HU Extension Assignment 04 E63 Big Data Analytics

Issued on: Feb. 18, 2017 Due on Saturday by 9:30AM EST, Feb. 25, 2017

You can do these problems in any of 4 languages: Python, Scala, Java or R.

**Problem 1.** Go to an online newspaper and select two articles in English on two new movies featured in theaters this spring 2017. Save those articles as .txt files and then import them into two Spark RDD objects, movieA and movieB. Use Spark transformation and action functions to transform those initial RDD-s into RDD-s that contain words and numbers of occurrence of those words in respective article. Eliminate

from both lists so called “stop words”. Take the list of stopwords from this Web page:

http://www.lextek.com/manuals/onix/stopwords1.html. List for us 10 most frequent words in each RDD. Subsequently create RDD-s that contain only words unique for each of text. Finally create an RDD that contains only the words common to both texts. Inelatest RDD preserve numbers of occurrences in two articles. In other words a row in your RDD will look like (actor 45 32). List for us a random samples containing 5% of words in the final RDD. We are just practicing RDD transformations and actions. You could implement this problem in a command shell or as a standalone program.

1. [GetOut Movie Review](https://www.nytimes.com/2017/02/23/movies/get-out-review-jordan-peele.html?rref=collection%2Fcollection%2Fmovie-guide&action=click&contentCollection=undefined&region=stream&module=stream_unit&version=latest-stories&contentPlacement=2&pgtype=collection&_r=0)
2. [MyLifeAsZucchini Movie Review](https://www.nytimes.com/2017/02/23/movies/my-life-as-a-zucchini-review.html?rref=collection%2Fcollection%2Fmovie-guide&action=click&contentCollection=undefined&region=stream&module=stream_unit&version=latest-stories&contentPlacement=9&pgtype=collection)

|  |
| --- |
| **See attached p1.Movies.py program.**  **------------------------------------------------------------------------------------------------------------------**  **[cloudera@quickstart HW4]$** *spark-submit p1.Movies.py*  ----------------------------------------------------------------------------------------------------------------------  StopWords count: 423  First 10 stop words: [(u'all', 1), (u'pointing', 1), (u'four', 1), (u'go', 1), (u'oldest', 1), (u'seemed', 1), (u'whose', 1), (u'certainly', 1), (u'young', 1), (u'presents', 1)]  ----------------------------------------------------------------------------------------------------------------------  Movie (GetOut) #unique words: 600  First 10 words in the movie: [(u'', 122), (u'voluble', 1), (u'all', 1), (u'Obama.', 1), (u'hordes', 1), (u'suburban', 1), (u'Out,\u201d', 3), (u'Out,\u2019', 1), (u'jarring', 1), (u'skin', 1)]  ----------------------------------------------------------------------------------------------------------------------  Movie (GetOut) #unique words (without stop words): 471  Top 10 most frequently used words: [(u'', 122), (u'\*', 11), (u'Peele', 10), (u'Mr.', 8), (u'Jordan', 6), (u'story', 6), (u'white', 6), (u'\u201cGet', 6), (u'The', 5), (u'Chris', 5)]  ----------------------------------------------------------------------------------------------------------------------  Movie (MyLifeAsZucchini) #unique words: 403  First 10 words in the movie: [(u'', 110), (u'all', 2), (u'<https://www.youtube.com/watch?v=bd2TOf1kmfk>', 1), (u'skip', 1), (u'00:00', 2), (u'Paulin', 1), (u'whose', 2), (u'technique', 1), (u'Watch', 1), (u'to', 6)]  ----------------------------------------------------------------------------------------------------------------------  Movie (MyLifeAsZucchini) #unique words (without stop words): 352  Top 10 most frequently used words: [(u'', 110), (u'\*', 11), (u'Life', 9), (u'The', 6), (u'Claude', 5), (u'Zucchini', 5), (u'Video', 4), (u'My', 3), (u'\u2018My', 3), (u'Barras', 3)]  ----------------------------------------------------------------------------------------------------------------------  First 10 words unique in the movie (GetOut): [(u'voluble', 1), (u'hordes', 1), (u'suburban', 1), (u'Out,\u201d', 3), (u'Out,\u2019', 1), (u'jarring', 1), (u'skin', 1), (u'rocking', 1), (u'Obama.', 1), (u'flash', 1)]  ----------------------------------------------------------------------------------------------------------------------  First 10 words unique in the movie (MyLifeAsZucchini): [(u'skip', 1), (u'00:00', 2), (u'Paulin', 1), (u'long-shot', 1), (u'<https://www.youtube.com/watch?v=bd2TOf1kmfk>', 1), (u'sent', 1), (u'whimsy.', 1), (u'Still,', 1), (u'<https://www.nytimes.com/by/a-o-scott>FEB.', 1), (u'disturbing,', 1)]  ----------------------------------------------------------------------------------------------------------------------  Sample of 5 perc common words to both movies: [(u'scene', (3, 1)), (u'It\u2019s', (1, 2)), (u'powered', (1, 1))]  ---------------------------------------------------------------------------------------------------------------------- |

