Practica3

January 7, 2022

[2]:
```
from pyspark.sql import SQLContext
sqlContext=SQLContext(sc)
```

[40]:
```
from pyspark.sql import functions as F
```

[6]:
```
bd = sqlContext.read.format(
    "com.databricks.spark.csv"
).option("header", "true").load("hdfs:///tmp/dcd/wordcount/input/flights.csv",␣
 ↪inferSchema=True)
sqlContext.registerDataFrameAsTable(bd, "bd")
```

[14]:
```
bd
```

[14]:
```
DataFrame[month: int, dayofmonth: int, dayofweek: int, carrier: string, flight:
int, origin: string, mile: int, depart: double, duration: int, delay: string]
```

0.0.1 Numero de registros

[8]:
```
bd.count()
```

[8]:
```
275000
```

0.0.2 Estructura

[18]:
```
bd.schema
```

[18]:
```
StructType(List(StructField(month,IntegerType,true),StructField(dayofmonth,Integ
erType,true),StructField(dayofweek,IntegerType,true),StructField(carrier,StringT
ype,true),StructField(flight,IntegerType,true),StructField(origin,StringType,tru
e),StructField(mile,IntegerType,true),StructField(depart,DoubleType,true),Struct
Field(duration,IntegerType,true),StructField(delay,StringType,true)))
```

[38]:
```
bd.printSchema()
```

```
root
 |-- month: integer (nullable = true)
 |-- dayofmonth: integer (nullable = true)
 |-- dayofweek: integer (nullable = true)
 |-- carrier: string (nullable = true)
 |-- flight: integer (nullable = true)
 |-- origin: string (nullable = true)
 |-- mile: integer (nullable = true)
 |-- depart: double (nullable = true)
 |-- duration: integer (nullable = true)
 |-- delay: string (nullable = true)
```

0.0.3 Nombre de las Columnas

[9]:
```
bd.columns
```

[9]:
```
['month',
 'dayofmonth',
 'dayofweek',
 'carrier',
 'flight',
 'origin',
 'mile',
 'depart',
 'duration',
 'delay']
```

0.0.4 Tipos de datos

[10]:
```
bd.dtypes
```

[10]:
```
[('month', 'int'),
 ('dayofmonth', 'int'),
 ('dayofweek', 'int'),
 ('carrier', 'string'),
 ('flight', 'int'),
 ('origin', 'string'),
 ('mile', 'int'),
 ('depart', 'double'),
 ('duration', 'int'),
 ('delay', 'string')]
```

0.0.5 Ver los primeros 20 registros

[20]:
```
bd.limit(20).toPandas()
```

[20]:

	month	dayofmonth	dayofweek	carrier	flight	origin	mile	depart	duration	delay
0	10	10	1	OO	5836	ORD	157	8.18	51	27
1	1	4	1	OO	5866	ORD	466	15.50	102	NA
2	11	22	1	OO	6016	ORD	738	7.17	127	-19
3	2	14	5	B6	199	JFK	2248	21.17	365	60
4	5	25	3	WN	1675	SJC	386	12.92	85	22
5	3	28	1	B6	377	LGA	1076	13.33	182	70
6	5	28	6	B6	904	ORD	740	9.58	130	47
7	1	19	2	UA	820	SFO	679	12.75	123	135
8	8	5	5	US	2175	LGA	214	13.00	71	-10
9	5	27	5	AA	1240	ORD	1197	14.42	195	-11
10	8	20	6	B6	119	JFK	1182	14.67	198	20
11	2	3	1	AA	1881	JFK	1090	15.92	200	-9
12	8	26	5	B6	35	JFK	1028	20.58	193	102
13	4	9	5	AA	336	ORD	733	20.50	125	32
14	3	8	2	UA	678	ORD	733	10.95	129	55
15	8	10	3	OH	6347	LGA	292	11.75	102	8
16	8	14	0	UA	624	ORD	612	17.92	109	57
17	4	8	4	OH	5585	JFK	301	13.25	88	23
18	1	14	4	UA	1524	SFO	414	14.87	91	27
19	1	2	6	AA	1341	ORD	1846	7.50	275	26

0.0.6 Descricipcion Estadistica

[30]:
```
bd.select('origin','carrier').summary('count','min','max').show()
```

```
+-------+------+-------+
|summary|origin|carrier|
+-------+------+-------+
|  count|275000| 275000|
|    min|   JFK|     AA|
|    max|   TUS|     WN|
+-------+------+-------+
```

[33]:
```
bd.select('month',
 'dayofmonth',
 'dayofweek',
 'flight',
 'mile',
 'depart',
 'duration',
 'delay').summary().toPandas()
```

[33]:

	summary	month	dayofmonth	dayofweek
0	count	275000	275000	275000
1	mean	5.24232	15.71406909090909	2.946090909090909
2	stddev	3.4273573316203576	8.805568383848067	1.9635141531217672
3	min	0	1	0
4	25%	2	8	1
5	50%	5	16	3
6	75%	8	23	5
7	max	11	31	6

	flight	mile	depart
0	275000	275000	275000
1	2063.0542763636363	881.2222872727273	14.124930981817384
2	2185.852169684581	700.5178890821038	4.683189503417866
3	1	11	0.12
4	425	344	10.0
5	1081	651	14.08
6	2778	1162	18.08
7	6941	4243	23.98

	duration	delay
0	275000	275000
1	151.64103636363637	28.34773064280709
2	87.0845640768675	54.01489538326629

```
3                  14                -1
4                  85              -6.0
5                 125              15.0
6                 192              43.0
7                 605                NA
```

