**VISVESVARAYA TECHNOLOGICAL UNIVERSITY**

**“JnanaSangama”, Belgaum -590014, Karnataka.**

****

**LAB REPORT**

**on**

**BIG DATA ANALYTICS**

***Submitted by***

**JIGAR D PATEL**

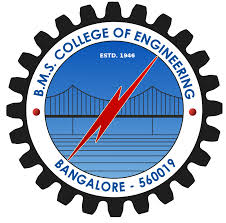
**(1BM21CS232)**

***in partial fulfillment for the award of the degree of***

**BACHELOR OF ENGINEERING**

***in***

**COMPUTER SCIENCE AND ENGINEERING**



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

**(Autonomous Institution under VTU)**

**BENGALURU-560019**

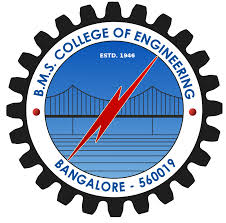
**Feb-2024 to July-2024**

**B. M. S. College of Engineering,**

**Bull Temple Road, Bangalore 560019**

(Affiliated To Visvesvaraya Technological University, Belgaum)

**Department of Computer Science and Engineering**



**CERTIFICATE**

This is to certify that the Lab work entitled “LAB COURSE **BIG DATA ANALYTICS**” carried out by **JIGAR D PATEL (1BM21CS081),** who is bonafide student of **B. M. S. College of Engineering.** It is in partial fulfillment for the award of **Bachelor of Engineering in Computer Science and Engineering** of the Visvesvaraya Technological University, Belgaum during the year 2024. The Lab report has been approved as it satisfies the academic requirements in respect of a **BIG DATA ANALYTICS - (22CS6PEBDA)** work prescribed for the said degree.

**Vikaranth B.M** **Dr. Jyothi S Nayak**

Assistant Professor Professor and Head

Department of CSE Department of CSE

BMSCE, Bengaluru BMSCE, Bengaluru

**Index Sheet**

|  |  |  |
| --- | --- | --- |
| **Sl. No.** | **Experiment Title** | **Page No.** |
| **1** | **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**  **4. Update Employee name and Department of Emp-Id 121**  **5. Sort the details of Employee records based on 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.**  **7. Update the altered table to add project names.**  **8. Create a TTL of 15 seconds to display the values of Employees.** | **1 - 3** |
| **2** | **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**  **4. Display the details of the table created and increase the value of the counter**  **5. Write a query to show that a student with id 112 has taken a book “BDA” 2 times.**  **6. Export the created column to a csv file**  **7. Import a given csv dataset from local file system into Cassandra column family** | **4 - 6** |
| **3** | **MongoDB- CRUD Demonstration** | **7 - 9** |
| **4** | **Screenshot of Hadoop installed** | **10** |
| **5** | **Execution of HDFS Commands for interaction with Hadoop Environment. (Minimum 10 commands to be executed)** | **11 – 12** |
| **6** | **Implement WordCount Program on Hadoop framework** | **13 - 15** |
| **7** | **From the following link extract the weather data https://github.com/tomwhite/hadoop-book/tree/master/input/ncdc/all**  **Create a Map Reduce program to**  **a) find average temperature for each year from NCDC data set.**  **b) find the mean max temperature for every month** | **16 - 23** |
| **8** | **For a given Text file, create a Map Reduce program to sort the content in an alphabetic order listing only top 10 maximum occurrences of words.** | **24 - 29** |

**Course Outcome**

|  |  |
| --- | --- |
| CO1 | Apply the concepts of NoSQL, Hadoop, Spark for a given task |
| CO2 | Analyse data analytic techniques for a given problem |
| CO3 | Conduct experiments using data analytics mechanisms for a given problem. |

**Program 1**

**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**

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

**5. Sort the details of Employee records based on 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.**

**7. Update the altered table to add project names.**

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

1. Create a keyspace by name Employee

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

1. Create a column family by name Employee-Info

CREATE TABLE Employee.Employee\_Info (

Emp\_Id int PRIMARY KEY,

Emp\_Name text,

Designation text,

Date\_of\_Joining date,

Salary decimal,

Dept\_Name text

);

