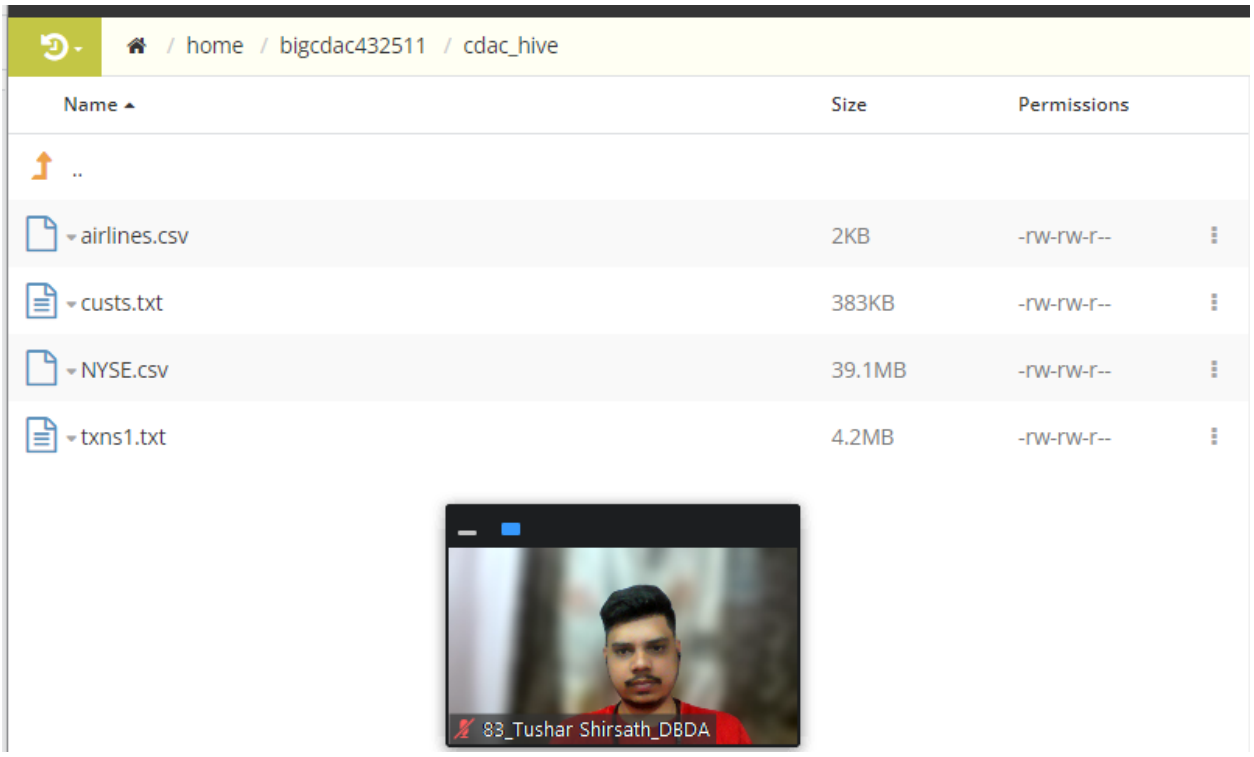
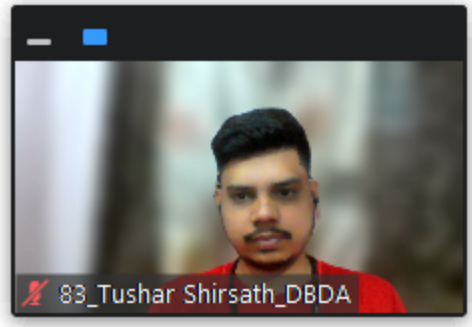
Roll Number : 220940325083
Name: Tushar Shirsath
Exam : Big Data Technologies

