**BDT EXAM(14/12/22)**

**Question N0 2 :**

Hive Please find the customer data set.

cust id

Firstname

lastname

Age

Profession

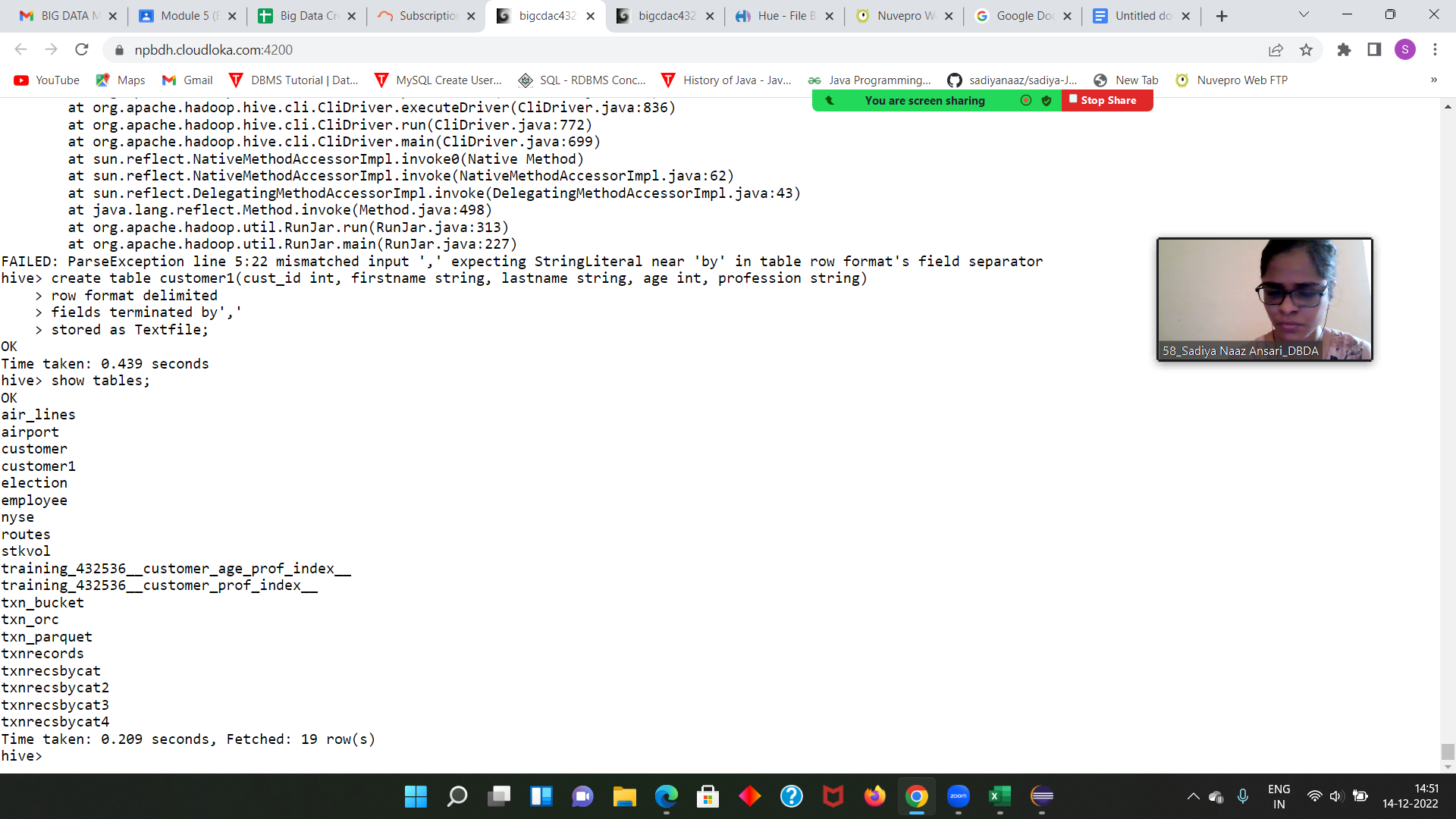
**1) Write a program to find the count of customers for each profession.**

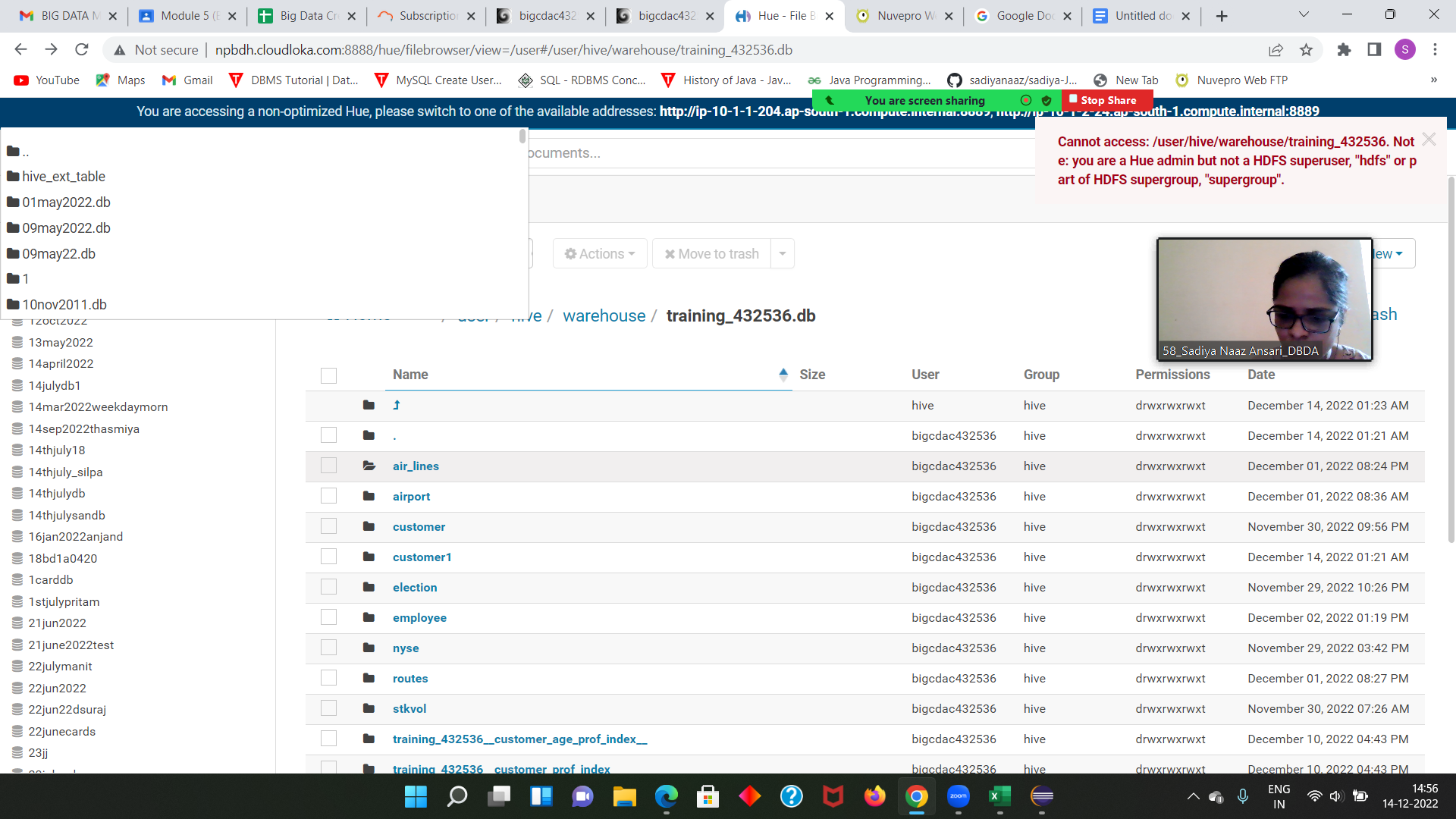
create table customer1 (cust\_id int, firstname string, lastname string, age int, profession string)