1. Insert the values into the table in batch

BEGIN BATCH

INSERT INTO Employee.Employee\_Info (Emp\_Id, Emp\_Name, Designation, Date\_of\_Joining, Salary, Dept\_Name) VALUES (121, 'John Doe', 'Software Engineer', '2022-01-15', 70000.00, 'IT');

INSERT INTO Employee.Employee\_Info (Emp\_Id, Emp\_Name, Designation, Date\_of\_Joining, Salary, Dept\_Name) VALUES (122, 'Jane Smith', 'Data Scientist', '2021-05-20', 80000.00, 'Data Science');

INSERT INTO Employee.Employee\_Info (Emp\_Id, Emp\_Name, Designation, Date\_of\_Joining, Salary, Dept\_Name) VALUES (123, 'Alice Johnson', 'Project Manager', '2020-07-18', 90000.00, 'Management');

APPLY BATCH;

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

UPDATE Employee.Employee\_Info SET Emp\_Name = 'Johnathon Doe', Dept\_Name =

'Software Development' WHERE Emp\_Id = 121;

1. Sort the details of Employee records based on salary

CREATE INDEX ON Employee.Employee\_Info (Salary);

1. Alter the schema of the table Employee\_Info to add a column Projects

ALTER TABLE Employee.Employee\_Info ADD Projects set<text>;

1. Update the altered table to add project names

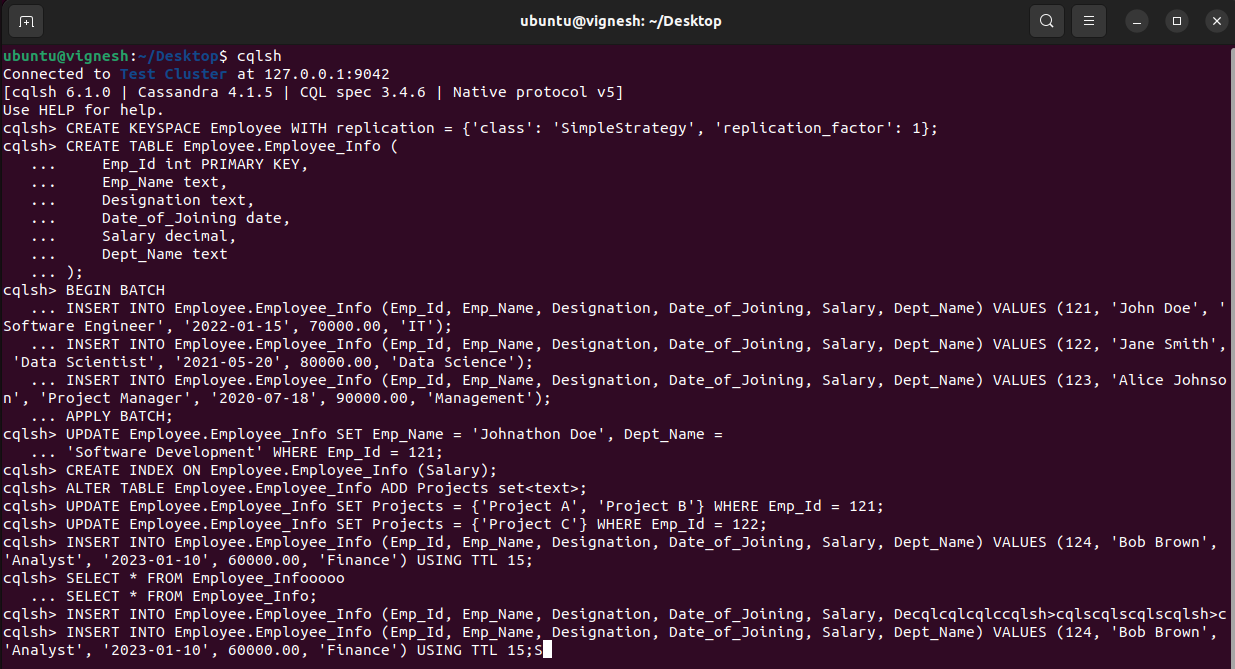
UPDATE Employee.Employee\_Info SET Projects = {'Project A', 'Project B'} WHERE Emp\_Id = 121;

