**CDAC MUMBAI**

PG-DBDA SEP 2022 BATCH KHARGHAR

MODULE: BIG DATA ANALYTICS

DATE : 14TH 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. ---------------------------------------------------------------------------------------------------------------

**MapReduce**

**Q1.**

**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 1 Answer:**

****

**steps :**

----------------------------------------

**1) Launch the eclipse application using Java perspective**

**2) Click OK for Workspace Launcher**

**3) create a new project :**

**a) File ->New --->Java project**

**b) Enter "exam" as project name and click Finish**

**4) create a new java program**

**5) Expand the said project in the Package explorer**

**6) Right click on Hadoop project --->New --->Class**

**7) Enter the class name as "StockVolume" and click Finish**

**8) Enter program in the eclipse and save**

**import** java.io.\*;

**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** StockVolume {

**public** **static** **class** MapClass **extends** Mapper<LongWritable,Text,Text,LongWritable>

{

**public** **void** map(LongWritable key, Text value, Context context)

{

**try**{

String[] str = value.toString().split(",");

**long** vol = Long.*parseLong*(str[7]);

context.write(**new** Text(str[1]),**new** LongWritable(vol));

}

**catch**(Exception e)

{

System.***out***.println(e.getMessage());

}

}

}

**public** **static** **class** ReduceClass **extends** Reducer<Text,LongWritable,Text,LongWritable>

{

**private** LongWritable result = **new** LongWritable();

**public** **void** reduce(Text key, Iterable<LongWritable> values,Context context) **throws** IOException, InterruptedException {

**long** sum = 0;

**for** (LongWritable val : values)

{

sum += val.get();

}

result.set(sum);

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, "Volume Count");

job.setJarByClass(StockVolume.**class**);

job.setMapperClass(MapClass.**class**);

//job.setCombinerClass(ReduceClass.class);

job.setReducerClass(ReduceClass.**class**);

job.setNumReduceTasks(1);

job.setOutputKeyClass(Text.**class**);

job.setOutputValueClass(LongWritable.**class**);

FileInputFormat.*addInputPath*(job, **new** Path(args[0]));

