Guidelines for Solving the Assignment

At this point, it is expected that you have already developed all the required pig scripts and Spark programs and have tested them on the sample dataset provided previously.

In this segment, please refer some basic guidelines for executing the Pig and Spark codes on the 7GB dataset in the EC2 box:

- Pig scripts can be run through "Grunt" shell.
- For Spark java programs, you will have to develop the program in Eclipse and package the entire code in a jar and execute this jar file in ec2 Unix box using "spark submit command". Detailed instructions are present in subsequent segments.
- Please note the default Java version in EC2 Unix box is Java 1.7, hence you will have to upgrade it to Java 1.8 for running Spark programs successfully. Please refer the below-mentioned document for updating the Java version.

NOTE: In the document for upgrading java version, the URL provided in the **wget** command for downloading Java 1.8 has expired. Hence, instead of the one provided in the document, use the following command for downloading JDK 1.8:

```
 wget "https://s3.amazonaws.com/javaupgrad1.8/jdk-8u161-linux-x64.tar.gz"
```

- Make sure that the output produced by Pig and Spark RDD is consistent. Some of the consistency aspects which needs to be analysed are:
    -

  - The number of output records for Pig and Spark should be same across all three methods.
  - For question 3, the count generated for each group for Pig and Spark should be same.
- Analyse both the approaches for the given problem statements with respect to the total time taken for the job to finish successfully ignoring the number of parallel operations taken by each approach. (Please note to process 7GB of data Pig is taking almost 35 mappers and 5 reducers and Spark requires 39 executors. So, the number of parallel operations required in both the cases is almost same.)

The table format to record stats is mentioned below for your reference:

Job Run Time

	Single Record Lookup	Filter	Group by Accompanied with Order by
Pig			
Spark RDD			

You should see some difference in processing time for Pig and Spark as the size of the input data is almost 7GB.

Total Count of Records in Output(The cells in each column should have the same value)

	Single Record Lookup	Filter	Group by Accompanied with Order by
Pig			

Spark RDD			

Please note that the number of records for each column has to be same. If the values are different in each column then either your Pig or Spark code is incorrect and you will have to revisit them again.

Steps for Downloading the Data

For solving the assignment you will have to download the 7GB dataset in your EC2 instance and load it into HDFS. The steps for doing the same is mentioned below:

- Enter the ec2 instance through the terminal(for Mac users) or putty(for Windows users) and enter the "root" user login.

```
[root@ip-10-0-0-219 ~]# pwd
/root
```

- Create the directory structure **/root/spark_assignment/input_dataset** and enter the "input_dataset" directory using the following sequence of commands :

-
  ```
  [root@ip-10-0-0-219 ~]# mkdir spark_assignment
  [root@ip-10-0-0-219 ~]# cd spark_assignment/
  [root@ip-10-0-0-219 spark_assignment]# mkdir input_dataset
  [root@ip-10-0-0-219 spark_assignment]# cd input_dataset/
  ```
- For downloading the dataset execute the following commands:

```
wget "https://s3.amazonaws.com/nyc-tlc/trip+data/yellow_tripdata_2017-12.csv"
wget "https://s3.amazonaws.com/nyc-tlc/trip+data/yellow_tripdata_2017-11.csv"
wget "https://s3.amazonaws.com/nyc-tlc/trip+data/yellow_tripdata_2017-10.csv"
```

wget "https://s3.amazonaws.com/nyc-tlc/trip+data/yellow_tripdata_2017-09.csv"

wget "https://s3.amazonaws.com/nyc-tlc/trip+data/yellow_tripdata_2017-08.csv"

wget "https://s3.amazonaws.com/nyc-tlc/trip+data/yellow_tripdata_2017-07.csv"

- After executing all the commands you can execute **ls -lrt** and check if all the six files are downloaded or not.
- Now for creating a directory in HDFS from "root" user, you will have to switch to "hdfs" user and run the command for creating the directory structure. The command for switching to HDFS user is:

  [root@ip-10-0-0-219 input_dataset]# su - hdfs

  The command to create the directory in HDFS is:

  [hdfs@ip-10-0-0-219 ~]$ hadoop fs -mkdir -p /user/root/spark_assignment/input_dataset

- Change the owner of the directory created in the previous step to "root": **[hdfs@ip-10-0-0-219 ~]$ hadoop fs -chown root /user/root/spark_assignment/input_dataset**
- Fire the exit command and toggle back to the root user:

  [hdfs@ip-10-0-0-219 ~]$ exit

NOTE: The steps for creating the above-mentioned directory structure will be the same as those mentioned in the Map-Reduce module for running word count program. Students who need more clarity regarding these steps can refer to the following document from page number 8 to page number 10:

- Using the “put” command add all the input data into the HDFS location created in the previous step. Please note you are running this command from the same directory in which the input files are present.

hadoop fs -put yellow_tripdata_* /user/root/spark_assignment/input_dataset

- Execute the following command and check if the datasets are present in the HDFS location /user/root/spark_assignment/input_dataset/ or not :

hadoop fs -ls /user/root/spark_assignment/input_dataset/

Now use this directory “**/user/root/spark_assignment/input_dataset/yellow_tripdata_***” as the input location for all your Spark and Pig programs.

