimized.foreach(IndexedSeqOptimized.scala:36)

at scala.collection.IndexedSeqOptimized.foreach$(IndexedSeqOptimized.scala:33)

at scala.collection.mutable.ArrayOps$ofRef.foreach(ArrayOps.scala:198)

at scala.collection.TraversableLike.map(TraversableLike.scala:286)

at scala.collection.TraversableLike.map$(TraversableLike.scala:279)

at scala.collection.mutable.ArrayOps$ofRef.map(ArrayOps.scala:198)

at org.apache.spark.sql.catalyst.OrderedFilters.$anonfun$predicates$3(OrderedFilters.scala:61)

at org.apache.spark.sql.catalyst.OrderedFilters.$anonfun$predicates$3$adapted(OrderedFilters.scala:50)

at scala.collection.immutable.List.foreach(List.scala:431)

at org.apache.spark.sql.catalyst.OrderedFilters.<init>(OrderedFilters.scala:50)

at org.apache.spark.sql.catalyst.csv.UnivocityParser.<init>(UnivocityParser.scala:103)

at org.apache.spark.sql.execution.datasources.csv.CSVFileFormat.$anonfun$buildReader$1(CSVFileFormat.scala:130)

at org.apache.spark.sql.execution.datasources.FileFormat$$anon$1.apply(FileFormat.scala:148)

at org.apache.spark.sql.execution.datasources.FileFormat$$anon$1.apply(FileFormat.scala:133)

at org.apache.spark.sql.execution.datasources.FileScanRDD$$anon$1.org$apache$spark$sql$execution$datasources$FileScanRDD$$anon$$readCurrentFile(FileScanRDD.scala:127)

at org.apache.spark.sql.execution.datasources.FileScanRDD$$anon$1.nextIterator(FileScanRDD.scala:187)

at org.apache.spark.sql.execution.datasources.FileScanRDD$$anon$1.hasNext(FileScanRDD.scala:104)

at scala.collection.Iterator$$anon$10.hasNext(Iterator.scala:460)

at org.apache.spark.sql.catalyst.expressions.GeneratedClass$GeneratedIteratorForCodegenStage1.processNext(Unknown Source)

at org.apache.spark.sql.execution.BufferedRowIterator.hasNext(BufferedRowIterator.java:43)

at org.apache.spark.sql.execution.WholeStageCodegenExec$$anon$1.hasNext(WholeStageCodegenExec.scala:759)

at org.apache.spark.sql.execution.SparkPlan.$anonfun$getByteArrayRdd$1(SparkPlan.scala:349)

at org.apache.spark.rdd.RDD.$anonfun$mapPartitionsInternal$2(RDD.scala:898)

at org.apache.spark.rdd.RDD.$anonfun$mapPartitionsInternal$2$adapted(RDD.scala:898)

at org.apache.spark.rdd.MapPartitionsRDD.compute(MapPartitionsRDD.scala:52)

at org.apache.spark.rdd.RDD.computeOrReadCheckpoint(RDD.scala:373)

at org.apache.spark.rdd.RDD.iterator(RDD.scala:337)

at org.apache.spark.scheduler.ResultTask.runTask(ResultTask.scala:90)

at org.apache.spark.scheduler.Task.run(Task.scala:131)

at org.apache.spark.executor.Executor$TaskRunner.$anonfun$run$3(Executor.scala:506)

at org.apache.spark.util.Utils$.tryWithSafeFinally(Utils.scala:1462)

at org.apache.spark.executor.Executor$TaskRunner.run(Executor.scala:509)

at java.util.concurrent.ThreadPoolExecutor.runWorker(ThreadPoolExecutor.java:1149)

at java.util.concurrent.ThreadPoolExecutor$Worker.run(ThreadPoolExecutor.java:624)

at java.lang.Thread.run(Thread.java:750)

24/07/18 20:37:47 WARN scheduler.TaskSetManager: Lost task 0.0 in stage 15.0 (TID 15) (192.168.112.129 executor driver): java.lang.IllegalArgumentException: RejectRows does not exist. Available: IssuerId, IssuerId2, BusinessDate, StateCode, SourceName, NetworkName, NetworkURL, custnum, MarketCoverage, DentalOnlyPlan

at org.apache.spark.sql.types.StructType.$anonfun$fieldIndex$1(StructType.scala:306)

at scala.collection.immutable.HashMap$HashTrieMap.getOrElse0(HashMap.scala:596)

at scala.collection.immutable.HashMap.getOrElse(HashMap.scala:73)

at org.apache.spark.sql.types.StructType.fieldIndex(StructType.scala:305)

at org.apache.spark.sql.catalyst.OrderedFilters.$anonfun$predicates$4(OrderedFilters.scala:61)

at org.apache.spark.sql.catalyst.OrderedFilters.$anonfun$predicates$4$adapted(OrderedFilters.scala:61)

at scala.collection.TraversableLike.$anonfun$map$1(TraversableLike.scala:286)

at scala.collection.IndexedSeqOptimized.foreach(IndexedSeqOptimized.scala:36)

at scala.collection.IndexedSeqOptimized.foreach$(IndexedSeqOptimized.scala:33)

at scala.collection.mutable.ArrayOps$ofRef.foreach(ArrayOps.scala:198)

at scala.collection.TraversableLike.map(TraversableLike.scala:286)

at scala.collection.TraversableLike.map$(TraversableLike.scala:279)

at scala.collection.mutable.ArrayOps$ofRef.map(ArrayOps.scala:198)

at org.apache.spark.sql.catalyst.OrderedFilters.$anonfun$predicates$3(OrderedFilters.scala:61)

at org.apache.spark.sql.catalyst.OrderedFilters.$anonfun$predicates$3$adapted(OrderedFilters.scala:50)

at scala.collection.immutable.List.foreach(List.scala:431)

at org.apache.spark.sql.catalyst.OrderedFilters.<init>(OrderedFilters.scala:50)

at org.apache.spark.sql.catalyst.csv.UnivocityParser.<init>(UnivocityParser.scala:103)

at org.apache.spark.sql.execution.datasources.csv.CSVFileFormat.$anonfun$buildReader$1(CSVFileFormat.scala:130)

at org.apache.spark.sql.execution.datasources.FileFormat$$anon$1.apply(FileFormat.scala:148)

at org.apache.spark.sql.execution.datasources.FileFormat$$anon$1.apply(FileFormat.scala:133)

at org.apache.spark.sql.execution.datasources.FileScanRDD$$anon$1.org$apache$spark$sql$execution$datasources$FileScanRDD$$anon$$readCurrentFile(FileScanRDD.scala:127)

at org.apache.spark.sql.execution.datasources.FileScanRDD$$anon$1.nextIterator(FileScanRDD.scala:187)

at org.apache.spark.sql.execution.datasources.FileScanRDD$$anon$1.hasNext(FileScanRDD.scala:104)

at scala.collection.Iterator$$anon$10.hasNext(Iterator.scala:460)

at org.apache.spark.sql.catalyst.expressions.GeneratedClass$GeneratedIteratorForCodegenStage1.processNext(Unknown Source)

at org.apache.spark.sql.execution.BufferedRowIterator.hasNext(BufferedRowIterator.java:43)

at org.apache.spark.sql.execution.WholeStageCodegenExec$$anon$1.hasNext(WholeStageCodegenExec.scala:759)

at org.apache.spark.sql.execution.SparkPlan.$anonfun$getByteArrayRdd$1(SparkPlan.scala:349)

at org.apache.spark.rdd.RDD.$anonfun$mapPartitionsInternal$2(RDD.scala:898)

at org.apache.spark.rdd.RDD.$anonfun$mapPartitionsInternal$2$adapted(RDD.scala:898)

at org.apache.spark.rdd.MapPartitionsRDD.compute(MapPartitionsRDD.scala:52)

at org.apache.spark.rdd.RDD.computeOrReadCheckpoint(RDD.scala:373)

at org.apache.spark.rdd.RDD.iterator(RDD.scala:337)

at org.apache.spark.scheduler.ResultTask.runTask(ResultTask.scala:90)

at org.apache.spark.scheduler.Task.run(Task.scala:131)

at org.apache.spark.executor.Executor$TaskRunner.$anonfun$run$3(Executor.scala:506)

at org.apache.spark.util.Utils$.tryWithSafeFinally(Utils.scala:1462)

at org.apache.spark.executor.Executor$TaskRunner.run(Executor.scala:509)

at java.util.concurrent.ThreadPoolExecutor.runWorker(ThreadPoolExecutor.java:1149)

at java.util.concurrent.ThreadPoolExecutor$Worker.run(ThreadPoolExecutor.java:624)

at java.lang.Thread.run(Thread.java:750)

24/07/18 20:37:47 ERROR scheduler.TaskSetManager: Task 0 in stage 15.0 failed 1 times; aborting job

Traceback (most recent call last):

File "<stdin>", line 1, in <module>

File "/usr/local/spark/python/pyspark/sql/dataframe.py", line 494, in show

print(self.\_jdf.showString(n, 20, vertical))

File "/usr/local/spark/python/lib/py4j-0.10.9.3-src.zip/py4j/java\_gateway.py", line 1322, in \_\_call\_\_

File "/usr/local/spark/python/pyspark/sql/utils.py", line 111, in deco

return f(\*a, \*\*kw)

File "/usr/local/spark/python/lib/py4j-0.10.9.3-src.zip/py4j/protocol.py", line 328, in get\_return\_value

py4j.protocol.Py4JJavaError: An error occurred while calling o227.showString.

: org.apache.spark.SparkException: Job aborted due to stage failure: Task 0 in stage 15.0 failed 1 times, most recent failure: Lost task 0.0 in stage 15.0 (TID 15) (192.168.112.129 executor driver): java.lang.IllegalArgumentException: RejectRows does not exist. Available: IssuerId, IssuerId2, BusinessDate, StateCode, SourceName, NetworkName, NetworkURL, custnum, MarketCoverage, DentalOnlyPlan

at org.apache.spark.sql.types.StructType.$anonfun$fieldIndex$1(StructType.scala:306)

at scala.collection.immutable.HashMap$HashTrieMap.getOrElse0(HashMap.scala:596)

at scala.collection.immutable.HashMap.getOrElse(HashMap.scala:73)

at org.apache.spark.sql.types.StructType.fieldIndex(StructType.scala:305)

at org.apache.spark.sql.catalyst.OrderedFilters.$anonfun$predicates$4(OrderedFilters.scala:61)

at org.apache.spark.sql.catalyst.OrderedFilters.$anonfun$predicates$4$adapted(OrderedFilters.scala:61)

at scala.collection.TraversableLike.$anonfun$map$1(TraversableLike.scala:286)

at scala.collection.IndexedSeqOptimized.foreach(IndexedSeqOptimized.scala:36)

at scala.collection.IndexedSeqOptimized.foreach$(IndexedSeqOptimized.scala:33)

at scala.collection.mutable.ArrayOps$ofRef.foreach(ArrayOps.scala:198)

at scala.collection.TraversableLike.map(TraversableLike.scala:286)

at scala.collection.TraversableLike.map$(TraversableLike.scala:279)

at scala.collection.mutable.ArrayOps$ofRef.map(ArrayOps.scala:198)

at org.apache.spark.sql.catalyst.OrderedFilters.$anonfun$predicates$3(OrderedFilters.scala:61)

at org.apache.spark.sql.catalyst.OrderedFilters.$anonfun$predicates$3$adapted(OrderedFilters.scala:50)

at scala.collection.immutable.List.foreach(List.scala:431)

at org.apache.spark.sql.catalyst.OrderedFilters.<init>(OrderedFilters.scala:50)

at org.apache.spark.sql.catalyst.csv.UnivocityParser.<init>(UnivocityParser.scala:103)

at org.apache.spark.sql.execution.datasources.csv.CSVFileFormat.$anonfun$buildReader$1(CSVFileFormat.scala:130)

at org.apache.spark.sql.execution.datasources.FileFormat$$anon$1.apply(FileFormat.scala:148)

at org.apache.spark.sql.execution.datasources.FileFormat$$anon$1.apply(FileFormat.scala:133)

at org.apache.spark.sql.execution.datasources.FileScanRDD$$anon$1.org$apache$spark$sql$execution$datasources$FileScanRDD$$anon$$readCurrentFile(FileScanRDD.scala:127)

at org.apache.spark.sql.execution.datasources.FileScanRDD$$anon$1.nextIterator(FileScanRDD.scala:187)

at org.apache.spark.sql.execution.datasources.FileScanRDD$$anon$1.hasNext(FileScanRDD.scala:104)

at scala.collection.Iterator$$anon$10.hasNext(Iterator.scala:460)

at org.apache.spark.sql.catalyst.expressions.GeneratedClass$GeneratedIteratorForCodegenStage1.processNext(Unknown Source)

at org.apache.spark.sql.execution.BufferedRowIterator.hasNext(BufferedRowIterator.java:43)

at org.apache.spark.sql.execution.WholeStageCodegenExec$$anon$1.hasNext(WholeStageCodegenExec.scala:759)

at org.apache.spark.sql.execution.SparkPlan.$anonfun$getByteArrayRdd$1(SparkPlan.scala:349)

at org.apache.spark.rdd.RDD.$anonfun$mapPartitionsInternal$2(RDD.scala:898)

at org.apache.spark.rdd.RDD.$anonfun$mapPartitionsInternal$2$adapted(RDD.scala:898)

at org.apache.spark.rdd.MapPartitionsRDD.compute(MapPartitionsRDD.scala:52)

at org.apache.spark.rdd.RDD.computeOrReadCheckpoint(RDD.scala:373)

at org.apache.spark.rdd.RDD.iterator(RDD.scala:337)

at org.apache.spark.scheduler.ResultTask.runTask(ResultTask.scala:90)

at org.apache.spark.scheduler.Task.run(Task.scala:131)

at org.apache.spark.executor.Executor$TaskRunner.$anonfun$run$3(Executor.scala:506)

at org.apache.spark.util.Utils$.tryWithSafeFinally(Utils.scala:1462)

at org.apache.spark.executor.Executor$TaskRunner.run(Executor.scala:509)

at java.util.concurrent.ThreadPoolExecutor.runWorker(ThreadPoolExecutor.java:1149)

at java.util.concurrent.ThreadPoolExecutor$Worker.run(ThreadPoolExecutor.java:624)

at java.lang.Thread.run(Thread.java:750)

Driver stacktrace:

at org.apache.spark.scheduler.DAGScheduler.failJobAndIndependentStages(DAGScheduler.scala:2454)

at org.apache.spark.scheduler.DAGScheduler.$anonfun$abortStage$2(DAGScheduler.scala:2403)

at org.apache.spark.scheduler.DAGScheduler.$anonfun$abortStage$2$adapted(DAGScheduler.scala:2402)

at scala.collection.mutable.ResizableArray.foreach(ResizableArray.scala:62)

at scala.collection.mutable.ResizableArray.foreach$(ResizableArray.scala:55)

at scala.collection.mutable.ArrayBuffer.foreach(ArrayBuffer.scala:49)

at org.apache.spark.scheduler.DAGScheduler.abortStage(DAGScheduler.scala:2402)

at org.apache.spark.scheduler.DAGScheduler.$anonfun$handleTaskSetFailed$1(DAGScheduler.scala:1160)

at org.apache.spark.scheduler.DAGScheduler.$anonfun$handleTaskSetFailed$1$adapted(DAGScheduler.scala:1160)

at scala.Option.foreach(Option.scala:407)

at org.apache.spark.scheduler.DAGScheduler.handleTaskSetFailed(DAGScheduler.scala:1160)

at org.apache.spark.scheduler.DAGSchedulerEventProcessLoop.doOnReceive(DAGScheduler.scala:2642)

at org.apache.spark.scheduler.DAGSchedulerEventProcessLoop.onReceive(DAGScheduler.scala:2584)

at org.apache.spark.scheduler.DAGSchedulerEventProcessLoop.onReceive(DAGScheduler.scala:2573)

at org.apache.spark.util.EventLoop$$anon$1.run(EventLoop.scala:49)

at org.apache.spark.scheduler.DAGScheduler.runJob(DAGScheduler.scala:938)

at org.apache.spark.SparkContext.runJob(SparkContext.scala:2214)

at org.apache.spark.SparkContext.runJob(SparkContext.scala:2235)

at org.apache.spark.SparkContext.runJob(SparkContext.scala:2254)

at org.apache.spark.sql.execution.SparkPlan.executeTake(SparkPlan.scala:476)

at org.apache.spark.sql.execution.SparkPlan.executeTake(SparkPlan.scala:429)

at org.apache.spark.sql.execution.CollectLimitExec.executeCollect(limit.scala:48)

at org.apache.spark.sql.Dataset.collectFromPlan(Dataset.scala:3715)

at org.apache.spark.sql.Dataset.$anonfun$head$1(Dataset.scala:2728)

at org.apache.spark.sql.Dataset.$anonfun$withAction$1(Dataset.scala:3706)

at org.apache.spark.sql.execution.SQLExecution$.$anonfun$withNewExecutionId$5(SQLExecution.scala:103)

at org.apache.spark.sql.execution.SQLExecution$.withSQLConfPropagated(SQLExecution.scala:163)

at org.apache.spark.sql.execution.SQLExecution$.$anonfun$withNewExecutionId$1(SQLExecution.scala:90)

at org.apache.spark.sql.SparkSession.withActive(SparkSession.scala:775)

at org.apache.spark.sql.execution.SQLExecution$.withNewExecutionId(SQLExecution.scala:64)

at org.apache.spark.sql.Dataset.withAction(Dataset.scala:3704)

at org.apache.spark.sql.Dataset.head(Dataset.scala:2728)

at org.apache.spark.sql.Dataset.take(Dataset.scala:2935)

at org.apache.spark.sql.Dataset.getRows(Dataset.scala:287)

at org.apache.spark.sql.Dataset.showString(Dataset.scala:326)

at sun.reflect.GeneratedMethodAccessor41.invoke(Unknown Source)

at sun.reflect.DelegatingMethodAccessorImpl.invoke(DelegatingMethodAccessorImpl.java:43)

at java.lang.reflect.Method.invoke(Method.java:498)

at py4j.reflection.MethodInvoker.invoke(MethodInvoker.java:244)

at py4j.reflection.ReflectionEngine.invoke(ReflectionEngine.java:357)

at py4j.Gateway.invoke(Gateway.java:282)

at py4j.commands.AbstractCommand.invokeMethod(AbstractCommand.java:132)

at py4j.commands.CallCommand.execute(CallCommand.java:79)

at py4j.ClientServerConnection.waitForCommands(ClientServerConnection.java:182)

at py4j.ClientServerConnection.run(ClientServerConnection.java:106)

at java.lang.Thread.run(Thread.java:750)

Caused by: java.lang.IllegalArgumentException: RejectRows does not exist. Available: IssuerId, IssuerId2, BusinessDate, StateCode, SourceName, NetworkName, NetworkURL, custnum, MarketCoverage, DentalOnlyPlan

at org.apache.spark.sql.types.StructType.$anonfun$fieldIndex$1(StructType.scala:306)

at scala.collection.immutable.HashMap$HashTrieMap.getOrElse0(HashMap.scala:596)

at scala.collection.immutable.HashMap.getOrElse(HashMap.scala:73)

at org.apache.spark.sql.types.StructType.fieldIndex(StructType.scala:305)

at org.apache.spark.sql.catalyst.OrderedFilters.$anonfun$predicates$4(OrderedFilters.scala:61)

at org.apache.spark.sql.catalyst.OrderedFilters.$anonfun$predicates$4$adapted(OrderedFilters.scala:61)

at scala.collection.TraversableLike.$anonfun$map$1(TraversableLike.scala:286)

at scala.collection.IndexedSeqOptimized.foreach(IndexedSeqOptimized.scala:36)

at scala.collection.IndexedSeqOptimized.foreach$(IndexedSeqOptimized.scala:33)

at scala.collection.mutable.ArrayOps$ofRef.foreach(ArrayOps.scala:198)

at scala.collection.TraversableLike.map(TraversableLike.scala:286)

at scala.collection.TraversableLike.map$(TraversableLike.scala:279)

at scala.collection.mutable.ArrayOps$ofRef.map(ArrayOps.scala:198)

at org.apache.spark.sql.catalyst.OrderedFilters.$anonfun$predicates$3(OrderedFilters.scala:61)

at org.apache.spark.sql.catalyst.OrderedFilters.$anonfun$predicates$3$adapted(OrderedFilters.scala:50)

at scala.collection.immutable.List.foreach(List.scala:431)

at org.apache.spark.sql.catalyst.OrderedFilters.<init>(OrderedFilters.scala:50)

at org.apache.spark.sql.catalyst.csv.UnivocityParser.<init>(UnivocityParser.scala:103)

at org.apache.spark.sql.execution.datasources.csv.CSVFileFormat.$anonfun$buildReader$1(CSVFileFormat.scala:130)

at org.apache.spark.sql.execution.datasources.FileFormat$$anon$1.apply(FileFormat.scala:148)

at org.apache.spark.sql.execution.datasources.FileFormat$$anon$1.apply(FileFormat.scala:133)

at org.apache.spark.sql.execution.datasources.FileScanRDD$$anon$1.org$apache$spark$sql$execution$datasources$FileScanRDD$$anon$$readCurrentFile(FileScanRDD.scala:127)

at org.apache.spark.sql.execution.datasources.FileScanRDD$$anon$1.nextIterator(FileScanRDD.scala:187)

at org.apache.spark.sql.execution.datasources.FileScanRDD$$anon$1.hasNext(FileScanRDD.scala:104)

at scala.collection.Iterator$$anon$10.hasNext(Iterator.scala:460)

at org.apache.spark.sql.catalyst.expressions.GeneratedClass$GeneratedIteratorForCodegenStage1.processNext(Unknown Source)

at org.apache.spark.sql.execution.BufferedRowIterator.hasNext(BufferedRowIterator.java:43)

at org.apache.spark.sql.execution.WholeStageCodegenExec$$anon$1.hasNext(WholeStageCodegenExec.scala:759)

at org.apache.spark.sql.execution.SparkPlan.$anonfun$getByteArrayRdd$1(SparkPlan.scala:349)

at org.apache.spark.rdd.RDD.$anonfun$mapPartitionsInternal$2(RDD.scala:898)

at org.apache.spark.rdd.RDD.$anonfun$mapPartitionsInternal$2$adapted(RDD.scala:898)

at org.apache.spark.rdd.MapPartitionsRDD.compute(MapPartitionsRDD.scala:52)

at org.apache.spark.rdd.RDD.computeOrReadCheckpoint(RDD.scala:373)

at org.apache.spark.rdd.RDD.iterator(RDD.scala:337)

at org.apache.spark.scheduler.ResultTask.runTask(ResultTask.scala:90)

at org.apache.spark.scheduler.Task.run(Task.scala:131)

at org.apache.spark.executor.Executor$TaskRunner.$anonfun$run$3(Executor.scala:506)

at org.apache.spark.util.Utils$.tryWithSafeFinally(Utils.scala:1462)

at org.apache.spark.executor.Executor$TaskRunner.run(Executor.scala:509)

at java.util.concurrent.ThreadPoolExecutor.runWorker(ThreadPoolExecutor.java:1149)

at java.util.concurrent.ThreadPoolExecutor$Worker.run(ThreadPoolExecutor.java:624)

... 1 more

>>> data\_frame\_1.show(3)

+--------+---------+------------+---------+----------+------------+--------------------+-------+--------------+--------------+--------------------+

|IssuerId|IssuerId2|BusinessDate|StateCode|SourceName| NetworkName| NetworkURL|custnum|MarketCoverage|DentalOnlyPlan| RejectRows|

+--------+---------+------------+---------+----------+------------+--------------------+-------+--------------+--------------+--------------------+

| 21989| 21989| 2019-10-01| AK| HIOS| ODS Premier|https://www.modah...| 13| null| null| null|

| 219| 219| 2019-10-01| AK| HIOS| ODS Premier| null| null| null| null|219,219,2019-10-0...|

| 38344| 38344| 2019-10-01| AK| HIOS|HeritagePlus|https://www.preme...| 13| null| null| null|

+--------+---------+------------+---------+----------+------------+--------------------+-------+--------------+--------------+--------------------+

only showing top 3 rows

>>> data\_frame\_1.where("RejectRows is not 'null'").show(3)

Traceback (most recent call last):

File "<stdin>", line 1, in <module>

File "/usr/local/spark/python/pyspark/sql/dataframe.py", line 1731, in filter

jdf = self.\_jdf.filter(condition)

File "/usr/local/spark/python/lib/py4j-0.10.9.3-src.zip/py4j/java\_gateway.py", line 1322, in \_\_call\_\_

File "/usr/local/spark/python/pyspark/sql/utils.py", line 117, in deco

raise converted from None

pyspark.sql.utils.ParseException:

no viable alternative at input 'is not 'null''(line 1, pos 18)

== SQL ==

RejectRows is not 'null'

------------------^^^

>>> data\_frame\_1.where("RejectRows is not null").show(3)

24/07/18 20:39:15 ERROR executor.Executor: Exception in task 0.0 in stage 17.0 (TID 17)

java.lang.IllegalArgumentException: RejectRows does not exist. Available: IssuerId, IssuerId2, BusinessDate, StateCode, SourceName, NetworkName, NetworkURL, custnum, MarketCoverage, DentalOnlyPlan

at org.apache.spark.sql.types.StructType.$anonfun$fieldIndex$1(StructType.scala:306)

at scala.collection.immutable.HashMap$HashTrieMap.getOrElse0(HashMap.scala:596)

at scala.collection.immutable.HashMap.getOrElse(HashMap.scala:73)

at org.apache.spark.sql.types.StructType.fieldIndex(StructType.scala:305)

at org.apache.spark.sql.catalyst.OrderedFilters.$anonfun$predicates$4(OrderedFilters.scala:61)

at org.apache.spark.sql.catalyst.OrderedFilters.$anonfun$predicates$4$adapted(OrderedFilters.scala:61)

at scala.collection.TraversableLike.$anonfun$map$1(TraversableLike.scala:286)

at scala.collection.IndexedSeqOptimized.foreach(IndexedSeqOptimized.scala:36)

at scala.collection.IndexedSeqOptimized.foreach$(IndexedSeqOptimized.scala:33)

at scala.collection.mutable.ArrayOps$ofRef.foreach(ArrayOps.scala:198)

at scala.collection.TraversableLike.map(TraversableLike.scala:286)

at scala.collection.TraversableLike.map$(TraversableLike.scala:279)

at scala.collection.mutable.ArrayOps$ofRef.map(ArrayOps.scala:198)

at org.apache.spark.sql.catalyst.OrderedFilters.$anonfun$predicates$3(OrderedFilters.scala:61)

at org.apache.spark.sql.catalyst.OrderedFilters.$anonfun$predicates$3$adapted(OrderedFilters.scala:50)

at scala.collection.immutable.List.foreach(List.scala:431)

at org.apache.spark.sql.catalyst.OrderedFilters.<init>(OrderedFilters.scala:50)

at org.apache.spark.sql.catalyst.csv.UnivocityParser.<init>(UnivocityParser.scala:103)

at org.apache.spark.sql.execution.datasources.csv.CSVFileFormat.$anonfun$buildReader$1(CSVFileFormat.scala:130)

at org.apache.spark.sql.execution.datasources.FileFormat$$anon$1.apply(FileFormat.scala:148)

at org.apache.spark.sql.execution.datasources.FileFormat$$anon$1.apply(FileFormat.scala:133)

at org.apache.spark.sql.execution.datasources.FileScanRDD$$anon$1.org$apache$spark$sql$execution$datasources$FileScanRDD$$anon$$readCurrentFile(FileScanRDD.scala:127)

at org.apache.spark.sql.execution.datasources.FileScanRDD$$anon$1.nextIterator(FileScanRDD.scala:187)

at org.apache.spark.sql.execution.datasources.FileScanRDD$$anon$1.hasNext(FileScanRDD.scala:104)

at scala.collection.Iterator$$anon$10.hasNext(Iterator.scala:460)

at org.apache.spark.sql.catalyst.expressions.GeneratedClass$GeneratedIteratorForCodegenStage1.processNext(Unknown Source)

at org.apache.spark.sql.execution.BufferedRowIterator.hasNext(BufferedRowIterator.java:43)

at org.apache.spark.sql.execution.WholeStageCodegenExec$$anon$1.hasNext(WholeStageCodegenExec.scala:759)

at org.apache.spark.sql.execution.SparkPlan.$anonfun$getByteArrayRdd$1(SparkPlan.scala:349)

at org.apache.spark.rdd.RDD.$anonfun$mapPartitionsInternal$2(RDD.scala:898)

at org.apache.spark.rdd.RDD.$anonfun$mapPartitionsInternal$2$adapted(RDD.scala:898)

at org.apache.spark.rdd.MapPartitionsRDD.compute(MapPartitionsRDD.scala:52)

at org.apache.spark.rdd.RDD.computeOrReadCheckpoint(RDD.scala:373)

at org.apache.spark.rdd.RDD.iterator(RDD.scala:337)

at org.apache.spark.scheduler.ResultTask.runTask(ResultTask.scala:90)

at org.apache.spark.scheduler.Task.run(Task.scala:131)

at org.apache.spark.executor.Executor$TaskRunner.$anonfun$run$3(Executor.scala:506)

at org.apache.spark.util.Utils$.tryWithSafeFinally(Utils.scala:1462)

at org.apache.spark.executor.Executor$TaskRunner.run(Executor.scala:509)

at java.util.concurrent.ThreadPoolExecutor.runWorker(ThreadPoolExecutor.java:1149)

at java.util.concurrent.ThreadPoolExecutor$Worker.run(ThreadPoolExecutor.java:624)

at java.lang.Thread.run(Thread.java:750)

24/07/18 20:39:16 WARN scheduler.TaskSetManager: Lost task 0.0 in stage 17.0 (TID 17) (192.168.112.129 executor driver): java.lang.IllegalArgumentException: RejectRows does not exist. Available: IssuerId, IssuerId2, BusinessDate, StateCode, SourceName, NetworkName, NetworkURL, custnum, MarketCoverage, DentalOnlyPlan

at org.apache.spark.sql.types.StructType.$anonfun$fieldIndex$1(StructType.scala:306)

at scala.collection.immutable.HashMap$HashTrieMap.getOrElse0(HashMap.scala:596)

at scala.collection.immutable.HashMap.getOrElse(HashMap.scala:73)

at org.apache.spark.sql.types.StructType.fieldIndex(StructType.scala:305)

at org.apache.spark.sql.catalyst.OrderedFilters.$anonfun$predicates$4(OrderedFilters.scala:61)

at org.apache.spark.sql.catalyst.OrderedFilters.$anonfun$predicates$4$adapted(OrderedFilters.scala:61)

at scala.collection.TraversableLike.$anonfun$map$1(TraversableLike.scala:286)

at scala.collection.IndexedSeqOptimized.foreach(IndexedSeqOptimized.scala:36)

at scala.collection.IndexedSeqOptimized.foreach$(IndexedSeqOptimized.scala:33)

at scala.collection.mutable.ArrayOps$ofRef.foreach(ArrayOps.scala:198)

at scala.collection.TraversableLike.map(TraversableLike.scala:286)

at scala.collection.TraversableLike.map$(TraversableLike.scala:279)

at scala.collection.mutable.ArrayOps$ofRef.map(ArrayOps.scala:198)

at org.apache.spark.sql.catalyst.OrderedFilters.$anonfun$predicates$3(OrderedFilters.scala:61)

at org.apache.spark.sql.catalyst.OrderedFilters.$anonfun$predicates$3$adapted(OrderedFilters.scala:50)

at scala.collection.immutable.List.foreach(List.scala:431)

at org.apache.spark.sql.catalyst.OrderedFilters.<init>(OrderedFilters.scala:50)

at org.apache.spark.sql.catalyst.csv.UnivocityParser.<init>(UnivocityParser.scala:103)

at org.apache.spark.sql.execution.datasources.csv.CSVFileFormat.$anonfun$buildReader$1(CSVFileFormat.scala:130)

at org.apache.spark.sql.execution.datasources.FileFormat$$anon$1.apply(FileFormat.scala:148)

at org.apache.spark.sql.execution.datasources.FileFormat$$anon$1.apply(FileFormat.scala:133)

at org.apache.spark.sql.execution.datasources.FileScanRDD$$anon$1.org$apache$spark$sql$execution$datasources$FileScanRDD$$anon$$readCurrentFile(FileScanRDD.scala:127)

at org.apache.spark.sql.execution.datasources.FileScanRDD$$anon$1.nextIterator(FileScanRDD.scala:187)

at org.apache.spark.sql.execution.datasources.FileScanRDD$$anon$1.hasNext(FileScanRDD.scala:104)

at scala.collection.Iterator$$anon$10.hasNext(Iterator.scala:460)

at org.apache.spark.sql.catalyst.expressions.GeneratedClass$GeneratedIteratorForCodegenStage1.processNext(Unknown Source)

at org.apache.spark.sql.execution.BufferedRowIterator.hasNext(BufferedRowIterator.java:43)

at org.apache.spark.sql.execution.WholeStageCodegenExec$$anon$1.hasNext(WholeStageCodegenExec.scala:759)

at org.apache.spark.sql.execution.SparkPlan.$anonfun$getByteArrayRdd$1(SparkPlan.scala:349)

at org.apache.spark.rdd.RDD.$anonfun$mapPartitionsInternal$2(RDD.scala:898)

at org.apache.spark.rdd.RDD.$anonfun$mapPartitionsInternal$2$adapted(RDD.scala:898)

at org.apache.spark.rdd.MapPartitionsRDD.compute(MapPartitionsRDD.scala:52)

at org.apache.spark.rdd.RDD.computeOrReadCheckpoint(RDD.scala:373)

at org.apache.spark.rdd.RDD.iterator(RDD.scala:337)

at org.apache.spark.scheduler.ResultTask.runTask(ResultTask.scala:90)

at org.apache.spark.scheduler.Task.run(Task.scala:131)

at org.apache.spark.executor.Executor$TaskRunner.$anonfun$run$3(Executor.scala:506)

at org.apache.spark.util.Utils$.tryWithSafeFinally(Utils.scala:1462)

at org.apache.spark.executor.Executor$TaskRunner.run(Executor.scala:509)

at java.util.concurrent.ThreadPoolExecutor.runWorker(ThreadPoolExecutor.java:1149)

at java.util.concurrent.ThreadPoolExecutor$Worker.run(ThreadPoolExecutor.java:624)

at java.lang.Thread.run(Thread.java:750)

24/07/18 20:39:16 ERROR scheduler.TaskSetManager: Task 0 in stage 17.0 failed 1 times; aborting job

Traceback (most recent call last):

File "<stdin>", line 1, in <module>

File "/usr/local/spark/python/pyspark/sql/dataframe.py", line 494, in show

print(self.\_jdf.showString(n, 20, vertical))

File "/usr/local/spark/python/lib/py4j-0.10.9.3-src.zip/py4j/java\_gateway.py", line 1322, in \_\_call\_\_

File "/usr/local/spark/python/pyspark/sql/utils.py", line 111, in deco

return f(\*a, \*\*kw)

File "/usr/local/spark/python/lib/py4j-0.10.9.3-src.zip/py4j/protocol.py", line 328, in get\_return\_value

py4j.protocol.Py4JJavaError: An error occurred while calling o232.showString.

: org.apache.spark.SparkException: Job aborted due to stage failure: Task 0 in stage 17.0 failed 1 times, most recent failure: Lost task 0.0 in stage 17.0 (TID 17) (192.168.112.129 executor driver): java.lang.IllegalArgumentException: RejectRows does not exist. Available: IssuerId, IssuerId2, BusinessDate, StateCode, SourceName, NetworkName, NetworkURL, custnum, MarketCoverage, DentalOnlyPlan

at org.apache.spark.sql.types.StructType.$anonfun$fieldIndex$1(StructType.scala:306)

at scala.collection.immutable.HashMap$HashTrieMap.getOrElse0(HashMap.scala:596)

at scala.collection.immutable.HashMap.getOrElse(HashMap.scala:73)

at org.apache.spark.sql.types.StructType.fieldIndex(StructType.scala:305)

at org.apache.spark.sql.catalyst.OrderedFilters.$anonfun$predicates$4(OrderedFilters.scala:61)

at org.apache.spark.sql.catalyst.OrderedFilters.$anonfun$predicates$4$adapted(OrderedFilters.scala:61)

at scala.collection.TraversableLike.$anonfun$map$1(TraversableLike.scala:286)

at scala.collection.IndexedSeqOptimized.foreach(IndexedSeqOptimized.scala:36)

at scala.collection.IndexedSeqOptimized.foreach$(IndexedSeqOptimized.scala:33)

at scala.collection.mutable.ArrayOps$ofRef.foreach(ArrayOps.scala:198)

at scala.collection.TraversableLike.map(TraversableLike.scala:286)

at scala.collection.TraversableLike.map$(TraversableLike.scala:279)

at scala.collection.mutable.ArrayOps$ofRef.map(ArrayOps.scala:198)

at org.apache.spark.sql.catalyst.OrderedFilters.$anonfun$predicates$3(OrderedFilters.scala:61)

at org.apache.spark.sql.catalyst.OrderedFilters.$anonfun$predicates$3$adapted(OrderedFilters.scala:50)

at scala.collection.immutable.List.foreach(List.scala:431)

at org.apache.spark.sql.catalyst.OrderedFilters.<init>(OrderedFilters.scala:50)

at org.apache.spark.sql.catalyst.csv.UnivocityParser.<init>(UnivocityParser.scala:103)

at org.apache.spark.sql.execution.datasources.csv.CSVFileFormat.$anonfun$buildReader$1(CSVFileFormat.scala:130)

at org.apache.spark.sql.execution.datasources.FileFormat$$anon$1.apply(FileFormat.scala:148)

at org.apache.spark.sql.execution.datasources.FileFormat$$anon$1.apply(FileFormat.scala:133)

at org.apache.spark.sql.execution.datasources.FileScanRDD$$anon$1.org$apache$spark$sql$execution$datasources$FileScanRDD$$anon$$readCurrentFile(FileScanRDD.scala:127)

at org.apache.spark.sql.execution.datasources.FileScanRDD$$anon$1.nextIterator(FileScanRDD.scala:187)

at org.apache.spark.sql.execution.datasources.FileScanRDD$$anon$1.hasNext(FileScanRDD.scala:104)

at scala.collection.Iterator$$anon$10.hasNext(Iterator.scala:460)

at org.apache.spark.sql.catalyst.expressions.GeneratedClass$GeneratedIteratorForCodegenStage1.processNext(Unknown Source)

at org.apache.spark.sql.execution.BufferedRowIterator.hasNext(BufferedRowIterator.java:43)

at org.apache.spark.sql.execution.WholeStageCodegenExec$$anon$1.hasNext(WholeStageCodegenExec.scala:759)

at org.apache.spark.sql.execution.SparkPlan.$anonfun$getByteArrayRdd$1(SparkPlan.scala:349)

at org.apache.spark.rdd.RDD.$anonfun$mapPartitionsInternal$2(RDD.scala:898)

at org.apache.spark.rdd.RDD.$anonfun$mapPartitionsInternal$2$adapted(RDD.scala:898)

at org.apache.spark.rdd.MapPartitionsRDD.compute(MapPartitionsRDD.scala:52)

at org.apache.spark.rdd.RDD.computeOrReadCheckpoint(RDD.scala:373)

at org.apache.spark.rdd.RDD.iterator(RDD.scala:337)

at org.apache.spark.scheduler.ResultTask.runTask(ResultTask.scala:90)

at org.apache.spark.scheduler.Task.run(Task.scala:131)

at org.apache.spark.executor.Executor$TaskRunner.$anonfun$run$3(Executor.scala:506)

at org.apache.spark.util.Utils$.tryWithSafeFinally(Utils.scala:1462)

at org.apache.spark.executor.Executor$TaskRunner.run(Executor.scala:509)

at java.util.concurrent.ThreadPoolExecutor.runWorker(ThreadPoolExecutor.java:1149)

at java.util.concurrent.ThreadPoolExecutor$Worker.run(ThreadPoolExecutor.java:624)

at java.lang.Thread.run(Thread.java:750)

Driver stacktrace:

at org.apache.spark.scheduler.DAGScheduler.failJobAndIndependentStages(DAGScheduler.scala:2454)

at org.apache.spark.scheduler.DAGScheduler.$anonfun$abortStage$2(DAGScheduler.scala:2403)

at org.apache.spark.scheduler.DAGScheduler.$anonfun$abortStage$2$adapted(DAGScheduler.scala:2402)

at scala.collection.mutable.ResizableArray.foreach(ResizableArray.scala:62)

at scala.collection.mutable.ResizableArray.foreach$(ResizableArray.scala:55)

at scala.collection.mutable.ArrayBuffer.foreach(ArrayBuffer.scala:49)

at org.apache.spark.scheduler.DAGScheduler.abortStage(DAGScheduler.scala:2402)

at org.apache.spark.scheduler.DAGScheduler.$anonfun$handleTaskSetFailed$1(DAGScheduler.scala:1160)

at org.apache.spark.scheduler.DAGScheduler.$anonfun$handleTaskSetFailed$1$adapted(DAGScheduler.scala:1160)

at scala.Option.foreach(Option.scala:407)

at org.apache.spark.scheduler.DAGScheduler.handleTaskSetFailed(DAGScheduler.scala:1160)

at org.apache.spark.scheduler.DAGSchedulerEventProcessLoop.doOnReceive(DAGScheduler.scala:2642)

at org.apache.spark.scheduler.DAGSchedulerEventProcessLoop.onReceive(DAGScheduler.scala:2584)

at org.apache.spark.scheduler.DAGSchedulerEventProcessLoop.onReceive(DAGScheduler.scala:2573)

at org.apache.spark.util.EventLoop$$anon$1.run(EventLoop.scala:49)

at org.apache.spark.scheduler.DAGScheduler.runJob(DAGScheduler.scala:938)

at org.apache.spark.SparkContext.runJob(SparkContext.scala:2214)

at org.apache.spark.SparkContext.runJob(SparkContext.scala:2235)

at org.apache.spark.SparkContext.runJob(SparkContext.scala:2254)

at org.apache.spark.sql.execution.SparkPlan.executeTake(SparkPlan.scala:476)

at org.apache.spark.sql.execution.SparkPlan.executeTake(SparkPlan.scala:429)

at org.apache.spark.sql.execution.CollectLimitExec.executeCollect(limit.scala:48)

at org.apache.spark.sql.Dataset.collectFromPlan(Dataset.scala:3715)

at org.apache.spark.sql.Dataset.$anonfun$head$1(Dataset.scala:2728)

at org.apache.spark.sql.Dataset.$anonfun$withAction$1(Dataset.scala:3706)

at org.apache.spark.sql.execution.SQLExecution$.$anonfun$withNewExecutionId$5(SQLExecution.scala:103)

at org.apache.spark.sql.execution.SQLExecution$.withSQLConfPropagated(SQLExecution.scala:163)

at org.apache.spark.sql.execution.SQLExecution$.$anonfun$withNewExecutionId$1(SQLExecution.scala:90)

at org.apache.spark.sql.SparkSession.withActive(SparkSession.scala:775)

at org.apache.spark.sql.execution.SQLExecution$.withNewExecutionId(SQLExecution.scala:64)

at org.apache.spark.sql.Dataset.withAction(Dataset.scala:3704)

at org.apache.spark.sql.Dataset.head(Dataset.scala:2728)

at org.apache.spark.sql.Dataset.take(Dataset.scala:2935)

at org.apache.spark.sql.Dataset.getRows(Dataset.scala:287)

at org.apache.spark.sql.Dataset.showString(Dataset.scala:326)

at sun.reflect.GeneratedMethodAccessor41.invoke(Unknown Source)

at sun.reflect.DelegatingMethodAccessorImpl.invoke(DelegatingMethodAccessorImpl.java:43)

at java.lang.reflect.Method.invoke(Method.java:498)

at py4j.reflection.MethodInvoker.invoke(MethodInvoker.java:244)

at py4j.reflection.ReflectionEngine.invoke(ReflectionEngine.java:357)

at py4j.Gateway.invoke(Gateway.java:282)

at py4j.commands.AbstractCommand.invokeMethod(AbstractCommand.java:132)

at py4j.commands.CallCommand.execute(CallCommand.java:79)

at py4j.ClientServerConnection.waitForCommands(ClientServerConnection.java:182)

at py4j.ClientServerConnection.run(ClientServerConnection.java:106)

at java.lang.Thread.run(Thread.java:750)

Caused by: java.lang.IllegalArgumentException: RejectRows does not exist. Available: IssuerId, IssuerId2, BusinessDate, StateCode, SourceName, NetworkName, NetworkURL, custnum, MarketCoverage, DentalOnlyPlan

at org.apache.spark.sql.types.StructType.$anonfun$fieldIndex$1(StructType.scala:306)

at scala.collection.immutable.HashMap$HashTrieMap.getOrElse0(HashMap.scala:596)

at scala.collection.immutable.HashMap.getOrElse(HashMap.scala:73)

at org.apache.spark.sql.types.StructType.fieldIndex(StructType.scala:305)

at org.apache.spark.sql.catalyst.OrderedFilters.$anonfun$predicates$4(OrderedFilters.scala:61)

at org.apache.spark.sql.catalyst.OrderedFilters.$anonfun$predicates$4$adapted(OrderedFilters.scala:61)

at scala.collection.TraversableLike.$anonfun$map$1(TraversableLike.scala:286)

at scala.collection.IndexedSeqOptimized.foreach(IndexedSeqOptimized.scala:36)

at scala.collection.IndexedSeqOptimized.foreach$(IndexedSeqOptimized.scala:33)

at scala.collection.mutable.ArrayOps$ofRef.foreach(ArrayOps.scala:198)

at scala.collection.TraversableLike.map(TraversableLike.scala:286)

at scala.collection.TraversableLike.map$(TraversableLike.scala:279)

at scala.collection.mutable.ArrayOps$ofRef.map(ArrayOps.scala:198)

at org.apache.spark.sql.catalyst.OrderedFilters.$anonfun$predicates$3(OrderedFilters.scala:61)

at org.apache.spark.sql.catalyst.OrderedFilters.$anonfun$predicates$3$adapted(OrderedFilters.scala:50)

at scala.collection.immutable.List.foreach(List.scala:431)

at org.apache.spark.sql.catalyst.OrderedFilters.<init>(OrderedFilters.scala:50)

at org.apache.spark.sql.catalyst.csv.UnivocityParser.<init>(UnivocityParser.scala:103)

at org.apache.spark.sql.execution.datasources.csv.CSVFileFormat.$anonfun$buildReader$1(CSVFileFormat.scala:130)

at org.apache.spark.sql.execution.datasources.FileFormat$$anon$1.apply(FileFormat.scala:148)

at org.apache.spark.sql.execution.datasources.FileFormat$$anon$1.apply(FileFormat.scala:133)

at org.apache.spark.sql.execution.datasources.FileScanRDD$$anon$1.org$apache$spark$sql$execution$datasources$FileScanRDD$$anon$$readCurrentFile(FileScanRDD.scala:127)

at org.apache.spark.sql.execution.datasources.FileScanRDD$$anon$1.nextIterator(FileScanRDD.scala:187)

at org.apache.spark.sql.execution.datasources.FileScanRDD$$anon$1.hasNext(FileScanRDD.scala:104)

at scala.collection.Iterator$$anon$10.hasNext(Iterator.scala:460)

at org.apache.spark.sql.catalyst.expressions.GeneratedClass$GeneratedIteratorForCodegenStage1.processNext(Unknown Source)

at org.apache.spark.sql.execution.BufferedRowIterator.hasNext(BufferedRowIterator.java:43)

at org.apache.spark.sql.execution.WholeStageCodegenExec$$anon$1.hasNext(WholeStageCodegenExec.scala:759)

at org.apache.spark.sql.execution.SparkPlan.$anonfun$getByteArrayRdd$1(SparkPlan.scala:349)

at org.apache.spark.rdd.RDD.$anonfun$mapPartitionsInternal$2(RDD.scala:898)

at org.apache.spark.rdd.RDD.$anonfun$mapPartitionsInternal$2$adapted(RDD.scala:898)

at org.apache.spark.rdd.MapPartitionsRDD.compute(MapPartitionsRDD.scala:52)

at org.apache.spark.rdd.RDD.computeOrReadCheckpoint(RDD.scala:373)

at org.apache.spark.rdd.RDD.iterator(RDD.scala:337)

at org.apache.spark.scheduler.ResultTask.runTask(ResultTask.scala:90)

at org.apache.spark.scheduler.Task.run(Task.scala:131)

at org.apache.spark.executor.Executor$TaskRunner.$anonfun$run$3(Executor.scala:506)

at org.apache.spark.util.Utils$.tryWithSafeFinally(Utils.scala:1462)

at org.apache.spark.executor.Executor$TaskRunner.run(Executor.scala:509)

at java.util.concurrent.ThreadPoolExecutor.runWorker(ThreadPoolExecutor.java:1149)

at java.util.concurrent.ThreadPoolExecutor$Worker.run(ThreadPoolExecutor.java:624)

... 1 more

>>> data\_frame\_1.printSchema

<bound method DataFrame.printSchema of DataFrame[IssuerId: int, IssuerId2: int, BusinessDate: date, StateCode: string, SourceName: string, NetworkName: string, NetworkURL: string, custnum: string, MarketCoverage: string, DentalOnlyPlan: string, RejectRows: string]>

>>> custom\_schema=StructType ([StructField("IssuerId",IntegerType(),True) , StructField("IssuerId2",IntegerType(),True) , StructField("BusinessDate",DateType(),True) , StructField("StateCode",StringType(),True) , StructField("SourceName",StringType(),True) , StructField("NetworkName",StringType(),True) , StructField("NetworkURL",StringType(),True) , StructField("custnum",StringType(),True) , StructField("MarketCoverage",StringType(),True) , StructField("DentalOnlyPlan",StringType(),True) , StructField("RejectRows",StringType(),True) ])

>>> data\_frame\_1=spark.read.csv(path="/user/hduser/Hackathon/insuranceinfo1.csv" , header=True , schema=custom\_schema , columnNameOfCorruptRecord="RejectRows" ,dateFormat="MM-dd-yyyy")

>>> data\_frame\_1.show(3)

+--------+---------+------------+---------+----------+------------+--------------------+-------+--------------+--------------+--------------------+

|IssuerId|IssuerId2|BusinessDate|StateCode|SourceName| NetworkName| NetworkURL|custnum|MarketCoverage|DentalOnlyPlan| RejectRows|

+--------+---------+------------+---------+----------+------------+--------------------+-------+--------------+--------------+--------------------+

| 21989| 21989| 2019-10-01| AK| HIOS| ODS Premier|https://www.modah...| 13| null| null| null|

| 219| 219| 2019-10-01| AK| HIOS| ODS Premier| null| null| null| null|219,219,2019-10-0...|

| 38344| 38344| 2019-10-01| AK| HIOS|HeritagePlus|https://www.preme...| 13| null| null| null|

+--------+---------+------------+---------+----------+------------+--------------------+-------+--------------+--------------+--------------------+

only showing top 3 rows

>>> data\_frame\_1.where("RejectRows is not 'null'").show(3)

Traceback (most recent call last):

File "<stdin>", line 1, in <module>

File "/usr/local/spark/python/pyspark/sql/dataframe.py", line 1731, in filter

jdf = self.\_jdf.filter(condition)

File "/usr/local/spark/python/lib/py4j-0.10.9.3-src.zip/py4j/java\_gateway.py", line 1322, in \_\_call\_\_

File "/usr/local/spark/python/pyspark/sql/utils.py", line 117, in deco

raise converted from None

pyspark.sql.utils.ParseException:

no viable alternative at input 'is not 'null''(line 1, pos 18)

== SQL ==

RejectRows is not 'null'

------------------^^^

>>> data\_frame\_1.where("RejectRows != 'null'").show(3)

24/07/18 20:41:13 ERROR executor.Executor: Exception in task 0.0 in stage 19.0 (TID 19)

java.lang.IllegalArgumentException: RejectRows does not exist. Available: IssuerId, IssuerId2, BusinessDate, StateCode, SourceName, NetworkName, NetworkURL, custnum, MarketCoverage, DentalOnlyPlan

at org.apache.spark.sql.types.StructType.$anonfun$fieldIndex$1(StructType.scala:306)

at scala.collection.immutable.HashMap$HashTrieMap.getOrElse0(HashMap.scala:596)

at scala.collection.immutable.HashMap.getOrElse(HashMap.scala:73)

at org.apache.spark.sql.types.StructType.fieldIndex(StructType.scala:305)

at org.apache.spark.sql.catalyst.OrderedFilters.$anonfun$predicates$4(OrderedFilters.scala:61)

at org.apache.spark.sql.catalyst.OrderedFilters.$anonfun$predicates$4$adapted(OrderedFilters.scala:61)

at scala.collection.TraversableLike.$anonfun$map$1(TraversableLike.scala:286)

at scala.collection.IndexedSeqOptimized.foreach(IndexedSeqOptimized.scala:36)

at scala.collection.IndexedSeqOptimized.foreach$(IndexedSeqOptimized.scala:33)

at scala.collection.mutable.ArrayOps$ofRef.foreach(ArrayOps.scala:198)

at scala.collection.TraversableLike.map(TraversableLike.scala:286)

at scala.collection.TraversableLike.map$(TraversableLike.scala:279)

at scala.collection.mutable.ArrayOps$ofRef.map(ArrayOps.scala:198)

at org.apache.spark.sql.catalyst.OrderedFilters.$anonfun$predicates$3(OrderedFilters.scala:61)

at org.apache.spark.sql.catalyst.OrderedFilters.$anonfun$predicates$3$adapted(OrderedFilters.scala:50)

at scala.collection.immutable.List.foreach(List.scala:431)

at org.apache.spark.sql.catalyst.OrderedFilters.<init>(OrderedFilters.scala:50)

at org.apache.spark.sql.catalyst.csv.UnivocityParser.<init>(UnivocityParser.scala:103)

at org.apache.spark.sql.execution.datasources.csv.CSVFileFormat.$anonfun$buildReader$1(CSVFileFormat.scala:130)

at org.apache.spark.sql.execution.datasources.FileFormat$$anon$1.apply(FileFormat.scala:148)

at org.apache.spark.sql.execution.datasources.FileFormat$$anon$1.apply(FileFormat.scala:133)

at org.apache.spark.sql.execution.datasources.FileScanRDD$$anon$1.org$apache$spark$sql$execution$datasources$FileScanRDD$$anon$$readCurrentFile(FileScanRDD.scala:127)

at org.apache.spark.sql.execution.datasources.FileScanRDD$$anon$1.nextIterator(FileScanRDD.scala:187)

at org.apache.spark.sql.execution.datasources.FileScanRDD$$anon$1.hasNext(FileScanRDD.scala:104)

at scala.collection.Iterator$$anon$10.hasNext(Iterator.scala:460)

at org.apache.spark.sql.catalyst.expressions.GeneratedClass$GeneratedIteratorForCodegenStage1.processNext(Unknown Source)

at org.apache.spark.sql.execution.BufferedRowIterator.hasNext(BufferedRowIterator.java:43)

at org.apache.spark.sql.execution.WholeStageCodegenExec$$anon$1.hasNext(WholeStageCodegenExec.scala:759)

at org.apache.spark.sql.execution.SparkPlan.$anonfun$getByteArrayRdd$1(SparkPlan.scala:349)

at org.apache.spark.rdd.RDD.$anonfun$mapPartitionsInternal$2(RDD.scala:898)

at org.apache.spark.rdd.RDD.$anonfun$mapPartitionsInternal$2$adapted(RDD.scala:898)

at org.apache.spark.rdd.MapPartitionsRDD.compute(MapPartitionsRDD.scala:52)

at org.apache.spark.rdd.RDD.computeOrReadCheckpoint(RDD.scala:373)

at org.apache.spark.rdd.RDD.iterator(RDD.scala:337)

at org.apache.spark.scheduler.ResultTask.runTask(ResultTask.scala:90)

at org.apache.spark.scheduler.Task.run(Task.scala:131)

at org.apache.spark.executor.Executor$TaskRunner.$anonfun$run$3(Executor.scala:506)

at org.apache.spark.util.Utils$.tryWithSafeFinally(Utils.scala:1462)

at org.apache.spark.executor.Executor$TaskRunner.run(Executor.scala:509)

at java.util.concurrent.ThreadPoolExecutor.runWorker(ThreadPoolExecutor.java:1149)

at java.util.concurrent.ThreadPoolExecutor$Worker.run(ThreadPoolExecutor.java:624)

at java.lang.Thread.run(Thread.java:750)

24/07/18 20:41:13 WARN scheduler.TaskSetManager: Lost task 0.0 in stage 19.0 (TID 19) (192.168.112.129 executor driver): java.lang.IllegalArgumentException: RejectRows does not exist. Available: IssuerId, IssuerId2, BusinessDate, StateCode, SourceName, NetworkName, NetworkURL, custnum, MarketCoverage, DentalOnlyPlan

at org.apache.spark.sql.types.StructType.$anonfun$fieldIndex$1(StructType.scala:306)

at scala.collection.immutable.HashMap$HashTrieMap.getOrElse0(HashMap.scala:596)

at scala.collection.immutable.HashMap.getOrElse(HashMap.scala:73)

at org.apache.spark.sql.types.StructType.fieldIndex(StructType.scala:305)

at org.apache.spark.sql.catalyst.OrderedFilters.$anonfun$predicates$4(OrderedFilters.scala:61)

at org.apache.spark.sql.catalyst.OrderedFilters.$anonfun$predicates$4$adapted(OrderedFilters.scala:61)

at scala.collection.TraversableLike.$anonfun$map$1(TraversableLike.scala:286)

at scala.collection.IndexedSeqOptimized.foreach(IndexedSeqOptimized.scala:36)

at scala.collection.IndexedSeqOptimized.foreach$(IndexedSeqOptimized.scala:33)

at scala.collection.mutable.ArrayOps$ofRef.foreach(ArrayOps.scala:198)

at scala.collection.TraversableLike.map(TraversableLike.scala:286)

at scala.collection.TraversableLike.map$(TraversableLike.scala:279)

at scala.collection.mutable.ArrayOps$ofRef.map(ArrayOps.scala:198)

at org.apache.spark.sql.catalyst.OrderedFilters.$anonfun$predicates$3(OrderedFilters.scala:61)

at org.apache.spark.sql.catalyst.OrderedFilters.$anonfun$predicates$3$adapted(OrderedFilters.scala:50)

at scala.collection.immutable.List.foreach(List.scala:431)

at org.apache.spark.sql.catalyst.OrderedFilters.<init>(OrderedFilters.scala:50)

at org.apache.spark.sql.catalyst.csv.UnivocityParser.<init>(UnivocityParser.scala:103)

at org.apache.spark.sql.execution.datasources.csv.CSVFileFormat.$anonfun$buildReader$1(CSVFileFormat.scala:130)

at org.apache.spark.sql.execution.datasources.FileFormat$$anon$1.apply(FileFormat.scala:148)

at org.apache.spark.sql.execution.datasources.FileFormat$$anon$1.apply(FileFormat.scala:133)

at org.apache.spark.sql.execution.datasources.FileScanRDD$$anon$1.org$apache$spark$sql$execution$datasources$FileScanRDD$$anon$$readCurrentFile(FileScanRDD.scala:127)

at org.apache.spark.sql.execution.datasources.FileScanRDD$$anon$1.nextIterator(FileScanRDD.scala:187)

at org.apache.spark.sql.execution.datasources.FileScanRDD$$anon$1.hasNext(FileScanRDD.scala:104)

at scala.collection.Iterator$$anon$10.hasNext(Iterator.scala:460)

at org.apache.spark.sql.catalyst.expressions.GeneratedClass$GeneratedIteratorForCodegenStage1.processNext(Unknown Source)

at org.apache.spark.sql.execution.BufferedRowIterator.hasNext(BufferedRowIterator.java:43)

at org.apache.spark.sql.execution.WholeStageCodegenExec$$anon$1.hasNext(WholeStageCodegenExec.scala:759)

at org.apache.spark.sql.execution.SparkPlan.$anonfun$getByteArrayRdd$1(SparkPlan.scala:349)

at org.apache.spark.rdd.RDD.$anonfun$mapPartitionsInternal$2(RDD.scala:898)

at org.apache.spark.rdd.RDD.$anonfun$mapPartitionsInternal$2$adapted(RDD.scala:898)

at org.apache.spark.rdd.MapPartitionsRDD.compute(MapPartitionsRDD.scala:52)

at org.apache.spark.rdd.RDD.computeOrReadCheckpoint(RDD.scala:373)

at org.apache.spark.rdd.RDD.iterator(RDD.scala:337)

at org.apache.spark.scheduler.ResultTask.runTask(ResultTask.scala:90)

at org.apache.spark.scheduler.Task.run(Task.scala:131)

at org.apache.spark.executor.Executor$TaskRunner.$anonfun$run$3(Executor.scala:506)

at org.apache.spark.util.Utils$.tryWithSafeFinally(Utils.scala:1462)

at org.apache.spark.executor.Executor$TaskRunner.run(Executor.scala:509)

at java.util.concurrent.ThreadPoolExecutor.runWorker(ThreadPoolExecutor.java:1149)

at java.util.concurrent.ThreadPoolExecutor$Worker.run(ThreadPoolExecutor.java:624)

at java.lang.Thread.run(Thread.java:750)

24/07/18 20:41:13 ERROR scheduler.TaskSetManager: Task 0 in stage 19.0 failed 1 times; aborting job

Traceback (most recent call last):

File "<stdin>", line 1, in <module>

File "/usr/local/spark/python/pyspark/sql/dataframe.py", line 494, in show

print(self.\_jdf.showString(n, 20, vertical))

File "/usr/local/spark/python/lib/py4j-0.10.9.3-src.zip/py4j/java\_gateway.py", line 1322, in \_\_call\_\_

File "/usr/local/spark/python/pyspark/sql/utils.py", line 111, in deco

return f(\*a, \*\*kw)

File "/usr/local/spark/python/lib/py4j-0.10.9.3-src.zip/py4j/protocol.py", line 328, in get\_return\_value

py4j.protocol.Py4JJavaError: An error occurred while calling o254.showString.

: org.apache.spark.SparkException: Job aborted due to stage failure: Task 0 in stage 19.0 failed 1 times, most recent failure: Lost task 0.0 in stage 19.0 (TID 19) (192.168.112.129 executor driver): java.lang.IllegalArgumentException: RejectRows does not exist. Available: IssuerId, IssuerId2, BusinessDate, StateCode, SourceName, NetworkName, NetworkURL, custnum, MarketCoverage, DentalOnlyPlan

at org.apache.spark.sql.types.StructType.$anonfun$fieldIndex$1(StructType.scala:306)

at scala.collection.immutable.HashMap$HashTrieMap.getOrElse0(HashMap.scala:596)

at scala.collection.immutable.HashMap.getOrElse(HashMap.scala:73)

at org.apache.spark.sql.types.StructType.fieldIndex(StructType.scala:305)

at org.apache.spark.sql.catalyst.OrderedFilters.$anonfun$predicates$4(OrderedFilters.scala:61)

at org.apache.spark.sql.catalyst.OrderedFilters.$anonfun$predicates$4$adapted(OrderedFilters.scala:61)

at scala.collection.TraversableLike.$anonfun$map$1(TraversableLike.scala:286)

at scala.collection.IndexedSeqOptimized.foreach(IndexedSeqOptimized.scala:36)

at scala.collection.IndexedSeqOptimized.foreach$(IndexedSeqOptimized.scala:33)

at scala.collection.mutable.ArrayOps$ofRef.foreach(ArrayOps.scala:198)

at scala.collection.TraversableLike.map(TraversableLike.scala:286)

at scala.collection.TraversableLike.map$(TraversableLike.scala:279)

at scala.collection.mutable.ArrayOps$ofRef.map(ArrayOps.scala:198)

at org.apache.spark.sql.catalyst.OrderedFilters.$anonfun$predicates$3(OrderedFilters.scala:61)

at org.apache.spark.sql.catalyst.OrderedFilters.$anonfun$predicates$3$adapted(OrderedFilters.scala:50)

at scala.collection.immutable.List.foreach(List.scala:431)

at org.apache.spark.sql.catalyst.OrderedFilters.<init>(OrderedFilters.scala:50)

at org.apache.spark.sql.catalyst.csv.UnivocityParser.<init>(UnivocityParser.scala:103)

at org.apache.spark.sql.execution.datasources.csv.CSVFileFormat.$anonfun$buildReader$1(CSVFileFormat.scala:130)

at org.apache.spark.sql.execution.datasources.FileFormat$$anon$1.apply(FileFormat.scala:148)

at org.apache.spark.sql.execution.datasources.FileFormat$$anon$1.apply(FileFormat.scala:133)

at org.apache.spark.sql.execution.datasources.FileScanRDD$$anon$1.org$apache$spark$sql$execution$datasources$FileScanRDD$$anon$$readCurrentFile(FileScanRDD.scala:127)

at org.apache.spark.sql.execution.datasources.FileScanRDD$$anon$1.nextIterator(FileScanRDD.scala:187)

at org.apache.spark.sql.execution.datasources.FileScanRDD$$anon$1.hasNext(FileScanRDD.scala:104)

at scala.collection.Iterator$$anon$10.hasNext(Iterator.scala:460)

at org.apache.spark.sql.catalyst.expressions.GeneratedClass$GeneratedIteratorForCodegenStage1.processNext(Unknown Source)

at org.apache.spark.sql.execution.BufferedRowIterator.hasNext(BufferedRowIterator.java:43)

at org.apache.spark.sql.execution.WholeStageCodegenExec$$anon$1.hasNext(WholeStageCodegenExec.scala:759)

at org.apache.spark.sql.execution.SparkPlan.$anonfun$getByteArrayRdd$1(SparkPlan.scala:349)

at org.apache.spark.rdd.RDD.$anonfun$mapPartitionsInternal$2(RDD.scala:898)

at org.apache.spark.rdd.RDD.$anonfun$mapPartitionsInternal$2$adapted(RDD.scala:898)

at org.apache.spark.rdd.MapPartitionsRDD.compute(MapPartitionsRDD.scala:52)

at org.apache.spark.rdd.RDD.computeOrReadCheckpoint(RDD.scala:373)

at org.apache.spark.rdd.RDD.iterator(RDD.scala:337)

at org.apache.spark.scheduler.ResultTask.runTask(ResultTask.scala:90)

at org.apache.spark.scheduler.Task.run(Task.scala:131)

at org.apache.spark.executor.Executor$TaskRunner.$anonfun$run$3(Executor.scala:506)

at org.apache.spark.util.Utils$.tryWithSafeFinally(Utils.scala:1462)

at org.apache.spark.executor.Executor$TaskRunner.run(Executor.scala:509)

at java.util.concurrent.ThreadPoolExecutor.runWorker(ThreadPoolExecutor.java:1149)

at java.util.concurrent.ThreadPoolExecutor$Worker.run(ThreadPoolExecutor.java:624)

at java.lang.Thread.run(Thread.java:750)

Driver stacktrace:

at org.apache.spark.scheduler.DAGScheduler.failJobAndIndependentStages(DAGScheduler.scala:2454)

at org.apache.spark.scheduler.DAGScheduler.$anonfun$abortStage$2(DAGScheduler.scala:2403)

at org.apache.spark.scheduler.DAGScheduler.$anonfun$abortStage$2$adapted(DAGScheduler.scala:2402)

at scala.collection.mutable.ResizableArray.foreach(ResizableArray.scala:62)

at scala.collection.mutable.ResizableArray.foreach$(ResizableArray.scala:55)

at scala.collection.mutable.ArrayBuffer.foreach(ArrayBuffer.scala:49)

at org.apache.spark.scheduler.DAGScheduler.abortStage(DAGScheduler.scala:2402)

at org.apache.spark.scheduler.DAGScheduler.$anonfun$handleTaskSetFailed$1(DAGScheduler.scala:1160)

at org.apache.spark.scheduler.DAGScheduler.$anonfun$handleTaskSetFailed$1$adapted(DAGScheduler.scala:1160)

at scala.Option.foreach(Option.scala:407)

at org.apache.spark.scheduler.DAGScheduler.handleTaskSetFailed(DAGScheduler.scala:1160)

at org.apache.spark.scheduler.DAGSchedulerEventProcessLoop.doOnReceive(DAGScheduler.scala:2642)

at org.apache.spark.scheduler.DAGSchedulerEventProcessLoop.onReceive(DAGScheduler.scala:2584)

at org.apache.spark.scheduler.DAGSchedulerEventProcessLoop.onReceive(DAGScheduler.scala:2573)

at org.apache.spark.util.EventLoop$$anon$1.run(EventLoop.scala:49)

at org.apache.spark.scheduler.DAGScheduler.runJob(DAGScheduler.scala:938)

at org.apache.spark.SparkContext.runJob(SparkContext.scala:2214)

at org.apache.spark.SparkContext.runJob(SparkContext.scala:2235)

at org.apache.spark.SparkContext.runJob(SparkContext.scala:2254)

at org.apache.spark.sql.execution.SparkPlan.executeTake(SparkPlan.scala:476)

at org.apache.spark.sql.execution.SparkPlan.executeTake(SparkPlan.scala:429)

at org.apache.spark.sql.execution.CollectLimitExec.executeCollect(limit.scala:48)

at org.apache.spark.sql.Dataset.collectFromPlan(Dataset.scala:3715)

at org.apache.spark.sql.Dataset.$anonfun$head$1(Dataset.scala:2728)

at org.apache.spark.sql.Dataset.$anonfun$withAction$1(Dataset.scala:3706)

at org.apache.spark.sql.execution.SQLExecution$.$anonfun$withNewExecutionId$5(SQLExecution.scala:103)

at org.apache.spark.sql.execution.SQLExecution$.withSQLConfPropagated(SQLExecution.scala:163)

at org.apache.spark.sql.execution.SQLExecution$.$anonfun$withNewExecutionId$1(SQLExecution.scala:90)

at org.apache.spark.sql.SparkSession.withActive(SparkSession.scala:775)

at org.apache.spark.sql.execution.SQLExecution$.withNewExecutionId(SQLExecution.scala:64)

at org.apache.spark.sql.Dataset.withAction(Dataset.scala:3704)

at org.apache.spark.sql.Dataset.head(Dataset.scala:2728)

at org.apache.spark.sql.Dataset.take(Dataset.scala:2935)

at org.apache.spark.sql.Dataset.getRows(Dataset.scala:287)

at org.apache.spark.sql.Dataset.showString(Dataset.scala:326)

at sun.reflect.GeneratedMethodAccessor41.invoke(Unknown Source)

at sun.reflect.DelegatingMethodAccessorImpl.invoke(DelegatingMethodAccessorImpl.java:43)

at java.lang.reflect.Method.invoke(Method.java:498)

at py4j.reflection.MethodInvoker.invoke(MethodInvoker.java:244)

at py4j.reflection.ReflectionEngine.invoke(ReflectionEngine.java:357)

at py4j.Gateway.invoke(Gateway.java:282)

at py4j.commands.AbstractCommand.invokeMethod(AbstractCommand.java:132)

at py4j.commands.CallCommand.execute(CallCommand.java:79)

at py4j.ClientServerConnection.waitForCommands(ClientServerConnection.java:182)

at py4j.ClientServerConnection.run(ClientServerConnection.java:106)

at java.lang.Thread.run(Thread.java:750)

Caused by: java.lang.IllegalArgumentException: RejectRows does not exist. Available: IssuerId, IssuerId2, BusinessDate, StateCode, SourceName, NetworkName, NetworkURL, custnum, MarketCoverage, DentalOnlyPlan

at org.apache.spark.sql.types.StructType.$anonfun$fieldIndex$1(StructType.scala:306)

at scala.collection.immutable.HashMap$HashTrieMap.getOrElse0(HashMap.scala:596)

at scala.collection.immutable.HashMap.getOrElse(HashMap.scala:73)

at org.apache.spark.sql.types.StructType.fieldIndex(StructType.scala:305)

at org.apache.spark.sql.catalyst.OrderedFilters.$anonfun$predicates$4(OrderedFilters.scala:61)

at org.apache.spark.sql.catalyst.OrderedFilters.$anonfun$predicates$4$adapted(OrderedFilters.scala:61)

at scala.collection.TraversableLike.$anonfun$map$1(TraversableLike.scala:286)

at scala.collection.IndexedSeqOptimized.foreach(IndexedSeqOptimized.scala:36)

at scala.collection.IndexedSeqOptimized.foreach$(IndexedSeqOptimized.scala:33)

at scala.collection.mutable.ArrayOps$ofRef.foreach(ArrayOps.scala:198)

at scala.collection.TraversableLike.map(TraversableLike.scala:286)

at scala.collection.TraversableLike.map$(TraversableLike.scala:279)

at scala.collection.mutable.ArrayOps$ofRef.map(ArrayOps.scala:198)

at org.apache.spark.sql.catalyst.OrderedFilters.$anonfun$predicates$3(OrderedFilters.scala:61)

at org.apache.spark.sql.catalyst.OrderedFilters.$anonfun$predicates$3$adapted(OrderedFilters.scala:50)

at scala.collection.immutable.List.foreach(List.scala:431)

at org.apache.spark.sql.catalyst.OrderedFilters.<init>(OrderedFilters.scala:50)

at org.apache.spark.sql.catalyst.csv.UnivocityParser.<init>(UnivocityParser.scala:103)

at org.apache.spark.sql.execution.datasources.csv.CSVFileFormat.$anonfun$buildReader$1(CSVFileFormat.scala:130)

at org.apache.spark.sql.execution.datasources.FileFormat$$anon$1.apply(FileFormat.scala:148)

at org.apache.spark.sql.execution.datasources.FileFormat$$anon$1.apply(FileFormat.scala:133)

at org.apache.spark.sql.execution.datasources.FileScanRDD$$anon$1.org$apache$spark$sql$execution$datasources$FileScanRDD$$anon$$readCurrentFile(FileScanRDD.scala:127)

at org.apache.spark.sql.execution.datasources.FileScanRDD$$anon$1.nextIterator(FileScanRDD.scala:187)

at org.apache.spark.sql.execution.datasources.FileScanRDD$$anon$1.hasNext(FileScanRDD.scala:104)

at scala.collection.Iterator$$anon$10.hasNext(Iterator.scala:460)

at org.apache.spark.sql.catalyst.expressions.GeneratedClass$GeneratedIteratorForCodegenStage1.processNext(Unknown Source)

at org.apache.spark.sql.execution.BufferedRowIterator.hasNext(BufferedRowIterator.java:43)

at org.apache.spark.sql.execution.WholeStageCodegenExec$$anon$1.hasNext(WholeStageCodegenExec.scala:759)

at org.apache.spark.sql.execution.SparkPlan.$anonfun$getByteArrayRdd$1(SparkPlan.scala:349)

at org.apache.spark.rdd.RDD.$anonfun$mapPartitionsInternal$2(RDD.scala:898)

at org.apache.spark.rdd.RDD.$anonfun$mapPartitionsInternal$2$adapted(RDD.scala:898)

at org.apache.spark.rdd.MapPartitionsRDD.compute(MapPartitionsRDD.scala:52)

at org.apache.spark.rdd.RDD.computeOrReadCheckpoint(RDD.scala:373)

at org.apache.spark.rdd.RDD.iterator(RDD.scala:337)

at org.apache.spark.scheduler.ResultTask.runTask(ResultTask.scala:90)

at org.apache.spark.scheduler.Task.run(Task.scala:131)

at org.apache.spark.executor.Executor$TaskRunner.$anonfun$run$3(Executor.scala:506)

at org.apache.spark.util.Utils$.tryWithSafeFinally(Utils.scala:1462)

at org.apache.spark.executor.Executor$TaskRunner.run(Executor.scala:509)

at java.util.concurrent.ThreadPoolExecutor.runWorker(ThreadPoolExecutor.java:1149)

at java.util.concurrent.ThreadPoolExecutor$Worker.run(ThreadPoolExecutor.java:624)

... 1 more

>>> data\_frame\_1.where("RejectRows != null").show(3)

+--------+---------+------------+---------+----------+-----------+----------+-------+--------------+--------------+----------+

|IssuerId|IssuerId2|BusinessDate|StateCode|SourceName|NetworkName|NetworkURL|custnum|MarketCoverage|DentalOnlyPlan|RejectRows|

+--------+---------+------------+---------+----------+-----------+----------+-------+--------------+--------------+----------+

+--------+---------+------------+---------+----------+-----------+----------+-------+--------------+--------------+----------+

>>> data\_frame\_1.where("RejectRows== null").show(3)

+--------+---------+------------+---------+----------+-----------+----------+-------+--------------+--------------+----------+

|IssuerId|IssuerId2|BusinessDate|StateCode|SourceName|NetworkName|NetworkURL|custnum|MarketCoverage|DentalOnlyPlan|RejectRows|

+--------+---------+------------+---------+----------+-----------+----------+-------+--------------+--------------+----------+

+--------+---------+------------+---------+----------+-----------+----------+-------+--------------+--------------+----------+

>>> data\_frame\_1.where("RejectRows== "null"").show(3)

File "<stdin>", line 1

data\_frame\_1.where("RejectRows== "null"").show(3)

^

SyntaxError: invalid syntax

>>> data\_frame\_1.where("RejectRows== 'null;").show(3)

Traceback (most recent call last):

File "<stdin>", line 1, in <module>

File "/usr/local/spark/python/pyspark/sql/dataframe.py", line 1731, in filter

jdf = self.\_jdf.filter(condition)

File "/usr/local/spark/python/lib/py4j-0.10.9.3-src.zip/py4j/java\_gateway.py", line 1322, in \_\_call\_\_

File "/usr/local/spark/python/pyspark/sql/utils.py", line 117, in deco

raise converted from None

pyspark.sql.utils.ParseException:

extraneous input ''' expecting {'(', 'ADD', 'AFTER', 'ALL', 'ALTER', 'ANALYZE', 'AND', 'ANTI', 'ANY', 'ARCHIVE', 'ARRAY', 'AS', 'ASC', 'AT', 'AUTHORIZATION', 'BETWEEN', 'BOTH', 'BUCKET', 'BUCKETS', 'BY', 'CACHE', 'CASCADE', 'CASE', 'CAST', 'CHANGE', 'CHECK', 'CLEAR', 'CLUSTER', 'CLUSTERED', 'CODEGEN', 'COLLATE', 'COLLECTION', 'COLUMN', 'COLUMNS', 'COMMENT', 'COMMIT', 'COMPACT', 'COMPACTIONS', 'COMPUTE', 'CONCATENATE', 'CONSTRAINT', 'COST', 'CREATE', 'CROSS', 'CUBE', 'CURRENT', 'CURRENT\_DATE', 'CURRENT\_TIME', 'CURRENT\_TIMESTAMP', 'CURRENT\_USER', 'DAY', 'DATA', 'DATABASE', DATABASES, 'DBPROPERTIES', 'DEFINED', 'DELETE', 'DELIMITED', 'DESC', 'DESCRIBE', 'DFS', 'DIRECTORIES', 'DIRECTORY', 'DISTINCT', 'DISTRIBUTE', 'DIV', 'DROP', 'ELSE', 'END', 'ESCAPE', 'ESCAPED', 'EXCEPT', 'EXCHANGE', 'EXISTS', 'EXPLAIN', 'EXPORT', 'EXTENDED', 'EXTERNAL', 'EXTRACT', 'FALSE', 'FETCH', 'FIELDS', 'FILTER', 'FILEFORMAT', 'FIRST', 'FOLLOWING', 'FOR', 'FOREIGN', 'FORMAT', 'FORMATTED', 'FROM', 'FULL', 'FUNCTION', 'FUNCTIONS', 'GLOBAL', 'GRANT', 'GROUP', 'GROUPING', 'HAVING', 'HOUR', 'IF', 'IGNORE', 'IMPORT', 'IN', 'INDEX', 'INDEXES', 'INNER', 'INPATH', 'INPUTFORMAT', 'INSERT', 'INTERSECT', 'INTERVAL', 'INTO', 'IS', 'ITEMS', 'JOIN', 'KEYS', 'LAST', 'LATERAL', 'LAZY', 'LEADING', 'LEFT', 'LIKE', 'LIMIT', 'LINES', 'LIST', 'LOAD', 'LOCAL', 'LOCATION', 'LOCK', 'LOCKS', 'LOGICAL', 'MACRO', 'MAP', 'MATCHED', 'MERGE', 'MINUTE', 'MONTH', 'MSCK', 'NAMESPACE', 'NAMESPACES', 'NATURAL', 'NO', NOT, 'NULL', 'NULLS', 'OF', 'ON', 'ONLY', 'OPTION', 'OPTIONS', 'OR', 'ORDER', 'OUT', 'OUTER', 'OUTPUTFORMAT', 'OVER', 'OVERLAPS', 'OVERLAY', 'OVERWRITE', 'PARTITION', 'PARTITIONED', 'PARTITIONS', 'PERCENT', 'PIVOT', 'PLACING', 'POSITION', 'PRECEDING', 'PRIMARY', 'PRINCIPALS', 'PROPERTIES', 'PURGE', 'QUERY', 'RANGE', 'RECORDREADER', 'RECORDWRITER', 'RECOVER', 'REDUCE', 'REFERENCES', 'REFRESH', 'RENAME', 'REPAIR', 'REPLACE', 'RESET', 'RESPECT', 'RESTRICT', 'REVOKE', 'RIGHT', RLIKE, 'ROLE', 'ROLES', 'ROLLBACK', 'ROLLUP', 'ROW', 'ROWS', 'SECOND', 'SCHEMA', 'SELECT', 'SEMI', 'SEPARATED', 'SERDE', 'SERDEPROPERTIES', 'SESSION\_USER', 'SET', 'MINUS', 'SETS', 'SHOW', 'SKEWED', 'SOME', 'SORT', 'SORTED', 'START', 'STATISTICS', 'STORED', 'STRATIFY', 'STRUCT', 'SUBSTR', 'SUBSTRING', 'SYNC', 'TABLE', 'TABLES', 'TABLESAMPLE', 'TBLPROPERTIES', TEMPORARY, 'TERMINATED', 'THEN', 'TIME', 'TO', 'TOUCH', 'TRAILING', 'TRANSACTION', 'TRANSACTIONS', 'TRANSFORM', 'TRIM', 'TRUE', 'TRUNCATE', 'TRY\_CAST', 'TYPE', 'UNARCHIVE', 'UNBOUNDED', 'UNCACHE', 'UNION', 'UNIQUE', 'UNKNOWN', 'UNLOCK', 'UNSET', 'UPDATE', 'USE', 'USER', 'USING', 'VALUES', 'VIEW', 'VIEWS', 'WHEN', 'WHERE', 'WINDOW', 'WITH', 'YEAR', 'ZONE', '+', '-', '\*', '~', STRING, BIGINT\_LITERAL, SMALLINT\_LITERAL, TINYINT\_LITERAL, INTEGER\_VALUE, EXPONENT\_VALUE, DECIMAL\_VALUE, FLOAT\_LITERAL, DOUBLE\_LITERAL, BIGDECIMAL\_LITERAL, IDENTIFIER, BACKQUOTED\_IDENTIFIER}(line 1, pos 13)

== SQL ==

RejectRows== 'null;

-------------^^^

>>>

>>> data\_frame\_1.show(3)

+--------+---------+------------+---------+----------+------------+--------------------+-------+--------------+--------------+--------------------+

|IssuerId|IssuerId2|BusinessDate|StateCode|SourceName| NetworkName| NetworkURL|custnum|MarketCoverage|DentalOnlyPlan| RejectRows|

+--------+---------+------------+---------+----------+------------+--------------------+-------+--------------+--------------+--------------------+

| 21989| 21989| 2019-10-01| AK| HIOS| ODS Premier|https://www.modah...| 13| null| null| null|

| 219| 219| 2019-10-01| AK| HIOS| ODS Premier| null| null| null| null|219,219,2019-10-0...|

| 38344| 38344| 2019-10-01| AK| HIOS|HeritagePlus|https://www.preme...| 13| null| null| null|

+--------+---------+------------+---------+----------+------------+--------------------+-------+--------------+--------------+--------------------+

only showing top 3 rows

>>> data\_frame\_2.show(3)

+--------+---------+------------+---------+----------+--------------------+--------------------+-------+--------------+--------------+

|IssuerId|IssuerId2|BusinessDate|StateCode|SourceName| NetworkName| NetworkURL|custnum|MarketCoverage|DentalOnlyPlan|

+--------+---------+------------+---------+----------+--------------------+--------------------+-------+--------------+--------------+

| 31195| 31195| 2019-01-10| SD| SERFF|Sanford Health Pl...|https://www.viiad...| 14| null| null|

| 60536| 60536| 2019-01-10| SD| SERFF| Avera Health Plans|www.Avera.org/Pro...| 13| null| null|

| 61214| 61214| 2019-01-10| SD| SERFF| DenteMax|http://www2.dente...| 14| null| null|

+--------+---------+------------+---------+----------+--------------------+--------------------+-------+--------------+--------------+

only showing top 3 rows

>>> data\_frame\_3.show(3)

+--------+---------+------------+---------+----------+--------------------+--------------------+-------+--------------+--------------+----------+

|IssuerId|IssuerId2|BusinessDate|StateCode|SourceName| NetworkName| NetworkURL|custnum|MarketCoverage|DentalOnlyPlan|RejectRows|

+--------+---------+------------+---------+----------+--------------------+--------------------+-------+--------------+--------------+----------+

| 31195| 31195| 2019-01-10| SD| SERFF|Sanford Health Pl...|https://www.viiad...| 14| null| null| null|

| 60536| 60536| 2019-01-10| SD| SERFF| Avera Health Plans|www.Avera.org/Pro...| 13| null| null| null|

| 61214| 61214| 2019-01-10| SD| SERFF| DenteMax|http://www2.dente...| 14| null| null| null|

+--------+---------+------------+---------+----------+--------------------+--------------------+-------+--------------+--------------+----------+

only showing top 3 rows

>>> data\_frame\_1.filter(data\_frame\_1.RejectRows.isNotNull()).show(3)

24/07/18 20:46:20 ERROR executor.Executor: Exception in task 0.0 in stage 23.0 (TID 23)

java.lang.IllegalArgumentException: RejectRows does not exist. Available: IssuerId, IssuerId2, BusinessDate, StateCode, SourceName, NetworkName, NetworkURL, custnum, MarketCoverage, DentalOnlyPlan

at org.apache.spark.sql.types.StructType.$anonfun$fieldIndex$1(StructType.scala:306)

at scala.collection.immutable.HashMap$HashTrieMap.getOrElse0(HashMap.scala:596)

at scala.collection.immutable.HashMap.getOrElse(HashMap.scala:73)

at org.apache.spark.sql.types.StructType.fieldIndex(StructType.scala:305)

at org.apache.spark.sql.catalyst.OrderedFilters.$anonfun$predicates$4(OrderedFilters.scala:61)

at org.apache.spark.sql.catalyst.OrderedFilters.$anonfun$predicates$4$adapted(OrderedFilters.scala:61)

at scala.collection.TraversableLike.$anonfun$map$1(TraversableLike.scala:286)

at scala.collection.IndexedSeqOptimized.foreach(IndexedSeqOptimized.scala:36)

at scala.collection.IndexedSeqOptimized.foreach$(IndexedSeqOptimized.scala:33)

at scala.collection.mutable.ArrayOps$ofRef.foreach(ArrayOps.scala:198)

at scala.collection.TraversableLike.map(TraversableLike.scala:286)

at scala.collection.TraversableLike.map$(TraversableLike.scala:279)

at scala.collection.mutable.ArrayOps$ofRef.map(ArrayOps.scala:198)

at org.apache.spark.sql.catalyst.OrderedFilters.$anonfun$predicates$3(OrderedFilters.scala:61)

at org.apache.spark.sql.catalyst.OrderedFilters.$anonfun$predicates$3$adapted(OrderedFilters.scala:50)

at scala.collection.immutable.List.foreach(List.scala:431)

at org.apache.spark.sql.catalyst.OrderedFilters.<init>(OrderedFilters.scala:50)

at org.apache.spark.sql.catalyst.csv.UnivocityParser.<init>(UnivocityParser.scala:103)

at org.apache.spark.sql.execution.datasources.csv.CSVFileFormat.$anonfun$buildReader$1(CSVFileFormat.scala:130)

at org.apache.spark.sql.execution.datasources.FileFormat$$anon$1.apply(FileFormat.scala:148)

at org.apache.spark.sql.execution.datasources.FileFormat$$anon$1.apply(FileFormat.scala:133)

at org.apache.spark.sql.execution.datasources.FileScanRDD$$anon$1.org$apache$spark$sql$execution$datasources$FileScanRDD$$anon$$readCurrentFile(FileScanRDD.scala:127)

at org.apache.spark.sql.execution.datasources.FileScanRDD$$anon$1.nextIterator(FileScanRDD.scala:187)

at org.apache.spark.sql.execution.datasources.FileScanRDD$$anon$1.hasNext(FileScanRDD.scala:104)

at scala.collection.Iterator$$anon$10.hasNext(Iterator.scala:460)

at org.apache.spark.sql.catalyst.expressions.GeneratedClass$GeneratedIteratorForCodegenStage1.processNext(Unknown Source)

at org.apache.spark.sql.execution.BufferedRowIterator.hasNext(BufferedRowIterator.java:43)

at org.apache.spark.sql.execution.WholeStageCodegenExec$$anon$1.hasNext(WholeStageCodegenExec.scala:759)

at org.apache.spark.sql.execution.SparkPlan.$anonfun$getByteArrayRdd$1(SparkPlan.scala:349)

at org.apache.spark.rdd.RDD.$anonfun$mapPartitionsInternal$2(RDD.scala:898)

at org.apache.spark.rdd.RDD.$anonfun$mapPartitionsInternal$2$adapted(RDD.scala:898)

at org.apache.spark.rdd.MapPartitionsRDD.compute(MapPartitionsRDD.scala:52)

at org.apache.spark.rdd.RDD.computeOrReadCheckpoint(RDD.scala:373)

at org.apache.spark.rdd.RDD.iterator(RDD.scala:337)

at org.apache.spark.scheduler.ResultTask.runTask(ResultTask.scala:90)

at org.apache.spark.scheduler.Task.run(Task.scala:131)

at org.apache.spark.executor.Executor$TaskRunner.$anonfun$run$3(Executor.scala:506)

at org.apache.spark.util.Utils$.tryWithSafeFinally(Utils.scala:1462)

at org.apache.spark.executor.Executor$TaskRunner.run(Executor.scala:509)

at java.util.concurrent.ThreadPoolExecutor.runWorker(ThreadPoolExecutor.java:1149)

at java.util.concurrent.ThreadPoolExecutor$Worker.run(ThreadPoolExecutor.java:624)

at java.lang.Thread.run(Thread.java:750)

24/07/18 20:46:20 WARN scheduler.TaskSetManager: Lost task 0.0 in stage 23.0 (TID 23) (192.168.112.129 executor driver): java.lang.IllegalArgumentException: RejectRows does not exist. Available: IssuerId, IssuerId2, BusinessDate, StateCode, SourceName, NetworkName, NetworkURL, custnum, MarketCoverage, DentalOnlyPlan

at org.apache.spark.sql.types.StructType.$anonfun$fieldIndex$1(StructType.scala:306)

at scala.collection.immutable.HashMap$HashTrieMap.getOrElse0(HashMap.scala:596)

at scala.collection.immutable.HashMap.getOrElse(HashMap.scala:73)

at org.apache.spark.sql.types.StructType.fieldIndex(StructType.scala:305)

at org.apache.spark.sql.catalyst.OrderedFilters.$anonfun$predicates$4(OrderedFilters.scala:61)

at org.apache.spark.sql.catalyst.OrderedFilters.$anonfun$predicates$4$adapted(OrderedFilters.scala:61)

at scala.collection.TraversableLike.$anonfun$map$1(TraversableLike.scala:286)

at scala.collection.IndexedSeqOptimized.foreach(IndexedSeqOptimized.scala:36)

at scala.collection.IndexedSeqOptimized.foreach$(IndexedSeqOptimized.scala:33)

at scala.collection.mutable.ArrayOps$ofRef.foreach(ArrayOps.scala:198)

at scala.collection.TraversableLike.map(TraversableLike.scala:286)

at scala.collection.TraversableLike.map$(TraversableLike.scala:279)

at scala.collection.mutable.ArrayOps$ofRef.map(ArrayOps.scala:198)

at org.apache.spark.sql.catalyst.OrderedFilters.$anonfun$predicates$3(OrderedFilters.scala:61)

at org.apache.spark.sql.catalyst.OrderedFilters.$anonfun$predicates$3$adapted(OrderedFilters.scala:50)

at scala.collection.immutable.List.foreach(List.scala:431)

at org.apache.spark.sql.catalyst.OrderedFilters.<init>(OrderedFilters.scala:50)

at org.apache.spark.sql.catalyst.csv.UnivocityParser.<init>(UnivocityParser.scala:103)

at org.apache.spark.sql.execution.datasources.csv.CSVFileFormat.$anonfun$buildReader$1(CSVFileFormat.scala:130)

at org.apache.spark.sql.execution.datasources.FileFormat$$anon$1.apply(FileFormat.scala:148)

at org.apache.spark.sql.execution.datasources.FileFormat$$anon$1.apply(FileFormat.scala:133)

at org.apache.spark.sql.execution.datasources.FileScanRDD$$anon$1.org$apache$spark$sql$execution$datasources$FileScanRDD$$anon$$readCurrentFile(FileScanRDD.scala:127)

at org.apache.spark.sql.execution.datasources.FileScanRDD$$anon$1.nextIterator(FileScanRDD.scala:187)

at org.apache.spark.sql.execution.datasources.FileScanRDD$$anon$1.hasNext(FileScanRDD.scala:104)

at scala.collection.Iterator$$anon$10.hasNext(Iterator.scala:460)

at org.apache.spark.sql.catalyst.expressions.GeneratedClass$GeneratedIteratorForCodegenStage1.processNext(Unknown Source)

at org.apache.spark.sql.execution.BufferedRowIterator.hasNext(BufferedRowIterator.java:43)

at org.apache.spark.sql.execution.WholeStageCodegenExec$$anon$1.hasNext(WholeStageCodegenExec.scala:759)

at org.apache.spark.sql.execution.SparkPlan.$anonfun$getByteArrayRdd$1(SparkPlan.scala:349)

at org.apache.spark.rdd.RDD.$anonfun$mapPartitionsInternal$2(RDD.scala:898)

at org.apache.spark.rdd.RDD.$anonfun$mapPartitionsInternal$2$adapted(RDD.scala:898)

at org.apache.spark.rdd.MapPartitionsRDD.compute(MapPartitionsRDD.scala:52)

at org.apache.spark.rdd.RDD.computeOrReadCheckpoint(RDD.scala:373)

at org.apache.spark.rdd.RDD.iterator(RDD.scala:337)

at org.apache.spark.scheduler.ResultTask.runTask(ResultTask.scala:90)

at org.apache.spark.scheduler.Task.run(Task.scala:131)

at org.apache.spark.executor.Executor$TaskRunner.$anonfun$run$3(Executor.scala:506)

at org.apache.spark.util.Utils$.tryWithSafeFinally(Utils.scala:1462)

at org.apache.spark.executor.Executor$TaskRunner.run(Executor.scala:509)

at java.util.concurrent.ThreadPoolExecutor.runWorker(ThreadPoolExecutor.java:1149)

at java.util.concurrent.ThreadPoolExecutor$Worker.run(ThreadPoolExecutor.java:624)

at java.lang.Thread.run(Thread.java:750)

24/07/18 20:46:20 ERROR scheduler.TaskSetManager: Task 0 in stage 23.0 failed 1 times; aborting job

Traceback (most recent call last):

File "<stdin>", line 1, in <module>

File "/usr/local/spark/python/pyspark/sql/dataframe.py", line 494, in show

print(self.\_jdf.showString(n, 20, vertical))

File "/usr/local/spark/python/lib/py4j-0.10.9.3-src.zip/py4j/java\_gateway.py", line 1322, in \_\_call\_\_

File "/usr/local/spark/python/pyspark/sql/utils.py", line 111, in deco

return f(\*a, \*\*kw)

File "/usr/local/spark/python/lib/py4j-0.10.9.3-src.zip/py4j/protocol.py", line 328, in get\_return\_value

py4j.protocol.Py4JJavaError: An error occurred while calling o264.showString.

: org.apache.spark.SparkException: Job aborted due to stage failure: Task 0 in stage 23.0 failed 1 times, most recent failure: Lost task 0.0 in stage 23.0 (TID 23) (192.168.112.129 executor driver): java.lang.IllegalArgumentException: RejectRows does not exist. Available: IssuerId, IssuerId2, BusinessDate, StateCode, SourceName, NetworkName, NetworkURL, custnum, MarketCoverage, DentalOnlyPlan

at org.apache.spark.sql.types.StructType.$anonfun$fieldIndex$1(StructType.scala:306)

at scala.collection.immutable.HashMap$HashTrieMap.getOrElse0(HashMap.scala:596)

at scala.collection.immutable.HashMap.getOrElse(HashMap.scala:73)

at org.apache.spark.sql.types.StructType.fieldIndex(StructType.scala:305)

at org.apache.spark.sql.catalyst.OrderedFilters.$anonfun$predicates$4(OrderedFilters.scala:61)

at org.apache.spark.sql.catalyst.OrderedFilters.$anonfun$predicates$4$adapted(OrderedFilters.scala:61)

at scala.collection.TraversableLike.$anonfun$map$1(TraversableLike.scala:286)

at scala.collection.IndexedSeqOptimized.foreach(IndexedSeqOptimized.scala:36)

at scala.collection.IndexedSeqOptimized.foreach$(IndexedSeqOptimized.scala:33)

at scala.collection.mutable.ArrayOps$ofRef.foreach(ArrayOps.scala:198)

at scala.collection.TraversableLike.map(TraversableLike.scala:286)

at scala.collection.TraversableLike.map$(TraversableLike.scala:279)

at scala.collection.mutable.ArrayOps$ofRef.map(ArrayOps.scala:198)

at org.apache.spark.sql.catalyst.OrderedFilters.$anonfun$predicates$3(OrderedFilters.scala:61)

at org.apache.spark.sql.catalyst.OrderedFilters.$anonfun$predicates$3$adapted(OrderedFilters.scala:50)

at scala.collection.immutable.List.foreach(List.scala:431)

at org.apache.spark.sql.catalyst.OrderedFilters.<init>(OrderedFilters.scala:50)

at org.apache.spark.sql.catalyst.csv.UnivocityParser.<init>(UnivocityParser.scala:103)

at org.apache.spark.sql.execution.datasources.csv.CSVFileFormat.$anonfun$buildReader$1(CSVFileFormat.scala:130)

at org.apache.spark.sql.execution.datasources.FileFormat$$anon$1.apply(FileFormat.scala:148)

at org.apache.spark.sql.execution.datasources.FileFormat$$anon$1.apply(FileFormat.scala:133)

at org.apache.spark.sql.execution.datasources.FileScanRDD$$anon$1.org$apache$spark$sql$execution$datasources$FileScanRDD$$anon$$readCurrentFile(FileScanRDD.scala:127)

at org.apache.spark.sql.execution.datasources.FileScanRDD$$anon$1.nextIterator(FileScanRDD.scala:187)

at org.apache.spark.sql.execution.datasources.FileScanRDD$$anon$1.hasNext(FileScanRDD.scala:104)

at scala.collection.Iterator$$anon$10.hasNext(Iterator.scala:460)

at org.apache.spark.sql.catalyst.expressions.GeneratedClass$GeneratedIteratorForCodegenStage1.processNext(Unknown Source)

at org.apache.spark.sql.execution.BufferedRowIterator.hasNext(BufferedRowIterator.java:43)

at org.apache.spark.sql.execution.WholeStageCodegenExec$$anon$1.hasNext(WholeStageCodegenExec.scala:759)

at org.apache.spark.sql.execution.SparkPlan.$anonfun$getByteArrayRdd$1(SparkPlan.scala:349)

at org.apache.spark.rdd.RDD.$anonfun$mapPartitionsInternal$2(RDD.scala:898)

at org.apache.spark.rdd.RDD.$anonfun$mapPartitionsInternal$2$adapted(RDD.scala:898)

at org.apache.spark.rdd.MapPartitionsRDD.compute(MapPartitionsRDD.scala:52)

at org.apache.spark.rdd.RDD.computeOrReadCheckpoint(RDD.scala:373)

at org.apache.spark.rdd.RDD.iterator(RDD.scala:337)

at org.apache.spark.scheduler.ResultTask.runTask(ResultTask.scala:90)

at org.apache.spark.scheduler.Task.run(Task.scala:131)

at org.apache.spark.executor.Executor$TaskRunner.$anonfun$run$3(Executor.scala:506)

at org.apache.spark.util.Utils$.tryWithSafeFinally(Utils.scala:1462)

at org.apache.spark.executor.Executor$TaskRunner.run(Executor.scala:509)

at java.util.concurrent.ThreadPoolExecutor.runWorker(ThreadPoolExecutor.java:1149)

at java.util.concurrent.ThreadPoolExecutor$Worker.run(ThreadPoolExecutor.java:624)

at java.lang.Thread.run(Thread.java:750)

Driver stacktrace:

at org.apache.spark.scheduler.DAGScheduler.failJobAndIndependentStages(DAGScheduler.scala:2454)

at org.apache.spark.scheduler.DAGScheduler.$anonfun$abortStage$2(DAGScheduler.scala:2403)

at org.apache.spark.scheduler.DAGScheduler.$anonfun$abortStage$2$adapted(DAGScheduler.scala:2402)

at scala.collection.mutable.ResizableArray.foreach(ResizableArray.scala:62)

at scala.collection.mutable.ResizableArray.foreach$(ResizableArray.scala:55)

at scala.collection.mutable.ArrayBuffer.foreach(ArrayBuffer.scala:49)

at org.apache.spark.scheduler.DAGScheduler.abortStage(DAGScheduler.scala:2402)

at org.apache.spark.scheduler.DAGScheduler.$anonfun$handleTaskSetFailed$1(DAGScheduler.scala:1160)

at org.apache.spark.scheduler.DAGScheduler.$anonfun$handleTaskSetFailed$1$adapted(DAGScheduler.scala:1160)

at scala.Option.foreach(Option.scala:407)

at org.apache.spark.scheduler.DAGScheduler.handleTaskSetFailed(DAGScheduler.scala:1160)

at org.apache.spark.scheduler.DAGSchedulerEventProcessLoop.doOnReceive(DAGScheduler.scala:2642)

at org.apache.spark.scheduler.DAGSchedulerEventProcessLoop.onReceive(DAGScheduler.scala:2584)

at org.apache.spark.scheduler.DAGSchedulerEventProcessLoop.onReceive(DAGScheduler.scala:2573)

at org.apache.spark.util.EventLoop$$anon$1.run(EventLoop.scala:49)

at org.apache.spark.scheduler.DAGScheduler.runJob(DAGScheduler.scala:938)

at org.apache.spark.SparkContext.runJob(SparkContext.scala:2214)

at org.apache.spark.SparkContext.runJob(SparkContext.scala:2235)

at org.apache.spark.SparkContext.runJob(SparkContext.scala:2254)

at org.apache.spark.sql.execution.SparkPlan.executeTake(SparkPlan.scala:476)

at org.apache.spark.sql.execution.SparkPlan.executeTake(SparkPlan.scala:429)

at org.apache.spark.sql.execution.CollectLimitExec.executeCollect(limit.scala:48)

at org.apache.spark.sql.Dataset.collectFromPlan(Dataset.scala:3715)

at org.apache.spark.sql.Dataset.$anonfun$head$1(Dataset.scala:2728)

at org.apache.spark.sql.Dataset.$anonfun$withAction$1(Dataset.scala:3706)

at org.apache.spark.sql.execution.SQLExecution$.$anonfun$withNewExecutionId$5(SQLExecution.scala:103)

at org.apache.spark.sql.execution.SQLExecution$.withSQLConfPropagated(SQLExecution.scala:163)

at org.apache.spark.sql.execution.SQLExecution$.$anonfun$withNewExecutionId$1(SQLExecution.scala:90)

at org.apache.spark.sql.SparkSession.withActive(SparkSession.scala:775)

at org.apache.spark.sql.execution.SQLExecution$.withNewExecutionId(SQLExecution.scala:64)

at org.apache.spark.sql.Dataset.withAction(Dataset.scala:3704)

at org.apache.spark.sql.Dataset.head(Dataset.scala:2728)

at org.apache.spark.sql.Dataset.take(Dataset.scala:2935)

at org.apache.spark.sql.Dataset.getRows(Dataset.scala:287)

at org.apache.spark.sql.Dataset.showString(Dataset.scala:326)

at sun.reflect.GeneratedMethodAccessor41.invoke(Unknown Source)

at sun.reflect.DelegatingMethodAccessorImpl.invoke(DelegatingMethodAccessorImpl.java:43)

at java.lang.reflect.Method.invoke(Method.java:498)

at py4j.reflection.MethodInvoker.invoke(MethodInvoker.java:244)

at py4j.reflection.ReflectionEngine.invoke(ReflectionEngine.java:357)

at py4j.Gateway.invoke(Gateway.java:282)

at py4j.commands.AbstractCommand.invokeMethod(AbstractCommand.java:132)

at py4j.commands.CallCommand.execute(CallCommand.java:79)

at py4j.ClientServerConnection.waitForCommands(ClientServerConnection.java:182)

at py4j.ClientServerConnection.run(ClientServerConnection.java:106)

at java.lang.Thread.run(Thread.java:750)

Caused by: java.lang.IllegalArgumentException: RejectRows does not exist. Available: IssuerId, IssuerId2, BusinessDate, StateCode, SourceName, NetworkName, NetworkURL, custnum, MarketCoverage, DentalOnlyPlan

at org.apache.spark.sql.types.StructType.$anonfun$fieldIndex$1(StructType.scala:306)

at scala.collection.immutable.HashMap$HashTrieMap.getOrElse0(HashMap.scala:596)

at scala.collection.immutable.HashMap.getOrElse(HashMap.scala:73)

at org.apache.spark.sql.types.StructType.fieldIndex(StructType.scala:305)

at org.apache.spark.sql.catalyst.OrderedFilters.$anonfun$predicates$4(OrderedFilters.scala:61)

at org.apache.spark.sql.catalyst.OrderedFilters.$anonfun$predicates$4$adapted(OrderedFilters.scala:61)

at scala.collection.TraversableLike.$anonfun$map$1(TraversableLike.scala:286)

at scala.collection.IndexedSeqOptimized.foreach(IndexedSeqOptimized.scala:36)

at scala.collection.IndexedSeqOptimized.foreach$(IndexedSeqOptimized.scala:33)

at scala.collection.mutable.ArrayOps$ofRef.foreach(ArrayOps.scala:198)

at scala.collection.TraversableLike.map(TraversableLike.scala:286)

at scala.collection.TraversableLike.map$(TraversableLike.scala:279)

at scala.collection.mutable.ArrayOps$ofRef.map(ArrayOps.scala:198)

at org.apache.spark.sql.catalyst.OrderedFilters.$anonfun$predicates$3(OrderedFilters.scala:61)

at org.apache.spark.sql.catalyst.OrderedFilters.$anonfun$predicates$3$adapted(OrderedFilters.scala:50)

at scala.collection.immutable.List.foreach(List.scala:431)

at org.apache.spark.sql.catalyst.OrderedFilters.<init>(OrderedFilters.scala:50)

at org.apache.spark.sql.catalyst.csv.UnivocityParser.<init>(UnivocityParser.scala:103)

at org.apache.spark.sql.execution.datasources.csv.CSVFileFormat.$anonfun$buildReader$1(CSVFileFormat.scala:130)

at org.apache.spark.sql.execution.datasources.FileFormat$$anon$1.apply(FileFormat.scala:148)

at org.apache.spark.sql.execution.datasources.FileFormat$$anon$1.apply(FileFormat.scala:133)

at org.apache.spark.sql.execution.datasources.FileScanRDD$$anon$1.org$apache$spark$sql$execution$datasources$FileScanRDD$$anon$$readCurrentFile(FileScanRDD.scala:127)

at org.apache.spark.sql.execution.datasources.FileScanRDD$$anon$1.nextIterator(FileScanRDD.scala:187)

at org.apache.spark.sql.execution.datasources.FileScanRDD$$anon$1.hasNext(FileScanRDD.scala:104)

at scala.collection.Iterator$$anon$10.hasNext(Iterator.scala:460)

at org.apache.spark.sql.catalyst.expressions.GeneratedClass$GeneratedIteratorForCodegenStage1.processNext(Unknown Source)

at org.apache.spark.sql.execution.BufferedRowIterator.hasNext(BufferedRowIterator.java:43)

at org.apache.spark.sql.execution.WholeStageCodegenExec$$anon$1.hasNext(WholeStageCodegenExec.scala:759)

at org.apache.spark.sql.execution.SparkPlan.$anonfun$getByteArrayRdd$1(SparkPlan.scala:349)

at org.apache.spark.rdd.RDD.$anonfun$mapPartitionsInternal$2(RDD.scala:898)

at org.apache.spark.rdd.RDD.$anonfun$mapPartitionsInternal$2$adapted(RDD.scala:898)

at org.apache.spark.rdd.MapPartitionsRDD.compute(MapPartitionsRDD.scala:52)

at org.apache.spark.rdd.RDD.computeOrReadCheckpoint(RDD.scala:373)

at org.apache.spark.rdd.RDD.iterator(RDD.scala:337)

at org.apache.spark.scheduler.ResultTask.runTask(ResultTask.scala:90)

at org.apache.spark.scheduler.Task.run(Task.scala:131)

at org.apache.spark.executor.Executor$TaskRunner.$anonfun$run$3(Executor.scala:506)

at org.apache.spark.util.Utils$.tryWithSafeFinally(Utils.scala:1462)

at org.apache.spark.executor.Executor$TaskRunner.run(Executor.scala:509)

at java.util.concurrent.ThreadPoolExecutor.runWorker(ThreadPoolExecutor.java:1149)

at java.util.concurrent.ThreadPoolExecutor$Worker.run(ThreadPoolExecutor.java:624)

... 1 more

>>> data\_frame\_1.where(data\_frame\_1.RejectRows.isNotNull()).show(3)

24/07/18 20:46:47 ERROR executor.Executor: Exception in task 0.0 in stage 24.0 (TID 24)

java.lang.IllegalArgumentException: RejectRows does not exist. Available: IssuerId, IssuerId2, BusinessDate, StateCode, SourceName, NetworkName, NetworkURL, custnum, MarketCoverage, DentalOnlyPlan

at org.apache.spark.sql.types.StructType.$anonfun$fieldIndex$1(StructType.scala:306)

at scala.collection.immutable.HashMap$HashTrieMap.getOrElse0(HashMap.scala:596)

at scala.collection.immutable.HashMap.getOrElse(HashMap.scala:73)

at org.apache.spark.sql.types.StructType.fieldIndex(StructType.scala:305)

at org.apache.spark.sql.catalyst.OrderedFilters.$anonfun$predicates$4(OrderedFilters.scala:61)

at org.apache.spark.sql.catalyst.OrderedFilters.$anonfun$predicates$4$adapted(OrderedFilters.scala:61)

at scala.collection.TraversableLike.$anonfun$map$1(TraversableLike.scala:286)

at scala.collection.IndexedSeqOptimized.foreach(IndexedSeqOptimized.scala:36)

at scala.collection.IndexedSeqOptimized.foreach$(IndexedSeqOptimized.scala:33)

at scala.collection.mutable.ArrayOps$ofRef.foreach(ArrayOps.scala:198)

at scala.collection.TraversableLike.map(TraversableLike.scala:286)

at scala.collection.TraversableLike.map$(TraversableLike.scala:279)

at scala.collection.mutable.ArrayOps$ofRef.map(ArrayOps.scala:198)

at org.apache.spark.sql.catalyst.OrderedFilters.$anonfun$predicates$3(OrderedFilters.scala:61)

at org.apache.spark.sql.catalyst.OrderedFilters.$anonfun$predicates$3$adapted(OrderedFilters.scala:50)

at scala.collection.immutable.List.foreach(List.scala:431)

at org.apache.spark.sql.catalyst.OrderedFilters.<init>(OrderedFilters.scala:50)

at org.apache.spark.sql.catalyst.csv.UnivocityParser.<init>(UnivocityParser.scala:103)

at org.apache.spark.sql.execution.datasources.csv.CSVFileFormat.$anonfun$buildReader$1(CSVFileFormat.scala:130)

at org.apache.spark.sql.execution.datasources.FileFormat$$anon$1.apply(FileFormat.scala:148)

at org.apache.spark.sql.execution.datasources.FileFormat$$anon$1.apply(FileFormat.scala:133)

at org.apache.spark.sql.execution.datasources.FileScanRDD$$anon$1.org$apache$spark$sql$execution$datasources$FileScanRDD$$anon$$readCurrentFile(FileScanRDD.scala:127)

at org.apache.spark.sql.execution.datasources.FileScanRDD$$anon$1.nextIterator(FileScanRDD.scala:187)

at org.apache.spark.sql.execution.datasources.FileScanRDD$$anon$1.hasNext(FileScanRDD.scala:104)

at scala.collection.Iterator$$anon$10.hasNext(Iterator.scala:460)

at org.apache.spark.sql.catalyst.expressions.GeneratedClass$GeneratedIteratorForCodegenStage1.processNext(Unknown Source)

at org.apache.spark.sql.execution.BufferedRowIterator.hasNext(BufferedRowIterator.java:43)

at org.apache.spark.sql.execution.WholeStageCodegenExec$$anon$1.hasNext(WholeStageCodegenExec.scala:759)

at org.apache.spark.sql.execution.SparkPlan.$anonfun$getByteArrayRdd$1(SparkPlan.scala:349)

at org.apache.spark.rdd.RDD.$anonfun$mapPartitionsInternal$2(RDD.scala:898)

at org.apache.spark.rdd.RDD.$anonfun$mapPartitionsInternal$2$adapted(RDD.scala:898)

at org.apache.spark.rdd.MapPartitionsRDD.compute(MapPartitionsRDD.scala:52)

at org.apache.spark.rdd.RDD.computeOrReadCheckpoint(RDD.scala:373)

at org.apache.spark.rdd.RDD.iterator(RDD.scala:337)

at org.apache.spark.scheduler.ResultTask.runTask(ResultTask.scala:90)

at org.apache.spark.scheduler.Task.run(Task.scala:131)

at org.apache.spark.executor.Executor$TaskRunner.$anonfun$run$3(Executor.scala:506)

at org.apache.spark.util.Utils$.tryWithSafeFinally(Utils.scala:1462)

at org.apache.spark.executor.Executor$TaskRunner.run(Executor.scala:509)

at java.util.concurrent.ThreadPoolExecutor.runWorker(ThreadPoolExecutor.java:1149)

at java.util.concurrent.ThreadPoolExecutor$Worker.run(ThreadPoolExecutor.java:624)

at java.lang.Thread.run(Thread.java:750)

24/07/18 20:46:47 WARN scheduler.TaskSetManager: Lost task 0.0 in stage 24.0 (TID 24) (192.168.112.129 executor driver): java.lang.IllegalArgumentException: RejectRows does not exist. Available: IssuerId, IssuerId2, BusinessDate, StateCode, SourceName, NetworkName, NetworkURL, custnum, MarketCoverage, DentalOnlyPlan

at org.apache.spark.sql.types.StructType.$anonfun$fieldIndex$1(StructType.scala:306)

at scala.collection.immutable.HashMap$HashTrieMap.getOrElse0(HashMap.scala:596)

at scala.collection.immutable.HashMap.getOrElse(HashMap.scala:73)

at org.apache.spark.sql.types.StructType.fieldIndex(StructType.scala:305)

at org.apache.spark.sql.catalyst.OrderedFilters.$anonfun$predicates$4(OrderedFilters.scala:61)

at org.apache.spark.sql.catalyst.OrderedFilters.$anonfun$predicates$4$adapted(OrderedFilters.scala:61)

at scala.collection.TraversableLike.$anonfun$map$1(TraversableLike.scala:286)

at scala.collection.IndexedSeqOptimized.foreach(IndexedSeqOptimized.scala:36)

at scala.collection.IndexedSeqOptimized.foreach$(IndexedSeqOptimized.scala:33)

at scala.collection.mutable.ArrayOps$ofRef.foreach(ArrayOps.scala:198)

at scala.collection.TraversableLike.map(TraversableLike.scala:286)

at scala.collection.TraversableLike.map$(TraversableLike.scala:279)

at scala.collection.mutable.ArrayOps$ofRef.map(ArrayOps.scala:198)

at org.apache.spark.sql.catalyst.OrderedFilters.$anonfun$predicates$3(OrderedFilters.scala:61)

at org.apache.spark.sql.catalyst.OrderedFilters.$anonfun$predicates$3$adapted(OrderedFilters.scala:50)

at scala.collection.immutable.List.foreach(List.scala:431)

at org.apache.spark.sql.catalyst.OrderedFilters.<init>(OrderedFilters.scala:50)

at org.apache.spark.sql.catalyst.csv.UnivocityParser.<init>(UnivocityParser.scala:103)

at org.apache.spark.sql.execution.datasources.csv.CSVFileFormat.$anonfun$buildReader$1(CSVFileFormat.scala:130)

at org.apache.spark.sql.execution.datasources.FileFormat$$anon$1.apply(FileFormat.scala:148)

at org.apache.spark.sql.execution.datasources.FileFormat$$anon$1.apply(FileFormat.scala:133)

at org.apache.spark.sql.execution.datasources.FileScanRDD$$anon$1.org$apache$spark$sql$execution$datasources$FileScanRDD$$anon$$readCurrentFile(FileScanRDD.scala:127)

at org.apache.spark.sql.execution.datasources.FileScanRDD$$anon$1.nextIterator(FileScanRDD.scala:187)

at org.apache.spark.sql.execution.datasources.FileScanRDD$$anon$1.hasNext(FileScanRDD.scala:104)

at scala.collection.Iterator$$anon$10.hasNext(Iterator.scala:460)

at org.apache.spark.sql.catalyst.expressions.GeneratedClass$GeneratedIteratorForCodegenStage1.processNext(Unknown Source)

at org.apache.spark.sql.execution.BufferedRowIterator.hasNext(BufferedRowIterator.java:43)

at org.apache.spark.sql.execution.WholeStageCodegenExec$$anon$1.hasNext(WholeStageCodegenExec.scala:759)

at org.apache.spark.sql.execution.SparkPlan.$anonfun$getByteArrayRdd$1(SparkPlan.scala:349)

at org.apache.spark.rdd.RDD.$anonfun$mapPartitionsInternal$2(RDD.scala:898)

at org.apache.spark.rdd.RDD.$anonfun$mapPartitionsInternal$2$adapted(RDD.scala:898)

at org.apache.spark.rdd.MapPartitionsRDD.compute(MapPartitionsRDD.scala:52)

at org.apache.spark.rdd.RDD.computeOrReadCheckpoint(RDD.scala:373)

at org.apache.spark.rdd.RDD.iterator(RDD.scala:337)

at org.apache.spark.scheduler.ResultTask.runTask(ResultTask.scala:90)

at org.apache.spark.scheduler.Task.run(Task.scala:131)

at org.apache.spark.executor.Executor$TaskRunner.$anonfun$run$3(Executor.scala:506)

at org.apache.spark.util.Utils$.tryWithSafeFinally(Utils.scala:1462)

at org.apache.spark.executor.Executor$TaskRunner.run(Executor.scala:509)

at java.util.concurrent.ThreadPoolExecutor.runWorker(ThreadPoolExecutor.java:1149)

at java.util.concurrent.ThreadPoolExecutor$Worker.run(ThreadPoolExecutor.java:624)

at java.lang.Thread.run(Thread.java:750)

24/07/18 20:46:47 ERROR scheduler.TaskSetManager: Task 0 in stage 24.0 failed 1 times; aborting job

Traceback (most recent call last):

File "<stdin>", line 1, in <module>

File "/usr/local/spark/python/pyspark/sql/dataframe.py", line 494, in show

print(self.\_jdf.showString(n, 20, vertical))

File "/usr/local/spark/python/lib/py4j-0.10.9.3-src.zip/py4j/java\_gateway.py", line 1322, in \_\_call\_\_

File "/usr/local/spark/python/pyspark/sql/utils.py", line 111, in deco

return f(\*a, \*\*kw)

File "/usr/local/spark/python/lib/py4j-0.10.9.3-src.zip/py4j/protocol.py", line 328, in get\_return\_value

py4j.protocol.Py4JJavaError: An error occurred while calling o269.showString.

: org.apache.spark.SparkException: Job aborted due to stage failure: Task 0 in stage 24.0 failed 1 times, most recent failure: Lost task 0.0 in stage 24.0 (TID 24) (192.168.112.129 executor driver): java.lang.IllegalArgumentException: RejectRows does not exist. Available: IssuerId, IssuerId2, BusinessDate, StateCode, SourceName, NetworkName, NetworkURL, custnum, MarketCoverage, DentalOnlyPlan

at org.apache.spark.sql.types.StructType.$anonfun$fieldIndex$1(StructType.scala:306)

at scala.collection.immutable.HashMap$HashTrieMap.getOrElse0(HashMap.scala:596)

at scala.collection.immutable.HashMap.getOrElse(HashMap.scala:73)

at org.apache.spark.sql.types.StructType.fieldIndex(StructType.scala:305)

at org.apache.spark.sql.catalyst.OrderedFilters.$anonfun$predicates$4(OrderedFilters.scala:61)

at org.apache.spark.sql.catalyst.OrderedFilters.$anonfun$predicates$4$adapted(OrderedFilters.scala:61)

at scala.collection.TraversableLike.$anonfun$map$1(TraversableLike.scala:286)

at scala.collection.IndexedSeqOptimized.foreach(IndexedSeqOptimized.scala:36)

at scala.collection.IndexedSeqOptimized.foreach$(IndexedSeqOptimized.scala:33)

at scala.collection.mutable.ArrayOps$ofRef.foreach(ArrayOps.scala:198)

at scala.collection.TraversableLike.map(TraversableLike.scala:286)

at scala.collection.TraversableLike.map$(TraversableLike.scala:279)

at scala.collection.mutable.ArrayOps$ofRef.map(ArrayOps.scala:198)

at org.apache.spark.sql.catalyst.OrderedFilters.$anonfun$predicates$3(OrderedFilters.scala:61)

at org.apache.spark.sql.catalyst.OrderedFilters.$anonfun$predicates$3$adapted(OrderedFilters.scala:50)

at scala.collection.immutable.List.foreach(List.scala:431)

at org.apache.spark.sql.catalyst.OrderedFilters.<init>(OrderedFilters.scala:50)

at org.apache.spark.sql.catalyst.csv.UnivocityParser.<init>(UnivocityParser.scala:103)

at org.apache.spark.sql.execution.datasources.csv.CSVFileFormat.$anonfun$buildReader$1(CSVFileFormat.scala:130)

at org.apache.spark.sql.execution.datasources.FileFormat$$anon$1.apply(FileFormat.scala:148)

at org.apache.spark.sql.execution.datasources.FileFormat$$anon$1.apply(FileFormat.scala:133)

at org.apache.spark.sql.execution.datasources.FileScanRDD$$anon$1.org$apache$spark$sql$execution$datasources$FileScanRDD$$anon$$readCurrentFile(FileScanRDD.scala:127)

at org.apache.spark.sql.execution.datasources.FileScanRDD$$anon$1.nextIterator(FileScanRDD.scala:187)

at org.apache.spark.sql.execution.datasources.FileScanRDD$$anon$1.hasNext(FileScanRDD.scala:104)

at scala.collection.Iterator$$anon$10.hasNext(Iterator.scala:460)

at org.apache.spark.sql.catalyst.expressions.GeneratedClass$GeneratedIteratorForCodegenStage1.processNext(Unknown Source)

at org.apache.spark.sql.execution.BufferedRowIterator.hasNext(BufferedRowIterator.java:43)

at org.apache.spark.sql.execution.WholeStageCodegenExec$$anon$1.hasNext(WholeStageCodegenExec.scala:759)

at org.apache.spark.sql.execution.SparkPlan.$anonfun$getByteArrayRdd$1(SparkPlan.scala:349)

at org.apache.spark.rdd.RDD.$anonfun$mapPartitionsInternal$2(RDD.scala:898)

at org.apache.spark.rdd.RDD.$anonfun$mapPartitionsInternal$2$adapted(RDD.scala:898)

at org.apache.spark.rdd.MapPartitionsRDD.compute(MapPartitionsRDD.scala:52)

at org.apache.spark.rdd.RDD.computeOrReadCheckpoint(RDD.scala:373)

at org.apache.spark.rdd.RDD.iterator(RDD.scala:337)

at org.apache.spark.scheduler.ResultTask.runTask(ResultTask.scala:90)

at org.apache.spark.scheduler.Task.run(Task.scala:131)

at org.apache.spark.executor.Executor$TaskRunner.$anonfun$run$3(Executor.scala:506)

at org.apache.spark.util.Utils$.tryWithSafeFinally(Utils.scala:1462)

at org.apache.spark.executor.Executor$TaskRunner.run(Executor.scala:509)

at java.util.concurrent.ThreadPoolExecutor.runWorker(ThreadPoolExecutor.java:1149)

at java.util.concurrent.ThreadPoolExecutor$Worker.run(ThreadPoolExecutor.java:624)

at java.lang.Thread.run(Thread.java:750)

Driver stacktrace:

at org.apache.spark.scheduler.DAGScheduler.failJobAndIndependentStages(DAGScheduler.scala:2454)

at org.apache.spark.scheduler.DAGScheduler.$anonfun$abortStage$2(DAGScheduler.scala:2403)

at org.apache.spark.scheduler.DAGScheduler.$anonfun$abortStage$2$adapted(DAGScheduler.scala:2402)

at scala.collection.mutable.ResizableArray.foreach(ResizableArray.scala:62)

at scala.collection.mutable.ResizableArray.foreach$(ResizableArray.scala:55)

at scala.collection.mutable.ArrayBuffer.foreach(ArrayBuffer.scala:49)

at org.apache.spark.scheduler.DAGScheduler.abortStage(DAGScheduler.scala:2402)

at org.apache.spark.scheduler.DAGScheduler.$anonfun$handleTaskSetFailed$1(DAGScheduler.scala:1160)

at org.apache.spark.scheduler.DAGScheduler.$anonfun$handleTaskSetFailed$1$adapted(DAGScheduler.scala:1160)

at scala.Option.foreach(Option.scala:407)

at org.apache.spark.scheduler.DAGScheduler.handleTaskSetFailed(DAGScheduler.scala:1160)

at org.apache.spark.scheduler.DAGSchedulerEventProcessLoop.doOnReceive(DAGScheduler.scala:2642)

at org.apache.spark.scheduler.DAGSchedulerEventProcessLoop.onReceive(DAGScheduler.scala:2584)

at org.apache.spark.scheduler.DAGSchedulerEventProcessLoop.onReceive(DAGScheduler.scala:2573)

at org.apache.spark.util.EventLoop$$anon$1.run(EventLoop.scala:49)

at org.apache.spark.scheduler.DAGScheduler.runJob(DAGScheduler.scala:938)

at org.apache.spark.SparkContext.runJob(SparkContext.scala:2214)

at org.apache.spark.SparkContext.runJob(SparkContext.scala:2235)

at org.apache.spark.SparkContext.runJob(SparkContext.scala:2254)

at org.apache.spark.sql.execution.SparkPlan.executeTake(SparkPlan.scala:476)

at org.apache.spark.sql.execution.SparkPlan.executeTake(SparkPlan.scala:429)

at org.apache.spark.sql.execution.CollectLimitExec.executeCollect(limit.scala:48)

at org.apache.spark.sql.Dataset.collectFromPlan(Dataset.scala:3715)

at org.apache.spark.sql.Dataset.$anonfun$head$1(Dataset.scala:2728)

at org.apache.spark.sql.Dataset.$anonfun$withAction$1(Dataset.scala:3706)

at org.apache.spark.sql.execution.SQLExecution$.$anonfun$withNewExecutionId$5(SQLExecution.scala:103)

at org.apache.spark.sql.execution.SQLExecution$.withSQLConfPropagated(SQLExecution.scala:163)

at org.apache.spark.sql.execution.SQLExecution$.$anonfun$withNewExecutionId$1(SQLExecution.scala:90)

at org.apache.spark.sql.SparkSession.withActive(SparkSession.scala:775)

at org.apache.spark.sql.execution.SQLExecution$.withNewExecutionId(SQLExecution.scala:64)

at org.apache.spark.sql.Dataset.withAction(Dataset.scala:3704)

at org.apache.spark.sql.Dataset.head(Dataset.scala:2728)

at org.apache.spark.sql.Dataset.take(Dataset.scala:2935)

at org.apache.spark.sql.Dataset.getRows(Dataset.scala:287)

at org.apache.spark.sql.Dataset.showString(Dataset.scala:326)

at sun.reflect.GeneratedMethodAccessor41.invoke(Unknown Source)

at sun.reflect.DelegatingMethodAccessorImpl.invoke(DelegatingMethodAccessorImpl.java:43)

at java.lang.reflect.Method.invoke(Method.java:498)

at py4j.reflection.MethodInvoker.invoke(MethodInvoker.java:244)

at py4j.reflection.ReflectionEngine.invoke(ReflectionEngine.java:357)

at py4j.Gateway.invoke(Gateway.java:282)

at py4j.commands.AbstractCommand.invokeMethod(AbstractCommand.java:132)

at py4j.commands.CallCommand.execute(CallCommand.java:79)

at py4j.ClientServerConnection.waitForCommands(ClientServerConnection.java:182)

at py4j.ClientServerConnection.run(ClientServerConnection.java:106)

at java.lang.Thread.run(Thread.java:750)

Caused by: java.lang.IllegalArgumentException: RejectRows does not exist. Available: IssuerId, IssuerId2, BusinessDate, StateCode, SourceName, NetworkName, NetworkURL, custnum, MarketCoverage, DentalOnlyPlan

at org.apache.spark.sql.types.StructType.$anonfun$fieldIndex$1(StructType.scala:306)

at scala.collection.immutable.HashMap$HashTrieMap.getOrElse0(HashMap.scala:596)

at scala.collection.immutable.HashMap.getOrElse(HashMap.scala:73)

at org.apache.spark.sql.types.StructType.fieldIndex(StructType.scala:305)

at org.apache.spark.sql.catalyst.OrderedFilters.$anonfun$predicates$4(OrderedFilters.scala:61)

at org.apache.spark.sql.catalyst.OrderedFilters.$anonfun$predicates$4$adapted(OrderedFilters.scala:61)

at scala.collection.TraversableLike.$anonfun$map$1(TraversableLike.scala:286)

at scala.collection.IndexedSeqOptimized.foreach(IndexedSeqOptimized.scala:36)

at scala.collection.IndexedSeqOptimized.foreach$(IndexedSeqOptimized.scala:33)

at scala.collection.mutable.ArrayOps$ofRef.foreach(ArrayOps.scala:198)

at scala.collection.TraversableLike.map(TraversableLike.scala:286)

at scala.collection.TraversableLike.map$(TraversableLike.scala:279)

at scala.collection.mutable.ArrayOps$ofRef.map(ArrayOps.scala:198)

at org.apache.spark.sql.catalyst.OrderedFilters.$anonfun$predicates$3(OrderedFilters.scala:61)

at org.apache.spark.sql.catalyst.OrderedFilters.$anonfun$predicates$3$adapted(OrderedFilters.scala:50)

at scala.collection.immutable.List.foreach(List.scala:431)

at org.apache.spark.sql.catalyst.OrderedFilters.<init>(OrderedFilters.scala:50)

at org.apache.spark.sql.catalyst.csv.UnivocityParser.<init>(UnivocityParser.scala:103)

at org.apache.spark.sql.execution.datasources.csv.CSVFileFormat.$anonfun$buildReader$1(CSVFileFormat.scala:130)

at org.apache.spark.sql.execution.datasources.FileFormat$$anon$1.apply(FileFormat.scala:148)

at org.apache.spark.sql.execution.datasources.FileFormat$$anon$1.apply(FileFormat.scala:133)

at org.apache.spark.sql.execution.datasources.FileScanRDD$$anon$1.org$apache$spark$sql$execution$datasources$FileScanRDD$$anon$$readCurrentFile(FileScanRDD.scala:127)

at org.apache.spark.sql.execution.datasources.FileScanRDD$$anon$1.nextIterator(FileScanRDD.scala:187)

at org.apache.spark.sql.execution.datasources.FileScanRDD$$anon$1.hasNext(FileScanRDD.scala:104)

at scala.collection.Iterator$$anon$10.hasNext(Iterator.scala:460)

at org.apache.spark.sql.catalyst.expressions.GeneratedClass$GeneratedIteratorForCodegenStage1.processNext(Unknown Source)

at org.apache.spark.sql.execution.BufferedRowIterator.hasNext(BufferedRowIterator.java:43)

at org.apache.spark.sql.execution.WholeStageCodegenExec$$anon$1.hasNext(WholeStageCodegenExec.scala:759)

at org.apache.spark.sql.execution.SparkPlan.$anonfun$getByteArrayRdd$1(SparkPlan.scala:349)

at org.apache.spark.rdd.RDD.$anonfun$mapPartitionsInternal$2(RDD.scala:898)

at org.apache.spark.rdd.RDD.$anonfun$mapPartitionsInternal$2$adapted(RDD.scala:898)

at org.apache.spark.rdd.MapPartitionsRDD.compute(MapPartitionsRDD.scala:52)

at org.apache.spark.rdd.RDD.computeOrReadCheckpoint(RDD.scala:373)

at org.apache.spark.rdd.RDD.iterator(RDD.scala:337)

at org.apache.spark.scheduler.ResultTask.runTask(ResultTask.scala:90)

at org.apache.spark.scheduler.Task.run(Task.scala:131)

at org.apache.spark.executor.Executor$TaskRunner.$anonfun$run$3(Executor.scala:506)

at org.apache.spark.util.Utils$.tryWithSafeFinally(Utils.scala:1462)

at org.apache.spark.executor.Executor$TaskRunner.run(Executor.scala:509)

at java.util.concurrent.ThreadPoolExecutor.runWorker(ThreadPoolExecutor.java:1149)

at java.util.concurrent.ThreadPoolExecutor$Worker.run(ThreadPoolExecutor.java:624)

... 1 more

>>> data\_frame\_3.where(data\_frame\_1.RejectRows.isNotNull()).show(3)

Traceback (most recent call last):

File "<stdin>", line 1, in <module>

File "/usr/local/spark/python/pyspark/sql/dataframe.py", line 1733, in filter

jdf = self.\_jdf.filter(condition.\_jc)

File "/usr/local/spark/python/lib/py4j-0.10.9.3-src.zip/py4j/java\_gateway.py", line 1322, in \_\_call\_\_

File "/usr/local/spark/python/pyspark/sql/utils.py", line 117, in deco

raise converted from None

pyspark.sql.utils.AnalysisException: Resolved attribute(s) RejectRows#1386 missing from IssuerId#491,IssuerId2#492,BusinessDate#493,StateCode#494,SourceName#495,NetworkName#496,NetworkURL#497,custnum#498,MarketCoverage#499,DentalOnlyPlan#500,RejectRows#501 in operator !Filter isnotnull(RejectRows#1386). Attribute(s) with the same name appear in the operation: RejectRows. Please check if the right attribute(s) are used.;

!Filter isnotnull(RejectRows#1386)

+- Relation [IssuerId#491,IssuerId2#492,BusinessDate#493,StateCode#494,SourceName#495,NetworkName#496,NetworkURL#497,custnum#498,MarketCoverage#499,DentalOnlyPlan#500,RejectRows#501] csv

>>> data\_frame\_3.where("RejectRows.isNotNull()").show(3)

24/07/18 20:47:13 WARN conf.HiveConf: HiveConf of name hive.stats.jdbc.timeout does not exist

24/07/18 20:47:13 WARN conf.HiveConf: HiveConf of name hive.stats.retries.wait does not exist

^CERROR:root:Exception while sending command.

Traceback (most recent call last):

File "/usr/local/spark/python/lib/py4j-0.10.9.3-src.zip/py4j/clientserver.py", line 475, in send\_command

answer = smart\_decode(self.stream.readline()[:-1])

RuntimeError: reentrant call inside <\_io.BufferedReader name=4>

During handling of the above exception, another exception occurred:

Traceback (most recent call last):

File "/usr/local/spark/python/lib/py4j-0.10.9.3-src.zip/py4j/java\_gateway.py", line 1038, in send\_command

response = connection.send\_command(command)

File "/usr/local/spark/python/lib/py4j-0.10.9.3-src.zip/py4j/clientserver.py", line 504, in send\_command

"Error while sending or receiving", e, proto.ERROR\_ON\_RECEIVE)

py4j.protocol.Py4JNetworkError: Error while sending or receiving

ERROR:root:Exception while sending command.

Traceback (most recent call last):

File "/usr/local/spark/python/lib/py4j-0.10.9.3-src.zip/py4j/clientserver.py", line 475, in send\_command

answer = smart\_decode(self.stream.readline()[:-1])

File "/usr/lib64/python3.6/socket.py", line 586, in readinto

return self.\_sock.recv\_into(b)

File "/usr/local/spark/python/pyspark/context.py", line 292, in signal\_handler

self.cancelAllJobs()

File "/usr/local/spark/python/pyspark/context.py", line 1195, in cancelAllJobs

self.\_jsc.sc().cancelAllJobs()

File "/usr/local/spark/python/lib/py4j-0.10.9.3-src.zip/py4j/java\_gateway.py", line 1322, in \_\_call\_\_

answer, self.gateway\_client, self.target\_id, self.name)

File "/usr/local/spark/python/pyspark/sql/utils.py", line 111, in deco

return f(\*a, \*\*kw)

File "/usr/local/spark/python/lib/py4j-0.10.9.3-src.zip/py4j/protocol.py", line 336, in get\_return\_value

format(target\_id, ".", name))

py4j.protocol.Py4JError: An error occurred while calling o15.sc

During handling of the above exception, another exception occurred:

Traceback (most recent call last):

File "/usr/local/spark/python/lib/py4j-0.10.9.3-src.zip/py4j/java\_gateway.py", line 1038, in send\_command

response = connection.send\_command(command)

File "/usr/local/spark/python/lib/py4j-0.10.9.3-src.zip/py4j/clientserver.py", line 504, in send\_command

"Error while sending or receiving", e, proto.ERROR\_ON\_RECEIVE)

py4j.protocol.Py4JNetworkError: Error while sending or receiving

Traceback (most recent call last):

File "<stdin>", line 1, in <module>

File "/usr/local/spark/python/pyspark/sql/dataframe.py", line 1731, in filter

jdf = self.\_jdf.filter(condition)

File "/usr/local/spark/python/lib/py4j-0.10.9.3-src.zip/py4j/java\_gateway.py", line 1322, in \_\_call\_\_

File "/usr/local/spark/python/pyspark/sql/utils.py", line 111, in deco

return f(\*a, \*\*kw)

File "/usr/local/spark/python/lib/py4j-0.10.9.3-src.zip/py4j/protocol.py", line 336, in get\_return\_value

py4j.protocol.Py4JError: An error occurred while calling o186.filter

>>>

>>> data\_frame\_1.create24/07/18 20:47:28 WARN metastore.ObjectStore: Failed to get database rejectrows, returning NoSuchObjectException

>>> data\_frame\_1.createOrReplaceTempView("table1")

>>> spark.sql("select \* from table1 where RejectRows is not NULL ").show()

24/07/18 20:49:13 ERROR executor.Executor: Exception in task 0.0 in stage 25.0 (TID 25)

java.lang.IllegalArgumentException: RejectRows does not exist. Available: IssuerId, IssuerId2, BusinessDate, StateCode, SourceName, NetworkName, NetworkURL, custnum, MarketCoverage, DentalOnlyPlan

at org.apache.spark.sql.types.StructType.$anonfun$fieldIndex$1(StructType.scala:306)

at scala.collection.immutable.HashMap$HashTrieMap.getOrElse0(HashMap.scala:596)

at scala.collection.immutable.HashMap.getOrElse(HashMap.scala:73)

at org.apache.spark.sql.types.StructType.fieldIndex(StructType.scala:305)

at org.apache.spark.sql.catalyst.OrderedFilters.$anonfun$predicates$4(OrderedFilters.scala:61)

at org.apache.spark.sql.catalyst.OrderedFilters.$anonfun$predicates$4$adapted(OrderedFilters.scala:61)

at scala.collection.TraversableLike.$anonfun$map$1(TraversableLike.scala:286)

at scala.collection.IndexedSeqOptimized.foreach(IndexedSeqOptimized.scala:36)

at scala.collection.IndexedSeqOptimized.foreach$(IndexedSeqOptimized.scala:33)

at scala.collection.mutable.ArrayOps$ofRef.foreach(ArrayOps.scala:198)

at scala.collection.TraversableLike.map(TraversableLike.scala:286)

at scala.collection.TraversableLike.map$(TraversableLike.scala:279)

at scala.collection.mutable.ArrayOps$ofRef.map(ArrayOps.scala:198)

at org.apache.spark.sql.catalyst.OrderedFilters.$anonfun$predicates$3(OrderedFilters.scala:61)

at org.apache.spark.sql.catalyst.OrderedFilters.$anonfun$predicates$3$adapted(OrderedFilters.scala:50)

at scala.collection.immutable.List.foreach(List.scala:431)

at org.apache.spark.sql.catalyst.OrderedFilters.<init>(OrderedFilters.scala:50)

at org.apache.spark.sql.catalyst.csv.UnivocityParser.<init>(UnivocityParser.scala:103)

at org.apache.spark.sql.execution.datasources.csv.CSVFileFormat.$anonfun$buildReader$1(CSVFileFormat.scala:130)

at org.apache.spark.sql.execution.datasources.FileFormat$$anon$1.apply(FileFormat.scala:148)

at org.apache.spark.sql.execution.datasources.FileFormat$$anon$1.apply(FileFormat.scala:133)

at org.apache.spark.sql.execution.datasources.FileScanRDD$$anon$1.org$apache$spark$sql$execution$datasources$FileScanRDD$$anon$$readCurrentFile(FileScanRDD.scala:127)

at org.apache.spark.sql.execution.datasources.FileScanRDD$$anon$1.nextIterator(FileScanRDD.scala:187)

at org.apache.spark.sql.execution.datasources.FileScanRDD$$anon$1.hasNext(FileScanRDD.scala:104)

at scala.collection.Iterator$$anon$10.hasNext(Iterator.scala:460)

at org.apache.spark.sql.catalyst.expressions.GeneratedClass$GeneratedIteratorForCodegenStage1.processNext(Unknown Source)

at org.apache.spark.sql.execution.BufferedRowIterator.hasNext(BufferedRowIterator.java:43)

at org.apache.spark.sql.execution.WholeStageCodegenExec$$anon$1.hasNext(WholeStageCodegenExec.scala:759)

at org.apache.spark.sql.execution.SparkPlan.$anonfun$getByteArrayRdd$1(SparkPlan.scala:349)

at org.apache.spark.rdd.RDD.$anonfun$mapPartitionsInternal$2(RDD.scala:898)

at org.apache.spark.rdd.RDD.$anonfun$mapPartitionsInternal$2$adapted(RDD.scala:898)

at org.apache.spark.rdd.MapPartitionsRDD.compute(MapPartitionsRDD.scala:52)

at org.apache.spark.rdd.RDD.computeOrReadCheckpoint(RDD.scala:373)

at org.apache.spark.rdd.RDD.iterator(RDD.scala:337)

at org.apache.spark.scheduler.ResultTask.runTask(ResultTask.scala:90)

at org.apache.spark.scheduler.Task.run(Task.scala:131)

at org.apache.spark.executor.Executor$TaskRunner.$anonfun$run$3(Executor.scala:506)

at org.apache.spark.util.Utils$.tryWithSafeFinally(Utils.scala:1462)

at org.apache.spark.executor.Executor$TaskRunner.run(Executor.scala:509)

at java.util.concurrent.ThreadPoolExecutor.runWorker(ThreadPoolExecutor.java:1149)

at java.util.concurrent.ThreadPoolExecutor$Worker.run(ThreadPoolExecutor.java:624)

at java.lang.Thread.run(Thread.java:750)

24/07/18 20:49:13 WARN scheduler.TaskSetManager: Lost task 0.0 in stage 25.0 (TID 25) (192.168.112.129 executor driver): java.lang.IllegalArgumentException: RejectRows does not exist. Available: IssuerId, IssuerId2, BusinessDate, StateCode, SourceName, NetworkName, NetworkURL, custnum, MarketCoverage, DentalOnlyPlan

at org.apache.spark.sql.types.StructType.$anonfun$fieldIndex$1(StructType.scala:306)

at scala.collection.immutable.HashMap$HashTrieMap.getOrElse0(HashMap.scala:596)

at scala.collection.immutable.HashMap.getOrElse(HashMap.scala:73)

at org.apache.spark.sql.types.StructType.fieldIndex(StructType.scala:305)

at org.apache.spark.sql.catalyst.OrderedFilters.$anonfun$predicates$4(OrderedFilters.scala:61)

at org.apache.spark.sql.catalyst.OrderedFilters.$anonfun$predicates$4$adapted(OrderedFilters.scala:61)

at scala.collection.TraversableLike.$anonfun$map$1(TraversableLike.scala:286)

at scala.collection.IndexedSeqOptimized.foreach(IndexedSeqOptimized.scala:36)

at scala.collection.IndexedSeqOptimized.foreach$(IndexedSeqOptimized.scala:33)

at scala.collection.mutable.ArrayOps$ofRef.foreach(ArrayOps.scala:198)

at scala.collection.TraversableLike.map(TraversableLike.scala:286)

at scala.collection.TraversableLike.map$(TraversableLike.scala:279)

at scala.collection.mutable.ArrayOps$ofRef.map(ArrayOps.scala:198)

at org.apache.spark.sql.catalyst.OrderedFilters.$anonfun$predicates$3(OrderedFilters.scala:61)

at org.apache.spark.sql.catalyst.OrderedFilters.$anonfun$predicates$3$adapted(OrderedFilters.scala:50)

at scala.collection.immutable.List.foreach(List.scala:431)

at org.apache.spark.sql.catalyst.OrderedFilters.<init>(OrderedFilters.scala:50)

at org.apache.spark.sql.catalyst.csv.UnivocityParser.<init>(UnivocityParser.scala:103)

at org.apache.spark.sql.execution.datasources.csv.CSVFileFormat.$anonfun$buildReader$1(CSVFileFormat.scala:130)

at org.apache.spark.sql.execution.datasources.FileFormat$$anon$1.apply(FileFormat.scala:148)

at org.apache.spark.sql.execution.datasources.FileFormat$$anon$1.apply(FileFormat.scala:133)

at org.apache.spark.sql.execution.datasources.FileScanRDD$$anon$1.org$apache$spark$sql$execution$datasources$FileScanRDD$$anon$$readCurrentFile(FileScanRDD.scala:127)

at org.apache.spark.sql.execution.datasources.FileScanRDD$$anon$1.nextIterator(FileScanRDD.scala:187)

at org.apache.spark.sql.execution.datasources.FileScanRDD$$anon$1.hasNext(FileScanRDD.scala:104)

at scala.collection.Iterator$$anon$10.hasNext(Iterator.scala:460)

at org.apache.spark.sql.catalyst.expressions.GeneratedClass$GeneratedIteratorForCodegenStage1.processNext(Unknown Source)

at org.apache.spark.sql.execution.BufferedRowIterator.hasNext(BufferedRowIterator.java:43)

at org.apache.spark.sql.execution.WholeStageCodegenExec$$anon$1.hasNext(WholeStageCodegenExec.scala:759)

at org.apache.spark.sql.execution.SparkPlan.$anonfun$getByteArrayRdd$1(SparkPlan.scala:349)

at org.apache.spark.rdd.RDD.$anonfun$mapPartitionsInternal$2(RDD.scala:898)

at org.apache.spark.rdd.RDD.$anonfun$mapPartitionsInternal$2$adapted(RDD.scala:898)

at org.apache.spark.rdd.MapPartitionsRDD.compute(MapPartitionsRDD.scala:52)

at org.apache.spark.rdd.RDD.computeOrReadCheckpoint(RDD.scala:373)

at org.apache.spark.rdd.RDD.iterator(RDD.scala:337)

at org.apache.spark.scheduler.ResultTask.runTask(ResultTask.scala:90)

at org.apache.spark.scheduler.Task.run(Task.scala:131)

at org.apache.spark.executor.Executor$TaskRunner.$anonfun$run$3(Executor.scala:506)

at org.apache.spark.util.Utils$.tryWithSafeFinally(Utils.scala:1462)

at org.apache.spark.executor.Executor$TaskRunner.run(Executor.scala:509)

at java.util.concurrent.ThreadPoolExecutor.runWorker(ThreadPoolExecutor.java:1149)

at java.util.concurrent.ThreadPoolExecutor$Worker.run(ThreadPoolExecutor.java:624)

at java.lang.Thread.run(Thread.java:750)

24/07/18 20:49:13 ERROR scheduler.TaskSetManager: Task 0 in stage 25.0 failed 1 times; aborting job

Traceback (most recent call last):

File "<stdin>", line 1, in <module>

File "/usr/local/spark/python/pyspark/sql/dataframe.py", line 494, in show

print(self.\_jdf.showString(n, 20, vertical))

File "/usr/local/spark/python/lib/py4j-0.10.9.3-src.zip/py4j/java\_gateway.py", line 1322, in \_\_call\_\_

File "/usr/local/spark/python/pyspark/sql/utils.py", line 111, in deco

return f(\*a, \*\*kw)

File "/usr/local/spark/python/lib/py4j-0.10.9.3-src.zip/py4j/protocol.py", line 328, in get\_return\_value

py4j.protocol.Py4JJavaError: An error occurred while calling o278.showString.

: org.apache.spark.SparkException: Job aborted due to stage failure: Task 0 in stage 25.0 failed 1 times, most recent failure: Lost task 0.0 in stage 25.0 (TID 25) (192.168.112.129 executor driver): java.lang.IllegalArgumentException: RejectRows does not exist. Available: IssuerId, IssuerId2, BusinessDate, StateCode, SourceName, NetworkName, NetworkURL, custnum, MarketCoverage, DentalOnlyPlan

at org.apache.spark.sql.types.StructType.$anonfun$fieldIndex$1(StructType.scala:306)

at scala.collection.immutable.HashMap$HashTrieMap.getOrElse0(HashMap.scala:596)

at scala.collection.immutable.HashMap.getOrElse(HashMap.scala:73)

at org.apache.spark.sql.types.StructType.fieldIndex(StructType.scala:305)

at org.apache.spark.sql.catalyst.OrderedFilters.$anonfun$predicates$4(OrderedFilters.scala:61)

at org.apache.spark.sql.catalyst.OrderedFilters.$anonfun$predicates$4$adapted(OrderedFilters.scala:61)

at scala.collection.TraversableLike.$anonfun$map$1(TraversableLike.scala:286)

at scala.collection.IndexedSeqOptimized.foreach(IndexedSeqOptimized.scala:36)

at scala.collection.IndexedSeqOptimized.foreach$(IndexedSeqOptimized.scala:33)

at scala.collection.mutable.ArrayOps$ofRef.foreach(ArrayOps.scala:198)

at scala.collection.TraversableLike.map(TraversableLike.scala:286)

at scala.collection.TraversableLike.map$(TraversableLike.scala:279)

at scala.collection.mutable.ArrayOps$ofRef.map(ArrayOps.scala:198)

at org.apache.spark.sql.catalyst.OrderedFilters.$anonfun$predicates$3(OrderedFilters.scala:61)

at org.apache.spark.sql.catalyst.OrderedFilters.$anonfun$predicates$3$adapted(OrderedFilters.scala:50)

at scala.collection.immutable.List.foreach(List.scala:431)

at org.apache.spark.sql.catalyst.OrderedFilters.<init>(OrderedFilters.scala:50)

at org.apache.spark.sql.catalyst.csv.UnivocityParser.<init>(UnivocityParser.scala:103)

at org.apache.spark.sql.execution.datasources.csv.CSVFileFormat.$anonfun$buildReader$1(CSVFileFormat.scala:130)

at org.apache.spark.sql.execution.datasources.FileFormat$$anon$1.apply(FileFormat.scala:148)

at org.apache.spark.sql.execution.datasources.FileFormat$$anon$1.apply(FileFormat.scala:133)

at org.apache.spark.sql.execution.datasources.FileScanRDD$$anon$1.org$apache$spark$sql$execution$datasources$FileScanRDD$$anon$$readCurrentFile(FileScanRDD.scala:127)

at org.apache.spark.sql.execution.datasources.FileScanRDD$$anon$1.nextIterator(FileScanRDD.scala:187)

at org.apache.spark.sql.execution.datasources.FileScanRDD$$anon$1.hasNext(FileScanRDD.scala:104)

at scala.collection.Iterator$$anon$10.hasNext(Iterator.scala:460)

at org.apache.spark.sql.catalyst.expressions.GeneratedClass$GeneratedIteratorForCodegenStage1.processNext(Unknown Source)

at org.apache.spark.sql.execution.BufferedRowIterator.hasNext(BufferedRowIterator.java:43)

at org.apache.spark.sql.execution.WholeStageCodegenExec$$anon$1.hasNext(WholeStageCodegenExec.scala:759)

at org.apache.spark.sql.execution.SparkPlan.$anonfun$getByteArrayRdd$1(SparkPlan.scala:349)

at org.apache.spark.rdd.RDD.$anonfun$mapPartitionsInternal$2(RDD.scala:898)

at org.apache.spark.rdd.RDD.$anonfun$mapPartitionsInternal$2$adapted(RDD.scala:898)

at org.apache.spark.rdd.MapPartitionsRDD.compute(MapPartitionsRDD.scala:52)

at org.apache.spark.rdd.RDD.computeOrReadCheckpoint(RDD.scala:373)

at org.apache.spark.rdd.RDD.iterator(RDD.scala:337)

at org.apache.spark.scheduler.ResultTask.runTask(ResultTask.scala:90)

at org.apache.spark.scheduler.Task.run(Task.scala:131)

at org.apache.spark.executor.Executor$TaskRunner.$anonfun$run$3(Executor.scala:506)

at org.apache.spark.util.Utils$.tryWithSafeFinally(Utils.scala:1462)

at org.apache.spark.executor.Executor$TaskRunner.run(Executor.scala:509)

at java.util.concurrent.ThreadPoolExecutor.runWorker(ThreadPoolExecutor.java:1149)

at java.util.concurrent.ThreadPoolExecutor$Worker.run(ThreadPoolExecutor.java:624)

at java.lang.Thread.run(Thread.java:750)

Driver stacktrace:

at org.apache.spark.scheduler.DAGScheduler.failJobAndIndependentStages(DAGScheduler.scala:2454)

at org.apache.spark.scheduler.DAGScheduler.$anonfun$abortStage$2(DAGScheduler.scala:2403)

at org.apache.spark.scheduler.DAGScheduler.$anonfun$abortStage$2$adapted(DAGScheduler.scala:2402)

at scala.collection.mutable.ResizableArray.foreach(ResizableArray.scala:62)

at scala.collection.mutable.ResizableArray.foreach$(ResizableArray.scala:55)

at scala.collection.mutable.ArrayBuffer.foreach(ArrayBuffer.scala:49)

at org.apache.spark.scheduler.DAGScheduler.abortStage(DAGScheduler.scala:2402)

at org.apache.spark.scheduler.DAGScheduler.$anonfun$handleTaskSetFailed$1(DAGScheduler.scala:1160)

at org.apache.spark.scheduler.DAGScheduler.$anonfun$handleTaskSetFailed$1$adapted(DAGScheduler.scala:1160)

at scala.Option.foreach(Option.scala:407)

at org.apache.spark.scheduler.DAGScheduler.handleTaskSetFailed(DAGScheduler.scala:1160)

at org.apache.spark.scheduler.DAGSchedulerEventProcessLoop.doOnReceive(DAGScheduler.scala:2642)

at org.apache.spark.scheduler.DAGSchedulerEventProcessLoop.onReceive(DAGScheduler.scala:2584)

at org.apache.spark.scheduler.DAGSchedulerEventProcessLoop.onReceive(DAGScheduler.scala:2573)

at org.apache.spark.util.EventLoop$$anon$1.run(EventLoop.scala:49)

at org.apache.spark.scheduler.DAGScheduler.runJob(DAGScheduler.scala:938)

at org.apache.spark.SparkContext.runJob(SparkContext.scala:2214)

at org.apache.spark.SparkContext.runJob(SparkContext.scala:2235)

at org.apache.spark.SparkContext.runJob(SparkContext.scala:2254)

at org.apache.spark.sql.execution.SparkPlan.executeTake(SparkPlan.scala:476)

at org.apache.spark.sql.execution.SparkPlan.executeTake(SparkPlan.scala:429)

at org.apache.spark.sql.execution.CollectLimitExec.executeCollect(limit.scala:48)

at org.apache.spark.sql.Dataset.collectFromPlan(Dataset.scala:3715)

at org.apache.spark.sql.Dataset.$anonfun$head$1(Dataset.scala:2728)

at org.apache.spark.sql.Dataset.$anonfun$withAction$1(Dataset.scala:3706)

at org.apache.spark.sql.execution.SQLExecution$.$anonfun$withNewExecutionId$5(SQLExecution.scala:103)

at org.apache.spark.sql.execution.SQLExecution$.withSQLConfPropagated(SQLExecution.scala:163)

at org.apache.spark.sql.execution.SQLExecution$.$anonfun$withNewExecutionId$1(SQLExecution.scala:90)

at org.apache.spark.sql.SparkSession.withActive(SparkSession.scala:775)

at org.apache.spark.sql.execution.SQLExecution$.withNewExecutionId(SQLExecution.scala:64)

at org.apache.spark.sql.Dataset.withAction(Dataset.scala:3704)

at org.apache.spark.sql.Dataset.head(Dataset.scala:2728)

at org.apache.spark.sql.Dataset.take(Dataset.scala:2935)

at org.apache.spark.sql.Dataset.getRows(Dataset.scala:287)

at org.apache.spark.sql.Dataset.showString(Dataset.scala:326)

at sun.reflect.GeneratedMethodAccessor41.invoke(Unknown Source)

at sun.reflect.DelegatingMethodAccessorImpl.invoke(DelegatingMethodAccessorImpl.java:43)

at java.lang.reflect.Method.invoke(Method.java:498)

at py4j.reflection.MethodInvoker.invoke(MethodInvoker.java:244)

at py4j.reflection.ReflectionEngine.invoke(ReflectionEngine.java:357)

at py4j.Gateway.invoke(Gateway.java:282)

at py4j.commands.AbstractCommand.invokeMethod(AbstractCommand.java:132)

at py4j.commands.CallCommand.execute(CallCommand.java:79)

at py4j.ClientServerConnection.waitForCommands(ClientServerConnection.java:182)

at py4j.ClientServerConnection.run(ClientServerConnection.java:106)

at java.lang.Thread.run(Thread.java:750)

Caused by: java.lang.IllegalArgumentException: RejectRows does not exist. Available: IssuerId, IssuerId2, BusinessDate, StateCode, SourceName, NetworkName, NetworkURL, custnum, MarketCoverage, DentalOnlyPlan

at org.apache.spark.sql.types.StructType.$anonfun$fieldIndex$1(StructType.scala:306)

at scala.collection.immutable.HashMap$HashTrieMap.getOrElse0(HashMap.scala:596)

at scala.collection.immutable.HashMap.getOrElse(HashMap.scala:73)

at org.apache.spark.sql.types.StructType.fieldIndex(StructType.scala:305)

at org.apache.spark.sql.catalyst.OrderedFilters.$anonfun$predicates$4(OrderedFilters.scala:61)

at org.apache.spark.sql.catalyst.OrderedFilters.$anonfun$predicates$4$adapted(OrderedFilters.scala:61)

at scala.collection.TraversableLike.$anonfun$map$1(TraversableLike.scala:286)

at scala.collection.IndexedSeqOptimized.foreach(IndexedSeqOptimized.scala:36)

at scala.collection.IndexedSeqOptimized.foreach$(IndexedSeqOptimized.scala:33)

at scala.collection.mutable.ArrayOps$ofRef.foreach(ArrayOps.scala:198)

at scala.collection.TraversableLike.map(TraversableLike.scala:286)

at scala.collection.TraversableLike.map$(TraversableLike.scala:279)

at scala.collection.mutable.ArrayOps$ofRef.map(ArrayOps.scala:198)

at org.apache.spark.sql.catalyst.OrderedFilters.$anonfun$predicates$3(OrderedFilters.scala:61)

at org.apache.spark.sql.catalyst.OrderedFilters.$anonfun$predicates$3$adapted(OrderedFilters.scala:50)

at scala.collection.immutable.List.foreach(List.scala:431)

at org.apache.spark.sql.catalyst.OrderedFilters.<init>(OrderedFilters.scala:50)

at org.apache.spark.sql.catalyst.csv.UnivocityParser.<init>(UnivocityParser.scala:103)

at org.apache.spark.sql.execution.datasources.csv.CSVFileFormat.$anonfun$buildReader$1(CSVFileFormat.scala:130)

at org.apache.spark.sql.execution.datasources.FileFormat$$anon$1.apply(FileFormat.scala:148)

at org.apache.spark.sql.execution.datasources.FileFormat$$anon$1.apply(FileFormat.scala:133)

at org.apache.spark.sql.execution.datasources.FileScanRDD$$anon$1.org$apache$spark$sql$execution$datasources$FileScanRDD$$anon$$readCurrentFile(FileScanRDD.scala:127)

at org.apache.spark.sql.execution.datasources.FileScanRDD$$anon$1.nextIterator(FileScanRDD.scala:187)

at org.apache.spark.sql.execution.datasources.FileScanRDD$$anon$1.hasNext(FileScanRDD.scala:104)

at scala.collection.Iterator$$anon$10.hasNext(Iterator.scala:460)

at org.apache.spark.sql.catalyst.expressions.GeneratedClass$GeneratedIteratorForCodegenStage1.processNext(Unknown Source)

at org.apache.spark.sql.execution.BufferedRowIterator.hasNext(BufferedRowIterator.java:43)

at org.apache.spark.sql.execution.WholeStageCodegenExec$$anon$1.hasNext(WholeStageCodegenExec.scala:759)

at org.apache.spark.sql.execution.SparkPlan.$anonfun$getByteArrayRdd$1(SparkPlan.scala:349)

at org.apache.spark.rdd.RDD.$anonfun$mapPartitionsInternal$2(RDD.scala:898)

at org.apache.spark.rdd.RDD.$anonfun$mapPartitionsInternal$2$adapted(RDD.scala:898)

at org.apache.spark.rdd.MapPartitionsRDD.compute(MapPartitionsRDD.scala:52)

at org.apache.spark.rdd.RDD.computeOrReadCheckpoint(RDD.scala:373)

at org.apache.spark.rdd.RDD.iterator(RDD.scala:337)

at org.apache.spark.scheduler.ResultTask.runTask(ResultTask.scala:90)

at org.apache.spark.scheduler.Task.run(Task.scala:131)

at org.apache.spark.executor.Executor$TaskRunner.$anonfun$run$3(Executor.scala:506)

at org.apache.spark.util.Utils$.tryWithSafeFinally(Utils.scala:1462)

at org.apache.spark.executor.Executor$TaskRunner.run(Executor.scala:509)

at java.util.concurrent.ThreadPoolExecutor.runWorker(ThreadPoolExecutor.java:1149)

at java.util.concurrent.ThreadPoolExecutor$Worker.run(ThreadPoolExecutor.java:624)

... 1 more

>>> spark.sql("select \* from table1 where RejectRows!=NULL ").show()

+--------+---------+------------+---------+----------+-----------+----------+-------+--------------+--------------+----------+

|IssuerId|IssuerId2|BusinessDate|StateCode|SourceName|NetworkName|NetworkURL|custnum|MarketCoverage|DentalOnlyPlan|RejectRows|

+--------+---------+------------+---------+----------+-----------+----------+-------+--------------+--------------+----------+

+--------+---------+------------+---------+----------+-----------+----------+-------+--------------+--------------+----------+

>>> data\_frame\_1.show(3)

+--------+---------+------------+---------+----------+------------+--------------------+-------+--------------+--------------+--------------------+

|IssuerId|IssuerId2|BusinessDate|StateCode|SourceName| NetworkName| NetworkURL|custnum|MarketCoverage|DentalOnlyPlan| RejectRows|

+--------+---------+------------+---------+----------+------------+--------------------+-------+--------------+--------------+--------------------+

| 21989| 21989| 2019-10-01| AK| HIOS| ODS Premier|https://www.modah...| 13| null| null| null|

| 219| 219| 2019-10-01| AK| HIOS| ODS Premier| null| null| null| null|219,219,2019-10-0...|

| 38344| 38344| 2019-10-01| AK| HIOS|HeritagePlus|https://www.preme...| 13| null| null| null|

+--------+---------+------------+---------+----------+------------+--------------------+-------+--------------+--------------+--------------------+

only showing top 3 rows

>>> spark.sql("select \* from table1 where NetworkName=='ODS Premier'")

DataFrame[IssuerId: int, IssuerId2: int, BusinessDate: date, StateCode: string, SourceName: string, NetworkName: string, NetworkURL: string, custnum: string, MarketCoverage: string, DentalOnlyPlan: string, RejectRows: string]

>>> spark.sql("select \* from table1 where NetworkName=='ODS Premier'").show()

+--------+---------+------------+---------+----------+-----------+--------------------+-------+--------------+--------------+--------------------+

|IssuerId|IssuerId2|BusinessDate|StateCode|SourceName|NetworkName| NetworkURL|custnum|MarketCoverage|DentalOnlyPlan| RejectRows|

+--------+---------+------------+---------+----------+-----------+--------------------+-------+--------------+--------------+--------------------+

| 21989| 21989| 2019-10-01| AK| HIOS|ODS Premier|https://www.modah...| 13| null| null| null|

| 219| 219| 2019-10-01| AK| HIOS|ODS Premier| null| null| null| null|219,219,2019-10-0...|

+--------+---------+------------+---------+----------+-----------+--------------------+-------+--------------+--------------+--------------------+

>>> spark.sql("select \* from table1 where NetworkName!='ODS Premier'").show()

+--------+---------+------------+---------+----------+--------------------+--------------------+-------+--------------+--------------+----------+

|IssuerId|IssuerId2|BusinessDate|StateCode|SourceName| NetworkName| NetworkURL|custnum|MarketCoverage|DentalOnlyPlan|RejectRows|

+--------+---------+------------+---------+----------+--------------------+--------------------+-------+--------------+--------------+----------+

| 38344| 38344| 2019-10-01| AK| HIOS| HeritagePlus|https://www.preme...| 13| null| null| null|

| 38536| 38536| 2019-10-01| AK| HIOS|Lincoln Dental Co...|http://lfg.go2den...| 13| null| null| null|

| 42507| 42507| 2019-10-01| AK| HIOS|DentalGuard Prefe...|https://www.guard...| 13| null| null| null|

| 73836| 73836| 2019-10-01| AK| HIOS|Moda Plus AK Regi...|https://www.modah...| 13| null| null| null|

| 73836| 73836| 2019-10-01| AK| HIOS|Moda Plus Providence|https://www.modah...| 14| null| null| null|

| 74819| 74819| 2019-10-01| AK| HIOS| DenteMax|http://www2.dente...| 13| null| null| null|

| 84859| 84859| 2019-10-01| AK| HIOS|Ameritas PPO Dent...| www.standard.com| 13| null| null| null|

| 12538| 12538| 2019-10-01| AL| HIOS| DenteMax|http://www2.dente...| 13| null| null| null|

| 12538| 12538| 2019-10-01| AL| HIOS|CONNECTION Dental...|http://www.ppousa...| 14| null| null| null|

| 17540| 17540| 2019-10-01| AL| HIOS|DentalGuard Prefe...|https://www.guard...| 13| null| null| null|

| 44580| 44580| 2019-10-01| AL| HIOS| Birmingham PPOx|http://pfp.humana...| 13| null| null| null|

| 44580| 44580| 2019-10-01| AL| HIOS| Huntsville PPOx|http://pfp.humana...| 14| null| null| null|

| 44580| 44580| 2019-10-01| AL| HIOS|Humana Dental PPO...|https://www.human...| 15| null| null| null|

| 44785| 44785| 2019-10-01| AL| HIOS| Advantage Plus|https://secure.uc...| 13| null| null| null|

| 46211| 46211| 2019-10-01| AL| HIOS|Lincoln Dental Co...|http://lfg.go2den...| 13| null| null| null|

| 46944| 46944| 2019-10-01| AL| HIOS| BCBSAL PPO|https://www.bcbsa...| 13| null| null| null|

| 59809| 59809| 2019-10-01| AL| HIOS| Choice Plus|http://www.provid...| 13| null| null| null|

| 59809| 59809| 2019-10-01| AL| HIOS| Choice Plus|http://www.provid...| 14| null| null| null|

| 77233| 77233| 2019-10-01| AL| HIOS| Dentegra Dental PPO|http://www.denteg...| 13| null| null| null|

| 82285| 82285| 2019-10-01| AL| HIOS| Delta Dental PPO|http://www.deltad...| 13| null| null| null|

+--------+---------+------------+---------+----------+--------------------+--------------------+-------+--------------+--------------+----------+

only showing top 20 rows

>>> spark.sql("select \* from table1 where NetworkName=='ODS Premier'").show()

+--------+---------+------------+---------+----------+-----------+--------------------+-------+--------------+--------------+--------------------+

|IssuerId|IssuerId2|BusinessDate|StateCode|SourceName|NetworkName| NetworkURL|custnum|MarketCoverage|DentalOnlyPlan| RejectRows|

+--------+---------+------------+---------+----------+-----------+--------------------+-------+--------------+--------------+--------------------+

| 21989| 21989| 2019-10-01| AK| HIOS|ODS Premier|https://www.modah...| 13| null| null| null|

| 219| 219| 2019-10-01| AK| HIOS|ODS Premier| null| null| null| null|219,219,2019-10-0...|

+--------+---------+------------+---------+----------+-----------+--------------------+-------+--------------+--------------+--------------------+

>>> spark.sql("select \* from table1 where RejectRows=='null'").show()

24/07/18 20:52:11 ERROR executor.Executor: Exception in task 0.0 in stage 30.0 (TID 30)

java.lang.IllegalArgumentException: RejectRows does not exist. Available: IssuerId, IssuerId2, BusinessDate, StateCode, SourceName, NetworkName, NetworkURL, custnum, MarketCoverage, DentalOnlyPlan

at org.apache.spark.sql.types.StructType.$anonfun$fieldIndex$1(StructType.scala:306)

at scala.collection.immutable.HashMap$HashTrieMap.getOrElse0(HashMap.scala:596)

at scala.collection.immutable.HashMap.getOrElse(HashMap.scala:73)

at org.apache.spark.sql.types.StructType.fieldIndex(StructType.scala:305)

at org.apache.spark.sql.catalyst.OrderedFilters.$anonfun$predicates$4(OrderedFilters.scala:61)

at org.apache.spark.sql.catalyst.OrderedFilters.$anonfun$predicates$4$adapted(OrderedFilters.scala:61)

at scala.collection.TraversableLike.$anonfun$map$1(TraversableLike.scala:286)

at scala.collection.IndexedSeqOptimized.foreach(IndexedSeqOptimized.scala:36)

at scala.collection.IndexedSeqOptimized.foreach$(IndexedSeqOptimized.scala:33)

at scala.collection.mutable.ArrayOps$ofRef.foreach(ArrayOps.scala:198)

at scala.collection.TraversableLike.map(TraversableLike.scala:286)

at scala.collection.TraversableLike.map$(TraversableLike.scala:279)

at scala.collection.mutable.ArrayOps$ofRef.map(ArrayOps.scala:198)

at org.apache.spark.sql.catalyst.OrderedFilters.$anonfun$predicates$3(OrderedFilters.scala:61)

at org.apache.spark.sql.catalyst.OrderedFilters.$anonfun$predicates$3$adapted(OrderedFilters.scala:50)

at scala.collection.immutable.List.foreach(List.scala:431)

at org.apache.spark.sql.catalyst.OrderedFilters.<init>(OrderedFilters.scala:50)

at org.apache.spark.sql.catalyst.csv.UnivocityParser.<init>(UnivocityParser.scala:103)

at org.apache.spark.sql.execution.datasources.csv.CSVFileFormat.$anonfun$buildReader$1(CSVFileFormat.scala:130)

at org.apache.spark.sql.execution.datasources.FileFormat$$anon$1.apply(FileFormat.scala:148)

at org.apache.spark.sql.execution.datasources.FileFormat$$anon$1.apply(FileFormat.scala:133)

at org.apache.spark.sql.execution.datasources.FileScanRDD$$anon$1.org$apache$spark$sql$execution$datasources$FileScanRDD$$anon$$readCurrentFile(FileScanRDD.scala:127)

at org.apache.spark.sql.execution.datasources.FileScanRDD$$anon$1.nextIterator(FileScanRDD.scala:187)

at org.apache.spark.sql.execution.datasources.FileScanRDD$$anon$1.hasNext(FileScanRDD.scala:104)

at scala.collection.Iterator$$anon$10.hasNext(Iterator.scala:460)

at org.apache.spark.sql.catalyst.expressions.GeneratedClass$GeneratedIteratorForCodegenStage1.processNext(Unknown Source)

at org.apache.spark.sql.execution.BufferedRowIterator.hasNext(BufferedRowIterator.java:43)

at org.apache.spark.sql.execution.WholeStageCodegenExec$$anon$1.hasNext(WholeStageCodegenExec.scala:759)

at org.apache.spark.sql.execution.SparkPlan.$anonfun$getByteArrayRdd$1(SparkPlan.scala:349)

at org.apache.spark.rdd.RDD.$anonfun$mapPartitionsInternal$2(RDD.scala:898)

at org.apache.spark.rdd.RDD.$anonfun$mapPartitionsInternal$2$adapted(RDD.scala:898)

at org.apache.spark.rdd.MapPartitionsRDD.compute(MapPartitionsRDD.scala:52)

at org.apache.spark.rdd.RDD.computeOrReadCheckpoint(RDD.scala:373)

at org.apache.spark.rdd.RDD.iterator(RDD.scala:337)

at org.apache.spark.scheduler.ResultTask.runTask(ResultTask.scala:90)

at org.apache.spark.scheduler.Task.run(Task.scala:131)

at org.apache.spark.executor.Executor$TaskRunner.$anonfun$run$3(Executor.scala:506)

at org.apache.spark.util.Utils$.tryWithSafeFinally(Utils.scala:1462)

at org.apache.spark.executor.Executor$TaskRunner.run(Executor.scala:509)

at java.util.concurrent.ThreadPoolExecutor.runWorker(ThreadPoolExecutor.java:1149)

at java.util.concurrent.ThreadPoolExecutor$Worker.run(ThreadPoolExecutor.java:624)

at java.lang.Thread.run(Thread.java:750)

24/07/18 20:52:11 WARN scheduler.TaskSetManager: Lost task 0.0 in stage 30.0 (TID 30) (192.168.112.129 executor driver): java.lang.IllegalArgumentException: RejectRows does not exist. Available: IssuerId, IssuerId2, BusinessDate, StateCode, SourceName, NetworkName, NetworkURL, custnum, MarketCoverage, DentalOnlyPlan

at org.apache.spark.sql.types.StructType.$anonfun$fieldIndex$1(StructType.scala:306)

at scala.collection.immutable.HashMap$HashTrieMap.getOrElse0(HashMap.scala:596)

at scala.collection.immutable.HashMap.getOrElse(HashMap.scala:73)

at org.apache.spark.sql.types.StructType.fieldIndex(StructType.scala:305)

at org.apache.spark.sql.catalyst.OrderedFilters.$anonfun$predicates$4(OrderedFilters.scala:61)

at org.apache.spark.sql.catalyst.OrderedFilters.$anonfun$predicates$4$adapted(OrderedFilters.scala:61)

at scala.collection.TraversableLike.$anonfun$map$1(TraversableLike.scala:286)

at scala.collection.IndexedSeqOptimized.foreach(IndexedSeqOptimized.scala:36)

at scala.collection.IndexedSeqOptimized.foreach$(IndexedSeqOptimized.scala:33)

at scala.collection.mutable.ArrayOps$ofRef.foreach(ArrayOps.scala:198)

at scala.collection.TraversableLike.map(TraversableLike.scala:286)

at scala.collection.TraversableLike.map$(TraversableLike.scala:279)

at scala.collection.mutable.ArrayOps$ofRef.map(ArrayOps.scala:198)

at org.apache.spark.sql.catalyst.OrderedFilters.$anonfun$predicates$3(OrderedFilters.scala:61)

at org.apache.spark.sql.catalyst.OrderedFilters.$anonfun$predicates$3$adapted(OrderedFilters.scala:50)

at scala.collection.immutable.List.foreach(List.scala:431)

at org.apache.spark.sql.catalyst.OrderedFilters.<init>(OrderedFilters.scala:50)

at org.apache.spark.sql.catalyst.csv.UnivocityParser.<init>(UnivocityParser.scala:103)

at org.apache.spark.sql.execution.datasources.csv.CSVFileFormat.$anonfun$buildReader$1(CSVFileFormat.scala:130)

at org.apache.spark.sql.execution.datasources.FileFormat$$anon$1.apply(FileFormat.scala:148)

at org.apache.spark.sql.execution.datasources.FileFormat$$anon$1.apply(FileFormat.scala:133)

at org.apache.spark.sql.execution.datasources.FileScanRDD$$anon$1.org$apache$spark$sql$execution$datasources$FileScanRDD$$anon$$readCurrentFile(FileScanRDD.scala:127)

at org.apache.spark.sql.execution.datasources.FileScanRDD$$anon$1.nextIterator(FileScanRDD.scala:187)

at org.apache.spark.sql.execution.datasources.FileScanRDD$$anon$1.hasNext(FileScanRDD.scala:104)

at scala.collection.Iterator$$anon$10.hasNext(Iterator.scala:460)

at org.apache.spark.sql.catalyst.expressions.GeneratedClass$GeneratedIteratorForCodegenStage1.processNext(Unknown Source)

at org.apache.spark.sql.execution.BufferedRowIterator.hasNext(BufferedRowIterator.java:43)

at org.apache.spark.sql.execution.WholeStageCodegenExec$$anon$1.hasNext(WholeStageCodegenExec.scala:759)

at org.apache.spark.sql.execution.SparkPlan.$anonfun$getByteArrayRdd$1(SparkPlan.scala:349)

at org.apache.spark.rdd.RDD.$anonfun$mapPartitionsInternal$2(RDD.scala:898)

at org.apache.spark.rdd.RDD.$anonfun$mapPartitionsInternal$2$adapted(RDD.scala:898)

at org.apache.spark.rdd.MapPartitionsRDD.compute(MapPartitionsRDD.scala:52)

at org.apache.spark.rdd.RDD.computeOrReadCheckpoint(RDD.scala:373)

at org.apache.spark.rdd.RDD.iterator(RDD.scala:337)

at org.apache.spark.scheduler.ResultTask.runTask(ResultTask.scala:90)

at org.apache.spark.scheduler.Task.run(Task.scala:131)

at org.apache.spark.executor.Executor$TaskRunner.$anonfun$run$3(Executor.scala:506)

at org.apache.spark.util.Utils$.tryWithSafeFinally(Utils.scala:1462)

at org.apache.spark.executor.Executor$TaskRunner.run(Executor.scala:509)

at java.util.concurrent.ThreadPoolExecutor.runWorker(ThreadPoolExecutor.java:1149)

at java.util.concurrent.ThreadPoolExecutor$Worker.run(ThreadPoolExecutor.java:624)

at java.lang.Thread.run(Thread.java:750)

24/07/18 20:52:11 ERROR scheduler.TaskSetManager: Task 0 in stage 30.0 failed 1 times; aborting job

Traceback (most recent call last):

File "<stdin>", line 1, in <module>

File "/usr/local/spark/python/pyspark/sql/dataframe.py", line 494, in show

print(self.\_jdf.showString(n, 20, vertical))

File "/usr/local/spark/python/lib/py4j-0.10.9.3-src.zip/py4j/java\_gateway.py", line 1322, in \_\_call\_\_

File "/usr/local/spark/python/pyspark/sql/utils.py", line 111, in deco

return f(\*a, \*\*kw)

File "/usr/local/spark/python/lib/py4j-0.10.9.3-src.zip/py4j/protocol.py", line 328, in get\_return\_value

py4j.protocol.Py4JJavaError: An error occurred while calling o289.showString.

: org.apache.spark.SparkException: Job aborted due to stage failure: Task 0 in stage 30.0 failed 1 times, most recent failure: Lost task 0.0 in stage 30.0 (TID 30) (192.168.112.129 executor driver): java.lang.IllegalArgumentException: RejectRows does not exist. Available: IssuerId, IssuerId2, BusinessDate, StateCode, SourceName, NetworkName, NetworkURL, custnum, MarketCoverage, DentalOnlyPlan

at org.apache.spark.sql.types.StructType.$anonfun$fieldIndex$1(StructType.scala:306)

at scala.collection.immutable.HashMap$HashTrieMap.getOrElse0(HashMap.scala:596)

at scala.collection.immutable.HashMap.getOrElse(HashMap.scala:73)

at org.apache.spark.sql.types.StructType.fieldIndex(StructType.scala:305)

at org.apache.spark.sql.catalyst.OrderedFilters.$anonfun$predicates$4(OrderedFilters.scala:61)

at org.apache.spark.sql.catalyst.OrderedFilters.$anonfun$predicates$4$adapted(OrderedFilters.scala:61)

at scala.collection.TraversableLike.$anonfun$map$1(TraversableLike.scala:286)

at scala.collection.IndexedSeqOptimized.foreach(IndexedSeqOptimized.scala:36)

at scala.collection.IndexedSeqOptimized.foreach$(IndexedSeqOptimized.scala:33)

at scala.collection.mutable.ArrayOps$ofRef.foreach(ArrayOps.scala:198)

at scala.collection.TraversableLike.map(TraversableLike.scala:286)

at scala.collection.TraversableLike.map$(TraversableLike.scala:279)

at scala.collection.mutable.ArrayOps$ofRef.map(ArrayOps.scala:198)

at org.apache.spark.sql.catalyst.OrderedFilters.$anonfun$predicates$3(OrderedFilters.scala:61)

at org.apache.spark.sql.catalyst.OrderedFilters.$anonfun$predicates$3$adapted(OrderedFilters.scala:50)

at scala.collection.immutable.List.foreach(List.scala:431)

at org.apache.spark.sql.catalyst.OrderedFilters.<init>(OrderedFilters.scala:50)

at org.apache.spark.sql.catalyst.csv.UnivocityParser.<init>(UnivocityParser.scala:103)

at org.apache.spark.sql.execution.datasources.csv.CSVFileFormat.$anonfun$buildReader$1(CSVFileFormat.scala:130)

at org.apache.spark.sql.execution.datasources.FileFormat$$anon$1.apply(FileFormat.scala:148)

at org.apache.spark.sql.execution.datasources.FileFormat$$anon$1.apply(FileFormat.scala:133)

at org.apache.spark.sql.execution.datasources.FileScanRDD$$anon$1.org$apache$spark$sql$execution$datasources$FileScanRDD$$anon$$readCurrentFile(FileScanRDD.scala:127)

at org.apache.spark.sql.execution.datasources.FileScanRDD$$anon$1.nextIterator(FileScanRDD.scala:187)

at org.apache.spark.sql.execution.datasources.FileScanRDD$$anon$1.hasNext(FileScanRDD.scala:104)

at scala.collection.Iterator$$anon$10.hasNext(Iterator.scala:460)

at org.apache.spark.sql.catalyst.expressions.GeneratedClass$GeneratedIteratorForCodegenStage1.processNext(Unknown Source)

at org.apache.spark.sql.execution.BufferedRowIterator.hasNext(BufferedRowIterator.java:43)

at org.apache.spark.sql.execution.WholeStageCodegenExec$$anon$1.hasNext(WholeStageCodegenExec.scala:759)

at org.apache.spark.sql.execution.SparkPlan.$anonfun$getByteArrayRdd$1(SparkPlan.scala:349)

at org.apache.spark.rdd.RDD.$anonfun$mapPartitionsInternal$2(RDD.scala:898)

at org.apache.spark.rdd.RDD.$anonfun$mapPartitionsInternal$2$adapted(RDD.scala:898)

at org.apache.spark.rdd.MapPartitionsRDD.compute(MapPartitionsRDD.scala:52)

at org.apache.spark.rdd.RDD.computeOrReadCheckpoint(RDD.scala:373)

at org.apache.spark.rdd.RDD.iterator(RDD.scala:337)

at org.apache.spark.scheduler.ResultTask.runTask(ResultTask.scala:90)

at org.apache.spark.scheduler.Task.run(Task.scala:131)

at org.apache.spark.executor.Executor$TaskRunner.$anonfun$run$3(Executor.scala:506)

at org.apache.spark.util.Utils$.tryWithSafeFinally(Utils.scala:1462)

at org.apache.spark.executor.Executor$TaskRunner.run(Executor.scala:509)

at java.util.concurrent.ThreadPoolExecutor.runWorker(ThreadPoolExecutor.java:1149)

at java.util.concurrent.ThreadPoolExecutor$Worker.run(ThreadPoolExecutor.java:624)

at java.lang.Thread.run(Thread.java:750)

Driver stacktrace:

at org.apache.spark.scheduler.DAGScheduler.failJobAndIndependentStages(DAGScheduler.scala:2454)

at org.apache.spark.scheduler.DAGScheduler.$anonfun$abortStage$2(DAGScheduler.scala:2403)

at org.apache.spark.scheduler.DAGScheduler.$anonfun$abortStage$2$adapted(DAGScheduler.scala:2402)

at scala.collection.mutable.ResizableArray.foreach(ResizableArray.scala:62)

at scala.collection.mutable.ResizableArray.foreach$(ResizableArray.scala:55)

at scala.collection.mutable.ArrayBuffer.foreach(ArrayBuffer.scala:49)

at org.apache.spark.scheduler.DAGScheduler.abortStage(DAGScheduler.scala:2402)

at org.apache.spark.scheduler.DAGScheduler.$anonfun$handleTaskSetFailed$1(DAGScheduler.scala:1160)

at org.apache.spark.scheduler.DAGScheduler.$anonfun$handleTaskSetFailed$1$adapted(DAGScheduler.scala:1160)

at scala.Option.foreach(Option.scala:407)

at org.apache.spark.scheduler.DAGScheduler.handleTaskSetFailed(DAGScheduler.scala:1160)

at org.apache.spark.scheduler.DAGSchedulerEventProcessLoop.doOnReceive(DAGScheduler.scala:2642)

at org.apache.spark.scheduler.DAGSchedulerEventProcessLoop.onReceive(DAGScheduler.scala:2584)

at org.apache.spark.scheduler.DAGSchedulerEventProcessLoop.onReceive(DAGScheduler.scala:2573)

at org.apache.spark.util.EventLoop$$anon$1.run(EventLoop.scala:49)

at org.apache.spark.scheduler.DAGScheduler.runJob(DAGScheduler.scala:938)

at org.apache.spark.SparkContext.runJob(SparkContext.scala:2214)

at org.apache.spark.SparkContext.runJob(SparkContext.scala:2235)

at org.apache.spark.SparkContext.runJob(SparkContext.scala:2254)

at org.apache.spark.sql.execution.SparkPlan.executeTake(SparkPlan.scala:476)

at org.apache.spark.sql.execution.SparkPlan.executeTake(SparkPlan.scala:429)

at org.apache.spark.sql.execution.CollectLimitExec.executeCollect(limit.scala:48)

at org.apache.spark.sql.Dataset.collectFromPlan(Dataset.scala:3715)

at org.apache.spark.sql.Dataset.$anonfun$head$1(Dataset.scala:2728)

at org.apache.spark.sql.Dataset.$anonfun$withAction$1(Dataset.scala:3706)

at org.apache.spark.sql.execution.SQLExecution$.$anonfun$withNewExecutionId$5(SQLExecution.scala:103)

at org.apache.spark.sql.execution.SQLExecution$.withSQLConfPropagated(SQLExecution.scala:163)

at org.apache.spark.sql.execution.SQLExecution$.$anonfun$withNewExecutionId$1(SQLExecution.scala:90)

at org.apache.spark.sql.SparkSession.withActive(SparkSession.scala:775)

at org.apache.spark.sql.execution.SQLExecution$.withNewExecutionId(SQLExecution.scala:64)

at org.apache.spark.sql.Dataset.withAction(Dataset.scala:3704)

at org.apache.spark.sql.Dataset.head(Dataset.scala:2728)

at org.apache.spark.sql.Dataset.take(Dataset.scala:2935)

at org.apache.spark.sql.Dataset.getRows(Dataset.scala:287)

at org.apache.spark.sql.Dataset.showString(Dataset.scala:326)

at sun.reflect.GeneratedMethodAccessor41.invoke(Unknown Source)

at sun.reflect.DelegatingMethodAccessorImpl.invoke(DelegatingMethodAccessorImpl.java:43)

at java.lang.reflect.Method.invoke(Method.java:498)

at py4j.reflection.MethodInvoker.invoke(MethodInvoker.java:244)

at py4j.reflection.ReflectionEngine.invoke(ReflectionEngine.java:357)

at py4j.Gateway.invoke(Gateway.java:282)

at py4j.commands.AbstractCommand.invokeMethod(AbstractCommand.java:132)

at py4j.commands.CallCommand.execute(CallCommand.java:79)

at py4j.ClientServerConnection.waitForCommands(ClientServerConnection.java:182)

at py4j.ClientServerConnection.run(ClientServerConnection.java:106)

at java.lang.Thread.run(Thread.java:750)

Caused by: java.lang.IllegalArgumentException: RejectRows does not exist. Available: IssuerId, IssuerId2, BusinessDate, StateCode, SourceName, NetworkName, NetworkURL, custnum, MarketCoverage, DentalOnlyPlan

at org.apache.spark.sql.types.StructType.$anonfun$fieldIndex$1(StructType.scala:306)

at scala.collection.immutable.HashMap$HashTrieMap.getOrElse0(HashMap.scala:596)

at scala.collection.immutable.HashMap.getOrElse(HashMap.scala:73)

at org.apache.spark.sql.types.StructType.fieldIndex(StructType.scala:305)

at org.apache.spark.sql.catalyst.OrderedFilters.$anonfun$predicates$4(OrderedFilters.scala:61)

at org.apache.spark.sql.catalyst.OrderedFilters.$anonfun$predicates$4$adapted(OrderedFilters.scala:61)

at scala.collection.TraversableLike.$anonfun$map$1(TraversableLike.scala:286)

at scala.collection.IndexedSeqOptimized.foreach(IndexedSeqOptimized.scala:36)

at scala.collection.IndexedSeqOptimized.foreach$(IndexedSeqOptimized.scala:33)

at scala.collection.mutable.ArrayOps$ofRef.foreach(ArrayOps.scala:198)

at scala.collection.TraversableLike.map(TraversableLike.scala:286)

at scala.collection.TraversableLike.map$(TraversableLike.scala:279)

at scala.collection.mutable.ArrayOps$ofRef.map(ArrayOps.scala:198)

at org.apache.spark.sql.catalyst.OrderedFilters.$anonfun$predicates$3(OrderedFilters.scala:61)

at org.apache.spark.sql.catalyst.OrderedFilters.$anonfun$predicates$3$adapted(OrderedFilters.scala:50)

at scala.collection.immutable.List.foreach(List.scala:431)

at org.apache.spark.sql.catalyst.OrderedFilters.<init>(OrderedFilters.scala:50)

at org.apache.spark.sql.catalyst.csv.UnivocityParser.<init>(UnivocityParser.scala:103)

at org.apache.spark.sql.execution.datasources.csv.CSVFileFormat.$anonfun$buildReader$1(CSVFileFormat.scala:130)

at org.apache.spark.sql.execution.datasources.FileFormat$$anon$1.apply(FileFormat.scala:148)

at org.apache.spark.sql.execution.datasources.FileFormat$$anon$1.apply(FileFormat.scala:133)

at org.apache.spark.sql.execution.datasources.FileScanRDD$$anon$1.org$apache$spark$sql$execution$datasources$FileScanRDD$$anon$$readCurrentFile(FileScanRDD.scala:127)

at org.apache.spark.sql.execution.datasources.FileScanRDD$$anon$1.nextIterator(FileScanRDD.scala:187)

at org.apache.spark.sql.execution.datasources.FileScanRDD$$anon$1.hasNext(FileScanRDD.scala:104)

at scala.collection.Iterator$$anon$10.hasNext(Iterator.scala:460)

at org.apache.spark.sql.catalyst.expressions.GeneratedClass$GeneratedIteratorForCodegenStage1.processNext(Unknown Source)

at org.apache.spark.sql.execution.BufferedRowIterator.hasNext(BufferedRowIterator.java:43)

at org.apache.spark.sql.execution.WholeStageCodegenExec$$anon$1.hasNext(WholeStageCodegenExec.scala:759)

at org.apache.spark.sql.execution.SparkPlan.$anonfun$getByteArrayRdd$1(SparkPlan.scala:349)

at org.apache.spark.rdd.RDD.$anonfun$mapPartitionsInternal$2(RDD.scala:898)

at org.apache.spark.rdd.RDD.$anonfun$mapPartitionsInternal$2$adapted(RDD.scala:898)

at org.apache.spark.rdd.MapPartitionsRDD.compute(MapPartitionsRDD.scala:52)

at org.apache.spark.rdd.RDD.computeOrReadCheckpoint(RDD.scala:373)

at org.apache.spark.rdd.RDD.iterator(RDD.scala:337)

at org.apache.spark.scheduler.ResultTask.runTask(ResultTask.scala:90)

at org.apache.spark.scheduler.Task.run(Task.scala:131)

at org.apache.spark.executor.Executor$TaskRunner.$anonfun$run$3(Executor.scala:506)

at org.apache.spark.util.Utils$.tryWithSafeFinally(Utils.scala:1462)

at org.apache.spark.executor.Executor$TaskRunner.run(Executor.scala:509)

at java.util.concurrent.ThreadPoolExecutor.runWorker(ThreadPoolExecutor.java:1149)

at java.util.concurrent.ThreadPoolExecutor$Worker.run(ThreadPoolExecutor.java:624)

... 1 more

>>>

>>> spark.sql("select \* from table1 where RejectRows==null").show()

+--------+---------+------------+---------+----------+-----------+----------+-------+--------------+--------------+----------+

|IssuerId|IssuerId2|BusinessDate|StateCode|SourceName|NetworkName|NetworkURL|custnum|MarketCoverage|DentalOnlyPlan|RejectRows|

+--------+---------+------------+---------+----------+-----------+----------+-------+--------------+--------------+----------+

+--------+---------+------------+---------+----------+-----------+----------+-------+--------------+--------------+----------+

>>> spark.sql("select \* from table1 where NetworkName==ODS Premier").show()

Traceback (most recent call last):

File "<stdin>", line 1, in <module>

File "/usr/local/spark/python/pyspark/sql/session.py", line 723, in sql

return DataFrame(self.\_jsparkSession.sql(sqlQuery), self.\_wrapped)

File "/usr/local/spark/python/lib/py4j-0.10.9.3-src.zip/py4j/java\_gateway.py", line 1322, in \_\_call\_\_

File "/usr/local/spark/python/pyspark/sql/utils.py", line 117, in deco

raise converted from None

pyspark.sql.utils.ParseException:

extraneous input 'Premier' expecting {<EOF>, ';'}(line 1, pos 44)

== SQL ==

select \* from table1 where NetworkName==ODS Premier

--------------------------------------------^^^

>>> spark.sql("select \* from table1 where NetworkName=='ODS Premier'").show()

+--------+---------+------------+---------+----------+-----------+--------------------+-------+--------------+--------------+--------------------+

|IssuerId|IssuerId2|BusinessDate|StateCode|SourceName|NetworkName| NetworkURL|custnum|MarketCoverage|DentalOnlyPlan| RejectRows|

+--------+---------+------------+---------+----------+-----------+--------------------+-------+--------------+--------------+--------------------+

| 21989| 21989| 2019-10-01| AK| HIOS|ODS Premier|https://www.modah...| 13| null| null| null|

| 219| 219| 2019-10-01| AK| HIOS|ODS Premier| null| null| null| null|219,219,2019-10-0...|

+--------+---------+------------+---------+----------+-----------+--------------------+-------+--------------+--------------+--------------------+

>>> spark.sql("select \* from table1 where NetworkName=='ODS Premier' ").show()

+--------+---------+------------+---------+----------+-----------+--------------------+-------+--------------+--------------+--------------------+

|IssuerId|IssuerId2|BusinessDate|StateCode|SourceName|NetworkName| NetworkURL|custnum|MarketCoverage|DentalOnlyPlan| RejectRows|

+--------+---------+------------+---------+----------+-----------+--------------------+-------+--------------+--------------+--------------------+

| 21989| 21989| 2019-10-01| AK| HIOS|ODS Premier|https://www.modah...| 13| null| null| null|

| 219| 219| 2019-10-01| AK| HIOS|ODS Premier| null| null| null| null|219,219,2019-10-0...|

+--------+---------+------------+---------+----------+-----------+--------------------+-------+--------------+--------------+--------------------+

>>> spark.sql("select \* from table1 where NetworkName!='null' ").show()

+--------+---------+------------+---------+----------+--------------------+--------------------+-------+--------------+--------------+--------------------+

|IssuerId|IssuerId2|BusinessDate|StateCode|SourceName| NetworkName| NetworkURL|custnum|MarketCoverage|DentalOnlyPlan| RejectRows|

+--------+---------+------------+---------+----------+--------------------+--------------------+-------+--------------+--------------+--------------------+

| 21989| 21989| 2019-10-01| AK| HIOS| ODS Premier|https://www.modah...| 13| null| null| null|

| 219| 219| 2019-10-01| AK| HIOS| ODS Premier| null| null| null| null|219,219,2019-10-0...|

| 38344| 38344| 2019-10-01| AK| HIOS| HeritagePlus|https://www.preme...| 13| null| null| null|

| 38536| 38536| 2019-10-01| AK| HIOS|Lincoln Dental Co...|http://lfg.go2den...| 13| null| null| null|

| 42507| 42507| 2019-10-01| AK| HIOS|DentalGuard Prefe...|https://www.guard...| 13| null| null| null|

| 73836| 73836| 2019-10-01| AK| HIOS|Moda Plus AK Regi...|https://www.modah...| 13| null| null| null|

| 73836| 73836| 2019-10-01| AK| HIOS|Moda Plus Providence|https://www.modah...| 14| null| null| null|

| 74819| 74819| 2019-10-01| AK| HIOS| DenteMax|http://www2.dente...| 13| null| null| null|

| 84859| 84859| 2019-10-01| AK| HIOS|Ameritas PPO Dent...| www.standard.com| 13| null| null| null|

| 12538| 12538| 2019-10-01| AL| HIOS| DenteMax|http://www2.dente...| 13| null| null| null|

| 12538| 12538| 2019-10-01| AL| HIOS|CONNECTION Dental...|http://www.ppousa...| 14| null| null| null|

| 17540| 17540| 2019-10-01| AL| HIOS|DentalGuard Prefe...|https://www.guard...| 13| null| null| null|

| 44580| 44580| 2019-10-01| AL| HIOS| Birmingham PPOx|http://pfp.humana...| 13| null| null| null|

| 44580| 44580| 2019-10-01| AL| HIOS| Huntsville PPOx|http://pfp.humana...| 14| null| null| null|

| 44580| 44580| 2019-10-01| AL| HIOS|Humana Dental PPO...|https://www.human...| 15| null| null| null|

| 44785| 44785| 2019-10-01| AL| HIOS| Advantage Plus|https://secure.uc...| 13| null| null| null|

| 46211| 46211| 2019-10-01| AL| HIOS|Lincoln Dental Co...|http://lfg.go2den...| 13| null| null| null|

| 46944| 46944| 2019-10-01| AL| HIOS| BCBSAL PPO|https://www.bcbsa...| 13| null| null| null|

| 59809| 59809| 2019-10-01| AL| HIOS| Choice Plus|http://www.provid...| 13| null| null| null|

| 59809| 59809| 2019-10-01| AL| HIOS| Choice Plus|http://www.provid...| 14| null| null| null|

+--------+---------+------------+---------+----------+--------------------+--------------------+-------+--------------+--------------+--------------------+

only showing top 20 rows

>>> spark.sql("select \* from table1 where NetworkURL!='null' ").show()

+--------+---------+------------+---------+----------+--------------------+--------------------+-------+--------------+--------------+----------+

|IssuerId|IssuerId2|BusinessDate|StateCode|SourceName| NetworkName| NetworkURL|custnum|MarketCoverage|DentalOnlyPlan|RejectRows|

+--------+---------+------------+---------+----------+--------------------+--------------------+-------+--------------+--------------+----------+

| 21989| 21989| 2019-10-01| AK| HIOS| ODS Premier|https://www.modah...| 13| null| null| null|

| 38344| 38344| 2019-10-01| AK| HIOS| HeritagePlus|https://www.preme...| 13| null| null| null|

| 38536| 38536| 2019-10-01| AK| HIOS|Lincoln Dental Co...|http://lfg.go2den...| 13| null| null| null|

| 42507| 42507| 2019-10-01| AK| HIOS|DentalGuard Prefe...|https://www.guard...| 13| null| null| null|

| 73836| 73836| 2019-10-01| AK| HIOS|Moda Plus AK Regi...|https://www.modah...| 13| null| null| null|

| 73836| 73836| 2019-10-01| AK| HIOS|Moda Plus Providence|https://www.modah...| 14| null| null| null|

| 74819| 74819| 2019-10-01| AK| HIOS| DenteMax|http://www2.dente...| 13| null| null| null|

| 84859| 84859| 2019-10-01| AK| HIOS|Ameritas PPO Dent...| www.standard.com| 13| null| null| null|

| 12538| 12538| 2019-10-01| AL| HIOS| DenteMax|http://www2.dente...| 13| null| null| null|

| 12538| 12538| 2019-10-01| AL| HIOS|CONNECTION Dental...|http://www.ppousa...| 14| null| null| null|

| 17540| 17540| 2019-10-01| AL| HIOS|DentalGuard Prefe...|https://www.guard...| 13| null| null| null|

| 44580| 44580| 2019-10-01| AL| HIOS| Birmingham PPOx|http://pfp.humana...| 13| null| null| null|

| 44580| 44580| 2019-10-01| AL| HIOS| Huntsville PPOx|http://pfp.humana...| 14| null| null| null|

| 44580| 44580| 2019-10-01| AL| HIOS|Humana Dental PPO...|https://www.human...| 15| null| null| null|

| 44785| 44785| 2019-10-01| AL| HIOS| Advantage Plus|https://secure.uc...| 13| null| null| null|

| 46211| 46211| 2019-10-01| AL| HIOS|Lincoln Dental Co...|http://lfg.go2den...| 13| null| null| null|

| 46944| 46944| 2019-10-01| AL| HIOS| BCBSAL PPO|https://www.bcbsa...| 13| null| null| null|

| 59809| 59809| 2019-10-01| AL| HIOS| Choice Plus|http://www.provid...| 13| null| null| null|

| 59809| 59809| 2019-10-01| AL| HIOS| Choice Plus|http://www.provid...| 14| null| null| null|

| 77233| 77233| 2019-10-01| AL| HIOS| Dentegra Dental PPO|http://www.denteg...| 13| null| null| null|

+--------+---------+------------+---------+----------+--------------------+--------------------+-------+--------------+--------------+----------+

only showing top 20 rows

>>> spark.sql("select \* from table1 where NetworkURL=='null' ").show()

+--------+---------+------------+---------+----------+-----------+----------+-------+--------------+--------------+----------+

|IssuerId|IssuerId2|BusinessDate|StateCode|SourceName|NetworkName|NetworkURL|custnum|MarketCoverage|DentalOnlyPlan|RejectRows|

+--------+---------+------------+---------+----------+-----------+----------+-------+--------------+--------------+----------+

+--------+---------+------------+---------+----------+-----------+----------+-------+--------------+--------------+----------+

>>> spark.sql("select \* from table1 where NetworkURL.isEmpty() ").show()

24/07/18 20:54:10 WARN conf.HiveConf: HiveConf of name hive.stats.jdbc.timeout does not exist

24/07/18 20:54:10 WARN conf.HiveConf: HiveConf of name hive.stats.retries.wait does not exist

24/07/18 20:54:11 WARN metastore.ObjectStore: Failed to get database networkurl, returning NoSuchObjectException

Traceback (most recent call last):

File "<stdin>", line 1, in <module>

File "/usr/local/spark/python/pyspark/sql/session.py", line 723, in sql

return DataFrame(self.\_jsparkSession.sql(sqlQuery), self.\_wrapped)

File "/usr/local/spark/python/lib/py4j-0.10.9.3-src.zip/py4j/java\_gateway.py", line 1322, in \_\_call\_\_

File "/usr/local/spark/python/pyspark/sql/utils.py", line 117, in deco

raise converted from None

pyspark.sql.utils.AnalysisException: Undefined function: 'isEmpty'. This function is neither a registered temporary function nor a permanent function registered in the database 'NetworkURL'.; line 1 pos 27

>>> data\_frame\_1.where("RejectRows.isEmpty()").show(3)

24/07/18 20:54:40 WARN metastore.ObjectStore: Failed to get database rejectrows, returning NoSuchObjectException

Traceback (most recent call last):

File "<stdin>", line 1, in <module>

File "/usr/local/spark/python/pyspark/sql/dataframe.py", line 1731, in filter

jdf = self.\_jdf.filter(condition)

File "/usr/local/spark/python/lib/py4j-0.10.9.3-src.zip/py4j/java\_gateway.py", line 1322, in \_\_call\_\_

File "/usr/local/spark/python/pyspark/sql/utils.py", line 117, in deco

raise converted from None

pyspark.sql.utils.AnalysisException: Undefined function: 'isEmpty'. This function is neither a registered temporary function nor a permanent function registered in the database 'RejectRows'.; line 1 pos 0

>>> data\_frame\_1.where("RejectRows.isNotNull)").show(3)

Traceback (most recent call last):

File "<stdin>", line 1, in <module>

File "/usr/local/spark/python/pyspark/sql/dataframe.py", line 1731, in filter

jdf = self.\_jdf.filter(condition)

File "/usr/local/spark/python/lib/py4j-0.10.9.3-src.zip/py4j/java\_gateway.py", line 1322, in \_\_call\_\_

File "/usr/local/spark/python/pyspark/sql/utils.py", line 117, in deco

raise converted from None

pyspark.sql.utils.ParseException:

extraneous input ')' expecting <EOF>(line 1, pos 20)

== SQL ==

RejectRows.isNotNull)

--------------------^^^

>>> data\_frame\_1.where("RejectRows.isNotNull()").show(3)

24/07/18 20:55:13 WARN metastore.ObjectStore: Failed to get database rejectrows, returning NoSuchObjectException

Traceback (most recent call last):

File "<stdin>", line 1, in <module>

File "/usr/local/spark/python/pyspark/sql/dataframe.py", line 1731, in filter

jdf = self.\_jdf.filter(condition)

File "/usr/local/spark/python/lib/py4j-0.10.9.3-src.zip/py4j/java\_gateway.py", line 1322, in \_\_call\_\_

File "/usr/local/spark/python/pyspark/sql/utils.py", line 117, in deco

raise converted from None

pyspark.sql.utils.AnalysisException: Undefined function: 'isNotNull'. This function is neither a registered temporary function nor a permanent function registered in the database 'RejectRows'.; line 1 pos 0

>>> data\_frame\_1.where("RejectRows.isEmpty()").show(3)

24/07/18 20:56:16 WARN metastore.ObjectStore: Failed to get database rejectrows, returning NoSuchObjectException

Traceback (most recent call last):

File "<stdin>", line 1, in <module>

File "/usr/local/spark/python/pyspark/sql/dataframe.py", line 1731, in filter

jdf = self.\_jdf.filter(condition)

File "/usr/local/spark/python/lib/py4j-0.10.9.3-src.zip/py4j/java\_gateway.py", line 1322, in \_\_call\_\_

File "/usr/local/spark/python/pyspark/sql/utils.py", line 117, in deco

raise converted from None

pyspark.sql.utils.AnalysisException: Undefined function: 'isEmpty'. This function is neither a registered temporary function nor a permanent function registered in the database 'RejectRows'.; line 1 pos 0

>>> data\_frame\_1.where(data\_frame\_1.RejectRows.isEmpty()).show(3)

Traceback (most recent call last):

File "<stdin>", line 1, in <module>

TypeError: 'Column' object is not callable

>>> data\_frame\_1.where(data\_frame\_1.RejectRows.isEmpty()).show(3)

Traceback (most recent call last):

File "<stdin>", line 1, in <module>

TypeError: 'Column' object is not callable

>>> data\_frame\_1.where(data\_frame\_1.RejectRows.isEmpty()).show(3)

>>> data\_frame\_1.show()

+--------+---------+------------+---------+----------+--------------------+--------------------+-------+--------------+--------------+--------------------+

|IssuerId|IssuerId2|BusinessDate|StateCode|SourceName| NetworkName| NetworkURL|custnum|MarketCoverage|DentalOnlyPlan| RejectRows|

+--------+---------+------------+---------+----------+--------------------+--------------------+-------+--------------+--------------+--------------------+

| 21989| 21989| 2019-10-01| AK| HIOS| ODS Premier|https://www.modah...| 13| null| null| null|

| 219| 219| 2019-10-01| AK| HIOS| ODS Premier| null| null| null| null|219,219,2019-10-0...|

| 38344| 38344| 2019-10-01| AK| HIOS| HeritagePlus|https://www.preme...| 13| null| null| null|

| 38536| 38536| 2019-10-01| AK| HIOS|Lincoln Dental Co...|http://lfg.go2den...| 13| null| null| null|

| 42507| 42507| 2019-10-01| AK| HIOS|DentalGuard Prefe...|https://www.guard...| 13| null| null| null|

| 73836| 73836| 2019-10-01| AK| HIOS|Moda Plus AK Regi...|https://www.modah...| 13| null| null| null|

| 73836| 73836| 2019-10-01| AK| HIOS|Moda Plus Providence|https://www.modah...| 14| null| null| null|

| 74819| 74819| 2019-10-01| AK| HIOS| DenteMax|http://www2.dente...| 13| null| null| null|

| 84859| 84859| 2019-10-01| AK| HIOS|Ameritas PPO Dent...| www.standard.com| 13| null| null| null|

| 12538| 12538| 2019-10-01| AL| HIOS| DenteMax|http://www2.dente...| 13| null| null| null|

| 12538| 12538| 2019-10-01| AL| HIOS|CONNECTION Dental...|http://www.ppousa...| 14| null| null| null|

| 17540| 17540| 2019-10-01| AL| HIOS|DentalGuard Prefe...|https://www.guard...| 13| null| null| null|

| 44580| 44580| 2019-10-01| AL| HIOS| Birmingham PPOx|http://pfp.humana...| 13| null| null| null|

| 44580| 44580| 2019-10-01| AL| HIOS| Huntsville PPOx|http://pfp.humana...| 14| null| null| null|

| 44580| 44580| 2019-10-01| AL| HIOS|Humana Dental PPO...|https://www.human...| 15| null| null| null|

| 44785| 44785| 2019-10-01| AL| HIOS| Advantage Plus|https://secure.uc...| 13| null| null| null|

| 46211| 46211| 2019-10-01| AL| HIOS|Lincoln Dental Co...|http://lfg.go2den...| 13| null| null| null|

| 46944| 46944| 2019-10-01| AL| HIOS| BCBSAL PPO|https://www.bcbsa...| 13| null| null| null|

| 59809| 59809| 2019-10-01| AL| HIOS| Choice Plus|http://www.provid...| 13| null| null| null|

| 59809| 59809| 2019-10-01| AL| HIOS| Choice Plus|http://www.provid...| 14| null| null| null|

+--------+---------+------------+---------+----------+--------------------+--------------------+-------+--------------+--------------+--------------------+

only showing top 20 rows

>>> data\_frame\_1.where("custnum=='13'").show(34)

+--------+---------+------------+---------+----------+--------------------+--------------------+-------+--------------+--------------+----------+

|IssuerId|IssuerId2|BusinessDate|StateCode|SourceName| NetworkName| NetworkURL|custnum|MarketCoverage|DentalOnlyPlan|RejectRows|

+--------+---------+------------+---------+----------+--------------------+--------------------+-------+--------------+--------------+----------+

| 21989| 21989| 2019-10-01| AK| HIOS| ODS Premier|https://www.modah...| 13| null| null| null|

| 38344| 38344| 2019-10-01| AK| HIOS| HeritagePlus|https://www.preme...| 13| null| null| null|

| 38536| 38536| 2019-10-01| AK| HIOS|Lincoln Dental Co...|http://lfg.go2den...| 13| null| null| null|

| 42507| 42507| 2019-10-01| AK| HIOS|DentalGuard Prefe...|https://www.guard...| 13| null| null| null|

| 73836| 73836| 2019-10-01| AK| HIOS|Moda Plus AK Regi...|https://www.modah...| 13| null| null| null|

| 74819| 74819| 2019-10-01| AK| HIOS| DenteMax|http://www2.dente...| 13| null| null| null|

| 84859| 84859| 2019-10-01| AK| HIOS|Ameritas PPO Dent...| www.standard.com| 13| null| null| null|

| 12538| 12538| 2019-10-01| AL| HIOS| DenteMax|http://www2.dente...| 13| null| null| null|

| 17540| 17540| 2019-10-01| AL| HIOS|DentalGuard Prefe...|https://www.guard...| 13| null| null| null|

| 44580| 44580| 2019-10-01| AL| HIOS| Birmingham PPOx|http://pfp.humana...| 13| null| null| null|

| 44785| 44785| 2019-10-01| AL| HIOS| Advantage Plus|https://secure.uc...| 13| null| null| null|

| 46211| 46211| 2019-10-01| AL| HIOS|Lincoln Dental Co...|http://lfg.go2den...| 13| null| null| null|

| 46944| 46944| 2019-10-01| AL| HIOS| BCBSAL PPO|https://www.bcbsa...| 13| null| null| null|

| 59809| 59809| 2019-10-01| AL| HIOS| Choice Plus|http://www.provid...| 13| null| null| null|

| 77233| 77233| 2019-10-01| AL| HIOS| Dentegra Dental PPO|http://www.denteg...| 13| null| null| null|

| 82285| 82285| 2019-10-01| AL| HIOS| Delta Dental PPO|http://www.deltad...| 13| null| null| null|

| 17100| 17100| 2019-10-01| AZ| HIOS|EHB Basic Dental ...|https://metlocato...| 13| null| null| null|

| 18156| 18156| 2019-10-01| AZ| HIOS|Lincoln Dental Co...|http://lfg.go2den...| 13| null| null| null|

| 23307| 23307| 2019-10-01| AZ| HIOS| Phoenix HMOx|http://pfp.humana...| 13| null| null| null|

| 30045| 30045| 2019-10-01| AZ| HIOS| Delta Dental PPO|http://www.deltad...| 13| null| null| null|

| 33851| 33851| 2019-10-01| AZ| HIOS|Premier Access De...|www.premierlife.c...| 13| null| null| null|

| 51485| 51485| 2019-10-01| AZ| HIOS| AZ PPO|https://www.healt...| 13| null| null| null|

| 52147| 52147| 2019-10-01| AZ| HIOS|Ameritas PPO Dent...| www.standard.com| 13| null| null| null|

| 53901| 53901| 2019-10-01| AZ| HIOS| Statewide PPO| www.azblue.com/PPO| 13| null| null| null|

| 58292| 58292| 2019-10-01| AZ| HIOS| Dentegra Dental PPO|http://www.denteg...| 13| null| null| null|

| 60761| 60761| 2019-10-01| AZ| HIOS| Meritus HMO Network|http://www.meritu...| 13| null| null| null|

| 66105| 66105| 2019-10-01| AZ| HIOS|Humana Dental PPO...|https://www.human...| 13| null| null| null|

| 70239| 70239| 2019-10-01| AZ| HIOS|Health Choice Ess...|http://www.health...| 13| null| null| null|

| 77352| 77352| 2019-10-01| AZ| HIOS| DenteMax|http://www2.dente...| 13| null| null| null|

| 84251| 84251| 2019-10-01| AZ| HIOS|QHP PPO\_OAMC\_Arizona|http://www.aetna....| 13| null| null| null|

| 86830| 86830| 2019-10-01| AZ| HIOS| Arizona Local Plus|www.Cigna.com/ifp...| 13| null| null| null|

| 88925| 88925| 2019-10-01| AZ| HIOS| UHM Primary|http://uamarketpl...| 13| null| null| null|

| 90264| 90264| 2019-10-01| AZ| HIOS|Ameritas PPO Dent...|www.ameritasgroup...| 13| null| null| null|

| 91450| 91450| 2019-10-01| AZ| HIOS|AZ HMO CommunityC...|https://www.healt...| 13| null| null| null|

+--------+---------+------------+---------+----------+--------------------+--------------------+-------+--------------+--------------+----------+

only showing top 34 rows

>>> data\_frame\_1.where("custnum=='13'").show(3)

+--------+---------+------------+---------+----------+--------------------+--------------------+-------+--------------+--------------+----------+

|IssuerId|IssuerId2|BusinessDate|StateCode|SourceName| NetworkName| NetworkURL|custnum|MarketCoverage|DentalOnlyPlan|RejectRows|

+--------+---------+------------+---------+----------+--------------------+--------------------+-------+--------------+--------------+----------+

| 21989| 21989| 2019-10-01| AK| HIOS| ODS Premier|https://www.modah...| 13| null| null| null|

| 38344| 38344| 2019-10-01| AK| HIOS| HeritagePlus|https://www.preme...| 13| null| null| null|

| 38536| 38536| 2019-10-01| AK| HIOS|Lincoln Dental Co...|http://lfg.go2den...| 13| null| null| null|

+--------+---------+------------+---------+----------+--------------------+--------------------+-------+--------------+--------------+----------+

only showing top 3 rows

>>> data\_frame\_1.where("custnum=='null'").show(3)

+--------+---------+------------+---------+----------+-----------+----------+-------+--------------+--------------+----------+

|IssuerId|IssuerId2|BusinessDate|StateCode|SourceName|NetworkName|NetworkURL|custnum|MarketCoverage|DentalOnlyPlan|RejectRows|

+--------+---------+------------+---------+----------+-----------+----------+-------+--------------+--------------+----------+

+--------+---------+------------+---------+----------+-----------+----------+-------+--------------+--------------+----------+

>>> from pyspark.sql.functions import col

>>> data\_frame\_1.where(col("custnum").isNull()).show(3)

+--------+---------+------------+---------+----------+-----------+----------+-------+--------------+--------------+--------------------+

|IssuerId|IssuerId2|BusinessDate|StateCode|SourceName|NetworkName|NetworkURL|custnum|MarketCoverage|DentalOnlyPlan| RejectRows|

+--------+---------+------------+---------+----------+-----------+----------+-------+--------------+--------------+--------------------+

| 219| 219| 2019-10-01| AK| HIOS|ODS Premier| null| null| null| null|219,219,2019-10-0...|

| null| null| null| null| null| null| null| null| null| null| footer count is 402|

+--------+---------+------------+---------+----------+-----------+----------+-------+--------------+--------------+--------------------+

>>> data\_frame\_1.where(col("RejectRows").isNotNull()).show(3)

24/07/18 20:59:49 ERROR executor.Executor: Exception in task 0.0 in stage 41.0 (TID 41)

java.lang.IllegalArgumentException: RejectRows does not exist. Available: IssuerId, IssuerId2, BusinessDate, StateCode, SourceName, NetworkName, NetworkURL, custnum, MarketCoverage, DentalOnlyPlan

at org.apache.spark.sql.types.StructType.$anonfun$fieldIndex$1(StructType.scala:306)

at scala.collection.immutable.HashMap$HashTrieMap.getOrElse0(HashMap.scala:596)

at scala.collection.immutable.HashMap.getOrElse(HashMap.scala:73)

at org.apache.spark.sql.types.StructType.fieldIndex(StructType.scala:305)

at org.apache.spark.sql.catalyst.OrderedFilters.$anonfun$predicates$4(OrderedFilters.scala:61)

at org.apache.spark.sql.catalyst.OrderedFilters.$anonfun$predicates$4$adapted(OrderedFilters.scala:61)

at scala.collection.TraversableLike.$anonfun$map$1(TraversableLike.scala:286)

at scala.collection.IndexedSeqOptimized.foreach(IndexedSeqOptimized.scala:36)

at scala.collection.IndexedSeqOptimized.foreach$(IndexedSeqOptimized.scala:33)

at scala.collection.mutable.ArrayOps$ofRef.foreach(ArrayOps.scala:198)

at scala.collection.TraversableLike.map(TraversableLike.scala:286)

at scala.collection.TraversableLike.map$(TraversableLike.scala:279)

at scala.collection.mutable.ArrayOps$ofRef.map(ArrayOps.scala:198)

at org.apache.spark.sql.catalyst.OrderedFilters.$anonfun$predicates$3(OrderedFilters.scala:61)

at org.apache.spark.sql.catalyst.OrderedFilters.$anonfun$predicates$3$adapted(OrderedFilters.scala:50)

at scala.collection.immutable.List.foreach(List.scala:431)

at org.apache.spark.sql.catalyst.OrderedFilters.<init>(OrderedFilters.scala:50)

at org.apache.spark.sql.catalyst.csv.UnivocityParser.<init>(UnivocityParser.scala:103)

at org.apache.spark.sql.execution.datasources.csv.CSVFileFormat.$anonfun$buildReader$1(CSVFileFormat.scala:130)

at org.apache.spark.sql.execution.datasources.FileFormat$$anon$1.apply(FileFormat.scala:148)

at org.apache.spark.sql.execution.datasources.FileFormat$$anon$1.apply(FileFormat.scala:133)

at org.apache.spark.sql.execution.datasources.FileScanRDD$$anon$1.org$apache$spark$sql$execution$datasources$FileScanRDD$$anon$$readCurrentFile(FileScanRDD.scala:127)

at org.apache.spark.sql.execution.datasources.FileScanRDD$$anon$1.nextIterator(FileScanRDD.scala:187)

at org.apache.spark.sql.execution.datasources.FileScanRDD$$anon$1.hasNext(FileScanRDD.scala:104)

at scala.collection.Iterator$$anon$10.hasNext(Iterator.scala:460)

at org.apache.spark.sql.catalyst.expressions.GeneratedClass$GeneratedIteratorForCodegenStage1.processNext(Unknown Source)

at org.apache.spark.sql.execution.BufferedRowIterator.hasNext(BufferedRowIterator.java:43)

at org.apache.spark.sql.execution.WholeStageCodegenExec$$anon$1.hasNext(WholeStageCodegenExec.scala:759)

at org.apache.spark.sql.execution.SparkPlan.$anonfun$getByteArrayRdd$1(SparkPlan.scala:349)

at org.apache.spark.rdd.RDD.$anonfun$mapPartitionsInternal$2(RDD.scala:898)

at org.apache.spark.rdd.RDD.$anonfun$mapPartitionsInternal$2$adapted(RDD.scala:898)

at org.apache.spark.rdd.MapPartitionsRDD.compute(MapPartitionsRDD.scala:52)

at org.apache.spark.rdd.RDD.computeOrReadCheckpoint(RDD.scala:373)

at org.apache.spark.rdd.RDD.iterator(RDD.scala:337)

at org.apache.spark.scheduler.ResultTask.runTask(ResultTask.scala:90)

at org.apache.spark.scheduler.Task.run(Task.scala:131)

at org.apache.spark.executor.Executor$TaskRunner.$anonfun$run$3(Executor.scala:506)

at org.apache.spark.util.Utils$.tryWithSafeFinally(Utils.scala:1462)

at org.apache.spark.executor.Executor$TaskRunner.run(Executor.scala:509)

at java.util.concurrent.ThreadPoolExecutor.runWorker(ThreadPoolExecutor.java:1149)

at java.util.concurrent.ThreadPoolExecutor$Worker.run(ThreadPoolExecutor.java:624)

at java.lang.Thread.run(Thread.java:750)

24/07/18 20:59:49 WARN scheduler.TaskSetManager: Lost task 0.0 in stage 41.0 (TID 41) (192.168.112.129 executor driver): java.lang.IllegalArgumentException: RejectRows does not exist. Available: IssuerId, IssuerId2, BusinessDate, StateCode, SourceName, NetworkName, NetworkURL, custnum, MarketCoverage, DentalOnlyPlan

at org.apache.spark.sql.types.StructType.$anonfun$fieldIndex$1(StructType.scala:306)

at scala.collection.immutable.HashMap$HashTrieMap.getOrElse0(HashMap.scala:596)

at scala.collection.immutable.HashMap.getOrElse(HashMap.scala:73)

at org.apache.spark.sql.types.StructType.fieldIndex(StructType.scala:305)

at org.apache.spark.sql.catalyst.OrderedFilters.$anonfun$predicates$4(OrderedFilters.scala:61)

at org.apache.spark.sql.catalyst.OrderedFilters.$anonfun$predicates$4$adapted(OrderedFilters.scala:61)

at scala.collection.TraversableLike.$anonfun$map$1(TraversableLike.scala:286)

at scala.collection.IndexedSeqOptimized.foreach(IndexedSeqOptimized.scala:36)

at scala.collection.IndexedSeqOptimized.foreach$(IndexedSeqOptimized.scala:33)

at scala.collection.mutable.ArrayOps$ofRef.foreach(ArrayOps.scala:198)

at scala.collection.TraversableLike.map(TraversableLike.scala:286)

at scala.collection.TraversableLike.map$(TraversableLike.scala:279)

at scala.collection.mutable.ArrayOps$ofRef.map(ArrayOps.scala:198)

at org.apache.spark.sql.catalyst.OrderedFilters.$anonfun$predicates$3(OrderedFilters.scala:61)

at org.apache.spark.sql.catalyst.OrderedFilters.$anonfun$predicates$3$adapted(OrderedFilters.scala:50)

at scala.collection.immutable.List.foreach(List.scala:431)

at org.apache.spark.sql.catalyst.OrderedFilters.<init>(OrderedFilters.scala:50)

at org.apache.spark.sql.catalyst.csv.UnivocityParser.<init>(UnivocityParser.scala:103)

at org.apache.spark.sql.execution.datasources.csv.CSVFileFormat.$anonfun$buildReader$1(CSVFileFormat.scala:130)

at org.apache.spark.sql.execution.datasources.FileFormat$$anon$1.apply(FileFormat.scala:148)

at org.apache.spark.sql.execution.datasources.FileFormat$$anon$1.apply(FileFormat.scala:133)

at org.apache.spark.sql.execution.datasources.FileScanRDD$$anon$1.org$apache$spark$sql$execution$datasources$FileScanRDD$$anon$$readCurrentFile(FileScanRDD.scala:127)

at org.apache.spark.sql.execution.datasources.FileScanRDD$$anon$1.nextIterator(FileScanRDD.scala:187)

at org.apache.spark.sql.execution.datasources.FileScanRDD$$anon$1.hasNext(FileScanRDD.scala:104)

at scala.collection.Iterator$$anon$10.hasNext(Iterator.scala:460)

at org.apache.spark.sql.catalyst.expressions.GeneratedClass$GeneratedIteratorForCodegenStage1.processNext(Unknown Source)

at org.apache.spark.sql.execution.BufferedRowIterator.hasNext(BufferedRowIterator.java:43)

at org.apache.spark.sql.execution.WholeStageCodegenExec$$anon$1.hasNext(WholeStageCodegenExec.scala:759)

at org.apache.spark.sql.execution.SparkPlan.$anonfun$getByteArrayRdd$1(SparkPlan.scala:349)

at org.apache.spark.rdd.RDD.$anonfun$mapPartitionsInternal$2(RDD.scala:898)

at org.apache.spark.rdd.RDD.$anonfun$mapPartitionsInternal$2$adapted(RDD.scala:898)

at org.apache.spark.rdd.MapPartitionsRDD.compute(MapPartitionsRDD.scala:52)

at org.apache.spark.rdd.RDD.computeOrReadCheckpoint(RDD.scala:373)

at org.apache.spark.rdd.RDD.iterator(RDD.scala:337)

at org.apache.spark.scheduler.ResultTask.runTask(ResultTask.scala:90)

at org.apache.spark.scheduler.Task.run(Task.scala:131)

at org.apache.spark.executor.Executor$TaskRunner.$anonfun$run$3(Executor.scala:506)

at org.apache.spark.util.Utils$.tryWithSafeFinally(Utils.scala:1462)

at org.apache.spark.executor.Executor$TaskRunner.run(Executor.scala:509)

at java.util.concurrent.ThreadPoolExecutor.runWorker(ThreadPoolExecutor.java:1149)

at java.util.concurrent.ThreadPoolExecutor$Worker.run(ThreadPoolExecutor.java:624)

at java.lang.Thread.run(Thread.java:750)

24/07/18 20:59:49 ERROR scheduler.TaskSetManager: Task 0 in stage 41.0 failed 1 times; aborting job

Traceback (most recent call last):

File "<stdin>", line 1, in <module>

File "/usr/local/spark/python/pyspark/sql/dataframe.py", line 494, in show

print(self.\_jdf.showString(n, 20, vertical))

File "/usr/local/spark/python/lib/py4j-0.10.9.3-src.zip/py4j/java\_gateway.py", line 1322, in \_\_call\_\_

File "/usr/local/spark/python/pyspark/sql/utils.py", line 111, in deco

return f(\*a, \*\*kw)

File "/usr/local/spark/python/lib/py4j-0.10.9.3-src.zip/py4j/protocol.py", line 328, in get\_return\_value

py4j.protocol.Py4JJavaError: An error occurred while calling o322.showString.

: org.apache.spark.SparkException: Job aborted due to stage failure: Task 0 in stage 41.0 failed 1 times, most recent failure: Lost task 0.0 in stage 41.0 (TID 41) (192.168.112.129 executor driver): java.lang.IllegalArgumentException: RejectRows does not exist. Available: IssuerId, IssuerId2, BusinessDate, StateCode, SourceName, NetworkName, NetworkURL, custnum, MarketCoverage, DentalOnlyPlan

at org.apache.spark.sql.types.StructType.$anonfun$fieldIndex$1(StructType.scala:306)

at scala.collection.immutable.HashMap$HashTrieMap.getOrElse0(HashMap.scala:596)

at scala.collection.immutable.HashMap.getOrElse(HashMap.scala:73)

at org.apache.spark.sql.types.StructType.fieldIndex(StructType.scala:305)

at org.apache.spark.sql.catalyst.OrderedFilters.$anonfun$predicates$4(OrderedFilters.scala:61)

at org.apache.spark.sql.catalyst.OrderedFilters.$anonfun$predicates$4$adapted(OrderedFilters.scala:61)

at scala.collection.TraversableLike.$anonfun$map$1(TraversableLike.scala:286)

at scala.collection.IndexedSeqOptimized.foreach(IndexedSeqOptimized.scala:36)

at scala.collection.IndexedSeqOptimized.foreach$(IndexedSeqOptimized.scala:33)

at scala.collection.mutable.ArrayOps$ofRef.foreach(ArrayOps.scala:198)

at scala.collection.TraversableLike.map(TraversableLike.scala:286)

at scala.collection.TraversableLike.map$(TraversableLike.scala:279)

at scala.collection.mutable.ArrayOps$ofRef.map(ArrayOps.scala:198)

at org.apache.spark.sql.catalyst.OrderedFilters.$anonfun$predicates$3(OrderedFilters.scala:61)

at org.apache.spark.sql.catalyst.OrderedFilters.$anonfun$predicates$3$adapted(OrderedFilters.scala:50)

at scala.collection.immutable.List.foreach(List.scala:431)

at org.apache.spark.sql.catalyst.OrderedFilters.<init>(OrderedFilters.scala:50)

at org.apache.spark.sql.catalyst.csv.UnivocityParser.<init>(UnivocityParser.scala:103)

at org.apache.spark.sql.execution.datasources.csv.CSVFileFormat.$anonfun$buildReader$1(CSVFileFormat.scala:130)

at org.apache.spark.sql.execution.datasources.FileFormat$$anon$1.apply(FileFormat.scala:148)

at org.apache.spark.sql.execution.datasources.FileFormat$$anon$1.apply(FileFormat.scala:133)

at org.apache.spark.sql.execution.datasources.FileScanRDD$$anon$1.org$apache$spark$sql$execution$datasources$FileScanRDD$$anon$$readCurrentFile(FileScanRDD.scala:127)

at org.apache.spark.sql.execution.datasources.FileScanRDD$$anon$1.nextIterator(FileScanRDD.scala:187)

at org.apache.spark.sql.execution.datasources.FileScanRDD$$anon$1.hasNext(FileScanRDD.scala:104)

at scala.collection.Iterator$$anon$10.hasNext(Iterator.scala:460)

at org.apache.spark.sql.catalyst.expressions.GeneratedClass$GeneratedIteratorForCodegenStage1.processNext(Unknown Source)

at org.apache.spark.sql.execution.BufferedRowIterator.hasNext(BufferedRowIterator.java:43)

at org.apache.spark.sql.execution.WholeStageCodegenExec$$anon$1.hasNext(WholeStageCodegenExec.scala:759)

at org.apache.spark.sql.execution.SparkPlan.$anonfun$getByteArrayRdd$1(SparkPlan.scala:349)

at org.apache.spark.rdd.RDD.$anonfun$mapPartitionsInternal$2(RDD.scala:898)

at org.apache.spark.rdd.RDD.$anonfun$mapPartitionsInternal$2$adapted(RDD.scala:898)

at org.apache.spark.rdd.MapPartitionsRDD.compute(MapPartitionsRDD.scala:52)

at org.apache.spark.rdd.RDD.computeOrReadCheckpoint(RDD.scala:373)

at org.apache.spark.rdd.RDD.iterator(RDD.scala:337)

at org.apache.spark.scheduler.ResultTask.runTask(ResultTask.scala:90)

at org.apache.spark.scheduler.Task.run(Task.scala:131)

at org.apache.spark.executor.Executor$TaskRunner.$anonfun$run$3(Executor.scala:506)

at org.apache.spark.util.Utils$.tryWithSafeFinally(Utils.scala:1462)

at org.apache.spark.executor.Executor$TaskRunner.run(Executor.scala:509)

at java.util.concurrent.ThreadPoolExecutor.runWorker(ThreadPoolExecutor.java:1149)

at java.util.concurrent.ThreadPoolExecutor$Worker.run(ThreadPoolExecutor.java:624)

at java.lang.Thread.run(Thread.java:750)

Driver stacktrace:

at org.apache.spark.scheduler.DAGScheduler.failJobAndIndependentStages(DAGScheduler.scala:2454)

at org.apache.spark.scheduler.DAGScheduler.$anonfun$abortStage$2(DAGScheduler.scala:2403)

at org.apache.spark.scheduler.DAGScheduler.$anonfun$abortStage$2$adapted(DAGScheduler.scala:2402)

at scala.collection.mutable.ResizableArray.foreach(ResizableArray.scala:62)

at scala.collection.mutable.ResizableArray.foreach$(ResizableArray.scala:55)

at scala.collection.mutable.ArrayBuffer.foreach(ArrayBuffer.scala:49)

at org.apache.spark.scheduler.DAGScheduler.abortStage(DAGScheduler.scala:2402)

at org.apache.spark.scheduler.DAGScheduler.$anonfun$handleTaskSetFailed$1(DAGScheduler.scala:1160)

at org.apache.spark.scheduler.DAGScheduler.$anonfun$handleTaskSetFailed$1$adapted(DAGScheduler.scala:1160)

at scala.Option.foreach(Option.scala:407)

at org.apache.spark.scheduler.DAGScheduler.handleTaskSetFailed(DAGScheduler.scala:1160)

at org.apache.spark.scheduler.DAGSchedulerEventProcessLoop.doOnReceive(DAGScheduler.scala:2642)

at org.apache.spark.scheduler.DAGSchedulerEventProcessLoop.onReceive(DAGScheduler.scala:2584)

at org.apache.spark.scheduler.DAGSchedulerEventProcessLoop.onReceive(DAGScheduler.scala:2573)

at org.apache.spark.util.EventLoop$$anon$1.run(EventLoop.scala:49)

at org.apache.spark.scheduler.DAGScheduler.runJob(DAGScheduler.scala:938)

at org.apache.spark.SparkContext.runJob(SparkContext.scala:2214)

at org.apache.spark.SparkContext.runJob(SparkContext.scala:2235)

at org.apache.spark.SparkContext.runJob(SparkContext.scala:2254)

at org.apache.spark.sql.execution.SparkPlan.executeTake(SparkPlan.scala:476)

at org.apache.spark.sql.execution.SparkPlan.executeTake(SparkPlan.scala:429)

at org.apache.spark.sql.execution.CollectLimitExec.executeCollect(limit.scala:48)

at org.apache.spark.sql.Dataset.collectFromPlan(Dataset.scala:3715)

at org.apache.spark.sql.Dataset.$anonfun$head$1(Dataset.scala:2728)

at org.apache.spark.sql.Dataset.$anonfun$withAction$1(Dataset.scala:3706)

at org.apache.spark.sql.execution.SQLExecution$.$anonfun$withNewExecutionId$5(SQLExecution.scala:103)

at org.apache.spark.sql.execution.SQLExecution$.withSQLConfPropagated(SQLExecution.scala:163)

at org.apache.spark.sql.execution.SQLExecution$.$anonfun$withNewExecutionId$1(SQLExecution.scala:90)

at org.apache.spark.sql.SparkSession.withActive(SparkSession.scala:775)

at org.apache.spark.sql.execution.SQLExecution$.withNewExecutionId(SQLExecution.scala:64)

at org.apache.spark.sql.Dataset.withAction(Dataset.scala:3704)

at org.apache.spark.sql.Dataset.head(Dataset.scala:2728)

at org.apache.spark.sql.Dataset.take(Dataset.scala:2935)

at org.apache.spark.sql.Dataset.getRows(Dataset.scala:287)

at org.apache.spark.sql.Dataset.showString(Dataset.scala:326)

at sun.reflect.GeneratedMethodAccessor41.invoke(Unknown Source)

at sun.reflect.DelegatingMethodAccessorImpl.invoke(DelegatingMethodAccessorImpl.java:43)

at java.lang.reflect.Method.invoke(Method.java:498)

at py4j.reflection.MethodInvoker.invoke(MethodInvoker.java:244)

at py4j.reflection.ReflectionEngine.invoke(ReflectionEngine.java:357)

at py4j.Gateway.invoke(Gateway.java:282)

at py4j.commands.AbstractCommand.invokeMethod(AbstractCommand.java:132)

at py4j.commands.CallCommand.execute(CallCommand.java:79)

at py4j.ClientServerConnection.waitForCommands(ClientServerConnection.java:182)

at py4j.ClientServerConnection.run(ClientServerConnection.java:106)

at java.lang.Thread.run(Thread.java:750)

Caused by: java.lang.IllegalArgumentException: RejectRows does not exist. Available: IssuerId, IssuerId2, BusinessDate, StateCode, SourceName, NetworkName, NetworkURL, custnum, MarketCoverage, DentalOnlyPlan

at org.apache.spark.sql.types.StructType.$anonfun$fieldIndex$1(StructType.scala:306)

at scala.collection.immutable.HashMap$HashTrieMap.getOrElse0(HashMap.scala:596)

at scala.collection.immutable.HashMap.getOrElse(HashMap.scala:73)

at org.apache.spark.sql.types.StructType.fieldIndex(StructType.scala:305)

at org.apache.spark.sql.catalyst.OrderedFilters.$anonfun$predicates$4(OrderedFilters.scala:61)

at org.apache.spark.sql.catalyst.OrderedFilters.$anonfun$predicates$4$adapted(OrderedFilters.scala:61)

at scala.collection.TraversableLike.$anonfun$map$1(TraversableLike.scala:286)

at scala.collection.IndexedSeqOptimized.foreach(IndexedSeqOptimized.scala:36)

at scala.collection.IndexedSeqOptimized.foreach$(IndexedSeqOptimized.scala:33)

at scala.collection.mutable.ArrayOps$ofRef.foreach(ArrayOps.scala:198)

at scala.collection.TraversableLike.map(TraversableLike.scala:286)

at scala.collection.TraversableLike.map$(TraversableLike.scala:279)

at scala.collection.mutable.ArrayOps$ofRef.map(ArrayOps.scala:198)

at org.apache.spark.sql.catalyst.OrderedFilters.$anonfun$predicates$3(OrderedFilters.scala:61)

at org.apache.spark.sql.catalyst.OrderedFilters.$anonfun$predicates$3$adapted(OrderedFilters.scala:50)

at scala.collection.immutable.List.foreach(List.scala:431)

at org.apache.spark.sql.catalyst.OrderedFilters.<init>(OrderedFilters.scala:50)

at org.apache.spark.sql.catalyst.csv.UnivocityParser.<init>(UnivocityParser.scala:103)

at org.apache.spark.sql.execution.datasources.csv.CSVFileFormat.$anonfun$buildReader$1(CSVFileFormat.scala:130)

at org.apache.spark.sql.execution.datasources.FileFormat$$anon$1.apply(FileFormat.scala:148)

at org.apache.spark.sql.execution.datasources.FileFormat$$anon$1.apply(FileFormat.scala:133)

at org.apache.spark.sql.execution.datasources.FileScanRDD$$anon$1.org$apache$spark$sql$execution$datasources$FileScanRDD$$anon$$readCurrentFile(FileScanRDD.scala:127)

at org.apache.spark.sql.execution.datasources.FileScanRDD$$anon$1.nextIterator(FileScanRDD.scala:187)

at org.apache.spark.sql.execution.datasources.FileScanRDD$$anon$1.hasNext(FileScanRDD.scala:104)

at scala.collection.Iterator$$anon$10.hasNext(Iterator.scala:460)

at org.apache.spark.sql.catalyst.expressions.GeneratedClass$GeneratedIteratorForCodegenStage1.processNext(Unknown Source)

at org.apache.spark.sql.execution.BufferedRowIterator.hasNext(BufferedRowIterator.java:43)

at org.apache.spark.sql.execution.WholeStageCodegenExec$$anon$1.hasNext(WholeStageCodegenExec.scala:759)

at org.apache.spark.sql.execution.SparkPlan.$anonfun$getByteArrayRdd$1(SparkPlan.scala:349)

at org.apache.spark.rdd.RDD.$anonfun$mapPartitionsInternal$2(RDD.scala:898)

at org.apache.spark.rdd.RDD.$anonfun$mapPartitionsInternal$2$adapted(RDD.scala:898)

at org.apache.spark.rdd.MapPartitionsRDD.compute(MapPartitionsRDD.scala:52)

at org.apache.spark.rdd.RDD.computeOrReadCheckpoint(RDD.scala:373)

at org.apache.spark.rdd.RDD.iterator(RDD.scala:337)

at org.apache.spark.scheduler.ResultTask.runTask(ResultTask.scala:90)

at org.apache.spark.scheduler.Task.run(Task.scala:131)

at org.apache.spark.executor.Executor$TaskRunner.$anonfun$run$3(Executor.scala:506)

at org.apache.spark.util.Utils$.tryWithSafeFinally(Utils.scala:1462)

at org.apache.spark.executor.Executor$TaskRunner.run(Executor.scala:509)

at java.util.concurrent.ThreadPoolExecutor.runWorker(ThreadPoolExecutor.java:1149)

at java.util.concurrent.ThreadPoolExecutor$Worker.run(ThreadPoolExecutor.java:624)

... 1 more

>>> data\_frame\_1.where(col("custnum").isNull()).show(3)

+--------+---------+------------+---------+----------+-----------+----------+-------+--------------+--------------+--------------------+

|IssuerId|IssuerId2|BusinessDate|StateCode|SourceName|NetworkName|NetworkURL|custnum|MarketCoverage|DentalOnlyPlan| RejectRows|

+--------+---------+------------+---------+----------+-----------+----------+-------+--------------+--------------+--------------------+

| 219| 219| 2019-10-01| AK| HIOS|ODS Premier| null| null| null| null|219,219,2019-10-0...|

| null| null| null| null| null| null| null| null| null| null| footer count is 402|

+--------+---------+------------+---------+----------+-----------+----------+-------+--------------+--------------+--------------------+

>>> data\_frame\_1.where(col("RejectRows").isNull()).show(3)

24/07/18 21:00:24 ERROR executor.Executor: Exception in task 0.0 in stage 43.0 (TID 43)

java.lang.IllegalArgumentException: RejectRows does not exist. Available: IssuerId, IssuerId2, BusinessDate, StateCode, SourceName, NetworkName, NetworkURL, custnum, MarketCoverage, DentalOnlyPlan

at org.apache.spark.sql.types.StructType.$anonfun$fieldIndex$1(StructType.scala:306)

at scala.collection.immutable.HashMap$HashTrieMap.getOrElse0(HashMap.scala:596)

at scala.collection.immutable.HashMap.getOrElse(HashMap.scala:73)

at org.apache.spark.sql.types.StructType.fieldIndex(StructType.scala:305)

at org.apache.spark.sql.catalyst.OrderedFilters.$anonfun$predicates$4(OrderedFilters.scala:61)

at org.apache.spark.sql.catalyst.OrderedFilters.$anonfun$predicates$4$adapted(OrderedFilters.scala:61)

at scala.collection.TraversableLike.$anonfun$map$1(TraversableLike.scala:286)

at scala.collection.IndexedSeqOptimized.foreach(IndexedSeqOptimized.scala:36)

at scala.collection.IndexedSeqOptimized.foreach$(IndexedSeqOptimized.scala:33)

at scala.collection.mutable.ArrayOps$ofRef.foreach(ArrayOps.scala:198)

at scala.collection.TraversableLike.map(TraversableLike.scala:286)

at scala.collection.TraversableLike.map$(TraversableLike.scala:279)

at scala.collection.mutable.ArrayOps$ofRef.map(ArrayOps.scala:198)

at org.apache.spark.sql.catalyst.OrderedFilters.$anonfun$predicates$3(OrderedFilters.scala:61)

at org.apache.spark.sql.catalyst.OrderedFilters.$anonfun$predicates$3$adapted(OrderedFilters.scala:50)

at scala.collection.immutable.List.foreach(List.scala:431)

at org.apache.spark.sql.catalyst.OrderedFilters.<init>(OrderedFilters.scala:50)

at org.apache.spark.sql.catalyst.csv.UnivocityParser.<init>(UnivocityParser.scala:103)

at org.apache.spark.sql.execution.datasources.csv.CSVFileFormat.$anonfun$buildReader$1(CSVFileFormat.scala:130)

at org.apache.spark.sql.execution.datasources.FileFormat$$anon$1.apply(FileFormat.scala:148)

at org.apache.spark.sql.execution.datasources.FileFormat$$anon$1.apply(FileFormat.scala:133)

at org.apache.spark.sql.execution.datasources.FileScanRDD$$anon$1.org$apache$spark$sql$execution$datasources$FileScanRDD$$anon$$readCurrentFile(FileScanRDD.scala:127)

at org.apache.spark.sql.execution.datasources.FileScanRDD$$anon$1.nextIterator(FileScanRDD.scala:187)

at org.apache.spark.sql.execution.datasources.FileScanRDD$$anon$1.hasNext(FileScanRDD.scala:104)

at scala.collection.Iterator$$anon$10.hasNext(Iterator.scala:460)

at org.apache.spark.sql.catalyst.expressions.GeneratedClass$GeneratedIteratorForCodegenStage1.processNext(Unknown Source)

at org.apache.spark.sql.execution.BufferedRowIterator.hasNext(BufferedRowIterator.java:43)

at org.apache.spark.sql.execution.WholeStageCodegenExec$$anon$1.hasNext(WholeStageCodegenExec.scala:759)

at org.apache.spark.sql.execution.SparkPlan.$anonfun$getByteArrayRdd$1(SparkPlan.scala:349)

at org.apache.spark.rdd.RDD.$anonfun$mapPartitionsInternal$2(RDD.scala:898)

at org.apache.spark.rdd.RDD.$anonfun$mapPartitionsInternal$2$adapted(RDD.scala:898)

at org.apache.spark.rdd.MapPartitionsRDD.compute(MapPartitionsRDD.scala:52)

at org.apache.spark.rdd.RDD.computeOrReadCheckpoint(RDD.scala:373)

at org.apache.spark.rdd.RDD.iterator(RDD.scala:337)

at org.apache.spark.scheduler.ResultTask.runTask(ResultTask.scala:90)

at org.apache.spark.scheduler.Task.run(Task.scala:131)

at org.apache.spark.executor.Executor$TaskRunner.$anonfun$run$3(Executor.scala:506)

at org.apache.spark.util.Utils$.tryWithSafeFinally(Utils.scala:1462)

at org.apache.spark.executor.Executor$TaskRunner.run(Executor.scala:509)

at java.util.concurrent.ThreadPoolExecutor.runWorker(ThreadPoolExecutor.java:1149)

at java.util.concurrent.ThreadPoolExecutor$Worker.run(ThreadPoolExecutor.java:624)

at java.lang.Thread.run(Thread.java:750)

24/07/18 21:00:24 WARN scheduler.TaskSetManager: Lost task 0.0 in stage 43.0 (TID 43) (192.168.112.129 executor driver): java.lang.IllegalArgumentException: RejectRows does not exist. Available: IssuerId, IssuerId2, BusinessDate, StateCode, SourceName, NetworkName, NetworkURL, custnum, MarketCoverage, DentalOnlyPlan

at org.apache.spark.sql.types.StructType.$anonfun$fieldIndex$1(StructType.scala:306)

at scala.collection.immutable.HashMap$HashTrieMap.getOrElse0(HashMap.scala:596)

at scala.collection.immutable.HashMap.getOrElse(HashMap.scala:73)

at org.apache.spark.sql.types.StructType.fieldIndex(StructType.scala:305)

at org.apache.spark.sql.catalyst.OrderedFilters.$anonfun$predicates$4(OrderedFilters.scala:61)

at org.apache.spark.sql.catalyst.OrderedFilters.$anonfun$predicates$4$adapted(OrderedFilters.scala:61)

at scala.collection.TraversableLike.$anonfun$map$1(TraversableLike.scala:286)

at scala.collection.IndexedSeqOptimized.foreach(IndexedSeqOptimized.scala:36)

at scala.collection.IndexedSeqOptimized.foreach$(IndexedSeqOptimized.scala:33)

at scala.collection.mutable.ArrayOps$ofRef.foreach(ArrayOps.scala:198)

at scala.collection.TraversableLike.map(TraversableLike.scala:286)

at scala.collection.TraversableLike.map$(TraversableLike.scala:279)

at scala.collection.mutable.ArrayOps$ofRef.map(ArrayOps.scala:198)

at org.apache.spark.sql.catalyst.OrderedFilters.$anonfun$predicates$3(OrderedFilters.scala:61)

at org.apache.spark.sql.catalyst.OrderedFilters.$anonfun$predicates$3$adapted(OrderedFilters.scala:50)

at scala.collection.immutable.List.foreach(List.scala:431)

at org.apache.spark.sql.catalyst.OrderedFilters.<init>(OrderedFilters.scala:50)

at org.apache.spark.sql.catalyst.csv.UnivocityParser.<init>(UnivocityParser.scala:103)

at org.apache.spark.sql.execution.datasources.csv.CSVFileFormat.$anonfun$buildReader$1(CSVFileFormat.scala:130)

at org.apache.spark.sql.execution.datasources.FileFormat$$anon$1.apply(FileFormat.scala:148)

at org.apache.spark.sql.execution.datasources.FileFormat$$anon$1.apply(FileFormat.scala:133)

at org.apache.spark.sql.execution.datasources.FileScanRDD$$anon$1.org$apache$spark$sql$execution$datasources$FileScanRDD$$anon$$readCurrentFile(FileScanRDD.scala:127)

at org.apache.spark.sql.execution.datasources.FileScanRDD$$anon$1.nextIterator(FileScanRDD.scala:187)

at org.apache.spark.sql.execution.datasources.FileScanRDD$$anon$1.hasNext(FileScanRDD.scala:104)

at scala.collection.Iterator$$anon$10.hasNext(Iterator.scala:460)

at org.apache.spark.sql.catalyst.expressions.GeneratedClass$GeneratedIteratorForCodegenStage1.processNext(Unknown Source)

at org.apache.spark.sql.execution.BufferedRowIterator.hasNext(BufferedRowIterator.java:43)

at org.apache.spark.sql.execution.WholeStageCodegenExec$$anon$1.hasNext(WholeStageCodegenExec.scala:759)

at org.apache.spark.sql.execution.SparkPlan.$anonfun$getByteArrayRdd$1(SparkPlan.scala:349)

at org.apache.spark.rdd.RDD.$anonfun$mapPartitionsInternal$2(RDD.scala:898)

at org.apache.spark.rdd.RDD.$anonfun$mapPartitionsInternal$2$adapted(RDD.scala:898)

at org.apache.spark.rdd.MapPartitionsRDD.compute(MapPartitionsRDD.scala:52)

at org.apache.spark.rdd.RDD.computeOrReadCheckpoint(RDD.scala:373)

at org.apache.spark.rdd.RDD.iterator(RDD.scala:337)

at org.apache.spark.scheduler.ResultTask.runTask(ResultTask.scala:90)

at org.apache.spark.scheduler.Task.run(Task.scala:131)

at org.apache.spark.executor.Executor$TaskRunner.$anonfun$run$3(Executor.scala:506)

at org.apache.spark.util.Utils$.tryWithSafeFinally(Utils.scala:1462)

at org.apache.spark.executor.Executor$TaskRunner.run(Executor.scala:509)

at java.util.concurrent.ThreadPoolExecutor.runWorker(ThreadPoolExecutor.java:1149)

at java.util.concurrent.ThreadPoolExecutor$Worker.run(ThreadPoolExecutor.java:624)

at java.lang.Thread.run(Thread.java:750)

24/07/18 21:00:24 ERROR scheduler.TaskSetManager: Task 0 in stage 43.0 failed 1 times; aborting job

Traceback (most recent call last):

File "<stdin>", line 1, in <module>

File "/usr/local/spark/python/pyspark/sql/dataframe.py", line 494, in show

print(self.\_jdf.showString(n, 20, vertical))

File "/usr/local/spark/python/lib/py4j-0.10.9.3-src.zip/py4j/java\_gateway.py", line 1322, in \_\_call\_\_

File "/usr/local/spark/python/pyspark/sql/utils.py", line 111, in deco

return f(\*a, \*\*kw)

File "/usr/local/spark/python/lib/py4j-0.10.9.3-src.zip/py4j/protocol.py", line 328, in get\_return\_value

py4j.protocol.Py4JJavaError: An error occurred while calling o330.showString.

: org.apache.spark.SparkException: Job aborted due to stage failure: Task 0 in stage 43.0 failed 1 times, most recent failure: Lost task 0.0 in stage 43.0 (TID 43) (192.168.112.129 executor driver): java.lang.IllegalArgumentException: RejectRows does not exist. Available: IssuerId, IssuerId2, BusinessDate, StateCode, SourceName, NetworkName, NetworkURL, custnum, MarketCoverage, DentalOnlyPlan

at org.apache.spark.sql.types.StructType.$anonfun$fieldIndex$1(StructType.scala:306)

at scala.collection.immutable.HashMap$HashTrieMap.getOrElse0(HashMap.scala:596)

at scala.collection.immutable.HashMap.getOrElse(HashMap.scala:73)

at org.apache.spark.sql.types.StructType.fieldIndex(StructType.scala:305)

at org.apache.spark.sql.catalyst.OrderedFilters.$anonfun$predicates$4(OrderedFilters.scala:61)

at org.apache.spark.sql.catalyst.OrderedFilters.$anonfun$predicates$4$adapted(OrderedFilters.scala:61)

at scala.collection.TraversableLike.$anonfun$map$1(TraversableLike.scala:286)

at scala.collection.IndexedSeqOptimized.foreach(IndexedSeqOptimized.scala:36)

at scala.collection.IndexedSeqOptimized.foreach$(IndexedSeqOptimized.scala:33)

at scala.collection.mutable.ArrayOps$ofRef.foreach(ArrayOps.scala:198)

at scala.collection.TraversableLike.map(TraversableLike.scala:286)

at scala.collection.TraversableLike.map$(TraversableLike.scala:279)

at scala.collection.mutable.ArrayOps$ofRef.map(ArrayOps.scala:198)

at org.apache.spark.sql.catalyst.OrderedFilters.$anonfun$predicates$3(OrderedFilters.scala:61)

at org.apache.spark.sql.catalyst.OrderedFilters.$anonfun$predicates$3$adapted(OrderedFilters.scala:50)

at scala.collection.immutable.List.foreach(List.scala:431)

at org.apache.spark.sql.catalyst.OrderedFilters.<init>(OrderedFilters.scala:50)

at org.apache.spark.sql.catalyst.csv.UnivocityParser.<init>(UnivocityParser.scala:103)

at org.apache.spark.sql.execution.datasources.csv.CSVFileFormat.$anonfun$buildReader$1(CSVFileFormat.scala:130)

at org.apache.spark.sql.execution.datasources.FileFormat$$anon$1.apply(FileFormat.scala:148)

at org.apache.spark.sql.execution.datasources.FileFormat$$anon$1.apply(FileFormat.scala:133)

at org.apache.spark.sql.execution.datasources.FileScanRDD$$anon$1.org$apache$spark$sql$execution$datasources$FileScanRDD$$anon$$readCurrentFile(FileScanRDD.scala:127)

at org.apache.spark.sql.execution.datasources.FileScanRDD$$anon$1.nextIterator(FileScanRDD.scala:187)

at org.apache.spark.sql.execution.datasources.FileScanRDD$$anon$1.hasNext(FileScanRDD.scala:104)

at scala.collection.Iterator$$anon$10.hasNext(Iterator.scala:460)

at org.apache.spark.sql.catalyst.expressions.GeneratedClass$GeneratedIteratorForCodegenStage1.processNext(Unknown Source)

at org.apache.spark.sql.execution.BufferedRowIterator.hasNext(BufferedRowIterator.java:43)

at org.apache.spark.sql.execution.WholeStageCodegenExec$$anon$1.hasNext(WholeStageCodegenExec.scala:759)

at org.apache.spark.sql.execution.SparkPlan.$anonfun$getByteArrayRdd$1(SparkPlan.scala:349)

at org.apache.spark.rdd.RDD.$anonfun$mapPartitionsInternal$2(RDD.scala:898)

at org.apache.spark.rdd.RDD.$anonfun$mapPartitionsInternal$2$adapted(RDD.scala:898)

at org.apache.spark.rdd.MapPartitionsRDD.compute(MapPartitionsRDD.scala:52)

at org.apache.spark.rdd.RDD.computeOrReadCheckpoint(RDD.scala:373)

at org.apache.spark.rdd.RDD.iterator(RDD.scala:337)

at org.apache.spark.scheduler.ResultTask.runTask(ResultTask.scala:90)

at org.apache.spark.scheduler.Task.run(Task.scala:131)

at org.apache.spark.executor.Executor$TaskRunner.$anonfun$run$3(Executor.scala:506)

at org.apache.spark.util.Utils$.tryWithSafeFinally(Utils.scala:1462)

at org.apache.spark.executor.Executor$TaskRunner.run(Executor.scala:509)

at java.util.concurrent.ThreadPoolExecutor.runWorker(ThreadPoolExecutor.java:1149)

at java.util.concurrent.ThreadPoolExecutor$Worker.run(ThreadPoolExecutor.java:624)

at java.lang.Thread.run(Thread.java:750)

Driver stacktrace:

at org.apache.spark.scheduler.DAGScheduler.failJobAndIndependentStages(DAGScheduler.scala:2454)

at org.apache.spark.scheduler.DAGScheduler.$anonfun$abortStage$2(DAGScheduler.scala:2403)

at org.apache.spark.scheduler.DAGScheduler.$anonfun$abortStage$2$adapted(DAGScheduler.scala:2402)

at scala.collection.mutable.ResizableArray.foreach(ResizableArray.scala:62)

at scala.collection.mutable.ResizableArray.foreach$(ResizableArray.scala:55)

at scala.collection.mutable.ArrayBuffer.foreach(ArrayBuffer.scala:49)

at org.apache.spark.scheduler.DAGScheduler.abortStage(DAGScheduler.scala:2402)

at org.apache.spark.scheduler.DAGScheduler.$anonfun$handleTaskSetFailed$1(DAGScheduler.scala:1160)

at org.apache.spark.scheduler.DAGScheduler.$anonfun$handleTaskSetFailed$1$adapted(DAGScheduler.scala:1160)

at scala.Option.foreach(Option.scala:407)

at org.apache.spark.scheduler.DAGScheduler.handleTaskSetFailed(DAGScheduler.scala:1160)

at org.apache.spark.scheduler.DAGSchedulerEventProcessLoop.doOnReceive(DAGScheduler.scala:2642)

at org.apache.spark.scheduler.DAGSchedulerEventProcessLoop.onReceive(DAGScheduler.scala:2584)

at org.apache.spark.scheduler.DAGSchedulerEventProcessLoop.onReceive(DAGScheduler.scala:2573)

at org.apache.spark.util.EventLoop$$anon$1.run(EventLoop.scala:49)

at org.apache.spark.scheduler.DAGScheduler.runJob(DAGScheduler.scala:938)

at org.apache.spark.SparkContext.runJob(SparkContext.scala:2214)

at org.apache.spark.SparkContext.runJob(SparkContext.scala:2235)

at org.apache.spark.SparkContext.runJob(SparkContext.scala:2254)

at org.apache.spark.sql.execution.SparkPlan.executeTake(SparkPlan.scala:476)

at org.apache.spark.sql.execution.SparkPlan.executeTake(SparkPlan.scala:429)

at org.apache.spark.sql.execution.CollectLimitExec.executeCollect(limit.scala:48)

at org.apache.spark.sql.Dataset.collectFromPlan(Dataset.scala:3715)

at org.apache.spark.sql.Dataset.$anonfun$head$1(Dataset.scala:2728)

at org.apache.spark.sql.Dataset.$anonfun$withAction$1(Dataset.scala:3706)

at org.apache.spark.sql.execution.SQLExecution$.$anonfun$withNewExecutionId$5(SQLExecution.scala:103)

at org.apache.spark.sql.execution.SQLExecution$.withSQLConfPropagated(SQLExecution.scala:163)

at org.apache.spark.sql.execution.SQLExecution$.$anonfun$withNewExecutionId$1(SQLExecution.scala:90)

at org.apache.spark.sql.SparkSession.withActive(SparkSession.scala:775)

at org.apache.spark.sql.execution.SQLExecution$.withNewExecutionId(SQLExecution.scala:64)

at org.apache.spark.sql.Dataset.withAction(Dataset.scala:3704)

at org.apache.spark.sql.Dataset.head(Dataset.scala:2728)

at org.apache.spark.sql.Dataset.take(Dataset.scala:2935)

at org.apache.spark.sql.Dataset.getRows(Dataset.scala:287)

at org.apache.spark.sql.Dataset.showString(Dataset.scala:326)

at sun.reflect.GeneratedMethodAccessor41.invoke(Unknown Source)

at sun.reflect.DelegatingMethodAccessorImpl.invoke(DelegatingMethodAccessorImpl.java:43)

at java.lang.reflect.Method.invoke(Method.java:498)

at py4j.reflection.MethodInvoker.invoke(MethodInvoker.java:244)

at py4j.reflection.ReflectionEngine.invoke(ReflectionEngine.java:357)

at py4j.Gateway.invoke(Gateway.java:282)

at py4j.commands.AbstractCommand.invokeMethod(AbstractCommand.java:132)

at py4j.commands.CallCommand.execute(CallCommand.java:79)

at py4j.ClientServerConnection.waitForCommands(ClientServerConnection.java:182)

at py4j.ClientServerConnection.run(ClientServerConnection.java:106)

at java.lang.Thread.run(Thread.java:750)

Caused by: java.lang.IllegalArgumentException: RejectRows does not exist. Available: IssuerId, IssuerId2, BusinessDate, StateCode, SourceName, NetworkName, NetworkURL, custnum, MarketCoverage, DentalOnlyPlan

at org.apache.spark.sql.types.StructType.$anonfun$fieldIndex$1(StructType.scala:306)

at scala.collection.immutable.HashMap$HashTrieMap.getOrElse0(HashMap.scala:596)

at scala.collection.immutable.HashMap.getOrElse(HashMap.scala:73)

at org.apache.spark.sql.types.StructType.fieldIndex(StructType.scala:305)

at org.apache.spark.sql.catalyst.OrderedFilters.$anonfun$predicates$4(OrderedFilters.scala:61)

at org.apache.spark.sql.catalyst.OrderedFilters.$anonfun$predicates$4$adapted(OrderedFilters.scala:61)

at scala.collection.TraversableLike.$anonfun$map$1(TraversableLike.scala:286)

at scala.collection.IndexedSeqOptimized.foreach(IndexedSeqOptimized.scala:36)

at scala.collection.IndexedSeqOptimized.foreach$(IndexedSeqOptimized.scala:33)

at scala.collection.mutable.ArrayOps$ofRef.foreach(ArrayOps.scala:198)

at scala.collection.TraversableLike.map(TraversableLike.scala:286)

at scala.collection.TraversableLike.map$(TraversableLike.scala:279)

at scala.collection.mutable.ArrayOps$ofRef.map(ArrayOps.scala:198)

at org.apache.spark.sql.catalyst.OrderedFilters.$anonfun$predicates$3(OrderedFilters.scala:61)

at org.apache.spark.sql.catalyst.OrderedFilters.$anonfun$predicates$3$adapted(OrderedFilters.scala:50)

at scala.collection.immutable.List.foreach(List.scala:431)

at org.apache.spark.sql.catalyst.OrderedFilters.<init>(OrderedFilters.scala:50)

at org.apache.spark.sql.catalyst.csv.UnivocityParser.<init>(UnivocityParser.scala:103)

at org.apache.spark.sql.execution.datasources.csv.CSVFileFormat.$anonfun$buildReader$1(CSVFileFormat.scala:130)

at org.apache.spark.sql.execution.datasources.FileFormat$$anon$1.apply(FileFormat.scala:148)

at org.apache.spark.sql.execution.datasources.FileFormat$$anon$1.apply(FileFormat.scala:133)

at org.apache.spark.sql.execution.datasources.FileScanRDD$$anon$1.org$apache$spark$sql$execution$datasources$FileScanRDD$$anon$$readCurrentFile(FileScanRDD.scala:127)

at org.apache.spark.sql.execution.datasources.FileScanRDD$$anon$1.nextIterator(FileScanRDD.scala:187)

at org.apache.spark.sql.execution.datasources.FileScanRDD$$anon$1.hasNext(FileScanRDD.scala:104)

at scala.collection.Iterator$$anon$10.hasNext(Iterator.scala:460)

at org.apache.spark.sql.catalyst.expressions.GeneratedClass$GeneratedIteratorForCodegenStage1.processNext(Unknown Source)

at org.apache.spark.sql.execution.BufferedRowIterator.hasNext(BufferedRowIterator.java:43)

at org.apache.spark.sql.execution.WholeStageCodegenExec$$anon$1.hasNext(WholeStageCodegenExec.scala:759)

at org.apache.spark.sql.execution.SparkPlan.$anonfun$getByteArrayRdd$1(SparkPlan.scala:349)

at org.apache.spark.rdd.RDD.$anonfun$mapPartitionsInternal$2(RDD.scala:898)

at org.apache.spark.rdd.RDD.$anonfun$mapPartitionsInternal$2$adapted(RDD.scala:898)

at org.apache.spark.rdd.MapPartitionsRDD.compute(MapPartitionsRDD.scala:52)

at org.apache.spark.rdd.RDD.computeOrReadCheckpoint(RDD.scala:373)

at org.apache.spark.rdd.RDD.iterator(RDD.scala:337)

at org.apache.spark.scheduler.ResultTask.runTask(ResultTask.scala:90)

at org.apache.spark.scheduler.Task.run(Task.scala:131)

at org.apache.spark.executor.Executor$TaskRunner.$anonfun$run$3(Executor.scala:506)

at org.apache.spark.util.Utils$.tryWithSafeFinally(Utils.scala:1462)

at org.apache.spark.executor.Executor$TaskRunner.run(Executor.scala:509)

at java.util.concurrent.ThreadPoolExecutor.runWorker(ThreadPoolExecutor.java:1149)

at java.util.concurrent.ThreadPoolExecutor$Worker.run(ThreadPoolExecutor.java:624)

... 1 more

>>> data\_frame\_1.describe

<bound method DataFrame.describe of DataFrame[IssuerId: int, IssuerId2: int, BusinessDate: date, StateCode: string, SourceName: string, NetworkName: string, NetworkURL: string, custnum: string, MarketCoverage: string, DentalOnlyPlan: string, RejectRows: string]>

>>> data\_frame\_1.filter((col("RejectRows").isNull()) | (col("RejectRows") == "")).show()

24/07/18 21:02:02 ERROR executor.Executor: Exception in task 0.0 in stage 44.0 (TID 44)

java.lang.IllegalArgumentException: RejectRows does not exist. Available: IssuerId, IssuerId2, BusinessDate, StateCode, SourceName, NetworkName, NetworkURL, custnum, MarketCoverage, DentalOnlyPlan

at org.apache.spark.sql.types.StructType.$anonfun$fieldIndex$1(StructType.scala:306)

at scala.collection.immutable.HashMap$HashTrieMap.getOrElse0(HashMap.scala:596)

at scala.collection.immutable.HashMap.getOrElse(HashMap.scala:73)

at org.apache.spark.sql.types.StructType.fieldIndex(StructType.scala:305)

at org.apache.spark.sql.catalyst.OrderedFilters.$anonfun$predicates$4(OrderedFilters.scala:61)

at org.apache.spark.sql.catalyst.OrderedFilters.$anonfun$predicates$4$adapted(OrderedFilters.scala:61)

at scala.collection.TraversableLike.$anonfun$map$1(TraversableLike.scala:286)

at scala.collection.IndexedSeqOptimized.foreach(IndexedSeqOptimized.scala:36)

at scala.collection.IndexedSeqOptimized.foreach$(IndexedSeqOptimized.scala:33)

at scala.collection.mutable.ArrayOps$ofRef.foreach(ArrayOps.scala:198)

at scala.collection.TraversableLike.map(TraversableLike.scala:286)

at scala.collection.TraversableLike.map$(TraversableLike.scala:279)

at scala.collection.mutable.ArrayOps$ofRef.map(ArrayOps.scala:198)

at org.apache.spark.sql.catalyst.OrderedFilters.$anonfun$predicates$3(OrderedFilters.scala:61)

at org.apache.spark.sql.catalyst.OrderedFilters.$anonfun$predicates$3$adapted(OrderedFilters.scala:50)

at scala.collection.immutable.List.foreach(List.scala:431)

at org.apache.spark.sql.catalyst.OrderedFilters.<init>(OrderedFilters.scala:50)

at org.apache.spark.sql.catalyst.csv.UnivocityParser.<init>(UnivocityParser.scala:103)

at org.apache.spark.sql.execution.datasources.csv.CSVFileFormat.$anonfun$buildReader$1(CSVFileFormat.scala:130)

at org.apache.spark.sql.execution.datasources.FileFormat$$anon$1.apply(FileFormat.scala:148)

at org.apache.spark.sql.execution.datasources.FileFormat$$anon$1.apply(FileFormat.scala:133)

at org.apache.spark.sql.execution.datasources.FileScanRDD$$anon$1.org$apache$spark$sql$execution$datasources$FileScanRDD$$anon$$readCurrentFile(FileScanRDD.scala:127)

at org.apache.spark.sql.execution.datasources.FileScanRDD$$anon$1.nextIterator(FileScanRDD.scala:187)

at org.apache.spark.sql.execution.datasources.FileScanRDD$$anon$1.hasNext(FileScanRDD.scala:104)

at scala.collection.Iterator$$anon$10.hasNext(Iterator.scala:460)

at org.apache.spark.sql.catalyst.expressions.GeneratedClass$GeneratedIteratorForCodegenStage1.processNext(Unknown Source)

at org.apache.spark.sql.execution.BufferedRowIterator.hasNext(BufferedRowIterator.java:43)

at org.apache.spark.sql.execution.WholeStageCodegenExec$$anon$1.hasNext(WholeStageCodegenExec.scala:759)

at org.apache.spark.sql.execution.SparkPlan.$anonfun$getByteArrayRdd$1(SparkPlan.scala:349)

at org.apache.spark.rdd.RDD.$anonfun$mapPartitionsInternal$2(RDD.scala:898)

at org.apache.spark.rdd.RDD.$anonfun$mapPartitionsInternal$2$adapted(RDD.scala:898)

at org.apache.spark.rdd.MapPartitionsRDD.compute(MapPartitionsRDD.scala:52)

at org.apache.spark.rdd.RDD.computeOrReadCheckpoint(RDD.scala:373)

at org.apache.spark.rdd.RDD.iterator(RDD.scala:337)

at org.apache.spark.scheduler.ResultTask.runTask(ResultTask.scala:90)

at org.apache.spark.scheduler.Task.run(Task.scala:131)

at org.apache.spark.executor.Executor$TaskRunner.$anonfun$run$3(Executor.scala:506)

at org.apache.spark.util.Utils$.tryWithSafeFinally(Utils.scala:1462)

at org.apache.spark.executor.Executor$TaskRunner.run(Executor.scala:509)

at java.util.concurrent.ThreadPoolExecutor.runWorker(ThreadPoolExecutor.java:1149)

at java.util.concurrent.ThreadPoolExecutor$Worker.run(ThreadPoolExecutor.java:624)

at java.lang.Thread.run(Thread.java:750)

24/07/18 21:02:02 WARN scheduler.TaskSetManager: Lost task 0.0 in stage 44.0 (TID 44) (192.168.112.129 executor driver): java.lang.IllegalArgumentException: RejectRows does not exist. Available: IssuerId, IssuerId2, BusinessDate, StateCode, SourceName, NetworkName, NetworkURL, custnum, MarketCoverage, DentalOnlyPlan

at org.apache.spark.sql.types.StructType.$anonfun$fieldIndex$1(StructType.scala:306)

at scala.collection.immutable.HashMap$HashTrieMap.getOrElse0(HashMap.scala:596)

at scala.collection.immutable.HashMap.getOrElse(HashMap.scala:73)

at org.apache.spark.sql.types.StructType.fieldIndex(StructType.scala:305)

at org.apache.spark.sql.catalyst.OrderedFilters.$anonfun$predicates$4(OrderedFilters.scala:61)

at org.apache.spark.sql.catalyst.OrderedFilters.$anonfun$predicates$4$adapted(OrderedFilters.scala:61)

at scala.collection.TraversableLike.$anonfun$map$1(TraversableLike.scala:286)

at scala.collection.IndexedSeqOptimized.foreach(IndexedSeqOptimized.scala:36)

at scala.collection.IndexedSeqOptimized.foreach$(IndexedSeqOptimized.scala:33)

at scala.collection.mutable.ArrayOps$ofRef.foreach(ArrayOps.scala:198)

at scala.collection.TraversableLike.map(TraversableLike.scala:286)

at scala.collection.TraversableLike.map$(TraversableLike.scala:279)

at scala.collection.mutable.ArrayOps$ofRef.map(ArrayOps.scala:198)

at org.apache.spark.sql.catalyst.OrderedFilters.$anonfun$predicates$3(OrderedFilters.scala:61)

at org.apache.spark.sql.catalyst.OrderedFilters.$anonfun$predicates$3$adapted(OrderedFilters.scala:50)

at scala.collection.immutable.List.foreach(List.scala:431)

at org.apache.spark.sql.catalyst.OrderedFilters.<init>(OrderedFilters.scala:50)

at org.apache.spark.sql.catalyst.csv.UnivocityParser.<init>(UnivocityParser.scala:103)

at org.apache.spark.sql.execution.datasources.csv.CSVFileFormat.$anonfun$buildReader$1(CSVFileFormat.scala:130)

at org.apache.spark.sql.execution.datasources.FileFormat$$anon$1.apply(FileFormat.scala:148)

at org.apache.spark.sql.execution.datasources.FileFormat$$anon$1.apply(FileFormat.scala:133)

at org.apache.spark.sql.execution.datasources.FileScanRDD$$anon$1.org$apache$spark$sql$execution$datasources$FileScanRDD$$anon$$readCurrentFile(FileScanRDD.scala:127)

at org.apache.spark.sql.execution.datasources.FileScanRDD$$anon$1.nextIterator(FileScanRDD.scala:187)

at org.apache.spark.sql.execution.datasources.FileScanRDD$$anon$1.hasNext(FileScanRDD.scala:104)

at scala.collection.Iterator$$anon$10.hasNext(Iterator.scala:460)

at org.apache.spark.sql.catalyst.expressions.GeneratedClass$GeneratedIteratorForCodegenStage1.processNext(Unknown Source)

at org.apache.spark.sql.execution.BufferedRowIterator.hasNext(BufferedRowIterator.java:43)

at org.apache.spark.sql.execution.WholeStageCodegenExec$$anon$1.hasNext(WholeStageCodegenExec.scala:759)

at org.apache.spark.sql.execution.SparkPlan.$anonfun$getByteArrayRdd$1(SparkPlan.scala:349)

at org.apache.spark.rdd.RDD.$anonfun$mapPartitionsInternal$2(RDD.scala:898)

at org.apache.spark.rdd.RDD.$anonfun$mapPartitionsInternal$2$adapted(RDD.scala:898)

at org.apache.spark.rdd.MapPartitionsRDD.compute(MapPartitionsRDD.scala:52)

at org.apache.spark.rdd.RDD.computeOrReadCheckpoint(RDD.scala:373)

at org.apache.spark.rdd.RDD.iterator(RDD.scala:337)

at org.apache.spark.scheduler.ResultTask.runTask(ResultTask.scala:90)

at org.apache.spark.scheduler.Task.run(Task.scala:131)

at org.apache.spark.executor.Executor$TaskRunner.$anonfun$run$3(Executor.scala:506)

at org.apache.spark.util.Utils$.tryWithSafeFinally(Utils.scala:1462)

at org.apache.spark.executor.Executor$TaskRunner.run(Executor.scala:509)

at java.util.concurrent.ThreadPoolExecutor.runWorker(ThreadPoolExecutor.java:1149)

at java.util.concurrent.ThreadPoolExecutor$Worker.run(ThreadPoolExecutor.java:624)

at java.lang.Thread.run(Thread.java:750)

24/07/18 21:02:02 ERROR scheduler.TaskSetManager: Task 0 in stage 44.0 failed 1 times; aborting job

Traceback (most recent call last):

File "<stdin>", line 1, in <module>

File "/usr/local/spark/python/pyspark/sql/dataframe.py", line 494, in show

print(self.\_jdf.showString(n, 20, vertical))

File "/usr/local/spark/python/lib/py4j-0.10.9.3-src.zip/py4j/java\_gateway.py", line 1322, in \_\_call\_\_

File "/usr/local/spark/python/pyspark/sql/utils.py", line 111, in deco

return f(\*a, \*\*kw)

File "/usr/local/spark/python/lib/py4j-0.10.9.3-src.zip/py4j/protocol.py", line 328, in get\_return\_value

py4j.protocol.Py4JJavaError: An error occurred while calling o340.showString.

