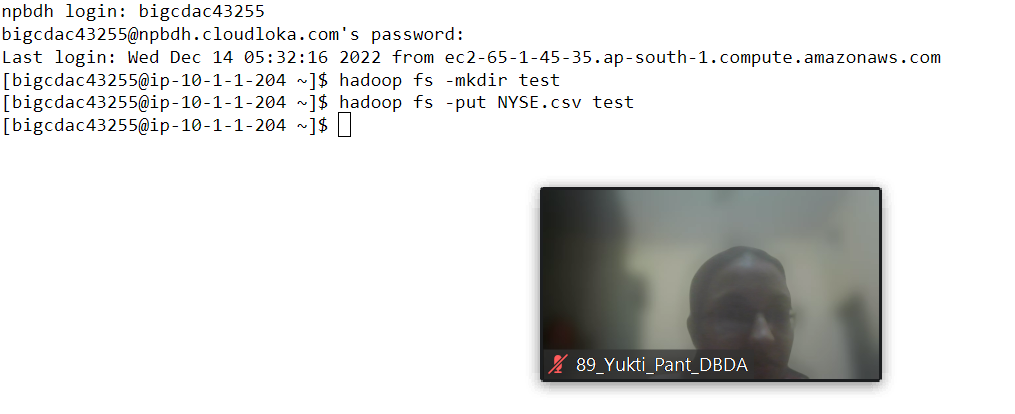
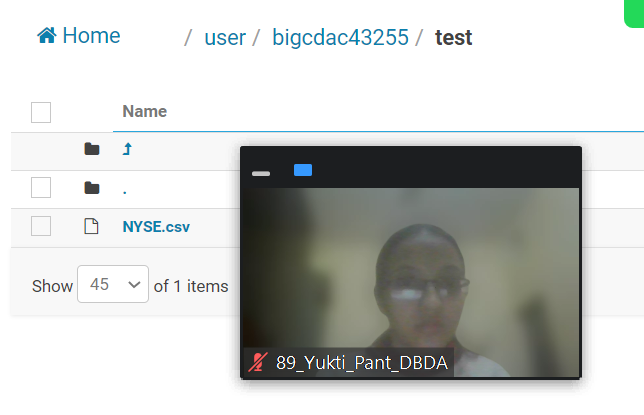
Map Reduce

1.





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.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 AllTimeHigh {

public static class MapClass extends Mapper<LongWritable, Text, Text, IntWritable>{

private void map(LongWritable Key, Text Value,Context context) {

try {

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

long vol=Long.parseLong(str[7]);

context.write(next Text(str[1]),new LongWritable(vol));

}

catch(Exception e){

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

}

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

private LongWritable result=new LongWritable();

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

{

long max=0;

for(LongWritable val:Value)

{

if(val.max()>max) {

max=val.get()

}

result.set(sum);

Content.write(Key,new LongWritable (sum);

}

}

}

}

}

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

Configuration conf = new Configuration();

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

job.setJarByClass(AllTimeHigh.class);

job.setMapperClass(MapClass.class);

job.setReducerClass(ReduceClass.class);

job.setNumReduceTasks(1);

job.setMapOutputKeyClass(Text.class);

job.setMapOutputValueClass(IntWritable.class);

job.setOutputKeyClass(Text.class);

job.setOutputValueClass(LongWritable.class);

