**Q1. MapReduce Problem Statement Here, we have chosen the stock market dataset on which we have performed map-reduce operations. Following is the structure of the data. Kindly find the solutions to the questions below.**

1) MAP REDUCE ALL TIME HIGH :

[bigdatamind43816@ip-10-1-1-204 ~]$ jar tvf myjar1.jar

   25 Mon Jun 20 15:08:00 UTC 2022 META-INF/MANIFEST.MF

 1353 Mon Jun 20 14:50:44 UTC 2022 exam/AllTimeHigh$MapClass.class

 1422 Mon Jun 20 14:50:44 UTC 2022 exam/AllTimeHigh$ReduceClass.class

  729 Mon Jun 20 14:50:44 UTC 2022 exam/AllTimeHigh.class

[bigdatamind43836@ip-10-1-1-204 ~]$ hadoop jar exam/myjar1.jar AllTimeHigh NYSE.csv cdac/eout

JAR does not exist or is not a normal file: /home/bigdatamind43816/exam/myjar1.jar

**import** java.io.\*;

**import** org.apache.hadoop.io.Text;

**import** org.apache.hadoop.io.LongWritable;

**import** org.apache.hadoop.io.DoubleWritable;

**import** org.apache.hadoop.mapreduce.Job;

**import** org.apache.hadoop.mapreduce.Mapper;

**import** org.apache.hadoop.mapreduce.Reducer;

**import** org.apache.hadoop.conf.\*;

**import** org.apache.hadoop.fs.\*;

**import** org.apache.hadoop.mapreduce.lib.input.\*;

**import** org.apache.hadoop.mapreduce.lib.output.\*;

**public** **class** AllTimeHigh {

**public** **static** **class** MapClass **extends** Mapper<LongWritable,Text,Text,DoubleWritable>

{

**private** Text stock\_id = **new** Text();

**private** DoubleWritable High = **new** DoubleWritable();

**public** **void** map(LongWritable key, Text value, Context context)

{

**try**{

String[] str = value.toString().split(",");

**double** high = Double.*parseDouble*(str[4]);

stock\_id.set(str[1]);

High.set(high);

context.write(stock\_id, High);

}

**catch**(Exception e)

{

System.***out***.println(e.getMessage());

}

}

}

**public** **static** **class** ReduceClass **extends** Reducer<Text,DoubleWritable,Text,DoubleWritable>

{

**private** DoubleWritable result = **new** DoubleWritable();

**public** **void** reduce(Text key, Iterable<DoubleWritable> values,Context context) **throws** IOException, InterruptedException {

**double** maxValue=0;

**double** temp\_val=0;

**for** (DoubleWritable value : values) {

temp\_val = value.get();

**if** (temp\_val > maxValue) {

maxValue = temp\_val;

}

}

result.set(maxValue);

context.write(key, result);

//context.write(key, new LongWritable(sum));

}

}

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

Configuration conf = **new** Configuration();

Job job = Job.*getInstance*(conf, "Highest Price for each stock");

job.setJarByClass(AllTimeHigh.**class**);

job.setMapperClass(MapClass.**class**);

job.setReducerClass(ReduceClass.**class**);

job.setNumReduceTasks(1);

job.setOutputKeyClass(Text.**class**);

job.setOutputValueClass(DoubleWritable.**class**);

FileInputFormat.*addInputPath*(job, **new** Path(args[0]));

FileOutputFormat.*setOutputPath*(job, **new** Path(args[1]));

System.*exit*(job.waitForCompletion(**true**) ? 0 : 1);

}

}

**Question 2 :**

**Find all time High price for each stock**

hive> use traning050;

hive> show tables;

airlines

airport

customer

employee

employee\_header

nyse

result1

routes

stkvol

table\_prac

testing

tran

txn\_orc

txn\_parquet

txnrecords

txnrecsbycat

txnrecsbycat3

txnrecsbycat4

1. **Write a program to find the count of customers for each profession.**

**Query:**

hive> select profession,count(\*) as customer\_count from customer group by profession order by customer\_count;

**Output**:

Social Worker   1

Writer  101

Artist  175

Environmental scientist 176

Carpenter       181

Dancer  185

Therapist       187

Economist       189

Real estate agent       191

Electrical engineer     192

Nurse   192

Civil engineer  193

Automotive mechanic     193

Psychologist    194

Electrician     194

Agricultural and food scientist 195

Athlete 196

Statistician    196

Judge   196

Doctor  197

Financial analyst       198

Accountant      199

Reporter        200

Secretary       200

Coach   201

Physicist       201

Farmer  201

Actor   202

Architect       203

Computer hardware engineer      204

Teacher 204

Engineering technician  204

Designer        205

Musician        205

Childcare worker        207

Veterinarian    208

Chemist 209

Police officer  210

Recreation and fitness worker   210

Lawyer  212

Social worker   212

Pilot   212

Human resources assistant       212

Pharmacist      213

Computer software engineer      216

Firefighter     217

Librarian       218

Loan officer    221

Photographer    222

Computer support specialist     222

Politician      228

Time taken: 80.14 seconds, Fetched: 51 row(s)

1. **Write a program to find the top 10 products sales wise**

**Query**

hive> select product,round(sum(product),2) as amt from txnrecords group by product order by amt desc limit 10;

Output:

Yoga&Pilates  47804.94

Swing Sets      47204.14

Lawn Games      46828.44

Golf    46577.68

Cardio Machine Accessories      46485.54

Exercise Balls  45143.84

Weightlifting Belts     45111.68

Mahjong 44995.2

Basketball      44954.68

Beach Volleyball        44890.67

Time taken: 93.233 seconds, Fetched: 10 row(s)

1. **Write a program to create partiioned table on category.**

**Query:**

set hive.exec.dynamic.partition.mode=nonstrict; set hive.exec.dynamic.partition=true;

hive>

hive> create table txnrecsbycat(txnno INT, txndate STRING, custno INT, amount DOUBLE,

   > product STRING, city STRING, state STRING, spendby STRING)

   > partitioned by (category STRING)

   > row format delimited

   > fields terminated by ','

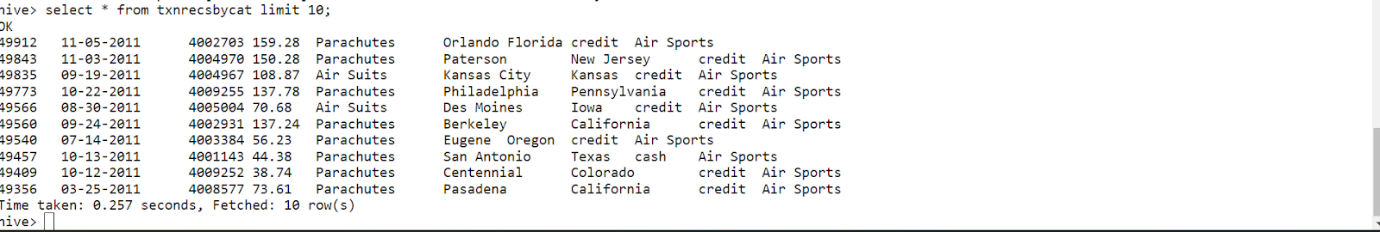
   > stored as textfile;

INSERT OVERWRITE TABLE txnrecsbycat  PARTITION(category) select txn.txnno, txn.txndate,txn.custno, txn.amount,txn.product,txn.city,txn.state, txn.spendby, txn.category

from txnrecords txn

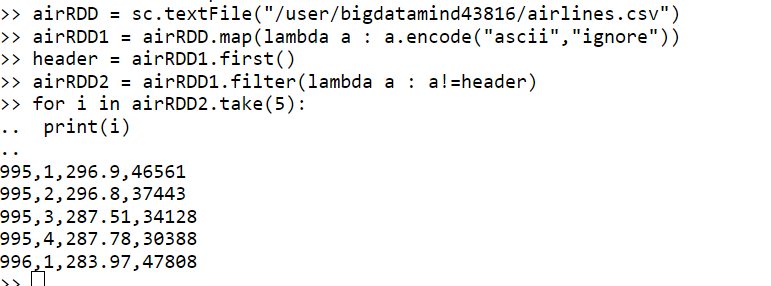
DISTRIBUTE By category;

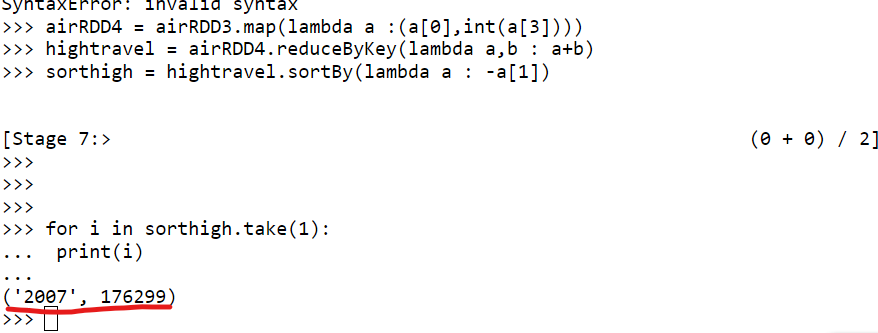
**Output:**



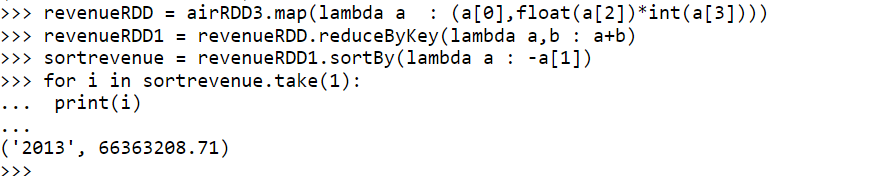
**QUESTION 3**

**1) What was the highest number of people travelled in which year**

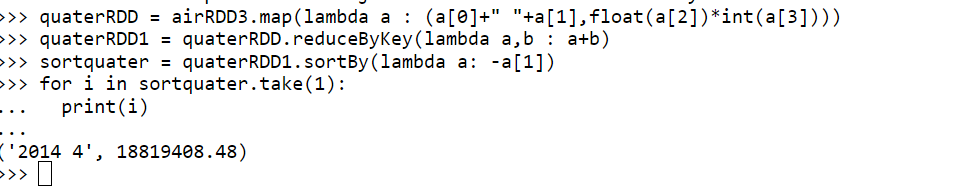




**2) Identifying the highest revenue generation for which year**

****

**3) Identifying the highest revenue generation for which year and quarter (Common group)**

****