9/9/2018

PIG Scripts & JAVA Codes

SPARK Project-1

Name: RUKSANA BHANU SHEIK

PIG SCRIPTS

PIG SCRIPT1: LOOKUP

```
REGISTER piggybank.jar;

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

INPUT_DATA = LOAD '/user/ec2-user/spark_assignment/input/yellow_tripdata*' USING
org.apache.pig.piggybank.storage.CSVExcelStorage(',', 'NO_MULTILINE', 'UNIX',
'SKIP_INPUT_HEADER')  AS
(VendorID:int,tpep_pickup_datetime:chararray,tpep_dropoff_datetime:chararray,passenger_count:int
,

trip_distance:double,RatecodeID:int,store_and_fwd_flag:chararray,PULocationID:int,DOLocationID:in
t,

payment_type:int,fare_amount:float,extra:float,mta_tax:float,tip_amount:float,tolls_amount:float,im
provement_surcharge:float,total_amount:float);

INPUT_DATA_CLEAN = FILTER INPUT_DATA BY NOT (VendorID IS NULL);

OUTPUT_DATA = filter INPUT_DATA_CLEAN by VendorID ==2 and

ToDate(tpep_pickup_datetime,'yyyy-MM-dd HH:mm:ss') == ToDate('2017-10-01 00:15:30','yyyy-MM-
dd HH:mm:ss') and

ToDate(tpep_dropoff_datetime,'yyyy-MM-dd HH:mm:ss') == ToDate('2017-10-01 00:25:11','yyyy-MM-
dd HH:mm:ss') and

passenger_count==1 and

trip_distance==2.17;

DUMP OUTPUT_DATA;

STORE OUTPUT_DATA INTO '/user/ec2-user/spark_assignment/output/pig1_single_row_lookup' using
PigStorage(',');
```

PIG SCRIPT2: FILTER

```
INPUT_DATA = LOAD '/user/ec2-user/spark_assignment/input/' USING PigStorage(',') AS
(VendorID:int,tpep_pickup_datetime:chararray,tpep_dropoff_datetime:chararray,passenger_count:int
,
trip_distance:double,RatecodeID:int,store_and_fwd_flag:chararray,PULocationID:int,DOLocationID:in
t,
payment_type:int,fare_amount:float,extra:float,mta_tax:float,tip_amount:float,tolls_amount:float,im
provement_surcharge:float,total_amount:float);

OUTPUT_DATA = FILTER INPUT_DATA by RatecodeID == 4;

STORE OUTPUT_DATA into '/user/ec2-user/spark_assignment/output/pig2_filter' using PigStorage(',');
```

PIG SCRIPT3: GROUPBY

```
REGISTER piggybank.jar;

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

INPUT_DATA = LOAD '/user/ec2-user/spark_assignment/input/yellow_tripdata*' USING
org.apache.pig.piggybank.storage.CSVExcelStorage(',', 'NO_MULTILINE', 'UNIX',
'SKIP_INPUT_HEADER')  AS
(VendorID:int,tpep_pickup_datetime:chararray,tpep_dropoff_datetime:chararray,passenger_count:int
,
trip_distance:double,RatecodeID:int,store_and_fwd_flag:chararray,PULocationID:int,DOLocationID:in
t,
payment_type:int,fare_amount:float,extra:float,mta_tax:float,tip_amount:float,tolls_amount:float,im
provement_surcharge:float,total_amount:float);

INPUT_DATA_CLEAN = FILTER INPUT_DATA BY NOT (VendorID IS NULL);

GROUP_INPUT = GROUP INPUT_DATA_CLEAN BY payment_type;

PYMTTYPE_COUNT = FOREACH GROUP_INPUT GENERATE group,COUNT(INPUT_DATA_CLEAN)
AS CNT;

SORT_PYMTTYPE_COUNT = ORDER PYMTTYPE_COUNT BY CNT ASC;

OUTPUT_DATA= foreach SORT_PYMTTYPE_COUNT generate $0,$1;

STORE OUTPUT_DATA INTO '/user/ec2-user/spark_assignment/output/pig3_groupby_orderby' using
PigStorage(',');
```

SPARK JAVA CODE

SPARK PROGRAM1: LOOKUP

```
package sparkAssignment.LookUp;


import java.text.DateFormat;

import java.text.SimpleDateFormat;

import java.util.List;

import org.apache.spark.SparkConf;

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

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

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


public class SingleLookUp {


	public static void main(String[] args) {


		SparkConf conf = new SparkConf().setAppName("SingleLookUp");

		@SuppressWarnings("resource")

		JavaSparkContext sc = new JavaSparkContext(conf);


		JavaRDD<String> inputDataRdd = sc.textFile(args[o]);


		//Removing Headers
```

```
		JavaRDD<String> inputDataNoHeaderRdd = inputDataRdd.filter(x -
> !(x.contains("VendorID")));


		//Filtering as per the paramters given in problem statement
		JavaRDD<String> lookUpRdd = inputDataNoHeaderRdd.filter(new Function<String,
Boolean>() {
			private static final long serialVersionUID = 1L;

			@Override
			public Boolean call(String record) throws Exception {
				if (record.length() > 0 && record.charAt(0) == ',')
					return false;
				else {
					String vals[] = record.split(",");
					DateFormat sdf = new SimpleDateFormat("yyyy-MM-dd
HH:mm:ss");

					if ((vals[0]).equals("2")
						&& (sdf.parse(vals[1])).equals(sdf.parse("2017-10-01
00:15:30"))
						&& (sdf.parse(vals[2])).equals(sdf.parse("2017-10-01
00:25:11"))
						&& (vals[3]).equals("1")
						&& (vals[4]).equals("2.17"))
						return true;
					else
						return false;
				}
```

```
                }
        }

        );

        List<String> inputDataNoHeaderRddList = lookUpRdd.collect();

        //Saving to the file
        lookUpRdd.saveAsTextFile(args[1]);

        for (String i : inputDataNoHeaderRddList) {
                System.out.println(i);

        }
    }

}
```