: org.apache.spark.SparkException: Job aborted due to stage failure: Task 0 in stage 44.0 failed 1 times, most recent failure: Lost task 0.0 in stage 44.0 (TID 44) (192.168.112.129 executor driver): java.lang.IllegalArgumentException: RejectRows does not exist. Available: IssuerId, IssuerId2, BusinessDate, StateCode, SourceName, NetworkName, NetworkURL, custnum, MarketCoverage, DentalOnlyPlan

at org.apache.spark.sql.types.StructType.$anonfun$fieldIndex$1(StructType.scala:306)

at scala.collection.immutable.HashMap$HashTrieMap.getOrElse0(HashMap.scala:596)

at scala.collection.immutable.HashMap.getOrElse(HashMap.scala:73)

at org.apache.spark.sql.types.StructType.fieldIndex(StructType.scala:305)

at org.apache.spark.sql.catalyst.OrderedFilters.$anonfun$predicates$4(OrderedFilters.scala:61)

at org.apache.spark.sql.catalyst.OrderedFilters.$anonfun$predicates$4$adapted(OrderedFilters.scala:61)

at scala.collection.TraversableLike.$anonfun$map$1(TraversableLike.scala:286)

at scala.collection.IndexedSeqOptimized.foreach(IndexedSeqOptimized.scala:36)

at scala.collection.IndexedSeqOptimized.foreach$(IndexedSeqOptimized.scala:33)

at scala.collection.mutable.ArrayOps$ofRef.foreach(ArrayOps.scala:198)

at scala.collection.TraversableLike.map(TraversableLike.scala:286)

at scala.collection.TraversableLike.map$(TraversableLike.scala:279)

at scala.collection.mutable.ArrayOps$ofRef.map(ArrayOps.scala:198)

at org.apache.spark.sql.catalyst.OrderedFilters.$anonfun$predicates$3(OrderedFilters.scala:61)

at org.apache.spark.sql.catalyst.OrderedFilters.$anonfun$predicates$3$adapted(OrderedFilters.scala:50)

at scala.collection.immutable.List.foreach(List.scala:431)

at org.apache.spark.sql.catalyst.OrderedFilters.<init>(OrderedFilters.scala:50)

at org.apache.spark.sql.catalyst.csv.UnivocityParser.<init>(UnivocityParser.scala:103)

at org.apache.spark.sql.execution.datasources.csv.CSVFileFormat.$anonfun$buildReader$1(CSVFileFormat.scala:130)

at org.apache.spark.sql.execution.datasources.FileFormat$$anon$1.apply(FileFormat.scala:148)

at org.apache.spark.sql.execution.datasources.FileFormat$$anon$1.apply(FileFormat.scala:133)

at org.apache.spark.sql.execution.datasources.FileScanRDD$$anon$1.org$apache$spark$sql$execution$datasources$FileScanRDD$$anon$$readCurrentFile(FileScanRDD.scala:127)

at org.apache.spark.sql.execution.datasources.FileScanRDD$$anon$1.nextIterator(FileScanRDD.scala:187)

at org.apache.spark.sql.execution.datasources.FileScanRDD$$anon$1.hasNext(FileScanRDD.scala:104)

at scala.collection.Iterator$$anon$10.hasNext(Iterator.scala:460)

at org.apache.spark.sql.catalyst.expressions.GeneratedClass$GeneratedIteratorForCodegenStage1.processNext(Unknown Source)

at org.apache.spark.sql.execution.BufferedRowIterator.hasNext(BufferedRowIterator.java:43)

at org.apache.spark.sql.execution.WholeStageCodegenExec$$anon$1.hasNext(WholeStageCodegenExec.scala:759)

at org.apache.spark.sql.execution.SparkPlan.$anonfun$getByteArrayRdd$1(SparkPlan.scala:349)

at org.apache.spark.rdd.RDD.$anonfun$mapPartitionsInternal$2(RDD.scala:898)

at org.apache.spark.rdd.RDD.$anonfun$mapPartitionsInternal$2$adapted(RDD.scala:898)

at org.apache.spark.rdd.MapPartitionsRDD.compute(MapPartitionsRDD.scala:52)

at org.apache.spark.rdd.RDD.computeOrReadCheckpoint(RDD.scala:373)

at org.apache.spark.rdd.RDD.iterator(RDD.scala:337)

at org.apache.spark.scheduler.ResultTask.runTask(ResultTask.scala:90)

at org.apache.spark.scheduler.Task.run(Task.scala:131)

at org.apache.spark.executor.Executor$TaskRunner.$anonfun$run$3(Executor.scala:506)

at org.apache.spark.util.Utils$.tryWithSafeFinally(Utils.scala:1462)

at org.apache.spark.executor.Executor$TaskRunner.run(Executor.scala:509)

at java.util.concurrent.ThreadPoolExecutor.runWorker(ThreadPoolExecutor.java:1149)

at java.util.concurrent.ThreadPoolExecutor$Worker.run(ThreadPoolExecutor.java:624)

at java.lang.Thread.run(Thread.java:750)

Driver stacktrace:

at org.apache.spark.scheduler.DAGScheduler.failJobAndIndependentStages(DAGScheduler.scala:2454)

at org.apache.spark.scheduler.DAGScheduler.$anonfun$abortStage$2(DAGScheduler.scala:2403)

at org.apache.spark.scheduler.DAGScheduler.$anonfun$abortStage$2$adapted(DAGScheduler.scala:2402)

at scala.collection.mutable.ResizableArray.foreach(ResizableArray.scala:62)

at scala.collection.mutable.ResizableArray.foreach$(ResizableArray.scala:55)

at scala.collection.mutable.ArrayBuffer.foreach(ArrayBuffer.scala:49)

at org.apache.spark.scheduler.DAGScheduler.abortStage(DAGScheduler.scala:2402)

at org.apache.spark.scheduler.DAGScheduler.$anonfun$handleTaskSetFailed$1(DAGScheduler.scala:1160)

at org.apache.spark.scheduler.DAGScheduler.$anonfun$handleTaskSetFailed$1$adapted(DAGScheduler.scala:1160)

at scala.Option.foreach(Option.scala:407)

at org.apache.spark.scheduler.DAGScheduler.handleTaskSetFailed(DAGScheduler.scala:1160)

at org.apache.spark.scheduler.DAGSchedulerEventProcessLoop.doOnReceive(DAGScheduler.scala:2642)

at org.apache.spark.scheduler.DAGSchedulerEventProcessLoop.onReceive(DAGScheduler.scala:2584)

at org.apache.spark.scheduler.DAGSchedulerEventProcessLoop.onReceive(DAGScheduler.scala:2573)

at org.apache.spark.util.EventLoop$$anon$1.run(EventLoop.scala:49)

at org.apache.spark.scheduler.DAGScheduler.runJob(DAGScheduler.scala:938)

at org.apache.spark.SparkContext.runJob(SparkContext.scala:2214)

at org.apache.spark.SparkContext.runJob(SparkContext.scala:2235)

at org.apache.spark.SparkContext.runJob(SparkContext.scala:2254)

at org.apache.spark.sql.execution.SparkPlan.executeTake(SparkPlan.scala:476)

at org.apache.spark.sql.execution.SparkPlan.executeTake(SparkPlan.scala:429)

at org.apache.spark.sql.execution.CollectLimitExec.executeCollect(limit.scala:48)

at org.apache.spark.sql.Dataset.collectFromPlan(Dataset.scala:3715)

at org.apache.spark.sql.Dataset.$anonfun$head$1(Dataset.scala:2728)

at org.apache.spark.sql.Dataset.$anonfun$withAction$1(Dataset.scala:3706)

at org.apache.spark.sql.execution.SQLExecution$.$anonfun$withNewExecutionId$5(SQLExecution.scala:103)

at org.apache.spark.sql.execution.SQLExecution$.withSQLConfPropagated(SQLExecution.scala:163)

at org.apache.spark.sql.execution.SQLExecution$.$anonfun$withNewExecutionId$1(SQLExecution.scala:90)

at org.apache.spark.sql.SparkSession.withActive(SparkSession.scala:775)

at org.apache.spark.sql.execution.SQLExecution$.withNewExecutionId(SQLExecution.scala:64)

at org.apache.spark.sql.Dataset.withAction(Dataset.scala:3704)

at org.apache.spark.sql.Dataset.head(Dataset.scala:2728)

at org.apache.spark.sql.Dataset.take(Dataset.scala:2935)

at org.apache.spark.sql.Dataset.getRows(Dataset.scala:287)

at org.apache.spark.sql.Dataset.showString(Dataset.scala:326)

at sun.reflect.GeneratedMethodAccessor41.invoke(Unknown Source)

at sun.reflect.DelegatingMethodAccessorImpl.invoke(DelegatingMethodAccessorImpl.java:43)

at java.lang.reflect.Method.invoke(Method.java:498)

at py4j.reflection.MethodInvoker.invoke(MethodInvoker.java:244)

at py4j.reflection.ReflectionEngine.invoke(ReflectionEngine.java:357)

at py4j.Gateway.invoke(Gateway.java:282)

at py4j.commands.AbstractCommand.invokeMethod(AbstractCommand.java:132)

at py4j.commands.CallCommand.execute(CallCommand.java:79)

at py4j.ClientServerConnection.waitForCommands(ClientServerConnection.java:182)

at py4j.ClientServerConnection.run(ClientServerConnection.java:106)

at java.lang.Thread.run(Thread.java:750)

Caused by: java.lang.IllegalArgumentException: RejectRows does not exist. Available: IssuerId, IssuerId2, BusinessDate, StateCode, SourceName, NetworkName, NetworkURL, custnum, MarketCoverage, DentalOnlyPlan

at org.apache.spark.sql.types.StructType.$anonfun$fieldIndex$1(StructType.scala:306)

at scala.collection.immutable.HashMap$HashTrieMap.getOrElse0(HashMap.scala:596)

at scala.collection.immutable.HashMap.getOrElse(HashMap.scala:73)

at org.apache.spark.sql.types.StructType.fieldIndex(StructType.scala:305)

at org.apache.spark.sql.catalyst.OrderedFilters.$anonfun$predicates$4(OrderedFilters.scala:61)

at org.apache.spark.sql.catalyst.OrderedFilters.$anonfun$predicates$4$adapted(OrderedFilters.scala:61)

at scala.collection.TraversableLike.$anonfun$map$1(TraversableLike.scala:286)

at scala.collection.IndexedSeqOptimized.foreach(IndexedSeqOptimized.scala:36)

at scala.collection.IndexedSeqOptimized.foreach$(IndexedSeqOptimized.scala:33)

at scala.collection.mutable.ArrayOps$ofRef.foreach(ArrayOps.scala:198)

at scala.collection.TraversableLike.map(TraversableLike.scala:286)

at scala.collection.TraversableLike.map$(TraversableLike.scala:279)

at scala.collection.mutable.ArrayOps$ofRef.map(ArrayOps.scala:198)

at org.apache.spark.sql.catalyst.OrderedFilters.$anonfun$predicates$3(OrderedFilters.scala:61)

at org.apache.spark.sql.catalyst.OrderedFilters.$anonfun$predicates$3$adapted(OrderedFilters.scala:50)

at scala.collection.immutable.List.foreach(List.scala:431)

at org.apache.spark.sql.catalyst.OrderedFilters.<init>(OrderedFilters.scala:50)

at org.apache.spark.sql.catalyst.csv.UnivocityParser.<init>(UnivocityParser.scala:103)

at org.apache.spark.sql.execution.datasources.csv.CSVFileFormat.$anonfun$buildReader$1(CSVFileFormat.scala:130)

at org.apache.spark.sql.execution.datasources.FileFormat$$anon$1.apply(FileFormat.scala:148)

at org.apache.spark.sql.execution.datasources.FileFormat$$anon$1.apply(FileFormat.scala:133)

at org.apache.spark.sql.execution.datasources.FileScanRDD$$anon$1.org$apache$spark$sql$execution$datasources$FileScanRDD$$anon$$readCurrentFile(FileScanRDD.scala:127)

at org.apache.spark.sql.execution.datasources.FileScanRDD$$anon$1.nextIterator(FileScanRDD.scala:187)

at org.apache.spark.sql.execution.datasources.FileScanRDD$$anon$1.hasNext(FileScanRDD.scala:104)

at scala.collection.Iterator$$anon$10.hasNext(Iterator.scala:460)

at org.apache.spark.sql.catalyst.expressions.GeneratedClass$GeneratedIteratorForCodegenStage1.processNext(Unknown Source)

at org.apache.spark.sql.execution.BufferedRowIterator.hasNext(BufferedRowIterator.java:43)

at org.apache.spark.sql.execution.WholeStageCodegenExec$$anon$1.hasNext(WholeStageCodegenExec.scala:759)

at org.apache.spark.sql.execution.SparkPlan.$anonfun$getByteArrayRdd$1(SparkPlan.scala:349)

at org.apache.spark.rdd.RDD.$anonfun$mapPartitionsInternal$2(RDD.scala:898)

at org.apache.spark.rdd.RDD.$anonfun$mapPartitionsInternal$2$adapted(RDD.scala:898)

at org.apache.spark.rdd.MapPartitionsRDD.compute(MapPartitionsRDD.scala:52)

at org.apache.spark.rdd.RDD.computeOrReadCheckpoint(RDD.scala:373)

at org.apache.spark.rdd.RDD.iterator(RDD.scala:337)

at org.apache.spark.scheduler.ResultTask.runTask(ResultTask.scala:90)

at org.apache.spark.scheduler.Task.run(Task.scala:131)

at org.apache.spark.executor.Executor$TaskRunner.$anonfun$run$3(Executor.scala:506)

at org.apache.spark.util.Utils$.tryWithSafeFinally(Utils.scala:1462)

at org.apache.spark.executor.Executor$TaskRunner.run(Executor.scala:509)

at java.util.concurrent.ThreadPoolExecutor.runWorker(ThreadPoolExecutor.java:1149)

at java.util.concurrent.ThreadPoolExecutor$Worker.run(ThreadPoolExecutor.java:624)

... 1 more

>>> if 'RejectRows' in data\_frame\_1.columns:

... print("Column 'RejectRows' exists.")

... else:

... print("Column 'RejectRows' does not exist.")

...

Column 'RejectRows' exists.

>>>

>>> data\_frame\_1.cache()

DataFrame[IssuerId: int, IssuerId2: int, BusinessDate: date, StateCode: string, SourceName: string, NetworkName: string, NetworkURL: string, custnum: string, MarketCoverage: string, DentalOnlyPlan: string, RejectRows: string]

>>> data\_frame\_1.where(col("RejectRows").isNotNull()).show(3)

+--------+---------+------------+---------+----------+-----------+----------+-------+--------------+--------------+--------------------+

|IssuerId|IssuerId2|BusinessDate|StateCode|SourceName|NetworkName|NetworkURL|custnum|MarketCoverage|DentalOnlyPlan| RejectRows|

+--------+---------+------------+---------+----------+-----------+----------+-------+--------------+--------------+--------------------+

| 219| 219| 2019-10-01| AK| HIOS|ODS Premier| null| null| null| null|219,219,2019-10-0...|

| null| null| null| null| null| null| null| null| null| null| footer count is 402|

+--------+---------+------------+---------+----------+-----------+----------+-------+--------------+--------------+--------------------+

>>> data\_frame\_1.where("RejectRows!=`null`").show(3)

Traceback (most recent call last):

File "<stdin>", line 1, in <module>

File "/usr/local/spark/python/pyspark/sql/dataframe.py", line 1731, in filter

jdf = self.\_jdf.filter(condition)

File "/usr/local/spark/python/lib/py4j-0.10.9.3-src.zip/py4j/java\_gateway.py", line 1322, in \_\_call\_\_

File "/usr/local/spark/python/pyspark/sql/utils.py", line 117, in deco

raise converted from None

pyspark.sql.utils.AnalysisException: cannot resolve 'null' given input columns: [BusinessDate, DentalOnlyPlan, IssuerId, IssuerId2, MarketCoverage, NetworkName, NetworkURL, RejectRows, SourceName, StateCode, custnum]; line 1 pos 12;

'Filter NOT (RejectRows#1386 = 'null)

+- Relation [IssuerId#1376,IssuerId2#1377,BusinessDate#1378,StateCode#1379,SourceName#1380,NetworkName#1381,NetworkURL#1382,custnum#1383,MarketCoverage#1384,DentalOnlyPlan#1385,RejectRows#1386] csv

>>> data\_frame\_1.where("RejectRows!='null'").show(3)

+--------+---------+------------+---------+----------+-----------+----------+-------+--------------+--------------+--------------------+

|IssuerId|IssuerId2|BusinessDate|StateCode|SourceName|NetworkName|NetworkURL|custnum|MarketCoverage|DentalOnlyPlan| RejectRows|

+--------+---------+------------+---------+----------+-----------+----------+-------+--------------+--------------+--------------------+

| 219| 219| 2019-10-01| AK| HIOS|ODS Premier| null| null| null| null|219,219,2019-10-0...|

| null| null| null| null| null| null| null| null| null| null| footer count is 402|

+--------+---------+------------+---------+----------+-----------+----------+-------+--------------+--------------+--------------------+

>>> data\_frame\_3.cache()

DataFrame[IssuerId: int, IssuerId2: int, BusinessDate: date, StateCode: string, SourceName: string, NetworkName: string, NetworkURL: string, custnum: string, MarketCoverage: string, DentalOnlyPlan: string, RejectRows: string]

>>> data\_frame\_3.where("RejectRows is not NULL ").show(3)

+--------+---------+------------+---------+----------+-----------+----------+-------+------------------+--------------+--------------------+

|IssuerId|IssuerId2|BusinessDate|StateCode|SourceName|NetworkName|NetworkURL|custnum| MarketCoverage|DentalOnlyPlan| RejectRows|

+--------+---------+------------+---------+----------+-----------+----------+-------+------------------+--------------+--------------------+

| null| null| null| null| null| null| null| 13|SHOP (Small Group)| null|,,,,,,,13,SHOP (S...|

| null| null| null| null| null| null| null| null| null| null|footer count is 3334|

+--------+---------+------------+---------+----------+-----------+----------+-------+------------------+--------------+--------------------+

>>> data\_frame\_3.where("RejectRows !='NULL' ").show(3)

+--------+---------+------------+---------+----------+-----------+----------+-------+------------------+--------------+--------------------+

|IssuerId|IssuerId2|BusinessDate|StateCode|SourceName|NetworkName|NetworkURL|custnum| MarketCoverage|DentalOnlyPlan| RejectRows|

+--------+---------+------------+---------+----------+-----------+----------+-------+------------------+--------------+--------------------+

| null| null| null| null| null| null| null| 13|SHOP (Small Group)| null|,,,,,,,13,SHOP (S...|

| null| null| null| null| null| null| null| null| null| null|footer count is 3334|

+--------+---------+------------+---------+----------+-----------+----------+-------+------------------+--------------+--------------------+

>>> rejected\_data\_frame=data\_frame\_3.where("RejectRows !='NULL' ")

>>> rejected\_data\_frame.show()

+--------+---------+------------+---------+----------+-----------+----------+-------+------------------+--------------+--------------------+

|IssuerId|IssuerId2|BusinessDate|StateCode|SourceName|NetworkName|NetworkURL|custnum| MarketCoverage|DentalOnlyPlan| RejectRows|

+--------+---------+------------+---------+----------+-----------+----------+-------+------------------+--------------+--------------------+

| null| null| null| null| null| null| null| 13|SHOP (Small Group)| null|,,,,,,,13,SHOP (S...|

| null| null| null| null| null| null| null| null| null| null|footer count is 3334|

+--------+---------+------------+---------+----------+-----------+----------+-------+------------------+--------------+--------------------+

>>> data\_frame\_1.where("RejectRows !='NULL' ").show(3)

+--------+---------+------------+---------+----------+-----------+----------+-------+--------------+--------------+--------------------+

|IssuerId|IssuerId2|BusinessDate|StateCode|SourceName|NetworkName|NetworkURL|custnum|MarketCoverage|DentalOnlyPlan| RejectRows|

+--------+---------+------------+---------+----------+-----------+----------+-------+--------------+--------------+--------------------+

| 219| 219| 2019-10-01| AK| HIOS|ODS Premier| null| null| null| null|219,219,2019-10-0...|

| null| null| null| null| null| null| null| null| null| null| footer count is 402|

+--------+---------+------------+---------+----------+-----------+----------+-------+--------------+--------------+--------------------+

>>> data\_frame\_1.where("RejectRows !='NULL' ").show(20)

+--------+---------+------------+---------+----------+-----------+----------+-------+--------------+--------------+--------------------+

|IssuerId|IssuerId2|BusinessDate|StateCode|SourceName|NetworkName|NetworkURL|custnum|MarketCoverage|DentalOnlyPlan| RejectRows|

+--------+---------+------------+---------+----------+-----------+----------+-------+--------------+--------------+--------------------+

| 219| 219| 2019-10-01| AK| HIOS|ODS Premier| null| null| null| null|219,219,2019-10-0...|

| null| null| null| null| null| null| null| null| null| null| footer count is 402|

+--------+---------+------------+---------+----------+-----------+----------+-------+--------------+--------------+--------------------+

>>> rejected\_data\_frame\_1=data\_frame\_1.where("RejectRows !='NULL' ")

>>> rejected\_data\_frame\_1.show()

+--------+---------+------------+---------+----------+-----------+----------+-------+--------------+--------------+--------------------+

|IssuerId|IssuerId2|BusinessDate|StateCode|SourceName|NetworkName|NetworkURL|custnum|MarketCoverage|DentalOnlyPlan| RejectRows|

+--------+---------+------------+---------+----------+-----------+----------+-------+--------------+--------------+--------------------+

| 219| 219| 2019-10-01| AK| HIOS|ODS Premier| null| null| null| null|219,219,2019-10-0...|

| null| null| null| null| null| null| null| null| null| null| footer count is 402|

+--------+---------+------------+---------+----------+-----------+----------+-------+--------------+--------------+--------------------+

>>> rejected\_data\_frame\_1.show()

>>> merged\_data\_frame=data\_frame\_1.union(data\_frame\_3)

>>> merged\_data\_frame.show(3)

+--------+---------+------------+---------+----------+------------+--------------------+-------+--------------+--------------+--------------------+

|IssuerId|IssuerId2|BusinessDate|StateCode|SourceName| NetworkName| NetworkURL|custnum|MarketCoverage|DentalOnlyPlan| RejectRows|

+--------+---------+------------+---------+----------+------------+--------------------+-------+--------------+--------------+--------------------+

| 21989| 21989| 2019-10-01| AK| HIOS| ODS Premier|https://www.modah...| 13| null| null| null|

| 219| 219| 2019-10-01| AK| HIOS| ODS Premier| null| null| null| null|219,219,2019-10-0...|

| 38344| 38344| 2019-10-01| AK| HIOS|HeritagePlus|https://www.preme...| 13| null| null| null|

+--------+---------+------------+---------+----------+------------+--------------------+-------+--------------+--------------+--------------------+

only showing top 3 rows

>>> merged\_data\_frame.cache()

DataFrame[IssuerId: int, IssuerId2: int, BusinessDate: date, StateCode: string, SourceName: string, NetworkName: string, NetworkURL: string, custnum: string, MarketCoverage: string, DentalOnlyPlan: string, RejectRows: string]

>>> merged\_data\_frame.where("RejectRows is not null ")

DataFrame[IssuerId: int, IssuerId2: int, BusinessDate: date, StateCode: string, SourceName: string, NetworkName: string, NetworkURL: string, custnum: string, MarketCoverage: string, DentalOnlyPlan: string, RejectRows: string]

>>> merged\_data\_frame.where("RejectRows is not null ").show()

+--------+---------+------------+---------+----------+-----------+----------+-------+------------------+--------------+--------------------+

|IssuerId|IssuerId2|BusinessDate|StateCode|SourceName|NetworkName|NetworkURL|custnum| MarketCoverage|DentalOnlyPlan| RejectRows|

+--------+---------+------------+---------+----------+-----------+----------+-------+------------------+--------------+--------------------+

| 219| 219| 2019-10-01| AK| HIOS|ODS Premier| null| null| null| null|219,219,2019-10-0...|

| null| null| null| null| null| null| null| null| null| null| footer count is 402|

| null| null| null| null| null| null| null| 13|SHOP (Small Group)| null|,,,,,,,13,SHOP (S...|

| null| null| null| null| null| null| null| null| null| null|footer count is 3334|

+--------+---------+------------+---------+----------+-----------+----------+-------+------------------+--------------+--------------------+

>>> merged\_data\_frame.unpersist(True)

DataFrame[IssuerId: int, IssuerId2: int, BusinessDate: date, StateCode: string, SourceName: string, NetworkName: string, NetworkURL: string, custnum: string, MarketCoverage: string, DentalOnlyPlan: string, RejectRows: string]

>>> merged\_data\_frame.where("RejectRows is not null ")

DataFrame[IssuerId: int, IssuerId2: int, BusinessDate: date, StateCode: string, SourceName: string, NetworkName: string, NetworkURL: string, custnum: string, MarketCoverage: string, DentalOnlyPlan: string, RejectRows: string]

>>> merged\_data\_frame.where("RejectRows is not null ").show()

+--------+---------+------------+---------+----------+-----------+----------+-------+------------------+--------------+--------------------+

|IssuerId|IssuerId2|BusinessDate|StateCode|SourceName|NetworkName|NetworkURL|custnum| MarketCoverage|DentalOnlyPlan| RejectRows|

+--------+---------+------------+---------+----------+-----------+----------+-------+------------------+--------------+--------------------+

| 219| 219| 2019-10-01| AK| HIOS|ODS Premier| null| null| null| null|219,219,2019-10-0...|

| null| null| null| null| null| null| null| null| null| null| footer count is 402|

| null| null| null| null| null| null| null| 13|SHOP (Small Group)| null|,,,,,,,13,SHOP (S...|

| null| null| null| null| null| null| null| null| null| null|footer count is 3334|

+--------+---------+------------+---------+----------+-----------+----------+-------+------------------+--------------+--------------------+

>>>

>>> merged\_data\_frame.where("RejectRows is not null ").show()

+--------+---------+------------+---------+----------+-----------+----------+-------+------------------+--------------+--------------------+

|IssuerId|IssuerId2|BusinessDate|StateCode|SourceName|NetworkName|NetworkURL|custnum| MarketCoverage|DentalOnlyPlan| RejectRows|

+--------+---------+------------+---------+----------+-----------+----------+-------+------------------+--------------+--------------------+

| 219| 219| 2019-10-01| AK| HIOS|ODS Premier| null| null| null| null|219,219,2019-10-0...|

| null| null| null| null| null| null| null| null| null| null| footer count is 402|

| null| null| null| null| null| null| null| 13|SHOP (Small Group)| null|,,,,,,,13,SHOP (S...|

| null| null| null| null| null| null| null| null| null| null|footer count is 3334|

+--------+---------+------------+---------+----------+-----------+----------+-------+------------------+--------------+--------------------+

>>> merged\_data\_frame.withColumnRenamed("StateCode","stcd")

DataFrame[IssuerId: int, IssuerId2: int, BusinessDate: date, stcd: string, SourceName: string, NetworkName: string, NetworkURL: string, custnum: string, MarketCoverage: string, DentalOnlyPlan: string, RejectRows: string]

>>> merged\_data\_frame.withColumnRenamed("SourceName","srcnm")

DataFrame[IssuerId: int, IssuerId2: int, BusinessDate: date, StateCode: string, srcnm: string, NetworkName: string, NetworkURL: string, custnum: string, MarketCoverage: string, DentalOnlyPlan: string, RejectRows: string]

>>> merged\_data\_frame=merged\_data\_frame.withColumnRenamed("SourceName","srcnm").withColumnRenamed("StateCode","stcd")

>>> merged\_data\_frame.show(3)

+--------+---------+------------+----+-----+------------+--------------------+-------+--------------+--------------+--------------------+

|IssuerId|IssuerId2|BusinessDate|stcd|srcnm| NetworkName| NetworkURL|custnum|MarketCoverage|DentalOnlyPlan| RejectRows|

+--------+---------+------------+----+-----+------------+--------------------+-------+--------------+--------------+--------------------+

| 21989| 21989| 2019-10-01| AK| HIOS| ODS Premier|https://www.modah...| 13| null| null| null|

| 219| 219| 2019-10-01| AK| HIOS| ODS Premier| null| null| null| null|219,219,2019-10-0...|

| 38344| 38344| 2019-10-01| AK| HIOS|HeritagePlus|https://www.preme...| 13| null| null| null|

+--------+---------+------------+----+-----+------------+--------------------+-------+--------------+--------------+--------------------+

only showing top 3 rows

>>> merged\_data\_frame=merged\_data\_frame.withColumnRenamed("IssuerId","IssuerIDD").withColumnRenamed("IssuerId2","IssuerIDDD")

>>> merged\_data\_frame.withColumn("IssuerId" , col("IssuerIDD").cast("string")).withColumn("IssuerId2" , col("IssuerIDDD").cast("string"))

DataFrame[IssuerIDD: int, IssuerIDDD: int, BusinessDate: date, stcd: string, srcnm: string, NetworkName: string, NetworkURL: string, custnum: string, MarketCoverage: string, DentalOnlyPlan: string, RejectRows: string, IssuerId: string, IssuerId2: string]

>>> merged\_data\_frame=merged\_data\_frame.withColumn("IssuerId" , col("IssuerIDD").cast("string")).withColumn("IssuerId2" , col("IssuerIDDD").cast("string"))

>>> merged\_data\_frame.dropColumn("IssuerIDD").dropColumn("IssuerIDDD")

Traceback (most recent call last):

File "<stdin>", line 1, in <module>

File "/usr/local/spark/python/pyspark/sql/dataframe.py", line 1660, in \_\_getattr\_\_

"'%s' object has no attribute '%s'" % (self.\_\_class\_\_.\_\_name\_\_, name))

AttributeError: 'DataFrame' object has no attribute 'dropColumn'

>>> merged\_data\_frame.drop("IssuerIDD").drop("IssuerIDDD")

DataFrame[BusinessDate: date, stcd: string, srcnm: string, NetworkName: string, NetworkURL: string, custnum: string, MarketCoverage: string, DentalOnlyPlan: string, RejectRows: string, IssuerId: string, IssuerId2: string]

>>> merged\_data\_frame=merged\_data\_frame.drop("IssuerIDD").drop("IssuerIDDD")

>>> merged\_data\_frame.show(3)

+------------+----+-----+------------+--------------------+-------+--------------+--------------+--------------------+--------+---------+

|BusinessDate|stcd|srcnm| NetworkName| NetworkURL|custnum|MarketCoverage|DentalOnlyPlan| RejectRows|IssuerId|IssuerId2|

+------------+----+-----+------------+--------------------+-------+--------------+--------------+--------------------+--------+---------+

| 2019-10-01| AK| HIOS| ODS Premier|https://www.modah...| 13| null| null| null| 21989| 21989|

| 2019-10-01| AK| HIOS| ODS Premier| null| null| null| null|219,219,2019-10-0...| 219| 219|

| 2019-10-01| AK| HIOS|HeritagePlus|https://www.preme...| 13| null| null| null| 38344| 38344|

+------------+----+-----+------------+--------------------+-------+--------------+--------------+--------------------+--------+---------+

only showing top 3 rows

>>> merged\_data\_frame.withColumn("issueridcomposite",col("IssuerId")+col("IssuerId2"))

DataFrame[BusinessDate: date, stcd: string, srcnm: string, NetworkName: string, NetworkURL: string, custnum: string, MarketCoverage: string, DentalOnlyPlan: string, RejectRows: string, IssuerId: string, IssuerId2: string, issueridcomposite: double]

>>> merged\_data\_frame.withColumn("issueridcomposite",concat(col("IssuerId"),col("IssuerId2")))

Traceback (most recent call last):

File "<stdin>", line 1, in <module>

NameError: name 'concat' is not defined

>>> merged\_data\_frame.withColumn("issueridcomposite",concat(col("IssuerId")+col("IssuerId2")))

Traceback (most recent call last):

File "<stdin>", line 1, in <module>

NameError: name 'concat' is not defined

>>> merged\_data\_frame.withColumn("issueridcomposite",concat(col("IssuerId"),col("IssuerId2")))

Traceback (most recent call last):

File "<stdin>", line 1, in <module>

NameError: name 'concat' is not defined

>>> from pyspark.sql.functions import concat, col

>>> merged\_data\_frame.withColumn("issueridcomposite",concat(col("IssuerId"),col("IssuerId2")))

DataFrame[BusinessDate: date, stcd: string, srcnm: string, NetworkName: string, NetworkURL: string, custnum: string, MarketCoverage: string, DentalOnlyPlan: string, RejectRows: string, IssuerId: string, IssuerId2: string, issueridcomposite: string]

>>> merged\_data\_frame=merged\_data\_frame.withColumn("issueridcomposite",concat(col("IssuerId"),col("IssuerId2")))

>>> merged\_data\_frame.drop("DentalOnlyPlan")

DataFrame[BusinessDate: date, stcd: string, srcnm: string, NetworkName: string, NetworkURL: string, custnum: string, MarketCoverage: string, RejectRows: string, IssuerId: string, IssuerId2: string, issueridcomposite: string]

>>> merged\_data\_frame=merged\_data\_frame.drop("DentalOnlyPlan")

>>> merged\_data\_frame.withColumn("sysdt" , current\_date()).withColumn("systs" , current\_time\_stamp())

Traceback (most recent call last):

File "<stdin>", line 1, in <module>

NameError: name 'current\_date' is not defined

>>> from pyspark.sql.functions import \*

>>> merged\_data\_frame.withColumn("sysdt" , current\_date()).withColumn("systs" , current\_time\_stamp())

Traceback (most recent call last):

File "<stdin>", line 1, in <module>

NameError: name 'current\_time\_stamp' is not defined

>>> merged\_data\_frame.withColumn("sysdt" , current\_date()).withColumn("systs" , current\_timestamp())

DataFrame[BusinessDate: date, stcd: string, srcnm: string, NetworkName: string, NetworkURL: string, custnum: string, MarketCoverage: string, RejectRows: string, IssuerId: string, IssuerId2: string, issueridcomposite: string, sysdt: date, systs: timestamp]

>>> from pyspark.sql.functions import \*

>>> merged\_data\_frame.withColumn("sysdt" , current\_date()).withColumn("systs" , current\_timestamp())

DataFrame[BusinessDate: date, stcd: string, srcnm: string, NetworkName: string, NetworkURL: string, custnum: string, MarketCoverage: string, RejectRows: string, IssuerId: string, IssuerId2: string, issueridcomposite: string, sysdt: date, systs: timestamp]

>>> merged\_data\_frame=merged\_data\_frame.withColumn("sysdt" , current\_date()).withColumn("systs" , current\_timestamp())

Day 2

>>> from pyspark.sql.functions import \*

>>> merged\_data\_frame.withColumn("sysdt" , current\_date()).withColumn("systs" , current\_timestamp())

DataFrame[BusinessDate: date, stcd: string, srcnm: string, NetworkName: string, NetworkURL: string, custnum: string, MarketCoverage: string, RejectRows: string, IssuerId: string, IssuerId2: string, issueridcomposite: string, sysdt: date, systs: timestamp]

>>> merged\_data\_frame=merged\_data\_frame.withColumn("sysdt" , current\_date()).withColumn("systs" , current\_timestamp())

>>> list=merged\_data\_frame.columnn()

Traceback (most recent call last):

File "<stdin>", line 1, in <module>

File "/usr/local/spark/python/pyspark/sql/dataframe.py", line 1660, in \_\_getattr\_\_

"'%s' object has no attribute '%s'" % (self.\_\_class\_\_.\_\_name\_\_, name))

AttributeError: 'DataFrame' object has no attribute 'columnn'

>>> list=merged\_data\_frame.columnn

Traceback (most recent call last):

File "<stdin>", line 1, in <module>

File "/usr/local/spark/python/pyspark/sql/dataframe.py", line 1660, in \_\_getattr\_\_

"'%s' object has no attribute '%s'" % (self.\_\_class\_\_.\_\_name\_\_, name))

AttributeError: 'DataFrame' object has no attribute 'columnn'

>>> list=merged\_data\_frame.column

Traceback (most recent call last):

File "<stdin>", line 1, in <module>

File "/usr/local/spark/python/pyspark/sql/dataframe.py", line 1660, in \_\_getattr\_\_

"'%s' object has no attribute '%s'" % (self.\_\_class\_\_.\_\_name\_\_, name))

AttributeError: 'DataFrame' object has no attribute 'column'

>>> list=merged\_data\_frame.column()

Traceback (most recent call last):

File "<stdin>", line 1, in <module>

File "/usr/local/spark/python/pyspark/sql/dataframe.py", line 1660, in \_\_getattr\_\_

"'%s' object has no attribute '%s'" % (self.\_\_class\_\_.\_\_name\_\_, name))

AttributeError: 'DataFrame' object has no attribute 'column'

>>> header\_list=merged\_data\_frame.column.toList()

Traceback (most recent call last):

File "<stdin>", line 1, in <module>

File "/usr/local/spark/python/pyspark/sql/dataframe.py", line 1660, in \_\_getattr\_\_

"'%s' object has no attribute '%s'" % (self.\_\_class\_\_.\_\_name\_\_, name))

AttributeError: 'DataFrame' object has no attribute 'column'

>>> header\_list=merged\_data\_frame.columns.toList()

Traceback (most recent call last):

File "<stdin>", line 1, in <module>

AttributeError: 'list' object has no attribute 'toList'

>>> header\_list=merged\_data\_frame.columns.tolist()

Traceback (most recent call last):

File "<stdin>", line 1, in <module>

AttributeError: 'list' object has no attribute 'tolist'

>>> header\_list=merged\_data\_frame.columns.toList

Traceback (most recent call last):

File "<stdin>", line 1, in <module>

AttributeError: 'list' object has no attribute 'toList'

>>> header\_list=merged\_data\_frame.columns.toList

Traceback (most recent call last):

File "<stdin>", line 1, in <module>

AttributeError: 'list' object has no attribute 'toList'

>>> header\_list : List(String) =merged\_data\_frame.columns.toList

Traceback (most recent call last):

File "<stdin>", line 1, in <module>

AttributeError: 'list' object has no attribute 'toList'

>>> header\_list=merged\_data\_frame.columns.toList

Traceback (most recent call last):

File "<stdin>", line 1, in <module>

AttributeError: 'list' object has no attribute 'toList'

>>> header\_list=merged\_data\_frame.columns

>>> print(header\_list)

['BusinessDate', 'stcd', 'srcnm', 'NetworkName', 'NetworkURL', 'custnum', 'MarketCoverage', 'RejectRows', 'IssuerId', 'IssuerId2', 'issueridcomposite', 'sysdt', 'systs']

>>> header\_with\_dtypes\_list=merged\_data\_frame.dtypes

>>> print(header\_with\_dtypes)

Traceback (most recent call last):

File "<stdin>", line 1, in <module>

NameError: name 'header\_with\_dtypes' is not defined

>>> print(header\_with\_dtypes\_list)

[('BusinessDate', 'date'), ('stcd', 'string'), ('srcnm', 'string'), ('NetworkName', 'string'), ('NetworkURL', 'string'), ('custnum', 'string'), ('MarketCoverage', 'string'), ('RejectRows', 'string'), ('IssuerId', 'string'), ('IssuerId2', 'string'), ('issueridcomposite', 'string'), ('sysdt', 'date'), ('systs', 'timestamp')]

>>> header\_with\_dtypes\_list=merged\_data\_frame.dtypes

>>> print(header\_with\_dtypes\_list)

[('BusinessDate', 'date'), ('stcd', 'string'), ('srcnm', 'string'), ('NetworkName', 'string'), ('NetworkURL', 'string'), ('custnum', 'string'), ('MarketCoverage', 'string'), ('RejectRows', 'string'), ('IssuerId', 'string'), ('IssuerId2', 'string'), ('issueridcomposite', 'string'), ('sysdt', 'date'), ('systs', 'timestamp')]

>>> header\_with\_dtypes\_list=data\_frame\_1.dtypes

>>> print(header\_with\_dtypes\_list)

[('IssuerId', 'int'), ('IssuerId2', 'int'), ('BusinessDate', 'date'), ('StateCode', 'string'), ('SourceName', 'string'), ('NetworkName', 'string'), ('NetworkURL', 'string'), ('custnum', 'string'), ('MarketCoverage', 'string'), ('DentalOnlyPlan', 'string'), ('RejectRows', 'string')]

>>> def function(arr):

... for i in range (arr):

... for j in range i :

File "<stdin>", line 3

for j in range i :

^

SyntaxError: invalid syntax

>>> def function(arr):

... for i in range (arr):

... for j in i :

... if j=='int':

... integer\_list.append(i)

...

...

>>> integer\_list=[]

>>> function(header\_with\_dtypes\_list)

Traceback (most recent call last):

File "<stdin>", line 1, in <module>

File "<stdin>", line 2, in function

TypeError: 'list' object cannot be interpreted as an integer

>>> def function(arr):

... for i in (arr):

... for j in i :

... if j=='int':

... integer\_list.append(i)

...

>>> function(header\_with\_dtypes\_list)

>>> print(integer\_list)

[('IssuerId', 'int'), ('IssuerId2', 'int')]

>>> data\_frame\_1.select(\*).show(3)

File "<stdin>", line 1

data\_frame\_1.select(\*).show(3)

^

SyntaxError: invalid syntax

>>> data\_frame\_1.select("\*").show(3)

+--------+---------+------------+---------+----------+------------+--------------------+-------+--------------+--------------+--------------------+

|IssuerId|IssuerId2|BusinessDate|StateCode|SourceName| NetworkName| NetworkURL|custnum|MarketCoverage|DentalOnlyPlan| RejectRows|

+--------+---------+------------+---------+----------+------------+--------------------+-------+--------------+--------------+--------------------+

| 21989| 21989| 2019-10-01| AK| HIOS| ODS Premier|https://www.modah...| 13| null| null| null|

| 219| 219| 2019-10-01| AK| HIOS| ODS Premier| null| null| null| null|219,219,2019-10-0...|

| 38344| 38344| 2019-10-01| AK| HIOS|HeritagePlus|https://www.preme...| 13| null| null| null|

+--------+---------+------------+---------+----------+------------+--------------------+-------+--------------+--------------+--------------------+

only showing top 3 rows

>>> data\_frame\_1.select(integer\_list).show(3)

Traceback (most recent call last):

File "<stdin>", line 1, in <module>

File "/usr/local/spark/python/pyspark/sql/dataframe.py", line 1685, in select

jdf = self.\_jdf.select(self.\_jcols(\*cols))

File "/usr/local/spark/python/pyspark/sql/dataframe.py", line 1441, in \_jcols

return self.\_jseq(cols, \_to\_java\_column)

File "/usr/local/spark/python/pyspark/sql/dataframe.py", line 1428, in \_jseq

return \_to\_seq(self.sql\_ctx.\_sc, cols, converter)

File "/usr/local/spark/python/pyspark/sql/column.py", line 61, in \_to\_seq

cols = [converter(c) for c in cols]

File "/usr/local/spark/python/pyspark/sql/column.py", line 61, in <listcomp>

cols = [converter(c) for c in cols]

File "/usr/local/spark/python/pyspark/sql/column.py", line 49, in \_to\_java\_column

"function.".format(col, type(col)))

TypeError: Invalid argument, not a string or column: ('IssuerId', 'int') of type <class 'tuple'>. For column literals, use 'lit', 'array', 'struct' or 'create\_map' function.

>>> print(integer\_list)

[('IssuerId', 'int'), ('IssuerId2', 'int')]

>>>

>>> print(integer\_list)

[('IssuerId', 'int'), ('IssuerId2', 'int')]

>>> integer\_col\_name=[]

>>> for i in integer\_list :

... for j in i :

... integer\_col\_name.append(j[0])

...

>>> integer\_col\_name=[]

>>> print(integer\_list)

[('IssuerId', 'int'), ('IssuerId2', 'int')]

>>> integer\_col\_name=[]

>>> def function(integer\_list):

... for i in integer\_list :

... for j in i :

... integer\_col\_name.append(j[0])

...

>>> function(integer\_list)

>>> print(integer\_col\_name=)

File "<stdin>", line 1

print(integer\_col\_name=)

^

SyntaxError: invalid syntax

>>> print(integer\_col\_name)

['I', 'i', 'I', 'i']

>>> print(integer\_list)

[('IssuerId', 'int'), ('IssuerId2', 'int')]

>>> integer\_col\_name=[]

>>> def function(integer\_list):

... for i in integer\_list :

... integer\_col\_name.append(i[0])

...

>>> function(integer\_list)

>>> print(integer\_col\_name)

['IssuerId', 'IssuerId2']

>>> data\_frame\_1.select(integer\_col\_name).show(3)

+--------+---------+

|IssuerId|IssuerId2|

+--------+---------+

| 21989| 21989|

| 219| 219|

| 38344| 38344|

+--------+---------+

only showing top 3 rows

>>> reject\_data\_frame.describe

Traceback (most recent call last):

File "<stdin>", line 1, in <module>

NameError: name 'reject\_data\_frame' is not defined

>>> reject\_data\_frame\_!.describe

File "<stdin>", line 1

reject\_data\_frame\_!.describe

^

SyntaxError: invalid syntax

>>> reject\_data\_frame\_1.describe

Traceback (most recent call last):

File "<stdin>", line 1, in <module>

NameError: name 'reject\_data\_frame\_1' is not defined

>>> reject\_data\_frame\_1.show()

Traceback (most recent call last):

File "<stdin>", line 1, in <module>

NameError: name 'reject\_data\_frame\_1' is not defined

>>> rejected\_data\_frame\_1.describe

<bound method DataFrame.describe of DataFrame[IssuerId: int, IssuerId2: int, BusinessDate: date, StateCode: string, SourceName: string, NetworkName: string, NetworkURL: string, custnum: string, MarketCoverage: string, DentalOnlyPlan: string, RejectRows: string]>

>>> rejected\_data\_frame.describe

<bound method DataFrame.describe of DataFrame[IssuerId: int, IssuerId2: int, BusinessDate: date, StateCode: string, SourceName: string, NetworkName: string, NetworkURL: string, custnum: string, MarketCoverage: string, DentalOnlyPlan: string, RejectRows: string]>

>>> datat\_frame\_!.describe

File "<stdin>", line 1

datat\_frame\_!.describe

^

SyntaxError: invalid syntax

>>> datat\_frame\_1.describe

Traceback (most recent call last):

File "<stdin>", line 1, in <module>

NameError: name 'datat\_frame\_1' is not defined

>>> data\_frame\_1.describe

<bound method DataFrame.describe of DataFrame[IssuerId: int, IssuerId2: int, BusinessDate: date, StateCode: string, SourceName: string, NetworkName: string, NetworkURL: string, custnum: string, MarketCoverage: string, DentalOnlyPlan: string, RejectRows: string]>

>>> data\_frame\_2.describe

<bound method DataFrame.describe of DataFrame[IssuerId: int, IssuerId2: int, BusinessDate: date, StateCode: string, SourceName: string, NetworkName: string, NetworkURL: string, custnum: string, MarketCoverage: string, DentalOnlyPlan: string]>

>>> data\_frame\_3.describe

<bound method DataFrame.describe of DataFrame[IssuerId: int, IssuerId2: int, BusinessDate: date, StateCode: string, SourceName: string, NetworkName: string, NetworkURL: string, custnum: string, MarketCoverage: string, DentalOnlyPlan: string, RejectRows: string]>

>>>

>>> df\_1\_columns=data\_frame\_1.columns

>>> df\_3\_columns=data\_frame\_3.columns

>>> print(len(df\_1\_columns))

11

>>> print(len(df\_3\_columns))

11

>>> df\_1\_columns=data\_frame\_1.columns

>>> df\_2\_columns=data\_frame\_2.columns

>>> print(len(df\_1\_columns))

11

>>> print(len(df\_2\_columns))

10

>>> def function\_find\_Additional\_col(arr1,arr2):

... for i in range arr1 :

File "<stdin>", line 2

for i in range arr1 :

^

SyntaxError: invalid syntax

>>> def function\_find\_Additional\_col(arr1,arr2):

... for i in arr1 :

... for j in arr2 :

... if i in arr2 :

...

...

File "<stdin>", line 6

^

IndentationError: expected an indented block

>>> def function\_find\_Additional\_col(arr1,arr2):

... for i in arr1 :

... if i in arr2 :

... print("present")

... else :

... add\_columns\_present.append(i)

...

>>> add\_columns\_present=[]

>>> function\_find\_Additional\_col(df\_1\_columns,df\_2\_columns)

present

present

present

present

present

present

present

present

present

present

>>> print(add\_columns\_present)

['RejectRows']

>>> merged\_data\_frame.show(3)

+------------+----+-----+------------+--------------------+-------+--------------+--------------------+--------+---------+-----------------+----------+--------------------+

|BusinessDate|stcd|srcnm| NetworkName| NetworkURL|custnum|MarketCoverage| RejectRows|IssuerId|IssuerId2|issueridcomposite| sysdt| systs|

+------------+----+-----+------------+--------------------+-------+--------------+--------------------+--------+---------+-----------------+----------+--------------------+

| 2019-10-01| AK| HIOS| ODS Premier|https://www.modah...| 13| null| null| 21989| 21989| 2198921989|2024-07-22|2024-07-22 07:56:...|

| 2019-10-01| AK| HIOS| ODS Premier| null| null| null|219,219,2019-10-0...| 219| 219| 219219|2024-07-22|2024-07-22 07:56:...|

| 2019-10-01| AK| HIOS|HeritagePlus|https://www.preme...| 13| null| null| 38344| 38344| 3834438344|2024-07-22|2024-07-22 07:56:...|

+------------+----+-----+------------+--------------------+-------+--------------+--------------------+--------+---------+-----------------+----------+--------------------+

only showing top 3 rows

>>> print(merged\_data\_frame.count())

3737

>>> drop\_rows\_with\_null=merged\_data\_frame.na.drop(how="any")

>>> print(drop\_rows\_with\_null.count())

0

>>> drop\_rows\_with\_null.show(3)

+------------+----+-----+-----------+----------+-------+--------------+----------+--------+---------+-----------------+-----+-----+

|BusinessDate|stcd|srcnm|NetworkName|NetworkURL|custnum|MarketCoverage|RejectRows|IssuerId|IssuerId2|issueridcomposite|sysdt|systs|

+------------+----+-----+-----------+----------+-------+--------------+----------+--------+---------+-----------------+-----+-----+

+------------+----+-----+-----------+----------+-------+--------------+----------+--------+---------+-----------------+-----+-----+

>>> print(merged\_data\_frame.count())

^[[A3737

>>> print(merged\_data\_frame.count())

3737

>>> drop\_rows\_with\_null=merged\_data\_frame.na.drop(how="any",subset=["BusinessDate","stcd" ,"srcnm","NetworkName","NetworkURL","custnum","MarketCoverage","IssuerId","IssuerId2","issueridcomposite","sysdt","systs"])

>>> print(drop\_rows\_with\_null.count())

1408

>>> drop\_rows\_with\_null.where("RejectRows is not NULL").show()

+------------+----+-----+-----------+----------+-------+--------------+----------+--------+---------+-----------------+-----+-----+

|BusinessDate|stcd|srcnm|NetworkName|NetworkURL|custnum|MarketCoverage|RejectRows|IssuerId|IssuerId2|issueridcomposite|sysdt|systs|

+------------+----+-----+-----------+----------+-------+--------------+----------+--------+---------+-----------------+-----+-----+

+------------+----+-----+-----------+----------+-------+--------------+----------+--------+---------+-----------------+-----+-----+

>>> drop\_rows\_with\_null.where("RejectRows is NULL").show()

+------------+----+-----+--------------------+--------------------+-------+------------------+----------+--------+---------+-----------------+----------+--------------------+

|BusinessDate|stcd|srcnm| NetworkName| NetworkURL|custnum| MarketCoverage|RejectRows|IssuerId|IssuerId2|issueridcomposite| sysdt| systs|

+------------+----+-----+--------------------+--------------------+-------+------------------+----------+--------+---------+-----------------+----------+--------------------+

| 2019-10-01| IL|SERFF| PPO|https://public.hc...| 13| Individual| null| 36096| 36096| 3609636096|2024-07-22|2024-07-22 08:02:...|

| 2019-10-01| IL|SERFF| Blue Choice PPO|https://public.hc...| 14|SHOP (Small Group)| null| 36096| 36096| 3609636096|2024-07-22|2024-07-22 08:02:...|

| 2019-10-01| IL|SERFF| Blue Precision HMO|https://public.hc...| 14| Individual| null| 36096| 36096| 3609636096|2024-07-22|2024-07-22 08:02:...|

| 2019-10-01| IL|SERFF| Renaissance Dental|www.renaissancede...| 15|SHOP (Small Group)| null| 46552| 46552| 4655246552|2024-07-22|2024-07-22 08:02:...|

| 2019-10-01| IL|SERFF| Delta Dental|http://www.deltad...| 15| Individual| null| 46552| 46552| 4655246552|2024-07-22|2024-07-22 08:02:...|

| 2019-10-01| IL|SERFF|NorthShore Comple...|http://pfp.humana...| 13|SHOP (Small Group)| null| 58288| 58288| 5828858288|2024-07-22|2024-07-22 08:02:...|

| 2019-10-01| IL|SERFF| Chicago HMOx|http://pfp.humana...| 13| Individual| null| 58288| 58288| 5828858288|2024-07-22|2024-07-22 08:02:...|

| 2019-10-01| IL|SERFF| Springfield HMOx|http://pfp.humana...| 14| Individual| null| 58288| 58288| 5828858288|2024-07-22|2024-07-22 08:02:...|

| 2019-10-01| IL|SERFF|Rockford Peoria HMOx|http://pfp.humana...| 13| Individual| null| 58288| 58288| 5828858288|2024-07-22|2024-07-22 08:02:...|

| 2019-10-01| IL|SERFF| Delta Dental PPO|www.deltadentalil...| 14| Individual| null| 60600| 60600| 6060060600|2024-07-22|2024-07-22 08:02:...|

| 2019-10-01| IL|SERFF| ChoicePOS|http://pfp.humana...| 15| Individual| null| 68303| 68303| 6830368303|2024-07-22|2024-07-22 08:02:...|

| 2019-10-01| IL|SERFF|Humana Dental PPO...|http://pfp.humana...| 16| Individual| null| 68303| 68303| 6830368303|2024-07-22|2024-07-22 08:02:...|

| 2019-10-01| IL|SERFF|Humana Dental PPO...|https://www.human...| 13| Individual| null| 68303| 68303| 6830368303|2024-07-22|2024-07-22 08:02:...|

| 2019-10-01| IL|SERFF|QHP PPO\_OAMC\_Chic...|http://www.aetna....| 13|SHOP (Small Group)| null| 72547| 72547| 7254772547|2024-07-22|2024-07-22 08:02:...|

| 2019-10-01| NC| HIOS|Carelink Duke Net...|http://nc.coventr...| 13| Individual| null| 56346| 56346| 5634656346|2024-07-22|2024-07-22 08:02:...|

| 2019-10-01| NC| HIOS| CoventryOne Network|http://nc.coventr...| 13| Individual| null| 56346| 56346| 5634656346|2024-07-22|2024-07-22 08:02:...|

| 2019-10-01| NC| HIOS| Delta Dental|http://www.deltad...| 13| Individual| null| 56891| 56891| 5689156891|2024-07-22|2024-07-22 08:02:...|

| 2019-10-01| NC| HIOS| DenteMax|http://www2.dente...| 13| Individual| null| 94482| 94482| 9448294482|2024-07-22|2024-07-22 08:02:...|

| 2019-01-10| SD|SERFF|Lincoln Dental Co...|http://lfg.go2den...| 13| Individual| null| 80261| 80261| 8026180261|2024-07-22|2024-07-22 08:02:...|

| 2019-01-10| SD|SERFF|DentalGuard Prefe...|https://www.guard...| 13|SHOP (Small Group)| null| 80608| 80608| 8060880608|2024-07-22|2024-07-22 08:02:...|

+------------+----+-----+--------------------+--------------------+-------+------------------+----------+--------+---------+-----------------+----------+--------------------+

only showing top 20 rows

>>> drop\_rows\_with\_null.where("RejectRows is not NULL").show(3)

+------------+----+-----+-----------+----------+-------+--------------+----------+--------+---------+-----------------+-----+-----+

|BusinessDate|stcd|srcnm|NetworkName|NetworkURL|custnum|MarketCoverage|RejectRows|IssuerId|IssuerId2|issueridcomposite|sysdt|systs|

+------------+----+-----+-----------+----------+-------+--------------+----------+--------+---------+-----------------+-----+-----+

+------------+----+-----+-----------+----------+-------+--------------+----------+--------+---------+-----------------+-----+-----+

>>> drop\_rows\_with\_null.where("RejectRows is NULL").show(3)

+------------+----+-----+------------------+--------------------+-------+------------------+----------+--------+---------+-----------------+----------+--------------------+

|BusinessDate|stcd|srcnm| NetworkName| NetworkURL|custnum| MarketCoverage|RejectRows|IssuerId|IssuerId2|issueridcomposite| sysdt| systs|

+------------+----+-----+------------------+--------------------+-------+------------------+----------+--------+---------+-----------------+----------+--------------------+

| 2019-10-01| IL|SERFF| PPO|https://public.hc...| 13| Individual| null| 36096| 36096| 3609636096|2024-07-22|2024-07-22 08:02:...|

| 2019-10-01| IL|SERFF| Blue Choice PPO|https://public.hc...| 14|SHOP (Small Group)| null| 36096| 36096| 3609636096|2024-07-22|2024-07-22 08:02:...|

| 2019-10-01| IL|SERFF|Blue Precision HMO|https://public.hc...| 14| Individual| null| 36096| 36096| 3609636096|2024-07-22|2024-07-22 08:02:...|

+------------+----+-----+------------------+--------------------+-------+------------------+----------+--------+---------+-----------------+----------+--------------------+

only showing top 3 rows

>>> from pyspark.sql import \*

>>> spark=SparkSession.builder.master("local[2]").appName("Hackathon").enableHiveSupport().getOrCreate()

>>> sc=spark.sparkContext

>>>

#this is inside the python\_package

import re

def clean\_string(string\_arg):

pattern = re.compile(r'[^A-Za-z" "]')

final\_string = pattern.sub(' ', string\_arg)

return final\_string

\

#THis is a file ur the python\_sub\_package folder wherer the python\_package and python\_sub\_package are under the working directory

from Python\_package.methods import \*

instance\_creation = clean\_string("Pathway - 2X (with dental)")

print(instance\_creation)

(Hackathon) [hduser@localhost Working]$ cd /home/hduser/PycharmProjects/Hackathon/Working

(Hackathon) [hduser@localhost Working]$ export PYTHONPATH=/home/hduser/PycharmProjects/Hackathon/Working

(Hackathon) [hduser@localhost Working]$ python Python\_sub\_package/main.py

Pathway X with dental

>>> from pyspark.sql import \*

>>> spark=SparkSession.builder.master("local[2]").appName("Hackathon").enableHiveSupport().getOrCreate()

>>> sc=spark.sparkContext

>>>

>>>

>>> def clean\_string(string\_arg):

... pattern = re.compile(r'[^A-Za-z" "]')

... final\_string = pattern.sub(' ', string\_arg)

... return final\_string

...

>>> print(clean\_string("Pathway - 2X (with dental) "))

Traceback (most recent call last):

File "<stdin>", line 1, in <module>

File "<stdin>", line 2, in clean\_string

NameError: name 're' is not defined

>>> import re

>>> def clean\_string(string\_arg):

... pattern = re.compile(r'[^A-Za-z" "]')

... final\_string = pattern.sub(' ', string\_arg)

... return final\_string

...

>>> print(clean\_string("Pathway - 2X (with dental) "))

Pathway X with dental

>>> df\_function=udf(clean\_string)

>>> merged\_data\_frame.show(3)

+------------+----+-----+------------+--------------------+-------+--------------+--------------------+--------+---------+-----------------+----------+--------------------+

|BusinessDate|stcd|srcnm| NetworkName| NetworkURL|custnum|MarketCoverage| RejectRows|IssuerId|IssuerId2|issueridcomposite| sysdt| systs|

+------------+----+-----+------------+--------------------+-------+--------------+--------------------+--------+---------+-----------------+----------+--------------------+

| 2019-10-01| AK| HIOS| ODS Premier|https://www.modah...| 13| null| null| 21989| 21989| 2198921989|2024-07-22|2024-07-22 09:04:...|

| 2019-10-01| AK| HIOS| ODS Premier| null| null| null|219,219,2019-10-0...| 219| 219| 219219|2024-07-22|2024-07-22 09:04:...|

| 2019-10-01| AK| HIOS|HeritagePlus|https://www.preme...| 13| null| null| 38344| 38344| 3834438344|2024-07-22|2024-07-22 09:04:...|

+------------+----+-----+------------+--------------------+-------+--------------+--------------------+--------+---------+-----------------+----------+--------------------+

only showing top 3 rows

>>> merged\_data\_frame.select("\*, df\_function(NetworkName).alias("New NetworkName")").show(1)

File "<stdin>", line 1

merged\_data\_frame.select("\*, df\_function(NetworkName).alias("New NetworkName")").show(1)

^

SyntaxError: invalid syntax

>>> df\_function=udf(clean\_string,StringType())

>>>

>>> df\_function=udf(clean\_string,StringType())

>>> merged\_data\_frame.select("\*, df\_function(NetworkName).alias("New NetworkName")").show(1)

File "<stdin>", line 1

merged\_data\_frame.select("\*, df\_function(NetworkName).alias("New NetworkName")").show(1)

^

SyntaxError: invalid syntax

>>> merged\_data\_frame.select("\*, df\_function("NetworkName").alias("New NetworkName")").show(1)

File "<stdin>", line 1

merged\_data\_frame.select("\*, df\_function("NetworkName").alias("New NetworkName")").show(1)

^

SyntaxError: invalid syntax

>>> df\_function=udf(clean\_string,StringType())

>>> merged\_data\_frame.select("\*, df\_function("NetworkName").alias("New NetworkName")).show(1)

File "<stdin>", line 1

merged\_data\_frame.select("\*, df\_function("NetworkName").alias("New NetworkName")).show(1)

^

SyntaxError: invalid syntax

>>> sample\_op=merged\_data\_frame.select("\*, df\_function("NetworkName").alias("New NetworkName")).show(1)

File "<stdin>", line 1

sample\_op=merged\_data\_frame.select("\*, df\_function("NetworkName").alias("New NetworkName")).show(1)

^

SyntaxError: invalid syntax

>>> sample\_op=merged\_data\_frame.withColumn("New NetworkName",df\_function(col("NetworkName"))).show(1)

+------------+----+-----+-----------+--------------------+-------+--------------+----------+--------+---------+-----------------+----------+--------------------+---------------+

|BusinessDate|stcd|srcnm|NetworkName| NetworkURL|custnum|MarketCoverage|RejectRows|IssuerId|IssuerId2|issueridcomposite| sysdt| systs|New NetworkName|

+------------+----+-----+-----------+--------------------+-------+--------------+----------+--------+---------+-----------------+----------+--------------------+---------------+

| 2019-10-01| AK| HIOS|ODS Premier|https://www.modah...| 13| null| null| 21989| 21989| 2198921989|2024-07-22|2024-07-22 09:09:...| ODS Premier|

+------------+----+-----+-----------+--------------------+-------+--------------+----------+--------+---------+-----------------+----------+--------------------+---------------+

only showing top 1 row

>>> df\_function=udf(clean\_string,StringType())

>>> df\_function=udf(clean\_string,StringType())

>>> sample\_op=merged\_data\_frame.withColumn("New NetworkName",df\_function(col("NetworkName"))).show(1)

+------------+----+-----+-----------+--------------------+-------+--------------+----------+--------+---------+-----------------+----------+--------------------+---------------+

|BusinessDate|stcd|srcnm|NetworkName| NetworkURL|custnum|MarketCoverage|RejectRows|IssuerId|IssuerId2|issueridcomposite| sysdt| systs|New NetworkName|

+------------+----+-----+-----------+--------------------+-------+--------------+----------+--------+---------+-----------------+----------+--------------------+---------------+

| 2019-10-01| AK| HIOS|ODS Premier|https://www.modah...| 13| null| null| 21989| 21989| 2198921989|2024-07-22|2024-07-22 09:10:...| ODS Premier|

+------------+----+-----+-----------+--------------------+-------+--------------+----------+--------+---------+-----------------+----------+--------------------+---------------+

only showing top 1 row

>>> sample\_op.write.json("hdfs:///user/hduser/Hackathon/JSON\_DATA",mode="overwrite")

Traceback (most recent call last):

File "<stdin>", line 1, in <module>

AttributeError: 'NoneType' object has no attribute 'write'

>>> sample\_op=merged\_data\_frame.withColumn("New NetworkName",df\_function(col("NetworkName")))

>>> sample\_op.write.json("hdfs:///user/hduser/Hackathon/JSON\_DATA",mode="overwrite")

24/07/22 09:14:01 ERROR util.Utils: Aborting task (0 + 2) / 2]

org.apache.spark.api.python.PythonException: Traceback (most recent call last):

File "/usr/local/spark/python/lib/pyspark.zip/pyspark/worker.py", line 619, in main

process()

File "/usr/local/spark/python/lib/pyspark.zip/pyspark/worker.py", line 611, in process

serializer.dump\_stream(out\_iter, outfile)

File "/usr/local/spark/python/lib/pyspark.zip/pyspark/serializers.py", line 211, in dump\_stream

self.serializer.dump\_stream(self.\_batched(iterator), stream)

File "/usr/local/spark/python/lib/pyspark.zip/pyspark/serializers.py", line 132, in dump\_stream

for obj in iterator:

File "/usr/local/spark/python/lib/pyspark.zip/pyspark/serializers.py", line 200, in \_batched

for item in iterator:

File "/usr/local/spark/python/lib/pyspark.zip/pyspark/worker.py", line 452, in mapper

result = tuple(f(\*[a[o] for o in arg\_offsets]) for (arg\_offsets, f) in udfs)

File "/usr/local/spark/python/lib/pyspark.zip/pyspark/worker.py", line 452, in <genexpr>

result = tuple(f(\*[a[o] for o in arg\_offsets]) for (arg\_offsets, f) in udfs)

File "/usr/local/spark/python/lib/pyspark.zip/pyspark/worker.py", line 87, in <lambda>

return lambda \*a: f(\*a)

File "/usr/local/spark/python/lib/pyspark.zip/pyspark/util.py", line 74, in wrapper

return f(\*args, \*\*kwargs)

File "<stdin>", line 3, in clean\_string

TypeError: expected string or bytes-like object

at org.apache.spark.api.python.BasePythonRunner$ReaderIterator.handlePythonException(PythonRunner.scala:555)

at org.apache.spark.sql.execution.python.PythonUDFRunner$$anon$2.read(PythonUDFRunner.scala:86)

at org.apache.spark.sql.execution.python.PythonUDFRunner$$anon$2.read(PythonUDFRunner.scala:68)

at org.apache.spark.api.python.BasePythonRunner$ReaderIterator.hasNext(PythonRunner.scala:508)

at org.apache.spark.InterruptibleIterator.hasNext(InterruptibleIterator.scala:37)

at scala.collection.Iterator$$anon$11.hasNext(Iterator.scala:491)

at scala.collection.Iterator$$anon$10.hasNext(Iterator.scala:460)

at scala.collection.Iterator$$anon$10.hasNext(Iterator.scala:460)

at org.apache.spark.sql.catalyst.expressions.GeneratedClass$GeneratedIteratorForCodegenStage1.processNext(Unknown Source)

at org.apache.spark.sql.execution.BufferedRowIterator.hasNext(BufferedRowIterator.java:43)

at org.apache.spark.sql.execution.WholeStageCodegenExec$$anon$1.hasNext(WholeStageCodegenExec.scala:759)

at org.apache.spark.sql.execution.datasources.FileFormatDataWriter.writeWithIterator(FileFormatDataWriter.scala:91)

at org.apache.spark.sql.execution.datasources.FileFormatWriter$.$anonfun$executeTask$1(FileFormatWriter.scala:304)

at org.apache.spark.util.Utils$.tryWithSafeFinallyAndFailureCallbacks(Utils.scala:1496)

at org.apache.spark.sql.execution.datasources.FileFormatWriter$.executeTask(FileFormatWriter.scala:311)

at org.apache.spark.sql.execution.datasources.FileFormatWriter$.$anonfun$write$16(FileFormatWriter.scala:229)

at org.apache.spark.scheduler.ResultTask.runTask(ResultTask.scala:90)

at org.apache.spark.scheduler.Task.run(Task.scala:131)

at org.apache.spark.executor.Executor$TaskRunner.$anonfun$run$3(Executor.scala:506)

at org.apache.spark.util.Utils$.tryWithSafeFinally(Utils.scala:1462)

at org.apache.spark.executor.Executor$TaskRunner.run(Executor.scala:509)

at java.util.concurrent.ThreadPoolExecutor.runWorker(ThreadPoolExecutor.java:1149)

at java.util.concurrent.ThreadPoolExecutor$Worker.run(ThreadPoolExecutor.java:624)

at java.lang.Thread.run(Thread.java:750)

24/07/22 09:14:02 ERROR util.Utils: Aborting task

org.apache.spark.api.python.PythonException: Traceback (most recent call last):

File "/usr/local/spark/python/lib/pyspark.zip/pyspark/worker.py", line 619, in main

process()

File "/usr/local/spark/python/lib/pyspark.zip/pyspark/worker.py", line 611, in process

serializer.dump\_stream(out\_iter, outfile)

File "/usr/local/spark/python/lib/pyspark.zip/pyspark/serializers.py", line 211, in dump\_stream

self.serializer.dump\_stream(self.\_batched(iterator), stream)

File "/usr/local/spark/python/lib/pyspark.zip/pyspark/serializers.py", line 132, in dump\_stream

for obj in iterator:

File "/usr/local/spark/python/lib/pyspark.zip/pyspark/serializers.py", line 200, in \_batched

for item in iterator:

File "/usr/local/spark/python/lib/pyspark.zip/pyspark/worker.py", line 452, in mapper

result = tuple(f(\*[a[o] for o in arg\_offsets]) for (arg\_offsets, f) in udfs)

File "/usr/local/spark/python/lib/pyspark.zip/pyspark/worker.py", line 452, in <genexpr>

result = tuple(f(\*[a[o] for o in arg\_offsets]) for (arg\_offsets, f) in udfs)

File "/usr/local/spark/python/lib/pyspark.zip/pyspark/worker.py", line 87, in <lambda>

return lambda \*a: f(\*a)

File "/usr/local/spark/python/lib/pyspark.zip/pyspark/util.py", line 74, in wrapper

return f(\*args, \*\*kwargs)

File "<stdin>", line 3, in clean\_string

TypeError: expected string or bytes-like object

at org.apache.spark.api.python.BasePythonRunner$ReaderIterator.handlePythonException(PythonRunner.scala:555)

at org.apache.spark.sql.execution.python.PythonUDFRunner$$anon$2.read(PythonUDFRunner.scala:86)

at org.apache.spark.sql.execution.python.PythonUDFRunner$$anon$2.read(PythonUDFRunner.scala:68)

at org.apache.spark.api.python.BasePythonRunner$ReaderIterator.hasNext(PythonRunner.scala:508)

at org.apache.spark.InterruptibleIterator.hasNext(InterruptibleIterator.scala:37)

at scala.collection.Iterator$$anon$11.hasNext(Iterator.scala:491)

at scala.collection.Iterator$$anon$10.hasNext(Iterator.scala:460)

at scala.collection.Iterator$$anon$10.hasNext(Iterator.scala:460)

at org.apache.spark.sql.catalyst.expressions.GeneratedClass$GeneratedIteratorForCodegenStage2.processNext(Unknown Source)

at org.apache.spark.sql.execution.BufferedRowIterator.hasNext(BufferedRowIterator.java:43)

at org.apache.spark.sql.execution.WholeStageCodegenExec$$anon$1.hasNext(WholeStageCodegenExec.scala:759)

at org.apache.spark.sql.execution.datasources.FileFormatDataWriter.writeWithIterator(FileFormatDataWriter.scala:91)

at org.apache.spark.sql.execution.datasources.FileFormatWriter$.$anonfun$executeTask$1(FileFormatWriter.scala:304)

at org.apache.spark.util.Utils$.tryWithSafeFinallyAndFailureCallbacks(Utils.scala:1496)

at org.apache.spark.sql.execution.datasources.FileFormatWriter$.executeTask(FileFormatWriter.scala:311)

at org.apache.spark.sql.execution.datasources.FileFormatWriter$.$anonfun$write$16(FileFormatWriter.scala:229)

at org.apache.spark.scheduler.ResultTask.runTask(ResultTask.scala:90)

at org.apache.spark.scheduler.Task.run(Task.scala:131)

at org.apache.spark.executor.Executor$TaskRunner.$anonfun$run$3(Executor.scala:506)

at org.apache.spark.util.Utils$.tryWithSafeFinally(Utils.scala:1462)

at org.apache.spark.executor.Executor$TaskRunner.run(Executor.scala:509)

at java.util.concurrent.ThreadPoolExecutor.runWorker(ThreadPoolExecutor.java:1149)

at java.util.concurrent.ThreadPoolExecutor$Worker.run(ThreadPoolExecutor.java:624)

at java.lang.Thread.run(Thread.java:750)

24/07/22 09:14:03 ERROR datasources.FileFormatWriter: Job job\_202407220913597215005389064139754\_0092 aborted.

24/07/22 09:14:03 ERROR datasources.FileFormatWriter: Job job\_202407220913594695056606308186655\_0092 aborted.

24/07/22 09:14:03 ERROR executor.Executor: Exception in task 0.0 in stage 92.0 (TID 92)

org.apache.spark.SparkException: Task failed while writing rows.

at org.apache.spark.sql.errors.QueryExecutionErrors$.taskFailedWhileWritingRowsError(QueryExecutionErrors.scala:500)

at org.apache.spark.sql.execution.datasources.FileFormatWriter$.executeTask(FileFormatWriter.scala:321)

at org.apache.spark.sql.execution.datasources.FileFormatWriter$.$anonfun$write$16(FileFormatWriter.scala:229)

at org.apache.spark.scheduler.ResultTask.runTask(ResultTask.scala:90)

at org.apache.spark.scheduler.Task.run(Task.scala:131)

at org.apache.spark.executor.Executor$TaskRunner.$anonfun$run$3(Executor.scala:506)

at org.apache.spark.util.Utils$.tryWithSafeFinally(Utils.scala:1462)

at org.apache.spark.executor.Executor$TaskRunner.run(Executor.scala:509)

at java.util.concurrent.ThreadPoolExecutor.runWorker(ThreadPoolExecutor.java:1149)

at java.util.concurrent.ThreadPoolExecutor$Worker.run(ThreadPoolExecutor.java:624)

at java.lang.Thread.run(Thread.java:750)

Caused by: org.apache.spark.api.python.PythonException: Traceback (most recent call last):

File "/usr/local/spark/python/lib/pyspark.zip/pyspark/worker.py", line 619, in main

process()

File "/usr/local/spark/python/lib/pyspark.zip/pyspark/worker.py", line 611, in process

serializer.dump\_stream(out\_iter, outfile)

File "/usr/local/spark/python/lib/pyspark.zip/pyspark/serializers.py", line 211, in dump\_stream

self.serializer.dump\_stream(self.\_batched(iterator), stream)

File "/usr/local/spark/python/lib/pyspark.zip/pyspark/serializers.py", line 132, in dump\_stream

for obj in iterator:

File "/usr/local/spark/python/lib/pyspark.zip/pyspark/serializers.py", line 200, in \_batched

for item in iterator:

File "/usr/local/spark/python/lib/pyspark.zip/pyspark/worker.py", line 452, in mapper

result = tuple(f(\*[a[o] for o in arg\_offsets]) for (arg\_offsets, f) in udfs)

File "/usr/local/spark/python/lib/pyspark.zip/pyspark/worker.py", line 452, in <genexpr>

result = tuple(f(\*[a[o] for o in arg\_offsets]) for (arg\_offsets, f) in udfs)

File "/usr/local/spark/python/lib/pyspark.zip/pyspark/worker.py", line 87, in <lambda>

return lambda \*a: f(\*a)

File "/usr/local/spark/python/lib/pyspark.zip/pyspark/util.py", line 74, in wrapper

return f(\*args, \*\*kwargs)

File "<stdin>", line 3, in clean\_string

TypeError: expected string or bytes-like object

at org.apache.spark.api.python.BasePythonRunner$ReaderIterator.handlePythonException(PythonRunner.scala:555)

at org.apache.spark.sql.execution.python.PythonUDFRunner$$anon$2.read(PythonUDFRunner.scala:86)

at org.apache.spark.sql.execution.python.PythonUDFRunner$$anon$2.read(PythonUDFRunner.scala:68)

at org.apache.spark.api.python.BasePythonRunner$ReaderIterator.hasNext(PythonRunner.scala:508)

at org.apache.spark.InterruptibleIterator.hasNext(InterruptibleIterator.scala:37)

at scala.collection.Iterator$$anon$11.hasNext(Iterator.scala:491)

at scala.collection.Iterator$$anon$10.hasNext(Iterator.scala:460)

at scala.collection.Iterator$$anon$10.hasNext(Iterator.scala:460)

at org.apache.spark.sql.catalyst.expressions.GeneratedClass$GeneratedIteratorForCodegenStage1.processNext(Unknown Source)

at org.apache.spark.sql.execution.BufferedRowIterator.hasNext(BufferedRowIterator.java:43)

at org.apache.spark.sql.execution.WholeStageCodegenExec$$anon$1.hasNext(WholeStageCodegenExec.scala:759)

at org.apache.spark.sql.execution.datasources.FileFormatDataWriter.writeWithIterator(FileFormatDataWriter.scala:91)

at org.apache.spark.sql.execution.datasources.FileFormatWriter$.$anonfun$executeTask$1(FileFormatWriter.scala:304)

at org.apache.spark.util.Utils$.tryWithSafeFinallyAndFailureCallbacks(Utils.scala:1496)

at org.apache.spark.sql.execution.datasources.FileFormatWriter$.executeTask(FileFormatWriter.scala:311)

... 9 more

24/07/22 09:14:03 ERROR executor.Executor: Exception in task 1.0 in stage 92.0 (TID 93)

org.apache.spark.SparkException: Task failed while writing rows.

at org.apache.spark.sql.errors.QueryExecutionErrors$.taskFailedWhileWritingRowsError(QueryExecutionErrors.scala:500)

at org.apache.spark.sql.execution.datasources.FileFormatWriter$.executeTask(FileFormatWriter.scala:321)

at org.apache.spark.sql.execution.datasources.FileFormatWriter$.$anonfun$write$16(FileFormatWriter.scala:229)

at org.apache.spark.scheduler.ResultTask.runTask(ResultTask.scala:90)

at org.apache.spark.scheduler.Task.run(Task.scala:131)

at org.apache.spark.executor.Executor$TaskRunner.$anonfun$run$3(Executor.scala:506)

at org.apache.spark.util.Utils$.tryWithSafeFinally(Utils.scala:1462)

at org.apache.spark.executor.Executor$TaskRunner.run(Executor.scala:509)

at java.util.concurrent.ThreadPoolExecutor.runWorker(ThreadPoolExecutor.java:1149)

at java.util.concurrent.ThreadPoolExecutor$Worker.run(ThreadPoolExecutor.java:624)

at java.lang.Thread.run(Thread.java:750)

Caused by: org.apache.spark.api.python.PythonException: Traceback (most recent call last):

File "/usr/local/spark/python/lib/pyspark.zip/pyspark/worker.py", line 619, in main

process()

File "/usr/local/spark/python/lib/pyspark.zip/pyspark/worker.py", line 611, in process

serializer.dump\_stream(out\_iter, outfile)

File "/usr/local/spark/python/lib/pyspark.zip/pyspark/serializers.py", line 211, in dump\_stream

self.serializer.dump\_stream(self.\_batched(iterator), stream)

File "/usr/local/spark/python/lib/pyspark.zip/pyspark/serializers.py", line 132, in dump\_stream

for obj in iterator:

File "/usr/local/spark/python/lib/pyspark.zip/pyspark/serializers.py", line 200, in \_batched

for item in iterator:

File "/usr/local/spark/python/lib/pyspark.zip/pyspark/worker.py", line 452, in mapper

result = tuple(f(\*[a[o] for o in arg\_offsets]) for (arg\_offsets, f) in udfs)

File "/usr/local/spark/python/lib/pyspark.zip/pyspark/worker.py", line 452, in <genexpr>

result = tuple(f(\*[a[o] for o in arg\_offsets]) for (arg\_offsets, f) in udfs)

File "/usr/local/spark/python/lib/pyspark.zip/pyspark/worker.py", line 87, in <lambda>

return lambda \*a: f(\*a)

File "/usr/local/spark/python/lib/pyspark.zip/pyspark/util.py", line 74, in wrapper

return f(\*args, \*\*kwargs)

File "<stdin>", line 3, in clean\_string

TypeError: expected string or bytes-like object

at org.apache.spark.api.python.BasePythonRunner$ReaderIterator.handlePythonException(PythonRunner.scala:555)

at org.apache.spark.sql.execution.python.PythonUDFRunner$$anon$2.read(PythonUDFRunner.scala:86)

at org.apache.spark.sql.execution.python.PythonUDFRunner$$anon$2.read(PythonUDFRunner.scala:68)

at org.apache.spark.api.python.BasePythonRunner$ReaderIterator.hasNext(PythonRunner.scala:508)

at org.apache.spark.InterruptibleIterator.hasNext(InterruptibleIterator.scala:37)

at scala.collection.Iterator$$anon$11.hasNext(Iterator.scala:491)

at scala.collection.Iterator$$anon$10.hasNext(Iterator.scala:460)

at scala.collection.Iterator$$anon$10.hasNext(Iterator.scala:460)

at org.apache.spark.sql.catalyst.expressions.GeneratedClass$GeneratedIteratorForCodegenStage2.processNext(Unknown Source)

at org.apache.spark.sql.execution.BufferedRowIterator.hasNext(BufferedRowIterator.java:43)

at org.apache.spark.sql.execution.WholeStageCodegenExec$$anon$1.hasNext(WholeStageCodegenExec.scala:759)

at org.apache.spark.sql.execution.datasources.FileFormatDataWriter.writeWithIterator(FileFormatDataWriter.scala:91)

at org.apache.spark.sql.execution.datasources.FileFormatWriter$.$anonfun$executeTask$1(FileFormatWriter.scala:304)

at org.apache.spark.util.Utils$.tryWithSafeFinallyAndFailureCallbacks(Utils.scala:1496)

at org.apache.spark.sql.execution.datasources.FileFormatWriter$.executeTask(FileFormatWriter.scala:311)

... 9 more

24/07/22 09:14:03 WARN scheduler.TaskSetManager: Lost task 0.0 in stage 92.0 (TID 92) (192.168.112.129 executor driver): org.apache.spark.SparkException: Task failed while writing rows.

at org.apache.spark.sql.errors.QueryExecutionErrors$.taskFailedWhileWritingRowsError(QueryExecutionErrors.scala:500)

at org.apache.spark.sql.execution.datasources.FileFormatWriter$.executeTask(FileFormatWriter.scala:321)

at org.apache.spark.sql.execution.datasources.FileFormatWriter$.$anonfun$write$16(FileFormatWriter.scala:229)

at org.apache.spark.scheduler.ResultTask.runTask(ResultTask.scala:90)

at org.apache.spark.scheduler.Task.run(Task.scala:131)

at org.apache.spark.executor.Executor$TaskRunner.$anonfun$run$3(Executor.scala:506)

at org.apache.spark.util.Utils$.tryWithSafeFinally(Utils.scala:1462)

at org.apache.spark.executor.Executor$TaskRunner.run(Executor.scala:509)

at java.util.concurrent.ThreadPoolExecutor.runWorker(ThreadPoolExecutor.java:1149)

at java.util.concurrent.ThreadPoolExecutor$Worker.run(ThreadPoolExecutor.java:624)

at java.lang.Thread.run(Thread.java:750)

Caused by: org.apache.spark.api.python.PythonException: Traceback (most recent call last):

File "/usr/local/spark/python/lib/pyspark.zip/pyspark/worker.py", line 619, in main

process()

File "/usr/local/spark/python/lib/pyspark.zip/pyspark/worker.py", line 611, in process

serializer.dump\_stream(out\_iter, outfile)

File "/usr/local/spark/python/lib/pyspark.zip/pyspark/serializers.py", line 211, in dump\_stream

self.serializer.dump\_stream(self.\_batched(iterator), stream)

File "/usr/local/spark/python/lib/pyspark.zip/pyspark/serializers.py", line 132, in dump\_stream

for obj in iterator:

File "/usr/local/spark/python/lib/pyspark.zip/pyspark/serializers.py", line 200, in \_batched

for item in iterator:

File "/usr/local/spark/python/lib/pyspark.zip/pyspark/worker.py", line 452, in mapper

result = tuple(f(\*[a[o] for o in arg\_offsets]) for (arg\_offsets, f) in udfs)

File "/usr/local/spark/python/lib/pyspark.zip/pyspark/worker.py", line 452, in <genexpr>

result = tuple(f(\*[a[o] for o in arg\_offsets]) for (arg\_offsets, f) in udfs)

File "/usr/local/spark/python/lib/pyspark.zip/pyspark/worker.py", line 87, in <lambda>

return lambda \*a: f(\*a)

File "/usr/local/spark/python/lib/pyspark.zip/pyspark/util.py", line 74, in wrapper

return f(\*args, \*\*kwargs)

File "<stdin>", line 3, in clean\_string

TypeError: expected string or bytes-like object

at org.apache.spark.api.python.BasePythonRunner$ReaderIterator.handlePythonException(PythonRunner.scala:555)

at org.apache.spark.sql.execution.python.PythonUDFRunner$$anon$2.read(PythonUDFRunner.scala:86)

at org.apache.spark.sql.execution.python.PythonUDFRunner$$anon$2.read(PythonUDFRunner.scala:68)

at org.apache.spark.api.python.BasePythonRunner$ReaderIterator.hasNext(PythonRunner.scala:508)

at org.apache.spark.InterruptibleIterator.hasNext(InterruptibleIterator.scala:37)

at scala.collection.Iterator$$anon$11.hasNext(Iterator.scala:491)

at scala.collection.Iterator$$anon$10.hasNext(Iterator.scala:460)

at scala.collection.Iterator$$anon$10.hasNext(Iterator.scala:460)

at org.apache.spark.sql.catalyst.expressions.GeneratedClass$GeneratedIteratorForCodegenStage1.processNext(Unknown Source)

at org.apache.spark.sql.execution.BufferedRowIterator.hasNext(BufferedRowIterator.java:43)

at org.apache.spark.sql.execution.WholeStageCodegenExec$$anon$1.hasNext(WholeStageCodegenExec.scala:759)

at org.apache.spark.sql.execution.datasources.FileFormatDataWriter.writeWithIterator(FileFormatDataWriter.scala:91)

at org.apache.spark.sql.execution.datasources.FileFormatWriter$.$anonfun$executeTask$1(FileFormatWriter.scala:304)

at org.apache.spark.util.Utils$.tryWithSafeFinallyAndFailureCallbacks(Utils.scala:1496)

at org.apache.spark.sql.execution.datasources.FileFormatWriter$.executeTask(FileFormatWriter.scala:311)

... 9 more

24/07/22 09:14:03 ERROR scheduler.TaskSetManager: Task 0 in stage 92.0 failed 1 times; aborting job

24/07/22 09:14:03 ERROR datasources.FileFormatWriter: Aborting job 715cc245-3774-44bb-9e10-d46ffd60e0da.

org.apache.spark.SparkException: Job aborted due to stage failure: Task 0 in stage 92.0 failed 1 times, most recent failure: Lost task 0.0 in stage 92.0 (TID 92) (192.168.112.129 executor driver): org.apache.spark.SparkException: Task failed while writing rows.

at org.apache.spark.sql.errors.QueryExecutionErrors$.taskFailedWhileWritingRowsError(QueryExecutionErrors.scala:500)

at org.apache.spark.sql.execution.datasources.FileFormatWriter$.executeTask(FileFormatWriter.scala:321)

at org.apache.spark.sql.execution.datasources.FileFormatWriter$.$anonfun$write$16(FileFormatWriter.scala:229)

at org.apache.spark.scheduler.ResultTask.runTask(ResultTask.scala:90)

at org.apache.spark.scheduler.Task.run(Task.scala:131)

at org.apache.spark.executor.Executor$TaskRunner.$anonfun$run$3(Executor.scala:506)

at org.apache.spark.util.Utils$.tryWithSafeFinally(Utils.scala:1462)

at org.apache.spark.executor.Executor$TaskRunner.run(Executor.scala:509)

at java.util.concurrent.ThreadPoolExecutor.runWorker(ThreadPoolExecutor.java:1149)

at java.util.concurrent.ThreadPoolExecutor$Worker.run(ThreadPoolExecutor.java:624)

at java.lang.Thread.run(Thread.java:750)

Caused by: org.apache.spark.api.python.PythonException: Traceback (most recent call last):

File "/usr/local/spark/python/lib/pyspark.zip/pyspark/worker.py", line 619, in main

process()

File "/usr/local/spark/python/lib/pyspark.zip/pyspark/worker.py", line 611, in process

serializer.dump\_stream(out\_iter, outfile)

File "/usr/local/spark/python/lib/pyspark.zip/pyspark/serializers.py", line 211, in dump\_stream

self.serializer.dump\_stream(self.\_batched(iterator), stream)

File "/usr/local/spark/python/lib/pyspark.zip/pyspark/serializers.py", line 132, in dump\_stream

for obj in iterator:

File "/usr/local/spark/python/lib/pyspark.zip/pyspark/serializers.py", line 200, in \_batched

for item in iterator:

File "/usr/local/spark/python/lib/pyspark.zip/pyspark/worker.py", line 452, in mapper

result = tuple(f(\*[a[o] for o in arg\_offsets]) for (arg\_offsets, f) in udfs)

File "/usr/local/spark/python/lib/pyspark.zip/pyspark/worker.py", line 452, in <genexpr>

result = tuple(f(\*[a[o] for o in arg\_offsets]) for (arg\_offsets, f) in udfs)

File "/usr/local/spark/python/lib/pyspark.zip/pyspark/worker.py", line 87, in <lambda>

return lambda \*a: f(\*a)

File "/usr/local/spark/python/lib/pyspark.zip/pyspark/util.py", line 74, in wrapper

return f(\*args, \*\*kwargs)

File "<stdin>", line 3, in clean\_string

TypeError: expected string or bytes-like object

at org.apache.spark.api.python.BasePythonRunner$ReaderIterator.handlePythonException(PythonRunner.scala:555)

at org.apache.spark.sql.execution.python.PythonUDFRunner$$anon$2.read(PythonUDFRunner.scala:86)

at org.apache.spark.sql.execution.python.PythonUDFRunner$$anon$2.read(PythonUDFRunner.scala:68)

at org.apache.spark.api.python.BasePythonRunner$ReaderIterator.hasNext(PythonRunner.scala:508)

at org.apache.spark.InterruptibleIterator.hasNext(InterruptibleIterator.scala:37)

at scala.collection.Iterator$$anon$11.hasNext(Iterator.scala:491)

at scala.collection.Iterator$$anon$10.hasNext(Iterator.scala:460)

at scala.collection.Iterator$$anon$10.hasNext(Iterator.scala:460)

at org.apache.spark.sql.catalyst.expressions.GeneratedClass$GeneratedIteratorForCodegenStage1.processNext(Unknown Source)

at org.apache.spark.sql.execution.BufferedRowIterator.hasNext(BufferedRowIterator.java:43)

at org.apache.spark.sql.execution.WholeStageCodegenExec$$anon$1.hasNext(WholeStageCodegenExec.scala:759)

at org.apache.spark.sql.execution.datasources.FileFormatDataWriter.writeWithIterator(FileFormatDataWriter.scala:91)

at org.apache.spark.sql.execution.datasources.FileFormatWriter$.$anonfun$executeTask$1(FileFormatWriter.scala:304)

at org.apache.spark.util.Utils$.tryWithSafeFinallyAndFailureCallbacks(Utils.scala:1496)

at org.apache.spark.sql.execution.datasources.FileFormatWriter$.executeTask(FileFormatWriter.scala:311)

... 9 more

Driver stacktrace:

at org.apache.spark.scheduler.DAGScheduler.failJobAndIndependentStages(DAGScheduler.scala:2454)

at org.apache.spark.scheduler.DAGScheduler.$anonfun$abortStage$2(DAGScheduler.scala:2403)

at org.apache.spark.scheduler.DAGScheduler.$anonfun$abortStage$2$adapted(DAGScheduler.scala:2402)

at scala.collection.mutable.ResizableArray.foreach(ResizableArray.scala:62)

at scala.collection.mutable.ResizableArray.foreach$(ResizableArray.scala:55)

at scala.collection.mutable.ArrayBuffer.foreach(ArrayBuffer.scala:49)

at org.apache.spark.scheduler.DAGScheduler.abortStage(DAGScheduler.scala:2402)

at org.apache.spark.scheduler.DAGScheduler.$anonfun$handleTaskSetFailed$1(DAGScheduler.scala:1160)

at org.apache.spark.scheduler.DAGScheduler.$anonfun$handleTaskSetFailed$1$adapted(DAGScheduler.scala:1160)

at scala.Option.foreach(Option.scala:407)

at org.apache.spark.scheduler.DAGScheduler.handleTaskSetFailed(DAGScheduler.scala:1160)

at org.apache.spark.scheduler.DAGSchedulerEventProcessLoop.doOnReceive(DAGScheduler.scala:2642)

at org.apache.spark.scheduler.DAGSchedulerEventProcessLoop.onReceive(DAGScheduler.scala:2584)

at org.apache.spark.scheduler.DAGSchedulerEventProcessLoop.onReceive(DAGScheduler.scala:2573)

at org.apache.spark.util.EventLoop$$anon$1.run(EventLoop.scala:49)

at org.apache.spark.scheduler.DAGScheduler.runJob(DAGScheduler.scala:938)

at org.apache.spark.SparkContext.runJob(SparkContext.scala:2214)

at org.apache.spark.sql.execution.datasources.FileFormatWriter$.write(FileFormatWriter.scala:218)

at org.apache.spark.sql.execution.datasources.InsertIntoHadoopFsRelationCommand.run(InsertIntoHadoopFsRelationCommand.scala:186)

at org.apache.spark.sql.execution.command.DataWritingCommandExec.sideEffectResult$lzycompute(commands.scala:113)

at org.apache.spark.sql.execution.command.DataWritingCommandExec.sideEffectResult(commands.scala:111)

at org.apache.spark.sql.execution.command.DataWritingCommandExec.executeCollect(commands.scala:125)

at org.apache.spark.sql.execution.QueryExecution$$anonfun$eagerlyExecuteCommands$1.$anonfun$applyOrElse$1(QueryExecution.scala:110)

at org.apache.spark.sql.execution.SQLExecution$.$anonfun$withNewExecutionId$5(SQLExecution.scala:103)

at org.apache.spark.sql.execution.SQLExecution$.withSQLConfPropagated(SQLExecution.scala:163)

at org.apache.spark.sql.execution.SQLExecution$.$anonfun$withNewExecutionId$1(SQLExecution.scala:90)

at org.apache.spark.sql.SparkSession.withActive(SparkSession.scala:775)

at org.apache.spark.sql.execution.SQLExecution$.withNewExecutionId(SQLExecution.scala:64)

at org.apache.spark.sql.execution.QueryExecution$$anonfun$eagerlyExecuteCommands$1.applyOrElse(QueryExecution.scala:110)

at org.apache.spark.sql.execution.QueryExecution$$anonfun$eagerlyExecuteCommands$1.applyOrElse(QueryExecution.scala:106)

at org.apache.spark.sql.catalyst.trees.TreeNode.$anonfun$transformDownWithPruning$1(TreeNode.scala:481)

at org.apache.spark.sql.catalyst.trees.CurrentOrigin$.withOrigin(TreeNode.scala:82)

at org.apache.spark.sql.catalyst.trees.TreeNode.transformDownWithPruning(TreeNode.scala:481)

at org.apache.spark.sql.catalyst.plans.logical.LogicalPlan.org$apache$spark$sql$catalyst$plans$logical$AnalysisHelper$$super$transformDownWithPruning(LogicalPlan.scala:30)

at org.apache.spark.sql.catalyst.plans.logical.AnalysisHelper.transformDownWithPruning(AnalysisHelper.scala:267)

at org.apache.spark.sql.catalyst.plans.logical.AnalysisHelper.transformDownWithPruning$(AnalysisHelper.scala:263)

at org.apache.spark.sql.catalyst.plans.logical.LogicalPlan.transformDownWithPruning(LogicalPlan.scala:30)

at org.apache.spark.sql.catalyst.plans.logical.LogicalPlan.transformDownWithPruning(LogicalPlan.scala:30)

at org.apache.spark.sql.catalyst.trees.TreeNode.transformDown(TreeNode.scala:457)

at org.apache.spark.sql.execution.QueryExecution.eagerlyExecuteCommands(QueryExecution.scala:106)

at org.apache.spark.sql.execution.QueryExecution.commandExecuted$lzycompute(QueryExecution.scala:93)

at org.apache.spark.sql.execution.QueryExecution.commandExecuted(QueryExecution.scala:91)

at org.apache.spark.sql.execution.QueryExecution.assertCommandExecuted(QueryExecution.scala:128)

at org.apache.spark.sql.DataFrameWriter.runCommand(DataFrameWriter.scala:848)

at org.apache.spark.sql.DataFrameWriter.saveToV1Source(DataFrameWriter.scala:382)

at org.apache.spark.sql.DataFrameWriter.saveInternal(DataFrameWriter.scala:355)

at org.apache.spark.sql.DataFrameWriter.save(DataFrameWriter.scala:239)

at org.apache.spark.sql.DataFrameWriter.json(DataFrameWriter.scala:763)

at sun.reflect.NativeMethodAccessorImpl.invoke0(Native Method)

at sun.reflect.NativeMethodAccessorImpl.invoke(NativeMethodAccessorImpl.java:62)

at sun.reflect.DelegatingMethodAccessorImpl.invoke(DelegatingMethodAccessorImpl.java:43)

at java.lang.reflect.Method.invoke(Method.java:498)

at py4j.reflection.MethodInvoker.invoke(MethodInvoker.java:244)

at py4j.reflection.ReflectionEngine.invoke(ReflectionEngine.java:357)

at py4j.Gateway.invoke(Gateway.java:282)

at py4j.commands.AbstractCommand.invokeMethod(AbstractCommand.java:132)

at py4j.commands.CallCommand.execute(CallCommand.java:79)

at py4j.ClientServerConnection.waitForCommands(ClientServerConnection.java:182)

at py4j.ClientServerConnection.run(ClientServerConnection.java:106)

at java.lang.Thread.run(Thread.java:750)

Caused by: org.apache.spark.SparkException: Task failed while writing rows.

at org.apache.spark.sql.errors.QueryExecutionErrors$.taskFailedWhileWritingRowsError(QueryExecutionErrors.scala:500)

at org.apache.spark.sql.execution.datasources.FileFormatWriter$.executeTask(FileFormatWriter.scala:321)

at org.apache.spark.sql.execution.datasources.FileFormatWriter$.$anonfun$write$16(FileFormatWriter.scala:229)

at org.apache.spark.scheduler.ResultTask.runTask(ResultTask.scala:90)

at org.apache.spark.scheduler.Task.run(Task.scala:131)

at org.apache.spark.executor.Executor$TaskRunner.$anonfun$run$3(Executor.scala:506)

at org.apache.spark.util.Utils$.tryWithSafeFinally(Utils.scala:1462)

at org.apache.spark.executor.Executor$TaskRunner.run(Executor.scala:509)

at java.util.concurrent.ThreadPoolExecutor.runWorker(ThreadPoolExecutor.java:1149)

at java.util.concurrent.ThreadPoolExecutor$Worker.run(ThreadPoolExecutor.java:624)

... 1 more

Caused by: org.apache.spark.api.python.PythonException: Traceback (most recent call last):

File "/usr/local/spark/python/lib/pyspark.zip/pyspark/worker.py", line 619, in main

process()

File "/usr/local/spark/python/lib/pyspark.zip/pyspark/worker.py", line 611, in process

serializer.dump\_stream(out\_iter, outfile)

File "/usr/local/spark/python/lib/pyspark.zip/pyspark/serializers.py", line 211, in dump\_stream

self.serializer.dump\_stream(self.\_batched(iterator), stream)

File "/usr/local/spark/python/lib/pyspark.zip/pyspark/serializers.py", line 132, in dump\_stream

for obj in iterator:

File "/usr/local/spark/python/lib/pyspark.zip/pyspark/serializers.py", line 200, in \_batched

for item in iterator:

File "/usr/local/spark/python/lib/pyspark.zip/pyspark/worker.py", line 452, in mapper

result = tuple(f(\*[a[o] for o in arg\_offsets]) for (arg\_offsets, f) in udfs)

File "/usr/local/spark/python/lib/pyspark.zip/pyspark/worker.py", line 452, in <genexpr>

result = tuple(f(\*[a[o] for o in arg\_offsets]) for (arg\_offsets, f) in udfs)

File "/usr/local/spark/python/lib/pyspark.zip/pyspark/worker.py", line 87, in <lambda>

return lambda \*a: f(\*a)

File "/usr/local/spark/python/lib/pyspark.zip/pyspark/util.py", line 74, in wrapper

return f(\*args, \*\*kwargs)

File "<stdin>", line 3, in clean\_string

TypeError: expected string or bytes-like object

at org.apache.spark.api.python.BasePythonRunner$ReaderIterator.handlePythonException(PythonRunner.scala:555)

at org.apache.spark.sql.execution.python.PythonUDFRunner$$anon$2.read(PythonUDFRunner.scala:86)

at org.apache.spark.sql.execution.python.PythonUDFRunner$$anon$2.read(PythonUDFRunner.scala:68)

at org.apache.spark.api.python.BasePythonRunner$ReaderIterator.hasNext(PythonRunner.scala:508)

at org.apache.spark.InterruptibleIterator.hasNext(InterruptibleIterator.scala:37)

at scala.collection.Iterator$$anon$11.hasNext(Iterator.scala:491)

at scala.collection.Iterator$$anon$10.hasNext(Iterator.scala:460)

at scala.collection.Iterator$$anon$10.hasNext(Iterator.scala:460)

at org.apache.spark.sql.catalyst.expressions.GeneratedClass$GeneratedIteratorForCodegenStage1.processNext(Unknown Source)

at org.apache.spark.sql.execution.BufferedRowIterator.hasNext(BufferedRowIterator.java:43)

at org.apache.spark.sql.execution.WholeStageCodegenExec$$anon$1.hasNext(WholeStageCodegenExec.scala:759)

at org.apache.spark.sql.execution.datasources.FileFormatDataWriter.writeWithIterator(FileFormatDataWriter.scala:91)

at org.apache.spark.sql.execution.datasources.FileFormatWriter$.$anonfun$executeTask$1(FileFormatWriter.scala:304)

at org.apache.spark.util.Utils$.tryWithSafeFinallyAndFailureCallbacks(Utils.scala:1496)

at org.apache.spark.sql.execution.datasources.FileFormatWriter$.executeTask(FileFormatWriter.scala:311)

... 9 more

Traceback (most recent call last):

File "<stdin>", line 1, in <module>

File "/usr/local/spark/python/pyspark/sql/readwriter.py", line 846, in json

self.\_jwrite.json(path)

File "/usr/local/spark/python/lib/py4j-0.10.9.3-src.zip/py4j/java\_gateway.py", line 1322, in \_\_call\_\_

File "/usr/local/spark/python/pyspark/sql/utils.py", line 111, in deco

return f(\*a, \*\*kw)

File "/usr/local/spark/python/lib/py4j-0.10.9.3-src.zip/py4j/protocol.py", line 328, in get\_return\_value

py4j.protocol.Py4JJavaError: An error occurred while calling o544.json.

: org.apache.spark.SparkException: Job aborted.

at org.apache.spark.sql.errors.QueryExecutionErrors$.jobAbortedError(QueryExecutionErrors.scala:496)

at org.apache.spark.sql.execution.datasources.FileFormatWriter$.write(FileFormatWriter.scala:251)

at org.apache.spark.sql.execution.datasources.InsertIntoHadoopFsRelationCommand.run(InsertIntoHadoopFsRelationCommand.scala:186)

at org.apache.spark.sql.execution.command.DataWritingCommandExec.sideEffectResult$lzycompute(commands.scala:113)

at org.apache.spark.sql.execution.command.DataWritingCommandExec.sideEffectResult(commands.scala:111)

at org.apache.spark.sql.execution.command.DataWritingCommandExec.executeCollect(commands.scala:125)

at org.apache.spark.sql.execution.QueryExecution$$anonfun$eagerlyExecuteCommands$1.$anonfun$applyOrElse$1(QueryExecution.scala:110)

at org.apache.spark.sql.execution.SQLExecution$.$anonfun$withNewExecutionId$5(SQLExecution.scala:103)

at org.apache.spark.sql.execution.SQLExecution$.withSQLConfPropagated(SQLExecution.scala:163)

at org.apache.spark.sql.execution.SQLExecution$.$anonfun$withNewExecutionId$1(SQLExecution.scala:90)

at org.apache.spark.sql.SparkSession.withActive(SparkSession.scala:775)

at org.apache.spark.sql.execution.SQLExecution$.withNewExecutionId(SQLExecution.scala:64)

at org.apache.spark.sql.execution.QueryExecution$$anonfun$eagerlyExecuteCommands$1.applyOrElse(QueryExecution.scala:110)

at org.apache.spark.sql.execution.QueryExecution$$anonfun$eagerlyExecuteCommands$1.applyOrElse(QueryExecution.scala:106)

at org.apache.spark.sql.catalyst.trees.TreeNode.$anonfun$transformDownWithPruning$1(TreeNode.scala:481)

at org.apache.spark.sql.catalyst.trees.CurrentOrigin$.withOrigin(TreeNode.scala:82)

at org.apache.spark.sql.catalyst.trees.TreeNode.transformDownWithPruning(TreeNode.scala:481)

at org.apache.spark.sql.catalyst.plans.logical.LogicalPlan.org$apache$spark$sql$catalyst$plans$logical$AnalysisHelper$$super$transformDownWithPruning(LogicalPlan.scala:30)

at org.apache.spark.sql.catalyst.plans.logical.AnalysisHelper.transformDownWithPruning(AnalysisHelper.scala:267)

at org.apache.spark.sql.catalyst.plans.logical.AnalysisHelper.transformDownWithPruning$(AnalysisHelper.scala:263)

at org.apache.spark.sql.catalyst.plans.logical.LogicalPlan.transformDownWithPruning(LogicalPlan.scala:30)

at org.apache.spark.sql.catalyst.plans.logical.LogicalPlan.transformDownWithPruning(LogicalPlan.scala:30)

at org.apache.spark.sql.catalyst.trees.TreeNode.transformDown(TreeNode.scala:457)

at org.apache.spark.sql.execution.QueryExecution.eagerlyExecuteCommands(QueryExecution.scala:106)

at org.apache.spark.sql.execution.QueryExecution.commandExecuted$lzycompute(QueryExecution.scala:93)

at org.apache.spark.sql.execution.QueryExecution.commandExecuted(QueryExecution.scala:91)

at org.apache.spark.sql.execution.QueryExecution.assertCommandExecuted(QueryExecution.scala:128)

at org.apache.spark.sql.DataFrameWriter.runCommand(DataFrameWriter.scala:848)

at org.apache.spark.sql.DataFrameWriter.saveToV1Source(DataFrameWriter.scala:382)

at org.apache.spark.sql.DataFrameWriter.saveInternal(DataFrameWriter.scala:355)

at org.apache.spark.sql.DataFrameWriter.save(DataFrameWriter.scala:239)

at org.apache.spark.sql.DataFrameWriter.json(DataFrameWriter.scala:763)

at sun.reflect.NativeMethodAccessorImpl.invoke0(Native Method)

at sun.reflect.NativeMethodAccessorImpl.invoke(NativeMethodAccessorImpl.java:62)

at sun.reflect.DelegatingMethodAccessorImpl.invoke(DelegatingMethodAccessorImpl.java:43)

at java.lang.reflect.Method.invoke(Method.java:498)

at py4j.reflection.MethodInvoker.invoke(MethodInvoker.java:244)

at py4j.reflection.ReflectionEngine.invoke(ReflectionEngine.java:357)

at py4j.Gateway.invoke(Gateway.java:282)

at py4j.commands.AbstractCommand.invokeMethod(AbstractCommand.java:132)

at py4j.commands.CallCommand.execute(CallCommand.java:79)

at py4j.ClientServerConnection.waitForCommands(ClientServerConnection.java:182)

at py4j.ClientServerConnection.run(ClientServerConnection.java:106)

at java.lang.Thread.run(Thread.java:750)

Caused by: org.apache.spark.SparkException: Job aborted due to stage failure: Task 0 in stage 92.0 failed 1 times, most recent failure: Lost task 0.0 in stage 92.0 (TID 92) (192.168.112.129 executor driver): org.apache.spark.SparkException: Task failed while writing rows.

at org.apache.spark.sql.errors.QueryExecutionErrors$.taskFailedWhileWritingRowsError(QueryExecutionErrors.scala:500)

at org.apache.spark.sql.execution.datasources.FileFormatWriter$.executeTask(FileFormatWriter.scala:321)

at org.apache.spark.sql.execution.datasources.FileFormatWriter$.$anonfun$write$16(FileFormatWriter.scala:229)

at org.apache.spark.scheduler.ResultTask.runTask(ResultTask.scala:90)

at org.apache.spark.scheduler.Task.run(Task.scala:131)

at org.apache.spark.executor.Executor$TaskRunner.$anonfun$run$3(Executor.scala:506)

at org.apache.spark.util.Utils$.tryWithSafeFinally(Utils.scala:1462)

at org.apache.spark.executor.Executor$TaskRunner.run(Executor.scala:509)

at java.util.concurrent.ThreadPoolExecutor.runWorker(ThreadPoolExecutor.java:1149)

at java.util.concurrent.ThreadPoolExecutor$Worker.run(ThreadPoolExecutor.java:624)

at java.lang.Thread.run(Thread.java:750)

Caused by: org.apache.spark.api.python.PythonException: Traceback (most recent call last):

File "/usr/local/spark/python/lib/pyspark.zip/pyspark/worker.py", line 619, in main

process()

File "/usr/local/spark/python/lib/pyspark.zip/pyspark/worker.py", line 611, in process

serializer.dump\_stream(out\_iter, outfile)

File "/usr/local/spark/python/lib/pyspark.zip/pyspark/serializers.py", line 211, in dump\_stream

self.serializer.dump\_stream(self.\_batched(iterator), stream)

File "/usr/local/spark/python/lib/pyspark.zip/pyspark/serializers.py", line 132, in dump\_stream

for obj in iterator:

File "/usr/local/spark/python/lib/pyspark.zip/pyspark/serializers.py", line 200, in \_batched

for item in iterator:

File "/usr/local/spark/python/lib/pyspark.zip/pyspark/worker.py", line 452, in mapper

result = tuple(f(\*[a[o] for o in arg\_offsets]) for (arg\_offsets, f) in udfs)

File "/usr/local/spark/python/lib/pyspark.zip/pyspark/worker.py", line 452, in <genexpr>

result = tuple(f(\*[a[o] for o in arg\_offsets]) for (arg\_offsets, f) in udfs)

File "/usr/local/spark/python/lib/pyspark.zip/pyspark/worker.py", line 87, in <lambda>

return lambda \*a: f(\*a)

File "/usr/local/spark/python/lib/pyspark.zip/pyspark/util.py", line 74, in wrapper

return f(\*args, \*\*kwargs)

File "<stdin>", line 3, in clean\_string

TypeError: expected string or bytes-like object

at org.apache.spark.api.python.BasePythonRunner$ReaderIterator.handlePythonException(PythonRunner.scala:555)

at org.apache.spark.sql.execution.python.PythonUDFRunner$$anon$2.read(PythonUDFRunner.scala:86)

at org.apache.spark.sql.execution.python.PythonUDFRunner$$anon$2.read(PythonUDFRunner.scala:68)

at org.apache.spark.api.python.BasePythonRunner$ReaderIterator.hasNext(PythonRunner.scala:508)

at org.apache.spark.InterruptibleIterator.hasNext(InterruptibleIterator.scala:37)

at scala.collection.Iterator$$anon$11.hasNext(Iterator.scala:491)

at scala.collection.Iterator$$anon$10.hasNext(Iterator.scala:460)

at scala.collection.Iterator$$anon$10.hasNext(Iterator.scala:460)

at org.apache.spark.sql.catalyst.expressions.GeneratedClass$GeneratedIteratorForCodegenStage1.processNext(Unknown Source)

at org.apache.spark.sql.execution.BufferedRowIterator.hasNext(BufferedRowIterator.java:43)

at org.apache.spark.sql.execution.WholeStageCodegenExec$$anon$1.hasNext(WholeStageCodegenExec.scala:759)

at org.apache.spark.sql.execution.datasources.FileFormatDataWriter.writeWithIterator(FileFormatDataWriter.scala:91)

at org.apache.spark.sql.execution.datasources.FileFormatWriter$.$anonfun$executeTask$1(FileFormatWriter.scala:304)

at org.apache.spark.util.Utils$.tryWithSafeFinallyAndFailureCallbacks(Utils.scala:1496)

at org.apache.spark.sql.execution.datasources.FileFormatWriter$.executeTask(FileFormatWriter.scala:311)

... 9 more

Driver stacktrace:

at org.apache.spark.scheduler.DAGScheduler.failJobAndIndependentStages(DAGScheduler.scala:2454)

at org.apache.spark.scheduler.DAGScheduler.$anonfun$abortStage$2(DAGScheduler.scala:2403)

at org.apache.spark.scheduler.DAGScheduler.$anonfun$abortStage$2$adapted(DAGScheduler.scala:2402)

at scala.collection.mutable.ResizableArray.foreach(ResizableArray.scala:62)

at scala.collection.mutable.ResizableArray.foreach$(ResizableArray.scala:55)

at scala.collection.mutable.ArrayBuffer.foreach(ArrayBuffer.scala:49)

at org.apache.spark.scheduler.DAGScheduler.abortStage(DAGScheduler.scala:2402)

at org.apache.spark.scheduler.DAGScheduler.$anonfun$handleTaskSetFailed$1(DAGScheduler.scala:1160)

at org.apache.spark.scheduler.DAGScheduler.$anonfun$handleTaskSetFailed$1$adapted(DAGScheduler.scala:1160)

at scala.Option.foreach(Option.scala:407)

at org.apache.spark.scheduler.DAGScheduler.handleTaskSetFailed(DAGScheduler.scala:1160)

at org.apache.spark.scheduler.DAGSchedulerEventProcessLoop.doOnReceive(DAGScheduler.scala:2642)

at org.apache.spark.scheduler.DAGSchedulerEventProcessLoop.onReceive(DAGScheduler.scala:2584)

at org.apache.spark.scheduler.DAGSchedulerEventProcessLoop.onReceive(DAGScheduler.scala:2573)

at org.apache.spark.util.EventLoop$$anon$1.run(EventLoop.scala:49)

at org.apache.spark.scheduler.DAGScheduler.runJob(DAGScheduler.scala:938)

at org.apache.spark.SparkContext.runJob(SparkContext.scala:2214)

at org.apache.spark.sql.execution.datasources.FileFormatWriter$.write(FileFormatWriter.scala:218)

... 42 more

Caused by: org.apache.spark.SparkException: Task failed while writing rows.

at org.apache.spark.sql.errors.QueryExecutionErrors$.taskFailedWhileWritingRowsError(QueryExecutionErrors.scala:500)

at org.apache.spark.sql.execution.datasources.FileFormatWriter$.executeTask(FileFormatWriter.scala:321)

at org.apache.spark.sql.execution.datasources.FileFormatWriter$.$anonfun$write$16(FileFormatWriter.scala:229)

at org.apache.spark.scheduler.ResultTask.runTask(ResultTask.scala:90)

at org.apache.spark.scheduler.Task.run(Task.scala:131)

at org.apache.spark.executor.Executor$TaskRunner.$anonfun$run$3(Executor.scala:506)

at org.apache.spark.util.Utils$.tryWithSafeFinally(Utils.scala:1462)

at org.apache.spark.executor.Executor$TaskRunner.run(Executor.scala:509)

at java.util.concurrent.ThreadPoolExecutor.runWorker(ThreadPoolExecutor.java:1149)

at java.util.concurrent.ThreadPoolExecutor$Worker.run(ThreadPoolExecutor.java:624)

... 1 more

Caused by: org.apache.spark.api.python.PythonException: Traceback (most recent call last):

File "/usr/local/spark/python/lib/pyspark.zip/pyspark/worker.py", line 619, in main

process()

File "/usr/local/spark/python/lib/pyspark.zip/pyspark/worker.py", line 611, in process

serializer.dump\_stream(out\_iter, outfile)

File "/usr/local/spark/python/lib/pyspark.zip/pyspark/serializers.py", line 211, in dump\_stream

self.serializer.dump\_stream(self.\_batched(iterator), stream)

File "/usr/local/spark/python/lib/pyspark.zip/pyspark/serializers.py", line 132, in dump\_stream

for obj in iterator:

File "/usr/local/spark/python/lib/pyspark.zip/pyspark/serializers.py", line 200, in \_batched

for item in iterator:

File "/usr/local/spark/python/lib/pyspark.zip/pyspark/worker.py", line 452, in mapper

result = tuple(f(\*[a[o] for o in arg\_offsets]) for (arg\_offsets, f) in udfs)

File "/usr/local/spark/python/lib/pyspark.zip/pyspark/worker.py", line 452, in <genexpr>

result = tuple(f(\*[a[o] for o in arg\_offsets]) for (arg\_offsets, f) in udfs)

File "/usr/local/spark/python/lib/pyspark.zip/pyspark/worker.py", line 87, in <lambda>

return lambda \*a: f(\*a)

File "/usr/local/spark/python/lib/pyspark.zip/pyspark/util.py", line 74, in wrapper

return f(\*args, \*\*kwargs)

File "<stdin>", line 3, in clean\_string

TypeError: expected string or bytes-like object

at org.apache.spark.api.python.BasePythonRunner$ReaderIterator.handlePythonException(PythonRunner.scala:555)

at org.apache.spark.sql.execution.python.PythonUDFRunner$$anon$2.read(PythonUDFRunner.scala:86)

at org.apache.spark.sql.execution.python.PythonUDFRunner$$anon$2.read(PythonUDFRunner.scala:68)

at org.apache.spark.api.python.BasePythonRunner$ReaderIterator.hasNext(PythonRunner.scala:508)

at org.apache.spark.InterruptibleIterator.hasNext(InterruptibleIterator.scala:37)

at scala.collection.Iterator$$anon$11.hasNext(Iterator.scala:491)

at scala.collection.Iterator$$anon$10.hasNext(Iterator.scala:460)

at scala.collection.Iterator$$anon$10.hasNext(Iterator.scala:460)

at org.apache.spark.sql.catalyst.expressions.GeneratedClass$GeneratedIteratorForCodegenStage1.processNext(Unknown Source)

at org.apache.spark.sql.execution.BufferedRowIterator.hasNext(BufferedRowIterator.java:43)

at org.apache.spark.sql.execution.WholeStageCodegenExec$$anon$1.hasNext(WholeStageCodegenExec.scala:759)

at org.apache.spark.sql.execution.datasources.FileFormatDataWriter.writeWithIterator(FileFormatDataWriter.scala:91)

at org.apache.spark.sql.execution.datasources.FileFormatWriter$.$anonfun$executeTask$1(FileFormatWriter.scala:304)

at org.apache.spark.util.Utils$.tryWithSafeFinallyAndFailureCallbacks(Utils.scala:1496)

at org.apache.spark.sql.execution.datasources.FileFormatWriter$.executeTask(FileFormatWriter.scala:311)

... 9 more

>>> sample\_op.write.json("/home/hduser/Hackathon/JSON\_DATA", mode="overwrite")

24/07/22 09:24:37 WARN storage.BlockManager: Putting block rdd\_531\_0 failed due to exception org.apache.spark.api.python.PythonException: Traceback (most recent call last):

File "/usr/local/spark/python/lib/pyspark.zip/pyspark/worker.py", line 619, in main

process()

File "/usr/local/spark/python/lib/pyspark.zip/pyspark/worker.py", line 611, in process

serializer.dump\_stream(out\_iter, outfile)

File "/usr/local/spark/python/lib/pyspark.zip/pyspark/serializers.py", line 211, in dump\_stream

self.serializer.dump\_stream(self.\_batched(iterator), stream)

File "/usr/local/spark/python/lib/pyspark.zip/pyspark/serializers.py", line 132, in dump\_stream

for obj in iterator:

File "/usr/local/spark/python/lib/pyspark.zip/pyspark/serializers.py", line 200, in \_batched

for item in iterator:

File "/usr/local/spark/python/lib/pyspark.zip/pyspark/worker.py", line 452, in mapper

result = tuple(f(\*[a[o] for o in arg\_offsets]) for (arg\_offsets, f) in udfs)

File "/usr/local/spark/python/lib/pyspark.zip/pyspark/worker.py", line 452, in <genexpr>

result = tuple(f(\*[a[o] for o in arg\_offsets]) for (arg\_offsets, f) in udfs)

File "/usr/local/spark/python/lib/pyspark.zip/pyspark/worker.py", line 87, in <lambda>

return lambda \*a: f(\*a)

File "/usr/local/spark/python/lib/pyspark.zip/pyspark/util.py", line 74, in wrapper

return f(\*args, \*\*kwargs)

File "<stdin>", line 3, in clean\_string

TypeError: expected string or bytes-like object

.

24/07/22 09:24:37 WARN storage.BlockManager: Block rdd\_531\_0 could not be removed as it was not found on disk or in memory

24/07/22 09:24:37 ERROR executor.Executor: Exception in task 0.0 in stage 98.0 (TID 103)

org.apache.spark.api.python.PythonException: Traceback (most recent call last):

File "/usr/local/spark/python/lib/pyspark.zip/pyspark/worker.py", line 619, in main

process()

File "/usr/local/spark/python/lib/pyspark.zip/pyspark/worker.py", line 611, in process

serializer.dump\_stream(out\_iter, outfile)

File "/usr/local/spark/python/lib/pyspark.zip/pyspark/serializers.py", line 211, in dump\_stream

self.serializer.dump\_stream(self.\_batched(iterator), stream)

File "/usr/local/spark/python/lib/pyspark.zip/pyspark/serializers.py", line 132, in dump\_stream

for obj in iterator:

File "/usr/local/spark/python/lib/pyspark.zip/pyspark/serializers.py", line 200, in \_batched

for item in iterator:

File "/usr/local/spark/python/lib/pyspark.zip/pyspark/worker.py", line 452, in mapper

result = tuple(f(\*[a[o] for o in arg\_offsets]) for (arg\_offsets, f) in udfs)

File "/usr/local/spark/python/lib/pyspark.zip/pyspark/worker.py", line 452, in <genexpr>

result = tuple(f(\*[a[o] for o in arg\_offsets]) for (arg\_offsets, f) in udfs)

File "/usr/local/spark/python/lib/pyspark.zip/pyspark/worker.py", line 87, in <lambda>

return lambda \*a: f(\*a)

File "/usr/local/spark/python/lib/pyspark.zip/pyspark/util.py", line 74, in wrapper

return f(\*args, \*\*kwargs)

File "<stdin>", line 3, in clean\_string

TypeError: expected string or bytes-like object

at org.apache.spark.api.python.BasePythonRunner$ReaderIterator.handlePythonException(PythonRunner.scala:555)

at org.apache.spark.sql.execution.python.PythonUDFRunner$$anon$2.read(PythonUDFRunner.scala:86)

at org.apache.spark.sql.execution.python.PythonUDFRunner$$anon$2.read(PythonUDFRunner.scala:68)

at org.apache.spark.api.python.BasePythonRunner$ReaderIterator.hasNext(PythonRunner.scala:508)

at org.apache.spark.InterruptibleIterator.hasNext(InterruptibleIterator.scala:37)

at scala.collection.Iterator$$anon$11.hasNext(Iterator.scala:491)

at scala.collection.Iterator$$anon$10.hasNext(Iterator.scala:460)

at scala.collection.Iterator$$anon$10.hasNext(Iterator.scala:460)

at org.apache.spark.sql.catalyst.expressions.GeneratedClass$GeneratedIteratorForCodegenStage1.processNext(Unknown Source)

at org.apache.spark.sql.execution.BufferedRowIterator.hasNext(BufferedRowIterator.java:43)

at org.apache.spark.sql.execution.WholeStageCodegenExec$$anon$1.hasNext(WholeStageCodegenExec.scala:759)

at org.apache.spark.sql.execution.columnar.DefaultCachedBatchSerializer$$anon$1.next(InMemoryRelation.scala:87)

at org.apache.spark.sql.execution.columnar.DefaultCachedBatchSerializer$$anon$1.next(InMemoryRelation.scala:79)

at scala.collection.Iterator$$anon$10.next(Iterator.scala:461)

at org.apache.spark.storage.memory.MemoryStore.putIterator(MemoryStore.scala:224)

at org.apache.spark.storage.memory.MemoryStore.putIteratorAsValues(MemoryStore.scala:302)

at org.apache.spark.storage.BlockManager.$anonfun$doPutIterator$1(BlockManager.scala:1481)

at org.apache.spark.storage.BlockManager.org$apache$spark$storage$BlockManager$$doPut(BlockManager.scala:1408)

at org.apache.spark.storage.BlockManager.doPutIterator(BlockManager.scala:1472)

at org.apache.spark.storage.BlockManager.getOrElseUpdate(BlockManager.scala:1295)

at org.apache.spark.rdd.RDD.getOrCompute(RDD.scala:384)

at org.apache.spark.rdd.RDD.iterator(RDD.scala:335)

at org.apache.spark.rdd.MapPartitionsRDD.compute(MapPartitionsRDD.scala:52)

at org.apache.spark.rdd.RDD.computeOrReadCheckpoint(RDD.scala:373)

at org.apache.spark.rdd.RDD.iterator(RDD.scala:337)

at org.apache.spark.rdd.MapPartitionsRDD.compute(MapPartitionsRDD.scala:52)

at org.apache.spark.rdd.RDD.computeOrReadCheckpoint(RDD.scala:373)

at org.apache.spark.rdd.RDD.iterator(RDD.scala:337)

at org.apache.spark.rdd.MapPartitionsRDD.compute(MapPartitionsRDD.scala:52)

at org.apache.spark.rdd.RDD.computeOrReadCheckpoint(RDD.scala:373)

at org.apache.spark.rdd.RDD.iterator(RDD.scala:337)

at org.apache.spark.scheduler.ResultTask.runTask(ResultTask.scala:90)

at org.apache.spark.scheduler.Task.run(Task.scala:131)

at org.apache.spark.executor.Executor$TaskRunner.$anonfun$run$3(Executor.scala:506)

at org.apache.spark.util.Utils$.tryWithSafeFinally(Utils.scala:1462)

at org.apache.spark.executor.Executor$TaskRunner.run(Executor.scala:509)

at java.util.concurrent.ThreadPoolExecutor.runWorker(ThreadPoolExecutor.java:1149)

at java.util.concurrent.ThreadPoolExecutor$Worker.run(ThreadPoolExecutor.java:624)

at java.lang.Thread.run(Thread.java:750)

24/07/22 09:24:37 WARN scheduler.TaskSetManager: Lost task 0.0 in stage 98.0 (TID 103) (192.168.112.129 executor driver): org.apache.spark.api.python.PythonException: Traceback (most recent call last):

File "/usr/local/spark/python/lib/pyspark.zip/pyspark/worker.py", line 619, in main

process()

File "/usr/local/spark/python/lib/pyspark.zip/pyspark/worker.py", line 611, in process

serializer.dump\_stream(out\_iter, outfile)

File "/usr/local/spark/python/lib/pyspark.zip/pyspark/serializers.py", line 211, in dump\_stream

self.serializer.dump\_stream(self.\_batched(iterator), stream)

File "/usr/local/spark/python/lib/pyspark.zip/pyspark/serializers.py", line 132, in dump\_stream

for obj in iterator:

File "/usr/local/spark/python/lib/pyspark.zip/pyspark/serializers.py", line 200, in \_batched

for item in iterator:

File "/usr/local/spark/python/lib/pyspark.zip/pyspark/worker.py", line 452, in mapper

result = tuple(f(\*[a[o] for o in arg\_offsets]) for (arg\_offsets, f) in udfs)

File "/usr/local/spark/python/lib/pyspark.zip/pyspark/worker.py", line 452, in <genexpr>

result = tuple(f(\*[a[o] for o in arg\_offsets]) for (arg\_offsets, f) in udfs)

File "/usr/local/spark/python/lib/pyspark.zip/pyspark/worker.py", line 87, in <lambda>

return lambda \*a: f(\*a)

File "/usr/local/spark/python/lib/pyspark.zip/pyspark/util.py", line 74, in wrapper

return f(\*args, \*\*kwargs)

File "<stdin>", line 3, in clean\_string

TypeError: expected string or bytes-like object

at org.apache.spark.api.python.BasePythonRunner$ReaderIterator.handlePythonException(PythonRunner.scala:555)

at org.apache.spark.sql.execution.python.PythonUDFRunner$$anon$2.read(PythonUDFRunner.scala:86)

at org.apache.spark.sql.execution.python.PythonUDFRunner$$anon$2.read(PythonUDFRunner.scala:68)

at org.apache.spark.api.python.BasePythonRunner$ReaderIterator.hasNext(PythonRunner.scala:508)

at org.apache.spark.InterruptibleIterator.hasNext(InterruptibleIterator.scala:37)

at scala.collection.Iterator$$anon$11.hasNext(Iterator.scala:491)

at scala.collection.Iterator$$anon$10.hasNext(Iterator.scala:460)

at scala.collection.Iterator$$anon$10.hasNext(Iterator.scala:460)

at org.apache.spark.sql.catalyst.expressions.GeneratedClass$GeneratedIteratorForCodegenStage1.processNext(Unknown Source)

at org.apache.spark.sql.execution.BufferedRowIterator.hasNext(BufferedRowIterator.java:43)

at org.apache.spark.sql.execution.WholeStageCodegenExec$$anon$1.hasNext(WholeStageCodegenExec.scala:759)

at org.apache.spark.sql.execution.columnar.DefaultCachedBatchSerializer$$anon$1.next(InMemoryRelation.scala:87)

at org.apache.spark.sql.execution.columnar.DefaultCachedBatchSerializer$$anon$1.next(InMemoryRelation.scala:79)

at scala.collection.Iterator$$anon$10.next(Iterator.scala:461)

at org.apache.spark.storage.memory.MemoryStore.putIterator(MemoryStore.scala:224)

at org.apache.spark.storage.memory.MemoryStore.putIteratorAsValues(MemoryStore.scala:302)

at org.apache.spark.storage.BlockManager.$anonfun$doPutIterator$1(BlockManager.scala:1481)

at org.apache.spark.storage.BlockManager.org$apache$spark$storage$BlockManager$$doPut(BlockManager.scala:1408)

at org.apache.spark.storage.BlockManager.doPutIterator(BlockManager.scala:1472)

at org.apache.spark.storage.BlockManager.getOrElseUpdate(BlockManager.scala:1295)

at org.apache.spark.rdd.RDD.getOrCompute(RDD.scala:384)

at org.apache.spark.rdd.RDD.iterator(RDD.scala:335)

at org.apache.spark.rdd.MapPartitionsRDD.compute(MapPartitionsRDD.scala:52)

at org.apache.spark.rdd.RDD.computeOrReadCheckpoint(RDD.scala:373)

at org.apache.spark.rdd.RDD.iterator(RDD.scala:337)

at org.apache.spark.rdd.MapPartitionsRDD.compute(MapPartitionsRDD.scala:52)

at org.apache.spark.rdd.RDD.computeOrReadCheckpoint(RDD.scala:373)

at org.apache.spark.rdd.RDD.iterator(RDD.scala:337)

at org.apache.spark.rdd.MapPartitionsRDD.compute(MapPartitionsRDD.scala:52)

at org.apache.spark.rdd.RDD.computeOrReadCheckpoint(RDD.scala:373)

at org.apache.spark.rdd.RDD.iterator(RDD.scala:337)

at org.apache.spark.scheduler.ResultTask.runTask(ResultTask.scala:90)

at org.apache.spark.scheduler.Task.run(Task.scala:131)

at org.apache.spark.executor.Executor$TaskRunner.$anonfun$run$3(Executor.scala:506)

at org.apache.spark.util.Utils$.tryWithSafeFinally(Utils.scala:1462)

at org.apache.spark.executor.Executor$TaskRunner.run(Executor.scala:509)

at java.util.concurrent.ThreadPoolExecutor.runWorker(ThreadPoolExecutor.java:1149)

at java.util.concurrent.ThreadPoolExecutor$Worker.run(ThreadPoolExecutor.java:624)

at java.lang.Thread.run(Thread.java:750)

24/07/22 09:24:37 ERROR scheduler.TaskSetManager: Task 0 in stage 98.0 failed 1 times; aborting job

24/07/22 09:24:37 ERROR datasources.FileFormatWriter: Aborting job 6e506df8-fe50-45b4-9d7a-e320aa8ef116.

org.apache.spark.SparkException: Job aborted due to stage failure: Task 0 in stage 98.0 failed 1 times, most recent failure: Lost task 0.0 in stage 98.0 (TID 103) (192.168.112.129 executor driver): org.apache.spark.api.python.PythonException: Traceback (most recent call last):

File "/usr/local/spark/python/lib/pyspark.zip/pyspark/worker.py", line 619, in main

process()

File "/usr/local/spark/python/lib/pyspark.zip/pyspark/worker.py", line 611, in process

serializer.dump\_stream(out\_iter, outfile)

File "/usr/local/spark/python/lib/pyspark.zip/pyspark/serializers.py", line 211, in dump\_stream

self.serializer.dump\_stream(self.\_batched(iterator), stream)

File "/usr/local/spark/python/lib/pyspark.zip/pyspark/serializers.py", line 132, in dump\_stream

for obj in iterator:

File "/usr/local/spark/python/lib/pyspark.zip/pyspark/serializers.py", line 200, in \_batched

for item in iterator:

File "/usr/local/spark/python/lib/pyspark.zip/pyspark/worker.py", line 452, in mapper

result = tuple(f(\*[a[o] for o in arg\_offsets]) for (arg\_offsets, f) in udfs)

File "/usr/local/spark/python/lib/pyspark.zip/pyspark/worker.py", line 452, in <genexpr>

result = tuple(f(\*[a[o] for o in arg\_offsets]) for (arg\_offsets, f) in udfs)

File "/usr/local/spark/python/lib/pyspark.zip/pyspark/worker.py", line 87, in <lambda>

return lambda \*a: f(\*a)

File "/usr/local/spark/python/lib/pyspark.zip/pyspark/util.py", line 74, in wrapper

return f(\*args, \*\*kwargs)

File "<stdin>", line 3, in clean\_string

TypeError: expected string or bytes-like object

at org.apache.spark.api.python.BasePythonRunner$ReaderIterator.handlePythonException(PythonRunner.scala:555)

at org.apache.spark.sql.execution.python.PythonUDFRunner$$anon$2.read(PythonUDFRunner.scala:86)

at org.apache.spark.sql.execution.python.PythonUDFRunner$$anon$2.read(PythonUDFRunner.scala:68)

at org.apache.spark.api.python.BasePythonRunner$ReaderIterator.hasNext(PythonRunner.scala:508)

at org.apache.spark.InterruptibleIterator.hasNext(InterruptibleIterator.scala:37)

at scala.collection.Iterator$$anon$11.hasNext(Iterator.scala:491)

at scala.collection.Iterator$$anon$10.hasNext(Iterator.scala:460)

at scala.collection.Iterator$$anon$10.hasNext(Iterator.scala:460)

at org.apache.spark.sql.catalyst.expressions.GeneratedClass$GeneratedIteratorForCodegenStage1.processNext(Unknown Source)

at org.apache.spark.sql.execution.BufferedRowIterator.hasNext(BufferedRowIterator.java:43)

at org.apache.spark.sql.execution.WholeStageCodegenExec$$anon$1.hasNext(WholeStageCodegenExec.scala:759)

at org.apache.spark.sql.execution.columnar.DefaultCachedBatchSerializer$$anon$1.next(InMemoryRelation.scala:87)

at org.apache.spark.sql.execution.columnar.DefaultCachedBatchSerializer$$anon$1.next(InMemoryRelation.scala:79)

at scala.collection.Iterator$$anon$10.next(Iterator.scala:461)

at org.apache.spark.storage.memory.MemoryStore.putIterator(MemoryStore.scala:224)

at org.apache.spark.storage.memory.MemoryStore.putIteratorAsValues(MemoryStore.scala:302)

at org.apache.spark.storage.BlockManager.$anonfun$doPutIterator$1(BlockManager.scala:1481)

at org.apache.spark.storage.BlockManager.org$apache$spark$storage$BlockManager$$doPut(BlockManager.scala:1408)

at org.apache.spark.storage.BlockManager.doPutIterator(BlockManager.scala:1472)

at org.apache.spark.storage.BlockManager.getOrElseUpdate(BlockManager.scala:1295)

at org.apache.spark.rdd.RDD.getOrCompute(RDD.scala:384)

at org.apache.spark.rdd.RDD.iterator(RDD.scala:335)

at org.apache.spark.rdd.MapPartitionsRDD.compute(MapPartitionsRDD.scala:52)

at org.apache.spark.rdd.RDD.computeOrReadCheckpoint(RDD.scala:373)

at org.apache.spark.rdd.RDD.iterator(RDD.scala:337)

at org.apache.spark.rdd.MapPartitionsRDD.compute(MapPartitionsRDD.scala:52)

at org.apache.spark.rdd.RDD.computeOrReadCheckpoint(RDD.scala:373)

at org.apache.spark.rdd.RDD.iterator(RDD.scala:337)

at org.apache.spark.rdd.MapPartitionsRDD.compute(MapPartitionsRDD.scala:52)

at org.apache.spark.rdd.RDD.computeOrReadCheckpoint(RDD.scala:373)

at org.apache.spark.rdd.RDD.iterator(RDD.scala:337)

at org.apache.spark.scheduler.ResultTask.runTask(ResultTask.scala:90)

at org.apache.spark.scheduler.Task.run(Task.scala:131)

at org.apache.spark.executor.Executor$TaskRunner.$anonfun$run$3(Executor.scala:506)

at org.apache.spark.util.Utils$.tryWithSafeFinally(Utils.scala:1462)

at org.apache.spark.executor.Executor$TaskRunner.run(Executor.scala:509)

at java.util.concurrent.ThreadPoolExecutor.runWorker(ThreadPoolExecutor.java:1149)

at java.util.concurrent.ThreadPoolExecutor$Worker.run(ThreadPoolExecutor.java:624)

at java.lang.Thread.run(Thread.java:750)

Driver stacktrace:

at org.apache.spark.scheduler.DAGScheduler.failJobAndIndependentStages(DAGScheduler.scala:2454)

at org.apache.spark.scheduler.DAGScheduler.$anonfun$abortStage$2(DAGScheduler.scala:2403)

at org.apache.spark.scheduler.DAGScheduler.$anonfun$abortStage$2$adapted(DAGScheduler.scala:2402)

at scala.collection.mutable.ResizableArray.foreach(ResizableArray.scala:62)

at scala.collection.mutable.ResizableArray.foreach$(ResizableArray.scala:55)

at scala.collection.mutable.ArrayBuffer.foreach(ArrayBuffer.scala:49)

at org.apache.spark.scheduler.DAGScheduler.abortStage(DAGScheduler.scala:2402)

at org.apache.spark.scheduler.DAGScheduler.$anonfun$handleTaskSetFailed$1(DAGScheduler.scala:1160)

at org.apache.spark.scheduler.DAGScheduler.$anonfun$handleTaskSetFailed$1$adapted(DAGScheduler.scala:1160)

at scala.Option.foreach(Option.scala:407)

at org.apache.spark.scheduler.DAGScheduler.handleTaskSetFailed(DAGScheduler.scala:1160)

at org.apache.spark.scheduler.DAGSchedulerEventProcessLoop.doOnReceive(DAGScheduler.scala:2642)

at org.apache.spark.scheduler.DAGSchedulerEventProcessLoop.onReceive(DAGScheduler.scala:2584)

at org.apache.spark.scheduler.DAGSchedulerEventProcessLoop.onReceive(DAGScheduler.scala:2573)

at org.apache.spark.util.EventLoop$$anon$1.run(EventLoop.scala:49)

at org.apache.spark.scheduler.DAGScheduler.runJob(DAGScheduler.scala:938)

at org.apache.spark.SparkContext.runJob(SparkContext.scala:2214)

at org.apache.spark.sql.execution.datasources.FileFormatWriter$.write(FileFormatWriter.scala:218)

at org.apache.spark.sql.execution.datasources.InsertIntoHadoopFsRelationCommand.run(InsertIntoHadoopFsRelationCommand.scala:186)

at org.apache.spark.sql.execution.command.DataWritingCommandExec.sideEffectResult$lzycompute(commands.scala:113)

at org.apache.spark.sql.execution.command.DataWritingCommandExec.sideEffectResult(commands.scala:111)

at org.apache.spark.sql.execution.command.DataWritingCommandExec.executeCollect(commands.scala:125)

at org.apache.spark.sql.execution.QueryExecution$$anonfun$eagerlyExecuteCommands$1.$anonfun$applyOrElse$1(QueryExecution.scala:110)

at org.apache.spark.sql.execution.SQLExecution$.$anonfun$withNewExecutionId$5(SQLExecution.scala:103)

at org.apache.spark.sql.execution.SQLExecution$.withSQLConfPropagated(SQLExecution.scala:163)

at org.apache.spark.sql.execution.SQLExecution$.$anonfun$withNewExecutionId$1(SQLExecution.scala:90)

at org.apache.spark.sql.SparkSession.withActive(SparkSession.scala:775)

at org.apache.spark.sql.execution.SQLExecution$.withNewExecutionId(SQLExecution.scala:64)

at org.apache.spark.sql.execution.QueryExecution$$anonfun$eagerlyExecuteCommands$1.applyOrElse(QueryExecution.scala:110)

at org.apache.spark.sql.execution.QueryExecution$$anonfun$eagerlyExecuteCommands$1.applyOrElse(QueryExecution.scala:106)

at org.apache.spark.sql.catalyst.trees.TreeNode.$anonfun$transformDownWithPruning$1(TreeNode.scala:481)

at org.apache.spark.sql.catalyst.trees.CurrentOrigin$.withOrigin(TreeNode.scala:82)

at org.apache.spark.sql.catalyst.trees.TreeNode.transformDownWithPruning(TreeNode.scala:481)

at org.apache.spark.sql.catalyst.plans.logical.LogicalPlan.org$apache$spark$sql$catalyst$plans$logical$AnalysisHelper$$super$transformDownWithPruning(LogicalPlan.scala:30)

at org.apache.spark.sql.catalyst.plans.logical.AnalysisHelper.transformDownWithPruning(AnalysisHelper.scala:267)

at org.apache.spark.sql.catalyst.plans.logical.AnalysisHelper.transformDownWithPruning$(AnalysisHelper.scala:263)

at org.apache.spark.sql.catalyst.plans.logical.LogicalPlan.transformDownWithPruning(LogicalPlan.scala:30)

at org.apache.spark.sql.catalyst.plans.logical.LogicalPlan.transformDownWithPruning(LogicalPlan.scala:30)

at org.apache.spark.sql.catalyst.trees.TreeNode.transformDown(TreeNode.scala:457)

at org.apache.spark.sql.execution.QueryExecution.eagerlyExecuteCommands(QueryExecution.scala:106)

at org.apache.spark.sql.execution.QueryExecution.commandExecuted$lzycompute(QueryExecution.scala:93)

at org.apache.spark.sql.execution.QueryExecution.commandExecuted(QueryExecution.scala:91)

at org.apache.spark.sql.execution.QueryExecution.assertCommandExecuted(QueryExecution.scala:128)

at org.apache.spark.sql.DataFrameWriter.runCommand(DataFrameWriter.scala:848)

at org.apache.spark.sql.DataFrameWriter.saveToV1Source(DataFrameWriter.scala:382)

at org.apache.spark.sql.DataFrameWriter.saveInternal(DataFrameWriter.scala:355)

at org.apache.spark.sql.DataFrameWriter.save(DataFrameWriter.scala:239)

at org.apache.spark.sql.DataFrameWriter.json(DataFrameWriter.scala:763)

at sun.reflect.NativeMethodAccessorImpl.invoke0(Native Method)

at sun.reflect.NativeMethodAccessorImpl.invoke(NativeMethodAccessorImpl.java:62)

at sun.reflect.DelegatingMethodAccessorImpl.invoke(DelegatingMethodAccessorImpl.java:43)

at java.lang.reflect.Method.invoke(Method.java:498)

at py4j.reflection.MethodInvoker.invoke(MethodInvoker.java:244)

at py4j.reflection.ReflectionEngine.invoke(ReflectionEngine.java:357)

at py4j.Gateway.invoke(Gateway.java:282)

at py4j.commands.AbstractCommand.invokeMethod(AbstractCommand.java:132)

at py4j.commands.CallCommand.execute(CallCommand.java:79)

at py4j.ClientServerConnection.waitForCommands(ClientServerConnection.java:182)

at py4j.ClientServerConnection.run(ClientServerConnection.java:106)

at java.lang.Thread.run(Thread.java:750)

Caused by: org.apache.spark.api.python.PythonException: Traceback (most recent call last):

File "/usr/local/spark/python/lib/pyspark.zip/pyspark/worker.py", line 619, in main

process()

File "/usr/local/spark/python/lib/pyspark.zip/pyspark/worker.py", line 611, in process

serializer.dump\_stream(out\_iter, outfile)

File "/usr/local/spark/python/lib/pyspark.zip/pyspark/serializers.py", line 211, in dump\_stream

self.serializer.dump\_stream(self.\_batched(iterator), stream)

File "/usr/local/spark/python/lib/pyspark.zip/pyspark/serializers.py", line 132, in dump\_stream

for obj in iterator:

File "/usr/local/spark/python/lib/pyspark.zip/pyspark/serializers.py", line 200, in \_batched

for item in iterator:

File "/usr/local/spark/python/lib/pyspark.zip/pyspark/worker.py", line 452, in mapper

result = tuple(f(\*[a[o] for o in arg\_offsets]) for (arg\_offsets, f) in udfs)

File "/usr/local/spark/python/lib/pyspark.zip/pyspark/worker.py", line 452, in <genexpr>

result = tuple(f(\*[a[o] for o in arg\_offsets]) for (arg\_offsets, f) in udfs)

File "/usr/local/spark/python/lib/pyspark.zip/pyspark/worker.py", line 87, in <lambda>

return lambda \*a: f(\*a)

File "/usr/local/spark/python/lib/pyspark.zip/pyspark/util.py", line 74, in wrapper

return f(\*args, \*\*kwargs)

File "<stdin>", line 3, in clean\_string

TypeError: expected string or bytes-like object

at org.apache.spark.api.python.BasePythonRunner$ReaderIterator.handlePythonException(PythonRunner.scala:555)

at org.apache.spark.sql.execution.python.PythonUDFRunner$$anon$2.read(PythonUDFRunner.scala:86)

at org.apache.spark.sql.execution.python.PythonUDFRunner$$anon$2.read(PythonUDFRunner.scala:68)

at org.apache.spark.api.python.BasePythonRunner$ReaderIterator.hasNext(PythonRunner.scala:508)

at org.apache.spark.InterruptibleIterator.hasNext(InterruptibleIterator.scala:37)

at scala.collection.Iterator$$anon$11.hasNext(Iterator.scala:491)

at scala.collection.Iterator$$anon$10.hasNext(Iterator.scala:460)

at scala.collection.Iterator$$anon$10.hasNext(Iterator.scala:460)

at org.apache.spark.sql.catalyst.expressions.GeneratedClass$GeneratedIteratorForCodegenStage1.processNext(Unknown Source)

at org.apache.spark.sql.execution.BufferedRowIterator.hasNext(BufferedRowIterator.java:43)

at org.apache.spark.sql.execution.WholeStageCodegenExec$$anon$1.hasNext(WholeStageCodegenExec.scala:759)

at org.apache.spark.sql.execution.columnar.DefaultCachedBatchSerializer$$anon$1.next(InMemoryRelation.scala:87)

at org.apache.spark.sql.execution.columnar.DefaultCachedBatchSerializer$$anon$1.next(InMemoryRelation.scala:79)

at scala.collection.Iterator$$anon$10.next(Iterator.scala:461)

at org.apache.spark.storage.memory.MemoryStore.putIterator(MemoryStore.scala:224)

at org.apache.spark.storage.memory.MemoryStore.putIteratorAsValues(MemoryStore.scala:302)

at org.apache.spark.storage.BlockManager.$anonfun$doPutIterator$1(BlockManager.scala:1481)

at org.apache.spark.storage.BlockManager.org$apache$spark$storage$BlockManager$$doPut(BlockManager.scala:1408)

at org.apache.spark.storage.BlockManager.doPutIterator(BlockManager.scala:1472)

at org.apache.spark.storage.BlockManager.getOrElseUpdate(BlockManager.scala:1295)

at org.apache.spark.rdd.RDD.getOrCompute(RDD.scala:384)

at org.apache.spark.rdd.RDD.iterator(RDD.scala:335)

at org.apache.spark.rdd.MapPartitionsRDD.compute(MapPartitionsRDD.scala:52)

at org.apache.spark.rdd.RDD.computeOrReadCheckpoint(RDD.scala:373)

at org.apache.spark.rdd.RDD.iterator(RDD.scala:337)

at org.apache.spark.rdd.MapPartitionsRDD.compute(MapPartitionsRDD.scala:52)

at org.apache.spark.rdd.RDD.computeOrReadCheckpoint(RDD.scala:373)

at org.apache.spark.rdd.RDD.iterator(RDD.scala:337)

at org.apache.spark.rdd.MapPartitionsRDD.compute(MapPartitionsRDD.scala:52)

at org.apache.spark.rdd.RDD.computeOrReadCheckpoint(RDD.scala:373)

at org.apache.spark.rdd.RDD.iterator(RDD.scala:337)

at org.apache.spark.scheduler.ResultTask.runTask(ResultTask.scala:90)

at org.apache.spark.scheduler.Task.run(Task.scala:131)

at org.apache.spark.executor.Executor$TaskRunner.$anonfun$run$3(Executor.scala:506)

at org.apache.spark.util.Utils$.tryWithSafeFinally(Utils.scala:1462)

at org.apache.spark.executor.Executor$TaskRunner.run(Executor.scala:509)

at java.util.concurrent.ThreadPoolExecutor.runWorker(ThreadPoolExecutor.java:1149)

at java.util.concurrent.ThreadPoolExecutor$Worker.run(ThreadPoolExecutor.java:624)

... 1 more

24/07/22 09:24:37 WARN storage.BlockManager: Putting block rdd\_531\_1 failed due to exception org.apache.spark.TaskKilledException.

24/07/22 09:24:37 WARN storage.BlockManager: Block rdd\_531\_1 could not be removed as it was not found on disk or in memory

24/07/22 09:24:37 WARN scheduler.TaskSetManager: Lost task 1.0 in stage 98.0 (TID 104) (192.168.112.129 executor driver): TaskKilled (Stage cancelled)

Traceback (most recent call last):

File "<stdin>", line 1, in <module>

File "/usr/local/spark/python/pyspark/sql/readwriter.py", line 846, in json

self.\_jwrite.json(path)

File "/usr/local/spark/python/lib/py4j-0.10.9.3-src.zip/py4j/java\_gateway.py", line 1322, in \_\_call\_\_

File "/usr/local/spark/python/pyspark/sql/utils.py", line 111, in deco

return f(\*a, \*\*kw)

File "/usr/local/spark/python/lib/py4j-0.10.9.3-src.zip/py4j/protocol.py", line 328, in get\_return\_value

py4j.protocol.Py4JJavaError: An error occurred while calling o596.json.

: org.apache.spark.SparkException: Job aborted.

at org.apache.spark.sql.errors.QueryExecutionErrors$.jobAbortedError(QueryExecutionErrors.scala:496)

at org.apache.spark.sql.execution.datasources.FileFormatWriter$.write(FileFormatWriter.scala:251)

at org.apache.spark.sql.execution.datasources.InsertIntoHadoopFsRelationCommand.run(InsertIntoHadoopFsRelationCommand.scala:186)

at org.apache.spark.sql.execution.command.DataWritingCommandExec.sideEffectResult$lzycompute(commands.scala:113)

at org.apache.spark.sql.execution.command.DataWritingCommandExec.sideEffectResult(commands.scala:111)

at org.apache.spark.sql.execution.command.DataWritingCommandExec.executeCollect(commands.scala:125)

at org.apache.spark.sql.execution.QueryExecution$$anonfun$eagerlyExecuteCommands$1.$anonfun$applyOrElse$1(QueryExecution.scala:110)

at org.apache.spark.sql.execution.SQLExecution$.$anonfun$withNewExecutionId$5(SQLExecution.scala:103)

at org.apache.spark.sql.execution.SQLExecution$.withSQLConfPropagated(SQLExecution.scala:163)

at org.apache.spark.sql.execution.SQLExecution$.$anonfun$withNewExecutionId$1(SQLExecution.scala:90)

at org.apache.spark.sql.SparkSession.withActive(SparkSession.scala:775)

at org.apache.spark.sql.execution.SQLExecution$.withNewExecutionId(SQLExecution.scala:64)

at org.apache.spark.sql.execution.QueryExecution$$anonfun$eagerlyExecuteCommands$1.applyOrElse(QueryExecution.scala:110)

at org.apache.spark.sql.execution.QueryExecution$$anonfun$eagerlyExecuteCommands$1.applyOrElse(QueryExecution.scala:106)

at org.apache.spark.sql.catalyst.trees.TreeNode.$anonfun$transformDownWithPruning$1(TreeNode.scala:481)

at org.apache.spark.sql.catalyst.trees.CurrentOrigin$.withOrigin(TreeNode.scala:82)

at org.apache.spark.sql.catalyst.trees.TreeNode.transformDownWithPruning(TreeNode.scala:481)

at org.apache.spark.sql.catalyst.plans.logical.LogicalPlan.org$apache$spark$sql$catalyst$plans$logical$AnalysisHelper$$super$transformDownWithPruning(LogicalPlan.scala:30)

at org.apache.spark.sql.catalyst.plans.logical.AnalysisHelper.transformDownWithPruning(AnalysisHelper.scala:267)

at org.apache.spark.sql.catalyst.plans.logical.AnalysisHelper.transformDownWithPruning$(AnalysisHelper.scala:263)

at org.apache.spark.sql.catalyst.plans.logical.LogicalPlan.transformDownWithPruning(LogicalPlan.scala:30)

at org.apache.spark.sql.catalyst.plans.logical.LogicalPlan.transformDownWithPruning(LogicalPlan.scala:30)

at org.apache.spark.sql.catalyst.trees.TreeNode.transformDown(TreeNode.scala:457)

at org.apache.spark.sql.execution.QueryExecution.eagerlyExecuteCommands(QueryExecution.scala:106)

at org.apache.spark.sql.execution.QueryExecution.commandExecuted$lzycompute(QueryExecution.scala:93)

at org.apache.spark.sql.execution.QueryExecution.commandExecuted(QueryExecution.scala:91)

at org.apache.spark.sql.execution.QueryExecution.assertCommandExecuted(QueryExecution.scala:128)

at org.apache.spark.sql.DataFrameWriter.runCommand(DataFrameWriter.scala:848)

at org.apache.spark.sql.DataFrameWriter.saveToV1Source(DataFrameWriter.scala:382)

at org.apache.spark.sql.DataFrameWriter.saveInternal(DataFrameWriter.scala:355)

at org.apache.spark.sql.DataFrameWriter.save(DataFrameWriter.scala:239)

at org.apache.spark.sql.DataFrameWriter.json(DataFrameWriter.scala:763)

at sun.reflect.NativeMethodAccessorImpl.invoke0(Native Method)

at sun.reflect.NativeMethodAccessorImpl.invoke(NativeMethodAccessorImpl.java:62)

at sun.reflect.DelegatingMethodAccessorImpl.invoke(DelegatingMethodAccessorImpl.java:43)

at java.lang.reflect.Method.invoke(Method.java:498)

at py4j.reflection.MethodInvoker.invoke(MethodInvoker.java:244)

at py4j.reflection.ReflectionEngine.invoke(ReflectionEngine.java:357)

at py4j.Gateway.invoke(Gateway.java:282)

at py4j.commands.AbstractCommand.invokeMethod(AbstractCommand.java:132)

at py4j.commands.CallCommand.execute(CallCommand.java:79)

at py4j.ClientServerConnection.waitForCommands(ClientServerConnection.java:182)

at py4j.ClientServerConnection.run(ClientServerConnection.java:106)

at java.lang.Thread.run(Thread.java:750)

Caused by: org.apache.spark.SparkException: Job aborted due to stage failure: Task 0 in stage 98.0 failed 1 times, most recent failure: Lost task 0.0 in stage 98.0 (TID 103) (192.168.112.129 executor driver): org.apache.spark.api.python.PythonException: Traceback (most recent call last):

File "/usr/local/spark/python/lib/pyspark.zip/pyspark/worker.py", line 619, in main

process()

File "/usr/local/spark/python/lib/pyspark.zip/pyspark/worker.py", line 611, in process

serializer.dump\_stream(out\_iter, outfile)

File "/usr/local/spark/python/lib/pyspark.zip/pyspark/serializers.py", line 211, in dump\_stream

self.serializer.dump\_stream(self.\_batched(iterator), stream)

File "/usr/local/spark/python/lib/pyspark.zip/pyspark/serializers.py", line 132, in dump\_stream

for obj in iterator:

File "/usr/local/spark/python/lib/pyspark.zip/pyspark/serializers.py", line 200, in \_batched

for item in iterator:

File "/usr/local/spark/python/lib/pyspark.zip/pyspark/worker.py", line 452, in mapper

result = tuple(f(\*[a[o] for o in arg\_offsets]) for (arg\_offsets, f) in udfs)

File "/usr/local/spark/python/lib/pyspark.zip/pyspark/worker.py", line 452, in <genexpr>

result = tuple(f(\*[a[o] for o in arg\_offsets]) for (arg\_offsets, f) in udfs)

File "/usr/local/spark/python/lib/pyspark.zip/pyspark/worker.py", line 87, in <lambda>

return lambda \*a: f(\*a)

File "/usr/local/spark/python/lib/pyspark.zip/pyspark/util.py", line 74, in wrapper

return f(\*args, \*\*kwargs)

File "<stdin>", line 3, in clean\_string

TypeError: expected string or bytes-like object

at org.apache.spark.api.python.BasePythonRunner$ReaderIterator.handlePythonException(PythonRunner.scala:555)

at org.apache.spark.sql.execution.python.PythonUDFRunner$$anon$2.read(PythonUDFRunner.scala:86)

at org.apache.spark.sql.execution.python.PythonUDFRunner$$anon$2.read(PythonUDFRunner.scala:68)

at org.apache.spark.api.python.BasePythonRunner$ReaderIterator.hasNext(PythonRunner.scala:508)

at org.apache.spark.InterruptibleIterator.hasNext(InterruptibleIterator.scala:37)

at scala.collection.Iterator$$anon$11.hasNext(Iterator.scala:491)

at scala.collection.Iterator$$anon$10.hasNext(Iterator.scala:460)

at scala.collection.Iterator$$anon$10.hasNext(Iterator.scala:460)

at org.apache.spark.sql.catalyst.expressions.GeneratedClass$GeneratedIteratorForCodegenStage1.processNext(Unknown Source)

at org.apache.spark.sql.execution.BufferedRowIterator.hasNext(BufferedRowIterator.java:43)

at org.apache.spark.sql.execution.WholeStageCodegenExec$$anon$1.hasNext(WholeStageCodegenExec.scala:759)

at org.apache.spark.sql.execution.columnar.DefaultCachedBatchSerializer$$anon$1.next(InMemoryRelation.scala:87)

at org.apache.spark.sql.execution.columnar.DefaultCachedBatchSerializer$$anon$1.next(InMemoryRelation.scala:79)

at scala.collection.Iterator$$anon$10.next(Iterator.scala:461)

at org.apache.spark.storage.memory.MemoryStore.putIterator(MemoryStore.scala:224)

at org.apache.spark.storage.memory.MemoryStore.putIteratorAsValues(MemoryStore.scala:302)

at org.apache.spark.storage.BlockManager.$anonfun$doPutIterator$1(BlockManager.scala:1481)

at org.apache.spark.storage.BlockManager.org$apache$spark$storage$BlockManager$$doPut(BlockManager.scala:1408)

at org.apache.spark.storage.BlockManager.doPutIterator(BlockManager.scala:1472)

at org.apache.spark.storage.BlockManager.getOrElseUpdate(BlockManager.scala:1295)

at org.apache.spark.rdd.RDD.getOrCompute(RDD.scala:384)

at org.apache.spark.rdd.RDD.iterator(RDD.scala:335)

at org.apache.spark.rdd.MapPartitionsRDD.compute(MapPartitionsRDD.scala:52)

at org.apache.spark.rdd.RDD.computeOrReadCheckpoint(RDD.scala:373)

at org.apache.spark.rdd.RDD.iterator(RDD.scala:337)

at org.apache.spark.rdd.MapPartitionsRDD.compute(MapPartitionsRDD.scala:52)

at org.apache.spark.rdd.RDD.computeOrReadCheckpoint(RDD.scala:373)

at org.apache.spark.rdd.RDD.iterator(RDD.scala:337)

at org.apache.spark.rdd.MapPartitionsRDD.compute(MapPartitionsRDD.scala:52)

at org.apache.spark.rdd.RDD.computeOrReadCheckpoint(RDD.scala:373)

at org.apache.spark.rdd.RDD.iterator(RDD.scala:337)

at org.apache.spark.scheduler.ResultTask.runTask(ResultTask.scala:90)

at org.apache.spark.scheduler.Task.run(Task.scala:131)

at org.apache.spark.executor.Executor$TaskRunner.$anonfun$run$3(Executor.scala:506)

at org.apache.spark.util.Utils$.tryWithSafeFinally(Utils.scala:1462)

at org.apache.spark.executor.Executor$TaskRunner.run(Executor.scala:509)

at java.util.concurrent.ThreadPoolExecutor.runWorker(ThreadPoolExecutor.java:1149)

at java.util.concurrent.ThreadPoolExecutor$Worker.run(ThreadPoolExecutor.java:624)

at java.lang.Thread.run(Thread.java:750)

Driver stacktrace:

at org.apache.spark.scheduler.DAGScheduler.failJobAndIndependentStages(DAGScheduler.scala:2454)

at org.apache.spark.scheduler.DAGScheduler.$anonfun$abortStage$2(DAGScheduler.scala:2403)

at org.apache.spark.scheduler.DAGScheduler.$anonfun$abortStage$2$adapted(DAGScheduler.scala:2402)

at scala.collection.mutable.ResizableArray.foreach(ResizableArray.scala:62)

at scala.collection.mutable.ResizableArray.foreach$(ResizableArray.scala:55)

at scala.collection.mutable.ArrayBuffer.foreach(ArrayBuffer.scala:49)

at org.apache.spark.scheduler.DAGScheduler.abortStage(DAGScheduler.scala:2402)

at org.apache.spark.scheduler.DAGScheduler.$anonfun$handleTaskSetFailed$1(DAGScheduler.scala:1160)

at org.apache.spark.scheduler.DAGScheduler.$anonfun$handleTaskSetFailed$1$adapted(DAGScheduler.scala:1160)

at scala.Option.foreach(Option.scala:407)

at org.apache.spark.scheduler.DAGScheduler.handleTaskSetFailed(DAGScheduler.scala:1160)

at org.apache.spark.scheduler.DAGSchedulerEventProcessLoop.doOnReceive(DAGScheduler.scala:2642)

at org.apache.spark.scheduler.DAGSchedulerEventProcessLoop.onReceive(DAGScheduler.scala:2584)

at org.apache.spark.scheduler.DAGSchedulerEventProcessLoop.onReceive(DAGScheduler.scala:2573)

at org.apache.spark.util.EventLoop$$anon$1.run(EventLoop.scala:49)

at org.apache.spark.scheduler.DAGScheduler.runJob(DAGScheduler.scala:938)

at org.apache.spark.SparkContext.runJob(SparkContext.scala:2214)

at org.apache.spark.sql.execution.datasources.FileFormatWriter$.write(FileFormatWriter.scala:218)

... 42 more

Caused by: org.apache.spark.api.python.PythonException: Traceback (most recent call last):

File "/usr/local/spark/python/lib/pyspark.zip/pyspark/worker.py", line 619, in main

process()

File "/usr/local/spark/python/lib/pyspark.zip/pyspark/worker.py", line 611, in process

serializer.dump\_stream(out\_iter, outfile)

File "/usr/local/spark/python/lib/pyspark.zip/pyspark/serializers.py", line 211, in dump\_stream

self.serializer.dump\_stream(self.\_batched(iterator), stream)

File "/usr/local/spark/python/lib/pyspark.zip/pyspark/serializers.py", line 132, in dump\_stream

for obj in iterator:

File "/usr/local/spark/python/lib/pyspark.zip/pyspark/serializers.py", line 200, in \_batched

for item in iterator:

File "/usr/local/spark/python/lib/pyspark.zip/pyspark/worker.py", line 452, in mapper

result = tuple(f(\*[a[o] for o in arg\_offsets]) for (arg\_offsets, f) in udfs)

File "/usr/local/spark/python/lib/pyspark.zip/pyspark/worker.py", line 452, in <genexpr>

result = tuple(f(\*[a[o] for o in arg\_offsets]) for (arg\_offsets, f) in udfs)

File "/usr/local/spark/python/lib/pyspark.zip/pyspark/worker.py", line 87, in <lambda>

return lambda \*a: f(\*a)

File "/usr/local/spark/python/lib/pyspark.zip/pyspark/util.py", line 74, in wrapper

return f(\*args, \*\*kwargs)

File "<stdin>", line 3, in clean\_string

TypeError: expected string or bytes-like object

at org.apache.spark.api.python.BasePythonRunner$ReaderIterator.handlePythonException(PythonRunner.scala:555)

at org.apache.spark.sql.execution.python.PythonUDFRunner$$anon$2.read(PythonUDFRunner.scala:86)

at org.apache.spark.sql.execution.python.PythonUDFRunner$$anon$2.read(PythonUDFRunner.scala:68)

at org.apache.spark.api.python.BasePythonRunner$ReaderIterator.hasNext(PythonRunner.scala:508)

at org.apache.spark.InterruptibleIterator.hasNext(InterruptibleIterator.scala:37)

at scala.collection.Iterator$$anon$11.hasNext(Iterator.scala:491)

at scala.collection.Iterator$$anon$10.hasNext(Iterator.scala:460)

at scala.collection.Iterator$$anon$10.hasNext(Iterator.scala:460)

at org.apache.spark.sql.catalyst.expressions.GeneratedClass$GeneratedIteratorForCodegenStage1.processNext(Unknown Source)

at org.apache.spark.sql.execution.BufferedRowIterator.hasNext(BufferedRowIterator.java:43)

at org.apache.spark.sql.execution.WholeStageCodegenExec$$anon$1.hasNext(WholeStageCodegenExec.scala:759)

at org.apache.spark.sql.execution.columnar.DefaultCachedBatchSerializer$$anon$1.next(InMemoryRelation.scala:87)

at org.apache.spark.sql.execution.columnar.DefaultCachedBatchSerializer$$anon$1.next(InMemoryRelation.scala:79)

at scala.collection.Iterator$$anon$10.next(Iterator.scala:461)

at org.apache.spark.storage.memory.MemoryStore.putIterator(MemoryStore.scala:224)

at org.apache.spark.storage.memory.MemoryStore.putIteratorAsValues(MemoryStore.scala:302)

at org.apache.spark.storage.BlockManager.$anonfun$doPutIterator$1(BlockManager.scala:1481)

at org.apache.spark.storage.BlockManager.org$apache$spark$storage$BlockManager$$doPut(BlockManager.scala:1408)

at org.apache.spark.storage.BlockManager.doPutIterator(BlockManager.scala:1472)

at org.apache.spark.storage.BlockManager.getOrElseUpdate(BlockManager.scala:1295)

at org.apache.spark.rdd.RDD.getOrCompute(RDD.scala:384)

at org.apache.spark.rdd.RDD.iterator(RDD.scala:335)

at org.apache.spark.rdd.MapPartitionsRDD.compute(MapPartitionsRDD.scala:52)

at org.apache.spark.rdd.RDD.computeOrReadCheckpoint(RDD.scala:373)

at org.apache.spark.rdd.RDD.iterator(RDD.scala:337)

at org.apache.spark.rdd.MapPartitionsRDD.compute(MapPartitionsRDD.scala:52)

at org.apache.spark.rdd.RDD.computeOrReadCheckpoint(RDD.scala:373)

at org.apache.spark.rdd.RDD.iterator(RDD.scala:337)

at org.apache.spark.rdd.MapPartitionsRDD.compute(MapPartitionsRDD.scala:52)

at org.apache.spark.rdd.RDD.computeOrReadCheckpoint(RDD.scala:373)

at org.apache.spark.rdd.RDD.iterator(RDD.scala:337)

at org.apache.spark.scheduler.ResultTask.runTask(ResultTask.scala:90)

at org.apache.spark.scheduler.Task.run(Task.scala:131)

at org.apache.spark.executor.Executor$TaskRunner.$anonfun$run$3(Executor.scala:506)

at org.apache.spark.util.Utils$.tryWithSafeFinally(Utils.scala:1462)

at org.apache.spark.executor.Executor$TaskRunner.run(Executor.scala:509)

at java.util.concurrent.ThreadPoolExecutor.runWorker(ThreadPoolExecutor.java:1149)

at java.util.concurrent.ThreadPoolExecutor$Worker.run(ThreadPoolExecutor.java:624)

... 1 more

>>> sample\_op.write.json("/user/hduser/Hackathon/JSON\_DATA", mode="overwrite")

24/07/22 09:24:58 WARN storage.BlockManager: Putting block rdd\_531\_0 failed due to exception org.apache.spark.api.python.PythonException: Traceback (most recent call last):

File "/usr/local/spark/python/lib/pyspark.zip/pyspark/worker.py", line 619, in main

process()

File "/usr/local/spark/python/lib/pyspark.zip/pyspark/worker.py", line 611, in process

serializer.dump\_stream(out\_iter, outfile)

File "/usr/local/spark/python/lib/pyspark.zip/pyspark/serializers.py", line 211, in dump\_stream

self.serializer.dump\_stream(self.\_batched(iterator), stream)

File "/usr/local/spark/python/lib/pyspark.zip/pyspark/serializers.py", line 132, in dump\_stream

for obj in iterator:

File "/usr/local/spark/python/lib/pyspark.zip/pyspark/serializers.py", line 200, in \_batched

for item in iterator:

File "/usr/local/spark/python/lib/pyspark.zip/pyspark/worker.py", line 452, in mapper

result = tuple(f(\*[a[o] for o in arg\_offsets]) for (arg\_offsets, f) in udfs)

File "/usr/local/spark/python/lib/pyspark.zip/pyspark/worker.py", line 452, in <genexpr>

result = tuple(f(\*[a[o] for o in arg\_offsets]) for (arg\_offsets, f) in udfs)

File "/usr/local/spark/python/lib/pyspark.zip/pyspark/worker.py", line 87, in <lambda>

return lambda \*a: f(\*a)

File "/usr/local/spark/python/lib/pyspark.zip/pyspark/util.py", line 74, in wrapper

return f(\*args, \*\*kwargs)

File "<stdin>", line 3, in clean\_string

TypeError: expected string or bytes-like object

.

24/07/22 09:24:58 WARN storage.BlockManager: Block rdd\_531\_0 could not be removed as it was not found on disk or in memory

24/07/22 09:24:58 ERROR executor.Executor: Exception in task 0.0 in stage 99.0 (TID 105)

org.apache.spark.api.python.PythonException: Traceback (most recent call last):

File "/usr/local/spark/python/lib/pyspark.zip/pyspark/worker.py", line 619, in main

process()

File "/usr/local/spark/python/lib/pyspark.zip/pyspark/worker.py", line 611, in process

serializer.dump\_stream(out\_iter, outfile)

File "/usr/local/spark/python/lib/pyspark.zip/pyspark/serializers.py", line 211, in dump\_stream

self.serializer.dump\_stream(self.\_batched(iterator), stream)

File "/usr/local/spark/python/lib/pyspark.zip/pyspark/serializers.py", line 132, in dump\_stream

for obj in iterator:

File "/usr/local/spark/python/lib/pyspark.zip/pyspark/serializers.py", line 200, in \_batched

for item in iterator:

File "/usr/local/spark/python/lib/pyspark.zip/pyspark/worker.py", line 452, in mapper

result = tuple(f(\*[a[o] for o in arg\_offsets]) for (arg\_offsets, f) in udfs)

File "/usr/local/spark/python/lib/pyspark.zip/pyspark/worker.py", line 452, in <genexpr>

result = tuple(f(\*[a[o] for o in arg\_offsets]) for (arg\_offsets, f) in udfs)

File "/usr/local/spark/python/lib/pyspark.zip/pyspark/worker.py", line 87, in <lambda>

return lambda \*a: f(\*a)

File "/usr/local/spark/python/lib/pyspark.zip/pyspark/util.py", line 74, in wrapper

return f(\*args, \*\*kwargs)

File "<stdin>", line 3, in clean\_string

TypeError: expected string or bytes-like object

at org.apache.spark.api.python.BasePythonRunner$ReaderIterator.handlePythonException(PythonRunner.scala:555)

at org.apache.spark.sql.execution.python.PythonUDFRunner$$anon$2.read(PythonUDFRunner.scala:86)

at org.apache.spark.sql.execution.python.PythonUDFRunner$$anon$2.read(PythonUDFRunner.scala:68)

at org.apache.spark.api.python.BasePythonRunner$ReaderIterator.hasNext(PythonRunner.scala:508)

at org.apache.spark.InterruptibleIterator.hasNext(InterruptibleIterator.scala:37)

at scala.collection.Iterator$$anon$11.hasNext(Iterator.scala:491)

at scala.collection.Iterator$$anon$10.hasNext(Iterator.scala:460)

at scala.collection.Iterator$$anon$10.hasNext(Iterator.scala:460)

at org.apache.spark.sql.catalyst.expressions.GeneratedClass$GeneratedIteratorForCodegenStage1.processNext(Unknown Source)

at org.apache.spark.sql.execution.BufferedRowIterator.hasNext(BufferedRowIterator.java:43)

at org.apache.spark.sql.execution.WholeStageCodegenExec$$anon$1.hasNext(WholeStageCodegenExec.scala:759)

at org.apache.spark.sql.execution.columnar.DefaultCachedBatchSerializer$$anon$1.next(InMemoryRelation.scala:87)

at org.apache.spark.sql.execution.columnar.DefaultCachedBatchSerializer$$anon$1.next(InMemoryRelation.scala:79)

at scala.collection.Iterator$$anon$10.next(Iterator.scala:461)

at org.apache.spark.storage.memory.MemoryStore.putIterator(MemoryStore.scala:224)

at org.apache.spark.storage.memory.MemoryStore.putIteratorAsValues(MemoryStore.scala:302)

at org.apache.spark.storage.BlockManager.$anonfun$doPutIterator$1(BlockManager.scala:1481)

at org.apache.spark.storage.BlockManager.org$apache$spark$storage$BlockManager$$doPut(BlockManager.scala:1408)

at org.apache.spark.storage.BlockManager.doPutIterator(BlockManager.scala:1472)

at org.apache.spark.storage.BlockManager.getOrElseUpdate(BlockManager.scala:1295)

at org.apache.spark.rdd.RDD.getOrCompute(RDD.scala:384)

at org.apache.spark.rdd.RDD.iterator(RDD.scala:335)

at org.apache.spark.rdd.MapPartitionsRDD.compute(MapPartitionsRDD.scala:52)

at org.apache.spark.rdd.RDD.computeOrReadCheckpoint(RDD.scala:373)

at org.apache.spark.rdd.RDD.iterator(RDD.scala:337)

at org.apache.spark.rdd.MapPartitionsRDD.compute(MapPartitionsRDD.scala:52)

at org.apache.spark.rdd.RDD.computeOrReadCheckpoint(RDD.scala:373)

at org.apache.spark.rdd.RDD.iterator(RDD.scala:337)

at org.apache.spark.rdd.MapPartitionsRDD.compute(MapPartitionsRDD.scala:52)

at org.apache.spark.rdd.RDD.computeOrReadCheckpoint(RDD.scala:373)

at org.apache.spark.rdd.RDD.iterator(RDD.scala:337)

at org.apache.spark.scheduler.ResultTask.runTask(ResultTask.scala:90)

at org.apache.spark.scheduler.Task.run(Task.scala:131)

at org.apache.spark.executor.Executor$TaskRunner.$anonfun$run$3(Executor.scala:506)

at org.apache.spark.util.Utils$.tryWithSafeFinally(Utils.scala:1462)

at org.apache.spark.executor.Executor$TaskRunner.run(Executor.scala:509)

at java.util.concurrent.ThreadPoolExecutor.runWorker(ThreadPoolExecutor.java:1149)

at java.util.concurrent.ThreadPoolExecutor$Worker.run(ThreadPoolExecutor.java:624)

at java.lang.Thread.run(Thread.java:750)

24/07/22 09:24:58 WARN scheduler.TaskSetManager: Lost task 0.0 in stage 99.0 (TID 105) (192.168.112.129 executor driver): org.apache.spark.api.python.PythonException: Traceback (most recent call last):

File "/usr/local/spark/python/lib/pyspark.zip/pyspark/worker.py", line 619, in main

process()

File "/usr/local/spark/python/lib/pyspark.zip/pyspark/worker.py", line 611, in process

serializer.dump\_stream(out\_iter, outfile)

File "/usr/local/spark/python/lib/pyspark.zip/pyspark/serializers.py", line 211, in dump\_stream

self.serializer.dump\_stream(self.\_batched(iterator), stream)

File "/usr/local/spark/python/lib/pyspark.zip/pyspark/serializers.py", line 132, in dump\_stream

for obj in iterator:

File "/usr/local/spark/python/lib/pyspark.zip/pyspark/serializers.py", line 200, in \_batched

for item in iterator:

File "/usr/local/spark/python/lib/pyspark.zip/pyspark/worker.py", line 452, in mapper

result = tuple(f(\*[a[o] for o in arg\_offsets]) for (arg\_offsets, f) in udfs)

File "/usr/local/spark/python/lib/pyspark.zip/pyspark/worker.py", line 452, in <genexpr>

result = tuple(f(\*[a[o] for o in arg\_offsets]) for (arg\_offsets, f) in udfs)

File "/usr/local/spark/python/lib/pyspark.zip/pyspark/worker.py", line 87, in <lambda>

return lambda \*a: f(\*a)

File "/usr/local/spark/python/lib/pyspark.zip/pyspark/util.py", line 74, in wrapper

return f(\*args, \*\*kwargs)

File "<stdin>", line 3, in clean\_string

TypeError: expected string or bytes-like object

at org.apache.spark.api.python.BasePythonRunner$ReaderIterator.handlePythonException(PythonRunner.scala:555)

at org.apache.spark.sql.execution.python.PythonUDFRunner$$anon$2.read(PythonUDFRunner.scala:86)

at org.apache.spark.sql.execution.python.PythonUDFRunner$$anon$2.read(PythonUDFRunner.scala:68)

at org.apache.spark.api.python.BasePythonRunner$ReaderIterator.hasNext(PythonRunner.scala:508)

at org.apache.spark.InterruptibleIterator.hasNext(InterruptibleIterator.scala:37)

at scala.collection.Iterator$$anon$11.hasNext(Iterator.scala:491)

at scala.collection.Iterator$$anon$10.hasNext(Iterator.scala:460)

at scala.collection.Iterator$$anon$10.hasNext(Iterator.scala:460)

at org.apache.spark.sql.catalyst.expressions.GeneratedClass$GeneratedIteratorForCodegenStage1.processNext(Unknown Source)

at org.apache.spark.sql.execution.BufferedRowIterator.hasNext(BufferedRowIterator.java:43)

at org.apache.spark.sql.execution.WholeStageCodegenExec$$anon$1.hasNext(WholeStageCodegenExec.scala:759)

at org.apache.spark.sql.execution.columnar.DefaultCachedBatchSerializer$$anon$1.next(InMemoryRelation.scala:87)

at org.apache.spark.sql.execution.columnar.DefaultCachedBatchSerializer$$anon$1.next(InMemoryRelation.scala:79)

at scala.collection.Iterator$$anon$10.next(Iterator.scala:461)

at org.apache.spark.storage.memory.MemoryStore.putIterator(MemoryStore.scala:224)

at org.apache.spark.storage.memory.MemoryStore.putIteratorAsValues(MemoryStore.scala:302)

at org.apache.spark.storage.BlockManager.$anonfun$doPutIterator$1(BlockManager.scala:1481)

at org.apache.spark.storage.BlockManager.org$apache$spark$storage$BlockManager$$doPut(BlockManager.scala:1408)

at org.apache.spark.storage.BlockManager.doPutIterator(BlockManager.scala:1472)

at org.apache.spark.storage.BlockManager.getOrElseUpdate(BlockManager.scala:1295)

at org.apache.spark.rdd.RDD.getOrCompute(RDD.scala:384)

at org.apache.spark.rdd.RDD.iterator(RDD.scala:335)

at org.apache.spark.rdd.MapPartitionsRDD.compute(MapPartitionsRDD.scala:52)

at org.apache.spark.rdd.RDD.computeOrReadCheckpoint(RDD.scala:373)

at org.apache.spark.rdd.RDD.iterator(RDD.scala:337)

at org.apache.spark.rdd.MapPartitionsRDD.compute(MapPartitionsRDD.scala:52)

at org.apache.spark.rdd.RDD.computeOrReadCheckpoint(RDD.scala:373)

at org.apache.spark.rdd.RDD.iterator(RDD.scala:337)

at org.apache.spark.rdd.MapPartitionsRDD.compute(MapPartitionsRDD.scala:52)

at org.apache.spark.rdd.RDD.computeOrReadCheckpoint(RDD.scala:373)

at org.apache.spark.rdd.RDD.iterator(RDD.scala:337)

at org.apache.spark.scheduler.ResultTask.runTask(ResultTask.scala:90)

at org.apache.spark.scheduler.Task.run(Task.scala:131)

at org.apache.spark.executor.Executor$TaskRunner.$anonfun$run$3(Executor.scala:506)

at org.apache.spark.util.Utils$.tryWithSafeFinally(Utils.scala:1462)

at org.apache.spark.executor.Executor$TaskRunner.run(Executor.scala:509)

at java.util.concurrent.ThreadPoolExecutor.runWorker(ThreadPoolExecutor.java:1149)

at java.util.concurrent.ThreadPoolExecutor$Worker.run(ThreadPoolExecutor.java:624)

at java.lang.Thread.run(Thread.java:750)

24/07/22 09:24:58 ERROR scheduler.TaskSetManager: Task 0 in stage 99.0 failed 1 times; aborting job

24/07/22 09:24:58 WARN storage.BlockManager: Putting block rdd\_531\_1 failed due to exception org.apache.spark.TaskKilledException.

24/07/22 09:24:58 WARN storage.BlockManager: Block rdd\_531\_1 could not be removed as it was not found on disk or in memory

24/07/22 09:24:58 ERROR datasources.FileFormatWriter: Aborting job c72afc06-3542-4045-b670-b6f8af5afada.

org.apache.spark.SparkException: Job aborted due to stage failure: Task 0 in stage 99.0 failed 1 times, most recent failure: Lost task 0.0 in stage 99.0 (TID 105) (192.168.112.129 executor driver): org.apache.spark.api.python.PythonException: Traceback (most recent call last):

File "/usr/local/spark/python/lib/pyspark.zip/pyspark/worker.py", line 619, in main

process()

File "/usr/local/spark/python/lib/pyspark.zip/pyspark/worker.py", line 611, in process

serializer.dump\_stream(out\_iter, outfile)

File "/usr/local/spark/python/lib/pyspark.zip/pyspark/serializers.py", line 211, in dump\_stream

self.serializer.dump\_stream(self.\_batched(iterator), stream)

File "/usr/local/spark/python/lib/pyspark.zip/pyspark/serializers.py", line 132, in dump\_stream

for obj in iterator:

File "/usr/local/spark/python/lib/pyspark.zip/pyspark/serializers.py", line 200, in \_batched

for item in iterator:

File "/usr/local/spark/python/lib/pyspark.zip/pyspark/worker.py", line 452, in mapper

result = tuple(f(\*[a[o] for o in arg\_offsets]) for (arg\_offsets, f) in udfs)

File "/usr/local/spark/python/lib/pyspark.zip/pyspark/worker.py", line 452, in <genexpr>

result = tuple(f(\*[a[o] for o in arg\_offsets]) for (arg\_offsets, f) in udfs)

File "/usr/local/spark/python/lib/pyspark.zip/pyspark/worker.py", line 87, in <lambda>

return lambda \*a: f(\*a)

File "/usr/local/spark/python/lib/pyspark.zip/pyspark/util.py", line 74, in wrapper

return f(\*args, \*\*kwargs)

File "<stdin>", line 3, in clean\_string

TypeError: expected string or bytes-like object

at org.apache.spark.api.python.BasePythonRunner$ReaderIterator.handlePythonException(PythonRunner.scala:555)

at org.apache.spark.sql.execution.python.PythonUDFRunner$$anon$2.read(PythonUDFRunner.scala:86)

at org.apache.spark.sql.execution.python.PythonUDFRunner$$anon$2.read(PythonUDFRunner.scala:68)

at org.apache.spark.api.python.BasePythonRunner$ReaderIterator.hasNext(PythonRunner.scala:508)

at org.apache.spark.InterruptibleIterator.hasNext(InterruptibleIterator.scala:37)

at scala.collection.Iterator$$anon$11.hasNext(Iterator.scala:491)

at scala.collection.Iterator$$anon$10.hasNext(Iterator.scala:460)

at scala.collection.Iterator$$anon$10.hasNext(Iterator.scala:460)

at org.apache.spark.sql.catalyst.expressions.GeneratedClass$GeneratedIteratorForCodegenStage1.processNext(Unknown Source)

at org.apache.spark.sql.execution.BufferedRowIterator.hasNext(BufferedRowIterator.java:43)

at org.apache.spark.sql.execution.WholeStageCodegenExec$$anon$1.hasNext(WholeStageCodegenExec.scala:759)

at org.apache.spark.sql.execution.columnar.DefaultCachedBatchSerializer$$anon$1.next(InMemoryRelation.scala:87)

at org.apache.spark.sql.execution.columnar.DefaultCachedBatchSerializer$$anon$1.next(InMemoryRelation.scala:79)

at scala.collection.Iterator$$anon$10.next(Iterator.scala:461)

at org.apache.spark.storage.memory.MemoryStore.putIterator(MemoryStore.scala:224)

at org.apache.spark.storage.memory.MemoryStore.putIteratorAsValues(MemoryStore.scala:302)

at org.apache.spark.storage.BlockManager.$anonfun$doPutIterator$1(BlockManager.scala:1481)

at org.apache.spark.storage.BlockManager.org$apache$spark$storage$BlockManager$$doPut(BlockManager.scala:1408)

at org.apache.spark.storage.BlockManager.doPutIterator(BlockManager.scala:1472)

at org.apache.spark.storage.BlockManager.getOrElseUpdate(BlockManager.scala:1295)

at org.apache.spark.rdd.RDD.getOrCompute(RDD.scala:384)

at org.apache.spark.rdd.RDD.iterator(RDD.scala:335)

at org.apache.spark.rdd.MapPartitionsRDD.compute(MapPartitionsRDD.scala:52)

at org.apache.spark.rdd.RDD.computeOrReadCheckpoint(RDD.scala:373)

at org.apache.spark.rdd.RDD.iterator(RDD.scala:337)

at org.apache.spark.rdd.MapPartitionsRDD.compute(MapPartitionsRDD.scala:52)

at org.apache.spark.rdd.RDD.computeOrReadCheckpoint(RDD.scala:373)

at org.apache.spark.rdd.RDD.iterator(RDD.scala:337)

at org.apache.spark.rdd.MapPartitionsRDD.compute(MapPartitionsRDD.scala:52)

at org.apache.spark.rdd.RDD.computeOrReadCheckpoint(RDD.scala:373)

at org.apache.spark.rdd.RDD.iterator(RDD.scala:337)

at org.apache.spark.scheduler.ResultTask.runTask(ResultTask.scala:90)

at org.apache.spark.scheduler.Task.run(Task.scala:131)

at org.apache.spark.executor.Executor$TaskRunner.$anonfun$run$3(Executor.scala:506)

at org.apache.spark.util.Utils$.tryWithSafeFinally(Utils.scala:1462)

at org.apache.spark.executor.Executor$TaskRunner.run(Executor.scala:509)

at java.util.concurrent.ThreadPoolExecutor.runWorker(ThreadPoolExecutor.java:1149)

at java.util.concurrent.ThreadPoolExecutor$Worker.run(ThreadPoolExecutor.java:624)

at java.lang.Thread.run(Thread.java:750)

Driver stacktrace:

at org.apache.spark.scheduler.DAGScheduler.failJobAndIndependentStages(DAGScheduler.scala:2454)

at org.apache.spark.scheduler.DAGScheduler.$anonfun$abortStage$2(DAGScheduler.scala:2403)

at org.apache.spark.scheduler.DAGScheduler.$anonfun$abortStage$2$adapted(DAGScheduler.scala:2402)

at scala.collection.mutable.ResizableArray.foreach(ResizableArray.scala:62)

at scala.collection.mutable.ResizableArray.foreach$(ResizableArray.scala:55)

at scala.collection.mutable.ArrayBuffer.foreach(ArrayBuffer.scala:49)

at org.apache.spark.scheduler.DAGScheduler.abortStage(DAGScheduler.scala:2402)

at org.apache.spark.scheduler.DAGScheduler.$anonfun$handleTaskSetFailed$1(DAGScheduler.scala:1160)

at org.apache.spark.scheduler.DAGScheduler.$anonfun$handleTaskSetFailed$1$adapted(DAGScheduler.scala:1160)

at scala.Option.foreach(Option.scala:407)

at org.apache.spark.scheduler.DAGScheduler.handleTaskSetFailed(DAGScheduler.scala:1160)

at org.apache.spark.scheduler.DAGSchedulerEventProcessLoop.doOnReceive(DAGScheduler.scala:2642)

at org.apache.spark.scheduler.DAGSchedulerEventProcessLoop.onReceive(DAGScheduler.scala:2584)

at org.apache.spark.scheduler.DAGSchedulerEventProcessLoop.onReceive(DAGScheduler.scala:2573)

at org.apache.spark.util.EventLoop$$anon$1.run(EventLoop.scala:49)

at org.apache.spark.scheduler.DAGScheduler.runJob(DAGScheduler.scala:938)

at org.apache.spark.SparkContext.runJob(SparkContext.scala:2214)

at org.apache.spark.sql.execution.datasources.FileFormatWriter$.write(FileFormatWriter.scala:218)

at org.apache.spark.sql.execution.datasources.InsertIntoHadoopFsRelationCommand.run(InsertIntoHadoopFsRelationCommand.scala:186)

at org.apache.spark.sql.execution.command.DataWritingCommandExec.sideEffectResult$lzycompute(commands.scala:113)

at org.apache.spark.sql.execution.command.DataWritingCommandExec.sideEffectResult(commands.scala:111)

at org.apache.spark.sql.execution.command.DataWritingCommandExec.executeCollect(commands.scala:125)

at org.apache.spark.sql.execution.QueryExecution$$anonfun$eagerlyExecuteCommands$1.$anonfun$applyOrElse$1(QueryExecution.scala:110)

at org.apache.spark.sql.execution.SQLExecution$.$anonfun$withNewExecutionId$5(SQLExecution.scala:103)

at org.apache.spark.sql.execution.SQLExecution$.withSQLConfPropagated(SQLExecution.scala:163)

at org.apache.spark.sql.execution.SQLExecution$.$anonfun$withNewExecutionId$1(SQLExecution.scala:90)

at org.apache.spark.sql.SparkSession.withActive(SparkSession.scala:775)

at org.apache.spark.sql.execution.SQLExecution$.withNewExecutionId(SQLExecution.scala:64)

at org.apache.spark.sql.execution.QueryExecution$$anonfun$eagerlyExecuteCommands$1.applyOrElse(QueryExecution.scala:110)

at org.apache.spark.sql.execution.QueryExecution$$anonfun$eagerlyExecuteCommands$1.applyOrElse(QueryExecution.scala:106)

at org.apache.spark.sql.catalyst.trees.TreeNode.$anonfun$transformDownWithPruning$1(TreeNode.scala:481)

at org.apache.spark.sql.catalyst.trees.CurrentOrigin$.withOrigin(TreeNode.scala:82)

at org.apache.spark.sql.catalyst.trees.TreeNode.transformDownWithPruning(TreeNode.scala:481)

at org.apache.spark.sql.catalyst.plans.logical.LogicalPlan.org$apache$spark$sql$catalyst$plans$logical$AnalysisHelper$$super$transformDownWithPruning(LogicalPlan.scala:30)

at org.apache.spark.sql.catalyst.plans.logical.AnalysisHelper.transformDownWithPruning(AnalysisHelper.scala:267)

at org.apache.spark.sql.catalyst.plans.logical.AnalysisHelper.transformDownWithPruning$(AnalysisHelper.scala:263)

at org.apache.spark.sql.catalyst.plans.logical.LogicalPlan.transformDownWithPruning(LogicalPlan.scala:30)

at org.apache.spark.sql.catalyst.plans.logical.LogicalPlan.transformDownWithPruning(LogicalPlan.scala:30)

at org.apache.spark.sql.catalyst.trees.TreeNode.transformDown(TreeNode.scala:457)

at org.apache.spark.sql.execution.QueryExecution.eagerlyExecuteCommands(QueryExecution.scala:106)

at org.apache.spark.sql.execution.QueryExecution.commandExecuted$lzycompute(QueryExecution.scala:93)

at org.apache.spark.sql.execution.QueryExecution.commandExecuted(QueryExecution.scala:91)

at org.apache.spark.sql.execution.QueryExecution.assertCommandExecuted(QueryExecution.scala:128)

at org.apache.spark.sql.DataFrameWriter.runCommand(DataFrameWriter.scala:848)

at org.apache.spark.sql.DataFrameWriter.saveToV1Source(DataFrameWriter.scala:382)

at org.apache.spark.sql.DataFrameWriter.saveInternal(DataFrameWriter.scala:355)

at org.apache.spark.sql.DataFrameWriter.save(DataFrameWriter.scala:239)

at org.apache.spark.sql.DataFrameWriter.json(DataFrameWriter.scala:763)

at sun.reflect.NativeMethodAccessorImpl.invoke0(Native Method)

at sun.reflect.NativeMethodAccessorImpl.invoke(NativeMethodAccessorImpl.java:62)

at sun.reflect.DelegatingMethodAccessorImpl.invoke(DelegatingMethodAccessorImpl.java:43)

at java.lang.reflect.Method.invoke(Method.java:498)

at py4j.reflection.MethodInvoker.invoke(MethodInvoker.java:244)

at py4j.reflection.ReflectionEngine.invoke(ReflectionEngine.java:357)

at py4j.Gateway.invoke(Gateway.java:282)

at py4j.commands.AbstractCommand.invokeMethod(AbstractCommand.java:132)

at py4j.commands.CallCommand.execute(CallCommand.java:79)

at py4j.ClientServerConnection.waitForCommands(ClientServerConnection.java:182)

at py4j.ClientServerConnection.run(ClientServerConnection.java:106)

at java.lang.Thread.run(Thread.java:750)

Caused by: org.apache.spark.api.python.PythonException: Traceback (most recent call last):

File "/usr/local/spark/python/lib/pyspark.zip/pyspark/worker.py", line 619, in main

process()

File "/usr/local/spark/python/lib/pyspark.zip/pyspark/worker.py", line 611, in process

serializer.dump\_stream(out\_iter, outfile)

File "/usr/local/spark/python/lib/pyspark.zip/pyspark/serializers.py", line 211, in dump\_stream

self.serializer.dump\_stream(self.\_batched(iterator), stream)

File "/usr/local/spark/python/lib/pyspark.zip/pyspark/serializers.py", line 132, in dump\_stream

for obj in iterator:

File "/usr/local/spark/python/lib/pyspark.zip/pyspark/serializers.py", line 200, in \_batched

for item in iterator:

File "/usr/local/spark/python/lib/pyspark.zip/pyspark/worker.py", line 452, in mapper

result = tuple(f(\*[a[o] for o in arg\_offsets]) for (arg\_offsets, f) in udfs)

File "/usr/local/spark/python/lib/pyspark.zip/pyspark/worker.py", line 452, in <genexpr>

result = tuple(f(\*[a[o] for o in arg\_offsets]) for (arg\_offsets, f) in udfs)

File "/usr/local/spark/python/lib/pyspark.zip/pyspark/worker.py", line 87, in <lambda>

return lambda \*a: f(\*a)

File "/usr/local/spark/python/lib/pyspark.zip/pyspark/util.py", line 74, in wrapper

return f(\*args, \*\*kwargs)

File "<stdin>", line 3, in clean\_string

TypeError: expected string or bytes-like object

at org.apache.spark.api.python.BasePythonRunner$ReaderIterator.handlePythonException(PythonRunner.scala:555)

at org.apache.spark.sql.execution.python.PythonUDFRunner$$anon$2.read(PythonUDFRunner.scala:86)

at org.apache.spark.sql.execution.python.PythonUDFRunner$$anon$2.read(PythonUDFRunner.scala:68)

at org.apache.spark.api.python.BasePythonRunner$ReaderIterator.hasNext(PythonRunner.scala:508)

at org.apache.spark.InterruptibleIterator.hasNext(InterruptibleIterator.scala:37)

at scala.collection.Iterator$$anon$11.hasNext(Iterator.scala:491)

at scala.collection.Iterator$$anon$10.hasNext(Iterator.scala:460)

at scala.collection.Iterator$$anon$10.hasNext(Iterator.scala:460)

at org.apache.spark.sql.catalyst.expressions.GeneratedClass$GeneratedIteratorForCodegenStage1.processNext(Unknown Source)

at org.apache.spark.sql.execution.BufferedRowIterator.hasNext(BufferedRowIterator.java:43)

at org.apache.spark.sql.execution.WholeStageCodegenExec$$anon$1.hasNext(WholeStageCodegenExec.scala:759)

at org.apache.spark.sql.execution.columnar.DefaultCachedBatchSerializer$$anon$1.next(InMemoryRelation.scala:87)

at org.apache.spark.sql.execution.columnar.DefaultCachedBatchSerializer$$anon$1.next(InMemoryRelation.scala:79)

at scala.collection.Iterator$$anon$10.next(Iterator.scala:461)

at org.apache.spark.storage.memory.MemoryStore.putIterator(MemoryStore.scala:224)

at org.apache.spark.storage.memory.MemoryStore.putIteratorAsValues(MemoryStore.scala:302)

at org.apache.spark.storage.BlockManager.$anonfun$doPutIterator$1(BlockManager.scala:1481)

at org.apache.spark.storage.BlockManager.org$apache$spark$storage$BlockManager$$doPut(BlockManager.scala:1408)

at org.apache.spark.storage.BlockManager.doPutIterator(BlockManager.scala:1472)

at org.apache.spark.storage.BlockManager.getOrElseUpdate(BlockManager.scala:1295)

at org.apache.spark.rdd.RDD.getOrCompute(RDD.scala:384)

at org.apache.spark.rdd.RDD.iterator(RDD.scala:335)

at org.apache.spark.rdd.MapPartitionsRDD.compute(MapPartitionsRDD.scala:52)

at org.apache.spark.rdd.RDD.computeOrReadCheckpoint(RDD.scala:373)

at org.apache.spark.rdd.RDD.iterator(RDD.scala:337)

at org.apache.spark.rdd.MapPartitionsRDD.compute(MapPartitionsRDD.scala:52)

at org.apache.spark.rdd.RDD.computeOrReadCheckpoint(RDD.scala:373)

at org.apache.spark.rdd.RDD.iterator(RDD.scala:337)

at org.apache.spark.rdd.MapPartitionsRDD.compute(MapPartitionsRDD.scala:52)

at org.apache.spark.rdd.RDD.computeOrReadCheckpoint(RDD.scala:373)

at org.apache.spark.rdd.RDD.iterator(RDD.scala:337)

at org.apache.spark.scheduler.ResultTask.runTask(ResultTask.scala:90)

at org.apache.spark.scheduler.Task.run(Task.scala:131)

at org.apache.spark.executor.Executor$TaskRunner.$anonfun$run$3(Executor.scala:506)

at org.apache.spark.util.Utils$.tryWithSafeFinally(Utils.scala:1462)

at org.apache.spark.executor.Executor$TaskRunner.run(Executor.scala:509)

at java.util.concurrent.ThreadPoolExecutor.runWorker(ThreadPoolExecutor.java:1149)

at java.util.concurrent.ThreadPoolExecutor$Worker.run(ThreadPoolExecutor.java:624)

... 1 more

24/07/22 09:24:58 WARN scheduler.TaskSetManager: Lost task 1.0 in stage 99.0 (TID 106) (192.168.112.129 executor driver): TaskKilled (Stage cancelled)

Traceback (most recent call last):

File "<stdin>", line 1, in <module>

File "/usr/local/spark/python/pyspark/sql/readwriter.py", line 846, in json

self.\_jwrite.json(path)

File "/usr/local/spark/python/lib/py4j-0.10.9.3-src.zip/py4j/java\_gateway.py", line 1322, in \_\_call\_\_

File "/usr/local/spark/python/pyspark/sql/utils.py", line 111, in deco

return f(\*a, \*\*kw)

File "/usr/local/spark/python/lib/py4j-0.10.9.3-src.zip/py4j/protocol.py", line 328, in get\_return\_value

py4j.protocol.Py4JJavaError: An error occurred while calling o604.json.

: org.apache.spark.SparkException: Job aborted.

at org.apache.spark.sql.errors.QueryExecutionErrors$.jobAbortedError(QueryExecutionErrors.scala:496)

at org.apache.spark.sql.execution.datasources.FileFormatWriter$.write(FileFormatWriter.scala:251)

at org.apache.spark.sql.execution.datasources.InsertIntoHadoopFsRelationCommand.run(InsertIntoHadoopFsRelationCommand.scala:186)

at org.apache.spark.sql.execution.command.DataWritingCommandExec.sideEffectResult$lzycompute(commands.scala:113)

at org.apache.spark.sql.execution.command.DataWritingCommandExec.sideEffectResult(commands.scala:111)

at org.apache.spark.sql.execution.command.DataWritingCommandExec.executeCollect(commands.scala:125)

at org.apache.spark.sql.execution.QueryExecution$$anonfun$eagerlyExecuteCommands$1.$anonfun$applyOrElse$1(QueryExecution.scala:110)

at org.apache.spark.sql.execution.SQLExecution$.$anonfun$withNewExecutionId$5(SQLExecution.scala:103)

at org.apache.spark.sql.execution.SQLExecution$.withSQLConfPropagated(SQLExecution.scala:163)

at org.apache.spark.sql.execution.SQLExecution$.$anonfun$withNewExecutionId$1(SQLExecution.scala:90)

at org.apache.spark.sql.SparkSession.withActive(SparkSession.scala:775)

at org.apache.spark.sql.execution.SQLExecution$.withNewExecutionId(SQLExecution.scala:64)

at org.apache.spark.sql.execution.QueryExecution$$anonfun$eagerlyExecuteCommands$1.applyOrElse(QueryExecution.scala:110)

at org.apache.spark.sql.execution.QueryExecution$$anonfun$eagerlyExecuteCommands$1.applyOrElse(QueryExecution.scala:106)

at org.apache.spark.sql.catalyst.trees.TreeNode.$anonfun$transformDownWithPruning$1(TreeNode.scala:481)

at org.apache.spark.sql.catalyst.trees.CurrentOrigin$.withOrigin(TreeNode.scala:82)

at org.apache.spark.sql.catalyst.trees.TreeNode.transformDownWithPruning(TreeNode.scala:481)

at org.apache.spark.sql.catalyst.plans.logical.LogicalPlan.org$apache$spark$sql$catalyst$plans$logical$AnalysisHelper$$super$transformDownWithPruning(LogicalPlan.scala:30)

at org.apache.spark.sql.catalyst.plans.logical.AnalysisHelper.transformDownWithPruning(AnalysisHelper.scala:267)

at org.apache.spark.sql.catalyst.plans.logical.AnalysisHelper.transformDownWithPruning$(AnalysisHelper.scala:263)

at org.apache.spark.sql.catalyst.plans.logical.LogicalPlan.transformDownWithPruning(LogicalPlan.scala:30)

at org.apache.spark.sql.catalyst.plans.logical.LogicalPlan.transformDownWithPruning(LogicalPlan.scala:30)

at org.apache.spark.sql.catalyst.trees.TreeNode.transformDown(TreeNode.scala:457)

at org.apache.spark.sql.execution.QueryExecution.eagerlyExecuteCommands(QueryExecution.scala:106)

at org.apache.spark.sql.execution.QueryExecution.commandExecuted$lzycompute(QueryExecution.scala:93)

at org.apache.spark.sql.execution.QueryExecution.commandExecuted(QueryExecution.scala:91)

at org.apache.spark.sql.execution.QueryExecution.assertCommandExecuted(QueryExecution.scala:128)

at org.apache.spark.sql.DataFrameWriter.runCommand(DataFrameWriter.scala:848)

at org.apache.spark.sql.DataFrameWriter.saveToV1Source(DataFrameWriter.scala:382)

at org.apache.spark.sql.DataFrameWriter.saveInternal(DataFrameWriter.scala:355)

at org.apache.spark.sql.DataFrameWriter.save(DataFrameWriter.scala:239)

at org.apache.spark.sql.DataFrameWriter.json(DataFrameWriter.scala:763)

at sun.reflect.NativeMethodAccessorImpl.invoke0(Native Method)

at sun.reflect.NativeMethodAccessorImpl.invoke(NativeMethodAccessorImpl.java:62)

at sun.reflect.DelegatingMethodAccessorImpl.invoke(DelegatingMethodAccessorImpl.java:43)

at java.lang.reflect.Method.invoke(Method.java:498)

at py4j.reflection.MethodInvoker.invoke(MethodInvoker.java:244)

at py4j.reflection.ReflectionEngine.invoke(ReflectionEngine.java:357)

at py4j.Gateway.invoke(Gateway.java:282)

at py4j.commands.AbstractCommand.invokeMethod(AbstractCommand.java:132)

at py4j.commands.CallCommand.execute(CallCommand.java:79)

at py4j.ClientServerConnection.waitForCommands(ClientServerConnection.java:182)

at py4j.ClientServerConnection.run(ClientServerConnection.java:106)

at java.lang.Thread.run(Thread.java:750)

Caused by: org.apache.spark.SparkException: Job aborted due to stage failure: Task 0 in stage 99.0 failed 1 times, most recent failure: Lost task 0.0 in stage 99.0 (TID 105) (192.168.112.129 executor driver): org.apache.spark.api.python.PythonException: Traceback (most recent call last):

File "/usr/local/spark/python/lib/pyspark.zip/pyspark/worker.py", line 619, in main

process()

File "/usr/local/spark/python/lib/pyspark.zip/pyspark/worker.py", line 611, in process

serializer.dump\_stream(out\_iter, outfile)

File "/usr/local/spark/python/lib/pyspark.zip/pyspark/serializers.py", line 211, in dump\_stream

self.serializer.dump\_stream(self.\_batched(iterator), stream)

File "/usr/local/spark/python/lib/pyspark.zip/pyspark/serializers.py", line 132, in dump\_stream

for obj in iterator:

File "/usr/local/spark/python/lib/pyspark.zip/pyspark/serializers.py", line 200, in \_batched

for item in iterator:

File "/usr/local/spark/python/lib/pyspark.zip/pyspark/worker.py", line 452, in mapper

result = tuple(f(\*[a[o] for o in arg\_offsets]) for (arg\_offsets, f) in udfs)

File "/usr/local/spark/python/lib/pyspark.zip/pyspark/worker.py", line 452, in <genexpr>

result = tuple(f(\*[a[o] for o in arg\_offsets]) for (arg\_offsets, f) in udfs)

File "/usr/local/spark/python/lib/pyspark.zip/pyspark/worker.py", line 87, in <lambda>

return lambda \*a: f(\*a)

File "/usr/local/spark/python/lib/pyspark.zip/pyspark/util.py", line 74, in wrapper

return f(\*args, \*\*kwargs)

File "<stdin>", line 3, in clean\_string

TypeError: expected string or bytes-like object

at org.apache.spark.api.python.BasePythonRunner$ReaderIterator.handlePythonException(PythonRunner.scala:555)

at org.apache.spark.sql.execution.python.PythonUDFRunner$$anon$2.read(PythonUDFRunner.scala:86)

at org.apache.spark.sql.execution.python.PythonUDFRunner$$anon$2.read(PythonUDFRunner.scala:68)

at org.apache.spark.api.python.BasePythonRunner$ReaderIterator.hasNext(PythonRunner.scala:508)

at org.apache.spark.InterruptibleIterator.hasNext(InterruptibleIterator.scala:37)

at scala.collection.Iterator$$anon$11.hasNext(Iterator.scala:491)

at scala.collection.Iterator$$anon$10.hasNext(Iterator.scala:460)

at scala.collection.Iterator$$anon$10.hasNext(Iterator.scala:460)

at org.apache.spark.sql.catalyst.expressions.GeneratedClass$GeneratedIteratorForCodegenStage1.processNext(Unknown Source)

at org.apache.spark.sql.execution.BufferedRowIterator.hasNext(BufferedRowIterator.java:43)

at org.apache.spark.sql.execution.WholeStageCodegenExec$$anon$1.hasNext(WholeStageCodegenExec.scala:759)

at org.apache.spark.sql.execution.columnar.DefaultCachedBatchSerializer$$anon$1.next(InMemoryRelation.scala:87)

at org.apache.spark.sql.execution.columnar.DefaultCachedBatchSerializer$$anon$1.next(InMemoryRelation.scala:79)

at scala.collection.Iterator$$anon$10.next(Iterator.scala:461)

at org.apache.spark.storage.memory.MemoryStore.putIterator(MemoryStore.scala:224)

at org.apache.spark.storage.memory.MemoryStore.putIteratorAsValues(MemoryStore.scala:302)

at org.apache.spark.storage.BlockManager.$anonfun$doPutIterator$1(BlockManager.scala:1481)

at org.apache.spark.storage.BlockManager.org$apache$spark$storage$BlockManager$$doPut(BlockManager.scala:1408)

at org.apache.spark.storage.BlockManager.doPutIterator(BlockManager.scala:1472)

at org.apache.spark.storage.BlockManager.getOrElseUpdate(BlockManager.scala:1295)

at org.apache.spark.rdd.RDD.getOrCompute(RDD.scala:384)

at org.apache.spark.rdd.RDD.iterator(RDD.scala:335)

at org.apache.spark.rdd.MapPartitionsRDD.compute(MapPartitionsRDD.scala:52)

at org.apache.spark.rdd.RDD.computeOrReadCheckpoint(RDD.scala:373)

at org.apache.spark.rdd.RDD.iterator(RDD.scala:337)

at org.apache.spark.rdd.MapPartitionsRDD.compute(MapPartitionsRDD.scala:52)

at org.apache.spark.rdd.RDD.computeOrReadCheckpoint(RDD.scala:373)

at org.apache.spark.rdd.RDD.iterator(RDD.scala:337)

at org.apache.spark.rdd.MapPartitionsRDD.compute(MapPartitionsRDD.scala:52)

at org.apache.spark.rdd.RDD.computeOrReadCheckpoint(RDD.scala:373)

at org.apache.spark.rdd.RDD.iterator(RDD.scala:337)

at org.apache.spark.scheduler.ResultTask.runTask(ResultTask.scala:90)

at org.apache.spark.scheduler.Task.run(Task.scala:131)

at org.apache.spark.executor.Executor$TaskRunner.$anonfun$run$3(Executor.scala:506)

at org.apache.spark.util.Utils$.tryWithSafeFinally(Utils.scala:1462)

at org.apache.spark.executor.Executor$TaskRunner.run(Executor.scala:509)

at java.util.concurrent.ThreadPoolExecutor.runWorker(ThreadPoolExecutor.java:1149)

at java.util.concurrent.ThreadPoolExecutor$Worker.run(ThreadPoolExecutor.java:624)

at java.lang.Thread.run(Thread.java:750)

Driver stacktrace:

at org.apache.spark.scheduler.DAGScheduler.failJobAndIndependentStages(DAGScheduler.scala:2454)

at org.apache.spark.scheduler.DAGScheduler.$anonfun$abortStage$2(DAGScheduler.scala:2403)

at org.apache.spark.scheduler.DAGScheduler.$anonfun$abortStage$2$adapted(DAGScheduler.scala:2402)

at scala.collection.mutable.ResizableArray.foreach(ResizableArray.scala:62)

at scala.collection.mutable.ResizableArray.foreach$(ResizableArray.scala:55)

at scala.collection.mutable.ArrayBuffer.foreach(ArrayBuffer.scala:49)

at org.apache.spark.scheduler.DAGScheduler.abortStage(DAGScheduler.scala:2402)

at org.apache.spark.scheduler.DAGScheduler.$anonfun$handleTaskSetFailed$1(DAGScheduler.scala:1160)

at org.apache.spark.scheduler.DAGScheduler.$anonfun$handleTaskSetFailed$1$adapted(DAGScheduler.scala:1160)

at scala.Option.foreach(Option.scala:407)

at org.apache.spark.scheduler.DAGScheduler.handleTaskSetFailed(DAGScheduler.scala:1160)

at org.apache.spark.scheduler.DAGSchedulerEventProcessLoop.doOnReceive(DAGScheduler.scala:2642)

at org.apache.spark.scheduler.DAGSchedulerEventProcessLoop.onReceive(DAGScheduler.scala:2584)

at org.apache.spark.scheduler.DAGSchedulerEventProcessLoop.onReceive(DAGScheduler.scala:2573)

at org.apache.spark.util.EventLoop$$anon$1.run(EventLoop.scala:49)

at org.apache.spark.scheduler.DAGScheduler.runJob(DAGScheduler.scala:938)

at org.apache.spark.SparkContext.runJob(SparkContext.scala:2214)

at org.apache.spark.sql.execution.datasources.FileFormatWriter$.write(FileFormatWriter.scala:218)

... 42 more

Caused by: org.apache.spark.api.python.PythonException: Traceback (most recent call last):

File "/usr/local/spark/python/lib/pyspark.zip/pyspark/worker.py", line 619, in main

process()

File "/usr/local/spark/python/lib/pyspark.zip/pyspark/worker.py", line 611, in process

serializer.dump\_stream(out\_iter, outfile)

File "/usr/local/spark/python/lib/pyspark.zip/pyspark/serializers.py", line 211, in dump\_stream

self.serializer.dump\_stream(self.\_batched(iterator), stream)

File "/usr/local/spark/python/lib/pyspark.zip/pyspark/serializers.py", line 132, in dump\_stream

for obj in iterator:

File "/usr/local/spark/python/lib/pyspark.zip/pyspark/serializers.py", line 200, in \_batched

for item in iterator:

File "/usr/local/spark/python/lib/pyspark.zip/pyspark/worker.py", line 452, in mapper

result = tuple(f(\*[a[o] for o in arg\_offsets]) for (arg\_offsets, f) in udfs)

File "/usr/local/spark/python/lib/pyspark.zip/pyspark/worker.py", line 452, in <genexpr>

result = tuple(f(\*[a[o] for o in arg\_offsets]) for (arg\_offsets, f) in udfs)

File "/usr/local/spark/python/lib/pyspark.zip/pyspark/worker.py", line 87, in <lambda>

return lambda \*a: f(\*a)

File "/usr/local/spark/python/lib/pyspark.zip/pyspark/util.py", line 74, in wrapper

return f(\*args, \*\*kwargs)

File "<stdin>", line 3, in clean\_string

TypeError: expected string or bytes-like object

at org.apache.spark.api.python.BasePythonRunner$ReaderIterator.handlePythonException(PythonRunner.scala:555)

at org.apache.spark.sql.execution.python.PythonUDFRunner$$anon$2.read(PythonUDFRunner.scala:86)

at org.apache.spark.sql.execution.python.PythonUDFRunner$$anon$2.read(PythonUDFRunner.scala:68)

at org.apache.spark.api.python.BasePythonRunner$ReaderIterator.hasNext(PythonRunner.scala:508)

at org.apache.spark.InterruptibleIterator.hasNext(InterruptibleIterator.scala:37)

at scala.collection.Iterator$$anon$11.hasNext(Iterator.scala:491)

at scala.collection.Iterator$$anon$10.hasNext(Iterator.scala:460)

at scala.collection.Iterator$$anon$10.hasNext(Iterator.scala:460)

at org.apache.spark.sql.catalyst.expressions.GeneratedClass$GeneratedIteratorForCodegenStage1.processNext(Unknown Source)

at org.apache.spark.sql.execution.BufferedRowIterator.hasNext(BufferedRowIterator.java:43)

at org.apache.spark.sql.execution.WholeStageCodegenExec$$anon$1.hasNext(WholeStageCodegenExec.scala:759)

at org.apache.spark.sql.execution.columnar.DefaultCachedBatchSerializer$$anon$1.next(InMemoryRelation.scala:87)

at org.apache.spark.sql.execution.columnar.DefaultCachedBatchSerializer$$anon$1.next(InMemoryRelation.scala:79)

at scala.collection.Iterator$$anon$10.next(Iterator.scala:461)

at org.apache.spark.storage.memory.MemoryStore.putIterator(MemoryStore.scala:224)

at org.apache.spark.storage.memory.MemoryStore.putIteratorAsValues(MemoryStore.scala:302)

at org.apache.spark.storage.BlockManager.$anonfun$doPutIterator$1(BlockManager.scala:1481)

at org.apache.spark.storage.BlockManager.org$apache$spark$storage$BlockManager$$doPut(BlockManager.scala:1408)

at org.apache.spark.storage.BlockManager.doPutIterator(BlockManager.scala:1472)

at org.apache.spark.storage.BlockManager.getOrElseUpdate(BlockManager.scala:1295)

at org.apache.spark.rdd.RDD.getOrCompute(RDD.scala:384)

at org.apache.spark.rdd.RDD.iterator(RDD.scala:335)

at org.apache.spark.rdd.MapPartitionsRDD.compute(MapPartitionsRDD.scala:52)

at org.apache.spark.rdd.RDD.computeOrReadCheckpoint(RDD.scala:373)

at org.apache.spark.rdd.RDD.iterator(RDD.scala:337)

at org.apache.spark.rdd.MapPartitionsRDD.compute(MapPartitionsRDD.scala:52)

at org.apache.spark.rdd.RDD.computeOrReadCheckpoint(RDD.scala:373)

at org.apache.spark.rdd.RDD.iterator(RDD.scala:337)

at org.apache.spark.rdd.MapPartitionsRDD.compute(MapPartitionsRDD.scala:52)

at org.apache.spark.rdd.RDD.computeOrReadCheckpoint(RDD.scala:373)

at org.apache.spark.rdd.RDD.iterator(RDD.scala:337)

at org.apache.spark.scheduler.ResultTask.runTask(ResultTask.scala:90)

at org.apache.spark.scheduler.Task.run(Task.scala:131)

at org.apache.spark.executor.Executor$TaskRunner.$anonfun$run$3(Executor.scala:506)

at org.apache.spark.util.Utils$.tryWithSafeFinally(Utils.scala:1462)

at org.apache.spark.executor.Executor$TaskRunner.run(Executor.scala:509)

at java.util.concurrent.ThreadPoolExecutor.runWorker(ThreadPoolExecutor.java:1149)

at java.util.concurrent.ThreadPoolExecutor$Worker.run(ThreadPoolExecutor.java:624)

... 1 more

>>> sample\_op.write.json("/user/hduser/Hackathon/JSON\_DATA", mode="overwrite")

24/07/22 09:25:09 WARN storage.BlockManager: Putting block rdd\_531\_0 failed due to exception org.apache.spark.api.python.PythonException: Traceback (most recent call last):

File "/usr/local/spark/python/lib/pyspark.zip/pyspark/worker.py", line 619, in main

process()

File "/usr/local/spark/python/lib/pyspark.zip/pyspark/worker.py", line 611, in process

serializer.dump\_stream(out\_iter, outfile)

File "/usr/local/spark/python/lib/pyspark.zip/pyspark/serializers.py", line 211, in dump\_stream

self.serializer.dump\_stream(self.\_batched(iterator), stream)

File "/usr/local/spark/python/lib/pyspark.zip/pyspark/serializers.py", line 132, in dump\_stream

for obj in iterator:

File "/usr/local/spark/python/lib/pyspark.zip/pyspark/serializers.py", line 200, in \_batched

for item in iterator:

File "/usr/local/spark/python/lib/pyspark.zip/pyspark/worker.py", line 452, in mapper

result = tuple(f(\*[a[o] for o in arg\_offsets]) for (arg\_offsets, f) in udfs)

File "/usr/local/spark/python/lib/pyspark.zip/pyspark/worker.py", line 452, in <genexpr>

result = tuple(f(\*[a[o] for o in arg\_offsets]) for (arg\_offsets, f) in udfs)

File "/usr/local/spark/python/lib/pyspark.zip/pyspark/worker.py", line 87, in <lambda>

return lambda \*a: f(\*a)

File "/usr/local/spark/python/lib/pyspark.zip/pyspark/util.py", line 74, in wrapper

return f(\*args, \*\*kwargs)

File "<stdin>", line 3, in clean\_string

TypeError: expected string or bytes-like object

.

24/07/22 09:25:09 WARN storage.BlockManager: Block rdd\_531\_0 could not be removed as it was not found on disk or in memory

24/07/22 09:25:09 ERROR executor.Executor: Exception in task 0.0 in stage 100.0 (TID 107)

org.apache.spark.api.python.PythonException: Traceback (most recent call last):

File "/usr/local/spark/python/lib/pyspark.zip/pyspark/worker.py", line 619, in main

process()

File "/usr/local/spark/python/lib/pyspark.zip/pyspark/worker.py", line 611, in process

serializer.dump\_stream(out\_iter, outfile)

File "/usr/local/spark/python/lib/pyspark.zip/pyspark/serializers.py", line 211, in dump\_stream

self.serializer.dump\_stream(self.\_batched(iterator), stream)

File "/usr/local/spark/python/lib/pyspark.zip/pyspark/serializers.py", line 132, in dump\_stream

for obj in iterator:

File "/usr/local/spark/python/lib/pyspark.zip/pyspark/serializers.py", line 200, in \_batched

for item in iterator:

File "/usr/local/spark/python/lib/pyspark.zip/pyspark/worker.py", line 452, in mapper

result = tuple(f(\*[a[o] for o in arg\_offsets]) for (arg\_offsets, f) in udfs)

File "/usr/local/spark/python/lib/pyspark.zip/pyspark/worker.py", line 452, in <genexpr>

result = tuple(f(\*[a[o] for o in arg\_offsets]) for (arg\_offsets, f) in udfs)

File "/usr/local/spark/python/lib/pyspark.zip/pyspark/worker.py", line 87, in <lambda>

return lambda \*a: f(\*a)

File "/usr/local/spark/python/lib/pyspark.zip/pyspark/util.py", line 74, in wrapper

return f(\*args, \*\*kwargs)

File "<stdin>", line 3, in clean\_string

TypeError: expected string or bytes-like object

at org.apache.spark.api.python.BasePythonRunner$ReaderIterator.handlePythonException(PythonRunner.scala:555)

at org.apache.spark.sql.execution.python.PythonUDFRunner$$anon$2.read(PythonUDFRunner.scala:86)

at org.apache.spark.sql.execution.python.PythonUDFRunner$$anon$2.read(PythonUDFRunner.scala:68)

at org.apache.spark.api.python.BasePythonRunner$ReaderIterator.hasNext(PythonRunner.scala:508)

at org.apache.spark.InterruptibleIterator.hasNext(InterruptibleIterator.scala:37)

at scala.collection.Iterator$$anon$11.hasNext(Iterator.scala:491)

at scala.collection.Iterator$$anon$10.hasNext(Iterator.scala:460)

at scala.collection.Iterator$$anon$10.hasNext(Iterator.scala:460)

at org.apache.spark.sql.catalyst.expressions.GeneratedClass$GeneratedIteratorForCodegenStage1.processNext(Unknown Source)

at org.apache.spark.sql.execution.BufferedRowIterator.hasNext(BufferedRowIterator.java:43)

at org.apache.spark.sql.execution.WholeStageCodegenExec$$anon$1.hasNext(WholeStageCodegenExec.scala:759)

at org.apache.spark.sql.execution.columnar.DefaultCachedBatchSerializer$$anon$1.next(InMemoryRelation.scala:87)

at org.apache.spark.sql.execution.columnar.DefaultCachedBatchSerializer$$anon$1.next(InMemoryRelation.scala:79)

at scala.collection.Iterator$$anon$10.next(Iterator.scala:461)

at org.apache.spark.storage.memory.MemoryStore.putIterator(MemoryStore.scala:224)

at org.apache.spark.storage.memory.MemoryStore.putIteratorAsValues(MemoryStore.scala:302)

at org.apache.spark.storage.BlockManager.$anonfun$doPutIterator$1(BlockManager.scala:1481)

at org.apache.spark.storage.BlockManager.org$apache$spark$storage$BlockManager$$doPut(BlockManager.scala:1408)

at org.apache.spark.storage.BlockManager.doPutIterator(BlockManager.scala:1472)

at org.apache.spark.storage.BlockManager.getOrElseUpdate(BlockManager.scala:1295)

at org.apache.spark.rdd.RDD.getOrCompute(RDD.scala:384)

at org.apache.spark.rdd.RDD.iterator(RDD.scala:335)

at org.apache.spark.rdd.MapPartitionsRDD.compute(MapPartitionsRDD.scala:52)

at org.apache.spark.rdd.RDD.computeOrReadCheckpoint(RDD.scala:373)

at org.apache.spark.rdd.RDD.iterator(RDD.scala:337)

at org.apache.spark.rdd.MapPartitionsRDD.compute(MapPartitionsRDD.scala:52)

at org.apache.spark.rdd.RDD.computeOrReadCheckpoint(RDD.scala:373)

at org.apache.spark.rdd.RDD.iterator(RDD.scala:337)

at org.apache.spark.rdd.MapPartitionsRDD.compute(MapPartitionsRDD.scala:52)

at org.apache.spark.rdd.RDD.computeOrReadCheckpoint(RDD.scala:373)

at org.apache.spark.rdd.RDD.iterator(RDD.scala:337)

at org.apache.spark.scheduler.ResultTask.runTask(ResultTask.scala:90)

at org.apache.spark.scheduler.Task.run(Task.scala:131)

at org.apache.spark.executor.Executor$TaskRunner.$anonfun$run$3(Executor.scala:506)

at org.apache.spark.util.Utils$.tryWithSafeFinally(Utils.scala:1462)

at org.apache.spark.executor.Executor$TaskRunner.run(Executor.scala:509)

at java.util.concurrent.ThreadPoolExecutor.runWorker(ThreadPoolExecutor.java:1149)

at java.util.concurrent.ThreadPoolExecutor$Worker.run(ThreadPoolExecutor.java:624)

at java.lang.Thread.run(Thread.java:750)

24/07/22 09:25:09 WARN scheduler.TaskSetManager: Lost task 0.0 in stage 100.0 (TID 107) (192.168.112.129 executor driver): org.apache.spark.api.python.PythonException: Traceback (most recent call last):

File "/usr/local/spark/python/lib/pyspark.zip/pyspark/worker.py", line 619, in main

process()

File "/usr/local/spark/python/lib/pyspark.zip/pyspark/worker.py", line 611, in process

serializer.dump\_stream(out\_iter, outfile)

File "/usr/local/spark/python/lib/pyspark.zip/pyspark/serializers.py", line 211, in dump\_stream

self.serializer.dump\_stream(self.\_batched(iterator), stream)

File "/usr/local/spark/python/lib/pyspark.zip/pyspark/serializers.py", line 132, in dump\_stream

for obj in iterator:

File "/usr/local/spark/python/lib/pyspark.zip/pyspark/serializers.py", line 200, in \_batched

for item in iterator:

File "/usr/local/spark/python/lib/pyspark.zip/pyspark/worker.py", line 452, in mapper

result = tuple(f(\*[a[o] for o in arg\_offsets]) for (arg\_offsets, f) in udfs)

File "/usr/local/spark/python/lib/pyspark.zip/pyspark/worker.py", line 452, in <genexpr>

result = tuple(f(\*[a[o] for o in arg\_offsets]) for (arg\_offsets, f) in udfs)

File "/usr/local/spark/python/lib/pyspark.zip/pyspark/worker.py", line 87, in <lambda>

return lambda \*a: f(\*a)

File "/usr/local/spark/python/lib/pyspark.zip/pyspark/util.py", line 74, in wrapper

return f(\*args, \*\*kwargs)

File "<stdin>", line 3, in clean\_string

TypeError: expected string or bytes-like object

at org.apache.spark.api.python.BasePythonRunner$ReaderIterator.handlePythonException(PythonRunner.scala:555)

at org.apache.spark.sql.execution.python.PythonUDFRunner$$anon$2.read(PythonUDFRunner.scala:86)

at org.apache.spark.sql.execution.python.PythonUDFRunner$$anon$2.read(PythonUDFRunner.scala:68)

at org.apache.spark.api.python.BasePythonRunner$ReaderIterator.hasNext(PythonRunner.scala:508)

at org.apache.spark.InterruptibleIterator.hasNext(InterruptibleIterator.scala:37)

at scala.collection.Iterator$$anon$11.hasNext(Iterator.scala:491)

at scala.collection.Iterator$$anon$10.hasNext(Iterator.scala:460)

at scala.collection.Iterator$$anon$10.hasNext(Iterator.scala:460)

at org.apache.spark.sql.catalyst.expressions.GeneratedClass$GeneratedIteratorForCodegenStage1.processNext(Unknown Source)

at org.apache.spark.sql.execution.BufferedRowIterator.hasNext(BufferedRowIterator.java:43)

at org.apache.spark.sql.execution.WholeStageCodegenExec$$anon$1.hasNext(WholeStageCodegenExec.scala:759)

at org.apache.spark.sql.execution.columnar.DefaultCachedBatchSerializer$$anon$1.next(InMemoryRelation.scala:87)

at org.apache.spark.sql.execution.columnar.DefaultCachedBatchSerializer$$anon$1.next(InMemoryRelation.scala:79)

at scala.collection.Iterator$$anon$10.next(Iterator.scala:461)

at org.apache.spark.storage.memory.MemoryStore.putIterator(MemoryStore.scala:224)

at org.apache.spark.storage.memory.MemoryStore.putIteratorAsValues(MemoryStore.scala:302)

at org.apache.spark.storage.BlockManager.$anonfun$doPutIterator$1(BlockManager.scala:1481)

at org.apache.spark.storage.BlockManager.org$apache$spark$storage$BlockManager$$doPut(BlockManager.scala:1408)

at org.apache.spark.storage.BlockManager.doPutIterator(BlockManager.scala:1472)

at org.apache.spark.storage.BlockManager.getOrElseUpdate(BlockManager.scala:1295)

at org.apache.spark.rdd.RDD.getOrCompute(RDD.scala:384)

at org.apache.spark.rdd.RDD.iterator(RDD.scala:335)

at org.apache.spark.rdd.MapPartitionsRDD.compute(MapPartitionsRDD.scala:52)

at org.apache.spark.rdd.RDD.computeOrReadCheckpoint(RDD.scala:373)

at org.apache.spark.rdd.RDD.iterator(RDD.scala:337)

at org.apache.spark.rdd.MapPartitionsRDD.compute(MapPartitionsRDD.scala:52)

at org.apache.spark.rdd.RDD.computeOrReadCheckpoint(RDD.scala:373)

at org.apache.spark.rdd.RDD.iterator(RDD.scala:337)

at org.apache.spark.rdd.MapPartitionsRDD.compute(MapPartitionsRDD.scala:52)

at org.apache.spark.rdd.RDD.computeOrReadCheckpoint(RDD.scala:373)

at org.apache.spark.rdd.RDD.iterator(RDD.scala:337)

at org.apache.spark.scheduler.ResultTask.runTask(ResultTask.scala:90)

at org.apache.spark.scheduler.Task.run(Task.scala:131)

at org.apache.spark.executor.Executor$TaskRunner.$anonfun$run$3(Executor.scala:506)

at org.apache.spark.util.Utils$.tryWithSafeFinally(Utils.scala:1462)

at org.apache.spark.executor.Executor$TaskRunner.run(Executor.scala:509)

at java.util.concurrent.ThreadPoolExecutor.runWorker(ThreadPoolExecutor.java:1149)

at java.util.concurrent.ThreadPoolExecutor$Worker.run(ThreadPoolExecutor.java:624)

at java.lang.Thread.run(Thread.java:750)

24/07/22 09:25:09 ERROR scheduler.TaskSetManager: Task 0 in stage 100.0 failed 1 times; aborting job

24/07/22 09:25:09 ERROR datasources.FileFormatWriter: Aborting job b1f8f9e0-e86b-4042-8047-ebacc269db59.

org.apache.spark.SparkException: Job aborted due to stage failure: Task 0 in stage 100.0 failed 1 times, most recent failure: Lost task 0.0 in stage 100.0 (TID 107) (192.168.112.129 executor driver): org.apache.spark.api.python.PythonException: Traceback (most recent call last):

File "/usr/local/spark/python/lib/pyspark.zip/pyspark/worker.py", line 619, in main

process()

File "/usr/local/spark/python/lib/pyspark.zip/pyspark/worker.py", line 611, in process

serializer.dump\_stream(out\_iter, outfile)

File "/usr/local/spark/python/lib/pyspark.zip/pyspark/serializers.py", line 211, in dump\_stream

self.serializer.dump\_stream(self.\_batched(iterator), stream)

File "/usr/local/spark/python/lib/pyspark.zip/pyspark/serializers.py", line 132, in dump\_stream

for obj in iterator:

File "/usr/local/spark/python/lib/pyspark.zip/pyspark/serializers.py", line 200, in \_batched

for item in iterator:

File "/usr/local/spark/python/lib/pyspark.zip/pyspark/worker.py", line 452, in mapper

result = tuple(f(\*[a[o] for o in arg\_offsets]) for (arg\_offsets, f) in udfs)

File "/usr/local/spark/python/lib/pyspark.zip/pyspark/worker.py", line 452, in <genexpr>

result = tuple(f(\*[a[o] for o in arg\_offsets]) for (arg\_offsets, f) in udfs)

File "/usr/local/spark/python/lib/pyspark.zip/pyspark/worker.py", line 87, in <lambda>

return lambda \*a: f(\*a)

File "/usr/local/spark/python/lib/pyspark.zip/pyspark/util.py", line 74, in wrapper

return f(\*args, \*\*kwargs)

File "<stdin>", line 3, in clean\_string

TypeError: expected string or bytes-like object

at org.apache.spark.api.python.BasePythonRunner$ReaderIterator.handlePythonException(PythonRunner.scala:555)

at org.apache.spark.sql.execution.python.PythonUDFRunner$$anon$2.read(PythonUDFRunner.scala:86)

at org.apache.spark.sql.execution.python.PythonUDFRunner$$anon$2.read(PythonUDFRunner.scala:68)

at org.apache.spark.api.python.BasePythonRunner$ReaderIterator.hasNext(PythonRunner.scala:508)

at org.apache.spark.InterruptibleIterator.hasNext(InterruptibleIterator.scala:37)

at scala.collection.Iterator$$anon$11.hasNext(Iterator.scala:491)

at scala.collection.Iterator$$anon$10.hasNext(Iterator.scala:460)

at scala.collection.Iterator$$anon$10.hasNext(Iterator.scala:460)

at org.apache.spark.sql.catalyst.expressions.GeneratedClass$GeneratedIteratorForCodegenStage1.processNext(Unknown Source)

at org.apache.spark.sql.execution.BufferedRowIterator.hasNext(BufferedRowIterator.java:43)

at org.apache.spark.sql.execution.WholeStageCodegenExec$$anon$1.hasNext(WholeStageCodegenExec.scala:759)

at org.apache.spark.sql.execution.columnar.DefaultCachedBatchSerializer$$anon$1.next(InMemoryRelation.scala:87)

at org.apache.spark.sql.execution.columnar.DefaultCachedBatchSerializer$$anon$1.next(InMemoryRelation.scala:79)

at scala.collection.Iterator$$anon$10.next(Iterator.scala:461)

at org.apache.spark.storage.memory.MemoryStore.putIterator(MemoryStore.scala:224)

at org.apache.spark.storage.memory.MemoryStore.putIteratorAsValues(MemoryStore.scala:302)

at org.apache.spark.storage.BlockManager.$anonfun$doPutIterator$1(BlockManager.scala:1481)

at org.apache.spark.storage.BlockManager.org$apache$spark$storage$BlockManager$$doPut(BlockManager.scala:1408)

at org.apache.spark.storage.BlockManager.doPutIterator(BlockManager.scala:1472)

at org.apache.spark.storage.BlockManager.getOrElseUpdate(BlockManager.scala:1295)

at org.apache.spark.rdd.RDD.getOrCompute(RDD.scala:384)

at org.apache.spark.rdd.RDD.iterator(RDD.scala:335)

at org.apache.spark.rdd.MapPartitionsRDD.compute(MapPartitionsRDD.scala:52)

at org.apache.spark.rdd.RDD.computeOrReadCheckpoint(RDD.scala:373)

at org.apache.spark.rdd.RDD.iterator(RDD.scala:337)

at org.apache.spark.rdd.MapPartitionsRDD.compute(MapPartitionsRDD.scala:52)

at org.apache.spark.rdd.RDD.computeOrReadCheckpoint(RDD.scala:373)

at org.apache.spark.rdd.RDD.iterator(RDD.scala:337)

at org.apache.spark.rdd.MapPartitionsRDD.compute(MapPartitionsRDD.scala:52)

at org.apache.spark.rdd.RDD.computeOrReadCheckpoint(RDD.scala:373)

at org.apache.spark.rdd.RDD.iterator(RDD.scala:337)

at org.apache.spark.scheduler.ResultTask.runTask(ResultTask.scala:90)

at org.apache.spark.scheduler.Task.run(Task.scala:131)

at org.apache.spark.executor.Executor$TaskRunner.$anonfun$run$3(Executor.scala:506)

at org.apache.spark.util.Utils$.tryWithSafeFinally(Utils.scala:1462)

at org.apache.spark.executor.Executor$TaskRunner.run(Executor.scala:509)

at java.util.concurrent.ThreadPoolExecutor.runWorker(ThreadPoolExecutor.java:1149)

at java.util.concurrent.ThreadPoolExecutor$Worker.run(ThreadPoolExecutor.java:624)

at java.lang.Thread.run(Thread.java:750)

Driver stacktrace:

at org.apache.spark.scheduler.DAGScheduler.failJobAndIndependentStages(DAGScheduler.scala:2454)

at org.apache.spark.scheduler.DAGScheduler.$anonfun$abortStage$2(DAGScheduler.scala:2403)

at org.apache.spark.scheduler.DAGScheduler.$anonfun$abortStage$2$adapted(DAGScheduler.scala:2402)

at scala.collection.mutable.ResizableArray.foreach(ResizableArray.scala:62)

at scala.collection.mutable.ResizableArray.foreach$(ResizableArray.scala:55)

at scala.collection.mutable.ArrayBuffer.foreach(ArrayBuffer.scala:49)

at org.apache.spark.scheduler.DAGScheduler.abortStage(DAGScheduler.scala:2402)

at org.apache.spark.scheduler.DAGScheduler.$anonfun$handleTaskSetFailed$1(DAGScheduler.scala:1160)

at org.apache.spark.scheduler.DAGScheduler.$anonfun$handleTaskSetFailed$1$adapted(DAGScheduler.scala:1160)

at scala.Option.foreach(Option.scala:407)

at org.apache.spark.scheduler.DAGScheduler.handleTaskSetFailed(DAGScheduler.scala:1160)

at org.apache.spark.scheduler.DAGSchedulerEventProcessLoop.doOnReceive(DAGScheduler.scala:2642)

at org.apache.spark.scheduler.DAGSchedulerEventProcessLoop.onReceive(DAGScheduler.scala:2584)

at org.apache.spark.scheduler.DAGSchedulerEventProcessLoop.onReceive(DAGScheduler.scala:2573)

at org.apache.spark.util.EventLoop$$anon$1.run(EventLoop.scala:49)

at org.apache.spark.scheduler.DAGScheduler.runJob(DAGScheduler.scala:938)

at org.apache.spark.SparkContext.runJob(SparkContext.scala:2214)

at org.apache.spark.sql.execution.datasources.FileFormatWriter$.write(FileFormatWriter.scala:218)

at org.apache.spark.sql.execution.datasources.InsertIntoHadoopFsRelationCommand.run(InsertIntoHadoopFsRelationCommand.scala:186)

at org.apache.spark.sql.execution.command.DataWritingCommandExec.sideEffectResult$lzycompute(commands.scala:113)

at org.apache.spark.sql.execution.command.DataWritingCommandExec.sideEffectResult(commands.scala:111)

at org.apache.spark.sql.execution.command.DataWritingCommandExec.executeCollect(commands.scala:125)

at org.apache.spark.sql.execution.QueryExecution$$anonfun$eagerlyExecuteCommands$1.$anonfun$applyOrElse$1(QueryExecution.scala:110)

at org.apache.spark.sql.execution.SQLExecution$.$anonfun$withNewExecutionId$5(SQLExecution.scala:103)

at org.apache.spark.sql.execution.SQLExecution$.withSQLConfPropagated(SQLExecution.scala:163)

at org.apache.spark.sql.execution.SQLExecution$.$anonfun$withNewExecutionId$1(SQLExecution.scala:90)

at org.apache.spark.sql.SparkSession.withActive(SparkSession.scala:775)

at org.apache.spark.sql.execution.SQLExecution$.withNewExecutionId(SQLExecution.scala:64)

at org.apache.spark.sql.execution.QueryExecution$$anonfun$eagerlyExecuteCommands$1.applyOrElse(QueryExecution.scala:110)

at org.apache.spark.sql.execution.QueryExecution$$anonfun$eagerlyExecuteCommands$1.applyOrElse(QueryExecution.scala:106)

at org.apache.spark.sql.catalyst.trees.TreeNode.$anonfun$transformDownWithPruning$1(TreeNode.scala:481)

at org.apache.spark.sql.catalyst.trees.CurrentOrigin$.withOrigin(TreeNode.scala:82)

at org.apache.spark.sql.catalyst.trees.TreeNode.transformDownWithPruning(TreeNode.scala:481)

at org.apache.spark.sql.catalyst.plans.logical.LogicalPlan.org$apache$spark$sql$catalyst$plans$logical$AnalysisHelper$$super$transformDownWithPruning(LogicalPlan.scala:30)

at org.apache.spark.sql.catalyst.plans.logical.AnalysisHelper.transformDownWithPruning(AnalysisHelper.scala:267)

at org.apache.spark.sql.catalyst.plans.logical.AnalysisHelper.transformDownWithPruning$(AnalysisHelper.scala:263)

at org.apache.spark.sql.catalyst.plans.logical.LogicalPlan.transformDownWithPruning(LogicalPlan.scala:30)

at org.apache.spark.sql.catalyst.plans.logical.LogicalPlan.transformDownWithPruning(LogicalPlan.scala:30)

at org.apache.spark.sql.catalyst.trees.TreeNode.transformDown(TreeNode.scala:457)

at org.apache.spark.sql.execution.QueryExecution.eagerlyExecuteCommands(QueryExecution.scala:106)

at org.apache.spark.sql.execution.QueryExecution.commandExecuted$lzycompute(QueryExecution.scala:93)

at org.apache.spark.sql.execution.QueryExecution.commandExecuted(QueryExecution.scala:91)

at org.apache.spark.sql.execution.QueryExecution.assertCommandExecuted(QueryExecution.scala:128)

at org.apache.spark.sql.DataFrameWriter.runCommand(DataFrameWriter.scala:848)

at org.apache.spark.sql.DataFrameWriter.saveToV1Source(DataFrameWriter.scala:382)

at org.apache.spark.sql.DataFrameWriter.saveInternal(DataFrameWriter.scala:355)

at org.apache.spark.sql.DataFrameWriter.save(DataFrameWriter.scala:239)

at org.apache.spark.sql.DataFrameWriter.json(DataFrameWriter.scala:763)

at sun.reflect.NativeMethodAccessorImpl.invoke0(Native Method)

at sun.reflect.NativeMethodAccessorImpl.invoke(NativeMethodAccessorImpl.java:62)

at sun.reflect.DelegatingMethodAccessorImpl.invoke(DelegatingMethodAccessorImpl.java:43)

at java.lang.reflect.Method.invoke(Method.java:498)

at py4j.reflection.MethodInvoker.invoke(MethodInvoker.java:244)

at py4j.reflection.ReflectionEngine.invoke(ReflectionEngine.java:357)

at py4j.Gateway.invoke(Gateway.java:282)

at py4j.commands.AbstractCommand.invokeMethod(AbstractCommand.java:132)

at py4j.commands.CallCommand.execute(CallCommand.java:79)

at py4j.ClientServerConnection.waitForCommands(ClientServerConnection.java:182)

at py4j.ClientServerConnection.run(ClientServerConnection.java:106)

at java.lang.Thread.run(Thread.java:750)

Caused by: org.apache.spark.api.python.PythonException: Traceback (most recent call last):

File "/usr/local/spark/python/lib/pyspark.zip/pyspark/worker.py", line 619, in main

process()

File "/usr/local/spark/python/lib/pyspark.zip/pyspark/worker.py", line 611, in process

serializer.dump\_stream(out\_iter, outfile)

File "/usr/local/spark/python/lib/pyspark.zip/pyspark/serializers.py", line 211, in dump\_stream

self.serializer.dump\_stream(self.\_batched(iterator), stream)

File "/usr/local/spark/python/lib/pyspark.zip/pyspark/serializers.py", line 132, in dump\_stream

for obj in iterator:

File "/usr/local/spark/python/lib/pyspark.zip/pyspark/serializers.py", line 200, in \_batched

for item in iterator:

File "/usr/local/spark/python/lib/pyspark.zip/pyspark/worker.py", line 452, in mapper

result = tuple(f(\*[a[o] for o in arg\_offsets]) for (arg\_offsets, f) in udfs)

File "/usr/local/spark/python/lib/pyspark.zip/pyspark/worker.py", line 452, in <genexpr>

result = tuple(f(\*[a[o] for o in arg\_offsets]) for (arg\_offsets, f) in udfs)

File "/usr/local/spark/python/lib/pyspark.zip/pyspark/worker.py", line 87, in <lambda>

return lambda \*a: f(\*a)

File "/usr/local/spark/python/lib/pyspark.zip/pyspark/util.py", line 74, in wrapper

return f(\*args, \*\*kwargs)

File "<stdin>", line 3, in clean\_string

TypeError: expected string or bytes-like object

at org.apache.spark.api.python.BasePythonRunner$ReaderIterator.handlePythonException(PythonRunner.scala:555)

at org.apache.spark.sql.execution.python.PythonUDFRunner$$anon$2.read(PythonUDFRunner.scala:86)

at org.apache.spark.sql.execution.python.PythonUDFRunner$$anon$2.read(PythonUDFRunner.scala:68)

at org.apache.spark.api.python.BasePythonRunner$ReaderIterator.hasNext(PythonRunner.scala:508)

at org.apache.spark.InterruptibleIterator.hasNext(InterruptibleIterator.scala:37)

at scala.collection.Iterator$$anon$11.hasNext(Iterator.scala:491)

at scala.collection.Iterator$$anon$10.hasNext(Iterator.scala:460)

at scala.collection.Iterator$$anon$10.hasNext(Iterator.scala:460)

at org.apache.spark.sql.catalyst.expressions.GeneratedClass$GeneratedIteratorForCodegenStage1.processNext(Unknown Source)

at org.apache.spark.sql.execution.BufferedRowIterator.hasNext(BufferedRowIterator.java:43)

at org.apache.spark.sql.execution.WholeStageCodegenExec$$anon$1.hasNext(WholeStageCodegenExec.scala:759)

at org.apache.spark.sql.execution.columnar.DefaultCachedBatchSerializer$$anon$1.next(InMemoryRelation.scala:87)

at org.apache.spark.sql.execution.columnar.DefaultCachedBatchSerializer$$anon$1.next(InMemoryRelation.scala:79)

at scala.collection.Iterator$$anon$10.next(Iterator.scala:461)

at org.apache.spark.storage.memory.MemoryStore.putIterator(MemoryStore.scala:224)

at org.apache.spark.storage.memory.MemoryStore.putIteratorAsValues(MemoryStore.scala:302)

at org.apache.spark.storage.BlockManager.$anonfun$doPutIterator$1(BlockManager.scala:1481)

at org.apache.spark.storage.BlockManager.org$apache$spark$storage$BlockManager$$doPut(BlockManager.scala:1408)

at org.apache.spark.storage.BlockManager.doPutIterator(BlockManager.scala:1472)

at org.apache.spark.storage.BlockManager.getOrElseUpdate(BlockManager.scala:1295)

at org.apache.spark.rdd.RDD.getOrCompute(RDD.scala:384)

at org.apache.spark.rdd.RDD.iterator(RDD.scala:335)

at org.apache.spark.rdd.MapPartitionsRDD.compute(MapPartitionsRDD.scala:52)

at org.apache.spark.rdd.RDD.computeOrReadCheckpoint(RDD.scala:373)

at org.apache.spark.rdd.RDD.iterator(RDD.scala:337)

at org.apache.spark.rdd.MapPartitionsRDD.compute(MapPartitionsRDD.scala:52)

at org.apache.spark.rdd.RDD.computeOrReadCheckpoint(RDD.scala:373)

at org.apache.spark.rdd.RDD.iterator(RDD.scala:337)

at org.apache.spark.rdd.MapPartitionsRDD.compute(MapPartitionsRDD.scala:52)

at org.apache.spark.rdd.RDD.computeOrReadCheckpoint(RDD.scala:373)

at org.apache.spark.rdd.RDD.iterator(RDD.scala:337)

at org.apache.spark.scheduler.ResultTask.runTask(ResultTask.scala:90)

at org.apache.spark.scheduler.Task.run(Task.scala:131)

at org.apache.spark.executor.Executor$TaskRunner.$anonfun$run$3(Executor.scala:506)

at org.apache.spark.util.Utils$.tryWithSafeFinally(Utils.scala:1462)

at org.apache.spark.executor.Executor$TaskRunner.run(Executor.scala:509)

at java.util.concurrent.ThreadPoolExecutor.runWorker(ThreadPoolExecutor.java:1149)

at java.util.concurrent.ThreadPoolExecutor$Worker.run(ThreadPoolExecutor.java:624)

... 1 more

24/07/22 09:25:09 WARN storage.BlockManager: Putting block rdd\_531\_1 failed due to exception org.apache.spark.TaskKilledException.

24/07/22 09:25:09 WARN storage.BlockManager: Block rdd\_531\_1 could not be removed as it was not found on disk or in memory

24/07/22 09:25:09 WARN scheduler.TaskSetManager: Lost task 1.0 in stage 100.0 (TID 108) (192.168.112.129 executor driver): TaskKilled (Stage cancelled)

Traceback (most recent call last):

File "<stdin>", line 1, in <module>

File "/usr/local/spark/python/pyspark/sql/readwriter.py", line 846, in json

self.\_jwrite.json(path)

File "/usr/local/spark/python/lib/py4j-0.10.9.3-src.zip/py4j/java\_gateway.py", line 1322, in \_\_call\_\_

File "/usr/local/spark/python/pyspark/sql/utils.py", line 111, in deco

return f(\*a, \*\*kw)

File "/usr/local/spark/python/lib/py4j-0.10.9.3-src.zip/py4j/protocol.py", line 328, in get\_return\_value

py4j.protocol.Py4JJavaError: An error occurred while calling o612.json.

: org.apache.spark.SparkException: Job aborted.

at org.apache.spark.sql.errors.QueryExecutionErrors$.jobAbortedError(QueryExecutionErrors.scala:496)

at org.apache.spark.sql.execution.datasources.FileFormatWriter$.write(FileFormatWriter.scala:251)

at org.apache.spark.sql.execution.datasources.InsertIntoHadoopFsRelationCommand.run(InsertIntoHadoopFsRelationCommand.scala:186)

at org.apache.spark.sql.execution.command.DataWritingCommandExec.sideEffectResult$lzycompute(commands.scala:113)

at org.apache.spark.sql.execution.command.DataWritingCommandExec.sideEffectResult(commands.scala:111)

at org.apache.spark.sql.execution.command.DataWritingCommandExec.executeCollect(commands.scala:125)

at org.apache.spark.sql.execution.QueryExecution$$anonfun$eagerlyExecuteCommands$1.$anonfun$applyOrElse$1(QueryExecution.scala:110)

at org.apache.spark.sql.execution.SQLExecution$.$anonfun$withNewExecutionId$5(SQLExecution.scala:103)

at org.apache.spark.sql.execution.SQLExecution$.withSQLConfPropagated(SQLExecution.scala:163)

at org.apache.spark.sql.execution.SQLExecution$.$anonfun$withNewExecutionId$1(SQLExecution.scala:90)

at org.apache.spark.sql.SparkSession.withActive(SparkSession.scala:775)

at org.apache.spark.sql.execution.SQLExecution$.withNewExecutionId(SQLExecution.scala:64)

at org.apache.spark.sql.execution.QueryExecution$$anonfun$eagerlyExecuteCommands$1.applyOrElse(QueryExecution.scala:110)

at org.apache.spark.sql.execution.QueryExecution$$anonfun$eagerlyExecuteCommands$1.applyOrElse(QueryExecution.scala:106)

at org.apache.spark.sql.catalyst.trees.TreeNode.$anonfun$transformDownWithPruning$1(TreeNode.scala:481)

at org.apache.spark.sql.catalyst.trees.CurrentOrigin$.withOrigin(TreeNode.scala:82)

at org.apache.spark.sql.catalyst.trees.TreeNode.transformDownWithPruning(TreeNode.scala:481)

at org.apache.spark.sql.catalyst.plans.logical.LogicalPlan.org$apache$spark$sql$catalyst$plans$logical$AnalysisHelper$$super$transformDownWithPruning(LogicalPlan.scala:30)

at org.apache.spark.sql.catalyst.plans.logical.AnalysisHelper.transformDownWithPruning(AnalysisHelper.scala:267)

at org.apache.spark.sql.catalyst.plans.logical.AnalysisHelper.transformDownWithPruning$(AnalysisHelper.scala:263)

at org.apache.spark.sql.catalyst.plans.logical.LogicalPlan.transformDownWithPruning(LogicalPlan.scala:30)

at org.apache.spark.sql.catalyst.plans.logical.LogicalPlan.transformDownWithPruning(LogicalPlan.scala:30)

at org.apache.spark.sql.catalyst.trees.TreeNode.transformDown(TreeNode.scala:457)

at org.apache.spark.sql.execution.QueryExecution.eagerlyExecuteCommands(QueryExecution.scala:106)

at org.apache.spark.sql.execution.QueryExecution.commandExecuted$lzycompute(QueryExecution.scala:93)

at org.apache.spark.sql.execution.QueryExecution.commandExecuted(QueryExecution.scala:91)

at org.apache.spark.sql.execution.QueryExecution.assertCommandExecuted(QueryExecution.scala:128)

at org.apache.spark.sql.DataFrameWriter.runCommand(DataFrameWriter.scala:848)

at org.apache.spark.sql.DataFrameWriter.saveToV1Source(DataFrameWriter.scala:382)

at org.apache.spark.sql.DataFrameWriter.saveInternal(DataFrameWriter.scala:355)

at org.apache.spark.sql.DataFrameWriter.save(DataFrameWriter.scala:239)

at org.apache.spark.sql.DataFrameWriter.json(DataFrameWriter.scala:763)

at sun.reflect.NativeMethodAccessorImpl.invoke0(Native Method)

at sun.reflect.NativeMethodAccessorImpl.invoke(NativeMethodAccessorImpl.java:62)

at sun.reflect.DelegatingMethodAccessorImpl.invoke(DelegatingMethodAccessorImpl.java:43)

at java.lang.reflect.Method.invoke(Method.java:498)

at py4j.reflection.MethodInvoker.invoke(MethodInvoker.java:244)

at py4j.reflection.ReflectionEngine.invoke(ReflectionEngine.java:357)

at py4j.Gateway.invoke(Gateway.java:282)

at py4j.commands.AbstractCommand.invokeMethod(AbstractCommand.java:132)

at py4j.commands.CallCommand.execute(CallCommand.java:79)

at py4j.ClientServerConnection.waitForCommands(ClientServerConnection.java:182)

at py4j.ClientServerConnection.run(ClientServerConnection.java:106)

at java.lang.Thread.run(Thread.java:750)

Caused by: org.apache.spark.SparkException: Job aborted due to stage failure: Task 0 in stage 100.0 failed 1 times, most recent failure: Lost task 0.0 in stage 100.0 (TID 107) (192.168.112.129 executor driver): org.apache.spark.api.python.PythonException: Traceback (most recent call last):

File "/usr/local/spark/python/lib/pyspark.zip/pyspark/worker.py", line 619, in main

process()

File "/usr/local/spark/python/lib/pyspark.zip/pyspark/worker.py", line 611, in process

serializer.dump\_stream(out\_iter, outfile)

File "/usr/local/spark/python/lib/pyspark.zip/pyspark/serializers.py", line 211, in dump\_stream

self.serializer.dump\_stream(self.\_batched(iterator), stream)

File "/usr/local/spark/python/lib/pyspark.zip/pyspark/serializers.py", line 132, in dump\_stream

for obj in iterator:

File "/usr/local/spark/python/lib/pyspark.zip/pyspark/serializers.py", line 200, in \_batched

for item in iterator:

File "/usr/local/spark/python/lib/pyspark.zip/pyspark/worker.py", line 452, in mapper

result = tuple(f(\*[a[o] for o in arg\_offsets]) for (arg\_offsets, f) in udfs)

File "/usr/local/spark/python/lib/pyspark.zip/pyspark/worker.py", line 452, in <genexpr>

result = tuple(f(\*[a[o] for o in arg\_offsets]) for (arg\_offsets, f) in udfs)

File "/usr/local/spark/python/lib/pyspark.zip/pyspark/worker.py", line 87, in <lambda>

return lambda \*a: f(\*a)

File "/usr/local/spark/python/lib/pyspark.zip/pyspark/util.py", line 74, in wrapper

return f(\*args, \*\*kwargs)

File "<stdin>", line 3, in clean\_string

TypeError: expected string or bytes-like object

at org.apache.spark.api.python.BasePythonRunner$ReaderIterator.handlePythonException(PythonRunner.scala:555)

at org.apache.spark.sql.execution.python.PythonUDFRunner$$anon$2.read(PythonUDFRunner.scala:86)

at org.apache.spark.sql.execution.python.PythonUDFRunner$$anon$2.read(PythonUDFRunner.scala:68)

at org.apache.spark.api.python.BasePythonRunner$ReaderIterator.hasNext(PythonRunner.scala:508)

at org.apache.spark.InterruptibleIterator.hasNext(InterruptibleIterator.scala:37)

at scala.collection.Iterator$$anon$11.hasNext(Iterator.scala:491)

at scala.collection.Iterator$$anon$10.hasNext(Iterator.scala:460)

at scala.collection.Iterator$$anon$10.hasNext(Iterator.scala:460)

at org.apache.spark.sql.catalyst.expressions.GeneratedClass$GeneratedIteratorForCodegenStage1.processNext(Unknown Source)

at org.apache.spark.sql.execution.BufferedRowIterator.hasNext(BufferedRowIterator.java:43)

at org.apache.spark.sql.execution.WholeStageCodegenExec$$anon$1.hasNext(WholeStageCodegenExec.scala:759)

at org.apache.spark.sql.execution.columnar.DefaultCachedBatchSerializer$$anon$1.next(InMemoryRelation.scala:87)

at org.apache.spark.sql.execution.columnar.DefaultCachedBatchSerializer$$anon$1.next(InMemoryRelation.scala:79)

at scala.collection.Iterator$$anon$10.next(Iterator.scala:461)

at org.apache.spark.storage.memory.MemoryStore.putIterator(MemoryStore.scala:224)

at org.apache.spark.storage.memory.MemoryStore.putIteratorAsValues(MemoryStore.scala:302)

at org.apache.spark.storage.BlockManager.$anonfun$doPutIterator$1(BlockManager.scala:1481)

at org.apache.spark.storage.BlockManager.org$apache$spark$storage$BlockManager$$doPut(BlockManager.scala:1408)

at org.apache.spark.storage.BlockManager.doPutIterator(BlockManager.scala:1472)

at org.apache.spark.storage.BlockManager.getOrElseUpdate(BlockManager.scala:1295)

at org.apache.spark.rdd.RDD.getOrCompute(RDD.scala:384)

at org.apache.spark.rdd.RDD.iterator(RDD.scala:335)

at org.apache.spark.rdd.MapPartitionsRDD.compute(MapPartitionsRDD.scala:52)

at org.apache.spark.rdd.RDD.computeOrReadCheckpoint(RDD.scala:373)

at org.apache.spark.rdd.RDD.iterator(RDD.scala:337)

at org.apache.spark.rdd.MapPartitionsRDD.compute(MapPartitionsRDD.scala:52)

at org.apache.spark.rdd.RDD.computeOrReadCheckpoint(RDD.scala:373)

at org.apache.spark.rdd.RDD.iterator(RDD.scala:337)

at org.apache.spark.rdd.MapPartitionsRDD.compute(MapPartitionsRDD.scala:52)

at org.apache.spark.rdd.RDD.computeOrReadCheckpoint(RDD.scala:373)

at org.apache.spark.rdd.RDD.iterator(RDD.scala:337)

at org.apache.spark.scheduler.ResultTask.runTask(ResultTask.scala:90)

at org.apache.spark.scheduler.Task.run(Task.scala:131)

at org.apache.spark.executor.Executor$TaskRunner.$anonfun$run$3(Executor.scala:506)

at org.apache.spark.util.Utils$.tryWithSafeFinally(Utils.scala:1462)

at org.apache.spark.executor.Executor$TaskRunner.run(Executor.scala:509)

at java.util.concurrent.ThreadPoolExecutor.runWorker(ThreadPoolExecutor.java:1149)

at java.util.concurrent.ThreadPoolExecutor$Worker.run(ThreadPoolExecutor.java:624)

at java.lang.Thread.run(Thread.java:750)

Driver stacktrace:

at org.apache.spark.scheduler.DAGScheduler.failJobAndIndependentStages(DAGScheduler.scala:2454)

at org.apache.spark.scheduler.DAGScheduler.$anonfun$abortStage$2(DAGScheduler.scala:2403)

at org.apache.spark.scheduler.DAGScheduler.$anonfun$abortStage$2$adapted(DAGScheduler.scala:2402)

at scala.collection.mutable.ResizableArray.foreach(ResizableArray.scala:62)

at scala.collection.mutable.ResizableArray.foreach$(ResizableArray.scala:55)

at scala.collection.mutable.ArrayBuffer.foreach(ArrayBuffer.scala:49)

at org.apache.spark.scheduler.DAGScheduler.abortStage(DAGScheduler.scala:2402)

at org.apache.spark.scheduler.DAGScheduler.$anonfun$handleTaskSetFailed$1(DAGScheduler.scala:1160)

at org.apache.spark.scheduler.DAGScheduler.$anonfun$handleTaskSetFailed$1$adapted(DAGScheduler.scala:1160)

at scala.Option.foreach(Option.scala:407)

at org.apache.spark.scheduler.DAGScheduler.handleTaskSetFailed(DAGScheduler.scala:1160)

at org.apache.spark.scheduler.DAGSchedulerEventProcessLoop.doOnReceive(DAGScheduler.scala:2642)

at org.apache.spark.scheduler.DAGSchedulerEventProcessLoop.onReceive(DAGScheduler.scala:2584)

at org.apache.spark.scheduler.DAGSchedulerEventProcessLoop.onReceive(DAGScheduler.scala:2573)

at org.apache.spark.util.EventLoop$$anon$1.run(EventLoop.scala:49)

at org.apache.spark.scheduler.DAGScheduler.runJob(DAGScheduler.scala:938)

at org.apache.spark.SparkContext.runJob(SparkContext.scala:2214)

at org.apache.spark.sql.execution.datasources.FileFormatWriter$.write(FileFormatWriter.scala:218)

... 42 more

Caused by: org.apache.spark.api.python.PythonException: Traceback (most recent call last):

File "/usr/local/spark/python/lib/pyspark.zip/pyspark/worker.py", line 619, in main

process()

File "/usr/local/spark/python/lib/pyspark.zip/pyspark/worker.py", line 611, in process

serializer.dump\_stream(out\_iter, outfile)

File "/usr/local/spark/python/lib/pyspark.zip/pyspark/serializers.py", line 211, in dump\_stream

self.serializer.dump\_stream(self.\_batched(iterator), stream)

File "/usr/local/spark/python/lib/pyspark.zip/pyspark/serializers.py", line 132, in dump\_stream

for obj in iterator:

File "/usr/local/spark/python/lib/pyspark.zip/pyspark/serializers.py", line 200, in \_batched

for item in iterator:

File "/usr/local/spark/python/lib/pyspark.zip/pyspark/worker.py", line 452, in mapper

result = tuple(f(\*[a[o] for o in arg\_offsets]) for (arg\_offsets, f) in udfs)

File "/usr/local/spark/python/lib/pyspark.zip/pyspark/worker.py", line 452, in <genexpr>

result = tuple(f(\*[a[o] for o in arg\_offsets]) for (arg\_offsets, f) in udfs)

File "/usr/local/spark/python/lib/pyspark.zip/pyspark/worker.py", line 87, in <lambda>

return lambda \*a: f(\*a)

File "/usr/local/spark/python/lib/pyspark.zip/pyspark/util.py", line 74, in wrapper

return f(\*args, \*\*kwargs)

File "<stdin>", line 3, in clean\_string

TypeError: expected string or bytes-like object

at org.apache.spark.api.python.BasePythonRunner$ReaderIterator.handlePythonException(PythonRunner.scala:555)

at org.apache.spark.sql.execution.python.PythonUDFRunner$$anon$2.read(PythonUDFRunner.scala:86)

at org.apache.spark.sql.execution.python.PythonUDFRunner$$anon$2.read(PythonUDFRunner.scala:68)

at org.apache.spark.api.python.BasePythonRunner$ReaderIterator.hasNext(PythonRunner.scala:508)

at org.apache.spark.InterruptibleIterator.hasNext(InterruptibleIterator.scala:37)

at scala.collection.Iterator$$anon$11.hasNext(Iterator.scala:491)

at scala.collection.Iterator$$anon$10.hasNext(Iterator.scala:460)

at scala.collection.Iterator$$anon$10.hasNext(Iterator.scala:460)

at org.apache.spark.sql.catalyst.expressions.GeneratedClass$GeneratedIteratorForCodegenStage1.processNext(Unknown Source)

at org.apache.spark.sql.execution.BufferedRowIterator.hasNext(BufferedRowIterator.java:43)

at org.apache.spark.sql.execution.WholeStageCodegenExec$$anon$1.hasNext(WholeStageCodegenExec.scala:759)

at org.apache.spark.sql.execution.columnar.DefaultCachedBatchSerializer$$anon$1.next(InMemoryRelation.scala:87)

at org.apache.spark.sql.execution.columnar.DefaultCachedBatchSerializer$$anon$1.next(InMemoryRelation.scala:79)

at scala.collection.Iterator$$anon$10.next(Iterator.scala:461)

at org.apache.spark.storage.memory.MemoryStore.putIterator(MemoryStore.scala:224)

at org.apache.spark.storage.memory.MemoryStore.putIteratorAsValues(MemoryStore.scala:302)

at org.apache.spark.storage.BlockManager.$anonfun$doPutIterator$1(BlockManager.scala:1481)

at org.apache.spark.storage.BlockManager.org$apache$spark$storage$BlockManager$$doPut(BlockManager.scala:1408)

at org.apache.spark.storage.BlockManager.doPutIterator(BlockManager.scala:1472)

at org.apache.spark.storage.BlockManager.getOrElseUpdate(BlockManager.scala:1295)

at org.apache.spark.rdd.RDD.getOrCompute(RDD.scala:384)

at org.apache.spark.rdd.RDD.iterator(RDD.scala:335)

at org.apache.spark.rdd.MapPartitionsRDD.compute(MapPartitionsRDD.scala:52)

at org.apache.spark.rdd.RDD.computeOrReadCheckpoint(RDD.scala:373)

at org.apache.spark.rdd.RDD.iterator(RDD.scala:337)

at org.apache.spark.rdd.MapPartitionsRDD.compute(MapPartitionsRDD.scala:52)

at org.apache.spark.rdd.RDD.computeOrReadCheckpoint(RDD.scala:373)

at org.apache.spark.rdd.RDD.iterator(RDD.scala:337)

at org.apache.spark.rdd.MapPartitionsRDD.compute(MapPartitionsRDD.scala:52)

at org.apache.spark.rdd.RDD.computeOrReadCheckpoint(RDD.scala:373)

at org.apache.spark.rdd.RDD.iterator(RDD.scala:337)

at org.apache.spark.scheduler.ResultTask.runTask(ResultTask.scala:90)

at org.apache.spark.scheduler.Task.run(Task.scala:131)

at org.apache.spark.executor.Executor$TaskRunner.$anonfun$run$3(Executor.scala:506)

at org.apache.spark.util.Utils$.tryWithSafeFinally(Utils.scala:1462)

at org.apache.spark.executor.Executor$TaskRunner.run(Executor.scala:509)

at java.util.concurrent.ThreadPoolExecutor.runWorker(ThreadPoolExecutor.java:1149)

at java.util.concurrent.ThreadPoolExecutor$Worker.run(ThreadPoolExecutor.java:624)

... 1 more

>>> sample\_op.write.json("hdfs:///user/hduser/Hackathon/JSON\_DATA", mode="overwrite")

24/07/22 09:25:20 WARN storage.BlockManager: Putting block rdd\_531\_0 failed due to exception org.apache.spark.api.python.PythonException: Traceback (most recent call last):

File "/usr/local/spark/python/lib/pyspark.zip/pyspark/worker.py", line 619, in main

process()

File "/usr/local/spark/python/lib/pyspark.zip/pyspark/worker.py", line 611, in process

serializer.dump\_stream(out\_iter, outfile)

File "/usr/local/spark/python/lib/pyspark.zip/pyspark/serializers.py", line 211, in dump\_stream

self.serializer.dump\_stream(self.\_batched(iterator), stream)

File "/usr/local/spark/python/lib/pyspark.zip/pyspark/serializers.py", line 132, in dump\_stream

for obj in iterator:

File "/usr/local/spark/python/lib/pyspark.zip/pyspark/serializers.py", line 200, in \_batched

for item in iterator:

File "/usr/local/spark/python/lib/pyspark.zip/pyspark/worker.py", line 452, in mapper

result = tuple(f(\*[a[o] for o in arg\_offsets]) for (arg\_offsets, f) in udfs)

File "/usr/local/spark/python/lib/pyspark.zip/pyspark/worker.py", line 452, in <genexpr>

result = tuple(f(\*[a[o] for o in arg\_offsets]) for (arg\_offsets, f) in udfs)

File "/usr/local/spark/python/lib/pyspark.zip/pyspark/worker.py", line 87, in <lambda>

return lambda \*a: f(\*a)

File "/usr/local/spark/python/lib/pyspark.zip/pyspark/util.py", line 74, in wrapper

return f(\*args, \*\*kwargs)

File "<stdin>", line 3, in clean\_string

TypeError: expected string or bytes-like object

.

