220940325013\_Anushka\_Umbre\_Dbda

Big Data Module End Exam

**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

**Solution:**

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>

{

private Text stock\_id = new Text();

private DoubleWritable High =new DoubleWritable();

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

{

try{

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

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

stock\_id.set(str[1]);

High.set(high);

context.write(stock\_id, 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 maxValue = 0;

double temp\_val=0;

for (DoubleWritable value : values)

{

temp\_val=value.get();

if (temp\_val > maxValue) {

maxValue = temp\_val;

}

}

result.set(maxValue);

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);

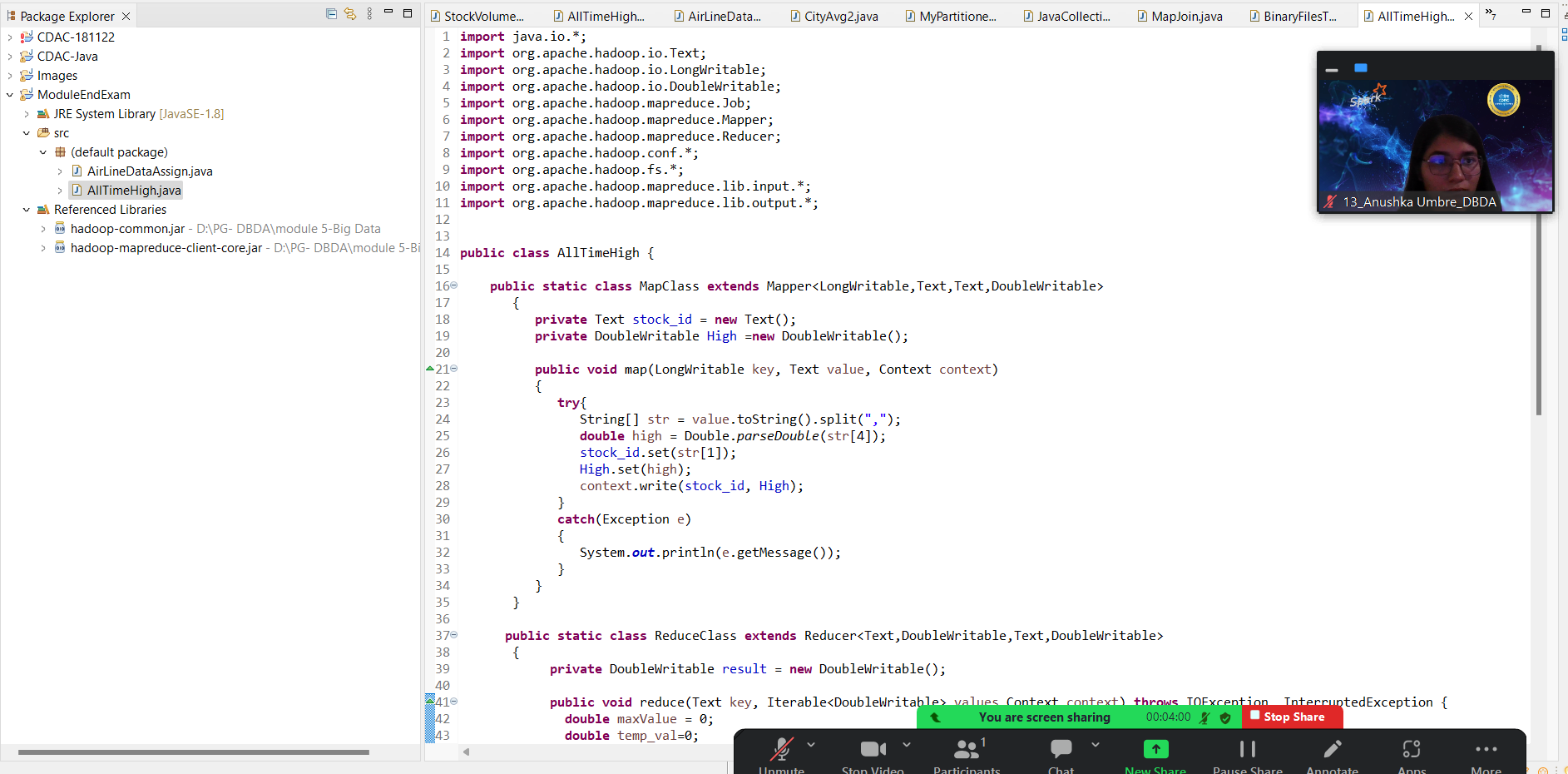
FileInputFormat.addInputPath(job, new Path(args[0]));