HIVE



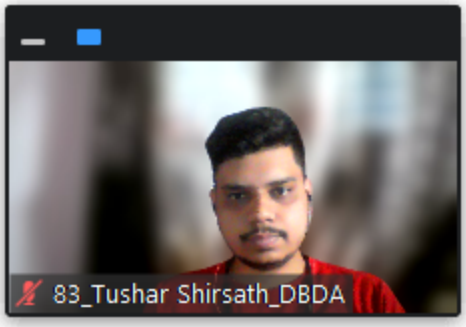
```
[bigcdac432511@ip-10-1-1-204 ~]$ mkdir cdac_hive
[bigcdac432511@ip-10-1-1-204 ~]$ cd cdac_hive
[bigcdac432511@ip-10-1-1-204 cdac_hive]$ ls
[bigcdac432511@ip-10-1-1-204 cdac_hive]$ ls
airlines.csv  custs.txt  NYSE.csv  txns1.txt
[bigcdac432511@ip-10-1-1-204 cdac_hive]$
```




```
[bigcdac432511@ip-10-1-1-204 ~]$ mkdir cdac_hive
[bigcdac432511@ip-10-1-1-204 ~]$ cd cdac_hive
[bigcdac432511@ip-10-1-1-204 cdac_hive]$ ls
[bigcdac432511@ip-10-1-1-204 cdac_hive]$ ls
airlines.csv  custs.txt  NYSE.csv  txns1.txt
```

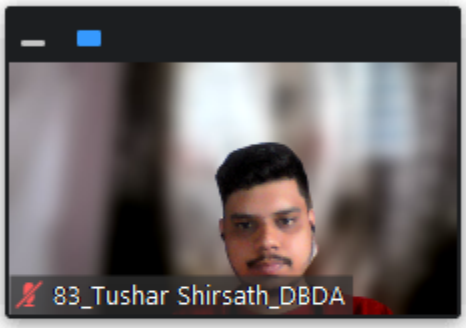

```
hive> create database hive_exam;
OK
Time taken: 0.212 seconds
hive> use hive_exam;
OK
Time taken: 0.039 seconds
hive> show tables;
OK
```

```
hive> create database hive_exam;
OK
Time taken: 0.212 seconds
hive> use hive_exam;
OK
Time taken: 0.039 seconds
hive> show tables;
OK
Time taken: 0.239 seconds
hive>
```



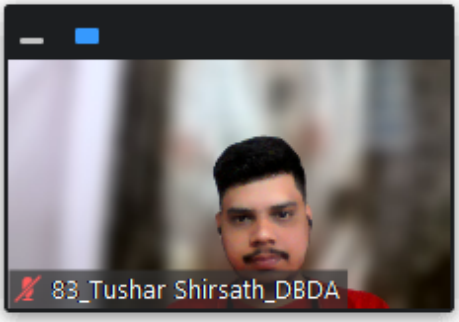
[bigcdac432511@ip-10-1-1-204 ~]$ hadoop fs -mkdir exam1

```
hive> 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
Time taken: 0.226 seconds
hive>
```



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

```
hive> show tables;
OK
customer
Time taken: 0.144 seconds, Fetched: 1 row(s)
hive> select * from customer limit 10;
OK
4000001 Kristina        Chung   55      Pilot
4000002 Paige   Chen    74      Teacher
4000003 Sherri  Melton  34      Firefighter
4000004 Gretchen        Hill    66      Computer hardware engineer
4000005 Karen   Puckett 74      Lawyer
4000006 Patrick Song    42      Veterinarian
4000007 Elsie   Hamilton        43      Pilot
4000008 Hazel   Bender  63      Carpenter
4000009 Malcolm Wagner  39      Artist
4000010 Dolores McLaughlin      60      Writer
Time taken: 0.519 seconds, Fetched: 10 row(s)
hive>
```




```
hive> select * from customer limit 10;
OK
4000001 Kristina        Chung   55      Pilot
4000002 Paige   Chen    74      Teacher
4000003 Sherri  Melton  34      Firefighter
4000004 Gretchen        Hill    66      Computer hardware engineer
4000005 Karen   Puckett 74      Lawyer
4000006 Patrick Song    42      Veterinarian
4000007 Elsie   Hamilton        43      Pilot
4000008 Hazel   Bender  63      Carpenter
4000009 Malcolm Wagner  39      Artist
4000010 Dolores McLaughlin      60      Writer
Time taken: 0.519 seconds, Fetched: 10 row(s)
```

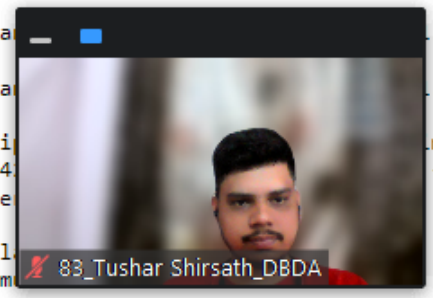
Q.1) Write a program to find the count of customers for each profession.

