Q1. MapReduce Problem Statement Here, we have chosen the stock market dataset on which we have performed map-reduce operations. Following is the structure of the data. Kindlyfind the solutions to the questions below.

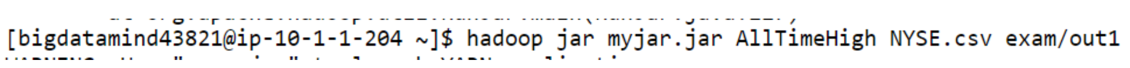
Q 1.Find all time High price for each stock

ANSWER:-

```
import java.io.*;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.io.LongWritable;
import org.apache.hadoop.io.DoubleWritable;
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 AllTimeHigh {

      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(new Text(str[1]),new LongWritable(vol));
                    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 maxValue=0;
                        double temp_val=0;

                        for (DoubleWritable value : values) {
                              temp_val = value.get();
                              if (temp_val > maxValue) {
                                    maxValue = temp_val;
                              }
                        }
                        result.set(maxValue);

                  context.write(key, result);
                  //context.write(key, new LongWritable(sum));

                }
         }
        public static void main(String[] args) throws Exception {
                Configuration conf = new Configuration();
                //conf.set("name", "value")
                //conf.set("mapreduce.input.fileinputformat.split.minsize",
"134217728");
                Job job = Job.getInstance(conf, "Highest Price for each
stock");
                job.setJarByClass(AllTimeHigh.class);
                job.setMapperClass(MapClass.class);
                //job.setCombinerClass(ReduceClass.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);
              }
}
```

COMMAND : -



```
[bigdatamind43821@ip-10-1-1-204 ~]$ hadoop jar myjar.jar AllTimeHigh NYSE.csv exam/out1
```

```
AA        94.62
AAI       57.88
AAN       35.21
AAP       83.65
AAR       25.25
AAV       24.78
AB        94.94
ABA       27.94
```

ABB	33.39
ABC	84.35
ABD	28.58
ABG	30.06
ABK	96.1
ABM	41.63
ABR	34.45
ABT	93.37
ABV	107.5
ABVT	100.0
ABX	54.74
ACC	37.0
ACE	104.0
ACF	64.9
ACG	12.63
ACH	111.6
ACI	112.89
ACL	178.56
ACM	38.25
ACN	44.03
ACO	42.7
ACS	109.55
ACV	65.32
ADC	37.7
ADI	185.5
ADM	48.95
ADP	84.31
ADS	80.79
ADX	40.56
ADY	44.0
AEA	23.94
AEB	26.5
AEC	17.6
AED	26.12
AEE	56.77
AEF	27.0
AEG	148.32
AEH	26.64
AEL	14.6
AEM	83.45
AEO	88.13

```
AEP       53.31
AER       32.82
AES       92.5
AET       154.67
AEV       26.78
AF        63.09
AFB       17.03
AFC       25.15
AFE       26.7
AFF       25.15
AFG       54.65
AFL       74.94
AFN       11.99
AGC       20.2
AGCO      71.95
AGD       25.5
AGL       44.67
AGM       80.0
AGN       125.0
AGO       31.99
AGP       80.89
AGU       113.88
AHC       16.35
AHD       47.12
AHL       30.8
AHS       37.4
AHT       13.48
AI        28.7
AIB       125.0
AIG       157.19
AIN       43.62
AIQ       15.4
AIR       46.58
AIT       59.0
AIV       65.79
AIZ       71.31
AJG       68.5
AKF       26.42
AKP       17.45
AKR       29.0
AKS       73.07
```

```
AKT         26.25

ALB         86.52

ALC         27.86

ALD         33.35

ALE         51.7

ALEX        44.52

ALF         26.75

ALG         29.23

ALJ         47.1

ALK         62.56

ALL         100.25

ALM         27.79

ALQ         28.5

ALU         86.25

ALV         65.09

ALX         471.0

ALY         28.1

ALZ         26.25

AM          62.88

AMB         66.86

AMD         97.0

AME         53.12

AMG         136.51

AMN         138.65

AMP         69.25

AMR         69.01

AMT         55.5

AMX         69.15

AN          53.93

ANF         101.5

ANH         16.65

ANN         53.06

ANR         119.3

ANW         48.63

AOB         14.48

AOD         21.85

AOI         23.38

AOL         27.0

AON         75.56

AOS         58.06

AP          54.46
```

```
APA       149.23

APB       36.14

APC       113.95

APD       106.06

APF       24.53

APH       121.06

APL       56.88

APU       42.94

APX       12.38

ARB       55.63

ARD       71.08

ARE       116.5

ARG       65.45

ARI       19.2

ARJ       48.02

ARK       8.29

ARL       22.25

ARM       32.5

ARO       47.82

ARP       39.0

ART       28.1

ARW       64.12

ASA       92.6

ASF       89.12

ASG       12.56

ASH       76.25

ASI       24.21

ASP       16.87

ASR       63.54

ASX       7.49

ATE       58.02

ATI       119.7

ATK       120.9

ATO       33.47

ATR       76.98

ATT       27.14

ATU       70.17

ATV       32.33

ATW       126.92

AU        62.2

AUO       28.5
```

```
AUY        19.93

AV         15.05

AVA        67.76

AVB        149.94

AVD        51.0

AVF        27.0

AVK        29.75

AVP        90.45

AVT        81.12

AVX        100.0

AVY        78.5

AWC        32.85

AWF        15.46

AWH        53.48

AWI        57.48

AWK        23.77

AWP        20.55

AWR        48.0

AXA        80.94

AXE        88.4

AXL        42.1

AXP        169.5

AXR        149.99

AXS        43.35

AYE        65.48

AYI        66.89

AYN        15.42

AYR        41.31

AZN        145.41

AZO        169.99

AZZ        59.2
```