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

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

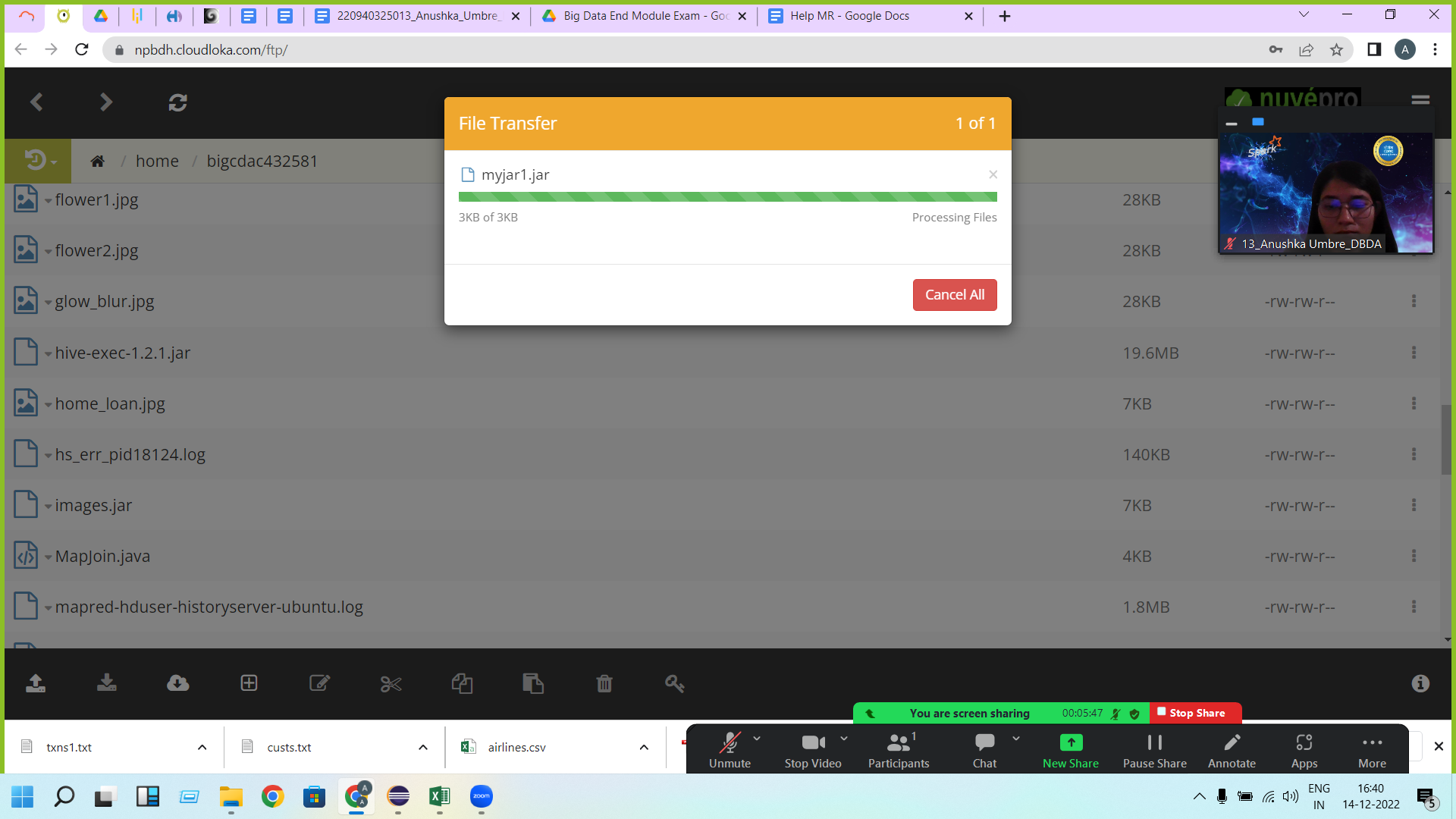
}

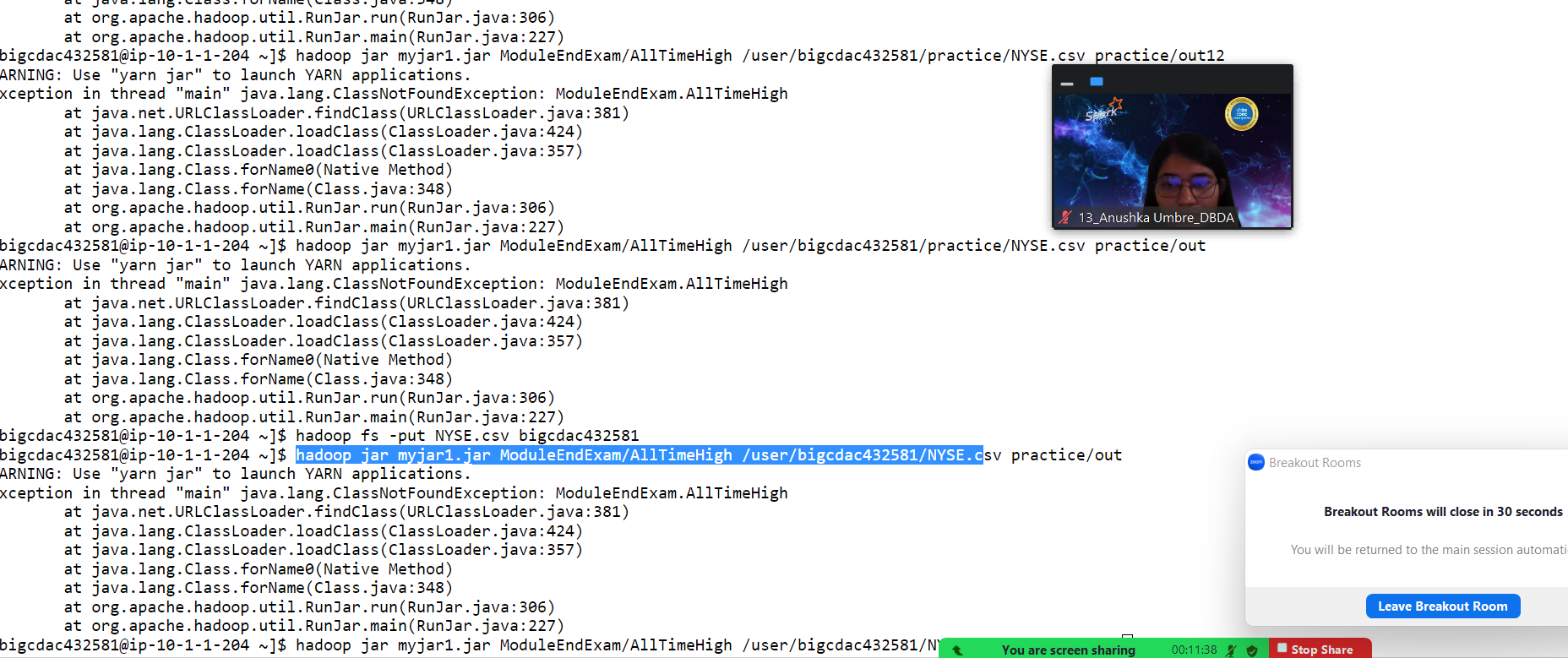
}











**Question 2 : Find all time High price for each stock [15 marks]**

**Hive**

Please find the customer data set.

