CSCE 587 Spring, 2020 Hadoop Project/Take Home Final Exam **Due: 5/5/20 Midnight**

Preliminary
The airline data that we've used for learning how to use map-reduce includes fields for the time it takes to taxi in (`TaxiIn`) and taxi out (`TaxiOut`) as well as airline (`UniqueCarrier`) and origin airport (`Origin`). The following three problems require you to process the airport data to answer questions about these times.

For each question, use the following data sets and approach:

Dataset: ~rose/public_html/590B/airline/200X.csv, where X is 0, 2, 3, 4, 5, 6, 7, or 8
If you are using wget to transfer the files, the command is of the form:

```
wget https://cse.sc.edu/~rose/590B/airline/2000.csv
```

Note: We are not including 2001.csv in our analysis since that data set is corrupted.

Method 1: map-reduce
Method 2: Pig Latin
Method 3: HiveQL

You were probably wondering when the data would become big. Each of these files contains records for several million flights. In total there are more than 53 million flights in this data set.

My suggestion is that you develop your code (map-reduce, pig, and hive) using the smallest data set of 12 records (testNA.csv). The 12 records in this data set include records for 5 different origin airports as well as missing values (NA). This data set is small enough that you can compute the expected results by hand and verify that your solution is correct for this data set. Next, try the larger data set containing ~25K flight records (test_25K.csv). When you get that to work, then move on to the large file comprised of all 8 years of data.

Development Datasets:

https://cse.sc.edu/~rose/587/CSV/testNA.csv
https://cse.sc.edu/~rose/587/CSV/test_25K.csv

Creating the 8-year data file:
1) Start by using `wget` to transfer each data set to the Linux filesystem of your virtual machine. 2) Next, concatenate the 8 years of data in the Linux filesystem into a single file in HDFS. Below is an example of the syntax where you have the 8 files in the current working directory in the Linux file system and you want concatenate and place the result in a single file in your top-level HDFS directory. (Note: there are spaces between "-put" and "-" and "wholeEnchilada.csv"): My suggestion is that you copy and paste this command directly into the Linux terminal on your VM:

cat 200?.csv | hadoop fs -put - wholeEnchilada.csv

Saving output to the Linux file system on your VM:
You will need to transfer your map-reduce and pig results to your VM's Linux file system. Consider the case where the map-reduce/pig directory is "HDFSmyOuput".

Map/Reduce case: Your output is a single file

From the Linux prompt type:

```
hadoop fs -get HDFSmyOuput/p* myOutput
```

This copies the results to "myOuput" in the linux current working directory.

Pig case: Your output is in two files. (Note: your output for the 8-year data set will be in two files.)

From the Linux prompt type:

```
hadoop fs -getmerge HDFSmyOutput Pig_Output
```

This copies and merges the two files into the single file "myOuput" in the linux current working directory.

CSCE 587 Spring, 2020 Hadoop Project/Take Home Final Exam **Due: 5/5/20 Midnight**

Transfering your output files from the linux files system of the VM to your home computer:
1) Using MobaXterm, click on the Session icon (leftmost icon)
2) click on the SFTP icon in the popup window
3) enter in your VM name in the remote host field
4) enter 'maria_dev' in the username field
5) set the port number to 2222
6) click on the 'ok' button at the bottom of the popup window
7) when prompted, type in your maria_dev password
MobaXterm will display the local file system in the left pane and the VM filesystem in the right pane
To transfer a file from the VM Linux system, simply click on the file and drag it over to the local file system

Problem1 [33 points]: Find minimum, maximum and average taxi time by origin-airport using Map-Reduce

This problem is an organic extension of our understanding of map-reduce from counting words, to counting flights, to calculating average arrival delay, to now calculating the minimum, maximum and average taxi time by origin airport. Start by extracting the `TaxiIn`, `TaxiOut`, `UniqueCarrier,` and `Origin` columns. For definition of the columns see: http://stat-computing.org/dataexpo/2009/the-data.html .

The goal of this problem is to find the minimum, maximum and average taxi time by airline/origin airport. Instead of returning a single value for the **value** and a single value for the **key** portions of the **key-value** tuple, we explore compound keys and values.
Key: We define an airline/origin airport combination as a pair of an airline combined with the origin airport. We will use the hyphen (`-`) as the separating character when pasting the two strings together. For example, flights on United Airline (UA) departing Columbia (CAE) would be UA-CAE. This problem will require you to paste the airline string and origin airport strings together to effectively create a multi-value airline/airport key.
Hint: the key returned by your map function should be the airline/airport combo. We've been using the "join" function in python to paste the key and value together in the mapper in our labs,.e.g. `"\t".join([key, value])`
I suggest using the "join" command in python to paste the airline and origin strings together with a hyphen, e.g., `"-".join([UniqueCarrier, Origin])` which would give you something like `UA-CAE`

Value: In this problem you are to find the minimum, maximum, and average sum of Taxi time (`TaxiIn + TaxiOut`) by airline-origin airport.
You can either deal with NA values in your map function or in your reduce function. Your mapper can either return separate TaxiIn and TaxiOut values or the sum TaxiIn+TaxiOut. However you do it, your reducer's expected input should match your mapper's output.
In any case, in your mapper use "join" to paste you compound key and values together `"\t".join([key, value])` if one value (i.e. sum of TaxiIn and TaxiOut) or `"\t".join([key, value1, value2])` if two values (i.e TaxiIn and TaxiOut).

