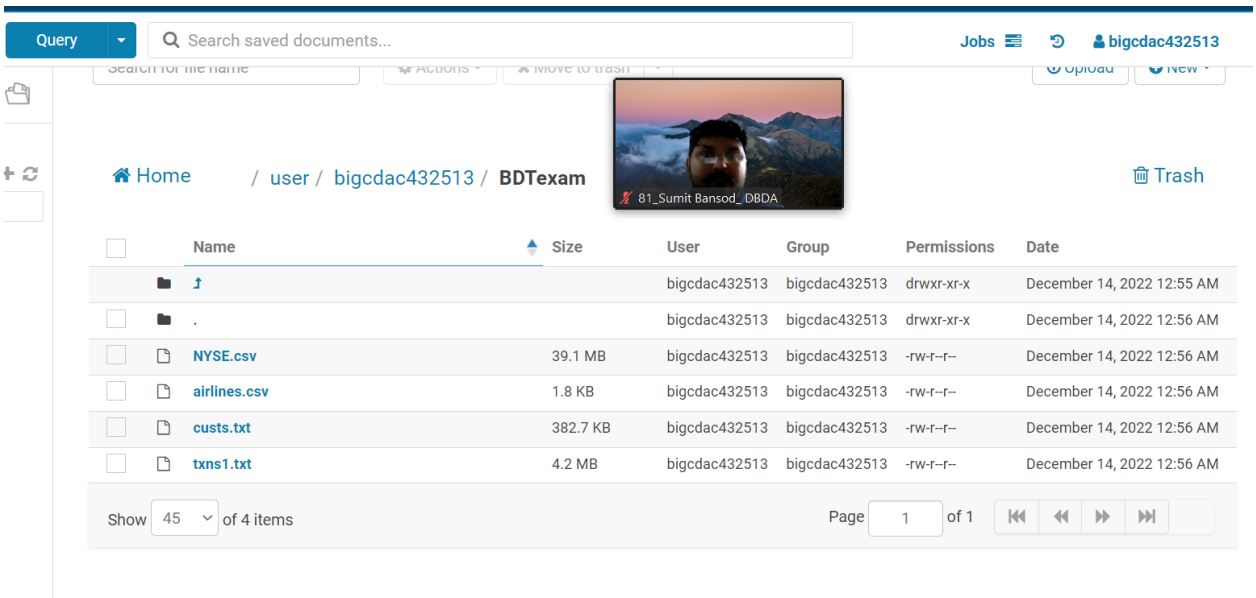
Roll no : 220940325081
Name : Sumit Bansod
Exam: Big Data Technologies

UPLOADING DATA into Hue directly from hue



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

MAPREDUCE CODE:

```
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(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 tempval=0;

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

                  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(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);
               }
}
```

—----------

<u>Question 2 : F</u>ind all time High price for each stock **[15 marks]**

```
[bigcdac432513@ip-10-1-1-204 ~]$ jar tvf myjar.jar
   25 Wed Dec 14 16:35:34 UTC 2022 META-INF/MANIFEST.MF
 2619 Wed Dec 14 16:35:18 UTC 2022 AllTimeHigh$MapClass.class
 2422 Wed Dec 14 16:35:18 UTC 2022 AllTimeHigh$ReduceClass.class
 1715 Wed Dec 14 16:35:18 UTC 2022 AllTimeHigh.class
```

```
hadoop jar myjar.jar AllTimeHigh
/user/bigcdac432513/BDTexam/NYSE.csv
/user/bigcdac432513/BDTexam/AllTimeHighOutput
```