**Problem 2**. Consider attached file emps.txt. It contains: name, age and salary of three employees. Create RDD emps by importing that file into Spark. Next create a new RDD

emps\_fields by transforming the content of every line in RDD emps into a tuple with

three individual elements by splitting the lines on commas. Spark has a class Row and you need to import it in your script or program. Row comes from the same package as class SQLContext. Row class creates rows with named and typed fields. You need to apply“constructor” Row to every tuple in RDD emps\_fields, like:

employees = emps\_fields.map(lambda e: Row(name = e[0], age =int(e[1]), salary = float(e[2])))

e[0], e[1] and e[2] are the first, second and third elements of the tuple e

representing a row (line) in RDD emps\_fields. Note that int and float are types of fields in new rows. Newly create RDD employees is now made of Row elements and is ready to be transformed into a DataFrame. You generate a DataFrame by passing an RDD of Row elements to the method createDataFrame() of class SQLContext. Doit. Show the schema of your new data frame. Select complete content of new DataFrame.Transform this DataFrame into a Temporary Table and select names of all employees who have a salary greater than 3500.Persist your DataFrame as a Parquet file and show that you could exit your pyspark or

spark-shell shell, come back into the shell and are able to read the data from the

parquet file and recreate the same DataFrame you had originally.Implement this problem in pyspark of spark-shell or as a standalone program that will be submitted to spark-submit utility.

**Approach:**

1. See attached p2.Emps.py and p2.Emps.parquet.py programs.
2. Load emps-1.txt file into a dataframe and run some filters. Save as parquet file.
3. Load emps parquet file back into a dataframe and show contents.

Load emps-1.txt into a dataframe and run some filters. Save as parquet file.

