**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.**

**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**

**Question 2 : Find all time High price for each stock**

**ANSWER CODE :**

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;

//Question 2 : Find all time High price for each stock

public class AllTimeHigh

{

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

{

protected void map(LongWritable key, Text values,Context context)throws IOException, InterruptedException

{

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

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

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

}

}

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

{

protected void reduce(Text keys, Iterable<DoubleWritable> values,Context context) throws IOException, InterruptedException

{

double max=0.00;

for(DoubleWritable var:values)

{

if(var.get()>max)

{

max=var.get();

}

}

context.write(keys, new DoubleWritable(max));

}

}

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

Configuration conf = new Configuration();

Job job = Job.getInstance(conf, "All time high price");

job.setJarByClass(AllTimeHigh.class);

job.setMapperClass(MapClass.class);

job.setReducerClass(ReduceClass.class);

job.setNumReduceTasks(1);

//job.setMapOutputKeyClass();

//job.setMapOutputValueClass();

job.setOutputKeyClass(Text.class);

job.setOutputValueClass(DoubleWritable.class);

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

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

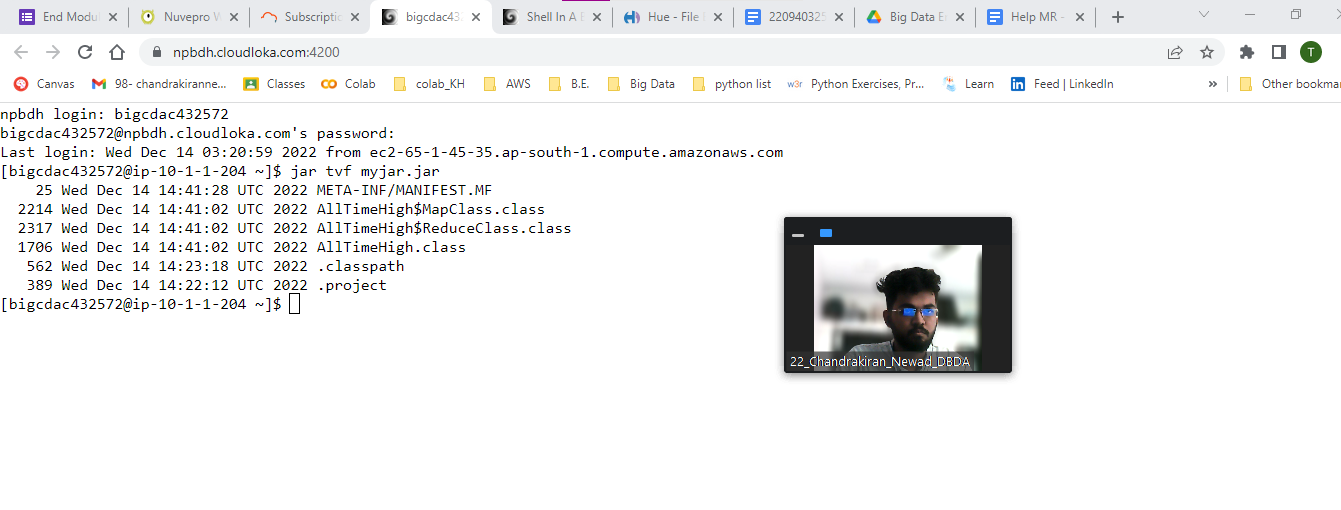
System.exit(job.waitForCompletion(true) ? 0 : 1);