24/07/22 09:25:20 WARN storage.BlockManager: Block rdd\_531\_0 could not be removed as it was not found on disk or in memory

24/07/22 09:25:20 ERROR executor.Executor: Exception in task 0.0 in stage 101.0 (TID 109)

org.apache.spark.api.python.PythonException: Traceback (most recent call last):

File "/usr/local/spark/python/lib/pyspark.zip/pyspark/worker.py", line 619, in main

process()

File "/usr/local/spark/python/lib/pyspark.zip/pyspark/worker.py", line 611, in process

serializer.dump\_stream(out\_iter, outfile)

File "/usr/local/spark/python/lib/pyspark.zip/pyspark/serializers.py", line 211, in dump\_stream

self.serializer.dump\_stream(self.\_batched(iterator), stream)

File "/usr/local/spark/python/lib/pyspark.zip/pyspark/serializers.py", line 132, in dump\_stream

for obj in iterator:

File "/usr/local/spark/python/lib/pyspark.zip/pyspark/serializers.py", line 200, in \_batched

for item in iterator:

File "/usr/local/spark/python/lib/pyspark.zip/pyspark/worker.py", line 452, in mapper

result = tuple(f(\*[a[o] for o in arg\_offsets]) for (arg\_offsets, f) in udfs)

File "/usr/local/spark/python/lib/pyspark.zip/pyspark/worker.py", line 452, in <genexpr>

result = tuple(f(\*[a[o] for o in arg\_offsets]) for (arg\_offsets, f) in udfs)

File "/usr/local/spark/python/lib/pyspark.zip/pyspark/worker.py", line 87, in <lambda>

return lambda \*a: f(\*a)

File "/usr/local/spark/python/lib/pyspark.zip/pyspark/util.py", line 74, in wrapper

return f(\*args, \*\*kwargs)

File "<stdin>", line 3, in clean\_string

TypeError: expected string or bytes-like object

at org.apache.spark.api.python.BasePythonRunner$ReaderIterator.handlePythonException(PythonRunner.scala:555)

at org.apache.spark.sql.execution.python.PythonUDFRunner$$anon$2.read(PythonUDFRunner.scala:86)

at org.apache.spark.sql.execution.python.PythonUDFRunner$$anon$2.read(PythonUDFRunner.scala:68)

at org.apache.spark.api.python.BasePythonRunner$ReaderIterator.hasNext(PythonRunner.scala:508)

at org.apache.spark.InterruptibleIterator.hasNext(InterruptibleIterator.scala:37)

at scala.collection.Iterator$$anon$11.hasNext(Iterator.scala:491)

at scala.collection.Iterator$$anon$10.hasNext(Iterator.scala:460)

at scala.collection.Iterator$$anon$10.hasNext(Iterator.scala:460)

at org.apache.spark.sql.catalyst.expressions.GeneratedClass$GeneratedIteratorForCodegenStage1.processNext(Unknown Source)

at org.apache.spark.sql.execution.BufferedRowIterator.hasNext(BufferedRowIterator.java:43)

at org.apache.spark.sql.execution.WholeStageCodegenExec$$anon$1.hasNext(WholeStageCodegenExec.scala:759)

at org.apache.spark.sql.execution.columnar.DefaultCachedBatchSerializer$$anon$1.next(InMemoryRelation.scala:87)

at org.apache.spark.sql.execution.columnar.DefaultCachedBatchSerializer$$anon$1.next(InMemoryRelation.scala:79)

at scala.collection.Iterator$$anon$10.next(Iterator.scala:461)

at org.apache.spark.storage.memory.MemoryStore.putIterator(MemoryStore.scala:224)

at org.apache.spark.storage.memory.MemoryStore.putIteratorAsValues(MemoryStore.scala:302)

at org.apache.spark.storage.BlockManager.$anonfun$doPutIterator$1(BlockManager.scala:1481)

at org.apache.spark.storage.BlockManager.org$apache$spark$storage$BlockManager$$doPut(BlockManager.scala:1408)

at org.apache.spark.storage.BlockManager.doPutIterator(BlockManager.scala:1472)

at org.apache.spark.storage.BlockManager.getOrElseUpdate(BlockManager.scala:1295)

at org.apache.spark.rdd.RDD.getOrCompute(RDD.scala:384)

at org.apache.spark.rdd.RDD.iterator(RDD.scala:335)

at org.apache.spark.rdd.MapPartitionsRDD.compute(MapPartitionsRDD.scala:52)

at org.apache.spark.rdd.RDD.computeOrReadCheckpoint(RDD.scala:373)

at org.apache.spark.rdd.RDD.iterator(RDD.scala:337)

at org.apache.spark.rdd.MapPartitionsRDD.compute(MapPartitionsRDD.scala:52)

at org.apache.spark.rdd.RDD.computeOrReadCheckpoint(RDD.scala:373)

at org.apache.spark.rdd.RDD.iterator(RDD.scala:337)

at org.apache.spark.rdd.MapPartitionsRDD.compute(MapPartitionsRDD.scala:52)

at org.apache.spark.rdd.RDD.computeOrReadCheckpoint(RDD.scala:373)

at org.apache.spark.rdd.RDD.iterator(RDD.scala:337)

at org.apache.spark.scheduler.ResultTask.runTask(ResultTask.scala:90)

at org.apache.spark.scheduler.Task.run(Task.scala:131)

at org.apache.spark.executor.Executor$TaskRunner.$anonfun$run$3(Executor.scala:506)

at org.apache.spark.util.Utils$.tryWithSafeFinally(Utils.scala:1462)

at org.apache.spark.executor.Executor$TaskRunner.run(Executor.scala:509)

at java.util.concurrent.ThreadPoolExecutor.runWorker(ThreadPoolExecutor.java:1149)

at java.util.concurrent.ThreadPoolExecutor$Worker.run(ThreadPoolExecutor.java:624)

at java.lang.Thread.run(Thread.java:750)

24/07/22 09:25:20 WARN scheduler.TaskSetManager: Lost task 0.0 in stage 101.0 (TID 109) (192.168.112.129 executor driver): org.apache.spark.api.python.PythonException: Traceback (most recent call last):

File "/usr/local/spark/python/lib/pyspark.zip/pyspark/worker.py", line 619, in main

process()

File "/usr/local/spark/python/lib/pyspark.zip/pyspark/worker.py", line 611, in process

serializer.dump\_stream(out\_iter, outfile)

File "/usr/local/spark/python/lib/pyspark.zip/pyspark/serializers.py", line 211, in dump\_stream

self.serializer.dump\_stream(self.\_batched(iterator), stream)

File "/usr/local/spark/python/lib/pyspark.zip/pyspark/serializers.py", line 132, in dump\_stream

for obj in iterator:

File "/usr/local/spark/python/lib/pyspark.zip/pyspark/serializers.py", line 200, in \_batched

for item in iterator:

File "/usr/local/spark/python/lib/pyspark.zip/pyspark/worker.py", line 452, in mapper

result = tuple(f(\*[a[o] for o in arg\_offsets]) for (arg\_offsets, f) in udfs)

File "/usr/local/spark/python/lib/pyspark.zip/pyspark/worker.py", line 452, in <genexpr>

result = tuple(f(\*[a[o] for o in arg\_offsets]) for (arg\_offsets, f) in udfs)

File "/usr/local/spark/python/lib/pyspark.zip/pyspark/worker.py", line 87, in <lambda>

return lambda \*a: f(\*a)

File "/usr/local/spark/python/lib/pyspark.zip/pyspark/util.py", line 74, in wrapper

return f(\*args, \*\*kwargs)

File "<stdin>", line 3, in clean\_string

TypeError: expected string or bytes-like object

at org.apache.spark.api.python.BasePythonRunner$ReaderIterator.handlePythonException(PythonRunner.scala:555)

at org.apache.spark.sql.execution.python.PythonUDFRunner$$anon$2.read(PythonUDFRunner.scala:86)

at org.apache.spark.sql.execution.python.PythonUDFRunner$$anon$2.read(PythonUDFRunner.scala:68)

at org.apache.spark.api.python.BasePythonRunner$ReaderIterator.hasNext(PythonRunner.scala:508)

at org.apache.spark.InterruptibleIterator.hasNext(InterruptibleIterator.scala:37)

at scala.collection.Iterator$$anon$11.hasNext(Iterator.scala:491)

at scala.collection.Iterator$$anon$10.hasNext(Iterator.scala:460)

at scala.collection.Iterator$$anon$10.hasNext(Iterator.scala:460)

at org.apache.spark.sql.catalyst.expressions.GeneratedClass$GeneratedIteratorForCodegenStage1.processNext(Unknown Source)

at org.apache.spark.sql.execution.BufferedRowIterator.hasNext(BufferedRowIterator.java:43)

at org.apache.spark.sql.execution.WholeStageCodegenExec$$anon$1.hasNext(WholeStageCodegenExec.scala:759)

at org.apache.spark.sql.execution.columnar.DefaultCachedBatchSerializer$$anon$1.next(InMemoryRelation.scala:87)

at org.apache.spark.sql.execution.columnar.DefaultCachedBatchSerializer$$anon$1.next(InMemoryRelation.scala:79)

at scala.collection.Iterator$$anon$10.next(Iterator.scala:461)

at org.apache.spark.storage.memory.MemoryStore.putIterator(MemoryStore.scala:224)

at org.apache.spark.storage.memory.MemoryStore.putIteratorAsValues(MemoryStore.scala:302)

at org.apache.spark.storage.BlockManager.$anonfun$doPutIterator$1(BlockManager.scala:1481)

at org.apache.spark.storage.BlockManager.org$apache$spark$storage$BlockManager$$doPut(BlockManager.scala:1408)

at org.apache.spark.storage.BlockManager.doPutIterator(BlockManager.scala:1472)

at org.apache.spark.storage.BlockManager.getOrElseUpdate(BlockManager.scala:1295)

at org.apache.spark.rdd.RDD.getOrCompute(RDD.scala:384)

at org.apache.spark.rdd.RDD.iterator(RDD.scala:335)

at org.apache.spark.rdd.MapPartitionsRDD.compute(MapPartitionsRDD.scala:52)

at org.apache.spark.rdd.RDD.computeOrReadCheckpoint(RDD.scala:373)

at org.apache.spark.rdd.RDD.iterator(RDD.scala:337)

at org.apache.spark.rdd.MapPartitionsRDD.compute(MapPartitionsRDD.scala:52)

at org.apache.spark.rdd.RDD.computeOrReadCheckpoint(RDD.scala:373)

at org.apache.spark.rdd.RDD.iterator(RDD.scala:337)

at org.apache.spark.rdd.MapPartitionsRDD.compute(MapPartitionsRDD.scala:52)

at org.apache.spark.rdd.RDD.computeOrReadCheckpoint(RDD.scala:373)

at org.apache.spark.rdd.RDD.iterator(RDD.scala:337)

at org.apache.spark.scheduler.ResultTask.runTask(ResultTask.scala:90)

at org.apache.spark.scheduler.Task.run(Task.scala:131)

at org.apache.spark.executor.Executor$TaskRunner.$anonfun$run$3(Executor.scala:506)

at org.apache.spark.util.Utils$.tryWithSafeFinally(Utils.scala:1462)

at org.apache.spark.executor.Executor$TaskRunner.run(Executor.scala:509)

at java.util.concurrent.ThreadPoolExecutor.runWorker(ThreadPoolExecutor.java:1149)

at java.util.concurrent.ThreadPoolExecutor$Worker.run(ThreadPoolExecutor.java:624)

at java.lang.Thread.run(Thread.java:750)

24/07/22 09:25:20 ERROR scheduler.TaskSetManager: Task 0 in stage 101.0 failed 1 times; aborting job

24/07/22 09:25:20 ERROR datasources.FileFormatWriter: Aborting job cd2d6f19-2fcd-4058-9b2f-cd83b6f36bb2.

org.apache.spark.SparkException: Job aborted due to stage failure: Task 0 in stage 101.0 failed 1 times, most recent failure: Lost task 0.0 in stage 101.0 (TID 109) (192.168.112.129 executor driver): org.apache.spark.api.python.PythonException: Traceback (most recent call last):

File "/usr/local/spark/python/lib/pyspark.zip/pyspark/worker.py", line 619, in main

process()

File "/usr/local/spark/python/lib/pyspark.zip/pyspark/worker.py", line 611, in process

serializer.dump\_stream(out\_iter, outfile)

File "/usr/local/spark/python/lib/pyspark.zip/pyspark/serializers.py", line 211, in dump\_stream

self.serializer.dump\_stream(self.\_batched(iterator), stream)

File "/usr/local/spark/python/lib/pyspark.zip/pyspark/serializers.py", line 132, in dump\_stream

for obj in iterator:

File "/usr/local/spark/python/lib/pyspark.zip/pyspark/serializers.py", line 200, in \_batched

for item in iterator:

File "/usr/local/spark/python/lib/pyspark.zip/pyspark/worker.py", line 452, in mapper

result = tuple(f(\*[a[o] for o in arg\_offsets]) for (arg\_offsets, f) in udfs)

File "/usr/local/spark/python/lib/pyspark.zip/pyspark/worker.py", line 452, in <genexpr>

result = tuple(f(\*[a[o] for o in arg\_offsets]) for (arg\_offsets, f) in udfs)

File "/usr/local/spark/python/lib/pyspark.zip/pyspark/worker.py", line 87, in <lambda>

return lambda \*a: f(\*a)

File "/usr/local/spark/python/lib/pyspark.zip/pyspark/util.py", line 74, in wrapper

return f(\*args, \*\*kwargs)

File "<stdin>", line 3, in clean\_string

TypeError: expected string or bytes-like object

at org.apache.spark.api.python.BasePythonRunner$ReaderIterator.handlePythonException(PythonRunner.scala:555)

at org.apache.spark.sql.execution.python.PythonUDFRunner$$anon$2.read(PythonUDFRunner.scala:86)

at org.apache.spark.sql.execution.python.PythonUDFRunner$$anon$2.read(PythonUDFRunner.scala:68)

at org.apache.spark.api.python.BasePythonRunner$ReaderIterator.hasNext(PythonRunner.scala:508)

at org.apache.spark.InterruptibleIterator.hasNext(InterruptibleIterator.scala:37)

at scala.collection.Iterator$$anon$11.hasNext(Iterator.scala:491)

at scala.collection.Iterator$$anon$10.hasNext(Iterator.scala:460)

at scala.collection.Iterator$$anon$10.hasNext(Iterator.scala:460)

at org.apache.spark.sql.catalyst.expressions.GeneratedClass$GeneratedIteratorForCodegenStage1.processNext(Unknown Source)

at org.apache.spark.sql.execution.BufferedRowIterator.hasNext(BufferedRowIterator.java:43)

at org.apache.spark.sql.execution.WholeStageCodegenExec$$anon$1.hasNext(WholeStageCodegenExec.scala:759)

at org.apache.spark.sql.execution.columnar.DefaultCachedBatchSerializer$$anon$1.next(InMemoryRelation.scala:87)

at org.apache.spark.sql.execution.columnar.DefaultCachedBatchSerializer$$anon$1.next(InMemoryRelation.scala:79)

at scala.collection.Iterator$$anon$10.next(Iterator.scala:461)

at org.apache.spark.storage.memory.MemoryStore.putIterator(MemoryStore.scala:224)

at org.apache.spark.storage.memory.MemoryStore.putIteratorAsValues(MemoryStore.scala:302)

at org.apache.spark.storage.BlockManager.$anonfun$doPutIterator$1(BlockManager.scala:1481)

at org.apache.spark.storage.BlockManager.org$apache$spark$storage$BlockManager$$doPut(BlockManager.scala:1408)

at org.apache.spark.storage.BlockManager.doPutIterator(BlockManager.scala:1472)

at org.apache.spark.storage.BlockManager.getOrElseUpdate(BlockManager.scala:1295)

at org.apache.spark.rdd.RDD.getOrCompute(RDD.scala:384)

at org.apache.spark.rdd.RDD.iterator(RDD.scala:335)

at org.apache.spark.rdd.MapPartitionsRDD.compute(MapPartitionsRDD.scala:52)

at org.apache.spark.rdd.RDD.computeOrReadCheckpoint(RDD.scala:373)

at org.apache.spark.rdd.RDD.iterator(RDD.scala:337)

at org.apache.spark.rdd.MapPartitionsRDD.compute(MapPartitionsRDD.scala:52)

at org.apache.spark.rdd.RDD.computeOrReadCheckpoint(RDD.scala:373)

at org.apache.spark.rdd.RDD.iterator(RDD.scala:337)

at org.apache.spark.rdd.MapPartitionsRDD.compute(MapPartitionsRDD.scala:52)

at org.apache.spark.rdd.RDD.computeOrReadCheckpoint(RDD.scala:373)

at org.apache.spark.rdd.RDD.iterator(RDD.scala:337)

at org.apache.spark.scheduler.ResultTask.runTask(ResultTask.scala:90)

at org.apache.spark.scheduler.Task.run(Task.scala:131)

at org.apache.spark.executor.Executor$TaskRunner.$anonfun$run$3(Executor.scala:506)

at org.apache.spark.util.Utils$.tryWithSafeFinally(Utils.scala:1462)

at org.apache.spark.executor.Executor$TaskRunner.run(Executor.scala:509)

at java.util.concurrent.ThreadPoolExecutor.runWorker(ThreadPoolExecutor.java:1149)

at java.util.concurrent.ThreadPoolExecutor$Worker.run(ThreadPoolExecutor.java:624)

at java.lang.Thread.run(Thread.java:750)

Driver stacktrace:

at org.apache.spark.scheduler.DAGScheduler.failJobAndIndependentStages(DAGScheduler.scala:2454)

at org.apache.spark.scheduler.DAGScheduler.$anonfun$abortStage$2(DAGScheduler.scala:2403)

at org.apache.spark.scheduler.DAGScheduler.$anonfun$abortStage$2$adapted(DAGScheduler.scala:2402)

at scala.collection.mutable.ResizableArray.foreach(ResizableArray.scala:62)

at scala.collection.mutable.ResizableArray.foreach$(ResizableArray.scala:55)

at scala.collection.mutable.ArrayBuffer.foreach(ArrayBuffer.scala:49)

at org.apache.spark.scheduler.DAGScheduler.abortStage(DAGScheduler.scala:2402)

at org.apache.spark.scheduler.DAGScheduler.$anonfun$handleTaskSetFailed$1(DAGScheduler.scala:1160)

at org.apache.spark.scheduler.DAGScheduler.$anonfun$handleTaskSetFailed$1$adapted(DAGScheduler.scala:1160)

at scala.Option.foreach(Option.scala:407)

at org.apache.spark.scheduler.DAGScheduler.handleTaskSetFailed(DAGScheduler.scala:1160)

at org.apache.spark.scheduler.DAGSchedulerEventProcessLoop.doOnReceive(DAGScheduler.scala:2642)

at org.apache.spark.scheduler.DAGSchedulerEventProcessLoop.onReceive(DAGScheduler.scala:2584)

at org.apache.spark.scheduler.DAGSchedulerEventProcessLoop.onReceive(DAGScheduler.scala:2573)

at org.apache.spark.util.EventLoop$$anon$1.run(EventLoop.scala:49)

at org.apache.spark.scheduler.DAGScheduler.runJob(DAGScheduler.scala:938)

at org.apache.spark.SparkContext.runJob(SparkContext.scala:2214)

at org.apache.spark.sql.execution.datasources.FileFormatWriter$.write(FileFormatWriter.scala:218)

at org.apache.spark.sql.execution.datasources.InsertIntoHadoopFsRelationCommand.run(InsertIntoHadoopFsRelationCommand.scala:186)

at org.apache.spark.sql.execution.command.DataWritingCommandExec.sideEffectResult$lzycompute(commands.scala:113)

at org.apache.spark.sql.execution.command.DataWritingCommandExec.sideEffectResult(commands.scala:111)

at org.apache.spark.sql.execution.command.DataWritingCommandExec.executeCollect(commands.scala:125)

at org.apache.spark.sql.execution.QueryExecution$$anonfun$eagerlyExecuteCommands$1.$anonfun$applyOrElse$1(QueryExecution.scala:110)

at org.apache.spark.sql.execution.SQLExecution$.$anonfun$withNewExecutionId$5(SQLExecution.scala:103)

at org.apache.spark.sql.execution.SQLExecution$.withSQLConfPropagated(SQLExecution.scala:163)

at org.apache.spark.sql.execution.SQLExecution$.$anonfun$withNewExecutionId$1(SQLExecution.scala:90)

at org.apache.spark.sql.SparkSession.withActive(SparkSession.scala:775)

at org.apache.spark.sql.execution.SQLExecution$.withNewExecutionId(SQLExecution.scala:64)

at org.apache.spark.sql.execution.QueryExecution$$anonfun$eagerlyExecuteCommands$1.applyOrElse(QueryExecution.scala:110)

at org.apache.spark.sql.execution.QueryExecution$$anonfun$eagerlyExecuteCommands$1.applyOrElse(QueryExecution.scala:106)

at org.apache.spark.sql.catalyst.trees.TreeNode.$anonfun$transformDownWithPruning$1(TreeNode.scala:481)

at org.apache.spark.sql.catalyst.trees.CurrentOrigin$.withOrigin(TreeNode.scala:82)

at org.apache.spark.sql.catalyst.trees.TreeNode.transformDownWithPruning(TreeNode.scala:481)

at org.apache.spark.sql.catalyst.plans.logical.LogicalPlan.org$apache$spark$sql$catalyst$plans$logical$AnalysisHelper$$super$transformDownWithPruning(LogicalPlan.scala:30)

at org.apache.spark.sql.catalyst.plans.logical.AnalysisHelper.transformDownWithPruning(AnalysisHelper.scala:267)

at org.apache.spark.sql.catalyst.plans.logical.AnalysisHelper.transformDownWithPruning$(AnalysisHelper.scala:263)

at org.apache.spark.sql.catalyst.plans.logical.LogicalPlan.transformDownWithPruning(LogicalPlan.scala:30)

at org.apache.spark.sql.catalyst.plans.logical.LogicalPlan.transformDownWithPruning(LogicalPlan.scala:30)

at org.apache.spark.sql.catalyst.trees.TreeNode.transformDown(TreeNode.scala:457)

at org.apache.spark.sql.execution.QueryExecution.eagerlyExecuteCommands(QueryExecution.scala:106)

at org.apache.spark.sql.execution.QueryExecution.commandExecuted$lzycompute(QueryExecution.scala:93)

at org.apache.spark.sql.execution.QueryExecution.commandExecuted(QueryExecution.scala:91)

at org.apache.spark.sql.execution.QueryExecution.assertCommandExecuted(QueryExecution.scala:128)

at org.apache.spark.sql.DataFrameWriter.runCommand(DataFrameWriter.scala:848)

at org.apache.spark.sql.DataFrameWriter.saveToV1Source(DataFrameWriter.scala:382)

at org.apache.spark.sql.DataFrameWriter.saveInternal(DataFrameWriter.scala:355)

at org.apache.spark.sql.DataFrameWriter.save(DataFrameWriter.scala:239)

at org.apache.spark.sql.DataFrameWriter.json(DataFrameWriter.scala:763)

at sun.reflect.NativeMethodAccessorImpl.invoke0(Native Method)

at sun.reflect.NativeMethodAccessorImpl.invoke(NativeMethodAccessorImpl.java:62)

at sun.reflect.DelegatingMethodAccessorImpl.invoke(DelegatingMethodAccessorImpl.java:43)

at java.lang.reflect.Method.invoke(Method.java:498)

at py4j.reflection.MethodInvoker.invoke(MethodInvoker.java:244)

at py4j.reflection.ReflectionEngine.invoke(ReflectionEngine.java:357)

at py4j.Gateway.invoke(Gateway.java:282)

at py4j.commands.AbstractCommand.invokeMethod(AbstractCommand.java:132)

at py4j.commands.CallCommand.execute(CallCommand.java:79)

at py4j.ClientServerConnection.waitForCommands(ClientServerConnection.java:182)

at py4j.ClientServerConnection.run(ClientServerConnection.java:106)

at java.lang.Thread.run(Thread.java:750)

Caused by: org.apache.spark.api.python.PythonException: Traceback (most recent call last):

File "/usr/local/spark/python/lib/pyspark.zip/pyspark/worker.py", line 619, in main

process()

File "/usr/local/spark/python/lib/pyspark.zip/pyspark/worker.py", line 611, in process

serializer.dump\_stream(out\_iter, outfile)

File "/usr/local/spark/python/lib/pyspark.zip/pyspark/serializers.py", line 211, in dump\_stream

self.serializer.dump\_stream(self.\_batched(iterator), stream)

File "/usr/local/spark/python/lib/pyspark.zip/pyspark/serializers.py", line 132, in dump\_stream

for obj in iterator:

File "/usr/local/spark/python/lib/pyspark.zip/pyspark/serializers.py", line 200, in \_batched

for item in iterator:

File "/usr/local/spark/python/lib/pyspark.zip/pyspark/worker.py", line 452, in mapper

result = tuple(f(\*[a[o] for o in arg\_offsets]) for (arg\_offsets, f) in udfs)

File "/usr/local/spark/python/lib/pyspark.zip/pyspark/worker.py", line 452, in <genexpr>

result = tuple(f(\*[a[o] for o in arg\_offsets]) for (arg\_offsets, f) in udfs)

File "/usr/local/spark/python/lib/pyspark.zip/pyspark/worker.py", line 87, in <lambda>

return lambda \*a: f(\*a)

File "/usr/local/spark/python/lib/pyspark.zip/pyspark/util.py", line 74, in wrapper

return f(\*args, \*\*kwargs)

File "<stdin>", line 3, in clean\_string

TypeError: expected string or bytes-like object

at org.apache.spark.api.python.BasePythonRunner$ReaderIterator.handlePythonException(PythonRunner.scala:555)

at org.apache.spark.sql.execution.python.PythonUDFRunner$$anon$2.read(PythonUDFRunner.scala:86)

at org.apache.spark.sql.execution.python.PythonUDFRunner$$anon$2.read(PythonUDFRunner.scala:68)

at org.apache.spark.api.python.BasePythonRunner$ReaderIterator.hasNext(PythonRunner.scala:508)

at org.apache.spark.InterruptibleIterator.hasNext(InterruptibleIterator.scala:37)

at scala.collection.Iterator$$anon$11.hasNext(Iterator.scala:491)

at scala.collection.Iterator$$anon$10.hasNext(Iterator.scala:460)

at scala.collection.Iterator$$anon$10.hasNext(Iterator.scala:460)

at org.apache.spark.sql.catalyst.expressions.GeneratedClass$GeneratedIteratorForCodegenStage1.processNext(Unknown Source)

at org.apache.spark.sql.execution.BufferedRowIterator.hasNext(BufferedRowIterator.java:43)

at org.apache.spark.sql.execution.WholeStageCodegenExec$$anon$1.hasNext(WholeStageCodegenExec.scala:759)

at org.apache.spark.sql.execution.columnar.DefaultCachedBatchSerializer$$anon$1.next(InMemoryRelation.scala:87)

at org.apache.spark.sql.execution.columnar.DefaultCachedBatchSerializer$$anon$1.next(InMemoryRelation.scala:79)

at scala.collection.Iterator$$anon$10.next(Iterator.scala:461)

at org.apache.spark.storage.memory.MemoryStore.putIterator(MemoryStore.scala:224)

at org.apache.spark.storage.memory.MemoryStore.putIteratorAsValues(MemoryStore.scala:302)

at org.apache.spark.storage.BlockManager.$anonfun$doPutIterator$1(BlockManager.scala:1481)

at org.apache.spark.storage.BlockManager.org$apache$spark$storage$BlockManager$$doPut(BlockManager.scala:1408)

at org.apache.spark.storage.BlockManager.doPutIterator(BlockManager.scala:1472)

at org.apache.spark.storage.BlockManager.getOrElseUpdate(BlockManager.scala:1295)

at org.apache.spark.rdd.RDD.getOrCompute(RDD.scala:384)

at org.apache.spark.rdd.RDD.iterator(RDD.scala:335)

at org.apache.spark.rdd.MapPartitionsRDD.compute(MapPartitionsRDD.scala:52)

at org.apache.spark.rdd.RDD.computeOrReadCheckpoint(RDD.scala:373)

at org.apache.spark.rdd.RDD.iterator(RDD.scala:337)

at org.apache.spark.rdd.MapPartitionsRDD.compute(MapPartitionsRDD.scala:52)

at org.apache.spark.rdd.RDD.computeOrReadCheckpoint(RDD.scala:373)

at org.apache.spark.rdd.RDD.iterator(RDD.scala:337)

at org.apache.spark.rdd.MapPartitionsRDD.compute(MapPartitionsRDD.scala:52)

at org.apache.spark.rdd.RDD.computeOrReadCheckpoint(RDD.scala:373)

at org.apache.spark.rdd.RDD.iterator(RDD.scala:337)

at org.apache.spark.scheduler.ResultTask.runTask(ResultTask.scala:90)

at org.apache.spark.scheduler.Task.run(Task.scala:131)

at org.apache.spark.executor.Executor$TaskRunner.$anonfun$run$3(Executor.scala:506)

at org.apache.spark.util.Utils$.tryWithSafeFinally(Utils.scala:1462)

at org.apache.spark.executor.Executor$TaskRunner.run(Executor.scala:509)

at java.util.concurrent.ThreadPoolExecutor.runWorker(ThreadPoolExecutor.java:1149)

at java.util.concurrent.ThreadPoolExecutor$Worker.run(ThreadPoolExecutor.java:624)

... 1 more

24/07/22 09:25:20 WARN storage.BlockManager: Putting block rdd\_531\_1 failed due to exception org.apache.spark.TaskKilledException.

24/07/22 09:25:20 WARN storage.BlockManager: Block rdd\_531\_1 could not be removed as it was not found on disk or in memory

24/07/22 09:25:20 WARN scheduler.TaskSetManager: Lost task 1.0 in stage 101.0 (TID 110) (192.168.112.129 executor driver): TaskKilled (Stage cancelled)

Traceback (most recent call last):

File "<stdin>", line 1, in <module>

File "/usr/local/spark/python/pyspark/sql/readwriter.py", line 846, in json

self.\_jwrite.json(path)

File "/usr/local/spark/python/lib/py4j-0.10.9.3-src.zip/py4j/java\_gateway.py", line 1322, in \_\_call\_\_

File "/usr/local/spark/python/pyspark/sql/utils.py", line 111, in deco

return f(\*a, \*\*kw)

File "/usr/local/spark/python/lib/py4j-0.10.9.3-src.zip/py4j/protocol.py", line 328, in get\_return\_value

py4j.protocol.Py4JJavaError: An error occurred while calling o620.json.

: org.apache.spark.SparkException: Job aborted.

at org.apache.spark.sql.errors.QueryExecutionErrors$.jobAbortedError(QueryExecutionErrors.scala:496)

at org.apache.spark.sql.execution.datasources.FileFormatWriter$.write(FileFormatWriter.scala:251)

at org.apache.spark.sql.execution.datasources.InsertIntoHadoopFsRelationCommand.run(InsertIntoHadoopFsRelationCommand.scala:186)

at org.apache.spark.sql.execution.command.DataWritingCommandExec.sideEffectResult$lzycompute(commands.scala:113)

at org.apache.spark.sql.execution.command.DataWritingCommandExec.sideEffectResult(commands.scala:111)

at org.apache.spark.sql.execution.command.DataWritingCommandExec.executeCollect(commands.scala:125)

at org.apache.spark.sql.execution.QueryExecution$$anonfun$eagerlyExecuteCommands$1.$anonfun$applyOrElse$1(QueryExecution.scala:110)

at org.apache.spark.sql.execution.SQLExecution$.$anonfun$withNewExecutionId$5(SQLExecution.scala:103)

at org.apache.spark.sql.execution.SQLExecution$.withSQLConfPropagated(SQLExecution.scala:163)

at org.apache.spark.sql.execution.SQLExecution$.$anonfun$withNewExecutionId$1(SQLExecution.scala:90)

at org.apache.spark.sql.SparkSession.withActive(SparkSession.scala:775)

at org.apache.spark.sql.execution.SQLExecution$.withNewExecutionId(SQLExecution.scala:64)

at org.apache.spark.sql.execution.QueryExecution$$anonfun$eagerlyExecuteCommands$1.applyOrElse(QueryExecution.scala:110)

at org.apache.spark.sql.execution.QueryExecution$$anonfun$eagerlyExecuteCommands$1.applyOrElse(QueryExecution.scala:106)

at org.apache.spark.sql.catalyst.trees.TreeNode.$anonfun$transformDownWithPruning$1(TreeNode.scala:481)

at org.apache.spark.sql.catalyst.trees.CurrentOrigin$.withOrigin(TreeNode.scala:82)

at org.apache.spark.sql.catalyst.trees.TreeNode.transformDownWithPruning(TreeNode.scala:481)

at org.apache.spark.sql.catalyst.plans.logical.LogicalPlan.org$apache$spark$sql$catalyst$plans$logical$AnalysisHelper$$super$transformDownWithPruning(LogicalPlan.scala:30)

at org.apache.spark.sql.catalyst.plans.logical.AnalysisHelper.transformDownWithPruning(AnalysisHelper.scala:267)

at org.apache.spark.sql.catalyst.plans.logical.AnalysisHelper.transformDownWithPruning$(AnalysisHelper.scala:263)

at org.apache.spark.sql.catalyst.plans.logical.LogicalPlan.transformDownWithPruning(LogicalPlan.scala:30)

at org.apache.spark.sql.catalyst.plans.logical.LogicalPlan.transformDownWithPruning(LogicalPlan.scala:30)

at org.apache.spark.sql.catalyst.trees.TreeNode.transformDown(TreeNode.scala:457)

at org.apache.spark.sql.execution.QueryExecution.eagerlyExecuteCommands(QueryExecution.scala:106)

at org.apache.spark.sql.execution.QueryExecution.commandExecuted$lzycompute(QueryExecution.scala:93)

at org.apache.spark.sql.execution.QueryExecution.commandExecuted(QueryExecution.scala:91)

at org.apache.spark.sql.execution.QueryExecution.assertCommandExecuted(QueryExecution.scala:128)

at org.apache.spark.sql.DataFrameWriter.runCommand(DataFrameWriter.scala:848)

at org.apache.spark.sql.DataFrameWriter.saveToV1Source(DataFrameWriter.scala:382)

at org.apache.spark.sql.DataFrameWriter.saveInternal(DataFrameWriter.scala:355)

at org.apache.spark.sql.DataFrameWriter.save(DataFrameWriter.scala:239)

at org.apache.spark.sql.DataFrameWriter.json(DataFrameWriter.scala:763)

at sun.reflect.NativeMethodAccessorImpl.invoke0(Native Method)

at sun.reflect.NativeMethodAccessorImpl.invoke(NativeMethodAccessorImpl.java:62)

at sun.reflect.DelegatingMethodAccessorImpl.invoke(DelegatingMethodAccessorImpl.java:43)

at java.lang.reflect.Method.invoke(Method.java:498)

at py4j.reflection.MethodInvoker.invoke(MethodInvoker.java:244)

at py4j.reflection.ReflectionEngine.invoke(ReflectionEngine.java:357)

at py4j.Gateway.invoke(Gateway.java:282)

at py4j.commands.AbstractCommand.invokeMethod(AbstractCommand.java:132)

at py4j.commands.CallCommand.execute(CallCommand.java:79)

at py4j.ClientServerConnection.waitForCommands(ClientServerConnection.java:182)

at py4j.ClientServerConnection.run(ClientServerConnection.java:106)

at java.lang.Thread.run(Thread.java:750)

Caused by: org.apache.spark.SparkException: Job aborted due to stage failure: Task 0 in stage 101.0 failed 1 times, most recent failure: Lost task 0.0 in stage 101.0 (TID 109) (192.168.112.129 executor driver): org.apache.spark.api.python.PythonException: Traceback (most recent call last):

File "/usr/local/spark/python/lib/pyspark.zip/pyspark/worker.py", line 619, in main

process()

File "/usr/local/spark/python/lib/pyspark.zip/pyspark/worker.py", line 611, in process

serializer.dump\_stream(out\_iter, outfile)

File "/usr/local/spark/python/lib/pyspark.zip/pyspark/serializers.py", line 211, in dump\_stream

self.serializer.dump\_stream(self.\_batched(iterator), stream)

File "/usr/local/spark/python/lib/pyspark.zip/pyspark/serializers.py", line 132, in dump\_stream

for obj in iterator:

File "/usr/local/spark/python/lib/pyspark.zip/pyspark/serializers.py", line 200, in \_batched

for item in iterator:

File "/usr/local/spark/python/lib/pyspark.zip/pyspark/worker.py", line 452, in mapper

result = tuple(f(\*[a[o] for o in arg\_offsets]) for (arg\_offsets, f) in udfs)

File "/usr/local/spark/python/lib/pyspark.zip/pyspark/worker.py", line 452, in <genexpr>

result = tuple(f(\*[a[o] for o in arg\_offsets]) for (arg\_offsets, f) in udfs)

File "/usr/local/spark/python/lib/pyspark.zip/pyspark/worker.py", line 87, in <lambda>

return lambda \*a: f(\*a)

File "/usr/local/spark/python/lib/pyspark.zip/pyspark/util.py", line 74, in wrapper

return f(\*args, \*\*kwargs)

File "<stdin>", line 3, in clean\_string

TypeError: expected string or bytes-like object

at org.apache.spark.api.python.BasePythonRunner$ReaderIterator.handlePythonException(PythonRunner.scala:555)

at org.apache.spark.sql.execution.python.PythonUDFRunner$$anon$2.read(PythonUDFRunner.scala:86)

at org.apache.spark.sql.execution.python.PythonUDFRunner$$anon$2.read(PythonUDFRunner.scala:68)

at org.apache.spark.api.python.BasePythonRunner$ReaderIterator.hasNext(PythonRunner.scala:508)

at org.apache.spark.InterruptibleIterator.hasNext(InterruptibleIterator.scala:37)

at scala.collection.Iterator$$anon$11.hasNext(Iterator.scala:491)

at scala.collection.Iterator$$anon$10.hasNext(Iterator.scala:460)

at scala.collection.Iterator$$anon$10.hasNext(Iterator.scala:460)

at org.apache.spark.sql.catalyst.expressions.GeneratedClass$GeneratedIteratorForCodegenStage1.processNext(Unknown Source)

at org.apache.spark.sql.execution.BufferedRowIterator.hasNext(BufferedRowIterator.java:43)

at org.apache.spark.sql.execution.WholeStageCodegenExec$$anon$1.hasNext(WholeStageCodegenExec.scala:759)

at org.apache.spark.sql.execution.columnar.DefaultCachedBatchSerializer$$anon$1.next(InMemoryRelation.scala:87)

at org.apache.spark.sql.execution.columnar.DefaultCachedBatchSerializer$$anon$1.next(InMemoryRelation.scala:79)

at scala.collection.Iterator$$anon$10.next(Iterator.scala:461)

at org.apache.spark.storage.memory.MemoryStore.putIterator(MemoryStore.scala:224)

at org.apache.spark.storage.memory.MemoryStore.putIteratorAsValues(MemoryStore.scala:302)

at org.apache.spark.storage.BlockManager.$anonfun$doPutIterator$1(BlockManager.scala:1481)

at org.apache.spark.storage.BlockManager.org$apache$spark$storage$BlockManager$$doPut(BlockManager.scala:1408)

at org.apache.spark.storage.BlockManager.doPutIterator(BlockManager.scala:1472)

at org.apache.spark.storage.BlockManager.getOrElseUpdate(BlockManager.scala:1295)

at org.apache.spark.rdd.RDD.getOrCompute(RDD.scala:384)

at org.apache.spark.rdd.RDD.iterator(RDD.scala:335)

at org.apache.spark.rdd.MapPartitionsRDD.compute(MapPartitionsRDD.scala:52)

at org.apache.spark.rdd.RDD.computeOrReadCheckpoint(RDD.scala:373)

at org.apache.spark.rdd.RDD.iterator(RDD.scala:337)

at org.apache.spark.rdd.MapPartitionsRDD.compute(MapPartitionsRDD.scala:52)

at org.apache.spark.rdd.RDD.computeOrReadCheckpoint(RDD.scala:373)

at org.apache.spark.rdd.RDD.iterator(RDD.scala:337)

at org.apache.spark.rdd.MapPartitionsRDD.compute(MapPartitionsRDD.scala:52)

at org.apache.spark.rdd.RDD.computeOrReadCheckpoint(RDD.scala:373)

at org.apache.spark.rdd.RDD.iterator(RDD.scala:337)

at org.apache.spark.scheduler.ResultTask.runTask(ResultTask.scala:90)

at org.apache.spark.scheduler.Task.run(Task.scala:131)

at org.apache.spark.executor.Executor$TaskRunner.$anonfun$run$3(Executor.scala:506)

at org.apache.spark.util.Utils$.tryWithSafeFinally(Utils.scala:1462)

at org.apache.spark.executor.Executor$TaskRunner.run(Executor.scala:509)

at java.util.concurrent.ThreadPoolExecutor.runWorker(ThreadPoolExecutor.java:1149)

at java.util.concurrent.ThreadPoolExecutor$Worker.run(ThreadPoolExecutor.java:624)

at java.lang.Thread.run(Thread.java:750)

Driver stacktrace:

at org.apache.spark.scheduler.DAGScheduler.failJobAndIndependentStages(DAGScheduler.scala:2454)

at org.apache.spark.scheduler.DAGScheduler.$anonfun$abortStage$2(DAGScheduler.scala:2403)

at org.apache.spark.scheduler.DAGScheduler.$anonfun$abortStage$2$adapted(DAGScheduler.scala:2402)

at scala.collection.mutable.ResizableArray.foreach(ResizableArray.scala:62)

at scala.collection.mutable.ResizableArray.foreach$(ResizableArray.scala:55)

at scala.collection.mutable.ArrayBuffer.foreach(ArrayBuffer.scala:49)

at org.apache.spark.scheduler.DAGScheduler.abortStage(DAGScheduler.scala:2402)

at org.apache.spark.scheduler.DAGScheduler.$anonfun$handleTaskSetFailed$1(DAGScheduler.scala:1160)

at org.apache.spark.scheduler.DAGScheduler.$anonfun$handleTaskSetFailed$1$adapted(DAGScheduler.scala:1160)

at scala.Option.foreach(Option.scala:407)

at org.apache.spark.scheduler.DAGScheduler.handleTaskSetFailed(DAGScheduler.scala:1160)

at org.apache.spark.scheduler.DAGSchedulerEventProcessLoop.doOnReceive(DAGScheduler.scala:2642)

at org.apache.spark.scheduler.DAGSchedulerEventProcessLoop.onReceive(DAGScheduler.scala:2584)

at org.apache.spark.scheduler.DAGSchedulerEventProcessLoop.onReceive(DAGScheduler.scala:2573)

at org.apache.spark.util.EventLoop$$anon$1.run(EventLoop.scala:49)

at org.apache.spark.scheduler.DAGScheduler.runJob(DAGScheduler.scala:938)

at org.apache.spark.SparkContext.runJob(SparkContext.scala:2214)

at org.apache.spark.sql.execution.datasources.FileFormatWriter$.write(FileFormatWriter.scala:218)

... 42 more

Caused by: org.apache.spark.api.python.PythonException: Traceback (most recent call last):

File "/usr/local/spark/python/lib/pyspark.zip/pyspark/worker.py", line 619, in main

process()

File "/usr/local/spark/python/lib/pyspark.zip/pyspark/worker.py", line 611, in process

serializer.dump\_stream(out\_iter, outfile)

File "/usr/local/spark/python/lib/pyspark.zip/pyspark/serializers.py", line 211, in dump\_stream

self.serializer.dump\_stream(self.\_batched(iterator), stream)

File "/usr/local/spark/python/lib/pyspark.zip/pyspark/serializers.py", line 132, in dump\_stream

for obj in iterator:

File "/usr/local/spark/python/lib/pyspark.zip/pyspark/serializers.py", line 200, in \_batched

for item in iterator:

File "/usr/local/spark/python/lib/pyspark.zip/pyspark/worker.py", line 452, in mapper

result = tuple(f(\*[a[o] for o in arg\_offsets]) for (arg\_offsets, f) in udfs)

File "/usr/local/spark/python/lib/pyspark.zip/pyspark/worker.py", line 452, in <genexpr>

result = tuple(f(\*[a[o] for o in arg\_offsets]) for (arg\_offsets, f) in udfs)

File "/usr/local/spark/python/lib/pyspark.zip/pyspark/worker.py", line 87, in <lambda>

return lambda \*a: f(\*a)

File "/usr/local/spark/python/lib/pyspark.zip/pyspark/util.py", line 74, in wrapper

return f(\*args, \*\*kwargs)

File "<stdin>", line 3, in clean\_string

TypeError: expected string or bytes-like object

at org.apache.spark.api.python.BasePythonRunner$ReaderIterator.handlePythonException(PythonRunner.scala:555)

at org.apache.spark.sql.execution.python.PythonUDFRunner$$anon$2.read(PythonUDFRunner.scala:86)

at org.apache.spark.sql.execution.python.PythonUDFRunner$$anon$2.read(PythonUDFRunner.scala:68)

at org.apache.spark.api.python.BasePythonRunner$ReaderIterator.hasNext(PythonRunner.scala:508)

at org.apache.spark.InterruptibleIterator.hasNext(InterruptibleIterator.scala:37)

at scala.collection.Iterator$$anon$11.hasNext(Iterator.scala:491)

at scala.collection.Iterator$$anon$10.hasNext(Iterator.scala:460)

at scala.collection.Iterator$$anon$10.hasNext(Iterator.scala:460)

at org.apache.spark.sql.catalyst.expressions.GeneratedClass$GeneratedIteratorForCodegenStage1.processNext(Unknown Source)

at org.apache.spark.sql.execution.BufferedRowIterator.hasNext(BufferedRowIterator.java:43)

at org.apache.spark.sql.execution.WholeStageCodegenExec$$anon$1.hasNext(WholeStageCodegenExec.scala:759)

at org.apache.spark.sql.execution.columnar.DefaultCachedBatchSerializer$$anon$1.next(InMemoryRelation.scala:87)

at org.apache.spark.sql.execution.columnar.DefaultCachedBatchSerializer$$anon$1.next(InMemoryRelation.scala:79)

at scala.collection.Iterator$$anon$10.next(Iterator.scala:461)

at org.apache.spark.storage.memory.MemoryStore.putIterator(MemoryStore.scala:224)

at org.apache.spark.storage.memory.MemoryStore.putIteratorAsValues(MemoryStore.scala:302)

at org.apache.spark.storage.BlockManager.$anonfun$doPutIterator$1(BlockManager.scala:1481)

at org.apache.spark.storage.BlockManager.org$apache$spark$storage$BlockManager$$doPut(BlockManager.scala:1408)

at org.apache.spark.storage.BlockManager.doPutIterator(BlockManager.scala:1472)

at org.apache.spark.storage.BlockManager.getOrElseUpdate(BlockManager.scala:1295)

at org.apache.spark.rdd.RDD.getOrCompute(RDD.scala:384)

at org.apache.spark.rdd.RDD.iterator(RDD.scala:335)

at org.apache.spark.rdd.MapPartitionsRDD.compute(MapPartitionsRDD.scala:52)

at org.apache.spark.rdd.RDD.computeOrReadCheckpoint(RDD.scala:373)

at org.apache.spark.rdd.RDD.iterator(RDD.scala:337)

at org.apache.spark.rdd.MapPartitionsRDD.compute(MapPartitionsRDD.scala:52)

at org.apache.spark.rdd.RDD.computeOrReadCheckpoint(RDD.scala:373)

at org.apache.spark.rdd.RDD.iterator(RDD.scala:337)

at org.apache.spark.rdd.MapPartitionsRDD.compute(MapPartitionsRDD.scala:52)

at org.apache.spark.rdd.RDD.computeOrReadCheckpoint(RDD.scala:373)

at org.apache.spark.rdd.RDD.iterator(RDD.scala:337)

at org.apache.spark.scheduler.ResultTask.runTask(ResultTask.scala:90)

at org.apache.spark.scheduler.Task.run(Task.scala:131)

at org.apache.spark.executor.Executor$TaskRunner.$anonfun$run$3(Executor.scala:506)

at org.apache.spark.util.Utils$.tryWithSafeFinally(Utils.scala:1462)

at org.apache.spark.executor.Executor$TaskRunner.run(Executor.scala:509)

at java.util.concurrent.ThreadPoolExecutor.runWorker(ThreadPoolExecutor.java:1149)

at java.util.concurrent.ThreadPoolExecutor$Worker.run(ThreadPoolExecutor.java:624)

... 1 more

>>> print(sample\_op.count())

24/07/22 09:27:31 WARN storage.BlockManager: Putting block rdd\_531\_0 failed due to exception org.apache.spark.api.python.PythonException: Traceback (most recent call last):

File "/usr/local/spark/python/lib/pyspark.zip/pyspark/worker.py", line 619, in main

process()

File "/usr/local/spark/python/lib/pyspark.zip/pyspark/worker.py", line 611, in process

serializer.dump\_stream(out\_iter, outfile)

File "/usr/local/spark/python/lib/pyspark.zip/pyspark/serializers.py", line 211, in dump\_stream

self.serializer.dump\_stream(self.\_batched(iterator), stream)

File "/usr/local/spark/python/lib/pyspark.zip/pyspark/serializers.py", line 132, in dump\_stream

for obj in iterator:

File "/usr/local/spark/python/lib/pyspark.zip/pyspark/serializers.py", line 200, in \_batched

for item in iterator:

File "/usr/local/spark/python/lib/pyspark.zip/pyspark/worker.py", line 452, in mapper

result = tuple(f(\*[a[o] for o in arg\_offsets]) for (arg\_offsets, f) in udfs)

File "/usr/local/spark/python/lib/pyspark.zip/pyspark/worker.py", line 452, in <genexpr>

result = tuple(f(\*[a[o] for o in arg\_offsets]) for (arg\_offsets, f) in udfs)

File "/usr/local/spark/python/lib/pyspark.zip/pyspark/worker.py", line 87, in <lambda>

return lambda \*a: f(\*a)

File "/usr/local/spark/python/lib/pyspark.zip/pyspark/util.py", line 74, in wrapper

return f(\*args, \*\*kwargs)

File "<stdin>", line 3, in clean\_string

TypeError: expected string or bytes-like object

.

24/07/22 09:27:31 WARN storage.BlockManager: Block rdd\_531\_0 could not be removed as it was not found on disk or in memory

24/07/22 09:27:31 ERROR executor.Executor: Exception in task 0.0 in stage 102.0 (TID 111)

org.apache.spark.api.python.PythonException: Traceback (most recent call last):

File "/usr/local/spark/python/lib/pyspark.zip/pyspark/worker.py", line 619, in main

process()

File "/usr/local/spark/python/lib/pyspark.zip/pyspark/worker.py", line 611, in process

serializer.dump\_stream(out\_iter, outfile)

File "/usr/local/spark/python/lib/pyspark.zip/pyspark/serializers.py", line 211, in dump\_stream

self.serializer.dump\_stream(self.\_batched(iterator), stream)

File "/usr/local/spark/python/lib/pyspark.zip/pyspark/serializers.py", line 132, in dump\_stream

for obj in iterator:

File "/usr/local/spark/python/lib/pyspark.zip/pyspark/serializers.py", line 200, in \_batched

for item in iterator:

File "/usr/local/spark/python/lib/pyspark.zip/pyspark/worker.py", line 452, in mapper

result = tuple(f(\*[a[o] for o in arg\_offsets]) for (arg\_offsets, f) in udfs)

File "/usr/local/spark/python/lib/pyspark.zip/pyspark/worker.py", line 452, in <genexpr>

result = tuple(f(\*[a[o] for o in arg\_offsets]) for (arg\_offsets, f) in udfs)

File "/usr/local/spark/python/lib/pyspark.zip/pyspark/worker.py", line 87, in <lambda>

return lambda \*a: f(\*a)

File "/usr/local/spark/python/lib/pyspark.zip/pyspark/util.py", line 74, in wrapper

return f(\*args, \*\*kwargs)

File "<stdin>", line 3, in clean\_string

TypeError: expected string or bytes-like object

at org.apache.spark.api.python.BasePythonRunner$ReaderIterator.handlePythonException(PythonRunner.scala:555)

at org.apache.spark.sql.execution.python.PythonUDFRunner$$anon$2.read(PythonUDFRunner.scala:86)

at org.apache.spark.sql.execution.python.PythonUDFRunner$$anon$2.read(PythonUDFRunner.scala:68)

at org.apache.spark.api.python.BasePythonRunner$ReaderIterator.hasNext(PythonRunner.scala:508)

at org.apache.spark.InterruptibleIterator.hasNext(InterruptibleIterator.scala:37)

at scala.collection.Iterator$$anon$11.hasNext(Iterator.scala:491)

at scala.collection.Iterator$$anon$10.hasNext(Iterator.scala:460)

at scala.collection.Iterator$$anon$10.hasNext(Iterator.scala:460)

at org.apache.spark.sql.catalyst.expressions.GeneratedClass$GeneratedIteratorForCodegenStage1.processNext(Unknown Source)

at org.apache.spark.sql.execution.BufferedRowIterator.hasNext(BufferedRowIterator.java:43)

at org.apache.spark.sql.execution.WholeStageCodegenExec$$anon$1.hasNext(WholeStageCodegenExec.scala:759)

at org.apache.spark.sql.execution.columnar.DefaultCachedBatchSerializer$$anon$1.next(InMemoryRelation.scala:87)

at org.apache.spark.sql.execution.columnar.DefaultCachedBatchSerializer$$anon$1.next(InMemoryRelation.scala:79)

at scala.collection.Iterator$$anon$10.next(Iterator.scala:461)

at org.apache.spark.storage.memory.MemoryStore.putIterator(MemoryStore.scala:224)

at org.apache.spark.storage.memory.MemoryStore.putIteratorAsValues(MemoryStore.scala:302)

at org.apache.spark.storage.BlockManager.$anonfun$doPutIterator$1(BlockManager.scala:1481)

at org.apache.spark.storage.BlockManager.org$apache$spark$storage$BlockManager$$doPut(BlockManager.scala:1408)

at org.apache.spark.storage.BlockManager.doPutIterator(BlockManager.scala:1472)

at org.apache.spark.storage.BlockManager.getOrElseUpdate(BlockManager.scala:1295)

at org.apache.spark.rdd.RDD.getOrCompute(RDD.scala:384)

at org.apache.spark.rdd.RDD.iterator(RDD.scala:335)

at org.apache.spark.rdd.MapPartitionsRDD.compute(MapPartitionsRDD.scala:52)

at org.apache.spark.rdd.RDD.computeOrReadCheckpoint(RDD.scala:373)

at org.apache.spark.rdd.RDD.iterator(RDD.scala:337)

at org.apache.spark.rdd.MapPartitionsRDD.compute(MapPartitionsRDD.scala:52)

at org.apache.spark.rdd.RDD.computeOrReadCheckpoint(RDD.scala:373)

at org.apache.spark.rdd.RDD.iterator(RDD.scala:337)

at org.apache.spark.rdd.MapPartitionsRDD.compute(MapPartitionsRDD.scala:52)

at org.apache.spark.rdd.RDD.computeOrReadCheckpoint(RDD.scala:373)

at org.apache.spark.rdd.RDD.iterator(RDD.scala:337)

at org.apache.spark.rdd.MapPartitionsRDD.compute(MapPartitionsRDD.scala:52)

at org.apache.spark.rdd.RDD.computeOrReadCheckpoint(RDD.scala:373)

at org.apache.spark.rdd.RDD.iterator(RDD.scala:337)

at org.apache.spark.rdd.MapPartitionsRDD.compute(MapPartitionsRDD.scala:52)

at org.apache.spark.rdd.RDD.computeOrReadCheckpoint(RDD.scala:373)

at org.apache.spark.rdd.RDD.iterator(RDD.scala:337)

at org.apache.spark.shuffle.ShuffleWriteProcessor.write(ShuffleWriteProcessor.scala:59)

at org.apache.spark.scheduler.ShuffleMapTask.runTask(ShuffleMapTask.scala:99)

at org.apache.spark.scheduler.ShuffleMapTask.runTask(ShuffleMapTask.scala:52)

at org.apache.spark.scheduler.Task.run(Task.scala:131)

at org.apache.spark.executor.Executor$TaskRunner.$anonfun$run$3(Executor.scala:506)

at org.apache.spark.util.Utils$.tryWithSafeFinally(Utils.scala:1462)

at org.apache.spark.executor.Executor$TaskRunner.run(Executor.scala:509)

at java.util.concurrent.ThreadPoolExecutor.runWorker(ThreadPoolExecutor.java:1149)

at java.util.concurrent.ThreadPoolExecutor$Worker.run(ThreadPoolExecutor.java:624)

at java.lang.Thread.run(Thread.java:750)

24/07/22 09:27:31 WARN scheduler.TaskSetManager: Lost task 0.0 in stage 102.0 (TID 111) (192.168.112.129 executor driver): org.apache.spark.api.python.PythonException: Traceback (most recent call last):

File "/usr/local/spark/python/lib/pyspark.zip/pyspark/worker.py", line 619, in main

process()

File "/usr/local/spark/python/lib/pyspark.zip/pyspark/worker.py", line 611, in process

serializer.dump\_stream(out\_iter, outfile)

File "/usr/local/spark/python/lib/pyspark.zip/pyspark/serializers.py", line 211, in dump\_stream

self.serializer.dump\_stream(self.\_batched(iterator), stream)

File "/usr/local/spark/python/lib/pyspark.zip/pyspark/serializers.py", line 132, in dump\_stream

for obj in iterator:

File "/usr/local/spark/python/lib/pyspark.zip/pyspark/serializers.py", line 200, in \_batched

for item in iterator:

File "/usr/local/spark/python/lib/pyspark.zip/pyspark/worker.py", line 452, in mapper

result = tuple(f(\*[a[o] for o in arg\_offsets]) for (arg\_offsets, f) in udfs)

File "/usr/local/spark/python/lib/pyspark.zip/pyspark/worker.py", line 452, in <genexpr>

result = tuple(f(\*[a[o] for o in arg\_offsets]) for (arg\_offsets, f) in udfs)

File "/usr/local/spark/python/lib/pyspark.zip/pyspark/worker.py", line 87, in <lambda>

return lambda \*a: f(\*a)

File "/usr/local/spark/python/lib/pyspark.zip/pyspark/util.py", line 74, in wrapper

return f(\*args, \*\*kwargs)

File "<stdin>", line 3, in clean\_string

TypeError: expected string or bytes-like object

at org.apache.spark.api.python.BasePythonRunner$ReaderIterator.handlePythonException(PythonRunner.scala:555)

at org.apache.spark.sql.execution.python.PythonUDFRunner$$anon$2.read(PythonUDFRunner.scala:86)

at org.apache.spark.sql.execution.python.PythonUDFRunner$$anon$2.read(PythonUDFRunner.scala:68)

at org.apache.spark.api.python.BasePythonRunner$ReaderIterator.hasNext(PythonRunner.scala:508)

at org.apache.spark.InterruptibleIterator.hasNext(InterruptibleIterator.scala:37)

at scala.collection.Iterator$$anon$11.hasNext(Iterator.scala:491)

at scala.collection.Iterator$$anon$10.hasNext(Iterator.scala:460)

at scala.collection.Iterator$$anon$10.hasNext(Iterator.scala:460)

at org.apache.spark.sql.catalyst.expressions.GeneratedClass$GeneratedIteratorForCodegenStage1.processNext(Unknown Source)

at org.apache.spark.sql.execution.BufferedRowIterator.hasNext(BufferedRowIterator.java:43)

at org.apache.spark.sql.execution.WholeStageCodegenExec$$anon$1.hasNext(WholeStageCodegenExec.scala:759)

at org.apache.spark.sql.execution.columnar.DefaultCachedBatchSerializer$$anon$1.next(InMemoryRelation.scala:87)

at org.apache.spark.sql.execution.columnar.DefaultCachedBatchSerializer$$anon$1.next(InMemoryRelation.scala:79)

at scala.collection.Iterator$$anon$10.next(Iterator.scala:461)

at org.apache.spark.storage.memory.MemoryStore.putIterator(MemoryStore.scala:224)

at org.apache.spark.storage.memory.MemoryStore.putIteratorAsValues(MemoryStore.scala:302)

at org.apache.spark.storage.BlockManager.$anonfun$doPutIterator$1(BlockManager.scala:1481)

at org.apache.spark.storage.BlockManager.org$apache$spark$storage$BlockManager$$doPut(BlockManager.scala:1408)

at org.apache.spark.storage.BlockManager.doPutIterator(BlockManager.scala:1472)

at org.apache.spark.storage.BlockManager.getOrElseUpdate(BlockManager.scala:1295)

at org.apache.spark.rdd.RDD.getOrCompute(RDD.scala:384)

at org.apache.spark.rdd.RDD.iterator(RDD.scala:335)

at org.apache.spark.rdd.MapPartitionsRDD.compute(MapPartitionsRDD.scala:52)

at org.apache.spark.rdd.RDD.computeOrReadCheckpoint(RDD.scala:373)

at org.apache.spark.rdd.RDD.iterator(RDD.scala:337)

at org.apache.spark.rdd.MapPartitionsRDD.compute(MapPartitionsRDD.scala:52)

at org.apache.spark.rdd.RDD.computeOrReadCheckpoint(RDD.scala:373)

at org.apache.spark.rdd.RDD.iterator(RDD.scala:337)

at org.apache.spark.rdd.MapPartitionsRDD.compute(MapPartitionsRDD.scala:52)

at org.apache.spark.rdd.RDD.computeOrReadCheckpoint(RDD.scala:373)

at org.apache.spark.rdd.RDD.iterator(RDD.scala:337)

at org.apache.spark.rdd.MapPartitionsRDD.compute(MapPartitionsRDD.scala:52)

at org.apache.spark.rdd.RDD.computeOrReadCheckpoint(RDD.scala:373)

at org.apache.spark.rdd.RDD.iterator(RDD.scala:337)

at org.apache.spark.rdd.MapPartitionsRDD.compute(MapPartitionsRDD.scala:52)

at org.apache.spark.rdd.RDD.computeOrReadCheckpoint(RDD.scala:373)

at org.apache.spark.rdd.RDD.iterator(RDD.scala:337)

at org.apache.spark.shuffle.ShuffleWriteProcessor.write(ShuffleWriteProcessor.scala:59)

at org.apache.spark.scheduler.ShuffleMapTask.runTask(ShuffleMapTask.scala:99)

at org.apache.spark.scheduler.ShuffleMapTask.runTask(ShuffleMapTask.scala:52)

at org.apache.spark.scheduler.Task.run(Task.scala:131)

at org.apache.spark.executor.Executor$TaskRunner.$anonfun$run$3(Executor.scala:506)

at org.apache.spark.util.Utils$.tryWithSafeFinally(Utils.scala:1462)

at org.apache.spark.executor.Executor$TaskRunner.run(Executor.scala:509)

at java.util.concurrent.ThreadPoolExecutor.runWorker(ThreadPoolExecutor.java:1149)

at java.util.concurrent.ThreadPoolExecutor$Worker.run(ThreadPoolExecutor.java:624)

at java.lang.Thread.run(Thread.java:750)

24/07/22 09:27:31 ERROR scheduler.TaskSetManager: Task 0 in stage 102.0 failed 1 times; aborting job

24/07/22 09:27:31 WARN storage.BlockManager: Putting block rdd\_531\_1 failed due to exception org.apache.spark.TaskKilledException.

24/07/22 09:27:31 WARN storage.BlockManager: Block rdd\_531\_1 could not be removed as it was not found on disk or in memory

24/07/22 09:27:32 WARN scheduler.TaskSetManager: Lost task 1.0 in stage 102.0 (TID 112) (192.168.112.129 executor driver): TaskKilled (Stage cancelled)

Traceback (most recent call last):

File "<stdin>", line 1, in <module>

File "/usr/local/spark/python/pyspark/sql/dataframe.py", line 680, in count

return int(self.\_jdf.count())

File "/usr/local/spark/python/lib/py4j-0.10.9.3-src.zip/py4j/java\_gateway.py", line 1322, in \_\_call\_\_

File "/usr/local/spark/python/pyspark/sql/utils.py", line 117, in deco

raise converted from None

pyspark.sql.utils.PythonException:

An exception was thrown from the Python worker. Please see the stack trace below.

Traceback (most recent call last):

File "/usr/local/spark/python/lib/pyspark.zip/pyspark/worker.py", line 619, in main

process()

File "/usr/local/spark/python/lib/pyspark.zip/pyspark/worker.py", line 611, in process

serializer.dump\_stream(out\_iter, outfile)

File "/usr/local/spark/python/lib/pyspark.zip/pyspark/serializers.py", line 211, in dump\_stream

self.serializer.dump\_stream(self.\_batched(iterator), stream)

File "/usr/local/spark/python/lib/pyspark.zip/pyspark/serializers.py", line 132, in dump\_stream

for obj in iterator:

File "/usr/local/spark/python/lib/pyspark.zip/pyspark/serializers.py", line 200, in \_batched

for item in iterator:

File "/usr/local/spark/python/lib/pyspark.zip/pyspark/worker.py", line 452, in mapper

result = tuple(f(\*[a[o] for o in arg\_offsets]) for (arg\_offsets, f) in udfs)

File "/usr/local/spark/python/lib/pyspark.zip/pyspark/worker.py", line 452, in <genexpr>

result = tuple(f(\*[a[o] for o in arg\_offsets]) for (arg\_offsets, f) in udfs)

File "/usr/local/spark/python/lib/pyspark.zip/pyspark/worker.py", line 87, in <lambda>

return lambda \*a: f(\*a)

File "/usr/local/spark/python/lib/pyspark.zip/pyspark/util.py", line 74, in wrapper

return f(\*args, \*\*kwargs)

File "<stdin>", line 3, in clean\_string

TypeError: expected string or bytes-like object

>>> print(sample\_op.type())

Traceback (most recent call last):

File "<stdin>", line 1, in <module>

File "/usr/local/spark/python/pyspark/sql/dataframe.py", line 1660, in \_\_getattr\_\_

"'%s' object has no attribute '%s'" % (self.\_\_class\_\_.\_\_name\_\_, name))

AttributeError: 'DataFrame' object has no attribute 'type'

>>> sample\_op.describe

<bound method DataFrame.describe of DataFrame[BusinessDate: date, stcd: string, srcnm: string, NetworkName: string, NetworkURL: string, custnum: string, MarketCoverage: string, RejectRows: string, IssuerId: string, IssuerId2: string, issueridcomposite: string, sysdt: date, systs: timestamp, New NetworkName: string]>

>>> sample\_op.

sample\_op.agg( sample\_op.dropna( sample\_op.na sample\_op.subtract(

sample\_op.alias( sample\_op.dtypes sample\_op.orderBy( sample\_op.summary(

sample\_op.approxQuantile( sample\_op.exceptAll( sample\_op.persist( sample\_op.tail(

sample\_op.cache( sample\_op.explain( sample\_op.printSchema( sample\_op.take(

sample\_op.checkpoint( sample\_op.fillna( sample\_op.randomSplit( sample\_op.toDF(

sample\_op.coalesce( sample\_op.filter( sample\_op.rdd sample\_op.toJSON(

sample\_op.colRegex( sample\_op.first( sample\_op.registerTempTable( sample\_op.toLocalIterator(

sample\_op.collect( sample\_op.foreach( sample\_op.repartition( sample\_op.toPandas(

sample\_op.columns sample\_op.foreachPartition( sample\_op.repartitionByRange( sample\_op.to\_koalas(

sample\_op.corr( sample\_op.freqItems( sample\_op.replace( sample\_op.to\_pandas\_on\_spark(

sample\_op.count( sample\_op.groupBy( sample\_op.rollup( sample\_op.transform(

sample\_op.cov( sample\_op.groupby( sample\_op.sameSemantics( sample\_op.union(

sample\_op.createGlobalTempView( sample\_op.head( sample\_op.sample( sample\_op.unionAll(

sample\_op.createOrReplaceGlobalTempView( sample\_op.hint( sample\_op.sampleBy( sample\_op.unionByName(

sample\_op.createOrReplaceTempView( sample\_op.inputFiles( sample\_op.schema sample\_op.unpersist(

sample\_op.createTempView( sample\_op.intersect( sample\_op.select( sample\_op.where(

sample\_op.crossJoin( sample\_op.intersectAll( sample\_op.selectExpr( sample\_op.withColumn(

sample\_op.crosstab( sample\_op.isLocal( sample\_op.semanticHash( sample\_op.withColumnRenamed(

sample\_op.cube( sample\_op.isStreaming sample\_op.show( sample\_op.withWatermark(

sample\_op.describe( sample\_op.is\_cached sample\_op.sort( sample\_op.write

sample\_op.distinct( sample\_op.join( sample\_op.sortWithinPartitions( sample\_op.writeStream

sample\_op.drop( sample\_op.limit( sample\_op.sql\_ctx sample\_op.writeTo(

sample\_op.dropDuplicates( sample\_op.localCheckpoint( sample\_op.stat

sample\_op.drop\_duplicates( sample\_op.mapInPandas( sample\_op.storageLevel

>>> sample\_op.

sample\_op.agg( sample\_op.dropna( sample\_op.na sample\_op.subtract(

sample\_op.alias( sample\_op.dtypes sample\_op.orderBy( sample\_op.summary(

sample\_op.approxQuantile( sample\_op.exceptAll( sample\_op.persist( sample\_op.tail(

sample\_op.cache( sample\_op.explain( sample\_op.printSchema( sample\_op.take(

sample\_op.checkpoint( sample\_op.fillna( sample\_op.randomSplit( sample\_op.toDF(

sample\_op.coalesce( sample\_op.filter( sample\_op.rdd sample\_op.toJSON(

sample\_op.colRegex( sample\_op.first( sample\_op.registerTempTable( sample\_op.toLocalIterator(

sample\_op.collect( sample\_op.foreach( sample\_op.repartition( sample\_op.toPandas(

sample\_op.columns sample\_op.foreachPartition( sample\_op.repartitionByRange( sample\_op.to\_koalas(

sample\_op.corr( sample\_op.freqItems( sample\_op.replace( sample\_op.to\_pandas\_on\_spark(

sample\_op.count( sample\_op.groupBy( sample\_op.rollup( sample\_op.transform(

sample\_op.cov( sample\_op.groupby( sample\_op.sameSemantics( sample\_op.union(

sample\_op.createGlobalTempView( sample\_op.head( sample\_op.sample( sample\_op.unionAll(

sample\_op.createOrReplaceGlobalTempView( sample\_op.hint( sample\_op.sampleBy( sample\_op.unionByName(

sample\_op.createOrReplaceTempView( sample\_op.inputFiles( sample\_op.schema sample\_op.unpersist(

sample\_op.createTempView( sample\_op.intersect( sample\_op.select( sample\_op.where(

sample\_op.crossJoin( sample\_op.intersectAll( sample\_op.selectExpr( sample\_op.withColumn(

sample\_op.crosstab( sample\_op.isLocal( sample\_op.semanticHash( sample\_op.withColumnRenamed(

sample\_op.cube( sample\_op.isStreaming sample\_op.show( sample\_op.withWatermark(

sample\_op.describe( sample\_op.is\_cached sample\_op.sort( sample\_op.write

sample\_op.distinct( sample\_op.join( sample\_op.sortWithinPartitions( sample\_op.writeStream

sample\_op.drop( sample\_op.limit( sample\_op.sql\_ctx sample\_op.writeTo(

sample\_op.dropDuplicates( sample\_op.localCheckpoint( sample\_op.stat

sample\_op.drop\_duplicates( sample\_op.mapInPandas( sample\_op.storageLevel

>>> sample\_op.w

sample\_op.where( sample\_op.withColumnRenamed( sample\_op.write sample\_op.writeTo(

sample\_op.withColumn( sample\_op.withWatermark( sample\_op.writeStream

>>> sample\_op.w

sample\_op.where( sample\_op.withColumnRenamed( sample\_op.write sample\_op.writeTo(

sample\_op.withColumn( sample\_op.withWatermark( sample\_op.writeStream

>>> sample\_op.write

sample\_op.write sample\_op.writeStream sample\_op.writeTo(

>>> sample\_op.write

sample\_op.write sample\_op.writeStream sample\_op.writeTo(

>>> sample\_op.write.

sample\_op.write.bucketBy( sample\_op.write.jdbc( sample\_op.write.options( sample\_op.write.save(

sample\_op.write.csv( sample\_op.write.json( sample\_op.write.orc( sample\_op.write.saveAsTable(

sample\_op.write.format( sample\_op.write.mode( sample\_op.write.parquet( sample\_op.write.sortBy(

sample\_op.write.insertInto( sample\_op.write.option( sample\_op.write.partitionBy( sample\_op.write.text(

>>> sample\_op.write.

sample\_op.write.bucketBy( sample\_op.write.jdbc( sample\_op.write.options( sample\_op.write.save(

sample\_op.write.csv( sample\_op.write.json( sample\_op.write.orc( sample\_op.write.saveAsTable(

sample\_op.write.format( sample\_op.write.mode( sample\_op.write.parquet( sample\_op.write.sortBy(

sample\_op.write.insertInto( sample\_op.write.option( sample\_op.write.partitionBy( sample\_op.write.text(

>>> sample\_op.write.j

sample\_op.write.jdbc( sample\_op.write.json(

>>> sample\_op.write.json("/user/hduser/Hackathon/JSON\_DATA" , mode="overwrite")

24/07/22 09:28:58 WARN storage.BlockManager: Putting block rdd\_531\_0 failed due to exception org.apache.spark.api.python.PythonException: Traceback (most recent call last):

File "/usr/local/spark/python/lib/pyspark.zip/pyspark/worker.py", line 619, in main

process()

File "/usr/local/spark/python/lib/pyspark.zip/pyspark/worker.py", line 611, in process

serializer.dump\_stream(out\_iter, outfile)

File "/usr/local/spark/python/lib/pyspark.zip/pyspark/serializers.py", line 211, in dump\_stream

self.serializer.dump\_stream(self.\_batched(iterator), stream)

File "/usr/local/spark/python/lib/pyspark.zip/pyspark/serializers.py", line 132, in dump\_stream

for obj in iterator:

File "/usr/local/spark/python/lib/pyspark.zip/pyspark/serializers.py", line 200, in \_batched

for item in iterator:

File "/usr/local/spark/python/lib/pyspark.zip/pyspark/worker.py", line 452, in mapper

result = tuple(f(\*[a[o] for o in arg\_offsets]) for (arg\_offsets, f) in udfs)

File "/usr/local/spark/python/lib/pyspark.zip/pyspark/worker.py", line 452, in <genexpr>

result = tuple(f(\*[a[o] for o in arg\_offsets]) for (arg\_offsets, f) in udfs)

File "/usr/local/spark/python/lib/pyspark.zip/pyspark/worker.py", line 87, in <lambda>

return lambda \*a: f(\*a)

File "/usr/local/spark/python/lib/pyspark.zip/pyspark/util.py", line 74, in wrapper

return f(\*args, \*\*kwargs)

File "<stdin>", line 3, in clean\_string

TypeError: expected string or bytes-like object

.

24/07/22 09:28:58 WARN storage.BlockManager: Block rdd\_531\_0 could not be removed as it was not found on disk or in memory

24/07/22 09:28:58 ERROR executor.Executor: Exception in task 0.0 in stage 103.0 (TID 113)

org.apache.spark.api.python.PythonException: Traceback (most recent call last):

File "/usr/local/spark/python/lib/pyspark.zip/pyspark/worker.py", line 619, in main

process()

File "/usr/local/spark/python/lib/pyspark.zip/pyspark/worker.py", line 611, in process

serializer.dump\_stream(out\_iter, outfile)

File "/usr/local/spark/python/lib/pyspark.zip/pyspark/serializers.py", line 211, in dump\_stream

self.serializer.dump\_stream(self.\_batched(iterator), stream)

File "/usr/local/spark/python/lib/pyspark.zip/pyspark/serializers.py", line 132, in dump\_stream

for obj in iterator:

File "/usr/local/spark/python/lib/pyspark.zip/pyspark/serializers.py", line 200, in \_batched

for item in iterator:

File "/usr/local/spark/python/lib/pyspark.zip/pyspark/worker.py", line 452, in mapper

result = tuple(f(\*[a[o] for o in arg\_offsets]) for (arg\_offsets, f) in udfs)

File "/usr/local/spark/python/lib/pyspark.zip/pyspark/worker.py", line 452, in <genexpr>

result = tuple(f(\*[a[o] for o in arg\_offsets]) for (arg\_offsets, f) in udfs)

File "/usr/local/spark/python/lib/pyspark.zip/pyspark/worker.py", line 87, in <lambda>

return lambda \*a: f(\*a)

File "/usr/local/spark/python/lib/pyspark.zip/pyspark/util.py", line 74, in wrapper

return f(\*args, \*\*kwargs)

File "<stdin>", line 3, in clean\_string

TypeError: expected string or bytes-like object

at org.apache.spark.api.python.BasePythonRunner$ReaderIterator.handlePythonException(PythonRunner.scala:555)

at org.apache.spark.sql.execution.python.PythonUDFRunner$$anon$2.read(PythonUDFRunner.scala:86)

at org.apache.spark.sql.execution.python.PythonUDFRunner$$anon$2.read(PythonUDFRunner.scala:68)

at org.apache.spark.api.python.BasePythonRunner$ReaderIterator.hasNext(PythonRunner.scala:508)

at org.apache.spark.InterruptibleIterator.hasNext(InterruptibleIterator.scala:37)

at scala.collection.Iterator$$anon$11.hasNext(Iterator.scala:491)

at scala.collection.Iterator$$anon$10.hasNext(Iterator.scala:460)

at scala.collection.Iterator$$anon$10.hasNext(Iterator.scala:460)

at org.apache.spark.sql.catalyst.expressions.GeneratedClass$GeneratedIteratorForCodegenStage1.processNext(Unknown Source)

at org.apache.spark.sql.execution.BufferedRowIterator.hasNext(BufferedRowIterator.java:43)

at org.apache.spark.sql.execution.WholeStageCodegenExec$$anon$1.hasNext(WholeStageCodegenExec.scala:759)

at org.apache.spark.sql.execution.columnar.DefaultCachedBatchSerializer$$anon$1.next(InMemoryRelation.scala:87)

at org.apache.spark.sql.execution.columnar.DefaultCachedBatchSerializer$$anon$1.next(InMemoryRelation.scala:79)

at scala.collection.Iterator$$anon$10.next(Iterator.scala:461)

at org.apache.spark.storage.memory.MemoryStore.putIterator(MemoryStore.scala:224)

at org.apache.spark.storage.memory.MemoryStore.putIteratorAsValues(MemoryStore.scala:302)

at org.apache.spark.storage.BlockManager.$anonfun$doPutIterator$1(BlockManager.scala:1481)

at org.apache.spark.storage.BlockManager.org$apache$spark$storage$BlockManager$$doPut(BlockManager.scala:1408)

at org.apache.spark.storage.BlockManager.doPutIterator(BlockManager.scala:1472)

at org.apache.spark.storage.BlockManager.getOrElseUpdate(BlockManager.scala:1295)

at org.apache.spark.rdd.RDD.getOrCompute(RDD.scala:384)

at org.apache.spark.rdd.RDD.iterator(RDD.scala:335)

at org.apache.spark.rdd.MapPartitionsRDD.compute(MapPartitionsRDD.scala:52)

at org.apache.spark.rdd.RDD.computeOrReadCheckpoint(RDD.scala:373)

at org.apache.spark.rdd.RDD.iterator(RDD.scala:337)

at org.apache.spark.rdd.MapPartitionsRDD.compute(MapPartitionsRDD.scala:52)

at org.apache.spark.rdd.RDD.computeOrReadCheckpoint(RDD.scala:373)

at org.apache.spark.rdd.RDD.iterator(RDD.scala:337)

at org.apache.spark.rdd.MapPartitionsRDD.compute(MapPartitionsRDD.scala:52)

at org.apache.spark.rdd.RDD.computeOrReadCheckpoint(RDD.scala:373)

at org.apache.spark.rdd.RDD.iterator(RDD.scala:337)

at org.apache.spark.scheduler.ResultTask.runTask(ResultTask.scala:90)

at org.apache.spark.scheduler.Task.run(Task.scala:131)

at org.apache.spark.executor.Executor$TaskRunner.$anonfun$run$3(Executor.scala:506)

at org.apache.spark.util.Utils$.tryWithSafeFinally(Utils.scala:1462)

at org.apache.spark.executor.Executor$TaskRunner.run(Executor.scala:509)

at java.util.concurrent.ThreadPoolExecutor.runWorker(ThreadPoolExecutor.java:1149)

at java.util.concurrent.ThreadPoolExecutor$Worker.run(ThreadPoolExecutor.java:624)

at java.lang.Thread.run(Thread.java:750)

24/07/22 09:28:58 WARN scheduler.TaskSetManager: Lost task 0.0 in stage 103.0 (TID 113) (192.168.112.129 executor driver): org.apache.spark.api.python.PythonException: Traceback (most recent call last):

File "/usr/local/spark/python/lib/pyspark.zip/pyspark/worker.py", line 619, in main

process()

File "/usr/local/spark/python/lib/pyspark.zip/pyspark/worker.py", line 611, in process

serializer.dump\_stream(out\_iter, outfile)

File "/usr/local/spark/python/lib/pyspark.zip/pyspark/serializers.py", line 211, in dump\_stream

self.serializer.dump\_stream(self.\_batched(iterator), stream)

File "/usr/local/spark/python/lib/pyspark.zip/pyspark/serializers.py", line 132, in dump\_stream

for obj in iterator:

File "/usr/local/spark/python/lib/pyspark.zip/pyspark/serializers.py", line 200, in \_batched

for item in iterator:

File "/usr/local/spark/python/lib/pyspark.zip/pyspark/worker.py", line 452, in mapper

result = tuple(f(\*[a[o] for o in arg\_offsets]) for (arg\_offsets, f) in udfs)

File "/usr/local/spark/python/lib/pyspark.zip/pyspark/worker.py", line 452, in <genexpr>

result = tuple(f(\*[a[o] for o in arg\_offsets]) for (arg\_offsets, f) in udfs)

File "/usr/local/spark/python/lib/pyspark.zip/pyspark/worker.py", line 87, in <lambda>

return lambda \*a: f(\*a)

File "/usr/local/spark/python/lib/pyspark.zip/pyspark/util.py", line 74, in wrapper

return f(\*args, \*\*kwargs)

File "<stdin>", line 3, in clean\_string

TypeError: expected string or bytes-like object

at org.apache.spark.api.python.BasePythonRunner$ReaderIterator.handlePythonException(PythonRunner.scala:555)

at org.apache.spark.sql.execution.python.PythonUDFRunner$$anon$2.read(PythonUDFRunner.scala:86)

at org.apache.spark.sql.execution.python.PythonUDFRunner$$anon$2.read(PythonUDFRunner.scala:68)

at org.apache.spark.api.python.BasePythonRunner$ReaderIterator.hasNext(PythonRunner.scala:508)

at org.apache.spark.InterruptibleIterator.hasNext(InterruptibleIterator.scala:37)

at scala.collection.Iterator$$anon$11.hasNext(Iterator.scala:491)

at scala.collection.Iterator$$anon$10.hasNext(Iterator.scala:460)

at scala.collection.Iterator$$anon$10.hasNext(Iterator.scala:460)

at org.apache.spark.sql.catalyst.expressions.GeneratedClass$GeneratedIteratorForCodegenStage1.processNext(Unknown Source)

at org.apache.spark.sql.execution.BufferedRowIterator.hasNext(BufferedRowIterator.java:43)

at org.apache.spark.sql.execution.WholeStageCodegenExec$$anon$1.hasNext(WholeStageCodegenExec.scala:759)

at org.apache.spark.sql.execution.columnar.DefaultCachedBatchSerializer$$anon$1.next(InMemoryRelation.scala:87)

at org.apache.spark.sql.execution.columnar.DefaultCachedBatchSerializer$$anon$1.next(InMemoryRelation.scala:79)

at scala.collection.Iterator$$anon$10.next(Iterator.scala:461)

at org.apache.spark.storage.memory.MemoryStore.putIterator(MemoryStore.scala:224)

at org.apache.spark.storage.memory.MemoryStore.putIteratorAsValues(MemoryStore.scala:302)

at org.apache.spark.storage.BlockManager.$anonfun$doPutIterator$1(BlockManager.scala:1481)

at org.apache.spark.storage.BlockManager.org$apache$spark$storage$BlockManager$$doPut(BlockManager.scala:1408)

at org.apache.spark.storage.BlockManager.doPutIterator(BlockManager.scala:1472)

at org.apache.spark.storage.BlockManager.getOrElseUpdate(BlockManager.scala:1295)

at org.apache.spark.rdd.RDD.getOrCompute(RDD.scala:384)

at org.apache.spark.rdd.RDD.iterator(RDD.scala:335)

at org.apache.spark.rdd.MapPartitionsRDD.compute(MapPartitionsRDD.scala:52)

at org.apache.spark.rdd.RDD.computeOrReadCheckpoint(RDD.scala:373)

at org.apache.spark.rdd.RDD.iterator(RDD.scala:337)

at org.apache.spark.rdd.MapPartitionsRDD.compute(MapPartitionsRDD.scala:52)

at org.apache.spark.rdd.RDD.computeOrReadCheckpoint(RDD.scala:373)

at org.apache.spark.rdd.RDD.iterator(RDD.scala:337)

at org.apache.spark.rdd.MapPartitionsRDD.compute(MapPartitionsRDD.scala:52)

at org.apache.spark.rdd.RDD.computeOrReadCheckpoint(RDD.scala:373)

at org.apache.spark.rdd.RDD.iterator(RDD.scala:337)

at org.apache.spark.scheduler.ResultTask.runTask(ResultTask.scala:90)

at org.apache.spark.scheduler.Task.run(Task.scala:131)

at org.apache.spark.executor.Executor$TaskRunner.$anonfun$run$3(Executor.scala:506)

at org.apache.spark.util.Utils$.tryWithSafeFinally(Utils.scala:1462)

at org.apache.spark.executor.Executor$TaskRunner.run(Executor.scala:509)

at java.util.concurrent.ThreadPoolExecutor.runWorker(ThreadPoolExecutor.java:1149)

at java.util.concurrent.ThreadPoolExecutor$Worker.run(ThreadPoolExecutor.java:624)

at java.lang.Thread.run(Thread.java:750)

24/07/22 09:28:58 ERROR scheduler.TaskSetManager: Task 0 in stage 103.0 failed 1 times; aborting job

24/07/22 09:28:58 WARN storage.BlockManager: Putting block rdd\_531\_1 failed due to exception org.apache.spark.TaskKilledException.

24/07/22 09:28:58 WARN storage.BlockManager: Block rdd\_531\_1 could not be removed as it was not found on disk or in memory

24/07/22 09:28:58 ERROR datasources.FileFormatWriter: Aborting job b2bb0d1f-3267-415d-9a58-56098688cbfd.

org.apache.spark.SparkException: Job aborted due to stage failure: Task 0 in stage 103.0 failed 1 times, most recent failure: Lost task 0.0 in stage 103.0 (TID 113) (192.168.112.129 executor driver): org.apache.spark.api.python.PythonException: Traceback (most recent call last):

File "/usr/local/spark/python/lib/pyspark.zip/pyspark/worker.py", line 619, in main

process()

File "/usr/local/spark/python/lib/pyspark.zip/pyspark/worker.py", line 611, in process

serializer.dump\_stream(out\_iter, outfile)

File "/usr/local/spark/python/lib/pyspark.zip/pyspark/serializers.py", line 211, in dump\_stream

self.serializer.dump\_stream(self.\_batched(iterator), stream)

File "/usr/local/spark/python/lib/pyspark.zip/pyspark/serializers.py", line 132, in dump\_stream

for obj in iterator:

File "/usr/local/spark/python/lib/pyspark.zip/pyspark/serializers.py", line 200, in \_batched

for item in iterator:

File "/usr/local/spark/python/lib/pyspark.zip/pyspark/worker.py", line 452, in mapper

result = tuple(f(\*[a[o] for o in arg\_offsets]) for (arg\_offsets, f) in udfs)

File "/usr/local/spark/python/lib/pyspark.zip/pyspark/worker.py", line 452, in <genexpr>

result = tuple(f(\*[a[o] for o in arg\_offsets]) for (arg\_offsets, f) in udfs)

File "/usr/local/spark/python/lib/pyspark.zip/pyspark/worker.py", line 87, in <lambda>

return lambda \*a: f(\*a)

File "/usr/local/spark/python/lib/pyspark.zip/pyspark/util.py", line 74, in wrapper

return f(\*args, \*\*kwargs)

File "<stdin>", line 3, in clean\_string

TypeError: expected string or bytes-like object

at org.apache.spark.api.python.BasePythonRunner$ReaderIterator.handlePythonException(PythonRunner.scala:555)

at org.apache.spark.sql.execution.python.PythonUDFRunner$$anon$2.read(PythonUDFRunner.scala:86)

at org.apache.spark.sql.execution.python.PythonUDFRunner$$anon$2.read(PythonUDFRunner.scala:68)

at org.apache.spark.api.python.BasePythonRunner$ReaderIterator.hasNext(PythonRunner.scala:508)

at org.apache.spark.InterruptibleIterator.hasNext(InterruptibleIterator.scala:37)

at scala.collection.Iterator$$anon$11.hasNext(Iterator.scala:491)

at scala.collection.Iterator$$anon$10.hasNext(Iterator.scala:460)

at scala.collection.Iterator$$anon$10.hasNext(Iterator.scala:460)

at org.apache.spark.sql.catalyst.expressions.GeneratedClass$GeneratedIteratorForCodegenStage1.processNext(Unknown Source)

at org.apache.spark.sql.execution.BufferedRowIterator.hasNext(BufferedRowIterator.java:43)

at org.apache.spark.sql.execution.WholeStageCodegenExec$$anon$1.hasNext(WholeStageCodegenExec.scala:759)

at org.apache.spark.sql.execution.columnar.DefaultCachedBatchSerializer$$anon$1.next(InMemoryRelation.scala:87)

at org.apache.spark.sql.execution.columnar.DefaultCachedBatchSerializer$$anon$1.next(InMemoryRelation.scala:79)

at scala.collection.Iterator$$anon$10.next(Iterator.scala:461)

at org.apache.spark.storage.memory.MemoryStore.putIterator(MemoryStore.scala:224)

at org.apache.spark.storage.memory.MemoryStore.putIteratorAsValues(MemoryStore.scala:302)

at org.apache.spark.storage.BlockManager.$anonfun$doPutIterator$1(BlockManager.scala:1481)

at org.apache.spark.storage.BlockManager.org$apache$spark$storage$BlockManager$$doPut(BlockManager.scala:1408)

at org.apache.spark.storage.BlockManager.doPutIterator(BlockManager.scala:1472)

at org.apache.spark.storage.BlockManager.getOrElseUpdate(BlockManager.scala:1295)

at org.apache.spark.rdd.RDD.getOrCompute(RDD.scala:384)

at org.apache.spark.rdd.RDD.iterator(RDD.scala:335)

at org.apache.spark.rdd.MapPartitionsRDD.compute(MapPartitionsRDD.scala:52)

at org.apache.spark.rdd.RDD.computeOrReadCheckpoint(RDD.scala:373)

at org.apache.spark.rdd.RDD.iterator(RDD.scala:337)

at org.apache.spark.rdd.MapPartitionsRDD.compute(MapPartitionsRDD.scala:52)

at org.apache.spark.rdd.RDD.computeOrReadCheckpoint(RDD.scala:373)

at org.apache.spark.rdd.RDD.iterator(RDD.scala:337)

at org.apache.spark.rdd.MapPartitionsRDD.compute(MapPartitionsRDD.scala:52)

at org.apache.spark.rdd.RDD.computeOrReadCheckpoint(RDD.scala:373)

at org.apache.spark.rdd.RDD.iterator(RDD.scala:337)

at org.apache.spark.scheduler.ResultTask.runTask(ResultTask.scala:90)

at org.apache.spark.scheduler.Task.run(Task.scala:131)

at org.apache.spark.executor.Executor$TaskRunner.$anonfun$run$3(Executor.scala:506)

at org.apache.spark.util.Utils$.tryWithSafeFinally(Utils.scala:1462)

at org.apache.spark.executor.Executor$TaskRunner.run(Executor.scala:509)

at java.util.concurrent.ThreadPoolExecutor.runWorker(ThreadPoolExecutor.java:1149)

at java.util.concurrent.ThreadPoolExecutor$Worker.run(ThreadPoolExecutor.java:624)

at java.lang.Thread.run(Thread.java:750)

Driver stacktrace:

at org.apache.spark.scheduler.DAGScheduler.failJobAndIndependentStages(DAGScheduler.scala:2454)

at org.apache.spark.scheduler.DAGScheduler.$anonfun$abortStage$2(DAGScheduler.scala:2403)

at org.apache.spark.scheduler.DAGScheduler.$anonfun$abortStage$2$adapted(DAGScheduler.scala:2402)

at scala.collection.mutable.ResizableArray.foreach(ResizableArray.scala:62)

at scala.collection.mutable.ResizableArray.foreach$(ResizableArray.scala:55)

at scala.collection.mutable.ArrayBuffer.foreach(ArrayBuffer.scala:49)

at org.apache.spark.scheduler.DAGScheduler.abortStage(DAGScheduler.scala:2402)

at org.apache.spark.scheduler.DAGScheduler.$anonfun$handleTaskSetFailed$1(DAGScheduler.scala:1160)

at org.apache.spark.scheduler.DAGScheduler.$anonfun$handleTaskSetFailed$1$adapted(DAGScheduler.scala:1160)

at scala.Option.foreach(Option.scala:407)

at org.apache.spark.scheduler.DAGScheduler.handleTaskSetFailed(DAGScheduler.scala:1160)

at org.apache.spark.scheduler.DAGSchedulerEventProcessLoop.doOnReceive(DAGScheduler.scala:2642)

at org.apache.spark.scheduler.DAGSchedulerEventProcessLoop.onReceive(DAGScheduler.scala:2584)

at org.apache.spark.scheduler.DAGSchedulerEventProcessLoop.onReceive(DAGScheduler.scala:2573)

at org.apache.spark.util.EventLoop$$anon$1.run(EventLoop.scala:49)

at org.apache.spark.scheduler.DAGScheduler.runJob(DAGScheduler.scala:938)

at org.apache.spark.SparkContext.runJob(SparkContext.scala:2214)

at org.apache.spark.sql.execution.datasources.FileFormatWriter$.write(FileFormatWriter.scala:218)

at org.apache.spark.sql.execution.datasources.InsertIntoHadoopFsRelationCommand.run(InsertIntoHadoopFsRelationCommand.scala:186)

at org.apache.spark.sql.execution.command.DataWritingCommandExec.sideEffectResult$lzycompute(commands.scala:113)

at org.apache.spark.sql.execution.command.DataWritingCommandExec.sideEffectResult(commands.scala:111)

at org.apache.spark.sql.execution.command.DataWritingCommandExec.executeCollect(commands.scala:125)

at org.apache.spark.sql.execution.QueryExecution$$anonfun$eagerlyExecuteCommands$1.$anonfun$applyOrElse$1(QueryExecution.scala:110)

at org.apache.spark.sql.execution.SQLExecution$.$anonfun$withNewExecutionId$5(SQLExecution.scala:103)

at org.apache.spark.sql.execution.SQLExecution$.withSQLConfPropagated(SQLExecution.scala:163)

at org.apache.spark.sql.execution.SQLExecution$.$anonfun$withNewExecutionId$1(SQLExecution.scala:90)

at org.apache.spark.sql.SparkSession.withActive(SparkSession.scala:775)

at org.apache.spark.sql.execution.SQLExecution$.withNewExecutionId(SQLExecution.scala:64)

at org.apache.spark.sql.execution.QueryExecution$$anonfun$eagerlyExecuteCommands$1.applyOrElse(QueryExecution.scala:110)

at org.apache.spark.sql.execution.QueryExecution$$anonfun$eagerlyExecuteCommands$1.applyOrElse(QueryExecution.scala:106)

at org.apache.spark.sql.catalyst.trees.TreeNode.$anonfun$transformDownWithPruning$1(TreeNode.scala:481)

at org.apache.spark.sql.catalyst.trees.CurrentOrigin$.withOrigin(TreeNode.scala:82)

at org.apache.spark.sql.catalyst.trees.TreeNode.transformDownWithPruning(TreeNode.scala:481)

at org.apache.spark.sql.catalyst.plans.logical.LogicalPlan.org$apache$spark$sql$catalyst$plans$logical$AnalysisHelper$$super$transformDownWithPruning(LogicalPlan.scala:30)

at org.apache.spark.sql.catalyst.plans.logical.AnalysisHelper.transformDownWithPruning(AnalysisHelper.scala:267)

at org.apache.spark.sql.catalyst.plans.logical.AnalysisHelper.transformDownWithPruning$(AnalysisHelper.scala:263)

at org.apache.spark.sql.catalyst.plans.logical.LogicalPlan.transformDownWithPruning(LogicalPlan.scala:30)

at org.apache.spark.sql.catalyst.plans.logical.LogicalPlan.transformDownWithPruning(LogicalPlan.scala:30)

at org.apache.spark.sql.catalyst.trees.TreeNode.transformDown(TreeNode.scala:457)

at org.apache.spark.sql.execution.QueryExecution.eagerlyExecuteCommands(QueryExecution.scala:106)

at org.apache.spark.sql.execution.QueryExecution.commandExecuted$lzycompute(QueryExecution.scala:93)

at org.apache.spark.sql.execution.QueryExecution.commandExecuted(QueryExecution.scala:91)

at org.apache.spark.sql.execution.QueryExecution.assertCommandExecuted(QueryExecution.scala:128)

at org.apache.spark.sql.DataFrameWriter.runCommand(DataFrameWriter.scala:848)

at org.apache.spark.sql.DataFrameWriter.saveToV1Source(DataFrameWriter.scala:382)

at org.apache.spark.sql.DataFrameWriter.saveInternal(DataFrameWriter.scala:355)

at org.apache.spark.sql.DataFrameWriter.save(DataFrameWriter.scala:239)

at org.apache.spark.sql.DataFrameWriter.json(DataFrameWriter.scala:763)

at sun.reflect.NativeMethodAccessorImpl.invoke0(Native Method)

at sun.reflect.NativeMethodAccessorImpl.invoke(NativeMethodAccessorImpl.java:62)

at sun.reflect.DelegatingMethodAccessorImpl.invoke(DelegatingMethodAccessorImpl.java:43)

at java.lang.reflect.Method.invoke(Method.java:498)

at py4j.reflection.MethodInvoker.invoke(MethodInvoker.java:244)

at py4j.reflection.ReflectionEngine.invoke(ReflectionEngine.java:357)

at py4j.Gateway.invoke(Gateway.java:282)

at py4j.commands.AbstractCommand.invokeMethod(AbstractCommand.java:132)

at py4j.commands.CallCommand.execute(CallCommand.java:79)

at py4j.ClientServerConnection.waitForCommands(ClientServerConnection.java:182)

at py4j.ClientServerConnection.run(ClientServerConnection.java:106)

at java.lang.Thread.run(Thread.java:750)

Caused by: org.apache.spark.api.python.PythonException: Traceback (most recent call last):

File "/usr/local/spark/python/lib/pyspark.zip/pyspark/worker.py", line 619, in main

process()

File "/usr/local/spark/python/lib/pyspark.zip/pyspark/worker.py", line 611, in process

serializer.dump\_stream(out\_iter, outfile)

File "/usr/local/spark/python/lib/pyspark.zip/pyspark/serializers.py", line 211, in dump\_stream

self.serializer.dump\_stream(self.\_batched(iterator), stream)

File "/usr/local/spark/python/lib/pyspark.zip/pyspark/serializers.py", line 132, in dump\_stream

for obj in iterator:

File "/usr/local/spark/python/lib/pyspark.zip/pyspark/serializers.py", line 200, in \_batched

for item in iterator:

File "/usr/local/spark/python/lib/pyspark.zip/pyspark/worker.py", line 452, in mapper

result = tuple(f(\*[a[o] for o in arg\_offsets]) for (arg\_offsets, f) in udfs)

File "/usr/local/spark/python/lib/pyspark.zip/pyspark/worker.py", line 452, in <genexpr>

result = tuple(f(\*[a[o] for o in arg\_offsets]) for (arg\_offsets, f) in udfs)

File "/usr/local/spark/python/lib/pyspark.zip/pyspark/worker.py", line 87, in <lambda>

return lambda \*a: f(\*a)

File "/usr/local/spark/python/lib/pyspark.zip/pyspark/util.py", line 74, in wrapper

return f(\*args, \*\*kwargs)

File "<stdin>", line 3, in clean\_string

TypeError: expected string or bytes-like object

at org.apache.spark.api.python.BasePythonRunner$ReaderIterator.handlePythonException(PythonRunner.scala:555)

at org.apache.spark.sql.execution.python.PythonUDFRunner$$anon$2.read(PythonUDFRunner.scala:86)

at org.apache.spark.sql.execution.python.PythonUDFRunner$$anon$2.read(PythonUDFRunner.scala:68)

at org.apache.spark.api.python.BasePythonRunner$ReaderIterator.hasNext(PythonRunner.scala:508)

at org.apache.spark.InterruptibleIterator.hasNext(InterruptibleIterator.scala:37)

at scala.collection.Iterator$$anon$11.hasNext(Iterator.scala:491)

at scala.collection.Iterator$$anon$10.hasNext(Iterator.scala:460)

at scala.collection.Iterator$$anon$10.hasNext(Iterator.scala:460)

at org.apache.spark.sql.catalyst.expressions.GeneratedClass$GeneratedIteratorForCodegenStage1.processNext(Unknown Source)

at org.apache.spark.sql.execution.BufferedRowIterator.hasNext(BufferedRowIterator.java:43)

at org.apache.spark.sql.execution.WholeStageCodegenExec$$anon$1.hasNext(WholeStageCodegenExec.scala:759)

at org.apache.spark.sql.execution.columnar.DefaultCachedBatchSerializer$$anon$1.next(InMemoryRelation.scala:87)

at org.apache.spark.sql.execution.columnar.DefaultCachedBatchSerializer$$anon$1.next(InMemoryRelation.scala:79)

at scala.collection.Iterator$$anon$10.next(Iterator.scala:461)

at org.apache.spark.storage.memory.MemoryStore.putIterator(MemoryStore.scala:224)

at org.apache.spark.storage.memory.MemoryStore.putIteratorAsValues(MemoryStore.scala:302)

at org.apache.spark.storage.BlockManager.$anonfun$doPutIterator$1(BlockManager.scala:1481)

at org.apache.spark.storage.BlockManager.org$apache$spark$storage$BlockManager$$doPut(BlockManager.scala:1408)

at org.apache.spark.storage.BlockManager.doPutIterator(BlockManager.scala:1472)

at org.apache.spark.storage.BlockManager.getOrElseUpdate(BlockManager.scala:1295)

at org.apache.spark.rdd.RDD.getOrCompute(RDD.scala:384)

at org.apache.spark.rdd.RDD.iterator(RDD.scala:335)

at org.apache.spark.rdd.MapPartitionsRDD.compute(MapPartitionsRDD.scala:52)

at org.apache.spark.rdd.RDD.computeOrReadCheckpoint(RDD.scala:373)

at org.apache.spark.rdd.RDD.iterator(RDD.scala:337)

at org.apache.spark.rdd.MapPartitionsRDD.compute(MapPartitionsRDD.scala:52)

at org.apache.spark.rdd.RDD.computeOrReadCheckpoint(RDD.scala:373)

at org.apache.spark.rdd.RDD.iterator(RDD.scala:337)

at org.apache.spark.rdd.MapPartitionsRDD.compute(MapPartitionsRDD.scala:52)

at org.apache.spark.rdd.RDD.computeOrReadCheckpoint(RDD.scala:373)

at org.apache.spark.rdd.RDD.iterator(RDD.scala:337)

at org.apache.spark.scheduler.ResultTask.runTask(ResultTask.scala:90)

at org.apache.spark.scheduler.Task.run(Task.scala:131)

at org.apache.spark.executor.Executor$TaskRunner.$anonfun$run$3(Executor.scala:506)

at org.apache.spark.util.Utils$.tryWithSafeFinally(Utils.scala:1462)

at org.apache.spark.executor.Executor$TaskRunner.run(Executor.scala:509)

at java.util.concurrent.ThreadPoolExecutor.runWorker(ThreadPoolExecutor.java:1149)

at java.util.concurrent.ThreadPoolExecutor$Worker.run(ThreadPoolExecutor.java:624)

... 1 more

24/07/22 09:28:58 WARN scheduler.TaskSetManager: Lost task 1.0 in stage 103.0 (TID 114) (192.168.112.129 executor driver): TaskKilled (Stage cancelled)

Traceback (most recent call last):

File "<stdin>", line 1, in <module>

File "/usr/local/spark/python/pyspark/sql/readwriter.py", line 846, in json

self.\_jwrite.json(path)

File "/usr/local/spark/python/lib/py4j-0.10.9.3-src.zip/py4j/java\_gateway.py", line 1322, in \_\_call\_\_

File "/usr/local/spark/python/pyspark/sql/utils.py", line 111, in deco

return f(\*a, \*\*kw)

File "/usr/local/spark/python/lib/py4j-0.10.9.3-src.zip/py4j/protocol.py", line 328, in get\_return\_value

py4j.protocol.Py4JJavaError: An error occurred while calling o666.json.

: org.apache.spark.SparkException: Job aborted.

at org.apache.spark.sql.errors.QueryExecutionErrors$.jobAbortedError(QueryExecutionErrors.scala:496)

at org.apache.spark.sql.execution.datasources.FileFormatWriter$.write(FileFormatWriter.scala:251)

at org.apache.spark.sql.execution.datasources.InsertIntoHadoopFsRelationCommand.run(InsertIntoHadoopFsRelationCommand.scala:186)

at org.apache.spark.sql.execution.command.DataWritingCommandExec.sideEffectResult$lzycompute(commands.scala:113)

at org.apache.spark.sql.execution.command.DataWritingCommandExec.sideEffectResult(commands.scala:111)

at org.apache.spark.sql.execution.command.DataWritingCommandExec.executeCollect(commands.scala:125)

at org.apache.spark.sql.execution.QueryExecution$$anonfun$eagerlyExecuteCommands$1.$anonfun$applyOrElse$1(QueryExecution.scala:110)

at org.apache.spark.sql.execution.SQLExecution$.$anonfun$withNewExecutionId$5(SQLExecution.scala:103)

at org.apache.spark.sql.execution.SQLExecution$.withSQLConfPropagated(SQLExecution.scala:163)

at org.apache.spark.sql.execution.SQLExecution$.$anonfun$withNewExecutionId$1(SQLExecution.scala:90)

at org.apache.spark.sql.SparkSession.withActive(SparkSession.scala:775)

at org.apache.spark.sql.execution.SQLExecution$.withNewExecutionId(SQLExecution.scala:64)

at org.apache.spark.sql.execution.QueryExecution$$anonfun$eagerlyExecuteCommands$1.applyOrElse(QueryExecution.scala:110)

at org.apache.spark.sql.execution.QueryExecution$$anonfun$eagerlyExecuteCommands$1.applyOrElse(QueryExecution.scala:106)

at org.apache.spark.sql.catalyst.trees.TreeNode.$anonfun$transformDownWithPruning$1(TreeNode.scala:481)

at org.apache.spark.sql.catalyst.trees.CurrentOrigin$.withOrigin(TreeNode.scala:82)

at org.apache.spark.sql.catalyst.trees.TreeNode.transformDownWithPruning(TreeNode.scala:481)

at org.apache.spark.sql.catalyst.plans.logical.LogicalPlan.org$apache$spark$sql$catalyst$plans$logical$AnalysisHelper$$super$transformDownWithPruning(LogicalPlan.scala:30)

at org.apache.spark.sql.catalyst.plans.logical.AnalysisHelper.transformDownWithPruning(AnalysisHelper.scala:267)

at org.apache.spark.sql.catalyst.plans.logical.AnalysisHelper.transformDownWithPruning$(AnalysisHelper.scala:263)

at org.apache.spark.sql.catalyst.plans.logical.LogicalPlan.transformDownWithPruning(LogicalPlan.scala:30)

at org.apache.spark.sql.catalyst.plans.logical.LogicalPlan.transformDownWithPruning(LogicalPlan.scala:30)

at org.apache.spark.sql.catalyst.trees.TreeNode.transformDown(TreeNode.scala:457)

at org.apache.spark.sql.execution.QueryExecution.eagerlyExecuteCommands(QueryExecution.scala:106)

at org.apache.spark.sql.execution.QueryExecution.commandExecuted$lzycompute(QueryExecution.scala:93)

at org.apache.spark.sql.execution.QueryExecution.commandExecuted(QueryExecution.scala:91)

at org.apache.spark.sql.execution.QueryExecution.assertCommandExecuted(QueryExecution.scala:128)

at org.apache.spark.sql.DataFrameWriter.runCommand(DataFrameWriter.scala:848)

at org.apache.spark.sql.DataFrameWriter.saveToV1Source(DataFrameWriter.scala:382)

at org.apache.spark.sql.DataFrameWriter.saveInternal(DataFrameWriter.scala:355)

at org.apache.spark.sql.DataFrameWriter.save(DataFrameWriter.scala:239)

at org.apache.spark.sql.DataFrameWriter.json(DataFrameWriter.scala:763)

at sun.reflect.NativeMethodAccessorImpl.invoke0(Native Method)

at sun.reflect.NativeMethodAccessorImpl.invoke(NativeMethodAccessorImpl.java:62)

at sun.reflect.DelegatingMethodAccessorImpl.invoke(DelegatingMethodAccessorImpl.java:43)

at java.lang.reflect.Method.invoke(Method.java:498)

at py4j.reflection.MethodInvoker.invoke(MethodInvoker.java:244)

at py4j.reflection.ReflectionEngine.invoke(ReflectionEngine.java:357)

at py4j.Gateway.invoke(Gateway.java:282)

at py4j.commands.AbstractCommand.invokeMethod(AbstractCommand.java:132)

at py4j.commands.CallCommand.execute(CallCommand.java:79)

at py4j.ClientServerConnection.waitForCommands(ClientServerConnection.java:182)

at py4j.ClientServerConnection.run(ClientServerConnection.java:106)

at java.lang.Thread.run(Thread.java:750)

Caused by: org.apache.spark.SparkException: Job aborted due to stage failure: Task 0 in stage 103.0 failed 1 times, most recent failure: Lost task 0.0 in stage 103.0 (TID 113) (192.168.112.129 executor driver): org.apache.spark.api.python.PythonException: Traceback (most recent call last):

File "/usr/local/spark/python/lib/pyspark.zip/pyspark/worker.py", line 619, in main

process()

File "/usr/local/spark/python/lib/pyspark.zip/pyspark/worker.py", line 611, in process

serializer.dump\_stream(out\_iter, outfile)

File "/usr/local/spark/python/lib/pyspark.zip/pyspark/serializers.py", line 211, in dump\_stream

self.serializer.dump\_stream(self.\_batched(iterator), stream)

File "/usr/local/spark/python/lib/pyspark.zip/pyspark/serializers.py", line 132, in dump\_stream

for obj in iterator:

File "/usr/local/spark/python/lib/pyspark.zip/pyspark/serializers.py", line 200, in \_batched

for item in iterator:

File "/usr/local/spark/python/lib/pyspark.zip/pyspark/worker.py", line 452, in mapper

result = tuple(f(\*[a[o] for o in arg\_offsets]) for (arg\_offsets, f) in udfs)

File "/usr/local/spark/python/lib/pyspark.zip/pyspark/worker.py", line 452, in <genexpr>

result = tuple(f(\*[a[o] for o in arg\_offsets]) for (arg\_offsets, f) in udfs)

File "/usr/local/spark/python/lib/pyspark.zip/pyspark/worker.py", line 87, in <lambda>

return lambda \*a: f(\*a)

File "/usr/local/spark/python/lib/pyspark.zip/pyspark/util.py", line 74, in wrapper

return f(\*args, \*\*kwargs)

File "<stdin>", line 3, in clean\_string

TypeError: expected string or bytes-like object

at org.apache.spark.api.python.BasePythonRunner$ReaderIterator.handlePythonException(PythonRunner.scala:555)

at org.apache.spark.sql.execution.python.PythonUDFRunner$$anon$2.read(PythonUDFRunner.scala:86)

at org.apache.spark.sql.execution.python.PythonUDFRunner$$anon$2.read(PythonUDFRunner.scala:68)

at org.apache.spark.api.python.BasePythonRunner$ReaderIterator.hasNext(PythonRunner.scala:508)

at org.apache.spark.InterruptibleIterator.hasNext(InterruptibleIterator.scala:37)

at scala.collection.Iterator$$anon$11.hasNext(Iterator.scala:491)

at scala.collection.Iterator$$anon$10.hasNext(Iterator.scala:460)

at scala.collection.Iterator$$anon$10.hasNext(Iterator.scala:460)

at org.apache.spark.sql.catalyst.expressions.GeneratedClass$GeneratedIteratorForCodegenStage1.processNext(Unknown Source)

at org.apache.spark.sql.execution.BufferedRowIterator.hasNext(BufferedRowIterator.java:43)

at org.apache.spark.sql.execution.WholeStageCodegenExec$$anon$1.hasNext(WholeStageCodegenExec.scala:759)

at org.apache.spark.sql.execution.columnar.DefaultCachedBatchSerializer$$anon$1.next(InMemoryRelation.scala:87)

at org.apache.spark.sql.execution.columnar.DefaultCachedBatchSerializer$$anon$1.next(InMemoryRelation.scala:79)

at scala.collection.Iterator$$anon$10.next(Iterator.scala:461)

at org.apache.spark.storage.memory.MemoryStore.putIterator(MemoryStore.scala:224)

at org.apache.spark.storage.memory.MemoryStore.putIteratorAsValues(MemoryStore.scala:302)

at org.apache.spark.storage.BlockManager.$anonfun$doPutIterator$1(BlockManager.scala:1481)

at org.apache.spark.storage.BlockManager.org$apache$spark$storage$BlockManager$$doPut(BlockManager.scala:1408)

at org.apache.spark.storage.BlockManager.doPutIterator(BlockManager.scala:1472)

at org.apache.spark.storage.BlockManager.getOrElseUpdate(BlockManager.scala:1295)

at org.apache.spark.rdd.RDD.getOrCompute(RDD.scala:384)

at org.apache.spark.rdd.RDD.iterator(RDD.scala:335)

at org.apache.spark.rdd.MapPartitionsRDD.compute(MapPartitionsRDD.scala:52)

at org.apache.spark.rdd.RDD.computeOrReadCheckpoint(RDD.scala:373)

at org.apache.spark.rdd.RDD.iterator(RDD.scala:337)

at org.apache.spark.rdd.MapPartitionsRDD.compute(MapPartitionsRDD.scala:52)

at org.apache.spark.rdd.RDD.computeOrReadCheckpoint(RDD.scala:373)

at org.apache.spark.rdd.RDD.iterator(RDD.scala:337)

at org.apache.spark.rdd.MapPartitionsRDD.compute(MapPartitionsRDD.scala:52)

at org.apache.spark.rdd.RDD.computeOrReadCheckpoint(RDD.scala:373)

at org.apache.spark.rdd.RDD.iterator(RDD.scala:337)

at org.apache.spark.scheduler.ResultTask.runTask(ResultTask.scala:90)

at org.apache.spark.scheduler.Task.run(Task.scala:131)

at org.apache.spark.executor.Executor$TaskRunner.$anonfun$run$3(Executor.scala:506)

at org.apache.spark.util.Utils$.tryWithSafeFinally(Utils.scala:1462)

at org.apache.spark.executor.Executor$TaskRunner.run(Executor.scala:509)

at java.util.concurrent.ThreadPoolExecutor.runWorker(ThreadPoolExecutor.java:1149)

at java.util.concurrent.ThreadPoolExecutor$Worker.run(ThreadPoolExecutor.java:624)

at java.lang.Thread.run(Thread.java:750)

Driver stacktrace:

at org.apache.spark.scheduler.DAGScheduler.failJobAndIndependentStages(DAGScheduler.scala:2454)

at org.apache.spark.scheduler.DAGScheduler.$anonfun$abortStage$2(DAGScheduler.scala:2403)

at org.apache.spark.scheduler.DAGScheduler.$anonfun$abortStage$2$adapted(DAGScheduler.scala:2402)

at scala.collection.mutable.ResizableArray.foreach(ResizableArray.scala:62)

at scala.collection.mutable.ResizableArray.foreach$(ResizableArray.scala:55)

at scala.collection.mutable.ArrayBuffer.foreach(ArrayBuffer.scala:49)

at org.apache.spark.scheduler.DAGScheduler.abortStage(DAGScheduler.scala:2402)

at org.apache.spark.scheduler.DAGScheduler.$anonfun$handleTaskSetFailed$1(DAGScheduler.scala:1160)

at org.apache.spark.scheduler.DAGScheduler.$anonfun$handleTaskSetFailed$1$adapted(DAGScheduler.scala:1160)

at scala.Option.foreach(Option.scala:407)

at org.apache.spark.scheduler.DAGScheduler.handleTaskSetFailed(DAGScheduler.scala:1160)

at org.apache.spark.scheduler.DAGSchedulerEventProcessLoop.doOnReceive(DAGScheduler.scala:2642)

at org.apache.spark.scheduler.DAGSchedulerEventProcessLoop.onReceive(DAGScheduler.scala:2584)

at org.apache.spark.scheduler.DAGSchedulerEventProcessLoop.onReceive(DAGScheduler.scala:2573)

at org.apache.spark.util.EventLoop$$anon$1.run(EventLoop.scala:49)

at org.apache.spark.scheduler.DAGScheduler.runJob(DAGScheduler.scala:938)

at org.apache.spark.SparkContext.runJob(SparkContext.scala:2214)

at org.apache.spark.sql.execution.datasources.FileFormatWriter$.write(FileFormatWriter.scala:218)

... 42 more

Caused by: org.apache.spark.api.python.PythonException: Traceback (most recent call last):

File "/usr/local/spark/python/lib/pyspark.zip/pyspark/worker.py", line 619, in main

process()

File "/usr/local/spark/python/lib/pyspark.zip/pyspark/worker.py", line 611, in process

serializer.dump\_stream(out\_iter, outfile)

File "/usr/local/spark/python/lib/pyspark.zip/pyspark/serializers.py", line 211, in dump\_stream

self.serializer.dump\_stream(self.\_batched(iterator), stream)

File "/usr/local/spark/python/lib/pyspark.zip/pyspark/serializers.py", line 132, in dump\_stream

for obj in iterator:

File "/usr/local/spark/python/lib/pyspark.zip/pyspark/serializers.py", line 200, in \_batched

for item in iterator:

File "/usr/local/spark/python/lib/pyspark.zip/pyspark/worker.py", line 452, in mapper

result = tuple(f(\*[a[o] for o in arg\_offsets]) for (arg\_offsets, f) in udfs)

File "/usr/local/spark/python/lib/pyspark.zip/pyspark/worker.py", line 452, in <genexpr>

result = tuple(f(\*[a[o] for o in arg\_offsets]) for (arg\_offsets, f) in udfs)

File "/usr/local/spark/python/lib/pyspark.zip/pyspark/worker.py", line 87, in <lambda>

return lambda \*a: f(\*a)

File "/usr/local/spark/python/lib/pyspark.zip/pyspark/util.py", line 74, in wrapper

return f(\*args, \*\*kwargs)

File "<stdin>", line 3, in clean\_string

TypeError: expected string or bytes-like object

at org.apache.spark.api.python.BasePythonRunner$ReaderIterator.handlePythonException(PythonRunner.scala:555)

at org.apache.spark.sql.execution.python.PythonUDFRunner$$anon$2.read(PythonUDFRunner.scala:86)

at org.apache.spark.sql.execution.python.PythonUDFRunner$$anon$2.read(PythonUDFRunner.scala:68)

at org.apache.spark.api.python.BasePythonRunner$ReaderIterator.hasNext(PythonRunner.scala:508)

at org.apache.spark.InterruptibleIterator.hasNext(InterruptibleIterator.scala:37)

at scala.collection.Iterator$$anon$11.hasNext(Iterator.scala:491)

at scala.collection.Iterator$$anon$10.hasNext(Iterator.scala:460)

at scala.collection.Iterator$$anon$10.hasNext(Iterator.scala:460)

at org.apache.spark.sql.catalyst.expressions.GeneratedClass$GeneratedIteratorForCodegenStage1.processNext(Unknown Source)

at org.apache.spark.sql.execution.BufferedRowIterator.hasNext(BufferedRowIterator.java:43)

at org.apache.spark.sql.execution.WholeStageCodegenExec$$anon$1.hasNext(WholeStageCodegenExec.scala:759)

at org.apache.spark.sql.execution.columnar.DefaultCachedBatchSerializer$$anon$1.next(InMemoryRelation.scala:87)

at org.apache.spark.sql.execution.columnar.DefaultCachedBatchSerializer$$anon$1.next(InMemoryRelation.scala:79)

at scala.collection.Iterator$$anon$10.next(Iterator.scala:461)

at org.apache.spark.storage.memory.MemoryStore.putIterator(MemoryStore.scala:224)

at org.apache.spark.storage.memory.MemoryStore.putIteratorAsValues(MemoryStore.scala:302)

at org.apache.spark.storage.BlockManager.$anonfun$doPutIterator$1(BlockManager.scala:1481)

at org.apache.spark.storage.BlockManager.org$apache$spark$storage$BlockManager$$doPut(BlockManager.scala:1408)

at org.apache.spark.storage.BlockManager.doPutIterator(BlockManager.scala:1472)

at org.apache.spark.storage.BlockManager.getOrElseUpdate(BlockManager.scala:1295)

at org.apache.spark.rdd.RDD.getOrCompute(RDD.scala:384)

at org.apache.spark.rdd.RDD.iterator(RDD.scala:335)

at org.apache.spark.rdd.MapPartitionsRDD.compute(MapPartitionsRDD.scala:52)

at org.apache.spark.rdd.RDD.computeOrReadCheckpoint(RDD.scala:373)

at org.apache.spark.rdd.RDD.iterator(RDD.scala:337)

at org.apache.spark.rdd.MapPartitionsRDD.compute(MapPartitionsRDD.scala:52)

at org.apache.spark.rdd.RDD.computeOrReadCheckpoint(RDD.scala:373)

at org.apache.spark.rdd.RDD.iterator(RDD.scala:337)

at org.apache.spark.rdd.MapPartitionsRDD.compute(MapPartitionsRDD.scala:52)

at org.apache.spark.rdd.RDD.computeOrReadCheckpoint(RDD.scala:373)

at org.apache.spark.rdd.RDD.iterator(RDD.scala:337)

at org.apache.spark.scheduler.ResultTask.runTask(ResultTask.scala:90)

at org.apache.spark.scheduler.Task.run(Task.scala:131)

at org.apache.spark.executor.Executor$TaskRunner.$anonfun$run$3(Executor.scala:506)

at org.apache.spark.util.Utils$.tryWithSafeFinally(Utils.scala:1462)

at org.apache.spark.executor.Executor$TaskRunner.run(Executor.scala:509)

at java.util.concurrent.ThreadPoolExecutor.runWorker(ThreadPoolExecutor.java:1149)

at java.util.concurrent.ThreadPoolExecutor$Worker.run(ThreadPoolExecutor.java:624)

... 1 more

>>> sample\_op.show()

24/07/22 09:31:43 WARN storage.BlockManager: Putting block rdd\_531\_0 failed due to exception org.apache.spark.api.python.PythonException: Traceback (most recent call last):

File "/usr/local/spark/python/lib/pyspark.zip/pyspark/worker.py", line 619, in main

process()

File "/usr/local/spark/python/lib/pyspark.zip/pyspark/worker.py", line 611, in process

serializer.dump\_stream(out\_iter, outfile)

File "/usr/local/spark/python/lib/pyspark.zip/pyspark/serializers.py", line 211, in dump\_stream

self.serializer.dump\_stream(self.\_batched(iterator), stream)

File "/usr/local/spark/python/lib/pyspark.zip/pyspark/serializers.py", line 132, in dump\_stream

for obj in iterator:

File "/usr/local/spark/python/lib/pyspark.zip/pyspark/serializers.py", line 200, in \_batched

for item in iterator:

File "/usr/local/spark/python/lib/pyspark.zip/pyspark/worker.py", line 452, in mapper

result = tuple(f(\*[a[o] for o in arg\_offsets]) for (arg\_offsets, f) in udfs)

File "/usr/local/spark/python/lib/pyspark.zip/pyspark/worker.py", line 452, in <genexpr>

result = tuple(f(\*[a[o] for o in arg\_offsets]) for (arg\_offsets, f) in udfs)

File "/usr/local/spark/python/lib/pyspark.zip/pyspark/worker.py", line 87, in <lambda>

return lambda \*a: f(\*a)

File "/usr/local/spark/python/lib/pyspark.zip/pyspark/util.py", line 74, in wrapper

return f(\*args, \*\*kwargs)

File "<stdin>", line 3, in clean\_string

TypeError: expected string or bytes-like object

.

24/07/22 09:31:43 WARN storage.BlockManager: Block rdd\_531\_0 could not be removed as it was not found on disk or in memory

24/07/22 09:31:43 ERROR executor.Executor: Exception in task 0.0 in stage 104.0 (TID 115)

org.apache.spark.api.python.PythonException: Traceback (most recent call last):

File "/usr/local/spark/python/lib/pyspark.zip/pyspark/worker.py", line 619, in main

process()

File "/usr/local/spark/python/lib/pyspark.zip/pyspark/worker.py", line 611, in process

serializer.dump\_stream(out\_iter, outfile)

File "/usr/local/spark/python/lib/pyspark.zip/pyspark/serializers.py", line 211, in dump\_stream

self.serializer.dump\_stream(self.\_batched(iterator), stream)

File "/usr/local/spark/python/lib/pyspark.zip/pyspark/serializers.py", line 132, in dump\_stream

for obj in iterator:

File "/usr/local/spark/python/lib/pyspark.zip/pyspark/serializers.py", line 200, in \_batched

for item in iterator:

File "/usr/local/spark/python/lib/pyspark.zip/pyspark/worker.py", line 452, in mapper

result = tuple(f(\*[a[o] for o in arg\_offsets]) for (arg\_offsets, f) in udfs)

File "/usr/local/spark/python/lib/pyspark.zip/pyspark/worker.py", line 452, in <genexpr>

result = tuple(f(\*[a[o] for o in arg\_offsets]) for (arg\_offsets, f) in udfs)

File "/usr/local/spark/python/lib/pyspark.zip/pyspark/worker.py", line 87, in <lambda>

return lambda \*a: f(\*a)

File "/usr/local/spark/python/lib/pyspark.zip/pyspark/util.py", line 74, in wrapper

return f(\*args, \*\*kwargs)

File "<stdin>", line 3, in clean\_string

TypeError: expected string or bytes-like object

at org.apache.spark.api.python.BasePythonRunner$ReaderIterator.handlePythonException(PythonRunner.scala:555)

at org.apache.spark.sql.execution.python.PythonUDFRunner$$anon$2.read(PythonUDFRunner.scala:86)

at org.apache.spark.sql.execution.python.PythonUDFRunner$$anon$2.read(PythonUDFRunner.scala:68)

at org.apache.spark.api.python.BasePythonRunner$ReaderIterator.hasNext(PythonRunner.scala:508)

at org.apache.spark.InterruptibleIterator.hasNext(InterruptibleIterator.scala:37)

at scala.collection.Iterator$$anon$11.hasNext(Iterator.scala:491)

at scala.collection.Iterator$$anon$10.hasNext(Iterator.scala:460)

at scala.collection.Iterator$$anon$10.hasNext(Iterator.scala:460)

at org.apache.spark.sql.catalyst.expressions.GeneratedClass$GeneratedIteratorForCodegenStage1.processNext(Unknown Source)

at org.apache.spark.sql.execution.BufferedRowIterator.hasNext(BufferedRowIterator.java:43)

at org.apache.spark.sql.execution.WholeStageCodegenExec$$anon$1.hasNext(WholeStageCodegenExec.scala:759)

at org.apache.spark.sql.execution.columnar.DefaultCachedBatchSerializer$$anon$1.next(InMemoryRelation.scala:87)

at org.apache.spark.sql.execution.columnar.DefaultCachedBatchSerializer$$anon$1.next(InMemoryRelation.scala:79)

at scala.collection.Iterator$$anon$10.next(Iterator.scala:461)

at org.apache.spark.storage.memory.MemoryStore.putIterator(MemoryStore.scala:224)

at org.apache.spark.storage.memory.MemoryStore.putIteratorAsValues(MemoryStore.scala:302)

at org.apache.spark.storage.BlockManager.$anonfun$doPutIterator$1(BlockManager.scala:1481)

at org.apache.spark.storage.BlockManager.org$apache$spark$storage$BlockManager$$doPut(BlockManager.scala:1408)

at org.apache.spark.storage.BlockManager.doPutIterator(BlockManager.scala:1472)

at org.apache.spark.storage.BlockManager.getOrElseUpdate(BlockManager.scala:1295)

at org.apache.spark.rdd.RDD.getOrCompute(RDD.scala:384)

at org.apache.spark.rdd.RDD.iterator(RDD.scala:335)

at org.apache.spark.rdd.MapPartitionsRDD.compute(MapPartitionsRDD.scala:52)

at org.apache.spark.rdd.RDD.computeOrReadCheckpoint(RDD.scala:373)

at org.apache.spark.rdd.RDD.iterator(RDD.scala:337)

at org.apache.spark.rdd.MapPartitionsRDD.compute(MapPartitionsRDD.scala:52)

at org.apache.spark.rdd.RDD.computeOrReadCheckpoint(RDD.scala:373)

at org.apache.spark.rdd.RDD.iterator(RDD.scala:337)

at org.apache.spark.rdd.MapPartitionsRDD.compute(MapPartitionsRDD.scala:52)

at org.apache.spark.rdd.RDD.computeOrReadCheckpoint(RDD.scala:373)

at org.apache.spark.rdd.RDD.iterator(RDD.scala:337)

at org.apache.spark.rdd.MapPartitionsRDD.compute(MapPartitionsRDD.scala:52)

at org.apache.spark.rdd.RDD.computeOrReadCheckpoint(RDD.scala:373)

at org.apache.spark.rdd.RDD.iterator(RDD.scala:337)

at org.apache.spark.rdd.MapPartitionsRDD.compute(MapPartitionsRDD.scala:52)

at org.apache.spark.rdd.RDD.computeOrReadCheckpoint(RDD.scala:373)

at org.apache.spark.rdd.RDD.iterator(RDD.scala:337)

at org.apache.spark.scheduler.ResultTask.runTask(ResultTask.scala:90)

at org.apache.spark.scheduler.Task.run(Task.scala:131)

at org.apache.spark.executor.Executor$TaskRunner.$anonfun$run$3(Executor.scala:506)

at org.apache.spark.util.Utils$.tryWithSafeFinally(Utils.scala:1462)

at org.apache.spark.executor.Executor$TaskRunner.run(Executor.scala:509)

at java.util.concurrent.ThreadPoolExecutor.runWorker(ThreadPoolExecutor.java:1149)

at java.util.concurrent.ThreadPoolExecutor$Worker.run(ThreadPoolExecutor.java:624)

at java.lang.Thread.run(Thread.java:750)

24/07/22 09:31:43 WARN scheduler.TaskSetManager: Lost task 0.0 in stage 104.0 (TID 115) (192.168.112.129 executor driver): org.apache.spark.api.python.PythonException: Traceback (most recent call last):

File "/usr/local/spark/python/lib/pyspark.zip/pyspark/worker.py", line 619, in main

process()

File "/usr/local/spark/python/lib/pyspark.zip/pyspark/worker.py", line 611, in process

serializer.dump\_stream(out\_iter, outfile)

File "/usr/local/spark/python/lib/pyspark.zip/pyspark/serializers.py", line 211, in dump\_stream

self.serializer.dump\_stream(self.\_batched(iterator), stream)

File "/usr/local/spark/python/lib/pyspark.zip/pyspark/serializers.py", line 132, in dump\_stream

for obj in iterator:

File "/usr/local/spark/python/lib/pyspark.zip/pyspark/serializers.py", line 200, in \_batched

for item in iterator:

File "/usr/local/spark/python/lib/pyspark.zip/pyspark/worker.py", line 452, in mapper

result = tuple(f(\*[a[o] for o in arg\_offsets]) for (arg\_offsets, f) in udfs)

File "/usr/local/spark/python/lib/pyspark.zip/pyspark/worker.py", line 452, in <genexpr>

result = tuple(f(\*[a[o] for o in arg\_offsets]) for (arg\_offsets, f) in udfs)

File "/usr/local/spark/python/lib/pyspark.zip/pyspark/worker.py", line 87, in <lambda>

return lambda \*a: f(\*a)

File "/usr/local/spark/python/lib/pyspark.zip/pyspark/util.py", line 74, in wrapper

return f(\*args, \*\*kwargs)

File "<stdin>", line 3, in clean\_string

TypeError: expected string or bytes-like object

at org.apache.spark.api.python.BasePythonRunner$ReaderIterator.handlePythonException(PythonRunner.scala:555)

at org.apache.spark.sql.execution.python.PythonUDFRunner$$anon$2.read(PythonUDFRunner.scala:86)

at org.apache.spark.sql.execution.python.PythonUDFRunner$$anon$2.read(PythonUDFRunner.scala:68)

at org.apache.spark.api.python.BasePythonRunner$ReaderIterator.hasNext(PythonRunner.scala:508)

at org.apache.spark.InterruptibleIterator.hasNext(InterruptibleIterator.scala:37)

at scala.collection.Iterator$$anon$11.hasNext(Iterator.scala:491)

at scala.collection.Iterator$$anon$10.hasNext(Iterator.scala:460)

at scala.collection.Iterator$$anon$10.hasNext(Iterator.scala:460)

at org.apache.spark.sql.catalyst.expressions.GeneratedClass$GeneratedIteratorForCodegenStage1.processNext(Unknown Source)

at org.apache.spark.sql.execution.BufferedRowIterator.hasNext(BufferedRowIterator.java:43)

at org.apache.spark.sql.execution.WholeStageCodegenExec$$anon$1.hasNext(WholeStageCodegenExec.scala:759)

at org.apache.spark.sql.execution.columnar.DefaultCachedBatchSerializer$$anon$1.next(InMemoryRelation.scala:87)

at org.apache.spark.sql.execution.columnar.DefaultCachedBatchSerializer$$anon$1.next(InMemoryRelation.scala:79)

at scala.collection.Iterator$$anon$10.next(Iterator.scala:461)

at org.apache.spark.storage.memory.MemoryStore.putIterator(MemoryStore.scala:224)

at org.apache.spark.storage.memory.MemoryStore.putIteratorAsValues(MemoryStore.scala:302)

at org.apache.spark.storage.BlockManager.$anonfun$doPutIterator$1(BlockManager.scala:1481)

at org.apache.spark.storage.BlockManager.org$apache$spark$storage$BlockManager$$doPut(BlockManager.scala:1408)

at org.apache.spark.storage.BlockManager.doPutIterator(BlockManager.scala:1472)

at org.apache.spark.storage.BlockManager.getOrElseUpdate(BlockManager.scala:1295)

at org.apache.spark.rdd.RDD.getOrCompute(RDD.scala:384)

at org.apache.spark.rdd.RDD.iterator(RDD.scala:335)

at org.apache.spark.rdd.MapPartitionsRDD.compute(MapPartitionsRDD.scala:52)

at org.apache.spark.rdd.RDD.computeOrReadCheckpoint(RDD.scala:373)

at org.apache.spark.rdd.RDD.iterator(RDD.scala:337)

at org.apache.spark.rdd.MapPartitionsRDD.compute(MapPartitionsRDD.scala:52)

at org.apache.spark.rdd.RDD.computeOrReadCheckpoint(RDD.scala:373)

at org.apache.spark.rdd.RDD.iterator(RDD.scala:337)

at org.apache.spark.rdd.MapPartitionsRDD.compute(MapPartitionsRDD.scala:52)

at org.apache.spark.rdd.RDD.computeOrReadCheckpoint(RDD.scala:373)

at org.apache.spark.rdd.RDD.iterator(RDD.scala:337)

at org.apache.spark.rdd.MapPartitionsRDD.compute(MapPartitionsRDD.scala:52)

at org.apache.spark.rdd.RDD.computeOrReadCheckpoint(RDD.scala:373)

at org.apache.spark.rdd.RDD.iterator(RDD.scala:337)

at org.apache.spark.rdd.MapPartitionsRDD.compute(MapPartitionsRDD.scala:52)

at org.apache.spark.rdd.RDD.computeOrReadCheckpoint(RDD.scala:373)

at org.apache.spark.rdd.RDD.iterator(RDD.scala:337)

at org.apache.spark.scheduler.ResultTask.runTask(ResultTask.scala:90)

at org.apache.spark.scheduler.Task.run(Task.scala:131)

at org.apache.spark.executor.Executor$TaskRunner.$anonfun$run$3(Executor.scala:506)

at org.apache.spark.util.Utils$.tryWithSafeFinally(Utils.scala:1462)

at org.apache.spark.executor.Executor$TaskRunner.run(Executor.scala:509)

at java.util.concurrent.ThreadPoolExecutor.runWorker(ThreadPoolExecutor.java:1149)

at java.util.concurrent.ThreadPoolExecutor$Worker.run(ThreadPoolExecutor.java:624)

at java.lang.Thread.run(Thread.java:750)

24/07/22 09:31:43 ERROR scheduler.TaskSetManager: Task 0 in stage 104.0 failed 1 times; aborting job

Traceback (most recent call last):

File "<stdin>", line 1, in <module>

File "/usr/local/spark/python/pyspark/sql/dataframe.py", line 494, in show

print(self.\_jdf.showString(n, 20, vertical))

File "/usr/local/spark/python/lib/py4j-0.10.9.3-src.zip/py4j/java\_gateway.py", line 1322, in \_\_call\_\_

File "/usr/local/spark/python/pyspark/sql/utils.py", line 117, in deco

raise converted from None

pyspark.sql.utils.PythonException:

An exception was thrown from the Python worker. Please see the stack trace below.

Traceback (most recent call last):

File "/usr/local/spark/python/lib/pyspark.zip/pyspark/worker.py", line 619, in main

process()

File "/usr/local/spark/python/lib/pyspark.zip/pyspark/worker.py", line 611, in process

serializer.dump\_stream(out\_iter, outfile)

File "/usr/local/spark/python/lib/pyspark.zip/pyspark/serializers.py", line 211, in dump\_stream

self.serializer.dump\_stream(self.\_batched(iterator), stream)

File "/usr/local/spark/python/lib/pyspark.zip/pyspark/serializers.py", line 132, in dump\_stream

for obj in iterator:

File "/usr/local/spark/python/lib/pyspark.zip/pyspark/serializers.py", line 200, in \_batched

for item in iterator:

File "/usr/local/spark/python/lib/pyspark.zip/pyspark/worker.py", line 452, in mapper

result = tuple(f(\*[a[o] for o in arg\_offsets]) for (arg\_offsets, f) in udfs)

File "/usr/local/spark/python/lib/pyspark.zip/pyspark/worker.py", line 452, in <genexpr>

result = tuple(f(\*[a[o] for o in arg\_offsets]) for (arg\_offsets, f) in udfs)

File "/usr/local/spark/python/lib/pyspark.zip/pyspark/worker.py", line 87, in <lambda>

return lambda \*a: f(\*a)

File "/usr/local/spark/python/lib/pyspark.zip/pyspark/util.py", line 74, in wrapper

return f(\*args, \*\*kwargs)

File "<stdin>", line 3, in clean\_string

TypeError: expected string or bytes-like object

>>> sample\_op.show(3)

24/07/22 09:31:48 WARN storage.BlockManager: Putting block rdd\_531\_0 failed due to exception org.apache.spark.api.python.PythonException: Traceback (most recent call last):

File "/usr/local/spark/python/lib/pyspark.zip/pyspark/worker.py", line 619, in main

process()

File "/usr/local/spark/python/lib/pyspark.zip/pyspark/worker.py", line 611, in process

serializer.dump\_stream(out\_iter, outfile)

File "/usr/local/spark/python/lib/pyspark.zip/pyspark/serializers.py", line 211, in dump\_stream

self.serializer.dump\_stream(self.\_batched(iterator), stream)

File "/usr/local/spark/python/lib/pyspark.zip/pyspark/serializers.py", line 132, in dump\_stream

for obj in iterator:

File "/usr/local/spark/python/lib/pyspark.zip/pyspark/serializers.py", line 200, in \_batched

for item in iterator:

File "/usr/local/spark/python/lib/pyspark.zip/pyspark/worker.py", line 452, in mapper

result = tuple(f(\*[a[o] for o in arg\_offsets]) for (arg\_offsets, f) in udfs)

File "/usr/local/spark/python/lib/pyspark.zip/pyspark/worker.py", line 452, in <genexpr>

result = tuple(f(\*[a[o] for o in arg\_offsets]) for (arg\_offsets, f) in udfs)

File "/usr/local/spark/python/lib/pyspark.zip/pyspark/worker.py", line 87, in <lambda>

return lambda \*a: f(\*a)

File "/usr/local/spark/python/lib/pyspark.zip/pyspark/util.py", line 74, in wrapper

return f(\*args, \*\*kwargs)

File "<stdin>", line 3, in clean\_string

TypeError: expected string or bytes-like object

.

24/07/22 09:31:48 WARN storage.BlockManager: Block rdd\_531\_0 could not be removed as it was not found on disk or in memory

24/07/22 09:31:48 ERROR executor.Executor: Exception in task 0.0 in stage 105.0 (TID 116)

org.apache.spark.api.python.PythonException: Traceback (most recent call last):

File "/usr/local/spark/python/lib/pyspark.zip/pyspark/worker.py", line 619, in main

process()

File "/usr/local/spark/python/lib/pyspark.zip/pyspark/worker.py", line 611, in process

serializer.dump\_stream(out\_iter, outfile)

File "/usr/local/spark/python/lib/pyspark.zip/pyspark/serializers.py", line 211, in dump\_stream

self.serializer.dump\_stream(self.\_batched(iterator), stream)

File "/usr/local/spark/python/lib/pyspark.zip/pyspark/serializers.py", line 132, in dump\_stream

for obj in iterator:

File "/usr/local/spark/python/lib/pyspark.zip/pyspark/serializers.py", line 200, in \_batched

for item in iterator:

File "/usr/local/spark/python/lib/pyspark.zip/pyspark/worker.py", line 452, in mapper

result = tuple(f(\*[a[o] for o in arg\_offsets]) for (arg\_offsets, f) in udfs)

File "/usr/local/spark/python/lib/pyspark.zip/pyspark/worker.py", line 452, in <genexpr>

result = tuple(f(\*[a[o] for o in arg\_offsets]) for (arg\_offsets, f) in udfs)

File "/usr/local/spark/python/lib/pyspark.zip/pyspark/worker.py", line 87, in <lambda>

return lambda \*a: f(\*a)

File "/usr/local/spark/python/lib/pyspark.zip/pyspark/util.py", line 74, in wrapper

return f(\*args, \*\*kwargs)

File "<stdin>", line 3, in clean\_string

TypeError: expected string or bytes-like object

at org.apache.spark.api.python.BasePythonRunner$ReaderIterator.handlePythonException(PythonRunner.scala:555)

at org.apache.spark.sql.execution.python.PythonUDFRunner$$anon$2.read(PythonUDFRunner.scala:86)

at org.apache.spark.sql.execution.python.PythonUDFRunner$$anon$2.read(PythonUDFRunner.scala:68)

at org.apache.spark.api.python.BasePythonRunner$ReaderIterator.hasNext(PythonRunner.scala:508)

at org.apache.spark.InterruptibleIterator.hasNext(InterruptibleIterator.scala:37)

at scala.collection.Iterator$$anon$11.hasNext(Iterator.scala:491)

at scala.collection.Iterator$$anon$10.hasNext(Iterator.scala:460)

at scala.collection.Iterator$$anon$10.hasNext(Iterator.scala:460)

at org.apache.spark.sql.catalyst.expressions.GeneratedClass$GeneratedIteratorForCodegenStage1.processNext(Unknown Source)

at org.apache.spark.sql.execution.BufferedRowIterator.hasNext(BufferedRowIterator.java:43)

at org.apache.spark.sql.execution.WholeStageCodegenExec$$anon$1.hasNext(WholeStageCodegenExec.scala:759)

at org.apache.spark.sql.execution.columnar.DefaultCachedBatchSerializer$$anon$1.next(InMemoryRelation.scala:87)

at org.apache.spark.sql.execution.columnar.DefaultCachedBatchSerializer$$anon$1.next(InMemoryRelation.scala:79)

at scala.collection.Iterator$$anon$10.next(Iterator.scala:461)

at org.apache.spark.storage.memory.MemoryStore.putIterator(MemoryStore.scala:224)

at org.apache.spark.storage.memory.MemoryStore.putIteratorAsValues(MemoryStore.scala:302)

at org.apache.spark.storage.BlockManager.$anonfun$doPutIterator$1(BlockManager.scala:1481)

at org.apache.spark.storage.BlockManager.org$apache$spark$storage$BlockManager$$doPut(BlockManager.scala:1408)

at org.apache.spark.storage.BlockManager.doPutIterator(BlockManager.scala:1472)

at org.apache.spark.storage.BlockManager.getOrElseUpdate(BlockManager.scala:1295)

at org.apache.spark.rdd.RDD.getOrCompute(RDD.scala:384)

at org.apache.spark.rdd.RDD.iterator(RDD.scala:335)

at org.apache.spark.rdd.MapPartitionsRDD.compute(MapPartitionsRDD.scala:52)

at org.apache.spark.rdd.RDD.computeOrReadCheckpoint(RDD.scala:373)

at org.apache.spark.rdd.RDD.iterator(RDD.scala:337)

at org.apache.spark.rdd.MapPartitionsRDD.compute(MapPartitionsRDD.scala:52)

at org.apache.spark.rdd.RDD.computeOrReadCheckpoint(RDD.scala:373)

at org.apache.spark.rdd.RDD.iterator(RDD.scala:337)

at org.apache.spark.rdd.MapPartitionsRDD.compute(MapPartitionsRDD.scala:52)

at org.apache.spark.rdd.RDD.computeOrReadCheckpoint(RDD.scala:373)

at org.apache.spark.rdd.RDD.iterator(RDD.scala:337)

at org.apache.spark.rdd.MapPartitionsRDD.compute(MapPartitionsRDD.scala:52)

at org.apache.spark.rdd.RDD.computeOrReadCheckpoint(RDD.scala:373)

at org.apache.spark.rdd.RDD.iterator(RDD.scala:337)

at org.apache.spark.rdd.MapPartitionsRDD.compute(MapPartitionsRDD.scala:52)

at org.apache.spark.rdd.RDD.computeOrReadCheckpoint(RDD.scala:373)

at org.apache.spark.rdd.RDD.iterator(RDD.scala:337)

at org.apache.spark.scheduler.ResultTask.runTask(ResultTask.scala:90)

at org.apache.spark.scheduler.Task.run(Task.scala:131)

at org.apache.spark.executor.Executor$TaskRunner.$anonfun$run$3(Executor.scala:506)

at org.apache.spark.util.Utils$.tryWithSafeFinally(Utils.scala:1462)

at org.apache.spark.executor.Executor$TaskRunner.run(Executor.scala:509)

at java.util.concurrent.ThreadPoolExecutor.runWorker(ThreadPoolExecutor.java:1149)

at java.util.concurrent.ThreadPoolExecutor$Worker.run(ThreadPoolExecutor.java:624)

at java.lang.Thread.run(Thread.java:750)

24/07/22 09:31:48 WARN scheduler.TaskSetManager: Lost task 0.0 in stage 105.0 (TID 116) (192.168.112.129 executor driver): org.apache.spark.api.python.PythonException: Traceback (most recent call last):

File "/usr/local/spark/python/lib/pyspark.zip/pyspark/worker.py", line 619, in main

process()

File "/usr/local/spark/python/lib/pyspark.zip/pyspark/worker.py", line 611, in process

serializer.dump\_stream(out\_iter, outfile)

File "/usr/local/spark/python/lib/pyspark.zip/pyspark/serializers.py", line 211, in dump\_stream

self.serializer.dump\_stream(self.\_batched(iterator), stream)

File "/usr/local/spark/python/lib/pyspark.zip/pyspark/serializers.py", line 132, in dump\_stream

for obj in iterator:

File "/usr/local/spark/python/lib/pyspark.zip/pyspark/serializers.py", line 200, in \_batched

for item in iterator:

File "/usr/local/spark/python/lib/pyspark.zip/pyspark/worker.py", line 452, in mapper

result = tuple(f(\*[a[o] for o in arg\_offsets]) for (arg\_offsets, f) in udfs)

File "/usr/local/spark/python/lib/pyspark.zip/pyspark/worker.py", line 452, in <genexpr>

result = tuple(f(\*[a[o] for o in arg\_offsets]) for (arg\_offsets, f) in udfs)

File "/usr/local/spark/python/lib/pyspark.zip/pyspark/worker.py", line 87, in <lambda>

return lambda \*a: f(\*a)

File "/usr/local/spark/python/lib/pyspark.zip/pyspark/util.py", line 74, in wrapper

return f(\*args, \*\*kwargs)

File "<stdin>", line 3, in clean\_string

TypeError: expected string or bytes-like object

at org.apache.spark.api.python.BasePythonRunner$ReaderIterator.handlePythonException(PythonRunner.scala:555)

at org.apache.spark.sql.execution.python.PythonUDFRunner$$anon$2.read(PythonUDFRunner.scala:86)

at org.apache.spark.sql.execution.python.PythonUDFRunner$$anon$2.read(PythonUDFRunner.scala:68)

at org.apache.spark.api.python.BasePythonRunner$ReaderIterator.hasNext(PythonRunner.scala:508)

at org.apache.spark.InterruptibleIterator.hasNext(InterruptibleIterator.scala:37)

at scala.collection.Iterator$$anon$11.hasNext(Iterator.scala:491)

at scala.collection.Iterator$$anon$10.hasNext(Iterator.scala:460)

at scala.collection.Iterator$$anon$10.hasNext(Iterator.scala:460)

at org.apache.spark.sql.catalyst.expressions.GeneratedClass$GeneratedIteratorForCodegenStage1.processNext(Unknown Source)

at org.apache.spark.sql.execution.BufferedRowIterator.hasNext(BufferedRowIterator.java:43)

at org.apache.spark.sql.execution.WholeStageCodegenExec$$anon$1.hasNext(WholeStageCodegenExec.scala:759)

at org.apache.spark.sql.execution.columnar.DefaultCachedBatchSerializer$$anon$1.next(InMemoryRelation.scala:87)

at org.apache.spark.sql.execution.columnar.DefaultCachedBatchSerializer$$anon$1.next(InMemoryRelation.scala:79)

at scala.collection.Iterator$$anon$10.next(Iterator.scala:461)

at org.apache.spark.storage.memory.MemoryStore.putIterator(MemoryStore.scala:224)

at org.apache.spark.storage.memory.MemoryStore.putIteratorAsValues(MemoryStore.scala:302)

at org.apache.spark.storage.BlockManager.$anonfun$doPutIterator$1(BlockManager.scala:1481)

at org.apache.spark.storage.BlockManager.org$apache$spark$storage$BlockManager$$doPut(BlockManager.scala:1408)

at org.apache.spark.storage.BlockManager.doPutIterator(BlockManager.scala:1472)

at org.apache.spark.storage.BlockManager.getOrElseUpdate(BlockManager.scala:1295)

at org.apache.spark.rdd.RDD.getOrCompute(RDD.scala:384)

at org.apache.spark.rdd.RDD.iterator(RDD.scala:335)

at org.apache.spark.rdd.MapPartitionsRDD.compute(MapPartitionsRDD.scala:52)

at org.apache.spark.rdd.RDD.computeOrReadCheckpoint(RDD.scala:373)

at org.apache.spark.rdd.RDD.iterator(RDD.scala:337)

at org.apache.spark.rdd.MapPartitionsRDD.compute(MapPartitionsRDD.scala:52)

at org.apache.spark.rdd.RDD.computeOrReadCheckpoint(RDD.scala:373)

at org.apache.spark.rdd.RDD.iterator(RDD.scala:337)

at org.apache.spark.rdd.MapPartitionsRDD.compute(MapPartitionsRDD.scala:52)

at org.apache.spark.rdd.RDD.computeOrReadCheckpoint(RDD.scala:373)

at org.apache.spark.rdd.RDD.iterator(RDD.scala:337)

at org.apache.spark.rdd.MapPartitionsRDD.compute(MapPartitionsRDD.scala:52)

at org.apache.spark.rdd.RDD.computeOrReadCheckpoint(RDD.scala:373)

at org.apache.spark.rdd.RDD.iterator(RDD.scala:337)

at org.apache.spark.rdd.MapPartitionsRDD.compute(MapPartitionsRDD.scala:52)

at org.apache.spark.rdd.RDD.computeOrReadCheckpoint(RDD.scala:373)

at org.apache.spark.rdd.RDD.iterator(RDD.scala:337)

at org.apache.spark.scheduler.ResultTask.runTask(ResultTask.scala:90)

at org.apache.spark.scheduler.Task.run(Task.scala:131)

at org.apache.spark.executor.Executor$TaskRunner.$anonfun$run$3(Executor.scala:506)

at org.apache.spark.util.Utils$.tryWithSafeFinally(Utils.scala:1462)

at org.apache.spark.executor.Executor$TaskRunner.run(Executor.scala:509)

at java.util.concurrent.ThreadPoolExecutor.runWorker(ThreadPoolExecutor.java:1149)

at java.util.concurrent.ThreadPoolExecutor$Worker.run(ThreadPoolExecutor.java:624)

at java.lang.Thread.run(Thread.java:750)

24/07/22 09:31:48 ERROR scheduler.TaskSetManager: Task 0 in stage 105.0 failed 1 times; aborting job

Traceback (most recent call last):

File "<stdin>", line 1, in <module>

File "/usr/local/spark/python/pyspark/sql/dataframe.py", line 494, in show

print(self.\_jdf.showString(n, 20, vertical))

File "/usr/local/spark/python/lib/py4j-0.10.9.3-src.zip/py4j/java\_gateway.py", line 1322, in \_\_call\_\_

File "/usr/local/spark/python/pyspark/sql/utils.py", line 117, in deco

raise converted from None

pyspark.sql.utils.PythonException:

An exception was thrown from the Python worker. Please see the stack trace below.

Traceback (most recent call last):

File "/usr/local/spark/python/lib/pyspark.zip/pyspark/worker.py", line 619, in main

process()

File "/usr/local/spark/python/lib/pyspark.zip/pyspark/worker.py", line 611, in process

serializer.dump\_stream(out\_iter, outfile)

File "/usr/local/spark/python/lib/pyspark.zip/pyspark/serializers.py", line 211, in dump\_stream

self.serializer.dump\_stream(self.\_batched(iterator), stream)

File "/usr/local/spark/python/lib/pyspark.zip/pyspark/serializers.py", line 132, in dump\_stream

for obj in iterator:

File "/usr/local/spark/python/lib/pyspark.zip/pyspark/serializers.py", line 200, in \_batched

for item in iterator:

File "/usr/local/spark/python/lib/pyspark.zip/pyspark/worker.py", line 452, in mapper

result = tuple(f(\*[a[o] for o in arg\_offsets]) for (arg\_offsets, f) in udfs)

File "/usr/local/spark/python/lib/pyspark.zip/pyspark/worker.py", line 452, in <genexpr>

result = tuple(f(\*[a[o] for o in arg\_offsets]) for (arg\_offsets, f) in udfs)

File "/usr/local/spark/python/lib/pyspark.zip/pyspark/worker.py", line 87, in <lambda>

return lambda \*a: f(\*a)

File "/usr/local/spark/python/lib/pyspark.zip/pyspark/util.py", line 74, in wrapper

return f(\*args, \*\*kwargs)

File "<stdin>", line 3, in clean\_string

TypeError: expected string or bytes-like object

>>> sample\_op=merged\_data\_frame.withColumn("New NetworkName",df\_function(col("NetworkName")))

>>> sample\_op.show(3)

24/07/22 09:32:00 WARN storage.BlockManager: Putting block rdd\_531\_0 failed due to exception org.apache.spark.api.python.PythonException: Traceback (most recent call last):

File "/usr/local/spark/python/lib/pyspark.zip/pyspark/worker.py", line 619, in main

process()

File "/usr/local/spark/python/lib/pyspark.zip/pyspark/worker.py", line 611, in process

serializer.dump\_stream(out\_iter, outfile)

File "/usr/local/spark/python/lib/pyspark.zip/pyspark/serializers.py", line 211, in dump\_stream

self.serializer.dump\_stream(self.\_batched(iterator), stream)

File "/usr/local/spark/python/lib/pyspark.zip/pyspark/serializers.py", line 132, in dump\_stream

for obj in iterator:

File "/usr/local/spark/python/lib/pyspark.zip/pyspark/serializers.py", line 200, in \_batched

for item in iterator:

File "/usr/local/spark/python/lib/pyspark.zip/pyspark/worker.py", line 452, in mapper

result = tuple(f(\*[a[o] for o in arg\_offsets]) for (arg\_offsets, f) in udfs)

File "/usr/local/spark/python/lib/pyspark.zip/pyspark/worker.py", line 452, in <genexpr>

result = tuple(f(\*[a[o] for o in arg\_offsets]) for (arg\_offsets, f) in udfs)

File "/usr/local/spark/python/lib/pyspark.zip/pyspark/worker.py", line 87, in <lambda>

return lambda \*a: f(\*a)

File "/usr/local/spark/python/lib/pyspark.zip/pyspark/util.py", line 74, in wrapper

return f(\*args, \*\*kwargs)

File "<stdin>", line 3, in clean\_string

TypeError: expected string or bytes-like object

.

24/07/22 09:32:00 WARN storage.BlockManager: Block rdd\_531\_0 could not be removed as it was not found on disk or in memory

24/07/22 09:32:00 ERROR executor.Executor: Exception in task 0.0 in stage 106.0 (TID 117)

org.apache.spark.api.python.PythonException: Traceback (most recent call last):

File "/usr/local/spark/python/lib/pyspark.zip/pyspark/worker.py", line 619, in main

process()

File "/usr/local/spark/python/lib/pyspark.zip/pyspark/worker.py", line 611, in process

serializer.dump\_stream(out\_iter, outfile)

File "/usr/local/spark/python/lib/pyspark.zip/pyspark/serializers.py", line 211, in dump\_stream

self.serializer.dump\_stream(self.\_batched(iterator), stream)

File "/usr/local/spark/python/lib/pyspark.zip/pyspark/serializers.py", line 132, in dump\_stream

for obj in iterator:

File "/usr/local/spark/python/lib/pyspark.zip/pyspark/serializers.py", line 200, in \_batched

for item in iterator:

File "/usr/local/spark/python/lib/pyspark.zip/pyspark/worker.py", line 452, in mapper

result = tuple(f(\*[a[o] for o in arg\_offsets]) for (arg\_offsets, f) in udfs)

File "/usr/local/spark/python/lib/pyspark.zip/pyspark/worker.py", line 452, in <genexpr>

result = tuple(f(\*[a[o] for o in arg\_offsets]) for (arg\_offsets, f) in udfs)

File "/usr/local/spark/python/lib/pyspark.zip/pyspark/worker.py", line 87, in <lambda>

return lambda \*a: f(\*a)

File "/usr/local/spark/python/lib/pyspark.zip/pyspark/util.py", line 74, in wrapper

return f(\*args, \*\*kwargs)

File "<stdin>", line 3, in clean\_string

TypeError: expected string or bytes-like object

at org.apache.spark.api.python.BasePythonRunner$ReaderIterator.handlePythonException(PythonRunner.scala:555)

at org.apache.spark.sql.execution.python.PythonUDFRunner$$anon$2.read(PythonUDFRunner.scala:86)

at org.apache.spark.sql.execution.python.PythonUDFRunner$$anon$2.read(PythonUDFRunner.scala:68)

at org.apache.spark.api.python.BasePythonRunner$ReaderIterator.hasNext(PythonRunner.scala:508)

at org.apache.spark.InterruptibleIterator.hasNext(InterruptibleIterator.scala:37)

at scala.collection.Iterator$$anon$11.hasNext(Iterator.scala:491)

at scala.collection.Iterator$$anon$10.hasNext(Iterator.scala:460)

at scala.collection.Iterator$$anon$10.hasNext(Iterator.scala:460)

at org.apache.spark.sql.catalyst.expressions.GeneratedClass$GeneratedIteratorForCodegenStage1.processNext(Unknown Source)

at org.apache.spark.sql.execution.BufferedRowIterator.hasNext(BufferedRowIterator.java:43)

at org.apache.spark.sql.execution.WholeStageCodegenExec$$anon$1.hasNext(WholeStageCodegenExec.scala:759)

at org.apache.spark.sql.execution.columnar.DefaultCachedBatchSerializer$$anon$1.next(InMemoryRelation.scala:87)

at org.apache.spark.sql.execution.columnar.DefaultCachedBatchSerializer$$anon$1.next(InMemoryRelation.scala:79)

at scala.collection.Iterator$$anon$10.next(Iterator.scala:461)

at org.apache.spark.storage.memory.MemoryStore.putIterator(MemoryStore.scala:224)

at org.apache.spark.storage.memory.MemoryStore.putIteratorAsValues(MemoryStore.scala:302)

at org.apache.spark.storage.BlockManager.$anonfun$doPutIterator$1(BlockManager.scala:1481)

at org.apache.spark.storage.BlockManager.org$apache$spark$storage$BlockManager$$doPut(BlockManager.scala:1408)

at org.apache.spark.storage.BlockManager.doPutIterator(BlockManager.scala:1472)

at org.apache.spark.storage.BlockManager.getOrElseUpdate(BlockManager.scala:1295)

at org.apache.spark.rdd.RDD.getOrCompute(RDD.scala:384)

at org.apache.spark.rdd.RDD.iterator(RDD.scala:335)

at org.apache.spark.rdd.MapPartitionsRDD.compute(MapPartitionsRDD.scala:52)

at org.apache.spark.rdd.RDD.computeOrReadCheckpoint(RDD.scala:373)

at org.apache.spark.rdd.RDD.iterator(RDD.scala:337)

at org.apache.spark.rdd.MapPartitionsRDD.compute(MapPartitionsRDD.scala:52)

at org.apache.spark.rdd.RDD.computeOrReadCheckpoint(RDD.scala:373)

at org.apache.spark.rdd.RDD.iterator(RDD.scala:337)

at org.apache.spark.rdd.MapPartitionsRDD.compute(MapPartitionsRDD.scala:52)

at org.apache.spark.rdd.RDD.computeOrReadCheckpoint(RDD.scala:373)

at org.apache.spark.rdd.RDD.iterator(RDD.scala:337)

at org.apache.spark.rdd.MapPartitionsRDD.compute(MapPartitionsRDD.scala:52)

at org.apache.spark.rdd.RDD.computeOrReadCheckpoint(RDD.scala:373)

at org.apache.spark.rdd.RDD.iterator(RDD.scala:337)

at org.apache.spark.rdd.MapPartitionsRDD.compute(MapPartitionsRDD.scala:52)

at org.apache.spark.rdd.RDD.computeOrReadCheckpoint(RDD.scala:373)

at org.apache.spark.rdd.RDD.iterator(RDD.scala:337)

at org.apache.spark.scheduler.ResultTask.runTask(ResultTask.scala:90)

at org.apache.spark.scheduler.Task.run(Task.scala:131)

at org.apache.spark.executor.Executor$TaskRunner.$anonfun$run$3(Executor.scala:506)

at org.apache.spark.util.Utils$.tryWithSafeFinally(Utils.scala:1462)

at org.apache.spark.executor.Executor$TaskRunner.run(Executor.scala:509)

at java.util.concurrent.ThreadPoolExecutor.runWorker(ThreadPoolExecutor.java:1149)

at java.util.concurrent.ThreadPoolExecutor$Worker.run(ThreadPoolExecutor.java:624)

at java.lang.Thread.run(Thread.java:750)

24/07/22 09:32:00 WARN scheduler.TaskSetManager: Lost task 0.0 in stage 106.0 (TID 117) (192.168.112.129 executor driver): org.apache.spark.api.python.PythonException: Traceback (most recent call last):

File "/usr/local/spark/python/lib/pyspark.zip/pyspark/worker.py", line 619, in main

process()

File "/usr/local/spark/python/lib/pyspark.zip/pyspark/worker.py", line 611, in process

serializer.dump\_stream(out\_iter, outfile)

File "/usr/local/spark/python/lib/pyspark.zip/pyspark/serializers.py", line 211, in dump\_stream

self.serializer.dump\_stream(self.\_batched(iterator), stream)

File "/usr/local/spark/python/lib/pyspark.zip/pyspark/serializers.py", line 132, in dump\_stream

for obj in iterator:

File "/usr/local/spark/python/lib/pyspark.zip/pyspark/serializers.py", line 200, in \_batched

for item in iterator:

File "/usr/local/spark/python/lib/pyspark.zip/pyspark/worker.py", line 452, in mapper

result = tuple(f(\*[a[o] for o in arg\_offsets]) for (arg\_offsets, f) in udfs)

File "/usr/local/spark/python/lib/pyspark.zip/pyspark/worker.py", line 452, in <genexpr>

result = tuple(f(\*[a[o] for o in arg\_offsets]) for (arg\_offsets, f) in udfs)

File "/usr/local/spark/python/lib/pyspark.zip/pyspark/worker.py", line 87, in <lambda>

return lambda \*a: f(\*a)

File "/usr/local/spark/python/lib/pyspark.zip/pyspark/util.py", line 74, in wrapper

return f(\*args, \*\*kwargs)

File "<stdin>", line 3, in clean\_string

TypeError: expected string or bytes-like object

at org.apache.spark.api.python.BasePythonRunner$ReaderIterator.handlePythonException(PythonRunner.scala:555)

at org.apache.spark.sql.execution.python.PythonUDFRunner$$anon$2.read(PythonUDFRunner.scala:86)

at org.apache.spark.sql.execution.python.PythonUDFRunner$$anon$2.read(PythonUDFRunner.scala:68)

at org.apache.spark.api.python.BasePythonRunner$ReaderIterator.hasNext(PythonRunner.scala:508)

at org.apache.spark.InterruptibleIterator.hasNext(InterruptibleIterator.scala:37)

at scala.collection.Iterator$$anon$11.hasNext(Iterator.scala:491)

at scala.collection.Iterator$$anon$10.hasNext(Iterator.scala:460)

at scala.collection.Iterator$$anon$10.hasNext(Iterator.scala:460)

at org.apache.spark.sql.catalyst.expressions.GeneratedClass$GeneratedIteratorForCodegenStage1.processNext(Unknown Source)

at org.apache.spark.sql.execution.BufferedRowIterator.hasNext(BufferedRowIterator.java:43)

at org.apache.spark.sql.execution.WholeStageCodegenExec$$anon$1.hasNext(WholeStageCodegenExec.scala:759)

at org.apache.spark.sql.execution.columnar.DefaultCachedBatchSerializer$$anon$1.next(InMemoryRelation.scala:87)

at org.apache.spark.sql.execution.columnar.DefaultCachedBatchSerializer$$anon$1.next(InMemoryRelation.scala:79)

at scala.collection.Iterator$$anon$10.next(Iterator.scala:461)

at org.apache.spark.storage.memory.MemoryStore.putIterator(MemoryStore.scala:224)

at org.apache.spark.storage.memory.MemoryStore.putIteratorAsValues(MemoryStore.scala:302)

at org.apache.spark.storage.BlockManager.$anonfun$doPutIterator$1(BlockManager.scala:1481)

at org.apache.spark.storage.BlockManager.org$apache$spark$storage$BlockManager$$doPut(BlockManager.scala:1408)

at org.apache.spark.storage.BlockManager.doPutIterator(BlockManager.scala:1472)

at org.apache.spark.storage.BlockManager.getOrElseUpdate(BlockManager.scala:1295)

at org.apache.spark.rdd.RDD.getOrCompute(RDD.scala:384)

at org.apache.spark.rdd.RDD.iterator(RDD.scala:335)

at org.apache.spark.rdd.MapPartitionsRDD.compute(MapPartitionsRDD.scala:52)

at org.apache.spark.rdd.RDD.computeOrReadCheckpoint(RDD.scala:373)

at org.apache.spark.rdd.RDD.iterator(RDD.scala:337)

at org.apache.spark.rdd.MapPartitionsRDD.compute(MapPartitionsRDD.scala:52)

at org.apache.spark.rdd.RDD.computeOrReadCheckpoint(RDD.scala:373)

at org.apache.spark.rdd.RDD.iterator(RDD.scala:337)

at org.apache.spark.rdd.MapPartitionsRDD.compute(MapPartitionsRDD.scala:52)

at org.apache.spark.rdd.RDD.computeOrReadCheckpoint(RDD.scala:373)

at org.apache.spark.rdd.RDD.iterator(RDD.scala:337)

at org.apache.spark.rdd.MapPartitionsRDD.compute(MapPartitionsRDD.scala:52)

at org.apache.spark.rdd.RDD.computeOrReadCheckpoint(RDD.scala:373)

at org.apache.spark.rdd.RDD.iterator(RDD.scala:337)

at org.apache.spark.rdd.MapPartitionsRDD.compute(MapPartitionsRDD.scala:52)

at org.apache.spark.rdd.RDD.computeOrReadCheckpoint(RDD.scala:373)

at org.apache.spark.rdd.RDD.iterator(RDD.scala:337)

at org.apache.spark.scheduler.ResultTask.runTask(ResultTask.scala:90)

at org.apache.spark.scheduler.Task.run(Task.scala:131)

at org.apache.spark.executor.Executor$TaskRunner.$anonfun$run$3(Executor.scala:506)

at org.apache.spark.util.Utils$.tryWithSafeFinally(Utils.scala:1462)

at org.apache.spark.executor.Executor$TaskRunner.run(Executor.scala:509)

at java.util.concurrent.ThreadPoolExecutor.runWorker(ThreadPoolExecutor.java:1149)

at java.util.concurrent.ThreadPoolExecutor$Worker.run(ThreadPoolExecutor.java:624)

at java.lang.Thread.run(Thread.java:750)

24/07/22 09:32:00 ERROR scheduler.TaskSetManager: Task 0 in stage 106.0 failed 1 times; aborting job

Traceback (most recent call last):

File "<stdin>", line 1, in <module>

File "/usr/local/spark/python/pyspark/sql/dataframe.py", line 494, in show

print(self.\_jdf.showString(n, 20, vertical))

File "/usr/local/spark/python/lib/py4j-0.10.9.3-src.zip/py4j/java\_gateway.py", line 1322, in \_\_call\_\_

File "/usr/local/spark/python/pyspark/sql/utils.py", line 117, in deco

raise converted from None

pyspark.sql.utils.PythonException:

An exception was thrown from the Python worker. Please see the stack trace below.

Traceback (most recent call last):

File "/usr/local/spark/python/lib/pyspark.zip/pyspark/worker.py", line 619, in main

process()

File "/usr/local/spark/python/lib/pyspark.zip/pyspark/worker.py", line 611, in process

serializer.dump\_stream(out\_iter, outfile)

File "/usr/local/spark/python/lib/pyspark.zip/pyspark/serializers.py", line 211, in dump\_stream

self.serializer.dump\_stream(self.\_batched(iterator), stream)

File "/usr/local/spark/python/lib/pyspark.zip/pyspark/serializers.py", line 132, in dump\_stream

for obj in iterator:

File "/usr/local/spark/python/lib/pyspark.zip/pyspark/serializers.py", line 200, in \_batched

for item in iterator:

File "/usr/local/spark/python/lib/pyspark.zip/pyspark/worker.py", line 452, in mapper

result = tuple(f(\*[a[o] for o in arg\_offsets]) for (arg\_offsets, f) in udfs)

File "/usr/local/spark/python/lib/pyspark.zip/pyspark/worker.py", line 452, in <genexpr>

result = tuple(f(\*[a[o] for o in arg\_offsets]) for (arg\_offsets, f) in udfs)

File "/usr/local/spark/python/lib/pyspark.zip/pyspark/worker.py", line 87, in <lambda>

return lambda \*a: f(\*a)

File "/usr/local/spark/python/lib/pyspark.zip/pyspark/util.py", line 74, in wrapper

return f(\*args, \*\*kwargs)

File "<stdin>", line 3, in clean\_string

TypeError: expected string or bytes-like object

>>> df\_function=udf(clean\_string,StringType())

>>> sample\_op=merged\_data\_frame.withColumn("New NetworkName",df\_function(col("NetworkName")))

>>> sample\_op.show(3)

24/07/22 09:32:25 WARN storage.BlockManager: Putting block rdd\_531\_0 failed due to exception org.apache.spark.api.python.PythonException: Traceback (most recent call last):

File "/usr/local/spark/python/lib/pyspark.zip/pyspark/worker.py", line 619, in main

process()

File "/usr/local/spark/python/lib/pyspark.zip/pyspark/worker.py", line 611, in process

serializer.dump\_stream(out\_iter, outfile)

File "/usr/local/spark/python/lib/pyspark.zip/pyspark/serializers.py", line 211, in dump\_stream

self.serializer.dump\_stream(self.\_batched(iterator), stream)

File "/usr/local/spark/python/lib/pyspark.zip/pyspark/serializers.py", line 132, in dump\_stream

for obj in iterator:

File "/usr/local/spark/python/lib/pyspark.zip/pyspark/serializers.py", line 200, in \_batched

for item in iterator:

File "/usr/local/spark/python/lib/pyspark.zip/pyspark/worker.py", line 452, in mapper

result = tuple(f(\*[a[o] for o in arg\_offsets]) for (arg\_offsets, f) in udfs)

File "/usr/local/spark/python/lib/pyspark.zip/pyspark/worker.py", line 452, in <genexpr>

result = tuple(f(\*[a[o] for o in arg\_offsets]) for (arg\_offsets, f) in udfs)

File "/usr/local/spark/python/lib/pyspark.zip/pyspark/worker.py", line 87, in <lambda>

return lambda \*a: f(\*a)

File "/usr/local/spark/python/lib/pyspark.zip/pyspark/util.py", line 74, in wrapper

return f(\*args, \*\*kwargs)

File "<stdin>", line 3, in clean\_string

TypeError: expected string or bytes-like object

.

24/07/22 09:32:25 WARN storage.BlockManager: Block rdd\_531\_0 could not be removed as it was not found on disk or in memory

24/07/22 09:32:25 ERROR executor.Executor: Exception in task 0.0 in stage 107.0 (TID 118)

org.apache.spark.api.python.PythonException: Traceback (most recent call last):

File "/usr/local/spark/python/lib/pyspark.zip/pyspark/worker.py", line 619, in main

process()

File "/usr/local/spark/python/lib/pyspark.zip/pyspark/worker.py", line 611, in process

serializer.dump\_stream(out\_iter, outfile)

File "/usr/local/spark/python/lib/pyspark.zip/pyspark/serializers.py", line 211, in dump\_stream

self.serializer.dump\_stream(self.\_batched(iterator), stream)

File "/usr/local/spark/python/lib/pyspark.zip/pyspark/serializers.py", line 132, in dump\_stream

for obj in iterator:

File "/usr/local/spark/python/lib/pyspark.zip/pyspark/serializers.py", line 200, in \_batched

for item in iterator:

File "/usr/local/spark/python/lib/pyspark.zip/pyspark/worker.py", line 452, in mapper

result = tuple(f(\*[a[o] for o in arg\_offsets]) for (arg\_offsets, f) in udfs)

File "/usr/local/spark/python/lib/pyspark.zip/pyspark/worker.py", line 452, in <genexpr>

result = tuple(f(\*[a[o] for o in arg\_offsets]) for (arg\_offsets, f) in udfs)

File "/usr/local/spark/python/lib/pyspark.zip/pyspark/worker.py", line 87, in <lambda>

return lambda \*a: f(\*a)

File "/usr/local/spark/python/lib/pyspark.zip/pyspark/util.py", line 74, in wrapper

return f(\*args, \*\*kwargs)

File "<stdin>", line 3, in clean\_string

TypeError: expected string or bytes-like object

at org.apache.spark.api.python.BasePythonRunner$ReaderIterator.handlePythonException(PythonRunner.scala:555)

at org.apache.spark.sql.execution.python.PythonUDFRunner$$anon$2.read(PythonUDFRunner.scala:86)

at org.apache.spark.sql.execution.python.PythonUDFRunner$$anon$2.read(PythonUDFRunner.scala:68)

at org.apache.spark.api.python.BasePythonRunner$ReaderIterator.hasNext(PythonRunner.scala:508)

at org.apache.spark.InterruptibleIterator.hasNext(InterruptibleIterator.scala:37)

at scala.collection.Iterator$$anon$11.hasNext(Iterator.scala:491)

at scala.collection.Iterator$$anon$10.hasNext(Iterator.scala:460)

at scala.collection.Iterator$$anon$10.hasNext(Iterator.scala:460)

at org.apache.spark.sql.catalyst.expressions.GeneratedClass$GeneratedIteratorForCodegenStage1.processNext(Unknown Source)

at org.apache.spark.sql.execution.BufferedRowIterator.hasNext(BufferedRowIterator.java:43)

at org.apache.spark.sql.execution.WholeStageCodegenExec$$anon$1.hasNext(WholeStageCodegenExec.scala:759)

at org.apache.spark.sql.execution.columnar.DefaultCachedBatchSerializer$$anon$1.next(InMemoryRelation.scala:87)

at org.apache.spark.sql.execution.columnar.DefaultCachedBatchSerializer$$anon$1.next(InMemoryRelation.scala:79)

at scala.collection.Iterator$$anon$10.next(Iterator.scala:461)

at org.apache.spark.storage.memory.MemoryStore.putIterator(MemoryStore.scala:224)

at org.apache.spark.storage.memory.MemoryStore.putIteratorAsValues(MemoryStore.scala:302)

at org.apache.spark.storage.BlockManager.$anonfun$doPutIterator$1(BlockManager.scala:1481)

at org.apache.spark.storage.BlockManager.org$apache$spark$storage$BlockManager$$doPut(BlockManager.scala:1408)

at org.apache.spark.storage.BlockManager.doPutIterator(BlockManager.scala:1472)

at org.apache.spark.storage.BlockManager.getOrElseUpdate(BlockManager.scala:1295)

at org.apache.spark.rdd.RDD.getOrCompute(RDD.scala:384)

at org.apache.spark.rdd.RDD.iterator(RDD.scala:335)

at org.apache.spark.rdd.MapPartitionsRDD.compute(MapPartitionsRDD.scala:52)

at org.apache.spark.rdd.RDD.computeOrReadCheckpoint(RDD.scala:373)

at org.apache.spark.rdd.RDD.iterator(RDD.scala:337)

at org.apache.spark.rdd.MapPartitionsRDD.compute(MapPartitionsRDD.scala:52)

at org.apache.spark.rdd.RDD.computeOrReadCheckpoint(RDD.scala:373)

at org.apache.spark.rdd.RDD.iterator(RDD.scala:337)

at org.apache.spark.rdd.MapPartitionsRDD.compute(MapPartitionsRDD.scala:52)

at org.apache.spark.rdd.RDD.computeOrReadCheckpoint(RDD.scala:373)

at org.apache.spark.rdd.RDD.iterator(RDD.scala:337)

at org.apache.spark.rdd.MapPartitionsRDD.compute(MapPartitionsRDD.scala:52)

at org.apache.spark.rdd.RDD.computeOrReadCheckpoint(RDD.scala:373)

at org.apache.spark.rdd.RDD.iterator(RDD.scala:337)

at org.apache.spark.rdd.MapPartitionsRDD.compute(MapPartitionsRDD.scala:52)

at org.apache.spark.rdd.RDD.computeOrReadCheckpoint(RDD.scala:373)

at org.apache.spark.rdd.RDD.iterator(RDD.scala:337)

at org.apache.spark.scheduler.ResultTask.runTask(ResultTask.scala:90)

at org.apache.spark.scheduler.Task.run(Task.scala:131)

at org.apache.spark.executor.Executor$TaskRunner.$anonfun$run$3(Executor.scala:506)

at org.apache.spark.util.Utils$.tryWithSafeFinally(Utils.scala:1462)

at org.apache.spark.executor.Executor$TaskRunner.run(Executor.scala:509)

at java.util.concurrent.ThreadPoolExecutor.runWorker(ThreadPoolExecutor.java:1149)

at java.util.concurrent.ThreadPoolExecutor$Worker.run(ThreadPoolExecutor.java:624)

at java.lang.Thread.run(Thread.java:750)

24/07/22 09:32:25 WARN scheduler.TaskSetManager: Lost task 0.0 in stage 107.0 (TID 118) (192.168.112.129 executor driver): org.apache.spark.api.python.PythonException: Traceback (most recent call last):

File "/usr/local/spark/python/lib/pyspark.zip/pyspark/worker.py", line 619, in main

process()

File "/usr/local/spark/python/lib/pyspark.zip/pyspark/worker.py", line 611, in process

serializer.dump\_stream(out\_iter, outfile)

File "/usr/local/spark/python/lib/pyspark.zip/pyspark/serializers.py", line 211, in dump\_stream

self.serializer.dump\_stream(self.\_batched(iterator), stream)

File "/usr/local/spark/python/lib/pyspark.zip/pyspark/serializers.py", line 132, in dump\_stream

for obj in iterator:

File "/usr/local/spark/python/lib/pyspark.zip/pyspark/serializers.py", line 200, in \_batched

for item in iterator:

File "/usr/local/spark/python/lib/pyspark.zip/pyspark/worker.py", line 452, in mapper

result = tuple(f(\*[a[o] for o in arg\_offsets]) for (arg\_offsets, f) in udfs)

File "/usr/local/spark/python/lib/pyspark.zip/pyspark/worker.py", line 452, in <genexpr>

result = tuple(f(\*[a[o] for o in arg\_offsets]) for (arg\_offsets, f) in udfs)

File "/usr/local/spark/python/lib/pyspark.zip/pyspark/worker.py", line 87, in <lambda>

return lambda \*a: f(\*a)

File "/usr/local/spark/python/lib/pyspark.zip/pyspark/util.py", line 74, in wrapper

return f(\*args, \*\*kwargs)

File "<stdin>", line 3, in clean\_string

TypeError: expected string or bytes-like object

at org.apache.spark.api.python.BasePythonRunner$ReaderIterator.handlePythonException(PythonRunner.scala:555)

at org.apache.spark.sql.execution.python.PythonUDFRunner$$anon$2.read(PythonUDFRunner.scala:86)

at org.apache.spark.sql.execution.python.PythonUDFRunner$$anon$2.read(PythonUDFRunner.scala:68)

at org.apache.spark.api.python.BasePythonRunner$ReaderIterator.hasNext(PythonRunner.scala:508)

at org.apache.spark.InterruptibleIterator.hasNext(InterruptibleIterator.scala:37)

at scala.collection.Iterator$$anon$11.hasNext(Iterator.scala:491)

at scala.collection.Iterator$$anon$10.hasNext(Iterator.scala:460)

at scala.collection.Iterator$$anon$10.hasNext(Iterator.scala:460)

at org.apache.spark.sql.catalyst.expressions.GeneratedClass$GeneratedIteratorForCodegenStage1.processNext(Unknown Source)

at org.apache.spark.sql.execution.BufferedRowIterator.hasNext(BufferedRowIterator.java:43)

at org.apache.spark.sql.execution.WholeStageCodegenExec$$anon$1.hasNext(WholeStageCodegenExec.scala:759)

at org.apache.spark.sql.execution.columnar.DefaultCachedBatchSerializer$$anon$1.next(InMemoryRelation.scala:87)

at org.apache.spark.sql.execution.columnar.DefaultCachedBatchSerializer$$anon$1.next(InMemoryRelation.scala:79)

at scala.collection.Iterator$$anon$10.next(Iterator.scala:461)

at org.apache.spark.storage.memory.MemoryStore.putIterator(MemoryStore.scala:224)

at org.apache.spark.storage.memory.MemoryStore.putIteratorAsValues(MemoryStore.scala:302)

at org.apache.spark.storage.BlockManager.$anonfun$doPutIterator$1(BlockManager.scala:1481)

at org.apache.spark.storage.BlockManager.org$apache$spark$storage$BlockManager$$doPut(BlockManager.scala:1408)

at org.apache.spark.storage.BlockManager.doPutIterator(BlockManager.scala:1472)

at org.apache.spark.storage.BlockManager.getOrElseUpdate(BlockManager.scala:1295)

at org.apache.spark.rdd.RDD.getOrCompute(RDD.scala:384)

at org.apache.spark.rdd.RDD.iterator(RDD.scala:335)

at org.apache.spark.rdd.MapPartitionsRDD.compute(MapPartitionsRDD.scala:52)

at org.apache.spark.rdd.RDD.computeOrReadCheckpoint(RDD.scala:373)

at org.apache.spark.rdd.RDD.iterator(RDD.scala:337)

at org.apache.spark.rdd.MapPartitionsRDD.compute(MapPartitionsRDD.scala:52)

at org.apache.spark.rdd.RDD.computeOrReadCheckpoint(RDD.scala:373)

at org.apache.spark.rdd.RDD.iterator(RDD.scala:337)

at org.apache.spark.rdd.MapPartitionsRDD.compute(MapPartitionsRDD.scala:52)

at org.apache.spark.rdd.RDD.computeOrReadCheckpoint(RDD.scala:373)

at org.apache.spark.rdd.RDD.iterator(RDD.scala:337)

at org.apache.spark.rdd.MapPartitionsRDD.compute(MapPartitionsRDD.scala:52)

at org.apache.spark.rdd.RDD.computeOrReadCheckpoint(RDD.scala:373)

at org.apache.spark.rdd.RDD.iterator(RDD.scala:337)

at org.apache.spark.rdd.MapPartitionsRDD.compute(MapPartitionsRDD.scala:52)

at org.apache.spark.rdd.RDD.computeOrReadCheckpoint(RDD.scala:373)

at org.apache.spark.rdd.RDD.iterator(RDD.scala:337)

at org.apache.spark.scheduler.ResultTask.runTask(ResultTask.scala:90)

at org.apache.spark.scheduler.Task.run(Task.scala:131)

at org.apache.spark.executor.Executor$TaskRunner.$anonfun$run$3(Executor.scala:506)

at org.apache.spark.util.Utils$.tryWithSafeFinally(Utils.scala:1462)

at org.apache.spark.executor.Executor$TaskRunner.run(Executor.scala:509)

at java.util.concurrent.ThreadPoolExecutor.runWorker(ThreadPoolExecutor.java:1149)

at java.util.concurrent.ThreadPoolExecutor$Worker.run(ThreadPoolExecutor.java:624)

at java.lang.Thread.run(Thread.java:750)

24/07/22 09:32:25 ERROR scheduler.TaskSetManager: Task 0 in stage 107.0 failed 1 times; aborting job

Traceback (most recent call last):

File "<stdin>", line 1, in <module>

File "/usr/local/spark/python/pyspark/sql/dataframe.py", line 494, in show

print(self.\_jdf.showString(n, 20, vertical))

File "/usr/local/spark/python/lib/py4j-0.10.9.3-src.zip/py4j/java\_gateway.py", line 1322, in \_\_call\_\_

File "/usr/local/spark/python/pyspark/sql/utils.py", line 117, in deco

raise converted from None

pyspark.sql.utils.PythonException:

An exception was thrown from the Python worker. Please see the stack trace below.

Traceback (most recent call last):

File "/usr/local/spark/python/lib/pyspark.zip/pyspark/worker.py", line 619, in main

process()

File "/usr/local/spark/python/lib/pyspark.zip/pyspark/worker.py", line 611, in process

serializer.dump\_stream(out\_iter, outfile)

File "/usr/local/spark/python/lib/pyspark.zip/pyspark/serializers.py", line 211, in dump\_stream

self.serializer.dump\_stream(self.\_batched(iterator), stream)

File "/usr/local/spark/python/lib/pyspark.zip/pyspark/serializers.py", line 132, in dump\_stream

for obj in iterator:

File "/usr/local/spark/python/lib/pyspark.zip/pyspark/serializers.py", line 200, in \_batched

for item in iterator:

File "/usr/local/spark/python/lib/pyspark.zip/pyspark/worker.py", line 452, in mapper

result = tuple(f(\*[a[o] for o in arg\_offsets]) for (arg\_offsets, f) in udfs)

File "/usr/local/spark/python/lib/pyspark.zip/pyspark/worker.py", line 452, in <genexpr>

result = tuple(f(\*[a[o] for o in arg\_offsets]) for (arg\_offsets, f) in udfs)

File "/usr/local/spark/python/lib/pyspark.zip/pyspark/worker.py", line 87, in <lambda>

return lambda \*a: f(\*a)

File "/usr/local/spark/python/lib/pyspark.zip/pyspark/util.py", line 74, in wrapper

return f(\*args, \*\*kwargs)

File "<stdin>", line 3, in clean\_string

TypeError: expected string or bytes-like object

>>> import re

>>>

>>> def clean\_string(string\_arg):

... if isinstance(string\_arg, str):

... # Define the regex pattern to match non-alphabetic characters and replace them with space

... pattern = re.compile(r'[^A-Za-z\s]')

... final\_string = pattern.sub(' ', string\_arg)

... return final\_string

... else:

... return None

...

>>> df\_function=udf(clean\_string,StringType())

>>> sample\_op=merged\_data\_frame.withColumn("New NetworkName",df\_function(col("NetworkName")))

>>> sample\_op.show(3)

+------------+----+-----+------------+--------------------+-------+--------------+--------------------+--------+---------+-----------------+----------+--------------------+---------------+

|BusinessDate|stcd|srcnm| NetworkName| NetworkURL|custnum|MarketCoverage| RejectRows|IssuerId|IssuerId2|issueridcomposite| sysdt| systs|New NetworkName|

+------------+----+-----+------------+--------------------+-------+--------------+--------------------+--------+---------+-----------------+----------+--------------------+---------------+

| 2019-10-01| AK| HIOS| ODS Premier|https://www.modah...| 13| null| null| 21989| 21989| 2198921989|2024-07-22|2024-07-22 09:35:...| ODS Premier|

| 2019-10-01| AK| HIOS| ODS Premier| null| null| null|219,219,2019-10-0...| 219| 219| 219219|2024-07-22|2024-07-22 09:35:...| ODS Premier|

| 2019-10-01| AK| HIOS|HeritagePlus|https://www.preme...| 13| null| null| 38344| 38344| 3834438344|2024-07-22|2024-07-22 09:35:...| HeritagePlus|

+------------+----+-----+------------+--------------------+-------+--------------+--------------------+--------+---------+-----------------+----------+--------------------+---------------+

only showing top 3 rows

>>> sample\_op.write.json("/user/hduser/Hackathon/JSON\_DATA" , mode="overwrite")

>>> sample\_op.write.csv("/user/hduser/Hackathon/CSV\_DATA" , header=True , sep="~")

>>>

Traceback (most recent call last):

File "/usr/local/spark/python/pyspark/context.py", line 293, in signal\_handler

raise KeyboardInterrupt()

KeyboardInterrupt

>>> database\_name = "Hackathon"

>>> table\_name = "Hive\_table"

>>> table\_location = "/user/hduser/Hackathon/Hive\_table"

>>>

>>> # Create Hive database if it doesn't exist

... spark.sql(f"CREATE DATABASE IF NOT EXISTS {database\_name}")

24/07/22 09:41:26 WARN metastore.ObjectStore: Failed to get database global\_temp, returning NoSuchObjectException

24/07/22 09:41:27 WARN metastore.ObjectStore: Failed to get database hackathon, returning NoSuchObjectException

DataFrame[]

>>>

>>> # Define the schema for the table

... schema = """

... BusinessDate DATE,

... stcd STRING,

... srcnm STRING,

... NetworkName STRING,

... NetworkURL STRING,

... custnum STRING,

... MarketCoverage STRING,

... RejectRows STRING,

... IssuerId INT,

... IssuerId2 INT,

... issueridcomposite STRING,

... sysdt TIMESTAMP,

... systs STRING,

... `New NetworkName` STRING

... """

>>>

>>> # Create the Hive table with the specified schema

... spark.sql(f"""

... CREATE EXTERNAL TABLE IF NOT EXISTS {database\_name}.{table\_name} (

... {schema}

... )

... LOCATION '{table\_location}'

... """)

24/07/22 09:41:29 WARN analysis.ResolveSessionCatalog: A Hive serde table will be created as there is no table provider specified. You can set spark.sql.legacy.createHiveTableByDefault to false so that native data source table will be created instead.

24/07/22 09:41:30 WARN session.SessionState: METASTORE\_FILTER\_HOOK will be ignored, since hive.security.authorization.manager is set to instance of HiveAuthorizerFactory.

24/07/22 09:41:33 WARN conf.HiveConf: HiveConf of name hive.internal.ss.authz.settings.applied.marker does not exist

24/07/22 09:41:33 WARN conf.HiveConf: HiveConf of name hive.stats.jdbc.timeout does not exist

24/07/22 09:41:33 WARN conf.HiveConf: HiveConf of name hive.stats.retries.wait does not exist

DataFrame[]

>>>

>>> print(f"Table {table\_name} created in database {database\_name} at location {table\_location}")

Table Hive\_table created in database Hackathon at location /user/hduser/Hackathon/Hive\_table

>>> hive> describe hive\_table;

OK

businessdate date

stcd string

srcnm string

networkname string

networkurl string

custnum string

marketcoverage string

rejectrows string

issuerid int

issuerid2 int

issueridcomposite string

sysdt timestamp

systs string

new networkname string

Time taken: 0.417 seconds, Fetched: 14 row(s)

hive> select \* from hive\_table;

OK

Time taken: 0.75 seconds

hive>

2

>>> rdd=read.textFile("/home/hduser/Desktop/Hackathon")

Traceback (most recent call last):

File "<stdin>", line 1, in <module>

NameError: name 'read' is not defined

>>> rdd=sc.read.textFile("/home/hduser/Desktop/Hackathon")

Traceback (most recent call last):

File "<stdin>", line 1, in <module>

AttributeError: 'SparkContext' object has no attribute 'read'

>>> rdd=sc.textFile("/home/hduser/Desktop/Hackathon")

>>> rdd.take(3)

Traceback (most recent call last):

File "<stdin>", line 1, in <module>

File "/usr/local/spark/python/pyspark/rdd.py", line 1535, in take

totalParts = self.getNumPartitions()

File "/usr/local/spark/python/pyspark/rdd.py", line 466, in getNumPartitions

return self.\_jrdd.partitions().size()

File "/usr/local/spark/python/lib/py4j-0.10.9.3-src.zip/py4j/java\_gateway.py", line 1322, in \_\_call\_\_

File "/usr/local/spark/python/pyspark/sql/utils.py", line 111, in deco

return f(\*a, \*\*kw)

File "/usr/local/spark/python/lib/py4j-0.10.9.3-src.zip/py4j/protocol.py", line 328, in get\_return\_value

py4j.protocol.Py4JJavaError: An error occurred while calling o751.partitions.

: org.apache.hadoop.mapred.InvalidInputException: Input path does not exist: hdfs://localhost:54310/home/hduser/Desktop/Hackathon

at org.apache.hadoop.mapred.FileInputFormat.singleThreadedListStatus(FileInputFormat.java:287)

at org.apache.hadoop.mapred.FileInputFormat.listStatus(FileInputFormat.java:229)

at org.apache.hadoop.mapred.FileInputFormat.getSplits(FileInputFormat.java:315)

at org.apache.spark.rdd.HadoopRDD.getPartitions(HadoopRDD.scala:205)

at org.apache.spark.rdd.RDD.$anonfun$partitions$2(RDD.scala:300)

at scala.Option.getOrElse(Option.scala:189)

at org.apache.spark.rdd.RDD.partitions(RDD.scala:296)

at org.apache.spark.rdd.MapPartitionsRDD.getPartitions(MapPartitionsRDD.scala:49)

at org.apache.spark.rdd.RDD.$anonfun$partitions$2(RDD.scala:300)

at scala.Option.getOrElse(Option.scala:189)

at org.apache.spark.rdd.RDD.partitions(RDD.scala:296)

at org.apache.spark.api.java.JavaRDDLike.partitions(JavaRDDLike.scala:61)

at org.apache.spark.api.java.JavaRDDLike.partitions$(JavaRDDLike.scala:61)

at org.apache.spark.api.java.AbstractJavaRDDLike.partitions(JavaRDDLike.scala:45)

at sun.reflect.NativeMethodAccessorImpl.invoke0(Native Method)

at sun.reflect.NativeMethodAccessorImpl.invoke(NativeMethodAccessorImpl.java:62)

at sun.reflect.DelegatingMethodAccessorImpl.invoke(DelegatingMethodAccessorImpl.java:43)

at java.lang.reflect.Method.invoke(Method.java:498)

at py4j.reflection.MethodInvoker.invoke(MethodInvoker.java:244)

at py4j.reflection.ReflectionEngine.invoke(ReflectionEngine.java:357)

at py4j.Gateway.invoke(Gateway.java:282)

at py4j.commands.AbstractCommand.invokeMethod(AbstractCommand.java:132)

at py4j.commands.CallCommand.execute(CallCommand.java:79)

at py4j.ClientServerConnection.waitForCommands(ClientServerConnection.java:182)

at py4j.ClientServerConnection.run(ClientServerConnection.java:106)

at java.lang.Thread.run(Thread.java:750)

>>> rdd=sc.textFile("file:///home/hduser/Desktop/Hackathon")

>>> rdd.take(3)

['1,Kristina,Chung,55,Pilot', '2,Paige,Chen,74,Teacher', 'NM,New Mexico']

>>> def function (rdd):

... for i in rdd :

... if len(i) ==5 :

... custfilter.append(i)

... else :

... statefilter.append(i)

...

>>> statefilter

Traceback (most recent call last):

File "<stdin>", line 1, in <module>

NameError: name 'statefilter' is not defined

>>> statefilter =[]

>>> custfilter =[]

>>> function(rdd)

Traceback (most recent call last):

File "<stdin>", line 1, in <module>

File "<stdin>", line 2, in function

TypeError: 'RDD' object is not iterable

>>> custfilter =rdd.filter(lambda x: if len(x)>5 return x )

File "<stdin>", line 1

custfilter =rdd.filter(lambda x: if len(x)>5 return x )

^

SyntaxError: invalid syntax

>>> custfilter =rdd.filter(lambda x: if len(x)>5 )

File "<stdin>", line 1

custfilter =rdd.filter(lambda x: if len(x)>5 )

^

SyntaxError: invalid syntax

>>> custfilter =rdd.filter(lambda x: len(x)>5 )

>>> print(custfilter)

PythonRDD[629] at RDD at PythonRDD.scala:53

>>> print(custfilter.take(5))

['1,Kristina,Chung,55,Pilot', '2,Paige,Chen,74,Teacher', 'NM,New Mexico', 'NY,New York', 'NC,North Carolina']

>>> custfilter =rdd.filter(lambda x: len(x)>2 )

>>> print(custfilter.take(5))

['1,Kristina,Chung,55,Pilot', '2,Paige,Chen,74,Teacher', 'NM,New Mexico', 'NY,New York', 'NC,North Carolina']

>>> custfilter =rdd.map(lambda x: len(x)>2 ).collect()

>>> print(custfilter.take(5))

Traceback (most recent call last):

File "<stdin>", line 1, in <module>

AttributeError: 'list' object has no attribute 'take'

>>> print(custfilter)

[True, True, True, True, True, True, …….]

>>> custfilter =rdd.map(lambda x: len(x)>2 ).collect()

>>> custfilter =rdd.map(lambda x: len(x)>2 )

>>> print(custfilter)

PythonRDD[634] at RDD at PythonRDD.scala:53

>>> print(custfilter.take()3)

File "<stdin>", line 1

print(custfilter.take()3)

^

SyntaxError: invalid syntax

>>> print(custfilter.take(4))

[True, True, True, True]

>>> print(custfilter.take(14))

[True, True, True, True, True, True, True, True, True, True, True, True, True, True]

>>> custfilter =rdd.map(lambda x: (len(x)>2).collect )

>>> print(custfilter.take(14))

24/07/22 11:12:52 ERROR executor.Executor: Exception in task 0.0 in stage 118.0 (TID 137)

org.apache.spark.api.python.PythonException: Traceback (most recent call last):

File "/usr/local/spark/python/lib/pyspark.zip/pyspark/worker.py", line 619, in main

process()

File "/usr/local/spark/python/lib/pyspark.zip/pyspark/worker.py", line 611, in process

serializer.dump\_stream(out\_iter, outfile)

File "/usr/local/spark/python/lib/pyspark.zip/pyspark/serializers.py", line 259, in dump\_stream

vs = list(itertools.islice(iterator, batch))

File "/usr/local/spark/python/pyspark/rdd.py", line 1562, in takeUpToNumLeft

yield next(iterator)

File "/usr/local/spark/python/lib/pyspark.zip/pyspark/util.py", line 74, in wrapper

return f(\*args, \*\*kwargs)

File "<stdin>", line 1, in <lambda>

AttributeError: 'bool' object has no attribute 'collect'

at org.apache.spark.api.python.BasePythonRunner$ReaderIterator.handlePythonException(PythonRunner.scala:555)

at org.apache.spark.api.python.PythonRunner$$anon$3.read(PythonRunner.scala:713)

at org.apache.spark.api.python.PythonRunner$$anon$3.read(PythonRunner.scala:695)

at org.apache.spark.api.python.BasePythonRunner$ReaderIterator.hasNext(PythonRunner.scala:508)

at org.apache.spark.InterruptibleIterator.hasNext(InterruptibleIterator.scala:37)

at scala.collection.Iterator.foreach(Iterator.scala:943)

at scala.collection.Iterator.foreach$(Iterator.scala:943)

at org.apache.spark.InterruptibleIterator.foreach(InterruptibleIterator.scala:28)

at scala.collection.generic.Growable.$plus$plus$eq(Growable.scala:62)

at scala.collection.generic.Growable.$plus$plus$eq$(Growable.scala:53)

at scala.collection.mutable.ArrayBuffer.$plus$plus$eq(ArrayBuffer.scala:105)

at scala.collection.mutable.ArrayBuffer.$plus$plus$eq(ArrayBuffer.scala:49)

at scala.collection.TraversableOnce.to(TraversableOnce.scala:366)

at scala.collection.TraversableOnce.to$(TraversableOnce.scala:364)

at org.apache.spark.InterruptibleIterator.to(InterruptibleIterator.scala:28)

at scala.collection.TraversableOnce.toBuffer(TraversableOnce.scala:358)

at scala.collection.TraversableOnce.toBuffer$(TraversableOnce.scala:358)

at org.apache.spark.InterruptibleIterator.toBuffer(InterruptibleIterator.scala:28)

at scala.collection.TraversableOnce.toArray(TraversableOnce.scala:345)

at scala.collection.TraversableOnce.toArray$(TraversableOnce.scala:339)

at org.apache.spark.InterruptibleIterator.toArray(InterruptibleIterator.scala:28)

at org.apache.spark.api.python.PythonRDD$.$anonfun$runJob$1(PythonRDD.scala:166)

at org.apache.spark.SparkContext.$anonfun$runJob$5(SparkContext.scala:2254)

at org.apache.spark.scheduler.ResultTask.runTask(ResultTask.scala:90)

at org.apache.spark.scheduler.Task.run(Task.scala:131)

at org.apache.spark.executor.Executor$TaskRunner.$anonfun$run$3(Executor.scala:506)

at org.apache.spark.util.Utils$.tryWithSafeFinally(Utils.scala:1462)

at org.apache.spark.executor.Executor$TaskRunner.run(Executor.scala:509)

at java.util.concurrent.ThreadPoolExecutor.runWorker(ThreadPoolExecutor.java:1149)

at java.util.concurrent.ThreadPoolExecutor$Worker.run(ThreadPoolExecutor.java:624)

at java.lang.Thread.run(Thread.java:750)

24/07/22 11:12:52 WARN scheduler.TaskSetManager: Lost task 0.0 in stage 118.0 (TID 137) (192.168.112.129 executor driver): org.apache.spark.api.python.PythonException: Traceback (most recent call last):

File "/usr/local/spark/python/lib/pyspark.zip/pyspark/worker.py", line 619, in main

process()

File "/usr/local/spark/python/lib/pyspark.zip/pyspark/worker.py", line 611, in process

serializer.dump\_stream(out\_iter, outfile)

File "/usr/local/spark/python/lib/pyspark.zip/pyspark/serializers.py", line 259, in dump\_stream

vs = list(itertools.islice(iterator, batch))

File "/usr/local/spark/python/pyspark/rdd.py", line 1562, in takeUpToNumLeft

yield next(iterator)

File "/usr/local/spark/python/lib/pyspark.zip/pyspark/util.py", line 74, in wrapper

return f(\*args, \*\*kwargs)

File "<stdin>", line 1, in <lambda>

AttributeError: 'bool' object has no attribute 'collect'

at org.apache.spark.api.python.BasePythonRunner$ReaderIterator.handlePythonException(PythonRunner.scala:555)

at org.apache.spark.api.python.PythonRunner$$anon$3.read(PythonRunner.scala:713)

at org.apache.spark.api.python.PythonRunner$$anon$3.read(PythonRunner.scala:695)

at org.apache.spark.api.python.BasePythonRunner$ReaderIterator.hasNext(PythonRunner.scala:508)

at org.apache.spark.InterruptibleIterator.hasNext(InterruptibleIterator.scala:37)

at scala.collection.Iterator.foreach(Iterator.scala:943)

at scala.collection.Iterator.foreach$(Iterator.scala:943)

at org.apache.spark.InterruptibleIterator.foreach(InterruptibleIterator.scala:28)

at scala.collection.generic.Growable.$plus$plus$eq(Growable.scala:62)

at scala.collection.generic.Growable.$plus$plus$eq$(Growable.scala:53)

at scala.collection.mutable.ArrayBuffer.$plus$plus$eq(ArrayBuffer.scala:105)

at scala.collection.mutable.ArrayBuffer.$plus$plus$eq(ArrayBuffer.scala:49)

at scala.collection.TraversableOnce.to(TraversableOnce.scala:366)

at scala.collection.TraversableOnce.to$(TraversableOnce.scala:364)

at org.apache.spark.InterruptibleIterator.to(InterruptibleIterator.scala:28)

at scala.collection.TraversableOnce.toBuffer(TraversableOnce.scala:358)

at scala.collection.TraversableOnce.toBuffer$(TraversableOnce.scala:358)

at org.apache.spark.InterruptibleIterator.toBuffer(InterruptibleIterator.scala:28)

at scala.collection.TraversableOnce.toArray(TraversableOnce.scala:345)

at scala.collection.TraversableOnce.toArray$(TraversableOnce.scala:339)

at org.apache.spark.InterruptibleIterator.toArray(InterruptibleIterator.scala:28)

at org.apache.spark.api.python.PythonRDD$.$anonfun$runJob$1(PythonRDD.scala:166)

at org.apache.spark.SparkContext.$anonfun$runJob$5(SparkContext.scala:2254)

at org.apache.spark.scheduler.ResultTask.runTask(ResultTask.scala:90)

at org.apache.spark.scheduler.Task.run(Task.scala:131)

at org.apache.spark.executor.Executor$TaskRunner.$anonfun$run$3(Executor.scala:506)

at org.apache.spark.util.Utils$.tryWithSafeFinally(Utils.scala:1462)

at org.apache.spark.executor.Executor$TaskRunner.run(Executor.scala:509)

at java.util.concurrent.ThreadPoolExecutor.runWorker(ThreadPoolExecutor.java:1149)

at java.util.concurrent.ThreadPoolExecutor$Worker.run(ThreadPoolExecutor.java:624)

at java.lang.Thread.run(Thread.java:750)

24/07/22 11:12:52 ERROR scheduler.TaskSetManager: Task 0 in stage 118.0 failed 1 times; aborting job

Traceback (most recent call last):

File "<stdin>", line 1, in <module>

File "/usr/local/spark/python/pyspark/rdd.py", line 1568, in take

res = self.context.runJob(self, takeUpToNumLeft, p)

File "/usr/local/spark/python/pyspark/context.py", line 1227, in runJob

sock\_info = self.\_jvm.PythonRDD.runJob(self.\_jsc.sc(), mappedRDD.\_jrdd, partitions)

File "/usr/local/spark/python/lib/py4j-0.10.9.3-src.zip/py4j/java\_gateway.py", line 1322, in \_\_call\_\_

File "/usr/local/spark/python/pyspark/sql/utils.py", line 111, in deco

return f(\*a, \*\*kw)

File "/usr/local/spark/python/lib/py4j-0.10.9.3-src.zip/py4j/protocol.py", line 328, in get\_return\_value

py4j.protocol.Py4JJavaError: An error occurred while calling z:org.apache.spark.api.python.PythonRDD.runJob.

: org.apache.spark.SparkException: Job aborted due to stage failure: Task 0 in stage 118.0 failed 1 times, most recent failure: Lost task 0.0 in stage 118.0 (TID 137) (192.168.112.129 executor driver): org.apache.spark.api.python.PythonException: Traceback (most recent call last):

File "/usr/local/spark/python/lib/pyspark.zip/pyspark/worker.py", line 619, in main

process()

File "/usr/local/spark/python/lib/pyspark.zip/pyspark/worker.py", line 611, in process

serializer.dump\_stream(out\_iter, outfile)

File "/usr/local/spark/python/lib/pyspark.zip/pyspark/serializers.py", line 259, in dump\_stream

vs = list(itertools.islice(iterator, batch))

File "/usr/local/spark/python/pyspark/rdd.py", line 1562, in takeUpToNumLeft

yield next(iterator)

File "/usr/local/spark/python/lib/pyspark.zip/pyspark/util.py", line 74, in wrapper

return f(\*args, \*\*kwargs)

File "<stdin>", line 1, in <lambda>

AttributeError: 'bool' object has no attribute 'collect'

at org.apache.spark.api.python.BasePythonRunner$ReaderIterator.handlePythonException(PythonRunner.scala:555)

at org.apache.spark.api.python.PythonRunner$$anon$3.read(PythonRunner.scala:713)

at org.apache.spark.api.python.PythonRunner$$anon$3.read(PythonRunner.scala:695)

at org.apache.spark.api.python.BasePythonRunner$ReaderIterator.hasNext(PythonRunner.scala:508)

at org.apache.spark.InterruptibleIterator.hasNext(InterruptibleIterator.scala:37)

at scala.collection.Iterator.foreach(Iterator.scala:943)

at scala.collection.Iterator.foreach$(Iterator.scala:943)

at org.apache.spark.InterruptibleIterator.foreach(InterruptibleIterator.scala:28)

at scala.collection.generic.Growable.$plus$plus$eq(Growable.scala:62)

at scala.collection.generic.Growable.$plus$plus$eq$(Growable.scala:53)

at scala.collection.mutable.ArrayBuffer.$plus$plus$eq(ArrayBuffer.scala:105)

at scala.collection.mutable.ArrayBuffer.$plus$plus$eq(ArrayBuffer.scala:49)

at scala.collection.TraversableOnce.to(TraversableOnce.scala:366)

at scala.collection.TraversableOnce.to$(TraversableOnce.scala:364)

at org.apache.spark.InterruptibleIterator.to(InterruptibleIterator.scala:28)

at scala.collection.TraversableOnce.toBuffer(TraversableOnce.scala:358)

at scala.collection.TraversableOnce.toBuffer$(TraversableOnce.scala:358)

at org.apache.spark.InterruptibleIterator.toBuffer(InterruptibleIterator.scala:28)

at scala.collection.TraversableOnce.toArray(TraversableOnce.scala:345)

at scala.collection.TraversableOnce.toArray$(TraversableOnce.scala:339)

at org.apache.spark.InterruptibleIterator.toArray(InterruptibleIterator.scala:28)

at org.apache.spark.api.python.PythonRDD$.$anonfun$runJob$1(PythonRDD.scala:166)

at org.apache.spark.SparkContext.$anonfun$runJob$5(SparkContext.scala:2254)

at org.apache.spark.scheduler.ResultTask.runTask(ResultTask.scala:90)

at org.apache.spark.scheduler.Task.run(Task.scala:131)

at org.apache.spark.executor.Executor$TaskRunner.$anonfun$run$3(Executor.scala:506)

at org.apache.spark.util.Utils$.tryWithSafeFinally(Utils.scala:1462)

at org.apache.spark.executor.Executor$TaskRunner.run(Executor.scala:509)

at java.util.concurrent.ThreadPoolExecutor.runWorker(ThreadPoolExecutor.java:1149)

at java.util.concurrent.ThreadPoolExecutor$Worker.run(ThreadPoolExecutor.java:624)

at java.lang.Thread.run(Thread.java:750)

Driver stacktrace:

at org.apache.spark.scheduler.DAGScheduler.failJobAndIndependentStages(DAGScheduler.scala:2454)

at org.apache.spark.scheduler.DAGScheduler.$anonfun$abortStage$2(DAGScheduler.scala:2403)

at org.apache.spark.scheduler.DAGScheduler.$anonfun$abortStage$2$adapted(DAGScheduler.scala:2402)

at scala.collection.mutable.ResizableArray.foreach(ResizableArray.scala:62)

at scala.collection.mutable.ResizableArray.foreach$(ResizableArray.scala:55)

at scala.collection.mutable.ArrayBuffer.foreach(ArrayBuffer.scala:49)

at org.apache.spark.scheduler.DAGScheduler.abortStage(DAGScheduler.scala:2402)

at org.apache.spark.scheduler.DAGScheduler.$anonfun$handleTaskSetFailed$1(DAGScheduler.scala:1160)

at org.apache.spark.scheduler.DAGScheduler.$anonfun$handleTaskSetFailed$1$adapted(DAGScheduler.scala:1160)

at scala.Option.foreach(Option.scala:407)

at org.apache.spark.scheduler.DAGScheduler.handleTaskSetFailed(DAGScheduler.scala:1160)

at org.apache.spark.scheduler.DAGSchedulerEventProcessLoop.doOnReceive(DAGScheduler.scala:2642)

at org.apache.spark.scheduler.DAGSchedulerEventProcessLoop.onReceive(DAGScheduler.scala:2584)

at org.apache.spark.scheduler.DAGSchedulerEventProcessLoop.onReceive(DAGScheduler.scala:2573)

at org.apache.spark.util.EventLoop$$anon$1.run(EventLoop.scala:49)

at org.apache.spark.scheduler.DAGScheduler.runJob(DAGScheduler.scala:938)

at org.apache.spark.SparkContext.runJob(SparkContext.scala:2214)

at org.apache.spark.SparkContext.runJob(SparkContext.scala:2235)

at org.apache.spark.SparkContext.runJob(SparkContext.scala:2254)

at org.apache.spark.api.python.PythonRDD$.runJob(PythonRDD.scala:166)

at org.apache.spark.api.python.PythonRDD.runJob(PythonRDD.scala)

at sun.reflect.NativeMethodAccessorImpl.invoke0(Native Method)

at sun.reflect.NativeMethodAccessorImpl.invoke(NativeMethodAccessorImpl.java:62)

at sun.reflect.DelegatingMethodAccessorImpl.invoke(DelegatingMethodAccessorImpl.java:43)

at java.lang.reflect.Method.invoke(Method.java:498)

at py4j.reflection.MethodInvoker.invoke(MethodInvoker.java:244)

at py4j.reflection.ReflectionEngine.invoke(ReflectionEngine.java:357)

at py4j.Gateway.invoke(Gateway.java:282)

at py4j.commands.AbstractCommand.invokeMethod(AbstractCommand.java:132)

at py4j.commands.CallCommand.execute(CallCommand.java:79)

at py4j.ClientServerConnection.waitForCommands(ClientServerConnection.java:182)

at py4j.ClientServerConnection.run(ClientServerConnection.java:106)

at java.lang.Thread.run(Thread.java:750)

Caused by: org.apache.spark.api.python.PythonException: Traceback (most recent call last):

File "/usr/local/spark/python/lib/pyspark.zip/pyspark/worker.py", line 619, in main

process()

File "/usr/local/spark/python/lib/pyspark.zip/pyspark/worker.py", line 611, in process

serializer.dump\_stream(out\_iter, outfile)

File "/usr/local/spark/python/lib/pyspark.zip/pyspark/serializers.py", line 259, in dump\_stream

vs = list(itertools.islice(iterator, batch))

File "/usr/local/spark/python/pyspark/rdd.py", line 1562, in takeUpToNumLeft

yield next(iterator)

File "/usr/local/spark/python/lib/pyspark.zip/pyspark/util.py", line 74, in wrapper

return f(\*args, \*\*kwargs)

File "<stdin>", line 1, in <lambda>

AttributeError: 'bool' object has no attribute 'collect'

at org.apache.spark.api.python.BasePythonRunner$ReaderIterator.handlePythonException(PythonRunner.scala:555)

at org.apache.spark.api.python.PythonRunner$$anon$3.read(PythonRunner.scala:713)

at org.apache.spark.api.python.PythonRunner$$anon$3.read(PythonRunner.scala:695)

at org.apache.spark.api.python.BasePythonRunner$ReaderIterator.hasNext(PythonRunner.scala:508)

at org.apache.spark.InterruptibleIterator.hasNext(InterruptibleIterator.scala:37)

at scala.collection.Iterator.foreach(Iterator.scala:943)

at scala.collection.Iterator.foreach$(Iterator.scala:943)

at org.apache.spark.InterruptibleIterator.foreach(InterruptibleIterator.scala:28)

at scala.collection.generic.Growable.$plus$plus$eq(Growable.scala:62)

at scala.collection.generic.Growable.$plus$plus$eq$(Growable.scala:53)

at scala.collection.mutable.ArrayBuffer.$plus$plus$eq(ArrayBuffer.scala:105)

at scala.collection.mutable.ArrayBuffer.$plus$plus$eq(ArrayBuffer.scala:49)

at scala.collection.TraversableOnce.to(TraversableOnce.scala:366)

at scala.collection.TraversableOnce.to$(TraversableOnce.scala:364)

at org.apache.spark.InterruptibleIterator.to(InterruptibleIterator.scala:28)

at scala.collection.TraversableOnce.toBuffer(TraversableOnce.scala:358)

at scala.collection.TraversableOnce.toBuffer$(TraversableOnce.scala:358)

at org.apache.spark.InterruptibleIterator.toBuffer(InterruptibleIterator.scala:28)

at scala.collection.TraversableOnce.toArray(TraversableOnce.scala:345)

at scala.collection.TraversableOnce.toArray$(TraversableOnce.scala:339)

at org.apache.spark.InterruptibleIterator.toArray(InterruptibleIterator.scala:28)

at org.apache.spark.api.python.PythonRDD$.$anonfun$runJob$1(PythonRDD.scala:166)

at org.apache.spark.SparkContext.$anonfun$runJob$5(SparkContext.scala:2254)

at org.apache.spark.scheduler.ResultTask.runTask(ResultTask.scala:90)

at org.apache.spark.scheduler.Task.run(Task.scala:131)

at org.apache.spark.executor.Executor$TaskRunner.$anonfun$run$3(Executor.scala:506)

at org.apache.spark.util.Utils$.tryWithSafeFinally(Utils.scala:1462)

at org.apache.spark.executor.Executor$TaskRunner.run(Executor.scala:509)

at java.util.concurrent.ThreadPoolExecutor.runWorker(ThreadPoolExecutor.java:1149)

at java.util.concurrent.ThreadPoolExecutor$Worker.run(ThreadPoolExecutor.java:624)

... 1 more

>>> rdd.take(4)

['1,Kristina,Chung,55,Pilot', '2,Paige,Chen,74,Teacher', 'NM,New Mexico', 'NY,New York']

>>> custfilter =rdd.map(lambda x: x.split(",") )

>>> print(custfilter.take(14))

[['1', 'Kristina', 'Chung', '55', 'Pilot'], ['2', 'Paige', 'Chen', '74', 'Teacher'], ['NM', 'New Mexico'], ['NY', 'New York'], ['NC', 'North Carolina'], ['ND', 'North Dakota'], ['OH', 'Ohio'], ['OK', 'Oklahoma'], ['OR', 'Oregon'], ['MD', 'Maryland'], ['MA', 'Massachusetts'], ['MI', 'Michigan'], ['MN', 'Minnesota'], ['MS', 'Mississippi']]

>>> splited\_rdd =rdd.map(lambda x: x.split(",") )

>>> custfilter =splited\_rdd.filter(lambda x: len(x)>3 )

>>> print(custfilter.take(4))

[['1', 'Kristina', 'Chung', '55', 'Pilot'], ['2', 'Paige', 'Chen', '74', 'Teacher'], ['3', 'Sherri', 'Melton', '34', 'Firefighter'], ['4', 'Gretchen', 'Hill', '66', 'Computer hardware engineer']]

>>> print(custfilter.take(5))

[['1', 'Kristina', 'Chung', '55', 'Pilot'], ['2', 'Paige', 'Chen', '74', 'Teacher'], ['3', 'Sherri', 'Melton', '34', 'Firefighter'], ['4', 'Gretchen', 'Hill', '66', 'Computer hardware engineer'], ['5', 'Karen', 'Puckett', '74', 'Lawyer']]

>>> statefilter =splited\_rdd.filter(lambda x: len(x)<3 )

>>> print(statefilter.take(4))

[['NM', 'New Mexico'], ['NY', 'New York'], ['NC', 'North Carolina'], ['ND', 'North Dakota']]

>>>

>>> complete\_df=spark.read.csv("file:///home/hduser/Desktop/Hackathon/custs\_states.csv",sep=",",header=False,inferSchema=True)

>>> complete\_df.show(3)

+---+----------+-----+----+-------+

|\_c0| \_c1| \_c2| \_c3| \_c4|

+---+----------+-----+----+-------+

| 1| Kristina|Chung| 55| Pilot|

| 2| Paige| Chen| 74|Teacher|

| NM|New Mexico| null|null| null|

+---+----------+-----+----+-------+

only showing top 3 rows

>>> custfilter\_df=complete\_df.select("\*").where("\_c0 is not null and \_c1 is not null and \_c2 is not null and \_c3 is not null and \_c4 is not null ")

>>> cust

custfilter custfilter\_df custom\_schema

>>> custfilter\_df.show(3)

+---+--------+------+---+-----------+

|\_c0| \_c1| \_c2|\_c3| \_c4|

+---+--------+------+---+-----------+

| 1|Kristina| Chung| 55| Pilot|

| 2| Paige| Chen| 74| Teacher|

| 3| Sherri|Melton| 34|Firefighter|

+---+--------+------+---+-----------+

only showing top 3 rows

>>> statefilter\_df=complete\_df.select("\_c0,\_c1").where(" \_c2 is null and \_c3 is null and \_c4 is null ")

Traceback (most recent call last):

File "<stdin>", line 1, in <module>

File "/usr/local/spark/python/pyspark/sql/dataframe.py", line 1685, in select

jdf = self.\_jdf.select(self.\_jcols(\*cols))

File "/usr/local/spark/python/lib/py4j-0.10.9.3-src.zip/py4j/java\_gateway.py", line 1322, in \_\_call\_\_

File "/usr/local/spark/python/pyspark/sql/utils.py", line 117, in deco

raise converted from None

pyspark.sql.utils.AnalysisException: cannot resolve '`\_c0,\_c1`' given input columns: [\_c0, \_c1, \_c2, \_c3, \_c4];

'Project ['\_c0,\_c1]

+- Relation [\_c0#26007,\_c1#26008,\_c2#26009,\_c3#26010,\_c4#26011] csv

>>> statefilter\_df=complete\_df.select("\_c0","\_c1").where(" \_c2 is null and \_c3 is null and \_c4 is null ")

>>> statefilter\_df.show(4)

+---+--------------+

|\_c0| \_c1|

+---+--------------+

| NM| New Mexico|

| NY| New York|

| NC|North Carolina|

| ND| North Dakota|

+---+--------------+

only showing top 4 rows

>>> custfilter\_df.createOrReplaceTempView("cust\_table")

>>> statefilter\_df.createOrReplaceTempView("state\_table")

>>> merged\_data\_frame.createOrReplaceTempView("insureview")

>>> df\_function("A+wer")

Column<'clean\_string(A+wer)'>

>>> df\_function("A+wer").show()

Traceback (most recent call last):

File "<stdin>", line 1, in <module>

TypeError: 'Column' object is not callable

>>> print(df\_function("A+wer"))

Column<'clean\_string(A+wer)'>

>>> print(df\_function(A+wer))

Traceback (most recent call last):

File "<stdin>", line 1, in <module>

NameError: name 'A' is not defined

>>> print(df\_function("A+wer"))

Column<'clean\_string(A+wer)'>

>>> print(df\_function("wer"))

Column<'clean\_string(wer)'>

>>> spark.conf.set("spark.sql.shuffle.partitions", 4)

>>> spark.udf.register("sql\_function", clean\_string, StringType())

<function clean\_string at 0x7fcc0c22b510>

>>> spark.sql("select \* from cust\_table").show(3)

+---+--------+------+---+-----------+

|\_c0| \_c1| \_c2|\_c3| \_c4|

+---+--------+------+---+-----------+

| 1|Kristina| Chung| 55| Pilot|

| 2| Paige| Chen| 74| Teacher|

| 3| Sherri|Melton| 34|Firefighter|

+---+--------+------+---+-----------+

only showing top 3 rows

>>> spark.sql("select \* from insureview").show(3)

+------------+----+-----+------------+--------------------+-------+--------------+--------------------+--------+---------+-----------------+----------+--------------------+

|BusinessDate|stcd|srcnm| NetworkName| NetworkURL|custnum|MarketCoverage| RejectRows|IssuerId|IssuerId2|issueridcomposite| sysdt| systs|

+------------+----+-----+------------+--------------------+-------+--------------+--------------------+--------+---------+-----------------+----------+--------------------+

| 2019-10-01| AK| HIOS| ODS Premier|https://www.modah...| 13| null| null| 21989| 21989| 2198921989|2024-07-22|2024-07-22 11:39:...|

| 2019-10-01| AK| HIOS| ODS Premier| null| null| null|219,219,2019-10-0...| 219| 219| 219219|2024-07-22|2024-07-22 11:39:...|

| 2019-10-01| AK| HIOS|HeritagePlus|https://www.preme...| 13| null| null| 38344| 38344| 3834438344|2024-07-22|2024-07-22 11:39:...|

+------------+----+-----+------------+--------------------+-------+--------------+--------------------+--------+---------+-----------------+----------+--------------------+

only showing top 3 rows

>>> spark.sql("select \*,sql\_function(NetworkName) as cleannetworkname from insureview").show(3)

+------------+----+-----+------------+--------------------+-------+--------------+--------------------+--------+---------+-----------------+----------+--------------------+----------------+

|BusinessDate|stcd|srcnm| NetworkName| NetworkURL|custnum|MarketCoverage| RejectRows|IssuerId|IssuerId2|issueridcomposite| sysdt| systs|cleannetworkname|

+------------+----+-----+------------+--------------------+-------+--------------+--------------------+--------+---------+-----------------+----------+--------------------+----------------+

| 2019-10-01| AK| HIOS| ODS Premier|https://www.modah...| 13| null| null| 21989| 21989| 2198921989|2024-07-22|2024-07-22 11:40:...| ODS Premier|

| 2019-10-01| AK| HIOS| ODS Premier| null| null| null|219,219,2019-10-0...| 219| 219| 219219|2024-07-22|2024-07-22 11:40:...| ODS Premier|

| 2019-10-01| AK| HIOS|HeritagePlus|https://www.preme...| 13| null| null| 38344| 38344| 3834438344|2024-07-22|2024-07-22 11:40:...| HeritagePlus|

+------------+----+-----+------------+--------------------+-------+--------------+--------------------+--------+---------+-----------------+----------+--------------------+----------------+

only showing top 3 rows

>>> spark.sql("select \*,current\_date as curdt , current\_timestamp as curts from insureview").show(3)

+------------+----+-----+------------+--------------------+-------+--------------+--------------------+--------+---------+-----------------+----------+--------------------+----------+--------------------+

|BusinessDate|stcd|srcnm| NetworkName| NetworkURL|custnum|MarketCoverage| RejectRows|IssuerId|IssuerId2|issueridcomposite| sysdt| systs| curdt| curts|

+------------+----+-----+------------+--------------------+-------+--------------+--------------------+--------+---------+-----------------+----------+--------------------+----------+--------------------+

| 2019-10-01| AK| HIOS| ODS Premier|https://www.modah...| 13| null| null| 21989| 21989| 2198921989|2024-07-22|2024-07-22 11:42:...|2024-07-22|2024-07-22 11:42:...|

| 2019-10-01| AK| HIOS| ODS Premier| null| null| null|219,219,2019-10-0...| 219| 219| 219219|2024-07-22|2024-07-22 11:42:...|2024-07-22|2024-07-22 11:42:...|

| 2019-10-01| AK| HIOS|HeritagePlus|https://www.preme...| 13| null| null| 38344| 38344| 3834438344|2024-07-22|2024-07-22 11:42:...|2024-07-22|2024-07-22 11:42:...|

+------------+----+-----+------------+--------------------+-------+--------------+--------------------+--------+---------+-----------------+----------+--------------------+----------+--------------------+

only showing top 3 rows

>>> spark.sql("select \*,BuisnessDate.month as month , BuisnessDate.year as year from insureview").show(3)

Traceback (most recent call last):

File "<stdin>", line 1, in <module>

File "/usr/local/spark/python/pyspark/sql/session.py", line 723, in sql

return DataFrame(self.\_jsparkSession.sql(sqlQuery), self.\_wrapped)

File "/usr/local/spark/python/lib/py4j-0.10.9.3-src.zip/py4j/java\_gateway.py", line 1322, in \_\_call\_\_

File "/usr/local/spark/python/pyspark/sql/utils.py", line 117, in deco

raise converted from None

pyspark.sql.utils.AnalysisException: cannot resolve 'BuisnessDate.month' given input columns: [insureview.BusinessDate, insureview.IssuerId, insureview.IssuerId2, insureview.MarketCoverage, insureview.NetworkName, insureview.NetworkURL, insureview.RejectRows, insureview.custnum, insureview.issueridcomposite, insureview.srcnm, insureview.stcd, insureview.sysdt, insureview.systs]; line 1 pos 9;

'Project [BusinessDate#1378, stcd#7759, srcnm#7747, NetworkName#1381, NetworkURL#1382, custnum#1383, MarketCoverage#1384, RejectRows#1386, IssuerId#8312, IssuerId2#8325, issueridcomposite#8901, sysdt#9003, systs#9016, 'BuisnessDate.month AS month#27694, 'BuisnessDate.year AS year#27695]

+- SubqueryAlias insureview

+- View (`insureview`, [BusinessDate#1378,stcd#7759,srcnm#7747,NetworkName#1381,NetworkURL#1382,custnum#1383,MarketCoverage#1384,RejectRows#1386,IssuerId#8312,IssuerId2#8325,issueridcomposite#8901,sysdt#9003,systs#9016])

+- Project [BusinessDate#1378, stcd#7759, srcnm#7747, NetworkName#1381, NetworkURL#1382, custnum#1383, MarketCoverage#1384, RejectRows#1386, IssuerId#8312, IssuerId2#8325, issueridcomposite#8901, sysdt#9003, current\_timestamp() AS systs#9016]

+- Project [BusinessDate#1378, stcd#7759, srcnm#7747, NetworkName#1381, NetworkURL#1382, custnum#1383, MarketCoverage#1384, RejectRows#1386, IssuerId#8312, IssuerId2#8325, issueridcomposite#8901, current\_date(Some(Asia/Kolkata)) AS sysdt#9003]

+- Project [BusinessDate#1378, stcd#7759, srcnm#7747, NetworkName#1381, NetworkURL#1382, custnum#1383, MarketCoverage#1384, RejectRows#1386, IssuerId#8312, IssuerId2#8325, issueridcomposite#8901]

+- Project [BusinessDate#1378, stcd#7759, srcnm#7747, NetworkName#1381, NetworkURL#1382, custnum#1383, MarketCoverage#1384, DentalOnlyPlan#1385, RejectRows#1386, IssuerId#8312, IssuerId2#8325, concat(IssuerId#8312, IssuerId2#8325) AS issueridcomposite#8901]

+- Project [BusinessDate#1378, stcd#7759, srcnm#7747, NetworkName#1381, NetworkURL#1382, custnum#1383, MarketCoverage#1384, DentalOnlyPlan#1385, RejectRows#1386, IssuerId#8312, IssuerId2#8325]

+- Project [IssuerIDDD#8273, BusinessDate#1378, stcd#7759, srcnm#7747, NetworkName#1381, NetworkURL#1382, custnum#1383, MarketCoverage#1384, DentalOnlyPlan#1385, RejectRows#1386, IssuerId#8312, IssuerId2#8325]

+- Project [IssuerIDD#8261, IssuerIDDD#8273, BusinessDate#1378, stcd#7759, srcnm#7747, NetworkName#1381, NetworkURL#1382, custnum#1383, MarketCoverage#1384, DentalOnlyPlan#1385, RejectRows#1386, IssuerId#8312, cast(IssuerIDDD#8273 as string) AS IssuerId2#8325]

+- Project [IssuerIDD#8261, IssuerIDDD#8273, BusinessDate#1378, stcd#7759, srcnm#7747, NetworkName#1381, NetworkURL#1382, custnum#1383, MarketCoverage#1384, DentalOnlyPlan#1385, RejectRows#1386, cast(IssuerIDD#8261 as string) AS IssuerId#8312]

+- Project [IssuerIDD#8261, IssuerId2#1377 AS IssuerIDDD#8273, BusinessDate#1378, stcd#7759, srcnm#7747, NetworkName#1381, NetworkURL#1382, custnum#1383, MarketCoverage#1384, DentalOnlyPlan#1385, RejectRows#1386]

+- Project [IssuerId#1376 AS IssuerIDD#8261, IssuerId2#1377, BusinessDate#1378, stcd#7759, srcnm#7747, NetworkName#1381, NetworkURL#1382, custnum#1383, MarketCoverage#1384, DentalOnlyPlan#1385, RejectRows#1386]

+- Project [IssuerId#1376, IssuerId2#1377, BusinessDate#1378, StateCode#1379 AS stcd#7759, srcnm#7747, NetworkName#1381, NetworkURL#1382, custnum#1383, MarketCoverage#1384, DentalOnlyPlan#1385, RejectRows#1386]

+- Project [IssuerId#1376, IssuerId2#1377, BusinessDate#1378, StateCode#1379, SourceName#1380 AS srcnm#7747, NetworkName#1381, NetworkURL#1382, custnum#1383, MarketCoverage#1384, DentalOnlyPlan#1385, RejectRows#1386]

+- Union false, false

:- Relation [IssuerId#1376,IssuerId2#1377,BusinessDate#1378,StateCode#1379,SourceName#1380,NetworkName#1381,NetworkURL#1382,custnum#1383,MarketCoverage#1384,DentalOnlyPlan#1385,RejectRows#1386] csv

+- Relation [IssuerId#491,IssuerId2#492,BusinessDate#493,StateCode#494,SourceName#495,NetworkName#496,NetworkURL#497,custnum#498,MarketCoverage#499,DentalOnlyPlan#500,RejectRows#501] csv

>>> spark.sql("select \*,BuisnessDate.month() as month , BuisnessDate.year() as year from insureview").show(3)

24/07/22 11:43:25 WARN metastore.ObjectStore: Failed to get database buisnessdate, returning NoSuchObjectException

Traceback (most recent call last):

File "<stdin>", line 1, in <module>

File "/usr/local/spark/python/pyspark/sql/session.py", line 723, in sql

return DataFrame(self.\_jsparkSession.sql(sqlQuery), self.\_wrapped)

File "/usr/local/spark/python/lib/py4j-0.10.9.3-src.zip/py4j/java\_gateway.py", line 1322, in \_\_call\_\_

File "/usr/local/spark/python/pyspark/sql/utils.py", line 117, in deco

raise converted from None

pyspark.sql.utils.AnalysisException: Undefined function: 'month'. This function is neither a registered temporary function nor a permanent function registered in the database 'BuisnessDate'.; line 1 pos 9

>>> spark.sql("select \*,month(BuisnessDate) as month ,year(BuisnessDate) as year from insureview").show(3)

Traceback (most recent call last):

File "<stdin>", line 1, in <module>

File "/usr/local/spark/python/pyspark/sql/session.py", line 723, in sql

return DataFrame(self.\_jsparkSession.sql(sqlQuery), self.\_wrapped)

File "/usr/local/spark/python/lib/py4j-0.10.9.3-src.zip/py4j/java\_gateway.py", line 1322, in \_\_call\_\_

File "/usr/local/spark/python/pyspark/sql/utils.py", line 117, in deco

raise converted from None

pyspark.sql.utils.AnalysisException: cannot resolve 'BuisnessDate' given input columns: [insureview.BusinessDate, insureview.IssuerId, insureview.IssuerId2, insureview.MarketCoverage, insureview.NetworkName, insureview.NetworkURL, insureview.RejectRows, insureview.custnum, insureview.issueridcomposite, insureview.srcnm, insureview.stcd, insureview.sysdt, insureview.systs]; line 1 pos 15;

'Project [BusinessDate#1378, stcd#7759, srcnm#7747, NetworkName#1381, NetworkURL#1382, custnum#1383, MarketCoverage#1384, RejectRows#1386, IssuerId#8312, IssuerId2#8325, issueridcomposite#8901, sysdt#9003, systs#9016, 'month('BuisnessDate) AS month#27698, 'year('BuisnessDate) AS year#27699]

+- SubqueryAlias insureview

+- View (`insureview`, [BusinessDate#1378,stcd#7759,srcnm#7747,NetworkName#1381,NetworkURL#1382,custnum#1383,MarketCoverage#1384,RejectRows#1386,IssuerId#8312,IssuerId2#8325,issueridcomposite#8901,sysdt#9003,systs#9016])

+- Project [BusinessDate#1378, stcd#7759, srcnm#7747, NetworkName#1381, NetworkURL#1382, custnum#1383, MarketCoverage#1384, RejectRows#1386, IssuerId#8312, IssuerId2#8325, issueridcomposite#8901, sysdt#9003, current\_timestamp() AS systs#9016]

+- Project [BusinessDate#1378, stcd#7759, srcnm#7747, NetworkName#1381, NetworkURL#1382, custnum#1383, MarketCoverage#1384, RejectRows#1386, IssuerId#8312, IssuerId2#8325, issueridcomposite#8901, current\_date(Some(Asia/Kolkata)) AS sysdt#9003]

+- Project [BusinessDate#1378, stcd#7759, srcnm#7747, NetworkName#1381, NetworkURL#1382, custnum#1383, MarketCoverage#1384, RejectRows#1386, IssuerId#8312, IssuerId2#8325, issueridcomposite#8901]

+- Project [BusinessDate#1378, stcd#7759, srcnm#7747, NetworkName#1381, NetworkURL#1382, custnum#1383, MarketCoverage#1384, DentalOnlyPlan#1385, RejectRows#1386, IssuerId#8312, IssuerId2#8325, concat(IssuerId#8312, IssuerId2#8325) AS issueridcomposite#8901]

+- Project [BusinessDate#1378, stcd#7759, srcnm#7747, NetworkName#1381, NetworkURL#1382, custnum#1383, MarketCoverage#1384, DentalOnlyPlan#1385, RejectRows#1386, IssuerId#8312, IssuerId2#8325]

+- Project [IssuerIDDD#8273, BusinessDate#1378, stcd#7759, srcnm#7747, NetworkName#1381, NetworkURL#1382, custnum#1383, MarketCoverage#1384, DentalOnlyPlan#1385, RejectRows#1386, IssuerId#8312, IssuerId2#8325]

+- Project [IssuerIDD#8261, IssuerIDDD#8273, BusinessDate#1378, stcd#7759, srcnm#7747, NetworkName#1381, NetworkURL#1382, custnum#1383, MarketCoverage#1384, DentalOnlyPlan#1385, RejectRows#1386, IssuerId#8312, cast(IssuerIDDD#8273 as string) AS IssuerId2#8325]

+- Project [IssuerIDD#8261, IssuerIDDD#8273, BusinessDate#1378, stcd#7759, srcnm#7747, NetworkName#1381, NetworkURL#1382, custnum#1383, MarketCoverage#1384, DentalOnlyPlan#1385, RejectRows#1386, cast(IssuerIDD#8261 as string) AS IssuerId#8312]

+- Project [IssuerIDD#8261, IssuerId2#1377 AS IssuerIDDD#8273, BusinessDate#1378, stcd#7759, srcnm#7747, NetworkName#1381, NetworkURL#1382, custnum#1383, MarketCoverage#1384, DentalOnlyPlan#1385, RejectRows#1386]

+- Project [IssuerId#1376 AS IssuerIDD#8261, IssuerId2#1377, BusinessDate#1378, stcd#7759, srcnm#7747, NetworkName#1381, NetworkURL#1382, custnum#1383, MarketCoverage#1384, DentalOnlyPlan#1385, RejectRows#1386]

+- Project [IssuerId#1376, IssuerId2#1377, BusinessDate#1378, StateCode#1379 AS stcd#7759, srcnm#7747, NetworkName#1381, NetworkURL#1382, custnum#1383, MarketCoverage#1384, DentalOnlyPlan#1385, RejectRows#1386]

+- Project [IssuerId#1376, IssuerId2#1377, BusinessDate#1378, StateCode#1379, SourceName#1380 AS srcnm#7747, NetworkName#1381, NetworkURL#1382, custnum#1383, MarketCoverage#1384, DentalOnlyPlan#1385, RejectRows#1386]

+- Union false, false

:- Relation [IssuerId#1376,IssuerId2#1377,BusinessDate#1378,StateCode#1379,SourceName#1380,NetworkName#1381,NetworkURL#1382,custnum#1383,MarketCoverage#1384,DentalOnlyPlan#1385,RejectRows#1386] csv

+- Relation [IssuerId#491,IssuerId2#492,BusinessDate#493,StateCode#494,SourceName#495,NetworkName#496,NetworkURL#497,custnum#498,MarketCoverage#499,DentalOnlyPlan#500,RejectRows#501] csv

>>> spark.sql("select \*,month(BuisnessDate) as month ,year(BuisnessDate) as year from insureview").show(3

...

Traceback (most recent call last):

File "/usr/local/spark/python/pyspark/context.py", line 293, in signal\_handler

raise KeyboardInterrupt()

KeyboardInterrupt

>>> spark.sql("describe insureview")

DataFrame[col\_name: string, data\_type: string, comment: string]

>>> spark.sql("describe insureview").show()

+-----------------+---------+-------+

| col\_name|data\_type|comment|

+-----------------+---------+-------+

| BusinessDate| date| null|

| stcd| string| null|

| srcnm| string| null|

| NetworkName| string| null|

| NetworkURL| string| null|

| custnum| string| null|

| MarketCoverage| string| null|

| RejectRows| string| null|

| IssuerId| string| null|

| IssuerId2| string| null|

|issueridcomposite| string| null|

| sysdt| date| null|

| systs|timestamp| null|

+-----------------+---------+-------+

>>> spark.sql("""select \*,month(BuisnessDate) as month ,year(BuisnessDate) as year from insureview""").show(3

... )

Traceback (most recent call last):

File "<stdin>", line 1, in <module>

File "/usr/local/spark/python/pyspark/sql/session.py", line 723, in sql

return DataFrame(self.\_jsparkSession.sql(sqlQuery), self.\_wrapped)

File "/usr/local/spark/python/lib/py4j-0.10.9.3-src.zip/py4j/java\_gateway.py", line 1322, in \_\_call\_\_

File "/usr/local/spark/python/pyspark/sql/utils.py", line 117, in deco

raise converted from None

pyspark.sql.utils.AnalysisException: cannot resolve 'BuisnessDate' given input columns: [insureview.BusinessDate, insureview.IssuerId, insureview.IssuerId2, insureview.MarketCoverage, insureview.NetworkName, insureview.NetworkURL, insureview.RejectRows, insureview.custnum, insureview.issueridcomposite, insureview.srcnm, insureview.stcd, insureview.sysdt, insureview.systs]; line 1 pos 15;

'Project [BusinessDate#1378, stcd#7759, srcnm#7747, NetworkName#1381, NetworkURL#1382, custnum#1383, MarketCoverage#1384, RejectRows#1386, IssuerId#8312, IssuerId2#8325, issueridcomposite#8901, sysdt#9003, systs#9016, 'month('BuisnessDate) AS month#27725, 'year('BuisnessDate) AS year#27726]

+- SubqueryAlias insureview

+- View (`insureview`, [BusinessDate#1378,stcd#7759,srcnm#7747,NetworkName#1381,NetworkURL#1382,custnum#1383,MarketCoverage#1384,RejectRows#1386,IssuerId#8312,IssuerId2#8325,issueridcomposite#8901,sysdt#9003,systs#9016])

+- Project [BusinessDate#1378, stcd#7759, srcnm#7747, NetworkName#1381, NetworkURL#1382, custnum#1383, MarketCoverage#1384, RejectRows#1386, IssuerId#8312, IssuerId2#8325, issueridcomposite#8901, sysdt#9003, current\_timestamp() AS systs#9016]

+- Project [BusinessDate#1378, stcd#7759, srcnm#7747, NetworkName#1381, NetworkURL#1382, custnum#1383, MarketCoverage#1384, RejectRows#1386, IssuerId#8312, IssuerId2#8325, issueridcomposite#8901, current\_date(Some(Asia/Kolkata)) AS sysdt#9003]

+- Project [BusinessDate#1378, stcd#7759, srcnm#7747, NetworkName#1381, NetworkURL#1382, custnum#1383, MarketCoverage#1384, RejectRows#1386, IssuerId#8312, IssuerId2#8325, issueridcomposite#8901]

+- Project [BusinessDate#1378, stcd#7759, srcnm#7747, NetworkName#1381, NetworkURL#1382, custnum#1383, MarketCoverage#1384, DentalOnlyPlan#1385, RejectRows#1386, IssuerId#8312, IssuerId2#8325, concat(IssuerId#8312, IssuerId2#8325) AS issueridcomposite#8901]

+- Project [BusinessDate#1378, stcd#7759, srcnm#7747, NetworkName#1381, NetworkURL#1382, custnum#1383, MarketCoverage#1384, DentalOnlyPlan#1385, RejectRows#1386, IssuerId#8312, IssuerId2#8325]

+- Project [IssuerIDDD#8273, BusinessDate#1378, stcd#7759, srcnm#7747, NetworkName#1381, NetworkURL#1382, custnum#1383, MarketCoverage#1384, DentalOnlyPlan#1385, RejectRows#1386, IssuerId#8312, IssuerId2#8325]

+- Project [IssuerIDD#8261, IssuerIDDD#8273, BusinessDate#1378, stcd#7759, srcnm#7747, NetworkName#1381, NetworkURL#1382, custnum#1383, MarketCoverage#1384, DentalOnlyPlan#1385, RejectRows#1386, IssuerId#8312, cast(IssuerIDDD#8273 as string) AS IssuerId2#8325]

+- Project [IssuerIDD#8261, IssuerIDDD#8273, BusinessDate#1378, stcd#7759, srcnm#7747, NetworkName#1381, NetworkURL#1382, custnum#1383, MarketCoverage#1384, DentalOnlyPlan#1385, RejectRows#1386, cast(IssuerIDD#8261 as string) AS IssuerId#8312]

+- Project [IssuerIDD#8261, IssuerId2#1377 AS IssuerIDDD#8273, BusinessDate#1378, stcd#7759, srcnm#7747, NetworkName#1381, NetworkURL#1382, custnum#1383, MarketCoverage#1384, DentalOnlyPlan#1385, RejectRows#1386]

+- Project [IssuerId#1376 AS IssuerIDD#8261, IssuerId2#1377, BusinessDate#1378, stcd#7759, srcnm#7747, NetworkName#1381, NetworkURL#1382, custnum#1383, MarketCoverage#1384, DentalOnlyPlan#1385, RejectRows#1386]

+- Project [IssuerId#1376, IssuerId2#1377, BusinessDate#1378, StateCode#1379 AS stcd#7759, srcnm#7747, NetworkName#1381, NetworkURL#1382, custnum#1383, MarketCoverage#1384, DentalOnlyPlan#1385, RejectRows#1386]

+- Project [IssuerId#1376, IssuerId2#1377, BusinessDate#1378, StateCode#1379, SourceName#1380 AS srcnm#7747, NetworkName#1381, NetworkURL#1382, custnum#1383, MarketCoverage#1384, DentalOnlyPlan#1385, RejectRows#1386]

+- Union false, false

:- Relation [IssuerId#1376,IssuerId2#1377,BusinessDate#1378,StateCode#1379,SourceName#1380,NetworkName#1381,NetworkURL#1382,custnum#1383,MarketCoverage#1384,DentalOnlyPlan#1385,RejectRows#1386] csv

+- Relation [IssuerId#491,IssuerId2#492,BusinessDate#493,StateCode#494,SourceName#495,NetworkName#496,NetworkURL#497,custnum#498,MarketCoverage#499,DentalOnlyPlan#500,RejectRows#501] csv

>>> spark.sql("""select \*,month(BuisnessDate) as month ,year(BuisnessDate) as year from insureview""").show(3)

Traceback (most recent call last):

File "<stdin>", line 1, in <module>

File "/usr/local/spark/python/pyspark/sql/session.py", line 723, in sql

return DataFrame(self.\_jsparkSession.sql(sqlQuery), self.\_wrapped)

File "/usr/local/spark/python/lib/py4j-0.10.9.3-src.zip/py4j/java\_gateway.py", line 1322, in \_\_call\_\_

File "/usr/local/spark/python/pyspark/sql/utils.py", line 117, in deco

raise converted from None

pyspark.sql.utils.AnalysisException: cannot resolve 'BuisnessDate' given input columns: [insureview.BusinessDate, insureview.IssuerId, insureview.IssuerId2, insureview.MarketCoverage, insureview.NetworkName, insureview.NetworkURL, insureview.RejectRows, insureview.custnum, insureview.issueridcomposite, insureview.srcnm, insureview.stcd, insureview.sysdt, insureview.systs]; line 1 pos 15;

'Project [BusinessDate#1378, stcd#7759, srcnm#7747, NetworkName#1381, NetworkURL#1382, custnum#1383, MarketCoverage#1384, RejectRows#1386, IssuerId#8312, IssuerId2#8325, issueridcomposite#8901, sysdt#9003, systs#9016, 'month('BuisnessDate) AS month#27727, 'year('BuisnessDate) AS year#27728]

+- SubqueryAlias insureview

+- View (`insureview`, [BusinessDate#1378,stcd#7759,srcnm#7747,NetworkName#1381,NetworkURL#1382,custnum#1383,MarketCoverage#1384,RejectRows#1386,IssuerId#8312,IssuerId2#8325,issueridcomposite#8901,sysdt#9003,systs#9016])

+- Project [BusinessDate#1378, stcd#7759, srcnm#7747, NetworkName#1381, NetworkURL#1382, custnum#1383, MarketCoverage#1384, RejectRows#1386, IssuerId#8312, IssuerId2#8325, issueridcomposite#8901, sysdt#9003, current\_timestamp() AS systs#9016]

+- Project [BusinessDate#1378, stcd#7759, srcnm#7747, NetworkName#1381, NetworkURL#1382, custnum#1383, MarketCoverage#1384, RejectRows#1386, IssuerId#8312, IssuerId2#8325, issueridcomposite#8901, current\_date(Some(Asia/Kolkata)) AS sysdt#9003]

+- Project [BusinessDate#1378, stcd#7759, srcnm#7747, NetworkName#1381, NetworkURL#1382, custnum#1383, MarketCoverage#1384, RejectRows#1386, IssuerId#8312, IssuerId2#8325, issueridcomposite#8901]

+- Project [BusinessDate#1378, stcd#7759, srcnm#7747, NetworkName#1381, NetworkURL#1382, custnum#1383, MarketCoverage#1384, DentalOnlyPlan#1385, RejectRows#1386, IssuerId#8312, IssuerId2#8325, concat(IssuerId#8312, IssuerId2#8325) AS issueridcomposite#8901]

+- Project [BusinessDate#1378, stcd#7759, srcnm#7747, NetworkName#1381, NetworkURL#1382, custnum#1383, MarketCoverage#1384, DentalOnlyPlan#1385, RejectRows#1386, IssuerId#8312, IssuerId2#8325]

+- Project [IssuerIDDD#8273, BusinessDate#1378, stcd#7759, srcnm#7747, NetworkName#1381, NetworkURL#1382, custnum#1383, MarketCoverage#1384, DentalOnlyPlan#1385, RejectRows#1386, IssuerId#8312, IssuerId2#8325]

+- Project [IssuerIDD#8261, IssuerIDDD#8273, BusinessDate#1378, stcd#7759, srcnm#7747, NetworkName#1381, NetworkURL#1382, custnum#1383, MarketCoverage#1384, DentalOnlyPlan#1385, RejectRows#1386, IssuerId#8312, cast(IssuerIDDD#8273 as string) AS IssuerId2#8325]

+- Project [IssuerIDD#8261, IssuerIDDD#8273, BusinessDate#1378, stcd#7759, srcnm#7747, NetworkName#1381, NetworkURL#1382, custnum#1383, MarketCoverage#1384, DentalOnlyPlan#1385, RejectRows#1386, cast(IssuerIDD#8261 as string) AS IssuerId#8312]

+- Project [IssuerIDD#8261, IssuerId2#1377 AS IssuerIDDD#8273, BusinessDate#1378, stcd#7759, srcnm#7747, NetworkName#1381, NetworkURL#1382, custnum#1383, MarketCoverage#1384, DentalOnlyPlan#1385, RejectRows#1386]

+- Project [IssuerId#1376 AS IssuerIDD#8261, IssuerId2#1377, BusinessDate#1378, stcd#7759, srcnm#7747, NetworkName#1381, NetworkURL#1382, custnum#1383, MarketCoverage#1384, DentalOnlyPlan#1385, RejectRows#1386]

+- Project [IssuerId#1376, IssuerId2#1377, BusinessDate#1378, StateCode#1379 AS stcd#7759, srcnm#7747, NetworkName#1381, NetworkURL#1382, custnum#1383, MarketCoverage#1384, DentalOnlyPlan#1385, RejectRows#1386]

+- Project [IssuerId#1376, IssuerId2#1377, BusinessDate#1378, StateCode#1379, SourceName#1380 AS srcnm#7747, NetworkName#1381, NetworkURL#1382, custnum#1383, MarketCoverage#1384, DentalOnlyPlan#1385, RejectRows#1386]

+- Union false, false

:- Relation [IssuerId#1376,IssuerId2#1377,BusinessDate#1378,StateCode#1379,SourceName#1380,NetworkName#1381,NetworkURL#1382,custnum#1383,MarketCoverage#1384,DentalOnlyPlan#1385,RejectRows#1386] csv

+- Relation [IssuerId#491,IssuerId2#492,BusinessDate#493,StateCode#494,SourceName#495,NetworkName#496,NetworkURL#497,custnum#498,MarketCoverage#499,DentalOnlyPlan#500,RejectRows#501] csv

>>> spark.sql("""

... SELECT \*,

... month(BusinessDate) AS month,

... year(BusinessDate) AS year

... FROM insureview

... """)

DataFrame[BusinessDate: date, stcd: string, srcnm: string, NetworkName: string, NetworkURL: string, custnum: string, MarketCoverage: string, RejectRows: string, IssuerId: string, IssuerId2: string, issueridcomposite: string, sysdt: date, systs: timestamp, month: int, year: int]

>>> result\_df.show(3)

Traceback (most recent call last):

File "<stdin>", line 1, in <module>

NameError: name 'result\_df' is not defined

>>> spark.sql("""spark.sql("""SELECT \*, month(BusinessDate) AS month, year(BusinessDate) AS year FROM insureview""")

File "<stdin>", line 1

spark.sql("""spark.sql("""SELECT \*, month(BusinessDate) AS month, year(BusinessDate) AS year FROM insureview""")

^

SyntaxError: invalid syntax

>>> spark.sql("""SELECT \*, month(BusinessDate) AS month, year(BusinessDate) AS year FROM insureview""")

DataFrame[BusinessDate: date, stcd: string, srcnm: string, NetworkName: string, NetworkURL: string, custnum: string, MarketCoverage: string, RejectRows: string, IssuerId: string, IssuerId2: string, issueridcomposite: string, sysdt: date, systs: timestamp, month: int, year: int]

>>> spark.sql("""SELECT \*, month(BusinessDate) AS month, year(BusinessDate) AS year FROM insureview""").show()

+------------+----+-----+--------------------+--------------------+-------+--------------+--------------------+--------+---------+-----------------+----------+--------------------+-----+----+

|BusinessDate|stcd|srcnm| NetworkName| NetworkURL|custnum|MarketCoverage| RejectRows|IssuerId|IssuerId2|issueridcomposite| sysdt| systs|month|year|

+------------+----+-----+--------------------+--------------------+-------+--------------+--------------------+--------+---------+-----------------+----------+--------------------+-----+----+

| 2019-10-01| AK| HIOS| ODS Premier|https://www.modah...| 13| null| null| 21989| 21989| 2198921989|2024-07-22|2024-07-22 11:47:...| 10|2019|

| 2019-10-01| AK| HIOS| ODS Premier| null| null| null|219,219,2019-10-0...| 219| 219| 219219|2024-07-22|2024-07-22 11:47:...| 10|2019|

| 2019-10-01| AK| HIOS| HeritagePlus|https://www.preme...| 13| null| null| 38344| 38344| 3834438344|2024-07-22|2024-07-22 11:47:...| 10|2019|

| 2019-10-01| AK| HIOS|Lincoln Dental Co...|http://lfg.go2den...| 13| null| null| 38536| 38536| 3853638536|2024-07-22|2024-07-22 11:47:...| 10|2019|

| 2019-10-01| AK| HIOS|DentalGuard Prefe...|https://www.guard...| 13| null| null| 42507| 42507| 4250742507|2024-07-22|2024-07-22 11:47:...| 10|2019|

| 2019-10-01| AK| HIOS|Moda Plus AK Regi...|https://www.modah...| 13| null| null| 73836| 73836| 7383673836|2024-07-22|2024-07-22 11:47:...| 10|2019|

| 2019-10-01| AK| HIOS|Moda Plus Providence|https://www.modah...| 14| null| null| 73836| 73836| 7383673836|2024-07-22|2024-07-22 11:47:...| 10|2019|

| 2019-10-01| AK| HIOS| DenteMax|http://www2.dente...| 13| null| null| 74819| 74819| 7481974819|2024-07-22|2024-07-22 11:47:...| 10|2019|

| 2019-10-01| AK| HIOS|Ameritas PPO Dent...| www.standard.com| 13| null| null| 84859| 84859| 8485984859|2024-07-22|2024-07-22 11:47:...| 10|2019|

| 2019-10-01| AL| HIOS| DenteMax|http://www2.dente...| 13| null| null| 12538| 12538| 1253812538|2024-07-22|2024-07-22 11:47:...| 10|2019|

| 2019-10-01| AL| HIOS|CONNECTION Dental...|http://www.ppousa...| 14| null| null| 12538| 12538| 1253812538|2024-07-22|2024-07-22 11:47:...| 10|2019|

| 2019-10-01| AL| HIOS|DentalGuard Prefe...|https://www.guard...| 13| null| null| 17540| 17540| 1754017540|2024-07-22|2024-07-22 11:47:...| 10|2019|

| 2019-10-01| AL| HIOS| Birmingham PPOx|http://pfp.humana...| 13| null| null| 44580| 44580| 4458044580|2024-07-22|2024-07-22 11:47:...| 10|2019|

| 2019-10-01| AL| HIOS| Huntsville PPOx|http://pfp.humana...| 14| null| null| 44580| 44580| 4458044580|2024-07-22|2024-07-22 11:47:...| 10|2019|

| 2019-10-01| AL| HIOS|Humana Dental PPO...|https://www.human...| 15| null| null| 44580| 44580| 4458044580|2024-07-22|2024-07-22 11:47:...| 10|2019|

| 2019-10-01| AL| HIOS| Advantage Plus|https://secure.uc...| 13| null| null| 44785| 44785| 4478544785|2024-07-22|2024-07-22 11:47:...| 10|2019|

| 2019-10-01| AL| HIOS|Lincoln Dental Co...|http://lfg.go2den...| 13| null| null| 46211| 46211| 4621146211|2024-07-22|2024-07-22 11:47:...| 10|2019|

| 2019-10-01| AL| HIOS| BCBSAL PPO|https://www.bcbsa...| 13| null| null| 46944| 46944| 4694446944|2024-07-22|2024-07-22 11:47:...| 10|2019|

| 2019-10-01| AL| HIOS| Choice Plus|http://www.provid...| 13| null| null| 59809| 59809| 5980959809|2024-07-22|2024-07-22 11:47:...| 10|2019|

| 2019-10-01| AL| HIOS| Choice Plus|http://www.provid...| 14| null| null| 59809| 59809| 5980959809|2024-07-22|2024-07-22 11:47:...| 10|2019|

+------------+----+-----+--------------------+--------------------+-------+--------------+--------------------+--------+---------+-----------------+----------+--------------------+-----+----+

only showing top 20 rows

>>> spark.sql("""SELECT \*, month(BusinessDate) AS month, year(BusinessDate) AS year FROM insureview""").show(4)

+------------+----+-----+--------------------+--------------------+-------+--------------+--------------------+--------+---------+-----------------+----------+--------------------+-----+----+

|BusinessDate|stcd|srcnm| NetworkName| NetworkURL|custnum|MarketCoverage| RejectRows|IssuerId|IssuerId2|issueridcomposite| sysdt| systs|month|year|

+------------+----+-----+--------------------+--------------------+-------+--------------+--------------------+--------+---------+-----------------+----------+--------------------+-----+----+

| 2019-10-01| AK| HIOS| ODS Premier|https://www.modah...| 13| null| null| 21989| 21989| 2198921989|2024-07-22|2024-07-22 11:48:...| 10|2019|

| 2019-10-01| AK| HIOS| ODS Premier| null| null| null|219,219,2019-10-0...| 219| 219| 219219|2024-07-22|2024-07-22 11:48:...| 10|2019|

| 2019-10-01| AK| HIOS| HeritagePlus|https://www.preme...| 13| null| null| 38344| 38344| 3834438344|2024-07-22|2024-07-22 11:48:...| 10|2019|

| 2019-10-01| AK| HIOS|Lincoln Dental Co...|http://lfg.go2den...| 13| null| null| 38536| 38536| 3853638536|2024-07-22|2024-07-22 11:48:...| 10|2019|

+------------+----+-----+--------------------+--------------------+-------+--------------+--------------------+--------+---------+-----------------+----------+--------------------+-----+----+

only showing top 4 rows

>>> spark.sql("""select \*,month(BuisnessDate) as month ,year(BuisnessDate) as year from insureview""").show(3)

Traceback (most recent call last):

File "<stdin>", line 1, in <module>

File "/usr/local/spark/python/pyspark/sql/session.py", line 723, in sql

return DataFrame(self.\_jsparkSession.sql(sqlQuery), self.\_wrapped)

File "/usr/local/spark/python/lib/py4j-0.10.9.3-src.zip/py4j/java\_gateway.py", line 1322, in \_\_call\_\_

File "/usr/local/spark/python/pyspark/sql/utils.py", line 117, in deco

raise converted from None

pyspark.sql.utils.AnalysisException: cannot resolve 'BuisnessDate' given input columns: [insureview.BusinessDate, insureview.IssuerId, insureview.IssuerId2, insureview.MarketCoverage, insureview.NetworkName, insureview.NetworkURL, insureview.RejectRows, insureview.custnum, insureview.issueridcomposite, insureview.srcnm, insureview.stcd, insureview.sysdt, insureview.systs]; line 1 pos 15;

'Project [BusinessDate#1378, stcd#7759, srcnm#7747, NetworkName#1381, NetworkURL#1382, custnum#1383, MarketCoverage#1384, RejectRows#1386, IssuerId#8312, IssuerId2#8325, issueridcomposite#8901, sysdt#9003, systs#9016, 'month('BuisnessDate) AS month#28819, 'year('BuisnessDate) AS year#28820]

+- SubqueryAlias insureview

+- View (`insureview`, [BusinessDate#1378,stcd#7759,srcnm#7747,NetworkName#1381,NetworkURL#1382,custnum#1383,MarketCoverage#1384,RejectRows#1386,IssuerId#8312,IssuerId2#8325,issueridcomposite#8901,sysdt#9003,systs#9016])

+- Project [BusinessDate#1378, stcd#7759, srcnm#7747, NetworkName#1381, NetworkURL#1382, custnum#1383, MarketCoverage#1384, RejectRows#1386, IssuerId#8312, IssuerId2#8325, issueridcomposite#8901, sysdt#9003, current\_timestamp() AS systs#9016]

+- Project [BusinessDate#1378, stcd#7759, srcnm#7747, NetworkName#1381, NetworkURL#1382, custnum#1383, MarketCoverage#1384, RejectRows#1386, IssuerId#8312, IssuerId2#8325, issueridcomposite#8901, current\_date(Some(Asia/Kolkata)) AS sysdt#9003]

+- Project [BusinessDate#1378, stcd#7759, srcnm#7747, NetworkName#1381, NetworkURL#1382, custnum#1383, MarketCoverage#1384, RejectRows#1386, IssuerId#8312, IssuerId2#8325, issueridcomposite#8901]

+- Project [BusinessDate#1378, stcd#7759, srcnm#7747, NetworkName#1381, NetworkURL#1382, custnum#1383, MarketCoverage#1384, DentalOnlyPlan#1385, RejectRows#1386, IssuerId#8312, IssuerId2#8325, concat(IssuerId#8312, IssuerId2#8325) AS issueridcomposite#8901]

+- Project [BusinessDate#1378, stcd#7759, srcnm#7747, NetworkName#1381, NetworkURL#1382, custnum#1383, MarketCoverage#1384, DentalOnlyPlan#1385, RejectRows#1386, IssuerId#8312, IssuerId2#8325]

+- Project [IssuerIDDD#8273, BusinessDate#1378, stcd#7759, srcnm#7747, NetworkName#1381, NetworkURL#1382, custnum#1383, MarketCoverage#1384, DentalOnlyPlan#1385, RejectRows#1386, IssuerId#8312, IssuerId2#8325]

+- Project [IssuerIDD#8261, IssuerIDDD#8273, BusinessDate#1378, stcd#7759, srcnm#7747, NetworkName#1381, NetworkURL#1382, custnum#1383, MarketCoverage#1384, DentalOnlyPlan#1385, RejectRows#1386, IssuerId#8312, cast(IssuerIDDD#8273 as string) AS IssuerId2#8325]

+- Project [IssuerIDD#8261, IssuerIDDD#8273, BusinessDate#1378, stcd#7759, srcnm#7747, NetworkName#1381, NetworkURL#1382, custnum#1383, MarketCoverage#1384, DentalOnlyPlan#1385, RejectRows#1386, cast(IssuerIDD#8261 as string) AS IssuerId#8312]

+- Project [IssuerIDD#8261, IssuerId2#1377 AS IssuerIDDD#8273, BusinessDate#1378, stcd#7759, srcnm#7747, NetworkName#1381, NetworkURL#1382, custnum#1383, MarketCoverage#1384, DentalOnlyPlan#1385, RejectRows#1386]

+- Project [IssuerId#1376 AS IssuerIDD#8261, IssuerId2#1377, BusinessDate#1378, stcd#7759, srcnm#7747, NetworkName#1381, NetworkURL#1382, custnum#1383, MarketCoverage#1384, DentalOnlyPlan#1385, RejectRows#1386]

+- Project [IssuerId#1376, IssuerId2#1377, BusinessDate#1378, StateCode#1379 AS stcd#7759, srcnm#7747, NetworkName#1381, NetworkURL#1382, custnum#1383, MarketCoverage#1384, DentalOnlyPlan#1385, RejectRows#1386]

+- Project [IssuerId#1376, IssuerId2#1377, BusinessDate#1378, StateCode#1379, SourceName#1380 AS srcnm#7747, NetworkName#1381, NetworkURL#1382, custnum#1383, MarketCoverage#1384, DentalOnlyPlan#1385, RejectRows#1386]

+- Union false, false

:- Relation [IssuerId#1376,IssuerId2#1377,BusinessDate#1378,StateCode#1379,SourceName#1380,NetworkName#1381,NetworkURL#1382,custnum#1383,MarketCoverage#1384,DentalOnlyPlan#1385,RejectRows#1386] csv

+- Relation [IssuerId#491,IssuerId2#492,BusinessDate#493,StateCode#494,SourceName#495,NetworkName#496,NetworkURL#497,custnum#498,MarketCoverage#499,DentalOnlyPlan#500,RejectRows#501] csv

>>> spark.sql("""SELECT \*, month(BusinessDate) AS month, year(BusinessDate) AS year FROM insureview""").show(4)

+------------+----+-----+--------------------+--------------------+-------+--------------+--------------------+--------+---------+-----------------+----------+--------------------+-----+----+

|BusinessDate|stcd|srcnm| NetworkName| NetworkURL|custnum|MarketCoverage| RejectRows|IssuerId|IssuerId2|issueridcomposite| sysdt| systs|month|year|

+------------+----+-----+--------------------+--------------------+-------+--------------+--------------------+--------+---------+-----------------+----------+--------------------+-----+----+

| 2019-10-01| AK| HIOS| ODS Premier|https://www.modah...| 13| null| null| 21989| 21989| 2198921989|2024-07-22|2024-07-22 11:48:...| 10|2019|

| 2019-10-01| AK| HIOS| ODS Premier| null| null| null|219,219,2019-10-0...| 219| 219| 219219|2024-07-22|2024-07-22 11:48:...| 10|2019|

| 2019-10-01| AK| HIOS| HeritagePlus|https://www.preme...| 13| null| null| 38344| 38344| 3834438344|2024-07-22|2024-07-22 11:48:...| 10|2019|

| 2019-10-01| AK| HIOS|Lincoln Dental Co...|http://lfg.go2den...| 13| null| null| 38536| 38536| 3853638536|2024-07-22|2024-07-22 11:48:...| 10|2019|

+------------+----+-----+--------------------+--------------------+-------+--------------+--------------------+--------+---------+-----------------+----------+--------------------+-----+----+

only showing top 4 rows

>>> spark.sql("""select \*,month(BusinessDate) as month ,year(BusinessDate) as year from insureview""").show(3)

+------------+----+-----+------------+--------------------+-------+--------------+--------------------+--------+---------+-----------------+----------+--------------------+-----+----+

|BusinessDate|stcd|srcnm| NetworkName| NetworkURL|custnum|MarketCoverage| RejectRows|IssuerId|IssuerId2|issueridcomposite| sysdt| systs|month|year|

+------------+----+-----+------------+--------------------+-------+--------------+--------------------+--------+---------+-----------------+----------+--------------------+-----+----+

| 2019-10-01| AK| HIOS| ODS Premier|https://www.modah...| 13| null| null| 21989| 21989| 2198921989|2024-07-22|2024-07-22 11:49:...| 10|2019|

| 2019-10-01| AK| HIOS| ODS Premier| null| null| null|219,219,2019-10-0...| 219| 219| 219219|2024-07-22|2024-07-22 11:49:...| 10|2019|

| 2019-10-01| AK| HIOS|HeritagePlus|https://www.preme...| 13| null| null| 38344| 38344| 3834438344|2024-07-22|2024-07-22 11:49:...| 10|2019|

+------------+----+-----+------------+--------------------+-------+--------------+--------------------+--------+---------+-----------------+----------+--------------------+-----+----+

only showing top 3 rows

>>> spark.sql("""select \*,case when NetworkURL like 'https://' then 'http secured' case when NetworkURL like 'http://' then 'http non secured' else 'No Protocol' as Protocol\_Used from insureview""").show(3)

Traceback (most recent call last):

File "<stdin>", line 1, in <module>

File "/usr/local/spark/python/pyspark/sql/session.py", line 723, in sql

return DataFrame(self.\_jsparkSession.sql(sqlQuery), self.\_wrapped)

File "/usr/local/spark/python/lib/py4j-0.10.9.3-src.zip/py4j/java\_gateway.py", line 1322, in \_\_call\_\_

File "/usr/local/spark/python/pyspark/sql/utils.py", line 117, in deco

raise converted from None

pyspark.sql.utils.ParseException:

mismatched input 'NetworkURL' expecting {<EOF>, ';'}(line 1, pos 19)

== SQL ==

select \*,case when NetworkURL like 'https://' then 'http secured' case when NetworkURL like 'http://' then 'http non secured' else 'No Protocol' as Protocol\_Used from insureview

-------------------^^^

>>> spark.sql("""select \*,case when NetworkURL like 'https://' then 'http secured' when NetworkURL like 'http://' then 'http non secured' else 'No Protocol' as Protocol\_Used from insureview""").show(3)

Traceback (most recent call last):

File "<stdin>", line 1, in <module>

File "/usr/local/spark/python/pyspark/sql/session.py", line 723, in sql

return DataFrame(self.\_jsparkSession.sql(sqlQuery), self.\_wrapped)

File "/usr/local/spark/python/lib/py4j-0.10.9.3-src.zip/py4j/java\_gateway.py", line 1322, in \_\_call\_\_

File "/usr/local/spark/python/pyspark/sql/utils.py", line 117, in deco

raise converted from None

pyspark.sql.utils.ParseException:

mismatched input 'NetworkURL' expecting {<EOF>, ';'}(line 1, pos 19)

== SQL ==

select \*,case when NetworkURL like 'https://' then 'http secured' when NetworkURL like 'http://' then 'http non secured' else 'No Protocol' as Protocol\_Used from insureview

-------------------^^^

>>> spark.sql("""select \*,case when NetworkURL like 'https://%' then 'http secured' when NetworkURL like 'http://%' then 'http non secured' else 'No Protocol' as Protocol\_Used from insureview""").show(3)

Traceback (most recent call last):

File "<stdin>", line 1, in <module>

File "/usr/local/spark/python/pyspark/sql/session.py", line 723, in sql

return DataFrame(self.\_jsparkSession.sql(sqlQuery), self.\_wrapped)

File "/usr/local/spark/python/lib/py4j-0.10.9.3-src.zip/py4j/java\_gateway.py", line 1322, in \_\_call\_\_

File "/usr/local/spark/python/pyspark/sql/utils.py", line 117, in deco

raise converted from None

pyspark.sql.utils.ParseException:

mismatched input 'NetworkURL' expecting {<EOF>, ';'}(line 1, pos 19)

== SQL ==

select \*,case when NetworkURL like 'https://%' then 'http secured' when NetworkURL like 'http://%' then 'http non secured' else 'No Protocol' as Protocol\_Used from insureview

-------------------^^^

>>> spark.sql("""select \*,case when NetworkURL like 'https://%' then 'http secured' when NetworkURL like 'http://%' then 'http non secured' else 'No Protocol' end as Protocol\_Used from insureview""").show(3)

+------------+----+-----+------------+--------------------+-------+--------------+--------------------+--------+---------+-----------------+----------+--------------------+-------------+

|BusinessDate|stcd|srcnm| NetworkName| NetworkURL|custnum|MarketCoverage| RejectRows|IssuerId|IssuerId2|issueridcomposite| sysdt| systs|Protocol\_Used|

+------------+----+-----+------------+--------------------+-------+--------------+--------------------+--------+---------+-----------------+----------+--------------------+-------------+

| 2019-10-01| AK| HIOS| ODS Premier|https://www.modah...| 13| null| null| 21989| 21989| 2198921989|2024-07-22|2024-07-22 11:55:...| http secured|

| 2019-10-01| AK| HIOS| ODS Premier| null| null| null|219,219,2019-10-0...| 219| 219| 219219|2024-07-22|2024-07-22 11:55:...| No Protocol|

| 2019-10-01| AK| HIOS|HeritagePlus|https://www.preme...| 13| null| null| 38344| 38344| 3834438344|2024-07-22|2024-07-22 11:55:...| http secured|

+------------+----+-----+------------+--------------------+-------+--------------+--------------------+--------+---------+-----------------+----------+--------------------+-------------+

only showing top 3 rows

>>> result\_df=spark.sql("""select \*,month(BuisnessDate) as month ,year(BuisnessDate) as year from insureview""")

Traceback (most recent call last):

File "<stdin>", line 1, in <module>

File "/usr/local/spark/python/pyspark/sql/session.py", line 723, in sql

return DataFrame(self.\_jsparkSession.sql(sqlQuery), self.\_wrapped)

File "/usr/local/spark/python/lib/py4j-0.10.9.3-src.zip/py4j/java\_gateway.py", line 1322, in \_\_call\_\_

File "/usr/local/spark/python/pyspark/sql/utils.py", line 117, in deco

raise converted from None

pyspark.sql.utils.AnalysisException: cannot resolve 'BuisnessDate' given input columns: [insureview.BusinessDate, insureview.IssuerId, insureview.IssuerId2, insureview.MarketCoverage, insureview.NetworkName, insureview.NetworkURL, insureview.RejectRows, insureview.custnum, insureview.issueridcomposite, insureview.srcnm, insureview.stcd, insureview.sysdt, insureview.systs]; line 1 pos 15;

'Project [BusinessDate#1378, stcd#7759, srcnm#7747, NetworkName#1381, NetworkURL#1382, custnum#1383, MarketCoverage#1384, RejectRows#1386, IssuerId#8312, IssuerId2#8325, issueridcomposite#8901, sysdt#9003, systs#9016, 'month('BuisnessDate) AS month#30398, 'year('BuisnessDate) AS year#30399]

+- SubqueryAlias insureview

+- View (`insureview`, [BusinessDate#1378,stcd#7759,srcnm#7747,NetworkName#1381,NetworkURL#1382,custnum#1383,MarketCoverage#1384,RejectRows#1386,IssuerId#8312,IssuerId2#8325,issueridcomposite#8901,sysdt#9003,systs#9016])

+- Project [BusinessDate#1378, stcd#7759, srcnm#7747, NetworkName#1381, NetworkURL#1382, custnum#1383, MarketCoverage#1384, RejectRows#1386, IssuerId#8312, IssuerId2#8325, issueridcomposite#8901, sysdt#9003, current\_timestamp() AS systs#9016]

+- Project [BusinessDate#1378, stcd#7759, srcnm#7747, NetworkName#1381, NetworkURL#1382, custnum#1383, MarketCoverage#1384, RejectRows#1386, IssuerId#8312, IssuerId2#8325, issueridcomposite#8901, current\_date(Some(Asia/Kolkata)) AS sysdt#9003]

+- Project [BusinessDate#1378, stcd#7759, srcnm#7747, NetworkName#1381, NetworkURL#1382, custnum#1383, MarketCoverage#1384, RejectRows#1386, IssuerId#8312, IssuerId2#8325, issueridcomposite#8901]

+- Project [BusinessDate#1378, stcd#7759, srcnm#7747, NetworkName#1381, NetworkURL#1382, custnum#1383, MarketCoverage#1384, DentalOnlyPlan#1385, RejectRows#1386, IssuerId#8312, IssuerId2#8325, concat(IssuerId#8312, IssuerId2#8325) AS issueridcomposite#8901]

+- Project [BusinessDate#1378, stcd#7759, srcnm#7747, NetworkName#1381, NetworkURL#1382, custnum#1383, MarketCoverage#1384, DentalOnlyPlan#1385, RejectRows#1386, IssuerId#8312, IssuerId2#8325]

+- Project [IssuerIDDD#8273, BusinessDate#1378, stcd#7759, srcnm#7747, NetworkName#1381, NetworkURL#1382, custnum#1383, MarketCoverage#1384, DentalOnlyPlan#1385, RejectRows#1386, IssuerId#8312, IssuerId2#8325]

+- Project [IssuerIDD#8261, IssuerIDDD#8273, BusinessDate#1378, stcd#7759, srcnm#7747, NetworkName#1381, NetworkURL#1382, custnum#1383, MarketCoverage#1384, DentalOnlyPlan#1385, RejectRows#1386, IssuerId#8312, cast(IssuerIDDD#8273 as string) AS IssuerId2#8325]

+- Project [IssuerIDD#8261, IssuerIDDD#8273, BusinessDate#1378, stcd#7759, srcnm#7747, NetworkName#1381, NetworkURL#1382, custnum#1383, MarketCoverage#1384, DentalOnlyPlan#1385, RejectRows#1386, cast(IssuerIDD#8261 as string) AS IssuerId#8312]

+- Project [IssuerIDD#8261, IssuerId2#1377 AS IssuerIDDD#8273, BusinessDate#1378, stcd#7759, srcnm#7747, NetworkName#1381, NetworkURL#1382, custnum#1383, MarketCoverage#1384, DentalOnlyPlan#1385, RejectRows#1386]

+- Project [IssuerId#1376 AS IssuerIDD#8261, IssuerId2#1377, BusinessDate#1378, stcd#7759, srcnm#7747, NetworkName#1381, NetworkURL#1382, custnum#1383, MarketCoverage#1384, DentalOnlyPlan#1385, RejectRows#1386]

+- Project [IssuerId#1376, IssuerId2#1377, BusinessDate#1378, StateCode#1379 AS stcd#7759, srcnm#7747, NetworkName#1381, NetworkURL#1382, custnum#1383, MarketCoverage#1384, DentalOnlyPlan#1385, RejectRows#1386]

+- Project [IssuerId#1376, IssuerId2#1377, BusinessDate#1378, StateCode#1379, SourceName#1380 AS srcnm#7747, NetworkName#1381, NetworkURL#1382, custnum#1383, MarketCoverage#1384, DentalOnlyPlan#1385, RejectRows#1386]

+- Union false, false

:- Relation [IssuerId#1376,IssuerId2#1377,BusinessDate#1378,StateCode#1379,SourceName#1380,NetworkName#1381,NetworkURL#1382,custnum#1383,MarketCoverage#1384,DentalOnlyPlan#1385,RejectRows#1386] csv

+- Relation [IssuerId#491,IssuerId2#492,BusinessDate#493,StateCode#494,SourceName#495,NetworkName#496,NetworkURL#497,custnum#498,MarketCoverage#499,DentalOnlyPlan#500,RejectRows#501] csv

>>> spark.sql("""select \*,month(BusinessDate) as month ,year(BusinessDate) as year from insureview""").show(3)

+------------+----+-----+------------+--------------------+-------+--------------+--------------------+--------+---------+-----------------+----------+--------------------+-----+----+

|BusinessDate|stcd|srcnm| NetworkName| NetworkURL|custnum|MarketCoverage| RejectRows|IssuerId|IssuerId2|issueridcomposite| sysdt| systs|month|year|

+------------+----+-----+------------+--------------------+-------+--------------+--------------------+--------+---------+-----------------+----------+--------------------+-----+----+

| 2019-10-01| AK| HIOS| ODS Premier|https://www.modah...| 13| null| null| 21989| 21989| 2198921989|2024-07-22|2024-07-22 11:57:...| 10|2019|

| 2019-10-01| AK| HIOS| ODS Premier| null| null| null|219,219,2019-10-0...| 219| 219| 219219|2024-07-22|2024-07-22 11:57:...| 10|2019|

| 2019-10-01| AK| HIOS|HeritagePlus|https://www.preme...| 13| null| null| 38344| 38344| 3834438344|2024-07-22|2024-07-22 11:57:...| 10|2019|

+------------+----+-----+------------+--------------------+-------+--------------+--------------------+--------+---------+-----------------+----------+--------------------+-----+----+

only showing top 3 rows

>>> result\_df\_SQL=spark.sql("""select \*,month(BusinessDate) as month ,year(BusinessDate) as year from insureview""")

>>> result\_df\_SQL\_1=spark.sql("""select \*,case when NetworkURL like 'https://%' then 'http secured' when NetworkURL like 'http://%' then 'http non secured' else 'No Protocol' as Protocol\_Used from insureview""")

Traceback (most recent call last):

File "<stdin>", line 1, in <module>

File "/usr/local/spark/python/pyspark/sql/session.py", line 723, in sql

return DataFrame(self.\_jsparkSession.sql(sqlQuery), self.\_wrapped)

File "/usr/local/spark/python/lib/py4j-0.10.9.3-src.zip/py4j/java\_gateway.py", line 1322, in \_\_call\_\_

File "/usr/local/spark/python/pyspark/sql/utils.py", line 117, in deco

raise converted from None

pyspark.sql.utils.ParseException:

mismatched input 'NetworkURL' expecting {<EOF>, ';'}(line 1, pos 19)

== SQL ==

select \*,case when NetworkURL like 'https://%' then 'http secured' when NetworkURL like 'http://%' then 'http non secured' else 'No Protocol' as Protocol\_Used from insureview

-------------------^^^

>>> result\_df\_SQL\_1=spark.sql("""select \*,case when NetworkURL like 'https://%' then 'http secured' when NetworkURL like 'http://%' then 'http non secured' else 'No Protocol' end as Protocol\_Used from insureview""")

>>> spark.sql("select \*,current\_date as curdt , current\_timestamp as curts from insureview").show(3)

+------------+----+-----+------------+--------------------+-------+--------------+--------------------+--------+---------+-----------------+----------+--------------------+----------+--------------------+

|BusinessDate|stcd|srcnm| NetworkName| NetworkURL|custnum|MarketCoverage| RejectRows|IssuerId|IssuerId2|issueridcomposite| sysdt| systs| curdt| curts|

+------------+----+-----+------------+--------------------+-------+--------------+--------------------+--------+---------+-----------------+----------+--------------------+----------+--------------------+

| 2019-10-01| AK| HIOS| ODS Premier|https://www.modah...| 13| null| null| 21989| 21989| 2198921989|2024-07-22|2024-07-22 12:00:...|2024-07-22|2024-07-22 12:00:...|

| 2019-10-01| AK| HIOS| ODS Premier| null| null| null|219,219,2019-10-0...| 219| 219| 219219|2024-07-22|2024-07-22 12:00:...|2024-07-22|2024-07-22 12:00:...|

| 2019-10-01| AK| HIOS|HeritagePlus|https://www.preme...| 13| null| null| 38344| 38344| 3834438344|2024-07-22|2024-07-22 12:00:...|2024-07-22|2024-07-22 12:00:...|

+------------+----+-----+------------+--------------------+-------+--------------+--------------------+--------+---------+-----------------+----------+--------------------+----------+--------------------+

only showing top 3 rows

>>> result\_df\_SQL\_2=spark.sql("select \*,current\_date as curdt , current\_timestamp as curts from insureview")

>>> spark.sql("select \*,sql\_function(NetworkName) as cleannetworkname from insureview").show(3)

+------------+----+-----+------------+--------------------+-------+--------------+--------------------+--------+---------+-----------------+----------+--------------------+----------------+

|BusinessDate|stcd|srcnm| NetworkName| NetworkURL|custnum|MarketCoverage| RejectRows|IssuerId|IssuerId2|issueridcomposite| sysdt| systs|cleannetworkname|

+------------+----+-----+------------+--------------------+-------+--------------+--------------------+--------+---------+-----------------+----------+--------------------+----------------+

| 2019-10-01| AK| HIOS| ODS Premier|https://www.modah...| 13| null| null| 21989| 21989| 2198921989|2024-07-22|2024-07-22 12:00:...| ODS Premier|

| 2019-10-01| AK| HIOS| ODS Premier| null| null| null|219,219,2019-10-0...| 219| 219| 219219|2024-07-22|2024-07-22 12:00:...| ODS Premier|

| 2019-10-01| AK| HIOS|HeritagePlus|https://www.preme...| 13| null| null| 38344| 38344| 3834438344|2024-07-22|2024-07-22 12:00:...| HeritagePlus|

+------------+----+-----+------------+--------------------+-------+--------------+--------------------+--------+---------+-----------------+----------+--------------------+----------------+

only showing top 3 rows

>>> result\_df\_SQL\_3=spark.sql("select \*,sql\_function(NetworkName) as cleannetworkname from insureview")

>>> result\_df\_SQL\_after\_join=result\_df\_SQL.alias("l").join(result\_df\_SQL\_1.alias("r") , on=[l.IssuerId=r.IssuerId],how="inner")

File "<stdin>", line 1

result\_df\_SQL\_after\_join=result\_df\_SQL.alias("l").join(result\_df\_SQL\_1.alias("r") , on=[l.IssuerId=r.IssuerId],how="inner")

^

SyntaxError: invalid syntax

>>> result\_df\_SQL\_after\_join=result\_df\_SQL.alias("l").join(result\_df\_SQL\_1.alias("r") , on=[l.IssuerId=r.IssuerId],how="inner")

File "<stdin>", line 1

result\_df\_SQL\_after\_join=result\_df\_SQL.alias("l").join(result\_df\_SQL\_1.alias("r") , on=[l.IssuerId=r.IssuerId],how="inner")

^

SyntaxError: invalid syntax

>>> result\_df\_SQL\_after\_join = result\_df\_SQL.alias("l").join(result\_df\_SQL\_1.alias("r"), on=col("l.IssuerId") == col("r.IssuerId"), how="inner")

>>>

>>> result\_df\_SQL\_after\_join = result\_df\_SQL\_after\_join.alias("l").join(result\_df\_SQL\_2.alias("r"), on=col("l.IssuerId") == col("r.IssuerId"), how="inner")

Traceback (most recent call last):

File "<stdin>", line 1, in <module>

File "/usr/local/spark/python/pyspark/sql/dataframe.py", line 1355, in join

jdf = self.\_jdf.join(other.\_jdf, on, how)

File "/usr/local/spark/python/lib/py4j-0.10.9.3-src.zip/py4j/java\_gateway.py", line 1322, in \_\_call\_\_

File "/usr/local/spark/python/pyspark/sql/utils.py", line 117, in deco

raise converted from None

pyspark.sql.utils.AnalysisException: Reference 'l.IssuerId' is ambiguous, could be: l.IssuerId, l.IssuerId.

