**B.M.S. COLLEGE OF ENGINEERING BENGALURU**

Autonomous Institute, Affiliated to VTU



Lab Record

**BIG DATA ANALYTICS**

th

Submitted in partial fulfilment for the 6 Semester Laboratory

Bachelor of Technology

in

Computer Science and Engineering

*Submitted by:*

**PREETHAM H**

1BM19CS120

Department of Computer Science and Engineering B.M.S. College of Engineering

Bull Temple Road, Basavanagudi, Bangalore 560 019

Mar-July 2022

**B.M.S. COLLEGE OF ENGINEERING**

**DEPARTMENT OF COMPUTER SCIENCE AND**

**ENGINEERING**



***CERTIFICATE***

This is to certify that the Big Data Analytics(20CS6PEBDA) laboratory has been carried out by PREETHAM H (1BM19CS120) during the 6 th Semester Mar-July-2022

Signature of the Faculty In charge:

ANTARA ROY CHOUDHURY( Assistant Professor)

Department of Computer Science and Engineering

B.M.S. College of Engineering, Bangalore

# TABLE OF CONTENTS

|  |  |
| --- | --- |
| **SL NO** | **TITLE** |
| 1 | EMPLOYEE DATABASE |
| 2 | LIBRARY DATABASE |
| 3 | MONGODB SAMPLE |
| 4 | HADOOP INSTALLATION |
| 5 | HADOOP SAMPLE |
| 6 | MAPREDUCE TEMPERATURE |
| 7 | MAPREDUCE TOPN |
| 8 | MAPREDUCE JOIN |
| 9 | SCALA INSTALLATION |
| 10 | SCALA WORDCOUNT |

# Employee database (CASSANDRA)

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

1. Insert the values into the table in batch
2. Update Employee name and Department of Emp-Id 121
3. Sort the details of Employee records based on salary
4. Alter the schema of the table Employee\_Info to add a column Projects which stores a set of Projects done by the corresponding Employee.
5. Update the altered table to add project names.

7.Create a TTL of 30 seconds to display the values of Employees.

cqlsh:employee\_info> begin batch

... insert into

employee\_details(emp\_id,emp\_name,designation,doj,salary,dept\_name)values(101,'Sakshi','manag er','2020-09-08',35000,'testing')

... insert into

employee\_details(emp\_id,emp\_name,designation,doj,salary,dept\_name)values(201,'Sneha','manag er','2020-08-08',85000,'development')

... insert into

employee\_details(emp\_id,emp\_name,designation,doj,salary,dept\_name)values(201,'Shreya','associ ate','2020-07-08',75000,'HR')

... apply batch;

cqlsh:employee\_info> select \*from employee\_details;

emp\_id | salary | dept\_name | designation | doj | emp\_name --------+--------+-------------+-------------+---------------------------------+----------

201 | 75000 | HR | associate | 2020-07-08 07:00:00.000000+0000 | Shreya

201 | 85000 | development | manager | 2020-08-08 07:00:00.000000+0000 | Sneha

101 | 35000 | testing | manager | 2020-09-08 07:00:00.000000+0000 | Sakshi

(3 rows) cqlsh:employee\_info> delete from employee\_details where emp\_id=201; cqlsh:employee\_info> select \*from employee\_details;

emp\_id | salary | dept\_name | designation | doj | emp\_name

--------+--------+-----------+-------------+---------------------------------+----------

101 | 35000 | testing | manager | 2020-09-08 07:00:00.000000+0000 | Sakshi

(1 rows)

cqlsh:employee\_info> begin batch insert into

employee\_details(emp\_id,emp\_name,designation,doj,salary,dept\_name)values(301,'Sneha','manag er','2020-08-08',85000,'development') insert into

employee\_details(emp\_id,emp\_name,designation,doj,salary,dept\_name)values(201,'Shreya','associ ate','2020-07-08',75000,'HR') apply batch; cqlsh:employee\_info> select \*from employee\_details;

emp\_id | salary | dept\_name | designation | doj | emp\_name

--------+--------+-------------+-------------+---------------------------------+----------

201 | 75000 | HR | associate | 2020-07-08 07:00:00.000000+0000 | Shreya

101 | 35000 | testing | manager | 2020-09-08 07:00:00.000000+0000 | Sakshi

301 | 85000 | development | manager | 2020-08-08 07:00:00.000000+0000 | Sneha

(3 rows) cqlsh:employee\_info> alter table employee\_details add project text;

cqlsh:employee\_info> update employee\_details set project='library app' where emp\_id=201 and salary=75000;

cqlsh:employee\_info> update employee\_details set project='medicine app' where emp\_id=301 and salary=85000;

cqlsh:employee\_info> update employee\_details set project='fitness app' where emp\_id=101 and salary=85000; cqlsh:employee\_info> select \*from employee\_details;

emp\_id | salary | dept\_name | designation | doj | emp\_name | project

--------+--------+-------------+-------------+---------------------------------+----------+--------------

201 | 75000 | HR | associate | 2020-07-08 07:00:00.000000+0000 | Shreya | library app

101 | 35000 | testing | manager | 2020-09-08 07:00:00.000000+0000 | Sakshi | null

101 | 85000 | null | null | null | null | fitness app

301 | 85000 | development | manager | 2020-08-08 07:00:00.000000+0000 | Sneha | medicine app

(4 rows)

cqlsh:employee\_info> update employee\_details set project='fitness app' where emp\_id=101 and salary=35000; cqlsh:employee\_info> select \*from employee\_details;

emp\_id | salary | dept\_name | designation | doj | emp\_name | project

--------+--------+-------------+-------------+---------------------------------+----------+--------------

201 | 75000 | HR | associate | 2020-07-08 07:00:00.000000+0000 | Shreya | library app

101 | 35000 | testing | manager | 2020-09-08 07:00:00.000000+0000 | Sakshi | fitness app

101 | 85000 | null | null | null | null | fitness app

301 | 85000 | development | manager | 2020-08-08 07:00:00.000000+0000 | Sneha | medicine app

(4 rows) cqlsh:employee\_info> delete from employee\_details where emp\_id=1 and salary=85000; cqlsh:employee\_info> select \*from employee\_details;

emp\_id | salary | dept\_name | designation | doj | emp\_name | project

--------+--------+-------------+-------------+---------------------------------+----------+--------------

201 | 75000 | HR | associate | 2020-07-08 07:00:00.000000+0000 | Shreya | library app 101 | 35000 | testing | manager | 2020-09-08 07:00:00.000000+0000 | Sakshi | fitness app

101 | 85000 | null | null | null | null | fitness app

301 | 85000 | development | manager | 2020-08-08 07:00:00.000000+0000 | Sneha | medicine app

(4 rows) cqlsh:employee\_info> select \*from employee\_details;

emp\_id | salary | dept\_name | designation | doj | emp\_name | project

--------+--------+-------------+-------------+---------------------------------+----------+--------------

201 | 75000 | HR | associate | 2020-07-08 07:00:00.000000+0000 | Shreya | library app

101 | 35000 | testing | manager | 2020-09-08 07:00:00.000000+0000 | Sakshi | fitness app

401 | 65000 | testing | manager | 2020-05-08 07:00:00.000000+0000 | Resh | null

301 | 85000 | development | manager | 2020-08-08 07:00:00.000000+0000 | Sneha | medicine app

(4 rows)

cqlsh:employee\_info> insert into

employee\_details(emp\_id,emp\_name,designation,doj,salary,dept\_name)values(501,'wesh','manage r','2020-04-08',95000,'testing') using ttl 30;

cqlsh:employee\_info> select ttl(emp\_name)from employee\_details where emp\_id=501 and salary=95000;

ttl(emp\_name)

---------------

24

(1 rows) cqlsh:employee\_info> select \*from employee\_details;

emp\_id | salary | dept\_name | designation | doj | emp\_name | project

--------+--------+-------------+-------------+---------------------------------+----------+--------------

201 | 75000 | HR | associate | 2020-07-08 07:00:00.000000+0000 | Shreya | library app 101 | 35000 | testing | manager | 2020-09-08 07:00:00.000000+0000 | Sakshi | fitness app

401 | 65000 | testing | manager | 2020-05-08 07:00:00.000000+0000 | Resh | null

301 | 85000 | development | manager | 2020-08-08 07:00:00.000000+0000 | Sneha | medicine app