Hive:

Please find the customer data set.

cust id

firstname

lastname

age

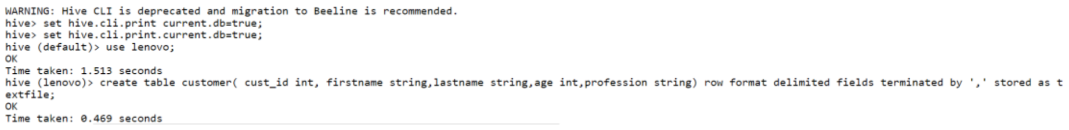
profession

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

Answer:-

Table creation command

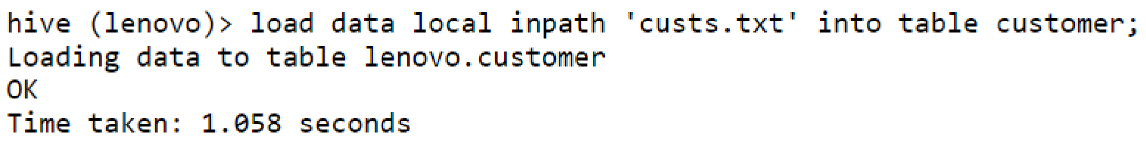
create table customer(cust_id int, firstname string,lastname string,age int,profession string) row format delimited fields terminated by ',' stored as textfile;



```
WARNING: Hive CLI is deprecated and migration to Beeline is recommended.
hive> set hive.cli.print current.db=true;
hive> set hive.cli.print.current.db=true;
hive (default)> use lenovo;
OK
Time taken: 1.513 seconds
hive (lenovo)> create table customer( cust_id int, firstname string,lastname string,age int,profession string) row format delimited fields terminated by ',' stored as t
extfile;
OK
Time taken: 0.469 seconds
```

Load Data Command:

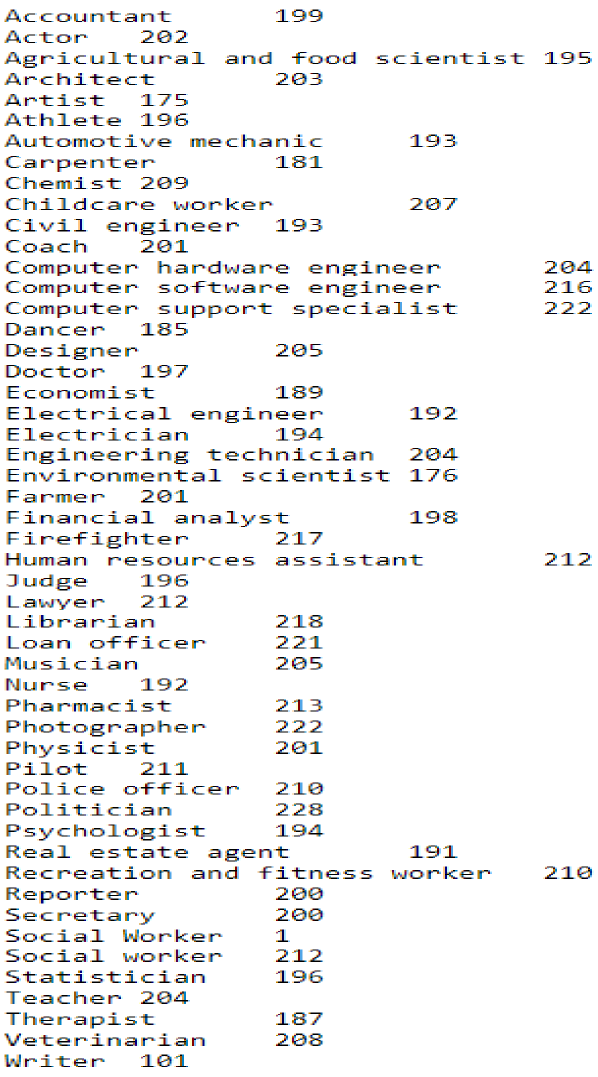
```
load data local inpath 'custs.txt' into table customer;
```



```
hive (lenovo)> load data local inpath 'custs.txt' into table customer;
Loading data to table lenovo.customer
OK
Time taken: 1.058 seconds
```

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

```
select profession, count(cust_id) from customer group by
profession;
```



```
Accountant      199
Actor   202
Agricultural and food scientist 195
Architect       203
Artist  175
Athlete 196
Automotive mechanic     193
Carpenter       181
Chemist 209
Childcare worker        207
Civil engineer  193
Coach   201
Computer hardware engineer      204
Computer software engineer      216
Computer support specialist     222
Dancer  185
Designer        205
Doctor  197
Economist       189
Electrical engineer     192
Electrician     194
Engineering technician  204
Environmental scientist 176
Farmer  201
Financial analyst       198
Firefighter     217
Human resources assistant       212
Judge   196
Lawyer  212
Librarian       218
Loan officer    221
Musician        205
Nurse   192
Pharmacist      213
Photographer    222
Physicist       201
Pilot   211
Police officer  210
Politician      228
Psychologist    194
Real estate agent       191
Recreation and fitness worker   210
Reporter        200
Secretary       200
Social Worker   1
Social worker   212
Statistician    196
Teacher 204
Therapist       187
Veterinarian    208
Writer  101
```

Please find the sales data set.

txn id ,txn date, cust id, amount, category ,product ,city, state ,spendby

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

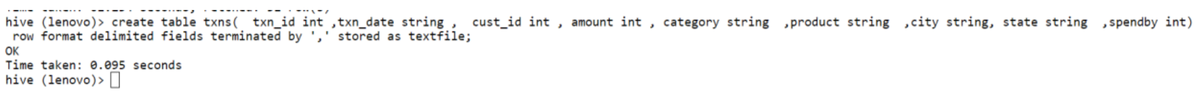
3) Write a program to create partiioned table on category