row format delimited

fields terminated by ‘,’

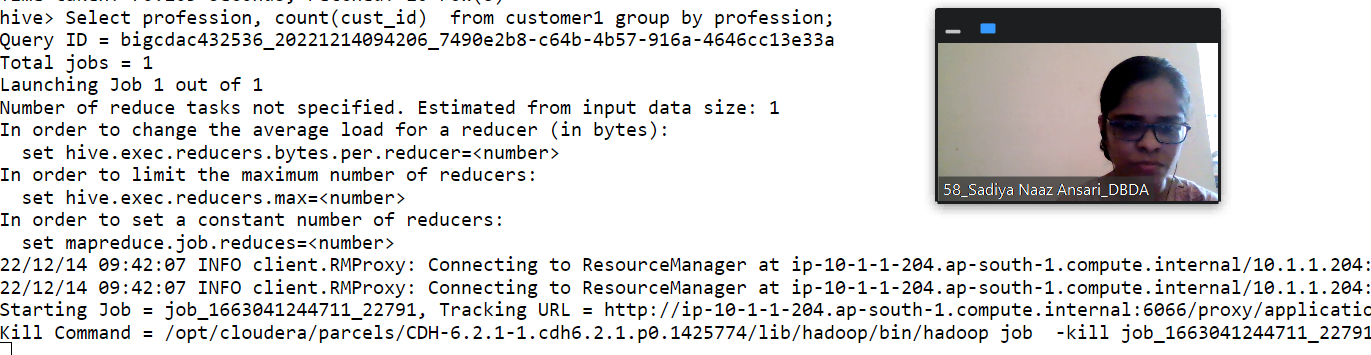
stored as Textfile;

