**BDA Lab Record**

**1BM17CS081 Romil K Balar**

**Program 1**

**Question:**

Perform the following DB operations using MongoDB.

1. Create a database “Student” with the following attributes Rollno, Age, ContactNo, Email-Id.

2. Insert appropriate values

3. Write query to update Email-Id of a student with rollno 10.

4. Replace the student name from “ABC” to “FEM” of rollno 11.

5. Export the created table into local file system

6. Drop the table

7. Import a given csv dataset from local file system into mongodb collection.

**Commands:**

show dbs

use local

db.createCollection('Student')

db.Student.insert({rollno: 1, name:"Romil", age:20, pno:1234, email:"a@a.com"})

db.Student.insert({rollno: 5, name:"BBB", age:21, pno:2345, email:"b@b.com"})

db.Student.insert({rollno: 10, name:"CCC", age:19, pno:3456, email:"c@c.com"})

db.Student.insert({rollno: 11, name:"ABC", age:22, pno:4567, email:"d@d.com"})

db.Student.update({rollno:10},{$set:{email:"e@e.com"}})

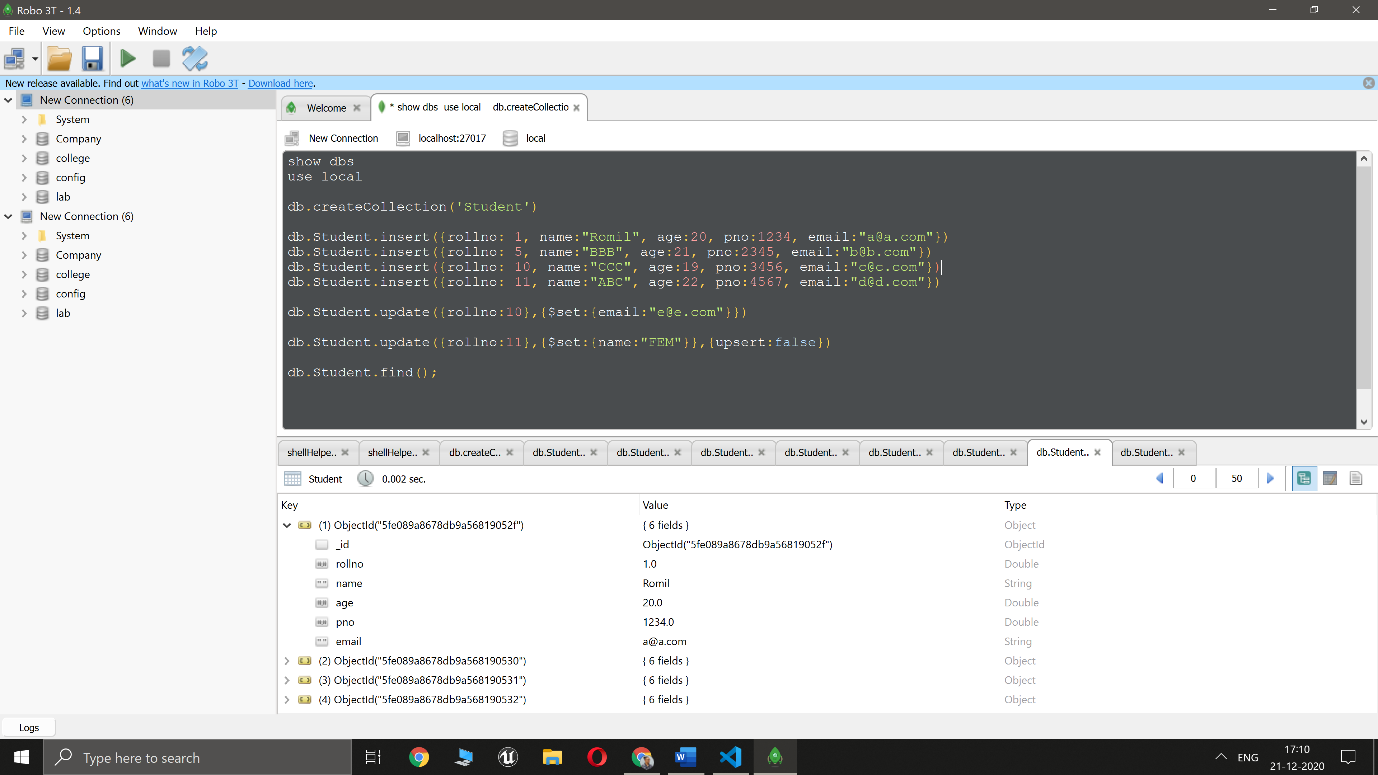
db.Student.update({rollno:11},{$set:{name:"FEM"}},{upsert:false})

db.Student.find();

mongoexport --collection=Student --db=local --out=temp.csv

db.Student.drop()

**Output:**



**Program 2**

**Question:**

Perform the following DB operations using MongoDB.

