BIG DATA MODULE END PAPER

ROLL NO:- 220340325007

Q1.

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>

       {

         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);

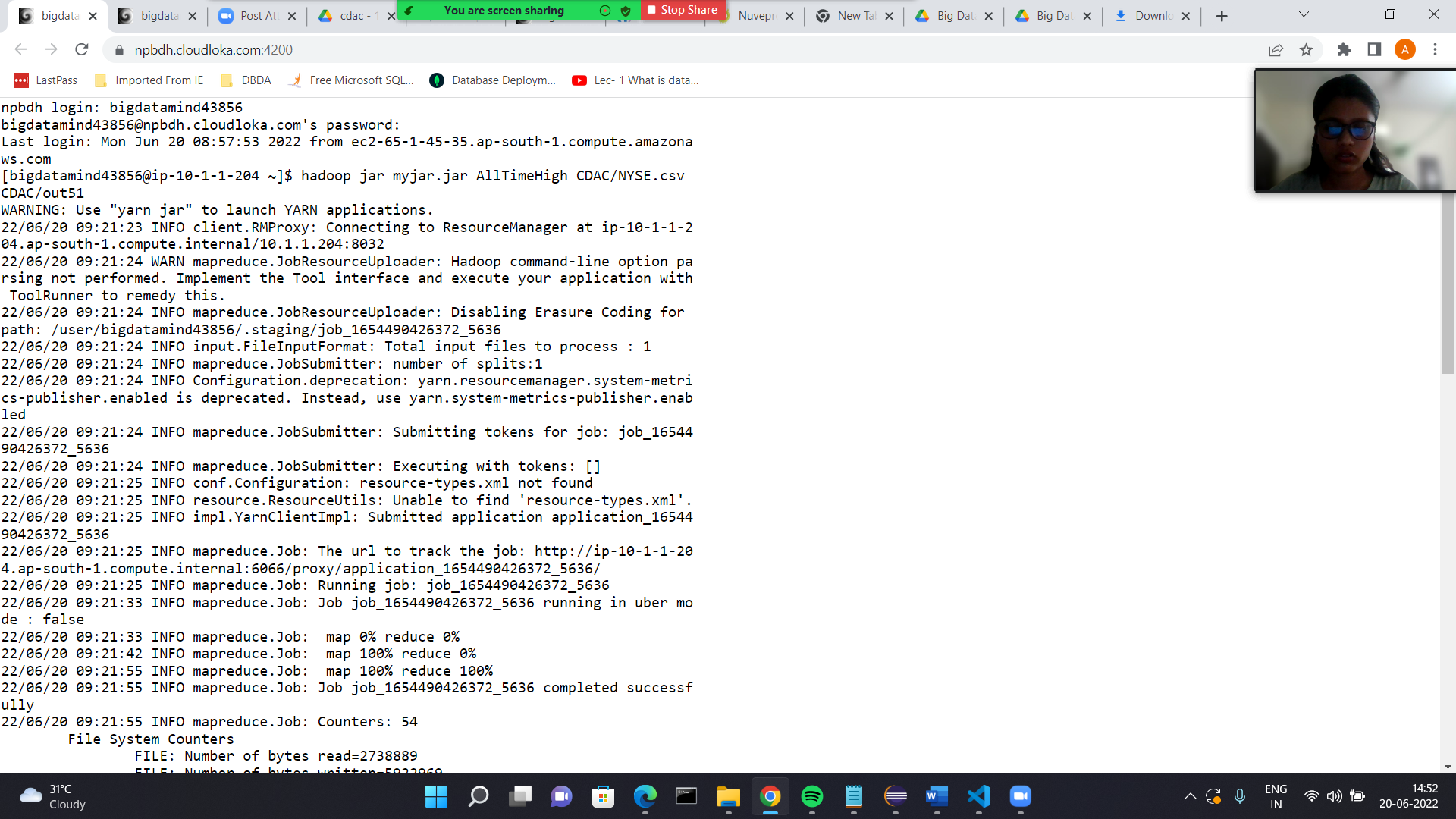
            FileInputFormat.addInputPath(job, new Path(args[0]));

