**INF 553 Homework 3**

Versions: Spark 2.3.1 Scala 2.11

My scala code was saved in a package called Georgios.HW3.
As a result in front of the scala name file, I have to specify the package.

This is the general code required to run it successfully:
spark-submit --class Georgios.HW3.SON Georgios_Iliadis_SON.jar <inputFile> <support> <outputFile>

This is the code to check the required support threshold values with each of the input files as specified in the homework:

Problem 1:
spark-submit --class Georgios.HW3.SON Georgios_Iliadis_SON.jar yelp_reviews_test.txt 30 Georgios_Iliadis_SON_yelp_reviews_test_30.txt
spark-submit --class Georgios.HW3.SON Georgios_Iliadis_SON.jar yelp_reviews_test.txt 40 Georgios_Iliadis_SON_yelp_reviews_test_40.txt

Problem 2:
spark-submit --class Georgios.HW3.SON Georgios_Iliadis_SON.jar yelp_reviews_small.txt 500 Georgios_Iliadis_SON_yelp_reviews_small_500.txt
spark-submit --class Georgios.HW3.SON Georgios_Iliadis_SON.jar yelp_reviews_small.txt 1000 Georgios_Iliadis_SON_yelp_reviews_small_1000.txt

Problem 3:
spark-submit --class Georgios.HW3.SON Georgios_Iliadis_SON.jar yelp_reviews_large.txt 100000 Georgios_Iliadis_SON_yelp_reviews_large_100000.txt
spark-submit --class Georgios.HW3.SON Georgios_Iliadis_SON.jar yelp_reviews_large.txt 120000 Georgios_Iliadis_SON_yelp_reviews_large_120000.txt

Notes:
I had to use
```
//conf.set("spark.network.timeout", "600s")
//conf.set("spark.sql.broadcastTimeout", "36000s")
//conf.set("spark.rpc.askTimeout", "600s")
//conf.set("spark.yarn.executor.memoryOverhead", "809")
```

on my machine for the large(Problem 3) to work since I was getting a timeout exception.
I am not sure if it is needed on your machine, that's why I left it as a comment in the first lines of my main method.

While running on my machine, for problem 3 I also used --executor-memory 8G --driver-memory 8G on the command line, after the Georgios.HW3.SON and before Georgios.HW3.SON.jar.

First you have to cd where the jar file is where I have the data(small and test) in the directory.
I have all the solution files at the folder OutputFiles with the names required.
The execution times can be printed when you run the program, the execution table is shown below.

For implementing the SON algorithm, I basically used the notes given in the class.
First I implemented Apriori algorithm, the two passes of it and then I compute using the Apriori the frequent itemsets. At the end, using FinalCounts, I make sure that they are correct, and then I write my results to the text file as required.

**Problem 2 execution Table:**

| Support Threshold | Execution Time |
|---|---|
| 500 | 25-27sec |
| 1000 | 50-52sec |

**Problem 3 execution Table:**

| Support Threshold | Execution Time |
|---|---|
| 100000 | ~500sec |
| 120000 | ~450sec |

The Execution Table has the time in ranges since for every time I run it, it's not always the same.

Bonus Question:
We used a large support threshold for problem 3 because if we used a small one, it would have taken forever. The data is large, so there would be many items that for example appear 30 times in such a huge data. We use a high support threshold because we want to 'eliminate' a lot of data and just look at the items that appear for a large number of times.

After creating the candidates, I get in a loop and I call the second pass of the algorithm, while the cardinality increases. So I will get pairs, triples etc.
The execution will be slow when the the data will be very large, and when there are many candidates as the cardinality increases.
Also the execution will be slow when the support threshold is at a low value, like Problem 1 and 2. Many more items will be considered as candidates so a lot more work needs to be done with such large data provided.

I left the filename of my scala code as SON.scala since it is included in a package called Georgios.HW3, so there is no point of adding my name twice.

Georgios Iliadis
3668057286