HIVE

1.

hive (lab\_exam)> create table customer(

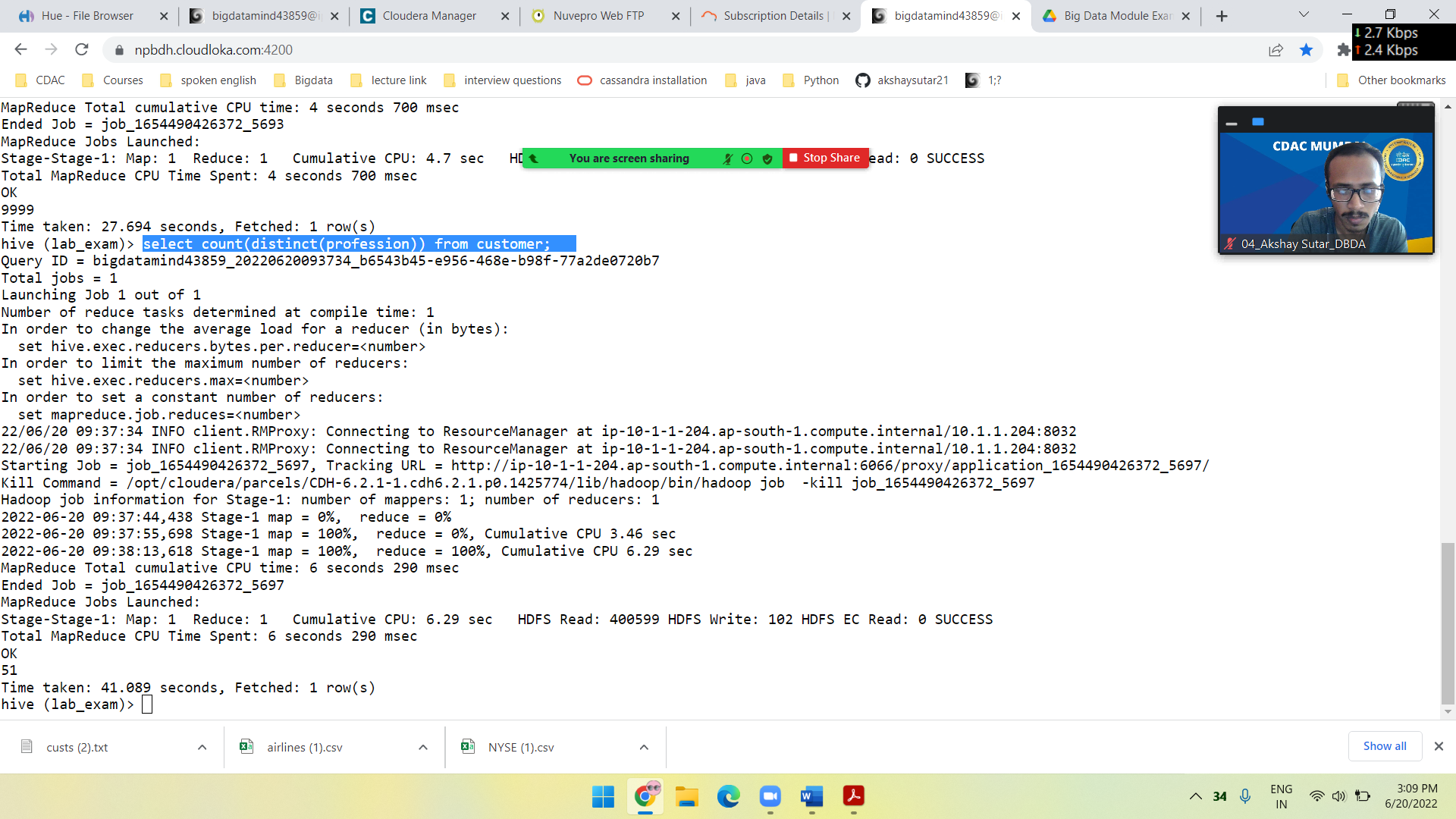
> custid int,firstname string ,lastname string,age int,profession string)

> row format delimited

> fields terminated by ','

> stored as textfile;

load data local inpath ‘custs.txt’ overwrite into table customer;



2.

hive (lab\_exam)> create table sales2(txnid int,txndate string,custid int,amount double,category string,product string,city string,state string,spendby string)

