Q1.

**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]));

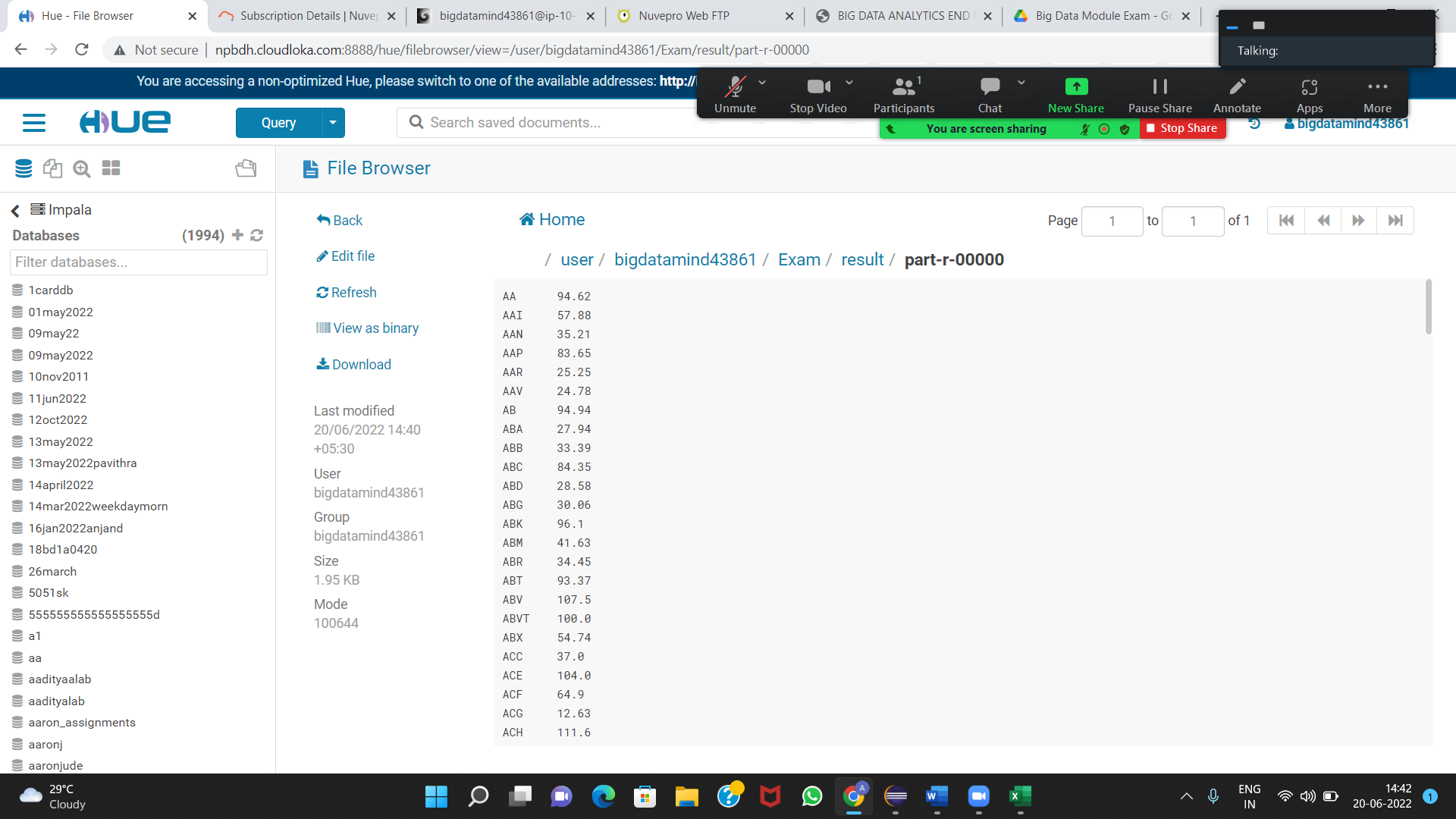
FileOutputFormat.*setOutputPath*(job,**new** Path(args[1]));