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

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

}

}

**9) add external jar files for the project**

**a) Right click on Hadoop project --->Build path --->Configure build path**

**b) select the Libraries tab and click on "Add external jars"**

**c) click on File System and add 2 jars as under**

**hadoop-common.jar**

**hadoop-mapreduce-client-core.jar**

**d) After saving the changes there should not be any ERROR in java program**

**10) Create a jar file from class files**

**a) Right click on Hadoop project --->Export--->Java--->Jar file-->Next**

**b) enter the jar file name with full path in export destination**

**jar1.jar**

**c) click on Finish**

**11) Upload the file jar1.jar and NYSE.csv to client from your base machine.**

**12)create a training folder**

**-----------------------**

**hadoop fs -mkdir exam**

**13)upload NYSE.csv on hdfs**

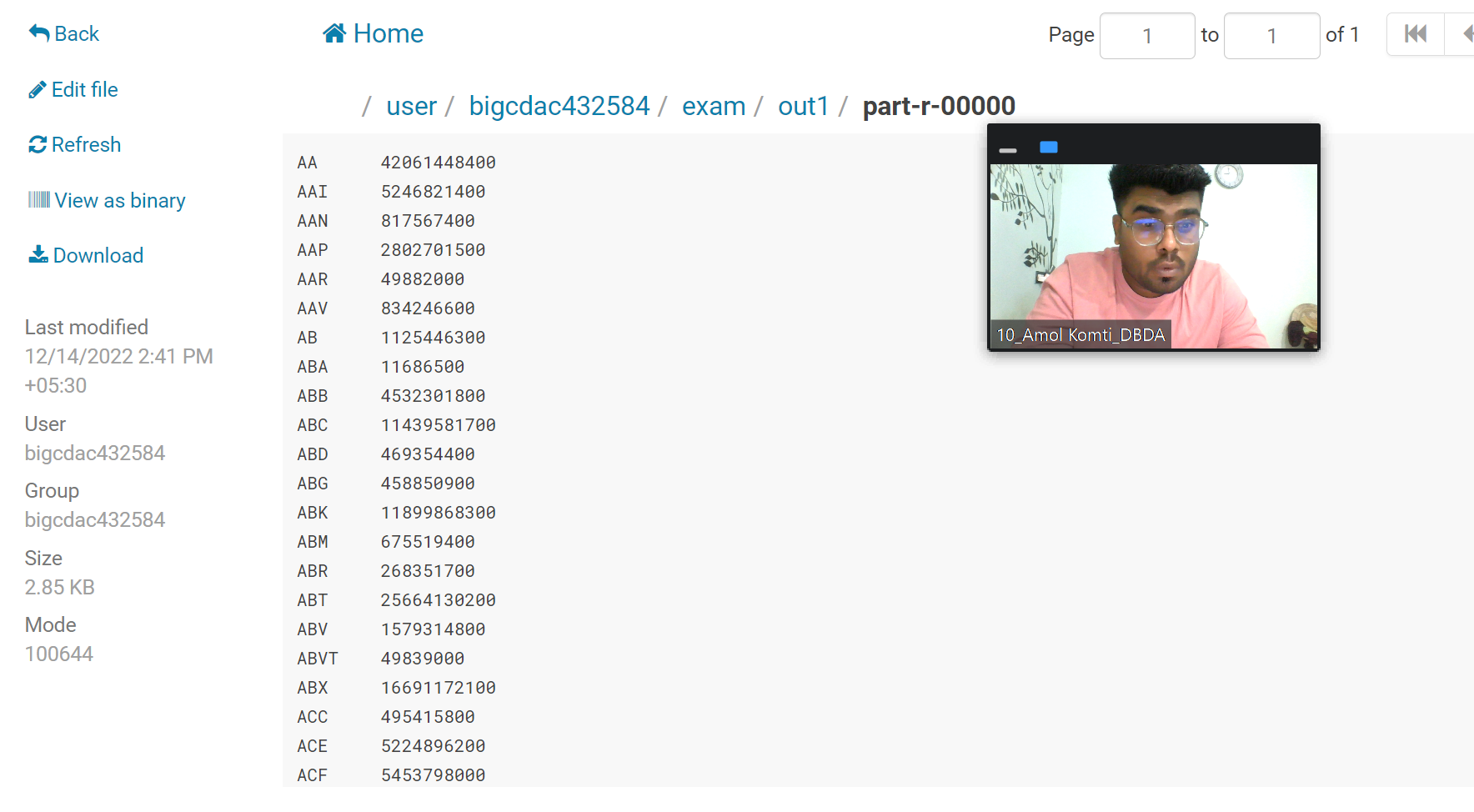
**------------------------**

**hadoop fs -put NYSE.csv exam**

**Run the java jar file on Hadoop for StockVolume**

**--------------------------------------------------------------------**

**hadoop jar jar1.jar StockVolume exam/NYSE.csv exam/out1**

****

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

**Question 2 Answer:**

****

**steps :**

----------------------------------------

**1) Launch the eclipse application using Java perspective**

**2) Click OK for Workspace Launcher**

**3) create a new project :**

**a) File ->New --->Java project**

**b) Enter "exam" as project name and click Finish**

**4) create a new java program**

**5) Expand the said project in the Package explorer**

**6) Right click on Hadoop project --->New --->Class**

**7) Enter the class name as "AllTimeHigh" and click Finish**

**8) Enter program in the eclipse and save**

**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>

{

**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 = 0.00;

**for** (DoubleWritable val : values)

{

**if** (val.get() > max) {

max = val.get();

}

}

result.set(max);

context.write(key, result);

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

}

}

**public** **static** **void** main(String[] args) **throws** Exception {

Configuration conf = **new** Configuration();

conf.set("mapreduce.output.textoutputformat.separator",",");

//conf.set("name", "value")

conf.set("mapreduce.input.fileinputformat.split.maxsize", "28311552");

Job job = Job.*getInstance*(conf, "All Time High 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);