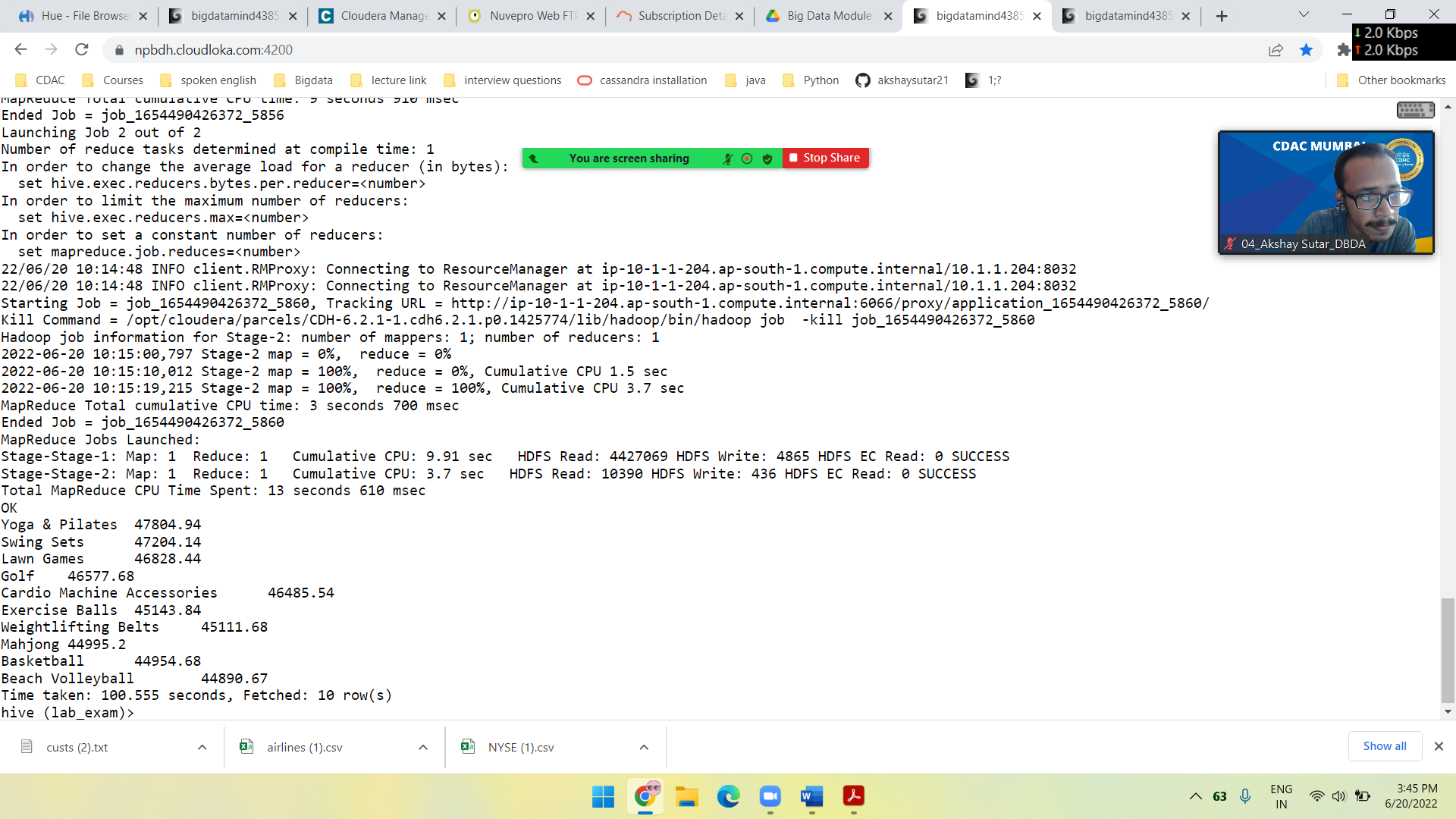
> row format delimited

> fields terminated by ','

> stored as textfile;

load data local inpath ‘txns1.txt’ into table sales2;

select product,round(sum(amount),2) as total from sales2 group by product order by total desc limit 10;



3.

hive (lab\_exam)> create table cat\_partition(txnid int,txndate string,custid int,amount double,product string,city string,state string,spendby string)

> partitioned by (category string)