Answer:-

Table creation command

create table txns(txn_id int ,txn_date string , cust_id int , amount int , category string ,product string ,city string, state string ,spendby int)

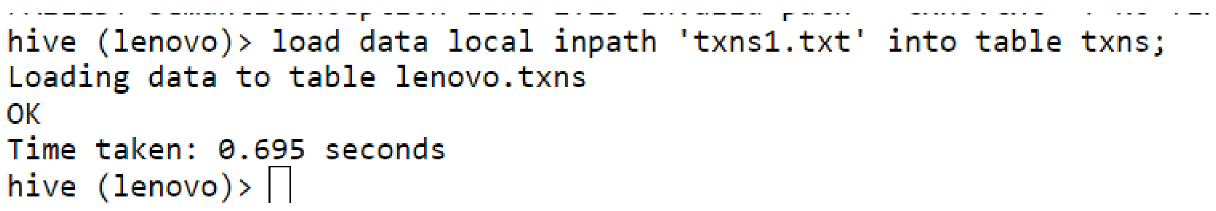
row format delimited fields terminated by ',' stored as textfile;



```
hive (lenovo)> create table txns(  txn_id int ,txn_date string ,  cust_id int , amount int , category string  ,product string  ,city string, state string  ,spendby int)
 row format delimited fields terminated by ',' stored as textfile;
OK
Time taken: 0.095 seconds
hive (lenovo)> 
```

Load Data Command:

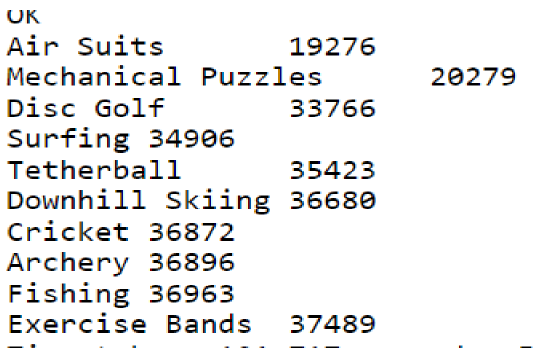
load data local inpath 'txns.txt' into table txns;



```
hive (lenovo)> load data local inpath 'txns1.txt' into table txns;
Loading data to table lenovo.txns
OK
Time taken: 0.695 seconds
hive (lenovo)> 
```

Write a program to find the top 10 products sales wise

```
select product,sum(amount) as sales from txns group by
product order by sales limit 10;
```



```
OK
Air Suits       19276
Mechanical Puzzles      20279
Disc Golf       33766
Surfing 34906
Tetherball      35423
Downhill Skiing 36680
Cricket 36872
Archery 36896
Fishing 36963
Exercise Bands  37489
```

3) Write a program to create partiioned table on category

Commands:-

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

set hive.exec.dynamic.partition=true;

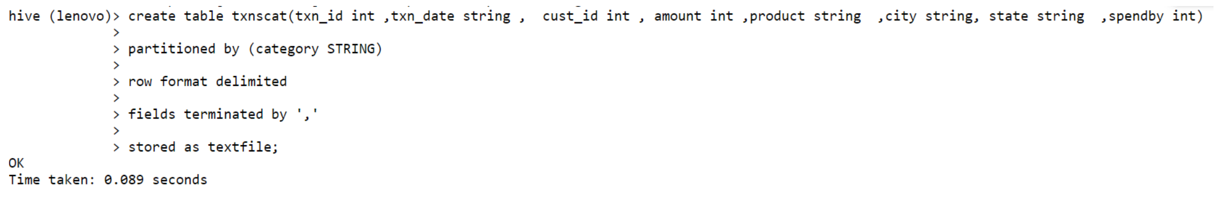
Table creation command

create table txnscat(txn_id int ,txn_date string , cust_id int , amount int ,product string ,city string, state string ,spendby int) partitioned by (category STRING)

row format delimited

fields terminated by ','

stored as textfile;