|  |
| --- |
| **[cloudera@quickstart HW4]$** *spark-submit p2.Emps.py*  ------------------Loading employee file-------------------------------  [u'Michael, 29, 3000.30', u'Andy, 30, 2500.25', u'Justin, 19, 4000.99']  ------------------Creating tuple from each row------------------------  [[u'Michael', u' 29', u' 3000.30'], [u'Andy', u' 30', u' 2500.25'], [u'Justin', u' 19', u' 4000.99']]  ------------------Creating rows of data-------------------------------  [Row(age=29, name=u'Michael', salary=3000.3000000000002), Row(age=30, name=u'Andy', salary=2500.25), Row(age=19, name=u'Justin', salary=4000.9899999999998)]  ------------------Creating dataframe----------------------------------  +---+-------+-------+  |age| name| salary|  +---+-------+-------+  | 29|Michael| 3000.3|  | 30| Andy|2500.25|  | 19| Justin|4000.99|  +---+-------+-------+  None  ------------------Dataframe schema------------------------------------  root  |-- age: long (nullable = true)  |-- name: string (nullable = true)  |-- salary: double (nullable = true)  None  ------------------Employess with salary > 3500------------------------  +---+------+-------+  |age| name| salary|  +---+------+-------+  | 19|Justin|4000.99|  +---+------+-------+  None  ---------------------Saving as parquet file---------------------------------  SLF4J: Class path contains multiple SLF4J bindings.  SLF4J: Found binding in [jar:file:/usr/lib/hive/lib/hive-exec-1.1.0-cdh5.8.0.jar!/shaded/parquet/org/slf4j/impl/StaticLoggerBinder.class]  SLF4J: Found binding in [jar:file:/usr/lib/hive/lib/hive-jdbc-1.1.0-cdh5.8.0-standalone.jar!/shaded/parquet/org/slf4j/impl/StaticLoggerBinder.class]  SLF4J: Found binding in [jar:file:/usr/lib/parquet/lib/parquet-format-2.1.0-cdh5.8.0.jar!/shaded/parquet/org/slf4j/impl/StaticLoggerBinder.class]  SLF4J: Found binding in [jar:file:/usr/lib/parquet/lib/parquet-hadoop-bundle-1.5.0-cdh5.8.0.jar!/shaded/parquet/org/slf4j/impl/StaticLoggerBinder.class]  SLF4J: Found binding in [jar:file:/usr/lib/parquet/lib/parquet-pig-bundle-1.5.0-cdh5.8.0.jar!/shaded/parquet/org/slf4j/impl/StaticLoggerBinder.class]  SLF4J: See http://www.slf4j.org/codes.html#multiple\_bindings for an explanation.  SLF4J: Actual binding is of type [shaded.parquet.org.slf4j.helpers.NOPLoggerFactory]  None  ---------------------Load empl data from parquet file---------------------------------  SLF4J: Class path contains multiple SLF4J bindings.  SLF4J: Found binding in [jar:file:/usr/lib/hive/lib/hive-exec-1.1.0-cdh5.8.0.jar!/shaded/parquet/org/slf4j/impl/StaticLoggerBinder.class]  SLF4J: Found binding in [jar:file:/usr/lib/hive/lib/hive-jdbc-1.1.0-cdh5.8.0-standalone.jar!/shaded/parquet/org/slf4j/impl/StaticLoggerBinder.class]  SLF4J: Found binding in [jar:file:/usr/lib/parquet/lib/parquet-format-2.1.0-cdh5.8.0.jar!/shaded/parquet/org/slf4j/impl/StaticLoggerBinder.class]  SLF4J: Found binding in [jar:file:/usr/lib/parquet/lib/parquet-hadoop-bundle-1.5.0-cdh5.8.0.jar!/shaded/parquet/org/slf4j/impl/StaticLoggerBinder.class]  SLF4J: Found binding in [jar:file:/usr/lib/parquet/lib/parquet-pig-bundle-1.5.0-cdh5.8.0.jar!/shaded/parquet/org/slf4j/impl/StaticLoggerBinder.class]  SLF4J: See http://www.slf4j.org/codes.html#multiple\_bindings for an explanation.  SLF4J: Actual binding is of type [shaded.parquet.org.slf4j.helpers.NOPLoggerFactory]  +-------+---+-------+  | name|age| salary|  +-------+---+-------+  |Michael| 29| 3000.3|  | Andy| 30|2500.25|  | Justin| 19|4000.99|  +-------+---+-------+ |

Load employees data back from parquet file and create the dataframe.