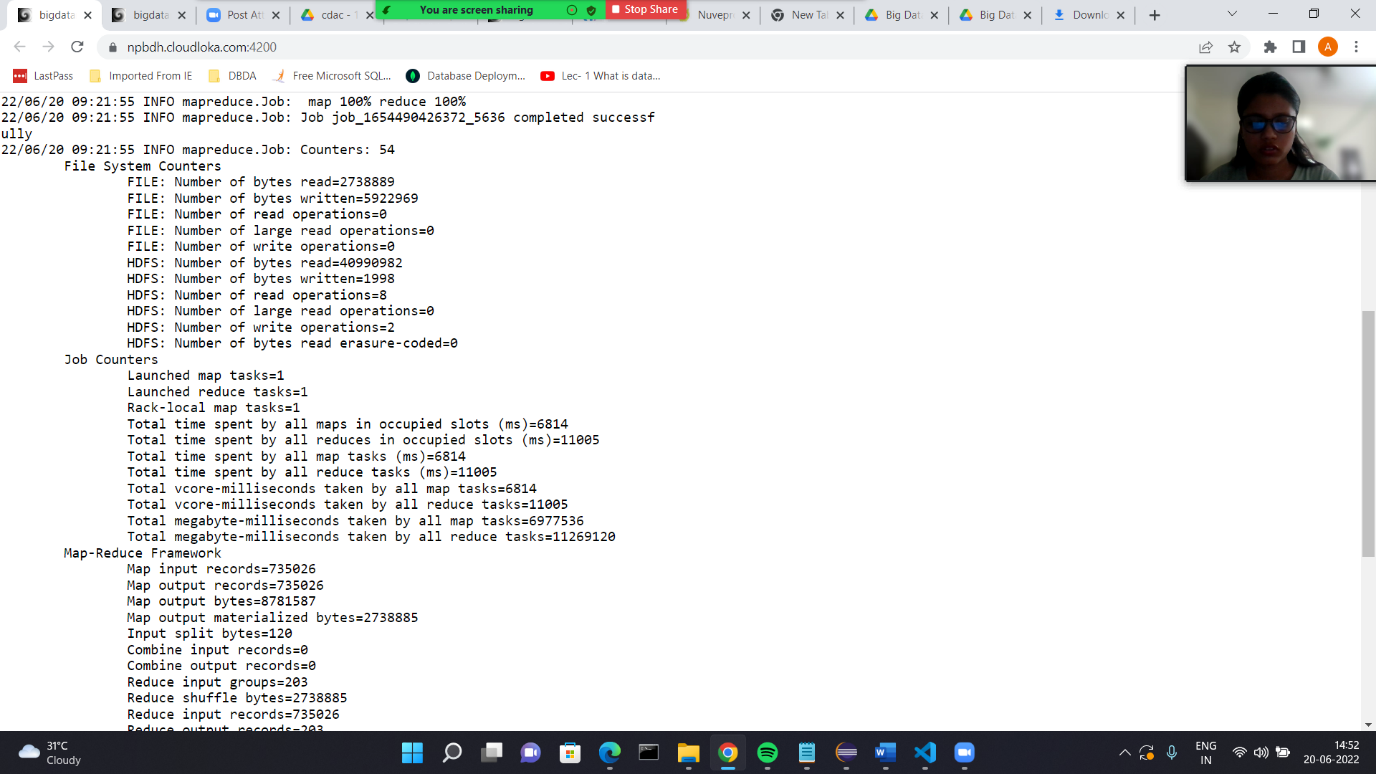
            FileOutputFormat.setOutputPath(job, new Path(args[1]));

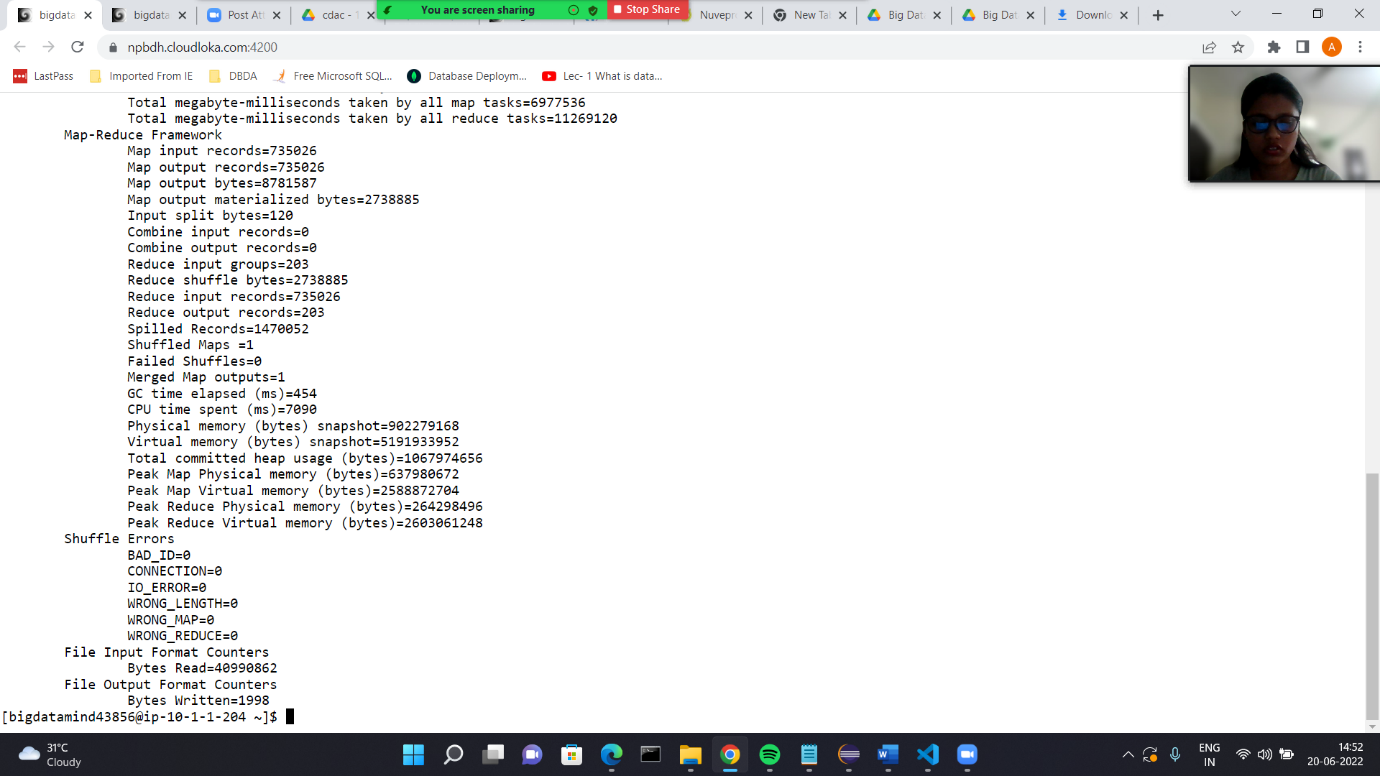
            System.exit(job.waitForCompletion(true) ? 0 : 1);

