

**CDAC MUMBAI**  
**PG-DBDA SEP 2022 BATCH KHARGHAR**  
**MODULE: BIG DATA ANALYTICS**  
**DATE : 14<sup>TH</sup> DEC, 2022**  
**MARKS : 40 MARKS**

**Please create a doc/txt/pdf file with 12 digits student id, which will contain the code along with the screenshots of the output or result. While taking the screenshot make sure that you are visible in all the images.**

-----

**Q1.**

**MapReduce**

**Problem Statement**

**[10 marks]**

Here, we have chosen the stock market dataset on which we have performed map-reduce operations. Following is the structure of the data. Kindly Find the solutions to the questions below.

Data Structure

1. Exchange Name
  - 2 Stock symbol
  3. Transaction date
  4. Opening price of the stock
  5. Intra day high price of the stock
  6. Intra day low price of the stock
  7. Closing price of the stock
  8. Total Volume of the stock on the particular day
  9. Adjustment Closing price of the stock
- Field Separator – comma

**Question 2 : Find all time High price for each stock****[15 marks]**

```
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.DoubleWritable;
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, DoubleWritable>{
        public void map(LongWritable key, Text value,Context context)throws
IOException,InterruptedException {
            String[] str = value.toString().split(",");
            String stk_id = str[1];
            double high = Double.parseDouble(str[4]);
            context.write(new Text(stk_id), new DoubleWritable(high));
        }
    }

    public static class ReduceClass extends Reducer<Text,DoubleWritable,Text,DoubleWritable> {

        public void reduce(Text key, Iterable<DoubleWritable> values,Context context)throws
IOException,InterruptedException {
            double max = 0;
            for (DoubleWritable val:values) {
                if (val.get() > max) {
                    max = val.get();
                }
            }
            context.write(key, new DoubleWritable(max));
        }
    }

    public static void main(String[] args) throws Exception {
        Configuration conf = new Configuration();
        Job job = Job.getInstance(conf, "All Time High");
        job.setJarByClass(allTimeHigh.class);
        job.setMapperClass(MapClass.class);
    }
}
```

```

job.setReducerClass(ReduceClass.class);
job.setNumReduceTasks(1);
job.setMapOutputKeyClass(Text.class);
job.setMapOutputValueClass(DoubleWritable.class);
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);
}
}

```

The screenshot shows the Hue web interface in a browser window. The address bar indicates the URL is `http://ip-10-1-1-204.ap-south-1.compute.internal:8889`. The interface includes a top navigation bar with the Hue logo and a search bar. The main content area is titled "File Browser" and shows a directory structure: `/ user / bigcdac432589 / EXAM / que1out / part-r-00000`. A table of data is displayed, with columns for customer identifiers and numerical values. A video call window is visible in the top right corner, showing a participant named "04\_Abhinav Prakash\_08DA". At the bottom, a Windows taskbar is visible with the date "14-12-2022" and time "15:09".

Customer ID	Value
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	38.06
ABK	96.1
ABM	41.63
ABR	
ABT	
ABV	

## Hive

Please find the customer data set.

cust id

firstname

lastname

age

profession

---

```
CREATE TABLE customer (
```

```
  custno 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;
```

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

```
SELECT profession , count(DISTINCT custno) AS num_of_cust FROM  
customer GROUP BY profession ORDER BY num_of_cust DESC;
```

The screenshot shows a web browser window with multiple tabs. The active tab displays a terminal window with the following output:

```
Ended: 500 - job_1005041244711_22002
MapReduce Jobs Launched:
Stage-Stage-1: Map: 1 Reduce: 1 Cumulative CPU: 5.75 sec HDFS Read: 400112 HDFS Write: 1746 HDFS EC Read: 0 SUCCESS
Stage-Stage-2: Map: 1 Reduce: 1 Cumulative CPU: 6.36 sec HDFS Read: 7301 HDFS Write: 1584 HDFS EC Read: 0 SUCCESS
Total MapReduce CPU Time Spent: 12 seconds 110 msec
OK
Politician 228
Computer support specialist 222
Photographer 222
Loan officer 221
Librarian 218
Firefighter 217
Computer software engineer 216
Pharmacist 213
Social worker 212
Human resources assistant 212
Pilot 212
Lawyer 212
Recreation and fitness worker 210
Police officer 210
Chemist 209
Veterinarian 208
Childcare worker 207
Designer 205
Musician 205
Engineering technician 204
Teacher 204
Computer hardware engineer 204
Architect 203
Actor 202
Physicist 201
Coach 201
```

