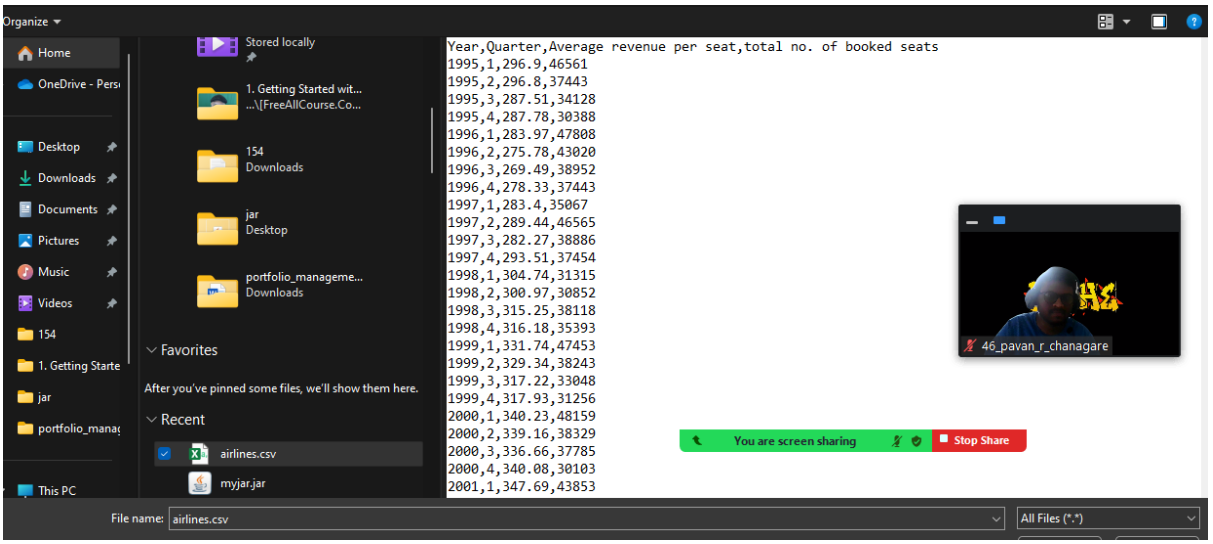
3)PYSPARK

UPLOAD THE AIRLINES DATA SET ON FTP

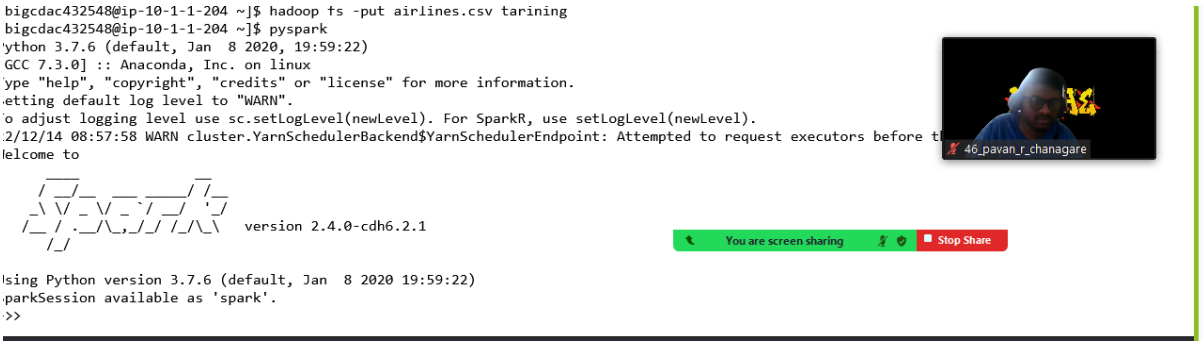


UPLOAD FILE FROM FTP TO HADOOP CLUSTER USING COMMANDS

hadoop fs -put airlines.csv training

START PYSPARK CLI

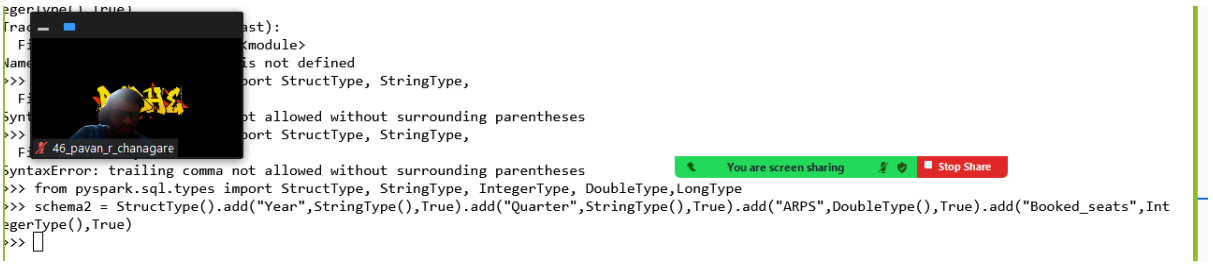
>>pyspark

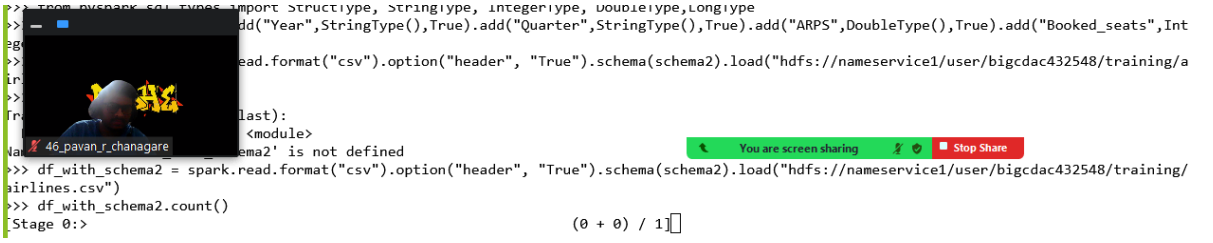


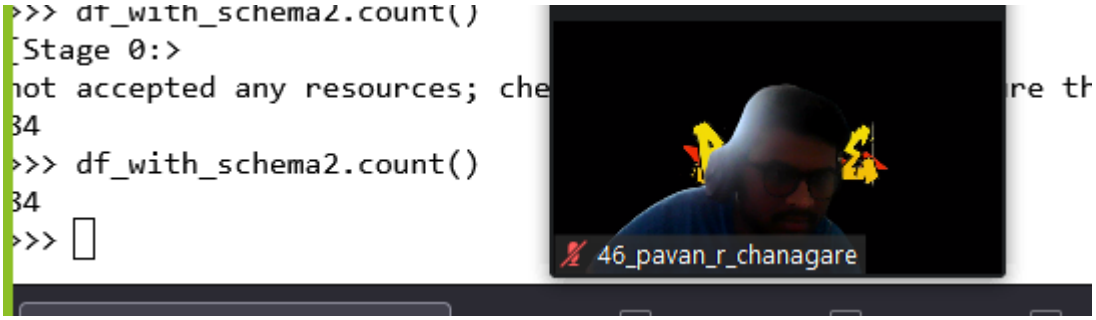
Create schema

```
>>> from pyspark.sql.types import StructType, StringType, IntegerType,
DoubleType,LongType
>>> schema2 =
StructType().add("Year",StringType(),True).add("Quarter",StringType(),True).add("ARPS",Do
ubleType(),True).add("Booked_seats",Int
egerType(),True)
>>>
df_with_schema2 = spark.read.format("csv").option("header",
"True").schema(schema2).load("hdfs://nameservice1/user/bigcdac432548/training/
airlines.csv")
>>> df_with_schema2.count()
```

[Stage 0:> (0 + 0) / 1]22/12/14 09:18:37 WARN cluster.YarnScheduler: Initial job has
not accepted any resources; check your cluster UI to ensure that workers are registered and have sufficient resources
84
>>> df_with_schema2.count()
84
df_with_schema2.registerTempTable("airlines")







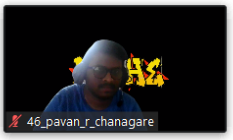
A)_What was the highest number of people travelled in which Year?

```
YrWisePsx =spark.sql("select year, sum(booked_seats) as total_psx
from airlines group by year order by total_psx desc")
```

>>YrWisePsx.show(15)

```