cust id

firstname

lastname

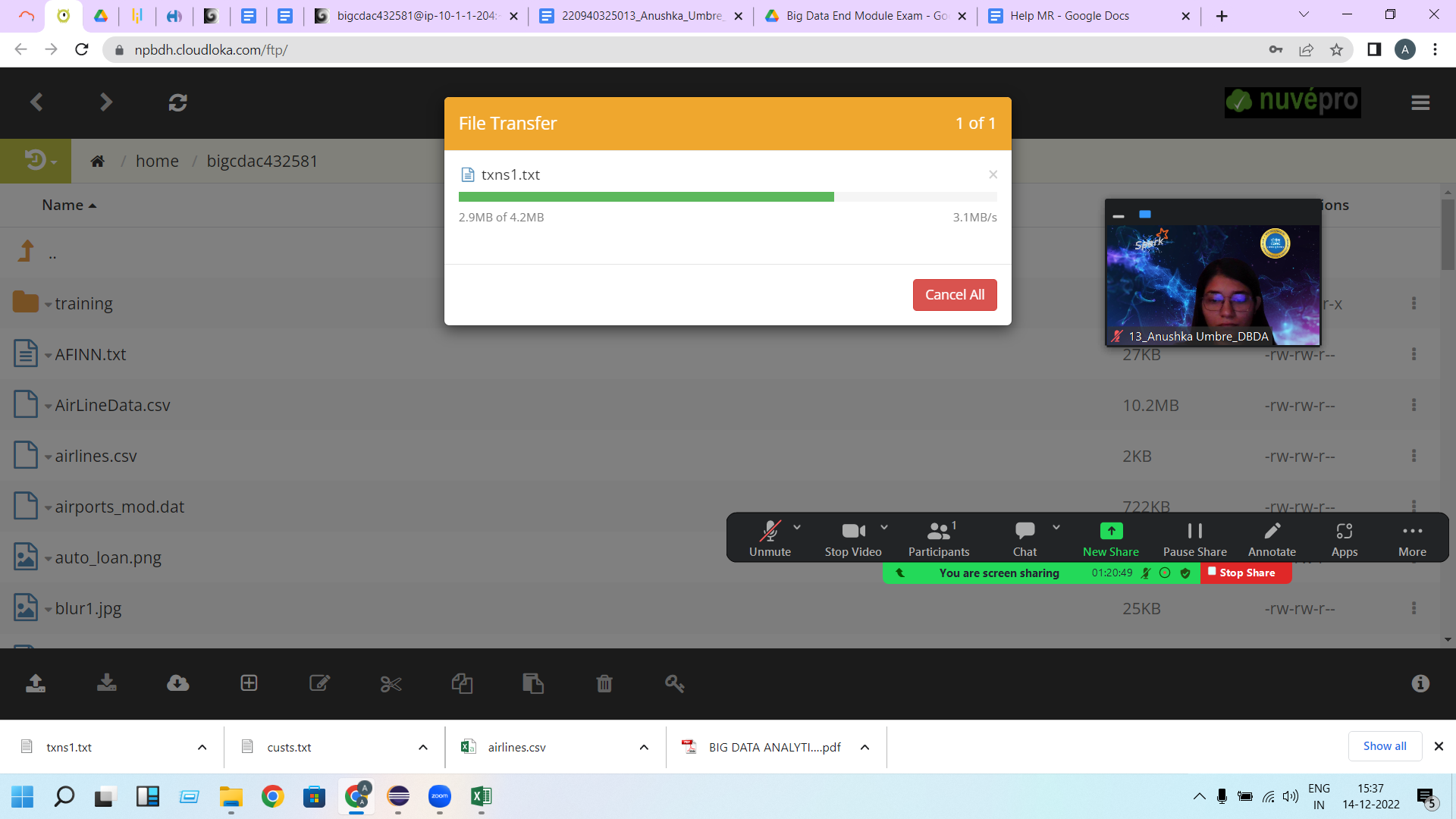
age

profession

[bigcdac432581@ip-10-1-1-204 ~]$ hadoop fs -put custs.txt practice

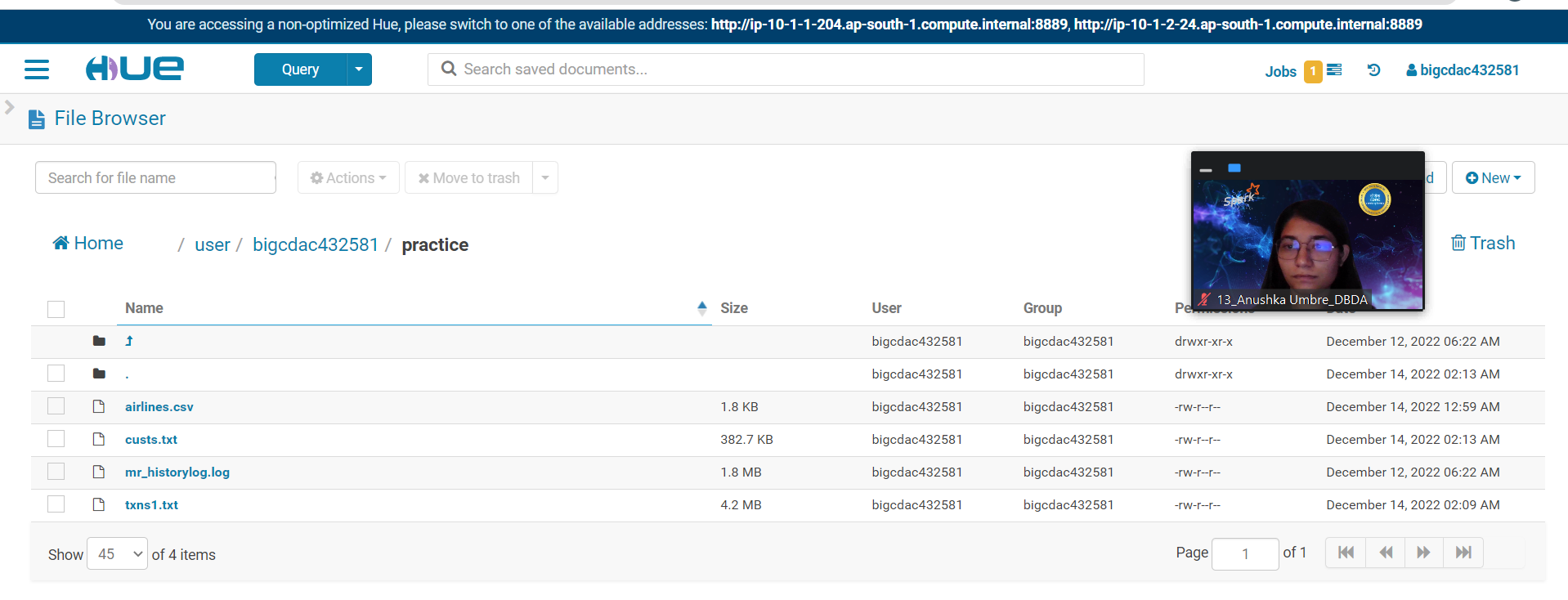
[bigcdac432581@ip-10-1-1-204 ~]$ hive

hive> create table cust13(cust\_id int, first\_name string, last\_name string, age int, profession string) row format delimited fields terminated by ',' stored as textfile;



hive> load data inpath '/user/bigcdac432581/practice/custs.txt' overwrite into table cust13;