Your reducer should then find the minimum, maximum, and average values for TaxiIn+TaxiOut. It probably makes the most sense to do this in the reducer. The output columns should be:
```
Airline-Origin     Min     Max     Average
```
Example of the first several lines of output from the 8-year data set:
```
[maria_dev@sandbox-hdp ~]$ hadoop fs -get MR_Output/part-00000
[maria_dev@sandbox-hdp ~]$ mv part-00000 TaxiMapReduceResults
[maria_dev@sandbox-hdp ~]$ more TaxiMapReduceResults
9E-ABE  0       220     28
```

```
9E-ALB  0       101     26
9E-ALO  0       83      24
9E-ATL  9       300     29
9E-ATW  0       191     27
9E-AUS  0       104     24
9E-AVL  0       168     26
9E-AVP  0       107     28
9E-AZO  0       152     26
9E-BDL  0       132     27
9E-BGM  0       157     25
9E-BGR  0       209     29
9E-BHM  0       161     26
9E-BIL  12      101     27
9E-BIS  0       93      21
9E-BJI  0       99      20
```

Submit the python code for your mapper, reducer, and your output results.

1) Name your mapper file TaxiMapper.py
2) Name your reducer file TaxiReducer.py
3) Name your output file TaxiMapReduceResults

Hints:

1) Start by refreshing your understanding of the aveArrDelay map-reduce example from 4/16/20. The solution to that problem is very close to this problem.
2) Be sure to explicitly cast objects to be the type you want in your reducer. If you want to interpret X as an int, use int(X).
3) You can get your output results by one of two methods:
    a. Login to Ambari (vm-hadoop-xx.cse.sc.edu:8080), select files view, navigate to user/maria_dev and then to the directory holding your results. Click on the output file and then click on the download button to download to your home machine.
    b. Alternatively, transfer the output from HDFS to the Linux files system of the vm and then use MobaXterm as explained on the top of the second page of this document.

Problem 2 [33 points]: Find minimum, maximum and average taxi time by origin-airport using Pig

Create a pig solution for finding the minimum, maximum and average taxi time by airline/origin airport. This is the same problem you solved in map-reduce, but now you are creating a solution in Pig Latin. When indexing columns, don't forget that column numbers start from 0 in Pig Latin.

Submit your Pig Latin script and your output results.

1) Name your Pig Latin script Taxi.pig
2) Name your output file TaxiPigResults

Hints:

1) Start by refreshing your understanding of the aveArrDelay Pig Latin example from 4/16/20. The solution to that problem is very close to this problem.
2) In Pig Latin, missing values are referred to as nulls. In the case of arithmetic, if either subexpression is null, the resulting expression is null. In our case, if either `TaxiIn` or `TaxiOut` is null, then the sum will also be null. Since the MIN(), MAX(), and AVG() function handle null values, you don't need worry missing values for this problem. The functions for min and max are MIN() and MAX(), respectively. Hint: Pig is case sensitive.

3) Pig will likely produce the output in the form of two files when you process the combined 8-year dataset. In the commands below (on the VM), assume that your pig results are stored in a directory called `HDFSmyOutput` in HDFS.
    a. You will need to merge them: `hadoop fs -getmerge HDFSmyOutput tmpResults` The above command can be used to merge the pig results and transfer them to a file called `tmpResults` in the current Linux current working directory.
    b. Since merging the files does not interleave the results, you will still need to sort the results: `sort <tmpResults >TaxiPigResults`
    c. This command sorts data from `tmpResults` and saves it in `TaxiPigResult` in the Linux current working directory. You can then use MobaXterm as explained on the top of the second page of this document. Alternatively, you can put the sorted results back to HDFS using a hadoop put command and then login to Ambari (vm-hadoop-xx.cse.sc.edu:8080), select files view, navigate to user/maria_dev and then to the directory holding your results. Click on the output file and then click on the download button to download to your home machine.

