Java Code for Spark Assignment

- lookUpRDD

```
package spark.question1;


import org.apache.log4j.Level;

import org.apache.log4j.Logger;

import org.apache.spark.SparkConf;

import org.apache.spark.api.java.JavaRDD;

import org.apache.spark.api.java.JavaSparkContext;


public class lookUpRDD {

	public static void main(String[] args) throws Exception {


		/********

		 * Problem Statement *********

		 *

		 * Fetch the record having VendorID as '2' AND tpep_pickup_datetime as

		 * '2017-10-01 00:15:30' AND tpep_dropoff_datetime as '2017-10-01 00:25:11' AND

		 * passenger_count as '1' AND trip_distance as '2.17'

		 */

		JavaSparkContext sc = null;

		try {

			// Logger.getLogger("org").setLevel(Level.ERROR);

			/**

			 * "System.setProperty" is written to avoid "java.io.IOException: Could not
```

```
                * locate executable null\bin\winutils.exe in the Hadoop binaries."
                */
                System.setProperty("hadoop.home.dir",
"/user/root/spark_assignment/winUtils/bin/winutils.exe");
                // C:\\Program Files\\winUtils
                SparkConf conf = new
SparkConf().setAppName("lookUp").setMaster("local[2]");

                sc = new JavaSparkContext(conf);

                JavaRDD<String> tripDetails = sc.textFile(args[0]);

                /**
                * To remove Header
                */
                JavaRDD<String> cleanTripDetails = tripDetails.filter(line -> (isNotHeader(line)
&& !(line.isEmpty())));

                /**
                * Filter for VendorID as '2'
                */
                JavaRDD<String> tripDetailsWithVendor = cleanTripDetails
                                .filter(line -> Integer.valueOf(line.toString().split(",")[0]) == 2);

                /**
                * Filter for tpep_pickup_datetime as '2017-10-01 00:15:30'
                */
```

```
                        JavaRDD<String> details1 = tripDetailsWithVendor
                                        .filter(line -> line.split(",")[1].equals("2017-10-01 00:15:30"));

                        /**
                         * Filter for tpep_dropoff_datetime as '2017-10-01 00:25:11'
                         */
                        JavaRDD<String> details2 = details1.filter(line -> line.split(",")[2].equals("2017-
10-01 00:25:11"));

                        /**
                         * Filter for passenger_count as '1'
                         */
                        JavaRDD<String> details3 = details2.filter(line ->
Integer.valueOf(line.toString().split(",")[3]) == 1);

                        /**
                         * Filter for trip_distance as '2.17'
                         */

                        JavaRDD<String> details4 = details3.filter(line ->
Double.valueOf(line.toString().split(",")[4]) == 2.17);

                        /**
                         * Save result in question1 folder
                         */
                        details4.saveAsTextFile(args[1]);
                } catch (Exception e) {
```

```
                e.printStackTrace();
            } finally {
                try {
                    sc.stop();
                } catch (Exception ex) {
                    ex.printStackTrace();
                }
            }

    }

    /**
     * Function to trace Header
     */
    private static boolean isNotHeader(String line) {
        return !(line.startsWith("VendorID") && line.contains("total_amount"));
    }

}
```

- FilterByRateCodeIdRDD

```
package spark.question2;


import org.apache.log4j.Level;

import org.apache.log4j.Logger;

import org.apache.spark.SparkConf;

import org.apache.spark.api.java.JavaRDD;

import org.apache.spark.api.java.JavaSparkContext;


public class FilterByRateCodeIdRDD {


	public static void main(String[] args) throws Exception {

		JavaSparkContext sc = null;

		try {

			// Logger.getLogger("org").setLevel(Level.ERROR);

			System.setProperty("hadoop.home.dir",
"/user/root/spark_assignment/winUtils/bin/winutils.exe");

			SparkConf conf = new
SparkConf().setAppName("lookUp").setMaster("local[*]");


			/********

			 * Problem Statement *********

			 *

			 * Filter all the records having RatecodeID as 4.

			 */
```

```
                sc = new JavaSparkContext(conf);

                JavaRDD<String> tripDetails = sc.textFile(args[0]);

                /**
                 * To remove Header
                 */
                JavaRDD<String> cleanTripDetails = tripDetails.filter(line -> (isNotHeader(line)
&& !(line.isEmpty())));

                /**
                 * Filter for RatecodeID as 4
                 */
                JavaRDD<String> tripDetailsWithRateCode = cleanTripDetails
                                .filter(line -> Integer.valueOf(line.toString().split(",")[5]) == 4);

                /**
                 * Save result in question2 folder
                 */
                tripDetailsWithRateCode.saveAsTextFile(args[1]);
        } catch (Exception e) {
                e.printStackTrace();
        } finally {
                try {
                        sc.stop();
                } catch (Exception ex) {
```

```
                                ex.printStackTrace();
                        }
                }


        }


        /**
         * Function to trace Header
         */
        private static boolean isNotHeader(String line) {
                return !(line.startsWith("VendorID") && line.contains("total_amount"));
        }
}
```