0.0.7 Descripcion estadistica de una sola columna (delay)

[35]:
```
bd.select('delay').summary().show()
```

```
+-------+-----------------+
|summary|            delay|
+-------+-----------------+
|  count|           275000|
|   mean|28.34773064280709|
| stddev|54.01489538326629|
|    min|               -1|
|    25%|             -6.0|
|    50%|             15.0|
|    75%|             43.0|
|    max|               NA|
+-------+-----------------+
```

0.0.8 Realizar un agrupamiento

[42]:
```
bd.groupBy(['month','dayofmonth']).count().show()
```

```
[Stage 57:==============================>                          (1 + 1) / 2]
+-----+----------+-----+
|month|dayofmonth|count|
+-----+----------+-----+
|    3|        22|  718|
|   10|         2|  569|
|    3|        15|  631|
|    3|        30|  620|
|    0|        25|  912|
|   11|        23|  956|
|    7|        21|  775|
|    6|        20|  888|
|    9|        10|  608|
|    9|        16|  667|
|    5|        16|  998|
|    4|        10|  576|
|    6|         1|  722|
```

```
|    3|         1|  899|
|    7|         4| 1063|
|    2|         2|  704|
|    8|         9|  737|
|    1|        25|  810|
|    6|        22|  825|
|    9|         4|  511|
+-----+----------+-----+
only showing top 20 rows
```

0.0.9 Mostrar la filas ordenas por un campo

[43]:
```
bd.sort('duration').show()
```

```
[Stage 60:============================>                            (1 + 1) / 2]
+-----+----------+---------+-------+------+------+----+------+--------+-----+
|month|dayofmonth|dayofweek|carrier|flight|origin|mile|depart|duration|delay|
+-----+----------+---------+-------+------+------+----+------+--------+-----+
|    5|         3|        2|     B6|   739|   JFK|1617|  4.42|      14|   NA|
|   11|        31|        3|     OO|  5613|   SFO|  30|  18.0|      20|   NA|
|    8|        12|        5|     AA|   707|   LGA|1389|  7.98|      24|   NA|
|    1|        22|        5|     AQ|    72|   OGG|  84| 12.58|      30|   -4|
|    0|         2|        3|     AQ|    72|   OGG|  84| 12.58|      30|   26|
|    1|        26|        2|     AQ|    72|   OGG|  84| 12.58|      30|   -7|
|    1|        10|        0|     AQ|    72|   OGG|  84| 12.58|      30|   -4|
|    1|        23|        6|     AQ|    72|   OGG|  84| 12.58|      30|   -1|
|    1|         6|        3|     AQ|    72|   OGG|  84| 12.58|      30|  -13|
|    0|        29|        2|     AQ|    72|   OGG|  84| 12.58|      30|   92|
|    0|        12|        6|     AQ|    72|   OGG|  84| 12.58|      30|   -2|
|    1|         9|        6|     AQ|    72|   OGG|  84| 12.58|      30|  -11|
|    1|        28|        4|     AQ|    72|   OGG|  84| 12.58|      30|    6|
|    0|        11|        5|     AQ|    72|   OGG|  84| 12.58|      30|   -5|
|    1|        21|        4|     AQ|    72|   OGG|  84| 12.58|      30|   -5|
|    0|        25|        5|     AQ|    72|   OGG|  84| 12.58|      30|  -12|
|    0|        31|        4|     AQ|    72|   OGG|  84| 12.58|      30|    0|
|    0|         7|        1|     AQ|    72|   OGG|  84| 12.58|      30|  -12|
|    3|         2|        3|     HA|   195|   OGG| 100| 19.62|      30|   -3|
|    1|        27|        3|     AQ|    72|   OGG|  84| 12.58|      30|   -4|
+-----+----------+---------+-------+------+------+----+------+--------+-----+
only showing top 20 rows
```

0.0.10 Generar una consulta SQL desde el DataFrame

[46]:
```
bd.registerTempTable('flight_table')
newDF = sqlContext.sql('select * from flight_table where origin=="OGG"')
newDF.show()
```

```
+-----+----------+---------+-------+------+------+----+------+--------+-----+
|month|dayofmonth|dayofweek|carrier|flight|origin|mile|depart|duration|delay|
+-----+----------+---------+-------+------+------+----+------+--------+-----+
|    6|         3|        4|     UA|    70|   OGG|3303| 20.17|     395|   -7|
|    9|        10|        5|     HA|   595|   OGG| 100|  21.0|      34|    4|
|    7|         2|        6|     HA|   155|   OGG| 100|  8.67|      34|   NA|
|    0|        29|        2|     AA|    14|   OGG|2486| 22.25|     300|  -23|
|    1|        28|        4|     HA|   525|   OGG| 100| 10.58|      34|   -4|
|    0|        25|        5|     AQ|    73|   OGG| 100| 11.75|      34|  -10|
|    4|        11|        0|     UA|    44|   OGG|2486|  13.2|     313|    8|
|    0|         1|        2|     AQ|   227|   OGG| 100|  20.0|      34|    2|
|    8|        17|        3|     AA|     6|   OGG|3711|  17.0|     420|    6|
|    3|        19|        6|     UA|    46|   OGG|2486| 20.58|     303|  -12|
|    7|        16|        6|     HA|   180|   OGG|  84| 12.08|      31|    1|
|    5|        25|        3|     UA|    70|   OGG|3303| 20.17|     395|   -7|
|    7|        15|        5|     HA|   133|   OGG| 100|   9.5|      34|   -3|
|    8|        14|        0|     HA|   595|   OGG| 100|  21.0|      34|   -1|
|    0|        19|        6|     HA|   595|   OGG| 100| 21.33|      34|   -3|
|    7|        25|        1|     HA|   595|   OGG| 100| 21.17|      34|   NA|
|    7|        16|        6|     UA|    46|   OGG|2486| 20.83|     301|  -17|
|    8|        12|        5|     UA|    46|   OGG|2486| 20.83|     300|   29|
|    9|         8|        3|     HA|   180|   OGG|  84|  11.9|      31|    3|
|    7|         7|        4|     HA|   177|   OGG| 100| 14.05|      34|    8|
+-----+----------+---------+-------+------+------+----+------+--------+-----+
only showing top 20 rows
```

