**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, LongWritable> {**

**public void map(LongWritable key, Text value, Context context) throws IOException, InterruptedException {**

**try {**

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

**Long high = Long.parseLong(str[4]);**

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

**} catch (Exception e) {**

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

**}**

**}**

**}**

**public static class ReduceClass extends Reducer<Text LongWritable,Text,LongWritable> {**

**private Longwritable result = new LongWritable();**

**public void reduce(Text Key,Iterable<LongWritable> Context context) throws IOException, InterruptedException {**

**double high =0;**

**double temp =0;**

**for(Longwritable val : values) {**

**temp = val.get();**

**if (temp > high) {**

**high = temp;**

**}**

**}**

**result.set(high);**

**context.write(key,result)**

**}**

**}**

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

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

**Job job = Job.getInstance(conf, "High Price For Each stock");**

**job.setJarByClass(Alltimehigh);**

**job.setMapperClass(MapClass);**

**job.setReducerClass(ReduceClass);**

**job.setNumReduceTasks(1);**

**job.setMapOutputKeyClass(Text);**

**job.setMapOutputValueClass(Longwritable);**

**job.setOutputKeyClass(Text);**

**job.setOutputValueClass(LongWritable);**

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

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

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

**}**

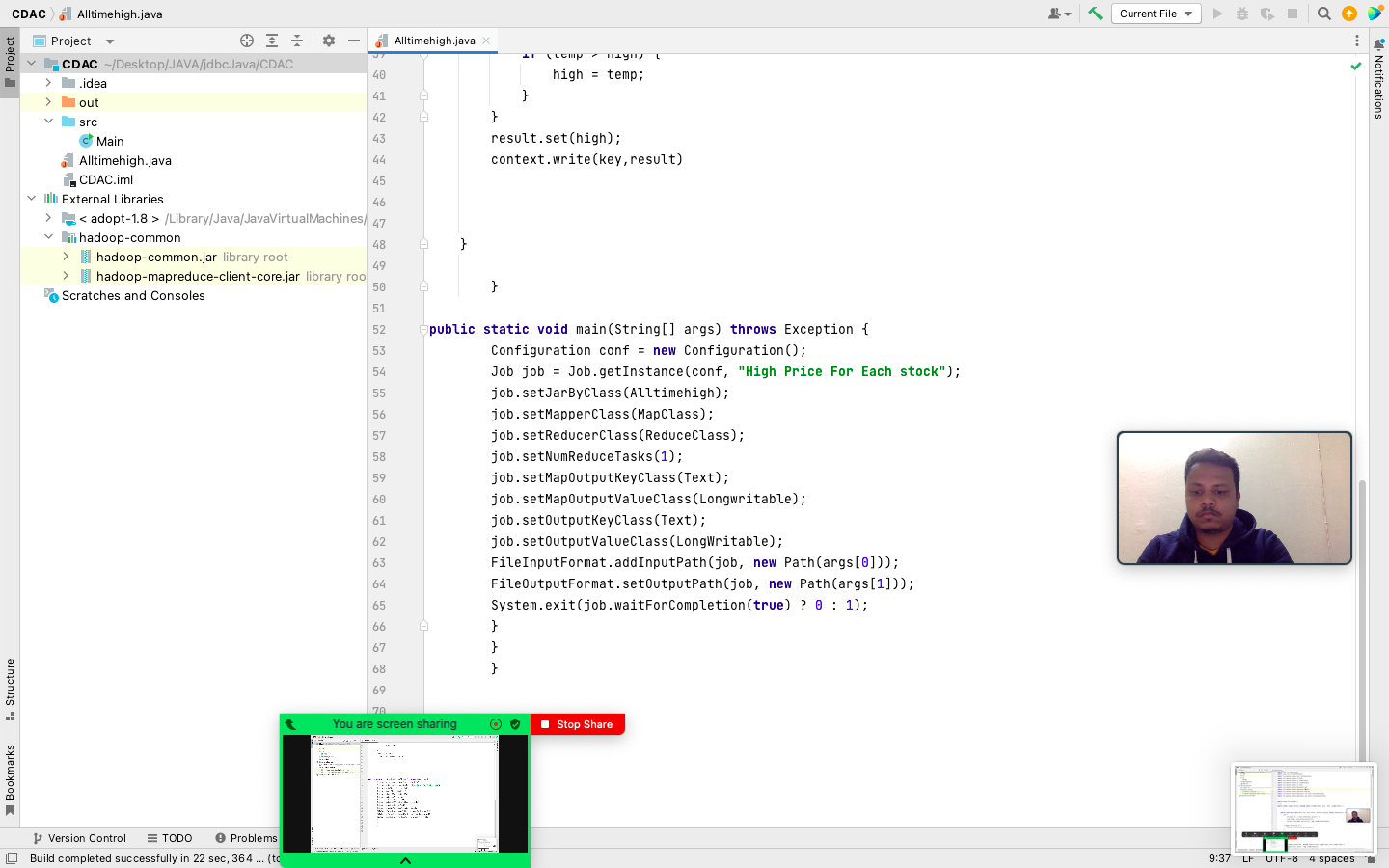
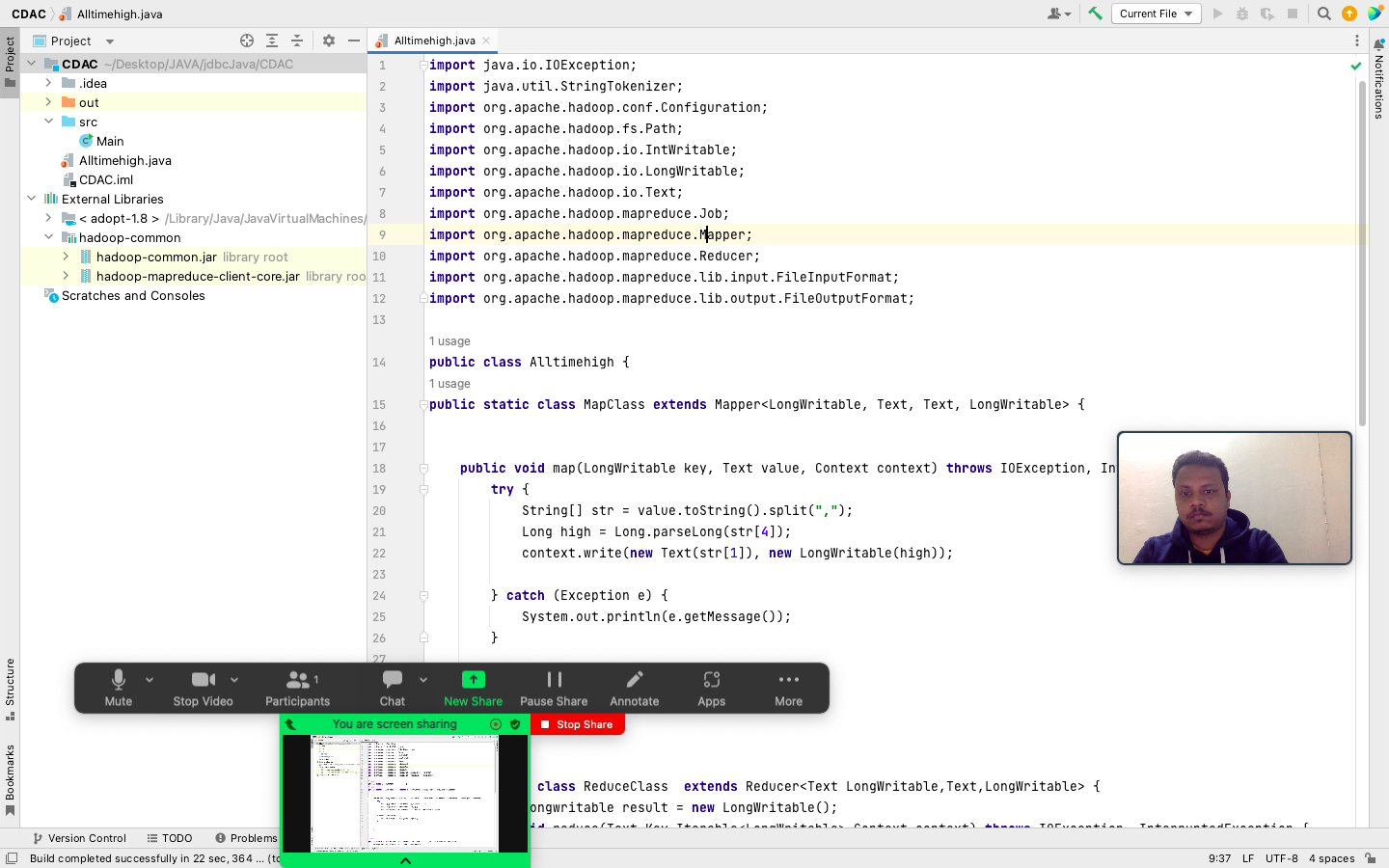
**}**