>>> result\_df\_SQL\_after\_join = result\_df\_SQL\_after\_join.alias("l").join(result\_df\_SQL\_2.alias("r"), on=[col("l.IssuerId") == col("r.IssuerId")], how="inner")

Traceback (most recent call last):

File "<stdin>", line 1, in <module>

File "/usr/local/spark/python/pyspark/sql/dataframe.py", line 1355, in join

jdf = self.\_jdf.join(other.\_jdf, on, how)

File "/usr/local/spark/python/lib/py4j-0.10.9.3-src.zip/py4j/java\_gateway.py", line 1322, in \_\_call\_\_

File "/usr/local/spark/python/pyspark/sql/utils.py", line 117, in deco

raise converted from None

pyspark.sql.utils.AnalysisException: Reference 'l.IssuerId' is ambiguous, could be: l.IssuerId, l.IssuerId.

>>> result\_df\_SQL\_after\_join=result\_df\_SQL.alias("l").join(result\_df\_SQL\_1.alias("r") , on=[l.IssuerId=r.IssuerId],how="inner")

>>> result\_df\_SQL\_after\_join=result\_df\_SQL.alias("l").join(result\_df\_SQL\_1.alias("r") , on=[l.IssuerId=r.IssuerId],how="inner")

File "<stdin>", line 1

result\_df\_SQL\_after\_join=result\_df\_SQL.alias("l").join(result\_df\_SQL\_1.alias("r") , on=[l.IssuerId=r.IssuerId],how="inner")

^

SyntaxError: invalid syntax

>>> result\_df\_SQL\_after\_join = result\_df\_SQL\_after\_join.alias("l").join(result\_df\_SQL\_2.alias("r"), on=[col("l.IssuerId") == col("r.IssuerId")], how="inner")

Traceback (most recent call last):

File "<stdin>", line 1, in <module>

File "/usr/local/spark/python/pyspark/sql/dataframe.py", line 1355, in join

jdf = self.\_jdf.join(other.\_jdf, on, how)

File "/usr/local/spark/python/lib/py4j-0.10.9.3-src.zip/py4j/java\_gateway.py", line 1322, in \_\_call\_\_

File "/usr/local/spark/python/pyspark/sql/utils.py", line 117, in deco

raise converted from None

pyspark.sql.utils.AnalysisException: Reference 'l.IssuerId' is ambiguous, could be: l.IssuerId, l.IssuerId.

>>> result\_df\_SQL\_after\_join = result\_df\_SQL.alias("l").join(result\_df\_SQL\_1.alias("r"), on=col("l.IssuerId") == col("r.IssuerId"), how="inner")

>>>

Final :

>>> print(custfilter.take()3)

File "<stdin>", line 1

print(custfilter.take()3)

^

SyntaxError: invalid syntax

>>> print(custfilter.take(4))

[True, True, True, True]

>>> print(custfilter.take(14))

[True, True, True, True, True, True, True, True, True, True, True, True, True, True]

>>> custfilter =rdd.map(lambda x: (len(x)>2).collect )

>>> print(custfilter.take(14))

24/07/22 11:12:52 ERROR executor.Executor: Exception in task 0.0 in stage 118.0 (TID 137)

org.apache.spark.api.python.PythonException: Traceback (most recent call last):

File "/usr/local/spark/python/lib/pyspark.zip/pyspark/worker.py", line 619, in main

process()

File "/usr/local/spark/python/lib/pyspark.zip/pyspark/worker.py", line 611, in process

serializer.dump\_stream(out\_iter, outfile)

File "/usr/local/spark/python/lib/pyspark.zip/pyspark/serializers.py", line 259, in dump\_stream

vs = list(itertools.islice(iterator, batch))

File "/usr/local/spark/python/pyspark/rdd.py", line 1562, in takeUpToNumLeft

yield next(iterator)

File "/usr/local/spark/python/lib/pyspark.zip/pyspark/util.py", line 74, in wrapper

return f(\*args, \*\*kwargs)

File "<stdin>", line 1, in <lambda>

AttributeError: 'bool' object has no attribute 'collect'

at org.apache.spark.api.python.BasePythonRunner$ReaderIterator.handlePythonException(PythonRunner.scala:555)

at org.apache.spark.api.python.PythonRunner$$anon$3.read(PythonRunner.scala:713)

at org.apache.spark.api.python.PythonRunner$$anon$3.read(PythonRunner.scala:695)

at org.apache.spark.api.python.BasePythonRunner$ReaderIterator.hasNext(PythonRunner.scala:508)

at org.apache.spark.InterruptibleIterator.hasNext(InterruptibleIterator.scala:37)

at scala.collection.Iterator.foreach(Iterator.scala:943)

at scala.collection.Iterator.foreach$(Iterator.scala:943)

at org.apache.spark.InterruptibleIterator.foreach(InterruptibleIterator.scala:28)

at scala.collection.generic.Growable.$plus$plus$eq(Growable.scala:62)

at scala.collection.generic.Growable.$plus$plus$eq$(Growable.scala:53)

at scala.collection.mutable.ArrayBuffer.$plus$plus$eq(ArrayBuffer.scala:105)

at scala.collection.mutable.ArrayBuffer.$plus$plus$eq(ArrayBuffer.scala:49)

at scala.collection.TraversableOnce.to(TraversableOnce.scala:366)

at scala.collection.TraversableOnce.to$(TraversableOnce.scala:364)

at org.apache.spark.InterruptibleIterator.to(InterruptibleIterator.scala:28)

at scala.collection.TraversableOnce.toBuffer(TraversableOnce.scala:358)

at scala.collection.TraversableOnce.toBuffer$(TraversableOnce.scala:358)

at org.apache.spark.InterruptibleIterator.toBuffer(InterruptibleIterator.scala:28)

at scala.collection.TraversableOnce.toArray(TraversableOnce.scala:345)

at scala.collection.TraversableOnce.toArray$(TraversableOnce.scala:339)

at org.apache.spark.InterruptibleIterator.toArray(InterruptibleIterator.scala:28)

at org.apache.spark.api.python.PythonRDD$.$anonfun$runJob$1(PythonRDD.scala:166)

at org.apache.spark.SparkContext.$anonfun$runJob$5(SparkContext.scala:2254)

at org.apache.spark.scheduler.ResultTask.runTask(ResultTask.scala:90)

at org.apache.spark.scheduler.Task.run(Task.scala:131)

at org.apache.spark.executor.Executor$TaskRunner.$anonfun$run$3(Executor.scala:506)

at org.apache.spark.util.Utils$.tryWithSafeFinally(Utils.scala:1462)

at org.apache.spark.executor.Executor$TaskRunner.run(Executor.scala:509)

at java.util.concurrent.ThreadPoolExecutor.runWorker(ThreadPoolExecutor.java:1149)

at java.util.concurrent.ThreadPoolExecutor$Worker.run(ThreadPoolExecutor.java:624)

at java.lang.Thread.run(Thread.java:750)

24/07/22 11:12:52 WARN scheduler.TaskSetManager: Lost task 0.0 in stage 118.0 (TID 137) (192.168.112.129 executor driver): org.apache.spark.api.python.PythonException: Traceback (most recent call last):

File "/usr/local/spark/python/lib/pyspark.zip/pyspark/worker.py", line 619, in main

process()

File "/usr/local/spark/python/lib/pyspark.zip/pyspark/worker.py", line 611, in process

serializer.dump\_stream(out\_iter, outfile)

File "/usr/local/spark/python/lib/pyspark.zip/pyspark/serializers.py", line 259, in dump\_stream

vs = list(itertools.islice(iterator, batch))

File "/usr/local/spark/python/pyspark/rdd.py", line 1562, in takeUpToNumLeft

yield next(iterator)

File "/usr/local/spark/python/lib/pyspark.zip/pyspark/util.py", line 74, in wrapper

return f(\*args, \*\*kwargs)

File "<stdin>", line 1, in <lambda>

AttributeError: 'bool' object has no attribute 'collect'

at org.apache.spark.api.python.BasePythonRunner$ReaderIterator.handlePythonException(PythonRunner.scala:555)

at org.apache.spark.api.python.PythonRunner$$anon$3.read(PythonRunner.scala:713)

at org.apache.spark.api.python.PythonRunner$$anon$3.read(PythonRunner.scala:695)

at org.apache.spark.api.python.BasePythonRunner$ReaderIterator.hasNext(PythonRunner.scala:508)

at org.apache.spark.InterruptibleIterator.hasNext(InterruptibleIterator.scala:37)

at scala.collection.Iterator.foreach(Iterator.scala:943)

at scala.collection.Iterator.foreach$(Iterator.scala:943)

at org.apache.spark.InterruptibleIterator.foreach(InterruptibleIterator.scala:28)

at scala.collection.generic.Growable.$plus$plus$eq(Growable.scala:62)

at scala.collection.generic.Growable.$plus$plus$eq$(Growable.scala:53)

at scala.collection.mutable.ArrayBuffer.$plus$plus$eq(ArrayBuffer.scala:105)

at scala.collection.mutable.ArrayBuffer.$plus$plus$eq(ArrayBuffer.scala:49)

at scala.collection.TraversableOnce.to(TraversableOnce.scala:366)

at scala.collection.TraversableOnce.to$(TraversableOnce.scala:364)

at org.apache.spark.InterruptibleIterator.to(InterruptibleIterator.scala:28)

at scala.collection.TraversableOnce.toBuffer(TraversableOnce.scala:358)

at scala.collection.TraversableOnce.toBuffer$(TraversableOnce.scala:358)

at org.apache.spark.InterruptibleIterator.toBuffer(InterruptibleIterator.scala:28)

at scala.collection.TraversableOnce.toArray(TraversableOnce.scala:345)

at scala.collection.TraversableOnce.toArray$(TraversableOnce.scala:339)

at org.apache.spark.InterruptibleIterator.toArray(InterruptibleIterator.scala:28)

at org.apache.spark.api.python.PythonRDD$.$anonfun$runJob$1(PythonRDD.scala:166)

at org.apache.spark.SparkContext.$anonfun$runJob$5(SparkContext.scala:2254)

at org.apache.spark.scheduler.ResultTask.runTask(ResultTask.scala:90)

at org.apache.spark.scheduler.Task.run(Task.scala:131)

at org.apache.spark.executor.Executor$TaskRunner.$anonfun$run$3(Executor.scala:506)

at org.apache.spark.util.Utils$.tryWithSafeFinally(Utils.scala:1462)

at org.apache.spark.executor.Executor$TaskRunner.run(Executor.scala:509)

at java.util.concurrent.ThreadPoolExecutor.runWorker(ThreadPoolExecutor.java:1149)

at java.util.concurrent.ThreadPoolExecutor$Worker.run(ThreadPoolExecutor.java:624)

at java.lang.Thread.run(Thread.java:750)

24/07/22 11:12:52 ERROR scheduler.TaskSetManager: Task 0 in stage 118.0 failed 1 times; aborting job

Traceback (most recent call last):

File "<stdin>", line 1, in <module>

File "/usr/local/spark/python/pyspark/rdd.py", line 1568, in take

res = self.context.runJob(self, takeUpToNumLeft, p)

File "/usr/local/spark/python/pyspark/context.py", line 1227, in runJob

sock\_info = self.\_jvm.PythonRDD.runJob(self.\_jsc.sc(), mappedRDD.\_jrdd, partitions)

File "/usr/local/spark/python/lib/py4j-0.10.9.3-src.zip/py4j/java\_gateway.py", line 1322, in \_\_call\_\_

File "/usr/local/spark/python/pyspark/sql/utils.py", line 111, in deco

return f(\*a, \*\*kw)

File "/usr/local/spark/python/lib/py4j-0.10.9.3-src.zip/py4j/protocol.py", line 328, in get\_return\_value

py4j.protocol.Py4JJavaError: An error occurred while calling z:org.apache.spark.api.python.PythonRDD.runJob.

: org.apache.spark.SparkException: Job aborted due to stage failure: Task 0 in stage 118.0 failed 1 times, most recent failure: Lost task 0.0 in stage 118.0 (TID 137) (192.168.112.129 executor driver): org.apache.spark.api.python.PythonException: Traceback (most recent call last):

File "/usr/local/spark/python/lib/pyspark.zip/pyspark/worker.py", line 619, in main

process()

File "/usr/local/spark/python/lib/pyspark.zip/pyspark/worker.py", line 611, in process

serializer.dump\_stream(out\_iter, outfile)

File "/usr/local/spark/python/lib/pyspark.zip/pyspark/serializers.py", line 259, in dump\_stream

vs = list(itertools.islice(iterator, batch))

File "/usr/local/spark/python/pyspark/rdd.py", line 1562, in takeUpToNumLeft

yield next(iterator)

File "/usr/local/spark/python/lib/pyspark.zip/pyspark/util.py", line 74, in wrapper

return f(\*args, \*\*kwargs)

File "<stdin>", line 1, in <lambda>

AttributeError: 'bool' object has no attribute 'collect'

at org.apache.spark.api.python.BasePythonRunner$ReaderIterator.handlePythonException(PythonRunner.scala:555)

at org.apache.spark.api.python.PythonRunner$$anon$3.read(PythonRunner.scala:713)

at org.apache.spark.api.python.PythonRunner$$anon$3.read(PythonRunner.scala:695)

at org.apache.spark.api.python.BasePythonRunner$ReaderIterator.hasNext(PythonRunner.scala:508)

at org.apache.spark.InterruptibleIterator.hasNext(InterruptibleIterator.scala:37)

at scala.collection.Iterator.foreach(Iterator.scala:943)

at scala.collection.Iterator.foreach$(Iterator.scala:943)

at org.apache.spark.InterruptibleIterator.foreach(InterruptibleIterator.scala:28)

at scala.collection.generic.Growable.$plus$plus$eq(Growable.scala:62)

at scala.collection.generic.Growable.$plus$plus$eq$(Growable.scala:53)

at scala.collection.mutable.ArrayBuffer.$plus$plus$eq(ArrayBuffer.scala:105)

at scala.collection.mutable.ArrayBuffer.$plus$plus$eq(ArrayBuffer.scala:49)

at scala.collection.TraversableOnce.to(TraversableOnce.scala:366)

at scala.collection.TraversableOnce.to$(TraversableOnce.scala:364)

at org.apache.spark.InterruptibleIterator.to(InterruptibleIterator.scala:28)

at scala.collection.TraversableOnce.toBuffer(TraversableOnce.scala:358)

at scala.collection.TraversableOnce.toBuffer$(TraversableOnce.scala:358)

at org.apache.spark.InterruptibleIterator.toBuffer(InterruptibleIterator.scala:28)

at scala.collection.TraversableOnce.toArray(TraversableOnce.scala:345)

at scala.collection.TraversableOnce.toArray$(TraversableOnce.scala:339)

at org.apache.spark.InterruptibleIterator.toArray(InterruptibleIterator.scala:28)

at org.apache.spark.api.python.PythonRDD$.$anonfun$runJob$1(PythonRDD.scala:166)

at org.apache.spark.SparkContext.$anonfun$runJob$5(SparkContext.scala:2254)

at org.apache.spark.scheduler.ResultTask.runTask(ResultTask.scala:90)

at org.apache.spark.scheduler.Task.run(Task.scala:131)

at org.apache.spark.executor.Executor$TaskRunner.$anonfun$run$3(Executor.scala:506)

at org.apache.spark.util.Utils$.tryWithSafeFinally(Utils.scala:1462)

at org.apache.spark.executor.Executor$TaskRunner.run(Executor.scala:509)

at java.util.concurrent.ThreadPoolExecutor.runWorker(ThreadPoolExecutor.java:1149)

at java.util.concurrent.ThreadPoolExecutor$Worker.run(ThreadPoolExecutor.java:624)

at java.lang.Thread.run(Thread.java:750)

Driver stacktrace:

at org.apache.spark.scheduler.DAGScheduler.failJobAndIndependentStages(DAGScheduler.scala:2454)

at org.apache.spark.scheduler.DAGScheduler.$anonfun$abortStage$2(DAGScheduler.scala:2403)

at org.apache.spark.scheduler.DAGScheduler.$anonfun$abortStage$2$adapted(DAGScheduler.scala:2402)

at scala.collection.mutable.ResizableArray.foreach(ResizableArray.scala:62)

at scala.collection.mutable.ResizableArray.foreach$(ResizableArray.scala:55)

at scala.collection.mutable.ArrayBuffer.foreach(ArrayBuffer.scala:49)

at org.apache.spark.scheduler.DAGScheduler.abortStage(DAGScheduler.scala:2402)

at org.apache.spark.scheduler.DAGScheduler.$anonfun$handleTaskSetFailed$1(DAGScheduler.scala:1160)

at org.apache.spark.scheduler.DAGScheduler.$anonfun$handleTaskSetFailed$1$adapted(DAGScheduler.scala:1160)

at scala.Option.foreach(Option.scala:407)

at org.apache.spark.scheduler.DAGScheduler.handleTaskSetFailed(DAGScheduler.scala:1160)

at org.apache.spark.scheduler.DAGSchedulerEventProcessLoop.doOnReceive(DAGScheduler.scala:2642)

at org.apache.spark.scheduler.DAGSchedulerEventProcessLoop.onReceive(DAGScheduler.scala:2584)

at org.apache.spark.scheduler.DAGSchedulerEventProcessLoop.onReceive(DAGScheduler.scala:2573)

at org.apache.spark.util.EventLoop$$anon$1.run(EventLoop.scala:49)

at org.apache.spark.scheduler.DAGScheduler.runJob(DAGScheduler.scala:938)

at org.apache.spark.SparkContext.runJob(SparkContext.scala:2214)

at org.apache.spark.SparkContext.runJob(SparkContext.scala:2235)

at org.apache.spark.SparkContext.runJob(SparkContext.scala:2254)

at org.apache.spark.api.python.PythonRDD$.runJob(PythonRDD.scala:166)

at org.apache.spark.api.python.PythonRDD.runJob(PythonRDD.scala)

at sun.reflect.NativeMethodAccessorImpl.invoke0(Native Method)

at sun.reflect.NativeMethodAccessorImpl.invoke(NativeMethodAccessorImpl.java:62)

at sun.reflect.DelegatingMethodAccessorImpl.invoke(DelegatingMethodAccessorImpl.java:43)

at java.lang.reflect.Method.invoke(Method.java:498)

at py4j.reflection.MethodInvoker.invoke(MethodInvoker.java:244)

at py4j.reflection.ReflectionEngine.invoke(ReflectionEngine.java:357)

at py4j.Gateway.invoke(Gateway.java:282)

at py4j.commands.AbstractCommand.invokeMethod(AbstractCommand.java:132)

at py4j.commands.CallCommand.execute(CallCommand.java:79)

at py4j.ClientServerConnection.waitForCommands(ClientServerConnection.java:182)

at py4j.ClientServerConnection.run(ClientServerConnection.java:106)

at java.lang.Thread.run(Thread.java:750)

Caused by: org.apache.spark.api.python.PythonException: Traceback (most recent call last):

File "/usr/local/spark/python/lib/pyspark.zip/pyspark/worker.py", line 619, in main

process()

File "/usr/local/spark/python/lib/pyspark.zip/pyspark/worker.py", line 611, in process

serializer.dump\_stream(out\_iter, outfile)

File "/usr/local/spark/python/lib/pyspark.zip/pyspark/serializers.py", line 259, in dump\_stream

vs = list(itertools.islice(iterator, batch))

File "/usr/local/spark/python/pyspark/rdd.py", line 1562, in takeUpToNumLeft

yield next(iterator)

File "/usr/local/spark/python/lib/pyspark.zip/pyspark/util.py", line 74, in wrapper

return f(\*args, \*\*kwargs)

File "<stdin>", line 1, in <lambda>

AttributeError: 'bool' object has no attribute 'collect'

at org.apache.spark.api.python.BasePythonRunner$ReaderIterator.handlePythonException(PythonRunner.scala:555)

at org.apache.spark.api.python.PythonRunner$$anon$3.read(PythonRunner.scala:713)

at org.apache.spark.api.python.PythonRunner$$anon$3.read(PythonRunner.scala:695)

at org.apache.spark.api.python.BasePythonRunner$ReaderIterator.hasNext(PythonRunner.scala:508)

at org.apache.spark.InterruptibleIterator.hasNext(InterruptibleIterator.scala:37)

at scala.collection.Iterator.foreach(Iterator.scala:943)

at scala.collection.Iterator.foreach$(Iterator.scala:943)

at org.apache.spark.InterruptibleIterator.foreach(InterruptibleIterator.scala:28)

at scala.collection.generic.Growable.$plus$plus$eq(Growable.scala:62)

at scala.collection.generic.Growable.$plus$plus$eq$(Growable.scala:53)

at scala.collection.mutable.ArrayBuffer.$plus$plus$eq(ArrayBuffer.scala:105)

at scala.collection.mutable.ArrayBuffer.$plus$plus$eq(ArrayBuffer.scala:49)

at scala.collection.TraversableOnce.to(TraversableOnce.scala:366)

at scala.collection.TraversableOnce.to$(TraversableOnce.scala:364)

at org.apache.spark.InterruptibleIterator.to(InterruptibleIterator.scala:28)

at scala.collection.TraversableOnce.toBuffer(TraversableOnce.scala:358)

at scala.collection.TraversableOnce.toBuffer$(TraversableOnce.scala:358)

at org.apache.spark.InterruptibleIterator.toBuffer(InterruptibleIterator.scala:28)

at scala.collection.TraversableOnce.toArray(TraversableOnce.scala:345)

at scala.collection.TraversableOnce.toArray$(TraversableOnce.scala:339)

at org.apache.spark.InterruptibleIterator.toArray(InterruptibleIterator.scala:28)

at org.apache.spark.api.python.PythonRDD$.$anonfun$runJob$1(PythonRDD.scala:166)

at org.apache.spark.SparkContext.$anonfun$runJob$5(SparkContext.scala:2254)

at org.apache.spark.scheduler.ResultTask.runTask(ResultTask.scala:90)

at org.apache.spark.scheduler.Task.run(Task.scala:131)

at org.apache.spark.executor.Executor$TaskRunner.$anonfun$run$3(Executor.scala:506)

at org.apache.spark.util.Utils$.tryWithSafeFinally(Utils.scala:1462)

at org.apache.spark.executor.Executor$TaskRunner.run(Executor.scala:509)

at java.util.concurrent.ThreadPoolExecutor.runWorker(ThreadPoolExecutor.java:1149)

at java.util.concurrent.ThreadPoolExecutor$Worker.run(ThreadPoolExecutor.java:624)

... 1 more

>>> rdd.take(4)

['1,Kristina,Chung,55,Pilot', '2,Paige,Chen,74,Teacher', 'NM,New Mexico', 'NY,New York']

>>> custfilter =rdd.map(lambda x: x.split(",") )

>>> print(custfilter.take(14))

[['1', 'Kristina', 'Chung', '55', 'Pilot'], ['2', 'Paige', 'Chen', '74', 'Teacher'], ['NM', 'New Mexico'], ['NY', 'New York'], ['NC', 'North Carolina'], ['ND', 'North Dakota'], ['OH', 'Ohio'], ['OK', 'Oklahoma'], ['OR', 'Oregon'], ['MD', 'Maryland'], ['MA', 'Massachusetts'], ['MI', 'Michigan'], ['MN', 'Minnesota'], ['MS', 'Mississippi']]

>>> splited\_rdd =rdd.map(lambda x: x.split(",") )

>>> custfilter =splited\_rdd.filter(lambda x: len(x)>3 )

>>> print(custfilter.take(4))

[['1', 'Kristina', 'Chung', '55', 'Pilot'], ['2', 'Paige', 'Chen', '74', 'Teacher'], ['3', 'Sherri', 'Melton', '34', 'Firefighter'], ['4', 'Gretchen', 'Hill', '66', 'Computer hardware engineer']]

>>> print(custfilter.take(5))

[['1', 'Kristina', 'Chung', '55', 'Pilot'], ['2', 'Paige', 'Chen', '74', 'Teacher'], ['3', 'Sherri', 'Melton', '34', 'Firefighter'], ['4', 'Gretchen', 'Hill', '66', 'Computer hardware engineer'], ['5', 'Karen', 'Puckett', '74', 'Lawyer']]

>>> statefilter =splited\_rdd.filter(lambda x: len(x)<3 )

>>> print(statefilter.take(4))

[['NM', 'New Mexico'], ['NY', 'New York'], ['NC', 'North Carolina'], ['ND', 'North Dakota']]

>>>

>>> complete\_df=spark.read.csv("/home/hduser/Desktop/Hackathon",sep=",",header=False,inferSchema=True)

Traceback (most recent call last):

File "<stdin>", line 1, in <module>

File "/usr/local/spark/python/pyspark/sql/readwriter.py", line 410, in csv

return self.\_df(self.\_jreader.csv(self.\_spark.\_sc.\_jvm.PythonUtils.toSeq(path)))

File "/usr/local/spark/python/lib/py4j-0.10.9.3-src.zip/py4j/java\_gateway.py", line 1322, in \_\_call\_\_

File "/usr/local/spark/python/pyspark/sql/utils.py", line 117, in deco

raise converted from None

pyspark.sql.utils.AnalysisException: Path does not exist: hdfs://localhost:54310/home/hduser/Desktop/Hackathon

>>> complete\_df=spark.read.csv("file:///home/hduser/Desktop/Hackathon",sep=",",header=False,inferSchema=True)

>>> complete\_df.show(3)

+--------+---------+------------+---------+----------+--------------------+--------------------+-------+--------------+--------------+

| \_c0| \_c1| \_c2| \_c3| \_c4| \_c5| \_c6| \_c7| \_c8| \_c9|

+--------+---------+------------+---------+----------+--------------------+--------------------+-------+--------------+--------------+

|IssuerId|IssuerId2|BusinessDate|StateCode|SourceName| NetworkName| NetworkURL|custnum|MarketCoverage|DentalOnlyPlan|

| 31195| 31195| 01-10-2019| SD| SERFF|Sanford Health Pl...|https://www.viiad...| 14| null| null|

| 60536| 60536| 01-10-2019| SD| SERFF| Avera Health Plans|www.Avera.org/Pro...| 13| null| null|

+--------+---------+------------+---------+----------+--------------------+--------------------+-------+--------------+--------------+

only showing top 3 rows

>>> complete\_df=spark.read.csv("file:///home/hduser/Desktop/Hackathon/custs\_states.csv",sep=",",header=False,inferSchema=True)

>>> complete\_df=spark.read.csv("file:///home/hduser/Desktop/Hackathon/custs\_states.csv",sep=",",header=False,inferSchema=True)

>>> complete\_df.show(3)

+---+----------+-----+----+-------+

|\_c0| \_c1| \_c2| \_c3| \_c4|

+---+----------+-----+----+-------+

| 1| Kristina|Chung| 55| Pilot|

| 2| Paige| Chen| 74|Teacher|

| NM|New Mexico| null|null| null|

+---+----------+-----+----+-------+

only showing top 3 rows

>>> custfilter\_df=complete\_df.select("\*").where("\_c0 is not null and \_c1 is not null and \_c2 is not null and \_c3 is not null and \_c4 is not null ")

>>> cust

custfilter custfilter\_df custom\_schema

>>> custfilter\_df.show(3)

+---+--------+------+---+-----------+

|\_c0| \_c1| \_c2|\_c3| \_c4|

+---+--------+------+---+-----------+

| 1|Kristina| Chung| 55| Pilot|

| 2| Paige| Chen| 74| Teacher|

| 3| Sherri|Melton| 34|Firefighter|

+---+--------+------+---+-----------+

only showing top 3 rows

>>> statefilter\_df=complete\_df.select("\_c0,\_c1").where(" \_c2 is null and \_c3 is null and \_c4 is null ")

Traceback (most recent call last):

File "<stdin>", line 1, in <module>

File "/usr/local/spark/python/pyspark/sql/dataframe.py", line 1685, in select

jdf = self.\_jdf.select(self.\_jcols(\*cols))

File "/usr/local/spark/python/lib/py4j-0.10.9.3-src.zip/py4j/java\_gateway.py", line 1322, in \_\_call\_\_

File "/usr/local/spark/python/pyspark/sql/utils.py", line 117, in deco

raise converted from None

pyspark.sql.utils.AnalysisException: cannot resolve '`\_c0,\_c1`' given input columns: [\_c0, \_c1, \_c2, \_c3, \_c4];

'Project ['\_c0,\_c1]

+- Relation [\_c0#26007,\_c1#26008,\_c2#26009,\_c3#26010,\_c4#26011] csv

>>> statefilter\_df=complete\_df.select("\_c0","\_c1").where(" \_c2 is null and \_c3 is null and \_c4 is null ")

>>> statefilter\_df.show(4)

+---+--------------+

|\_c0| \_c1|

+---+--------------+

| NM| New Mexico|

| NY| New York|

| NC|North Carolina|

| ND| North Dakota|

+---+--------------+

only showing top 4 rows

>>> custfilter\_df.createOrReplaceTempView("cust\_table")

>>> statefilter\_df.createOrReplaceTempView("state\_table")

>>> merged\_data\_frame.createOrReplaceTempView("insureview")

>>> df\_function("A+wer")

Column<'clean\_string(A+wer)'>

>>> df\_function("A+wer").show()

Traceback (most recent call last):

File "<stdin>", line 1, in <module>

TypeError: 'Column' object is not callable

>>> print(df\_function("A+wer"))

Column<'clean\_string(A+wer)'>

>>> print(df\_function(A+wer))

Traceback (most recent call last):

File "<stdin>", line 1, in <module>

NameError: name 'A' is not defined

>>> print(df\_function("A+wer"))

Column<'clean\_string(A+wer)'>

>>> print(df\_function("wer"))

Column<'clean\_string(wer)'>

>>> spark.conf.set("spark.sql.shuffle.partitions", 4)

>>> spark.udf.register("sql\_function", clean\_string, StringType())

<function clean\_string at 0x7fcc0c22b510>

>>> spark.sql("select \* from cust\_table").show(3)

+---+--------+------+---+-----------+

|\_c0| \_c1| \_c2|\_c3| \_c4|

+---+--------+------+---+-----------+

| 1|Kristina| Chung| 55| Pilot|

| 2| Paige| Chen| 74| Teacher|

| 3| Sherri|Melton| 34|Firefighter|

+---+--------+------+---+-----------+

only showing top 3 rows

>>> spark.sql("select \* from insureview").show(3)

+------------+----+-----+------------+--------------------+-------+--------------+--------------------+--------+---------+-----------------+----------+--------------------+

|BusinessDate|stcd|srcnm| NetworkName| NetworkURL|custnum|MarketCoverage| RejectRows|IssuerId|IssuerId2|issueridcomposite| sysdt| systs|

+------------+----+-----+------------+--------------------+-------+--------------+--------------------+--------+---------+-----------------+----------+--------------------+

| 2019-10-01| AK| HIOS| ODS Premier|https://www.modah...| 13| null| null| 21989| 21989| 2198921989|2024-07-22|2024-07-22 11:39:...|

| 2019-10-01| AK| HIOS| ODS Premier| null| null| null|219,219,2019-10-0...| 219| 219| 219219|2024-07-22|2024-07-22 11:39:...|

| 2019-10-01| AK| HIOS|HeritagePlus|https://www.preme...| 13| null| null| 38344| 38344| 3834438344|2024-07-22|2024-07-22 11:39:...|

+------------+----+-----+------------+--------------------+-------+--------------+--------------------+--------+---------+-----------------+----------+--------------------+

only showing top 3 rows

>>> spark.sql("select \*,sql\_function(NetworkName) as cleannetworkname from insureview").show(3)

+------------+----+-----+------------+--------------------+-------+--------------+--------------------+--------+---------+-----------------+----------+--------------------+----------------+

|BusinessDate|stcd|srcnm| NetworkName| NetworkURL|custnum|MarketCoverage| RejectRows|IssuerId|IssuerId2|issueridcomposite| sysdt| systs|cleannetworkname|

+------------+----+-----+------------+--------------------+-------+--------------+--------------------+--------+---------+-----------------+----------+--------------------+----------------+

| 2019-10-01| AK| HIOS| ODS Premier|https://www.modah...| 13| null| null| 21989| 21989| 2198921989|2024-07-22|2024-07-22 11:40:...| ODS Premier|

| 2019-10-01| AK| HIOS| ODS Premier| null| null| null|219,219,2019-10-0...| 219| 219| 219219|2024-07-22|2024-07-22 11:40:...| ODS Premier|

| 2019-10-01| AK| HIOS|HeritagePlus|https://www.preme...| 13| null| null| 38344| 38344| 3834438344|2024-07-22|2024-07-22 11:40:...| HeritagePlus|

+------------+----+-----+------------+--------------------+-------+--------------+--------------------+--------+---------+-----------------+----------+--------------------+----------------+

only showing top 3 rows

>>> spark.sql("select \*,current\_date as curdt , current\_timestamp as curts from insureview").show(3)

+------------+----+-----+------------+--------------------+-------+--------------+--------------------+--------+---------+-----------------+----------+--------------------+----------+--------------------+

|BusinessDate|stcd|srcnm| NetworkName| NetworkURL|custnum|MarketCoverage| RejectRows|IssuerId|IssuerId2|issueridcomposite| sysdt| systs| curdt| curts|

+------------+----+-----+------------+--------------------+-------+--------------+--------------------+--------+---------+-----------------+----------+--------------------+----------+--------------------+

| 2019-10-01| AK| HIOS| ODS Premier|https://www.modah...| 13| null| null| 21989| 21989| 2198921989|2024-07-22|2024-07-22 11:42:...|2024-07-22|2024-07-22 11:42:...|

| 2019-10-01| AK| HIOS| ODS Premier| null| null| null|219,219,2019-10-0...| 219| 219| 219219|2024-07-22|2024-07-22 11:42:...|2024-07-22|2024-07-22 11:42:...|

| 2019-10-01| AK| HIOS|HeritagePlus|https://www.preme...| 13| null| null| 38344| 38344| 3834438344|2024-07-22|2024-07-22 11:42:...|2024-07-22|2024-07-22 11:42:...|

+------------+----+-----+------------+--------------------+-------+--------------+--------------------+--------+---------+-----------------+----------+--------------------+----------+--------------------+

only showing top 3 rows

>>> spark.sql("select \*,BuisnessDate.month as month , BuisnessDate.year as year from insureview").show(3)

Traceback (most recent call last):

File "<stdin>", line 1, in <module>

File "/usr/local/spark/python/pyspark/sql/session.py", line 723, in sql

return DataFrame(self.\_jsparkSession.sql(sqlQuery), self.\_wrapped)

File "/usr/local/spark/python/lib/py4j-0.10.9.3-src.zip/py4j/java\_gateway.py", line 1322, in \_\_call\_\_

File "/usr/local/spark/python/pyspark/sql/utils.py", line 117, in deco

raise converted from None

pyspark.sql.utils.AnalysisException: cannot resolve 'BuisnessDate.month' given input columns: [insureview.BusinessDate, insureview.IssuerId, insureview.IssuerId2, insureview.MarketCoverage, insureview.NetworkName, insureview.NetworkURL, insureview.RejectRows, insureview.custnum, insureview.issueridcomposite, insureview.srcnm, insureview.stcd, insureview.sysdt, insureview.systs]; line 1 pos 9;

'Project [BusinessDate#1378, stcd#7759, srcnm#7747, NetworkName#1381, NetworkURL#1382, custnum#1383, MarketCoverage#1384, RejectRows#1386, IssuerId#8312, IssuerId2#8325, issueridcomposite#8901, sysdt#9003, systs#9016, 'BuisnessDate.month AS month#27694, 'BuisnessDate.year AS year#27695]

+- SubqueryAlias insureview

+- View (`insureview`, [BusinessDate#1378,stcd#7759,srcnm#7747,NetworkName#1381,NetworkURL#1382,custnum#1383,MarketCoverage#1384,RejectRows#1386,IssuerId#8312,IssuerId2#8325,issueridcomposite#8901,sysdt#9003,systs#9016])

+- Project [BusinessDate#1378, stcd#7759, srcnm#7747, NetworkName#1381, NetworkURL#1382, custnum#1383, MarketCoverage#1384, RejectRows#1386, IssuerId#8312, IssuerId2#8325, issueridcomposite#8901, sysdt#9003, current\_timestamp() AS systs#9016]

+- Project [BusinessDate#1378, stcd#7759, srcnm#7747, NetworkName#1381, NetworkURL#1382, custnum#1383, MarketCoverage#1384, RejectRows#1386, IssuerId#8312, IssuerId2#8325, issueridcomposite#8901, current\_date(Some(Asia/Kolkata)) AS sysdt#9003]

+- Project [BusinessDate#1378, stcd#7759, srcnm#7747, NetworkName#1381, NetworkURL#1382, custnum#1383, MarketCoverage#1384, RejectRows#1386, IssuerId#8312, IssuerId2#8325, issueridcomposite#8901]

+- Project [BusinessDate#1378, stcd#7759, srcnm#7747, NetworkName#1381, NetworkURL#1382, custnum#1383, MarketCoverage#1384, DentalOnlyPlan#1385, RejectRows#1386, IssuerId#8312, IssuerId2#8325, concat(IssuerId#8312, IssuerId2#8325) AS issueridcomposite#8901]

+- Project [BusinessDate#1378, stcd#7759, srcnm#7747, NetworkName#1381, NetworkURL#1382, custnum#1383, MarketCoverage#1384, DentalOnlyPlan#1385, RejectRows#1386, IssuerId#8312, IssuerId2#8325]

+- Project [IssuerIDDD#8273, BusinessDate#1378, stcd#7759, srcnm#7747, NetworkName#1381, NetworkURL#1382, custnum#1383, MarketCoverage#1384, DentalOnlyPlan#1385, RejectRows#1386, IssuerId#8312, IssuerId2#8325]

+- Project [IssuerIDD#8261, IssuerIDDD#8273, BusinessDate#1378, stcd#7759, srcnm#7747, NetworkName#1381, NetworkURL#1382, custnum#1383, MarketCoverage#1384, DentalOnlyPlan#1385, RejectRows#1386, IssuerId#8312, cast(IssuerIDDD#8273 as string) AS IssuerId2#8325]

+- Project [IssuerIDD#8261, IssuerIDDD#8273, BusinessDate#1378, stcd#7759, srcnm#7747, NetworkName#1381, NetworkURL#1382, custnum#1383, MarketCoverage#1384, DentalOnlyPlan#1385, RejectRows#1386, cast(IssuerIDD#8261 as string) AS IssuerId#8312]

+- Project [IssuerIDD#8261, IssuerId2#1377 AS IssuerIDDD#8273, BusinessDate#1378, stcd#7759, srcnm#7747, NetworkName#1381, NetworkURL#1382, custnum#1383, MarketCoverage#1384, DentalOnlyPlan#1385, RejectRows#1386]

+- Project [IssuerId#1376 AS IssuerIDD#8261, IssuerId2#1377, BusinessDate#1378, stcd#7759, srcnm#7747, NetworkName#1381, NetworkURL#1382, custnum#1383, MarketCoverage#1384, DentalOnlyPlan#1385, RejectRows#1386]

+- Project [IssuerId#1376, IssuerId2#1377, BusinessDate#1378, StateCode#1379 AS stcd#7759, srcnm#7747, NetworkName#1381, NetworkURL#1382, custnum#1383, MarketCoverage#1384, DentalOnlyPlan#1385, RejectRows#1386]

+- Project [IssuerId#1376, IssuerId2#1377, BusinessDate#1378, StateCode#1379, SourceName#1380 AS srcnm#7747, NetworkName#1381, NetworkURL#1382, custnum#1383, MarketCoverage#1384, DentalOnlyPlan#1385, RejectRows#1386]

+- Union false, false

:- Relation [IssuerId#1376,IssuerId2#1377,BusinessDate#1378,StateCode#1379,SourceName#1380,NetworkName#1381,NetworkURL#1382,custnum#1383,MarketCoverage#1384,DentalOnlyPlan#1385,RejectRows#1386] csv

+- Relation [IssuerId#491,IssuerId2#492,BusinessDate#493,StateCode#494,SourceName#495,NetworkName#496,NetworkURL#497,custnum#498,MarketCoverage#499,DentalOnlyPlan#500,RejectRows#501] csv

>>> spark.sql("select \*,BuisnessDate.month() as month , BuisnessDate.year() as year from insureview").show(3)

24/07/22 11:43:25 WARN metastore.ObjectStore: Failed to get database buisnessdate, returning NoSuchObjectException

Traceback (most recent call last):

File "<stdin>", line 1, in <module>

File "/usr/local/spark/python/pyspark/sql/session.py", line 723, in sql

return DataFrame(self.\_jsparkSession.sql(sqlQuery), self.\_wrapped)

File "/usr/local/spark/python/lib/py4j-0.10.9.3-src.zip/py4j/java\_gateway.py", line 1322, in \_\_call\_\_

File "/usr/local/spark/python/pyspark/sql/utils.py", line 117, in deco

raise converted from None

pyspark.sql.utils.AnalysisException: Undefined function: 'month'. This function is neither a registered temporary function nor a permanent function registered in the database 'BuisnessDate'.; line 1 pos 9

>>> spark.sql("select \*,month(BuisnessDate) as month ,year(BuisnessDate) as year from insureview").show(3)

Traceback (most recent call last):

File "<stdin>", line 1, in <module>

File "/usr/local/spark/python/pyspark/sql/session.py", line 723, in sql

return DataFrame(self.\_jsparkSession.sql(sqlQuery), self.\_wrapped)

File "/usr/local/spark/python/lib/py4j-0.10.9.3-src.zip/py4j/java\_gateway.py", line 1322, in \_\_call\_\_

File "/usr/local/spark/python/pyspark/sql/utils.py", line 117, in deco

raise converted from None

pyspark.sql.utils.AnalysisException: cannot resolve 'BuisnessDate' given input columns: [insureview.BusinessDate, insureview.IssuerId, insureview.IssuerId2, insureview.MarketCoverage, insureview.NetworkName, insureview.NetworkURL, insureview.RejectRows, insureview.custnum, insureview.issueridcomposite, insureview.srcnm, insureview.stcd, insureview.sysdt, insureview.systs]; line 1 pos 15;

'Project [BusinessDate#1378, stcd#7759, srcnm#7747, NetworkName#1381, NetworkURL#1382, custnum#1383, MarketCoverage#1384, RejectRows#1386, IssuerId#8312, IssuerId2#8325, issueridcomposite#8901, sysdt#9003, systs#9016, 'month('BuisnessDate) AS month#27698, 'year('BuisnessDate) AS year#27699]

+- SubqueryAlias insureview

+- View (`insureview`, [BusinessDate#1378,stcd#7759,srcnm#7747,NetworkName#1381,NetworkURL#1382,custnum#1383,MarketCoverage#1384,RejectRows#1386,IssuerId#8312,IssuerId2#8325,issueridcomposite#8901,sysdt#9003,systs#9016])

+- Project [BusinessDate#1378, stcd#7759, srcnm#7747, NetworkName#1381, NetworkURL#1382, custnum#1383, MarketCoverage#1384, RejectRows#1386, IssuerId#8312, IssuerId2#8325, issueridcomposite#8901, sysdt#9003, current\_timestamp() AS systs#9016]

+- Project [BusinessDate#1378, stcd#7759, srcnm#7747, NetworkName#1381, NetworkURL#1382, custnum#1383, MarketCoverage#1384, RejectRows#1386, IssuerId#8312, IssuerId2#8325, issueridcomposite#8901, current\_date(Some(Asia/Kolkata)) AS sysdt#9003]

+- Project [BusinessDate#1378, stcd#7759, srcnm#7747, NetworkName#1381, NetworkURL#1382, custnum#1383, MarketCoverage#1384, RejectRows#1386, IssuerId#8312, IssuerId2#8325, issueridcomposite#8901]

+- Project [BusinessDate#1378, stcd#7759, srcnm#7747, NetworkName#1381, NetworkURL#1382, custnum#1383, MarketCoverage#1384, DentalOnlyPlan#1385, RejectRows#1386, IssuerId#8312, IssuerId2#8325, concat(IssuerId#8312, IssuerId2#8325) AS issueridcomposite#8901]

+- Project [BusinessDate#1378, stcd#7759, srcnm#7747, NetworkName#1381, NetworkURL#1382, custnum#1383, MarketCoverage#1384, DentalOnlyPlan#1385, RejectRows#1386, IssuerId#8312, IssuerId2#8325]

+- Project [IssuerIDDD#8273, BusinessDate#1378, stcd#7759, srcnm#7747, NetworkName#1381, NetworkURL#1382, custnum#1383, MarketCoverage#1384, DentalOnlyPlan#1385, RejectRows#1386, IssuerId#8312, IssuerId2#8325]

+- Project [IssuerIDD#8261, IssuerIDDD#8273, BusinessDate#1378, stcd#7759, srcnm#7747, NetworkName#1381, NetworkURL#1382, custnum#1383, MarketCoverage#1384, DentalOnlyPlan#1385, RejectRows#1386, IssuerId#8312, cast(IssuerIDDD#8273 as string) AS IssuerId2#8325]

+- Project [IssuerIDD#8261, IssuerIDDD#8273, BusinessDate#1378, stcd#7759, srcnm#7747, NetworkName#1381, NetworkURL#1382, custnum#1383, MarketCoverage#1384, DentalOnlyPlan#1385, RejectRows#1386, cast(IssuerIDD#8261 as string) AS IssuerId#8312]

+- Project [IssuerIDD#8261, IssuerId2#1377 AS IssuerIDDD#8273, BusinessDate#1378, stcd#7759, srcnm#7747, NetworkName#1381, NetworkURL#1382, custnum#1383, MarketCoverage#1384, DentalOnlyPlan#1385, RejectRows#1386]

+- Project [IssuerId#1376 AS IssuerIDD#8261, IssuerId2#1377, BusinessDate#1378, stcd#7759, srcnm#7747, NetworkName#1381, NetworkURL#1382, custnum#1383, MarketCoverage#1384, DentalOnlyPlan#1385, RejectRows#1386]

+- Project [IssuerId#1376, IssuerId2#1377, BusinessDate#1378, StateCode#1379 AS stcd#7759, srcnm#7747, NetworkName#1381, NetworkURL#1382, custnum#1383, MarketCoverage#1384, DentalOnlyPlan#1385, RejectRows#1386]

+- Project [IssuerId#1376, IssuerId2#1377, BusinessDate#1378, StateCode#1379, SourceName#1380 AS srcnm#7747, NetworkName#1381, NetworkURL#1382, custnum#1383, MarketCoverage#1384, DentalOnlyPlan#1385, RejectRows#1386]

+- Union false, false

:- Relation [IssuerId#1376,IssuerId2#1377,BusinessDate#1378,StateCode#1379,SourceName#1380,NetworkName#1381,NetworkURL#1382,custnum#1383,MarketCoverage#1384,DentalOnlyPlan#1385,RejectRows#1386] csv

+- Relation [IssuerId#491,IssuerId2#492,BusinessDate#493,StateCode#494,SourceName#495,NetworkName#496,NetworkURL#497,custnum#498,MarketCoverage#499,DentalOnlyPlan#500,RejectRows#501] csv

>>> spark.sql("select \*,month(BuisnessDate) as month ,year(BuisnessDate) as year from insureview").show(3

...

Traceback (most recent call last):

File "/usr/local/spark/python/pyspark/context.py", line 293, in signal\_handler

raise KeyboardInterrupt()

KeyboardInterrupt

>>> spark.sql("describe insureview")

DataFrame[col\_name: string, data\_type: string, comment: string]

>>> spark.sql("describe insureview").show()

+-----------------+---------+-------+

| col\_name|data\_type|comment|

+-----------------+---------+-------+

| BusinessDate| date| null|

| stcd| string| null|

| srcnm| string| null|

| NetworkName| string| null|

| NetworkURL| string| null|

| custnum| string| null|

| MarketCoverage| string| null|

| RejectRows| string| null|

| IssuerId| string| null|

| IssuerId2| string| null|

|issueridcomposite| string| null|

| sysdt| date| null|

| systs|timestamp| null|

+-----------------+---------+-------+

>>> spark.sql("""select \*,month(BuisnessDate) as month ,year(BuisnessDate) as year from insureview""").show(3

... )

Traceback (most recent call last):

File "<stdin>", line 1, in <module>

File "/usr/local/spark/python/pyspark/sql/session.py", line 723, in sql

return DataFrame(self.\_jsparkSession.sql(sqlQuery), self.\_wrapped)

File "/usr/local/spark/python/lib/py4j-0.10.9.3-src.zip/py4j/java\_gateway.py", line 1322, in \_\_call\_\_

File "/usr/local/spark/python/pyspark/sql/utils.py", line 117, in deco

raise converted from None

pyspark.sql.utils.AnalysisException: cannot resolve 'BuisnessDate' given input columns: [insureview.BusinessDate, insureview.IssuerId, insureview.IssuerId2, insureview.MarketCoverage, insureview.NetworkName, insureview.NetworkURL, insureview.RejectRows, insureview.custnum, insureview.issueridcomposite, insureview.srcnm, insureview.stcd, insureview.sysdt, insureview.systs]; line 1 pos 15;

'Project [BusinessDate#1378, stcd#7759, srcnm#7747, NetworkName#1381, NetworkURL#1382, custnum#1383, MarketCoverage#1384, RejectRows#1386, IssuerId#8312, IssuerId2#8325, issueridcomposite#8901, sysdt#9003, systs#9016, 'month('BuisnessDate) AS month#27725, 'year('BuisnessDate) AS year#27726]

+- SubqueryAlias insureview

+- View (`insureview`, [BusinessDate#1378,stcd#7759,srcnm#7747,NetworkName#1381,NetworkURL#1382,custnum#1383,MarketCoverage#1384,RejectRows#1386,IssuerId#8312,IssuerId2#8325,issueridcomposite#8901,sysdt#9003,systs#9016])

+- Project [BusinessDate#1378, stcd#7759, srcnm#7747, NetworkName#1381, NetworkURL#1382, custnum#1383, MarketCoverage#1384, RejectRows#1386, IssuerId#8312, IssuerId2#8325, issueridcomposite#8901, sysdt#9003, current\_timestamp() AS systs#9016]

+- Project [BusinessDate#1378, stcd#7759, srcnm#7747, NetworkName#1381, NetworkURL#1382, custnum#1383, MarketCoverage#1384, RejectRows#1386, IssuerId#8312, IssuerId2#8325, issueridcomposite#8901, current\_date(Some(Asia/Kolkata)) AS sysdt#9003]

+- Project [BusinessDate#1378, stcd#7759, srcnm#7747, NetworkName#1381, NetworkURL#1382, custnum#1383, MarketCoverage#1384, RejectRows#1386, IssuerId#8312, IssuerId2#8325, issueridcomposite#8901]

+- Project [BusinessDate#1378, stcd#7759, srcnm#7747, NetworkName#1381, NetworkURL#1382, custnum#1383, MarketCoverage#1384, DentalOnlyPlan#1385, RejectRows#1386, IssuerId#8312, IssuerId2#8325, concat(IssuerId#8312, IssuerId2#8325) AS issueridcomposite#8901]

+- Project [BusinessDate#1378, stcd#7759, srcnm#7747, NetworkName#1381, NetworkURL#1382, custnum#1383, MarketCoverage#1384, DentalOnlyPlan#1385, RejectRows#1386, IssuerId#8312, IssuerId2#8325]

+- Project [IssuerIDDD#8273, BusinessDate#1378, stcd#7759, srcnm#7747, NetworkName#1381, NetworkURL#1382, custnum#1383, MarketCoverage#1384, DentalOnlyPlan#1385, RejectRows#1386, IssuerId#8312, IssuerId2#8325]

+- Project [IssuerIDD#8261, IssuerIDDD#8273, BusinessDate#1378, stcd#7759, srcnm#7747, NetworkName#1381, NetworkURL#1382, custnum#1383, MarketCoverage#1384, DentalOnlyPlan#1385, RejectRows#1386, IssuerId#8312, cast(IssuerIDDD#8273 as string) AS IssuerId2#8325]

+- Project [IssuerIDD#8261, IssuerIDDD#8273, BusinessDate#1378, stcd#7759, srcnm#7747, NetworkName#1381, NetworkURL#1382, custnum#1383, MarketCoverage#1384, DentalOnlyPlan#1385, RejectRows#1386, cast(IssuerIDD#8261 as string) AS IssuerId#8312]

+- Project [IssuerIDD#8261, IssuerId2#1377 AS IssuerIDDD#8273, BusinessDate#1378, stcd#7759, srcnm#7747, NetworkName#1381, NetworkURL#1382, custnum#1383, MarketCoverage#1384, DentalOnlyPlan#1385, RejectRows#1386]

+- Project [IssuerId#1376 AS IssuerIDD#8261, IssuerId2#1377, BusinessDate#1378, stcd#7759, srcnm#7747, NetworkName#1381, NetworkURL#1382, custnum#1383, MarketCoverage#1384, DentalOnlyPlan#1385, RejectRows#1386]

+- Project [IssuerId#1376, IssuerId2#1377, BusinessDate#1378, StateCode#1379 AS stcd#7759, srcnm#7747, NetworkName#1381, NetworkURL#1382, custnum#1383, MarketCoverage#1384, DentalOnlyPlan#1385, RejectRows#1386]

+- Project [IssuerId#1376, IssuerId2#1377, BusinessDate#1378, StateCode#1379, SourceName#1380 AS srcnm#7747, NetworkName#1381, NetworkURL#1382, custnum#1383, MarketCoverage#1384, DentalOnlyPlan#1385, RejectRows#1386]

+- Union false, false

:- Relation [IssuerId#1376,IssuerId2#1377,BusinessDate#1378,StateCode#1379,SourceName#1380,NetworkName#1381,NetworkURL#1382,custnum#1383,MarketCoverage#1384,DentalOnlyPlan#1385,RejectRows#1386] csv

+- Relation [IssuerId#491,IssuerId2#492,BusinessDate#493,StateCode#494,SourceName#495,NetworkName#496,NetworkURL#497,custnum#498,MarketCoverage#499,DentalOnlyPlan#500,RejectRows#501] csv

>>> spark.sql("""select \*,month(BuisnessDate) as month ,year(BuisnessDate) as year from insureview""").show(3)

Traceback (most recent call last):

File "<stdin>", line 1, in <module>

File "/usr/local/spark/python/pyspark/sql/session.py", line 723, in sql

return DataFrame(self.\_jsparkSession.sql(sqlQuery), self.\_wrapped)

File "/usr/local/spark/python/lib/py4j-0.10.9.3-src.zip/py4j/java\_gateway.py", line 1322, in \_\_call\_\_

File "/usr/local/spark/python/pyspark/sql/utils.py", line 117, in deco

raise converted from None

pyspark.sql.utils.AnalysisException: cannot resolve 'BuisnessDate' given input columns: [insureview.BusinessDate, insureview.IssuerId, insureview.IssuerId2, insureview.MarketCoverage, insureview.NetworkName, insureview.NetworkURL, insureview.RejectRows, insureview.custnum, insureview.issueridcomposite, insureview.srcnm, insureview.stcd, insureview.sysdt, insureview.systs]; line 1 pos 15;

'Project [BusinessDate#1378, stcd#7759, srcnm#7747, NetworkName#1381, NetworkURL#1382, custnum#1383, MarketCoverage#1384, RejectRows#1386, IssuerId#8312, IssuerId2#8325, issueridcomposite#8901, sysdt#9003, systs#9016, 'month('BuisnessDate) AS month#27727, 'year('BuisnessDate) AS year#27728]

+- SubqueryAlias insureview

+- View (`insureview`, [BusinessDate#1378,stcd#7759,srcnm#7747,NetworkName#1381,NetworkURL#1382,custnum#1383,MarketCoverage#1384,RejectRows#1386,IssuerId#8312,IssuerId2#8325,issueridcomposite#8901,sysdt#9003,systs#9016])

+- Project [BusinessDate#1378, stcd#7759, srcnm#7747, NetworkName#1381, NetworkURL#1382, custnum#1383, MarketCoverage#1384, RejectRows#1386, IssuerId#8312, IssuerId2#8325, issueridcomposite#8901, sysdt#9003, current\_timestamp() AS systs#9016]

+- Project [BusinessDate#1378, stcd#7759, srcnm#7747, NetworkName#1381, NetworkURL#1382, custnum#1383, MarketCoverage#1384, RejectRows#1386, IssuerId#8312, IssuerId2#8325, issueridcomposite#8901, current\_date(Some(Asia/Kolkata)) AS sysdt#9003]

+- Project [BusinessDate#1378, stcd#7759, srcnm#7747, NetworkName#1381, NetworkURL#1382, custnum#1383, MarketCoverage#1384, RejectRows#1386, IssuerId#8312, IssuerId2#8325, issueridcomposite#8901]

+- Project [BusinessDate#1378, stcd#7759, srcnm#7747, NetworkName#1381, NetworkURL#1382, custnum#1383, MarketCoverage#1384, DentalOnlyPlan#1385, RejectRows#1386, IssuerId#8312, IssuerId2#8325, concat(IssuerId#8312, IssuerId2#8325) AS issueridcomposite#8901]

+- Project [BusinessDate#1378, stcd#7759, srcnm#7747, NetworkName#1381, NetworkURL#1382, custnum#1383, MarketCoverage#1384, DentalOnlyPlan#1385, RejectRows#1386, IssuerId#8312, IssuerId2#8325]

+- Project [IssuerIDDD#8273, BusinessDate#1378, stcd#7759, srcnm#7747, NetworkName#1381, NetworkURL#1382, custnum#1383, MarketCoverage#1384, DentalOnlyPlan#1385, RejectRows#1386, IssuerId#8312, IssuerId2#8325]

+- Project [IssuerIDD#8261, IssuerIDDD#8273, BusinessDate#1378, stcd#7759, srcnm#7747, NetworkName#1381, NetworkURL#1382, custnum#1383, MarketCoverage#1384, DentalOnlyPlan#1385, RejectRows#1386, IssuerId#8312, cast(IssuerIDDD#8273 as string) AS IssuerId2#8325]

+- Project [IssuerIDD#8261, IssuerIDDD#8273, BusinessDate#1378, stcd#7759, srcnm#7747, NetworkName#1381, NetworkURL#1382, custnum#1383, MarketCoverage#1384, DentalOnlyPlan#1385, RejectRows#1386, cast(IssuerIDD#8261 as string) AS IssuerId#8312]

+- Project [IssuerIDD#8261, IssuerId2#1377 AS IssuerIDDD#8273, BusinessDate#1378, stcd#7759, srcnm#7747, NetworkName#1381, NetworkURL#1382, custnum#1383, MarketCoverage#1384, DentalOnlyPlan#1385, RejectRows#1386]

+- Project [IssuerId#1376 AS IssuerIDD#8261, IssuerId2#1377, BusinessDate#1378, stcd#7759, srcnm#7747, NetworkName#1381, NetworkURL#1382, custnum#1383, MarketCoverage#1384, DentalOnlyPlan#1385, RejectRows#1386]

+- Project [IssuerId#1376, IssuerId2#1377, BusinessDate#1378, StateCode#1379 AS stcd#7759, srcnm#7747, NetworkName#1381, NetworkURL#1382, custnum#1383, MarketCoverage#1384, DentalOnlyPlan#1385, RejectRows#1386]

+- Project [IssuerId#1376, IssuerId2#1377, BusinessDate#1378, StateCode#1379, SourceName#1380 AS srcnm#7747, NetworkName#1381, NetworkURL#1382, custnum#1383, MarketCoverage#1384, DentalOnlyPlan#1385, RejectRows#1386]

+- Union false, false

:- Relation [IssuerId#1376,IssuerId2#1377,BusinessDate#1378,StateCode#1379,SourceName#1380,NetworkName#1381,NetworkURL#1382,custnum#1383,MarketCoverage#1384,DentalOnlyPlan#1385,RejectRows#1386] csv

+- Relation [IssuerId#491,IssuerId2#492,BusinessDate#493,StateCode#494,SourceName#495,NetworkName#496,NetworkURL#497,custnum#498,MarketCoverage#499,DentalOnlyPlan#500,RejectRows#501] csv

>>> spark.sql("""

... SELECT \*,

... month(BusinessDate) AS month,

... year(BusinessDate) AS year

... FROM insureview

... """)

DataFrame[BusinessDate: date, stcd: string, srcnm: string, NetworkName: string, NetworkURL: string, custnum: string, MarketCoverage: string, RejectRows: string, IssuerId: string, IssuerId2: string, issueridcomposite: string, sysdt: date, systs: timestamp, month: int, year: int]

>>> result\_df.show(3)

Traceback (most recent call last):

File "<stdin>", line 1, in <module>

NameError: name 'result\_df' is not defined

>>> spark.sql("""spark.sql("""SELECT \*, month(BusinessDate) AS month, year(BusinessDate) AS year FROM insureview""")

File "<stdin>", line 1

spark.sql("""spark.sql("""SELECT \*, month(BusinessDate) AS month, year(BusinessDate) AS year FROM insureview""")

^

SyntaxError: invalid syntax

>>> spark.sql("""SELECT \*, month(BusinessDate) AS month, year(BusinessDate) AS year FROM insureview""")

DataFrame[BusinessDate: date, stcd: string, srcnm: string, NetworkName: string, NetworkURL: string, custnum: string, MarketCoverage: string, RejectRows: string, IssuerId: string, IssuerId2: string, issueridcomposite: string, sysdt: date, systs: timestamp, month: int, year: int]

>>> spark.sql("""SELECT \*, month(BusinessDate) AS month, year(BusinessDate) AS year FROM insureview""").show()

+------------+----+-----+--------------------+--------------------+-------+--------------+--------------------+--------+---------+-----------------+----------+--------------------+-----+----+

|BusinessDate|stcd|srcnm| NetworkName| NetworkURL|custnum|MarketCoverage| RejectRows|IssuerId|IssuerId2|issueridcomposite| sysdt| systs|month|year|

+------------+----+-----+--------------------+--------------------+-------+--------------+--------------------+--------+---------+-----------------+----------+--------------------+-----+----+

| 2019-10-01| AK| HIOS| ODS Premier|https://www.modah...| 13| null| null| 21989| 21989| 2198921989|2024-07-22|2024-07-22 11:47:...| 10|2019|

| 2019-10-01| AK| HIOS| ODS Premier| null| null| null|219,219,2019-10-0...| 219| 219| 219219|2024-07-22|2024-07-22 11:47:...| 10|2019|

| 2019-10-01| AK| HIOS| HeritagePlus|https://www.preme...| 13| null| null| 38344| 38344| 3834438344|2024-07-22|2024-07-22 11:47:...| 10|2019|

| 2019-10-01| AK| HIOS|Lincoln Dental Co...|http://lfg.go2den...| 13| null| null| 38536| 38536| 3853638536|2024-07-22|2024-07-22 11:47:...| 10|2019|

| 2019-10-01| AK| HIOS|DentalGuard Prefe...|https://www.guard...| 13| null| null| 42507| 42507| 4250742507|2024-07-22|2024-07-22 11:47:...| 10|2019|

| 2019-10-01| AK| HIOS|Moda Plus AK Regi...|https://www.modah...| 13| null| null| 73836| 73836| 7383673836|2024-07-22|2024-07-22 11:47:...| 10|2019|

| 2019-10-01| AK| HIOS|Moda Plus Providence|https://www.modah...| 14| null| null| 73836| 73836| 7383673836|2024-07-22|2024-07-22 11:47:...| 10|2019|

| 2019-10-01| AK| HIOS| DenteMax|http://www2.dente...| 13| null| null| 74819| 74819| 7481974819|2024-07-22|2024-07-22 11:47:...| 10|2019|

| 2019-10-01| AK| HIOS|Ameritas PPO Dent...| www.standard.com| 13| null| null| 84859| 84859| 8485984859|2024-07-22|2024-07-22 11:47:...| 10|2019|

| 2019-10-01| AL| HIOS| DenteMax|http://www2.dente...| 13| null| null| 12538| 12538| 1253812538|2024-07-22|2024-07-22 11:47:...| 10|2019|

| 2019-10-01| AL| HIOS|CONNECTION Dental...|http://www.ppousa...| 14| null| null| 12538| 12538| 1253812538|2024-07-22|2024-07-22 11:47:...| 10|2019|

| 2019-10-01| AL| HIOS|DentalGuard Prefe...|https://www.guard...| 13| null| null| 17540| 17540| 1754017540|2024-07-22|2024-07-22 11:47:...| 10|2019|

| 2019-10-01| AL| HIOS| Birmingham PPOx|http://pfp.humana...| 13| null| null| 44580| 44580| 4458044580|2024-07-22|2024-07-22 11:47:...| 10|2019|

| 2019-10-01| AL| HIOS| Huntsville PPOx|http://pfp.humana...| 14| null| null| 44580| 44580| 4458044580|2024-07-22|2024-07-22 11:47:...| 10|2019|

| 2019-10-01| AL| HIOS|Humana Dental PPO...|https://www.human...| 15| null| null| 44580| 44580| 4458044580|2024-07-22|2024-07-22 11:47:...| 10|2019|

| 2019-10-01| AL| HIOS| Advantage Plus|https://secure.uc...| 13| null| null| 44785| 44785| 4478544785|2024-07-22|2024-07-22 11:47:...| 10|2019|

| 2019-10-01| AL| HIOS|Lincoln Dental Co...|http://lfg.go2den...| 13| null| null| 46211| 46211| 4621146211|2024-07-22|2024-07-22 11:47:...| 10|2019|

| 2019-10-01| AL| HIOS| BCBSAL PPO|https://www.bcbsa...| 13| null| null| 46944| 46944| 4694446944|2024-07-22|2024-07-22 11:47:...| 10|2019|

| 2019-10-01| AL| HIOS| Choice Plus|http://www.provid...| 13| null| null| 59809| 59809| 5980959809|2024-07-22|2024-07-22 11:47:...| 10|2019|

| 2019-10-01| AL| HIOS| Choice Plus|http://www.provid...| 14| null| null| 59809| 59809| 5980959809|2024-07-22|2024-07-22 11:47:...| 10|2019|

+------------+----+-----+--------------------+--------------------+-------+--------------+--------------------+--------+---------+-----------------+----------+--------------------+-----+----+

only showing top 20 rows

>>> spark.sql("""SELECT \*, month(BusinessDate) AS month, year(BusinessDate) AS year FROM insureview""").show(4)

+------------+----+-----+--------------------+--------------------+-------+--------------+--------------------+--------+---------+-----------------+----------+--------------------+-----+----+

|BusinessDate|stcd|srcnm| NetworkName| NetworkURL|custnum|MarketCoverage| RejectRows|IssuerId|IssuerId2|issueridcomposite| sysdt| systs|month|year|

+------------+----+-----+--------------------+--------------------+-------+--------------+--------------------+--------+---------+-----------------+----------+--------------------+-----+----+

| 2019-10-01| AK| HIOS| ODS Premier|https://www.modah...| 13| null| null| 21989| 21989| 2198921989|2024-07-22|2024-07-22 11:48:...| 10|2019|

| 2019-10-01| AK| HIOS| ODS Premier| null| null| null|219,219,2019-10-0...| 219| 219| 219219|2024-07-22|2024-07-22 11:48:...| 10|2019|

| 2019-10-01| AK| HIOS| HeritagePlus|https://www.preme...| 13| null| null| 38344| 38344| 3834438344|2024-07-22|2024-07-22 11:48:...| 10|2019|

| 2019-10-01| AK| HIOS|Lincoln Dental Co...|http://lfg.go2den...| 13| null| null| 38536| 38536| 3853638536|2024-07-22|2024-07-22 11:48:...| 10|2019|

+------------+----+-----+--------------------+--------------------+-------+--------------+--------------------+--------+---------+-----------------+----------+--------------------+-----+----+

only showing top 4 rows

>>> spark.sql("""select \*,month(BuisnessDate) as month ,year(BuisnessDate) as year from insureview""").show(3)

Traceback (most recent call last):

File "<stdin>", line 1, in <module>

File "/usr/local/spark/python/pyspark/sql/session.py", line 723, in sql

return DataFrame(self.\_jsparkSession.sql(sqlQuery), self.\_wrapped)

File "/usr/local/spark/python/lib/py4j-0.10.9.3-src.zip/py4j/java\_gateway.py", line 1322, in \_\_call\_\_

File "/usr/local/spark/python/pyspark/sql/utils.py", line 117, in deco

raise converted from None

pyspark.sql.utils.AnalysisException: cannot resolve 'BuisnessDate' given input columns: [insureview.BusinessDate, insureview.IssuerId, insureview.IssuerId2, insureview.MarketCoverage, insureview.NetworkName, insureview.NetworkURL, insureview.RejectRows, insureview.custnum, insureview.issueridcomposite, insureview.srcnm, insureview.stcd, insureview.sysdt, insureview.systs]; line 1 pos 15;

'Project [BusinessDate#1378, stcd#7759, srcnm#7747, NetworkName#1381, NetworkURL#1382, custnum#1383, MarketCoverage#1384, RejectRows#1386, IssuerId#8312, IssuerId2#8325, issueridcomposite#8901, sysdt#9003, systs#9016, 'month('BuisnessDate) AS month#28819, 'year('BuisnessDate) AS year#28820]

+- SubqueryAlias insureview

+- View (`insureview`, [BusinessDate#1378,stcd#7759,srcnm#7747,NetworkName#1381,NetworkURL#1382,custnum#1383,MarketCoverage#1384,RejectRows#1386,IssuerId#8312,IssuerId2#8325,issueridcomposite#8901,sysdt#9003,systs#9016])

+- Project [BusinessDate#1378, stcd#7759, srcnm#7747, NetworkName#1381, NetworkURL#1382, custnum#1383, MarketCoverage#1384, RejectRows#1386, IssuerId#8312, IssuerId2#8325, issueridcomposite#8901, sysdt#9003, current\_timestamp() AS systs#9016]

+- Project [BusinessDate#1378, stcd#7759, srcnm#7747, NetworkName#1381, NetworkURL#1382, custnum#1383, MarketCoverage#1384, RejectRows#1386, IssuerId#8312, IssuerId2#8325, issueridcomposite#8901, current\_date(Some(Asia/Kolkata)) AS sysdt#9003]

+- Project [BusinessDate#1378, stcd#7759, srcnm#7747, NetworkName#1381, NetworkURL#1382, custnum#1383, MarketCoverage#1384, RejectRows#1386, IssuerId#8312, IssuerId2#8325, issueridcomposite#8901]

+- Project [BusinessDate#1378, stcd#7759, srcnm#7747, NetworkName#1381, NetworkURL#1382, custnum#1383, MarketCoverage#1384, DentalOnlyPlan#1385, RejectRows#1386, IssuerId#8312, IssuerId2#8325, concat(IssuerId#8312, IssuerId2#8325) AS issueridcomposite#8901]

+- Project [BusinessDate#1378, stcd#7759, srcnm#7747, NetworkName#1381, NetworkURL#1382, custnum#1383, MarketCoverage#1384, DentalOnlyPlan#1385, RejectRows#1386, IssuerId#8312, IssuerId2#8325]

+- Project [IssuerIDDD#8273, BusinessDate#1378, stcd#7759, srcnm#7747, NetworkName#1381, NetworkURL#1382, custnum#1383, MarketCoverage#1384, DentalOnlyPlan#1385, RejectRows#1386, IssuerId#8312, IssuerId2#8325]

+- Project [IssuerIDD#8261, IssuerIDDD#8273, BusinessDate#1378, stcd#7759, srcnm#7747, NetworkName#1381, NetworkURL#1382, custnum#1383, MarketCoverage#1384, DentalOnlyPlan#1385, RejectRows#1386, IssuerId#8312, cast(IssuerIDDD#8273 as string) AS IssuerId2#8325]

+- Project [IssuerIDD#8261, IssuerIDDD#8273, BusinessDate#1378, stcd#7759, srcnm#7747, NetworkName#1381, NetworkURL#1382, custnum#1383, MarketCoverage#1384, DentalOnlyPlan#1385, RejectRows#1386, cast(IssuerIDD#8261 as string) AS IssuerId#8312]

+- Project [IssuerIDD#8261, IssuerId2#1377 AS IssuerIDDD#8273, BusinessDate#1378, stcd#7759, srcnm#7747, NetworkName#1381, NetworkURL#1382, custnum#1383, MarketCoverage#1384, DentalOnlyPlan#1385, RejectRows#1386]

+- Project [IssuerId#1376 AS IssuerIDD#8261, IssuerId2#1377, BusinessDate#1378, stcd#7759, srcnm#7747, NetworkName#1381, NetworkURL#1382, custnum#1383, MarketCoverage#1384, DentalOnlyPlan#1385, RejectRows#1386]

+- Project [IssuerId#1376, IssuerId2#1377, BusinessDate#1378, StateCode#1379 AS stcd#7759, srcnm#7747, NetworkName#1381, NetworkURL#1382, custnum#1383, MarketCoverage#1384, DentalOnlyPlan#1385, RejectRows#1386]

+- Project [IssuerId#1376, IssuerId2#1377, BusinessDate#1378, StateCode#1379, SourceName#1380 AS srcnm#7747, NetworkName#1381, NetworkURL#1382, custnum#1383, MarketCoverage#1384, DentalOnlyPlan#1385, RejectRows#1386]

+- Union false, false

:- Relation [IssuerId#1376,IssuerId2#1377,BusinessDate#1378,StateCode#1379,SourceName#1380,NetworkName#1381,NetworkURL#1382,custnum#1383,MarketCoverage#1384,DentalOnlyPlan#1385,RejectRows#1386] csv

+- Relation [IssuerId#491,IssuerId2#492,BusinessDate#493,StateCode#494,SourceName#495,NetworkName#496,NetworkURL#497,custnum#498,MarketCoverage#499,DentalOnlyPlan#500,RejectRows#501] csv

>>> spark.sql("""SELECT \*, month(BusinessDate) AS month, year(BusinessDate) AS year FROM insureview""").show(4)

+------------+----+-----+--------------------+--------------------+-------+--------------+--------------------+--------+---------+-----------------+----------+--------------------+-----+----+

|BusinessDate|stcd|srcnm| NetworkName| NetworkURL|custnum|MarketCoverage| RejectRows|IssuerId|IssuerId2|issueridcomposite| sysdt| systs|month|year|

+------------+----+-----+--------------------+--------------------+-------+--------------+--------------------+--------+---------+-----------------+----------+--------------------+-----+----+

| 2019-10-01| AK| HIOS| ODS Premier|https://www.modah...| 13| null| null| 21989| 21989| 2198921989|2024-07-22|2024-07-22 11:48:...| 10|2019|

| 2019-10-01| AK| HIOS| ODS Premier| null| null| null|219,219,2019-10-0...| 219| 219| 219219|2024-07-22|2024-07-22 11:48:...| 10|2019|

| 2019-10-01| AK| HIOS| HeritagePlus|https://www.preme...| 13| null| null| 38344| 38344| 3834438344|2024-07-22|2024-07-22 11:48:...| 10|2019|

| 2019-10-01| AK| HIOS|Lincoln Dental Co...|http://lfg.go2den...| 13| null| null| 38536| 38536| 3853638536|2024-07-22|2024-07-22 11:48:...| 10|2019|

+------------+----+-----+--------------------+--------------------+-------+--------------+--------------------+--------+---------+-----------------+----------+--------------------+-----+----+

only showing top 4 rows

>>> spark.sql("""select \*,month(BusinessDate) as month ,year(BusinessDate) as year from insureview""").show(3)

+------------+----+-----+------------+--------------------+-------+--------------+--------------------+--------+---------+-----------------+----------+--------------------+-----+----+

|BusinessDate|stcd|srcnm| NetworkName| NetworkURL|custnum|MarketCoverage| RejectRows|IssuerId|IssuerId2|issueridcomposite| sysdt| systs|month|year|

+------------+----+-----+------------+--------------------+-------+--------------+--------------------+--------+---------+-----------------+----------+--------------------+-----+----+

| 2019-10-01| AK| HIOS| ODS Premier|https://www.modah...| 13| null| null| 21989| 21989| 2198921989|2024-07-22|2024-07-22 11:49:...| 10|2019|

| 2019-10-01| AK| HIOS| ODS Premier| null| null| null|219,219,2019-10-0...| 219| 219| 219219|2024-07-22|2024-07-22 11:49:...| 10|2019|

| 2019-10-01| AK| HIOS|HeritagePlus|https://www.preme...| 13| null| null| 38344| 38344| 3834438344|2024-07-22|2024-07-22 11:49:...| 10|2019|

+------------+----+-----+------------+--------------------+-------+--------------+--------------------+--------+---------+-----------------+----------+--------------------+-----+----+

only showing top 3 rows

>>> spark.sql("""select \*,case when NetworkURL like 'https://' then 'http secured' case when NetworkURL like 'http://' then 'http non secured' else 'No Protocol' as Protocol\_Used from insureview""").show(3)

Traceback (most recent call last):

File "<stdin>", line 1, in <module>

File "/usr/local/spark/python/pyspark/sql/session.py", line 723, in sql

return DataFrame(self.\_jsparkSession.sql(sqlQuery), self.\_wrapped)

File "/usr/local/spark/python/lib/py4j-0.10.9.3-src.zip/py4j/java\_gateway.py", line 1322, in \_\_call\_\_

File "/usr/local/spark/python/pyspark/sql/utils.py", line 117, in deco

raise converted from None

pyspark.sql.utils.ParseException:

mismatched input 'NetworkURL' expecting {<EOF>, ';'}(line 1, pos 19)

== SQL ==

select \*,case when NetworkURL like 'https://' then 'http secured' case when NetworkURL like 'http://' then 'http non secured' else 'No Protocol' as Protocol\_Used from insureview

-------------------^^^

>>> spark.sql("""select \*,case when NetworkURL like 'https://' then 'http secured' when NetworkURL like 'http://' then 'http non secured' else 'No Protocol' as Protocol\_Used from insureview""").show(3)

Traceback (most recent call last):

File "<stdin>", line 1, in <module>

File "/usr/local/spark/python/pyspark/sql/session.py", line 723, in sql

return DataFrame(self.\_jsparkSession.sql(sqlQuery), self.\_wrapped)

File "/usr/local/spark/python/lib/py4j-0.10.9.3-src.zip/py4j/java\_gateway.py", line 1322, in \_\_call\_\_

File "/usr/local/spark/python/pyspark/sql/utils.py", line 117, in deco

raise converted from None

pyspark.sql.utils.ParseException:

mismatched input 'NetworkURL' expecting {<EOF>, ';'}(line 1, pos 19)

== SQL ==

select \*,case when NetworkURL like 'https://' then 'http secured' when NetworkURL like 'http://' then 'http non secured' else 'No Protocol' as Protocol\_Used from insureview

-------------------^^^

>>> spark.sql("""select \*,case when NetworkURL like 'https://%' then 'http secured' when NetworkURL like 'http://%' then 'http non secured' else 'No Protocol' as Protocol\_Used from insureview""").show(3)

Traceback (most recent call last):

File "<stdin>", line 1, in <module>

File "/usr/local/spark/python/pyspark/sql/session.py", line 723, in sql

return DataFrame(self.\_jsparkSession.sql(sqlQuery), self.\_wrapped)

File "/usr/local/spark/python/lib/py4j-0.10.9.3-src.zip/py4j/java\_gateway.py", line 1322, in \_\_call\_\_

File "/usr/local/spark/python/pyspark/sql/utils.py", line 117, in deco

raise converted from None

pyspark.sql.utils.ParseException:

mismatched input 'NetworkURL' expecting {<EOF>, ';'}(line 1, pos 19)

== SQL ==

select \*,case when NetworkURL like 'https://%' then 'http secured' when NetworkURL like 'http://%' then 'http non secured' else 'No Protocol' as Protocol\_Used from insureview

-------------------^^^

>>> spark.sql("""select \*,case when NetworkURL like 'https://%' then 'http secured' when NetworkURL like 'http://%' then 'http non secured' else 'No Protocol' end as Protocol\_Used from insureview""").show(3)

+------------+----+-----+------------+--------------------+-------+--------------+--------------------+--------+---------+-----------------+----------+--------------------+-------------+

|BusinessDate|stcd|srcnm| NetworkName| NetworkURL|custnum|MarketCoverage| RejectRows|IssuerId|IssuerId2|issueridcomposite| sysdt| systs|Protocol\_Used|

+------------+----+-----+------------+--------------------+-------+--------------+--------------------+--------+---------+-----------------+----------+--------------------+-------------+

| 2019-10-01| AK| HIOS| ODS Premier|https://www.modah...| 13| null| null| 21989| 21989| 2198921989|2024-07-22|2024-07-22 11:55:...| http secured|

| 2019-10-01| AK| HIOS| ODS Premier| null| null| null|219,219,2019-10-0...| 219| 219| 219219|2024-07-22|2024-07-22 11:55:...| No Protocol|

| 2019-10-01| AK| HIOS|HeritagePlus|https://www.preme...| 13| null| null| 38344| 38344| 3834438344|2024-07-22|2024-07-22 11:55:...| http secured|

+------------+----+-----+------------+--------------------+-------+--------------+--------------------+--------+---------+-----------------+----------+--------------------+-------------+

only showing top 3 rows

>>> result\_df=spark.sql("""select \*,month(BuisnessDate) as month ,year(BuisnessDate) as year from insureview""")

Traceback (most recent call last):

File "<stdin>", line 1, in <module>

File "/usr/local/spark/python/pyspark/sql/session.py", line 723, in sql

return DataFrame(self.\_jsparkSession.sql(sqlQuery), self.\_wrapped)

File "/usr/local/spark/python/lib/py4j-0.10.9.3-src.zip/py4j/java\_gateway.py", line 1322, in \_\_call\_\_

File "/usr/local/spark/python/pyspark/sql/utils.py", line 117, in deco

raise converted from None

pyspark.sql.utils.AnalysisException: cannot resolve 'BuisnessDate' given input columns: [insureview.BusinessDate, insureview.IssuerId, insureview.IssuerId2, insureview.MarketCoverage, insureview.NetworkName, insureview.NetworkURL, insureview.RejectRows, insureview.custnum, insureview.issueridcomposite, insureview.srcnm, insureview.stcd, insureview.sysdt, insureview.systs]; line 1 pos 15;

'Project [BusinessDate#1378, stcd#7759, srcnm#7747, NetworkName#1381, NetworkURL#1382, custnum#1383, MarketCoverage#1384, RejectRows#1386, IssuerId#8312, IssuerId2#8325, issueridcomposite#8901, sysdt#9003, systs#9016, 'month('BuisnessDate) AS month#30398, 'year('BuisnessDate) AS year#30399]

+- SubqueryAlias insureview

+- View (`insureview`, [BusinessDate#1378,stcd#7759,srcnm#7747,NetworkName#1381,NetworkURL#1382,custnum#1383,MarketCoverage#1384,RejectRows#1386,IssuerId#8312,IssuerId2#8325,issueridcomposite#8901,sysdt#9003,systs#9016])

+- Project [BusinessDate#1378, stcd#7759, srcnm#7747, NetworkName#1381, NetworkURL#1382, custnum#1383, MarketCoverage#1384, RejectRows#1386, IssuerId#8312, IssuerId2#8325, issueridcomposite#8901, sysdt#9003, current\_timestamp() AS systs#9016]

+- Project [BusinessDate#1378, stcd#7759, srcnm#7747, NetworkName#1381, NetworkURL#1382, custnum#1383, MarketCoverage#1384, RejectRows#1386, IssuerId#8312, IssuerId2#8325, issueridcomposite#8901, current\_date(Some(Asia/Kolkata)) AS sysdt#9003]

+- Project [BusinessDate#1378, stcd#7759, srcnm#7747, NetworkName#1381, NetworkURL#1382, custnum#1383, MarketCoverage#1384, RejectRows#1386, IssuerId#8312, IssuerId2#8325, issueridcomposite#8901]

+- Project [BusinessDate#1378, stcd#7759, srcnm#7747, NetworkName#1381, NetworkURL#1382, custnum#1383, MarketCoverage#1384, DentalOnlyPlan#1385, RejectRows#1386, IssuerId#8312, IssuerId2#8325, concat(IssuerId#8312, IssuerId2#8325) AS issueridcomposite#8901]

+- Project [BusinessDate#1378, stcd#7759, srcnm#7747, NetworkName#1381, NetworkURL#1382, custnum#1383, MarketCoverage#1384, DentalOnlyPlan#1385, RejectRows#1386, IssuerId#8312, IssuerId2#8325]

+- Project [IssuerIDDD#8273, BusinessDate#1378, stcd#7759, srcnm#7747, NetworkName#1381, NetworkURL#1382, custnum#1383, MarketCoverage#1384, DentalOnlyPlan#1385, RejectRows#1386, IssuerId#8312, IssuerId2#8325]

+- Project [IssuerIDD#8261, IssuerIDDD#8273, BusinessDate#1378, stcd#7759, srcnm#7747, NetworkName#1381, NetworkURL#1382, custnum#1383, MarketCoverage#1384, DentalOnlyPlan#1385, RejectRows#1386, IssuerId#8312, cast(IssuerIDDD#8273 as string) AS IssuerId2#8325]

+- Project [IssuerIDD#8261, IssuerIDDD#8273, BusinessDate#1378, stcd#7759, srcnm#7747, NetworkName#1381, NetworkURL#1382, custnum#1383, MarketCoverage#1384, DentalOnlyPlan#1385, RejectRows#1386, cast(IssuerIDD#8261 as string) AS IssuerId#8312]

+- Project [IssuerIDD#8261, IssuerId2#1377 AS IssuerIDDD#8273, BusinessDate#1378, stcd#7759, srcnm#7747, NetworkName#1381, NetworkURL#1382, custnum#1383, MarketCoverage#1384, DentalOnlyPlan#1385, RejectRows#1386]

+- Project [IssuerId#1376 AS IssuerIDD#8261, IssuerId2#1377, BusinessDate#1378, stcd#7759, srcnm#7747, NetworkName#1381, NetworkURL#1382, custnum#1383, MarketCoverage#1384, DentalOnlyPlan#1385, RejectRows#1386]

+- Project [IssuerId#1376, IssuerId2#1377, BusinessDate#1378, StateCode#1379 AS stcd#7759, srcnm#7747, NetworkName#1381, NetworkURL#1382, custnum#1383, MarketCoverage#1384, DentalOnlyPlan#1385, RejectRows#1386]

+- Project [IssuerId#1376, IssuerId2#1377, BusinessDate#1378, StateCode#1379, SourceName#1380 AS srcnm#7747, NetworkName#1381, NetworkURL#1382, custnum#1383, MarketCoverage#1384, DentalOnlyPlan#1385, RejectRows#1386]

+- Union false, false

:- Relation [IssuerId#1376,IssuerId2#1377,BusinessDate#1378,StateCode#1379,SourceName#1380,NetworkName#1381,NetworkURL#1382,custnum#1383,MarketCoverage#1384,DentalOnlyPlan#1385,RejectRows#1386] csv

+- Relation [IssuerId#491,IssuerId2#492,BusinessDate#493,StateCode#494,SourceName#495,NetworkName#496,NetworkURL#497,custnum#498,MarketCoverage#499,DentalOnlyPlan#500,RejectRows#501] csv

>>> spark.sql("""select \*,month(BusinessDate) as month ,year(BusinessDate) as year from insureview""").show(3)

+------------+----+-----+------------+--------------------+-------+--------------+--------------------+--------+---------+-----------------+----------+--------------------+-----+----+

|BusinessDate|stcd|srcnm| NetworkName| NetworkURL|custnum|MarketCoverage| RejectRows|IssuerId|IssuerId2|issueridcomposite| sysdt| systs|month|year|

+------------+----+-----+------------+--------------------+-------+--------------+--------------------+--------+---------+-----------------+----------+--------------------+-----+----+

| 2019-10-01| AK| HIOS| ODS Premier|https://www.modah...| 13| null| null| 21989| 21989| 2198921989|2024-07-22|2024-07-22 11:57:...| 10|2019|

| 2019-10-01| AK| HIOS| ODS Premier| null| null| null|219,219,2019-10-0...| 219| 219| 219219|2024-07-22|2024-07-22 11:57:...| 10|2019|

| 2019-10-01| AK| HIOS|HeritagePlus|https://www.preme...| 13| null| null| 38344| 38344| 3834438344|2024-07-22|2024-07-22 11:57:...| 10|2019|

+------------+----+-----+------------+--------------------+-------+--------------+--------------------+--------+---------+-----------------+----------+--------------------+-----+----+

only showing top 3 rows

>>> result\_df\_SQL=spark.sql("""select \*,month(BusinessDate) as month ,year(BusinessDate) as year from insureview""")

>>> result\_df\_SQL\_1=spark.sql("""select \*,case when NetworkURL like 'https://%' then 'http secured' when NetworkURL like 'http://%' then 'http non secured' else 'No Protocol' as Protocol\_Used from insureview""")

Traceback (most recent call last):

File "<stdin>", line 1, in <module>

File "/usr/local/spark/python/pyspark/sql/session.py", line 723, in sql

return DataFrame(self.\_jsparkSession.sql(sqlQuery), self.\_wrapped)

File "/usr/local/spark/python/lib/py4j-0.10.9.3-src.zip/py4j/java\_gateway.py", line 1322, in \_\_call\_\_

File "/usr/local/spark/python/pyspark/sql/utils.py", line 117, in deco

raise converted from None

pyspark.sql.utils.ParseException:

mismatched input 'NetworkURL' expecting {<EOF>, ';'}(line 1, pos 19)

== SQL ==

select \*,case when NetworkURL like 'https://%' then 'http secured' when NetworkURL like 'http://%' then 'http non secured' else 'No Protocol' as Protocol\_Used from insureview

-------------------^^^

>>> result\_df\_SQL\_1=spark.sql("""select \*,case when NetworkURL like 'https://%' then 'http secured' when NetworkURL like 'http://%' then 'http non secured' else 'No Protocol' end as Protocol\_Used from insureview""")

>>> spark.sql("select \*,current\_date as curdt , current\_timestamp as curts from insureview").show(3)

+------------+----+-----+------------+--------------------+-------+--------------+--------------------+--------+---------+-----------------+----------+--------------------+----------+--------------------+

|BusinessDate|stcd|srcnm| NetworkName| NetworkURL|custnum|MarketCoverage| RejectRows|IssuerId|IssuerId2|issueridcomposite| sysdt| systs| curdt| curts|

+------------+----+-----+------------+--------------------+-------+--------------+--------------------+--------+---------+-----------------+----------+--------------------+----------+--------------------+

| 2019-10-01| AK| HIOS| ODS Premier|https://www.modah...| 13| null| null| 21989| 21989| 2198921989|2024-07-22|2024-07-22 12:00:...|2024-07-22|2024-07-22 12:00:...|

| 2019-10-01| AK| HIOS| ODS Premier| null| null| null|219,219,2019-10-0...| 219| 219| 219219|2024-07-22|2024-07-22 12:00:...|2024-07-22|2024-07-22 12:00:...|

| 2019-10-01| AK| HIOS|HeritagePlus|https://www.preme...| 13| null| null| 38344| 38344| 3834438344|2024-07-22|2024-07-22 12:00:...|2024-07-22|2024-07-22 12:00:...|

+------------+----+-----+------------+--------------------+-------+--------------+--------------------+--------+---------+-----------------+----------+--------------------+----------+--------------------+

only showing top 3 rows

>>> result\_df\_SQL\_2=spark.sql("select \*,current\_date as curdt , current\_timestamp as curts from insureview")

>>> spark.sql("select \*,sql\_function(NetworkName) as cleannetworkname from insureview").show(3)

+------------+----+-----+------------+--------------------+-------+--------------+--------------------+--------+---------+-----------------+----------+--------------------+----------------+

|BusinessDate|stcd|srcnm| NetworkName| NetworkURL|custnum|MarketCoverage| RejectRows|IssuerId|IssuerId2|issueridcomposite| sysdt| systs|cleannetworkname|

+------------+----+-----+------------+--------------------+-------+--------------+--------------------+--------+---------+-----------------+----------+--------------------+----------------+

| 2019-10-01| AK| HIOS| ODS Premier|https://www.modah...| 13| null| null| 21989| 21989| 2198921989|2024-07-22|2024-07-22 12:00:...| ODS Premier|

| 2019-10-01| AK| HIOS| ODS Premier| null| null| null|219,219,2019-10-0...| 219| 219| 219219|2024-07-22|2024-07-22 12:00:...| ODS Premier|

| 2019-10-01| AK| HIOS|HeritagePlus|https://www.preme...| 13| null| null| 38344| 38344| 3834438344|2024-07-22|2024-07-22 12:00:...| HeritagePlus|

+------------+----+-----+------------+--------------------+-------+--------------+--------------------+--------+---------+-----------------+----------+--------------------+----------------+

only showing top 3 rows

>>> result\_df\_SQL\_3=spark.sql("select \*,sql\_function(NetworkName) as cleannetworkname from insureview")

>>> result\_df\_SQL\_after\_join=result\_df\_SQL.alias("l").join(result\_df\_SQL\_1.alias("r") , on=[l.IssuerId=r.IssuerId],how="inner")

File "<stdin>", line 1

result\_df\_SQL\_after\_join=result\_df\_SQL.alias("l").join(result\_df\_SQL\_1.alias("r") , on=[l.IssuerId=r.IssuerId],how="inner")

^

SyntaxError: invalid syntax

>>> result\_df\_SQL\_after\_join=result\_df\_SQL.alias("l").join(result\_df\_SQL\_1.alias("r") , on=[l.IssuerId=r.IssuerId],how="inner")

File "<stdin>", line 1

result\_df\_SQL\_after\_join=result\_df\_SQL.alias("l").join(result\_df\_SQL\_1.alias("r") , on=[l.IssuerId=r.IssuerId],how="inner")

^

SyntaxError: invalid syntax

>>> result\_df\_SQL\_after\_join = result\_df\_SQL.alias("l").join(result\_df\_SQL\_1.alias("r"), on=col("l.IssuerId") == col("r.IssuerId"), how="inner")

>>>

>>> result\_df\_SQL\_after\_join = result\_df\_SQL\_after\_join.alias("l").join(result\_df\_SQL\_2.alias("r"), on=col("l.IssuerId") == col("r.IssuerId"), how="inner")

Traceback (most recent call last):

File "<stdin>", line 1, in <module>

File "/usr/local/spark/python/pyspark/sql/dataframe.py", line 1355, in join

jdf = self.\_jdf.join(other.\_jdf, on, how)

File "/usr/local/spark/python/lib/py4j-0.10.9.3-src.zip/py4j/java\_gateway.py", line 1322, in \_\_call\_\_

File "/usr/local/spark/python/pyspark/sql/utils.py", line 117, in deco

raise converted from None

pyspark.sql.utils.AnalysisException: Reference 'l.IssuerId' is ambiguous, could be: l.IssuerId, l.IssuerId.

>>> result\_df\_SQL\_after\_join = result\_df\_SQL\_after\_join.alias("l").join(result\_df\_SQL\_2.alias("r"), on=[col("l.IssuerId") == col("r.IssuerId")], how="inner")

Traceback (most recent call last):

File "<stdin>", line 1, in <module>

File "/usr/local/spark/python/pyspark/sql/dataframe.py", line 1355, in join

jdf = self.\_jdf.join(other.\_jdf, on, how)

File "/usr/local/spark/python/lib/py4j-0.10.9.3-src.zip/py4j/java\_gateway.py", line 1322, in \_\_call\_\_

File "/usr/local/spark/python/pyspark/sql/utils.py", line 117, in deco

raise converted from None

pyspark.sql.utils.AnalysisException: Reference 'l.IssuerId' is ambiguous, could be: l.IssuerId, l.IssuerId.

>>> result\_df\_SQL\_after\_join=result\_df\_SQL.alias("l").join(result\_df\_SQL\_1.alias("r") , on=[l.IssuerId=r.IssuerId],how="inner")

>>> result\_df\_SQL\_after\_join=result\_df\_SQL.alias("l").join(result\_df\_SQL\_1.alias("r") , on=[l.IssuerId=r.IssuerId],how="inner")

File "<stdin>", line 1

result\_df\_SQL\_after\_join=result\_df\_SQL.alias("l").join(result\_df\_SQL\_1.alias("r") , on=[l.IssuerId=r.IssuerId],how="inner")

^

SyntaxError: invalid syntax

>>> result\_df\_SQL\_after\_join = result\_df\_SQL\_after\_join.alias("l").join(result\_df\_SQL\_2.alias("r"), on=[col("l.IssuerId") == col("r.IssuerId")], how="inner")

Traceback (most recent call last):

File "<stdin>", line 1, in <module>

File "/usr/local/spark/python/pyspark/sql/dataframe.py", line 1355, in join

jdf = self.\_jdf.join(other.\_jdf, on, how)

File "/usr/local/spark/python/lib/py4j-0.10.9.3-src.zip/py4j/java\_gateway.py", line 1322, in \_\_call\_\_

File "/usr/local/spark/python/pyspark/sql/utils.py", line 117, in deco

raise converted from None

pyspark.sql.utils.AnalysisException: Reference 'l.IssuerId' is ambiguous, could be: l.IssuerId, l.IssuerId.

>>> result\_df\_SQL\_after\_join = result\_df\_SQL.alias("l").join(result\_df\_SQL\_1.alias("r"), on=col("l.IssuerId") == col("r.IssuerId"), how="inner")

>>>

>>> result\_df\_SQL\_after\_join = result\_df\_SQL.alias("l").join(result\_df\_SQL\_1.alias("r"), on=col("l.IssuerId") == col("r.IssuerId"), how="inner")

>>> result\_df\_SQL\_after\_join = result\_df\_SQL\_after\_join.join(result\_df\_SQL\_2, on=[col("IssuerId") == col("IssuerId")], how="inner")

24/07/22 12:08:43 WARN sql.Column: Constructing trivially true equals predicate, ''IssuerId = 'IssuerId'. Perhaps you need to use aliases.

Traceback (most recent call last):

File "<stdin>", line 1, in <module>

File "/usr/local/spark/python/pyspark/sql/dataframe.py", line 1355, in join

jdf = self.\_jdf.join(other.\_jdf, on, how)

File "/usr/local/spark/python/lib/py4j-0.10.9.3-src.zip/py4j/java\_gateway.py", line 1322, in \_\_call\_\_

File "/usr/local/spark/python/pyspark/sql/utils.py", line 117, in deco

raise converted from None

pyspark.sql.utils.AnalysisException: Reference 'IssuerId' is ambiguous, could be: l.IssuerId, r.IssuerId, insureview.IssuerId.

>>> result\_df\_SQL\_after\_join = result\_df\_SQL\_after\_join.alias("l").join(result\_df\_SQL\_2.alias("r"), on=[col("l.IssuerId") == col("r.IssuerId")], how="inner")

Traceback (most recent call last):

File "<stdin>", line 1, in <module>

File "/usr/local/spark/python/pyspark/sql/dataframe.py", line 1355, in join

jdf = self.\_jdf.join(other.\_jdf, on, how)

File "/usr/local/spark/python/lib/py4j-0.10.9.3-src.zip/py4j/java\_gateway.py", line 1322, in \_\_call\_\_

File "/usr/local/spark/python/pyspark/sql/utils.py", line 117, in deco

raise converted from None

pyspark.sql.utils.AnalysisException: Reference 'l.IssuerId' is ambiguous, could be: l.IssuerId, l.IssuerId.

>>> result\_df\_SQL\_after\_join = result\_df\_SQL\_after\_join.join(result\_df\_SQL\_2, on=[col("IssuerId") == col("IssuerId")], how="inner")

24/07/22 12:09:44 WARN sql.Column: Constructing trivially true equals predicate, ''IssuerId = 'IssuerId'. Perhaps you need to use aliases.

Traceback (most recent call last):

File "<stdin>", line 1, in <module>

File "/usr/local/spark/python/pyspark/sql/dataframe.py", line 1355, in join

jdf = self.\_jdf.join(other.\_jdf, on, how)

File "/usr/local/spark/python/lib/py4j-0.10.9.3-src.zip/py4j/java\_gateway.py", line 1322, in \_\_call\_\_

File "/usr/local/spark/python/pyspark/sql/utils.py", line 117, in deco

raise converted from None

pyspark.sql.utils.AnalysisException: Reference 'IssuerId' is ambiguous, could be: l.IssuerId, r.IssuerId, insureview.IssuerId.

>>>

>>> result\_df\_SQL\_after\_join = result\_df\_SQL.alias("l").join(result\_df\_SQL\_1.alias("r"), on=col("l.IssuerId") == col("r.IssuerId"), how="inner")

>>> result\_df\_SQL\_after\_join = result\_df\_SQL\_after\_join.alias("l").join(

... result\_df\_SQL\_2.alias("r"),

... col("l.IssuerId") == col("r.IssuerId"),

... how="inner"

... )

Traceback (most recent call last):

File "<stdin>", line 4, in <module>

File "/usr/local/spark/python/pyspark/sql/dataframe.py", line 1355, in join

jdf = self.\_jdf.join(other.\_jdf, on, how)

File "/usr/local/spark/python/lib/py4j-0.10.9.3-src.zip/py4j/java\_gateway.py", line 1322, in \_\_call\_\_

File "/usr/local/spark/python/pyspark/sql/utils.py", line 117, in deco

raise converted from None

pyspark.sql.utils.AnalysisException: Reference 'l.IssuerId' is ambiguous, could be: l.IssuerId, l.IssuerId.

>>> result\_df\_SQL.show(3)

+------------+----+-----+------------+--------------------+-------+--------------+--------------------+--------+---------+-----------------+----------+--------------------+-----+----+

|BusinessDate|stcd|srcnm| NetworkName| NetworkURL|custnum|MarketCoverage| RejectRows|IssuerId|IssuerId2|issueridcomposite| sysdt| systs|month|year|

+------------+----+-----+------------+--------------------+-------+--------------+--------------------+--------+---------+-----------------+----------+--------------------+-----+----+

| 2019-10-01| AK| HIOS| ODS Premier|https://www.modah...| 13| null| null| 21989| 21989| 2198921989|2024-07-22|2024-07-22 12:16:...| 10|2019|

| 2019-10-01| AK| HIOS| ODS Premier| null| null| null|219,219,2019-10-0...| 219| 219| 219219|2024-07-22|2024-07-22 12:16:...| 10|2019|

| 2019-10-01| AK| HIOS|HeritagePlus|https://www.preme...| 13| null| null| 38344| 38344| 3834438344|2024-07-22|2024-07-22 12:16:...| 10|2019|

+------------+----+-----+------------+--------------------+-------+--------------+--------------------+--------+---------+-----------------+----------+--------------------+-----+----+

only showing top 3 rows

>>> result\_df\_SQL.show(2)

>>> result\_df\_SQL.show(2)

+------------+----+-----+-----------+--------------------+-------+--------------+--------------------+--------+---------+-----------------+----------+--------------------+-----+----+

|BusinessDate|stcd|srcnm|NetworkName| NetworkURL|custnum|MarketCoverage| RejectRows|IssuerId|IssuerId2|issueridcomposite| sysdt| systs|month|year|

+------------+----+-----+-----------+--------------------+-------+--------------+--------------------+--------+---------+-----------------+----------+--------------------+-----+----+

| 2019-10-01| AK| HIOS|ODS Premier|https://www.modah...| 13| null| null| 21989| 21989| 2198921989|2024-07-22|2024-07-22 12:16:...| 10|2019|

| 2019-10-01| AK| HIOS|ODS Premier| null| null| null|219,219,2019-10-0...| 219| 219| 219219|2024-07-22|2024-07-22 12:16:...| 10|2019|

+------------+----+-----+-----------+--------------------+-------+--------------+--------------------+--------+---------+-----------------+----------+--------------------+-----+----+

only showing top 2 rows

>>> result\_df\_SQL\_1.show(2)

+------------+----+-----+-----------+--------------------+-------+--------------+--------------------+--------+---------+-----------------+----------+--------------------+-------------+

|BusinessDate|stcd|srcnm|NetworkName| NetworkURL|custnum|MarketCoverage| RejectRows|IssuerId|IssuerId2|issueridcomposite| sysdt| systs|Protocol\_Used|

+------------+----+-----+-----------+--------------------+-------+--------------+--------------------+--------+---------+-----------------+----------+--------------------+-------------+

| 2019-10-01| AK| HIOS|ODS Premier|https://www.modah...| 13| null| null| 21989| 21989| 2198921989|2024-07-22|2024-07-22 12:16:...| http secured|

| 2019-10-01| AK| HIOS|ODS Premier| null| null| null|219,219,2019-10-0...| 219| 219| 219219|2024-07-22|2024-07-22 12:16:...| No Protocol|

+------------+----+-----+-----------+--------------------+-------+--------------+--------------------+--------+---------+-----------------+----------+--------------------+-------------+

only showing top 2 rows

>>> result\_df\_SQL\_2.show(2)

+------------+----+-----+-----------+--------------------+-------+--------------+--------------------+--------+---------+-----------------+----------+--------------------+----------+--------------------+

|BusinessDate|stcd|srcnm|NetworkName| NetworkURL|custnum|MarketCoverage| RejectRows|IssuerId|IssuerId2|issueridcomposite| sysdt| systs| curdt| curts|

+------------+----+-----+-----------+--------------------+-------+--------------+--------------------+--------+---------+-----------------+----------+--------------------+----------+--------------------+

| 2019-10-01| AK| HIOS|ODS Premier|https://www.modah...| 13| null| null| 21989| 21989| 2198921989|2024-07-22|2024-07-22 12:16:...|2024-07-22|2024-07-22 12:16:...|

| 2019-10-01| AK| HIOS|ODS Premier| null| null| null|219,219,2019-10-0...| 219| 219| 219219|2024-07-22|2024-07-22 12:16:...|2024-07-22|2024-07-22 12:16:...|

+------------+----+-----+-----------+--------------------+-------+--------------+--------------------+--------+---------+-----------------+----------+--------------------+----------+--------------------+

only showing top 2 rows

>>> result\_df\_SQL\_3.show(2)

+------------+----+-----+-----------+--------------------+-------+--------------+--------------------+--------+---------+-----------------+----------+--------------------+----------------+

|BusinessDate|stcd|srcnm|NetworkName| NetworkURL|custnum|MarketCoverage| RejectRows|IssuerId|IssuerId2|issueridcomposite| sysdt| systs|cleannetworkname|

+------------+----+-----+-----------+--------------------+-------+--------------+--------------------+--------+---------+-----------------+----------+--------------------+----------------+

| 2019-10-01| AK| HIOS|ODS Premier|https://www.modah...| 13| null| null| 21989| 21989| 2198921989|2024-07-22|2024-07-22 12:16:...| ODS Premier|

| 2019-10-01| AK| HIOS|ODS Premier| null| null| null|219,219,2019-10-0...| 219| 219| 219219|2024-07-22|2024-07-22 12:16:...| ODS Premier|

+------------+----+-----+-----------+--------------------+-------+--------------+--------------------+--------+---------+-----------------+----------+--------------------+----------------+

only showing top 2 rows

>>> result\_df\_SQL.show(1)

>>> result\_df\_SQL.show(1)

+------------+----+-----+-----------+--------------------+-------+--------------+----------+--------+---------+-----------------+----------+--------------------+-----+----+

|BusinessDate|stcd|srcnm|NetworkName| NetworkURL|custnum|MarketCoverage|RejectRows|IssuerId|IssuerId2|issueridcomposite| sysdt| systs|month|year|

+------------+----+-----+-----------+--------------------+-------+--------------+----------+--------+---------+-----------------+----------+--------------------+-----+----+

| 2019-10-01| AK| HIOS|ODS Premier|https://www.modah...| 13| null| null| 21989| 21989| 2198921989|2024-07-22|2024-07-22 12:17:...| 10|2019|

+------------+----+-----+-----------+--------------------+-------+--------------+----------+--------+---------+-----------------+----------+--------------------+-----+----+

only showing top 1 row

>>> result\_df\_SQL\_1.show(1)

+------------+----+-----+-----------+--------------------+-------+--------------+----------+--------+---------+-----------------+----------+--------------------+-------------+

|BusinessDate|stcd|srcnm|NetworkName| NetworkURL|custnum|MarketCoverage|RejectRows|IssuerId|IssuerId2|issueridcomposite| sysdt| systs|Protocol\_Used|

+------------+----+-----+-----------+--------------------+-------+--------------+----------+--------+---------+-----------------+----------+--------------------+-------------+

| 2019-10-01| AK| HIOS|ODS Premier|https://www.modah...| 13| null| null| 21989| 21989| 2198921989|2024-07-22|2024-07-22 12:17:...| http secured|

+------------+----+-----+-----------+--------------------+-------+--------------+----------+--------+---------+-----------------+----------+--------------------+-------------+

only showing top 1 row

>>> result\_df\_SQL\_2.show(1)

+------------+----+-----+-----------+--------------------+-------+--------------+----------+--------+---------+-----------------+----------+--------------------+----------+--------------------+

|BusinessDate|stcd|srcnm|NetworkName| NetworkURL|custnum|MarketCoverage|RejectRows|IssuerId|IssuerId2|issueridcomposite| sysdt| systs| curdt| curts|

+------------+----+-----+-----------+--------------------+-------+--------------+----------+--------+---------+-----------------+----------+--------------------+----------+--------------------+

| 2019-10-01| AK| HIOS|ODS Premier|https://www.modah...| 13| null| null| 21989| 21989| 2198921989|2024-07-22|2024-07-22 12:17:...|2024-07-22|2024-07-22 12:17:...|

+------------+----+-----+-----------+--------------------+-------+--------------+----------+--------+---------+-----------------+----------+--------------------+----------+--------------------+

only showing top 1 row

>>> result\_df\_SQL\_3.show(1)

+------------+----+-----+-----------+--------------------+-------+--------------+----------+--------+---------+-----------------+----------+--------------------+----------------+

|BusinessDate|stcd|srcnm|NetworkName| NetworkURL|custnum|MarketCoverage|RejectRows|IssuerId|IssuerId2|issueridcomposite| sysdt| systs|cleannetworkname|

+------------+----+-----+-----------+--------------------+-------+--------------+----------+--------+---------+-----------------+----------+--------------------+----------------+

| 2019-10-01| AK| HIOS|ODS Premier|https://www.modah...| 13| null| null| 21989| 21989| 2198921989|2024-07-22|2024-07-22 12:20:...| ODS Premier|

+------------+----+-----+-----------+--------------------+-------+--------------+----------+--------+---------+-----------------+----------+--------------------+----------------+

only showing top 1 row

>>>

>>> df\_combined = result\_df\_SQL \

... .alias("a") \

... .join(result\_df\_SQL\_1.alias("b"), on=col("a.IssuerId") == col("b.IssuerId"), how="inner") \

... .join(result\_df\_SQL\_2.alias("c"), on=col("a.IssuerId") == col("c.IssuerId"), how="inner") \

... .join(result\_df\_SQL\_3.alias("d"), on=col("a.IssuerId") == col("d.IssuerId"), how="inner")

>>> df\_combined.show(1)

24/07/22 12:20:34 WARN util.package: Truncated the string representation of a plan since it was too large. This behavior can be adjusted by setting 'spark.sql.debug.maxToStringFields'.

+------------+----+-----+----------------+--------------------+-------+------------------+----------+--------+---------+-----------------+----------+--------------------+-----+----+------------+----+-----+-----------+--------------------+-------+--------------+----------+--------+---------+-----------------+----------+--------------------+-------------+------------+----+-----+----------------+--------------------+-------+------------------+----------+--------+---------+-----------------+----------+--------------------+----------+--------------------+------------+----+-----+----------------+--------------------+-------+------------------+----------+--------+---------+-----------------+----------+--------------------+----------------+

|BusinessDate|stcd|srcnm| NetworkName| NetworkURL|custnum| MarketCoverage|RejectRows|IssuerId|IssuerId2|issueridcomposite| sysdt| systs|month|year|BusinessDate|stcd|srcnm|NetworkName| NetworkURL|custnum|MarketCoverage|RejectRows|IssuerId|IssuerId2|issueridcomposite| sysdt| systs|Protocol\_Used|BusinessDate|stcd|srcnm| NetworkName| NetworkURL|custnum| MarketCoverage|RejectRows|IssuerId|IssuerId2|issueridcomposite| sysdt| systs| curdt| curts|BusinessDate|stcd|srcnm| NetworkName| NetworkURL|custnum| MarketCoverage|RejectRows|IssuerId|IssuerId2|issueridcomposite| sysdt| systs|cleannetworkname|

+------------+----+-----+----------------+--------------------+-------+------------------+----------+--------+---------+-----------------+----------+--------------------+-----+----+------------+----+-----+-----------+--------------------+-------+--------------+----------+--------+---------+-----------------+----------+--------------------+-------------+------------+----+-----+----------------+--------------------+-------+------------------+----------+--------+---------+-----------------+----------+--------------------+----------+--------------------+------------+----+-----+----------------+--------------------+-------+------------------+----------+--------+---------+-----------------+----------+--------------------+----------------+

| 2019-01-10| AK| HIOS|Delta Dental PPO|https://www.modah...| 17|SHOP (Small Group)| null| 21989| 21989| 2198921989|2024-07-22|2024-07-22 12:20:...| 1|2019| 2019-10-01| AK| HIOS|ODS Premier|https://www.modah...| 13| null| null| 21989| 21989| 2198921989|2024-07-22|2024-07-22 12:20:...| http secured| 2019-01-10| AK| HIOS|Delta Dental PPO|https://www.modah...| 17|SHOP (Small Group)| null| 21989| 21989| 2198921989|2024-07-22|2024-07-22 12:20:...|2024-07-22|2024-07-22 12:20:...| 2019-01-10| AK| HIOS|Delta Dental PPO|https://www.modah...| 17|SHOP (Small Group)| null| 21989| 21989| 2198921989|2024-07-22|2024-07-22 12:20:...|Delta Dental PPO|

+------------+----+-----+----------------+--------------------+-------+------------------+----------+--------+---------+-----------------+----------+--------------------+-----+----+------------+----+-----+-----------+--------------------+-------+--------------+----------+--------+---------+-----------------+----------+--------------------+-------------+------------+----+-----+----------------+--------------------+-------+------------------+----------+--------+---------+-----------------+----------+--------------------+----------+--------------------+------------+----+-----+----------------+--------------------+-------+------------------+----------+--------+---------+-----------------+----------+--------------------+----------------+

only showing top 1 row

>>> df\_with\_states = df\_combined \

... .alias("i") \

... .join(statesview.alias("s"), col("i.stcd") == col("s.stated"), how="inner")

Traceback (most recent call last):

File "<stdin>", line 3, in <module>

NameError: name 'statesview' is not defined

>>>

>>> # Joining result with custview

... df\_final = df\_with\_states \

... .alias("is") \

... .join(custview.alias("c"), col("is.custnum") == col("c.custid"), how="inner")

Traceback (most recent call last):

File "<stdin>", line 2, in <module>

NameError: name 'df\_with\_states' is not defined

>>> joined\_df\_state = df\_combined.alias("i").join(

... state\_filter\_df.alias("s"),

... col("i.stcd") == col("s.stated"),

... how="inner"

... )

Traceback (most recent call last):

File "<stdin>", line 2, in <module>

NameError: name 'state\_filter\_df' is not defined

>>>

>>> # Join the result with custfilter\_df

... final\_df = joined\_df\_state.alias("j").join(

... custfilter\_df.alias("c"),

... col("j.custnum") == col("c.custid"),

... how="inner"

... )

Traceback (most recent call last):

File "<stdin>", line 2, in <module>

NameError: name 'joined\_df\_state' is not defined

>>> )joined\_df\_state = df\_combined.alias("i").join(

File "<stdin>", line 1

)joined\_df\_state = df\_combined.alias("i").join(

^

SyntaxError: invalid syntax

>>> statefilter\_df.alias("s"),

File "<stdin>", line 1

statefilter\_df.alias("s"),

^

IndentationError: unexpected indent

>>> col("i.stcd") == col("s.stated"),

File "<stdin>", line 1

col("i.stcd") == col("s.stated"),

^

IndentationError: unexpected indent

>>> how="inner"

File "<stdin>", line 1

how="inner"

^

IndentationError: unexpected indent

>>> )

File "<stdin>", line 1

)

^

SyntaxError: invalid syntax

>>> joined\_df\_state = df\_combined.alias("i").join(

... statefilter\_df.alias("s"),

... col("i.stcd") == col("s.stated"),

... how="inner"

... )

Traceback (most recent call last):

File "<stdin>", line 4, in <module>

File "/usr/local/spark/python/pyspark/sql/dataframe.py", line 1355, in join

jdf = self.\_jdf.join(other.\_jdf, on, how)

File "/usr/local/spark/python/lib/py4j-0.10.9.3-src.zip/py4j/java\_gateway.py", line 1322, in \_\_call\_\_

File "/usr/local/spark/python/pyspark/sql/utils.py", line 117, in deco

raise converted from None

pyspark.sql.utils.AnalysisException: Reference 'i.stcd' is ambiguous, could be: i.stcd, i.stcd, i.stcd, i.stcd.

>>> joined\_df\_state = df\_combined.alias("i").join(

... state\_filter\_df.alias("s"),

... col("i.stcd") == col("s.stated"),

... how="inner"

... )

Traceback (most recent call last):

File "<stdin>", line 2, in <module>

NameError: name 'state\_filter\_df' is not defined

>>> joined\_df\_state = df\_combined.alias("i").join(

... statefilter\_df.alias("s"),

... col("i.stcd") == col("s.stated"),

... how="inner"

... )

Traceback (most recent call last):

File "<stdin>", line 4, in <module>

File "/usr/local/spark/python/pyspark/sql/dataframe.py", line 1355, in join

jdf = self.\_jdf.join(other.\_jdf, on, how)

File "/usr/local/spark/python/lib/py4j-0.10.9.3-src.zip/py4j/java\_gateway.py", line 1322, in \_\_call\_\_

File "/usr/local/spark/python/pyspark/sql/utils.py", line 117, in deco

raise converted from None

pyspark.sql.utils.AnalysisException: Reference 'i.stcd' is ambiguous, could be: i.stcd, i.stcd, i.stcd, i.stcd.

>>>

>>> df\_combined\_renamed = df\_combined.withColumnRenamed("stcd", "i\_stcd")

>>> state\_filter\_df\_renamed = statefilter\_df.withColumnRenamed("stated", "s\_stated")

>>>

>>> # Step 2: Perform the join with renamed columns

... joined\_df\_state = df\_combined\_renamed.alias("i").join(

... statefilter\_df\_renamed.alias("s"),

... col("i.i\_stcd") == col("s.s\_stated"),

... how="inner"

... )

Traceback (most recent call last):

File "<stdin>", line 3, in <module>

NameError: name 'statefilter\_df\_renamed' is not defined

>>>

>>> # Step 3: Rename columns in the result to remove ambiguity

... df\_combined\_state\_renamed = joined\_df\_state.withColumnRenamed("statedesc", "state\_desc")

Traceback (most recent call last):

File "<stdin>", line 2, in <module>

NameError: name 'joined\_df\_state' is not defined

>>>

>>> # Step 4: Join with custfilter\_df

... final\_df = df\_combined\_state\_renamed.alias("j").join(

... custfilter\_df.alias("c"),

... col("j.custnum") == col("c.custid"),

... how="inner"

... )

Traceback (most recent call last):

File "<stdin>", line 2, in <module>

NameError: name 'df\_combined\_state\_renamed' is not defined

>>> state\_filter\_df\_renamed.SHOW()

Traceback (most recent call last):

File "<stdin>", line 1, in <module>

File "/usr/local/spark/python/pyspark/sql/dataframe.py", line 1660, in \_\_getattr\_\_

"'%s' object has no attribute '%s'" % (self.\_\_class\_\_.\_\_name\_\_, name))

AttributeError: 'DataFrame' object has no attribute 'SHOW'

>>> state\_filter\_df\_renamed.show()

+---+--------------+

|\_c0| \_c1|

+---+--------------+

| NM| New Mexico|

| NY| New York|

| NC|North Carolina|

| ND| North Dakota|

| OH| Ohio|

| OK| Oklahoma|

| OR| Oregon|

| MD| Maryland|

| MA| Massachusetts|

| MI| Michigan|

| MN| Minnesota|

| MS| Mississippi|

| MO| Missouri|

| PA| Pennsylvania|

| RI| Rhode Island|

| SC|South Carolina|

| SD| South Dakota|

| TN| Tennessee|

| TX| Texas|

| UT| Utah|

+---+--------------+

only showing top 20 rows

>>> joined\_df\_state = df\_combined\_renamed.alias("i").join(

... statefilter\_df\_renamed.alias("s"),

... col("i.i\_stcd") == col("s.s\_stated"),

... how="inner"

... )

Traceback (most recent call last):

File "<stdin>", line 2, in <module>

NameError: name 'statefilter\_df\_renamed' is not defined

>>>

>>> col("i.i\_stcd") == col("s.s\_stated"),

>>> col("i.i\_stcd") == col("s.s\_stated"),

>>> col("i.i\_stcd") == col("s.s\_stated"),

File "<stdin>", line 1

col("i.i\_stcd") == col("s.s\_stated"),

^

IndentationError: unexpected indent

>>>

>>> df\_combined\_renamed = df\_combined.withColumnRenamed("stcd", "i\_stcd")

>>> state\_filter\_df\_renamed = statefilter\_df.withColumnRenamed("stated", "s\_stated")

>>>

>>> # Step 2: Perform the join with renamed columns

... joined\_df\_state = df\_combined\_renamed.alias("i").join(

... state\_filter\_df\_renamed.alias("s"),

... col("i.i\_stcd") == col("s.s\_stated"),

... how="inner"

... )

Traceback (most recent call last):

File "<stdin>", line 5, in <module>

File "/usr/local/spark/python/pyspark/sql/dataframe.py", line 1355, in join

jdf = self.\_jdf.join(other.\_jdf, on, how)

File "/usr/local/spark/python/lib/py4j-0.10.9.3-src.zip/py4j/java\_gateway.py", line 1322, in \_\_call\_\_

File "/usr/local/spark/python/pyspark/sql/utils.py", line 117, in deco

raise converted from None

pyspark.sql.utils.AnalysisException: Reference 'i.i\_stcd' is ambiguous, could be: i.i\_stcd, i.i\_stcd, i.i\_stcd, i.i\_stcd.

>>>

>>> # Step 3: Rename columns in the result to remove ambiguity

... df\_combined\_state\_renamed = joined\_df\_state.withColumnRenamed("statedesc", "state\_desc")

Traceback (most recent call last):

File "<stdin>", line 2, in <module>

NameError: name 'joined\_df\_state' is not defined

>>>

>>> # Step 4: Join with custfilter\_df

... final\_df = df\_combined\_state\_renamed.alias("j").join(

... custfilter\_df.alias("c"),

... col("j.custnum") == col("c.custid"),

... how="inner"

... )

Traceback (most recent call last):

File "<stdin>", line 2, in <module>

NameError: name 'df\_combined\_state\_renamed' is not defined

>>> df\_combined.show(1)

+------------+----+-----+----------------+--------------------+-------+------------------+----------+--------+---------+-----------------+----------+--------------------+-----+----+------------+----+-----+-----------+--------------------+-------+--------------+----------+--------+---------+-----------------+----------+--------------------+-------------+------------+----+-----+----------------+--------------------+-------+------------------+----------+--------+---------+-----------------+----------+--------------------+----------+--------------------+------------+----+-----+----------------+--------------------+-------+------------------+----------+--------+---------+-----------------+----------+--------------------+----------------+

|BusinessDate|stcd|srcnm| NetworkName| NetworkURL|custnum| MarketCoverage|RejectRows|IssuerId|IssuerId2|issueridcomposite| sysdt| systs|month|year|BusinessDate|stcd|srcnm|NetworkName| NetworkURL|custnum|MarketCoverage|RejectRows|IssuerId|IssuerId2|issueridcomposite| sysdt| systs|Protocol\_Used|BusinessDate|stcd|srcnm| NetworkName| NetworkURL|custnum| MarketCoverage|RejectRows|IssuerId|IssuerId2|issueridcomposite| sysdt| systs| curdt| curts|BusinessDate|stcd|srcnm| NetworkName| NetworkURL|custnum| MarketCoverage|RejectRows|IssuerId|IssuerId2|issueridcomposite| sysdt| systs|cleannetworkname|

+------------+----+-----+----------------+--------------------+-------+------------------+----------+--------+---------+-----------------+----------+--------------------+-----+----+------------+----+-----+-----------+--------------------+-------+--------------+----------+--------+---------+-----------------+----------+--------------------+-------------+------------+----+-----+----------------+--------------------+-------+------------------+----------+--------+---------+-----------------+----------+--------------------+----------+--------------------+------------+----+-----+----------------+--------------------+-------+------------------+----------+--------+---------+-----------------+----------+--------------------+----------------+

| 2019-01-10| AK| HIOS|Delta Dental PPO|https://www.modah...| 17|SHOP (Small Group)| null| 21989| 21989| 2198921989|2024-07-22|2024-07-22 12:31:...| 1|2019| 2019-10-01| AK| HIOS|ODS Premier|https://www.modah...| 13| null| null| 21989| 21989| 2198921989|2024-07-22|2024-07-22 12:31:...| http secured| 2019-01-10| AK| HIOS|Delta Dental PPO|https://www.modah...| 17|SHOP (Small Group)| null| 21989| 21989| 2198921989|2024-07-22|2024-07-22 12:31:...|2024-07-22|2024-07-22 12:31:...| 2019-01-10| AK| HIOS|Delta Dental PPO|https://www.modah...| 17|SHOP (Small Group)| null| 21989| 21989| 2198921989|2024-07-22|2024-07-22 12:31:...|Delta Dental PPO|

+------------+----+-----+----------------+--------------------+-------+------------------+----------+--------+---------+-----------------+----------+--------------------+-----+----+------------+----+-----+-----------+--------------------+-------+--------------+----------+--------+---------+-----------------+----------+--------------------+-------------+------------+----+-----+----------------+--------------------+-------+------------------+----------+--------+---------+-----------------+----------+--------------------+----------+--------------------+------------+----+-----+----------------+--------------------+-------+------------------+----------+--------+---------+-----------------+----------+--------------------+----------------+

only showing top 1 row

>>>

>>> df\_combined.write.parquet("hdfs:///user/hduser/Hackathon/Parquet")

Traceback (most recent call last):

File "<stdin>", line 1, in <module>

File "/usr/local/spark/python/pyspark/sql/readwriter.py", line 885, in parquet

self.\_jwrite.parquet(path)

File "/usr/local/spark/python/lib/py4j-0.10.9.3-src.zip/py4j/java\_gateway.py", line 1322, in \_\_call\_\_

File "/usr/local/spark/python/pyspark/sql/utils.py", line 117, in deco

raise converted from None

pyspark.sql.utils.AnalysisException: Found duplicate column(s) when inserting into hdfs://localhost:54310/user/hduser/Hackathon/Parquet: `businessdate`, `custnum`, `issuerid2`, `issuerid`, `issueridcomposite`, `marketcoverage`, `networkname`, `networkurl`, `rejectrows`, `srcnm`, `stcd`, `sysdt`, `systs`

>>> df\_combined.describe

<bound method DataFrame.describe of DataFrame[BusinessDate: date, stcd: string, srcnm: string, NetworkName: string, NetworkURL: string, custnum: string, MarketCoverage: string, RejectRows: string, IssuerId: string, IssuerId2: string, issueridcomposite: string, sysdt: date, systs: timestamp, month: int, year: int, BusinessDate: date, stcd: string, srcnm: string, NetworkName: string, NetworkURL: string, custnum: string, MarketCoverage: string, RejectRows: string, IssuerId: string, IssuerId2: string, issueridcomposite: string, sysdt: date, systs: timestamp, Protocol\_Used: string, BusinessDate: date, stcd: string, srcnm: string, NetworkName: string, NetworkURL: string, custnum: string, MarketCoverage: string, RejectRows: string, IssuerId: string, IssuerId2: string, issueridcomposite: string, sysdt: date, systs: timestamp, curdt: date, curts: timestamp, BusinessDate: date, stcd: string, srcnm: string, NetworkName: string, NetworkURL: string, custnum: string, MarketCoverage: string, RejectRows: string, IssuerId: string, IssuerId2: string, issueridcomposite: string, sysdt: date, systs: timestamp, cleannetworkname: string]>

>>> df\_combined.describe

>>> list\_of\_col=df\_combined.columns

>>> print(list\_of\_col)

['BusinessDate', 'stcd', 'srcnm', 'NetworkName', 'NetworkURL', 'custnum', 'MarketCoverage', 'RejectRows', 'IssuerId', 'IssuerId2', 'issueridcomposite', 'sysdt', 'systs', 'month', 'year', 'BusinessDate', 'stcd', 'srcnm', 'NetworkName', 'NetworkURL', 'custnum', 'MarketCoverage', 'RejectRows', 'IssuerId', 'IssuerId2', 'issueridcomposite', 'sysdt', 'systs', 'Protocol\_Used', 'BusinessDate', 'stcd', 'srcnm', 'NetworkName', 'NetworkURL', 'custnum', 'MarketCoverage', 'RejectRows', 'IssuerId', 'IssuerId2', 'issueridcomposite', 'sysdt', 'systs', 'curdt', 'curts', 'BusinessDate', 'stcd', 'srcnm', 'NetworkName', 'NetworkURL', 'custnum', 'MarketCoverage', 'RejectRows', 'IssuerId', 'IssuerId2', 'issueridcomposite', 'sysdt', 'systs', 'cleannetworkname']

>>> tuple\_list=tuple(list\_of\_col)

>>> print(tuple\_list)

('BusinessDate', 'stcd', 'srcnm', 'NetworkName', 'NetworkURL', 'custnum', 'MarketCoverage', 'RejectRows', 'IssuerId', 'IssuerId2', 'issueridcomposite', 'sysdt', 'systs', 'month', 'year', 'BusinessDate', 'stcd', 'srcnm', 'NetworkName', 'NetworkURL', 'custnum', 'MarketCoverage', 'RejectRows', 'IssuerId', 'IssuerId2', 'issueridcomposite', 'sysdt', 'systs', 'Protocol\_Used', 'BusinessDate', 'stcd', 'srcnm', 'NetworkName', 'NetworkURL', 'custnum', 'MarketCoverage', 'RejectRows', 'IssuerId', 'IssuerId2', 'issueridcomposite', 'sysdt', 'systs', 'curdt', 'curts', 'BusinessDate', 'stcd', 'srcnm', 'NetworkName', 'NetworkURL', 'custnum', 'MarketCoverage', 'RejectRows', 'IssuerId', 'IssuerId2', 'issueridcomposite', 'sysdt', 'systs', 'cleannetworkname')

>>> set\_list=set(list\_of\_col)

>>> print(set\_list)

{'NetworkURL', 'year', 'issueridcomposite', 'NetworkName', 'RejectRows', 'IssuerId', 'stcd', 'IssuerId2', 'systs', 'sysdt', 'BusinessDate', 'MarketCoverage', 'Protocol\_Used', 'curts', 'curdt', 'month', 'cleannetworkname', 'srcnm', 'custnum'}

>>> list\_col\_no\_duplicate=list(set\_list)

>>> df\_combined.select(list\_col\_no\_duplicate).show(1)

Traceback (most recent call last):

File "<stdin>", line 1, in <module>

File "/usr/local/spark/python/pyspark/sql/dataframe.py", line 1685, in select

jdf = self.\_jdf.select(self.\_jcols(\*cols))

File "/usr/local/spark/python/lib/py4j-0.10.9.3-src.zip/py4j/java\_gateway.py", line 1322, in \_\_call\_\_

File "/usr/local/spark/python/pyspark/sql/utils.py", line 117, in deco

raise converted from None

pyspark.sql.utils.AnalysisException: Reference 'NetworkURL' is ambiguous, could be: a.NetworkURL, b.NetworkURL, c.NetworkURL, d.NetworkURL.

>>> df\_combined.select(list\_col\_no\_duplicate).show(1)

Traceback (most recent call last):

File "<stdin>", line 1, in <module>

File "/usr/local/spark/python/pyspark/sql/dataframe.py", line 1685, in select

jdf = self.\_jdf.select(self.\_jcols(\*cols))

File "/usr/local/spark/python/lib/py4j-0.10.9.3-src.zip/py4j/java\_gateway.py", line 1322, in \_\_call\_\_

File "/usr/local/spark/python/pyspark/sql/utils.py", line 117, in deco

raise converted from None

pyspark.sql.utils.AnalysisException: Reference 'NetworkURL' is ambiguous, could be: a.NetworkURL, b.NetworkURL, c.NetworkURL, d.NetworkURL.

>>>

>>> # Rename columns in each DataFrame before joining

... result\_df\_SQL = result\_df\_SQL.withColumnRenamed("NetworkURL", "a\_NetworkURL") \

... .withColumnRenamed("NetworkName", "a\_NetworkName") \

... .withColumnRenamed("IssuerId", "a\_IssuerId") \

... .withColumnRenamed("BusinessDate", "a\_BusinessDate")

>>>

>>> result\_df\_SQL\_1 = result\_df\_SQL\_1.withColumnRenamed("NetworkURL", "b\_NetworkURL") \

... .withColumnRenamed("NetworkName", "b\_NetworkName") \

... .withColumnRenamed("IssuerId", "b\_IssuerId") \

... .withColumnRenamed("BusinessDate", "b\_BusinessDate")

>>>

>>> result\_df\_SQL\_2 = result\_df\_SQL\_2.withColumnRenamed("NetworkURL", "c\_NetworkURL") \

... .withColumnRenamed("NetworkName", "c\_NetworkName") \

... .withColumnRenamed("IssuerId", "c\_IssuerId") \

... .withColumnRenamed("BusinessDate", "c\_BusinessDate")

>>>

>>> result\_df\_SQL\_3 = result\_df\_SQL\_3.withColumnRenamed("NetworkURL", "d\_NetworkURL") \

... .withColumnRenamed("NetworkName", "d\_NetworkName") \

... .withColumnRenamed("IssuerId", "d\_IssuerId") \

... .withColumnRenamed("BusinessDate", "d\_BusinessDate")

>>>

>>> # Perform the join

... df\_combined = result\_df\_SQL \

... .alias("a") \

... .join(result\_df\_SQL\_1.alias("b"), col("a.IssuerId") == col("b.IssuerId"), how="inner") \

... .join(result\_df\_SQL\_2.alias("c"), col("a.IssuerId") == col("c.IssuerId"), how="inner") \

... .join(result\_df\_SQL\_3.alias("d"), col("a.IssuerId") == col("d.IssuerId"), how="inner")

Traceback (most recent call last):

File "<stdin>", line 4, in <module>

File "/usr/local/spark/python/pyspark/sql/dataframe.py", line 1355, in join

jdf = self.\_jdf.join(other.\_jdf, on, how)

File "/usr/local/spark/python/lib/py4j-0.10.9.3-src.zip/py4j/java\_gateway.py", line 1322, in \_\_call\_\_

File "/usr/local/spark/python/pyspark/sql/utils.py", line 117, in deco

raise converted from None

pyspark.sql.utils.AnalysisException: cannot resolve 'a.IssuerId' given input columns: [a.IssuerId2, b.IssuerId2, a.MarketCoverage, b.MarketCoverage, b.Protocol\_Used, a.RejectRows, b.RejectRows, a.a\_BusinessDate, a.a\_IssuerId, a.a\_NetworkName, a.a\_NetworkURL, b.b\_BusinessDate, b.b\_IssuerId, b.b\_NetworkName, b.b\_NetworkURL, a.custnum, b.custnum, a.issueridcomposite, b.issueridcomposite, a.month, a.srcnm, b.srcnm, a.stcd, b.stcd, a.sysdt, b.sysdt, a.systs, b.systs, a.year];

'Join Inner, ('a.IssuerId = 'b.IssuerId)

:- SubqueryAlias a

: +- Project [BusinessDate#1378 AS a\_BusinessDate#43854, stcd#7759, srcnm#7747, a\_NetworkName#43822, a\_NetworkURL#43806, custnum#1383, MarketCoverage#1384, RejectRows#1386, a\_IssuerId#43838, IssuerId2#8325, issueridcomposite#8901, sysdt#9003, systs#9016, month#30928, year#30929]

: +- Project [BusinessDate#1378, stcd#7759, srcnm#7747, a\_NetworkName#43822, a\_NetworkURL#43806, custnum#1383, MarketCoverage#1384, RejectRows#1386, IssuerId#8312 AS a\_IssuerId#43838, IssuerId2#8325, issueridcomposite#8901, sysdt#9003, systs#9016, month#30928, year#30929]

: +- Project [BusinessDate#1378, stcd#7759, srcnm#7747, NetworkName#1381 AS a\_NetworkName#43822, a\_NetworkURL#43806, custnum#1383, MarketCoverage#1384, RejectRows#1386, IssuerId#8312, IssuerId2#8325, issueridcomposite#8901, sysdt#9003, systs#9016, month#30928, year#30929]

: +- Project [BusinessDate#1378, stcd#7759, srcnm#7747, NetworkName#1381, NetworkURL#1382 AS a\_NetworkURL#43806, custnum#1383, MarketCoverage#1384, RejectRows#1386, IssuerId#8312, IssuerId2#8325, issueridcomposite#8901, sysdt#9003, systs#9016, month#30928, year#30929]

: +- Project [BusinessDate#1378, stcd#7759, srcnm#7747, NetworkName#1381, NetworkURL#1382, custnum#1383, MarketCoverage#1384, RejectRows#1386, IssuerId#8312, IssuerId2#8325, issueridcomposite#8901, sysdt#9003, systs#9016, month(BusinessDate#1378) AS month#30928, year(BusinessDate#1378) AS year#30929]

: +- SubqueryAlias insureview

: +- View (`insureview`, [BusinessDate#1378,stcd#7759,srcnm#7747,NetworkName#1381,NetworkURL#1382,custnum#1383,MarketCoverage#1384,RejectRows#1386,IssuerId#8312,IssuerId2#8325,issueridcomposite#8901,sysdt#9003,systs#9016])

: +- Project [BusinessDate#1378, stcd#7759, srcnm#7747, NetworkName#1381, NetworkURL#1382, custnum#1383, MarketCoverage#1384, RejectRows#1386, IssuerId#8312, IssuerId2#8325, issueridcomposite#8901, sysdt#9003, current\_timestamp() AS systs#9016]

: +- Project [BusinessDate#1378, stcd#7759, srcnm#7747, NetworkName#1381, NetworkURL#1382, custnum#1383, MarketCoverage#1384, RejectRows#1386, IssuerId#8312, IssuerId2#8325, issueridcomposite#8901, current\_date(Some(Asia/Kolkata)) AS sysdt#9003]

: +- Project [BusinessDate#1378, stcd#7759, srcnm#7747, NetworkName#1381, NetworkURL#1382, custnum#1383, MarketCoverage#1384, RejectRows#1386, IssuerId#8312, IssuerId2#8325, issueridcomposite#8901]

: +- Project [BusinessDate#1378, stcd#7759, srcnm#7747, NetworkName#1381, NetworkURL#1382, custnum#1383, MarketCoverage#1384, DentalOnlyPlan#1385, RejectRows#1386, IssuerId#8312, IssuerId2#8325, concat(IssuerId#8312, IssuerId2#8325) AS issueridcomposite#8901]

: +- Project [BusinessDate#1378, stcd#7759, srcnm#7747, NetworkName#1381, NetworkURL#1382, custnum#1383, MarketCoverage#1384, DentalOnlyPlan#1385, RejectRows#1386, IssuerId#8312, IssuerId2#8325]

: +- Project [IssuerIDDD#8273, BusinessDate#1378, stcd#7759, srcnm#7747, NetworkName#1381, NetworkURL#1382, custnum#1383, MarketCoverage#1384, DentalOnlyPlan#1385, RejectRows#1386, IssuerId#8312, IssuerId2#8325]

: +- Project [IssuerIDD#8261, IssuerIDDD#8273, BusinessDate#1378, stcd#7759, srcnm#7747, NetworkName#1381, NetworkURL#1382, custnum#1383, MarketCoverage#1384, DentalOnlyPlan#1385, RejectRows#1386, IssuerId#8312, cast(IssuerIDDD#8273 as string) AS IssuerId2#8325]

: +- Project [IssuerIDD#8261, IssuerIDDD#8273, BusinessDate#1378, stcd#7759, srcnm#7747, NetworkName#1381, NetworkURL#1382, custnum#1383, MarketCoverage#1384, DentalOnlyPlan#1385, RejectRows#1386, cast(IssuerIDD#8261 as string) AS IssuerId#8312]

: +- Project [IssuerIDD#8261, IssuerId2#1377 AS IssuerIDDD#8273, BusinessDate#1378, stcd#7759, srcnm#7747, NetworkName#1381, NetworkURL#1382, custnum#1383, MarketCoverage#1384, DentalOnlyPlan#1385, RejectRows#1386]

: +- Project [IssuerId#1376 AS IssuerIDD#8261, IssuerId2#1377, BusinessDate#1378, stcd#7759, srcnm#7747, NetworkName#1381, NetworkURL#1382, custnum#1383, MarketCoverage#1384, DentalOnlyPlan#1385, RejectRows#1386]

: +- Project [IssuerId#1376, IssuerId2#1377, BusinessDate#1378, StateCode#1379 AS stcd#7759, srcnm#7747, NetworkName#1381, NetworkURL#1382, custnum#1383, MarketCoverage#1384, DentalOnlyPlan#1385, RejectRows#1386]

: +- Project [IssuerId#1376, IssuerId2#1377, BusinessDate#1378, StateCode#1379, SourceName#1380 AS srcnm#7747, NetworkName#1381, NetworkURL#1382, custnum#1383, MarketCoverage#1384, DentalOnlyPlan#1385, RejectRows#1386]

: +- Union false, false

: :- Relation [IssuerId#1376,IssuerId2#1377,BusinessDate#1378,StateCode#1379,SourceName#1380,NetworkName#1381,NetworkURL#1382,custnum#1383,MarketCoverage#1384,DentalOnlyPlan#1385,RejectRows#1386] csv

: +- Relation [IssuerId#491,IssuerId2#492,BusinessDate#493,StateCode#494,SourceName#495,NetworkName#496,NetworkURL#497,custnum#498,MarketCoverage#499,DentalOnlyPlan#500,RejectRows#501] csv

+- SubqueryAlias b

+- Project [BusinessDate#44056 AS b\_BusinessDate#43915, stcd#44080, srcnm#44081, b\_NetworkName#43885, b\_NetworkURL#43870, custnum#44061, MarketCoverage#44062, RejectRows#44064, b\_IssuerId#43900, IssuerId2#44079, issueridcomposite#44078, sysdt#44077, systs#44076, Protocol\_Used#30945]

+- Project [BusinessDate#44056, stcd#44080, srcnm#44081, b\_NetworkName#43885, b\_NetworkURL#43870, custnum#44061, MarketCoverage#44062, RejectRows#44064, IssuerId#8312 AS b\_IssuerId#43900, IssuerId2#44079, issueridcomposite#44078, sysdt#44077, systs#44076, Protocol\_Used#30945]

+- Project [BusinessDate#44056, stcd#44080, srcnm#44081, NetworkName#44059 AS b\_NetworkName#43885, b\_NetworkURL#43870, custnum#44061, MarketCoverage#44062, RejectRows#44064, IssuerId#8312, IssuerId2#44079, issueridcomposite#44078, sysdt#44077, systs#44076, Protocol\_Used#30945]

+- Project [BusinessDate#44056, stcd#44080, srcnm#44081, NetworkName#44059, NetworkURL#44060 AS b\_NetworkURL#43870, custnum#44061, MarketCoverage#44062, RejectRows#44064, IssuerId#8312, IssuerId2#44079, issueridcomposite#44078, sysdt#44077, systs#44076, Protocol\_Used#30945]

+- Project [BusinessDate#44056, stcd#44080, srcnm#44081, NetworkName#44059, NetworkURL#44060, custnum#44061, MarketCoverage#44062, RejectRows#44064, IssuerId#8312, IssuerId2#44079, issueridcomposite#44078, sysdt#44077, systs#44076, CASE WHEN NetworkURL#44060 LIKE https://% THEN http secured WHEN NetworkURL#44060 LIKE http://% THEN http non secured ELSE No Protocol END AS Protocol\_Used#30945]

+- SubqueryAlias insureview

+- View (`insureview`, [BusinessDate#44056,stcd#44080,srcnm#44081,NetworkName#44059,NetworkURL#44060,custnum#44061,MarketCoverage#44062,RejectRows#44064,IssuerId#8312,IssuerId2#44079,issueridcomposite#44078,sysdt#44077,systs#44076])

+- Project [BusinessDate#44056, stcd#44080, srcnm#44081, NetworkName#44059, NetworkURL#44060, custnum#44061, MarketCoverage#44062, RejectRows#44064, IssuerId#8312, IssuerId2#44079, issueridcomposite#44078, sysdt#44077, current\_timestamp() AS systs#44076]

+- Project [BusinessDate#44056, stcd#44080, srcnm#44081, NetworkName#44059, NetworkURL#44060, custnum#44061, MarketCoverage#44062, RejectRows#44064, IssuerId#8312, IssuerId2#44079, issueridcomposite#44078, current\_date(Some(Asia/Kolkata)) AS sysdt#44077]

+- Project [BusinessDate#44056, stcd#44080, srcnm#44081, NetworkName#44059, NetworkURL#44060, custnum#44061, MarketCoverage#44062, RejectRows#44064, IssuerId#8312, IssuerId2#44079, issueridcomposite#44078]

+- Project [BusinessDate#44056, stcd#44080, srcnm#44081, NetworkName#44059, NetworkURL#44060, custnum#44061, MarketCoverage#44062, DentalOnlyPlan#44063, RejectRows#44064, IssuerId#8312, IssuerId2#44079, concat(IssuerId#8312, IssuerId2#44079) AS issueridcomposite#44078]

+- Project [BusinessDate#44056, stcd#44080, srcnm#44081, NetworkName#44059, NetworkURL#44060, custnum#44061, MarketCoverage#44062, DentalOnlyPlan#44063, RejectRows#44064, IssuerId#8312, IssuerId2#44079]

+- Project [IssuerIDDD#8273, BusinessDate#44056, stcd#44080, srcnm#44081, NetworkName#44059, NetworkURL#44060, custnum#44061, MarketCoverage#44062, DentalOnlyPlan#44063, RejectRows#44064, IssuerId#8312, IssuerId2#44079]

+- Project [IssuerIDD#8261, IssuerIDDD#8273, BusinessDate#44056, stcd#44080, srcnm#44081, NetworkName#44059, NetworkURL#44060, custnum#44061, MarketCoverage#44062, DentalOnlyPlan#44063, RejectRows#44064, IssuerId#8312, cast(IssuerIDDD#8273 as string) AS IssuerId2#44079]

+- Project [IssuerIDD#8261, IssuerIDDD#8273, BusinessDate#44056, stcd#44080, srcnm#44081, NetworkName#44059, NetworkURL#44060, custnum#44061, MarketCoverage#44062, DentalOnlyPlan#44063, RejectRows#44064, cast(IssuerIDD#8261 as string) AS IssuerId#8312]

+- Project [IssuerIDD#8261, IssuerId2#44055 AS IssuerIDDD#8273, BusinessDate#44056, stcd#44080, srcnm#44081, NetworkName#44059, NetworkURL#44060, custnum#44061, MarketCoverage#44062, DentalOnlyPlan#44063, RejectRows#44064]

+- Project [IssuerId#44054 AS IssuerIDD#8261, IssuerId2#44055, BusinessDate#44056, stcd#44080, srcnm#44081, NetworkName#44059, NetworkURL#44060, custnum#44061, MarketCoverage#44062, DentalOnlyPlan#44063, RejectRows#44064]

+- Project [IssuerId#44054, IssuerId2#44055, BusinessDate#44056, StateCode#44057 AS stcd#44080, srcnm#44081, NetworkName#44059, NetworkURL#44060, custnum#44061, MarketCoverage#44062, DentalOnlyPlan#44063, RejectRows#44064]

+- Project [IssuerId#44054, IssuerId2#44055, BusinessDate#44056, StateCode#44057, SourceName#44058 AS srcnm#44081, NetworkName#44059, NetworkURL#44060, custnum#44061, MarketCoverage#44062, DentalOnlyPlan#44063, RejectRows#44064]

+- Union false, false

:- Relation [IssuerId#44054,IssuerId2#44055,BusinessDate#44056,StateCode#44057,SourceName#44058,NetworkName#44059,NetworkURL#44060,custnum#44061,MarketCoverage#44062,DentalOnlyPlan#44063,RejectRows#44064] csv

+- Relation [IssuerId#44065,IssuerId2#44066,BusinessDate#44067,StateCode#44068,SourceName#44069,NetworkName#44070,NetworkURL#44071,custnum#44072,MarketCoverage#44073,DentalOnlyPlan#44074,RejectRows#44075] csv

>>>

>>> # Show the columns to ensure uniqueness

... df\_combined.columns

['BusinessDate', 'stcd', 'srcnm', 'NetworkName', 'NetworkURL', 'custnum', 'MarketCoverage', 'RejectRows', 'IssuerId', 'IssuerId2', 'issueridcomposite', 'sysdt', 'systs', 'month', 'year', 'BusinessDate', 'stcd', 'srcnm', 'NetworkName', 'NetworkURL', 'custnum', 'MarketCoverage', 'RejectRows', 'IssuerId', 'IssuerId2', 'issueridcomposite', 'sysdt', 'systs', 'Protocol\_Used', 'BusinessDate', 'stcd', 'srcnm', 'NetworkName', 'NetworkURL', 'custnum', 'MarketCoverage', 'RejectRows', 'IssuerId', 'IssuerId2', 'issueridcomposite', 'sysdt', 'systs', 'curdt', 'curts', 'BusinessDate', 'stcd', 'srcnm', 'NetworkName', 'NetworkURL', 'custnum', 'MarketCoverage', 'RejectRows', 'IssuerId', 'IssuerId2', 'issueridcomposite', 'sysdt', 'systs', 'cleannetworkname']

>>>

>>> from pyspark.sql.functions import col

>>>

>>> # Rename columns in each DataFrame to avoid ambiguity

... df1 = result\_df\_SQL.alias("a").select(

... col("BusinessDate").alias("a\_BusinessDate"),

... col("stcd").alias("a\_stcd"),

... col("srcnm").alias("a\_srcnm"),

... col("NetworkName").alias("a\_NetworkName"),

... col("NetworkURL").alias("a\_NetworkURL"),

... col("custnum").alias("a\_custnum"),

... col("MarketCoverage").alias("a\_MarketCoverage"),

... col("RejectRows").alias("a\_RejectRows"),

... col("IssuerId").alias("a\_IssuerId"),

... col("IssuerId2").alias("a\_IssuerId2"),

... col("issueridcomposite").alias("a\_issueridcomposite"),

... col("sysdt").alias("a\_sysdt"),

... col("systs").alias("a\_systs"),

... col("month").alias("a\_month"),

... col("year").alias("a\_year")

... )

Traceback (most recent call last):

File "<stdin>", line 17, in <module>

File "/usr/local/spark/python/pyspark/sql/dataframe.py", line 1685, in select

jdf = self.\_jdf.select(self.\_jcols(\*cols))

File "/usr/local/spark/python/lib/py4j-0.10.9.3-src.zip/py4j/java\_gateway.py", line 1322, in \_\_call\_\_

File "/usr/local/spark/python/pyspark/sql/utils.py", line 117, in deco

raise converted from None

pyspark.sql.utils.AnalysisException: cannot resolve 'BusinessDate' given input columns: [a.IssuerId2, a.MarketCoverage, a.RejectRows, a.a\_BusinessDate, a.a\_IssuerId, a.a\_NetworkName, a.a\_NetworkURL, a.custnum, a.issueridcomposite, a.month, a.srcnm, a.stcd, a.sysdt, a.systs, a.year];

'Project ['BusinessDate AS a\_BusinessDate#44082, stcd#7759 AS a\_stcd#44083, srcnm#7747 AS a\_srcnm#44084, 'NetworkName AS a\_NetworkName#44085, 'NetworkURL AS a\_NetworkURL#44086, custnum#1383 AS a\_custnum#44087, MarketCoverage#1384 AS a\_MarketCoverage#44088, RejectRows#1386 AS a\_RejectRows#44089, 'IssuerId AS a\_IssuerId#44090, IssuerId2#8325 AS a\_IssuerId2#44091, issueridcomposite#8901 AS a\_issueridcomposite#44092, sysdt#9003 AS a\_sysdt#44093, systs#9016 AS a\_systs#44094, month#30928 AS a\_month#44095, year#30929 AS a\_year#44096]

+- SubqueryAlias a

+- Project [BusinessDate#1378 AS a\_BusinessDate#43854, stcd#7759, srcnm#7747, a\_NetworkName#43822, a\_NetworkURL#43806, custnum#1383, MarketCoverage#1384, RejectRows#1386, a\_IssuerId#43838, IssuerId2#8325, issueridcomposite#8901, sysdt#9003, systs#9016, month#30928, year#30929]

+- Project [BusinessDate#1378, stcd#7759, srcnm#7747, a\_NetworkName#43822, a\_NetworkURL#43806, custnum#1383, MarketCoverage#1384, RejectRows#1386, IssuerId#8312 AS a\_IssuerId#43838, IssuerId2#8325, issueridcomposite#8901, sysdt#9003, systs#9016, month#30928, year#30929]

+- Project [BusinessDate#1378, stcd#7759, srcnm#7747, NetworkName#1381 AS a\_NetworkName#43822, a\_NetworkURL#43806, custnum#1383, MarketCoverage#1384, RejectRows#1386, IssuerId#8312, IssuerId2#8325, issueridcomposite#8901, sysdt#9003, systs#9016, month#30928, year#30929]

+- Project [BusinessDate#1378, stcd#7759, srcnm#7747, NetworkName#1381, NetworkURL#1382 AS a\_NetworkURL#43806, custnum#1383, MarketCoverage#1384, RejectRows#1386, IssuerId#8312, IssuerId2#8325, issueridcomposite#8901, sysdt#9003, systs#9016, month#30928, year#30929]

+- Project [BusinessDate#1378, stcd#7759, srcnm#7747, NetworkName#1381, NetworkURL#1382, custnum#1383, MarketCoverage#1384, RejectRows#1386, IssuerId#8312, IssuerId2#8325, issueridcomposite#8901, sysdt#9003, systs#9016, month(BusinessDate#1378) AS month#30928, year(BusinessDate#1378) AS year#30929]

+- SubqueryAlias insureview

+- View (`insureview`, [BusinessDate#1378,stcd#7759,srcnm#7747,NetworkName#1381,NetworkURL#1382,custnum#1383,MarketCoverage#1384,RejectRows#1386,IssuerId#8312,IssuerId2#8325,issueridcomposite#8901,sysdt#9003,systs#9016])

+- Project [BusinessDate#1378, stcd#7759, srcnm#7747, NetworkName#1381, NetworkURL#1382, custnum#1383, MarketCoverage#1384, RejectRows#1386, IssuerId#8312, IssuerId2#8325, issueridcomposite#8901, sysdt#9003, current\_timestamp() AS systs#9016]

+- Project [BusinessDate#1378, stcd#7759, srcnm#7747, NetworkName#1381, NetworkURL#1382, custnum#1383, MarketCoverage#1384, RejectRows#1386, IssuerId#8312, IssuerId2#8325, issueridcomposite#8901, current\_date(Some(Asia/Kolkata)) AS sysdt#9003]

+- Project [BusinessDate#1378, stcd#7759, srcnm#7747, NetworkName#1381, NetworkURL#1382, custnum#1383, MarketCoverage#1384, RejectRows#1386, IssuerId#8312, IssuerId2#8325, issueridcomposite#8901]

+- Project [BusinessDate#1378, stcd#7759, srcnm#7747, NetworkName#1381, NetworkURL#1382, custnum#1383, MarketCoverage#1384, DentalOnlyPlan#1385, RejectRows#1386, IssuerId#8312, IssuerId2#8325, concat(IssuerId#8312, IssuerId2#8325) AS issueridcomposite#8901]

+- Project [BusinessDate#1378, stcd#7759, srcnm#7747, NetworkName#1381, NetworkURL#1382, custnum#1383, MarketCoverage#1384, DentalOnlyPlan#1385, RejectRows#1386, IssuerId#8312, IssuerId2#8325]

+- Project [IssuerIDDD#8273, BusinessDate#1378, stcd#7759, srcnm#7747, NetworkName#1381, NetworkURL#1382, custnum#1383, MarketCoverage#1384, DentalOnlyPlan#1385, RejectRows#1386, IssuerId#8312, IssuerId2#8325]

+- Project [IssuerIDD#8261, IssuerIDDD#8273, BusinessDate#1378, stcd#7759, srcnm#7747, NetworkName#1381, NetworkURL#1382, custnum#1383, MarketCoverage#1384, DentalOnlyPlan#1385, RejectRows#1386, IssuerId#8312, cast(IssuerIDDD#8273 as string) AS IssuerId2#8325]

+- Project [IssuerIDD#8261, IssuerIDDD#8273, BusinessDate#1378, stcd#7759, srcnm#7747, NetworkName#1381, NetworkURL#1382, custnum#1383, MarketCoverage#1384, DentalOnlyPlan#1385, RejectRows#1386, cast(IssuerIDD#8261 as string) AS IssuerId#8312]

+- Project [IssuerIDD#8261, IssuerId2#1377 AS IssuerIDDD#8273, BusinessDate#1378, stcd#7759, srcnm#7747, NetworkName#1381, NetworkURL#1382, custnum#1383, MarketCoverage#1384, DentalOnlyPlan#1385, RejectRows#1386]

+- Project [IssuerId#1376 AS IssuerIDD#8261, IssuerId2#1377, BusinessDate#1378, stcd#7759, srcnm#7747, NetworkName#1381, NetworkURL#1382, custnum#1383, MarketCoverage#1384, DentalOnlyPlan#1385, RejectRows#1386]

+- Project [IssuerId#1376, IssuerId2#1377, BusinessDate#1378, StateCode#1379 AS stcd#7759, srcnm#7747, NetworkName#1381, NetworkURL#1382, custnum#1383, MarketCoverage#1384, DentalOnlyPlan#1385, RejectRows#1386]

+- Project [IssuerId#1376, IssuerId2#1377, BusinessDate#1378, StateCode#1379, SourceName#1380 AS srcnm#7747, NetworkName#1381, NetworkURL#1382, custnum#1383, MarketCoverage#1384, DentalOnlyPlan#1385, RejectRows#1386]

+- Union false, false

:- Relation [IssuerId#1376,IssuerId2#1377,BusinessDate#1378,StateCode#1379,SourceName#1380,NetworkName#1381,NetworkURL#1382,custnum#1383,MarketCoverage#1384,DentalOnlyPlan#1385,RejectRows#1386] csv

+- Relation [IssuerId#491,IssuerId2#492,BusinessDate#493,StateCode#494,SourceName#495,NetworkName#496,NetworkURL#497,custnum#498,MarketCoverage#499,DentalOnlyPlan#500,RejectRows#501] csv

>>>

>>> df2 = result\_df\_SQL\_1.alias("b").select(

... col("BusinessDate").alias("b\_BusinessDate"),

... col("stcd").alias("b\_stcd"),

... col("srcnm").alias("b\_srcnm"),

... col("NetworkName").alias("b\_NetworkName"),

... col("NetworkURL").alias("b\_NetworkURL"),

... col("custnum").alias("b\_custnum"),

... col("MarketCoverage").alias("b\_MarketCoverage"),

... col("RejectRows").alias("b\_RejectRows"),

... col("IssuerId").alias("b\_IssuerId"),

... col("IssuerId2").alias("b\_IssuerId2"),

... col("issueridcomposite").alias("b\_issueridcomposite"),

... col("sysdt").alias("b\_sysdt"),

... col("systs").alias("b\_systs"),

... col("Protocol\_Used").alias("b\_Protocol\_Used")

... )

Traceback (most recent call last):

File "<stdin>", line 15, in <module>

File "/usr/local/spark/python/pyspark/sql/dataframe.py", line 1685, in select

jdf = self.\_jdf.select(self.\_jcols(\*cols))

File "/usr/local/spark/python/lib/py4j-0.10.9.3-src.zip/py4j/java\_gateway.py", line 1322, in \_\_call\_\_

File "/usr/local/spark/python/pyspark/sql/utils.py", line 117, in deco

raise converted from None

pyspark.sql.utils.AnalysisException: cannot resolve 'BusinessDate' given input columns: [b.IssuerId2, b.MarketCoverage, b.Protocol\_Used, b.RejectRows, b.b\_BusinessDate, b.b\_IssuerId, b.b\_NetworkName, b.b\_NetworkURL, b.custnum, b.issueridcomposite, b.srcnm, b.stcd, b.sysdt, b.systs];

'Project ['BusinessDate AS b\_BusinessDate#44097, stcd#7759 AS b\_stcd#44098, srcnm#7747 AS b\_srcnm#44099, 'NetworkName AS b\_NetworkName#44100, 'NetworkURL AS b\_NetworkURL#44101, custnum#1383 AS b\_custnum#44102, MarketCoverage#1384 AS b\_MarketCoverage#44103, RejectRows#1386 AS b\_RejectRows#44104, 'IssuerId AS b\_IssuerId#44105, IssuerId2#8325 AS b\_IssuerId2#44106, issueridcomposite#8901 AS b\_issueridcomposite#44107, sysdt#9003 AS b\_sysdt#44108, systs#9016 AS b\_systs#44109, Protocol\_Used#30945 AS b\_Protocol\_Used#44110]

+- SubqueryAlias b

+- Project [BusinessDate#1378 AS b\_BusinessDate#43915, stcd#7759, srcnm#7747, b\_NetworkName#43885, b\_NetworkURL#43870, custnum#1383, MarketCoverage#1384, RejectRows#1386, b\_IssuerId#43900, IssuerId2#8325, issueridcomposite#8901, sysdt#9003, systs#9016, Protocol\_Used#30945]

+- Project [BusinessDate#1378, stcd#7759, srcnm#7747, b\_NetworkName#43885, b\_NetworkURL#43870, custnum#1383, MarketCoverage#1384, RejectRows#1386, IssuerId#8312 AS b\_IssuerId#43900, IssuerId2#8325, issueridcomposite#8901, sysdt#9003, systs#9016, Protocol\_Used#30945]

+- Project [BusinessDate#1378, stcd#7759, srcnm#7747, NetworkName#1381 AS b\_NetworkName#43885, b\_NetworkURL#43870, custnum#1383, MarketCoverage#1384, RejectRows#1386, IssuerId#8312, IssuerId2#8325, issueridcomposite#8901, sysdt#9003, systs#9016, Protocol\_Used#30945]

+- Project [BusinessDate#1378, stcd#7759, srcnm#7747, NetworkName#1381, NetworkURL#1382 AS b\_NetworkURL#43870, custnum#1383, MarketCoverage#1384, RejectRows#1386, IssuerId#8312, IssuerId2#8325, issueridcomposite#8901, sysdt#9003, systs#9016, Protocol\_Used#30945]

+- Project [BusinessDate#1378, stcd#7759, srcnm#7747, NetworkName#1381, NetworkURL#1382, custnum#1383, MarketCoverage#1384, RejectRows#1386, IssuerId#8312, IssuerId2#8325, issueridcomposite#8901, sysdt#9003, systs#9016, CASE WHEN NetworkURL#1382 LIKE https://% THEN http secured WHEN NetworkURL#1382 LIKE http://% THEN http non secured ELSE No Protocol END AS Protocol\_Used#30945]

+- SubqueryAlias insureview

+- View (`insureview`, [BusinessDate#1378,stcd#7759,srcnm#7747,NetworkName#1381,NetworkURL#1382,custnum#1383,MarketCoverage#1384,RejectRows#1386,IssuerId#8312,IssuerId2#8325,issueridcomposite#8901,sysdt#9003,systs#9016])

+- Project [BusinessDate#1378, stcd#7759, srcnm#7747, NetworkName#1381, NetworkURL#1382, custnum#1383, MarketCoverage#1384, RejectRows#1386, IssuerId#8312, IssuerId2#8325, issueridcomposite#8901, sysdt#9003, current\_timestamp() AS systs#9016]

+- Project [BusinessDate#1378, stcd#7759, srcnm#7747, NetworkName#1381, NetworkURL#1382, custnum#1383, MarketCoverage#1384, RejectRows#1386, IssuerId#8312, IssuerId2#8325, issueridcomposite#8901, current\_date(Some(Asia/Kolkata)) AS sysdt#9003]

+- Project [BusinessDate#1378, stcd#7759, srcnm#7747, NetworkName#1381, NetworkURL#1382, custnum#1383, MarketCoverage#1384, RejectRows#1386, IssuerId#8312, IssuerId2#8325, issueridcomposite#8901]

+- Project [BusinessDate#1378, stcd#7759, srcnm#7747, NetworkName#1381, NetworkURL#1382, custnum#1383, MarketCoverage#1384, DentalOnlyPlan#1385, RejectRows#1386, IssuerId#8312, IssuerId2#8325, concat(IssuerId#8312, IssuerId2#8325) AS issueridcomposite#8901]

+- Project [BusinessDate#1378, stcd#7759, srcnm#7747, NetworkName#1381, NetworkURL#1382, custnum#1383, MarketCoverage#1384, DentalOnlyPlan#1385, RejectRows#1386, IssuerId#8312, IssuerId2#8325]

+- Project [IssuerIDDD#8273, BusinessDate#1378, stcd#7759, srcnm#7747, NetworkName#1381, NetworkURL#1382, custnum#1383, MarketCoverage#1384, DentalOnlyPlan#1385, RejectRows#1386, IssuerId#8312, IssuerId2#8325]

+- Project [IssuerIDD#8261, IssuerIDDD#8273, BusinessDate#1378, stcd#7759, srcnm#7747, NetworkName#1381, NetworkURL#1382, custnum#1383, MarketCoverage#1384, DentalOnlyPlan#1385, RejectRows#1386, IssuerId#8312, cast(IssuerIDDD#8273 as string) AS IssuerId2#8325]

+- Project [IssuerIDD#8261, IssuerIDDD#8273, BusinessDate#1378, stcd#7759, srcnm#7747, NetworkName#1381, NetworkURL#1382, custnum#1383, MarketCoverage#1384, DentalOnlyPlan#1385, RejectRows#1386, cast(IssuerIDD#8261 as string) AS IssuerId#8312]

+- Project [IssuerIDD#8261, IssuerId2#1377 AS IssuerIDDD#8273, BusinessDate#1378, stcd#7759, srcnm#7747, NetworkName#1381, NetworkURL#1382, custnum#1383, MarketCoverage#1384, DentalOnlyPlan#1385, RejectRows#1386]

+- Project [IssuerId#1376 AS IssuerIDD#8261, IssuerId2#1377, BusinessDate#1378, stcd#7759, srcnm#7747, NetworkName#1381, NetworkURL#1382, custnum#1383, MarketCoverage#1384, DentalOnlyPlan#1385, RejectRows#1386]

+- Project [IssuerId#1376, IssuerId2#1377, BusinessDate#1378, StateCode#1379 AS stcd#7759, srcnm#7747, NetworkName#1381, NetworkURL#1382, custnum#1383, MarketCoverage#1384, DentalOnlyPlan#1385, RejectRows#1386]

+- Project [IssuerId#1376, IssuerId2#1377, BusinessDate#1378, StateCode#1379, SourceName#1380 AS srcnm#7747, NetworkName#1381, NetworkURL#1382, custnum#1383, MarketCoverage#1384, DentalOnlyPlan#1385, RejectRows#1386]

+- Union false, false

:- Relation [IssuerId#1376,IssuerId2#1377,BusinessDate#1378,StateCode#1379,SourceName#1380,NetworkName#1381,NetworkURL#1382,custnum#1383,MarketCoverage#1384,DentalOnlyPlan#1385,RejectRows#1386] csv

+- Relation [IssuerId#491,IssuerId2#492,BusinessDate#493,StateCode#494,SourceName#495,NetworkName#496,NetworkURL#497,custnum#498,MarketCoverage#499,DentalOnlyPlan#500,RejectRows#501] csv

>>>

>>> df3 = result\_df\_SQL\_2.alias("c").select(

... col("BusinessDate").alias("c\_BusinessDate"),

... col("stcd").alias("c\_stcd"),

... col("srcnm").alias("c\_srcnm"),

... col("NetworkName").alias("c\_NetworkName"),

... col("NetworkURL").alias("c\_NetworkURL"),

... col("custnum").alias("c\_custnum"),

... col("MarketCoverage").alias("c\_MarketCoverage"),

... col("RejectRows").alias("c\_RejectRows"),

... col("IssuerId").alias("c\_IssuerId"),

... col("IssuerId2").alias("c\_IssuerId2"),

... col("issueridcomposite").alias("c\_issueridcomposite"),

... col("sysdt").alias("c\_sysdt"),

... col("systs").alias("c\_systs"),

... col("Protocol\_Used").alias("c\_Protocol\_Used")

... )

Traceback (most recent call last):

File "<stdin>", line 15, in <module>

File "/usr/local/spark/python/pyspark/sql/dataframe.py", line 1685, in select

jdf = self.\_jdf.select(self.\_jcols(\*cols))

File "/usr/local/spark/python/lib/py4j-0.10.9.3-src.zip/py4j/java\_gateway.py", line 1322, in \_\_call\_\_

File "/usr/local/spark/python/pyspark/sql/utils.py", line 117, in deco

raise converted from None

pyspark.sql.utils.AnalysisException: cannot resolve 'BusinessDate' given input columns: [c.IssuerId2, c.MarketCoverage, c.RejectRows, c.c\_BusinessDate, c.c\_IssuerId, c.c\_NetworkName, c.c\_NetworkURL, c.curdt, c.curts, c.custnum, c.issueridcomposite, c.srcnm, c.stcd, c.sysdt, c.systs];

'Project ['BusinessDate AS c\_BusinessDate#44111, stcd#7759 AS c\_stcd#44112, srcnm#7747 AS c\_srcnm#44113, 'NetworkName AS c\_NetworkName#44114, 'NetworkURL AS c\_NetworkURL#44115, custnum#1383 AS c\_custnum#44116, MarketCoverage#1384 AS c\_MarketCoverage#44117, RejectRows#1386 AS c\_RejectRows#44118, 'IssuerId AS c\_IssuerId#44119, IssuerId2#8325 AS c\_IssuerId2#44120, issueridcomposite#8901 AS c\_issueridcomposite#44121, sysdt#9003 AS c\_sysdt#44122, systs#9016 AS c\_systs#44123, 'Protocol\_Used AS c\_Protocol\_Used#44124]

+- SubqueryAlias c

+- Project [BusinessDate#1378 AS c\_BusinessDate#43978, stcd#7759, srcnm#7747, c\_NetworkName#43946, c\_NetworkURL#43930, custnum#1383, MarketCoverage#1384, RejectRows#1386, c\_IssuerId#43962, IssuerId2#8325, issueridcomposite#8901, sysdt#9003, systs#9016, curdt#31490, curts#31491]

+- Project [BusinessDate#1378, stcd#7759, srcnm#7747, c\_NetworkName#43946, c\_NetworkURL#43930, custnum#1383, MarketCoverage#1384, RejectRows#1386, IssuerId#8312 AS c\_IssuerId#43962, IssuerId2#8325, issueridcomposite#8901, sysdt#9003, systs#9016, curdt#31490, curts#31491]

+- Project [BusinessDate#1378, stcd#7759, srcnm#7747, NetworkName#1381 AS c\_NetworkName#43946, c\_NetworkURL#43930, custnum#1383, MarketCoverage#1384, RejectRows#1386, IssuerId#8312, IssuerId2#8325, issueridcomposite#8901, sysdt#9003, systs#9016, curdt#31490, curts#31491]

+- Project [BusinessDate#1378, stcd#7759, srcnm#7747, NetworkName#1381, NetworkURL#1382 AS c\_NetworkURL#43930, custnum#1383, MarketCoverage#1384, RejectRows#1386, IssuerId#8312, IssuerId2#8325, issueridcomposite#8901, sysdt#9003, systs#9016, curdt#31490, curts#31491]

+- Project [BusinessDate#1378, stcd#7759, srcnm#7747, NetworkName#1381, NetworkURL#1382, custnum#1383, MarketCoverage#1384, RejectRows#1386, IssuerId#8312, IssuerId2#8325, issueridcomposite#8901, sysdt#9003, systs#9016, current\_date(Some(Asia/Kolkata)) AS curdt#31490, current\_timestamp() AS curts#31491]

+- SubqueryAlias insureview

+- View (`insureview`, [BusinessDate#1378,stcd#7759,srcnm#7747,NetworkName#1381,NetworkURL#1382,custnum#1383,MarketCoverage#1384,RejectRows#1386,IssuerId#8312,IssuerId2#8325,issueridcomposite#8901,sysdt#9003,systs#9016])

+- Project [BusinessDate#1378, stcd#7759, srcnm#7747, NetworkName#1381, NetworkURL#1382, custnum#1383, MarketCoverage#1384, RejectRows#1386, IssuerId#8312, IssuerId2#8325, issueridcomposite#8901, sysdt#9003, current\_timestamp() AS systs#9016]

+- Project [BusinessDate#1378, stcd#7759, srcnm#7747, NetworkName#1381, NetworkURL#1382, custnum#1383, MarketCoverage#1384, RejectRows#1386, IssuerId#8312, IssuerId2#8325, issueridcomposite#8901, current\_date(Some(Asia/Kolkata)) AS sysdt#9003]

+- Project [BusinessDate#1378, stcd#7759, srcnm#7747, NetworkName#1381, NetworkURL#1382, custnum#1383, MarketCoverage#1384, RejectRows#1386, IssuerId#8312, IssuerId2#8325, issueridcomposite#8901]

+- Project [BusinessDate#1378, stcd#7759, srcnm#7747, NetworkName#1381, NetworkURL#1382, custnum#1383, MarketCoverage#1384, DentalOnlyPlan#1385, RejectRows#1386, IssuerId#8312, IssuerId2#8325, concat(IssuerId#8312, IssuerId2#8325) AS issueridcomposite#8901]

+- Project [BusinessDate#1378, stcd#7759, srcnm#7747, NetworkName#1381, NetworkURL#1382, custnum#1383, MarketCoverage#1384, DentalOnlyPlan#1385, RejectRows#1386, IssuerId#8312, IssuerId2#8325]

+- Project [IssuerIDDD#8273, BusinessDate#1378, stcd#7759, srcnm#7747, NetworkName#1381, NetworkURL#1382, custnum#1383, MarketCoverage#1384, DentalOnlyPlan#1385, RejectRows#1386, IssuerId#8312, IssuerId2#8325]

+- Project [IssuerIDD#8261, IssuerIDDD#8273, BusinessDate#1378, stcd#7759, srcnm#7747, NetworkName#1381, NetworkURL#1382, custnum#1383, MarketCoverage#1384, DentalOnlyPlan#1385, RejectRows#1386, IssuerId#8312, cast(IssuerIDDD#8273 as string) AS IssuerId2#8325]

+- Project [IssuerIDD#8261, IssuerIDDD#8273, BusinessDate#1378, stcd#7759, srcnm#7747, NetworkName#1381, NetworkURL#1382, custnum#1383, MarketCoverage#1384, DentalOnlyPlan#1385, RejectRows#1386, cast(IssuerIDD#8261 as string) AS IssuerId#8312]

+- Project [IssuerIDD#8261, IssuerId2#1377 AS IssuerIDDD#8273, BusinessDate#1378, stcd#7759, srcnm#7747, NetworkName#1381, NetworkURL#1382, custnum#1383, MarketCoverage#1384, DentalOnlyPlan#1385, RejectRows#1386]

+- Project [IssuerId#1376 AS IssuerIDD#8261, IssuerId2#1377, BusinessDate#1378, stcd#7759, srcnm#7747, NetworkName#1381, NetworkURL#1382, custnum#1383, MarketCoverage#1384, DentalOnlyPlan#1385, RejectRows#1386]

+- Project [IssuerId#1376, IssuerId2#1377, BusinessDate#1378, StateCode#1379 AS stcd#7759, srcnm#7747, NetworkName#1381, NetworkURL#1382, custnum#1383, MarketCoverage#1384, DentalOnlyPlan#1385, RejectRows#1386]

+- Project [IssuerId#1376, IssuerId2#1377, BusinessDate#1378, StateCode#1379, SourceName#1380 AS srcnm#7747, NetworkName#1381, NetworkURL#1382, custnum#1383, MarketCoverage#1384, DentalOnlyPlan#1385, RejectRows#1386]

+- Union false, false

:- Relation [IssuerId#1376,IssuerId2#1377,BusinessDate#1378,StateCode#1379,SourceName#1380,NetworkName#1381,NetworkURL#1382,custnum#1383,MarketCoverage#1384,DentalOnlyPlan#1385,RejectRows#1386] csv

+- Relation [IssuerId#491,IssuerId2#492,BusinessDate#493,StateCode#494,SourceName#495,NetworkName#496,NetworkURL#497,custnum#498,MarketCoverage#499,DentalOnlyPlan#500,RejectRows#501] csv

>>>

>>> df4 = result\_df\_SQL\_3.alias("d").select(

... col("BusinessDate").alias("d\_BusinessDate"),

... col("stcd").alias("d\_stcd"),

... col("srcnm").alias("d\_srcnm"),

... col("NetworkName").alias("d\_NetworkName"),

... col("NetworkURL").alias("d\_NetworkURL"),

... col("custnum").alias("d\_custnum"),

... col("MarketCoverage").alias("d\_MarketCoverage"),

... col("RejectRows").alias("d\_RejectRows"),

... col("IssuerId").alias("d\_IssuerId"),

... col("IssuerId2").alias("d\_IssuerId2"),

... col("issueridcomposite").alias("d\_issueridcomposite"),

... col("sysdt").alias("d\_sysdt"),

... col("systs").alias("d\_systs"),

... col("cleannetworkname").alias("d\_cleannetworkname")

... )

Traceback (most recent call last):

File "<stdin>", line 15, in <module>

File "/usr/local/spark/python/pyspark/sql/dataframe.py", line 1685, in select

jdf = self.\_jdf.select(self.\_jcols(\*cols))

File "/usr/local/spark/python/lib/py4j-0.10.9.3-src.zip/py4j/java\_gateway.py", line 1322, in \_\_call\_\_

File "/usr/local/spark/python/pyspark/sql/utils.py", line 117, in deco

raise converted from None

pyspark.sql.utils.AnalysisException: cannot resolve 'BusinessDate' given input columns: [d.IssuerId2, d.MarketCoverage, d.RejectRows, d.cleannetworkname, d.custnum, d.d\_BusinessDate, d.d\_IssuerId, d.d\_NetworkName, d.d\_NetworkURL, d.issueridcomposite, d.srcnm, d.stcd, d.sysdt, d.systs];

'Project ['BusinessDate AS d\_BusinessDate#44125, stcd#7759 AS d\_stcd#44126, srcnm#7747 AS d\_srcnm#44127, 'NetworkName AS d\_NetworkName#44128, 'NetworkURL AS d\_NetworkURL#44129, custnum#1383 AS d\_custnum#44130, MarketCoverage#1384 AS d\_MarketCoverage#44131, RejectRows#1386 AS d\_RejectRows#44132, 'IssuerId AS d\_IssuerId#44133, IssuerId2#8325 AS d\_IssuerId2#44134, issueridcomposite#8901 AS d\_issueridcomposite#44135, sysdt#9003 AS d\_sysdt#44136, systs#9016 AS d\_systs#44137, cleannetworkname#32033 AS d\_cleannetworkname#44138]

+- SubqueryAlias d

+- Project [BusinessDate#1378 AS d\_BusinessDate#44039, stcd#7759, srcnm#7747, d\_NetworkName#44009, d\_NetworkURL#43994, custnum#1383, MarketCoverage#1384, RejectRows#1386, d\_IssuerId#44024, IssuerId2#8325, issueridcomposite#8901, sysdt#9003, systs#9016, cleannetworkname#32033]

+- Project [BusinessDate#1378, stcd#7759, srcnm#7747, d\_NetworkName#44009, d\_NetworkURL#43994, custnum#1383, MarketCoverage#1384, RejectRows#1386, IssuerId#8312 AS d\_IssuerId#44024, IssuerId2#8325, issueridcomposite#8901, sysdt#9003, systs#9016, cleannetworkname#32033]

+- Project [BusinessDate#1378, stcd#7759, srcnm#7747, NetworkName#1381 AS d\_NetworkName#44009, d\_NetworkURL#43994, custnum#1383, MarketCoverage#1384, RejectRows#1386, IssuerId#8312, IssuerId2#8325, issueridcomposite#8901, sysdt#9003, systs#9016, cleannetworkname#32033]

+- Project [BusinessDate#1378, stcd#7759, srcnm#7747, NetworkName#1381, NetworkURL#1382 AS d\_NetworkURL#43994, custnum#1383, MarketCoverage#1384, RejectRows#1386, IssuerId#8312, IssuerId2#8325, issueridcomposite#8901, sysdt#9003, systs#9016, cleannetworkname#32033]

+- Project [BusinessDate#1378, stcd#7759, srcnm#7747, NetworkName#1381, NetworkURL#1382, custnum#1383, MarketCoverage#1384, RejectRows#1386, IssuerId#8312, IssuerId2#8325, issueridcomposite#8901, sysdt#9003, systs#9016, sql\_function(NetworkName#1381) AS cleannetworkname#32033]

+- SubqueryAlias insureview

+- View (`insureview`, [BusinessDate#1378,stcd#7759,srcnm#7747,NetworkName#1381,NetworkURL#1382,custnum#1383,MarketCoverage#1384,RejectRows#1386,IssuerId#8312,IssuerId2#8325,issueridcomposite#8901,sysdt#9003,systs#9016])

+- Project [BusinessDate#1378, stcd#7759, srcnm#7747, NetworkName#1381, NetworkURL#1382, custnum#1383, MarketCoverage#1384, RejectRows#1386, IssuerId#8312, IssuerId2#8325, issueridcomposite#8901, sysdt#9003, current\_timestamp() AS systs#9016]

+- Project [BusinessDate#1378, stcd#7759, srcnm#7747, NetworkName#1381, NetworkURL#1382, custnum#1383, MarketCoverage#1384, RejectRows#1386, IssuerId#8312, IssuerId2#8325, issueridcomposite#8901, current\_date(Some(Asia/Kolkata)) AS sysdt#9003]

+- Project [BusinessDate#1378, stcd#7759, srcnm#7747, NetworkName#1381, NetworkURL#1382, custnum#1383, MarketCoverage#1384, RejectRows#1386, IssuerId#8312, IssuerId2#8325, issueridcomposite#8901]

+- Project [BusinessDate#1378, stcd#7759, srcnm#7747, NetworkName#1381, NetworkURL#1382, custnum#1383, MarketCoverage#1384, DentalOnlyPlan#1385, RejectRows#1386, IssuerId#8312, IssuerId2#8325, concat(IssuerId#8312, IssuerId2#8325) AS issueridcomposite#8901]

+- Project [BusinessDate#1378, stcd#7759, srcnm#7747, NetworkName#1381, NetworkURL#1382, custnum#1383, MarketCoverage#1384, DentalOnlyPlan#1385, RejectRows#1386, IssuerId#8312, IssuerId2#8325]

+- Project [IssuerIDDD#8273, BusinessDate#1378, stcd#7759, srcnm#7747, NetworkName#1381, NetworkURL#1382, custnum#1383, MarketCoverage#1384, DentalOnlyPlan#1385, RejectRows#1386, IssuerId#8312, IssuerId2#8325]

+- Project [IssuerIDD#8261, IssuerIDDD#8273, BusinessDate#1378, stcd#7759, srcnm#7747, NetworkName#1381, NetworkURL#1382, custnum#1383, MarketCoverage#1384, DentalOnlyPlan#1385, RejectRows#1386, IssuerId#8312, cast(IssuerIDDD#8273 as string) AS IssuerId2#8325]

+- Project [IssuerIDD#8261, IssuerIDDD#8273, BusinessDate#1378, stcd#7759, srcnm#7747, NetworkName#1381, NetworkURL#1382, custnum#1383, MarketCoverage#1384, DentalOnlyPlan#1385, RejectRows#1386, cast(IssuerIDD#8261 as string) AS IssuerId#8312]

+- Project [IssuerIDD#8261, IssuerId2#1377 AS IssuerIDDD#8273, BusinessDate#1378, stcd#7759, srcnm#7747, NetworkName#1381, NetworkURL#1382, custnum#1383, MarketCoverage#1384, DentalOnlyPlan#1385, RejectRows#1386]

+- Project [IssuerId#1376 AS IssuerIDD#8261, IssuerId2#1377, BusinessDate#1378, stcd#7759, srcnm#7747, NetworkName#1381, NetworkURL#1382, custnum#1383, MarketCoverage#1384, DentalOnlyPlan#1385, RejectRows#1386]

+- Project [IssuerId#1376, IssuerId2#1377, BusinessDate#1378, StateCode#1379 AS stcd#7759, srcnm#7747, NetworkName#1381, NetworkURL#1382, custnum#1383, MarketCoverage#1384, DentalOnlyPlan#1385, RejectRows#1386]

+- Project [IssuerId#1376, IssuerId2#1377, BusinessDate#1378, StateCode#1379, SourceName#1380 AS srcnm#7747, NetworkName#1381, NetworkURL#1382, custnum#1383, MarketCoverage#1384, DentalOnlyPlan#1385, RejectRows#1386]

+- Union false, false

:- Relation [IssuerId#1376,IssuerId2#1377,BusinessDate#1378,StateCode#1379,SourceName#1380,NetworkName#1381,NetworkURL#1382,custnum#1383,MarketCoverage#1384,DentalOnlyPlan#1385,RejectRows#1386] csv

+- Relation [IssuerId#491,IssuerId2#492,BusinessDate#493,StateCode#494,SourceName#495,NetworkName#496,NetworkURL#497,custnum#498,MarketCoverage#499,DentalOnlyPlan#500,RejectRows#501] csv

>>>

>>> # Perform the join with renamed columns

... df\_combined = df1 \

... .join(df2, col("a\_IssuerId") == col("b\_IssuerId"), how="inner") \

... .join(df3, col("a\_IssuerId") == col("c\_IssuerId"), how="inner") \

... .join(df4, col("a\_IssuerId") == col("d\_IssuerId"), how="inner")

Traceback (most recent call last):

File "<stdin>", line 2, in <module>

NameError: name 'df1' is not defined

>>>

>>> # Show the columns to verify uniqueness

... df\_combined.columns

['BusinessDate', 'stcd', 'srcnm', 'NetworkName', 'NetworkURL', 'custnum', 'MarketCoverage', 'RejectRows', 'IssuerId', 'IssuerId2', 'issueridcomposite', 'sysdt', 'systs', 'month', 'year', 'BusinessDate', 'stcd', 'srcnm', 'NetworkName', 'NetworkURL', 'custnum', 'MarketCoverage', 'RejectRows', 'IssuerId', 'IssuerId2', 'issueridcomposite', 'sysdt', 'systs', 'Protocol\_Used', 'BusinessDate', 'stcd', 'srcnm', 'NetworkName', 'NetworkURL', 'custnum', 'MarketCoverage', 'RejectRows', 'IssuerId', 'IssuerId2', 'issueridcomposite', 'sysdt', 'systs', 'curdt', 'curts', 'BusinessDate', 'stcd', 'srcnm', 'NetworkName', 'NetworkURL', 'custnum', 'MarketCoverage', 'RejectRows', 'IssuerId', 'IssuerId2', 'issueridcomposite', 'sysdt', 'systs', 'cleannetworkname']

>>> result\_df\_SQL.describe

<bound method DataFrame.describe of DataFrame[a\_BusinessDate: date, stcd: string, srcnm: string, a\_NetworkName: string, a\_NetworkURL: string, custnum: string, MarketCoverage: string, RejectRows: string, a\_IssuerId: string, IssuerId2: string, issueridcomposite: string, sysdt: date, systs: timestamp, month: int, year: int]>

>>> result\_df\_SQL\_!.describe

File "<stdin>", line 1

result\_df\_SQL\_!.describe

^

SyntaxError: invalid syntax

>>> result\_df\_SQL\_!.describe

>>> result\_df\_SQL.describe

<bound method DataFrame.describe of DataFrame[a\_BusinessDate: date, stcd: string, srcnm: string, a\_NetworkName: string, a\_NetworkURL: string, custnum: string, MarketCoverage: string, RejectRows: string, a\_IssuerId: string, IssuerId2: string, issueridcomposite: string, sysdt: date, systs: timestamp, month: int, year: int]>

>>> result\_df\_SQL\_!.describe

File "<stdin>", line 1

result\_df\_SQL\_!.describe

^

SyntaxError: invalid syntax

>>> result\_df\_SQL.describe

<bound method DataFrame.describe of DataFrame[a\_BusinessDate: date, stcd: string, srcnm: string, a\_NetworkName: string, a\_NetworkURL: string, custnum: string, MarketCoverage: string, RejectRows: string, a\_IssuerId: string, IssuerId2: string, issueridcomposite: string, sysdt: date, systs: timestamp, month: int, year: int]>

>>> result\_df\_SQL\_1.describe

<bound method DataFrame.describe of DataFrame[b\_BusinessDate: date, stcd: string, srcnm: string, b\_NetworkName: string, b\_NetworkURL: string, custnum: string, MarketCoverage: string, RejectRows: string, b\_IssuerId: string, IssuerId2: string, issueridcomposite: string, sysdt: date, systs: timestamp, Protocol\_Used: string]>

>>> result\_df\_SQL\_2.describe

<bound method DataFrame.describe of DataFrame[c\_BusinessDate: date, stcd: string, srcnm: string, c\_NetworkName: string, c\_NetworkURL: string, custnum: string, MarketCoverage: string, RejectRows: string, c\_IssuerId: string, IssuerId2: string, issueridcomposite: string, sysdt: date, systs: timestamp, curdt: date, curts: timestamp]>

>>> result\_df\_SQL\_3.describe

<bound method DataFrame.describe of DataFrame[d\_BusinessDate: date, stcd: string, srcnm: string, d\_NetworkName: string, d\_NetworkURL: string, custnum: string, MarketCoverage: string, RejectRows: string, d\_IssuerId: string, IssuerId2: string, issueridcomposite: string, sysdt: date, systs: timestamp, cleannetworkname: string]>

>>> result\_df\_SQL = result\_df\_SQL \

... .withColumnRenamed("a\_IssuerId", "IssuerId") \

... .withColumnRenamed("a\_NetworkName", "NetworkName") \

... .withColumnRenamed("a\_NetworkURL", "NetworkURL") \

... .withColumnRenamed("a\_BusinessDate", "BusinessDate")

>>> result\_df\_SQL\_1 = result\_df\_SQL\_1 \

... .withColumnRenamed("b\_IssuerId", "IssuerId") \

... .withColumnRenamed("b\_NetworkName", "NetworkName") \

... .withColumnRenamed("b\_NetworkURL", "NetworkURL") \

... .withColumnRenamed("b\_BusinessDate", "BusinessDate")

>>> result\_df\_SQL\_3 = result\_df\_SQL\_3 \

... .withColumnRenamed("d\_IssuerId", "IssuerId") \

... .withColumnRenamed("d\_NetworkName", "NetworkName") \

... .withColumnRenamed("d\_NetworkURL", "NetworkURL") \

... .withColumnRenamed("d\_BusinessDate", "BusinessDate")

>>> result\_df\_SQL\_2 = result\_df\_SQL\_2 \

... .withColumnRenamed("c\_IssuerId", "IssuerId") \

... .withColumnRenamed("c\_NetworkName", "NetworkName") \

... .withColumnRenamed("c\_NetworkURL", "NetworkURL") \

... .withColumnRenamed("c\_BusinessDate", "BusinessDate")

>>> df\_combined = result\_df\_SQL \

... .alias("a") \

... .join(result\_df\_SQL\_1.alias("b"), col("a.IssuerId") == col("b.IssuerId"), how="inner") \

... .join(result\_df\_SQL\_2.alias("c"), col("a.IssuerId") == col("c.IssuerId"), how="inner") \

... .join(result\_df\_SQL\_3.alias("d"), col("a.IssuerId") == col("d.IssuerId"), how="inner")

>>> df\_combined.columns()

Traceback (most recent call last):

File "<stdin>", line 1, in <module>

TypeError: 'list' object is not callable

>>> list=df\_combined.columns()

Traceback (most recent call last):

File "<stdin>", line 1, in <module>

TypeError: 'list' object is not callable

>>> list\_ele=df\_combined.columns()

Traceback (most recent call last):

File "<stdin>", line 1, in <module>

TypeError: 'list' object is not callable

>>> list\_ele=df\_combined.columns

>>> print(list\_Ele)

Traceback (most recent call last):

File "<stdin>", line 1, in <module>

NameError: name 'list\_Ele' is not defined

>>> print(list\_ele)

['BusinessDate', 'stcd', 'srcnm', 'NetworkName', 'NetworkURL', 'custnum', 'MarketCoverage', 'RejectRows', 'IssuerId', 'IssuerId2', 'issueridcomposite', 'sysdt', 'systs', 'month', 'year', 'BusinessDate', 'stcd', 'srcnm', 'NetworkName', 'NetworkURL', 'custnum', 'MarketCoverage', 'RejectRows', 'IssuerId', 'IssuerId2', 'issueridcomposite', 'sysdt', 'systs', 'Protocol\_Used', 'BusinessDate', 'stcd', 'srcnm', 'NetworkName', 'NetworkURL', 'custnum', 'MarketCoverage', 'RejectRows', 'IssuerId', 'IssuerId2', 'issueridcomposite', 'sysdt', 'systs', 'curdt', 'curts', 'BusinessDate', 'stcd', 'srcnm', 'NetworkName', 'NetworkURL', 'custnum', 'MarketCoverage', 'RejectRows', 'IssuerId', 'IssuerId2', 'issueridcomposite', 'sysdt', 'systs', 'cleannetworkname']

>>> df\_combined.write.parquet("hdfs:///user/hduser/Hackathon/PARQUET\_DATA")

Traceback (most recent call last):

File "<stdin>", line 1, in <module>

File "/usr/local/spark/python/pyspark/sql/readwriter.py", line 885, in parquet

self.\_jwrite.parquet(path)

File "/usr/local/spark/python/lib/py4j-0.10.9.3-src.zip/py4j/java\_gateway.py", line 1322, in \_\_call\_\_

File "/usr/local/spark/python/pyspark/sql/utils.py", line 117, in deco

raise converted from None

pyspark.sql.utils.AnalysisException: Found duplicate column(s) when inserting into hdfs://localhost:54310/user/hduser/Hackathon/PARQUET\_DATA: `businessdate`, `custnum`, `issuerid2`, `issuerid`, `issueridcomposite`, `marketcoverage`, `networkname`, `networkurl`, `rejectrows`, `srcnm`, `stcd`, `sysdt`, `systs`

>>> result\_df=df\_combined.drop("")

>>> result\_df\_SQL\_renamed = result\_df\_SQL \

... .withColumnRenamed("a\_BusinessDate", "a\_BusinessDate") \

... .withColumnRenamed("stcd", "a\_stcd") \

... .withColumnRenamed("srcnm", "a\_srcnm") \

... .withColumnRenamed("a\_NetworkName", "a\_NetworkName") \

... .withColumnRenamed("a\_NetworkURL", "a\_NetworkURL") \

... .withColumnRenamed("custnum", "a\_custnum") \

... .withColumnRenamed("MarketCoverage", "a\_MarketCoverage") \

... .withColumnRenamed("RejectRows", "a\_RejectRows") \

... .withColumnRenamed("a\_IssuerId", "a\_IssuerId") \

... .withColumnRenamed("IssuerId2", "a\_IssuerId2") \

... .withColumnRenamed("issueridcomposite", "a\_issueridcomposite") \

... .withColumnRenamed("sysdt", "a\_sysdt") \

... .withColumnRenamed("systs", "a\_systs") \

... .withColumnRenamed("month", "a\_month") \

... .withColumnRenamed("year", "a\_year")

>>> result\_df\_SQL\_1\_renamed = result\_df\_SQL\_1 \

... .withColumnRenamed("b\_BusinessDate", "b\_BusinessDate") \

... .withColumnRenamed("stcd", "b\_stcd") \

... .withColumnRenamed("srcnm", "b\_srcnm") \

... .withColumnRenamed("b\_NetworkName", "b\_NetworkName") \

... .withColumnRenamed("b\_NetworkURL", "b\_NetworkURL") \

... .withColumnRenamed("custnum", "b\_custnum") \

... .withColumnRenamed("MarketCoverage", "b\_MarketCoverage") \

... .withColumnRenamed("RejectRows", "b\_RejectRows") \

... .withColumnRenamed("b\_IssuerId", "b\_IssuerId") \

... .withColumnRenamed("IssuerId2", "b\_IssuerId2") \

... .withColumnRenamed("issueridcomposite", "b\_issueridcomposite") \

... .withColumnRenamed("sysdt", "b\_sysdt") \

... .withColumnRenamed("systs", "b\_systs") \

... .withColumnRenamed("Protocol\_Used", "b\_Protocol\_Used")

>>>

>>> result\_df\_SQL\_2\_renamed = result\_df\_SQL\_2 \

... .withColumnRenamed("c\_BusinessDate", "c\_BusinessDate") \

... .withColumnRenamed("stcd", "c\_stcd") \

... .withColumnRenamed("srcnm", "c\_srcnm") \

... .withColumnRenamed("c\_NetworkName", "c\_NetworkName") \

... .withColumnRenamed("c\_NetworkURL", "c\_NetworkURL") \

... .withColumnRenamed("custnum", "c\_custnum") \

... .withColumnRenamed("MarketCoverage", "c\_MarketCoverage") \

... .withColumnRenamed("RejectRows", "c\_RejectRows") \

... .withColumnRenamed("c\_IssuerId", "c\_IssuerId") \

... .withColumnRenamed("IssuerId2", "c\_IssuerId2") \

... .withColumnRenamed("issueridcomposite", "c\_issueridcomposite") \

... .withColumnRenamed("sysdt", "c\_sysdt") \

... .withColumnRenamed("systs", "c\_systs") \

... .withColumnRenamed("curdt", "c\_curdt") \

... .withColumnRenamed("curts", "c\_curts")

>>> result\_df\_SQL\_3\_renamed = result\_df\_SQL\_3 \

... .withColumnRenamed("d\_BusinessDate", "d\_BusinessDate") \

... .withColumnRenamed("stcd", "d\_stcd") \

... .withColumnRenamed("srcnm", "d\_srcnm") \

... .withColumnRenamed("d\_NetworkName", "d\_NetworkName") \

... .withColumnRenamed("d\_NetworkURL", "d\_NetworkURL") \

... .withColumnRenamed("custnum", "d\_custnum") \

... .withColumnRenamed("MarketCoverage", "d\_MarketCoverage") \

... .withColumnRenamed("RejectRows", "d\_RejectRows") \

... .withColumnRenamed("d\_IssuerId", "d\_IssuerId") \

... .withColumnRenamed("IssuerId2", "d\_IssuerId2") \

... .withColumnRenamed("issueridcomposite", "d\_issueridcomposite") \

... .withColumnRenamed("sysdt", "d\_sysdt") \

... .withColumnRenamed("systs", "d\_systs") \

... .withColumnRenamed("cleannetworkname", "d\_cleannetworkname")

>>> final\_joined\_df = (result\_df\_SQL\_renamed

... .join(result\_df\_SQL\_1\_renamed,

... (F.col("a\_BusinessDate") == F.col("b\_BusinessDate")) &

... (F.col("a\_stcd") == F.col("b\_stcd")) &

... (F.col("a\_srcnm") == F.col("b\_srcnm")) &

... (F.col("a\_custnum") == F.col("b\_custnum")),

... "left")

... .join(result\_df\_SQL\_2\_renamed,

... (F.col("a\_BusinessDate") == F.col("c\_BusinessDate")) &

... (F.col("a\_stcd") == F.col("c\_stcd")) &

... (F.col("a\_srcnm") == F.col("c\_srcnm")) &

... (F.col("a\_custnum") == F.col("c\_custnum")),

... "left")

... .join(result\_df\_SQL\_3\_renamed,

... (F.col("a\_BusinessDate") == F.col("d\_BusinessDate")) &

... (F.col("a\_stcd") == F.col("d\_stcd")) &

... (F.col("a\_srcnm") == F.col("d\_srcnm")) &

... (F.col("a\_custnum") == F.col("d\_custnum")),

... "left")

... )

Traceback (most recent call last):

File "<stdin>", line 5, in <module>

NameError: name 'F' is not defined

>>> from pyspark.sql import functions as F

>>> final\_joined\_df = (result\_df\_SQL\_renamed

... .join(result\_df\_SQL\_1\_renamed,

... (F.col("a\_BusinessDate") == F.col("b\_BusinessDate")) &

... (F.col("a\_stcd") == F.col("b\_stcd")) &

... (F.col("a\_srcnm") == F.col("b\_srcnm")) &

... (F.col("a\_custnum") == F.col("b\_custnum")),

... "left")

... .join(result\_df\_SQL\_2\_renamed,

... (F.col("a\_BusinessDate") == F.col("c\_BusinessDate")) &

... (F.col("a\_stcd") == F.col("c\_stcd")) &

... (F.col("a\_srcnm") == F.col("c\_srcnm")) &

... (F.col("a\_custnum") == F.col("c\_custnum")),

... "left")

... .join(result\_df\_SQL\_3\_renamed,

... (F.col("a\_BusinessDate") == F.col("d\_BusinessDate")) &

... (F.col("a\_stcd") == F.col("d\_stcd")) &

... (F.col("a\_srcnm") == F.col("d\_srcnm")) &

... (F.col("a\_custnum") == F.col("d\_custnum")),

... "left")

... )

Traceback (most recent call last):

File "<stdin>", line 7, in <module>

File "/usr/local/spark/python/pyspark/sql/dataframe.py", line 1355, in join

jdf = self.\_jdf.join(other.\_jdf, on, how)

File "/usr/local/spark/python/lib/py4j-0.10.9.3-src.zip/py4j/java\_gateway.py", line 1322, in \_\_call\_\_

File "/usr/local/spark/python/pyspark/sql/utils.py", line 117, in deco

raise converted from None

pyspark.sql.utils.AnalysisException: cannot resolve 'a\_BusinessDate' given input columns: [BusinessDate, BusinessDate, IssuerId, IssuerId, NetworkName, NetworkName, NetworkURL, NetworkURL, a\_IssuerId2, a\_MarketCoverage, a\_RejectRows, a\_custnum, a\_issueridcomposite, a\_month, a\_srcnm, a\_stcd, a\_sysdt, a\_systs, a\_year, b\_IssuerId2, b\_MarketCoverage, b\_Protocol\_Used, b\_RejectRows, b\_custnum, b\_issueridcomposite, b\_srcnm, b\_stcd, b\_sysdt, b\_systs];

'Join LeftOuter, (((('a\_BusinessDate = 'b\_BusinessDate) AND (a\_stcd#46162 = b\_stcd#46397)) AND (a\_srcnm#46178 = b\_srcnm#46412)) AND (a\_custnum#46224 = b\_custnum#46455))

:- Project [BusinessDate#44187, a\_stcd#46162, a\_srcnm#46178, NetworkName#44155, NetworkURL#44171, a\_custnum#46224, a\_MarketCoverage#46240, a\_RejectRows#46256, IssuerId#44139, a\_IssuerId2#46287, a\_issueridcomposite#46303, a\_sysdt#46319, a\_systs#46335, a\_month#46351, year#30929 AS a\_year#46367]

: +- Project [BusinessDate#44187, a\_stcd#46162, a\_srcnm#46178, NetworkName#44155, NetworkURL#44171, a\_custnum#46224, a\_MarketCoverage#46240, a\_RejectRows#46256, IssuerId#44139, a\_IssuerId2#46287, a\_issueridcomposite#46303, a\_sysdt#46319, a\_systs#46335, month#30928 AS a\_month#46351, year#30929]

: +- Project [BusinessDate#44187, a\_stcd#46162, a\_srcnm#46178, NetworkName#44155, NetworkURL#44171, a\_custnum#46224, a\_MarketCoverage#46240, a\_RejectRows#46256, IssuerId#44139, a\_IssuerId2#46287, a\_issueridcomposite#46303, a\_sysdt#46319, systs#9016 AS a\_systs#46335, month#30928, year#30929]

: +- Project [BusinessDate#44187, a\_stcd#46162, a\_srcnm#46178, NetworkName#44155, NetworkURL#44171, a\_custnum#46224, a\_MarketCoverage#46240, a\_RejectRows#46256, IssuerId#44139, a\_IssuerId2#46287, a\_issueridcomposite#46303, sysdt#9003 AS a\_sysdt#46319, systs#9016, month#30928, year#30929]

: +- Project [BusinessDate#44187, a\_stcd#46162, a\_srcnm#46178, NetworkName#44155, NetworkURL#44171, a\_custnum#46224, a\_MarketCoverage#46240, a\_RejectRows#46256, IssuerId#44139, a\_IssuerId2#46287, issueridcomposite#8901 AS a\_issueridcomposite#46303, sysdt#9003, systs#9016, month#30928, year#30929]

: +- Project [BusinessDate#44187, a\_stcd#46162, a\_srcnm#46178, NetworkName#44155, NetworkURL#44171, a\_custnum#46224, a\_MarketCoverage#46240, a\_RejectRows#46256, IssuerId#44139, IssuerId2#8325 AS a\_IssuerId2#46287, issueridcomposite#8901, sysdt#9003, systs#9016, month#30928, year#30929]

: +- Project [BusinessDate#44187, a\_stcd#46162, a\_srcnm#46178, NetworkName#44155, NetworkURL#44171, a\_custnum#46224, a\_MarketCoverage#46240, RejectRows#1386 AS a\_RejectRows#46256, IssuerId#44139, IssuerId2#8325, issueridcomposite#8901, sysdt#9003, systs#9016, month#30928, year#30929]

: +- Project [BusinessDate#44187, a\_stcd#46162, a\_srcnm#46178, NetworkName#44155, NetworkURL#44171, a\_custnum#46224, MarketCoverage#1384 AS a\_MarketCoverage#46240, RejectRows#1386, IssuerId#44139, IssuerId2#8325, issueridcomposite#8901, sysdt#9003, systs#9016, month#30928, year#30929]

: +- Project [BusinessDate#44187, a\_stcd#46162, a\_srcnm#46178, NetworkName#44155, NetworkURL#44171, custnum#1383 AS a\_custnum#46224, MarketCoverage#1384, RejectRows#1386, IssuerId#44139, IssuerId2#8325, issueridcomposite#8901, sysdt#9003, systs#9016, month#30928, year#30929]

: +- Project [BusinessDate#44187, a\_stcd#46162, srcnm#7747 AS a\_srcnm#46178, NetworkName#44155, NetworkURL#44171, custnum#1383, MarketCoverage#1384, RejectRows#1386, IssuerId#44139, IssuerId2#8325, issueridcomposite#8901, sysdt#9003, systs#9016, month#30928, year#30929]

: +- Project [BusinessDate#44187, stcd#7759 AS a\_stcd#46162, srcnm#7747, NetworkName#44155, NetworkURL#44171, custnum#1383, MarketCoverage#1384, RejectRows#1386, IssuerId#44139, IssuerId2#8325, issueridcomposite#8901, sysdt#9003, systs#9016, month#30928, year#30929]

: +- Project [a\_BusinessDate#43854 AS BusinessDate#44187, stcd#7759, srcnm#7747, NetworkName#44155, NetworkURL#44171, custnum#1383, MarketCoverage#1384, RejectRows#1386, IssuerId#44139, IssuerId2#8325, issueridcomposite#8901, sysdt#9003, systs#9016, month#30928, year#30929]

: +- Project [a\_BusinessDate#43854, stcd#7759, srcnm#7747, NetworkName#44155, a\_NetworkURL#43806 AS NetworkURL#44171, custnum#1383, MarketCoverage#1384, RejectRows#1386, IssuerId#44139, IssuerId2#8325, issueridcomposite#8901, sysdt#9003, systs#9016, month#30928, year#30929]

: +- Project [a\_BusinessDate#43854, stcd#7759, srcnm#7747, a\_NetworkName#43822 AS NetworkName#44155, a\_NetworkURL#43806, custnum#1383, MarketCoverage#1384, RejectRows#1386, IssuerId#44139, IssuerId2#8325, issueridcomposite#8901, sysdt#9003, systs#9016, month#30928, year#30929]

: +- Project [a\_BusinessDate#43854, stcd#7759, srcnm#7747, a\_NetworkName#43822, a\_NetworkURL#43806, custnum#1383, MarketCoverage#1384, RejectRows#1386, a\_IssuerId#43838 AS IssuerId#44139, IssuerId2#8325, issueridcomposite#8901, sysdt#9003, systs#9016, month#30928, year#30929]

: +- Project [BusinessDate#1378 AS a\_BusinessDate#43854, stcd#7759, srcnm#7747, a\_NetworkName#43822, a\_NetworkURL#43806, custnum#1383, MarketCoverage#1384, RejectRows#1386, a\_IssuerId#43838, IssuerId2#8325, issueridcomposite#8901, sysdt#9003, systs#9016, month#30928, year#30929]

: +- Project [BusinessDate#1378, stcd#7759, srcnm#7747, a\_NetworkName#43822, a\_NetworkURL#43806, custnum#1383, MarketCoverage#1384, RejectRows#1386, IssuerId#8312 AS a\_IssuerId#43838, IssuerId2#8325, issueridcomposite#8901, sysdt#9003, systs#9016, month#30928, year#30929]

: +- Project [BusinessDate#1378, stcd#7759, srcnm#7747, NetworkName#1381 AS a\_NetworkName#43822, a\_NetworkURL#43806, custnum#1383, MarketCoverage#1384, RejectRows#1386, IssuerId#8312, IssuerId2#8325, issueridcomposite#8901, sysdt#9003, systs#9016, month#30928, year#30929]

: +- Project [BusinessDate#1378, stcd#7759, srcnm#7747, NetworkName#1381, NetworkURL#1382 AS a\_NetworkURL#43806, custnum#1383, MarketCoverage#1384, RejectRows#1386, IssuerId#8312, IssuerId2#8325, issueridcomposite#8901, sysdt#9003, systs#9016, month#30928, year#30929]

: +- Project [BusinessDate#1378, stcd#7759, srcnm#7747, NetworkName#1381, NetworkURL#1382, custnum#1383, MarketCoverage#1384, RejectRows#1386, IssuerId#8312, IssuerId2#8325, issueridcomposite#8901, sysdt#9003, systs#9016, month(BusinessDate#1378) AS month#30928, year(BusinessDate#1378) AS year#30929]

: +- SubqueryAlias insureview

: +- View (`insureview`, [BusinessDate#1378,stcd#7759,srcnm#7747,NetworkName#1381,NetworkURL#1382,custnum#1383,MarketCoverage#1384,RejectRows#1386,IssuerId#8312,IssuerId2#8325,issueridcomposite#8901,sysdt#9003,systs#9016])

: +- Project [BusinessDate#1378, stcd#7759, srcnm#7747, NetworkName#1381, NetworkURL#1382, custnum#1383, MarketCoverage#1384, RejectRows#1386, IssuerId#8312, IssuerId2#8325, issueridcomposite#8901, sysdt#9003, current\_timestamp() AS systs#9016]

: +- Project [BusinessDate#1378, stcd#7759, srcnm#7747, NetworkName#1381, NetworkURL#1382, custnum#1383, MarketCoverage#1384, RejectRows#1386, IssuerId#8312, IssuerId2#8325, issueridcomposite#8901, current\_date(Some(Asia/Kolkata)) AS sysdt#9003]

: +- Project [BusinessDate#1378, stcd#7759, srcnm#7747, NetworkName#1381, NetworkURL#1382, custnum#1383, MarketCoverage#1384, RejectRows#1386, IssuerId#8312, IssuerId2#8325, issueridcomposite#8901]

: +- Project [BusinessDate#1378, stcd#7759, srcnm#7747, NetworkName#1381, NetworkURL#1382, custnum#1383, MarketCoverage#1384, DentalOnlyPlan#1385, RejectRows#1386, IssuerId#8312, IssuerId2#8325, concat(IssuerId#8312, IssuerId2#8325) AS issueridcomposite#8901]

: +- Project [BusinessDate#1378, stcd#7759, srcnm#7747, NetworkName#1381, NetworkURL#1382, custnum#1383, MarketCoverage#1384, DentalOnlyPlan#1385, RejectRows#1386, IssuerId#8312, IssuerId2#8325]

: +- Project [IssuerIDDD#8273, BusinessDate#1378, stcd#7759, srcnm#7747, NetworkName#1381, NetworkURL#1382, custnum#1383, MarketCoverage#1384, DentalOnlyPlan#1385, RejectRows#1386, IssuerId#8312, IssuerId2#8325]

: +- Project [IssuerIDD#8261, IssuerIDDD#8273, BusinessDate#1378, stcd#7759, srcnm#7747, NetworkName#1381, NetworkURL#1382, custnum#1383, MarketCoverage#1384, DentalOnlyPlan#1385, RejectRows#1386, IssuerId#8312, cast(IssuerIDDD#8273 as string) AS IssuerId2#8325]

: +- Project [IssuerIDD#8261, IssuerIDDD#8273, BusinessDate#1378, stcd#7759, srcnm#7747, NetworkName#1381, NetworkURL#1382, custnum#1383, MarketCoverage#1384, DentalOnlyPlan#1385, RejectRows#1386, cast(IssuerIDD#8261 as string) AS IssuerId#8312]

: +- Project [IssuerIDD#8261, IssuerId2#1377 AS IssuerIDDD#8273, BusinessDate#1378, stcd#7759, srcnm#7747, NetworkName#1381, NetworkURL#1382, custnum#1383, MarketCoverage#1384, DentalOnlyPlan#1385, RejectRows#1386]

: +- Project [IssuerId#1376 AS IssuerIDD#8261, IssuerId2#1377, BusinessDate#1378, stcd#7759, srcnm#7747, NetworkName#1381, NetworkURL#1382, custnum#1383, MarketCoverage#1384, DentalOnlyPlan#1385, RejectRows#1386]

: +- Project [IssuerId#1376, IssuerId2#1377, BusinessDate#1378, StateCode#1379 AS stcd#7759, srcnm#7747, NetworkName#1381, NetworkURL#1382, custnum#1383, MarketCoverage#1384, DentalOnlyPlan#1385, RejectRows#1386]

: +- Project [IssuerId#1376, IssuerId2#1377, BusinessDate#1378, StateCode#1379, SourceName#1380 AS srcnm#7747, NetworkName#1381, NetworkURL#1382, custnum#1383, MarketCoverage#1384, DentalOnlyPlan#1385, RejectRows#1386]

: +- Union false, false

: :- Relation [IssuerId#1376,IssuerId2#1377,BusinessDate#1378,StateCode#1379,SourceName#1380,NetworkName#1381,NetworkURL#1382,custnum#1383,MarketCoverage#1384,DentalOnlyPlan#1385,RejectRows#1386] csv

: +- Relation [IssuerId#491,IssuerId2#492,BusinessDate#493,StateCode#494,SourceName#495,NetworkName#496,NetworkURL#497,custnum#498,MarketCoverage#499,DentalOnlyPlan#500,RejectRows#501] csv

+- Project [BusinessDate#44248, b\_stcd#46397, b\_srcnm#46412, NetworkName#44218, NetworkURL#44233, b\_custnum#46455, b\_MarketCoverage#46470, b\_RejectRows#46485, IssuerId#44203, b\_IssuerId2#46514, b\_issueridcomposite#46529, b\_sysdt#46544, b\_systs#46559, Protocol\_Used#30945 AS b\_Protocol\_Used#46574]

+- Project [BusinessDate#44248, b\_stcd#46397, b\_srcnm#46412, NetworkName#44218, NetworkURL#44233, b\_custnum#46455, b\_MarketCoverage#46470, b\_RejectRows#46485, IssuerId#44203, b\_IssuerId2#46514, b\_issueridcomposite#46529, b\_sysdt#46544, systs#9016 AS b\_systs#46559, Protocol\_Used#30945]

+- Project [BusinessDate#44248, b\_stcd#46397, b\_srcnm#46412, NetworkName#44218, NetworkURL#44233, b\_custnum#46455, b\_MarketCoverage#46470, b\_RejectRows#46485, IssuerId#44203, b\_IssuerId2#46514, b\_issueridcomposite#46529, sysdt#9003 AS b\_sysdt#46544, systs#9016, Protocol\_Used#30945]

+- Project [BusinessDate#44248, b\_stcd#46397, b\_srcnm#46412, NetworkName#44218, NetworkURL#44233, b\_custnum#46455, b\_MarketCoverage#46470, b\_RejectRows#46485, IssuerId#44203, b\_IssuerId2#46514, issueridcomposite#8901 AS b\_issueridcomposite#46529, sysdt#9003, systs#9016, Protocol\_Used#30945]

+- Project [BusinessDate#44248, b\_stcd#46397, b\_srcnm#46412, NetworkName#44218, NetworkURL#44233, b\_custnum#46455, b\_MarketCoverage#46470, b\_RejectRows#46485, IssuerId#44203, IssuerId2#8325 AS b\_IssuerId2#46514, issueridcomposite#8901, sysdt#9003, systs#9016, Protocol\_Used#30945]

+- Project [BusinessDate#44248, b\_stcd#46397, b\_srcnm#46412, NetworkName#44218, NetworkURL#44233, b\_custnum#46455, b\_MarketCoverage#46470, RejectRows#47041 AS b\_RejectRows#46485, IssuerId#44203, IssuerId2#8325, issueridcomposite#8901, sysdt#9003, systs#9016, Protocol\_Used#30945]

+- Project [BusinessDate#44248, b\_stcd#46397, b\_srcnm#46412, NetworkName#44218, NetworkURL#44233, b\_custnum#46455, MarketCoverage#47039 AS b\_MarketCoverage#46470, RejectRows#47041, IssuerId#44203, IssuerId2#8325, issueridcomposite#8901, sysdt#9003, systs#9016, Protocol\_Used#30945]

+- Project [BusinessDate#44248, b\_stcd#46397, b\_srcnm#46412, NetworkName#44218, NetworkURL#44233, custnum#47038 AS b\_custnum#46455, MarketCoverage#47039, RejectRows#47041, IssuerId#44203, IssuerId2#8325, issueridcomposite#8901, sysdt#9003, systs#9016, Protocol\_Used#30945]

+- Project [BusinessDate#44248, b\_stcd#46397, srcnm#7747 AS b\_srcnm#46412, NetworkName#44218, NetworkURL#44233, custnum#47038, MarketCoverage#47039, RejectRows#47041, IssuerId#44203, IssuerId2#8325, issueridcomposite#8901, sysdt#9003, systs#9016, Protocol\_Used#30945]

+- Project [BusinessDate#44248, stcd#7759 AS b\_stcd#46397, srcnm#7747, NetworkName#44218, NetworkURL#44233, custnum#47038, MarketCoverage#47039, RejectRows#47041, IssuerId#44203, IssuerId2#8325, issueridcomposite#8901, sysdt#9003, systs#9016, Protocol\_Used#30945]

+- Project [b\_BusinessDate#43915 AS BusinessDate#44248, stcd#7759, srcnm#7747, NetworkName#44218, NetworkURL#44233, custnum#47038, MarketCoverage#47039, RejectRows#47041, IssuerId#44203, IssuerId2#8325, issueridcomposite#8901, sysdt#9003, systs#9016, Protocol\_Used#30945]

+- Project [b\_BusinessDate#43915, stcd#7759, srcnm#7747, NetworkName#44218, b\_NetworkURL#43870 AS NetworkURL#44233, custnum#47038, MarketCoverage#47039, RejectRows#47041, IssuerId#44203, IssuerId2#8325, issueridcomposite#8901, sysdt#9003, systs#9016, Protocol\_Used#30945]

+- Project [b\_BusinessDate#43915, stcd#7759, srcnm#7747, b\_NetworkName#43885 AS NetworkName#44218, b\_NetworkURL#43870, custnum#47038, MarketCoverage#47039, RejectRows#47041, IssuerId#44203, IssuerId2#8325, issueridcomposite#8901, sysdt#9003, systs#9016, Protocol\_Used#30945]

+- Project [b\_BusinessDate#43915, stcd#7759, srcnm#7747, b\_NetworkName#43885, b\_NetworkURL#43870, custnum#47038, MarketCoverage#47039, RejectRows#47041, b\_IssuerId#43900 AS IssuerId#44203, IssuerId2#8325, issueridcomposite#8901, sysdt#9003, systs#9016, Protocol\_Used#30945]

+- Project [BusinessDate#47033 AS b\_BusinessDate#43915, stcd#7759, srcnm#7747, b\_NetworkName#43885, b\_NetworkURL#43870, custnum#47038, MarketCoverage#47039, RejectRows#47041, b\_IssuerId#43900, IssuerId2#8325, issueridcomposite#8901, sysdt#9003, systs#9016, Protocol\_Used#30945]

+- Project [BusinessDate#47033, stcd#7759, srcnm#7747, b\_NetworkName#43885, b\_NetworkURL#43870, custnum#47038, MarketCoverage#47039, RejectRows#47041, IssuerId#8312 AS b\_IssuerId#43900, IssuerId2#8325, issueridcomposite#8901, sysdt#9003, systs#9016, Protocol\_Used#30945]

+- Project [BusinessDate#47033, stcd#7759, srcnm#7747, NetworkName#47036 AS b\_NetworkName#43885, b\_NetworkURL#43870, custnum#47038, MarketCoverage#47039, RejectRows#47041, IssuerId#8312, IssuerId2#8325, issueridcomposite#8901, sysdt#9003, systs#9016, Protocol\_Used#30945]

+- Project [BusinessDate#47033, stcd#7759, srcnm#7747, NetworkName#47036, NetworkURL#47037 AS b\_NetworkURL#43870, custnum#47038, MarketCoverage#47039, RejectRows#47041, IssuerId#8312, IssuerId2#8325, issueridcomposite#8901, sysdt#9003, systs#9016, Protocol\_Used#30945]

+- Project [BusinessDate#47033, stcd#7759, srcnm#7747, NetworkName#47036, NetworkURL#47037, custnum#47038, MarketCoverage#47039, RejectRows#47041, IssuerId#8312, IssuerId2#8325, issueridcomposite#8901, sysdt#9003, systs#9016, CASE WHEN NetworkURL#47037 LIKE https://% THEN http secured WHEN NetworkURL#47037 LIKE http://% THEN http non secured ELSE No Protocol END AS Protocol\_Used#30945]

+- SubqueryAlias insureview

+- View (`insureview`, [BusinessDate#47033,stcd#7759,srcnm#7747,NetworkName#47036,NetworkURL#47037,custnum#47038,MarketCoverage#47039,RejectRows#47041,IssuerId#8312,IssuerId2#8325,issueridcomposite#8901,sysdt#9003,systs#9016])

+- Project [BusinessDate#47033, stcd#7759, srcnm#7747, NetworkName#47036, NetworkURL#47037, custnum#47038, MarketCoverage#47039, RejectRows#47041, IssuerId#8312, IssuerId2#8325, issueridcomposite#8901, sysdt#9003, current\_timestamp() AS systs#9016]

+- Project [BusinessDate#47033, stcd#7759, srcnm#7747, NetworkName#47036, NetworkURL#47037, custnum#47038, MarketCoverage#47039, RejectRows#47041, IssuerId#8312, IssuerId2#8325, issueridcomposite#8901, current\_date(Some(Asia/Kolkata)) AS sysdt#9003]

+- Project [BusinessDate#47033, stcd#7759, srcnm#7747, NetworkName#47036, NetworkURL#47037, custnum#47038, MarketCoverage#47039, RejectRows#47041, IssuerId#8312, IssuerId2#8325, issueridcomposite#8901]

+- Project [BusinessDate#47033, stcd#7759, srcnm#7747, NetworkName#47036, NetworkURL#47037, custnum#47038, MarketCoverage#47039, DentalOnlyPlan#47040, RejectRows#47041, IssuerId#8312, IssuerId2#8325, concat(IssuerId#8312, IssuerId2#8325) AS issueridcomposite#8901]

+- Project [BusinessDate#47033, stcd#7759, srcnm#7747, NetworkName#47036, NetworkURL#47037, custnum#47038, MarketCoverage#47039, DentalOnlyPlan#47040, RejectRows#47041, IssuerId#8312, IssuerId2#8325]

+- Project [IssuerIDDD#8273, BusinessDate#47033, stcd#7759, srcnm#7747, NetworkName#47036, NetworkURL#47037, custnum#47038, MarketCoverage#47039, DentalOnlyPlan#47040, RejectRows#47041, IssuerId#8312, IssuerId2#8325]

+- Project [IssuerIDD#8261, IssuerIDDD#8273, BusinessDate#47033, stcd#7759, srcnm#7747, NetworkName#47036, NetworkURL#47037, custnum#47038, MarketCoverage#47039, DentalOnlyPlan#47040, RejectRows#47041, IssuerId#8312, cast(IssuerIDDD#8273 as string) AS IssuerId2#8325]

+- Project [IssuerIDD#8261, IssuerIDDD#8273, BusinessDate#47033, stcd#7759, srcnm#7747, NetworkName#47036, NetworkURL#47037, custnum#47038, MarketCoverage#47039, DentalOnlyPlan#47040, RejectRows#47041, cast(IssuerIDD#8261 as string) AS IssuerId#8312]

+- Project [IssuerIDD#8261, IssuerId2#47032 AS IssuerIDDD#8273, BusinessDate#47033, stcd#7759, srcnm#7747, NetworkName#47036, NetworkURL#47037, custnum#47038, MarketCoverage#47039, DentalOnlyPlan#47040, RejectRows#47041]

+- Project [IssuerId#47031 AS IssuerIDD#8261, IssuerId2#47032, BusinessDate#47033, stcd#7759, srcnm#7747, NetworkName#47036, NetworkURL#47037, custnum#47038, MarketCoverage#47039, DentalOnlyPlan#47040, RejectRows#47041]

+- Project [IssuerId#47031, IssuerId2#47032, BusinessDate#47033, StateCode#47034 AS stcd#7759, srcnm#7747, NetworkName#47036, NetworkURL#47037, custnum#47038, MarketCoverage#47039, DentalOnlyPlan#47040, RejectRows#47041]

+- Project [IssuerId#47031, IssuerId2#47032, BusinessDate#47033, StateCode#47034, SourceName#47035 AS srcnm#7747, NetworkName#47036, NetworkURL#47037, custnum#47038, MarketCoverage#47039, DentalOnlyPlan#47040, RejectRows#47041]

+- Union false, false

:- Relation [IssuerId#47031,IssuerId2#47032,BusinessDate#47033,StateCode#47034,SourceName#47035,NetworkName#47036,NetworkURL#47037,custnum#47038,MarketCoverage#47039,DentalOnlyPlan#47040,RejectRows#47041] csv

+- Relation [IssuerId#47042,IssuerId2#47043,BusinessDate#47044,StateCode#47045,SourceName#47046,NetworkName#47047,NetworkURL#47048,custnum#47049,MarketCoverage#47050,DentalOnlyPlan#47051,RejectRows#47052] csv

>>>

>>> # Perform inner join on all the renamed DataFrames

... df\_combined = (

... result\_df\_SQL\_renamed

... .join(result\_df\_SQL\_1\_renamed, on=['BusinessDate', 'stcd', 'srcnm', 'custnum', 'MarketCoverage', 'RejectRows', 'IssuerId', 'IssuerId2', 'issueridcomposite', 'sysdt', 'systs'], how='inner')

... .join(result\_df\_SQL\_2\_renamed, on=['BusinessDate', 'stcd', 'srcnm', 'custnum', 'MarketCoverage', 'RejectRows', 'IssuerId', 'IssuerId2', 'issueridcomposite', 'sysdt', 'systs'], how='inner')

... .join(result\_df\_SQL\_3\_renamed, on=['BusinessDate', 'stcd', 'srcnm', 'custnum', 'MarketCoverage', 'RejectRows', 'IssuerId', 'IssuerId2', 'issueridcomposite', 'sysdt', 'systs'], how='inner')

... )

Traceback (most recent call last):

File "<stdin>", line 4, in <module>

File "/usr/local/spark/python/pyspark/sql/dataframe.py", line 1355, in join

jdf = self.\_jdf.join(other.\_jdf, on, how)

File "/usr/local/spark/python/lib/py4j-0.10.9.3-src.zip/py4j/java\_gateway.py", line 1322, in \_\_call\_\_

File "/usr/local/spark/python/pyspark/sql/utils.py", line 117, in deco

raise converted from None

pyspark.sql.utils.AnalysisException: USING column `stcd` cannot be resolved on the left side of the join. The left-side columns: [BusinessDate, a\_stcd, a\_srcnm, NetworkName, NetworkURL, a\_custnum, a\_MarketCoverage, a\_RejectRows, IssuerId, a\_IssuerId2, a\_issueridcomposite, a\_sysdt, a\_systs, a\_month, a\_year]

>>>

>>> # Perform inner join on all the renamed DataFrames

... df\_combined = (

... result\_df\_SQL\_renamed

... .join(result\_df\_SQL\_1\_renamed,

... on=['a\_BusinessDate', 'a\_stcd', 'a\_srcnm', 'a\_custnum', 'a\_MarketCoverage', 'a\_RejectRows', 'a\_IssuerId', 'a\_IssuerId2', 'a\_issueridcomposite', 'a\_sysdt', 'a\_systs'],

... how='inner')

... .join(result\_df\_SQL\_2\_renamed,

... on=['a\_BusinessDate', 'a\_stcd', 'a\_srcnm', 'a\_custnum', 'a\_MarketCoverage', 'a\_RejectRows', 'a\_IssuerId', 'a\_IssuerId2', 'a\_issueridcomposite', 'a\_sysdt', 'a\_systs'],

... how='inner')

... .join(result\_df\_SQL\_3\_renamed,

... on=['a\_BusinessDate', 'a\_stcd', 'a\_srcnm', 'a\_custnum', 'a\_MarketCoverage', 'a\_RejectRows', 'a\_IssuerId', 'a\_IssuerId2', 'a\_issueridcomposite', 'a\_sysdt', 'a\_systs'],

... how='inner')

... )

Traceback (most recent call last):

File "<stdin>", line 6, in <module>

File "/usr/local/spark/python/pyspark/sql/dataframe.py", line 1355, in join

jdf = self.\_jdf.join(other.\_jdf, on, how)

File "/usr/local/spark/python/lib/py4j-0.10.9.3-src.zip/py4j/java\_gateway.py", line 1322, in \_\_call\_\_

File "/usr/local/spark/python/pyspark/sql/utils.py", line 117, in deco

raise converted from None

pyspark.sql.utils.AnalysisException: USING column `a\_BusinessDate` cannot be resolved on the left side of the join. The left-side columns: [BusinessDate, a\_stcd, a\_srcnm, NetworkName, NetworkURL, a\_custnum, a\_MarketCoverage, a\_RejectRows, IssuerId, a\_IssuerId2, a\_issueridcomposite, a\_sysdt, a\_systs, a\_month, a\_year]

>>> c

>>> print("result\_df\_SQL\_renamed columns:", result\_df\_SQL\_renamed.columns)

result\_df\_SQL\_renamed columns: ['BusinessDate', 'a\_stcd', 'a\_srcnm', 'NetworkName', 'NetworkURL', 'a\_custnum', 'a\_MarketCoverage', 'a\_RejectRows', 'IssuerId', 'a\_IssuerId2', 'a\_issueridcomposite', 'a\_sysdt', 'a\_systs', 'a\_month', 'a\_year']

>>> print("result\_df\_SQL\_1\_renamed columns:", result\_df\_SQL\_1\_renamed.columns)

result\_df\_SQL\_1\_renamed columns: ['BusinessDate', 'b\_stcd', 'b\_srcnm', 'NetworkName', 'NetworkURL', 'b\_custnum', 'b\_MarketCoverage', 'b\_RejectRows', 'IssuerId', 'b\_IssuerId2', 'b\_issueridcomposite', 'b\_sysdt', 'b\_systs', 'b\_Protocol\_Used']

>>> print("result\_df\_SQL\_2\_renamed columns:", result\_df\_SQL\_2\_renamed.columns)

result\_df\_SQL\_2\_renamed columns: ['BusinessDate', 'c\_stcd', 'c\_srcnm', 'NetworkName', 'NetworkURL', 'c\_custnum', 'c\_MarketCoverage', 'c\_RejectRows', 'IssuerId', 'c\_IssuerId2', 'c\_issueridcomposite', 'c\_sysdt', 'c\_systs', 'c\_curdt', 'c\_curts']

>>> print("result\_df\_SQL\_3\_renamed columns:", result\_df\_SQL\_3\_renamed.columns)

result\_df\_SQL\_3\_renamed columns: ['BusinessDate', 'd\_stcd', 'd\_srcnm', 'NetworkName', 'NetworkURL', 'd\_custnum', 'd\_MarketCoverage', 'd\_RejectRows', 'IssuerId', 'd\_IssuerId2', 'd\_issueridcomposite', 'd\_sysdt', 'd\_systs', 'd\_cleannetworkname']

>>>

>>> # Perform inner join on all the renamed DataFrames

... df\_combined = (

... result\_df\_SQL\_renamed

... .join(result\_df\_SQL\_1\_renamed,

... on=['BusinessDate', 'a\_stcd', 'a\_srcnm', 'a\_custnum', 'a\_MarketCoverage', 'a\_RejectRows', 'IssuerId', 'a\_IssuerId2', 'a\_issueridcomposite', 'a\_sysdt', 'a\_systs'],

... how='inner')

... .join(result\_df\_SQL\_2\_renamed,

... on=['BusinessDate', 'a\_stcd', 'a\_srcnm', 'a\_custnum', 'a\_MarketCoverage', 'a\_RejectRows', 'IssuerId', 'a\_IssuerId2', 'a\_issueridcomposite', 'a\_sysdt', 'a\_systs'],

... how='inner')

... .join(result\_df\_SQL\_3\_renamed,

... on=['BusinessDate', 'a\_stcd', 'a\_srcnm', 'a\_custnum', 'a\_MarketCoverage', 'a\_RejectRows', 'IssuerId', 'a\_IssuerId2', 'a\_issueridcomposite', 'a\_sysdt', 'a\_systs'],

... how='inner')

... )

Traceback (most recent call last):

File "<stdin>", line 6, in <module>

File "/usr/local/spark/python/pyspark/sql/dataframe.py", line 1355, in join

jdf = self.\_jdf.join(other.\_jdf, on, how)

File "/usr/local/spark/python/lib/py4j-0.10.9.3-src.zip/py4j/java\_gateway.py", line 1322, in \_\_call\_\_

File "/usr/local/spark/python/pyspark/sql/utils.py", line 117, in deco

raise converted from None

pyspark.sql.utils.AnalysisException: USING column `a\_stcd` cannot be resolved on the right side of the join. The right-side columns: [BusinessDate, b\_stcd, b\_srcnm, NetworkName, NetworkURL, b\_custnum, b\_MarketCoverage, b\_RejectRows, IssuerId, b\_IssuerId2, b\_issueridcomposite, b\_sysdt, b\_systs, b\_Protocol\_Used]

>>>

>>> df\_combined = result\_df\_SQL\_renamed \

... .join(result\_df\_SQL\_1\_renamed, on="a\_IssuerId2", how="inner") \

... .join(result\_df\_SQL\_2\_renamed, on="a\_IssuerId2", how="inner") \

... .join(result\_df\_SQL\_3\_renamed, on="a\_IssuerId2", how="inner")

Traceback (most recent call last):

File "<stdin>", line 2, in <module>

File "/usr/local/spark/python/pyspark/sql/dataframe.py", line 1355, in join

jdf = self.\_jdf.join(other.\_jdf, on, how)

File "/usr/local/spark/python/lib/py4j-0.10.9.3-src.zip/py4j/java\_gateway.py", line 1322, in \_\_call\_\_

File "/usr/local/spark/python/pyspark/sql/utils.py", line 117, in deco

raise converted from None

pyspark.sql.utils.AnalysisException: USING column `a\_IssuerId2` cannot be resolved on the right side of the join. The right-side columns: [BusinessDate, b\_stcd, b\_srcnm, NetworkName, NetworkURL, b\_custnum, b\_MarketCoverage, b\_RejectRows, IssuerId, b\_IssuerId2, b\_issueridcomposite, b\_sysdt, b\_systs, b\_Protocol\_Used]

>>> result\_df\_SQL\_renamed.describe

<bound method DataFrame.describe of DataFrame[BusinessDate: date, a\_stcd: string, a\_srcnm: string, NetworkName: string, NetworkURL: string, a\_custnum: string, a\_MarketCoverage: string, a\_RejectRows: string, IssuerId: string, a\_IssuerId2: string, a\_issueridcomposite: string, a\_sysdt: date, a\_systs: timestamp, a\_month: int, a\_year: int]>

>>> .withColumnRenamed("curdt", "c\_curdt") \spark.sql("select \*,sql\_function(NetworkName) as cleannetworkname from insureview")

>>> spark.sql("select \*,sql\_function(NetworkName) as cleannetworkname from insureview").show(1)

+------------+----+-----+-----------+--------------------+-------+--------------+----------+--------+---------+-----------------+----------+--------------------+----------------+

|BusinessDate|stcd|srcnm|NetworkName| NetworkURL|custnum|MarketCoverage|RejectRows|IssuerId|IssuerId2|issueridcomposite| sysdt| systs|cleannetworkname|

+------------+----+-----+-----------+--------------------+-------+--------------+----------+--------+---------+-----------------+----------+--------------------+----------------+

| 2019-10-01| AK| HIOS|ODS Premier|https://www.modah...| 13| null| null| 21989| 21989| 2198921989|2024-07-22|2024-07-22 14:57:...| ODS Premier|

+------------+----+-----+-----------+--------------------+-------+--------------+----------+--------+---------+-----------------+----------+--------------------+----------------+

only showing top 1 row

>>> result\_df=spark.sql("select \*,sql\_function(NetworkName) as cleannetworkname from insureview")

>>> result\_df.createOrReplaceTempView("insureview")

Traceback (most recent call last):

File "<stdin>", line 1, in <module>

File "/usr/local/spark/python/pyspark/sql/dataframe.py", line 188, in createOrReplaceTempView

self.\_jdf.createOrReplaceTempView(name)

File "/usr/local/spark/python/lib/py4j-0.10.9.3-src.zip/py4j/java\_gateway.py", line 1322, in \_\_call\_\_

File "/usr/local/spark/python/pyspark/sql/utils.py", line 117, in deco

raise converted from None

pyspark.sql.utils.AnalysisException: Recursive view `insureview` detected (cycle: `insureview` -> `insureview`)

>>> spark.catalog.dropTempView("insureview")

>>> result\_df.createOrReplaceTempView("insureview")

>>> result\_df=spark.sql("select \*,current\_date as curdt , current\_timestamp as curts from insureview")

>>> spark.catalog.dropTempView("insureview")

>>> result\_df.createOrReplaceTempView("insureview")

>>> result\_df=spark.sql("select \*,sql\_function(NetworkName) as cleannetworkname from insureview")

>>> spark.sql("""select \*,case when NetworkURL like 'https://%' then 'http secured' when NetworkURL like 'http://%' then 'http non secured' else 'No Protocol' end as Protocol\_Used from insureview""").show(1)

+------------+----+-----+-----------+--------------------+-------+--------------+----------+--------+---------+-----------------+----------+--------------------+----------------+----------+--------------------+-------------+

|BusinessDate|stcd|srcnm|NetworkName| NetworkURL|custnum|MarketCoverage|RejectRows|IssuerId|IssuerId2|issueridcomposite| sysdt| systs|cleannetworkname| curdt| curts|Protocol\_Used|

+------------+----+-----+-----------+--------------------+-------+--------------+----------+--------+---------+-----------------+----------+--------------------+----------------+----------+--------------------+-------------+

| 2019-10-01| AK| HIOS|ODS Premier|https://www.modah...| 13| null| null| 21989| 21989| 2198921989|2024-07-22|2024-07-22 15:09:...| ODS Premier|2024-07-22|2024-07-22 15:09:...| http secured|

+------------+----+-----+-----------+--------------------+-------+--------------+----------+--------+---------+-----------------+----------+--------------------+----------------+----------+--------------------+-------------+

only showing top 1 row

>>> spark.catalog.dropTempView("insureview")

>>> result\_df.createOrReplaceTempView("insureview")

>>> result\_df=spark.sql("""select \*,case when NetworkURL like 'https://%' then 'http secured' when NetworkURL like 'http://%' then 'http non secured' else 'No Protocol' end as Protocol\_Used from insureview""")

>>>

>>> result\_df.show(1)

+------------+----+-----+-----------+--------------------+-------+--------------+----------+--------+---------+-----------------+----------+--------------------+----------------+----------+--------------------+----------------+-------------+

|BusinessDate|stcd|srcnm|NetworkName| NetworkURL|custnum|MarketCoverage|RejectRows|IssuerId|IssuerId2|issueridcomposite| sysdt| systs|cleannetworkname| curdt| curts|cleannetworkname|Protocol\_Used|

+------------+----+-----+-----------+--------------------+-------+--------------+----------+--------+---------+-----------------+----------+--------------------+----------------+----------+--------------------+----------------+-------------+

| 2019-10-01| AK| HIOS|ODS Premier|https://www.modah...| 13| null| null| 21989| 21989| 2198921989|2024-07-22|2024-07-22 15:11:...| ODS Premier|2024-07-22|2024-07-22 15:11:...| ODS Premier| http secured|

+------------+----+-----+-----------+--------------------+-------+--------------+----------+--------+---------+-----------------+----------+--------------------+----------------+----------+--------------------+----------------+-------------+

only showing top 1 row

>>> result\_df.describe

>>> result\_df.describe

<bound method DataFrame.describe of DataFrame[BusinessDate: date, stcd: string, srcnm: string, NetworkName: string, NetworkURL: string, custnum: string, MarketCoverage: string, RejectRows: string, IssuerId: string, IssuerId2: string, issueridcomposite: string, sysdt: date, systs: timestamp, cleannetworkname: string, curdt: date, curts: timestamp, cleannetworkname: string, Protocol\_Used: string]>

>>> statefilter\_df.describe

<bound method DataFrame.describe of DataFrame[\_c0: string, \_c1: string]>

>>> spark.sql("select \* from stateview").show(2)

Traceback (most recent call last):

File "<stdin>", line 1, in <module>

File "/usr/local/spark/python/pyspark/sql/session.py", line 723, in sql

return DataFrame(self.\_jsparkSession.sql(sqlQuery), self.\_wrapped)

File "/usr/local/spark/python/lib/py4j-0.10.9.3-src.zip/py4j/java\_gateway.py", line 1322, in \_\_call\_\_

File "/usr/local/spark/python/pyspark/sql/utils.py", line 117, in deco

raise converted from None

pyspark.sql.utils.AnalysisException: Table or view not found: stateview; line 1 pos 14;

'Project [\*]

+- 'UnresolvedRelation [stateview], [], false

>>> statefilter\_df.show(1)

+---+----------+

|\_c0| \_c1|

+---+----------+

| NM|New Mexico|

+---+----------+

only showing top 1 row

>>> spark.sql("select \* from cust\_table").show(2)

+---+--------+-----+---+-------+

|\_c0| \_c1| \_c2|\_c3| \_c4|

+---+--------+-----+---+-------+

| 1|Kristina|Chung| 55| Pilot|

| 2| Paige| Chen| 74|Teacher|

+---+--------+-----+---+-------+

only showing top 2 rows

>>> spark.sql("select \* from state\_table").show(2)

+---+----------+

|\_c0| \_c1|

+---+----------+

| NM|New Mexico|

| NY| New York|

+---+----------+

only showing top 2 rows

>>> spark.sql("select \* from insureview").show(2)

+------------+----+-----+-----------+--------------------+-------+--------------+--------------------+--------+---------+-----------------+----------+--------------------+----------------+----------+--------------------+----------------+

|BusinessDate|stcd|srcnm|NetworkName| NetworkURL|custnum|MarketCoverage| RejectRows|IssuerId|IssuerId2|issueridcomposite| sysdt| systs|cleannetworkname| curdt| curts|cleannetworkname|

+------------+----+-----+-----------+--------------------+-------+--------------+--------------------+--------+---------+-----------------+----------+--------------------+----------------+----------+--------------------+----------------+

| 2019-10-01| AK| HIOS|ODS Premier|https://www.modah...| 13| null| null| 21989| 21989| 2198921989|2024-07-22|2024-07-22 15:31:...| ODS Premier|2024-07-22|2024-07-22 15:31:...| ODS Premier|

| 2019-10-01| AK| HIOS|ODS Premier| null| null| null|219,219,2019-10-0...| 219| 219| 219219|2024-07-22|2024-07-22 15:31:...| ODS Premier|2024-07-22|2024-07-22 15:31:...| ODS Premier|

+------------+----+-----+-----------+--------------------+-------+--------------+--------------------+--------+---------+-----------------+----------+--------------------+----------------+----------+--------------------+----------------+

only showing top 2 rows

>>> spark.sql("select \* from insureview as l join state\_table as r on l.stcd=r.\_c0 ").show(1)

+------------+----+-----+-----------+--------------------+-------+--------------+----------+--------+---------+-----------------+----------+--------------------+----------------+----------+--------------------+----------------+---+------+

|BusinessDate|stcd|srcnm|NetworkName| NetworkURL|custnum|MarketCoverage|RejectRows|IssuerId|IssuerId2|issueridcomposite| sysdt| systs|cleannetworkname| curdt| curts|cleannetworkname|\_c0| \_c1|

+------------+----+-----+-----------+--------------------+-------+--------------+----------+--------+---------+-----------------+----------+--------------------+----------------+----------+--------------------+----------------+---+------+

| 2019-10-01| AK| HIOS|ODS Premier|https://www.modah...| 13| null| null| 21989| 21989| 2198921989|2024-07-22|2024-07-22 15:35:...| ODS Premier|2024-07-22|2024-07-22 15:35:...| ODS Premier| AK|Alaska|

+------------+----+-----+-----------+--------------------+-------+--------------+----------+--------+---------+-----------------+----------+--------------------+----------------+----------+--------------------+----------------+---+------+

only showing top 1 row

>>> first\_join=spark.sql("select \* from insureview as l join state\_table as r on l.stcd=r.\_c0 ")

>>> spark.catalog.dropTempview("insureview")

Traceback (most recent call last):

File "<stdin>", line 1, in <module>

AttributeError: 'Catalog' object has no attribute 'dropTempview'

>>> spark.catelog.dropTempview("insureview")

Traceback (most recent call last):

File "<stdin>", line 1, in <module>

AttributeError: 'SparkSession' object has no attribute 'catelog'

>>> spark.catalog.dropTempview("insureview")

Traceback (most recent call last):

File "<stdin>", line 1, in <module>

AttributeError: 'Catalog' object has no attribute 'dropTempview'

>>> spark.catalog.dropTempView("insureview")

>>> first\_join.describe

<bound method DataFrame.describe of DataFrame[BusinessDate: date, stcd: string, srcnm: string, NetworkName: string, NetworkURL: string, custnum: string, MarketCoverage: string, RejectRows: string, IssuerId: string, IssuerId2: string, issueridcomposite: string, sysdt: date, systs: timestamp, cleannetworkname: string, curdt: date, curts: timestamp, cleannetworkname: string, \_c0: string, \_c1: string]>

>>> first\_join.createOrReplaceTempView("insureview")

>>> spark.sql("select \* from insureview").show(1)

+------------+----+-----+-----------+--------------------+-------+--------------+----------+--------+---------+-----------------+----------+--------------------+----------------+----------+--------------------+----------------+---+------+

|BusinessDate|stcd|srcnm|NetworkName| NetworkURL|custnum|MarketCoverage|RejectRows|IssuerId|IssuerId2|issueridcomposite| sysdt| systs|cleannetworkname| curdt| curts|cleannetworkname|\_c0| \_c1|

+------------+----+-----+-----------+--------------------+-------+--------------+----------+--------+---------+-----------------+----------+--------------------+----------------+----------+--------------------+----------------+---+------+

| 2019-10-01| AK| HIOS|ODS Premier|https://www.modah...| 13| null| null| 21989| 21989| 2198921989|2024-07-22|2024-07-22 15:43:...| ODS Premier|2024-07-22|2024-07-22 15:43:...| ODS Premier| AK|Alaska|

+------------+----+-----+-----------+--------------------+-------+--------------+----------+--------+---------+-----------------+----------+--------------------+----------------+----------+--------------------+----------------+---+------+

only showing top 1 row

>>> spark.sql("select \* from insureview as l join cust\_table as r on l.custnum=r.\_c3").show(1)

+------------+----+-----+-----------+--------------------+-------+--------------+----------+--------+---------+-----------------+----------+--------------------+----------------+----------+--------------------+----------------+---+-------+---+-----+-------+---+-----+

|BusinessDate|stcd|srcnm|NetworkName| NetworkURL|custnum|MarketCoverage|RejectRows|IssuerId|IssuerId2|issueridcomposite| sysdt| systs|cleannetworkname| curdt| curts|cleannetworkname|\_c0| \_c1|\_c0| \_c1| \_c2|\_c3| \_c4|

+------------+----+-----+-----------+--------------------+-------+--------------+----------+--------+---------+-----------------+----------+--------------------+----------------+----------+--------------------+----------------+---+-------+---+-----+-------+---+-----+

| 2019-10-01| AZ| HIOS| La Paz|http://uamarketpl...| 21| null| null| 88925| 88925| 8892588925|2024-07-22|2024-07-22 15:48:...| La Paz|2024-07-22|2024-07-22 15:48:...| La Paz| AZ|Arizona|779|Jason|Bowling| 21|Coach|

+------------+----+-----+-----------+--------------------+-------+--------------+----------+--------+---------+-----------------+----------+--------------------+----------------+----------+--------------------+----------------+---+-------+---+-----+-------+---+-----+

only showing top 1 row

>>> spark.sql("select \* from insureview as l join cust\_table as r on l.custnum=r.\_c3").show(4)

+------------+----+-----+-----------+--------------------+-------+--------------+----------+--------+---------+-----------------+----------+--------------------+----------------+----------+--------------------+----------------+---+-------+---+-------+-------+---+----------+

|BusinessDate|stcd|srcnm|NetworkName| NetworkURL|custnum|MarketCoverage|RejectRows|IssuerId|IssuerId2|issueridcomposite| sysdt| systs|cleannetworkname| curdt| curts|cleannetworkname|\_c0| \_c1|\_c0| \_c1| \_c2|\_c3| \_c4|

+------------+----+-----+-----------+--------------------+-------+--------------+----------+--------+---------+-----------------+----------+--------------------+----------------+----------+--------------------+----------------+---+-------+---+-------+-------+---+----------+

| 2019-10-01| AZ| HIOS| La Paz|http://uamarketpl...| 21| null| null| 88925| 88925| 8892588925|2024-07-22|2024-07-22 15:49:...| La Paz|2024-07-22|2024-07-22 15:49:...| La Paz| AZ|Arizona|779| Jason|Bowling| 21| Coach|

| 2019-10-01| AZ| HIOS| La Paz|http://uamarketpl...| 21| null| null| 88925| 88925| 8892588925|2024-07-22|2024-07-22 15:49:...| La Paz|2024-07-22|2024-07-22 15:49:...| La Paz| AZ|Arizona|765| Janice| Bowden| 21|Politician|

| 2019-10-01| AZ| HIOS| La Paz|http://uamarketpl...| 21| null| null| 88925| 88925| 8892588925|2024-07-22|2024-07-22 15:49:...| La Paz|2024-07-22|2024-07-22 15:49:...| La Paz| AZ|Arizona|465| Peter|Guthrie| 21| Nurse|

| 2019-10-01| AZ| HIOS| La Paz|http://uamarketpl...| 21| null| null| 88925| 88925| 8892588925|2024-07-22|2024-07-22 15:49:...| La Paz|2024-07-22|2024-07-22 15:49:...| La Paz| AZ|Arizona|451|Russell| Hess| 21| Architect|

+------------+----+-----+-----------+--------------------+-------+--------------+----------+--------+---------+-----------------+----------+--------------------+----------------+----------+--------------------+----------------+---+-------+---+-------+-------+---+----------+

only showing top 4 rows

>>> second\_join=spark.sql("select \* from insureview as l join cust\_table as r on l.custnum=r.\_c3")

>>> second\_join.count()

684

>>> second\_join.write.parquet("/user/hduser/Hackathon/PARQUET\_DATA")

Traceback (most recent call last):

File "<stdin>", line 1, in <module>

File "/usr/local/spark/python/pyspark/sql/readwriter.py", line 885, in parquet

self.\_jwrite.parquet(path)

File "/usr/local/spark/python/lib/py4j-0.10.9.3-src.zip/py4j/java\_gateway.py", line 1322, in \_\_call\_\_

File "/usr/local/spark/python/pyspark/sql/utils.py", line 117, in deco

raise converted from None

pyspark.sql.utils.AnalysisException: Found duplicate column(s) when inserting into hdfs://localhost:54310/user/hduser/Hackathon/PARQUET\_DATA: `\_c0`, `\_c1`, `cleannetworkname`

>>> second\_join.show()

+------------+----+-----+-----------+--------------------+-------+--------------+----------+--------+---------+-----------------+----------+--------------------+----------------+----------+--------------------+----------------+---+-------+---+--------+---------+---+--------------------+

|BusinessDate|stcd|srcnm|NetworkName| NetworkURL|custnum|MarketCoverage|RejectRows|IssuerId|IssuerId2|issueridcomposite| sysdt| systs|cleannetworkname| curdt| curts|cleannetworkname|\_c0| \_c1|\_c0| \_c1| \_c2|\_c3| \_c4|

+------------+----+-----+-----------+--------------------+-------+--------------+----------+--------+---------+-----------------+----------+--------------------+----------------+----------+--------------------+----------------+---+-------+---+--------+---------+---+--------------------+

| 2019-10-01| AZ| HIOS| La Paz|http://uamarketpl...| 21| null| null| 88925| 88925| 8892588925|2024-07-22|2024-07-22 15:57:...| La Paz|2024-07-22|2024-07-22 15:57:...| La Paz| AZ|Arizona|779| Jason| Bowling| 21| Coach|

| 2019-10-01| AZ| HIOS| La Paz|http://uamarketpl...| 21| null| null| 88925| 88925| 8892588925|2024-07-22|2024-07-22 15:57:...| La Paz|2024-07-22|2024-07-22 15:57:...| La Paz| AZ|Arizona|765| Janice| Bowden| 21| Politician|

| 2019-10-01| AZ| HIOS| La Paz|http://uamarketpl...| 21| null| null| 88925| 88925| 8892588925|2024-07-22|2024-07-22 15:57:...| La Paz|2024-07-22|2024-07-22 15:57:...| La Paz| AZ|Arizona|465| Peter| Guthrie| 21| Nurse|

| 2019-10-01| AZ| HIOS| La Paz|http://uamarketpl...| 21| null| null| 88925| 88925| 8892588925|2024-07-22|2024-07-22 15:57:...| La Paz|2024-07-22|2024-07-22 15:57:...| La Paz| AZ|Arizona|451| Russell| Hess| 21| Architect|

| 2019-10-01| AZ| HIOS| La Paz|http://uamarketpl...| 21| null| null| 88925| 88925| 8892588925|2024-07-22|2024-07-22 15:57:...| La Paz|2024-07-22|2024-07-22 15:57:...| La Paz| AZ|Arizona|349| Russell| Dalton| 21| Civil engineer|

| 2019-10-01| AZ| HIOS| La Paz|http://uamarketpl...| 21| null| null| 88925| 88925| 8892588925|2024-07-22|2024-07-22 15:57:...| La Paz|2024-07-22|2024-07-22 15:57:...| La Paz| AZ|Arizona|292| Donna| Rice| 21| Social worker|

| 2019-10-01| AZ| HIOS| La Paz|http://uamarketpl...| 21| null| null| 88925| 88925| 8892588925|2024-07-22|2024-07-22 15:57:...| La Paz|2024-07-22|2024-07-22 15:57:...| La Paz| AZ|Arizona|290| Gail|Whitehead| 21| Nurse|

| 2019-10-01| AZ| HIOS| La Paz|http://uamarketpl...| 21| null| null| 88925| 88925| 8892588925|2024-07-22|2024-07-22 15:57:...| La Paz|2024-07-22|2024-07-22 15:57:...| La Paz| AZ|Arizona|176| Kristin|Alexander| 21| Coach|

| 2019-10-01| AZ| HIOS| La Paz|http://uamarketpl...| 21| null| null| 88925| 88925| 8892588925|2024-07-22|2024-07-22 15:57:...| La Paz|2024-07-22|2024-07-22 15:57:...| La Paz| AZ|Arizona| 52| Shirley| Merritt| 21| Reporter|

| 2019-10-01| AZ| HIOS| Maricopa|http://uamarketpl...| 22| null| null| 88925| 88925| 8892588925|2024-07-22|2024-07-22 15:57:...| Maricopa|2024-07-22|2024-07-22 15:57:...| Maricopa| AZ|Arizona|545|Danielle| Smith| 22| Pharmacist|

| 2019-10-01| AZ| HIOS| Maricopa|http://uamarketpl...| 22| null| null| 88925| 88925| 8892588925|2024-07-22|2024-07-22 15:57:...| Maricopa|2024-07-22|2024-07-22 15:57:...| Maricopa| AZ|Arizona|472| Kristin| Drake| 22| Financial analyst|

| 2019-10-01| AZ| HIOS| Maricopa|http://uamarketpl...| 22| null| null| 88925| 88925| 8892588925|2024-07-22|2024-07-22 15:57:...| Maricopa|2024-07-22|2024-07-22 15:57:...| Maricopa| AZ|Arizona|454|Virginia| Turner| 22|Computer hardware...|

| 2019-10-01| AZ| HIOS| Maricopa|http://uamarketpl...| 22| null| null| 88925| 88925| 8892588925|2024-07-22|2024-07-22 15:57:...| Maricopa|2024-07-22|2024-07-22 15:57:...| Maricopa| AZ|Arizona|359| Leon| Bender| 22| Firefighter|

| 2019-10-01| AZ| HIOS| Maricopa|http://uamarketpl...| 22| null| null| 88925| 88925| 8892588925|2024-07-22|2024-07-22 15:57:...| Maricopa|2024-07-22|2024-07-22 15:57:...| Maricopa| AZ|Arizona|218| Brett| Snyder| 22| Doctor|

| 2019-10-01| AZ| HIOS| Maricopa|http://uamarketpl...| 22| null| null| 88925| 88925| 8892588925|2024-07-22|2024-07-22 15:57:...| Maricopa|2024-07-22|2024-07-22 15:57:...| Maricopa| AZ|Arizona|152| Vincent| Goldman| 22| Electrician|

| 2019-10-01| AZ| HIOS| Maricopa|http://uamarketpl...| 22| null| null| 88925| 88925| 8892588925|2024-07-22|2024-07-22 15:57:...| Maricopa|2024-07-22|2024-07-22 15:57:...| Maricopa| AZ|Arizona| 54| Judith| Cooper| 22| Economist|

| 2019-10-01| AZ| HIOS| Mohave|http://uamarketpl...| 23| null| null| 88925| 88925| 8892588925|2024-07-22|2024-07-22 15:57:...| Mohave|2024-07-22|2024-07-22 15:57:...| Mohave| AZ|Arizona|752| Ethel| Stevens| 23| Designer|

| 2019-10-01| AZ| HIOS| Mohave|http://uamarketpl...| 23| null| null| 88925| 88925| 8892588925|2024-07-22|2024-07-22 15:57:...| Mohave|2024-07-22|2024-07-22 15:57:...| Mohave| AZ|Arizona|655| Hannah| Wall| 23| Librarian|

| 2019-10-01| AZ| HIOS| Mohave|http://uamarketpl...| 23| null| null| 88925| 88925| 8892588925|2024-07-22|2024-07-22 15:57:...| Mohave|2024-07-22|2024-07-22 15:57:...| Mohave| AZ|Arizona|639|Florence|McFarland| 23| Electrical engineer|

| 2019-10-01| AZ| HIOS| Mohave|http://uamarketpl...| 23| null| null| 88925| 88925| 8892588925|2024-07-22|2024-07-22 15:57:...| Mohave|2024-07-22|2024-07-22 15:57:...| Mohave| AZ|Arizona|248| Allison| Choi| 23|Computer hardware...|

+------------+----+-----+-----------+--------------------+-------+--------------+----------+--------+---------+-----------------+----------+--------------------+----------------+----------+--------------------+----------------+---+-------+---+--------+---------+---+--------------------+

only showing top 20 rows

>>> df\_renamed = second\_join.withColumnRenamed("\_c0", "state\_code") .withColumnRenamed("\_c1", "City")

>>> df.describe

Traceback (most recent call last):

File "<stdin>", line 1, in <module>

NameError: name 'df' is not defined

>>> df\_renamed.describe

<bound method DataFrame.describe of DataFrame[BusinessDate: date, stcd: string, srcnm: string, NetworkName: string, NetworkURL: string, custnum: string, MarketCoverage: string, RejectRows: string, IssuerId: string, IssuerId2: string, issueridcomposite: string, sysdt: date, systs: timestamp, cleannetworkname: string, curdt: date, curts: timestamp, cleannetworkname: string, state\_code: string, City: string, state\_code: string, City: string, \_c2: string, \_c3: int, \_c4: string]>

>>> first\_join=spark.sql("select \* from insureview as l join state\_table as r on l.stcd=r.\_c0 ")

>>>

>>> df\_renamed = df.select(

... col("BusinessDate"),

... col("stcd"),

... col("srcnm"),

... col("NetworkName"),

... col("NetworkURL"),

... col("custnum"),

... col("MarketCoverage"),

... col("RejectRows"),

... col("IssuerId"),

... col("IssuerId2"),

... col("issueridcomposite"),

... col("sysdt"),

... col("systs"),

... col("cleannetworkname").alias("cleannetworkname\_1"),

... col("curdt"),

... col("curts"),

... col("cleannetworkname").alias("cleannetworkname\_2"),

... col("state\_code").alias("state\_code\_1"),

... col("City").alias("City\_1"),

... col("state\_code").alias("state\_code\_2"),

... col("City").alias("City\_2"),

... col("\_c2"),

... col("\_c3"),

... col("\_c4")

... )

Traceback (most recent call last):

File "<stdin>", line 1, in <module>

NameError: name 'df' is not defined

>>> df\_renamed = df\_renamed.select(

... col("BusinessDate"),

... col("stcd"),

... col("srcnm"),

... col("NetworkName"),

... col("NetworkURL"),

... col("custnum"),

... col("MarketCoverage"),

... col("RejectRows"),

... col("IssuerId"),

... col("IssuerId2"),

... col("issueridcomposite"),

... col("sysdt"),

... col("systs"),

... col("cleannetworkname").alias("cleannetworkname\_1"),

... col("curdt"),

... col("curts"),

... col("cleannetworkname").alias("cleannetworkname\_2"),

... col("state\_code").alias("state\_code\_1"),

... col("City").alias("City\_1"),

... col("state\_code").alias("state\_code\_2"),

... col("City").alias("City\_2"),

... col("\_c2"),

... col("\_c3"),

... col("\_c4")

... )

Traceback (most recent call last):

File "<stdin>", line 25, in <module>

File "/usr/local/spark/python/pyspark/sql/dataframe.py", line 1685, in select

jdf = self.\_jdf.select(self.\_jcols(\*cols))

File "/usr/local/spark/python/lib/py4j-0.10.9.3-src.zip/py4j/java\_gateway.py", line 1322, in \_\_call\_\_

File "/usr/local/spark/python/pyspark/sql/utils.py", line 117, in deco

raise converted from None

pyspark.sql.utils.AnalysisException: Reference 'cleannetworkname' is ambiguous, could be: l.cleannetworkname, l.cleannetworkname.

>>> df\_temp = df\_named\

... .withColumnRenamed("cleannetworkname", "cleannetworkname\_temp") \

... .withColumnRenamed("state\_code", "state\_code\_temp") \

... .withColumnRenamed("City", "City\_temp")

Traceback (most recent call last):

File "<stdin>", line 1, in <module>

NameError: name 'df\_named' is not defined

>>>

>>> df\_temp = df\_renamed\

... .withColumnRenamed("cleannetworkname", "cleannetworkname\_temp") \

... .withColumnRenamed("state\_code", "state\_code\_temp") \

... .withColumnRenamed("City", "City\_temp")

>>>

>>> df\_renamed = df\_temp.select(

... col("BusinessDate"),

... col("stcd"),

... col("srcnm"),

... col("NetworkName"),

... col("NetworkURL"),

... col("custnum"),

... col("MarketCoverage"),

... col("RejectRows"),

... col("IssuerId"),

... col("IssuerId2"),

... col("issueridcomposite"),

... col("sysdt"),

... col("systs"),

... col("cleannetworkname\_temp").alias("cleannetworkname\_1"),

... col("curdt"),

... col("curts"),

... col("cleannetworkname\_temp").alias("cleannetworkname\_2"),

... col("state\_code\_temp").alias("state\_code\_1"),

... col("City\_temp").alias("City\_1"),

... col("state\_code\_temp").alias("state\_code\_2"),

... col("City\_temp").alias("City\_2"),

... col("\_c2"),

... col("\_c3"),

... col("\_c4")

... )

Traceback (most recent call last):

File "<stdin>", line 25, in <module>

File "/usr/local/spark/python/pyspark/sql/dataframe.py", line 1685, in select

jdf = self.\_jdf.select(self.\_jcols(\*cols))

File "/usr/local/spark/python/lib/py4j-0.10.9.3-src.zip/py4j/java\_gateway.py", line 1322, in \_\_call\_\_

File "/usr/local/spark/python/pyspark/sql/utils.py", line 117, in deco

raise converted from None

pyspark.sql.utils.AnalysisException: Reference 'cleannetworkname\_temp' is ambiguous, could be: cleannetworkname\_temp, cleannetworkname\_temp.