UPDATE Employee.Employee\_Info SET Projects = {'Project C'} WHERE Emp\_Id = 122;

UPDATE Employee.Employee\_Info SET Projects = {'Project D', 'Project E'} WHERE Emp\_Id = 123;

1. Create a TTL of 15 seconds to display the values of Employeee

INSERT INTO Employee.Employee\_Info (Emp\_Id, Emp\_Name, Designation, Date\_of\_Joining, Salary, Dept\_Name) VALUES (124, 'Bob Brown', 'Analyst', '2023-01-10', 60000.00, 'Finance') USING TTL 15;



**Program 2**

**Perform the following DB operations using Cassandra.**

1. **Create a keyspace by name Library**

CREATE KEYSPACE Library WITH replication = { 'class' : 'SimpleStrategy', 'replication\_factor' : 3 };

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**

USE Library;

CREATE TABLE Library\_Info (

Stud\_Id int PRIMARY KEY,

Counter\_value counter,

Stud\_Name text,

Book\_Name text,

Book\_Id text,

Date\_of\_issue timestamp

);

1. **Insert the values into the table in batch**

BEGIN BATCH;

INSERT INTO Library\_Info (Stud\_Id, Counter\_value, Stud\_Name, Book\_Name, Book\_Id, Date\_of\_issue)

VALUES (1, 101, 'Alice Smith', 'Introduction to Algorithms', 'B001', '2024-05-01');

INSERT INTO Library\_Info (Stud\_Id, Counter\_value, Stud\_Name, Book\_Name, Book\_Id, Date\_of\_issue)

VALUES (2, 102, 'Bob Johnson', 'Clean Code', 'B002', '2024-05-02');

INSERT INTO Library\_Info (Stud\_Id, Counter\_value, Stud\_Name, Book\_Name, Book\_Id, Date\_of\_issue)

VALUES (3, 103, 'Charlie Brown', 'Design Patterns', 'B003', '2024-05-03');

INSERT INTO Library\_Info (Stud\_Id, Counter\_value, Stud\_Name, Book\_Name, Book\_Id, Date\_of\_issue)

VALUES (4, 104, 'Diana Prince', 'The Pragmatic Programmer', 'B004', '2024-05-04');

INSERT INTO Library\_Info (Stud\_Id, Counter\_value, Stud\_Name, Book\_Name, Book\_Id, Date\_of\_issue)

VALUES (5, 105, 'Ethan Hunt', 'Effective Java', 'B005', '2024-05-05');

APPLY BATCH;

1. **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 Stud\_Id = 111;

SELECT \* FROM Library\_Info;

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

SELECT Stud\_Name, Book\_Name, Counter\_value FROM Library\_Info

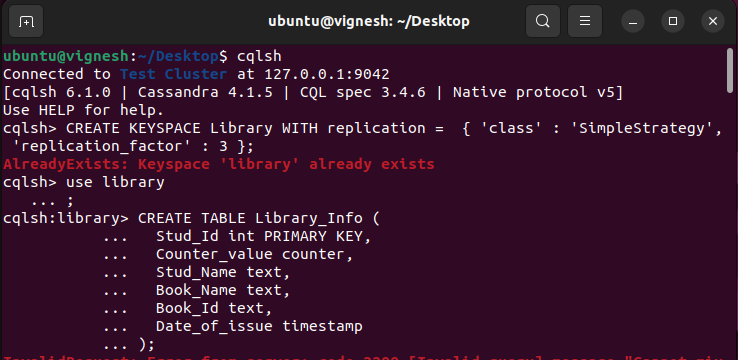
WHERE Stud\_Id = 112 AND Book\_Name = 'BDA';

1. **Export the created column to a csv file**

COPY Library\_Info TO '/path/to/<lib\_info>.csv' WITH DELIMITER = ',' QUOTE = '"' HEADER = TRUE;

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

COPY Library\_Info FROM '/path/to/<filename>.csv' WITH DELIMITER = ',' QUOTE = '"' HEADER = TRUE;