(4 rows) cqlsh:employee\_info> paging off

Disabled Query paging.

cqlsh:employee\_info> select \*from employee\_details where emp\_id in(201,101,301) order by salary;

emp\_id | salary | dept\_name | designation | doj | emp\_name | project

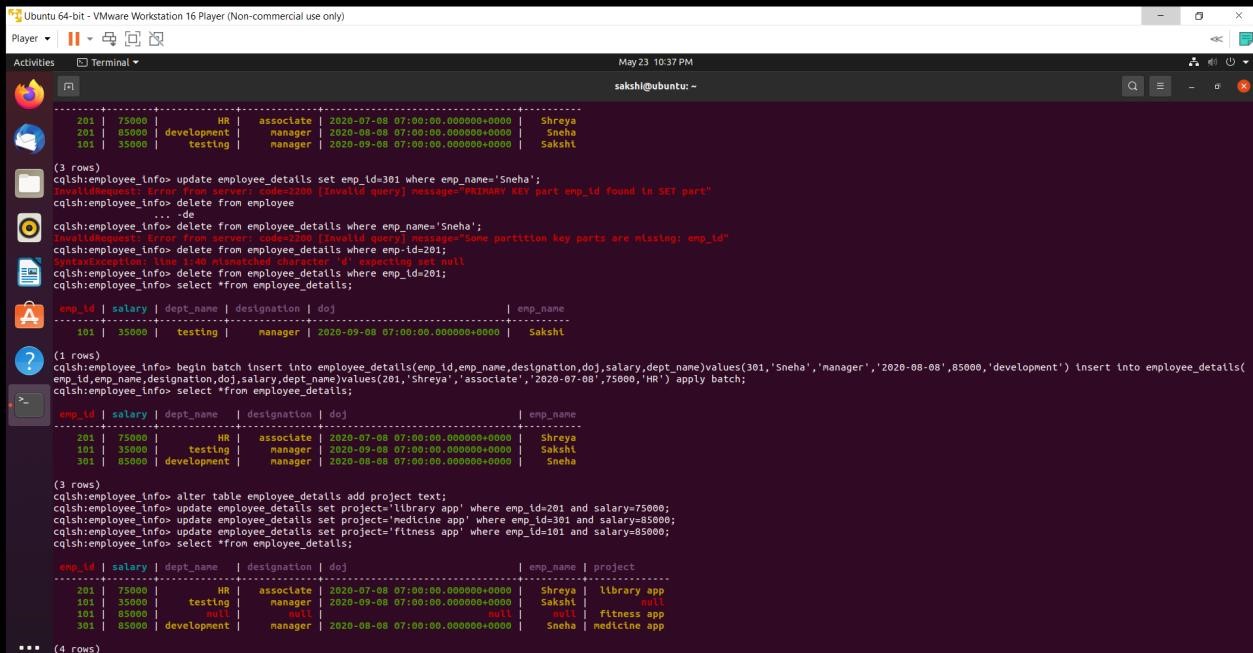
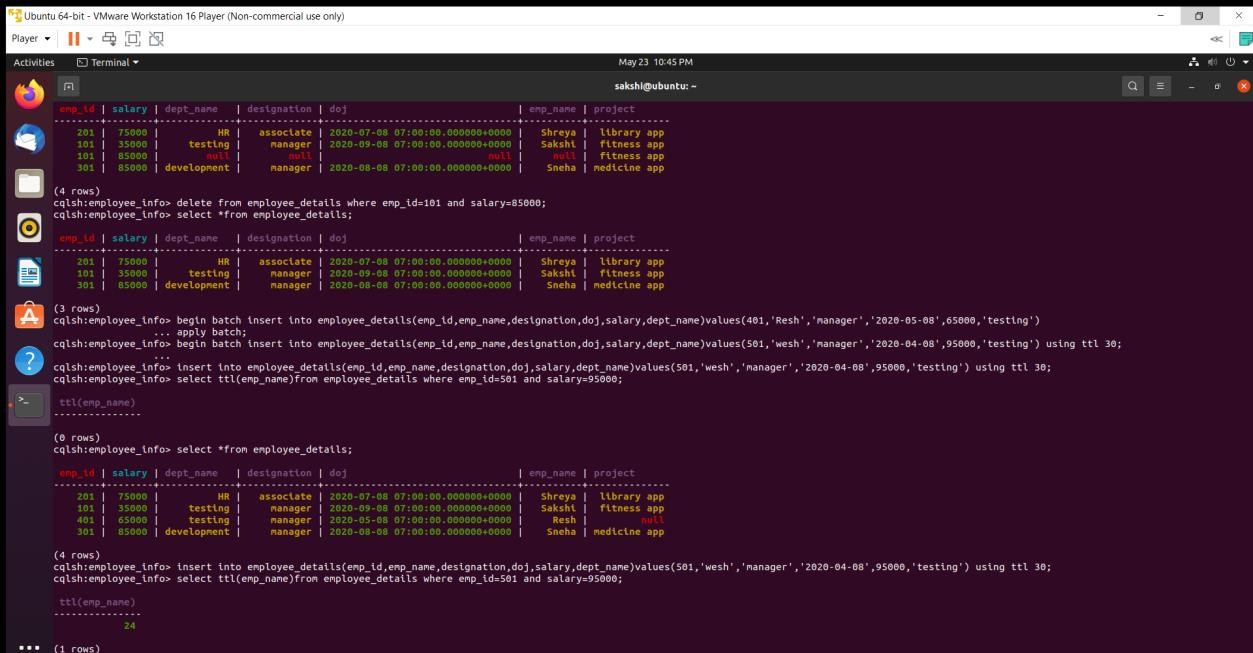
--------+--------+-------------+-------------+---------------------------------+----------+--------------

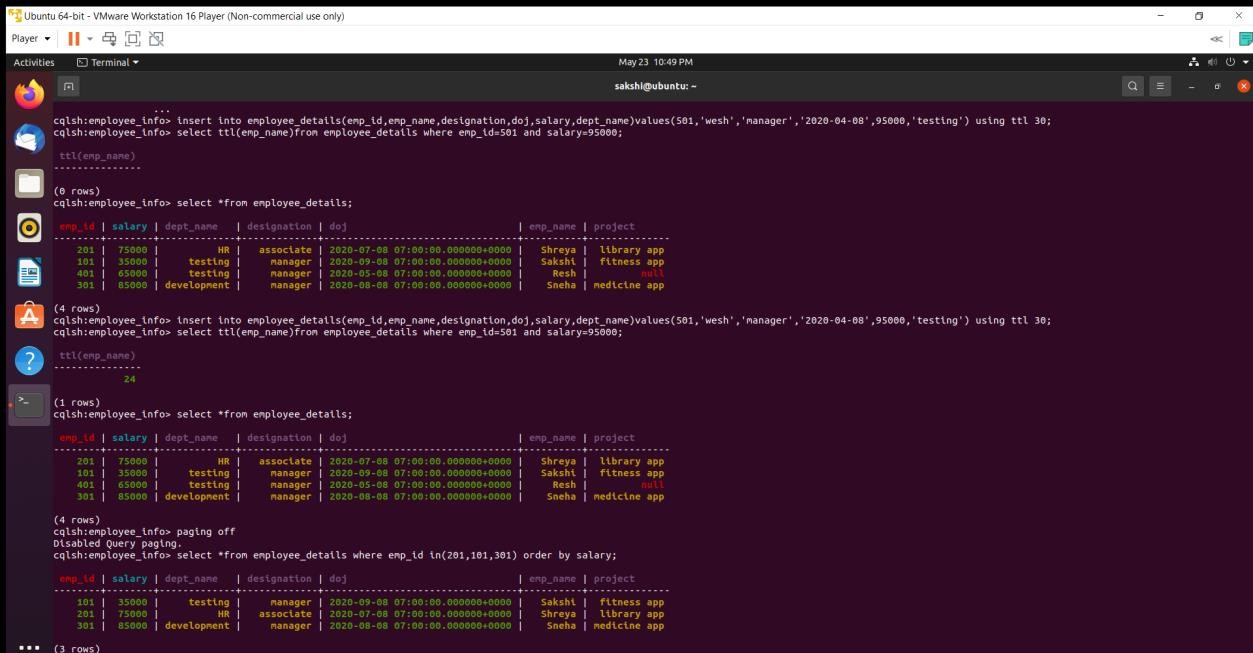
101 | 35000 | testing | manager | 2020-09-08 07:00:00.000000+0000 | Sakshi | fitness app

201 | 75000 | HR | associate | 2020-07-08 07:00:00.000000+0000 | Shreya | library app

301 | 85000 | development | manager | 2020-08-08 07:00:00.000000+0000 | Sneha | medicine app

SCREENSHOTS:





# LIBRARY DATABASE (CASSANDRA)

Question -

Perform the following DB operations using Cassandra.

