SALINA M S MAMMY

1BM18CS417

BIG DATA ANALYTICS LAB RECORD

25 December 2020

1. Perform the following DB operations using MongoDB.
2. Create a database “Student” with the following attributes Roll no, Age, Contact No, Email-Id.

use Student

db.createCollection("Details")

****2.Insert appropriate values

db.Details.insert([{\_id:1,"Name":"Salina","Rollno":54,"Age":20,"Contact":7406621501,"email":"nvs@gmail.com"},{\_id:2,"Name":"Shubha","Rollno":56,"Age":20,"Contact":8951237313,"email":"shubhapatil@gmail.com"},{\_id:3,"Name":"Shwetha","Rollno":53,"Age":20,"Contact":8762050013,"email":"shwetha@gmail.com"},{



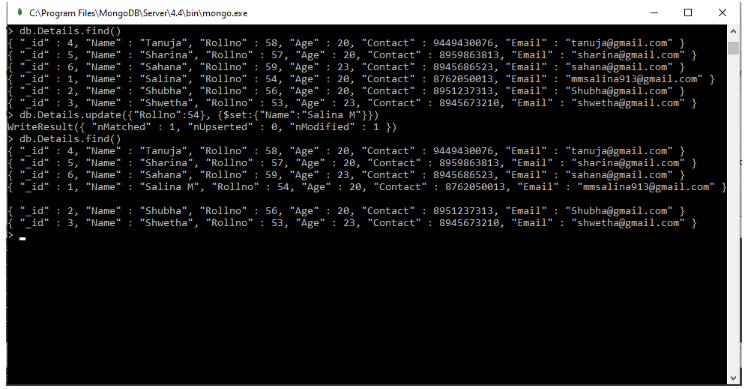
3. Write query to update Email-Id of a student with roll no 20.

db.Details.update({"Rollno":20},{$set:{"email":"20@gmail.com"}})

## 

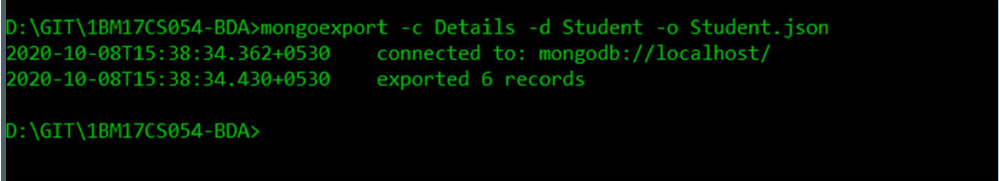
4. Replace the student name from “ABC” to “FEM” of roll no 11.

db.Details.update({"Rollno" : 11},{$set:{"Name":"FEM"}})



5. Export the created table into local file system

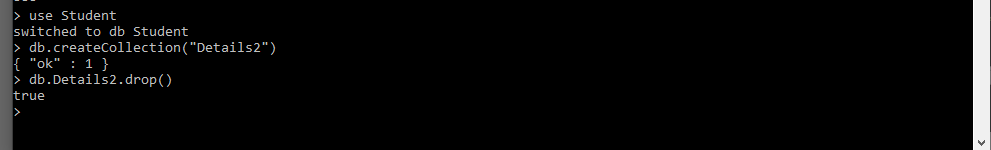
mongoexport -c Details -d Student –o Student.csv





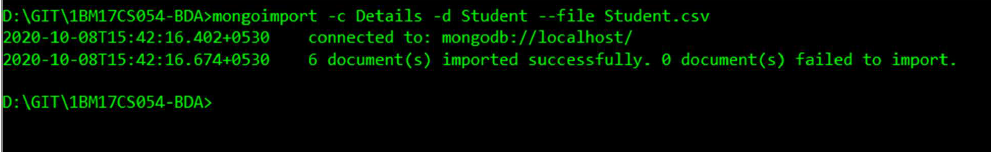
6. Drop the table

db.Details.drop()



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

mongoimport -c Details -d Student –-file Student.csv



1. Perform the following DB operations using MongoDB.

1. Create a collection by name Customers with the following attributes. Cust\_id, Acc\_Bal, Acc\_Type

use Customer

db.createCollection("Details")

2. Insert at least 5 values into the table

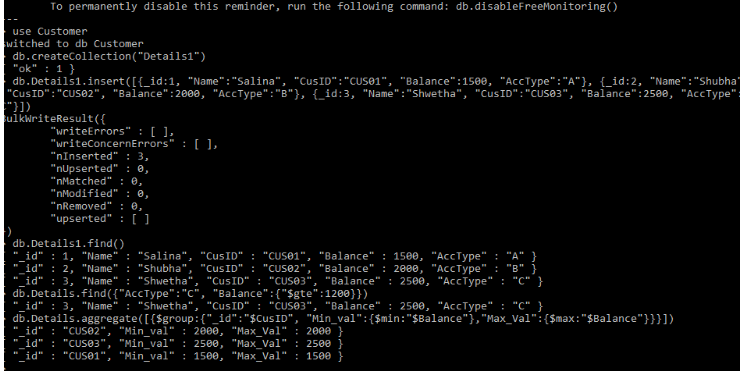
db.Details.insert([{\_id:1,"Name":"Salina","CustomerID":"CUS10","Balance":1500,"AccountType":"Z"},{\_id:2,"Name":"Shubha","CustomerID":"CUS11","Balance":1600,"AccountType":"A"},{\_id:3,"Name":"Shwetha","CustomerID":"CUS12","Balance":1000,"AccountType":"Z"},{\_id:4,"Name":"Shamina","CustomerID":"CUS13","Balance":10000,"AccountType":"A"},{\_id:5,"Name":"Yamuna","CustmerID":"CUS14","Balance":800,"AccountType":"Z"}])



3. Write a query to display those records whose total account balance is greater than 1200 of account

type ‘Z’ for each customer\_id.