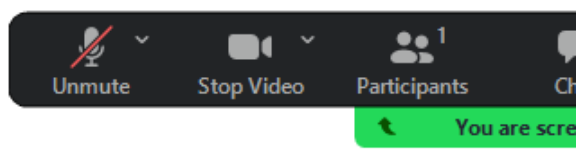
```
hive> select profession, count(cust_id) as count from customer group by profession;
```

output->

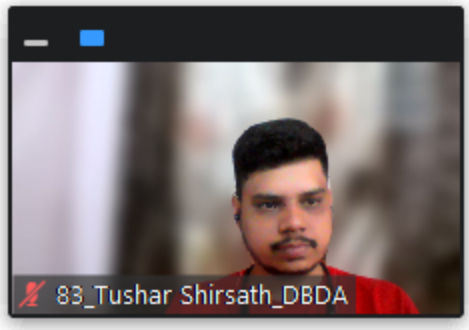
OK
Accountant 199
Actor 202
Agricultural and food scientist 195
Architect 203
Artist 175
Athlete 196
Automotive mechanic 193
Carpenter 181
etc.

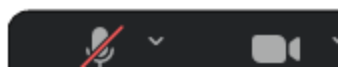
```
hive> select profession, count(cust_id) as count from customer group by profession;
Query ID = bigcdac432511_20221214091111_a17abbe7-1189-400e-b7bd-95bb278dbc2e
Total jobs = 1
Launching Job 1 out of 1
Number of reduce tasks not specified. Estimated from input data size: 1
In order to change the average load for a reducer (in bytes):
  set hive.exec.reducers.bytes.per.reducer=<number>
In order to limit the maximum number of reducers:
  set hive.exec.reducers.max=<number>
In order to set a constant number of reducers:
  set mapreduce.job.reduces=<number>
22/12/14 09:11:11 INFO client.RMProxy: Connecting to ResourceMa
nal/10.1.1.204:8032
22/12/14 09:11:12 INFO client.RMProxy: Connecting to ResourceMa
nal/10.1.1.204:8032
Starting Job = job_1663041244711_22613, Tracking URL = http://i
Kill Command = /opt/cloudera/parcels/CDH-6.2.1-1.cdh6.2.1.p0.14
Hadoop job information for Stage-1: number of mappers: 1; numbe
2022-12-14 09:11:42,033 Stage-1 map = 0%,  reduce = 0%
2022-12-14 09:11:52,633 Stage-1 map = 100%,  reduce = 0%, Cumul
2022-12-14 09:12:13,642 Stage-1 map = 100%,  reduce = 100%, Cum
MapReduce Total cumulative CPU time: 8 seconds 500 msec
Ended Job = job_1663041244711_22613
MapReduce Jobs Launched:
Stage-Stage-1: Map: 1  Reduce: 1   Cumulative CPU: 8.5 sec   HDFS Read: 400722 HDFS Write: 1584 HDF
Total MapReduce CPU Time Spent: 8 seconds 500 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
Civil engineer  193
Coach   201
Computer hardware engineer      204
Computer software engineer      216
```