>>> YrWisePsx =spark.sql("select year, sum(booked_seats) as total_psx from airlines group by year order by total_psx desc")
>>> YrWisePsx.show(15)
+----+---------+
|year|total_psx|
+----+---------+
|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|
+----+---------+
only showing top 15 rows

>>>
```



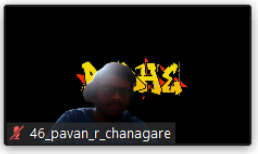
b)Identifying the highest revenue generation for which year

>>> df_with_schema2.count()
84
>>> df_with_schema2.registerTempTable("airlines")

>>> YrWiseRev =spark.sql("select year, sum(arps*booked_seats) as total from airlines group by year order by total desc")

>>> YrWiseRev.show(15)

```
>>> df_with_schema2.count()
84
>>> df_with_schema2.registerTempTable("airlines")
>>> YrWiseRev =spark.sql("select year, sum(arps*booked_seats) as total from airlines group by year order by total desc")
>>> YrWiseRev.show(15)
+----+--------------------+
|year|               total|
+----+--------------------+
|2013|       6.636320871E7|
|2014| 6.262417585000001E7|
|2015|       6.237899057E7|
|2012|       6.219912728E7|
|2008|5.7653170760000005E7|
|2007|       5.730921607E7|
|2001| 5.553377999999999E7|
|2010|       5.486152129E7|
|2000|5.2342926550000004E7|
|2011|       5.188828622E7|
|2004|5.0631364949999996E7|
|2006|5.0437898419999994E7|
|2003|       4.927321083E7|
|1999|       4.875771448E7|
|2002|        4.74991465E7|
+----+--------------------+
only showing top 15 rows
```