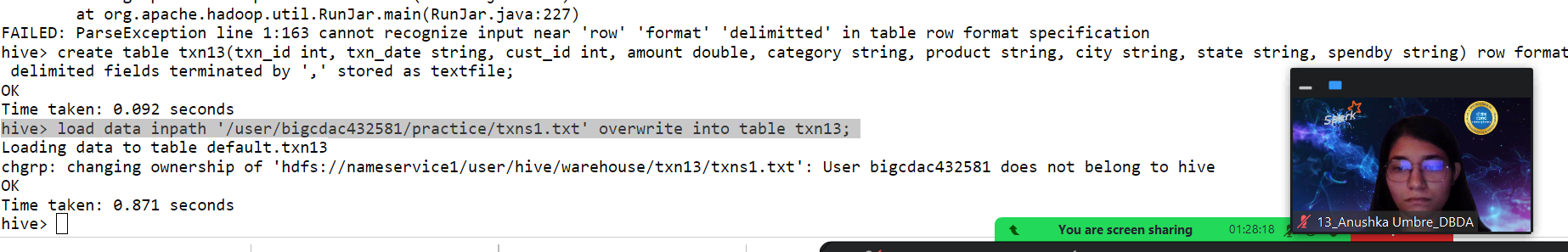
[bigcdac432581@ip-10-1-1-204 ~]$ hadoop fs -put txns1.txt practice



hive> create table txn13(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;

hive> load data inpath '/user/bigcdac432581/practice/txns1.txt' overwrite into table txn13;



hive> select \* from txn13 limit 10;

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

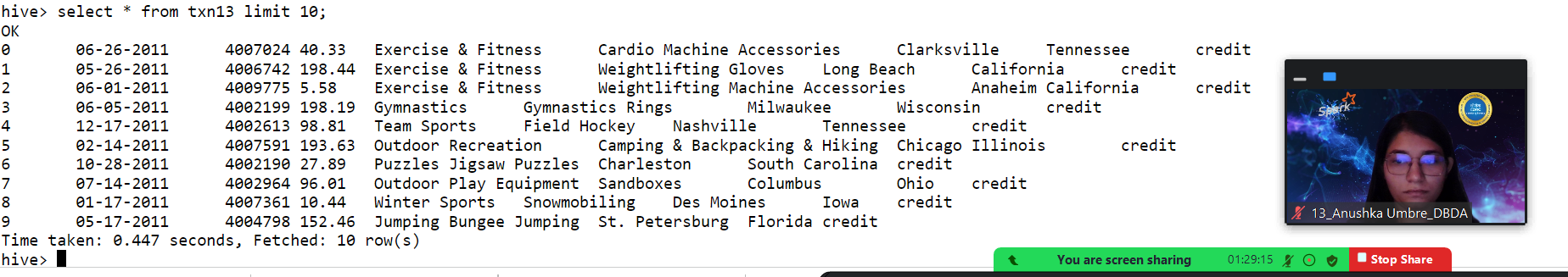
5 02-14-2011 4007591 193.63 Outdoor Recreation Camping & Backpacking & Hiking Chicago Illinois credit

6 10-28-2011 4002190 27.89 Puzzles Jigsaw Puzzles Charleston South Carolina credit

7 07-14-2011 4002964 96.01 Outdoor Play Equipment Sandboxes Columbus Ohio credit

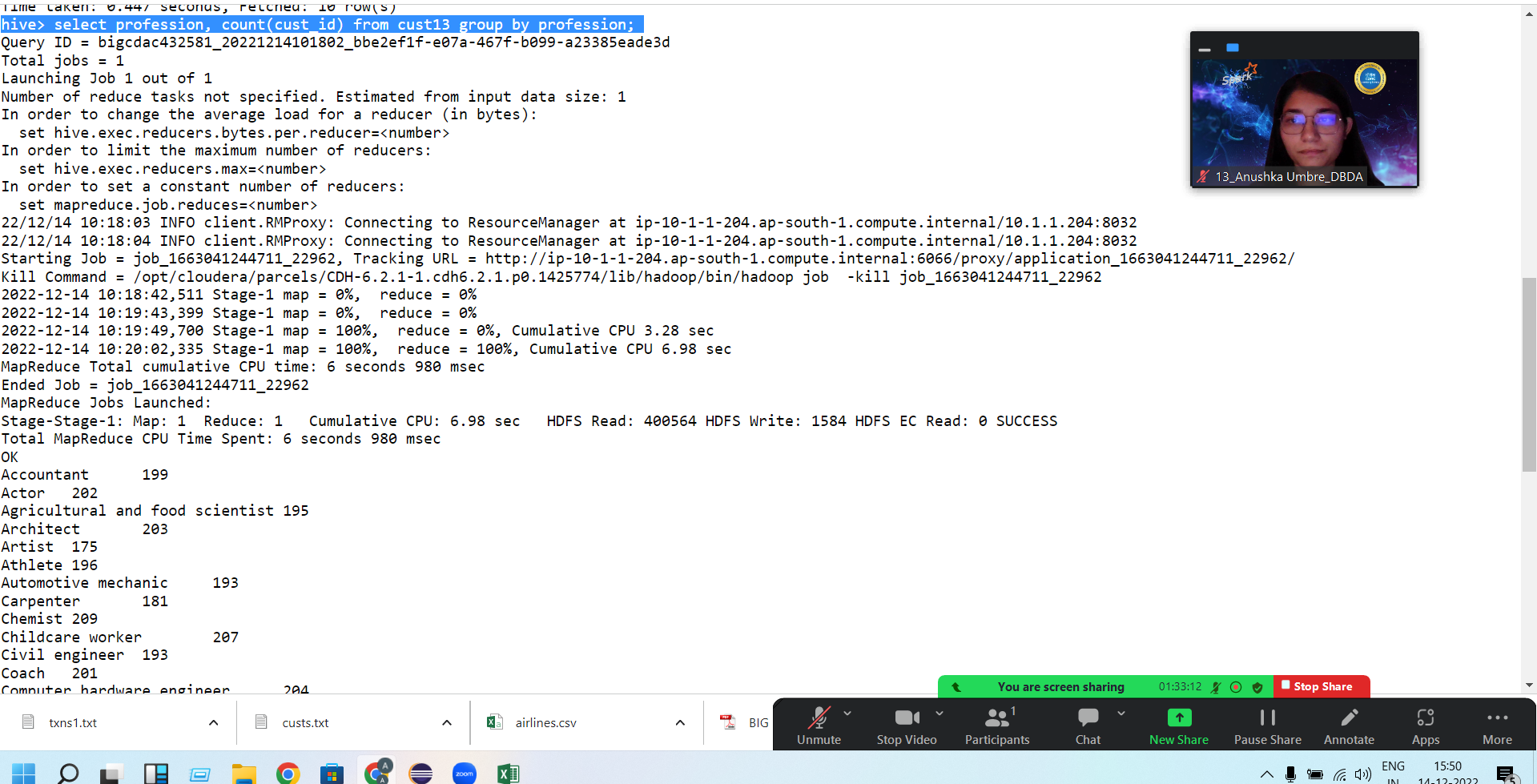
8 01-17-2011 4007361 10.44 Winter Sports Snowmobiling Des Moines Iowa credit