1. Create a collection by name Customers with the following attributes.

Cust\_id, Acc\_Bal, Acc\_Type

2. Insert at least 5 values into the table

3. Write a query to display those records whose total account balance is greater than

1200 of account type ‘Z’ for each customer\_id.

4. Determine Minimum and Maximum account balance for each customer\_id.

5. Export the created collection into local file system

6. Drop the table

7. Import a given csv dataset from local file system into mongodb collection.

**Commands:**

show dbs

use local

db.createCollection('Customers')

db.Customers.insert({cId: 1, cur\_acc\_bal:1500, acc\_bal:[100,1400,1600,1500] ,acc\_type:"Z"})

db.Customers.insert({cId: 2, cur\_acc\_bal:500, acc\_bal:[1000,1200,1500,500] ,acc\_type:"Y"})

db.Customers.insert({cId: 3, cur\_acc\_bal:1000, acc\_bal:[400,500,600,1000] ,acc\_type:"Z"})

db.Customers.insert({cId: 4, cur\_acc\_bal:1300, acc\_bal:[700,1200,1000,1300] ,acc\_type:"Y"})

db.Customers.insert({cId: 5, cur\_acc\_bal:1400, acc\_bal:[400,200,1200,1400] ,acc\_type:"Z"})

db.Customers.find()

db.Customers.find({cur\_acc\_bal:{$gt:1200}, acc\_type:"Z"},{})

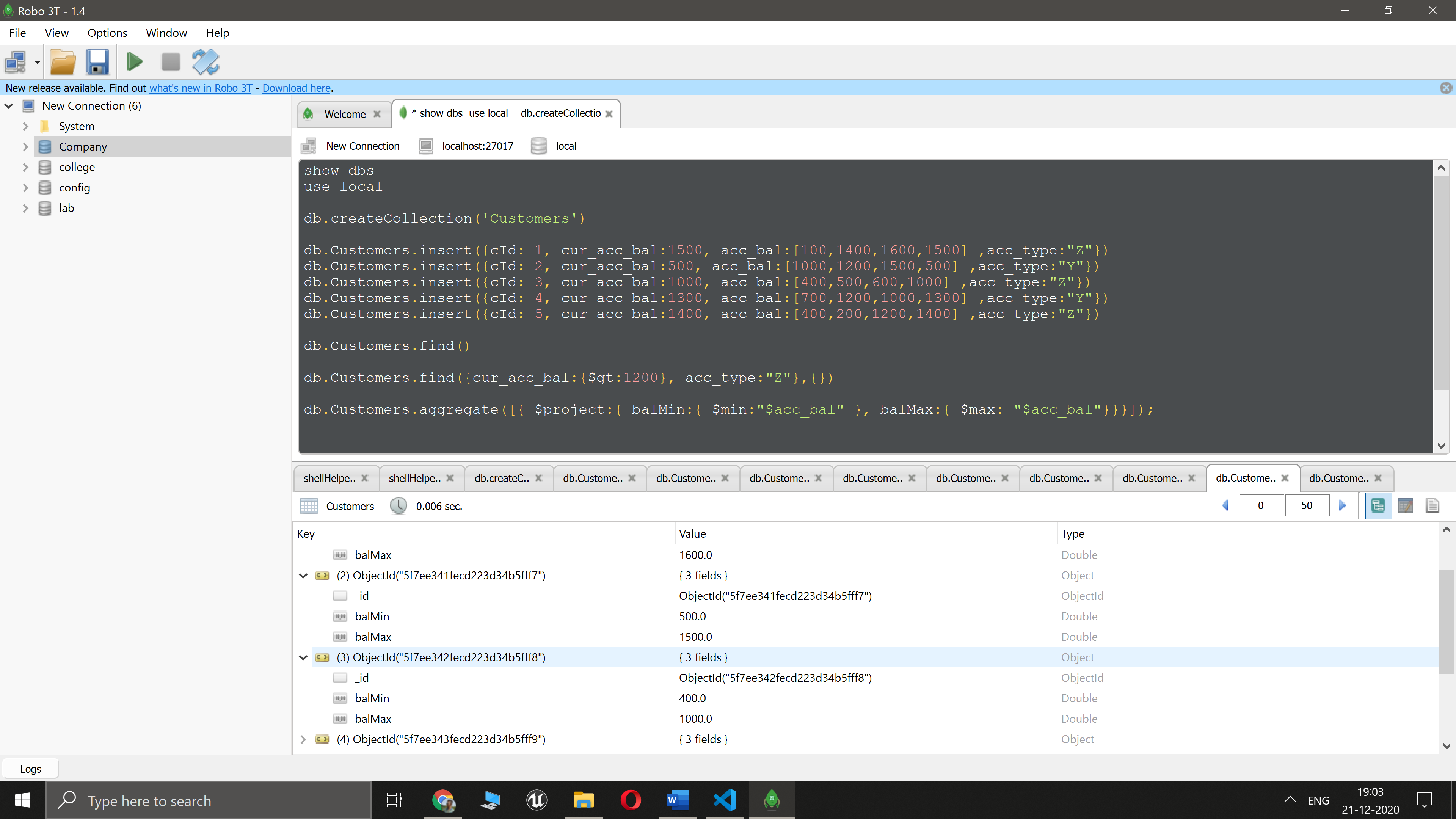
db.Customers.aggregate([{ $project:{ balMin:{ $min:"$acc\_bal" }, balMax:{ $max: "$acc\_bal"}}}]);