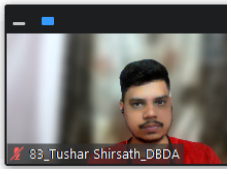


```
Dancer  185
Designer        205
Doctor  197
Economist       189
Electrical engineer     192
Electrician     194
Engineering technician  204
Environmental scientist 176
Farmer  201
Financial analyst       198
Firefighter     217
Human resources assistant       212
Judge   196
Lawyer  212
Librarian       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
```





```
hive> load data local inpath 'txns1.txt' overwrite into table txn1;
Loading data to table hive_exam.txn1
OK
Time taken: 1.389 seconds
hive> select * from txn1 limit 10;
OK
00000000        06-26-2011      4007024 40.33   Exercise & Fitness      Cardio Machine Accessories      Clarksville     Tennessee       credit
00000001        05-26-2011      4006742 198.44  Exercise & Fitness      Weightlifting Gloves    Long Beach      California      credit
00000002        06-01-2011      4009775 5.58    Exercise & Fitness      Weightlifting Machine Accessories       Anaheim California      credit
00000003        06-05-2011      4002199 198.19  Gymnastics      Gymnastics Rings        Milwaukee       Wisconsin       credit
00000004        12-17-2011      4002613 98.81   Team Sports     Field Hockey    Nashville       Tennessee       credit
00000005        02-14-2011      4007591 193.63  Outdoor Recreation      Camping & Backpacking & Hiking  Chicago Illinois        credit
00000006        10-28-2011      4002190 27.89   Puzzles Jigsaw Puzzles  Charleston      South Carolina  credit
00000007        07-14-2011      4002964 96.01   Outdoor Play Equipment  Sandboxes       Columbus        Ohio    credit
00000008        01-17-2011      4007361 10.44   Winter Sports   Snowmobiling    Des Moines      Iowa    credit
00000009        05-17-2011      4004798 152.46  Jumping Bungee Jumping  St. Petersburg  Florida credit
Time taken: 0.106 seconds, Fetched: 10 row(s)
hive>
```




```
create table txn(
```

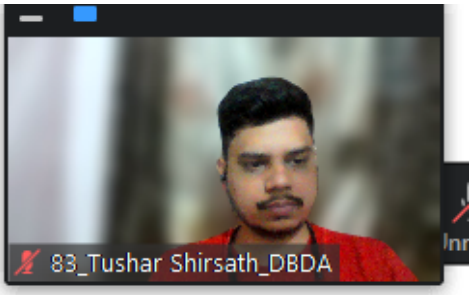
```
txn_id string,
txn_date string,
cust_id string,
amount double,
category string,
product string,
city string,
state string,
spendby string)
row format delimited
fields terminated by ','
stored as textfile;
```

```
hive> load data local inpath 'txns1.txt' overwrite into table txn;
```

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

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

```
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: 158.118 seconds, Fetched: 10 row(s)
```

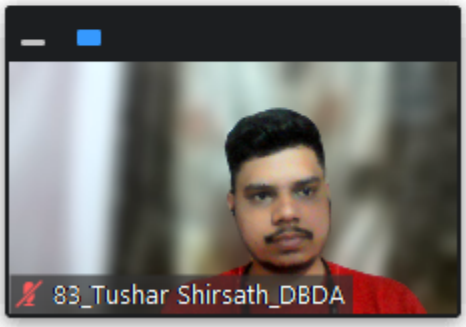


Q.3) Write a program to create partitioned table on category

```
create table txn1_partition(
txn_id string,
txn_date string,
cust_id string,
amount double,
product string,
city string,
state string,
spendby string)
```

```
partitioned by (category string)
row format delimited
fields terminated by ','
stored as textfile;
```

```
hive> create table txn1_partition(
    >
    > txn_id string,
    >
    > txn_date string,
    >
    > cust_id string,
    >
    > amount double,
    >
    > product string,
    >
    > city string,
    >
    > state string,
    >
    > spendby string)
    >
    > partitioned by (category string)
    >
    > row format delimited
    >
    > fields terminated by ','
    >
    > stored as textfile;
OK
Time taken: 0.136 seconds
hive> show tables;
OK
customer
txn
txn1
txn1_partition
Time taken: 0.043 seconds, Fetched: 4 row(s)
hive>
```