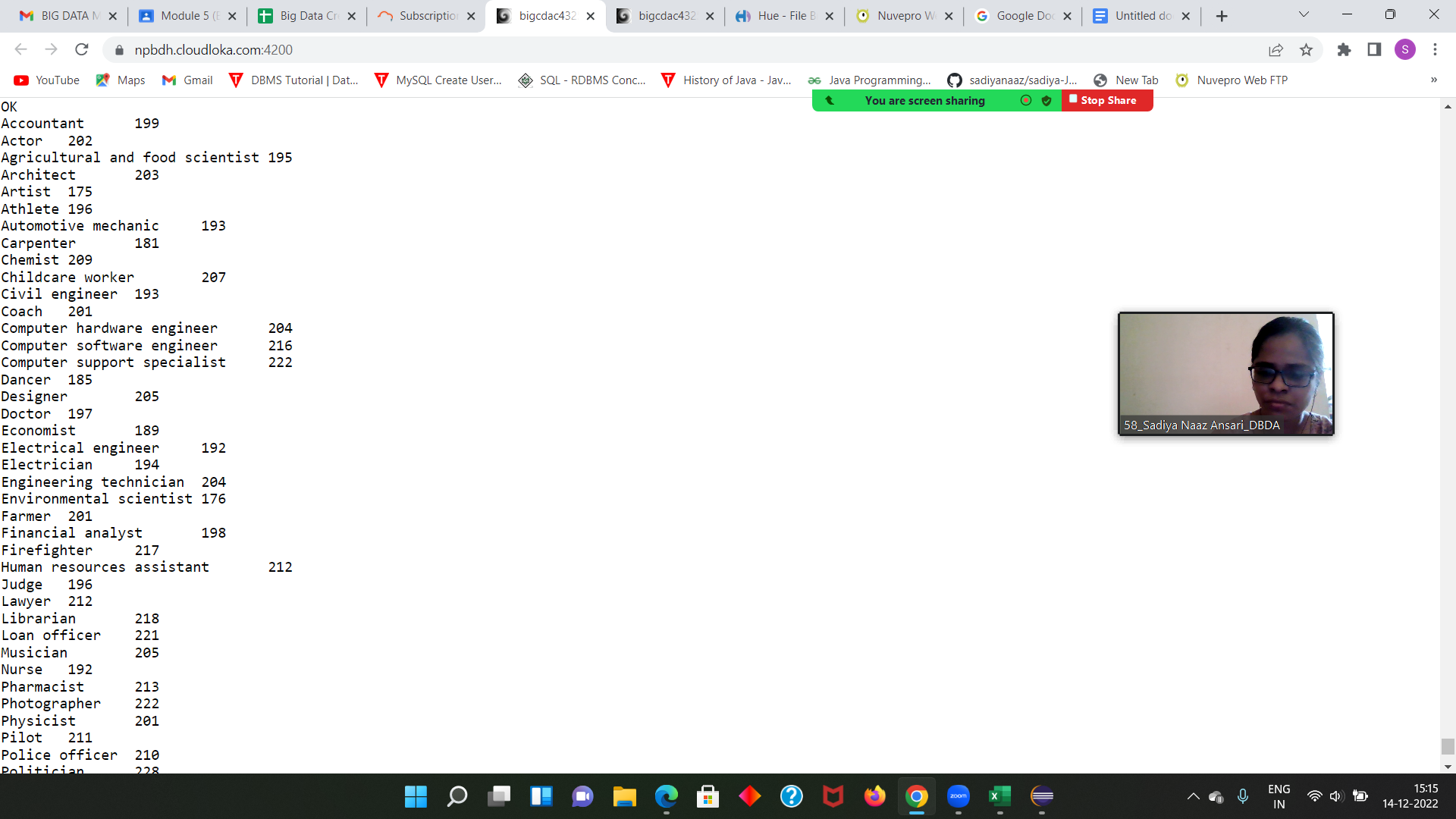




load data local inpath 'custs.txt' overwrite into table customer1;

Select profession, count(cust\_id) from customer1 group by profession;





Please find the sales data set.

txn id

txn date

cust id

amount

category

product

city

State

spendby

create table sales1 (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 ‘,’

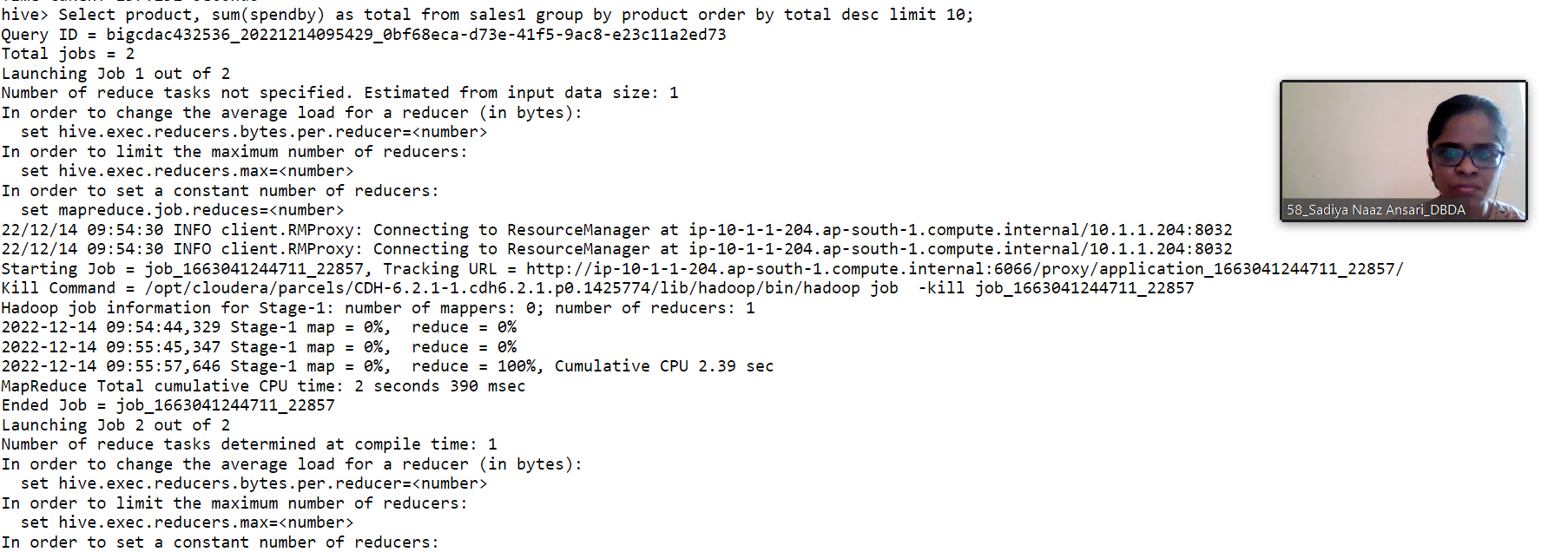
stored as Textfile;

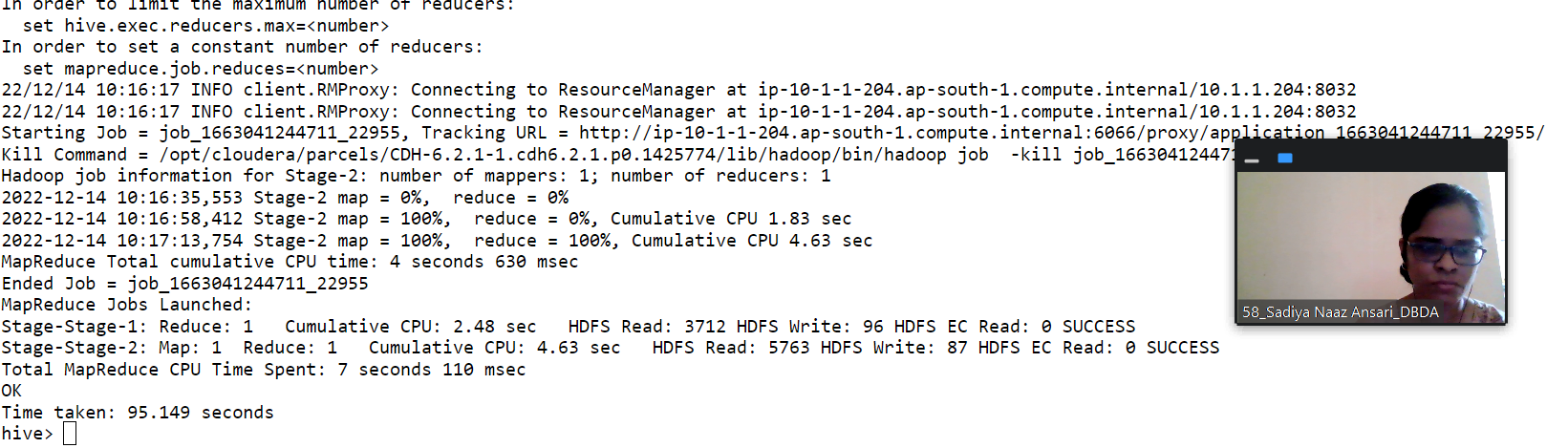
**2) Write a program to find the top 10 products sales wise**

