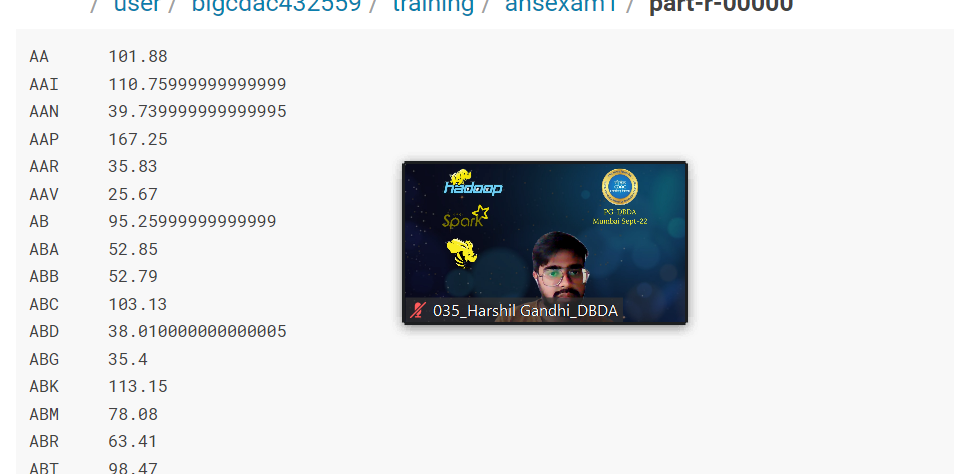
**Q1 mapreduce**

[bigcdac432559@ip-10-1-1-204 ~]$ hadoop fs -put NYSE\_exam.csv training/

[bigcdac432559@ip-10-1-1-204 ~]$ hadoop jar exammr.jar Exam training/NYSE\_exam.csv training/ansexam1



Program:

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.DoubleWritable;

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 Exam {

public static class Mappered extends Mapper<LongWritable, Text, Text, DoubleWritable>{

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

{

try

{

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

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

context.write(new Text(str[1]), new DoubleWritable(highs));

}

catch(Exception e)

{

System.out.println(e);

}

}

}

public static class Reducered extends Reducer<Text,DoubleWritable,Text,DoubleWritable> {

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

{

double max = 0.00;

for(DoubleWritable val : value)

{

if(val.get()>max)

{

max += val.get();

}

}

context.write(new Text(key),new DoubleWritable(max));

}

}

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

Configuration conf = new Configuration();

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

job.setJarByClass(Exam.class);

job.setMapperClass(Mappered.class);

job.setReducerClass(Reducered.class);

job.setNumReduceTasks(1);

job.setMapOutputKeyClass(Text.class);

job.setMapOutputValueClass(DoubleWritable.class);

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);

}

}

Q2

create table customer\_exam

(

customer\_id INT,

firstname STRING,

lastname STRING,

age INT,

profession STRING

)

row format delimited

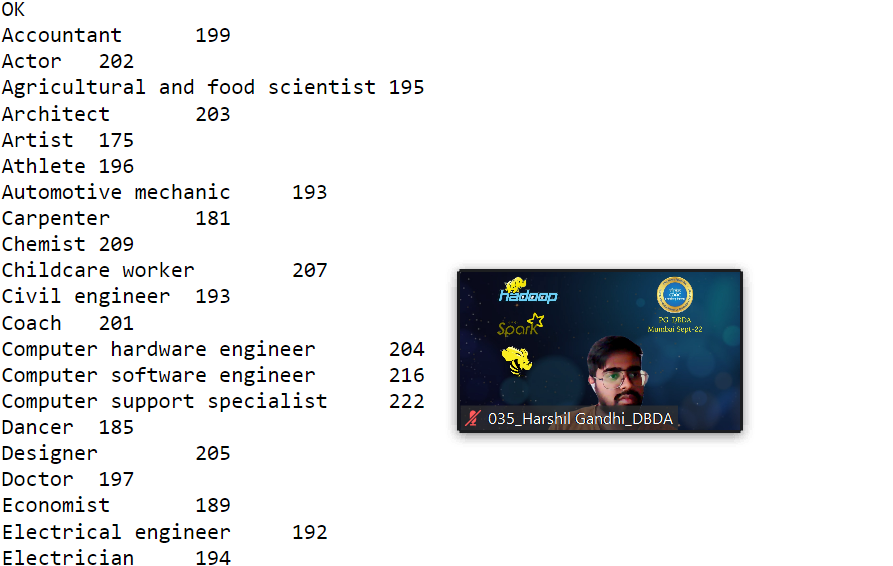
fields terminated by ','

stored as textfile;

load data local inpath 'custs\_exam.txt' into table customer\_exam;