Guidelines for Running Spark Codes in EC2 Unix Box

For running spark codes in Amazon EC2 instance its required to package the codes in ".jar" format. You already know that we need some dependency jars to compile or build a spark job. We will be using maven for automating the addition of dependency jars to the project's build path and generating the .jar file with all the dependency classes included in it.

The link to the document having steps for building and generating the ".jar" file for a spark java program is mentioned below for your reference:

The link to the pom.xml used in the above demonstration is mentioned below for your reference:

Steps for running the jar file in EC2 instance is given below:

- Mac or Unix users can scp the jar file whose name ends with “jar-with-dependencies.jar” to the EC2 Unix box. You can use the below mentioned SCP command:

scp -i <path of .pem> <jar file path> ec2-user@<public ip>:/home/ec2-user/

NOTE: Windows users can follow from page number 4 to page number 7 in the following document attached for transferring the jar files to EC2 instance using WinSCP:

- You will again have to refer the previous document named "Running Map Reduce Program in EC2 Instance" from page number 12 to 17 for modifying the following YARN configurations:
    - **yarn.nodemanager.resource.memory-mb** to 14 GB (default 1GB)
    - **yarn.scheduler.maximum-allocation-mb** 10 GB (default 1GB)
- Once the jar file is moved to AWS EC2 Unix box and the yarn configurations are fixed, execute the following command for running spark job:

spark-submit --class <*full path of the main class in the java project*>\

--master yarn --deploy-mode client \

--name <*any_user_defined_name*> --conf "spark.app.id=<*any_user_defined_id*>" \

<*jar name with full qualified path*> arg1 arg2 ...

Please note in the above command arg1, arg2 ... etc will go as command line arguments to the main method of your Spark program. Hence, in the above-mentioned command, you can pass the input path and output path as arg1 and arg2 respectively. As the input path for all the three spark programs is same, you can also hard code it in the code and only pass the output path as arg1 in the spark submit command. Use the command line arguments appropriately in the main method else the code will not run properly.

Few examples of the spark run commands is given below for your reference:

```
spark-submit --class spark.testproject.App \
--master yarn --deploy-mode client \
--name aws_spark --conf "spark.app.id=spark_aws_run" \
testproject-0.0.1-SNAPSHOT-jar-with-dependencies.jar
/user/root/spark_assignment/yellow_tripdata/yellow_tripdata*
/user/root/spark_assignment/output/single_row_lookup_SparkRDD
```

```
spark-submit --class Spark.Spark_Single_Row_Lookup.SparkSingleRowLookup \
--master yarn --deploy-mode client \
--name spark_singlerow_lookup --conf "spark.app.id=spark_singlerow_lookup" \
Spark_Single_Row_Lookup-0.0.1-SNAPSHOT-jar-with-dependencies.jar
/user/root/spark_assignment/input_dataset/yellow_tripdata*
/user/root/spark_assignment/output/single_row_lookup/
```

NOTE: In the above command the full path of the jar is not specified because the command is run from the same location where the jar is present.

Submission Guidelines

Student submission will include the following things:

- Three Jar Files, i.e. for Single Row Lookup, filter operation and group by with order by operation.
- A PDF document containing all the pig scripts and java codes.
- A PDF document having both the tables filled with appropriate stats. Few sentences describing your inference based on the observation filled in the tables.
- Copy paste the entire job run logs for all six programs in a single doc and share it in PDF format. Please provide proper headings before the logs so that it is easy to identify which log belongs to which assignment question
- Along with the logs, also include the "spark submit" run commands used to execute the three spark programs in an EC2 instance
- In a PDF doc, provide Pig and Spark output for problem 1 and 3. A sample command is given for your reference: **hadoop fs -cat /user/root/spark_assignment/output/single_row_lookup_SparkRDD/part***
- Also, provide the line count as of problem 2 for both Pig and Spark. A sample command for finding the line count is given for your reference: **hadoop fs -cat /user/root/spark_assignment/output/single_row_lookup_SparkRDD/part* | wc -l**