**Q1.**

**MapReduce**

**Problem Statement [10 marks]**

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.

Data Structure

1. Exchange Name

2 Stock symbol

3. Transaction date

4. Opening price of the stock

5. Intra day high price of the stock

6. Intra day low price of the stock

7. Closing price of the stock

8. Total Volume of the stock on the particular day

9. Adjustment Closing price of the stock

Field Separator – comma

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

package exam;

import java.io.IOException;

import java.util.StringTokenizer;

import org.apache.hadoop.conf.Configuration;

import org.apache.hadoop.fs.Path;

import org.apache.hadoop.io.IntWritable;

import org.apache.hadoop.io.LongWritable;

import org.apache.hadoop.io.Text;

import org.apache.hadoop.mapreduce.Job;

import org.apache.hadoop.mapreduce.Mapper;

import org.apache.hadoop.mapreduce.Reducer;

import org.apache.hadoop.mapreduce.lib.input.FileInputFormat;

import org.apache.hadoop.mapreduce.lib.output.FileOutputFormat;

public class AllTimeHigh1{

public static class mapclass extends Mapper<LongWritable, Text, Text, IntWritable>{

}

public static class reducerclass extends Reducer<Text,IntWritable,Text,IntWritable> {

}

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

Configuration conf = new Configuration();

Job job = Job.getInstance(conf, " ");

job.setJarByClass(AllTimeHigh1.class);

job.setMapperClass(mapclass.class);

job.setReducerClass(reducerclass.class);

job.setNumReduceTasks(1);

job.setMapOutputKeyClass(Text.class);

job.setMapOutputValueClass(IntWritable.class);

job.setOutputKeyClass(Text.class);

job.setOutputValueClass(LongWritable.class);

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

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

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

}

}

}

**Upload jar on ftp AllTimeHigh1.jar**

**Jar tvf AllTimeHigh1.jar**

**hadoop jar AllTimeHigh1.jar exam/AllTimeHigh1 user/bigcdac432537/exam/NYSE.csv user/bigcdac432537/exam/mapreduce**

**Question 2 : [15 marks]**

**Hive**

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

Please find the customer data set.

cust id

firstname

lastname

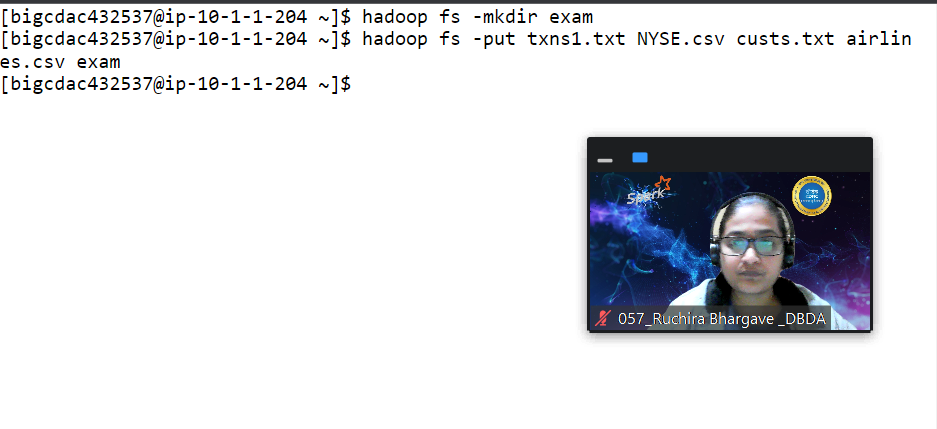
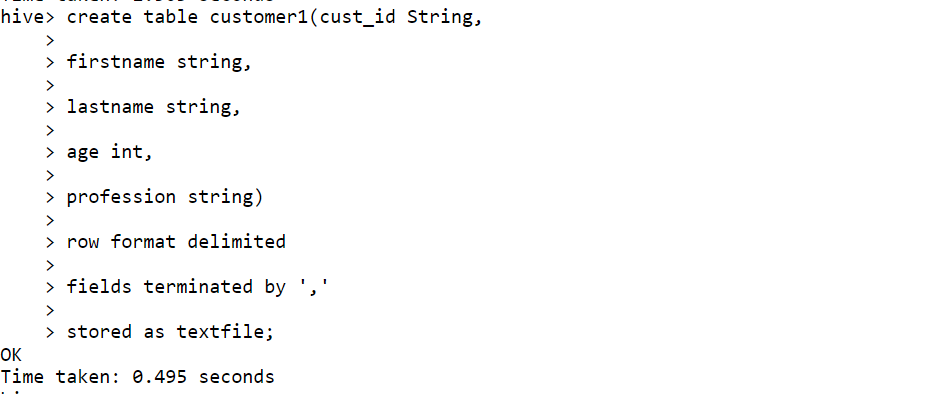
age

