QUESTION 1

MapReduce Question :-

*******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(new Text(str[1]),new LongWritable(vol));
                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 temp_val=0;

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

		 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, "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);
		}
}
```

```
select count(*) from nyse;
```

```
2022-06-20 09:39:42,945 Stage-1 map = 0%,  reduce = 0%
2022-06-20 09:39:57,133 Stage-1 map = 100%,  reduce = 0%, Cumulative
CPU 4.75 sec
```

```
2022-06-20 09:40:06,416 Stage-1 map = 100%,  reduce = 100%,
Cumulative CPU 7.56 sec
MapReduce Total cumulative CPU time: 7 seconds 560 msec
Ended Job = job_1654490426372_5707
MapReduce Jobs Launched:
Stage-Stage-1: Map: 1  Reduce: 1   Cumulative CPU: 7.56 sec   HDFS
Read: 41000131 HDFS Write: 106 HDFS EC Read: 0 SUCCESS
Total MapReduce CPU Time Spent: 7 seconds 560 msec
OK
735026
Time taken: 59.27 seconds, Fetched: 1 row(s)


select stock_id, max(high) from nyse group by stock_id;
Hadoop job information for Stage-1: number of mappers: 1; number of
reducers: 1
2022-06-20 09:44:09,105 Stage-1 map = 0%,  reduce = 0%
2022-06-20 09:44:58,510 Stage-1 map = 100%,  reduce = 0%, Cumulative
CPU 4.98 sec
2022-06-20 09:45:08,917 Stage-1 map = 100%,  reduce = 100%,
Cumulative CPU 8.73 sec
MapReduce Total cumulative CPU time: 8 seconds 730 msec
Ended Job = job_1654490426372_5731
MapReduce Jobs Launched:
Stage-Stage-1: Map: 1  Reduce: 1   Cumulative CPU: 8.73 sec   HDFS
Read: 41000641 HDFS Write: 4521 HDFS EC Read: 0 SUCCESS
Total MapReduce CPU Time Spent: 8 seconds 730 msec
OK
AA      94.62
AAI     57.88
AAN     35.21
AAP     83.65
AAR     25.25
AAV     24.78
AB      94.94
.
.
.
.
.
.
.

Time taken: 29.413 seconds, Fetched: 203 row(s)
```

QUESTION 2

```
hive (navnath5028)>
create table customers(
cust_id INT,
firstname STRING,
lastname STRING,
age INT,
profession STRING
)
row format delimited
fields terminated by ','
stored as textfile;
OK
Time taken: 0.165 seconds

hive (navnath5028)> describe customer;
OK
cust_id                   int
firstname                string
lastname                 string
age                      int
profession               string


hive (navnath5028)> show tables;
OK
airlines
airport
customer

hive (navnath5028)> select count(*) from customer;
Hadoop job information for Stage-1: number of mappers: 2; number of
reducers: 1
2022-06-20 10:08:06,185 Stage-1 map = 0%,  reduce = 0%
2022-06-20 10:08:17,707 Stage-1 map = 50%,  reduce = 0%, Cumulative
CPU 3.19 sec
2022-06-20 10:08:18,753 Stage-1 map = 100%,  reduce = 0%, Cumulative
CPU 6.11 sec
2022-06-20 10:08:34,344 Stage-1 map = 100%,  reduce = 100%,
Cumulative CPU 8.62 sec
MapReduce Total cumulative CPU time: 8 seconds 620 msec
Ended Job = job_1654490426372_5831
MapReduce Jobs Launched:
```

```
Stage-Stage-1: Map: 2  Reduce: 1   Cumulative CPU: 8.62 sec   HDFS
Read: 404450 HDFS Write: 105 HDFS EC Read: 0 SUCCESS
Total MapReduce CPU Time Spent: 8 seconds 620 msec
OK
10000
```

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

```
hive (navnath5028)>
```

select profession, count(profession) from customer group by profession;

```
Total MapReduce CPU Time Spent: 9 seconds 450 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
Computer support specialist     222
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
```

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

```
hive (navnath5028)>
create table sales(
txn_id INT,
txn_date STRING,
cust_id INT,
amount DOUBLE,
category STRING,
product STRING,
city STRING,
state STRING,
spendby STRING)
row format delimited
fields terminated by ','
stored as textfile;
OK
Time taken: 0.103 seconds


hive (navnath5028)> describe sales;
OK
txn_id                  int
txn_date                string
cust_id                 int
amount                  double
category                string
product                 string
city                    string
state                   string
spendby                 string
Time taken: 0.062 seconds, Fetched: 9 row(s)
```

select product,round(sum(amount),2) as amt from sales group by product order by amt desc limit 10;

```
Total MapReduce CPU Time Spent: 11 seconds 960 msec
OK
Yoga & Pilates  47804.94
```

```
Swing Sets      47204.14
Lawn Games      46828.44
Golf    46577.68
Cardio Machine Accessories      46485.54
Exercise Balls  45143.84
Weightlifting Belts     45111.68
Mahjong 44995.2
Basketball      44954.68
Beach Volleyball        44890.67
Time taken: 56.32 seconds, Fetched: 10 row(s)
```

3) Write a program to create partitioned table on category

```
   hive (navnath5028)>
create table txnrecsByCat(
txn_id INT,
txn_date STRING,
cust_id INT,
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: 0.165 seconds
```

QUESTION 3 PySpark

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

```
dataRDD=sc.textFile("/user/bigdatamind43836/airlines.csv")
dataRDD2=dataRDD.map(lambda a : a.encode("ascii","ignore"))
header=dataRDD2.first()
```

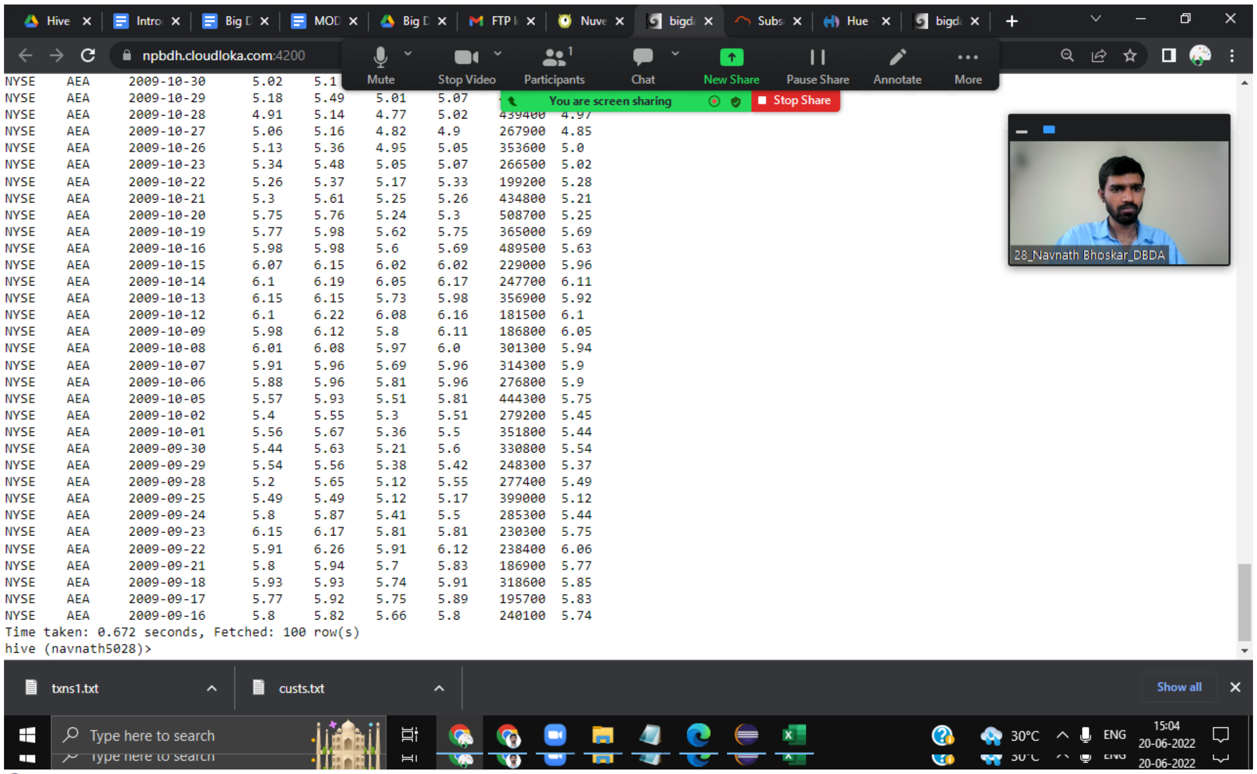
```
dataRDD3=dataRDD2.map(lambda a : a != header)
dataRDD4=dataRDD3.map(lambda a : a.split(","))
keyword=dataRDD4.map(lambda a : (a[0],int(a[3])))
count=keyword.reduceByKey(lambda a,b : a+b)
sortbyval=count.sortBy(lambda a : -a[1])