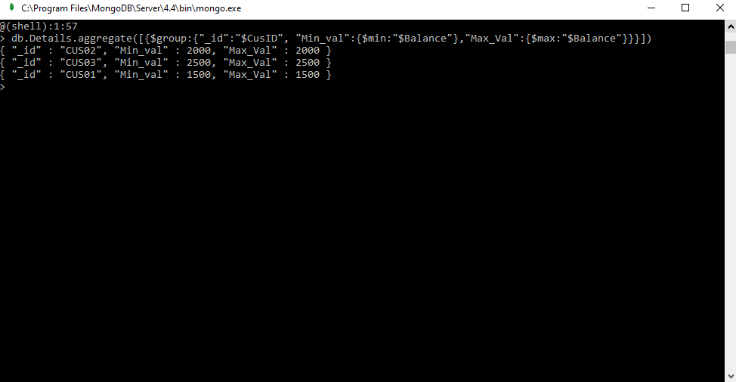
db.Details.find({"AccountType":"Z","Balance":{$gte:1200}})



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

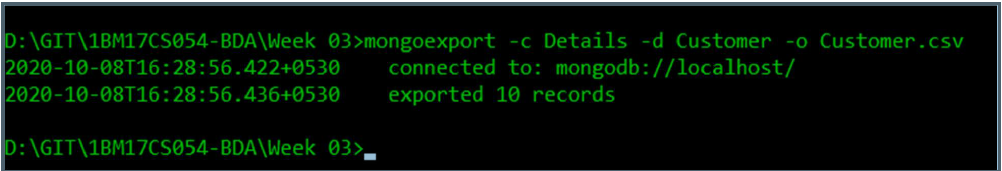
db.Details.aggregate([{$group:{"\_id":"$CustomerID","Min\_val":{$min:"$Balance"},"Max\_val":{$max

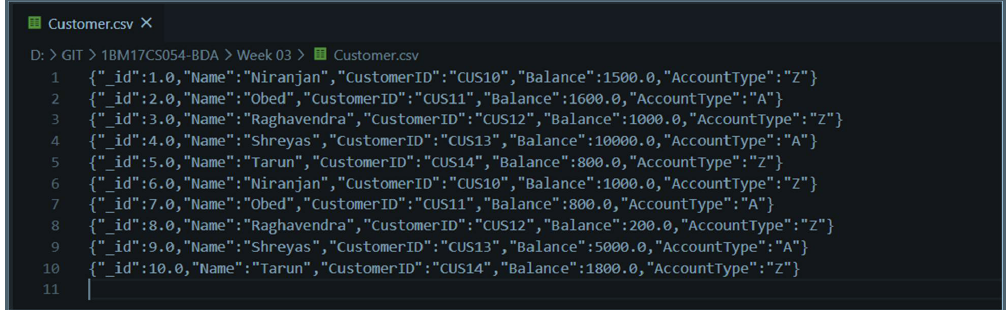
:"$Balance"}}}])



5. Export the created collection into local file system

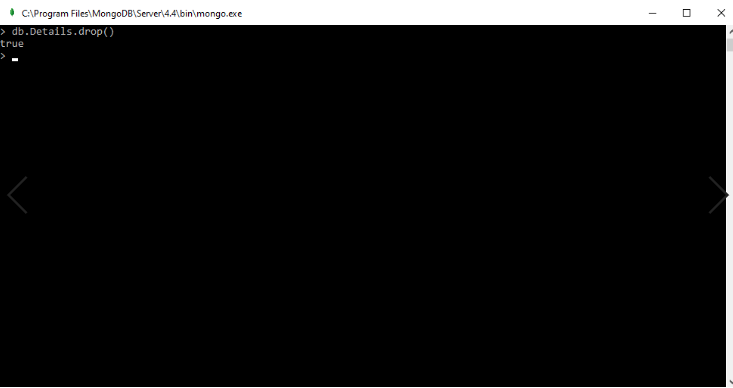
mongoexport -c Details -d Customer –o Customer.csv





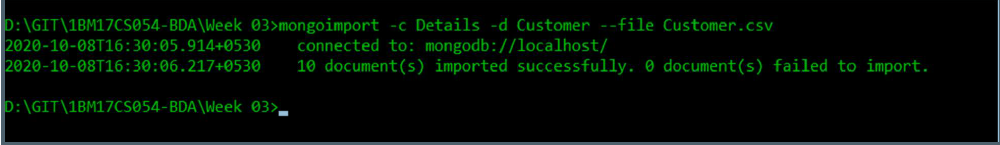
6. Drop the table

db.Details.drop()



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

mongoimport -c Details -d Customer –-file Customer.csv



1. Perform the following DB operations using Cassandra.

1. Create a keyspace by name Employee

CREATE KEYSPACE employee WITH REPLICATION =

{'class':'SimpleStrategy','replication\_factor':1};

USE 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

CREATE TABLE employee\_info (employee\_id int, employee\_name text,designation text,

date\_of\_joining timestamp, salary double,department\_name text, PRIMARY KEY(employee\_id,

salary));

3. Insert the values into the table in batch

BEGIN BATCH

INSERT INTO employee\_info(employee\_id , employee\_name ,designation , date\_of\_joining ,salary

,department\_name) VALUES(117,'Niranjan V S','Software Developer','2020-10-

19',60000,'Development')

INSERT INTO employee\_info(employee\_id , employee\_name ,designation , date\_of\_joining ,salary

,department\_name) VALUES(118,'Obed Junias','Software Developer in Test','2020-01-

15',55000,'Testing')

INSERT INTO employee\_info(employee\_id , employee\_name ,designation , date\_of\_joining ,salary

,department\_name) VALUES(119,'Raghavendra','Operations Lead','2018-04-17',90000,'Operations')

INSERT INTO employee\_info(employee\_id , employee\_name ,designation , date\_of\_joining ,salary

,department\_name) VALUES(120,'Tarun M Krishna','New Talent Recruiter','2019-02-

03',65000,'Recruitment')

INSERT INTO employee\_info(employee\_id , employee\_name ,designation , date\_of\_joining ,salary

,department\_name) VALUES(121,'Ranveer Singh','HR','2019-12-23',62000,'HR')

APPLY BATCH;

SELECT \* FROM employee\_info;

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

UPDATE employee\_info SET employee\_name = 'Shreyas K' , department\_name='Sales' WHERE

employee\_id=121;

SELECT \* FROM employee\_info;