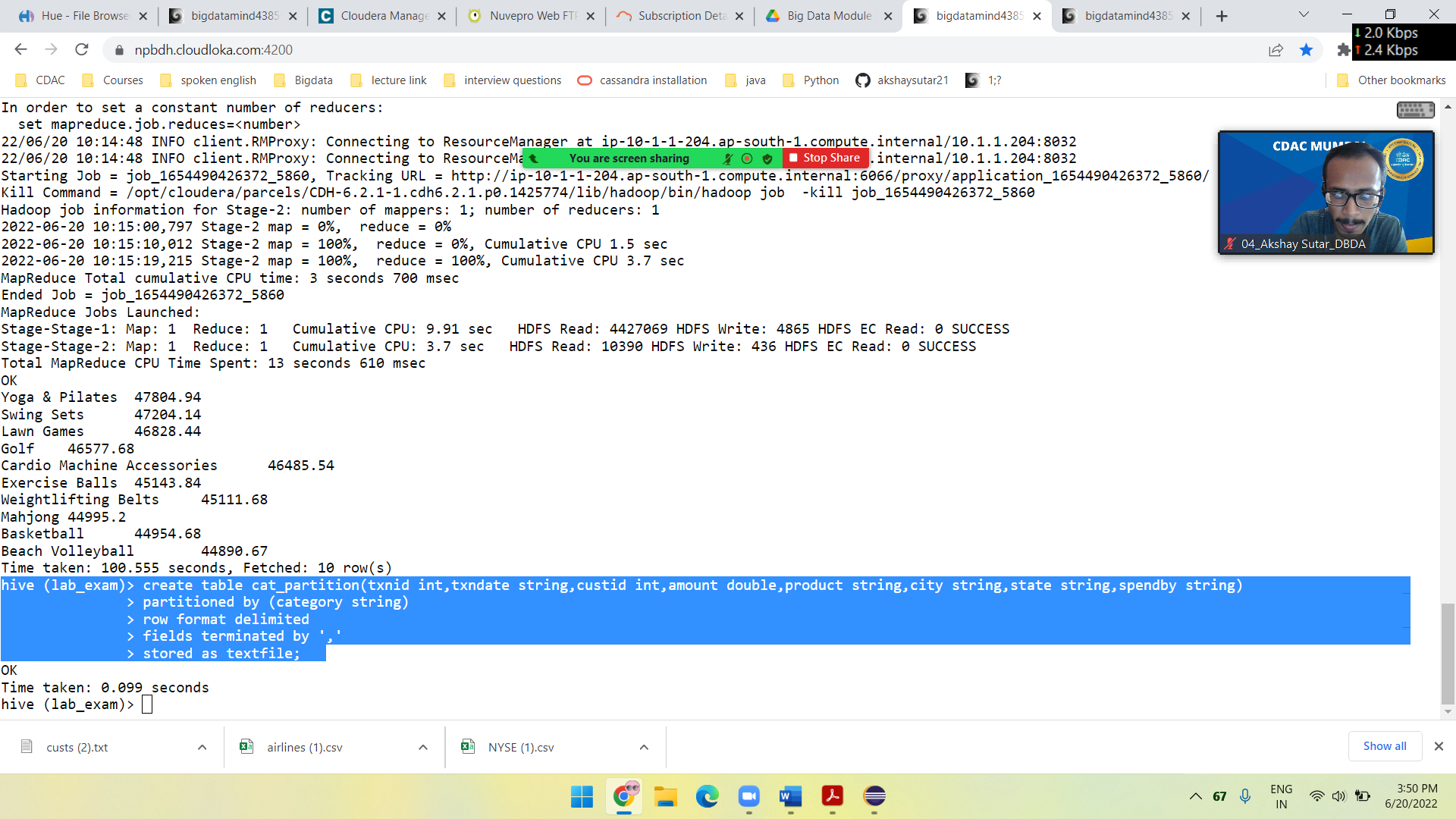
> row format delimited

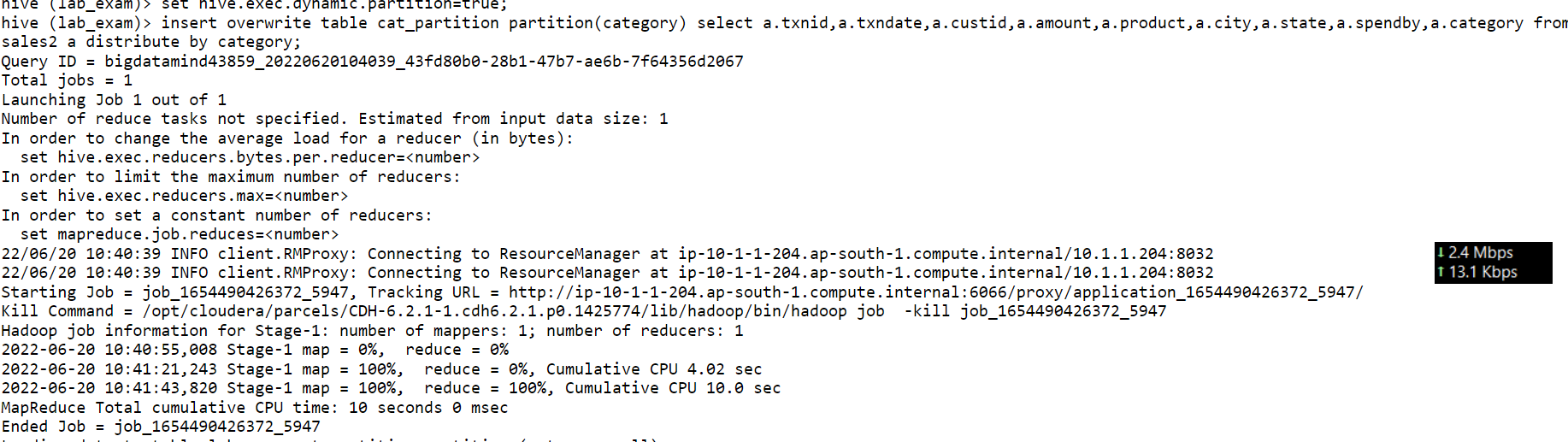
> fields terminated by ','