}

}

**9) add external jar files for the project**

**a) Right click on Hadoop project --->Build path --->Configure build path**

**b) select the Libraries tab and click on "Add external jars"**

**c) click on File System and add 2 jars as under**

**hadoop-common.jar**

**hadoop-mapreduce-client-core.jar**

**d) After saving the changes there should not be any ERROR in java program**

**10) Create a jar file from class files**

**a) Right click on Hadoop project --->Export--->Java--->Jar file-->Next**

**b) enter the jar file name with full path in export destination**

**jar1.jar**

**c) click on Finish**

**11) Upload the file jar2.jar and NYSE.csv to client from your base machine.**

**12)create a training folder**

**-----------------------**

**hadoop fs -mkdir exam**

**13)upload NYSE.csv on hdfs**

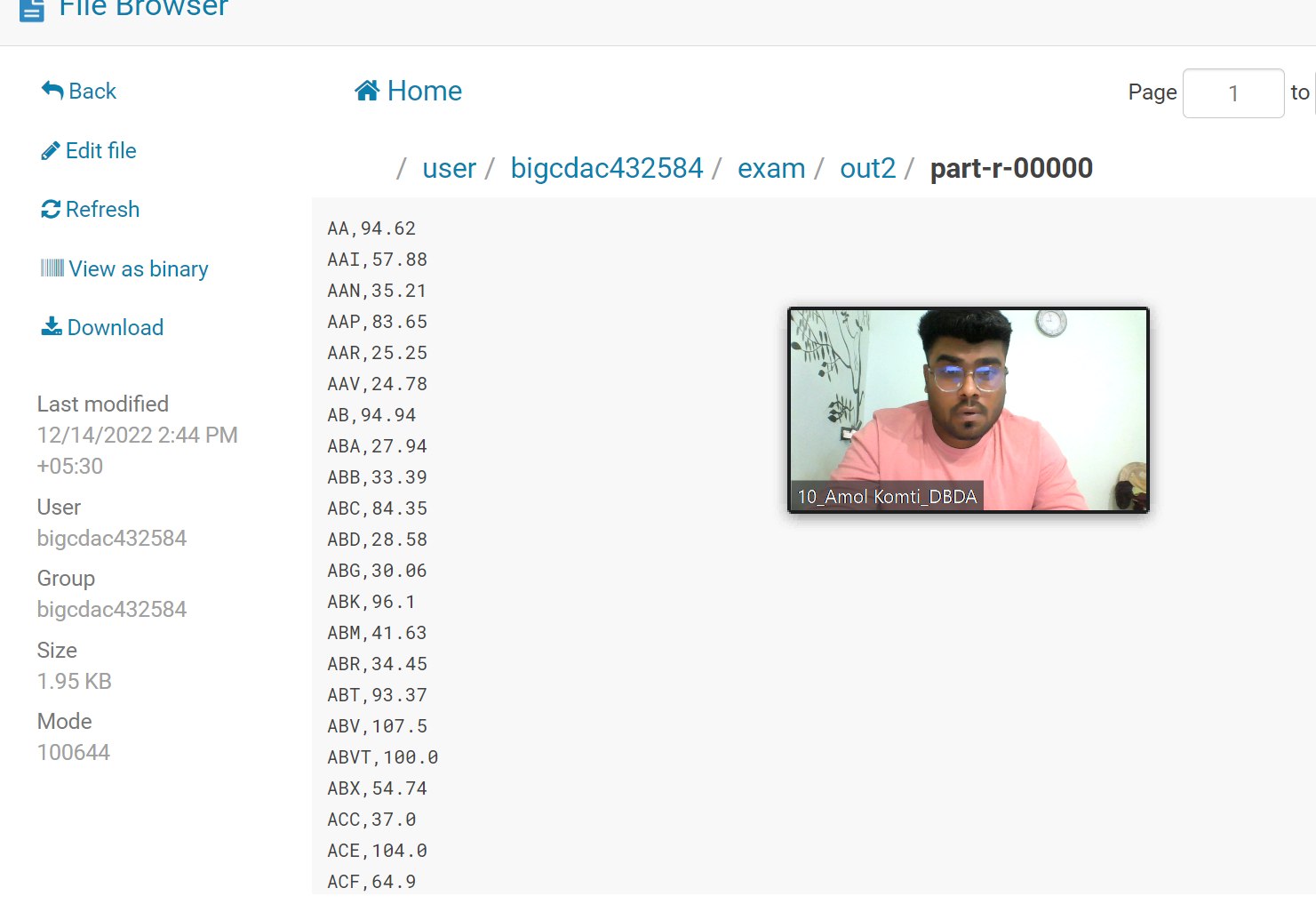
**------------------------**

**hadoop fs -put NYSE.csv exam**

**Run the java jar file on Hadoop for AllTimeHIgh**

**--------------------------------------------------------------------**

**hadoop jar jar2.jar AllTimeHIgh exam/NYSE.csv exam/out2**

****

**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.**

Please find the sales data set.

