Assignment 2

The following are the steps I followed for execution
Firstly I had placed all input files needed and the python script(2.7) in the bin folder of spark 2.2.1

Then I generated all the 10 output files using the following command. Everytime I generated an output file I renamed it to the specific file name and then placed in the outputfile folder

```
spark-submit Shyamala_Sundararajan_SON.py 1 Small2.csv 3
spark-submit Shyamala_Sundararajan_SON.py 2 Small2.csv 5

spark-submit Shyamala_Sundararajan_SON.py 1 MovieLens.Small.csv 120
spark-submit Shyamala_Sundararajan_SON.py 1 MovieLens.Small.csv 150
spark-submit Shyamala_Sundararajan_SON.py 2 MovieLens.Small.csv 180
spark-submit Shyamala_Sundararajan_SON.py 2 MovieLens.Small.csv 200

spark-submit Shyamala_Sundararajan_SON.py 1 MovieLens.Big.csv 30000
spark-submit Shyamala_Sundararajan_SON.py 1 MovieLens.Big.csv 35000
spark-submit Shyamala_Sundararajan_SON.py 2 MovieLens.Big.csv 2800
spark-submit Shyamala_Sundararajan_SON.py 2 MovieLens.Big.csv 3000
```

The execution time

Case1	MovieLens.Small.csv	Case2	MovieLens.Small.csv
120	18.60s	180	492.52s
150	7s	200	294.088s

Case1	MovieLens.Big.csv	Case2	MovieLens.Big.csv
30000	590.76s	2800	289.834s
35000	367.476s	3000	248.87s

For the algorithm

1.I converted the given csv file to spark RDD then then extracted the user_id and movie_id . Based on the caseNo I either stotred it as user_id, set(movie_id) or movie_id, set(user_id)
2.For singletons I had used mappartitions on this RDD and simply counted the occurrences of each item and filtered those whose count was above or equal to threshold.

3.Using the above result I used to generated pairs . Now I passed these candidate pairs to Phase1_SON_MR function which would count the frequency of each candidate pair in that chunk of data
4.I had reduced the threshold support for each chunk and only sent back those candidate pairs whose count was above this new support threshold.
5.The new candidate pairs are sent to the phase2_SON_MR which would the frequent itemsets and these itemsets will be filtered based on the actual support threshold.

6. For rest of the frequent itemsets generation these steps a re followed
Now that we have frequent pairs I assign it as candidate sets which is passed on the generateCandidates function which will generate candidate tuples of size k given frequent pairs of size k-1
7. The result from generateCandidate sets will be used for phase1_SON_MR to count the occurrences of each candidate tuple in the chunk of data. Also a filter step is introduced before mapPartitions to filter out those items in the chunk data whose length is less than the candidates length.

8. The result from 7 will be sent to phase2_SON_MR to generate frequent itemsets and this process will be repeated till either the candidate pair is empty of frequent pair becomes empty

The result is then written to a file