- **groupedRecordsRDD**

```
package spark.question3;


import org.apache.log4j.Level;

import org.apache.log4j.Logger;

import org.apache.spark.SparkConf;

import org.apache.spark.api.java.JavaPairRDD;

import org.apache.spark.api.java.JavaRDD;

import org.apache.spark.api.java.JavaSparkContext;

import org.apache.spark.api.java.function.Function2;


import scala.Tuple2;


public class groupedRecordsRDD {


	public static void main(String[] args) {


		/**********

		 * Problem Statement *********

		 *

		 * Group By all the records based on payment type and find the count for each

		 * group. Sort the payment types in ascending order of their count.

		 */


		JavaSparkContext sc = null;
```

```
		try {
			// Logger.getLogger("org").setLevel(Level.ERROR);
			System.setProperty("hadoop.home.dir",
"/user/root/spark_assignment/winUtils/bin/winutils.exe");
			SparkConf conf = new
SparkConf().setAppName("groupBy").setMaster("local[2]");

			sc = new JavaSparkContext(conf);

			/**
			 * function used submation of records (Called in reduceByKey)
			 */
			Function2<Integer, Integer, Integer> reduceSumFunc = (accum, n) -> (accum +
n);
			JavaRDD<String> tripDetails = sc.textFile(args[0]);

			/**
			 * To remove Header
			 */
			JavaRDD<String> cleanTripDetails = tripDetails.filter(line -> (isNotHeader(line)
&& !(line.isEmpty())));

			/**
			 * Generate map syntax (payment type value -> 1)
			 */
			JavaPairRDD<Integer, Integer> rddX = cleanTripDetails
```

```
                                        .mapToPair(line -> new Tuple2<Integer,
Integer>(Integer.valueOf(line.toString().split(",")[9]), 1));


                        /**
                         * To perform order By on grouped By records
                         */
                        JavaPairRDD<Integer, Integer> rddY = rddX.reduceByKey(reduceSumFunc);


                        JavaPairRDD<Integer, Integer> rddYSwapMap = rddY
                                        .mapToPair(line -> new Tuple2<Integer, Integer>(line._2,
line._1)).sortByKey(true);


                        JavaPairRDD<Integer, Integer> rddYFinal = rddYSwapMap
                                        .mapToPair(line -> new Tuple2<Integer, Integer>(line._2,
line._1));


                        // Print tuples
                        for (Tuple2<Integer, Integer> element : rddYFinal.collect()) {
                                System.out.println("(" + element._1 + ", " + element._2 + ")");
                        }
                        /**
                         * Save result in question3 folder
                         */
                        rddYFinal.saveAsTextFile(args[1]);
                        // System.out.println(rddY.collect());
                } catch (Exception e) {
                        e.printStackTrace();
```

```
		} finally {
			try {
				sc.stop();
			} catch (Exception ex) {
				ex.printStackTrace();
			}
		}

	}

	/**
	 * Function to trace Header
	 */
	private static boolean isNotHeader(String line) {
		return !(line.startsWith("VendorID") && line.contains("total_amount"));
	}

}
```

Pig Scripts

- ## lookUp

```
register piggybank-0.11.0.jar

 DEFINE CSVExcelStorage org.apache.pig.piggybank.storage.CSVExcelStorage();

file = load '/user/root/spark_assignment/input_dataset/yellow_tripdata*' using
CSVExcelStorage(',', 'NO_MULTILINE', 'NOCHANGE', 'SKIP_INPUT_HEADER') as
(VendorID,tpep_pickup_datetime,tpep_dropoff_datetime,passenger_count,trip_distance,R
atecodeID,store_and_fwd_flag,PULocationID,DOLocationID,payment_type,fare_amount,ext
ra,mta_tax,tip_amount,tolls_amount,improvement_surcharge,total_amount);

 ranked = rank file;

 NoHeader = filter ranked by ($0 > 2);

 records = filter NoHeader by ($1 == 2);

 records1 = filter records by ($2 == '2017-10-01 00:15:30');

 records2 = filter records1 by ($3 == '2017-10-01 00:25:11');

 records3 = filter records2 by ($4 == 1);

 finalRecords = filter records3 by ($5 == 2.17);

 STORE finalRecords INTO '/user/root/pigAssignment2/Q1Output/' USING PigStorage (',');
```

- **FilterByRateCode:**

```
register piggybank-0.11.0.jar


DEFINE CSVExcelStorage org.apache.pig.piggybank.storage.CSVExcelStorage();


 file = load '/user/root/spark_assignment/input_dataset/yellow_tripdata*' using
CSVExcelStorage(',', 'NO_MULTILINE', 'NOCHANGE', 'SKIP_INPUT_HEADER') as
(VendorID,tpep_pickup_datetime,tpep_dropoff_datetime,passenger_count,trip_distance,R
atecodeID,store_and_fwd_flag,PULocationID,DOLocationID,payment_type,fare_amount,ext
ra,mta_tax,tip_amount,tolls_amount,improvement_surcharge,total_amount);


 ranked = rank file;

 NoHeader = filter ranked by ($0 > 2);

 record = filter ranked by ($6 == 4);

 STORE record INTO '/user/root/pigAssignment2/Q2Output/' USING PigStorage (',');
```

- **groupedRecords**

```
register piggybank-0.11.0.jar

DEFINE CSVExcelStorage org.apache.pig.piggybank.storage.CSVExcelStorage();

file = load '/user/root/spark_assignment/input_dataset/yellow_tripdata*' using
CSVExcelStorage(',', 'NO_MULTILINE', 'NOCHANGE', 'SKIP_INPUT_HEADER') as
(VendorID,tpep_pickup_datetime:chararray,tpep_dropoff_datetime,passenger_count,trip_
distance,RatecodeID,store_and_fwd_flag,PULocationID,DOLocationID,payment_type:int,far
e_amount,extra,mta_tax,tip_amount,tolls_amount,improvement_surcharge,total_amount)
;

 ranked = rank file;

 filteredRecords = filter ranked by ($0 > 2) AND $10 is not null;

filteredRecordsByPayment_type = group filteredRecords by $10;

result = foreach filteredRecordsByPayment_type generate group as grp
,COUNT(filteredRecords);

order_by_data = ORDER result BY $1 ASC;

STORE order_by_data INTO '/user/root/pigAssignment2/Q3Output/' USING PigStorage (',');
```