SPARK ROGRAM2: FILTER

```
package sparkAssignment.Filter;

import java.util.List;

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

```
import org.apache.spark.api.java.JavaSparkContext;
import org.apache.spark.api.java.function.Function;

public class Filter {

	public static void main(String[] args) {
		SparkConf conf = new SparkConf().setAppName("Filter");
		JavaSparkContext sc = new JavaSparkContext(conf);
		JavaRDD<String> inputDataRdd = sc.textFile(args[0]);

		//Removing Headers
		JavaRDD<String> inputDataNoHeaderRdd = inputDataRdd.filter(x -
> !(x.contains("VendorID")));

		//Filtering with ratecode ==4
		JavaRDD<String> filterRdd = inputDataNoHeaderRdd.filter(new Function<String,
Boolean>() {
			private static final long serialVersionUID = 1L;

			@Override
			public Boolean call(String v1) throws Exception {
				if (v1.length() >= 5 && v1.charAt(0) == ',')
					return false;
				else {
					String v11[] = v1.split(",");
					return ((v1.length() >= 5) ? (Integer.parseInt(v11[5]) == 4 ? true :
false) : false);
```

```
                    }
                }

            });

            List<String> finalList = filterRdd.collect();

            //Saving to text file.
            filterRdd.saveAsTextFile(args[1]);

            sc.close();
        }

}
```

SPARK PROGRAM3: GROUPBY

```
package sparkAssignment.GroupBy;

import java.util.List;
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.Function;
import org.apache.spark.api.java.function.PairFunction;
import scala.Tuple2;
```

```
public class GroupBy {


	public static void main(String[] args) {

		SparkConf conf = new SparkConf().setAppName("count");

		JavaSparkContext sc = new JavaSparkContext(conf);

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

		//JavaRDD<String> inputDataRdd =
sc.textFile("C:\\USERS\\RUKSANA\\Desktop\\SparkRelated\\yellow_tripdata_2017-07.csv");


		//Remove Header

		JavaRDD<String> inputDataNoHeaderRdd = inputDataRdd.filter(x -
> !(x.contains("VendorID")));


		//Paired Rdd with (payment type,1) as a pair. in case of any Exception (0,1) will be a pair

		JavaPairRDD<Integer, Long> pairRdd = inputDataNoHeaderRdd.mapToPair(new
PairFunction<String, Integer, Long>() {

			private static final long serialVersionUID = 1L;

			@Override

			public Tuple2<Integer, Long> call(String record) throws Exception {

				if (!(record.length() > 0 && record.charAt(0) == ',')) {

					String[] record_values = record.split(",");

					if (record_values.length >= 9)

						return ((record_values.length >= 9

								? new Tuple2<Integer,
Long>(Integer.parseInt(record_values[9]), (long) 1)

								: new Tuple2<Integer, Long>(0, (long)
1)));
```

```
                }
                return new Tuple2<Integer, Long>(0, (long) 1);
            }
        });

        //Reducing the paired RDD using reduce by key
        JavaPairRDD<Integer, Long> ReduceCntRdd = pairRdd.reduceByKey((x, y) -> x + y);

        //Filtering out Exceptional pairs such as (0,1)
        JavaPairRDD<Integer, Long> ReduceCntRdd1 = ReduceCntRdd.filter(new
Function<Tuple2<Integer, Long>, Boolean>() {
            private static final long serialVersionUID = 1L;
            @Override
            public Boolean call(Tuple2<Integer, Long> v1) throws Exception {
                if (v1._1 == 0)
                    return false;
                else
                    return true;
            }
        });

        //Sorting
        List<Tuple2<Integer, Long>> sortedList = ReduceCntRdd1.top((int)
ReduceCntRdd1.count(), new TupleSorter());

        JavaRDD<Tuple2<Integer, Long>> finalRdd = sc.parallelize(sortedList, 1);
```

```
		List<Tuple2<Integer, Long>> finalList = finalRdd.collect();

		//Saving to file
		finalRdd.saveAsTextFile(args[1]);

		for (Tuple2<Integer, Long> t : finalList) {
			System.out.println(t);
		}

		sc.close();

	}}
package sparkAssignment.GroupBy;

import java.util.Comparator;
import java.io.Serializable;

import scala.Tuple2;

public class TupleSorter implements Comparator<Tuple2<Integer,Long>>, Serializable {

	private static final long serialVersionUID = 1L;

	@Override
	public int compare(Tuple2<Integer,Long> o1, Tuple2<Integer,Long> o2) {
```

```
                if (o1._1 > o2._1)
                        return 1;
                else if (o1._1 < o2._1)
                        return -1;
                else
                        return 0;
        }

}
```