          }

}







Q2]

🡺

Hive

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

ANS:

hive> use anuradha3011;

hive> set hive.cli.print.current.db = true;

hive (anuradha3011)> create external table customer(custno string, firstname string, lastname string, age int,profession string)

row format delimited fields terminated by ',' stored as textfile

> ;

OK

Time taken: 0.426 seconds

hive (anuradha3011)> show tables;

OK

airlines

customer

+

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

hive (anuradha3011)> select profession,count(custno) from customer group by profession;

Query ID = bigdatamind43851\_20220620091132\_ae6be093-8b27-4b5c-9196-e2a12e306276

Total jobs = 1

35

Please find the sales data set. txn id txn date cust id amount category product city state spendb

hive (anuradha3011)> create table txnrecords(txnno INT, txndate STRING, custno INT, amount DOUBLE, category STRING, product STR

ING, city STRING, state STRING, spendby STRING) row format delimited fields terminated by ',' stored as textfile

> ;

OK

Time taken: 0.094 seconds

hive (anuradha3011)> show tables;

OK

airlines

customer

txnrecords

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

hive (anuradha3011)> select product, sum(amount)as total from txnrecords group by product order by total desc limit 10;

3) Write a program to create partiioned table on category

hive (anuradha3011)> 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 textfi

le;

OK

Time taken: 0.088 seconds

hive (anuradha3011)> create table txnrecsByCat2(txnno INT, txndate STRING, custno INT, amount DOUBLE, product STRING, city STRIN

G, state STRING, spendby STRING) partitioned by (category STRING) clustered by (state) into 10 buckets row format delimited fie

lds terminated by ',' stored as textfile;

OK

Time taken: 0.115 seconds

hive (anuradha3011)> create table txnrecsByCat4(txnno INT, txndate STRING, custno INT, amount DOUBLE, category String, product S

TRING, city STRING, state STRING, spendby STRING) partitioned by (month STRING) row format delimited fields terminated by ',' s

tored as textfile;

OK

Time taken: 0.088 seconds

hive (anuradha3011)> create table txnrecsByCat3(txnno INT, txndate STRING, custno INT, amount DOUBLE, product STRING, city STRIN

G, state STRING) partitioned by (category STRING,spendby STRING) row format delimited fields terminated by ',' stored as textfi

le;

OK

Time taken: 0.166 seconds

hive (anuradha3011)> set hive.exec.dynamic.partition.mode=nonstrict;

hive (anuradha3011)> set hive.exec.dynamic.partition=true;

hive (anuradha3011)> show tables;

OK

airlines

customer

txnrecords

txnrecsbycat

txnrecsbycat2

txnrecsbycat3

txnrecsbycat4

Time taken: 0.04 seconds, Fetched: 7 row(s)

Q3]

🡺

airlineRDD = sc.textFile("/user/bigdatamind43856/Spark/airlines.csv")

airlineRDD1 = airlineRDD.map(lambda a : a.encode("ascii", "ignore"))

header = airlineRDD1.first()

airlineRDD2 = airlineRDD1.filter(lambda a : a != header)

arrayRDD = airlineRDD2.map(lambda a : a.split(","))

from pyspark.sql.types import StructType, IntegerType, DoubleType, LongType, StringType

schema2 = StructType().add("Year",StringType(),True).add("qtr",IntegerType(),True).add("revenue",DoubleType(),True).add("seats",LongType(),True)

airlinesDF = spark.read.format("csv").option("header","true").schema(schema2).load("/user/bigdatamind43856/Spark/airlines.csv")