>>> df\_renamed = df\_temp.select(

... col("BusinessDate"),

... col("stcd"),

... col("srcnm"),

... col("NetworkName"),

... col("NetworkURL"),

... col("custnum"),

... col("MarketCoverage"),

... col("RejectRows"),

... col("IssuerId"),

... col("IssuerId2"),

... col("issueridcomposite"),

... col("sysdt"),

... col("systs"),

... col("cleannetworkname\_temp").alias("cleannetworkname\_1"),

... col("curdt"),

... col("curts"),

... col("cleannetworkname\_temp").alias("cleannetworkname\_2"),

... col("state\_code\_temp").alias("state\_code\_1"),

... col("City\_temp").alias("City\_1"),

... col("state\_code\_temp").alias("state\_code\_2"),

... col("City\_temp").alias("City\_2"),

... col("\_c2"),

... col("\_c3"),

... col("\_c4")

... )

Traceback (most recent call last):

File "<stdin>", line 25, in <module>

File "/usr/local/spark/python/pyspark/sql/dataframe.py", line 1685, in select

jdf = self.\_jdf.select(self.\_jcols(\*cols))

File "/usr/local/spark/python/lib/py4j-0.10.9.3-src.zip/py4j/java\_gateway.py", line 1322, in \_\_call\_\_

File "/usr/local/spark/python/pyspark/sql/utils.py", line 117, in deco

raise converted from None

pyspark.sql.utils.AnalysisException: Reference 'cleannetworkname\_temp' is ambiguous, could be: cleannetworkname\_temp, cleannetworkname\_temp.

>>> df\_temp.describe

<bound method DataFrame.describe of DataFrame[BusinessDate: date, stcd: string, srcnm: string, NetworkName: string, NetworkURL: string, custnum: string, MarketCoverage: string, RejectRows: string, IssuerId: string, IssuerId2: string, issueridcomposite: string, sysdt: date, systs: timestamp, cleannetworkname\_temp: string, curdt: date, curts: timestamp, cleannetworkname\_temp: string, state\_code\_temp: string, City\_temp: string, state\_code\_temp: string, City\_temp: string, \_c2: string, \_c3: int, \_c4: string]>

>>> df\_temp.drop("cleannetworkname\_temp")

DataFrame[BusinessDate: date, stcd: string, srcnm: string, NetworkName: string, NetworkURL: string, custnum: string, MarketCoverage: string, RejectRows: string, IssuerId: string, IssuerId2: string, issueridcomposite: string, sysdt: date, systs: timestamp, curdt: date, curts: timestamp, state\_code\_temp: string, City\_temp: string, state\_code\_temp: string, City\_temp: string, \_c2: string, \_c3: int, \_c4: string]

>>> df\_temp=df\_temp.drop("cleannetworkname\_temp")

>>> df\_temp.describe

<bound method DataFrame.describe of DataFrame[BusinessDate: date, stcd: string, srcnm: string, NetworkName: string, NetworkURL: string, custnum: string, MarketCoverage: string, RejectRows: string, IssuerId: string, IssuerId2: string, issueridcomposite: string, sysdt: date, systs: timestamp, curdt: date, curts: timestamp, state\_code\_temp: string, City\_temp: string, state\_code\_temp: string, City\_temp: string, \_c2: string, \_c3: int, \_c4: string]>

>>> df\_temp.write.parquet("hdfs:///user/hduser/Hackathon/PARQUET\_DATA")

Traceback (most recent call last):

File "<stdin>", line 1, in <module>

File "/usr/local/spark/python/pyspark/sql/readwriter.py", line 885, in parquet

self.\_jwrite.parquet(path)

File "/usr/local/spark/python/lib/py4j-0.10.9.3-src.zip/py4j/java\_gateway.py", line 1322, in \_\_call\_\_

File "/usr/local/spark/python/pyspark/sql/utils.py", line 117, in deco

raise converted from None

pyspark.sql.utils.AnalysisException: Found duplicate column(s) when inserting into hdfs://localhost:54310/user/hduser/Hackathon/PARQUET\_DATA: `city\_temp`, `state\_code\_temp`

>>> df\_temp=df\_temp.drop("city\_temp","state\_code\_temp")

>>> df\_temp.write.parquet("hdfs:///user/hduser/Hackathon/PARQUET\_DATA")

>>>

>>> WITH CountData AS (

File "<stdin>", line 1

WITH CountData AS (

^

SyntaxError: invalid syntax

>>> SELECT

File "<stdin>", line 1

SELECT

^

IndentationError: unexpected indent

>>> statedesc,

File "<stdin>", line 1

statedesc,

^

IndentationError: unexpected indent

>>> protocol,

File "<stdin>", line 1

protocol,

^

IndentationError: unexpected indent

>>> profession,

File "<stdin>", line 1

profession,

^

IndentationError: unexpected indent

>>> AVG(age) AS avg\_age, -- Calculating average age

File "<stdin>", line 1

AVG(age) AS avg\_age, -- Calculating average age

^

IndentationError: unexpected indent

>>> COUNT(\*) AS profession\_count, -- Counting number of professions

File "<stdin>", line 1

COUNT(\*) AS profession\_count, -- Counting number of professions

^

IndentationError: unexpected indent

>>> ROW\_NUMBER() OVER (PARTITION BY statedesc, protocol ORDER BY COUNT(\*) DESC) AS seqno -- Generating sequence number

File "<stdin>", line 1

ROW\_NUMBER() OVER (PARTITION BY statedesc, protocol ORDER BY COUNT(\*) DESC) AS seqno -- Generating sequence number

^

IndentationError: unexpected indent

>>> FROM your\_table

File "<stdin>", line 1

FROM your\_table

^

IndentationError: unexpected indent

>>> GROUP BY statedesc, protocol, profession

File "<stdin>", line 1

GROUP BY statedesc, protocol, profession

^

IndentationError: unexpected indent

>>> ),

File "<stdin>", line 1

),

^

SyntaxError: invalid syntax

>>> SecondHighestCount AS (

File "<stdin>", line 1

SecondHighestCount AS (

^

SyntaxError: invalid syntax

>>> SELECT

File "<stdin>", line 1

SELECT

^

IndentationError: unexpected indent

>>> statedesc,

File "<stdin>", line 1

statedesc,

^

IndentationError: unexpected indent

>>> protocol,

File "<stdin>", line 1

protocol,

^

IndentationError: unexpected indent

>>> profession,

File "<stdin>", line 1

profession,

^

IndentationError: unexpected indent

>>> avg\_age,

File "<stdin>", line 1

avg\_age,

^

IndentationError: unexpected indent

>>> profession\_count,

File "<stdin>", line 1

profession\_count,

^

IndentationError: unexpected indent

>>> seqno

File "<stdin>", line 1

seqno

^

IndentationError: unexpected indent

>>> FROM CountData

File "<stdin>", line 1

FROM CountData

^

IndentationError: unexpected indent

>>> WHERE seqno = 2 -- Filter to get the profession with the second highest count

File "<stdin>", line 1

WHERE seqno = 2 -- Filter to get the profession with the second highest count

^

IndentationError: unexpected indent

>>> )

File "<stdin>", line 1

)

^

SyntaxError: invalid syntax

>>> SELECT

Traceback (most recent call last):

File "<stdin>", line 1, in <module>

NameError: name 'SELECT' is not defined

>>> statedesc,

File "<stdin>", line 1

statedesc,

^

IndentationError: unexpected indent

>>> protocol,

File "<stdin>", line 1

protocol,

^

IndentationError: unexpected indent

>>> profession,

File "<stdin>", line 1

profession,

^

IndentationError: unexpected indent

>>> avg\_age,

File "<stdin>", line 1

avg\_age,

^

IndentationError: unexpected indent

>>> profession\_count

File "<stdin>", line 1

profession\_count

^

IndentationError: unexpected indent

>>> FROM SecondHighestCount

File "<stdin>", line 1

FROM SecondHighestCount

^

SyntaxError: invalid syntax

>>> ORDER BY statedesc, protocol, profession\_count DESC;

File "<stdin>", line 1

ORDER BY statedesc, protocol, profession\_count DESC;

^

SyntaxError: invalid syntax

>>>

>>> WITH CountData AS (

File "<stdin>", line 1

WITH CountData AS (

^

SyntaxError: invalid syntax

>>> SELECT

File "<stdin>", line 1

SELECT

^

IndentationError: unexpected indent

>>> statedesc,

File "<stdin>", line 1

statedesc,

^

IndentationError: unexpected indent

>>> protocol,

File "<stdin>", line 1

protocol,

^

IndentationError: unexpected indent

>>> profession,

File "<stdin>", line 1

profession,

^

IndentationError: unexpected indent

>>> AVG(age) AS avg\_age, -- Calculate average age

File "<stdin>", line 1

AVG(age) AS avg\_age, -- Calculate average age

^

IndentationError: unexpected indent

>>> COUNT(\*) AS profession\_count, -- Count occurrences of each profession

File "<stdin>", line 1

COUNT(\*) AS profession\_count, -- Count occurrences of each profession

^

IndentationError: unexpected indent

>>> ROW\_NUMBER() OVER (PARTITION BY statedesc, protocol ORDER BY COUNT(\*) DESC) AS seqno -- Generate sequence number

File "<stdin>", line 1

ROW\_NUMBER() OVER (PARTITION BY statedesc, protocol ORDER BY COUNT(\*) DESC) AS seqno -- Generate sequence number

^

IndentationError: unexpected indent

>>> FROM your\_table

File "<stdin>", line 1

FROM your\_table

^

IndentationError: unexpected indent

>>> GROUP BY statedesc, protocol, profession

File "<stdin>", line 1

GROUP BY statedesc, protocol, profession

^

IndentationError: unexpected indent

>>> ),

File "<stdin>", line 1

),

^

SyntaxError: invalid syntax

>>> SecondHighestCount AS (

File "<stdin>", line 1

SecondHighestCount AS (

^

SyntaxError: invalid syntax

>>> SELECT

File "<stdin>", line 1

SELECT

^

IndentationError: unexpected indent

>>> statedesc,

File "<stdin>", line 1

statedesc,

^

IndentationError: unexpected indent

>>> protocol,

File "<stdin>", line 1

protocol,

^

IndentationError: unexpected indent

>>> profession,

File "<stdin>", line 1

profession,

^

IndentationError: unexpected indent

>>> avg\_age,

File "<stdin>", line 1

avg\_age,

^

IndentationError: unexpected indent

>>> profession\_count

File "<stdin>", line 1

profession\_count

^

IndentationError: unexpected indent

>>> FROM CountData

File "<stdin>", line 1

FROM CountData

^

IndentationError: unexpected indent

>>> WHERE seqno = 2 -- Filter to get the profession with the second highest count

File "<stdin>", line 1

WHERE seqno = 2 -- Filter to get the profession with the second highest count

^

IndentationError: unexpected indent

>>> )query = """

File "<stdin>", line 1

)query = """

^

SyntaxError: invalid syntax

>>> WITH CountData AS (

File "<stdin>", line 1

WITH CountData AS (

^

SyntaxError: invalid syntax

>>> SELECT

File "<stdin>", line 1

SELECT

^

IndentationError: unexpected indent

>>> statedesc,

File "<stdin>", line 1

statedesc,

^

IndentationError: unexpected indent

>>> protocol,

File "<stdin>", line 1

protocol,

^

IndentationError: unexpected indent

>>> profession,

File "<stdin>", line 1

profession,

^

IndentationError: unexpected indent

>>> AVG(age) AS avg\_age,

File "<stdin>", line 1

AVG(age) AS avg\_age,

^

IndentationError: unexpected indent

>>> COUNT(\*) AS profession\_count,

File "<stdin>", line 1

COUNT(\*) AS profession\_count,

^

IndentationError: unexpected indent

>>> ROW\_NUMBER() OVER (PARTITION BY statedesc, protocol ORDER BY COUNT(\*) DESC) AS seqno

File "<stdin>", line 1

ROW\_NUMBER() OVER (PARTITION BY statedesc, protocol ORDER BY COUNT(\*) DESC) AS seqno

^

IndentationError: unexpected indent

>>> FROM your\_table

File "<stdin>", line 1

FROM your\_table

^

IndentationError: unexpected indent

>>> GROUP BY statedesc, protocol, profession

File "<stdin>", line 1

GROUP BY statedesc, protocol, profession

^

IndentationError: unexpected indent

>>> ),

File "<stdin>", line 1

),

^

SyntaxError: invalid syntax

>>> SecondHighestCount AS (

File "<stdin>", line 1

SecondHighestCount AS (

^

SyntaxError: invalid syntax

>>> SELECT

File "<stdin>", line 1

SELECT

^

IndentationError: unexpected indent

>>> statedesc,

File "<stdin>", line 1

statedesc,

^

IndentationError: unexpected indent

>>> protocol,

File "<stdin>", line 1

protocol,

^

IndentationError: unexpected indent

>>> profession,

File "<stdin>", line 1

profession,

^

IndentationError: unexpected indent

>>> avg\_age,

File "<stdin>", line 1

avg\_age,

^

IndentationError: unexpected indent

>>> profession\_count

File "<stdin>", line 1

profession\_count

^

IndentationError: unexpected indent

>>> FROM CountData

File "<stdin>", line 1

FROM CountData

^

IndentationError: unexpected indent

>>> WHERE seqno = 2

File "<stdin>", line 1

WHERE seqno = 2

^

IndentationError: unexpected indent

>>> )

File "<stdin>", line 1

)

^

SyntaxError: invalid syntax

>>> SELECT

Traceback (most recent call last):

File "<stdin>", line 1, in <module>

NameError: name 'SELECT' is not defined

>>> statedesc,

File "<stdin>", line 1

statedesc,

^

IndentationError: unexpected indent

>>> protocol,

File "<stdin>", line 1

protocol,

^

IndentationError: unexpected indent

>>> profession,

File "<stdin>", line 1

profession,

^

IndentationError: unexpected indent

>>> avg\_age,

File "<stdin>", line 1

avg\_age,

^

IndentationError: unexpected indent

>>> profession\_count

File "<stdin>", line 1

profession\_count

^

IndentationError: unexpected indent

>>> FROM SecondHighestCount

File "<stdin>", line 1

FROM SecondHighestCount

^

SyntaxError: invalid syntax

>>> ORDER BY statedesc, protocol;

File "<stdin>", line 1

ORDER BY statedesc, protocol;

^

SyntaxError: invalid syntax

>>> """\

...

...

...

...

... ;

...

File "<stdin>", line 5

;

^

SyntaxError: EOF while scanning triple-quoted string literal

>>>

>>> query = """

... WITH CountData AS (

... SELECT

... statedesc,

... protocol,

... profession,

... AVG(age) AS avg\_age,

... COUNT(\*) AS profession\_count,

... ROW\_NUMBER() OVER (PARTITION BY statedesc, protocol ORDER BY COUNT(\*) DESC) AS seqno

... FROM your\_table

... GROUP BY statedesc, protocol, profession

... ),

... SecondHighestCount AS (

... SELECT

... statedesc,

... protocol,

... profession,

... avg\_age,

... profession\_count

... FROM CountData

... WHERE seqno = 2

... )

... SELECT

... statedesc,

... protocol,

... profession,

... avg\_age,

... profession\_count

... FROM SecondHighestCount

... ORDER BY statedesc, protocol;

... """

>>>

>>> second\_join.describe

<bound method DataFrame.describe of DataFrame[BusinessDate: date, stcd: string, srcnm: string, NetworkName: string, NetworkURL: string, custnum: string, MarketCoverage: string, RejectRows: string, IssuerId: string, IssuerId2: string, issueridcomposite: string, sysdt: date, systs: timestamp, cleannetworkname: string, curdt: date, curts: timestamp, cleannetworkname: string, \_c0: string, \_c1: string, \_c0: string, \_c1: string, \_c2: string, \_c3: int, \_c4: string]>

>>> second\_join.show(2)

+------------+----+-----+-----------+--------------------+-------+--------------+----------+--------+---------+-----------------+----------+--------------------+----------------+----------+--------------------+----------------+---+-------+---+------+-------+---+----------+

|BusinessDate|stcd|srcnm|NetworkName| NetworkURL|custnum|MarketCoverage|RejectRows|IssuerId|IssuerId2|issueridcomposite| sysdt| systs|cleannetworkname| curdt| curts|cleannetworkname|\_c0| \_c1|\_c0| \_c1| \_c2|\_c3| \_c4|

+------------+----+-----+-----------+--------------------+-------+--------------+----------+--------+---------+-----------------+----------+--------------------+----------------+----------+--------------------+----------------+---+-------+---+------+-------+---+----------+

| 2019-10-01| AZ| HIOS| La Paz|http://uamarketpl...| 21| null| null| 88925| 88925| 8892588925|2024-07-22|2024-07-22 16:28:...| La Paz|2024-07-22|2024-07-22 16:28:...| La Paz| AZ|Arizona|779| Jason|Bowling| 21| Coach|

| 2019-10-01| AZ| HIOS| La Paz|http://uamarketpl...| 21| null| null| 88925| 88925| 8892588925|2024-07-22|2024-07-22 16:28:...| La Paz|2024-07-22|2024-07-22 16:28:...| La Paz| AZ|Arizona|765|Janice| Bowden| 21|Politician|

+------------+----+-----+-----------+--------------------+-------+--------------+----------+--------+---------+-----------------+----------+--------------------+----------------+----------+--------------------+----------------+---+-------+---+------+-------+---+----------+

only showing top 2 rows

>>> second\_join.createOrReplaceView("view1")

Traceback (most recent call last):

File "<stdin>", line 1, in <module>

File "/usr/local/spark/python/pyspark/sql/dataframe.py", line 1660, in \_\_getattr\_\_

"'%s' object has no attribute '%s'" % (self.\_\_class\_\_.\_\_name\_\_, name))

AttributeError: 'DataFrame' object has no attribute 'createOrReplaceView'

>>> second\_join.createOrReplaceTempView("view1")

>>> result\_df.show(2)

+------------+----+-----+-----------+--------------------+-------+--------------+--------------------+--------+---------+-----------------+----------+--------------------+----------------+----------+--------------------+----------------+-------------+

|BusinessDate|stcd|srcnm|NetworkName| NetworkURL|custnum|MarketCoverage| RejectRows|IssuerId|IssuerId2|issueridcomposite| sysdt| systs|cleannetworkname| curdt| curts|cleannetworkname|Protocol\_Used|

+------------+----+-----+-----------+--------------------+-------+--------------+--------------------+--------+---------+-----------------+----------+--------------------+----------------+----------+--------------------+----------------+-------------+

| 2019-10-01| AK| HIOS|ODS Premier|https://www.modah...| 13| null| null| 21989| 21989| 2198921989|2024-07-22|2024-07-22 16:33:...| ODS Premier|2024-07-22|2024-07-22 16:33:...| ODS Premier| http secured|

| 2019-10-01| AK| HIOS|ODS Premier| null| null| null|219,219,2019-10-0...| 219| 219| 219219|2024-07-22|2024-07-22 16:33:...| ODS Premier|2024-07-22|2024-07-22 16:33:...| ODS Premier| No Protocol|

+------------+----+-----+-----------+--------------------+-------+--------------+--------------------+--------+---------+-----------------+----------+--------------------+----------------+----------+--------------------+----------------+-------------+

only showing top 2 rows

>>> result\_df.createOrReplaceTempView("view2")

>>> spark.sql("select l.\*,r.Protocol\_Used from view1 as l and view2 as r ").show()

Traceback (most recent call last):

File "<stdin>", line 1, in <module>

File "/usr/local/spark/python/pyspark/sql/session.py", line 723, in sql

return DataFrame(self.\_jsparkSession.sql(sqlQuery), self.\_wrapped)

File "/usr/local/spark/python/lib/py4j-0.10.9.3-src.zip/py4j/java\_gateway.py", line 1322, in \_\_call\_\_

File "/usr/local/spark/python/pyspark/sql/utils.py", line 117, in deco

raise converted from None

pyspark.sql.utils.ParseException:

mismatched input 'and' expecting {<EOF>, ';'}(line 1, pos 43)

== SQL ==

select l.\*,r.Protocol\_Used from view1 as l and view2 as r

-------------------------------------------^^^

>>> spark.sql("select l.\*,r.Protocol\_Used from view1 as l,view2 as r ").show()

+------------+----+-----+-----------+--------------------+-------+--------------+----------+--------+---------+-----------------+----------+--------------------+----------------+----------+--------------------+----------------+---+-------+---+-----+-------+---+-----+----------------+

|BusinessDate|stcd|srcnm|NetworkName| NetworkURL|custnum|MarketCoverage|RejectRows|IssuerId|IssuerId2|issueridcomposite| sysdt| systs|cleannetworkname| curdt| curts|cleannetworkname|\_c0| \_c1|\_c0| \_c1| \_c2|\_c3| \_c4| Protocol\_Used|

+------------+----+-----+-----------+--------------------+-------+--------------+----------+--------+---------+-----------------+----------+--------------------+----------------+----------+--------------------+----------------+---+-------+---+-----+-------+---+-----+----------------+

| 2019-10-01| AZ| HIOS| La Paz|http://uamarketpl...| 21| null| null| 88925| 88925| 8892588925|2024-07-22|2024-07-22 16:37:...| La Paz|2024-07-22|2024-07-22 16:37:...| La Paz| AZ|Arizona|779|Jason|Bowling| 21|Coach| http secured|

| 2019-10-01| AZ| HIOS| La Paz|http://uamarketpl...| 21| null| null| 88925| 88925| 8892588925|2024-07-22|2024-07-22 16:37:...| La Paz|2024-07-22|2024-07-22 16:37:...| La Paz| AZ|Arizona|779|Jason|Bowling| 21|Coach| No Protocol|

| 2019-10-01| AZ| HIOS| La Paz|http://uamarketpl...| 21| null| null| 88925| 88925| 8892588925|2024-07-22|2024-07-22 16:37:...| La Paz|2024-07-22|2024-07-22 16:37:...| La Paz| AZ|Arizona|779|Jason|Bowling| 21|Coach| http secured|

| 2019-10-01| AZ| HIOS| La Paz|http://uamarketpl...| 21| null| null| 88925| 88925| 8892588925|2024-07-22|2024-07-22 16:37:...| La Paz|2024-07-22|2024-07-22 16:37:...| La Paz| AZ|Arizona|779|Jason|Bowling| 21|Coach|http non secured|

| 2019-10-01| AZ| HIOS| La Paz|http://uamarketpl...| 21| null| null| 88925| 88925| 8892588925|2024-07-22|2024-07-22 16:37:...| La Paz|2024-07-22|2024-07-22 16:37:...| La Paz| AZ|Arizona|779|Jason|Bowling| 21|Coach| http secured|

| 2019-10-01| AZ| HIOS| La Paz|http://uamarketpl...| 21| null| null| 88925| 88925| 8892588925|2024-07-22|2024-07-22 16:37:...| La Paz|2024-07-22|2024-07-22 16:37:...| La Paz| AZ|Arizona|779|Jason|Bowling| 21|Coach| http secured|

| 2019-10-01| AZ| HIOS| La Paz|http://uamarketpl...| 21| null| null| 88925| 88925| 8892588925|2024-07-22|2024-07-22 16:37:...| La Paz|2024-07-22|2024-07-22 16:37:...| La Paz| AZ|Arizona|779|Jason|Bowling| 21|Coach| http secured|

| 2019-10-01| AZ| HIOS| La Paz|http://uamarketpl...| 21| null| null| 88925| 88925| 8892588925|2024-07-22|2024-07-22 16:37:...| La Paz|2024-07-22|2024-07-22 16:37:...| La Paz| AZ|Arizona|779|Jason|Bowling| 21|Coach|http non secured|

| 2019-10-01| AZ| HIOS| La Paz|http://uamarketpl...| 21| null| null| 88925| 88925| 8892588925|2024-07-22|2024-07-22 16:37:...| La Paz|2024-07-22|2024-07-22 16:37:...| La Paz| AZ|Arizona|779|Jason|Bowling| 21|Coach| No Protocol|

| 2019-10-01| AZ| HIOS| La Paz|http://uamarketpl...| 21| null| null| 88925| 88925| 8892588925|2024-07-22|2024-07-22 16:37:...| La Paz|2024-07-22|2024-07-22 16:37:...| La Paz| AZ|Arizona|779|Jason|Bowling| 21|Coach|http non secured|

| 2019-10-01| AZ| HIOS| La Paz|http://uamarketpl...| 21| null| null| 88925| 88925| 8892588925|2024-07-22|2024-07-22 16:37:...| La Paz|2024-07-22|2024-07-22 16:37:...| La Paz| AZ|Arizona|779|Jason|Bowling| 21|Coach|http non secured|

| 2019-10-01| AZ| HIOS| La Paz|http://uamarketpl...| 21| null| null| 88925| 88925| 8892588925|2024-07-22|2024-07-22 16:37:...| La Paz|2024-07-22|2024-07-22 16:37:...| La Paz| AZ|Arizona|779|Jason|Bowling| 21|Coach| http secured|

| 2019-10-01| AZ| HIOS| La Paz|http://uamarketpl...| 21| null| null| 88925| 88925| 8892588925|2024-07-22|2024-07-22 16:37:...| La Paz|2024-07-22|2024-07-22 16:37:...| La Paz| AZ|Arizona|779|Jason|Bowling| 21|Coach|http non secured|

| 2019-10-01| AZ| HIOS| La Paz|http://uamarketpl...| 21| null| null| 88925| 88925| 8892588925|2024-07-22|2024-07-22 16:37:...| La Paz|2024-07-22|2024-07-22 16:37:...| La Paz| AZ|Arizona|779|Jason|Bowling| 21|Coach|http non secured|

| 2019-10-01| AZ| HIOS| La Paz|http://uamarketpl...| 21| null| null| 88925| 88925| 8892588925|2024-07-22|2024-07-22 16:37:...| La Paz|2024-07-22|2024-07-22 16:37:...| La Paz| AZ|Arizona|779|Jason|Bowling| 21|Coach| http secured|

| 2019-10-01| AZ| HIOS| La Paz|http://uamarketpl...| 21| null| null| 88925| 88925| 8892588925|2024-07-22|2024-07-22 16:37:...| La Paz|2024-07-22|2024-07-22 16:37:...| La Paz| AZ|Arizona|779|Jason|Bowling| 21|Coach| http secured|

| 2019-10-01| AZ| HIOS| La Paz|http://uamarketpl...| 21| null| null| 88925| 88925| 8892588925|2024-07-22|2024-07-22 16:37:...| La Paz|2024-07-22|2024-07-22 16:37:...| La Paz| AZ|Arizona|779|Jason|Bowling| 21|Coach|http non secured|

| 2019-10-01| AZ| HIOS| La Paz|http://uamarketpl...| 21| null| null| 88925| 88925| 8892588925|2024-07-22|2024-07-22 16:37:...| La Paz|2024-07-22|2024-07-22 16:37:...| La Paz| AZ|Arizona|779|Jason|Bowling| 21|Coach| http secured|

| 2019-10-01| AZ| HIOS| La Paz|http://uamarketpl...| 21| null| null| 88925| 88925| 8892588925|2024-07-22|2024-07-22 16:37:...| La Paz|2024-07-22|2024-07-22 16:37:...| La Paz| AZ|Arizona|779|Jason|Bowling| 21|Coach|http non secured|

| 2019-10-01| AZ| HIOS| La Paz|http://uamarketpl...| 21| null| null| 88925| 88925| 8892588925|2024-07-22|2024-07-22 16:37:...| La Paz|2024-07-22|2024-07-22 16:37:...| La Paz| AZ|Arizona|779|Jason|Bowling| 21|Coach|http non secured|

+------------+----+-----+-----------+--------------------+-------+--------------+----------+--------+---------+-----------------+----------+--------------------+----------------+----------+--------------------+----------------+---+-------+---+-----+-------+---+-----+----------------+

only showing top 20 rows

>>> second\_join=spark.sql("select l.\*,r.Protocol\_Used from view1 as l,view2 as r ")

>>> second\_join.createOrReplaceView("view1")

Traceback (most recent call last):

File "<stdin>", line 1, in <module>

File "/usr/local/spark/python/pyspark/sql/dataframe.py", line 1660, in \_\_getattr\_\_

"'%s' object has no attribute '%s'" % (self.\_\_class\_\_.\_\_name\_\_, name))

AttributeError: 'DataFrame' object has no attribute 'createOrReplaceView'

>>> spark.catalog.dropTempView("view`")

>>> spark.catalog.dropTempView("view1")

>>> second\_join.createOrReplaceView("view1")

Traceback (most recent call last):

File "<stdin>", line 1, in <module>

File "/usr/local/spark/python/pyspark/sql/dataframe.py", line 1660, in \_\_getattr\_\_

"'%s' object has no attribute '%s'" % (self.\_\_class\_\_.\_\_name\_\_, name))

AttributeError: 'DataFrame' object has no attribute 'createOrReplaceView'

>>> spark.catalog.dropTempView("view1")

>>> second\_join.createOrReplaceTempView("view1")

>>> spark.sql("select \* from view1 group by Protocol\_Used,srcnm,\_c3 ").show(3)

Traceback (most recent call last):

File "<stdin>", line 1, in <module>

File "/usr/local/spark/python/pyspark/sql/session.py", line 723, in sql

return DataFrame(self.\_jsparkSession.sql(sqlQuery), self.\_wrapped)

File "/usr/local/spark/python/lib/py4j-0.10.9.3-src.zip/py4j/java\_gateway.py", line 1322, in \_\_call\_\_

File "/usr/local/spark/python/pyspark/sql/utils.py", line 117, in deco

raise converted from None

pyspark.sql.utils.AnalysisException: expression 'view1.BusinessDate' is neither present in the group by, nor is it an aggregate function. Add to group by or wrap in first() (or first\_value) if you don't care which value you get.;

Aggregate [Protocol\_Used#48671, srcnm#7747, \_c3#52127], [BusinessDate#1378, stcd#7759, srcnm#7747, NetworkName#1381, NetworkURL#1382, custnum#1383, MarketCoverage#1384, RejectRows#1386, IssuerId#8312, IssuerId2#8325, issueridcomposite#8901, sysdt#9003, systs#9016, cleannetworkname#47665, curdt#48091, curts#48092, cleannetworkname#48111, \_c0#26007, \_c1#26008, \_c0#52124, \_c1#52125, \_c2#52126, \_c3#52127, \_c4#52128, Protocol\_Used#48671]

+- SubqueryAlias view1

+- View (`view1`, [BusinessDate#1378,stcd#7759,srcnm#7747,NetworkName#1381,NetworkURL#1382,custnum#1383,MarketCoverage#1384,RejectRows#1386,IssuerId#8312,IssuerId2#8325,issueridcomposite#8901,sysdt#9003,systs#9016,cleannetworkname#47665,curdt#48091,curts#48092,cleannetworkname#48111,\_c0#26007,\_c1#26008,\_c0#52124,\_c1#52125,\_c2#52126,\_c3#52127,\_c4#52128,Protocol\_Used#48671])

+- Project [BusinessDate#1378, stcd#7759, srcnm#7747, NetworkName#1381, NetworkURL#1382, custnum#1383, MarketCoverage#1384, RejectRows#1386, IssuerId#8312, IssuerId2#8325, issueridcomposite#8901, sysdt#9003, systs#9016, cleannetworkname#47665, curdt#48091, curts#48092, cleannetworkname#48111, \_c0#26007, \_c1#26008, \_c0#52124, \_c1#52125, \_c2#52126, \_c3#52127, \_c4#52128, Protocol\_Used#48671]

+- Join Inner

:- SubqueryAlias l

: +- SubqueryAlias view1

: +- View (`view1`, [BusinessDate#1378,stcd#7759,srcnm#7747,NetworkName#1381,NetworkURL#1382,custnum#1383,MarketCoverage#1384,RejectRows#1386,IssuerId#8312,IssuerId2#8325,issueridcomposite#8901,sysdt#9003,systs#9016,cleannetworkname#47665,curdt#48091,curts#48092,cleannetworkname#48111,\_c0#26007,\_c1#26008,\_c0#52124,\_c1#52125,\_c2#52126,\_c3#52127,\_c4#52128])

: +- Project [BusinessDate#1378, stcd#7759, srcnm#7747, NetworkName#1381, NetworkURL#1382, custnum#1383, MarketCoverage#1384, RejectRows#1386, IssuerId#8312, IssuerId2#8325, issueridcomposite#8901, sysdt#9003, systs#9016, cleannetworkname#47665, curdt#48091, curts#48092, cleannetworkname#48111, \_c0#26007, \_c1#26008, \_c0#52124, \_c1#52125, \_c2#52126, \_c3#52127, \_c4#52128]

: +- Join Inner, (cast(custnum#1383 as int) = \_c3#52127)

: :- SubqueryAlias l

: : +- SubqueryAlias insureview

: : +- View (`insureview`, [BusinessDate#1378,stcd#7759,srcnm#7747,NetworkName#1381,NetworkURL#1382,custnum#1383,MarketCoverage#1384,RejectRows#1386,IssuerId#8312,IssuerId2#8325,issueridcomposite#8901,sysdt#9003,systs#9016,cleannetworkname#47665,curdt#48091,curts#48092,cleannetworkname#48111,\_c0#26007,\_c1#26008])

: : +- Project [BusinessDate#1378, stcd#7759, srcnm#7747, NetworkName#1381, NetworkURL#1382, custnum#1383, MarketCoverage#1384, RejectRows#1386, IssuerId#8312, IssuerId2#8325, issueridcomposite#8901, sysdt#9003, systs#9016, cleannetworkname#47665, curdt#48091, curts#48092, cleannetworkname#48111, \_c0#26007, \_c1#26008]

: : +- Join Inner, (stcd#7759 = \_c0#26007)

: : :- SubqueryAlias l

: : : +- SubqueryAlias insureview

: : : +- View (`insureview`, [BusinessDate#1378,stcd#7759,srcnm#7747,NetworkName#1381,NetworkURL#1382,custnum#1383,MarketCoverage#1384,RejectRows#1386,IssuerId#8312,IssuerId2#8325,issueridcomposite#8901,sysdt#9003,systs#9016,cleannetworkname#47665,curdt#48091,curts#48092,cleannetworkname#48111])

: : : +- Project [BusinessDate#1378, stcd#7759, srcnm#7747, NetworkName#1381, NetworkURL#1382, custnum#1383, MarketCoverage#1384, RejectRows#1386, IssuerId#8312, IssuerId2#8325, issueridcomposite#8901, sysdt#9003, systs#9016, cleannetworkname#47665, curdt#48091, curts#48092, sql\_function(NetworkName#1381) AS cleannetworkname#48111]

: : : +- SubqueryAlias insureview

: : : +- View (`insureview`, [BusinessDate#1378,stcd#7759,srcnm#7747,NetworkName#1381,NetworkURL#1382,custnum#1383,MarketCoverage#1384,RejectRows#1386,IssuerId#8312,IssuerId2#8325,issueridcomposite#8901,sysdt#9003,systs#9016,cleannetworkname#47665,curdt#48091,curts#48092])

: : : +- Project [BusinessDate#1378, stcd#7759, srcnm#7747, NetworkName#1381, NetworkURL#1382, custnum#1383, MarketCoverage#1384, RejectRows#1386, IssuerId#8312, IssuerId2#8325, issueridcomposite#8901, sysdt#9003, systs#9016, cleannetworkname#47665, current\_date(Some(Asia/Kolkata)) AS curdt#48091, current\_timestamp() AS curts#48092]

: : : +- SubqueryAlias insureview

: : : +- View (`insureview`, [BusinessDate#1378,stcd#7759,srcnm#7747,NetworkName#1381,NetworkURL#1382,custnum#1383,MarketCoverage#1384,RejectRows#1386,IssuerId#8312,IssuerId2#8325,issueridcomposite#8901,sysdt#9003,systs#9016,cleannetworkname#47665])

: : : +- Project [BusinessDate#1378, stcd#7759, srcnm#7747, NetworkName#1381, NetworkURL#1382, custnum#1383, MarketCoverage#1384, RejectRows#1386, IssuerId#8312, IssuerId2#8325, issueridcomposite#8901, sysdt#9003, systs#9016, sql\_function(NetworkName#1381) AS cleannetworkname#47665]

: : : +- SubqueryAlias insureview

: : : +- View (`insureview`, [BusinessDate#1378,stcd#7759,srcnm#7747,NetworkName#1381,NetworkURL#1382,custnum#1383,MarketCoverage#1384,RejectRows#1386,IssuerId#8312,IssuerId2#8325,issueridcomposite#8901,sysdt#9003,systs#9016])

: : : +- Project [BusinessDate#1378, stcd#7759, srcnm#7747, NetworkName#1381, NetworkURL#1382, custnum#1383, MarketCoverage#1384, RejectRows#1386, IssuerId#8312, IssuerId2#8325, issueridcomposite#8901, sysdt#9003, current\_timestamp() AS systs#9016]

: : : +- Project [BusinessDate#1378, stcd#7759, srcnm#7747, NetworkName#1381, NetworkURL#1382, custnum#1383, MarketCoverage#1384, RejectRows#1386, IssuerId#8312, IssuerId2#8325, issueridcomposite#8901, current\_date(Some(Asia/Kolkata)) AS sysdt#9003]

: : : +- Project [BusinessDate#1378, stcd#7759, srcnm#7747, NetworkName#1381, NetworkURL#1382, custnum#1383, MarketCoverage#1384, RejectRows#1386, IssuerId#8312, IssuerId2#8325, issueridcomposite#8901]

: : : +- Project [BusinessDate#1378, stcd#7759, srcnm#7747, NetworkName#1381, NetworkURL#1382, custnum#1383, MarketCoverage#1384, DentalOnlyPlan#1385, RejectRows#1386, IssuerId#8312, IssuerId2#8325, concat(IssuerId#8312, IssuerId2#8325) AS issueridcomposite#8901]

: : : +- Project [BusinessDate#1378, stcd#7759, srcnm#7747, NetworkName#1381, NetworkURL#1382, custnum#1383, MarketCoverage#1384, DentalOnlyPlan#1385, RejectRows#1386, IssuerId#8312, IssuerId2#8325]

: : : +- Project [IssuerIDDD#8273, BusinessDate#1378, stcd#7759, srcnm#7747, NetworkName#1381, NetworkURL#1382, custnum#1383, MarketCoverage#1384, DentalOnlyPlan#1385, RejectRows#1386, IssuerId#8312, IssuerId2#8325]

: : : +- Project [IssuerIDD#8261, IssuerIDDD#8273, BusinessDate#1378, stcd#7759, srcnm#7747, NetworkName#1381, NetworkURL#1382, custnum#1383, MarketCoverage#1384, DentalOnlyPlan#1385, RejectRows#1386, IssuerId#8312, cast(IssuerIDDD#8273 as string) AS IssuerId2#8325]

: : : +- Project [IssuerIDD#8261, IssuerIDDD#8273, BusinessDate#1378, stcd#7759, srcnm#7747, NetworkName#1381, NetworkURL#1382, custnum#1383, MarketCoverage#1384, DentalOnlyPlan#1385, RejectRows#1386, cast(IssuerIDD#8261 as string) AS IssuerId#8312]

: : : +- Project [IssuerIDD#8261, IssuerId2#1377 AS IssuerIDDD#8273, BusinessDate#1378, stcd#7759, srcnm#7747, NetworkName#1381, NetworkURL#1382, custnum#1383, MarketCoverage#1384, DentalOnlyPlan#1385, RejectRows#1386]

: : : +- Project [IssuerId#1376 AS IssuerIDD#8261, IssuerId2#1377, BusinessDate#1378, stcd#7759, srcnm#7747, NetworkName#1381, NetworkURL#1382, custnum#1383, MarketCoverage#1384, DentalOnlyPlan#1385, RejectRows#1386]

: : : +- Project [IssuerId#1376, IssuerId2#1377, BusinessDate#1378, StateCode#1379 AS stcd#7759, srcnm#7747, NetworkName#1381, NetworkURL#1382, custnum#1383, MarketCoverage#1384, DentalOnlyPlan#1385, RejectRows#1386]

: : : +- Project [IssuerId#1376, IssuerId2#1377, BusinessDate#1378, StateCode#1379, SourceName#1380 AS srcnm#7747, NetworkName#1381, NetworkURL#1382, custnum#1383, MarketCoverage#1384, DentalOnlyPlan#1385, RejectRows#1386]

: : : +- Union false, false

: : : :- Relation [IssuerId#1376,IssuerId2#1377,BusinessDate#1378,StateCode#1379,SourceName#1380,NetworkName#1381,NetworkURL#1382,custnum#1383,MarketCoverage#1384,DentalOnlyPlan#1385,RejectRows#1386] csv

: : : +- Relation [IssuerId#491,IssuerId2#492,BusinessDate#493,StateCode#494,SourceName#495,NetworkName#496,NetworkURL#497,custnum#498,MarketCoverage#499,DentalOnlyPlan#500,RejectRows#501] csv

: : +- SubqueryAlias r

: : +- SubqueryAlias state\_table

: : +- View (`state\_table`, [\_c0#26007,\_c1#26008])

: : +- Project [\_c0#26007, \_c1#26008]

: : +- Filter ((isnull(\_c2#26009) AND isnull(\_c3#26010)) AND isnull(\_c4#26011))

: : +- Project [\_c0#26007, \_c1#26008, \_c2#26009, \_c3#26010, \_c4#26011]

: : +- Relation [\_c0#26007,\_c1#26008,\_c2#26009,\_c3#26010,\_c4#26011] csv

: +- SubqueryAlias r

: +- SubqueryAlias cust\_table

: +- View (`cust\_table`, [\_c0#52124,\_c1#52125,\_c2#52126,\_c3#52127,\_c4#52128])

: +- Filter (((isnotnull(\_c0#52124) AND isnotnull(\_c1#52125)) AND isnotnull(\_c2#52126)) AND (isnotnull(\_c3#52127) AND isnotnull(\_c4#52128)))

: +- Project [\_c0#52124, \_c1#52125, \_c2#52126, \_c3#52127, \_c4#52128]

: +- Relation [\_c0#52124,\_c1#52125,\_c2#52126,\_c3#52127,\_c4#52128] csv

+- SubqueryAlias r

+- SubqueryAlias view2

+- View (`view2`, [BusinessDate#57031,stcd#57060,srcnm#57061,NetworkName#57034,NetworkURL#57035,custnum#57036,MarketCoverage#57037,RejectRows#57039,IssuerId#57059,IssuerId2#57058,issueridcomposite#57057,sysdt#57056,systs#57055,cleannetworkname#57054,curdt#57052,curts#57053,cleannetworkname#57051,Protocol\_Used#48671])

+- Project [BusinessDate#57031, stcd#57060, srcnm#57061, NetworkName#57034, NetworkURL#57035, custnum#57036, MarketCoverage#57037, RejectRows#57039, IssuerId#57059, IssuerId2#57058, issueridcomposite#57057, sysdt#57056, systs#57055, cleannetworkname#57054, curdt#57052, curts#57053, cleannetworkname#57051, CASE WHEN NetworkURL#57035 LIKE https://% THEN http secured WHEN NetworkURL#57035 LIKE http://% THEN http non secured ELSE No Protocol END AS Protocol\_Used#48671]

+- SubqueryAlias insureview

+- View (`insureview`, [BusinessDate#57031,stcd#57060,srcnm#57061,NetworkName#57034,NetworkURL#57035,custnum#57036,MarketCoverage#57037,RejectRows#57039,IssuerId#57059,IssuerId2#57058,issueridcomposite#57057,sysdt#57056,systs#57055,cleannetworkname#57054,curdt#57052,curts#57053,cleannetworkname#57051])

+- Project [BusinessDate#57031, stcd#57060, srcnm#57061, NetworkName#57034, NetworkURL#57035, custnum#57036, MarketCoverage#57037, RejectRows#57039, IssuerId#57059, IssuerId2#57058, issueridcomposite#57057, sysdt#57056, systs#57055, cleannetworkname#57054, curdt#57052, curts#57053, sql\_function(NetworkName#57034) AS cleannetworkname#57051]

+- SubqueryAlias insureview

+- View (`insureview`, [BusinessDate#57031,stcd#57060,srcnm#57061,NetworkName#57034,NetworkURL#57035,custnum#57036,MarketCoverage#57037,RejectRows#57039,IssuerId#57059,IssuerId2#57058,issueridcomposite#57057,sysdt#57056,systs#57055,cleannetworkname#57054,curdt#57052,curts#57053])

+- Project [BusinessDate#57031, stcd#57060, srcnm#57061, NetworkName#57034, NetworkURL#57035, custnum#57036, MarketCoverage#57037, RejectRows#57039, IssuerId#57059, IssuerId2#57058, issueridcomposite#57057, sysdt#57056, systs#57055, cleannetworkname#57054, current\_date(Some(Asia/Kolkata)) AS curdt#57052, current\_timestamp() AS curts#57053]

+- SubqueryAlias insureview

+- View (`insureview`, [BusinessDate#57031,stcd#57060,srcnm#57061,NetworkName#57034,NetworkURL#57035,custnum#57036,MarketCoverage#57037,RejectRows#57039,IssuerId#57059,IssuerId2#57058,issueridcomposite#57057,sysdt#57056,systs#57055,cleannetworkname#57054])

+- Project [BusinessDate#57031, stcd#57060, srcnm#57061, NetworkName#57034, NetworkURL#57035, custnum#57036, MarketCoverage#57037, RejectRows#57039, IssuerId#57059, IssuerId2#57058, issueridcomposite#57057, sysdt#57056, systs#57055, sql\_function(NetworkName#57034) AS cleannetworkname#57054]

+- SubqueryAlias insureview

+- View (`insureview`, [BusinessDate#57031,stcd#57060,srcnm#57061,NetworkName#57034,NetworkURL#57035,custnum#57036,MarketCoverage#57037,RejectRows#57039,IssuerId#57059,IssuerId2#57058,issueridcomposite#57057,sysdt#57056,systs#57055])

+- Project [BusinessDate#57031, stcd#57060, srcnm#57061, NetworkName#57034, NetworkURL#57035, custnum#57036, MarketCoverage#57037, RejectRows#57039, IssuerId#57059, IssuerId2#57058, issueridcomposite#57057, sysdt#57056, current\_timestamp() AS systs#57055]

+- Project [BusinessDate#57031, stcd#57060, srcnm#57061, NetworkName#57034, NetworkURL#57035, custnum#57036, MarketCoverage#57037, RejectRows#57039, IssuerId#57059, IssuerId2#57058, issueridcomposite#57057, current\_date(Some(Asia/Kolkata)) AS sysdt#57056]

+- Project [BusinessDate#57031, stcd#57060, srcnm#57061, NetworkName#57034, NetworkURL#57035, custnum#57036, MarketCoverage#57037, RejectRows#57039, IssuerId#57059, IssuerId2#57058, issueridcomposite#57057]

+- Project [BusinessDate#57031, stcd#57060, srcnm#57061, NetworkName#57034, NetworkURL#57035, custnum#57036, MarketCoverage#57037, DentalOnlyPlan#57038, RejectRows#57039, IssuerId#57059, IssuerId2#57058, concat(IssuerId#57059, IssuerId2#57058) AS issueridcomposite#57057]

+- Project [BusinessDate#57031, stcd#57060, srcnm#57061, NetworkName#57034, NetworkURL#57035, custnum#57036, MarketCoverage#57037, DentalOnlyPlan#57038, RejectRows#57039, IssuerId#57059, IssuerId2#57058]

+- Project [IssuerIDDD#8273, BusinessDate#57031, stcd#57060, srcnm#57061, NetworkName#57034, NetworkURL#57035, custnum#57036, MarketCoverage#57037, DentalOnlyPlan#57038, RejectRows#57039, IssuerId#57059, IssuerId2#57058]

+- Project [IssuerIDD#8261, IssuerIDDD#8273, BusinessDate#57031, stcd#57060, srcnm#57061, NetworkName#57034, NetworkURL#57035, custnum#57036, MarketCoverage#57037, DentalOnlyPlan#57038, RejectRows#57039, IssuerId#57059, cast(IssuerIDDD#8273 as string) AS IssuerId2#57058]

+- Project [IssuerIDD#8261, IssuerIDDD#8273, BusinessDate#57031, stcd#57060, srcnm#57061, NetworkName#57034, NetworkURL#57035, custnum#57036, MarketCoverage#57037, DentalOnlyPlan#57038, RejectRows#57039, cast(IssuerIDD#8261 as string) AS IssuerId#57059]

+- Project [IssuerIDD#8261, IssuerId2#57030 AS IssuerIDDD#8273, BusinessDate#57031, stcd#57060, srcnm#57061, NetworkName#57034, NetworkURL#57035, custnum#57036, MarketCoverage#57037, DentalOnlyPlan#57038, RejectRows#57039]

+- Project [IssuerId#57029 AS IssuerIDD#8261, IssuerId2#57030, BusinessDate#57031, stcd#57060, srcnm#57061, NetworkName#57034, NetworkURL#57035, custnum#57036, MarketCoverage#57037, DentalOnlyPlan#57038, RejectRows#57039]

+- Project [IssuerId#57029, IssuerId2#57030, BusinessDate#57031, StateCode#57032 AS stcd#57060, srcnm#57061, NetworkName#57034, NetworkURL#57035, custnum#57036, MarketCoverage#57037, DentalOnlyPlan#57038, RejectRows#57039]

+- Project [IssuerId#57029, IssuerId2#57030, BusinessDate#57031, StateCode#57032, SourceName#57033 AS srcnm#57061, NetworkName#57034, NetworkURL#57035, custnum#57036, MarketCoverage#57037, DentalOnlyPlan#57038, RejectRows#57039]

+- Union false, false

:- Relation [IssuerId#57029,IssuerId2#57030,BusinessDate#57031,StateCode#57032,SourceName#57033,NetworkName#57034,NetworkURL#57035,custnum#57036,MarketCoverage#57037,DentalOnlyPlan#57038,RejectRows#57039] csv

+- Relation [IssuerId#57040,IssuerId2#57041,BusinessDate#57042,StateCode#57043,SourceName#57044,NetworkName#57045,NetworkURL#57046,custnum#57047,MarketCoverage#57048,DentalOnlyPlan#57049,RejectRows#57050] csv

>>>

>>> WITH CountData AS (

File "<stdin>", line 1

WITH CountData AS (

^

SyntaxError: invalid syntax

>>> SELECT

File "<stdin>", line 1

SELECT

^

IndentationError: unexpected indent

>>> statedesc,

File "<stdin>", line 1

statedesc,

^

IndentationError: unexpected indent

>>> Protocol\_Used AS protocol,

File "<stdin>", line 1

Protocol\_Used AS protocol,

^

IndentationError: unexpected indent

>>> profession,

File "<stdin>", line 1

profession,

^

IndentationError: unexpected indent

>>> AVG(age) AS avg\_age,

File "<stdin>", line 1

AVG(age) AS avg\_age,

^

IndentationError: unexpected indent

>>> COUNT(\*) AS profession\_count,

File "<stdin>", line 1

COUNT(\*) AS profession\_count,

^

IndentationError: unexpected indent

>>> ROW\_NUMBER() OVER (PARTITION BY statedesc, protocol ORDER BY COUNT(\*) DESC) AS seqno

File "<stdin>", line 1

ROW\_NUMBER() OVER (PARTITION BY statedesc, protocol ORDER BY COUNT(\*) DESC) AS seqno

^

IndentationError: unexpected indent

>>> FROM view1

File "<stdin>", line 1

FROM view1

^

IndentationError: unexpected indent

>>> GROUP BY statedesc, protocol, profession

File "<stdin>", line 1

GROUP BY statedesc, protocol, profession

^

IndentationError: unexpected indent

>>> )

File "<stdin>", line 1

)

^

SyntaxError: invalid syntax

>>>

>>> SELECT

Traceback (most recent call last):

File "<stdin>", line 1, in <module>

NameError: name 'SELECT' is not defined

>>> statedesc,

File "<stdin>", line 1

statedesc,

^

IndentationError: unexpected indent

>>> protocol,

File "<stdin>", line 1

protocol,

^

IndentationError: unexpected indent

>>> profession,

File "<stdin>", line 1

profession,

^

IndentationError: unexpected indent

>>> avg\_age,

File "<stdin>", line 1

avg\_age,

^

IndentationError: unexpected indent

>>> profession\_count

File "<stdin>", line 1

profession\_count

^

IndentationError: unexpected indent

>>> FROM CountData

File "<stdin>", line 1

FROM CountData

^

SyntaxError: invalid syntax

>>> WHERE seqno = 2

>>> from pyspark.sql.functions import col, avg, count

>>> from pyspark.sql.window import Window

>>> from pyspark.sql.functions import row\_number

>>>

>>> # Calculate average age and count, and add seqno

... window\_spec = Window.partitionBy("stcd", "Protocol\_Used").orderBy(col("profession\_count").desc())

>>> count\_df = df.groupBy("stcd", "Protocol\_Used", "profession") \

... .agg(

... avg("age").alias("avg\_age"),

... count("\*").alias("profession\_count")

... ) \

... .withColumn("seqno", row\_number().over(window\_spec))

Traceback (most recent call last):

File "<stdin>", line 1, in <module>

NameError: name 'df' is not defined

>>> count\_df = df\_renamed.groupBy("stcd", "Protocol\_Used", "profession") \

... .agg(

... avg("age").alias("avg\_age"),

... count("\*").alias("profession\_count")

... ) \

... .withColumn("seqno", row\_number().over(window\_spec))

Traceback (most recent call last):

File "<stdin>", line 4, in <module>

File "/usr/local/spark/python/pyspark/sql/group.py", line 119, in agg

\_to\_seq(self.sql\_ctx.\_sc, [c.\_jc for c in exprs[1:]]))

File "/usr/local/spark/python/lib/py4j-0.10.9.3-src.zip/py4j/java\_gateway.py", line 1322, in \_\_call\_\_

File "/usr/local/spark/python/pyspark/sql/utils.py", line 117, in deco

raise converted from None

pyspark.sql.utils.AnalysisException: cannot resolve 'Protocol\_Used' given input columns: [l.BusinessDate, City, City, l.IssuerId, l.IssuerId2, l.MarketCoverage, l.NetworkName, l.NetworkURL, l.RejectRows, r.\_c2, r.\_c3, r.\_c4, l.cleannetworkname, l.cleannetworkname, l.curdt, l.curts, l.custnum, l.issueridcomposite, l.srcnm, state\_code, state\_code, l.stcd, l.sysdt, l.systs];

'Aggregate [stcd#7759, 'Protocol\_Used, 'profession], [stcd#7759, 'Protocol\_Used, 'profession, avg('age) AS avg\_age#57112, count(1) AS profession\_count#57114L]

+- Project [BusinessDate#1378, stcd#7759, srcnm#7747, NetworkName#1381, NetworkURL#1382, custnum#1383, MarketCoverage#1384, RejectRows#1386, IssuerId#8312, IssuerId2#8325, issueridcomposite#8901, sysdt#9003, systs#9016, cleannetworkname#47665, curdt#48091, curts#48092, cleannetworkname#48111, state\_code#53694, \_c1#26008 AS City#53720, state\_code#53695, \_c1#52125 AS City#53721, \_c2#52126, \_c3#52127, \_c4#52128]

+- Project [BusinessDate#1378, stcd#7759, srcnm#7747, NetworkName#1381, NetworkURL#1382, custnum#1383, MarketCoverage#1384, RejectRows#1386, IssuerId#8312, IssuerId2#8325, issueridcomposite#8901, sysdt#9003, systs#9016, cleannetworkname#47665, curdt#48091, curts#48092, cleannetworkname#48111, \_c0#26007 AS state\_code#53694, \_c1#26008, \_c0#52124 AS state\_code#53695, \_c1#52125, \_c2#52126, \_c3#52127, \_c4#52128]

+- Project [BusinessDate#1378, stcd#7759, srcnm#7747, NetworkName#1381, NetworkURL#1382, custnum#1383, MarketCoverage#1384, RejectRows#1386, IssuerId#8312, IssuerId2#8325, issueridcomposite#8901, sysdt#9003, systs#9016, cleannetworkname#47665, curdt#48091, curts#48092, cleannetworkname#48111, \_c0#26007, \_c1#26008, \_c0#52124, \_c1#52125, \_c2#52126, \_c3#52127, \_c4#52128]

+- Join Inner, (cast(custnum#1383 as int) = \_c3#52127)

:- SubqueryAlias l

: +- SubqueryAlias insureview

: +- View (`insureview`, [BusinessDate#1378,stcd#7759,srcnm#7747,NetworkName#1381,NetworkURL#1382,custnum#1383,MarketCoverage#1384,RejectRows#1386,IssuerId#8312,IssuerId2#8325,issueridcomposite#8901,sysdt#9003,systs#9016,cleannetworkname#47665,curdt#48091,curts#48092,cleannetworkname#48111,\_c0#26007,\_c1#26008])

: +- Project [BusinessDate#1378, stcd#7759, srcnm#7747, NetworkName#1381, NetworkURL#1382, custnum#1383, MarketCoverage#1384, RejectRows#1386, IssuerId#8312, IssuerId2#8325, issueridcomposite#8901, sysdt#9003, systs#9016, cleannetworkname#47665, curdt#48091, curts#48092, cleannetworkname#48111, \_c0#26007, \_c1#26008]

: +- Join Inner, (stcd#7759 = \_c0#26007)

: :- SubqueryAlias l

: : +- SubqueryAlias insureview

: : +- View (`insureview`, [BusinessDate#1378,stcd#7759,srcnm#7747,NetworkName#1381,NetworkURL#1382,custnum#1383,MarketCoverage#1384,RejectRows#1386,IssuerId#8312,IssuerId2#8325,issueridcomposite#8901,sysdt#9003,systs#9016,cleannetworkname#47665,curdt#48091,curts#48092,cleannetworkname#48111])

: : +- Project [BusinessDate#1378, stcd#7759, srcnm#7747, NetworkName#1381, NetworkURL#1382, custnum#1383, MarketCoverage#1384, RejectRows#1386, IssuerId#8312, IssuerId2#8325, issueridcomposite#8901, sysdt#9003, systs#9016, cleannetworkname#47665, curdt#48091, curts#48092, sql\_function(NetworkName#1381) AS cleannetworkname#48111]

: : +- SubqueryAlias insureview

: : +- View (`insureview`, [BusinessDate#1378,stcd#7759,srcnm#7747,NetworkName#1381,NetworkURL#1382,custnum#1383,MarketCoverage#1384,RejectRows#1386,IssuerId#8312,IssuerId2#8325,issueridcomposite#8901,sysdt#9003,systs#9016,cleannetworkname#47665,curdt#48091,curts#48092])

: : +- Project [BusinessDate#1378, stcd#7759, srcnm#7747, NetworkName#1381, NetworkURL#1382, custnum#1383, MarketCoverage#1384, RejectRows#1386, IssuerId#8312, IssuerId2#8325, issueridcomposite#8901, sysdt#9003, systs#9016, cleannetworkname#47665, current\_date(Some(Asia/Kolkata)) AS curdt#48091, current\_timestamp() AS curts#48092]

: : +- SubqueryAlias insureview

: : +- View (`insureview`, [BusinessDate#1378,stcd#7759,srcnm#7747,NetworkName#1381,NetworkURL#1382,custnum#1383,MarketCoverage#1384,RejectRows#1386,IssuerId#8312,IssuerId2#8325,issueridcomposite#8901,sysdt#9003,systs#9016,cleannetworkname#47665])

: : +- Project [BusinessDate#1378, stcd#7759, srcnm#7747, NetworkName#1381, NetworkURL#1382, custnum#1383, MarketCoverage#1384, RejectRows#1386, IssuerId#8312, IssuerId2#8325, issueridcomposite#8901, sysdt#9003, systs#9016, sql\_function(NetworkName#1381) AS cleannetworkname#47665]

: : +- SubqueryAlias insureview

: : +- View (`insureview`, [BusinessDate#1378,stcd#7759,srcnm#7747,NetworkName#1381,NetworkURL#1382,custnum#1383,MarketCoverage#1384,RejectRows#1386,IssuerId#8312,IssuerId2#8325,issueridcomposite#8901,sysdt#9003,systs#9016])

: : +- Project [BusinessDate#1378, stcd#7759, srcnm#7747, NetworkName#1381, NetworkURL#1382, custnum#1383, MarketCoverage#1384, RejectRows#1386, IssuerId#8312, IssuerId2#8325, issueridcomposite#8901, sysdt#9003, current\_timestamp() AS systs#9016]

: : +- Project [BusinessDate#1378, stcd#7759, srcnm#7747, NetworkName#1381, NetworkURL#1382, custnum#1383, MarketCoverage#1384, RejectRows#1386, IssuerId#8312, IssuerId2#8325, issueridcomposite#8901, current\_date(Some(Asia/Kolkata)) AS sysdt#9003]

: : +- Project [BusinessDate#1378, stcd#7759, srcnm#7747, NetworkName#1381, NetworkURL#1382, custnum#1383, MarketCoverage#1384, RejectRows#1386, IssuerId#8312, IssuerId2#8325, issueridcomposite#8901]

: : +- Project [BusinessDate#1378, stcd#7759, srcnm#7747, NetworkName#1381, NetworkURL#1382, custnum#1383, MarketCoverage#1384, DentalOnlyPlan#1385, RejectRows#1386, IssuerId#8312, IssuerId2#8325, concat(IssuerId#8312, IssuerId2#8325) AS issueridcomposite#8901]

: : +- Project [BusinessDate#1378, stcd#7759, srcnm#7747, NetworkName#1381, NetworkURL#1382, custnum#1383, MarketCoverage#1384, DentalOnlyPlan#1385, RejectRows#1386, IssuerId#8312, IssuerId2#8325]

: : +- Project [IssuerIDDD#8273, BusinessDate#1378, stcd#7759, srcnm#7747, NetworkName#1381, NetworkURL#1382, custnum#1383, MarketCoverage#1384, DentalOnlyPlan#1385, RejectRows#1386, IssuerId#8312, IssuerId2#8325]

: : +- Project [IssuerIDD#8261, IssuerIDDD#8273, BusinessDate#1378, stcd#7759, srcnm#7747, NetworkName#1381, NetworkURL#1382, custnum#1383, MarketCoverage#1384, DentalOnlyPlan#1385, RejectRows#1386, IssuerId#8312, cast(IssuerIDDD#8273 as string) AS IssuerId2#8325]

: : +- Project [IssuerIDD#8261, IssuerIDDD#8273, BusinessDate#1378, stcd#7759, srcnm#7747, NetworkName#1381, NetworkURL#1382, custnum#1383, MarketCoverage#1384, DentalOnlyPlan#1385, RejectRows#1386, cast(IssuerIDD#8261 as string) AS IssuerId#8312]

: : +- Project [IssuerIDD#8261, IssuerId2#1377 AS IssuerIDDD#8273, BusinessDate#1378, stcd#7759, srcnm#7747, NetworkName#1381, NetworkURL#1382, custnum#1383, MarketCoverage#1384, DentalOnlyPlan#1385, RejectRows#1386]

: : +- Project [IssuerId#1376 AS IssuerIDD#8261, IssuerId2#1377, BusinessDate#1378, stcd#7759, srcnm#7747, NetworkName#1381, NetworkURL#1382, custnum#1383, MarketCoverage#1384, DentalOnlyPlan#1385, RejectRows#1386]

: : +- Project [IssuerId#1376, IssuerId2#1377, BusinessDate#1378, StateCode#1379 AS stcd#7759, srcnm#7747, NetworkName#1381, NetworkURL#1382, custnum#1383, MarketCoverage#1384, DentalOnlyPlan#1385, RejectRows#1386]

: : +- Project [IssuerId#1376, IssuerId2#1377, BusinessDate#1378, StateCode#1379, SourceName#1380 AS srcnm#7747, NetworkName#1381, NetworkURL#1382, custnum#1383, MarketCoverage#1384, DentalOnlyPlan#1385, RejectRows#1386]

: : +- Union false, false

: : :- Relation [IssuerId#1376,IssuerId2#1377,BusinessDate#1378,StateCode#1379,SourceName#1380,NetworkName#1381,NetworkURL#1382,custnum#1383,MarketCoverage#1384,DentalOnlyPlan#1385,RejectRows#1386] csv

: : +- Relation [IssuerId#491,IssuerId2#492,BusinessDate#493,StateCode#494,SourceName#495,NetworkName#496,NetworkURL#497,custnum#498,MarketCoverage#499,DentalOnlyPlan#500,RejectRows#501] csv

: +- SubqueryAlias r

: +- SubqueryAlias state\_table

: +- View (`state\_table`, [\_c0#26007,\_c1#26008])

: +- Project [\_c0#26007, \_c1#26008]

: +- Filter ((isnull(\_c2#26009) AND isnull(\_c3#26010)) AND isnull(\_c4#26011))

: +- Project [\_c0#26007, \_c1#26008, \_c2#26009, \_c3#26010, \_c4#26011]

: +- Relation [\_c0#26007,\_c1#26008,\_c2#26009,\_c3#26010,\_c4#26011] csv

+- SubqueryAlias r

+- SubqueryAlias cust\_table

+- View (`cust\_table`, [\_c0#52124,\_c1#52125,\_c2#52126,\_c3#52127,\_c4#52128])

+- Filter (((isnotnull(\_c0#52124) AND isnotnull(\_c1#52125)) AND isnotnull(\_c2#52126)) AND (isnotnull(\_c3#52127) AND isnotnull(\_c4#52128)))

+- Project [\_c0#52124, \_c1#52125, \_c2#52126, \_c3#52127, \_c4#52128]

+- Relation [\_c0#52124,\_c1#52125,\_c2#52126,\_c3#52127,\_c4#52128] csv

>>>

>>> from pyspark.sql.functions import col, avg, count

>>> from pyspark.sql.window import Window

>>> from pyspark.sql.functions import row\_number

>>>

>>> # Calculate average age and count, and add seqno

... window\_spec = Window.partitionBy("stcd", "Protocol\_Used").orderBy(col("profession\_count").desc())

>>>

>>> count\_df = df.groupBy("stcd", "Protocol\_Used", "profession") \

... .agg(

... avg("\_c1").alias("avg\_age"),

... count("\*").alias("profession\_count")

... ) \

... .withColumn("seqno", row\_number().over(window\_spec))

Traceback (most recent call last):

File "<stdin>", line 1, in <module>

NameError: name 'df' is not defined

>>> count\_df = df\_renamed.groupBy("stcd", "Protocol\_Used", "profession") \

... .agg(

... avg("\_c1").alias("avg\_age"),

... count("\*").alias("profession\_count")

... ) \

... .withColumn("seqno", row\_number().over(window\_spec))

Traceback (most recent call last):

File "<stdin>", line 4, in <module>

File "/usr/local/spark/python/pyspark/sql/group.py", line 119, in agg

\_to\_seq(self.sql\_ctx.\_sc, [c.\_jc for c in exprs[1:]]))

File "/usr/local/spark/python/lib/py4j-0.10.9.3-src.zip/py4j/java\_gateway.py", line 1322, in \_\_call\_\_

File "/usr/local/spark/python/pyspark/sql/utils.py", line 117, in deco

raise converted from None

pyspark.sql.utils.AnalysisException: cannot resolve 'Protocol\_Used' given input columns: [l.BusinessDate, City, City, l.IssuerId, l.IssuerId2, l.MarketCoverage, l.NetworkName, l.NetworkURL, l.RejectRows, r.\_c2, r.\_c3, r.\_c4, l.cleannetworkname, l.cleannetworkname, l.curdt, l.curts, l.custnum, l.issueridcomposite, l.srcnm, state\_code, state\_code, l.stcd, l.sysdt, l.systs];

'Aggregate [stcd#7759, 'Protocol\_Used, 'profession], [stcd#7759, 'Protocol\_Used, 'profession, avg('\_c1) AS avg\_age#57140, count(1) AS profession\_count#57142L]

+- Project [BusinessDate#1378, stcd#7759, srcnm#7747, NetworkName#1381, NetworkURL#1382, custnum#1383, MarketCoverage#1384, RejectRows#1386, IssuerId#8312, IssuerId2#8325, issueridcomposite#8901, sysdt#9003, systs#9016, cleannetworkname#47665, curdt#48091, curts#48092, cleannetworkname#48111, state\_code#53694, \_c1#26008 AS City#53720, state\_code#53695, \_c1#52125 AS City#53721, \_c2#52126, \_c3#52127, \_c4#52128]

+- Project [BusinessDate#1378, stcd#7759, srcnm#7747, NetworkName#1381, NetworkURL#1382, custnum#1383, MarketCoverage#1384, RejectRows#1386, IssuerId#8312, IssuerId2#8325, issueridcomposite#8901, sysdt#9003, systs#9016, cleannetworkname#47665, curdt#48091, curts#48092, cleannetworkname#48111, \_c0#26007 AS state\_code#53694, \_c1#26008, \_c0#52124 AS state\_code#53695, \_c1#52125, \_c2#52126, \_c3#52127, \_c4#52128]

+- Project [BusinessDate#1378, stcd#7759, srcnm#7747, NetworkName#1381, NetworkURL#1382, custnum#1383, MarketCoverage#1384, RejectRows#1386, IssuerId#8312, IssuerId2#8325, issueridcomposite#8901, sysdt#9003, systs#9016, cleannetworkname#47665, curdt#48091, curts#48092, cleannetworkname#48111, \_c0#26007, \_c1#26008, \_c0#52124, \_c1#52125, \_c2#52126, \_c3#52127, \_c4#52128]

+- Join Inner, (cast(custnum#1383 as int) = \_c3#52127)

:- SubqueryAlias l

: +- SubqueryAlias insureview

: +- View (`insureview`, [BusinessDate#1378,stcd#7759,srcnm#7747,NetworkName#1381,NetworkURL#1382,custnum#1383,MarketCoverage#1384,RejectRows#1386,IssuerId#8312,IssuerId2#8325,issueridcomposite#8901,sysdt#9003,systs#9016,cleannetworkname#47665,curdt#48091,curts#48092,cleannetworkname#48111,\_c0#26007,\_c1#26008])

: +- Project [BusinessDate#1378, stcd#7759, srcnm#7747, NetworkName#1381, NetworkURL#1382, custnum#1383, MarketCoverage#1384, RejectRows#1386, IssuerId#8312, IssuerId2#8325, issueridcomposite#8901, sysdt#9003, systs#9016, cleannetworkname#47665, curdt#48091, curts#48092, cleannetworkname#48111, \_c0#26007, \_c1#26008]

: +- Join Inner, (stcd#7759 = \_c0#26007)

: :- SubqueryAlias l

: : +- SubqueryAlias insureview

: : +- View (`insureview`, [BusinessDate#1378,stcd#7759,srcnm#7747,NetworkName#1381,NetworkURL#1382,custnum#1383,MarketCoverage#1384,RejectRows#1386,IssuerId#8312,IssuerId2#8325,issueridcomposite#8901,sysdt#9003,systs#9016,cleannetworkname#47665,curdt#48091,curts#48092,cleannetworkname#48111])

: : +- Project [BusinessDate#1378, stcd#7759, srcnm#7747, NetworkName#1381, NetworkURL#1382, custnum#1383, MarketCoverage#1384, RejectRows#1386, IssuerId#8312, IssuerId2#8325, issueridcomposite#8901, sysdt#9003, systs#9016, cleannetworkname#47665, curdt#48091, curts#48092, sql\_function(NetworkName#1381) AS cleannetworkname#48111]

: : +- SubqueryAlias insureview

: : +- View (`insureview`, [BusinessDate#1378,stcd#7759,srcnm#7747,NetworkName#1381,NetworkURL#1382,custnum#1383,MarketCoverage#1384,RejectRows#1386,IssuerId#8312,IssuerId2#8325,issueridcomposite#8901,sysdt#9003,systs#9016,cleannetworkname#47665,curdt#48091,curts#48092])

: : +- Project [BusinessDate#1378, stcd#7759, srcnm#7747, NetworkName#1381, NetworkURL#1382, custnum#1383, MarketCoverage#1384, RejectRows#1386, IssuerId#8312, IssuerId2#8325, issueridcomposite#8901, sysdt#9003, systs#9016, cleannetworkname#47665, current\_date(Some(Asia/Kolkata)) AS curdt#48091, current\_timestamp() AS curts#48092]

: : +- SubqueryAlias insureview

: : +- View (`insureview`, [BusinessDate#1378,stcd#7759,srcnm#7747,NetworkName#1381,NetworkURL#1382,custnum#1383,MarketCoverage#1384,RejectRows#1386,IssuerId#8312,IssuerId2#8325,issueridcomposite#8901,sysdt#9003,systs#9016,cleannetworkname#47665])

: : +- Project [BusinessDate#1378, stcd#7759, srcnm#7747, NetworkName#1381, NetworkURL#1382, custnum#1383, MarketCoverage#1384, RejectRows#1386, IssuerId#8312, IssuerId2#8325, issueridcomposite#8901, sysdt#9003, systs#9016, sql\_function(NetworkName#1381) AS cleannetworkname#47665]

: : +- SubqueryAlias insureview

: : +- View (`insureview`, [BusinessDate#1378,stcd#7759,srcnm#7747,NetworkName#1381,NetworkURL#1382,custnum#1383,MarketCoverage#1384,RejectRows#1386,IssuerId#8312,IssuerId2#8325,issueridcomposite#8901,sysdt#9003,systs#9016])

: : +- Project [BusinessDate#1378, stcd#7759, srcnm#7747, NetworkName#1381, NetworkURL#1382, custnum#1383, MarketCoverage#1384, RejectRows#1386, IssuerId#8312, IssuerId2#8325, issueridcomposite#8901, sysdt#9003, current\_timestamp() AS systs#9016]

: : +- Project [BusinessDate#1378, stcd#7759, srcnm#7747, NetworkName#1381, NetworkURL#1382, custnum#1383, MarketCoverage#1384, RejectRows#1386, IssuerId#8312, IssuerId2#8325, issueridcomposite#8901, current\_date(Some(Asia/Kolkata)) AS sysdt#9003]

: : +- Project [BusinessDate#1378, stcd#7759, srcnm#7747, NetworkName#1381, NetworkURL#1382, custnum#1383, MarketCoverage#1384, RejectRows#1386, IssuerId#8312, IssuerId2#8325, issueridcomposite#8901]

: : +- Project [BusinessDate#1378, stcd#7759, srcnm#7747, NetworkName#1381, NetworkURL#1382, custnum#1383, MarketCoverage#1384, DentalOnlyPlan#1385, RejectRows#1386, IssuerId#8312, IssuerId2#8325, concat(IssuerId#8312, IssuerId2#8325) AS issueridcomposite#8901]

: : +- Project [BusinessDate#1378, stcd#7759, srcnm#7747, NetworkName#1381, NetworkURL#1382, custnum#1383, MarketCoverage#1384, DentalOnlyPlan#1385, RejectRows#1386, IssuerId#8312, IssuerId2#8325]

: : +- Project [IssuerIDDD#8273, BusinessDate#1378, stcd#7759, srcnm#7747, NetworkName#1381, NetworkURL#1382, custnum#1383, MarketCoverage#1384, DentalOnlyPlan#1385, RejectRows#1386, IssuerId#8312, IssuerId2#8325]

: : +- Project [IssuerIDD#8261, IssuerIDDD#8273, BusinessDate#1378, stcd#7759, srcnm#7747, NetworkName#1381, NetworkURL#1382, custnum#1383, MarketCoverage#1384, DentalOnlyPlan#1385, RejectRows#1386, IssuerId#8312, cast(IssuerIDDD#8273 as string) AS IssuerId2#8325]

: : +- Project [IssuerIDD#8261, IssuerIDDD#8273, BusinessDate#1378, stcd#7759, srcnm#7747, NetworkName#1381, NetworkURL#1382, custnum#1383, MarketCoverage#1384, DentalOnlyPlan#1385, RejectRows#1386, cast(IssuerIDD#8261 as string) AS IssuerId#8312]

: : +- Project [IssuerIDD#8261, IssuerId2#1377 AS IssuerIDDD#8273, BusinessDate#1378, stcd#7759, srcnm#7747, NetworkName#1381, NetworkURL#1382, custnum#1383, MarketCoverage#1384, DentalOnlyPlan#1385, RejectRows#1386]

: : +- Project [IssuerId#1376 AS IssuerIDD#8261, IssuerId2#1377, BusinessDate#1378, stcd#7759, srcnm#7747, NetworkName#1381, NetworkURL#1382, custnum#1383, MarketCoverage#1384, DentalOnlyPlan#1385, RejectRows#1386]

: : +- Project [IssuerId#1376, IssuerId2#1377, BusinessDate#1378, StateCode#1379 AS stcd#7759, srcnm#7747, NetworkName#1381, NetworkURL#1382, custnum#1383, MarketCoverage#1384, DentalOnlyPlan#1385, RejectRows#1386]

: : +- Project [IssuerId#1376, IssuerId2#1377, BusinessDate#1378, StateCode#1379, SourceName#1380 AS srcnm#7747, NetworkName#1381, NetworkURL#1382, custnum#1383, MarketCoverage#1384, DentalOnlyPlan#1385, RejectRows#1386]

: : +- Union false, false

: : :- Relation [IssuerId#1376,IssuerId2#1377,BusinessDate#1378,StateCode#1379,SourceName#1380,NetworkName#1381,NetworkURL#1382,custnum#1383,MarketCoverage#1384,DentalOnlyPlan#1385,RejectRows#1386] csv

: : +- Relation [IssuerId#491,IssuerId2#492,BusinessDate#493,StateCode#494,SourceName#495,NetworkName#496,NetworkURL#497,custnum#498,MarketCoverage#499,DentalOnlyPlan#500,RejectRows#501] csv

: +- SubqueryAlias r

: +- SubqueryAlias state\_table

: +- View (`state\_table`, [\_c0#26007,\_c1#26008])

: +- Project [\_c0#26007, \_c1#26008]

: +- Filter ((isnull(\_c2#26009) AND isnull(\_c3#26010)) AND isnull(\_c4#26011))

: +- Project [\_c0#26007, \_c1#26008, \_c2#26009, \_c3#26010, \_c4#26011]

: +- Relation [\_c0#26007,\_c1#26008,\_c2#26009,\_c3#26010,\_c4#26011] csv

+- SubqueryAlias r

+- SubqueryAlias cust\_table

+- View (`cust\_table`, [\_c0#52124,\_c1#52125,\_c2#52126,\_c3#52127,\_c4#52128])

+- Filter (((isnotnull(\_c0#52124) AND isnotnull(\_c1#52125)) AND isnotnull(\_c2#52126)) AND (isnotnull(\_c3#52127) AND isnotnull(\_c4#52128)))

+- Project [\_c0#52124, \_c1#52125, \_c2#52126, \_c3#52127, \_c4#52128]

+- Relation [\_c0#52124,\_c1#52125,\_c2#52126,\_c3#52127,\_c4#52128] csv

>>> second\_join.show(1)

+------------+----+-----+-----------+--------------------+-------+--------------+----------+--------+---------+-----------------+----------+--------------------+----------------+----------+--------------------+----------------+---+-------+---+-----+-------+---+-----+-------------+

|BusinessDate|stcd|srcnm|NetworkName| NetworkURL|custnum|MarketCoverage|RejectRows|IssuerId|IssuerId2|issueridcomposite| sysdt| systs|cleannetworkname| curdt| curts|cleannetworkname|\_c0| \_c1|\_c0| \_c1| \_c2|\_c3| \_c4|Protocol\_Used|

+------------+----+-----+-----------+--------------------+-------+--------------+----------+--------+---------+-----------------+----------+--------------------+----------------+----------+--------------------+----------------+---+-------+---+-----+-------+---+-----+-------------+

| 2019-10-01| AZ| HIOS| La Paz|http://uamarketpl...| 21| null| null| 88925| 88925| 8892588925|2024-07-22|2024-07-22 16:54:...| La Paz|2024-07-22|2024-07-22 16:54:...| La Paz| AZ|Arizona|779|Jason|Bowling| 21|Coach| http secured|

+------------+----+-----+-----------+--------------------+-------+--------------+----------+--------+---------+-----------------+----------+--------------------+----------------+----------+--------------------+----------------+---+-------+---+-----+-------+---+-----+-------------+

only showing top 1 row

>>> from pyspark.sql.functions import col, avg, count

>>> from pyspark.sql.window import Window

>>> from pyspark.sql.functions import row\_number

>>>

>>> # Calculate average age and count, and add seqno

... window\_spec = Window.partitionBy("stcd", "Protocol\_Used").orderBy(col("profession\_count").desc())

>>>

>>> count\_df = second\_join.groupBy("stcd", "Protocol\_Used", "profession") \

... .agg(

... avg("age").alias("avg\_age"),

... count("\*").alias("profession\_count")

... ) \

... .withColumn("seqno", row\_number().over(window\_spec))

Traceback (most recent call last):

File "<stdin>", line 4, in <module>

File "/usr/local/spark/python/pyspark/sql/group.py", line 119, in agg

\_to\_seq(self.sql\_ctx.\_sc, [c.\_jc for c in exprs[1:]]))

File "/usr/local/spark/python/lib/py4j-0.10.9.3-src.zip/py4j/java\_gateway.py", line 1322, in \_\_call\_\_

File "/usr/local/spark/python/pyspark/sql/utils.py", line 117, in deco

raise converted from None

pyspark.sql.utils.AnalysisException: cannot resolve 'profession' given input columns: [l.BusinessDate, l.IssuerId, l.IssuerId2, l.MarketCoverage, l.NetworkName, l.NetworkURL, r.Protocol\_Used, l.RejectRows, l.\_c0, l.\_c0, l.\_c1, l.\_c1, l.\_c2, l.\_c3, l.\_c4, l.cleannetworkname, l.cleannetworkname, l.curdt, l.curts, l.custnum, l.issueridcomposite, l.srcnm, l.stcd, l.sysdt, l.systs];

'Aggregate [stcd#7759, Protocol\_Used#48671, 'profession], [stcd#7759, Protocol\_Used#48671, 'profession, avg('age) AS avg\_age#58285, count(1) AS profession\_count#58287L]

+- Project [BusinessDate#1378, stcd#7759, srcnm#7747, NetworkName#1381, NetworkURL#1382, custnum#1383, MarketCoverage#1384, RejectRows#1386, IssuerId#8312, IssuerId2#8325, issueridcomposite#8901, sysdt#9003, systs#9016, cleannetworkname#47665, curdt#48091, curts#48092, cleannetworkname#48111, \_c0#26007, \_c1#26008, \_c0#52124, \_c1#52125, \_c2#52126, \_c3#52127, \_c4#52128, Protocol\_Used#48671]

+- Join Inner

:- SubqueryAlias l

: +- SubqueryAlias view1

: +- View (`view1`, [BusinessDate#1378,stcd#7759,srcnm#7747,NetworkName#1381,NetworkURL#1382,custnum#1383,MarketCoverage#1384,RejectRows#1386,IssuerId#8312,IssuerId2#8325,issueridcomposite#8901,sysdt#9003,systs#9016,cleannetworkname#47665,curdt#48091,curts#48092,cleannetworkname#48111,\_c0#26007,\_c1#26008,\_c0#52124,\_c1#52125,\_c2#52126,\_c3#52127,\_c4#52128])

: +- Project [BusinessDate#1378, stcd#7759, srcnm#7747, NetworkName#1381, NetworkURL#1382, custnum#1383, MarketCoverage#1384, RejectRows#1386, IssuerId#8312, IssuerId2#8325, issueridcomposite#8901, sysdt#9003, systs#9016, cleannetworkname#47665, curdt#48091, curts#48092, cleannetworkname#48111, \_c0#26007, \_c1#26008, \_c0#52124, \_c1#52125, \_c2#52126, \_c3#52127, \_c4#52128]

: +- Join Inner, (cast(custnum#1383 as int) = \_c3#52127)

: :- SubqueryAlias l

: : +- SubqueryAlias insureview

: : +- View (`insureview`, [BusinessDate#1378,stcd#7759,srcnm#7747,NetworkName#1381,NetworkURL#1382,custnum#1383,MarketCoverage#1384,RejectRows#1386,IssuerId#8312,IssuerId2#8325,issueridcomposite#8901,sysdt#9003,systs#9016,cleannetworkname#47665,curdt#48091,curts#48092,cleannetworkname#48111,\_c0#26007,\_c1#26008])

: : +- Project [BusinessDate#1378, stcd#7759, srcnm#7747, NetworkName#1381, NetworkURL#1382, custnum#1383, MarketCoverage#1384, RejectRows#1386, IssuerId#8312, IssuerId2#8325, issueridcomposite#8901, sysdt#9003, systs#9016, cleannetworkname#47665, curdt#48091, curts#48092, cleannetworkname#48111, \_c0#26007, \_c1#26008]

: : +- Join Inner, (stcd#7759 = \_c0#26007)

: : :- SubqueryAlias l

: : : +- SubqueryAlias insureview

: : : +- View (`insureview`, [BusinessDate#1378,stcd#7759,srcnm#7747,NetworkName#1381,NetworkURL#1382,custnum#1383,MarketCoverage#1384,RejectRows#1386,IssuerId#8312,IssuerId2#8325,issueridcomposite#8901,sysdt#9003,systs#9016,cleannetworkname#47665,curdt#48091,curts#48092,cleannetworkname#48111])

: : : +- Project [BusinessDate#1378, stcd#7759, srcnm#7747, NetworkName#1381, NetworkURL#1382, custnum#1383, MarketCoverage#1384, RejectRows#1386, IssuerId#8312, IssuerId2#8325, issueridcomposite#8901, sysdt#9003, systs#9016, cleannetworkname#47665, curdt#48091, curts#48092, sql\_function(NetworkName#1381) AS cleannetworkname#48111]

: : : +- SubqueryAlias insureview

: : : +- View (`insureview`, [BusinessDate#1378,stcd#7759,srcnm#7747,NetworkName#1381,NetworkURL#1382,custnum#1383,MarketCoverage#1384,RejectRows#1386,IssuerId#8312,IssuerId2#8325,issueridcomposite#8901,sysdt#9003,systs#9016,cleannetworkname#47665,curdt#48091,curts#48092])

: : : +- Project [BusinessDate#1378, stcd#7759, srcnm#7747, NetworkName#1381, NetworkURL#1382, custnum#1383, MarketCoverage#1384, RejectRows#1386, IssuerId#8312, IssuerId2#8325, issueridcomposite#8901, sysdt#9003, systs#9016, cleannetworkname#47665, current\_date(Some(Asia/Kolkata)) AS curdt#48091, current\_timestamp() AS curts#48092]

: : : +- SubqueryAlias insureview

: : : +- View (`insureview`, [BusinessDate#1378,stcd#7759,srcnm#7747,NetworkName#1381,NetworkURL#1382,custnum#1383,MarketCoverage#1384,RejectRows#1386,IssuerId#8312,IssuerId2#8325,issueridcomposite#8901,sysdt#9003,systs#9016,cleannetworkname#47665])

: : : +- Project [BusinessDate#1378, stcd#7759, srcnm#7747, NetworkName#1381, NetworkURL#1382, custnum#1383, MarketCoverage#1384, RejectRows#1386, IssuerId#8312, IssuerId2#8325, issueridcomposite#8901, sysdt#9003, systs#9016, sql\_function(NetworkName#1381) AS cleannetworkname#47665]

: : : +- SubqueryAlias insureview

: : : +- View (`insureview`, [BusinessDate#1378,stcd#7759,srcnm#7747,NetworkName#1381,NetworkURL#1382,custnum#1383,MarketCoverage#1384,RejectRows#1386,IssuerId#8312,IssuerId2#8325,issueridcomposite#8901,sysdt#9003,systs#9016])

: : : +- Project [BusinessDate#1378, stcd#7759, srcnm#7747, NetworkName#1381, NetworkURL#1382, custnum#1383, MarketCoverage#1384, RejectRows#1386, IssuerId#8312, IssuerId2#8325, issueridcomposite#8901, sysdt#9003, current\_timestamp() AS systs#9016]

: : : +- Project [BusinessDate#1378, stcd#7759, srcnm#7747, NetworkName#1381, NetworkURL#1382, custnum#1383, MarketCoverage#1384, RejectRows#1386, IssuerId#8312, IssuerId2#8325, issueridcomposite#8901, current\_date(Some(Asia/Kolkata)) AS sysdt#9003]

: : : +- Project [BusinessDate#1378, stcd#7759, srcnm#7747, NetworkName#1381, NetworkURL#1382, custnum#1383, MarketCoverage#1384, RejectRows#1386, IssuerId#8312, IssuerId2#8325, issueridcomposite#8901]

: : : +- Project [BusinessDate#1378, stcd#7759, srcnm#7747, NetworkName#1381, NetworkURL#1382, custnum#1383, MarketCoverage#1384, DentalOnlyPlan#1385, RejectRows#1386, IssuerId#8312, IssuerId2#8325, concat(IssuerId#8312, IssuerId2#8325) AS issueridcomposite#8901]

: : : +- Project [BusinessDate#1378, stcd#7759, srcnm#7747, NetworkName#1381, NetworkURL#1382, custnum#1383, MarketCoverage#1384, DentalOnlyPlan#1385, RejectRows#1386, IssuerId#8312, IssuerId2#8325]

: : : +- Project [IssuerIDDD#8273, BusinessDate#1378, stcd#7759, srcnm#7747, NetworkName#1381, NetworkURL#1382, custnum#1383, MarketCoverage#1384, DentalOnlyPlan#1385, RejectRows#1386, IssuerId#8312, IssuerId2#8325]

: : : +- Project [IssuerIDD#8261, IssuerIDDD#8273, BusinessDate#1378, stcd#7759, srcnm#7747, NetworkName#1381, NetworkURL#1382, custnum#1383, MarketCoverage#1384, DentalOnlyPlan#1385, RejectRows#1386, IssuerId#8312, cast(IssuerIDDD#8273 as string) AS IssuerId2#8325]

: : : +- Project [IssuerIDD#8261, IssuerIDDD#8273, BusinessDate#1378, stcd#7759, srcnm#7747, NetworkName#1381, NetworkURL#1382, custnum#1383, MarketCoverage#1384, DentalOnlyPlan#1385, RejectRows#1386, cast(IssuerIDD#8261 as string) AS IssuerId#8312]

: : : +- Project [IssuerIDD#8261, IssuerId2#1377 AS IssuerIDDD#8273, BusinessDate#1378, stcd#7759, srcnm#7747, NetworkName#1381, NetworkURL#1382, custnum#1383, MarketCoverage#1384, DentalOnlyPlan#1385, RejectRows#1386]

: : : +- Project [IssuerId#1376 AS IssuerIDD#8261, IssuerId2#1377, BusinessDate#1378, stcd#7759, srcnm#7747, NetworkName#1381, NetworkURL#1382, custnum#1383, MarketCoverage#1384, DentalOnlyPlan#1385, RejectRows#1386]

: : : +- Project [IssuerId#1376, IssuerId2#1377, BusinessDate#1378, StateCode#1379 AS stcd#7759, srcnm#7747, NetworkName#1381, NetworkURL#1382, custnum#1383, MarketCoverage#1384, DentalOnlyPlan#1385, RejectRows#1386]

: : : +- Project [IssuerId#1376, IssuerId2#1377, BusinessDate#1378, StateCode#1379, SourceName#1380 AS srcnm#7747, NetworkName#1381, NetworkURL#1382, custnum#1383, MarketCoverage#1384, DentalOnlyPlan#1385, RejectRows#1386]

: : : +- Union false, false

: : : :- Relation [IssuerId#1376,IssuerId2#1377,BusinessDate#1378,StateCode#1379,SourceName#1380,NetworkName#1381,NetworkURL#1382,custnum#1383,MarketCoverage#1384,DentalOnlyPlan#1385,RejectRows#1386] csv

: : : +- Relation [IssuerId#491,IssuerId2#492,BusinessDate#493,StateCode#494,SourceName#495,NetworkName#496,NetworkURL#497,custnum#498,MarketCoverage#499,DentalOnlyPlan#500,RejectRows#501] csv

: : +- SubqueryAlias r

: : +- SubqueryAlias state\_table

: : +- View (`state\_table`, [\_c0#26007,\_c1#26008])

: : +- Project [\_c0#26007, \_c1#26008]

: : +- Filter ((isnull(\_c2#26009) AND isnull(\_c3#26010)) AND isnull(\_c4#26011))

: : +- Project [\_c0#26007, \_c1#26008, \_c2#26009, \_c3#26010, \_c4#26011]

: : +- Relation [\_c0#26007,\_c1#26008,\_c2#26009,\_c3#26010,\_c4#26011] csv

: +- SubqueryAlias r

: +- SubqueryAlias cust\_table

: +- View (`cust\_table`, [\_c0#52124,\_c1#52125,\_c2#52126,\_c3#52127,\_c4#52128])

: +- Filter (((isnotnull(\_c0#52124) AND isnotnull(\_c1#52125)) AND isnotnull(\_c2#52126)) AND (isnotnull(\_c3#52127) AND isnotnull(\_c4#52128)))

: +- Project [\_c0#52124, \_c1#52125, \_c2#52126, \_c3#52127, \_c4#52128]

: +- Relation [\_c0#52124,\_c1#52125,\_c2#52126,\_c3#52127,\_c4#52128] csv

+- SubqueryAlias r

+- SubqueryAlias view2

+- View (`view2`, [BusinessDate#57031,stcd#57060,srcnm#57061,NetworkName#57034,NetworkURL#57035,custnum#57036,MarketCoverage#57037,RejectRows#57039,IssuerId#57059,IssuerId2#57058,issueridcomposite#57057,sysdt#57056,systs#57055,cleannetworkname#57054,curdt#57052,curts#57053,cleannetworkname#57051,Protocol\_Used#48671])

+- Project [BusinessDate#57031, stcd#57060, srcnm#57061, NetworkName#57034, NetworkURL#57035, custnum#57036, MarketCoverage#57037, RejectRows#57039, IssuerId#57059, IssuerId2#57058, issueridcomposite#57057, sysdt#57056, systs#57055, cleannetworkname#57054, curdt#57052, curts#57053, cleannetworkname#57051, CASE WHEN NetworkURL#57035 LIKE https://% THEN http secured WHEN NetworkURL#57035 LIKE http://% THEN http non secured ELSE No Protocol END AS Protocol\_Used#48671]

+- SubqueryAlias insureview

+- View (`insureview`, [BusinessDate#57031,stcd#57060,srcnm#57061,NetworkName#57034,NetworkURL#57035,custnum#57036,MarketCoverage#57037,RejectRows#57039,IssuerId#57059,IssuerId2#57058,issueridcomposite#57057,sysdt#57056,systs#57055,cleannetworkname#57054,curdt#57052,curts#57053,cleannetworkname#57051])

+- Project [BusinessDate#57031, stcd#57060, srcnm#57061, NetworkName#57034, NetworkURL#57035, custnum#57036, MarketCoverage#57037, RejectRows#57039, IssuerId#57059, IssuerId2#57058, issueridcomposite#57057, sysdt#57056, systs#57055, cleannetworkname#57054, curdt#57052, curts#57053, sql\_function(NetworkName#57034) AS cleannetworkname#57051]

+- SubqueryAlias insureview

+- View (`insureview`, [BusinessDate#57031,stcd#57060,srcnm#57061,NetworkName#57034,NetworkURL#57035,custnum#57036,MarketCoverage#57037,RejectRows#57039,IssuerId#57059,IssuerId2#57058,issueridcomposite#57057,sysdt#57056,systs#57055,cleannetworkname#57054,curdt#57052,curts#57053])

+- Project [BusinessDate#57031, stcd#57060, srcnm#57061, NetworkName#57034, NetworkURL#57035, custnum#57036, MarketCoverage#57037, RejectRows#57039, IssuerId#57059, IssuerId2#57058, issueridcomposite#57057, sysdt#57056, systs#57055, cleannetworkname#57054, current\_date(Some(Asia/Kolkata)) AS curdt#57052, current\_timestamp() AS curts#57053]

+- SubqueryAlias insureview

+- View (`insureview`, [BusinessDate#57031,stcd#57060,srcnm#57061,NetworkName#57034,NetworkURL#57035,custnum#57036,MarketCoverage#57037,RejectRows#57039,IssuerId#57059,IssuerId2#57058,issueridcomposite#57057,sysdt#57056,systs#57055,cleannetworkname#57054])

+- Project [BusinessDate#57031, stcd#57060, srcnm#57061, NetworkName#57034, NetworkURL#57035, custnum#57036, MarketCoverage#57037, RejectRows#57039, IssuerId#57059, IssuerId2#57058, issueridcomposite#57057, sysdt#57056, systs#57055, sql\_function(NetworkName#57034) AS cleannetworkname#57054]

+- SubqueryAlias insureview

+- View (`insureview`, [BusinessDate#57031,stcd#57060,srcnm#57061,NetworkName#57034,NetworkURL#57035,custnum#57036,MarketCoverage#57037,RejectRows#57039,IssuerId#57059,IssuerId2#57058,issueridcomposite#57057,sysdt#57056,systs#57055])

+- Project [BusinessDate#57031, stcd#57060, srcnm#57061, NetworkName#57034, NetworkURL#57035, custnum#57036, MarketCoverage#57037, RejectRows#57039, IssuerId#57059, IssuerId2#57058, issueridcomposite#57057, sysdt#57056, current\_timestamp() AS systs#57055]

+- Project [BusinessDate#57031, stcd#57060, srcnm#57061, NetworkName#57034, NetworkURL#57035, custnum#57036, MarketCoverage#57037, RejectRows#57039, IssuerId#57059, IssuerId2#57058, issueridcomposite#57057, current\_date(Some(Asia/Kolkata)) AS sysdt#57056]

+- Project [BusinessDate#57031, stcd#57060, srcnm#57061, NetworkName#57034, NetworkURL#57035, custnum#57036, MarketCoverage#57037, RejectRows#57039, IssuerId#57059, IssuerId2#57058, issueridcomposite#57057]

+- Project [BusinessDate#57031, stcd#57060, srcnm#57061, NetworkName#57034, NetworkURL#57035, custnum#57036, MarketCoverage#57037, DentalOnlyPlan#57038, RejectRows#57039, IssuerId#57059, IssuerId2#57058, concat(IssuerId#57059, IssuerId2#57058) AS issueridcomposite#57057]

+- Project [BusinessDate#57031, stcd#57060, srcnm#57061, NetworkName#57034, NetworkURL#57035, custnum#57036, MarketCoverage#57037, DentalOnlyPlan#57038, RejectRows#57039, IssuerId#57059, IssuerId2#57058]

+- Project [IssuerIDDD#8273, BusinessDate#57031, stcd#57060, srcnm#57061, NetworkName#57034, NetworkURL#57035, custnum#57036, MarketCoverage#57037, DentalOnlyPlan#57038, RejectRows#57039, IssuerId#57059, IssuerId2#57058]

+- Project [IssuerIDD#8261, IssuerIDDD#8273, BusinessDate#57031, stcd#57060, srcnm#57061, NetworkName#57034, NetworkURL#57035, custnum#57036, MarketCoverage#57037, DentalOnlyPlan#57038, RejectRows#57039, IssuerId#57059, cast(IssuerIDDD#8273 as string) AS IssuerId2#57058]

+- Project [IssuerIDD#8261, IssuerIDDD#8273, BusinessDate#57031, stcd#57060, srcnm#57061, NetworkName#57034, NetworkURL#57035, custnum#57036, MarketCoverage#57037, DentalOnlyPlan#57038, RejectRows#57039, cast(IssuerIDD#8261 as string) AS IssuerId#57059]

+- Project [IssuerIDD#8261, IssuerId2#57030 AS IssuerIDDD#8273, BusinessDate#57031, stcd#57060, srcnm#57061, NetworkName#57034, NetworkURL#57035, custnum#57036, MarketCoverage#57037, DentalOnlyPlan#57038, RejectRows#57039]

+- Project [IssuerId#57029 AS IssuerIDD#8261, IssuerId2#57030, BusinessDate#57031, stcd#57060, srcnm#57061, NetworkName#57034, NetworkURL#57035, custnum#57036, MarketCoverage#57037, DentalOnlyPlan#57038, RejectRows#57039]

+- Project [IssuerId#57029, IssuerId2#57030, BusinessDate#57031, StateCode#57032 AS stcd#57060, srcnm#57061, NetworkName#57034, NetworkURL#57035, custnum#57036, MarketCoverage#57037, DentalOnlyPlan#57038, RejectRows#57039]

+- Project [IssuerId#57029, IssuerId2#57030, BusinessDate#57031, StateCode#57032, SourceName#57033 AS srcnm#57061, NetworkName#57034, NetworkURL#57035, custnum#57036, MarketCoverage#57037, DentalOnlyPlan#57038, RejectRows#57039]

+- Union false, false

:- Relation [IssuerId#57029,IssuerId2#57030,BusinessDate#57031,StateCode#57032,SourceName#57033,NetworkName#57034,NetworkURL#57035,custnum#57036,MarketCoverage#57037,DentalOnlyPlan#57038,RejectRows#57039] csv

+- Relation [IssuerId#57040,IssuerId2#57041,BusinessDate#57042,StateCode#57043,SourceName#57044,NetworkName#57045,NetworkURL#57046,custnum#57047,MarketCoverage#57048,DentalOnlyPlan#57049,RejectRows#57050] csv

>>> count\_df = second\_join.groupBy("stcd", "Protocol\_Used", "profession") \

... .agg(

... avg("age").alias("avg\_age"),

... count("\*").alias("profession\_count")

... ) \

... .withColumn("seqno", row\_number().over(window\_spec))sdbksdv

File "<stdin>", line 6

.withColumn("seqno", row\_number().over(window\_spec))sdbksdv

^

SyntaxError: invalid syntax

>>> count\_df = second\_join.groupBy("stcd", "Protocol\_Used", "profession") \

... .agg(

... avg("\_c1").alias("avg\_age"),

... count("\*").alias("profession\_count")

... ) \

... .withColumn("seqno", row\_number().over(window\_spec))

Traceback (most recent call last):

File "<stdin>", line 4, in <module>

File "/usr/local/spark/python/pyspark/sql/group.py", line 119, in agg

\_to\_seq(self.sql\_ctx.\_sc, [c.\_jc for c in exprs[1:]]))

File "/usr/local/spark/python/lib/py4j-0.10.9.3-src.zip/py4j/java\_gateway.py", line 1322, in \_\_call\_\_

File "/usr/local/spark/python/pyspark/sql/utils.py", line 117, in deco

raise converted from None

pyspark.sql.utils.AnalysisException: Reference '\_c1' is ambiguous, could be: l.\_c1, l.\_c1.

>>> count\_df = second\_join.groupBy("stcd", "Protocol\_Used", "profession") \

... .agg(

... avg("\_c3").alias("avg\_age"),

... count("\*").alias("profession\_count")

... ) \

... .withColumn("seqno", row\_number().over(window\_spec))

Traceback (most recent call last):

File "<stdin>", line 4, in <module>

File "/usr/local/spark/python/pyspark/sql/group.py", line 119, in agg

\_to\_seq(self.sql\_ctx.\_sc, [c.\_jc for c in exprs[1:]]))

File "/usr/local/spark/python/lib/py4j-0.10.9.3-src.zip/py4j/java\_gateway.py", line 1322, in \_\_call\_\_

File "/usr/local/spark/python/pyspark/sql/utils.py", line 117, in deco

raise converted from None

pyspark.sql.utils.AnalysisException: cannot resolve 'profession' given input columns: [l.BusinessDate, l.IssuerId, l.IssuerId2, l.MarketCoverage, l.NetworkName, l.NetworkURL, r.Protocol\_Used, l.RejectRows, l.\_c0, l.\_c0, l.\_c1, l.\_c1, l.\_c2, l.\_c3, l.\_c4, l.cleannetworkname, l.cleannetworkname, l.curdt, l.curts, l.custnum, l.issueridcomposite, l.srcnm, l.stcd, l.sysdt, l.systs];

'Aggregate [stcd#7759, Protocol\_Used#48671, 'profession], [stcd#7759, Protocol\_Used#48671, 'profession, avg(\_c3#52127) AS avg\_age#58343, count(1) AS profession\_count#58345L]

+- Project [BusinessDate#1378, stcd#7759, srcnm#7747, NetworkName#1381, NetworkURL#1382, custnum#1383, MarketCoverage#1384, RejectRows#1386, IssuerId#8312, IssuerId2#8325, issueridcomposite#8901, sysdt#9003, systs#9016, cleannetworkname#47665, curdt#48091, curts#48092, cleannetworkname#48111, \_c0#26007, \_c1#26008, \_c0#52124, \_c1#52125, \_c2#52126, \_c3#52127, \_c4#52128, Protocol\_Used#48671]

+- Join Inner

:- SubqueryAlias l

: +- SubqueryAlias view1

: +- View (`view1`, [BusinessDate#1378,stcd#7759,srcnm#7747,NetworkName#1381,NetworkURL#1382,custnum#1383,MarketCoverage#1384,RejectRows#1386,IssuerId#8312,IssuerId2#8325,issueridcomposite#8901,sysdt#9003,systs#9016,cleannetworkname#47665,curdt#48091,curts#48092,cleannetworkname#48111,\_c0#26007,\_c1#26008,\_c0#52124,\_c1#52125,\_c2#52126,\_c3#52127,\_c4#52128])

: +- Project [BusinessDate#1378, stcd#7759, srcnm#7747, NetworkName#1381, NetworkURL#1382, custnum#1383, MarketCoverage#1384, RejectRows#1386, IssuerId#8312, IssuerId2#8325, issueridcomposite#8901, sysdt#9003, systs#9016, cleannetworkname#47665, curdt#48091, curts#48092, cleannetworkname#48111, \_c0#26007, \_c1#26008, \_c0#52124, \_c1#52125, \_c2#52126, \_c3#52127, \_c4#52128]

: +- Join Inner, (cast(custnum#1383 as int) = \_c3#52127)

: :- SubqueryAlias l

: : +- SubqueryAlias insureview

: : +- View (`insureview`, [BusinessDate#1378,stcd#7759,srcnm#7747,NetworkName#1381,NetworkURL#1382,custnum#1383,MarketCoverage#1384,RejectRows#1386,IssuerId#8312,IssuerId2#8325,issueridcomposite#8901,sysdt#9003,systs#9016,cleannetworkname#47665,curdt#48091,curts#48092,cleannetworkname#48111,\_c0#26007,\_c1#26008])

: : +- Project [BusinessDate#1378, stcd#7759, srcnm#7747, NetworkName#1381, NetworkURL#1382, custnum#1383, MarketCoverage#1384, RejectRows#1386, IssuerId#8312, IssuerId2#8325, issueridcomposite#8901, sysdt#9003, systs#9016, cleannetworkname#47665, curdt#48091, curts#48092, cleannetworkname#48111, \_c0#26007, \_c1#26008]

: : +- Join Inner, (stcd#7759 = \_c0#26007)

: : :- SubqueryAlias l

: : : +- SubqueryAlias insureview

: : : +- View (`insureview`, [BusinessDate#1378,stcd#7759,srcnm#7747,NetworkName#1381,NetworkURL#1382,custnum#1383,MarketCoverage#1384,RejectRows#1386,IssuerId#8312,IssuerId2#8325,issueridcomposite#8901,sysdt#9003,systs#9016,cleannetworkname#47665,curdt#48091,curts#48092,cleannetworkname#48111])

: : : +- Project [BusinessDate#1378, stcd#7759, srcnm#7747, NetworkName#1381, NetworkURL#1382, custnum#1383, MarketCoverage#1384, RejectRows#1386, IssuerId#8312, IssuerId2#8325, issueridcomposite#8901, sysdt#9003, systs#9016, cleannetworkname#47665, curdt#48091, curts#48092, sql\_function(NetworkName#1381) AS cleannetworkname#48111]

: : : +- SubqueryAlias insureview

: : : +- View (`insureview`, [BusinessDate#1378,stcd#7759,srcnm#7747,NetworkName#1381,NetworkURL#1382,custnum#1383,MarketCoverage#1384,RejectRows#1386,IssuerId#8312,IssuerId2#8325,issueridcomposite#8901,sysdt#9003,systs#9016,cleannetworkname#47665,curdt#48091,curts#48092])

: : : +- Project [BusinessDate#1378, stcd#7759, srcnm#7747, NetworkName#1381, NetworkURL#1382, custnum#1383, MarketCoverage#1384, RejectRows#1386, IssuerId#8312, IssuerId2#8325, issueridcomposite#8901, sysdt#9003, systs#9016, cleannetworkname#47665, current\_date(Some(Asia/Kolkata)) AS curdt#48091, current\_timestamp() AS curts#48092]

: : : +- SubqueryAlias insureview

: : : +- View (`insureview`, [BusinessDate#1378,stcd#7759,srcnm#7747,NetworkName#1381,NetworkURL#1382,custnum#1383,MarketCoverage#1384,RejectRows#1386,IssuerId#8312,IssuerId2#8325,issueridcomposite#8901,sysdt#9003,systs#9016,cleannetworkname#47665])

: : : +- Project [BusinessDate#1378, stcd#7759, srcnm#7747, NetworkName#1381, NetworkURL#1382, custnum#1383, MarketCoverage#1384, RejectRows#1386, IssuerId#8312, IssuerId2#8325, issueridcomposite#8901, sysdt#9003, systs#9016, sql\_function(NetworkName#1381) AS cleannetworkname#47665]

: : : +- SubqueryAlias insureview

: : : +- View (`insureview`, [BusinessDate#1378,stcd#7759,srcnm#7747,NetworkName#1381,NetworkURL#1382,custnum#1383,MarketCoverage#1384,RejectRows#1386,IssuerId#8312,IssuerId2#8325,issueridcomposite#8901,sysdt#9003,systs#9016])

: : : +- Project [BusinessDate#1378, stcd#7759, srcnm#7747, NetworkName#1381, NetworkURL#1382, custnum#1383, MarketCoverage#1384, RejectRows#1386, IssuerId#8312, IssuerId2#8325, issueridcomposite#8901, sysdt#9003, current\_timestamp() AS systs#9016]

: : : +- Project [BusinessDate#1378, stcd#7759, srcnm#7747, NetworkName#1381, NetworkURL#1382, custnum#1383, MarketCoverage#1384, RejectRows#1386, IssuerId#8312, IssuerId2#8325, issueridcomposite#8901, current\_date(Some(Asia/Kolkata)) AS sysdt#9003]

: : : +- Project [BusinessDate#1378, stcd#7759, srcnm#7747, NetworkName#1381, NetworkURL#1382, custnum#1383, MarketCoverage#1384, RejectRows#1386, IssuerId#8312, IssuerId2#8325, issueridcomposite#8901]

: : : +- Project [BusinessDate#1378, stcd#7759, srcnm#7747, NetworkName#1381, NetworkURL#1382, custnum#1383, MarketCoverage#1384, DentalOnlyPlan#1385, RejectRows#1386, IssuerId#8312, IssuerId2#8325, concat(IssuerId#8312, IssuerId2#8325) AS issueridcomposite#8901]

: : : +- Project [BusinessDate#1378, stcd#7759, srcnm#7747, NetworkName#1381, NetworkURL#1382, custnum#1383, MarketCoverage#1384, DentalOnlyPlan#1385, RejectRows#1386, IssuerId#8312, IssuerId2#8325]

: : : +- Project [IssuerIDDD#8273, BusinessDate#1378, stcd#7759, srcnm#7747, NetworkName#1381, NetworkURL#1382, custnum#1383, MarketCoverage#1384, DentalOnlyPlan#1385, RejectRows#1386, IssuerId#8312, IssuerId2#8325]

: : : +- Project [IssuerIDD#8261, IssuerIDDD#8273, BusinessDate#1378, stcd#7759, srcnm#7747, NetworkName#1381, NetworkURL#1382, custnum#1383, MarketCoverage#1384, DentalOnlyPlan#1385, RejectRows#1386, IssuerId#8312, cast(IssuerIDDD#8273 as string) AS IssuerId2#8325]

: : : +- Project [IssuerIDD#8261, IssuerIDDD#8273, BusinessDate#1378, stcd#7759, srcnm#7747, NetworkName#1381, NetworkURL#1382, custnum#1383, MarketCoverage#1384, DentalOnlyPlan#1385, RejectRows#1386, cast(IssuerIDD#8261 as string) AS IssuerId#8312]

: : : +- Project [IssuerIDD#8261, IssuerId2#1377 AS IssuerIDDD#8273, BusinessDate#1378, stcd#7759, srcnm#7747, NetworkName#1381, NetworkURL#1382, custnum#1383, MarketCoverage#1384, DentalOnlyPlan#1385, RejectRows#1386]

: : : +- Project [IssuerId#1376 AS IssuerIDD#8261, IssuerId2#1377, BusinessDate#1378, stcd#7759, srcnm#7747, NetworkName#1381, NetworkURL#1382, custnum#1383, MarketCoverage#1384, DentalOnlyPlan#1385, RejectRows#1386]

: : : +- Project [IssuerId#1376, IssuerId2#1377, BusinessDate#1378, StateCode#1379 AS stcd#7759, srcnm#7747, NetworkName#1381, NetworkURL#1382, custnum#1383, MarketCoverage#1384, DentalOnlyPlan#1385, RejectRows#1386]

: : : +- Project [IssuerId#1376, IssuerId2#1377, BusinessDate#1378, StateCode#1379, SourceName#1380 AS srcnm#7747, NetworkName#1381, NetworkURL#1382, custnum#1383, MarketCoverage#1384, DentalOnlyPlan#1385, RejectRows#1386]

: : : +- Union false, false

: : : :- Relation [IssuerId#1376,IssuerId2#1377,BusinessDate#1378,StateCode#1379,SourceName#1380,NetworkName#1381,NetworkURL#1382,custnum#1383,MarketCoverage#1384,DentalOnlyPlan#1385,RejectRows#1386] csv

: : : +- Relation [IssuerId#491,IssuerId2#492,BusinessDate#493,StateCode#494,SourceName#495,NetworkName#496,NetworkURL#497,custnum#498,MarketCoverage#499,DentalOnlyPlan#500,RejectRows#501] csv

: : +- SubqueryAlias r

: : +- SubqueryAlias state\_table

: : +- View (`state\_table`, [\_c0#26007,\_c1#26008])

: : +- Project [\_c0#26007, \_c1#26008]

: : +- Filter ((isnull(\_c2#26009) AND isnull(\_c3#26010)) AND isnull(\_c4#26011))

: : +- Project [\_c0#26007, \_c1#26008, \_c2#26009, \_c3#26010, \_c4#26011]

: : +- Relation [\_c0#26007,\_c1#26008,\_c2#26009,\_c3#26010,\_c4#26011] csv

: +- SubqueryAlias r

: +- SubqueryAlias cust\_table

: +- View (`cust\_table`, [\_c0#52124,\_c1#52125,\_c2#52126,\_c3#52127,\_c4#52128])

: +- Filter (((isnotnull(\_c0#52124) AND isnotnull(\_c1#52125)) AND isnotnull(\_c2#52126)) AND (isnotnull(\_c3#52127) AND isnotnull(\_c4#52128)))

: +- Project [\_c0#52124, \_c1#52125, \_c2#52126, \_c3#52127, \_c4#52128]

: +- Relation [\_c0#52124,\_c1#52125,\_c2#52126,\_c3#52127,\_c4#52128] csv

+- SubqueryAlias r

+- SubqueryAlias view2

+- View (`view2`, [BusinessDate#57031,stcd#57060,srcnm#57061,NetworkName#57034,NetworkURL#57035,custnum#57036,MarketCoverage#57037,RejectRows#57039,IssuerId#57059,IssuerId2#57058,issueridcomposite#57057,sysdt#57056,systs#57055,cleannetworkname#57054,curdt#57052,curts#57053,cleannetworkname#57051,Protocol\_Used#48671])

+- Project [BusinessDate#57031, stcd#57060, srcnm#57061, NetworkName#57034, NetworkURL#57035, custnum#57036, MarketCoverage#57037, RejectRows#57039, IssuerId#57059, IssuerId2#57058, issueridcomposite#57057, sysdt#57056, systs#57055, cleannetworkname#57054, curdt#57052, curts#57053, cleannetworkname#57051, CASE WHEN NetworkURL#57035 LIKE https://% THEN http secured WHEN NetworkURL#57035 LIKE http://% THEN http non secured ELSE No Protocol END AS Protocol\_Used#48671]

+- SubqueryAlias insureview

+- View (`insureview`, [BusinessDate#57031,stcd#57060,srcnm#57061,NetworkName#57034,NetworkURL#57035,custnum#57036,MarketCoverage#57037,RejectRows#57039,IssuerId#57059,IssuerId2#57058,issueridcomposite#57057,sysdt#57056,systs#57055,cleannetworkname#57054,curdt#57052,curts#57053,cleannetworkname#57051])

+- Project [BusinessDate#57031, stcd#57060, srcnm#57061, NetworkName#57034, NetworkURL#57035, custnum#57036, MarketCoverage#57037, RejectRows#57039, IssuerId#57059, IssuerId2#57058, issueridcomposite#57057, sysdt#57056, systs#57055, cleannetworkname#57054, curdt#57052, curts#57053, sql\_function(NetworkName#57034) AS cleannetworkname#57051]

+- SubqueryAlias insureview

+- View (`insureview`, [BusinessDate#57031,stcd#57060,srcnm#57061,NetworkName#57034,NetworkURL#57035,custnum#57036,MarketCoverage#57037,RejectRows#57039,IssuerId#57059,IssuerId2#57058,issueridcomposite#57057,sysdt#57056,systs#57055,cleannetworkname#57054,curdt#57052,curts#57053])

+- Project [BusinessDate#57031, stcd#57060, srcnm#57061, NetworkName#57034, NetworkURL#57035, custnum#57036, MarketCoverage#57037, RejectRows#57039, IssuerId#57059, IssuerId2#57058, issueridcomposite#57057, sysdt#57056, systs#57055, cleannetworkname#57054, current\_date(Some(Asia/Kolkata)) AS curdt#57052, current\_timestamp() AS curts#57053]

+- SubqueryAlias insureview

+- View (`insureview`, [BusinessDate#57031,stcd#57060,srcnm#57061,NetworkName#57034,NetworkURL#57035,custnum#57036,MarketCoverage#57037,RejectRows#57039,IssuerId#57059,IssuerId2#57058,issueridcomposite#57057,sysdt#57056,systs#57055,cleannetworkname#57054])

+- Project [BusinessDate#57031, stcd#57060, srcnm#57061, NetworkName#57034, NetworkURL#57035, custnum#57036, MarketCoverage#57037, RejectRows#57039, IssuerId#57059, IssuerId2#57058, issueridcomposite#57057, sysdt#57056, systs#57055, sql\_function(NetworkName#57034) AS cleannetworkname#57054]

+- SubqueryAlias insureview

+- View (`insureview`, [BusinessDate#57031,stcd#57060,srcnm#57061,NetworkName#57034,NetworkURL#57035,custnum#57036,MarketCoverage#57037,RejectRows#57039,IssuerId#57059,IssuerId2#57058,issueridcomposite#57057,sysdt#57056,systs#57055])

+- Project [BusinessDate#57031, stcd#57060, srcnm#57061, NetworkName#57034, NetworkURL#57035, custnum#57036, MarketCoverage#57037, RejectRows#57039, IssuerId#57059, IssuerId2#57058, issueridcomposite#57057, sysdt#57056, current\_timestamp() AS systs#57055]

+- Project [BusinessDate#57031, stcd#57060, srcnm#57061, NetworkName#57034, NetworkURL#57035, custnum#57036, MarketCoverage#57037, RejectRows#57039, IssuerId#57059, IssuerId2#57058, issueridcomposite#57057, current\_date(Some(Asia/Kolkata)) AS sysdt#57056]

+- Project [BusinessDate#57031, stcd#57060, srcnm#57061, NetworkName#57034, NetworkURL#57035, custnum#57036, MarketCoverage#57037, RejectRows#57039, IssuerId#57059, IssuerId2#57058, issueridcomposite#57057]

+- Project [BusinessDate#57031, stcd#57060, srcnm#57061, NetworkName#57034, NetworkURL#57035, custnum#57036, MarketCoverage#57037, DentalOnlyPlan#57038, RejectRows#57039, IssuerId#57059, IssuerId2#57058, concat(IssuerId#57059, IssuerId2#57058) AS issueridcomposite#57057]

+- Project [BusinessDate#57031, stcd#57060, srcnm#57061, NetworkName#57034, NetworkURL#57035, custnum#57036, MarketCoverage#57037, DentalOnlyPlan#57038, RejectRows#57039, IssuerId#57059, IssuerId2#57058]

+- Project [IssuerIDDD#8273, BusinessDate#57031, stcd#57060, srcnm#57061, NetworkName#57034, NetworkURL#57035, custnum#57036, MarketCoverage#57037, DentalOnlyPlan#57038, RejectRows#57039, IssuerId#57059, IssuerId2#57058]

+- Project [IssuerIDD#8261, IssuerIDDD#8273, BusinessDate#57031, stcd#57060, srcnm#57061, NetworkName#57034, NetworkURL#57035, custnum#57036, MarketCoverage#57037, DentalOnlyPlan#57038, RejectRows#57039, IssuerId#57059, cast(IssuerIDDD#8273 as string) AS IssuerId2#57058]

+- Project [IssuerIDD#8261, IssuerIDDD#8273, BusinessDate#57031, stcd#57060, srcnm#57061, NetworkName#57034, NetworkURL#57035, custnum#57036, MarketCoverage#57037, DentalOnlyPlan#57038, RejectRows#57039, cast(IssuerIDD#8261 as string) AS IssuerId#57059]

+- Project [IssuerIDD#8261, IssuerId2#57030 AS IssuerIDDD#8273, BusinessDate#57031, stcd#57060, srcnm#57061, NetworkName#57034, NetworkURL#57035, custnum#57036, MarketCoverage#57037, DentalOnlyPlan#57038, RejectRows#57039]

+- Project [IssuerId#57029 AS IssuerIDD#8261, IssuerId2#57030, BusinessDate#57031, stcd#57060, srcnm#57061, NetworkName#57034, NetworkURL#57035, custnum#57036, MarketCoverage#57037, DentalOnlyPlan#57038, RejectRows#57039]

+- Project [IssuerId#57029, IssuerId2#57030, BusinessDate#57031, StateCode#57032 AS stcd#57060, srcnm#57061, NetworkName#57034, NetworkURL#57035, custnum#57036, MarketCoverage#57037, DentalOnlyPlan#57038, RejectRows#57039]

+- Project [IssuerId#57029, IssuerId2#57030, BusinessDate#57031, StateCode#57032, SourceName#57033 AS srcnm#57061, NetworkName#57034, NetworkURL#57035, custnum#57036, MarketCoverage#57037, DentalOnlyPlan#57038, RejectRows#57039]

+- Union false, false

:- Relation [IssuerId#57029,IssuerId2#57030,BusinessDate#57031,StateCode#57032,SourceName#57033,NetworkName#57034,NetworkURL#57035,custnum#57036,MarketCoverage#57037,DentalOnlyPlan#57038,RejectRows#57039] csv

+- Relation [IssuerId#57040,IssuerId2#57041,BusinessDate#57042,StateCode#57043,SourceName#57044,NetworkName#57045,NetworkURL#57046,custnum#57047,MarketCoverage#57048,DentalOnlyPlan#57049,RejectRows#57050] csv

>>> second\_join.describe

>>> second\_join.describe

<bound method DataFrame.describe of DataFrame[BusinessDate: date, stcd: string, srcnm: string, NetworkName: string, NetworkURL: string, custnum: string, MarketCoverage: string, RejectRows: string, IssuerId: string, IssuerId2: string, issueridcomposite: string, sysdt: date, systs: timestamp, cleannetworkname: string, curdt: date, curts: timestamp, cleannetworkname: string, \_c0: string, \_c1: string, \_c0: string, \_c1: string, \_c2: string, \_c3: int, \_c4: string, Protocol\_Used: string]>

>>> second\_join.show(1)

+------------+----+-----+-----------+--------------------+-------+--------------+----------+--------+---------+-----------------+----------+--------------------+----------------+----------+--------------------+----------------+---+-------+---+-----+-------+---+-----+-------------+

|BusinessDate|stcd|srcnm|NetworkName| NetworkURL|custnum|MarketCoverage|RejectRows|IssuerId|IssuerId2|issueridcomposite| sysdt| systs|cleannetworkname| curdt| curts|cleannetworkname|\_c0| \_c1|\_c0| \_c1| \_c2|\_c3| \_c4|Protocol\_Used|

+------------+----+-----+-----------+--------------------+-------+--------------+----------+--------+---------+-----------------+----------+--------------------+----------------+----------+--------------------+----------------+---+-------+---+-----+-------+---+-----+-------------+

| 2019-10-01| AZ| HIOS| La Paz|http://uamarketpl...| 21| null| null| 88925| 88925| 8892588925|2024-07-22|2024-07-22 16:59:...| La Paz|2024-07-22|2024-07-22 16:59:...| La Paz| AZ|Arizona|779|Jason|Bowling| 21|Coach| http secured|

+------------+----+-----+-----------+--------------------+-------+--------------+----------+--------+---------+-----------------+----------+--------------------+----------------+----------+--------------------+----------------+---+-------+---+-----+-------+---+-----+-------------+

only showing top 1 row

>>> from pyspark.sql import functions as F

>>> from pyspark.sql.window import Window

>>>

>>> # Step 1: Calculate average age and count

... grouped\_df = second\_join.groupBy('srcnm', 'Protocol\_Used', '\_c4') \

... .agg(

... F.avg('\_c3').alias('avg\_age'),

... F.count('\*').alias('profession\_count')

... )

>>> window\_spec = Window.partitionBy('Protocol\_Used', 'srcnm') \

... .orderBy(F.desc('profession\_count'))

>>> ranked\_df = grouped\_df.withColumn('seqno', F.row\_number().over(window\_spec))

>>> result\_df = ranked\_df.filter(ranked\_df.seqno == 2)

>>> result\_df.show(2)

+-----+-------------+-----+------------------+----------------+-----+

|srcnm|Protocol\_Used| \_c4| avg\_age|profession\_count|seqno|

+-----+-------------+-----+------------------+----------------+-----+

| HIOS| No Protocol|Nurse|22.076923076923077| 28158| 2|

|SERFF| No Protocol|Nurse| 21.0| 12996| 2|

+-----+-------------+-----+------------------+----------------+-----+

only showing top 2 rows

>>>

>>> mysql\_url = "jdbc://mysql://localhost/Hackathon"

>>> mysql\_table = "hackathon\_tbl"

>>> mysql\_user = "root"

>>> mysql\_password = "Root123$"

>>> result\_df.write .format("jdbc") .option("url", mysql\_url) .option("dbtable", mysql\_table) .option("user", mysql\_user) .option("password", mysql\_password) .option("driver", "com.mysql.cj.jdbc.Driver") .mode("overwrite") .save()

Traceback (most recent call last):

File "<stdin>", line 1, in <module>

File "/usr/local/spark/python/pyspark/sql/readwriter.py", line 738, in save

self.\_jwrite.save()

File "/usr/local/spark/python/lib/py4j-0.10.9.3-src.zip/py4j/java\_gateway.py", line 1322, in \_\_call\_\_

File "/usr/local/spark/python/pyspark/sql/utils.py", line 117, in deco

raise converted from None

pyspark.sql.utils.IllegalArgumentException: requirement failed: The driver could not open a JDBC connection. Check the URL: jdbc://mysql://localhost/Hackathon

>>> mysql\_url = "jdbc:mysql://localhost/Hackathon"

>>> mysql\_table = "hackathon\_tbl"

>>> mysql\_user = "root"

>>> mysql\_password = "Root123$"

>>> result\_df.write .format("jdbc") .option("url", mysql\_url) .option("dbtable", mysql\_table) .option("user", mysql\_user) .option("password", mysql\_password) .option("driver", "com.mysql.cj.jdbc.Driver") .mode("overwrite") .save()

>>> >>> rdd.take(4)

File "<stdin>", line 1

>>> rdd.take(4)

^

SyntaxError: invalid syntax

>>> ['1,Kristina,Chung,55,Pilot', '2,Paige,Chen,74,Teacher', 'NM,New Mexico', 'NY,New York']

['1,Kristina,Chung,55,Pilot', '2,Paige,Chen,74,Teacher', 'NM,New Mexico', 'NY,New York']

>>> >>> custfilter =rdd.map(lambda x: x.split(",") )

File "<stdin>", line 1

>>> custfilter =rdd.map(lambda x: x.split(",") )

^

SyntaxError: invalid syntax

>>> >>> print(custfilter.take(14))

File "<stdin>", line 1

>>> print(custfilter.take(14))

^

SyntaxError: invalid syntax

>>> [['1', 'Kristina', 'Chung', '55', 'Pilot'], ['2', 'Paige', 'Chen', '74', 'Teacher'], ['NM', 'New Mexico'], ['NY', 'New York'], ['NC', 'North Carolina'], ['ND', 'North Dakota'], ['OH', 'Ohio'], ['OK', 'Oklahoma'], ['OR', 'Oregon'], ['MD', 'Maryland'], ['MA', 'Massachusetts'], ['MI', 'Michigan'], ['MN', 'Minnesota'], ['MS', 'Mississippi']]

[['1', 'Kristina', 'Chung', '55', 'Pilot'], ['2', 'Paige', 'Chen', '74', 'Teacher'], ['NM', 'New Mexico'], ['NY', 'New York'], ['NC', 'North Carolina'], ['ND', 'North Dakota'], ['OH', 'Ohio'], ['OK', 'Oklahoma'], ['OR', 'Oregon'], ['MD', 'Maryland'], ['MA', 'Massachusetts'], ['MI', 'Michigan'], ['MN', 'Minnesota'], ['MS', 'Mississippi']]

>>> >>> splited\_rdd =rdd.map(lambda x: x.split(",") )

File "<stdin>", line 1

>>> splited\_rdd =rdd.map(lambda x: x.split(",") )

^

SyntaxError: invalid syntax

>>> >>> custfilter =splited\_rdd.filter(lambda x: len(x)>3 )

File "<stdin>", line 1

>>> custfilter =splited\_rdd.filter(lambda x: len(x)>3 )

^

SyntaxError: invalid syntax

>>> >>> print(custfilter.take(4))

File "<stdin>", line 1

>>> print(custfilter.take(4))

^

SyntaxError: invalid syntax

>>> [['1', 'Kristina', 'Chung', '55', 'Pilot'], ['2', 'Paige', 'Chen', '74', 'Teacher'], ['3', 'Sherri', 'Melton', '34', 'Firefighter'], ['4', 'Gretchen', 'Hill', '66', 'Computer hardware engineer']]

[['1', 'Kristina', 'Chung', '55', 'Pilot'], ['2', 'Paige', 'Chen', '74', 'Teacher'], ['3', 'Sherri', 'Melton', '34', 'Firefighter'], ['4', 'Gretchen', 'Hill', '66', 'Computer hardware engineer']]

>>> >>> print(custfilter.take(5))

File "<stdin>", line 1

>>> print(custfilter.take(5))

^

SyntaxError: invalid syntax

>>> [['1', 'Kristina', 'Chung', '55', 'Pilot'], ['2', 'Paige', 'Chen', '74', 'Teacher'], ['3', 'Sherri', 'Melton', '34', 'Firefighter'], ['4', 'Gretchen', 'Hill', '66', 'Computer hardware engineer'], ['5', 'Karen', 'Puckett', '74', 'Lawyer']]

[['1', 'Kristina', 'Chung', '55', 'Pilot'], ['2', 'Paige', 'Chen', '74', 'Teacher'], ['3', 'Sherri', 'Melton', '34', 'Firefighter'], ['4', 'Gretchen', 'Hill', '66', 'Computer hardware engineer'], ['5', 'Karen', 'Puckett', '74', 'Lawyer']]

>>> >>> statefilter =splited\_rdd.filter(lambda x: len(x)<3 )

File "<stdin>", line 1

>>> statefilter =splited\_rdd.filter(lambda x: len(x)<3 )

^

SyntaxError: invalid syntax

>>> >>> print(statefilter.take(4))

File "<stdin>", line 1

>>> print(statefilter.take(4))

^

SyntaxError: invalid syntax

>>> [['NM', 'New Mexico'], ['NY', 'New York'], ['NC', 'North Carolina'], ['ND', 'North Dakota']]

[['NM', 'New Mexico'], ['NY', 'New York'], ['NC', 'North Carolina'], ['ND', 'North Dakota']]

>>> >>>

File "<stdin>", line 1

>>>

^

SyntaxError: invalid syntax

>>> >>> complete\_df=spark.read.csv("/home/hduser/Desktop/Hackathon",sep=",",header=False,inferSchema=True)

File "<stdin>", line 1

>>> complete\_df=spark.read.csv("/home/hduser/Desktop/Hackathon",sep=",",header=False,inferSchema=True)

^

SyntaxError: invalid syntax

>>> Traceback (most recent call last):

File "<stdin>", line 1

Traceback (most recent call last):

^

SyntaxError: invalid syntax

>>> File "<stdin>", line 1, in <module>

File "<stdin>", line 1

File "<stdin>", line 1, in <module>

^

IndentationError: unexpected indent

>>> File "/usr/local/spark/python/pyspark/sql/readwriter.py", line 410, in csv

File "<stdin>", line 1

File "/usr/local/spark/python/pyspark/sql/readwriter.py", line 410, in csv

^

IndentationError: unexpected indent

>>> return self.\_df(self.\_jreader.csv(self.\_spark.\_sc.\_jvm.PythonUtils.toSeq(path)))

File "<stdin>", line 1

return self.\_df(self.\_jreader.csv(self.\_spark.\_sc.\_jvm.PythonUtils.toSeq(path)))

^

IndentationError: unexpected indent

>>> File "/usr/local/spark/python/lib/py4j-0.10.9.3-src.zip/py4j/java\_gateway.py", line 1322, in \_\_call\_\_

File "<stdin>", line 1

File "/usr/local/spark/python/lib/py4j-0.10.9.3-src.zip/py4j/java\_gateway.py", line 1322, in \_\_call\_\_

^

IndentationError: unexpected indent

>>>

Final final log :

4/07/22 12:20:34 WARN util.package: Truncated the string representation of a plan since it was too large. This behavior can be adjusted by setting 'spark.sql.debug.maxToStringFields'.

+------------+----+-----+----------------+--------------------+-------+------------------+----------+--------+---------+-----------------+----------+--------------------+-----+----+------------+----+-----+-----------+--------------------+-------+--------------+----------+--------+---------+-----------------+----------+--------------------+-------------+------------+----+-----+----------------+--------------------+-------+------------------+----------+--------+---------+-----------------+----------+--------------------+----------+--------------------+------------+----+-----+----------------+--------------------+-------+------------------+----------+--------+---------+-----------------+----------+--------------------+----------------+

|BusinessDate|stcd|srcnm| NetworkName| NetworkURL|custnum| MarketCoverage|RejectRows|IssuerId|IssuerId2|issueridcomposite| sysdt| systs|month|year|BusinessDate|stcd|srcnm|NetworkName| NetworkURL|custnum|MarketCoverage|RejectRows|IssuerId|IssuerId2|issueridcomposite| sysdt| systs|Protocol\_Used|BusinessDate|stcd|srcnm| NetworkName| NetworkURL|custnum| MarketCoverage|RejectRows|IssuerId|IssuerId2|issueridcomposite| sysdt| systs| curdt| curts|BusinessDate|stcd|srcnm| NetworkName| NetworkURL|custnum| MarketCoverage|RejectRows|IssuerId|IssuerId2|issueridcomposite| sysdt| systs|cleannetworkname|

+------------+----+-----+----------------+--------------------+-------+------------------+----------+--------+---------+-----------------+----------+--------------------+-----+----+------------+----+-----+-----------+--------------------+-------+--------------+----------+--------+---------+-----------------+----------+--------------------+-------------+------------+----+-----+----------------+--------------------+-------+------------------+----------+--------+---------+-----------------+----------+--------------------+----------+--------------------+------------+----+-----+----------------+--------------------+-------+------------------+----------+--------+---------+-----------------+----------+--------------------+----------------+

| 2019-01-10| AK| HIOS|Delta Dental PPO|https://www.modah...| 17|SHOP (Small Group)| null| 21989| 21989| 2198921989|2024-07-22|2024-07-22 12:20:...| 1|2019| 2019-10-01| AK| HIOS|ODS Premier|https://www.modah...| 13| null| null| 21989| 21989| 2198921989|2024-07-22|2024-07-22 12:20:...| http secured| 2019-01-10| AK| HIOS|Delta Dental PPO|https://www.modah...| 17|SHOP (Small Group)| null| 21989| 21989| 2198921989|2024-07-22|2024-07-22 12:20:...|2024-07-22|2024-07-22 12:20:...| 2019-01-10| AK| HIOS|Delta Dental PPO|https://www.modah...| 17|SHOP (Small Group)| null| 21989| 21989| 2198921989|2024-07-22|2024-07-22 12:20:...|Delta Dental PPO|

+------------+----+-----+----------------+--------------------+-------+------------------+----------+--------+---------+-----------------+----------+--------------------+-----+----+------------+----+-----+-----------+--------------------+-------+--------------+----------+--------+---------+-----------------+----------+--------------------+-------------+------------+----+-----+----------------+--------------------+-------+------------------+----------+--------+---------+-----------------+----------+--------------------+----------+--------------------+------------+----+-----+----------------+--------------------+-------+------------------+----------+--------+---------+-----------------+----------+--------------------+----------------+

only showing top 1 row

>>> df\_with\_states = df\_combined \

... .alias("i") \

... .join(statesview.alias("s"), col("i.stcd") == col("s.stated"), how="inner")

Traceback (most recent call last):

File "<stdin>", line 3, in <module>

NameError: name 'statesview' is not defined

>>>

>>> # Joining result with custview

... df\_final = df\_with\_states \

... .alias("is") \

... .join(custview.alias("c"), col("is.custnum") == col("c.custid"), how="inner")

Traceback (most recent call last):

File "<stdin>", line 2, in <module>

NameError: name 'df\_with\_states' is not defined

>>> joined\_df\_state = df\_combined.alias("i").join(

... state\_filter\_df.alias("s"),

... col("i.stcd") == col("s.stated"),

... how="inner"

... )

Traceback (most recent call last):

File "<stdin>", line 2, in <module>

NameError: name 'state\_filter\_df' is not defined

>>>

>>> # Join the result with custfilter\_df

... final\_df = joined\_df\_state.alias("j").join(

... custfilter\_df.alias("c"),

... col("j.custnum") == col("c.custid"),

... how="inner"

... )

Traceback (most recent call last):

File "<stdin>", line 2, in <module>

NameError: name 'joined\_df\_state' is not defined

>>> )joined\_df\_state = df\_combined.alias("i").join(

File "<stdin>", line 1

)joined\_df\_state = df\_combined.alias("i").join(

^

SyntaxError: invalid syntax

>>> statefilter\_df.alias("s"),

File "<stdin>", line 1

statefilter\_df.alias("s"),

^

IndentationError: unexpected indent

>>> col("i.stcd") == col("s.stated"),

File "<stdin>", line 1

col("i.stcd") == col("s.stated"),

^

IndentationError: unexpected indent

>>> how="inner"

File "<stdin>", line 1

how="inner"

^

IndentationError: unexpected indent

>>> )

File "<stdin>", line 1

)

^

SyntaxError: invalid syntax

>>> joined\_df\_state = df\_combined.alias("i").join(

... statefilter\_df.alias("s"),

... col("i.stcd") == col("s.stated"),

... how="inner"

... )

Traceback (most recent call last):

File "<stdin>", line 4, in <module>

File "/usr/local/spark/python/pyspark/sql/dataframe.py", line 1355, in join

jdf = self.\_jdf.join(other.\_jdf, on, how)

File "/usr/local/spark/python/lib/py4j-0.10.9.3-src.zip/py4j/java\_gateway.py", line 1322, in \_\_call\_\_

File "/usr/local/spark/python/pyspark/sql/utils.py", line 117, in deco

raise converted from None

pyspark.sql.utils.AnalysisException: Reference 'i.stcd' is ambiguous, could be: i.stcd, i.stcd, i.stcd, i.stcd.

>>> joined\_df\_state = df\_combined.alias("i").join(

... state\_filter\_df.alias("s"),

... col("i.stcd") == col("s.stated"),

... how="inner"

... )

Traceback (most recent call last):

File "<stdin>", line 2, in <module>

NameError: name 'state\_filter\_df' is not defined

>>> joined\_df\_state = df\_combined.alias("i").join(

... statefilter\_df.alias("s"),

... col("i.stcd") == col("s.stated"),

... how="inner"

... )

Traceback (most recent call last):

File "<stdin>", line 4, in <module>

File "/usr/local/spark/python/pyspark/sql/dataframe.py", line 1355, in join

jdf = self.\_jdf.join(other.\_jdf, on, how)

File "/usr/local/spark/python/lib/py4j-0.10.9.3-src.zip/py4j/java\_gateway.py", line 1322, in \_\_call\_\_

File "/usr/local/spark/python/pyspark/sql/utils.py", line 117, in deco

raise converted from None

pyspark.sql.utils.AnalysisException: Reference 'i.stcd' is ambiguous, could be: i.stcd, i.stcd, i.stcd, i.stcd.

>>>

>>> df\_combined\_renamed = df\_combined.withColumnRenamed("stcd", "i\_stcd")

>>> state\_filter\_df\_renamed = statefilter\_df.withColumnRenamed("stated", "s\_stated")

>>>

>>> # Step 2: Perform the join with renamed columns

... joined\_df\_state = df\_combined\_renamed.alias("i").join(

... statefilter\_df\_renamed.alias("s"),

... col("i.i\_stcd") == col("s.s\_stated"),

... how="inner"

... )

Traceback (most recent call last):

File "<stdin>", line 3, in <module>

NameError: name 'statefilter\_df\_renamed' is not defined

>>>

>>> # Step 3: Rename columns in the result to remove ambiguity

... df\_combined\_state\_renamed = joined\_df\_state.withColumnRenamed("statedesc", "state\_desc")

Traceback (most recent call last):

File "<stdin>", line 2, in <module>

NameError: name 'joined\_df\_state' is not defined

>>>

>>> # Step 4: Join with custfilter\_df

... final\_df = df\_combined\_state\_renamed.alias("j").join(

... custfilter\_df.alias("c"),

... col("j.custnum") == col("c.custid"),

... how="inner"

... )

Traceback (most recent call last):

File "<stdin>", line 2, in <module>

NameError: name 'df\_combined\_state\_renamed' is not defined

>>> state\_filter\_df\_renamed.SHOW()

Traceback (most recent call last):

File "<stdin>", line 1, in <module>

File "/usr/local/spark/python/pyspark/sql/dataframe.py", line 1660, in \_\_getattr\_\_

"'%s' object has no attribute '%s'" % (self.\_\_class\_\_.\_\_name\_\_, name))

AttributeError: 'DataFrame' object has no attribute 'SHOW'

>>> state\_filter\_df\_renamed.show()

+---+--------------+

|\_c0| \_c1|

+---+--------------+

| NM| New Mexico|

| NY| New York|

| NC|North Carolina|

| ND| North Dakota|

| OH| Ohio|

| OK| Oklahoma|

| OR| Oregon|

| MD| Maryland|

| MA| Massachusetts|

| MI| Michigan|

| MN| Minnesota|

| MS| Mississippi|

| MO| Missouri|

| PA| Pennsylvania|

| RI| Rhode Island|

| SC|South Carolina|

| SD| South Dakota|

| TN| Tennessee|

| TX| Texas|

| UT| Utah|

+---+--------------+

only showing top 20 rows

>>> joined\_df\_state = df\_combined\_renamed.alias("i").join(

... statefilter\_df\_renamed.alias("s"),

... col("i.i\_stcd") == col("s.s\_stated"),

... how="inner"

... )

Traceback (most recent call last):

File "<stdin>", line 2, in <module>

NameError: name 'statefilter\_df\_renamed' is not defined

>>>

>>> col("i.i\_stcd") == col("s.s\_stated"),

>>> col("i.i\_stcd") == col("s.s\_stated"),

>>> col("i.i\_stcd") == col("s.s\_stated"),

File "<stdin>", line 1

col("i.i\_stcd") == col("s.s\_stated"),

^

IndentationError: unexpected indent

>>>

>>> df\_combined\_renamed = df\_combined.withColumnRenamed("stcd", "i\_stcd")

>>> state\_filter\_df\_renamed = statefilter\_df.withColumnRenamed("stated", "s\_stated")

>>>

>>> # Step 2: Perform the join with renamed columns

... joined\_df\_state = df\_combined\_renamed.alias("i").join(

... state\_filter\_df\_renamed.alias("s"),

... col("i.i\_stcd") == col("s.s\_stated"),

... how="inner"

... )

Traceback (most recent call last):

File "<stdin>", line 5, in <module>

File "/usr/local/spark/python/pyspark/sql/dataframe.py", line 1355, in join

jdf = self.\_jdf.join(other.\_jdf, on, how)

File "/usr/local/spark/python/lib/py4j-0.10.9.3-src.zip/py4j/java\_gateway.py", line 1322, in \_\_call\_\_

File "/usr/local/spark/python/pyspark/sql/utils.py", line 117, in deco

raise converted from None

pyspark.sql.utils.AnalysisException: Reference 'i.i\_stcd' is ambiguous, could be: i.i\_stcd, i.i\_stcd, i.i\_stcd, i.i\_stcd.

>>>

>>> # Step 3: Rename columns in the result to remove ambiguity

... df\_combined\_state\_renamed = joined\_df\_state.withColumnRenamed("statedesc", "state\_desc")

Traceback (most recent call last):

File "<stdin>", line 2, in <module>

NameError: name 'joined\_df\_state' is not defined

>>>

>>> # Step 4: Join with custfilter\_df

... final\_df = df\_combined\_state\_renamed.alias("j").join(

... custfilter\_df.alias("c"),

... col("j.custnum") == col("c.custid"),

... how="inner"

... )

Traceback (most recent call last):

File "<stdin>", line 2, in <module>

NameError: name 'df\_combined\_state\_renamed' is not defined

>>> df\_combined.show(1)

+------------+----+-----+----------------+--------------------+-------+------------------+----------+--------+---------+-----------------+----------+--------------------+-----+----+------------+----+-----+-----------+--------------------+-------+--------------+----------+--------+---------+-----------------+----------+--------------------+-------------+------------+----+-----+----------------+--------------------+-------+------------------+----------+--------+---------+-----------------+----------+--------------------+----------+--------------------+------------+----+-----+----------------+--------------------+-------+------------------+----------+--------+---------+-----------------+----------+--------------------+----------------+

|BusinessDate|stcd|srcnm| NetworkName| NetworkURL|custnum| MarketCoverage|RejectRows|IssuerId|IssuerId2|issueridcomposite| sysdt| systs|month|year|BusinessDate|stcd|srcnm|NetworkName| NetworkURL|custnum|MarketCoverage|RejectRows|IssuerId|IssuerId2|issueridcomposite| sysdt| systs|Protocol\_Used|BusinessDate|stcd|srcnm| NetworkName| NetworkURL|custnum| MarketCoverage|RejectRows|IssuerId|IssuerId2|issueridcomposite| sysdt| systs| curdt| curts|BusinessDate|stcd|srcnm| NetworkName| NetworkURL|custnum| MarketCoverage|RejectRows|IssuerId|IssuerId2|issueridcomposite| sysdt| systs|cleannetworkname|

+------------+----+-----+----------------+--------------------+-------+------------------+----------+--------+---------+-----------------+----------+--------------------+-----+----+------------+----+-----+-----------+--------------------+-------+--------------+----------+--------+---------+-----------------+----------+--------------------+-------------+------------+----+-----+----------------+--------------------+-------+------------------+----------+--------+---------+-----------------+----------+--------------------+----------+--------------------+------------+----+-----+----------------+--------------------+-------+------------------+----------+--------+---------+-----------------+----------+--------------------+----------------+

| 2019-01-10| AK| HIOS|Delta Dental PPO|https://www.modah...| 17|SHOP (Small Group)| null| 21989| 21989| 2198921989|2024-07-22|2024-07-22 12:31:...| 1|2019| 2019-10-01| AK| HIOS|ODS Premier|https://www.modah...| 13| null| null| 21989| 21989| 2198921989|2024-07-22|2024-07-22 12:31:...| http secured| 2019-01-10| AK| HIOS|Delta Dental PPO|https://www.modah...| 17|SHOP (Small Group)| null| 21989| 21989| 2198921989|2024-07-22|2024-07-22 12:31:...|2024-07-22|2024-07-22 12:31:...| 2019-01-10| AK| HIOS|Delta Dental PPO|https://www.modah...| 17|SHOP (Small Group)| null| 21989| 21989| 2198921989|2024-07-22|2024-07-22 12:31:...|Delta Dental PPO|

+------------+----+-----+----------------+--------------------+-------+------------------+----------+--------+---------+-----------------+----------+--------------------+-----+----+------------+----+-----+-----------+--------------------+-------+--------------+----------+--------+---------+-----------------+----------+--------------------+-------------+------------+----+-----+----------------+--------------------+-------+------------------+----------+--------+---------+-----------------+----------+--------------------+----------+--------------------+------------+----+-----+----------------+--------------------+-------+------------------+----------+--------+---------+-----------------+----------+--------------------+----------------+

only showing top 1 row

>>>

>>> df\_combined.write.parquet("hdfs:///user/hduser/Hackathon/Parquet")

Traceback (most recent call last):

File "<stdin>", line 1, in <module>

File "/usr/local/spark/python/pyspark/sql/readwriter.py", line 885, in parquet

self.\_jwrite.parquet(path)

File "/usr/local/spark/python/lib/py4j-0.10.9.3-src.zip/py4j/java\_gateway.py", line 1322, in \_\_call\_\_

File "/usr/local/spark/python/pyspark/sql/utils.py", line 117, in deco

raise converted from None

pyspark.sql.utils.AnalysisException: Found duplicate column(s) when inserting into hdfs://localhost:54310/user/hduser/Hackathon/Parquet: `businessdate`, `custnum`, `issuerid2`, `issuerid`, `issueridcomposite`, `marketcoverage`, `networkname`, `networkurl`, `rejectrows`, `srcnm`, `stcd`, `sysdt`, `systs`

>>> df\_combined.describe

<bound method DataFrame.describe of DataFrame[BusinessDate: date, stcd: string, srcnm: string, NetworkName: string, NetworkURL: string, custnum: string, MarketCoverage: string, RejectRows: string, IssuerId: string, IssuerId2: string, issueridcomposite: string, sysdt: date, systs: timestamp, month: int, year: int, BusinessDate: date, stcd: string, srcnm: string, NetworkName: string, NetworkURL: string, custnum: string, MarketCoverage: string, RejectRows: string, IssuerId: string, IssuerId2: string, issueridcomposite: string, sysdt: date, systs: timestamp, Protocol\_Used: string, BusinessDate: date, stcd: string, srcnm: string, NetworkName: string, NetworkURL: string, custnum: string, MarketCoverage: string, RejectRows: string, IssuerId: string, IssuerId2: string, issueridcomposite: string, sysdt: date, systs: timestamp, curdt: date, curts: timestamp, BusinessDate: date, stcd: string, srcnm: string, NetworkName: string, NetworkURL: string, custnum: string, MarketCoverage: string, RejectRows: string, IssuerId: string, IssuerId2: string, issueridcomposite: string, sysdt: date, systs: timestamp, cleannetworkname: string]>

>>> df\_combined.describe

>>> list\_of\_col=df\_combined.columns

>>> print(list\_of\_col)

['BusinessDate', 'stcd', 'srcnm', 'NetworkName', 'NetworkURL', 'custnum', 'MarketCoverage', 'RejectRows', 'IssuerId', 'IssuerId2', 'issueridcomposite', 'sysdt', 'systs', 'month', 'year', 'BusinessDate', 'stcd', 'srcnm', 'NetworkName', 'NetworkURL', 'custnum', 'MarketCoverage', 'RejectRows', 'IssuerId', 'IssuerId2', 'issueridcomposite', 'sysdt', 'systs', 'Protocol\_Used', 'BusinessDate', 'stcd', 'srcnm', 'NetworkName', 'NetworkURL', 'custnum', 'MarketCoverage', 'RejectRows', 'IssuerId', 'IssuerId2', 'issueridcomposite', 'sysdt', 'systs', 'curdt', 'curts', 'BusinessDate', 'stcd', 'srcnm', 'NetworkName', 'NetworkURL', 'custnum', 'MarketCoverage', 'RejectRows', 'IssuerId', 'IssuerId2', 'issueridcomposite', 'sysdt', 'systs', 'cleannetworkname']

>>> tuple\_list=tuple(list\_of\_col)

>>> print(tuple\_list)

('BusinessDate', 'stcd', 'srcnm', 'NetworkName', 'NetworkURL', 'custnum', 'MarketCoverage', 'RejectRows', 'IssuerId', 'IssuerId2', 'issueridcomposite', 'sysdt', 'systs', 'month', 'year', 'BusinessDate', 'stcd', 'srcnm', 'NetworkName', 'NetworkURL', 'custnum', 'MarketCoverage', 'RejectRows', 'IssuerId', 'IssuerId2', 'issueridcomposite', 'sysdt', 'systs', 'Protocol\_Used', 'BusinessDate', 'stcd', 'srcnm', 'NetworkName', 'NetworkURL', 'custnum', 'MarketCoverage', 'RejectRows', 'IssuerId', 'IssuerId2', 'issueridcomposite', 'sysdt', 'systs', 'curdt', 'curts', 'BusinessDate', 'stcd', 'srcnm', 'NetworkName', 'NetworkURL', 'custnum', 'MarketCoverage', 'RejectRows', 'IssuerId', 'IssuerId2', 'issueridcomposite', 'sysdt', 'systs', 'cleannetworkname')

>>> set\_list=set(list\_of\_col)

>>> print(set\_list)

{'NetworkURL', 'year', 'issueridcomposite', 'NetworkName', 'RejectRows', 'IssuerId', 'stcd', 'IssuerId2', 'systs', 'sysdt', 'BusinessDate', 'MarketCoverage', 'Protocol\_Used', 'curts', 'curdt', 'month', 'cleannetworkname', 'srcnm', 'custnum'}

>>> list\_col\_no\_duplicate=list(set\_list)

>>> df\_combined.select(list\_col\_no\_duplicate).show(1)

Traceback (most recent call last):

File "<stdin>", line 1, in <module>

File "/usr/local/spark/python/pyspark/sql/dataframe.py", line 1685, in select

jdf = self.\_jdf.select(self.\_jcols(\*cols))

File "/usr/local/spark/python/lib/py4j-0.10.9.3-src.zip/py4j/java\_gateway.py", line 1322, in \_\_call\_\_

File "/usr/local/spark/python/pyspark/sql/utils.py", line 117, in deco

raise converted from None

pyspark.sql.utils.AnalysisException: Reference 'NetworkURL' is ambiguous, could be: a.NetworkURL, b.NetworkURL, c.NetworkURL, d.NetworkURL.

>>> df\_combined.select(list\_col\_no\_duplicate).show(1)

Traceback (most recent call last):

File "<stdin>", line 1, in <module>

File "/usr/local/spark/python/pyspark/sql/dataframe.py", line 1685, in select

jdf = self.\_jdf.select(self.\_jcols(\*cols))

File "/usr/local/spark/python/lib/py4j-0.10.9.3-src.zip/py4j/java\_gateway.py", line 1322, in \_\_call\_\_

File "/usr/local/spark/python/pyspark/sql/utils.py", line 117, in deco

raise converted from None

pyspark.sql.utils.AnalysisException: Reference 'NetworkURL' is ambiguous, could be: a.NetworkURL, b.NetworkURL, c.NetworkURL, d.NetworkURL.

>>>

>>> # Rename columns in each DataFrame before joining

... result\_df\_SQL = result\_df\_SQL.withColumnRenamed("NetworkURL", "a\_NetworkURL") \

... .withColumnRenamed("NetworkName", "a\_NetworkName") \

... .withColumnRenamed("IssuerId", "a\_IssuerId") \

... .withColumnRenamed("BusinessDate", "a\_BusinessDate")

>>>

>>> result\_df\_SQL\_1 = result\_df\_SQL\_1.withColumnRenamed("NetworkURL", "b\_NetworkURL") \

... .withColumnRenamed("NetworkName", "b\_NetworkName") \

... .withColumnRenamed("IssuerId", "b\_IssuerId") \

... .withColumnRenamed("BusinessDate", "b\_BusinessDate")

>>>

>>> result\_df\_SQL\_2 = result\_df\_SQL\_2.withColumnRenamed("NetworkURL", "c\_NetworkURL") \

... .withColumnRenamed("NetworkName", "c\_NetworkName") \

... .withColumnRenamed("IssuerId", "c\_IssuerId") \

... .withColumnRenamed("BusinessDate", "c\_BusinessDate")

>>>

>>> result\_df\_SQL\_3 = result\_df\_SQL\_3.withColumnRenamed("NetworkURL", "d\_NetworkURL") \

... .withColumnRenamed("NetworkName", "d\_NetworkName") \

... .withColumnRenamed("IssuerId", "d\_IssuerId") \

... .withColumnRenamed("BusinessDate", "d\_BusinessDate")

>>>

>>> # Perform the join

... df\_combined = result\_df\_SQL \

... .alias("a") \

... .join(result\_df\_SQL\_1.alias("b"), col("a.IssuerId") == col("b.IssuerId"), how="inner") \

... .join(result\_df\_SQL\_2.alias("c"), col("a.IssuerId") == col("c.IssuerId"), how="inner") \

... .join(result\_df\_SQL\_3.alias("d"), col("a.IssuerId") == col("d.IssuerId"), how="inner")

Traceback (most recent call last):

File "<stdin>", line 4, in <module>

File "/usr/local/spark/python/pyspark/sql/dataframe.py", line 1355, in join

jdf = self.\_jdf.join(other.\_jdf, on, how)

File "/usr/local/spark/python/lib/py4j-0.10.9.3-src.zip/py4j/java\_gateway.py", line 1322, in \_\_call\_\_

File "/usr/local/spark/python/pyspark/sql/utils.py", line 117, in deco

raise converted from None

pyspark.sql.utils.AnalysisException: cannot resolve 'a.IssuerId' given input columns: [a.IssuerId2, b.IssuerId2, a.MarketCoverage, b.MarketCoverage, b.Protocol\_Used, a.RejectRows, b.RejectRows, a.a\_BusinessDate, a.a\_IssuerId, a.a\_NetworkName, a.a\_NetworkURL, b.b\_BusinessDate, b.b\_IssuerId, b.b\_NetworkName, b.b\_NetworkURL, a.custnum, b.custnum, a.issueridcomposite, b.issueridcomposite, a.month, a.srcnm, b.srcnm, a.stcd, b.stcd, a.sysdt, b.sysdt, a.systs, b.systs, a.year];

'Join Inner, ('a.IssuerId = 'b.IssuerId)

:- SubqueryAlias a

: +- Project [BusinessDate#1378 AS a\_BusinessDate#43854, stcd#7759, srcnm#7747, a\_NetworkName#43822, a\_NetworkURL#43806, custnum#1383, MarketCoverage#1384, RejectRows#1386, a\_IssuerId#43838, IssuerId2#8325, issueridcomposite#8901, sysdt#9003, systs#9016, month#30928, year#30929]

: +- Project [BusinessDate#1378, stcd#7759, srcnm#7747, a\_NetworkName#43822, a\_NetworkURL#43806, custnum#1383, MarketCoverage#1384, RejectRows#1386, IssuerId#8312 AS a\_IssuerId#43838, IssuerId2#8325, issueridcomposite#8901, sysdt#9003, systs#9016, month#30928, year#30929]

: +- Project [BusinessDate#1378, stcd#7759, srcnm#7747, NetworkName#1381 AS a\_NetworkName#43822, a\_NetworkURL#43806, custnum#1383, MarketCoverage#1384, RejectRows#1386, IssuerId#8312, IssuerId2#8325, issueridcomposite#8901, sysdt#9003, systs#9016, month#30928, year#30929]

: +- Project [BusinessDate#1378, stcd#7759, srcnm#7747, NetworkName#1381, NetworkURL#1382 AS a\_NetworkURL#43806, custnum#1383, MarketCoverage#1384, RejectRows#1386, IssuerId#8312, IssuerId2#8325, issueridcomposite#8901, sysdt#9003, systs#9016, month#30928, year#30929]

: +- Project [BusinessDate#1378, stcd#7759, srcnm#7747, NetworkName#1381, NetworkURL#1382, custnum#1383, MarketCoverage#1384, RejectRows#1386, IssuerId#8312, IssuerId2#8325, issueridcomposite#8901, sysdt#9003, systs#9016, month(BusinessDate#1378) AS month#30928, year(BusinessDate#1378) AS year#30929]

: +- SubqueryAlias insureview

: +- View (`insureview`, [BusinessDate#1378,stcd#7759,srcnm#7747,NetworkName#1381,NetworkURL#1382,custnum#1383,MarketCoverage#1384,RejectRows#1386,IssuerId#8312,IssuerId2#8325,issueridcomposite#8901,sysdt#9003,systs#9016])

: +- Project [BusinessDate#1378, stcd#7759, srcnm#7747, NetworkName#1381, NetworkURL#1382, custnum#1383, MarketCoverage#1384, RejectRows#1386, IssuerId#8312, IssuerId2#8325, issueridcomposite#8901, sysdt#9003, current\_timestamp() AS systs#9016]

: +- Project [BusinessDate#1378, stcd#7759, srcnm#7747, NetworkName#1381, NetworkURL#1382, custnum#1383, MarketCoverage#1384, RejectRows#1386, IssuerId#8312, IssuerId2#8325, issueridcomposite#8901, current\_date(Some(Asia/Kolkata)) AS sysdt#9003]

: +- Project [BusinessDate#1378, stcd#7759, srcnm#7747, NetworkName#1381, NetworkURL#1382, custnum#1383, MarketCoverage#1384, RejectRows#1386, IssuerId#8312, IssuerId2#8325, issueridcomposite#8901]

: +- Project [BusinessDate#1378, stcd#7759, srcnm#7747, NetworkName#1381, NetworkURL#1382, custnum#1383, MarketCoverage#1384, DentalOnlyPlan#1385, RejectRows#1386, IssuerId#8312, IssuerId2#8325, concat(IssuerId#8312, IssuerId2#8325) AS issueridcomposite#8901]

: +- Project [BusinessDate#1378, stcd#7759, srcnm#7747, NetworkName#1381, NetworkURL#1382, custnum#1383, MarketCoverage#1384, DentalOnlyPlan#1385, RejectRows#1386, IssuerId#8312, IssuerId2#8325]

: +- Project [IssuerIDDD#8273, BusinessDate#1378, stcd#7759, srcnm#7747, NetworkName#1381, NetworkURL#1382, custnum#1383, MarketCoverage#1384, DentalOnlyPlan#1385, RejectRows#1386, IssuerId#8312, IssuerId2#8325]

: +- Project [IssuerIDD#8261, IssuerIDDD#8273, BusinessDate#1378, stcd#7759, srcnm#7747, NetworkName#1381, NetworkURL#1382, custnum#1383, MarketCoverage#1384, DentalOnlyPlan#1385, RejectRows#1386, IssuerId#8312, cast(IssuerIDDD#8273 as string) AS IssuerId2#8325]

: +- Project [IssuerIDD#8261, IssuerIDDD#8273, BusinessDate#1378, stcd#7759, srcnm#7747, NetworkName#1381, NetworkURL#1382, custnum#1383, MarketCoverage#1384, DentalOnlyPlan#1385, RejectRows#1386, cast(IssuerIDD#8261 as string) AS IssuerId#8312]

: +- Project [IssuerIDD#8261, IssuerId2#1377 AS IssuerIDDD#8273, BusinessDate#1378, stcd#7759, srcnm#7747, NetworkName#1381, NetworkURL#1382, custnum#1383, MarketCoverage#1384, DentalOnlyPlan#1385, RejectRows#1386]

: +- Project [IssuerId#1376 AS IssuerIDD#8261, IssuerId2#1377, BusinessDate#1378, stcd#7759, srcnm#7747, NetworkName#1381, NetworkURL#1382, custnum#1383, MarketCoverage#1384, DentalOnlyPlan#1385, RejectRows#1386]

: +- Project [IssuerId#1376, IssuerId2#1377, BusinessDate#1378, StateCode#1379 AS stcd#7759, srcnm#7747, NetworkName#1381, NetworkURL#1382, custnum#1383, MarketCoverage#1384, DentalOnlyPlan#1385, RejectRows#1386]

: +- Project [IssuerId#1376, IssuerId2#1377, BusinessDate#1378, StateCode#1379, SourceName#1380 AS srcnm#7747, NetworkName#1381, NetworkURL#1382, custnum#1383, MarketCoverage#1384, DentalOnlyPlan#1385, RejectRows#1386]

: +- Union false, false

: :- Relation [IssuerId#1376,IssuerId2#1377,BusinessDate#1378,StateCode#1379,SourceName#1380,NetworkName#1381,NetworkURL#1382,custnum#1383,MarketCoverage#1384,DentalOnlyPlan#1385,RejectRows#1386] csv

: +- Relation [IssuerId#491,IssuerId2#492,BusinessDate#493,StateCode#494,SourceName#495,NetworkName#496,NetworkURL#497,custnum#498,MarketCoverage#499,DentalOnlyPlan#500,RejectRows#501] csv

+- SubqueryAlias b

+- Project [BusinessDate#44056 AS b\_BusinessDate#43915, stcd#44080, srcnm#44081, b\_NetworkName#43885, b\_NetworkURL#43870, custnum#44061, MarketCoverage#44062, RejectRows#44064, b\_IssuerId#43900, IssuerId2#44079, issueridcomposite#44078, sysdt#44077, systs#44076, Protocol\_Used#30945]

+- Project [BusinessDate#44056, stcd#44080, srcnm#44081, b\_NetworkName#43885, b\_NetworkURL#43870, custnum#44061, MarketCoverage#44062, RejectRows#44064, IssuerId#8312 AS b\_IssuerId#43900, IssuerId2#44079, issueridcomposite#44078, sysdt#44077, systs#44076, Protocol\_Used#30945]

+- Project [BusinessDate#44056, stcd#44080, srcnm#44081, NetworkName#44059 AS b\_NetworkName#43885, b\_NetworkURL#43870, custnum#44061, MarketCoverage#44062, RejectRows#44064, IssuerId#8312, IssuerId2#44079, issueridcomposite#44078, sysdt#44077, systs#44076, Protocol\_Used#30945]

+- Project [BusinessDate#44056, stcd#44080, srcnm#44081, NetworkName#44059, NetworkURL#44060 AS b\_NetworkURL#43870, custnum#44061, MarketCoverage#44062, RejectRows#44064, IssuerId#8312, IssuerId2#44079, issueridcomposite#44078, sysdt#44077, systs#44076, Protocol\_Used#30945]

+- Project [BusinessDate#44056, stcd#44080, srcnm#44081, NetworkName#44059, NetworkURL#44060, custnum#44061, MarketCoverage#44062, RejectRows#44064, IssuerId#8312, IssuerId2#44079, issueridcomposite#44078, sysdt#44077, systs#44076, CASE WHEN NetworkURL#44060 LIKE https://% THEN http secured WHEN NetworkURL#44060 LIKE http://% THEN http non secured ELSE No Protocol END AS Protocol\_Used#30945]

+- SubqueryAlias insureview

+- View (`insureview`, [BusinessDate#44056,stcd#44080,srcnm#44081,NetworkName#44059,NetworkURL#44060,custnum#44061,MarketCoverage#44062,RejectRows#44064,IssuerId#8312,IssuerId2#44079,issueridcomposite#44078,sysdt#44077,systs#44076])

+- Project [BusinessDate#44056, stcd#44080, srcnm#44081, NetworkName#44059, NetworkURL#44060, custnum#44061, MarketCoverage#44062, RejectRows#44064, IssuerId#8312, IssuerId2#44079, issueridcomposite#44078, sysdt#44077, current\_timestamp() AS systs#44076]

+- Project [BusinessDate#44056, stcd#44080, srcnm#44081, NetworkName#44059, NetworkURL#44060, custnum#44061, MarketCoverage#44062, RejectRows#44064, IssuerId#8312, IssuerId2#44079, issueridcomposite#44078, current\_date(Some(Asia/Kolkata)) AS sysdt#44077]

+- Project [BusinessDate#44056, stcd#44080, srcnm#44081, NetworkName#44059, NetworkURL#44060, custnum#44061, MarketCoverage#44062, RejectRows#44064, IssuerId#8312, IssuerId2#44079, issueridcomposite#44078]

+- Project [BusinessDate#44056, stcd#44080, srcnm#44081, NetworkName#44059, NetworkURL#44060, custnum#44061, MarketCoverage#44062, DentalOnlyPlan#44063, RejectRows#44064, IssuerId#8312, IssuerId2#44079, concat(IssuerId#8312, IssuerId2#44079) AS issueridcomposite#44078]

+- Project [BusinessDate#44056, stcd#44080, srcnm#44081, NetworkName#44059, NetworkURL#44060, custnum#44061, MarketCoverage#44062, DentalOnlyPlan#44063, RejectRows#44064, IssuerId#8312, IssuerId2#44079]

+- Project [IssuerIDDD#8273, BusinessDate#44056, stcd#44080, srcnm#44081, NetworkName#44059, NetworkURL#44060, custnum#44061, MarketCoverage#44062, DentalOnlyPlan#44063, RejectRows#44064, IssuerId#8312, IssuerId2#44079]

+- Project [IssuerIDD#8261, IssuerIDDD#8273, BusinessDate#44056, stcd#44080, srcnm#44081, NetworkName#44059, NetworkURL#44060, custnum#44061, MarketCoverage#44062, DentalOnlyPlan#44063, RejectRows#44064, IssuerId#8312, cast(IssuerIDDD#8273 as string) AS IssuerId2#44079]

+- Project [IssuerIDD#8261, IssuerIDDD#8273, BusinessDate#44056, stcd#44080, srcnm#44081, NetworkName#44059, NetworkURL#44060, custnum#44061, MarketCoverage#44062, DentalOnlyPlan#44063, RejectRows#44064, cast(IssuerIDD#8261 as string) AS IssuerId#8312]

+- Project [IssuerIDD#8261, IssuerId2#44055 AS IssuerIDDD#8273, BusinessDate#44056, stcd#44080, srcnm#44081, NetworkName#44059, NetworkURL#44060, custnum#44061, MarketCoverage#44062, DentalOnlyPlan#44063, RejectRows#44064]

+- Project [IssuerId#44054 AS IssuerIDD#8261, IssuerId2#44055, BusinessDate#44056, stcd#44080, srcnm#44081, NetworkName#44059, NetworkURL#44060, custnum#44061, MarketCoverage#44062, DentalOnlyPlan#44063, RejectRows#44064]

+- Project [IssuerId#44054, IssuerId2#44055, BusinessDate#44056, StateCode#44057 AS stcd#44080, srcnm#44081, NetworkName#44059, NetworkURL#44060, custnum#44061, MarketCoverage#44062, DentalOnlyPlan#44063, RejectRows#44064]

+- Project [IssuerId#44054, IssuerId2#44055, BusinessDate#44056, StateCode#44057, SourceName#44058 AS srcnm#44081, NetworkName#44059, NetworkURL#44060, custnum#44061, MarketCoverage#44062, DentalOnlyPlan#44063, RejectRows#44064]

+- Union false, false

:- Relation [IssuerId#44054,IssuerId2#44055,BusinessDate#44056,StateCode#44057,SourceName#44058,NetworkName#44059,NetworkURL#44060,custnum#44061,MarketCoverage#44062,DentalOnlyPlan#44063,RejectRows#44064] csv

+- Relation [IssuerId#44065,IssuerId2#44066,BusinessDate#44067,StateCode#44068,SourceName#44069,NetworkName#44070,NetworkURL#44071,custnum#44072,MarketCoverage#44073,DentalOnlyPlan#44074,RejectRows#44075] csv

>>>

>>> # Show the columns to ensure uniqueness

... df\_combined.columns

['BusinessDate', 'stcd', 'srcnm', 'NetworkName', 'NetworkURL', 'custnum', 'MarketCoverage', 'RejectRows', 'IssuerId', 'IssuerId2', 'issueridcomposite', 'sysdt', 'systs', 'month', 'year', 'BusinessDate', 'stcd', 'srcnm', 'NetworkName', 'NetworkURL', 'custnum', 'MarketCoverage', 'RejectRows', 'IssuerId', 'IssuerId2', 'issueridcomposite', 'sysdt', 'systs', 'Protocol\_Used', 'BusinessDate', 'stcd', 'srcnm', 'NetworkName', 'NetworkURL', 'custnum', 'MarketCoverage', 'RejectRows', 'IssuerId', 'IssuerId2', 'issueridcomposite', 'sysdt', 'systs', 'curdt', 'curts', 'BusinessDate', 'stcd', 'srcnm', 'NetworkName', 'NetworkURL', 'custnum', 'MarketCoverage', 'RejectRows', 'IssuerId', 'IssuerId2', 'issueridcomposite', 'sysdt', 'systs', 'cleannetworkname']

>>>

>>> from pyspark.sql.functions import col

>>>

>>> # Rename columns in each DataFrame to avoid ambiguity

... df1 = result\_df\_SQL.alias("a").select(

... col("BusinessDate").alias("a\_BusinessDate"),

... col("stcd").alias("a\_stcd"),

... col("srcnm").alias("a\_srcnm"),

... col("NetworkName").alias("a\_NetworkName"),

... col("NetworkURL").alias("a\_NetworkURL"),

... col("custnum").alias("a\_custnum"),

... col("MarketCoverage").alias("a\_MarketCoverage"),

... col("RejectRows").alias("a\_RejectRows"),

... col("IssuerId").alias("a\_IssuerId"),

... col("IssuerId2").alias("a\_IssuerId2"),

... col("issueridcomposite").alias("a\_issueridcomposite"),

... col("sysdt").alias("a\_sysdt"),

... col("systs").alias("a\_systs"),

... col("month").alias("a\_month"),

... col("year").alias("a\_year")

... )

Traceback (most recent call last):

File "<stdin>", line 17, in <module>

File "/usr/local/spark/python/pyspark/sql/dataframe.py", line 1685, in select

jdf = self.\_jdf.select(self.\_jcols(\*cols))

File "/usr/local/spark/python/lib/py4j-0.10.9.3-src.zip/py4j/java\_gateway.py", line 1322, in \_\_call\_\_

File "/usr/local/spark/python/pyspark/sql/utils.py", line 117, in deco

raise converted from None

pyspark.sql.utils.AnalysisException: cannot resolve 'BusinessDate' given input columns: [a.IssuerId2, a.MarketCoverage, a.RejectRows, a.a\_BusinessDate, a.a\_IssuerId, a.a\_NetworkName, a.a\_NetworkURL, a.custnum, a.issueridcomposite, a.month, a.srcnm, a.stcd, a.sysdt, a.systs, a.year];

'Project ['BusinessDate AS a\_BusinessDate#44082, stcd#7759 AS a\_stcd#44083, srcnm#7747 AS a\_srcnm#44084, 'NetworkName AS a\_NetworkName#44085, 'NetworkURL AS a\_NetworkURL#44086, custnum#1383 AS a\_custnum#44087, MarketCoverage#1384 AS a\_MarketCoverage#44088, RejectRows#1386 AS a\_RejectRows#44089, 'IssuerId AS a\_IssuerId#44090, IssuerId2#8325 AS a\_IssuerId2#44091, issueridcomposite#8901 AS a\_issueridcomposite#44092, sysdt#9003 AS a\_sysdt#44093, systs#9016 AS a\_systs#44094, month#30928 AS a\_month#44095, year#30929 AS a\_year#44096]

+- SubqueryAlias a

+- Project [BusinessDate#1378 AS a\_BusinessDate#43854, stcd#7759, srcnm#7747, a\_NetworkName#43822, a\_NetworkURL#43806, custnum#1383, MarketCoverage#1384, RejectRows#1386, a\_IssuerId#43838, IssuerId2#8325, issueridcomposite#8901, sysdt#9003, systs#9016, month#30928, year#30929]

+- Project [BusinessDate#1378, stcd#7759, srcnm#7747, a\_NetworkName#43822, a\_NetworkURL#43806, custnum#1383, MarketCoverage#1384, RejectRows#1386, IssuerId#8312 AS a\_IssuerId#43838, IssuerId2#8325, issueridcomposite#8901, sysdt#9003, systs#9016, month#30928, year#30929]

+- Project [BusinessDate#1378, stcd#7759, srcnm#7747, NetworkName#1381 AS a\_NetworkName#43822, a\_NetworkURL#43806, custnum#1383, MarketCoverage#1384, RejectRows#1386, IssuerId#8312, IssuerId2#8325, issueridcomposite#8901, sysdt#9003, systs#9016, month#30928, year#30929]

+- Project [BusinessDate#1378, stcd#7759, srcnm#7747, NetworkName#1381, NetworkURL#1382 AS a\_NetworkURL#43806, custnum#1383, MarketCoverage#1384, RejectRows#1386, IssuerId#8312, IssuerId2#8325, issueridcomposite#8901, sysdt#9003, systs#9016, month#30928, year#30929]

+- Project [BusinessDate#1378, stcd#7759, srcnm#7747, NetworkName#1381, NetworkURL#1382, custnum#1383, MarketCoverage#1384, RejectRows#1386, IssuerId#8312, IssuerId2#8325, issueridcomposite#8901, sysdt#9003, systs#9016, month(BusinessDate#1378) AS month#30928, year(BusinessDate#1378) AS year#30929]

+- SubqueryAlias insureview

+- View (`insureview`, [BusinessDate#1378,stcd#7759,srcnm#7747,NetworkName#1381,NetworkURL#1382,custnum#1383,MarketCoverage#1384,RejectRows#1386,IssuerId#8312,IssuerId2#8325,issueridcomposite#8901,sysdt#9003,systs#9016])

+- Project [BusinessDate#1378, stcd#7759, srcnm#7747, NetworkName#1381, NetworkURL#1382, custnum#1383, MarketCoverage#1384, RejectRows#1386, IssuerId#8312, IssuerId2#8325, issueridcomposite#8901, sysdt#9003, current\_timestamp() AS systs#9016]

+- Project [BusinessDate#1378, stcd#7759, srcnm#7747, NetworkName#1381, NetworkURL#1382, custnum#1383, MarketCoverage#1384, RejectRows#1386, IssuerId#8312, IssuerId2#8325, issueridcomposite#8901, current\_date(Some(Asia/Kolkata)) AS sysdt#9003]

+- Project [BusinessDate#1378, stcd#7759, srcnm#7747, NetworkName#1381, NetworkURL#1382, custnum#1383, MarketCoverage#1384, RejectRows#1386, IssuerId#8312, IssuerId2#8325, issueridcomposite#8901]

+- Project [BusinessDate#1378, stcd#7759, srcnm#7747, NetworkName#1381, NetworkURL#1382, custnum#1383, MarketCoverage#1384, DentalOnlyPlan#1385, RejectRows#1386, IssuerId#8312, IssuerId2#8325, concat(IssuerId#8312, IssuerId2#8325) AS issueridcomposite#8901]

+- Project [BusinessDate#1378, stcd#7759, srcnm#7747, NetworkName#1381, NetworkURL#1382, custnum#1383, MarketCoverage#1384, DentalOnlyPlan#1385, RejectRows#1386, IssuerId#8312, IssuerId2#8325]

+- Project [IssuerIDDD#8273, BusinessDate#1378, stcd#7759, srcnm#7747, NetworkName#1381, NetworkURL#1382, custnum#1383, MarketCoverage#1384, DentalOnlyPlan#1385, RejectRows#1386, IssuerId#8312, IssuerId2#8325]

+- Project [IssuerIDD#8261, IssuerIDDD#8273, BusinessDate#1378, stcd#7759, srcnm#7747, NetworkName#1381, NetworkURL#1382, custnum#1383, MarketCoverage#1384, DentalOnlyPlan#1385, RejectRows#1386, IssuerId#8312, cast(IssuerIDDD#8273 as string) AS IssuerId2#8325]

+- Project [IssuerIDD#8261, IssuerIDDD#8273, BusinessDate#1378, stcd#7759, srcnm#7747, NetworkName#1381, NetworkURL#1382, custnum#1383, MarketCoverage#1384, DentalOnlyPlan#1385, RejectRows#1386, cast(IssuerIDD#8261 as string) AS IssuerId#8312]

+- Project [IssuerIDD#8261, IssuerId2#1377 AS IssuerIDDD#8273, BusinessDate#1378, stcd#7759, srcnm#7747, NetworkName#1381, NetworkURL#1382, custnum#1383, MarketCoverage#1384, DentalOnlyPlan#1385, RejectRows#1386]

+- Project [IssuerId#1376 AS IssuerIDD#8261, IssuerId2#1377, BusinessDate#1378, stcd#7759, srcnm#7747, NetworkName#1381, NetworkURL#1382, custnum#1383, MarketCoverage#1384, DentalOnlyPlan#1385, RejectRows#1386]

+- Project [IssuerId#1376, IssuerId2#1377, BusinessDate#1378, StateCode#1379 AS stcd#7759, srcnm#7747, NetworkName#1381, NetworkURL#1382, custnum#1383, MarketCoverage#1384, DentalOnlyPlan#1385, RejectRows#1386]

+- Project [IssuerId#1376, IssuerId2#1377, BusinessDate#1378, StateCode#1379, SourceName#1380 AS srcnm#7747, NetworkName#1381, NetworkURL#1382, custnum#1383, MarketCoverage#1384, DentalOnlyPlan#1385, RejectRows#1386]

+- Union false, false

:- Relation [IssuerId#1376,IssuerId2#1377,BusinessDate#1378,StateCode#1379,SourceName#1380,NetworkName#1381,NetworkURL#1382,custnum#1383,MarketCoverage#1384,DentalOnlyPlan#1385,RejectRows#1386] csv

+- Relation [IssuerId#491,IssuerId2#492,BusinessDate#493,StateCode#494,SourceName#495,NetworkName#496,NetworkURL#497,custnum#498,MarketCoverage#499,DentalOnlyPlan#500,RejectRows#501] csv

>>>

>>> df2 = result\_df\_SQL\_1.alias("b").select(

... col("BusinessDate").alias("b\_BusinessDate"),

... col("stcd").alias("b\_stcd"),

... col("srcnm").alias("b\_srcnm"),

... col("NetworkName").alias("b\_NetworkName"),

... col("NetworkURL").alias("b\_NetworkURL"),

... col("custnum").alias("b\_custnum"),

... col("MarketCoverage").alias("b\_MarketCoverage"),

... col("RejectRows").alias("b\_RejectRows"),

... col("IssuerId").alias("b\_IssuerId"),

... col("IssuerId2").alias("b\_IssuerId2"),

... col("issueridcomposite").alias("b\_issueridcomposite"),

... col("sysdt").alias("b\_sysdt"),

... col("systs").alias("b\_systs"),

... col("Protocol\_Used").alias("b\_Protocol\_Used")

... )

Traceback (most recent call last):

File "<stdin>", line 15, in <module>

File "/usr/local/spark/python/pyspark/sql/dataframe.py", line 1685, in select

jdf = self.\_jdf.select(self.\_jcols(\*cols))

File "/usr/local/spark/python/lib/py4j-0.10.9.3-src.zip/py4j/java\_gateway.py", line 1322, in \_\_call\_\_

File "/usr/local/spark/python/pyspark/sql/utils.py", line 117, in deco

raise converted from None

pyspark.sql.utils.AnalysisException: cannot resolve 'BusinessDate' given input columns: [b.IssuerId2, b.MarketCoverage, b.Protocol\_Used, b.RejectRows, b.b\_BusinessDate, b.b\_IssuerId, b.b\_NetworkName, b.b\_NetworkURL, b.custnum, b.issueridcomposite, b.srcnm, b.stcd, b.sysdt, b.systs];

'Project ['BusinessDate AS b\_BusinessDate#44097, stcd#7759 AS b\_stcd#44098, srcnm#7747 AS b\_srcnm#44099, 'NetworkName AS b\_NetworkName#44100, 'NetworkURL AS b\_NetworkURL#44101, custnum#1383 AS b\_custnum#44102, MarketCoverage#1384 AS b\_MarketCoverage#44103, RejectRows#1386 AS b\_RejectRows#44104, 'IssuerId AS b\_IssuerId#44105, IssuerId2#8325 AS b\_IssuerId2#44106, issueridcomposite#8901 AS b\_issueridcomposite#44107, sysdt#9003 AS b\_sysdt#44108, systs#9016 AS b\_systs#44109, Protocol\_Used#30945 AS b\_Protocol\_Used#44110]

+- SubqueryAlias b

+- Project [BusinessDate#1378 AS b\_BusinessDate#43915, stcd#7759, srcnm#7747, b\_NetworkName#43885, b\_NetworkURL#43870, custnum#1383, MarketCoverage#1384, RejectRows#1386, b\_IssuerId#43900, IssuerId2#8325, issueridcomposite#8901, sysdt#9003, systs#9016, Protocol\_Used#30945]

+- Project [BusinessDate#1378, stcd#7759, srcnm#7747, b\_NetworkName#43885, b\_NetworkURL#43870, custnum#1383, MarketCoverage#1384, RejectRows#1386, IssuerId#8312 AS b\_IssuerId#43900, IssuerId2#8325, issueridcomposite#8901, sysdt#9003, systs#9016, Protocol\_Used#30945]

+- Project [BusinessDate#1378, stcd#7759, srcnm#7747, NetworkName#1381 AS b\_NetworkName#43885, b\_NetworkURL#43870, custnum#1383, MarketCoverage#1384, RejectRows#1386, IssuerId#8312, IssuerId2#8325, issueridcomposite#8901, sysdt#9003, systs#9016, Protocol\_Used#30945]

+- Project [BusinessDate#1378, stcd#7759, srcnm#7747, NetworkName#1381, NetworkURL#1382 AS b\_NetworkURL#43870, custnum#1383, MarketCoverage#1384, RejectRows#1386, IssuerId#8312, IssuerId2#8325, issueridcomposite#8901, sysdt#9003, systs#9016, Protocol\_Used#30945]

+- Project [BusinessDate#1378, stcd#7759, srcnm#7747, NetworkName#1381, NetworkURL#1382, custnum#1383, MarketCoverage#1384, RejectRows#1386, IssuerId#8312, IssuerId2#8325, issueridcomposite#8901, sysdt#9003, systs#9016, CASE WHEN NetworkURL#1382 LIKE https://% THEN http secured WHEN NetworkURL#1382 LIKE http://% THEN http non secured ELSE No Protocol END AS Protocol\_Used#30945]

+- SubqueryAlias insureview

+- View (`insureview`, [BusinessDate#1378,stcd#7759,srcnm#7747,NetworkName#1381,NetworkURL#1382,custnum#1383,MarketCoverage#1384,RejectRows#1386,IssuerId#8312,IssuerId2#8325,issueridcomposite#8901,sysdt#9003,systs#9016])

+- Project [BusinessDate#1378, stcd#7759, srcnm#7747, NetworkName#1381, NetworkURL#1382, custnum#1383, MarketCoverage#1384, RejectRows#1386, IssuerId#8312, IssuerId2#8325, issueridcomposite#8901, sysdt#9003, current\_timestamp() AS systs#9016]

+- Project [BusinessDate#1378, stcd#7759, srcnm#7747, NetworkName#1381, NetworkURL#1382, custnum#1383, MarketCoverage#1384, RejectRows#1386, IssuerId#8312, IssuerId2#8325, issueridcomposite#8901, current\_date(Some(Asia/Kolkata)) AS sysdt#9003]

+- Project [BusinessDate#1378, stcd#7759, srcnm#7747, NetworkName#1381, NetworkURL#1382, custnum#1383, MarketCoverage#1384, RejectRows#1386, IssuerId#8312, IssuerId2#8325, issueridcomposite#8901]

+- Project [BusinessDate#1378, stcd#7759, srcnm#7747, NetworkName#1381, NetworkURL#1382, custnum#1383, MarketCoverage#1384, DentalOnlyPlan#1385, RejectRows#1386, IssuerId#8312, IssuerId2#8325, concat(IssuerId#8312, IssuerId2#8325) AS issueridcomposite#8901]

+- Project [BusinessDate#1378, stcd#7759, srcnm#7747, NetworkName#1381, NetworkURL#1382, custnum#1383, MarketCoverage#1384, DentalOnlyPlan#1385, RejectRows#1386, IssuerId#8312, IssuerId2#8325]

+- Project [IssuerIDDD#8273, BusinessDate#1378, stcd#7759, srcnm#7747, NetworkName#1381, NetworkURL#1382, custnum#1383, MarketCoverage#1384, DentalOnlyPlan#1385, RejectRows#1386, IssuerId#8312, IssuerId2#8325]

+- Project [IssuerIDD#8261, IssuerIDDD#8273, BusinessDate#1378, stcd#7759, srcnm#7747, NetworkName#1381, NetworkURL#1382, custnum#1383, MarketCoverage#1384, DentalOnlyPlan#1385, RejectRows#1386, IssuerId#8312, cast(IssuerIDDD#8273 as string) AS IssuerId2#8325]

+- Project [IssuerIDD#8261, IssuerIDDD#8273, BusinessDate#1378, stcd#7759, srcnm#7747, NetworkName#1381, NetworkURL#1382, custnum#1383, MarketCoverage#1384, DentalOnlyPlan#1385, RejectRows#1386, cast(IssuerIDD#8261 as string) AS IssuerId#8312]

+- Project [IssuerIDD#8261, IssuerId2#1377 AS IssuerIDDD#8273, BusinessDate#1378, stcd#7759, srcnm#7747, NetworkName#1381, NetworkURL#1382, custnum#1383, MarketCoverage#1384, DentalOnlyPlan#1385, RejectRows#1386]

+- Project [IssuerId#1376 AS IssuerIDD#8261, IssuerId2#1377, BusinessDate#1378, stcd#7759, srcnm#7747, NetworkName#1381, NetworkURL#1382, custnum#1383, MarketCoverage#1384, DentalOnlyPlan#1385, RejectRows#1386]

+- Project [IssuerId#1376, IssuerId2#1377, BusinessDate#1378, StateCode#1379 AS stcd#7759, srcnm#7747, NetworkName#1381, NetworkURL#1382, custnum#1383, MarketCoverage#1384, DentalOnlyPlan#1385, RejectRows#1386]

+- Project [IssuerId#1376, IssuerId2#1377, BusinessDate#1378, StateCode#1379, SourceName#1380 AS srcnm#7747, NetworkName#1381, NetworkURL#1382, custnum#1383, MarketCoverage#1384, DentalOnlyPlan#1385, RejectRows#1386]

+- Union false, false

:- Relation [IssuerId#1376,IssuerId2#1377,BusinessDate#1378,StateCode#1379,SourceName#1380,NetworkName#1381,NetworkURL#1382,custnum#1383,MarketCoverage#1384,DentalOnlyPlan#1385,RejectRows#1386] csv

+- Relation [IssuerId#491,IssuerId2#492,BusinessDate#493,StateCode#494,SourceName#495,NetworkName#496,NetworkURL#497,custnum#498,MarketCoverage#499,DentalOnlyPlan#500,RejectRows#501] csv

>>>

>>> df3 = result\_df\_SQL\_2.alias("c").select(

... col("BusinessDate").alias("c\_BusinessDate"),

... col("stcd").alias("c\_stcd"),

... col("srcnm").alias("c\_srcnm"),

... col("NetworkName").alias("c\_NetworkName"),

... col("NetworkURL").alias("c\_NetworkURL"),

... col("custnum").alias("c\_custnum"),

... col("MarketCoverage").alias("c\_MarketCoverage"),

... col("RejectRows").alias("c\_RejectRows"),

... col("IssuerId").alias("c\_IssuerId"),

... col("IssuerId2").alias("c\_IssuerId2"),

... col("issueridcomposite").alias("c\_issueridcomposite"),

... col("sysdt").alias("c\_sysdt"),

... col("systs").alias("c\_systs"),

... col("Protocol\_Used").alias("c\_Protocol\_Used")

... )

Traceback (most recent call last):

File "<stdin>", line 15, in <module>

File "/usr/local/spark/python/pyspark/sql/dataframe.py", line 1685, in select

jdf = self.\_jdf.select(self.\_jcols(\*cols))

File "/usr/local/spark/python/lib/py4j-0.10.9.3-src.zip/py4j/java\_gateway.py", line 1322, in \_\_call\_\_

File "/usr/local/spark/python/pyspark/sql/utils.py", line 117, in deco

raise converted from None

pyspark.sql.utils.AnalysisException: cannot resolve 'BusinessDate' given input columns: [c.IssuerId2, c.MarketCoverage, c.RejectRows, c.c\_BusinessDate, c.c\_IssuerId, c.c\_NetworkName, c.c\_NetworkURL, c.curdt, c.curts, c.custnum, c.issueridcomposite, c.srcnm, c.stcd, c.sysdt, c.systs];

'Project ['BusinessDate AS c\_BusinessDate#44111, stcd#7759 AS c\_stcd#44112, srcnm#7747 AS c\_srcnm#44113, 'NetworkName AS c\_NetworkName#44114, 'NetworkURL AS c\_NetworkURL#44115, custnum#1383 AS c\_custnum#44116, MarketCoverage#1384 AS c\_MarketCoverage#44117, RejectRows#1386 AS c\_RejectRows#44118, 'IssuerId AS c\_IssuerId#44119, IssuerId2#8325 AS c\_IssuerId2#44120, issueridcomposite#8901 AS c\_issueridcomposite#44121, sysdt#9003 AS c\_sysdt#44122, systs#9016 AS c\_systs#44123, 'Protocol\_Used AS c\_Protocol\_Used#44124]

+- SubqueryAlias c

+- Project [BusinessDate#1378 AS c\_BusinessDate#43978, stcd#7759, srcnm#7747, c\_NetworkName#43946, c\_NetworkURL#43930, custnum#1383, MarketCoverage#1384, RejectRows#1386, c\_IssuerId#43962, IssuerId2#8325, issueridcomposite#8901, sysdt#9003, systs#9016, curdt#31490, curts#31491]

+- Project [BusinessDate#1378, stcd#7759, srcnm#7747, c\_NetworkName#43946, c\_NetworkURL#43930, custnum#1383, MarketCoverage#1384, RejectRows#1386, IssuerId#8312 AS c\_IssuerId#43962, IssuerId2#8325, issueridcomposite#8901, sysdt#9003, systs#9016, curdt#31490, curts#31491]

+- Project [BusinessDate#1378, stcd#7759, srcnm#7747, NetworkName#1381 AS c\_NetworkName#43946, c\_NetworkURL#43930, custnum#1383, MarketCoverage#1384, RejectRows#1386, IssuerId#8312, IssuerId2#8325, issueridcomposite#8901, sysdt#9003, systs#9016, curdt#31490, curts#31491]

+- Project [BusinessDate#1378, stcd#7759, srcnm#7747, NetworkName#1381, NetworkURL#1382 AS c\_NetworkURL#43930, custnum#1383, MarketCoverage#1384, RejectRows#1386, IssuerId#8312, IssuerId2#8325, issueridcomposite#8901, sysdt#9003, systs#9016, curdt#31490, curts#31491]

+- Project [BusinessDate#1378, stcd#7759, srcnm#7747, NetworkName#1381, NetworkURL#1382, custnum#1383, MarketCoverage#1384, RejectRows#1386, IssuerId#8312, IssuerId2#8325, issueridcomposite#8901, sysdt#9003, systs#9016, current\_date(Some(Asia/Kolkata)) AS curdt#31490, current\_timestamp() AS curts#31491]

+- SubqueryAlias insureview

+- View (`insureview`, [BusinessDate#1378,stcd#7759,srcnm#7747,NetworkName#1381,NetworkURL#1382,custnum#1383,MarketCoverage#1384,RejectRows#1386,IssuerId#8312,IssuerId2#8325,issueridcomposite#8901,sysdt#9003,systs#9016])

+- Project [BusinessDate#1378, stcd#7759, srcnm#7747, NetworkName#1381, NetworkURL#1382, custnum#1383, MarketCoverage#1384, RejectRows#1386, IssuerId#8312, IssuerId2#8325, issueridcomposite#8901, sysdt#9003, current\_timestamp() AS systs#9016]

+- Project [BusinessDate#1378, stcd#7759, srcnm#7747, NetworkName#1381, NetworkURL#1382, custnum#1383, MarketCoverage#1384, RejectRows#1386, IssuerId#8312, IssuerId2#8325, issueridcomposite#8901, current\_date(Some(Asia/Kolkata)) AS sysdt#9003]

+- Project [BusinessDate#1378, stcd#7759, srcnm#7747, NetworkName#1381, NetworkURL#1382, custnum#1383, MarketCoverage#1384, RejectRows#1386, IssuerId#8312, IssuerId2#8325, issueridcomposite#8901]

+- Project [BusinessDate#1378, stcd#7759, srcnm#7747, NetworkName#1381, NetworkURL#1382, custnum#1383, MarketCoverage#1384, DentalOnlyPlan#1385, RejectRows#1386, IssuerId#8312, IssuerId2#8325, concat(IssuerId#8312, IssuerId2#8325) AS issueridcomposite#8901]

+- Project [BusinessDate#1378, stcd#7759, srcnm#7747, NetworkName#1381, NetworkURL#1382, custnum#1383, MarketCoverage#1384, DentalOnlyPlan#1385, RejectRows#1386, IssuerId#8312, IssuerId2#8325]

+- Project [IssuerIDDD#8273, BusinessDate#1378, stcd#7759, srcnm#7747, NetworkName#1381, NetworkURL#1382, custnum#1383, MarketCoverage#1384, DentalOnlyPlan#1385, RejectRows#1386, IssuerId#8312, IssuerId2#8325]

+- Project [IssuerIDD#8261, IssuerIDDD#8273, BusinessDate#1378, stcd#7759, srcnm#7747, NetworkName#1381, NetworkURL#1382, custnum#1383, MarketCoverage#1384, DentalOnlyPlan#1385, RejectRows#1386, IssuerId#8312, cast(IssuerIDDD#8273 as string) AS IssuerId2#8325]

+- Project [IssuerIDD#8261, IssuerIDDD#8273, BusinessDate#1378, stcd#7759, srcnm#7747, NetworkName#1381, NetworkURL#1382, custnum#1383, MarketCoverage#1384, DentalOnlyPlan#1385, RejectRows#1386, cast(IssuerIDD#8261 as string) AS IssuerId#8312]

+- Project [IssuerIDD#8261, IssuerId2#1377 AS IssuerIDDD#8273, BusinessDate#1378, stcd#7759, srcnm#7747, NetworkName#1381, NetworkURL#1382, custnum#1383, MarketCoverage#1384, DentalOnlyPlan#1385, RejectRows#1386]

+- Project [IssuerId#1376 AS IssuerIDD#8261, IssuerId2#1377, BusinessDate#1378, stcd#7759, srcnm#7747, NetworkName#1381, NetworkURL#1382, custnum#1383, MarketCoverage#1384, DentalOnlyPlan#1385, RejectRows#1386]

+- Project [IssuerId#1376, IssuerId2#1377, BusinessDate#1378, StateCode#1379 AS stcd#7759, srcnm#7747, NetworkName#1381, NetworkURL#1382, custnum#1383, MarketCoverage#1384, DentalOnlyPlan#1385, RejectRows#1386]

+- Project [IssuerId#1376, IssuerId2#1377, BusinessDate#1378, StateCode#1379, SourceName#1380 AS srcnm#7747, NetworkName#1381, NetworkURL#1382, custnum#1383, MarketCoverage#1384, DentalOnlyPlan#1385, RejectRows#1386]

+- Union false, false

:- Relation [IssuerId#1376,IssuerId2#1377,BusinessDate#1378,StateCode#1379,SourceName#1380,NetworkName#1381,NetworkURL#1382,custnum#1383,MarketCoverage#1384,DentalOnlyPlan#1385,RejectRows#1386] csv

+- Relation [IssuerId#491,IssuerId2#492,BusinessDate#493,StateCode#494,SourceName#495,NetworkName#496,NetworkURL#497,custnum#498,MarketCoverage#499,DentalOnlyPlan#500,RejectRows#501] csv

>>>

>>> df4 = result\_df\_SQL\_3.alias("d").select(

... col("BusinessDate").alias("d\_BusinessDate"),

... col("stcd").alias("d\_stcd"),

... col("srcnm").alias("d\_srcnm"),

... col("NetworkName").alias("d\_NetworkName"),

... col("NetworkURL").alias("d\_NetworkURL"),

... col("custnum").alias("d\_custnum"),

... col("MarketCoverage").alias("d\_MarketCoverage"),

... col("RejectRows").alias("d\_RejectRows"),

... col("IssuerId").alias("d\_IssuerId"),

... col("IssuerId2").alias("d\_IssuerId2"),

... col("issueridcomposite").alias("d\_issueridcomposite"),

... col("sysdt").alias("d\_sysdt"),

... col("systs").alias("d\_systs"),

... col("cleannetworkname").alias("d\_cleannetworkname")

... )

Traceback (most recent call last):

File "<stdin>", line 15, in <module>

File "/usr/local/spark/python/pyspark/sql/dataframe.py", line 1685, in select

jdf = self.\_jdf.select(self.\_jcols(\*cols))

File "/usr/local/spark/python/lib/py4j-0.10.9.3-src.zip/py4j/java\_gateway.py", line 1322, in \_\_call\_\_

File "/usr/local/spark/python/pyspark/sql/utils.py", line 117, in deco

raise converted from None

pyspark.sql.utils.AnalysisException: cannot resolve 'BusinessDate' given input columns: [d.IssuerId2, d.MarketCoverage, d.RejectRows, d.cleannetworkname, d.custnum, d.d\_BusinessDate, d.d\_IssuerId, d.d\_NetworkName, d.d\_NetworkURL, d.issueridcomposite, d.srcnm, d.stcd, d.sysdt, d.systs];

'Project ['BusinessDate AS d\_BusinessDate#44125, stcd#7759 AS d\_stcd#44126, srcnm#7747 AS d\_srcnm#44127, 'NetworkName AS d\_NetworkName#44128, 'NetworkURL AS d\_NetworkURL#44129, custnum#1383 AS d\_custnum#44130, MarketCoverage#1384 AS d\_MarketCoverage#44131, RejectRows#1386 AS d\_RejectRows#44132, 'IssuerId AS d\_IssuerId#44133, IssuerId2#8325 AS d\_IssuerId2#44134, issueridcomposite#8901 AS d\_issueridcomposite#44135, sysdt#9003 AS d\_sysdt#44136, systs#9016 AS d\_systs#44137, cleannetworkname#32033 AS d\_cleannetworkname#44138]

+- SubqueryAlias d

+- Project [BusinessDate#1378 AS d\_BusinessDate#44039, stcd#7759, srcnm#7747, d\_NetworkName#44009, d\_NetworkURL#43994, custnum#1383, MarketCoverage#1384, RejectRows#1386, d\_IssuerId#44024, IssuerId2#8325, issueridcomposite#8901, sysdt#9003, systs#9016, cleannetworkname#32033]

+- Project [BusinessDate#1378, stcd#7759, srcnm#7747, d\_NetworkName#44009, d\_NetworkURL#43994, custnum#1383, MarketCoverage#1384, RejectRows#1386, IssuerId#8312 AS d\_IssuerId#44024, IssuerId2#8325, issueridcomposite#8901, sysdt#9003, systs#9016, cleannetworkname#32033]

+- Project [BusinessDate#1378, stcd#7759, srcnm#7747, NetworkName#1381 AS d\_NetworkName#44009, d\_NetworkURL#43994, custnum#1383, MarketCoverage#1384, RejectRows#1386, IssuerId#8312, IssuerId2#8325, issueridcomposite#8901, sysdt#9003, systs#9016, cleannetworkname#32033]

+- Project [BusinessDate#1378, stcd#7759, srcnm#7747, NetworkName#1381, NetworkURL#1382 AS d\_NetworkURL#43994, custnum#1383, MarketCoverage#1384, RejectRows#1386, IssuerId#8312, IssuerId2#8325, issueridcomposite#8901, sysdt#9003, systs#9016, cleannetworkname#32033]

+- Project [BusinessDate#1378, stcd#7759, srcnm#7747, NetworkName#1381, NetworkURL#1382, custnum#1383, MarketCoverage#1384, RejectRows#1386, IssuerId#8312, IssuerId2#8325, issueridcomposite#8901, sysdt#9003, systs#9016, sql\_function(NetworkName#1381) AS cleannetworkname#32033]

+- SubqueryAlias insureview

+- View (`insureview`, [BusinessDate#1378,stcd#7759,srcnm#7747,NetworkName#1381,NetworkURL#1382,custnum#1383,MarketCoverage#1384,RejectRows#1386,IssuerId#8312,IssuerId2#8325,issueridcomposite#8901,sysdt#9003,systs#9016])

+- Project [BusinessDate#1378, stcd#7759, srcnm#7747, NetworkName#1381, NetworkURL#1382, custnum#1383, MarketCoverage#1384, RejectRows#1386, IssuerId#8312, IssuerId2#8325, issueridcomposite#8901, sysdt#9003, current\_timestamp() AS systs#9016]

+- Project [BusinessDate#1378, stcd#7759, srcnm#7747, NetworkName#1381, NetworkURL#1382, custnum#1383, MarketCoverage#1384, RejectRows#1386, IssuerId#8312, IssuerId2#8325, issueridcomposite#8901, current\_date(Some(Asia/Kolkata)) AS sysdt#9003]

+- Project [BusinessDate#1378, stcd#7759, srcnm#7747, NetworkName#1381, NetworkURL#1382, custnum#1383, MarketCoverage#1384, RejectRows#1386, IssuerId#8312, IssuerId2#8325, issueridcomposite#8901]

+- Project [BusinessDate#1378, stcd#7759, srcnm#7747, NetworkName#1381, NetworkURL#1382, custnum#1383, MarketCoverage#1384, DentalOnlyPlan#1385, RejectRows#1386, IssuerId#8312, IssuerId2#8325, concat(IssuerId#8312, IssuerId2#8325) AS issueridcomposite#8901]

+- Project [BusinessDate#1378, stcd#7759, srcnm#7747, NetworkName#1381, NetworkURL#1382, custnum#1383, MarketCoverage#1384, DentalOnlyPlan#1385, RejectRows#1386, IssuerId#8312, IssuerId2#8325]

+- Project [IssuerIDDD#8273, BusinessDate#1378, stcd#7759, srcnm#7747, NetworkName#1381, NetworkURL#1382, custnum#1383, MarketCoverage#1384, DentalOnlyPlan#1385, RejectRows#1386, IssuerId#8312, IssuerId2#8325]

+- Project [IssuerIDD#8261, IssuerIDDD#8273, BusinessDate#1378, stcd#7759, srcnm#7747, NetworkName#1381, NetworkURL#1382, custnum#1383, MarketCoverage#1384, DentalOnlyPlan#1385, RejectRows#1386, IssuerId#8312, cast(IssuerIDDD#8273 as string) AS IssuerId2#8325]

+- Project [IssuerIDD#8261, IssuerIDDD#8273, BusinessDate#1378, stcd#7759, srcnm#7747, NetworkName#1381, NetworkURL#1382, custnum#1383, MarketCoverage#1384, DentalOnlyPlan#1385, RejectRows#1386, cast(IssuerIDD#8261 as string) AS IssuerId#8312]

+- Project [IssuerIDD#8261, IssuerId2#1377 AS IssuerIDDD#8273, BusinessDate#1378, stcd#7759, srcnm#7747, NetworkName#1381, NetworkURL#1382, custnum#1383, MarketCoverage#1384, DentalOnlyPlan#1385, RejectRows#1386]

+- Project [IssuerId#1376 AS IssuerIDD#8261, IssuerId2#1377, BusinessDate#1378, stcd#7759, srcnm#7747, NetworkName#1381, NetworkURL#1382, custnum#1383, MarketCoverage#1384, DentalOnlyPlan#1385, RejectRows#1386]

+- Project [IssuerId#1376, IssuerId2#1377, BusinessDate#1378, StateCode#1379 AS stcd#7759, srcnm#7747, NetworkName#1381, NetworkURL#1382, custnum#1383, MarketCoverage#1384, DentalOnlyPlan#1385, RejectRows#1386]

+- Project [IssuerId#1376, IssuerId2#1377, BusinessDate#1378, StateCode#1379, SourceName#1380 AS srcnm#7747, NetworkName#1381, NetworkURL#1382, custnum#1383, MarketCoverage#1384, DentalOnlyPlan#1385, RejectRows#1386]

+- Union false, false

:- Relation [IssuerId#1376,IssuerId2#1377,BusinessDate#1378,StateCode#1379,SourceName#1380,NetworkName#1381,NetworkURL#1382,custnum#1383,MarketCoverage#1384,DentalOnlyPlan#1385,RejectRows#1386] csv

+- Relation [IssuerId#491,IssuerId2#492,BusinessDate#493,StateCode#494,SourceName#495,NetworkName#496,NetworkURL#497,custnum#498,MarketCoverage#499,DentalOnlyPlan#500,RejectRows#501] csv

>>>

>>> # Perform the join with renamed columns

... df\_combined = df1 \

... .join(df2, col("a\_IssuerId") == col("b\_IssuerId"), how="inner") \

... .join(df3, col("a\_IssuerId") == col("c\_IssuerId"), how="inner") \

... .join(df4, col("a\_IssuerId") == col("d\_IssuerId"), how="inner")

Traceback (most recent call last):

File "<stdin>", line 2, in <module>

NameError: name 'df1' is not defined

>>>

>>> # Show the columns to verify uniqueness

... df\_combined.columns

['BusinessDate', 'stcd', 'srcnm', 'NetworkName', 'NetworkURL', 'custnum', 'MarketCoverage', 'RejectRows', 'IssuerId', 'IssuerId2', 'issueridcomposite', 'sysdt', 'systs', 'month', 'year', 'BusinessDate', 'stcd', 'srcnm', 'NetworkName', 'NetworkURL', 'custnum', 'MarketCoverage', 'RejectRows', 'IssuerId', 'IssuerId2', 'issueridcomposite', 'sysdt', 'systs', 'Protocol\_Used', 'BusinessDate', 'stcd', 'srcnm', 'NetworkName', 'NetworkURL', 'custnum', 'MarketCoverage', 'RejectRows', 'IssuerId', 'IssuerId2', 'issueridcomposite', 'sysdt', 'systs', 'curdt', 'curts', 'BusinessDate', 'stcd', 'srcnm', 'NetworkName', 'NetworkURL', 'custnum', 'MarketCoverage', 'RejectRows', 'IssuerId', 'IssuerId2', 'issueridcomposite', 'sysdt', 'systs', 'cleannetworkname']

>>> result\_df\_SQL.describe

<bound method DataFrame.describe of DataFrame[a\_BusinessDate: date, stcd: string, srcnm: string, a\_NetworkName: string, a\_NetworkURL: string, custnum: string, MarketCoverage: string, RejectRows: string, a\_IssuerId: string, IssuerId2: string, issueridcomposite: string, sysdt: date, systs: timestamp, month: int, year: int]>

>>> result\_df\_SQL\_!.describe

File "<stdin>", line 1

result\_df\_SQL\_!.describe

^

SyntaxError: invalid syntax

>>> result\_df\_SQL\_!.describe

>>> result\_df\_SQL.describe

<bound method DataFrame.describe of DataFrame[a\_BusinessDate: date, stcd: string, srcnm: string, a\_NetworkName: string, a\_NetworkURL: string, custnum: string, MarketCoverage: string, RejectRows: string, a\_IssuerId: string, IssuerId2: string, issueridcomposite: string, sysdt: date, systs: timestamp, month: int, year: int]>

>>> result\_df\_SQL\_!.describe

File "<stdin>", line 1

result\_df\_SQL\_!.describe

^

SyntaxError: invalid syntax

>>> result\_df\_SQL.describe

<bound method DataFrame.describe of DataFrame[a\_BusinessDate: date, stcd: string, srcnm: string, a\_NetworkName: string, a\_NetworkURL: string, custnum: string, MarketCoverage: string, RejectRows: string, a\_IssuerId: string, IssuerId2: string, issueridcomposite: string, sysdt: date, systs: timestamp, month: int, year: int]>

>>> result\_df\_SQL\_1.describe

<bound method DataFrame.describe of DataFrame[b\_BusinessDate: date, stcd: string, srcnm: string, b\_NetworkName: string, b\_NetworkURL: string, custnum: string, MarketCoverage: string, RejectRows: string, b\_IssuerId: string, IssuerId2: string, issueridcomposite: string, sysdt: date, systs: timestamp, Protocol\_Used: string]>

>>> result\_df\_SQL\_2.describe

<bound method DataFrame.describe of DataFrame[c\_BusinessDate: date, stcd: string, srcnm: string, c\_NetworkName: string, c\_NetworkURL: string, custnum: string, MarketCoverage: string, RejectRows: string, c\_IssuerId: string, IssuerId2: string, issueridcomposite: string, sysdt: date, systs: timestamp, curdt: date, curts: timestamp]>

>>> result\_df\_SQL\_3.describe

<bound method DataFrame.describe of DataFrame[d\_BusinessDate: date, stcd: string, srcnm: string, d\_NetworkName: string, d\_NetworkURL: string, custnum: string, MarketCoverage: string, RejectRows: string, d\_IssuerId: string, IssuerId2: string, issueridcomposite: string, sysdt: date, systs: timestamp, cleannetworkname: string]>

>>> result\_df\_SQL = result\_df\_SQL \

... .withColumnRenamed("a\_IssuerId", "IssuerId") \

... .withColumnRenamed("a\_NetworkName", "NetworkName") \

... .withColumnRenamed("a\_NetworkURL", "NetworkURL") \

... .withColumnRenamed("a\_BusinessDate", "BusinessDate")

>>> result\_df\_SQL\_1 = result\_df\_SQL\_1 \

... .withColumnRenamed("b\_IssuerId", "IssuerId") \

... .withColumnRenamed("b\_NetworkName", "NetworkName") \

... .withColumnRenamed("b\_NetworkURL", "NetworkURL") \

... .withColumnRenamed("b\_BusinessDate", "BusinessDate")

>>> result\_df\_SQL\_3 = result\_df\_SQL\_3 \

... .withColumnRenamed("d\_IssuerId", "IssuerId") \

... .withColumnRenamed("d\_NetworkName", "NetworkName") \

... .withColumnRenamed("d\_NetworkURL", "NetworkURL") \

... .withColumnRenamed("d\_BusinessDate", "BusinessDate")

>>> result\_df\_SQL\_2 = result\_df\_SQL\_2 \

... .withColumnRenamed("c\_IssuerId", "IssuerId") \

... .withColumnRenamed("c\_NetworkName", "NetworkName") \

... .withColumnRenamed("c\_NetworkURL", "NetworkURL") \

... .withColumnRenamed("c\_BusinessDate", "BusinessDate")

>>> df\_combined = result\_df\_SQL \

... .alias("a") \

... .join(result\_df\_SQL\_1.alias("b"), col("a.IssuerId") == col("b.IssuerId"), how="inner") \

... .join(result\_df\_SQL\_2.alias("c"), col("a.IssuerId") == col("c.IssuerId"), how="inner") \

... .join(result\_df\_SQL\_3.alias("d"), col("a.IssuerId") == col("d.IssuerId"), how="inner")

>>> df\_combined.columns()

Traceback (most recent call last):

File "<stdin>", line 1, in <module>

TypeError: 'list' object is not callable

>>> list=df\_combined.columns()

Traceback (most recent call last):

File "<stdin>", line 1, in <module>

TypeError: 'list' object is not callable

>>> list\_ele=df\_combined.columns()

Traceback (most recent call last):

File "<stdin>", line 1, in <module>

TypeError: 'list' object is not callable

>>> list\_ele=df\_combined.columns

>>> print(list\_Ele)

Traceback (most recent call last):

File "<stdin>", line 1, in <module>

NameError: name 'list\_Ele' is not defined

>>> print(list\_ele)

['BusinessDate', 'stcd', 'srcnm', 'NetworkName', 'NetworkURL', 'custnum', 'MarketCoverage', 'RejectRows', 'IssuerId', 'IssuerId2', 'issueridcomposite', 'sysdt', 'systs', 'month', 'year', 'BusinessDate', 'stcd', 'srcnm', 'NetworkName', 'NetworkURL', 'custnum', 'MarketCoverage', 'RejectRows', 'IssuerId', 'IssuerId2', 'issueridcomposite', 'sysdt', 'systs', 'Protocol\_Used', 'BusinessDate', 'stcd', 'srcnm', 'NetworkName', 'NetworkURL', 'custnum', 'MarketCoverage', 'RejectRows', 'IssuerId', 'IssuerId2', 'issueridcomposite', 'sysdt', 'systs', 'curdt', 'curts', 'BusinessDate', 'stcd', 'srcnm', 'NetworkName', 'NetworkURL', 'custnum', 'MarketCoverage', 'RejectRows', 'IssuerId', 'IssuerId2', 'issueridcomposite', 'sysdt', 'systs', 'cleannetworkname']

>>> df\_combined.write.parquet("hdfs:///user/hduser/Hackathon/PARQUET\_DATA")

Traceback (most recent call last):

File "<stdin>", line 1, in <module>

File "/usr/local/spark/python/pyspark/sql/readwriter.py", line 885, in parquet

self.\_jwrite.parquet(path)

File "/usr/local/spark/python/lib/py4j-0.10.9.3-src.zip/py4j/java\_gateway.py", line 1322, in \_\_call\_\_

File "/usr/local/spark/python/pyspark/sql/utils.py", line 117, in deco

raise converted from None

pyspark.sql.utils.AnalysisException: Found duplicate column(s) when inserting into hdfs://localhost:54310/user/hduser/Hackathon/PARQUET\_DATA: `businessdate`, `custnum`, `issuerid2`, `issuerid`, `issueridcomposite`, `marketcoverage`, `networkname`, `networkurl`, `rejectrows`, `srcnm`, `stcd`, `sysdt`, `systs`

>>> result\_df=df\_combined.drop("")

>>> result\_df\_SQL\_renamed = result\_df\_SQL \

... .withColumnRenamed("a\_BusinessDate", "a\_BusinessDate") \

... .withColumnRenamed("stcd", "a\_stcd") \

... .withColumnRenamed("srcnm", "a\_srcnm") \

... .withColumnRenamed("a\_NetworkName", "a\_NetworkName") \

... .withColumnRenamed("a\_NetworkURL", "a\_NetworkURL") \

... .withColumnRenamed("custnum", "a\_custnum") \

... .withColumnRenamed("MarketCoverage", "a\_MarketCoverage") \

... .withColumnRenamed("RejectRows", "a\_RejectRows") \

... .withColumnRenamed("a\_IssuerId", "a\_IssuerId") \

... .withColumnRenamed("IssuerId2", "a\_IssuerId2") \

... .withColumnRenamed("issueridcomposite", "a\_issueridcomposite") \

... .withColumnRenamed("sysdt", "a\_sysdt") \

... .withColumnRenamed("systs", "a\_systs") \

... .withColumnRenamed("month", "a\_month") \

... .withColumnRenamed("year", "a\_year")

>>> result\_df\_SQL\_1\_renamed = result\_df\_SQL\_1 \

... .withColumnRenamed("b\_BusinessDate", "b\_BusinessDate") \

... .withColumnRenamed("stcd", "b\_stcd") \

... .withColumnRenamed("srcnm", "b\_srcnm") \

... .withColumnRenamed("b\_NetworkName", "b\_NetworkName") \

... .withColumnRenamed("b\_NetworkURL", "b\_NetworkURL") \

... .withColumnRenamed("custnum", "b\_custnum") \

... .withColumnRenamed("MarketCoverage", "b\_MarketCoverage") \

... .withColumnRenamed("RejectRows", "b\_RejectRows") \

... .withColumnRenamed("b\_IssuerId", "b\_IssuerId") \

... .withColumnRenamed("IssuerId2", "b\_IssuerId2") \

... .withColumnRenamed("issueridcomposite", "b\_issueridcomposite") \

... .withColumnRenamed("sysdt", "b\_sysdt") \

... .withColumnRenamed("systs", "b\_systs") \

... .withColumnRenamed("Protocol\_Used", "b\_Protocol\_Used")

>>>

>>> result\_df\_SQL\_2\_renamed = result\_df\_SQL\_2 \

... .withColumnRenamed("c\_BusinessDate", "c\_BusinessDate") \

... .withColumnRenamed("stcd", "c\_stcd") \

... .withColumnRenamed("srcnm", "c\_srcnm") \

... .withColumnRenamed("c\_NetworkName", "c\_NetworkName") \

... .withColumnRenamed("c\_NetworkURL", "c\_NetworkURL") \

... .withColumnRenamed("custnum", "c\_custnum") \

... .withColumnRenamed("MarketCoverage", "c\_MarketCoverage") \

... .withColumnRenamed("RejectRows", "c\_RejectRows") \

... .withColumnRenamed("c\_IssuerId", "c\_IssuerId") \

... .withColumnRenamed("IssuerId2", "c\_IssuerId2") \

... .withColumnRenamed("issueridcomposite", "c\_issueridcomposite") \

... .withColumnRenamed("sysdt", "c\_sysdt") \

... .withColumnRenamed("systs", "c\_systs") \

... .withColumnRenamed("curdt", "c\_curdt") \

... .withColumnRenamed("curts", "c\_curts")

>>> result\_df\_SQL\_3\_renamed = result\_df\_SQL\_3 \

... .withColumnRenamed("d\_BusinessDate", "d\_BusinessDate") \

... .withColumnRenamed("stcd", "d\_stcd") \

... .withColumnRenamed("srcnm", "d\_srcnm") \

... .withColumnRenamed("d\_NetworkName", "d\_NetworkName") \

... .withColumnRenamed("d\_NetworkURL", "d\_NetworkURL") \

... .withColumnRenamed("custnum", "d\_custnum") \

... .withColumnRenamed("MarketCoverage", "d\_MarketCoverage") \

... .withColumnRenamed("RejectRows", "d\_RejectRows") \

... .withColumnRenamed("d\_IssuerId", "d\_IssuerId") \

... .withColumnRenamed("IssuerId2", "d\_IssuerId2") \

... .withColumnRenamed("issueridcomposite", "d\_issueridcomposite") \

... .withColumnRenamed("sysdt", "d\_sysdt") \

... .withColumnRenamed("systs", "d\_systs") \

... .withColumnRenamed("cleannetworkname", "d\_cleannetworkname")

>>> final\_joined\_df = (result\_df\_SQL\_renamed

... .join(result\_df\_SQL\_1\_renamed,

... (F.col("a\_BusinessDate") == F.col("b\_BusinessDate")) &

... (F.col("a\_stcd") == F.col("b\_stcd")) &

... (F.col("a\_srcnm") == F.col("b\_srcnm")) &

... (F.col("a\_custnum") == F.col("b\_custnum")),

... "left")

... .join(result\_df\_SQL\_2\_renamed,

... (F.col("a\_BusinessDate") == F.col("c\_BusinessDate")) &

... (F.col("a\_stcd") == F.col("c\_stcd")) &

... (F.col("a\_srcnm") == F.col("c\_srcnm")) &

... (F.col("a\_custnum") == F.col("c\_custnum")),

... "left")

... .join(result\_df\_SQL\_3\_renamed,

... (F.col("a\_BusinessDate") == F.col("d\_BusinessDate")) &

... (F.col("a\_stcd") == F.col("d\_stcd")) &

... (F.col("a\_srcnm") == F.col("d\_srcnm")) &

... (F.col("a\_custnum") == F.col("d\_custnum")),

... "left")

... )

Traceback (most recent call last):

File "<stdin>", line 5, in <module>

NameError: name 'F' is not defined

>>> from pyspark.sql import functions as F

>>> final\_joined\_df = (result\_df\_SQL\_renamed

... .join(result\_df\_SQL\_1\_renamed,

... (F.col("a\_BusinessDate") == F.col("b\_BusinessDate")) &

... (F.col("a\_stcd") == F.col("b\_stcd")) &

... (F.col("a\_srcnm") == F.col("b\_srcnm")) &

... (F.col("a\_custnum") == F.col("b\_custnum")),

... "left")

... .join(result\_df\_SQL\_2\_renamed,

... (F.col("a\_BusinessDate") == F.col("c\_BusinessDate")) &

... (F.col("a\_stcd") == F.col("c\_stcd")) &

... (F.col("a\_srcnm") == F.col("c\_srcnm")) &

... (F.col("a\_custnum") == F.col("c\_custnum")),

... "left")

... .join(result\_df\_SQL\_3\_renamed,

... (F.col("a\_BusinessDate") == F.col("d\_BusinessDate")) &

... (F.col("a\_stcd") == F.col("d\_stcd")) &

... (F.col("a\_srcnm") == F.col("d\_srcnm")) &

... (F.col("a\_custnum") == F.col("d\_custnum")),

... "left")

... )

Traceback (most recent call last):

File "<stdin>", line 7, in <module>

File "/usr/local/spark/python/pyspark/sql/dataframe.py", line 1355, in join

jdf = self.\_jdf.join(other.\_jdf, on, how)

File "/usr/local/spark/python/lib/py4j-0.10.9.3-src.zip/py4j/java\_gateway.py", line 1322, in \_\_call\_\_

File "/usr/local/spark/python/pyspark/sql/utils.py", line 117, in deco

raise converted from None

pyspark.sql.utils.AnalysisException: cannot resolve 'a\_BusinessDate' given input columns: [BusinessDate, BusinessDate, IssuerId, IssuerId, NetworkName, NetworkName, NetworkURL, NetworkURL, a\_IssuerId2, a\_MarketCoverage, a\_RejectRows, a\_custnum, a\_issueridcomposite, a\_month, a\_srcnm, a\_stcd, a\_sysdt, a\_systs, a\_year, b\_IssuerId2, b\_MarketCoverage, b\_Protocol\_Used, b\_RejectRows, b\_custnum, b\_issueridcomposite, b\_srcnm, b\_stcd, b\_sysdt, b\_systs];

'Join LeftOuter, (((('a\_BusinessDate = 'b\_BusinessDate) AND (a\_stcd#46162 = b\_stcd#46397)) AND (a\_srcnm#46178 = b\_srcnm#46412)) AND (a\_custnum#46224 = b\_custnum#46455))

:- Project [BusinessDate#44187, a\_stcd#46162, a\_srcnm#46178, NetworkName#44155, NetworkURL#44171, a\_custnum#46224, a\_MarketCoverage#46240, a\_RejectRows#46256, IssuerId#44139, a\_IssuerId2#46287, a\_issueridcomposite#46303, a\_sysdt#46319, a\_systs#46335, a\_month#46351, year#30929 AS a\_year#46367]

: +- Project [BusinessDate#44187, a\_stcd#46162, a\_srcnm#46178, NetworkName#44155, NetworkURL#44171, a\_custnum#46224, a\_MarketCoverage#46240, a\_RejectRows#46256, IssuerId#44139, a\_IssuerId2#46287, a\_issueridcomposite#46303, a\_sysdt#46319, a\_systs#46335, month#30928 AS a\_month#46351, year#30929]

: +- Project [BusinessDate#44187, a\_stcd#46162, a\_srcnm#46178, NetworkName#44155, NetworkURL#44171, a\_custnum#46224, a\_MarketCoverage#46240, a\_RejectRows#46256, IssuerId#44139, a\_IssuerId2#46287, a\_issueridcomposite#46303, a\_sysdt#46319, systs#9016 AS a\_systs#46335, month#30928, year#30929]

: +- Project [BusinessDate#44187, a\_stcd#46162, a\_srcnm#46178, NetworkName#44155, NetworkURL#44171, a\_custnum#46224, a\_MarketCoverage#46240, a\_RejectRows#46256, IssuerId#44139, a\_IssuerId2#46287, a\_issueridcomposite#46303, sysdt#9003 AS a\_sysdt#46319, systs#9016, month#30928, year#30929]

: +- Project [BusinessDate#44187, a\_stcd#46162, a\_srcnm#46178, NetworkName#44155, NetworkURL#44171, a\_custnum#46224, a\_MarketCoverage#46240, a\_RejectRows#46256, IssuerId#44139, a\_IssuerId2#46287, issueridcomposite#8901 AS a\_issueridcomposite#46303, sysdt#9003, systs#9016, month#30928, year#30929]

: +- Project [BusinessDate#44187, a\_stcd#46162, a\_srcnm#46178, NetworkName#44155, NetworkURL#44171, a\_custnum#46224, a\_MarketCoverage#46240, a\_RejectRows#46256, IssuerId#44139, IssuerId2#8325 AS a\_IssuerId2#46287, issueridcomposite#8901, sysdt#9003, systs#9016, month#30928, year#30929]

: +- Project [BusinessDate#44187, a\_stcd#46162, a\_srcnm#46178, NetworkName#44155, NetworkURL#44171, a\_custnum#46224, a\_MarketCoverage#46240, RejectRows#1386 AS a\_RejectRows#46256, IssuerId#44139, IssuerId2#8325, issueridcomposite#8901, sysdt#9003, systs#9016, month#30928, year#30929]

: +- Project [BusinessDate#44187, a\_stcd#46162, a\_srcnm#46178, NetworkName#44155, NetworkURL#44171, a\_custnum#46224, MarketCoverage#1384 AS a\_MarketCoverage#46240, RejectRows#1386, IssuerId#44139, IssuerId2#8325, issueridcomposite#8901, sysdt#9003, systs#9016, month#30928, year#30929]

: +- Project [BusinessDate#44187, a\_stcd#46162, a\_srcnm#46178, NetworkName#44155, NetworkURL#44171, custnum#1383 AS a\_custnum#46224, MarketCoverage#1384, RejectRows#1386, IssuerId#44139, IssuerId2#8325, issueridcomposite#8901, sysdt#9003, systs#9016, month#30928, year#30929]

: +- Project [BusinessDate#44187, a\_stcd#46162, srcnm#7747 AS a\_srcnm#46178, NetworkName#44155, NetworkURL#44171, custnum#1383, MarketCoverage#1384, RejectRows#1386, IssuerId#44139, IssuerId2#8325, issueridcomposite#8901, sysdt#9003, systs#9016, month#30928, year#30929]

: +- Project [BusinessDate#44187, stcd#7759 AS a\_stcd#46162, srcnm#7747, NetworkName#44155, NetworkURL#44171, custnum#1383, MarketCoverage#1384, RejectRows#1386, IssuerId#44139, IssuerId2#8325, issueridcomposite#8901, sysdt#9003, systs#9016, month#30928, year#30929]

: +- Project [a\_BusinessDate#43854 AS BusinessDate#44187, stcd#7759, srcnm#7747, NetworkName#44155, NetworkURL#44171, custnum#1383, MarketCoverage#1384, RejectRows#1386, IssuerId#44139, IssuerId2#8325, issueridcomposite#8901, sysdt#9003, systs#9016, month#30928, year#30929]

: +- Project [a\_BusinessDate#43854, stcd#7759, srcnm#7747, NetworkName#44155, a\_NetworkURL#43806 AS NetworkURL#44171, custnum#1383, MarketCoverage#1384, RejectRows#1386, IssuerId#44139, IssuerId2#8325, issueridcomposite#8901, sysdt#9003, systs#9016, month#30928, year#30929]

: +- Project [a\_BusinessDate#43854, stcd#7759, srcnm#7747, a\_NetworkName#43822 AS NetworkName#44155, a\_NetworkURL#43806, custnum#1383, MarketCoverage#1384, RejectRows#1386, IssuerId#44139, IssuerId2#8325, issueridcomposite#8901, sysdt#9003, systs#9016, month#30928, year#30929]

: +- Project [a\_BusinessDate#43854, stcd#7759, srcnm#7747, a\_NetworkName#43822, a\_NetworkURL#43806, custnum#1383, MarketCoverage#1384, RejectRows#1386, a\_IssuerId#43838 AS IssuerId#44139, IssuerId2#8325, issueridcomposite#8901, sysdt#9003, systs#9016, month#30928, year#30929]

: +- Project [BusinessDate#1378 AS a\_BusinessDate#43854, stcd#7759, srcnm#7747, a\_NetworkName#43822, a\_NetworkURL#43806, custnum#1383, MarketCoverage#1384, RejectRows#1386, a\_IssuerId#43838, IssuerId2#8325, issueridcomposite#8901, sysdt#9003, systs#9016, month#30928, year#30929]

: +- Project [BusinessDate#1378, stcd#7759, srcnm#7747, a\_NetworkName#43822, a\_NetworkURL#43806, custnum#1383, MarketCoverage#1384, RejectRows#1386, IssuerId#8312 AS a\_IssuerId#43838, IssuerId2#8325, issueridcomposite#8901, sysdt#9003, systs#9016, month#30928, year#30929]

: +- Project [BusinessDate#1378, stcd#7759, srcnm#7747, NetworkName#1381 AS a\_NetworkName#43822, a\_NetworkURL#43806, custnum#1383, MarketCoverage#1384, RejectRows#1386, IssuerId#8312, IssuerId2#8325, issueridcomposite#8901, sysdt#9003, systs#9016, month#30928, year#30929]

: +- Project [BusinessDate#1378, stcd#7759, srcnm#7747, NetworkName#1381, NetworkURL#1382 AS a\_NetworkURL#43806, custnum#1383, MarketCoverage#1384, RejectRows#1386, IssuerId#8312, IssuerId2#8325, issueridcomposite#8901, sysdt#9003, systs#9016, month#30928, year#30929]

: +- Project [BusinessDate#1378, stcd#7759, srcnm#7747, NetworkName#1381, NetworkURL#1382, custnum#1383, MarketCoverage#1384, RejectRows#1386, IssuerId#8312, IssuerId2#8325, issueridcomposite#8901, sysdt#9003, systs#9016, month(BusinessDate#1378) AS month#30928, year(BusinessDate#1378) AS year#30929]

: +- SubqueryAlias insureview

: +- View (`insureview`, [BusinessDate#1378,stcd#7759,srcnm#7747,NetworkName#1381,NetworkURL#1382,custnum#1383,MarketCoverage#1384,RejectRows#1386,IssuerId#8312,IssuerId2#8325,issueridcomposite#8901,sysdt#9003,systs#9016])

: +- Project [BusinessDate#1378, stcd#7759, srcnm#7747, NetworkName#1381, NetworkURL#1382, custnum#1383, MarketCoverage#1384, RejectRows#1386, IssuerId#8312, IssuerId2#8325, issueridcomposite#8901, sysdt#9003, current\_timestamp() AS systs#9016]

: +- Project [BusinessDate#1378, stcd#7759, srcnm#7747, NetworkName#1381, NetworkURL#1382, custnum#1383, MarketCoverage#1384, RejectRows#1386, IssuerId#8312, IssuerId2#8325, issueridcomposite#8901, current\_date(Some(Asia/Kolkata)) AS sysdt#9003]

: +- Project [BusinessDate#1378, stcd#7759, srcnm#7747, NetworkName#1381, NetworkURL#1382, custnum#1383, MarketCoverage#1384, RejectRows#1386, IssuerId#8312, IssuerId2#8325, issueridcomposite#8901]

: +- Project [BusinessDate#1378, stcd#7759, srcnm#7747, NetworkName#1381, NetworkURL#1382, custnum#1383, MarketCoverage#1384, DentalOnlyPlan#1385, RejectRows#1386, IssuerId#8312, IssuerId2#8325, concat(IssuerId#8312, IssuerId2#8325) AS issueridcomposite#8901]

: +- Project [BusinessDate#1378, stcd#7759, srcnm#7747, NetworkName#1381, NetworkURL#1382, custnum#1383, MarketCoverage#1384, DentalOnlyPlan#1385, RejectRows#1386, IssuerId#8312, IssuerId2#8325]

: +- Project [IssuerIDDD#8273, BusinessDate#1378, stcd#7759, srcnm#7747, NetworkName#1381, NetworkURL#1382, custnum#1383, MarketCoverage#1384, DentalOnlyPlan#1385, RejectRows#1386, IssuerId#8312, IssuerId2#8325]

: +- Project [IssuerIDD#8261, IssuerIDDD#8273, BusinessDate#1378, stcd#7759, srcnm#7747, NetworkName#1381, NetworkURL#1382, custnum#1383, MarketCoverage#1384, DentalOnlyPlan#1385, RejectRows#1386, IssuerId#8312, cast(IssuerIDDD#8273 as string) AS IssuerId2#8325]

: +- Project [IssuerIDD#8261, IssuerIDDD#8273, BusinessDate#1378, stcd#7759, srcnm#7747, NetworkName#1381, NetworkURL#1382, custnum#1383, MarketCoverage#1384, DentalOnlyPlan#1385, RejectRows#1386, cast(IssuerIDD#8261 as string) AS IssuerId#8312]

: +- Project [IssuerIDD#8261, IssuerId2#1377 AS IssuerIDDD#8273, BusinessDate#1378, stcd#7759, srcnm#7747, NetworkName#1381, NetworkURL#1382, custnum#1383, MarketCoverage#1384, DentalOnlyPlan#1385, RejectRows#1386]

: +- Project [IssuerId#1376 AS IssuerIDD#8261, IssuerId2#1377, BusinessDate#1378, stcd#7759, srcnm#7747, NetworkName#1381, NetworkURL#1382, custnum#1383, MarketCoverage#1384, DentalOnlyPlan#1385, RejectRows#1386]

: +- Project [IssuerId#1376, IssuerId2#1377, BusinessDate#1378, StateCode#1379 AS stcd#7759, srcnm#7747, NetworkName#1381, NetworkURL#1382, custnum#1383, MarketCoverage#1384, DentalOnlyPlan#1385, RejectRows#1386]

: +- Project [IssuerId#1376, IssuerId2#1377, BusinessDate#1378, StateCode#1379, SourceName#1380 AS srcnm#7747, NetworkName#1381, NetworkURL#1382, custnum#1383, MarketCoverage#1384, DentalOnlyPlan#1385, RejectRows#1386]

: +- Union false, false

: :- Relation [IssuerId#1376,IssuerId2#1377,BusinessDate#1378,StateCode#1379,SourceName#1380,NetworkName#1381,NetworkURL#1382,custnum#1383,MarketCoverage#1384,DentalOnlyPlan#1385,RejectRows#1386] csv

: +- Relation [IssuerId#491,IssuerId2#492,BusinessDate#493,StateCode#494,SourceName#495,NetworkName#496,NetworkURL#497,custnum#498,MarketCoverage#499,DentalOnlyPlan#500,RejectRows#501] csv

+- Project [BusinessDate#44248, b\_stcd#46397, b\_srcnm#46412, NetworkName#44218, NetworkURL#44233, b\_custnum#46455, b\_MarketCoverage#46470, b\_RejectRows#46485, IssuerId#44203, b\_IssuerId2#46514, b\_issueridcomposite#46529, b\_sysdt#46544, b\_systs#46559, Protocol\_Used#30945 AS b\_Protocol\_Used#46574]

+- Project [BusinessDate#44248, b\_stcd#46397, b\_srcnm#46412, NetworkName#44218, NetworkURL#44233, b\_custnum#46455, b\_MarketCoverage#46470, b\_RejectRows#46485, IssuerId#44203, b\_IssuerId2#46514, b\_issueridcomposite#46529, b\_sysdt#46544, systs#9016 AS b\_systs#46559, Protocol\_Used#30945]

+- Project [BusinessDate#44248, b\_stcd#46397, b\_srcnm#46412, NetworkName#44218, NetworkURL#44233, b\_custnum#46455, b\_MarketCoverage#46470, b\_RejectRows#46485, IssuerId#44203, b\_IssuerId2#46514, b\_issueridcomposite#46529, sysdt#9003 AS b\_sysdt#46544, systs#9016, Protocol\_Used#30945]

+- Project [BusinessDate#44248, b\_stcd#46397, b\_srcnm#46412, NetworkName#44218, NetworkURL#44233, b\_custnum#46455, b\_MarketCoverage#46470, b\_RejectRows#46485, IssuerId#44203, b\_IssuerId2#46514, issueridcomposite#8901 AS b\_issueridcomposite#46529, sysdt#9003, systs#9016, Protocol\_Used#30945]

+- Project [BusinessDate#44248, b\_stcd#46397, b\_srcnm#46412, NetworkName#44218, NetworkURL#44233, b\_custnum#46455, b\_MarketCoverage#46470, b\_RejectRows#46485, IssuerId#44203, IssuerId2#8325 AS b\_IssuerId2#46514, issueridcomposite#8901, sysdt#9003, systs#9016, Protocol\_Used#30945]

+- Project [BusinessDate#44248, b\_stcd#46397, b\_srcnm#46412, NetworkName#44218, NetworkURL#44233, b\_custnum#46455, b\_MarketCoverage#46470, RejectRows#47041 AS b\_RejectRows#46485, IssuerId#44203, IssuerId2#8325, issueridcomposite#8901, sysdt#9003, systs#9016, Protocol\_Used#30945]

+- Project [BusinessDate#44248, b\_stcd#46397, b\_srcnm#46412, NetworkName#44218, NetworkURL#44233, b\_custnum#46455, MarketCoverage#47039 AS b\_MarketCoverage#46470, RejectRows#47041, IssuerId#44203, IssuerId2#8325, issueridcomposite#8901, sysdt#9003, systs#9016, Protocol\_Used#30945]

+- Project [BusinessDate#44248, b\_stcd#46397, b\_srcnm#46412, NetworkName#44218, NetworkURL#44233, custnum#47038 AS b\_custnum#46455, MarketCoverage#47039, RejectRows#47041, IssuerId#44203, IssuerId2#8325, issueridcomposite#8901, sysdt#9003, systs#9016, Protocol\_Used#30945]

+- Project [BusinessDate#44248, b\_stcd#46397, srcnm#7747 AS b\_srcnm#46412, NetworkName#44218, NetworkURL#44233, custnum#47038, MarketCoverage#47039, RejectRows#47041, IssuerId#44203, IssuerId2#8325, issueridcomposite#8901, sysdt#9003, systs#9016, Protocol\_Used#30945]

+- Project [BusinessDate#44248, stcd#7759 AS b\_stcd#46397, srcnm#7747, NetworkName#44218, NetworkURL#44233, custnum#47038, MarketCoverage#47039, RejectRows#47041, IssuerId#44203, IssuerId2#8325, issueridcomposite#8901, sysdt#9003, systs#9016, Protocol\_Used#30945]

+- Project [b\_BusinessDate#43915 AS BusinessDate#44248, stcd#7759, srcnm#7747, NetworkName#44218, NetworkURL#44233, custnum#47038, MarketCoverage#47039, RejectRows#47041, IssuerId#44203, IssuerId2#8325, issueridcomposite#8901, sysdt#9003, systs#9016, Protocol\_Used#30945]

+- Project [b\_BusinessDate#43915, stcd#7759, srcnm#7747, NetworkName#44218, b\_NetworkURL#43870 AS NetworkURL#44233, custnum#47038, MarketCoverage#47039, RejectRows#47041, IssuerId#44203, IssuerId2#8325, issueridcomposite#8901, sysdt#9003, systs#9016, Protocol\_Used#30945]

+- Project [b\_BusinessDate#43915, stcd#7759, srcnm#7747, b\_NetworkName#43885 AS NetworkName#44218, b\_NetworkURL#43870, custnum#47038, MarketCoverage#47039, RejectRows#47041, IssuerId#44203, IssuerId2#8325, issueridcomposite#8901, sysdt#9003, systs#9016, Protocol\_Used#30945]

+- Project [b\_BusinessDate#43915, stcd#7759, srcnm#7747, b\_NetworkName#43885, b\_NetworkURL#43870, custnum#47038, MarketCoverage#47039, RejectRows#47041, b\_IssuerId#43900 AS IssuerId#44203, IssuerId2#8325, issueridcomposite#8901, sysdt#9003, systs#9016, Protocol\_Used#30945]

+- Project [BusinessDate#47033 AS b\_BusinessDate#43915, stcd#7759, srcnm#7747, b\_NetworkName#43885, b\_NetworkURL#43870, custnum#47038, MarketCoverage#47039, RejectRows#47041, b\_IssuerId#43900, IssuerId2#8325, issueridcomposite#8901, sysdt#9003, systs#9016, Protocol\_Used#30945]

+- Project [BusinessDate#47033, stcd#7759, srcnm#7747, b\_NetworkName#43885, b\_NetworkURL#43870, custnum#47038, MarketCoverage#47039, RejectRows#47041, IssuerId#8312 AS b\_IssuerId#43900, IssuerId2#8325, issueridcomposite#8901, sysdt#9003, systs#9016, Protocol\_Used#30945]

+- Project [BusinessDate#47033, stcd#7759, srcnm#7747, NetworkName#47036 AS b\_NetworkName#43885, b\_NetworkURL#43870, custnum#47038, MarketCoverage#47039, RejectRows#47041, IssuerId#8312, IssuerId2#8325, issueridcomposite#8901, sysdt#9003, systs#9016, Protocol\_Used#30945]

+- Project [BusinessDate#47033, stcd#7759, srcnm#7747, NetworkName#47036, NetworkURL#47037 AS b\_NetworkURL#43870, custnum#47038, MarketCoverage#47039, RejectRows#47041, IssuerId#8312, IssuerId2#8325, issueridcomposite#8901, sysdt#9003, systs#9016, Protocol\_Used#30945]

+- Project [BusinessDate#47033, stcd#7759, srcnm#7747, NetworkName#47036, NetworkURL#47037, custnum#47038, MarketCoverage#47039, RejectRows#47041, IssuerId#8312, IssuerId2#8325, issueridcomposite#8901, sysdt#9003, systs#9016, CASE WHEN NetworkURL#47037 LIKE https://% THEN http secured WHEN NetworkURL#47037 LIKE http://% THEN http non secured ELSE No Protocol END AS Protocol\_Used#30945]

+- SubqueryAlias insureview

+- View (`insureview`, [BusinessDate#47033,stcd#7759,srcnm#7747,NetworkName#47036,NetworkURL#47037,custnum#47038,MarketCoverage#47039,RejectRows#47041,IssuerId#8312,IssuerId2#8325,issueridcomposite#8901,sysdt#9003,systs#9016])

+- Project [BusinessDate#47033, stcd#7759, srcnm#7747, NetworkName#47036, NetworkURL#47037, custnum#47038, MarketCoverage#47039, RejectRows#47041, IssuerId#8312, IssuerId2#8325, issueridcomposite#8901, sysdt#9003, current\_timestamp() AS systs#9016]

+- Project [BusinessDate#47033, stcd#7759, srcnm#7747, NetworkName#47036, NetworkURL#47037, custnum#47038, MarketCoverage#47039, RejectRows#47041, IssuerId#8312, IssuerId2#8325, issueridcomposite#8901, current\_date(Some(Asia/Kolkata)) AS sysdt#9003]

+- Project [BusinessDate#47033, stcd#7759, srcnm#7747, NetworkName#47036, NetworkURL#47037, custnum#47038, MarketCoverage#47039, RejectRows#47041, IssuerId#8312, IssuerId2#8325, issueridcomposite#8901]

+- Project [BusinessDate#47033, stcd#7759, srcnm#7747, NetworkName#47036, NetworkURL#47037, custnum#47038, MarketCoverage#47039, DentalOnlyPlan#47040, RejectRows#47041, IssuerId#8312, IssuerId2#8325, concat(IssuerId#8312, IssuerId2#8325) AS issueridcomposite#8901]

+- Project [BusinessDate#47033, stcd#7759, srcnm#7747, NetworkName#47036, NetworkURL#47037, custnum#47038, MarketCoverage#47039, DentalOnlyPlan#47040, RejectRows#47041, IssuerId#8312, IssuerId2#8325]

+- Project [IssuerIDDD#8273, BusinessDate#47033, stcd#7759, srcnm#7747, NetworkName#47036, NetworkURL#47037, custnum#47038, MarketCoverage#47039, DentalOnlyPlan#47040, RejectRows#47041, IssuerId#8312, IssuerId2#8325]

+- Project [IssuerIDD#8261, IssuerIDDD#8273, BusinessDate#47033, stcd#7759, srcnm#7747, NetworkName#47036, NetworkURL#47037, custnum#47038, MarketCoverage#47039, DentalOnlyPlan#47040, RejectRows#47041, IssuerId#8312, cast(IssuerIDDD#8273 as string) AS IssuerId2#8325]

+- Project [IssuerIDD#8261, IssuerIDDD#8273, BusinessDate#47033, stcd#7759, srcnm#7747, NetworkName#47036, NetworkURL#47037, custnum#47038, MarketCoverage#47039, DentalOnlyPlan#47040, RejectRows#47041, cast(IssuerIDD#8261 as string) AS IssuerId#8312]

+- Project [IssuerIDD#8261, IssuerId2#47032 AS IssuerIDDD#8273, BusinessDate#47033, stcd#7759, srcnm#7747, NetworkName#47036, NetworkURL#47037, custnum#47038, MarketCoverage#47039, DentalOnlyPlan#47040, RejectRows#47041]

+- Project [IssuerId#47031 AS IssuerIDD#8261, IssuerId2#47032, BusinessDate#47033, stcd#7759, srcnm#7747, NetworkName#47036, NetworkURL#47037, custnum#47038, MarketCoverage#47039, DentalOnlyPlan#47040, RejectRows#47041]

+- Project [IssuerId#47031, IssuerId2#47032, BusinessDate#47033, StateCode#47034 AS stcd#7759, srcnm#7747, NetworkName#47036, NetworkURL#47037, custnum#47038, MarketCoverage#47039, DentalOnlyPlan#47040, RejectRows#47041]

+- Project [IssuerId#47031, IssuerId2#47032, BusinessDate#47033, StateCode#47034, SourceName#47035 AS srcnm#7747, NetworkName#47036, NetworkURL#47037, custnum#47038, MarketCoverage#47039, DentalOnlyPlan#47040, RejectRows#47041]

+- Union false, false

:- Relation [IssuerId#47031,IssuerId2#47032,BusinessDate#47033,StateCode#47034,SourceName#47035,NetworkName#47036,NetworkURL#47037,custnum#47038,MarketCoverage#47039,DentalOnlyPlan#47040,RejectRows#47041] csv

+- Relation [IssuerId#47042,IssuerId2#47043,BusinessDate#47044,StateCode#47045,SourceName#47046,NetworkName#47047,NetworkURL#47048,custnum#47049,MarketCoverage#47050,DentalOnlyPlan#47051,RejectRows#47052] csv

>>>

>>> # Perform inner join on all the renamed DataFrames

... df\_combined = (

... result\_df\_SQL\_renamed

... .join(result\_df\_SQL\_1\_renamed, on=['BusinessDate', 'stcd', 'srcnm', 'custnum', 'MarketCoverage', 'RejectRows', 'IssuerId', 'IssuerId2', 'issueridcomposite', 'sysdt', 'systs'], how='inner')

... .join(result\_df\_SQL\_2\_renamed, on=['BusinessDate', 'stcd', 'srcnm', 'custnum', 'MarketCoverage', 'RejectRows', 'IssuerId', 'IssuerId2', 'issueridcomposite', 'sysdt', 'systs'], how='inner')

... .join(result\_df\_SQL\_3\_renamed, on=['BusinessDate', 'stcd', 'srcnm', 'custnum', 'MarketCoverage', 'RejectRows', 'IssuerId', 'IssuerId2', 'issueridcomposite', 'sysdt', 'systs'], how='inner')

... )

Traceback (most recent call last):

File "<stdin>", line 4, in <module>

File "/usr/local/spark/python/pyspark/sql/dataframe.py", line 1355, in join

jdf = self.\_jdf.join(other.\_jdf, on, how)

File "/usr/local/spark/python/lib/py4j-0.10.9.3-src.zip/py4j/java\_gateway.py", line 1322, in \_\_call\_\_

File "/usr/local/spark/python/pyspark/sql/utils.py", line 117, in deco

raise converted from None

pyspark.sql.utils.AnalysisException: USING column `stcd` cannot be resolved on the left side of the join. The left-side columns: [BusinessDate, a\_stcd, a\_srcnm, NetworkName, NetworkURL, a\_custnum, a\_MarketCoverage, a\_RejectRows, IssuerId, a\_IssuerId2, a\_issueridcomposite, a\_sysdt, a\_systs, a\_month, a\_year]

>>>

>>> # Perform inner join on all the renamed DataFrames

... df\_combined = (

... result\_df\_SQL\_renamed

... .join(result\_df\_SQL\_1\_renamed,

... on=['a\_BusinessDate', 'a\_stcd', 'a\_srcnm', 'a\_custnum', 'a\_MarketCoverage', 'a\_RejectRows', 'a\_IssuerId', 'a\_IssuerId2', 'a\_issueridcomposite', 'a\_sysdt', 'a\_systs'],

... how='inner')

... .join(result\_df\_SQL\_2\_renamed,

... on=['a\_BusinessDate', 'a\_stcd', 'a\_srcnm', 'a\_custnum', 'a\_MarketCoverage', 'a\_RejectRows', 'a\_IssuerId', 'a\_IssuerId2', 'a\_issueridcomposite', 'a\_sysdt', 'a\_systs'],

... how='inner')

... .join(result\_df\_SQL\_3\_renamed,

... on=['a\_BusinessDate', 'a\_stcd', 'a\_srcnm', 'a\_custnum', 'a\_MarketCoverage', 'a\_RejectRows', 'a\_IssuerId', 'a\_IssuerId2', 'a\_issueridcomposite', 'a\_sysdt', 'a\_systs'],

... how='inner')

... )

Traceback (most recent call last):

File "<stdin>", line 6, in <module>

File "/usr/local/spark/python/pyspark/sql/dataframe.py", line 1355, in join

jdf = self.\_jdf.join(other.\_jdf, on, how)

File "/usr/local/spark/python/lib/py4j-0.10.9.3-src.zip/py4j/java\_gateway.py", line 1322, in \_\_call\_\_

File "/usr/local/spark/python/pyspark/sql/utils.py", line 117, in deco

raise converted from None

pyspark.sql.utils.AnalysisException: USING column `a\_BusinessDate` cannot be resolved on the left side of the join. The left-side columns: [BusinessDate, a\_stcd, a\_srcnm, NetworkName, NetworkURL, a\_custnum, a\_MarketCoverage, a\_RejectRows, IssuerId, a\_IssuerId2, a\_issueridcomposite, a\_sysdt, a\_systs, a\_month, a\_year]

>>> c

>>> print("result\_df\_SQL\_renamed columns:", result\_df\_SQL\_renamed.columns)

result\_df\_SQL\_renamed columns: ['BusinessDate', 'a\_stcd', 'a\_srcnm', 'NetworkName', 'NetworkURL', 'a\_custnum', 'a\_MarketCoverage', 'a\_RejectRows', 'IssuerId', 'a\_IssuerId2', 'a\_issueridcomposite', 'a\_sysdt', 'a\_systs', 'a\_month', 'a\_year']

>>> print("result\_df\_SQL\_1\_renamed columns:", result\_df\_SQL\_1\_renamed.columns)

result\_df\_SQL\_1\_renamed columns: ['BusinessDate', 'b\_stcd', 'b\_srcnm', 'NetworkName', 'NetworkURL', 'b\_custnum', 'b\_MarketCoverage', 'b\_RejectRows', 'IssuerId', 'b\_IssuerId2', 'b\_issueridcomposite', 'b\_sysdt', 'b\_systs', 'b\_Protocol\_Used']

>>> print("result\_df\_SQL\_2\_renamed columns:", result\_df\_SQL\_2\_renamed.columns)

result\_df\_SQL\_2\_renamed columns: ['BusinessDate', 'c\_stcd', 'c\_srcnm', 'NetworkName', 'NetworkURL', 'c\_custnum', 'c\_MarketCoverage', 'c\_RejectRows', 'IssuerId', 'c\_IssuerId2', 'c\_issueridcomposite', 'c\_sysdt', 'c\_systs', 'c\_curdt', 'c\_curts']

>>> print("result\_df\_SQL\_3\_renamed columns:", result\_df\_SQL\_3\_renamed.columns)

result\_df\_SQL\_3\_renamed columns: ['BusinessDate', 'd\_stcd', 'd\_srcnm', 'NetworkName', 'NetworkURL', 'd\_custnum', 'd\_MarketCoverage', 'd\_RejectRows', 'IssuerId', 'd\_IssuerId2', 'd\_issueridcomposite', 'd\_sysdt', 'd\_systs', 'd\_cleannetworkname']

>>>

>>> # Perform inner join on all the renamed DataFrames

... df\_combined = (

... result\_df\_SQL\_renamed

... .join(result\_df\_SQL\_1\_renamed,

... on=['BusinessDate', 'a\_stcd', 'a\_srcnm', 'a\_custnum', 'a\_MarketCoverage', 'a\_RejectRows', 'IssuerId', 'a\_IssuerId2', 'a\_issueridcomposite', 'a\_sysdt', 'a\_systs'],

... how='inner')

... .join(result\_df\_SQL\_2\_renamed,

... on=['BusinessDate', 'a\_stcd', 'a\_srcnm', 'a\_custnum', 'a\_MarketCoverage', 'a\_RejectRows', 'IssuerId', 'a\_IssuerId2', 'a\_issueridcomposite', 'a\_sysdt', 'a\_systs'],

... how='inner')

... .join(result\_df\_SQL\_3\_renamed,

... on=['BusinessDate', 'a\_stcd', 'a\_srcnm', 'a\_custnum', 'a\_MarketCoverage', 'a\_RejectRows', 'IssuerId', 'a\_IssuerId2', 'a\_issueridcomposite', 'a\_sysdt', 'a\_systs'],

... how='inner')

... )

Traceback (most recent call last):

File "<stdin>", line 6, in <module>

File "/usr/local/spark/python/pyspark/sql/dataframe.py", line 1355, in join

jdf = self.\_jdf.join(other.\_jdf, on, how)

File "/usr/local/spark/python/lib/py4j-0.10.9.3-src.zip/py4j/java\_gateway.py", line 1322, in \_\_call\_\_

File "/usr/local/spark/python/pyspark/sql/utils.py", line 117, in deco

raise converted from None

pyspark.sql.utils.AnalysisException: USING column `a\_stcd` cannot be resolved on the right side of the join. The right-side columns: [BusinessDate, b\_stcd, b\_srcnm, NetworkName, NetworkURL, b\_custnum, b\_MarketCoverage, b\_RejectRows, IssuerId, b\_IssuerId2, b\_issueridcomposite, b\_sysdt, b\_systs, b\_Protocol\_Used]

>>>

>>> df\_combined = result\_df\_SQL\_renamed \

... .join(result\_df\_SQL\_1\_renamed, on="a\_IssuerId2", how="inner") \

... .join(result\_df\_SQL\_2\_renamed, on="a\_IssuerId2", how="inner") \

... .join(result\_df\_SQL\_3\_renamed, on="a\_IssuerId2", how="inner")

Traceback (most recent call last):

File "<stdin>", line 2, in <module>

File "/usr/local/spark/python/pyspark/sql/dataframe.py", line 1355, in join

jdf = self.\_jdf.join(other.\_jdf, on, how)

File "/usr/local/spark/python/lib/py4j-0.10.9.3-src.zip/py4j/java\_gateway.py", line 1322, in \_\_call\_\_

File "/usr/local/spark/python/pyspark/sql/utils.py", line 117, in deco

raise converted from None

pyspark.sql.utils.AnalysisException: USING column `a\_IssuerId2` cannot be resolved on the right side of the join. The right-side columns: [BusinessDate, b\_stcd, b\_srcnm, NetworkName, NetworkURL, b\_custnum, b\_MarketCoverage, b\_RejectRows, IssuerId, b\_IssuerId2, b\_issueridcomposite, b\_sysdt, b\_systs, b\_Protocol\_Used]

>>> result\_df\_SQL\_renamed.describe

<bound method DataFrame.describe of DataFrame[BusinessDate: date, a\_stcd: string, a\_srcnm: string, NetworkName: string, NetworkURL: string, a\_custnum: string, a\_MarketCoverage: string, a\_RejectRows: string, IssuerId: string, a\_IssuerId2: string, a\_issueridcomposite: string, a\_sysdt: date, a\_systs: timestamp, a\_month: int, a\_year: int]>

>>> .withColumnRenamed("curdt", "c\_curdt") \spark.sql("select \*,sql\_function(NetworkName) as cleannetworkname from insureview")

>>> spark.sql("select \*,sql\_function(NetworkName) as cleannetworkname from insureview").show(1)

+------------+----+-----+-----------+--------------------+-------+--------------+----------+--------+---------+-----------------+----------+--------------------+----------------+

|BusinessDate|stcd|srcnm|NetworkName| NetworkURL|custnum|MarketCoverage|RejectRows|IssuerId|IssuerId2|issueridcomposite| sysdt| systs|cleannetworkname|

+------------+----+-----+-----------+--------------------+-------+--------------+----------+--------+---------+-----------------+----------+--------------------+----------------+

| 2019-10-01| AK| HIOS|ODS Premier|https://www.modah...| 13| null| null| 21989| 21989| 2198921989|2024-07-22|2024-07-22 14:57:...| ODS Premier|

+------------+----+-----+-----------+--------------------+-------+--------------+----------+--------+---------+-----------------+----------+--------------------+----------------+

only showing top 1 row

>>> result\_df=spark.sql("select \*,sql\_function(NetworkName) as cleannetworkname from insureview")

>>> result\_df.createOrReplaceTempView("insureview")

Traceback (most recent call last):

File "<stdin>", line 1, in <module>

File "/usr/local/spark/python/pyspark/sql/dataframe.py", line 188, in createOrReplaceTempView

self.\_jdf.createOrReplaceTempView(name)

File "/usr/local/spark/python/lib/py4j-0.10.9.3-src.zip/py4j/java\_gateway.py", line 1322, in \_\_call\_\_

File "/usr/local/spark/python/pyspark/sql/utils.py", line 117, in deco

raise converted from None

pyspark.sql.utils.AnalysisException: Recursive view `insureview` detected (cycle: `insureview` -> `insureview`)

>>> spark.catalog.dropTempView("insureview")

>>> result\_df.createOrReplaceTempView("insureview")

>>> result\_df=spark.sql("select \*,current\_date as curdt , current\_timestamp as curts from insureview")

>>> spark.catalog.dropTempView("insureview")

>>> result\_df.createOrReplaceTempView("insureview")

>>> result\_df=spark.sql("select \*,sql\_function(NetworkName) as cleannetworkname from insureview")

>>> spark.sql("""select \*,case when NetworkURL like 'https://%' then 'http secured' when NetworkURL like 'http://%' then 'http non secured' else 'No Protocol' end as Protocol\_Used from insureview""").show(1)

+------------+----+-----+-----------+--------------------+-------+--------------+----------+--------+---------+-----------------+----------+--------------------+----------------+----------+--------------------+-------------+

|BusinessDate|stcd|srcnm|NetworkName| NetworkURL|custnum|MarketCoverage|RejectRows|IssuerId|IssuerId2|issueridcomposite| sysdt| systs|cleannetworkname| curdt| curts|Protocol\_Used|

+------------+----+-----+-----------+--------------------+-------+--------------+----------+--------+---------+-----------------+----------+--------------------+----------------+----------+--------------------+-------------+

| 2019-10-01| AK| HIOS|ODS Premier|https://www.modah...| 13| null| null| 21989| 21989| 2198921989|2024-07-22|2024-07-22 15:09:...| ODS Premier|2024-07-22|2024-07-22 15:09:...| http secured|

+------------+----+-----+-----------+--------------------+-------+--------------+----------+--------+---------+-----------------+----------+--------------------+----------------+----------+--------------------+-------------+

only showing top 1 row

>>> spark.catalog.dropTempView("insureview")

>>> result\_df.createOrReplaceTempView("insureview")

>>> result\_df=spark.sql("""select \*,case when NetworkURL like 'https://%' then 'http secured' when NetworkURL like 'http://%' then 'http non secured' else 'No Protocol' end as Protocol\_Used from insureview""")

>>>

>>> result\_df.show(1)

+------------+----+-----+-----------+--------------------+-------+--------------+----------+--------+---------+-----------------+----------+--------------------+----------------+----------+--------------------+----------------+-------------+

|BusinessDate|stcd|srcnm|NetworkName| NetworkURL|custnum|MarketCoverage|RejectRows|IssuerId|IssuerId2|issueridcomposite| sysdt| systs|cleannetworkname| curdt| curts|cleannetworkname|Protocol\_Used|

+------------+----+-----+-----------+--------------------+-------+--------------+----------+--------+---------+-----------------+----------+--------------------+----------------+----------+--------------------+----------------+-------------+

| 2019-10-01| AK| HIOS|ODS Premier|https://www.modah...| 13| null| null| 21989| 21989| 2198921989|2024-07-22|2024-07-22 15:11:...| ODS Premier|2024-07-22|2024-07-22 15:11:...| ODS Premier| http secured|

+------------+----+-----+-----------+--------------------+-------+--------------+----------+--------+---------+-----------------+----------+--------------------+----------------+----------+--------------------+----------------+-------------+

only showing top 1 row

>>> result\_df.describe

>>> result\_df.describe

<bound method DataFrame.describe of DataFrame[BusinessDate: date, stcd: string, srcnm: string, NetworkName: string, NetworkURL: string, custnum: string, MarketCoverage: string, RejectRows: string, IssuerId: string, IssuerId2: string, issueridcomposite: string, sysdt: date, systs: timestamp, cleannetworkname: string, curdt: date, curts: timestamp, cleannetworkname: string, Protocol\_Used: string]>

>>> statefilter\_df.describe

<bound method DataFrame.describe of DataFrame[\_c0: string, \_c1: string]>

>>> spark.sql("select \* from stateview").show(2)

Traceback (most recent call last):

File "<stdin>", line 1, in <module>

File "/usr/local/spark/python/pyspark/sql/session.py", line 723, in sql

return DataFrame(self.\_jsparkSession.sql(sqlQuery), self.\_wrapped)

File "/usr/local/spark/python/lib/py4j-0.10.9.3-src.zip/py4j/java\_gateway.py", line 1322, in \_\_call\_\_

File "/usr/local/spark/python/pyspark/sql/utils.py", line 117, in deco

raise converted from None

pyspark.sql.utils.AnalysisException: Table or view not found: stateview; line 1 pos 14;

'Project [\*]

+- 'UnresolvedRelation [stateview], [], false

>>> statefilter\_df.show(1)

+---+----------+

|\_c0| \_c1|

+---+----------+

| NM|New Mexico|

+---+----------+

only showing top 1 row

>>> spark.sql("select \* from cust\_table").show(2)

+---+--------+-----+---+-------+

|\_c0| \_c1| \_c2|\_c3| \_c4|

+---+--------+-----+---+-------+

| 1|Kristina|Chung| 55| Pilot|

| 2| Paige| Chen| 74|Teacher|

+---+--------+-----+---+-------+

only showing top 2 rows

>>> spark.sql("select \* from state\_table").show(2)

+---+----------+

|\_c0| \_c1|

+---+----------+

| NM|New Mexico|

| NY| New York|

+---+----------+

only showing top 2 rows

>>> spark.sql("select \* from insureview").show(2)

+------------+----+-----+-----------+--------------------+-------+--------------+--------------------+--------+---------+-----------------+----------+--------------------+----------------+----------+--------------------+----------------+

|BusinessDate|stcd|srcnm|NetworkName| NetworkURL|custnum|MarketCoverage| RejectRows|IssuerId|IssuerId2|issueridcomposite| sysdt| systs|cleannetworkname| curdt| curts|cleannetworkname|

+------------+----+-----+-----------+--------------------+-------+--------------+--------------------+--------+---------+-----------------+----------+--------------------+----------------+----------+--------------------+----------------+

| 2019-10-01| AK| HIOS|ODS Premier|https://www.modah...| 13| null| null| 21989| 21989| 2198921989|2024-07-22|2024-07-22 15:31:...| ODS Premier|2024-07-22|2024-07-22 15:31:...| ODS Premier|

| 2019-10-01| AK| HIOS|ODS Premier| null| null| null|219,219,2019-10-0...| 219| 219| 219219|2024-07-22|2024-07-22 15:31:...| ODS Premier|2024-07-22|2024-07-22 15:31:...| ODS Premier|

+------------+----+-----+-----------+--------------------+-------+--------------+--------------------+--------+---------+-----------------+----------+--------------------+----------------+----------+--------------------+----------------+

only showing top 2 rows

>>> spark.sql("select \* from insureview as l join state\_table as r on l.stcd=r.\_c0 ").show(1)

+------------+----+-----+-----------+--------------------+-------+--------------+----------+--------+---------+-----------------+----------+--------------------+----------------+----------+--------------------+----------------+---+------+

|BusinessDate|stcd|srcnm|NetworkName| NetworkURL|custnum|MarketCoverage|RejectRows|IssuerId|IssuerId2|issueridcomposite| sysdt| systs|cleannetworkname| curdt| curts|cleannetworkname|\_c0| \_c1|

+------------+----+-----+-----------+--------------------+-------+--------------+----------+--------+---------+-----------------+----------+--------------------+----------------+----------+--------------------+----------------+---+------+

| 2019-10-01| AK| HIOS|ODS Premier|https://www.modah...| 13| null| null| 21989| 21989| 2198921989|2024-07-22|2024-07-22 15:35:...| ODS Premier|2024-07-22|2024-07-22 15:35:...| ODS Premier| AK|Alaska|

+------------+----+-----+-----------+--------------------+-------+--------------+----------+--------+---------+-----------------+----------+--------------------+----------------+----------+--------------------+----------------+---+------+

only showing top 1 row

>>> first\_join=spark.sql("select \* from insureview as l join state\_table as r on l.stcd=r.\_c0 ")

>>> spark.catalog.dropTempview("insureview")

Traceback (most recent call last):

File "<stdin>", line 1, in <module>

AttributeError: 'Catalog' object has no attribute 'dropTempview'

>>> spark.catelog.dropTempview("insureview")

Traceback (most recent call last):

File "<stdin>", line 1, in <module>

AttributeError: 'SparkSession' object has no attribute 'catelog'

>>> spark.catalog.dropTempview("insureview")

Traceback (most recent call last):

File "<stdin>", line 1, in <module>

AttributeError: 'Catalog' object has no attribute 'dropTempview'

>>> spark.catalog.dropTempView("insureview")

>>> first\_join.describe

<bound method DataFrame.describe of DataFrame[BusinessDate: date, stcd: string, srcnm: string, NetworkName: string, NetworkURL: string, custnum: string, MarketCoverage: string, RejectRows: string, IssuerId: string, IssuerId2: string, issueridcomposite: string, sysdt: date, systs: timestamp, cleannetworkname: string, curdt: date, curts: timestamp, cleannetworkname: string, \_c0: string, \_c1: string]>

>>> first\_join.createOrReplaceTempView("insureview")

>>> spark.sql("select \* from insureview").show(1)

+------------+----+-----+-----------+--------------------+-------+--------------+----------+--------+---------+-----------------+----------+--------------------+----------------+----------+--------------------+----------------+---+------+

|BusinessDate|stcd|srcnm|NetworkName| NetworkURL|custnum|MarketCoverage|RejectRows|IssuerId|IssuerId2|issueridcomposite| sysdt| systs|cleannetworkname| curdt| curts|cleannetworkname|\_c0| \_c1|

+------------+----+-----+-----------+--------------------+-------+--------------+----------+--------+---------+-----------------+----------+--------------------+----------------+----------+--------------------+----------------+---+------+

| 2019-10-01| AK| HIOS|ODS Premier|https://www.modah...| 13| null| null| 21989| 21989| 2198921989|2024-07-22|2024-07-22 15:43:...| ODS Premier|2024-07-22|2024-07-22 15:43:...| ODS Premier| AK|Alaska|

+------------+----+-----+-----------+--------------------+-------+--------------+----------+--------+---------+-----------------+----------+--------------------+----------------+----------+--------------------+----------------+---+------+

only showing top 1 row

>>> spark.sql("select \* from insureview as l join cust\_table as r on l.custnum=r.\_c3").show(1)

+------------+----+-----+-----------+--------------------+-------+--------------+----------+--------+---------+-----------------+----------+--------------------+----------------+----------+--------------------+----------------+---+-------+---+-----+-------+---+-----+

|BusinessDate|stcd|srcnm|NetworkName| NetworkURL|custnum|MarketCoverage|RejectRows|IssuerId|IssuerId2|issueridcomposite| sysdt| systs|cleannetworkname| curdt| curts|cleannetworkname|\_c0| \_c1|\_c0| \_c1| \_c2|\_c3| \_c4|

+------------+----+-----+-----------+--------------------+-------+--------------+----------+--------+---------+-----------------+----------+--------------------+----------------+----------+--------------------+----------------+---+-------+---+-----+-------+---+-----+

| 2019-10-01| AZ| HIOS| La Paz|http://uamarketpl...| 21| null| null| 88925| 88925| 8892588925|2024-07-22|2024-07-22 15:48:...| La Paz|2024-07-22|2024-07-22 15:48:...| La Paz| AZ|Arizona|779|Jason|Bowling| 21|Coach|

+------------+----+-----+-----------+--------------------+-------+--------------+----------+--------+---------+-----------------+----------+--------------------+----------------+----------+--------------------+----------------+---+-------+---+-----+-------+---+-----+

only showing top 1 row

>>> spark.sql("select \* from insureview as l join cust\_table as r on l.custnum=r.\_c3").show(4)

+------------+----+-----+-----------+--------------------+-------+--------------+----------+--------+---------+-----------------+----------+--------------------+----------------+----------+--------------------+----------------+---+-------+---+-------+-------+---+----------+

|BusinessDate|stcd|srcnm|NetworkName| NetworkURL|custnum|MarketCoverage|RejectRows|IssuerId|IssuerId2|issueridcomposite| sysdt| systs|cleannetworkname| curdt| curts|cleannetworkname|\_c0| \_c1|\_c0| \_c1| \_c2|\_c3| \_c4|

+------------+----+-----+-----------+--------------------+-------+--------------+----------+--------+---------+-----------------+----------+--------------------+----------------+----------+--------------------+----------------+---+-------+---+-------+-------+---+----------+

| 2019-10-01| AZ| HIOS| La Paz|http://uamarketpl...| 21| null| null| 88925| 88925| 8892588925|2024-07-22|2024-07-22 15:49:...| La Paz|2024-07-22|2024-07-22 15:49:...| La Paz| AZ|Arizona|779| Jason|Bowling| 21| Coach|

| 2019-10-01| AZ| HIOS| La Paz|http://uamarketpl...| 21| null| null| 88925| 88925| 8892588925|2024-07-22|2024-07-22 15:49:...| La Paz|2024-07-22|2024-07-22 15:49:...| La Paz| AZ|Arizona|765| Janice| Bowden| 21|Politician|

| 2019-10-01| AZ| HIOS| La Paz|http://uamarketpl...| 21| null| null| 88925| 88925| 8892588925|2024-07-22|2024-07-22 15:49:...| La Paz|2024-07-22|2024-07-22 15:49:...| La Paz| AZ|Arizona|465| Peter|Guthrie| 21| Nurse|

| 2019-10-01| AZ| HIOS| La Paz|http://uamarketpl...| 21| null| null| 88925| 88925| 8892588925|2024-07-22|2024-07-22 15:49:...| La Paz|2024-07-22|2024-07-22 15:49:...| La Paz| AZ|Arizona|451|Russell| Hess| 21| Architect|

+------------+----+-----+-----------+--------------------+-------+--------------+----------+--------+---------+-----------------+----------+--------------------+----------------+----------+--------------------+----------------+---+-------+---+-------+-------+---+----------+

only showing top 4 rows

>>> second\_join=spark.sql("select \* from insureview as l join cust\_table as r on l.custnum=r.\_c3")

>>> second\_join.count()

684

>>> second\_join.write.parquet("/user/hduser/Hackathon/PARQUET\_DATA")

Traceback (most recent call last):

File "<stdin>", line 1, in <module>

File "/usr/local/spark/python/pyspark/sql/readwriter.py", line 885, in parquet

self.\_jwrite.parquet(path)

File "/usr/local/spark/python/lib/py4j-0.10.9.3-src.zip/py4j/java\_gateway.py", line 1322, in \_\_call\_\_

File "/usr/local/spark/python/pyspark/sql/utils.py", line 117, in deco

raise converted from None

pyspark.sql.utils.AnalysisException: Found duplicate column(s) when inserting into hdfs://localhost:54310/user/hduser/Hackathon/PARQUET\_DATA: `\_c0`, `\_c1`, `cleannetworkname`

>>> second\_join.show()

+------------+----+-----+-----------+--------------------+-------+--------------+----------+--------+---------+-----------------+----------+--------------------+----------------+----------+--------------------+----------------+---+-------+---+--------+---------+---+--------------------+

|BusinessDate|stcd|srcnm|NetworkName| NetworkURL|custnum|MarketCoverage|RejectRows|IssuerId|IssuerId2|issueridcomposite| sysdt| systs|cleannetworkname| curdt| curts|cleannetworkname|\_c0| \_c1|\_c0| \_c1| \_c2|\_c3| \_c4|

+------------+----+-----+-----------+--------------------+-------+--------------+----------+--------+---------+-----------------+----------+--------------------+----------------+----------+--------------------+----------------+---+-------+---+--------+---------+---+--------------------+

| 2019-10-01| AZ| HIOS| La Paz|http://uamarketpl...| 21| null| null| 88925| 88925| 8892588925|2024-07-22|2024-07-22 15:57:...| La Paz|2024-07-22|2024-07-22 15:57:...| La Paz| AZ|Arizona|779| Jason| Bowling| 21| Coach|

| 2019-10-01| AZ| HIOS| La Paz|http://uamarketpl...| 21| null| null| 88925| 88925| 8892588925|2024-07-22|2024-07-22 15:57:...| La Paz|2024-07-22|2024-07-22 15:57:...| La Paz| AZ|Arizona|765| Janice| Bowden| 21| Politician|

| 2019-10-01| AZ| HIOS| La Paz|http://uamarketpl...| 21| null| null| 88925| 88925| 8892588925|2024-07-22|2024-07-22 15:57:...| La Paz|2024-07-22|2024-07-22 15:57:...| La Paz| AZ|Arizona|465| Peter| Guthrie| 21| Nurse|

| 2019-10-01| AZ| HIOS| La Paz|http://uamarketpl...| 21| null| null| 88925| 88925| 8892588925|2024-07-22|2024-07-22 15:57:...| La Paz|2024-07-22|2024-07-22 15:57:...| La Paz| AZ|Arizona|451| Russell| Hess| 21| Architect|

| 2019-10-01| AZ| HIOS| La Paz|http://uamarketpl...| 21| null| null| 88925| 88925| 8892588925|2024-07-22|2024-07-22 15:57:...| La Paz|2024-07-22|2024-07-22 15:57:...| La Paz| AZ|Arizona|349| Russell| Dalton| 21| Civil engineer|

| 2019-10-01| AZ| HIOS| La Paz|http://uamarketpl...| 21| null| null| 88925| 88925| 8892588925|2024-07-22|2024-07-22 15:57:...| La Paz|2024-07-22|2024-07-22 15:57:...| La Paz| AZ|Arizona|292| Donna| Rice| 21| Social worker|

| 2019-10-01| AZ| HIOS| La Paz|http://uamarketpl...| 21| null| null| 88925| 88925| 8892588925|2024-07-22|2024-07-22 15:57:...| La Paz|2024-07-22|2024-07-22 15:57:...| La Paz| AZ|Arizona|290| Gail|Whitehead| 21| Nurse|

| 2019-10-01| AZ| HIOS| La Paz|http://uamarketpl...| 21| null| null| 88925| 88925| 8892588925|2024-07-22|2024-07-22 15:57:...| La Paz|2024-07-22|2024-07-22 15:57:...| La Paz| AZ|Arizona|176| Kristin|Alexander| 21| Coach|

| 2019-10-01| AZ| HIOS| La Paz|http://uamarketpl...| 21| null| null| 88925| 88925| 8892588925|2024-07-22|2024-07-22 15:57:...| La Paz|2024-07-22|2024-07-22 15:57:...| La Paz| AZ|Arizona| 52| Shirley| Merritt| 21| Reporter|

| 2019-10-01| AZ| HIOS| Maricopa|http://uamarketpl...| 22| null| null| 88925| 88925| 8892588925|2024-07-22|2024-07-22 15:57:...| Maricopa|2024-07-22|2024-07-22 15:57:...| Maricopa| AZ|Arizona|545|Danielle| Smith| 22| Pharmacist|

| 2019-10-01| AZ| HIOS| Maricopa|http://uamarketpl...| 22| null| null| 88925| 88925| 8892588925|2024-07-22|2024-07-22 15:57:...| Maricopa|2024-07-22|2024-07-22 15:57:...| Maricopa| AZ|Arizona|472| Kristin| Drake| 22| Financial analyst|

| 2019-10-01| AZ| HIOS| Maricopa|http://uamarketpl...| 22| null| null| 88925| 88925| 8892588925|2024-07-22|2024-07-22 15:57:...| Maricopa|2024-07-22|2024-07-22 15:57:...| Maricopa| AZ|Arizona|454|Virginia| Turner| 22|Computer hardware...|

| 2019-10-01| AZ| HIOS| Maricopa|http://uamarketpl...| 22| null| null| 88925| 88925| 8892588925|2024-07-22|2024-07-22 15:57:...| Maricopa|2024-07-22|2024-07-22 15:57:...| Maricopa| AZ|Arizona|359| Leon| Bender| 22| Firefighter|

| 2019-10-01| AZ| HIOS| Maricopa|http://uamarketpl...| 22| null| null| 88925| 88925| 8892588925|2024-07-22|2024-07-22 15:57:...| Maricopa|2024-07-22|2024-07-22 15:57:...| Maricopa| AZ|Arizona|218| Brett| Snyder| 22| Doctor|

| 2019-10-01| AZ| HIOS| Maricopa|http://uamarketpl...| 22| null| null| 88925| 88925| 8892588925|2024-07-22|2024-07-22 15:57:...| Maricopa|2024-07-22|2024-07-22 15:57:...| Maricopa| AZ|Arizona|152| Vincent| Goldman| 22| Electrician|

| 2019-10-01| AZ| HIOS| Maricopa|http://uamarketpl...| 22| null| null| 88925| 88925| 8892588925|2024-07-22|2024-07-22 15:57:...| Maricopa|2024-07-22|2024-07-22 15:57:...| Maricopa| AZ|Arizona| 54| Judith| Cooper| 22| Economist|

| 2019-10-01| AZ| HIOS| Mohave|http://uamarketpl...| 23| null| null| 88925| 88925| 8892588925|2024-07-22|2024-07-22 15:57:...| Mohave|2024-07-22|2024-07-22 15:57:...| Mohave| AZ|Arizona|752| Ethel| Stevens| 23| Designer|

| 2019-10-01| AZ| HIOS| Mohave|http://uamarketpl...| 23| null| null| 88925| 88925| 8892588925|2024-07-22|2024-07-22 15:57:...| Mohave|2024-07-22|2024-07-22 15:57:...| Mohave| AZ|Arizona|655| Hannah| Wall| 23| Librarian|

| 2019-10-01| AZ| HIOS| Mohave|http://uamarketpl...| 23| null| null| 88925| 88925| 8892588925|2024-07-22|2024-07-22 15:57:...| Mohave|2024-07-22|2024-07-22 15:57:...| Mohave| AZ|Arizona|639|Florence|McFarland| 23| Electrical engineer|

| 2019-10-01| AZ| HIOS| Mohave|http://uamarketpl...| 23| null| null| 88925| 88925| 8892588925|2024-07-22|2024-07-22 15:57:...| Mohave|2024-07-22|2024-07-22 15:57:...| Mohave| AZ|Arizona|248| Allison| Choi| 23|Computer hardware...|

+------------+----+-----+-----------+--------------------+-------+--------------+----------+--------+---------+-----------------+----------+--------------------+----------------+----------+--------------------+----------------+---+-------+---+--------+---------+---+--------------------+

only showing top 20 rows

>>> df\_renamed = second\_join.withColumnRenamed("\_c0", "state\_code") .withColumnRenamed("\_c1", "City")

>>> df.describe

Traceback (most recent call last):

File "<stdin>", line 1, in <module>

NameError: name 'df' is not defined

>>> df\_renamed.describe

<bound method DataFrame.describe of DataFrame[BusinessDate: date, stcd: string, srcnm: string, NetworkName: string, NetworkURL: string, custnum: string, MarketCoverage: string, RejectRows: string, IssuerId: string, IssuerId2: string, issueridcomposite: string, sysdt: date, systs: timestamp, cleannetworkname: string, curdt: date, curts: timestamp, cleannetworkname: string, state\_code: string, City: string, state\_code: string, City: string, \_c2: string, \_c3: int, \_c4: string]>

>>> first\_join=spark.sql("select \* from insureview as l join state\_table as r on l.stcd=r.\_c0 ")

>>>

>>> df\_renamed = df.select(

... col("BusinessDate"),

... col("stcd"),

... col("srcnm"),

... col("NetworkName"),

... col("NetworkURL"),

... col("custnum"),

... col("MarketCoverage"),

... col("RejectRows"),

... col("IssuerId"),

... col("IssuerId2"),

... col("issueridcomposite"),

... col("sysdt"),

... col("systs"),

... col("cleannetworkname").alias("cleannetworkname\_1"),

... col("curdt"),

... col("curts"),

... col("cleannetworkname").alias("cleannetworkname\_2"),

... col("state\_code").alias("state\_code\_1"),

... col("City").alias("City\_1"),

... col("state\_code").alias("state\_code\_2"),

... col("City").alias("City\_2"),

... col("\_c2"),

... col("\_c3"),

... col("\_c4")

... )

Traceback (most recent call last):

File "<stdin>", line 1, in <module>

NameError: name 'df' is not defined

>>> df\_renamed = df\_renamed.select(

... col("BusinessDate"),

... col("stcd"),

... col("srcnm"),

... col("NetworkName"),

... col("NetworkURL"),

... col("custnum"),

... col("MarketCoverage"),

... col("RejectRows"),

... col("IssuerId"),

... col("IssuerId2"),

... col("issueridcomposite"),

... col("sysdt"),

... col("systs"),

... col("cleannetworkname").alias("cleannetworkname\_1"),

... col("curdt"),

... col("curts"),

... col("cleannetworkname").alias("cleannetworkname\_2"),

... col("state\_code").alias("state\_code\_1"),

... col("City").alias("City\_1"),

... col("state\_code").alias("state\_code\_2"),

... col("City").alias("City\_2"),

... col("\_c2"),

... col("\_c3"),

... col("\_c4")

... )

Traceback (most recent call last):

File "<stdin>", line 25, in <module>

File "/usr/local/spark/python/pyspark/sql/dataframe.py", line 1685, in select

jdf = self.\_jdf.select(self.\_jcols(\*cols))

File "/usr/local/spark/python/lib/py4j-0.10.9.3-src.zip/py4j/java\_gateway.py", line 1322, in \_\_call\_\_

File "/usr/local/spark/python/pyspark/sql/utils.py", line 117, in deco

raise converted from None

pyspark.sql.utils.AnalysisException: Reference 'cleannetworkname' is ambiguous, could be: l.cleannetworkname, l.cleannetworkname.