txn id

txn date

cust id

amount

category

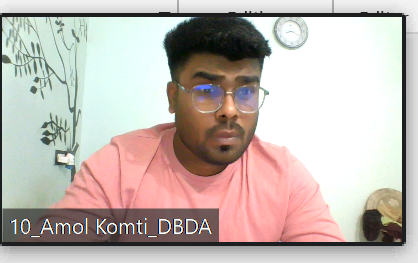
product

city

state

spendby

**Question 1 : Answer**

1. create database 

------------------

create database **exam\_amol**;

--this creates a folder by the name of **exam\_amol.db** under /user/hive/warehouse

2. show all the databases in hive

----------------------------------

**show databases;**

3. Select a database

--------------------

**use exam\_amol;**

4. Show tables under the database

----------------------------------

**show tables;**

5. Create transaction table

-------------------------------

**create table txnrecords(txnno INT, txndate STRING, custno INT, amount DOUBLE,**

**category STRING, product STRING, city STRING, state STRING, spendby STRING)**

**row format delimited**

**fields terminated by ','**

**stored as textfile;**

6. Create transaction customer

-------------------------------

**create table customer(custno INT, firstname STRING, lastname STRING, age INT, profession STRING)**

**row format delimited**

**fields terminated by ','**

**stored as textfile;**

7. Load the data into the table (from local file system)

---------------------------------------------------------------------------------

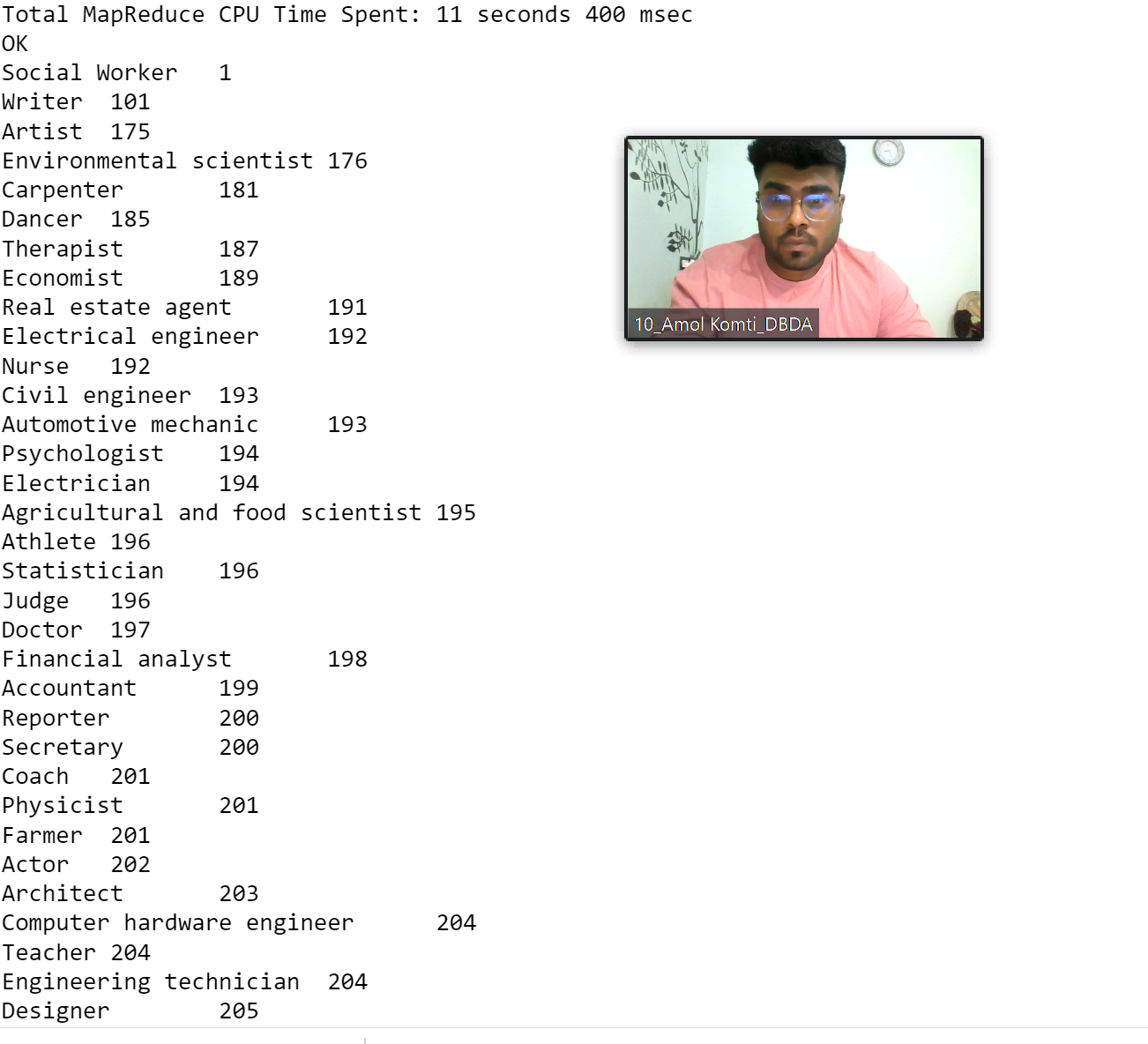
**LOAD DATA LOCAL INPATH 'txns1.txt' OVERWRITE INTO TABLE txnrecords;**