for line in sortbyval.collect():
     print(line)
...     print(i)
... output

('2007', 176299)
```

2) Identifying the highest revenue generation for which year

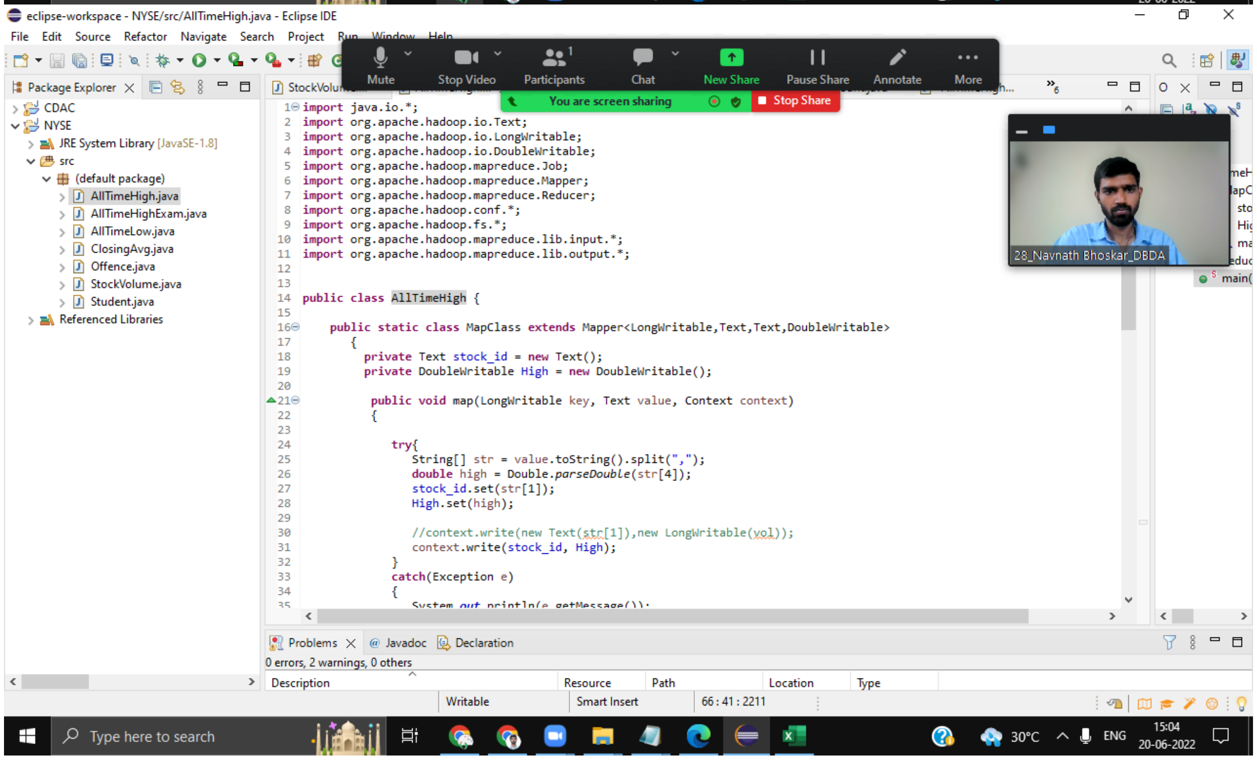
```
kvrdd2 = arrayrdd.map(lambda a : (a[0]+"
"+a[1],float(a[2])*int(a[3])))
counts2 = kvrdd2.reduceByKey(lambda a,b : a+b)
sort2 = counts2.sortBy(lambda a : -a[1])
>>> for i in sort2.take(1):
...     print(i)
...
('2014 4', 18819408.48)
```

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

```
kvrdd = arrayrdd.map(lambda a :(a[0],float(a[2])*int(a[3])))
counts = kvrdd.reduceByKey(lambda a,b : a+b)
>>> sort = counts.sortBy(lambda a : -a[1])
>>> for i in sort.take(1):
...     print(i)
('2013', 66363208.71)
```

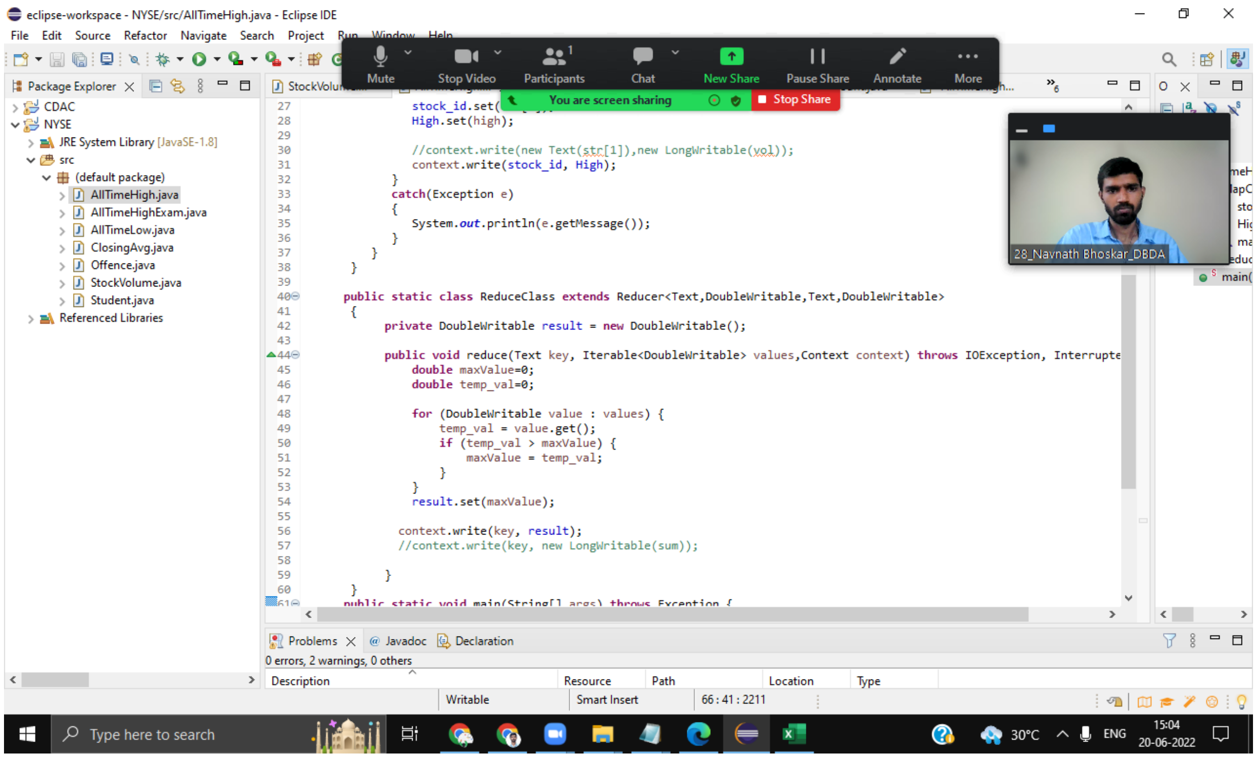