**Program 3**

**MongoDB- CRUD Demonstration**

**I. Perform the following DB operations using MongoDB.**

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

> use StudentDB

2. Insert appropriate values

> db.students.insertMany([

{ Rollno: 1, Age: 20, ContactNo: "1234567890", EmailId: "student1@example.com" },

{ Rollno: 2, Age: 21, ContactNo: "1234567891", EmailId: "student2@example.com" },

{ Rollno: 10, Age: 22, ContactNo: "1234567892", EmailId: "student10@example.com" },

{ Rollno: 11, Age: 23, ContactNo: "1234567893", EmailId: "student11@example.com", Name: "ABC" }

])

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

> db.students.updateOne(

{ Rollno: 10 },

{ $set: { EmailId: "newemail10@example.com" } }

)

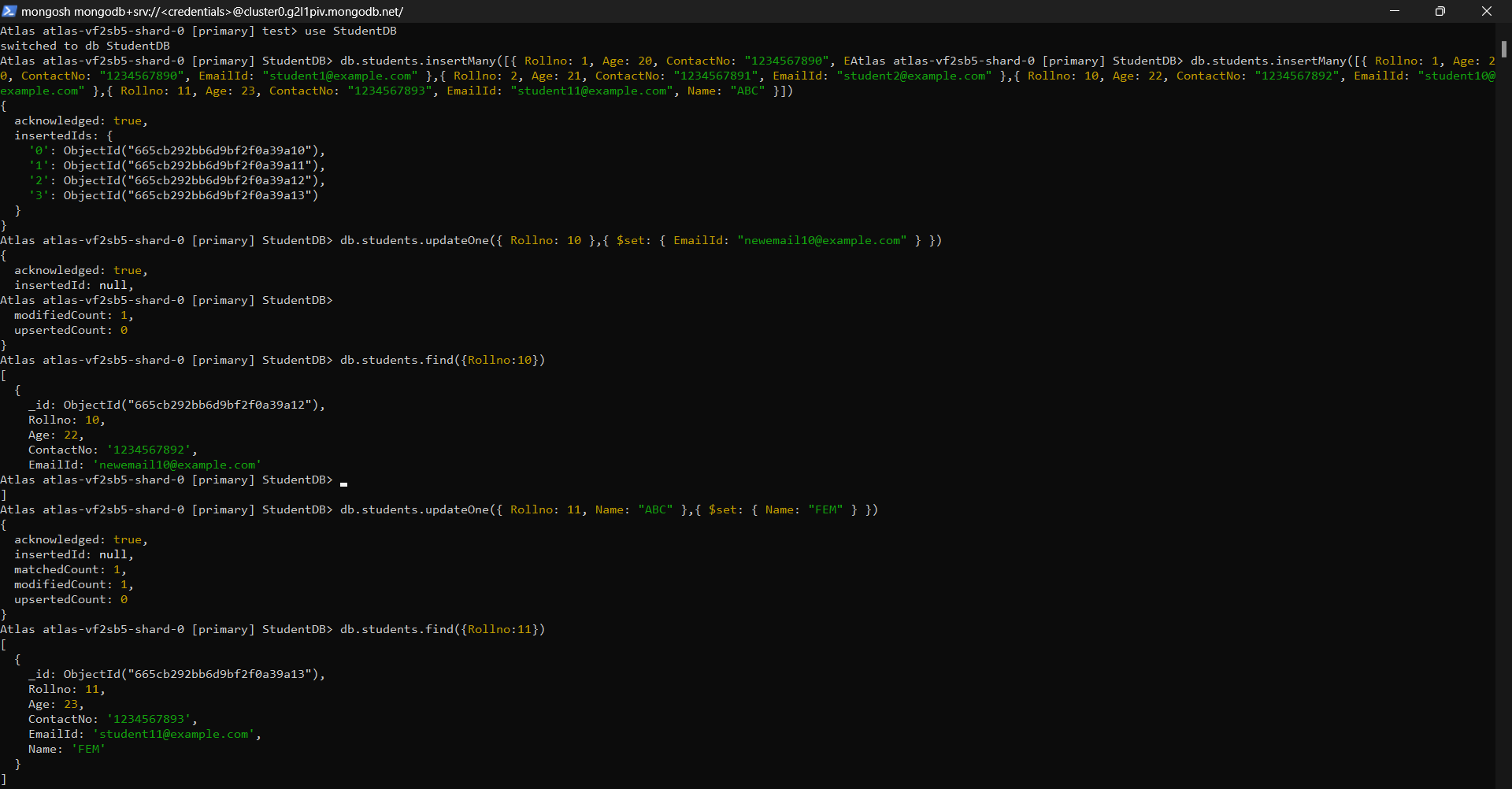
4. Replace the student’s name from “ABC” to “FEM” of rollno 11

> db.students.updateOne(

{ Rollno: 11, Name: "ABC" },

{ $set: { Name: "FEM" } }

)



**II. Perform the following DB operations using MongoDB.**

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

use Bank;

db.Customers.insertOne({

Cust\_id: 1,

Acc\_Bal: 1000,

Acc\_Type: "A"

});

1. Insert at least 5 values into the table

> use CustomerDB

db.customers.insertMany([

{ Cust\_id: 1, Acc\_Bal: 1500, Acc\_Type: 'Z' },

{ Cust\_id: 2, Acc\_Bal: 800, Acc\_Type: 'Y' },

{ Cust\_id: 3, Acc\_Bal: 2000, Acc\_Type: 'Z' },

{ Cust\_id: 4, Acc\_Bal: 1000, Acc\_Type: 'X' },

{ Cust\_id: 5, Acc\_Bal: 1300, Acc\_Type: 'Z' }

])

1. Write a query to display those records whose total account balance is greater than 1200 of account type ‘Z’ for each customer\_id.

db.Customers.find({

Acc\_Type: "Z",

Acc\_Bal: { $gt: 1200 }

});

1. Determine Minimum and Maximum account balance for each customer\_i

db.Customers.aggregate([

{

$group: {

\_id: "$Cust\_id",

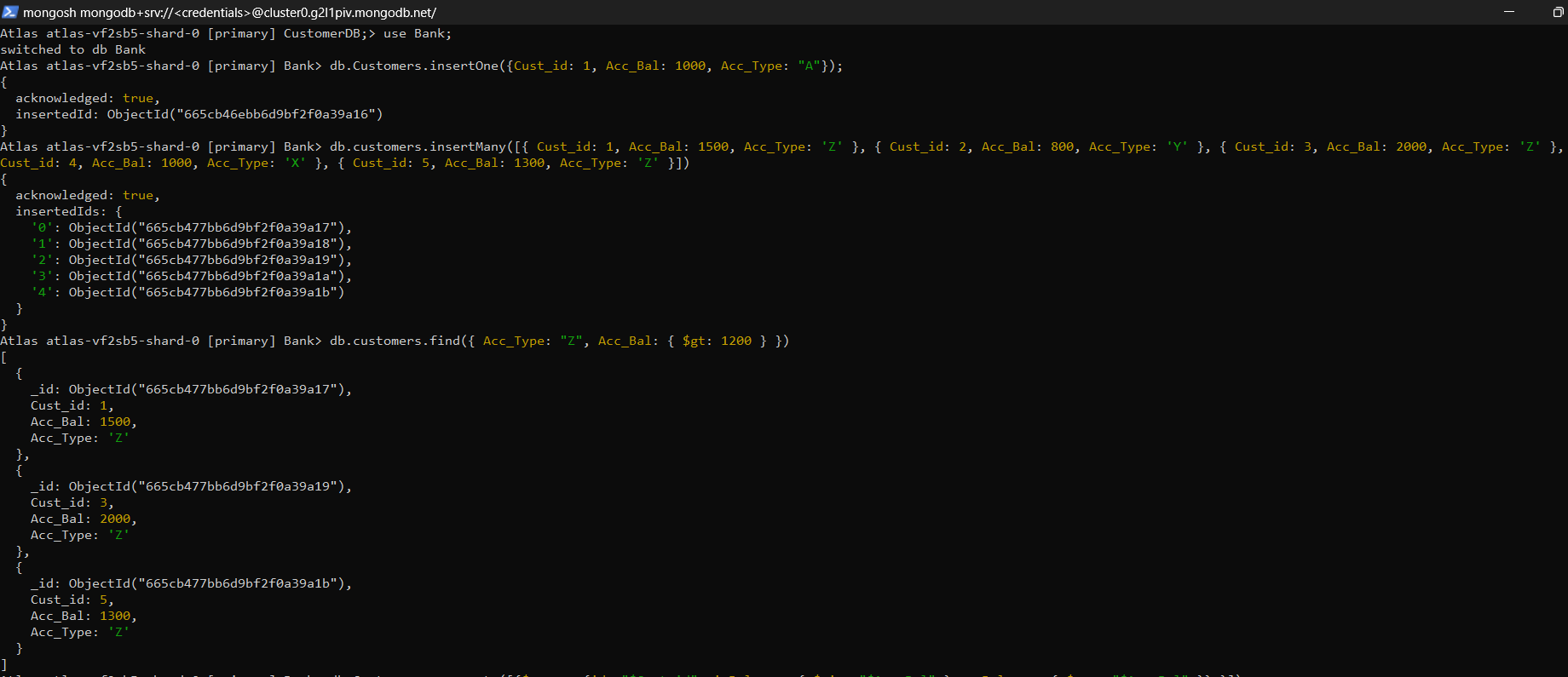
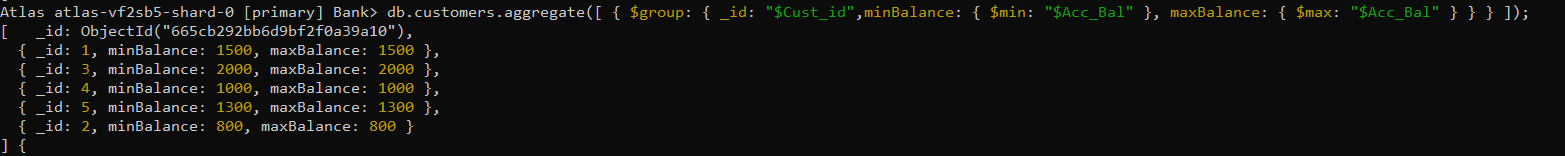
minBalance: { $min: "$Acc\_Bal" },

maxBalance: { $max: "$Acc\_Bal" }

}

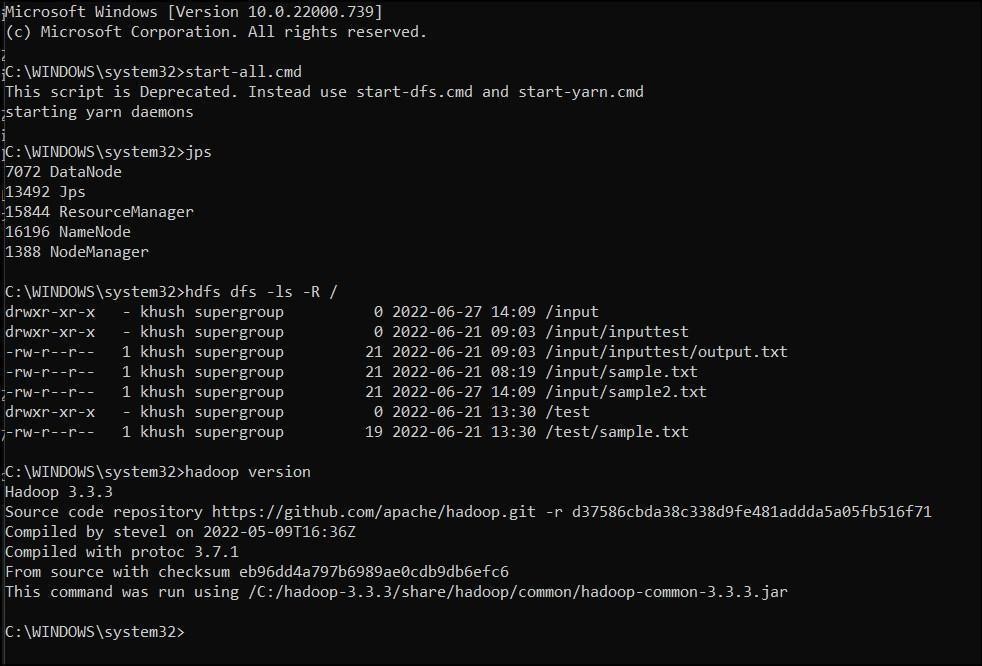
}

]);

