**Question 1 : - MapReduce**

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);

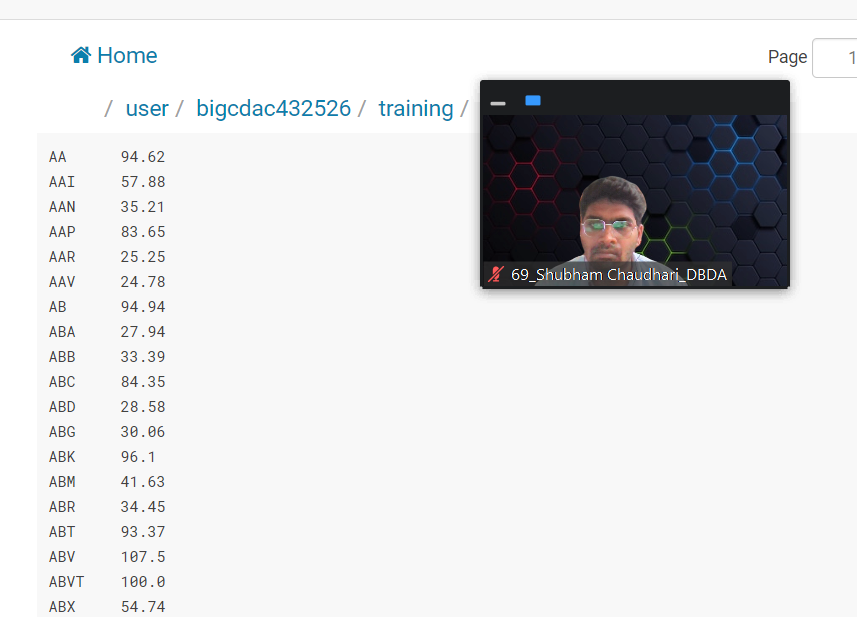
FileInputFormat.addInputPath(job, new Path(args[0]));

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

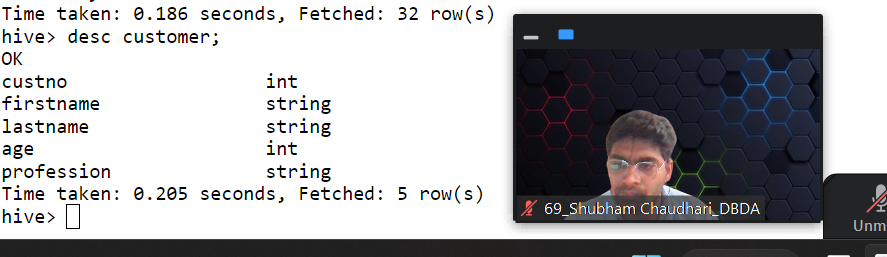
System.exit(job.waitForCompletion(true) ? 0 : 1);

}

}



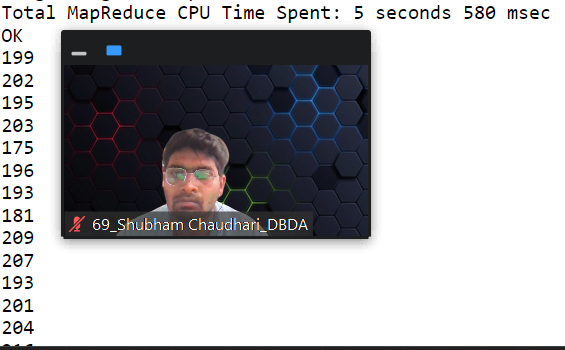
**Question 2**

****

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

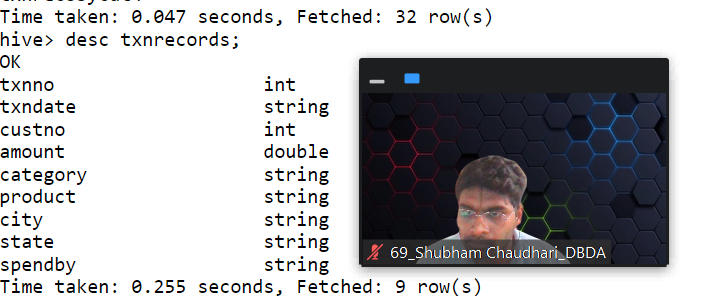
hive> create table customer(custno int, firstname string, lastname string, age int, profession string) row format delimited fields terminated by ‘ , ‘ stored as textfile;

Select count(custno) from customer group by profession;

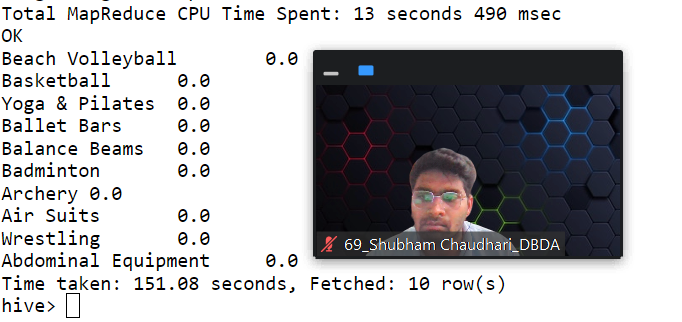


**2) Write a program to find the top 10 products sales wise**

hive> create table txnrecords(txnno int, txndate string, custno int, amount double, category string, product string, state string, spendby string) row format delimited fields terminited by ‘,’ stored as textfile;