0.0.11 Generar un agrupamiento que muestre funciones de agregacion, minimo tres (sum, max, min, avg)f

[60]:
```
newdf = bd.groupBy(['carrier']).agg(F.count('carrier').alias('Number_flights'),␣
 ↪F.avg('mile').alias('Average_mileage'),  F.max('mile').alias('Max_mileage'))
newdf.show()
```

```
+-------+--------------+------------------+-----------+
|carrier|Number_flights|   Average_mileage|Max_mileage|
+-------+--------------+------------------+-----------+
|     UA|         72378| 1111.116913979386|       4243|
|     AA|         61809|1177.3037583523435|       4243|
|     B6|         28600|1132.7745454545454|       2704|
|     OO|         45060| 382.7752108300044|       1846|
|     US|         15117| 881.6521135145862|       2845|
```

```
|     AQ|           492| 616.4573170731708|       2541|
|     OH|         17818| 503.4366932315636|       1522|
|     HA|          3936| 623.7385670731708|       2640|
|     WN|         29790|484.92903658945954|       1855|
+-------+--------------+------------------+-----------+
```

0.0.12 Guardar el resultado en una tabla de hive con un directorio hdfs específico

[68]:
```
bd.registerTempTable('carrier_summary')
```

```
22/01/07 05:26:25 WARN org.apache.spark.scheduler.TaskSetManager: Lost task 0.0
in stage 107.0 (TID 144) (cluster-56fd-w-1.us-central1-a.c.focus-
blueprint-334116.internal executor 2): java.io.FileNotFoundException: File does
not exist: /tmp/dcd/wordcount/input/flights.csv
        at
org.apache.hadoop.hdfs.server.namenode.INodeFile.valueOf(INodeFile.java:86)
        at
org.apache.hadoop.hdfs.server.namenode.INodeFile.valueOf(INodeFile.java:76)
        at org.apache.hadoop.hdfs.server.namenode.FSDirStatAndListingOp.getBlock
Locations(FSDirStatAndListingOp.java:156)
        at org.apache.hadoop.hdfs.server.namenode.FSNamesystem.getBlockLocations
(FSNamesystem.java:1990)
        at org.apache.hadoop.hdfs.server.namenode.NameNodeRpcServer.getBlockLoca
tions(NameNodeRpcServer.java:768)
        at org.apache.hadoop.hdfs.protocolPB.ClientNamenodeProtocolServerSideTra
nslatorPB.getBlockLocations(ClientNamenodeProtocolServerSideTranslatorPB.java:44
2)
        at org.apache.hadoop.hdfs.protocol.proto.ClientNamenodeProtocolProtos$Cl
ientNamenodeProtocol$2.callBlockingMethod(ClientNamenodeProtocolProtos.java)
        at org.apache.hadoop.ipc.ProtobufRpcEngine$Server$ProtoBufRpcInvoker.cal
l(ProtobufRpcEngine.java:528)
        at org.apache.hadoop.ipc.RPC$Server.call(RPC.java:1086)
        at org.apache.hadoop.ipc.Server$RpcCall.run(Server.java:1029)
        at org.apache.hadoop.ipc.Server$RpcCall.run(Server.java:957)
        at java.security.AccessController.doPrivileged(Native Method)
        at javax.security.auth.Subject.doAs(Subject.java:422)
        at org.apache.hadoop.security.UserGroupInformation.doAs(UserGroupInforma
tion.java:1762)
        at org.apache.hadoop.ipc.Server$Handler.run(Server.java:2957)

It is possible the underlying files have been updated. You can explicitly
invalidate the cache in Spark by running 'REFRESH TABLE tableName' command in
SQL or by recreating the Dataset/DataFrame involved.
        at org.apache.spark.sql.execution.datasources.FileScanRDD$$anon$1.org$ap
ache$spark$sql$execution$datasources$FileScanRDD$$anon$$readCurrentFile(FileScan
RDD.scala:124)
        at org.apache.spark.sql.execution.datasources.FileScanRDD$$anon$1.nextIt
```

```
erator(FileScanRDD.scala:169)
        at org.apache.spark.sql.execution.datasources.FileScanRDD$$anon$1.hasNex
t(FileScanRDD.scala:93)
        at scala.collection.Iterator$$anon$10.hasNext(Iterator.scala:460)
        at org.apache.spark.sql.catalyst.expressions.GeneratedClass$GeneratedIte
ratorForCodegenStage1.agg_doAggregateWithKeys_0$(Unknown Source)
        at org.apache.spark.sql.catalyst.expressions.GeneratedClass$GeneratedIte
ratorForCodegenStage1.processNext(Unknown Source)
        at org.apache.spark.sql.execution.BufferedRowIterator.hasNext(BufferedRo
wIterator.java:43)
        at org.apache.spark.sql.execution.WholeStageCodegenExec$$anon$1.hasNext(
WholeStageCodegenExec.scala:755)
        at scala.collection.Iterator$$anon$10.hasNext(Iterator.scala:460)
        at org.apache.spark.shuffle.sort.BypassMergeSortShuffleWriter.write(Bypa
ssMergeSortShuffleWriter.java:132)
        at org.apache.spark.shuffle.ShuffleWriteProcessor.write(ShuffleWriteProc
essor.scala:59)
        at
org.apache.spark.scheduler.ShuffleMapTask.runTask(ShuffleMapTask.scala:99)
        at
org.apache.spark.scheduler.ShuffleMapTask.runTask(ShuffleMapTask.scala:52)
        at org.apache.spark.scheduler.Task.run(Task.scala:131)
        at
org.apache.spark.executor.Executor$TaskRunner.$anonfun$run$3(Executor.scala:497)