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

**select profession,count(customer\_id) from customer\_exam group by profession;**



HIVE Q2

create table txn\_exam1

(

txn\_id DOUBLE,

Txn\_date STRING,

cust\_id INT,

amount DOUBLE,

category STRING,

product STRING,

city STRING,

state STRING,

spendby STRING

)

row format delimited

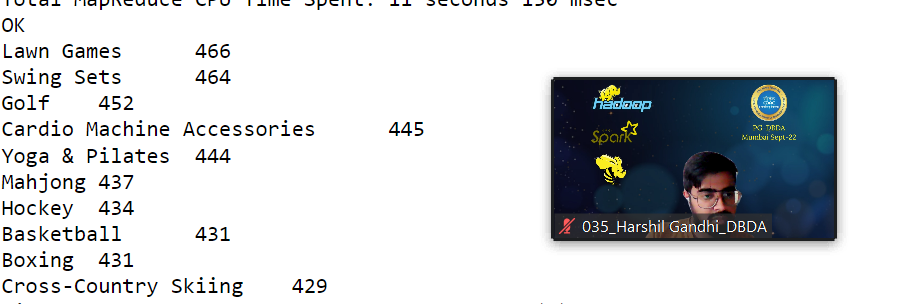
fields terminated by ','

stored as textfile;

load data local inpath 'txns1\_exam.txt' into table txn\_exam1;

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

Select product,count(amount) as amt from txn\_exam1 group by product order by amt desc limit 10;



Q3)**) Write a program to create partiioned table on category**

create table txn\_exampart

(

txn\_id DOUBLE,

Txn\_date STRING,

cust\_id INT,

amount DOUBLE,

product STRING,

city STRING,

state STRING,

spendby STRING

)partitioned by (category STRING)

row format delimited

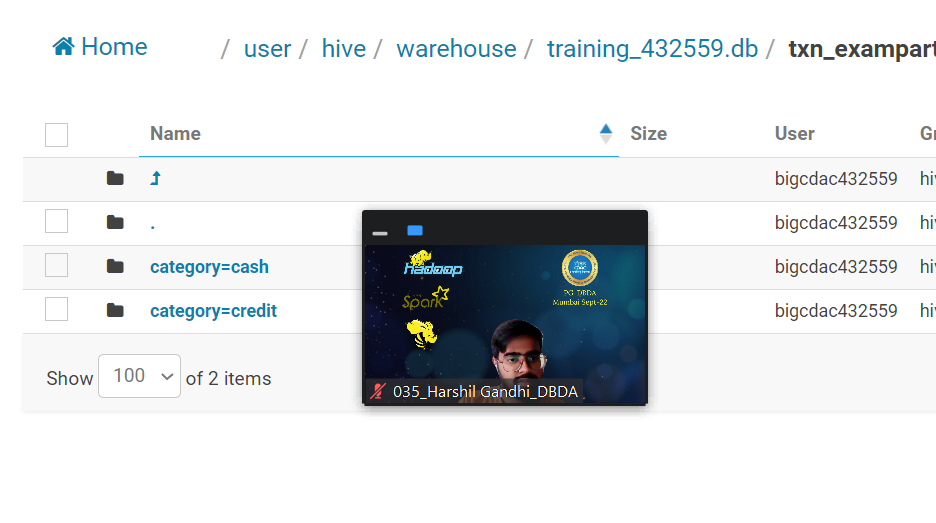
fields terminated by ','

stored as textfile;

hive (training\_432559)> set hive.exec.dynamic.partition.mode = nonstrict;

hive (training\_432559)> set hive.exec.dynamic.partition = true;

Insert overwrite table txn\_exampart partition(category) select t.txn\_id,t.txn\_date,t.cust\_id,t.amount,t.category,t.product,t.city,t.state,t.spendby from txn\_exam1 t;



**SPARK**

**from pyspark.sql.types import StringType,IntegerType,DoubleType,StructType**

Year

Quarter

Average revenue per seat

Total number of booked seats

**exam = StructType().add(“year”,IntegerType(),False).add(“quarter”,IntegerType(),False).add(“avg\_revenue”,DoubleType(),False).add(“total\_seats”,IntegerType(),False)**

**resultexam = spark.read.format("csv").option("header",True).schema(examschema).load("hdfs://nameservice1/user/bigcdac432559/training/examl")**

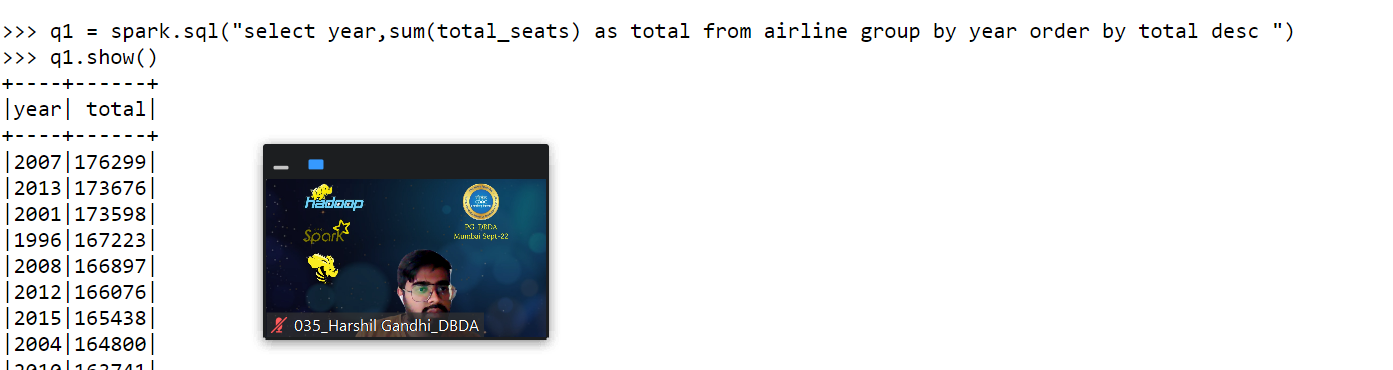
**resultexam.registerTempTable("airline")**

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

**Year?**

q1 = spark.sql("select year,sum(total\_seats) as total from airline group by year order by total desc ")

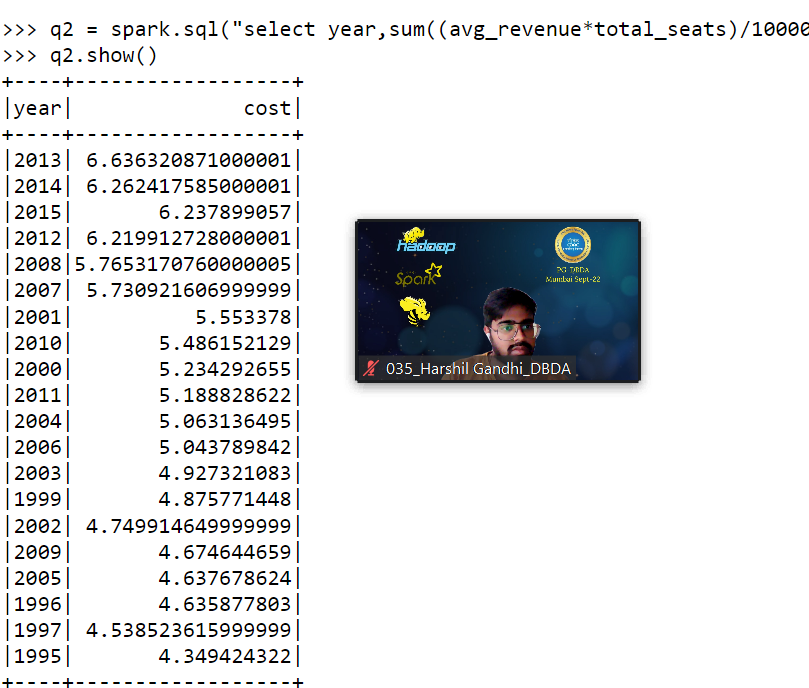
2007 has highest no of booked seats so highest number of people is travelled in 2007



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

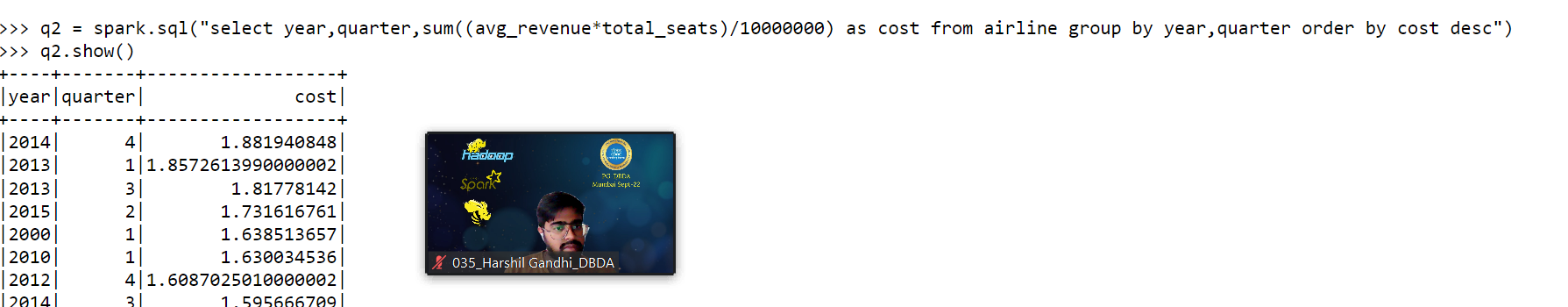
**q2 = spark.sql("select year,sum((avg\_revenue\*total\_seats)/10000000) as cost from airline group by year order by cost desc")**

**2013 has the highest revenue generation**

****

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

**q2 = spark.sql("select year,quarter,sum((avg\_revenue\*total\_seats)/10000000) as cost from airline group by year,quarter order by cost desc")**

****