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

}

}

hadoop jar myjar.jar AllTimeH cdac/NYSE.csv Exam/result



Q2.

create database cdac\_exam02;

use cdac\_exam02;

create table customer (custno INT,fname STRING,lname STRING, age INT,prof STRING) row format delimited fields terminated by ',' stored as textfile;

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

**1).**

select prof,count(\*) as hcount from customer group by prof order by hcount;

OUTPUT:

Student 1

Social Worker 1

Writer 101

Artist 175

Environmental scientist 176

Carpenter 180

Dancer 185

Therapist 187

Economist 189

Real estate agent 191

Electrical engineer 192

Nurse 192

Automotive mechanic 193

Civil engineer 193

Psychologist 194

Electrician 194

Agricultural and food scientist 195

Athlete 196

Judge 196

Statistician 196

Doctor 197

Financial analyst 198

Accountant 199

Reporter 200

Secretary 200

Coach 201

Farmer 201

Physicist 201

Actor 202

Architect 203

Teacher 204

Engineering technician 204

Computer hardware engineer 204

Designer 205

Musician 205

Childcare worker 207

Veterinarian 208

Chemist 209

Recreation and fitness worker 210

Police officer 210

Pilot 211

Social worker 212

Lawyer 212

Human resources assistant 212

Pharmacist 213

Computer software engineer 216

Firefighter 217

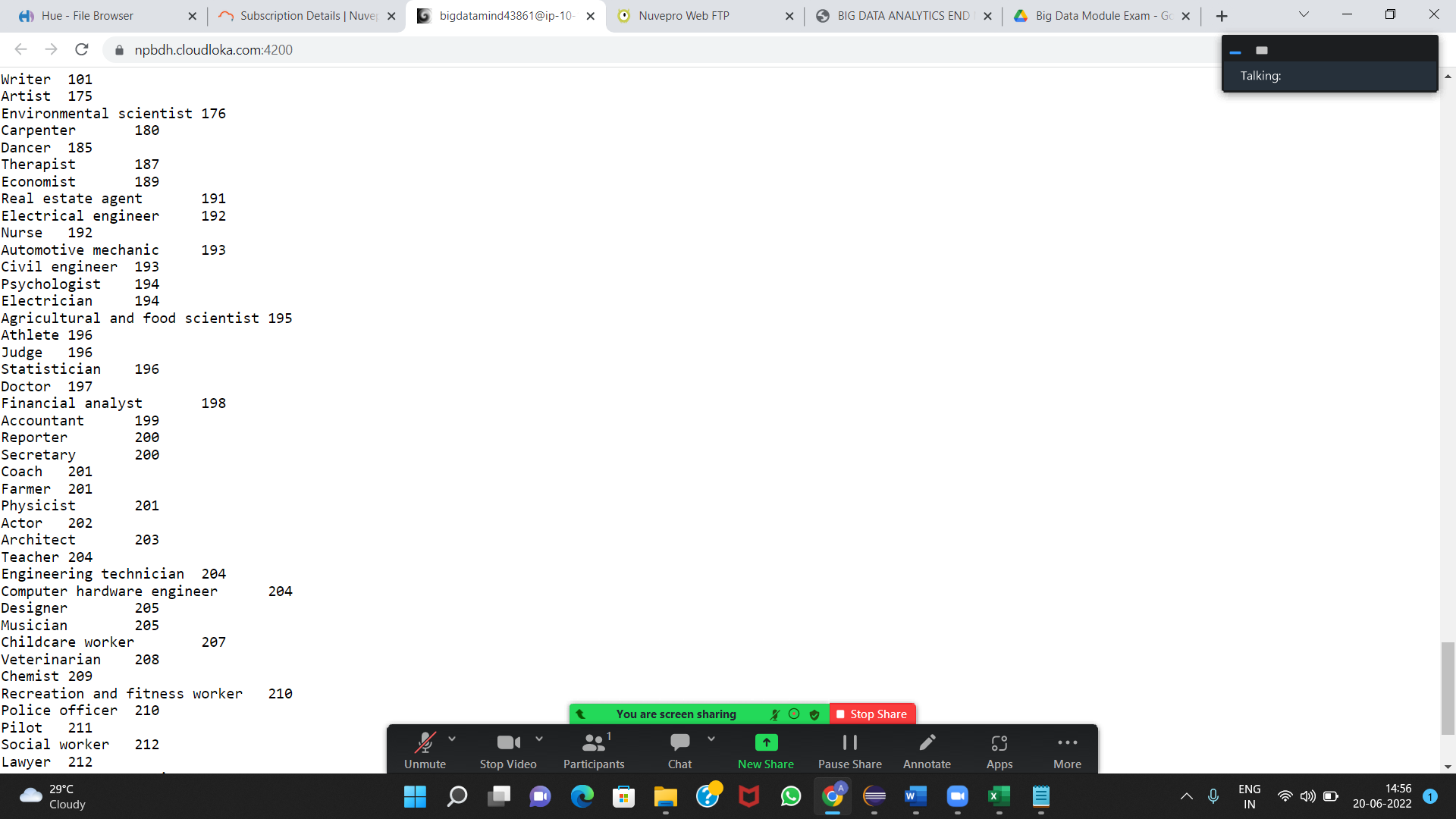
Librarian 218

Loan officer 221

Computer support specialist 222

Photographer 222

Politician 228



**2).**

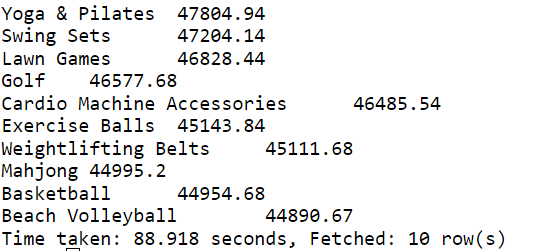
create table txnrecords(txno INT,txndate STRING,custno INT, amount DOUBlE,category STRING, product STRING, city STRING, state STRING, spendby STRING) row format d

elimited fields terminated by ',' stored as textfile;

load data local inpath 'txns1.txt' overwrite into table txnrecords;

select product,round(sum(amount),2) as amt from txnrecords group by product order by amt desc limit 10;s

OUTPUT:



**3).**

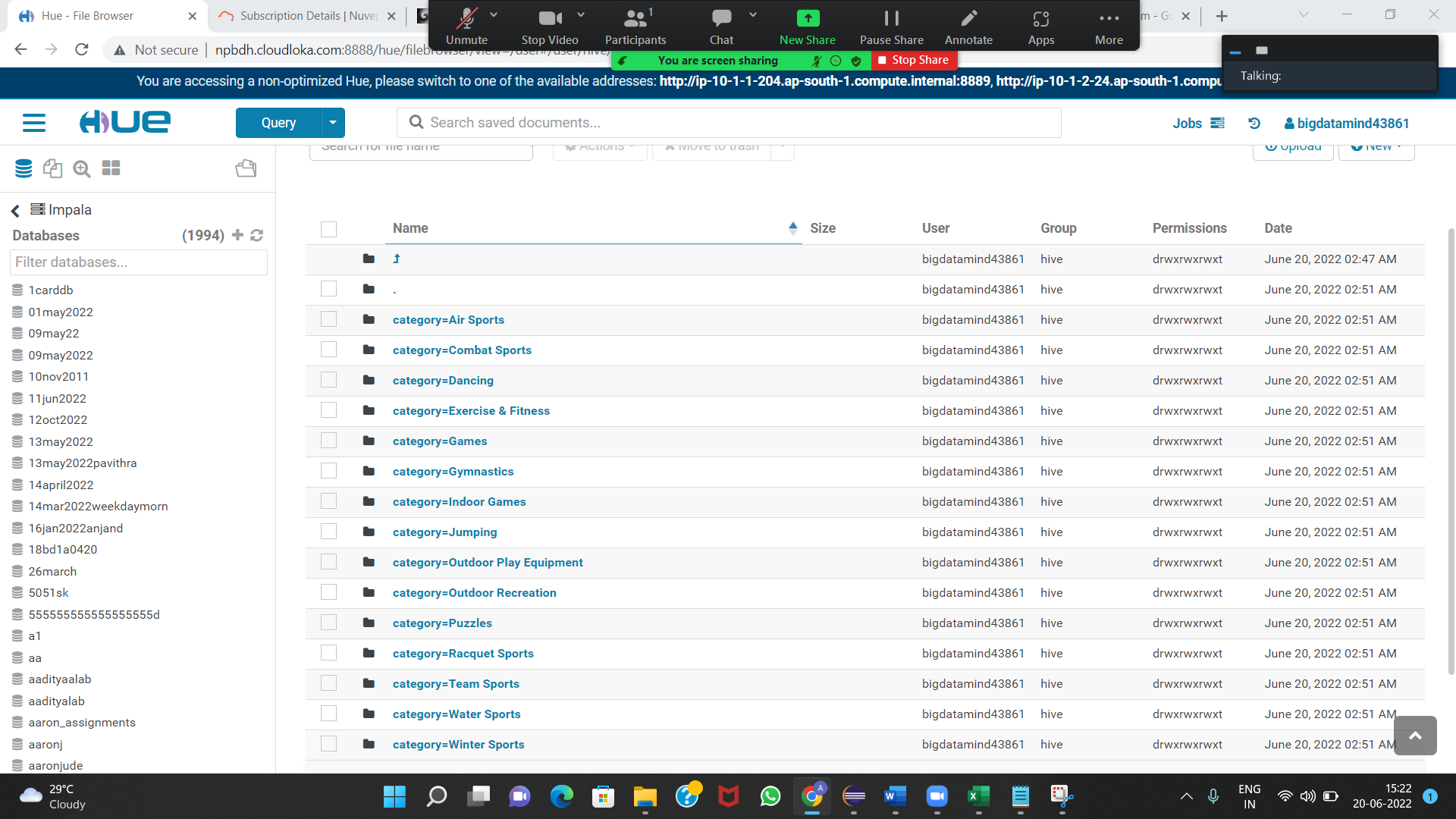
create table txnByCat(txno INT,txndate STRING, custno INT, amount DOUBLE, product STRING, city STRING, state STRING, spendby STRING) partitioned by (category STRI

NG) row format delimited fields terminated by ',' stored as textfile;

insert overwrite table txnByCat partition(category) select t.txno, t.txndate, t.custno, t.amount, t.product,t.city,t.state,t.spendby,t.category from txnrecords t

DISTRIBUTE By category;





Q3.

pyspark

>>>airlineRDD=sc.textFile("/user/bigdatamind43861/airlines.csv")

>>> airlineRDD.count()

85

>>> for line in airlineRDD.take(5):

... print(line)

...

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

>>> 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(","))

Please refer next page.

**1).**

key\_value1=arrayRDD.map(lambda a:(a[0],int(a[3])))

>>> add\_total1=key\_value1.reduceByKey(lambda a,b:a+b)

>>> sortbyval1=add\_total1.sortBy(lambda a:-a[1])

>>> for line in sortbyval1.take(1):

... print(line)

...

('2007', 176299)

*Thus, 176299 (highest) number of people travelled in year 2007.*

**2).**

key\_value=arrayRDD.map(lambda a:(a[0],float(a[2])\*int(a[3])))

>>> add\_total=key\_value.reduceByKey(lambda a,b:a+b)

>>> sortbyval=add\_total.sortBy(lambda a : -a[1])

>>> for line in sortbyval.take(1):

... print(line)

...

('2013', 66363208.71)

*Thus,* 66363208.71 *(highest) revenue was generated in year 2007.*

**3).**

>>> key\_value3=arrayRDD.map(lambda a:(a[0]+","+a[1],float(a[2])\*int(a[3])))

>>> add\_total3=key\_value3.reduceByKey(lambda a,b:a+b)

>>> sortbyval3=add\_total3.sortBy(lambda a : -a[1])

>>> for line in sortbyval3.take(1):

... print(line)

...

('2014,4', 18819408.48)

*Thus,* 18819408.48 *(highest) revenue was generated in year 2004 and 4th Quarter.*