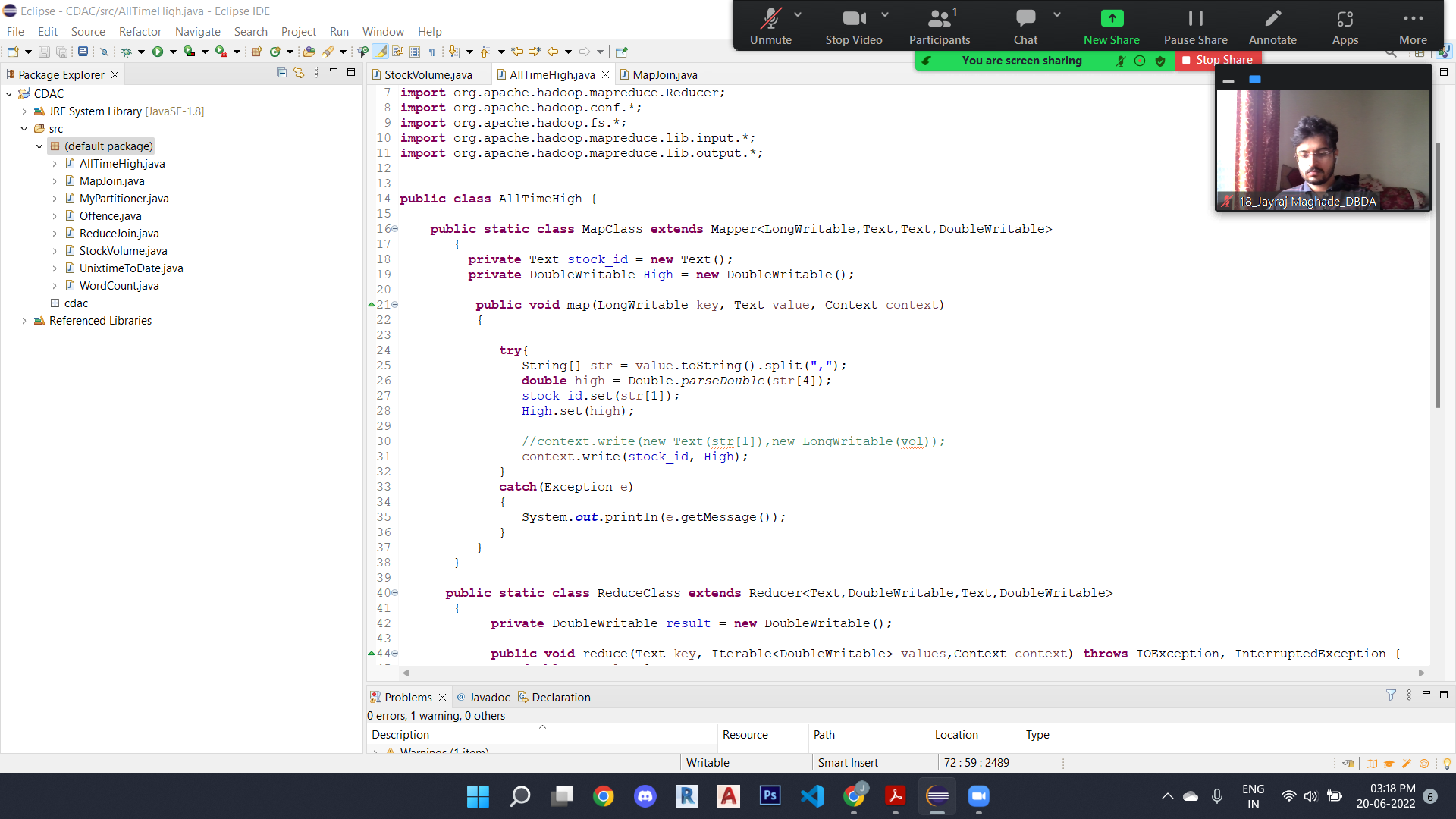
**Q.1 Map Reduce -**

All time high for all stocks -

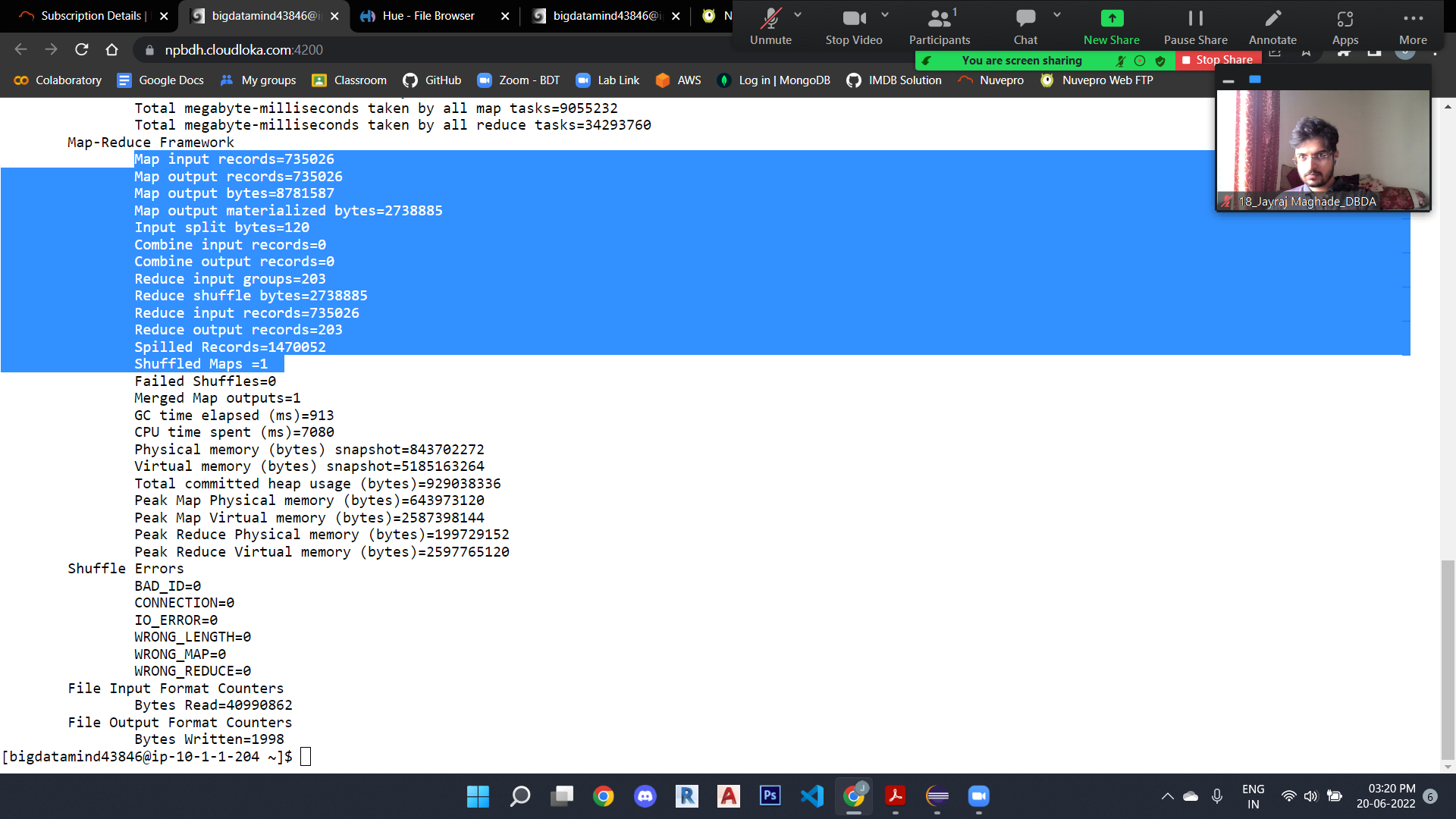
haddop fs -put NYSE.csv cdac

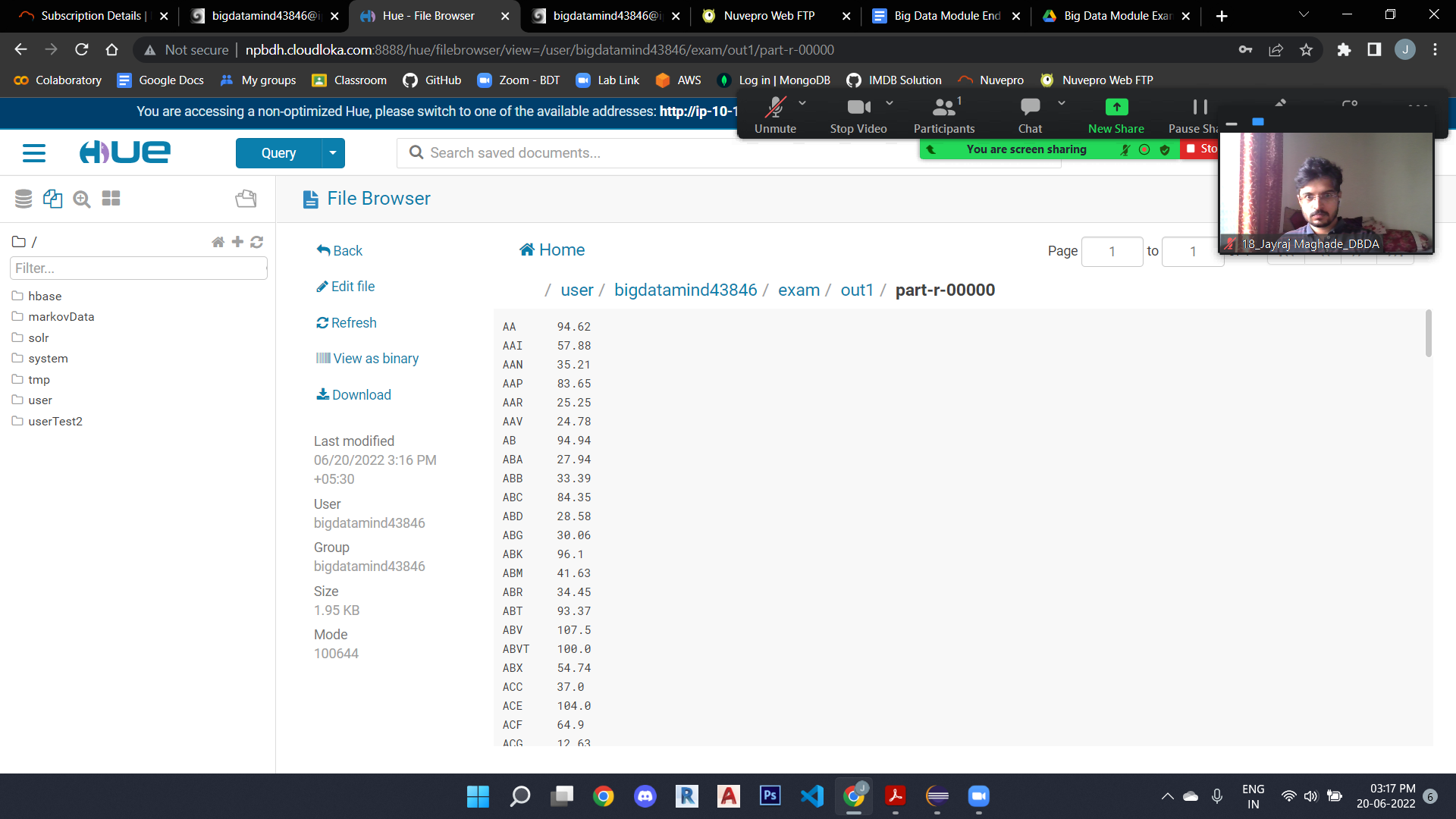
hadoop jar MyJar.jar AllTimeHigh cdac/NYSE.csv exam/out1

AllTimeHigh.class in eclipse -



**Output -(In shell)**

****

**Output -(In Hue)**

**Q.2 HIVE -**

**Load custs.txt on hadoop -**

hadoop fs -put custs.txt

**Load txns1.txt on hadoop -**

hadoop fs -put txns1.txt

**Creating customer table -**

create external table customer(custno string, firstname

String, lastname string, age int,profession string)

row format delimited

fields terminated by ','

stored as textfile

location '/user/bigdatamind43846/custs.txt';

(Data is loaded while creating table)

**Creating table txnrecords-**

create table txnrecords(txnno INT, txndate STRING, custno INT,

amount DOUBLE,

category STRING, product STRING, city STRING, state STRING, spendby STRING)

row format delimited

fields terminated by ','

stored as textfile

location '/user/bigdatamind43846/sales';

