**Assignment**

Resilient Distributed Datasets (RDD) is a simple and immutable distributed collection of objects. Each RDD is split into multiple partitions which may be computed on different nodes of the cluster. In Spark, every function is performed on RDDs only. RDD is the building block of Spark processing, Spark works on data locality principle. Worker nodes takes the data for processing that are nearer to them. By doing partitioning network I/O will be reduced so that data can be processed a lot faster.

In Spark, operations like co-group, groupBy, groupByKey and many more will need lots of I/O operations. In this scenario, if we apply partitioning, then we can reduce the number of I/O operations rapidly so that we can speed up the data processing. Similarly, sometimes in order to speed up the execution we need to increase the parallelism. Parallelism can be increased only by having more number of tasks.Number of tasks directly depends upon on the number of partitions in the RDD.

In this assignment, you need to partition the given data NYSE data into 6 partitions such that Stocks having same initial character should go to same partition.

Example: Stocks such as LTC , LUV , LTD,LVB should be stored in the same partition as they name is starting with same ‘L’ character.

Schema of the given data is:

exchange,stock\_symbol,date,stock\_price\_open,stock\_price\_high,stock\_price\_low,stock\_price\_close,stock\_volume,stock\_price\_adj\_close

**Expected steps:**

1. Read the given NYC data and convert it into RDD, Please check number of partitions

2. Perform map() transformation to parse the data set by using ‘,’ as delimiter . After parsing the data take the stock symbol (index no 1) and stock\_price\_close(index 6) and convert the data into RDD1( named as “closePriceBySymbol”) containing tuple of symbol and amount.

3. Find out max closing price for each of the stock. You need to sort the output based on the closing price of the stock and assign it to the RDD2(( named as “maxPriceSymbolRDD”). This RDD will contain stock symbol and max price sorted on the closing price. Please ensure to run 10 instances of this transformation tasks

4. This RDD2(( named as “maxPriceSymbolRDD”) you need to save in local or HDFS directory “stock\_min\_max”

5. Now, you need to custom partition the data such that stocks having same initial character should go to same partition. You need to write Customer Partition class and override getPartition() method . In this method you need to write logic to distribute stocks to 6 different partitions.

6. Use groupByKey transformation function to use your custom partition class. Output RDD of this function should be pair RDD(named as stockGroupedData) having stock symbol as key and stock\_close\_price as value. Check number of partitions in this RDD

7 On this RDD(named as stockGroupedData) perform mapPartition transformation to find min and max stock\_close\_price for each of the 6 paritions. Save this output RDD “minMaxForPartition” into local or HDFS directory “partition\_output”

8. Finally, you need to find out the min and max of the stock\_close\_price from all the 6 partitions and print it in the console.