mongoexport --collection=Customers --db=local --out=custTemp.csv

mongoimport -d BankImported --collection CustomerImported --type csv --file bank\_data.csv --headerline

db.Customers.drop()

**Output:**



**Program 3**

**Question:**

Perform the following DB operations using Cassandra.

1. Create a keyspace by name Employee

2. Create a column family by name Employee-Info with attributes

Emp\_Id Primary Key, Emp\_Name, Designation, Date\_of\_Joining, Salary,

Dept\_Name

3. Insert the values into the table in batch

3. Update Employee name and Department of Emp-Id 121

4. Sort the details of Employee records based on salary

5. Alter the schema of the table Employee\_Info to add a column Projects which stores a set of Projects done by the corresponding Employee.

6. Update the altered table to add project names.

7.Create a TTL of 15 seconds to display the values of Employees.

**Commands:**

CREATE KEYSPACE Employee WITH replication =

{'class':'SimpleStrategy','replication\_factor':3};

CREATE COLUMNFAMILY employee\_info(emp\_id INT PRIMARY

KEY,emp\_name VARCHAR,desgination VARCHAR,doj

VARCHAR,dept\_name VARCHAR,salary INT);

BEGIN BATCH

INSERT INTO

employee\_info(emp\_id,dept\_name,desgination,doj,emp\_name,salary)val

ues(120,'Development','CTO','10/11/2015','Hritik',2000000);

INSERT INTO

employee\_info(emp\_id,dept\_name,desgination,doj,emp\_name,salary)val

ues(121,'HR','Employee','20/01/2011','Romil',1500000);

INSERT INTO

employee\_info(emp\_id,dept\_name,desgination,doj,emp\_name,salary)val

ues(122,'Maintainance','staff','10/07/2020','Sanjay',50000);

APPLY BATCH;

select \* from employee\_info;

UPDATE employee\_info SET emp\_name = 'Rohit',dept\_name = 'IT'

WHERE emp\_id = 121;

select \* from employee\_info;

ALTER TABLE employee\_info ADD Project VARCHAR;

UPDATE employee\_info SET project=’BDA Analytics' WHERE

emp\_id=121;

INSERT INTO employee\_info(emp\_id, dept\_name, desgination, doj,

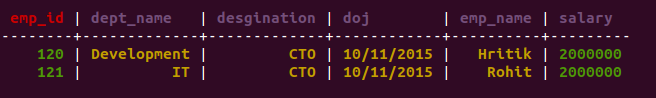
emp\_name,project,salary)values(124, 'PR','Senior

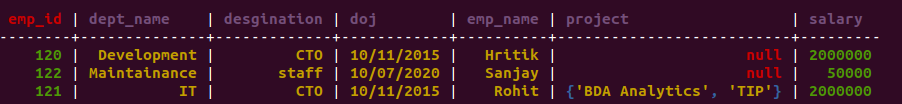
Manager','8/8/2020','Load balancing server','Abhi',20000) USING TTL 60;

SELECT TTL(desgination) FROM employee\_info where emp\_id=124;

**Output:**

**1.**



**2.**

**Program 4**

**Question:**

Perform the following DB operations using Cassandra.