|  |
| --- |
| **[cloudera@quickstart HW4]$** *spark-submit p2.Emps.parquet.py*  ------------------Load empl data from parquet file--------------------  SLF4J: Class path contains multiple SLF4J bindings.  SLF4J: Found binding in [jar:file:/usr/lib/hive/lib/hive-exec-1.1.0-cdh5.8.0.jar!/shaded/parquet/org/slf4j/impl/StaticLoggerBinder.class]  SLF4J: Found binding in [jar:file:/usr/lib/hive/lib/hive-jdbc-1.1.0-cdh5.8.0-standalone.jar!/shaded/parquet/org/slf4j/impl/StaticLoggerBinder.class]  SLF4J: Found binding in [jar:file:/usr/lib/parquet/lib/parquet-format-2.1.0-cdh5.8.0.jar!/shaded/parquet/org/slf4j/impl/StaticLoggerBinder.class]  SLF4J: Found binding in [jar:file:/usr/lib/parquet/lib/parquet-hadoop-bundle-1.5.0-cdh5.8.0.jar!/shaded/parquet/org/slf4j/impl/StaticLoggerBinder.class]  SLF4J: Found binding in [jar:file:/usr/lib/parquet/lib/parquet-pig-bundle-1.5.0-cdh5.8.0.jar!/shaded/parquet/org/slf4j/impl/StaticLoggerBinder.class]  SLF4J: See http://www.slf4j.org/codes.html#multiple\_bindings for an explanation.  SLF4J: Actual binding is of type [shaded.parquet.org.slf4j.helpers.NOPLoggerFactory]  +-------+---+-------+  | name|age| salary|  +-------+---+-------+  |Michael| 29| 3000.3|  | Andy| 30|2500.25|  | Justin| 19|4000.99|  +-------+---+-------+ |

**Problem 3**. Make sure that two Hive services are up and running. You will find them in the same /etc/init.d directory you were looking for Hadoop services. Click on the browser in your Cloudera QuickStart VM. The welcome screen will open. On the top navigation bar hit Hue. Hue is Cloudera’s browser for Hadoop products. Hue might complain that its configuration is not right. I believe that some user names are missing for some products we do not care right now. On the new navigation bar that will appear select Query Editors and then select Hive. On the left navigation bar you will see a “default” database and several demo tables. In the query window to the right, type “select \* from customers” and then hit the green triangle left of the query window. Next, type “select addresses from customers”. You will see that the customers have shipping and billing addresses.Hive stores its table as Parquet files in HDFS. Use command:

$ sudo –u hdfs hadoop fs –ls /usr/hive/warehouse

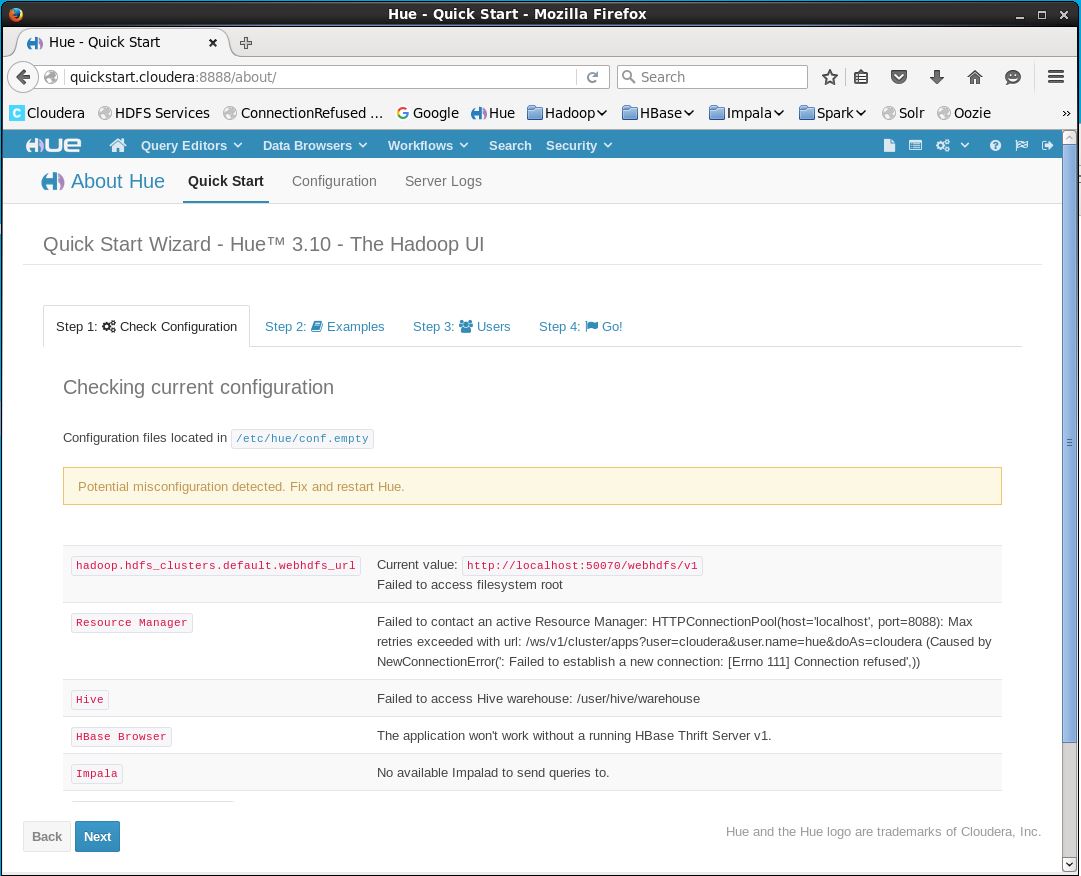
to compare the content of Hive’s warehouse HDFS directory and the table list you see in

Hue. Next use the command:

$ sudo –u hdfs Hadoop fs –cat

to expose the actual content of the file stored for the customer table. The content is half readable. That is fine. Capture a portion of it and present in your solution. What type of storage format is used for that file? You can see that “meta information in the file itself.

|  |
| --- |
| 1. Make sure Hive services are up.  **[cloudera@quickstart init.d]$** *for i in `cd /etc/init.d; ls hive-\*`; do sudo service $i start; done*  Starting Hive Metastore (hive-metastore): [ OK ]  Started Hive Server2 (hive-server2): [ OK ]  **[cloudera@quickstart init.d]$** *for i in `cd /etc/init.d; ls hive-\*`; do sudo service $i status; done*  Hive Metastore is running [ OK ]  Hive Server2 is running [ OK ]  2. Make sure Hue web service is up.  **[cloudera@quickstart init.d]$** *sudo service hue start*  Starting hue: [ OK ]  **[cloudera@quickstart init.d]$** *sudo service hue status*  supervisor (pid 32186) is running... |