Profession

**1. Created directory and uploaded files in it using : mkdir and -put commands**

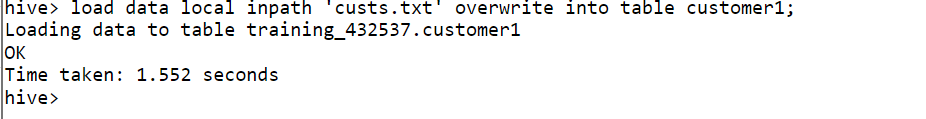
hadoop fs -mkdir exam

hadoop fs -put txns1.txt NYSE.csv custs.txt airlines.csv exam

 ****

**Customer1 table is created:**

**load data local inpath 'custs.txt' overwrite into table customer1;**

****

**query:**

**select profession, count(cust\_id) from customer1 group by profession limit 10;**

****

**2) Write a program to find the top 10 products sales wise**

Please find the sales data set.

txn id

txn date

cust id

amount

category

product

city

state

spendby

**Table created:**

**create table txn1(txn\_id int, txn\_date string, cust\_id int, amount double,**

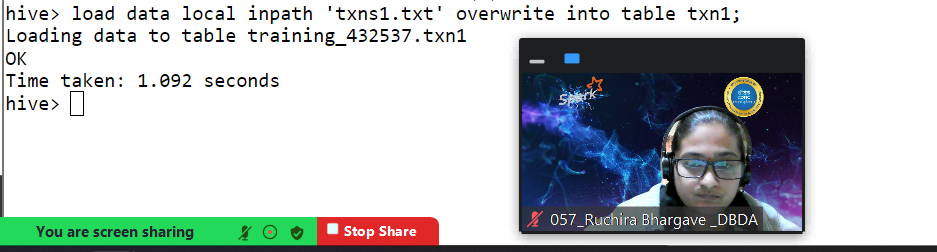
**category string, product string, city string, state string, spendby string)**

**row format delimited**

**fields terminated by ','**

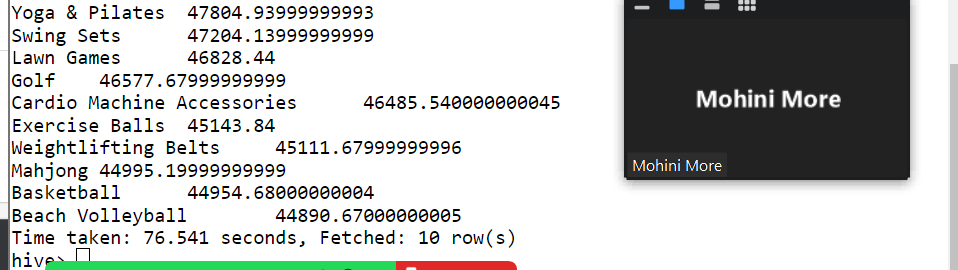
**stored as textfile;**

**load data local inpath 'txns1.txt' overwrite into table txn1;**

****

**Query:**

select product, sum(amount) as total from txn1 group by product order by total desc limit 10;