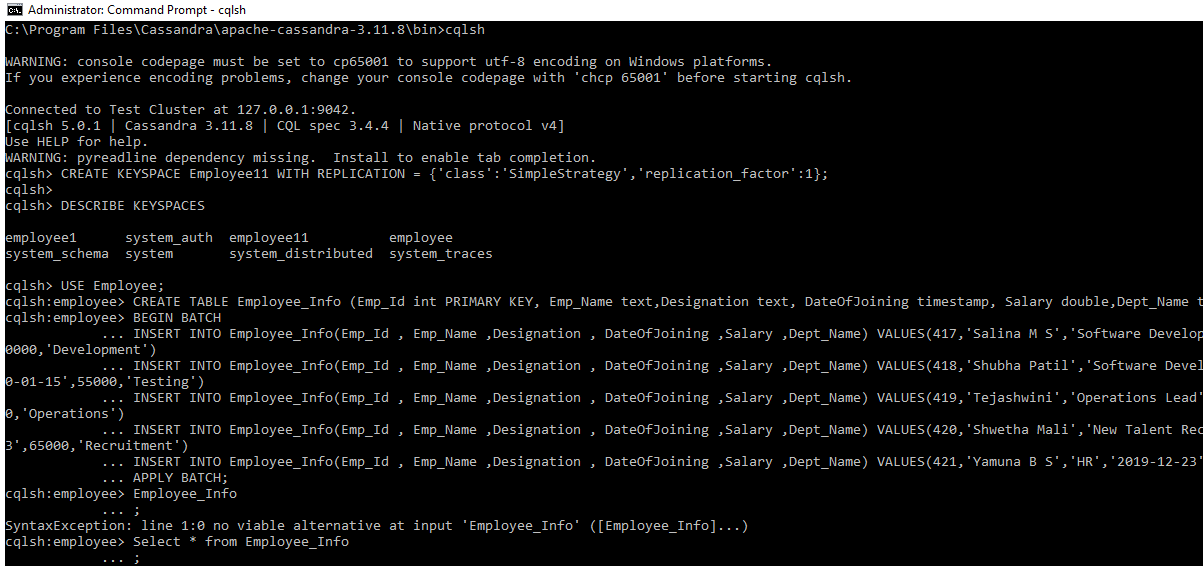
5. Sort the details of Employee records based on salary

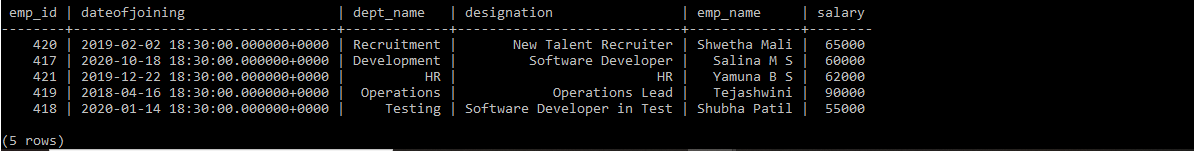
SELECT \* FROM employee\_info WHERE employee\_id IN (117,118,119,120,121) ORDER BY salary;

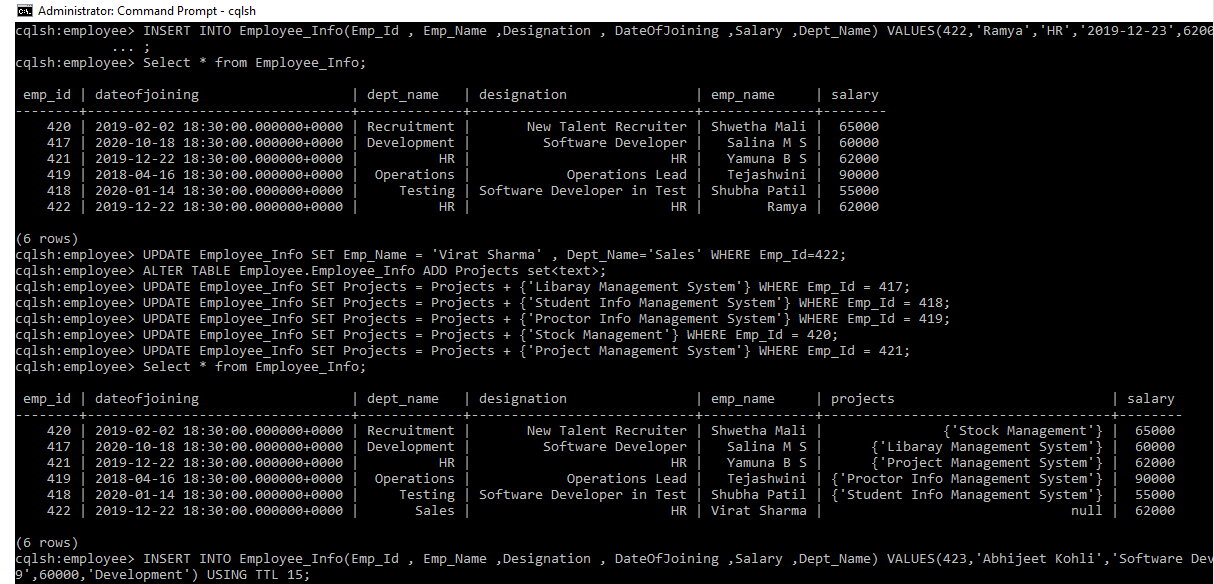
6. Alter the schema of the table Employee\_Info to add a column Projects which stores a set of

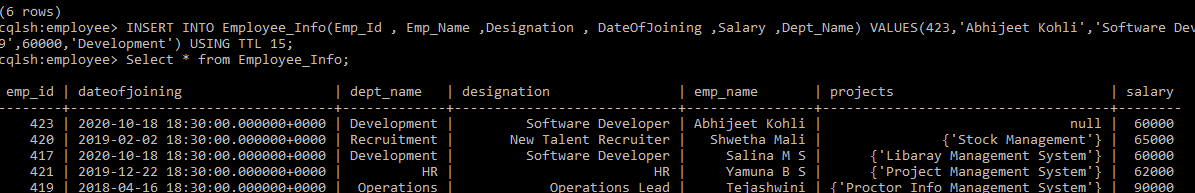
Projects done by the corresponding Employee.

ALTER TABLE employee.employee\_info ADD projects set<text>;









7. Update the altered table to add project names.

UPDATE employee\_info SET projects = projects + {'Libaray Management System'} WHERE

employee\_id = 117;

UPDATE employee\_info SET projects = projects + {'Student Information System'} WHERE

employee\_id = 118;

UPDATE employee\_info SET projects = projects + {'Student Information Management System'}

WHERE employee\_id = 119;

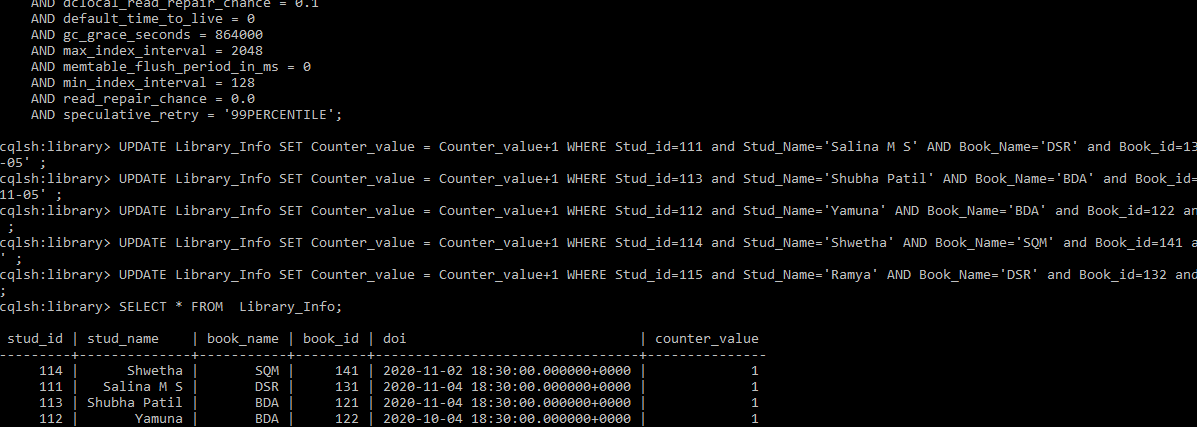
UPDATE employee\_info SET projects = projects + {'Stock Management System'} WHERE

employee\_id = 120;