****Cloudera froze after this****

```
        at org.apache.hadoop.util.RunJar.main(RunJar.java:227)
[bigcdac432513@ip-10-1-1-204 ~]$ hadoop jar myjar.jar AllTimeHigh /user/bigcdac432513/BDTexam/NYSE.csv /user/bigcdac432513/BDTexam/AllTimeHighOutput
WARNING: Use "yarn jar" to launch YARN applications.
22/12/14 11:10:13 INFO client.RMProxy: Connecting to ResourceManager at ip-10-1-1-204.ap-south-1.compute.internal/10.1.1.204:8032
22/12/14 11:10:13 WARN mapreduce.JobResourceUploader: Hadoop command-line option parsing not performed. Implement the Tool interface and execute your
application with ToolRunner to remedy this.
22/12/14 11:10:13 INFO mapreduce.JobResourceUploader: Disabling Erasure Coding for path: /user/bigcdac432513/.staging/job_1663041244711_23128
22/12/14 11:10:14 INFO input.FileInputFormat: Total input files to process : 1
22/12/14 11:10:14 INFO mapreduce.JobSubmitter: number of splits:1
22/12/14 11:10:14 INFO Configuration.deprecation: yarn.resourcemanager.system-metrics-publisher.enabled is deprecated. Instead, use yarn.system-metric
s-publisher.enabled
22/12/14 11:10:14 INFO mapreduce.JobSubmitter: Submitting tokens for job: job_1663041244711_23128
22/12/14 11:10:14 INFO mapreduce.JobSubmitter: Executing with tokens: []
22/12/14 11:10:15 INFO conf.Configuration: resource-types.xml not found
22/12/14 11:10:15 INFO resource.ResourceUtils: Unable to find 'resource-types.xml'.
22/12/14 11:10:15 INFO impl.YarnClientImpl: Submitted application application_1663041244711_23128
22/12/14 11:10:15 INFO mapreduce.Job: The url to track the job: http://ip-10-1-1-204.ap-south-1.compute.internal:6066/proxy/application_1663041244711_
23128/
22/12/14 11:10:15 INFO mapreduce.Job: Running job: job_1663041244711_23128
```

Hive

Please find the customer data set.

cust id

firstname

lastname

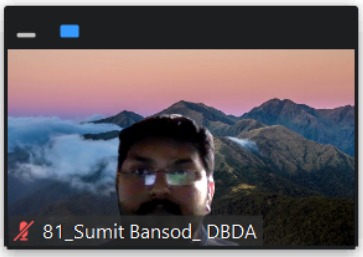
age

profession

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

```
[bigcdac432513@ip-10-1-1-204 ~]$ hadoop fs -put custs.txt
BDTexam
```

```
[bigcdac432513@ip-10-1-1-204 ~]$ hadoop fs -put custs.txt BDTexam
put: `BDTexam/custs.txt': File exists
[bigcdac432513@ip-10-1-1-204 ~]$
```



Launching hive

```
[bigcdac432513@ip-10-1-1-204 ~]$ hive
hive> create database bdt_hiveexam;
OK
Time taken: 1.79 seconds
hive> use bdt_hiveexam;
OK
Time taken: 0.188 seconds
hive> show tables;
OK
Time taken: 0.301 seconds
hive> set hive.cli.print.current.db=true;
hive (bdt_hiveexam)>
```

Create table:

```
hive (bdt_hiveexam)> create table customer(
                   >
                   > cust_id string,
                   >
                   > first_name string,
                   >
                   > last_name string,
                   >
                   > age string,
                   >
```

```
> profession string)
>
> row format delimited
>
> fields terminated by ","
>
> stored as textfile;
```