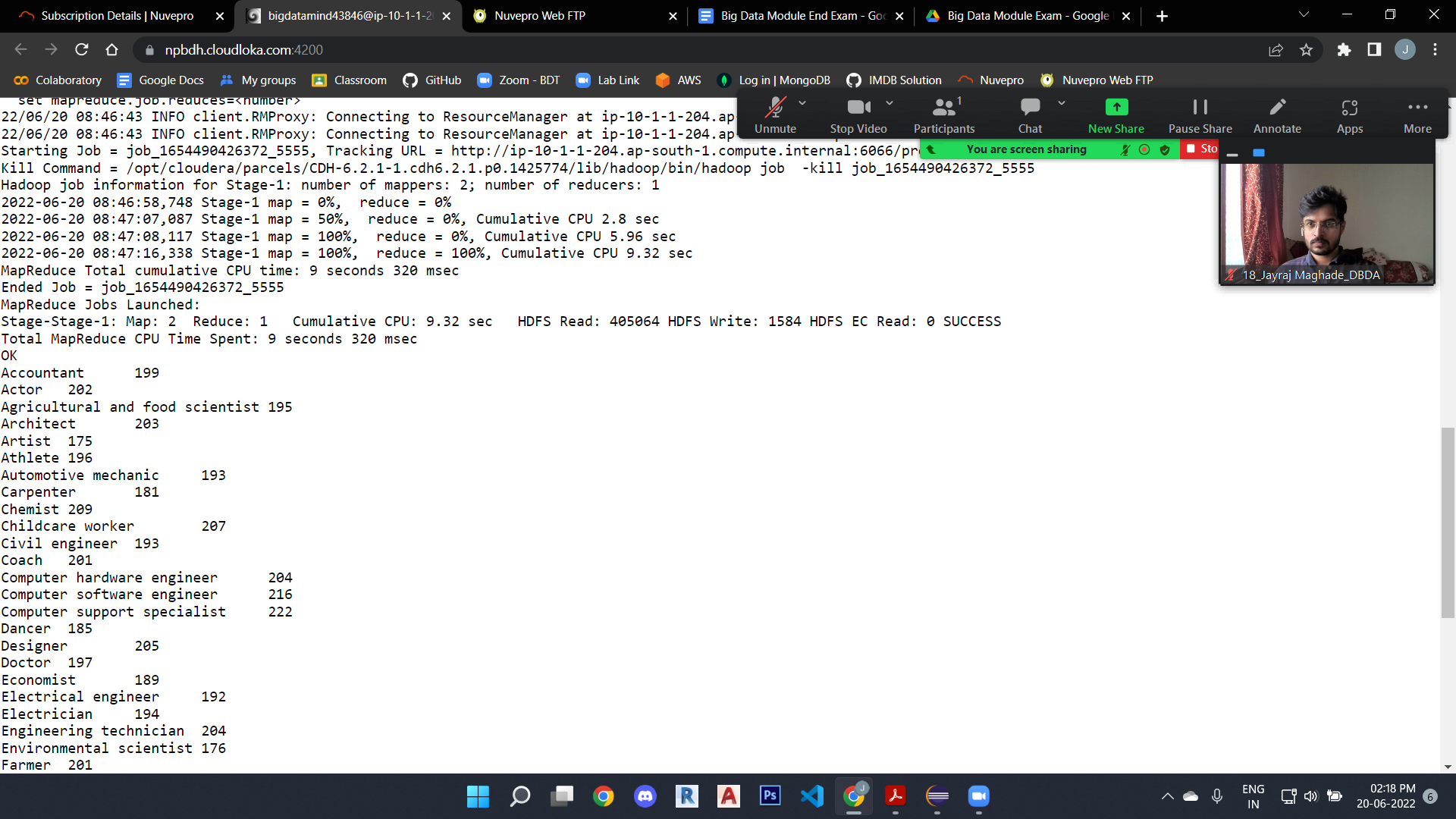
**Load data into txnrecords -**

LOAD DATA LOCAL INPATH 'txns1.txt' OVERWRITE INTO TABLE txnrecords;

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

hive (training\_18)> select profession, count(custno) from customer group by profession;