1.Create a keyspace by name Library

2. Create a column family by name Library-Info with attributes

Stud\_Id Primary Key,

Counter\_value of type Counter,

Stud\_Name, Book-Name, Book-Id, Date\_of\_issue

3. Insert the values into the table in batch

3. Display the details of the table created and increase the value of the counter

4. Write a query to show that a student with id 112 has taken a book “BDA” 2 times.

5. Export the created column to a csv file

6. Import a given csv dataset from local file system into Cassandra column family

**Commands:**

CREATE COLUMNFAMILY libraryinfo(stud\_id uuid, counter\_value

counter, stud\_name VARCHAR, book\_name VARCHAR, book\_id INT, DOI

VARCHAR, PRIMARY KEY(stud\_id,stud\_name,book\_name,book\_id,doi));

UPDATE libraryinfo set counter\_value=counter\_value+1 where

stud\_id=cdd43759-bc64-4119-91bf-3e9ec59ca966 and stud\_name='Anjali'

and book\_name='malgudi' and book\_id=1 and doi='1/2/2020';

UPDATE libraryinfo set counter\_value=counter\_value+1 where

stud\_id=uuid() and stud\_name='Ayush' and book\_name='Hardy boys' and

book\_id=1 and doi='1/2/2020';

UPDATE libraryinfo set counter\_value=counter\_value+1 where

stud\_id=uuid() and stud\_name='meghna' and book\_name='BDA' and

book\_id=1 and doi='1/2/2020';

select \* from libraryinfo;

UPDATE libraryinfo set counter\_value=counter\_value+1 where

stud\_id=a56165fc-11c0-4e47-ba0b-e4ad8dd7b69d and

stud\_name='meghna' and book\_name='BDA' and book\_id=1 and

doi='1/2/2020';

select \* from libraryinfo;

SELECT \* from libraryinfo where counter\_value=2 ALLOW FILTERING;