1.Create a keyspace by name Library

1. 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

1. Insert the values into the table in batch
2. Display the details of the table created and increase the value of the counter
3. Write a query to show that a student with id 112 has taken a book “BDA” 2 times.
4. Export the created column to a csv file
5. Import a given csv dataset from local file system into Cassandra column family

cqlsh> use library\_info; cqlsh:library\_info> create table library\_details(stud\_id int,counter\_value

... counter,stud\_name text,book\_name text,date\_of\_issue timestamp,book\_id ... int,primary key(stud\_id,stud\_name,book\_name,date\_of\_issue,book\_id)); cqlsh:library\_info> update library\_details set counter\_value=counter\_value+1

... where stud\_id=111 and stud\_name='sam' and book\_name='ML' and

... date\_of\_issue='2020-11-09' and book\_id=200; cqlsh:library\_info> select \*from library\_details;

stud\_id | stud\_name | book\_name | date\_of\_issue | book\_id | counter\_value

---------+-----------+-----------+---------------------------------+---------+---------------

111 | sam | ML | 2020-11-09 08:00:00.000000+0000 | 200 | 1

(1 rows)

cqlsh:library\_info> update library\_details set counter\_value=counter\_value+1 where stud\_id=112 and stud\_name='sakshi' and book\_name='BDA' and date\_of\_issue='2020-01-01' and book\_id=300;

cqlsh:library\_info> update library\_details set counter\_value=counter\_value+1 where stud\_id=115 and stud\_name='aditya' and book\_name='OOMD' and date\_of\_issue='2020-06-01' and book\_id=400;

cqlsh:library\_info> select \*from library\_details;

stud\_id | stud\_name | book\_name | date\_of\_issue | book\_id | counter\_value

---------+-----------+-----------+---------------------------------+---------+---------------

1. | sam | ML | 2020-11-09 08:00:00.000000+0000 | 200 | 1
2. | sakshi | BDA | 2020-01-01 08:00:00.000000+0000 | 300 | 1 115 | aditya | OOMD | 2020-06-01 07:00:00.000000+0000 | 400 | 1

qlsh:library\_info> copy

library\_details(stud\_id,stud\_name,book\_name,book\_id,date\_of\_issue,counter\_value) to 'C:\Desktop\sample.csv';

Using 1 child processes

Starting copy of library\_info.library\_details with columns [stud\_id, stud\_name, book\_name, book\_id, date\_of\_issue, counter\_value].

cqlshlib.copyutil.ExportProcess.write\_rows\_to\_csv(): writing row cqlshlib.copyutil.ExportProcess.write\_rows\_to\_csv(): writing row cqlshlib.copyutil.ExportProcess.write\_rows\_to\_csv(): writing row/s

Processed: 3 rows; Rate: 37 rows/s; Avg. rate: 6 rows/s

3 rows exported to 1 files in 0.500 seconds. cqlsh:library\_info> truncate library\_details;

cqlsh:library\_info> copy

library\_details(stud\_id,stud\_name,book\_name,book\_id,date\_of\_issue,counter\_value) from 'C:\Desktop\sample.csv';

Using 1 child processes

Starting copy of library\_info.library\_details with columns [stud\_id, stud\_name, book\_name, book\_id, date\_of\_issue, counter\_value].

Processed: 3 rows; Rate: 3 rows/s; Avg. rate: 5 rows/s

3 rows imported from 1 files in 0.592 seconds (0 skipped).

cqlsh:library\_info> select \*from library\_details;

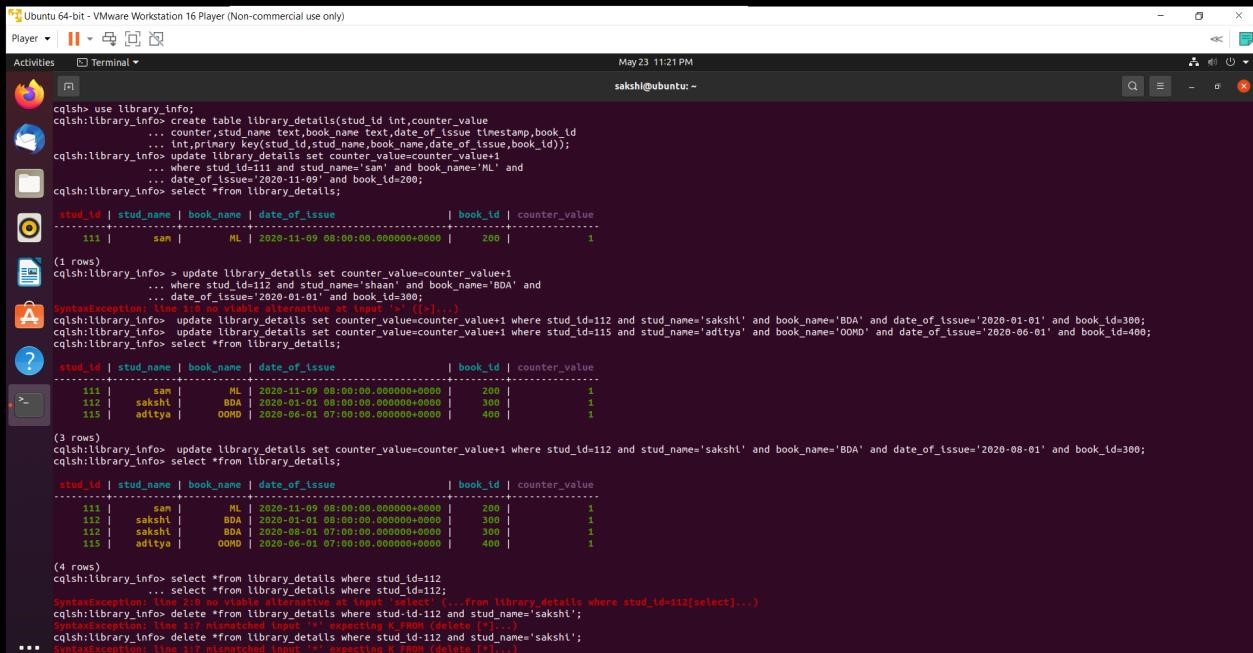
stud\_id | stud\_name | book\_name | date\_of\_issue | book\_id | counter\_value

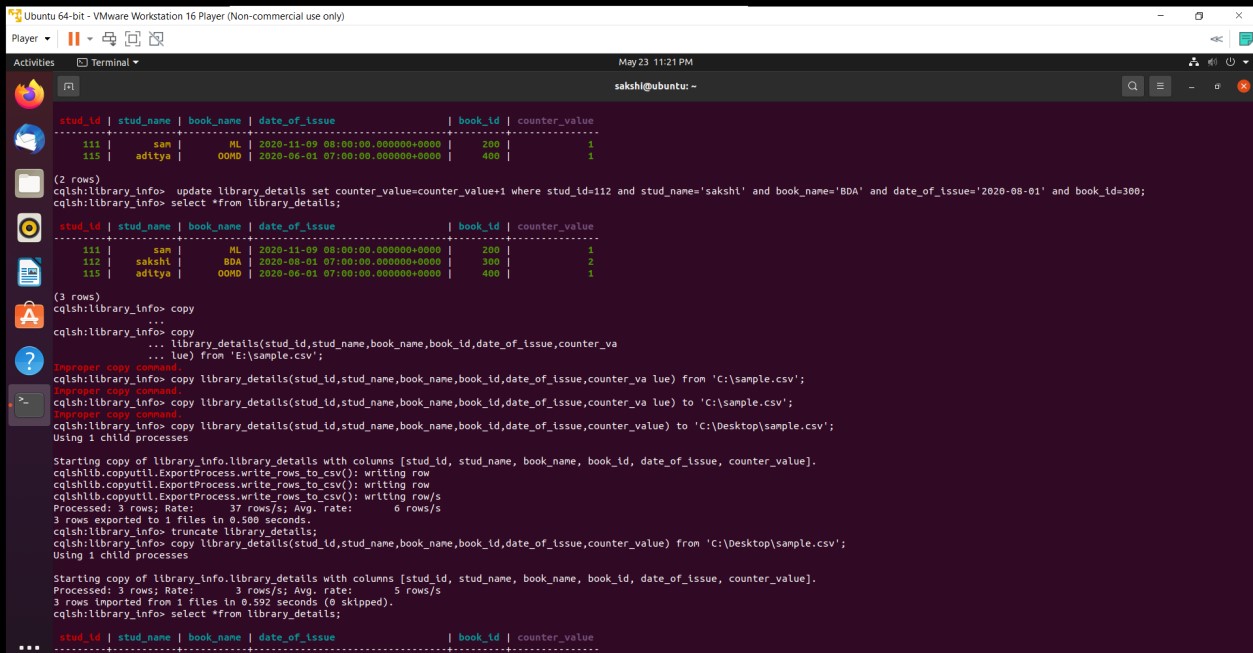
---------+-----------+-----------+---------------------------------+---------+---------------