Remove e7

YrWiseRev =spark.sql("select year, sum(arps*booked_seats)/1000000 as total_in_mill from airlines group by year order by total_in_mill desc
")

>>> YrWiseRev.show(15)

```
>>> YrWiseRev =spark.sql("select year, sum(arps*booked_seats)/1000000 as total_in_mill from airlines group by year order by total_in_mill desc
")
>>> YrWiseRev.show(15)
+----+------------------+
|year|     total_in_mill|
+----+------------------+
|2013|       66.36320871|
|2014| 62.62417585000001|
|2015|       62.37899057|
|2012|       62.19912728|
|2008|       57.65317076|
|2007|       57.30921607|
|2001| 55.53377999999999|
|2010|       54.86152129|
|2000|       52.34292655|
|2011|       51.88828622|
|2004|       50.63136495|
|2006|50.437898419999996|
|2003|       49.27321083|
|1999|       48.75771448|
|2002|        47.4991465|
+----+------------------+
only showing top 15 rows
```

Rounding off

```
YrWiseRev =spark.sql("select year,
round(sum(arps*booked_seats)/1000000,2) as total_in_mill from
airlines group by year order by total_in_mill desc")
```

>>>>YrWiseRev.show(15)

```
>>> YrWiseRev =spark.sql("select year, round(sum(arps*booked_seats)/1000000,2) as total_in_mill from airlines group by year order by total_in_
mill desc")
>>> YrWiseRev.show(15)
+----+-------------+
|year|total_in_mill|
+----+-------------+
|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|
+----+-------------+
only showing top 15 rows
```

1)

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. 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

Make directory
hadoop fs -mkdir cdac

UPLOAD DATA SET NYSE.csv to ftp

UPLOAD TO HADOOP
[bigcdac432548@ip-10-1-1-204 ~]$ hadoop fs -put NYSE.csv exam

OPEN ECLIPSE AND WRITE THE JAVA CODE AND EXPORT THE JAR FILE

```
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);
              }
}
```

Project name: EXAMDAY

Use default location

Location: C:\Users\pavan\eclipse-workspace\EXAMDAY

Browse...

JRE

Use an execution environment JRE: JavaSE-1.8

Use a project specific JRE: jdk-18.0.2

Use defa preferences

Configure JREs...

Project layo

Use proj

Create s

Configure default...

46_pavan_r_chanagare

Working sets

Add project to working sets

New...

Working sets:

Select...

Module

< Back | Next > | Finish | Cancel

hadoop-common.jar - C:\Users\pavan\OneDrive\Desktop\jar

hadoop-mapreduce-client-core.jar - C:\Users\pavan\OneDrive\Desktop\jar

JRE System Library [JavaSE-1.8]

Add JARs...

Add External JARs...

Add Variable...

Add Library...

Add Class Folder...

Add External Class Folder...

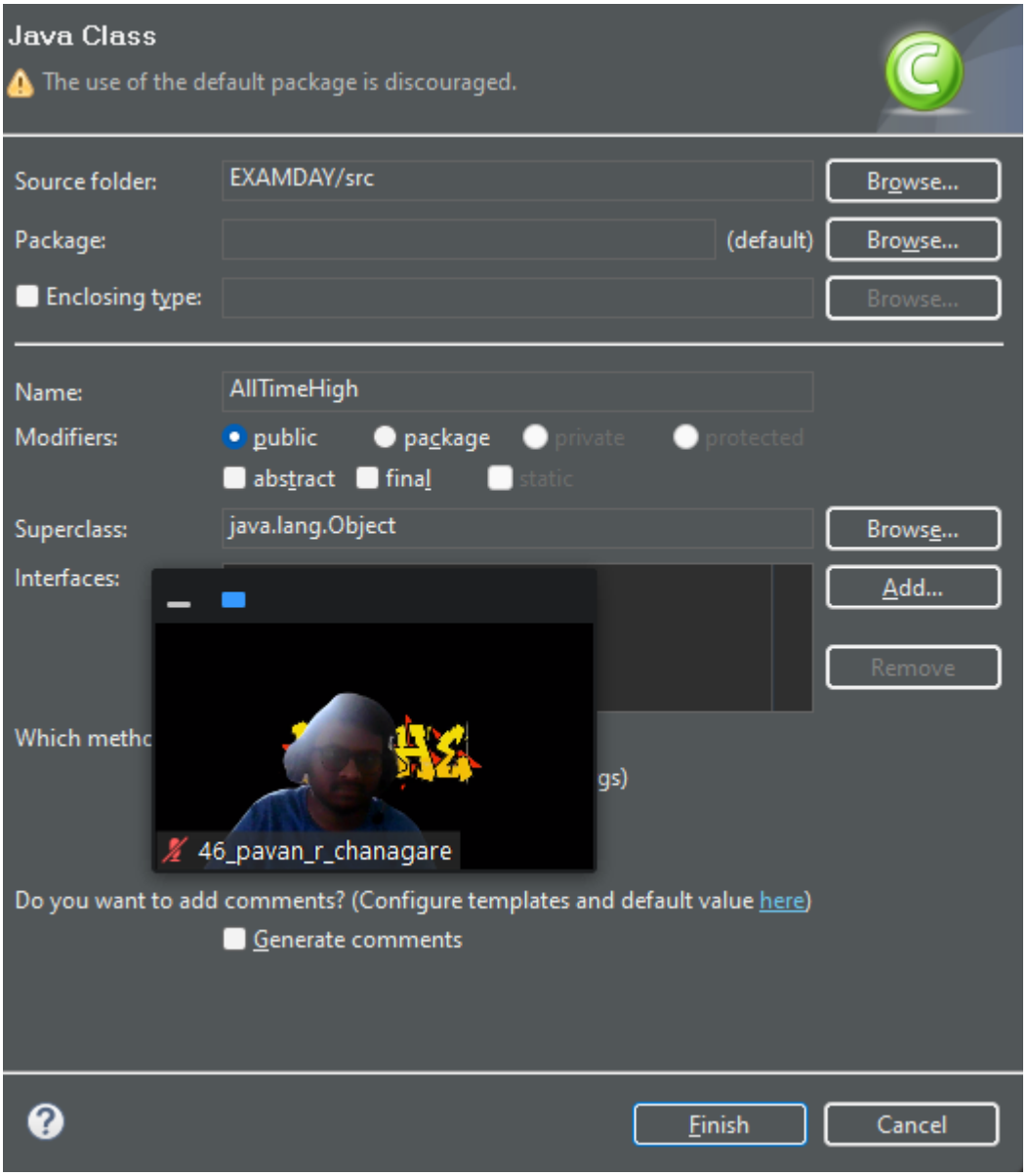
Edit...