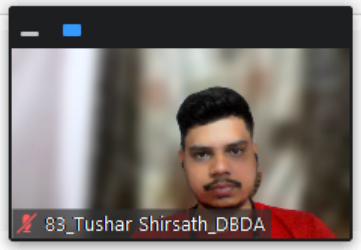


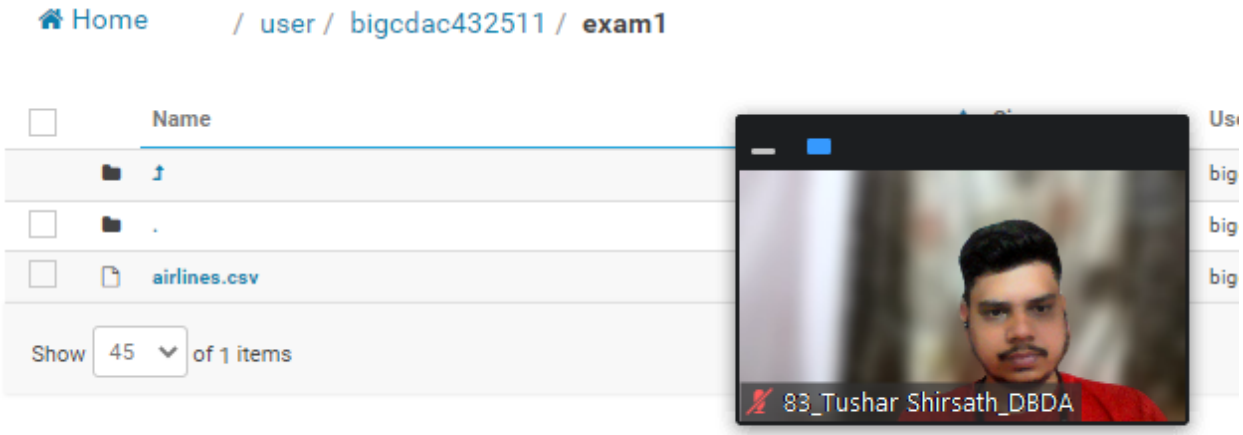
```
### SPARK ###

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

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

Spark

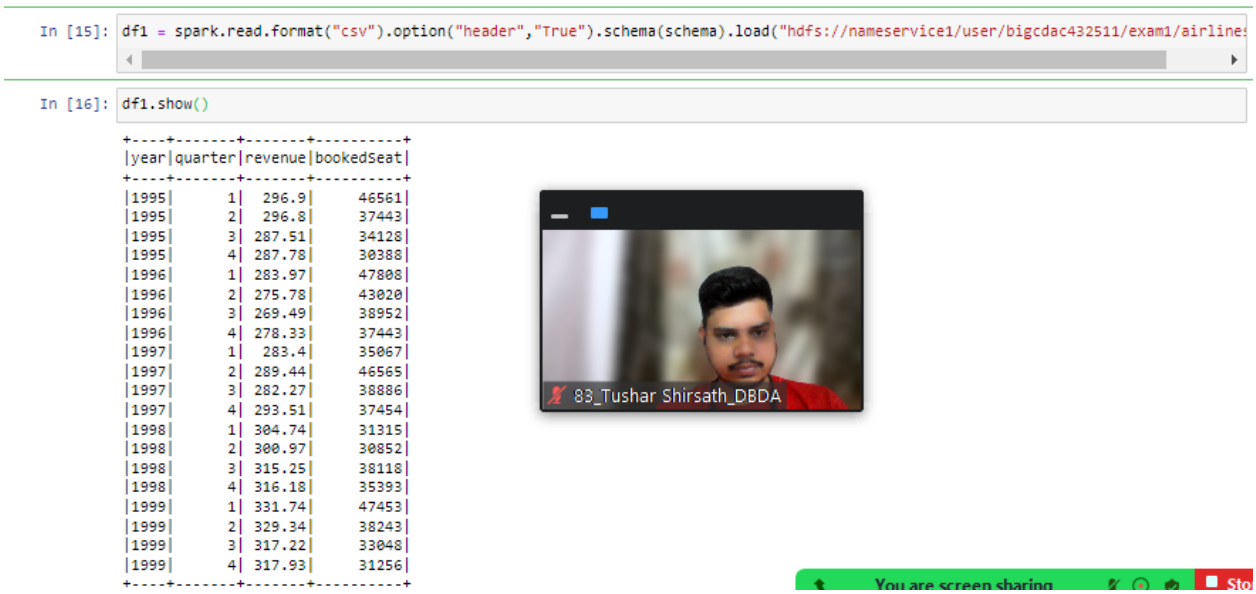
```
from pyspark.sql.types import StringType, StringType, IntegerType,
DoubleType, DataType, LongType

schema = StructType([
    StructField('year', StringType(), True),
    StructField('quarter',StringType(), True),
    StructField('revenue',DoubleType(), True),
    StructField('bookedSeat',IntegerType(), True)
])
```

```
In [6]: from pyspark.sql import SparkSession
        from pyspark.sql.types import *

In [7]: spark = SparkSession.builder.config('spark.some.config.option', 'some-value').getOrCreate()