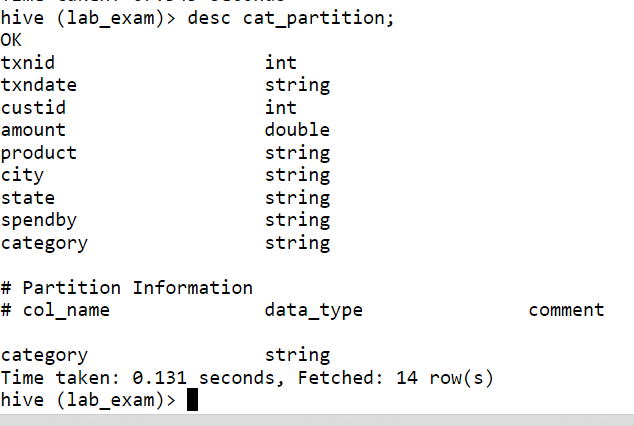
> stored as textfile;

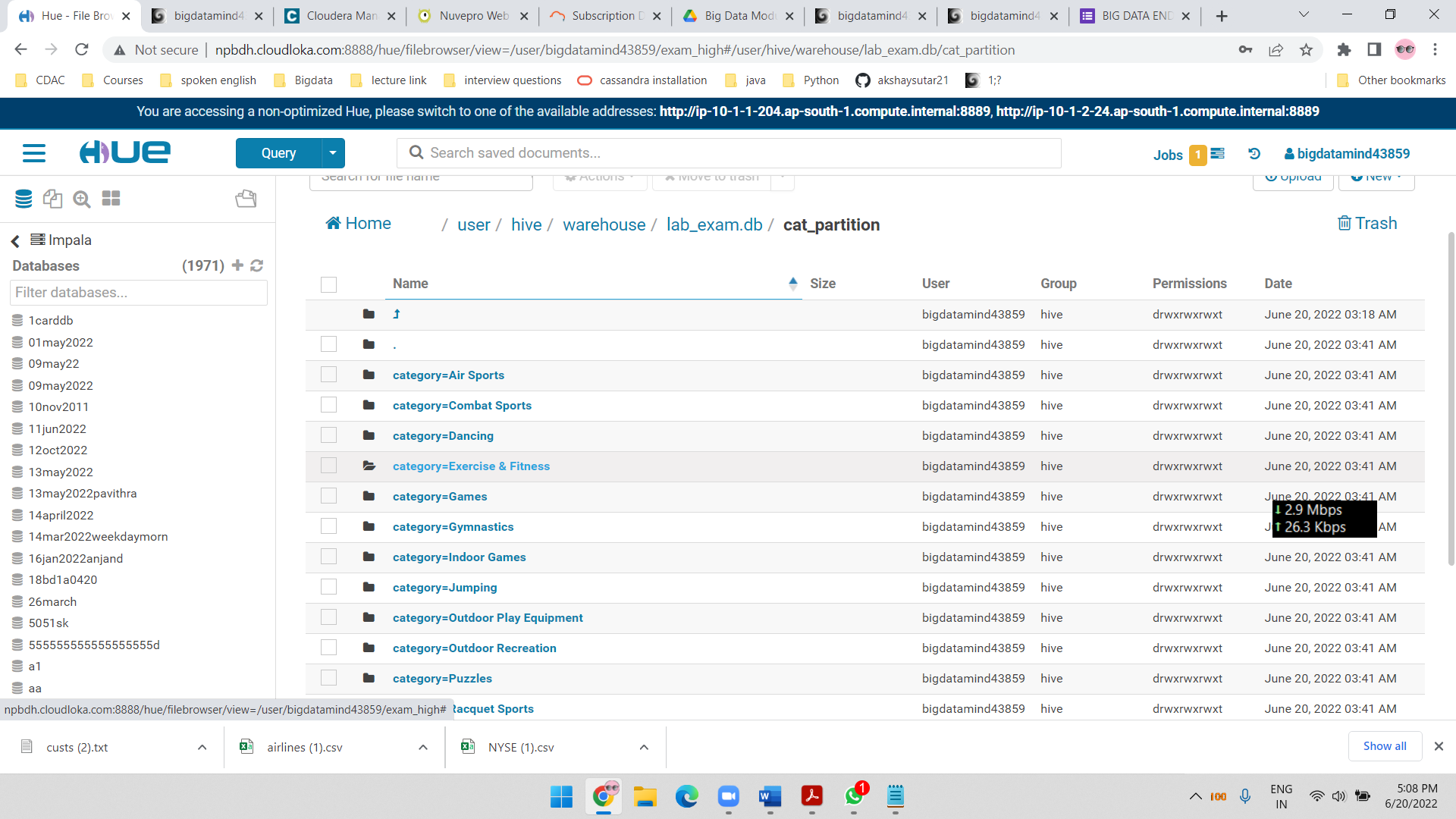
insert overwrite table cat\_partition partition(category) select a.txnid,a.txndate,a.custid,a.amount,a.product,a.city,a.state,a.spendby,a.category from

sales2 a distribute by category;









Pyspark

1. What was the highest number of people travelled in which year?

>>> airlineRDD=sc.textFile("/user/bigdatamind43859/airlines.csv")

>>> ascii\_ignore=airlineRDD.map(lambda a:a.encode("ascii","ignore"))

>>> airlineRDD.count()

85

>>> header=ascii\_ignore.first()

>>> alRDD=ascii\_ignore.filter(lambda a:a!=header)

>>> alRDD.count()