}

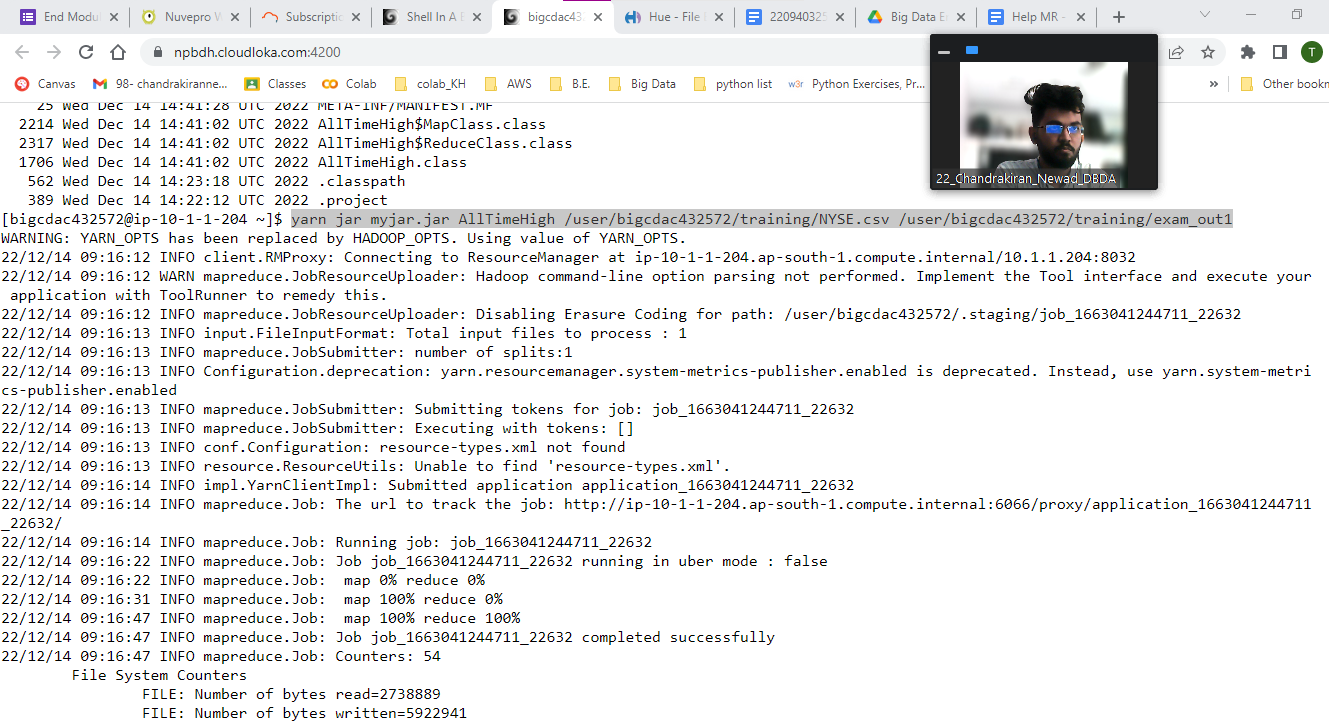
}

**COMMANDS:-**

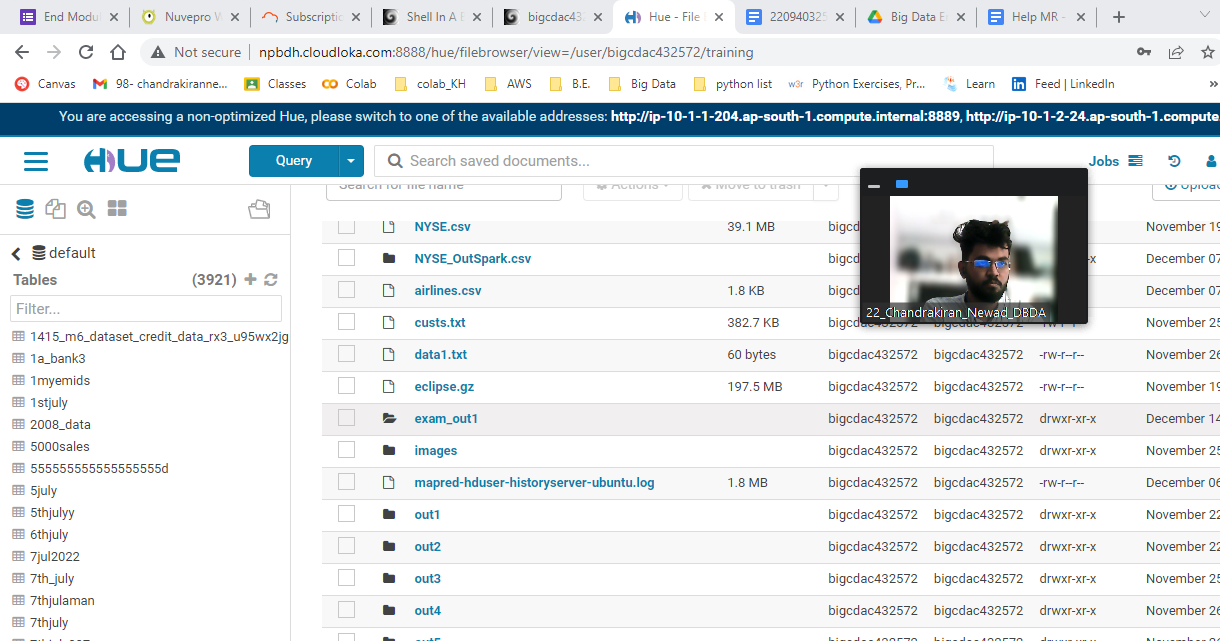
**jar tvf myjar.jar**

****

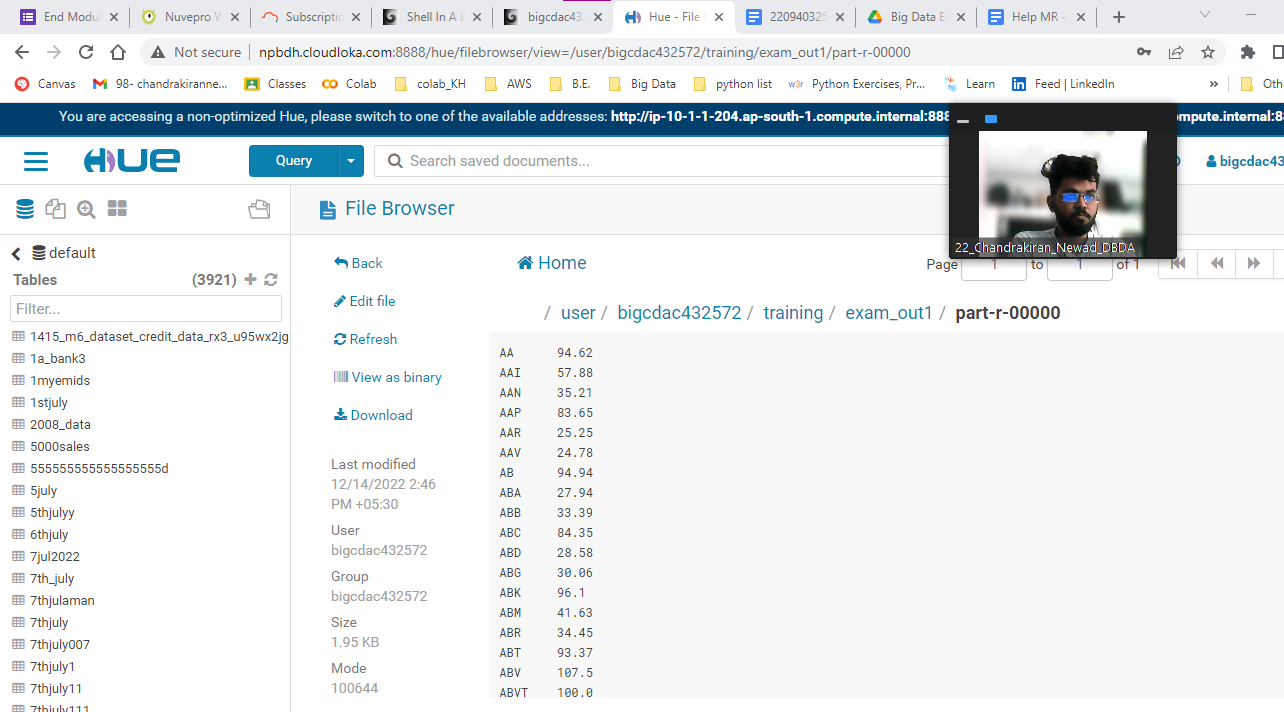
**yarn jar myjar.jar AllTimeHigh /user/bigcdac432572/training/NYSE.csv /user/bigcdac432572/training/exam\_out1**