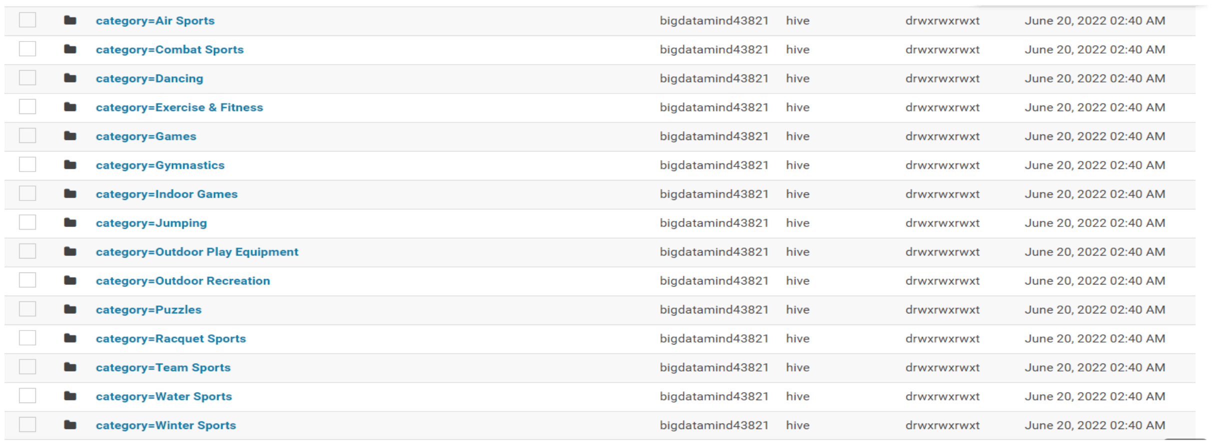


```
hive (lenovo)> create table txnscat(txn_id int ,txn_date string ,  cust_id int , amount int ,product string  ,city string, state string  ,spendby int)
             >
             > partitioned by (category STRING)
             >
             > row format delimited
             >
             > fields terminated by ','
             >
             > stored as textfile;
OK
Time taken: 0.089 seconds
```

Load Data Command:

INSERT OVERWRITE TABLE txnscat PARTITION(category) select txn.txn_id, txn.txn_date,txn.cust_id, txn.amount,txn.product,txn.city,txn.state, txn.spendby, txn.category from txns txn

DISTRIBUTE By category;



category=Air Sports	bigdatamind43821	hive	drwxrwxrwxt	June 20, 2022 02:40 AM
category=Combat Sports	bigdatamind43821	hive	drwxrwxrwxt	June 20, 2022 02:40 AM
category=Dancing	bigdatamind43821	hive	drwxrwxrwxt	June 20, 2022 02:40 AM
category=Exercise & Fitness	bigdatamind43821	hive	drwxrwxrwxt	June 20, 2022 02:40 AM
category=Games	bigdatamind43821	hive	drwxrwxrwxt	June 20, 2022 02:40 AM
category=Gymnastics	bigdatamind43821	hive	drwxrwxrwxt	June 20, 2022 02:40 AM
category=Indoor Games	bigdatamind43821	hive	drwxrwxrwxt	June 20, 2022 02:40 AM
category=Jumping	bigdatamind43821	hive	drwxrwxrwxt	June 20, 2022 02:40 AM
category=Outdoor Play Equipment	bigdatamind43821	hive	drwxrwxrwxt	June 20, 2022 02:40 AM
category=Outdoor Recreation	bigdatamind43821	hive	drwxrwxrwxt	June 20, 2022 02:40 AM
category=Puzzles	bigdatamind43821	hive	drwxrwxrwxt	June 20, 2022 02:40 AM
category=Racquet Sports	bigdatamind43821	hive	drwxrwxrwxt	June 20, 2022 02:40 AM
category=Team Sports	bigdatamind43821	hive	drwxrwxrwxt	June 20, 2022 02:40 AM
category=Water Sports	bigdatamind43821	hive	drwxrwxrwxt	June 20, 2022 02:40 AM
category=Winter Sports	bigdatamind43821	hive	drwxrwxrwxt	June 20, 2022 02:40 AM

QUESTION 3

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?