UPDATE employee\_info SET projects = projects + {'Project Management System'} WHERE

employee\_id = 121;

SELECT \* FROM employee\_info;



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

INSERT INTO employee\_info(employee\_id , employee\_name ,designation , date\_of\_joining ,salary

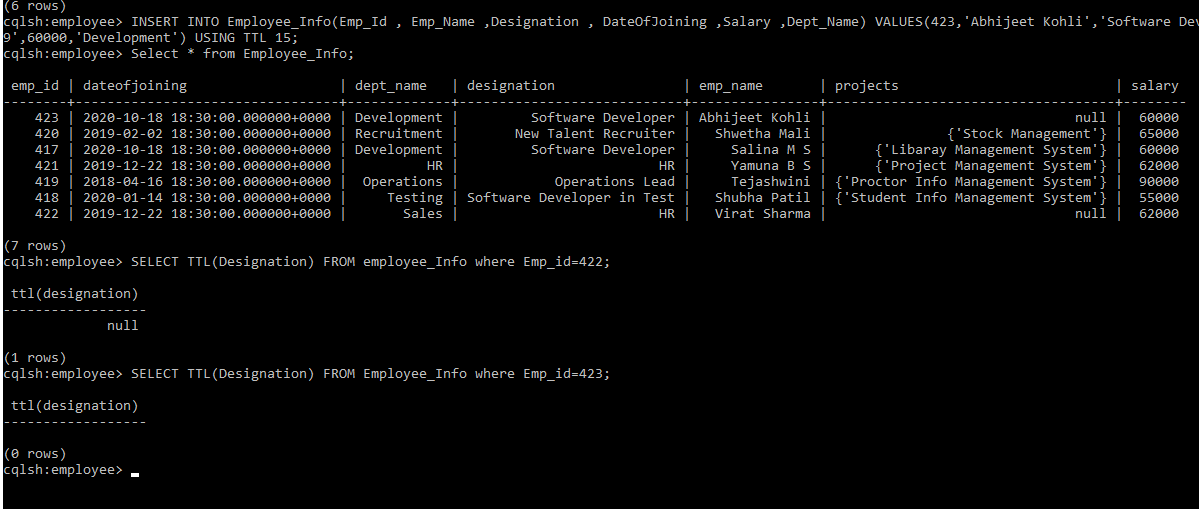
,department\_name) VALUES(122,'Abhijeet Kohli','Software Developer','2020-10-

19',60000,'Development') USING TTL 15;

SELECT \* FROM employee\_info;

SELECT TTL(designation) FROM employee\_Info where employee\_id=122;

SELECT \* FROM employee\_info;



1. Perform the following DB operations using Cassandra.

1.Create a keyspace by name Library

CREATE KEYSPACE library WITH REPLICATION = {'class':'SimpleStrategy','replication\_factor':1};

DESCRIBE KEYSPACES;

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

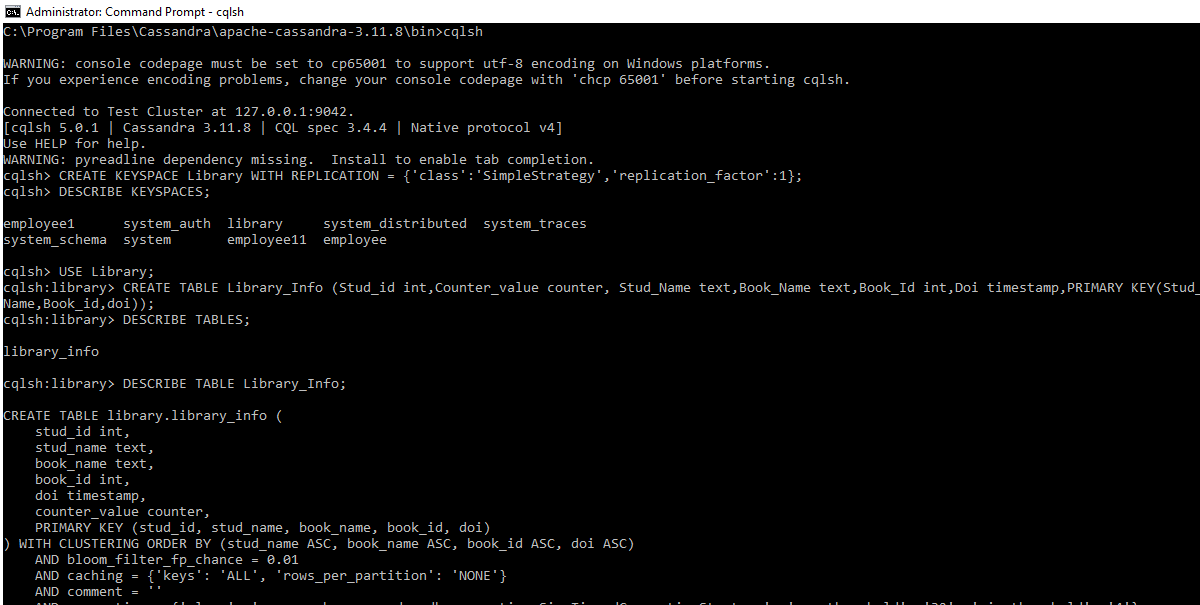
USE library;

CREATE TABLE library\_info (student\_id int,counter\_value counter, student\_name text,book\_name

text,book\_id int,date\_of\_issue timestamp,PRIMARY

KEY(student\_id,student\_name,book\_name,book\_id,date\_of\_issue));

DESCRIBE TABLES;



3. Insert the values into the table in batch

BEGIN BATCH

UPDATE library\_info SET counter\_value = counter\_value+1 WHERE student\_id=114 and

student\_name='Niranjan V S' AND book\_name='SQM' and book\_id=141 and date\_of\_issue='2020-11-