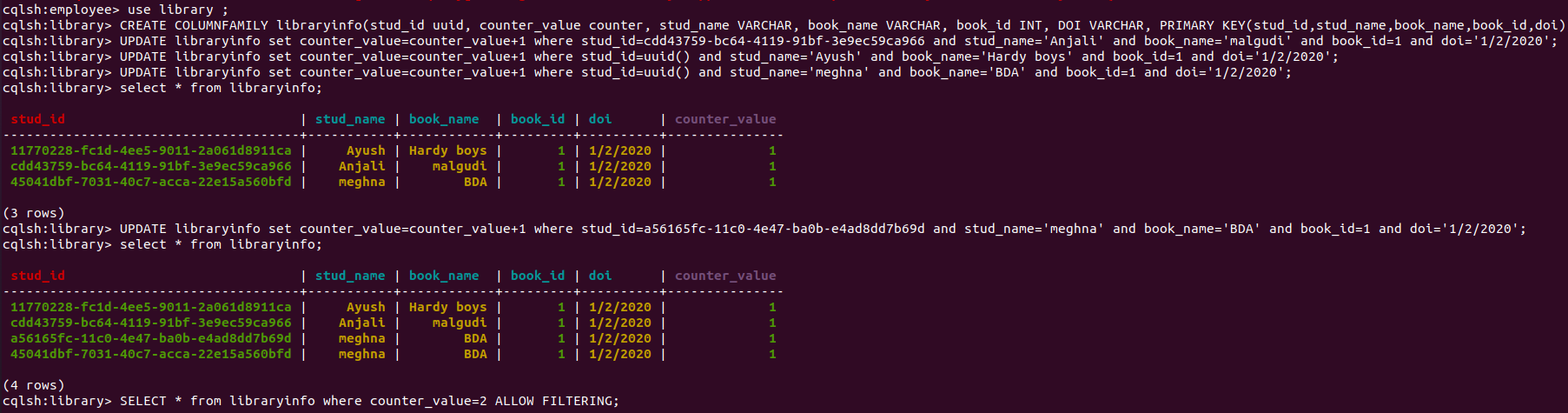
CAPTURE

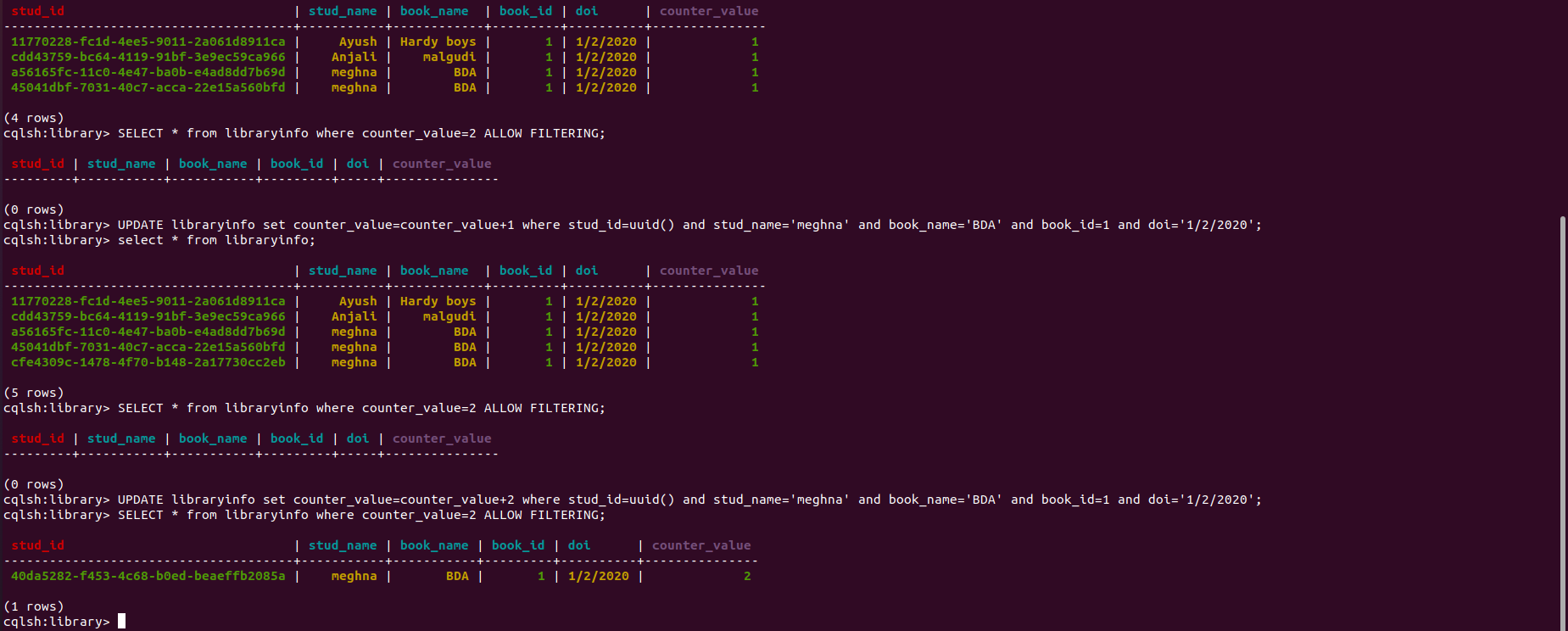
CAPTURE 'C:\Users\Romil Balar\Documents\BMSCE\BDA\user.csv';

select \* from libraryinfo;

**Output:**

**1.**



**2.** 

**Program 5**

**Question:**

Develop a MapReduce program to count the number of occurrences of words in a given file.

**Commands:**

WordCount\_Mapper.java

import java.io.IOException;

import java.util.StringTokenizer;

import org.apache.hadoop.io.IntWritable;

import org.apache.hadoop.io.LongWritable;

import org.apache.hadoop.io.Text;

import org.apache.hadoop.mapred.MapReduceBase;

import org.apache.hadoop.mapred.Mapper;

import org.apache.hadoop.mapred.OutputCollector;

import org.apache.hadoop.mapred.Reporter;

public class WC\_Mapper extends MapReduceBase implements Mapper<LongWritable,Text,Text,IntWritable>{

private final static IntWritable one = new IntWritable(1);

private Text word = new Text();

public void map(LongWritable key, Text value,OutputCollector<Text,IntWritable> output,

Reporter reporter) throws IOException{

String line = value.toString();

StringTokenizer tokenizer = new StringTokenizer(line);

while (tokenizer.hasMoreTokens()){

word.set(tokenizer.nextToken());

output.collect(word, one);

}

}

}

WordCount\_Reducer.java

import java.io.IOException;

import java.util.Iterator;

import org.apache.hadoop.io.IntWritable;

import org.apache.hadoop.io.Text;

import org.apache.hadoop.mapred.MapReduceBase;

import org.apache.hadoop.mapred.OutputCollector;

import org.apache.hadoop.mapred.Reducer;

import org.apache.hadoop.mapred.Reporter;

public class WC\_Reducer extends MapReduceBase implements Reducer<Text,IntWritable,Text,IntWritable> {

public void reduce(Text key, Iterator<IntWritable> values,OutputCollector<Text,IntWritable> output,

Reporter reporter) throws IOException {

int sum=0;

while (values.hasNext()) {

sum+=values.next().get();

}

output.collect(key,new IntWritable(sum));

}

}

WordCount\_Driver

import java.io.IOException;

import org.apache.hadoop.fs.Path;

import org.apache.hadoop.io.IntWritable;

import org.apache.hadoop.io.Text;

import org.apache.hadoop.mapred.FileInputFormat;

import org.apache.hadoop.mapred.FileOutputFormat;

import org.apache.hadoop.mapred.JobClient;

import org.apache.hadoop.mapred.JobConf;

import org.apache.hadoop.mapred.TextInputFormat;

import org.apache.hadoop.mapred.TextOutputFormat;

public class WC\_Runner {

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

JobConf conf = new JobConf(WC\_Runner.class);

conf.setJobName("WordCount");

conf.setOutputKeyClass(Text.class);

conf.setOutputValueClass(IntWritable.class);

conf.setMapperClass(WC\_Mapper.class);

conf.setCombinerClass(WC\_Reducer.class);

conf.setReducerClass(WC\_Reducer.class);

conf.setInputFormat(TextInputFormat.class);

conf.setOutputFormat(TextOutputFormat.class);