Overlaid on the right side of the terminal is a video call window showing a man with glasses and a white shirt. Below the video call is a control bar with buttons: Unmute, Stop Video, Participants, Chat, New Share, Pause Share, Annotate, Apps, and More. A green banner at the bottom of the video call says "You are screen sharing" and a red button says "Stop Share".

The Windows taskbar at the bottom shows the date and time as 15:16 on 14-12-2022, along with various system icons and the language set to ENG IN.

---

Please find the sales data set.

txn id

txn date

cust id

amount

category

product

city

state

spendby

---

Creating Table:---

CREATE TABLE sales (

txn\_id int,

txn\_date String,

Custno int,

amount float,

category String,

product String,

city String,

state String,

spendby String

)

Row format delimited

Fields terminated by ','

Stored as textFile;

Loading Data into table:-----

Load data local inpath 'txns1.txt' overwrite into table sales;

---

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

SELECT product, sum(amount) as total\_sales from sales GROUP BY product ORDER BY total\_sales DESC LIMIT 10;

The screenshot shows a web browser window with multiple tabs. The active tab displays a Hadoop job execution log for job\_id 1663041244711\_22852. The log includes the following information:

- Command: `/opt/cloudera/parcels/CDH-6.2.1-1.cdh6.2.1.p0.1425774/lib/hadoop/bin/hadoop job -kill job_1663041244711_22852`
- Hadoop job information for Stage-2: number of mappers: 1; number of reducers: 1
- Progress logs for Stage-2 map and reduce tasks, showing cumulative CPU time.
- MapReduce Total cumulative CPU time: 4 seconds 910 msec
- Ended Job = job\_1663041244711\_22852
- MapReduce Jobs Launched:
- Stage-Stage-1: Map: 1 Reduce: 1 Cumulative CPU: 7.31 sec HDFS Read: 4426878 HDFS Write: 4865 HDFS EC Read: 0 SUCCESS
- Stage-Stage-2: Map: 1 Reduce: 1 Cumulative CPU: 4.91 sec HDFS Read: 10542 HDFS Write: 528 HDFS EC Read: 0 SUCCESS
- Total MapReduce CPU Time Spent: 12 seconds 220 msec
- OK
- Yoga & Pilates 47804.94006872177
- Swing Sets 47204.14009475708
- Lawn Games 46828.43995523453
- Golf 46577.6800737381
- Cardio Machine Accessories 46485.54000759125
- Exercise Balls 45143.839904785156
- Weightlifting Belts 45111.67999744415
- Mahjong 44995.20000743866
- Basketball 44954.680015563965
- Beach Volleyball 44890.67011499405
- Time taken: 211.563 seconds, Fetched: 10 row(s)
- hive (training\_432589)>

On the right side of the browser window, there is a video call interface. It shows a small video feed of a person with the name '04\_Abhinav Prakash,DBDA' below it. Below the video feed, there is a toolbar with buttons for 'Unmute', 'Stop Video', 'Participants', 'Chat', 'New Share', 'Pause Share', 'Annotate', 'Apps', and 'More'. A green status bar at the bottom of the toolbar indicates 'You are screen sharing' and a red button labeled 'Stop Share'.

The bottom of the browser window shows a Windows taskbar with the date '14-12-2022' and time '15:25'.