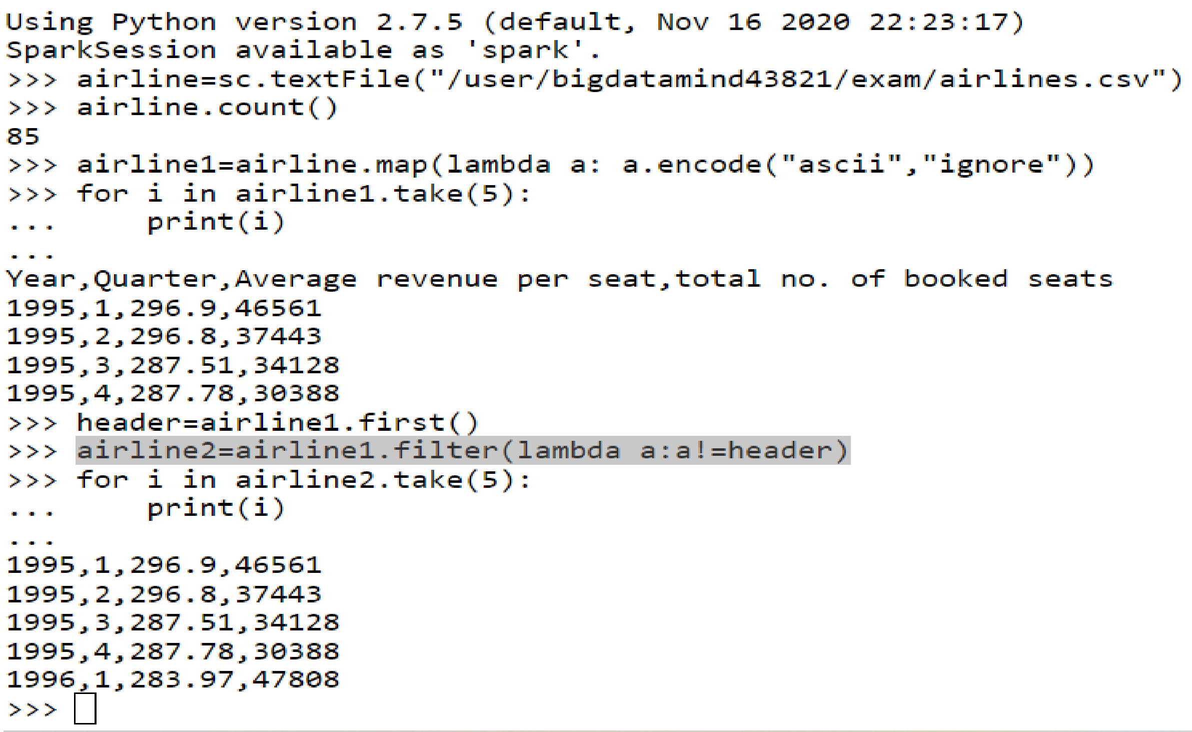
2) Identifying the highest revenue generation for which year