```
NYSE    AEA     2009-10-30      5.02    5.1
NYSE    AEA     2009-10-29      5.18    5.49    5.01    5.07
NYSE    AEA     2009-10-28      4.91    5.14    4.77    5.02    439400  4.97
NYSE    AEA     2009-10-27      5.06    5.16    4.82    4.9     267900  4.85
NYSE    AEA     2009-10-26      5.13    5.36    4.95    5.05    353600  5.0
NYSE    AEA     2009-10-23      5.34    5.48    5.05    5.07    266500  5.02
NYSE    AEA     2009-10-22      5.26    5.37    5.17    5.33    199200  5.28
NYSE    AEA     2009-10-21      5.3     5.61    5.25    5.26    434800  5.21
NYSE    AEA     2009-10-20      5.75    5.76    5.24    5.3     508700  5.25
NYSE    AEA     2009-10-19      5.77    5.98    5.62    5.75    365000  5.69
NYSE    AEA     2009-10-16      5.98    5.98    5.6     5.69    489500  5.63
NYSE    AEA     2009-10-15      6.07    6.15    6.02    6.02    229000  5.96
NYSE    AEA     2009-10-14      6.1     6.19    6.05    6.17    247700  6.11
NYSE    AEA     2009-10-13      6.15    6.15    5.73    5.98    356900  5.92
NYSE    AEA     2009-10-12      6.1     6.22    6.08    6.16    181500  6.1
NYSE    AEA     2009-10-09      5.98    6.12    5.8     6.11    186800  6.05
NYSE    AEA     2009-10-08      6.01    6.08    5.97    6.0     301300  5.94
NYSE    AEA     2009-10-07      5.91    5.96    5.69    5.96    314300  5.9
NYSE    AEA     2009-10-06      5.88    5.96    5.81    5.96    276800  5.9
NYSE    AEA     2009-10-05      5.57    5.93    5.51    5.81    444300  5.75
NYSE    AEA     2009-10-02      5.4     5.55    5.3     5.51    279200  5.45
NYSE    AEA     2009-10-01      5.56    5.67    5.36    5.5     351800  5.44
NYSE    AEA     2009-09-30      5.44    5.63    5.21    5.6     330800  5.54
NYSE    AEA     2009-09-29      5.54    5.56    5.38    5.42    248300  5.37
NYSE    AEA     2009-09-28      5.2     5.65    5.12    5.55    277400  5.49
NYSE    AEA     2009-09-25      5.49    5.49    5.12    5.17    399000  5.12
NYSE    AEA     2009-09-24      5.8     5.87    5.41    5.5     285300  5.44
NYSE    AEA     2009-09-23      6.15    6.17    5.81    5.81    230300  5.75
NYSE    AEA     2009-09-22      5.91    6.26    5.91    6.12    238400  6.06
NYSE    AEA     2009-09-21      5.8     5.94    5.7     5.83    186900  5.77
NYSE    AEA     2009-09-18      5.93    5.93    5.74    5.91    318600  5.85
NYSE    AEA     2009-09-17      5.77    5.92    5.75    5.89    195700  5.83
NYSE    AEA     2009-09-16      5.8     5.82    5.66    5.8     240100  5.74
Time taken: 0.672 seconds, Fetched: 100 row(s)
hive (navnath5028)>
```



```java
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(new Text(str[1]),new LongWritable(vol));
                context.write(stock_id, High);
             }
             catch(Exception e)
             {
                System.out.println(e.getMessage());
```

0 errors, 2 warnings, 0 others

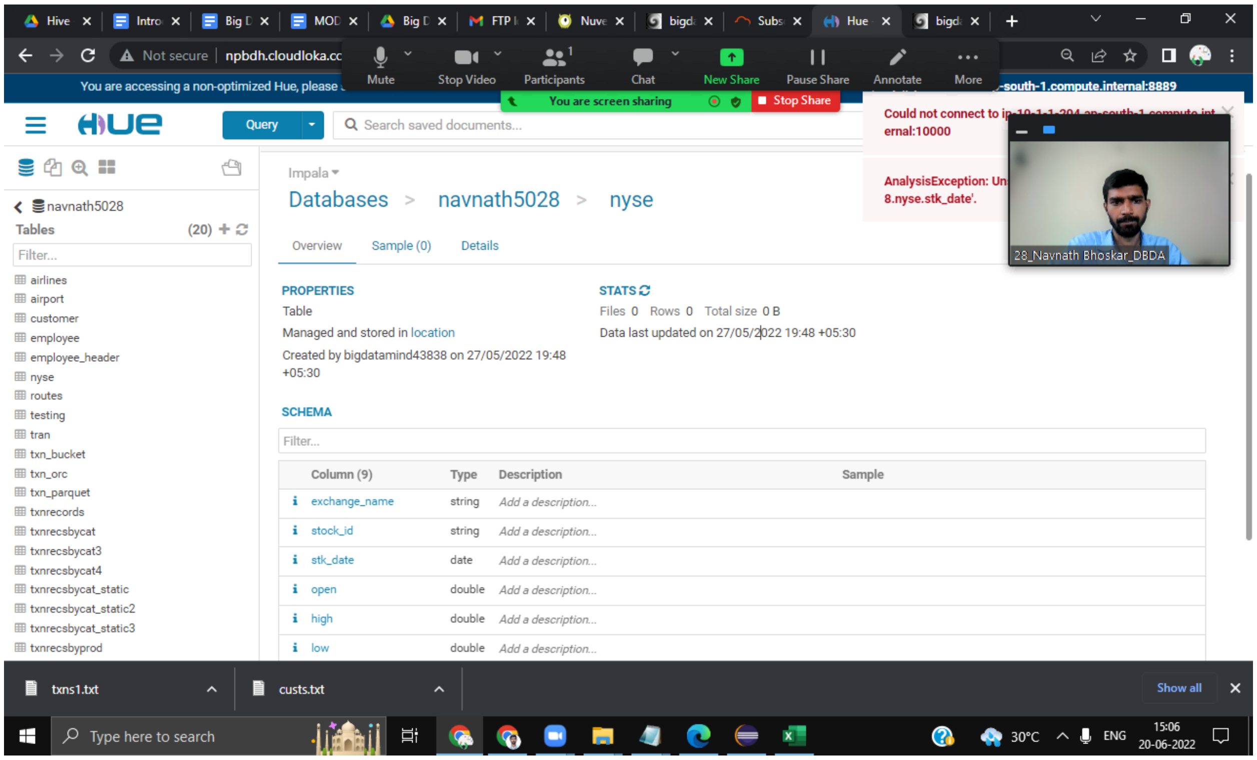


eclipse-workspace - NYSE/src/AllTimeHigh.java - Eclipse IDE

```
            stock_id.set(
            High.set(high);

            //context.write(new Text(str[1]),new LongWritable(vol));
            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, Interrupte
            double maxValue=0;
            double temp_val=0;

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

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

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

0 errors, 2 warnings, 0 others

Writable | Smart Insert | 66 : 41 : 2211



Could not connect to ip-10-1-1-204.ap-south-1.compute.internal:10000

AnalysisException: Un 8.nyse.stk_date'.