        at org.apache.spark.util.Utils$.tryWithSafeFinally(Utils.scala:1439)
        at org.apache.spark.executor.Executor$TaskRunner.run(Executor.scala:500)
        at
java.util.concurrent.ThreadPoolExecutor.runWorker(ThreadPoolExecutor.java:1149)
        at
java.util.concurrent.ThreadPoolExecutor$Worker.run(ThreadPoolExecutor.java:624)
        at java.lang.Thread.run(Thread.java:748)

22/01/07 05:26:25 ERROR org.apache.spark.scheduler.TaskSetManager: Task 0 in
stage 107.0 failed 4 times; aborting job
22/01/07 05:26:25 ERROR
org.apache.spark.sql.execution.datasources.FileFormatWriter: Aborting job
2f7487cd-8dd7-48f0-95e6-f0a4cc3ff4a0.
org.apache.spark.SparkException: Job aborted due to stage failure: Task 0 in
stage 107.0 failed 4 times, most recent failure: Lost task 0.3 in stage 107.0
(TID 150) (cluster-56fd-w-1.us-central1-a.c.focus-blueprint-334116.internal
executor 2): java.io.FileNotFoundException: File does not exist:
/tmp/dcd/wordcount/input/flights.csv
        at
org.apache.hadoop.hdfs.server.namenode.INodeFile.valueOf(INodeFile.java:86)
        at
org.apache.hadoop.hdfs.server.namenode.INodeFile.valueOf(INodeFile.java:76)
        at org.apache.hadoop.hdfs.server.namenode.FSDirStatAndListingOp.getBlock
Locations(FSDirStatAndListingOp.java:156)
```

```
        at org.apache.hadoop.hdfs.server.namenode.FSNamesystem.getBlockLocations
(FSNamesystem.java:1990)
        at org.apache.hadoop.hdfs.server.namenode.NameNodeRpcServer.getBlockLoca
tions(NameNodeRpcServer.java:768)
        at org.apache.hadoop.hdfs.protocolPB.ClientNamenodeProtocolServerSideTra
nslatorPB.getBlockLocations(ClientNamenodeProtocolServerSideTranslatorPB.java:44
2)
        at org.apache.hadoop.hdfs.protocol.proto.ClientNamenodeProtocolProtos$Cl
ientNamenodeProtocol$2.callBlockingMethod(ClientNamenodeProtocolProtos.java)
        at org.apache.hadoop.ipc.ProtobufRpcEngine$Server$ProtoBufRpcInvoker.cal
l(ProtobufRpcEngine.java:528)
        at org.apache.hadoop.ipc.RPC$Server.call(RPC.java:1086)
        at org.apache.hadoop.ipc.Server$RpcCall.run(Server.java:1029)
        at org.apache.hadoop.ipc.Server$RpcCall.run(Server.java:957)
        at java.security.AccessController.doPrivileged(Native Method)
        at javax.security.auth.Subject.doAs(Subject.java:422)
        at org.apache.hadoop.security.UserGroupInformation.doAs(UserGroupInforma
tion.java:1762)
        at org.apache.hadoop.ipc.Server$Handler.run(Server.java:2957)

It is possible the underlying files have been updated. You can explicitly
invalidate the cache in Spark by running 'REFRESH TABLE tableName' command in
SQL or by recreating the Dataset/DataFrame involved.
        at org.apache.spark.sql.execution.datasources.FileScanRDD$$anon$1.org$ap
ache$spark$sql$execution$datasources$FileScanRDD$$anon$$readCurrentFile(FileScan
RDD.scala:124)
        at org.apache.spark.sql.execution.datasources.FileScanRDD$$anon$1.nextIt
erator(FileScanRDD.scala:169)
        at org.apache.spark.sql.execution.datasources.FileScanRDD$$anon$1.hasNex
t(FileScanRDD.scala:93)
        at scala.collection.Iterator$$anon$10.hasNext(Iterator.scala:460)
        at org.apache.spark.sql.catalyst.expressions.GeneratedClass$GeneratedIte
ratorForCodegenStage1.agg_doAggregateWithKeys_0$(Unknown Source)
        at org.apache.spark.sql.catalyst.expressions.GeneratedClass$GeneratedIte
ratorForCodegenStage1.processNext(Unknown Source)
        at org.apache.spark.sql.execution.BufferedRowIterator.hasNext(BufferedRo
wIterator.java:43)
        at org.apache.spark.sql.execution.WholeStageCodegenExec$$anon$1.hasNext(
WholeStageCodegenExec.scala:755)
        at scala.collection.Iterator$$anon$10.hasNext(Iterator.scala:460)
        at org.apache.spark.shuffle.sort.BypassMergeSortShuffleWriter.write(Bypa
ssMergeSortShuffleWriter.java:132)
        at org.apache.spark.shuffle.ShuffleWriteProcessor.write(ShuffleWriteProc
essor.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$3(Executor.scala:497)
        at org.apache.spark.util.Utils$.tryWithSafeFinally(Utils.scala:1439)
        at org.apache.spark.executor.Executor$TaskRunner.run(Executor.scala:500)
        at
java.util.concurrent.ThreadPoolExecutor.runWorker(ThreadPoolExecutor.java:1149)
        at
java.util.concurrent.ThreadPoolExecutor$Worker.run(ThreadPoolExecutor.java:624)
        at java.lang.Thread.run(Thread.java:748)

Driver stacktrace:
        at org.apache.spark.scheduler.DAGScheduler.failJobAndIndependentStages(D
AGScheduler.scala:2259)
        at org.apache.spark.scheduler.DAGScheduler.$anonfun$abortStage$2(DAGSche
duler.scala:2208)
        at org.apache.spark.scheduler.DAGScheduler.$anonfun$abortStage$2$adapted
(DAGScheduler.scala:2207)
        at
scala.collection.mutable.ResizableArray.foreach(ResizableArray.scala:62)