In [8]: spark

Out[8]: SparkSession - in-memory
        SparkContext

        Spark UI
        Version
        v2.4.0
        Master
        local[*]
        AppName
        pyspark-shell

In [12]: from pyspark.sql.types import StringType, StringType, IntegerType, DoubleType, DataType, LongType

In [14]: schema = StructType([
             StructField('year', StringType(), True),
             StructField('quarter',StringType(), True),
             StructField('revenue',DoubleType(), True),
             StructField('bookedSeat',IntegerType(), True)
         ])
```






```
df1 =
spark.read.format("csv").option("header","True").schema(schema).load("
hdfs://nameservice1/user/bigcdac432511/exam1/airlines.csv")
```

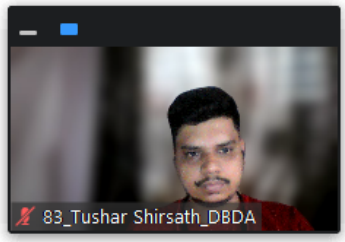



```
+----+-------+-------+----------+
|year|quarter|revenue|bookedSeat|
+----+-------+-------+----------+
|1995|      1|  296.9|     46561|
|1995|      2|  296.8|     37443|
|1995|      3| 287.51|     34128|
|1995|      4| 287.78|     30388|
|1996|      1| 283.97|     47808|
|1996|      2| 275.78|     43020|
|1996|      3| 269.49|     38952|
|1996|      4| 278.33|     37443|
|1997|      1|  283.4|     35067|
|1997|      2| 289.44|     46565|
|1997|      3| 282.27|     38886|
|1997|      4| 293.51|     37454|
|1998|      1| 304.74|     31315|
|1998|      2| 300.97|     30852|
|1998|      3| 315.25|     38118|
|1998|      4| 316.18|     35393|
|1999|      1| 331.74|     47453|
|1999|      2| 329.34|     38243|
|1999|      3| 317.22|     33048|
|1999|      4| 317.93|     31256|
+----+-------+-------+----------+
```

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

```
q1 = spark.sql('select year, sum(bookedSeat) as totalSeat from
airlines group by year order by totalSeat desc limit 10');
```

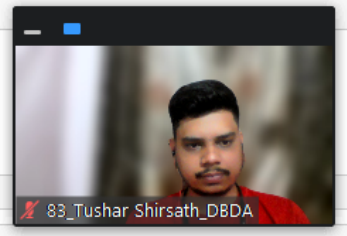
```
In [20]: q1 = spark.sql('select year, sum(bookedSeat) as totalSeat from airlines group by year order by totalSeat desc limit 10');

In [21]: q1.show()

+----+---------+
|year|totalSeat|
+----+---------+
|2007|   176299|
|2013|   173676|
|2001|   173598|
|1996|   167223|
|2008|   166897|
|2012|   166076|
|2015|   165438|
|2004|   164800|
|2010|   163741|
|2014|   159823|
+----+---------+
```



Q.2) Identifying the highest revenue generation for which year

```
q2 = spark.sql('select year, sum(revenue*bookedSeat) as avgRev from
airlines group by year order by avgRev desc limit 1');
```

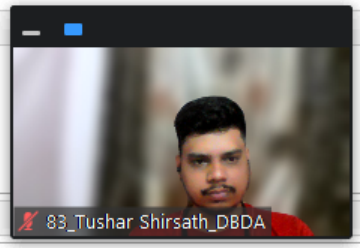
```
In [22]: q2 = spark.sql('select year, sum(revenue*bookedSeat) as avgRev from airlines group by year order by avgRev desc limit 1');

In [23]: q2.show()

+----+-------------+
|year|       avgRev|
+----+-------------+
|2013|6.636320871E7|
+----+-------------+

In [ ]:

In [ ]:
```



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

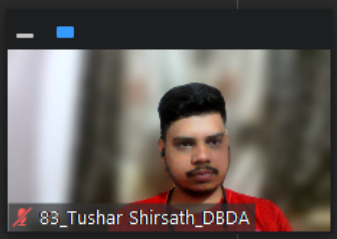
```
q3 = spark.sql('select year, quarter , sum(revenue*bookedSeat) as
highRev from airlines group by year, quarter limit 1');
```

```
In [34]: q3 = spark.sql('select year, quarter , sum(revenue*bookedSeat) as highRev from airlines group by year, quarter limit 1');