1. | sam | ML | 2020-11-09 08:00:00.000000+0000 | 200 | 1
2. | sakshi | BDA | 2020-08-01 07:00:00.000000+0000 | 300 | 2

115 | aditya | OOMD | 2020-06-01 07:00:00.000000+0000 | 400 | 1

SCREENSHOTS:





# MONGODB SAMPLE

Question -

Perform the following DB operations using MongoDB.

1. Create a database “Student” with the following attributes Rollno, Age, ContactNo, EmailId.
2. Insert appropriate values
3. Write a 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 the local file system into mongodb collection.

> use students switched to db students

> db.createCollection("stud\_details")

{ "ok" : 1 }

>

db.stud\_details.insert({'name':'Sakshi','rollno':1,'age':19,'contactno':'8670794779','email':'sakshi@b msce.ac.in'})

WriteResult({ "nInserted" : 1 })

>

db.stud\_details.insert({'name':'Sneha','rollno':2,'age':20,'contactno':'8670794789','email':'sneha@b msce.ac.in'})

WriteResult({ "nInserted" : 1 })

>

db.stud\_details.insert({'name':'Shruti','rollno':3,'age':21,'contactno':'8630394789','email':'shruti@b msce.ac.in'})

WriteResult({ "nInserted" : 1 })

> db.stud\_details.find({})

{ "\_id" : ObjectId("60aaabf1b1aea56bb97beef8"), "name" : "Sakshi", "rollno" : 1, "age" : 19, "contactno" : "8670794779", "email" : "sakshi@bmsce.ac.in" }

{ "\_id" : ObjectId("60aaac16b1aea56bb97beef9"), "name" : "Sneha", "rollno" : 2, "age" : 20, "contactno" : "8670794789", "email" : "sneha@bmsce.ac.in" }

{ "\_id" : ObjectId("60aaac41b1aea56bb97beefa"), "name" : "Shruti", "rollno" : 3, "age" : 21, "contactno" : "8630394789", "email" : "shruti@bmsce.ac.in" }

> db.student\_details.update({'rollno':3},{$set:{'email':'update@lab.com'}})

WriteResult({ "nMatched" : 0, "nUpserted" : 0, "nModified" : 0 })

> db.stud\_details.find({'rollno':3})

{ "\_id" : ObjectId("60aaac41b1aea56bb97beefa"), "name" : "Shruti", "rollno" : 3, "age" : 21,

"contactno" : "8630394789", "email" : "shruti@bmsce.ac.in" }

mongoexport --db students --collection stud\_details --out C:\Desktop\sample.json

2021-05-22T10:43:30.687+0530 connected to: mongodb://localhost/ 2021-05-

22T10:43:31.026+0530 exported 3 records mongoimport --db students --collection stud\_details --type=json --file= C:\Desktop\sample.json

2021-05-22T10:46:49.898+0530 connected to: mongodb://localhost/ 2021-05-

22T10:46:50.044+0530 3 document(s) imported successfully. 0 document(s) failed to import. db.stud\_details.find({})

{ "\_id" : ObjectId("60aaabf1b1aea56bb97beef8"), "name" : "Sakshi", "rollno" : 1, "age" : 19, "contactno" : "8670794779", "email" : "sakshi@bmsce.ac.in" }

{ "\_id" : ObjectId("60aaac16b1aea56bb97beef9"), "name" : "Sneha", "rollno" : 2, "age" : 20, "contactno" : "8670794789", "email" : "sneha@bmsce.ac.in" }

{ "\_id" : ObjectId("60aaac41b1aea56bb97beefa"), "name" : "Shruti", "rollno" : 3, "age" : 21, "contactno" : "8630394789", "email" : "shruti@bmsce.ac.in" }

> db.student\_details.remove({age:{$gt:20}})

WriteResult({ "nRemoved" : 0 })

> db.stud\_details.find({})

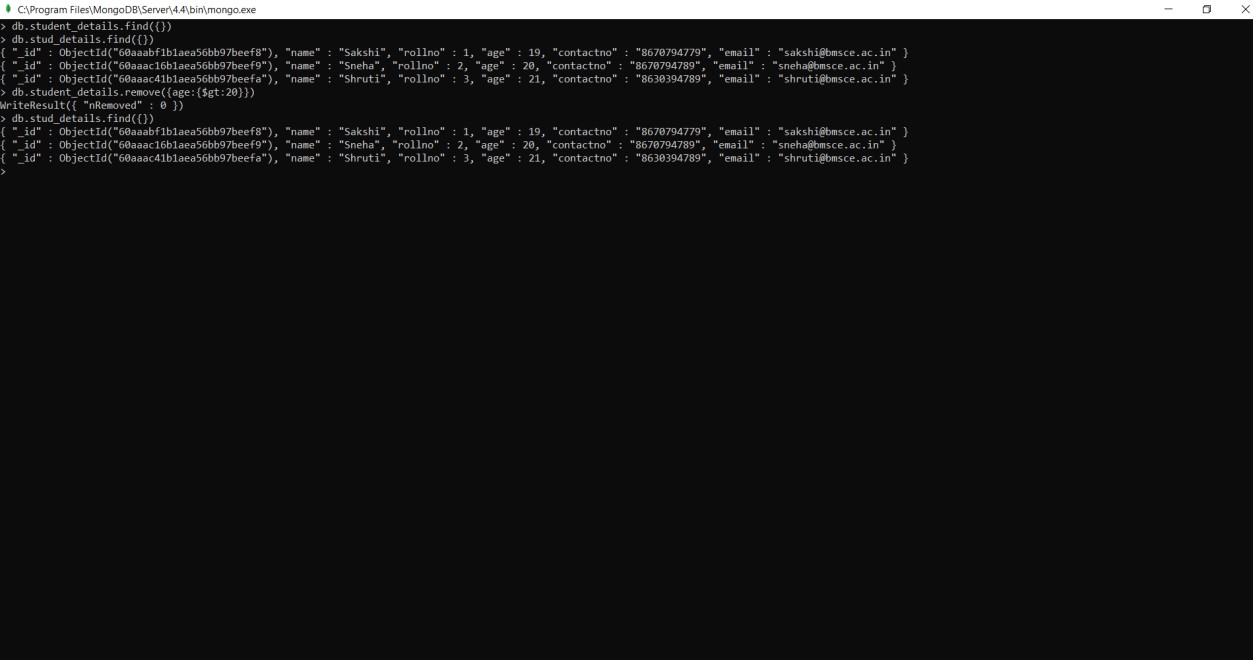
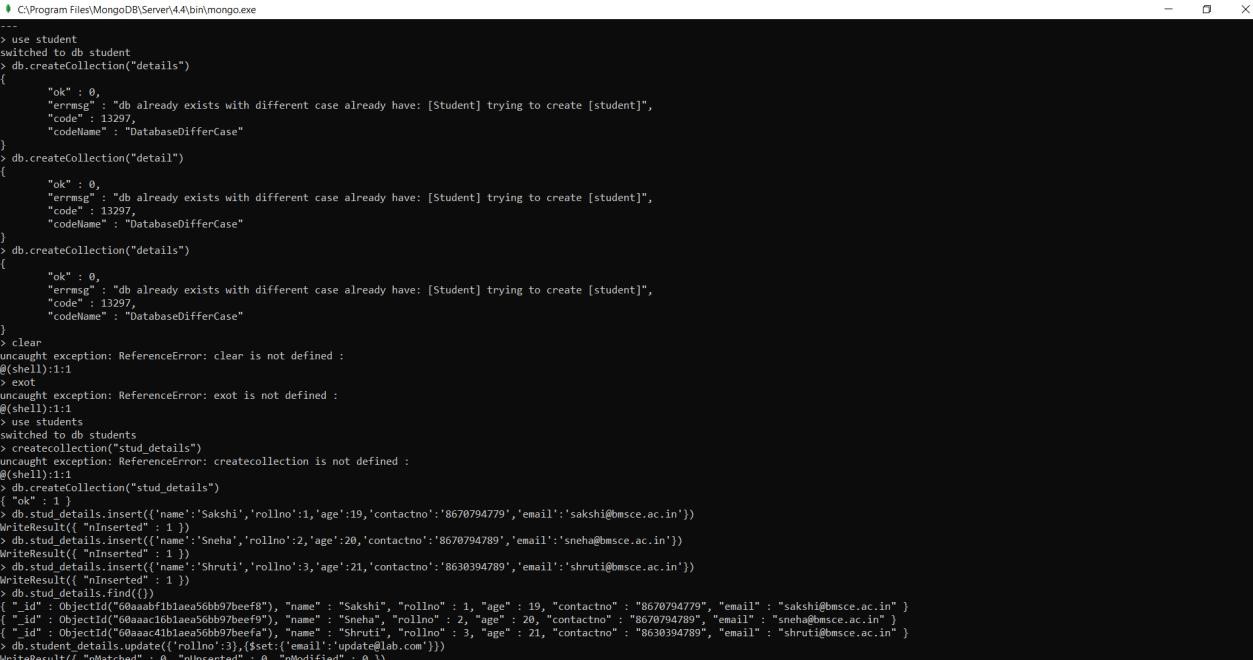
{ "\_id" : ObjectId("60aaabf1b1aea56bb97beef8"), "name" : "Sakshi", "rollno" : 1, "age" : 19, "contactno" : "8670794779", "email" : "sakshi@bmsce.ac.in" }