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

Answer:-

Creation of RDD:-

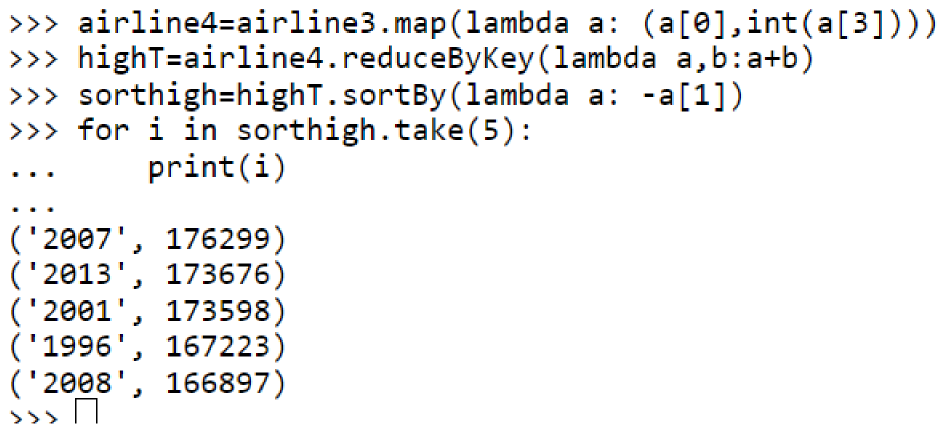
```
airline=sc.textFile("/user/bigdatamind43821/exam/airlines.csv")

airline1=airline.map(lambda a: a.encode("ascii","ignore"))

header=airline1.first()

airline2=airline1.filter(lambda a:a!=header)
```



```
Using Python version 2.7.5 (default, Nov 16 2020 22:23:17)
SparkSession available as 'spark'.
>>> airline=sc.textFile("/user/bigdatamind43821/exam/airlines.csv")
>>> airline.count()
85
>>> airline1=airline.map(lambda a: a.encode("ascii","ignore"))
>>> for i in airline1.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=airline1.first()
>>> airline2=airline1.filter(lambda a:a!=header)
>>> for i in airline2.take(5):
...     print(i)
...
1995,1,296.9,46561
1995,2,296.8,37443
1995,3,287.51,34128
1995,4,287.78,30388
1996,1,283.97,47808
>>> 
```

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

```
airline4=airline3.map(lambda a: (a[0],int(a[3])))

highT=airline4.reduceByKey(lambda a,b:a+b)

sorthigh=highT.sortBy(lambda a: -a[1])

for i in sorthigh.take(5):
      print(i)
```



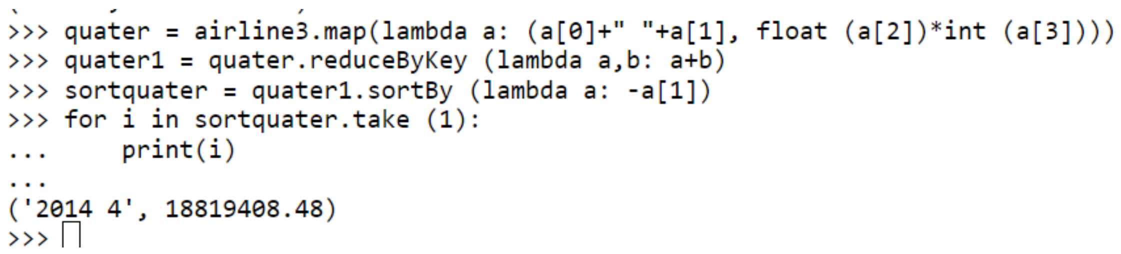
```
>>> airline4=airline3.map(lambda a: (a[0],int(a[3])))
>>> highT=airline4.reduceByKey(lambda a,b:a+b)
>>> sorthigh=highT.sortBy(lambda a: -a[1])
>>> for i in sorthigh.take(5):
...     print(i)
...
('2007', 176299)
('2013', 173676)
('2001', 173598)
('1996', 167223)
('2008', 166897)
>>> 
```