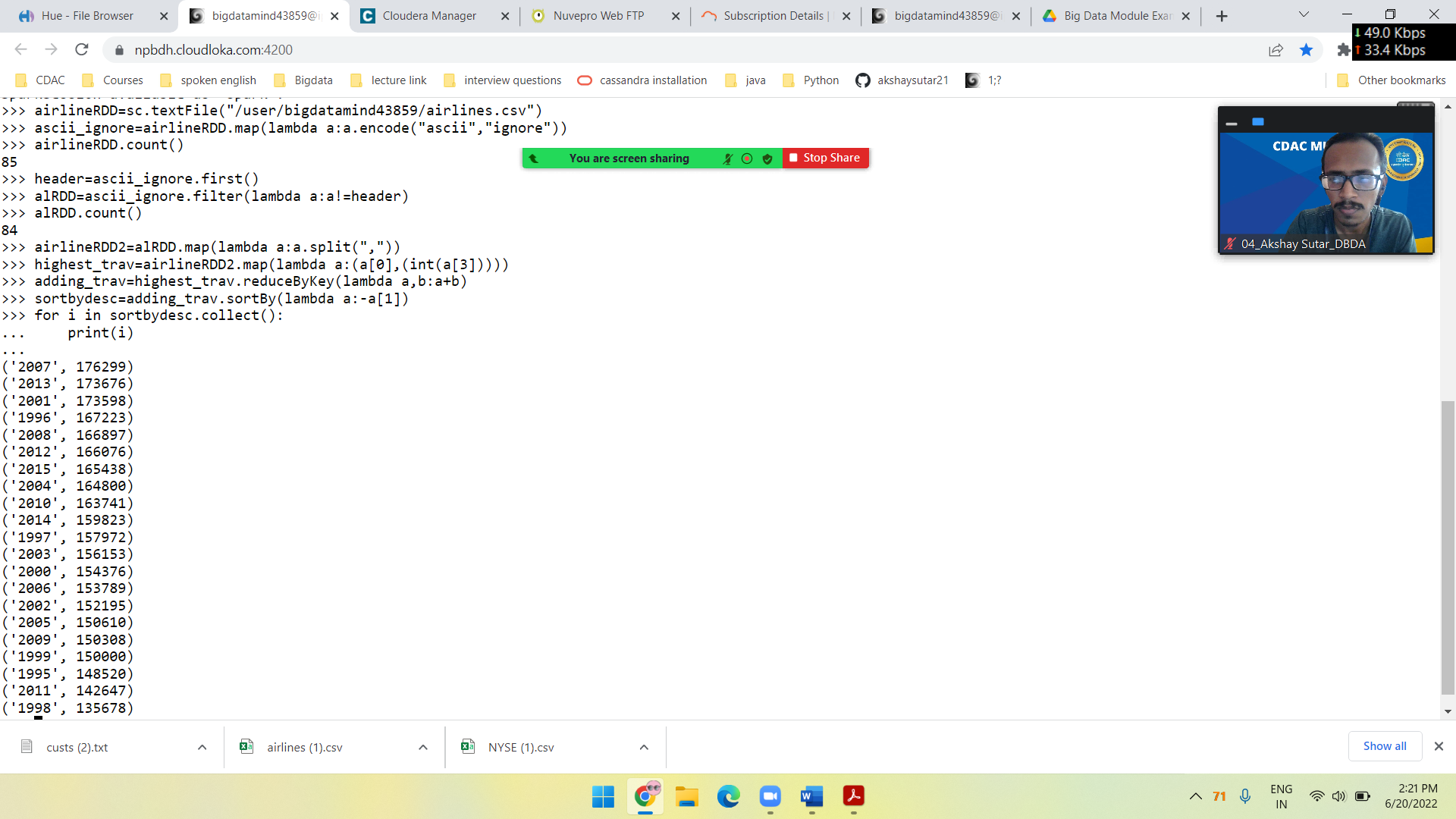
84

>>> airlineRDD2=alRDD.map(lambda a:a.split(","))

>>> highest\_trav=airlineRDD2.map(lambda a:(a[0],(int(a[3]))))

>>> adding\_trav=highest\_trav.reduceByKey(lambda a,b:a+b)

>>> sortbydesc=adding\_trav.sortBy(lambda a:-a[1])