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

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

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

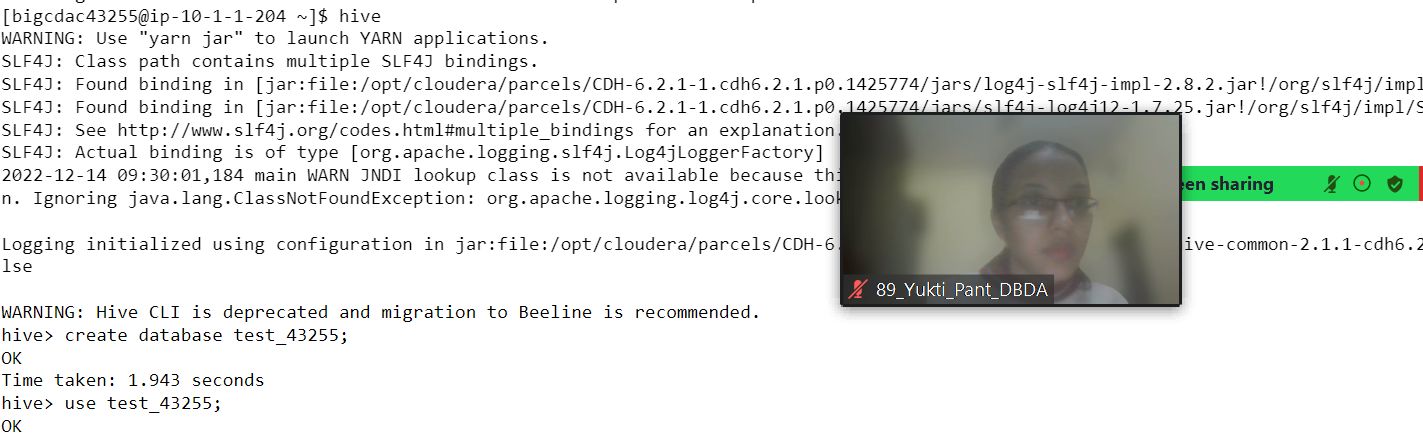
