Week 6 Assessment Task

Part A - Hive with text (4 marks)

In Part A your task is to answer a question about the data in an unprocessed text file, first using MRJob, and then using Hive. By using both to answer the same question about the same file you can more readily see how the two techniques compare.

When you click the panel on the right you'll get a connection to a server that has, in your home directory, a text file called "walden.txt", containing some sample text (feel free to open the file and explore its contents)(it's an extract from Walden, by Henry David Thoreau).

In this text file, each line is a **sentence**. It is worth noting that there are **multiple spaces** at the end of each line in this unprocessed text file.

Your task is to **find the average word lengths according to the first letters of sentences**. For example, given a toy input file as shown below:

```
Aaa bbb cc.
Ab b.
```

The output should be:

```
"Letter A:"    2.6
```

Because, for A, we have (3*3+2*2)/5 = 2.6.

You can assume that sentences are separated by full stops, and words are separated by spaces. For simplicity, we **include all punctuations**, like ',' and '.', when calculating word length, like what we did in example1 and example2. (So, the length of 'cc.' is 3 instead of 2.) The case of letters can be ignored.

Given the walden.txt file as input, the format of the output is "*letter*: *avg_word_length*" (The result should be rounded to two decimal places, with round(*x*,2)), as shown below:

```
"Letter A:"    4.17
"Letter S:"    4.32
"Letter T:"    4.09
"Letter I:"    4.16
"Letter W:"    4.89
"Letter B:"    4.82
"Letter F:"    4.18
```

(There is no need to sort the results or remove the quotation marks.)

Write a Hive script to do this. A file called "script.hql" has been created for you - you just need to fill in the details. You should be able to modify Hive scripts that you have already seen in this week's content. You might use some User-Defined Functions (UDFs) which can be found here.

You can test your script by running the following command (it tells Hive to execute the commands contained in the file script.hql):

```
$ hive -f script.hql
```

This is worth 4 marks

When you are happy that your job and script are correct, click "Submit".

Part B - Spark SQL with CSV (2 marks)

In Part B your task is to answer a question about the data in a CSV file, first using Spark RDDs, and then using Spark DataFrames and SQL. By using both to answer the same question about the same file you can more readily see how the two techniques compare.

When you click the panel on the right you'll get a connection to a server that has, in your home directory, the CSV file "orders.csv". It's one that you've seen before. Here are the fields in the file:

```
OrderDate (date)
ISBN (string)
Title (string)
Category (string)
PriceEach (decimal)
Quantity (integer)
FirstName (string)
LastName (string)
City (string)
```

Your task is to **find the number of books ordered each day,** sorted by the number of books **descending**, then order date **ascending**.

Your results should appear as the following:

```
2009-04-03,10
2009-04-02,8
2009-04-01,7
2009-04-04,6
2009-03-31,5
2009-04-05,4
2009-04-08,4
```

Write a Python program that uses Spark DataFrames and SQL to do this. A file called "sql.py" has been created for you - you just need to fill in the details. Again, you should be able to modify programs that you have already seen in this week's content.

You can test your program by running the following command:

```
$ spark-submit sql.py
```

Please save your results in the '**result-sql**' folder in HDFS.

When you are happy that your two programs are correct, click "Submit".

Part C- Spark SQL with CSV (6 marks)

COVID-19 has affected our lives significantly in recent years. In Part B your task is to do a data analysis task over a COVID-19 data set stored in the CSV format, first using Spark RDD, and then using Spark DataFrames and SQL. By using both to answer the same question about the same file you can more readily see how the two techniques compare.

The COVID-19 dataset contains the cases by notification date and postcode, local health district, and local government area in NSW, Australia. The dataset is updated daily, except on weekends. Here are the fields in the file:

```
notification_date (date) -- e.g. 2020-03-29, 2020-03-30 etc.
postcode (integer) -- e.g. 2011, 2035, etc.
lhd_2010_code (string) -- local health district code, e.g. X720, X760, etc.
lhd_2010_name (string) -- local health district name, e.g. South Eastern Sydney, Northern Sydney, e
lga_code19 (string) -- local government area code, e.g. 17200, 16550, etc.
lga_name19 (string) -- local government area name, e.g. Sydney (C), Randwick (C), etc.
```

When you click the panel on the right you'll get a connection to a server, and in your home directory you can see a sample of the data set named "cases-locations.csv".

Your task is to find **the maximum daily cases number** in **each local health district (lhd)** together with **the date**. Each line of your result should contain the **local health district**, the **local health district code**, the **date** and the **maximum daily increase of total confirmed cases**. The results should be sorted first by the daily increase in **descending** order, and then by the date in **ascending** order, and finally by the local health district name(lhd_2010_name) in **descending** order. For a certain local health district, if there are multiple dates that have the same maximum daily cases number, please return all such dates.

For example, given the sample data set, your results should be as below:

```
Northern Sydney,X760,2020-03-27,44
South Eastern Sydney,X720,2020-03-27,41
Western Sydney,X740,2020-03-29,24
Hunter New England,X800,2020-03-28,22
South Western Sydney,X710,2020-03-27,18
Sydney,X700,2020-03-29,16
Central Coast,X770,2020-03-27,15
Illawarra Shoalhaven,X730,2020-03-27,14
Nepean Blue Mountains,X750,2020-03-27,10
Mid North Coast,X820,2020-03-28,7
Southern NSW,X830,2020-03-27,6
Northern NSW,X810,2020-03-27,5
Northern NSW,X810,2020-03-30,5
Murrumbidgee,X840,2020-03-30,5
Western NSW,X850,2020-03-27,4
```

```
Far West,X860,2020-03-31,1
Far West,X860,2020-04-02,1
```

Write a Python program that uses Spark DataFrames and SQL to do this. A file called "sql.py" has been created for you - you just need to fill in the details. Again, you should be able to modify programs that you have already seen in this week's content.

You can test your program by running the following command:

```
$ spark-submit sql.py
```

Please save your results in the '**result-sql**' folder in HDFS.

When you are happy that your two programs are correct, click "Submit".