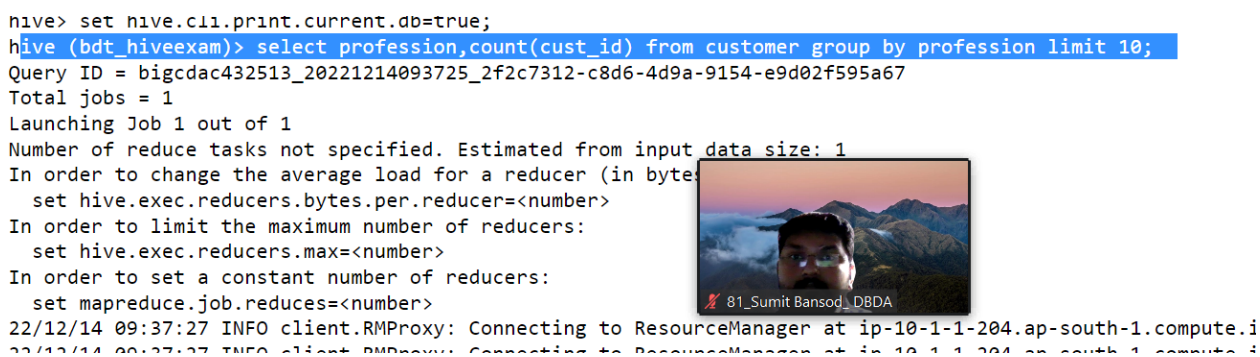
OK

Loading data

```
hive (bdt_hiveexam)> load data local inpath 'custs.txt'
overwrite into table customer;
Loading data to table bdt_hiveexam.customer
OK
```

Query:

```
hive (bdt_hiveexam)> select profession,count(cust_id) from
customer group by profession limit 10;
```

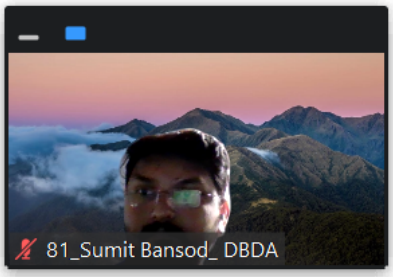


Output:

```
OK
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
Time taken: 111.908 seconds, Fetched: 10 row(s)
```

```
Total MapReduce CPU Time Spent: 7 seconds 750 msec
OK
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
Time taken: 111.908 seconds, Fetched: 10 row(s)
hive (bdt hiveexam)>
```



Please find the sales data set.

txn id

txn date

cust id

amount

category

product

city

state

spendby

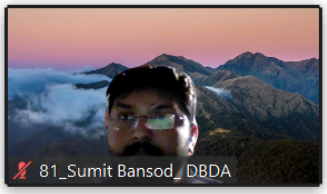
Creating table:

```
hive (bdt_hiveexam)> create table txn(
                   >
                   > txn_id string,
                   >
                   > txn_date string,
                   >
                   > cust_id string,
                   >
                   > amount string,
                   >
                   > category string,
                   >
                   > product string,
                   >
                   > city string,
                   >
                   > state string,
                   >
                   > spendby string)
                   >
                   > row format delimited
                   >
                   > fields terminated by ","
                   >
                   > stored as textfile;
OK
```

Loading data:

```
hive (bdt_hiveexam)> load data local inpath 'txns1.txt'
overwrite into table txn;
Loading data to table bdt_hiveexam.txn
OK
```

```
hive (bdt_hiveexam)> create table txn(
                   >
                   > txn_id string,
                   >
                   > txn_date string,
                   >
                   > cust_id string,
                   >
                   > amount string,
                   >
                   > category string,
                   >
                   > product string,
                   >
                   > city string,
                   >
                   > state string,
                   >
                   > spendby string)
                   >
                   > row format delimited
                   >
                   > fields terminated by ","
                   >
                   > stored as textfile;
OK
Time taken: 0.486 seconds
hive (bdt_hiveexam)> load data local inpath 'txns1.txt' overwrite into table txn;
Loading data to table bdt_hiveexam.txn
OK
Time taken: 1.193 seconds
```



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

Query:

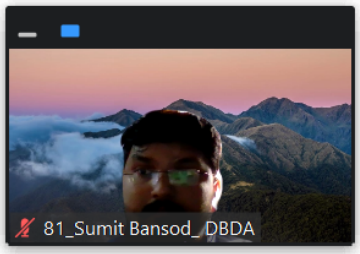
```
hive (bdt_hiveexam)> select product,sum(amount) as Total_amt
from txn group by product order by Total_amt desc limit 10;
```