**}**

**hadoop fs -put NYSE.csv**

**Hadoop jar myjar Alltimehigh /user/bigcdac432531/NYSE.csv /user/bigcdac432531/out2**

**MY eclipse is not opening so i am not able to create jar file**

****

**2)**

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

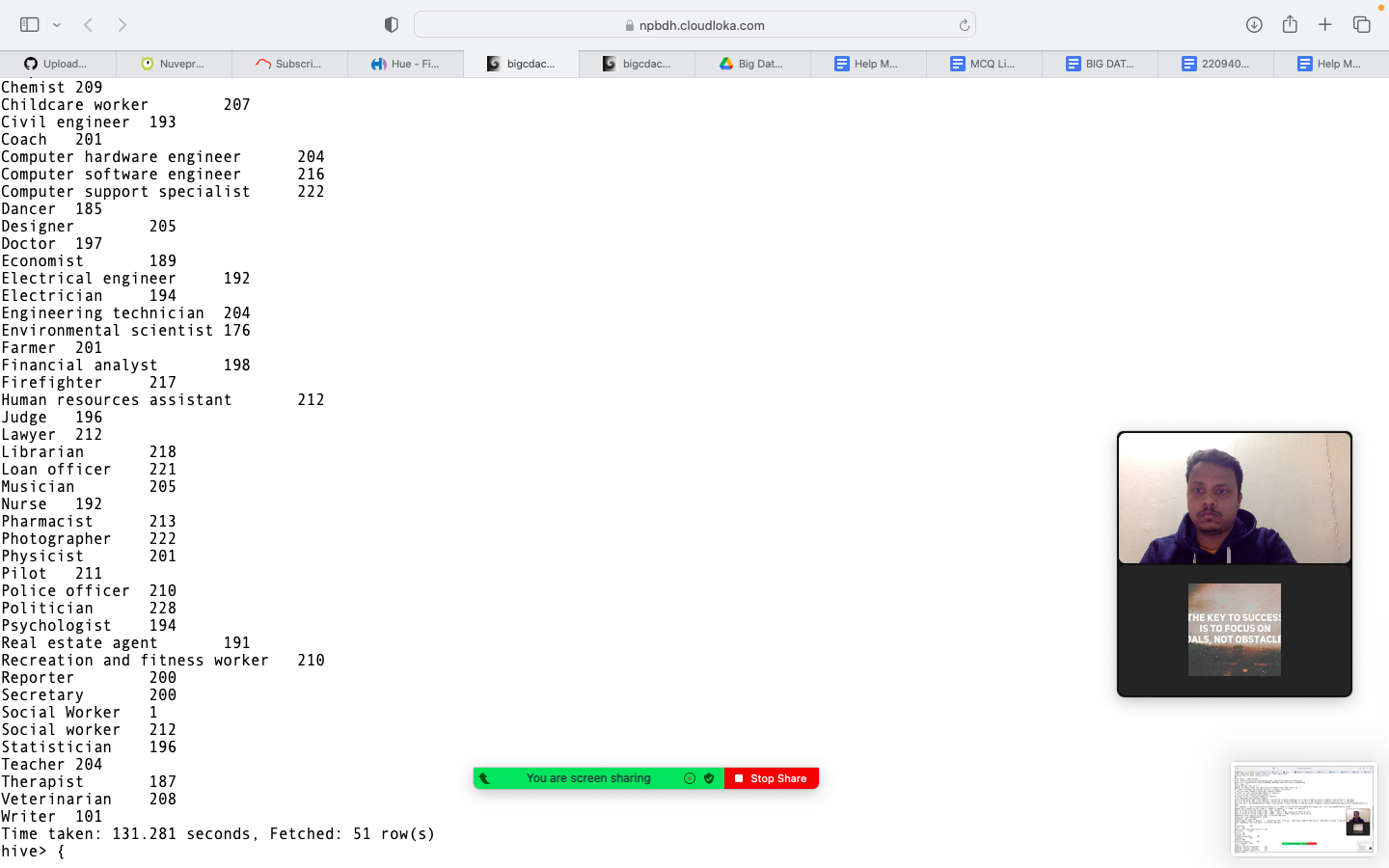
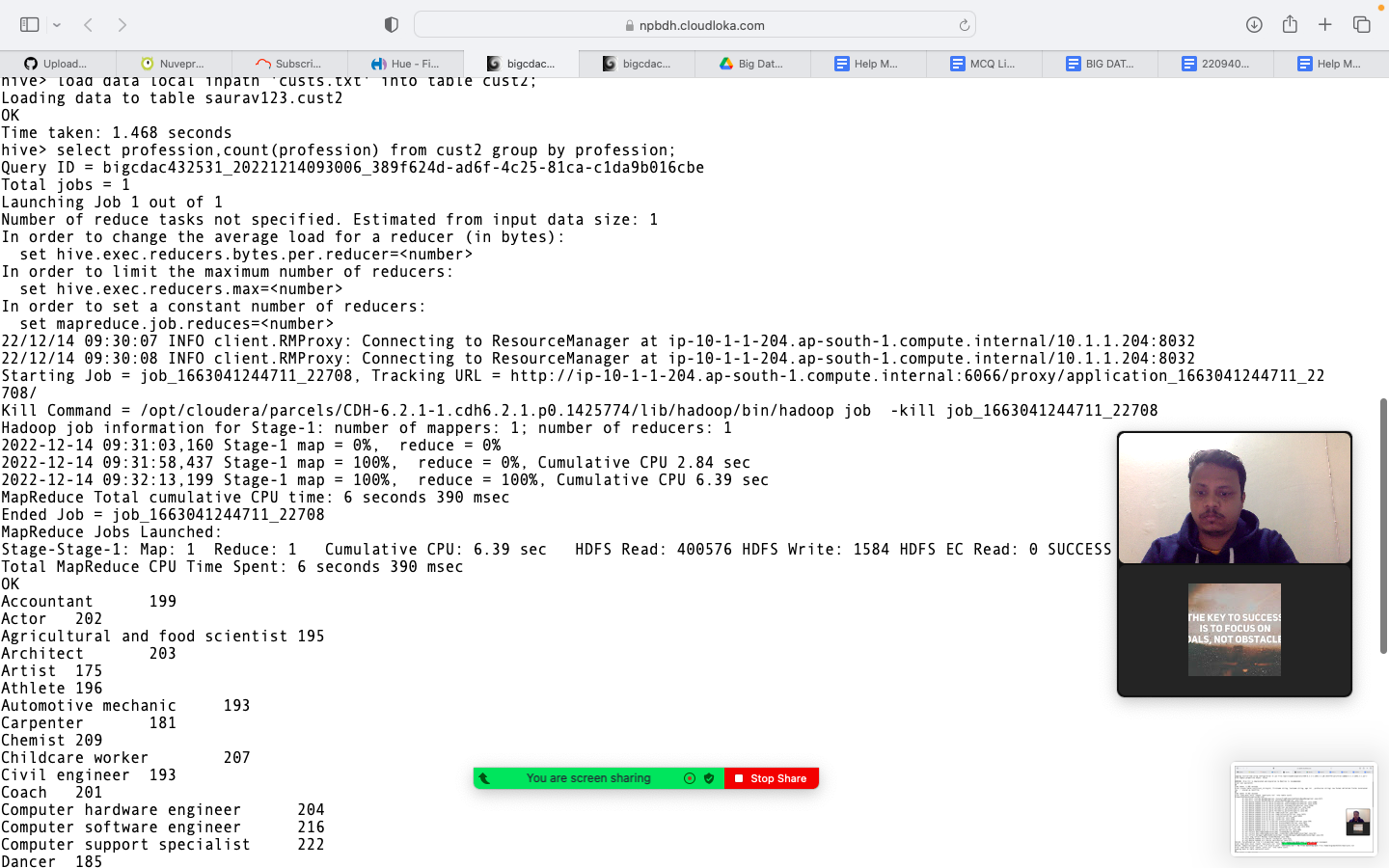
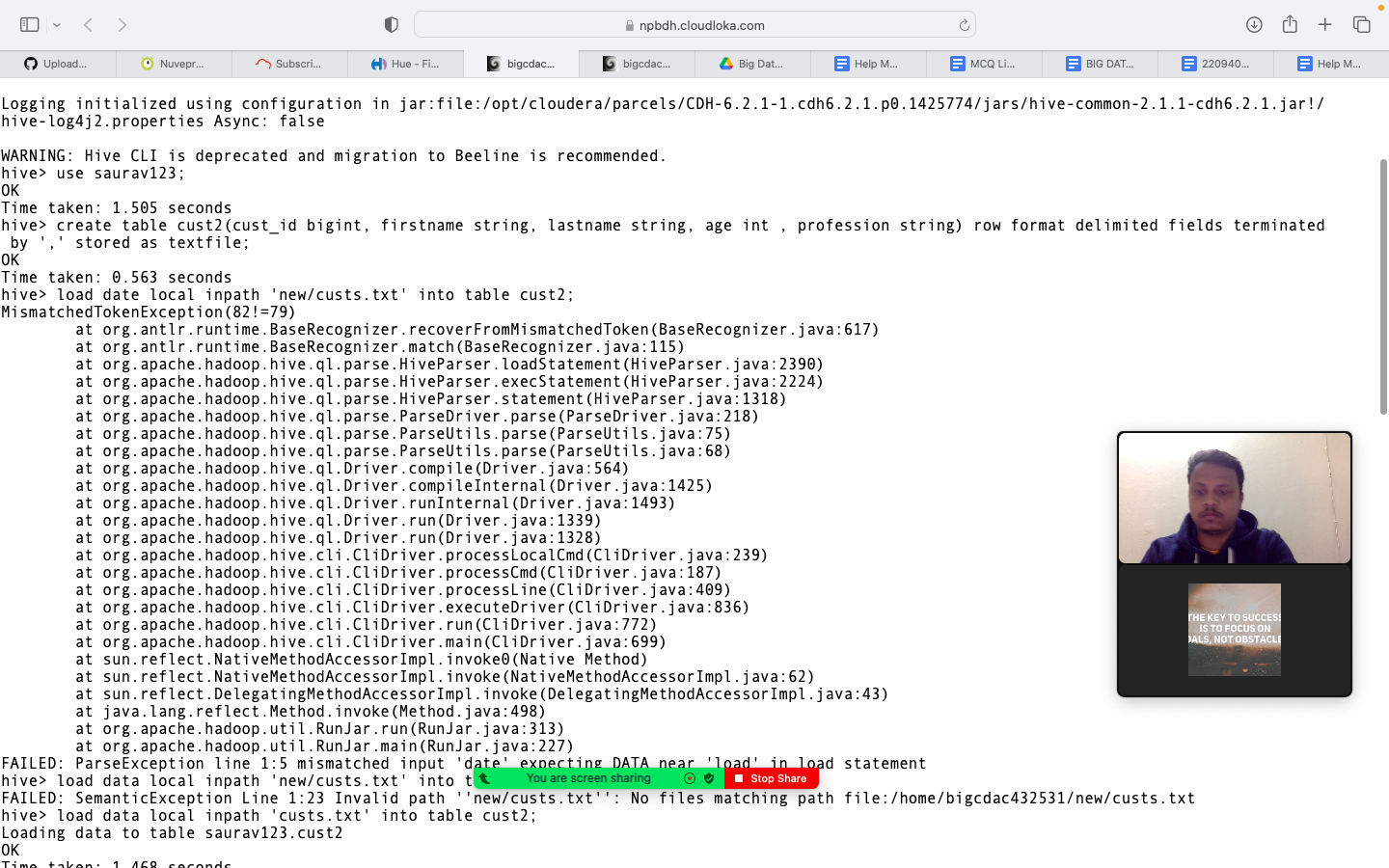
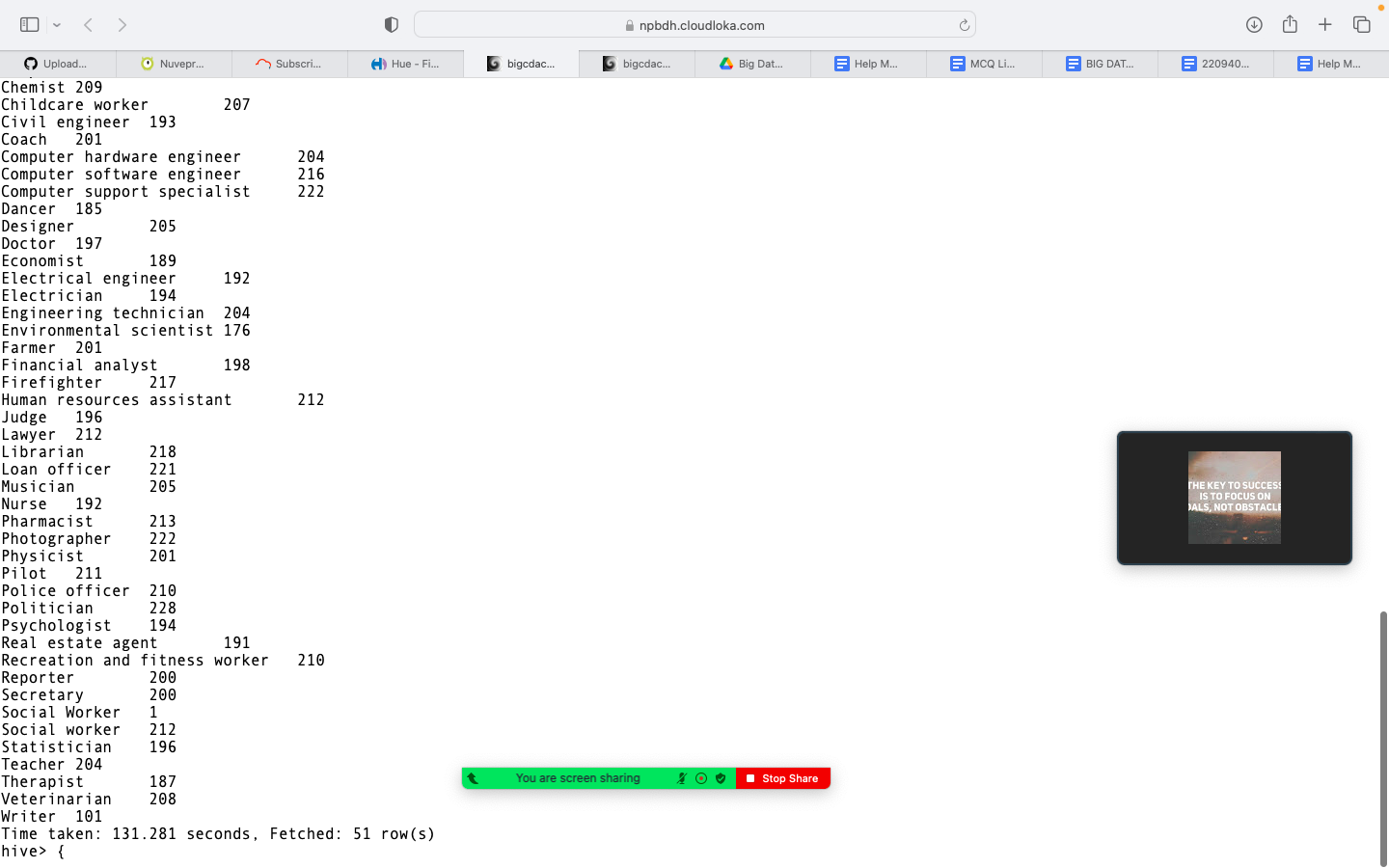
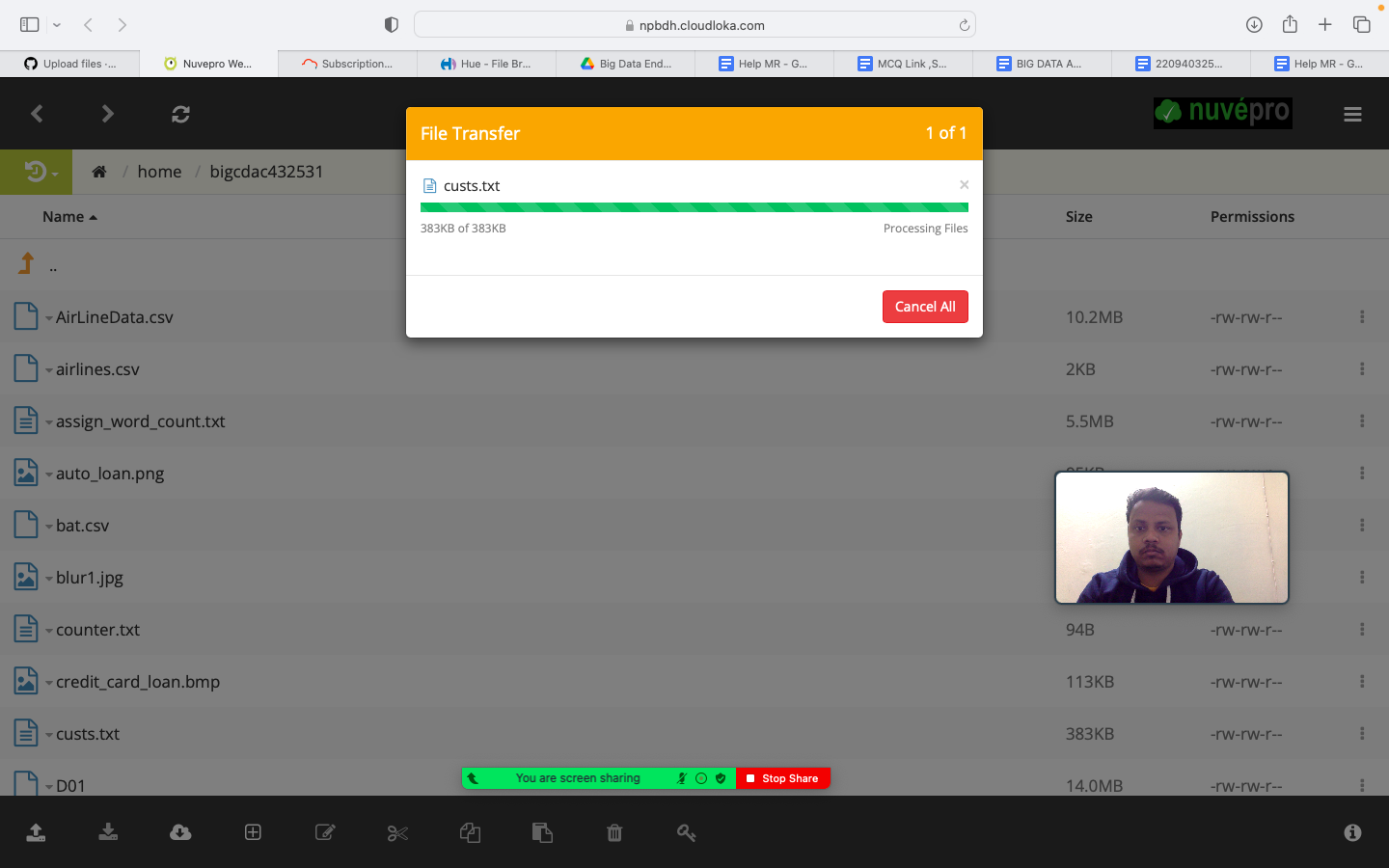
**use saurav123;**