**LOAD DATA LOCAL INPATH 'custs.txt' OVERWRITE INTO TABLE customer;**

8. Count of customer for each profession

---------------------------------------------------

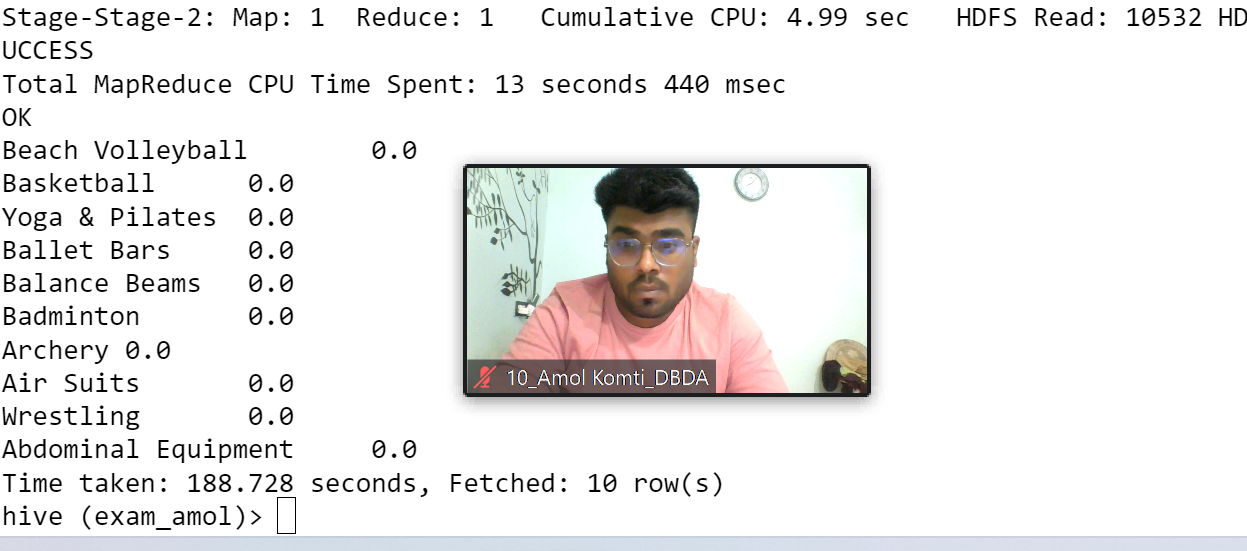
**select profession, count(\*) as headcount from customer group by profession order by headcount;**