****

**3) Write a program to create partitioned table on category**

**Create partitioned table:**

create table txnByPartition(

txn\_id int,

txn\_date string,

cust\_id int,

amount double,

product string,

city string,

state string,

spendby string)

partitioned by (category string)

row format delimited

fields terminated by ','

stored as textfile;

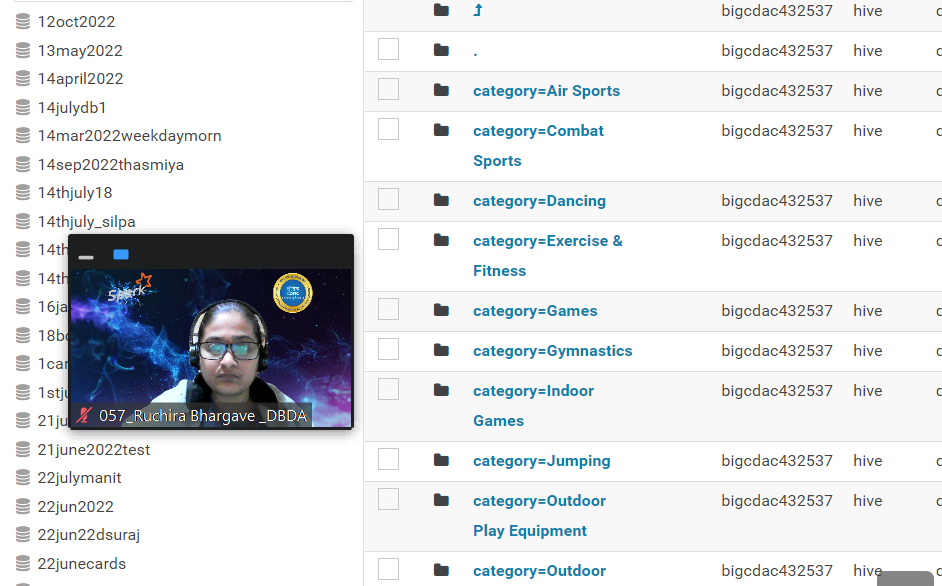
**Insert data:**

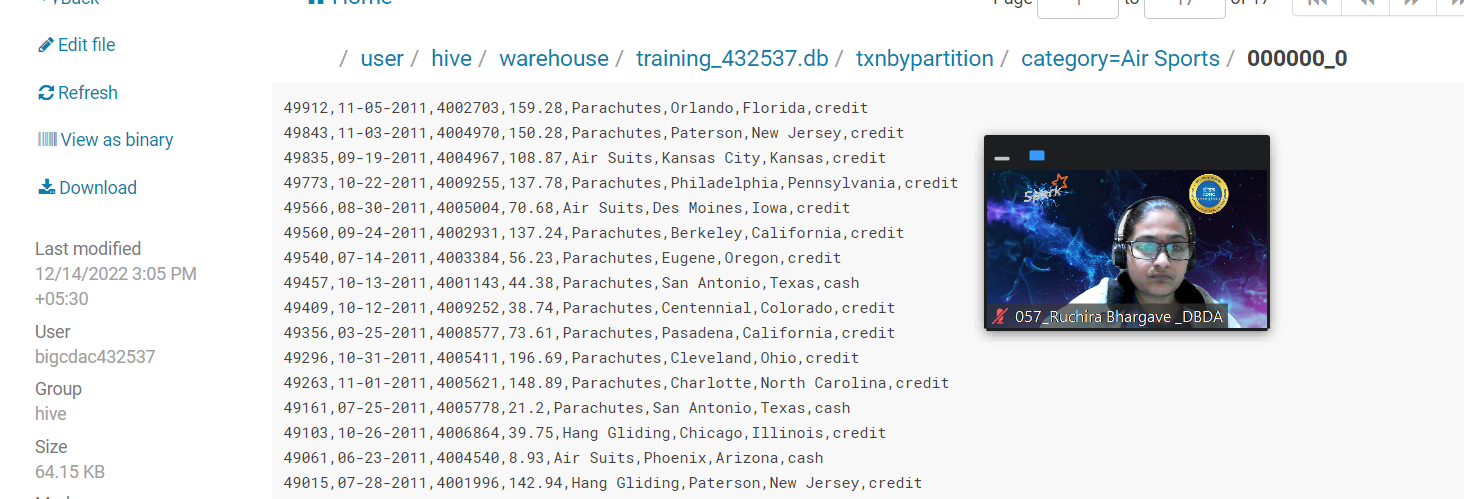
INSERT OVERWRITE TABLE txnByPartition PARTITION(category)