### 3) Write a program to create partitioned table on category

First set dynamic partitioning true:---

Set hive.exec.dynamic.partition = true;

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

Create table partitioned\_txns (

txn\_id int,

txn\_date String,

Custno int,

amount float,

product String,

city String,

state String,

spendby String

)

PARTITIONED BY (category String)

Row format delimited

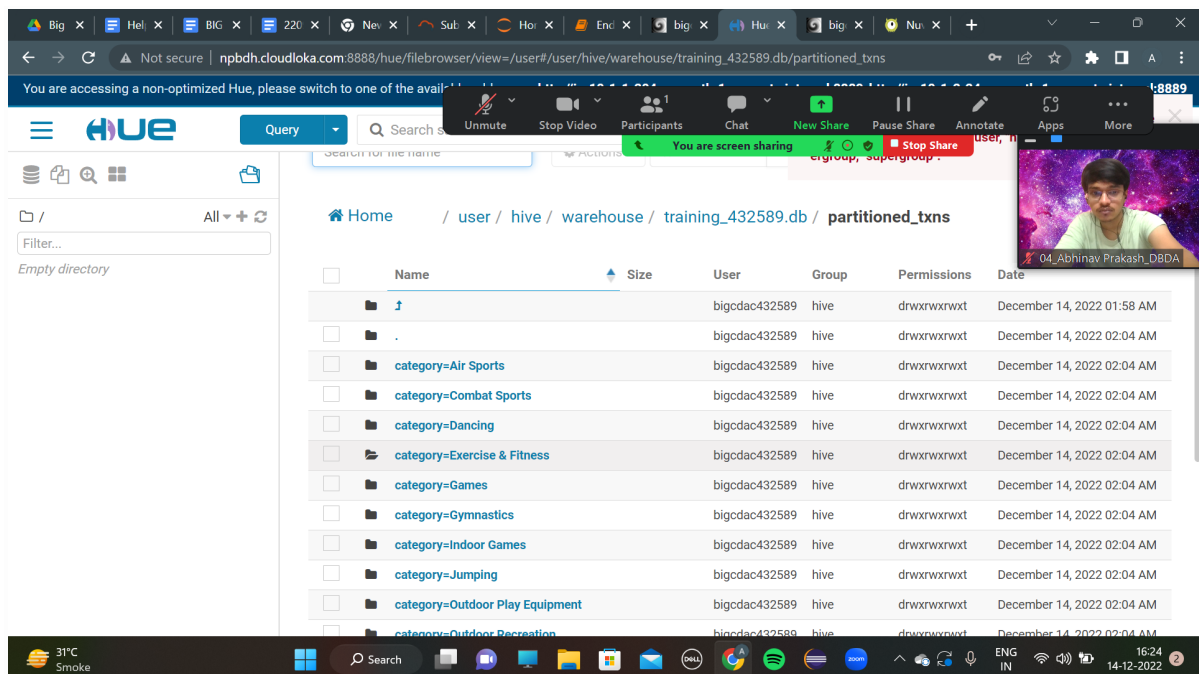
Fields terminated by ','

stored as textFile;

Inserting Data:-----

INSERT OVERWRITE TABLE partitioned\_txns PARTITION(category)

SELECT txn\_id , txn\_date, custno,amount,product,city,state,spendby,category FROM sales  
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

---

## SOLUTION USING DATAFRAMES.

```
schema =  
StructType().add('Year',IntegerType(),True).add('Quarter',IntegerType(),True).add('rev_per_s  
eat',DoubleType(),True).add('num_of_seats',IntegerType(),True)
```

```
df1 =  
ssc.read.format('csv').option('header','True').schema(schema).load('hdfs://nameservice1/user/  
bigcdac432589/EXAM/airlines.csv')
```

```
df1.registerTempTable('airlines')
```

The screenshot shows a Jupyter Notebook window titled "End Mod Exam" with a URL bar indicating it's running on a Databricks instance. The notebook contains the following code cells:

```
In [4]: sc = SparkContext()  
       ssc = SQLContext(sc)
```

```
In [6]: schema = StructType().add('Year',IntegerType(),True).add('Quarter',IntegerType(),True).add('rev_per_seat',DoubleType(),True).add('num_of_seats',IntegerType(),True)
```

```
In [8]: df1 = ssc.read.format('csv').option('header','True').schema(schema).load('hdfs://nameservice1/user/bigcdac432589/EXAM/airlines.csv')
```

```
In [10]: df1.show(5)
```