}

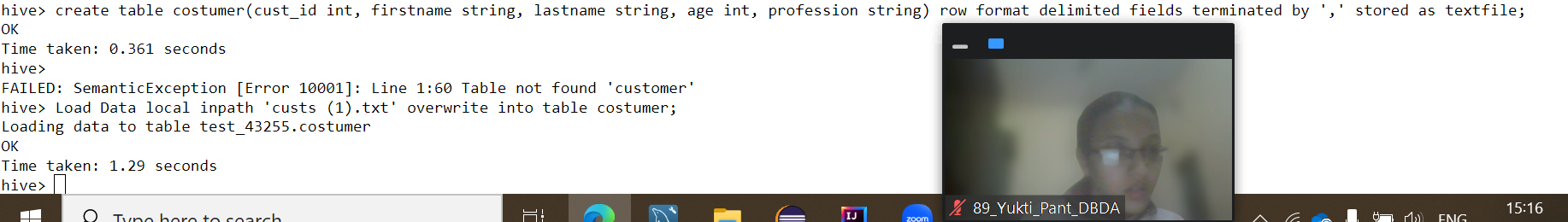
}

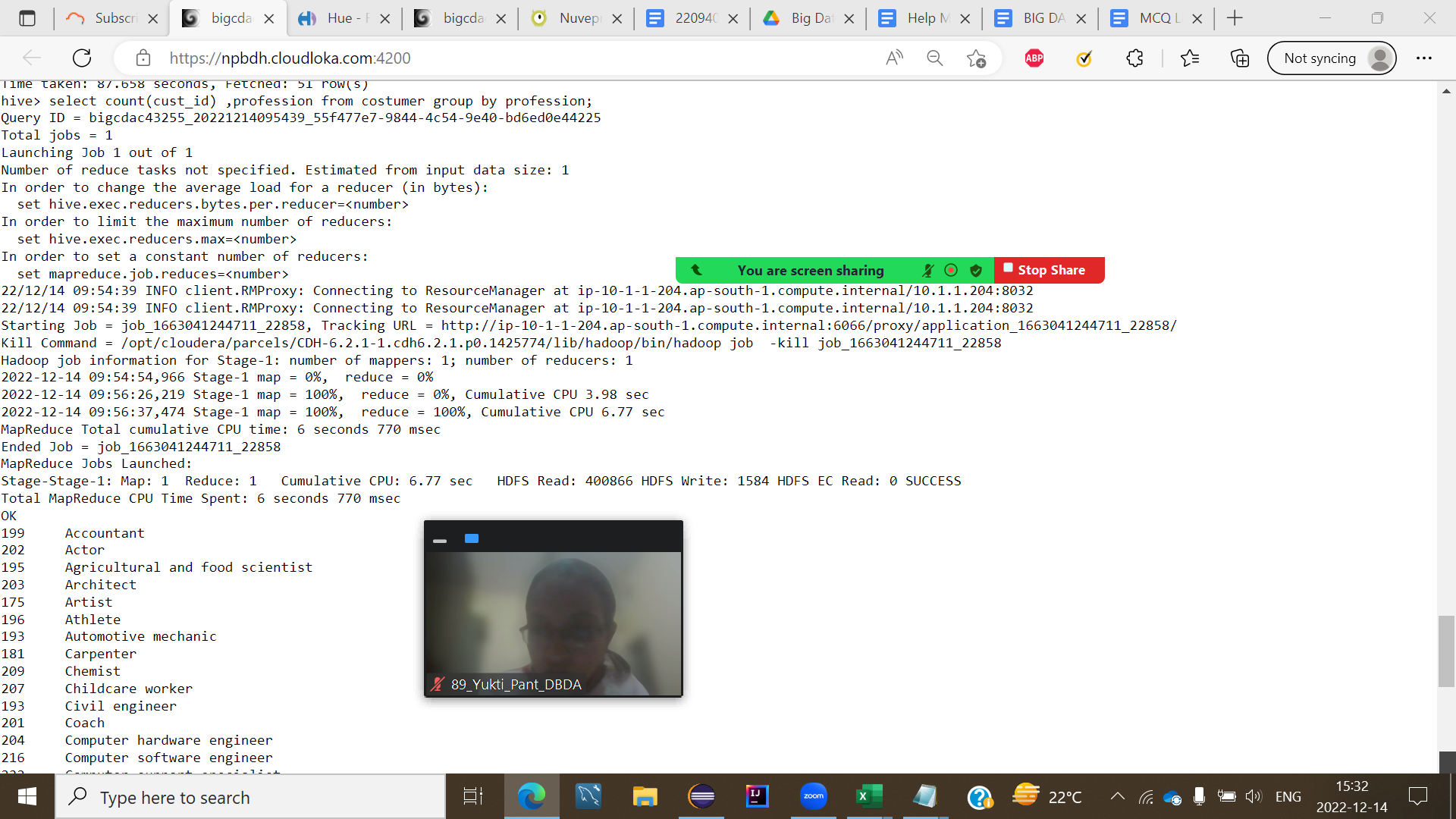
}

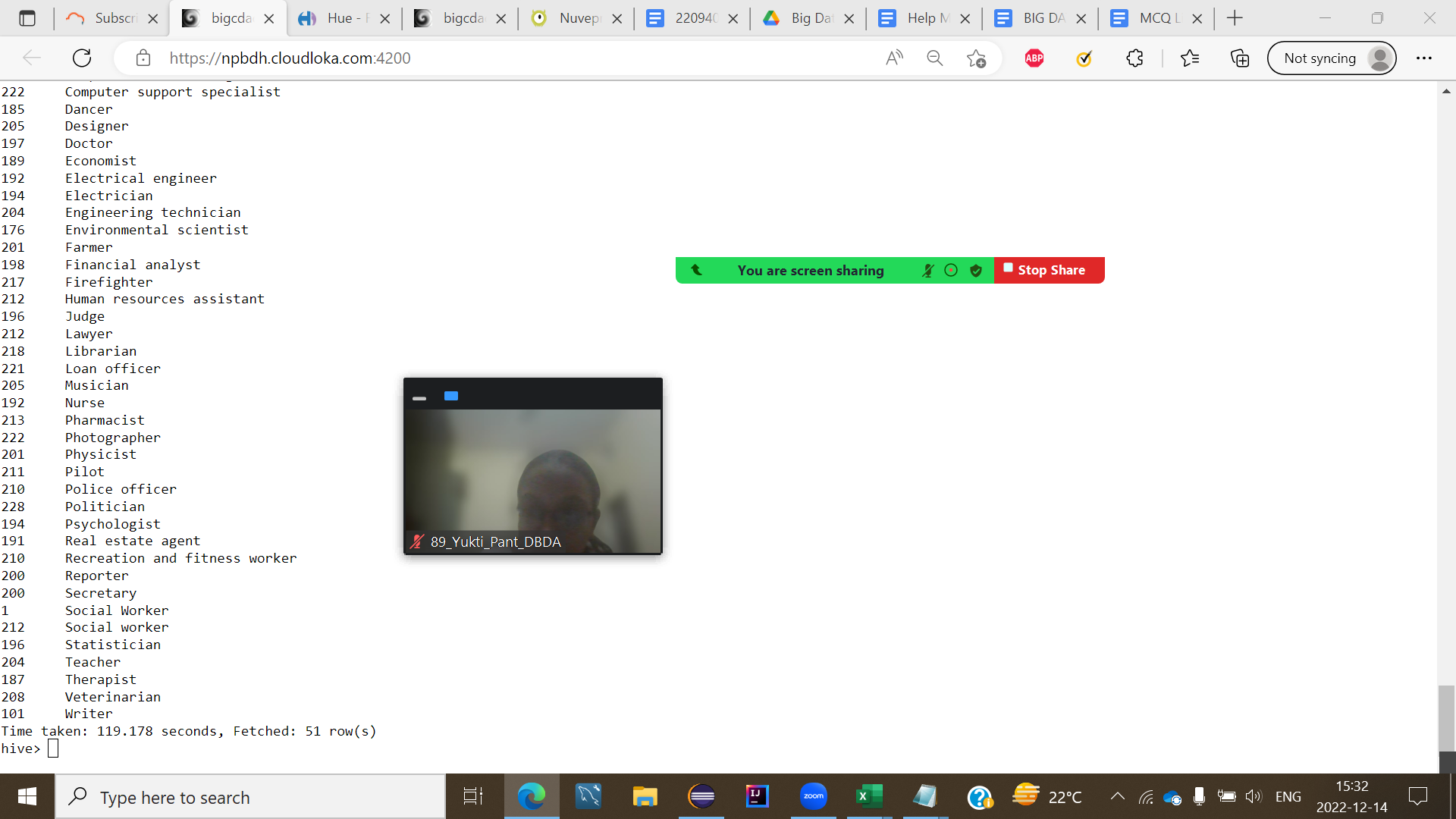
Hive

1.