        at
scala.collection.mutable.ResizableArray.foreach$(ResizableArray.scala:55)
        at scala.collection.mutable.ArrayBuffer.foreach(ArrayBuffer.scala:49)
        at
org.apache.spark.scheduler.DAGScheduler.abortStage(DAGScheduler.scala:2207)
        at org.apache.spark.scheduler.DAGScheduler.$anonfun$handleTaskSetFailed$
1(DAGScheduler.scala:1079)
        at org.apache.spark.scheduler.DAGScheduler.$anonfun$handleTaskSetFailed$
1$adapted(DAGScheduler.scala:1079)
        at scala.Option.foreach(Option.scala:407)
        at org.apache.spark.scheduler.DAGScheduler.handleTaskSetFailed(DAGSchedu
ler.scala:1079)
        at org.apache.spark.scheduler.DAGSchedulerEventProcessLoop.doOnReceive(D
AGScheduler.scala:2446)
        at org.apache.spark.scheduler.DAGSchedulerEventProcessLoop.onReceive(DAG
Scheduler.scala:2388)
        at org.apache.spark.scheduler.DAGSchedulerEventProcessLoop.onReceive(DAG
Scheduler.scala:2377)
        at org.apache.spark.util.EventLoop$$anon$1.run(EventLoop.scala:49)
Caused by: java.io.FileNotFoundException: File does not exist:
/tmp/dcd/wordcount/input/flights.csv
        at
org.apache.hadoop.hdfs.server.namenode.INodeFile.valueOf(INodeFile.java:86)
        at
org.apache.hadoop.hdfs.server.namenode.INodeFile.valueOf(INodeFile.java:76)
        at org.apache.hadoop.hdfs.server.namenode.FSDirStatAndListingOp.getBlock
Locations(FSDirStatAndListingOp.java:156)
        at org.apache.hadoop.hdfs.server.namenode.FSNamesystem.getBlockLocations
```

```
(FSNamesystem.java:1990)
        at org.apache.hadoop.hdfs.server.namenode.NameNodeRpcServer.getBlockLoca
tions(NameNodeRpcServer.java:768)
        at org.apache.hadoop.hdfs.protocolPB.ClientNamenodeProtocolServerSideTra
nslatorPB.getBlockLocations(ClientNamenodeProtocolServerSideTranslatorPB.java:44
2)
        at org.apache.hadoop.hdfs.protocol.proto.ClientNamenodeProtocolProtos$Cl
ientNamenodeProtocol$2.callBlockingMethod(ClientNamenodeProtocolProtos.java)
        at org.apache.hadoop.ipc.ProtobufRpcEngine$Server$ProtoBufRpcInvoker.cal
l(ProtobufRpcEngine.java:528)
        at org.apache.hadoop.ipc.RPC$Server.call(RPC.java:1086)
        at org.apache.hadoop.ipc.Server$RpcCall.run(Server.java:1029)
        at org.apache.hadoop.ipc.Server$RpcCall.run(Server.java:957)
        at java.security.AccessController.doPrivileged(Native Method)
        at javax.security.auth.Subject.doAs(Subject.java:422)
        at org.apache.hadoop.security.UserGroupInformation.doAs(UserGroupInforma
tion.java:1762)
        at org.apache.hadoop.ipc.Server$Handler.run(Server.java:2957)

It is possible the underlying files have been updated. You can explicitly
invalidate the cache in Spark by running 'REFRESH TABLE tableName' command in
SQL or by recreating the Dataset/DataFrame involved.
        at org.apache.spark.sql.execution.datasources.FileScanRDD$$anon$1.org$ap
ache$spark$sql$execution$datasources$FileScanRDD$$anon$$readCurrentFile(FileScan
RDD.scala:124)
        at org.apache.spark.sql.execution.datasources.FileScanRDD$$anon$1.nextIt
erator(FileScanRDD.scala:169)
        at org.apache.spark.sql.execution.datasources.FileScanRDD$$anon$1.hasNex
t(FileScanRDD.scala:93)
        at scala.collection.Iterator$$anon$10.hasNext(Iterator.scala:460)
        at org.apache.spark.sql.catalyst.expressions.GeneratedClass$GeneratedIte
ratorForCodegenStage1.agg_doAggregateWithKeys_0$(Unknown Source)
        at org.apache.spark.sql.catalyst.expressions.GeneratedClass$GeneratedIte
ratorForCodegenStage1.processNext(Unknown Source)
        at org.apache.spark.sql.execution.BufferedRowIterator.hasNext(BufferedRo
wIterator.java:43)
        at org.apache.spark.sql.execution.WholeStageCodegenExec$$anon$1.hasNext(
WholeStageCodegenExec.scala:755)
        at scala.collection.Iterator$$anon$10.hasNext(Iterator.scala:460)
        at org.apache.spark.shuffle.sort.BypassMergeSortShuffleWriter.write(Bypa
ssMergeSortShuffleWriter.java:132)
        at org.apache.spark.shuffle.ShuffleWriteProcessor.write(ShuffleWriteProc
essor.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$3(Executor.scala:497)
        at org.apache.spark.util.Utils$.tryWithSafeFinally(Utils.scala:1439)
        at org.apache.spark.executor.Executor$TaskRunner.run(Executor.scala:500)
        at
java.util.concurrent.ThreadPoolExecutor.runWorker(ThreadPoolExecutor.java:1149)
        at
java.util.concurrent.ThreadPoolExecutor$Worker.run(ThreadPoolExecutor.java:624)
        at java.lang.Thread.run(Thread.java:748)
22/01/07 05:26:25 WARN org.apache.spark.scheduler.TaskSetManager: Lost task 1.3
in stage 107.0 (TID 151) (cluster-56fd-w-1.us-central1-a.c.focus-
blueprint-334116.internal executor 2): TaskKilled (Stage cancelled)
```