The output of the last cell shows the top 5 rows of the DataFrame:

Year	Quarter	rev_per_seat	num_of_seats
1995	1	296.9	46561
1995	2	296.8	37443
1995	3	287.51	34128
1995	4	287.78	38388
1996	1	283.97	47888

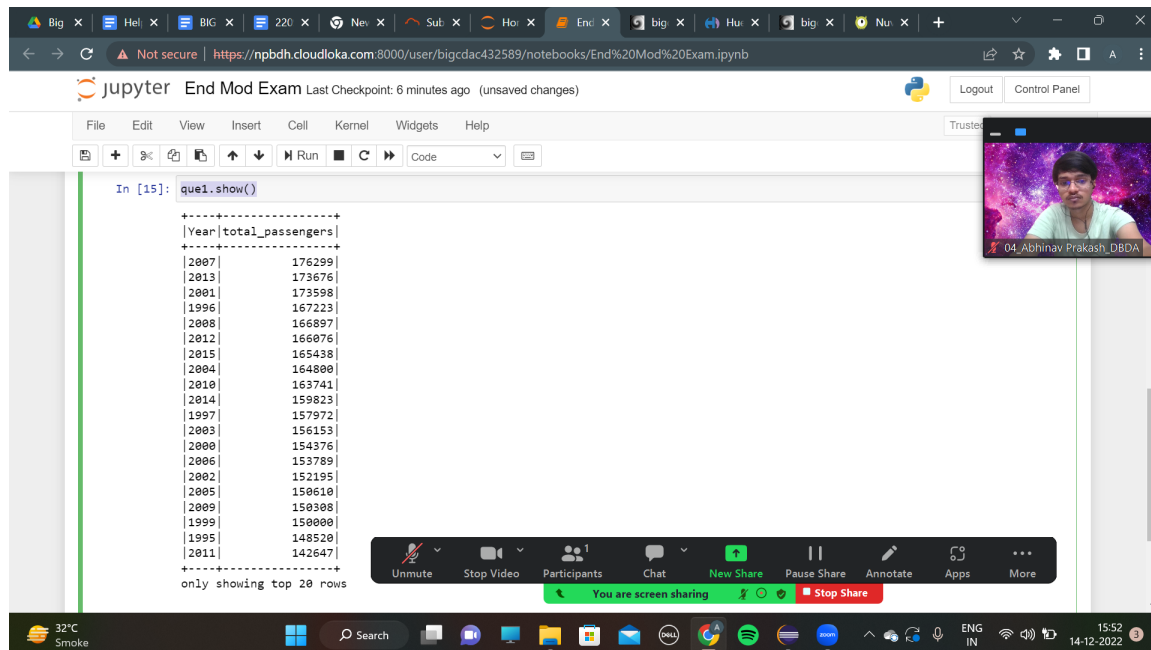
only showing top 5 rows

The interface also includes a video call window on the right showing a participant named "04\_Abhinav Prakash\_D8DA" and a bottom status bar with system icons and a "You are screen sharing" notification.

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

```
que1 = ssc.sql('SELECT Year, sum(num_of_seats) as total_passengers FROM airlines  
GROUP BY Year ORDER BY total_passengers DESC')
```

```
que1.show()
```



The screenshot shows a Jupyter Notebook interface with a browser window at the top displaying the URL <https://npbdh.cloudloka.com:8000/user/bigcdac432589/notebooks/End%20Mod%20Exam.ipynb>. The notebook title is "End Mod Exam" and it shows "Last Checkpoint: 6 minutes ago (unsaved changes)". The interface includes a menu bar (File, Edit, View, Insert, Cell, Kernel, Widgets, Help) and a toolbar with icons for file operations, running cells, and code execution. The code cell contains the following SQL query:

```
In [15]: que1.show()
```

The output of the query is displayed as a table with 20 rows, showing the highest number of passengers for each year. The table is as follows:

Year	total_passengers
2007	176299
2013	173676
2001	173598
1996	167223
2008	166897
2012	166076
2015	165438
2004	164800
2010	163741
2014	159823
1997	157972
2003	156153
2000	154376
2006	153789
2002	152195
2005	150610
2009	150308
1999	150000
1995	148520
2011	142647

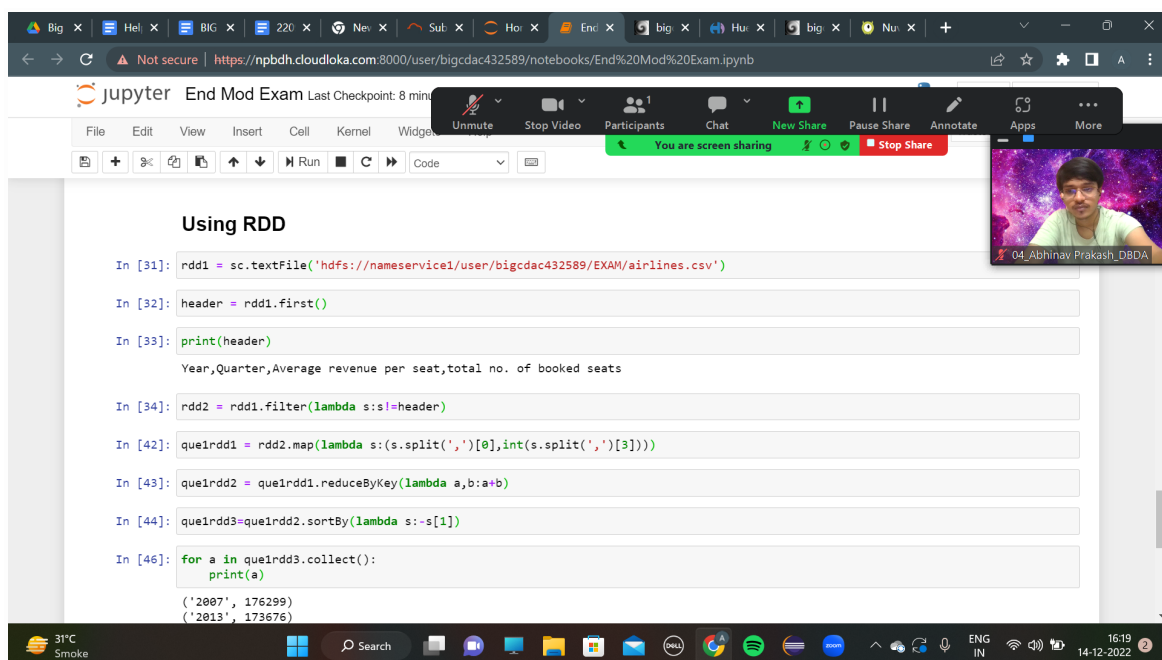
At the bottom of the output, it says "only showing top 20 rows". A video call window is visible in the top right corner, showing a person with the name "D4\_Abhinav Prakash\_DBDA". The bottom of the screen shows a Windows taskbar with various icons and the system clock indicating 15:52 on 14-12-2022.

```
In [16]: que1.first()
```

```
Out[16]: Row(Year=2007, total_passengers=176299)
```

**Highest Number of people traveled in 2007 with a total of 1,76,299 passengers.**

## Solution Using RDD



```
In [31]: rdd1 = sc.textFile('hdfs://nameservice1/user/bigcdac432589/EXAM/airlines.csv')
In [32]: header = rdd1.first()
In [33]: print(header)
Year,Quarter,Average revenue per seat,total no. of booked seats

In [34]: rdd2 = rdd1.filter(lambda s:s!=header)
In [42]: que1rdd1 = rdd2.map(lambda s:(s.split(',')[0],int(s.split(',')[3])))
In [43]: que1rdd2 = que1rdd1.reduceByKey(lambda a,b:a+b)
In [44]: que1rdd3=que1rdd2.sortBy(lambda s:-s[1])
In [46]: for a in que1rdd3.collect():
        print(a)
('2007', 176299)
('2013', 173676)
```