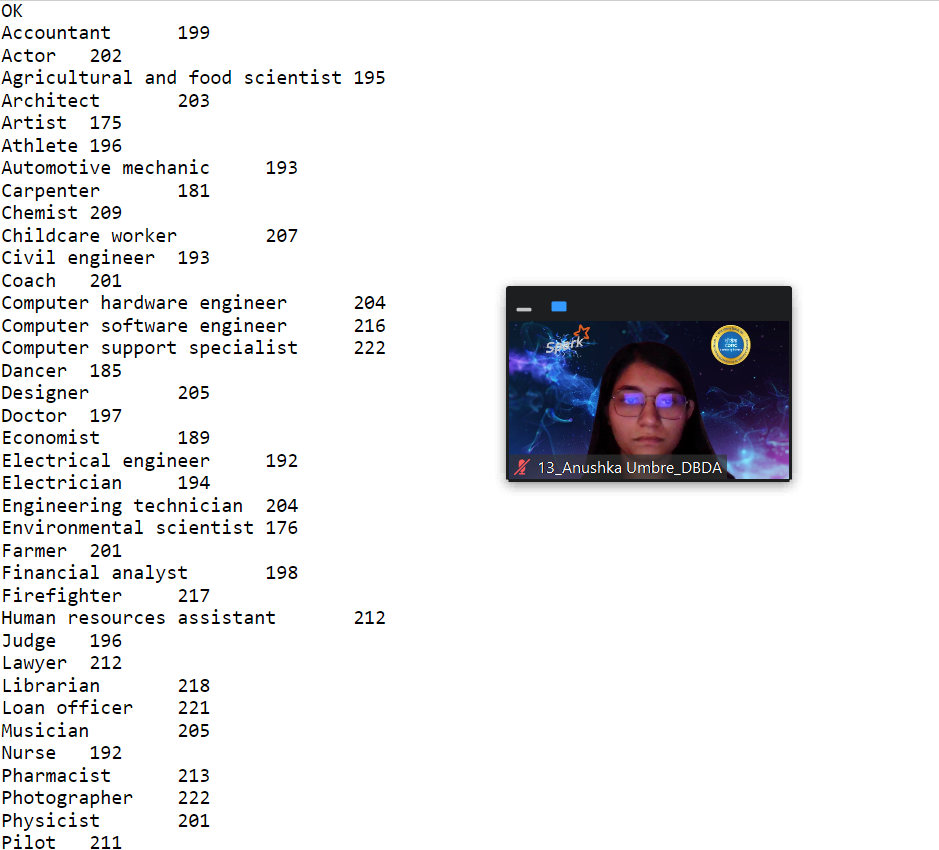
9 05-17-2011 4004798 152.46 Jumping Bungee Jumping St. Petersburg Florida credit

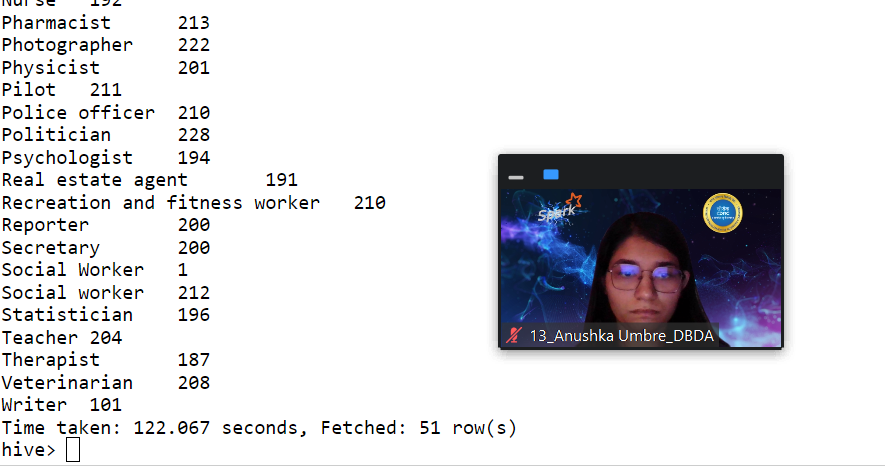


**1) Write a program to find the count of customers for each profession.**

hive> select profession, count(cust\_id) from cust13 group by profession;