****

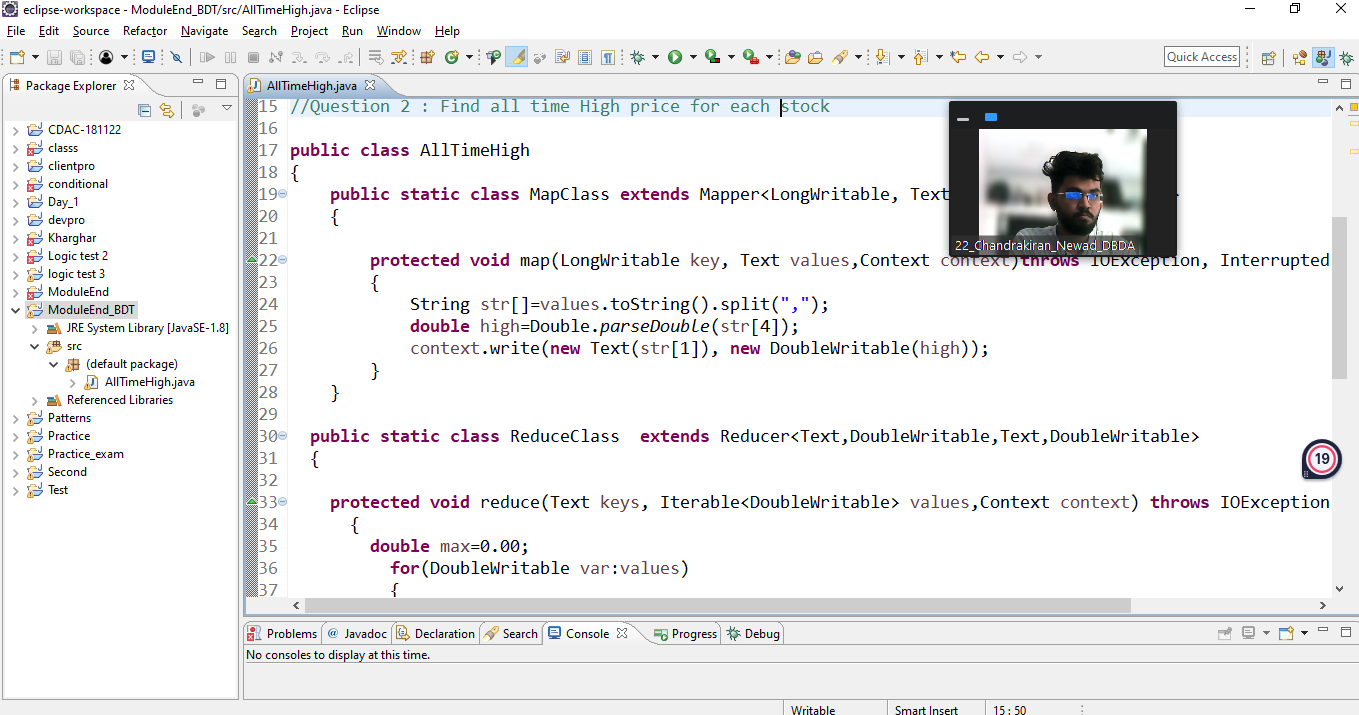
**OUTPUT FILE :**

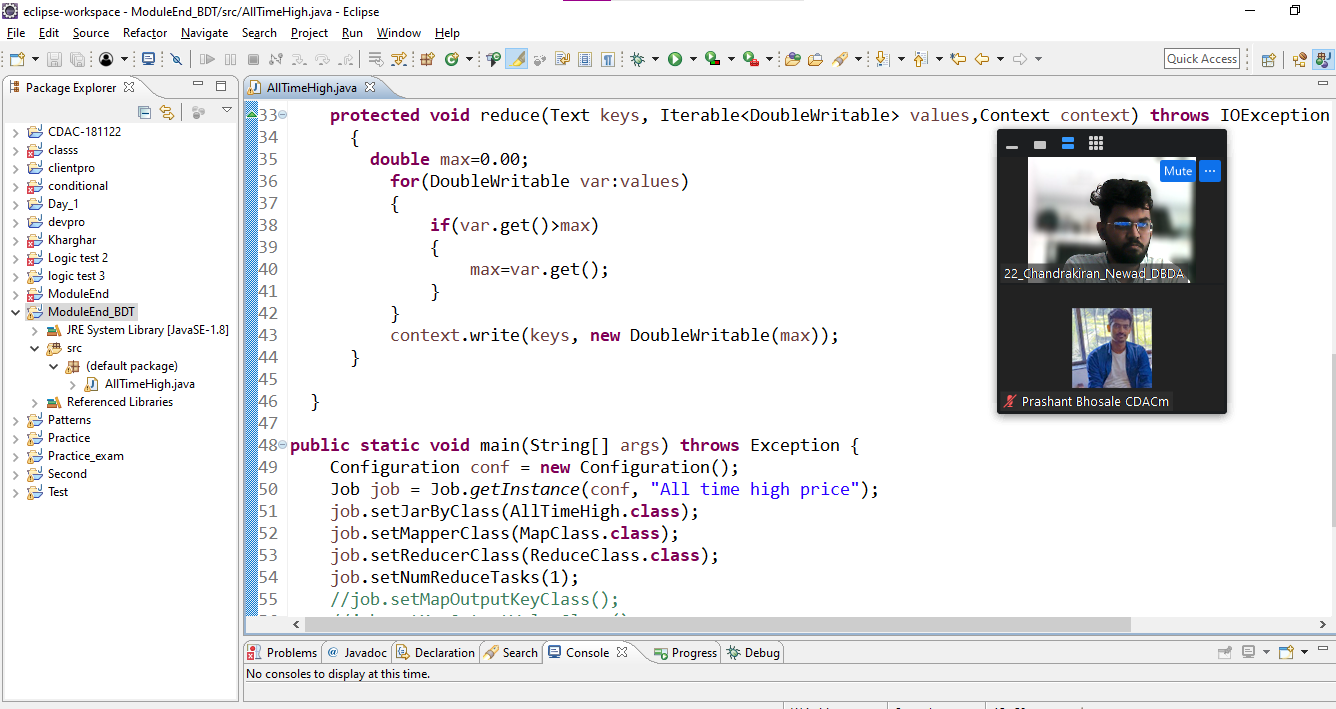
****

**FINAL OUTPUT:**

****

**JAVA CODE:**

****

****

**HIVE PRACTICAL :**

**Hive**

**Please find the customer data set.**

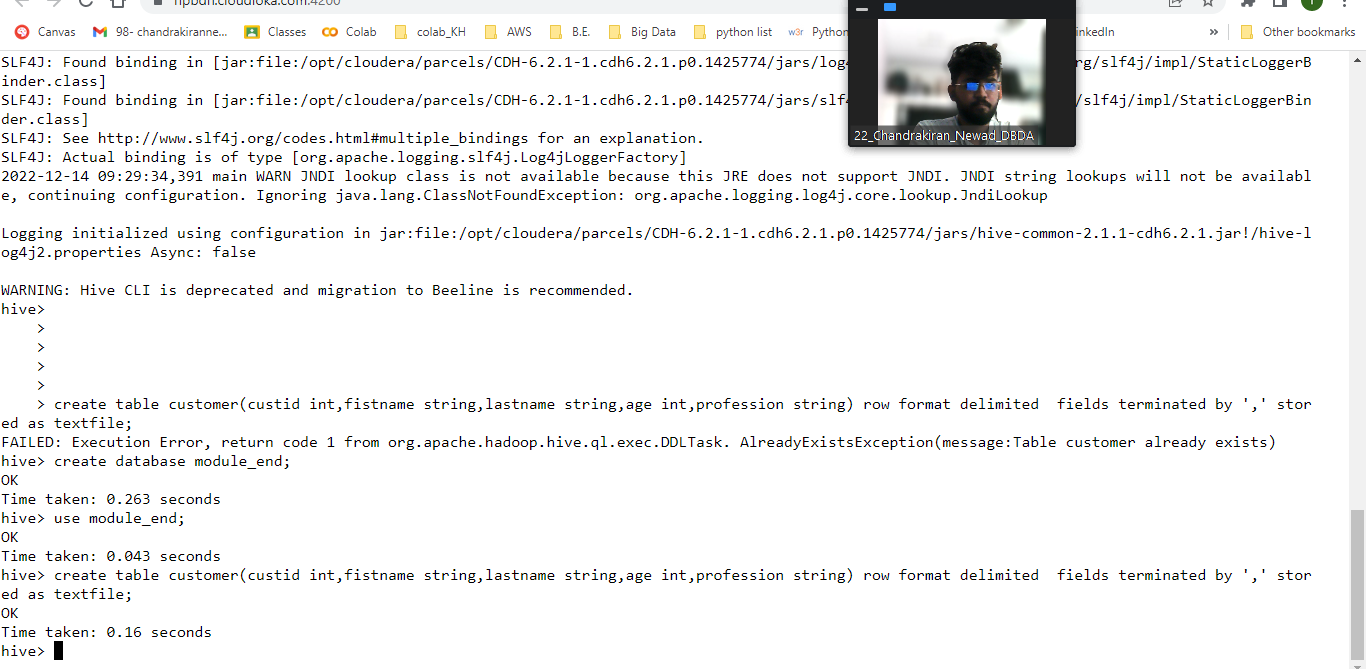
**cust id**

**firstname**

**lastname**

**age**

**profession**

****

**1-create database module\_end;**

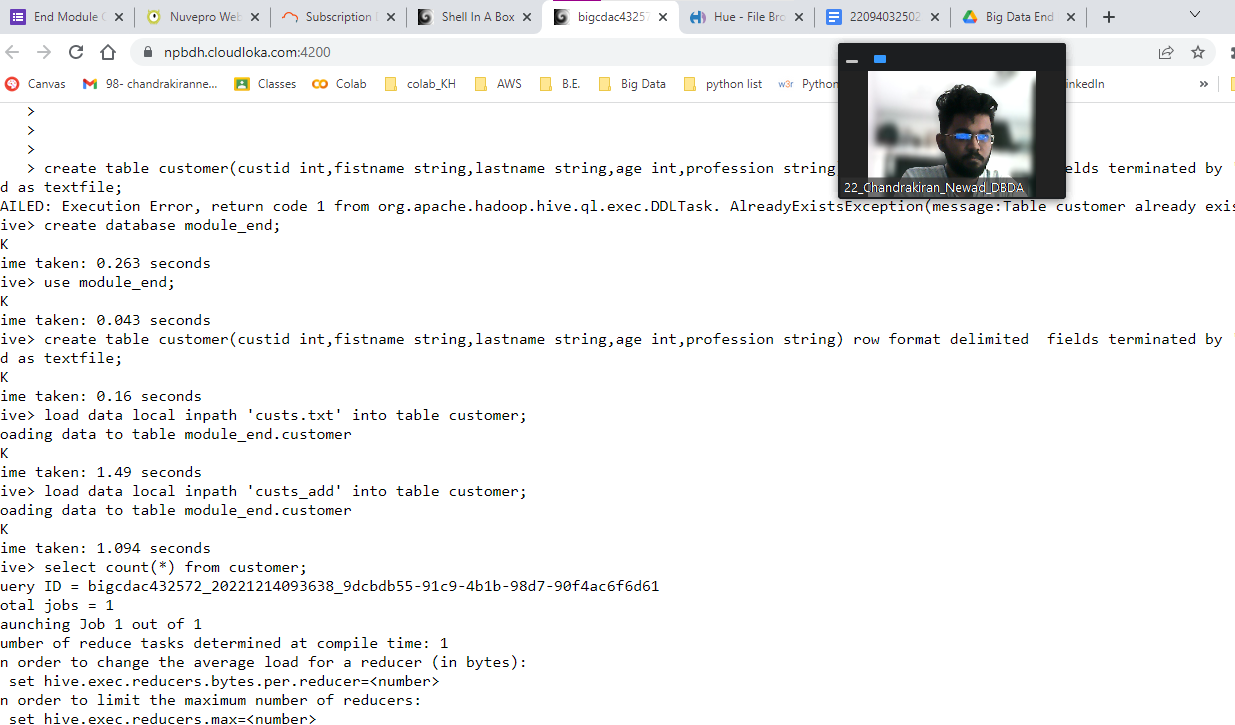
**2-use module\_end;**

**3-create table customer(custid int,fistname string,lastname string,age int,profession string)**

**row format delimited**

**fields terminated by ','**

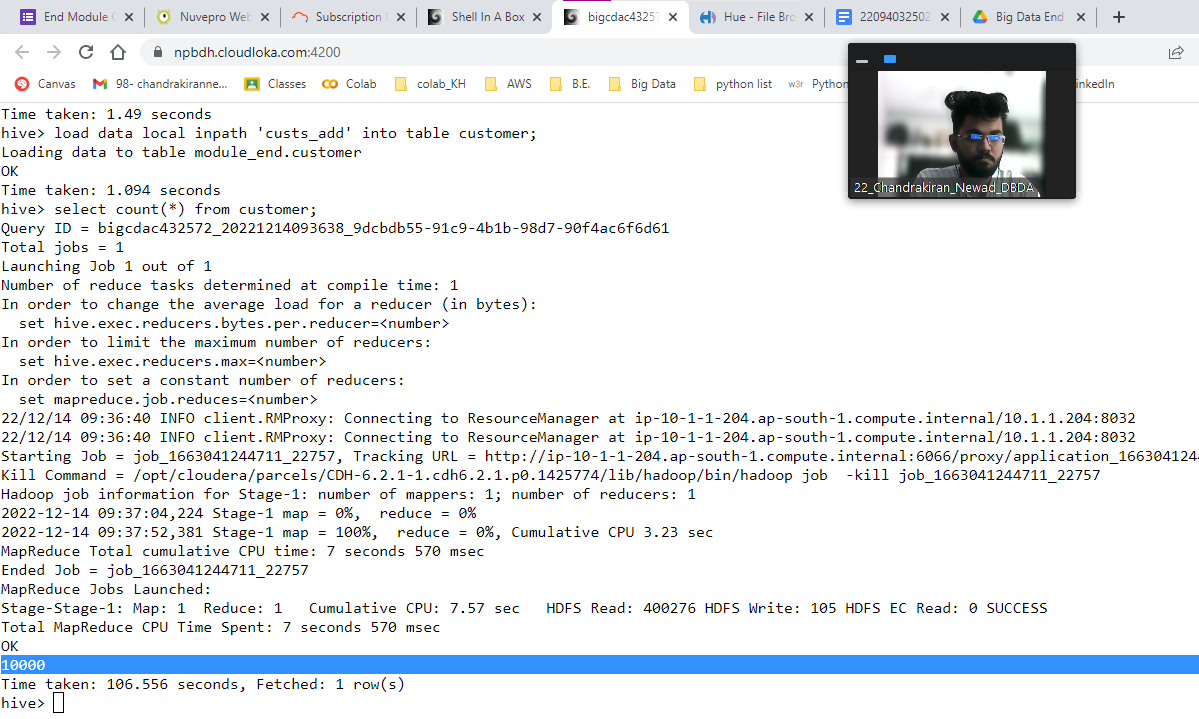
**stored as textfile;**

****

**4-load data local inpath 'custs.txt' into table customer;**

**5-load data local inpath 'custs\_add' into table customer;**

**6-select Count(\*) from customer;**

****

**Please find the sales data set.**

**txn id**

**txn date**

**cust id**

**amount**

**category**

**product**

**city**

**state**

**spendby**

**7-[bigcdac432572@ip-10-1-1-204 ~]$ hdfs dfs -mkdir /user/bigcdac432572/exam**

**[bigcdac432572@ip-10-1-1-204 ~]$ hdfs dfs -put txns1.txt exam**

**hive> create table txnrecords(txnid int,txndate string,custid int,amount double,**

**>**

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

**>**

**> row format delimited**

**>**

**> fields terminated by ','**

**>**

**> stored as textfile**

**>**

**> location '/user/bigcdac432572/exam'**

**> ;**

**hive> set hive.cli.print.current.db=true;**

**hive (module\_end)> select \* from txnrecords limit 5;**

**OK**

**0 06-26-2011 4007024 40.33 Exercise & Fitness Cardio Machine Accessories Clarksville Tennessee credit**

**1 05-26-2011 4006742 198.44 Exercise & Fitness Weightlifting Gloves Long Beach California credit**

**2 06-01-2011 4009775 5.58 Exercise & Fitness Weightlifting Machine Accessories Anaheim California credit**

**3 06-05-2011 4002199 198.19 Gymnastics Gymnastics Rings Milwaukee Wisconsin credit**

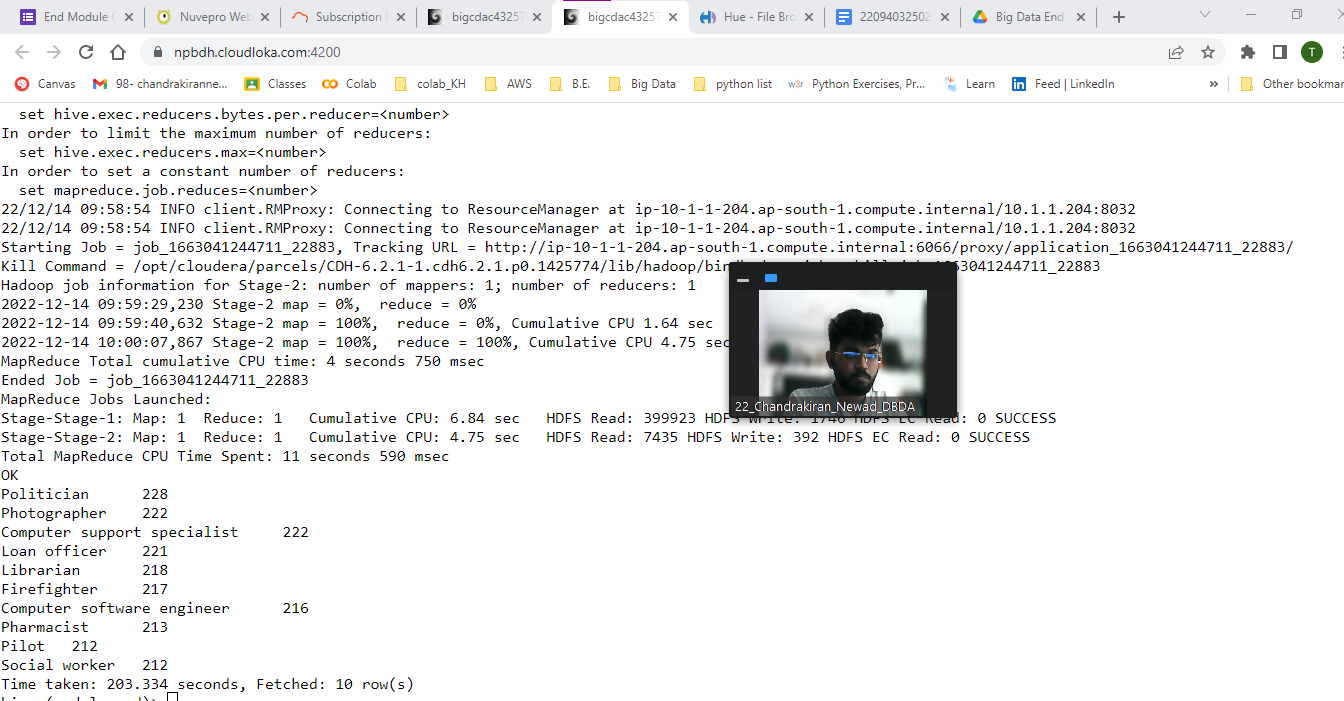
**4 12-17-2011 4002613 98.81 Team Sports Field Hockey Nashville Tennessee credit**