{ "\_id" : ObjectId("60aaac16b1aea56bb97beef9"), "name" : "Sneha", "rollno" : 2, "age" : 20, "contactno" : "8670794789", "email" : "sneha@bmsce.ac.in" }

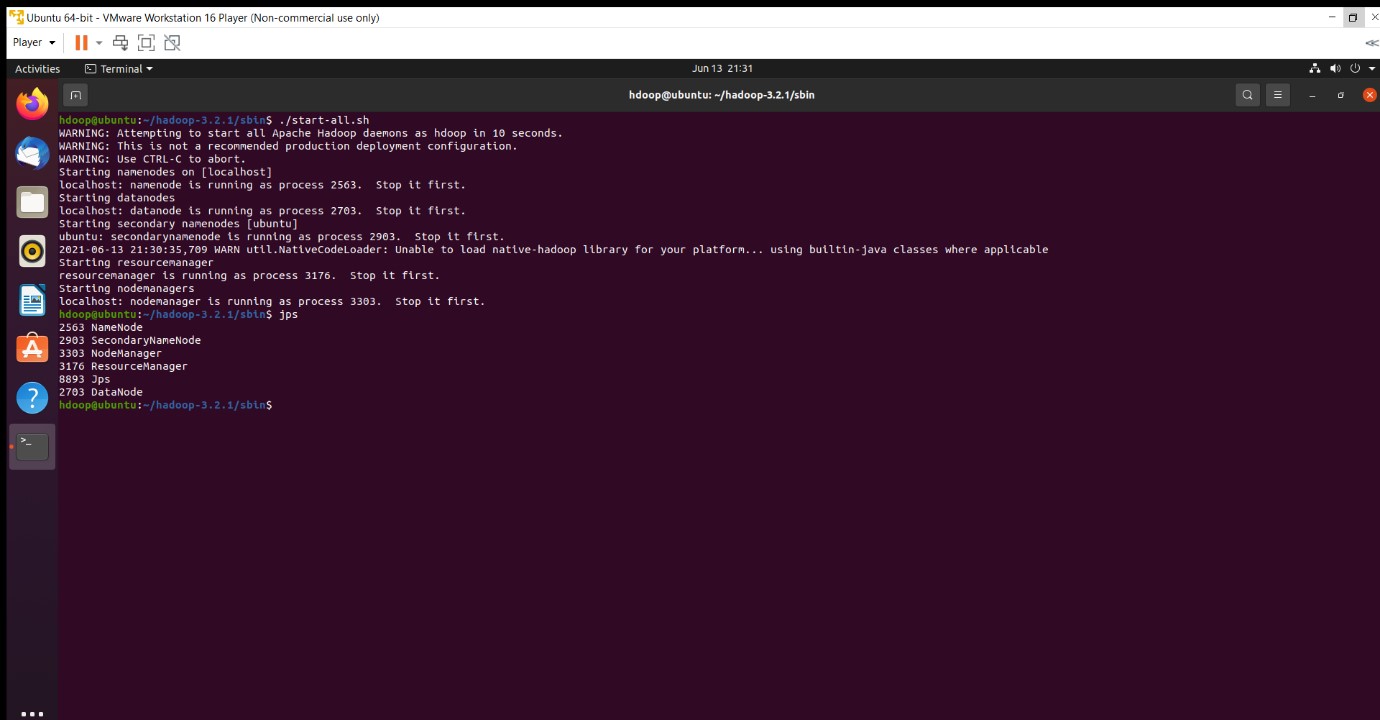
{ "\_id" : ObjectId("60aaac41b1aea56bb97beefa"), "name" : "Shruti", "rollno" : 3, "age" : 21,

"contactno" : "8630394789", "email" : "shruti@bmsce.ac.in" }

SCREENSHOTS:



# SCREENSHOT OF HADOOP INSTALLATION



# HADOOP SAMPLE

Execution of HDFS Commands for interaction with Hadoop Environment. (Minimum 10 commands to be executed)

c:\hadoop\_new\sbin>hdfs dfs -mkdir /temp

c:\hadoop\_new\sbin>hdfs dfs -copyFromLocal E:\Desktop\sample.txt \temp

c:\hadoop\_new\sbin>hdfs dfs -ls \temp

Found 1 items

-rw-r--r-- 1 Admin supergroup 11 2021-06-11 21:12 /temp/sample.txt c:\hadoop\_new\sbin>hdfs dfs -cat \temp\sample.txt hello world

c:\hadoop\_new\sbin>hdfs dfs -get \temp\sample.txt E:\Desktop\temp

c:\hadoop\_new\sbin>hdfs dfs -put E:\Desktop\temp \temp

c:\hadoop\_new\sbin>hdfs dfs -ls \temp

Found 2 items

-rw-r--r-- 1 Admin supergroup 11 2021-06-11 21:12 /temp/sample.txt drwxr-xr-x - Admin supergroup 0 2021-06-11 21:15 /temp/temp

c:\hadoop\_new\sbin>hdfs dfs -mv \lab1 \temp

c:\hadoop\_new\sbin>hdfs dfs -ls \temp Found 3 items drwxr-xr-x - Admin supergroup 0 2021-04-19 15:07 /temp/lab1 -rw-r--r-- 1 Admin supergroup 11 2021-06-11 21:12 /temp/sample.txt drwxr-xr-x - Admin supergroup 0 2021-06-11 21:15 /temp/temp

c:\hadoop\_new\sbin>hdfs dfs -rm /temp/sample.txt

Deleted /temp/sample.txt

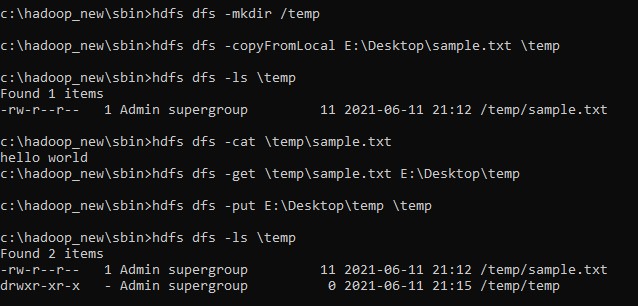
c:\hadoop\_new\sbin>hdfs dfs -ls \temp

Found 2 items drwxr-xr-x - Admin supergroup 0 2021-04-19 15:07 /temp/lab1 drwxr-xr-x - Admin supergroup 0 2021-06-11 21:15 /temp/temp

c:\hadoop\_new\sbin>hdfs dfs -copyFromLocal E:\Desktop\sample.txt \temp

c:\hadoop\_new\sbin>hdfs dfs -ls \temp Found 3 items drwxr-xr-x - Admin supergroup 0 2021-04-19 15:07 /temp/lab1 -rw-r--r-- 1 Admin supergroup 11 2021-06-11 21:17 /temp/sample.txt drwxr-xr-x - Admin supergroup 0 2021-06-11 21:15 /temp/temp

c:\hadoop\_new\sbin>hdfs dfs -copyToLocal \temp\sample.txt E:\Desktop\sample.txt SCREENSHOTS:



# MAPREDUCE TEMPERATURE

For the given file, Create a Map Reduce program to

a) Find the average temperature for each year from the NCDC data set.

// AverageDriver.java package temperature;

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 package temperature;

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 package temperature;

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;

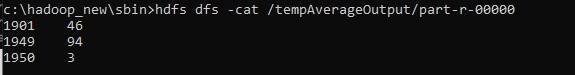
}

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

}

}

SCREENSHOT –



b) Find the mean max temperature for every month.

//TempDriver.java package temperatureMax; 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 TempDriver

{ 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(TempDriver.class); job.setJobName("Max temperature");

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

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

job.setMapperClass(TempMapper.class); job.setReducerClass(TempReducer.class); job.setOutputKeyClass(Text.class); job.setOutputValueClass(IntWritable.class);

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

}

}

//TempMapper.java package temperatureMax;

import org.apache.hadoop.io.\*; import org.apache.hadoop.mapreduce.\*; import java.io.IOException;

public class TempMapper 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 month = line.substring(19,21); 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(month),new IntWritable(temperature));

}

}

//TempReducer.java package temperatureMax;

import org.apache.hadoop.io.\*; import org.apache.hadoop.mapreduce.\*; import java.io.IOException;

public class TempMapper 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 month = line.substring(19,21); int temperature;

if (line.charAt(87)=='+') temperature = Integer.parseInt(line.substring(88, 92));

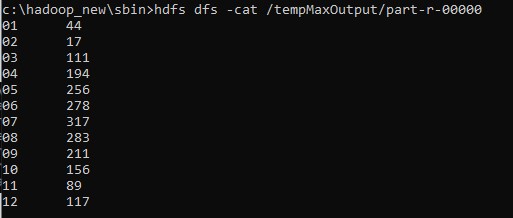
else

temperature = Integer.parseInt(line.substring(87, 92)); String quality = line.substring(92, 93); if(temperature != MISSING && quality.matches("[01459]")) context.write(new Text(month),new IntWritable(temperature));

}

}

SCREENSHOT -



# MAPREDUCE TOP-N

Date - 03/05/2021

For a given Text file, create a Map Reduce program to sort the content in an alphabetic order listing only top ‘n’ maximum occurrence of words.

// TopN.java package sortWords;

import org.apache.hadoop.conf.Configuration; import org.apache.hadoop.fs.Path; import org.apache.hadoop.io.IntWritable; import org.apache.hadoop.io.Text; import org.apache.hadoop.mapreduce.Job; import org.apache.hadoop.mapreduce.Mapper; import org.apache.hadoop.mapreduce.Reducer; import org.apache.hadoop.mapreduce.lib.input.FileInputFormat; import org.apache.hadoop.mapreduce.lib.output.FileOutputFormat; import org.apache.hadoop.util.GenericOptionsParser; import utils.MiscUtils;

import java.io.IOException; import java.util.\*;

public class TopN {

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

Configuration conf = new Configuration();

String[] otherArgs = new GenericOptionsParser(conf, args).getRemainingArgs(); if (otherArgs.length != 2) {

System.err.println("Usage: TopN <in> <out>");

System.exit(2);

}

Job job = Job.getInstance(conf); job.setJobName("Top N"); job.setJarByClass(TopN.class); job.setMapperClass(TopNMapper.class); //job.setCombinerClass(TopNReducer.class); job.setReducerClass(TopNReducer.class); job.setOutputKeyClass(Text.class); job.setOutputValueClass(IntWritable.class);

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

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

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

}

/\*\*

* The mapper reads one line at the time, splits it into an array of single words and emits every
* word to the reducers with the value of 1.

\*/

public static class TopNMapper extends Mapper<Object, Text, Text, IntWritable> {

private final static IntWritable one = new IntWritable(1); private Text word = new Text(); private String tokens = "[\_|$#<>\\^=\\[\\]\\\*/\\\\,;,.\\-:()?!\"']";

@Override public void map(Object key, Text value, Context context) throws IOException, InterruptedException {

String cleanLine = value.toString().toLowerCase().replaceAll(tokens, " "); StringTokenizer itr = new StringTokenizer(cleanLine); while (itr.hasMoreTokens()) { word.set(itr.nextToken().trim()); context.write(word, one);

}

}

}

/\*\*

\* The reducer retrieves every word and puts it into a Map: if the word already exists in the \* map, increments its value, otherwise sets it to 1.

\*/

public static class TopNReducer extends Reducer<Text, IntWritable, Text, IntWritable> { private Map<Text, IntWritable> countMap = new HashMap<>();

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

// computes the number of occurrences of a single word int sum = 0; for (IntWritable val : values) { sum += val.get();

}

// puts the number of occurrences of this word into the map.

// We need to create another Text object because the Text instance

// we receive is the same for all the words countMap.put(new Text(key), new IntWritable(sum));

}

@Override protected void cleanup(Context context) throws IOException, InterruptedException {

Map<Text, IntWritable> sortedMap = MiscUtils.sortByValues(countMap);

int counter = 0; for (Text key : sortedMap.keySet()) { if (counter++ == 3) { break;

}

context.write(key, sortedMap.get(key));

}

}

}

/\*\*

* The combiner retrieves every word and puts it into a Map: if the word already exists in the
* map, increments its value, otherwise sets it to 1.

\*/

public static class TopNCombiner extends Reducer<Text, IntWritable, Text, IntWritable> {

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

// computes the number of occurrences of a single word int sum = 0; for (IntWritable val : values) { sum += val.get();

}

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

}

}

}

// MiscUtils.java package utils;

import java.util.\*;

public class MiscUtils {

/\*\*

* sorts the map by values. Taken from:
* http://javarevisited.blogspot.it/2012/12/how-to-sort-hashmap-java-by-key-andvalue.html

\*/

public static <K extends Comparable, V extends Comparable> Map<K, V> sortByValues(Map<K, V> map) {

List<Map.Entry<K, V>> entries = new LinkedList<Map.Entry<K,

V>>(map.entrySet());

Collections.sort(entries, new Comparator<Map.Entry<K, V>>() {

@Override

public int compare(Map.Entry<K, V> o1, Map.Entry<K, V> o2) { return o2.getValue().compareTo(o1.getValue());

}

});

//LinkedHashMap will keep the keys in the order they are inserted

//which is currently sorted on natural ordering

Map<K, V> sortedMap = new LinkedHashMap<K, V>();

for (Map.Entry<K, V> entry : entries) { sortedMap.put(entry.getKey(), entry.getValue());