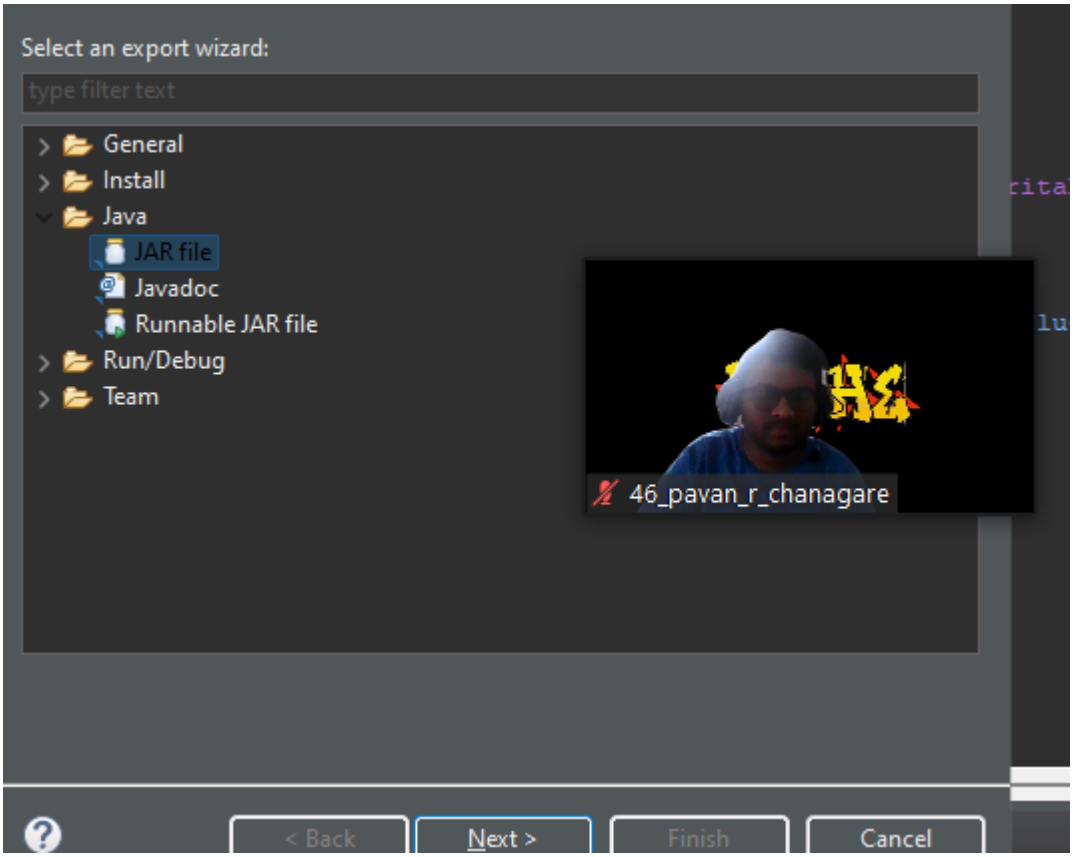
Remove

Migrate JAR File...