FileInputFormat.setInputPaths(conf,new Path(args[0]));

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

JobClient.runJob(conf);

}

}

**Output:**



**Program 6**

**Question:**

For the given file, Create a Map Reduce program to find the average temperature for each year from NCDC data set.

**Commands:**

Average\_Mapper.java

import org.apache.hadoop.mapreduce.\*;

import java.io.IOException;

public class AverageMapper extends Mapper <LongWritable, Text, Text, IntWritable>

{

public static final int MISSING = 9999;

public void map(LongWritable key, Text value, Context context) throws IOException, InterruptedException

{

String line = value.toString();

String year = line.substring(15,19);

int temperature;

if (line.charAt(87)=='+')

temperature = Integer.parseInt(line.substring(88, 92));

else

temperature = Integer.parseInt(line.substring(87, 92));

String quality = line.substring(92, 93);

if(temperature != MISSING && quality.matches("[01459]"))

context.write(new Text(year),new IntWritable(temperature));

}

}

Average\_reducer.java

import org.apache.hadoop.io.IntWritable;

import org.apache.hadoop.io.Text;

import org.apache.hadoop.mapreduce.\*;

import java.io.IOException;

public class AverageReducer extends Reducer <Text, IntWritable,Text, IntWritable >

{

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

{

int max\_temp = 0;

int count = 0;

for (IntWritable value : values)

{

max\_temp += value.get();

count+=1;

}

context.write(key, new IntWritable(max\_temp/count));

}

}

Average\_Driver.java

import org.apache.hadoop.io.\*;

import org.apache.hadoop.fs.\*;

import org.apache.hadoop.mapreduce.\*;

import org.apache.hadoop.mapreduce.lib.input.FileInputFormat;

import org.apache.hadoop.mapreduce.lib.output.FileOutputFormat;

public class AverageDriver

{

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

{

if (args.length != 2)

{

System.err.println("Please Enter the input and output parameters");

System.exit(-1);

}

Job job = new Job();

job.setJarByClass(AverageDriver.class);

job.setJobName("Max temperature");

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

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

job.setMapperClass(AverageMapper.class);

job.setReducerClass(AverageReducer.class);

job.setOutputKeyClass(Text.class);