**Time taken: 0.34 seconds, Fetched: 5 row(s)**

**hive (module\_end)>**

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

select profession,count(custid)as total\_count from customer group by profession order by total\_count desc limit 10;

****

**Politician 228**

**Photographer 222**

**Computer support specialist 222**

**Loan officer 221**

**Librarian 218**

**Firefighter 217**

**Computer software engineer 216**

**Pharmacist 213**

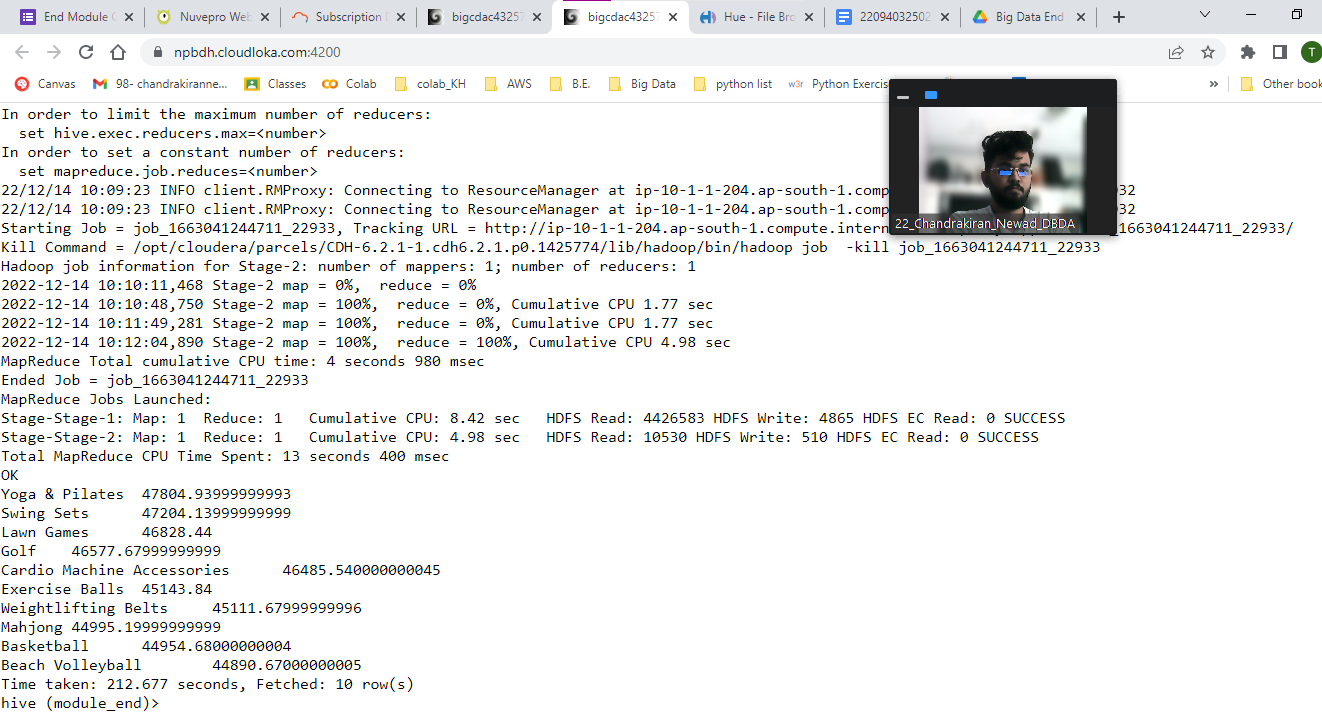
**Pilot 212**

**Social worker 212**

**Time taken: 203.334 seconds, Fetched: 10 row(s)**

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

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

****

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

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

set hive.exec.dynamic.partition=true;

create table txnrecsBycat(txnid int,txndate string,custid int,amount double,

product string,city string,state string,spendby string)

partitioned by(category string)