```
---------------------------------------------------------------------------
Py4JJavaError                             Traceback (most recent call last)
/tmp/ipykernel_28514/4161844309.py in <module>
      1 bd.registerTempTable('carrier_summary')
----> 2 newdf.write.option('path','hdfs:///tmp/dcd/wordcount/').
 ↪saveAsTable("carrier.Summary")

/usr/lib/spark/python/pyspark/sql/readwriter.py in saveAsTable(self, name,␣
 ↪format, mode, partitionBy, **options)
   1156         if format is not None:
   1157             self.format(format)
-> 1158         self._jwrite.saveAsTable(name)
   1159
   1160     def json(self, path, mode=None, compression=None, dateFormat=None,␣
 ↪timestampFormat=None,

/usr/lib/spark/python/lib/py4j-0.10.9-src.zip/py4j/java_gateway.py in␣
 ↪__call__(self, *args)
   1302
   1303         answer = self.gateway_client.send_command(command)
-> 1304         return_value = get_return_value(
   1305             answer, self.gateway_client, self.target_id, self.name)
   1306

/usr/lib/spark/python/pyspark/sql/utils.py in deco(*a, **kw)
    109     def deco(*a, **kw):
    110         try:
--> 111             return f(*a, **kw)
    112         except py4j.protocol.Py4JJavaError as e:
    113             converted = convert_exception(e.java_exception)

/usr/lib/spark/python/lib/py4j-0.10.9-src.zip/py4j/protocol.py in␣
 ↪get_return_value(answer, gateway_client, target_id, name)
    324             value = OUTPUT_CONVERTER[type](answer[2:], gateway_client)
```

```
    325             if answer[1] == REFERENCE_TYPE:
--> 326                 raise Py4JJavaError(
    327                     "An error occurred while calling {0}{1}{2}.\n".
    328                     format(target_id, ".", name), value)

Py4JJavaError: An error occurred while calling o486.saveAsTable.
: org.apache.spark.SparkException: Job aborted.
        at org.apache.spark.sql.execution.datasources.FileFormatWriter$.
↪write(FileFormatWriter.scala:231)
        at org.apache.spark.sql.execution.datasources.
↪InsertIntoHadoopFsRelationCommand.run(InsertIntoHadoopFsRelationCommand.scala
↪188)
        at org.apache.spark.sql.execution.datasources.DataSource.
↪writeAndRead(DataSource.scala:550)
        at org.apache.spark.sql.execution.command.
↪CreateDataSourceTableAsSelectCommand.saveDataIntoTable(createDataSourceTables
↪scala:220)
        at org.apache.spark.sql.execution.command.
↪CreateDataSourceTableAsSelectCommand.run(createDataSourceTables.scala:177)
        at org.apache.spark.sql.execution.command.DataWritingCommandExec.
↪sideEffectResult$lzycompute(commands.scala:108)
        at org.apache.spark.sql.execution.command.DataWritingCommandExec.
↪sideEffectResult(commands.scala:106)
        at org.apache.spark.sql.execution.command.DataWritingCommandExec.
↪doExecute(commands.scala:131)
        at org.apache.spark.sql.execution.SparkPlan.$anonfun$execute$1(SparkPla
↪scala:180)
        at org.apache.spark.sql.execution.SparkPlan.
↪$anonfun$executeQuery$1(SparkPlan.scala:218)
        at org.apache.spark.rdd.RDDOperationScope$.withScope(RDDOperationScope.
↪scala:151)
        at org.apache.spark.sql.execution.SparkPlan.executeQuery(SparkPlan.scal
↪215)
        at org.apache.spark.sql.execution.SparkPlan.execute(SparkPlan.scala:176
        at org.apache.spark.sql.execution.QueryExecution.
↪toRdd$lzycompute(QueryExecution.scala:132)
        at org.apache.spark.sql.execution.QueryExecution.toRdd(QueryExecution.
↪scala:131)
        at org.apache.spark.sql.DataFrameWriter.
↪$anonfun$runCommand$1(DataFrameWriter.scala:989)
        at org.apache.spark.sql.execution.SQLExecution$.
↪$anonfun$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.DataFrameWriter.runCommand(DataFrameWriter.scala:
↪989)
        at org.apache.spark.sql.DataFrameWriter.createTable(DataFrameWriter.
↪scala:753)
        at org.apache.spark.sql.DataFrameWriter.saveAsTable(DataFrameWriter.
↪scala:731)
        at org.apache.spark.sql.DataFrameWriter.saveAsTable(DataFrameWriter.
↪scala:626)
        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.GatewayConnection.run(GatewayConnection.java:238)
        at java.lang.Thread.run(Thread.java:748)
Caused by: org.apache.spark.SparkException: Job aborted due to stage failure:␣
↪Task 0 in stage 107.0 failed 4 times, most recent failure: Lost task 0.3 in␣
↪stage 107.0 (TID 150) (cluster-56fd-w-1.us-central1-a.c.focus-blueprint-33411[illegible].
↪internal executor 2): java.io.FileNotFoundException: File does not exist: /tm[illegible]/
↪dcd/wordcount/input/flights.csv
        at org.apache.hadoop.hdfs.server.namenode.INodeFile.valueOf(INodeFile.
↪java:86)
        at org.apache.hadoop.hdfs.server.namenode.INodeFile.valueOf(INodeFile.
↪java:76)