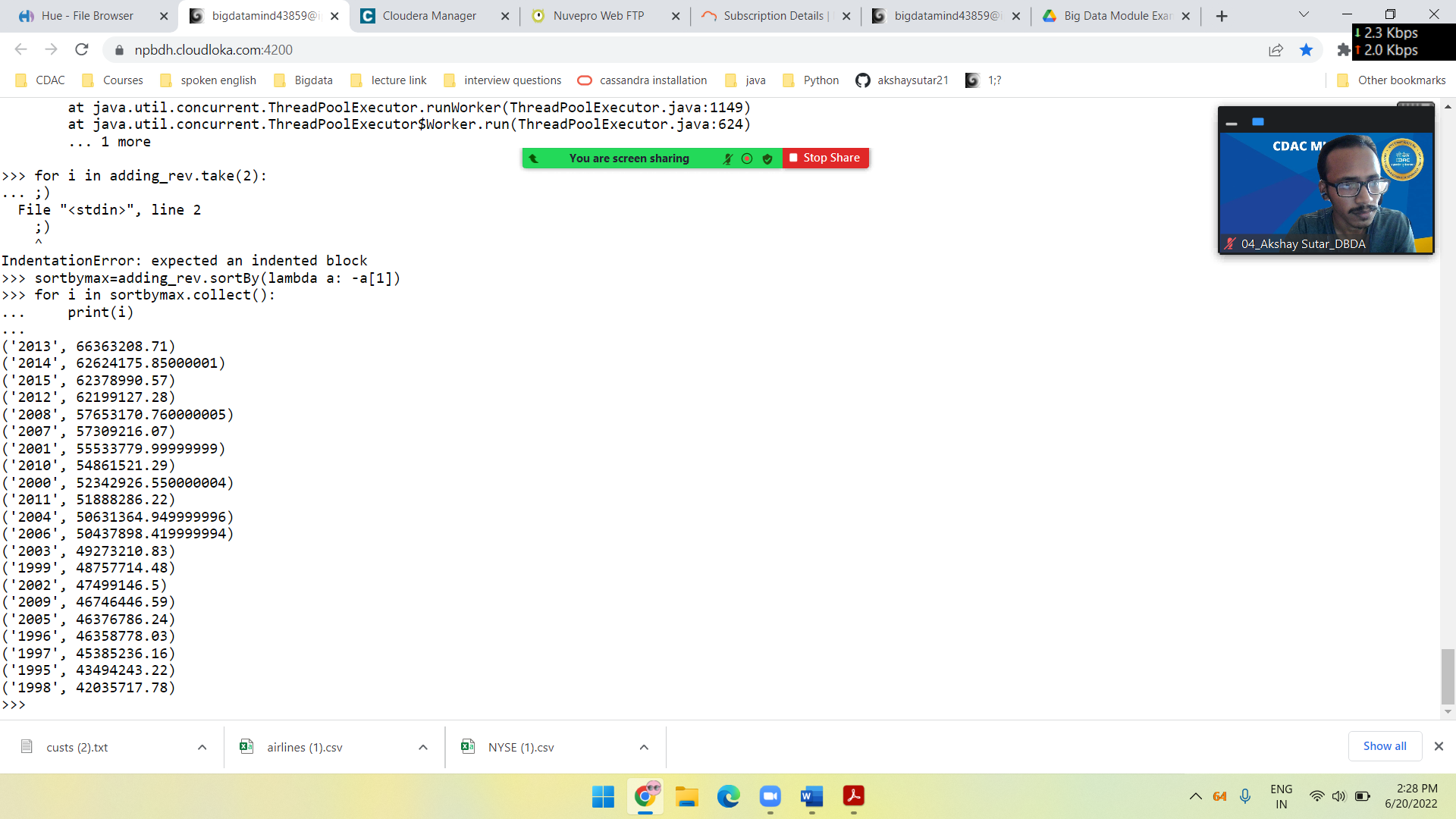


2.identifying the highest revenue generation for which year

>>> rev\_kv=airlineRDD2.map(lambda a: (a[0],float(a[2])\*int(a[3])))

>>> adding\_rev=rev\_kv.reduceByKey(lambda a,b:a+b)

>>> sortbymax=adding\_rev.map(lambda a:-a[1])