>>> df\_temp = df\_named\

... .withColumnRenamed("cleannetworkname", "cleannetworkname\_temp") \

... .withColumnRenamed("state\_code", "state\_code\_temp") \

... .withColumnRenamed("City", "City\_temp")

Traceback (most recent call last):

File "<stdin>", line 1, in <module>

NameError: name 'df\_named' is not defined

>>>

>>> df\_temp = df\_renamed\

... .withColumnRenamed("cleannetworkname", "cleannetworkname\_temp") \

... .withColumnRenamed("state\_code", "state\_code\_temp") \

... .withColumnRenamed("City", "City\_temp")

>>>

>>> df\_renamed = df\_temp.select(

... col("BusinessDate"),

... col("stcd"),

... col("srcnm"),

... col("NetworkName"),

... col("NetworkURL"),

... col("custnum"),

... col("MarketCoverage"),

... col("RejectRows"),

... col("IssuerId"),

... col("IssuerId2"),

... col("issueridcomposite"),

... col("sysdt"),

... col("systs"),

... col("cleannetworkname\_temp").alias("cleannetworkname\_1"),

... col("curdt"),

... col("curts"),

... col("cleannetworkname\_temp").alias("cleannetworkname\_2"),

... col("state\_code\_temp").alias("state\_code\_1"),

... col("City\_temp").alias("City\_1"),

... col("state\_code\_temp").alias("state\_code\_2"),

... col("City\_temp").alias("City\_2"),

... col("\_c2"),

... col("\_c3"),

... col("\_c4")

... )

Traceback (most recent call last):

File "<stdin>", line 25, in <module>

File "/usr/local/spark/python/pyspark/sql/dataframe.py", line 1685, in select

jdf = self.\_jdf.select(self.\_jcols(\*cols))

File "/usr/local/spark/python/lib/py4j-0.10.9.3-src.zip/py4j/java\_gateway.py", line 1322, in \_\_call\_\_

File "/usr/local/spark/python/pyspark/sql/utils.py", line 117, in deco

raise converted from None

pyspark.sql.utils.AnalysisException: Reference 'cleannetworkname\_temp' is ambiguous, could be: cleannetworkname\_temp, cleannetworkname\_temp.

>>> df\_renamed = df\_temp.select(

... col("BusinessDate"),

... col("stcd"),

... col("srcnm"),

... col("NetworkName"),

... col("NetworkURL"),

... col("custnum"),

... col("MarketCoverage"),

... col("RejectRows"),

... col("IssuerId"),

... col("IssuerId2"),

... col("issueridcomposite"),

... col("sysdt"),

... col("systs"),

... col("cleannetworkname\_temp").alias("cleannetworkname\_1"),

... col("curdt"),

... col("curts"),

... col("cleannetworkname\_temp").alias("cleannetworkname\_2"),

... col("state\_code\_temp").alias("state\_code\_1"),

... col("City\_temp").alias("City\_1"),

... col("state\_code\_temp").alias("state\_code\_2"),

... col("City\_temp").alias("City\_2"),

... col("\_c2"),

... col("\_c3"),

... col("\_c4")

... )

Traceback (most recent call last):

File "<stdin>", line 25, in <module>

File "/usr/local/spark/python/pyspark/sql/dataframe.py", line 1685, in select

jdf = self.\_jdf.select(self.\_jcols(\*cols))

File "/usr/local/spark/python/lib/py4j-0.10.9.3-src.zip/py4j/java\_gateway.py", line 1322, in \_\_call\_\_

File "/usr/local/spark/python/pyspark/sql/utils.py", line 117, in deco

raise converted from None

pyspark.sql.utils.AnalysisException: Reference 'cleannetworkname\_temp' is ambiguous, could be: cleannetworkname\_temp, cleannetworkname\_temp.

>>> df\_temp.describe

<bound method DataFrame.describe of DataFrame[BusinessDate: date, stcd: string, srcnm: string, NetworkName: string, NetworkURL: string, custnum: string, MarketCoverage: string, RejectRows: string, IssuerId: string, IssuerId2: string, issueridcomposite: string, sysdt: date, systs: timestamp, cleannetworkname\_temp: string, curdt: date, curts: timestamp, cleannetworkname\_temp: string, state\_code\_temp: string, City\_temp: string, state\_code\_temp: string, City\_temp: string, \_c2: string, \_c3: int, \_c4: string]>

>>> df\_temp.drop("cleannetworkname\_temp")

DataFrame[BusinessDate: date, stcd: string, srcnm: string, NetworkName: string, NetworkURL: string, custnum: string, MarketCoverage: string, RejectRows: string, IssuerId: string, IssuerId2: string, issueridcomposite: string, sysdt: date, systs: timestamp, curdt: date, curts: timestamp, state\_code\_temp: string, City\_temp: string, state\_code\_temp: string, City\_temp: string, \_c2: string, \_c3: int, \_c4: string]

>>> df\_temp=df\_temp.drop("cleannetworkname\_temp")

>>> df\_temp.describe

<bound method DataFrame.describe of DataFrame[BusinessDate: date, stcd: string, srcnm: string, NetworkName: string, NetworkURL: string, custnum: string, MarketCoverage: string, RejectRows: string, IssuerId: string, IssuerId2: string, issueridcomposite: string, sysdt: date, systs: timestamp, curdt: date, curts: timestamp, state\_code\_temp: string, City\_temp: string, state\_code\_temp: string, City\_temp: string, \_c2: string, \_c3: int, \_c4: string]>

>>> df\_temp.write.parquet("hdfs:///user/hduser/Hackathon/PARQUET\_DATA")

Traceback (most recent call last):

File "<stdin>", line 1, in <module>

File "/usr/local/spark/python/pyspark/sql/readwriter.py", line 885, in parquet

self.\_jwrite.parquet(path)

File "/usr/local/spark/python/lib/py4j-0.10.9.3-src.zip/py4j/java\_gateway.py", line 1322, in \_\_call\_\_

File "/usr/local/spark/python/pyspark/sql/utils.py", line 117, in deco

raise converted from None

pyspark.sql.utils.AnalysisException: Found duplicate column(s) when inserting into hdfs://localhost:54310/user/hduser/Hackathon/PARQUET\_DATA: `city\_temp`, `state\_code\_temp`

>>> df\_temp=df\_temp.drop("city\_temp","state\_code\_temp")

>>> df\_temp.write.parquet("hdfs:///user/hduser/Hackathon/PARQUET\_DATA")

>>>

>>> WITH CountData AS (

File "<stdin>", line 1

WITH CountData AS (

^

SyntaxError: invalid syntax

>>> SELECT

File "<stdin>", line 1

SELECT

^

IndentationError: unexpected indent

>>> statedesc,

File "<stdin>", line 1

statedesc,

^

IndentationError: unexpected indent

>>> protocol,

File "<stdin>", line 1

protocol,

^

IndentationError: unexpected indent

>>> profession,

File "<stdin>", line 1

profession,

^

IndentationError: unexpected indent

>>> AVG(age) AS avg\_age, -- Calculating average age

File "<stdin>", line 1

AVG(age) AS avg\_age, -- Calculating average age

^

IndentationError: unexpected indent

>>> COUNT(\*) AS profession\_count, -- Counting number of professions

File "<stdin>", line 1

COUNT(\*) AS profession\_count, -- Counting number of professions

^

IndentationError: unexpected indent

>>> ROW\_NUMBER() OVER (PARTITION BY statedesc, protocol ORDER BY COUNT(\*) DESC) AS seqno -- Generating sequence number

File "<stdin>", line 1

ROW\_NUMBER() OVER (PARTITION BY statedesc, protocol ORDER BY COUNT(\*) DESC) AS seqno -- Generating sequence number

^

IndentationError: unexpected indent

>>> FROM your\_table

File "<stdin>", line 1

FROM your\_table

^

IndentationError: unexpected indent

>>> GROUP BY statedesc, protocol, profession

File "<stdin>", line 1

GROUP BY statedesc, protocol, profession

^

IndentationError: unexpected indent

>>> ),

File "<stdin>", line 1

),

^

SyntaxError: invalid syntax

>>> SecondHighestCount AS (

File "<stdin>", line 1

SecondHighestCount AS (

^

SyntaxError: invalid syntax

>>> SELECT

File "<stdin>", line 1

SELECT

^

IndentationError: unexpected indent

>>> statedesc,

File "<stdin>", line 1

statedesc,

^

IndentationError: unexpected indent

>>> protocol,

File "<stdin>", line 1

protocol,

^

IndentationError: unexpected indent

>>> profession,

File "<stdin>", line 1

profession,

^

IndentationError: unexpected indent

>>> avg\_age,

File "<stdin>", line 1

avg\_age,

^

IndentationError: unexpected indent

>>> profession\_count,

File "<stdin>", line 1

profession\_count,

^

IndentationError: unexpected indent

>>> seqno

File "<stdin>", line 1

seqno

^

IndentationError: unexpected indent

>>> FROM CountData

File "<stdin>", line 1

FROM CountData

^

IndentationError: unexpected indent

>>> WHERE seqno = 2 -- Filter to get the profession with the second highest count

File "<stdin>", line 1

WHERE seqno = 2 -- Filter to get the profession with the second highest count

^

IndentationError: unexpected indent

>>> )

File "<stdin>", line 1

)

^

SyntaxError: invalid syntax

>>> SELECT

Traceback (most recent call last):

File "<stdin>", line 1, in <module>

NameError: name 'SELECT' is not defined

>>> statedesc,

File "<stdin>", line 1

statedesc,

^

IndentationError: unexpected indent

>>> protocol,

File "<stdin>", line 1

protocol,

^

IndentationError: unexpected indent

>>> profession,

File "<stdin>", line 1

profession,

^

IndentationError: unexpected indent

>>> avg\_age,

File "<stdin>", line 1

avg\_age,

^

IndentationError: unexpected indent

>>> profession\_count

File "<stdin>", line 1

profession\_count

^

IndentationError: unexpected indent

>>> FROM SecondHighestCount

File "<stdin>", line 1

FROM SecondHighestCount

^

SyntaxError: invalid syntax

>>> ORDER BY statedesc, protocol, profession\_count DESC;

File "<stdin>", line 1

ORDER BY statedesc, protocol, profession\_count DESC;

^

SyntaxError: invalid syntax

>>>

>>> WITH CountData AS (

File "<stdin>", line 1

WITH CountData AS (

^

SyntaxError: invalid syntax

>>> SELECT

File "<stdin>", line 1

SELECT

^

IndentationError: unexpected indent

>>> statedesc,

File "<stdin>", line 1

statedesc,

^

IndentationError: unexpected indent

>>> protocol,

File "<stdin>", line 1

protocol,

^

IndentationError: unexpected indent

>>> profession,

File "<stdin>", line 1

profession,

^

IndentationError: unexpected indent

>>> AVG(age) AS avg\_age, -- Calculate average age

File "<stdin>", line 1

AVG(age) AS avg\_age, -- Calculate average age

^

IndentationError: unexpected indent

>>> COUNT(\*) AS profession\_count, -- Count occurrences of each profession

File "<stdin>", line 1

COUNT(\*) AS profession\_count, -- Count occurrences of each profession

^

IndentationError: unexpected indent

>>> ROW\_NUMBER() OVER (PARTITION BY statedesc, protocol ORDER BY COUNT(\*) DESC) AS seqno -- Generate sequence number

File "<stdin>", line 1

ROW\_NUMBER() OVER (PARTITION BY statedesc, protocol ORDER BY COUNT(\*) DESC) AS seqno -- Generate sequence number

^

IndentationError: unexpected indent

>>> FROM your\_table

File "<stdin>", line 1

FROM your\_table

^

IndentationError: unexpected indent

>>> GROUP BY statedesc, protocol, profession

File "<stdin>", line 1

GROUP BY statedesc, protocol, profession

^

IndentationError: unexpected indent

>>> ),

File "<stdin>", line 1

),

^

SyntaxError: invalid syntax

>>> SecondHighestCount AS (

File "<stdin>", line 1

SecondHighestCount AS (

^

SyntaxError: invalid syntax

>>> SELECT

File "<stdin>", line 1

SELECT

^

IndentationError: unexpected indent

>>> statedesc,

File "<stdin>", line 1

statedesc,

^

IndentationError: unexpected indent

>>> protocol,

File "<stdin>", line 1

protocol,

^

IndentationError: unexpected indent

>>> profession,

File "<stdin>", line 1

profession,

^

IndentationError: unexpected indent

>>> avg\_age,

File "<stdin>", line 1

avg\_age,

^

IndentationError: unexpected indent

>>> profession\_count

File "<stdin>", line 1

profession\_count

^

IndentationError: unexpected indent

>>> FROM CountData

File "<stdin>", line 1

FROM CountData

^

IndentationError: unexpected indent

>>> WHERE seqno = 2 -- Filter to get the profession with the second highest count

File "<stdin>", line 1

WHERE seqno = 2 -- Filter to get the profession with the second highest count

^

IndentationError: unexpected indent

>>> )query = """

File "<stdin>", line 1

)query = """

^

SyntaxError: invalid syntax

>>> WITH CountData AS (

File "<stdin>", line 1

WITH CountData AS (

^

SyntaxError: invalid syntax

>>> SELECT

File "<stdin>", line 1

SELECT

^

IndentationError: unexpected indent

>>> statedesc,

File "<stdin>", line 1

statedesc,

^

IndentationError: unexpected indent

>>> protocol,

File "<stdin>", line 1

protocol,

^

IndentationError: unexpected indent

>>> profession,

File "<stdin>", line 1

profession,

^

IndentationError: unexpected indent

>>> AVG(age) AS avg\_age,

File "<stdin>", line 1

AVG(age) AS avg\_age,

^

IndentationError: unexpected indent

>>> COUNT(\*) AS profession\_count,

File "<stdin>", line 1

COUNT(\*) AS profession\_count,

^

IndentationError: unexpected indent

>>> ROW\_NUMBER() OVER (PARTITION BY statedesc, protocol ORDER BY COUNT(\*) DESC) AS seqno

File "<stdin>", line 1

ROW\_NUMBER() OVER (PARTITION BY statedesc, protocol ORDER BY COUNT(\*) DESC) AS seqno

^

IndentationError: unexpected indent

>>> FROM your\_table

File "<stdin>", line 1

FROM your\_table

^

IndentationError: unexpected indent

>>> GROUP BY statedesc, protocol, profession

File "<stdin>", line 1

GROUP BY statedesc, protocol, profession

^

IndentationError: unexpected indent

>>> ),

File "<stdin>", line 1

),

^

SyntaxError: invalid syntax

>>> SecondHighestCount AS (

File "<stdin>", line 1

SecondHighestCount AS (

^

SyntaxError: invalid syntax

>>> SELECT

File "<stdin>", line 1

SELECT

^

IndentationError: unexpected indent

>>> statedesc,

File "<stdin>", line 1

statedesc,

^

IndentationError: unexpected indent

>>> protocol,

File "<stdin>", line 1

protocol,

^

IndentationError: unexpected indent

>>> profession,

File "<stdin>", line 1

profession,

^

IndentationError: unexpected indent

>>> avg\_age,

File "<stdin>", line 1

avg\_age,

^

IndentationError: unexpected indent

>>> profession\_count

File "<stdin>", line 1

profession\_count

^

IndentationError: unexpected indent

>>> FROM CountData

File "<stdin>", line 1

FROM CountData

^

IndentationError: unexpected indent

>>> WHERE seqno = 2

File "<stdin>", line 1

WHERE seqno = 2

^

IndentationError: unexpected indent

>>> )

File "<stdin>", line 1

)

^

SyntaxError: invalid syntax

>>> SELECT

Traceback (most recent call last):

File "<stdin>", line 1, in <module>

NameError: name 'SELECT' is not defined

>>> statedesc,

File "<stdin>", line 1

statedesc,

^

IndentationError: unexpected indent

>>> protocol,

File "<stdin>", line 1

protocol,

^

IndentationError: unexpected indent

>>> profession,

File "<stdin>", line 1

profession,

^

IndentationError: unexpected indent

>>> avg\_age,

File "<stdin>", line 1

avg\_age,

^

IndentationError: unexpected indent

>>> profession\_count

File "<stdin>", line 1

profession\_count

^

IndentationError: unexpected indent

>>> FROM SecondHighestCount

File "<stdin>", line 1

FROM SecondHighestCount

^

SyntaxError: invalid syntax

>>> ORDER BY statedesc, protocol;

File "<stdin>", line 1

ORDER BY statedesc, protocol;

^

SyntaxError: invalid syntax

>>> """\

...

...

...

...

... ;

...

File "<stdin>", line 5

;

^

SyntaxError: EOF while scanning triple-quoted string literal

>>>

>>> query = """

... WITH CountData AS (

... SELECT

... statedesc,

... protocol,

... profession,

... AVG(age) AS avg\_age,

... COUNT(\*) AS profession\_count,

... ROW\_NUMBER() OVER (PARTITION BY statedesc, protocol ORDER BY COUNT(\*) DESC) AS seqno

... FROM your\_table

... GROUP BY statedesc, protocol, profession

... ),

... SecondHighestCount AS (

... SELECT

... statedesc,

... protocol,

... profession,

... avg\_age,

... profession\_count

... FROM CountData

... WHERE seqno = 2

... )

... SELECT

... statedesc,

... protocol,

... profession,

... avg\_age,

... profession\_count

... FROM SecondHighestCount

... ORDER BY statedesc, protocol;

... """

>>>

>>> second\_join.describe

<bound method DataFrame.describe of DataFrame[BusinessDate: date, stcd: string, srcnm: string, NetworkName: string, NetworkURL: string, custnum: string, MarketCoverage: string, RejectRows: string, IssuerId: string, IssuerId2: string, issueridcomposite: string, sysdt: date, systs: timestamp, cleannetworkname: string, curdt: date, curts: timestamp, cleannetworkname: string, \_c0: string, \_c1: string, \_c0: string, \_c1: string, \_c2: string, \_c3: int, \_c4: string]>

>>> second\_join.show(2)

+------------+----+-----+-----------+--------------------+-------+--------------+----------+--------+---------+-----------------+----------+--------------------+----------------+----------+--------------------+----------------+---+-------+---+------+-------+---+----------+

|BusinessDate|stcd|srcnm|NetworkName| NetworkURL|custnum|MarketCoverage|RejectRows|IssuerId|IssuerId2|issueridcomposite| sysdt| systs|cleannetworkname| curdt| curts|cleannetworkname|\_c0| \_c1|\_c0| \_c1| \_c2|\_c3| \_c4|

+------------+----+-----+-----------+--------------------+-------+--------------+----------+--------+---------+-----------------+----------+--------------------+----------------+----------+--------------------+----------------+---+-------+---+------+-------+---+----------+

| 2019-10-01| AZ| HIOS| La Paz|http://uamarketpl...| 21| null| null| 88925| 88925| 8892588925|2024-07-22|2024-07-22 16:28:...| La Paz|2024-07-22|2024-07-22 16:28:...| La Paz| AZ|Arizona|779| Jason|Bowling| 21| Coach|

| 2019-10-01| AZ| HIOS| La Paz|http://uamarketpl...| 21| null| null| 88925| 88925| 8892588925|2024-07-22|2024-07-22 16:28:...| La Paz|2024-07-22|2024-07-22 16:28:...| La Paz| AZ|Arizona|765|Janice| Bowden| 21|Politician|

+------------+----+-----+-----------+--------------------+-------+--------------+----------+--------+---------+-----------------+----------+--------------------+----------------+----------+--------------------+----------------+---+-------+---+------+-------+---+----------+

only showing top 2 rows

>>> second\_join.createOrReplaceView("view1")

Traceback (most recent call last):

File "<stdin>", line 1, in <module>

File "/usr/local/spark/python/pyspark/sql/dataframe.py", line 1660, in \_\_getattr\_\_

"'%s' object has no attribute '%s'" % (self.\_\_class\_\_.\_\_name\_\_, name))

AttributeError: 'DataFrame' object has no attribute 'createOrReplaceView'

>>> second\_join.createOrReplaceTempView("view1")

>>> result\_df.show(2)

+------------+----+-----+-----------+--------------------+-------+--------------+--------------------+--------+---------+-----------------+----------+--------------------+----------------+----------+--------------------+----------------+-------------+

|BusinessDate|stcd|srcnm|NetworkName| NetworkURL|custnum|MarketCoverage| RejectRows|IssuerId|IssuerId2|issueridcomposite| sysdt| systs|cleannetworkname| curdt| curts|cleannetworkname|Protocol\_Used|

+------------+----+-----+-----------+--------------------+-------+--------------+--------------------+--------+---------+-----------------+----------+--------------------+----------------+----------+--------------------+----------------+-------------+

| 2019-10-01| AK| HIOS|ODS Premier|https://www.modah...| 13| null| null| 21989| 21989| 2198921989|2024-07-22|2024-07-22 16:33:...| ODS Premier|2024-07-22|2024-07-22 16:33:...| ODS Premier| http secured|

| 2019-10-01| AK| HIOS|ODS Premier| null| null| null|219,219,2019-10-0...| 219| 219| 219219|2024-07-22|2024-07-22 16:33:...| ODS Premier|2024-07-22|2024-07-22 16:33:...| ODS Premier| No Protocol|

+------------+----+-----+-----------+--------------------+-------+--------------+--------------------+--------+---------+-----------------+----------+--------------------+----------------+----------+--------------------+----------------+-------------+

only showing top 2 rows

>>> result\_df.createOrReplaceTempView("view2")

>>> spark.sql("select l.\*,r.Protocol\_Used from view1 as l and view2 as r ").show()

Traceback (most recent call last):

File "<stdin>", line 1, in <module>

File "/usr/local/spark/python/pyspark/sql/session.py", line 723, in sql

return DataFrame(self.\_jsparkSession.sql(sqlQuery), self.\_wrapped)

File "/usr/local/spark/python/lib/py4j-0.10.9.3-src.zip/py4j/java\_gateway.py", line 1322, in \_\_call\_\_

File "/usr/local/spark/python/pyspark/sql/utils.py", line 117, in deco

raise converted from None

pyspark.sql.utils.ParseException:

mismatched input 'and' expecting {<EOF>, ';'}(line 1, pos 43)

== SQL ==

select l.\*,r.Protocol\_Used from view1 as l and view2 as r

-------------------------------------------^^^

>>> spark.sql("select l.\*,r.Protocol\_Used from view1 as l,view2 as r ").show()

+------------+----+-----+-----------+--------------------+-------+--------------+----------+--------+---------+-----------------+----------+--------------------+----------------+----------+--------------------+----------------+---+-------+---+-----+-------+---+-----+----------------+

|BusinessDate|stcd|srcnm|NetworkName| NetworkURL|custnum|MarketCoverage|RejectRows|IssuerId|IssuerId2|issueridcomposite| sysdt| systs|cleannetworkname| curdt| curts|cleannetworkname|\_c0| \_c1|\_c0| \_c1| \_c2|\_c3| \_c4| Protocol\_Used|

+------------+----+-----+-----------+--------------------+-------+--------------+----------+--------+---------+-----------------+----------+--------------------+----------------+----------+--------------------+----------------+---+-------+---+-----+-------+---+-----+----------------+

| 2019-10-01| AZ| HIOS| La Paz|http://uamarketpl...| 21| null| null| 88925| 88925| 8892588925|2024-07-22|2024-07-22 16:37:...| La Paz|2024-07-22|2024-07-22 16:37:...| La Paz| AZ|Arizona|779|Jason|Bowling| 21|Coach| http secured|

| 2019-10-01| AZ| HIOS| La Paz|http://uamarketpl...| 21| null| null| 88925| 88925| 8892588925|2024-07-22|2024-07-22 16:37:...| La Paz|2024-07-22|2024-07-22 16:37:...| La Paz| AZ|Arizona|779|Jason|Bowling| 21|Coach| No Protocol|

| 2019-10-01| AZ| HIOS| La Paz|http://uamarketpl...| 21| null| null| 88925| 88925| 8892588925|2024-07-22|2024-07-22 16:37:...| La Paz|2024-07-22|2024-07-22 16:37:...| La Paz| AZ|Arizona|779|Jason|Bowling| 21|Coach| http secured|

| 2019-10-01| AZ| HIOS| La Paz|http://uamarketpl...| 21| null| null| 88925| 88925| 8892588925|2024-07-22|2024-07-22 16:37:...| La Paz|2024-07-22|2024-07-22 16:37:...| La Paz| AZ|Arizona|779|Jason|Bowling| 21|Coach|http non secured|

| 2019-10-01| AZ| HIOS| La Paz|http://uamarketpl...| 21| null| null| 88925| 88925| 8892588925|2024-07-22|2024-07-22 16:37:...| La Paz|2024-07-22|2024-07-22 16:37:...| La Paz| AZ|Arizona|779|Jason|Bowling| 21|Coach| http secured|

| 2019-10-01| AZ| HIOS| La Paz|http://uamarketpl...| 21| null| null| 88925| 88925| 8892588925|2024-07-22|2024-07-22 16:37:...| La Paz|2024-07-22|2024-07-22 16:37:...| La Paz| AZ|Arizona|779|Jason|Bowling| 21|Coach| http secured|

| 2019-10-01| AZ| HIOS| La Paz|http://uamarketpl...| 21| null| null| 88925| 88925| 8892588925|2024-07-22|2024-07-22 16:37:...| La Paz|2024-07-22|2024-07-22 16:37:...| La Paz| AZ|Arizona|779|Jason|Bowling| 21|Coach| http secured|

| 2019-10-01| AZ| HIOS| La Paz|http://uamarketpl...| 21| null| null| 88925| 88925| 8892588925|2024-07-22|2024-07-22 16:37:...| La Paz|2024-07-22|2024-07-22 16:37:...| La Paz| AZ|Arizona|779|Jason|Bowling| 21|Coach|http non secured|

| 2019-10-01| AZ| HIOS| La Paz|http://uamarketpl...| 21| null| null| 88925| 88925| 8892588925|2024-07-22|2024-07-22 16:37:...| La Paz|2024-07-22|2024-07-22 16:37:...| La Paz| AZ|Arizona|779|Jason|Bowling| 21|Coach| No Protocol|

| 2019-10-01| AZ| HIOS| La Paz|http://uamarketpl...| 21| null| null| 88925| 88925| 8892588925|2024-07-22|2024-07-22 16:37:...| La Paz|2024-07-22|2024-07-22 16:37:...| La Paz| AZ|Arizona|779|Jason|Bowling| 21|Coach|http non secured|

| 2019-10-01| AZ| HIOS| La Paz|http://uamarketpl...| 21| null| null| 88925| 88925| 8892588925|2024-07-22|2024-07-22 16:37:...| La Paz|2024-07-22|2024-07-22 16:37:...| La Paz| AZ|Arizona|779|Jason|Bowling| 21|Coach|http non secured|

| 2019-10-01| AZ| HIOS| La Paz|http://uamarketpl...| 21| null| null| 88925| 88925| 8892588925|2024-07-22|2024-07-22 16:37:...| La Paz|2024-07-22|2024-07-22 16:37:...| La Paz| AZ|Arizona|779|Jason|Bowling| 21|Coach| http secured|

| 2019-10-01| AZ| HIOS| La Paz|http://uamarketpl...| 21| null| null| 88925| 88925| 8892588925|2024-07-22|2024-07-22 16:37:...| La Paz|2024-07-22|2024-07-22 16:37:...| La Paz| AZ|Arizona|779|Jason|Bowling| 21|Coach|http non secured|

| 2019-10-01| AZ| HIOS| La Paz|http://uamarketpl...| 21| null| null| 88925| 88925| 8892588925|2024-07-22|2024-07-22 16:37:...| La Paz|2024-07-22|2024-07-22 16:37:...| La Paz| AZ|Arizona|779|Jason|Bowling| 21|Coach|http non secured|

| 2019-10-01| AZ| HIOS| La Paz|http://uamarketpl...| 21| null| null| 88925| 88925| 8892588925|2024-07-22|2024-07-22 16:37:...| La Paz|2024-07-22|2024-07-22 16:37:...| La Paz| AZ|Arizona|779|Jason|Bowling| 21|Coach| http secured|

| 2019-10-01| AZ| HIOS| La Paz|http://uamarketpl...| 21| null| null| 88925| 88925| 8892588925|2024-07-22|2024-07-22 16:37:...| La Paz|2024-07-22|2024-07-22 16:37:...| La Paz| AZ|Arizona|779|Jason|Bowling| 21|Coach| http secured|

| 2019-10-01| AZ| HIOS| La Paz|http://uamarketpl...| 21| null| null| 88925| 88925| 8892588925|2024-07-22|2024-07-22 16:37:...| La Paz|2024-07-22|2024-07-22 16:37:...| La Paz| AZ|Arizona|779|Jason|Bowling| 21|Coach|http non secured|

| 2019-10-01| AZ| HIOS| La Paz|http://uamarketpl...| 21| null| null| 88925| 88925| 8892588925|2024-07-22|2024-07-22 16:37:...| La Paz|2024-07-22|2024-07-22 16:37:...| La Paz| AZ|Arizona|779|Jason|Bowling| 21|Coach| http secured|

| 2019-10-01| AZ| HIOS| La Paz|http://uamarketpl...| 21| null| null| 88925| 88925| 8892588925|2024-07-22|2024-07-22 16:37:...| La Paz|2024-07-22|2024-07-22 16:37:...| La Paz| AZ|Arizona|779|Jason|Bowling| 21|Coach|http non secured|

| 2019-10-01| AZ| HIOS| La Paz|http://uamarketpl...| 21| null| null| 88925| 88925| 8892588925|2024-07-22|2024-07-22 16:37:...| La Paz|2024-07-22|2024-07-22 16:37:...| La Paz| AZ|Arizona|779|Jason|Bowling| 21|Coach|http non secured|

+------------+----+-----+-----------+--------------------+-------+--------------+----------+--------+---------+-----------------+----------+--------------------+----------------+----------+--------------------+----------------+---+-------+---+-----+-------+---+-----+----------------+

only showing top 20 rows

>>> second\_join=spark.sql("select l.\*,r.Protocol\_Used from view1 as l,view2 as r ")

>>> second\_join.createOrReplaceView("view1")

Traceback (most recent call last):

File "<stdin>", line 1, in <module>

File "/usr/local/spark/python/pyspark/sql/dataframe.py", line 1660, in \_\_getattr\_\_

"'%s' object has no attribute '%s'" % (self.\_\_class\_\_.\_\_name\_\_, name))

AttributeError: 'DataFrame' object has no attribute 'createOrReplaceView'

>>> spark.catalog.dropTempView("view`")

>>> spark.catalog.dropTempView("view1")

>>> second\_join.createOrReplaceView("view1")

Traceback (most recent call last):

File "<stdin>", line 1, in <module>

File "/usr/local/spark/python/pyspark/sql/dataframe.py", line 1660, in \_\_getattr\_\_

"'%s' object has no attribute '%s'" % (self.\_\_class\_\_.\_\_name\_\_, name))

AttributeError: 'DataFrame' object has no attribute 'createOrReplaceView'

>>> spark.catalog.dropTempView("view1")

>>> second\_join.createOrReplaceTempView("view1")

>>> spark.sql("select \* from view1 group by Protocol\_Used,srcnm,\_c3 ").show(3)

Traceback (most recent call last):

File "<stdin>", line 1, in <module>

File "/usr/local/spark/python/pyspark/sql/session.py", line 723, in sql

return DataFrame(self.\_jsparkSession.sql(sqlQuery), self.\_wrapped)

File "/usr/local/spark/python/lib/py4j-0.10.9.3-src.zip/py4j/java\_gateway.py", line 1322, in \_\_call\_\_

File "/usr/local/spark/python/pyspark/sql/utils.py", line 117, in deco

raise converted from None

pyspark.sql.utils.AnalysisException: expression 'view1.BusinessDate' is neither present in the group by, nor is it an aggregate function. Add to group by or wrap in first() (or first\_value) if you don't care which value you get.;

Aggregate [Protocol\_Used#48671, srcnm#7747, \_c3#52127], [BusinessDate#1378, stcd#7759, srcnm#7747, NetworkName#1381, NetworkURL#1382, custnum#1383, MarketCoverage#1384, RejectRows#1386, IssuerId#8312, IssuerId2#8325, issueridcomposite#8901, sysdt#9003, systs#9016, cleannetworkname#47665, curdt#48091, curts#48092, cleannetworkname#48111, \_c0#26007, \_c1#26008, \_c0#52124, \_c1#52125, \_c2#52126, \_c3#52127, \_c4#52128, Protocol\_Used#48671]

+- SubqueryAlias view1

+- View (`view1`, [BusinessDate#1378,stcd#7759,srcnm#7747,NetworkName#1381,NetworkURL#1382,custnum#1383,MarketCoverage#1384,RejectRows#1386,IssuerId#8312,IssuerId2#8325,issueridcomposite#8901,sysdt#9003,systs#9016,cleannetworkname#47665,curdt#48091,curts#48092,cleannetworkname#48111,\_c0#26007,\_c1#26008,\_c0#52124,\_c1#52125,\_c2#52126,\_c3#52127,\_c4#52128,Protocol\_Used#48671])

+- Project [BusinessDate#1378, stcd#7759, srcnm#7747, NetworkName#1381, NetworkURL#1382, custnum#1383, MarketCoverage#1384, RejectRows#1386, IssuerId#8312, IssuerId2#8325, issueridcomposite#8901, sysdt#9003, systs#9016, cleannetworkname#47665, curdt#48091, curts#48092, cleannetworkname#48111, \_c0#26007, \_c1#26008, \_c0#52124, \_c1#52125, \_c2#52126, \_c3#52127, \_c4#52128, Protocol\_Used#48671]

+- Join Inner

:- SubqueryAlias l

: +- SubqueryAlias view1

: +- View (`view1`, [BusinessDate#1378,stcd#7759,srcnm#7747,NetworkName#1381,NetworkURL#1382,custnum#1383,MarketCoverage#1384,RejectRows#1386,IssuerId#8312,IssuerId2#8325,issueridcomposite#8901,sysdt#9003,systs#9016,cleannetworkname#47665,curdt#48091,curts#48092,cleannetworkname#48111,\_c0#26007,\_c1#26008,\_c0#52124,\_c1#52125,\_c2#52126,\_c3#52127,\_c4#52128])

: +- Project [BusinessDate#1378, stcd#7759, srcnm#7747, NetworkName#1381, NetworkURL#1382, custnum#1383, MarketCoverage#1384, RejectRows#1386, IssuerId#8312, IssuerId2#8325, issueridcomposite#8901, sysdt#9003, systs#9016, cleannetworkname#47665, curdt#48091, curts#48092, cleannetworkname#48111, \_c0#26007, \_c1#26008, \_c0#52124, \_c1#52125, \_c2#52126, \_c3#52127, \_c4#52128]

: +- Join Inner, (cast(custnum#1383 as int) = \_c3#52127)

: :- SubqueryAlias l

: : +- SubqueryAlias insureview

: : +- View (`insureview`, [BusinessDate#1378,stcd#7759,srcnm#7747,NetworkName#1381,NetworkURL#1382,custnum#1383,MarketCoverage#1384,RejectRows#1386,IssuerId#8312,IssuerId2#8325,issueridcomposite#8901,sysdt#9003,systs#9016,cleannetworkname#47665,curdt#48091,curts#48092,cleannetworkname#48111,\_c0#26007,\_c1#26008])

: : +- Project [BusinessDate#1378, stcd#7759, srcnm#7747, NetworkName#1381, NetworkURL#1382, custnum#1383, MarketCoverage#1384, RejectRows#1386, IssuerId#8312, IssuerId2#8325, issueridcomposite#8901, sysdt#9003, systs#9016, cleannetworkname#47665, curdt#48091, curts#48092, cleannetworkname#48111, \_c0#26007, \_c1#26008]

: : +- Join Inner, (stcd#7759 = \_c0#26007)

: : :- SubqueryAlias l

: : : +- SubqueryAlias insureview

: : : +- View (`insureview`, [BusinessDate#1378,stcd#7759,srcnm#7747,NetworkName#1381,NetworkURL#1382,custnum#1383,MarketCoverage#1384,RejectRows#1386,IssuerId#8312,IssuerId2#8325,issueridcomposite#8901,sysdt#9003,systs#9016,cleannetworkname#47665,curdt#48091,curts#48092,cleannetworkname#48111])

: : : +- Project [BusinessDate#1378, stcd#7759, srcnm#7747, NetworkName#1381, NetworkURL#1382, custnum#1383, MarketCoverage#1384, RejectRows#1386, IssuerId#8312, IssuerId2#8325, issueridcomposite#8901, sysdt#9003, systs#9016, cleannetworkname#47665, curdt#48091, curts#48092, sql\_function(NetworkName#1381) AS cleannetworkname#48111]

: : : +- SubqueryAlias insureview

: : : +- View (`insureview`, [BusinessDate#1378,stcd#7759,srcnm#7747,NetworkName#1381,NetworkURL#1382,custnum#1383,MarketCoverage#1384,RejectRows#1386,IssuerId#8312,IssuerId2#8325,issueridcomposite#8901,sysdt#9003,systs#9016,cleannetworkname#47665,curdt#48091,curts#48092])

: : : +- Project [BusinessDate#1378, stcd#7759, srcnm#7747, NetworkName#1381, NetworkURL#1382, custnum#1383, MarketCoverage#1384, RejectRows#1386, IssuerId#8312, IssuerId2#8325, issueridcomposite#8901, sysdt#9003, systs#9016, cleannetworkname#47665, current\_date(Some(Asia/Kolkata)) AS curdt#48091, current\_timestamp() AS curts#48092]

: : : +- SubqueryAlias insureview

: : : +- View (`insureview`, [BusinessDate#1378,stcd#7759,srcnm#7747,NetworkName#1381,NetworkURL#1382,custnum#1383,MarketCoverage#1384,RejectRows#1386,IssuerId#8312,IssuerId2#8325,issueridcomposite#8901,sysdt#9003,systs#9016,cleannetworkname#47665])

: : : +- Project [BusinessDate#1378, stcd#7759, srcnm#7747, NetworkName#1381, NetworkURL#1382, custnum#1383, MarketCoverage#1384, RejectRows#1386, IssuerId#8312, IssuerId2#8325, issueridcomposite#8901, sysdt#9003, systs#9016, sql\_function(NetworkName#1381) AS cleannetworkname#47665]

: : : +- SubqueryAlias insureview

: : : +- View (`insureview`, [BusinessDate#1378,stcd#7759,srcnm#7747,NetworkName#1381,NetworkURL#1382,custnum#1383,MarketCoverage#1384,RejectRows#1386,IssuerId#8312,IssuerId2#8325,issueridcomposite#8901,sysdt#9003,systs#9016])

: : : +- Project [BusinessDate#1378, stcd#7759, srcnm#7747, NetworkName#1381, NetworkURL#1382, custnum#1383, MarketCoverage#1384, RejectRows#1386, IssuerId#8312, IssuerId2#8325, issueridcomposite#8901, sysdt#9003, current\_timestamp() AS systs#9016]

: : : +- Project [BusinessDate#1378, stcd#7759, srcnm#7747, NetworkName#1381, NetworkURL#1382, custnum#1383, MarketCoverage#1384, RejectRows#1386, IssuerId#8312, IssuerId2#8325, issueridcomposite#8901, current\_date(Some(Asia/Kolkata)) AS sysdt#9003]

: : : +- Project [BusinessDate#1378, stcd#7759, srcnm#7747, NetworkName#1381, NetworkURL#1382, custnum#1383, MarketCoverage#1384, RejectRows#1386, IssuerId#8312, IssuerId2#8325, issueridcomposite#8901]

: : : +- Project [BusinessDate#1378, stcd#7759, srcnm#7747, NetworkName#1381, NetworkURL#1382, custnum#1383, MarketCoverage#1384, DentalOnlyPlan#1385, RejectRows#1386, IssuerId#8312, IssuerId2#8325, concat(IssuerId#8312, IssuerId2#8325) AS issueridcomposite#8901]

: : : +- Project [BusinessDate#1378, stcd#7759, srcnm#7747, NetworkName#1381, NetworkURL#1382, custnum#1383, MarketCoverage#1384, DentalOnlyPlan#1385, RejectRows#1386, IssuerId#8312, IssuerId2#8325]

: : : +- Project [IssuerIDDD#8273, BusinessDate#1378, stcd#7759, srcnm#7747, NetworkName#1381, NetworkURL#1382, custnum#1383, MarketCoverage#1384, DentalOnlyPlan#1385, RejectRows#1386, IssuerId#8312, IssuerId2#8325]

: : : +- Project [IssuerIDD#8261, IssuerIDDD#8273, BusinessDate#1378, stcd#7759, srcnm#7747, NetworkName#1381, NetworkURL#1382, custnum#1383, MarketCoverage#1384, DentalOnlyPlan#1385, RejectRows#1386, IssuerId#8312, cast(IssuerIDDD#8273 as string) AS IssuerId2#8325]

: : : +- Project [IssuerIDD#8261, IssuerIDDD#8273, BusinessDate#1378, stcd#7759, srcnm#7747, NetworkName#1381, NetworkURL#1382, custnum#1383, MarketCoverage#1384, DentalOnlyPlan#1385, RejectRows#1386, cast(IssuerIDD#8261 as string) AS IssuerId#8312]

: : : +- Project [IssuerIDD#8261, IssuerId2#1377 AS IssuerIDDD#8273, BusinessDate#1378, stcd#7759, srcnm#7747, NetworkName#1381, NetworkURL#1382, custnum#1383, MarketCoverage#1384, DentalOnlyPlan#1385, RejectRows#1386]

: : : +- Project [IssuerId#1376 AS IssuerIDD#8261, IssuerId2#1377, BusinessDate#1378, stcd#7759, srcnm#7747, NetworkName#1381, NetworkURL#1382, custnum#1383, MarketCoverage#1384, DentalOnlyPlan#1385, RejectRows#1386]

: : : +- Project [IssuerId#1376, IssuerId2#1377, BusinessDate#1378, StateCode#1379 AS stcd#7759, srcnm#7747, NetworkName#1381, NetworkURL#1382, custnum#1383, MarketCoverage#1384, DentalOnlyPlan#1385, RejectRows#1386]

: : : +- Project [IssuerId#1376, IssuerId2#1377, BusinessDate#1378, StateCode#1379, SourceName#1380 AS srcnm#7747, NetworkName#1381, NetworkURL#1382, custnum#1383, MarketCoverage#1384, DentalOnlyPlan#1385, RejectRows#1386]

: : : +- Union false, false

: : : :- Relation [IssuerId#1376,IssuerId2#1377,BusinessDate#1378,StateCode#1379,SourceName#1380,NetworkName#1381,NetworkURL#1382,custnum#1383,MarketCoverage#1384,DentalOnlyPlan#1385,RejectRows#1386] csv

: : : +- Relation [IssuerId#491,IssuerId2#492,BusinessDate#493,StateCode#494,SourceName#495,NetworkName#496,NetworkURL#497,custnum#498,MarketCoverage#499,DentalOnlyPlan#500,RejectRows#501] csv

: : +- SubqueryAlias r

: : +- SubqueryAlias state\_table

: : +- View (`state\_table`, [\_c0#26007,\_c1#26008])

: : +- Project [\_c0#26007, \_c1#26008]

: : +- Filter ((isnull(\_c2#26009) AND isnull(\_c3#26010)) AND isnull(\_c4#26011))

: : +- Project [\_c0#26007, \_c1#26008, \_c2#26009, \_c3#26010, \_c4#26011]

: : +- Relation [\_c0#26007,\_c1#26008,\_c2#26009,\_c3#26010,\_c4#26011] csv

: +- SubqueryAlias r

: +- SubqueryAlias cust\_table

: +- View (`cust\_table`, [\_c0#52124,\_c1#52125,\_c2#52126,\_c3#52127,\_c4#52128])

: +- Filter (((isnotnull(\_c0#52124) AND isnotnull(\_c1#52125)) AND isnotnull(\_c2#52126)) AND (isnotnull(\_c3#52127) AND isnotnull(\_c4#52128)))

: +- Project [\_c0#52124, \_c1#52125, \_c2#52126, \_c3#52127, \_c4#52128]

: +- Relation [\_c0#52124,\_c1#52125,\_c2#52126,\_c3#52127,\_c4#52128] csv

+- SubqueryAlias r

+- SubqueryAlias view2

+- View (`view2`, [BusinessDate#57031,stcd#57060,srcnm#57061,NetworkName#57034,NetworkURL#57035,custnum#57036,MarketCoverage#57037,RejectRows#57039,IssuerId#57059,IssuerId2#57058,issueridcomposite#57057,sysdt#57056,systs#57055,cleannetworkname#57054,curdt#57052,curts#57053,cleannetworkname#57051,Protocol\_Used#48671])

+- Project [BusinessDate#57031, stcd#57060, srcnm#57061, NetworkName#57034, NetworkURL#57035, custnum#57036, MarketCoverage#57037, RejectRows#57039, IssuerId#57059, IssuerId2#57058, issueridcomposite#57057, sysdt#57056, systs#57055, cleannetworkname#57054, curdt#57052, curts#57053, cleannetworkname#57051, CASE WHEN NetworkURL#57035 LIKE https://% THEN http secured WHEN NetworkURL#57035 LIKE http://% THEN http non secured ELSE No Protocol END AS Protocol\_Used#48671]

+- SubqueryAlias insureview

+- View (`insureview`, [BusinessDate#57031,stcd#57060,srcnm#57061,NetworkName#57034,NetworkURL#57035,custnum#57036,MarketCoverage#57037,RejectRows#57039,IssuerId#57059,IssuerId2#57058,issueridcomposite#57057,sysdt#57056,systs#57055,cleannetworkname#57054,curdt#57052,curts#57053,cleannetworkname#57051])

+- Project [BusinessDate#57031, stcd#57060, srcnm#57061, NetworkName#57034, NetworkURL#57035, custnum#57036, MarketCoverage#57037, RejectRows#57039, IssuerId#57059, IssuerId2#57058, issueridcomposite#57057, sysdt#57056, systs#57055, cleannetworkname#57054, curdt#57052, curts#57053, sql\_function(NetworkName#57034) AS cleannetworkname#57051]

+- SubqueryAlias insureview

+- View (`insureview`, [BusinessDate#57031,stcd#57060,srcnm#57061,NetworkName#57034,NetworkURL#57035,custnum#57036,MarketCoverage#57037,RejectRows#57039,IssuerId#57059,IssuerId2#57058,issueridcomposite#57057,sysdt#57056,systs#57055,cleannetworkname#57054,curdt#57052,curts#57053])

+- Project [BusinessDate#57031, stcd#57060, srcnm#57061, NetworkName#57034, NetworkURL#57035, custnum#57036, MarketCoverage#57037, RejectRows#57039, IssuerId#57059, IssuerId2#57058, issueridcomposite#57057, sysdt#57056, systs#57055, cleannetworkname#57054, current\_date(Some(Asia/Kolkata)) AS curdt#57052, current\_timestamp() AS curts#57053]

+- SubqueryAlias insureview

+- View (`insureview`, [BusinessDate#57031,stcd#57060,srcnm#57061,NetworkName#57034,NetworkURL#57035,custnum#57036,MarketCoverage#57037,RejectRows#57039,IssuerId#57059,IssuerId2#57058,issueridcomposite#57057,sysdt#57056,systs#57055,cleannetworkname#57054])

+- Project [BusinessDate#57031, stcd#57060, srcnm#57061, NetworkName#57034, NetworkURL#57035, custnum#57036, MarketCoverage#57037, RejectRows#57039, IssuerId#57059, IssuerId2#57058, issueridcomposite#57057, sysdt#57056, systs#57055, sql\_function(NetworkName#57034) AS cleannetworkname#57054]

+- SubqueryAlias insureview

+- View (`insureview`, [BusinessDate#57031,stcd#57060,srcnm#57061,NetworkName#57034,NetworkURL#57035,custnum#57036,MarketCoverage#57037,RejectRows#57039,IssuerId#57059,IssuerId2#57058,issueridcomposite#57057,sysdt#57056,systs#57055])

+- Project [BusinessDate#57031, stcd#57060, srcnm#57061, NetworkName#57034, NetworkURL#57035, custnum#57036, MarketCoverage#57037, RejectRows#57039, IssuerId#57059, IssuerId2#57058, issueridcomposite#57057, sysdt#57056, current\_timestamp() AS systs#57055]

+- Project [BusinessDate#57031, stcd#57060, srcnm#57061, NetworkName#57034, NetworkURL#57035, custnum#57036, MarketCoverage#57037, RejectRows#57039, IssuerId#57059, IssuerId2#57058, issueridcomposite#57057, current\_date(Some(Asia/Kolkata)) AS sysdt#57056]

+- Project [BusinessDate#57031, stcd#57060, srcnm#57061, NetworkName#57034, NetworkURL#57035, custnum#57036, MarketCoverage#57037, RejectRows#57039, IssuerId#57059, IssuerId2#57058, issueridcomposite#57057]

+- Project [BusinessDate#57031, stcd#57060, srcnm#57061, NetworkName#57034, NetworkURL#57035, custnum#57036, MarketCoverage#57037, DentalOnlyPlan#57038, RejectRows#57039, IssuerId#57059, IssuerId2#57058, concat(IssuerId#57059, IssuerId2#57058) AS issueridcomposite#57057]

+- Project [BusinessDate#57031, stcd#57060, srcnm#57061, NetworkName#57034, NetworkURL#57035, custnum#57036, MarketCoverage#57037, DentalOnlyPlan#57038, RejectRows#57039, IssuerId#57059, IssuerId2#57058]

+- Project [IssuerIDDD#8273, BusinessDate#57031, stcd#57060, srcnm#57061, NetworkName#57034, NetworkURL#57035, custnum#57036, MarketCoverage#57037, DentalOnlyPlan#57038, RejectRows#57039, IssuerId#57059, IssuerId2#57058]

+- Project [IssuerIDD#8261, IssuerIDDD#8273, BusinessDate#57031, stcd#57060, srcnm#57061, NetworkName#57034, NetworkURL#57035, custnum#57036, MarketCoverage#57037, DentalOnlyPlan#57038, RejectRows#57039, IssuerId#57059, cast(IssuerIDDD#8273 as string) AS IssuerId2#57058]

+- Project [IssuerIDD#8261, IssuerIDDD#8273, BusinessDate#57031, stcd#57060, srcnm#57061, NetworkName#57034, NetworkURL#57035, custnum#57036, MarketCoverage#57037, DentalOnlyPlan#57038, RejectRows#57039, cast(IssuerIDD#8261 as string) AS IssuerId#57059]

+- Project [IssuerIDD#8261, IssuerId2#57030 AS IssuerIDDD#8273, BusinessDate#57031, stcd#57060, srcnm#57061, NetworkName#57034, NetworkURL#57035, custnum#57036, MarketCoverage#57037, DentalOnlyPlan#57038, RejectRows#57039]

+- Project [IssuerId#57029 AS IssuerIDD#8261, IssuerId2#57030, BusinessDate#57031, stcd#57060, srcnm#57061, NetworkName#57034, NetworkURL#57035, custnum#57036, MarketCoverage#57037, DentalOnlyPlan#57038, RejectRows#57039]

+- Project [IssuerId#57029, IssuerId2#57030, BusinessDate#57031, StateCode#57032 AS stcd#57060, srcnm#57061, NetworkName#57034, NetworkURL#57035, custnum#57036, MarketCoverage#57037, DentalOnlyPlan#57038, RejectRows#57039]

+- Project [IssuerId#57029, IssuerId2#57030, BusinessDate#57031, StateCode#57032, SourceName#57033 AS srcnm#57061, NetworkName#57034, NetworkURL#57035, custnum#57036, MarketCoverage#57037, DentalOnlyPlan#57038, RejectRows#57039]

+- Union false, false

:- Relation [IssuerId#57029,IssuerId2#57030,BusinessDate#57031,StateCode#57032,SourceName#57033,NetworkName#57034,NetworkURL#57035,custnum#57036,MarketCoverage#57037,DentalOnlyPlan#57038,RejectRows#57039] csv

+- Relation [IssuerId#57040,IssuerId2#57041,BusinessDate#57042,StateCode#57043,SourceName#57044,NetworkName#57045,NetworkURL#57046,custnum#57047,MarketCoverage#57048,DentalOnlyPlan#57049,RejectRows#57050] csv

>>>

>>> WITH CountData AS (

File "<stdin>", line 1

WITH CountData AS (

^

SyntaxError: invalid syntax

>>> SELECT

File "<stdin>", line 1

SELECT

^

IndentationError: unexpected indent

>>> statedesc,

File "<stdin>", line 1

statedesc,

^

IndentationError: unexpected indent

>>> Protocol\_Used AS protocol,

File "<stdin>", line 1

Protocol\_Used AS protocol,

^

IndentationError: unexpected indent

>>> profession,

File "<stdin>", line 1

profession,

^

IndentationError: unexpected indent

>>> AVG(age) AS avg\_age,

File "<stdin>", line 1

AVG(age) AS avg\_age,

^

IndentationError: unexpected indent

>>> COUNT(\*) AS profession\_count,

File "<stdin>", line 1

COUNT(\*) AS profession\_count,

^

IndentationError: unexpected indent

>>> ROW\_NUMBER() OVER (PARTITION BY statedesc, protocol ORDER BY COUNT(\*) DESC) AS seqno

File "<stdin>", line 1

ROW\_NUMBER() OVER (PARTITION BY statedesc, protocol ORDER BY COUNT(\*) DESC) AS seqno

^

IndentationError: unexpected indent

>>> FROM view1

File "<stdin>", line 1

FROM view1

^

IndentationError: unexpected indent

>>> GROUP BY statedesc, protocol, profession

File "<stdin>", line 1

GROUP BY statedesc, protocol, profession

^

IndentationError: unexpected indent

>>> )

File "<stdin>", line 1

)

^

SyntaxError: invalid syntax

>>>

>>> SELECT

Traceback (most recent call last):

File "<stdin>", line 1, in <module>

NameError: name 'SELECT' is not defined

>>> statedesc,

File "<stdin>", line 1

statedesc,

^

IndentationError: unexpected indent

>>> protocol,

File "<stdin>", line 1

protocol,

^

IndentationError: unexpected indent

>>> profession,

File "<stdin>", line 1

profession,

^

IndentationError: unexpected indent

>>> avg\_age,

File "<stdin>", line 1

avg\_age,

^

IndentationError: unexpected indent

>>> profession\_count

File "<stdin>", line 1

profession\_count

^

IndentationError: unexpected indent

>>> FROM CountData

File "<stdin>", line 1

FROM CountData

^

SyntaxError: invalid syntax

>>> WHERE seqno = 2

>>> from pyspark.sql.functions import col, avg, count

>>> from pyspark.sql.window import Window

>>> from pyspark.sql.functions import row\_number

>>>

>>> # Calculate average age and count, and add seqno

... window\_spec = Window.partitionBy("stcd", "Protocol\_Used").orderBy(col("profession\_count").desc())

>>> count\_df = df.groupBy("stcd", "Protocol\_Used", "profession") \

... .agg(

... avg("age").alias("avg\_age"),

... count("\*").alias("profession\_count")

... ) \

... .withColumn("seqno", row\_number().over(window\_spec))

Traceback (most recent call last):

File "<stdin>", line 1, in <module>

NameError: name 'df' is not defined

>>> count\_df = df\_renamed.groupBy("stcd", "Protocol\_Used", "profession") \

... .agg(

... avg("age").alias("avg\_age"),

... count("\*").alias("profession\_count")

... ) \

... .withColumn("seqno", row\_number().over(window\_spec))

Traceback (most recent call last):

File "<stdin>", line 4, in <module>

File "/usr/local/spark/python/pyspark/sql/group.py", line 119, in agg

\_to\_seq(self.sql\_ctx.\_sc, [c.\_jc for c in exprs[1:]]))

File "/usr/local/spark/python/lib/py4j-0.10.9.3-src.zip/py4j/java\_gateway.py", line 1322, in \_\_call\_\_

File "/usr/local/spark/python/pyspark/sql/utils.py", line 117, in deco

raise converted from None

pyspark.sql.utils.AnalysisException: cannot resolve 'Protocol\_Used' given input columns: [l.BusinessDate, City, City, l.IssuerId, l.IssuerId2, l.MarketCoverage, l.NetworkName, l.NetworkURL, l.RejectRows, r.\_c2, r.\_c3, r.\_c4, l.cleannetworkname, l.cleannetworkname, l.curdt, l.curts, l.custnum, l.issueridcomposite, l.srcnm, state\_code, state\_code, l.stcd, l.sysdt, l.systs];

'Aggregate [stcd#7759, 'Protocol\_Used, 'profession], [stcd#7759, 'Protocol\_Used, 'profession, avg('age) AS avg\_age#57112, count(1) AS profession\_count#57114L]

+- Project [BusinessDate#1378, stcd#7759, srcnm#7747, NetworkName#1381, NetworkURL#1382, custnum#1383, MarketCoverage#1384, RejectRows#1386, IssuerId#8312, IssuerId2#8325, issueridcomposite#8901, sysdt#9003, systs#9016, cleannetworkname#47665, curdt#48091, curts#48092, cleannetworkname#48111, state\_code#53694, \_c1#26008 AS City#53720, state\_code#53695, \_c1#52125 AS City#53721, \_c2#52126, \_c3#52127, \_c4#52128]

+- Project [BusinessDate#1378, stcd#7759, srcnm#7747, NetworkName#1381, NetworkURL#1382, custnum#1383, MarketCoverage#1384, RejectRows#1386, IssuerId#8312, IssuerId2#8325, issueridcomposite#8901, sysdt#9003, systs#9016, cleannetworkname#47665, curdt#48091, curts#48092, cleannetworkname#48111, \_c0#26007 AS state\_code#53694, \_c1#26008, \_c0#52124 AS state\_code#53695, \_c1#52125, \_c2#52126, \_c3#52127, \_c4#52128]

+- Project [BusinessDate#1378, stcd#7759, srcnm#7747, NetworkName#1381, NetworkURL#1382, custnum#1383, MarketCoverage#1384, RejectRows#1386, IssuerId#8312, IssuerId2#8325, issueridcomposite#8901, sysdt#9003, systs#9016, cleannetworkname#47665, curdt#48091, curts#48092, cleannetworkname#48111, \_c0#26007, \_c1#26008, \_c0#52124, \_c1#52125, \_c2#52126, \_c3#52127, \_c4#52128]

+- Join Inner, (cast(custnum#1383 as int) = \_c3#52127)

:- SubqueryAlias l

: +- SubqueryAlias insureview

: +- View (`insureview`, [BusinessDate#1378,stcd#7759,srcnm#7747,NetworkName#1381,NetworkURL#1382,custnum#1383,MarketCoverage#1384,RejectRows#1386,IssuerId#8312,IssuerId2#8325,issueridcomposite#8901,sysdt#9003,systs#9016,cleannetworkname#47665,curdt#48091,curts#48092,cleannetworkname#48111,\_c0#26007,\_c1#26008])

: +- Project [BusinessDate#1378, stcd#7759, srcnm#7747, NetworkName#1381, NetworkURL#1382, custnum#1383, MarketCoverage#1384, RejectRows#1386, IssuerId#8312, IssuerId2#8325, issueridcomposite#8901, sysdt#9003, systs#9016, cleannetworkname#47665, curdt#48091, curts#48092, cleannetworkname#48111, \_c0#26007, \_c1#26008]

: +- Join Inner, (stcd#7759 = \_c0#26007)

: :- SubqueryAlias l

: : +- SubqueryAlias insureview

: : +- View (`insureview`, [BusinessDate#1378,stcd#7759,srcnm#7747,NetworkName#1381,NetworkURL#1382,custnum#1383,MarketCoverage#1384,RejectRows#1386,IssuerId#8312,IssuerId2#8325,issueridcomposite#8901,sysdt#9003,systs#9016,cleannetworkname#47665,curdt#48091,curts#48092,cleannetworkname#48111])

: : +- Project [BusinessDate#1378, stcd#7759, srcnm#7747, NetworkName#1381, NetworkURL#1382, custnum#1383, MarketCoverage#1384, RejectRows#1386, IssuerId#8312, IssuerId2#8325, issueridcomposite#8901, sysdt#9003, systs#9016, cleannetworkname#47665, curdt#48091, curts#48092, sql\_function(NetworkName#1381) AS cleannetworkname#48111]

: : +- SubqueryAlias insureview

: : +- View (`insureview`, [BusinessDate#1378,stcd#7759,srcnm#7747,NetworkName#1381,NetworkURL#1382,custnum#1383,MarketCoverage#1384,RejectRows#1386,IssuerId#8312,IssuerId2#8325,issueridcomposite#8901,sysdt#9003,systs#9016,cleannetworkname#47665,curdt#48091,curts#48092])

: : +- Project [BusinessDate#1378, stcd#7759, srcnm#7747, NetworkName#1381, NetworkURL#1382, custnum#1383, MarketCoverage#1384, RejectRows#1386, IssuerId#8312, IssuerId2#8325, issueridcomposite#8901, sysdt#9003, systs#9016, cleannetworkname#47665, current\_date(Some(Asia/Kolkata)) AS curdt#48091, current\_timestamp() AS curts#48092]

: : +- SubqueryAlias insureview

: : +- View (`insureview`, [BusinessDate#1378,stcd#7759,srcnm#7747,NetworkName#1381,NetworkURL#1382,custnum#1383,MarketCoverage#1384,RejectRows#1386,IssuerId#8312,IssuerId2#8325,issueridcomposite#8901,sysdt#9003,systs#9016,cleannetworkname#47665])

: : +- Project [BusinessDate#1378, stcd#7759, srcnm#7747, NetworkName#1381, NetworkURL#1382, custnum#1383, MarketCoverage#1384, RejectRows#1386, IssuerId#8312, IssuerId2#8325, issueridcomposite#8901, sysdt#9003, systs#9016, sql\_function(NetworkName#1381) AS cleannetworkname#47665]

: : +- SubqueryAlias insureview

: : +- View (`insureview`, [BusinessDate#1378,stcd#7759,srcnm#7747,NetworkName#1381,NetworkURL#1382,custnum#1383,MarketCoverage#1384,RejectRows#1386,IssuerId#8312,IssuerId2#8325,issueridcomposite#8901,sysdt#9003,systs#9016])

: : +- Project [BusinessDate#1378, stcd#7759, srcnm#7747, NetworkName#1381, NetworkURL#1382, custnum#1383, MarketCoverage#1384, RejectRows#1386, IssuerId#8312, IssuerId2#8325, issueridcomposite#8901, sysdt#9003, current\_timestamp() AS systs#9016]

: : +- Project [BusinessDate#1378, stcd#7759, srcnm#7747, NetworkName#1381, NetworkURL#1382, custnum#1383, MarketCoverage#1384, RejectRows#1386, IssuerId#8312, IssuerId2#8325, issueridcomposite#8901, current\_date(Some(Asia/Kolkata)) AS sysdt#9003]

: : +- Project [BusinessDate#1378, stcd#7759, srcnm#7747, NetworkName#1381, NetworkURL#1382, custnum#1383, MarketCoverage#1384, RejectRows#1386, IssuerId#8312, IssuerId2#8325, issueridcomposite#8901]

: : +- Project [BusinessDate#1378, stcd#7759, srcnm#7747, NetworkName#1381, NetworkURL#1382, custnum#1383, MarketCoverage#1384, DentalOnlyPlan#1385, RejectRows#1386, IssuerId#8312, IssuerId2#8325, concat(IssuerId#8312, IssuerId2#8325) AS issueridcomposite#8901]

: : +- Project [BusinessDate#1378, stcd#7759, srcnm#7747, NetworkName#1381, NetworkURL#1382, custnum#1383, MarketCoverage#1384, DentalOnlyPlan#1385, RejectRows#1386, IssuerId#8312, IssuerId2#8325]

: : +- Project [IssuerIDDD#8273, BusinessDate#1378, stcd#7759, srcnm#7747, NetworkName#1381, NetworkURL#1382, custnum#1383, MarketCoverage#1384, DentalOnlyPlan#1385, RejectRows#1386, IssuerId#8312, IssuerId2#8325]

: : +- Project [IssuerIDD#8261, IssuerIDDD#8273, BusinessDate#1378, stcd#7759, srcnm#7747, NetworkName#1381, NetworkURL#1382, custnum#1383, MarketCoverage#1384, DentalOnlyPlan#1385, RejectRows#1386, IssuerId#8312, cast(IssuerIDDD#8273 as string) AS IssuerId2#8325]

: : +- Project [IssuerIDD#8261, IssuerIDDD#8273, BusinessDate#1378, stcd#7759, srcnm#7747, NetworkName#1381, NetworkURL#1382, custnum#1383, MarketCoverage#1384, DentalOnlyPlan#1385, RejectRows#1386, cast(IssuerIDD#8261 as string) AS IssuerId#8312]

: : +- Project [IssuerIDD#8261, IssuerId2#1377 AS IssuerIDDD#8273, BusinessDate#1378, stcd#7759, srcnm#7747, NetworkName#1381, NetworkURL#1382, custnum#1383, MarketCoverage#1384, DentalOnlyPlan#1385, RejectRows#1386]

: : +- Project [IssuerId#1376 AS IssuerIDD#8261, IssuerId2#1377, BusinessDate#1378, stcd#7759, srcnm#7747, NetworkName#1381, NetworkURL#1382, custnum#1383, MarketCoverage#1384, DentalOnlyPlan#1385, RejectRows#1386]

: : +- Project [IssuerId#1376, IssuerId2#1377, BusinessDate#1378, StateCode#1379 AS stcd#7759, srcnm#7747, NetworkName#1381, NetworkURL#1382, custnum#1383, MarketCoverage#1384, DentalOnlyPlan#1385, RejectRows#1386]

: : +- Project [IssuerId#1376, IssuerId2#1377, BusinessDate#1378, StateCode#1379, SourceName#1380 AS srcnm#7747, NetworkName#1381, NetworkURL#1382, custnum#1383, MarketCoverage#1384, DentalOnlyPlan#1385, RejectRows#1386]

: : +- Union false, false

: : :- Relation [IssuerId#1376,IssuerId2#1377,BusinessDate#1378,StateCode#1379,SourceName#1380,NetworkName#1381,NetworkURL#1382,custnum#1383,MarketCoverage#1384,DentalOnlyPlan#1385,RejectRows#1386] csv

: : +- Relation [IssuerId#491,IssuerId2#492,BusinessDate#493,StateCode#494,SourceName#495,NetworkName#496,NetworkURL#497,custnum#498,MarketCoverage#499,DentalOnlyPlan#500,RejectRows#501] csv

: +- SubqueryAlias r

: +- SubqueryAlias state\_table

: +- View (`state\_table`, [\_c0#26007,\_c1#26008])

: +- Project [\_c0#26007, \_c1#26008]

: +- Filter ((isnull(\_c2#26009) AND isnull(\_c3#26010)) AND isnull(\_c4#26011))

: +- Project [\_c0#26007, \_c1#26008, \_c2#26009, \_c3#26010, \_c4#26011]

: +- Relation [\_c0#26007,\_c1#26008,\_c2#26009,\_c3#26010,\_c4#26011] csv

+- SubqueryAlias r

+- SubqueryAlias cust\_table

+- View (`cust\_table`, [\_c0#52124,\_c1#52125,\_c2#52126,\_c3#52127,\_c4#52128])

+- Filter (((isnotnull(\_c0#52124) AND isnotnull(\_c1#52125)) AND isnotnull(\_c2#52126)) AND (isnotnull(\_c3#52127) AND isnotnull(\_c4#52128)))

+- Project [\_c0#52124, \_c1#52125, \_c2#52126, \_c3#52127, \_c4#52128]

+- Relation [\_c0#52124,\_c1#52125,\_c2#52126,\_c3#52127,\_c4#52128] csv

>>>

>>> from pyspark.sql.functions import col, avg, count

>>> from pyspark.sql.window import Window

>>> from pyspark.sql.functions import row\_number

>>>

>>> # Calculate average age and count, and add seqno

... window\_spec = Window.partitionBy("stcd", "Protocol\_Used").orderBy(col("profession\_count").desc())

>>>

>>> count\_df = df.groupBy("stcd", "Protocol\_Used", "profession") \

... .agg(

... avg("\_c1").alias("avg\_age"),

... count("\*").alias("profession\_count")

... ) \

... .withColumn("seqno", row\_number().over(window\_spec))

Traceback (most recent call last):

File "<stdin>", line 1, in <module>

NameError: name 'df' is not defined

>>> count\_df = df\_renamed.groupBy("stcd", "Protocol\_Used", "profession") \

... .agg(

... avg("\_c1").alias("avg\_age"),

... count("\*").alias("profession\_count")

... ) \

... .withColumn("seqno", row\_number().over(window\_spec))

Traceback (most recent call last):

File "<stdin>", line 4, in <module>

File "/usr/local/spark/python/pyspark/sql/group.py", line 119, in agg

\_to\_seq(self.sql\_ctx.\_sc, [c.\_jc for c in exprs[1:]]))

File "/usr/local/spark/python/lib/py4j-0.10.9.3-src.zip/py4j/java\_gateway.py", line 1322, in \_\_call\_\_

File "/usr/local/spark/python/pyspark/sql/utils.py", line 117, in deco

raise converted from None

pyspark.sql.utils.AnalysisException: cannot resolve 'Protocol\_Used' given input columns: [l.BusinessDate, City, City, l.IssuerId, l.IssuerId2, l.MarketCoverage, l.NetworkName, l.NetworkURL, l.RejectRows, r.\_c2, r.\_c3, r.\_c4, l.cleannetworkname, l.cleannetworkname, l.curdt, l.curts, l.custnum, l.issueridcomposite, l.srcnm, state\_code, state\_code, l.stcd, l.sysdt, l.systs];

'Aggregate [stcd#7759, 'Protocol\_Used, 'profession], [stcd#7759, 'Protocol\_Used, 'profession, avg('\_c1) AS avg\_age#57140, count(1) AS profession\_count#57142L]

+- Project [BusinessDate#1378, stcd#7759, srcnm#7747, NetworkName#1381, NetworkURL#1382, custnum#1383, MarketCoverage#1384, RejectRows#1386, IssuerId#8312, IssuerId2#8325, issueridcomposite#8901, sysdt#9003, systs#9016, cleannetworkname#47665, curdt#48091, curts#48092, cleannetworkname#48111, state\_code#53694, \_c1#26008 AS City#53720, state\_code#53695, \_c1#52125 AS City#53721, \_c2#52126, \_c3#52127, \_c4#52128]

+- Project [BusinessDate#1378, stcd#7759, srcnm#7747, NetworkName#1381, NetworkURL#1382, custnum#1383, MarketCoverage#1384, RejectRows#1386, IssuerId#8312, IssuerId2#8325, issueridcomposite#8901, sysdt#9003, systs#9016, cleannetworkname#47665, curdt#48091, curts#48092, cleannetworkname#48111, \_c0#26007 AS state\_code#53694, \_c1#26008, \_c0#52124 AS state\_code#53695, \_c1#52125, \_c2#52126, \_c3#52127, \_c4#52128]

+- Project [BusinessDate#1378, stcd#7759, srcnm#7747, NetworkName#1381, NetworkURL#1382, custnum#1383, MarketCoverage#1384, RejectRows#1386, IssuerId#8312, IssuerId2#8325, issueridcomposite#8901, sysdt#9003, systs#9016, cleannetworkname#47665, curdt#48091, curts#48092, cleannetworkname#48111, \_c0#26007, \_c1#26008, \_c0#52124, \_c1#52125, \_c2#52126, \_c3#52127, \_c4#52128]

+- Join Inner, (cast(custnum#1383 as int) = \_c3#52127)

:- SubqueryAlias l

: +- SubqueryAlias insureview

: +- View (`insureview`, [BusinessDate#1378,stcd#7759,srcnm#7747,NetworkName#1381,NetworkURL#1382,custnum#1383,MarketCoverage#1384,RejectRows#1386,IssuerId#8312,IssuerId2#8325,issueridcomposite#8901,sysdt#9003,systs#9016,cleannetworkname#47665,curdt#48091,curts#48092,cleannetworkname#48111,\_c0#26007,\_c1#26008])

: +- Project [BusinessDate#1378, stcd#7759, srcnm#7747, NetworkName#1381, NetworkURL#1382, custnum#1383, MarketCoverage#1384, RejectRows#1386, IssuerId#8312, IssuerId2#8325, issueridcomposite#8901, sysdt#9003, systs#9016, cleannetworkname#47665, curdt#48091, curts#48092, cleannetworkname#48111, \_c0#26007, \_c1#26008]

: +- Join Inner, (stcd#7759 = \_c0#26007)

: :- SubqueryAlias l

: : +- SubqueryAlias insureview

: : +- View (`insureview`, [BusinessDate#1378,stcd#7759,srcnm#7747,NetworkName#1381,NetworkURL#1382,custnum#1383,MarketCoverage#1384,RejectRows#1386,IssuerId#8312,IssuerId2#8325,issueridcomposite#8901,sysdt#9003,systs#9016,cleannetworkname#47665,curdt#48091,curts#48092,cleannetworkname#48111])

: : +- Project [BusinessDate#1378, stcd#7759, srcnm#7747, NetworkName#1381, NetworkURL#1382, custnum#1383, MarketCoverage#1384, RejectRows#1386, IssuerId#8312, IssuerId2#8325, issueridcomposite#8901, sysdt#9003, systs#9016, cleannetworkname#47665, curdt#48091, curts#48092, sql\_function(NetworkName#1381) AS cleannetworkname#48111]

: : +- SubqueryAlias insureview

: : +- View (`insureview`, [BusinessDate#1378,stcd#7759,srcnm#7747,NetworkName#1381,NetworkURL#1382,custnum#1383,MarketCoverage#1384,RejectRows#1386,IssuerId#8312,IssuerId2#8325,issueridcomposite#8901,sysdt#9003,systs#9016,cleannetworkname#47665,curdt#48091,curts#48092])

: : +- Project [BusinessDate#1378, stcd#7759, srcnm#7747, NetworkName#1381, NetworkURL#1382, custnum#1383, MarketCoverage#1384, RejectRows#1386, IssuerId#8312, IssuerId2#8325, issueridcomposite#8901, sysdt#9003, systs#9016, cleannetworkname#47665, current\_date(Some(Asia/Kolkata)) AS curdt#48091, current\_timestamp() AS curts#48092]

: : +- SubqueryAlias insureview

: : +- View (`insureview`, [BusinessDate#1378,stcd#7759,srcnm#7747,NetworkName#1381,NetworkURL#1382,custnum#1383,MarketCoverage#1384,RejectRows#1386,IssuerId#8312,IssuerId2#8325,issueridcomposite#8901,sysdt#9003,systs#9016,cleannetworkname#47665])

: : +- Project [BusinessDate#1378, stcd#7759, srcnm#7747, NetworkName#1381, NetworkURL#1382, custnum#1383, MarketCoverage#1384, RejectRows#1386, IssuerId#8312, IssuerId2#8325, issueridcomposite#8901, sysdt#9003, systs#9016, sql\_function(NetworkName#1381) AS cleannetworkname#47665]

: : +- SubqueryAlias insureview

: : +- View (`insureview`, [BusinessDate#1378,stcd#7759,srcnm#7747,NetworkName#1381,NetworkURL#1382,custnum#1383,MarketCoverage#1384,RejectRows#1386,IssuerId#8312,IssuerId2#8325,issueridcomposite#8901,sysdt#9003,systs#9016])

: : +- Project [BusinessDate#1378, stcd#7759, srcnm#7747, NetworkName#1381, NetworkURL#1382, custnum#1383, MarketCoverage#1384, RejectRows#1386, IssuerId#8312, IssuerId2#8325, issueridcomposite#8901, sysdt#9003, current\_timestamp() AS systs#9016]

: : +- Project [BusinessDate#1378, stcd#7759, srcnm#7747, NetworkName#1381, NetworkURL#1382, custnum#1383, MarketCoverage#1384, RejectRows#1386, IssuerId#8312, IssuerId2#8325, issueridcomposite#8901, current\_date(Some(Asia/Kolkata)) AS sysdt#9003]

: : +- Project [BusinessDate#1378, stcd#7759, srcnm#7747, NetworkName#1381, NetworkURL#1382, custnum#1383, MarketCoverage#1384, RejectRows#1386, IssuerId#8312, IssuerId2#8325, issueridcomposite#8901]

: : +- Project [BusinessDate#1378, stcd#7759, srcnm#7747, NetworkName#1381, NetworkURL#1382, custnum#1383, MarketCoverage#1384, DentalOnlyPlan#1385, RejectRows#1386, IssuerId#8312, IssuerId2#8325, concat(IssuerId#8312, IssuerId2#8325) AS issueridcomposite#8901]

: : +- Project [BusinessDate#1378, stcd#7759, srcnm#7747, NetworkName#1381, NetworkURL#1382, custnum#1383, MarketCoverage#1384, DentalOnlyPlan#1385, RejectRows#1386, IssuerId#8312, IssuerId2#8325]

: : +- Project [IssuerIDDD#8273, BusinessDate#1378, stcd#7759, srcnm#7747, NetworkName#1381, NetworkURL#1382, custnum#1383, MarketCoverage#1384, DentalOnlyPlan#1385, RejectRows#1386, IssuerId#8312, IssuerId2#8325]

: : +- Project [IssuerIDD#8261, IssuerIDDD#8273, BusinessDate#1378, stcd#7759, srcnm#7747, NetworkName#1381, NetworkURL#1382, custnum#1383, MarketCoverage#1384, DentalOnlyPlan#1385, RejectRows#1386, IssuerId#8312, cast(IssuerIDDD#8273 as string) AS IssuerId2#8325]

: : +- Project [IssuerIDD#8261, IssuerIDDD#8273, BusinessDate#1378, stcd#7759, srcnm#7747, NetworkName#1381, NetworkURL#1382, custnum#1383, MarketCoverage#1384, DentalOnlyPlan#1385, RejectRows#1386, cast(IssuerIDD#8261 as string) AS IssuerId#8312]

: : +- Project [IssuerIDD#8261, IssuerId2#1377 AS IssuerIDDD#8273, BusinessDate#1378, stcd#7759, srcnm#7747, NetworkName#1381, NetworkURL#1382, custnum#1383, MarketCoverage#1384, DentalOnlyPlan#1385, RejectRows#1386]

: : +- Project [IssuerId#1376 AS IssuerIDD#8261, IssuerId2#1377, BusinessDate#1378, stcd#7759, srcnm#7747, NetworkName#1381, NetworkURL#1382, custnum#1383, MarketCoverage#1384, DentalOnlyPlan#1385, RejectRows#1386]

: : +- Project [IssuerId#1376, IssuerId2#1377, BusinessDate#1378, StateCode#1379 AS stcd#7759, srcnm#7747, NetworkName#1381, NetworkURL#1382, custnum#1383, MarketCoverage#1384, DentalOnlyPlan#1385, RejectRows#1386]

: : +- Project [IssuerId#1376, IssuerId2#1377, BusinessDate#1378, StateCode#1379, SourceName#1380 AS srcnm#7747, NetworkName#1381, NetworkURL#1382, custnum#1383, MarketCoverage#1384, DentalOnlyPlan#1385, RejectRows#1386]

: : +- Union false, false

: : :- Relation [IssuerId#1376,IssuerId2#1377,BusinessDate#1378,StateCode#1379,SourceName#1380,NetworkName#1381,NetworkURL#1382,custnum#1383,MarketCoverage#1384,DentalOnlyPlan#1385,RejectRows#1386] csv

: : +- Relation [IssuerId#491,IssuerId2#492,BusinessDate#493,StateCode#494,SourceName#495,NetworkName#496,NetworkURL#497,custnum#498,MarketCoverage#499,DentalOnlyPlan#500,RejectRows#501] csv

: +- SubqueryAlias r

: +- SubqueryAlias state\_table

: +- View (`state\_table`, [\_c0#26007,\_c1#26008])

: +- Project [\_c0#26007, \_c1#26008]

: +- Filter ((isnull(\_c2#26009) AND isnull(\_c3#26010)) AND isnull(\_c4#26011))

: +- Project [\_c0#26007, \_c1#26008, \_c2#26009, \_c3#26010, \_c4#26011]

: +- Relation [\_c0#26007,\_c1#26008,\_c2#26009,\_c3#26010,\_c4#26011] csv

+- SubqueryAlias r

+- SubqueryAlias cust\_table

+- View (`cust\_table`, [\_c0#52124,\_c1#52125,\_c2#52126,\_c3#52127,\_c4#52128])

+- Filter (((isnotnull(\_c0#52124) AND isnotnull(\_c1#52125)) AND isnotnull(\_c2#52126)) AND (isnotnull(\_c3#52127) AND isnotnull(\_c4#52128)))

+- Project [\_c0#52124, \_c1#52125, \_c2#52126, \_c3#52127, \_c4#52128]

+- Relation [\_c0#52124,\_c1#52125,\_c2#52126,\_c3#52127,\_c4#52128] csv

>>> second\_join.show(1)

+------------+----+-----+-----------+--------------------+-------+--------------+----------+--------+---------+-----------------+----------+--------------------+----------------+----------+--------------------+----------------+---+-------+---+-----+-------+---+-----+-------------+

|BusinessDate|stcd|srcnm|NetworkName| NetworkURL|custnum|MarketCoverage|RejectRows|IssuerId|IssuerId2|issueridcomposite| sysdt| systs|cleannetworkname| curdt| curts|cleannetworkname|\_c0| \_c1|\_c0| \_c1| \_c2|\_c3| \_c4|Protocol\_Used|

+------------+----+-----+-----------+--------------------+-------+--------------+----------+--------+---------+-----------------+----------+--------------------+----------------+----------+--------------------+----------------+---+-------+---+-----+-------+---+-----+-------------+

| 2019-10-01| AZ| HIOS| La Paz|http://uamarketpl...| 21| null| null| 88925| 88925| 8892588925|2024-07-22|2024-07-22 16:54:...| La Paz|2024-07-22|2024-07-22 16:54:...| La Paz| AZ|Arizona|779|Jason|Bowling| 21|Coach| http secured|

+------------+----+-----+-----------+--------------------+-------+--------------+----------+--------+---------+-----------------+----------+--------------------+----------------+----------+--------------------+----------------+---+-------+---+-----+-------+---+-----+-------------+

only showing top 1 row

>>> from pyspark.sql.functions import col, avg, count

>>> from pyspark.sql.window import Window

>>> from pyspark.sql.functions import row\_number

>>>

>>> # Calculate average age and count, and add seqno

... window\_spec = Window.partitionBy("stcd", "Protocol\_Used").orderBy(col("profession\_count").desc())

>>>

>>> count\_df = second\_join.groupBy("stcd", "Protocol\_Used", "profession") \

... .agg(

... avg("age").alias("avg\_age"),

... count("\*").alias("profession\_count")

... ) \

... .withColumn("seqno", row\_number().over(window\_spec))

Traceback (most recent call last):

File "<stdin>", line 4, in <module>

File "/usr/local/spark/python/pyspark/sql/group.py", line 119, in agg

\_to\_seq(self.sql\_ctx.\_sc, [c.\_jc for c in exprs[1:]]))

File "/usr/local/spark/python/lib/py4j-0.10.9.3-src.zip/py4j/java\_gateway.py", line 1322, in \_\_call\_\_

File "/usr/local/spark/python/pyspark/sql/utils.py", line 117, in deco

raise converted from None

pyspark.sql.utils.AnalysisException: cannot resolve 'profession' given input columns: [l.BusinessDate, l.IssuerId, l.IssuerId2, l.MarketCoverage, l.NetworkName, l.NetworkURL, r.Protocol\_Used, l.RejectRows, l.\_c0, l.\_c0, l.\_c1, l.\_c1, l.\_c2, l.\_c3, l.\_c4, l.cleannetworkname, l.cleannetworkname, l.curdt, l.curts, l.custnum, l.issueridcomposite, l.srcnm, l.stcd, l.sysdt, l.systs];

'Aggregate [stcd#7759, Protocol\_Used#48671, 'profession], [stcd#7759, Protocol\_Used#48671, 'profession, avg('age) AS avg\_age#58285, count(1) AS profession\_count#58287L]

+- Project [BusinessDate#1378, stcd#7759, srcnm#7747, NetworkName#1381, NetworkURL#1382, custnum#1383, MarketCoverage#1384, RejectRows#1386, IssuerId#8312, IssuerId2#8325, issueridcomposite#8901, sysdt#9003, systs#9016, cleannetworkname#47665, curdt#48091, curts#48092, cleannetworkname#48111, \_c0#26007, \_c1#26008, \_c0#52124, \_c1#52125, \_c2#52126, \_c3#52127, \_c4#52128, Protocol\_Used#48671]

+- Join Inner

:- SubqueryAlias l

: +- SubqueryAlias view1

: +- View (`view1`, [BusinessDate#1378,stcd#7759,srcnm#7747,NetworkName#1381,NetworkURL#1382,custnum#1383,MarketCoverage#1384,RejectRows#1386,IssuerId#8312,IssuerId2#8325,issueridcomposite#8901,sysdt#9003,systs#9016,cleannetworkname#47665,curdt#48091,curts#48092,cleannetworkname#48111,\_c0#26007,\_c1#26008,\_c0#52124,\_c1#52125,\_c2#52126,\_c3#52127,\_c4#52128])

: +- Project [BusinessDate#1378, stcd#7759, srcnm#7747, NetworkName#1381, NetworkURL#1382, custnum#1383, MarketCoverage#1384, RejectRows#1386, IssuerId#8312, IssuerId2#8325, issueridcomposite#8901, sysdt#9003, systs#9016, cleannetworkname#47665, curdt#48091, curts#48092, cleannetworkname#48111, \_c0#26007, \_c1#26008, \_c0#52124, \_c1#52125, \_c2#52126, \_c3#52127, \_c4#52128]

: +- Join Inner, (cast(custnum#1383 as int) = \_c3#52127)

: :- SubqueryAlias l

: : +- SubqueryAlias insureview

: : +- View (`insureview`, [BusinessDate#1378,stcd#7759,srcnm#7747,NetworkName#1381,NetworkURL#1382,custnum#1383,MarketCoverage#1384,RejectRows#1386,IssuerId#8312,IssuerId2#8325,issueridcomposite#8901,sysdt#9003,systs#9016,cleannetworkname#47665,curdt#48091,curts#48092,cleannetworkname#48111,\_c0#26007,\_c1#26008])

: : +- Project [BusinessDate#1378, stcd#7759, srcnm#7747, NetworkName#1381, NetworkURL#1382, custnum#1383, MarketCoverage#1384, RejectRows#1386, IssuerId#8312, IssuerId2#8325, issueridcomposite#8901, sysdt#9003, systs#9016, cleannetworkname#47665, curdt#48091, curts#48092, cleannetworkname#48111, \_c0#26007, \_c1#26008]

: : +- Join Inner, (stcd#7759 = \_c0#26007)

: : :- SubqueryAlias l

: : : +- SubqueryAlias insureview

: : : +- View (`insureview`, [BusinessDate#1378,stcd#7759,srcnm#7747,NetworkName#1381,NetworkURL#1382,custnum#1383,MarketCoverage#1384,RejectRows#1386,IssuerId#8312,IssuerId2#8325,issueridcomposite#8901,sysdt#9003,systs#9016,cleannetworkname#47665,curdt#48091,curts#48092,cleannetworkname#48111])

: : : +- Project [BusinessDate#1378, stcd#7759, srcnm#7747, NetworkName#1381, NetworkURL#1382, custnum#1383, MarketCoverage#1384, RejectRows#1386, IssuerId#8312, IssuerId2#8325, issueridcomposite#8901, sysdt#9003, systs#9016, cleannetworkname#47665, curdt#48091, curts#48092, sql\_function(NetworkName#1381) AS cleannetworkname#48111]

: : : +- SubqueryAlias insureview

: : : +- View (`insureview`, [BusinessDate#1378,stcd#7759,srcnm#7747,NetworkName#1381,NetworkURL#1382,custnum#1383,MarketCoverage#1384,RejectRows#1386,IssuerId#8312,IssuerId2#8325,issueridcomposite#8901,sysdt#9003,systs#9016,cleannetworkname#47665,curdt#48091,curts#48092])

: : : +- Project [BusinessDate#1378, stcd#7759, srcnm#7747, NetworkName#1381, NetworkURL#1382, custnum#1383, MarketCoverage#1384, RejectRows#1386, IssuerId#8312, IssuerId2#8325, issueridcomposite#8901, sysdt#9003, systs#9016, cleannetworkname#47665, current\_date(Some(Asia/Kolkata)) AS curdt#48091, current\_timestamp() AS curts#48092]

: : : +- SubqueryAlias insureview

: : : +- View (`insureview`, [BusinessDate#1378,stcd#7759,srcnm#7747,NetworkName#1381,NetworkURL#1382,custnum#1383,MarketCoverage#1384,RejectRows#1386,IssuerId#8312,IssuerId2#8325,issueridcomposite#8901,sysdt#9003,systs#9016,cleannetworkname#47665])

: : : +- Project [BusinessDate#1378, stcd#7759, srcnm#7747, NetworkName#1381, NetworkURL#1382, custnum#1383, MarketCoverage#1384, RejectRows#1386, IssuerId#8312, IssuerId2#8325, issueridcomposite#8901, sysdt#9003, systs#9016, sql\_function(NetworkName#1381) AS cleannetworkname#47665]

: : : +- SubqueryAlias insureview

: : : +- View (`insureview`, [BusinessDate#1378,stcd#7759,srcnm#7747,NetworkName#1381,NetworkURL#1382,custnum#1383,MarketCoverage#1384,RejectRows#1386,IssuerId#8312,IssuerId2#8325,issueridcomposite#8901,sysdt#9003,systs#9016])

: : : +- Project [BusinessDate#1378, stcd#7759, srcnm#7747, NetworkName#1381, NetworkURL#1382, custnum#1383, MarketCoverage#1384, RejectRows#1386, IssuerId#8312, IssuerId2#8325, issueridcomposite#8901, sysdt#9003, current\_timestamp() AS systs#9016]

: : : +- Project [BusinessDate#1378, stcd#7759, srcnm#7747, NetworkName#1381, NetworkURL#1382, custnum#1383, MarketCoverage#1384, RejectRows#1386, IssuerId#8312, IssuerId2#8325, issueridcomposite#8901, current\_date(Some(Asia/Kolkata)) AS sysdt#9003]

: : : +- Project [BusinessDate#1378, stcd#7759, srcnm#7747, NetworkName#1381, NetworkURL#1382, custnum#1383, MarketCoverage#1384, RejectRows#1386, IssuerId#8312, IssuerId2#8325, issueridcomposite#8901]

: : : +- Project [BusinessDate#1378, stcd#7759, srcnm#7747, NetworkName#1381, NetworkURL#1382, custnum#1383, MarketCoverage#1384, DentalOnlyPlan#1385, RejectRows#1386, IssuerId#8312, IssuerId2#8325, concat(IssuerId#8312, IssuerId2#8325) AS issueridcomposite#8901]

: : : +- Project [BusinessDate#1378, stcd#7759, srcnm#7747, NetworkName#1381, NetworkURL#1382, custnum#1383, MarketCoverage#1384, DentalOnlyPlan#1385, RejectRows#1386, IssuerId#8312, IssuerId2#8325]

: : : +- Project [IssuerIDDD#8273, BusinessDate#1378, stcd#7759, srcnm#7747, NetworkName#1381, NetworkURL#1382, custnum#1383, MarketCoverage#1384, DentalOnlyPlan#1385, RejectRows#1386, IssuerId#8312, IssuerId2#8325]

: : : +- Project [IssuerIDD#8261, IssuerIDDD#8273, BusinessDate#1378, stcd#7759, srcnm#7747, NetworkName#1381, NetworkURL#1382, custnum#1383, MarketCoverage#1384, DentalOnlyPlan#1385, RejectRows#1386, IssuerId#8312, cast(IssuerIDDD#8273 as string) AS IssuerId2#8325]

: : : +- Project [IssuerIDD#8261, IssuerIDDD#8273, BusinessDate#1378, stcd#7759, srcnm#7747, NetworkName#1381, NetworkURL#1382, custnum#1383, MarketCoverage#1384, DentalOnlyPlan#1385, RejectRows#1386, cast(IssuerIDD#8261 as string) AS IssuerId#8312]

: : : +- Project [IssuerIDD#8261, IssuerId2#1377 AS IssuerIDDD#8273, BusinessDate#1378, stcd#7759, srcnm#7747, NetworkName#1381, NetworkURL#1382, custnum#1383, MarketCoverage#1384, DentalOnlyPlan#1385, RejectRows#1386]

: : : +- Project [IssuerId#1376 AS IssuerIDD#8261, IssuerId2#1377, BusinessDate#1378, stcd#7759, srcnm#7747, NetworkName#1381, NetworkURL#1382, custnum#1383, MarketCoverage#1384, DentalOnlyPlan#1385, RejectRows#1386]

: : : +- Project [IssuerId#1376, IssuerId2#1377, BusinessDate#1378, StateCode#1379 AS stcd#7759, srcnm#7747, NetworkName#1381, NetworkURL#1382, custnum#1383, MarketCoverage#1384, DentalOnlyPlan#1385, RejectRows#1386]

: : : +- Project [IssuerId#1376, IssuerId2#1377, BusinessDate#1378, StateCode#1379, SourceName#1380 AS srcnm#7747, NetworkName#1381, NetworkURL#1382, custnum#1383, MarketCoverage#1384, DentalOnlyPlan#1385, RejectRows#1386]

: : : +- Union false, false

: : : :- Relation [IssuerId#1376,IssuerId2#1377,BusinessDate#1378,StateCode#1379,SourceName#1380,NetworkName#1381,NetworkURL#1382,custnum#1383,MarketCoverage#1384,DentalOnlyPlan#1385,RejectRows#1386] csv

: : : +- Relation [IssuerId#491,IssuerId2#492,BusinessDate#493,StateCode#494,SourceName#495,NetworkName#496,NetworkURL#497,custnum#498,MarketCoverage#499,DentalOnlyPlan#500,RejectRows#501] csv

: : +- SubqueryAlias r

: : +- SubqueryAlias state\_table

: : +- View (`state\_table`, [\_c0#26007,\_c1#26008])

: : +- Project [\_c0#26007, \_c1#26008]

: : +- Filter ((isnull(\_c2#26009) AND isnull(\_c3#26010)) AND isnull(\_c4#26011))

: : +- Project [\_c0#26007, \_c1#26008, \_c2#26009, \_c3#26010, \_c4#26011]

: : +- Relation [\_c0#26007,\_c1#26008,\_c2#26009,\_c3#26010,\_c4#26011] csv

: +- SubqueryAlias r

: +- SubqueryAlias cust\_table

: +- View (`cust\_table`, [\_c0#52124,\_c1#52125,\_c2#52126,\_c3#52127,\_c4#52128])

: +- Filter (((isnotnull(\_c0#52124) AND isnotnull(\_c1#52125)) AND isnotnull(\_c2#52126)) AND (isnotnull(\_c3#52127) AND isnotnull(\_c4#52128)))

: +- Project [\_c0#52124, \_c1#52125, \_c2#52126, \_c3#52127, \_c4#52128]

: +- Relation [\_c0#52124,\_c1#52125,\_c2#52126,\_c3#52127,\_c4#52128] csv

+- SubqueryAlias r

+- SubqueryAlias view2

+- View (`view2`, [BusinessDate#57031,stcd#57060,srcnm#57061,NetworkName#57034,NetworkURL#57035,custnum#57036,MarketCoverage#57037,RejectRows#57039,IssuerId#57059,IssuerId2#57058,issueridcomposite#57057,sysdt#57056,systs#57055,cleannetworkname#57054,curdt#57052,curts#57053,cleannetworkname#57051,Protocol\_Used#48671])

+- Project [BusinessDate#57031, stcd#57060, srcnm#57061, NetworkName#57034, NetworkURL#57035, custnum#57036, MarketCoverage#57037, RejectRows#57039, IssuerId#57059, IssuerId2#57058, issueridcomposite#57057, sysdt#57056, systs#57055, cleannetworkname#57054, curdt#57052, curts#57053, cleannetworkname#57051, CASE WHEN NetworkURL#57035 LIKE https://% THEN http secured WHEN NetworkURL#57035 LIKE http://% THEN http non secured ELSE No Protocol END AS Protocol\_Used#48671]

+- SubqueryAlias insureview

+- View (`insureview`, [BusinessDate#57031,stcd#57060,srcnm#57061,NetworkName#57034,NetworkURL#57035,custnum#57036,MarketCoverage#57037,RejectRows#57039,IssuerId#57059,IssuerId2#57058,issueridcomposite#57057,sysdt#57056,systs#57055,cleannetworkname#57054,curdt#57052,curts#57053,cleannetworkname#57051])

+- Project [BusinessDate#57031, stcd#57060, srcnm#57061, NetworkName#57034, NetworkURL#57035, custnum#57036, MarketCoverage#57037, RejectRows#57039, IssuerId#57059, IssuerId2#57058, issueridcomposite#57057, sysdt#57056, systs#57055, cleannetworkname#57054, curdt#57052, curts#57053, sql\_function(NetworkName#57034) AS cleannetworkname#57051]

+- SubqueryAlias insureview

+- View (`insureview`, [BusinessDate#57031,stcd#57060,srcnm#57061,NetworkName#57034,NetworkURL#57035,custnum#57036,MarketCoverage#57037,RejectRows#57039,IssuerId#57059,IssuerId2#57058,issueridcomposite#57057,sysdt#57056,systs#57055,cleannetworkname#57054,curdt#57052,curts#57053])

+- Project [BusinessDate#57031, stcd#57060, srcnm#57061, NetworkName#57034, NetworkURL#57035, custnum#57036, MarketCoverage#57037, RejectRows#57039, IssuerId#57059, IssuerId2#57058, issueridcomposite#57057, sysdt#57056, systs#57055, cleannetworkname#57054, current\_date(Some(Asia/Kolkata)) AS curdt#57052, current\_timestamp() AS curts#57053]

+- SubqueryAlias insureview

+- View (`insureview`, [BusinessDate#57031,stcd#57060,srcnm#57061,NetworkName#57034,NetworkURL#57035,custnum#57036,MarketCoverage#57037,RejectRows#57039,IssuerId#57059,IssuerId2#57058,issueridcomposite#57057,sysdt#57056,systs#57055,cleannetworkname#57054])

+- Project [BusinessDate#57031, stcd#57060, srcnm#57061, NetworkName#57034, NetworkURL#57035, custnum#57036, MarketCoverage#57037, RejectRows#57039, IssuerId#57059, IssuerId2#57058, issueridcomposite#57057, sysdt#57056, systs#57055, sql\_function(NetworkName#57034) AS cleannetworkname#57054]

+- SubqueryAlias insureview

+- View (`insureview`, [BusinessDate#57031,stcd#57060,srcnm#57061,NetworkName#57034,NetworkURL#57035,custnum#57036,MarketCoverage#57037,RejectRows#57039,IssuerId#57059,IssuerId2#57058,issueridcomposite#57057,sysdt#57056,systs#57055])

+- Project [BusinessDate#57031, stcd#57060, srcnm#57061, NetworkName#57034, NetworkURL#57035, custnum#57036, MarketCoverage#57037, RejectRows#57039, IssuerId#57059, IssuerId2#57058, issueridcomposite#57057, sysdt#57056, current\_timestamp() AS systs#57055]

+- Project [BusinessDate#57031, stcd#57060, srcnm#57061, NetworkName#57034, NetworkURL#57035, custnum#57036, MarketCoverage#57037, RejectRows#57039, IssuerId#57059, IssuerId2#57058, issueridcomposite#57057, current\_date(Some(Asia/Kolkata)) AS sysdt#57056]

+- Project [BusinessDate#57031, stcd#57060, srcnm#57061, NetworkName#57034, NetworkURL#57035, custnum#57036, MarketCoverage#57037, RejectRows#57039, IssuerId#57059, IssuerId2#57058, issueridcomposite#57057]

+- Project [BusinessDate#57031, stcd#57060, srcnm#57061, NetworkName#57034, NetworkURL#57035, custnum#57036, MarketCoverage#57037, DentalOnlyPlan#57038, RejectRows#57039, IssuerId#57059, IssuerId2#57058, concat(IssuerId#57059, IssuerId2#57058) AS issueridcomposite#57057]

+- Project [BusinessDate#57031, stcd#57060, srcnm#57061, NetworkName#57034, NetworkURL#57035, custnum#57036, MarketCoverage#57037, DentalOnlyPlan#57038, RejectRows#57039, IssuerId#57059, IssuerId2#57058]

+- Project [IssuerIDDD#8273, BusinessDate#57031, stcd#57060, srcnm#57061, NetworkName#57034, NetworkURL#57035, custnum#57036, MarketCoverage#57037, DentalOnlyPlan#57038, RejectRows#57039, IssuerId#57059, IssuerId2#57058]

+- Project [IssuerIDD#8261, IssuerIDDD#8273, BusinessDate#57031, stcd#57060, srcnm#57061, NetworkName#57034, NetworkURL#57035, custnum#57036, MarketCoverage#57037, DentalOnlyPlan#57038, RejectRows#57039, IssuerId#57059, cast(IssuerIDDD#8273 as string) AS IssuerId2#57058]

+- Project [IssuerIDD#8261, IssuerIDDD#8273, BusinessDate#57031, stcd#57060, srcnm#57061, NetworkName#57034, NetworkURL#57035, custnum#57036, MarketCoverage#57037, DentalOnlyPlan#57038, RejectRows#57039, cast(IssuerIDD#8261 as string) AS IssuerId#57059]

+- Project [IssuerIDD#8261, IssuerId2#57030 AS IssuerIDDD#8273, BusinessDate#57031, stcd#57060, srcnm#57061, NetworkName#57034, NetworkURL#57035, custnum#57036, MarketCoverage#57037, DentalOnlyPlan#57038, RejectRows#57039]

+- Project [IssuerId#57029 AS IssuerIDD#8261, IssuerId2#57030, BusinessDate#57031, stcd#57060, srcnm#57061, NetworkName#57034, NetworkURL#57035, custnum#57036, MarketCoverage#57037, DentalOnlyPlan#57038, RejectRows#57039]

+- Project [IssuerId#57029, IssuerId2#57030, BusinessDate#57031, StateCode#57032 AS stcd#57060, srcnm#57061, NetworkName#57034, NetworkURL#57035, custnum#57036, MarketCoverage#57037, DentalOnlyPlan#57038, RejectRows#57039]

+- Project [IssuerId#57029, IssuerId2#57030, BusinessDate#57031, StateCode#57032, SourceName#57033 AS srcnm#57061, NetworkName#57034, NetworkURL#57035, custnum#57036, MarketCoverage#57037, DentalOnlyPlan#57038, RejectRows#57039]

+- Union false, false

:- Relation [IssuerId#57029,IssuerId2#57030,BusinessDate#57031,StateCode#57032,SourceName#57033,NetworkName#57034,NetworkURL#57035,custnum#57036,MarketCoverage#57037,DentalOnlyPlan#57038,RejectRows#57039] csv

+- Relation [IssuerId#57040,IssuerId2#57041,BusinessDate#57042,StateCode#57043,SourceName#57044,NetworkName#57045,NetworkURL#57046,custnum#57047,MarketCoverage#57048,DentalOnlyPlan#57049,RejectRows#57050] csv

>>> count\_df = second\_join.groupBy("stcd", "Protocol\_Used", "profession") \

... .agg(

... avg("age").alias("avg\_age"),

... count("\*").alias("profession\_count")

... ) \

... .withColumn("seqno", row\_number().over(window\_spec))sdbksdv

File "<stdin>", line 6

.withColumn("seqno", row\_number().over(window\_spec))sdbksdv

^

SyntaxError: invalid syntax

>>> count\_df = second\_join.groupBy("stcd", "Protocol\_Used", "profession") \

... .agg(

... avg("\_c1").alias("avg\_age"),

... count("\*").alias("profession\_count")

... ) \

... .withColumn("seqno", row\_number().over(window\_spec))

Traceback (most recent call last):

File "<stdin>", line 4, in <module>

File "/usr/local/spark/python/pyspark/sql/group.py", line 119, in agg

\_to\_seq(self.sql\_ctx.\_sc, [c.\_jc for c in exprs[1:]]))

File "/usr/local/spark/python/lib/py4j-0.10.9.3-src.zip/py4j/java\_gateway.py", line 1322, in \_\_call\_\_

File "/usr/local/spark/python/pyspark/sql/utils.py", line 117, in deco

raise converted from None

pyspark.sql.utils.AnalysisException: Reference '\_c1' is ambiguous, could be: l.\_c1, l.\_c1.

>>> count\_df = second\_join.groupBy("stcd", "Protocol\_Used", "profession") \

... .agg(

... avg("\_c3").alias("avg\_age"),

... count("\*").alias("profession\_count")

... ) \

... .withColumn("seqno", row\_number().over(window\_spec))

Traceback (most recent call last):

File "<stdin>", line 4, in <module>

File "/usr/local/spark/python/pyspark/sql/group.py", line 119, in agg

\_to\_seq(self.sql\_ctx.\_sc, [c.\_jc for c in exprs[1:]]))

File "/usr/local/spark/python/lib/py4j-0.10.9.3-src.zip/py4j/java\_gateway.py", line 1322, in \_\_call\_\_

File "/usr/local/spark/python/pyspark/sql/utils.py", line 117, in deco

raise converted from None

pyspark.sql.utils.AnalysisException: cannot resolve 'profession' given input columns: [l.BusinessDate, l.IssuerId, l.IssuerId2, l.MarketCoverage, l.NetworkName, l.NetworkURL, r.Protocol\_Used, l.RejectRows, l.\_c0, l.\_c0, l.\_c1, l.\_c1, l.\_c2, l.\_c3, l.\_c4, l.cleannetworkname, l.cleannetworkname, l.curdt, l.curts, l.custnum, l.issueridcomposite, l.srcnm, l.stcd, l.sysdt, l.systs];

'Aggregate [stcd#7759, Protocol\_Used#48671, 'profession], [stcd#7759, Protocol\_Used#48671, 'profession, avg(\_c3#52127) AS avg\_age#58343, count(1) AS profession\_count#58345L]

+- Project [BusinessDate#1378, stcd#7759, srcnm#7747, NetworkName#1381, NetworkURL#1382, custnum#1383, MarketCoverage#1384, RejectRows#1386, IssuerId#8312, IssuerId2#8325, issueridcomposite#8901, sysdt#9003, systs#9016, cleannetworkname#47665, curdt#48091, curts#48092, cleannetworkname#48111, \_c0#26007, \_c1#26008, \_c0#52124, \_c1#52125, \_c2#52126, \_c3#52127, \_c4#52128, Protocol\_Used#48671]

+- Join Inner

:- SubqueryAlias l

: +- SubqueryAlias view1

: +- View (`view1`, [BusinessDate#1378,stcd#7759,srcnm#7747,NetworkName#1381,NetworkURL#1382,custnum#1383,MarketCoverage#1384,RejectRows#1386,IssuerId#8312,IssuerId2#8325,issueridcomposite#8901,sysdt#9003,systs#9016,cleannetworkname#47665,curdt#48091,curts#48092,cleannetworkname#48111,\_c0#26007,\_c1#26008,\_c0#52124,\_c1#52125,\_c2#52126,\_c3#52127,\_c4#52128])

: +- Project [BusinessDate#1378, stcd#7759, srcnm#7747, NetworkName#1381, NetworkURL#1382, custnum#1383, MarketCoverage#1384, RejectRows#1386, IssuerId#8312, IssuerId2#8325, issueridcomposite#8901, sysdt#9003, systs#9016, cleannetworkname#47665, curdt#48091, curts#48092, cleannetworkname#48111, \_c0#26007, \_c1#26008, \_c0#52124, \_c1#52125, \_c2#52126, \_c3#52127, \_c4#52128]

: +- Join Inner, (cast(custnum#1383 as int) = \_c3#52127)

: :- SubqueryAlias l

: : +- SubqueryAlias insureview

: : +- View (`insureview`, [BusinessDate#1378,stcd#7759,srcnm#7747,NetworkName#1381,NetworkURL#1382,custnum#1383,MarketCoverage#1384,RejectRows#1386,IssuerId#8312,IssuerId2#8325,issueridcomposite#8901,sysdt#9003,systs#9016,cleannetworkname#47665,curdt#48091,curts#48092,cleannetworkname#48111,\_c0#26007,\_c1#26008])

: : +- Project [BusinessDate#1378, stcd#7759, srcnm#7747, NetworkName#1381, NetworkURL#1382, custnum#1383, MarketCoverage#1384, RejectRows#1386, IssuerId#8312, IssuerId2#8325, issueridcomposite#8901, sysdt#9003, systs#9016, cleannetworkname#47665, curdt#48091, curts#48092, cleannetworkname#48111, \_c0#26007, \_c1#26008]

: : +- Join Inner, (stcd#7759 = \_c0#26007)

: : :- SubqueryAlias l

: : : +- SubqueryAlias insureview

: : : +- View (`insureview`, [BusinessDate#1378,stcd#7759,srcnm#7747,NetworkName#1381,NetworkURL#1382,custnum#1383,MarketCoverage#1384,RejectRows#1386,IssuerId#8312,IssuerId2#8325,issueridcomposite#8901,sysdt#9003,systs#9016,cleannetworkname#47665,curdt#48091,curts#48092,cleannetworkname#48111])

: : : +- Project [BusinessDate#1378, stcd#7759, srcnm#7747, NetworkName#1381, NetworkURL#1382, custnum#1383, MarketCoverage#1384, RejectRows#1386, IssuerId#8312, IssuerId2#8325, issueridcomposite#8901, sysdt#9003, systs#9016, cleannetworkname#47665, curdt#48091, curts#48092, sql\_function(NetworkName#1381) AS cleannetworkname#48111]

: : : +- SubqueryAlias insureview

: : : +- View (`insureview`, [BusinessDate#1378,stcd#7759,srcnm#7747,NetworkName#1381,NetworkURL#1382,custnum#1383,MarketCoverage#1384,RejectRows#1386,IssuerId#8312,IssuerId2#8325,issueridcomposite#8901,sysdt#9003,systs#9016,cleannetworkname#47665,curdt#48091,curts#48092])

: : : +- Project [BusinessDate#1378, stcd#7759, srcnm#7747, NetworkName#1381, NetworkURL#1382, custnum#1383, MarketCoverage#1384, RejectRows#1386, IssuerId#8312, IssuerId2#8325, issueridcomposite#8901, sysdt#9003, systs#9016, cleannetworkname#47665, current\_date(Some(Asia/Kolkata)) AS curdt#48091, current\_timestamp() AS curts#48092]

: : : +- SubqueryAlias insureview

: : : +- View (`insureview`, [BusinessDate#1378,stcd#7759,srcnm#7747,NetworkName#1381,NetworkURL#1382,custnum#1383,MarketCoverage#1384,RejectRows#1386,IssuerId#8312,IssuerId2#8325,issueridcomposite#8901,sysdt#9003,systs#9016,cleannetworkname#47665])

: : : +- Project [BusinessDate#1378, stcd#7759, srcnm#7747, NetworkName#1381, NetworkURL#1382, custnum#1383, MarketCoverage#1384, RejectRows#1386, IssuerId#8312, IssuerId2#8325, issueridcomposite#8901, sysdt#9003, systs#9016, sql\_function(NetworkName#1381) AS cleannetworkname#47665]

: : : +- SubqueryAlias insureview

: : : +- View (`insureview`, [BusinessDate#1378,stcd#7759,srcnm#7747,NetworkName#1381,NetworkURL#1382,custnum#1383,MarketCoverage#1384,RejectRows#1386,IssuerId#8312,IssuerId2#8325,issueridcomposite#8901,sysdt#9003,systs#9016])

: : : +- Project [BusinessDate#1378, stcd#7759, srcnm#7747, NetworkName#1381, NetworkURL#1382, custnum#1383, MarketCoverage#1384, RejectRows#1386, IssuerId#8312, IssuerId2#8325, issueridcomposite#8901, sysdt#9003, current\_timestamp() AS systs#9016]

: : : +- Project [BusinessDate#1378, stcd#7759, srcnm#7747, NetworkName#1381, NetworkURL#1382, custnum#1383, MarketCoverage#1384, RejectRows#1386, IssuerId#8312, IssuerId2#8325, issueridcomposite#8901, current\_date(Some(Asia/Kolkata)) AS sysdt#9003]

: : : +- Project [BusinessDate#1378, stcd#7759, srcnm#7747, NetworkName#1381, NetworkURL#1382, custnum#1383, MarketCoverage#1384, RejectRows#1386, IssuerId#8312, IssuerId2#8325, issueridcomposite#8901]

: : : +- Project [BusinessDate#1378, stcd#7759, srcnm#7747, NetworkName#1381, NetworkURL#1382, custnum#1383, MarketCoverage#1384, DentalOnlyPlan#1385, RejectRows#1386, IssuerId#8312, IssuerId2#8325, concat(IssuerId#8312, IssuerId2#8325) AS issueridcomposite#8901]

: : : +- Project [BusinessDate#1378, stcd#7759, srcnm#7747, NetworkName#1381, NetworkURL#1382, custnum#1383, MarketCoverage#1384, DentalOnlyPlan#1385, RejectRows#1386, IssuerId#8312, IssuerId2#8325]

: : : +- Project [IssuerIDDD#8273, BusinessDate#1378, stcd#7759, srcnm#7747, NetworkName#1381, NetworkURL#1382, custnum#1383, MarketCoverage#1384, DentalOnlyPlan#1385, RejectRows#1386, IssuerId#8312, IssuerId2#8325]

: : : +- Project [IssuerIDD#8261, IssuerIDDD#8273, BusinessDate#1378, stcd#7759, srcnm#7747, NetworkName#1381, NetworkURL#1382, custnum#1383, MarketCoverage#1384, DentalOnlyPlan#1385, RejectRows#1386, IssuerId#8312, cast(IssuerIDDD#8273 as string) AS IssuerId2#8325]

: : : +- Project [IssuerIDD#8261, IssuerIDDD#8273, BusinessDate#1378, stcd#7759, srcnm#7747, NetworkName#1381, NetworkURL#1382, custnum#1383, MarketCoverage#1384, DentalOnlyPlan#1385, RejectRows#1386, cast(IssuerIDD#8261 as string) AS IssuerId#8312]

: : : +- Project [IssuerIDD#8261, IssuerId2#1377 AS IssuerIDDD#8273, BusinessDate#1378, stcd#7759, srcnm#7747, NetworkName#1381, NetworkURL#1382, custnum#1383, MarketCoverage#1384, DentalOnlyPlan#1385, RejectRows#1386]

: : : +- Project [IssuerId#1376 AS IssuerIDD#8261, IssuerId2#1377, BusinessDate#1378, stcd#7759, srcnm#7747, NetworkName#1381, NetworkURL#1382, custnum#1383, MarketCoverage#1384, DentalOnlyPlan#1385, RejectRows#1386]

: : : +- Project [IssuerId#1376, IssuerId2#1377, BusinessDate#1378, StateCode#1379 AS stcd#7759, srcnm#7747, NetworkName#1381, NetworkURL#1382, custnum#1383, MarketCoverage#1384, DentalOnlyPlan#1385, RejectRows#1386]

: : : +- Project [IssuerId#1376, IssuerId2#1377, BusinessDate#1378, StateCode#1379, SourceName#1380 AS srcnm#7747, NetworkName#1381, NetworkURL#1382, custnum#1383, MarketCoverage#1384, DentalOnlyPlan#1385, RejectRows#1386]

: : : +- Union false, false

: : : :- Relation [IssuerId#1376,IssuerId2#1377,BusinessDate#1378,StateCode#1379,SourceName#1380,NetworkName#1381,NetworkURL#1382,custnum#1383,MarketCoverage#1384,DentalOnlyPlan#1385,RejectRows#1386] csv

: : : +- Relation [IssuerId#491,IssuerId2#492,BusinessDate#493,StateCode#494,SourceName#495,NetworkName#496,NetworkURL#497,custnum#498,MarketCoverage#499,DentalOnlyPlan#500,RejectRows#501] csv

: : +- SubqueryAlias r

: : +- SubqueryAlias state\_table

: : +- View (`state\_table`, [\_c0#26007,\_c1#26008])

: : +- Project [\_c0#26007, \_c1#26008]

: : +- Filter ((isnull(\_c2#26009) AND isnull(\_c3#26010)) AND isnull(\_c4#26011))

: : +- Project [\_c0#26007, \_c1#26008, \_c2#26009, \_c3#26010, \_c4#26011]

: : +- Relation [\_c0#26007,\_c1#26008,\_c2#26009,\_c3#26010,\_c4#26011] csv

: +- SubqueryAlias r

: +- SubqueryAlias cust\_table

: +- View (`cust\_table`, [\_c0#52124,\_c1#52125,\_c2#52126,\_c3#52127,\_c4#52128])

: +- Filter (((isnotnull(\_c0#52124) AND isnotnull(\_c1#52125)) AND isnotnull(\_c2#52126)) AND (isnotnull(\_c3#52127) AND isnotnull(\_c4#52128)))

: +- Project [\_c0#52124, \_c1#52125, \_c2#52126, \_c3#52127, \_c4#52128]

: +- Relation [\_c0#52124,\_c1#52125,\_c2#52126,\_c3#52127,\_c4#52128] csv

+- SubqueryAlias r

+- SubqueryAlias view2

+- View (`view2`, [BusinessDate#57031,stcd#57060,srcnm#57061,NetworkName#57034,NetworkURL#57035,custnum#57036,MarketCoverage#57037,RejectRows#57039,IssuerId#57059,IssuerId2#57058,issueridcomposite#57057,sysdt#57056,systs#57055,cleannetworkname#57054,curdt#57052,curts#57053,cleannetworkname#57051,Protocol\_Used#48671])

+- Project [BusinessDate#57031, stcd#57060, srcnm#57061, NetworkName#57034, NetworkURL#57035, custnum#57036, MarketCoverage#57037, RejectRows#57039, IssuerId#57059, IssuerId2#57058, issueridcomposite#57057, sysdt#57056, systs#57055, cleannetworkname#57054, curdt#57052, curts#57053, cleannetworkname#57051, CASE WHEN NetworkURL#57035 LIKE https://% THEN http secured WHEN NetworkURL#57035 LIKE http://% THEN http non secured ELSE No Protocol END AS Protocol\_Used#48671]

+- SubqueryAlias insureview

+- View (`insureview`, [BusinessDate#57031,stcd#57060,srcnm#57061,NetworkName#57034,NetworkURL#57035,custnum#57036,MarketCoverage#57037,RejectRows#57039,IssuerId#57059,IssuerId2#57058,issueridcomposite#57057,sysdt#57056,systs#57055,cleannetworkname#57054,curdt#57052,curts#57053,cleannetworkname#57051])

+- Project [BusinessDate#57031, stcd#57060, srcnm#57061, NetworkName#57034, NetworkURL#57035, custnum#57036, MarketCoverage#57037, RejectRows#57039, IssuerId#57059, IssuerId2#57058, issueridcomposite#57057, sysdt#57056, systs#57055, cleannetworkname#57054, curdt#57052, curts#57053, sql\_function(NetworkName#57034) AS cleannetworkname#57051]

+- SubqueryAlias insureview

+- View (`insureview`, [BusinessDate#57031,stcd#57060,srcnm#57061,NetworkName#57034,NetworkURL#57035,custnum#57036,MarketCoverage#57037,RejectRows#57039,IssuerId#57059,IssuerId2#57058,issueridcomposite#57057,sysdt#57056,systs#57055,cleannetworkname#57054,curdt#57052,curts#57053])

+- Project [BusinessDate#57031, stcd#57060, srcnm#57061, NetworkName#57034, NetworkURL#57035, custnum#57036, MarketCoverage#57037, RejectRows#57039, IssuerId#57059, IssuerId2#57058, issueridcomposite#57057, sysdt#57056, systs#57055, cleannetworkname#57054, current\_date(Some(Asia/Kolkata)) AS curdt#57052, current\_timestamp() AS curts#57053]

+- SubqueryAlias insureview

+- View (`insureview`, [BusinessDate#57031,stcd#57060,srcnm#57061,NetworkName#57034,NetworkURL#57035,custnum#57036,MarketCoverage#57037,RejectRows#57039,IssuerId#57059,IssuerId2#57058,issueridcomposite#57057,sysdt#57056,systs#57055,cleannetworkname#57054])

+- Project [BusinessDate#57031, stcd#57060, srcnm#57061, NetworkName#57034, NetworkURL#57035, custnum#57036, MarketCoverage#57037, RejectRows#57039, IssuerId#57059, IssuerId2#57058, issueridcomposite#57057, sysdt#57056, systs#57055, sql\_function(NetworkName#57034) AS cleannetworkname#57054]

+- SubqueryAlias insureview

+- View (`insureview`, [BusinessDate#57031,stcd#57060,srcnm#57061,NetworkName#57034,NetworkURL#57035,custnum#57036,MarketCoverage#57037,RejectRows#57039,IssuerId#57059,IssuerId2#57058,issueridcomposite#57057,sysdt#57056,systs#57055])

+- Project [BusinessDate#57031, stcd#57060, srcnm#57061, NetworkName#57034, NetworkURL#57035, custnum#57036, MarketCoverage#57037, RejectRows#57039, IssuerId#57059, IssuerId2#57058, issueridcomposite#57057, sysdt#57056, current\_timestamp() AS systs#57055]

+- Project [BusinessDate#57031, stcd#57060, srcnm#57061, NetworkName#57034, NetworkURL#57035, custnum#57036, MarketCoverage#57037, RejectRows#57039, IssuerId#57059, IssuerId2#57058, issueridcomposite#57057, current\_date(Some(Asia/Kolkata)) AS sysdt#57056]

+- Project [BusinessDate#57031, stcd#57060, srcnm#57061, NetworkName#57034, NetworkURL#57035, custnum#57036, MarketCoverage#57037, RejectRows#57039, IssuerId#57059, IssuerId2#57058, issueridcomposite#57057]

+- Project [BusinessDate#57031, stcd#57060, srcnm#57061, NetworkName#57034, NetworkURL#57035, custnum#57036, MarketCoverage#57037, DentalOnlyPlan#57038, RejectRows#57039, IssuerId#57059, IssuerId2#57058, concat(IssuerId#57059, IssuerId2#57058) AS issueridcomposite#57057]

+- Project [BusinessDate#57031, stcd#57060, srcnm#57061, NetworkName#57034, NetworkURL#57035, custnum#57036, MarketCoverage#57037, DentalOnlyPlan#57038, RejectRows#57039, IssuerId#57059, IssuerId2#57058]

+- Project [IssuerIDDD#8273, BusinessDate#57031, stcd#57060, srcnm#57061, NetworkName#57034, NetworkURL#57035, custnum#57036, MarketCoverage#57037, DentalOnlyPlan#57038, RejectRows#57039, IssuerId#57059, IssuerId2#57058]

+- Project [IssuerIDD#8261, IssuerIDDD#8273, BusinessDate#57031, stcd#57060, srcnm#57061, NetworkName#57034, NetworkURL#57035, custnum#57036, MarketCoverage#57037, DentalOnlyPlan#57038, RejectRows#57039, IssuerId#57059, cast(IssuerIDDD#8273 as string) AS IssuerId2#57058]

+- Project [IssuerIDD#8261, IssuerIDDD#8273, BusinessDate#57031, stcd#57060, srcnm#57061, NetworkName#57034, NetworkURL#57035, custnum#57036, MarketCoverage#57037, DentalOnlyPlan#57038, RejectRows#57039, cast(IssuerIDD#8261 as string) AS IssuerId#57059]

+- Project [IssuerIDD#8261, IssuerId2#57030 AS IssuerIDDD#8273, BusinessDate#57031, stcd#57060, srcnm#57061, NetworkName#57034, NetworkURL#57035, custnum#57036, MarketCoverage#57037, DentalOnlyPlan#57038, RejectRows#57039]

+- Project [IssuerId#57029 AS IssuerIDD#8261, IssuerId2#57030, BusinessDate#57031, stcd#57060, srcnm#57061, NetworkName#57034, NetworkURL#57035, custnum#57036, MarketCoverage#57037, DentalOnlyPlan#57038, RejectRows#57039]

+- Project [IssuerId#57029, IssuerId2#57030, BusinessDate#57031, StateCode#57032 AS stcd#57060, srcnm#57061, NetworkName#57034, NetworkURL#57035, custnum#57036, MarketCoverage#57037, DentalOnlyPlan#57038, RejectRows#57039]

+- Project [IssuerId#57029, IssuerId2#57030, BusinessDate#57031, StateCode#57032, SourceName#57033 AS srcnm#57061, NetworkName#57034, NetworkURL#57035, custnum#57036, MarketCoverage#57037, DentalOnlyPlan#57038, RejectRows#57039]

+- Union false, false

:- Relation [IssuerId#57029,IssuerId2#57030,BusinessDate#57031,StateCode#57032,SourceName#57033,NetworkName#57034,NetworkURL#57035,custnum#57036,MarketCoverage#57037,DentalOnlyPlan#57038,RejectRows#57039] csv

+- Relation [IssuerId#57040,IssuerId2#57041,BusinessDate#57042,StateCode#57043,SourceName#57044,NetworkName#57045,NetworkURL#57046,custnum#57047,MarketCoverage#57048,DentalOnlyPlan#57049,RejectRows#57050] csv

>>> second\_join.describe

>>> second\_join.describe

<bound method DataFrame.describe of DataFrame[BusinessDate: date, stcd: string, srcnm: string, NetworkName: string, NetworkURL: string, custnum: string, MarketCoverage: string, RejectRows: string, IssuerId: string, IssuerId2: string, issueridcomposite: string, sysdt: date, systs: timestamp, cleannetworkname: string, curdt: date, curts: timestamp, cleannetworkname: string, \_c0: string, \_c1: string, \_c0: string, \_c1: string, \_c2: string, \_c3: int, \_c4: string, Protocol\_Used: string]>

>>> second\_join.show(1)

+------------+----+-----+-----------+--------------------+-------+--------------+----------+--------+---------+-----------------+----------+--------------------+----------------+----------+--------------------+----------------+---+-------+---+-----+-------+---+-----+-------------+

|BusinessDate|stcd|srcnm|NetworkName| NetworkURL|custnum|MarketCoverage|RejectRows|IssuerId|IssuerId2|issueridcomposite| sysdt| systs|cleannetworkname| curdt| curts|cleannetworkname|\_c0| \_c1|\_c0| \_c1| \_c2|\_c3| \_c4|Protocol\_Used|

+------------+----+-----+-----------+--------------------+-------+--------------+----------+--------+---------+-----------------+----------+--------------------+----------------+----------+--------------------+----------------+---+-------+---+-----+-------+---+-----+-------------+

| 2019-10-01| AZ| HIOS| La Paz|http://uamarketpl...| 21| null| null| 88925| 88925| 8892588925|2024-07-22|2024-07-22 16:59:...| La Paz|2024-07-22|2024-07-22 16:59:...| La Paz| AZ|Arizona|779|Jason|Bowling| 21|Coach| http secured|

+------------+----+-----+-----------+--------------------+-------+--------------+----------+--------+---------+-----------------+----------+--------------------+----------------+----------+--------------------+----------------+---+-------+---+-----+-------+---+-----+-------------+

only showing top 1 row

>>> from pyspark.sql import functions as F

>>> from pyspark.sql.window import Window

>>>

>>> # Step 1: Calculate average age and count

... grouped\_df = second\_join.groupBy('srcnm', 'Protocol\_Used', '\_c4') \

... .agg(

... F.avg('\_c3').alias('avg\_age'),

... F.count('\*').alias('profession\_count')

... )

>>> window\_spec = Window.partitionBy('Protocol\_Used', 'srcnm') \

... .orderBy(F.desc('profession\_count'))

>>> ranked\_df = grouped\_df.withColumn('seqno', F.row\_number().over(window\_spec))

>>> result\_df = ranked\_df.filter(ranked\_df.seqno == 2)

>>> result\_df.show(2)

+-----+-------------+-----+------------------+----------------+-----+

|srcnm|Protocol\_Used| \_c4| avg\_age|profession\_count|seqno|

+-----+-------------+-----+------------------+----------------+-----+

| HIOS| No Protocol|Nurse|22.076923076923077| 28158| 2|

|SERFF| No Protocol|Nurse| 21.0| 12996| 2|

+-----+-------------+-----+------------------+----------------+-----+

only showing top 2 rows

>>>

>>> mysql\_url = "jdbc://mysql://localhost/Hackathon"

>>> mysql\_table = "hackathon\_tbl"

>>> mysql\_user = "root"

>>> mysql\_password = "Root123$"

>>> result\_df.write .format("jdbc") .option("url", mysql\_url) .option("dbtable", mysql\_table) .option("user", mysql\_user) .option("password", mysql\_password) .option("driver", "com.mysql.cj.jdbc.Driver") .mode("overwrite") .save()

Traceback (most recent call last):

File "<stdin>", line 1, in <module>

File "/usr/local/spark/python/pyspark/sql/readwriter.py", line 738, in save

self.\_jwrite.save()

File "/usr/local/spark/python/lib/py4j-0.10.9.3-src.zip/py4j/java\_gateway.py", line 1322, in \_\_call\_\_

File "/usr/local/spark/python/pyspark/sql/utils.py", line 117, in deco

raise converted from None

pyspark.sql.utils.IllegalArgumentException: requirement failed: The driver could not open a JDBC connection. Check the URL: jdbc://mysql://localhost/Hackathon

>>> mysql\_url = "jdbc:mysql://localhost/Hackathon"

>>> mysql\_table = "hackathon\_tbl"

>>> mysql\_user = "root"

>>> mysql\_password = "Root123$"

>>> result\_df.write .format("jdbc") .option("url", mysql\_url) .option("dbtable", mysql\_table) .option("user", mysql\_user) .option("password", mysql\_password) .option("driver", "com.mysql.cj.jdbc.Driver") .mode("overwrite") .save()

>>> >>> rdd.take(4)

File "<stdin>", line 1

>>> rdd.take(4)

^

SyntaxError: invalid syntax

>>> ['1,Kristina,Chung,55,Pilot', '2,Paige,Chen,74,Teacher', 'NM,New Mexico', 'NY,New York']

['1,Kristina,Chung,55,Pilot', '2,Paige,Chen,74,Teacher', 'NM,New Mexico', 'NY,New York']

>>> >>> custfilter =rdd.map(lambda x: x.split(",") )

File "<stdin>", line 1

>>> custfilter =rdd.map(lambda x: x.split(",") )

^

SyntaxError: invalid syntax

>>> >>> print(custfilter.take(14))

File "<stdin>", line 1

>>> print(custfilter.take(14))

^

SyntaxError: invalid syntax

>>> [['1', 'Kristina', 'Chung', '55', 'Pilot'], ['2', 'Paige', 'Chen', '74', 'Teacher'], ['NM', 'New Mexico'], ['NY', 'New York'], ['NC', 'North Carolina'], ['ND', 'North Dakota'], ['OH', 'Ohio'], ['OK', 'Oklahoma'], ['OR', 'Oregon'], ['MD', 'Maryland'], ['MA', 'Massachusetts'], ['MI', 'Michigan'], ['MN', 'Minnesota'], ['MS', 'Mississippi']]

[['1', 'Kristina', 'Chung', '55', 'Pilot'], ['2', 'Paige', 'Chen', '74', 'Teacher'], ['NM', 'New Mexico'], ['NY', 'New York'], ['NC', 'North Carolina'], ['ND', 'North Dakota'], ['OH', 'Ohio'], ['OK', 'Oklahoma'], ['OR', 'Oregon'], ['MD', 'Maryland'], ['MA', 'Massachusetts'], ['MI', 'Michigan'], ['MN', 'Minnesota'], ['MS', 'Mississippi']]

>>> >>> splited\_rdd =rdd.map(lambda x: x.split(",") )

File "<stdin>", line 1

>>> splited\_rdd =rdd.map(lambda x: x.split(",") )

^

SyntaxError: invalid syntax

>>> >>> custfilter =splited\_rdd.filter(lambda x: len(x)>3 )

File "<stdin>", line 1

>>> custfilter =splited\_rdd.filter(lambda x: len(x)>3 )

^

SyntaxError: invalid syntax

>>> >>> print(custfilter.take(4))

File "<stdin>", line 1

>>> print(custfilter.take(4))

^

SyntaxError: invalid syntax

>>> [['1', 'Kristina', 'Chung', '55', 'Pilot'], ['2', 'Paige', 'Chen', '74', 'Teacher'], ['3', 'Sherri', 'Melton', '34', 'Firefighter'], ['4', 'Gretchen', 'Hill', '66', 'Computer hardware engineer']]

[['1', 'Kristina', 'Chung', '55', 'Pilot'], ['2', 'Paige', 'Chen', '74', 'Teacher'], ['3', 'Sherri', 'Melton', '34', 'Firefighter'], ['4', 'Gretchen', 'Hill', '66', 'Computer hardware engineer']]

>>> >>> print(custfilter.take(5))

File "<stdin>", line 1

>>> print(custfilter.take(5))

^

SyntaxError: invalid syntax

>>> [['1', 'Kristina', 'Chung', '55', 'Pilot'], ['2', 'Paige', 'Chen', '74', 'Teacher'], ['3', 'Sherri', 'Melton', '34', 'Firefighter'], ['4', 'Gretchen', 'Hill', '66', 'Computer hardware engineer'], ['5', 'Karen', 'Puckett', '74', 'Lawyer']]

[['1', 'Kristina', 'Chung', '55', 'Pilot'], ['2', 'Paige', 'Chen', '74', 'Teacher'], ['3', 'Sherri', 'Melton', '34', 'Firefighter'], ['4', 'Gretchen', 'Hill', '66', 'Computer hardware engineer'], ['5', 'Karen', 'Puckett', '74', 'Lawyer']]

>>> >>> statefilter =splited\_rdd.filter(lambda x: len(x)<3 )

File "<stdin>", line 1

>>> statefilter =splited\_rdd.filter(lambda x: len(x)<3 )

^

SyntaxError: invalid syntax

>>> >>> print(statefilter.take(4))

File "<stdin>", line 1

>>> print(statefilter.take(4))

^

SyntaxError: invalid syntax

>>> [['NM', 'New Mexico'], ['NY', 'New York'], ['NC', 'North Carolina'], ['ND', 'North Dakota']]

[['NM', 'New Mexico'], ['NY', 'New York'], ['NC', 'North Carolina'], ['ND', 'North Dakota']]

>>> >>>

File "<stdin>", line 1

>>>

^

SyntaxError: invalid syntax

>>> >>> complete\_df=spark.read.csv("/home/hduser/Desktop/Hackathon",sep=",",header=False,inferSchema=True)

File "<stdin>", line 1

>>> complete\_df=spark.read.csv("/home/hduser/Desktop/Hackathon",sep=",",header=False,inferSchema=True)

^

SyntaxError: invalid syntax

>>> Traceback (most recent call last):

File "<stdin>", line 1

Traceback (most recent call last):

^

SyntaxError: invalid syntax

>>> File "<stdin>", line 1, in <module>

File "<stdin>", line 1

File "<stdin>", line 1, in <module>

^

IndentationError: unexpected indent

>>> File "/usr/local/spark/python/pyspark/sql/readwriter.py", line 410, in csv

File "<stdin>", line 1

File "/usr/local/spark/python/pyspark/sql/readwriter.py", line 410, in csv

^

IndentationError: unexpected indent

>>> return self.\_df(self.\_jreader.csv(self.\_spark.\_sc.\_jvm.PythonUtils.toSeq(path)))

File "<stdin>", line 1

return self.\_df(self.\_jreader.csv(self.\_spark.\_sc.\_jvm.PythonUtils.toSeq(path)))

^

IndentationError: unexpected indent

>>> File "/usr/local/spark/python/lib/py4j-0.10.9.3-src.zip/py4j/java\_gateway.py", line 1322, in \_\_call\_\_

File "<stdin>", line 1

File "/usr/local/spark/python/lib/py4j-0.10.9.3-src.zip/py4j/java\_gateway.py", line 1322, in \_\_call\_\_

^

IndentationError: unexpected indent

>>>

>>> from pyspark.sql.types import \*

>>> schema = StructType([

... StructField("results", ArrayType(StructType([

... StructField("gender", StringType()),

... StructField("location",

... StructType([StructField("city", StringType()),StructField("state", StringType()),StructField("country", StringType()),StructField("postcode", StringType()),

... StructField("coordinates", StructType([StructField("latitude", StringType()),StructField("longitude", StringType())])),])),

... StructField("name", StructType([StructField("title", StringType()),StructField("first", StringType()),StructField("last", StringType())])),

... StructField("email", StringType()),

... StructField("login", StructType([StructField("uuid", StringType()),StructField("username", StringType())])),

... StructField("dob", StructType([StructField("date", StringType()),StructField("age", StringType())])),

... StructField("registered", StructType([StructField("date", StringType()),StructField("age", StringType())])),

...

... ])))])

>>>

>>> df\_new=spark.read.json("/home/hduser/Desktop/apidata.json",schema=schema)

Traceback (most recent call last):

File "<stdin>", line 1, in <module>

File "/usr/local/spark/python/pyspark/sql/readwriter.py", line 229, in json

return self.\_df(self.\_jreader.json(self.\_spark.\_sc.\_jvm.PythonUtils.toSeq(path)))

File "/usr/local/spark/python/lib/py4j-0.10.9.3-src.zip/py4j/java\_gateway.py", line 1322, in \_\_call\_\_

File "/usr/local/spark/python/pyspark/sql/utils.py", line 117, in deco

raise converted from None

pyspark.sql.utils.AnalysisException: Path does not exist: hdfs://localhost:54310/home/hduser/Desktop/apidata.json

>>> df\_new=spark.read.json("file:///home/hduser/Desktop/apidata.json",schema=schema)

Traceback (most recent call last):

File "<stdin>", line 1, in <module>

File "/usr/local/spark/python/pyspark/sql/readwriter.py", line 229, in json

return self.\_df(self.\_jreader.json(self.\_spark.\_sc.\_jvm.PythonUtils.toSeq(path)))

File "/usr/local/spark/python/lib/py4j-0.10.9.3-src.zip/py4j/java\_gateway.py", line 1322, in \_\_call\_\_

File "/usr/local/spark/python/pyspark/sql/utils.py", line 117, in deco

raise converted from None

pyspark.sql.utils.AnalysisException: Path does not exist: file:/home/hduser/Desktop/apidata.json

>>> df\_new=spark.read.json("file://home/hduser/Desktop/apidata.json",schema=schema)

24/07/22 17:39:26 WARN streaming.FileStreamSink: Assume no metadata directory. Error while looking for metadata directory in the path: file://home/hduser/Desktop/apidata.json.

java.lang.IllegalArgumentException: Wrong FS: file://home/hduser/Desktop/apidata.json, expected: file:///

at org.apache.hadoop.fs.FileSystem.checkPath(FileSystem.java:649)

at org.apache.hadoop.fs.RawLocalFileSystem.pathToFile(RawLocalFileSystem.java:82)

at org.apache.hadoop.fs.RawLocalFileSystem.deprecatedGetFileStatus(RawLocalFileSystem.java:606)

at org.apache.hadoop.fs.RawLocalFileSystem.getFileLinkStatusInternal(RawLocalFileSystem.java:824)

at org.apache.hadoop.fs.RawLocalFileSystem.getFileStatus(RawLocalFileSystem.java:601)

at org.apache.hadoop.fs.FilterFileSystem.getFileStatus(FilterFileSystem.java:428)

at org.apache.hadoop.fs.FileSystem.isDirectory(FileSystem.java:1439)

at org.apache.spark.sql.execution.streaming.FileStreamSink$.hasMetadata(FileStreamSink.scala:54)

at org.apache.spark.sql.execution.datasources.DataSource.resolveRelation(DataSource.scala:370)

at org.apache.spark.sql.DataFrameReader.loadV1Source(DataFrameReader.scala:274)

at org.apache.spark.sql.DataFrameReader.$anonfun$load$3(DataFrameReader.scala:245)

at scala.Option.getOrElse(Option.scala:189)

at org.apache.spark.sql.DataFrameReader.load(DataFrameReader.scala:245)

at org.apache.spark.sql.DataFrameReader.json(DataFrameReader.scala:405)

at sun.reflect.NativeMethodAccessorImpl.invoke0(Native Method)

at sun.reflect.NativeMethodAccessorImpl.invoke(NativeMethodAccessorImpl.java:62)

at sun.reflect.DelegatingMethodAccessorImpl.invoke(DelegatingMethodAccessorImpl.java:43)

at java.lang.reflect.Method.invoke(Method.java:498)

at py4j.reflection.MethodInvoker.invoke(MethodInvoker.java:244)

at py4j.reflection.ReflectionEngine.invoke(ReflectionEngine.java:357)

at py4j.Gateway.invoke(Gateway.java:282)

at py4j.commands.AbstractCommand.invokeMethod(AbstractCommand.java:132)

at py4j.commands.CallCommand.execute(CallCommand.java:79)

at py4j.ClientServerConnection.waitForCommands(ClientServerConnection.java:182)

at py4j.ClientServerConnection.run(ClientServerConnection.java:106)

at java.lang.Thread.run(Thread.java:750)

Traceback (most recent call last):

File "<stdin>", line 1, in <module>

File "/usr/local/spark/python/pyspark/sql/readwriter.py", line 229, in json

return self.\_df(self.\_jreader.json(self.\_spark.\_sc.\_jvm.PythonUtils.toSeq(path)))

File "/usr/local/spark/python/lib/py4j-0.10.9.3-src.zip/py4j/java\_gateway.py", line 1322, in \_\_call\_\_

File "/usr/local/spark/python/pyspark/sql/utils.py", line 117, in deco

raise converted from None

pyspark.sql.utils.IllegalArgumentException: Wrong FS: file://home/hduser/Desktop/apidata.json, expected: file:///

>>> df\_new=spark.read.json("file:///home/hduser/Desktop/apidata.json",schema=schema)

Traceback (most recent call last):

File "<stdin>", line 1, in <module>

File "/usr/local/spark/python/pyspark/sql/readwriter.py", line 229, in json

return self.\_df(self.\_jreader.json(self.\_spark.\_sc.\_jvm.PythonUtils.toSeq(path)))

File "/usr/local/spark/python/lib/py4j-0.10.9.3-src.zip/py4j/java\_gateway.py", line 1322, in \_\_call\_\_

File "/usr/local/spark/python/pyspark/sql/utils.py", line 117, in deco

raise converted from None

pyspark.sql.utils.AnalysisException: Path does not exist: file:/home/hduser/Desktop/apidata.json

>>> file:/home/hduser/Desktop/apidata.json

File "<stdin>", line 1

file:/home/hduser/Desktop/apidata.json

^

SyntaxError: invalid syntax

>>> df\_new=spark.read.json("/home/hduser/Desktop/Hackathon/apidata.json",schema=schema)

Traceback (most recent call last):

File "<stdin>", line 1, in <module>

File "/usr/local/spark/python/pyspark/sql/readwriter.py", line 229, in json

return self.\_df(self.\_jreader.json(self.\_spark.\_sc.\_jvm.PythonUtils.toSeq(path)))

File "/usr/local/spark/python/lib/py4j-0.10.9.3-src.zip/py4j/java\_gateway.py", line 1322, in \_\_call\_\_

File "/usr/local/spark/python/pyspark/sql/utils.py", line 117, in deco

raise converted from None

pyspark.sql.utils.AnalysisException: Path does not exist: hdfs://localhost:54310/home/hduser/Desktop/Hackathon/apidata.json

>>> df\_new=spark.read.json("file:///home/hduser/Desktop/Hackathon/apidata.json",schema=schema)

>>> df\_new.show(3)

+-------+

|results|

+-------+

| null|

| null|

| null|

+-------+

only showing top 3 rows

>>>

>>> from pyspark.sql.types import \*

>>> custom\_schema = StructType([

... StructField("results", ArrayType(StructType([

... StructField("gender", StringType()),

... StructField("location",

... StructType([StructField("city", StringType()),StructField("state", StringType()),StructField("country", StringType()),StructField("postcode", StringType()),

... StructField("coordinates", StructType([StructField("latitude", StringType()),StructField("longitude", StringType())])),])),

... StructField("name", StructType([StructField("title", StringType()),StructField("first", StringType()),StructField("last", StringType())])),

... StructField("email", StringType()),

... StructField("login", StructType([StructField("uuid", StringType()),StructField("username", StringType())])),

... StructField("dob", StructType([StructField("date", StringType()),StructField("age", StringType())])),

... StructField("registered", StructType([StructField("date", StringType()),StructField("age", StringType())])),

...

... ])))])

>>> df\_new=spark.read.json("file:///home/hduser/Desktop/Hackathon/apidata.json",schema=custom\_schema)

>>> df\_new.show(1)

+-------+

|results|

+-------+

| null|

+-------+

only showing top 1 row

>>>

>>> from pyspark.sql.types import \*

>>>

>>> schema = StructType([

... StructField("results", ArrayType(StructType([

... StructField("gender", StringType()),

... StructField("location", StructType([

... StructField("city", StringType()),

... StructField("state", StringType()),

... StructField("country", StringType()),

... StructField("postcode", StringType()),

... StructField("coordinates", StructType([

... StructField("latitude", StringType()),

... StructField("longitude", StringType())

... ]))

... ])),

... StructField("name", StructType([

... StructField("title", StringType()),

... StructField("first", StringType()),

... StructField("last", StringType())

... ])),

... StructField("email", StringType()),

... StructField("login", StructType([

... StructField("uuid", StringType()),

... StructField("username", StringType())

... ])),

... StructField("dob", StructType([

... StructField("date", StringType()),

... StructField("age", StringType())

... ])),

... StructField("registered", StructType([

... StructField("date", StringType()),

... StructField("age", StringType())

... ]))

... ]))

... ])

File "<stdin>", line 33

])

^

SyntaxError: invalid syntax

>>>

>>> from pyspark.sql.types import \*

>>>

>>> custom\_schema = StructType([

... StructField("results", ArrayType(

... StructType([

... StructField("gender", StringType()),

... StructField("location", StructType([

... StructField("city", StringType()),

... StructField("state", StringType()),

... StructField("country", StringType()),

... StructField("postcode", StringType()),

... StructField("coordinates", StructType([

... StructField("latitude", StringType()),

... StructField("longitude", StringType())

... ]))

... ])),

... StructField("name", StructType([

... StructField("title", StringType()),

... StructField("first", StringType()),

... StructField("last", StringType())

... ])),

... StructField("email", StringType()),

... StructField("login", StructType([

... StructField("uuid", StringType()),

... StructField("username", StringType())

... ])),

... StructField("dob", StructType([

... StructField("date", StringType()),

... StructField("age", StringType())

... ])),

... StructField("registered", StructType([

... StructField("date", StringType()),

... StructField("age", StringType())

... ]))

... ])

... ))

... ])

>>> df\_new=spark.read.json("file:///home/hduser/Desktop/Hackathon/apidata.json",schema=custom\_schema)

>>> df\_new.show(1)

+-------+

|results|

+-------+

| null|

+-------+

only showing top 1 row

>>> df\_new=spark.read.json("file:///home/hduser/Desktop/Hackathon/apidata.json")

>>> df\_new.show(1)

Traceback (most recent call last):

File "<stdin>", line 1, in <module>

File "/usr/local/spark/python/pyspark/sql/dataframe.py", line 494, in show

print(self.\_jdf.showString(n, 20, vertical))

File "/usr/local/spark/python/lib/py4j-0.10.9.3-src.zip/py4j/java\_gateway.py", line 1322, in \_\_call\_\_

File "/usr/local/spark/python/pyspark/sql/utils.py", line 117, in deco

raise converted from None

pyspark.sql.utils.AnalysisException:

Since Spark 2.3, the queries from raw JSON/CSV files are disallowed when the

referenced columns only include the internal corrupt record column

(named \_corrupt\_record by default). For example:

spark.read.schema(schema).csv(file).filter($"\_corrupt\_record".isNotNull).count()

and spark.read.schema(schema).csv(file).select("\_corrupt\_record").show().

Instead, you can cache or save the parsed results and then send the same query.

For example, val df = spark.read.schema(schema).csv(file).cache() and then

df.filter($"\_corrupt\_record".isNotNull).count().

>>> df\_new = spark.read.json("file:///home/hduser/Desktop/Hackathon/apidata.json", schema=custom\_schema)

>>> df\_new.show(1, truncate=False)

+-------+

|results|

+-------+

|null |

+-------+

only showing top 1 row

>>> df\_new.printSchema

<bound method DataFrame.printSchema of DataFrame[results: array<struct<gender:string,location:struct<city:string,state:string,country:string,postcode:string,coordinates:struct<latitude:string,longitude:string>>,name:struct<title:string,first:string,last:string>,email:string,login:struct<uuid:string,username:string>,dob:struct<date:string,age:string>,registered:struct<date:string,age:string>>>]>

>>>

>>> from pyspark.sql.functions import col, explode

>>>

>>> # Explode the 'results' array to flatten the DataFrame

... exploded\_df = jsondf.select(explode(col("results")).alias("exploded\_cols"))

Traceback (most recent call last):

File "<stdin>", line 2, in <module>

NameError: name 'jsondf' is not defined

>>>

>>> # Flatten the exploded DataFrame and extract the nested fields

... flattened\_df = exploded\_df \

... .withColumn('gender', col('exploded\_cols.gender')) \

... .withColumn('city', col('exploded\_cols.location.city')) \

... .withColumn('state', col('exploded\_cols.location.state')) \

... .withColumn('title', col('exploded\_cols.name.title')) \

... .withColumn('first', col('exploded\_cols.name.first')) \

... .withColumn('last', col('exploded\_cols.name.last')) \

... .withColumn('latitude', col('exploded\_cols.location.coordinates.latitude')) \

... .withColumn('longitude', col('exploded\_cols.location.coordinates.longitude')) \

... .withColumn('email', col('exploded\_cols.email')) \

... .withColumn('uuid', col('exploded\_cols.login.uuid')) \

... .withColumn('username', col('exploded\_cols.login.username')) \

... .withColumn('dob\_dt', col('exploded\_cols.dob.date')) \

... .withColumn('dob\_age', col('exploded\_cols.dob.age')) \

... .withColumn('login\_dt', col('exploded\_cols.registered.date')) \

... .withColumn('login\_age', col('exploded\_cols.registered.age')) \

... .drop('exploded\_cols')

Traceback (most recent call last):

File "<stdin>", line 2, in <module>

NameError: name 'exploded\_df' is not defined

>>>

>>> # Show the resulting flattened DataFrame

... flattened\_df.show(truncate=False)

Traceback (most recent call last):

File "<stdin>", line 2, in <module>

NameError: name 'flattened\_df' is not defined

>>>

>>> from pyspark.sql.functions import col, explode

>>>

>>> # Explode the 'results' array to flatten the DataFrame

... exploded\_df = df\_new.select(explode(col("results")).alias("exploded\_cols"))

>>>

>>> # Flatten the exploded DataFrame and extract the nested fields

... flattened\_df = exploded\_df \

... .withColumn('gender', col('exploded\_cols.gender')) \

... .withColumn('city', col('exploded\_cols.location.city')) \

... .withColumn('state', col('exploded\_cols.location.state')) \

... .withColumn('title', col('exploded\_cols.name.title')) \

... .withColumn('first', col('exploded\_cols.name.first')) \

... .withColumn('last', col('exploded\_cols.name.last')) \

... .withColumn('latitude', col('exploded\_cols.location.coordinates.latitude')) \

... .withColumn('longitude', col('exploded\_cols.location.coordinates.longitude')) \

... .withColumn('email', col('exploded\_cols.email')) \

... .withColumn('uuid', col('exploded\_cols.login.uuid')) \

... .withColumn('username', col('exploded\_cols.login.username')) \

... .withColumn('dob\_dt', col('exploded\_cols.dob.date')) \

... .withColumn('dob\_age', col('exploded\_cols.dob.age')) \

... .withColumn('login\_dt', col('exploded\_cols.registered.date')) \

... .withColumn('login\_age', col('exploded\_cols.registered.age')) \

... .drop('exploded\_cols')

>>>

>>> # Show the resulting flattened DataFrame

... flattened\_df.show(truncate=False)

+------+----+-----+-----+-----+----+--------+---------+-----+----+--------+------+-------+--------+---------+

|gender|city|state|title|first|last|latitude|longitude|email|uuid|username|dob\_dt|dob\_age|login\_dt|login\_age|

+------+----+-----+-----+-----+----+--------+---------+-----+----+--------+------+-------+--------+---------+

+------+----+-----+-----+-----+----+--------+---------+-----+----+--------+------+-------+--------+---------+

>>> gender\_count\_df = flattened\_df.groupBy('gender').agg(count('\*').alias('count'))

>>> gender\_count\_df.count()

0

>>> from pyspark.sql.functions import count

>>>

>>> # Count the number of occurrences for each gender

... gender\_count\_df = flattened\_df.groupBy('gender').agg(count('\*').alias('count'))

>>>

>>> # Show the result

... gender\_count\_df.show()

+------+-----+

|gender|count|

+------+-----+

+------+-----+

>>> flattened\_df.show(5) # Display the first 5 rows to check data

+------+----+-----+-----+-----+----+--------+---------+-----+----+--------+------+-------+--------+---------+

|gender|city|state|title|first|last|latitude|longitude|email|uuid|username|dob\_dt|dob\_age|login\_dt|login\_age|

+------+----+-----+-----+-----+----+--------+---------+-----+----+--------+------+-------+--------+---------+

+------+----+-----+-----+-----+----+--------+---------+-----+----+--------+------+-------+--------+---------+

>>>

>>> exploded\_df = df\_new.select(explode(col('results')).alias('exploded\_cols'))

>>> exploded\_df.show(truncate=False)

+-------------+

|exploded\_cols|

+-------------+

+-------------+

>>> df\_new.show(truncate=False)

+-------+

|results|

+-------+

|null |

|null |

|null |

|null |

|null |

|null |

|null |

|null |

|null |

|null |

|null |

|null |

|null |

|null |

|null |

|null |

|null |

|null |

|null |

|null |

+-------+

only showing top 20 rows

>>> c

>>> df\_raw = spark.read.json("file:///home/hduser/Desktop/Hackathon/apidata.json")

>>> df\_raw.show(truncate=False)

Traceback (most recent call last):

File "<stdin>", line 1, in <module>

File "/usr/local/spark/python/pyspark/sql/dataframe.py", line 502, in show

print(self.\_jdf.showString(n, int\_truncate, vertical))

File "/usr/local/spark/python/lib/py4j-0.10.9.3-src.zip/py4j/java\_gateway.py", line 1322, in \_\_call\_\_

File "/usr/local/spark/python/pyspark/sql/utils.py", line 117, in deco

raise converted from None

pyspark.sql.utils.AnalysisException:

Since Spark 2.3, the queries from raw JSON/CSV files are disallowed when the

referenced columns only include the internal corrupt record column

(named \_corrupt\_record by default). For example:

spark.read.schema(schema).csv(file).filter($"\_corrupt\_record".isNotNull).count()

and spark.read.schema(schema).csv(file).select("\_corrupt\_record").show().

Instead, you can cache or save the parsed results and then send the same query.

For example, val df = spark.read.schema(schema).csv(file).cache() and then

df.filter($"\_corrupt\_record".isNotNull).count().

>>>

>>> # Print schema of raw DataFrame

... df\_raw.printSchema()

root

|-- \_corrupt\_record: string (nullable = true)

>>> df\_raw = spark.read.json("file:///home/hduser/Desktop/Hackathon/apidata.json" , schema=custom\_schema)

>>> df\_r

df\_raw df\_renamed

>>> df\_raw.cache()

DataFrame[results: array<struct<gender:string,location:struct<city:string,state:string,country:string,postcode:string,coordinates:struct<latitude:string,longitude:string>>,name:struct<title:string,first:string,last:string>,email:string,login:struct<uuid:string,username:string>,dob:struct<date:string,age:string>,registered:struct<date:string,age:string>>>]

>>> df\_raw.show(truncate=False)

+-------+

|results|

+-------+

|null |

|null |

|null |

|null |

|null |

|null |

|null |

|null |

|null |

|null |

|null |

|null |

|null |

|null |

|null |

|null |

|null |

|null |

|null |

|null |

+-------+

only showing top 20 rows

>>>

>>> exploded\_df = df\_new.selectExpr('explode(results) as exploded\_cols') flattened\_df = exploded\_df \

File "<stdin>", line 1

exploded\_df = df\_new.selectExpr('explode(results) as exploded\_cols') flattened\_df = exploded\_df \

^

SyntaxError: invalid syntax

>>> .withColumn('gender', col('exploded\_cols.gender')) \

File "<stdin>", line 1

.withColumn('gender', col('exploded\_cols.gender')) \

^

IndentationError: unexpected indent

>>> .withColumn('city',col('exploded\_cols.location.city'))\

File "<stdin>", line 1

.withColumn('city',col('exploded\_cols.location.city'))\

^

IndentationError: unexpected indent

>>> .withColumn('state',col('exploded\_cols.location.state'))\

File "<stdin>", line 1

.withColumn('state',col('exploded\_cols.location.state'))\

^

IndentationError: unexpected indent

>>> .withColumn('title', col('exploded\_cols.name.title')) \

File "<stdin>", line 1

.withColumn('title', col('exploded\_cols.name.title')) \

^

IndentationError: unexpected indent

>>> .withColumn('first',col('exploded\_cols.name.first'))\

File "<stdin>", line 1

.withColumn('first',col('exploded\_cols.name.first'))\

^

IndentationError: unexpected indent

>>> .withColumn('last',col('exploded\_cols.name.last'))\

File "<stdin>", line 1

.withColumn('last',col('exploded\_cols.name.last'))\

^

IndentationError: unexpected indent

>>> .withColumn('latitude',col('exploded\_cols.location.coordinates.latitude'))\

File "<stdin>", line 1

.withColumn('latitude',col('exploded\_cols.location.coordinates.latitude'))\

^

IndentationError: unexpected indent

>>> .withColumn('longitude',col('exploded\_cols.location.coordinates.longitude'))\

File "<stdin>", line 1

.withColumn('longitude',col('exploded\_cols.location.coordinates.longitude'))\

^

IndentationError: unexpected indent

>>> .withColumn('email', col('exploded\_cols.email')) \

File "<stdin>", line 1

.withColumn('email', col('exploded\_cols.email')) \

^

IndentationError: unexpected indent

>>> .withColumn('uuid',col('exploded\_cols.login.uuid'))\

File "<stdin>", line 1

.withColumn('uuid',col('exploded\_cols.login.uuid'))\

^

IndentationError: unexpected indent

>>> .withColumn('username',col('exploded\_cols.login.username'))\

File "<stdin>", line 1

.withColumn('username',col('exploded\_cols.login.username'))\

^

IndentationError: unexpected indent

>>> .withColumn('dob\_dt',col('exploded\_cols.dob.date'))\

File "<stdin>", line 1

.withColumn('dob\_dt',col('exploded\_cols.dob.date'))\

^

IndentationError: unexpected indent

>>> .withColumn('dob\_age',col('exploded\_cols.dob.age'))\

File "<stdin>", line 1

.withColumn('dob\_age',col('exploded\_cols.dob.age'))\

^

IndentationError: unexpected indent

>>> .withColumn('login\_dt', col('exploded\_cols.registered.date')) \

File "<stdin>", line 1

.withColumn('login\_dt', col('exploded\_cols.registered.date')) \

^

IndentationError: unexpected indent

>>> .withColumn('login\_age',col('exploded\_cols.registered.age'))\

File "<stdin>", line 1

.withColumn('login\_age',col('exploded\_cols.registered.age'))\

^

IndentationError: unexpected indent

>>> .drop('exploded\_cols')

File "<stdin>", line 1

.drop('exploded\_cols')

^

IndentationError: unexpected indent

>>>

>>> exploded\_df = df\_new.selectExpr('explode(results) as exploded\_cols') flattened\_df = exploded\_df \

File "<stdin>", line 1

exploded\_df = df\_new.selectExpr('explode(results) as exploded\_cols') flattened\_df = exploded\_df \

^

SyntaxError: invalid syntax

>>> .withColumn('gender', col('exploded\_cols.gender')) \

File "<stdin>", line 1

.withColumn('gender', col('exploded\_cols.gender')) \

^

IndentationError: unexpected indent

>>> .withColumn('city',col('exploded\_cols.location.city'))\

File "<stdin>", line 1

.withColumn('city',col('exploded\_cols.location.city'))\

^

IndentationError: unexpected indent

>>> .withColumn('state',col('exploded\_cols.location.state'))\

File "<stdin>", line 1

.withColumn('state',col('exploded\_cols.location.state'))\

^

IndentationError: unexpected indent

>>> .withColumn('title', col('exploded\_cols.name.title')) \

File "<stdin>", line 1

.withColumn('title', col('exploded\_cols.name.title')) \

^

IndentationError: unexpected indent

>>> .withColumn('first',col('exploded\_cols.name.first'))\

File "<stdin>", line 1

.withColumn('first',col('exploded\_cols.name.first'))\

^

IndentationError: unexpected indent

>>> .withColumn('last',col('exploded\_cols.name.last'))\

File "<stdin>", line 1

.withColumn('last',col('exploded\_cols.name.last'))\

^

IndentationError: unexpected indent

>>> .withColumn('latitude',col('exploded\_cols.location.coordinates.latitude'))\ .withColumn('longitude',col('exploded\_cols.location.coordinates.longitude'))\

File "<stdin>", line 1

.withColumn('latitude',col('exploded\_cols.location.coordinates.latitude'))\ .withColumn('longitude',col('exploded\_cols.location.coordinates.longitude'))\

^

IndentationError: unexpected indent

>>> .withColumn('email', col('exploded\_cols.email')) \

File "<stdin>", line 1

.withColumn('email', col('exploded\_cols.email')) \

^

IndentationError: unexpected indent

>>> .withColumn('uuid',col('exploded\_cols.login.uuid'))\

File "<stdin>", line 1

.withColumn('uuid',col('exploded\_cols.login.uuid'))\

^

IndentationError: unexpected indent

>>> .withColumn('username',col('exploded\_cols.login.username'))\

File "<stdin>", line 1

.withColumn('username',col('exploded\_cols.login.username'))\

^

IndentationError: unexpected indent

>>> .withColumn('dob\_dt',col('exploded\_cols.dob.date'))\

File "<stdin>", line 1

.withColumn('dob\_dt',col('exploded\_cols.dob.date'))\

^

IndentationError: unexpected indent

>>> .withColumn('dob\_age',col('exploded\_cols.dob.age'))\

File "<stdin>", line 1

.withColumn('dob\_age',col('exploded\_cols.dob.age'))\

^

IndentationError: unexpected indent

>>> .withColumn('login\_dt', col('exploded\_cols.registered.date')) \

File "<stdin>", line 1

.withColumn('login\_dt', col('exploded\_cols.registered.date')) \

^

IndentationError: unexpected indent

>>> .withColumn('login\_age',col('exploded\_cols.registered.age'))\

File "<stdin>", line 1

.withColumn('login\_age',col('exploded\_cols.registered.age'))\

^

IndentationError: unexpected indent

>>> .drop('exploded\_cols')

File "<stdin>", line 1

.drop('exploded\_cols')

^

IndentationError: unexpected indent

>>>

>>> exploded\_df = df\_new.selectExpr('explode(results) as exploded\_cols')

>>>

>>> flattened\_df = exploded\_df \

... .withColumn('gender', col('exploded\_cols.gender')) \

... .withColumn('city', col('exploded\_cols.location.city')) \

... .withColumn('state', col('exploded\_cols.location.state')) \

... .withColumn('title', col('exploded\_cols.name.title')) \

... .withColumn('first', col('exploded\_cols.name.first')) \

... .withColumn('last', col('exploded\_cols.name.last')) \

... .withColumn('latitude', col('exploded\_cols.location.coordinates.latitude')) \

... .withColumn('longitude', col('exploded\_cols.location.coordinates.longitude')) \

... .withColumn('email', col('exploded\_cols.email')) \

... .withColumn('uuid', col('exploded\_cols.login.uuid')) \

... .withColumn('username', col('exploded\_cols.login.username')) \

... .withColumn('dob\_dt', col('exploded\_cols.dob.date')) \

... .withColumn('dob\_age', col('exploded\_cols.dob.age')) \

... .withColumn('login\_dt', col('exploded\_cols.registered.date')) \

... .withColumn('login\_age', col('exploded\_cols.registered.age')) \

... .drop('exploded\_cols')

>>> flattened\_df.show(3)

+------+----+-----+-----+-----+----+--------+---------+-----+----+--------+------+-------+--------+---------+

|gender|city|state|title|first|last|latitude|longitude|email|uuid|username|dob\_dt|dob\_age|login\_dt|login\_age|

+------+----+-----+-----+-----+----+--------+---------+-----+----+--------+------+-------+--------+---------+

+------+----+-----+-----+-----+----+--------+---------+-----+----+--------+------+-------+--------+---------+

>>> df\_raw = spark.read.json("file:///home/hduser/Desktop/Hackathon/apidata.json")

>>> df\_raw.show(truncate=False)

Traceback (most recent call last):

File "<stdin>", line 1, in <module>

File "/usr/local/spark/python/pyspark/sql/dataframe.py", line 502, in show

print(self.\_jdf.showString(n, int\_truncate, vertical))

File "/usr/local/spark/python/lib/py4j-0.10.9.3-src.zip/py4j/java\_gateway.py", line 1322, in \_\_call\_\_

File "/usr/local/spark/python/pyspark/sql/utils.py", line 117, in deco

raise converted from None

pyspark.sql.utils.AnalysisException:

Since Spark 2.3, the queries from raw JSON/CSV files are disallowed when the

referenced columns only include the internal corrupt record column

(named \_corrupt\_record by default). For example:

spark.read.schema(schema).csv(file).filter($"\_corrupt\_record".isNotNull).count()

and spark.read.schema(schema).csv(file).select("\_corrupt\_record").show().

Instead, you can cache or save the parsed results and then send the same query.

For example, val df = spark.read.schema(schema).csv(file).cache() and then

df.filter($"\_corrupt\_record".isNotNull).count().

>>> df\_raw.printSchema()

root

|-- \_corrupt\_record: string (nullable = true)

>>>

>>> # Read the JSON file

... df\_raw = spark.read.json("file:///home/hduser/Desktop/Hackathon/apidata.json")

>>>

>>> # Show the raw DataFrame

... df\_raw.show(truncate=False)

Traceback (most recent call last):

File "<stdin>", line 2, in <module>

File "/usr/local/spark/python/pyspark/sql/dataframe.py", line 502, in show

print(self.\_jdf.showString(n, int\_truncate, vertical))

File "/usr/local/spark/python/lib/py4j-0.10.9.3-src.zip/py4j/java\_gateway.py", line 1322, in \_\_call\_\_

File "/usr/local/spark/python/pyspark/sql/utils.py", line 117, in deco

raise converted from None

pyspark.sql.utils.AnalysisException:

Since Spark 2.3, the queries from raw JSON/CSV files are disallowed when the

referenced columns only include the internal corrupt record column

(named \_corrupt\_record by default). For example:

spark.read.schema(schema).csv(file).filter($"\_corrupt\_record".isNotNull).count()

and spark.read.schema(schema).csv(file).select("\_corrupt\_record").show().

Instead, you can cache or save the parsed results and then send the same query.

For example, val df = spark.read.schema(schema).csv(file).cache() and then

df.filter($"\_corrupt\_record".isNotNull).count().

>>> df\_raw = spark.read.json("file:///home/hduser/Desktop/Hackathon/apidata.json").cache()

>>> df\_raw.printSchema()

root

|-- \_corrupt\_record: string (nullable = true)

>>> df\_raw = spark.read.option("multiLine", "true").option("mode", "PERMISSIVE").json("file:///home/hduser/Desktop/Hackathon/apidata.json")

>>>

>>> df\_raw.printSchema()

root

|-- info: struct (nullable = true)

| |-- page: long (nullable = true)

| |-- results: long (nullable = true)

| |-- seed: string (nullable = true)

| |-- version: string (nullable = true)

|-- results: array (nullable = true)

| |-- element: struct (containsNull = true)

| | |-- cell: string (nullable = true)

| | |-- dob: struct (nullable = true)

| | | |-- age: long (nullable = true)

| | | |-- date: string (nullable = true)

| | |-- email: string (nullable = true)

| | |-- gender: string (nullable = true)

| | |-- id: struct (nullable = true)

| | | |-- name: string (nullable = true)

| | | |-- value: string (nullable = true)

| | |-- location: struct (nullable = true)

| | | |-- city: string (nullable = true)

| | | |-- coordinates: struct (nullable = true)

| | | | |-- latitude: string (nullable = true)

| | | | |-- longitude: string (nullable = true)

| | | |-- country: string (nullable = true)

| | | |-- postcode: string (nullable = true)

| | | |-- state: string (nullable = true)

| | | |-- street: struct (nullable = true)

| | | | |-- name: string (nullable = true)

| | | | |-- number: long (nullable = true)

| | | |-- timezone: struct (nullable = true)

| | | | |-- description: string (nullable = true)

| | | | |-- offset: string (nullable = true)

| | |-- login: struct (nullable = true)

| | | |-- md5: string (nullable = true)

| | | |-- password: string (nullable = true)

| | | |-- salt: string (nullable = true)

| | | |-- sha1: string (nullable = true)

| | | |-- sha256: string (nullable = true)

| | | |-- username: string (nullable = true)

| | | |-- uuid: string (nullable = true)

| | |-- name: struct (nullable = true)

| | | |-- first: string (nullable = true)

| | | |-- last: string (nullable = true)

| | | |-- title: string (nullable = true)

| | |-- nat: string (nullable = true)

| | |-- phone: string (nullable = true)

| | |-- picture: struct (nullable = true)

| | | |-- large: string (nullable = true)

| | | |-- medium: string (nullable = true)

| | | |-- thumbnail: string (nullable = true)

| | |-- registered: struct (nullable = true)

| | | |-- age: long (nullable = true)

| | | |-- date: string (nullable = true)

>>> exploded\_df = df\_raw.selectExpr('explode(results) as exploded\_cols')

>>>

>>> exploded\_df = df\_raw.selectExpr('explode(results) as exploded\_cols')

>>> flattened\_df = exploded\_df \

... .withColumn('gender', col('exploded\_cols.gender')) \

... .withColumn('city',col('exploded\_cols.location.city'))\

... .withColumn('state',col('exploded\_cols.location.state'))\

... .withColumn('title', col('exploded\_cols.name.title')) \

... .withColumn('first',col('exploded\_cols.name.first'))\

... .withColumn('last',col('exploded\_cols.name.last'))\

... .withColumn('latitude',col('exploded\_cols.location.coordinates.latitude'))\ .withColumn('longitude',col('exploded\_cols.location.coordinates.longitude'))\

File "<stdin>", line 8

.withColumn('latitude',col('exploded\_cols.location.coordinates.latitude'))\ .withColumn('longitude',col('exploded\_cols.location.coordinates.longitude'))\

^

SyntaxError: unexpected character after line continuation character

>>> .withColumn('email', col('exploded\_cols.email')) \

File "<stdin>", line 1

.withColumn('email', col('exploded\_cols.email')) \

^

IndentationError: unexpected indent

>>> .withColumn('uuid',col('exploded\_cols.login.uuid'))\

File "<stdin>", line 1

.withColumn('uuid',col('exploded\_cols.login.uuid'))\

^

IndentationError: unexpected indent

>>> .withColumn('username',col('exploded\_cols.login.username'))\

File "<stdin>", line 1

.withColumn('username',col('exploded\_cols.login.username'))\

^

IndentationError: unexpected indent

>>> .withColumn('dob\_dt',col('exploded\_cols.dob.date'))\

File "<stdin>", line 1

.withColumn('dob\_dt',col('exploded\_cols.dob.date'))\

^

IndentationError: unexpected indent

>>> .withColumn('dob\_age',col('exploded\_cols.dob.age'))\

File "<stdin>", line 1

.withColumn('dob\_age',col('exploded\_cols.dob.age'))\

^

IndentationError: unexpected indent

>>> .withColumn('login\_dt', col('exploded\_cols.registered.date')) \

File "<stdin>", line 1

.withColumn('login\_dt', col('exploded\_cols.registered.date')) \

^

IndentationError: unexpected indent

>>> .withColumn('login\_age',col('exploded\_cols.registered.age'))\

File "<stdin>", line 1

.withColumn('login\_age',col('exploded\_cols.registered.age'))\

^

IndentationError: unexpected indent

>>> .drop('exploded\_cols')

File "<stdin>", line 1

.drop('exploded\_cols')

^

IndentationError: unexpected indent

>>> exploded\_df = df\_raw.selectExpr('explode(results) as exploded\_cols')

>>> exploded\_df = df\_raw.selectExpr('explode(results) as exploded\_cols')

>>> flattened\_df = exploded\_df \

... .withColumn('gender', col('exploded\_cols.gender')) \

... .withColumn('city', col('exploded\_cols.location.city')) \

... .withColumn('state', col('exploded\_cols.location.state')) \

... .withColumn('title', col('exploded\_cols.name.title')) \

... .withColumn('first', col('exploded\_cols.name.first')) \

... .withColumn('last', col('exploded\_cols.name.last')) \

... .withColumn('latitude', col('exploded\_cols.location.coordinates.latitude')) \

... .withColumn('longitude', col('exploded\_cols.location.coordinates.longitude')) \

... .withColumn('email', col('exploded\_cols.email')) \

... .withColumn('uuid', col('exploded\_cols.login.uuid')) \

... .withColumn('username', col('exploded\_cols.login.username')) \

... .withColumn('dob\_dt', col('exploded\_cols.dob.date')) \

... .withColumn('dob\_age', col('exploded\_cols.dob.age')) \

... .withColumn('login\_dt', col('exploded\_cols.registered.date')) \

... .withColumn('login\_age', col('exploded\_cols.registered.age')) \

... .drop('exploded\_cols')

>>>

>>> fl

flatten( flattened\_df float( floor(

>>> flattened\_df.show(2)

+------+--------------+--------+-----+-----+---------+--------+---------+--------------------+--------------------+-------------+--------------------+-------+--------------------+---------+

|gender| city| state|title|first| last|latitude|longitude| email| uuid| username| dob\_dt|dob\_age| login\_dt|login\_age|

+------+--------------+--------+-----+-----+---------+--------+---------+--------------------+--------------------+-------------+--------------------+-------+--------------------+---------+

|female|Carrickmacross|Longford| Miss|Becky|Carpenter| 19.0157| 56.4959|becky.carpenter@e...|21312f38-cdf4-4fe...| bluefrog953|1961-12-27T07:26:...| 58|2003-02-07T11:28:...| 16|

|female| Bjerka|Rogaland| Miss|Freja| Hestetun| 49.8515| 166.0684|freja.hestetun@ex...|553a7a09-9b0b-41e...|browntiger686|1947-03-07T14:10:...| 76|2012-09-10T03:07:...| 7|

+------+--------------+--------+-----+-----+---------+--------+---------+--------------------+--------------------+-------------+--------------------+-------+--------------------+---------+

only showing top 2 rows

>>> flatten

flatten( flattened\_df

>>> flattened\_df.createOrReplaceTempView("view1")

>>> spark.sql("select gender,count(\*) from view1 group by gender ").show(1)

+------+--------+

|gender|count(1)|

+------+--------+

| male| 1|

+------+--------+

only showing top 1 row

>>> spark.sql("select gender,count(\*) from view1 group by gender ").show(2)

+------+--------+

|gender|count(1)|

+------+--------+

| male| 1|

|female| 3|

+------+--------+

>>> spark.sql("select gender,count(\*) from view1 group by gender ").show(5)

+------+--------+

|gender|count(1)|

+------+--------+

| male| 1|

|female| 3|

+------+--------+

>>> spark.sql("select gender,avg(dob\_age) from view1 group by gender ").show(5)

+------+-----------------+

|gender| avg(dob\_age)|

+------+-----------------+

| male| 47.0|

|female|66.33333333333333|

+------+-----------------+

>>> spark.sql("select date(dob\_dt) from view1 group by gender ").show(5)

Traceback (most recent call last):

File "<stdin>", line 1, in <module>

File "/usr/local/spark/python/pyspark/sql/session.py", line 723, in sql

return DataFrame(self.\_jsparkSession.sql(sqlQuery), self.\_wrapped)

File "/usr/local/spark/python/lib/py4j-0.10.9.3-src.zip/py4j/java\_gateway.py", line 1322, in \_\_call\_\_

File "/usr/local/spark/python/pyspark/sql/utils.py", line 117, in deco

raise converted from None

pyspark.sql.utils.AnalysisException: expression 'view1.dob\_dt' is neither present in the group by, nor is it an aggregate function. Add to group by or wrap in first() (or first\_value) if you don't care which value you get.;

Aggregate [gender#62602], [cast(dob\_dt#62701 as date) AS dob\_dt#62920]

+- SubqueryAlias view1

+- View (`view1`, [gender#62602,city#62606,state#62611,title#62617,first#62624,last#62632,latitude#62641,longitude#62651,email#62662,uuid#62674,username#62687,dob\_dt#62701,dob\_age#62716L,login\_dt#62732,login\_age#62749L])

+- Project [gender#62602, city#62606, state#62611, title#62617, first#62624, last#62632, latitude#62641, longitude#62651, email#62662, uuid#62674, username#62687, dob\_dt#62701, dob\_age#62716L, login\_dt#62732, login\_age#62749L]

+- Project [exploded\_cols#62600, gender#62602, city#62606, state#62611, title#62617, first#62624, last#62632, latitude#62641, longitude#62651, email#62662, uuid#62674, username#62687, dob\_dt#62701, dob\_age#62716L, login\_dt#62732, exploded\_cols#62600.registered.age AS login\_age#62749L]

+- Project [exploded\_cols#62600, gender#62602, city#62606, state#62611, title#62617, first#62624, last#62632, latitude#62641, longitude#62651, email#62662, uuid#62674, username#62687, dob\_dt#62701, dob\_age#62716L, exploded\_cols#62600.registered.date AS login\_dt#62732]

+- Project [exploded\_cols#62600, gender#62602, city#62606, state#62611, title#62617, first#62624, last#62632, latitude#62641, longitude#62651, email#62662, uuid#62674, username#62687, dob\_dt#62701, exploded\_cols#62600.dob.age AS dob\_age#62716L]

+- Project [exploded\_cols#62600, gender#62602, city#62606, state#62611, title#62617, first#62624, last#62632, latitude#62641, longitude#62651, email#62662, uuid#62674, username#62687, exploded\_cols#62600.dob.date AS dob\_dt#62701]

+- Project [exploded\_cols#62600, gender#62602, city#62606, state#62611, title#62617, first#62624, last#62632, latitude#62641, longitude#62651, email#62662, uuid#62674, exploded\_cols#62600.login.username AS username#62687]

+- Project [exploded\_cols#62600, gender#62602, city#62606, state#62611, title#62617, first#62624, last#62632, latitude#62641, longitude#62651, email#62662, exploded\_cols#62600.login.uuid AS uuid#62674]

+- Project [exploded\_cols#62600, gender#62602, city#62606, state#62611, title#62617, first#62624, last#62632, latitude#62641, longitude#62651, exploded\_cols#62600.email AS email#62662]

+- Project [exploded\_cols#62600, gender#62602, city#62606, state#62611, title#62617, first#62624, last#62632, latitude#62641, exploded\_cols#62600.location.coordinates.longitude AS longitude#62651]

+- Project [exploded\_cols#62600, gender#62602, city#62606, state#62611, title#62617, first#62624, last#62632, exploded\_cols#62600.location.coordinates.latitude AS latitude#62641]

+- Project [exploded\_cols#62600, gender#62602, city#62606, state#62611, title#62617, first#62624, exploded\_cols#62600.name.last AS last#62632]

+- Project [exploded\_cols#62600, gender#62602, city#62606, state#62611, title#62617, exploded\_cols#62600.name.first AS first#62624]

+- Project [exploded\_cols#62600, gender#62602, city#62606, state#62611, exploded\_cols#62600.name.title AS title#62617]

+- Project [exploded\_cols#62600, gender#62602, city#62606, exploded\_cols#62600.location.state AS state#62611]

+- Project [exploded\_cols#62600, gender#62602, exploded\_cols#62600.location.city AS city#62606]

+- Project [exploded\_cols#62600, exploded\_cols#62600.gender AS gender#62602]

+- Project [exploded\_cols#62600]

+- Generate explode(results#62590), false, [exploded\_cols#62600]

+- Relation [info#62589,results#62590] json

>>> flattened\_df = flattened\_df.withColumn('dob\_date\_only', date\_format('dob\_dt', 'yyyy-MM-dd'))

>>>

>>> flattened\_df = flattened\_df.withColumn('dob\_date\_only', date\_format('dob\_dt', 'yyyy-MM-dd'))

>>> spark.sql("select date(dob\_dt) from view1 group by gender ").show(5)

Traceback (most recent call last):

File "<stdin>", line 1, in <module>

File "/usr/local/spark/python/pyspark/sql/session.py", line 723, in sql

return DataFrame(self.\_jsparkSession.sql(sqlQuery), self.\_wrapped)

File "/usr/local/spark/python/lib/py4j-0.10.9.3-src.zip/py4j/java\_gateway.py", line 1322, in \_\_call\_\_

File "/usr/local/spark/python/pyspark/sql/utils.py", line 117, in deco

raise converted from None

pyspark.sql.utils.AnalysisException: expression 'view1.dob\_dt' is neither present in the group by, nor is it an aggregate function. Add to group by or wrap in first() (or first\_value) if you don't care which value you get.;

Aggregate [gender#62602], [cast(dob\_dt#62701 as date) AS dob\_dt#62955]

+- SubqueryAlias view1

+- View (`view1`, [gender#62602,city#62606,state#62611,title#62617,first#62624,last#62632,latitude#62641,longitude#62651,email#62662,uuid#62674,username#62687,dob\_dt#62701,dob\_age#62716L,login\_dt#62732,login\_age#62749L])

+- Project [gender#62602, city#62606, state#62611, title#62617, first#62624, last#62632, latitude#62641, longitude#62651, email#62662, uuid#62674, username#62687, dob\_dt#62701, dob\_age#62716L, login\_dt#62732, login\_age#62749L]

+- Project [exploded\_cols#62600, gender#62602, city#62606, state#62611, title#62617, first#62624, last#62632, latitude#62641, longitude#62651, email#62662, uuid#62674, username#62687, dob\_dt#62701, dob\_age#62716L, login\_dt#62732, exploded\_cols#62600.registered.age AS login\_age#62749L]

+- Project [exploded\_cols#62600, gender#62602, city#62606, state#62611, title#62617, first#62624, last#62632, latitude#62641, longitude#62651, email#62662, uuid#62674, username#62687, dob\_dt#62701, dob\_age#62716L, exploded\_cols#62600.registered.date AS login\_dt#62732]

+- Project [exploded\_cols#62600, gender#62602, city#62606, state#62611, title#62617, first#62624, last#62632, latitude#62641, longitude#62651, email#62662, uuid#62674, username#62687, dob\_dt#62701, exploded\_cols#62600.dob.age AS dob\_age#62716L]

+- Project [exploded\_cols#62600, gender#62602, city#62606, state#62611, title#62617, first#62624, last#62632, latitude#62641, longitude#62651, email#62662, uuid#62674, username#62687, exploded\_cols#62600.dob.date AS dob\_dt#62701]

+- Project [exploded\_cols#62600, gender#62602, city#62606, state#62611, title#62617, first#62624, last#62632, latitude#62641, longitude#62651, email#62662, uuid#62674, exploded\_cols#62600.login.username AS username#62687]

+- Project [exploded\_cols#62600, gender#62602, city#62606, state#62611, title#62617, first#62624, last#62632, latitude#62641, longitude#62651, email#62662, exploded\_cols#62600.login.uuid AS uuid#62674]

+- Project [exploded\_cols#62600, gender#62602, city#62606, state#62611, title#62617, first#62624, last#62632, latitude#62641, longitude#62651, exploded\_cols#62600.email AS email#62662]

+- Project [exploded\_cols#62600, gender#62602, city#62606, state#62611, title#62617, first#62624, last#62632, latitude#62641, exploded\_cols#62600.location.coordinates.longitude AS longitude#62651]

+- Project [exploded\_cols#62600, gender#62602, city#62606, state#62611, title#62617, first#62624, last#62632, exploded\_cols#62600.location.coordinates.latitude AS latitude#62641]

+- Project [exploded\_cols#62600, gender#62602, city#62606, state#62611, title#62617, first#62624, exploded\_cols#62600.name.last AS last#62632]

+- Project [exploded\_cols#62600, gender#62602, city#62606, state#62611, title#62617, exploded\_cols#62600.name.first AS first#62624]

+- Project [exploded\_cols#62600, gender#62602, city#62606, state#62611, exploded\_cols#62600.name.title AS title#62617]

+- Project [exploded\_cols#62600, gender#62602, city#62606, exploded\_cols#62600.location.state AS state#62611]

+- Project [exploded\_cols#62600, gender#62602, exploded\_cols#62600.location.city AS city#62606]

+- Project [exploded\_cols#62600, exploded\_cols#62600.gender AS gender#62602]

+- Project [exploded\_cols#62600]

+- Generate explode(results#62590), false, [exploded\_cols#62600]

+- Relation [info#62589,results#62590] json

>>> flattened\_df = flattened\_df.withColumn('dob\_date\_only', date\_format('dob\_dt', 'yyyy-MM-dd'))

>>>

>>> flattened\_df = flattened\_df.withColumn('dob\_date\_only', date\_format('dob\_dt', 'yyyy-MM-dd'))

>>> spark.catalog.dropTempView("view1")

>>> flattened\_df = df\_raw.withColumn('dob\_date\_only', date\_format('dob\_dt', 'yyyy-MM-dd'))

Traceback (most recent call last):

File "<stdin>", line 1, in <module>

File "/usr/local/spark/python/pyspark/sql/dataframe.py", line 2478, in withColumn

return DataFrame(self.\_jdf.withColumn(colName, col.\_jc), self.sql\_ctx)

File "/usr/local/spark/python/lib/py4j-0.10.9.3-src.zip/py4j/java\_gateway.py", line 1322, in \_\_call\_\_

File "/usr/local/spark/python/pyspark/sql/utils.py", line 117, in deco

raise converted from None

pyspark.sql.utils.AnalysisException: cannot resolve 'dob\_dt' given input columns: [info, results];

'Project [info#62589, results#62590, date\_format('dob\_dt, yyyy-MM-dd, Some(Asia/Kolkata)) AS dob\_date\_only#62990]

+- Relation [info#62589,results#62590] json

>>> flattened\_df = flattened\_df.withColumn('dob\_date\_only', date\_format('dob\_dt', 'yyyy-MM-dd'))

>>> flattened\_df = flattened\_df.withColumn('dob\_date\_only', date\_format('dob\_dt', 'yyyy-MM-dd'))

>>> flattened\_df.createOrReplaceTempView("view1")

>>> spark.sql("select date(dob\_dt) from view1 group by gender ").show(5)

Traceback (most recent call last):

File "<stdin>", line 1, in <module>

File "/usr/local/spark/python/pyspark/sql/session.py", line 723, in sql

return DataFrame(self.\_jsparkSession.sql(sqlQuery), self.\_wrapped)

File "/usr/local/spark/python/lib/py4j-0.10.9.3-src.zip/py4j/java\_gateway.py", line 1322, in \_\_call\_\_

File "/usr/local/spark/python/pyspark/sql/utils.py", line 117, in deco

raise converted from None

pyspark.sql.utils.AnalysisException: expression 'view1.dob\_dt' is neither present in the group by, nor is it an aggregate function. Add to group by or wrap in first() (or first\_value) if you don't care which value you get.;

Aggregate [gender#62602], [cast(dob\_dt#62701 as date) AS dob\_dt#63025]

+- SubqueryAlias view1

+- View (`view1`, [gender#62602,city#62606,state#62611,title#62617,first#62624,last#62632,latitude#62641,longitude#62651,email#62662,uuid#62674,username#62687,dob\_dt#62701,dob\_age#62716L,login\_dt#62732,login\_age#62749L,dob\_date\_only#63008])

+- Project [gender#62602, city#62606, state#62611, title#62617, first#62624, last#62632, latitude#62641, longitude#62651, email#62662, uuid#62674, username#62687, dob\_dt#62701, dob\_age#62716L, login\_dt#62732, login\_age#62749L, date\_format(cast(dob\_dt#62701 as timestamp), yyyy-MM-dd, Some(Asia/Kolkata)) AS dob\_date\_only#63008]

+- Project [gender#62602, city#62606, state#62611, title#62617, first#62624, last#62632, latitude#62641, longitude#62651, email#62662, uuid#62674, username#62687, dob\_dt#62701, dob\_age#62716L, login\_dt#62732, login\_age#62749L, date\_format(cast(dob\_dt#62701 as timestamp), yyyy-MM-dd, Some(Asia/Kolkata)) AS dob\_date\_only#62991]

+- Project [gender#62602, city#62606, state#62611, title#62617, first#62624, last#62632, latitude#62641, longitude#62651, email#62662, uuid#62674, username#62687, dob\_dt#62701, dob\_age#62716L, login\_dt#62732, login\_age#62749L, date\_format(cast(dob\_dt#62701 as timestamp), yyyy-MM-dd, Some(Asia/Kolkata)) AS dob\_date\_only#62973]

+- Project [gender#62602, city#62606, state#62611, title#62617, first#62624, last#62632, latitude#62641, longitude#62651, email#62662, uuid#62674, username#62687, dob\_dt#62701, dob\_age#62716L, login\_dt#62732, login\_age#62749L, date\_format(cast(dob\_dt#62701 as timestamp), yyyy-MM-dd, Some(Asia/Kolkata)) AS dob\_date\_only#62956]

+- Project [gender#62602, city#62606, state#62611, title#62617, first#62624, last#62632, latitude#62641, longitude#62651, email#62662, uuid#62674, username#62687, dob\_dt#62701, dob\_age#62716L, login\_dt#62732, login\_age#62749L, date\_format(cast(dob\_dt#62701 as timestamp), yyyy-MM-dd, Some(Asia/Kolkata)) AS dob\_date\_only#62938]

+- Project [gender#62602, city#62606, state#62611, title#62617, first#62624, last#62632, latitude#62641, longitude#62651, email#62662, uuid#62674, username#62687, dob\_dt#62701, dob\_age#62716L, login\_dt#62732, login\_age#62749L, date\_format(cast(dob\_dt#62701 as timestamp), yyyy-MM-dd, Some(Asia/Kolkata)) AS dob\_date\_only#62921]

+- Project [gender#62602, city#62606, state#62611, title#62617, first#62624, last#62632, latitude#62641, longitude#62651, email#62662, uuid#62674, username#62687, dob\_dt#62701, dob\_age#62716L, login\_dt#62732, login\_age#62749L]

+- Project [exploded\_cols#62600, gender#62602, city#62606, state#62611, title#62617, first#62624, last#62632, latitude#62641, longitude#62651, email#62662, uuid#62674, username#62687, dob\_dt#62701, dob\_age#62716L, login\_dt#62732, exploded\_cols#62600.registered.age AS login\_age#62749L]

+- Project [exploded\_cols#62600, gender#62602, city#62606, state#62611, title#62617, first#62624, last#62632, latitude#62641, longitude#62651, email#62662, uuid#62674, username#62687, dob\_dt#62701, dob\_age#62716L, exploded\_cols#62600.registered.date AS login\_dt#62732]

+- Project [exploded\_cols#62600, gender#62602, city#62606, state#62611, title#62617, first#62624, last#62632, latitude#62641, longitude#62651, email#62662, uuid#62674, username#62687, dob\_dt#62701, exploded\_cols#62600.dob.age AS dob\_age#62716L]

+- Project [exploded\_cols#62600, gender#62602, city#62606, state#62611, title#62617, first#62624, last#62632, latitude#62641, longitude#62651, email#62662, uuid#62674, username#62687, exploded\_cols#62600.dob.date AS dob\_dt#62701]

+- Project [exploded\_cols#62600, gender#62602, city#62606, state#62611, title#62617, first#62624, last#62632, latitude#62641, longitude#62651, email#62662, uuid#62674, exploded\_cols#62600.login.username AS username#62687]

+- Project [exploded\_cols#62600, gender#62602, city#62606, state#62611, title#62617, first#62624, last#62632, latitude#62641, longitude#62651, email#62662, exploded\_cols#62600.login.uuid AS uuid#62674]

+- Project [exploded\_cols#62600, gender#62602, city#62606, state#62611, title#62617, first#62624, last#62632, latitude#62641, longitude#62651, exploded\_cols#62600.email AS email#62662]

+- Project [exploded\_cols#62600, gender#62602, city#62606, state#62611, title#62617, first#62624, last#62632, latitude#62641, exploded\_cols#62600.location.coordinates.longitude AS longitude#62651]

+- Project [exploded\_cols#62600, gender#62602, city#62606, state#62611, title#62617, first#62624, last#62632, exploded\_cols#62600.location.coordinates.latitude AS latitude#62641]

+- Project [exploded\_cols#62600, gender#62602, city#62606, state#62611, title#62617, first#62624, exploded\_cols#62600.name.last AS last#62632]

+- Project [exploded\_cols#62600, gender#62602, city#62606, state#62611, title#62617, exploded\_cols#62600.name.first AS first#62624]

+- Project [exploded\_cols#62600, gender#62602, city#62606, state#62611, exploded\_cols#62600.name.title AS title#62617]

+- Project [exploded\_cols#62600, gender#62602, city#62606, exploded\_cols#62600.location.state AS state#62611]

+- Project [exploded\_cols#62600, gender#62602, exploded\_cols#62600.location.city AS city#62606]

+- Project [exploded\_cols#62600, exploded\_cols#62600.gender AS gender#62602]

+- Project [exploded\_cols#62600]

+- Generate explode(results#62590), false, [exploded\_cols#62600]

+- Relation [info#62589,results#62590] json

>>> flattened\_df.show(5)

+------+--------------+-----------+-----+------+---------+--------+---------+--------------------+--------------------+--------------+--------------------+-------+--------------------+---------+-------------+

|gender| city| state|title| first| last|latitude|longitude| email| uuid| username| dob\_dt|dob\_age| login\_dt|login\_age|dob\_date\_only|

+------+--------------+-----------+-----+------+---------+--------+---------+--------------------+--------------------+--------------+--------------------+-------+--------------------+---------+-------------+

|female|Carrickmacross| Longford| Miss| Becky|Carpenter| 19.0157| 56.4959|becky.carpenter@e...|21312f38-cdf4-4fe...| bluefrog953|1961-12-27T07:26:...| 58|2003-02-07T11:28:...| 16| 1961-12-27|

|female| Bjerka| Rogaland| Miss| Freja| Hestetun| 49.8515| 166.0684|freja.hestetun@ex...|553a7a09-9b0b-41e...| browntiger686|1947-03-07T14:10:...| 76|2012-09-10T03:07:...| 7| 1947-03-07|

| male| Roslev|Hovedstaden| Mr|Alfred| Møller|-36.9245| -0.9917|alfred.moller@exa...|18bb2a51-14d3-47d...|orangegoose716|1975-08-23T13:17:...| 47|2014-04-06T14:09:...| 5| 1975-08-23|

|female| Trabzon| Elazığ| Miss| Oya| Ilıcalı| 72.4704| 66.0216|oya.ilicali@examp...|c74c81c4-8100-487...| redwolf988|1957-11-16T21:46:...| 65|2012-05-11T14:17:...| 7| 1957-11-17|

+------+--------------+-----------+-----+------+---------+--------+---------+--------------------+--------------------+--------------+--------------------+-------+--------------------+---------+-------------+

>>> spark.sql("select date(dob\_date\_only) from view1 group by gender ").show(5)

Traceback (most recent call last):

File "<stdin>", line 1, in <module>

File "/usr/local/spark/python/pyspark/sql/session.py", line 723, in sql

return DataFrame(self.\_jsparkSession.sql(sqlQuery), self.\_wrapped)

File "/usr/local/spark/python/lib/py4j-0.10.9.3-src.zip/py4j/java\_gateway.py", line 1322, in \_\_call\_\_

File "/usr/local/spark/python/pyspark/sql/utils.py", line 117, in deco

raise converted from None

pyspark.sql.utils.AnalysisException: expression 'view1.dob\_date\_only' is neither present in the group by, nor is it an aggregate function. Add to group by or wrap in first() (or first\_value) if you don't care which value you get.;

Aggregate [gender#62602], [cast(dob\_date\_only#63008 as date) AS dob\_date\_only#63092]

+- SubqueryAlias view1

+- View (`view1`, [gender#62602,city#62606,state#62611,title#62617,first#62624,last#62632,latitude#62641,longitude#62651,email#62662,uuid#62674,username#62687,dob\_dt#62701,dob\_age#62716L,login\_dt#62732,login\_age#62749L,dob\_date\_only#63008])

+- Project [gender#62602, city#62606, state#62611, title#62617, first#62624, last#62632, latitude#62641, longitude#62651, email#62662, uuid#62674, username#62687, dob\_dt#62701, dob\_age#62716L, login\_dt#62732, login\_age#62749L, date\_format(cast(dob\_dt#62701 as timestamp), yyyy-MM-dd, Some(Asia/Kolkata)) AS dob\_date\_only#63008]

+- Project [gender#62602, city#62606, state#62611, title#62617, first#62624, last#62632, latitude#62641, longitude#62651, email#62662, uuid#62674, username#62687, dob\_dt#62701, dob\_age#62716L, login\_dt#62732, login\_age#62749L, date\_format(cast(dob\_dt#62701 as timestamp), yyyy-MM-dd, Some(Asia/Kolkata)) AS dob\_date\_only#62991]

+- Project [gender#62602, city#62606, state#62611, title#62617, first#62624, last#62632, latitude#62641, longitude#62651, email#62662, uuid#62674, username#62687, dob\_dt#62701, dob\_age#62716L, login\_dt#62732, login\_age#62749L, date\_format(cast(dob\_dt#62701 as timestamp), yyyy-MM-dd, Some(Asia/Kolkata)) AS dob\_date\_only#62973]

+- Project [gender#62602, city#62606, state#62611, title#62617, first#62624, last#62632, latitude#62641, longitude#62651, email#62662, uuid#62674, username#62687, dob\_dt#62701, dob\_age#62716L, login\_dt#62732, login\_age#62749L, date\_format(cast(dob\_dt#62701 as timestamp), yyyy-MM-dd, Some(Asia/Kolkata)) AS dob\_date\_only#62956]

+- Project [gender#62602, city#62606, state#62611, title#62617, first#62624, last#62632, latitude#62641, longitude#62651, email#62662, uuid#62674, username#62687, dob\_dt#62701, dob\_age#62716L, login\_dt#62732, login\_age#62749L, date\_format(cast(dob\_dt#62701 as timestamp), yyyy-MM-dd, Some(Asia/Kolkata)) AS dob\_date\_only#62938]

+- Project [gender#62602, city#62606, state#62611, title#62617, first#62624, last#62632, latitude#62641, longitude#62651, email#62662, uuid#62674, username#62687, dob\_dt#62701, dob\_age#62716L, login\_dt#62732, login\_age#62749L, date\_format(cast(dob\_dt#62701 as timestamp), yyyy-MM-dd, Some(Asia/Kolkata)) AS dob\_date\_only#62921]

+- Project [gender#62602, city#62606, state#62611, title#62617, first#62624, last#62632, latitude#62641, longitude#62651, email#62662, uuid#62674, username#62687, dob\_dt#62701, dob\_age#62716L, login\_dt#62732, login\_age#62749L]

+- Project [exploded\_cols#62600, gender#62602, city#62606, state#62611, title#62617, first#62624, last#62632, latitude#62641, longitude#62651, email#62662, uuid#62674, username#62687, dob\_dt#62701, dob\_age#62716L, login\_dt#62732, exploded\_cols#62600.registered.age AS login\_age#62749L]

+- Project [exploded\_cols#62600, gender#62602, city#62606, state#62611, title#62617, first#62624, last#62632, latitude#62641, longitude#62651, email#62662, uuid#62674, username#62687, dob\_dt#62701, dob\_age#62716L, exploded\_cols#62600.registered.date AS login\_dt#62732]

+- Project [exploded\_cols#62600, gender#62602, city#62606, state#62611, title#62617, first#62624, last#62632, latitude#62641, longitude#62651, email#62662, uuid#62674, username#62687, dob\_dt#62701, exploded\_cols#62600.dob.age AS dob\_age#62716L]

+- Project [exploded\_cols#62600, gender#62602, city#62606, state#62611, title#62617, first#62624, last#62632, latitude#62641, longitude#62651, email#62662, uuid#62674, username#62687, exploded\_cols#62600.dob.date AS dob\_dt#62701]

+- Project [exploded\_cols#62600, gender#62602, city#62606, state#62611, title#62617, first#62624, last#62632, latitude#62641, longitude#62651, email#62662, uuid#62674, exploded\_cols#62600.login.username AS username#62687]

+- Project [exploded\_cols#62600, gender#62602, city#62606, state#62611, title#62617, first#62624, last#62632, latitude#62641, longitude#62651, email#62662, exploded\_cols#62600.login.uuid AS uuid#62674]

+- Project [exploded\_cols#62600, gender#62602, city#62606, state#62611, title#62617, first#62624, last#62632, latitude#62641, longitude#62651, exploded\_cols#62600.email AS email#62662]

+- Project [exploded\_cols#62600, gender#62602, city#62606, state#62611, title#62617, first#62624, last#62632, latitude#62641, exploded\_cols#62600.location.coordinates.longitude AS longitude#62651]

+- Project [exploded\_cols#62600, gender#62602, city#62606, state#62611, title#62617, first#62624, last#62632, exploded\_cols#62600.location.coordinates.latitude AS latitude#62641]

+- Project [exploded\_cols#62600, gender#62602, city#62606, state#62611, title#62617, first#62624, exploded\_cols#62600.name.last AS last#62632]

+- Project [exploded\_cols#62600, gender#62602, city#62606, state#62611, title#62617, exploded\_cols#62600.name.first AS first#62624]

+- Project [exploded\_cols#62600, gender#62602, city#62606, state#62611, exploded\_cols#62600.name.title AS title#62617]

+- Project [exploded\_cols#62600, gender#62602, city#62606, exploded\_cols#62600.location.state AS state#62611]

+- Project [exploded\_cols#62600, gender#62602, exploded\_cols#62600.location.city AS city#62606]

+- Project [exploded\_cols#62600, exploded\_cols#62600.gender AS gender#62602]

+- Project [exploded\_cols#62600]

+- Generate explode(results#62590), false, [exploded\_cols#62600]

+- Relation [info#62589,results#62590] json

>>> flattened\_df.show(5)

>>> flattened\_df.show(5)

+------+--------------+-----------+-----+------+---------+--------+---------+--------------------+--------------------+--------------+--------------------+-------+--------------------+---------+-------------+

|gender| city| state|title| first| last|latitude|longitude| email| uuid| username| dob\_dt|dob\_age| login\_dt|login\_age|dob\_date\_only|

+------+--------------+-----------+-----+------+---------+--------+---------+--------------------+--------------------+--------------+--------------------+-------+--------------------+---------+-------------+

|female|Carrickmacross| Longford| Miss| Becky|Carpenter| 19.0157| 56.4959|becky.carpenter@e...|21312f38-cdf4-4fe...| bluefrog953|1961-12-27T07:26:...| 58|2003-02-07T11:28:...| 16| 1961-12-27|

|female| Bjerka| Rogaland| Miss| Freja| Hestetun| 49.8515| 166.0684|freja.hestetun@ex...|553a7a09-9b0b-41e...| browntiger686|1947-03-07T14:10:...| 76|2012-09-10T03:07:...| 7| 1947-03-07|

| male| Roslev|Hovedstaden| Mr|Alfred| Møller|-36.9245| -0.9917|alfred.moller@exa...|18bb2a51-14d3-47d...|orangegoose716|1975-08-23T13:17:...| 47|2014-04-06T14:09:...| 5| 1975-08-23|

|female| Trabzon| Elazığ| Miss| Oya| Ilıcalı| 72.4704| 66.0216|oya.ilicali@examp...|c74c81c4-8100-487...| redwolf988|1957-11-16T21:46:...| 65|2012-05-11T14:17:...| 7| 1957-11-17|

+------+--------------+-----------+-----+------+---------+--------+---------+--------------------+--------------------+--------------+--------------------+-------+--------------------+---------+-------------+

>>> spark.sql("select \* from view1").show(5)

+------+--------------+-----------+-----+------+---------+--------+---------+--------------------+--------------------+--------------+--------------------+-------+--------------------+---------+-------------+

|gender| city| state|title| first| last|latitude|longitude| email| uuid| username| dob\_dt|dob\_age| login\_dt|login\_age|dob\_date\_only|

+------+--------------+-----------+-----+------+---------+--------+---------+--------------------+--------------------+--------------+--------------------+-------+--------------------+---------+-------------+

|female|Carrickmacross| Longford| Miss| Becky|Carpenter| 19.0157| 56.4959|becky.carpenter@e...|21312f38-cdf4-4fe...| bluefrog953|1961-12-27T07:26:...| 58|2003-02-07T11:28:...| 16| 1961-12-27|

|female| Bjerka| Rogaland| Miss| Freja| Hestetun| 49.8515| 166.0684|freja.hestetun@ex...|553a7a09-9b0b-41e...| browntiger686|1947-03-07T14:10:...| 76|2012-09-10T03:07:...| 7| 1947-03-07|

| male| Roslev|Hovedstaden| Mr|Alfred| Møller|-36.9245| -0.9917|alfred.moller@exa...|18bb2a51-14d3-47d...|orangegoose716|1975-08-23T13:17:...| 47|2014-04-06T14:09:...| 5| 1975-08-23|

|female| Trabzon| Elazığ| Miss| Oya| Ilıcalı| 72.4704| 66.0216|oya.ilicali@examp...|c74c81c4-8100-487...| redwolf988|1957-11-16T21:46:...| 65|2012-05-11T14:17:...| 7| 1957-11-17|

+------+--------------+-----------+-----+------+---------+--------+---------+--------------------+--------------------+--------------+--------------------+-------+--------------------+---------+-------------+

>>> spark.sql("""

... SELECT

... gender,

... DATE(dob\_dt) as dob\_date\_only

... FROM view1

... GROUP BY gender, dob\_date\_only

... """).show(5, truncate=False)

Traceback (most recent call last):

File "<stdin>", line 7, in <module>

File "/usr/local/spark/python/pyspark/sql/session.py", line 723, in sql

return DataFrame(self.\_jsparkSession.sql(sqlQuery), self.\_wrapped)

File "/usr/local/spark/python/lib/py4j-0.10.9.3-src.zip/py4j/java\_gateway.py", line 1322, in \_\_call\_\_

File "/usr/local/spark/python/pyspark/sql/utils.py", line 117, in deco

raise converted from None

pyspark.sql.utils.AnalysisException: expression 'view1.dob\_dt' is neither present in the group by, nor is it an aggregate function. Add to group by or wrap in first() (or first\_value) if you don't care which value you get.;

Aggregate [gender#62602, dob\_date\_only#63008], [gender#62602, cast(dob\_dt#62701 as date) AS dob\_date\_only#63241]

+- SubqueryAlias view1

+- View (`view1`, [gender#62602,city#62606,state#62611,title#62617,first#62624,last#62632,latitude#62641,longitude#62651,email#62662,uuid#62674,username#62687,dob\_dt#62701,dob\_age#62716L,login\_dt#62732,login\_age#62749L,dob\_date\_only#63008])

+- Project [gender#62602, city#62606, state#62611, title#62617, first#62624, last#62632, latitude#62641, longitude#62651, email#62662, uuid#62674, username#62687, dob\_dt#62701, dob\_age#62716L, login\_dt#62732, login\_age#62749L, date\_format(cast(dob\_dt#62701 as timestamp), yyyy-MM-dd, Some(Asia/Kolkata)) AS dob\_date\_only#63008]

+- Project [gender#62602, city#62606, state#62611, title#62617, first#62624, last#62632, latitude#62641, longitude#62651, email#62662, uuid#62674, username#62687, dob\_dt#62701, dob\_age#62716L, login\_dt#62732, login\_age#62749L, date\_format(cast(dob\_dt#62701 as timestamp), yyyy-MM-dd, Some(Asia/Kolkata)) AS dob\_date\_only#62991]

+- Project [gender#62602, city#62606, state#62611, title#62617, first#62624, last#62632, latitude#62641, longitude#62651, email#62662, uuid#62674, username#62687, dob\_dt#62701, dob\_age#62716L, login\_dt#62732, login\_age#62749L, date\_format(cast(dob\_dt#62701 as timestamp), yyyy-MM-dd, Some(Asia/Kolkata)) AS dob\_date\_only#62973]

+- Project [gender#62602, city#62606, state#62611, title#62617, first#62624, last#62632, latitude#62641, longitude#62651, email#62662, uuid#62674, username#62687, dob\_dt#62701, dob\_age#62716L, login\_dt#62732, login\_age#62749L, date\_format(cast(dob\_dt#62701 as timestamp), yyyy-MM-dd, Some(Asia/Kolkata)) AS dob\_date\_only#62956]

+- Project [gender#62602, city#62606, state#62611, title#62617, first#62624, last#62632, latitude#62641, longitude#62651, email#62662, uuid#62674, username#62687, dob\_dt#62701, dob\_age#62716L, login\_dt#62732, login\_age#62749L, date\_format(cast(dob\_dt#62701 as timestamp), yyyy-MM-dd, Some(Asia/Kolkata)) AS dob\_date\_only#62938]

+- Project [gender#62602, city#62606, state#62611, title#62617, first#62624, last#62632, latitude#62641, longitude#62651, email#62662, uuid#62674, username#62687, dob\_dt#62701, dob\_age#62716L, login\_dt#62732, login\_age#62749L, date\_format(cast(dob\_dt#62701 as timestamp), yyyy-MM-dd, Some(Asia/Kolkata)) AS dob\_date\_only#62921]

+- Project [gender#62602, city#62606, state#62611, title#62617, first#62624, last#62632, latitude#62641, longitude#62651, email#62662, uuid#62674, username#62687, dob\_dt#62701, dob\_age#62716L, login\_dt#62732, login\_age#62749L]

+- Project [exploded\_cols#62600, gender#62602, city#62606, state#62611, title#62617, first#62624, last#62632, latitude#62641, longitude#62651, email#62662, uuid#62674, username#62687, dob\_dt#62701, dob\_age#62716L, login\_dt#62732, exploded\_cols#62600.registered.age AS login\_age#62749L]

+- Project [exploded\_cols#62600, gender#62602, city#62606, state#62611, title#62617, first#62624, last#62632, latitude#62641, longitude#62651, email#62662, uuid#62674, username#62687, dob\_dt#62701, dob\_age#62716L, exploded\_cols#62600.registered.date AS login\_dt#62732]

+- Project [exploded\_cols#62600, gender#62602, city#62606, state#62611, title#62617, first#62624, last#62632, latitude#62641, longitude#62651, email#62662, uuid#62674, username#62687, dob\_dt#62701, exploded\_cols#62600.dob.age AS dob\_age#62716L]

+- Project [exploded\_cols#62600, gender#62602, city#62606, state#62611, title#62617, first#62624, last#62632, latitude#62641, longitude#62651, email#62662, uuid#62674, username#62687, exploded\_cols#62600.dob.date AS dob\_dt#62701]

+- Project [exploded\_cols#62600, gender#62602, city#62606, state#62611, title#62617, first#62624, last#62632, latitude#62641, longitude#62651, email#62662, uuid#62674, exploded\_cols#62600.login.username AS username#62687]

+- Project [exploded\_cols#62600, gender#62602, city#62606, state#62611, title#62617, first#62624, last#62632, latitude#62641, longitude#62651, email#62662, exploded\_cols#62600.login.uuid AS uuid#62674]

+- Project [exploded\_cols#62600, gender#62602, city#62606, state#62611, title#62617, first#62624, last#62632, latitude#62641, longitude#62651, exploded\_cols#62600.email AS email#62662]

+- Project [exploded\_cols#62600, gender#62602, city#62606, state#62611, title#62617, first#62624, last#62632, latitude#62641, exploded\_cols#62600.location.coordinates.longitude AS longitude#62651]

+- Project [exploded\_cols#62600, gender#62602, city#62606, state#62611, title#62617, first#62624, last#62632, exploded\_cols#62600.location.coordinates.latitude AS latitude#62641]

+- Project [exploded\_cols#62600, gender#62602, city#62606, state#62611, title#62617, first#62624, exploded\_cols#62600.name.last AS last#62632]

+- Project [exploded\_cols#62600, gender#62602, city#62606, state#62611, title#62617, exploded\_cols#62600.name.first AS first#62624]

+- Project [exploded\_cols#62600, gender#62602, city#62606, state#62611, exploded\_cols#62600.name.title AS title#62617]

+- Project [exploded\_cols#62600, gender#62602, city#62606, exploded\_cols#62600.location.state AS state#62611]

+- Project [exploded\_cols#62600, gender#62602, exploded\_cols#62600.location.city AS city#62606]

+- Project [exploded\_cols#62600, exploded\_cols#62600.gender AS gender#62602]

+- Project [exploded\_cols#62600]

+- Generate explode(results#62590), false, [exploded\_cols#62600]

+- Relation [info#62589,results#62590] json

>>> spark.sql("select date(dob\_date\_only) from view1 ").show(5)

+-------------+

|dob\_date\_only|

+-------------+

| 1961-12-27|

| 1947-03-07|

| 1975-08-23|

| 1957-11-17|

+-------------+

>>> spark.sql("select date(dob\_dt) from view1 ").show(5)

+----------+

| dob\_dt|

+----------+

|1961-12-27|

|1947-03-07|

|1975-08-23|

|1957-11-16|

+----------+

>>> df = flattened\_df.withColumn('password\_md5', md5(col('password')))

Traceback (most recent call last):

File "<stdin>", line 1, in <module>

File "/usr/local/spark/python/pyspark/sql/dataframe.py", line 2478, in withColumn

return DataFrame(self.\_jdf.withColumn(colName, col.\_jc), self.sql\_ctx)

File "/usr/local/spark/python/lib/py4j-0.10.9.3-src.zip/py4j/java\_gateway.py", line 1322, in \_\_call\_\_

File "/usr/local/spark/python/pyspark/sql/utils.py", line 117, in deco

raise converted from None

pyspark.sql.utils.AnalysisException: cannot resolve 'password' given input columns: [city, dob\_age, dob\_date\_only, dob\_dt, email, first, gender, last, latitude, login\_age, login\_dt, longitude, state, title, username, uuid];

'Project [gender#62602, city#62606, state#62611, title#62617, first#62624, last#62632, latitude#62641, longitude#62651, email#62662, uuid#62674, username#62687, dob\_dt#62701, dob\_age#62716L, login\_dt#62732, login\_age#62749L, dob\_date\_only#63008, md5('password) AS password\_md5#63262]

+- Project [gender#62602, city#62606, state#62611, title#62617, first#62624, last#62632, latitude#62641, longitude#62651, email#62662, uuid#62674, username#62687, dob\_dt#62701, dob\_age#62716L, login\_dt#62732, login\_age#62749L, date\_format(cast(dob\_dt#62701 as timestamp), yyyy-MM-dd, Some(Asia/Kolkata)) AS dob\_date\_only#63008]

+- Project [gender#62602, city#62606, state#62611, title#62617, first#62624, last#62632, latitude#62641, longitude#62651, email#62662, uuid#62674, username#62687, dob\_dt#62701, dob\_age#62716L, login\_dt#62732, login\_age#62749L, date\_format(cast(dob\_dt#62701 as timestamp), yyyy-MM-dd, Some(Asia/Kolkata)) AS dob\_date\_only#62991]

+- Project [gender#62602, city#62606, state#62611, title#62617, first#62624, last#62632, latitude#62641, longitude#62651, email#62662, uuid#62674, username#62687, dob\_dt#62701, dob\_age#62716L, login\_dt#62732, login\_age#62749L, date\_format(cast(dob\_dt#62701 as timestamp), yyyy-MM-dd, Some(Asia/Kolkata)) AS dob\_date\_only#62973]

+- Project [gender#62602, city#62606, state#62611, title#62617, first#62624, last#62632, latitude#62641, longitude#62651, email#62662, uuid#62674, username#62687, dob\_dt#62701, dob\_age#62716L, login\_dt#62732, login\_age#62749L, date\_format(cast(dob\_dt#62701 as timestamp), yyyy-MM-dd, Some(Asia/Kolkata)) AS dob\_date\_only#62956]

+- Project [gender#62602, city#62606, state#62611, title#62617, first#62624, last#62632, latitude#62641, longitude#62651, email#62662, uuid#62674, username#62687, dob\_dt#62701, dob\_age#62716L, login\_dt#62732, login\_age#62749L, date\_format(cast(dob\_dt#62701 as timestamp), yyyy-MM-dd, Some(Asia/Kolkata)) AS dob\_date\_only#62938]

+- Project [gender#62602, city#62606, state#62611, title#62617, first#62624, last#62632, latitude#62641, longitude#62651, email#62662, uuid#62674, username#62687, dob\_dt#62701, dob\_age#62716L, login\_dt#62732, login\_age#62749L, date\_format(cast(dob\_dt#62701 as timestamp), yyyy-MM-dd, Some(Asia/Kolkata)) AS dob\_date\_only#62921]

+- Project [gender#62602, city#62606, state#62611, title#62617, first#62624, last#62632, latitude#62641, longitude#62651, email#62662, uuid#62674, username#62687, dob\_dt#62701, dob\_age#62716L, login\_dt#62732, login\_age#62749L]

+- Project [exploded\_cols#62600, gender#62602, city#62606, state#62611, title#62617, first#62624, last#62632, latitude#62641, longitude#62651, email#62662, uuid#62674, username#62687, dob\_dt#62701, dob\_age#62716L, login\_dt#62732, exploded\_cols#62600.registered.age AS login\_age#62749L]

+- Project [exploded\_cols#62600, gender#62602, city#62606, state#62611, title#62617, first#62624, last#62632, latitude#62641, longitude#62651, email#62662, uuid#62674, username#62687, dob\_dt#62701, dob\_age#62716L, exploded\_cols#62600.registered.date AS login\_dt#62732]

+- Project [exploded\_cols#62600, gender#62602, city#62606, state#62611, title#62617, first#62624, last#62632, latitude#62641, longitude#62651, email#62662, uuid#62674, username#62687, dob\_dt#62701, exploded\_cols#62600.dob.age AS dob\_age#62716L]

+- Project [exploded\_cols#62600, gender#62602, city#62606, state#62611, title#62617, first#62624, last#62632, latitude#62641, longitude#62651, email#62662, uuid#62674, username#62687, exploded\_cols#62600.dob.date AS dob\_dt#62701]

+- Project [exploded\_cols#62600, gender#62602, city#62606, state#62611, title#62617, first#62624, last#62632, latitude#62641, longitude#62651, email#62662, uuid#62674, exploded\_cols#62600.login.username AS username#62687]

+- Project [exploded\_cols#62600, gender#62602, city#62606, state#62611, title#62617, first#62624, last#62632, latitude#62641, longitude#62651, email#62662, exploded\_cols#62600.login.uuid AS uuid#62674]

+- Project [exploded\_cols#62600, gender#62602, city#62606, state#62611, title#62617, first#62624, last#62632, latitude#62641, longitude#62651, exploded\_cols#62600.email AS email#62662]

+- Project [exploded\_cols#62600, gender#62602, city#62606, state#62611, title#62617, first#62624, last#62632, latitude#62641, exploded\_cols#62600.location.coordinates.longitude AS longitude#62651]

+- Project [exploded\_cols#62600, gender#62602, city#62606, state#62611, title#62617, first#62624, last#62632, exploded\_cols#62600.location.coordinates.latitude AS latitude#62641]

+- Project [exploded\_cols#62600, gender#62602, city#62606, state#62611, title#62617, first#62624, exploded\_cols#62600.name.last AS last#62632]

+- Project [exploded\_cols#62600, gender#62602, city#62606, state#62611, title#62617, exploded\_cols#62600.name.first AS first#62624]

+- Project [exploded\_cols#62600, gender#62602, city#62606, state#62611, exploded\_cols#62600.name.title AS title#62617]

+- Project [exploded\_cols#62600, gender#62602, city#62606, exploded\_cols#62600.location.state AS state#62611]

+- Project [exploded\_cols#62600, gender#62602, exploded\_cols#62600.location.city AS city#62606]

+- Project [exploded\_cols#62600, exploded\_cols#62600.gender AS gender#62602]

+- Project [exploded\_cols#62600]

+- Generate explode(results#62590), false, [exploded\_cols#62600]

+- Relation [info#62589,results#62590] json

>>>

>>> flattened\_df.show(4)

+------+--------------+-----------+-----+------+---------+--------+---------+--------------------+--------------------+--------------+--------------------+-------+--------------------+---------+-------------+

|gender| city| state|title| first| last|latitude|longitude| email| uuid| username| dob\_dt|dob\_age| login\_dt|login\_age|dob\_date\_only|

+------+--------------+-----------+-----+------+---------+--------+---------+--------------------+--------------------+--------------+--------------------+-------+--------------------+---------+-------------+

|female|Carrickmacross| Longford| Miss| Becky|Carpenter| 19.0157| 56.4959|becky.carpenter@e...|21312f38-cdf4-4fe...| bluefrog953|1961-12-27T07:26:...| 58|2003-02-07T11:28:...| 16| 1961-12-27|

|female| Bjerka| Rogaland| Miss| Freja| Hestetun| 49.8515| 166.0684|freja.hestetun@ex...|553a7a09-9b0b-41e...| browntiger686|1947-03-07T14:10:...| 76|2012-09-10T03:07:...| 7| 1947-03-07|

| male| Roslev|Hovedstaden| Mr|Alfred| Møller|-36.9245| -0.9917|alfred.moller@exa...|18bb2a51-14d3-47d...|orangegoose716|1975-08-23T13:17:...| 47|2014-04-06T14:09:...| 5| 1975-08-23|

|female| Trabzon| Elazığ| Miss| Oya| Ilıcalı| 72.4704| 66.0216|oya.ilicali@examp...|c74c81c4-8100-487...| redwolf988|1957-11-16T21:46:...| 65|2012-05-11T14:17:...| 7| 1957-11-17|

+------+--------------+-----------+-----+------+---------+--------+---------+--------------------+--------------------+--------------+--------------------+-------+--------------------+---------+-------------+

>>> df = flattened\_df.withColumn('password\_md5', md5(col('uuid')))

>>> df.drop("uuid")

DataFrame[gender: string, city: string, state: string, title: string, first: string, last: string, latitude: string, longitude: string, email: string, username: string, dob\_dt: string, dob\_age: bigint, login\_dt: string, login\_age: bigint, dob\_date\_only: string, password\_md5: string]

>>> df.show(4)

+------+--------------+-----------+-----+------+---------+--------+---------+--------------------+--------------------+--------------+--------------------+-------+--------------------+---------+-------------+--------------------+

|gender| city| state|title| first| last|latitude|longitude| email| uuid| username| dob\_dt|dob\_age| login\_dt|login\_age|dob\_date\_only| password\_md5|

+------+--------------+-----------+-----+------+---------+--------+---------+--------------------+--------------------+--------------+--------------------+-------+--------------------+---------+-------------+--------------------+

|female|Carrickmacross| Longford| Miss| Becky|Carpenter| 19.0157| 56.4959|becky.carpenter@e...|21312f38-cdf4-4fe...| bluefrog953|1961-12-27T07:26:...| 58|2003-02-07T11:28:...| 16| 1961-12-27|15448ee18f5970ccf...|

|female| Bjerka| Rogaland| Miss| Freja| Hestetun| 49.8515| 166.0684|freja.hestetun@ex...|553a7a09-9b0b-41e...| browntiger686|1947-03-07T14:10:...| 76|2012-09-10T03:07:...| 7| 1947-03-07|f36987978f4f1c537...|

| male| Roslev|Hovedstaden| Mr|Alfred| Møller|-36.9245| -0.9917|alfred.moller@exa...|18bb2a51-14d3-47d...|orangegoose716|1975-08-23T13:17:...| 47|2014-04-06T14:09:...| 5| 1975-08-23|af1db29687ba8d038...|

|female| Trabzon| Elazığ| Miss| Oya| Ilıcalı| 72.4704| 66.0216|oya.ilicali@examp...|c74c81c4-8100-487...| redwolf988|1957-11-16T21:46:...| 65|2012-05-11T14:17:...| 7| 1957-11-17|829399af08f9fbb85...|

+------+--------------+-----------+-----+------+---------+--------+---------+--------------------+--------------------+--------------+--------------------+-------+--------------------+---------+-------------+--------------------+

>>> df = df.withColumn(

... 'formatted\_cell\_number',

... regexp\_replace(col('cell\_number'), '([0-9]{3})([0-9]{3})([0-9]{4})', '$1-$2-$3')

... )

Traceback (most recent call last):

File "<stdin>", line 3, in <module>

File "/usr/local/spark/python/pyspark/sql/dataframe.py", line 2478, in withColumn

return DataFrame(self.\_jdf.withColumn(colName, col.\_jc), self.sql\_ctx)

File "/usr/local/spark/python/lib/py4j-0.10.9.3-src.zip/py4j/java\_gateway.py", line 1322, in \_\_call\_\_

File "/usr/local/spark/python/pyspark/sql/utils.py", line 117, in deco

raise converted from None

pyspark.sql.utils.AnalysisException: cannot resolve 'cell\_number' given input columns: [city, dob\_age, dob\_date\_only, dob\_dt, email, first, gender, last, latitude, login\_age, login\_dt, longitude, password\_md5, state, title, username, uuid];

'Project [gender#62602, city#62606, state#62611, title#62617, first#62624, last#62632, latitude#62641, longitude#62651, email#62662, uuid#62674, username#62687, dob\_dt#62701, dob\_age#62716L, login\_dt#62732, login\_age#62749L, dob\_date\_only#63008, password\_md5#63329, regexp\_replace('cell\_number, ([0-9]{3})([0-9]{3})([0-9]{4}), $1-$2-$3, 1) AS formatted\_cell\_number#63433]

+- Project [gender#62602, city#62606, state#62611, title#62617, first#62624, last#62632, latitude#62641, longitude#62651, email#62662, uuid#62674, username#62687, dob\_dt#62701, dob\_age#62716L, login\_dt#62732, login\_age#62749L, dob\_date\_only#63008, md5(cast(uuid#62674 as binary)) AS password\_md5#63329]

+- Project [gender#62602, city#62606, state#62611, title#62617, first#62624, last#62632, latitude#62641, longitude#62651, email#62662, uuid#62674, username#62687, dob\_dt#62701, dob\_age#62716L, login\_dt#62732, login\_age#62749L, date\_format(cast(dob\_dt#62701 as timestamp), yyyy-MM-dd, Some(Asia/Kolkata)) AS dob\_date\_only#63008]

+- Project [gender#62602, city#62606, state#62611, title#62617, first#62624, last#62632, latitude#62641, longitude#62651, email#62662, uuid#62674, username#62687, dob\_dt#62701, dob\_age#62716L, login\_dt#62732, login\_age#62749L, date\_format(cast(dob\_dt#62701 as timestamp), yyyy-MM-dd, Some(Asia/Kolkata)) AS dob\_date\_only#62991]

+- Project [gender#62602, city#62606, state#62611, title#62617, first#62624, last#62632, latitude#62641, longitude#62651, email#62662, uuid#62674, username#62687, dob\_dt#62701, dob\_age#62716L, login\_dt#62732, login\_age#62749L, date\_format(cast(dob\_dt#62701 as timestamp), yyyy-MM-dd, Some(Asia/Kolkata)) AS dob\_date\_only#62973]

+- Project [gender#62602, city#62606, state#62611, title#62617, first#62624, last#62632, latitude#62641, longitude#62651, email#62662, uuid#62674, username#62687, dob\_dt#62701, dob\_age#62716L, login\_dt#62732, login\_age#62749L, date\_format(cast(dob\_dt#62701 as timestamp), yyyy-MM-dd, Some(Asia/Kolkata)) AS dob\_date\_only#62956]

+- Project [gender#62602, city#62606, state#62611, title#62617, first#62624, last#62632, latitude#62641, longitude#62651, email#62662, uuid#62674, username#62687, dob\_dt#62701, dob\_age#62716L, login\_dt#62732, login\_age#62749L, date\_format(cast(dob\_dt#62701 as timestamp), yyyy-MM-dd, Some(Asia/Kolkata)) AS dob\_date\_only#62938]

+- Project [gender#62602, city#62606, state#62611, title#62617, first#62624, last#62632, latitude#62641, longitude#62651, email#62662, uuid#62674, username#62687, dob\_dt#62701, dob\_age#62716L, login\_dt#62732, login\_age#62749L, date\_format(cast(dob\_dt#62701 as timestamp), yyyy-MM-dd, Some(Asia/Kolkata)) AS dob\_date\_only#62921]

+- Project [gender#62602, city#62606, state#62611, title#62617, first#62624, last#62632, latitude#62641, longitude#62651, email#62662, uuid#62674, username#62687, dob\_dt#62701, dob\_age#62716L, login\_dt#62732, login\_age#62749L]

+- Project [exploded\_cols#62600, gender#62602, city#62606, state#62611, title#62617, first#62624, last#62632, latitude#62641, longitude#62651, email#62662, uuid#62674, username#62687, dob\_dt#62701, dob\_age#62716L, login\_dt#62732, exploded\_cols#62600.registered.age AS login\_age#62749L]

+- Project [exploded\_cols#62600, gender#62602, city#62606, state#62611, title#62617, first#62624, last#62632, latitude#62641, longitude#62651, email#62662, uuid#62674, username#62687, dob\_dt#62701, dob\_age#62716L, exploded\_cols#62600.registered.date AS login\_dt#62732]

+- Project [exploded\_cols#62600, gender#62602, city#62606, state#62611, title#62617, first#62624, last#62632, latitude#62641, longitude#62651, email#62662, uuid#62674, username#62687, dob\_dt#62701, exploded\_cols#62600.dob.age AS dob\_age#62716L]

+- Project [exploded\_cols#62600, gender#62602, city#62606, state#62611, title#62617, first#62624, last#62632, latitude#62641, longitude#62651, email#62662, uuid#62674, username#62687, exploded\_cols#62600.dob.date AS dob\_dt#62701]

+- Project [exploded\_cols#62600, gender#62602, city#62606, state#62611, title#62617, first#62624, last#62632, latitude#62641, longitude#62651, email#62662, uuid#62674, exploded\_cols#62600.login.username AS username#62687]

+- Project [exploded\_cols#62600, gender#62602, city#62606, state#62611, title#62617, first#62624, last#62632, latitude#62641, longitude#62651, email#62662, exploded\_cols#62600.login.uuid AS uuid#62674]

+- Project [exploded\_cols#62600, gender#62602, city#62606, state#62611, title#62617, first#62624, last#62632, latitude#62641, longitude#62651, exploded\_cols#62600.email AS email#62662]

+- Project [exploded\_cols#62600, gender#62602, city#62606, state#62611, title#62617, first#62624, last#62632, latitude#62641, exploded\_cols#62600.location.coordinates.longitude AS longitude#62651]

+- Project [exploded\_cols#62600, gender#62602, city#62606, state#62611, title#62617, first#62624, last#62632, exploded\_cols#62600.location.coordinates.latitude AS latitude#62641]

+- Project [exploded\_cols#62600, gender#62602, city#62606, state#62611, title#62617, first#62624, exploded\_cols#62600.name.last AS last#62632]

+- Project [exploded\_cols#62600, gender#62602, city#62606, state#62611, title#62617, exploded\_cols#62600.name.first AS first#62624]

+- Project [exploded\_cols#62600, gender#62602, city#62606, state#62611, exploded\_cols#62600.name.title AS title#62617]

+- Project [exploded\_cols#62600, gender#62602, city#62606, exploded\_cols#62600.location.state AS state#62611]

+- Project [exploded\_cols#62600, gender#62602, exploded\_cols#62600.location.city AS city#62606]

+- Project [exploded\_cols#62600, exploded\_cols#62600.gender AS gender#62602]

+- Project [exploded\_cols#62600]

+- Generate explode(results#62590), false, [exploded\_cols#62600]

+- Relation [info#62589,results#62590] json

>>> df.show(3)

+------+--------------+-----------+-----+------+---------+--------+---------+--------------------+--------------------+--------------+--------------------+-------+--------------------+---------+-------------+--------------------+

|gender| city| state|title| first| last|latitude|longitude| email| uuid| username| dob\_dt|dob\_age| login\_dt|login\_age|dob\_date\_only| password\_md5|

+------+--------------+-----------+-----+------+---------+--------+---------+--------------------+--------------------+--------------+--------------------+-------+--------------------+---------+-------------+--------------------+

|female|Carrickmacross| Longford| Miss| Becky|Carpenter| 19.0157| 56.4959|becky.carpenter@e...|21312f38-cdf4-4fe...| bluefrog953|1961-12-27T07:26:...| 58|2003-02-07T11:28:...| 16| 1961-12-27|15448ee18f5970ccf...|

|female| Bjerka| Rogaland| Miss| Freja| Hestetun| 49.8515| 166.0684|freja.hestetun@ex...|553a7a09-9b0b-41e...| browntiger686|1947-03-07T14:10:...| 76|2012-09-10T03:07:...| 7| 1947-03-07|f36987978f4f1c537...|

| male| Roslev|Hovedstaden| Mr|Alfred| Møller|-36.9245| -0.9917|alfred.moller@exa...|18bb2a51-14d3-47d...|orangegoose716|1975-08-23T13:17:...| 47|2014-04-06T14:09:...| 5| 1975-08-23|af1db29687ba8d038...|

+------+--------------+-----------+-----+------+---------+--------+---------+--------------------+--------------------+--------------+--------------------+-------+--------------------+---------+-------------+--------------------+

only showing top 3 rows

>>>

>>> df = df.withColumn("dob", F.to\_date("dob\_date\_only", "yyyy-MM-dd"))

>>>

>>> # Calculate age from dob\_date\_only

... df = df.withColumn(

... "calculated\_age",

... F.floor(F.datediff(F.current\_date(), F.col("dob\_date\_only")) / 365.25)

... )

>>>

>>> # Compare calculated\_age with existing age

... df = df.withColumn(

... "age\_correct",

... F.col("dob\_age") == F.col("calculated\_age")

... )

>>> df.show(3)

+------+--------------+-----------+-----+------+---------+--------+---------+--------------------+--------------------+--------------+--------------------+-------+--------------------+---------+-------------+--------------------+----------+--------------+-----------+

|gender| city| state|title| first| last|latitude|longitude| email| uuid| username| dob\_dt|dob\_age| login\_dt|login\_age|dob\_date\_only| password\_md5| dob|calculated\_age|age\_correct|

+------+--------------+-----------+-----+------+---------+--------+---------+--------------------+--------------------+--------------+--------------------+-------+--------------------+---------+-------------+--------------------+----------+--------------+-----------+

|female|Carrickmacross| Longford| Miss| Becky|Carpenter| 19.0157| 56.4959|becky.carpenter@e...|21312f38-cdf4-4fe...| bluefrog953|1961-12-27T07:26:...| 58|2003-02-07T11:28:...| 16| 1961-12-27|15448ee18f5970ccf...|1961-12-27| 62| false|

|female| Bjerka| Rogaland| Miss| Freja| Hestetun| 49.8515| 166.0684|freja.hestetun@ex...|553a7a09-9b0b-41e...| browntiger686|1947-03-07T14:10:...| 76|2012-09-10T03:07:...| 7| 1947-03-07|f36987978f4f1c537...|1947-03-07| 77| false|

| male| Roslev|Hovedstaden| Mr|Alfred| Møller|-36.9245| -0.9917|alfred.moller@exa...|18bb2a51-14d3-47d...|orangegoose716|1975-08-23T13:17:...| 47|2014-04-06T14:09:...| 5| 1975-08-23|af1db29687ba8d038...|1975-08-23| 48| false|

+------+--------------+-----------+-----+------+---------+--------+---------+--------------------+--------------------+--------------+--------------------+-------+--------------------+---------+-------------+--------------------+----------+--------------+-----------+

only showing top 3 rows

>>> df.select("email").show(4)

+--------------------+

| email|

+--------------------+

|becky.carpenter@e...|

|freja.hestetun@ex...|

|alfred.moller@exa...|

|oya.ilicali@examp...|

+--------------------+

>>> df.select("email").show(4,True)

+--------------------+

| email|

+--------------------+

|becky.carpenter@e...|

|freja.hestetun@ex...|

|alfred.moller@exa...|

|oya.ilicali@examp...|

+--------------------+

>>> df.select("email").show(4,False)

+---------------------------+

|email |

+---------------------------+

|becky.carpenter@example.com|

|freja.hestetun@example.com |

|alfred.moller@example.com |

|oya.ilicali@example.com |

+---------------------------+

>>>

>>> result\_df = df.withColumn(

... "expected\_email",

... concat\_ws( lower(col('first')) +lit('.') + lower(col('last'))) + lit('@example.com')

... ).withColumn(

... "email\_status",

... when(lower(col('email')) == col('expected\_email'), 'Correct').otherwise('Incorrect')

... )

Traceback (most recent call last):

File "<stdin>", line 3, in <module>

File "/usr/local/spark/python/pyspark/sql/functions.py", line 2647, in concat\_ws

return Column(sc.\_jvm.functions.concat\_ws(sep, \_to\_seq(sc, cols, \_to\_java\_column)))

File "/usr/local/spark/python/lib/py4j-0.10.9.3-src.zip/py4j/java\_gateway.py", line 1313, in \_\_call\_\_

File "/usr/local/spark/python/lib/py4j-0.10.9.3-src.zip/py4j/java\_gateway.py", line 1277, in \_build\_args

File "/usr/local/spark/python/lib/py4j-0.10.9.3-src.zip/py4j/java\_gateway.py", line 1264, in \_get\_args

File "/usr/local/spark/python/lib/py4j-0.10.9.3-src.zip/py4j/java\_collections.py", line 510, in convert

File "/usr/local/spark/python/pyspark/sql/column.py", line 463, in \_\_iter\_\_

raise TypeError("Column is not iterable")

TypeError: Column is not iterable

>>> result\_df = df.withColumn(

... "expected\_email",

... concat(

... lower(col('first')),

... lit('.'),

... lower(col('last')),

... lit('@example.com')

... )

... ).withColumn(

... "email\_status",

... when(lower(col('email')) == col('expected\_email'), 'Correct').otherwise('Incorrect')

... )

>>> result\_df.show(4)

+------+--------------+-----------+-----+------+---------+--------+---------+--------------------+--------------------+--------------+--------------------+-------+--------------------+---------+-------------+--------------------+----------+--------------+-----------+--------------------+------------+

|gender| city| state|title| first| last|latitude|longitude| email| uuid| username| dob\_dt|dob\_age| login\_dt|login\_age|dob\_date\_only| password\_md5| dob|calculated\_age|age\_correct| expected\_email|email\_status|

+------+--------------+-----------+-----+------+---------+--------+---------+--------------------+--------------------+--------------+--------------------+-------+--------------------+---------+-------------+--------------------+----------+--------------+-----------+--------------------+------------+

|female|Carrickmacross| Longford| Miss| Becky|Carpenter| 19.0157| 56.4959|becky.carpenter@e...|21312f38-cdf4-4fe...| bluefrog953|1961-12-27T07:26:...| 58|2003-02-07T11:28:...| 16| 1961-12-27|15448ee18f5970ccf...|1961-12-27| 62| false|becky.carpenter@e...| Correct|

|female| Bjerka| Rogaland| Miss| Freja| Hestetun| 49.8515| 166.0684|freja.hestetun@ex...|553a7a09-9b0b-41e...| browntiger686|1947-03-07T14:10:...| 76|2012-09-10T03:07:...| 7| 1947-03-07|f36987978f4f1c537...|1947-03-07| 77| false|freja.hestetun@ex...| Correct|

| male| Roslev|Hovedstaden| Mr|Alfred| Møller|-36.9245| -0.9917|alfred.moller@exa...|18bb2a51-14d3-47d...|orangegoose716|1975-08-23T13:17:...| 47|2014-04-06T14:09:...| 5| 1975-08-23|af1db29687ba8d038...|1975-08-23| 48| false|alfred.møller@exa...| Incorrect|

|female| Trabzon| Elazığ| Miss| Oya| Ilıcalı| 72.4704| 66.0216|oya.ilicali@examp...|c74c81c4-8100-487...| redwolf988|1957-11-16T21:46:...| 65|2012-05-11T14:17:...| 7| 1957-11-17|829399af08f9fbb85...|1957-11-17| 66| false|oya.ilıcalı@examp...| Incorrect|

+------+--------------+-----------+-----+------+---------+--------+---------+--------------------+--------------------+--------------+--------------------+-------+--------------------+---------+-------------+--------------------+----------+--------------+-----------+--------------------+------------+

>>>

>>> result\_df\_SQL\_after\_join = result\_df\_SQL.alias("l").join(result\_df\_SQL\_1.alias("r"), on=col("l.IssuerId") == col("r.IssuerId"), how="inner")

>>> result\_df\_SQL\_after\_join = result\_df\_SQL\_after\_join.join(result\_df\_SQL\_2, on=[col("IssuerId") == col("IssuerId")], how="inner")

24/07/22 12:08:43 WARN sql.Column: Constructing trivially true equals predicate, ''IssuerId = 'IssuerId'. Perhaps you need to use aliases.

Traceback (most recent call last):

File "<stdin>", line 1, in <module>

File "/usr/local/spark/python/pyspark/sql/dataframe.py", line 1355, in join

jdf = self.\_jdf.join(other.\_jdf, on, how)

File "/usr/local/spark/python/lib/py4j-0.10.9.3-src.zip/py4j/java\_gateway.py", line 1322, in \_\_call\_\_

File "/usr/local/spark/python/pyspark/sql/utils.py", line 117, in deco

raise converted from None

pyspark.sql.utils.AnalysisException: Reference 'IssuerId' is ambiguous, could be: l.IssuerId, r.IssuerId, insureview.IssuerId.

>>> result\_df\_SQL\_after\_join = result\_df\_SQL\_after\_join.alias("l").join(result\_df\_SQL\_2.alias("r"), on=[col("l.IssuerId") == col("r.IssuerId")], how="inner")

Traceback (most recent call last):

File "<stdin>", line 1, in <module>

File "/usr/local/spark/python/pyspark/sql/dataframe.py", line 1355, in join

jdf = self.\_jdf.join(other.\_jdf, on, how)

File "/usr/local/spark/python/lib/py4j-0.10.9.3-src.zip/py4j/java\_gateway.py", line 1322, in \_\_call\_\_

File "/usr/local/spark/python/pyspark/sql/utils.py", line 117, in deco

raise converted from None

pyspark.sql.utils.AnalysisException: Reference 'l.IssuerId' is ambiguous, could be: l.IssuerId, l.IssuerId.

>>> result\_df\_SQL\_after\_join = result\_df\_SQL\_after\_join.join(result\_df\_SQL\_2, on=[col("IssuerId") == col("IssuerId")], how="inner")

24/07/22 12:09:44 WARN sql.Column: Constructing trivially true equals predicate, ''IssuerId = 'IssuerId'. Perhaps you need to use aliases.

Traceback (most recent call last):

File "<stdin>", line 1, in <module>

File "/usr/local/spark/python/pyspark/sql/dataframe.py", line 1355, in join

jdf = self.\_jdf.join(other.\_jdf, on, how)

File "/usr/local/spark/python/lib/py4j-0.10.9.3-src.zip/py4j/java\_gateway.py", line 1322, in \_\_call\_\_

File "/usr/local/spark/python/pyspark/sql/utils.py", line 117, in deco

raise converted from None

pyspark.sql.utils.AnalysisException: Reference 'IssuerId' is ambiguous, could be: l.IssuerId, r.IssuerId, insureview.IssuerId.

>>>

>>> result\_df\_SQL\_after\_join = result\_df\_SQL.alias("l").join(result\_df\_SQL\_1.alias("r"), on=col("l.IssuerId") == col("r.IssuerId"), how="inner")

>>> result\_df\_SQL\_after\_join = result\_df\_SQL\_after\_join.alias("l").join(

... result\_df\_SQL\_2.alias("r"),

... col("l.IssuerId") == col("r.IssuerId"),

... how="inner"

... )

Traceback (most recent call last):

File "<stdin>", line 4, in <module>

File "/usr/local/spark/python/pyspark/sql/dataframe.py", line 1355, in join

jdf = self.\_jdf.join(other.\_jdf, on, how)

File "/usr/local/spark/python/lib/py4j-0.10.9.3-src.zip/py4j/java\_gateway.py", line 1322, in \_\_call\_\_

File "/usr/local/spark/python/pyspark/sql/utils.py", line 117, in deco

raise converted from None

pyspark.sql.utils.AnalysisException: Reference 'l.IssuerId' is ambiguous, could be: l.IssuerId, l.IssuerId.

>>>

at org.apache.spark.util.Utils$.tryWithSafeFinally(Utils.scala:1462)

at org.apache.spark.executor.Executor$TaskRunner.run(Executor.scala:509)

at java.util.concurrent.ThreadPoolExecutor.runWorker(ThreadPoolExecutor.java:1149)

at java.util.concurrent.ThreadPoolExecutor$Worker.run(ThreadPoolExecutor.java:624)

at java.lang.Thread.run(Thread.java:750)

24/07/22 09:31:48 WARN scheduler.TaskSetManager: Lost task 0.0 in stage 105.0 (TID 116) (192.168.112.129 executor driver): org.apache.spark.api.python.PythonException: Traceback (most recent call last):

File "/usr/local/spark/python/lib/pyspark.zip/pyspark/worker.py", line 619, in main

process()

File "/usr/local/spark/python/lib/pyspark.zip/pyspark/worker.py", line 611, in process

serializer.dump\_stream(out\_iter, outfile)

File "/usr/local/spark/python/lib/pyspark.zip/pyspark/serializers.py", line 211, in dump\_stream

self.serializer.dump\_stream(self.\_batched(iterator), stream)

File "/usr/local/spark/python/lib/pyspark.zip/pyspark/serializers.py", line 132, in dump\_stream

for obj in iterator:

File "/usr/local/spark/python/lib/pyspark.zip/pyspark/serializers.py", line 200, in \_batched

for item in iterator:

File "/usr/local/spark/python/lib/pyspark.zip/pyspark/worker.py", line 452, in mapper

result = tuple(f(\*[a[o] for o in arg\_offsets]) for (arg\_offsets, f) in udfs)

File "/usr/local/spark/python/lib/pyspark.zip/pyspark/worker.py", line 452, in <genexpr>

result = tuple(f(\*[a[o] for o in arg\_offsets]) for (arg\_offsets, f) in udfs)

File "/usr/local/spark/python/lib/pyspark.zip/pyspark/worker.py", line 87, in <lambda>

return lambda \*a: f(\*a)

File "/usr/local/spark/python/lib/pyspark.zip/pyspark/util.py", line 74, in wrapper

return f(\*args, \*\*kwargs)

File "<stdin>", line 3, in clean\_string

TypeError: expected string or bytes-like object

at org.apache.spark.api.python.BasePythonRunner$ReaderIterator.handlePythonException(PythonRunner.scala:555)

at org.apache.spark.sql.execution.python.PythonUDFRunner$$anon$2.read(PythonUDFRunner.scala:86)

at org.apache.spark.sql.execution.python.PythonUDFRunner$$anon$2.read(PythonUDFRunner.scala:68)

at org.apache.spark.api.python.BasePythonRunner$ReaderIterator.hasNext(PythonRunner.scala:508)

at org.apache.spark.InterruptibleIterator.hasNext(InterruptibleIterator.scala:37)

at scala.collection.Iterator$$anon$11.hasNext(Iterator.scala:491)

at scala.collection.Iterator$$anon$10.hasNext(Iterator.scala:460)

at scala.collection.Iterator$$anon$10.hasNext(Iterator.scala:460)

at org.apache.spark.sql.catalyst.expressions.GeneratedClass$GeneratedIteratorForCodegenStage1.processNext(Unknown Source)

at org.apache.spark.sql.execution.BufferedRowIterator.hasNext(BufferedRowIterator.java:43)

at org.apache.spark.sql.execution.WholeStageCodegenExec$$anon$1.hasNext(WholeStageCodegenExec.scala:759)

at org.apache.spark.sql.execution.columnar.DefaultCachedBatchSerializer$$anon$1.next(InMemoryRelation.scala:87)

at org.apache.spark.sql.execution.columnar.DefaultCachedBatchSerializer$$anon$1.next(InMemoryRelation.scala:79)

at scala.collection.Iterator$$anon$10.next(Iterator.scala:461)

at org.apache.spark.storage.memory.MemoryStore.putIterator(MemoryStore.scala:224)

at org.apache.spark.storage.memory.MemoryStore.putIteratorAsValues(MemoryStore.scala:302)

at org.apache.spark.storage.BlockManager.$anonfun$doPutIterator$1(BlockManager.scala:1481)

at org.apache.spark.storage.BlockManager.org$apache$spark$storage$BlockManager$$doPut(BlockManager.scala:1408)

at org.apache.spark.storage.BlockManager.doPutIterator(BlockManager.scala:1472)

at org.apache.spark.storage.BlockManager.getOrElseUpdate(BlockManager.scala:1295)

at org.apache.spark.rdd.RDD.getOrCompute(RDD.scala:384)

at org.apache.spark.rdd.RDD.iterator(RDD.scala:335)

at org.apache.spark.rdd.MapPartitionsRDD.compute(MapPartitionsRDD.scala:52)

at org.apache.spark.rdd.RDD.computeOrReadCheckpoint(RDD.scala:373)

at org.apache.spark.rdd.RDD.iterator(RDD.scala:337)

at org.apache.spark.rdd.MapPartitionsRDD.compute(MapPartitionsRDD.scala:52)

at org.apache.spark.rdd.RDD.computeOrReadCheckpoint(RDD.scala:373)

at org.apache.spark.rdd.RDD.iterator(RDD.scala:337)

at org.apache.spark.rdd.MapPartitionsRDD.compute(MapPartitionsRDD.scala:52)

at org.apache.spark.rdd.RDD.computeOrReadCheckpoint(RDD.scala:373)

at org.apache.spark.rdd.RDD.iterator(RDD.scala:337)

at org.apache.spark.rdd.MapPartitionsRDD.compute(MapPartitionsRDD.scala:52)

at org.apache.spark.rdd.RDD.computeOrReadCheckpoint(RDD.scala:373)

at org.apache.spark.rdd.RDD.iterator(RDD.scala:337)

at org.apache.spark.rdd.MapPartitionsRDD.compute(MapPartitionsRDD.scala:52)

at org.apache.spark.rdd.RDD.computeOrReadCheckpoint(RDD.scala:373)

at org.apache.spark.rdd.RDD.iterator(RDD.scala:337)

at org.apache.spark.scheduler.ResultTask.runTask(ResultTask.scala:90)

at org.apache.spark.scheduler.Task.run(Task.scala:131)

at org.apache.spark.executor.Executor$TaskRunner.$anonfun$run$3(Executor.scala:506)

at org.apache.spark.util.Utils$.tryWithSafeFinally(Utils.scala:1462)

at org.apache.spark.executor.Executor$TaskRunner.run(Executor.scala:509)

at java.util.concurrent.ThreadPoolExecutor.runWorker(ThreadPoolExecutor.java:1149)

at java.util.concurrent.ThreadPoolExecutor$Worker.run(ThreadPoolExecutor.java:624)

at java.lang.Thread.run(Thread.java:750)

24/07/22 09:31:48 ERROR scheduler.TaskSetManager: Task 0 in stage 105.0 failed 1 times; aborting job

Traceback (most recent call last):

File "<stdin>", line 1, in <module>

File "/usr/local/spark/python/pyspark/sql/dataframe.py", line 494, in show

print(self.\_jdf.showString(n, 20, vertical))

File "/usr/local/spark/python/lib/py4j-0.10.9.3-src.zip/py4j/java\_gateway.py", line 1322, in \_\_call\_\_

File "/usr/local/spark/python/pyspark/sql/utils.py", line 117, in deco

raise converted from None

pyspark.sql.utils.PythonException:

An exception was thrown from the Python worker. Please see the stack trace below.

Traceback (most recent call last):

File "/usr/local/spark/python/lib/pyspark.zip/pyspark/worker.py", line 619, in main

process()

File "/usr/local/spark/python/lib/pyspark.zip/pyspark/worker.py", line 611, in process

serializer.dump\_stream(out\_iter, outfile)

File "/usr/local/spark/python/lib/pyspark.zip/pyspark/serializers.py", line 211, in dump\_stream

self.serializer.dump\_stream(self.\_batched(iterator), stream)

File "/usr/local/spark/python/lib/pyspark.zip/pyspark/serializers.py", line 132, in dump\_stream

for obj in iterator:

File "/usr/local/spark/python/lib/pyspark.zip/pyspark/serializers.py", line 200, in \_batched

for item in iterator:

File "/usr/local/spark/python/lib/pyspark.zip/pyspark/worker.py", line 452, in mapper

result = tuple(f(\*[a[o] for o in arg\_offsets]) for (arg\_offsets, f) in udfs)

File "/usr/local/spark/python/lib/pyspark.zip/pyspark/worker.py", line 452, in <genexpr>

result = tuple(f(\*[a[o] for o in arg\_offsets]) for (arg\_offsets, f) in udfs)

File "/usr/local/spark/python/lib/pyspark.zip/pyspark/worker.py", line 87, in <lambda>

return lambda \*a: f(\*a)

File "/usr/local/spark/python/lib/pyspark.zip/pyspark/util.py", line 74, in wrapper

return f(\*args, \*\*kwargs)

File "<stdin>", line 3, in clean\_string

TypeError: expected string or bytes-like object

>>> sample\_op=merged\_data\_frame.withColumn("New NetworkName",df\_function(col("NetworkName")))

>>> sample\_op.show(3)

24/07/22 09:32:00 WARN storage.BlockManager: Putting block rdd\_531\_0 failed due to exception org.apache.spark.api.python.PythonException: Traceback (most recent call last):

File "/usr/local/spark/python/lib/pyspark.zip/pyspark/worker.py", line 619, in main

process()

File "/usr/local/spark/python/lib/pyspark.zip/pyspark/worker.py", line 611, in process

serializer.dump\_stream(out\_iter, outfile)

File "/usr/local/spark/python/lib/pyspark.zip/pyspark/serializers.py", line 211, in dump\_stream

self.serializer.dump\_stream(self.\_batched(iterator), stream)

File "/usr/local/spark/python/lib/pyspark.zip/pyspark/serializers.py", line 132, in dump\_stream

for obj in iterator:

File "/usr/local/spark/python/lib/pyspark.zip/pyspark/serializers.py", line 200, in \_batched

for item in iterator:

File "/usr/local/spark/python/lib/pyspark.zip/pyspark/worker.py", line 452, in mapper

result = tuple(f(\*[a[o] for o in arg\_offsets]) for (arg\_offsets, f) in udfs)

File "/usr/local/spark/python/lib/pyspark.zip/pyspark/worker.py", line 452, in <genexpr>

result = tuple(f(\*[a[o] for o in arg\_offsets]) for (arg\_offsets, f) in udfs)

File "/usr/local/spark/python/lib/pyspark.zip/pyspark/worker.py", line 87, in <lambda>

return lambda \*a: f(\*a)

File "/usr/local/spark/python/lib/pyspark.zip/pyspark/util.py", line 74, in wrapper

return f(\*args, \*\*kwargs)

File "<stdin>", line 3, in clean\_string

TypeError: expected string or bytes-like object

.

24/07/22 09:32:00 WARN storage.BlockManager: Block rdd\_531\_0 could not be removed as it was not found on disk or in memory

24/07/22 09:32:00 ERROR executor.Executor: Exception in task 0.0 in stage 106.0 (TID 117)

org.apache.spark.api.python.PythonException: Traceback (most recent call last):

File "/usr/local/spark/python/lib/pyspark.zip/pyspark/worker.py", line 619, in main

process()

File "/usr/local/spark/python/lib/pyspark.zip/pyspark/worker.py", line 611, in process

serializer.dump\_stream(out\_iter, outfile)

File "/usr/local/spark/python/lib/pyspark.zip/pyspark/serializers.py", line 211, in dump\_stream

self.serializer.dump\_stream(self.\_batched(iterator), stream)

File "/usr/local/spark/python/lib/pyspark.zip/pyspark/serializers.py", line 132, in dump\_stream

for obj in iterator:

File "/usr/local/spark/python/lib/pyspark.zip/pyspark/serializers.py", line 200, in \_batched

for item in iterator:

File "/usr/local/spark/python/lib/pyspark.zip/pyspark/worker.py", line 452, in mapper

result = tuple(f(\*[a[o] for o in arg\_offsets]) for (arg\_offsets, f) in udfs)

File "/usr/local/spark/python/lib/pyspark.zip/pyspark/worker.py", line 452, in <genexpr>

result = tuple(f(\*[a[o] for o in arg\_offsets]) for (arg\_offsets, f) in udfs)

File "/usr/local/spark/python/lib/pyspark.zip/pyspark/worker.py", line 87, in <lambda>

return lambda \*a: f(\*a)

File "/usr/local/spark/python/lib/pyspark.zip/pyspark/util.py", line 74, in wrapper

return f(\*args, \*\*kwargs)

File "<stdin>", line 3, in clean\_string

TypeError: expected string or bytes-like object

at org.apache.spark.api.python.BasePythonRunner$ReaderIterator.handlePythonException(PythonRunner.scala:555)

at org.apache.spark.sql.execution.python.PythonUDFRunner$$anon$2.read(PythonUDFRunner.scala:86)

at org.apache.spark.sql.execution.python.PythonUDFRunner$$anon$2.read(PythonUDFRunner.scala:68)

at org.apache.spark.api.python.BasePythonRunner$ReaderIterator.hasNext(PythonRunner.scala:508)

at org.apache.spark.InterruptibleIterator.hasNext(InterruptibleIterator.scala:37)

at scala.collection.Iterator$$anon$11.hasNext(Iterator.scala:491)

at scala.collection.Iterator$$anon$10.hasNext(Iterator.scala:460)

at scala.collection.Iterator$$anon$10.hasNext(Iterator.scala:460)

at org.apache.spark.sql.catalyst.expressions.GeneratedClass$GeneratedIteratorForCodegenStage1.processNext(Unknown Source)

at org.apache.spark.sql.execution.BufferedRowIterator.hasNext(BufferedRowIterator.java:43)

at org.apache.spark.sql.execution.WholeStageCodegenExec$$anon$1.hasNext(WholeStageCodegenExec.scala:759)

at org.apache.spark.sql.execution.columnar.DefaultCachedBatchSerializer$$anon$1.next(InMemoryRelation.scala:87)

at org.apache.spark.sql.execution.columnar.DefaultCachedBatchSerializer$$anon$1.next(InMemoryRelation.scala:79)

at scala.collection.Iterator$$anon$10.next(Iterator.scala:461)

at org.apache.spark.storage.memory.MemoryStore.putIterator(MemoryStore.scala:224)

at org.apache.spark.storage.memory.MemoryStore.putIteratorAsValues(MemoryStore.scala:302)

at org.apache.spark.storage.BlockManager.$anonfun$doPutIterator$1(BlockManager.scala:1481)

at org.apache.spark.storage.BlockManager.org$apache$spark$storage$BlockManager$$doPut(BlockManager.scala:1408)

at org.apache.spark.storage.BlockManager.doPutIterator(BlockManager.scala:1472)

at org.apache.spark.storage.BlockManager.getOrElseUpdate(BlockManager.scala:1295)

at org.apache.spark.rdd.RDD.getOrCompute(RDD.scala:384)

at org.apache.spark.rdd.RDD.iterator(RDD.scala:335)

at org.apache.spark.rdd.MapPartitionsRDD.compute(MapPartitionsRDD.scala:52)

at org.apache.spark.rdd.RDD.computeOrReadCheckpoint(RDD.scala:373)

at org.apache.spark.rdd.RDD.iterator(RDD.scala:337)

at org.apache.spark.rdd.MapPartitionsRDD.compute(MapPartitionsRDD.scala:52)

at org.apache.spark.rdd.RDD.computeOrReadCheckpoint(RDD.scala:373)

at org.apache.spark.rdd.RDD.iterator(RDD.scala:337)

at org.apache.spark.rdd.MapPartitionsRDD.compute(MapPartitionsRDD.scala:52)

at org.apache.spark.rdd.RDD.computeOrReadCheckpoint(RDD.scala:373)

at org.apache.spark.rdd.RDD.iterator(RDD.scala:337)

at org.apache.spark.rdd.MapPartitionsRDD.compute(MapPartitionsRDD.scala:52)

at org.apache.spark.rdd.RDD.computeOrReadCheckpoint(RDD.scala:373)

at org.apache.spark.rdd.RDD.iterator(RDD.scala:337)

at org.apache.spark.rdd.MapPartitionsRDD.compute(MapPartitionsRDD.scala:52)

at org.apache.spark.rdd.RDD.computeOrReadCheckpoint(RDD.scala:373)

at org.apache.spark.rdd.RDD.iterator(RDD.scala:337)

at org.apache.spark.scheduler.ResultTask.runTask(ResultTask.scala:90)

at org.apache.spark.scheduler.Task.run(Task.scala:131)

at org.apache.spark.executor.Executor$TaskRunner.$anonfun$run$3(Executor.scala:506)

at org.apache.spark.util.Utils$.tryWithSafeFinally(Utils.scala:1462)

at org.apache.spark.executor.Executor$TaskRunner.run(Executor.scala:509)

at java.util.concurrent.ThreadPoolExecutor.runWorker(ThreadPoolExecutor.java:1149)

at java.util.concurrent.ThreadPoolExecutor$Worker.run(ThreadPoolExecutor.java:624)

at java.lang.Thread.run(Thread.java:750)

24/07/22 09:32:00 WARN scheduler.TaskSetManager: Lost task 0.0 in stage 106.0 (TID 117) (192.168.112.129 executor driver): org.apache.spark.api.python.PythonException: Traceback (most recent call last):

File "/usr/local/spark/python/lib/pyspark.zip/pyspark/worker.py", line 619, in main

process()

File "/usr/local/spark/python/lib/pyspark.zip/pyspark/worker.py", line 611, in process

serializer.dump\_stream(out\_iter, outfile)

File "/usr/local/spark/python/lib/pyspark.zip/pyspark/serializers.py", line 211, in dump\_stream

self.serializer.dump\_stream(self.\_batched(iterator), stream)

File "/usr/local/spark/python/lib/pyspark.zip/pyspark/serializers.py", line 132, in dump\_stream

for obj in iterator:

File "/usr/local/spark/python/lib/pyspark.zip/pyspark/serializers.py", line 200, in \_batched

for item in iterator:

File "/usr/local/spark/python/lib/pyspark.zip/pyspark/worker.py", line 452, in mapper

result = tuple(f(\*[a[o] for o in arg\_offsets]) for (arg\_offsets, f) in udfs)

File "/usr/local/spark/python/lib/pyspark.zip/pyspark/worker.py", line 452, in <genexpr>

result = tuple(f(\*[a[o] for o in arg\_offsets]) for (arg\_offsets, f) in udfs)

File "/usr/local/spark/python/lib/pyspark.zip/pyspark/worker.py", line 87, in <lambda>

return lambda \*a: f(\*a)

File "/usr/local/spark/python/lib/pyspark.zip/pyspark/util.py", line 74, in wrapper

return f(\*args, \*\*kwargs)

File "<stdin>", line 3, in clean\_string

TypeError: expected string or bytes-like object

at org.apache.spark.api.python.BasePythonRunner$ReaderIterator.handlePythonException(PythonRunner.scala:555)

at org.apache.spark.sql.execution.python.PythonUDFRunner$$anon$2.read(PythonUDFRunner.scala:86)

at org.apache.spark.sql.execution.python.PythonUDFRunner$$anon$2.read(PythonUDFRunner.scala:68)

at org.apache.spark.api.python.BasePythonRunner$ReaderIterator.hasNext(PythonRunner.scala:508)

at org.apache.spark.InterruptibleIterator.hasNext(InterruptibleIterator.scala:37)

at scala.collection.Iterator$$anon$11.hasNext(Iterator.scala:491)

at scala.collection.Iterator$$anon$10.hasNext(Iterator.scala:460)

at scala.collection.Iterator$$anon$10.hasNext(Iterator.scala:460)

at org.apache.spark.sql.catalyst.expressions.GeneratedClass$GeneratedIteratorForCodegenStage1.processNext(Unknown Source)

at org.apache.spark.sql.execution.BufferedRowIterator.hasNext(BufferedRowIterator.java:43)

at org.apache.spark.sql.execution.WholeStageCodegenExec$$anon$1.hasNext(WholeStageCodegenExec.scala:759)

at org.apache.spark.sql.execution.columnar.DefaultCachedBatchSerializer$$anon$1.next(InMemoryRelation.scala:87)

at org.apache.spark.sql.execution.columnar.DefaultCachedBatchSerializer$$anon$1.next(InMemoryRelation.scala:79)

at scala.collection.Iterator$$anon$10.next(Iterator.scala:461)

at org.apache.spark.storage.memory.MemoryStore.putIterator(MemoryStore.scala:224)

at org.apache.spark.storage.memory.MemoryStore.putIteratorAsValues(MemoryStore.scala:302)

at org.apache.spark.storage.BlockManager.$anonfun$doPutIterator$1(BlockManager.scala:1481)

at org.apache.spark.storage.BlockManager.org$apache$spark$storage$BlockManager$$doPut(BlockManager.scala:1408)

at org.apache.spark.storage.BlockManager.doPutIterator(BlockManager.scala:1472)

at org.apache.spark.storage.BlockManager.getOrElseUpdate(BlockManager.scala:1295)

at org.apache.spark.rdd.RDD.getOrCompute(RDD.scala:384)

at org.apache.spark.rdd.RDD.iterator(RDD.scala:335)

at org.apache.spark.rdd.MapPartitionsRDD.compute(MapPartitionsRDD.scala:52)

at org.apache.spark.rdd.RDD.computeOrReadCheckpoint(RDD.scala:373)

at org.apache.spark.rdd.RDD.iterator(RDD.scala:337)

at org.apache.spark.rdd.MapPartitionsRDD.compute(MapPartitionsRDD.scala:52)

at org.apache.spark.rdd.RDD.computeOrReadCheckpoint(RDD.scala:373)

at org.apache.spark.rdd.RDD.iterator(RDD.scala:337)

at org.apache.spark.rdd.MapPartitionsRDD.compute(MapPartitionsRDD.scala:52)

at org.apache.spark.rdd.RDD.computeOrReadCheckpoint(RDD.scala:373)

at org.apache.spark.rdd.RDD.iterator(RDD.scala:337)

at org.apache.spark.rdd.MapPartitionsRDD.compute(MapPartitionsRDD.scala:52)

at org.apache.spark.rdd.RDD.computeOrReadCheckpoint(RDD.scala:373)

at org.apache.spark.rdd.RDD.iterator(RDD.scala:337)

at org.apache.spark.rdd.MapPartitionsRDD.compute(MapPartitionsRDD.scala:52)

at org.apache.spark.rdd.RDD.computeOrReadCheckpoint(RDD.scala:373)

at org.apache.spark.rdd.RDD.iterator(RDD.scala:337)

at org.apache.spark.scheduler.ResultTask.runTask(ResultTask.scala:90)

at org.apache.spark.scheduler.Task.run(Task.scala:131)

at org.apache.spark.executor.Executor$TaskRunner.$anonfun$run$3(Executor.scala:506)

at org.apache.spark.util.Utils$.tryWithSafeFinally(Utils.scala:1462)

at org.apache.spark.executor.Executor$TaskRunner.run(Executor.scala:509)

at java.util.concurrent.ThreadPoolExecutor.runWorker(ThreadPoolExecutor.java:1149)

at java.util.concurrent.ThreadPoolExecutor$Worker.run(ThreadPoolExecutor.java:624)

at java.lang.Thread.run(Thread.java:750)

24/07/22 09:32:00 ERROR scheduler.TaskSetManager: Task 0 in stage 106.0 failed 1 times; aborting job

Traceback (most recent call last):

File "<stdin>", line 1, in <module>

File "/usr/local/spark/python/pyspark/sql/dataframe.py", line 494, in show

print(self.\_jdf.showString(n, 20, vertical))

File "/usr/local/spark/python/lib/py4j-0.10.9.3-src.zip/py4j/java\_gateway.py", line 1322, in \_\_call\_\_

File "/usr/local/spark/python/pyspark/sql/utils.py", line 117, in deco

raise converted from None

pyspark.sql.utils.PythonException:

An exception was thrown from the Python worker. Please see the stack trace below.