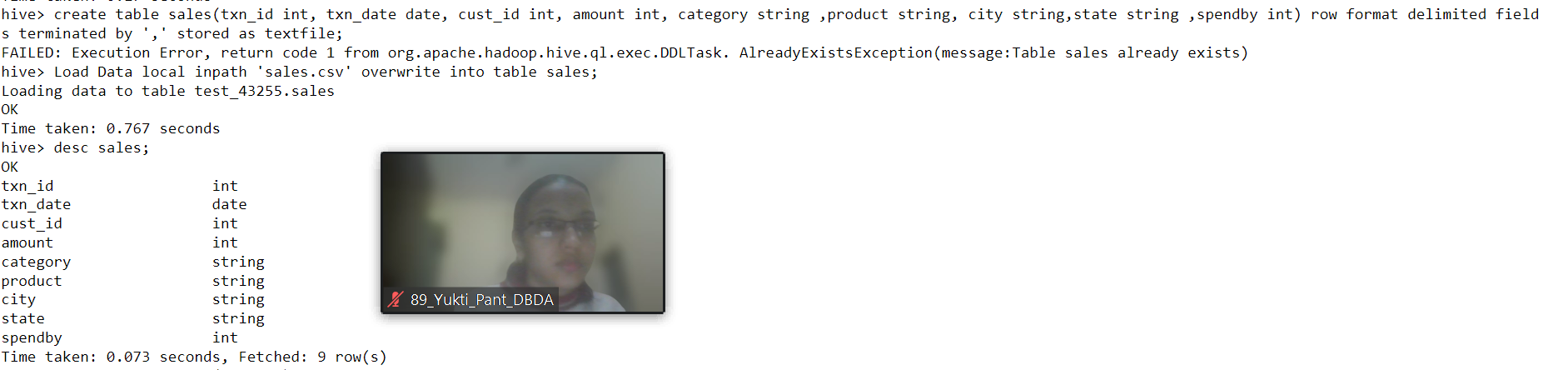
create table costumer(cust\_id int, firstname string, lastname string, age int, profession string) row format delimited fields terminated by ',' stored as textfile;