Impala

Databases > navnath5028 > nyse

Overview | Sample (0) | Details

PROPERTIES

Table

Managed and stored in location

Created by bigdatamind43838 on 27/05/2022 19:48 +05:30

STATS

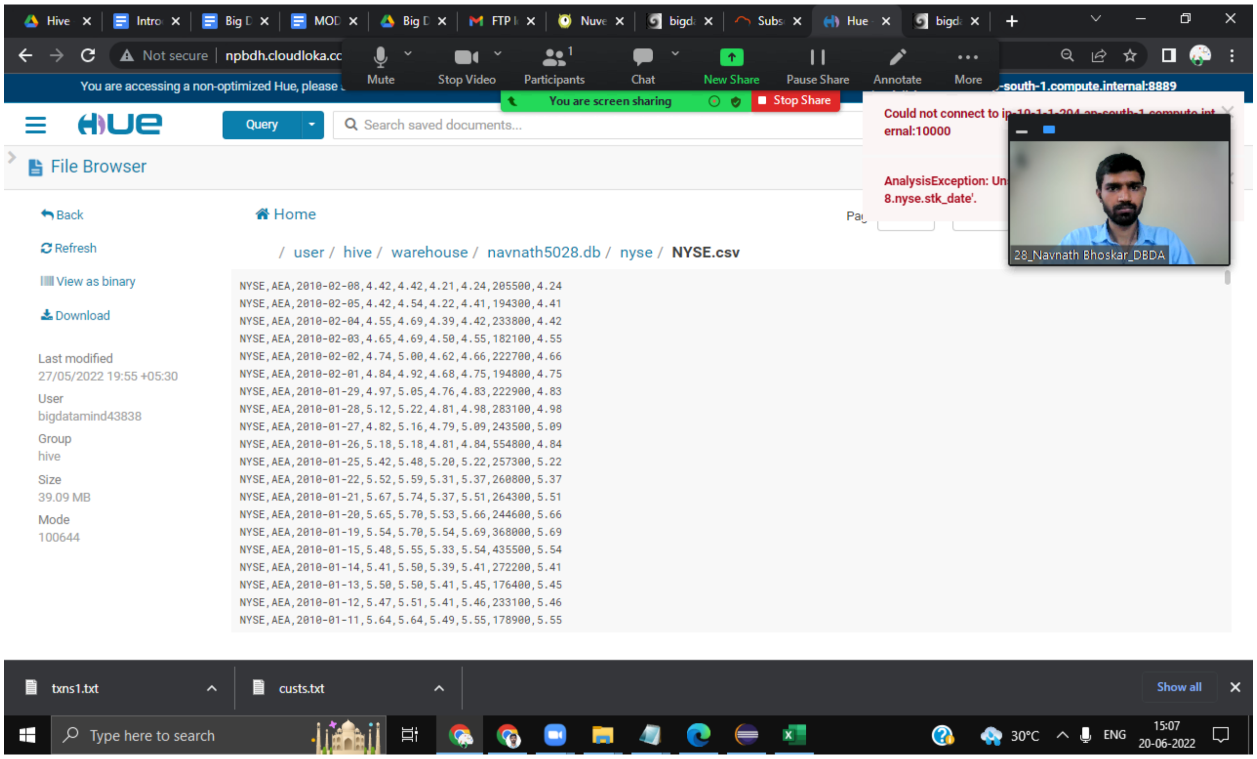
Files 0 Rows 0 Total size 0 B

Data last updated on 27/05/2022 19:48 +05:30

SCHEMA

Column (9)	Type	Description	Sample
exchange_name	string	Add a description...	
stock_id	string	Add a description...	
stk_date	date	Add a description...	
open	double	Add a description...	
high	double	Add a description...	
low	double	Add a description...	

navnath5028 Tables (20): airlines, airport, customer, employee, employee_header, nyse, routes, testing, tran, txn_bucket, txn_orc, txn_parquet, txnrecords, txnrecsbycat, txnrecsbycat3, txnrecsbycat4, txnrecsbycat_static, txnrecsbycat_static2, txnrecsbycat_static3, txnrecsbyprod

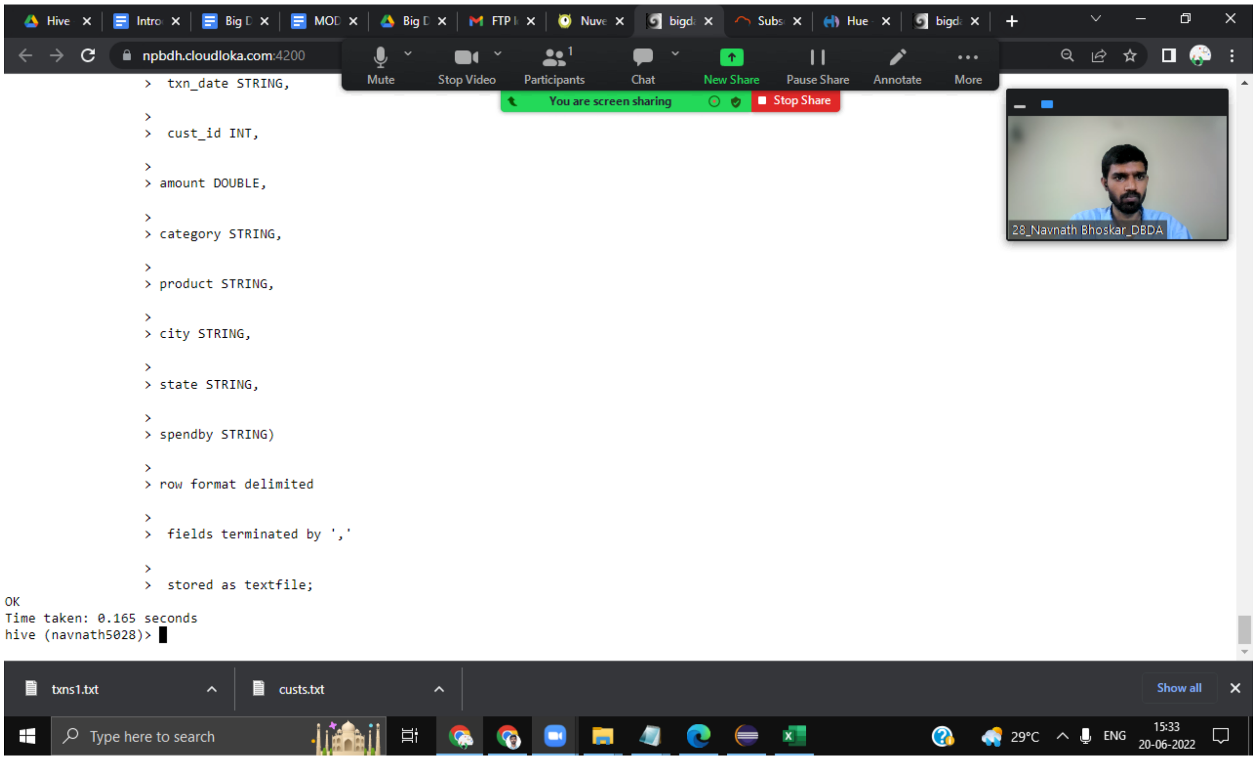

File Browser
Back
Refresh
View as binary
Download
Home
/ user / hive / warehouse / navnath5028.db / nyse / NYSE.csv

Last modified
27/05/2022 19:55 +05:30
User
bigdatamind43838
Group
hive
Size
39.09 MB
Mode
100644