03' ;

UPDATE library\_info SET counter\_value = counter\_value+1 WHERE student\_id=111 and

student\_name='Obed Junias' AND book\_name='DSR' and book\_id=131 and date\_of\_issue='2020-11-05'

;

UPDATE library\_info SET counter\_value = counter\_value+1 WHERE student\_id=113 and

student\_name='Raghavendra' AND book\_name='BDA' and book\_id=121 and date\_of\_issue='2020-11-

05' ;

UPDATE library\_info SET counter\_value = counter\_value+1 WHERE student\_id=112 and

student\_name='Tarun M Krishna' AND book\_name='BDA' and book\_id=122 and date\_of\_issue='2020-

10-05' ;

UPDATE library\_info SET counter\_value = counter\_value+1 WHERE student\_id=115 and

student\_name='Shreyas K' AND book\_name='DSR' and book\_id=132 and date\_of\_issue='2020-11-04' ;

APPLY BATCH

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

SELECT \* FROM library\_info;

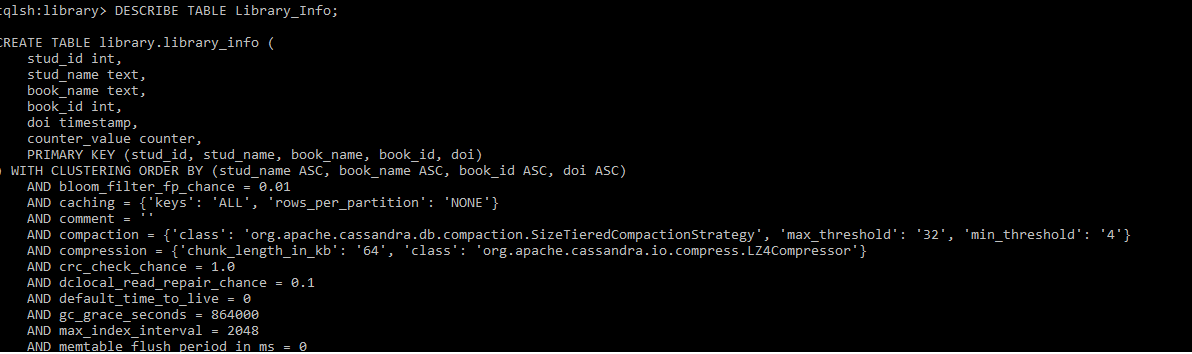
UPDATE library\_info SET counter\_value = counter\_value+1 WHERE student\_id=112 and

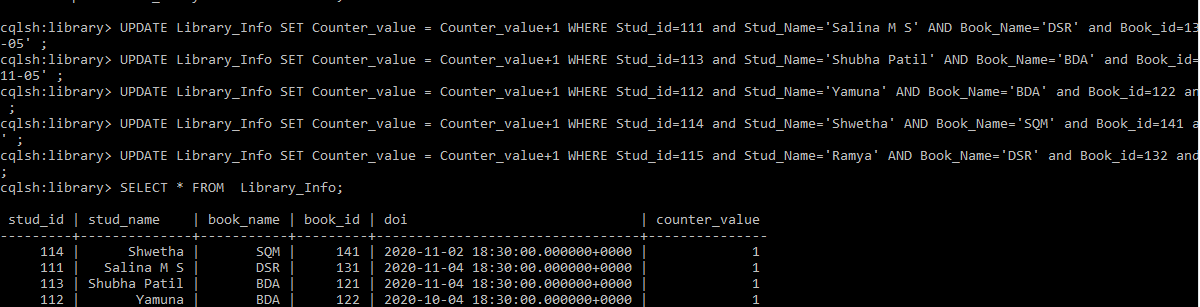
student\_name='Tarun M Krishna' AND book\_name='BDA' and book\_id=122 and date\_of\_issue='2020-

10-05' ;

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

SELECT book\_name,counter\_value FROM library\_info WHERE student\_id=112;





6. Export the created column to a csv file

COPY library\_info(student\_id,counter\_value,student\_name,book\_name,book\_id,date\_of\_issue) TO

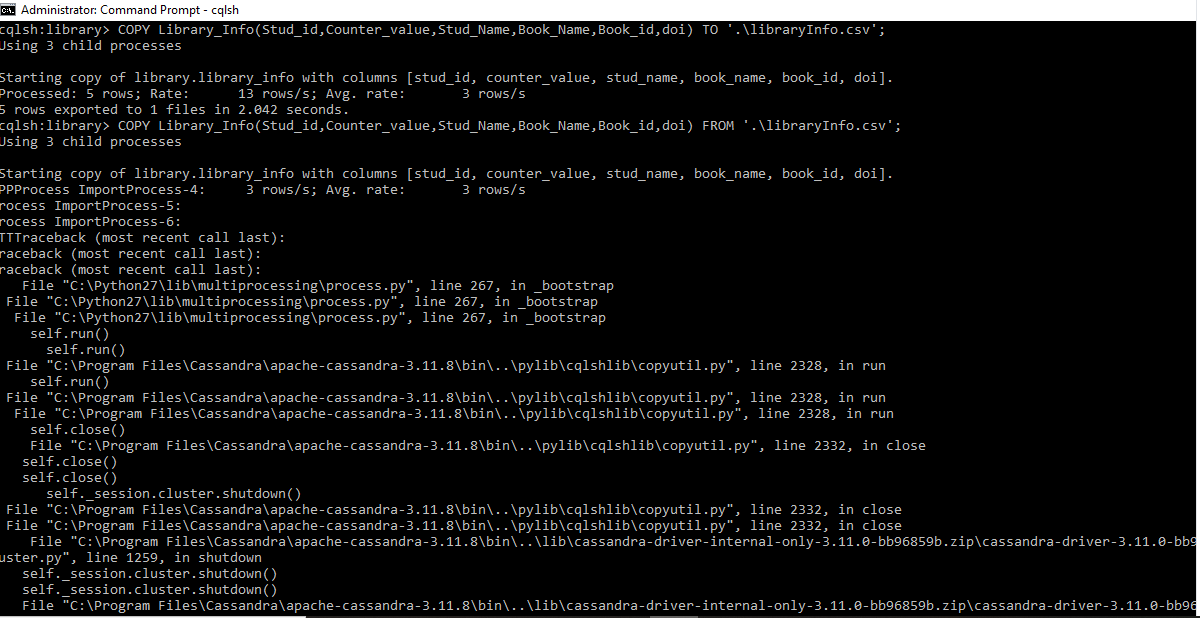
'.\library\_information.csv';

