Name: Priyanka Shah
USC ID: 3853888644
Email: shahph@usc.edu

Spark Version: **spark-2.1.0-bin-hadoop2.7**
Python Version: **2.7**

1.Succinctly describe your approach to implement the algorithm.

I have implemented SON algorithm to calculate frequent itemset. Inside phase 1 of SON algorithm, I have used A-priori algorithm. I have created all the combinations of singleton, doubleton, tripleton… and found out the frequent itemsets from them by counting them in original basket. If they occur greater than or equal to support, then I add it to Frequent itemset.

2. Command line command to execute your program

Command to run : Python Priyanka_Shah_SON.py [arg1 = Case Number] [arg2 = path to input file] [arg3 = Support]

Example of the command line: python Priyanka_Shah_SON.py 2 ml-20m/ratings.csv 3000

**** I have set Spark Home variable inside my Source code (Priyanka_Shah_SON.py file). I have commented out that 4 line of code for setting the spark home variable. If you want to run the program, either you need to set spark home as environment variable or change 4 lines of my code and give respective path names of your spark directory inside my source file.**

***** I HAVE USED SUPPORT VALUE 4 FOR CASE 2 OF PROBLEM 1.

Problem 1,2,3 Execution Table:

Case	Support	Time (Seconds)
Problem 1 - Case1	3	16
Problem 1 - Case2	4	86
Problem 2 - Case1	120	5
Problem 2 - Case1	150	8
Problem 2 - Case2	180	24
Problem 2 - Case2	200	157
Problem 3 - Case1	29000	158
Problem 3 - Case1	30000	140
Problem 3 - Case2	2500	126
Problem 3 - Case2	3000	98

Bonus (5pts): Describe why did we need to use such a large support threshold and where do you think there could be a bottleneck that could result in a slow execution for your implementation, if any.

New Support = Original Support (Divided by) / Number of Partitions

If there are high number of partitions and the original value of support is less than new support will be even lesser.If the support value is very low then all the itemsets will come as frequent itemset and the computation and memory needed will increase very much. If we keep the support value high enough then number of candidate frequent itemsets will be less and so we need to count less number of items from our original basket.

For Problem 1, I ran my program using only one partition because the original support value is very less (3 for case 1 and 4 for case 2). So if the partition increases then new support becomes 1 where all the itemsets becomes eligible to count for frequency and so we need to count all of them in basket which takes a lot of time. If you don’t keep partition 1 then I think it will take a lot of time and it could be bottleneck.