**Q2) Write a program to find the top 10 products sales wise**

**Question 2 Answer:**

**Select product,sum(spendby) as total from txnrecords group by product order by total desc limit 10;**



**Q3) Write a program to create partiioned table on category**

**Question 3 Answer:**

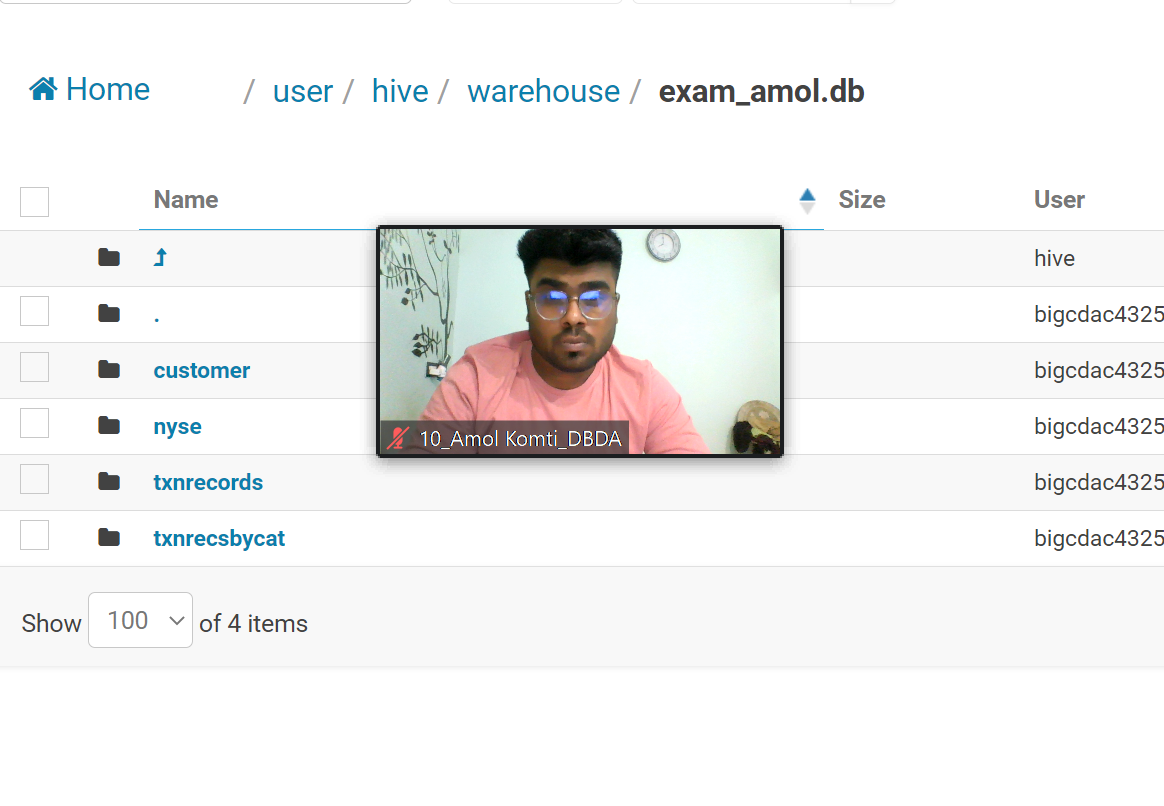
**create table txnrecsByCat(txnno INT, txndate STRING, custno INT, amount DOUBLE,product STRING, city STRING, state STRING, spendby STRING)**