        at org.apache.hadoop.hdfs.server.namenode.FSDirStatAndListingOp.
↪getBlockLocations(FSDirStatAndListingOp.java:156)
        at org.apache.hadoop.hdfs.server.namenode.FSNamesystem.
↪getBlockLocations(FSNamesystem.java:1990)
        at org.apache.hadoop.hdfs.server.namenode.NameNodeRpcServer.
↪getBlockLocations(NameNodeRpcServer.java:768)
        at org.apache.hadoop.hdfs.protocolPB.
↪ClientNamenodeProtocolServerSideTranslatorPB.
↪getBlockLocations(ClientNamenodeProtocolServerSideTranslatorPB.java:442)
        at org.apache.hadoop.hdfs.protocol.proto.
↪ClientNamenodeProtocolProtos$ClientNamenodeProtocol$2.
↪callBlockingMethod(ClientNamenodeProtocolProtos.java)
        at org.apache.hadoop.ipc.ProtobufRpcEngine$Server$ProtoBufRpcInvoker.
↪call(ProtobufRpcEngine.java:528)
        at org.apache.hadoop.ipc.RPC$Server.call(RPC.java:1086)
        at org.apache.hadoop.ipc.Server$RpcCall.run(Server.java:1029)
        at org.apache.hadoop.ipc.Server$RpcCall.run(Server.java:957)
```

```
        at java.security.AccessController.doPrivileged(Native Method)
        at javax.security.auth.Subject.doAs(Subject.java:422)
        at org.apache.hadoop.security.UserGroupInformation.
↪doAs(UserGroupInformation.java:1762)
        at org.apache.hadoop.ipc.Server$Handler.run(Server.java:2957)

It is possible the underlying files have been updated. You can explicitly␣
↪invalidate the cache in Spark by running 'REFRESH TABLE tableName' command in␣
↪SQL or by recreating the Dataset/DataFrame involved.
        at org.apache.spark.sql.execution.datasources.FileScanRDD$$anon$1.
↪org$apache$spark$sql$execution$datasources$FileScanRDD$$anon$$readCurrentFile(FileScanRDD.
↪scala:124)
        at org.apache.spark.sql.execution.datasources.FileScanRDD$$anon$1.
↪nextIterator(FileScanRDD.scala:169)
        at org.apache.spark.sql.execution.datasources.FileScanRDD$$anon$1.
↪hasNext(FileScanRDD.scala:93)
        at scala.collection.Iterator$$anon$10.hasNext(Iterator.scala:460)
        at org.apache.spark.sql.catalyst.expressions.
↪GeneratedClass$GeneratedIteratorForCodegenStage1.
↪agg_doAggregateWithKeys_0$(Unknown Source)
        at org.apache.spark.sql.catalyst.expressions.
↪GeneratedClass$GeneratedIteratorForCodegenStage1.processNext(Unknown Source)
        at org.apache.spark.sql.execution.BufferedRowIterator.
↪hasNext(BufferedRowIterator.java:43)
        at org.apache.spark.sql.execution.WholeStageCodegenExec$$anon$1.
↪hasNext(WholeStageCodegenExec.scala:755)
        at scala.collection.Iterator$$anon$10.hasNext(Iterator.scala:460)
        at org.apache.spark.shuffle.sort.BypassMergeSortShuffleWriter.
↪write(BypassMergeSortShuffleWriter.java:132)
        at org.apache.spark.shuffle.ShuffleWriteProcessor.
↪write(ShuffleWriteProcessor.scala:59)
        at org.apache.spark.scheduler.ShuffleMapTask.runTask(ShuffleMapTask.
↪scala:99)
        at org.apache.spark.scheduler.ShuffleMapTask.runTask(ShuffleMapTask.
↪scala:52)
        at org.apache.spark.scheduler.Task.run(Task.scala:131)
        at org.apache.spark.executor.Executor$TaskRunner.$anonfun$run$3(Executor.
↪scala:497)
        at org.apache.spark.util.Utils$.tryWithSafeFinally(Utils.scala:1439)
        at org.apache.spark.executor.Executor$TaskRunner.run(Executor.scala:500
        at java.util.concurrent.ThreadPoolExecutor.runWorker(ThreadPoolExecutor
↪java:1149)
        at java.util.concurrent.ThreadPoolExecutor$Worker.run(ThreadPoolExecutor.
↪java:624)
        at java.lang.Thread.run(Thread.java:748)

Driver stacktrace:
```

```
        at org.apache.spark.scheduler.DAGScheduler.
↪failJobAndIndependentStages(DAGScheduler.scala:2259)
        at org.apache.spark.scheduler.DAGScheduler.
↪$anonfun$abortStage$2(DAGScheduler.scala:2208)
        at org.apache.spark.scheduler.DAGScheduler.
↪$anonfun$abortStage$2$adapted(DAGScheduler.scala:2207)
        at scala.collection.mutable.ResizableArray.foreach(ResizableArray.scala
↪62)
        at scala.collection.mutable.ResizableArray.foreach$(ResizableArray.scal :
↪55)
        at scala.collection.mutable.ArrayBuffer.foreach(ArrayBuffer.scala:49)
        at org.apache.spark.scheduler.DAGScheduler.abortStage(DAGScheduler.scal :
↪2207)
        at org.apache.spark.scheduler.DAGScheduler.
↪$anonfun$handleTaskSetFailed$1(DAGScheduler.scala:1079)
        at org.apache.spark.scheduler.DAGScheduler.
↪$anonfun$handleTaskSetFailed$1$adapted(DAGScheduler.scala:1079)
        at scala.Option.foreach(Option.scala:407)
        at org.apache.spark.scheduler.DAGScheduler.
↪handleTaskSetFailed(DAGScheduler.scala:1079)
        at org.apache.spark.scheduler.DAGSchedulerEventProcessLoop.
↪doOnReceive(DAGScheduler.scala:2446)
        at org.apache.spark.scheduler.DAGSchedulerEventProcessLoop.