**create table cust2(cust\_id bigint, firstname string, lastname string, age int , profession string) row format delimited fields terminated**

**by ',' stored as textfile;**

**load data local inpath 'custs.txt' into table cust2;**

**select profession,count(profession) from cust2 group by profession;**

****

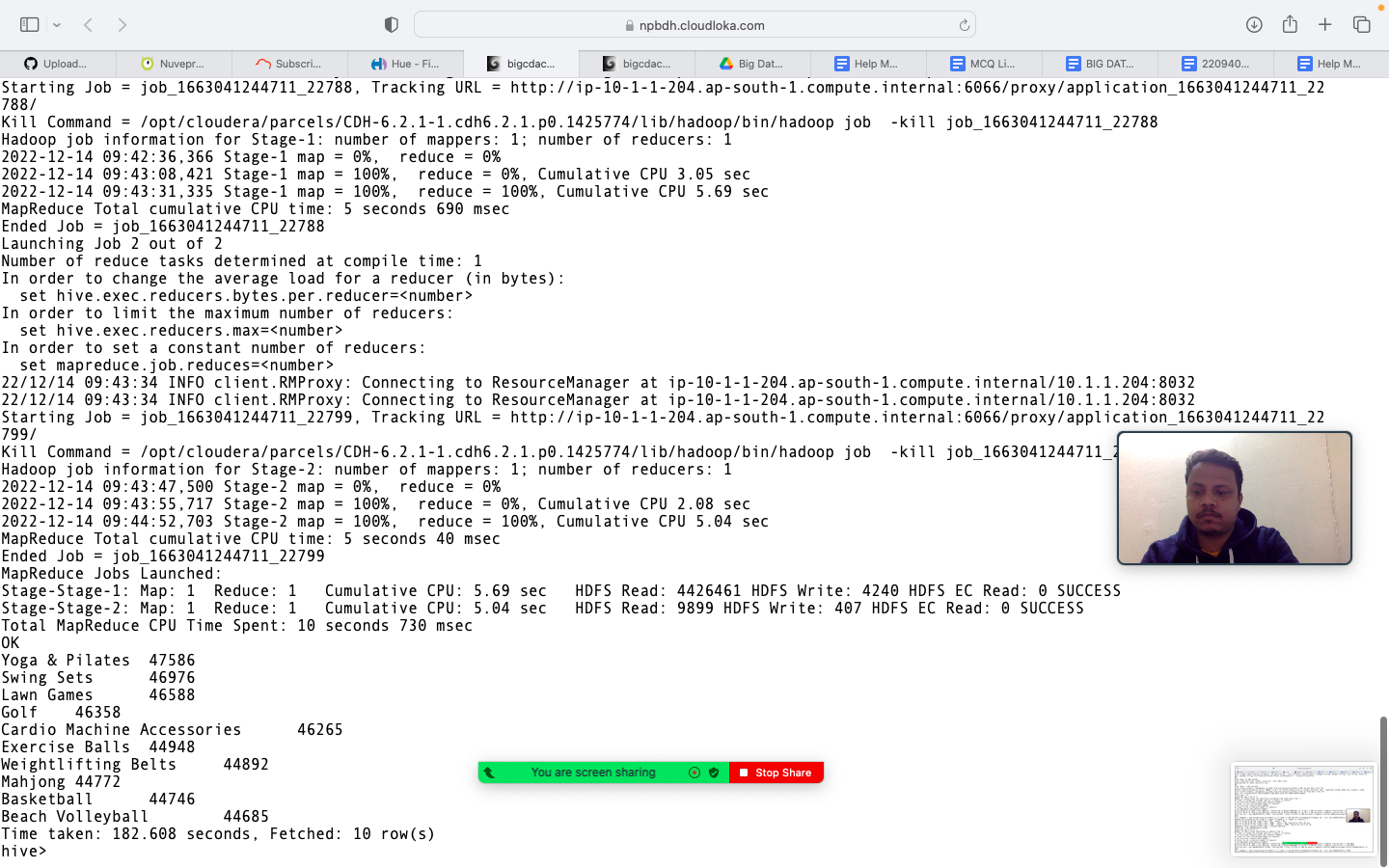
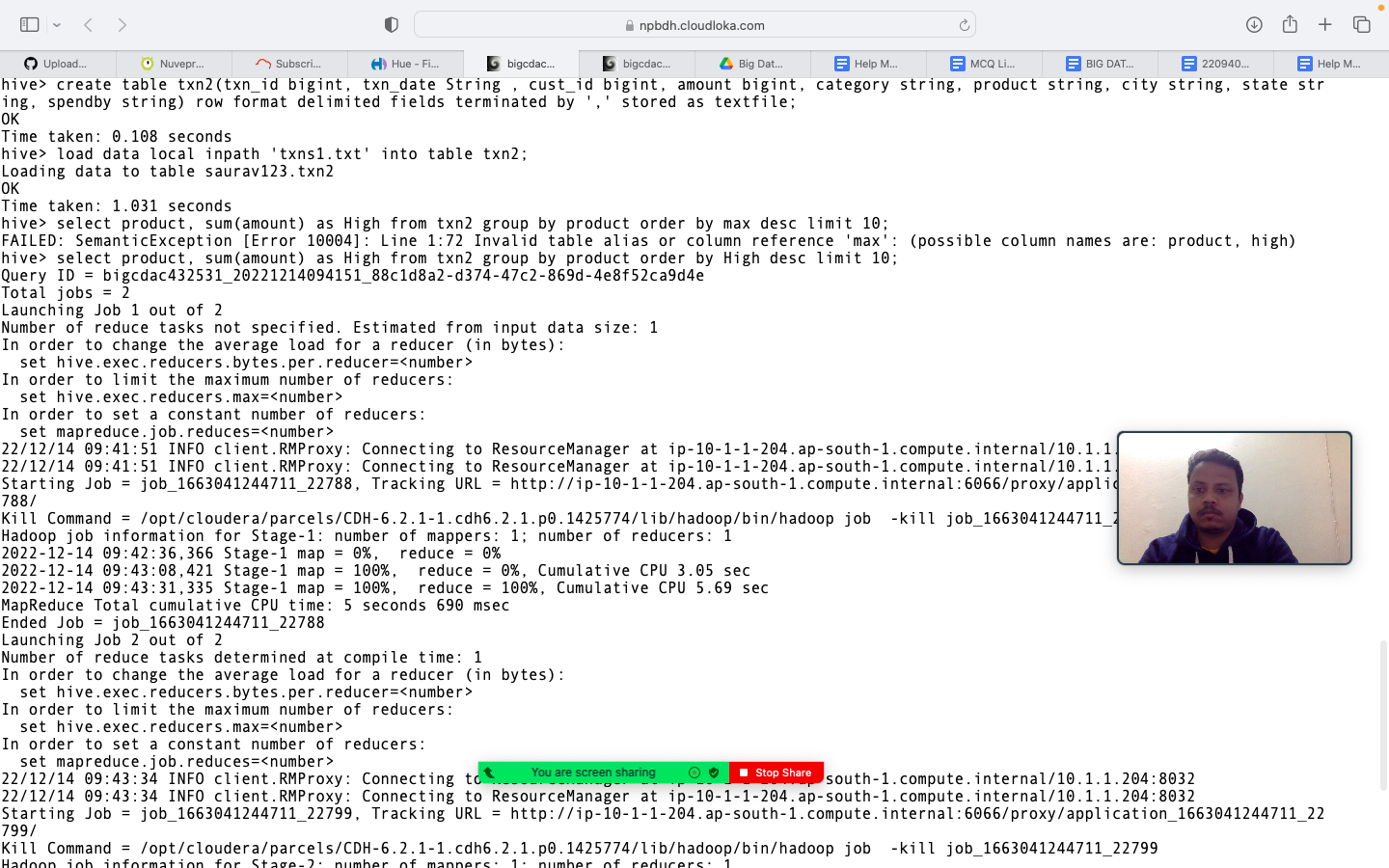
**2) Write a program to find the top 10 products sales wise**

**create table txn2(txn\_id bigint, txn\_date String , cust\_id bigint, amount bigint, category string, product string, city string, state str**