In [35]: q3.show()

+----+-------+-------------+
|year|quarter|      highRev|
+----+-------+-------------+
|2014|      2|1.385223868E7|
+----+-------+-------------+

In [ ]:

In [ ]:
```

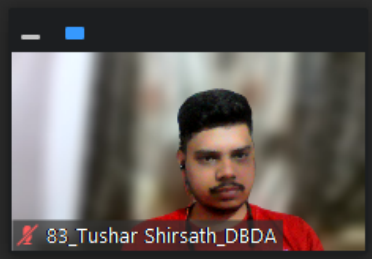


MAPREDUCE

```
import java.io.*;

import org.apache.hadoop.io.DoubleWritable;
import org.apache.hadoop.io.Text;
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 AllTimeHigh {

    public static class MapClass extends Mapper<LongWritable,Text,Text, DoubleWritable>
    {
        public void map(LongWritable key, Text value, Context context)
        {
            try{...}
            catch(Exception e)
            {
                System.out.println(e.getMessage());
            }
        }
    }
```



```java
    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;

            for (DoubleWritable val : values)
            {
                if(val.get() > max){
                    max = val.get();
                }
            }
            int a=10;
            result.set(max);
            context.write(key, result);

        }
    }
    public static void main(String[] args) throws Exception {
        Configuration conf = new Configuration();
        Job job = Job.getInstance(conf, jobName: "High Price");
        job.setJarByClass(AllTimeHigh.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( verbose: true) ? 0 : 1);
    }
}
```



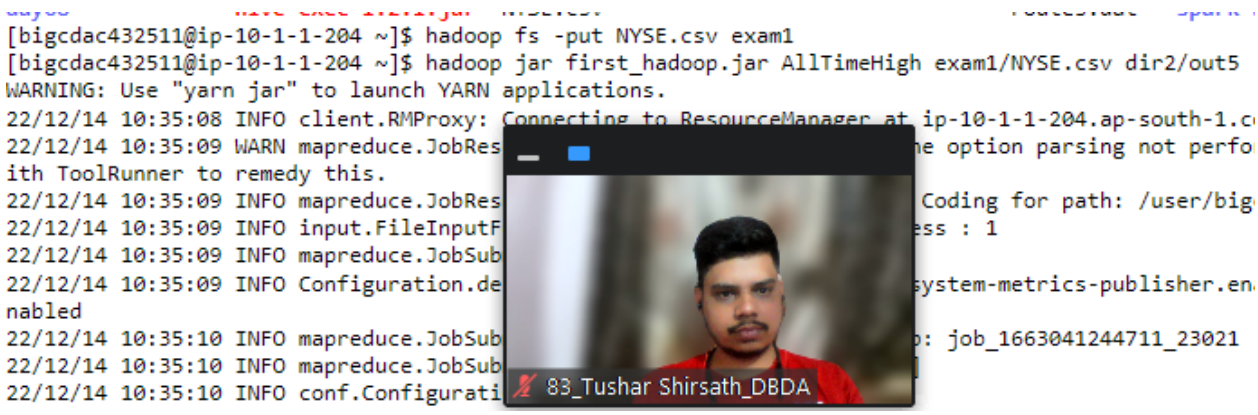
You a

```java
import java.io.*;

import org.apache.hadoop.io.DoubleWritable;
import org.apache.hadoop.io.Text;
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 AllTimeHigh {

    public static class MapClass extends
Mapper<LongWritable,Text,Text, DoubleWritable>
    {
```

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

            for (DoubleWritable val : values)
            {
                if(val.get() > max){
                    max = val.get();
                }
            }
            int a=10;
            result.set(max);
            context.write(key, result);

        }
    }
    public static void main(String[] args) throws Exception {
        Configuration conf = new Configuration();
        Job job = Job.getInstance(conf, "High Price");
        job.setJarByClass(AllTimeHigh.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);
    }
}
```

```
[bigcdac432511@ip-10-1-1-204 ~]$ hadoop fs -put NYSE.csv exam1
[bigcdac432511@ip-10-1-1-204 ~]$ hadoop jar first_hadoop.jar
AllTimeHigh exam1/NYSE.csv dir2/out5
```



Home

/ user / bigcdac432511 / dir2 / out5 / **part-r-00000**

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