↪onReceive(DAGScheduler.scala:2388)
        at org.apache.spark.scheduler.DAGSchedulerEventProcessLoop.
↪onReceive(DAGScheduler.scala:2377)
        at org.apache.spark.util.EventLoop$$anon$1.run(EventLoop.scala:49)
Caused by: java.io.FileNotFoundException: File does not exist: /tmp/dcd/
↪wordcount/input/flights.csv
        at org.apache.hadoop.hdfs.server.namenode.INodeFile.valueOf(INodeFile.
↪java:86)
        at org.apache.hadoop.hdfs.server.namenode.INodeFile.valueOf(INodeFile.
↪java:76)
        at org.apache.hadoop.hdfs.server.namenode.FSDirStatAndListingOp.
↪getBlockLocations(FSDirStatAndListingOp.java:156)
        at org.apache.hadoop.hdfs.server.namenode.FSNamesystem.
↪getBlockLocations(FSNamesystem.java:1990)
        at org.apache.hadoop.hdfs.server.namenode.NameNodeRpcServer.
↪getBlockLocations(NameNodeRpcServer.java:768)
        at org.apache.hadoop.hdfs.protocolPB.
↪ClientNamenodeProtocolServerSideTranslatorPB.
↪getBlockLocations(ClientNamenodeProtocolServerSideTranslatorPB.java:442)
        at org.apache.hadoop.hdfs.protocol.proto.
↪ClientNamenodeProtocolProtos$ClientNamenodeProtocol$2.
↪callBlockingMethod(ClientNamenodeProtocolProtos.java)
        at org.apache.hadoop.ipc.ProtobufRpcEngine$Server$ProtoBufRpcInvoker.
↪call(ProtobufRpcEngine.java:528)
        at org.apache.hadoop.ipc.RPC$Server.call(RPC.java:1086)
```

```
        at org.apache.hadoop.ipc.Server$RpcCall.run(Server.java:1029)
        at org.apache.hadoop.ipc.Server$RpcCall.run(Server.java:957)
        at java.security.AccessController.doPrivileged(Native Method)
        at javax.security.auth.Subject.doAs(Subject.java:422)
        at org.apache.hadoop.security.UserGroupInformation.
↪doAs(UserGroupInformation.java:1762)
        at org.apache.hadoop.ipc.Server$Handler.run(Server.java:2957)

It is possible the underlying files have been updated. You can explicitly␣
↪invalidate the cache in Spark by running 'REFRESH TABLE tableName' command in␣
↪SQL or by recreating the Dataset/DataFrame involved.
        at org.apache.spark.sql.execution.datasources.FileScanRDD$$anon$1.
↪org$apache$spark$sql$execution$datasources$FileScanRDD$$anon$$readCurrentFile(FileScanRDD.
↪scala:124)
        at org.apache.spark.sql.execution.datasources.FileScanRDD$$anon$1.
↪nextIterator(FileScanRDD.scala:169)
        at org.apache.spark.sql.execution.datasources.FileScanRDD$$anon$1.
↪hasNext(FileScanRDD.scala:93)
        at scala.collection.Iterator$$anon$10.hasNext(Iterator.scala:460)
        at org.apache.spark.sql.catalyst.expressions.
↪GeneratedClass$GeneratedIteratorForCodegenStage1.
↪agg_doAggregateWithKeys_0$(Unknown Source)
        at org.apache.spark.sql.catalyst.expressions.
↪GeneratedClass$GeneratedIteratorForCodegenStage1.processNext(Unknown Source)
        at org.apache.spark.sql.execution.BufferedRowIterator.
↪hasNext(BufferedRowIterator.java:43)
        at org.apache.spark.sql.execution.WholeStageCodegenExec$$anon$1.
↪hasNext(WholeStageCodegenExec.scala:755)
        at scala.collection.Iterator$$anon$10.hasNext(Iterator.scala:460)
        at org.apache.spark.shuffle.sort.BypassMergeSortShuffleWriter.
↪write(BypassMergeSortShuffleWriter.java:132)
        at org.apache.spark.shuffle.ShuffleWriteProcessor.
↪write(ShuffleWriteProcessor.scala:59)
        at org.apache.spark.scheduler.ShuffleMapTask.runTask(ShuffleMapTask.
↪scala:99)
        at org.apache.spark.scheduler.ShuffleMapTask.runTask(ShuffleMapTask.
↪scala:52)
        at org.apache.spark.scheduler.Task.run(Task.scala:131)
        at org.apache.spark.executor.Executor$TaskRunner.$anonfun$run$3(Executo
↪scala:497)
        at org.apache.spark.util.Utils$.tryWithSafeFinally(Utils.scala:1439)
        at org.apache.spark.executor.Executor$TaskRunner.run(Executor.scala:500
        at java.util.concurrent.ThreadPoolExecutor.runWorker(ThreadPoolExecutor
↪java:1149)
        at java.util.concurrent.ThreadPoolExecutor$Worker.run(ThreadPoolExecuto
↪java:624)
        at java.lang.Thread.run(Thread.java:748)
```

[58]:
```
type(newdf)
```

[58]:
```
pyspark.sql.dataframe.DataFrame
```

0.0.13 Listar las tablas de la base de datos

[69]:
```
spark.sql("show tables").show()
```

```
+--------+---------------+-----------+
|database|      tableName|isTemporary|
+--------+---------------+-----------+
|        |             bd|       true|
|        |carrier_summary|       true|
|        |   flight_table|       true|
+--------+---------------+-----------+
```

0.0.14 Mostrar el esquema de la nueva tabla

[70]:
```
newdf.printSchema()
```

```
root
 |-- carrier: string (nullable = true)
 |-- Number_flights: long (nullable = false)
 |-- Average_mileage: double (nullable = true)
 |-- Max_mileage: integer (nullable = true)
```

[ ]:
```

```