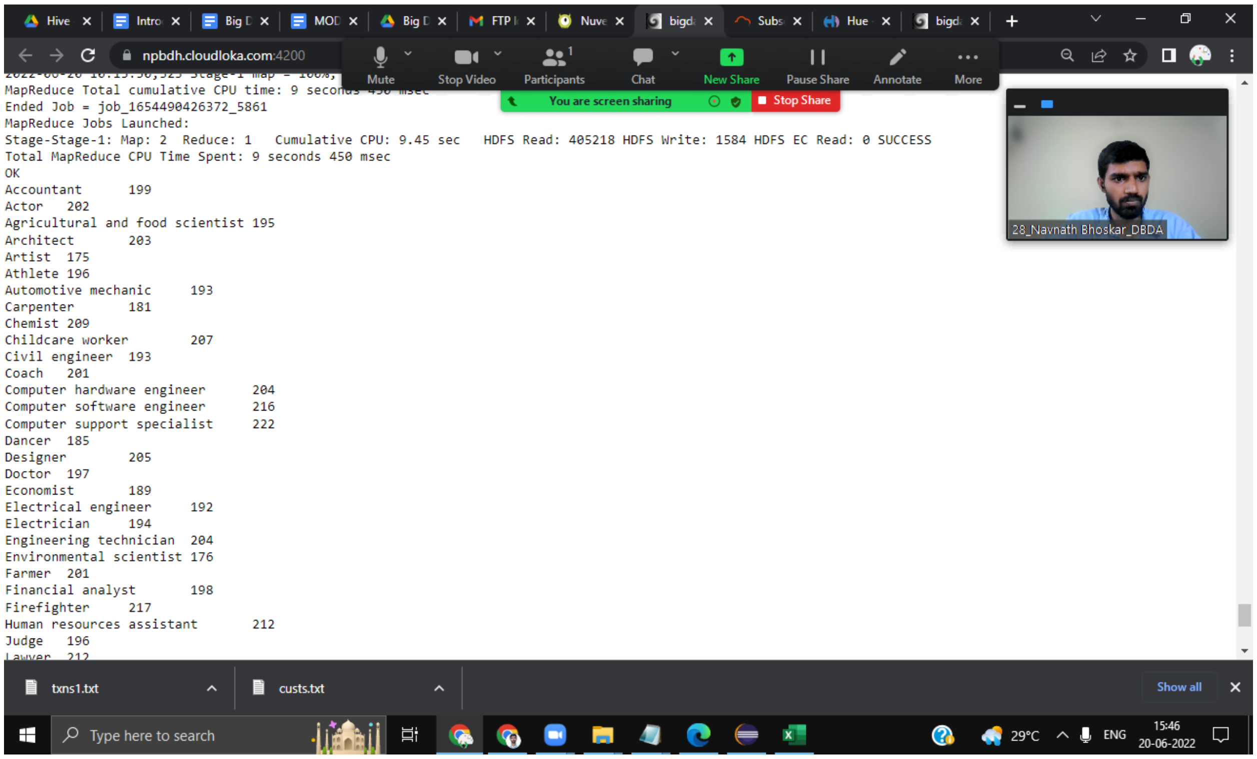
Could not connect to ip-10-1-1-204.ap-south-1.compute.internal:10000

AnalysisException: Un
8.nyse.stk_date'.

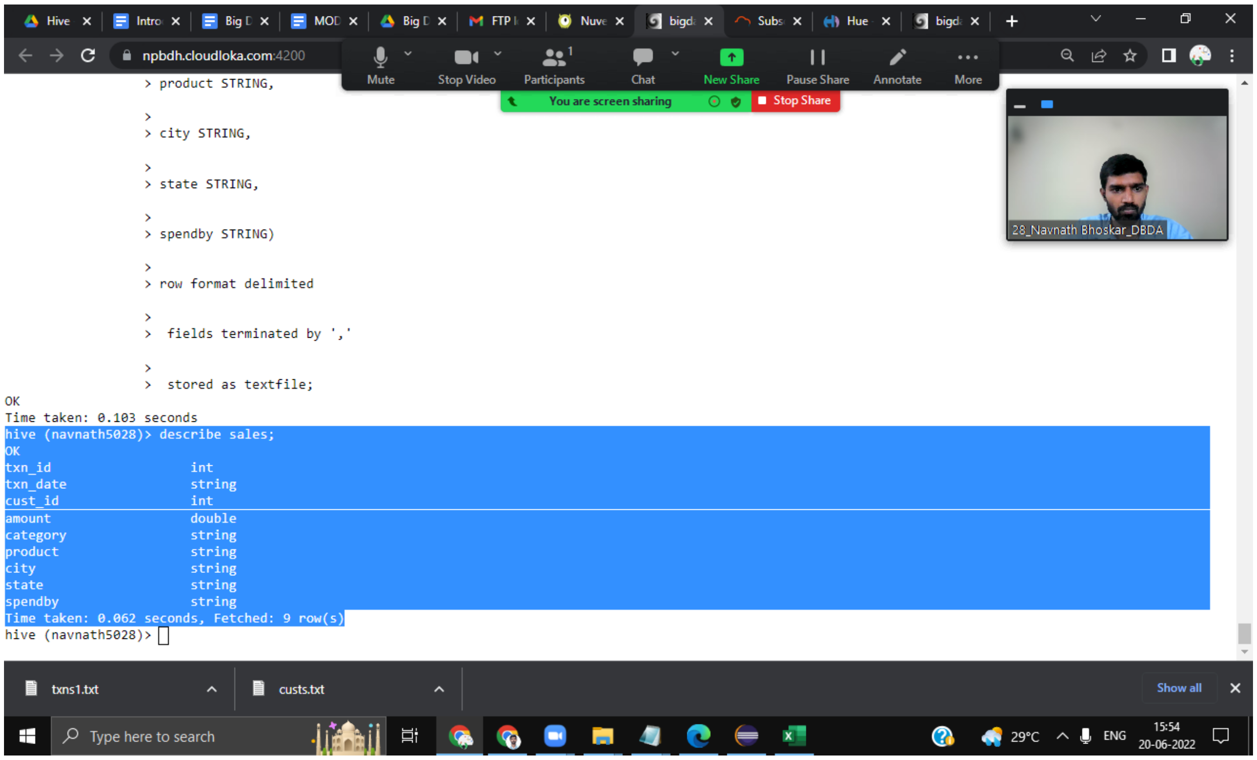
```
NYSE,AEA,2010-02-08,4.42,4.42,4.21,4.24,205500,4.24
NYSE,AEA,2010-02-05,4.42,4.54,4.22,4.41,194300,4.41
NYSE,AEA,2010-02-04,4.55,4.69,4.39,4.42,233800,4.42
NYSE,AEA,2010-02-03,4.65,4.69,4.50,4.55,182100,4.55
NYSE,AEA,2010-02-02,4.74,5.00,4.62,4.66,222700,4.66
NYSE,AEA,2010-02-01,4.84,4.92,4.68,4.75,194800,4.75
NYSE,AEA,2010-01-29,4.97,5.05,4.76,4.83,222900,4.83
NYSE,AEA,2010-01-28,5.12,5.22,4.81,4.98,283100,4.98
NYSE,AEA,2010-01-27,4.82,5.16,4.79,5.09,243500,5.09
NYSE,AEA,2010-01-26,5.18,5.18,4.81,4.84,554800,4.84
NYSE,AEA,2010-01-25,5.42,5.48,5.20,5.22,257300,5.22
NYSE,AEA,2010-01-22,5.52,5.59,5.31,5.37,260800,5.37
NYSE,AEA,2010-01-21,5.67,5.74,5.37,5.51,264300,5.51
NYSE,AEA,2010-01-20,5.65,5.70,5.53,5.66,244600,5.66
NYSE,AEA,2010-01-19,5.54,5.70,5.54,5.69,368000,5.69
NYSE,AEA,2010-01-15,5.48,5.55,5.33,5.54,435500,5.54
NYSE,AEA,2010-01-14,5.41,5.50,5.39,5.41,272200,5.41
NYSE,AEA,2010-01-13,5.50,5.50,5.41,5.45,176400,5.45
NYSE,AEA,2010-01-12,5.47,5.51,5.41,5.46,233100,5.46
NYSE,AEA,2010-01-11,5.64,5.64,5.49,5.55,178900,5.55
```