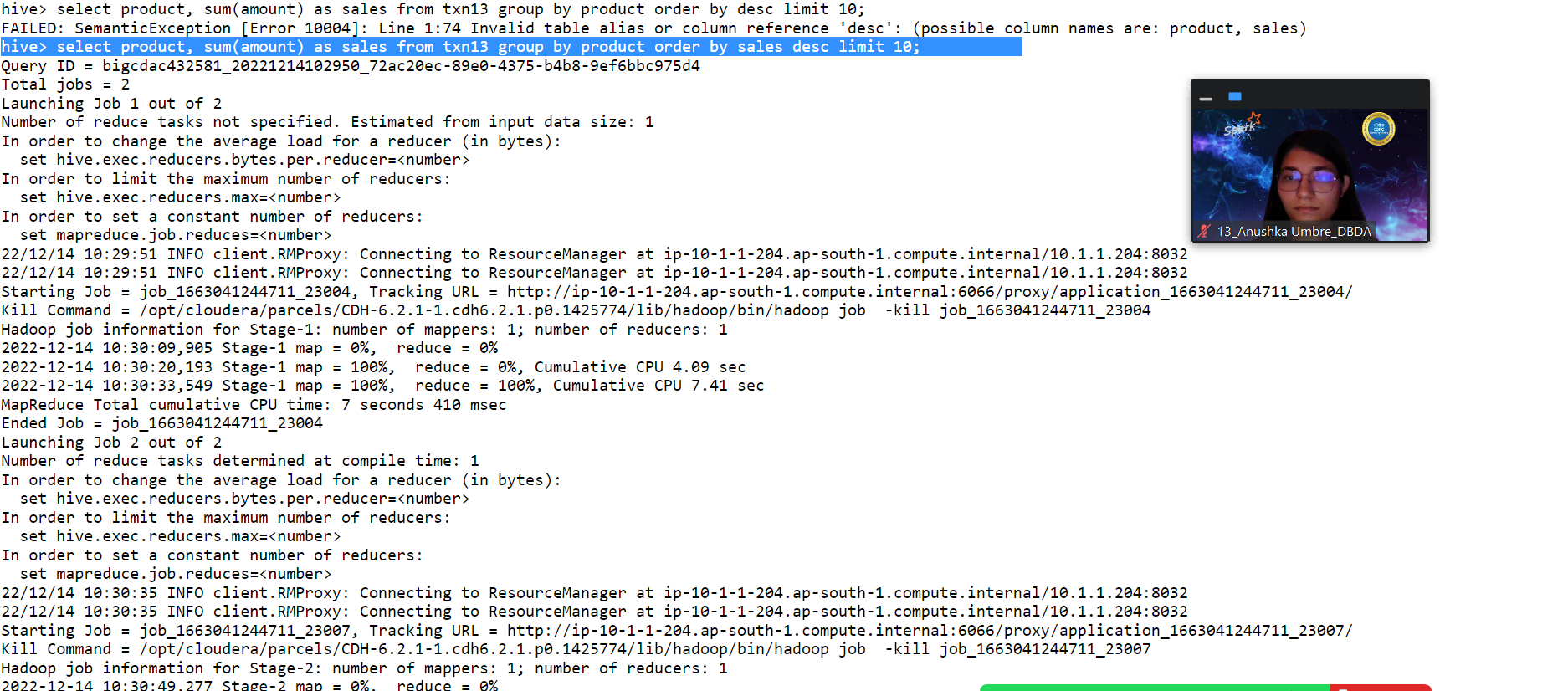






**2) Write a program to find the top 10 products sales wise**

**hive> select product, sum(amount) as sales from txn13 group by product order by sales desc limit 10;**

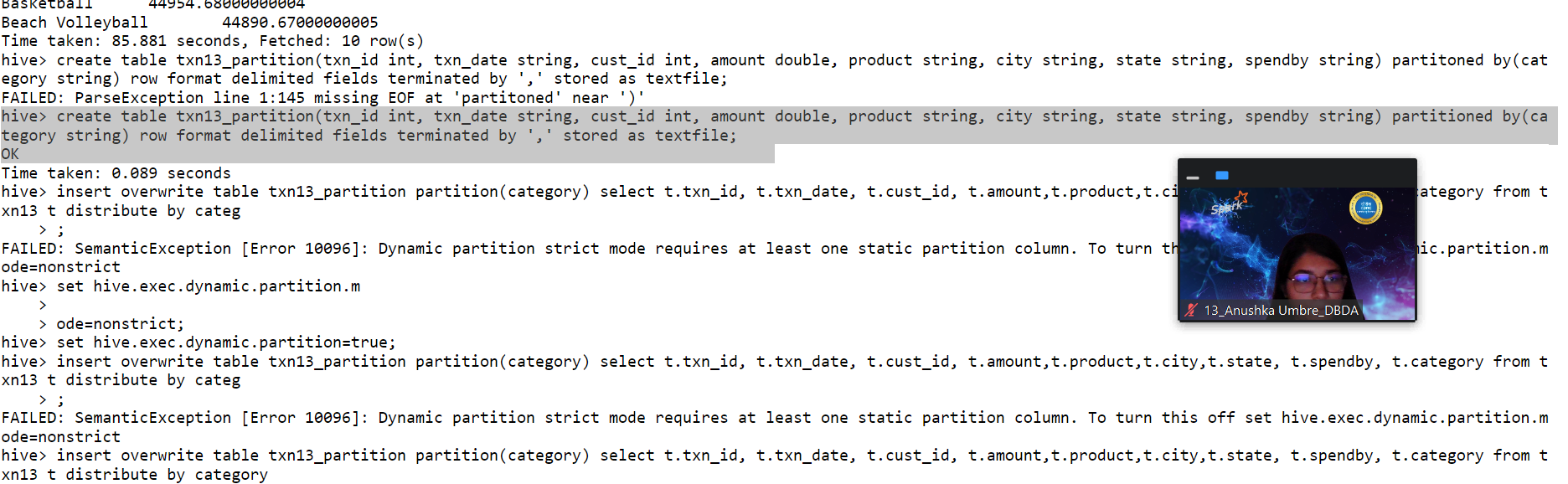
****

****

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

**hive> create table txn13\_partition(txn\_id int, txn\_date string, cust\_id int, amount double, product string, city string, state string, spendby string) partitioned by(ca**