****

**Select product, sum(spendby) as total from sales1 group by product order by total desc limit 10;**

****

****



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

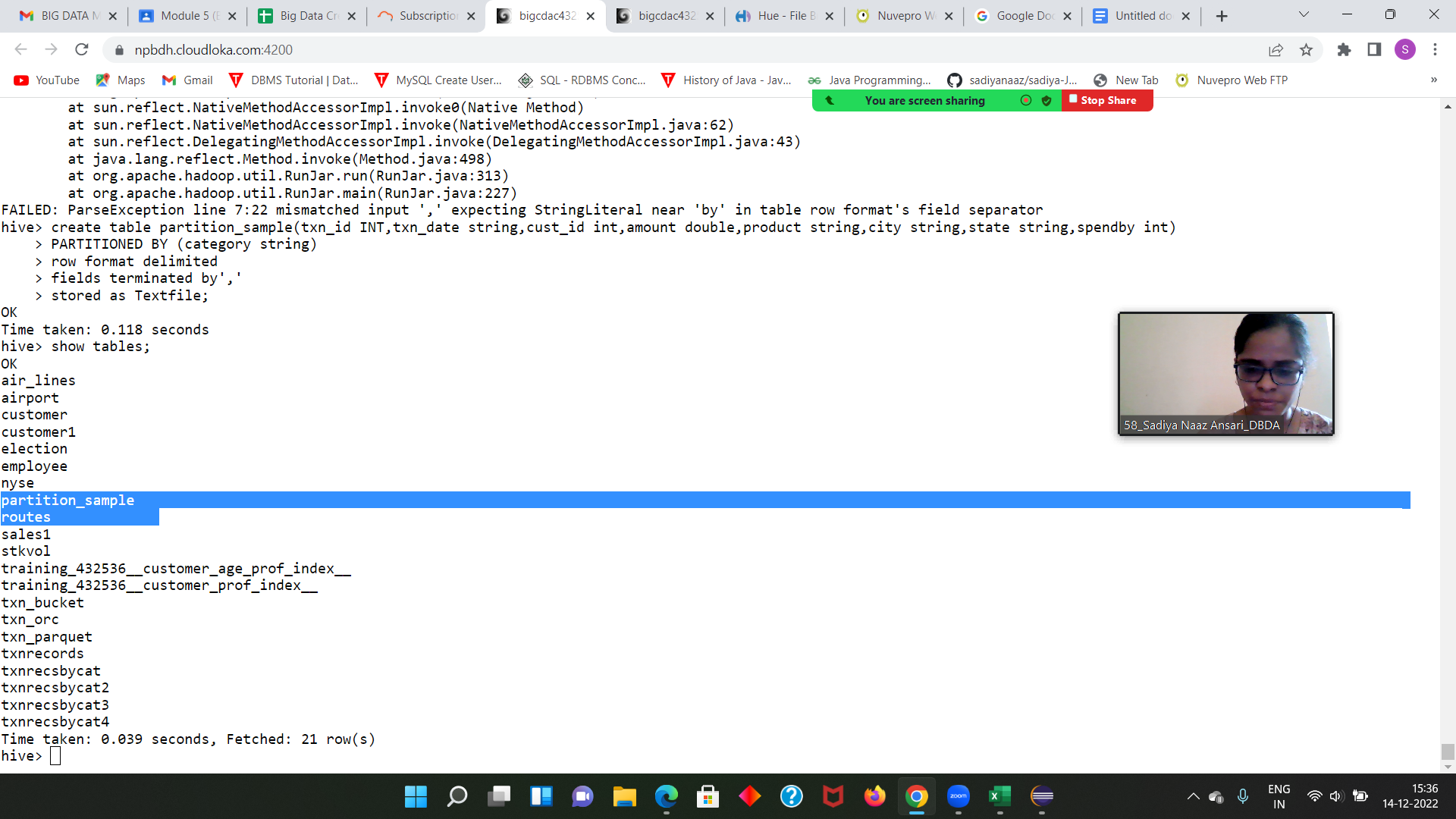
create table partition\_sample(txn\_id int,txn\_date string,cust\_id int,amount double,product string,city string,state string,spendby int)

Partitioned by(category string)

row format delimited

fields terminated by ‘,’

stored as Textfile;



**QUESTION 3 [15 marks] PySpark**

**Please find the AIRLINES data set**

**Year ,Quarter, Average revenue per seat, Total number of booked seats**

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

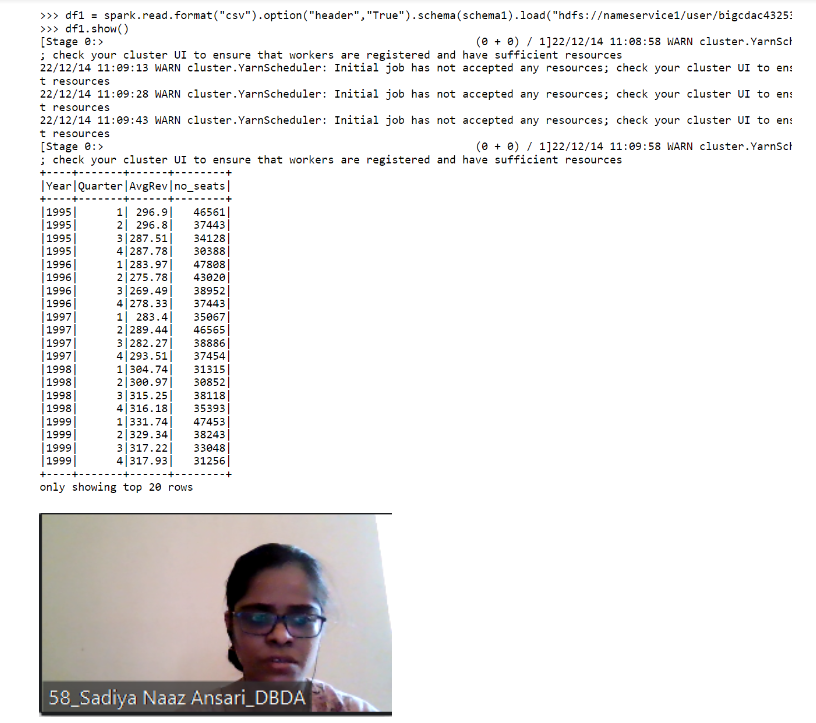
**schema1 = StructType().add("Year",StringType(),True).add("Quarter",IntegerType(),True).add("AvgRev",DoubleType(),True).add("no\_seats",LongType(),True)**