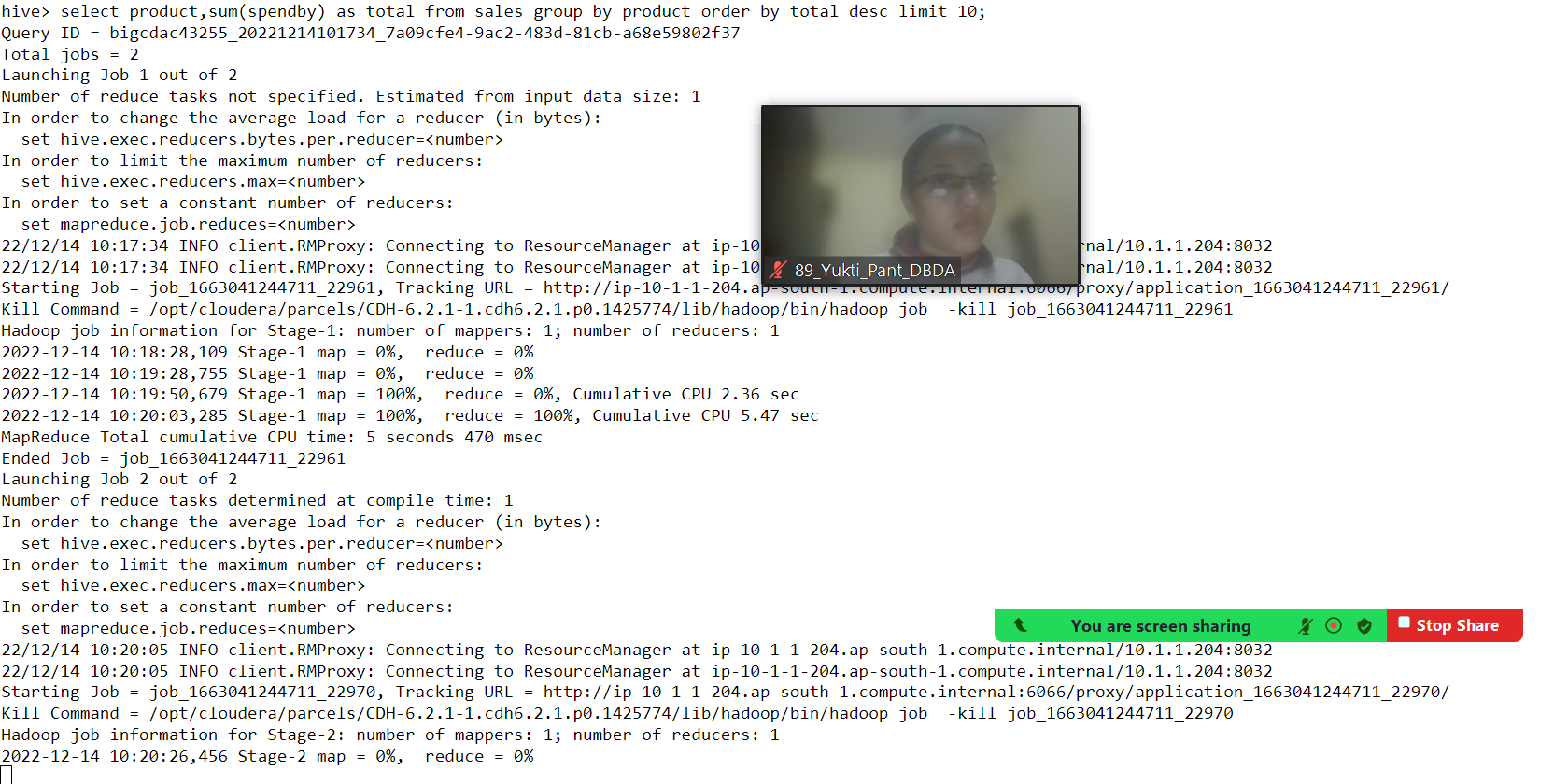
Load Data local inpath 'custs (1).txt' overwrite into table costumer;