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

**load data local inpath 'txns1.txt' into table txn2;**

**select product, sum(amount) as High from txn2 group by product order by High desc limit 10;**

****

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

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

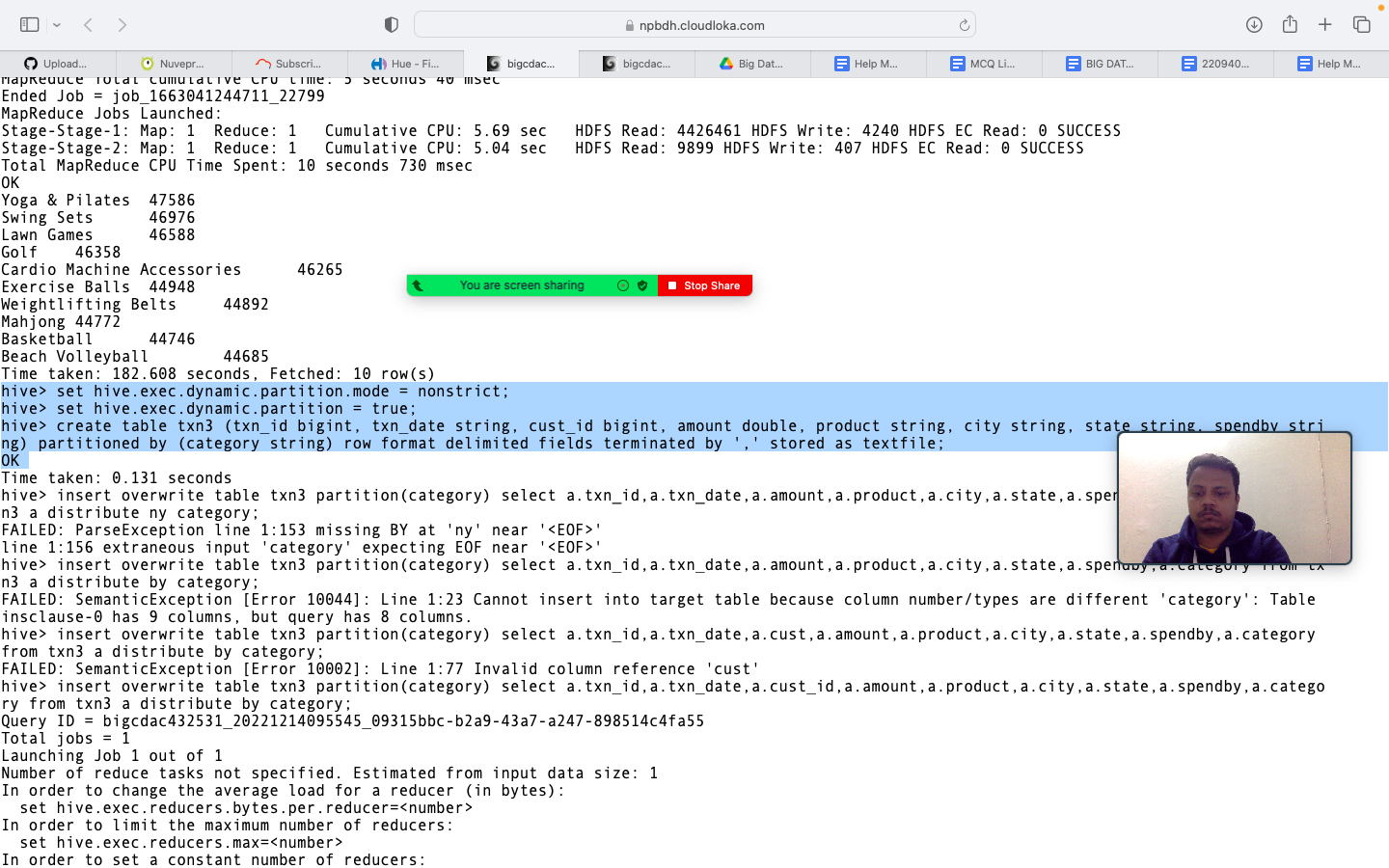
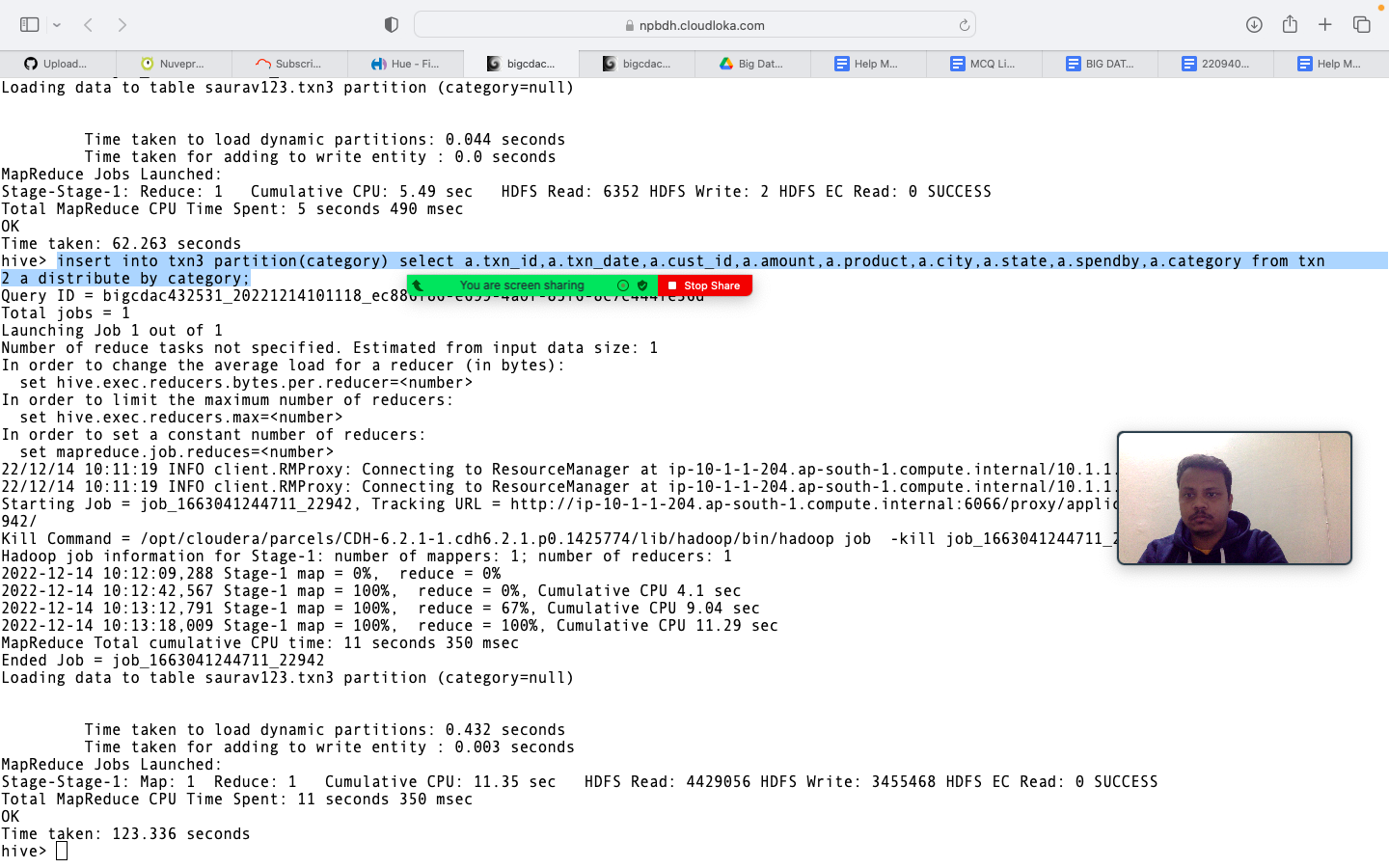
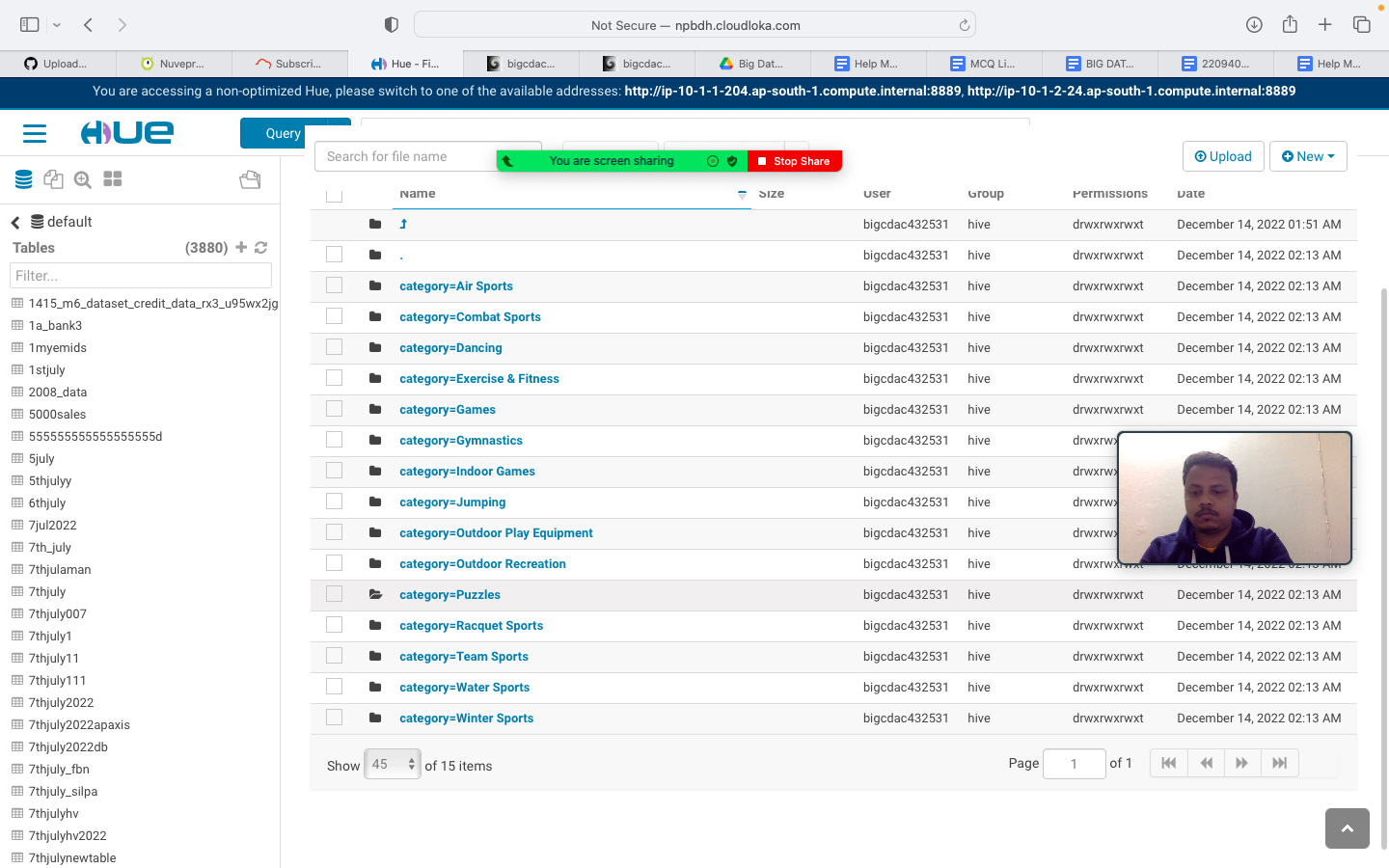
**hive> set hive.exec.dynamic.partition = true;**