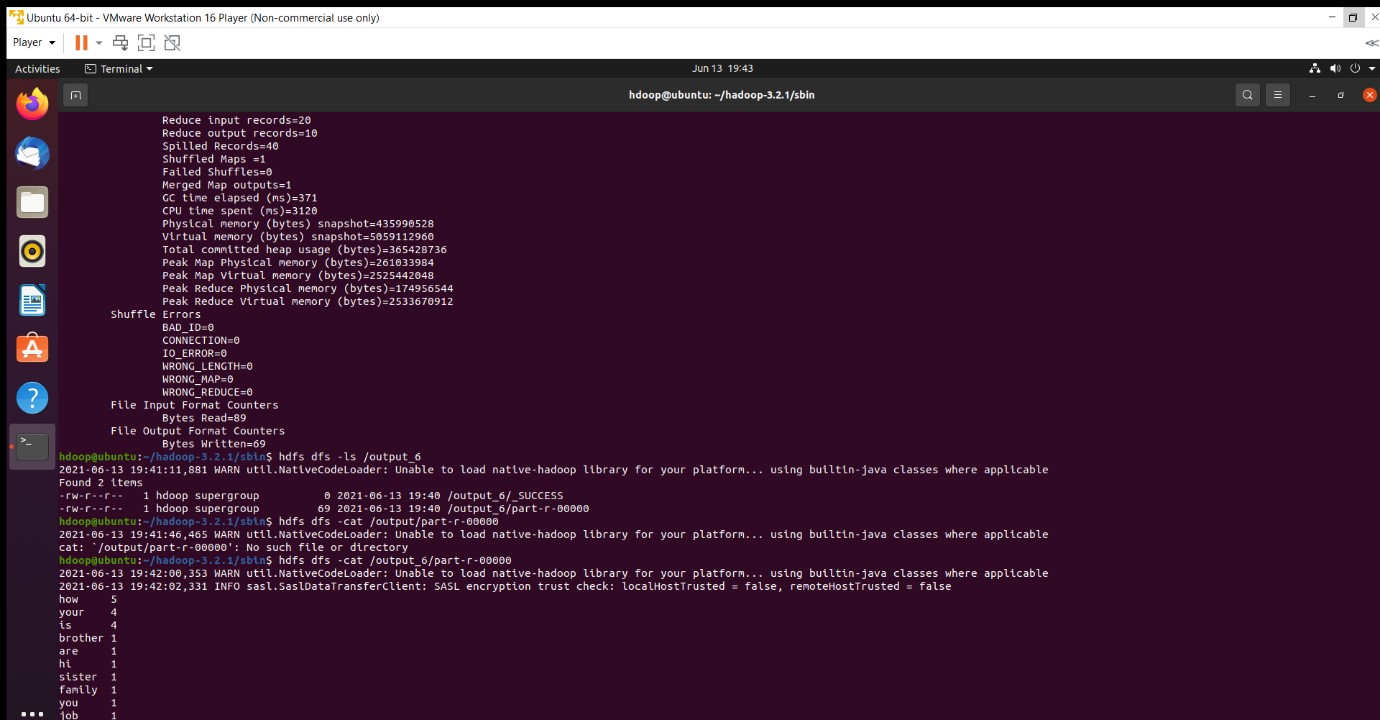
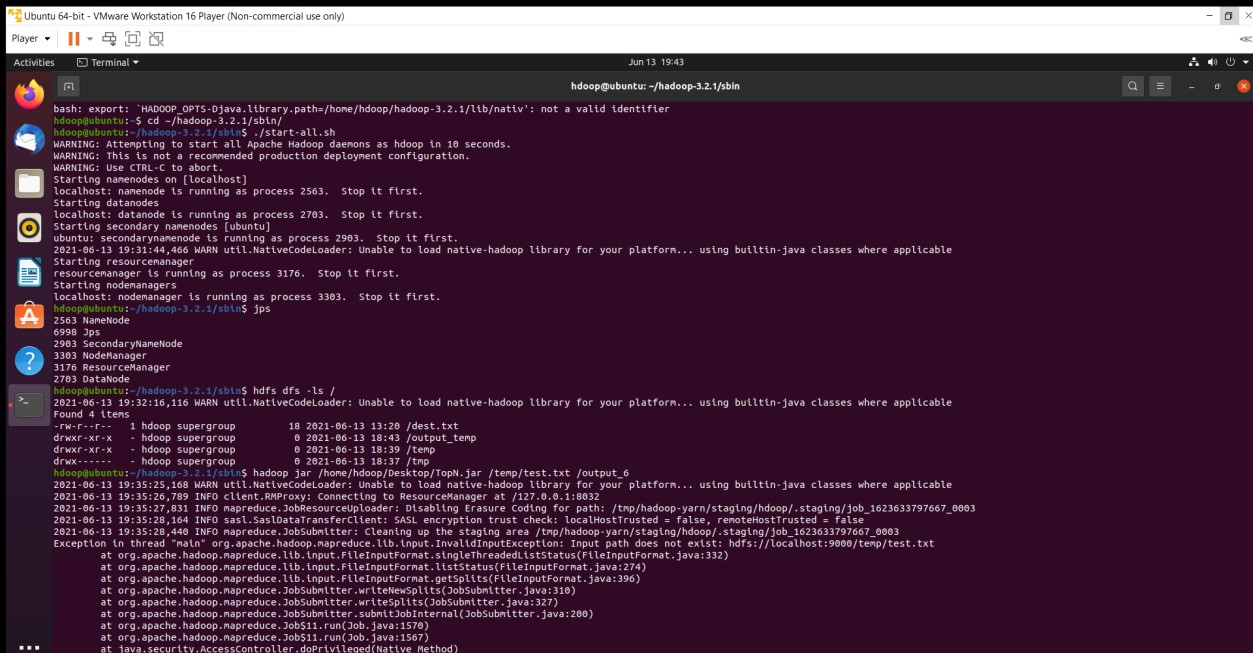
}

return sortedMap;

}

}

SCREENSHOTS:



# MAPREDUCE JOIN

Create a Hadoop Map Reduce program to combine information from the users file along with Information from the posts file by using the concept of join and display user\_id, Reputation and Score.

// JoinDriver.java

import org.apache.hadoop.conf.Configured; import org.apache.hadoop.fs.Path; import org.apache.hadoop.io.Text; import org.apache.hadoop.mapred.\*; import org.apache.hadoop.mapred.lib.MultipleInputs; import org.apache.hadoop.util.\*;

public class JoinDriver extends Configured implements Tool {

public static class KeyPartitioner implements Partitioner<TextPair, Text> {

@Override

public void configure(JobConf job) {}

@Override

public int getPartition(TextPair key, Text value, int numPartitions) { return (key.getFirst().hashCode() & Integer.MAX\_VALUE) % numPartitions;

}

}

@Override

public int run(String[] args) throws Exception {

if (args.length != 3) {

System.out.println("Usage: <Department Emp Strength input>

<Department Name input> <output>");

return -1;

}

JobConf conf = new JobConf(getConf(), getClass()); conf.setJobName("Join 'Department Emp Strength input' with 'Department Name input'");

|  |  |
| --- | --- |
|  | Path AInputPath = new Path(args[0]); |
|  | Path BInputPath = new Path(args[1]); |
|  | Path outputPath = new Path(args[2]); |
| Posts.class); | MultipleInputs.addInputPath(conf, AInputPath, TextInputFormat.class, |
| User.class); | MultipleInputs.addInputPath(conf, BInputPath, TextInputFormat.class, |
|  | FileOutputFormat.setOutputPath(conf, outputPath); |
|  | conf.setPartitionerClass(KeyPartitioner.class); |
|  | conf.setOutputValueGroupingComparator(TextPair.FirstComparator.class); |
|  | conf.setMapOutputKeyClass(TextPair.class); |

conf.setReducerClass(JoinReducer.class);

conf.setOutputKeyClass(Text.class);

JobClient.runJob(conf);

return 0;

}

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

int exitCode = ToolRunner.run(new JoinDriver(), args);

System.exit(exitCode);

}

}

// JoinReducer.java import java.io.IOException; import java.util.Iterator;

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

public class JoinReducer extends MapReduceBase implements Reducer<TextPair, Text,

Text, Text> {

@Override

public void reduce (TextPair key, Iterator<Text> values, OutputCollector<Text, Text> output, Reporter reporter)

throws IOException

{

Text nodeId = new Text(values.next()); while (values.hasNext()) {

Text node = values.next();

Text outValue = new Text(nodeId.toString() + "\t\t" + node.toString()); output.collect(key.getFirst(), outValue);

}

}

}

// User.java import java.io.IOException; import java.util.Iterator; import org.apache.hadoop.conf.Configuration; import org.apache.hadoop.fs.FSDataInputStream; import org.apache.hadoop.fs.FSDataOutputStream; import org.apache.hadoop.fs.FileSystem; import org.apache.hadoop.fs.Path; import org.apache.hadoop.io.LongWritable;