Problem 3 [34 points]: Find minimum, maximum and average taxi time by origin-airport using HIVE

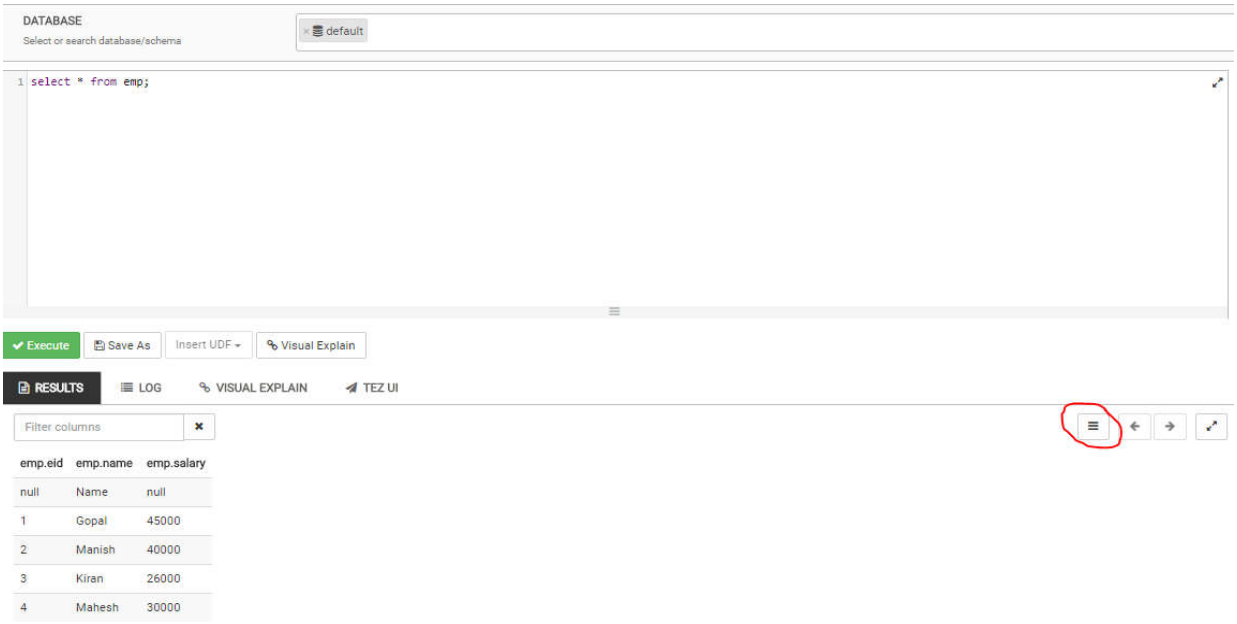
Create a hive solution for finding the minimum, maximum and average taxi time by airline/origin airport. This is the same problem you solved in map-reduce, but now you are creating a solution in HiveQL. When indexing columns, don't forget that column numbers start from 1 in HiveQL.

Submit your HiveQL script and your output results.

1) Name your HiveQL script Taxi.hive
2) Name your output file TaxiHiveResults.csv

Hints:

1) Start by refreshing your understanding of aveArrDelay HiveQL example from 4/16/20. The solution to that problem is very close to this problem.
2) Follow the same general outline:
    a. Create a temp table to read in the data set.
    b. Create a second table and extract UniqueCarrier, Origin, TaxiIn, and TaxiOut from the temp table.
3) You can use the HiveQL command CONCAT() to paste the airline and origin together.
   My solution uses CONCAT() in a subquery to paste the airline and origin together and returns the three fields: airline-origin, TaxiIn, and TaxiOut. I also check to for missing values (NA) for TaxiIn and TaxiOut in this subquery.
4) My outer query groups by airline-origin and finds the MIN(), MAX(), and AVG() taxi times. Note: Hive is case sensitive.
5) **Save output from HIVE:** You can save the output of your select command by selecting the "Download as CSV option". This will download to the downloads directory on your computer at home. See the figure below. Click on the button circled in red to download:



6) If you compare the Hive results (TaxiHiveResults.csv) to the map-reduce results, you will notice that some airline-origin entries appear to be missing, e.g. `9E-ABE` and `9E-BDL`. See below:

airlineorigin	mintaxi	maxtaxi	avgtaxi
9E-ALB	0	101	26.54437
9E-ALO	0	83	24.94542
9E-ATL	9	300	29.65468
9E-ATW	0	191	27.14186
9E-AUS	0	104	24.8308
9E-AVL	0	168	26.95926
9E-AVP	0	107	28.39558
9E-AZO	0	152	26.15218
9E-BGM	0	157	25.59056

Actually, they appear later in the file. Hive processes the large dataset in the same way that Pig has, i.e, two files that were merged. We need to sort results. Since this is a csv file, we can use excel to sort on the first column. Now the results match those of map-reduce and Pig. We see that the entries `9E-ABE` and `9E-BDL` appear where we expect them.

airlineorigin	mintaxi	maxtaxi	avgtaxi
9E-ABE	0	220	28.01667
9E-ALB	0	101	26.54437
9E-ALO	0	83	24.94542
9E-ATL	9	300	29.65468
9E-ATW	0	191	27.14186
9E-AUS	0	104	24.8308
9E-AVL	0	168	26.95926
9E-AVP	0	107	28.39558
9E-AZO	0	152	26.15218
9E-BDL	0	132	27.66311
9E-BGM	0	157	25.59056