**df1 = spark.read.format("csv").option("header","True").schema(schema1).load("hdfs://nameservice1/user/bigcdac432536/exam/airlines.csv")**

**df1.show()**

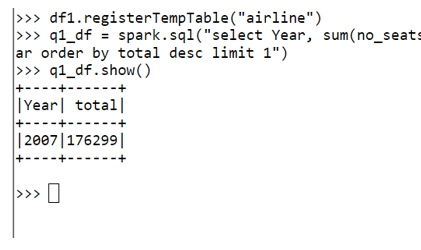
**df1.registerTempTable("airline")**

****

****

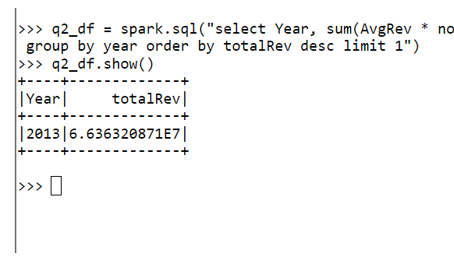
**Q.1**

**df = spark.sql("select year, sum(no\_seats) as total from airlines group by year order by total desc limit 1")**

****

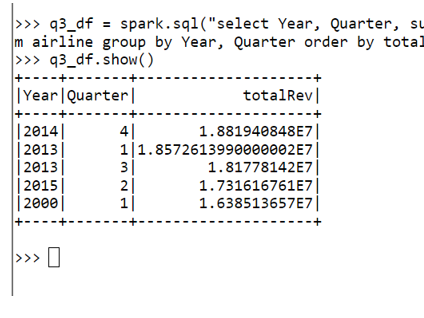
**Q.2**

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

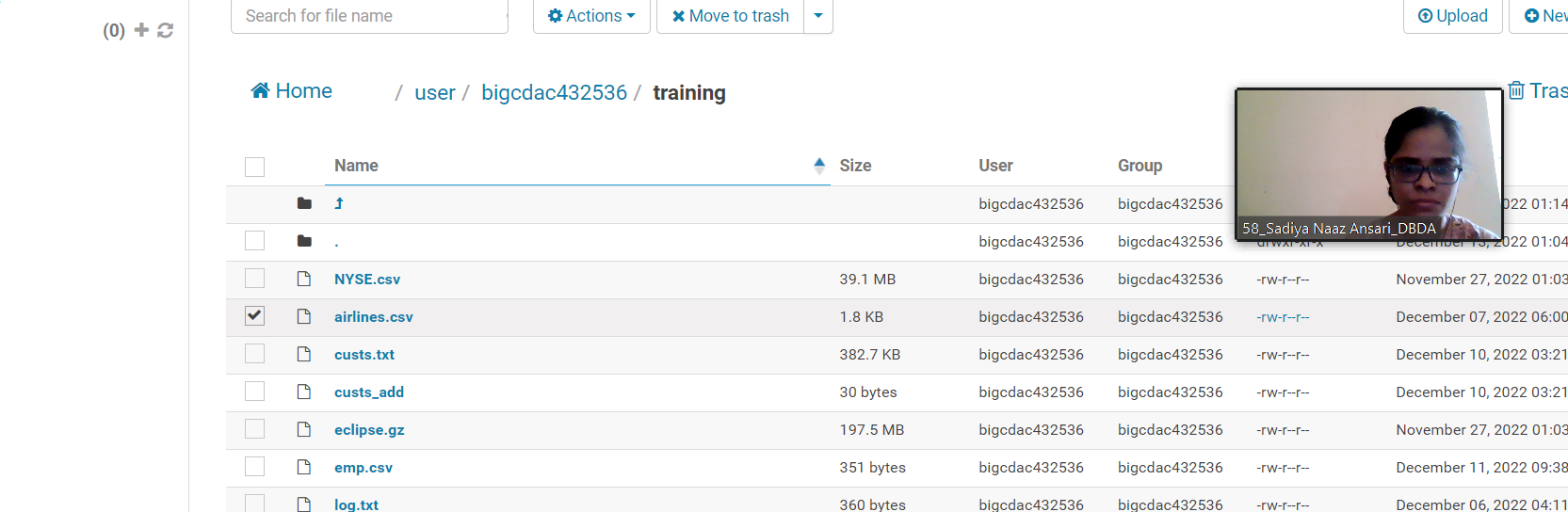
****

**Q.3**

**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")**

****

****

****

**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**

**package cdac;**

**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,IntWritable>**

**{**

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

**{**

**try{**

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

**int high = Integer.parseInt(str[4]);**

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

**}**

**catch(Exception e)**

**{**

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

**}**

**}**

**}**

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

**{**

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

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

**double max = 0.00;**

**for (IntWritable 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", "134217728");**

**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(IntWritable.class);**

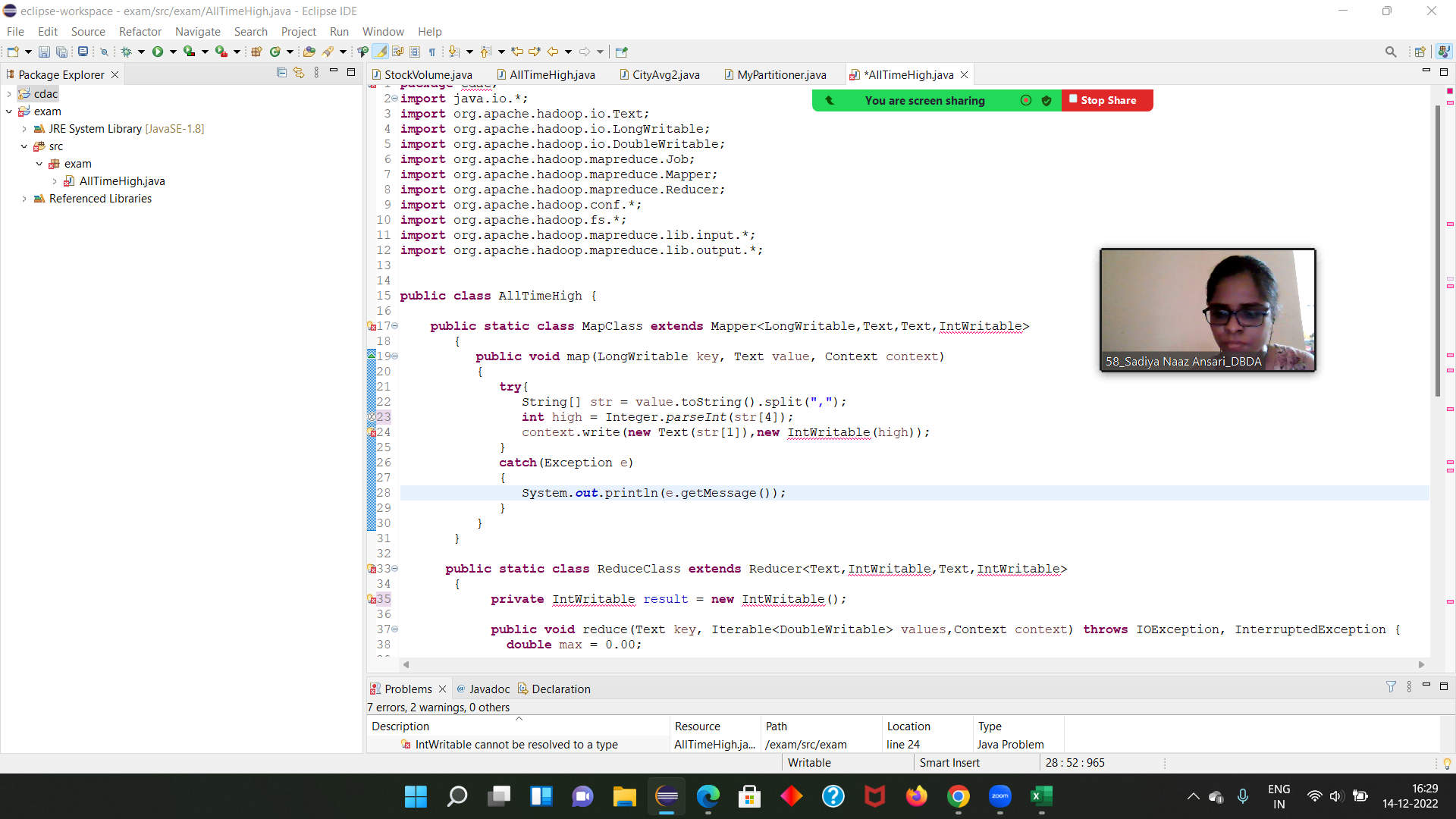
**FileInputFormat.addInputPath(job, new Path(args[0]));**

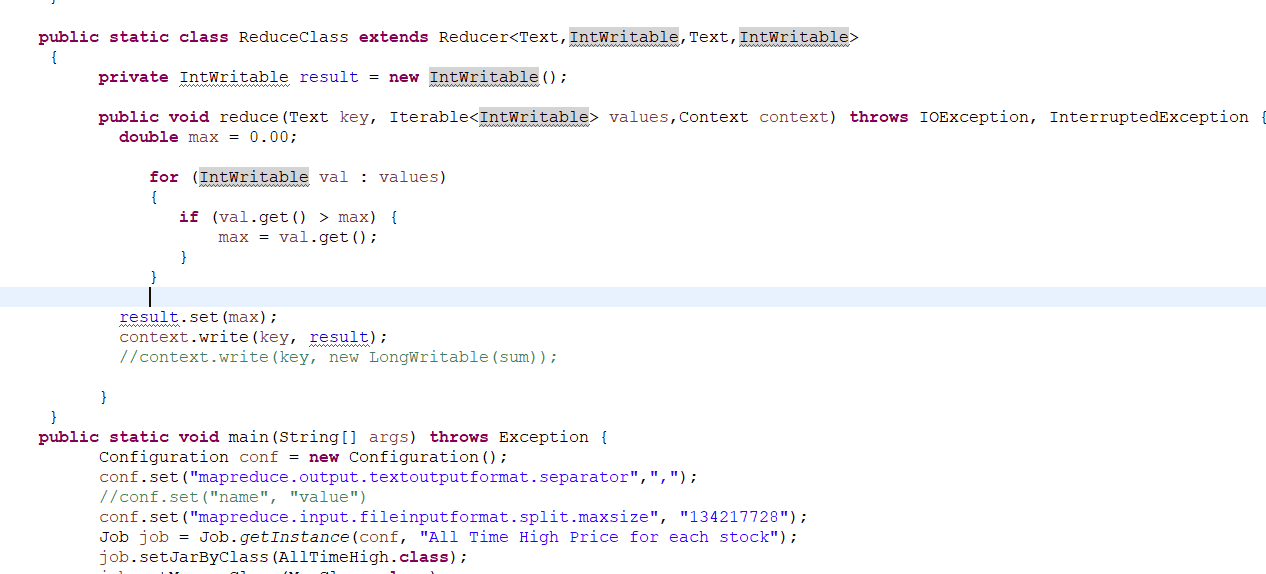
**FileOutputFormat.setOutputPath(job, new Path(args[1]));**