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

import org.apache.hadoop.io.IntWritable;

public class User extends MapReduceBase implements Mapper<LongWritable, Text, TextPair, Text> {

@Override

public void map(LongWritable key, Text value, OutputCollector<TextPair, Text> output, Reporter reporter)

throws IOException

{

String valueString = value.toString(); String[] SingleNodeData = valueString.split("\t"); output.collect(new TextPair(SingleNodeData[0], "1"), new Text(SingleNodeData[1]));

}

}

//Posts.java import java.io.IOException;

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

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

public class Posts extends MapReduceBase implements Mapper<LongWritable, Text, TextPair, Text> {

@Override

public void map(LongWritable key, Text value, OutputCollector<TextPair, Text> output, Reporter reporter)

throws IOException

{

String valueString = value.toString(); String[] SingleNodeData = valueString.split("\t"); output.collect(new TextPair(SingleNodeData[3], "0"), new Text(SingleNodeData[9]));

}

}

// TextPair.java import java.io.\*;

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

public class TextPair implements WritableComparable<TextPair> {

private Text first;

private Text second;

public TextPair() { set(new Text(), new Text());

}

public TextPair(String first, String second) { set(new Text(first), new Text(second));

}

public TextPair(Text first, Text second) { set(first, second);

}

public void set(Text first, Text second) {

this.first = first; this.second = second;

}

public Text getFirst() { return first;

}

public Text getSecond() {

return second;

}

@Override public void write(DataOutput out) throws IOException {

first.write(out); second.write(out);

}

@Override public void readFields(DataInput in) throws IOException { first.readFields(in); second.readFields(in);

}

@Override public int hashCode() { return first.hashCode() \* 163 + second.hashCode();

}

@Override public boolean equals(Object o) { if (o instanceof TextPair) {

TextPair tp = (TextPair) o;

return first.equals(tp.first) && second.equals(tp.second);

} return false;

}

@Override public String toString() { return first + "\t" + second;

}

@Override public int compareTo(TextPair tp) { int cmp = first.compareTo(tp.first); if (cmp != 0) { return cmp;

}

return second.compareTo(tp.second);

}

// ^^ TextPair

// vv TextPairComparator public static class Comparator extends WritableComparator {

private static final Text.Comparator TEXT\_COMPARATOR = new Text.Comparator(); public Comparator() { super(TextPair.class);

}

@Override public int compare(byte[] b1, int s1, int l1, byte[] b2, int s2, int l2) {

try {

int firstL1 = WritableUtils.decodeVIntSize(b1[s1]) + readVInt(b1, s1); int firstL2 = WritableUtils.decodeVIntSize(b2[s2]) + readVInt(b2, s2); int cmp = TEXT\_COMPARATOR.compare(b1, s1, firstL1, b2, s2, firstL2);

if (cmp != 0) { return cmp;

}

return TEXT\_COMPARATOR.compare(b1, s1 + firstL1, l1 - firstL1,

b2, s2 + firstL2, l2 - firstL2);

} catch (IOException e) { throw new IllegalArgumentException(e);

}

}

}

static {

WritableComparator.define(TextPair.class, new Comparator());

}

public static class FirstComparator extends WritableComparator {

private static final Text.Comparator TEXT\_COMPARATOR = new Text.Comparator();

public FirstComparator() { super(TextPair.class);

}

@Override public int compare(byte[] b1, int s1, int l1, byte[] b2, int s2, int l2) {

try {

int firstL1 = WritableUtils.decodeVIntSize(b1[s1]) + readVInt(b1, s1); int firstL2 = WritableUtils.decodeVIntSize(b2[s2]) + readVInt(b2, s2); return TEXT\_COMPARATOR.compare(b1, s1, firstL1, b2, s2, firstL2);

} catch (IOException e) { throw new IllegalArgumentException(e);

}

}

@Override public int compare(WritableComparable a, WritableComparable b) { if (a instanceof TextPair && b instanceof TextPair) { return ((TextPair) a).first.compareTo(((TextPair) b).first);

}

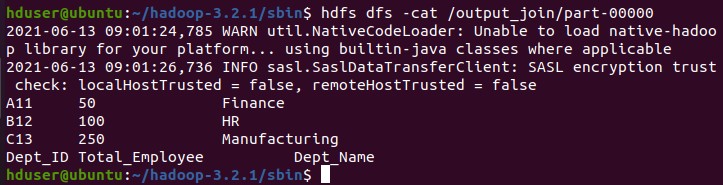
return super.compare(a, b);

}

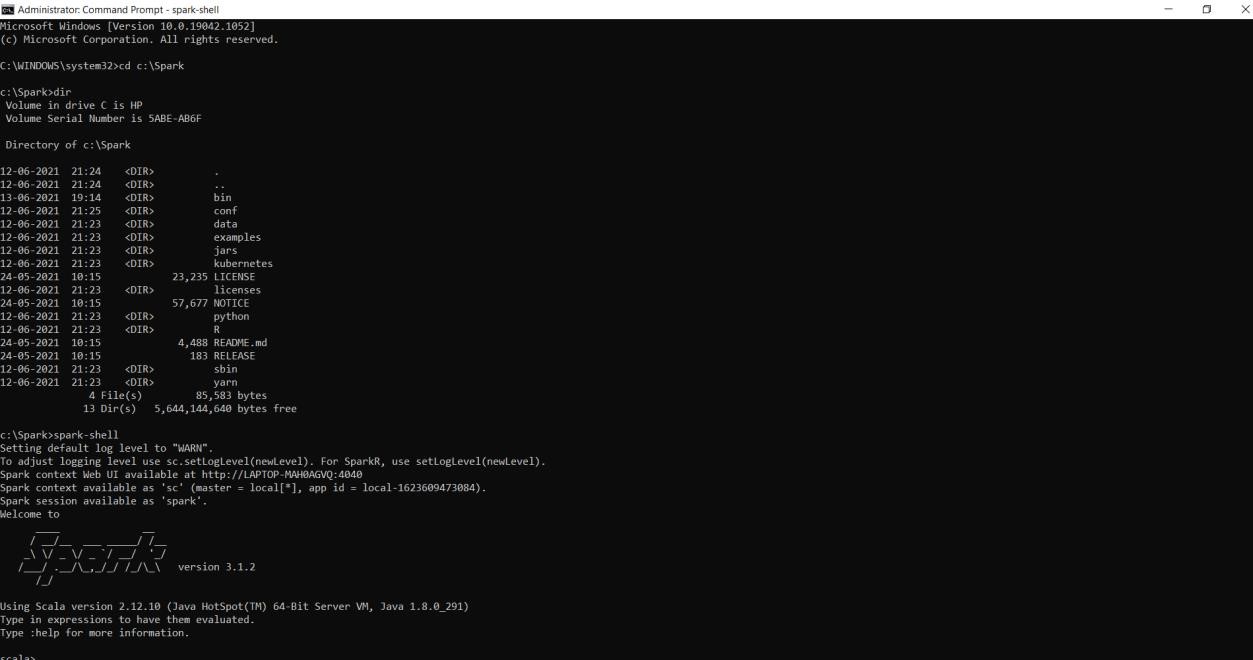
}

}

SCREENSHOTS –



**SCALA INSTALLATION SCREENSHOT**



# SCALA WORDCOUNT

package count

import org.apache.spark.SparkConf

import org.apache.spark.SparkContext

import org.apache.spark.rdd.RDD.rddToPairRDDFunctions

object count {

def

main(args: Array[String]) = {

//Start the Spark context

val conf = new SparkConf()

.setAppName("count")

.setMaster("local")

val sc = new SparkContext(conf)

//Read some example file to a test RDD

val test = sc.textFile("C:\\Spark\\spark-2.4.8-bin-hadoop2.7\\bin\\testdata\\sparkdata.txt")

test.flatMap {

line => //for each line

line.split(" ") //split the line in word by word.

}

.map {

word => //for each word

(word, 1) //Return a key/value tuple, with the word as key and 1 as value

}

.reduceByKey(\_ + \_) //Sum all of the value same key

.saveAsTextFile("C:\\Spark\\spark-2.4.8-bin-hadoop2.7\\bin\\testdata\\output2.txt") //Save to a text file

//Stop the Spark context

sc.stop}}

SCREENSHOTS:

