Qyestion 2

Hive

Commands –

create table customer(

custno INT,

firstname STRING,

lastname STRING,

age INT,

profession STRING)

row format delimited

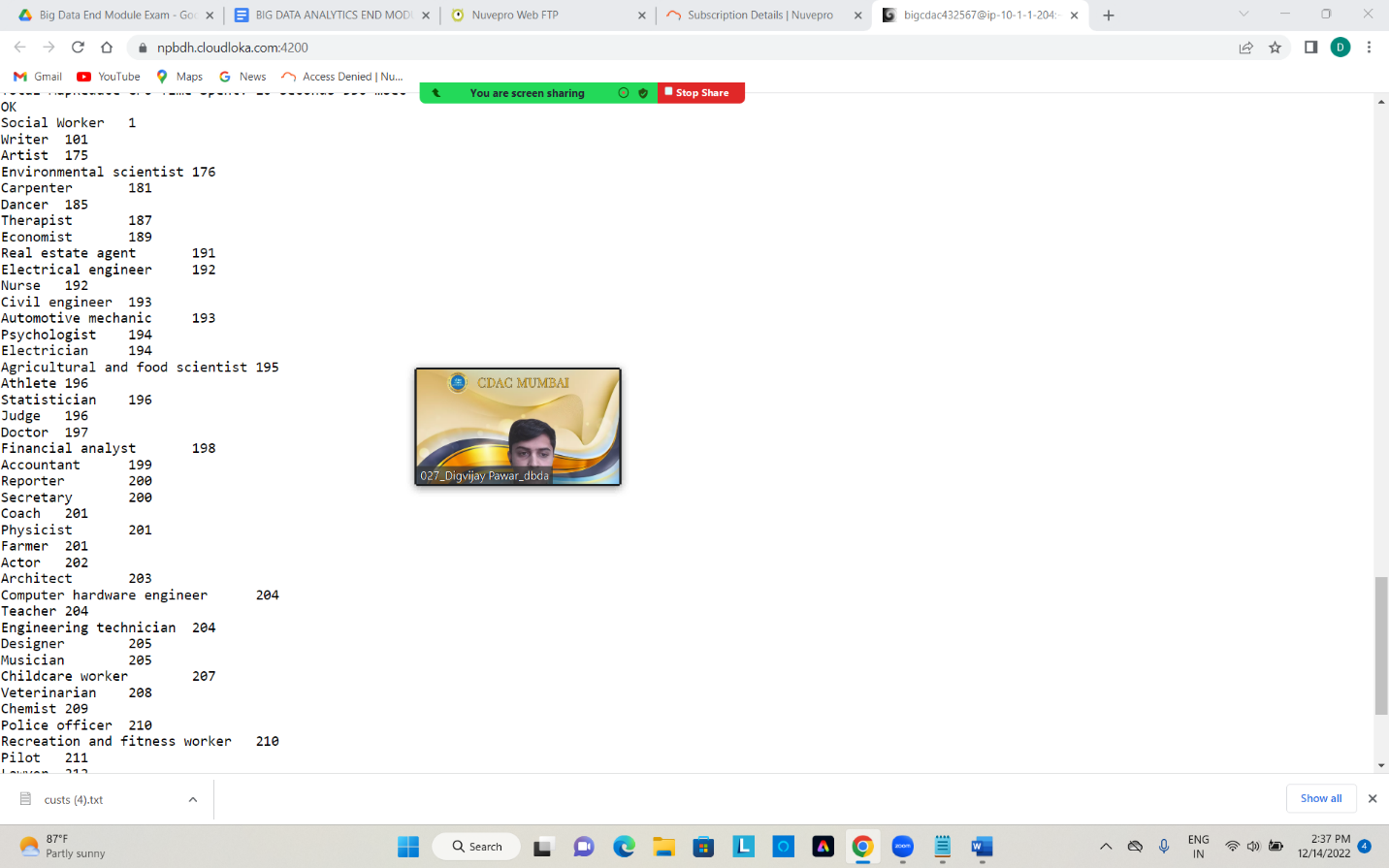
fields terminated by ','

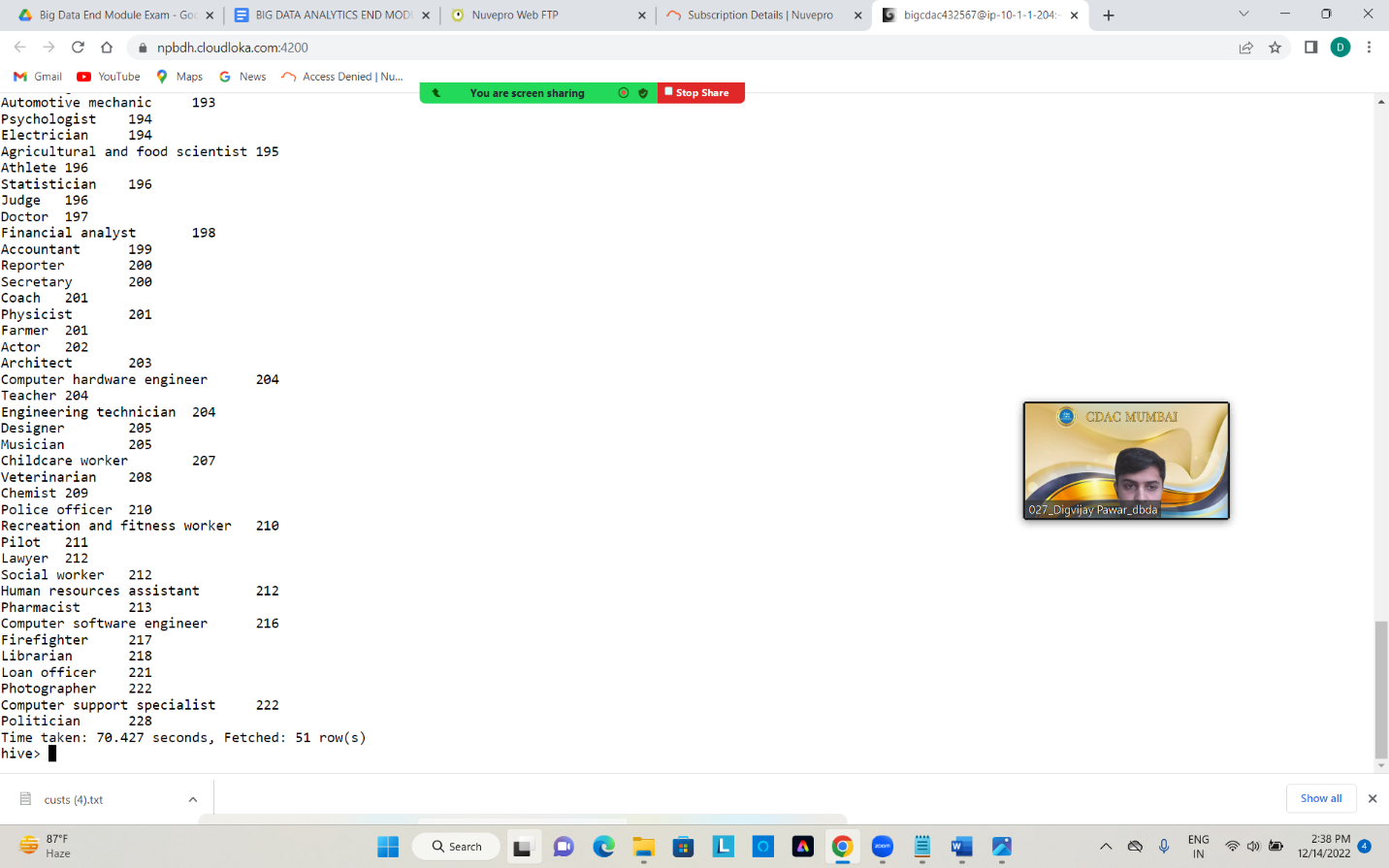
stored as textfile;

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

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

select profession,count(\*) as headcount from customer group by profession order by headcount;





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

create table txnrecords(

txnno int,

txndate string,

custno 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 txnrecords;

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

Yoga & Pilates 47804.93999999993

Swing Sets 47204.13999999999

Lawn Games 46828.44

Golf 46577.67999999999

Cardio Machine Accessories 46485.540000000045

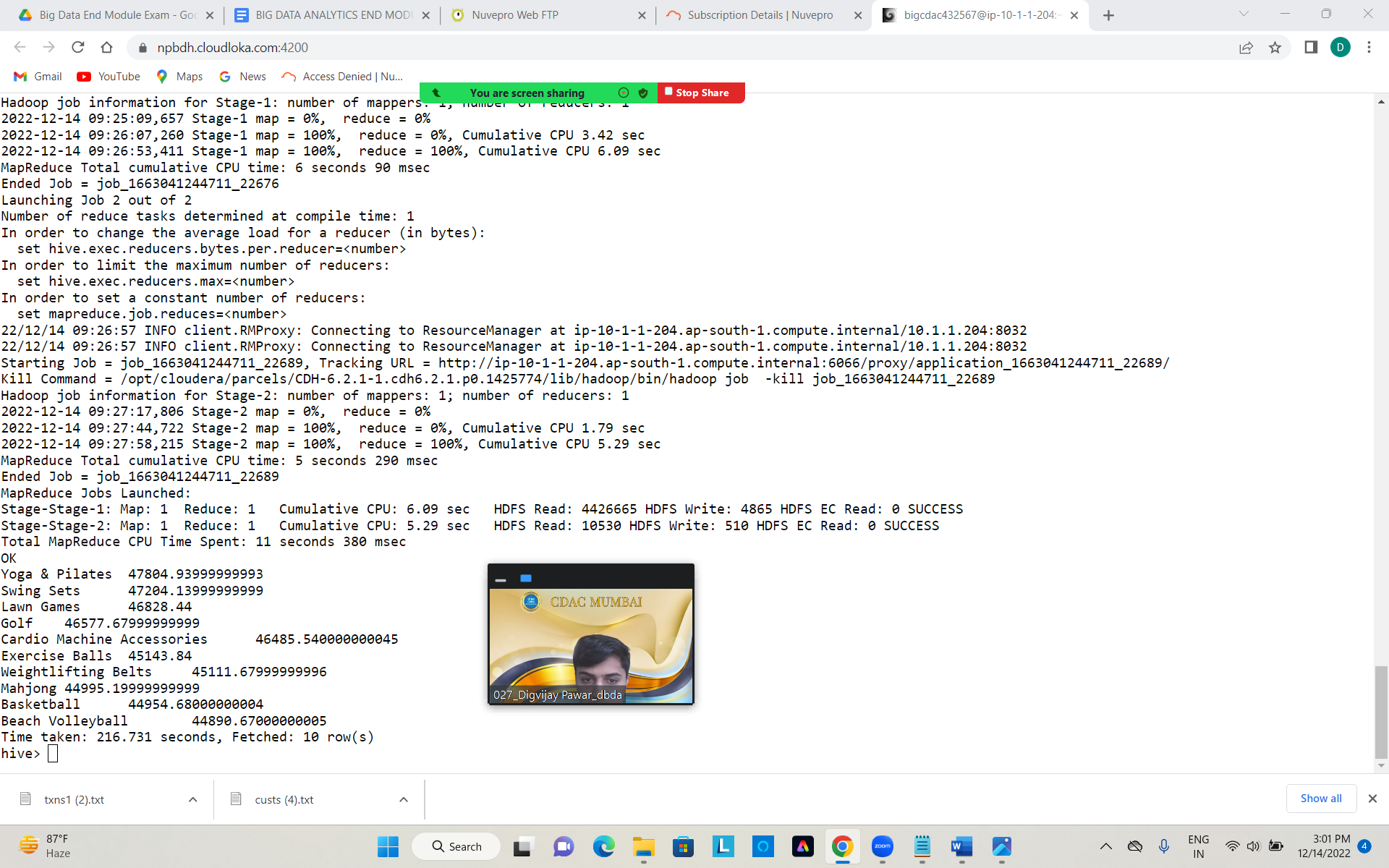
Exercise Balls 45143.84

Weightlifting Belts 45111.67999999996

Mahjong 44995.19999999999

Basketball 44954.68000000004

Beach Volleyball 44890.67000000005



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

create table txnrecspar(

txnno int,

txndate string,

custno int,

amount double,

product string,

city string,

state string,

spendby string)

partitioned by (category string)

row format delimited

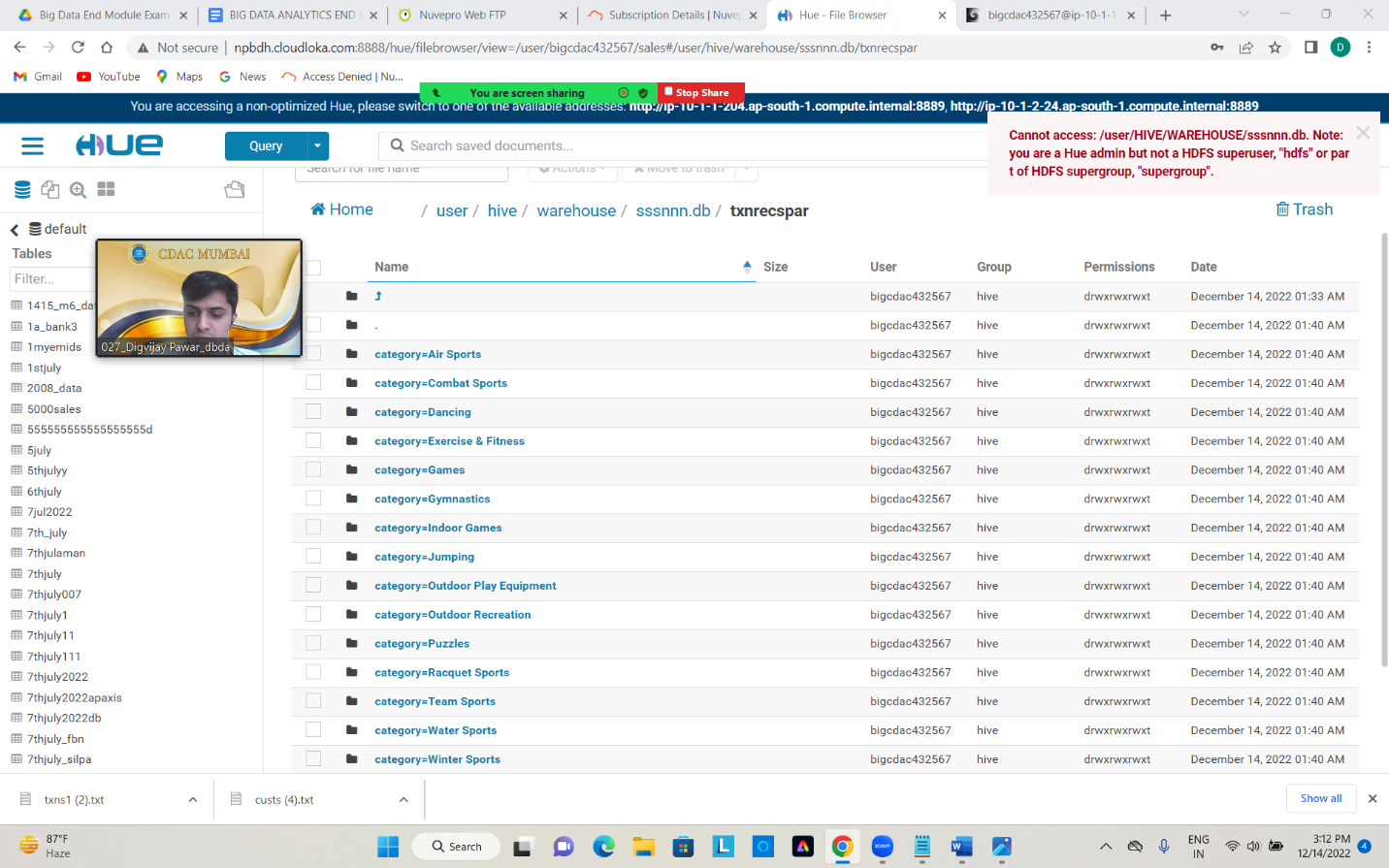
fields terminated by ','

stored as textfile;

set hive.exec.dynamic.partiton.mode=nonstrict;

INSERT OVERWRITE TABLE txnrecspar 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;



**Question 1&2 : Find all time High price for each stock and map reduce code for all time high**

**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>

{

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

{

**try**{

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

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

context.write(**new** Text(str[1]),**new** DoubleWritable(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** max = 0.00;

**for** (DoubleWritable val : values)

{

**if** (val.get() > max) {

max = val.get();

}

}

result.set(max);

context.write(key, result);

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

}

}

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

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

conf.set("mapreduce.output.textoutputformat.separator",",");

//conf.set("name", "value")

conf.set("mapreduce.input.fileinputformat.split.maxsize", "28311552");

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

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

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

job.setCombinerClass(ReduceClass.**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);

}

}

Command

hadoop jar myjar.jar AllTimeHigh /user/bigcdac432567/training/NYSE.csv /user/bigcdac432567/training/out1704

few records output

AVP,90.45

AVT,81.12

AVX,100.0

AVY,78.5

AWC,32.85

AWF,15.46

AWH,53.48

AWI,57.48

AWK,23.77

AWP,20.55

AWR,48.0

AXA,80.94

AXE,88.4

AXL,42.1

AXP,169.5

AXR,149.99

AXS,43.35

AYE,65.48

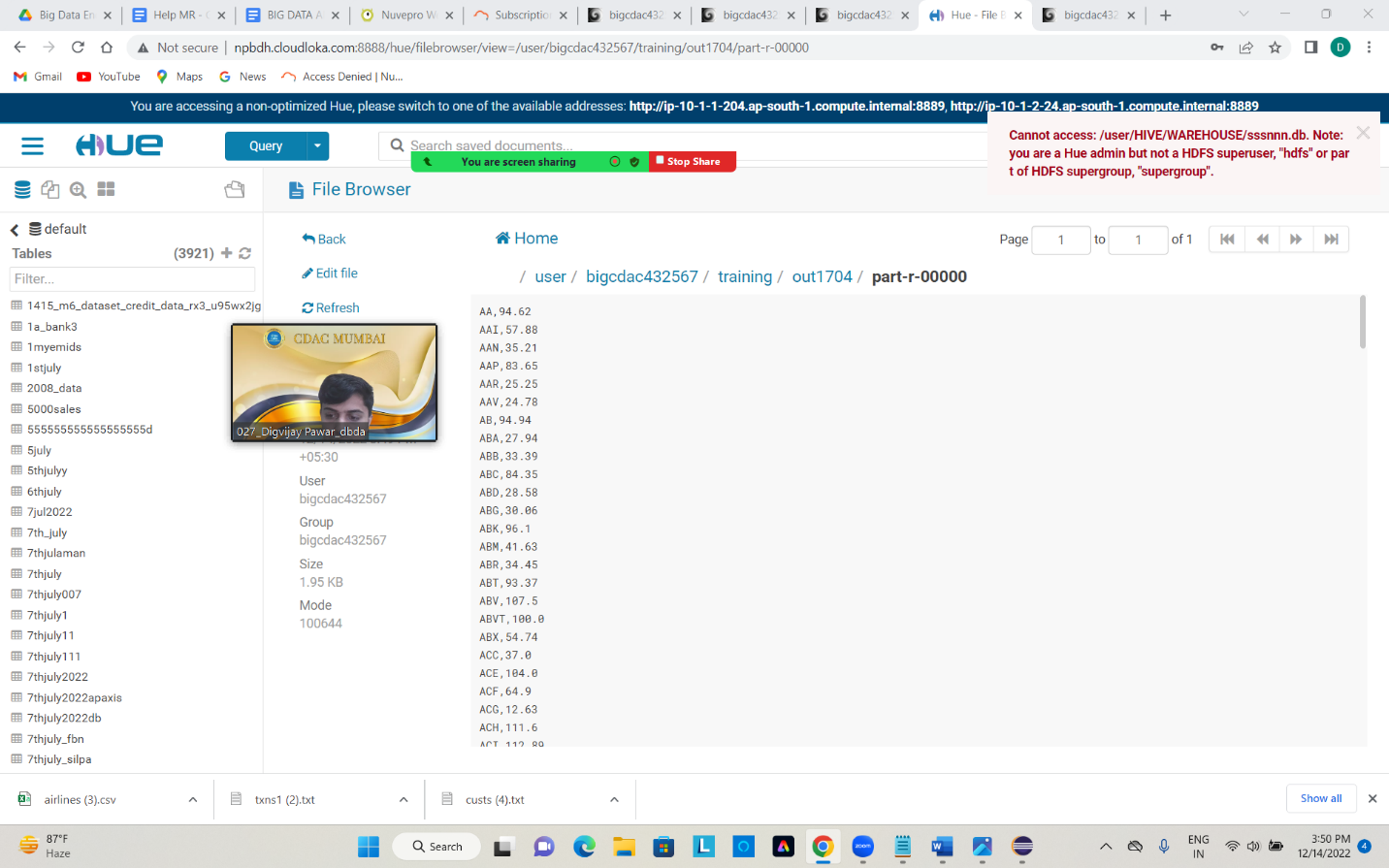
AYI,66.89

AYN,15.42

AYR,41.31

AZN,145.41

AZO,169.99



**QUESTION 3 [15 marks]**

**PySpark**

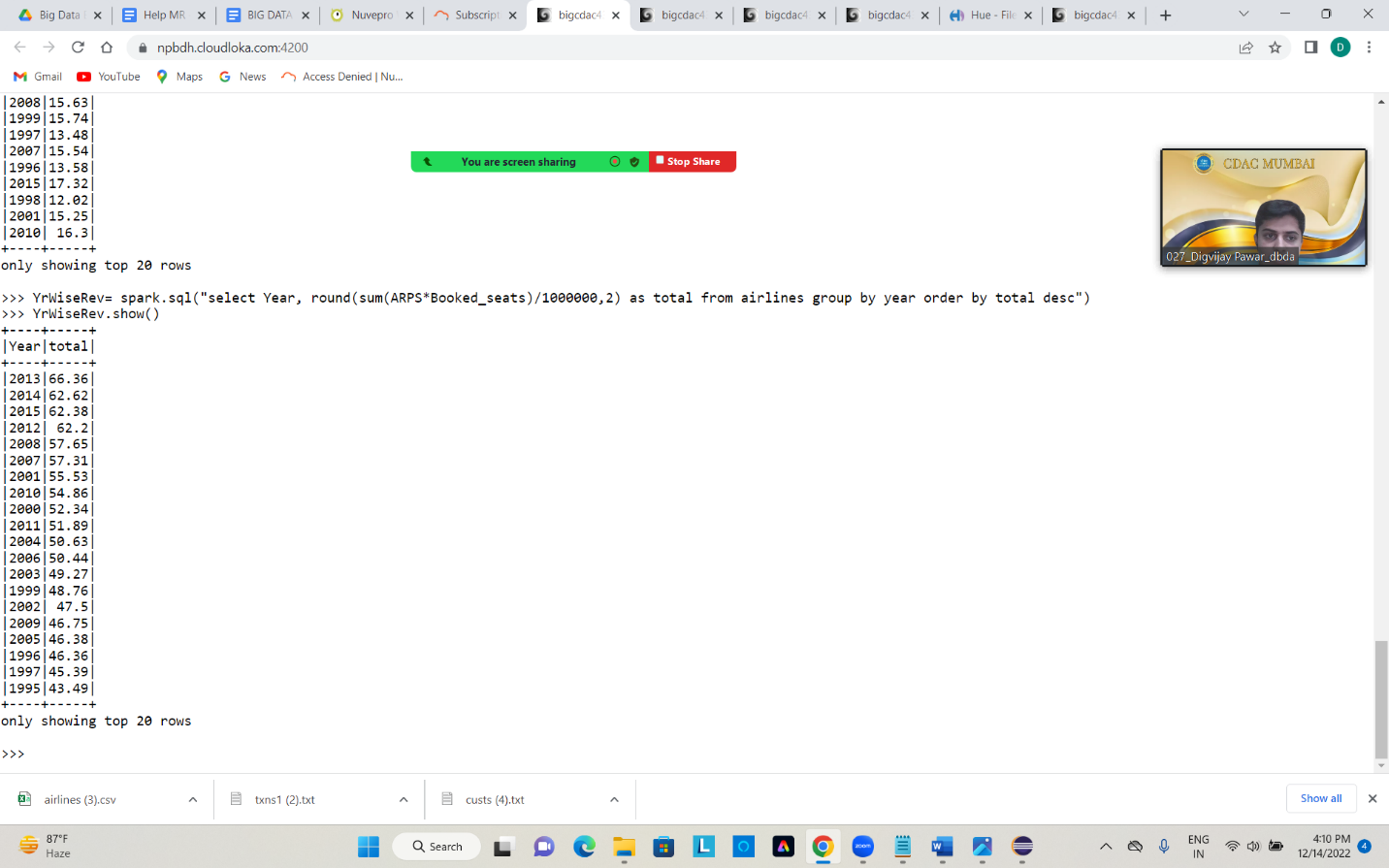
from pyspark.sql.types import StructType,StringType,IntegerType,DoubleType,LongType

schema2122=StructType().add("Year",StringType(),True).add("Quarter",StringType(),True).add("ARPS",DoubleType(),True).add("Booked\_seats",IntegerType(),True)

df\_with\_schema21= spark.read.format("csv").option("header","True").schema(schema2122).load("hdfs://nameservice1/user/bigcdac432567/training/airlines.csv")

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

YrWiseRev= spark.sql("select Year, round(sum(ARPS\*Booked\_seats)/1000000,2) as total from airlines group by year order by total desc")

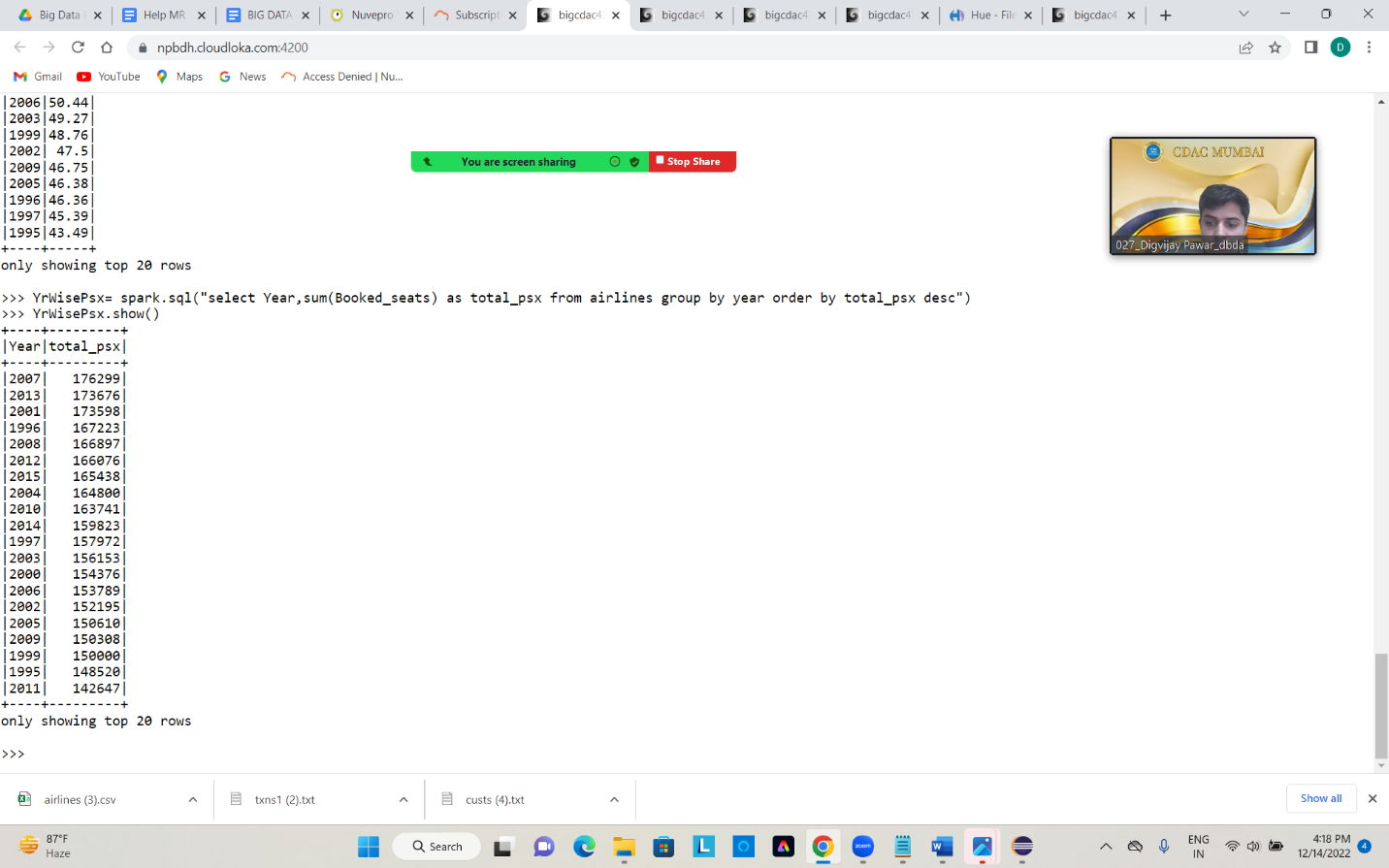


**highest revenue generation for which year=2013**

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

YrWisePsx= spark.sql("select Year,sum(Booked\_seats) as total\_psx from airlines group by year order by total\_psx desc")

YrWisePsx.show()

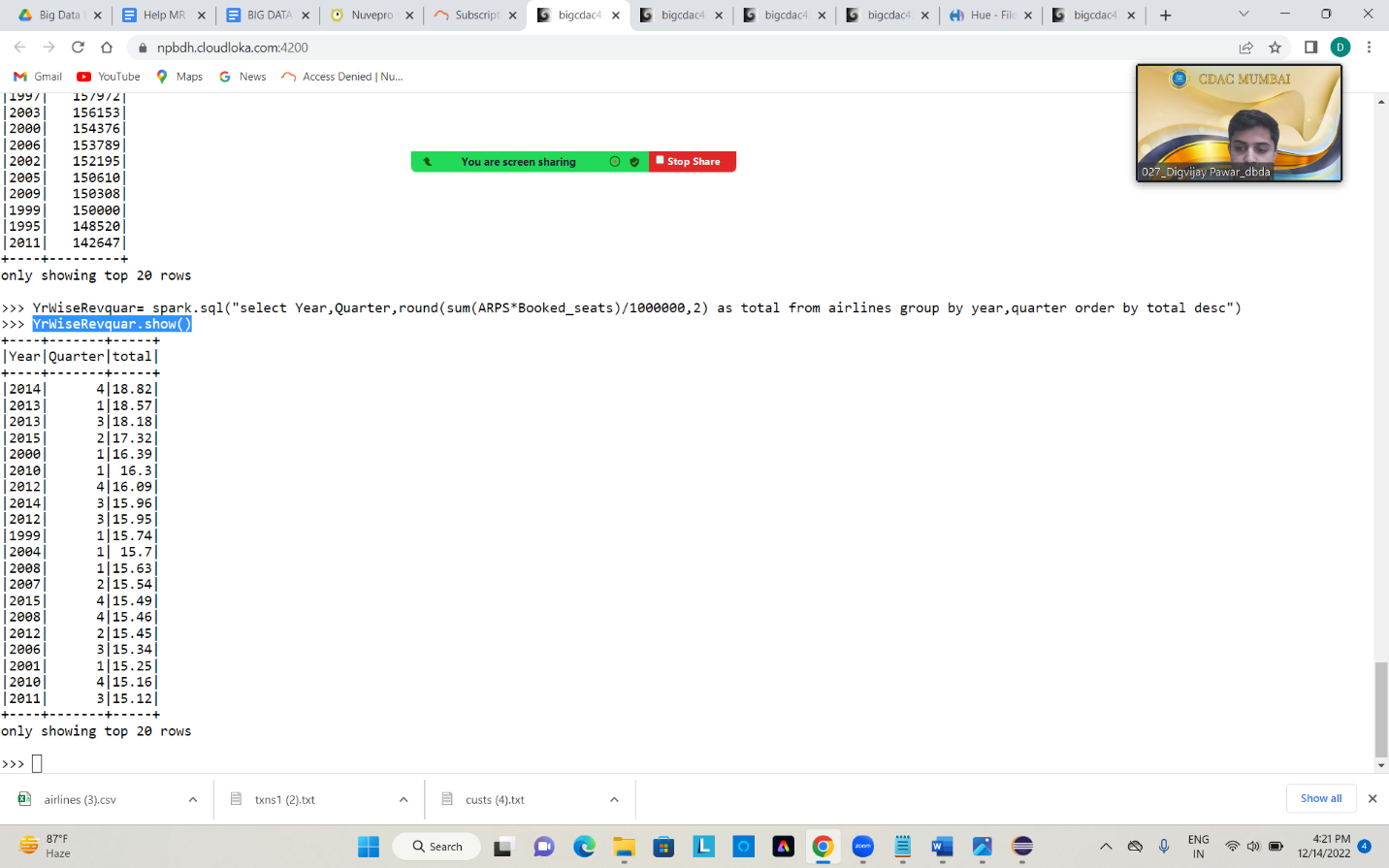


**highest number of people travelled in which year=2007**

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

YrWiseRevquar= spark.sql("select Year,Quarter,round(sum(ARPS\*Booked\_seats)/1000000,2) as total from airlines group by year,quarter order by total desc")

YrWiseRevquar.show()



**highest revenue generation for which year and quarter= year 2014 and quarter 4**