**Output -**

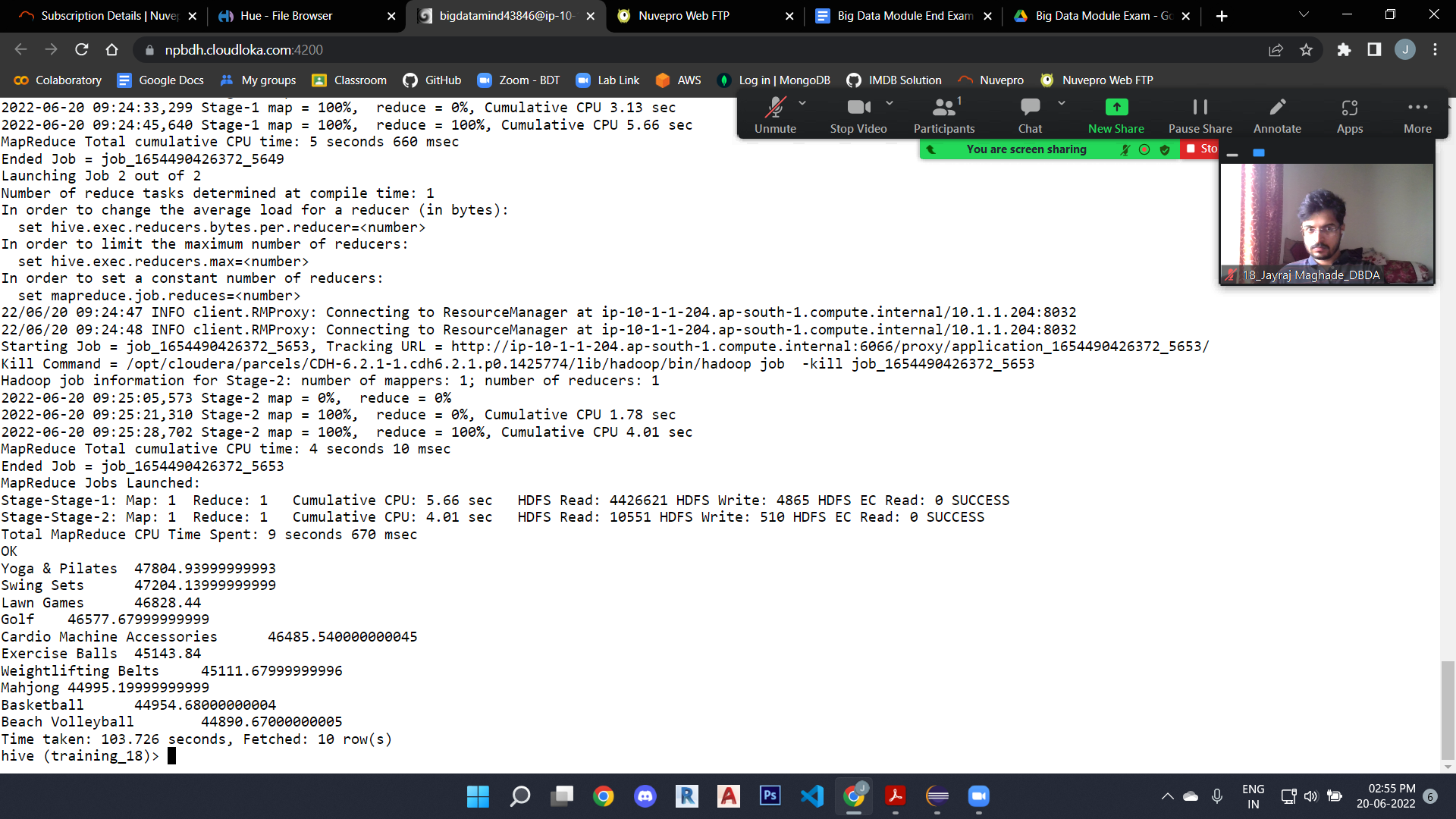


1. **Write a program to find the top 10 products sales wise.**

hive (training\_18)> select product, sum(amount) as total from

txnrecords group by product order by total desc limit 10;

**Output -**



1. **Write a program to create partitioned table on category -**

set hive.exec.dynamicpartition.mode=nonstrict;

set hive.exec.dynamicpartition=true;

create table txnrecsByCat(txnno INT, txndate STRING, custno 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 txnrecsByCat PARTITION(category)

select txn.txnno, txn.txndate,txn.custno, txn.amount,txn.product,txn.city,txn.state, txn.spendby, txn.category from txnrecords txn DISTRIBUTE By category;

**Q.3 Pypspark**

Creating required RDDs -

airlineRDD = sc.textFile("/user/bigdatamind43846/Spark/airlines.csv")

airlineRDD1 = airlineRDD.map(lambda a : a.encode("ascii", "ignore"))

header = airlineRDD1.first()

airlineRDD2 = airlineRDD1.filter(lambda a : a != header)

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

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

schema2 = StructType().add("Year",StringType(),True).add("qtr",IntegerType(),True).add("revenue",DoubleType(),True).add("seats",LongType(),True)

airlinesDF = spark.read.format("csv").option("header","true").schema(schema2).load("/user/bigdatamind43846/Spark/airlines.csv")