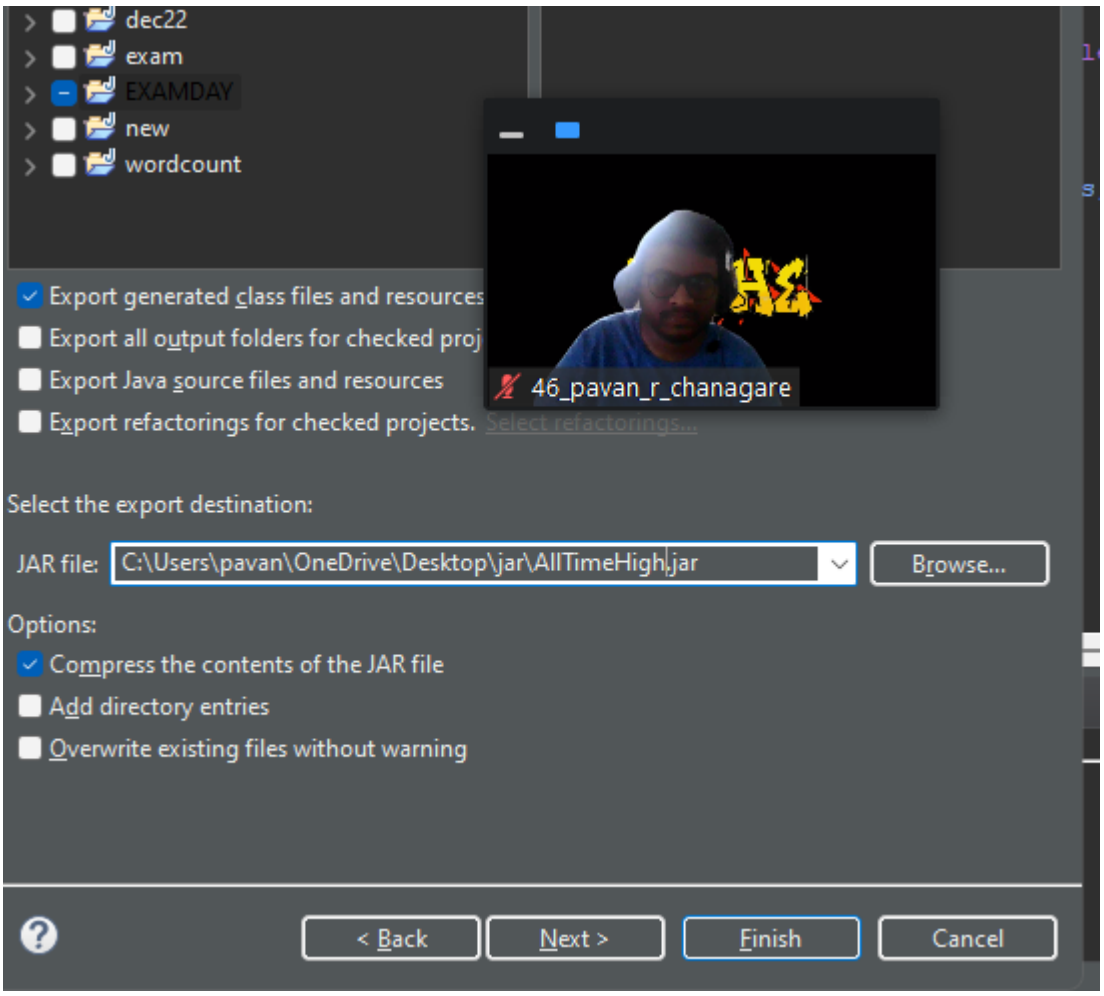
46_pavan_r_chanagare

< Back | Next > | Finish | Cancel


Java Class
The use of the default package is discouraged.
Source folder:
EXAMDAY/src
Browse...
Package:
(default)
Browse...
Enclosing type:
Browse...
Name:
AllTimeHigh
Modifiers:
public
package
private
protected
abstract
final
static
Superclass:
java.lang.Object
Browse...
Interfaces:
Add...
Remove
Which metho
gs)
46_pavan_r_chanagare
Do you want to add comments? (Configure templates and default value here)
Generate comments
Finish
Cancel