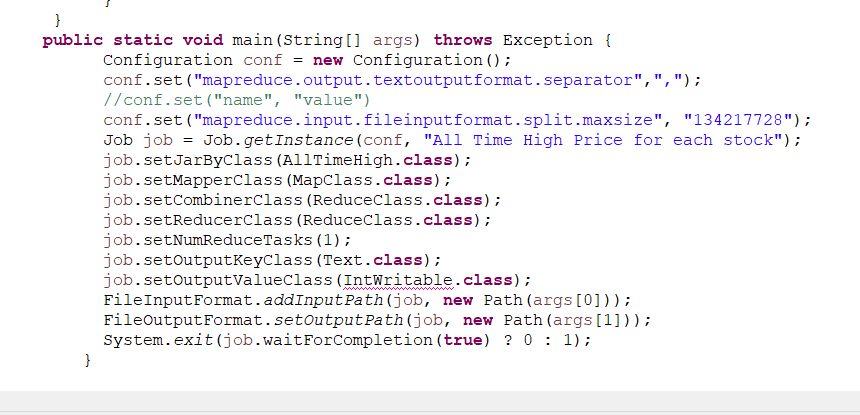
**System.exit(job.waitForCompletion(true) ? 0 : 1);**

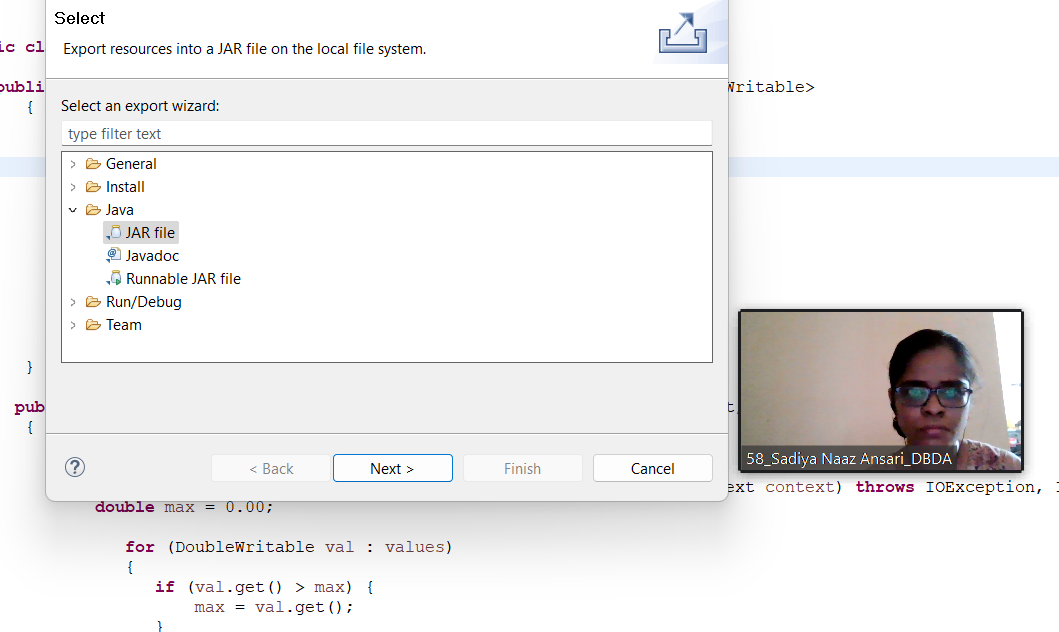
**}**

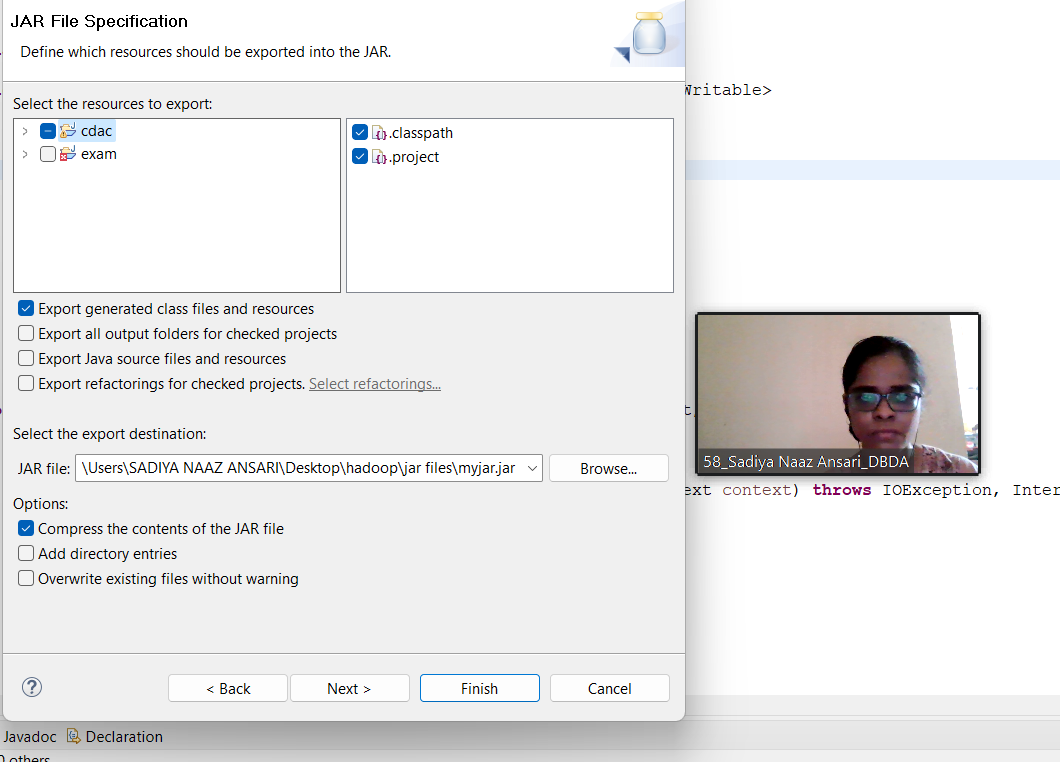