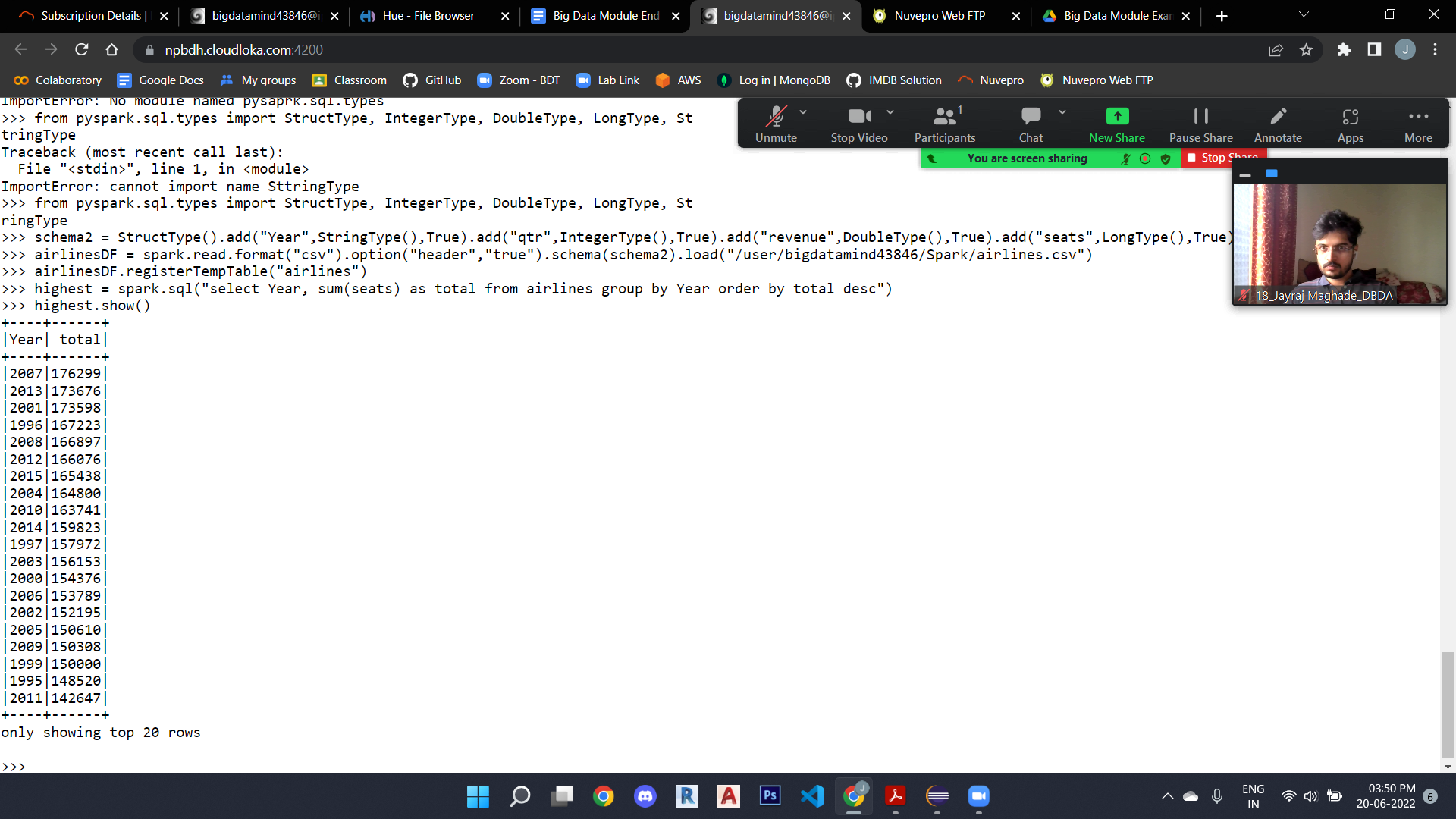
airlinesDF.registerTempTable("airlines")

Questions:

1. highest = spark.sql("select Year, sum(seats) as total from airlines group by Year order by total desc")

highest.show()

Output:



+----+------+

|Year| total|

+----+------+

**|2007|176299|**

|2013|173676|

|2001|173598|

|1996|167223|

|2008|166897|

|2012|166076|

|2015|165438|

|2004|164800|

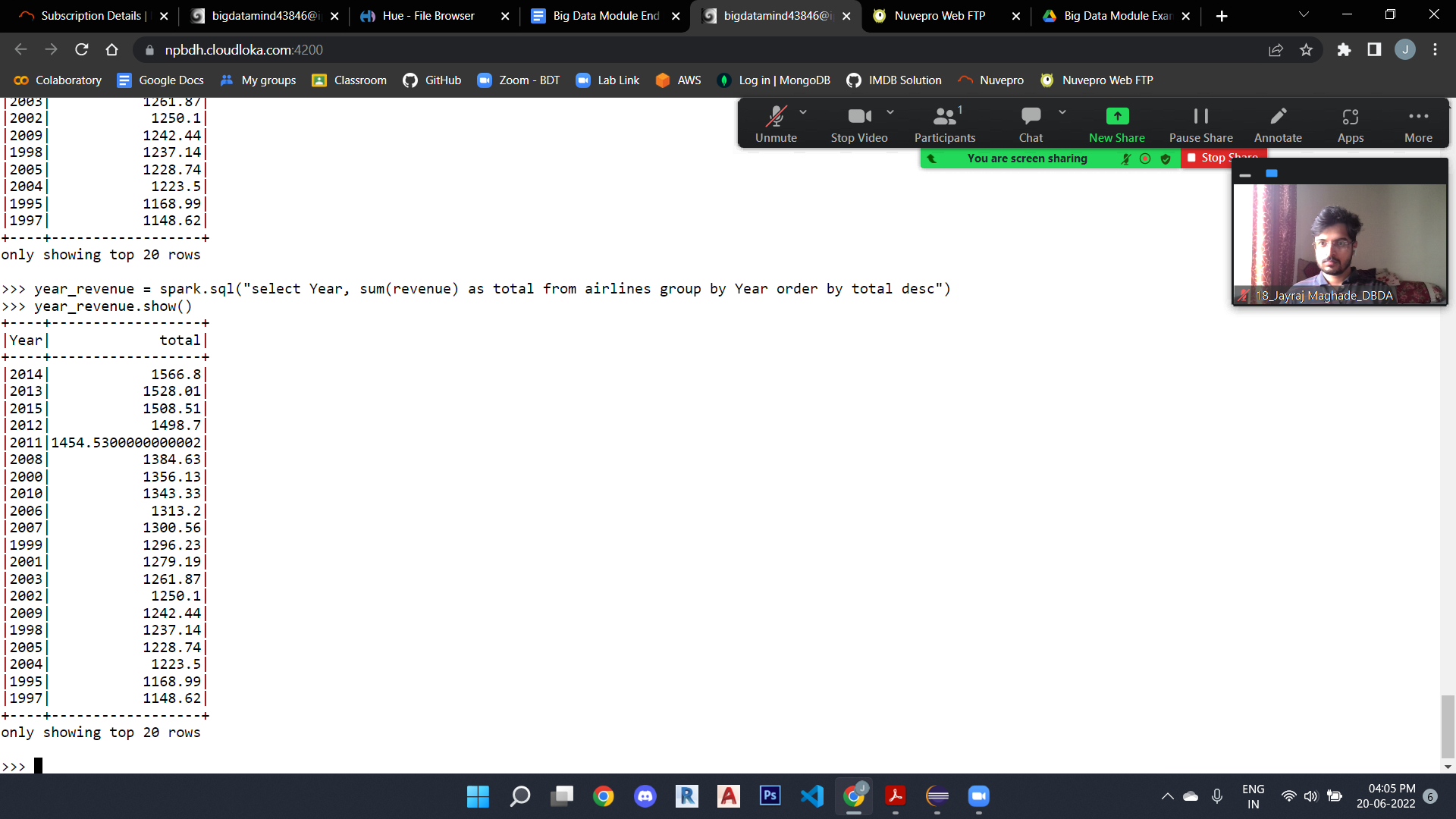
|2010|163741|

1. Identifying the highest revenue generation for which year

year\_revenue = spark.sql("select Year, sum(revenue) as total from airlines group by Year order by total desc")

year\_revenue.show()