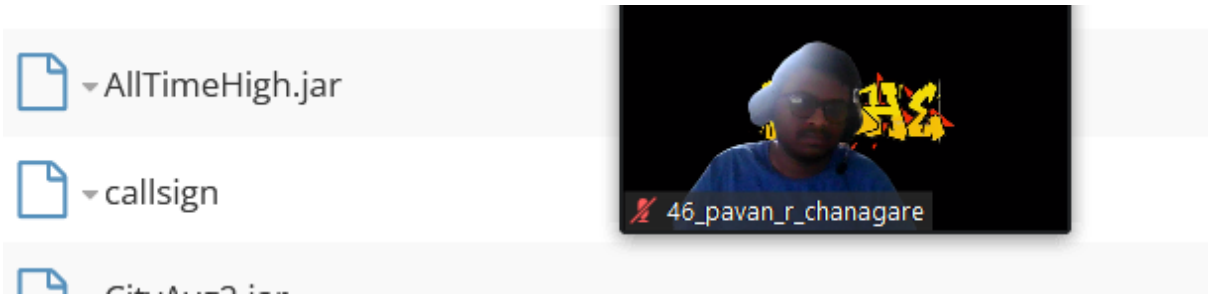


Field Separator – comma

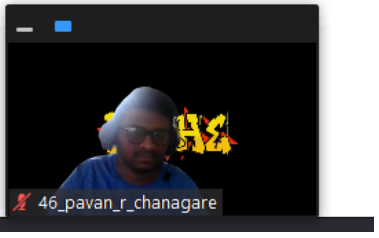


Question 2 : Find all time High price for each stock

Upload jar file on ftv



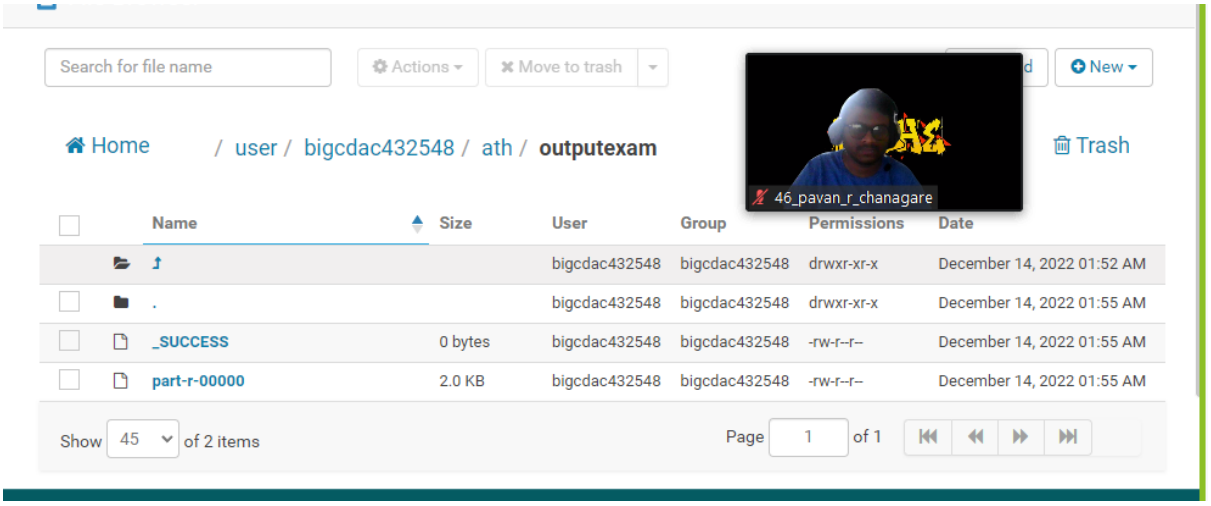

```
[bigcdac432548@ip-10-1-1-204 ~]$ hadoop fs -mkdir exam
[bigcdac432548@ip-10-1-1-204 ~]$ hadoop fs -put NYSE.csv exam
[bigcdac432548@ip-10-1-1-204 ~]$ jar tvf alltimehigh.jar
java.io.FileNotFoundException: alltimehigh.jar (No such file or directory)
        at java.util.zip.ZipFile.open(Native Method)
        at java.util.zip.ZipFile.<init>(ZipFile.java:225)
        at java.util.zip.ZipFile.<init>(ZipFile.java:155)
        at java.util.zip.ZipFile.<init>(ZipFile.java:126)
        at sun.tools.jar.Main.list(Main.java:1115)
        at sun.tools.jar.Main.run(Main.java:293)
        at sun.tools.jar.Main.main(Main.java:1288)
[bigcdac432548@ip-10-1-1-204 ~]$
```



[bigcdac432548@ip-10-1-1-204 ~]$ hadoop fs -mkdir exam
[bigcdac432548@ip-10-1-1-204 ~]$ hadoop fs -put NYSE.csv exam
[bigcdac432548@ip-10-1-1-204 ~]$ jar tvf alltimehigh.jar