select txn.txn\_id, txn.txn\_date,txn.cust\_id, txn.amount,txn.product,

txn.city,txn.state, txn.spendby,

txn.category from txn1 txn DISTRIBUTE By category;





**QUESTION 3 [15 marks]**

**PySpark**

Please find the AIRLINES data set

Year

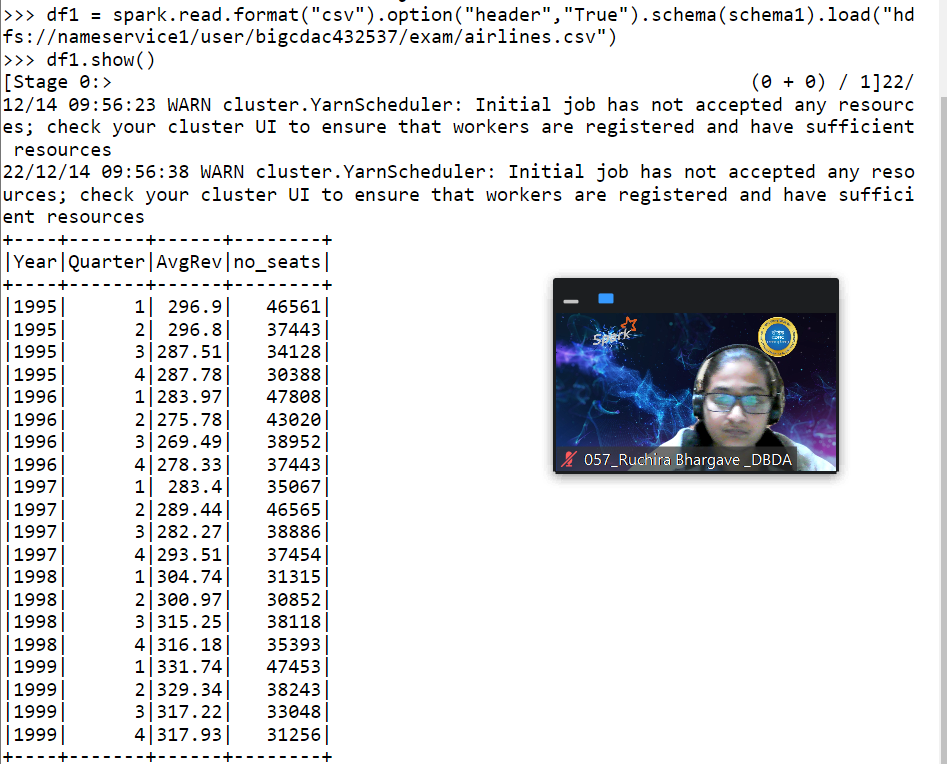
Quarter

Average revenue per seat

Total number of booked seats

**Commands to create dataframe and scheme:**

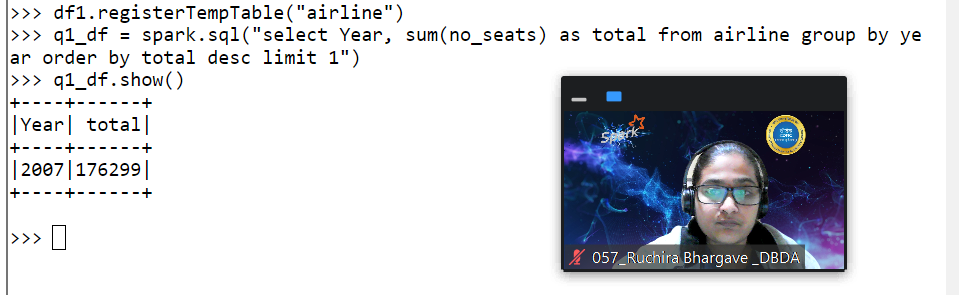
1. from pyspark.sql.types import StructType, IntegerType, DoubleType, StringType, LongType
2. schema1 = StructType().add("Year",StringType(),True).add("Quarter",IntegerType(),True).add("AvgRev",DoubleType(),True).add("no\_seats",LongType(),True)
3. df1 = spark.read.format("csv").option("header","True").schema(schema1).load("hdfs://nameservice1/user/bigcdac432537/exam/airlines.csv")
4. df1.show()
5. df1.registerTempTable("airline")