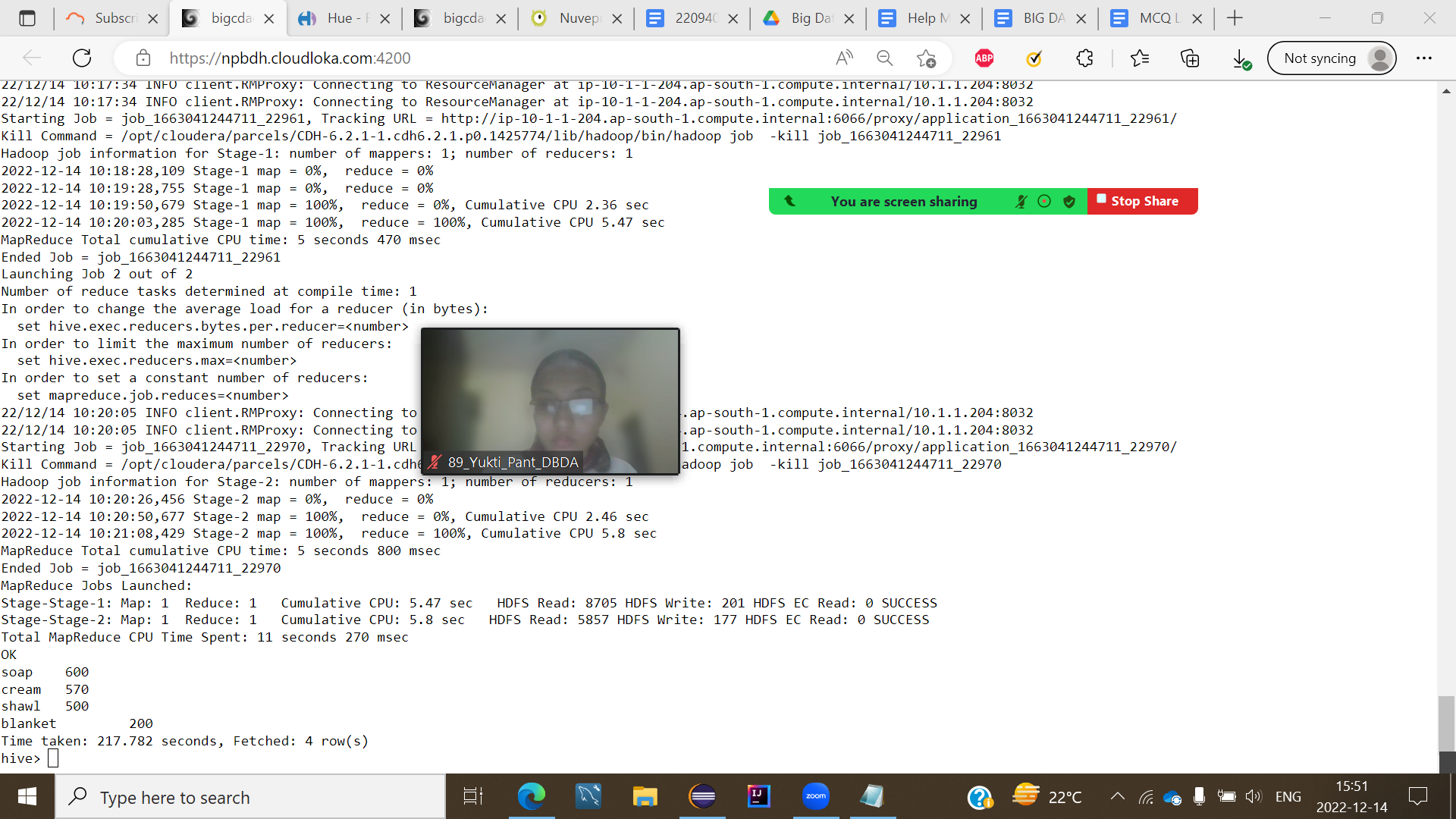
Select count(cust\_id),profession from costumer group by profession;

Sales dataset

2.