|  |
| --- |
| My sqoop command fails to import data into hive for some reason.  It runs for more than 30mins but fails ultimately. I don't see any data imported under /user/hive/warehouse. I have highlighted some important warnings below. Hive metaservice is indeed running in the background. Any advise would be highly appreciated. I have highlighted ERRORs and WARnings below.  **[cloudera@quickstart]$** *for i in `cd /etc/init.d; ls hive-\*`; do sudo service $i status; done*  Hive Metastore is running [ OK ]  Hive Server2 is running [ OK ]  **[cloudera@quickstart CSCIE63]$** *sqoop import-all-tables -m 1 --connect jdbc:mysql://quickstart.cloudera:3306/retail\_db --username=retail\_dba --password=cloudera --compression-codec=snappy --as-parquetfile --warehouse-dir=/user/hive/warehouse --hive-import*  Warning: /usr/lib/sqoop/../accumulo does not exist! Accumulo imports will fail.  Please set $ACCUMULO\_HOME to the root of your Accumulo installation.  17/02/25 12:55:03 INFO sqoop.Sqoop: Running Sqoop version: 1.4.6-cdh5.8.0  17/02/25 12:55:03 WARN tool.BaseSqoopTool: Setting your password on the command-line is insecure. Consider using -P instead.  17/02/25 12:55:03 INFO tool.BaseSqoopTool: Using Hive-specific delimiters for output. You can override  17/02/25 12:55:03 INFO tool.BaseSqoopTool: delimiters with --fields-terminated-by, etc.  17/02/25 12:55:03 WARN tool.BaseSqoopTool: It seems that you're doing hive import directly into default  17/02/25 12:55:03 WARN tool.BaseSqoopTool: hive warehouse directory which is not supported. Sqoop is  17/02/25 12:55:03 WARN tool.BaseSqoopTool: firstly importing data into separate directory and then  17/02/25 12:55:03 WARN tool.BaseSqoopTool: inserting data into hive. Please consider removing  17/02/25 12:55:03 WARN tool.BaseSqoopTool: --target-dir or --warehouse-dir into /user/hive/warehouse in  17/02/25 12:55:03 WARN tool.BaseSqoopTool: case that you will detect any issues.  17/02/25 12:55:03 INFO manager.MySQLManager: Preparing to use a MySQL streaming resultset.  17/02/25 12:55:05 INFO tool.CodeGenTool: Beginning code generation  17/02/25 12:55:05 INFO tool.CodeGenTool: Will generate java class as codegen\_categories  17/02/25 12:55:05 INFO manager.SqlManager: Executing SQL statement: SELECT t.\* FROM `categories` AS t LIMIT 1  17/02/25 12:55:05 INFO manager.SqlManager: Executing SQL statement: SELECT t.\* FROM `categories` AS t LIMIT 1  17/02/25 12:55:05 INFO orm.CompilationManager: HADOOP\_MAPRED\_HOME is /usr/lib/hadoop-mapreduce  Note: /tmp/sqoop-cloudera/compile/39c03b41f84654cc0924a4ea49e76e52/codegen\_categories.java uses or overrides a deprecated API.  Note: Recompile with -Xlint:deprecation for details.  17/02/25 12:55:10 INFO orm.CompilationManager: Writing jar file: /tmp/sqoop-cloudera/compile/39c03b41f84654cc0924a4ea49e76e52/codegen\_categories.jar  17/02/25 12:55:10 WARN manager.MySQLManager: It looks like you are importing from mysql.  17/02/25 12:55:10 WARN manager.MySQLManager: This transfer can be faster! Use the --direct  17/02/25 12:55:10 WARN manager.MySQLManager: option to exercise a MySQL-specific fast path.  17/02/25 12:55:10 INFO manager.MySQLManager: Setting zero DATETIME behavior to convertToNull (mysql)  17/02/25 12:55:10 INFO mapreduce.ImportJobBase: Beginning import of categories  17/02/25 12:55:11 INFO Configuration.deprecation: mapred.jar is deprecated. Instead, use mapreduce.job.jar  17/02/25 12:55:14 INFO manager.SqlManager: Executing SQL statement: SELECT t.\* FROM `categories` AS t LIMIT 1  17/02/25 12:55:14 INFO manager.SqlManager: Executing SQL statement: SELECT t.\* FROM `categories` AS t LIMIT 1  17/02/25 12:55:18 INFO hive.metastore: Trying to connect to metastore with URI thrift://quickstart.cloudera:9083  17/02/25 12:55:18 INFO hive.metastore: Opened a connection to metastore, current connections: 1  **17/02/25 13:00:18 WARN hive.metastore: set\_ugi() not successful, Likely cause: new client talking to old server. Continuing without it.