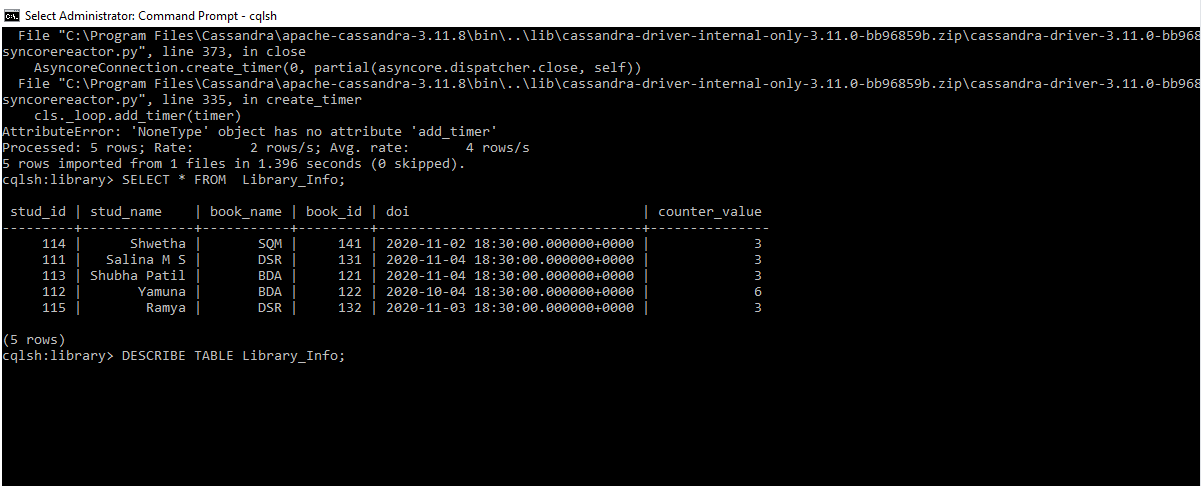
7. Import a given csv dataset from local file system into Cassandra column family

COPY library\_info(student\_id,counter\_value,student\_name,book\_name,book\_id,date\_of\_issue) FROM

'.\library\_information.csv';

SELECT \* FROM library\_info;





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

To start all the Hadoop deamons

$ ssh localhost

$ cd Hadoop/hadoop-3.2.1

$ sbin/start-dfs.sh

$ sbin/start-yarn.sh

To create a directory in hdfs

$ hadoop fs -mkdir /rgs1

To view all the directories in hdfs

$ hadoop fs -ls /

To copy a file from local system to hdfs directory

$ Hadoop fs -copyFromLocal /home/niranjanvs/Desktop/file1.txt /rgs1/test.txt

To view all files in /rgs1 hdfs directory

$ hadoop fs -ls /rgs1

To run a MapReduce program

$ hadoop jar /home/niranjanvs/Desktop/wordcount.jar WordCount /rgs1/test.txt /rgs1/output

To view the output text

$ hadoop fs -cat /rgs1/output/part-r-00000

To stop all the Hadoop deamons

$ sbin/stop-yarn.sh

$ sbin/stop-dfs.sh

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.mapreduce.Mapper;

import org.apache.hadoop.mapreduce.Reducer;

import org.apache.hadoop.conf.Configuration;

import org.apache.hadoop.mapreduce.Job;

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

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

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

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

import org.apache.hadoop.fs.Path;

public class WordCount{

public static class Map extends

Mapper&lt;LongWritable,Text,Text,IntWritable&gt; {

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

IOException,InterruptedException{

String line = value.toString();

StringTokenizer tokenizer = new StringTokenizer(line);

while (tokenizer.hasMoreTokens()) {

value.set(tokenizer.nextToken());

context.write(value, new IntWritable(1));

}

}

}

public static class Reduce extends

Reducer&lt;Text,IntWritable,Text,IntWritable&gt; {

public void reduce(Text key, Iterable&lt;IntWritable&gt;

values,Context context) throws IOException,InterruptedException {

int sum=0;

for(IntWritable x: values)

{

sum+=x.get();

}

context.write(key, new IntWritable(sum));

}

}

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

Configuration conf= new Configuration();

Job job = new Job(conf,"My Word Count Program");

job.setJarByClass(WordCount.class);

job.setMapperClass(Map.class);

job.setReducerClass(Reduce.class);

job.setOutputKeyClass(Text.class);

job.setOutputValueClass(IntWritable.class);

job.setInputFormatClass(TextInputFormat.class);

job.setOutputFormatClass(TextOutputFormat.class);

Path outputPath = new Path(args[1]);

//Configuring the input/output path from the filesystem into the job

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

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

//deleting the output path automatically from hdfs so that we don't

have to delete it explicitly

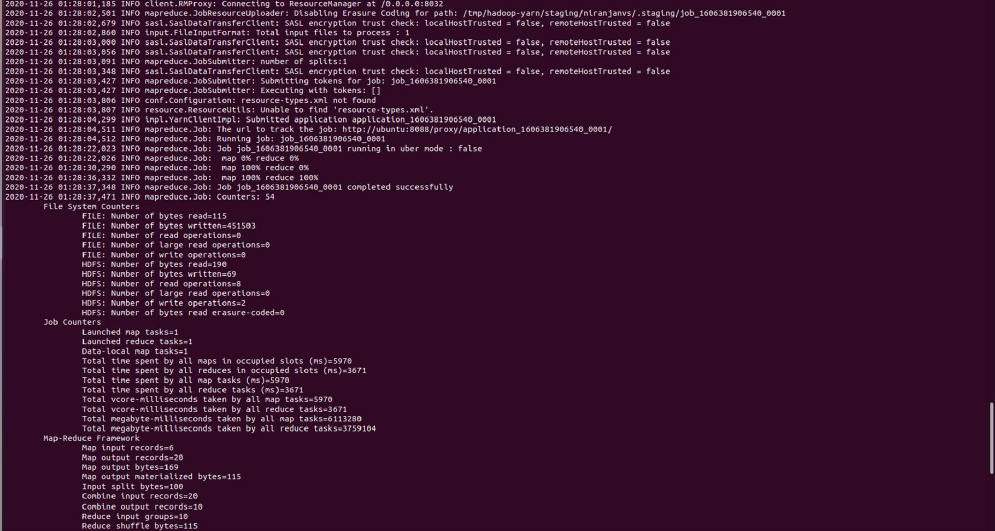
outputPath.getFileSystem(conf).delete(outputPath);