**partitioned by (category STRING)**

**row format delimited**

**fields terminated by ','**

**stored as textfile;**



**Pyspark**

**QUESTION 3 [15 marks]**

**PySpark**

**Please find the AIRLINES data set**

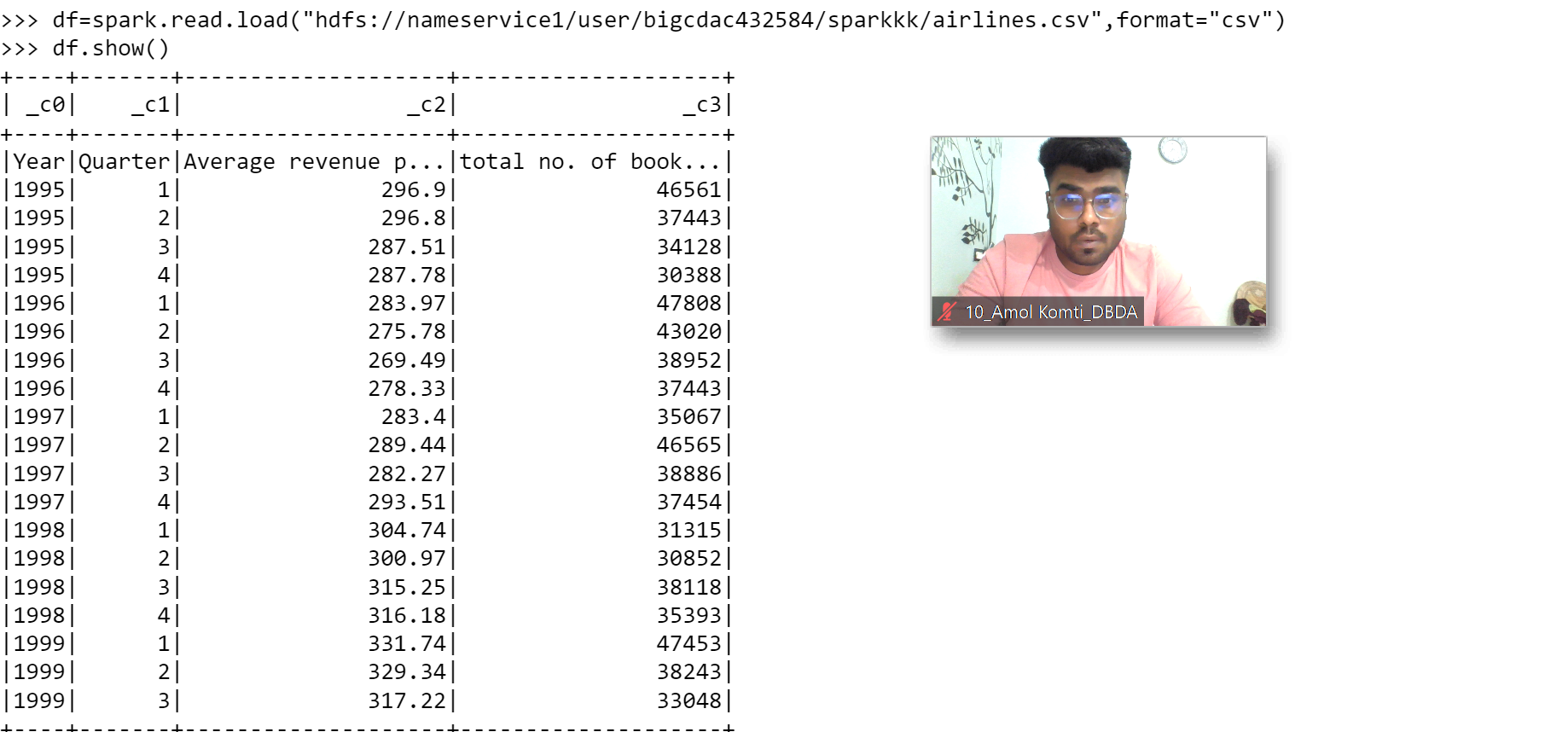
**Year**

**Quarter**

**Average revenue per seat**

**Total number of booked seats**

**Answer:**



**df.registerTempTable("airlines")**

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

**select year,sum(Total number of booked seats) as totalSeats from airlines group by year having totalSeats in**

**(select max(totalSeats) from ((select year,sum(Total number of booked seats) as totalSeats from airlines group by year)abctable));**

1. Identifying the highest revenue generation for which year

**select year,sum(Average revenue per seat \* Total number of booked seats) as avgRev from airlines group by year order by avgRev desc limit 1;**

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

**select year,quarter,sum(Average revenue per seat \* Total number of booked seats) as Rev from airlines group by year,quarter order by Rev desc limit 1;**