row format delimited

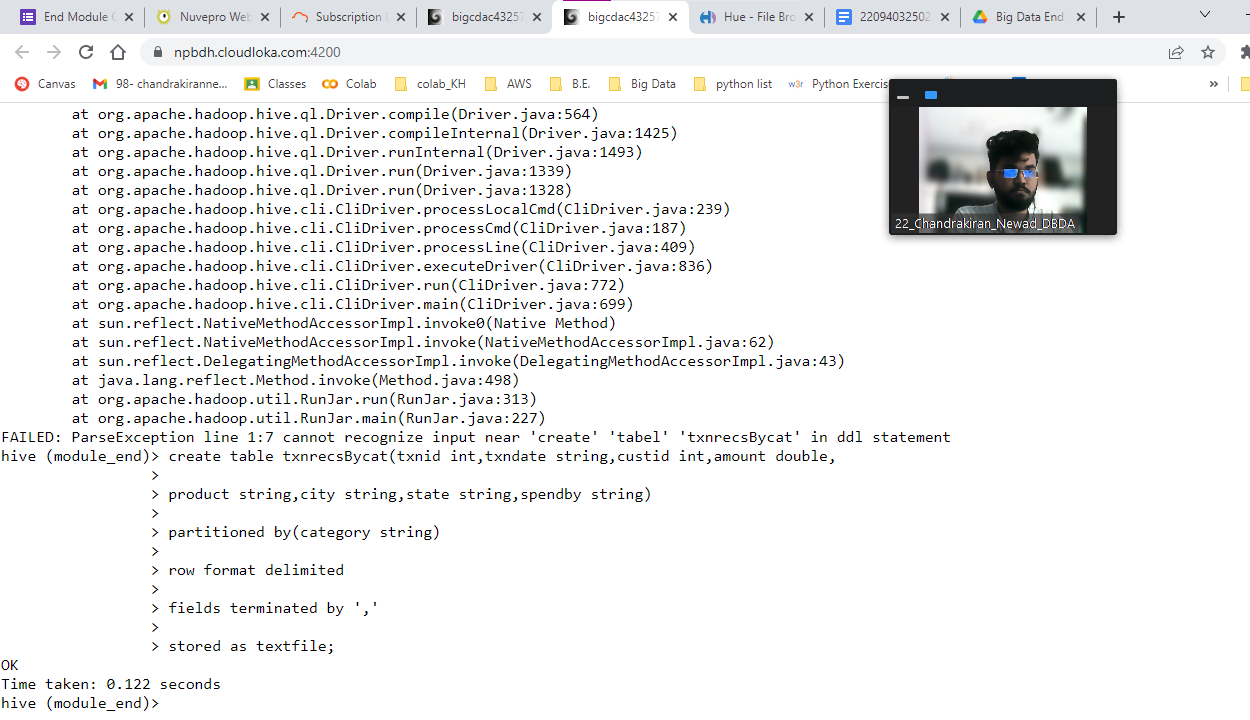
fields terminated by ','

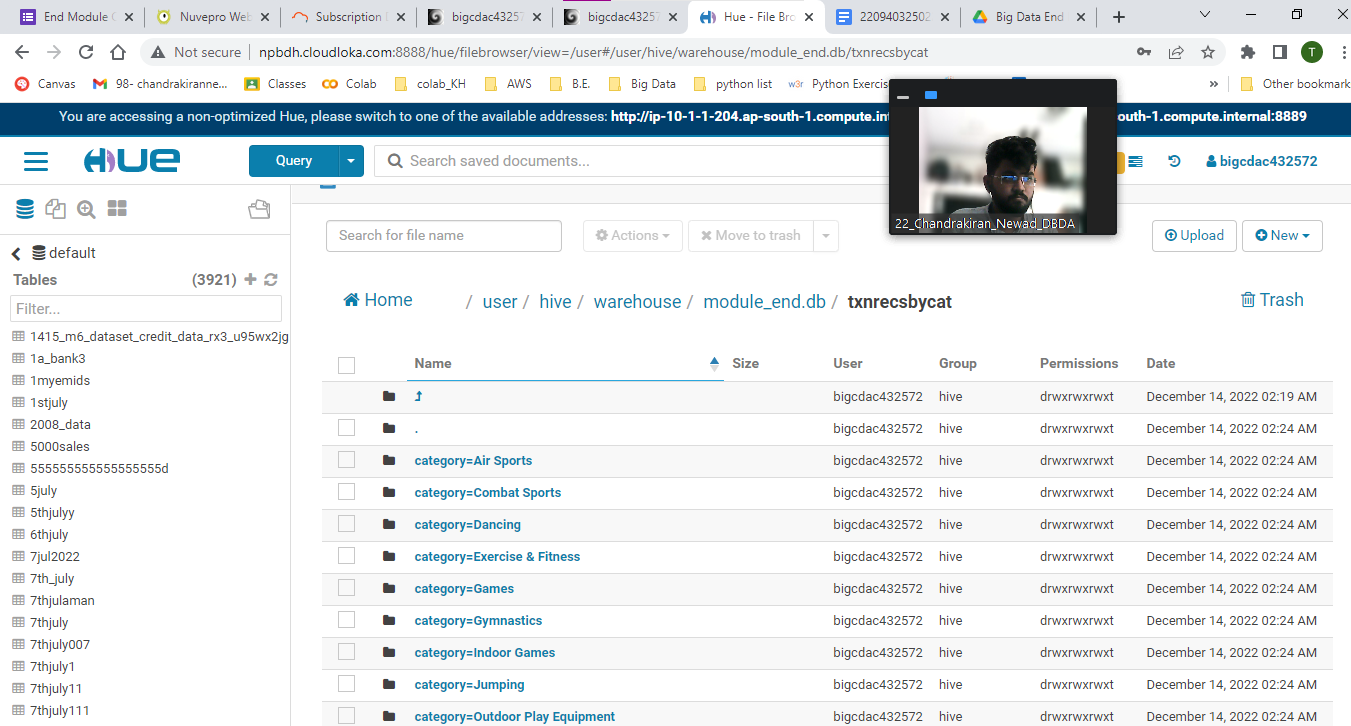
stored as textfile;

insert into table txnrecsBycat PARTITION(category)

select txn.txnid ,txn.txndate,txn.custid,txn.amount,txn.product,txn.city,

txn.state,txn.spendby,txn.category from txnrecords txn DISTRIBUTE by category;