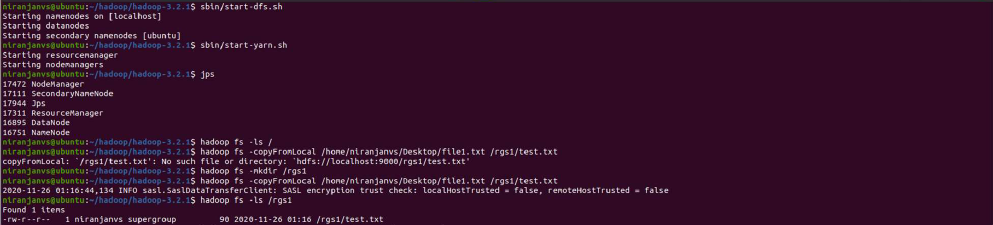
//exiting the job only if the flag value becomes false

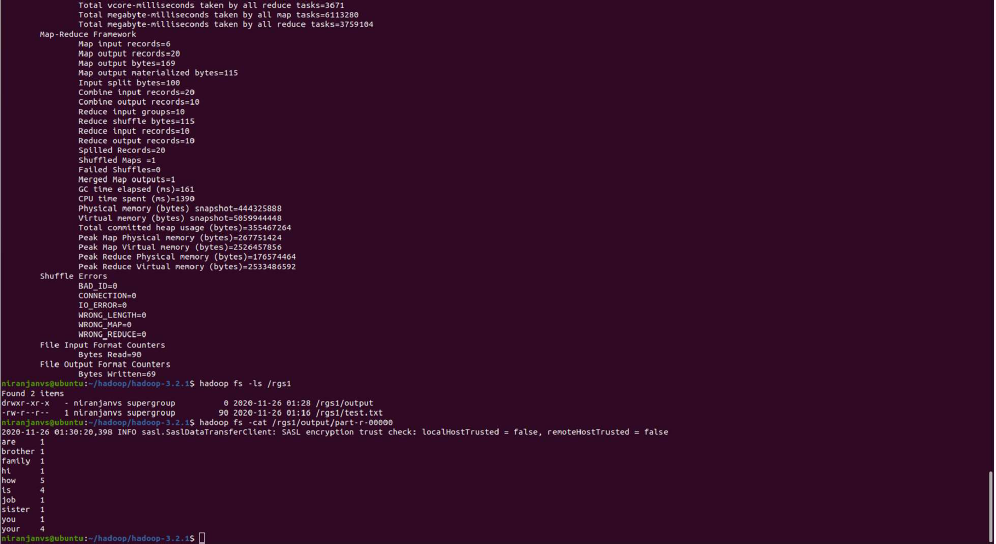
System.exit(job.waitForCompletion(true) ? 0 : 1);

}

}







1. For the given file, Create a Map Reduce program to Find the average temperature

for each year from NCDC data set.

To create jar files using .java files

$ javac AverageReducer.java AverageDriver.java AverageMapper.java -cp $(hadoop classpath)

$ jar -cf Average.jar AverageReducer.class AverageDriver.class AverageMapper.class

To start all the Hadoop deamons

$ ssh localhost

$ cd Hadoop/hadoop-3.2.1

$ sbin/start-dfs.sh

$ sbin/start-yarn.sh

To create a directory in hdfs

$ hadoop fs -mkdir /rgs1

To view all the directories in hdfs

$ hadoop fs -ls /

To copy a file from local system to hdfs directory

$ Hadoop fs -copyFromLocal /home/niranjanvs/Desktop/1901 /rgs1/AverageTest.txt

To view all files in /rgs1 hdfs directory

$ hadoop fs -ls /rgs1

To run a MapReduce program

$ hadoop jar /home/niranjanvs/Desktop/Average.jar AverageDriver /rgs1/AverageTest.txt

/rgs1/AverageOutput

To view the output text

$ hadoop fs -cat /rgs1/output/part-r-00000

To stop all the Hadoop deamons

$ sbin/stop-yarn.sh

$ sbin/stop-dfs.sh

AverageDriver.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);

}

}

AverageMapper.java

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

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));

}

}

AverageReducer.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)

{

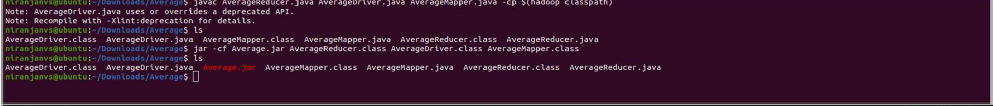
max\_temp += value.get();

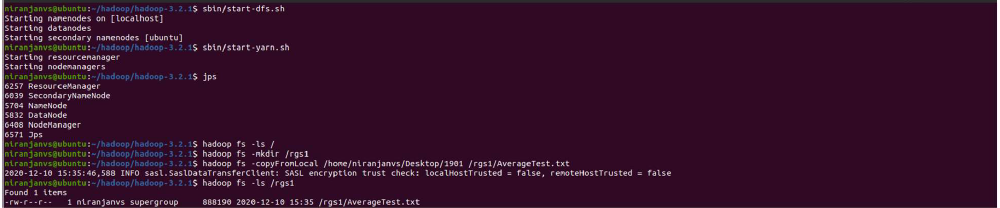
count+=1;