2) Identifying the highest revenue generation for which year

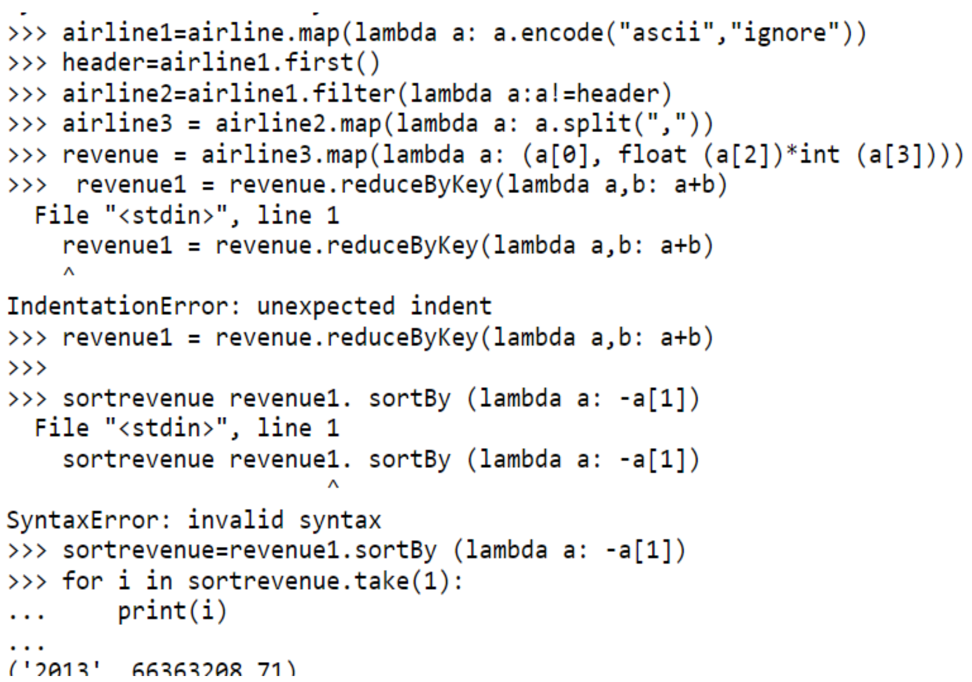
```
revenue = airline3.map(lambda a: (a[0], float (a[2])*int (a[3])))

revenue1 = revenue.reduceByKey(lambda a,b: a+b)

sortrevenue revenue1. sortBy (lambda a: -a[1])

for i in sortrevenue.take(1):

print(i)
```

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

```
quater = airline3.map(lambda a: (a[0]+" "+a[1], float (a[2])*int (a[3])))
quater1 = quater.reduceByKey (lambda a,b: a+b)
sortquater = quater1.sortBy (lambda a: -a[1])
for i in sortquater.take (1):
print(i)
```



```
>>> quater = airline3.map(lambda a: (a[0]+" "+a[1], float (a[2])*int (a[3])))
>>> quater1 = quater.reduceByKey (lambda a,b: a+b)
>>> sortquater = quater1.sortBy (lambda a: -a[1])
>>> for i in sortquater.take (1):
...     print(i)
...
('2014 4', 18819408.48)
>>> 
```



```
>>> airline1=airline.map(lambda a: a.encode("ascii","ignore"))
>>> header=airline1.first()
>>> airline2=airline1.filter(lambda a:a!=header)
>>> airline3 = airline2.map(lambda a: a.split(","))
>>> revenue = airline3.map(lambda a: (a[0], float (a[2])*int (a[3])))
>>>  revenue1 = revenue.reduceByKey(lambda a,b: a+b)
  File "<stdin>", line 1
    revenue1 = revenue.reduceByKey(lambda a,b: a+b)
    ^
IndentationError: unexpected indent
>>> revenue1 = revenue.reduceByKey(lambda a,b: a+b)
>>>
>>> sortrevenue revenue1. sortBy (lambda a: -a[1])
  File "<stdin>", line 1
    sortrevenue revenue1. sortBy (lambda a: -a[1])
                       ^
SyntaxError: invalid syntax
>>> sortrevenue=revenue1.sortBy (lambda a: -a[1])
>>> for i in sortrevenue.take(1):
...     print(i)
...
('2013', 66363208.71)
```