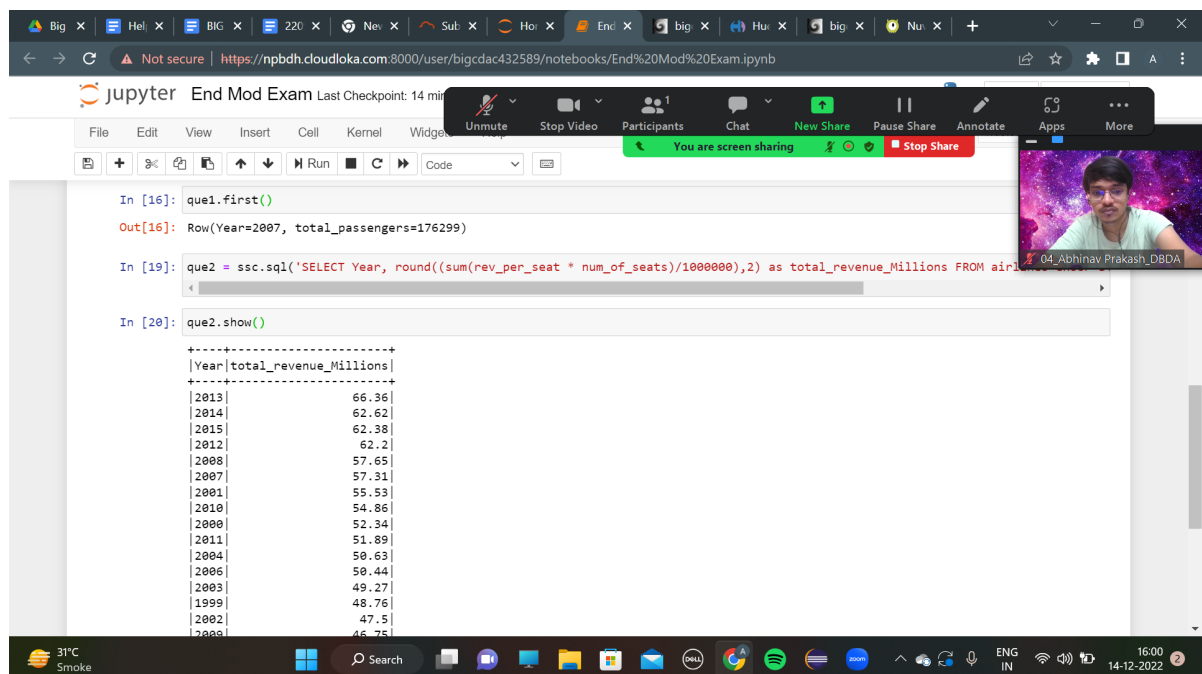
```
In [47]: que1rdd3.first()
```

```
Out[47]: ('2007', 176299)
```

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

```
que2 = ssc.sql('SELECT Year, round((sum(rev_per_seat * num_of_seats)/1000000),2) as  
total_revenue_Millions FROM airlines GROUP BY Year ORDER BY  
total_revenue_Millions DESC')
```

```
que2.show()
```



```
In [16]: que1.first()
Out[16]: Row(Year=2007, total_passengers=176299)

In [19]: que2 = ssc.sql('SELECT Year, round((sum(rev_per_seat * num_of_seats)/1000000),2) as total_revenue_Millions FROM airlines GROUP BY Year ORDER BY total_revenue_Millions DESC')

In [20]: que2.show()
```

Year	total_revenue_Millions
2013	66.36
2014	62.62
2015	62.38
2012	62.2
2008	57.65
2007	57.31
2001	55.53
2010	54.86
2000	52.34
2011	51.89
2004	50.63
2006	50.44
2003	49.27
1999	48.76
2002	47.5
2009	46.75

```
In [21]: que2.first()
```