**hive> create table txn3 (txn\_id bigint, txn\_date string, cust\_id bigint, amount double, product string, city string, state string, spendby stri**

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

**insert into txn3 partition(category) select a.txn\_id,a.txn\_date,a.cust\_id,a.amount,a.product,a.city,a.state,a.spendby,a.category from txn**

**2 a distribute by category;**

****

**QUESTION 3 [15 marks]**

**PySpark**

Please find the AIRLINES data set

Year

Quarter

Average revenue per seat

Total number of booked seats

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

**year?**

**airRDD= sc.textFile("hdfs://nameservice1/user/bigcdac432531/airlines.csv")**

**>>> airRDD3 = airRDD.map(lambda a: a.split(","))**

**>>> for i in airRDD3.atke(5):**

**... print(i)**

**...**

**Traceback (most recent call last):**

**File "<stdin>", line 1, in <module>**

**AttributeError: 'PipelinedRDD' object has no attribute 'atke'**

**>>> for i in airRDD3.take(5):**

**... print(i)**

**...**

**['Year', 'Quarter', 'Average revenue per seat', 'total no. of booked seats']**

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

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

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

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

**>>> header = airRDD1.first()**

**>>> header = airRDD3.first()**

**>>> airRDD2=airRDD3.filter(lambda a : a!=header)**

**>>> key = airRDD2.map(lambda a : (a[0],int(a[3])))**

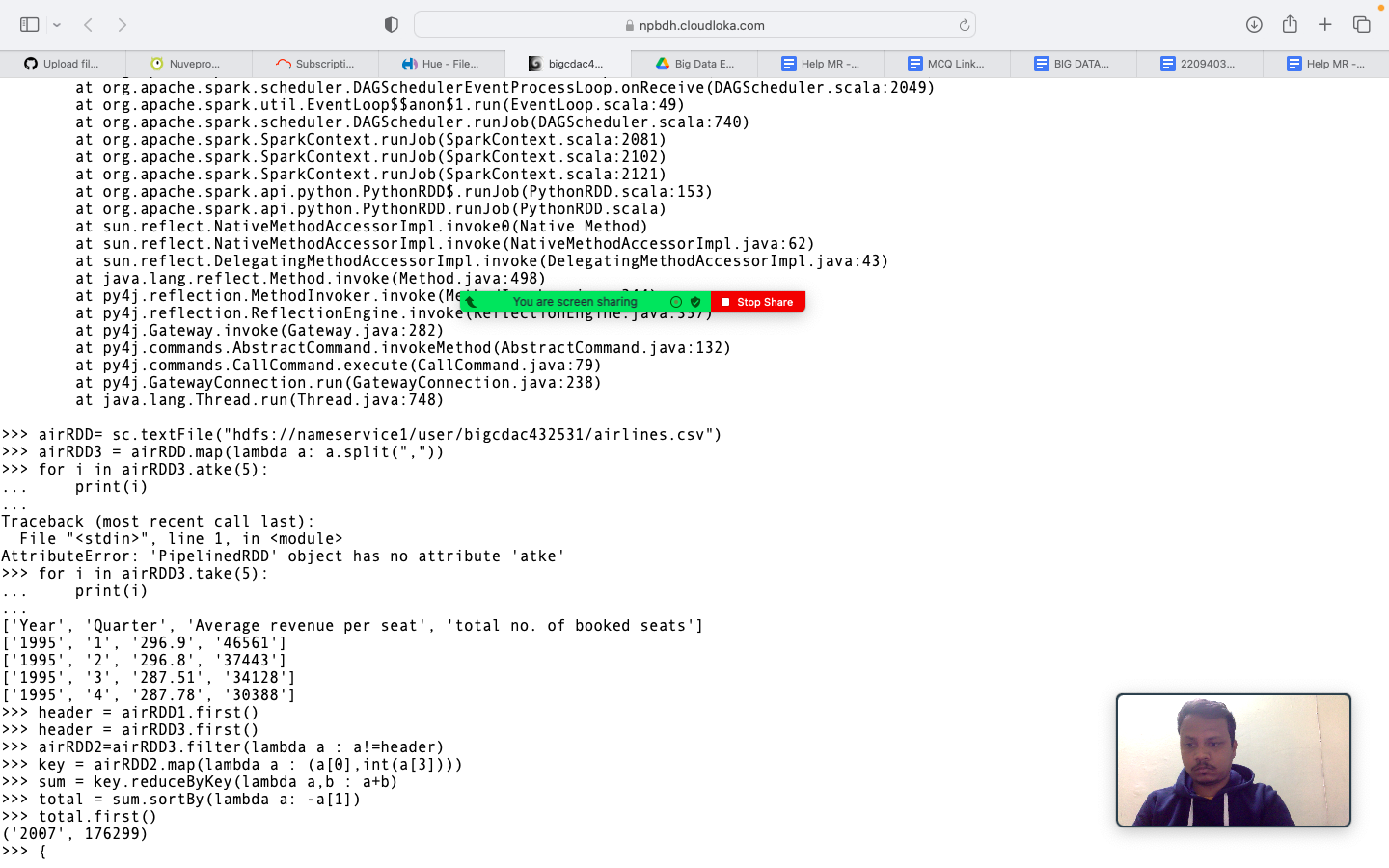
**>>> sum = key.reduceByKey(lambda a,b : a+b)**

**>>> total = sum.sortBy(lambda a: -a[1])**

**>>> total.first()**

**('2007', 176299)**

**>>> {**

****

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

**keyvalue = airRDD2.map(lambda a : (a[0], float(a[2]\*int(a[3])))**

**...**

**...**

**Traceback (most recent call last):**

**File "/opt/cloudera/parcels/CDH-6.2.1-1.cdh6.2.1.p0.1425774/lib/spark/python/pyspark/context.py", line 257, in signal\_handler**