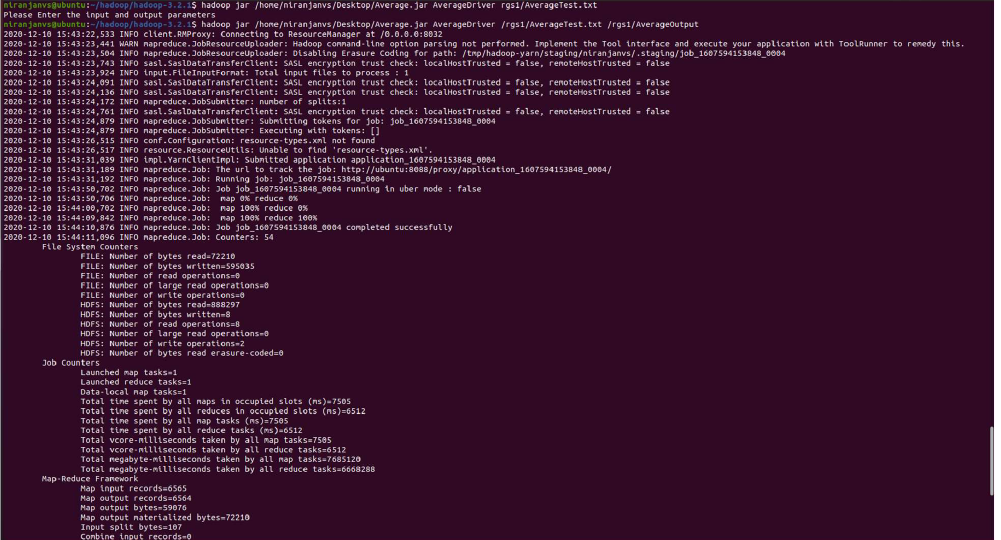
}

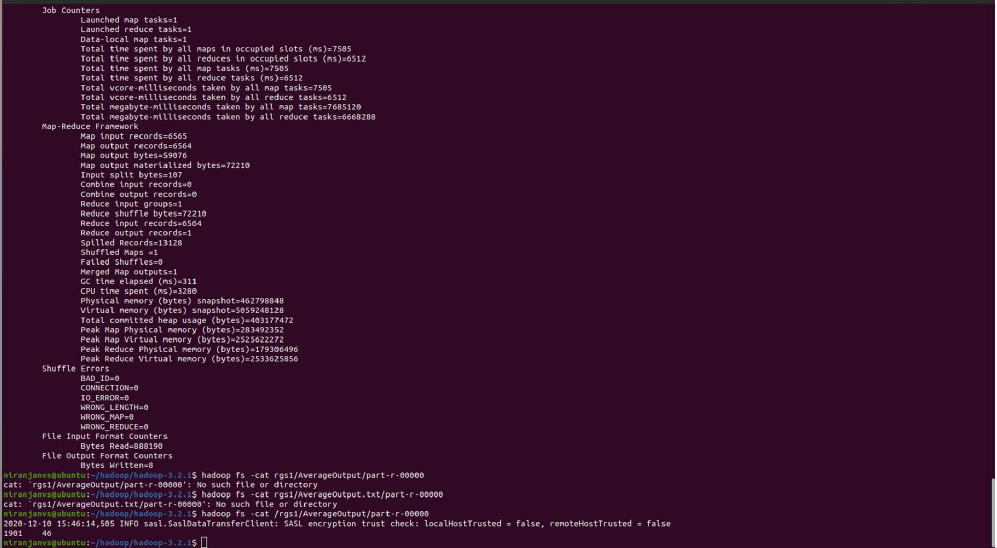
context.write(key, new IntWritable(max\_temp/count));

}









7.Write Queries in Hive to do the following

1. Create an external table named with the following attributes -> Empl\_ID ->Emp\_Name ->

Designation -> Salary

create database if not exists Employee comment 'BDA LAB WEEK 09';

use Employee;

create external table if not exists Employee (Empl\_ID int, Emp\_Name String, Designation String, Salary

int) row format delimited fields terminated by ',' lines terminated by '\n';

2. Load data into table from a given file

load data local inpath '/home/niranjanvs/Desktop/employee' overwrite into table Employee;

select \* from Employee;

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

than Rs 30000.

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

select \* from Employee\_view;

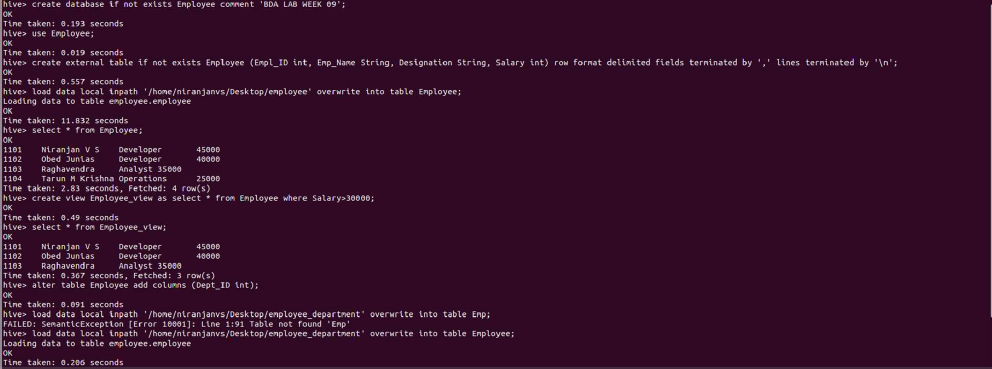
4. Alter the table to add a column Dept\_Id and Generate a query to retrieve the employee details

in order by using Dept\_Id

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

load data local inpath '/home/niranjanvs/Desktop/employee\_department' overwrite into table Employee;

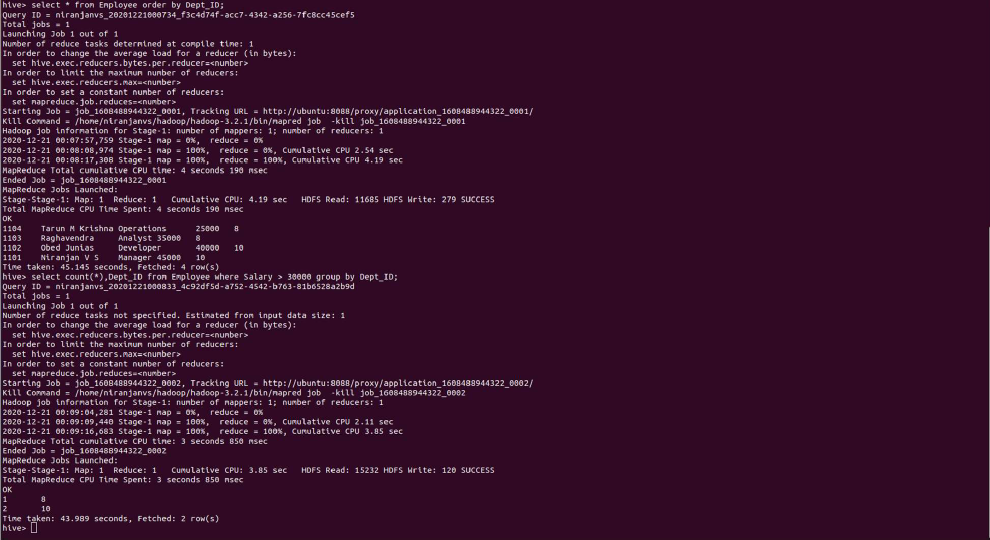
select \* from Employee order by Dept\_ID;



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

greater than 30000

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



6. Create another table Department with attributes -> Dept\_Id ->Dept\_name ->Emp\_Id

create table if not exists Department (Dept\_ID int , Dept\_name String, Emp\_ID int) row format

delimited fields terminated by ',' lines terminated by '\n';

load data local inpath '/home/niranjanvs/Desktop/department' overwrite into table Department;

select \* from Department;

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

select e.Empl\_ID, e.Emp\_Name, e.Designation, e.Salary, e.Dept\_ID, d.Dept\_Name from Employee e

join Department d ON (d.Dept\_ID = e.Dept\_ID);