**1) What was the highest number of people who traveled in which year?**

q1\_df = spark.sql("select Year, sum(no\_seats) as total from airline group by year order by total desc limit 1")

q1\_df.show()

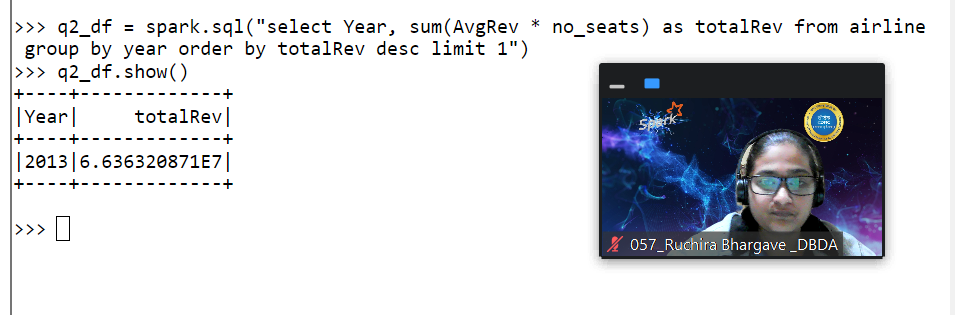
****

**Ans: highest number of passengers 176299 in 2017.**

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

q2\_df = spark.sql("select Year, sum(AvgRev \* no\_seats) as totalRev from airline group by year order by totalRev desc limit 1")

q2\_show()

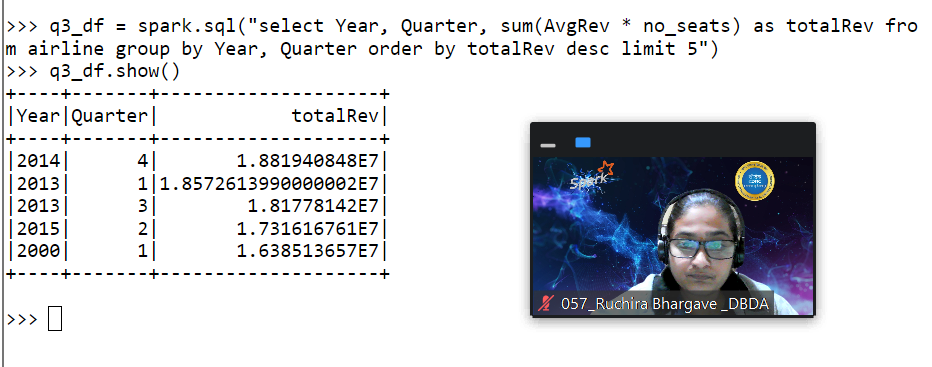
****

**Ans: highest revenue 6.636320871E7 was generated in the year 2013**

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

q3\_df = spark.sql("select Year, Quarter, sum(AvgRev \* no\_seats) as totalRev from airline group by Year, Quarter order by totalRev desc limit 5")

q3\_df.show()



**Write output on hdfs location:**

q3\_df.write.csv("hdfs://nameservice1/user/bigcdac432537/exam/pysparkop")