**raise KeyboardInterrupt()**

**KeyboardInterrupt**

**>>> keyvalue = airRDD2.map(lambda a : (a[0], float(a[2])\*int(a[3])))**

**>>> addtotal = keyvalue.reduceByKey(lambda a : -a[1])**

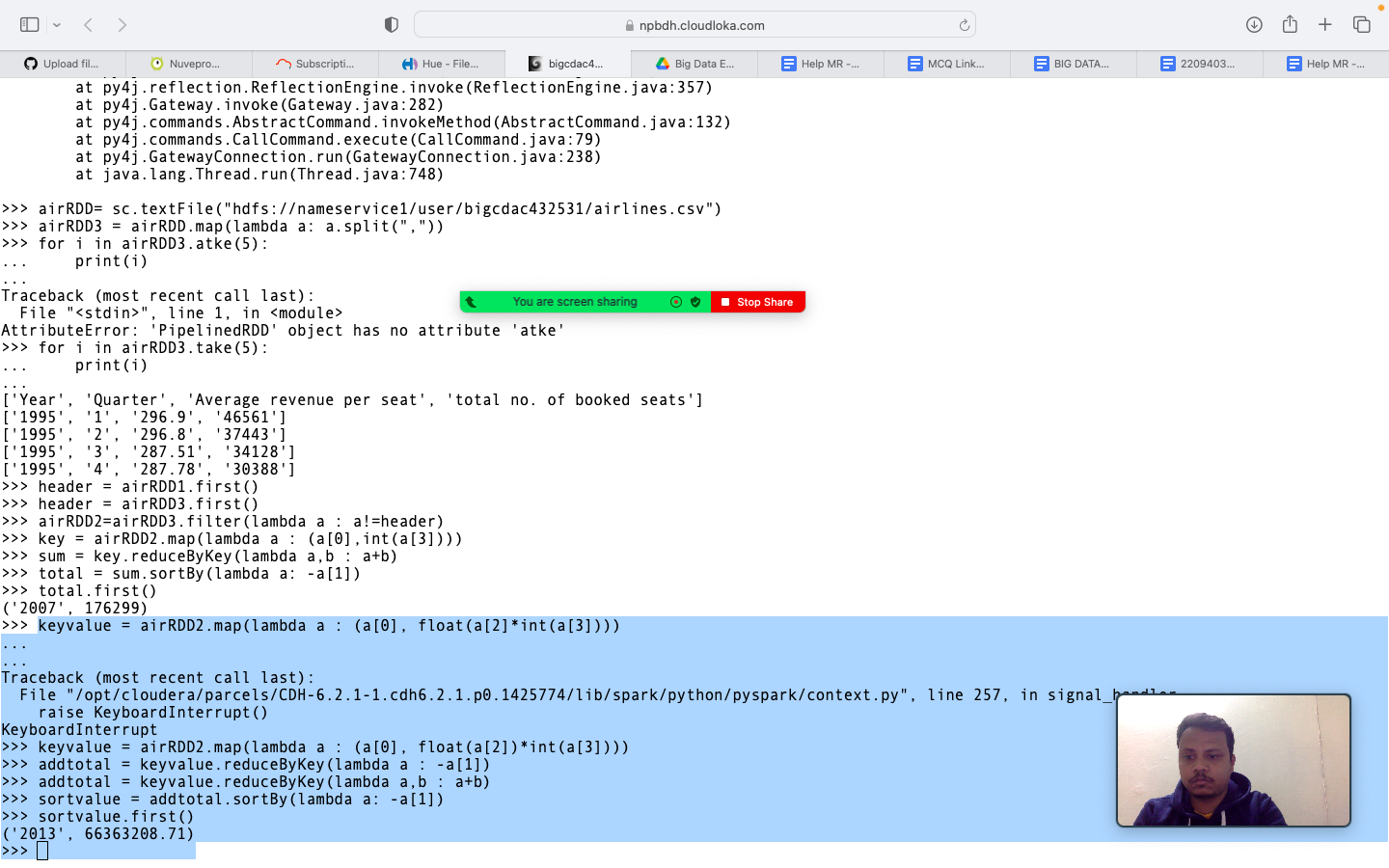
**>>> addtotal = keyvalue.reduceByKey(lambda a,b : a+b)**

**>>> sortvalue = addtotal.sortBy(lambda a: -a[1])**

**>>> sortvalue.first()**

**('2013', 66363208.71)**

**>>>**

****

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

**quarter (Common group)**

**keyvalue = airRDD2.map(lambda a : (a[0], float(a[2])\*int(a[3])))**

**>>> addtotal = keyvalue.reduceByKey(lambda a : -a[1])**

**>>> addtotal = keyvalue.reduceByKey(lambda a,b : a+b)**

**>>> sortvalue = addtotal.sortBy(lambda a: -a[1])**

**>>> sortvalue.first()**

**('2013', 66363208.71)**

**>>> key = airRD2.map(lambda a : (a[0]+" "+a[1],float(a[2])\*int(a[3])))**

**Traceback (most recent call last):**

**File "<stdin>", line 1, in <module>**

**NameError: name 'airRD2' is not defined**

**>>> key = airRDD2.map(lambda a : (a[0]+" "+a[1],float(a[2])\*int(a[3])))**

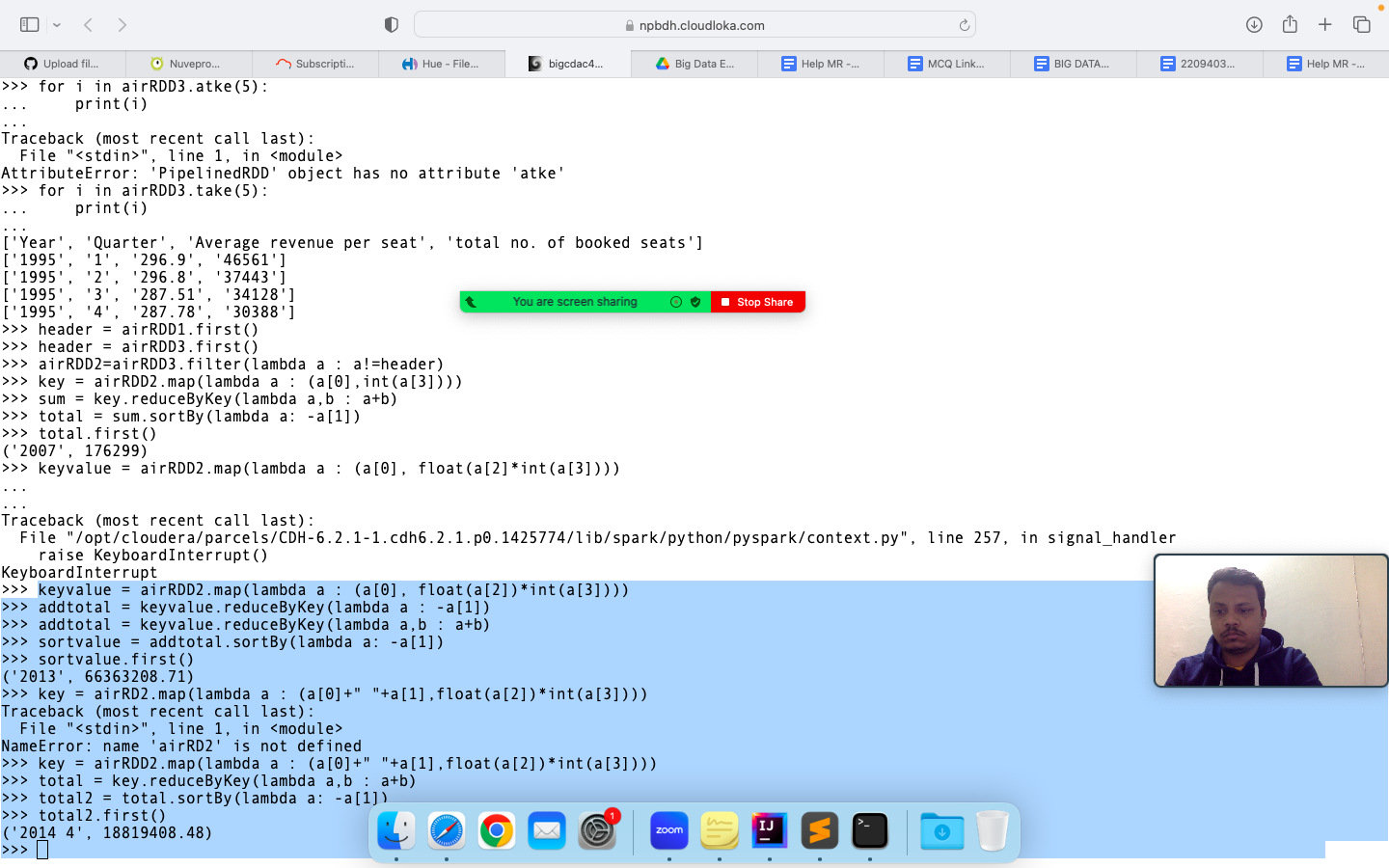
**>>> total = key.reduceByKey(lambda a,b : a+b)**

**>>> total2 = total.sortBy(lambda a: -a[1])**

**>>> total2.first()**

**('2014 4', 18819408.48)**

**>>>**

****