Output:

```
OK
Yoga & Pilates   47804.93999999993
Swing Sets       47204.13999999999
Lawn Games       46828.44
```

```
Golf    46577.67999999999
Cardio Machine Accessories      46485.540000000045
Exercise Balls  45143.84
Weightlifting Belts     45111.67999999996
Mahjong 44995.19999999999
Basketball      44954.68000000004
Beach Volleyball        44890.67000000005
```

```
Ended Job = job_1663041244711_22836
MapReduce Jobs Launched:
Stage-Stage-1: Map: 1  Reduce: 1   Cumulative CPU: 6.79 sec   HDFS Read: 4426685 HD
Stage-Stage-2: Map: 1  Reduce: 1   Cumulative CPU: 6.16 sec   HDFS Read: 10538 HDFS
Total MapReduce CPU Time Spent: 12 seconds 950 msec
OK
Yoga & Pilates  47804.93999999993
Swing Sets      47204.13999999999
Lawn Games      46828.44
Golf    46577.67999999999
Cardio Machine Accessories      46485.540000000045
Exercise Balls  45143.84
Weightlifting Belts     45111.67999999996
Mahjong 44995.19999999999
Basketball      44954.68000000004
Beach Volleyball        44890.67000000005
Time taken: 192.722 seconds, Fetched: 10 row(s)
```



3) Write a program to create partitioned table on category
Enabling partition:

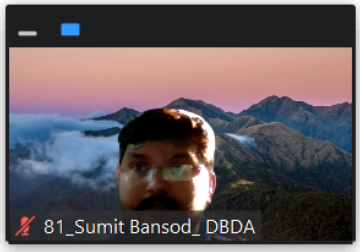
```
hive (bdt_hiveexam)> set
hive.exec.dynamic.partiton.mode=nonstrict;

hive (bdt_hiveexam)> set hive.exec.dunamic.partition=true;
```

Creating partition table:

```
hive (bdt_hiveexam)> create table txn_partition(
                   >
                   > txn_id string,
                   >
                   > txn_date string,
                   >
                   > cust_id string,
                   >
```

```
                    > amount string,
                    >
                    > product string,
                    >
                    > city string,
                    >
                    > state string,
                    >
                    > spendby string) partitioned by (category
string)
                    >
                    > row format delimited
                    >
                    > fields terminated by ","
                    >
                    > stored as textfile;
OK
Time taken: 0.215 seconds
```

```
hive (bdt_hiveexam)> create table txn_partition(
                   >
                   > txn_id string,
                   >
                   > txn_date string,
                   >
                   > cust_id string,
                   >
                   > amount string,
                   >
                   > product string,
                   >
                   > city string,
                   >
                   > state string,
                   >
                   > spendby string) partitioned by (category string)
                   >
                   > row format delimited
                   >
                   > fields terminated by ","
                   >
                   > stored as textfile;
OK
Time taken: 0.215 seconds
```



```
hive (bdt_hiveexam)> show tables;
OK
customer
txn
txn_partition
```

QUESTION 3 [15 marks]