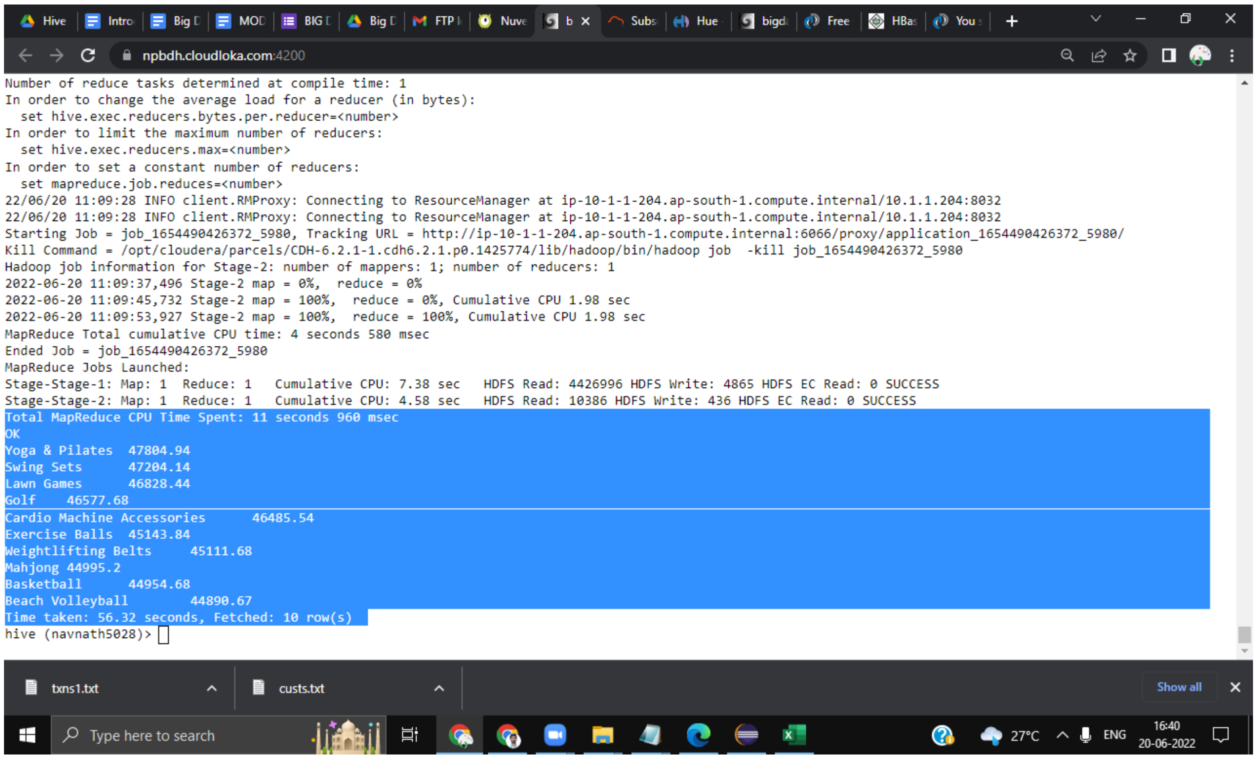
```
> txn_date STRING,

>
> cust_id INT,

>
> amount DOUBLE,

>
> category STRING,

>
> product STRING,

>
> city STRING,

>
> state STRING,

>
> spendby STRING)

>
> row format delimited

>
> fields terminated by ','

>
> stored as textfile;
OK
Time taken: 0.165 seconds
hive (navnath5028)>
```



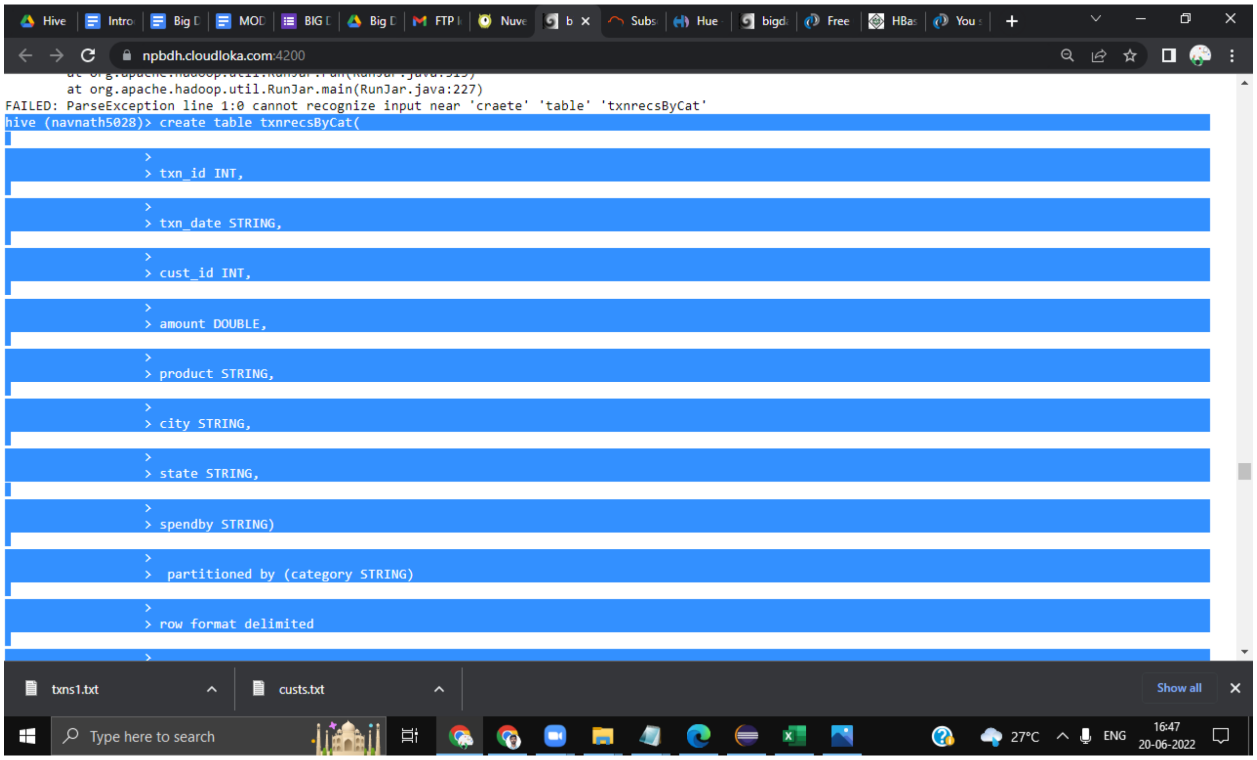
```
MapReduce Total cumulative CPU time: 9 seconds 450 msec
Ended Job = job_1654490426372_5861
MapReduce Jobs Launched:
Stage-Stage-1: Map: 2  Reduce: 1   Cumulative CPU: 9.45 sec   HDFS Read: 405218 HDFS Write: 1584 HDFS EC Read: 0 SUCCESS
Total MapReduce CPU Time Spent: 9 seconds 450 msec
OK
Accountant      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
Computer support specialist     222
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
```