**  org.apache.thrift.transport.TTransportException: java.net.SocketTimeoutException: Read timed out  at org.apache.thrift.transport.TIOStreamTransport.read(TIOStreamTransport.java:129)  at org.apache.thrift.transport.TTransport.readAll(TTransport.java:86)  at org.apache.thrift.protocol.TBinaryProtocol.readAll(TBinaryProtocol.java:429)  at org.apache.thrift.protocol.TBinaryProtocol.readI32(TBinaryProtocol.java:318)  at org.apache.thrift.protocol.TBinaryProtocol.readMessageBegin(TBinaryProtocol.java:219)  at org.apache.thrift.TServiceClient.receiveBase(TServiceClient.java:77)  at org.apache.hadoop.hive.metastore.api.ThriftHiveMetastore$Client.recv\_set\_ugi(ThriftHiveMetastore.java:3745)  at org.apache.hadoop.hive.metastore.api.ThriftHiveMetastore$Client.set\_ugi(ThriftHiveMetastore.java:3731)  at org.apache.hadoop.hive.metastore.HiveMetaStoreClient.open(HiveMetaStoreClient.java:440)  at org.apache.hadoop.hive.metastore.HiveMetaStoreClient.<init>(HiveMetaStoreClient.java:238)  at org.apache.hadoop.hive.metastore.HiveMetaStoreClient.<init>(HiveMetaStoreClient.java:183)  at org.kitesdk.data.spi.hive.MetaStoreUtil.<init>(MetaStoreUtil.java:133)  at org.kitesdk.data.spi.hive.MetaStoreUtil.get(MetaStoreUtil.java:101)  at org.kitesdk.data.spi.hive.HiveAbstractMetadataProvider.getMetaStoreUtil(HiveAbstractMetadataProvider.java:63)  at org.kitesdk.data.spi.hive.HiveAbstractMetadataProvider.resolveNamespace(HiveAbstractMetadataProvider.java:270)  at org.kitesdk.data.spi.hive.HiveAbstractMetadataProvider.resolveNamespace(HiveAbstractMetadataProvider.java:255)  at org.kitesdk.data.spi.hive.HiveAbstractMetadataProvider.exists(HiveAbstractMetadataProvider.java:159)  at org.kitesdk.data.spi.filesystem.FileSystemDatasetRepository.exists(FileSystemDatasetRepository.java:262)  at org.kitesdk.data.Datasets.exists(Datasets.java:629)  at org.kitesdk.data.Datasets.exists(Datasets.java:646)  at org.apache.sqoop.mapreduce.DataDrivenImportJob.configureMapper(DataDrivenImportJob.java:117)  at org.apache.sqoop.mapreduce.ImportJobBase.runImport(ImportJobBase.java:267)  at org.apache.sqoop.manager.SqlManager.importTable(SqlManager.java:692)  at org.apache.sqoop.manager.MySQLManager.importTable(MySQLManager.java:127)  at org.apache.sqoop.tool.ImportTool.importTable(ImportTool.java:507)  at org.apache.sqoop.tool.ImportAllTablesTool.run(ImportAllTablesTool.java:111)  at org.apache.sqoop.Sqoop.run(Sqoop.java:143)  at org.apache.hadoop.util.ToolRunner.run(ToolRunner.java:70)  at org.apache.sqoop.Sqoop.runSqoop(Sqoop.java:179)  at org.apache.sqoop.Sqoop.runTool(Sqoop.java:218)  at org.apache.sqoop.Sqoop.runTool(Sqoop.java:227)  at org.apache.sqoop.Sqoop.main(Sqoop.java:236)  Caused by: java.net.SocketTimeoutException: Read timed out  at java.net.SocketInputStream.socketRead0(Native Method)  at java.net.SocketInputStream.read(SocketInputStream.java:152)  at java.net.SocketInputStream.read(SocketInputStream.java:122)  at java.io.BufferedInputStream.fill(BufferedInputStream.java:235)  at java.io.BufferedInputStream.read1(BufferedInputStream.java:275)  at java.io.BufferedInputStream.read(BufferedInputStream.java:334)  at org.apache.thrift.transport.TIOStreamTransport.read(TIOStreamTransport.java:127)  ... 31 more  17/02/25 13:00:18 INFO hive.metastore: Connected to metastore.  7/02/25 13:06:01 INFO hive.metastore: Closed a connection to metastore, current connections: 0  17/02/25 13:06:06 INFO hive.metastore: Trying to connect to metastore with URI thrift://quickstart.cloudera:9083  17/02/25 13:06:06 INFO hive.metastore: Opened a connection to metastore, current connections: 1  **17/02/25 13:06:17 WARN security.Groups: Potential performance problem: getGroups(user=cloudera) took 10263 milliseconds.**  **17/02/25 13:16:41 ERROR sqoop.Sqoop: Got exception running Sqoop: org.kitesdk.data.DatasetOperationException: Exception communicating with the Hive MetaStore**  org.kitesdk.data.DatasetOperationException: Exception communicating with the Hive MetaStore  at org.kitesdk.data.spi.hive.MetaStoreUtil.tableExists(MetaStoreUtil.java:192)  at org.kitesdk.data.spi.hive.MetaStoreUtil.exists(MetaStoreUtil.java:396)  at org.kitesdk.data.spi.hive.HiveAbstractMetadataProvider.resolveNamespace(HiveAbstractMetadataProvider.java:270)  at org.kitesdk.data.spi.hive.HiveAbstractMetadataProvider.resolveNamespace(HiveAbstractMetadataProvider.java:255)  at org.kitesdk.data.spi.hive.HiveAbstractMetadataProvider.exists(HiveAbstractMetadataProvider.java:159)  at org.kitesdk.data.spi.filesystem.FileSystemDatasetRepository.exists(FileSystemDatasetRepository.java:262)  at org.kitesdk.data.Datasets.exists(Datasets.java:629)  at org.kitesdk.data.Datasets.exists(Datasets.java:646)  at org.apache.sqoop.mapreduce.DataDrivenImportJob.configureMapper(DataDrivenImportJob. |