[bigcdac432548@ip-10-1-1-204 ~]$ hadoop jar AllTimeHigh.jar AllTimeHigh exam/NYSE.CSV ath/outputexam;

```
[bigcdac432548@ip-10-1-1-204 ~]$ hadoop jar AllTimeHigh.jar AllTimeHigh ath/NYSE.csv ath/outputexam;
WARNING: Use "yarn jar" to launch YARN applications.
22/12/14 09:51:56 INFO client.RMProxy: Connecting to ResourceManager at ip-10-1-1-204.ap-south-1.c                    :8032
22/12/14 09:51:57 WARN mapreduce.JobResourceUploader: Hadoop command-line option parsing not perfo                    nterface and execu
te your application with ToolRunner to remedy this.
22/12/14 09:51:57 INFO mapreduce.JobResourceUploader: Disabling Erasure Coding for path: /user/big                    63041244711_22839
22/12/14 09:51:58 INFO input.FileInputFormat: Total input files to process : 1
22/12/14 09:51:58 INFO mapreduce.JobSubmitter: number of splits:1
22/12/14 09:51:59 INFO Configuration.deprecation: yarn.resourcemanager.system-metrics-publisher.en                    ad, use yarn.syste
m-metrics-publisher.enabled
22/12/14 09:51:59 INFO mapreduce.JobSubmitter: Submitting tokens for job: job_1663041244711_22839
22/12/14 09:51:59 INFO mapreduce.JobSubmitter: Executing with tokens: []
22/12/14 09:51:59 INFO conf.Configuration: resource-types.xml not found
22/12/14 09:51:59 INFO resource.ResourceUtils: Unable to find 'resource-types.xml'.
22/12/14 09:51:59 INFO impl.YarnClientImpl: Submitted application application_1663041244711_22839
22/12/14 09:51:59 INFO mapreduce.Job: The url to track the job: http://ip-10-1-1-204.ap-south-1.compute.internal:6066/proxy/application_166304
1244711_22839/
22/12/14 09:51:59 INFO mapreduce.Job: Running job: job_1663041244711_22839
22/12/14 09:52:23 INFO mapreduce.Job: Job job_1663041244711_22839 running in uber mode : false
22/12/14 09:52:23 INFO mapreduce.Job:  map 0% reduce 0%
22/12/14 09:52:33 INFO mapreduce.Job:  map 100% reduce 0%
```



on 2 : Find all time High price for each stock

[15 marks]

Hive
Please find the customer data set.
cust id
firstname
lastname
age
profession

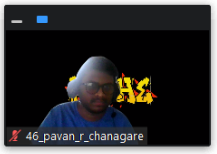
Start hive

[bigcdac432548@ip-10-1-1-204 ~]$ hive

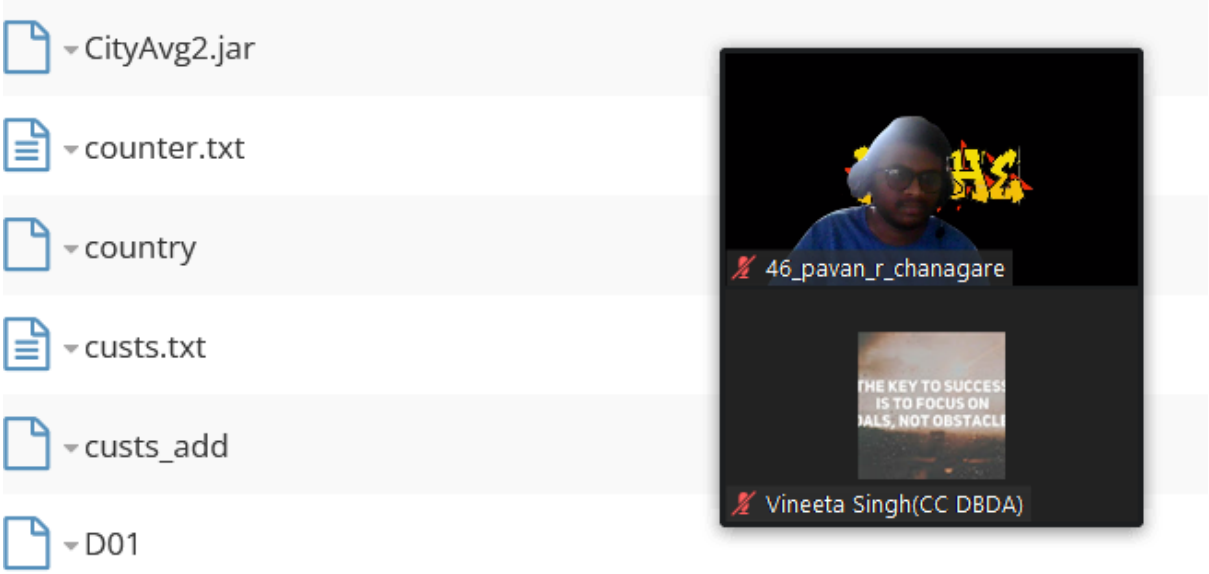
Create and select database with database indicator command

hive> set hive.cli.print.current.db = true;
hive (default)> use pavanhive;
desc customer;

```
hive (pavanhive)> create table customer(custno INT, firstname STRING, lastname STRING, age INT, profession STRING) row format delimited fields
 terminated by ',' stored as textfile;
FAILED: Execution Error, return code 1 from org.apache.hadoop.hive.ql.exec.DDLTask. AlreadyExistsException(message:Table customer already exis
ts)
hive (pavanhive)> desc customer;
OK
custno                  int
firstname               string
lastname                string
age                     int
profession              string
Time taken: 0.48 seconds, Fetched: 5 row(s)
hive (pavanhive)>
```