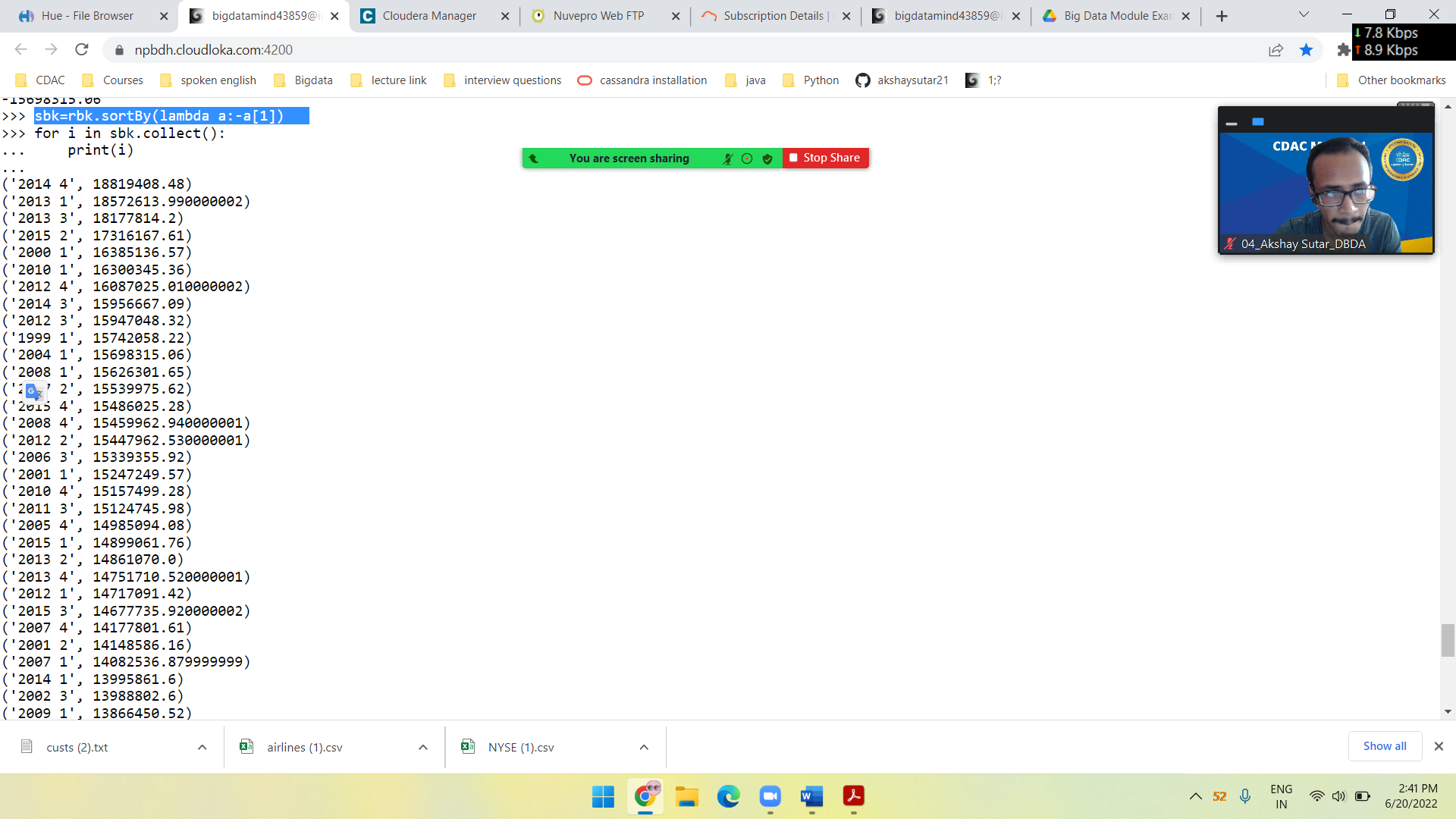


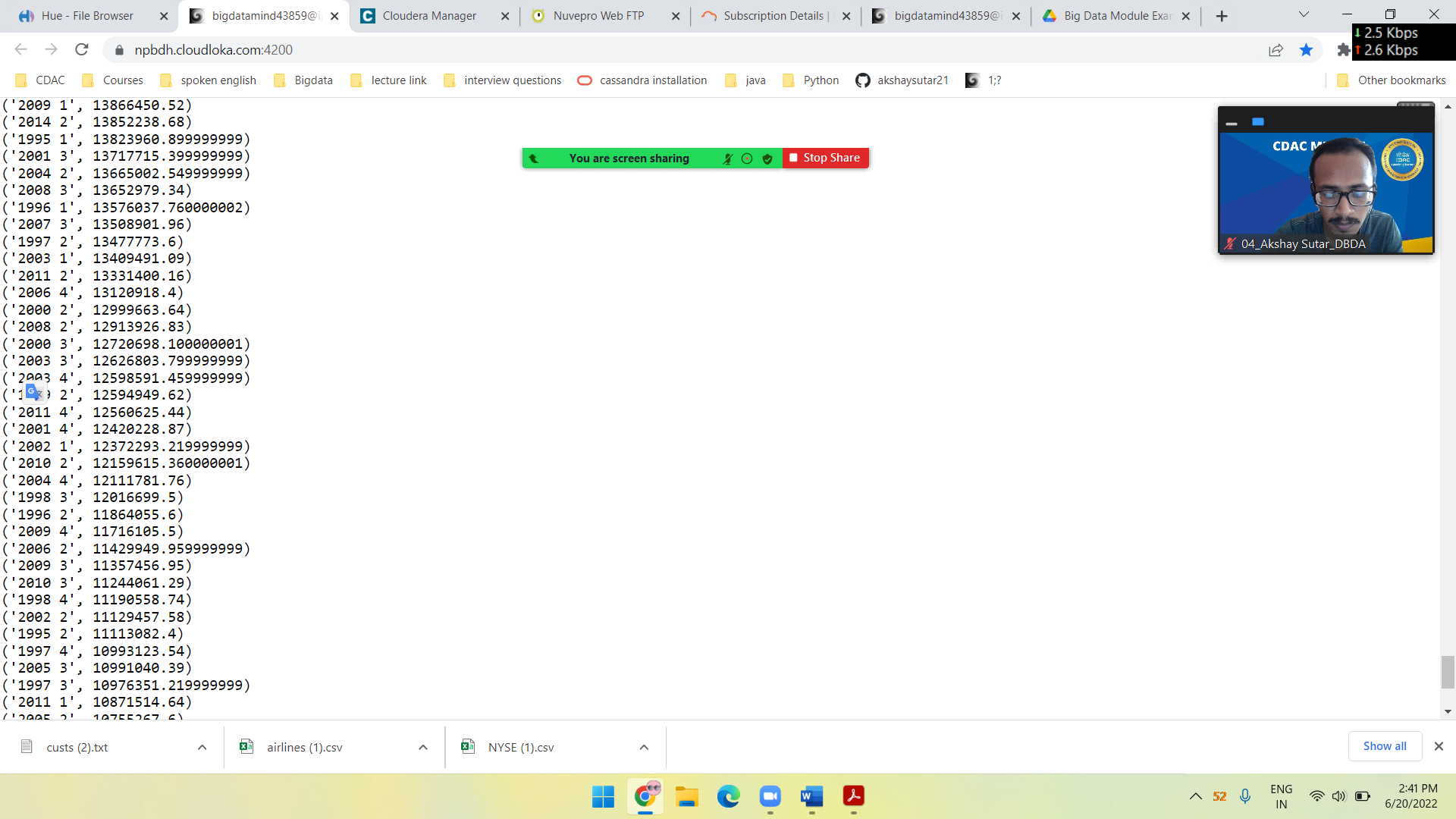
3.