|  |
| --- |
| **[cloudera@quickstart]$** *sudo –u hdfs hadoop fs –ls /usr/hive/warehouse*  <shows empty folder> |

**Problem 4**. We will explore that customers Hive table in Spark. Follow instructions on slide 62 of lecture notes and create a HiveContext object. Use that object to transfer the data in Hive’s customers table into a Spark DataFrame object. Use that

dataFrame to tell us how many shipping addresses there are in each US state.

Implement this problem in pyspark of spark-shell or as a standalone program that will be submitted to spark-submit utility.

|  |
| --- |
| See attached **p4.customer.py** script.  Attaching python script without results as i have problems connecting to Hive from Spark.  **cloudera@quickstart HW4]$** *spark-submit p4.Customers.py*  17/02/25 15:45:20 ERROR spark.SparkContext: Error initializing SparkContext.  org.apache.hadoop.ipc.RemoteException(org.apache.hadoop.hdfs.server.namenode.SafeModeException): Cannot create file/user/spark/applicationHistory/local-1488066319001.inprogress. Name node is in safe mode.  The reported blocks 80 needs additional 209 blocks to reach the threshold 0.9990 of total blocks 289.  The number of live datanodes 1 has reached the minimum number 0. Safe mode will be turned off automatically once the thresholds have been reached.  ... |

Each problem carries 25% of the grade.

Please, describe every step of your work and present all intermediate and final results in a Word document. Please, copy past text version of all essential command and snippets of results into the Word document. We cannot retype text that is in JPG images. Please,always submit a separate copy of the original, working scripts and/or class files you used as separate files. Sometimes we need to run your code and retyping is too costly. Please include in your MS Word document only relevant portion of the console output or outputfiles. Sometime either console output or the result file is too long and including it into theMS Word document makes that document too hard to read. PLEASE DO NOT EMBEDfiles into your MS Word document. Please, submit to the class drop box. For issues andcomments visit the class Discussion Board. You can solve these problems using anylanguage of your choice.