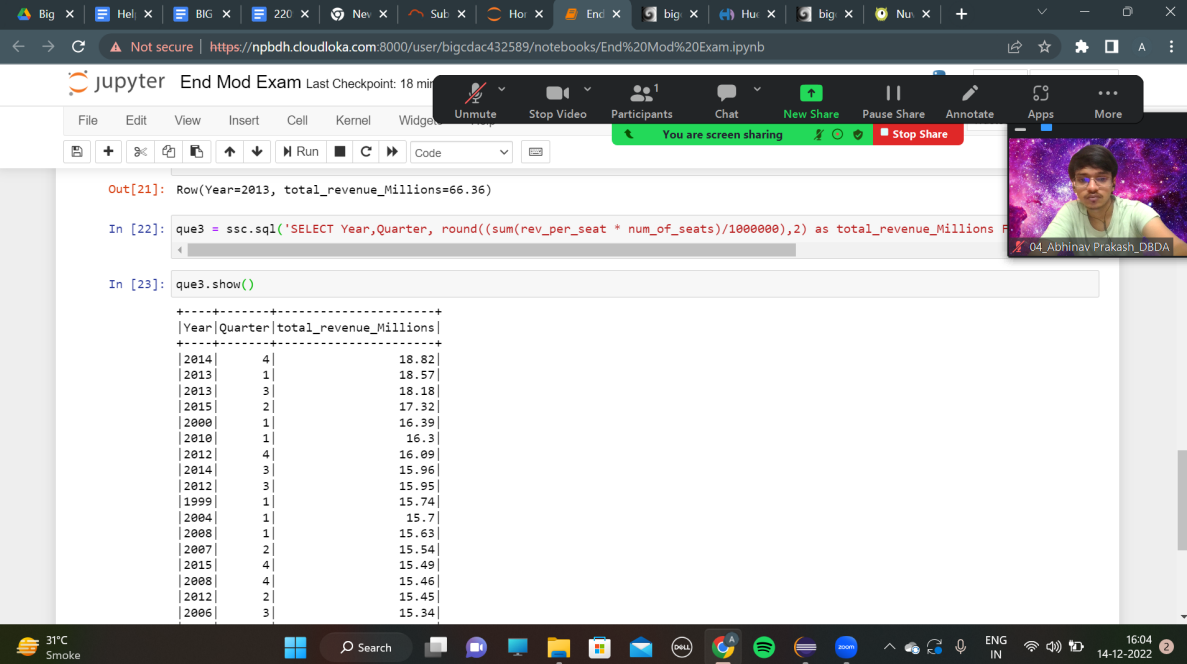
```
Out[21]: Row(Year=2013, total_revenue_Millions=66.36)
```

**Highest Revenue was generated in 2013 with a total of 66.36 Million.**

---

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

```
que3 = ssc.sql('SELECT Year,Quarter, round((sum(rev_per_seat *  
num_of_seats)/1000000),2) as total_revenue_Millions FROM airlines GROUP BY  
Year,Quarter ORDER BY total_revenue_Millions DESC')
```



The screenshot shows a Jupyter Notebook interface with a browser window at the top displaying the URL <https://npbdh.cloudloka.com:8000/user/bigcdac432589/notebooks/End%20Mod%20Exam.ipynb>. The notebook has a menu bar (File, Edit, View, Insert, Cell, Kernel, Widgets) and a toolbar with icons for running, saving, and other actions. A video call window is visible on the right side of the notebook interface.

```
Out[21]: Row(Year=2013, total_revenue_Millions=66.36)
```

```
In [22]: que3 = ssc.sql('SELECT Year,Quarter, round((sum(rev_per_seat * num_of_seats)/1000000),2) as total_revenue_Millions F
```

```
In [23]: que3.show()
```

Year	Quarter	total_revenue_Millions
2014	4	18.82
2013	1	18.57
2013	3	18.18
2015	2	17.32
2000	1	16.39
2010	1	16.3
2012	4	16.09
2014	3	15.96
2012	3	15.95
1999	1	15.74
2004	1	15.7
2008	1	15.63
2007	2	15.54
2015	4	15.49
2008	4	15.46
2012	2	15.45
2006	3	15.34

```
In [24]: que3.first()
```

```
Out[24]: Row(Year=2014, Quarter=4, total_revenue_Millions=18.82)
```

---

Highest Revenue was generated in 2014's fourth Quarter with a total sum of 18.82 Millions.