**tegory string) row format delimited fields terminated by ',' stored as textfile;**

****

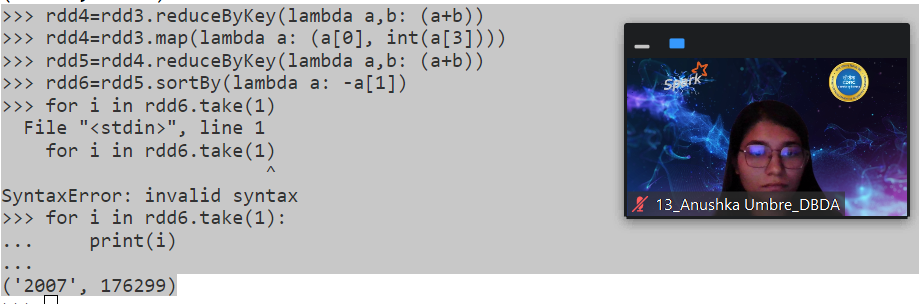
**QUESTION 3 [15 marks]**

**PySpark**

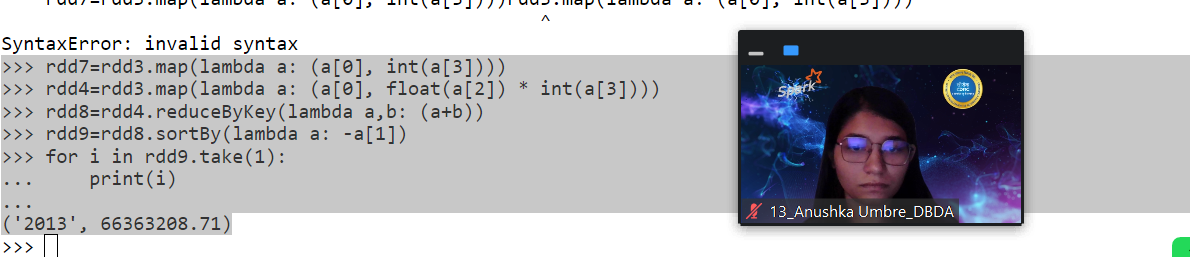
**1) What was the highest number of people travelled in which**

**year?**

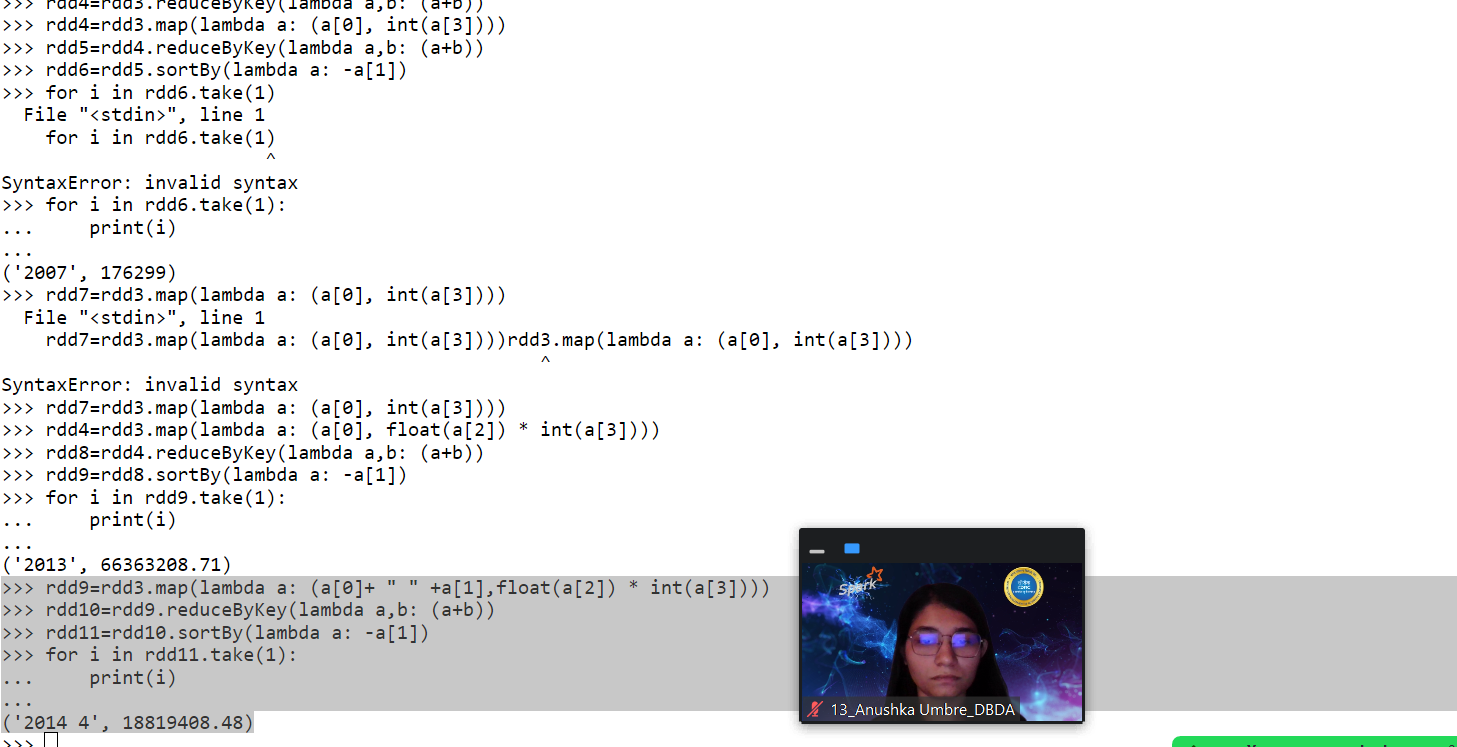
**Answer:**



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

****

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

****

Solution:

[bigcdac432581@ip-10-1-1-204 ~]$ hadoop fs -put airlines.csv practice

>>> dataRdd=sc.textFile("hdfs://nameservice1/user/bigcdac432581/practice/airlines.csv",1)

>>> dataRdd.first()



>>> header=dataRdd.first()

>>> print(header)

>>> rdd2=dataRdd.filter(lambda a: a!= header)

>>> for a in rdd2.take(5):

... print(a)

...