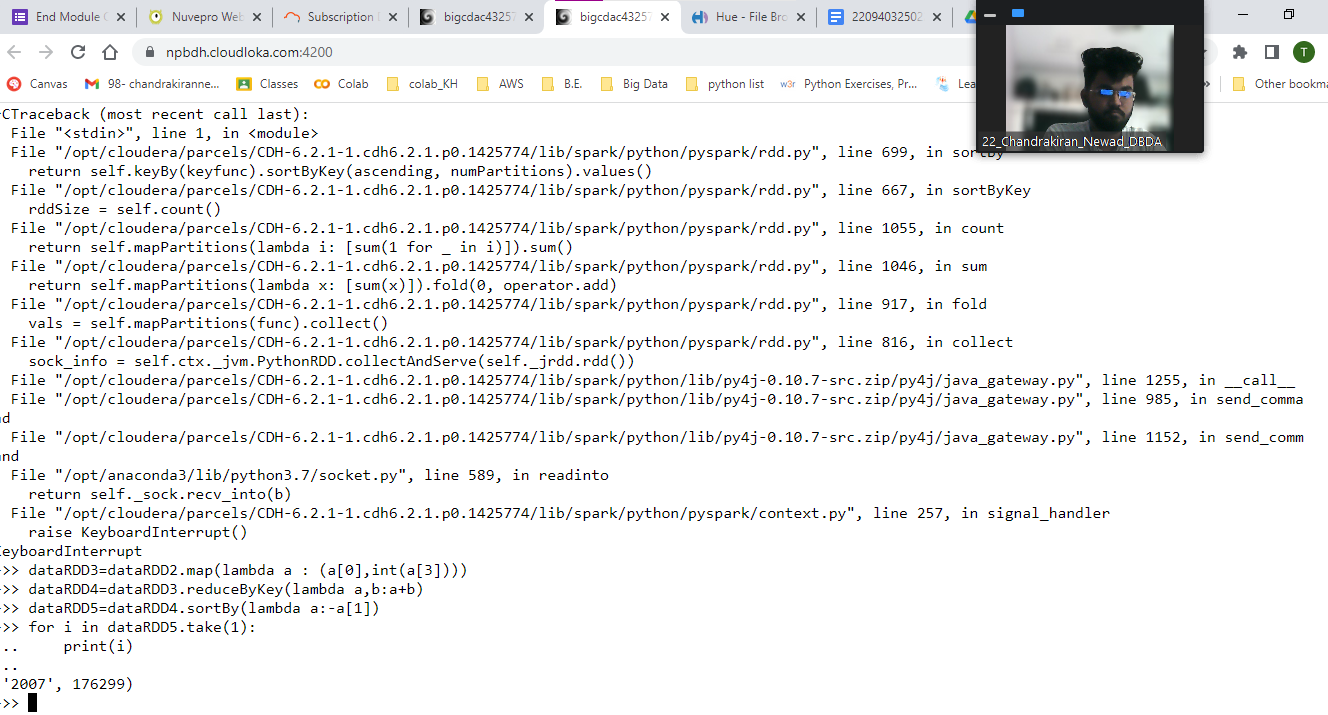
**PYSPARK :**

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

**dataRDD=sc.textFile("hdfs://nameservice1/user/bigcdac432572/training/airlines.csv")**

**>>> for i in dataRDD.take(5):**

**... print(i)**

****

**header=dataRDD.first()**

**eliminate=dataRDD.filter(lambda line: line!=header)**

**dataRDD2=eliminate.map(lambda a : a.split(","))**

**dataRDD3=dataRDD2.map(lambda a : (a[0],int(a[3])))**

**dataRDD4=dataRDD3.reduceByKey(lambda a,b:a+b)**

**dataRDD5=dataRDD4.sortBy(lambda a:-a[1])**

**('2007', 176299)**

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

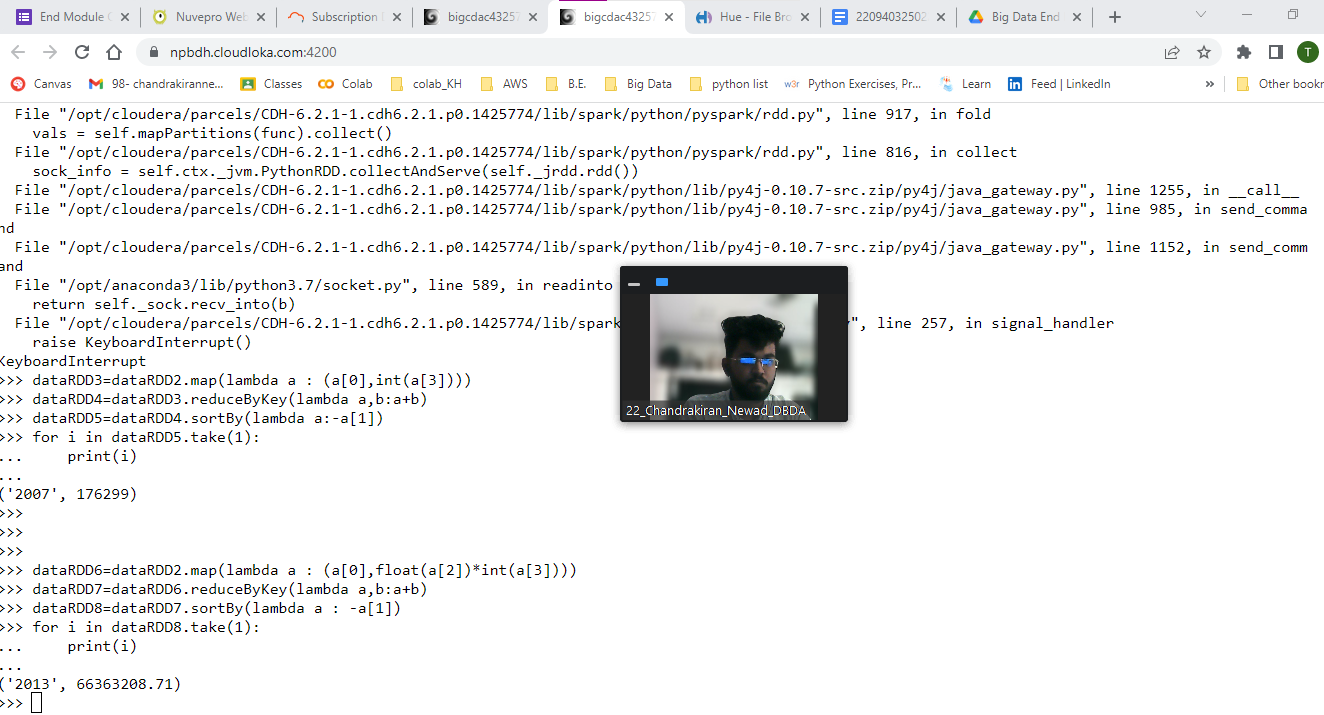
**dataRDD6=dataRDD2.map(lambda a : (a[0],float(a[2])\*int(a[3])))**

**dataRDD7=dataRDD6.reduceByKey(lambda a,b:a+b)**

**dataRDD8=dataRDD7.sortBy(lambda a : -a[1])**

**for i in dataRDD8.take(1):**

**print(i)**

****

**('2013', 66363208.71)**

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

**group)**