Traceback (most recent call last):

File "/usr/local/spark/python/lib/pyspark.zip/pyspark/worker.py", line 619, in main

process()

File "/usr/local/spark/python/lib/pyspark.zip/pyspark/worker.py", line 611, in process

serializer.dump\_stream(out\_iter, outfile)

File "/usr/local/spark/python/lib/pyspark.zip/pyspark/serializers.py", line 211, in dump\_stream

self.serializer.dump\_stream(self.\_batched(iterator), stream)

File "/usr/local/spark/python/lib/pyspark.zip/pyspark/serializers.py", line 132, in dump\_stream

for obj in iterator:

File "/usr/local/spark/python/lib/pyspark.zip/pyspark/serializers.py", line 200, in \_batched

for item in iterator:

File "/usr/local/spark/python/lib/pyspark.zip/pyspark/worker.py", line 452, in mapper

result = tuple(f(\*[a[o] for o in arg\_offsets]) for (arg\_offsets, f) in udfs)

File "/usr/local/spark/python/lib/pyspark.zip/pyspark/worker.py", line 452, in <genexpr>

result = tuple(f(\*[a[o] for o in arg\_offsets]) for (arg\_offsets, f) in udfs)

File "/usr/local/spark/python/lib/pyspark.zip/pyspark/worker.py", line 87, in <lambda>

return lambda \*a: f(\*a)

File "/usr/local/spark/python/lib/pyspark.zip/pyspark/util.py", line 74, in wrapper

return f(\*args, \*\*kwargs)

File "<stdin>", line 3, in clean\_string

TypeError: expected string or bytes-like object

>>> df\_function=udf(clean\_string,StringType())

>>> sample\_op=merged\_data\_frame.withColumn("New NetworkName",df\_function(col("NetworkName")))

>>> sample\_op.show(3)

24/07/22 09:32:25 WARN storage.BlockManager: Putting block rdd\_531\_0 failed due to exception org.apache.spark.api.python.PythonException: Traceback (most recent call last):

File "/usr/local/spark/python/lib/pyspark.zip/pyspark/worker.py", line 619, in main

process()

File "/usr/local/spark/python/lib/pyspark.zip/pyspark/worker.py", line 611, in process

serializer.dump\_stream(out\_iter, outfile)

File "/usr/local/spark/python/lib/pyspark.zip/pyspark/serializers.py", line 211, in dump\_stream

self.serializer.dump\_stream(self.\_batched(iterator), stream)

File "/usr/local/spark/python/lib/pyspark.zip/pyspark/serializers.py", line 132, in dump\_stream

for obj in iterator:

File "/usr/local/spark/python/lib/pyspark.zip/pyspark/serializers.py", line 200, in \_batched

for item in iterator:

File "/usr/local/spark/python/lib/pyspark.zip/pyspark/worker.py", line 452, in mapper

result = tuple(f(\*[a[o] for o in arg\_offsets]) for (arg\_offsets, f) in udfs)

File "/usr/local/spark/python/lib/pyspark.zip/pyspark/worker.py", line 452, in <genexpr>

result = tuple(f(\*[a[o] for o in arg\_offsets]) for (arg\_offsets, f) in udfs)

File "/usr/local/spark/python/lib/pyspark.zip/pyspark/worker.py", line 87, in <lambda>

return lambda \*a: f(\*a)

File "/usr/local/spark/python/lib/pyspark.zip/pyspark/util.py", line 74, in wrapper

return f(\*args, \*\*kwargs)

File "<stdin>", line 3, in clean\_string

TypeError: expected string or bytes-like object

.

24/07/22 09:32:25 WARN storage.BlockManager: Block rdd\_531\_0 could not be removed as it was not found on disk or in memory

24/07/22 09:32:25 ERROR executor.Executor: Exception in task 0.0 in stage 107.0 (TID 118)

org.apache.spark.api.python.PythonException: Traceback (most recent call last):

File "/usr/local/spark/python/lib/pyspark.zip/pyspark/worker.py", line 619, in main

process()

File "/usr/local/spark/python/lib/pyspark.zip/pyspark/worker.py", line 611, in process

serializer.dump\_stream(out\_iter, outfile)

File "/usr/local/spark/python/lib/pyspark.zip/pyspark/serializers.py", line 211, in dump\_stream

self.serializer.dump\_stream(self.\_batched(iterator), stream)

File "/usr/local/spark/python/lib/pyspark.zip/pyspark/serializers.py", line 132, in dump\_stream

for obj in iterator:

File "/usr/local/spark/python/lib/pyspark.zip/pyspark/serializers.py", line 200, in \_batched

for item in iterator:

File "/usr/local/spark/python/lib/pyspark.zip/pyspark/worker.py", line 452, in mapper

result = tuple(f(\*[a[o] for o in arg\_offsets]) for (arg\_offsets, f) in udfs)

File "/usr/local/spark/python/lib/pyspark.zip/pyspark/worker.py", line 452, in <genexpr>

result = tuple(f(\*[a[o] for o in arg\_offsets]) for (arg\_offsets, f) in udfs)

File "/usr/local/spark/python/lib/pyspark.zip/pyspark/worker.py", line 87, in <lambda>

return lambda \*a: f(\*a)

File "/usr/local/spark/python/lib/pyspark.zip/pyspark/util.py", line 74, in wrapper

return f(\*args, \*\*kwargs)

File "<stdin>", line 3, in clean\_string

TypeError: expected string or bytes-like object

at org.apache.spark.api.python.BasePythonRunner$ReaderIterator.handlePythonException(PythonRunner.scala:555)

at org.apache.spark.sql.execution.python.PythonUDFRunner$$anon$2.read(PythonUDFRunner.scala:86)

at org.apache.spark.sql.execution.python.PythonUDFRunner$$anon$2.read(PythonUDFRunner.scala:68)

at org.apache.spark.api.python.BasePythonRunner$ReaderIterator.hasNext(PythonRunner.scala:508)

at org.apache.spark.InterruptibleIterator.hasNext(InterruptibleIterator.scala:37)

at scala.collection.Iterator$$anon$11.hasNext(Iterator.scala:491)

at scala.collection.Iterator$$anon$10.hasNext(Iterator.scala:460)

at scala.collection.Iterator$$anon$10.hasNext(Iterator.scala:460)

at org.apache.spark.sql.catalyst.expressions.GeneratedClass$GeneratedIteratorForCodegenStage1.processNext(Unknown Source)

at org.apache.spark.sql.execution.BufferedRowIterator.hasNext(BufferedRowIterator.java:43)

at org.apache.spark.sql.execution.WholeStageCodegenExec$$anon$1.hasNext(WholeStageCodegenExec.scala:759)

at org.apache.spark.sql.execution.columnar.DefaultCachedBatchSerializer$$anon$1.next(InMemoryRelation.scala:87)

at org.apache.spark.sql.execution.columnar.DefaultCachedBatchSerializer$$anon$1.next(InMemoryRelation.scala:79)

at scala.collection.Iterator$$anon$10.next(Iterator.scala:461)

at org.apache.spark.storage.memory.MemoryStore.putIterator(MemoryStore.scala:224)

at org.apache.spark.storage.memory.MemoryStore.putIteratorAsValues(MemoryStore.scala:302)

at org.apache.spark.storage.BlockManager.$anonfun$doPutIterator$1(BlockManager.scala:1481)

at org.apache.spark.storage.BlockManager.org$apache$spark$storage$BlockManager$$doPut(BlockManager.scala:1408)

at org.apache.spark.storage.BlockManager.doPutIterator(BlockManager.scala:1472)

at org.apache.spark.storage.BlockManager.getOrElseUpdate(BlockManager.scala:1295)

at org.apache.spark.rdd.RDD.getOrCompute(RDD.scala:384)

at org.apache.spark.rdd.RDD.iterator(RDD.scala:335)

at org.apache.spark.rdd.MapPartitionsRDD.compute(MapPartitionsRDD.scala:52)

at org.apache.spark.rdd.RDD.computeOrReadCheckpoint(RDD.scala:373)

at org.apache.spark.rdd.RDD.iterator(RDD.scala:337)

at org.apache.spark.rdd.MapPartitionsRDD.compute(MapPartitionsRDD.scala:52)

at org.apache.spark.rdd.RDD.computeOrReadCheckpoint(RDD.scala:373)

at org.apache.spark.rdd.RDD.iterator(RDD.scala:337)

at org.apache.spark.rdd.MapPartitionsRDD.compute(MapPartitionsRDD.scala:52)

at org.apache.spark.rdd.RDD.computeOrReadCheckpoint(RDD.scala:373)

at org.apache.spark.rdd.RDD.iterator(RDD.scala:337)

at org.apache.spark.rdd.MapPartitionsRDD.compute(MapPartitionsRDD.scala:52)

at org.apache.spark.rdd.RDD.computeOrReadCheckpoint(RDD.scala:373)

at org.apache.spark.rdd.RDD.iterator(RDD.scala:337)

at org.apache.spark.rdd.MapPartitionsRDD.compute(MapPartitionsRDD.scala:52)

at org.apache.spark.rdd.RDD.computeOrReadCheckpoint(RDD.scala:373)

at org.apache.spark.rdd.RDD.iterator(RDD.scala:337)

at org.apache.spark.scheduler.ResultTask.runTask(ResultTask.scala:90)

at org.apache.spark.scheduler.Task.run(Task.scala:131)

at org.apache.spark.executor.Executor$TaskRunner.$anonfun$run$3(Executor.scala:506)

at org.apache.spark.util.Utils$.tryWithSafeFinally(Utils.scala:1462)

at org.apache.spark.executor.Executor$TaskRunner.run(Executor.scala:509)

at java.util.concurrent.ThreadPoolExecutor.runWorker(ThreadPoolExecutor.java:1149)

at java.util.concurrent.ThreadPoolExecutor$Worker.run(ThreadPoolExecutor.java:624)

at java.lang.Thread.run(Thread.java:750)

24/07/22 09:32:25 WARN scheduler.TaskSetManager: Lost task 0.0 in stage 107.0 (TID 118) (192.168.112.129 executor driver): org.apache.spark.api.python.PythonException: Traceback (most recent call last):

File "/usr/local/spark/python/lib/pyspark.zip/pyspark/worker.py", line 619, in main

process()

File "/usr/local/spark/python/lib/pyspark.zip/pyspark/worker.py", line 611, in process

serializer.dump\_stream(out\_iter, outfile)

File "/usr/local/spark/python/lib/pyspark.zip/pyspark/serializers.py", line 211, in dump\_stream

self.serializer.dump\_stream(self.\_batched(iterator), stream)

File "/usr/local/spark/python/lib/pyspark.zip/pyspark/serializers.py", line 132, in dump\_stream

for obj in iterator:

File "/usr/local/spark/python/lib/pyspark.zip/pyspark/serializers.py", line 200, in \_batched

for item in iterator:

File "/usr/local/spark/python/lib/pyspark.zip/pyspark/worker.py", line 452, in mapper

result = tuple(f(\*[a[o] for o in arg\_offsets]) for (arg\_offsets, f) in udfs)

File "/usr/local/spark/python/lib/pyspark.zip/pyspark/worker.py", line 452, in <genexpr>

result = tuple(f(\*[a[o] for o in arg\_offsets]) for (arg\_offsets, f) in udfs)

File "/usr/local/spark/python/lib/pyspark.zip/pyspark/worker.py", line 87, in <lambda>

return lambda \*a: f(\*a)

File "/usr/local/spark/python/lib/pyspark.zip/pyspark/util.py", line 74, in wrapper

return f(\*args, \*\*kwargs)

File "<stdin>", line 3, in clean\_string

TypeError: expected string or bytes-like object

at org.apache.spark.api.python.BasePythonRunner$ReaderIterator.handlePythonException(PythonRunner.scala:555)

at org.apache.spark.sql.execution.python.PythonUDFRunner$$anon$2.read(PythonUDFRunner.scala:86)

at org.apache.spark.sql.execution.python.PythonUDFRunner$$anon$2.read(PythonUDFRunner.scala:68)

at org.apache.spark.api.python.BasePythonRunner$ReaderIterator.hasNext(PythonRunner.scala:508)

at org.apache.spark.InterruptibleIterator.hasNext(InterruptibleIterator.scala:37)

at scala.collection.Iterator$$anon$11.hasNext(Iterator.scala:491)

at scala.collection.Iterator$$anon$10.hasNext(Iterator.scala:460)

at scala.collection.Iterator$$anon$10.hasNext(Iterator.scala:460)

at org.apache.spark.sql.catalyst.expressions.GeneratedClass$GeneratedIteratorForCodegenStage1.processNext(Unknown Source)

at org.apache.spark.sql.execution.BufferedRowIterator.hasNext(BufferedRowIterator.java:43)

at org.apache.spark.sql.execution.WholeStageCodegenExec$$anon$1.hasNext(WholeStageCodegenExec.scala:759)

at org.apache.spark.sql.execution.columnar.DefaultCachedBatchSerializer$$anon$1.next(InMemoryRelation.scala:87)

at org.apache.spark.sql.execution.columnar.DefaultCachedBatchSerializer$$anon$1.next(InMemoryRelation.scala:79)

at scala.collection.Iterator$$anon$10.next(Iterator.scala:461)

at org.apache.spark.storage.memory.MemoryStore.putIterator(MemoryStore.scala:224)

at org.apache.spark.storage.memory.MemoryStore.putIteratorAsValues(MemoryStore.scala:302)

at org.apache.spark.storage.BlockManager.$anonfun$doPutIterator$1(BlockManager.scala:1481)

at org.apache.spark.storage.BlockManager.org$apache$spark$storage$BlockManager$$doPut(BlockManager.scala:1408)

at org.apache.spark.storage.BlockManager.doPutIterator(BlockManager.scala:1472)

at org.apache.spark.storage.BlockManager.getOrElseUpdate(BlockManager.scala:1295)

at org.apache.spark.rdd.RDD.getOrCompute(RDD.scala:384)

at org.apache.spark.rdd.RDD.iterator(RDD.scala:335)

at org.apache.spark.rdd.MapPartitionsRDD.compute(MapPartitionsRDD.scala:52)

at org.apache.spark.rdd.RDD.computeOrReadCheckpoint(RDD.scala:373)

at org.apache.spark.rdd.RDD.iterator(RDD.scala:337)

at org.apache.spark.rdd.MapPartitionsRDD.compute(MapPartitionsRDD.scala:52)

at org.apache.spark.rdd.RDD.computeOrReadCheckpoint(RDD.scala:373)

at org.apache.spark.rdd.RDD.iterator(RDD.scala:337)

at org.apache.spark.rdd.MapPartitionsRDD.compute(MapPartitionsRDD.scala:52)

at org.apache.spark.rdd.RDD.computeOrReadCheckpoint(RDD.scala:373)

at org.apache.spark.rdd.RDD.iterator(RDD.scala:337)

at org.apache.spark.rdd.MapPartitionsRDD.compute(MapPartitionsRDD.scala:52)

at org.apache.spark.rdd.RDD.computeOrReadCheckpoint(RDD.scala:373)

at org.apache.spark.rdd.RDD.iterator(RDD.scala:337)

at org.apache.spark.rdd.MapPartitionsRDD.compute(MapPartitionsRDD.scala:52)

at org.apache.spark.rdd.RDD.computeOrReadCheckpoint(RDD.scala:373)

at org.apache.spark.rdd.RDD.iterator(RDD.scala:337)

at org.apache.spark.scheduler.ResultTask.runTask(ResultTask.scala:90)

at org.apache.spark.scheduler.Task.run(Task.scala:131)

at org.apache.spark.executor.Executor$TaskRunner.$anonfun$run$3(Executor.scala:506)

at org.apache.spark.util.Utils$.tryWithSafeFinally(Utils.scala:1462)

at org.apache.spark.executor.Executor$TaskRunner.run(Executor.scala:509)

at java.util.concurrent.ThreadPoolExecutor.runWorker(ThreadPoolExecutor.java:1149)

at java.util.concurrent.ThreadPoolExecutor$Worker.run(ThreadPoolExecutor.java:624)

at java.lang.Thread.run(Thread.java:750)

24/07/22 09:32:25 ERROR scheduler.TaskSetManager: Task 0 in stage 107.0 failed 1 times; aborting job

Traceback (most recent call last):

File "<stdin>", line 1, in <module>

File "/usr/local/spark/python/pyspark/sql/dataframe.py", line 494, in show

print(self.\_jdf.showString(n, 20, vertical))

File "/usr/local/spark/python/lib/py4j-0.10.9.3-src.zip/py4j/java\_gateway.py", line 1322, in \_\_call\_\_

File "/usr/local/spark/python/pyspark/sql/utils.py", line 117, in deco

raise converted from None

pyspark.sql.utils.PythonException:

An exception was thrown from the Python worker. Please see the stack trace below.

Traceback (most recent call last):

File "/usr/local/spark/python/lib/pyspark.zip/pyspark/worker.py", line 619, in main

process()

File "/usr/local/spark/python/lib/pyspark.zip/pyspark/worker.py", line 611, in process

serializer.dump\_stream(out\_iter, outfile)

File "/usr/local/spark/python/lib/pyspark.zip/pyspark/serializers.py", line 211, in dump\_stream

self.serializer.dump\_stream(self.\_batched(iterator), stream)

File "/usr/local/spark/python/lib/pyspark.zip/pyspark/serializers.py", line 132, in dump\_stream

for obj in iterator:

File "/usr/local/spark/python/lib/pyspark.zip/pyspark/serializers.py", line 200, in \_batched

for item in iterator:

File "/usr/local/spark/python/lib/pyspark.zip/pyspark/worker.py", line 452, in mapper

result = tuple(f(\*[a[o] for o in arg\_offsets]) for (arg\_offsets, f) in udfs)

File "/usr/local/spark/python/lib/pyspark.zip/pyspark/worker.py", line 452, in <genexpr>

result = tuple(f(\*[a[o] for o in arg\_offsets]) for (arg\_offsets, f) in udfs)

File "/usr/local/spark/python/lib/pyspark.zip/pyspark/worker.py", line 87, in <lambda>

return lambda \*a: f(\*a)

File "/usr/local/spark/python/lib/pyspark.zip/pyspark/util.py", line 74, in wrapper

return f(\*args, \*\*kwargs)

File "<stdin>", line 3, in clean\_string

TypeError: expected string or bytes-like object

>>> import re

>>>

>>> def clean\_string(string\_arg):

... if isinstance(string\_arg, str):

... # Define the regex pattern to match non-alphabetic characters and replace them with space

... pattern = re.compile(r'[^A-Za-z\s]')

... final\_string = pattern.sub(' ', string\_arg)

... return final\_string

... else:

... return None

...

>>> df\_function=udf(clean\_string,StringType())

>>> sample\_op=merged\_data\_frame.withColumn("New NetworkName",df\_function(col("NetworkName")))

>>> sample\_op.show(3)

+------------+----+-----+------------+--------------------+-------+--------------+--------------------+--------+---------+-----------------+----------+--------------------+---------------+

|BusinessDate|stcd|srcnm| NetworkName| NetworkURL|custnum|MarketCoverage| RejectRows|IssuerId|IssuerId2|issueridcomposite| sysdt| systs|New NetworkName|

+------------+----+-----+------------+--------------------+-------+--------------+--------------------+--------+---------+-----------------+----------+--------------------+---------------+

| 2019-10-01| AK| HIOS| ODS Premier|https://www.modah...| 13| null| null| 21989| 21989| 2198921989|2024-07-22|2024-07-22 09:35:...| ODS Premier|

| 2019-10-01| AK| HIOS| ODS Premier| null| null| null|219,219,2019-10-0...| 219| 219| 219219|2024-07-22|2024-07-22 09:35:...| ODS Premier|

| 2019-10-01| AK| HIOS|HeritagePlus|https://www.preme...| 13| null| null| 38344| 38344| 3834438344|2024-07-22|2024-07-22 09:35:...| HeritagePlus|

+------------+----+-----+------------+--------------------+-------+--------------+--------------------+--------+---------+-----------------+----------+--------------------+---------------+

only showing top 3 rows

>>> sample\_op.write.json("/user/hduser/Hackathon/JSON\_DATA" , mode="overwrite")

>>> sample\_op.write.csv("/user/hduser/Hackathon/CSV\_DATA" , header=True , sep="~")

>>>

Traceback (most recent call last):

File "/usr/local/spark/python/pyspark/context.py", line 293, in signal\_handler

raise KeyboardInterrupt()

KeyboardInterrupt

>>> database\_name = "Hackathon"

>>> table\_name = "Hive\_table"

>>> table\_location = "/user/hduser/Hackathon/Hive\_table"

>>>

>>> # Create Hive database if it doesn't exist

... spark.sql(f"CREATE DATABASE IF NOT EXISTS {database\_name}")

24/07/22 09:41:26 WARN metastore.ObjectStore: Failed to get database global\_temp, returning NoSuchObjectException

24/07/22 09:41:27 WARN metastore.ObjectStore: Failed to get database hackathon, returning NoSuchObjectException

DataFrame[]

>>>

>>> # Define the schema for the table

... schema = """

... BusinessDate DATE,

... stcd STRING,

... srcnm STRING,

... NetworkName STRING,

... NetworkURL STRING,

... custnum STRING,

... MarketCoverage STRING,

... RejectRows STRING,

... IssuerId INT,

... IssuerId2 INT,

... issueridcomposite STRING,

... sysdt TIMESTAMP,

... systs STRING,

... `New NetworkName` STRING

... """

>>>

>>> # Create the Hive table with the specified schema

... spark.sql(f"""

... CREATE EXTERNAL TABLE IF NOT EXISTS {database\_name}.{table\_name} (

... {schema}

... )

... LOCATION '{table\_location}'

... """)

24/07/22 09:41:29 WARN analysis.ResolveSessionCatalog: A Hive serde table will be created as there is no table provider specified. You can set spark.sql.legacy.createHiveTableByDefault to false so that native data source table will be created instead.

24/07/22 09:41:30 WARN session.SessionState: METASTORE\_FILTER\_HOOK will be ignored, since hive.security.authorization.manager is set to instance of HiveAuthorizerFactory.

24/07/22 09:41:33 WARN conf.HiveConf: HiveConf of name hive.internal.ss.authz.settings.applied.marker does not exist

24/07/22 09:41:33 WARN conf.HiveConf: HiveConf of name hive.stats.jdbc.timeout does not exist

24/07/22 09:41:33 WARN conf.HiveConf: HiveConf of name hive.stats.retries.wait does not exist

DataFrame[]

>>>

>>> print(f"Table {table\_name} created in database {database\_name} at location {table\_location}")

Table Hive\_table created in database Hackathon at location /user/hduser/Hackathon/Hive\_table

>>>

>>> rdd=read.textFile("/home/hduser/Desktop/Hackathon")

Traceback (most recent call last):

File "<stdin>", line 1, in <module>

NameError: name 'read' is not defined

>>> rdd=sc.read.textFile("/home/hduser/Desktop/Hackathon")

Traceback (most recent call last):

File "<stdin>", line 1, in <module>

AttributeError: 'SparkContext' object has no attribute 'read'

>>> rdd=sc.textFile("/home/hduser/Desktop/Hackathon")

>>> rdd.take(3)

Traceback (most recent call last):

File "<stdin>", line 1, in <module>

File "/usr/local/spark/python/pyspark/rdd.py", line 1535, in take

totalParts = self.getNumPartitions()

File "/usr/local/spark/python/pyspark/rdd.py", line 466, in getNumPartitions

return self.\_jrdd.partitions().size()

File "/usr/local/spark/python/lib/py4j-0.10.9.3-src.zip/py4j/java\_gateway.py", line 1322, in \_\_call\_\_

File "/usr/local/spark/python/pyspark/sql/utils.py", line 111, in deco

return f(\*a, \*\*kw)

File "/usr/local/spark/python/lib/py4j-0.10.9.3-src.zip/py4j/protocol.py", line 328, in get\_return\_value

py4j.protocol.Py4JJavaError: An error occurred while calling o751.partitions.

: org.apache.hadoop.mapred.InvalidInputException: Input path does not exist: hdfs://localhost:54310/home/hduser/Desktop/Hackathon

at org.apache.hadoop.mapred.FileInputFormat.singleThreadedListStatus(FileInputFormat.java:287)

at org.apache.hadoop.mapred.FileInputFormat.listStatus(FileInputFormat.java:229)

at org.apache.hadoop.mapred.FileInputFormat.getSplits(FileInputFormat.java:315)

at org.apache.spark.rdd.HadoopRDD.getPartitions(HadoopRDD.scala:205)

at org.apache.spark.rdd.RDD.$anonfun$partitions$2(RDD.scala:300)

at scala.Option.getOrElse(Option.scala:189)

at org.apache.spark.rdd.RDD.partitions(RDD.scala:296)

at org.apache.spark.rdd.MapPartitionsRDD.getPartitions(MapPartitionsRDD.scala:49)

at org.apache.spark.rdd.RDD.$anonfun$partitions$2(RDD.scala:300)

at scala.Option.getOrElse(Option.scala:189)

at org.apache.spark.rdd.RDD.partitions(RDD.scala:296)

at org.apache.spark.api.java.JavaRDDLike.partitions(JavaRDDLike.scala:61)

at org.apache.spark.api.java.JavaRDDLike.partitions$(JavaRDDLike.scala:61)

at org.apache.spark.api.java.AbstractJavaRDDLike.partitions(JavaRDDLike.scala:45)

at sun.reflect.NativeMethodAccessorImpl.invoke0(Native Method)

at sun.reflect.NativeMethodAccessorImpl.invoke(NativeMethodAccessorImpl.java:62)

at sun.reflect.DelegatingMethodAccessorImpl.invoke(DelegatingMethodAccessorImpl.java:43)

at java.lang.reflect.Method.invoke(Method.java:498)

at py4j.reflection.MethodInvoker.invoke(MethodInvoker.java:244)

at py4j.reflection.ReflectionEngine.invoke(ReflectionEngine.java:357)

at py4j.Gateway.invoke(Gateway.java:282)

at py4j.commands.AbstractCommand.invokeMethod(AbstractCommand.java:132)

at py4j.commands.CallCommand.execute(CallCommand.java:79)

at py4j.ClientServerConnection.waitForCommands(ClientServerConnection.java:182)

at py4j.ClientServerConnection.run(ClientServerConnection.java:106)

at java.lang.Thread.run(Thread.java:750)

>>> rdd=sc.textFile("file:///home/hduser/Desktop/Hackathon")

>>> rdd.take(3)

['1,Kristina,Chung,55,Pilot', '2,Paige,Chen,74,Teacher', 'NM,New Mexico']

>>> def function (rdd):

... for i in rdd :

... if len(i) ==5 :

... custfilter.append(i)

... else :

... statefilter.append(i)

...

>>> statefilter

Traceback (most recent call last):

File "<stdin>", line 1, in <module>

NameError: name 'statefilter' is not defined

>>> statefilter =[]

>>> custfilter =[]

>>> function(rdd)

Traceback (most recent call last):

File "<stdin>", line 1, in <module>

File "<stdin>", line 2, in function

TypeError: 'RDD' object is not iterable

>>> custfilter =rdd.filter(lambda x: if len(x)>5 return x )

File "<stdin>", line 1

custfilter =rdd.filter(lambda x: if len(x)>5 return x )

^

SyntaxError: invalid syntax

>>> custfilter =rdd.filter(lambda x: if len(x)>5 )

File "<stdin>", line 1

custfilter =rdd.filter(lambda x: if len(x)>5 )

^

SyntaxError: invalid syntax

>>> custfilter =rdd.filter(lambda x: len(x)>5 )

>>> print(custfilter)

PythonRDD[629] at RDD at PythonRDD.scala:53

>>> print(custfilter.take(5))

['1,Kristina,Chung,55,Pilot', '2,Paige,Chen,74,Teacher', 'NM,New Mexico', 'NY,New York', 'NC,North Carolina']

>>> custfilter =rdd.filter(lambda x: len(x)>2 )

>>> print(custfilter.take(5))

['1,Kristina,Chung,55,Pilot', '2,Paige,Chen,74,Teacher', 'NM,New Mexico', 'NY,New York', 'NC,North Carolina']

>>> custfilter =rdd.map(lambda x: len(x)>2 ).collect()

>>> print(custfilter.take(5))

Traceback (most recent call last):

File "<stdin>", line 1, in <module>

AttributeError: 'list' object has no attribute 'take'

>>> print(custfilter)

[True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True]

>>> custfilter =rdd.map(lambda x: len(x)>2 ).collect()

>>> custfilter =rdd.map(lambda x: len(x)>2 )

>>> print(custfilter)

PythonRDD[634] at RDD at PythonRDD.scala:53

>>> print(custfilter.take()3)

File "<stdin>", line 1

print(custfilter.take()3)

^

SyntaxError: invalid syntax

>>> print(custfilter.take(4))

[True, True, True, True]

>>> print(custfilter.take(14))

[True, True, True, True, True, True, True, True, True, True, True, True, True, True]

>>> custfilter =rdd.map(lambda x: (len(x)>2).collect )

>>> print(custfilter.take(14))

24/07/22 11:12:52 ERROR executor.Executor: Exception in task 0.0 in stage 118.0 (TID 137)

org.apache.spark.api.python.PythonException: Traceback (most recent call last):

File "/usr/local/spark/python/lib/pyspark.zip/pyspark/worker.py", line 619, in main

process()

File "/usr/local/spark/python/lib/pyspark.zip/pyspark/worker.py", line 611, in process

serializer.dump\_stream(out\_iter, outfile)

File "/usr/local/spark/python/lib/pyspark.zip/pyspark/serializers.py", line 259, in dump\_stream

vs = list(itertools.islice(iterator, batch))

File "/usr/local/spark/python/pyspark/rdd.py", line 1562, in takeUpToNumLeft

yield next(iterator)

File "/usr/local/spark/python/lib/pyspark.zip/pyspark/util.py", line 74, in wrapper

return f(\*args, \*\*kwargs)

File "<stdin>", line 1, in <lambda>

AttributeError: 'bool' object has no attribute 'collect'

at org.apache.spark.api.python.BasePythonRunner$ReaderIterator.handlePythonException(PythonRunner.scala:555)

at org.apache.spark.api.python.PythonRunner$$anon$3.read(PythonRunner.scala:713)

at org.apache.spark.api.python.PythonRunner$$anon$3.read(PythonRunner.scala:695)

at org.apache.spark.api.python.BasePythonRunner$ReaderIterator.hasNext(PythonRunner.scala:508)

at org.apache.spark.InterruptibleIterator.hasNext(InterruptibleIterator.scala:37)

at scala.collection.Iterator.foreach(Iterator.scala:943)

at scala.collection.Iterator.foreach$(Iterator.scala:943)

at org.apache.spark.InterruptibleIterator.foreach(InterruptibleIterator.scala:28)

at scala.collection.generic.Growable.$plus$plus$eq(Growable.scala:62)

at scala.collection.generic.Growable.$plus$plus$eq$(Growable.scala:53)

at scala.collection.mutable.ArrayBuffer.$plus$plus$eq(ArrayBuffer.scala:105)

at scala.collection.mutable.ArrayBuffer.$plus$plus$eq(ArrayBuffer.scala:49)

at scala.collection.TraversableOnce.to(TraversableOnce.scala:366)

at scala.collection.TraversableOnce.to$(TraversableOnce.scala:364)

at org.apache.spark.InterruptibleIterator.to(InterruptibleIterator.scala:28)

at scala.collection.TraversableOnce.toBuffer(TraversableOnce.scala:358)

at scala.collection.TraversableOnce.toBuffer$(TraversableOnce.scala:358)

at org.apache.spark.InterruptibleIterator.toBuffer(InterruptibleIterator.scala:28)

at scala.collection.TraversableOnce.toArray(TraversableOnce.scala:345)

at scala.collection.TraversableOnce.toArray$(TraversableOnce.scala:339)

at org.apache.spark.InterruptibleIterator.toArray(InterruptibleIterator.scala:28)

at org.apache.spark.api.python.PythonRDD$.$anonfun$runJob$1(PythonRDD.scala:166)

at org.apache.spark.SparkContext.$anonfun$runJob$5(SparkContext.scala:2254)

at org.apache.spark.scheduler.ResultTask.runTask(ResultTask.scala:90)

at org.apache.spark.scheduler.Task.run(Task.scala:131)

at org.apache.spark.executor.Executor$TaskRunner.$anonfun$run$3(Executor.scala:506)

at org.apache.spark.util.Utils$.tryWithSafeFinally(Utils.scala:1462)

at org.apache.spark.executor.Executor$TaskRunner.run(Executor.scala:509)

at java.util.concurrent.ThreadPoolExecutor.runWorker(ThreadPoolExecutor.java:1149)

at java.util.concurrent.ThreadPoolExecutor$Worker.run(ThreadPoolExecutor.java:624)

at java.lang.Thread.run(Thread.java:750)

24/07/22 11:12:52 WARN scheduler.TaskSetManager: Lost task 0.0 in stage 118.0 (TID 137) (192.168.112.129 executor driver): org.apache.spark.api.python.PythonException: Traceback (most recent call last):

File "/usr/local/spark/python/lib/pyspark.zip/pyspark/worker.py", line 619, in main

process()

File "/usr/local/spark/python/lib/pyspark.zip/pyspark/worker.py", line 611, in process

serializer.dump\_stream(out\_iter, outfile)

File "/usr/local/spark/python/lib/pyspark.zip/pyspark/serializers.py", line 259, in dump\_stream

vs = list(itertools.islice(iterator, batch))

File "/usr/local/spark/python/pyspark/rdd.py", line 1562, in takeUpToNumLeft

yield next(iterator)

File "/usr/local/spark/python/lib/pyspark.zip/pyspark/util.py", line 74, in wrapper

return f(\*args, \*\*kwargs)

File "<stdin>", line 1, in <lambda>

AttributeError: 'bool' object has no attribute 'collect'

at org.apache.spark.api.python.BasePythonRunner$ReaderIterator.handlePythonException(PythonRunner.scala:555)

at org.apache.spark.api.python.PythonRunner$$anon$3.read(PythonRunner.scala:713)

at org.apache.spark.api.python.PythonRunner$$anon$3.read(PythonRunner.scala:695)

at org.apache.spark.api.python.BasePythonRunner$ReaderIterator.hasNext(PythonRunner.scala:508)

at org.apache.spark.InterruptibleIterator.hasNext(InterruptibleIterator.scala:37)

at scala.collection.Iterator.foreach(Iterator.scala:943)

at scala.collection.Iterator.foreach$(Iterator.scala:943)

at org.apache.spark.InterruptibleIterator.foreach(InterruptibleIterator.scala:28)

at scala.collection.generic.Growable.$plus$plus$eq(Growable.scala:62)

at scala.collection.generic.Growable.$plus$plus$eq$(Growable.scala:53)

at scala.collection.mutable.ArrayBuffer.$plus$plus$eq(ArrayBuffer.scala:105)

at scala.collection.mutable.ArrayBuffer.$plus$plus$eq(ArrayBuffer.scala:49)

at scala.collection.TraversableOnce.to(TraversableOnce.scala:366)

at scala.collection.TraversableOnce.to$(TraversableOnce.scala:364)

at org.apache.spark.InterruptibleIterator.to(InterruptibleIterator.scala:28)

at scala.collection.TraversableOnce.toBuffer(TraversableOnce.scala:358)

at scala.collection.TraversableOnce.toBuffer$(TraversableOnce.scala:358)

at org.apache.spark.InterruptibleIterator.toBuffer(InterruptibleIterator.scala:28)

at scala.collection.TraversableOnce.toArray(TraversableOnce.scala:345)

at scala.collection.TraversableOnce.toArray$(TraversableOnce.scala:339)

at org.apache.spark.InterruptibleIterator.toArray(InterruptibleIterator.scala:28)

at org.apache.spark.api.python.PythonRDD$.$anonfun$runJob$1(PythonRDD.scala:166)

at org.apache.spark.SparkContext.$anonfun$runJob$5(SparkContext.scala:2254)

at org.apache.spark.scheduler.ResultTask.runTask(ResultTask.scala:90)

at org.apache.spark.scheduler.Task.run(Task.scala:131)

at org.apache.spark.executor.Executor$TaskRunner.$anonfun$run$3(Executor.scala:506)

at org.apache.spark.util.Utils$.tryWithSafeFinally(Utils.scala:1462)

at org.apache.spark.executor.Executor$TaskRunner.run(Executor.scala:509)

at java.util.concurrent.ThreadPoolExecutor.runWorker(ThreadPoolExecutor.java:1149)

at java.util.concurrent.ThreadPoolExecutor$Worker.run(ThreadPoolExecutor.java:624)

at java.lang.Thread.run(Thread.java:750)

24/07/22 11:12:52 ERROR scheduler.TaskSetManager: Task 0 in stage 118.0 failed 1 times; aborting job

Traceback (most recent call last):

File "<stdin>", line 1, in <module>

File "/usr/local/spark/python/pyspark/rdd.py", line 1568, in take

res = self.context.runJob(self, takeUpToNumLeft, p)

File "/usr/local/spark/python/pyspark/context.py", line 1227, in runJob

sock\_info = self.\_jvm.PythonRDD.runJob(self.\_jsc.sc(), mappedRDD.\_jrdd, partitions)

File "/usr/local/spark/python/lib/py4j-0.10.9.3-src.zip/py4j/java\_gateway.py", line 1322, in \_\_call\_\_

File "/usr/local/spark/python/pyspark/sql/utils.py", line 111, in deco

return f(\*a, \*\*kw)

File "/usr/local/spark/python/lib/py4j-0.10.9.3-src.zip/py4j/protocol.py", line 328, in get\_return\_value

py4j.protocol.Py4JJavaError: An error occurred while calling z:org.apache.spark.api.python.PythonRDD.runJob.

: org.apache.spark.SparkException: Job aborted due to stage failure: Task 0 in stage 118.0 failed 1 times, most recent failure: Lost task 0.0 in stage 118.0 (TID 137) (192.168.112.129 executor driver): org.apache.spark.api.python.PythonException: Traceback (most recent call last):

File "/usr/local/spark/python/lib/pyspark.zip/pyspark/worker.py", line 619, in main

process()

File "/usr/local/spark/python/lib/pyspark.zip/pyspark/worker.py", line 611, in process

serializer.dump\_stream(out\_iter, outfile)

File "/usr/local/spark/python/lib/pyspark.zip/pyspark/serializers.py", line 259, in dump\_stream

vs = list(itertools.islice(iterator, batch))

File "/usr/local/spark/python/pyspark/rdd.py", line 1562, in takeUpToNumLeft

yield next(iterator)

File "/usr/local/spark/python/lib/pyspark.zip/pyspark/util.py", line 74, in wrapper

return f(\*args, \*\*kwargs)

File "<stdin>", line 1, in <lambda>

AttributeError: 'bool' object has no attribute 'collect'

at org.apache.spark.api.python.BasePythonRunner$ReaderIterator.handlePythonException(PythonRunner.scala:555)

at org.apache.spark.api.python.PythonRunner$$anon$3.read(PythonRunner.scala:713)

at org.apache.spark.api.python.PythonRunner$$anon$3.read(PythonRunner.scala:695)

at org.apache.spark.api.python.BasePythonRunner$ReaderIterator.hasNext(PythonRunner.scala:508)

at org.apache.spark.InterruptibleIterator.hasNext(InterruptibleIterator.scala:37)

at scala.collection.Iterator.foreach(Iterator.scala:943)

at scala.collection.Iterator.foreach$(Iterator.scala:943)

at org.apache.spark.InterruptibleIterator.foreach(InterruptibleIterator.scala:28)

at scala.collection.generic.Growable.$plus$plus$eq(Growable.scala:62)

at scala.collection.generic.Growable.$plus$plus$eq$(Growable.scala:53)

at scala.collection.mutable.ArrayBuffer.$plus$plus$eq(ArrayBuffer.scala:105)

at scala.collection.mutable.ArrayBuffer.$plus$plus$eq(ArrayBuffer.scala:49)

at scala.collection.TraversableOnce.to(TraversableOnce.scala:366)

at scala.collection.TraversableOnce.to$(TraversableOnce.scala:364)

at org.apache.spark.InterruptibleIterator.to(InterruptibleIterator.scala:28)

at scala.collection.TraversableOnce.toBuffer(TraversableOnce.scala:358)

at scala.collection.TraversableOnce.toBuffer$(TraversableOnce.scala:358)

at org.apache.spark.InterruptibleIterator.toBuffer(InterruptibleIterator.scala:28)

at scala.collection.TraversableOnce.toArray(TraversableOnce.scala:345)

at scala.collection.TraversableOnce.toArray$(TraversableOnce.scala:339)

at org.apache.spark.InterruptibleIterator.toArray(InterruptibleIterator.scala:28)

at org.apache.spark.api.python.PythonRDD$.$anonfun$runJob$1(PythonRDD.scala:166)

at org.apache.spark.SparkContext.$anonfun$runJob$5(SparkContext.scala:2254)

at org.apache.spark.scheduler.ResultTask.runTask(ResultTask.scala:90)

at org.apache.spark.scheduler.Task.run(Task.scala:131)

at org.apache.spark.executor.Executor$TaskRunner.$anonfun$run$3(Executor.scala:506)

at org.apache.spark.util.Utils$.tryWithSafeFinally(Utils.scala:1462)

at org.apache.spark.executor.Executor$TaskRunner.run(Executor.scala:509)

at java.util.concurrent.ThreadPoolExecutor.runWorker(ThreadPoolExecutor.java:1149)

at java.util.concurrent.ThreadPoolExecutor$Worker.run(ThreadPoolExecutor.java:624)

at java.lang.Thread.run(Thread.java:750)

Driver stacktrace:

at org.apache.spark.scheduler.DAGScheduler.failJobAndIndependentStages(DAGScheduler.scala:2454)

at org.apache.spark.scheduler.DAGScheduler.$anonfun$abortStage$2(DAGScheduler.scala:2403)

at org.apache.spark.scheduler.DAGScheduler.$anonfun$abortStage$2$adapted(DAGScheduler.scala:2402)

at scala.collection.mutable.ResizableArray.foreach(ResizableArray.scala:62)

at scala.collection.mutable.ResizableArray.foreach$(ResizableArray.scala:55)

at scala.collection.mutable.ArrayBuffer.foreach(ArrayBuffer.scala:49)

at org.apache.spark.scheduler.DAGScheduler.abortStage(DAGScheduler.scala:2402)

at org.apache.spark.scheduler.DAGScheduler.$anonfun$handleTaskSetFailed$1(DAGScheduler.scala:1160)

at org.apache.spark.scheduler.DAGScheduler.$anonfun$handleTaskSetFailed$1$adapted(DAGScheduler.scala:1160)

at scala.Option.foreach(Option.scala:407)

at org.apache.spark.scheduler.DAGScheduler.handleTaskSetFailed(DAGScheduler.scala:1160)

at org.apache.spark.scheduler.DAGSchedulerEventProcessLoop.doOnReceive(DAGScheduler.scala:2642)

at org.apache.spark.scheduler.DAGSchedulerEventProcessLoop.onReceive(DAGScheduler.scala:2584)

at org.apache.spark.scheduler.DAGSchedulerEventProcessLoop.onReceive(DAGScheduler.scala:2573)

at org.apache.spark.util.EventLoop$$anon$1.run(EventLoop.scala:49)

at org.apache.spark.scheduler.DAGScheduler.runJob(DAGScheduler.scala:938)

at org.apache.spark.SparkContext.runJob(SparkContext.scala:2214)

at org.apache.spark.SparkContext.runJob(SparkContext.scala:2235)

at org.apache.spark.SparkContext.runJob(SparkContext.scala:2254)

at org.apache.spark.api.python.PythonRDD$.runJob(PythonRDD.scala:166)

at org.apache.spark.api.python.PythonRDD.runJob(PythonRDD.scala)

at sun.reflect.NativeMethodAccessorImpl.invoke0(Native Method)

at sun.reflect.NativeMethodAccessorImpl.invoke(NativeMethodAccessorImpl.java:62)

at sun.reflect.DelegatingMethodAccessorImpl.invoke(DelegatingMethodAccessorImpl.java:43)

at java.lang.reflect.Method.invoke(Method.java:498)

at py4j.reflection.MethodInvoker.invoke(MethodInvoker.java:244)

at py4j.reflection.ReflectionEngine.invoke(ReflectionEngine.java:357)

at py4j.Gateway.invoke(Gateway.java:282)

at py4j.commands.AbstractCommand.invokeMethod(AbstractCommand.java:132)

at py4j.commands.CallCommand.execute(CallCommand.java:79)

at py4j.ClientServerConnection.waitForCommands(ClientServerConnection.java:182)

at py4j.ClientServerConnection.run(ClientServerConnection.java:106)

at java.lang.Thread.run(Thread.java:750)

Caused by: org.apache.spark.api.python.PythonException: Traceback (most recent call last):

File "/usr/local/spark/python/lib/pyspark.zip/pyspark/worker.py", line 619, in main

process()

File "/usr/local/spark/python/lib/pyspark.zip/pyspark/worker.py", line 611, in process

serializer.dump\_stream(out\_iter, outfile)

File "/usr/local/spark/python/lib/pyspark.zip/pyspark/serializers.py", line 259, in dump\_stream

vs = list(itertools.islice(iterator, batch))

File "/usr/local/spark/python/pyspark/rdd.py", line 1562, in takeUpToNumLeft

yield next(iterator)

File "/usr/local/spark/python/lib/pyspark.zip/pyspark/util.py", line 74, in wrapper

return f(\*args, \*\*kwargs)

File "<stdin>", line 1, in <lambda>

AttributeError: 'bool' object has no attribute 'collect'

at org.apache.spark.api.python.BasePythonRunner$ReaderIterator.handlePythonException(PythonRunner.scala:555)

at org.apache.spark.api.python.PythonRunner$$anon$3.read(PythonRunner.scala:713)

at org.apache.spark.api.python.PythonRunner$$anon$3.read(PythonRunner.scala:695)

at org.apache.spark.api.python.BasePythonRunner$ReaderIterator.hasNext(PythonRunner.scala:508)

at org.apache.spark.InterruptibleIterator.hasNext(InterruptibleIterator.scala:37)

at scala.collection.Iterator.foreach(Iterator.scala:943)

at scala.collection.Iterator.foreach$(Iterator.scala:943)

at org.apache.spark.InterruptibleIterator.foreach(InterruptibleIterator.scala:28)

at scala.collection.generic.Growable.$plus$plus$eq(Growable.scala:62)

at scala.collection.generic.Growable.$plus$plus$eq$(Growable.scala:53)

at scala.collection.mutable.ArrayBuffer.$plus$plus$eq(ArrayBuffer.scala:105)

at scala.collection.mutable.ArrayBuffer.$plus$plus$eq(ArrayBuffer.scala:49)

at scala.collection.TraversableOnce.to(TraversableOnce.scala:366)

at scala.collection.TraversableOnce.to$(TraversableOnce.scala:364)

at org.apache.spark.InterruptibleIterator.to(InterruptibleIterator.scala:28)

at scala.collection.TraversableOnce.toBuffer(TraversableOnce.scala:358)

at scala.collection.TraversableOnce.toBuffer$(TraversableOnce.scala:358)

at org.apache.spark.InterruptibleIterator.toBuffer(InterruptibleIterator.scala:28)

at scala.collection.TraversableOnce.toArray(TraversableOnce.scala:345)

at scala.collection.TraversableOnce.toArray$(TraversableOnce.scala:339)

at org.apache.spark.InterruptibleIterator.toArray(InterruptibleIterator.scala:28)

at org.apache.spark.api.python.PythonRDD$.$anonfun$runJob$1(PythonRDD.scala:166)

at org.apache.spark.SparkContext.$anonfun$runJob$5(SparkContext.scala:2254)

at org.apache.spark.scheduler.ResultTask.runTask(ResultTask.scala:90)

at org.apache.spark.scheduler.Task.run(Task.scala:131)

at org.apache.spark.executor.Executor$TaskRunner.$anonfun$run$3(Executor.scala:506)

at org.apache.spark.util.Utils$.tryWithSafeFinally(Utils.scala:1462)

at org.apache.spark.executor.Executor$TaskRunner.run(Executor.scala:509)

at java.util.concurrent.ThreadPoolExecutor.runWorker(ThreadPoolExecutor.java:1149)

at java.util.concurrent.ThreadPoolExecutor$Worker.run(ThreadPoolExecutor.java:624)

... 1 more

>>> rdd.take(4)

['1,Kristina,Chung,55,Pilot', '2,Paige,Chen,74,Teacher', 'NM,New Mexico', 'NY,New York']

>>> custfilter =rdd.map(lambda x: x.split(",") )

>>> print(custfilter.take(14))

[['1', 'Kristina', 'Chung', '55', 'Pilot'], ['2', 'Paige', 'Chen', '74', 'Teacher'], ['NM', 'New Mexico'], ['NY', 'New York'], ['NC', 'North Carolina'], ['ND', 'North Dakota'], ['OH', 'Ohio'], ['OK', 'Oklahoma'], ['OR', 'Oregon'], ['MD', 'Maryland'], ['MA', 'Massachusetts'], ['MI', 'Michigan'], ['MN', 'Minnesota'], ['MS', 'Mississippi']]

>>> splited\_rdd =rdd.map(lambda x: x.split(",") )

>>> custfilter =splited\_rdd.filter(lambda x: len(x)>3 )

>>> print(custfilter.take(4))

[['1', 'Kristina', 'Chung', '55', 'Pilot'], ['2', 'Paige', 'Chen', '74', 'Teacher'], ['3', 'Sherri', 'Melton', '34', 'Firefighter'], ['4', 'Gretchen', 'Hill', '66', 'Computer hardware engineer']]

>>> print(custfilter.take(5))

[['1', 'Kristina', 'Chung', '55', 'Pilot'], ['2', 'Paige', 'Chen', '74', 'Teacher'], ['3', 'Sherri', 'Melton', '34', 'Firefighter'], ['4', 'Gretchen', 'Hill', '66', 'Computer hardware engineer'], ['5', 'Karen', 'Puckett', '74', 'Lawyer']]

>>> statefilter =splited\_rdd.filter(lambda x: len(x)<3 )

>>> print(statefilter.take(4))

[['NM', 'New Mexico'], ['NY', 'New York'], ['NC', 'North Carolina'], ['ND', 'North Dakota']]

>>>

>>> complete\_df=spark.read.csv("/home/hduser/Desktop/Hackathon",sep=",",header=False,inferSchema=True)

Traceback (most recent call last):

File "<stdin>", line 1, in <module>

File "/usr/local/spark/python/pyspark/sql/readwriter.py", line 410, in csv

return self.\_df(self.\_jreader.csv(self.\_spark.\_sc.\_jvm.PythonUtils.toSeq(path)))

File "/usr/local/spark/python/lib/py4j-0.10.9.3-src.zip/py4j/java\_gateway.py", line 1322, in \_\_call\_\_

File "/usr/local/spark/python/pyspark/sql/utils.py", line 117, in deco

raise converted from None

pyspark.sql.utils.AnalysisException: Path does not exist: hdfs://localhost:54310/home/hduser/Desktop/Hackathon

>>> complete\_df=spark.read.csv("file:///home/hduser/Desktop/Hackathon",sep=",",header=False,inferSchema=True)

>>> complete\_df.show(3)

+--------+---------+------------+---------+----------+--------------------+--------------------+-------+--------------+--------------+

| \_c0| \_c1| \_c2| \_c3| \_c4| \_c5| \_c6| \_c7| \_c8| \_c9|

+--------+---------+------------+---------+----------+--------------------+--------------------+-------+--------------+--------------+

|IssuerId|IssuerId2|BusinessDate|StateCode|SourceName| NetworkName| NetworkURL|custnum|MarketCoverage|DentalOnlyPlan|

| 31195| 31195| 01-10-2019| SD| SERFF|Sanford Health Pl...|https://www.viiad...| 14| null| null|

| 60536| 60536| 01-10-2019| SD| SERFF| Avera Health Plans|www.Avera.org/Pro...| 13| null| null|

+--------+---------+------------+---------+----------+--------------------+--------------------+-------+--------------+--------------+

only showing top 3 rows

>>> complete\_df=spark.read.csv("file:///home/hduser/Desktop/Hackathon/custs\_states.csv",sep=",",header=False,inferSchema=True)

>>> complete\_df=spark.read.csv("file:///home/hduser/Desktop/Hackathon/custs\_states.csv",sep=",",header=False,inferSchema=True)

>>> complete\_df.show(3)

+---+----------+-----+----+-------+

|\_c0| \_c1| \_c2| \_c3| \_c4|

+---+----------+-----+----+-------+

| 1| Kristina|Chung| 55| Pilot|

| 2| Paige| Chen| 74|Teacher|

| NM|New Mexico| null|null| null|

+---+----------+-----+----+-------+

only showing top 3 rows

>>> custfilter\_df=complete\_df.select("\*").where("\_c0 is not null and \_c1 is not null and \_c2 is not null and \_c3 is not null and \_c4 is not null ")

>>> cust

custfilter custfilter\_df custom\_schema

>>> custfilter\_df.show(3)

+---+--------+------+---+-----------+

|\_c0| \_c1| \_c2|\_c3| \_c4|

+---+--------+------+---+-----------+

| 1|Kristina| Chung| 55| Pilot|

| 2| Paige| Chen| 74| Teacher|

| 3| Sherri|Melton| 34|Firefighter|

+---+--------+------+---+-----------+

only showing top 3 rows

>>> statefilter\_df=complete\_df.select("\_c0,\_c1").where(" \_c2 is null and \_c3 is null and \_c4 is null ")

Traceback (most recent call last):

File "<stdin>", line 1, in <module>

File "/usr/local/spark/python/pyspark/sql/dataframe.py", line 1685, in select

jdf = self.\_jdf.select(self.\_jcols(\*cols))

File "/usr/local/spark/python/lib/py4j-0.10.9.3-src.zip/py4j/java\_gateway.py", line 1322, in \_\_call\_\_

File "/usr/local/spark/python/pyspark/sql/utils.py", line 117, in deco

raise converted from None

pyspark.sql.utils.AnalysisException: cannot resolve '`\_c0,\_c1`' given input columns: [\_c0, \_c1, \_c2, \_c3, \_c4];

'Project ['\_c0,\_c1]

+- Relation [\_c0#26007,\_c1#26008,\_c2#26009,\_c3#26010,\_c4#26011] csv

>>> statefilter\_df=complete\_df.select("\_c0","\_c1").where(" \_c2 is null and \_c3 is null and \_c4 is null ")

>>> statefilter\_df.show(4)

+---+--------------+

|\_c0| \_c1|

+---+--------------+

| NM| New Mexico|

| NY| New York|

| NC|North Carolina|

| ND| North Dakota|

+---+--------------+

only showing top 4 rows

>>> custfilter\_df.createOrReplaceTempView("cust\_table")

>>> statefilter\_df.createOrReplaceTempView("state\_table")

>>> merged\_data\_frame.createOrReplaceTempView("insureview")

>>> df\_function("A+wer")

Column<'clean\_string(A+wer)'>

>>> df\_function("A+wer").show()

Traceback (most recent call last):

File "<stdin>", line 1, in <module>

TypeError: 'Column' object is not callable

>>> print(df\_function("A+wer"))

Column<'clean\_string(A+wer)'>

>>> print(df\_function(A+wer))

Traceback (most recent call last):

File "<stdin>", line 1, in <module>

NameError: name 'A' is not defined

>>> print(df\_function("A+wer"))

Column<'clean\_string(A+wer)'>

>>> print(df\_function("wer"))

Column<'clean\_string(wer)'>

>>> spark.conf.set("spark.sql.shuffle.partitions", 4)

>>> spark.udf.register("sql\_function", clean\_string, StringType())

<function clean\_string at 0x7fcc0c22b510>

>>> spark.sql("select \* from cust\_table").show(3)

+---+--------+------+---+-----------+

|\_c0| \_c1| \_c2|\_c3| \_c4|

+---+--------+------+---+-----------+

| 1|Kristina| Chung| 55| Pilot|

| 2| Paige| Chen| 74| Teacher|

| 3| Sherri|Melton| 34|Firefighter|

+---+--------+------+---+-----------+

only showing top 3 rows

>>> spark.sql("select \* from insureview").show(3)

+------------+----+-----+------------+--------------------+-------+--------------+--------------------+--------+---------+-----------------+----------+--------------------+

|BusinessDate|stcd|srcnm| NetworkName| NetworkURL|custnum|MarketCoverage| RejectRows|IssuerId|IssuerId2|issueridcomposite| sysdt| systs|

+------------+----+-----+------------+--------------------+-------+--------------+--------------------+--------+---------+-----------------+----------+--------------------+

| 2019-10-01| AK| HIOS| ODS Premier|https://www.modah...| 13| null| null| 21989| 21989| 2198921989|2024-07-22|2024-07-22 11:39:...|

| 2019-10-01| AK| HIOS| ODS Premier| null| null| null|219,219,2019-10-0...| 219| 219| 219219|2024-07-22|2024-07-22 11:39:...|

| 2019-10-01| AK| HIOS|HeritagePlus|https://www.preme...| 13| null| null| 38344| 38344| 3834438344|2024-07-22|2024-07-22 11:39:...|

+------------+----+-----+------------+--------------------+-------+--------------+--------------------+--------+---------+-----------------+----------+--------------------+

only showing top 3 rows

>>> spark.sql("select \*,sql\_function(NetworkName) as cleannetworkname from insureview").show(3)

+------------+----+-----+------------+--------------------+-------+--------------+--------------------+--------+---------+-----------------+----------+--------------------+----------------+

|BusinessDate|stcd|srcnm| NetworkName| NetworkURL|custnum|MarketCoverage| RejectRows|IssuerId|IssuerId2|issueridcomposite| sysdt| systs|cleannetworkname|

+------------+----+-----+------------+--------------------+-------+--------------+--------------------+--------+---------+-----------------+----------+--------------------+----------------+

| 2019-10-01| AK| HIOS| ODS Premier|https://www.modah...| 13| null| null| 21989| 21989| 2198921989|2024-07-22|2024-07-22 11:40:...| ODS Premier|

| 2019-10-01| AK| HIOS| ODS Premier| null| null| null|219,219,2019-10-0...| 219| 219| 219219|2024-07-22|2024-07-22 11:40:...| ODS Premier|

| 2019-10-01| AK| HIOS|HeritagePlus|https://www.preme...| 13| null| null| 38344| 38344| 3834438344|2024-07-22|2024-07-22 11:40:...| HeritagePlus|

+------------+----+-----+------------+--------------------+-------+--------------+--------------------+--------+---------+-----------------+----------+--------------------+----------------+

only showing top 3 rows

>>> spark.sql("select \*,current\_date as curdt , current\_timestamp as curts from insureview").show(3)

+------------+----+-----+------------+--------------------+-------+--------------+--------------------+--------+---------+-----------------+----------+--------------------+----------+--------------------+

|BusinessDate|stcd|srcnm| NetworkName| NetworkURL|custnum|MarketCoverage| RejectRows|IssuerId|IssuerId2|issueridcomposite| sysdt| systs| curdt| curts|

+------------+----+-----+------------+--------------------+-------+--------------+--------------------+--------+---------+-----------------+----------+--------------------+----------+--------------------+

| 2019-10-01| AK| HIOS| ODS Premier|https://www.modah...| 13| null| null| 21989| 21989| 2198921989|2024-07-22|2024-07-22 11:42:...|2024-07-22|2024-07-22 11:42:...|

| 2019-10-01| AK| HIOS| ODS Premier| null| null| null|219,219,2019-10-0...| 219| 219| 219219|2024-07-22|2024-07-22 11:42:...|2024-07-22|2024-07-22 11:42:...|

| 2019-10-01| AK| HIOS|HeritagePlus|https://www.preme...| 13| null| null| 38344| 38344| 3834438344|2024-07-22|2024-07-22 11:42:...|2024-07-22|2024-07-22 11:42:...|

+------------+----+-----+------------+--------------------+-------+--------------+--------------------+--------+---------+-----------------+----------+--------------------+----------+--------------------+

only showing top 3 rows

>>> spark.sql("select \*,BuisnessDate.month as month , BuisnessDate.year as year from insureview").show(3)

Traceback (most recent call last):

File "<stdin>", line 1, in <module>

File "/usr/local/spark/python/pyspark/sql/session.py", line 723, in sql

return DataFrame(self.\_jsparkSession.sql(sqlQuery), self.\_wrapped)

File "/usr/local/spark/python/lib/py4j-0.10.9.3-src.zip/py4j/java\_gateway.py", line 1322, in \_\_call\_\_

File "/usr/local/spark/python/pyspark/sql/utils.py", line 117, in deco

raise converted from None

pyspark.sql.utils.AnalysisException: cannot resolve 'BuisnessDate.month' given input columns: [insureview.BusinessDate, insureview.IssuerId, insureview.IssuerId2, insureview.MarketCoverage, insureview.NetworkName, insureview.NetworkURL, insureview.RejectRows, insureview.custnum, insureview.issueridcomposite, insureview.srcnm, insureview.stcd, insureview.sysdt, insureview.systs]; line 1 pos 9;

'Project [BusinessDate#1378, stcd#7759, srcnm#7747, NetworkName#1381, NetworkURL#1382, custnum#1383, MarketCoverage#1384, RejectRows#1386, IssuerId#8312, IssuerId2#8325, issueridcomposite#8901, sysdt#9003, systs#9016, 'BuisnessDate.month AS month#27694, 'BuisnessDate.year AS year#27695]

+- SubqueryAlias insureview

+- View (`insureview`, [BusinessDate#1378,stcd#7759,srcnm#7747,NetworkName#1381,NetworkURL#1382,custnum#1383,MarketCoverage#1384,RejectRows#1386,IssuerId#8312,IssuerId2#8325,issueridcomposite#8901,sysdt#9003,systs#9016])

+- Project [BusinessDate#1378, stcd#7759, srcnm#7747, NetworkName#1381, NetworkURL#1382, custnum#1383, MarketCoverage#1384, RejectRows#1386, IssuerId#8312, IssuerId2#8325, issueridcomposite#8901, sysdt#9003, current\_timestamp() AS systs#9016]

+- Project [BusinessDate#1378, stcd#7759, srcnm#7747, NetworkName#1381, NetworkURL#1382, custnum#1383, MarketCoverage#1384, RejectRows#1386, IssuerId#8312, IssuerId2#8325, issueridcomposite#8901, current\_date(Some(Asia/Kolkata)) AS sysdt#9003]

+- Project [BusinessDate#1378, stcd#7759, srcnm#7747, NetworkName#1381, NetworkURL#1382, custnum#1383, MarketCoverage#1384, RejectRows#1386, IssuerId#8312, IssuerId2#8325, issueridcomposite#8901]

+- Project [BusinessDate#1378, stcd#7759, srcnm#7747, NetworkName#1381, NetworkURL#1382, custnum#1383, MarketCoverage#1384, DentalOnlyPlan#1385, RejectRows#1386, IssuerId#8312, IssuerId2#8325, concat(IssuerId#8312, IssuerId2#8325) AS issueridcomposite#8901]

+- Project [BusinessDate#1378, stcd#7759, srcnm#7747, NetworkName#1381, NetworkURL#1382, custnum#1383, MarketCoverage#1384, DentalOnlyPlan#1385, RejectRows#1386, IssuerId#8312, IssuerId2#8325]

+- Project [IssuerIDDD#8273, BusinessDate#1378, stcd#7759, srcnm#7747, NetworkName#1381, NetworkURL#1382, custnum#1383, MarketCoverage#1384, DentalOnlyPlan#1385, RejectRows#1386, IssuerId#8312, IssuerId2#8325]

+- Project [IssuerIDD#8261, IssuerIDDD#8273, BusinessDate#1378, stcd#7759, srcnm#7747, NetworkName#1381, NetworkURL#1382, custnum#1383, MarketCoverage#1384, DentalOnlyPlan#1385, RejectRows#1386, IssuerId#8312, cast(IssuerIDDD#8273 as string) AS IssuerId2#8325]

+- Project [IssuerIDD#8261, IssuerIDDD#8273, BusinessDate#1378, stcd#7759, srcnm#7747, NetworkName#1381, NetworkURL#1382, custnum#1383, MarketCoverage#1384, DentalOnlyPlan#1385, RejectRows#1386, cast(IssuerIDD#8261 as string) AS IssuerId#8312]

+- Project [IssuerIDD#8261, IssuerId2#1377 AS IssuerIDDD#8273, BusinessDate#1378, stcd#7759, srcnm#7747, NetworkName#1381, NetworkURL#1382, custnum#1383, MarketCoverage#1384, DentalOnlyPlan#1385, RejectRows#1386]

+- Project [IssuerId#1376 AS IssuerIDD#8261, IssuerId2#1377, BusinessDate#1378, stcd#7759, srcnm#7747, NetworkName#1381, NetworkURL#1382, custnum#1383, MarketCoverage#1384, DentalOnlyPlan#1385, RejectRows#1386]

+- Project [IssuerId#1376, IssuerId2#1377, BusinessDate#1378, StateCode#1379 AS stcd#7759, srcnm#7747, NetworkName#1381, NetworkURL#1382, custnum#1383, MarketCoverage#1384, DentalOnlyPlan#1385, RejectRows#1386]

+- Project [IssuerId#1376, IssuerId2#1377, BusinessDate#1378, StateCode#1379, SourceName#1380 AS srcnm#7747, NetworkName#1381, NetworkURL#1382, custnum#1383, MarketCoverage#1384, DentalOnlyPlan#1385, RejectRows#1386]

+- Union false, false

:- Relation [IssuerId#1376,IssuerId2#1377,BusinessDate#1378,StateCode#1379,SourceName#1380,NetworkName#1381,NetworkURL#1382,custnum#1383,MarketCoverage#1384,DentalOnlyPlan#1385,RejectRows#1386] csv

+- Relation [IssuerId#491,IssuerId2#492,BusinessDate#493,StateCode#494,SourceName#495,NetworkName#496,NetworkURL#497,custnum#498,MarketCoverage#499,DentalOnlyPlan#500,RejectRows#501] csv

>>> spark.sql("select \*,BuisnessDate.month() as month , BuisnessDate.year() as year from insureview").show(3)

24/07/22 11:43:25 WARN metastore.ObjectStore: Failed to get database buisnessdate, returning NoSuchObjectException

Traceback (most recent call last):

File "<stdin>", line 1, in <module>

File "/usr/local/spark/python/pyspark/sql/session.py", line 723, in sql

return DataFrame(self.\_jsparkSession.sql(sqlQuery), self.\_wrapped)

File "/usr/local/spark/python/lib/py4j-0.10.9.3-src.zip/py4j/java\_gateway.py", line 1322, in \_\_call\_\_

File "/usr/local/spark/python/pyspark/sql/utils.py", line 117, in deco

raise converted from None

pyspark.sql.utils.AnalysisException: Undefined function: 'month'. This function is neither a registered temporary function nor a permanent function registered in the database 'BuisnessDate'.; line 1 pos 9

>>> spark.sql("select \*,month(BuisnessDate) as month ,year(BuisnessDate) as year from insureview").show(3)

Traceback (most recent call last):

File "<stdin>", line 1, in <module>

File "/usr/local/spark/python/pyspark/sql/session.py", line 723, in sql

return DataFrame(self.\_jsparkSession.sql(sqlQuery), self.\_wrapped)

File "/usr/local/spark/python/lib/py4j-0.10.9.3-src.zip/py4j/java\_gateway.py", line 1322, in \_\_call\_\_

File "/usr/local/spark/python/pyspark/sql/utils.py", line 117, in deco

raise converted from None

pyspark.sql.utils.AnalysisException: cannot resolve 'BuisnessDate' given input columns: [insureview.BusinessDate, insureview.IssuerId, insureview.IssuerId2, insureview.MarketCoverage, insureview.NetworkName, insureview.NetworkURL, insureview.RejectRows, insureview.custnum, insureview.issueridcomposite, insureview.srcnm, insureview.stcd, insureview.sysdt, insureview.systs]; line 1 pos 15;

'Project [BusinessDate#1378, stcd#7759, srcnm#7747, NetworkName#1381, NetworkURL#1382, custnum#1383, MarketCoverage#1384, RejectRows#1386, IssuerId#8312, IssuerId2#8325, issueridcomposite#8901, sysdt#9003, systs#9016, 'month('BuisnessDate) AS month#27698, 'year('BuisnessDate) AS year#27699]

+- SubqueryAlias insureview

+- View (`insureview`, [BusinessDate#1378,stcd#7759,srcnm#7747,NetworkName#1381,NetworkURL#1382,custnum#1383,MarketCoverage#1384,RejectRows#1386,IssuerId#8312,IssuerId2#8325,issueridcomposite#8901,sysdt#9003,systs#9016])

+- Project [BusinessDate#1378, stcd#7759, srcnm#7747, NetworkName#1381, NetworkURL#1382, custnum#1383, MarketCoverage#1384, RejectRows#1386, IssuerId#8312, IssuerId2#8325, issueridcomposite#8901, sysdt#9003, current\_timestamp() AS systs#9016]

+- Project [BusinessDate#1378, stcd#7759, srcnm#7747, NetworkName#1381, NetworkURL#1382, custnum#1383, MarketCoverage#1384, RejectRows#1386, IssuerId#8312, IssuerId2#8325, issueridcomposite#8901, current\_date(Some(Asia/Kolkata)) AS sysdt#9003]

+- Project [BusinessDate#1378, stcd#7759, srcnm#7747, NetworkName#1381, NetworkURL#1382, custnum#1383, MarketCoverage#1384, RejectRows#1386, IssuerId#8312, IssuerId2#8325, issueridcomposite#8901]

+- Project [BusinessDate#1378, stcd#7759, srcnm#7747, NetworkName#1381, NetworkURL#1382, custnum#1383, MarketCoverage#1384, DentalOnlyPlan#1385, RejectRows#1386, IssuerId#8312, IssuerId2#8325, concat(IssuerId#8312, IssuerId2#8325) AS issueridcomposite#8901]

+- Project [BusinessDate#1378, stcd#7759, srcnm#7747, NetworkName#1381, NetworkURL#1382, custnum#1383, MarketCoverage#1384, DentalOnlyPlan#1385, RejectRows#1386, IssuerId#8312, IssuerId2#8325]

+- Project [IssuerIDDD#8273, BusinessDate#1378, stcd#7759, srcnm#7747, NetworkName#1381, NetworkURL#1382, custnum#1383, MarketCoverage#1384, DentalOnlyPlan#1385, RejectRows#1386, IssuerId#8312, IssuerId2#8325]

+- Project [IssuerIDD#8261, IssuerIDDD#8273, BusinessDate#1378, stcd#7759, srcnm#7747, NetworkName#1381, NetworkURL#1382, custnum#1383, MarketCoverage#1384, DentalOnlyPlan#1385, RejectRows#1386, IssuerId#8312, cast(IssuerIDDD#8273 as string) AS IssuerId2#8325]

+- Project [IssuerIDD#8261, IssuerIDDD#8273, BusinessDate#1378, stcd#7759, srcnm#7747, NetworkName#1381, NetworkURL#1382, custnum#1383, MarketCoverage#1384, DentalOnlyPlan#1385, RejectRows#1386, cast(IssuerIDD#8261 as string) AS IssuerId#8312]

+- Project [IssuerIDD#8261, IssuerId2#1377 AS IssuerIDDD#8273, BusinessDate#1378, stcd#7759, srcnm#7747, NetworkName#1381, NetworkURL#1382, custnum#1383, MarketCoverage#1384, DentalOnlyPlan#1385, RejectRows#1386]

+- Project [IssuerId#1376 AS IssuerIDD#8261, IssuerId2#1377, BusinessDate#1378, stcd#7759, srcnm#7747, NetworkName#1381, NetworkURL#1382, custnum#1383, MarketCoverage#1384, DentalOnlyPlan#1385, RejectRows#1386]

+- Project [IssuerId#1376, IssuerId2#1377, BusinessDate#1378, StateCode#1379 AS stcd#7759, srcnm#7747, NetworkName#1381, NetworkURL#1382, custnum#1383, MarketCoverage#1384, DentalOnlyPlan#1385, RejectRows#1386]

+- Project [IssuerId#1376, IssuerId2#1377, BusinessDate#1378, StateCode#1379, SourceName#1380 AS srcnm#7747, NetworkName#1381, NetworkURL#1382, custnum#1383, MarketCoverage#1384, DentalOnlyPlan#1385, RejectRows#1386]

+- Union false, false

:- Relation [IssuerId#1376,IssuerId2#1377,BusinessDate#1378,StateCode#1379,SourceName#1380,NetworkName#1381,NetworkURL#1382,custnum#1383,MarketCoverage#1384,DentalOnlyPlan#1385,RejectRows#1386] csv

+- Relation [IssuerId#491,IssuerId2#492,BusinessDate#493,StateCode#494,SourceName#495,NetworkName#496,NetworkURL#497,custnum#498,MarketCoverage#499,DentalOnlyPlan#500,RejectRows#501] csv

>>> spark.sql("select \*,month(BuisnessDate) as month ,year(BuisnessDate) as year from insureview").show(3

...

Traceback (most recent call last):

File "/usr/local/spark/python/pyspark/context.py", line 293, in signal\_handler

raise KeyboardInterrupt()

KeyboardInterrupt

>>> spark.sql("describe insureview")

DataFrame[col\_name: string, data\_type: string, comment: string]

>>> spark.sql("describe insureview").show()

+-----------------+---------+-------+

| col\_name|data\_type|comment|

+-----------------+---------+-------+

| BusinessDate| date| null|

| stcd| string| null|

| srcnm| string| null|

| NetworkName| string| null|

| NetworkURL| string| null|

| custnum| string| null|

| MarketCoverage| string| null|

| RejectRows| string| null|

| IssuerId| string| null|

| IssuerId2| string| null|

|issueridcomposite| string| null|

| sysdt| date| null|

| systs|timestamp| null|

+-----------------+---------+-------+

>>> spark.sql("""select \*,month(BuisnessDate) as month ,year(BuisnessDate) as year from insureview""").show(3

... )

Traceback (most recent call last):

File "<stdin>", line 1, in <module>

File "/usr/local/spark/python/pyspark/sql/session.py", line 723, in sql

return DataFrame(self.\_jsparkSession.sql(sqlQuery), self.\_wrapped)

File "/usr/local/spark/python/lib/py4j-0.10.9.3-src.zip/py4j/java\_gateway.py", line 1322, in \_\_call\_\_

File "/usr/local/spark/python/pyspark/sql/utils.py", line 117, in deco

raise converted from None

pyspark.sql.utils.AnalysisException: cannot resolve 'BuisnessDate' given input columns: [insureview.BusinessDate, insureview.IssuerId, insureview.IssuerId2, insureview.MarketCoverage, insureview.NetworkName, insureview.NetworkURL, insureview.RejectRows, insureview.custnum, insureview.issueridcomposite, insureview.srcnm, insureview.stcd, insureview.sysdt, insureview.systs]; line 1 pos 15;

'Project [BusinessDate#1378, stcd#7759, srcnm#7747, NetworkName#1381, NetworkURL#1382, custnum#1383, MarketCoverage#1384, RejectRows#1386, IssuerId#8312, IssuerId2#8325, issueridcomposite#8901, sysdt#9003, systs#9016, 'month('BuisnessDate) AS month#27725, 'year('BuisnessDate) AS year#27726]

+- SubqueryAlias insureview

+- View (`insureview`, [BusinessDate#1378,stcd#7759,srcnm#7747,NetworkName#1381,NetworkURL#1382,custnum#1383,MarketCoverage#1384,RejectRows#1386,IssuerId#8312,IssuerId2#8325,issueridcomposite#8901,sysdt#9003,systs#9016])

+- Project [BusinessDate#1378, stcd#7759, srcnm#7747, NetworkName#1381, NetworkURL#1382, custnum#1383, MarketCoverage#1384, RejectRows#1386, IssuerId#8312, IssuerId2#8325, issueridcomposite#8901, sysdt#9003, current\_timestamp() AS systs#9016]

+- Project [BusinessDate#1378, stcd#7759, srcnm#7747, NetworkName#1381, NetworkURL#1382, custnum#1383, MarketCoverage#1384, RejectRows#1386, IssuerId#8312, IssuerId2#8325, issueridcomposite#8901, current\_date(Some(Asia/Kolkata)) AS sysdt#9003]

+- Project [BusinessDate#1378, stcd#7759, srcnm#7747, NetworkName#1381, NetworkURL#1382, custnum#1383, MarketCoverage#1384, RejectRows#1386, IssuerId#8312, IssuerId2#8325, issueridcomposite#8901]

+- Project [BusinessDate#1378, stcd#7759, srcnm#7747, NetworkName#1381, NetworkURL#1382, custnum#1383, MarketCoverage#1384, DentalOnlyPlan#1385, RejectRows#1386, IssuerId#8312, IssuerId2#8325, concat(IssuerId#8312, IssuerId2#8325) AS issueridcomposite#8901]

+- Project [BusinessDate#1378, stcd#7759, srcnm#7747, NetworkName#1381, NetworkURL#1382, custnum#1383, MarketCoverage#1384, DentalOnlyPlan#1385, RejectRows#1386, IssuerId#8312, IssuerId2#8325]

+- Project [IssuerIDDD#8273, BusinessDate#1378, stcd#7759, srcnm#7747, NetworkName#1381, NetworkURL#1382, custnum#1383, MarketCoverage#1384, DentalOnlyPlan#1385, RejectRows#1386, IssuerId#8312, IssuerId2#8325]

+- Project [IssuerIDD#8261, IssuerIDDD#8273, BusinessDate#1378, stcd#7759, srcnm#7747, NetworkName#1381, NetworkURL#1382, custnum#1383, MarketCoverage#1384, DentalOnlyPlan#1385, RejectRows#1386, IssuerId#8312, cast(IssuerIDDD#8273 as string) AS IssuerId2#8325]

+- Project [IssuerIDD#8261, IssuerIDDD#8273, BusinessDate#1378, stcd#7759, srcnm#7747, NetworkName#1381, NetworkURL#1382, custnum#1383, MarketCoverage#1384, DentalOnlyPlan#1385, RejectRows#1386, cast(IssuerIDD#8261 as string) AS IssuerId#8312]

+- Project [IssuerIDD#8261, IssuerId2#1377 AS IssuerIDDD#8273, BusinessDate#1378, stcd#7759, srcnm#7747, NetworkName#1381, NetworkURL#1382, custnum#1383, MarketCoverage#1384, DentalOnlyPlan#1385, RejectRows#1386]

+- Project [IssuerId#1376 AS IssuerIDD#8261, IssuerId2#1377, BusinessDate#1378, stcd#7759, srcnm#7747, NetworkName#1381, NetworkURL#1382, custnum#1383, MarketCoverage#1384, DentalOnlyPlan#1385, RejectRows#1386]

+- Project [IssuerId#1376, IssuerId2#1377, BusinessDate#1378, StateCode#1379 AS stcd#7759, srcnm#7747, NetworkName#1381, NetworkURL#1382, custnum#1383, MarketCoverage#1384, DentalOnlyPlan#1385, RejectRows#1386]

+- Project [IssuerId#1376, IssuerId2#1377, BusinessDate#1378, StateCode#1379, SourceName#1380 AS srcnm#7747, NetworkName#1381, NetworkURL#1382, custnum#1383, MarketCoverage#1384, DentalOnlyPlan#1385, RejectRows#1386]

+- Union false, false

:- Relation [IssuerId#1376,IssuerId2#1377,BusinessDate#1378,StateCode#1379,SourceName#1380,NetworkName#1381,NetworkURL#1382,custnum#1383,MarketCoverage#1384,DentalOnlyPlan#1385,RejectRows#1386] csv

+- Relation [IssuerId#491,IssuerId2#492,BusinessDate#493,StateCode#494,SourceName#495,NetworkName#496,NetworkURL#497,custnum#498,MarketCoverage#499,DentalOnlyPlan#500,RejectRows#501] csv

>>> spark.sql("""select \*,month(BuisnessDate) as month ,year(BuisnessDate) as year from insureview""").show(3)

Traceback (most recent call last):

File "<stdin>", line 1, in <module>

File "/usr/local/spark/python/pyspark/sql/session.py", line 723, in sql

return DataFrame(self.\_jsparkSession.sql(sqlQuery), self.\_wrapped)

File "/usr/local/spark/python/lib/py4j-0.10.9.3-src.zip/py4j/java\_gateway.py", line 1322, in \_\_call\_\_

File "/usr/local/spark/python/pyspark/sql/utils.py", line 117, in deco

raise converted from None

pyspark.sql.utils.AnalysisException: cannot resolve 'BuisnessDate' given input columns: [insureview.BusinessDate, insureview.IssuerId, insureview.IssuerId2, insureview.MarketCoverage, insureview.NetworkName, insureview.NetworkURL, insureview.RejectRows, insureview.custnum, insureview.issueridcomposite, insureview.srcnm, insureview.stcd, insureview.sysdt, insureview.systs]; line 1 pos 15;

'Project [BusinessDate#1378, stcd#7759, srcnm#7747, NetworkName#1381, NetworkURL#1382, custnum#1383, MarketCoverage#1384, RejectRows#1386, IssuerId#8312, IssuerId2#8325, issueridcomposite#8901, sysdt#9003, systs#9016, 'month('BuisnessDate) AS month#27727, 'year('BuisnessDate) AS year#27728]

+- SubqueryAlias insureview

+- View (`insureview`, [BusinessDate#1378,stcd#7759,srcnm#7747,NetworkName#1381,NetworkURL#1382,custnum#1383,MarketCoverage#1384,RejectRows#1386,IssuerId#8312,IssuerId2#8325,issueridcomposite#8901,sysdt#9003,systs#9016])

+- Project [BusinessDate#1378, stcd#7759, srcnm#7747, NetworkName#1381, NetworkURL#1382, custnum#1383, MarketCoverage#1384, RejectRows#1386, IssuerId#8312, IssuerId2#8325, issueridcomposite#8901, sysdt#9003, current\_timestamp() AS systs#9016]

+- Project [BusinessDate#1378, stcd#7759, srcnm#7747, NetworkName#1381, NetworkURL#1382, custnum#1383, MarketCoverage#1384, RejectRows#1386, IssuerId#8312, IssuerId2#8325, issueridcomposite#8901, current\_date(Some(Asia/Kolkata)) AS sysdt#9003]

+- Project [BusinessDate#1378, stcd#7759, srcnm#7747, NetworkName#1381, NetworkURL#1382, custnum#1383, MarketCoverage#1384, RejectRows#1386, IssuerId#8312, IssuerId2#8325, issueridcomposite#8901]

+- Project [BusinessDate#1378, stcd#7759, srcnm#7747, NetworkName#1381, NetworkURL#1382, custnum#1383, MarketCoverage#1384, DentalOnlyPlan#1385, RejectRows#1386, IssuerId#8312, IssuerId2#8325, concat(IssuerId#8312, IssuerId2#8325) AS issueridcomposite#8901]

+- Project [BusinessDate#1378, stcd#7759, srcnm#7747, NetworkName#1381, NetworkURL#1382, custnum#1383, MarketCoverage#1384, DentalOnlyPlan#1385, RejectRows#1386, IssuerId#8312, IssuerId2#8325]

+- Project [IssuerIDDD#8273, BusinessDate#1378, stcd#7759, srcnm#7747, NetworkName#1381, NetworkURL#1382, custnum#1383, MarketCoverage#1384, DentalOnlyPlan#1385, RejectRows#1386, IssuerId#8312, IssuerId2#8325]

+- Project [IssuerIDD#8261, IssuerIDDD#8273, BusinessDate#1378, stcd#7759, srcnm#7747, NetworkName#1381, NetworkURL#1382, custnum#1383, MarketCoverage#1384, DentalOnlyPlan#1385, RejectRows#1386, IssuerId#8312, cast(IssuerIDDD#8273 as string) AS IssuerId2#8325]

+- Project [IssuerIDD#8261, IssuerIDDD#8273, BusinessDate#1378, stcd#7759, srcnm#7747, NetworkName#1381, NetworkURL#1382, custnum#1383, MarketCoverage#1384, DentalOnlyPlan#1385, RejectRows#1386, cast(IssuerIDD#8261 as string) AS IssuerId#8312]

+- Project [IssuerIDD#8261, IssuerId2#1377 AS IssuerIDDD#8273, BusinessDate#1378, stcd#7759, srcnm#7747, NetworkName#1381, NetworkURL#1382, custnum#1383, MarketCoverage#1384, DentalOnlyPlan#1385, RejectRows#1386]

+- Project [IssuerId#1376 AS IssuerIDD#8261, IssuerId2#1377, BusinessDate#1378, stcd#7759, srcnm#7747, NetworkName#1381, NetworkURL#1382, custnum#1383, MarketCoverage#1384, DentalOnlyPlan#1385, RejectRows#1386]

+- Project [IssuerId#1376, IssuerId2#1377, BusinessDate#1378, StateCode#1379 AS stcd#7759, srcnm#7747, NetworkName#1381, NetworkURL#1382, custnum#1383, MarketCoverage#1384, DentalOnlyPlan#1385, RejectRows#1386]

+- Project [IssuerId#1376, IssuerId2#1377, BusinessDate#1378, StateCode#1379, SourceName#1380 AS srcnm#7747, NetworkName#1381, NetworkURL#1382, custnum#1383, MarketCoverage#1384, DentalOnlyPlan#1385, RejectRows#1386]

+- Union false, false

:- Relation [IssuerId#1376,IssuerId2#1377,BusinessDate#1378,StateCode#1379,SourceName#1380,NetworkName#1381,NetworkURL#1382,custnum#1383,MarketCoverage#1384,DentalOnlyPlan#1385,RejectRows#1386] csv

+- Relation [IssuerId#491,IssuerId2#492,BusinessDate#493,StateCode#494,SourceName#495,NetworkName#496,NetworkURL#497,custnum#498,MarketCoverage#499,DentalOnlyPlan#500,RejectRows#501] csv

>>> spark.sql("""

... SELECT \*,

... month(BusinessDate) AS month,

... year(BusinessDate) AS year

... FROM insureview

... """)

DataFrame[BusinessDate: date, stcd: string, srcnm: string, NetworkName: string, NetworkURL: string, custnum: string, MarketCoverage: string, RejectRows: string, IssuerId: string, IssuerId2: string, issueridcomposite: string, sysdt: date, systs: timestamp, month: int, year: int]

>>> result\_df.show(3)

Traceback (most recent call last):

File "<stdin>", line 1, in <module>

NameError: name 'result\_df' is not defined

>>> spark.sql("""spark.sql("""SELECT \*, month(BusinessDate) AS month, year(BusinessDate) AS year FROM insureview""")

File "<stdin>", line 1

spark.sql("""spark.sql("""SELECT \*, month(BusinessDate) AS month, year(BusinessDate) AS year FROM insureview""")

^

SyntaxError: invalid syntax

>>> spark.sql("""SELECT \*, month(BusinessDate) AS month, year(BusinessDate) AS year FROM insureview""")

DataFrame[BusinessDate: date, stcd: string, srcnm: string, NetworkName: string, NetworkURL: string, custnum: string, MarketCoverage: string, RejectRows: string, IssuerId: string, IssuerId2: string, issueridcomposite: string, sysdt: date, systs: timestamp, month: int, year: int]

>>> spark.sql("""SELECT \*, month(BusinessDate) AS month, year(BusinessDate) AS year FROM insureview""").show()

+------------+----+-----+--------------------+--------------------+-------+--------------+--------------------+--------+---------+-----------------+----------+--------------------+-----+----+

|BusinessDate|stcd|srcnm| NetworkName| NetworkURL|custnum|MarketCoverage| RejectRows|IssuerId|IssuerId2|issueridcomposite| sysdt| systs|month|year|

+------------+----+-----+--------------------+--------------------+-------+--------------+--------------------+--------+---------+-----------------+----------+--------------------+-----+----+

| 2019-10-01| AK| HIOS| ODS Premier|https://www.modah...| 13| null| null| 21989| 21989| 2198921989|2024-07-22|2024-07-22 11:47:...| 10|2019|

| 2019-10-01| AK| HIOS| ODS Premier| null| null| null|219,219,2019-10-0...| 219| 219| 219219|2024-07-22|2024-07-22 11:47:...| 10|2019|

| 2019-10-01| AK| HIOS| HeritagePlus|https://www.preme...| 13| null| null| 38344| 38344| 3834438344|2024-07-22|2024-07-22 11:47:...| 10|2019|

| 2019-10-01| AK| HIOS|Lincoln Dental Co...|http://lfg.go2den...| 13| null| null| 38536| 38536| 3853638536|2024-07-22|2024-07-22 11:47:...| 10|2019|

| 2019-10-01| AK| HIOS|DentalGuard Prefe...|https://www.guard...| 13| null| null| 42507| 42507| 4250742507|2024-07-22|2024-07-22 11:47:...| 10|2019|

| 2019-10-01| AK| HIOS|Moda Plus AK Regi...|https://www.modah...| 13| null| null| 73836| 73836| 7383673836|2024-07-22|2024-07-22 11:47:...| 10|2019|

| 2019-10-01| AK| HIOS|Moda Plus Providence|https://www.modah...| 14| null| null| 73836| 73836| 7383673836|2024-07-22|2024-07-22 11:47:...| 10|2019|

| 2019-10-01| AK| HIOS| DenteMax|http://www2.dente...| 13| null| null| 74819| 74819| 7481974819|2024-07-22|2024-07-22 11:47:...| 10|2019|

| 2019-10-01| AK| HIOS|Ameritas PPO Dent...| www.standard.com| 13| null| null| 84859| 84859| 8485984859|2024-07-22|2024-07-22 11:47:...| 10|2019|

| 2019-10-01| AL| HIOS| DenteMax|http://www2.dente...| 13| null| null| 12538| 12538| 1253812538|2024-07-22|2024-07-22 11:47:...| 10|2019|

| 2019-10-01| AL| HIOS|CONNECTION Dental...|http://www.ppousa...| 14| null| null| 12538| 12538| 1253812538|2024-07-22|2024-07-22 11:47:...| 10|2019|

| 2019-10-01| AL| HIOS|DentalGuard Prefe...|https://www.guard...| 13| null| null| 17540| 17540| 1754017540|2024-07-22|2024-07-22 11:47:...| 10|2019|

| 2019-10-01| AL| HIOS| Birmingham PPOx|http://pfp.humana...| 13| null| null| 44580| 44580| 4458044580|2024-07-22|2024-07-22 11:47:...| 10|2019|

| 2019-10-01| AL| HIOS| Huntsville PPOx|http://pfp.humana...| 14| null| null| 44580| 44580| 4458044580|2024-07-22|2024-07-22 11:47:...| 10|2019|

| 2019-10-01| AL| HIOS|Humana Dental PPO...|https://www.human...| 15| null| null| 44580| 44580| 4458044580|2024-07-22|2024-07-22 11:47:...| 10|2019|

| 2019-10-01| AL| HIOS| Advantage Plus|https://secure.uc...| 13| null| null| 44785| 44785| 4478544785|2024-07-22|2024-07-22 11:47:...| 10|2019|

| 2019-10-01| AL| HIOS|Lincoln Dental Co...|http://lfg.go2den...| 13| null| null| 46211| 46211| 4621146211|2024-07-22|2024-07-22 11:47:...| 10|2019|

| 2019-10-01| AL| HIOS| BCBSAL PPO|https://www.bcbsa...| 13| null| null| 46944| 46944| 4694446944|2024-07-22|2024-07-22 11:47:...| 10|2019|

| 2019-10-01| AL| HIOS| Choice Plus|http://www.provid...| 13| null| null| 59809| 59809| 5980959809|2024-07-22|2024-07-22 11:47:...| 10|2019|

| 2019-10-01| AL| HIOS| Choice Plus|http://www.provid...| 14| null| null| 59809| 59809| 5980959809|2024-07-22|2024-07-22 11:47:...| 10|2019|

+------------+----+-----+--------------------+--------------------+-------+--------------+--------------------+--------+---------+-----------------+----------+--------------------+-----+----+

only showing top 20 rows

>>> spark.sql("""SELECT \*, month(BusinessDate) AS month, year(BusinessDate) AS year FROM insureview""").show(4)

+------------+----+-----+--------------------+--------------------+-------+--------------+--------------------+--------+---------+-----------------+----------+--------------------+-----+----+

|BusinessDate|stcd|srcnm| NetworkName| NetworkURL|custnum|MarketCoverage| RejectRows|IssuerId|IssuerId2|issueridcomposite| sysdt| systs|month|year|

+------------+----+-----+--------------------+--------------------+-------+--------------+--------------------+--------+---------+-----------------+----------+--------------------+-----+----+

| 2019-10-01| AK| HIOS| ODS Premier|https://www.modah...| 13| null| null| 21989| 21989| 2198921989|2024-07-22|2024-07-22 11:48:...| 10|2019|

| 2019-10-01| AK| HIOS| ODS Premier| null| null| null|219,219,2019-10-0...| 219| 219| 219219|2024-07-22|2024-07-22 11:48:...| 10|2019|

| 2019-10-01| AK| HIOS| HeritagePlus|https://www.preme...| 13| null| null| 38344| 38344| 3834438344|2024-07-22|2024-07-22 11:48:...| 10|2019|

| 2019-10-01| AK| HIOS|Lincoln Dental Co...|http://lfg.go2den...| 13| null| null| 38536| 38536| 3853638536|2024-07-22|2024-07-22 11:48:...| 10|2019|

+------------+----+-----+--------------------+--------------------+-------+--------------+--------------------+--------+---------+-----------------+----------+--------------------+-----+----+

only showing top 4 rows

>>> spark.sql("""select \*,month(BuisnessDate) as month ,year(BuisnessDate) as year from insureview""").show(3)

Traceback (most recent call last):

File "<stdin>", line 1, in <module>

File "/usr/local/spark/python/pyspark/sql/session.py", line 723, in sql

return DataFrame(self.\_jsparkSession.sql(sqlQuery), self.\_wrapped)

File "/usr/local/spark/python/lib/py4j-0.10.9.3-src.zip/py4j/java\_gateway.py", line 1322, in \_\_call\_\_

File "/usr/local/spark/python/pyspark/sql/utils.py", line 117, in deco

raise converted from None

pyspark.sql.utils.AnalysisException: cannot resolve 'BuisnessDate' given input columns: [insureview.BusinessDate, insureview.IssuerId, insureview.IssuerId2, insureview.MarketCoverage, insureview.NetworkName, insureview.NetworkURL, insureview.RejectRows, insureview.custnum, insureview.issueridcomposite, insureview.srcnm, insureview.stcd, insureview.sysdt, insureview.systs]; line 1 pos 15;

'Project [BusinessDate#1378, stcd#7759, srcnm#7747, NetworkName#1381, NetworkURL#1382, custnum#1383, MarketCoverage#1384, RejectRows#1386, IssuerId#8312, IssuerId2#8325, issueridcomposite#8901, sysdt#9003, systs#9016, 'month('BuisnessDate) AS month#28819, 'year('BuisnessDate) AS year#28820]

+- SubqueryAlias insureview

+- View (`insureview`, [BusinessDate#1378,stcd#7759,srcnm#7747,NetworkName#1381,NetworkURL#1382,custnum#1383,MarketCoverage#1384,RejectRows#1386,IssuerId#8312,IssuerId2#8325,issueridcomposite#8901,sysdt#9003,systs#9016])

+- Project [BusinessDate#1378, stcd#7759, srcnm#7747, NetworkName#1381, NetworkURL#1382, custnum#1383, MarketCoverage#1384, RejectRows#1386, IssuerId#8312, IssuerId2#8325, issueridcomposite#8901, sysdt#9003, current\_timestamp() AS systs#9016]

+- Project [BusinessDate#1378, stcd#7759, srcnm#7747, NetworkName#1381, NetworkURL#1382, custnum#1383, MarketCoverage#1384, RejectRows#1386, IssuerId#8312, IssuerId2#8325, issueridcomposite#8901, current\_date(Some(Asia/Kolkata)) AS sysdt#9003]

+- Project [BusinessDate#1378, stcd#7759, srcnm#7747, NetworkName#1381, NetworkURL#1382, custnum#1383, MarketCoverage#1384, RejectRows#1386, IssuerId#8312, IssuerId2#8325, issueridcomposite#8901]

+- Project [BusinessDate#1378, stcd#7759, srcnm#7747, NetworkName#1381, NetworkURL#1382, custnum#1383, MarketCoverage#1384, DentalOnlyPlan#1385, RejectRows#1386, IssuerId#8312, IssuerId2#8325, concat(IssuerId#8312, IssuerId2#8325) AS issueridcomposite#8901]

+- Project [BusinessDate#1378, stcd#7759, srcnm#7747, NetworkName#1381, NetworkURL#1382, custnum#1383, MarketCoverage#1384, DentalOnlyPlan#1385, RejectRows#1386, IssuerId#8312, IssuerId2#8325]

+- Project [IssuerIDDD#8273, BusinessDate#1378, stcd#7759, srcnm#7747, NetworkName#1381, NetworkURL#1382, custnum#1383, MarketCoverage#1384, DentalOnlyPlan#1385, RejectRows#1386, IssuerId#8312, IssuerId2#8325]

+- Project [IssuerIDD#8261, IssuerIDDD#8273, BusinessDate#1378, stcd#7759, srcnm#7747, NetworkName#1381, NetworkURL#1382, custnum#1383, MarketCoverage#1384, DentalOnlyPlan#1385, RejectRows#1386, IssuerId#8312, cast(IssuerIDDD#8273 as string) AS IssuerId2#8325]

+- Project [IssuerIDD#8261, IssuerIDDD#8273, BusinessDate#1378, stcd#7759, srcnm#7747, NetworkName#1381, NetworkURL#1382, custnum#1383, MarketCoverage#1384, DentalOnlyPlan#1385, RejectRows#1386, cast(IssuerIDD#8261 as string) AS IssuerId#8312]

+- Project [IssuerIDD#8261, IssuerId2#1377 AS IssuerIDDD#8273, BusinessDate#1378, stcd#7759, srcnm#7747, NetworkName#1381, NetworkURL#1382, custnum#1383, MarketCoverage#1384, DentalOnlyPlan#1385, RejectRows#1386]

+- Project [IssuerId#1376 AS IssuerIDD#8261, IssuerId2#1377, BusinessDate#1378, stcd#7759, srcnm#7747, NetworkName#1381, NetworkURL#1382, custnum#1383, MarketCoverage#1384, DentalOnlyPlan#1385, RejectRows#1386]

+- Project [IssuerId#1376, IssuerId2#1377, BusinessDate#1378, StateCode#1379 AS stcd#7759, srcnm#7747, NetworkName#1381, NetworkURL#1382, custnum#1383, MarketCoverage#1384, DentalOnlyPlan#1385, RejectRows#1386]

+- Project [IssuerId#1376, IssuerId2#1377, BusinessDate#1378, StateCode#1379, SourceName#1380 AS srcnm#7747, NetworkName#1381, NetworkURL#1382, custnum#1383, MarketCoverage#1384, DentalOnlyPlan#1385, RejectRows#1386]

+- Union false, false

:- Relation [IssuerId#1376,IssuerId2#1377,BusinessDate#1378,StateCode#1379,SourceName#1380,NetworkName#1381,NetworkURL#1382,custnum#1383,MarketCoverage#1384,DentalOnlyPlan#1385,RejectRows#1386] csv

+- Relation [IssuerId#491,IssuerId2#492,BusinessDate#493,StateCode#494,SourceName#495,NetworkName#496,NetworkURL#497,custnum#498,MarketCoverage#499,DentalOnlyPlan#500,RejectRows#501] csv

>>> spark.sql("""SELECT \*, month(BusinessDate) AS month, year(BusinessDate) AS year FROM insureview""").show(4)

+------------+----+-----+--------------------+--------------------+-------+--------------+--------------------+--------+---------+-----------------+----------+--------------------+-----+----+

|BusinessDate|stcd|srcnm| NetworkName| NetworkURL|custnum|MarketCoverage| RejectRows|IssuerId|IssuerId2|issueridcomposite| sysdt| systs|month|year|

+------------+----+-----+--------------------+--------------------+-------+--------------+--------------------+--------+---------+-----------------+----------+--------------------+-----+----+

| 2019-10-01| AK| HIOS| ODS Premier|https://www.modah...| 13| null| null| 21989| 21989| 2198921989|2024-07-22|2024-07-22 11:48:...| 10|2019|

| 2019-10-01| AK| HIOS| ODS Premier| null| null| null|219,219,2019-10-0...| 219| 219| 219219|2024-07-22|2024-07-22 11:48:...| 10|2019|

| 2019-10-01| AK| HIOS| HeritagePlus|https://www.preme...| 13| null| null| 38344| 38344| 3834438344|2024-07-22|2024-07-22 11:48:...| 10|2019|

| 2019-10-01| AK| HIOS|Lincoln Dental Co...|http://lfg.go2den...| 13| null| null| 38536| 38536| 3853638536|2024-07-22|2024-07-22 11:48:...| 10|2019|

+------------+----+-----+--------------------+--------------------+-------+--------------+--------------------+--------+---------+-----------------+----------+--------------------+-----+----+

only showing top 4 rows

>>> spark.sql("""select \*,month(BusinessDate) as month ,year(BusinessDate) as year from insureview""").show(3)

+------------+----+-----+------------+--------------------+-------+--------------+--------------------+--------+---------+-----------------+----------+--------------------+-----+----+

|BusinessDate|stcd|srcnm| NetworkName| NetworkURL|custnum|MarketCoverage| RejectRows|IssuerId|IssuerId2|issueridcomposite| sysdt| systs|month|year|

+------------+----+-----+------------+--------------------+-------+--------------+--------------------+--------+---------+-----------------+----------+--------------------+-----+----+

| 2019-10-01| AK| HIOS| ODS Premier|https://www.modah...| 13| null| null| 21989| 21989| 2198921989|2024-07-22|2024-07-22 11:49:...| 10|2019|

| 2019-10-01| AK| HIOS| ODS Premier| null| null| null|219,219,2019-10-0...| 219| 219| 219219|2024-07-22|2024-07-22 11:49:...| 10|2019|

| 2019-10-01| AK| HIOS|HeritagePlus|https://www.preme...| 13| null| null| 38344| 38344| 3834438344|2024-07-22|2024-07-22 11:49:...| 10|2019|

+------------+----+-----+------------+--------------------+-------+--------------+--------------------+--------+---------+-----------------+----------+--------------------+-----+----+

only showing top 3 rows

>>> spark.sql("""select \*,case when NetworkURL like 'https://' then 'http secured' case when NetworkURL like 'http://' then 'http non secured' else 'No Protocol' as Protocol\_Used from insureview""").show(3)

Traceback (most recent call last):

File "<stdin>", line 1, in <module>

File "/usr/local/spark/python/pyspark/sql/session.py", line 723, in sql

return DataFrame(self.\_jsparkSession.sql(sqlQuery), self.\_wrapped)

File "/usr/local/spark/python/lib/py4j-0.10.9.3-src.zip/py4j/java\_gateway.py", line 1322, in \_\_call\_\_

File "/usr/local/spark/python/pyspark/sql/utils.py", line 117, in deco

raise converted from None

pyspark.sql.utils.ParseException:

mismatched input 'NetworkURL' expecting {<EOF>, ';'}(line 1, pos 19)

== SQL ==

select \*,case when NetworkURL like 'https://' then 'http secured' case when NetworkURL like 'http://' then 'http non secured' else 'No Protocol' as Protocol\_Used from insureview

-------------------^^^

>>> spark.sql("""select \*,case when NetworkURL like 'https://' then 'http secured' when NetworkURL like 'http://' then 'http non secured' else 'No Protocol' as Protocol\_Used from insureview""").show(3)

Traceback (most recent call last):

File "<stdin>", line 1, in <module>

File "/usr/local/spark/python/pyspark/sql/session.py", line 723, in sql

return DataFrame(self.\_jsparkSession.sql(sqlQuery), self.\_wrapped)

File "/usr/local/spark/python/lib/py4j-0.10.9.3-src.zip/py4j/java\_gateway.py", line 1322, in \_\_call\_\_

File "/usr/local/spark/python/pyspark/sql/utils.py", line 117, in deco

raise converted from None

pyspark.sql.utils.ParseException:

mismatched input 'NetworkURL' expecting {<EOF>, ';'}(line 1, pos 19)

== SQL ==

select \*,case when NetworkURL like 'https://' then 'http secured' when NetworkURL like 'http://' then 'http non secured' else 'No Protocol' as Protocol\_Used from insureview

-------------------^^^

>>> spark.sql("""select \*,case when NetworkURL like 'https://%' then 'http secured' when NetworkURL like 'http://%' then 'http non secured' else 'No Protocol' as Protocol\_Used from insureview""").show(3)

Traceback (most recent call last):

File "<stdin>", line 1, in <module>

File "/usr/local/spark/python/pyspark/sql/session.py", line 723, in sql

return DataFrame(self.\_jsparkSession.sql(sqlQuery), self.\_wrapped)

File "/usr/local/spark/python/lib/py4j-0.10.9.3-src.zip/py4j/java\_gateway.py", line 1322, in \_\_call\_\_

File "/usr/local/spark/python/pyspark/sql/utils.py", line 117, in deco

raise converted from None

pyspark.sql.utils.ParseException:

mismatched input 'NetworkURL' expecting {<EOF>, ';'}(line 1, pos 19)

== SQL ==

select \*,case when NetworkURL like 'https://%' then 'http secured' when NetworkURL like 'http://%' then 'http non secured' else 'No Protocol' as Protocol\_Used from insureview

-------------------^^^

>>> spark.sql("""select \*,case when NetworkURL like 'https://%' then 'http secured' when NetworkURL like 'http://%' then 'http non secured' else 'No Protocol' end as Protocol\_Used from insureview""").show(3)

+------------+----+-----+------------+--------------------+-------+--------------+--------------------+--------+---------+-----------------+----------+--------------------+-------------+

|BusinessDate|stcd|srcnm| NetworkName| NetworkURL|custnum|MarketCoverage| RejectRows|IssuerId|IssuerId2|issueridcomposite| sysdt| systs|Protocol\_Used|

+------------+----+-----+------------+--------------------+-------+--------------+--------------------+--------+---------+-----------------+----------+--------------------+-------------+

| 2019-10-01| AK| HIOS| ODS Premier|https://www.modah...| 13| null| null| 21989| 21989| 2198921989|2024-07-22|2024-07-22 11:55:...| http secured|

| 2019-10-01| AK| HIOS| ODS Premier| null| null| null|219,219,2019-10-0...| 219| 219| 219219|2024-07-22|2024-07-22 11:55:...| No Protocol|

| 2019-10-01| AK| HIOS|HeritagePlus|https://www.preme...| 13| null| null| 38344| 38344| 3834438344|2024-07-22|2024-07-22 11:55:...| http secured|

+------------+----+-----+------------+--------------------+-------+--------------+--------------------+--------+---------+-----------------+----------+--------------------+-------------+

only showing top 3 rows

>>> result\_df=spark.sql("""select \*,month(BuisnessDate) as month ,year(BuisnessDate) as year from insureview""")

Traceback (most recent call last):

File "<stdin>", line 1, in <module>

File "/usr/local/spark/python/pyspark/sql/session.py", line 723, in sql

return DataFrame(self.\_jsparkSession.sql(sqlQuery), self.\_wrapped)

File "/usr/local/spark/python/lib/py4j-0.10.9.3-src.zip/py4j/java\_gateway.py", line 1322, in \_\_call\_\_

File "/usr/local/spark/python/pyspark/sql/utils.py", line 117, in deco

raise converted from None

pyspark.sql.utils.AnalysisException: cannot resolve 'BuisnessDate' given input columns: [insureview.BusinessDate, insureview.IssuerId, insureview.IssuerId2, insureview.MarketCoverage, insureview.NetworkName, insureview.NetworkURL, insureview.RejectRows, insureview.custnum, insureview.issueridcomposite, insureview.srcnm, insureview.stcd, insureview.sysdt, insureview.systs]; line 1 pos 15;

'Project [BusinessDate#1378, stcd#7759, srcnm#7747, NetworkName#1381, NetworkURL#1382, custnum#1383, MarketCoverage#1384, RejectRows#1386, IssuerId#8312, IssuerId2#8325, issueridcomposite#8901, sysdt#9003, systs#9016, 'month('BuisnessDate) AS month#30398, 'year('BuisnessDate) AS year#30399]

+- SubqueryAlias insureview

+- View (`insureview`, [BusinessDate#1378,stcd#7759,srcnm#7747,NetworkName#1381,NetworkURL#1382,custnum#1383,MarketCoverage#1384,RejectRows#1386,IssuerId#8312,IssuerId2#8325,issueridcomposite#8901,sysdt#9003,systs#9016])

+- Project [BusinessDate#1378, stcd#7759, srcnm#7747, NetworkName#1381, NetworkURL#1382, custnum#1383, MarketCoverage#1384, RejectRows#1386, IssuerId#8312, IssuerId2#8325, issueridcomposite#8901, sysdt#9003, current\_timestamp() AS systs#9016]

+- Project [BusinessDate#1378, stcd#7759, srcnm#7747, NetworkName#1381, NetworkURL#1382, custnum#1383, MarketCoverage#1384, RejectRows#1386, IssuerId#8312, IssuerId2#8325, issueridcomposite#8901, current\_date(Some(Asia/Kolkata)) AS sysdt#9003]

+- Project [BusinessDate#1378, stcd#7759, srcnm#7747, NetworkName#1381, NetworkURL#1382, custnum#1383, MarketCoverage#1384, RejectRows#1386, IssuerId#8312, IssuerId2#8325, issueridcomposite#8901]

+- Project [BusinessDate#1378, stcd#7759, srcnm#7747, NetworkName#1381, NetworkURL#1382, custnum#1383, MarketCoverage#1384, DentalOnlyPlan#1385, RejectRows#1386, IssuerId#8312, IssuerId2#8325, concat(IssuerId#8312, IssuerId2#8325) AS issueridcomposite#8901]

+- Project [BusinessDate#1378, stcd#7759, srcnm#7747, NetworkName#1381, NetworkURL#1382, custnum#1383, MarketCoverage#1384, DentalOnlyPlan#1385, RejectRows#1386, IssuerId#8312, IssuerId2#8325]

+- Project [IssuerIDDD#8273, BusinessDate#1378, stcd#7759, srcnm#7747, NetworkName#1381, NetworkURL#1382, custnum#1383, MarketCoverage#1384, DentalOnlyPlan#1385, RejectRows#1386, IssuerId#8312, IssuerId2#8325]

+- Project [IssuerIDD#8261, IssuerIDDD#8273, BusinessDate#1378, stcd#7759, srcnm#7747, NetworkName#1381, NetworkURL#1382, custnum#1383, MarketCoverage#1384, DentalOnlyPlan#1385, RejectRows#1386, IssuerId#8312, cast(IssuerIDDD#8273 as string) AS IssuerId2#8325]

+- Project [IssuerIDD#8261, IssuerIDDD#8273, BusinessDate#1378, stcd#7759, srcnm#7747, NetworkName#1381, NetworkURL#1382, custnum#1383, MarketCoverage#1384, DentalOnlyPlan#1385, RejectRows#1386, cast(IssuerIDD#8261 as string) AS IssuerId#8312]

+- Project [IssuerIDD#8261, IssuerId2#1377 AS IssuerIDDD#8273, BusinessDate#1378, stcd#7759, srcnm#7747, NetworkName#1381, NetworkURL#1382, custnum#1383, MarketCoverage#1384, DentalOnlyPlan#1385, RejectRows#1386]

+- Project [IssuerId#1376 AS IssuerIDD#8261, IssuerId2#1377, BusinessDate#1378, stcd#7759, srcnm#7747, NetworkName#1381, NetworkURL#1382, custnum#1383, MarketCoverage#1384, DentalOnlyPlan#1385, RejectRows#1386]

+- Project [IssuerId#1376, IssuerId2#1377, BusinessDate#1378, StateCode#1379 AS stcd#7759, srcnm#7747, NetworkName#1381, NetworkURL#1382, custnum#1383, MarketCoverage#1384, DentalOnlyPlan#1385, RejectRows#1386]

+- Project [IssuerId#1376, IssuerId2#1377, BusinessDate#1378, StateCode#1379, SourceName#1380 AS srcnm#7747, NetworkName#1381, NetworkURL#1382, custnum#1383, MarketCoverage#1384, DentalOnlyPlan#1385, RejectRows#1386]

+- Union false, false

:- Relation [IssuerId#1376,IssuerId2#1377,BusinessDate#1378,StateCode#1379,SourceName#1380,NetworkName#1381,NetworkURL#1382,custnum#1383,MarketCoverage#1384,DentalOnlyPlan#1385,RejectRows#1386] csv

+- Relation [IssuerId#491,IssuerId2#492,BusinessDate#493,StateCode#494,SourceName#495,NetworkName#496,NetworkURL#497,custnum#498,MarketCoverage#499,DentalOnlyPlan#500,RejectRows#501] csv

>>> spark.sql("""select \*,month(BusinessDate) as month ,year(BusinessDate) as year from insureview""").show(3)

+------------+----+-----+------------+--------------------+-------+--------------+--------------------+--------+---------+-----------------+----------+--------------------+-----+----+

|BusinessDate|stcd|srcnm| NetworkName| NetworkURL|custnum|MarketCoverage| RejectRows|IssuerId|IssuerId2|issueridcomposite| sysdt| systs|month|year|

+------------+----+-----+------------+--------------------+-------+--------------+--------------------+--------+---------+-----------------+----------+--------------------+-----+----+

| 2019-10-01| AK| HIOS| ODS Premier|https://www.modah...| 13| null| null| 21989| 21989| 2198921989|2024-07-22|2024-07-22 11:57:...| 10|2019|

| 2019-10-01| AK| HIOS| ODS Premier| null| null| null|219,219,2019-10-0...| 219| 219| 219219|2024-07-22|2024-07-22 11:57:...| 10|2019|

| 2019-10-01| AK| HIOS|HeritagePlus|https://www.preme...| 13| null| null| 38344| 38344| 3834438344|2024-07-22|2024-07-22 11:57:...| 10|2019|

+------------+----+-----+------------+--------------------+-------+--------------+--------------------+--------+---------+-----------------+----------+--------------------+-----+----+

only showing top 3 rows

>>> result\_df\_SQL=spark.sql("""select \*,month(BusinessDate) as month ,year(BusinessDate) as year from insureview""")

>>> result\_df\_SQL\_1=spark.sql("""select \*,case when NetworkURL like 'https://%' then 'http secured' when NetworkURL like 'http://%' then 'http non secured' else 'No Protocol' as Protocol\_Used from insureview""")

Traceback (most recent call last):

File "<stdin>", line 1, in <module>

File "/usr/local/spark/python/pyspark/sql/session.py", line 723, in sql

return DataFrame(self.\_jsparkSession.sql(sqlQuery), self.\_wrapped)

File "/usr/local/spark/python/lib/py4j-0.10.9.3-src.zip/py4j/java\_gateway.py", line 1322, in \_\_call\_\_

File "/usr/local/spark/python/pyspark/sql/utils.py", line 117, in deco

raise converted from None

pyspark.sql.utils.ParseException:

mismatched input 'NetworkURL' expecting {<EOF>, ';'}(line 1, pos 19)

== SQL ==

select \*,case when NetworkURL like 'https://%' then 'http secured' when NetworkURL like 'http://%' then 'http non secured' else 'No Protocol' as Protocol\_Used from insureview

-------------------^^^

>>> result\_df\_SQL\_1=spark.sql("""select \*,case when NetworkURL like 'https://%' then 'http secured' when NetworkURL like 'http://%' then 'http non secured' else 'No Protocol' end as Protocol\_Used from insureview""")

>>> spark.sql("select \*,current\_date as curdt , current\_timestamp as curts from insureview").show(3)

+------------+----+-----+------------+--------------------+-------+--------------+--------------------+--------+---------+-----------------+----------+--------------------+----------+--------------------+

|BusinessDate|stcd|srcnm| NetworkName| NetworkURL|custnum|MarketCoverage| RejectRows|IssuerId|IssuerId2|issueridcomposite| sysdt| systs| curdt| curts|

+------------+----+-----+------------+--------------------+-------+--------------+--------------------+--------+---------+-----------------+----------+--------------------+----------+--------------------+

| 2019-10-01| AK| HIOS| ODS Premier|https://www.modah...| 13| null| null| 21989| 21989| 2198921989|2024-07-22|2024-07-22 12:00:...|2024-07-22|2024-07-22 12:00:...|

| 2019-10-01| AK| HIOS| ODS Premier| null| null| null|219,219,2019-10-0...| 219| 219| 219219|2024-07-22|2024-07-22 12:00:...|2024-07-22|2024-07-22 12:00:...|

| 2019-10-01| AK| HIOS|HeritagePlus|https://www.preme...| 13| null| null| 38344| 38344| 3834438344|2024-07-22|2024-07-22 12:00:...|2024-07-22|2024-07-22 12:00:...|

+------------+----+-----+------------+--------------------+-------+--------------+--------------------+--------+---------+-----------------+----------+--------------------+----------+--------------------+

only showing top 3 rows

>>> result\_df\_SQL\_2=spark.sql("select \*,current\_date as curdt , current\_timestamp as curts from insureview")

>>> spark.sql("select \*,sql\_function(NetworkName) as cleannetworkname from insureview").show(3)

+------------+----+-----+------------+--------------------+-------+--------------+--------------------+--------+---------+-----------------+----------+--------------------+----------------+

|BusinessDate|stcd|srcnm| NetworkName| NetworkURL|custnum|MarketCoverage| RejectRows|IssuerId|IssuerId2|issueridcomposite| sysdt| systs|cleannetworkname|

+------------+----+-----+------------+--------------------+-------+--------------+--------------------+--------+---------+-----------------+----------+--------------------+----------------+

| 2019-10-01| AK| HIOS| ODS Premier|https://www.modah...| 13| null| null| 21989| 21989| 2198921989|2024-07-22|2024-07-22 12:00:...| ODS Premier|

| 2019-10-01| AK| HIOS| ODS Premier| null| null| null|219,219,2019-10-0...| 219| 219| 219219|2024-07-22|2024-07-22 12:00:...| ODS Premier|

| 2019-10-01| AK| HIOS|HeritagePlus|https://www.preme...| 13| null| null| 38344| 38344| 3834438344|2024-07-22|2024-07-22 12:00:...| HeritagePlus|

+------------+----+-----+------------+--------------------+-------+--------------+--------------------+--------+---------+-----------------+----------+--------------------+----------------+

only showing top 3 rows

>>> result\_df\_SQL\_3=spark.sql("select \*,sql\_function(NetworkName) as cleannetworkname from insureview")

>>> result\_df\_SQL\_after\_join=result\_df\_SQL.alias("l").join(result\_df\_SQL\_1.alias("r") , on=[l.IssuerId=r.IssuerId],how="inner")

File "<stdin>", line 1

result\_df\_SQL\_after\_join=result\_df\_SQL.alias("l").join(result\_df\_SQL\_1.alias("r") , on=[l.IssuerId=r.IssuerId],how="inner")

^

SyntaxError: invalid syntax

>>> result\_df\_SQL\_after\_join=result\_df\_SQL.alias("l").join(result\_df\_SQL\_1.alias("r") , on=[l.IssuerId=r.IssuerId],how="inner")

File "<stdin>", line 1

result\_df\_SQL\_after\_join=result\_df\_SQL.alias("l").join(result\_df\_SQL\_1.alias("r") , on=[l.IssuerId=r.IssuerId],how="inner")

^

SyntaxError: invalid syntax

>>> result\_df\_SQL\_after\_join = result\_df\_SQL.alias("l").join(result\_df\_SQL\_1.alias("r"), on=col("l.IssuerId") == col("r.IssuerId"), how="inner")

>>>

>>> result\_df\_SQL\_after\_join = result\_df\_SQL\_after\_join.alias("l").join(result\_df\_SQL\_2.alias("r"), on=col("l.IssuerId") == col("r.IssuerId"), how="inner")

Traceback (most recent call last):

File "<stdin>", line 1, in <module>

File "/usr/local/spark/python/pyspark/sql/dataframe.py", line 1355, in join

jdf = self.\_jdf.join(other.\_jdf, on, how)

File "/usr/local/spark/python/lib/py4j-0.10.9.3-src.zip/py4j/java\_gateway.py", line 1322, in \_\_call\_\_

File "/usr/local/spark/python/pyspark/sql/utils.py", line 117, in deco

raise converted from None

pyspark.sql.utils.AnalysisException: Reference 'l.IssuerId' is ambiguous, could be: l.IssuerId, l.IssuerId.

>>> result\_df\_SQL\_after\_join = result\_df\_SQL\_after\_join.alias("l").join(result\_df\_SQL\_2.alias("r"), on=[col("l.IssuerId") == col("r.IssuerId")], how="inner")

Traceback (most recent call last):

File "<stdin>", line 1, in <module>

File "/usr/local/spark/python/pyspark/sql/dataframe.py", line 1355, in join

jdf = self.\_jdf.join(other.\_jdf, on, how)

File "/usr/local/spark/python/lib/py4j-0.10.9.3-src.zip/py4j/java\_gateway.py", line 1322, in \_\_call\_\_

File "/usr/local/spark/python/pyspark/sql/utils.py", line 117, in deco

raise converted from None

pyspark.sql.utils.AnalysisException: Reference 'l.IssuerId' is ambiguous, could be: l.IssuerId, l.IssuerId.

>>> result\_df\_SQL\_after\_join=result\_df\_SQL.alias("l").join(result\_df\_SQL\_1.alias("r") , on=[l.IssuerId=r.IssuerId],how="inner")

>>> result\_df\_SQL\_after\_join=result\_df\_SQL.alias("l").join(result\_df\_SQL\_1.alias("r") , on=[l.IssuerId=r.IssuerId],how="inner")

File "<stdin>", line 1

result\_df\_SQL\_after\_join=result\_df\_SQL.alias("l").join(result\_df\_SQL\_1.alias("r") , on=[l.IssuerId=r.IssuerId],how="inner")

^

SyntaxError: invalid syntax

>>> result\_df\_SQL\_after\_join = result\_df\_SQL\_after\_join.alias("l").join(result\_df\_SQL\_2.alias("r"), on=[col("l.IssuerId") == col("r.IssuerId")], how="inner")

Traceback (most recent call last):

File "<stdin>", line 1, in <module>

File "/usr/local/spark/python/pyspark/sql/dataframe.py", line 1355, in join

jdf = self.\_jdf.join(other.\_jdf, on, how)

File "/usr/local/spark/python/lib/py4j-0.10.9.3-src.zip/py4j/java\_gateway.py", line 1322, in \_\_call\_\_

File "/usr/local/spark/python/pyspark/sql/utils.py", line 117, in deco

raise converted from None

pyspark.sql.utils.AnalysisException: Reference 'l.IssuerId' is ambiguous, could be: l.IssuerId, l.IssuerId.

>>> result\_df\_SQL\_after\_join = result\_df\_SQL.alias("l").join(result\_df\_SQL\_1.alias("r"), on=col("l.IssuerId") == col("r.IssuerId"), how="inner")

>>>

>>> result\_df\_SQL\_after\_join = result\_df\_SQL.alias("l").join(result\_df\_SQL\_1.alias("r"), on=col("l.IssuerId") == col("r.IssuerId"), how="inner")

>>> result\_df\_SQL\_after\_join = result\_df\_SQL\_after\_join.join(result\_df\_SQL\_2, on=[col("IssuerId") == col("IssuerId")], how="inner")

24/07/22 12:08:43 WARN sql.Column: Constructing trivially true equals predicate, ''IssuerId = 'IssuerId'. Perhaps you need to use aliases.

Traceback (most recent call last):

File "<stdin>", line 1, in <module>

File "/usr/local/spark/python/pyspark/sql/dataframe.py", line 1355, in join

jdf = self.\_jdf.join(other.\_jdf, on, how)

File "/usr/local/spark/python/lib/py4j-0.10.9.3-src.zip/py4j/java\_gateway.py", line 1322, in \_\_call\_\_

File "/usr/local/spark/python/pyspark/sql/utils.py", line 117, in deco

raise converted from None

pyspark.sql.utils.AnalysisException: Reference 'IssuerId' is ambiguous, could be: l.IssuerId, r.IssuerId, insureview.IssuerId.

>>> result\_df\_SQL\_after\_join = result\_df\_SQL\_after\_join.alias("l").join(result\_df\_SQL\_2.alias("r"), on=[col("l.IssuerId") == col("r.IssuerId")], how="inner")

Traceback (most recent call last):

File "<stdin>", line 1, in <module>

File "/usr/local/spark/python/pyspark/sql/dataframe.py", line 1355, in join

jdf = self.\_jdf.join(other.\_jdf, on, how)

File "/usr/local/spark/python/lib/py4j-0.10.9.3-src.zip/py4j/java\_gateway.py", line 1322, in \_\_call\_\_

File "/usr/local/spark/python/pyspark/sql/utils.py", line 117, in deco

raise converted from None

pyspark.sql.utils.AnalysisException: Reference 'l.IssuerId' is ambiguous, could be: l.IssuerId, l.IssuerId.

>>> result\_df\_SQL\_after\_join = result\_df\_SQL\_after\_join.join(result\_df\_SQL\_2, on=[col("IssuerId") == col("IssuerId")], how="inner")

24/07/22 12:09:44 WARN sql.Column: Constructing trivially true equals predicate, ''IssuerId = 'IssuerId'. Perhaps you need to use aliases.

Traceback (most recent call last):

File "<stdin>", line 1, in <module>

File "/usr/local/spark/python/pyspark/sql/dataframe.py", line 1355, in join

jdf = self.\_jdf.join(other.\_jdf, on, how)

File "/usr/local/spark/python/lib/py4j-0.10.9.3-src.zip/py4j/java\_gateway.py", line 1322, in \_\_call\_\_

File "/usr/local/spark/python/pyspark/sql/utils.py", line 117, in deco

raise converted from None

pyspark.sql.utils.AnalysisException: Reference 'IssuerId' is ambiguous, could be: l.IssuerId, r.IssuerId, insureview.IssuerId.

>>>

>>> result\_df\_SQL\_after\_join = result\_df\_SQL.alias("l").join(result\_df\_SQL\_1.alias("r"), on=col("l.IssuerId") == col("r.IssuerId"), how="inner")

>>> result\_df\_SQL\_after\_join = result\_df\_SQL\_after\_join.alias("l").join(

... result\_df\_SQL\_2.alias("r"),

... col("l.IssuerId") == col("r.IssuerId"),

... how="inner"

... )

Traceback (most recent call last):

File "<stdin>", line 4, in <module>

File "/usr/local/spark/python/pyspark/sql/dataframe.py", line 1355, in join

jdf = self.\_jdf.join(other.\_jdf, on, how)

File "/usr/local/spark/python/lib/py4j-0.10.9.3-src.zip/py4j/java\_gateway.py", line 1322, in \_\_call\_\_

File "/usr/local/spark/python/pyspark/sql/utils.py", line 117, in deco

raise converted from None

pyspark.sql.utils.AnalysisException: Reference 'l.IssuerId' is ambiguous, could be: l.IssuerId, l.IssuerId.

>>> result\_df\_SQL.show(3)

+------------+----+-----+------------+--------------------+-------+--------------+--------------------+--------+---------+-----------------+----------+--------------------+-----+----+

|BusinessDate|stcd|srcnm| NetworkName| NetworkURL|custnum|MarketCoverage| RejectRows|IssuerId|IssuerId2|issueridcomposite| sysdt| systs|month|year|

+------------+----+-----+------------+--------------------+-------+--------------+--------------------+--------+---------+-----------------+----------+--------------------+-----+----+

| 2019-10-01| AK| HIOS| ODS Premier|https://www.modah...| 13| null| null| 21989| 21989| 2198921989|2024-07-22|2024-07-22 12:16:...| 10|2019|

| 2019-10-01| AK| HIOS| ODS Premier| null| null| null|219,219,2019-10-0...| 219| 219| 219219|2024-07-22|2024-07-22 12:16:...| 10|2019|

| 2019-10-01| AK| HIOS|HeritagePlus|https://www.preme...| 13| null| null| 38344| 38344| 3834438344|2024-07-22|2024-07-22 12:16:...| 10|2019|

+------------+----+-----+------------+--------------------+-------+--------------+--------------------+--------+---------+-----------------+----------+--------------------+-----+----+

only showing top 3 rows

>>> result\_df\_SQL.show(2)

>>> result\_df\_SQL.show(2)

+------------+----+-----+-----------+--------------------+-------+--------------+--------------------+--------+---------+-----------------+----------+--------------------+-----+----+

|BusinessDate|stcd|srcnm|NetworkName| NetworkURL|custnum|MarketCoverage| RejectRows|IssuerId|IssuerId2|issueridcomposite| sysdt| systs|month|year|

+------------+----+-----+-----------+--------------------+-------+--------------+--------------------+--------+---------+-----------------+----------+--------------------+-----+----+

| 2019-10-01| AK| HIOS|ODS Premier|https://www.modah...| 13| null| null| 21989| 21989| 2198921989|2024-07-22|2024-07-22 12:16:...| 10|2019|

| 2019-10-01| AK| HIOS|ODS Premier| null| null| null|219,219,2019-10-0...| 219| 219| 219219|2024-07-22|2024-07-22 12:16:...| 10|2019|

+------------+----+-----+-----------+--------------------+-------+--------------+--------------------+--------+---------+-----------------+----------+--------------------+-----+----+

only showing top 2 rows

>>> result\_df\_SQL\_1.show(2)

+------------+----+-----+-----------+--------------------+-------+--------------+--------------------+--------+---------+-----------------+----------+--------------------+-------------+

|BusinessDate|stcd|srcnm|NetworkName| NetworkURL|custnum|MarketCoverage| RejectRows|IssuerId|IssuerId2|issueridcomposite| sysdt| systs|Protocol\_Used|

+------------+----+-----+-----------+--------------------+-------+--------------+--------------------+--------+---------+-----------------+----------+--------------------+-------------+

| 2019-10-01| AK| HIOS|ODS Premier|https://www.modah...| 13| null| null| 21989| 21989| 2198921989|2024-07-22|2024-07-22 12:16:...| http secured|

| 2019-10-01| AK| HIOS|ODS Premier| null| null| null|219,219,2019-10-0...| 219| 219| 219219|2024-07-22|2024-07-22 12:16:...| No Protocol|

+------------+----+-----+-----------+--------------------+-------+--------------+--------------------+--------+---------+-----------------+----------+--------------------+-------------+

only showing top 2 rows

>>> result\_df\_SQL\_2.show(2)

+------------+----+-----+-----------+--------------------+-------+--------------+--------------------+--------+---------+-----------------+----------+--------------------+----------+--------------------+

|BusinessDate|stcd|srcnm|NetworkName| NetworkURL|custnum|MarketCoverage| RejectRows|IssuerId|IssuerId2|issueridcomposite| sysdt| systs| curdt| curts|

+------------+----+-----+-----------+--------------------+-------+--------------+--------------------+--------+---------+-----------------+----------+--------------------+----------+--------------------+

| 2019-10-01| AK| HIOS|ODS Premier|https://www.modah...| 13| null| null| 21989| 21989| 2198921989|2024-07-22|2024-07-22 12:16:...|2024-07-22|2024-07-22 12:16:...|

| 2019-10-01| AK| HIOS|ODS Premier| null| null| null|219,219,2019-10-0...| 219| 219| 219219|2024-07-22|2024-07-22 12:16:...|2024-07-22|2024-07-22 12:16:...|

+------------+----+-----+-----------+--------------------+-------+--------------+--------------------+--------+---------+-----------------+----------+--------------------+----------+--------------------+

only showing top 2 rows

>>> result\_df\_SQL\_3.show(2)

+------------+----+-----+-----------+--------------------+-------+--------------+--------------------+--------+---------+-----------------+----------+--------------------+----------------+

|BusinessDate|stcd|srcnm|NetworkName| NetworkURL|custnum|MarketCoverage| RejectRows|IssuerId|IssuerId2|issueridcomposite| sysdt| systs|cleannetworkname|

+------------+----+-----+-----------+--------------------+-------+--------------+--------------------+--------+---------+-----------------+----------+--------------------+----------------+

| 2019-10-01| AK| HIOS|ODS Premier|https://www.modah...| 13| null| null| 21989| 21989| 2198921989|2024-07-22|2024-07-22 12:16:...| ODS Premier|

| 2019-10-01| AK| HIOS|ODS Premier| null| null| null|219,219,2019-10-0...| 219| 219| 219219|2024-07-22|2024-07-22 12:16:...| ODS Premier|

+------------+----+-----+-----------+--------------------+-------+--------------+--------------------+--------+---------+-----------------+----------+--------------------+----------------+

only showing top 2 rows

>>> result\_df\_SQL.show(1)

>>> result\_df\_SQL.show(1)

+------------+----+-----+-----------+--------------------+-------+--------------+----------+--------+---------+-----------------+----------+--------------------+-----+----+

|BusinessDate|stcd|srcnm|NetworkName| NetworkURL|custnum|MarketCoverage|RejectRows|IssuerId|IssuerId2|issueridcomposite| sysdt| systs|month|year|

+------------+----+-----+-----------+--------------------+-------+--------------+----------+--------+---------+-----------------+----------+--------------------+-----+----+

| 2019-10-01| AK| HIOS|ODS Premier|https://www.modah...| 13| null| null| 21989| 21989| 2198921989|2024-07-22|2024-07-22 12:17:...| 10|2019|

+------------+----+-----+-----------+--------------------+-------+--------------+----------+--------+---------+-----------------+----------+--------------------+-----+----+

only showing top 1 row

>>> result\_df\_SQL\_1.show(1)

+------------+----+-----+-----------+--------------------+-------+--------------+----------+--------+---------+-----------------+----------+--------------------+-------------+

|BusinessDate|stcd|srcnm|NetworkName| NetworkURL|custnum|MarketCoverage|RejectRows|IssuerId|IssuerId2|issueridcomposite| sysdt| systs|Protocol\_Used|

+------------+----+-----+-----------+--------------------+-------+--------------+----------+--------+---------+-----------------+----------+--------------------+-------------+

| 2019-10-01| AK| HIOS|ODS Premier|https://www.modah...| 13| null| null| 21989| 21989| 2198921989|2024-07-22|2024-07-22 12:17:...| http secured|

+------------+----+-----+-----------+--------------------+-------+--------------+----------+--------+---------+-----------------+----------+--------------------+-------------+

only showing top 1 row

>>> result\_df\_SQL\_2.show(1)

+------------+----+-----+-----------+--------------------+-------+--------------+----------+--------+---------+-----------------+----------+--------------------+----------+--------------------+

|BusinessDate|stcd|srcnm|NetworkName| NetworkURL|custnum|MarketCoverage|RejectRows|IssuerId|IssuerId2|issueridcomposite| sysdt| systs| curdt| curts|

+------------+----+-----+-----------+--------------------+-------+--------------+----------+--------+---------+-----------------+----------+--------------------+----------+--------------------+

| 2019-10-01| AK| HIOS|ODS Premier|https://www.modah...| 13| null| null| 21989| 21989| 2198921989|2024-07-22|2024-07-22 12:17:...|2024-07-22|2024-07-22 12:17:...|

+------------+----+-----+-----------+--------------------+-------+--------------+----------+--------+---------+-----------------+----------+--------------------+----------+--------------------+

only showing top 1 row

>>> result\_df\_SQL\_3.show(1)

+------------+----+-----+-----------+--------------------+-------+--------------+----------+--------+---------+-----------------+----------+--------------------+----------------+

|BusinessDate|stcd|srcnm|NetworkName| NetworkURL|custnum|MarketCoverage|RejectRows|IssuerId|IssuerId2|issueridcomposite| sysdt| systs|cleannetworkname|

+------------+----+-----+-----------+--------------------+-------+--------------+----------+--------+---------+-----------------+----------+--------------------+----------------+

| 2019-10-01| AK| HIOS|ODS Premier|https://www.modah...| 13| null| null| 21989| 21989| 2198921989|2024-07-22|2024-07-22 12:20:...| ODS Premier|

+------------+----+-----+-----------+--------------------+-------+--------------+----------+--------+---------+-----------------+----------+--------------------+----------------+

only showing top 1 row

>>>

>>> df\_combined = result\_df\_SQL \

... .alias("a") \

... .join(result\_df\_SQL\_1.alias("b"), on=col("a.IssuerId") == col("b.IssuerId"), how="inner") \

... .join(result\_df\_SQL\_2.alias("c"), on=col("a.IssuerId") == col("c.IssuerId"), how="inner") \

... .join(result\_df\_SQL\_3.alias("d"), on=col("a.IssuerId") == col("d.IssuerId"), how="inner")

>>> df\_combined.show(1)

24/07/22 12:20:34 WARN util.package: Truncated the string representation of a plan since it was too large. This behavior can be adjusted by setting 'spark.sql.debug.maxToStringFields'.

+------------+----+-----+----------------+--------------------+-------+------------------+----------+--------+---------+-----------------+----------+--------------------+-----+----+------------+----+-----+-----------+--------------------+-------+--------------+----------+--------+---------+-----------------+----------+--------------------+-------------+------------+----+-----+----------------+--------------------+-------+------------------+----------+--------+---------+-----------------+----------+--------------------+----------+--------------------+------------+----+-----+----------------+--------------------+-------+------------------+----------+--------+---------+-----------------+----------+--------------------+----------------+

|BusinessDate|stcd|srcnm| NetworkName| NetworkURL|custnum| MarketCoverage|RejectRows|IssuerId|IssuerId2|issueridcomposite| sysdt| systs|month|year|BusinessDate|stcd|srcnm|NetworkName| NetworkURL|custnum|MarketCoverage|RejectRows|IssuerId|IssuerId2|issueridcomposite| sysdt| systs|Protocol\_Used|BusinessDate|stcd|srcnm| NetworkName| NetworkURL|custnum| MarketCoverage|RejectRows|IssuerId|IssuerId2|issueridcomposite| sysdt| systs| curdt| curts|BusinessDate|stcd|srcnm| NetworkName| NetworkURL|custnum| MarketCoverage|RejectRows|IssuerId|IssuerId2|issueridcomposite| sysdt| systs|cleannetworkname|

+------------+----+-----+----------------+--------------------+-------+------------------+----------+--------+---------+-----------------+----------+--------------------+-----+----+------------+----+-----+-----------+--------------------+-------+--------------+----------+--------+---------+-----------------+----------+--------------------+-------------+------------+----+-----+----------------+--------------------+-------+------------------+----------+--------+---------+-----------------+----------+--------------------+----------+--------------------+------------+----+-----+----------------+--------------------+-------+------------------+----------+--------+---------+-----------------+----------+--------------------+----------------+

| 2019-01-10| AK| HIOS|Delta Dental PPO|https://www.modah...| 17|SHOP (Small Group)| null| 21989| 21989| 2198921989|2024-07-22|2024-07-22 12:20:...| 1|2019| 2019-10-01| AK| HIOS|ODS Premier|https://www.modah...| 13| null| null| 21989| 21989| 2198921989|2024-07-22|2024-07-22 12:20:...| http secured| 2019-01-10| AK| HIOS|Delta Dental PPO|https://www.modah...| 17|SHOP (Small Group)| null| 21989| 21989| 2198921989|2024-07-22|2024-07-22 12:20:...|2024-07-22|2024-07-22 12:20:...| 2019-01-10| AK| HIOS|Delta Dental PPO|https://www.modah...| 17|SHOP (Small Group)| null| 21989| 21989| 2198921989|2024-07-22|2024-07-22 12:20:...|Delta Dental PPO|

+------------+----+-----+----------------+--------------------+-------+------------------+----------+--------+---------+-----------------+----------+--------------------+-----+----+------------+----+-----+-----------+--------------------+-------+--------------+----------+--------+---------+-----------------+----------+--------------------+-------------+------------+----+-----+----------------+--------------------+-------+------------------+----------+--------+---------+-----------------+----------+--------------------+----------+--------------------+------------+----+-----+----------------+--------------------+-------+------------------+----------+--------+---------+-----------------+----------+--------------------+----------------+

only showing top 1 row

>>> df\_with\_states = df\_combined \

... .alias("i") \

... .join(statesview.alias("s"), col("i.stcd") == col("s.stated"), how="inner")

Traceback (most recent call last):

File "<stdin>", line 3, in <module>

NameError: name 'statesview' is not defined

>>>

>>> # Joining result with custview

... df\_final = df\_with\_states \

... .alias("is") \

... .join(custview.alias("c"), col("is.custnum") == col("c.custid"), how="inner")

Traceback (most recent call last):

File "<stdin>", line 2, in <module>

NameError: name 'df\_with\_states' is not defined

>>> joined\_df\_state = df\_combined.alias("i").join(

... state\_filter\_df.alias("s"),

... col("i.stcd") == col("s.stated"),

... how="inner"

... )

Traceback (most recent call last):

File "<stdin>", line 2, in <module>

NameError: name 'state\_filter\_df' is not defined

>>>

>>> # Join the result with custfilter\_df

... final\_df = joined\_df\_state.alias("j").join(

... custfilter\_df.alias("c"),

... col("j.custnum") == col("c.custid"),

... how="inner"

... )

Traceback (most recent call last):

File "<stdin>", line 2, in <module>

NameError: name 'joined\_df\_state' is not defined

>>> )joined\_df\_state = df\_combined.alias("i").join(

File "<stdin>", line 1

)joined\_df\_state = df\_combined.alias("i").join(

^

SyntaxError: invalid syntax

>>> statefilter\_df.alias("s"),

File "<stdin>", line 1

statefilter\_df.alias("s"),

^

IndentationError: unexpected indent

>>> col("i.stcd") == col("s.stated"),

File "<stdin>", line 1

col("i.stcd") == col("s.stated"),

^

IndentationError: unexpected indent

>>> how="inner"

File "<stdin>", line 1

how="inner"

^

IndentationError: unexpected indent

>>> )

File "<stdin>", line 1

)

^

SyntaxError: invalid syntax

>>> joined\_df\_state = df\_combined.alias("i").join(

... statefilter\_df.alias("s"),

... col("i.stcd") == col("s.stated"),

... how="inner"

... )

Traceback (most recent call last):

File "<stdin>", line 4, in <module>

File "/usr/local/spark/python/pyspark/sql/dataframe.py", line 1355, in join

jdf = self.\_jdf.join(other.\_jdf, on, how)

File "/usr/local/spark/python/lib/py4j-0.10.9.3-src.zip/py4j/java\_gateway.py", line 1322, in \_\_call\_\_

File "/usr/local/spark/python/pyspark/sql/utils.py", line 117, in deco

raise converted from None

pyspark.sql.utils.AnalysisException: Reference 'i.stcd' is ambiguous, could be: i.stcd, i.stcd, i.stcd, i.stcd.

>>> joined\_df\_state = df\_combined.alias("i").join(

... state\_filter\_df.alias("s"),

... col("i.stcd") == col("s.stated"),

... how="inner"

... )

Traceback (most recent call last):

File "<stdin>", line 2, in <module>

NameError: name 'state\_filter\_df' is not defined

>>> joined\_df\_state = df\_combined.alias("i").join(

... statefilter\_df.alias("s"),

... col("i.stcd") == col("s.stated"),

... how="inner"

... )

Traceback (most recent call last):

File "<stdin>", line 4, in <module>

File "/usr/local/spark/python/pyspark/sql/dataframe.py", line 1355, in join

jdf = self.\_jdf.join(other.\_jdf, on, how)

File "/usr/local/spark/python/lib/py4j-0.10.9.3-src.zip/py4j/java\_gateway.py", line 1322, in \_\_call\_\_

File "/usr/local/spark/python/pyspark/sql/utils.py", line 117, in deco

raise converted from None

pyspark.sql.utils.AnalysisException: Reference 'i.stcd' is ambiguous, could be: i.stcd, i.stcd, i.stcd, i.stcd.

>>>

>>> df\_combined\_renamed = df\_combined.withColumnRenamed("stcd", "i\_stcd")

>>> state\_filter\_df\_renamed = statefilter\_df.withColumnRenamed("stated", "s\_stated")

>>>

>>> # Step 2: Perform the join with renamed columns

... joined\_df\_state = df\_combined\_renamed.alias("i").join(

... statefilter\_df\_renamed.alias("s"),

... col("i.i\_stcd") == col("s.s\_stated"),

... how="inner"

... )

Traceback (most recent call last):

File "<stdin>", line 3, in <module>

NameError: name 'statefilter\_df\_renamed' is not defined

>>>

>>> # Step 3: Rename columns in the result to remove ambiguity

... df\_combined\_state\_renamed = joined\_df\_state.withColumnRenamed("statedesc", "state\_desc")

Traceback (most recent call last):

File "<stdin>", line 2, in <module>

NameError: name 'joined\_df\_state' is not defined

>>>

>>> # Step 4: Join with custfilter\_df

... final\_df = df\_combined\_state\_renamed.alias("j").join(

... custfilter\_df.alias("c"),

... col("j.custnum") == col("c.custid"),

... how="inner"

... )

Traceback (most recent call last):

File "<stdin>", line 2, in <module>

NameError: name 'df\_combined\_state\_renamed' is not defined

>>> state\_filter\_df\_renamed.SHOW()

Traceback (most recent call last):

File "<stdin>", line 1, in <module>

File "/usr/local/spark/python/pyspark/sql/dataframe.py", line 1660, in \_\_getattr\_\_

"'%s' object has no attribute '%s'" % (self.\_\_class\_\_.\_\_name\_\_, name))

AttributeError: 'DataFrame' object has no attribute 'SHOW'

>>> state\_filter\_df\_renamed.show()

+---+--------------+

|\_c0| \_c1|

+---+--------------+

| NM| New Mexico|

| NY| New York|

| NC|North Carolina|

| ND| North Dakota|

| OH| Ohio|

| OK| Oklahoma|

| OR| Oregon|

| MD| Maryland|

| MA| Massachusetts|

| MI| Michigan|

| MN| Minnesota|

| MS| Mississippi|

| MO| Missouri|

| PA| Pennsylvania|

| RI| Rhode Island|

| SC|South Carolina|

| SD| South Dakota|

| TN| Tennessee|

| TX| Texas|

| UT| Utah|

+---+--------------+

only showing top 20 rows

>>> joined\_df\_state = df\_combined\_renamed.alias("i").join(

... statefilter\_df\_renamed.alias("s"),

... col("i.i\_stcd") == col("s.s\_stated"),

... how="inner"

... )

Traceback (most recent call last):

File "<stdin>", line 2, in <module>

NameError: name 'statefilter\_df\_renamed' is not defined

>>>

>>> col("i.i\_stcd") == col("s.s\_stated"),

>>> col("i.i\_stcd") == col("s.s\_stated"),

>>> col("i.i\_stcd") == col("s.s\_stated"),

File "<stdin>", line 1

col("i.i\_stcd") == col("s.s\_stated"),

^

IndentationError: unexpected indent

>>>

>>> df\_combined\_renamed = df\_combined.withColumnRenamed("stcd", "i\_stcd")

>>> state\_filter\_df\_renamed = statefilter\_df.withColumnRenamed("stated", "s\_stated")

>>>

>>> # Step 2: Perform the join with renamed columns

... joined\_df\_state = df\_combined\_renamed.alias("i").join(

... state\_filter\_df\_renamed.alias("s"),

... col("i.i\_stcd") == col("s.s\_stated"),

... how="inner"

... )

Traceback (most recent call last):

File "<stdin>", line 5, in <module>

File "/usr/local/spark/python/pyspark/sql/dataframe.py", line 1355, in join

jdf = self.\_jdf.join(other.\_jdf, on, how)

File "/usr/local/spark/python/lib/py4j-0.10.9.3-src.zip/py4j/java\_gateway.py", line 1322, in \_\_call\_\_

File "/usr/local/spark/python/pyspark/sql/utils.py", line 117, in deco

raise converted from None

pyspark.sql.utils.AnalysisException: Reference 'i.i\_stcd' is ambiguous, could be: i.i\_stcd, i.i\_stcd, i.i\_stcd, i.i\_stcd.

>>>

>>> # Step 3: Rename columns in the result to remove ambiguity

... df\_combined\_state\_renamed = joined\_df\_state.withColumnRenamed("statedesc", "state\_desc")

Traceback (most recent call last):

File "<stdin>", line 2, in <module>

NameError: name 'joined\_df\_state' is not defined

>>>

>>> # Step 4: Join with custfilter\_df

... final\_df = df\_combined\_state\_renamed.alias("j").join(

... custfilter\_df.alias("c"),

... col("j.custnum") == col("c.custid"),

... how="inner"

... )

Traceback (most recent call last):

File "<stdin>", line 2, in <module>

NameError: name 'df\_combined\_state\_renamed' is not defined

>>> df\_combined.show(1)

+------------+----+-----+----------------+--------------------+-------+------------------+----------+--------+---------+-----------------+----------+--------------------+-----+----+------------+----+-----+-----------+--------------------+-------+--------------+----------+--------+---------+-----------------+----------+--------------------+-------------+------------+----+-----+----------------+--------------------+-------+------------------+----------+--------+---------+-----------------+----------+--------------------+----------+--------------------+------------+----+-----+----------------+--------------------+-------+------------------+----------+--------+---------+-----------------+----------+--------------------+----------------+

|BusinessDate|stcd|srcnm| NetworkName| NetworkURL|custnum| MarketCoverage|RejectRows|IssuerId|IssuerId2|issueridcomposite| sysdt| systs|month|year|BusinessDate|stcd|srcnm|NetworkName| NetworkURL|custnum|MarketCoverage|RejectRows|IssuerId|IssuerId2|issueridcomposite| sysdt| systs|Protocol\_Used|BusinessDate|stcd|srcnm| NetworkName| NetworkURL|custnum| MarketCoverage|RejectRows|IssuerId|IssuerId2|issueridcomposite| sysdt| systs| curdt| curts|BusinessDate|stcd|srcnm| NetworkName| NetworkURL|custnum| MarketCoverage|RejectRows|IssuerId|IssuerId2|issueridcomposite| sysdt| systs|cleannetworkname|

+------------+----+-----+----------------+--------------------+-------+------------------+----------+--------+---------+-----------------+----------+--------------------+-----+----+------------+----+-----+-----------+--------------------+-------+--------------+----------+--------+---------+-----------------+----------+--------------------+-------------+------------+----+-----+----------------+--------------------+-------+------------------+----------+--------+---------+-----------------+----------+--------------------+----------+--------------------+------------+----+-----+----------------+--------------------+-------+------------------+----------+--------+---------+-----------------+----------+--------------------+----------------+

| 2019-01-10| AK| HIOS|Delta Dental PPO|https://www.modah...| 17|SHOP (Small Group)| null| 21989| 21989| 2198921989|2024-07-22|2024-07-22 12:31:...| 1|2019| 2019-10-01| AK| HIOS|ODS Premier|https://www.modah...| 13| null| null| 21989| 21989| 2198921989|2024-07-22|2024-07-22 12:31:...| http secured| 2019-01-10| AK| HIOS|Delta Dental PPO|https://www.modah...| 17|SHOP (Small Group)| null| 21989| 21989| 2198921989|2024-07-22|2024-07-22 12:31:...|2024-07-22|2024-07-22 12:31:...| 2019-01-10| AK| HIOS|Delta Dental PPO|https://www.modah...| 17|SHOP (Small Group)| null| 21989| 21989| 2198921989|2024-07-22|2024-07-22 12:31:...|Delta Dental PPO|

+------------+----+-----+----------------+--------------------+-------+------------------+----------+--------+---------+-----------------+----------+--------------------+-----+----+------------+----+-----+-----------+--------------------+-------+--------------+----------+--------+---------+-----------------+----------+--------------------+-------------+------------+----+-----+----------------+--------------------+-------+------------------+----------+--------+---------+-----------------+----------+--------------------+----------+--------------------+------------+----+-----+----------------+--------------------+-------+------------------+----------+--------+---------+-----------------+----------+--------------------+----------------+

only showing top 1 row

>>>

>>> df\_combined.write.parquet("hdfs:///user/hduser/Hackathon/Parquet")

Traceback (most recent call last):

File "<stdin>", line 1, in <module>

File "/usr/local/spark/python/pyspark/sql/readwriter.py", line 885, in parquet

self.\_jwrite.parquet(path)

File "/usr/local/spark/python/lib/py4j-0.10.9.3-src.zip/py4j/java\_gateway.py", line 1322, in \_\_call\_\_

File "/usr/local/spark/python/pyspark/sql/utils.py", line 117, in deco

raise converted from None

pyspark.sql.utils.AnalysisException: Found duplicate column(s) when inserting into hdfs://localhost:54310/user/hduser/Hackathon/Parquet: `businessdate`, `custnum`, `issuerid2`, `issuerid`, `issueridcomposite`, `marketcoverage`, `networkname`, `networkurl`, `rejectrows`, `srcnm`, `stcd`, `sysdt`, `systs`

>>> df\_combined.describe

<bound method DataFrame.describe of DataFrame[BusinessDate: date, stcd: string, srcnm: string, NetworkName: string, NetworkURL: string, custnum: string, MarketCoverage: string, RejectRows: string, IssuerId: string, IssuerId2: string, issueridcomposite: string, sysdt: date, systs: timestamp, month: int, year: int, BusinessDate: date, stcd: string, srcnm: string, NetworkName: string, NetworkURL: string, custnum: string, MarketCoverage: string, RejectRows: string, IssuerId: string, IssuerId2: string, issueridcomposite: string, sysdt: date, systs: timestamp, Protocol\_Used: string, BusinessDate: date, stcd: string, srcnm: string, NetworkName: string, NetworkURL: string, custnum: string, MarketCoverage: string, RejectRows: string, IssuerId: string, IssuerId2: string, issueridcomposite: string, sysdt: date, systs: timestamp, curdt: date, curts: timestamp, BusinessDate: date, stcd: string, srcnm: string, NetworkName: string, NetworkURL: string, custnum: string, MarketCoverage: string, RejectRows: string, IssuerId: string, IssuerId2: string, issueridcomposite: string, sysdt: date, systs: timestamp, cleannetworkname: string]>

>>> df\_combined.describe

>>> list\_of\_col=df\_combined.columns

>>> print(list\_of\_col)

['BusinessDate', 'stcd', 'srcnm', 'NetworkName', 'NetworkURL', 'custnum', 'MarketCoverage', 'RejectRows', 'IssuerId', 'IssuerId2', 'issueridcomposite', 'sysdt', 'systs', 'month', 'year', 'BusinessDate', 'stcd', 'srcnm', 'NetworkName', 'NetworkURL', 'custnum', 'MarketCoverage', 'RejectRows', 'IssuerId', 'IssuerId2', 'issueridcomposite', 'sysdt', 'systs', 'Protocol\_Used', 'BusinessDate', 'stcd', 'srcnm', 'NetworkName', 'NetworkURL', 'custnum', 'MarketCoverage', 'RejectRows', 'IssuerId', 'IssuerId2', 'issueridcomposite', 'sysdt', 'systs', 'curdt', 'curts', 'BusinessDate', 'stcd', 'srcnm', 'NetworkName', 'NetworkURL', 'custnum', 'MarketCoverage', 'RejectRows', 'IssuerId', 'IssuerId2', 'issueridcomposite', 'sysdt', 'systs', 'cleannetworkname']

>>> tuple\_list=tuple(list\_of\_col)

>>> print(tuple\_list)

('BusinessDate', 'stcd', 'srcnm', 'NetworkName', 'NetworkURL', 'custnum', 'MarketCoverage', 'RejectRows', 'IssuerId', 'IssuerId2', 'issueridcomposite', 'sysdt', 'systs', 'month', 'year', 'BusinessDate', 'stcd', 'srcnm', 'NetworkName', 'NetworkURL', 'custnum', 'MarketCoverage', 'RejectRows', 'IssuerId', 'IssuerId2', 'issueridcomposite', 'sysdt', 'systs', 'Protocol\_Used', 'BusinessDate', 'stcd', 'srcnm', 'NetworkName', 'NetworkURL', 'custnum', 'MarketCoverage', 'RejectRows', 'IssuerId', 'IssuerId2', 'issueridcomposite', 'sysdt', 'systs', 'curdt', 'curts', 'BusinessDate', 'stcd', 'srcnm', 'NetworkName', 'NetworkURL', 'custnum', 'MarketCoverage', 'RejectRows', 'IssuerId', 'IssuerId2', 'issueridcomposite', 'sysdt', 'systs', 'cleannetworkname')

>>> set\_list=set(list\_of\_col)

>>> print(set\_list)

{'NetworkURL', 'year', 'issueridcomposite', 'NetworkName', 'RejectRows', 'IssuerId', 'stcd', 'IssuerId2', 'systs', 'sysdt', 'BusinessDate', 'MarketCoverage', 'Protocol\_Used', 'curts', 'curdt', 'month', 'cleannetworkname', 'srcnm', 'custnum'}

>>> list\_col\_no\_duplicate=list(set\_list)

>>> df\_combined.select(list\_col\_no\_duplicate).show(1)

Traceback (most recent call last):

File "<stdin>", line 1, in <module>

File "/usr/local/spark/python/pyspark/sql/dataframe.py", line 1685, in select

jdf = self.\_jdf.select(self.\_jcols(\*cols))

File "/usr/local/spark/python/lib/py4j-0.10.9.3-src.zip/py4j/java\_gateway.py", line 1322, in \_\_call\_\_

File "/usr/local/spark/python/pyspark/sql/utils.py", line 117, in deco

raise converted from None

pyspark.sql.utils.AnalysisException: Reference 'NetworkURL' is ambiguous, could be: a.NetworkURL, b.NetworkURL, c.NetworkURL, d.NetworkURL.

>>> df\_combined.select(list\_col\_no\_duplicate).show(1)

Traceback (most recent call last):

File "<stdin>", line 1, in <module>

File "/usr/local/spark/python/pyspark/sql/dataframe.py", line 1685, in select

jdf = self.\_jdf.select(self.\_jcols(\*cols))

File "/usr/local/spark/python/lib/py4j-0.10.9.3-src.zip/py4j/java\_gateway.py", line 1322, in \_\_call\_\_

File "/usr/local/spark/python/pyspark/sql/utils.py", line 117, in deco

raise converted from None

pyspark.sql.utils.AnalysisException: Reference 'NetworkURL' is ambiguous, could be: a.NetworkURL, b.NetworkURL, c.NetworkURL, d.NetworkURL.

>>>

>>> # Rename columns in each DataFrame before joining

... result\_df\_SQL = result\_df\_SQL.withColumnRenamed("NetworkURL", "a\_NetworkURL") \

... .withColumnRenamed("NetworkName", "a\_NetworkName") \

... .withColumnRenamed("IssuerId", "a\_IssuerId") \

... .withColumnRenamed("BusinessDate", "a\_BusinessDate")

>>>

>>> result\_df\_SQL\_1 = result\_df\_SQL\_1.withColumnRenamed("NetworkURL", "b\_NetworkURL") \

... .withColumnRenamed("NetworkName", "b\_NetworkName") \

... .withColumnRenamed("IssuerId", "b\_IssuerId") \

... .withColumnRenamed("BusinessDate", "b\_BusinessDate")

>>>

>>> result\_df\_SQL\_2 = result\_df\_SQL\_2.withColumnRenamed("NetworkURL", "c\_NetworkURL") \

... .withColumnRenamed("NetworkName", "c\_NetworkName") \

... .withColumnRenamed("IssuerId", "c\_IssuerId") \

... .withColumnRenamed("BusinessDate", "c\_BusinessDate")

>>>

>>> result\_df\_SQL\_3 = result\_df\_SQL\_3.withColumnRenamed("NetworkURL", "d\_NetworkURL") \

... .withColumnRenamed("NetworkName", "d\_NetworkName") \

... .withColumnRenamed("IssuerId", "d\_IssuerId") \

... .withColumnRenamed("BusinessDate", "d\_BusinessDate")

>>>

>>> # Perform the join

... df\_combined = result\_df\_SQL \

... .alias("a") \

... .join(result\_df\_SQL\_1.alias("b"), col("a.IssuerId") == col("b.IssuerId"), how="inner") \

... .join(result\_df\_SQL\_2.alias("c"), col("a.IssuerId") == col("c.IssuerId"), how="inner") \

... .join(result\_df\_SQL\_3.alias("d"), col("a.IssuerId") == col("d.IssuerId"), how="inner")

Traceback (most recent call last):

File "<stdin>", line 4, in <module>

File "/usr/local/spark/python/pyspark/sql/dataframe.py", line 1355, in join

jdf = self.\_jdf.join(other.\_jdf, on, how)

File "/usr/local/spark/python/lib/py4j-0.10.9.3-src.zip/py4j/java\_gateway.py", line 1322, in \_\_call\_\_

File "/usr/local/spark/python/pyspark/sql/utils.py", line 117, in deco

raise converted from None

pyspark.sql.utils.AnalysisException: cannot resolve 'a.IssuerId' given input columns: [a.IssuerId2, b.IssuerId2, a.MarketCoverage, b.MarketCoverage, b.Protocol\_Used, a.RejectRows, b.RejectRows, a.a\_BusinessDate, a.a\_IssuerId, a.a\_NetworkName, a.a\_NetworkURL, b.b\_BusinessDate, b.b\_IssuerId, b.b\_NetworkName, b.b\_NetworkURL, a.custnum, b.custnum, a.issueridcomposite, b.issueridcomposite, a.month, a.srcnm, b.srcnm, a.stcd, b.stcd, a.sysdt, b.sysdt, a.systs, b.systs, a.year];

'Join Inner, ('a.IssuerId = 'b.IssuerId)

:- SubqueryAlias a

: +- Project [BusinessDate#1378 AS a\_BusinessDate#43854, stcd#7759, srcnm#7747, a\_NetworkName#43822, a\_NetworkURL#43806, custnum#1383, MarketCoverage#1384, RejectRows#1386, a\_IssuerId#43838, IssuerId2#8325, issueridcomposite#8901, sysdt#9003, systs#9016, month#30928, year#30929]

: +- Project [BusinessDate#1378, stcd#7759, srcnm#7747, a\_NetworkName#43822, a\_NetworkURL#43806, custnum#1383, MarketCoverage#1384, RejectRows#1386, IssuerId#8312 AS a\_IssuerId#43838, IssuerId2#8325, issueridcomposite#8901, sysdt#9003, systs#9016, month#30928, year#30929]

: +- Project [BusinessDate#1378, stcd#7759, srcnm#7747, NetworkName#1381 AS a\_NetworkName#43822, a\_NetworkURL#43806, custnum#1383, MarketCoverage#1384, RejectRows#1386, IssuerId#8312, IssuerId2#8325, issueridcomposite#8901, sysdt#9003, systs#9016, month#30928, year#30929]

: +- Project [BusinessDate#1378, stcd#7759, srcnm#7747, NetworkName#1381, NetworkURL#1382 AS a\_NetworkURL#43806, custnum#1383, MarketCoverage#1384, RejectRows#1386, IssuerId#8312, IssuerId2#8325, issueridcomposite#8901, sysdt#9003, systs#9016, month#30928, year#30929]

: +- Project [BusinessDate#1378, stcd#7759, srcnm#7747, NetworkName#1381, NetworkURL#1382, custnum#1383, MarketCoverage#1384, RejectRows#1386, IssuerId#8312, IssuerId2#8325, issueridcomposite#8901, sysdt#9003, systs#9016, month(BusinessDate#1378) AS month#30928, year(BusinessDate#1378) AS year#30929]

: +- SubqueryAlias insureview

: +- View (`insureview`, [BusinessDate#1378,stcd#7759,srcnm#7747,NetworkName#1381,NetworkURL#1382,custnum#1383,MarketCoverage#1384,RejectRows#1386,IssuerId#8312,IssuerId2#8325,issueridcomposite#8901,sysdt#9003,systs#9016])

: +- Project [BusinessDate#1378, stcd#7759, srcnm#7747, NetworkName#1381, NetworkURL#1382, custnum#1383, MarketCoverage#1384, RejectRows#1386, IssuerId#8312, IssuerId2#8325, issueridcomposite#8901, sysdt#9003, current\_timestamp() AS systs#9016]

: +- Project [BusinessDate#1378, stcd#7759, srcnm#7747, NetworkName#1381, NetworkURL#1382, custnum#1383, MarketCoverage#1384, RejectRows#1386, IssuerId#8312, IssuerId2#8325, issueridcomposite#8901, current\_date(Some(Asia/Kolkata)) AS sysdt#9003]

: +- Project [BusinessDate#1378, stcd#7759, srcnm#7747, NetworkName#1381, NetworkURL#1382, custnum#1383, MarketCoverage#1384, RejectRows#1386, IssuerId#8312, IssuerId2#8325, issueridcomposite#8901]

: +- Project [BusinessDate#1378, stcd#7759, srcnm#7747, NetworkName#1381, NetworkURL#1382, custnum#1383, MarketCoverage#1384, DentalOnlyPlan#1385, RejectRows#1386, IssuerId#8312, IssuerId2#8325, concat(IssuerId#8312, IssuerId2#8325) AS issueridcomposite#8901]

: +- Project [BusinessDate#1378, stcd#7759, srcnm#7747, NetworkName#1381, NetworkURL#1382, custnum#1383, MarketCoverage#1384, DentalOnlyPlan#1385, RejectRows#1386, IssuerId#8312, IssuerId2#8325]

: +- Project [IssuerIDDD#8273, BusinessDate#1378, stcd#7759, srcnm#7747, NetworkName#1381, NetworkURL#1382, custnum#1383, MarketCoverage#1384, DentalOnlyPlan#1385, RejectRows#1386, IssuerId#8312, IssuerId2#8325]

: +- Project [IssuerIDD#8261, IssuerIDDD#8273, BusinessDate#1378, stcd#7759, srcnm#7747, NetworkName#1381, NetworkURL#1382, custnum#1383, MarketCoverage#1384, DentalOnlyPlan#1385, RejectRows#1386, IssuerId#8312, cast(IssuerIDDD#8273 as string) AS IssuerId2#8325]

: +- Project [IssuerIDD#8261, IssuerIDDD#8273, BusinessDate#1378, stcd#7759, srcnm#7747, NetworkName#1381, NetworkURL#1382, custnum#1383, MarketCoverage#1384, DentalOnlyPlan#1385, RejectRows#1386, cast(IssuerIDD#8261 as string) AS IssuerId#8312]

: +- Project [IssuerIDD#8261, IssuerId2#1377 AS IssuerIDDD#8273, BusinessDate#1378, stcd#7759, srcnm#7747, NetworkName#1381, NetworkURL#1382, custnum#1383, MarketCoverage#1384, DentalOnlyPlan#1385, RejectRows#1386]

: +- Project [IssuerId#1376 AS IssuerIDD#8261, IssuerId2#1377, BusinessDate#1378, stcd#7759, srcnm#7747, NetworkName#1381, NetworkURL#1382, custnum#1383, MarketCoverage#1384, DentalOnlyPlan#1385, RejectRows#1386]

: +- Project [IssuerId#1376, IssuerId2#1377, BusinessDate#1378, StateCode#1379 AS stcd#7759, srcnm#7747, NetworkName#1381, NetworkURL#1382, custnum#1383, MarketCoverage#1384, DentalOnlyPlan#1385, RejectRows#1386]

: +- Project [IssuerId#1376, IssuerId2#1377, BusinessDate#1378, StateCode#1379, SourceName#1380 AS srcnm#7747, NetworkName#1381, NetworkURL#1382, custnum#1383, MarketCoverage#1384, DentalOnlyPlan#1385, RejectRows#1386]

: +- Union false, false

: :- Relation [IssuerId#1376,IssuerId2#1377,BusinessDate#1378,StateCode#1379,SourceName#1380,NetworkName#1381,NetworkURL#1382,custnum#1383,MarketCoverage#1384,DentalOnlyPlan#1385,RejectRows#1386] csv

: +- Relation [IssuerId#491,IssuerId2#492,BusinessDate#493,StateCode#494,SourceName#495,NetworkName#496,NetworkURL#497,custnum#498,MarketCoverage#499,DentalOnlyPlan#500,RejectRows#501] csv

+- SubqueryAlias b

+- Project [BusinessDate#44056 AS b\_BusinessDate#43915, stcd#44080, srcnm#44081, b\_NetworkName#43885, b\_NetworkURL#43870, custnum#44061, MarketCoverage#44062, RejectRows#44064, b\_IssuerId#43900, IssuerId2#44079, issueridcomposite#44078, sysdt#44077, systs#44076, Protocol\_Used#30945]

+- Project [BusinessDate#44056, stcd#44080, srcnm#44081, b\_NetworkName#43885, b\_NetworkURL#43870, custnum#44061, MarketCoverage#44062, RejectRows#44064, IssuerId#8312 AS b\_IssuerId#43900, IssuerId2#44079, issueridcomposite#44078, sysdt#44077, systs#44076, Protocol\_Used#30945]

+- Project [BusinessDate#44056, stcd#44080, srcnm#44081, NetworkName#44059 AS b\_NetworkName#43885, b\_NetworkURL#43870, custnum#44061, MarketCoverage#44062, RejectRows#44064, IssuerId#8312, IssuerId2#44079, issueridcomposite#44078, sysdt#44077, systs#44076, Protocol\_Used#30945]

+- Project [BusinessDate#44056, stcd#44080, srcnm#44081, NetworkName#44059, NetworkURL#44060 AS b\_NetworkURL#43870, custnum#44061, MarketCoverage#44062, RejectRows#44064, IssuerId#8312, IssuerId2#44079, issueridcomposite#44078, sysdt#44077, systs#44076, Protocol\_Used#30945]

+- Project [BusinessDate#44056, stcd#44080, srcnm#44081, NetworkName#44059, NetworkURL#44060, custnum#44061, MarketCoverage#44062, RejectRows#44064, IssuerId#8312, IssuerId2#44079, issueridcomposite#44078, sysdt#44077, systs#44076, CASE WHEN NetworkURL#44060 LIKE https://% THEN http secured WHEN NetworkURL#44060 LIKE http://% THEN http non secured ELSE No Protocol END AS Protocol\_Used#30945]

+- SubqueryAlias insureview

+- View (`insureview`, [BusinessDate#44056,stcd#44080,srcnm#44081,NetworkName#44059,NetworkURL#44060,custnum#44061,MarketCoverage#44062,RejectRows#44064,IssuerId#8312,IssuerId2#44079,issueridcomposite#44078,sysdt#44077,systs#44076])

+- Project [BusinessDate#44056, stcd#44080, srcnm#44081, NetworkName#44059, NetworkURL#44060, custnum#44061, MarketCoverage#44062, RejectRows#44064, IssuerId#8312, IssuerId2#44079, issueridcomposite#44078, sysdt#44077, current\_timestamp() AS systs#44076]

+- Project [BusinessDate#44056, stcd#44080, srcnm#44081, NetworkName#44059, NetworkURL#44060, custnum#44061, MarketCoverage#44062, RejectRows#44064, IssuerId#8312, IssuerId2#44079, issueridcomposite#44078, current\_date(Some(Asia/Kolkata)) AS sysdt#44077]

+- Project [BusinessDate#44056, stcd#44080, srcnm#44081, NetworkName#44059, NetworkURL#44060, custnum#44061, MarketCoverage#44062, RejectRows#44064, IssuerId#8312, IssuerId2#44079, issueridcomposite#44078]

+- Project [BusinessDate#44056, stcd#44080, srcnm#44081, NetworkName#44059, NetworkURL#44060, custnum#44061, MarketCoverage#44062, DentalOnlyPlan#44063, RejectRows#44064, IssuerId#8312, IssuerId2#44079, concat(IssuerId#8312, IssuerId2#44079) AS issueridcomposite#44078]

+- Project [BusinessDate#44056, stcd#44080, srcnm#44081, NetworkName#44059, NetworkURL#44060, custnum#44061, MarketCoverage#44062, DentalOnlyPlan#44063, RejectRows#44064, IssuerId#8312, IssuerId2#44079]

+- Project [IssuerIDDD#8273, BusinessDate#44056, stcd#44080, srcnm#44081, NetworkName#44059, NetworkURL#44060, custnum#44061, MarketCoverage#44062, DentalOnlyPlan#44063, RejectRows#44064, IssuerId#8312, IssuerId2#44079]

+- Project [IssuerIDD#8261, IssuerIDDD#8273, BusinessDate#44056, stcd#44080, srcnm#44081, NetworkName#44059, NetworkURL#44060, custnum#44061, MarketCoverage#44062, DentalOnlyPlan#44063, RejectRows#44064, IssuerId#8312, cast(IssuerIDDD#8273 as string) AS IssuerId2#44079]

+- Project [IssuerIDD#8261, IssuerIDDD#8273, BusinessDate#44056, stcd#44080, srcnm#44081, NetworkName#44059, NetworkURL#44060, custnum#44061, MarketCoverage#44062, DentalOnlyPlan#44063, RejectRows#44064, cast(IssuerIDD#8261 as string) AS IssuerId#8312]

+- Project [IssuerIDD#8261, IssuerId2#44055 AS IssuerIDDD#8273, BusinessDate#44056, stcd#44080, srcnm#44081, NetworkName#44059, NetworkURL#44060, custnum#44061, MarketCoverage#44062, DentalOnlyPlan#44063, RejectRows#44064]

+- Project [IssuerId#44054 AS IssuerIDD#8261, IssuerId2#44055, BusinessDate#44056, stcd#44080, srcnm#44081, NetworkName#44059, NetworkURL#44060, custnum#44061, MarketCoverage#44062, DentalOnlyPlan#44063, RejectRows#44064]

+- Project [IssuerId#44054, IssuerId2#44055, BusinessDate#44056, StateCode#44057 AS stcd#44080, srcnm#44081, NetworkName#44059, NetworkURL#44060, custnum#44061, MarketCoverage#44062, DentalOnlyPlan#44063, RejectRows#44064]

+- Project [IssuerId#44054, IssuerId2#44055, BusinessDate#44056, StateCode#44057, SourceName#44058 AS srcnm#44081, NetworkName#44059, NetworkURL#44060, custnum#44061, MarketCoverage#44062, DentalOnlyPlan#44063, RejectRows#44064]

+- Union false, false

:- Relation [IssuerId#44054,IssuerId2#44055,BusinessDate#44056,StateCode#44057,SourceName#44058,NetworkName#44059,NetworkURL#44060,custnum#44061,MarketCoverage#44062,DentalOnlyPlan#44063,RejectRows#44064] csv

+- Relation [IssuerId#44065,IssuerId2#44066,BusinessDate#44067,StateCode#44068,SourceName#44069,NetworkName#44070,NetworkURL#44071,custnum#44072,MarketCoverage#44073,DentalOnlyPlan#44074,RejectRows#44075] csv

>>>

>>> # Show the columns to ensure uniqueness

... df\_combined.columns

['BusinessDate', 'stcd', 'srcnm', 'NetworkName', 'NetworkURL', 'custnum', 'MarketCoverage', 'RejectRows', 'IssuerId', 'IssuerId2', 'issueridcomposite', 'sysdt', 'systs', 'month', 'year', 'BusinessDate', 'stcd', 'srcnm', 'NetworkName', 'NetworkURL', 'custnum', 'MarketCoverage', 'RejectRows', 'IssuerId', 'IssuerId2', 'issueridcomposite', 'sysdt', 'systs', 'Protocol\_Used', 'BusinessDate', 'stcd', 'srcnm', 'NetworkName', 'NetworkURL', 'custnum', 'MarketCoverage', 'RejectRows', 'IssuerId', 'IssuerId2', 'issueridcomposite', 'sysdt', 'systs', 'curdt', 'curts', 'BusinessDate', 'stcd', 'srcnm', 'NetworkName', 'NetworkURL', 'custnum', 'MarketCoverage', 'RejectRows', 'IssuerId', 'IssuerId2', 'issueridcomposite', 'sysdt', 'systs', 'cleannetworkname']

>>>

>>> from pyspark.sql.functions import col

>>>

>>> # Rename columns in each DataFrame to avoid ambiguity

... df1 = result\_df\_SQL.alias("a").select(

... col("BusinessDate").alias("a\_BusinessDate"),

... col("stcd").alias("a\_stcd"),

... col("srcnm").alias("a\_srcnm"),

... col("NetworkName").alias("a\_NetworkName"),

... col("NetworkURL").alias("a\_NetworkURL"),

... col("custnum").alias("a\_custnum"),

... col("MarketCoverage").alias("a\_MarketCoverage"),

... col("RejectRows").alias("a\_RejectRows"),

... col("IssuerId").alias("a\_IssuerId"),

... col("IssuerId2").alias("a\_IssuerId2"),

... col("issueridcomposite").alias("a\_issueridcomposite"),

... col("sysdt").alias("a\_sysdt"),

... col("systs").alias("a\_systs"),

... col("month").alias("a\_month"),

... col("year").alias("a\_year")

... )

Traceback (most recent call last):

File "<stdin>", line 17, in <module>

File "/usr/local/spark/python/pyspark/sql/dataframe.py", line 1685, in select

jdf = self.\_jdf.select(self.\_jcols(\*cols))

File "/usr/local/spark/python/lib/py4j-0.10.9.3-src.zip/py4j/java\_gateway.py", line 1322, in \_\_call\_\_

File "/usr/local/spark/python/pyspark/sql/utils.py", line 117, in deco

raise converted from None

pyspark.sql.utils.AnalysisException: cannot resolve 'BusinessDate' given input columns: [a.IssuerId2, a.MarketCoverage, a.RejectRows, a.a\_BusinessDate, a.a\_IssuerId, a.a\_NetworkName, a.a\_NetworkURL, a.custnum, a.issueridcomposite, a.month, a.srcnm, a.stcd, a.sysdt, a.systs, a.year];

'Project ['BusinessDate AS a\_BusinessDate#44082, stcd#7759 AS a\_stcd#44083, srcnm#7747 AS a\_srcnm#44084, 'NetworkName AS a\_NetworkName#44085, 'NetworkURL AS a\_NetworkURL#44086, custnum#1383 AS a\_custnum#44087, MarketCoverage#1384 AS a\_MarketCoverage#44088, RejectRows#1386 AS a\_RejectRows#44089, 'IssuerId AS a\_IssuerId#44090, IssuerId2#8325 AS a\_IssuerId2#44091, issueridcomposite#8901 AS a\_issueridcomposite#44092, sysdt#9003 AS a\_sysdt#44093, systs#9016 AS a\_systs#44094, month#30928 AS a\_month#44095, year#30929 AS a\_year#44096]

+- SubqueryAlias a

+- Project [BusinessDate#1378 AS a\_BusinessDate#43854, stcd#7759, srcnm#7747, a\_NetworkName#43822, a\_NetworkURL#43806, custnum#1383, MarketCoverage#1384, RejectRows#1386, a\_IssuerId#43838, IssuerId2#8325, issueridcomposite#8901, sysdt#9003, systs#9016, month#30928, year#30929]

+- Project [BusinessDate#1378, stcd#7759, srcnm#7747, a\_NetworkName#43822, a\_NetworkURL#43806, custnum#1383, MarketCoverage#1384, RejectRows#1386, IssuerId#8312 AS a\_IssuerId#43838, IssuerId2#8325, issueridcomposite#8901, sysdt#9003, systs#9016, month#30928, year#30929]

+- Project [BusinessDate#1378, stcd#7759, srcnm#7747, NetworkName#1381 AS a\_NetworkName#43822, a\_NetworkURL#43806, custnum#1383, MarketCoverage#1384, RejectRows#1386, IssuerId#8312, IssuerId2#8325, issueridcomposite#8901, sysdt#9003, systs#9016, month#30928, year#30929]

+- Project [BusinessDate#1378, stcd#7759, srcnm#7747, NetworkName#1381, NetworkURL#1382 AS a\_NetworkURL#43806, custnum#1383, MarketCoverage#1384, RejectRows#1386, IssuerId#8312, IssuerId2#8325, issueridcomposite#8901, sysdt#9003, systs#9016, month#30928, year#30929]

+- Project [BusinessDate#1378, stcd#7759, srcnm#7747, NetworkName#1381, NetworkURL#1382, custnum#1383, MarketCoverage#1384, RejectRows#1386, IssuerId#8312, IssuerId2#8325, issueridcomposite#8901, sysdt#9003, systs#9016, month(BusinessDate#1378) AS month#30928, year(BusinessDate#1378) AS year#30929]

+- SubqueryAlias insureview

+- View (`insureview`, [BusinessDate#1378,stcd#7759,srcnm#7747,NetworkName#1381,NetworkURL#1382,custnum#1383,MarketCoverage#1384,RejectRows#1386,IssuerId#8312,IssuerId2#8325,issueridcomposite#8901,sysdt#9003,systs#9016])

+- Project [BusinessDate#1378, stcd#7759, srcnm#7747, NetworkName#1381, NetworkURL#1382, custnum#1383, MarketCoverage#1384, RejectRows#1386, IssuerId#8312, IssuerId2#8325, issueridcomposite#8901, sysdt#9003, current\_timestamp() AS systs#9016]

+- Project [BusinessDate#1378, stcd#7759, srcnm#7747, NetworkName#1381, NetworkURL#1382, custnum#1383, MarketCoverage#1384, RejectRows#1386, IssuerId#8312, IssuerId2#8325, issueridcomposite#8901, current\_date(Some(Asia/Kolkata)) AS sysdt#9003]

+- Project [BusinessDate#1378, stcd#7759, srcnm#7747, NetworkName#1381, NetworkURL#1382, custnum#1383, MarketCoverage#1384, RejectRows#1386, IssuerId#8312, IssuerId2#8325, issueridcomposite#8901]

+- Project [BusinessDate#1378, stcd#7759, srcnm#7747, NetworkName#1381, NetworkURL#1382, custnum#1383, MarketCoverage#1384, DentalOnlyPlan#1385, RejectRows#1386, IssuerId#8312, IssuerId2#8325, concat(IssuerId#8312, IssuerId2#8325) AS issueridcomposite#8901]

+- Project [BusinessDate#1378, stcd#7759, srcnm#7747, NetworkName#1381, NetworkURL#1382, custnum#1383, MarketCoverage#1384, DentalOnlyPlan#1385, RejectRows#1386, IssuerId#8312, IssuerId2#8325]

+- Project [IssuerIDDD#8273, BusinessDate#1378, stcd#7759, srcnm#7747, NetworkName#1381, NetworkURL#1382, custnum#1383, MarketCoverage#1384, DentalOnlyPlan#1385, RejectRows#1386, IssuerId#8312, IssuerId2#8325]

+- Project [IssuerIDD#8261, IssuerIDDD#8273, BusinessDate#1378, stcd#7759, srcnm#7747, NetworkName#1381, NetworkURL#1382, custnum#1383, MarketCoverage#1384, DentalOnlyPlan#1385, RejectRows#1386, IssuerId#8312, cast(IssuerIDDD#8273 as string) AS IssuerId2#8325]

+- Project [IssuerIDD#8261, IssuerIDDD#8273, BusinessDate#1378, stcd#7759, srcnm#7747, NetworkName#1381, NetworkURL#1382, custnum#1383, MarketCoverage#1384, DentalOnlyPlan#1385, RejectRows#1386, cast(IssuerIDD#8261 as string) AS IssuerId#8312]

+- Project [IssuerIDD#8261, IssuerId2#1377 AS IssuerIDDD#8273, BusinessDate#1378, stcd#7759, srcnm#7747, NetworkName#1381, NetworkURL#1382, custnum#1383, MarketCoverage#1384, DentalOnlyPlan#1385, RejectRows#1386]

+- Project [IssuerId#1376 AS IssuerIDD#8261, IssuerId2#1377, BusinessDate#1378, stcd#7759, srcnm#7747, NetworkName#1381, NetworkURL#1382, custnum#1383, MarketCoverage#1384, DentalOnlyPlan#1385, RejectRows#1386]

+- Project [IssuerId#1376, IssuerId2#1377, BusinessDate#1378, StateCode#1379 AS stcd#7759, srcnm#7747, NetworkName#1381, NetworkURL#1382, custnum#1383, MarketCoverage#1384, DentalOnlyPlan#1385, RejectRows#1386]

+- Project [IssuerId#1376, IssuerId2#1377, BusinessDate#1378, StateCode#1379, SourceName#1380 AS srcnm#7747, NetworkName#1381, NetworkURL#1382, custnum#1383, MarketCoverage#1384, DentalOnlyPlan#1385, RejectRows#1386]

+- Union false, false

:- Relation [IssuerId#1376,IssuerId2#1377,BusinessDate#1378,StateCode#1379,SourceName#1380,NetworkName#1381,NetworkURL#1382,custnum#1383,MarketCoverage#1384,DentalOnlyPlan#1385,RejectRows#1386] csv

+- Relation [IssuerId#491,IssuerId2#492,BusinessDate#493,StateCode#494,SourceName#495,NetworkName#496,NetworkURL#497,custnum#498,MarketCoverage#499,DentalOnlyPlan#500,RejectRows#501] csv

>>>

>>> df2 = result\_df\_SQL\_1.alias("b").select(

... col("BusinessDate").alias("b\_BusinessDate"),

... col("stcd").alias("b\_stcd"),

... col("srcnm").alias("b\_srcnm"),

... col("NetworkName").alias("b\_NetworkName"),

... col("NetworkURL").alias("b\_NetworkURL"),

... col("custnum").alias("b\_custnum"),

... col("MarketCoverage").alias("b\_MarketCoverage"),

... col("RejectRows").alias("b\_RejectRows"),

... col("IssuerId").alias("b\_IssuerId"),

... col("IssuerId2").alias("b\_IssuerId2"),

... col("issueridcomposite").alias("b\_issueridcomposite"),

... col("sysdt").alias("b\_sysdt"),

... col("systs").alias("b\_systs"),

... col("Protocol\_Used").alias("b\_Protocol\_Used")

... )

Traceback (most recent call last):

File "<stdin>", line 15, in <module>

File "/usr/local/spark/python/pyspark/sql/dataframe.py", line 1685, in select

jdf = self.\_jdf.select(self.\_jcols(\*cols))

File "/usr/local/spark/python/lib/py4j-0.10.9.3-src.zip/py4j/java\_gateway.py", line 1322, in \_\_call\_\_

File "/usr/local/spark/python/pyspark/sql/utils.py", line 117, in deco

raise converted from None

pyspark.sql.utils.AnalysisException: cannot resolve 'BusinessDate' given input columns: [b.IssuerId2, b.MarketCoverage, b.Protocol\_Used, b.RejectRows, b.b\_BusinessDate, b.b\_IssuerId, b.b\_NetworkName, b.b\_NetworkURL, b.custnum, b.issueridcomposite, b.srcnm, b.stcd, b.sysdt, b.systs];

'Project ['BusinessDate AS b\_BusinessDate#44097, stcd#7759 AS b\_stcd#44098, srcnm#7747 AS b\_srcnm#44099, 'NetworkName AS b\_NetworkName#44100, 'NetworkURL AS b\_NetworkURL#44101, custnum#1383 AS b\_custnum#44102, MarketCoverage#1384 AS b\_MarketCoverage#44103, RejectRows#1386 AS b\_RejectRows#44104, 'IssuerId AS b\_IssuerId#44105, IssuerId2#8325 AS b\_IssuerId2#44106, issueridcomposite#8901 AS b\_issueridcomposite#44107, sysdt#9003 AS b\_sysdt#44108, systs#9016 AS b\_systs#44109, Protocol\_Used#30945 AS b\_Protocol\_Used#44110]

+- SubqueryAlias b

+- Project [BusinessDate#1378 AS b\_BusinessDate#43915, stcd#7759, srcnm#7747, b\_NetworkName#43885, b\_NetworkURL#43870, custnum#1383, MarketCoverage#1384, RejectRows#1386, b\_IssuerId#43900, IssuerId2#8325, issueridcomposite#8901, sysdt#9003, systs#9016, Protocol\_Used#30945]

+- Project [BusinessDate#1378, stcd#7759, srcnm#7747, b\_NetworkName#43885, b\_NetworkURL#43870, custnum#1383, MarketCoverage#1384, RejectRows#1386, IssuerId#8312 AS b\_IssuerId#43900, IssuerId2#8325, issueridcomposite#8901, sysdt#9003, systs#9016, Protocol\_Used#30945]

+- Project [BusinessDate#1378, stcd#7759, srcnm#7747, NetworkName#1381 AS b\_NetworkName#43885, b\_NetworkURL#43870, custnum#1383, MarketCoverage#1384, RejectRows#1386, IssuerId#8312, IssuerId2#8325, issueridcomposite#8901, sysdt#9003, systs#9016, Protocol\_Used#30945]

+- Project [BusinessDate#1378, stcd#7759, srcnm#7747, NetworkName#1381, NetworkURL#1382 AS b\_NetworkURL#43870, custnum#1383, MarketCoverage#1384, RejectRows#1386, IssuerId#8312, IssuerId2#8325, issueridcomposite#8901, sysdt#9003, systs#9016, Protocol\_Used#30945]

+- Project [BusinessDate#1378, stcd#7759, srcnm#7747, NetworkName#1381, NetworkURL#1382, custnum#1383, MarketCoverage#1384, RejectRows#1386, IssuerId#8312, IssuerId2#8325, issueridcomposite#8901, sysdt#9003, systs#9016, CASE WHEN NetworkURL#1382 LIKE https://% THEN http secured WHEN NetworkURL#1382 LIKE http://% THEN http non secured ELSE No Protocol END AS Protocol\_Used#30945]

+- SubqueryAlias insureview

+- View (`insureview`, [BusinessDate#1378,stcd#7759,srcnm#7747,NetworkName#1381,NetworkURL#1382,custnum#1383,MarketCoverage#1384,RejectRows#1386,IssuerId#8312,IssuerId2#8325,issueridcomposite#8901,sysdt#9003,systs#9016])

+- Project [BusinessDate#1378, stcd#7759, srcnm#7747, NetworkName#1381, NetworkURL#1382, custnum#1383, MarketCoverage#1384, RejectRows#1386, IssuerId#8312, IssuerId2#8325, issueridcomposite#8901, sysdt#9003, current\_timestamp() AS systs#9016]

+- Project [BusinessDate#1378, stcd#7759, srcnm#7747, NetworkName#1381, NetworkURL#1382, custnum#1383, MarketCoverage#1384, RejectRows#1386, IssuerId#8312, IssuerId2#8325, issueridcomposite#8901, current\_date(Some(Asia/Kolkata)) AS sysdt#9003]

+- Project [BusinessDate#1378, stcd#7759, srcnm#7747, NetworkName#1381, NetworkURL#1382, custnum#1383, MarketCoverage#1384, RejectRows#1386, IssuerId#8312, IssuerId2#8325, issueridcomposite#8901]

+- Project [BusinessDate#1378, stcd#7759, srcnm#7747, NetworkName#1381, NetworkURL#1382, custnum#1383, MarketCoverage#1384, DentalOnlyPlan#1385, RejectRows#1386, IssuerId#8312, IssuerId2#8325, concat(IssuerId#8312, IssuerId2#8325) AS issueridcomposite#8901]

+- Project [BusinessDate#1378, stcd#7759, srcnm#7747, NetworkName#1381, NetworkURL#1382, custnum#1383, MarketCoverage#1384, DentalOnlyPlan#1385, RejectRows#1386, IssuerId#8312, IssuerId2#8325]

+- Project [IssuerIDDD#8273, BusinessDate#1378, stcd#7759, srcnm#7747, NetworkName#1381, NetworkURL#1382, custnum#1383, MarketCoverage#1384, DentalOnlyPlan#1385, RejectRows#1386, IssuerId#8312, IssuerId2#8325]

+- Project [IssuerIDD#8261, IssuerIDDD#8273, BusinessDate#1378, stcd#7759, srcnm#7747, NetworkName#1381, NetworkURL#1382, custnum#1383, MarketCoverage#1384, DentalOnlyPlan#1385, RejectRows#1386, IssuerId#8312, cast(IssuerIDDD#8273 as string) AS IssuerId2#8325]

+- Project [IssuerIDD#8261, IssuerIDDD#8273, BusinessDate#1378, stcd#7759, srcnm#7747, NetworkName#1381, NetworkURL#1382, custnum#1383, MarketCoverage#1384, DentalOnlyPlan#1385, RejectRows#1386, cast(IssuerIDD#8261 as string) AS IssuerId#8312]

+- Project [IssuerIDD#8261, IssuerId2#1377 AS IssuerIDDD#8273, BusinessDate#1378, stcd#7759, srcnm#7747, NetworkName#1381, NetworkURL#1382, custnum#1383, MarketCoverage#1384, DentalOnlyPlan#1385, RejectRows#1386]

+- Project [IssuerId#1376 AS IssuerIDD#8261, IssuerId2#1377, BusinessDate#1378, stcd#7759, srcnm#7747, NetworkName#1381, NetworkURL#1382, custnum#1383, MarketCoverage#1384, DentalOnlyPlan#1385, RejectRows#1386]

+- Project [IssuerId#1376, IssuerId2#1377, BusinessDate#1378, StateCode#1379 AS stcd#7759, srcnm#7747, NetworkName#1381, NetworkURL#1382, custnum#1383, MarketCoverage#1384, DentalOnlyPlan#1385, RejectRows#1386]

+- Project [IssuerId#1376, IssuerId2#1377, BusinessDate#1378, StateCode#1379, SourceName#1380 AS srcnm#7747, NetworkName#1381, NetworkURL#1382, custnum#1383, MarketCoverage#1384, DentalOnlyPlan#1385, RejectRows#1386]

+- Union false, false

:- Relation [IssuerId#1376,IssuerId2#1377,BusinessDate#1378,StateCode#1379,SourceName#1380,NetworkName#1381,NetworkURL#1382,custnum#1383,MarketCoverage#1384,DentalOnlyPlan#1385,RejectRows#1386] csv

+- Relation [IssuerId#491,IssuerId2#492,BusinessDate#493,StateCode#494,SourceName#495,NetworkName#496,NetworkURL#497,custnum#498,MarketCoverage#499,DentalOnlyPlan#500,RejectRows#501] csv

>>>

>>> df3 = result\_df\_SQL\_2.alias("c").select(

... col("BusinessDate").alias("c\_BusinessDate"),

... col("stcd").alias("c\_stcd"),

... col("srcnm").alias("c\_srcnm"),

... col("NetworkName").alias("c\_NetworkName"),

... col("NetworkURL").alias("c\_NetworkURL"),

... col("custnum").alias("c\_custnum"),

... col("MarketCoverage").alias("c\_MarketCoverage"),

... col("RejectRows").alias("c\_RejectRows"),

... col("IssuerId").alias("c\_IssuerId"),

... col("IssuerId2").alias("c\_IssuerId2"),

... col("issueridcomposite").alias("c\_issueridcomposite"),

... col("sysdt").alias("c\_sysdt"),

... col("systs").alias("c\_systs"),

... col("Protocol\_Used").alias("c\_Protocol\_Used")

... )

Traceback (most recent call last):

File "<stdin>", line 15, in <module>

File "/usr/local/spark/python/pyspark/sql/dataframe.py", line 1685, in select

jdf = self.\_jdf.select(self.\_jcols(\*cols))

File "/usr/local/spark/python/lib/py4j-0.10.9.3-src.zip/py4j/java\_gateway.py", line 1322, in \_\_call\_\_

File "/usr/local/spark/python/pyspark/sql/utils.py", line 117, in deco

raise converted from None

pyspark.sql.utils.AnalysisException: cannot resolve 'BusinessDate' given input columns: [c.IssuerId2, c.MarketCoverage, c.RejectRows, c.c\_BusinessDate, c.c\_IssuerId, c.c\_NetworkName, c.c\_NetworkURL, c.curdt, c.curts, c.custnum, c.issueridcomposite, c.srcnm, c.stcd, c.sysdt, c.systs];

'Project ['BusinessDate AS c\_BusinessDate#44111, stcd#7759 AS c\_stcd#44112, srcnm#7747 AS c\_srcnm#44113, 'NetworkName AS c\_NetworkName#44114, 'NetworkURL AS c\_NetworkURL#44115, custnum#1383 AS c\_custnum#44116, MarketCoverage#1384 AS c\_MarketCoverage#44117, RejectRows#1386 AS c\_RejectRows#44118, 'IssuerId AS c\_IssuerId#44119, IssuerId2#8325 AS c\_IssuerId2#44120, issueridcomposite#8901 AS c\_issueridcomposite#44121, sysdt#9003 AS c\_sysdt#44122, systs#9016 AS c\_systs#44123, 'Protocol\_Used AS c\_Protocol\_Used#44124]

+- SubqueryAlias c

+- Project [BusinessDate#1378 AS c\_BusinessDate#43978, stcd#7759, srcnm#7747, c\_NetworkName#43946, c\_NetworkURL#43930, custnum#1383, MarketCoverage#1384, RejectRows#1386, c\_IssuerId#43962, IssuerId2#8325, issueridcomposite#8901, sysdt#9003, systs#9016, curdt#31490, curts#31491]

+- Project [BusinessDate#1378, stcd#7759, srcnm#7747, c\_NetworkName#43946, c\_NetworkURL#43930, custnum#1383, MarketCoverage#1384, RejectRows#1386, IssuerId#8312 AS c\_IssuerId#43962, IssuerId2#8325, issueridcomposite#8901, sysdt#9003, systs#9016, curdt#31490, curts#31491]

+- Project [BusinessDate#1378, stcd#7759, srcnm#7747, NetworkName#1381 AS c\_NetworkName#43946, c\_NetworkURL#43930, custnum#1383, MarketCoverage#1384, RejectRows#1386, IssuerId#8312, IssuerId2#8325, issueridcomposite#8901, sysdt#9003, systs#9016, curdt#31490, curts#31491]

+- Project [BusinessDate#1378, stcd#7759, srcnm#7747, NetworkName#1381, NetworkURL#1382 AS c\_NetworkURL#43930, custnum#1383, MarketCoverage#1384, RejectRows#1386, IssuerId#8312, IssuerId2#8325, issueridcomposite#8901, sysdt#9003, systs#9016, curdt#31490, curts#31491]

+- Project [BusinessDate#1378, stcd#7759, srcnm#7747, NetworkName#1381, NetworkURL#1382, custnum#1383, MarketCoverage#1384, RejectRows#1386, IssuerId#8312, IssuerId2#8325, issueridcomposite#8901, sysdt#9003, systs#9016, current\_date(Some(Asia/Kolkata)) AS curdt#31490, current\_timestamp() AS curts#31491]

+- SubqueryAlias insureview

+- View (`insureview`, [BusinessDate#1378,stcd#7759,srcnm#7747,NetworkName#1381,NetworkURL#1382,custnum#1383,MarketCoverage#1384,RejectRows#1386,IssuerId#8312,IssuerId2#8325,issueridcomposite#8901,sysdt#9003,systs#9016])

+- Project [BusinessDate#1378, stcd#7759, srcnm#7747, NetworkName#1381, NetworkURL#1382, custnum#1383, MarketCoverage#1384, RejectRows#1386, IssuerId#8312, IssuerId2#8325, issueridcomposite#8901, sysdt#9003, current\_timestamp() AS systs#9016]

+- Project [BusinessDate#1378, stcd#7759, srcnm#7747, NetworkName#1381, NetworkURL#1382, custnum#1383, MarketCoverage#1384, RejectRows#1386, IssuerId#8312, IssuerId2#8325, issueridcomposite#8901, current\_date(Some(Asia/Kolkata)) AS sysdt#9003]

+- Project [BusinessDate#1378, stcd#7759, srcnm#7747, NetworkName#1381, NetworkURL#1382, custnum#1383, MarketCoverage#1384, RejectRows#1386, IssuerId#8312, IssuerId2#8325, issueridcomposite#8901]

+- Project [BusinessDate#1378, stcd#7759, srcnm#7747, NetworkName#1381, NetworkURL#1382, custnum#1383, MarketCoverage#1384, DentalOnlyPlan#1385, RejectRows#1386, IssuerId#8312, IssuerId2#8325, concat(IssuerId#8312, IssuerId2#8325) AS issueridcomposite#8901]

+- Project [BusinessDate#1378, stcd#7759, srcnm#7747, NetworkName#1381, NetworkURL#1382, custnum#1383, MarketCoverage#1384, DentalOnlyPlan#1385, RejectRows#1386, IssuerId#8312, IssuerId2#8325]

+- Project [IssuerIDDD#8273, BusinessDate#1378, stcd#7759, srcnm#7747, NetworkName#1381, NetworkURL#1382, custnum#1383, MarketCoverage#1384, DentalOnlyPlan#1385, RejectRows#1386, IssuerId#8312, IssuerId2#8325]

+- Project [IssuerIDD#8261, IssuerIDDD#8273, BusinessDate#1378, stcd#7759, srcnm#7747, NetworkName#1381, NetworkURL#1382, custnum#1383, MarketCoverage#1384, DentalOnlyPlan#1385, RejectRows#1386, IssuerId#8312, cast(IssuerIDDD#8273 as string) AS IssuerId2#8325]

+- Project [IssuerIDD#8261, IssuerIDDD#8273, BusinessDate#1378, stcd#7759, srcnm#7747, NetworkName#1381, NetworkURL#1382, custnum#1383, MarketCoverage#1384, DentalOnlyPlan#1385, RejectRows#1386, cast(IssuerIDD#8261 as string) AS IssuerId#8312]

+- Project [IssuerIDD#8261, IssuerId2#1377 AS IssuerIDDD#8273, BusinessDate#1378, stcd#7759, srcnm#7747, NetworkName#1381, NetworkURL#1382, custnum#1383, MarketCoverage#1384, DentalOnlyPlan#1385, RejectRows#1386]

+- Project [IssuerId#1376 AS IssuerIDD#8261, IssuerId2#1377, BusinessDate#1378, stcd#7759, srcnm#7747, NetworkName#1381, NetworkURL#1382, custnum#1383, MarketCoverage#1384, DentalOnlyPlan#1385, RejectRows#1386]

+- Project [IssuerId#1376, IssuerId2#1377, BusinessDate#1378, StateCode#1379 AS stcd#7759, srcnm#7747, NetworkName#1381, NetworkURL#1382, custnum#1383, MarketCoverage#1384, DentalOnlyPlan#1385, RejectRows#1386]

+- Project [IssuerId#1376, IssuerId2#1377, BusinessDate#1378, StateCode#1379, SourceName#1380 AS srcnm#7747, NetworkName#1381, NetworkURL#1382, custnum#1383, MarketCoverage#1384, DentalOnlyPlan#1385, RejectRows#1386]

+- Union false, false

:- Relation [IssuerId#1376,IssuerId2#1377,BusinessDate#1378,StateCode#1379,SourceName#1380,NetworkName#1381,NetworkURL#1382,custnum#1383,MarketCoverage#1384,DentalOnlyPlan#1385,RejectRows#1386] csv

+- Relation [IssuerId#491,IssuerId2#492,BusinessDate#493,StateCode#494,SourceName#495,NetworkName#496,NetworkURL#497,custnum#498,MarketCoverage#499,DentalOnlyPlan#500,RejectRows#501] csv

>>>

>>> df4 = result\_df\_SQL\_3.alias("d").select(

... col("BusinessDate").alias("d\_BusinessDate"),

... col("stcd").alias("d\_stcd"),

... col("srcnm").alias("d\_srcnm"),

... col("NetworkName").alias("d\_NetworkName"),

... col("NetworkURL").alias("d\_NetworkURL"),

... col("custnum").alias("d\_custnum"),

... col("MarketCoverage").alias("d\_MarketCoverage"),

... col("RejectRows").alias("d\_RejectRows"),

... col("IssuerId").alias("d\_IssuerId"),

... col("IssuerId2").alias("d\_IssuerId2"),

... col("issueridcomposite").alias("d\_issueridcomposite"),

... col("sysdt").alias("d\_sysdt"),

... col("systs").alias("d\_systs"),

... col("cleannetworkname").alias("d\_cleannetworkname")

... )

Traceback (most recent call last):

File "<stdin>", line 15, in <module>

File "/usr/local/spark/python/pyspark/sql/dataframe.py", line 1685, in select

jdf = self.\_jdf.select(self.\_jcols(\*cols))

File "/usr/local/spark/python/lib/py4j-0.10.9.3-src.zip/py4j/java\_gateway.py", line 1322, in \_\_call\_\_

File "/usr/local/spark/python/pyspark/sql/utils.py", line 117, in deco

raise converted from None

pyspark.sql.utils.AnalysisException: cannot resolve 'BusinessDate' given input columns: [d.IssuerId2, d.MarketCoverage, d.RejectRows, d.cleannetworkname, d.custnum, d.d\_BusinessDate, d.d\_IssuerId, d.d\_NetworkName, d.d\_NetworkURL, d.issueridcomposite, d.srcnm, d.stcd, d.sysdt, d.systs];

'Project ['BusinessDate AS d\_BusinessDate#44125, stcd#7759 AS d\_stcd#44126, srcnm#7747 AS d\_srcnm#44127, 'NetworkName AS d\_NetworkName#44128, 'NetworkURL AS d\_NetworkURL#44129, custnum#1383 AS d\_custnum#44130, MarketCoverage#1384 AS d\_MarketCoverage#44131, RejectRows#1386 AS d\_RejectRows#44132, 'IssuerId AS d\_IssuerId#44133, IssuerId2#8325 AS d\_IssuerId2#44134, issueridcomposite#8901 AS d\_issueridcomposite#44135, sysdt#9003 AS d\_sysdt#44136, systs#9016 AS d\_systs#44137, cleannetworkname#32033 AS d\_cleannetworkname#44138]

+- SubqueryAlias d

+- Project [BusinessDate#1378 AS d\_BusinessDate#44039, stcd#7759, srcnm#7747, d\_NetworkName#44009, d\_NetworkURL#43994, custnum#1383, MarketCoverage#1384, RejectRows#1386, d\_IssuerId#44024, IssuerId2#8325, issueridcomposite#8901, sysdt#9003, systs#9016, cleannetworkname#32033]

+- Project [BusinessDate#1378, stcd#7759, srcnm#7747, d\_NetworkName#44009, d\_NetworkURL#43994, custnum#1383, MarketCoverage#1384, RejectRows#1386, IssuerId#8312 AS d\_IssuerId#44024, IssuerId2#8325, issueridcomposite#8901, sysdt#9003, systs#9016, cleannetworkname#32033]

+- Project [BusinessDate#1378, stcd#7759, srcnm#7747, NetworkName#1381 AS d\_NetworkName#44009, d\_NetworkURL#43994, custnum#1383, MarketCoverage#1384, RejectRows#1386, IssuerId#8312, IssuerId2#8325, issueridcomposite#8901, sysdt#9003, systs#9016, cleannetworkname#32033]

+- Project [BusinessDate#1378, stcd#7759, srcnm#7747, NetworkName#1381, NetworkURL#1382 AS d\_NetworkURL#43994, custnum#1383, MarketCoverage#1384, RejectRows#1386, IssuerId#8312, IssuerId2#8325, issueridcomposite#8901, sysdt#9003, systs#9016, cleannetworkname#32033]

+- Project [BusinessDate#1378, stcd#7759, srcnm#7747, NetworkName#1381, NetworkURL#1382, custnum#1383, MarketCoverage#1384, RejectRows#1386, IssuerId#8312, IssuerId2#8325, issueridcomposite#8901, sysdt#9003, systs#9016, sql\_function(NetworkName#1381) AS cleannetworkname#32033]

+- SubqueryAlias insureview

+- View (`insureview`, [BusinessDate#1378,stcd#7759,srcnm#7747,NetworkName#1381,NetworkURL#1382,custnum#1383,MarketCoverage#1384,RejectRows#1386,IssuerId#8312,IssuerId2#8325,issueridcomposite#8901,sysdt#9003,systs#9016])

+- Project [BusinessDate#1378, stcd#7759, srcnm#7747, NetworkName#1381, NetworkURL#1382, custnum#1383, MarketCoverage#1384, RejectRows#1386, IssuerId#8312, IssuerId2#8325, issueridcomposite#8901, sysdt#9003, current\_timestamp() AS systs#9016]

+- Project [BusinessDate#1378, stcd#7759, srcnm#7747, NetworkName#1381, NetworkURL#1382, custnum#1383, MarketCoverage#1384, RejectRows#1386, IssuerId#8312, IssuerId2#8325, issueridcomposite#8901, current\_date(Some(Asia/Kolkata)) AS sysdt#9003]

+- Project [BusinessDate#1378, stcd#7759, srcnm#7747, NetworkName#1381, NetworkURL#1382, custnum#1383, MarketCoverage#1384, RejectRows#1386, IssuerId#8312, IssuerId2#8325, issueridcomposite#8901]

+- Project [BusinessDate#1378, stcd#7759, srcnm#7747, NetworkName#1381, NetworkURL#1382, custnum#1383, MarketCoverage#1384, DentalOnlyPlan#1385, RejectRows#1386, IssuerId#8312, IssuerId2#8325, concat(IssuerId#8312, IssuerId2#8325) AS issueridcomposite#8901]

+- Project [BusinessDate#1378, stcd#7759, srcnm#7747, NetworkName#1381, NetworkURL#1382, custnum#1383, MarketCoverage#1384, DentalOnlyPlan#1385, RejectRows#1386, IssuerId#8312, IssuerId2#8325]

+- Project [IssuerIDDD#8273, BusinessDate#1378, stcd#7759, srcnm#7747, NetworkName#1381, NetworkURL#1382, custnum#1383, MarketCoverage#1384, DentalOnlyPlan#1385, RejectRows#1386, IssuerId#8312, IssuerId2#8325]

+- Project [IssuerIDD#8261, IssuerIDDD#8273, BusinessDate#1378, stcd#7759, srcnm#7747, NetworkName#1381, NetworkURL#1382, custnum#1383, MarketCoverage#1384, DentalOnlyPlan#1385, RejectRows#1386, IssuerId#8312, cast(IssuerIDDD#8273 as string) AS IssuerId2#8325]

+- Project [IssuerIDD#8261, IssuerIDDD#8273, BusinessDate#1378, stcd#7759, srcnm#7747, NetworkName#1381, NetworkURL#1382, custnum#1383, MarketCoverage#1384, DentalOnlyPlan#1385, RejectRows#1386, cast(IssuerIDD#8261 as string) AS IssuerId#8312]

+- Project [IssuerIDD#8261, IssuerId2#1377 AS IssuerIDDD#8273, BusinessDate#1378, stcd#7759, srcnm#7747, NetworkName#1381, NetworkURL#1382, custnum#1383, MarketCoverage#1384, DentalOnlyPlan#1385, RejectRows#1386]

+- Project [IssuerId#1376 AS IssuerIDD#8261, IssuerId2#1377, BusinessDate#1378, stcd#7759, srcnm#7747, NetworkName#1381, NetworkURL#1382, custnum#1383, MarketCoverage#1384, DentalOnlyPlan#1385, RejectRows#1386]

+- Project [IssuerId#1376, IssuerId2#1377, BusinessDate#1378, StateCode#1379 AS stcd#7759, srcnm#7747, NetworkName#1381, NetworkURL#1382, custnum#1383, MarketCoverage#1384, DentalOnlyPlan#1385, RejectRows#1386]

+- Project [IssuerId#1376, IssuerId2#1377, BusinessDate#1378, StateCode#1379, SourceName#1380 AS srcnm#7747, NetworkName#1381, NetworkURL#1382, custnum#1383, MarketCoverage#1384, DentalOnlyPlan#1385, RejectRows#1386]

+- Union false, false

:- Relation [IssuerId#1376,IssuerId2#1377,BusinessDate#1378,StateCode#1379,SourceName#1380,NetworkName#1381,NetworkURL#1382,custnum#1383,MarketCoverage#1384,DentalOnlyPlan#1385,RejectRows#1386] csv

+- Relation [IssuerId#491,IssuerId2#492,BusinessDate#493,StateCode#494,SourceName#495,NetworkName#496,NetworkURL#497,custnum#498,MarketCoverage#499,DentalOnlyPlan#500,RejectRows#501] csv

>>>

>>> # Perform the join with renamed columns

... df\_combined = df1 \

... .join(df2, col("a\_IssuerId") == col("b\_IssuerId"), how="inner") \

... .join(df3, col("a\_IssuerId") == col("c\_IssuerId"), how="inner") \

... .join(df4, col("a\_IssuerId") == col("d\_IssuerId"), how="inner")

Traceback (most recent call last):

File "<stdin>", line 2, in <module>

NameError: name 'df1' is not defined

>>>

>>> # Show the columns to verify uniqueness

... df\_combined.columns

['BusinessDate', 'stcd', 'srcnm', 'NetworkName', 'NetworkURL', 'custnum', 'MarketCoverage', 'RejectRows', 'IssuerId', 'IssuerId2', 'issueridcomposite', 'sysdt', 'systs', 'month', 'year', 'BusinessDate', 'stcd', 'srcnm', 'NetworkName', 'NetworkURL', 'custnum', 'MarketCoverage', 'RejectRows', 'IssuerId', 'IssuerId2', 'issueridcomposite', 'sysdt', 'systs', 'Protocol\_Used', 'BusinessDate', 'stcd', 'srcnm', 'NetworkName', 'NetworkURL', 'custnum', 'MarketCoverage', 'RejectRows', 'IssuerId', 'IssuerId2', 'issueridcomposite', 'sysdt', 'systs', 'curdt', 'curts', 'BusinessDate', 'stcd', 'srcnm', 'NetworkName', 'NetworkURL', 'custnum', 'MarketCoverage', 'RejectRows', 'IssuerId', 'IssuerId2', 'issueridcomposite', 'sysdt', 'systs', 'cleannetworkname']

>>> result\_df\_SQL.describe

<bound method DataFrame.describe of DataFrame[a\_BusinessDate: date, stcd: string, srcnm: string, a\_NetworkName: string, a\_NetworkURL: string, custnum: string, MarketCoverage: string, RejectRows: string, a\_IssuerId: string, IssuerId2: string, issueridcomposite: string, sysdt: date, systs: timestamp, month: int, year: int]>

>>> result\_df\_SQL\_!.describe

File "<stdin>", line 1

result\_df\_SQL\_!.describe

^

SyntaxError: invalid syntax

>>> result\_df\_SQL\_!.describe

>>> result\_df\_SQL.describe

<bound method DataFrame.describe of DataFrame[a\_BusinessDate: date, stcd: string, srcnm: string, a\_NetworkName: string, a\_NetworkURL: string, custnum: string, MarketCoverage: string, RejectRows: string, a\_IssuerId: string, IssuerId2: string, issueridcomposite: string, sysdt: date, systs: timestamp, month: int, year: int]>

>>> result\_df\_SQL\_!.describe

File "<stdin>", line 1

result\_df\_SQL\_!.describe

^

SyntaxError: invalid syntax

>>> result\_df\_SQL.describe

<bound method DataFrame.describe of DataFrame[a\_BusinessDate: date, stcd: string, srcnm: string, a\_NetworkName: string, a\_NetworkURL: string, custnum: string, MarketCoverage: string, RejectRows: string, a\_IssuerId: string, IssuerId2: string, issueridcomposite: string, sysdt: date, systs: timestamp, month: int, year: int]>

>>> result\_df\_SQL\_1.describe

<bound method DataFrame.describe of DataFrame[b\_BusinessDate: date, stcd: string, srcnm: string, b\_NetworkName: string, b\_NetworkURL: string, custnum: string, MarketCoverage: string, RejectRows: string, b\_IssuerId: string, IssuerId2: string, issueridcomposite: string, sysdt: date, systs: timestamp, Protocol\_Used: string]>

>>> result\_df\_SQL\_2.describe

<bound method DataFrame.describe of DataFrame[c\_BusinessDate: date, stcd: string, srcnm: string, c\_NetworkName: string, c\_NetworkURL: string, custnum: string, MarketCoverage: string, RejectRows: string, c\_IssuerId: string, IssuerId2: string, issueridcomposite: string, sysdt: date, systs: timestamp, curdt: date, curts: timestamp]>

>>> result\_df\_SQL\_3.describe

<bound method DataFrame.describe of DataFrame[d\_BusinessDate: date, stcd: string, srcnm: string, d\_NetworkName: string, d\_NetworkURL: string, custnum: string, MarketCoverage: string, RejectRows: string, d\_IssuerId: string, IssuerId2: string, issueridcomposite: string, sysdt: date, systs: timestamp, cleannetworkname: string]>

>>> result\_df\_SQL = result\_df\_SQL \

... .withColumnRenamed("a\_IssuerId", "IssuerId") \

... .withColumnRenamed("a\_NetworkName", "NetworkName") \

... .withColumnRenamed("a\_NetworkURL", "NetworkURL") \

... .withColumnRenamed("a\_BusinessDate", "BusinessDate")

>>> result\_df\_SQL\_1 = result\_df\_SQL\_1 \

... .withColumnRenamed("b\_IssuerId", "IssuerId") \

... .withColumnRenamed("b\_NetworkName", "NetworkName") \

... .withColumnRenamed("b\_NetworkURL", "NetworkURL") \

... .withColumnRenamed("b\_BusinessDate", "BusinessDate")

>>> result\_df\_SQL\_3 = result\_df\_SQL\_3 \

... .withColumnRenamed("d\_IssuerId", "IssuerId") \

... .withColumnRenamed("d\_NetworkName", "NetworkName") \

... .withColumnRenamed("d\_NetworkURL", "NetworkURL") \

... .withColumnRenamed("d\_BusinessDate", "BusinessDate")

>>> result\_df\_SQL\_2 = result\_df\_SQL\_2 \

... .withColumnRenamed("c\_IssuerId", "IssuerId") \

... .withColumnRenamed("c\_NetworkName", "NetworkName") \

... .withColumnRenamed("c\_NetworkURL", "NetworkURL") \

... .withColumnRenamed("c\_BusinessDate", "BusinessDate")

>>> df\_combined = result\_df\_SQL \

... .alias("a") \

... .join(result\_df\_SQL\_1.alias("b"), col("a.IssuerId") == col("b.IssuerId"), how="inner") \

... .join(result\_df\_SQL\_2.alias("c"), col("a.IssuerId") == col("c.IssuerId"), how="inner") \

... .join(result\_df\_SQL\_3.alias("d"), col("a.IssuerId") == col("d.IssuerId"), how="inner")

>>> df\_combined.columns()

Traceback (most recent call last):

File "<stdin>", line 1, in <module>

TypeError: 'list' object is not callable

>>> list=df\_combined.columns()

Traceback (most recent call last):

File "<stdin>", line 1, in <module>

TypeError: 'list' object is not callable

>>> list\_ele=df\_combined.columns()

Traceback (most recent call last):

File "<stdin>", line 1, in <module>

TypeError: 'list' object is not callable

>>> list\_ele=df\_combined.columns

>>> print(list\_Ele)

Traceback (most recent call last):

File "<stdin>", line 1, in <module>

NameError: name 'list\_Ele' is not defined

>>> print(list\_ele)

['BusinessDate', 'stcd', 'srcnm', 'NetworkName', 'NetworkURL', 'custnum', 'MarketCoverage', 'RejectRows', 'IssuerId', 'IssuerId2', 'issueridcomposite', 'sysdt', 'systs', 'month', 'year', 'BusinessDate', 'stcd', 'srcnm', 'NetworkName', 'NetworkURL', 'custnum', 'MarketCoverage', 'RejectRows', 'IssuerId', 'IssuerId2', 'issueridcomposite', 'sysdt', 'systs', 'Protocol\_Used', 'BusinessDate', 'stcd', 'srcnm', 'NetworkName', 'NetworkURL', 'custnum', 'MarketCoverage', 'RejectRows', 'IssuerId', 'IssuerId2', 'issueridcomposite', 'sysdt', 'systs', 'curdt', 'curts', 'BusinessDate', 'stcd', 'srcnm', 'NetworkName', 'NetworkURL', 'custnum', 'MarketCoverage', 'RejectRows', 'IssuerId', 'IssuerId2', 'issueridcomposite', 'sysdt', 'systs', 'cleannetworkname']

>>> df\_combined.write.parquet("hdfs:///user/hduser/Hackathon/PARQUET\_DATA")

Traceback (most recent call last):

File "<stdin>", line 1, in <module>

File "/usr/local/spark/python/pyspark/sql/readwriter.py", line 885, in parquet

self.\_jwrite.parquet(path)

File "/usr/local/spark/python/lib/py4j-0.10.9.3-src.zip/py4j/java\_gateway.py", line 1322, in \_\_call\_\_

File "/usr/local/spark/python/pyspark/sql/utils.py", line 117, in deco

raise converted from None

pyspark.sql.utils.AnalysisException: Found duplicate column(s) when inserting into hdfs://localhost:54310/user/hduser/Hackathon/PARQUET\_DATA: `businessdate`, `custnum`, `issuerid2`, `issuerid`, `issueridcomposite`, `marketcoverage`, `networkname`, `networkurl`, `rejectrows`, `srcnm`, `stcd`, `sysdt`, `systs`

>>> result\_df=df\_combined.drop("")

>>> result\_df\_SQL\_renamed = result\_df\_SQL \

... .withColumnRenamed("a\_BusinessDate", "a\_BusinessDate") \

... .withColumnRenamed("stcd", "a\_stcd") \

... .withColumnRenamed("srcnm", "a\_srcnm") \

... .withColumnRenamed("a\_NetworkName", "a\_NetworkName") \

... .withColumnRenamed("a\_NetworkURL", "a\_NetworkURL") \

... .withColumnRenamed("custnum", "a\_custnum") \

... .withColumnRenamed("MarketCoverage", "a\_MarketCoverage") \

... .withColumnRenamed("RejectRows", "a\_RejectRows") \

... .withColumnRenamed("a\_IssuerId", "a\_IssuerId") \

... .withColumnRenamed("IssuerId2", "a\_IssuerId2") \

... .withColumnRenamed("issueridcomposite", "a\_issueridcomposite") \

... .withColumnRenamed("sysdt", "a\_sysdt") \

... .withColumnRenamed("systs", "a\_systs") \

... .withColumnRenamed("month", "a\_month") \

... .withColumnRenamed("year", "a\_year")

>>> result\_df\_SQL\_1\_renamed = result\_df\_SQL\_1 \

... .withColumnRenamed("b\_BusinessDate", "b\_BusinessDate") \

... .withColumnRenamed("stcd", "b\_stcd") \

... .withColumnRenamed("srcnm", "b\_srcnm") \

... .withColumnRenamed("b\_NetworkName", "b\_NetworkName") \

... .withColumnRenamed("b\_NetworkURL", "b\_NetworkURL") \

... .withColumnRenamed("custnum", "b\_custnum") \

... .withColumnRenamed("MarketCoverage", "b\_MarketCoverage") \

... .withColumnRenamed("RejectRows", "b\_RejectRows") \

... .withColumnRenamed("b\_IssuerId", "b\_IssuerId") \

... .withColumnRenamed("IssuerId2", "b\_IssuerId2") \

... .withColumnRenamed("issueridcomposite", "b\_issueridcomposite") \

... .withColumnRenamed("sysdt", "b\_sysdt") \

... .withColumnRenamed("systs", "b\_systs") \

... .withColumnRenamed("Protocol\_Used", "b\_Protocol\_Used")

>>>

>>> result\_df\_SQL\_2\_renamed = result\_df\_SQL\_2 \

... .withColumnRenamed("c\_BusinessDate", "c\_BusinessDate") \

... .withColumnRenamed("stcd", "c\_stcd") \

... .withColumnRenamed("srcnm", "c\_srcnm") \

... .withColumnRenamed("c\_NetworkName", "c\_NetworkName") \

... .withColumnRenamed("c\_NetworkURL", "c\_NetworkURL") \

... .withColumnRenamed("custnum", "c\_custnum") \

... .withColumnRenamed("MarketCoverage", "c\_MarketCoverage") \

... .withColumnRenamed("RejectRows", "c\_RejectRows") \

... .withColumnRenamed("c\_IssuerId", "c\_IssuerId") \

... .withColumnRenamed("IssuerId2", "c\_IssuerId2") \

... .withColumnRenamed("issueridcomposite", "c\_issueridcomposite") \

... .withColumnRenamed("sysdt", "c\_sysdt") \

... .withColumnRenamed("systs", "c\_systs") \

... .withColumnRenamed("curdt", "c\_curdt") \

... .withColumnRenamed("curts", "c\_curts")

>>> result\_df\_SQL\_3\_renamed = result\_df\_SQL\_3 \

... .withColumnRenamed("d\_BusinessDate", "d\_BusinessDate") \

... .withColumnRenamed("stcd", "d\_stcd") \

... .withColumnRenamed("srcnm", "d\_srcnm") \

... .withColumnRenamed("d\_NetworkName", "d\_NetworkName") \

... .withColumnRenamed("d\_NetworkURL", "d\_NetworkURL") \

... .withColumnRenamed("custnum", "d\_custnum") \

... .withColumnRenamed("MarketCoverage", "d\_MarketCoverage") \

... .withColumnRenamed("RejectRows", "d\_RejectRows") \

... .withColumnRenamed("d\_IssuerId", "d\_IssuerId") \

... .withColumnRenamed("IssuerId2", "d\_IssuerId2") \

... .withColumnRenamed("issueridcomposite", "d\_issueridcomposite") \

... .withColumnRenamed("sysdt", "d\_sysdt") \

... .withColumnRenamed("systs", "d\_systs") \

... .withColumnRenamed("cleannetworkname", "d\_cleannetworkname")

>>> final\_joined\_df = (result\_df\_SQL\_renamed

... .join(result\_df\_SQL\_1\_renamed,

... (F.col("a\_BusinessDate") == F.col("b\_BusinessDate")) &

... (F.col("a\_stcd") == F.col("b\_stcd")) &

... (F.col("a\_srcnm") == F.col("b\_srcnm")) &

... (F.col("a\_custnum") == F.col("b\_custnum")),

... "left")

... .join(result\_df\_SQL\_2\_renamed,

... (F.col("a\_BusinessDate") == F.col("c\_BusinessDate")) &

... (F.col("a\_stcd") == F.col("c\_stcd")) &

... (F.col("a\_srcnm") == F.col("c\_srcnm")) &

... (F.col("a\_custnum") == F.col("c\_custnum")),

... "left")

... .join(result\_df\_SQL\_3\_renamed,

... (F.col("a\_BusinessDate") == F.col("d\_BusinessDate")) &

... (F.col("a\_stcd") == F.col("d\_stcd")) &

... (F.col("a\_srcnm") == F.col("d\_srcnm")) &

... (F.col("a\_custnum") == F.col("d\_custnum")),

... "left")

... )

Traceback (most recent call last):

File "<stdin>", line 5, in <module>

NameError: name 'F' is not defined

>>> from pyspark.sql import functions as F

>>> final\_joined\_df = (result\_df\_SQL\_renamed

... .join(result\_df\_SQL\_1\_renamed,

... (F.col("a\_BusinessDate") == F.col("b\_BusinessDate")) &

... (F.col("a\_stcd") == F.col("b\_stcd")) &

... (F.col("a\_srcnm") == F.col("b\_srcnm")) &

... (F.col("a\_custnum") == F.col("b\_custnum")),

... "left")

... .join(result\_df\_SQL\_2\_renamed,

... (F.col("a\_BusinessDate") == F.col("c\_BusinessDate")) &

... (F.col("a\_stcd") == F.col("c\_stcd")) &

... (F.col("a\_srcnm") == F.col("c\_srcnm")) &

... (F.col("a\_custnum") == F.col("c\_custnum")),

... "left")

... .join(result\_df\_SQL\_3\_renamed,

... (F.col("a\_BusinessDate") == F.col("d\_BusinessDate")) &

... (F.col("a\_stcd") == F.col("d\_stcd")) &

... (F.col("a\_srcnm") == F.col("d\_srcnm")) &

... (F.col("a\_custnum") == F.col("d\_custnum")),

... "left")

... )

Traceback (most recent call last):

File "<stdin>", line 7, in <module>

File "/usr/local/spark/python/pyspark/sql/dataframe.py", line 1355, in join

jdf = self.\_jdf.join(other.\_jdf, on, how)

File "/usr/local/spark/python/lib/py4j-0.10.9.3-src.zip/py4j/java\_gateway.py", line 1322, in \_\_call\_\_

File "/usr/local/spark/python/pyspark/sql/utils.py", line 117, in deco

raise converted from None

pyspark.sql.utils.AnalysisException: cannot resolve 'a\_BusinessDate' given input columns: [BusinessDate, BusinessDate, IssuerId, IssuerId, NetworkName, NetworkName, NetworkURL, NetworkURL, a\_IssuerId2, a\_MarketCoverage, a\_RejectRows, a\_custnum, a\_issueridcomposite, a\_month, a\_srcnm, a\_stcd, a\_sysdt, a\_systs, a\_year, b\_IssuerId2, b\_MarketCoverage, b\_Protocol\_Used, b\_RejectRows, b\_custnum, b\_issueridcomposite, b\_srcnm, b\_stcd, b\_sysdt, b\_systs];

'Join LeftOuter, (((('a\_BusinessDate = 'b\_BusinessDate) AND (a\_stcd#46162 = b\_stcd#46397)) AND (a\_srcnm#46178 = b\_srcnm#46412)) AND (a\_custnum#46224 = b\_custnum#46455))

:- Project [BusinessDate#44187, a\_stcd#46162, a\_srcnm#46178, NetworkName#44155, NetworkURL#44171, a\_custnum#46224, a\_MarketCoverage#46240, a\_RejectRows#46256, IssuerId#44139, a\_IssuerId2#46287, a\_issueridcomposite#46303, a\_sysdt#46319, a\_systs#46335, a\_month#46351, year#30929 AS a\_year#46367]

: +- Project [BusinessDate#44187, a\_stcd#46162, a\_srcnm#46178, NetworkName#44155, NetworkURL#44171, a\_custnum#46224, a\_MarketCoverage#46240, a\_RejectRows#46256, IssuerId#44139, a\_IssuerId2#46287, a\_issueridcomposite#46303, a\_sysdt#46319, a\_systs#46335, month#30928 AS a\_month#46351, year#30929]

: +- Project [BusinessDate#44187, a\_stcd#46162, a\_srcnm#46178, NetworkName#44155, NetworkURL#44171, a\_custnum#46224, a\_MarketCoverage#46240, a\_RejectRows#46256, IssuerId#44139, a\_IssuerId2#46287, a\_issueridcomposite#46303, a\_sysdt#46319, systs#9016 AS a\_systs#46335, month#30928, year#30929]

: +- Project [BusinessDate#44187, a\_stcd#46162, a\_srcnm#46178, NetworkName#44155, NetworkURL#44171, a\_custnum#46224, a\_MarketCoverage#46240, a\_RejectRows#46256, IssuerId#44139, a\_IssuerId2#46287, a\_issueridcomposite#46303, sysdt#9003 AS a\_sysdt#46319, systs#9016, month#30928, year#30929]

: +- Project [BusinessDate#44187, a\_stcd#46162, a\_srcnm#46178, NetworkName#44155, NetworkURL#44171, a\_custnum#46224, a\_MarketCoverage#46240, a\_RejectRows#46256, IssuerId#44139, a\_IssuerId2#46287, issueridcomposite#8901 AS a\_issueridcomposite#46303, sysdt#9003, systs#9016, month#30928, year#30929]

: +- Project [BusinessDate#44187, a\_stcd#46162, a\_srcnm#46178, NetworkName#44155, NetworkURL#44171, a\_custnum#46224, a\_MarketCoverage#46240, a\_RejectRows#46256, IssuerId#44139, IssuerId2#8325 AS a\_IssuerId2#46287, issueridcomposite#8901, sysdt#9003, systs#9016, month#30928, year#30929]

: +- Project [BusinessDate#44187, a\_stcd#46162, a\_srcnm#46178, NetworkName#44155, NetworkURL#44171, a\_custnum#46224, a\_MarketCoverage#46240, RejectRows#1386 AS a\_RejectRows#46256, IssuerId#44139, IssuerId2#8325, issueridcomposite#8901, sysdt#9003, systs#9016, month#30928, year#30929]

: +- Project [BusinessDate#44187, a\_stcd#46162, a\_srcnm#46178, NetworkName#44155, NetworkURL#44171, a\_custnum#46224, MarketCoverage#1384 AS a\_MarketCoverage#46240, RejectRows#1386, IssuerId#44139, IssuerId2#8325, issueridcomposite#8901, sysdt#9003, systs#9016, month#30928, year#30929]

: +- Project [BusinessDate#44187, a\_stcd#46162, a\_srcnm#46178, NetworkName#44155, NetworkURL#44171, custnum#1383 AS a\_custnum#46224, MarketCoverage#1384, RejectRows#1386, IssuerId#44139, IssuerId2#8325, issueridcomposite#8901, sysdt#9003, systs#9016, month#30928, year#30929]

: +- Project [BusinessDate#44187, a\_stcd#46162, srcnm#7747 AS a\_srcnm#46178, NetworkName#44155, NetworkURL#44171, custnum#1383, MarketCoverage#1384, RejectRows#1386, IssuerId#44139, IssuerId2#8325, issueridcomposite#8901, sysdt#9003, systs#9016, month#30928, year#30929]

: +- Project [BusinessDate#44187, stcd#7759 AS a\_stcd#46162, srcnm#7747, NetworkName#44155, NetworkURL#44171, custnum#1383, MarketCoverage#1384, RejectRows#1386, IssuerId#44139, IssuerId2#8325, issueridcomposite#8901, sysdt#9003, systs#9016, month#30928, year#30929]

: +- Project [a\_BusinessDate#43854 AS BusinessDate#44187, stcd#7759, srcnm#7747, NetworkName#44155, NetworkURL#44171, custnum#1383, MarketCoverage#1384, RejectRows#1386, IssuerId#44139, IssuerId2#8325, issueridcomposite#8901, sysdt#9003, systs#9016, month#30928, year#30929]

: +- Project [a\_BusinessDate#43854, stcd#7759, srcnm#7747, NetworkName#44155, a\_NetworkURL#43806 AS NetworkURL#44171, custnum#1383, MarketCoverage#1384, RejectRows#1386, IssuerId#44139, IssuerId2#8325, issueridcomposite#8901, sysdt#9003, systs#9016, month#30928, year#30929]

: +- Project [a\_BusinessDate#43854, stcd#7759, srcnm#7747, a\_NetworkName#43822 AS NetworkName#44155, a\_NetworkURL#43806, custnum#1383, MarketCoverage#1384, RejectRows#1386, IssuerId#44139, IssuerId2#8325, issueridcomposite#8901, sysdt#9003, systs#9016, month#30928, year#30929]

: +- Project [a\_BusinessDate#43854, stcd#7759, srcnm#7747, a\_NetworkName#43822, a\_NetworkURL#43806, custnum#1383, MarketCoverage#1384, RejectRows#1386, a\_IssuerId#43838 AS IssuerId#44139, IssuerId2#8325, issueridcomposite#8901, sysdt#9003, systs#9016, month#30928, year#30929]

: +- Project [BusinessDate#1378 AS a\_BusinessDate#43854, stcd#7759, srcnm#7747, a\_NetworkName#43822, a\_NetworkURL#43806, custnum#1383, MarketCoverage#1384, RejectRows#1386, a\_IssuerId#43838, IssuerId2#8325, issueridcomposite#8901, sysdt#9003, systs#9016, month#30928, year#30929]

: +- Project [BusinessDate#1378, stcd#7759, srcnm#7747, a\_NetworkName#43822, a\_NetworkURL#43806, custnum#1383, MarketCoverage#1384, RejectRows#1386, IssuerId#8312 AS a\_IssuerId#43838, IssuerId2#8325, issueridcomposite#8901, sysdt#9003, systs#9016, month#30928, year#30929]

: +- Project [BusinessDate#1378, stcd#7759, srcnm#7747, NetworkName#1381 AS a\_NetworkName#43822, a\_NetworkURL#43806, custnum#1383, MarketCoverage#1384, RejectRows#1386, IssuerId#8312, IssuerId2#8325, issueridcomposite#8901, sysdt#9003, systs#9016, month#30928, year#30929]

: +- Project [BusinessDate#1378, stcd#7759, srcnm#7747, NetworkName#1381, NetworkURL#1382 AS a\_NetworkURL#43806, custnum#1383, MarketCoverage#1384, RejectRows#1386, IssuerId#8312, IssuerId2#8325, issueridcomposite#8901, sysdt#9003, systs#9016, month#30928, year#30929]

: +- Project [BusinessDate#1378, stcd#7759, srcnm#7747, NetworkName#1381, NetworkURL#1382, custnum#1383, MarketCoverage#1384, RejectRows#1386, IssuerId#8312, IssuerId2#8325, issueridcomposite#8901, sysdt#9003, systs#9016, month(BusinessDate#1378) AS month#30928, year(BusinessDate#1378) AS year#30929]

: +- SubqueryAlias insureview

: +- View (`insureview`, [BusinessDate#1378,stcd#7759,srcnm#7747,NetworkName#1381,NetworkURL#1382,custnum#1383,MarketCoverage#1384,RejectRows#1386,IssuerId#8312,IssuerId2#8325,issueridcomposite#8901,sysdt#9003,systs#9016])

: +- Project [BusinessDate#1378, stcd#7759, srcnm#7747, NetworkName#1381, NetworkURL#1382, custnum#1383, MarketCoverage#1384, RejectRows#1386, IssuerId#8312, IssuerId2#8325, issueridcomposite#8901, sysdt#9003, current\_timestamp() AS systs#9016]

: +- Project [BusinessDate#1378, stcd#7759, srcnm#7747, NetworkName#1381, NetworkURL#1382, custnum#1383, MarketCoverage#1384, RejectRows#1386, IssuerId#8312, IssuerId2#8325, issueridcomposite#8901, current\_date(Some(Asia/Kolkata)) AS sysdt#9003]

: +- Project [BusinessDate#1378, stcd#7759, srcnm#7747, NetworkName#1381, NetworkURL#1382, custnum#1383, MarketCoverage#1384, RejectRows#1386, IssuerId#8312, IssuerId2#8325, issueridcomposite#8901]

: +- Project [BusinessDate#1378, stcd#7759, srcnm#7747, NetworkName#1381, NetworkURL#1382, custnum#1383, MarketCoverage#1384, DentalOnlyPlan#1385, RejectRows#1386, IssuerId#8312, IssuerId2#8325, concat(IssuerId#8312, IssuerId2#8325) AS issueridcomposite#8901]

: +- Project [BusinessDate#1378, stcd#7759, srcnm#7747, NetworkName#1381, NetworkURL#1382, custnum#1383, MarketCoverage#1384, DentalOnlyPlan#1385, RejectRows#1386, IssuerId#8312, IssuerId2#8325]

: +- Project [IssuerIDDD#8273, BusinessDate#1378, stcd#7759, srcnm#7747, NetworkName#1381, NetworkURL#1382, custnum#1383, MarketCoverage#1384, DentalOnlyPlan#1385, RejectRows#1386, IssuerId#8312, IssuerId2#8325]

: +- Project [IssuerIDD#8261, IssuerIDDD#8273, BusinessDate#1378, stcd#7759, srcnm#7747, NetworkName#1381, NetworkURL#1382, custnum#1383, MarketCoverage#1384, DentalOnlyPlan#1385, RejectRows#1386, IssuerId#8312, cast(IssuerIDDD#8273 as string) AS IssuerId2#8325]

: +- Project [IssuerIDD#8261, IssuerIDDD#8273, BusinessDate#1378, stcd#7759, srcnm#7747, NetworkName#1381, NetworkURL#1382, custnum#1383, MarketCoverage#1384, DentalOnlyPlan#1385, RejectRows#1386, cast(IssuerIDD#8261 as string) AS IssuerId#8312]

: +- Project [IssuerIDD#8261, IssuerId2#1377 AS IssuerIDDD#8273, BusinessDate#1378, stcd#7759, srcnm#7747, NetworkName#1381, NetworkURL#1382, custnum#1383, MarketCoverage#1384, DentalOnlyPlan#1385, RejectRows#1386]

: +- Project [IssuerId#1376 AS IssuerIDD#8261, IssuerId2#1377, BusinessDate#1378, stcd#7759, srcnm#7747, NetworkName#1381, NetworkURL#1382, custnum#1383, MarketCoverage#1384, DentalOnlyPlan#1385, RejectRows#1386]

: +- Project [IssuerId#1376, IssuerId2#1377, BusinessDate#1378, StateCode#1379 AS stcd#7759, srcnm#7747, NetworkName#1381, NetworkURL#1382, custnum#1383, MarketCoverage#1384, DentalOnlyPlan#1385, RejectRows#1386]

: +- Project [IssuerId#1376, IssuerId2#1377, BusinessDate#1378, StateCode#1379, SourceName#1380 AS srcnm#7747, NetworkName#1381, NetworkURL#1382, custnum#1383, MarketCoverage#1384, DentalOnlyPlan#1385, RejectRows#1386]

: +- Union false, false

: :- Relation [IssuerId#1376,IssuerId2#1377,BusinessDate#1378,StateCode#1379,SourceName#1380,NetworkName#1381,NetworkURL#1382,custnum#1383,MarketCoverage#1384,DentalOnlyPlan#1385,RejectRows#1386] csv

: +- Relation [IssuerId#491,IssuerId2#492,BusinessDate#493,StateCode#494,SourceName#495,NetworkName#496,NetworkURL#497,custnum#498,MarketCoverage#499,DentalOnlyPlan#500,RejectRows#501] csv

+- Project [BusinessDate#44248, b\_stcd#46397, b\_srcnm#46412, NetworkName#44218, NetworkURL#44233, b\_custnum#46455, b\_MarketCoverage#46470, b\_RejectRows#46485, IssuerId#44203, b\_IssuerId2#46514, b\_issueridcomposite#46529, b\_sysdt#46544, b\_systs#46559, Protocol\_Used#30945 AS b\_Protocol\_Used#46574]

+- Project [BusinessDate#44248, b\_stcd#46397, b\_srcnm#46412, NetworkName#44218, NetworkURL#44233, b\_custnum#46455, b\_MarketCoverage#46470, b\_RejectRows#46485, IssuerId#44203, b\_IssuerId2#46514, b\_issueridcomposite#46529, b\_sysdt#46544, systs#9016 AS b\_systs#46559, Protocol\_Used#30945]

+- Project [BusinessDate#44248, b\_stcd#46397, b\_srcnm#46412, NetworkName#44218, NetworkURL#44233, b\_custnum#46455, b\_MarketCoverage#46470, b\_RejectRows#46485, IssuerId#44203, b\_IssuerId2#46514, b\_issueridcomposite#46529, sysdt#9003 AS b\_sysdt#46544, systs#9016, Protocol\_Used#30945]

+- Project [BusinessDate#44248, b\_stcd#46397, b\_srcnm#46412, NetworkName#44218, NetworkURL#44233, b\_custnum#46455, b\_MarketCoverage#46470, b\_RejectRows#46485, IssuerId#44203, b\_IssuerId2#46514, issueridcomposite#8901 AS b\_issueridcomposite#46529, sysdt#9003, systs#9016, Protocol\_Used#30945]

+- Project [BusinessDate#44248, b\_stcd#46397, b\_srcnm#46412, NetworkName#44218, NetworkURL#44233, b\_custnum#46455, b\_MarketCoverage#46470, b\_RejectRows#46485, IssuerId#44203, IssuerId2#8325 AS b\_IssuerId2#46514, issueridcomposite#8901, sysdt#9003, systs#9016, Protocol\_Used#30945]

+- Project [BusinessDate#44248, b\_stcd#46397, b\_srcnm#46412, NetworkName#44218, NetworkURL#44233, b\_custnum#46455, b\_MarketCoverage#46470, RejectRows#47041 AS b\_RejectRows#46485, IssuerId#44203, IssuerId2#8325, issueridcomposite#8901, sysdt#9003, systs#9016, Protocol\_Used#30945]

+- Project [BusinessDate#44248, b\_stcd#46397, b\_srcnm#46412, NetworkName#44218, NetworkURL#44233, b\_custnum#46455, MarketCoverage#47039 AS b\_MarketCoverage#46470, RejectRows#47041, IssuerId#44203, IssuerId2#8325, issueridcomposite#8901, sysdt#9003, systs#9016, Protocol\_Used#30945]

+- Project [BusinessDate#44248, b\_stcd#46397, b\_srcnm#46412, NetworkName#44218, NetworkURL#44233, custnum#47038 AS b\_custnum#46455, MarketCoverage#47039, RejectRows#47041, IssuerId#44203, IssuerId2#8325, issueridcomposite#8901, sysdt#9003, systs#9016, Protocol\_Used#30945]

+- Project [BusinessDate#44248, b\_stcd#46397, srcnm#7747 AS b\_srcnm#46412, NetworkName#44218, NetworkURL#44233, custnum#47038, MarketCoverage#47039, RejectRows#47041, IssuerId#44203, IssuerId2#8325, issueridcomposite#8901, sysdt#9003, systs#9016, Protocol\_Used#30945]

+- Project [BusinessDate#44248, stcd#7759 AS b\_stcd#46397, srcnm#7747, NetworkName#44218, NetworkURL#44233, custnum#47038, MarketCoverage#47039, RejectRows#47041, IssuerId#44203, IssuerId2#8325, issueridcomposite#8901, sysdt#9003, systs#9016, Protocol\_Used#30945]

+- Project [b\_BusinessDate#43915 AS BusinessDate#44248, stcd#7759, srcnm#7747, NetworkName#44218, NetworkURL#44233, custnum#47038, MarketCoverage#47039, RejectRows#47041, IssuerId#44203, IssuerId2#8325, issueridcomposite#8901, sysdt#9003, systs#9016, Protocol\_Used#30945]

+- Project [b\_BusinessDate#43915, stcd#7759, srcnm#7747, NetworkName#44218, b\_NetworkURL#43870 AS NetworkURL#44233, custnum#47038, MarketCoverage#47039, RejectRows#47041, IssuerId#44203, IssuerId2#8325, issueridcomposite#8901, sysdt#9003, systs#9016, Protocol\_Used#30945]

+- Project [b\_BusinessDate#43915, stcd#7759, srcnm#7747, b\_NetworkName#43885 AS NetworkName#44218, b\_NetworkURL#43870, custnum#47038, MarketCoverage#47039, RejectRows#47041, IssuerId#44203, IssuerId2#8325, issueridcomposite#8901, sysdt#9003, systs#9016, Protocol\_Used#30945]

+- Project [b\_BusinessDate#43915, stcd#7759, srcnm#7747, b\_NetworkName#43885, b\_NetworkURL#43870, custnum#47038, MarketCoverage#47039, RejectRows#47041, b\_IssuerId#43900 AS IssuerId#44203, IssuerId2#8325, issueridcomposite#8901, sysdt#9003, systs#9016, Protocol\_Used#30945]

+- Project [BusinessDate#47033 AS b\_BusinessDate#43915, stcd#7759, srcnm#7747, b\_NetworkName#43885, b\_NetworkURL#43870, custnum#47038, MarketCoverage#47039, RejectRows#47041, b\_IssuerId#43900, IssuerId2#8325, issueridcomposite#8901, sysdt#9003, systs#9016, Protocol\_Used#30945]

+- Project [BusinessDate#47033, stcd#7759, srcnm#7747, b\_NetworkName#43885, b\_NetworkURL#43870, custnum#47038, MarketCoverage#47039, RejectRows#47041, IssuerId#8312 AS b\_IssuerId#43900, IssuerId2#8325, issueridcomposite#8901, sysdt#9003, systs#9016, Protocol\_Used#30945]

+- Project [BusinessDate#47033, stcd#7759, srcnm#7747, NetworkName#47036 AS b\_NetworkName#43885, b\_NetworkURL#43870, custnum#47038, MarketCoverage#47039, RejectRows#47041, IssuerId#8312, IssuerId2#8325, issueridcomposite#8901, sysdt#9003, systs#9016, Protocol\_Used#30945]

+- Project [BusinessDate#47033, stcd#7759, srcnm#7747, NetworkName#47036, NetworkURL#47037 AS b\_NetworkURL#43870, custnum#47038, MarketCoverage#47039, RejectRows#47041, IssuerId#8312, IssuerId2#8325, issueridcomposite#8901, sysdt#9003, systs#9016, Protocol\_Used#30945]

+- Project [BusinessDate#47033, stcd#7759, srcnm#7747, NetworkName#47036, NetworkURL#47037, custnum#47038, MarketCoverage#47039, RejectRows#47041, IssuerId#8312, IssuerId2#8325, issueridcomposite#8901, sysdt#9003, systs#9016, CASE WHEN NetworkURL#47037 LIKE https://% THEN http secured WHEN NetworkURL#47037 LIKE http://% THEN http non secured ELSE No Protocol END AS Protocol\_Used#30945]

+- SubqueryAlias insureview

+- View (`insureview`, [BusinessDate#47033,stcd#7759,srcnm#7747,NetworkName#47036,NetworkURL#47037,custnum#47038,MarketCoverage#47039,RejectRows#47041,IssuerId#8312,IssuerId2#8325,issueridcomposite#8901,sysdt#9003,systs#9016])

+- Project [BusinessDate#47033, stcd#7759, srcnm#7747, NetworkName#47036, NetworkURL#47037, custnum#47038, MarketCoverage#47039, RejectRows#47041, IssuerId#8312, IssuerId2#8325, issueridcomposite#8901, sysdt#9003, current\_timestamp() AS systs#9016]

+- Project [BusinessDate#47033, stcd#7759, srcnm#7747, NetworkName#47036, NetworkURL#47037, custnum#47038, MarketCoverage#47039, RejectRows#47041, IssuerId#8312, IssuerId2#8325, issueridcomposite#8901, current\_date(Some(Asia/Kolkata)) AS sysdt#9003]

+- Project [BusinessDate#47033, stcd#7759, srcnm#7747, NetworkName#47036, NetworkURL#47037, custnum#47038, MarketCoverage#47039, RejectRows#47041, IssuerId#8312, IssuerId2#8325, issueridcomposite#8901]

+- Project [BusinessDate#47033, stcd#7759, srcnm#7747, NetworkName#47036, NetworkURL#47037, custnum#47038, MarketCoverage#47039, DentalOnlyPlan#47040, RejectRows#47041, IssuerId#8312, IssuerId2#8325, concat(IssuerId#8312, IssuerId2#8325) AS issueridcomposite#8901]

+- Project [BusinessDate#47033, stcd#7759, srcnm#7747, NetworkName#47036, NetworkURL#47037, custnum#47038, MarketCoverage#47039, DentalOnlyPlan#47040, RejectRows#47041, IssuerId#8312, IssuerId2#8325]

+- Project [IssuerIDDD#8273, BusinessDate#47033, stcd#7759, srcnm#7747, NetworkName#47036, NetworkURL#47037, custnum#47038, MarketCoverage#47039, DentalOnlyPlan#47040, RejectRows#47041, IssuerId#8312, IssuerId2#8325]

+- Project [IssuerIDD#8261, IssuerIDDD#8273, BusinessDate#47033, stcd#7759, srcnm#7747, NetworkName#47036, NetworkURL#47037, custnum#47038, MarketCoverage#47039, DentalOnlyPlan#47040, RejectRows#47041, IssuerId#8312, cast(IssuerIDDD#8273 as string) AS IssuerId2#8325]

+- Project [IssuerIDD#8261, IssuerIDDD#8273, BusinessDate#47033, stcd#7759, srcnm#7747, NetworkName#47036, NetworkURL#47037, custnum#47038, MarketCoverage#47039, DentalOnlyPlan#47040, RejectRows#47041, cast(IssuerIDD#8261 as string) AS IssuerId#8312]

+- Project [IssuerIDD#8261, IssuerId2#47032 AS IssuerIDDD#8273, BusinessDate#47033, stcd#7759, srcnm#7747, NetworkName#47036, NetworkURL#47037, custnum#47038, MarketCoverage#47039, DentalOnlyPlan#47040, RejectRows#47041]

+- Project [IssuerId#47031 AS IssuerIDD#8261, IssuerId2#47032, BusinessDate#47033, stcd#7759, srcnm#7747, NetworkName#47036, NetworkURL#47037, custnum#47038, MarketCoverage#47039, DentalOnlyPlan#47040, RejectRows#47041]

+- Project [IssuerId#47031, IssuerId2#47032, BusinessDate#47033, StateCode#47034 AS stcd#7759, srcnm#7747, NetworkName#47036, NetworkURL#47037, custnum#47038, MarketCoverage#47039, DentalOnlyPlan#47040, RejectRows#47041]

+- Project [IssuerId#47031, IssuerId2#47032, BusinessDate#47033, StateCode#47034, SourceName#47035 AS srcnm#7747, NetworkName#47036, NetworkURL#47037, custnum#47038, MarketCoverage#47039, DentalOnlyPlan#47040, RejectRows#47041]

+- Union false, false

:- Relation [IssuerId#47031,IssuerId2#47032,BusinessDate#47033,StateCode#47034,SourceName#47035,NetworkName#47036,NetworkURL#47037,custnum#47038,MarketCoverage#47039,DentalOnlyPlan#47040,RejectRows#47041] csv

+- Relation [IssuerId#47042,IssuerId2#47043,BusinessDate#47044,StateCode#47045,SourceName#47046,NetworkName#47047,NetworkURL#47048,custnum#47049,MarketCoverage#47050,DentalOnlyPlan#47051,RejectRows#47052] csv

>>>

>>> # Perform inner join on all the renamed DataFrames

... df\_combined = (

... result\_df\_SQL\_renamed

... .join(result\_df\_SQL\_1\_renamed, on=['BusinessDate', 'stcd', 'srcnm', 'custnum', 'MarketCoverage', 'RejectRows', 'IssuerId', 'IssuerId2', 'issueridcomposite', 'sysdt', 'systs'], how='inner')

... .join(result\_df\_SQL\_2\_renamed, on=['BusinessDate', 'stcd', 'srcnm', 'custnum', 'MarketCoverage', 'RejectRows', 'IssuerId', 'IssuerId2', 'issueridcomposite', 'sysdt', 'systs'], how='inner')

... .join(result\_df\_SQL\_3\_renamed, on=['BusinessDate', 'stcd', 'srcnm', 'custnum', 'MarketCoverage', 'RejectRows', 'IssuerId', 'IssuerId2', 'issueridcomposite', 'sysdt', 'systs'], how='inner')

... )

Traceback (most recent call last):

File "<stdin>", line 4, in <module>

File "/usr/local/spark/python/pyspark/sql/dataframe.py", line 1355, in join

jdf = self.\_jdf.join(other.\_jdf, on, how)

File "/usr/local/spark/python/lib/py4j-0.10.9.3-src.zip/py4j/java\_gateway.py", line 1322, in \_\_call\_\_

File "/usr/local/spark/python/pyspark/sql/utils.py", line 117, in deco

raise converted from None

pyspark.sql.utils.AnalysisException: USING column `stcd` cannot be resolved on the left side of the join. The left-side columns: [BusinessDate, a\_stcd, a\_srcnm, NetworkName, NetworkURL, a\_custnum, a\_MarketCoverage, a\_RejectRows, IssuerId, a\_IssuerId2, a\_issueridcomposite, a\_sysdt, a\_systs, a\_month, a\_year]

>>>

>>> # Perform inner join on all the renamed DataFrames

... df\_combined = (

... result\_df\_SQL\_renamed

... .join(result\_df\_SQL\_1\_renamed,

... on=['a\_BusinessDate', 'a\_stcd', 'a\_srcnm', 'a\_custnum', 'a\_MarketCoverage', 'a\_RejectRows', 'a\_IssuerId', 'a\_IssuerId2', 'a\_issueridcomposite', 'a\_sysdt', 'a\_systs'],

... how='inner')

... .join(result\_df\_SQL\_2\_renamed,

... on=['a\_BusinessDate', 'a\_stcd', 'a\_srcnm', 'a\_custnum', 'a\_MarketCoverage', 'a\_RejectRows', 'a\_IssuerId', 'a\_IssuerId2', 'a\_issueridcomposite', 'a\_sysdt', 'a\_systs'],

... how='inner')

... .join(result\_df\_SQL\_3\_renamed,

... on=['a\_BusinessDate', 'a\_stcd', 'a\_srcnm', 'a\_custnum', 'a\_MarketCoverage', 'a\_RejectRows', 'a\_IssuerId', 'a\_IssuerId2', 'a\_issueridcomposite', 'a\_sysdt', 'a\_systs'],

... how='inner')

... )

Traceback (most recent call last):

File "<stdin>", line 6, in <module>

File "/usr/local/spark/python/pyspark/sql/dataframe.py", line 1355, in join

jdf = self.\_jdf.join(other.\_jdf, on, how)

File "/usr/local/spark/python/lib/py4j-0.10.9.3-src.zip/py4j/java\_gateway.py", line 1322, in \_\_call\_\_

File "/usr/local/spark/python/pyspark/sql/utils.py", line 117, in deco

raise converted from None

pyspark.sql.utils.AnalysisException: USING column `a\_BusinessDate` cannot be resolved on the left side of the join. The left-side columns: [BusinessDate, a\_stcd, a\_srcnm, NetworkName, NetworkURL, a\_custnum, a\_MarketCoverage, a\_RejectRows, IssuerId, a\_IssuerId2, a\_issueridcomposite, a\_sysdt, a\_systs, a\_month, a\_year]

>>> c

>>> print("result\_df\_SQL\_renamed columns:", result\_df\_SQL\_renamed.columns)

result\_df\_SQL\_renamed columns: ['BusinessDate', 'a\_stcd', 'a\_srcnm', 'NetworkName', 'NetworkURL', 'a\_custnum', 'a\_MarketCoverage', 'a\_RejectRows', 'IssuerId', 'a\_IssuerId2', 'a\_issueridcomposite', 'a\_sysdt', 'a\_systs', 'a\_month', 'a\_year']

>>> print("result\_df\_SQL\_1\_renamed columns:", result\_df\_SQL\_1\_renamed.columns)

result\_df\_SQL\_1\_renamed columns: ['BusinessDate', 'b\_stcd', 'b\_srcnm', 'NetworkName', 'NetworkURL', 'b\_custnum', 'b\_MarketCoverage', 'b\_RejectRows', 'IssuerId', 'b\_IssuerId2', 'b\_issueridcomposite', 'b\_sysdt', 'b\_systs', 'b\_Protocol\_Used']

>>> print("result\_df\_SQL\_2\_renamed columns:", result\_df\_SQL\_2\_renamed.columns)

result\_df\_SQL\_2\_renamed columns: ['BusinessDate', 'c\_stcd', 'c\_srcnm', 'NetworkName', 'NetworkURL', 'c\_custnum', 'c\_MarketCoverage', 'c\_RejectRows', 'IssuerId', 'c\_IssuerId2', 'c\_issueridcomposite', 'c\_sysdt', 'c\_systs', 'c\_curdt', 'c\_curts']

>>> print("result\_df\_SQL\_3\_renamed columns:", result\_df\_SQL\_3\_renamed.columns)

result\_df\_SQL\_3\_renamed columns: ['BusinessDate', 'd\_stcd', 'd\_srcnm', 'NetworkName', 'NetworkURL', 'd\_custnum', 'd\_MarketCoverage', 'd\_RejectRows', 'IssuerId', 'd\_IssuerId2', 'd\_issueridcomposite', 'd\_sysdt', 'd\_systs', 'd\_cleannetworkname']

>>>

>>> # Perform inner join on all the renamed DataFrames

... df\_combined = (

... result\_df\_SQL\_renamed

... .join(result\_df\_SQL\_1\_renamed,

... on=['BusinessDate', 'a\_stcd', 'a\_srcnm', 'a\_custnum', 'a\_MarketCoverage', 'a\_RejectRows', 'IssuerId', 'a\_IssuerId2', 'a\_issueridcomposite', 'a\_sysdt', 'a\_systs'],

... how='inner')

... .join(result\_df\_SQL\_2\_renamed,

... on=['BusinessDate', 'a\_stcd', 'a\_srcnm', 'a\_custnum', 'a\_MarketCoverage', 'a\_RejectRows', 'IssuerId', 'a\_IssuerId2', 'a\_issueridcomposite', 'a\_sysdt', 'a\_systs'],

... how='inner')

... .join(result\_df\_SQL\_3\_renamed,

... on=['BusinessDate', 'a\_stcd', 'a\_srcnm', 'a\_custnum', 'a\_MarketCoverage', 'a\_RejectRows', 'IssuerId', 'a\_IssuerId2', 'a\_issueridcomposite', 'a\_sysdt', 'a\_systs'],

... how='inner')

... )

Traceback (most recent call last):

File "<stdin>", line 6, in <module>

File "/usr/local/spark/python/pyspark/sql/dataframe.py", line 1355, in join

jdf = self.\_jdf.join(other.\_jdf, on, how)

File "/usr/local/spark/python/lib/py4j-0.10.9.3-src.zip/py4j/java\_gateway.py", line 1322, in \_\_call\_\_

File "/usr/local/spark/python/pyspark/sql/utils.py", line 117, in deco

raise converted from None

pyspark.sql.utils.AnalysisException: USING column `a\_stcd` cannot be resolved on the right side of the join. The right-side columns: [BusinessDate, b\_stcd, b\_srcnm, NetworkName, NetworkURL, b\_custnum, b\_MarketCoverage, b\_RejectRows, IssuerId, b\_IssuerId2, b\_issueridcomposite, b\_sysdt, b\_systs, b\_Protocol\_Used]

>>>

>>> df\_combined = result\_df\_SQL\_renamed \

... .join(result\_df\_SQL\_1\_renamed, on="a\_IssuerId2", how="inner") \

... .join(result\_df\_SQL\_2\_renamed, on="a\_IssuerId2", how="inner") \

... .join(result\_df\_SQL\_3\_renamed, on="a\_IssuerId2", how="inner")

Traceback (most recent call last):

File "<stdin>", line 2, in <module>

File "/usr/local/spark/python/pyspark/sql/dataframe.py", line 1355, in join

jdf = self.\_jdf.join(other.\_jdf, on, how)

File "/usr/local/spark/python/lib/py4j-0.10.9.3-src.zip/py4j/java\_gateway.py", line 1322, in \_\_call\_\_

File "/usr/local/spark/python/pyspark/sql/utils.py", line 117, in deco

raise converted from None

pyspark.sql.utils.AnalysisException: USING column `a\_IssuerId2` cannot be resolved on the right side of the join. The right-side columns: [BusinessDate, b\_stcd, b\_srcnm, NetworkName, NetworkURL, b\_custnum, b\_MarketCoverage, b\_RejectRows, IssuerId, b\_IssuerId2, b\_issueridcomposite, b\_sysdt, b\_systs, b\_Protocol\_Used]

>>> result\_df\_SQL\_renamed.describe

<bound method DataFrame.describe of DataFrame[BusinessDate: date, a\_stcd: string, a\_srcnm: string, NetworkName: string, NetworkURL: string, a\_custnum: string, a\_MarketCoverage: string, a\_RejectRows: string, IssuerId: string, a\_IssuerId2: string, a\_issueridcomposite: string, a\_sysdt: date, a\_systs: timestamp, a\_month: int, a\_year: int]>

>>> .withColumnRenamed("curdt", "c\_curdt") \spark.sql("select \*,sql\_function(NetworkName) as cleannetworkname from insureview")

>>> spark.sql("select \*,sql\_function(NetworkName) as cleannetworkname from insureview").show(1)

+------------+----+-----+-----------+--------------------+-------+--------------+----------+--------+---------+-----------------+----------+--------------------+----------------+

|BusinessDate|stcd|srcnm|NetworkName| NetworkURL|custnum|MarketCoverage|RejectRows|IssuerId|IssuerId2|issueridcomposite| sysdt| systs|cleannetworkname|

+------------+----+-----+-----------+--------------------+-------+--------------+----------+--------+---------+-----------------+----------+--------------------+----------------+

| 2019-10-01| AK| HIOS|ODS Premier|https://www.modah...| 13| null| null| 21989| 21989| 2198921989|2024-07-22|2024-07-22 14:57:...| ODS Premier|

+------------+----+-----+-----------+--------------------+-------+--------------+----------+--------+---------+-----------------+----------+--------------------+----------------+

only showing top 1 row

>>> result\_df=spark.sql("select \*,sql\_function(NetworkName) as cleannetworkname from insureview")

>>> result\_df.createOrReplaceTempView("insureview")

Traceback (most recent call last):

File "<stdin>", line 1, in <module>

File "/usr/local/spark/python/pyspark/sql/dataframe.py", line 188, in createOrReplaceTempView

self.\_jdf.createOrReplaceTempView(name)

File "/usr/local/spark/python/lib/py4j-0.10.9.3-src.zip/py4j/java\_gateway.py", line 1322, in \_\_call\_\_

File "/usr/local/spark/python/pyspark/sql/utils.py", line 117, in deco

raise converted from None

pyspark.sql.utils.AnalysisException: Recursive view `insureview` detected (cycle: `insureview` -> `insureview`)

>>> spark.catalog.dropTempView("insureview")

>>> result\_df.createOrReplaceTempView("insureview")

>>> result\_df=spark.sql("select \*,current\_date as curdt , current\_timestamp as curts from insureview")

>>> spark.catalog.dropTempView("insureview")

>>> result\_df.createOrReplaceTempView("insureview")

>>> result\_df=spark.sql("select \*,sql\_function(NetworkName) as cleannetworkname from insureview")

>>> spark.sql("""select \*,case when NetworkURL like 'https://%' then 'http secured' when NetworkURL like 'http://%' then 'http non secured' else 'No Protocol' end as Protocol\_Used from insureview""").show(1)

+------------+----+-----+-----------+--------------------+-------+--------------+----------+--------+---------+-----------------+----------+--------------------+----------------+----------+--------------------+-------------+

|BusinessDate|stcd|srcnm|NetworkName| NetworkURL|custnum|MarketCoverage|RejectRows|IssuerId|IssuerId2|issueridcomposite| sysdt| systs|cleannetworkname| curdt| curts|Protocol\_Used|

+------------+----+-----+-----------+--------------------+-------+--------------+----------+--------+---------+-----------------+----------+--------------------+----------------+----------+--------------------+-------------+

| 2019-10-01| AK| HIOS|ODS Premier|https://www.modah...| 13| null| null| 21989| 21989| 2198921989|2024-07-22|2024-07-22 15:09:...| ODS Premier|2024-07-22|2024-07-22 15:09:...| http secured|

+------------+----+-----+-----------+--------------------+-------+--------------+----------+--------+---------+-----------------+----------+--------------------+----------------+----------+--------------------+-------------+

only showing top 1 row

>>> spark.catalog.dropTempView("insureview")

>>> result\_df.createOrReplaceTempView("insureview")

>>> result\_df=spark.sql("""select \*,case when NetworkURL like 'https://%' then 'http secured' when NetworkURL like 'http://%' then 'http non secured' else 'No Protocol' end as Protocol\_Used from insureview""")

>>>

>>> result\_df.show(1)

+------------+----+-----+-----------+--------------------+-------+--------------+----------+--------+---------+-----------------+----------+--------------------+----------------+----------+--------------------+----------------+-------------+

|BusinessDate|stcd|srcnm|NetworkName| NetworkURL|custnum|MarketCoverage|RejectRows|IssuerId|IssuerId2|issueridcomposite| sysdt| systs|cleannetworkname| curdt| curts|cleannetworkname|Protocol\_Used|

+------------+----+-----+-----------+--------------------+-------+--------------+----------+--------+---------+-----------------+----------+--------------------+----------------+----------+--------------------+----------------+-------------+

| 2019-10-01| AK| HIOS|ODS Premier|https://www.modah...| 13| null| null| 21989| 21989| 2198921989|2024-07-22|2024-07-22 15:11:...| ODS Premier|2024-07-22|2024-07-22 15:11:...| ODS Premier| http secured|

+------------+----+-----+-----------+--------------------+-------+--------------+----------+--------+---------+-----------------+----------+--------------------+----------------+----------+--------------------+----------------+-------------+

only showing top 1 row

>>> result\_df.describe

>>> result\_df.describe

<bound method DataFrame.describe of DataFrame[BusinessDate: date, stcd: string, srcnm: string, NetworkName: string, NetworkURL: string, custnum: string, MarketCoverage: string, RejectRows: string, IssuerId: string, IssuerId2: string, issueridcomposite: string, sysdt: date, systs: timestamp, cleannetworkname: string, curdt: date, curts: timestamp, cleannetworkname: string, Protocol\_Used: string]>

>>> statefilter\_df.describe

<bound method DataFrame.describe of DataFrame[\_c0: string, \_c1: string]>

>>> spark.sql("select \* from stateview").show(2)

Traceback (most recent call last):

File "<stdin>", line 1, in <module>

File "/usr/local/spark/python/pyspark/sql/session.py", line 723, in sql

return DataFrame(self.\_jsparkSession.sql(sqlQuery), self.\_wrapped)

File "/usr/local/spark/python/lib/py4j-0.10.9.3-src.zip/py4j/java\_gateway.py", line 1322, in \_\_call\_\_

File "/usr/local/spark/python/pyspark/sql/utils.py", line 117, in deco

raise converted from None

pyspark.sql.utils.AnalysisException: Table or view not found: stateview; line 1 pos 14;

'Project [\*]

+- 'UnresolvedRelation [stateview], [], false

>>> statefilter\_df.show(1)

+---+----------+

|\_c0| \_c1|

+---+----------+

| NM|New Mexico|

+---+----------+

only showing top 1 row

>>> spark.sql("select \* from cust\_table").show(2)

+---+--------+-----+---+-------+

|\_c0| \_c1| \_c2|\_c3| \_c4|

+---+--------+-----+---+-------+

| 1|Kristina|Chung| 55| Pilot|

| 2| Paige| Chen| 74|Teacher|

+---+--------+-----+---+-------+

only showing top 2 rows

>>> spark.sql("select \* from state\_table").show(2)

+---+----------+

|\_c0| \_c1|

+---+----------+

| NM|New Mexico|

| NY| New York|

+---+----------+

only showing top 2 rows

>>> spark.sql("select \* from insureview").show(2)

+------------+----+-----+-----------+--------------------+-------+--------------+--------------------+--------+---------+-----------------+----------+--------------------+----------------+----------+--------------------+----------------+

|BusinessDate|stcd|srcnm|NetworkName| NetworkURL|custnum|MarketCoverage| RejectRows|IssuerId|IssuerId2|issueridcomposite| sysdt| systs|cleannetworkname| curdt| curts|cleannetworkname|

+------------+----+-----+-----------+--------------------+-------+--------------+--------------------+--------+---------+-----------------+----------+--------------------+----------------+----------+--------------------+----------------+

| 2019-10-01| AK| HIOS|ODS Premier|https://www.modah...| 13| null| null| 21989| 21989| 2198921989|2024-07-22|2024-07-22 15:31:...| ODS Premier|2024-07-22|2024-07-22 15:31:...| ODS Premier|

| 2019-10-01| AK| HIOS|ODS Premier| null| null| null|219,219,2019-10-0...| 219| 219| 219219|2024-07-22|2024-07-22 15:31:...| ODS Premier|2024-07-22|2024-07-22 15:31:...| ODS Premier|

+------------+----+-----+-----------+--------------------+-------+--------------+--------------------+--------+---------+-----------------+----------+--------------------+----------------+----------+--------------------+----------------+

only showing top 2 rows

>>> spark.sql("select \* from insureview as l join state\_table as r on l.stcd=r.\_c0 ").show(1)

+------------+----+-----+-----------+--------------------+-------+--------------+----------+--------+---------+-----------------+----------+--------------------+----------------+----------+--------------------+----------------+---+------+

|BusinessDate|stcd|srcnm|NetworkName| NetworkURL|custnum|MarketCoverage|RejectRows|IssuerId|IssuerId2|issueridcomposite| sysdt| systs|cleannetworkname| curdt| curts|cleannetworkname|\_c0| \_c1|

+------------+----+-----+-----------+--------------------+-------+--------------+----------+--------+---------+-----------------+----------+--------------------+----------------+----------+--------------------+----------------+---+------+

| 2019-10-01| AK| HIOS|ODS Premier|https://www.modah...| 13| null| null| 21989| 21989| 2198921989|2024-07-22|2024-07-22 15:35:...| ODS Premier|2024-07-22|2024-07-22 15:35:...| ODS Premier| AK|Alaska|

+------------+----+-----+-----------+--------------------+-------+--------------+----------+--------+---------+-----------------+----------+--------------------+----------------+----------+--------------------+----------------+---+------+

only showing top 1 row

>>> first\_join=spark.sql("select \* from insureview as l join state\_table as r on l.stcd=r.\_c0 ")

>>> spark.catalog.dropTempview("insureview")

Traceback (most recent call last):

File "<stdin>", line 1, in <module>

AttributeError: 'Catalog' object has no attribute 'dropTempview'

>>> spark.catelog.dropTempview("insureview")

Traceback (most recent call last):

File "<stdin>", line 1, in <module>

AttributeError: 'SparkSession' object has no attribute 'catelog'

>>> spark.catalog.dropTempview("insureview")

Traceback (most recent call last):

File "<stdin>", line 1, in <module>

AttributeError: 'Catalog' object has no attribute 'dropTempview'

>>> spark.catalog.dropTempView("insureview")

>>> first\_join.describe

<bound method DataFrame.describe of DataFrame[BusinessDate: date, stcd: string, srcnm: string, NetworkName: string, NetworkURL: string, custnum: string, MarketCoverage: string, RejectRows: string, IssuerId: string, IssuerId2: string, issueridcomposite: string, sysdt: date, systs: timestamp, cleannetworkname: string, curdt: date, curts: timestamp, cleannetworkname: string, \_c0: string, \_c1: string]>

>>> first\_join.createOrReplaceTempView("insureview")

>>> spark.sql("select \* from insureview").show(1)

+------------+----+-----+-----------+--------------------+-------+--------------+----------+--------+---------+-----------------+----------+--------------------+----------------+----------+--------------------+----------------+---+------+

|BusinessDate|stcd|srcnm|NetworkName| NetworkURL|custnum|MarketCoverage|RejectRows|IssuerId|IssuerId2|issueridcomposite| sysdt| systs|cleannetworkname| curdt| curts|cleannetworkname|\_c0| \_c1|

+------------+----+-----+-----------+--------------------+-------+--------------+----------+--------+---------+-----------------+----------+--------------------+----------------+----------+--------------------+----------------+---+------+

| 2019-10-01| AK| HIOS|ODS Premier|https://www.modah...| 13| null| null| 21989| 21989| 2198921989|2024-07-22|2024-07-22 15:43:...| ODS Premier|2024-07-22|2024-07-22 15:43:...| ODS Premier| AK|Alaska|

+------------+----+-----+-----------+--------------------+-------+--------------+----------+--------+---------+-----------------+----------+--------------------+----------------+----------+--------------------+----------------+---+------+

only showing top 1 row

>>> spark.sql("select \* from insureview as l join cust\_table as r on l.custnum=r.\_c3").show(1)

+------------+----+-----+-----------+--------------------+-------+--------------+----------+--------+---------+-----------------+----------+--------------------+----------------+----------+--------------------+----------------+---+-------+---+-----+-------+---+-----+

|BusinessDate|stcd|srcnm|NetworkName| NetworkURL|custnum|MarketCoverage|RejectRows|IssuerId|IssuerId2|issueridcomposite| sysdt| systs|cleannetworkname| curdt| curts|cleannetworkname|\_c0| \_c1|\_c0| \_c1| \_c2|\_c3| \_c4|

+------------+----+-----+-----------+--------------------+-------+--------------+----------+--------+---------+-----------------+----------+--------------------+----------------+----------+--------------------+----------------+---+-------+---+-----+-------+---+-----+

| 2019-10-01| AZ| HIOS| La Paz|http://uamarketpl...| 21| null| null| 88925| 88925| 8892588925|2024-07-22|2024-07-22 15:48:...| La Paz|2024-07-22|2024-07-22 15:48:...| La Paz| AZ|Arizona|779|Jason|Bowling| 21|Coach|

+------------+----+-----+-----------+--------------------+-------+--------------+----------+--------+---------+-----------------+----------+--------------------+----------------+----------+--------------------+----------------+---+-------+---+-----+-------+---+-----+

only showing top 1 row

>>> spark.sql("select \* from insureview as l join cust\_table as r on l.custnum=r.\_c3").show(4)

+------------+----+-----+-----------+--------------------+-------+--------------+----------+--------+---------+-----------------+----------+--------------------+----------------+----------+--------------------+----------------+---+-------+---+-------+-------+---+----------+

|BusinessDate|stcd|srcnm|NetworkName| NetworkURL|custnum|MarketCoverage|RejectRows|IssuerId|IssuerId2|issueridcomposite| sysdt| systs|cleannetworkname| curdt| curts|cleannetworkname|\_c0| \_c1|\_c0| \_c1| \_c2|\_c3| \_c4|

+------------+----+-----+-----------+--------------------+-------+--------------+----------+--------+---------+-----------------+----------+--------------------+----------------+----------+--------------------+----------------+---+-------+---+-------+-------+---+----------+

| 2019-10-01| AZ| HIOS| La Paz|http://uamarketpl...| 21| null| null| 88925| 88925| 8892588925|2024-07-22|2024-07-22 15:49:...| La Paz|2024-07-22|2024-07-22 15:49:...| La Paz| AZ|Arizona|779| Jason|Bowling| 21| Coach|

| 2019-10-01| AZ| HIOS| La Paz|http://uamarketpl...| 21| null| null| 88925| 88925| 8892588925|2024-07-22|2024-07-22 15:49:...| La Paz|2024-07-22|2024-07-22 15:49:...| La Paz| AZ|Arizona|765| Janice| Bowden| 21|Politician|

| 2019-10-01| AZ| HIOS| La Paz|http://uamarketpl...| 21| null| null| 88925| 88925| 8892588925|2024-07-22|2024-07-22 15:49:...| La Paz|2024-07-22|2024-07-22 15:49:...| La Paz| AZ|Arizona|465| Peter|Guthrie| 21| Nurse|

| 2019-10-01| AZ| HIOS| La Paz|http://uamarketpl...| 21| null| null| 88925| 88925| 8892588925|2024-07-22|2024-07-22 15:49:...| La Paz|2024-07-22|2024-07-22 15:49:...| La Paz| AZ|Arizona|451|Russell| Hess| 21| Architect|

+------------+----+-----+-----------+--------------------+-------+--------------+----------+--------+---------+-----------------+----------+--------------------+----------------+----------+--------------------+----------------+---+-------+---+-------+-------+---+----------+

only showing top 4 rows

>>> second\_join=spark.sql("select \* from insureview as l join cust\_table as r on l.custnum=r.\_c3")

>>> second\_join.count()

684

>>> second\_join.write.parquet("/user/hduser/Hackathon/PARQUET\_DATA")

Traceback (most recent call last):

File "<stdin>", line 1, in <module>

File "/usr/local/spark/python/pyspark/sql/readwriter.py", line 885, in parquet

self.\_jwrite.parquet(path)

File "/usr/local/spark/python/lib/py4j-0.10.9.3-src.zip/py4j/java\_gateway.py", line 1322, in \_\_call\_\_

File "/usr/local/spark/python/pyspark/sql/utils.py", line 117, in deco

raise converted from None

pyspark.sql.utils.AnalysisException: Found duplicate column(s) when inserting into hdfs://localhost:54310/user/hduser/Hackathon/PARQUET\_DATA: `\_c0`, `\_c1`, `cleannetworkname`

>>> second\_join.show()

+------------+----+-----+-----------+--------------------+-------+--------------+----------+--------+---------+-----------------+----------+--------------------+----------------+----------+--------------------+----------------+---+-------+---+--------+---------+---+--------------------+

|BusinessDate|stcd|srcnm|NetworkName| NetworkURL|custnum|MarketCoverage|RejectRows|IssuerId|IssuerId2|issueridcomposite| sysdt| systs|cleannetworkname| curdt| curts|cleannetworkname|\_c0| \_c1|\_c0| \_c1| \_c2|\_c3| \_c4|

+------------+----+-----+-----------+--------------------+-------+--------------+----------+--------+---------+-----------------+----------+--------------------+----------------+----------+--------------------+----------------+---+-------+---+--------+---------+---+--------------------+

| 2019-10-01| AZ| HIOS| La Paz|http://uamarketpl...| 21| null| null| 88925| 88925| 8892588925|2024-07-22|2024-07-22 15:57:...| La Paz|2024-07-22|2024-07-22 15:57:...| La Paz| AZ|Arizona|779| Jason| Bowling| 21| Coach|

| 2019-10-01| AZ| HIOS| La Paz|http://uamarketpl...| 21| null| null| 88925| 88925| 8892588925|2024-07-22|2024-07-22 15:57:...| La Paz|2024-07-22|2024-07-22 15:57:...| La Paz| AZ|Arizona|765| Janice| Bowden| 21| Politician|

| 2019-10-01| AZ| HIOS| La Paz|http://uamarketpl...| 21| null| null| 88925| 88925| 8892588925|2024-07-22|2024-07-22 15:57:...| La Paz|2024-07-22|2024-07-22 15:57:...| La Paz| AZ|Arizona|465| Peter| Guthrie| 21| Nurse|

| 2019-10-01| AZ| HIOS| La Paz|http://uamarketpl...| 21| null| null| 88925| 88925| 8892588925|2024-07-22|2024-07-22 15:57:...| La Paz|2024-07-22|2024-07-22 15:57:...| La Paz| AZ|Arizona|451| Russell| Hess| 21| Architect|

| 2019-10-01| AZ| HIOS| La Paz|http://uamarketpl...| 21| null| null| 88925| 88925| 8892588925|2024-07-22|2024-07-22 15:57:...| La Paz|2024-07-22|2024-07-22 15:57:...| La Paz| AZ|Arizona|349| Russell| Dalton| 21| Civil engineer|

| 2019-10-01| AZ| HIOS| La Paz|http://uamarketpl...| 21| null| null| 88925| 88925| 8892588925|2024-07-22|2024-07-22 15:57:...| La Paz|2024-07-22|2024-07-22 15:57:...| La Paz| AZ|Arizona|292| Donna| Rice| 21| Social worker|

| 2019-10-01| AZ| HIOS| La Paz|http://uamarketpl...| 21| null| null| 88925| 88925| 8892588925|2024-07-22|2024-07-22 15:57:...| La Paz|2024-07-22|2024-07-22 15:57:...| La Paz| AZ|Arizona|290| Gail|Whitehead| 21| Nurse|

| 2019-10-01| AZ| HIOS| La Paz|http://uamarketpl...| 21| null| null| 88925| 88925| 8892588925|2024-07-22|2024-07-22 15:57:...| La Paz|2024-07-22|2024-07-22 15:57:...| La Paz| AZ|Arizona|176| Kristin|Alexander| 21| Coach|

| 2019-10-01| AZ| HIOS| La Paz|http://uamarketpl...| 21| null| null| 88925| 88925| 8892588925|2024-07-22|2024-07-22 15:57:...| La Paz|2024-07-22|2024-07-22 15:57:...| La Paz| AZ|Arizona| 52| Shirley| Merritt| 21| Reporter|

| 2019-10-01| AZ| HIOS| Maricopa|http://uamarketpl...| 22| null| null| 88925| 88925| 8892588925|2024-07-22|2024-07-22 15:57:...| Maricopa|2024-07-22|2024-07-22 15:57:...| Maricopa| AZ|Arizona|545|Danielle| Smith| 22| Pharmacist|

| 2019-10-01| AZ| HIOS| Maricopa|http://uamarketpl...| 22| null| null| 88925| 88925| 8892588925|2024-07-22|2024-07-22 15:57:...| Maricopa|2024-07-22|2024-07-22 15:57:...| Maricopa| AZ|Arizona|472| Kristin| Drake| 22| Financial analyst|

| 2019-10-01| AZ| HIOS| Maricopa|http://uamarketpl...| 22| null| null| 88925| 88925| 8892588925|2024-07-22|2024-07-22 15:57:...| Maricopa|2024-07-22|2024-07-22 15:57:...| Maricopa| AZ|Arizona|454|Virginia| Turner| 22|Computer hardware...|

| 2019-10-01| AZ| HIOS| Maricopa|http://uamarketpl...| 22| null| null| 88925| 88925| 8892588925|2024-07-22|2024-07-22 15:57:...| Maricopa|2024-07-22|2024-07-22 15:57:...| Maricopa| AZ|Arizona|359| Leon| Bender| 22| Firefighter|

| 2019-10-01| AZ| HIOS| Maricopa|http://uamarketpl...| 22| null| null| 88925| 88925| 8892588925|2024-07-22|2024-07-22 15:57:...| Maricopa|2024-07-22|2024-07-22 15:57:...| Maricopa| AZ|Arizona|218| Brett| Snyder| 22| Doctor|

| 2019-10-01| AZ| HIOS| Maricopa|http://uamarketpl...| 22| null| null| 88925| 88925| 8892588925|2024-07-22|2024-07-22 15:57:...| Maricopa|2024-07-22|2024-07-22 15:57:...| Maricopa| AZ|Arizona|152| Vincent| Goldman| 22| Electrician|

| 2019-10-01| AZ| HIOS| Maricopa|http://uamarketpl...| 22| null| null| 88925| 88925| 8892588925|2024-07-22|2024-07-22 15:57:...| Maricopa|2024-07-22|2024-07-22 15:57:...| Maricopa| AZ|Arizona| 54| Judith| Cooper| 22| Economist|

| 2019-10-01| AZ| HIOS| Mohave|http://uamarketpl...| 23| null| null| 88925| 88925| 8892588925|2024-07-22|2024-07-22 15:57:...| Mohave|2024-07-22|2024-07-22 15:57:...| Mohave| AZ|Arizona|752| Ethel| Stevens| 23| Designer|

| 2019-10-01| AZ| HIOS| Mohave|http://uamarketpl...| 23| null| null| 88925| 88925| 8892588925|2024-07-22|2024-07-22 15:57:...| Mohave|2024-07-22|2024-07-22 15:57:...| Mohave| AZ|Arizona|655| Hannah| Wall| 23| Librarian|

| 2019-10-01| AZ| HIOS| Mohave|http://uamarketpl...| 23| null| null| 88925| 88925| 8892588925|2024-07-22|2024-07-22 15:57:...| Mohave|2024-07-22|2024-07-22 15:57:...| Mohave| AZ|Arizona|639|Florence|McFarland| 23| Electrical engineer|

| 2019-10-01| AZ| HIOS| Mohave|http://uamarketpl...| 23| null| null| 88925| 88925| 8892588925|2024-07-22|2024-07-22 15:57:...| Mohave|2024-07-22|2024-07-22 15:57:...| Mohave| AZ|Arizona|248| Allison| Choi| 23|Computer hardware...|

+------------+----+-----+-----------+--------------------+-------+--------------+----------+--------+---------+-----------------+----------+--------------------+----------------+----------+--------------------+----------------+---+-------+---+--------+---------+---+--------------------+

only showing top 20 rows

>>> df\_renamed = second\_join.withColumnRenamed("\_c0", "state\_code") .withColumnRenamed("\_c1", "City")

>>> df.describe

Traceback (most recent call last):

File "<stdin>", line 1, in <module>

NameError: name 'df' is not defined

>>> df\_renamed.describe

<bound method DataFrame.describe of DataFrame[BusinessDate: date, stcd: string, srcnm: string, NetworkName: string, NetworkURL: string, custnum: string, MarketCoverage: string, RejectRows: string, IssuerId: string, IssuerId2: string, issueridcomposite: string, sysdt: date, systs: timestamp, cleannetworkname: string, curdt: date, curts: timestamp, cleannetworkname: string, state\_code: string, City: string, state\_code: string, City: string, \_c2: string, \_c3: int, \_c4: string]>

>>> first\_join=spark.sql("select \* from insureview as l join state\_table as r on l.stcd=r.\_c0 ")

>>>

>>> df\_renamed = df.select(

... col("BusinessDate"),

... col("stcd"),

... col("srcnm"),

... col("NetworkName"),

... col("NetworkURL"),

... col("custnum"),

... col("MarketCoverage"),

... col("RejectRows"),

... col("IssuerId"),

... col("IssuerId2"),

... col("issueridcomposite"),

... col("sysdt"),

... col("systs"),

... col("cleannetworkname").alias("cleannetworkname\_1"),

... col("curdt"),

... col("curts"),

... col("cleannetworkname").alias("cleannetworkname\_2"),

... col("state\_code").alias("state\_code\_1"),

... col("City").alias("City\_1"),

... col("state\_code").alias("state\_code\_2"),

... col("City").alias("City\_2"),

... col("\_c2"),

... col("\_c3"),

... col("\_c4")

... )

Traceback (most recent call last):

File "<stdin>", line 1, in <module>

NameError: name 'df' is not defined

>>> df\_renamed = df\_renamed.select(

... col("BusinessDate"),

... col("stcd"),

... col("srcnm"),

... col("NetworkName"),

... col("NetworkURL"),

... col("custnum"),

... col("MarketCoverage"),

... col("RejectRows"),

... col("IssuerId"),

... col("IssuerId2"),

... col("issueridcomposite"),

... col("sysdt"),

... col("systs"),

... col("cleannetworkname").alias("cleannetworkname\_1"),

... col("curdt"),

... col("curts"),

... col("cleannetworkname").alias("cleannetworkname\_2"),

... col("state\_code").alias("state\_code\_1"),

... col("City").alias("City\_1"),

... col("state\_code").alias("state\_code\_2"),

... col("City").alias("City\_2"),

... col("\_c2"),

... col("\_c3"),

... col("\_c4")

... )

Traceback (most recent call last):

File "<stdin>", line 25, in <module>

File "/usr/local/spark/python/pyspark/sql/dataframe.py", line 1685, in select

jdf = self.\_jdf.select(self.\_jcols(\*cols))

File "/usr/local/spark/python/lib/py4j-0.10.9.3-src.zip/py4j/java\_gateway.py", line 1322, in \_\_call\_\_

File "/usr/local/spark/python/pyspark/sql/utils.py", line 117, in deco

raise converted from None

pyspark.sql.utils.AnalysisException: Reference 'cleannetworkname' is ambiguous, could be: l.cleannetworkname, l.cleannetworkname.