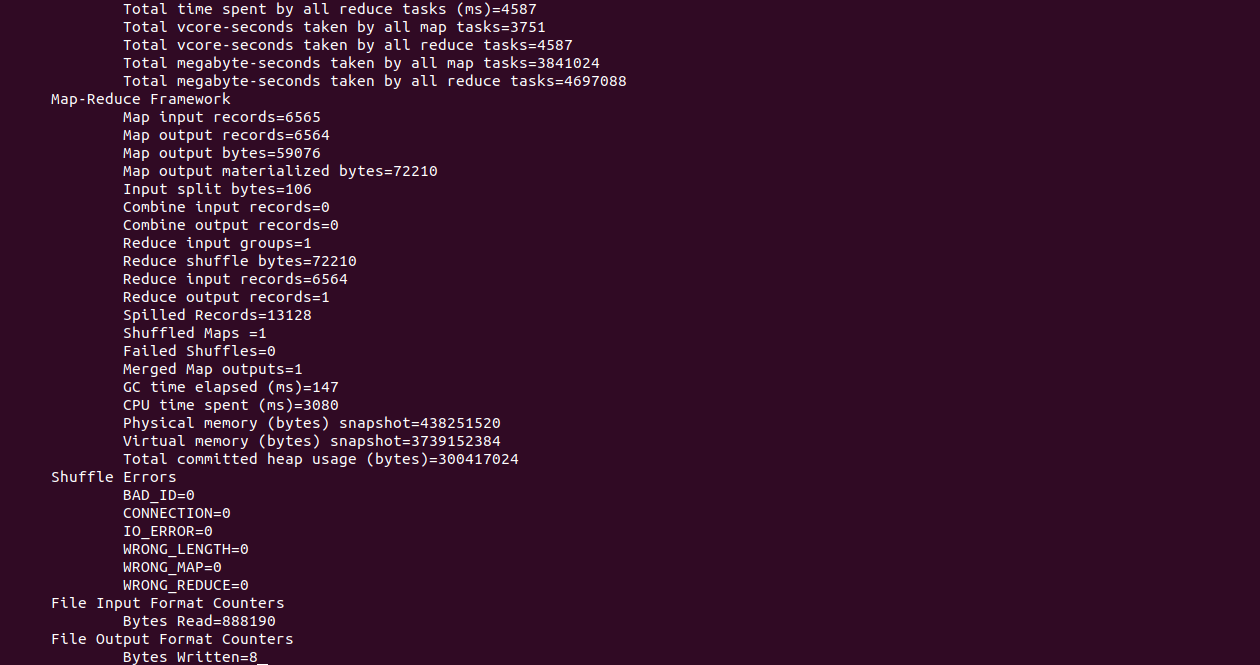
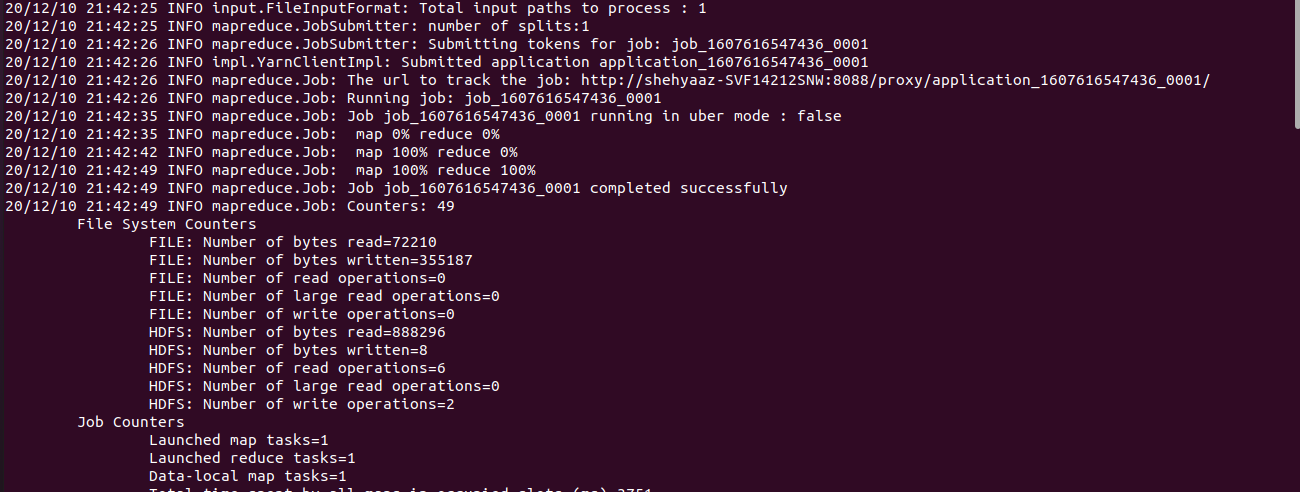
job.setOutputValueClass(IntWritable.class);