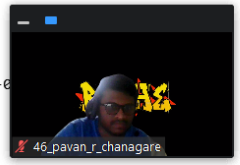
Upload custs.txt to ftp



Load custs.txt from ftp to hadoop

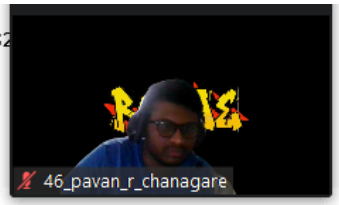
```
LOAD DATA LOCAL INPATH 'custs.txt' OVERWRITE INTO TABLE customer;
hive (pavanhive)> Select count(custno) from customer;
```

```
age                     int
profession              string
Time taken: 0.48 seconds, Fetched: 5 row(s)
hive (pavanhive)> LOAD DATA LOCAL INPATH 'custs.txt' OVERWRITE INTO TABLE customer;
Loading data to table pavanhive.customer
OK
Time taken: 1.551 seconds
hive (pavanhive)> Select count(custno) from customer;
Query ID = bigcdac432548_20221214101102_6ea141c1-c78d-44bf-a936-0
Total jobs = 1
Launching Job 1 out of 1
Number of reduce tasks determined at compile time: 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 10:11:03 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 10:11:04 INFO client.RMProxy: Connecting to ResourceManager at ip-10-1-1-204.ap-south-1.compute.internal/10.1.1.204:8032
Starting Job = job_1663041244711_22941, Tracking URL = http://ip-10-1-1-204.ap-south-1.compute.internal:6066/proxy/application_166304124471
2941/
```



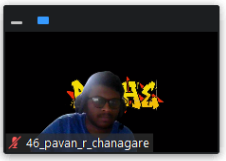

```
Select count(custno) from customer;
```

```
hive (pavanhive)> Select count(custno) from customer;
Query ID = bigcdac432548_20221214101924_964394f7-5363-4df3-8c82
Total jobs = 1
Launching Job 1 out of 1
Number of reduce tasks determined at compile time: 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:
```



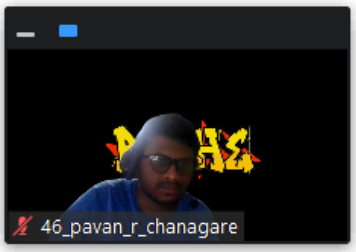

```
hive (pavanhive)> create table txnrecords(txnno INT, txndate
STRING, custno INT, amount DOUBLE,  category STRING, product
STRING, city STRING,
state STRING, spendby STRING) row format delimited fields
terminated by ',' stored as textfile location
'/user/bigcdac432581/sales';
```

```
hive (pavanhive)> 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 location '/user/bigcdac432581/sales';
```




```
hive (pavanhive)> desc txnrecords;
```

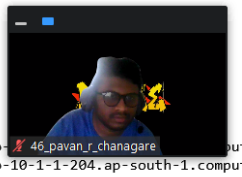
```
hive (pavanhive)> desc txnrecords;
OK
txnno                   int
txndate                 string
custno                  int
amount                  double
category                string
product                 string
city                    string
state                   string
spendby                 string
Time taken: 0.073 seconds, Fetched: 9 row(s)
hive (pavanhive)>
```



profession:

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

```
hive (pavanhive)> select profession, count(*) as headcount from customer group by profession order by headcount;
Query ID = bigcdac432548_20221214103008_9780a1e7-e6a6-455b-bbba-e3790042e1a5
Total jobs = 2
Launching Job 1 out of 2
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 10:30:09 INFO client.RMProxy: Connecting to ResourceManager at ip-[redacted]ute.internal/10.1.1.204:8032
22/12/14 10:30:09 INFO client.RMProxy: Connecting to ResourceManager at ip-10-1-1-204.ap-south-1.compute.internal/10.1.1.204:8032
Starting Job = job_1663041244711_23005, Tracking URL = http://ip-10-1-1-204.ap-south-1.compute.internal:6066/proxy/application_1663041244711_23005/
Kill Command = /opt/cloudera/parcels/CDH-6.2.1-1.cdh6.2.1.p0.1425774/lib/hadoop/bin/hadoop job  -kill job_1663041244711_23005
Hadoop job information for Stage-1: number of mappers: 1; number of reducers: 1
2022-12-14 10:30:43,669 Stage-1 map = 0%,  reduce = 0%
```




```
OK
Social Worker   1
Writer  101
Artist  175
Environmental scientist 176
Carpenter       181
Dancer  185
Therapist       187
Economist       189
Real estate agent       191
Electrical engineer     192
Nurse   192
Civil engineer  193
Automotive mechanic     193
Psychologist    194
Electrician     194
Agricultural and food scientist 195
Athlete 196
Statistician    196
Judge   196
Doctor  197
Financial analyst       198
```

3) Write a program to create partiioned table on category
QUESTION 3 [15 marks]

```
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;
```

```
hive (pavanhive)>  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;
FAILED: Execution Error, return code 1 from org.apache.hadoop.hive.ql.exec.DDLTask. AlreadyExistsException(message:Table txnrecsByCat already
exists)
hive (pavanhive)>
```