Name: Arun Kumar N

Student ID : 2017CBDE030

Spark Assignment

Spark Codes:

1. **Single Row LookUp:**

```
package com.spark.SparkAssign;


/**
 *  Name: Arun Kumar N
        Student ID : 2017CBDE030
        Spark Assignment
 *
 */
import java.util.Arrays;


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


public class SparkTask1 {


        public void Task1(String path) {


                //Running time which denotes the start time
                long starttime = System.currentTimeMillis();
```

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


			 // Create a Java version of the Spark Context from the configuration


			JavaSparkContext sc = new JavaSparkContext(conf);


			// Load the input data, which is a text file read from the command line

			JavaRDD<String> Datafile = sc.textFile(path);


			//splitting the column using delimeter "," and get VendorID, Pickuptime, Drop time,
Passenger_count and Trip Distance and displaying the value

			Datafile.map(x->x.split(",")).filter(x -> x[0].contains("2") && x[1].contains("2017-10-01
00:15:30") && x[2].contains("2017-10-01 00:25:11") && x[3].contains("1") &&
x[4].contains("2.17")).map(x -> Arrays.toString(x)).foreach(x->System.out.println(x));


			//Close of Spark Context

			sc.close();

			long endtime = System.currentTimeMillis();


			//Running Time which denotes EndTime

			System.out.println(starttime + "-------------" + endtime);


		}


}
```

2. Filter Operation

```
package com.spark.SparkAssign;

/**
 *  Name: Arun Kumar N
        Student ID : 2017CBDE030
        Spark Assignment
 *
 */
import java.util.Arrays;

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

public class SparkTask2 {

        public void Task2(String path)
        {

                //Running time which denotes the start time
        long starttime = System.currentTimeMillis();
        // Define a configuration to use to interact with Spark
        SparkConf conf = new SparkConf().setAppName("Filter
Operation").setMaster("local[*]");

         // Create a Java version of the Spark Context from the configuration

        JavaSparkContext sc = new JavaSparkContext(conf);

        // Load the input data, which is a text file read from the command line
        JavaRDD<String> Datafile = sc.textFile(path);

        //splitting the column using delimeter "," and get x[5] column which is ratecodeID and
check whether it contains 4 and printing the values
```

```
        Datafile.map(x->x.split(",")).filter(x -> x[5].contains("4")).map(x ->
Arrays.toString(x)).foreach(x->System.out.println(x));

        //Close of Spark Context
        sc.close();

        //Running time denotes the end time
        long endtime = System.currentTimeMillis();

        System.out.println(starttime + "-------------" + endtime);
        }

}
```

3. Group By Operation

```
package com.spark.SparkAssign;

/**
 *  Name: Arun Kumar N
        Student ID : 2017CBDE030
        Spark Assignment
 *
 */
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 scala.Tuple2;

public class SparkTask3 {

        public void Task3(String path) {

                //Running time which denotes the start time
                long starttime = System.currentTimeMillis();

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

                 // Create a Java version of the Spark Context from the configuration

                JavaSparkContext sc = new JavaSparkContext(conf);

                // Load the input data, which is a text file read from the command line
                JavaRDD<String> Datafile = sc.textFile(path);

                //Extracting the InputData without Header
```

```
			JavaRDD<String> DatafileWithoutHeader = Datafile.zipWithUniqueId().filter(x -> 
x._2 != 0).map(x -> x._1);



			//Splitting the Data by delimeter "," and loading the column x[9] that is
			JavaRDD<String> PayRDD = DatafileWithoutHeader.map(x->x.split(",")).map(x-
>x[9]);

			JavaPairRDD<String,Integer> pairRDD = PayRDD.mapToPair(
					x -> new Tuple2<String,Integer>(x,1)
					);

			//pairRDD.foreach(x->System.out.println(x._1+":"+x._2));

			//Getting the count of Payment Type

			JavaPairRDD<String, Integer> countRDD = pairRDD.reduceByKey(
					(x,y) -> x+y
					);

			//To sort in ascending order, change the key to value and value to key
			JavaPairRDD<Integer,String> sortRDD =  countRDD.mapToPair( x -> new 
Tuple2<Integer,String>(x._2,x._1));

			//Sorting in Ascending Order
			sortRDD.sortByKey().foreach(x->System.out.println(x._2+":"+x._1));


//			countRDD.foreach(x->System.out.println(x._1+":"+x._2));

			//Running time which denotes end time
			long endtime = System.currentTimeMillis();

			//Close of Spark Context
			sc.close();
			System.out.println(starttime +"-------------" + endtime );
		}

}
```

Pig Codes:

1. **Single Row LookUp:**

//Loading data with Delimeter ‘,’ and assigning datatypes

i. data = LOAD 'trip_yellow_taxi.data' using PigStorage(',') AS (VendorID:int, tpep_pickup_datetime:chararray, tpep_dropoff_datetime:chararray, passenger_count:int, trip_distance:float, RatecodeID:int, store_and_fwd_flag:chararray, PULocationID:int, DOLocationID:int, payment_type:int, fare_amount:float, extra:float, mta_tax:float, tip_amount:float, tolls_amount:float, improvement_surcharge:float, total_amount:float);

//Filtering the data using specific datatypes

ii. rec = FILTER data BY ((VendorID == 2) AND (tpep_dropoff_datetime=='2017-10-01 00:25:11') AND (tpep_pickup_datetime=='2017-10-01 00:15:30') AND (passenger_count==1) AND (trip_distance==2.17));

//Record will get store into location

iii. STORE rec into '/user/cloudera/workspace/log/finalfirstproboutput1.out';

2. **Filter Operation**

   i. data = LOAD 'trip_yellow_taxi.data' using PigStorage(',') AS (VendorID:int, tpep_pickup_datetime:chararray, tpep_dropoff_datetime:chararray, passenger_count:int, trip_distance:float, RatecodeID:int, store_and_fwd_flag:chararray, PULocationID:int, DOLocationID:int, payment_type:int, fare_amount:float, extra:float, mta_tax:float, tip_amount:float, tolls_amount:float, improvement_surcharge:float, total_amount:float);

   ii. rec = FILTER data BY RatecodeID==4;

   iii. STORE rec into '/user/cloudera/workspace/log/secondproboutput1.out';

3. **Group By Operation**

i. data = LOAD 'trip_yellow_taxi.data' using PigStorage(',') AS (VendorID:int, tpep_pickup_datetime:chararray, tpep_dropoff_datetime:chararray, passenger_count:int, trip_distance:float, RatecodeID:int, store_and_fwd_flag:chararray, PULocationID:int, DOLocationID:int, payment_type:int, fare_amount:float, extra:float, mta_tax:float, tip_amount:float, tolls_amount:float, improvement_surcharge:float, total_amount:float);

ii. group_rec = GROUP data BY payment_type;

iii. final_count = FOREACH group_rec GENERATE group, COUNT (data);

iv. sort_count = ORDER final_count BY $1;

v. STORE sort_count into '/user/cloudera/workspace/log/thirdproboutput2.out';