1995,1,296.9,46561

1995,2,296.8,37443

1995,3,287.51,34128

1995,4,287.78,30388

1996,1,283.97,47808

>>> rdd3=rdd2.map(lambda a:a.split(","))

>>> for a in rdd3.take(5):

... print(a)

...

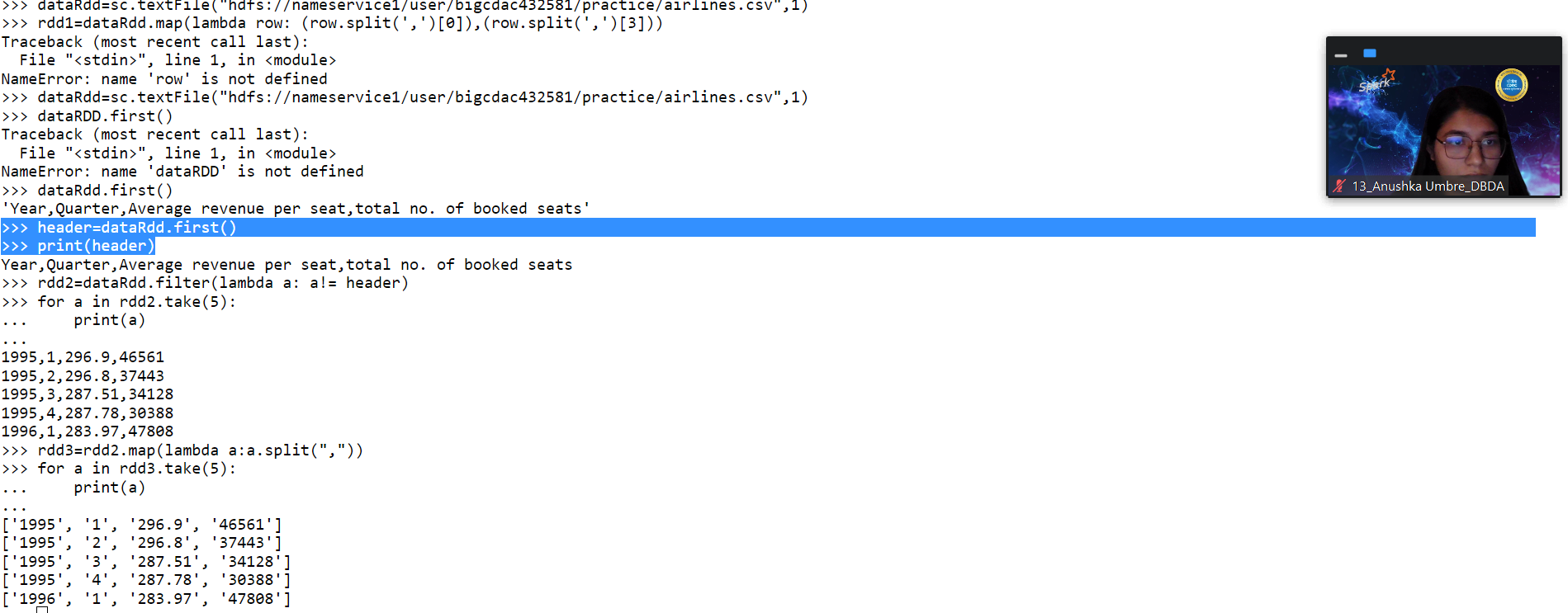
['1995', '1', '296.9', '46561']

['1995', '2', '296.8', '37443']

['1995', '3', '287.51', '34128']

['1995', '4', '287.78', '30388']

['1996', '1', '283.97', '47808']



>>> rdd4=rdd3.map(lambda a: (a[0],int(a[3])))

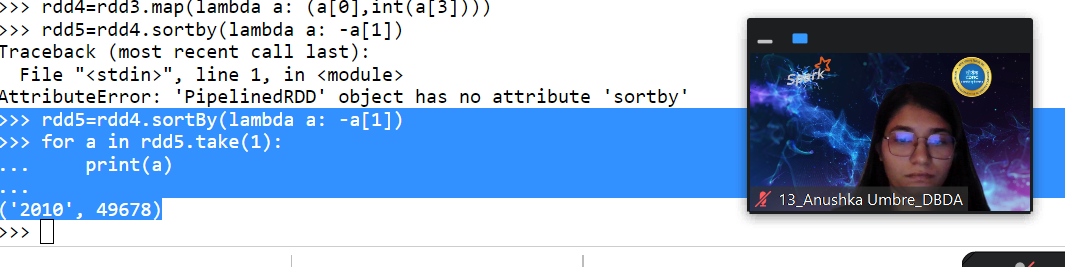
>>> rdd5=rdd4.sortBy(lambda a: -a[1])

>>> for a in rdd5.take(1):

... print(a)

...

('2010', 49678)



>>> rdd4=rdd3.reduceByKey(lambda a,b: (a+b))

>>> rdd4=rdd3.map(lambda a: (a[0], int(a[3])))

>>> rdd5=rdd4.reduceByKey(lambda a,b: (a+b))

>>> rdd6=rdd5.sortBy(lambda a: -a[1])

>>> for i in rdd6.take(1):

... print(i)

...

('2007', 176299)

>>> rdd7=rdd3.map(lambda a: (a[0], int(a[3])))

>>> rdd4=rdd3.map(lambda a: (a[0], float(a[2]) \* int(a[3])))

>>> rdd8=rdd4.reduceByKey(lambda a,b: (a+b))

>>> rdd9=rdd8.sortBy(lambda a: -a[1])

>>> for i in rdd9.take(1):

... print(i)

...

('2013', 66363208.71)

>>> rdd9=rdd3.map(lambda a: (a[0]+ " " +a[1],float(a[2]) \* int(a[3])))

>>> rdd10=rdd9.reduceByKey(lambda a,b: (a+b))

>>> rdd11=rdd10.sortBy(lambda a: -a[1])

>>> for i in rdd11.take(1):

... print(i)

...

('2014 4', 18819408.48)