3.

create table sales(txn\_id int, txn\_date date, cust\_id int, amount int, category string ,product string, city string,state string ,spendby int) row format delimited fields terminated by ',' stored as textfile;

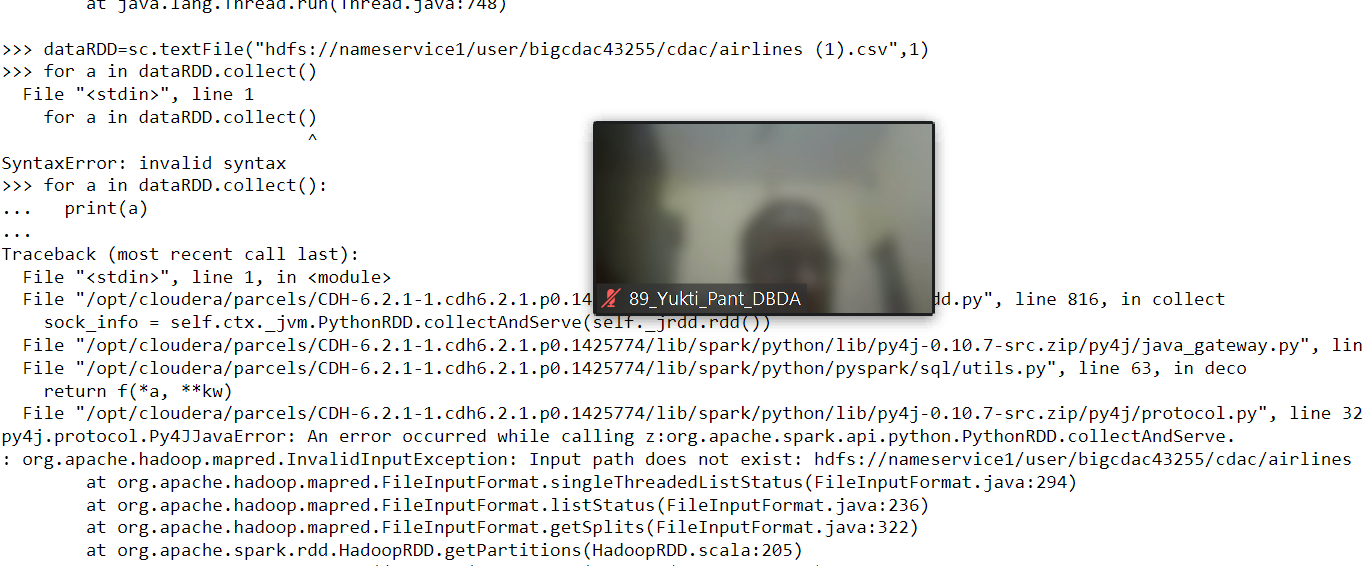
Load Data local inpath 'sales.csv' overwrite into table sales;