rev\_kv=airlineRDD2.map(lambda a: (a[0]+" "+a[1],float(a[2])\*int(a[3])))

rbk=rev\_kv.reduceByKey(lambda a,b:a+b)

sbk=rbk.sortBy(lambda a:-a[1])





('2014 4', 18819408.48)

('2013 1', 18572613.990000002)

('2013 3', 18177814.2)

('2015 2', 17316167.61)

('2000 1', 16385136.57)

('2010 1', 16300345.36)

('2012 4', 16087025.010000002)

('2014 3', 15956667.09)

('2012 3', 15947048.32)

('1999 1', 15742058.22)

('2004 1', 15698315.06)

('2008 1', 15626301.65)

('2007 2', 15539975.62)

('2015 4', 15486025.28)

('2008 4', 15459962.940000001)

('2012 2', 15447962.530000001)

('2006 3', 15339355.92)

('2001 1', 15247249.57)

('2010 4', 15157499.28)

('2011 3', 15124745.98)

('2005 4', 14985094.08)

('2015 1', 14899061.76)

('2013 2', 14861070.0)

('2013 4', 14751710.520000001)

('2012 1', 14717091.42)

('2015 3', 14677735.920000002)

('2007 4', 14177801.61)

('2001 2', 14148586.16)

('2007 1', 14082536.879999999)

('2014 1', 13995861.6)

('2002 3', 13988802.6)

('2009 1', 13866450.52)

('2014 2', 13852238.68)

('1995 1', 13823960.899999999)

('2001 3', 13717715.399999999)

('2004 2', 13665002.549999999)

('2008 3', 13652979.34)

('1996 1', 13576037.760000002)

('2007 3', 13508901.96)

('1997 2', 13477773.6)

('2003 1', 13409491.09)

('2011 2', 13331400.16)

('2006 4', 13120918.4)

('2000 2', 12999663.64)

('2008 2', 12913926.83)

('2000 3', 12720698.100000001)

('2003 3', 12626803.799999999)

('2003 4', 12598591.459999999)

('1999 2', 12594949.62)

('2011 4', 12560625.44)

('2001 4', 12420228.87)

('2002 1', 12372293.219999999)

('2010 2', 12159615.360000001)

('2004 4', 12111781.76)

('1998 3', 12016699.5)

('1996 2', 11864055.6)

('2009 4', 11716105.5)

('2006 2', 11429949.959999999)

('2009 3', 11357456.95)

('2010 3', 11244061.29)

('1998 4', 11190558.74)

('2002 2', 11129457.58)

('1995 2', 11113082.4)

('1997 4', 10993123.54)

('2005 3', 10991040.39)

('1997 3', 10976351.219999999)

('2011 1', 10871514.64)

('2005 2', 10755267.6)

('2003 2', 10638324.479999999)

('2006 1', 10547674.139999999)

('1996 3', 10497174.48)

('1999 3', 10483486.56)

('1996 4', 10421510.19)

('2000 4', 10237428.24)

('2002 4', 10008593.100000001)

('1997 1', 9937987.799999999)

('1999 4', 9937220.08)

('1995 3', 9812141.28)

('2009 2', 9806433.62)

('2005 1', 9645384.17)

('1998 1', 9542933.1)

('1998 2', 9285526.440000001)

('2004 3', 9156265.58)

('1995 4', 8745058.639999999)

Map reduce

Package cdac

import java.io.\*;

import org.apache.hadoop.io.Text;

import org.apache.hadoop.io.DoubleWritable;

import org.apache.hadoop.io.LongWritable;

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 AllTimeH {

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 max=0;

double temp=0;

for (DoubleWritable value:values) {

temp=value.get();

if (temp>max) {

max=temp;

}

}

result.set(max);

context.write(key, result);

}

}

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

Configuration conf=new Configuration();

Job job=Job.getInstance(conf,"Highest Price for each stock");

job.setJarByClass(AllTimeH.class);

job.setMapperClass(MapClass.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);

}

}

hadoop jar myjar.jarAllTimeHigh'/user/bigdatamind43859/NYSE.csv' '/user/bigdatamind43859/exam\_high'