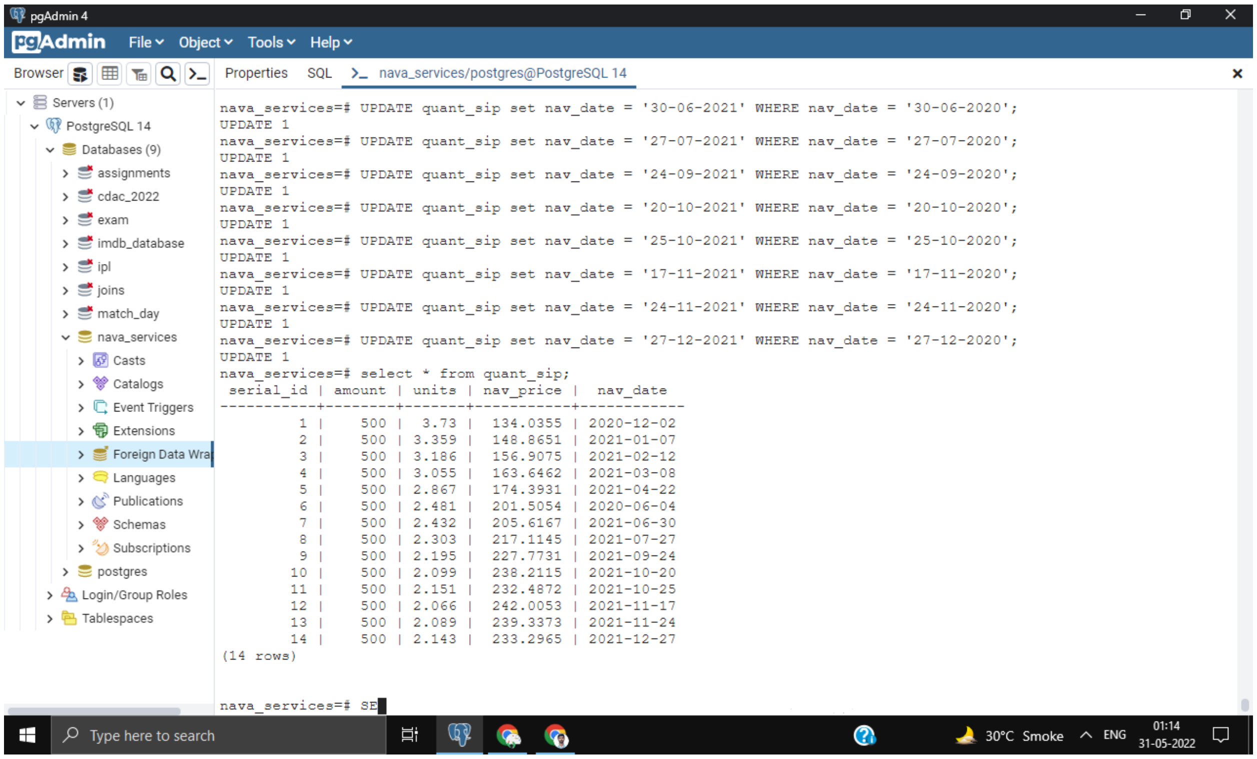
```
                  > product STRING,

                  >
                  > city STRING,

                  >
                  > state STRING,

                  >
                  > spendby STRING)

                  >
                  > row format delimited

                  >
                  >  fields terminated by ','

                  >
                  >  stored as textfile;
OK
Time taken: 0.103 seconds
hive (navnath5028)> describe sales;
OK
txn_id                  int
txn_date                string
cust_id                 int
amount                  double
category                string
product                 string
city                    string
state                   string
spendby                 string
Time taken: 0.062 seconds, Fetched: 9 row(s)
hive (navnath5028)>
```



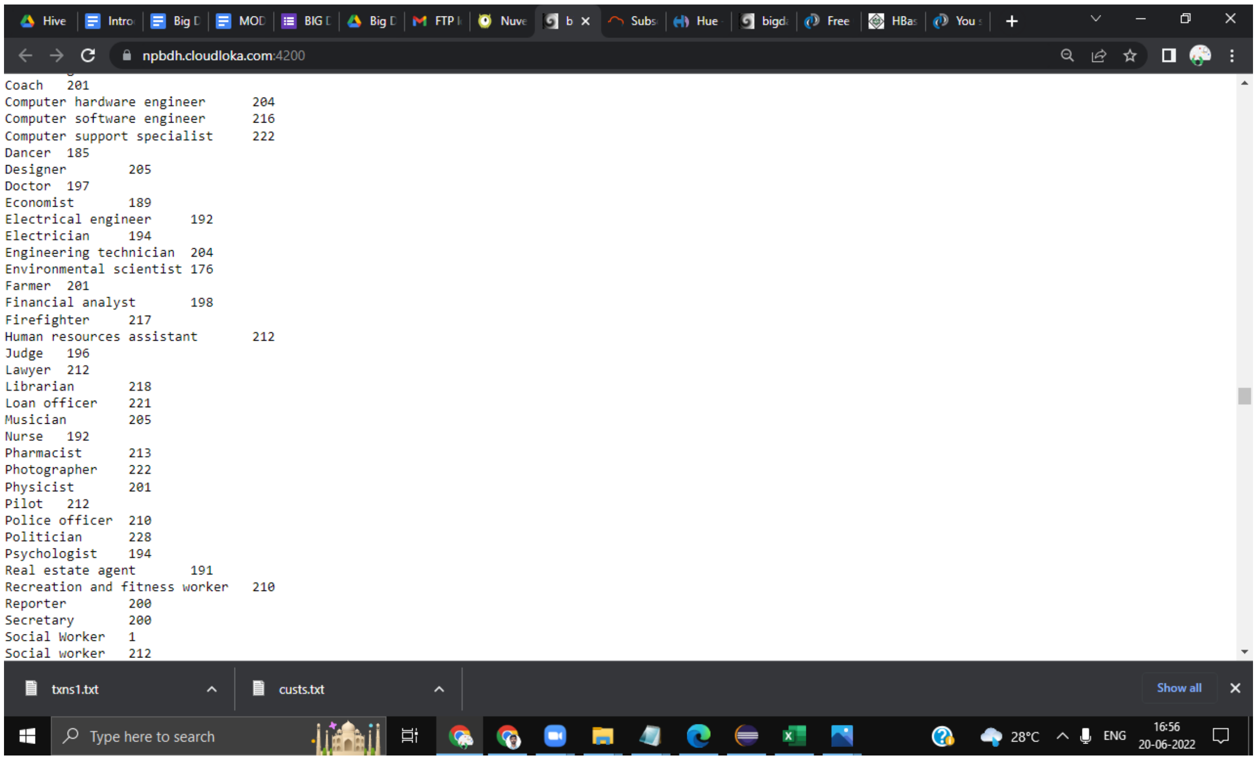
```
Number of reduce tasks determined at compile time: 1
In order to change the average load for a reducer (in bytes):
  set hive.exec.reducers.bytes.per.reducer=<number>
In order to limit the maximum number of reducers:
  set hive.exec.reducers.max=<number>
In order to set a constant number of reducers:
  set mapreduce.job.reduces=<number>
22/06/20 11:09:28 INFO client.RMProxy: Connecting to ResourceManager at ip-10-1-1-204.ap-south-1.compute.internal/10.1.1.204:8032
22/06/20 11:09:28 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_1654490426372_5980, Tracking URL = http://ip-10-1-1-204.ap-south-1.compute.internal:6066/proxy/application_1654490426372_5980/
Kill Command = /opt/cloudera/parcels/CDH-6.2.1-1.cdh6.2.1.p0.1425774/lib/hadoop/bin/hadoop job  -kill job_1654490426372_5980
Hadoop job information for Stage-2: number of mappers: 1; number of reducers: 1
2022-06-20 11:09:37,496 Stage-2 map = 0%,  reduce = 0%
2022-06-20 11:09:45,732 Stage-2 map = 100%,  reduce = 0%, Cumulative CPU 1.98 sec
2022-06-20 11:09:53,927 Stage-2 map = 100%,  reduce = 100%, Cumulative CPU 1.98 sec
MapReduce Total cumulative CPU time: 4 seconds 580 msec
Ended Job = job_1654490426372_5980
MapReduce Jobs Launched:
Stage-Stage-1: Map: 1  Reduce: 1   Cumulative CPU: 7.38 sec   HDFS Read: 4426996 HDFS Write: 4865 HDFS EC Read: 0 SUCCESS
Stage-Stage-2: Map: 1  Reduce: 1   Cumulative CPU: 4.58 sec   HDFS Read: 10386 HDFS Write: 436 HDFS EC Read: 0 SUCCESS
Total MapReduce CPU Time Spent: 11 seconds 960 msec
OK
Yoga & Pilates  47804.94
Swing Sets      47204.14
Lawn Games      46828.44
Golf    46577.68
Cardio Machine Accessories      46485.54
Exercise Balls  45143.84
Weightlifting Belts     45111.68
Mahjong 44995.2
Basketball      44954.68
Beach Volleyball        44890.67
Time taken: 56.32 seconds, Fetched: 10 row(s)
hive (navnath5028)>
```