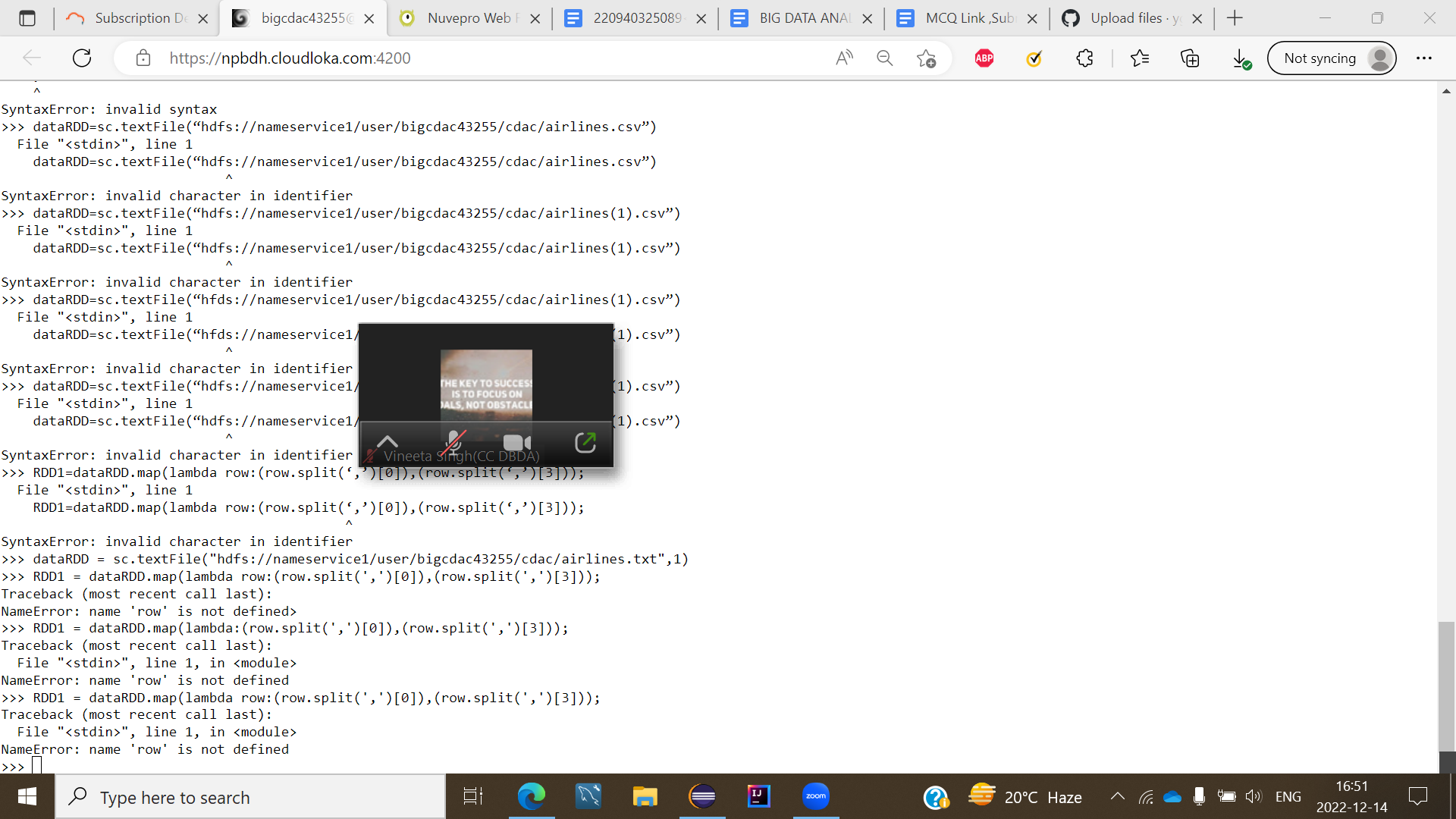
select product,sum(spendby) as total from sales group by product order by total desc limit 10;

create table sales partitioned (txn\_id int, txn\_date date, cust\_id int, amount int, category string ,product string, city string,state string ,spendby int)

partitioned by (category string)row format delimited fields terminated by ',' stored as textfile;

Pyspark(got stuck)





1.

dataRDD = sc.textFile("hdfs://nameservice1/user/bigcdac43255/cdac/airlines.txt",1)

for row in dataRDD.collect():

print(row)

RDD1 = dataRDD.map(lambda row:(row.split(',')[0]),(row.split(',')[3]));

RDD2 = RDD1.reduceByKey(lambda a,b : a+b)

select year,sum(Total number of booked seats) as totalSeats from airlines group by year having totalSeats in

(select max(totalSeats) from ((select year,sum(Total number of booked seats) as totalSeats from airlines group by year)abctable));

2.

select year,sum(Average revenue per seat \* Total number of booked seats) as avgRev from airlines group by year order by avgRev desc limit 1;

3.

select year,quarter,sum(Average revenue per seat \* Total number of booked seats) as Rev from airlines group by year,quarter order by Rev desc limit 1;