**}**

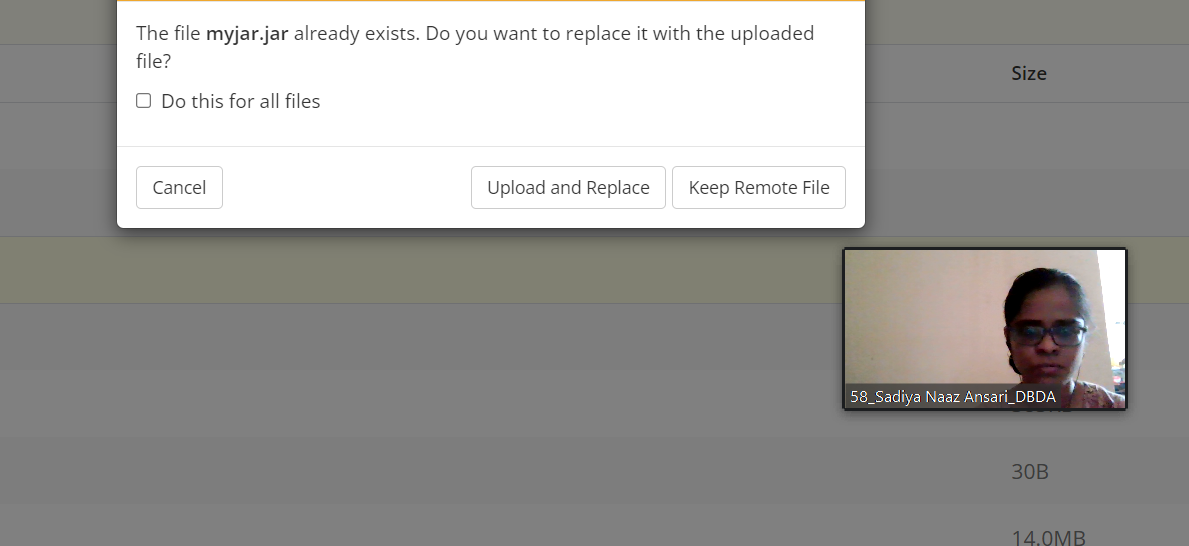
****

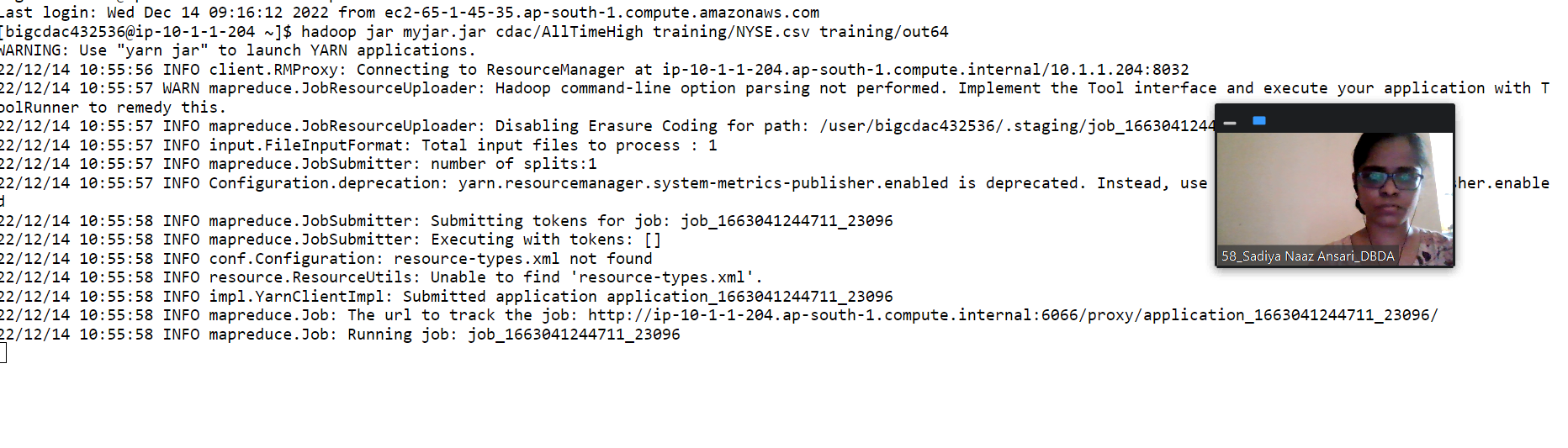
****

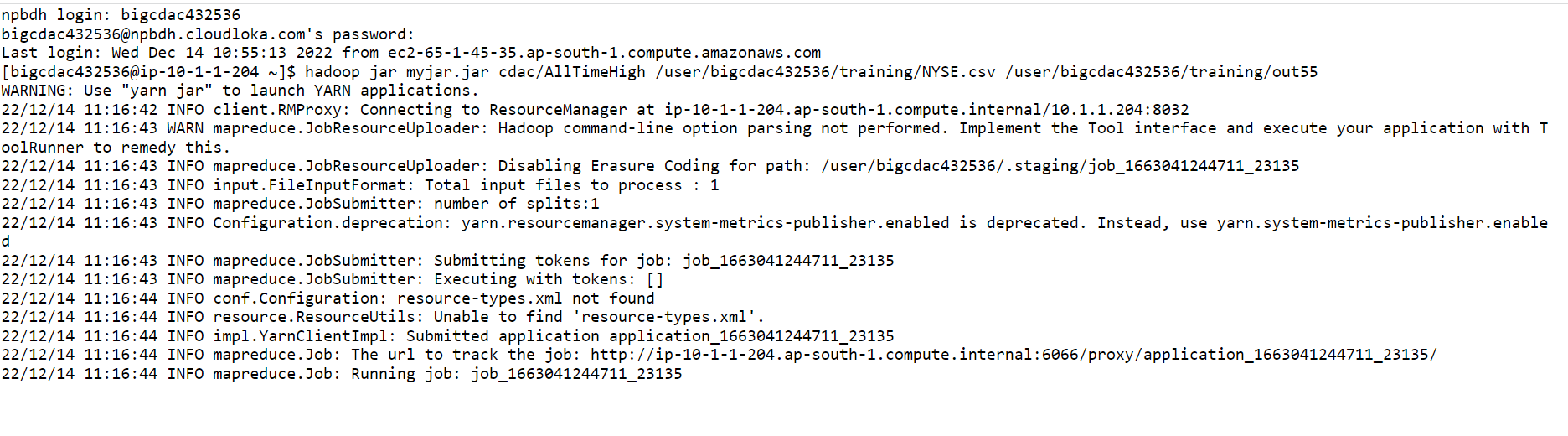
****

****

****

****

****

****

**Got stuck here……**