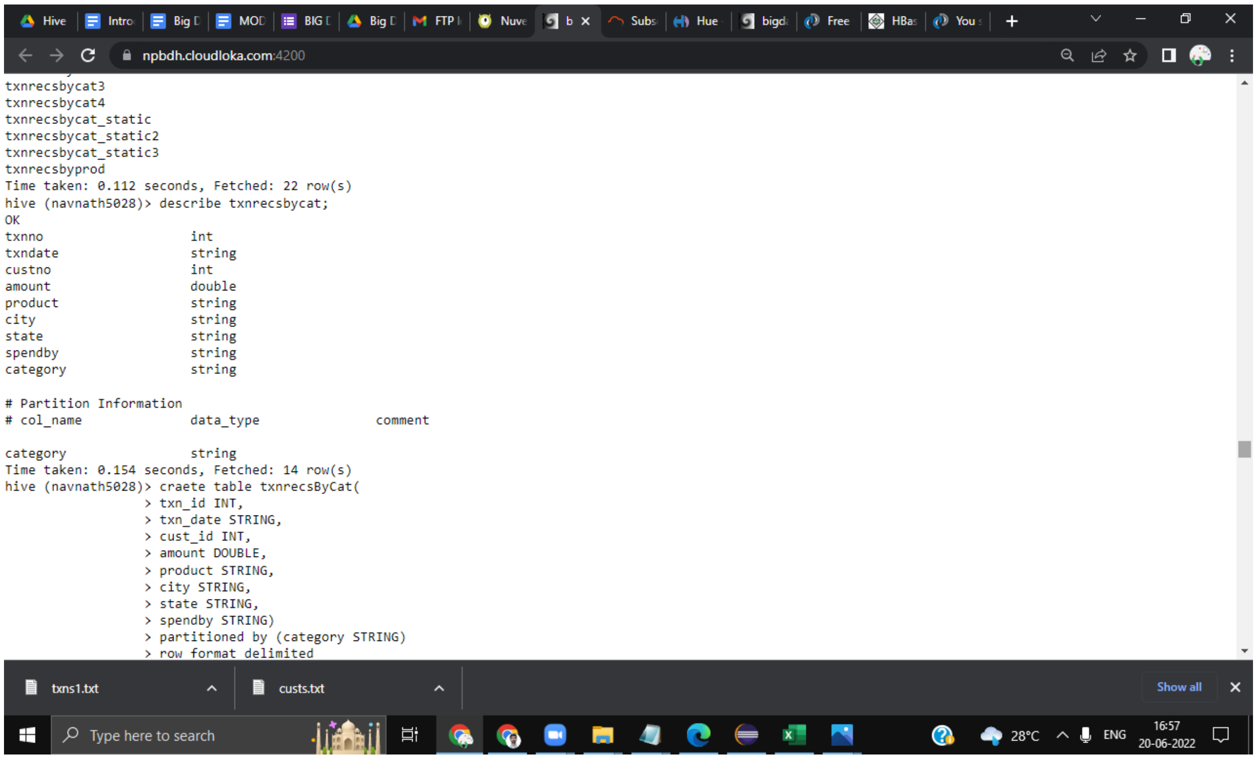
```
        at org.apache.hadoop.util.RunJar.main(RunJar.java:227)
FAILED: ParseException line 1:0 cannot recognize input near 'craete' 'table' 'txnrecsByCat'
hive (navnath5028)> create table txnrecsByCat(
                  >
                  > txn_id INT,
                  >
                  > txn_date STRING,
                  >
                  > cust_id INT,
                  >
                  > amount DOUBLE,
                  >
                  > product STRING,
                  >
                  > city STRING,
                  >
                  > state STRING,
                  >
                  > spendby STRING)
                  >
                  >  partitioned by (category STRING)
                  >
                  > row format delimited
                  >
```



```
nava_services=# UPDATE quant_sip set nav_date = '30-06-2021' WHERE nav_date = '30-06-2020';
UPDATE 1
nava_services=# UPDATE quant_sip set nav_date = '27-07-2021' WHERE nav_date = '27-07-2020';
UPDATE 1
nava_services=# UPDATE quant_sip set nav_date = '24-09-2021' WHERE nav_date = '24-09-2020';
UPDATE 1
nava_services=# UPDATE quant_sip set nav_date = '20-10-2021' WHERE nav_date = '20-10-2020';
UPDATE 1
nava_services=# UPDATE quant_sip set nav_date = '25-10-2021' WHERE nav_date = '25-10-2020';
UPDATE 1
nava_services=# UPDATE quant_sip set nav_date = '17-11-2021' WHERE nav_date = '17-11-2020';
UPDATE 1
nava_services=# UPDATE quant_sip set nav_date = '24-11-2021' WHERE nav_date = '24-11-2020';
UPDATE 1
nava_services=# UPDATE quant_sip set nav_date = '27-12-2021' WHERE nav_date = '27-12-2020';
UPDATE 1
nava_services=# select * from quant_sip;
 serial_id | amount | units | nav_price |  nav_date
-----------+--------+-------+-----------+------------
         1 |    500 |  3.73 |  134.0355 | 2020-12-02
         2 |    500 | 3.359 |  148.8651 | 2021-01-07
         3 |    500 | 3.186 |  156.9075 | 2021-02-12
         4 |    500 | 3.055 |  163.6462 | 2021-03-08
         5 |    500 | 2.867 |  174.3931 | 2021-04-22
         6 |    500 | 2.481 |  201.5054 | 2020-06-04
         7 |    500 | 2.432 |  205.6167 | 2021-06-30
         8 |    500 | 2.303 |  217.1145 | 2021-07-27
         9 |    500 | 2.195 |  227.7731 | 2021-09-24
        10 |    500 | 2.099 |  238.2115 | 2021-10-20
        11 |    500 | 2.151 |  232.4872 | 2021-10-25
        12 |    500 | 2.066 |  242.0053 | 2021-11-17
        13 |    500 | 2.089 |  239.3373 | 2021-11-24
        14 |    500 | 2.143 |  233.2965 | 2021-12-27
(14 rows)


nava_services=# SE
```



```
Coach   201
Computer hardware engineer      204
Computer software engineer      216
Computer support specialist     222
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   212
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
```



```
txnrecsbycat3
txnrecsbycat4
txnrecsbycat_static
txnrecsbycat_static2
txnrecsbycat_static3
txnrecsbyprod
Time taken: 0.112 seconds, Fetched: 22 row(s)
hive (navnath5028)> describe txnrecsbycat;
OK
txnno                   int
txndate                 string
custno                  int
amount                  double
product                 string
city                    string
state                   string
spendby                 string
category                string

# Partition Information
# col_name              data_type               comment

category                string
Time taken: 0.154 seconds, Fetched: 14 row(s)
hive (navnath5028)> craete table txnrecsByCat(
                  > txn_id INT,
                  > txn_date STRING,
                  > cust_id INT,
                  > amount DOUBLE,
                  > product STRING,
                  > city STRING,
                  > state STRING,
                  > spendby STRING)
                  > partitioned by (category STRING)
                  > row format delimited
```