>>> df\_temp = df\_named\

... .withColumnRenamed("cleannetworkname", "cleannetworkname\_temp") \

... .withColumnRenamed("state\_code", "state\_code\_temp") \

... .withColumnRenamed("City", "City\_temp")

Traceback (most recent call last):

File "<stdin>", line 1, in <module>

NameError: name 'df\_named' is not defined

>>>

>>> df\_temp = df\_renamed\

... .withColumnRenamed("cleannetworkname", "cleannetworkname\_temp") \

... .withColumnRenamed("state\_code", "state\_code\_temp") \

... .withColumnRenamed("City", "City\_temp")

>>>

>>> df\_renamed = df\_temp.select(

... col("BusinessDate"),

... col("stcd"),

... col("srcnm"),

... col("NetworkName"),

... col("NetworkURL"),

... col("custnum"),

... col("MarketCoverage"),

... col("RejectRows"),

... col("IssuerId"),

... col("IssuerId2"),

... col("issueridcomposite"),

... col("sysdt"),

... col("systs"),

... col("cleannetworkname\_temp").alias("cleannetworkname\_1"),

... col("curdt"),

... col("curts"),

... col("cleannetworkname\_temp").alias("cleannetworkname\_2"),

... col("state\_code\_temp").alias("state\_code\_1"),

... col("City\_temp").alias("City\_1"),

... col("state\_code\_temp").alias("state\_code\_2"),

... col("City\_temp").alias("City\_2"),

... col("\_c2"),

... col("\_c3"),

... col("\_c4")

... )

Traceback (most recent call last):

File "<stdin>", line 25, in <module>

File "/usr/local/spark/python/pyspark/sql/dataframe.py", line 1685, in select

jdf = self.\_jdf.select(self.\_jcols(\*cols))

File "/usr/local/spark/python/lib/py4j-0.10.9.3-src.zip/py4j/java\_gateway.py", line 1322, in \_\_call\_\_

File "/usr/local/spark/python/pyspark/sql/utils.py", line 117, in deco

raise converted from None

pyspark.sql.utils.AnalysisException: Reference 'cleannetworkname\_temp' is ambiguous, could be: cleannetworkname\_temp, cleannetworkname\_temp.

>>> df\_renamed = df\_temp.select(

... col("BusinessDate"),

... col("stcd"),

... col("srcnm"),

... col("NetworkName"),

... col("NetworkURL"),

... col("custnum"),

... col("MarketCoverage"),

... col("RejectRows"),

... col("IssuerId"),

... col("IssuerId2"),

... col("issueridcomposite"),

... col("sysdt"),

... col("systs"),

... col("cleannetworkname\_temp").alias("cleannetworkname\_1"),

... col("curdt"),

... col("curts"),

... col("cleannetworkname\_temp").alias("cleannetworkname\_2"),

... col("state\_code\_temp").alias("state\_code\_1"),

... col("City\_temp").alias("City\_1"),

... col("state\_code\_temp").alias("state\_code\_2"),

... col("City\_temp").alias("City\_2"),

... col("\_c2"),

... col("\_c3"),

... col("\_c4")

... )

Traceback (most recent call last):

File "<stdin>", line 25, in <module>

File "/usr/local/spark/python/pyspark/sql/dataframe.py", line 1685, in select

jdf = self.\_jdf.select(self.\_jcols(\*cols))

File "/usr/local/spark/python/lib/py4j-0.10.9.3-src.zip/py4j/java\_gateway.py", line 1322, in \_\_call\_\_

File "/usr/local/spark/python/pyspark/sql/utils.py", line 117, in deco

raise converted from None

pyspark.sql.utils.AnalysisException: Reference 'cleannetworkname\_temp' is ambiguous, could be: cleannetworkname\_temp, cleannetworkname\_temp.

>>> df\_temp.describe

<bound method DataFrame.describe of DataFrame[BusinessDate: date, stcd: string, srcnm: string, NetworkName: string, NetworkURL: string, custnum: string, MarketCoverage: string, RejectRows: string, IssuerId: string, IssuerId2: string, issueridcomposite: string, sysdt: date, systs: timestamp, cleannetworkname\_temp: string, curdt: date, curts: timestamp, cleannetworkname\_temp: string, state\_code\_temp: string, City\_temp: string, state\_code\_temp: string, City\_temp: string, \_c2: string, \_c3: int, \_c4: string]>

>>> df\_temp.drop("cleannetworkname\_temp")

DataFrame[BusinessDate: date, stcd: string, srcnm: string, NetworkName: string, NetworkURL: string, custnum: string, MarketCoverage: string, RejectRows: string, IssuerId: string, IssuerId2: string, issueridcomposite: string, sysdt: date, systs: timestamp, curdt: date, curts: timestamp, state\_code\_temp: string, City\_temp: string, state\_code\_temp: string, City\_temp: string, \_c2: string, \_c3: int, \_c4: string]

>>> df\_temp=df\_temp.drop("cleannetworkname\_temp")

>>> df\_temp.describe

<bound method DataFrame.describe of DataFrame[BusinessDate: date, stcd: string, srcnm: string, NetworkName: string, NetworkURL: string, custnum: string, MarketCoverage: string, RejectRows: string, IssuerId: string, IssuerId2: string, issueridcomposite: string, sysdt: date, systs: timestamp, curdt: date, curts: timestamp, state\_code\_temp: string, City\_temp: string, state\_code\_temp: string, City\_temp: string, \_c2: string, \_c3: int, \_c4: string]>

>>> df\_temp.write.parquet("hdfs:///user/hduser/Hackathon/PARQUET\_DATA")

Traceback (most recent call last):

File "<stdin>", line 1, in <module>

File "/usr/local/spark/python/pyspark/sql/readwriter.py", line 885, in parquet

self.\_jwrite.parquet(path)

File "/usr/local/spark/python/lib/py4j-0.10.9.3-src.zip/py4j/java\_gateway.py", line 1322, in \_\_call\_\_

File "/usr/local/spark/python/pyspark/sql/utils.py", line 117, in deco

raise converted from None

pyspark.sql.utils.AnalysisException: Found duplicate column(s) when inserting into hdfs://localhost:54310/user/hduser/Hackathon/PARQUET\_DATA: `city\_temp`, `state\_code\_temp`

>>> df\_temp=df\_temp.drop("city\_temp","state\_code\_temp")

>>> df\_temp.write.parquet("hdfs:///user/hduser/Hackathon/PARQUET\_DATA")

>>>

>>> WITH CountData AS (

File "<stdin>", line 1

WITH CountData AS (

^

SyntaxError: invalid syntax

>>> SELECT

File "<stdin>", line 1

SELECT

^

IndentationError: unexpected indent

>>> statedesc,

File "<stdin>", line 1

statedesc,

^

IndentationError: unexpected indent

>>> protocol,

File "<stdin>", line 1

protocol,

^

IndentationError: unexpected indent

>>> profession,

File "<stdin>", line 1

profession,

^

IndentationError: unexpected indent

>>> AVG(age) AS avg\_age, -- Calculating average age

File "<stdin>", line 1

AVG(age) AS avg\_age, -- Calculating average age

^

IndentationError: unexpected indent

>>> COUNT(\*) AS profession\_count, -- Counting number of professions

File "<stdin>", line 1

COUNT(\*) AS profession\_count, -- Counting number of professions

^

IndentationError: unexpected indent

>>> ROW\_NUMBER() OVER (PARTITION BY statedesc, protocol ORDER BY COUNT(\*) DESC) AS seqno -- Generating sequence number

File "<stdin>", line 1

ROW\_NUMBER() OVER (PARTITION BY statedesc, protocol ORDER BY COUNT(\*) DESC) AS seqno -- Generating sequence number

^

IndentationError: unexpected indent

>>> FROM your\_table

File "<stdin>", line 1

FROM your\_table

^

IndentationError: unexpected indent

>>> GROUP BY statedesc, protocol, profession

File "<stdin>", line 1

GROUP BY statedesc, protocol, profession

^

IndentationError: unexpected indent

>>> ),

File "<stdin>", line 1

),

^

SyntaxError: invalid syntax

>>> SecondHighestCount AS (

File "<stdin>", line 1

SecondHighestCount AS (

^

SyntaxError: invalid syntax

>>> SELECT

File "<stdin>", line 1

SELECT

^

IndentationError: unexpected indent

>>> statedesc,

File "<stdin>", line 1

statedesc,

^

IndentationError: unexpected indent

>>> protocol,

File "<stdin>", line 1

protocol,

^

IndentationError: unexpected indent

>>> profession,

File "<stdin>", line 1

profession,

^

IndentationError: unexpected indent

>>> avg\_age,

File "<stdin>", line 1

avg\_age,

^

IndentationError: unexpected indent

>>> profession\_count,

File "<stdin>", line 1

profession\_count,

^

IndentationError: unexpected indent

>>> seqno

File "<stdin>", line 1

seqno

^

IndentationError: unexpected indent

>>> FROM CountData

File "<stdin>", line 1

FROM CountData

^

IndentationError: unexpected indent

>>> WHERE seqno = 2 -- Filter to get the profession with the second highest count

File "<stdin>", line 1

WHERE seqno = 2 -- Filter to get the profession with the second highest count

^

IndentationError: unexpected indent

>>> )

File "<stdin>", line 1

)

^

SyntaxError: invalid syntax

>>> SELECT

Traceback (most recent call last):

File "<stdin>", line 1, in <module>

NameError: name 'SELECT' is not defined

>>> statedesc,

File "<stdin>", line 1

statedesc,

^

IndentationError: unexpected indent

>>> protocol,

File "<stdin>", line 1

protocol,

^

IndentationError: unexpected indent

>>> profession,

File "<stdin>", line 1

profession,

^

IndentationError: unexpected indent

>>> avg\_age,

File "<stdin>", line 1

avg\_age,

^

IndentationError: unexpected indent

>>> profession\_count

File "<stdin>", line 1

profession\_count

^

IndentationError: unexpected indent

>>> FROM SecondHighestCount

File "<stdin>", line 1

FROM SecondHighestCount

^

SyntaxError: invalid syntax

>>> ORDER BY statedesc, protocol, profession\_count DESC;

File "<stdin>", line 1

ORDER BY statedesc, protocol, profession\_count DESC;

^

SyntaxError: invalid syntax

>>>

>>> WITH CountData AS (

File "<stdin>", line 1

WITH CountData AS (

^

SyntaxError: invalid syntax

>>> SELECT

File "<stdin>", line 1

SELECT

^

IndentationError: unexpected indent

>>> statedesc,

File "<stdin>", line 1

statedesc,

^

IndentationError: unexpected indent

>>> protocol,

File "<stdin>", line 1

protocol,

^

IndentationError: unexpected indent

>>> profession,

File "<stdin>", line 1

profession,

^

IndentationError: unexpected indent

>>> AVG(age) AS avg\_age, -- Calculate average age

File "<stdin>", line 1

AVG(age) AS avg\_age, -- Calculate average age

^

IndentationError: unexpected indent

>>> COUNT(\*) AS profession\_count, -- Count occurrences of each profession

File "<stdin>", line 1

COUNT(\*) AS profession\_count, -- Count occurrences of each profession

^

IndentationError: unexpected indent

>>> ROW\_NUMBER() OVER (PARTITION BY statedesc, protocol ORDER BY COUNT(\*) DESC) AS seqno -- Generate sequence number

File "<stdin>", line 1

ROW\_NUMBER() OVER (PARTITION BY statedesc, protocol ORDER BY COUNT(\*) DESC) AS seqno -- Generate sequence number

^

IndentationError: unexpected indent

>>> FROM your\_table

File "<stdin>", line 1

FROM your\_table

^

IndentationError: unexpected indent

>>> GROUP BY statedesc, protocol, profession

File "<stdin>", line 1

GROUP BY statedesc, protocol, profession

^

IndentationError: unexpected indent

>>> ),

File "<stdin>", line 1

),

^

SyntaxError: invalid syntax

>>> SecondHighestCount AS (

File "<stdin>", line 1

SecondHighestCount AS (

^

SyntaxError: invalid syntax

>>> SELECT

File "<stdin>", line 1

SELECT

^

IndentationError: unexpected indent

>>> statedesc,

File "<stdin>", line 1

statedesc,

^

IndentationError: unexpected indent

>>> protocol,

File "<stdin>", line 1

protocol,

^

IndentationError: unexpected indent

>>> profession,

File "<stdin>", line 1

profession,

^

IndentationError: unexpected indent

>>> avg\_age,

File "<stdin>", line 1

avg\_age,

^

IndentationError: unexpected indent

>>> profession\_count

File "<stdin>", line 1

profession\_count

^

IndentationError: unexpected indent

>>> FROM CountData

File "<stdin>", line 1

FROM CountData

^

IndentationError: unexpected indent

>>> WHERE seqno = 2 -- Filter to get the profession with the second highest count

File "<stdin>", line 1

WHERE seqno = 2 -- Filter to get the profession with the second highest count

^

IndentationError: unexpected indent

>>> )query = """

File "<stdin>", line 1

)query = """

^

SyntaxError: invalid syntax

>>> WITH CountData AS (

File "<stdin>", line 1

WITH CountData AS (

^

SyntaxError: invalid syntax

>>> SELECT

File "<stdin>", line 1

SELECT

^

IndentationError: unexpected indent

>>> statedesc,

File "<stdin>", line 1

statedesc,

^

IndentationError: unexpected indent

>>> protocol,

File "<stdin>", line 1

protocol,

^

IndentationError: unexpected indent

>>> profession,

File "<stdin>", line 1

profession,

^

IndentationError: unexpected indent

>>> AVG(age) AS avg\_age,

File "<stdin>", line 1

AVG(age) AS avg\_age,

^

IndentationError: unexpected indent

>>> COUNT(\*) AS profession\_count,

File "<stdin>", line 1

COUNT(\*) AS profession\_count,

^

IndentationError: unexpected indent

>>> ROW\_NUMBER() OVER (PARTITION BY statedesc, protocol ORDER BY COUNT(\*) DESC) AS seqno

File "<stdin>", line 1

ROW\_NUMBER() OVER (PARTITION BY statedesc, protocol ORDER BY COUNT(\*) DESC) AS seqno

^

IndentationError: unexpected indent

>>> FROM your\_table

File "<stdin>", line 1

FROM your\_table

^

IndentationError: unexpected indent

>>> GROUP BY statedesc, protocol, profession

File "<stdin>", line 1

GROUP BY statedesc, protocol, profession

^

IndentationError: unexpected indent

>>> ),

File "<stdin>", line 1

),

^

SyntaxError: invalid syntax

>>> SecondHighestCount AS (

File "<stdin>", line 1

SecondHighestCount AS (

^

SyntaxError: invalid syntax

>>> SELECT

File "<stdin>", line 1

SELECT

^

IndentationError: unexpected indent

>>> statedesc,

File "<stdin>", line 1

statedesc,

^

IndentationError: unexpected indent

>>> protocol,

File "<stdin>", line 1

protocol,

^

IndentationError: unexpected indent

>>> profession,

File "<stdin>", line 1

profession,

^

IndentationError: unexpected indent

>>> avg\_age,

File "<stdin>", line 1

avg\_age,

^

IndentationError: unexpected indent

>>> profession\_count

File "<stdin>", line 1

profession\_count

^

IndentationError: unexpected indent

>>> FROM CountData

File "<stdin>", line 1

FROM CountData

^

IndentationError: unexpected indent

>>> WHERE seqno = 2

File "<stdin>", line 1

WHERE seqno = 2

^

IndentationError: unexpected indent

>>> )

File "<stdin>", line 1

)

^

SyntaxError: invalid syntax

>>> SELECT

Traceback (most recent call last):

File "<stdin>", line 1, in <module>

NameError: name 'SELECT' is not defined

>>> statedesc,

File "<stdin>", line 1

statedesc,

^

IndentationError: unexpected indent

>>> protocol,

File "<stdin>", line 1

protocol,

^

IndentationError: unexpected indent

>>> profession,

File "<stdin>", line 1

profession,

^

IndentationError: unexpected indent

>>> avg\_age,

File "<stdin>", line 1

avg\_age,

^

IndentationError: unexpected indent

>>> profession\_count

File "<stdin>", line 1

profession\_count

^

IndentationError: unexpected indent

>>> FROM SecondHighestCount

File "<stdin>", line 1

FROM SecondHighestCount

^

SyntaxError: invalid syntax

>>> ORDER BY statedesc, protocol;

File "<stdin>", line 1

ORDER BY statedesc, protocol;

^

SyntaxError: invalid syntax

>>> """\

...

...

...

...

... ;

...

File "<stdin>", line 5

;

^

SyntaxError: EOF while scanning triple-quoted string literal

>>>

>>> query = """

... WITH CountData AS (

... SELECT

... statedesc,

... protocol,

... profession,

... AVG(age) AS avg\_age,

... COUNT(\*) AS profession\_count,

... ROW\_NUMBER() OVER (PARTITION BY statedesc, protocol ORDER BY COUNT(\*) DESC) AS seqno

... FROM your\_table

... GROUP BY statedesc, protocol, profession

... ),

... SecondHighestCount AS (

... SELECT

... statedesc,

... protocol,

... profession,

... avg\_age,

... profession\_count

... FROM CountData

... WHERE seqno = 2

... )

... SELECT

... statedesc,

... protocol,

... profession,

... avg\_age,

... profession\_count

... FROM SecondHighestCount

... ORDER BY statedesc, protocol;

... """

>>>

>>> second\_join.describe

<bound method DataFrame.describe of DataFrame[BusinessDate: date, stcd: string, srcnm: string, NetworkName: string, NetworkURL: string, custnum: string, MarketCoverage: string, RejectRows: string, IssuerId: string, IssuerId2: string, issueridcomposite: string, sysdt: date, systs: timestamp, cleannetworkname: string, curdt: date, curts: timestamp, cleannetworkname: string, \_c0: string, \_c1: string, \_c0: string, \_c1: string, \_c2: string, \_c3: int, \_c4: string]>

>>> second\_join.show(2)

+------------+----+-----+-----------+--------------------+-------+--------------+----------+--------+---------+-----------------+----------+--------------------+----------------+----------+--------------------+----------------+---+-------+---+------+-------+---+----------+

|BusinessDate|stcd|srcnm|NetworkName| NetworkURL|custnum|MarketCoverage|RejectRows|IssuerId|IssuerId2|issueridcomposite| sysdt| systs|cleannetworkname| curdt| curts|cleannetworkname|\_c0| \_c1|\_c0| \_c1| \_c2|\_c3| \_c4|

+------------+----+-----+-----------+--------------------+-------+--------------+----------+--------+---------+-----------------+----------+--------------------+----------------+----------+--------------------+----------------+---+-------+---+------+-------+---+----------+

| 2019-10-01| AZ| HIOS| La Paz|http://uamarketpl...| 21| null| null| 88925| 88925| 8892588925|2024-07-22|2024-07-22 16:28:...| La Paz|2024-07-22|2024-07-22 16:28:...| La Paz| AZ|Arizona|779| Jason|Bowling| 21| Coach|

| 2019-10-01| AZ| HIOS| La Paz|http://uamarketpl...| 21| null| null| 88925| 88925| 8892588925|2024-07-22|2024-07-22 16:28:...| La Paz|2024-07-22|2024-07-22 16:28:...| La Paz| AZ|Arizona|765|Janice| Bowden| 21|Politician|

+------------+----+-----+-----------+--------------------+-------+--------------+----------+--------+---------+-----------------+----------+--------------------+----------------+----------+--------------------+----------------+---+-------+---+------+-------+---+----------+

only showing top 2 rows

>>> second\_join.createOrReplaceView("view1")

Traceback (most recent call last):

File "<stdin>", line 1, in <module>

File "/usr/local/spark/python/pyspark/sql/dataframe.py", line 1660, in \_\_getattr\_\_

"'%s' object has no attribute '%s'" % (self.\_\_class\_\_.\_\_name\_\_, name))

AttributeError: 'DataFrame' object has no attribute 'createOrReplaceView'

>>> second\_join.createOrReplaceTempView("view1")

>>> result\_df.show(2)

+------------+----+-----+-----------+--------------------+-------+--------------+--------------------+--------+---------+-----------------+----------+--------------------+----------------+----------+--------------------+----------------+-------------+

|BusinessDate|stcd|srcnm|NetworkName| NetworkURL|custnum|MarketCoverage| RejectRows|IssuerId|IssuerId2|issueridcomposite| sysdt| systs|cleannetworkname| curdt| curts|cleannetworkname|Protocol\_Used|

+------------+----+-----+-----------+--------------------+-------+--------------+--------------------+--------+---------+-----------------+----------+--------------------+----------------+----------+--------------------+----------------+-------------+

| 2019-10-01| AK| HIOS|ODS Premier|https://www.modah...| 13| null| null| 21989| 21989| 2198921989|2024-07-22|2024-07-22 16:33:...| ODS Premier|2024-07-22|2024-07-22 16:33:...| ODS Premier| http secured|

| 2019-10-01| AK| HIOS|ODS Premier| null| null| null|219,219,2019-10-0...| 219| 219| 219219|2024-07-22|2024-07-22 16:33:...| ODS Premier|2024-07-22|2024-07-22 16:33:...| ODS Premier| No Protocol|

+------------+----+-----+-----------+--------------------+-------+--------------+--------------------+--------+---------+-----------------+----------+--------------------+----------------+----------+--------------------+----------------+-------------+

only showing top 2 rows

>>> result\_df.createOrReplaceTempView("view2")

>>> spark.sql("select l.\*,r.Protocol\_Used from view1 as l and view2 as r ").show()

Traceback (most recent call last):

File "<stdin>", line 1, in <module>

File "/usr/local/spark/python/pyspark/sql/session.py", line 723, in sql

return DataFrame(self.\_jsparkSession.sql(sqlQuery), self.\_wrapped)

File "/usr/local/spark/python/lib/py4j-0.10.9.3-src.zip/py4j/java\_gateway.py", line 1322, in \_\_call\_\_

File "/usr/local/spark/python/pyspark/sql/utils.py", line 117, in deco

raise converted from None

pyspark.sql.utils.ParseException:

mismatched input 'and' expecting {<EOF>, ';'}(line 1, pos 43)

== SQL ==

select l.\*,r.Protocol\_Used from view1 as l and view2 as r

-------------------------------------------^^^

>>> spark.sql("select l.\*,r.Protocol\_Used from view1 as l,view2 as r ").show()

+------------+----+-----+-----------+--------------------+-------+--------------+----------+--------+---------+-----------------+----------+--------------------+----------------+----------+--------------------+----------------+---+-------+---+-----+-------+---+-----+----------------+

|BusinessDate|stcd|srcnm|NetworkName| NetworkURL|custnum|MarketCoverage|RejectRows|IssuerId|IssuerId2|issueridcomposite| sysdt| systs|cleannetworkname| curdt| curts|cleannetworkname|\_c0| \_c1|\_c0| \_c1| \_c2|\_c3| \_c4| Protocol\_Used|

+------------+----+-----+-----------+--------------------+-------+--------------+----------+--------+---------+-----------------+----------+--------------------+----------------+----------+--------------------+----------------+---+-------+---+-----+-------+---+-----+----------------+

| 2019-10-01| AZ| HIOS| La Paz|http://uamarketpl...| 21| null| null| 88925| 88925| 8892588925|2024-07-22|2024-07-22 16:37:...| La Paz|2024-07-22|2024-07-22 16:37:...| La Paz| AZ|Arizona|779|Jason|Bowling| 21|Coach| http secured|

| 2019-10-01| AZ| HIOS| La Paz|http://uamarketpl...| 21| null| null| 88925| 88925| 8892588925|2024-07-22|2024-07-22 16:37:...| La Paz|2024-07-22|2024-07-22 16:37:...| La Paz| AZ|Arizona|779|Jason|Bowling| 21|Coach| No Protocol|

| 2019-10-01| AZ| HIOS| La Paz|http://uamarketpl...| 21| null| null| 88925| 88925| 8892588925|2024-07-22|2024-07-22 16:37:...| La Paz|2024-07-22|2024-07-22 16:37:...| La Paz| AZ|Arizona|779|Jason|Bowling| 21|Coach| http secured|

| 2019-10-01| AZ| HIOS| La Paz|http://uamarketpl...| 21| null| null| 88925| 88925| 8892588925|2024-07-22|2024-07-22 16:37:...| La Paz|2024-07-22|2024-07-22 16:37:...| La Paz| AZ|Arizona|779|Jason|Bowling| 21|Coach|http non secured|

| 2019-10-01| AZ| HIOS| La Paz|http://uamarketpl...| 21| null| null| 88925| 88925| 8892588925|2024-07-22|2024-07-22 16:37:...| La Paz|2024-07-22|2024-07-22 16:37:...| La Paz| AZ|Arizona|779|Jason|Bowling| 21|Coach| http secured|

| 2019-10-01| AZ| HIOS| La Paz|http://uamarketpl...| 21| null| null| 88925| 88925| 8892588925|2024-07-22|2024-07-22 16:37:...| La Paz|2024-07-22|2024-07-22 16:37:...| La Paz| AZ|Arizona|779|Jason|Bowling| 21|Coach| http secured|

| 2019-10-01| AZ| HIOS| La Paz|http://uamarketpl...| 21| null| null| 88925| 88925| 8892588925|2024-07-22|2024-07-22 16:37:...| La Paz|2024-07-22|2024-07-22 16:37:...| La Paz| AZ|Arizona|779|Jason|Bowling| 21|Coach| http secured|

| 2019-10-01| AZ| HIOS| La Paz|http://uamarketpl...| 21| null| null| 88925| 88925| 8892588925|2024-07-22|2024-07-22 16:37:...| La Paz|2024-07-22|2024-07-22 16:37:...| La Paz| AZ|Arizona|779|Jason|Bowling| 21|Coach|http non secured|

| 2019-10-01| AZ| HIOS| La Paz|http://uamarketpl...| 21| null| null| 88925| 88925| 8892588925|2024-07-22|2024-07-22 16:37:...| La Paz|2024-07-22|2024-07-22 16:37:...| La Paz| AZ|Arizona|779|Jason|Bowling| 21|Coach| No Protocol|

| 2019-10-01| AZ| HIOS| La Paz|http://uamarketpl...| 21| null| null| 88925| 88925| 8892588925|2024-07-22|2024-07-22 16:37:...| La Paz|2024-07-22|2024-07-22 16:37:...| La Paz| AZ|Arizona|779|Jason|Bowling| 21|Coach|http non secured|

| 2019-10-01| AZ| HIOS| La Paz|http://uamarketpl...| 21| null| null| 88925| 88925| 8892588925|2024-07-22|2024-07-22 16:37:...| La Paz|2024-07-22|2024-07-22 16:37:...| La Paz| AZ|Arizona|779|Jason|Bowling| 21|Coach|http non secured|

| 2019-10-01| AZ| HIOS| La Paz|http://uamarketpl...| 21| null| null| 88925| 88925| 8892588925|2024-07-22|2024-07-22 16:37:...| La Paz|2024-07-22|2024-07-22 16:37:...| La Paz| AZ|Arizona|779|Jason|Bowling| 21|Coach| http secured|

| 2019-10-01| AZ| HIOS| La Paz|http://uamarketpl...| 21| null| null| 88925| 88925| 8892588925|2024-07-22|2024-07-22 16:37:...| La Paz|2024-07-22|2024-07-22 16:37:...| La Paz| AZ|Arizona|779|Jason|Bowling| 21|Coach|http non secured|

| 2019-10-01| AZ| HIOS| La Paz|http://uamarketpl...| 21| null| null| 88925| 88925| 8892588925|2024-07-22|2024-07-22 16:37:...| La Paz|2024-07-22|2024-07-22 16:37:...| La Paz| AZ|Arizona|779|Jason|Bowling| 21|Coach|http non secured|

| 2019-10-01| AZ| HIOS| La Paz|http://uamarketpl...| 21| null| null| 88925| 88925| 8892588925|2024-07-22|2024-07-22 16:37:...| La Paz|2024-07-22|2024-07-22 16:37:...| La Paz| AZ|Arizona|779|Jason|Bowling| 21|Coach| http secured|

| 2019-10-01| AZ| HIOS| La Paz|http://uamarketpl...| 21| null| null| 88925| 88925| 8892588925|2024-07-22|2024-07-22 16:37:...| La Paz|2024-07-22|2024-07-22 16:37:...| La Paz| AZ|Arizona|779|Jason|Bowling| 21|Coach| http secured|

| 2019-10-01| AZ| HIOS| La Paz|http://uamarketpl...| 21| null| null| 88925| 88925| 8892588925|2024-07-22|2024-07-22 16:37:...| La Paz|2024-07-22|2024-07-22 16:37:...| La Paz| AZ|Arizona|779|Jason|Bowling| 21|Coach|http non secured|

| 2019-10-01| AZ| HIOS| La Paz|http://uamarketpl...| 21| null| null| 88925| 88925| 8892588925|2024-07-22|2024-07-22 16:37:...| La Paz|2024-07-22|2024-07-22 16:37:...| La Paz| AZ|Arizona|779|Jason|Bowling| 21|Coach| http secured|

| 2019-10-01| AZ| HIOS| La Paz|http://uamarketpl...| 21| null| null| 88925| 88925| 8892588925|2024-07-22|2024-07-22 16:37:...| La Paz|2024-07-22|2024-07-22 16:37:...| La Paz| AZ|Arizona|779|Jason|Bowling| 21|Coach|http non secured|

| 2019-10-01| AZ| HIOS| La Paz|http://uamarketpl...| 21| null| null| 88925| 88925| 8892588925|2024-07-22|2024-07-22 16:37:...| La Paz|2024-07-22|2024-07-22 16:37:...| La Paz| AZ|Arizona|779|Jason|Bowling| 21|Coach|http non secured|

+------------+----+-----+-----------+--------------------+-------+--------------+----------+--------+---------+-----------------+----------+--------------------+----------------+----------+--------------------+----------------+---+-------+---+-----+-------+---+-----+----------------+

only showing top 20 rows

>>> second\_join=spark.sql("select l.\*,r.Protocol\_Used from view1 as l,view2 as r ")

>>> second\_join.createOrReplaceView("view1")

Traceback (most recent call last):

File "<stdin>", line 1, in <module>

File "/usr/local/spark/python/pyspark/sql/dataframe.py", line 1660, in \_\_getattr\_\_

"'%s' object has no attribute '%s'" % (self.\_\_class\_\_.\_\_name\_\_, name))

AttributeError: 'DataFrame' object has no attribute 'createOrReplaceView'

>>> spark.catalog.dropTempView("view`")

>>> spark.catalog.dropTempView("view1")

>>> second\_join.createOrReplaceView("view1")

Traceback (most recent call last):

File "<stdin>", line 1, in <module>

File "/usr/local/spark/python/pyspark/sql/dataframe.py", line 1660, in \_\_getattr\_\_

"'%s' object has no attribute '%s'" % (self.\_\_class\_\_.\_\_name\_\_, name))

AttributeError: 'DataFrame' object has no attribute 'createOrReplaceView'

>>> spark.catalog.dropTempView("view1")

>>> second\_join.createOrReplaceTempView("view1")

>>> spark.sql("select \* from view1 group by Protocol\_Used,srcnm,\_c3 ").show(3)

Traceback (most recent call last):

File "<stdin>", line 1, in <module>

File "/usr/local/spark/python/pyspark/sql/session.py", line 723, in sql

return DataFrame(self.\_jsparkSession.sql(sqlQuery), self.\_wrapped)

File "/usr/local/spark/python/lib/py4j-0.10.9.3-src.zip/py4j/java\_gateway.py", line 1322, in \_\_call\_\_

File "/usr/local/spark/python/pyspark/sql/utils.py", line 117, in deco

raise converted from None

pyspark.sql.utils.AnalysisException: expression 'view1.BusinessDate' is neither present in the group by, nor is it an aggregate function. Add to group by or wrap in first() (or first\_value) if you don't care which value you get.;

Aggregate [Protocol\_Used#48671, srcnm#7747, \_c3#52127], [BusinessDate#1378, stcd#7759, srcnm#7747, NetworkName#1381, NetworkURL#1382, custnum#1383, MarketCoverage#1384, RejectRows#1386, IssuerId#8312, IssuerId2#8325, issueridcomposite#8901, sysdt#9003, systs#9016, cleannetworkname#47665, curdt#48091, curts#48092, cleannetworkname#48111, \_c0#26007, \_c1#26008, \_c0#52124, \_c1#52125, \_c2#52126, \_c3#52127, \_c4#52128, Protocol\_Used#48671]

+- SubqueryAlias view1

+- View (`view1`, [BusinessDate#1378,stcd#7759,srcnm#7747,NetworkName#1381,NetworkURL#1382,custnum#1383,MarketCoverage#1384,RejectRows#1386,IssuerId#8312,IssuerId2#8325,issueridcomposite#8901,sysdt#9003,systs#9016,cleannetworkname#47665,curdt#48091,curts#48092,cleannetworkname#48111,\_c0#26007,\_c1#26008,\_c0#52124,\_c1#52125,\_c2#52126,\_c3#52127,\_c4#52128,Protocol\_Used#48671])

+- Project [BusinessDate#1378, stcd#7759, srcnm#7747, NetworkName#1381, NetworkURL#1382, custnum#1383, MarketCoverage#1384, RejectRows#1386, IssuerId#8312, IssuerId2#8325, issueridcomposite#8901, sysdt#9003, systs#9016, cleannetworkname#47665, curdt#48091, curts#48092, cleannetworkname#48111, \_c0#26007, \_c1#26008, \_c0#52124, \_c1#52125, \_c2#52126, \_c3#52127, \_c4#52128, Protocol\_Used#48671]

+- Join Inner

:- SubqueryAlias l

: +- SubqueryAlias view1

: +- View (`view1`, [BusinessDate#1378,stcd#7759,srcnm#7747,NetworkName#1381,NetworkURL#1382,custnum#1383,MarketCoverage#1384,RejectRows#1386,IssuerId#8312,IssuerId2#8325,issueridcomposite#8901,sysdt#9003,systs#9016,cleannetworkname#47665,curdt#48091,curts#48092,cleannetworkname#48111,\_c0#26007,\_c1#26008,\_c0#52124,\_c1#52125,\_c2#52126,\_c3#52127,\_c4#52128])

: +- Project [BusinessDate#1378, stcd#7759, srcnm#7747, NetworkName#1381, NetworkURL#1382, custnum#1383, MarketCoverage#1384, RejectRows#1386, IssuerId#8312, IssuerId2#8325, issueridcomposite#8901, sysdt#9003, systs#9016, cleannetworkname#47665, curdt#48091, curts#48092, cleannetworkname#48111, \_c0#26007, \_c1#26008, \_c0#52124, \_c1#52125, \_c2#52126, \_c3#52127, \_c4#52128]

: +- Join Inner, (cast(custnum#1383 as int) = \_c3#52127)

: :- SubqueryAlias l

: : +- SubqueryAlias insureview

: : +- View (`insureview`, [BusinessDate#1378,stcd#7759,srcnm#7747,NetworkName#1381,NetworkURL#1382,custnum#1383,MarketCoverage#1384,RejectRows#1386,IssuerId#8312,IssuerId2#8325,issueridcomposite#8901,sysdt#9003,systs#9016,cleannetworkname#47665,curdt#48091,curts#48092,cleannetworkname#48111,\_c0#26007,\_c1#26008])

: : +- Project [BusinessDate#1378, stcd#7759, srcnm#7747, NetworkName#1381, NetworkURL#1382, custnum#1383, MarketCoverage#1384, RejectRows#1386, IssuerId#8312, IssuerId2#8325, issueridcomposite#8901, sysdt#9003, systs#9016, cleannetworkname#47665, curdt#48091, curts#48092, cleannetworkname#48111, \_c0#26007, \_c1#26008]

: : +- Join Inner, (stcd#7759 = \_c0#26007)

: : :- SubqueryAlias l

: : : +- SubqueryAlias insureview

: : : +- View (`insureview`, [BusinessDate#1378,stcd#7759,srcnm#7747,NetworkName#1381,NetworkURL#1382,custnum#1383,MarketCoverage#1384,RejectRows#1386,IssuerId#8312,IssuerId2#8325,issueridcomposite#8901,sysdt#9003,systs#9016,cleannetworkname#47665,curdt#48091,curts#48092,cleannetworkname#48111])

: : : +- Project [BusinessDate#1378, stcd#7759, srcnm#7747, NetworkName#1381, NetworkURL#1382, custnum#1383, MarketCoverage#1384, RejectRows#1386, IssuerId#8312, IssuerId2#8325, issueridcomposite#8901, sysdt#9003, systs#9016, cleannetworkname#47665, curdt#48091, curts#48092, sql\_function(NetworkName#1381) AS cleannetworkname#48111]

: : : +- SubqueryAlias insureview

: : : +- View (`insureview`, [BusinessDate#1378,stcd#7759,srcnm#7747,NetworkName#1381,NetworkURL#1382,custnum#1383,MarketCoverage#1384,RejectRows#1386,IssuerId#8312,IssuerId2#8325,issueridcomposite#8901,sysdt#9003,systs#9016,cleannetworkname#47665,curdt#48091,curts#48092])

: : : +- Project [BusinessDate#1378, stcd#7759, srcnm#7747, NetworkName#1381, NetworkURL#1382, custnum#1383, MarketCoverage#1384, RejectRows#1386, IssuerId#8312, IssuerId2#8325, issueridcomposite#8901, sysdt#9003, systs#9016, cleannetworkname#47665, current\_date(Some(Asia/Kolkata)) AS curdt#48091, current\_timestamp() AS curts#48092]

: : : +- SubqueryAlias insureview

: : : +- View (`insureview`, [BusinessDate#1378,stcd#7759,srcnm#7747,NetworkName#1381,NetworkURL#1382,custnum#1383,MarketCoverage#1384,RejectRows#1386,IssuerId#8312,IssuerId2#8325,issueridcomposite#8901,sysdt#9003,systs#9016,cleannetworkname#47665])

: : : +- Project [BusinessDate#1378, stcd#7759, srcnm#7747, NetworkName#1381, NetworkURL#1382, custnum#1383, MarketCoverage#1384, RejectRows#1386, IssuerId#8312, IssuerId2#8325, issueridcomposite#8901, sysdt#9003, systs#9016, sql\_function(NetworkName#1381) AS cleannetworkname#47665]

: : : +- SubqueryAlias insureview

: : : +- View (`insureview`, [BusinessDate#1378,stcd#7759,srcnm#7747,NetworkName#1381,NetworkURL#1382,custnum#1383,MarketCoverage#1384,RejectRows#1386,IssuerId#8312,IssuerId2#8325,issueridcomposite#8901,sysdt#9003,systs#9016])

: : : +- Project [BusinessDate#1378, stcd#7759, srcnm#7747, NetworkName#1381, NetworkURL#1382, custnum#1383, MarketCoverage#1384, RejectRows#1386, IssuerId#8312, IssuerId2#8325, issueridcomposite#8901, sysdt#9003, current\_timestamp() AS systs#9016]

: : : +- Project [BusinessDate#1378, stcd#7759, srcnm#7747, NetworkName#1381, NetworkURL#1382, custnum#1383, MarketCoverage#1384, RejectRows#1386, IssuerId#8312, IssuerId2#8325, issueridcomposite#8901, current\_date(Some(Asia/Kolkata)) AS sysdt#9003]

: : : +- Project [BusinessDate#1378, stcd#7759, srcnm#7747, NetworkName#1381, NetworkURL#1382, custnum#1383, MarketCoverage#1384, RejectRows#1386, IssuerId#8312, IssuerId2#8325, issueridcomposite#8901]

: : : +- Project [BusinessDate#1378, stcd#7759, srcnm#7747, NetworkName#1381, NetworkURL#1382, custnum#1383, MarketCoverage#1384, DentalOnlyPlan#1385, RejectRows#1386, IssuerId#8312, IssuerId2#8325, concat(IssuerId#8312, IssuerId2#8325) AS issueridcomposite#8901]

: : : +- Project [BusinessDate#1378, stcd#7759, srcnm#7747, NetworkName#1381, NetworkURL#1382, custnum#1383, MarketCoverage#1384, DentalOnlyPlan#1385, RejectRows#1386, IssuerId#8312, IssuerId2#8325]

: : : +- Project [IssuerIDDD#8273, BusinessDate#1378, stcd#7759, srcnm#7747, NetworkName#1381, NetworkURL#1382, custnum#1383, MarketCoverage#1384, DentalOnlyPlan#1385, RejectRows#1386, IssuerId#8312, IssuerId2#8325]

: : : +- Project [IssuerIDD#8261, IssuerIDDD#8273, BusinessDate#1378, stcd#7759, srcnm#7747, NetworkName#1381, NetworkURL#1382, custnum#1383, MarketCoverage#1384, DentalOnlyPlan#1385, RejectRows#1386, IssuerId#8312, cast(IssuerIDDD#8273 as string) AS IssuerId2#8325]

: : : +- Project [IssuerIDD#8261, IssuerIDDD#8273, BusinessDate#1378, stcd#7759, srcnm#7747, NetworkName#1381, NetworkURL#1382, custnum#1383, MarketCoverage#1384, DentalOnlyPlan#1385, RejectRows#1386, cast(IssuerIDD#8261 as string) AS IssuerId#8312]

: : : +- Project [IssuerIDD#8261, IssuerId2#1377 AS IssuerIDDD#8273, BusinessDate#1378, stcd#7759, srcnm#7747, NetworkName#1381, NetworkURL#1382, custnum#1383, MarketCoverage#1384, DentalOnlyPlan#1385, RejectRows#1386]

: : : +- Project [IssuerId#1376 AS IssuerIDD#8261, IssuerId2#1377, BusinessDate#1378, stcd#7759, srcnm#7747, NetworkName#1381, NetworkURL#1382, custnum#1383, MarketCoverage#1384, DentalOnlyPlan#1385, RejectRows#1386]

: : : +- Project [IssuerId#1376, IssuerId2#1377, BusinessDate#1378, StateCode#1379 AS stcd#7759, srcnm#7747, NetworkName#1381, NetworkURL#1382, custnum#1383, MarketCoverage#1384, DentalOnlyPlan#1385, RejectRows#1386]

: : : +- Project [IssuerId#1376, IssuerId2#1377, BusinessDate#1378, StateCode#1379, SourceName#1380 AS srcnm#7747, NetworkName#1381, NetworkURL#1382, custnum#1383, MarketCoverage#1384, DentalOnlyPlan#1385, RejectRows#1386]

: : : +- Union false, false

: : : :- Relation [IssuerId#1376,IssuerId2#1377,BusinessDate#1378,StateCode#1379,SourceName#1380,NetworkName#1381,NetworkURL#1382,custnum#1383,MarketCoverage#1384,DentalOnlyPlan#1385,RejectRows#1386] csv

: : : +- Relation [IssuerId#491,IssuerId2#492,BusinessDate#493,StateCode#494,SourceName#495,NetworkName#496,NetworkURL#497,custnum#498,MarketCoverage#499,DentalOnlyPlan#500,RejectRows#501] csv

: : +- SubqueryAlias r

: : +- SubqueryAlias state\_table

: : +- View (`state\_table`, [\_c0#26007,\_c1#26008])

: : +- Project [\_c0#26007, \_c1#26008]

: : +- Filter ((isnull(\_c2#26009) AND isnull(\_c3#26010)) AND isnull(\_c4#26011))

: : +- Project [\_c0#26007, \_c1#26008, \_c2#26009, \_c3#26010, \_c4#26011]

: : +- Relation [\_c0#26007,\_c1#26008,\_c2#26009,\_c3#26010,\_c4#26011] csv

: +- SubqueryAlias r

: +- SubqueryAlias cust\_table

: +- View (`cust\_table`, [\_c0#52124,\_c1#52125,\_c2#52126,\_c3#52127,\_c4#52128])

: +- Filter (((isnotnull(\_c0#52124) AND isnotnull(\_c1#52125)) AND isnotnull(\_c2#52126)) AND (isnotnull(\_c3#52127) AND isnotnull(\_c4#52128)))

: +- Project [\_c0#52124, \_c1#52125, \_c2#52126, \_c3#52127, \_c4#52128]

: +- Relation [\_c0#52124,\_c1#52125,\_c2#52126,\_c3#52127,\_c4#52128] csv

+- SubqueryAlias r

+- SubqueryAlias view2

+- View (`view2`, [BusinessDate#57031,stcd#57060,srcnm#57061,NetworkName#57034,NetworkURL#57035,custnum#57036,MarketCoverage#57037,RejectRows#57039,IssuerId#57059,IssuerId2#57058,issueridcomposite#57057,sysdt#57056,systs#57055,cleannetworkname#57054,curdt#57052,curts#57053,cleannetworkname#57051,Protocol\_Used#48671])

+- Project [BusinessDate#57031, stcd#57060, srcnm#57061, NetworkName#57034, NetworkURL#57035, custnum#57036, MarketCoverage#57037, RejectRows#57039, IssuerId#57059, IssuerId2#57058, issueridcomposite#57057, sysdt#57056, systs#57055, cleannetworkname#57054, curdt#57052, curts#57053, cleannetworkname#57051, CASE WHEN NetworkURL#57035 LIKE https://% THEN http secured WHEN NetworkURL#57035 LIKE http://% THEN http non secured ELSE No Protocol END AS Protocol\_Used#48671]

+- SubqueryAlias insureview

+- View (`insureview`, [BusinessDate#57031,stcd#57060,srcnm#57061,NetworkName#57034,NetworkURL#57035,custnum#57036,MarketCoverage#57037,RejectRows#57039,IssuerId#57059,IssuerId2#57058,issueridcomposite#57057,sysdt#57056,systs#57055,cleannetworkname#57054,curdt#57052,curts#57053,cleannetworkname#57051])

+- Project [BusinessDate#57031, stcd#57060, srcnm#57061, NetworkName#57034, NetworkURL#57035, custnum#57036, MarketCoverage#57037, RejectRows#57039, IssuerId#57059, IssuerId2#57058, issueridcomposite#57057, sysdt#57056, systs#57055, cleannetworkname#57054, curdt#57052, curts#57053, sql\_function(NetworkName#57034) AS cleannetworkname#57051]

+- SubqueryAlias insureview

+- View (`insureview`, [BusinessDate#57031,stcd#57060,srcnm#57061,NetworkName#57034,NetworkURL#57035,custnum#57036,MarketCoverage#57037,RejectRows#57039,IssuerId#57059,IssuerId2#57058,issueridcomposite#57057,sysdt#57056,systs#57055,cleannetworkname#57054,curdt#57052,curts#57053])

+- Project [BusinessDate#57031, stcd#57060, srcnm#57061, NetworkName#57034, NetworkURL#57035, custnum#57036, MarketCoverage#57037, RejectRows#57039, IssuerId#57059, IssuerId2#57058, issueridcomposite#57057, sysdt#57056, systs#57055, cleannetworkname#57054, current\_date(Some(Asia/Kolkata)) AS curdt#57052, current\_timestamp() AS curts#57053]

+- SubqueryAlias insureview

+- View (`insureview`, [BusinessDate#57031,stcd#57060,srcnm#57061,NetworkName#57034,NetworkURL#57035,custnum#57036,MarketCoverage#57037,RejectRows#57039,IssuerId#57059,IssuerId2#57058,issueridcomposite#57057,sysdt#57056,systs#57055,cleannetworkname#57054])

+- Project [BusinessDate#57031, stcd#57060, srcnm#57061, NetworkName#57034, NetworkURL#57035, custnum#57036, MarketCoverage#57037, RejectRows#57039, IssuerId#57059, IssuerId2#57058, issueridcomposite#57057, sysdt#57056, systs#57055, sql\_function(NetworkName#57034) AS cleannetworkname#57054]

+- SubqueryAlias insureview

+- View (`insureview`, [BusinessDate#57031,stcd#57060,srcnm#57061,NetworkName#57034,NetworkURL#57035,custnum#57036,MarketCoverage#57037,RejectRows#57039,IssuerId#57059,IssuerId2#57058,issueridcomposite#57057,sysdt#57056,systs#57055])

+- Project [BusinessDate#57031, stcd#57060, srcnm#57061, NetworkName#57034, NetworkURL#57035, custnum#57036, MarketCoverage#57037, RejectRows#57039, IssuerId#57059, IssuerId2#57058, issueridcomposite#57057, sysdt#57056, current\_timestamp() AS systs#57055]

+- Project [BusinessDate#57031, stcd#57060, srcnm#57061, NetworkName#57034, NetworkURL#57035, custnum#57036, MarketCoverage#57037, RejectRows#57039, IssuerId#57059, IssuerId2#57058, issueridcomposite#57057, current\_date(Some(Asia/Kolkata)) AS sysdt#57056]

+- Project [BusinessDate#57031, stcd#57060, srcnm#57061, NetworkName#57034, NetworkURL#57035, custnum#57036, MarketCoverage#57037, RejectRows#57039, IssuerId#57059, IssuerId2#57058, issueridcomposite#57057]

+- Project [BusinessDate#57031, stcd#57060, srcnm#57061, NetworkName#57034, NetworkURL#57035, custnum#57036, MarketCoverage#57037, DentalOnlyPlan#57038, RejectRows#57039, IssuerId#57059, IssuerId2#57058, concat(IssuerId#57059, IssuerId2#57058) AS issueridcomposite#57057]

+- Project [BusinessDate#57031, stcd#57060, srcnm#57061, NetworkName#57034, NetworkURL#57035, custnum#57036, MarketCoverage#57037, DentalOnlyPlan#57038, RejectRows#57039, IssuerId#57059, IssuerId2#57058]

+- Project [IssuerIDDD#8273, BusinessDate#57031, stcd#57060, srcnm#57061, NetworkName#57034, NetworkURL#57035, custnum#57036, MarketCoverage#57037, DentalOnlyPlan#57038, RejectRows#57039, IssuerId#57059, IssuerId2#57058]

+- Project [IssuerIDD#8261, IssuerIDDD#8273, BusinessDate#57031, stcd#57060, srcnm#57061, NetworkName#57034, NetworkURL#57035, custnum#57036, MarketCoverage#57037, DentalOnlyPlan#57038, RejectRows#57039, IssuerId#57059, cast(IssuerIDDD#8273 as string) AS IssuerId2#57058]

+- Project [IssuerIDD#8261, IssuerIDDD#8273, BusinessDate#57031, stcd#57060, srcnm#57061, NetworkName#57034, NetworkURL#57035, custnum#57036, MarketCoverage#57037, DentalOnlyPlan#57038, RejectRows#57039, cast(IssuerIDD#8261 as string) AS IssuerId#57059]

+- Project [IssuerIDD#8261, IssuerId2#57030 AS IssuerIDDD#8273, BusinessDate#57031, stcd#57060, srcnm#57061, NetworkName#57034, NetworkURL#57035, custnum#57036, MarketCoverage#57037, DentalOnlyPlan#57038, RejectRows#57039]

+- Project [IssuerId#57029 AS IssuerIDD#8261, IssuerId2#57030, BusinessDate#57031, stcd#57060, srcnm#57061, NetworkName#57034, NetworkURL#57035, custnum#57036, MarketCoverage#57037, DentalOnlyPlan#57038, RejectRows#57039]

+- Project [IssuerId#57029, IssuerId2#57030, BusinessDate#57031, StateCode#57032 AS stcd#57060, srcnm#57061, NetworkName#57034, NetworkURL#57035, custnum#57036, MarketCoverage#57037, DentalOnlyPlan#57038, RejectRows#57039]

+- Project [IssuerId#57029, IssuerId2#57030, BusinessDate#57031, StateCode#57032, SourceName#57033 AS srcnm#57061, NetworkName#57034, NetworkURL#57035, custnum#57036, MarketCoverage#57037, DentalOnlyPlan#57038, RejectRows#57039]

+- Union false, false

:- Relation [IssuerId#57029,IssuerId2#57030,BusinessDate#57031,StateCode#57032,SourceName#57033,NetworkName#57034,NetworkURL#57035,custnum#57036,MarketCoverage#57037,DentalOnlyPlan#57038,RejectRows#57039] csv

+- Relation [IssuerId#57040,IssuerId2#57041,BusinessDate#57042,StateCode#57043,SourceName#57044,NetworkName#57045,NetworkURL#57046,custnum#57047,MarketCoverage#57048,DentalOnlyPlan#57049,RejectRows#57050] csv

>>>

>>> WITH CountData AS (

File "<stdin>", line 1

WITH CountData AS (

^

SyntaxError: invalid syntax

>>> SELECT

File "<stdin>", line 1

SELECT

^

IndentationError: unexpected indent

>>> statedesc,

File "<stdin>", line 1

statedesc,

^

IndentationError: unexpected indent

>>> Protocol\_Used AS protocol,

File "<stdin>", line 1

Protocol\_Used AS protocol,

^

IndentationError: unexpected indent

>>> profession,

File "<stdin>", line 1

profession,

^

IndentationError: unexpected indent

>>> AVG(age) AS avg\_age,

File "<stdin>", line 1

AVG(age) AS avg\_age,

^

IndentationError: unexpected indent

>>> COUNT(\*) AS profession\_count,

File "<stdin>", line 1

COUNT(\*) AS profession\_count,

^

IndentationError: unexpected indent

>>> ROW\_NUMBER() OVER (PARTITION BY statedesc, protocol ORDER BY COUNT(\*) DESC) AS seqno

File "<stdin>", line 1

ROW\_NUMBER() OVER (PARTITION BY statedesc, protocol ORDER BY COUNT(\*) DESC) AS seqno

^

IndentationError: unexpected indent

>>> FROM view1

File "<stdin>", line 1

FROM view1

^

IndentationError: unexpected indent

>>> GROUP BY statedesc, protocol, profession

File "<stdin>", line 1

GROUP BY statedesc, protocol, profession

^

IndentationError: unexpected indent

>>> )

File "<stdin>", line 1

)

^

SyntaxError: invalid syntax

>>>

>>> SELECT

Traceback (most recent call last):

File "<stdin>", line 1, in <module>

NameError: name 'SELECT' is not defined

>>> statedesc,

File "<stdin>", line 1

statedesc,

^

IndentationError: unexpected indent

>>> protocol,

File "<stdin>", line 1

protocol,

^

IndentationError: unexpected indent

>>> profession,

File "<stdin>", line 1

profession,

^

IndentationError: unexpected indent

>>> avg\_age,

File "<stdin>", line 1

avg\_age,

^

IndentationError: unexpected indent

>>> profession\_count

File "<stdin>", line 1

profession\_count

^

IndentationError: unexpected indent

>>> FROM CountData

File "<stdin>", line 1

FROM CountData

^

SyntaxError: invalid syntax

>>> WHERE seqno = 2

>>> from pyspark.sql.functions import col, avg, count

>>> from pyspark.sql.window import Window

>>> from pyspark.sql.functions import row\_number

>>>

>>> # Calculate average age and count, and add seqno

... window\_spec = Window.partitionBy("stcd", "Protocol\_Used").orderBy(col("profession\_count").desc())

>>> count\_df = df.groupBy("stcd", "Protocol\_Used", "profession") \

... .agg(

... avg("age").alias("avg\_age"),

... count("\*").alias("profession\_count")

... ) \

... .withColumn("seqno", row\_number().over(window\_spec))

Traceback (most recent call last):

File "<stdin>", line 1, in <module>

NameError: name 'df' is not defined

>>> count\_df = df\_renamed.groupBy("stcd", "Protocol\_Used", "profession") \

... .agg(

... avg("age").alias("avg\_age"),

... count("\*").alias("profession\_count")

... ) \

... .withColumn("seqno", row\_number().over(window\_spec))

Traceback (most recent call last):

File "<stdin>", line 4, in <module>

File "/usr/local/spark/python/pyspark/sql/group.py", line 119, in agg

\_to\_seq(self.sql\_ctx.\_sc, [c.\_jc for c in exprs[1:]]))

File "/usr/local/spark/python/lib/py4j-0.10.9.3-src.zip/py4j/java\_gateway.py", line 1322, in \_\_call\_\_

File "/usr/local/spark/python/pyspark/sql/utils.py", line 117, in deco

raise converted from None

pyspark.sql.utils.AnalysisException: cannot resolve 'Protocol\_Used' given input columns: [l.BusinessDate, City, City, l.IssuerId, l.IssuerId2, l.MarketCoverage, l.NetworkName, l.NetworkURL, l.RejectRows, r.\_c2, r.\_c3, r.\_c4, l.cleannetworkname, l.cleannetworkname, l.curdt, l.curts, l.custnum, l.issueridcomposite, l.srcnm, state\_code, state\_code, l.stcd, l.sysdt, l.systs];

'Aggregate [stcd#7759, 'Protocol\_Used, 'profession], [stcd#7759, 'Protocol\_Used, 'profession, avg('age) AS avg\_age#57112, count(1) AS profession\_count#57114L]

+- Project [BusinessDate#1378, stcd#7759, srcnm#7747, NetworkName#1381, NetworkURL#1382, custnum#1383, MarketCoverage#1384, RejectRows#1386, IssuerId#8312, IssuerId2#8325, issueridcomposite#8901, sysdt#9003, systs#9016, cleannetworkname#47665, curdt#48091, curts#48092, cleannetworkname#48111, state\_code#53694, \_c1#26008 AS City#53720, state\_code#53695, \_c1#52125 AS City#53721, \_c2#52126, \_c3#52127, \_c4#52128]

+- Project [BusinessDate#1378, stcd#7759, srcnm#7747, NetworkName#1381, NetworkURL#1382, custnum#1383, MarketCoverage#1384, RejectRows#1386, IssuerId#8312, IssuerId2#8325, issueridcomposite#8901, sysdt#9003, systs#9016, cleannetworkname#47665, curdt#48091, curts#48092, cleannetworkname#48111, \_c0#26007 AS state\_code#53694, \_c1#26008, \_c0#52124 AS state\_code#53695, \_c1#52125, \_c2#52126, \_c3#52127, \_c4#52128]

+- Project [BusinessDate#1378, stcd#7759, srcnm#7747, NetworkName#1381, NetworkURL#1382, custnum#1383, MarketCoverage#1384, RejectRows#1386, IssuerId#8312, IssuerId2#8325, issueridcomposite#8901, sysdt#9003, systs#9016, cleannetworkname#47665, curdt#48091, curts#48092, cleannetworkname#48111, \_c0#26007, \_c1#26008, \_c0#52124, \_c1#52125, \_c2#52126, \_c3#52127, \_c4#52128]

+- Join Inner, (cast(custnum#1383 as int) = \_c3#52127)

:- SubqueryAlias l

: +- SubqueryAlias insureview

: +- View (`insureview`, [BusinessDate#1378,stcd#7759,srcnm#7747,NetworkName#1381,NetworkURL#1382,custnum#1383,MarketCoverage#1384,RejectRows#1386,IssuerId#8312,IssuerId2#8325,issueridcomposite#8901,sysdt#9003,systs#9016,cleannetworkname#47665,curdt#48091,curts#48092,cleannetworkname#48111,\_c0#26007,\_c1#26008])

: +- Project [BusinessDate#1378, stcd#7759, srcnm#7747, NetworkName#1381, NetworkURL#1382, custnum#1383, MarketCoverage#1384, RejectRows#1386, IssuerId#8312, IssuerId2#8325, issueridcomposite#8901, sysdt#9003, systs#9016, cleannetworkname#47665, curdt#48091, curts#48092, cleannetworkname#48111, \_c0#26007, \_c1#26008]

: +- Join Inner, (stcd#7759 = \_c0#26007)

: :- SubqueryAlias l

: : +- SubqueryAlias insureview

: : +- View (`insureview`, [BusinessDate#1378,stcd#7759,srcnm#7747,NetworkName#1381,NetworkURL#1382,custnum#1383,MarketCoverage#1384,RejectRows#1386,IssuerId#8312,IssuerId2#8325,issueridcomposite#8901,sysdt#9003,systs#9016,cleannetworkname#47665,curdt#48091,curts#48092,cleannetworkname#48111])

: : +- Project [BusinessDate#1378, stcd#7759, srcnm#7747, NetworkName#1381, NetworkURL#1382, custnum#1383, MarketCoverage#1384, RejectRows#1386, IssuerId#8312, IssuerId2#8325, issueridcomposite#8901, sysdt#9003, systs#9016, cleannetworkname#47665, curdt#48091, curts#48092, sql\_function(NetworkName#1381) AS cleannetworkname#48111]

: : +- SubqueryAlias insureview

: : +- View (`insureview`, [BusinessDate#1378,stcd#7759,srcnm#7747,NetworkName#1381,NetworkURL#1382,custnum#1383,MarketCoverage#1384,RejectRows#1386,IssuerId#8312,IssuerId2#8325,issueridcomposite#8901,sysdt#9003,systs#9016,cleannetworkname#47665,curdt#48091,curts#48092])

: : +- Project [BusinessDate#1378, stcd#7759, srcnm#7747, NetworkName#1381, NetworkURL#1382, custnum#1383, MarketCoverage#1384, RejectRows#1386, IssuerId#8312, IssuerId2#8325, issueridcomposite#8901, sysdt#9003, systs#9016, cleannetworkname#47665, current\_date(Some(Asia/Kolkata)) AS curdt#48091, current\_timestamp() AS curts#48092]

: : +- SubqueryAlias insureview

: : +- View (`insureview`, [BusinessDate#1378,stcd#7759,srcnm#7747,NetworkName#1381,NetworkURL#1382,custnum#1383,MarketCoverage#1384,RejectRows#1386,IssuerId#8312,IssuerId2#8325,issueridcomposite#8901,sysdt#9003,systs#9016,cleannetworkname#47665])

: : +- Project [BusinessDate#1378, stcd#7759, srcnm#7747, NetworkName#1381, NetworkURL#1382, custnum#1383, MarketCoverage#1384, RejectRows#1386, IssuerId#8312, IssuerId2#8325, issueridcomposite#8901, sysdt#9003, systs#9016, sql\_function(NetworkName#1381) AS cleannetworkname#47665]

: : +- SubqueryAlias insureview

: : +- View (`insureview`, [BusinessDate#1378,stcd#7759,srcnm#7747,NetworkName#1381,NetworkURL#1382,custnum#1383,MarketCoverage#1384,RejectRows#1386,IssuerId#8312,IssuerId2#8325,issueridcomposite#8901,sysdt#9003,systs#9016])

: : +- Project [BusinessDate#1378, stcd#7759, srcnm#7747, NetworkName#1381, NetworkURL#1382, custnum#1383, MarketCoverage#1384, RejectRows#1386, IssuerId#8312, IssuerId2#8325, issueridcomposite#8901, sysdt#9003, current\_timestamp() AS systs#9016]

: : +- Project [BusinessDate#1378, stcd#7759, srcnm#7747, NetworkName#1381, NetworkURL#1382, custnum#1383, MarketCoverage#1384, RejectRows#1386, IssuerId#8312, IssuerId2#8325, issueridcomposite#8901, current\_date(Some(Asia/Kolkata)) AS sysdt#9003]

: : +- Project [BusinessDate#1378, stcd#7759, srcnm#7747, NetworkName#1381, NetworkURL#1382, custnum#1383, MarketCoverage#1384, RejectRows#1386, IssuerId#8312, IssuerId2#8325, issueridcomposite#8901]

: : +- Project [BusinessDate#1378, stcd#7759, srcnm#7747, NetworkName#1381, NetworkURL#1382, custnum#1383, MarketCoverage#1384, DentalOnlyPlan#1385, RejectRows#1386, IssuerId#8312, IssuerId2#8325, concat(IssuerId#8312, IssuerId2#8325) AS issueridcomposite#8901]

: : +- Project [BusinessDate#1378, stcd#7759, srcnm#7747, NetworkName#1381, NetworkURL#1382, custnum#1383, MarketCoverage#1384, DentalOnlyPlan#1385, RejectRows#1386, IssuerId#8312, IssuerId2#8325]

: : +- Project [IssuerIDDD#8273, BusinessDate#1378, stcd#7759, srcnm#7747, NetworkName#1381, NetworkURL#1382, custnum#1383, MarketCoverage#1384, DentalOnlyPlan#1385, RejectRows#1386, IssuerId#8312, IssuerId2#8325]

: : +- Project [IssuerIDD#8261, IssuerIDDD#8273, BusinessDate#1378, stcd#7759, srcnm#7747, NetworkName#1381, NetworkURL#1382, custnum#1383, MarketCoverage#1384, DentalOnlyPlan#1385, RejectRows#1386, IssuerId#8312, cast(IssuerIDDD#8273 as string) AS IssuerId2#8325]

: : +- Project [IssuerIDD#8261, IssuerIDDD#8273, BusinessDate#1378, stcd#7759, srcnm#7747, NetworkName#1381, NetworkURL#1382, custnum#1383, MarketCoverage#1384, DentalOnlyPlan#1385, RejectRows#1386, cast(IssuerIDD#8261 as string) AS IssuerId#8312]

: : +- Project [IssuerIDD#8261, IssuerId2#1377 AS IssuerIDDD#8273, BusinessDate#1378, stcd#7759, srcnm#7747, NetworkName#1381, NetworkURL#1382, custnum#1383, MarketCoverage#1384, DentalOnlyPlan#1385, RejectRows#1386]

: : +- Project [IssuerId#1376 AS IssuerIDD#8261, IssuerId2#1377, BusinessDate#1378, stcd#7759, srcnm#7747, NetworkName#1381, NetworkURL#1382, custnum#1383, MarketCoverage#1384, DentalOnlyPlan#1385, RejectRows#1386]

: : +- Project [IssuerId#1376, IssuerId2#1377, BusinessDate#1378, StateCode#1379 AS stcd#7759, srcnm#7747, NetworkName#1381, NetworkURL#1382, custnum#1383, MarketCoverage#1384, DentalOnlyPlan#1385, RejectRows#1386]

: : +- Project [IssuerId#1376, IssuerId2#1377, BusinessDate#1378, StateCode#1379, SourceName#1380 AS srcnm#7747, NetworkName#1381, NetworkURL#1382, custnum#1383, MarketCoverage#1384, DentalOnlyPlan#1385, RejectRows#1386]

: : +- Union false, false

: : :- Relation [IssuerId#1376,IssuerId2#1377,BusinessDate#1378,StateCode#1379,SourceName#1380,NetworkName#1381,NetworkURL#1382,custnum#1383,MarketCoverage#1384,DentalOnlyPlan#1385,RejectRows#1386] csv

: : +- Relation [IssuerId#491,IssuerId2#492,BusinessDate#493,StateCode#494,SourceName#495,NetworkName#496,NetworkURL#497,custnum#498,MarketCoverage#499,DentalOnlyPlan#500,RejectRows#501] csv

: +- SubqueryAlias r

: +- SubqueryAlias state\_table

: +- View (`state\_table`, [\_c0#26007,\_c1#26008])

: +- Project [\_c0#26007, \_c1#26008]

: +- Filter ((isnull(\_c2#26009) AND isnull(\_c3#26010)) AND isnull(\_c4#26011))

: +- Project [\_c0#26007, \_c1#26008, \_c2#26009, \_c3#26010, \_c4#26011]

: +- Relation [\_c0#26007,\_c1#26008,\_c2#26009,\_c3#26010,\_c4#26011] csv

+- SubqueryAlias r

+- SubqueryAlias cust\_table

+- View (`cust\_table`, [\_c0#52124,\_c1#52125,\_c2#52126,\_c3#52127,\_c4#52128])

+- Filter (((isnotnull(\_c0#52124) AND isnotnull(\_c1#52125)) AND isnotnull(\_c2#52126)) AND (isnotnull(\_c3#52127) AND isnotnull(\_c4#52128)))

+- Project [\_c0#52124, \_c1#52125, \_c2#52126, \_c3#52127, \_c4#52128]

+- Relation [\_c0#52124,\_c1#52125,\_c2#52126,\_c3#52127,\_c4#52128] csv

>>>

>>> from pyspark.sql.functions import col, avg, count

>>> from pyspark.sql.window import Window

>>> from pyspark.sql.functions import row\_number

>>>

>>> # Calculate average age and count, and add seqno

... window\_spec = Window.partitionBy("stcd", "Protocol\_Used").orderBy(col("profession\_count").desc())

>>>

>>> count\_df = df.groupBy("stcd", "Protocol\_Used", "profession") \

... .agg(

... avg("\_c1").alias("avg\_age"),

... count("\*").alias("profession\_count")

... ) \

... .withColumn("seqno", row\_number().over(window\_spec))

Traceback (most recent call last):

File "<stdin>", line 1, in <module>

NameError: name 'df' is not defined

>>> count\_df = df\_renamed.groupBy("stcd", "Protocol\_Used", "profession") \

... .agg(

... avg("\_c1").alias("avg\_age"),

... count("\*").alias("profession\_count")

... ) \

... .withColumn("seqno", row\_number().over(window\_spec))

Traceback (most recent call last):

File "<stdin>", line 4, in <module>

File "/usr/local/spark/python/pyspark/sql/group.py", line 119, in agg

\_to\_seq(self.sql\_ctx.\_sc, [c.\_jc for c in exprs[1:]]))

File "/usr/local/spark/python/lib/py4j-0.10.9.3-src.zip/py4j/java\_gateway.py", line 1322, in \_\_call\_\_

File "/usr/local/spark/python/pyspark/sql/utils.py", line 117, in deco

raise converted from None

pyspark.sql.utils.AnalysisException: cannot resolve 'Protocol\_Used' given input columns: [l.BusinessDate, City, City, l.IssuerId, l.IssuerId2, l.MarketCoverage, l.NetworkName, l.NetworkURL, l.RejectRows, r.\_c2, r.\_c3, r.\_c4, l.cleannetworkname, l.cleannetworkname, l.curdt, l.curts, l.custnum, l.issueridcomposite, l.srcnm, state\_code, state\_code, l.stcd, l.sysdt, l.systs];

'Aggregate [stcd#7759, 'Protocol\_Used, 'profession], [stcd#7759, 'Protocol\_Used, 'profession, avg('\_c1) AS avg\_age#57140, count(1) AS profession\_count#57142L]

+- Project [BusinessDate#1378, stcd#7759, srcnm#7747, NetworkName#1381, NetworkURL#1382, custnum#1383, MarketCoverage#1384, RejectRows#1386, IssuerId#8312, IssuerId2#8325, issueridcomposite#8901, sysdt#9003, systs#9016, cleannetworkname#47665, curdt#48091, curts#48092, cleannetworkname#48111, state\_code#53694, \_c1#26008 AS City#53720, state\_code#53695, \_c1#52125 AS City#53721, \_c2#52126, \_c3#52127, \_c4#52128]

+- Project [BusinessDate#1378, stcd#7759, srcnm#7747, NetworkName#1381, NetworkURL#1382, custnum#1383, MarketCoverage#1384, RejectRows#1386, IssuerId#8312, IssuerId2#8325, issueridcomposite#8901, sysdt#9003, systs#9016, cleannetworkname#47665, curdt#48091, curts#48092, cleannetworkname#48111, \_c0#26007 AS state\_code#53694, \_c1#26008, \_c0#52124 AS state\_code#53695, \_c1#52125, \_c2#52126, \_c3#52127, \_c4#52128]

+- Project [BusinessDate#1378, stcd#7759, srcnm#7747, NetworkName#1381, NetworkURL#1382, custnum#1383, MarketCoverage#1384, RejectRows#1386, IssuerId#8312, IssuerId2#8325, issueridcomposite#8901, sysdt#9003, systs#9016, cleannetworkname#47665, curdt#48091, curts#48092, cleannetworkname#48111, \_c0#26007, \_c1#26008, \_c0#52124, \_c1#52125, \_c2#52126, \_c3#52127, \_c4#52128]

+- Join Inner, (cast(custnum#1383 as int) = \_c3#52127)

:- SubqueryAlias l

: +- SubqueryAlias insureview

: +- View (`insureview`, [BusinessDate#1378,stcd#7759,srcnm#7747,NetworkName#1381,NetworkURL#1382,custnum#1383,MarketCoverage#1384,RejectRows#1386,IssuerId#8312,IssuerId2#8325,issueridcomposite#8901,sysdt#9003,systs#9016,cleannetworkname#47665,curdt#48091,curts#48092,cleannetworkname#48111,\_c0#26007,\_c1#26008])

: +- Project [BusinessDate#1378, stcd#7759, srcnm#7747, NetworkName#1381, NetworkURL#1382, custnum#1383, MarketCoverage#1384, RejectRows#1386, IssuerId#8312, IssuerId2#8325, issueridcomposite#8901, sysdt#9003, systs#9016, cleannetworkname#47665, curdt#48091, curts#48092, cleannetworkname#48111, \_c0#26007, \_c1#26008]

: +- Join Inner, (stcd#7759 = \_c0#26007)

: :- SubqueryAlias l

: : +- SubqueryAlias insureview

: : +- View (`insureview`, [BusinessDate#1378,stcd#7759,srcnm#7747,NetworkName#1381,NetworkURL#1382,custnum#1383,MarketCoverage#1384,RejectRows#1386,IssuerId#8312,IssuerId2#8325,issueridcomposite#8901,sysdt#9003,systs#9016,cleannetworkname#47665,curdt#48091,curts#48092,cleannetworkname#48111])

: : +- Project [BusinessDate#1378, stcd#7759, srcnm#7747, NetworkName#1381, NetworkURL#1382, custnum#1383, MarketCoverage#1384, RejectRows#1386, IssuerId#8312, IssuerId2#8325, issueridcomposite#8901, sysdt#9003, systs#9016, cleannetworkname#47665, curdt#48091, curts#48092, sql\_function(NetworkName#1381) AS cleannetworkname#48111]

: : +- SubqueryAlias insureview

: : +- View (`insureview`, [BusinessDate#1378,stcd#7759,srcnm#7747,NetworkName#1381,NetworkURL#1382,custnum#1383,MarketCoverage#1384,RejectRows#1386,IssuerId#8312,IssuerId2#8325,issueridcomposite#8901,sysdt#9003,systs#9016,cleannetworkname#47665,curdt#48091,curts#48092])

: : +- Project [BusinessDate#1378, stcd#7759, srcnm#7747, NetworkName#1381, NetworkURL#1382, custnum#1383, MarketCoverage#1384, RejectRows#1386, IssuerId#8312, IssuerId2#8325, issueridcomposite#8901, sysdt#9003, systs#9016, cleannetworkname#47665, current\_date(Some(Asia/Kolkata)) AS curdt#48091, current\_timestamp() AS curts#48092]

: : +- SubqueryAlias insureview

: : +- View (`insureview`, [BusinessDate#1378,stcd#7759,srcnm#7747,NetworkName#1381,NetworkURL#1382,custnum#1383,MarketCoverage#1384,RejectRows#1386,IssuerId#8312,IssuerId2#8325,issueridcomposite#8901,sysdt#9003,systs#9016,cleannetworkname#47665])

: : +- Project [BusinessDate#1378, stcd#7759, srcnm#7747, NetworkName#1381, NetworkURL#1382, custnum#1383, MarketCoverage#1384, RejectRows#1386, IssuerId#8312, IssuerId2#8325, issueridcomposite#8901, sysdt#9003, systs#9016, sql\_function(NetworkName#1381) AS cleannetworkname#47665]

: : +- SubqueryAlias insureview

: : +- View (`insureview`, [BusinessDate#1378,stcd#7759,srcnm#7747,NetworkName#1381,NetworkURL#1382,custnum#1383,MarketCoverage#1384,RejectRows#1386,IssuerId#8312,IssuerId2#8325,issueridcomposite#8901,sysdt#9003,systs#9016])

: : +- Project [BusinessDate#1378, stcd#7759, srcnm#7747, NetworkName#1381, NetworkURL#1382, custnum#1383, MarketCoverage#1384, RejectRows#1386, IssuerId#8312, IssuerId2#8325, issueridcomposite#8901, sysdt#9003, current\_timestamp() AS systs#9016]

: : +- Project [BusinessDate#1378, stcd#7759, srcnm#7747, NetworkName#1381, NetworkURL#1382, custnum#1383, MarketCoverage#1384, RejectRows#1386, IssuerId#8312, IssuerId2#8325, issueridcomposite#8901, current\_date(Some(Asia/Kolkata)) AS sysdt#9003]

: : +- Project [BusinessDate#1378, stcd#7759, srcnm#7747, NetworkName#1381, NetworkURL#1382, custnum#1383, MarketCoverage#1384, RejectRows#1386, IssuerId#8312, IssuerId2#8325, issueridcomposite#8901]

: : +- Project [BusinessDate#1378, stcd#7759, srcnm#7747, NetworkName#1381, NetworkURL#1382, custnum#1383, MarketCoverage#1384, DentalOnlyPlan#1385, RejectRows#1386, IssuerId#8312, IssuerId2#8325, concat(IssuerId#8312, IssuerId2#8325) AS issueridcomposite#8901]

: : +- Project [BusinessDate#1378, stcd#7759, srcnm#7747, NetworkName#1381, NetworkURL#1382, custnum#1383, MarketCoverage#1384, DentalOnlyPlan#1385, RejectRows#1386, IssuerId#8312, IssuerId2#8325]

: : +- Project [IssuerIDDD#8273, BusinessDate#1378, stcd#7759, srcnm#7747, NetworkName#1381, NetworkURL#1382, custnum#1383, MarketCoverage#1384, DentalOnlyPlan#1385, RejectRows#1386, IssuerId#8312, IssuerId2#8325]

: : +- Project [IssuerIDD#8261, IssuerIDDD#8273, BusinessDate#1378, stcd#7759, srcnm#7747, NetworkName#1381, NetworkURL#1382, custnum#1383, MarketCoverage#1384, DentalOnlyPlan#1385, RejectRows#1386, IssuerId#8312, cast(IssuerIDDD#8273 as string) AS IssuerId2#8325]

: : +- Project [IssuerIDD#8261, IssuerIDDD#8273, BusinessDate#1378, stcd#7759, srcnm#7747, NetworkName#1381, NetworkURL#1382, custnum#1383, MarketCoverage#1384, DentalOnlyPlan#1385, RejectRows#1386, cast(IssuerIDD#8261 as string) AS IssuerId#8312]

: : +- Project [IssuerIDD#8261, IssuerId2#1377 AS IssuerIDDD#8273, BusinessDate#1378, stcd#7759, srcnm#7747, NetworkName#1381, NetworkURL#1382, custnum#1383, MarketCoverage#1384, DentalOnlyPlan#1385, RejectRows#1386]

: : +- Project [IssuerId#1376 AS IssuerIDD#8261, IssuerId2#1377, BusinessDate#1378, stcd#7759, srcnm#7747, NetworkName#1381, NetworkURL#1382, custnum#1383, MarketCoverage#1384, DentalOnlyPlan#1385, RejectRows#1386]

: : +- Project [IssuerId#1376, IssuerId2#1377, BusinessDate#1378, StateCode#1379 AS stcd#7759, srcnm#7747, NetworkName#1381, NetworkURL#1382, custnum#1383, MarketCoverage#1384, DentalOnlyPlan#1385, RejectRows#1386]

: : +- Project [IssuerId#1376, IssuerId2#1377, BusinessDate#1378, StateCode#1379, SourceName#1380 AS srcnm#7747, NetworkName#1381, NetworkURL#1382, custnum#1383, MarketCoverage#1384, DentalOnlyPlan#1385, RejectRows#1386]

: : +- Union false, false

: : :- Relation [IssuerId#1376,IssuerId2#1377,BusinessDate#1378,StateCode#1379,SourceName#1380,NetworkName#1381,NetworkURL#1382,custnum#1383,MarketCoverage#1384,DentalOnlyPlan#1385,RejectRows#1386] csv

: : +- Relation [IssuerId#491,IssuerId2#492,BusinessDate#493,StateCode#494,SourceName#495,NetworkName#496,NetworkURL#497,custnum#498,MarketCoverage#499,DentalOnlyPlan#500,RejectRows#501] csv

: +- SubqueryAlias r

: +- SubqueryAlias state\_table

: +- View (`state\_table`, [\_c0#26007,\_c1#26008])

: +- Project [\_c0#26007, \_c1#26008]

: +- Filter ((isnull(\_c2#26009) AND isnull(\_c3#26010)) AND isnull(\_c4#26011))

: +- Project [\_c0#26007, \_c1#26008, \_c2#26009, \_c3#26010, \_c4#26011]

: +- Relation [\_c0#26007,\_c1#26008,\_c2#26009,\_c3#26010,\_c4#26011] csv

+- SubqueryAlias r

+- SubqueryAlias cust\_table

+- View (`cust\_table`, [\_c0#52124,\_c1#52125,\_c2#52126,\_c3#52127,\_c4#52128])

+- Filter (((isnotnull(\_c0#52124) AND isnotnull(\_c1#52125)) AND isnotnull(\_c2#52126)) AND (isnotnull(\_c3#52127) AND isnotnull(\_c4#52128)))

+- Project [\_c0#52124, \_c1#52125, \_c2#52126, \_c3#52127, \_c4#52128]

+- Relation [\_c0#52124,\_c1#52125,\_c2#52126,\_c3#52127,\_c4#52128] csv

>>> second\_join.show(1)

+------------+----+-----+-----------+--------------------+-------+--------------+----------+--------+---------+-----------------+----------+--------------------+----------------+----------+--------------------+----------------+---+-------+---+-----+-------+---+-----+-------------+

|BusinessDate|stcd|srcnm|NetworkName| NetworkURL|custnum|MarketCoverage|RejectRows|IssuerId|IssuerId2|issueridcomposite| sysdt| systs|cleannetworkname| curdt| curts|cleannetworkname|\_c0| \_c1|\_c0| \_c1| \_c2|\_c3| \_c4|Protocol\_Used|

+------------+----+-----+-----------+--------------------+-------+--------------+----------+--------+---------+-----------------+----------+--------------------+----------------+----------+--------------------+----------------+---+-------+---+-----+-------+---+-----+-------------+

| 2019-10-01| AZ| HIOS| La Paz|http://uamarketpl...| 21| null| null| 88925| 88925| 8892588925|2024-07-22|2024-07-22 16:54:...| La Paz|2024-07-22|2024-07-22 16:54:...| La Paz| AZ|Arizona|779|Jason|Bowling| 21|Coach| http secured|

+------------+----+-----+-----------+--------------------+-------+--------------+----------+--------+---------+-----------------+----------+--------------------+----------------+----------+--------------------+----------------+---+-------+---+-----+-------+---+-----+-------------+

only showing top 1 row

>>> from pyspark.sql.functions import col, avg, count

>>> from pyspark.sql.window import Window

>>> from pyspark.sql.functions import row\_number

>>>

>>> # Calculate average age and count, and add seqno

... window\_spec = Window.partitionBy("stcd", "Protocol\_Used").orderBy(col("profession\_count").desc())

>>>

>>> count\_df = second\_join.groupBy("stcd", "Protocol\_Used", "profession") \

... .agg(

... avg("age").alias("avg\_age"),

... count("\*").alias("profession\_count")

... ) \

... .withColumn("seqno", row\_number().over(window\_spec))

Traceback (most recent call last):

File "<stdin>", line 4, in <module>

File "/usr/local/spark/python/pyspark/sql/group.py", line 119, in agg

\_to\_seq(self.sql\_ctx.\_sc, [c.\_jc for c in exprs[1:]]))

File "/usr/local/spark/python/lib/py4j-0.10.9.3-src.zip/py4j/java\_gateway.py", line 1322, in \_\_call\_\_

File "/usr/local/spark/python/pyspark/sql/utils.py", line 117, in deco

raise converted from None

pyspark.sql.utils.AnalysisException: cannot resolve 'profession' given input columns: [l.BusinessDate, l.IssuerId, l.IssuerId2, l.MarketCoverage, l.NetworkName, l.NetworkURL, r.Protocol\_Used, l.RejectRows, l.\_c0, l.\_c0, l.\_c1, l.\_c1, l.\_c2, l.\_c3, l.\_c4, l.cleannetworkname, l.cleannetworkname, l.curdt, l.curts, l.custnum, l.issueridcomposite, l.srcnm, l.stcd, l.sysdt, l.systs];

'Aggregate [stcd#7759, Protocol\_Used#48671, 'profession], [stcd#7759, Protocol\_Used#48671, 'profession, avg('age) AS avg\_age#58285, count(1) AS profession\_count#58287L]

+- Project [BusinessDate#1378, stcd#7759, srcnm#7747, NetworkName#1381, NetworkURL#1382, custnum#1383, MarketCoverage#1384, RejectRows#1386, IssuerId#8312, IssuerId2#8325, issueridcomposite#8901, sysdt#9003, systs#9016, cleannetworkname#47665, curdt#48091, curts#48092, cleannetworkname#48111, \_c0#26007, \_c1#26008, \_c0#52124, \_c1#52125, \_c2#52126, \_c3#52127, \_c4#52128, Protocol\_Used#48671]

+- Join Inner

:- SubqueryAlias l

: +- SubqueryAlias view1

: +- View (`view1`, [BusinessDate#1378,stcd#7759,srcnm#7747,NetworkName#1381,NetworkURL#1382,custnum#1383,MarketCoverage#1384,RejectRows#1386,IssuerId#8312,IssuerId2#8325,issueridcomposite#8901,sysdt#9003,systs#9016,cleannetworkname#47665,curdt#48091,curts#48092,cleannetworkname#48111,\_c0#26007,\_c1#26008,\_c0#52124,\_c1#52125,\_c2#52126,\_c3#52127,\_c4#52128])

: +- Project [BusinessDate#1378, stcd#7759, srcnm#7747, NetworkName#1381, NetworkURL#1382, custnum#1383, MarketCoverage#1384, RejectRows#1386, IssuerId#8312, IssuerId2#8325, issueridcomposite#8901, sysdt#9003, systs#9016, cleannetworkname#47665, curdt#48091, curts#48092, cleannetworkname#48111, \_c0#26007, \_c1#26008, \_c0#52124, \_c1#52125, \_c2#52126, \_c3#52127, \_c4#52128]

: +- Join Inner, (cast(custnum#1383 as int) = \_c3#52127)

: :- SubqueryAlias l

: : +- SubqueryAlias insureview

: : +- View (`insureview`, [BusinessDate#1378,stcd#7759,srcnm#7747,NetworkName#1381,NetworkURL#1382,custnum#1383,MarketCoverage#1384,RejectRows#1386,IssuerId#8312,IssuerId2#8325,issueridcomposite#8901,sysdt#9003,systs#9016,cleannetworkname#47665,curdt#48091,curts#48092,cleannetworkname#48111,\_c0#26007,\_c1#26008])

: : +- Project [BusinessDate#1378, stcd#7759, srcnm#7747, NetworkName#1381, NetworkURL#1382, custnum#1383, MarketCoverage#1384, RejectRows#1386, IssuerId#8312, IssuerId2#8325, issueridcomposite#8901, sysdt#9003, systs#9016, cleannetworkname#47665, curdt#48091, curts#48092, cleannetworkname#48111, \_c0#26007, \_c1#26008]

: : +- Join Inner, (stcd#7759 = \_c0#26007)

: : :- SubqueryAlias l

: : : +- SubqueryAlias insureview

: : : +- View (`insureview`, [BusinessDate#1378,stcd#7759,srcnm#7747,NetworkName#1381,NetworkURL#1382,custnum#1383,MarketCoverage#1384,RejectRows#1386,IssuerId#8312,IssuerId2#8325,issueridcomposite#8901,sysdt#9003,systs#9016,cleannetworkname#47665,curdt#48091,curts#48092,cleannetworkname#48111])

: : : +- Project [BusinessDate#1378, stcd#7759, srcnm#7747, NetworkName#1381, NetworkURL#1382, custnum#1383, MarketCoverage#1384, RejectRows#1386, IssuerId#8312, IssuerId2#8325, issueridcomposite#8901, sysdt#9003, systs#9016, cleannetworkname#47665, curdt#48091, curts#48092, sql\_function(NetworkName#1381) AS cleannetworkname#48111]

: : : +- SubqueryAlias insureview

: : : +- View (`insureview`, [BusinessDate#1378,stcd#7759,srcnm#7747,NetworkName#1381,NetworkURL#1382,custnum#1383,MarketCoverage#1384,RejectRows#1386,IssuerId#8312,IssuerId2#8325,issueridcomposite#8901,sysdt#9003,systs#9016,cleannetworkname#47665,curdt#48091,curts#48092])

: : : +- Project [BusinessDate#1378, stcd#7759, srcnm#7747, NetworkName#1381, NetworkURL#1382, custnum#1383, MarketCoverage#1384, RejectRows#1386, IssuerId#8312, IssuerId2#8325, issueridcomposite#8901, sysdt#9003, systs#9016, cleannetworkname#47665, current\_date(Some(Asia/Kolkata)) AS curdt#48091, current\_timestamp() AS curts#48092]

: : : +- SubqueryAlias insureview

: : : +- View (`insureview`, [BusinessDate#1378,stcd#7759,srcnm#7747,NetworkName#1381,NetworkURL#1382,custnum#1383,MarketCoverage#1384,RejectRows#1386,IssuerId#8312,IssuerId2#8325,issueridcomposite#8901,sysdt#9003,systs#9016,cleannetworkname#47665])

: : : +- Project [BusinessDate#1378, stcd#7759, srcnm#7747, NetworkName#1381, NetworkURL#1382, custnum#1383, MarketCoverage#1384, RejectRows#1386, IssuerId#8312, IssuerId2#8325, issueridcomposite#8901, sysdt#9003, systs#9016, sql\_function(NetworkName#1381) AS cleannetworkname#47665]

: : : +- SubqueryAlias insureview

: : : +- View (`insureview`, [BusinessDate#1378,stcd#7759,srcnm#7747,NetworkName#1381,NetworkURL#1382,custnum#1383,MarketCoverage#1384,RejectRows#1386,IssuerId#8312,IssuerId2#8325,issueridcomposite#8901,sysdt#9003,systs#9016])

: : : +- Project [BusinessDate#1378, stcd#7759, srcnm#7747, NetworkName#1381, NetworkURL#1382, custnum#1383, MarketCoverage#1384, RejectRows#1386, IssuerId#8312, IssuerId2#8325, issueridcomposite#8901, sysdt#9003, current\_timestamp() AS systs#9016]

: : : +- Project [BusinessDate#1378, stcd#7759, srcnm#7747, NetworkName#1381, NetworkURL#1382, custnum#1383, MarketCoverage#1384, RejectRows#1386, IssuerId#8312, IssuerId2#8325, issueridcomposite#8901, current\_date(Some(Asia/Kolkata)) AS sysdt#9003]

: : : +- Project [BusinessDate#1378, stcd#7759, srcnm#7747, NetworkName#1381, NetworkURL#1382, custnum#1383, MarketCoverage#1384, RejectRows#1386, IssuerId#8312, IssuerId2#8325, issueridcomposite#8901]

: : : +- Project [BusinessDate#1378, stcd#7759, srcnm#7747, NetworkName#1381, NetworkURL#1382, custnum#1383, MarketCoverage#1384, DentalOnlyPlan#1385, RejectRows#1386, IssuerId#8312, IssuerId2#8325, concat(IssuerId#8312, IssuerId2#8325) AS issueridcomposite#8901]

: : : +- Project [BusinessDate#1378, stcd#7759, srcnm#7747, NetworkName#1381, NetworkURL#1382, custnum#1383, MarketCoverage#1384, DentalOnlyPlan#1385, RejectRows#1386, IssuerId#8312, IssuerId2#8325]

: : : +- Project [IssuerIDDD#8273, BusinessDate#1378, stcd#7759, srcnm#7747, NetworkName#1381, NetworkURL#1382, custnum#1383, MarketCoverage#1384, DentalOnlyPlan#1385, RejectRows#1386, IssuerId#8312, IssuerId2#8325]

: : : +- Project [IssuerIDD#8261, IssuerIDDD#8273, BusinessDate#1378, stcd#7759, srcnm#7747, NetworkName#1381, NetworkURL#1382, custnum#1383, MarketCoverage#1384, DentalOnlyPlan#1385, RejectRows#1386, IssuerId#8312, cast(IssuerIDDD#8273 as string) AS IssuerId2#8325]

: : : +- Project [IssuerIDD#8261, IssuerIDDD#8273, BusinessDate#1378, stcd#7759, srcnm#7747, NetworkName#1381, NetworkURL#1382, custnum#1383, MarketCoverage#1384, DentalOnlyPlan#1385, RejectRows#1386, cast(IssuerIDD#8261 as string) AS IssuerId#8312]

: : : +- Project [IssuerIDD#8261, IssuerId2#1377 AS IssuerIDDD#8273, BusinessDate#1378, stcd#7759, srcnm#7747, NetworkName#1381, NetworkURL#1382, custnum#1383, MarketCoverage#1384, DentalOnlyPlan#1385, RejectRows#1386]

: : : +- Project [IssuerId#1376 AS IssuerIDD#8261, IssuerId2#1377, BusinessDate#1378, stcd#7759, srcnm#7747, NetworkName#1381, NetworkURL#1382, custnum#1383, MarketCoverage#1384, DentalOnlyPlan#1385, RejectRows#1386]

: : : +- Project [IssuerId#1376, IssuerId2#1377, BusinessDate#1378, StateCode#1379 AS stcd#7759, srcnm#7747, NetworkName#1381, NetworkURL#1382, custnum#1383, MarketCoverage#1384, DentalOnlyPlan#1385, RejectRows#1386]

: : : +- Project [IssuerId#1376, IssuerId2#1377, BusinessDate#1378, StateCode#1379, SourceName#1380 AS srcnm#7747, NetworkName#1381, NetworkURL#1382, custnum#1383, MarketCoverage#1384, DentalOnlyPlan#1385, RejectRows#1386]

: : : +- Union false, false

: : : :- Relation [IssuerId#1376,IssuerId2#1377,BusinessDate#1378,StateCode#1379,SourceName#1380,NetworkName#1381,NetworkURL#1382,custnum#1383,MarketCoverage#1384,DentalOnlyPlan#1385,RejectRows#1386] csv

: : : +- Relation [IssuerId#491,IssuerId2#492,BusinessDate#493,StateCode#494,SourceName#495,NetworkName#496,NetworkURL#497,custnum#498,MarketCoverage#499,DentalOnlyPlan#500,RejectRows#501] csv

: : +- SubqueryAlias r

: : +- SubqueryAlias state\_table

: : +- View (`state\_table`, [\_c0#26007,\_c1#26008])

: : +- Project [\_c0#26007, \_c1#26008]

: : +- Filter ((isnull(\_c2#26009) AND isnull(\_c3#26010)) AND isnull(\_c4#26011))

: : +- Project [\_c0#26007, \_c1#26008, \_c2#26009, \_c3#26010, \_c4#26011]

: : +- Relation [\_c0#26007,\_c1#26008,\_c2#26009,\_c3#26010,\_c4#26011] csv

: +- SubqueryAlias r

: +- SubqueryAlias cust\_table

: +- View (`cust\_table`, [\_c0#52124,\_c1#52125,\_c2#52126,\_c3#52127,\_c4#52128])

: +- Filter (((isnotnull(\_c0#52124) AND isnotnull(\_c1#52125)) AND isnotnull(\_c2#52126)) AND (isnotnull(\_c3#52127) AND isnotnull(\_c4#52128)))

: +- Project [\_c0#52124, \_c1#52125, \_c2#52126, \_c3#52127, \_c4#52128]

: +- Relation [\_c0#52124,\_c1#52125,\_c2#52126,\_c3#52127,\_c4#52128] csv

+- SubqueryAlias r

+- SubqueryAlias view2

+- View (`view2`, [BusinessDate#57031,stcd#57060,srcnm#57061,NetworkName#57034,NetworkURL#57035,custnum#57036,MarketCoverage#57037,RejectRows#57039,IssuerId#57059,IssuerId2#57058,issueridcomposite#57057,sysdt#57056,systs#57055,cleannetworkname#57054,curdt#57052,curts#57053,cleannetworkname#57051,Protocol\_Used#48671])

+- Project [BusinessDate#57031, stcd#57060, srcnm#57061, NetworkName#57034, NetworkURL#57035, custnum#57036, MarketCoverage#57037, RejectRows#57039, IssuerId#57059, IssuerId2#57058, issueridcomposite#57057, sysdt#57056, systs#57055, cleannetworkname#57054, curdt#57052, curts#57053, cleannetworkname#57051, CASE WHEN NetworkURL#57035 LIKE https://% THEN http secured WHEN NetworkURL#57035 LIKE http://% THEN http non secured ELSE No Protocol END AS Protocol\_Used#48671]

+- SubqueryAlias insureview

+- View (`insureview`, [BusinessDate#57031,stcd#57060,srcnm#57061,NetworkName#57034,NetworkURL#57035,custnum#57036,MarketCoverage#57037,RejectRows#57039,IssuerId#57059,IssuerId2#57058,issueridcomposite#57057,sysdt#57056,systs#57055,cleannetworkname#57054,curdt#57052,curts#57053,cleannetworkname#57051])

+- Project [BusinessDate#57031, stcd#57060, srcnm#57061, NetworkName#57034, NetworkURL#57035, custnum#57036, MarketCoverage#57037, RejectRows#57039, IssuerId#57059, IssuerId2#57058, issueridcomposite#57057, sysdt#57056, systs#57055, cleannetworkname#57054, curdt#57052, curts#57053, sql\_function(NetworkName#57034) AS cleannetworkname#57051]

+- SubqueryAlias insureview

+- View (`insureview`, [BusinessDate#57031,stcd#57060,srcnm#57061,NetworkName#57034,NetworkURL#57035,custnum#57036,MarketCoverage#57037,RejectRows#57039,IssuerId#57059,IssuerId2#57058,issueridcomposite#57057,sysdt#57056,systs#57055,cleannetworkname#57054,curdt#57052,curts#57053])

+- Project [BusinessDate#57031, stcd#57060, srcnm#57061, NetworkName#57034, NetworkURL#57035, custnum#57036, MarketCoverage#57037, RejectRows#57039, IssuerId#57059, IssuerId2#57058, issueridcomposite#57057, sysdt#57056, systs#57055, cleannetworkname#57054, current\_date(Some(Asia/Kolkata)) AS curdt#57052, current\_timestamp() AS curts#57053]

+- SubqueryAlias insureview

+- View (`insureview`, [BusinessDate#57031,stcd#57060,srcnm#57061,NetworkName#57034,NetworkURL#57035,custnum#57036,MarketCoverage#57037,RejectRows#57039,IssuerId#57059,IssuerId2#57058,issueridcomposite#57057,sysdt#57056,systs#57055,cleannetworkname#57054])

+- Project [BusinessDate#57031, stcd#57060, srcnm#57061, NetworkName#57034, NetworkURL#57035, custnum#57036, MarketCoverage#57037, RejectRows#57039, IssuerId#57059, IssuerId2#57058, issueridcomposite#57057, sysdt#57056, systs#57055, sql\_function(NetworkName#57034) AS cleannetworkname#57054]

+- SubqueryAlias insureview

+- View (`insureview`, [BusinessDate#57031,stcd#57060,srcnm#57061,NetworkName#57034,NetworkURL#57035,custnum#57036,MarketCoverage#57037,RejectRows#57039,IssuerId#57059,IssuerId2#57058,issueridcomposite#57057,sysdt#57056,systs#57055])

+- Project [BusinessDate#57031, stcd#57060, srcnm#57061, NetworkName#57034, NetworkURL#57035, custnum#57036, MarketCoverage#57037, RejectRows#57039, IssuerId#57059, IssuerId2#57058, issueridcomposite#57057, sysdt#57056, current\_timestamp() AS systs#57055]

+- Project [BusinessDate#57031, stcd#57060, srcnm#57061, NetworkName#57034, NetworkURL#57035, custnum#57036, MarketCoverage#57037, RejectRows#57039, IssuerId#57059, IssuerId2#57058, issueridcomposite#57057, current\_date(Some(Asia/Kolkata)) AS sysdt#57056]

+- Project [BusinessDate#57031, stcd#57060, srcnm#57061, NetworkName#57034, NetworkURL#57035, custnum#57036, MarketCoverage#57037, RejectRows#57039, IssuerId#57059, IssuerId2#57058, issueridcomposite#57057]

+- Project [BusinessDate#57031, stcd#57060, srcnm#57061, NetworkName#57034, NetworkURL#57035, custnum#57036, MarketCoverage#57037, DentalOnlyPlan#57038, RejectRows#57039, IssuerId#57059, IssuerId2#57058, concat(IssuerId#57059, IssuerId2#57058) AS issueridcomposite#57057]

+- Project [BusinessDate#57031, stcd#57060, srcnm#57061, NetworkName#57034, NetworkURL#57035, custnum#57036, MarketCoverage#57037, DentalOnlyPlan#57038, RejectRows#57039, IssuerId#57059, IssuerId2#57058]

+- Project [IssuerIDDD#8273, BusinessDate#57031, stcd#57060, srcnm#57061, NetworkName#57034, NetworkURL#57035, custnum#57036, MarketCoverage#57037, DentalOnlyPlan#57038, RejectRows#57039, IssuerId#57059, IssuerId2#57058]

+- Project [IssuerIDD#8261, IssuerIDDD#8273, BusinessDate#57031, stcd#57060, srcnm#57061, NetworkName#57034, NetworkURL#57035, custnum#57036, MarketCoverage#57037, DentalOnlyPlan#57038, RejectRows#57039, IssuerId#57059, cast(IssuerIDDD#8273 as string) AS IssuerId2#57058]

+- Project [IssuerIDD#8261, IssuerIDDD#8273, BusinessDate#57031, stcd#57060, srcnm#57061, NetworkName#57034, NetworkURL#57035, custnum#57036, MarketCoverage#57037, DentalOnlyPlan#57038, RejectRows#57039, cast(IssuerIDD#8261 as string) AS IssuerId#57059]

+- Project [IssuerIDD#8261, IssuerId2#57030 AS IssuerIDDD#8273, BusinessDate#57031, stcd#57060, srcnm#57061, NetworkName#57034, NetworkURL#57035, custnum#57036, MarketCoverage#57037, DentalOnlyPlan#57038, RejectRows#57039]

+- Project [IssuerId#57029 AS IssuerIDD#8261, IssuerId2#57030, BusinessDate#57031, stcd#57060, srcnm#57061, NetworkName#57034, NetworkURL#57035, custnum#57036, MarketCoverage#57037, DentalOnlyPlan#57038, RejectRows#57039]

+- Project [IssuerId#57029, IssuerId2#57030, BusinessDate#57031, StateCode#57032 AS stcd#57060, srcnm#57061, NetworkName#57034, NetworkURL#57035, custnum#57036, MarketCoverage#57037, DentalOnlyPlan#57038, RejectRows#57039]

+- Project [IssuerId#57029, IssuerId2#57030, BusinessDate#57031, StateCode#57032, SourceName#57033 AS srcnm#57061, NetworkName#57034, NetworkURL#57035, custnum#57036, MarketCoverage#57037, DentalOnlyPlan#57038, RejectRows#57039]

+- Union false, false

:- Relation [IssuerId#57029,IssuerId2#57030,BusinessDate#57031,StateCode#57032,SourceName#57033,NetworkName#57034,NetworkURL#57035,custnum#57036,MarketCoverage#57037,DentalOnlyPlan#57038,RejectRows#57039] csv

+- Relation [IssuerId#57040,IssuerId2#57041,BusinessDate#57042,StateCode#57043,SourceName#57044,NetworkName#57045,NetworkURL#57046,custnum#57047,MarketCoverage#57048,DentalOnlyPlan#57049,RejectRows#57050] csv

>>> count\_df = second\_join.groupBy("stcd", "Protocol\_Used", "profession") \

... .agg(

... avg("age").alias("avg\_age"),

... count("\*").alias("profession\_count")

... ) \

... .withColumn("seqno", row\_number().over(window\_spec))sdbksdv

File "<stdin>", line 6

.withColumn("seqno", row\_number().over(window\_spec))sdbksdv

^

SyntaxError: invalid syntax

>>> count\_df = second\_join.groupBy("stcd", "Protocol\_Used", "profession") \

... .agg(

... avg("\_c1").alias("avg\_age"),

... count("\*").alias("profession\_count")

... ) \

... .withColumn("seqno", row\_number().over(window\_spec))

Traceback (most recent call last):

File "<stdin>", line 4, in <module>

File "/usr/local/spark/python/pyspark/sql/group.py", line 119, in agg

\_to\_seq(self.sql\_ctx.\_sc, [c.\_jc for c in exprs[1:]]))

File "/usr/local/spark/python/lib/py4j-0.10.9.3-src.zip/py4j/java\_gateway.py", line 1322, in \_\_call\_\_

File "/usr/local/spark/python/pyspark/sql/utils.py", line 117, in deco

raise converted from None

pyspark.sql.utils.AnalysisException: Reference '\_c1' is ambiguous, could be: l.\_c1, l.\_c1.

>>> count\_df = second\_join.groupBy("stcd", "Protocol\_Used", "profession") \

... .agg(

... avg("\_c3").alias("avg\_age"),

... count("\*").alias("profession\_count")

... ) \

... .withColumn("seqno", row\_number().over(window\_spec))

Traceback (most recent call last):

File "<stdin>", line 4, in <module>

File "/usr/local/spark/python/pyspark/sql/group.py", line 119, in agg

\_to\_seq(self.sql\_ctx.\_sc, [c.\_jc for c in exprs[1:]]))

File "/usr/local/spark/python/lib/py4j-0.10.9.3-src.zip/py4j/java\_gateway.py", line 1322, in \_\_call\_\_

File "/usr/local/spark/python/pyspark/sql/utils.py", line 117, in deco

raise converted from None

pyspark.sql.utils.AnalysisException: cannot resolve 'profession' given input columns: [l.BusinessDate, l.IssuerId, l.IssuerId2, l.MarketCoverage, l.NetworkName, l.NetworkURL, r.Protocol\_Used, l.RejectRows, l.\_c0, l.\_c0, l.\_c1, l.\_c1, l.\_c2, l.\_c3, l.\_c4, l.cleannetworkname, l.cleannetworkname, l.curdt, l.curts, l.custnum, l.issueridcomposite, l.srcnm, l.stcd, l.sysdt, l.systs];

'Aggregate [stcd#7759, Protocol\_Used#48671, 'profession], [stcd#7759, Protocol\_Used#48671, 'profession, avg(\_c3#52127) AS avg\_age#58343, count(1) AS profession\_count#58345L]

+- Project [BusinessDate#1378, stcd#7759, srcnm#7747, NetworkName#1381, NetworkURL#1382, custnum#1383, MarketCoverage#1384, RejectRows#1386, IssuerId#8312, IssuerId2#8325, issueridcomposite#8901, sysdt#9003, systs#9016, cleannetworkname#47665, curdt#48091, curts#48092, cleannetworkname#48111, \_c0#26007, \_c1#26008, \_c0#52124, \_c1#52125, \_c2#52126, \_c3#52127, \_c4#52128, Protocol\_Used#48671]

+- Join Inner

:- SubqueryAlias l

: +- SubqueryAlias view1

: +- View (`view1`, [BusinessDate#1378,stcd#7759,srcnm#7747,NetworkName#1381,NetworkURL#1382,custnum#1383,MarketCoverage#1384,RejectRows#1386,IssuerId#8312,IssuerId2#8325,issueridcomposite#8901,sysdt#9003,systs#9016,cleannetworkname#47665,curdt#48091,curts#48092,cleannetworkname#48111,\_c0#26007,\_c1#26008,\_c0#52124,\_c1#52125,\_c2#52126,\_c3#52127,\_c4#52128])

: +- Project [BusinessDate#1378, stcd#7759, srcnm#7747, NetworkName#1381, NetworkURL#1382, custnum#1383, MarketCoverage#1384, RejectRows#1386, IssuerId#8312, IssuerId2#8325, issueridcomposite#8901, sysdt#9003, systs#9016, cleannetworkname#47665, curdt#48091, curts#48092, cleannetworkname#48111, \_c0#26007, \_c1#26008, \_c0#52124, \_c1#52125, \_c2#52126, \_c3#52127, \_c4#52128]

: +- Join Inner, (cast(custnum#1383 as int) = \_c3#52127)

: :- SubqueryAlias l

: : +- SubqueryAlias insureview

: : +- View (`insureview`, [BusinessDate#1378,stcd#7759,srcnm#7747,NetworkName#1381,NetworkURL#1382,custnum#1383,MarketCoverage#1384,RejectRows#1386,IssuerId#8312,IssuerId2#8325,issueridcomposite#8901,sysdt#9003,systs#9016,cleannetworkname#47665,curdt#48091,curts#48092,cleannetworkname#48111,\_c0#26007,\_c1#26008])

: : +- Project [BusinessDate#1378, stcd#7759, srcnm#7747, NetworkName#1381, NetworkURL#1382, custnum#1383, MarketCoverage#1384, RejectRows#1386, IssuerId#8312, IssuerId2#8325, issueridcomposite#8901, sysdt#9003, systs#9016, cleannetworkname#47665, curdt#48091, curts#48092, cleannetworkname#48111, \_c0#26007, \_c1#26008]

: : +- Join Inner, (stcd#7759 = \_c0#26007)

: : :- SubqueryAlias l

: : : +- SubqueryAlias insureview

: : : +- View (`insureview`, [BusinessDate#1378,stcd#7759,srcnm#7747,NetworkName#1381,NetworkURL#1382,custnum#1383,MarketCoverage#1384,RejectRows#1386,IssuerId#8312,IssuerId2#8325,issueridcomposite#8901,sysdt#9003,systs#9016,cleannetworkname#47665,curdt#48091,curts#48092,cleannetworkname#48111])

: : : +- Project [BusinessDate#1378, stcd#7759, srcnm#7747, NetworkName#1381, NetworkURL#1382, custnum#1383, MarketCoverage#1384, RejectRows#1386, IssuerId#8312, IssuerId2#8325, issueridcomposite#8901, sysdt#9003, systs#9016, cleannetworkname#47665, curdt#48091, curts#48092, sql\_function(NetworkName#1381) AS cleannetworkname#48111]

: : : +- SubqueryAlias insureview

: : : +- View (`insureview`, [BusinessDate#1378,stcd#7759,srcnm#7747,NetworkName#1381,NetworkURL#1382,custnum#1383,MarketCoverage#1384,RejectRows#1386,IssuerId#8312,IssuerId2#8325,issueridcomposite#8901,sysdt#9003,systs#9016,cleannetworkname#47665,curdt#48091,curts#48092])

: : : +- Project [BusinessDate#1378, stcd#7759, srcnm#7747, NetworkName#1381, NetworkURL#1382, custnum#1383, MarketCoverage#1384, RejectRows#1386, IssuerId#8312, IssuerId2#8325, issueridcomposite#8901, sysdt#9003, systs#9016, cleannetworkname#47665, current\_date(Some(Asia/Kolkata)) AS curdt#48091, current\_timestamp() AS curts#48092]

: : : +- SubqueryAlias insureview

: : : +- View (`insureview`, [BusinessDate#1378,stcd#7759,srcnm#7747,NetworkName#1381,NetworkURL#1382,custnum#1383,MarketCoverage#1384,RejectRows#1386,IssuerId#8312,IssuerId2#8325,issueridcomposite#8901,sysdt#9003,systs#9016,cleannetworkname#47665])

: : : +- Project [BusinessDate#1378, stcd#7759, srcnm#7747, NetworkName#1381, NetworkURL#1382, custnum#1383, MarketCoverage#1384, RejectRows#1386, IssuerId#8312, IssuerId2#8325, issueridcomposite#8901, sysdt#9003, systs#9016, sql\_function(NetworkName#1381) AS cleannetworkname#47665]

: : : +- SubqueryAlias insureview

: : : +- View (`insureview`, [BusinessDate#1378,stcd#7759,srcnm#7747,NetworkName#1381,NetworkURL#1382,custnum#1383,MarketCoverage#1384,RejectRows#1386,IssuerId#8312,IssuerId2#8325,issueridcomposite#8901,sysdt#9003,systs#9016])

: : : +- Project [BusinessDate#1378, stcd#7759, srcnm#7747, NetworkName#1381, NetworkURL#1382, custnum#1383, MarketCoverage#1384, RejectRows#1386, IssuerId#8312, IssuerId2#8325, issueridcomposite#8901, sysdt#9003, current\_timestamp() AS systs#9016]

: : : +- Project [BusinessDate#1378, stcd#7759, srcnm#7747, NetworkName#1381, NetworkURL#1382, custnum#1383, MarketCoverage#1384, RejectRows#1386, IssuerId#8312, IssuerId2#8325, issueridcomposite#8901, current\_date(Some(Asia/Kolkata)) AS sysdt#9003]

: : : +- Project [BusinessDate#1378, stcd#7759, srcnm#7747, NetworkName#1381, NetworkURL#1382, custnum#1383, MarketCoverage#1384, RejectRows#1386, IssuerId#8312, IssuerId2#8325, issueridcomposite#8901]

: : : +- Project [BusinessDate#1378, stcd#7759, srcnm#7747, NetworkName#1381, NetworkURL#1382, custnum#1383, MarketCoverage#1384, DentalOnlyPlan#1385, RejectRows#1386, IssuerId#8312, IssuerId2#8325, concat(IssuerId#8312, IssuerId2#8325) AS issueridcomposite#8901]

: : : +- Project [BusinessDate#1378, stcd#7759, srcnm#7747, NetworkName#1381, NetworkURL#1382, custnum#1383, MarketCoverage#1384, DentalOnlyPlan#1385, RejectRows#1386, IssuerId#8312, IssuerId2#8325]

: : : +- Project [IssuerIDDD#8273, BusinessDate#1378, stcd#7759, srcnm#7747, NetworkName#1381, NetworkURL#1382, custnum#1383, MarketCoverage#1384, DentalOnlyPlan#1385, RejectRows#1386, IssuerId#8312, IssuerId2#8325]

: : : +- Project [IssuerIDD#8261, IssuerIDDD#8273, BusinessDate#1378, stcd#7759, srcnm#7747, NetworkName#1381, NetworkURL#1382, custnum#1383, MarketCoverage#1384, DentalOnlyPlan#1385, RejectRows#1386, IssuerId#8312, cast(IssuerIDDD#8273 as string) AS IssuerId2#8325]

: : : +- Project [IssuerIDD#8261, IssuerIDDD#8273, BusinessDate#1378, stcd#7759, srcnm#7747, NetworkName#1381, NetworkURL#1382, custnum#1383, MarketCoverage#1384, DentalOnlyPlan#1385, RejectRows#1386, cast(IssuerIDD#8261 as string) AS IssuerId#8312]

: : : +- Project [IssuerIDD#8261, IssuerId2#1377 AS IssuerIDDD#8273, BusinessDate#1378, stcd#7759, srcnm#7747, NetworkName#1381, NetworkURL#1382, custnum#1383, MarketCoverage#1384, DentalOnlyPlan#1385, RejectRows#1386]

: : : +- Project [IssuerId#1376 AS IssuerIDD#8261, IssuerId2#1377, BusinessDate#1378, stcd#7759, srcnm#7747, NetworkName#1381, NetworkURL#1382, custnum#1383, MarketCoverage#1384, DentalOnlyPlan#1385, RejectRows#1386]

: : : +- Project [IssuerId#1376, IssuerId2#1377, BusinessDate#1378, StateCode#1379 AS stcd#7759, srcnm#7747, NetworkName#1381, NetworkURL#1382, custnum#1383, MarketCoverage#1384, DentalOnlyPlan#1385, RejectRows#1386]

: : : +- Project [IssuerId#1376, IssuerId2#1377, BusinessDate#1378, StateCode#1379, SourceName#1380 AS srcnm#7747, NetworkName#1381, NetworkURL#1382, custnum#1383, MarketCoverage#1384, DentalOnlyPlan#1385, RejectRows#1386]

: : : +- Union false, false

: : : :- Relation [IssuerId#1376,IssuerId2#1377,BusinessDate#1378,StateCode#1379,SourceName#1380,NetworkName#1381,NetworkURL#1382,custnum#1383,MarketCoverage#1384,DentalOnlyPlan#1385,RejectRows#1386] csv

: : : +- Relation [IssuerId#491,IssuerId2#492,BusinessDate#493,StateCode#494,SourceName#495,NetworkName#496,NetworkURL#497,custnum#498,MarketCoverage#499,DentalOnlyPlan#500,RejectRows#501] csv

: : +- SubqueryAlias r

: : +- SubqueryAlias state\_table

: : +- View (`state\_table`, [\_c0#26007,\_c1#26008])

: : +- Project [\_c0#26007, \_c1#26008]

: : +- Filter ((isnull(\_c2#26009) AND isnull(\_c3#26010)) AND isnull(\_c4#26011))

: : +- Project [\_c0#26007, \_c1#26008, \_c2#26009, \_c3#26010, \_c4#26011]

: : +- Relation [\_c0#26007,\_c1#26008,\_c2#26009,\_c3#26010,\_c4#26011] csv

: +- SubqueryAlias r

: +- SubqueryAlias cust\_table

: +- View (`cust\_table`, [\_c0#52124,\_c1#52125,\_c2#52126,\_c3#52127,\_c4#52128])

: +- Filter (((isnotnull(\_c0#52124) AND isnotnull(\_c1#52125)) AND isnotnull(\_c2#52126)) AND (isnotnull(\_c3#52127) AND isnotnull(\_c4#52128)))

: +- Project [\_c0#52124, \_c1#52125, \_c2#52126, \_c3#52127, \_c4#52128]

: +- Relation [\_c0#52124,\_c1#52125,\_c2#52126,\_c3#52127,\_c4#52128] csv

+- SubqueryAlias r

+- SubqueryAlias view2

+- View (`view2`, [BusinessDate#57031,stcd#57060,srcnm#57061,NetworkName#57034,NetworkURL#57035,custnum#57036,MarketCoverage#57037,RejectRows#57039,IssuerId#57059,IssuerId2#57058,issueridcomposite#57057,sysdt#57056,systs#57055,cleannetworkname#57054,curdt#57052,curts#57053,cleannetworkname#57051,Protocol\_Used#48671])

+- Project [BusinessDate#57031, stcd#57060, srcnm#57061, NetworkName#57034, NetworkURL#57035, custnum#57036, MarketCoverage#57037, RejectRows#57039, IssuerId#57059, IssuerId2#57058, issueridcomposite#57057, sysdt#57056, systs#57055, cleannetworkname#57054, curdt#57052, curts#57053, cleannetworkname#57051, CASE WHEN NetworkURL#57035 LIKE https://% THEN http secured WHEN NetworkURL#57035 LIKE http://% THEN http non secured ELSE No Protocol END AS Protocol\_Used#48671]

+- SubqueryAlias insureview

+- View (`insureview`, [BusinessDate#57031,stcd#57060,srcnm#57061,NetworkName#57034,NetworkURL#57035,custnum#57036,MarketCoverage#57037,RejectRows#57039,IssuerId#57059,IssuerId2#57058,issueridcomposite#57057,sysdt#57056,systs#57055,cleannetworkname#57054,curdt#57052,curts#57053,cleannetworkname#57051])

+- Project [BusinessDate#57031, stcd#57060, srcnm#57061, NetworkName#57034, NetworkURL#57035, custnum#57036, MarketCoverage#57037, RejectRows#57039, IssuerId#57059, IssuerId2#57058, issueridcomposite#57057, sysdt#57056, systs#57055, cleannetworkname#57054, curdt#57052, curts#57053, sql\_function(NetworkName#57034) AS cleannetworkname#57051]

+- SubqueryAlias insureview

+- View (`insureview`, [BusinessDate#57031,stcd#57060,srcnm#57061,NetworkName#57034,NetworkURL#57035,custnum#57036,MarketCoverage#57037,RejectRows#57039,IssuerId#57059,IssuerId2#57058,issueridcomposite#57057,sysdt#57056,systs#57055,cleannetworkname#57054,curdt#57052,curts#57053])

+- Project [BusinessDate#57031, stcd#57060, srcnm#57061, NetworkName#57034, NetworkURL#57035, custnum#57036, MarketCoverage#57037, RejectRows#57039, IssuerId#57059, IssuerId2#57058, issueridcomposite#57057, sysdt#57056, systs#57055, cleannetworkname#57054, current\_date(Some(Asia/Kolkata)) AS curdt#57052, current\_timestamp() AS curts#57053]

+- SubqueryAlias insureview

+- View (`insureview`, [BusinessDate#57031,stcd#57060,srcnm#57061,NetworkName#57034,NetworkURL#57035,custnum#57036,MarketCoverage#57037,RejectRows#57039,IssuerId#57059,IssuerId2#57058,issueridcomposite#57057,sysdt#57056,systs#57055,cleannetworkname#57054])

+- Project [BusinessDate#57031, stcd#57060, srcnm#57061, NetworkName#57034, NetworkURL#57035, custnum#57036, MarketCoverage#57037, RejectRows#57039, IssuerId#57059, IssuerId2#57058, issueridcomposite#57057, sysdt#57056, systs#57055, sql\_function(NetworkName#57034) AS cleannetworkname#57054]

+- SubqueryAlias insureview

+- View (`insureview`, [BusinessDate#57031,stcd#57060,srcnm#57061,NetworkName#57034,NetworkURL#57035,custnum#57036,MarketCoverage#57037,RejectRows#57039,IssuerId#57059,IssuerId2#57058,issueridcomposite#57057,sysdt#57056,systs#57055])

+- Project [BusinessDate#57031, stcd#57060, srcnm#57061, NetworkName#57034, NetworkURL#57035, custnum#57036, MarketCoverage#57037, RejectRows#57039, IssuerId#57059, IssuerId2#57058, issueridcomposite#57057, sysdt#57056, current\_timestamp() AS systs#57055]

+- Project [BusinessDate#57031, stcd#57060, srcnm#57061, NetworkName#57034, NetworkURL#57035, custnum#57036, MarketCoverage#57037, RejectRows#57039, IssuerId#57059, IssuerId2#57058, issueridcomposite#57057, current\_date(Some(Asia/Kolkata)) AS sysdt#57056]

+- Project [BusinessDate#57031, stcd#57060, srcnm#57061, NetworkName#57034, NetworkURL#57035, custnum#57036, MarketCoverage#57037, RejectRows#57039, IssuerId#57059, IssuerId2#57058, issueridcomposite#57057]

+- Project [BusinessDate#57031, stcd#57060, srcnm#57061, NetworkName#57034, NetworkURL#57035, custnum#57036, MarketCoverage#57037, DentalOnlyPlan#57038, RejectRows#57039, IssuerId#57059, IssuerId2#57058, concat(IssuerId#57059, IssuerId2#57058) AS issueridcomposite#57057]

+- Project [BusinessDate#57031, stcd#57060, srcnm#57061, NetworkName#57034, NetworkURL#57035, custnum#57036, MarketCoverage#57037, DentalOnlyPlan#57038, RejectRows#57039, IssuerId#57059, IssuerId2#57058]

+- Project [IssuerIDDD#8273, BusinessDate#57031, stcd#57060, srcnm#57061, NetworkName#57034, NetworkURL#57035, custnum#57036, MarketCoverage#57037, DentalOnlyPlan#57038, RejectRows#57039, IssuerId#57059, IssuerId2#57058]

+- Project [IssuerIDD#8261, IssuerIDDD#8273, BusinessDate#57031, stcd#57060, srcnm#57061, NetworkName#57034, NetworkURL#57035, custnum#57036, MarketCoverage#57037, DentalOnlyPlan#57038, RejectRows#57039, IssuerId#57059, cast(IssuerIDDD#8273 as string) AS IssuerId2#57058]

+- Project [IssuerIDD#8261, IssuerIDDD#8273, BusinessDate#57031, stcd#57060, srcnm#57061, NetworkName#57034, NetworkURL#57035, custnum#57036, MarketCoverage#57037, DentalOnlyPlan#57038, RejectRows#57039, cast(IssuerIDD#8261 as string) AS IssuerId#57059]

+- Project [IssuerIDD#8261, IssuerId2#57030 AS IssuerIDDD#8273, BusinessDate#57031, stcd#57060, srcnm#57061, NetworkName#57034, NetworkURL#57035, custnum#57036, MarketCoverage#57037, DentalOnlyPlan#57038, RejectRows#57039]

+- Project [IssuerId#57029 AS IssuerIDD#8261, IssuerId2#57030, BusinessDate#57031, stcd#57060, srcnm#57061, NetworkName#57034, NetworkURL#57035, custnum#57036, MarketCoverage#57037, DentalOnlyPlan#57038, RejectRows#57039]

+- Project [IssuerId#57029, IssuerId2#57030, BusinessDate#57031, StateCode#57032 AS stcd#57060, srcnm#57061, NetworkName#57034, NetworkURL#57035, custnum#57036, MarketCoverage#57037, DentalOnlyPlan#57038, RejectRows#57039]

+- Project [IssuerId#57029, IssuerId2#57030, BusinessDate#57031, StateCode#57032, SourceName#57033 AS srcnm#57061, NetworkName#57034, NetworkURL#57035, custnum#57036, MarketCoverage#57037, DentalOnlyPlan#57038, RejectRows#57039]

+- Union false, false

:- Relation [IssuerId#57029,IssuerId2#57030,BusinessDate#57031,StateCode#57032,SourceName#57033,NetworkName#57034,NetworkURL#57035,custnum#57036,MarketCoverage#57037,DentalOnlyPlan#57038,RejectRows#57039] csv

+- Relation [IssuerId#57040,IssuerId2#57041,BusinessDate#57042,StateCode#57043,SourceName#57044,NetworkName#57045,NetworkURL#57046,custnum#57047,MarketCoverage#57048,DentalOnlyPlan#57049,RejectRows#57050] csv

>>> second\_join.describe

>>> second\_join.describe

<bound method DataFrame.describe of DataFrame[BusinessDate: date, stcd: string, srcnm: string, NetworkName: string, NetworkURL: string, custnum: string, MarketCoverage: string, RejectRows: string, IssuerId: string, IssuerId2: string, issueridcomposite: string, sysdt: date, systs: timestamp, cleannetworkname: string, curdt: date, curts: timestamp, cleannetworkname: string, \_c0: string, \_c1: string, \_c0: string, \_c1: string, \_c2: string, \_c3: int, \_c4: string, Protocol\_Used: string]>

>>> second\_join.show(1)

+------------+----+-----+-----------+--------------------+-------+--------------+----------+--------+---------+-----------------+----------+--------------------+----------------+----------+--------------------+----------------+---+-------+---+-----+-------+---+-----+-------------+

|BusinessDate|stcd|srcnm|NetworkName| NetworkURL|custnum|MarketCoverage|RejectRows|IssuerId|IssuerId2|issueridcomposite| sysdt| systs|cleannetworkname| curdt| curts|cleannetworkname|\_c0| \_c1|\_c0| \_c1| \_c2|\_c3| \_c4|Protocol\_Used|

+------------+----+-----+-----------+--------------------+-------+--------------+----------+--------+---------+-----------------+----------+--------------------+----------------+----------+--------------------+----------------+---+-------+---+-----+-------+---+-----+-------------+

| 2019-10-01| AZ| HIOS| La Paz|http://uamarketpl...| 21| null| null| 88925| 88925| 8892588925|2024-07-22|2024-07-22 16:59:...| La Paz|2024-07-22|2024-07-22 16:59:...| La Paz| AZ|Arizona|779|Jason|Bowling| 21|Coach| http secured|

+------------+----+-----+-----------+--------------------+-------+--------------+----------+--------+---------+-----------------+----------+--------------------+----------------+----------+--------------------+----------------+---+-------+---+-----+-------+---+-----+-------------+

only showing top 1 row

>>> from pyspark.sql import functions as F

>>> from pyspark.sql.window import Window

>>>

>>> # Step 1: Calculate average age and count

... grouped\_df = second\_join.groupBy('srcnm', 'Protocol\_Used', '\_c4') \

... .agg(

... F.avg('\_c3').alias('avg\_age'),

... F.count('\*').alias('profession\_count')

... )

>>> window\_spec = Window.partitionBy('Protocol\_Used', 'srcnm') \

... .orderBy(F.desc('profession\_count'))

>>> ranked\_df = grouped\_df.withColumn('seqno', F.row\_number().over(window\_spec))

>>> result\_df = ranked\_df.filter(ranked\_df.seqno == 2)

>>> result\_df.show(2)

+-----+-------------+-----+------------------+----------------+-----+

|srcnm|Protocol\_Used| \_c4| avg\_age|profession\_count|seqno|

+-----+-------------+-----+------------------+----------------+-----+

| HIOS| No Protocol|Nurse|22.076923076923077| 28158| 2|

|SERFF| No Protocol|Nurse| 21.0| 12996| 2|

+-----+-------------+-----+------------------+----------------+-----+

only showing top 2 rows

>>>

>>> mysql\_url = "jdbc://mysql://localhost/Hackathon"

>>> mysql\_table = "hackathon\_tbl"

>>> mysql\_user = "root"

>>> mysql\_password = "Root123$"

>>> result\_df.write .format("jdbc") .option("url", mysql\_url) .option("dbtable", mysql\_table) .option("user", mysql\_user) .option("password", mysql\_password) .option("driver", "com.mysql.cj.jdbc.Driver") .mode("overwrite") .save()

Traceback (most recent call last):

File "<stdin>", line 1, in <module>

File "/usr/local/spark/python/pyspark/sql/readwriter.py", line 738, in save

self.\_jwrite.save()

File "/usr/local/spark/python/lib/py4j-0.10.9.3-src.zip/py4j/java\_gateway.py", line 1322, in \_\_call\_\_

File "/usr/local/spark/python/pyspark/sql/utils.py", line 117, in deco

raise converted from None

pyspark.sql.utils.IllegalArgumentException: requirement failed: The driver could not open a JDBC connection. Check the URL: jdbc://mysql://localhost/Hackathon

>>> mysql\_url = "jdbc:mysql://localhost/Hackathon"

>>> mysql\_table = "hackathon\_tbl"

>>> mysql\_user = "root"

>>> mysql\_password = "Root123$"

>>> result\_df.write .format("jdbc") .option("url", mysql\_url) .option("dbtable", mysql\_table) .option("user", mysql\_user) .option("password", mysql\_password) .option("driver", "com.mysql.cj.jdbc.Driver") .mode("overwrite") .save()

>>> >>> rdd.take(4)

File "<stdin>", line 1

>>> rdd.take(4)

^

SyntaxError: invalid syntax

>>> ['1,Kristina,Chung,55,Pilot', '2,Paige,Chen,74,Teacher', 'NM,New Mexico', 'NY,New York']

['1,Kristina,Chung,55,Pilot', '2,Paige,Chen,74,Teacher', 'NM,New Mexico', 'NY,New York']

>>> >>> custfilter =rdd.map(lambda x: x.split(",") )

File "<stdin>", line 1

>>> custfilter =rdd.map(lambda x: x.split(",") )

^

SyntaxError: invalid syntax

>>> >>> print(custfilter.take(14))

File "<stdin>", line 1

>>> print(custfilter.take(14))

^

SyntaxError: invalid syntax

>>> [['1', 'Kristina', 'Chung', '55', 'Pilot'], ['2', 'Paige', 'Chen', '74', 'Teacher'], ['NM', 'New Mexico'], ['NY', 'New York'], ['NC', 'North Carolina'], ['ND', 'North Dakota'], ['OH', 'Ohio'], ['OK', 'Oklahoma'], ['OR', 'Oregon'], ['MD', 'Maryland'], ['MA', 'Massachusetts'], ['MI', 'Michigan'], ['MN', 'Minnesota'], ['MS', 'Mississippi']]

[['1', 'Kristina', 'Chung', '55', 'Pilot'], ['2', 'Paige', 'Chen', '74', 'Teacher'], ['NM', 'New Mexico'], ['NY', 'New York'], ['NC', 'North Carolina'], ['ND', 'North Dakota'], ['OH', 'Ohio'], ['OK', 'Oklahoma'], ['OR', 'Oregon'], ['MD', 'Maryland'], ['MA', 'Massachusetts'], ['MI', 'Michigan'], ['MN', 'Minnesota'], ['MS', 'Mississippi']]

>>> >>> splited\_rdd =rdd.map(lambda x: x.split(",") )

File "<stdin>", line 1

>>> splited\_rdd =rdd.map(lambda x: x.split(",") )

^

SyntaxError: invalid syntax

>>> >>> custfilter =splited\_rdd.filter(lambda x: len(x)>3 )

File "<stdin>", line 1

>>> custfilter =splited\_rdd.filter(lambda x: len(x)>3 )

^

SyntaxError: invalid syntax

>>> >>> print(custfilter.take(4))

File "<stdin>", line 1

>>> print(custfilter.take(4))

^

SyntaxError: invalid syntax

>>> [['1', 'Kristina', 'Chung', '55', 'Pilot'], ['2', 'Paige', 'Chen', '74', 'Teacher'], ['3', 'Sherri', 'Melton', '34', 'Firefighter'], ['4', 'Gretchen', 'Hill', '66', 'Computer hardware engineer']]

[['1', 'Kristina', 'Chung', '55', 'Pilot'], ['2', 'Paige', 'Chen', '74', 'Teacher'], ['3', 'Sherri', 'Melton', '34', 'Firefighter'], ['4', 'Gretchen', 'Hill', '66', 'Computer hardware engineer']]

>>> >>> print(custfilter.take(5))

File "<stdin>", line 1

>>> print(custfilter.take(5))

^

SyntaxError: invalid syntax

>>> [['1', 'Kristina', 'Chung', '55', 'Pilot'], ['2', 'Paige', 'Chen', '74', 'Teacher'], ['3', 'Sherri', 'Melton', '34', 'Firefighter'], ['4', 'Gretchen', 'Hill', '66', 'Computer hardware engineer'], ['5', 'Karen', 'Puckett', '74', 'Lawyer']]

[['1', 'Kristina', 'Chung', '55', 'Pilot'], ['2', 'Paige', 'Chen', '74', 'Teacher'], ['3', 'Sherri', 'Melton', '34', 'Firefighter'], ['4', 'Gretchen', 'Hill', '66', 'Computer hardware engineer'], ['5', 'Karen', 'Puckett', '74', 'Lawyer']]

>>> >>> statefilter =splited\_rdd.filter(lambda x: len(x)<3 )

File "<stdin>", line 1

>>> statefilter =splited\_rdd.filter(lambda x: len(x)<3 )

^

SyntaxError: invalid syntax

>>> >>> print(statefilter.take(4))

File "<stdin>", line 1

>>> print(statefilter.take(4))

^

SyntaxError: invalid syntax

>>> [['NM', 'New Mexico'], ['NY', 'New York'], ['NC', 'North Carolina'], ['ND', 'North Dakota']]

[['NM', 'New Mexico'], ['NY', 'New York'], ['NC', 'North Carolina'], ['ND', 'North Dakota']]

>>> >>>

File "<stdin>", line 1

>>>

^

SyntaxError: invalid syntax

>>> >>> complete\_df=spark.read.csv("/home/hduser/Desktop/Hackathon",sep=",",header=False,inferSchema=True)

File "<stdin>", line 1

>>> complete\_df=spark.read.csv("/home/hduser/Desktop/Hackathon",sep=",",header=False,inferSchema=True)

^

SyntaxError: invalid syntax

>>> Traceback (most recent call last):

File "<stdin>", line 1

Traceback (most recent call last):

^

SyntaxError: invalid syntax

>>> File "<stdin>", line 1, in <module>

File "<stdin>", line 1

File "<stdin>", line 1, in <module>

^

IndentationError: unexpected indent

>>> File "/usr/local/spark/python/pyspark/sql/readwriter.py", line 410, in csv

File "<stdin>", line 1

File "/usr/local/spark/python/pyspark/sql/readwriter.py", line 410, in csv

^

IndentationError: unexpected indent

>>> return self.\_df(self.\_jreader.csv(self.\_spark.\_sc.\_jvm.PythonUtils.toSeq(path)))

File "<stdin>", line 1

return self.\_df(self.\_jreader.csv(self.\_spark.\_sc.\_jvm.PythonUtils.toSeq(path)))

^

IndentationError: unexpected indent

>>> File "/usr/local/spark/python/lib/py4j-0.10.9.3-src.zip/py4j/java\_gateway.py", line 1322, in \_\_call\_\_

File "<stdin>", line 1

File "/usr/local/spark/python/lib/py4j-0.10.9.3-src.zip/py4j/java\_gateway.py", line 1322, in \_\_call\_\_

^

IndentationError: unexpected indent

>>>

>>> from pyspark.sql.types import \*

>>> schema = StructType([

... StructField("results", ArrayType(StructType([

... StructField("gender", StringType()),

... StructField("location",

... StructType([StructField("city", StringType()),StructField("state", StringType()),StructField("country", StringType()),StructField("postcode", StringType()),

... StructField("coordinates", StructType([StructField("latitude", StringType()),StructField("longitude", StringType())])),])),

... StructField("name", StructType([StructField("title", StringType()),StructField("first", StringType()),StructField("last", StringType())])),

... StructField("email", StringType()),

... StructField("login", StructType([StructField("uuid", StringType()),StructField("username", StringType())])),

... StructField("dob", StructType([StructField("date", StringType()),StructField("age", StringType())])),

... StructField("registered", StructType([StructField("date", StringType()),StructField("age", StringType())])),

...

... ])))])

>>>

>>> df\_new=spark.read.json("/home/hduser/Desktop/apidata.json",schema=schema)

Traceback (most recent call last):

File "<stdin>", line 1, in <module>

File "/usr/local/spark/python/pyspark/sql/readwriter.py", line 229, in json

return self.\_df(self.\_jreader.json(self.\_spark.\_sc.\_jvm.PythonUtils.toSeq(path)))

File "/usr/local/spark/python/lib/py4j-0.10.9.3-src.zip/py4j/java\_gateway.py", line 1322, in \_\_call\_\_

File "/usr/local/spark/python/pyspark/sql/utils.py", line 117, in deco

raise converted from None

pyspark.sql.utils.AnalysisException: Path does not exist: hdfs://localhost:54310/home/hduser/Desktop/apidata.json

>>> df\_new=spark.read.json("file:///home/hduser/Desktop/apidata.json",schema=schema)

Traceback (most recent call last):

File "<stdin>", line 1, in <module>

File "/usr/local/spark/python/pyspark/sql/readwriter.py", line 229, in json

return self.\_df(self.\_jreader.json(self.\_spark.\_sc.\_jvm.PythonUtils.toSeq(path)))

File "/usr/local/spark/python/lib/py4j-0.10.9.3-src.zip/py4j/java\_gateway.py", line 1322, in \_\_call\_\_

File "/usr/local/spark/python/pyspark/sql/utils.py", line 117, in deco

raise converted from None

pyspark.sql.utils.AnalysisException: Path does not exist: file:/home/hduser/Desktop/apidata.json

>>> df\_new=spark.read.json("file://home/hduser/Desktop/apidata.json",schema=schema)

24/07/22 17:39:26 WARN streaming.FileStreamSink: Assume no metadata directory. Error while looking for metadata directory in the path: file://home/hduser/Desktop/apidata.json.

java.lang.IllegalArgumentException: Wrong FS: file://home/hduser/Desktop/apidata.json, expected: file:///

at org.apache.hadoop.fs.FileSystem.checkPath(FileSystem.java:649)

at org.apache.hadoop.fs.RawLocalFileSystem.pathToFile(RawLocalFileSystem.java:82)

at org.apache.hadoop.fs.RawLocalFileSystem.deprecatedGetFileStatus(RawLocalFileSystem.java:606)

at org.apache.hadoop.fs.RawLocalFileSystem.getFileLinkStatusInternal(RawLocalFileSystem.java:824)

at org.apache.hadoop.fs.RawLocalFileSystem.getFileStatus(RawLocalFileSystem.java:601)

at org.apache.hadoop.fs.FilterFileSystem.getFileStatus(FilterFileSystem.java:428)

at org.apache.hadoop.fs.FileSystem.isDirectory(FileSystem.java:1439)

at org.apache.spark.sql.execution.streaming.FileStreamSink$.hasMetadata(FileStreamSink.scala:54)

at org.apache.spark.sql.execution.datasources.DataSource.resolveRelation(DataSource.scala:370)

at org.apache.spark.sql.DataFrameReader.loadV1Source(DataFrameReader.scala:274)

at org.apache.spark.sql.DataFrameReader.$anonfun$load$3(DataFrameReader.scala:245)

at scala.Option.getOrElse(Option.scala:189)

at org.apache.spark.sql.DataFrameReader.load(DataFrameReader.scala:245)

at org.apache.spark.sql.DataFrameReader.json(DataFrameReader.scala:405)

at sun.reflect.NativeMethodAccessorImpl.invoke0(Native Method)

at sun.reflect.NativeMethodAccessorImpl.invoke(NativeMethodAccessorImpl.java:62)

at sun.reflect.DelegatingMethodAccessorImpl.invoke(DelegatingMethodAccessorImpl.java:43)

at java.lang.reflect.Method.invoke(Method.java:498)

at py4j.reflection.MethodInvoker.invoke(MethodInvoker.java:244)

at py4j.reflection.ReflectionEngine.invoke(ReflectionEngine.java:357)

at py4j.Gateway.invoke(Gateway.java:282)

at py4j.commands.AbstractCommand.invokeMethod(AbstractCommand.java:132)

at py4j.commands.CallCommand.execute(CallCommand.java:79)

at py4j.ClientServerConnection.waitForCommands(ClientServerConnection.java:182)

at py4j.ClientServerConnection.run(ClientServerConnection.java:106)

at java.lang.Thread.run(Thread.java:750)

Traceback (most recent call last):

File "<stdin>", line 1, in <module>

File "/usr/local/spark/python/pyspark/sql/readwriter.py", line 229, in json

return self.\_df(self.\_jreader.json(self.\_spark.\_sc.\_jvm.PythonUtils.toSeq(path)))

File "/usr/local/spark/python/lib/py4j-0.10.9.3-src.zip/py4j/java\_gateway.py", line 1322, in \_\_call\_\_

File "/usr/local/spark/python/pyspark/sql/utils.py", line 117, in deco

raise converted from None

pyspark.sql.utils.IllegalArgumentException: Wrong FS: file://home/hduser/Desktop/apidata.json, expected: file:///

>>> df\_new=spark.read.json("file:///home/hduser/Desktop/apidata.json",schema=schema)

Traceback (most recent call last):

File "<stdin>", line 1, in <module>

File "/usr/local/spark/python/pyspark/sql/readwriter.py", line 229, in json

return self.\_df(self.\_jreader.json(self.\_spark.\_sc.\_jvm.PythonUtils.toSeq(path)))

File "/usr/local/spark/python/lib/py4j-0.10.9.3-src.zip/py4j/java\_gateway.py", line 1322, in \_\_call\_\_

File "/usr/local/spark/python/pyspark/sql/utils.py", line 117, in deco

raise converted from None

pyspark.sql.utils.AnalysisException: Path does not exist: file:/home/hduser/Desktop/apidata.json

>>> file:/home/hduser/Desktop/apidata.json

File "<stdin>", line 1

file:/home/hduser/Desktop/apidata.json

^

SyntaxError: invalid syntax

>>> df\_new=spark.read.json("/home/hduser/Desktop/Hackathon/apidata.json",schema=schema)

Traceback (most recent call last):

File "<stdin>", line 1, in <module>

File "/usr/local/spark/python/pyspark/sql/readwriter.py", line 229, in json

return self.\_df(self.\_jreader.json(self.\_spark.\_sc.\_jvm.PythonUtils.toSeq(path)))

File "/usr/local/spark/python/lib/py4j-0.10.9.3-src.zip/py4j/java\_gateway.py", line 1322, in \_\_call\_\_

File "/usr/local/spark/python/pyspark/sql/utils.py", line 117, in deco

raise converted from None

pyspark.sql.utils.AnalysisException: Path does not exist: hdfs://localhost:54310/home/hduser/Desktop/Hackathon/apidata.json

>>> df\_new=spark.read.json("file:///home/hduser/Desktop/Hackathon/apidata.json",schema=schema)

>>> df\_new.show(3)

+-------+

|results|

+-------+

| null|

| null|

| null|

+-------+

only showing top 3 rows

>>>

>>> from pyspark.sql.types import \*

>>> custom\_schema = StructType([

... StructField("results", ArrayType(StructType([

... StructField("gender", StringType()),

... StructField("location",

... StructType([StructField("city", StringType()),StructField("state", StringType()),StructField("country", StringType()),StructField("postcode", StringType()),

... StructField("coordinates", StructType([StructField("latitude", StringType()),StructField("longitude", StringType())])),])),

... StructField("name", StructType([StructField("title", StringType()),StructField("first", StringType()),StructField("last", StringType())])),

... StructField("email", StringType()),

... StructField("login", StructType([StructField("uuid", StringType()),StructField("username", StringType())])),

... StructField("dob", StructType([StructField("date", StringType()),StructField("age", StringType())])),

... StructField("registered", StructType([StructField("date", StringType()),StructField("age", StringType())])),

...

... ])))])

>>> df\_new=spark.read.json("file:///home/hduser/Desktop/Hackathon/apidata.json",schema=custom\_schema)

>>> df\_new.show(1)

+-------+

|results|

+-------+

| null|

+-------+

only showing top 1 row

>>>

>>> from pyspark.sql.types import \*

>>>

>>> schema = StructType([

... StructField("results", ArrayType(StructType([

... StructField("gender", StringType()),

... StructField("location", StructType([

... StructField("city", StringType()),

... StructField("state", StringType()),

... StructField("country", StringType()),

... StructField("postcode", StringType()),

... StructField("coordinates", StructType([

... StructField("latitude", StringType()),

... StructField("longitude", StringType())

... ]))

... ])),

... StructField("name", StructType([

... StructField("title", StringType()),

... StructField("first", StringType()),

... StructField("last", StringType())

... ])),

... StructField("email", StringType()),

... StructField("login", StructType([

... StructField("uuid", StringType()),

... StructField("username", StringType())

... ])),

... StructField("dob", StructType([

... StructField("date", StringType()),

... StructField("age", StringType())

... ])),

... StructField("registered", StructType([

... StructField("date", StringType()),

... StructField("age", StringType())

... ]))

... ]))

... ])

File "<stdin>", line 33

])

^

SyntaxError: invalid syntax

>>>

>>> from pyspark.sql.types import \*

>>>

>>> custom\_schema = StructType([

... StructField("results", ArrayType(

... StructType([

... StructField("gender", StringType()),

... StructField("location", StructType([

... StructField("city", StringType()),

... StructField("state", StringType()),

... StructField("country", StringType()),

... StructField("postcode", StringType()),

... StructField("coordinates", StructType([

... StructField("latitude", StringType()),

... StructField("longitude", StringType())

... ]))

... ])),

... StructField("name", StructType([

... StructField("title", StringType()),

... StructField("first", StringType()),

... StructField("last", StringType())

... ])),

... StructField("email", StringType()),

... StructField("login", StructType([

... StructField("uuid", StringType()),

... StructField("username", StringType())

... ])),

... StructField("dob", StructType([

... StructField("date", StringType()),

... StructField("age", StringType())

... ])),

... StructField("registered", StructType([

... StructField("date", StringType()),

... StructField("age", StringType())

... ]))

... ])

... ))

... ])

>>> df\_new=spark.read.json("file:///home/hduser/Desktop/Hackathon/apidata.json",schema=custom\_schema)

>>> df\_new.show(1)

+-------+

|results|

+-------+

| null|

+-------+

only showing top 1 row

>>> df\_new=spark.read.json("file:///home/hduser/Desktop/Hackathon/apidata.json")

>>> df\_new.show(1)

Traceback (most recent call last):

File "<stdin>", line 1, in <module>

File "/usr/local/spark/python/pyspark/sql/dataframe.py", line 494, in show

print(self.\_jdf.showString(n, 20, vertical))

File "/usr/local/spark/python/lib/py4j-0.10.9.3-src.zip/py4j/java\_gateway.py", line 1322, in \_\_call\_\_

File "/usr/local/spark/python/pyspark/sql/utils.py", line 117, in deco

raise converted from None

pyspark.sql.utils.AnalysisException:

Since Spark 2.3, the queries from raw JSON/CSV files are disallowed when the

referenced columns only include the internal corrupt record column

(named \_corrupt\_record by default). For example:

spark.read.schema(schema).csv(file).filter($"\_corrupt\_record".isNotNull).count()

and spark.read.schema(schema).csv(file).select("\_corrupt\_record").show().

Instead, you can cache or save the parsed results and then send the same query.

For example, val df = spark.read.schema(schema).csv(file).cache() and then

df.filter($"\_corrupt\_record".isNotNull).count().

>>> df\_new = spark.read.json("file:///home/hduser/Desktop/Hackathon/apidata.json", schema=custom\_schema)

>>> df\_new.show(1, truncate=False)

+-------+

|results|

+-------+

|null |

+-------+

only showing top 1 row

>>> df\_new.printSchema

<bound method DataFrame.printSchema of DataFrame[results: array<struct<gender:string,location:struct<city:string,state:string,country:string,postcode:string,coordinates:struct<latitude:string,longitude:string>>,name:struct<title:string,first:string,last:string>,email:string,login:struct<uuid:string,username:string>,dob:struct<date:string,age:string>,registered:struct<date:string,age:string>>>]>

>>>

>>> from pyspark.sql.functions import col, explode

>>>

>>> # Explode the 'results' array to flatten the DataFrame

... exploded\_df = jsondf.select(explode(col("results")).alias("exploded\_cols"))

Traceback (most recent call last):

File "<stdin>", line 2, in <module>

NameError: name 'jsondf' is not defined

>>>

>>> # Flatten the exploded DataFrame and extract the nested fields

... flattened\_df = exploded\_df \

... .withColumn('gender', col('exploded\_cols.gender')) \

... .withColumn('city', col('exploded\_cols.location.city')) \

... .withColumn('state', col('exploded\_cols.location.state')) \

... .withColumn('title', col('exploded\_cols.name.title')) \

... .withColumn('first', col('exploded\_cols.name.first')) \

... .withColumn('last', col('exploded\_cols.name.last')) \

... .withColumn('latitude', col('exploded\_cols.location.coordinates.latitude')) \

... .withColumn('longitude', col('exploded\_cols.location.coordinates.longitude')) \

... .withColumn('email', col('exploded\_cols.email')) \

... .withColumn('uuid', col('exploded\_cols.login.uuid')) \

... .withColumn('username', col('exploded\_cols.login.username')) \

... .withColumn('dob\_dt', col('exploded\_cols.dob.date')) \

... .withColumn('dob\_age', col('exploded\_cols.dob.age')) \

... .withColumn('login\_dt', col('exploded\_cols.registered.date')) \

... .withColumn('login\_age', col('exploded\_cols.registered.age')) \

... .drop('exploded\_cols')

Traceback (most recent call last):

File "<stdin>", line 2, in <module>

NameError: name 'exploded\_df' is not defined

>>>

>>> # Show the resulting flattened DataFrame

... flattened\_df.show(truncate=False)

Traceback (most recent call last):

File "<stdin>", line 2, in <module>

NameError: name 'flattened\_df' is not defined

>>>

>>> from pyspark.sql.functions import col, explode

>>>

>>> # Explode the 'results' array to flatten the DataFrame

... exploded\_df = df\_new.select(explode(col("results")).alias("exploded\_cols"))

>>>

>>> # Flatten the exploded DataFrame and extract the nested fields

... flattened\_df = exploded\_df \

... .withColumn('gender', col('exploded\_cols.gender')) \

... .withColumn('city', col('exploded\_cols.location.city')) \

... .withColumn('state', col('exploded\_cols.location.state')) \

... .withColumn('title', col('exploded\_cols.name.title')) \

... .withColumn('first', col('exploded\_cols.name.first')) \

... .withColumn('last', col('exploded\_cols.name.last')) \

... .withColumn('latitude', col('exploded\_cols.location.coordinates.latitude')) \

... .withColumn('longitude', col('exploded\_cols.location.coordinates.longitude')) \

... .withColumn('email', col('exploded\_cols.email')) \

... .withColumn('uuid', col('exploded\_cols.login.uuid')) \

... .withColumn('username', col('exploded\_cols.login.username')) \

... .withColumn('dob\_dt', col('exploded\_cols.dob.date')) \

... .withColumn('dob\_age', col('exploded\_cols.dob.age')) \

... .withColumn('login\_dt', col('exploded\_cols.registered.date')) \

... .withColumn('login\_age', col('exploded\_cols.registered.age')) \

... .drop('exploded\_cols')

>>>

>>> # Show the resulting flattened DataFrame

... flattened\_df.show(truncate=False)

+------+----+-----+-----+-----+----+--------+---------+-----+----+--------+------+-------+--------+---------+

|gender|city|state|title|first|last|latitude|longitude|email|uuid|username|dob\_dt|dob\_age|login\_dt|login\_age|

+------+----+-----+-----+-----+----+--------+---------+-----+----+--------+------+-------+--------+---------+

+------+----+-----+-----+-----+----+--------+---------+-----+----+--------+------+-------+--------+---------+

>>> gender\_count\_df = flattened\_df.groupBy('gender').agg(count('\*').alias('count'))

>>> gender\_count\_df.count()

0

>>> from pyspark.sql.functions import count

>>>

>>> # Count the number of occurrences for each gender

... gender\_count\_df = flattened\_df.groupBy('gender').agg(count('\*').alias('count'))

>>>

>>> # Show the result

... gender\_count\_df.show()

+------+-----+

|gender|count|

+------+-----+

+------+-----+

>>> flattened\_df.show(5) # Display the first 5 rows to check data

+------+----+-----+-----+-----+----+--------+---------+-----+----+--------+------+-------+--------+---------+

|gender|city|state|title|first|last|latitude|longitude|email|uuid|username|dob\_dt|dob\_age|login\_dt|login\_age|

+------+----+-----+-----+-----+----+--------+---------+-----+----+--------+------+-------+--------+---------+

+------+----+-----+-----+-----+----+--------+---------+-----+----+--------+------+-------+--------+---------+

>>>

>>> exploded\_df = df\_new.select(explode(col('results')).alias('exploded\_cols'))

>>> exploded\_df.show(truncate=False)

+-------------+

|exploded\_cols|

+-------------+

+-------------+

>>> df\_new.show(truncate=False)

+-------+

|results|

+-------+

|null |

|null |

|null |

|null |

|null |

|null |

|null |

|null |

|null |

|null |

|null |

|null |

|null |

|null |

|null |

|null |

|null |

|null |

|null |

|null |

+-------+

only showing top 20 rows

>>> c

>>> df\_raw = spark.read.json("file:///home/hduser/Desktop/Hackathon/apidata.json")

>>> df\_raw.show(truncate=False)

Traceback (most recent call last):

File "<stdin>", line 1, in <module>

File "/usr/local/spark/python/pyspark/sql/dataframe.py", line 502, in show

print(self.\_jdf.showString(n, int\_truncate, vertical))

File "/usr/local/spark/python/lib/py4j-0.10.9.3-src.zip/py4j/java\_gateway.py", line 1322, in \_\_call\_\_

File "/usr/local/spark/python/pyspark/sql/utils.py", line 117, in deco

raise converted from None

pyspark.sql.utils.AnalysisException:

Since Spark 2.3, the queries from raw JSON/CSV files are disallowed when the

referenced columns only include the internal corrupt record column

(named \_corrupt\_record by default). For example:

spark.read.schema(schema).csv(file).filter($"\_corrupt\_record".isNotNull).count()

and spark.read.schema(schema).csv(file).select("\_corrupt\_record").show().

Instead, you can cache or save the parsed results and then send the same query.

For example, val df = spark.read.schema(schema).csv(file).cache() and then

df.filter($"\_corrupt\_record".isNotNull).count().

>>>

>>> # Print schema of raw DataFrame

... df\_raw.printSchema()

root

|-- \_corrupt\_record: string (nullable = true)

>>> df\_raw = spark.read.json("file:///home/hduser/Desktop/Hackathon/apidata.json" , schema=custom\_schema)

>>> df\_r

df\_raw df\_renamed

>>> df\_raw.cache()

DataFrame[results: array<struct<gender:string,location:struct<city:string,state:string,country:string,postcode:string,coordinates:struct<latitude:string,longitude:string>>,name:struct<title:string,first:string,last:string>,email:string,login:struct<uuid:string,username:string>,dob:struct<date:string,age:string>,registered:struct<date:string,age:string>>>]

>>> df\_raw.show(truncate=False)

+-------+

|results|

+-------+

|null |

|null |

|null |

|null |

|null |

|null |

|null |

|null |

|null |

|null |

|null |

|null |

|null |

|null |

|null |

|null |

|null |

|null |

|null |

|null |

+-------+

only showing top 20 rows

>>>

>>> exploded\_df = df\_new.selectExpr('explode(results) as exploded\_cols') flattened\_df = exploded\_df \

File "<stdin>", line 1

exploded\_df = df\_new.selectExpr('explode(results) as exploded\_cols') flattened\_df = exploded\_df \

^

SyntaxError: invalid syntax

>>> .withColumn('gender', col('exploded\_cols.gender')) \

File "<stdin>", line 1

.withColumn('gender', col('exploded\_cols.gender')) \

^

IndentationError: unexpected indent

>>> .withColumn('city',col('exploded\_cols.location.city'))\

File "<stdin>", line 1

.withColumn('city',col('exploded\_cols.location.city'))\

^

IndentationError: unexpected indent

>>> .withColumn('state',col('exploded\_cols.location.state'))\

File "<stdin>", line 1

.withColumn('state',col('exploded\_cols.location.state'))\

^

IndentationError: unexpected indent

>>> .withColumn('title', col('exploded\_cols.name.title')) \

File "<stdin>", line 1

.withColumn('title', col('exploded\_cols.name.title')) \

^

IndentationError: unexpected indent

>>> .withColumn('first',col('exploded\_cols.name.first'))\

File "<stdin>", line 1

.withColumn('first',col('exploded\_cols.name.first'))\

^

IndentationError: unexpected indent

>>> .withColumn('last',col('exploded\_cols.name.last'))\

File "<stdin>", line 1

.withColumn('last',col('exploded\_cols.name.last'))\

^

IndentationError: unexpected indent

>>> .withColumn('latitude',col('exploded\_cols.location.coordinates.latitude'))\

File "<stdin>", line 1

.withColumn('latitude',col('exploded\_cols.location.coordinates.latitude'))\

^

IndentationError: unexpected indent

>>> .withColumn('longitude',col('exploded\_cols.location.coordinates.longitude'))\

File "<stdin>", line 1

.withColumn('longitude',col('exploded\_cols.location.coordinates.longitude'))\

^

IndentationError: unexpected indent

>>> .withColumn('email', col('exploded\_cols.email')) \

File "<stdin>", line 1

.withColumn('email', col('exploded\_cols.email')) \

^

IndentationError: unexpected indent

>>> .withColumn('uuid',col('exploded\_cols.login.uuid'))\

File "<stdin>", line 1

.withColumn('uuid',col('exploded\_cols.login.uuid'))\

^

IndentationError: unexpected indent

>>> .withColumn('username',col('exploded\_cols.login.username'))\

File "<stdin>", line 1

.withColumn('username',col('exploded\_cols.login.username'))\

^

IndentationError: unexpected indent

>>> .withColumn('dob\_dt',col('exploded\_cols.dob.date'))\

File "<stdin>", line 1

.withColumn('dob\_dt',col('exploded\_cols.dob.date'))\

^

IndentationError: unexpected indent

>>> .withColumn('dob\_age',col('exploded\_cols.dob.age'))\

File "<stdin>", line 1

.withColumn('dob\_age',col('exploded\_cols.dob.age'))\

^

IndentationError: unexpected indent

>>> .withColumn('login\_dt', col('exploded\_cols.registered.date')) \

File "<stdin>", line 1

.withColumn('login\_dt', col('exploded\_cols.registered.date')) \

^

IndentationError: unexpected indent

>>> .withColumn('login\_age',col('exploded\_cols.registered.age'))\

File "<stdin>", line 1

.withColumn('login\_age',col('exploded\_cols.registered.age'))\

^

IndentationError: unexpected indent

>>> .drop('exploded\_cols')

File "<stdin>", line 1

.drop('exploded\_cols')

^

IndentationError: unexpected indent

>>>

>>> exploded\_df = df\_new.selectExpr('explode(results) as exploded\_cols') flattened\_df = exploded\_df \

File "<stdin>", line 1

exploded\_df = df\_new.selectExpr('explode(results) as exploded\_cols') flattened\_df = exploded\_df \

^

SyntaxError: invalid syntax

>>> .withColumn('gender', col('exploded\_cols.gender')) \

File "<stdin>", line 1

.withColumn('gender', col('exploded\_cols.gender')) \

^

IndentationError: unexpected indent

>>> .withColumn('city',col('exploded\_cols.location.city'))\

File "<stdin>", line 1

.withColumn('city',col('exploded\_cols.location.city'))\

^

IndentationError: unexpected indent

>>> .withColumn('state',col('exploded\_cols.location.state'))\

File "<stdin>", line 1

.withColumn('state',col('exploded\_cols.location.state'))\

^

IndentationError: unexpected indent

>>> .withColumn('title', col('exploded\_cols.name.title')) \

File "<stdin>", line 1

.withColumn('title', col('exploded\_cols.name.title')) \

^

IndentationError: unexpected indent

>>> .withColumn('first',col('exploded\_cols.name.first'))\

File "<stdin>", line 1

.withColumn('first',col('exploded\_cols.name.first'))\

^

IndentationError: unexpected indent

>>> .withColumn('last',col('exploded\_cols.name.last'))\

File "<stdin>", line 1

.withColumn('last',col('exploded\_cols.name.last'))\

^

IndentationError: unexpected indent

>>> .withColumn('latitude',col('exploded\_cols.location.coordinates.latitude'))\ .withColumn('longitude',col('exploded\_cols.location.coordinates.longitude'))\

File "<stdin>", line 1

.withColumn('latitude',col('exploded\_cols.location.coordinates.latitude'))\ .withColumn('longitude',col('exploded\_cols.location.coordinates.longitude'))\

^

IndentationError: unexpected indent

>>> .withColumn('email', col('exploded\_cols.email')) \

File "<stdin>", line 1

.withColumn('email', col('exploded\_cols.email')) \

^

IndentationError: unexpected indent

>>> .withColumn('uuid',col('exploded\_cols.login.uuid'))\

File "<stdin>", line 1

.withColumn('uuid',col('exploded\_cols.login.uuid'))\

^

IndentationError: unexpected indent

>>> .withColumn('username',col('exploded\_cols.login.username'))\

File "<stdin>", line 1

.withColumn('username',col('exploded\_cols.login.username'))\

^

IndentationError: unexpected indent

>>> .withColumn('dob\_dt',col('exploded\_cols.dob.date'))\

File "<stdin>", line 1

.withColumn('dob\_dt',col('exploded\_cols.dob.date'))\

^

IndentationError: unexpected indent

>>> .withColumn('dob\_age',col('exploded\_cols.dob.age'))\

File "<stdin>", line 1

.withColumn('dob\_age',col('exploded\_cols.dob.age'))\

^

IndentationError: unexpected indent

>>> .withColumn('login\_dt', col('exploded\_cols.registered.date')) \

File "<stdin>", line 1

.withColumn('login\_dt', col('exploded\_cols.registered.date')) \

^

IndentationError: unexpected indent

>>> .withColumn('login\_age',col('exploded\_cols.registered.age'))\

File "<stdin>", line 1

.withColumn('login\_age',col('exploded\_cols.registered.age'))\

^

IndentationError: unexpected indent

>>> .drop('exploded\_cols')

File "<stdin>", line 1

.drop('exploded\_cols')

^

IndentationError: unexpected indent

>>>

>>> exploded\_df = df\_new.selectExpr('explode(results) as exploded\_cols')

>>>

>>> flattened\_df = exploded\_df \

... .withColumn('gender', col('exploded\_cols.gender')) \

... .withColumn('city', col('exploded\_cols.location.city')) \

... .withColumn('state', col('exploded\_cols.location.state')) \

... .withColumn('title', col('exploded\_cols.name.title')) \

... .withColumn('first', col('exploded\_cols.name.first')) \

... .withColumn('last', col('exploded\_cols.name.last')) \

... .withColumn('latitude', col('exploded\_cols.location.coordinates.latitude')) \

... .withColumn('longitude', col('exploded\_cols.location.coordinates.longitude')) \

... .withColumn('email', col('exploded\_cols.email')) \

... .withColumn('uuid', col('exploded\_cols.login.uuid')) \

... .withColumn('username', col('exploded\_cols.login.username')) \

... .withColumn('dob\_dt', col('exploded\_cols.dob.date')) \

... .withColumn('dob\_age', col('exploded\_cols.dob.age')) \

... .withColumn('login\_dt', col('exploded\_cols.registered.date')) \

... .withColumn('login\_age', col('exploded\_cols.registered.age')) \

... .drop('exploded\_cols')

>>> flattened\_df.show(3)

+------+----+-----+-----+-----+----+--------+---------+-----+----+--------+------+-------+--------+---------+

|gender|city|state|title|first|last|latitude|longitude|email|uuid|username|dob\_dt|dob\_age|login\_dt|login\_age|

+------+----+-----+-----+-----+----+--------+---------+-----+----+--------+------+-------+--------+---------+

+------+----+-----+-----+-----+----+--------+---------+-----+----+--------+------+-------+--------+---------+

>>> df\_raw = spark.read.json("file:///home/hduser/Desktop/Hackathon/apidata.json")

>>> df\_raw.show(truncate=False)

Traceback (most recent call last):

File "<stdin>", line 1, in <module>

File "/usr/local/spark/python/pyspark/sql/dataframe.py", line 502, in show

print(self.\_jdf.showString(n, int\_truncate, vertical))

File "/usr/local/spark/python/lib/py4j-0.10.9.3-src.zip/py4j/java\_gateway.py", line 1322, in \_\_call\_\_

File "/usr/local/spark/python/pyspark/sql/utils.py", line 117, in deco

raise converted from None

pyspark.sql.utils.AnalysisException:

Since Spark 2.3, the queries from raw JSON/CSV files are disallowed when the

referenced columns only include the internal corrupt record column

(named \_corrupt\_record by default). For example:

spark.read.schema(schema).csv(file).filter($"\_corrupt\_record".isNotNull).count()

and spark.read.schema(schema).csv(file).select("\_corrupt\_record").show().

Instead, you can cache or save the parsed results and then send the same query.

For example, val df = spark.read.schema(schema).csv(file).cache() and then

df.filter($"\_corrupt\_record".isNotNull).count().

>>> df\_raw.printSchema()

root

|-- \_corrupt\_record: string (nullable = true)

>>>

>>> # Read the JSON file

... df\_raw = spark.read.json("file:///home/hduser/Desktop/Hackathon/apidata.json")

>>>

>>> # Show the raw DataFrame

... df\_raw.show(truncate=False)

Traceback (most recent call last):

File "<stdin>", line 2, in <module>

File "/usr/local/spark/python/pyspark/sql/dataframe.py", line 502, in show

print(self.\_jdf.showString(n, int\_truncate, vertical))

File "/usr/local/spark/python/lib/py4j-0.10.9.3-src.zip/py4j/java\_gateway.py", line 1322, in \_\_call\_\_

File "/usr/local/spark/python/pyspark/sql/utils.py", line 117, in deco

raise converted from None

pyspark.sql.utils.AnalysisException:

Since Spark 2.3, the queries from raw JSON/CSV files are disallowed when the

referenced columns only include the internal corrupt record column

(named \_corrupt\_record by default). For example:

spark.read.schema(schema).csv(file).filter($"\_corrupt\_record".isNotNull).count()

and spark.read.schema(schema).csv(file).select("\_corrupt\_record").show().

Instead, you can cache or save the parsed results and then send the same query.

For example, val df = spark.read.schema(schema).csv(file).cache() and then

df.filter($"\_corrupt\_record".isNotNull).count().

>>> df\_raw = spark.read.json("file:///home/hduser/Desktop/Hackathon/apidata.json").cache()

>>> df\_raw.printSchema()

root

|-- \_corrupt\_record: string (nullable = true)

>>> df\_raw = spark.read.option("multiLine", "true").option("mode", "PERMISSIVE").json("file:///home/hduser/Desktop/Hackathon/apidata.json")

>>>

>>> df\_raw.printSchema()

root

|-- info: struct (nullable = true)

| |-- page: long (nullable = true)

| |-- results: long (nullable = true)

| |-- seed: string (nullable = true)

| |-- version: string (nullable = true)

|-- results: array (nullable = true)

| |-- element: struct (containsNull = true)

| | |-- cell: string (nullable = true)

| | |-- dob: struct (nullable = true)

| | | |-- age: long (nullable = true)

| | | |-- date: string (nullable = true)

| | |-- email: string (nullable = true)

| | |-- gender: string (nullable = true)

| | |-- id: struct (nullable = true)

| | | |-- name: string (nullable = true)

| | | |-- value: string (nullable = true)

| | |-- location: struct (nullable = true)

| | | |-- city: string (nullable = true)

| | | |-- coordinates: struct (nullable = true)

| | | | |-- latitude: string (nullable = true)

| | | | |-- longitude: string (nullable = true)

| | | |-- country: string (nullable = true)

| | | |-- postcode: string (nullable = true)

| | | |-- state: string (nullable = true)

| | | |-- street: struct (nullable = true)

| | | | |-- name: string (nullable = true)

| | | | |-- number: long (nullable = true)

| | | |-- timezone: struct (nullable = true)

| | | | |-- description: string (nullable = true)

| | | | |-- offset: string (nullable = true)

| | |-- login: struct (nullable = true)

| | | |-- md5: string (nullable = true)

| | | |-- password: string (nullable = true)

| | | |-- salt: string (nullable = true)

| | | |-- sha1: string (nullable = true)

| | | |-- sha256: string (nullable = true)

| | | |-- username: string (nullable = true)

| | | |-- uuid: string (nullable = true)

| | |-- name: struct (nullable = true)

| | | |-- first: string (nullable = true)

| | | |-- last: string (nullable = true)

| | | |-- title: string (nullable = true)

| | |-- nat: string (nullable = true)

| | |-- phone: string (nullable = true)

| | |-- picture: struct (nullable = true)

| | | |-- large: string (nullable = true)

| | | |-- medium: string (nullable = true)

| | | |-- thumbnail: string (nullable = true)

| | |-- registered: struct (nullable = true)

| | | |-- age: long (nullable = true)

| | | |-- date: string (nullable = true)

>>> exploded\_df = df\_raw.selectExpr('explode(results) as exploded\_cols')

>>>

>>> exploded\_df = df\_raw.selectExpr('explode(results) as exploded\_cols')

>>> flattened\_df = exploded\_df \

... .withColumn('gender', col('exploded\_cols.gender')) \

... .withColumn('city',col('exploded\_cols.location.city'))\

... .withColumn('state',col('exploded\_cols.location.state'))\

... .withColumn('title', col('exploded\_cols.name.title')) \

... .withColumn('first',col('exploded\_cols.name.first'))\

... .withColumn('last',col('exploded\_cols.name.last'))\

... .withColumn('latitude',col('exploded\_cols.location.coordinates.latitude'))\ .withColumn('longitude',col('exploded\_cols.location.coordinates.longitude'))\

File "<stdin>", line 8

.withColumn('latitude',col('exploded\_cols.location.coordinates.latitude'))\ .withColumn('longitude',col('exploded\_cols.location.coordinates.longitude'))\

^

SyntaxError: unexpected character after line continuation character

>>> .withColumn('email', col('exploded\_cols.email')) \

File "<stdin>", line 1

.withColumn('email', col('exploded\_cols.email')) \

^

IndentationError: unexpected indent

>>> .withColumn('uuid',col('exploded\_cols.login.uuid'))\

File "<stdin>", line 1

.withColumn('uuid',col('exploded\_cols.login.uuid'))\

^

IndentationError: unexpected indent

>>> .withColumn('username',col('exploded\_cols.login.username'))\

File "<stdin>", line 1

.withColumn('username',col('exploded\_cols.login.username'))\

^

IndentationError: unexpected indent

>>> .withColumn('dob\_dt',col('exploded\_cols.dob.date'))\

File "<stdin>", line 1

.withColumn('dob\_dt',col('exploded\_cols.dob.date'))\

^

IndentationError: unexpected indent

>>> .withColumn('dob\_age',col('exploded\_cols.dob.age'))\

File "<stdin>", line 1

.withColumn('dob\_age',col('exploded\_cols.dob.age'))\

^

IndentationError: unexpected indent

>>> .withColumn('login\_dt', col('exploded\_cols.registered.date')) \

File "<stdin>", line 1

.withColumn('login\_dt', col('exploded\_cols.registered.date')) \

^

IndentationError: unexpected indent

>>> .withColumn('login\_age',col('exploded\_cols.registered.age'))\

File "<stdin>", line 1

.withColumn('login\_age',col('exploded\_cols.registered.age'))\

^

IndentationError: unexpected indent

>>> .drop('exploded\_cols')

File "<stdin>", line 1

.drop('exploded\_cols')

^

IndentationError: unexpected indent

>>> exploded\_df = df\_raw.selectExpr('explode(results) as exploded\_cols')

>>> exploded\_df = df\_raw.selectExpr('explode(results) as exploded\_cols')

>>> flattened\_df = exploded\_df \

... .withColumn('gender', col('exploded\_cols.gender')) \

... .withColumn('city', col('exploded\_cols.location.city')) \

... .withColumn('state', col('exploded\_cols.location.state')) \

... .withColumn('title', col('exploded\_cols.name.title')) \

... .withColumn('first', col('exploded\_cols.name.first')) \

... .withColumn('last', col('exploded\_cols.name.last')) \

... .withColumn('latitude', col('exploded\_cols.location.coordinates.latitude')) \

... .withColumn('longitude', col('exploded\_cols.location.coordinates.longitude')) \

... .withColumn('email', col('exploded\_cols.email')) \

... .withColumn('uuid', col('exploded\_cols.login.uuid')) \

... .withColumn('username', col('exploded\_cols.login.username')) \

... .withColumn('dob\_dt', col('exploded\_cols.dob.date')) \

... .withColumn('dob\_age', col('exploded\_cols.dob.age')) \

... .withColumn('login\_dt', col('exploded\_cols.registered.date')) \

... .withColumn('login\_age', col('exploded\_cols.registered.age')) \

... .drop('exploded\_cols')

>>>

>>> fl

flatten( flattened\_df float( floor(

>>> flattened\_df.show(2)

+------+--------------+--------+-----+-----+---------+--------+---------+--------------------+--------------------+-------------+--------------------+-------+--------------------+---------+

|gender| city| state|title|first| last|latitude|longitude| email| uuid| username| dob\_dt|dob\_age| login\_dt|login\_age|

+------+--------------+--------+-----+-----+---------+--------+---------+--------------------+--------------------+-------------+--------------------+-------+--------------------+---------+

|female|Carrickmacross|Longford| Miss|Becky|Carpenter| 19.0157| 56.4959|becky.carpenter@e...|21312f38-cdf4-4fe...| bluefrog953|1961-12-27T07:26:...| 58|2003-02-07T11:28:...| 16|

|female| Bjerka|Rogaland| Miss|Freja| Hestetun| 49.8515| 166.0684|freja.hestetun@ex...|553a7a09-9b0b-41e...|browntiger686|1947-03-07T14:10:...| 76|2012-09-10T03:07:...| 7|

+------+--------------+--------+-----+-----+---------+--------+---------+--------------------+--------------------+-------------+--------------------+-------+--------------------+---------+

only showing top 2 rows

>>> flatten

flatten( flattened\_df

>>> flattened\_df.createOrReplaceTempView("view1")

>>> spark.sql("select gender,count(\*) from view1 group by gender ").show(1)

+------+--------+

|gender|count(1)|

+------+--------+

| male| 1|

+------+--------+

only showing top 1 row

>>> spark.sql("select gender,count(\*) from view1 group by gender ").show(2)

+------+--------+

|gender|count(1)|

+------+--------+

| male| 1|

|female| 3|

+------+--------+

>>> spark.sql("select gender,count(\*) from view1 group by gender ").show(5)

+------+--------+

|gender|count(1)|

+------+--------+

| male| 1|

|female| 3|

+------+--------+

>>> spark.sql("select gender,avg(dob\_age) from view1 group by gender ").show(5)

+------+-----------------+

|gender| avg(dob\_age)|

+------+-----------------+

| male| 47.0|

|female|66.33333333333333|

+------+-----------------+

>>> spark.sql("select date(dob\_dt) from view1 group by gender ").show(5)

Traceback (most recent call last):

File "<stdin>", line 1, in <module>

File "/usr/local/spark/python/pyspark/sql/session.py", line 723, in sql

return DataFrame(self.\_jsparkSession.sql(sqlQuery), self.\_wrapped)

File "/usr/local/spark/python/lib/py4j-0.10.9.3-src.zip/py4j/java\_gateway.py", line 1322, in \_\_call\_\_

File "/usr/local/spark/python/pyspark/sql/utils.py", line 117, in deco

raise converted from None

pyspark.sql.utils.AnalysisException: expression 'view1.dob\_dt' is neither present in the group by, nor is it an aggregate function. Add to group by or wrap in first() (or first\_value) if you don't care which value you get.;

Aggregate [gender#62602], [cast(dob\_dt#62701 as date) AS dob\_dt#62920]

+- SubqueryAlias view1

+- View (`view1`, [gender#62602,city#62606,state#62611,title#62617,first#62624,last#62632,latitude#62641,longitude#62651,email#62662,uuid#62674,username#62687,dob\_dt#62701,dob\_age#62716L,login\_dt#62732,login\_age#62749L])

+- Project [gender#62602, city#62606, state#62611, title#62617, first#62624, last#62632, latitude#62641, longitude#62651, email#62662, uuid#62674, username#62687, dob\_dt#62701, dob\_age#62716L, login\_dt#62732, login\_age#62749L]

+- Project [exploded\_cols#62600, gender#62602, city#62606, state#62611, title#62617, first#62624, last#62632, latitude#62641, longitude#62651, email#62662, uuid#62674, username#62687, dob\_dt#62701, dob\_age#62716L, login\_dt#62732, exploded\_cols#62600.registered.age AS login\_age#62749L]

+- Project [exploded\_cols#62600, gender#62602, city#62606, state#62611, title#62617, first#62624, last#62632, latitude#62641, longitude#62651, email#62662, uuid#62674, username#62687, dob\_dt#62701, dob\_age#62716L, exploded\_cols#62600.registered.date AS login\_dt#62732]

+- Project [exploded\_cols#62600, gender#62602, city#62606, state#62611, title#62617, first#62624, last#62632, latitude#62641, longitude#62651, email#62662, uuid#62674, username#62687, dob\_dt#62701, exploded\_cols#62600.dob.age AS dob\_age#62716L]

+- Project [exploded\_cols#62600, gender#62602, city#62606, state#62611, title#62617, first#62624, last#62632, latitude#62641, longitude#62651, email#62662, uuid#62674, username#62687, exploded\_cols#62600.dob.date AS dob\_dt#62701]

+- Project [exploded\_cols#62600, gender#62602, city#62606, state#62611, title#62617, first#62624, last#62632, latitude#62641, longitude#62651, email#62662, uuid#62674, exploded\_cols#62600.login.username AS username#62687]

+- Project [exploded\_cols#62600, gender#62602, city#62606, state#62611, title#62617, first#62624, last#62632, latitude#62641, longitude#62651, email#62662, exploded\_cols#62600.login.uuid AS uuid#62674]

+- Project [exploded\_cols#62600, gender#62602, city#62606, state#62611, title#62617, first#62624, last#62632, latitude#62641, longitude#62651, exploded\_cols#62600.email AS email#62662]

+- Project [exploded\_cols#62600, gender#62602, city#62606, state#62611, title#62617, first#62624, last#62632, latitude#62641, exploded\_cols#62600.location.coordinates.longitude AS longitude#62651]

+- Project [exploded\_cols#62600, gender#62602, city#62606, state#62611, title#62617, first#62624, last#62632, exploded\_cols#62600.location.coordinates.latitude AS latitude#62641]

+- Project [exploded\_cols#62600, gender#62602, city#62606, state#62611, title#62617, first#62624, exploded\_cols#62600.name.last AS last#62632]

+- Project [exploded\_cols#62600, gender#62602, city#62606, state#62611, title#62617, exploded\_cols#62600.name.first AS first#62624]

+- Project [exploded\_cols#62600, gender#62602, city#62606, state#62611, exploded\_cols#62600.name.title AS title#62617]

+- Project [exploded\_cols#62600, gender#62602, city#62606, exploded\_cols#62600.location.state AS state#62611]

+- Project [exploded\_cols#62600, gender#62602, exploded\_cols#62600.location.city AS city#62606]

+- Project [exploded\_cols#62600, exploded\_cols#62600.gender AS gender#62602]

+- Project [exploded\_cols#62600]

+- Generate explode(results#62590), false, [exploded\_cols#62600]

+- Relation [info#62589,results#62590] json

>>> flattened\_df = flattened\_df.withColumn('dob\_date\_only', date\_format('dob\_dt', 'yyyy-MM-dd'))

>>>

>>> flattened\_df = flattened\_df.withColumn('dob\_date\_only', date\_format('dob\_dt', 'yyyy-MM-dd'))

>>> spark.sql("select date(dob\_dt) from view1 group by gender ").show(5)

Traceback (most recent call last):

File "<stdin>", line 1, in <module>

File "/usr/local/spark/python/pyspark/sql/session.py", line 723, in sql

return DataFrame(self.\_jsparkSession.sql(sqlQuery), self.\_wrapped)

File "/usr/local/spark/python/lib/py4j-0.10.9.3-src.zip/py4j/java\_gateway.py", line 1322, in \_\_call\_\_

File "/usr/local/spark/python/pyspark/sql/utils.py", line 117, in deco

raise converted from None

pyspark.sql.utils.AnalysisException: expression 'view1.dob\_dt' is neither present in the group by, nor is it an aggregate function. Add to group by or wrap in first() (or first\_value) if you don't care which value you get.;

Aggregate [gender#62602], [cast(dob\_dt#62701 as date) AS dob\_dt#62955]

+- SubqueryAlias view1

+- View (`view1`, [gender#62602,city#62606,state#62611,title#62617,first#62624,last#62632,latitude#62641,longitude#62651,email#62662,uuid#62674,username#62687,dob\_dt#62701,dob\_age#62716L,login\_dt#62732,login\_age#62749L])

+- Project [gender#62602, city#62606, state#62611, title#62617, first#62624, last#62632, latitude#62641, longitude#62651, email#62662, uuid#62674, username#62687, dob\_dt#62701, dob\_age#62716L, login\_dt#62732, login\_age#62749L]

+- Project [exploded\_cols#62600, gender#62602, city#62606, state#62611, title#62617, first#62624, last#62632, latitude#62641, longitude#62651, email#62662, uuid#62674, username#62687, dob\_dt#62701, dob\_age#62716L, login\_dt#62732, exploded\_cols#62600.registered.age AS login\_age#62749L]

+- Project [exploded\_cols#62600, gender#62602, city#62606, state#62611, title#62617, first#62624, last#62632, latitude#62641, longitude#62651, email#62662, uuid#62674, username#62687, dob\_dt#62701, dob\_age#62716L, exploded\_cols#62600.registered.date AS login\_dt#62732]

+- Project [exploded\_cols#62600, gender#62602, city#62606, state#62611, title#62617, first#62624, last#62632, latitude#62641, longitude#62651, email#62662, uuid#62674, username#62687, dob\_dt#62701, exploded\_cols#62600.dob.age AS dob\_age#62716L]

+- Project [exploded\_cols#62600, gender#62602, city#62606, state#62611, title#62617, first#62624, last#62632, latitude#62641, longitude#62651, email#62662, uuid#62674, username#62687, exploded\_cols#62600.dob.date AS dob\_dt#62701]

+- Project [exploded\_cols#62600, gender#62602, city#62606, state#62611, title#62617, first#62624, last#62632, latitude#62641, longitude#62651, email#62662, uuid#62674, exploded\_cols#62600.login.username AS username#62687]

+- Project [exploded\_cols#62600, gender#62602, city#62606, state#62611, title#62617, first#62624, last#62632, latitude#62641, longitude#62651, email#62662, exploded\_cols#62600.login.uuid AS uuid#62674]

+- Project [exploded\_cols#62600, gender#62602, city#62606, state#62611, title#62617, first#62624, last#62632, latitude#62641, longitude#62651, exploded\_cols#62600.email AS email#62662]

+- Project [exploded\_cols#62600, gender#62602, city#62606, state#62611, title#62617, first#62624, last#62632, latitude#62641, exploded\_cols#62600.location.coordinates.longitude AS longitude#62651]

+- Project [exploded\_cols#62600, gender#62602, city#62606, state#62611, title#62617, first#62624, last#62632, exploded\_cols#62600.location.coordinates.latitude AS latitude#62641]

+- Project [exploded\_cols#62600, gender#62602, city#62606, state#62611, title#62617, first#62624, exploded\_cols#62600.name.last AS last#62632]

+- Project [exploded\_cols#62600, gender#62602, city#62606, state#62611, title#62617, exploded\_cols#62600.name.first AS first#62624]

+- Project [exploded\_cols#62600, gender#62602, city#62606, state#62611, exploded\_cols#62600.name.title AS title#62617]

+- Project [exploded\_cols#62600, gender#62602, city#62606, exploded\_cols#62600.location.state AS state#62611]

+- Project [exploded\_cols#62600, gender#62602, exploded\_cols#62600.location.city AS city#62606]

+- Project [exploded\_cols#62600, exploded\_cols#62600.gender AS gender#62602]

+- Project [exploded\_cols#62600]

+- Generate explode(results#62590), false, [exploded\_cols#62600]

+- Relation [info#62589,results#62590] json

>>> flattened\_df = flattened\_df.withColumn('dob\_date\_only', date\_format('dob\_dt', 'yyyy-MM-dd'))

>>>

>>> flattened\_df = flattened\_df.withColumn('dob\_date\_only', date\_format('dob\_dt', 'yyyy-MM-dd'))

>>> spark.catalog.dropTempView("view1")

>>> flattened\_df = df\_raw.withColumn('dob\_date\_only', date\_format('dob\_dt', 'yyyy-MM-dd'))

Traceback (most recent call last):

File "<stdin>", line 1, in <module>

File "/usr/local/spark/python/pyspark/sql/dataframe.py", line 2478, in withColumn

return DataFrame(self.\_jdf.withColumn(colName, col.\_jc), self.sql\_ctx)

File "/usr/local/spark/python/lib/py4j-0.10.9.3-src.zip/py4j/java\_gateway.py", line 1322, in \_\_call\_\_

File "/usr/local/spark/python/pyspark/sql/utils.py", line 117, in deco

raise converted from None

pyspark.sql.utils.AnalysisException: cannot resolve 'dob\_dt' given input columns: [info, results];

'Project [info#62589, results#62590, date\_format('dob\_dt, yyyy-MM-dd, Some(Asia/Kolkata)) AS dob\_date\_only#62990]

+- Relation [info#62589,results#62590] json

>>> flattened\_df = flattened\_df.withColumn('dob\_date\_only', date\_format('dob\_dt', 'yyyy-MM-dd'))

>>> flattened\_df = flattened\_df.withColumn('dob\_date\_only', date\_format('dob\_dt', 'yyyy-MM-dd'))

>>> flattened\_df.createOrReplaceTempView("view1")

>>> spark.sql("select date(dob\_dt) from view1 group by gender ").show(5)

Traceback (most recent call last):

File "<stdin>", line 1, in <module>

File "/usr/local/spark/python/pyspark/sql/session.py", line 723, in sql

return DataFrame(self.\_jsparkSession.sql(sqlQuery), self.\_wrapped)

File "/usr/local/spark/python/lib/py4j-0.10.9.3-src.zip/py4j/java\_gateway.py", line 1322, in \_\_call\_\_

File "/usr/local/spark/python/pyspark/sql/utils.py", line 117, in deco

raise converted from None

pyspark.sql.utils.AnalysisException: expression 'view1.dob\_dt' is neither present in the group by, nor is it an aggregate function. Add to group by or wrap in first() (or first\_value) if you don't care which value you get.;

Aggregate [gender#62602], [cast(dob\_dt#62701 as date) AS dob\_dt#63025]

+- SubqueryAlias view1

+- View (`view1`, [gender#62602,city#62606,state#62611,title#62617,first#62624,last#62632,latitude#62641,longitude#62651,email#62662,uuid#62674,username#62687,dob\_dt#62701,dob\_age#62716L,login\_dt#62732,login\_age#62749L,dob\_date\_only#63008])

+- Project [gender#62602, city#62606, state#62611, title#62617, first#62624, last#62632, latitude#62641, longitude#62651, email#62662, uuid#62674, username#62687, dob\_dt#62701, dob\_age#62716L, login\_dt#62732, login\_age#62749L, date\_format(cast(dob\_dt#62701 as timestamp), yyyy-MM-dd, Some(Asia/Kolkata)) AS dob\_date\_only#63008]

+- Project [gender#62602, city#62606, state#62611, title#62617, first#62624, last#62632, latitude#62641, longitude#62651, email#62662, uuid#62674, username#62687, dob\_dt#62701, dob\_age#62716L, login\_dt#62732, login\_age#62749L, date\_format(cast(dob\_dt#62701 as timestamp), yyyy-MM-dd, Some(Asia/Kolkata)) AS dob\_date\_only#62991]

+- Project [gender#62602, city#62606, state#62611, title#62617, first#62624, last#62632, latitude#62641, longitude#62651, email#62662, uuid#62674, username#62687, dob\_dt#62701, dob\_age#62716L, login\_dt#62732, login\_age#62749L, date\_format(cast(dob\_dt#62701 as timestamp), yyyy-MM-dd, Some(Asia/Kolkata)) AS dob\_date\_only#62973]

+- Project [gender#62602, city#62606, state#62611, title#62617, first#62624, last#62632, latitude#62641, longitude#62651, email#62662, uuid#62674, username#62687, dob\_dt#62701, dob\_age#62716L, login\_dt#62732, login\_age#62749L, date\_format(cast(dob\_dt#62701 as timestamp), yyyy-MM-dd, Some(Asia/Kolkata)) AS dob\_date\_only#62956]

+- Project [gender#62602, city#62606, state#62611, title#62617, first#62624, last#62632, latitude#62641, longitude#62651, email#62662, uuid#62674, username#62687, dob\_dt#62701, dob\_age#62716L, login\_dt#62732, login\_age#62749L, date\_format(cast(dob\_dt#62701 as timestamp), yyyy-MM-dd, Some(Asia/Kolkata)) AS dob\_date\_only#62938]

+- Project [gender#62602, city#62606, state#62611, title#62617, first#62624, last#62632, latitude#62641, longitude#62651, email#62662, uuid#62674, username#62687, dob\_dt#62701, dob\_age#62716L, login\_dt#62732, login\_age#62749L, date\_format(cast(dob\_dt#62701 as timestamp), yyyy-MM-dd, Some(Asia/Kolkata)) AS dob\_date\_only#62921]

+- Project [gender#62602, city#62606, state#62611, title#62617, first#62624, last#62632, latitude#62641, longitude#62651, email#62662, uuid#62674, username#62687, dob\_dt#62701, dob\_age#62716L, login\_dt#62732, login\_age#62749L]

+- Project [exploded\_cols#62600, gender#62602, city#62606, state#62611, title#62617, first#62624, last#62632, latitude#62641, longitude#62651, email#62662, uuid#62674, username#62687, dob\_dt#62701, dob\_age#62716L, login\_dt#62732, exploded\_cols#62600.registered.age AS login\_age#62749L]

+- Project [exploded\_cols#62600, gender#62602, city#62606, state#62611, title#62617, first#62624, last#62632, latitude#62641, longitude#62651, email#62662, uuid#62674, username#62687, dob\_dt#62701, dob\_age#62716L, exploded\_cols#62600.registered.date AS login\_dt#62732]

+- Project [exploded\_cols#62600, gender#62602, city#62606, state#62611, title#62617, first#62624, last#62632, latitude#62641, longitude#62651, email#62662, uuid#62674, username#62687, dob\_dt#62701, exploded\_cols#62600.dob.age AS dob\_age#62716L]

+- Project [exploded\_cols#62600, gender#62602, city#62606, state#62611, title#62617, first#62624, last#62632, latitude#62641, longitude#62651, email#62662, uuid#62674, username#62687, exploded\_cols#62600.dob.date AS dob\_dt#62701]

+- Project [exploded\_cols#62600, gender#62602, city#62606, state#62611, title#62617, first#62624, last#62632, latitude#62641, longitude#62651, email#62662, uuid#62674, exploded\_cols#62600.login.username AS username#62687]

+- Project [exploded\_cols#62600, gender#62602, city#62606, state#62611, title#62617, first#62624, last#62632, latitude#62641, longitude#62651, email#62662, exploded\_cols#62600.login.uuid AS uuid#62674]

+- Project [exploded\_cols#62600, gender#62602, city#62606, state#62611, title#62617, first#62624, last#62632, latitude#62641, longitude#62651, exploded\_cols#62600.email AS email#62662]

+- Project [exploded\_cols#62600, gender#62602, city#62606, state#62611, title#62617, first#62624, last#62632, latitude#62641, exploded\_cols#62600.location.coordinates.longitude AS longitude#62651]

+- Project [exploded\_cols#62600, gender#62602, city#62606, state#62611, title#62617, first#62624, last#62632, exploded\_cols#62600.location.coordinates.latitude AS latitude#62641]

+- Project [exploded\_cols#62600, gender#62602, city#62606, state#62611, title#62617, first#62624, exploded\_cols#62600.name.last AS last#62632]

+- Project [exploded\_cols#62600, gender#62602, city#62606, state#62611, title#62617, exploded\_cols#62600.name.first AS first#62624]

+- Project [exploded\_cols#62600, gender#62602, city#62606, state#62611, exploded\_cols#62600.name.title AS title#62617]

+- Project [exploded\_cols#62600, gender#62602, city#62606, exploded\_cols#62600.location.state AS state#62611]

+- Project [exploded\_cols#62600, gender#62602, exploded\_cols#62600.location.city AS city#62606]

+- Project [exploded\_cols#62600, exploded\_cols#62600.gender AS gender#62602]

+- Project [exploded\_cols#62600]

+- Generate explode(results#62590), false, [exploded\_cols#62600]

+- Relation [info#62589,results#62590] json

>>> flattened\_df.show(5)

+------+--------------+-----------+-----+------+---------+--------+---------+--------------------+--------------------+--------------+--------------------+-------+--------------------+---------+-------------+

|gender| city| state|title| first| last|latitude|longitude| email| uuid| username| dob\_dt|dob\_age| login\_dt|login\_age|dob\_date\_only|

+------+--------------+-----------+-----+------+---------+--------+---------+--------------------+--------------------+--------------+--------------------+-------+--------------------+---------+-------------+

|female|Carrickmacross| Longford| Miss| Becky|Carpenter| 19.0157| 56.4959|becky.carpenter@e...|21312f38-cdf4-4fe...| bluefrog953|1961-12-27T07:26:...| 58|2003-02-07T11:28:...| 16| 1961-12-27|

|female| Bjerka| Rogaland| Miss| Freja| Hestetun| 49.8515| 166.0684|freja.hestetun@ex...|553a7a09-9b0b-41e...| browntiger686|1947-03-07T14:10:...| 76|2012-09-10T03:07:...| 7| 1947-03-07|

| male| Roslev|Hovedstaden| Mr|Alfred| Møller|-36.9245| -0.9917|alfred.moller@exa...|18bb2a51-14d3-47d...|orangegoose716|1975-08-23T13:17:...| 47|2014-04-06T14:09:...| 5| 1975-08-23|

|female| Trabzon| Elazığ| Miss| Oya| Ilıcalı| 72.4704| 66.0216|oya.ilicali@examp...|c74c81c4-8100-487...| redwolf988|1957-11-16T21:46:...| 65|2012-05-11T14:17:...| 7| 1957-11-17|

+------+--------------+-----------+-----+------+---------+--------+---------+--------------------+--------------------+--------------+--------------------+-------+--------------------+---------+-------------+

>>> spark.sql("select date(dob\_date\_only) from view1 group by gender ").show(5)

Traceback (most recent call last):

File "<stdin>", line 1, in <module>

File "/usr/local/spark/python/pyspark/sql/session.py", line 723, in sql

return DataFrame(self.\_jsparkSession.sql(sqlQuery), self.\_wrapped)

File "/usr/local/spark/python/lib/py4j-0.10.9.3-src.zip/py4j/java\_gateway.py", line 1322, in \_\_call\_\_

File "/usr/local/spark/python/pyspark/sql/utils.py", line 117, in deco

raise converted from None

pyspark.sql.utils.AnalysisException: expression 'view1.dob\_date\_only' is neither present in the group by, nor is it an aggregate function. Add to group by or wrap in first() (or first\_value) if you don't care which value you get.;

Aggregate [gender#62602], [cast(dob\_date\_only#63008 as date) AS dob\_date\_only#63092]

+- SubqueryAlias view1

+- View (`view1`, [gender#62602,city#62606,state#62611,title#62617,first#62624,last#62632,latitude#62641,longitude#62651,email#62662,uuid#62674,username#62687,dob\_dt#62701,dob\_age#62716L,login\_dt#62732,login\_age#62749L,dob\_date\_only#63008])

+- Project [gender#62602, city#62606, state#62611, title#62617, first#62624, last#62632, latitude#62641, longitude#62651, email#62662, uuid#62674, username#62687, dob\_dt#62701, dob\_age#62716L, login\_dt#62732, login\_age#62749L, date\_format(cast(dob\_dt#62701 as timestamp), yyyy-MM-dd, Some(Asia/Kolkata)) AS dob\_date\_only#63008]

+- Project [gender#62602, city#62606, state#62611, title#62617, first#62624, last#62632, latitude#62641, longitude#62651, email#62662, uuid#62674, username#62687, dob\_dt#62701, dob\_age#62716L, login\_dt#62732, login\_age#62749L, date\_format(cast(dob\_dt#62701 as timestamp), yyyy-MM-dd, Some(Asia/Kolkata)) AS dob\_date\_only#62991]

+- Project [gender#62602, city#62606, state#62611, title#62617, first#62624, last#62632, latitude#62641, longitude#62651, email#62662, uuid#62674, username#62687, dob\_dt#62701, dob\_age#62716L, login\_dt#62732, login\_age#62749L, date\_format(cast(dob\_dt#62701 as timestamp), yyyy-MM-dd, Some(Asia/Kolkata)) AS dob\_date\_only#62973]

+- Project [gender#62602, city#62606, state#62611, title#62617, first#62624, last#62632, latitude#62641, longitude#62651, email#62662, uuid#62674, username#62687, dob\_dt#62701, dob\_age#62716L, login\_dt#62732, login\_age#62749L, date\_format(cast(dob\_dt#62701 as timestamp), yyyy-MM-dd, Some(Asia/Kolkata)) AS dob\_date\_only#62956]

+- Project [gender#62602, city#62606, state#62611, title#62617, first#62624, last#62632, latitude#62641, longitude#62651, email#62662, uuid#62674, username#62687, dob\_dt#62701, dob\_age#62716L, login\_dt#62732, login\_age#62749L, date\_format(cast(dob\_dt#62701 as timestamp), yyyy-MM-dd, Some(Asia/Kolkata)) AS dob\_date\_only#62938]

+- Project [gender#62602, city#62606, state#62611, title#62617, first#62624, last#62632, latitude#62641, longitude#62651, email#62662, uuid#62674, username#62687, dob\_dt#62701, dob\_age#62716L, login\_dt#62732, login\_age#62749L, date\_format(cast(dob\_dt#62701 as timestamp), yyyy-MM-dd, Some(Asia/Kolkata)) AS dob\_date\_only#62921]

+- Project [gender#62602, city#62606, state#62611, title#62617, first#62624, last#62632, latitude#62641, longitude#62651, email#62662, uuid#62674, username#62687, dob\_dt#62701, dob\_age#62716L, login\_dt#62732, login\_age#62749L]

+- Project [exploded\_cols#62600, gender#62602, city#62606, state#62611, title#62617, first#62624, last#62632, latitude#62641, longitude#62651, email#62662, uuid#62674, username#62687, dob\_dt#62701, dob\_age#62716L, login\_dt#62732, exploded\_cols#62600.registered.age AS login\_age#62749L]

+- Project [exploded\_cols#62600, gender#62602, city#62606, state#62611, title#62617, first#62624, last#62632, latitude#62641, longitude#62651, email#62662, uuid#62674, username#62687, dob\_dt#62701, dob\_age#62716L, exploded\_cols#62600.registered.date AS login\_dt#62732]

+- Project [exploded\_cols#62600, gender#62602, city#62606, state#62611, title#62617, first#62624, last#62632, latitude#62641, longitude#62651, email#62662, uuid#62674, username#62687, dob\_dt#62701, exploded\_cols#62600.dob.age AS dob\_age#62716L]

+- Project [exploded\_cols#62600, gender#62602, city#62606, state#62611, title#62617, first#62624, last#62632, latitude#62641, longitude#62651, email#62662, uuid#62674, username#62687, exploded\_cols#62600.dob.date AS dob\_dt#62701]

+- Project [exploded\_cols#62600, gender#62602, city#62606, state#62611, title#62617, first#62624, last#62632, latitude#62641, longitude#62651, email#62662, uuid#62674, exploded\_cols#62600.login.username AS username#62687]

+- Project [exploded\_cols#62600, gender#62602, city#62606, state#62611, title#62617, first#62624, last#62632, latitude#62641, longitude#62651, email#62662, exploded\_cols#62600.login.uuid AS uuid#62674]

+- Project [exploded\_cols#62600, gender#62602, city#62606, state#62611, title#62617, first#62624, last#62632, latitude#62641, longitude#62651, exploded\_cols#62600.email AS email#62662]

+- Project [exploded\_cols#62600, gender#62602, city#62606, state#62611, title#62617, first#62624, last#62632, latitude#62641, exploded\_cols#62600.location.coordinates.longitude AS longitude#62651]

+- Project [exploded\_cols#62600, gender#62602, city#62606, state#62611, title#62617, first#62624, last#62632, exploded\_cols#62600.location.coordinates.latitude AS latitude#62641]

+- Project [exploded\_cols#62600, gender#62602, city#62606, state#62611, title#62617, first#62624, exploded\_cols#62600.name.last AS last#62632]

+- Project [exploded\_cols#62600, gender#62602, city#62606, state#62611, title#62617, exploded\_cols#62600.name.first AS first#62624]

+- Project [exploded\_cols#62600, gender#62602, city#62606, state#62611, exploded\_cols#62600.name.title AS title#62617]

+- Project [exploded\_cols#62600, gender#62602, city#62606, exploded\_cols#62600.location.state AS state#62611]

+- Project [exploded\_cols#62600, gender#62602, exploded\_cols#62600.location.city AS city#62606]

+- Project [exploded\_cols#62600, exploded\_cols#62600.gender AS gender#62602]

+- Project [exploded\_cols#62600]

+- Generate explode(results#62590), false, [exploded\_cols#62600]

+- Relation [info#62589,results#62590] json

>>> flattened\_df.show(5)

>>> flattened\_df.show(5)

+------+--------------+-----------+-----+------+---------+--------+---------+--------------------+--------------------+--------------+--------------------+-------+--------------------+---------+-------------+

|gender| city| state|title| first| last|latitude|longitude| email| uuid| username| dob\_dt|dob\_age| login\_dt|login\_age|dob\_date\_only|

+------+--------------+-----------+-----+------+---------+--------+---------+--------------------+--------------------+--------------+--------------------+-------+--------------------+---------+-------------+

|female|Carrickmacross| Longford| Miss| Becky|Carpenter| 19.0157| 56.4959|becky.carpenter@e...|21312f38-cdf4-4fe...| bluefrog953|1961-12-27T07:26:...| 58|2003-02-07T11:28:...| 16| 1961-12-27|

|female| Bjerka| Rogaland| Miss| Freja| Hestetun| 49.8515| 166.0684|freja.hestetun@ex...|553a7a09-9b0b-41e...| browntiger686|1947-03-07T14:10:...| 76|2012-09-10T03:07:...| 7| 1947-03-07|

| male| Roslev|Hovedstaden| Mr|Alfred| Møller|-36.9245| -0.9917|alfred.moller@exa...|18bb2a51-14d3-47d...|orangegoose716|1975-08-23T13:17:...| 47|2014-04-06T14:09:...| 5| 1975-08-23|

|female| Trabzon| Elazığ| Miss| Oya| Ilıcalı| 72.4704| 66.0216|oya.ilicali@examp...|c74c81c4-8100-487...| redwolf988|1957-11-16T21:46:...| 65|2012-05-11T14:17:...| 7| 1957-11-17|

+------+--------------+-----------+-----+------+---------+--------+---------+--------------------+--------------------+--------------+--------------------+-------+--------------------+---------+-------------+

>>> spark.sql("select \* from view1").show(5)

+------+--------------+-----------+-----+------+---------+--------+---------+--------------------+--------------------+--------------+--------------------+-------+--------------------+---------+-------------+

|gender| city| state|title| first| last|latitude|longitude| email| uuid| username| dob\_dt|dob\_age| login\_dt|login\_age|dob\_date\_only|

+------+--------------+-----------+-----+------+---------+--------+---------+--------------------+--------------------+--------------+--------------------+-------+--------------------+---------+-------------+

|female|Carrickmacross| Longford| Miss| Becky|Carpenter| 19.0157| 56.4959|becky.carpenter@e...|21312f38-cdf4-4fe...| bluefrog953|1961-12-27T07:26:...| 58|2003-02-07T11:28:...| 16| 1961-12-27|

|female| Bjerka| Rogaland| Miss| Freja| Hestetun| 49.8515| 166.0684|freja.hestetun@ex...|553a7a09-9b0b-41e...| browntiger686|1947-03-07T14:10:...| 76|2012-09-10T03:07:...| 7| 1947-03-07|

| male| Roslev|Hovedstaden| Mr|Alfred| Møller|-36.9245| -0.9917|alfred.moller@exa...|18bb2a51-14d3-47d...|orangegoose716|1975-08-23T13:17:...| 47|2014-04-06T14:09:...| 5| 1975-08-23|

|female| Trabzon| Elazığ| Miss| Oya| Ilıcalı| 72.4704| 66.0216|oya.ilicali@examp...|c74c81c4-8100-487...| redwolf988|1957-11-16T21:46:...| 65|2012-05-11T14:17:...| 7| 1957-11-17|

+------+--------------+-----------+-----+------+---------+--------+---------+--------------------+--------------------+--------------+--------------------+-------+--------------------+---------+-------------+

>>> spark.sql("""

... SELECT

... gender,

... DATE(dob\_dt) as dob\_date\_only

... FROM view1

... GROUP BY gender, dob\_date\_only

... """).show(5, truncate=False)

Traceback (most recent call last):

File "<stdin>", line 7, in <module>

File "/usr/local/spark/python/pyspark/sql/session.py", line 723, in sql

return DataFrame(self.\_jsparkSession.sql(sqlQuery), self.\_wrapped)

File "/usr/local/spark/python/lib/py4j-0.10.9.3-src.zip/py4j/java\_gateway.py", line 1322, in \_\_call\_\_

File "/usr/local/spark/python/pyspark/sql/utils.py", line 117, in deco

raise converted from None

pyspark.sql.utils.AnalysisException: expression 'view1.dob\_dt' is neither present in the group by, nor is it an aggregate function. Add to group by or wrap in first() (or first\_value) if you don't care which value you get.;

Aggregate [gender#62602, dob\_date\_only#63008], [gender#62602, cast(dob\_dt#62701 as date) AS dob\_date\_only#63241]

+- SubqueryAlias view1

+- View (`view1`, [gender#62602,city#62606,state#62611,title#62617,first#62624,last#62632,latitude#62641,longitude#62651,email#62662,uuid#62674,username#62687,dob\_dt#62701,dob\_age#62716L,login\_dt#62732,login\_age#62749L,dob\_date\_only#63008])

+- Project [gender#62602, city#62606, state#62611, title#62617, first#62624, last#62632, latitude#62641, longitude#62651, email#62662, uuid#62674, username#62687, dob\_dt#62701, dob\_age#62716L, login\_dt#62732, login\_age#62749L, date\_format(cast(dob\_dt#62701 as timestamp), yyyy-MM-dd, Some(Asia/Kolkata)) AS dob\_date\_only#63008]

+- Project [gender#62602, city#62606, state#62611, title#62617, first#62624, last#62632, latitude#62641, longitude#62651, email#62662, uuid#62674, username#62687, dob\_dt#62701, dob\_age#62716L, login\_dt#62732, login\_age#62749L, date\_format(cast(dob\_dt#62701 as timestamp), yyyy-MM-dd, Some(Asia/Kolkata)) AS dob\_date\_only#62991]

+- Project [gender#62602, city#62606, state#62611, title#62617, first#62624, last#62632, latitude#62641, longitude#62651, email#62662, uuid#62674, username#62687, dob\_dt#62701, dob\_age#62716L, login\_dt#62732, login\_age#62749L, date\_format(cast(dob\_dt#62701 as timestamp), yyyy-MM-dd, Some(Asia/Kolkata)) AS dob\_date\_only#62973]

+- Project [gender#62602, city#62606, state#62611, title#62617, first#62624, last#62632, latitude#62641, longitude#62651, email#62662, uuid#62674, username#62687, dob\_dt#62701, dob\_age#62716L, login\_dt#62732, login\_age#62749L, date\_format(cast(dob\_dt#62701 as timestamp), yyyy-MM-dd, Some(Asia/Kolkata)) AS dob\_date\_only#62956]

+- Project [gender#62602, city#62606, state#62611, title#62617, first#62624, last#62632, latitude#62641, longitude#62651, email#62662, uuid#62674, username#62687, dob\_dt#62701, dob\_age#62716L, login\_dt#62732, login\_age#62749L, date\_format(cast(dob\_dt#62701 as timestamp), yyyy-MM-dd, Some(Asia/Kolkata)) AS dob\_date\_only#62938]

+- Project [gender#62602, city#62606, state#62611, title#62617, first#62624, last#62632, latitude#62641, longitude#62651, email#62662, uuid#62674, username#62687, dob\_dt#62701, dob\_age#62716L, login\_dt#62732, login\_age#62749L, date\_format(cast(dob\_dt#62701 as timestamp), yyyy-MM-dd, Some(Asia/Kolkata)) AS dob\_date\_only#62921]

+- Project [gender#62602, city#62606, state#62611, title#62617, first#62624, last#62632, latitude#62641, longitude#62651, email#62662, uuid#62674, username#62687, dob\_dt#62701, dob\_age#62716L, login\_dt#62732, login\_age#62749L]

+- Project [exploded\_cols#62600, gender#62602, city#62606, state#62611, title#62617, first#62624, last#62632, latitude#62641, longitude#62651, email#62662, uuid#62674, username#62687, dob\_dt#62701, dob\_age#62716L, login\_dt#62732, exploded\_cols#62600.registered.age AS login\_age#62749L]

+- Project [exploded\_cols#62600, gender#62602, city#62606, state#62611, title#62617, first#62624, last#62632, latitude#62641, longitude#62651, email#62662, uuid#62674, username#62687, dob\_dt#62701, dob\_age#62716L, exploded\_cols#62600.registered.date AS login\_dt#62732]

+- Project [exploded\_cols#62600, gender#62602, city#62606, state#62611, title#62617, first#62624, last#62632, latitude#62641, longitude#62651, email#62662, uuid#62674, username#62687, dob\_dt#62701, exploded\_cols#62600.dob.age AS dob\_age#62716L]

+- Project [exploded\_cols#62600, gender#62602, city#62606, state#62611, title#62617, first#62624, last#62632, latitude#62641, longitude#62651, email#62662, uuid#62674, username#62687, exploded\_cols#62600.dob.date AS dob\_dt#62701]

+- Project [exploded\_cols#62600, gender#62602, city#62606, state#62611, title#62617, first#62624, last#62632, latitude#62641, longitude#62651, email#62662, uuid#62674, exploded\_cols#62600.login.username AS username#62687]

+- Project [exploded\_cols#62600, gender#62602, city#62606, state#62611, title#62617, first#62624, last#62632, latitude#62641, longitude#62651, email#62662, exploded\_cols#62600.login.uuid AS uuid#62674]

+- Project [exploded\_cols#62600, gender#62602, city#62606, state#62611, title#62617, first#62624, last#62632, latitude#62641, longitude#62651, exploded\_cols#62600.email AS email#62662]

+- Project [exploded\_cols#62600, gender#62602, city#62606, state#62611, title#62617, first#62624, last#62632, latitude#62641, exploded\_cols#62600.location.coordinates.longitude AS longitude#62651]

+- Project [exploded\_cols#62600, gender#62602, city#62606, state#62611, title#62617, first#62624, last#62632, exploded\_cols#62600.location.coordinates.latitude AS latitude#62641]

+- Project [exploded\_cols#62600, gender#62602, city#62606, state#62611, title#62617, first#62624, exploded\_cols#62600.name.last AS last#62632]

+- Project [exploded\_cols#62600, gender#62602, city#62606, state#62611, title#62617, exploded\_cols#62600.name.first AS first#62624]

+- Project [exploded\_cols#62600, gender#62602, city#62606, state#62611, exploded\_cols#62600.name.title AS title#62617]

+- Project [exploded\_cols#62600, gender#62602, city#62606, exploded\_cols#62600.location.state AS state#62611]

+- Project [exploded\_cols#62600, gender#62602, exploded\_cols#62600.location.city AS city#62606]

+- Project [exploded\_cols#62600, exploded\_cols#62600.gender AS gender#62602]

+- Project [exploded\_cols#62600]

+- Generate explode(results#62590), false, [exploded\_cols#62600]

+- Relation [info#62589,results#62590] json

>>> spark.sql("select date(dob\_date\_only) from view1 ").show(5)

+-------------+

|dob\_date\_only|

+-------------+

| 1961-12-27|

| 1947-03-07|

| 1975-08-23|

| 1957-11-17|

+-------------+

>>> spark.sql("select date(dob\_dt) from view1 ").show(5)

+----------+

| dob\_dt|

+----------+

|1961-12-27|

|1947-03-07|

|1975-08-23|

|1957-11-16|

+----------+

>>> df = flattened\_df.withColumn('password\_md5', md5(col('password')))

Traceback (most recent call last):

File "<stdin>", line 1, in <module>

File "/usr/local/spark/python/pyspark/sql/dataframe.py", line 2478, in withColumn

return DataFrame(self.\_jdf.withColumn(colName, col.\_jc), self.sql\_ctx)

File "/usr/local/spark/python/lib/py4j-0.10.9.3-src.zip/py4j/java\_gateway.py", line 1322, in \_\_call\_\_

File "/usr/local/spark/python/pyspark/sql/utils.py", line 117, in deco

raise converted from None

pyspark.sql.utils.AnalysisException: cannot resolve 'password' given input columns: [city, dob\_age, dob\_date\_only, dob\_dt, email, first, gender, last, latitude, login\_age, login\_dt, longitude, state, title, username, uuid];

'Project [gender#62602, city#62606, state#62611, title#62617, first#62624, last#62632, latitude#62641, longitude#62651, email#62662, uuid#62674, username#62687, dob\_dt#62701, dob\_age#62716L, login\_dt#62732, login\_age#62749L, dob\_date\_only#63008, md5('password) AS password\_md5#63262]

+- Project [gender#62602, city#62606, state#62611, title#62617, first#62624, last#62632, latitude#62641, longitude#62651, email#62662, uuid#62674, username#62687, dob\_dt#62701, dob\_age#62716L, login\_dt#62732, login\_age#62749L, date\_format(cast(dob\_dt#62701 as timestamp), yyyy-MM-dd, Some(Asia/Kolkata)) AS dob\_date\_only#63008]

+- Project [gender#62602, city#62606, state#62611, title#62617, first#62624, last#62632, latitude#62641, longitude#62651, email#62662, uuid#62674, username#62687, dob\_dt#62701, dob\_age#62716L, login\_dt#62732, login\_age#62749L, date\_format(cast(dob\_dt#62701 as timestamp), yyyy-MM-dd, Some(Asia/Kolkata)) AS dob\_date\_only#62991]

+- Project [gender#62602, city#62606, state#62611, title#62617, first#62624, last#62632, latitude#62641, longitude#62651, email#62662, uuid#62674, username#62687, dob\_dt#62701, dob\_age#62716L, login\_dt#62732, login\_age#62749L, date\_format(cast(dob\_dt#62701 as timestamp), yyyy-MM-dd, Some(Asia/Kolkata)) AS dob\_date\_only#62973]

+- Project [gender#62602, city#62606, state#62611, title#62617, first#62624, last#62632, latitude#62641, longitude#62651, email#62662, uuid#62674, username#62687, dob\_dt#62701, dob\_age#62716L, login\_dt#62732, login\_age#62749L, date\_format(cast(dob\_dt#62701 as timestamp), yyyy-MM-dd, Some(Asia/Kolkata)) AS dob\_date\_only#62956]

+- Project [gender#62602, city#62606, state#62611, title#62617, first#62624, last#62632, latitude#62641, longitude#62651, email#62662, uuid#62674, username#62687, dob\_dt#62701, dob\_age#62716L, login\_dt#62732, login\_age#62749L, date\_format(cast(dob\_dt#62701 as timestamp), yyyy-MM-dd, Some(Asia/Kolkata)) AS dob\_date\_only#62938]

+- Project [gender#62602, city#62606, state#62611, title#62617, first#62624, last#62632, latitude#62641, longitude#62651, email#62662, uuid#62674, username#62687, dob\_dt#62701, dob\_age#62716L, login\_dt#62732, login\_age#62749L, date\_format(cast(dob\_dt#62701 as timestamp), yyyy-MM-dd, Some(Asia/Kolkata)) AS dob\_date\_only#62921]

+- Project [gender#62602, city#62606, state#62611, title#62617, first#62624, last#62632, latitude#62641, longitude#62651, email#62662, uuid#62674, username#62687, dob\_dt#62701, dob\_age#62716L, login\_dt#62732, login\_age#62749L]

+- Project [exploded\_cols#62600, gender#62602, city#62606, state#62611, title#62617, first#62624, last#62632, latitude#62641, longitude#62651, email#62662, uuid#62674, username#62687, dob\_dt#62701, dob\_age#62716L, login\_dt#62732, exploded\_cols#62600.registered.age AS login\_age#62749L]

+- Project [exploded\_cols#62600, gender#62602, city#62606, state#62611, title#62617, first#62624, last#62632, latitude#62641, longitude#62651, email#62662, uuid#62674, username#62687, dob\_dt#62701, dob\_age#62716L, exploded\_cols#62600.registered.date AS login\_dt#62732]

+- Project [exploded\_cols#62600, gender#62602, city#62606, state#62611, title#62617, first#62624, last#62632, latitude#62641, longitude#62651, email#62662, uuid#62674, username#62687, dob\_dt#62701, exploded\_cols#62600.dob.age AS dob\_age#62716L]

+- Project [exploded\_cols#62600, gender#62602, city#62606, state#62611, title#62617, first#62624, last#62632, latitude#62641, longitude#62651, email#62662, uuid#62674, username#62687, exploded\_cols#62600.dob.date AS dob\_dt#62701]

+- Project [exploded\_cols#62600, gender#62602, city#62606, state#62611, title#62617, first#62624, last#62632, latitude#62641, longitude#62651, email#62662, uuid#62674, exploded\_cols#62600.login.username AS username#62687]

+- Project [exploded\_cols#62600, gender#62602, city#62606, state#62611, title#62617, first#62624, last#62632, latitude#62641, longitude#62651, email#62662, exploded\_cols#62600.login.uuid AS uuid#62674]

+- Project [exploded\_cols#62600, gender#62602, city#62606, state#62611, title#62617, first#62624, last#62632, latitude#62641, longitude#62651, exploded\_cols#62600.email AS email#62662]

+- Project [exploded\_cols#62600, gender#62602, city#62606, state#62611, title#62617, first#62624, last#62632, latitude#62641, exploded\_cols#62600.location.coordinates.longitude AS longitude#62651]

+- Project [exploded\_cols#62600, gender#62602, city#62606, state#62611, title#62617, first#62624, last#62632, exploded\_cols#62600.location.coordinates.latitude AS latitude#62641]

+- Project [exploded\_cols#62600, gender#62602, city#62606, state#62611, title#62617, first#62624, exploded\_cols#62600.name.last AS last#62632]

+- Project [exploded\_cols#62600, gender#62602, city#62606, state#62611, title#62617, exploded\_cols#62600.name.first AS first#62624]

+- Project [exploded\_cols#62600, gender#62602, city#62606, state#62611, exploded\_cols#62600.name.title AS title#62617]

+- Project [exploded\_cols#62600, gender#62602, city#62606, exploded\_cols#62600.location.state AS state#62611]

+- Project [exploded\_cols#62600, gender#62602, exploded\_cols#62600.location.city AS city#62606]

+- Project [exploded\_cols#62600, exploded\_cols#62600.gender AS gender#62602]

+- Project [exploded\_cols#62600]

+- Generate explode(results#62590), false, [exploded\_cols#62600]

+- Relation [info#62589,results#62590] json

>>>

>>> flattened\_df.show(4)

+------+--------------+-----------+-----+------+---------+--------+---------+--------------------+--------------------+--------------+--------------------+-------+--------------------+---------+-------------+

|gender| city| state|title| first| last|latitude|longitude| email| uuid| username| dob\_dt|dob\_age| login\_dt|login\_age|dob\_date\_only|

+------+--------------+-----------+-----+------+---------+--------+---------+--------------------+--------------------+--------------+--------------------+-------+--------------------+---------+-------------+

|female|Carrickmacross| Longford| Miss| Becky|Carpenter| 19.0157| 56.4959|becky.carpenter@e...|21312f38-cdf4-4fe...| bluefrog953|1961-12-27T07:26:...| 58|2003-02-07T11:28:...| 16| 1961-12-27|

|female| Bjerka| Rogaland| Miss| Freja| Hestetun| 49.8515| 166.0684|freja.hestetun@ex...|553a7a09-9b0b-41e...| browntiger686|1947-03-07T14:10:...| 76|2012-09-10T03:07:...| 7| 1947-03-07|

| male| Roslev|Hovedstaden| Mr|Alfred| Møller|-36.9245| -0.9917|alfred.moller@exa...|18bb2a51-14d3-47d...|orangegoose716|1975-08-23T13:17:...| 47|2014-04-06T14:09:...| 5| 1975-08-23|

|female| Trabzon| Elazığ| Miss| Oya| Ilıcalı| 72.4704| 66.0216|oya.ilicali@examp...|c74c81c4-8100-487...| redwolf988|1957-11-16T21:46:...| 65|2012-05-11T14:17:...| 7| 1957-11-17|

+------+--------------+-----------+-----+------+---------+--------+---------+--------------------+--------------------+--------------+--------------------+-------+--------------------+---------+-------------+

>>> df = flattened\_df.withColumn('password\_md5', md5(col('uuid')))

>>> df.drop("uuid")

DataFrame[gender: string, city: string, state: string, title: string, first: string, last: string, latitude: string, longitude: string, email: string, username: string, dob\_dt: string, dob\_age: bigint, login\_dt: string, login\_age: bigint, dob\_date\_only: string, password\_md5: string]

>>> df.show(4)

+------+--------------+-----------+-----+------+---------+--------+---------+--------------------+--------------------+--------------+--------------------+-------+--------------------+---------+-------------+--------------------+

|gender| city| state|title| first| last|latitude|longitude| email| uuid| username| dob\_dt|dob\_age| login\_dt|login\_age|dob\_date\_only| password\_md5|

+------+--------------+-----------+-----+------+---------+--------+---------+--------------------+--------------------+--------------+--------------------+-------+--------------------+---------+-------------+--------------------+

|female|Carrickmacross| Longford| Miss| Becky|Carpenter| 19.0157| 56.4959|becky.carpenter@e...|21312f38-cdf4-4fe...| bluefrog953|1961-12-27T07:26:...| 58|2003-02-07T11:28:...| 16| 1961-12-27|15448ee18f5970ccf...|

|female| Bjerka| Rogaland| Miss| Freja| Hestetun| 49.8515| 166.0684|freja.hestetun@ex...|553a7a09-9b0b-41e...| browntiger686|1947-03-07T14:10:...| 76|2012-09-10T03:07:...| 7| 1947-03-07|f36987978f4f1c537...|

| male| Roslev|Hovedstaden| Mr|Alfred| Møller|-36.9245| -0.9917|alfred.moller@exa...|18bb2a51-14d3-47d...|orangegoose716|1975-08-23T13:17:...| 47|2014-04-06T14:09:...| 5| 1975-08-23|af1db29687ba8d038...|

|female| Trabzon| Elazığ| Miss| Oya| Ilıcalı| 72.4704| 66.0216|oya.ilicali@examp...|c74c81c4-8100-487...| redwolf988|1957-11-16T21:46:...| 65|2012-05-11T14:17:...| 7| 1957-11-17|829399af08f9fbb85...|

+------+--------------+-----------+-----+------+---------+--------+---------+--------------------+--------------------+--------------+--------------------+-------+--------------------+---------+-------------+--------------------+

>>> df = df.withColumn(

... 'formatted\_cell\_number',

... regexp\_replace(col('cell\_number'), '([0-9]{3})([0-9]{3})([0-9]{4})', '$1-$2-$3')

... )

Traceback (most recent call last):

File "<stdin>", line 3, in <module>

File "/usr/local/spark/python/pyspark/sql/dataframe.py", line 2478, in withColumn

return DataFrame(self.\_jdf.withColumn(colName, col.\_jc), self.sql\_ctx)

File "/usr/local/spark/python/lib/py4j-0.10.9.3-src.zip/py4j/java\_gateway.py", line 1322, in \_\_call\_\_

File "/usr/local/spark/python/pyspark/sql/utils.py", line 117, in deco

raise converted from None

pyspark.sql.utils.AnalysisException: cannot resolve 'cell\_number' given input columns: [city, dob\_age, dob\_date\_only, dob\_dt, email, first, gender, last, latitude, login\_age, login\_dt, longitude, password\_md5, state, title, username, uuid];

'Project [gender#62602, city#62606, state#62611, title#62617, first#62624, last#62632, latitude#62641, longitude#62651, email#62662, uuid#62674, username#62687, dob\_dt#62701, dob\_age#62716L, login\_dt#62732, login\_age#62749L, dob\_date\_only#63008, password\_md5#63329, regexp\_replace('cell\_number, ([0-9]{3})([0-9]{3})([0-9]{4}), $1-$2-$3, 1) AS formatted\_cell\_number#63433]

+- Project [gender#62602, city#62606, state#62611, title#62617, first#62624, last#62632, latitude#62641, longitude#62651, email#62662, uuid#62674, username#62687, dob\_dt#62701, dob\_age#62716L, login\_dt#62732, login\_age#62749L, dob\_date\_only#63008, md5(cast(uuid#62674 as binary)) AS password\_md5#63329]

+- Project [gender#62602, city#62606, state#62611, title#62617, first#62624, last#62632, latitude#62641, longitude#62651, email#62662, uuid#62674, username#62687, dob\_dt#62701, dob\_age#62716L, login\_dt#62732, login\_age#62749L, date\_format(cast(dob\_dt#62701 as timestamp), yyyy-MM-dd, Some(Asia/Kolkata)) AS dob\_date\_only#63008]

+- Project [gender#62602, city#62606, state#62611, title#62617, first#62624, last#62632, latitude#62641, longitude#62651, email#62662, uuid#62674, username#62687, dob\_dt#62701, dob\_age#62716L, login\_dt#62732, login\_age#62749L, date\_format(cast(dob\_dt#62701 as timestamp), yyyy-MM-dd, Some(Asia/Kolkata)) AS dob\_date\_only#62991]

+- Project [gender#62602, city#62606, state#62611, title#62617, first#62624, last#62632, latitude#62641, longitude#62651, email#62662, uuid#62674, username#62687, dob\_dt#62701, dob\_age#62716L, login\_dt#62732, login\_age#62749L, date\_format(cast(dob\_dt#62701 as timestamp), yyyy-MM-dd, Some(Asia/Kolkata)) AS dob\_date\_only#62973]

+- Project [gender#62602, city#62606, state#62611, title#62617, first#62624, last#62632, latitude#62641, longitude#62651, email#62662, uuid#62674, username#62687, dob\_dt#62701, dob\_age#62716L, login\_dt#62732, login\_age#62749L, date\_format(cast(dob\_dt#62701 as timestamp), yyyy-MM-dd, Some(Asia/Kolkata)) AS dob\_date\_only#62956]

+- Project [gender#62602, city#62606, state#62611, title#62617, first#62624, last#62632, latitude#62641, longitude#62651, email#62662, uuid#62674, username#62687, dob\_dt#62701, dob\_age#62716L, login\_dt#62732, login\_age#62749L, date\_format(cast(dob\_dt#62701 as timestamp), yyyy-MM-dd, Some(Asia/Kolkata)) AS dob\_date\_only#62938]

+- Project [gender#62602, city#62606, state#62611, title#62617, first#62624, last#62632, latitude#62641, longitude#62651, email#62662, uuid#62674, username#62687, dob\_dt#62701, dob\_age#62716L, login\_dt#62732, login\_age#62749L, date\_format(cast(dob\_dt#62701 as timestamp), yyyy-MM-dd, Some(Asia/Kolkata)) AS dob\_date\_only#62921]

+- Project [gender#62602, city#62606, state#62611, title#62617, first#62624, last#62632, latitude#62641, longitude#62651, email#62662, uuid#62674, username#62687, dob\_dt#62701, dob\_age#62716L, login\_dt#62732, login\_age#62749L]

+- Project [exploded\_cols#62600, gender#62602, city#62606, state#62611, title#62617, first#62624, last#62632, latitude#62641, longitude#62651, email#62662, uuid#62674, username#62687, dob\_dt#62701, dob\_age#62716L, login\_dt#62732, exploded\_cols#62600.registered.age AS login\_age#62749L]

+- Project [exploded\_cols#62600, gender#62602, city#62606, state#62611, title#62617, first#62624, last#62632, latitude#62641, longitude#62651, email#62662, uuid#62674, username#62687, dob\_dt#62701, dob\_age#62716L, exploded\_cols#62600.registered.date AS login\_dt#62732]

+- Project [exploded\_cols#62600, gender#62602, city#62606, state#62611, title#62617, first#62624, last#62632, latitude#62641, longitude#62651, email#62662, uuid#62674, username#62687, dob\_dt#62701, exploded\_cols#62600.dob.age AS dob\_age#62716L]

+- Project [exploded\_cols#62600, gender#62602, city#62606, state#62611, title#62617, first#62624, last#62632, latitude#62641, longitude#62651, email#62662, uuid#62674, username#62687, exploded\_cols#62600.dob.date AS dob\_dt#62701]

+- Project [exploded\_cols#62600, gender#62602, city#62606, state#62611, title#62617, first#62624, last#62632, latitude#62641, longitude#62651, email#62662, uuid#62674, exploded\_cols#62600.login.username AS username#62687]

+- Project [exploded\_cols#62600, gender#62602, city#62606, state#62611, title#62617, first#62624, last#62632, latitude#62641, longitude#62651, email#62662, exploded\_cols#62600.login.uuid AS uuid#62674]

+- Project [exploded\_cols#62600, gender#62602, city#62606, state#62611, title#62617, first#62624, last#62632, latitude#62641, longitude#62651, exploded\_cols#62600.email AS email#62662]

+- Project [exploded\_cols#62600, gender#62602, city#62606, state#62611, title#62617, first#62624, last#62632, latitude#62641, exploded\_cols#62600.location.coordinates.longitude AS longitude#62651]

+- Project [exploded\_cols#62600, gender#62602, city#62606, state#62611, title#62617, first#62624, last#62632, exploded\_cols#62600.location.coordinates.latitude AS latitude#62641]

+- Project [exploded\_cols#62600, gender#62602, city#62606, state#62611, title#62617, first#62624, exploded\_cols#62600.name.last AS last#62632]

+- Project [exploded\_cols#62600, gender#62602, city#62606, state#62611, title#62617, exploded\_cols#62600.name.first AS first#62624]

+- Project [exploded\_cols#62600, gender#62602, city#62606, state#62611, exploded\_cols#62600.name.title AS title#62617]

+- Project [exploded\_cols#62600, gender#62602, city#62606, exploded\_cols#62600.location.state AS state#62611]

+- Project [exploded\_cols#62600, gender#62602, exploded\_cols#62600.location.city AS city#62606]

+- Project [exploded\_cols#62600, exploded\_cols#62600.gender AS gender#62602]

+- Project [exploded\_cols#62600]

+- Generate explode(results#62590), false, [exploded\_cols#62600]

+- Relation [info#62589,results#62590] json

>>> df.show(3)

+------+--------------+-----------+-----+------+---------+--------+---------+--------------------+--------------------+--------------+--------------------+-------+--------------------+---------+-------------+--------------------+

|gender| city| state|title| first| last|latitude|longitude| email| uuid| username| dob\_dt|dob\_age| login\_dt|login\_age|dob\_date\_only| password\_md5|

+------+--------------+-----------+-----+------+---------+--------+---------+--------------------+--------------------+--------------+--------------------+-------+--------------------+---------+-------------+--------------------+

|female|Carrickmacross| Longford| Miss| Becky|Carpenter| 19.0157| 56.4959|becky.carpenter@e...|21312f38-cdf4-4fe...| bluefrog953|1961-12-27T07:26:...| 58|2003-02-07T11:28:...| 16| 1961-12-27|15448ee18f5970ccf...|

|female| Bjerka| Rogaland| Miss| Freja| Hestetun| 49.8515| 166.0684|freja.hestetun@ex...|553a7a09-9b0b-41e...| browntiger686|1947-03-07T14:10:...| 76|2012-09-10T03:07:...| 7| 1947-03-07|f36987978f4f1c537...|

| male| Roslev|Hovedstaden| Mr|Alfred| Møller|-36.9245| -0.9917|alfred.moller@exa...|18bb2a51-14d3-47d...|orangegoose716|1975-08-23T13:17:...| 47|2014-04-06T14:09:...| 5| 1975-08-23|af1db29687ba8d038...|

+------+--------------+-----------+-----+------+---------+--------+---------+--------------------+--------------------+--------------+--------------------+-------+--------------------+---------+-------------+--------------------+

only showing top 3 rows

>>>

>>> df = df.withColumn("dob", F.to\_date("dob\_date\_only", "yyyy-MM-dd"))

>>>

>>> # Calculate age from dob\_date\_only

... df = df.withColumn(

... "calculated\_age",

... F.floor(F.datediff(F.current\_date(), F.col("dob\_date\_only")) / 365.25)

... )

>>>

>>> # Compare calculated\_age with existing age

... df = df.withColumn(

... "age\_correct",

... F.col("dob\_age") == F.col("calculated\_age")

... )

>>> df.show(3)

+------+--------------+-----------+-----+------+---------+--------+---------+--------------------+--------------------+--------------+--------------------+-------+--------------------+---------+-------------+--------------------+----------+--------------+-----------+

|gender| city| state|title| first| last|latitude|longitude| email| uuid| username| dob\_dt|dob\_age| login\_dt|login\_age|dob\_date\_only| password\_md5| dob|calculated\_age|age\_correct|

+------+--------------+-----------+-----+------+---------+--------+---------+--------------------+--------------------+--------------+--------------------+-------+--------------------+---------+-------------+--------------------+----------+--------------+-----------+

|female|Carrickmacross| Longford| Miss| Becky|Carpenter| 19.0157| 56.4959|becky.carpenter@e...|21312f38-cdf4-4fe...| bluefrog953|1961-12-27T07:26:...| 58|2003-02-07T11:28:...| 16| 1961-12-27|15448ee18f5970ccf...|1961-12-27| 62| false|

|female| Bjerka| Rogaland| Miss| Freja| Hestetun| 49.8515| 166.0684|freja.hestetun@ex...|553a7a09-9b0b-41e...| browntiger686|1947-03-07T14:10:...| 76|2012-09-10T03:07:...| 7| 1947-03-07|f36987978f4f1c537...|1947-03-07| 77| false|

| male| Roslev|Hovedstaden| Mr|Alfred| Møller|-36.9245| -0.9917|alfred.moller@exa...|18bb2a51-14d3-47d...|orangegoose716|1975-08-23T13:17:...| 47|2014-04-06T14:09:...| 5| 1975-08-23|af1db29687ba8d038...|1975-08-23| 48| false|

+------+--------------+-----------+-----+------+---------+--------+---------+--------------------+--------------------+--------------+--------------------+-------+--------------------+---------+-------------+--------------------+----------+--------------+-----------+

only showing top 3 rows

>>> df.select("email").show(4)

+--------------------+

| email|

+--------------------+

|becky.carpenter@e...|

|freja.hestetun@ex...|

|alfred.moller@exa...|

|oya.ilicali@examp...|

+--------------------+

>>> df.select("email").show(4,True)

+--------------------+

| email|

+--------------------+

|becky.carpenter@e...|

|freja.hestetun@ex...|

|alfred.moller@exa...|

|oya.ilicali@examp...|

+--------------------+

>>> df.select("email").show(4,False)

+---------------------------+

|email |

+---------------------------+

|becky.carpenter@example.com|

|freja.hestetun@example.com |

|alfred.moller@example.com |

|oya.ilicali@example.com |

+---------------------------+

>>>

>>> result\_df = df.withColumn(

... "expected\_email",

... concat\_ws( lower(col('first')) +lit('.') + lower(col('last'))) + lit('@example.com')

... ).withColumn(

... "email\_status",

... when(lower(col('email')) == col('expected\_email'), 'Correct').otherwise('Incorrect')

... )

Traceback (most recent call last):

File "<stdin>", line 3, in <module>

File "/usr/local/spark/python/pyspark/sql/functions.py", line 2647, in concat\_ws

return Column(sc.\_jvm.functions.concat\_ws(sep, \_to\_seq(sc, cols, \_to\_java\_column)))

File "/usr/local/spark/python/lib/py4j-0.10.9.3-src.zip/py4j/java\_gateway.py", line 1313, in \_\_call\_\_

File "/usr/local/spark/python/lib/py4j-0.10.9.3-src.zip/py4j/java\_gateway.py", line 1277, in \_build\_args

File "/usr/local/spark/python/lib/py4j-0.10.9.3-src.zip/py4j/java\_gateway.py", line 1264, in \_get\_args

File "/usr/local/spark/python/lib/py4j-0.10.9.3-src.zip/py4j/java\_collections.py", line 510, in convert

File "/usr/local/spark/python/pyspark/sql/column.py", line 463, in \_\_iter\_\_

raise TypeError("Column is not iterable")

TypeError: Column is not iterable

>>> result\_df = df.withColumn(

... "expected\_email",

... concat(

... lower(col('first')),

... lit('.'),

... lower(col('last')),

... lit('@example.com')

... )

... ).withColumn(

... "email\_status",

... when(lower(col('email')) == col('expected\_email'), 'Correct').otherwise('Incorrect')

... )

>>> result\_df.show(4)

+------+--------------+-----------+-----+------+---------+--------+---------+--------------------+--------------------+--------------+--------------------+-------+--------------------+---------+-------------+--------------------+----------+--------------+-----------+--------------------+------------+

|gender| city| state|title| first| last|latitude|longitude| email| uuid| username| dob\_dt|dob\_age| login\_dt|login\_age|dob\_date\_only| password\_md5| dob|calculated\_age|age\_correct| expected\_email|email\_status|

+------+--------------+-----------+-----+------+---------+--------+---------+--------------------+--------------------+--------------+--------------------+-------+--------------------+---------+-------------+--------------------+----------+--------------+-----------+--------------------+------------+

|female|Carrickmacross| Longford| Miss| Becky|Carpenter| 19.0157| 56.4959|becky.carpenter@e...|21312f38-cdf4-4fe...| bluefrog953|1961-12-27T07:26:...| 58|2003-02-07T11:28:...| 16| 1961-12-27|15448ee18f5970ccf...|1961-12-27| 62| false|becky.carpenter@e...| Correct|

|female| Bjerka| Rogaland| Miss| Freja| Hestetun| 49.8515| 166.0684|freja.hestetun@ex...|553a7a09-9b0b-41e...| browntiger686|1947-03-07T14:10:...| 76|2012-09-10T03:07:...| 7| 1947-03-07|f36987978f4f1c537...|1947-03-07| 77| false|freja.hestetun@ex...| Correct|

| male| Roslev|Hovedstaden| Mr|Alfred| Møller|-36.9245| -0.9917|alfred.moller@exa...|18bb2a51-14d3-47d...|orangegoose716|1975-08-23T13:17:...| 47|2014-04-06T14:09:...| 5| 1975-08-23|af1db29687ba8d038...|1975-08-23| 48| false|alfred.møller@exa...| Incorrect|

|female| Trabzon| Elazığ| Miss| Oya| Ilıcalı| 72.4704| 66.0216|oya.ilicali@examp...|c74c81c4-8100-487...| redwolf988|1957-11-16T21:46:...| 65|2012-05-11T14:17:...| 7| 1957-11-17|829399af08f9fbb85...|1957-11-17| 66| false|oya.ilıcalı@examp...| Incorrect|

+------+--------------+-----------+-----+------+---------+--------+---------+--------------------+--------------------+--------------+--------------------+-------+--------------------+---------+-------------+--------------------+----------+--------------+-----------+--------------------+------------+

>>> script -a my\_terminal\_session.log

File "<stdin>", line 1

script -a my\_terminal\_session.log

^

SyntaxError: invalid syntax

>>> sc.version | tee -a my\_command\_output.log

File "<stdin>", line 1

sc.version | tee -a my\_command\_output.log

^

SyntaxError: invalid syntax

>>>

Using Python version 3.6.8 (default, Nov 14 2023 16:29:52)

Spark context Web UI available at http://192.168.122.1:4040

Spark context available as 'sc' (master = local[\*], app id = local-1721790185318).

SparkSession available as 'spark'.

>>>

>>> from pyspark.sql import \*

>>> spark=SparkSession.builder.master("local[2]").appName("Hackathon").enableHiveSupport().getOrCreate()

>>> sc=spark.sparkContext

>>>

>>> from pyspark.sql import \*

>>> spark=SparkSession.builder.master("local[2]").appName("Hackathon").enableHiveSupport().getOrCreate()

>>> sc=spark.sparkContext

>>> df\_1=spark.read.csv("/home/hduser/Desktop/Hackathon/manuf\_data.csv",header=True,inferSchema=True)

Traceback (most recent call last):

File "<stdin>", line 1, in <module>

File "/usr/local/spark/python/pyspark/sql/readwriter.py", line 410, in csv

return self.\_df(self.\_jreader.csv(self.\_spark.\_sc.\_jvm.PythonUtils.toSeq(path)))

File "/usr/local/spark/python/lib/py4j-0.10.9.3-src.zip/py4j/java\_gateway.py", line 1322, in \_\_call\_\_

File "/usr/local/spark/python/pyspark/sql/utils.py", line 117, in deco

raise converted from None

pyspark.sql.utils.AnalysisException: Path does not exist: hdfs://localhost:54310/home/hduser/Desktop/Hackathon/manuf\_data.csv

>>> df\_1=spark.read.csv("file:///home/hduser/Desktop/Hackathon/manuf\_data.csv",header=True,inferSchema=True)

>>> df\_1.show(1)

24/07/24 08:36:09 WARN csv.CSVHeaderChecker: CSV header does not conform to the schema.

Header: timestamp, prod\_id, station\_id, sensor, value, station\_id, prodid, part\_id

Schema: timestamp, prod\_id, station\_id2, sensor, value, station\_id5, prodid, part\_id

Expected: station\_id2 but found: station\_id

CSV file: file:///home/hduser/Desktop/Hackathon/manuf\_data.csv

+---------+-------+-----------+------+------------+-----------+-------+--------------------+

|timestamp|prod\_id|station\_id2|sensor| value|station\_id5| prodid| part\_id|

+---------+-------+-----------+------+------------+-----------+-------+--------------------+

| 05:31.4|ACME123| 234| A|-0.799545863| 234|ACME123|2f10d3bf-e5ed-450...|

+---------+-------+-----------+------+------------+-----------+-------+--------------------+

only showing top 1 row

>>>

>>> # Data munging-Remove rows with missing values

... df\_clean = df\_1.dropna()

>>> df\_clean.show(1)

24/07/24 08:44:24 WARN csv.CSVHeaderChecker: CSV header does not conform to the schema.

Header: timestamp, prod\_id, station\_id, sensor, value, station\_id, prodid, part\_id

Schema: timestamp, prod\_id, station\_id2, sensor, value, station\_id5, prodid, part\_id

Expected: station\_id2 but found: station\_id

CSV file: file:///home/hduser/Desktop/Hackathon/manuf\_data.csv

+---------+-------+-----------+------+------------+-----------+-------+--------------------+

|timestamp|prod\_id|station\_id2|sensor| value|station\_id5| prodid| part\_id|

+---------+-------+-----------+------+------------+-----------+-------+--------------------+

| 05:31.4|ACME123| 234| A|-0.799545863| 234|ACME123|2f10d3bf-e5ed-450...|

+---------+-------+-----------+------+------------+-----------+-------+--------------------+

only showing top 1 row

>>> # Data wrangling: Rename columns or change data types

... df\_final = df\_enriched.withColumnRenamed("station\_id2", "station\_id2")

Traceback (most recent call last):

File "<stdin>", line 2, in <module>

NameError: name 'df\_enriched' is not defined

>>> df\_final = df\_clean.withColumnRenamed("station\_id2", "station\_id2")

>>> df\_final.show(1)

24/07/24 08:46:17 WARN csv.CSVHeaderChecker: CSV header does not conform to the schema.

Header: timestamp, prod\_id, station\_id, sensor, value, station\_id, prodid, part\_id

Schema: timestamp, prod\_id, station\_id2, sensor, value, station\_id5, prodid, part\_id

Expected: station\_id2 but found: station\_id

CSV file: file:///home/hduser/Desktop/Hackathon/manuf\_data.csv

+---------+-------+-----------+------+------------+-----------+-------+--------------------+

|timestamp|prod\_id|station\_id2|sensor| value|station\_id5| prodid| part\_id|

+---------+-------+-----------+------+------------+-----------+-------+--------------------+

| 05:31.4|ACME123| 234| A|-0.799545863| 234|ACME123|2f10d3bf-e5ed-450...|

+---------+-------+-----------+------+------------+-----------+-------+--------------------+

only showing top 1 row

>>> df\_final = df\_clean.withColumnRenamed("station\_id2", "station\_id")

>>> df\_final.show(1)

24/07/24 08:46:33 WARN csv.CSVHeaderChecker: CSV header does not conform to the schema.

Header: timestamp, prod\_id, station\_id, sensor, value, station\_id, prodid, part\_id

Schema: timestamp, prod\_id, station\_id2, sensor, value, station\_id5, prodid, part\_id

Expected: station\_id2 but found: station\_id

CSV file: file:///home/hduser/Desktop/Hackathon/manuf\_data.csv

+---------+-------+----------+------+------------+-----------+-------+--------------------+

|timestamp|prod\_id|station\_id|sensor| value|station\_id5| prodid| part\_id|

+---------+-------+----------+------+------------+-----------+-------+--------------------+

| 05:31.4|ACME123| 234| A|-0.799545863| 234|ACME123|2f10d3bf-e5ed-450...|

+---------+-------+----------+------+------------+-----------+-------+--------------------+

only showing top 1 row

>>>

>>> #data enrichment

... df\_enriched = df\_final.withColumn("sensor\_flag", when(col("sensor") == "Specific\_Sensor\_Value", lit(1)).otherwise(lit(0)))

Traceback (most recent call last):

File "<stdin>", line 2, in <module>

NameError: name 'when' is not defined

>>>

>>> # Create a new column combining 'prod\_id' and 'station\_id2'

... df\_enriched = df\_enriched.withColumn("prod\_station\_combined", col("prod\_id").cast("string") + "\_" + col("station\_id2").cast("string"))

Traceback (most recent call last):

File "<stdin>", line 2, in <module>

NameError: name 'df\_enriched' is not defined

>>>

>>> from pyspark.sql.functions import col, lit, to\_timestamp, when

>>> # Data Enrichment

...

>>> # Convert 'timestamp'

... df\_enriched = df\_final.withColumn("timestamp", to\_timestamp(col("timestamp"), "yyyy-MM-dd HH:mm:ss"))

>>>

>>> # Extract additional time-related features from 'timestamp'

... df\_enriched = df\_enriched.withColumn("year", col("timestamp").cast("date").substr(1, 4))

>>> df\_enriched = df\_enriched.withColumn("month", col("timestamp").cast("date").substr(6, 2))

>>> df\_enriched = df\_enriched.withColumn("day", col("timestamp").cast("date").substr(9, 2))

>>>

>>> # Example: Create a new column combining 'prod\_id' and 'station\_id2'

... df\_enriched = df\_enriched.withColumn("prod\_station\_combined", col("prod\_id").cast("string") + "\_" + col("station\_id2").cast("string"))

Traceback (most recent call last):

File "<stdin>", line 2, in <module>

File "/usr/local/spark/python/pyspark/sql/dataframe.py", line 2478, in withColumn

return DataFrame(self.\_jdf.withColumn(colName, col.\_jc), self.sql\_ctx)

File "/usr/local/spark/python/lib/py4j-0.10.9.3-src.zip/py4j/java\_gateway.py", line 1322, in \_\_call\_\_

File "/usr/local/spark/python/pyspark/sql/utils.py", line 117, in deco

raise converted from None

pyspark.sql.utils.AnalysisException: cannot resolve 'station\_id2' given input columns: [day, month, part\_id, prod\_id, prodid, sensor, station\_id, station\_id5, timestamp, value, year];

'Project [timestamp#222, prod\_id#17, station\_id#172, sensor#19, value#20, station\_id5#21, prodid#22, part\_id#23, year#231, month#241, day#252, ((cast(cast(prod\_id#17 as string) as double) + cast(\_ as double)) + cast('station\_id2 as string)) AS prod\_station\_combined#264]

+- Project [timestamp#222, prod\_id#17, station\_id#172, sensor#19, value#20, station\_id5#21, prodid#22, part\_id#23, year#231, month#241, substring(cast(cast(timestamp#222 as date) as string), 9, 2) AS day#252]

+- Project [timestamp#222, prod\_id#17, station\_id#172, sensor#19, value#20, station\_id5#21, prodid#22, part\_id#23, year#231, substring(cast(cast(timestamp#222 as date) as string), 6, 2) AS month#241]

+- Project [timestamp#222, prod\_id#17, station\_id#172, sensor#19, value#20, station\_id5#21, prodid#22, part\_id#23, substring(cast(cast(timestamp#222 as date) as string), 1, 4) AS year#231]

+- Project [to\_timestamp('timestamp, Some(yyyy-MM-dd HH:mm:ss)) AS timestamp#222, prod\_id#17, station\_id#172, sensor#19, value#20, station\_id5#21, prodid#22, part\_id#23]

+- Project [timestamp#16, prod\_id#17, station\_id2#18 AS station\_id#172, sensor#19, value#20, station\_id5#21, prodid#22, part\_id#23]

+- Filter atleastnnonnulls(8, timestamp#16, prod\_id#17, station\_id2#18, sensor#19, value#20, station\_id5#21, prodid#22, part\_id#23)

+- Relation [timestamp#16,prod\_id#17,station\_id2#18,sensor#19,value#20,station\_id5#21,prodid#22,part\_id#23] csv

>>> df\_enriched.show(1)

24/07/24 08:51:55 WARN csv.CSVHeaderChecker: CSV header does not conform to the schema.

Header: timestamp, prod\_id, station\_id, sensor, value, station\_id, prodid, part\_id

Schema: timestamp, prod\_id, station\_id2, sensor, value, station\_id5, prodid, part\_id

Expected: station\_id2 but found: station\_id

CSV file: file:///home/hduser/Desktop/Hackathon/manuf\_data.csv

+---------+-------+----------+------+------------+-----------+-------+--------------------+----+-----+----+

|timestamp|prod\_id|station\_id|sensor| value|station\_id5| prodid| part\_id|year|month| day|

+---------+-------+----------+------+------------+-----------+-------+--------------------+----+-----+----+

| null|ACME123| 234| A|-0.799545863| 234|ACME123|2f10d3bf-e5ed-450...|null| null|null|

+---------+-------+----------+------+------------+-----------+-------+--------------------+----+-----+----+

only showing top 1 row

>>> df\_enriched = df\_final.withColumn("timestamp", to\_timestamp(col("timestamp"), "yyyy-MM-dd HH:mm:ss"))

>>> df\_enriched.show(1)

24/07/24 08:52:27 WARN csv.CSVHeaderChecker: CSV header does not conform to the schema.

Header: timestamp, prod\_id, station\_id, sensor, value, station\_id, prodid, part\_id

Schema: timestamp, prod\_id, station\_id2, sensor, value, station\_id5, prodid, part\_id

Expected: station\_id2 but found: station\_id

CSV file: file:///home/hduser/Desktop/Hackathon/manuf\_data.csv

+---------+-------+----------+------+------------+-----------+-------+--------------------+

|timestamp|prod\_id|station\_id|sensor| value|station\_id5| prodid| part\_id|

+---------+-------+----------+------+------------+-----------+-------+--------------------+

| null|ACME123| 234| A|-0.799545863| 234|ACME123|2f10d3bf-e5ed-450...|

+---------+-------+----------+------+------------+-----------+-------+--------------------+

only showing top 1 row

>>> df\_enriched=df\_final.withColumn("Current\_Date",current\_date())

Traceback (most recent call last):

File "<stdin>", line 1, in <module>

NameError: name 'current\_date' is not defined

>>>

>>> from pyspark.sql.functions import col, lit, to\_timestamp, when, current\_date

>>> df\_enriched=df\_final.withColumn("Current\_Date",current\_date())

>>> df\_enriched.show(1)

24/07/24 08:53:58 WARN csv.CSVHeaderChecker: CSV header does not conform to the schema.

Header: timestamp, prod\_id, station\_id, sensor, value, station\_id, prodid, part\_id

Schema: timestamp, prod\_id, station\_id2, sensor, value, station\_id5, prodid, part\_id

Expected: station\_id2 but found: station\_id

CSV file: file:///home/hduser/Desktop/Hackathon/manuf\_data.csv

+---------+-------+----------+------+------------+-----------+-------+--------------------+------------+

|timestamp|prod\_id|station\_id|sensor| value|station\_id5| prodid| part\_id|Current\_Date|

+---------+-------+----------+------+------------+-----------+-------+--------------------+------------+

| 05:31.4|ACME123| 234| A|-0.799545863| 234|ACME123|2f10d3bf-e5ed-450...| 2024-07-24|

+---------+-------+----------+------+------------+-----------+-------+--------------------+------------+

only showing top 1 row

>>> df\_enriched = df\_enriched.withColumn("Year", year(col("Current\_Date"))) \

... .withColumn("Month", month(col("Current\_Date"))) \

... .withColumn("Day", dayofmonth(col("Current\_Date")))

Traceback (most recent call last):

File "<stdin>", line 1, in <module>

NameError: name 'year' is not defined

>>>

>>> # Show the enriched DataFrame

... df\_enriched.show()

24/07/24 08:55:31 WARN csv.CSVHeaderChecker: CSV header does not conform to the schema.

Header: timestamp, prod\_id, station\_id, sensor, value, station\_id, prodid, part\_id

Schema: timestamp, prod\_id, station\_id2, sensor, value, station\_id5, prodid, part\_id

Expected: station\_id2 but found: station\_id

CSV file: file:///home/hduser/Desktop/Hackathon/manuf\_data.csv

+---------+-------+----------+--------------------+------------+-----------+-------+--------------------+------------+

|timestamp|prod\_id|station\_id| sensor| value|station\_id5| prodid| part\_id|Current\_Date|

+---------+-------+----------+--------------------+------------+-----------+-------+--------------------+------------+

| 05:31.4|ACME123| 234| A|-0.799545863| 234|ACME123|2f10d3bf-e5ed-450...| 2024-07-24|

| 05:29.4|ACME123| 235| tol|-0.388959181| 235|ACME123|2f10d3bf-e5ed-450...| 2024-07-24|

| 05:31.6|ACME123| 235|Rückführkrft ball...| 15.81086033| 235|ACME123|2f10d3bf-e5ed-450...| 2024-07-24|

| 05:33.1|ACME123| 235| r\_nul| 9| 235|ACME123|2f10d3bf-e5ed-450...| 2024-07-24|

| 05:33.5|ACME123| 235| STATUS| OK| 235|ACME123|2f10d3bf-e5ed-450...| 2024-07-24|

| 05:49.0|ACME123| 234| A|-0.013833051| 234|ACME123|e2ec58a7-274a-41a...| 2024-07-24|

| 05:40.1|ACME123| 235|Rückführkrft ball...| 19.22737103| 235|ACME123|e2ec58a7-274a-41a...| 2024-07-24|

| 05:45.2|ACME123| 235| r\_nul| 5| 235|ACME123|e2ec58a7-274a-41a...| 2024-07-24|

| 05:45.6|ACME123| 235| STATUS| OK| 235|ACME123|e2ec58a7-274a-41a...| 2024-07-24|

| 05:49.5|ACME123| 234| A|-0.028555796| 234|ACME123|509cab7f-f18b-431...| 2024-07-24|

| 05:49.6|ACME123| 234| T\_max| 54.4140451| 234|ACME123|509cab7f-f18b-431...| 2024-07-24|

| 05:51.3|ACME123| 234| C| 0| 234|ACME123|509cab7f-f18b-431...| 2024-07-24|

| 05:54.3|ACME123| 234| STATUS| OK| 234|ACME123|509cab7f-f18b-431...| 2024-07-24|

| 05:50.5|ACME123| 235| tol|-1.212287529| 235|ACME123|509cab7f-f18b-431...| 2024-07-24|

| 05:50.9|ACME123| 235|Rückführkrft ball...| 23.76597676| 235|ACME123|509cab7f-f18b-431...| 2024-07-24|

| 05:51.7|ACME123| 235| r\_nul| 3| 235|ACME123|509cab7f-f18b-431...| 2024-07-24|

| 05:53.4|ACME123| 235| STATUS| OK| 235|ACME123|509cab7f-f18b-431...| 2024-07-24|

| 05:59.7|ACME123| 234| A| 0.183833191| 234|ACME123|054485d2-cc42-4c8...| 2024-07-24|

| 06:08.1|ACME123| 234| T\_max| 42.81660303| 234|ACME123|054485d2-cc42-4c8...| 2024-07-24|

| 06:08.2|ACME123| 234| C| 2| 234|ACME123|054485d2-cc42-4c8...| 2024-07-24|

+---------+-------+----------+--------------------+------------+-----------+-------+--------------------+------------+

only showing top 20 rows

>>>

>>> from pyspark.sql.functions import col, lit, to\_timestamp, when, current\_date, year, month, dayofmonth

>>> df\_enriched = df\_enriched.withColumn("Year", year(col("Current\_Date"))) \

... .withColumn("Month", month(col("Current\_Date"))) \

... .withColumn("Day", dayofmonth(col("Current\_Date")))

>>>

>>> # Show the enriched DataFrame

... df\_enriched.show()

24/07/24 08:55:53 WARN csv.CSVHeaderChecker: CSV header does not conform to the schema.

Header: timestamp, prod\_id, station\_id, sensor, value, station\_id, prodid, part\_id

Schema: timestamp, prod\_id, station\_id2, sensor, value, station\_id5, prodid, part\_id

Expected: station\_id2 but found: station\_id

CSV file: file:///home/hduser/Desktop/Hackathon/manuf\_data.csv

+---------+-------+----------+--------------------+------------+-----------+-------+--------------------+------------+----+-----+---+

|timestamp|prod\_id|station\_id| sensor| value|station\_id5| prodid| part\_id|Current\_Date|Year|Month|Day|

+---------+-------+----------+--------------------+------------+-----------+-------+--------------------+------------+----+-----+---+

| 05:31.4|ACME123| 234| A|-0.799545863| 234|ACME123|2f10d3bf-e5ed-450...| 2024-07-24|2024| 7| 24|

| 05:29.4|ACME123| 235| tol|-0.388959181| 235|ACME123|2f10d3bf-e5ed-450...| 2024-07-24|2024| 7| 24|

| 05:31.6|ACME123| 235|Rückführkrft ball...| 15.81086033| 235|ACME123|2f10d3bf-e5ed-450...| 2024-07-24|2024| 7| 24|

| 05:33.1|ACME123| 235| r\_nul| 9| 235|ACME123|2f10d3bf-e5ed-450...| 2024-07-24|2024| 7| 24|

| 05:33.5|ACME123| 235| STATUS| OK| 235|ACME123|2f10d3bf-e5ed-450...| 2024-07-24|2024| 7| 24|

| 05:49.0|ACME123| 234| A|-0.013833051| 234|ACME123|e2ec58a7-274a-41a...| 2024-07-24|2024| 7| 24|

| 05:40.1|ACME123| 235|Rückführkrft ball...| 19.22737103| 235|ACME123|e2ec58a7-274a-41a...| 2024-07-24|2024| 7| 24|

| 05:45.2|ACME123| 235| r\_nul| 5| 235|ACME123|e2ec58a7-274a-41a...| 2024-07-24|2024| 7| 24|

| 05:45.6|ACME123| 235| STATUS| OK| 235|ACME123|e2ec58a7-274a-41a...| 2024-07-24|2024| 7| 24|

| 05:49.5|ACME123| 234| A|-0.028555796| 234|ACME123|509cab7f-f18b-431...| 2024-07-24|2024| 7| 24|

| 05:49.6|ACME123| 234| T\_max| 54.4140451| 234|ACME123|509cab7f-f18b-431...| 2024-07-24|2024| 7| 24|

| 05:51.3|ACME123| 234| C| 0| 234|ACME123|509cab7f-f18b-431...| 2024-07-24|2024| 7| 24|

| 05:54.3|ACME123| 234| STATUS| OK| 234|ACME123|509cab7f-f18b-431...| 2024-07-24|2024| 7| 24|

| 05:50.5|ACME123| 235| tol|-1.212287529| 235|ACME123|509cab7f-f18b-431...| 2024-07-24|2024| 7| 24|

| 05:50.9|ACME123| 235|Rückführkrft ball...| 23.76597676| 235|ACME123|509cab7f-f18b-431...| 2024-07-24|2024| 7| 24|

| 05:51.7|ACME123| 235| r\_nul| 3| 235|ACME123|509cab7f-f18b-431...| 2024-07-24|2024| 7| 24|

| 05:53.4|ACME123| 235| STATUS| OK| 235|ACME123|509cab7f-f18b-431...| 2024-07-24|2024| 7| 24|

| 05:59.7|ACME123| 234| A| 0.183833191| 234|ACME123|054485d2-cc42-4c8...| 2024-07-24|2024| 7| 24|

| 06:08.1|ACME123| 234| T\_max| 42.81660303| 234|ACME123|054485d2-cc42-4c8...| 2024-07-24|2024| 7| 24|

| 06:08.2|ACME123| 234| C| 2| 234|ACME123|054485d2-cc42-4c8...| 2024-07-24|2024| 7| 24|

+---------+-------+----------+--------------------+------------+-----------+-------+--------------------+------------+----+-----+---+

only showing top 20 rows

>>> df\_enriched.write.csv("file:///home/hduser/Desktop/Hackathon/enriched.csv", header=True, mode="overwrite")

24/07/24 09:07:47 WARN csv.CSVHeaderChecker: CSV header does not conform to the schema.

Header: timestamp, prod\_id, station\_id, sensor, value, station\_id, prodid, part\_id

Schema: timestamp, prod\_id, station\_id2, sensor, value, station\_id5, prodid, part\_id

Expected: station\_id2 but found: station\_id

CSV file: file:///home/hduser/Desktop/Hackathon/manuf\_data.csv

>>> mkdir Final\_Folder

File "<stdin>", line 1

mkdir Final\_Folder

^

SyntaxError: invalid syntax

>>> exit

Use exit() or Ctrl-D (i.e. EOF) to exit

>>> exit()