**Output -**



+----+------------------+

|Year| total|

+----+------------------+

**|2014| 1566.8|**

|2013| 1528.01|

|2015| 1508.51|

|2012| 1498.7|

|2011|1454.5300000000002|

|2008| 1384.63|

|2000| 1356.13|

|2010| 1343.33|

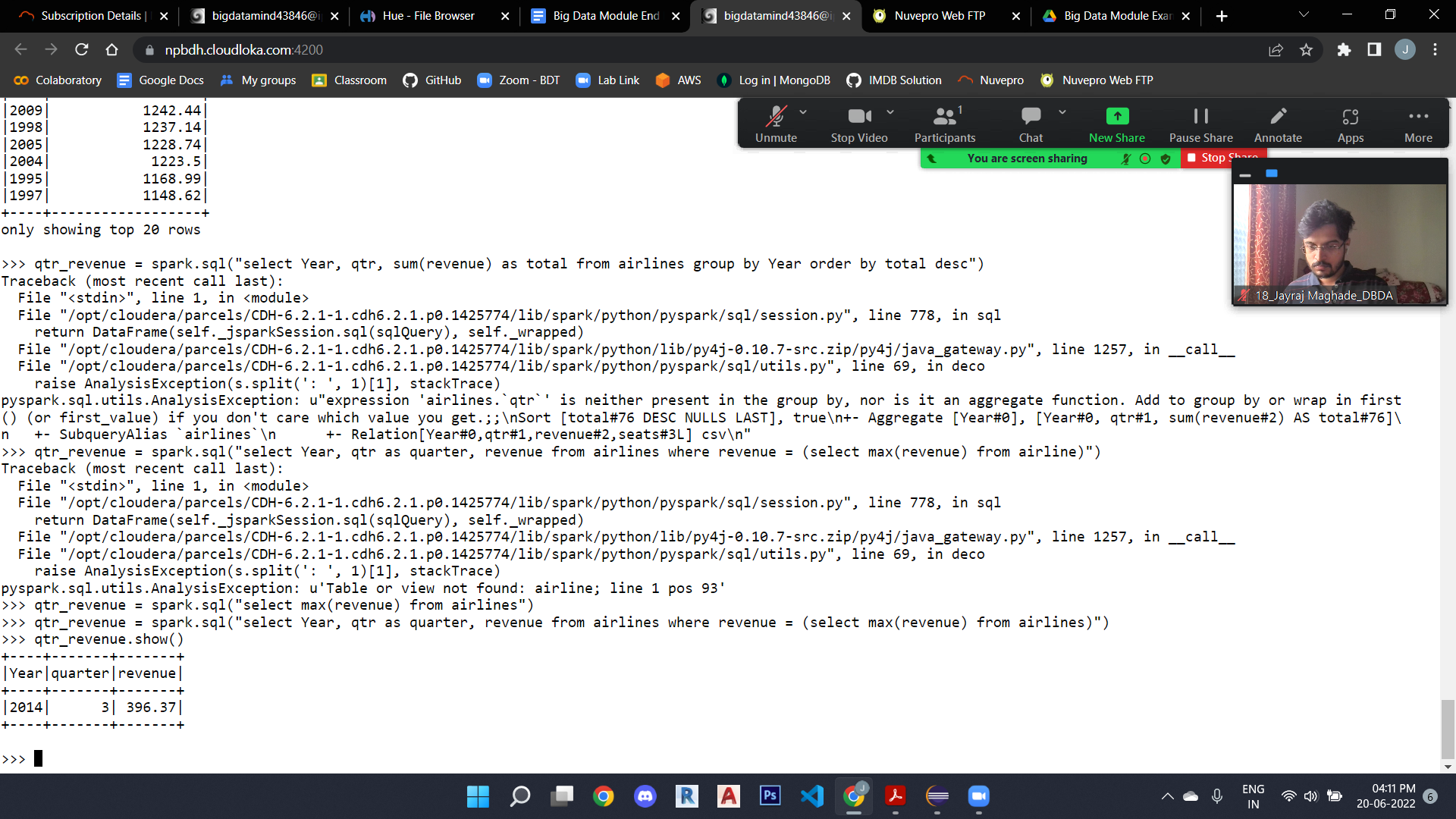
|2006| 1313.2|

1. Identifying the highest revenue generation for which year and quarter (Common group)

qtr\_revenue = spark.sql("select Year, qtr as quarter, revenue from airlines where revenue = (select max(revenue) from airlines)")

qtr\_revenue.show()

**Output -**



+----+-------+-------+

|Year|quarter|revenue|

+----+-------+-------+

**|2014| 3| 396.37|**

+----+-------+-------+