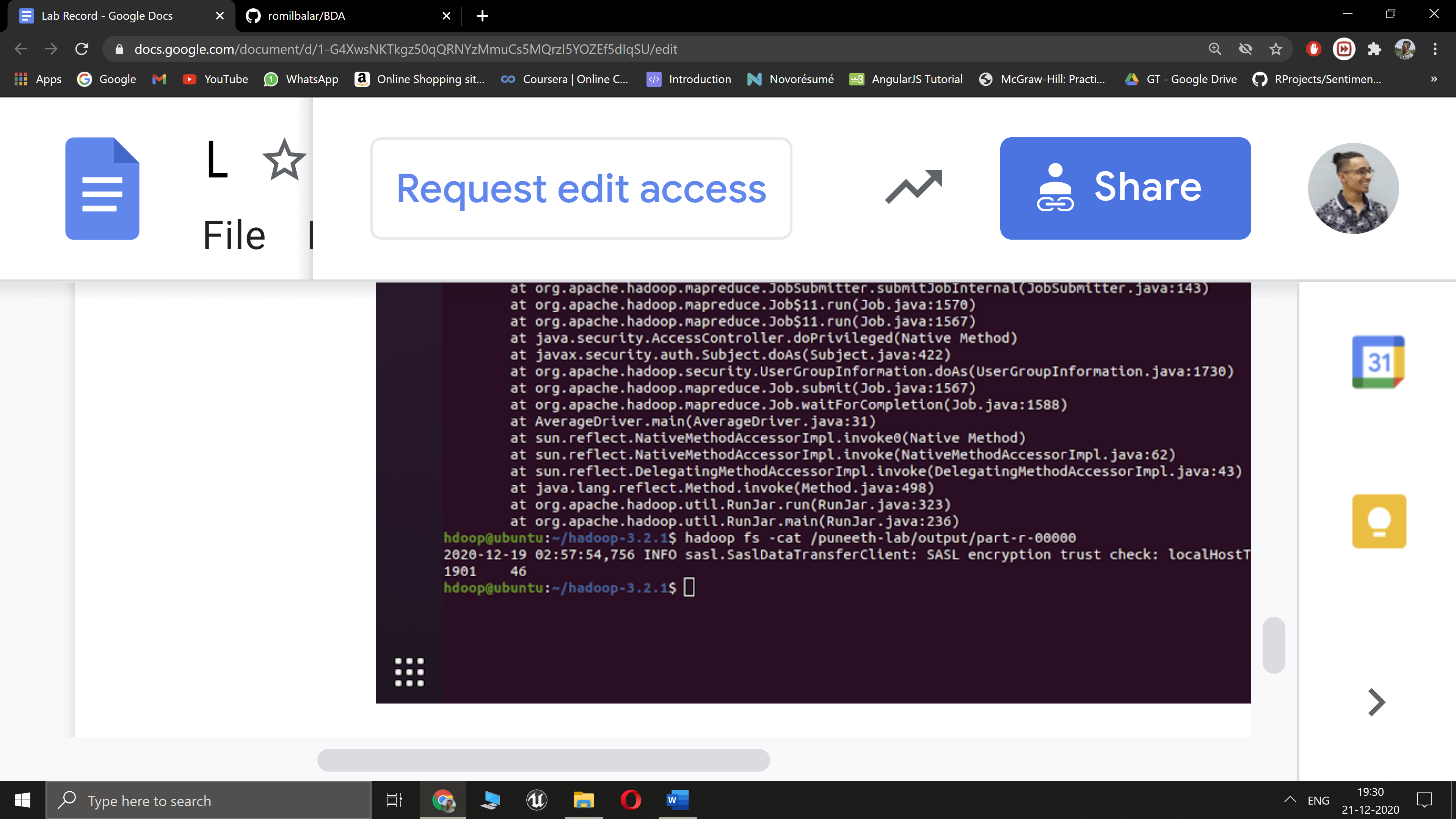
System.exit(job.waitForCompletion(true)?0:1);

}

}

**Output:**





**Program 7**

**Question:**

1. Create an external table named with the following attributes

-> Empl\_ID ->Emp\_Name -> Designation -> Salary

2. Load data into table from a given file

3. Create a view to Generate a query to retrieve the employee details who earn a salary of more than Rs 30000.

4. Alter the table to add a column Dept\_Id and Generate a query to retrieve the employee details in order by using Dept\_Id

5. Generate a query to retrieve the number of employees in each department whose salary is greater than 30000

6. Create another table Department with attributes

-> Dept\_Id ->Dept\_name ->Emp\_Id

7.Display the cumulative details of each employee along with department details

**Commands:**

su hduser

$HADOOP\_HOME/sbin/start-dfs.sh

$HADOOP\_HOME/sbin/start-yarn.sh

jps

cd $HIVE\_HOME/bin

hive

// Create database

create database if not exists Employee comment 'skn bda hive';

use Employee;

// 1. Create external table Emp

create external table if not exists Emp (Empl\_ID int, Emp\_Name String, Designation String, Salary int) row format delimited fields terminated by '\t' lines terminated by '\n';

// 2. Load data from external file

load data local inpath '/home/hduser/employee.txt' overwrite into table Emp;

select \* from Emp;

// 3. View with salary > 30000

create view Emp\_view as select \* from Emp where Salary>30000;

select \* from Emp\_view;

// 4. Alter table to add new column:

alter table Emp add columns (Dept\_ID int);

// Adding values into new column:

load data local inpath '/home/hduser/employee\_dept.txt' overwrite into table Emp;

select \* from Emp order by Dept\_ID;

// 5. Query to retrieve the number of employees in each department whose salary is greater than 30000:

select count(\*),Dept\_ID from Emp where Salary > 30000 group by Dept\_ID;

// 6. New table Department

create table if not exists Department (Dept\_ID int , Dept\_name String, Emp\_ID int) row format delimited fields terminated by '\t' lines terminated by '\n';

load data local inpath '/home/hduser/dept.txt' overwrite into table Department;

select \* from Department;

// 7. Cumulative details:

select e.Empl\_ID, e.Emp\_Name, e.Designation, e.Salary, e.Dept\_ID, d.Dept\_Name from Emp e join Department d ON (d.Dept\_ID = e.Dept\_ID);

/\* Stop HDFS and YARN \*/

$HADOOP\_HOME/sbin/stop-all.sh

**Output:**