PySpark

Please find the AIRLINES data set

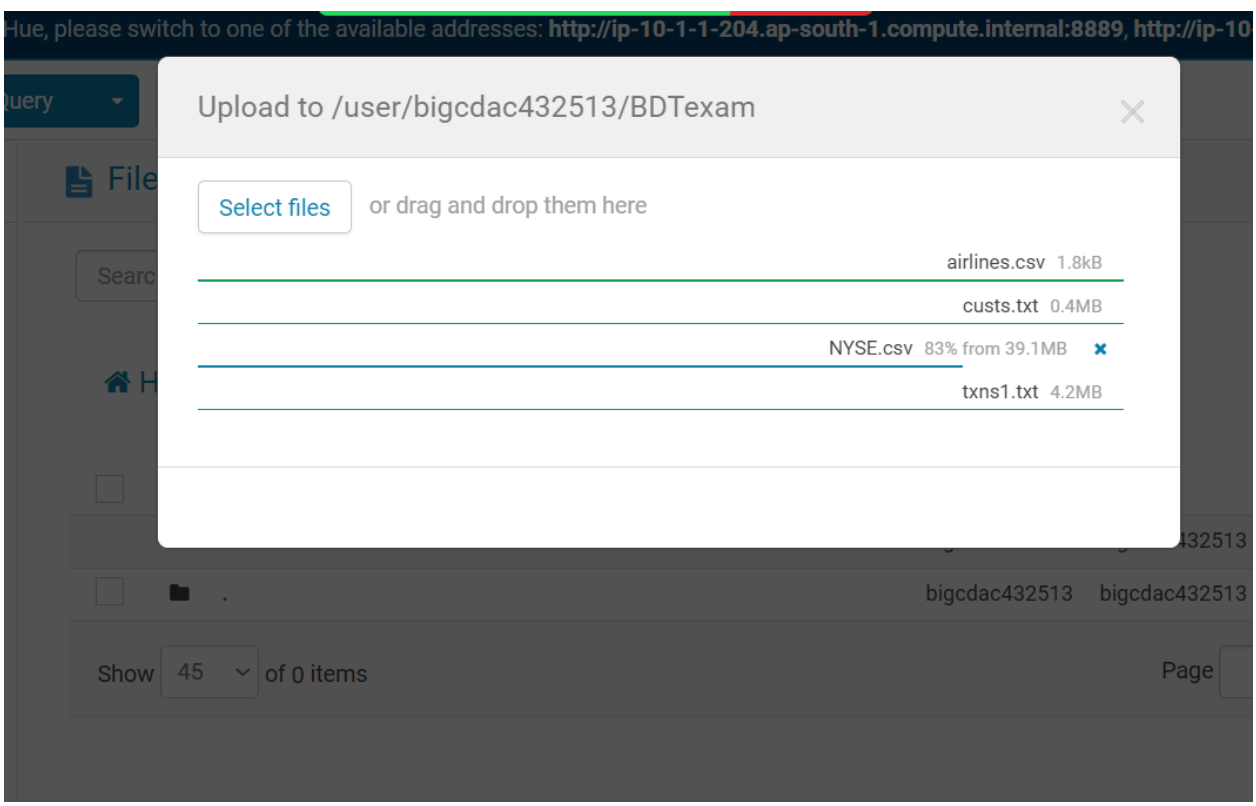
Year

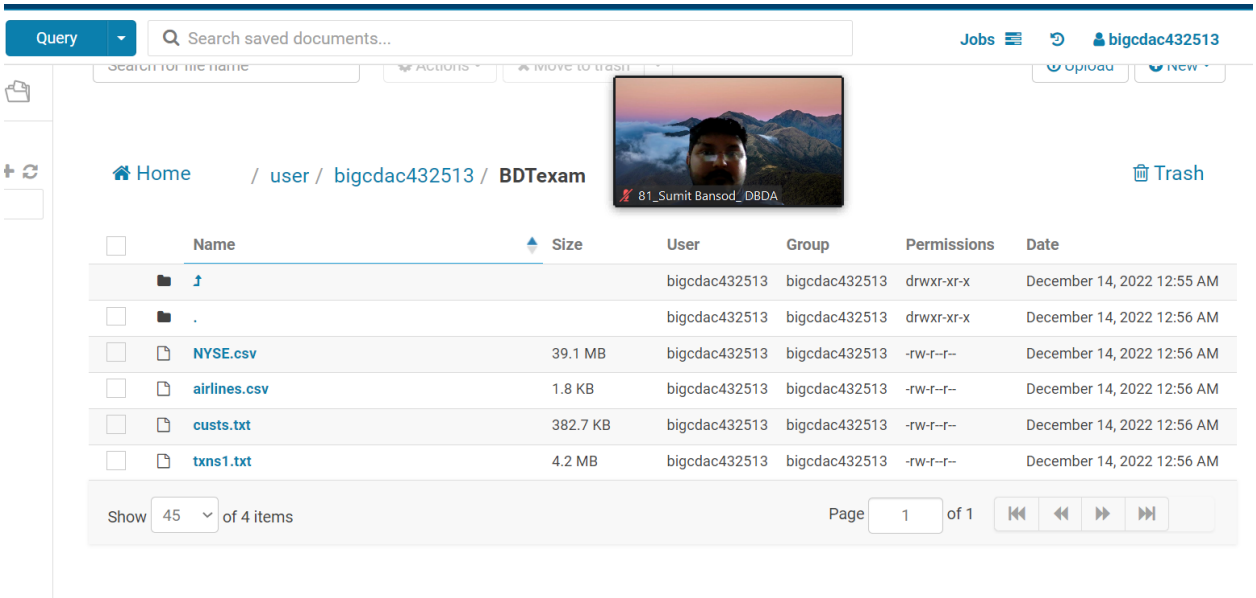
Quarter

Average revenue per seat

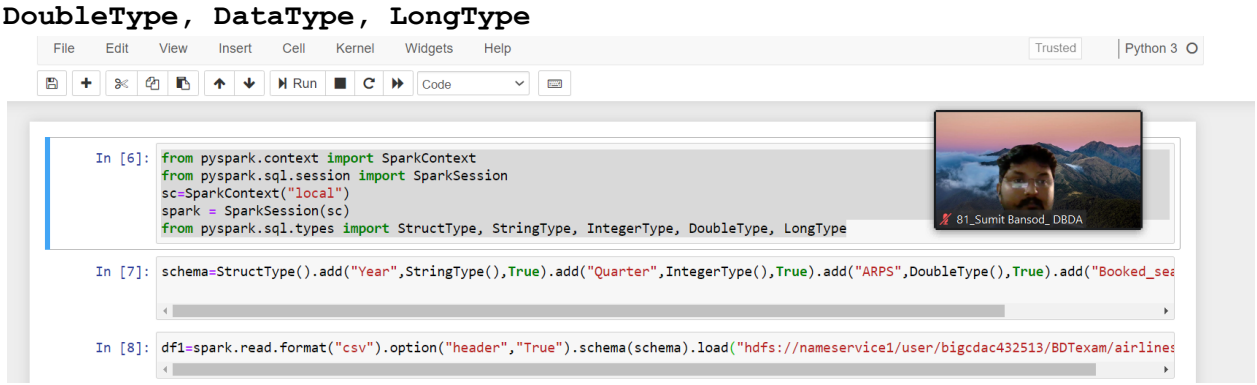
Total number of booked seats

Uploading data into Hue






```
from pyspark.sql import SparkSession
from pyspark.sql.types import *

spark = SparkSession.builder.config('spark.some.config.option',
'some-value').getOrCreate()

from pyspark.sql.types import StringType, StringType, IntegerType,
DoubleType, DataType, LongType
```



schema=StructType().add("Year",StringType(),True).add("Quarter",IntegerType(),True).add("ARPS",DoubleType(),True).add("Booked_seats",IntegerType(),True)

Spark got frozen after this

df1=spark.read.format("csv").option("header","True").schema(schema).load("hdfs://nameservice1/user/bigcdac432513/BDTexam/airlines.csv",1)

df1.registerTempTable("*airlines*")

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

Year?

```
dfque1 = spark.sql('select year, sum(booked_Seat) as totalSeat from airlines group by year order by totalSeat desc limit 10');
```

2) Identifying the highest revenue generation for which year

```
dfque2 = spark.sql('select year, sum(arps*Booked_Seat) as avgRev from airlines group by year order by avgRev desc limit 1');
```

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

```
dfque3 = spark.sql('select year, quarter , sum(arps*Booked_Seat) as highRev from airlines group by year, quarter limit 1');
```