****

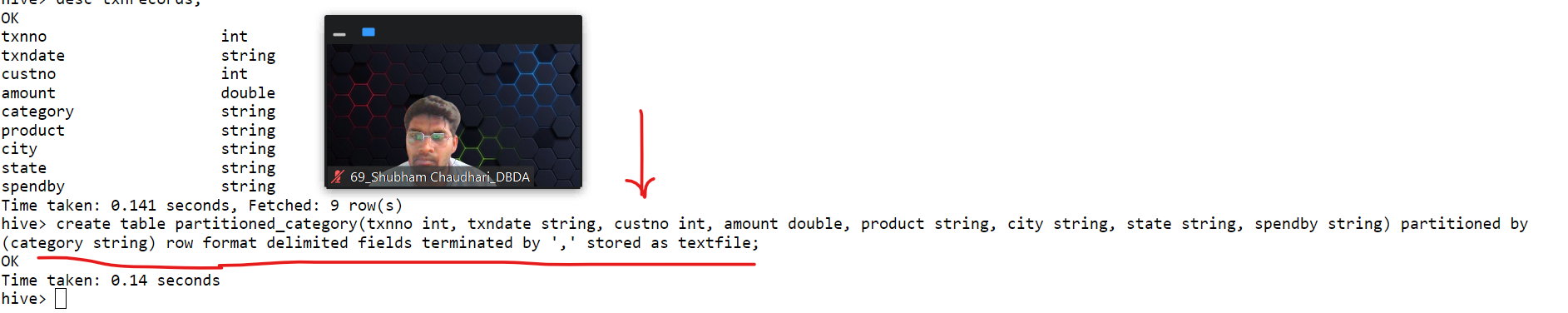
hive> select product, sum(spendby) as total from txnrecords group by product order by total desc limit 10;



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

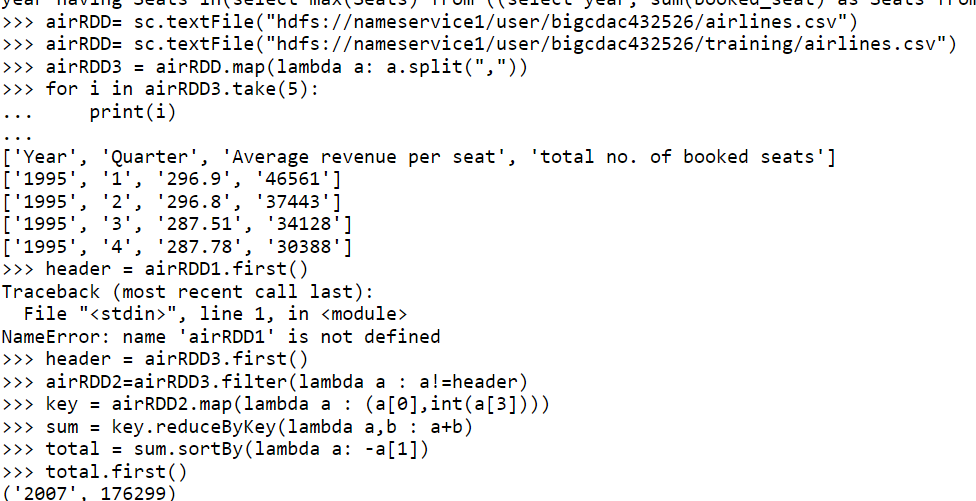
hive> create table partitioned\_category(txnno int, txndate string, custno int, amount double, product string, city string, state string, spendby string) partitioned by

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

****

**Question 3: - Pyspark**

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



**2) Identifying the highest revenue generation for which year**

keyvalue = airRDD2.map(lambda a : (a[0], float(a[2])\*int(a[3])))

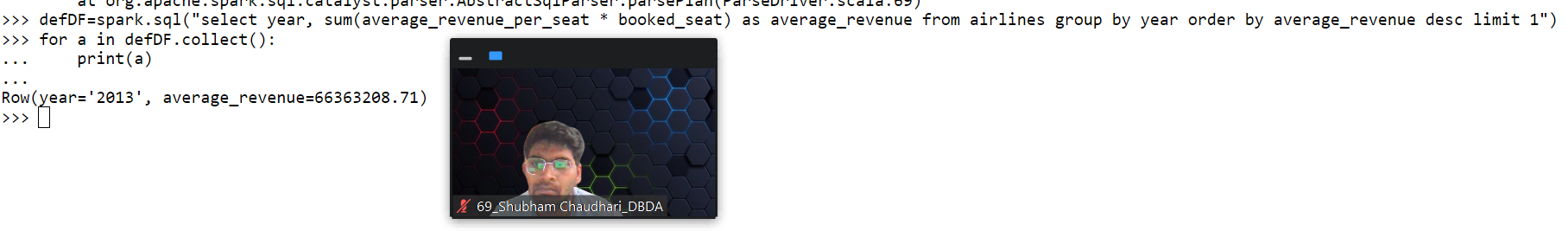
>>> addtotal = keyvalue.reduceByKey(lambda a : -a[1])

>>> addtotal = keyvalue.reduceByKey(lambda a,b : a+b)

>>> sortvalue = addtotal.sortBy(lambda a: -a[1])

>>> sortvalue.first()

**Output** -----> Row(year='2013', average\_revenue=66363208.71)



**3) Identifying the highest revenue generation for which year and quarter (Commongroup)**

>>> key = airRD2.map(lambda a : (a[0]+" "+a[1],float(a[2])\*int(a[3])))

Traceback (most recent call last):

File "<stdin>", line 1, in <module>

NameError: name 'airRD2' is not defined

>>> key = airRDD2.map(lambda a : (a[0]+" "+a[1],float(a[2])\*int(a[3])))

>>> total = key.reduceByKey(lambda a,b : a+b)

>>> total2 = total.sortBy(lambda a: -a[1])

>>> total2.first()

('2014 4', 18819408.48**)**

**Output : -** Row(year='2014', quarter='4', Revenue=18819408.48)