**Program 4**

**Screenshot of Hadoop installed**



**Program 5**

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

hadoop@bmscecse-HP-Elite-Tower-800-G9-Desktop-PC:~$ start-all.sh

WARNING: Attempting to start all Apache Hadoop daemons as hadoop in 10 seconds.

WARNING: This is not a recommended production deployment configuration.

WARNING: Use CTRL-C to abort.

Starting namenodes on [localhost]

Starting datanodes

Starting secondary namenodes [bmscecse-HP-Elite-Tower-800-G9-Desktop-PC]

Starting resourcemanager

Starting nodemanagers

hadoop@bmscecse-HP-Elite-Tower-800-G9-Desktop-PC:~$ hadoop dfs -mkdir /sadh

WARNING: Use of this script to execute dfs is deprecated.

WARNING: Attempting to execute replacement "hdfs dfs" instead.

hadoop@bmscecse-HP-Elite-Tower-800-G9-Desktop-PC:~$ hdfs dfs -mkdir /sadh

mkdir: `/sadh': File exists

hadoop@bmscecse-HP-Elite-Tower-800-G9-Desktop-PC:~$ hadoop fs -ls /

Found 1 items

drwxr-xr-x - hadoop supergroup 0 2024-05-13 14:27 /sadh

hadoop@bmscecse-HP-Elite-Tower-800-G9-Desktop-PC:~$ hadoop fs -ls /sadh

hadoop@bmscecse-HP-Elite-Tower-800-G9-Desktop-PC:~$ hdfs dfs -put /home/hadoop/Desktop/example/Welcome.txt /sadh/WC.txt

hadoop@bmscecse-HP-Elite-Tower-800-G9-Desktop-PC:~$ hdfs dfs -cat /sadh/WC.txt

hiiii

hadoop@bmscecse-HP-Elite-Tower-800-G9-Desktop-PC:~$ hdfs dfs -get /sadh/WC.txt /home/hadoop/Desktop/example/WWC.txt

hadoop@bmscecse-HP-Elite-Tower-800-G9-Desktop-PC:~$ hdfs dfs -get /sadh/WC.txt /home/hadoop/Desktop/example/WWC2.txt

hadoop@bmscecse-HP-Elite-Tower-800-G9-Desktop-PC:~$ hdfs dfs -put /home/hadoop/Desktop/example/Welcome.txt /sadh/WC2.txt

hadoop@bmscecse-HP-Elite-Tower-800-G9-Desktop-PC:~$ hdfs dfs -getmerge /sadh/WC.txt /sadh/WC2.txt /home/hadoop/Desktop/example/Merge.txt

hadoop@bmscecse-HP-Elite-Tower-800-G9-Desktop-PC:~$ hadoop fs -getfacl /sadh/

# file: /sadh

# owner: hadoop

# group: supergroup

user::rwx

group::r-x

other::r-x

hadoop@bmscecse-HP-Elite-Tower-800-G9-Desktop-PC:~$ hadoop fs -mv /sadh /WC2.txt

hadoop@bmscecse-HP-Elite-Tower-800-G9-Desktop-PC:~$ hadoop fs -ls /sadh /WC2.txt

ls: `/sadh': No such file or directory

Found 2 items

-rw-r--r-- 1 hadoop supergroup 6 2024-05-13 14:51 /WC2.txt/WC.txt

-rw-r--r-- 1 hadoop supergroup 6 2024-05-13 15:03 /WC2.txt/WC2.txt

hadoop@bmscecse-HP-Elite-Tower-800-G9-Desktop-PC:~$ hadoop fs -cp /WC2.txt/ /WC.txt

**Program 6**

**Implement WordCount Program on Hadoop framework**

Mapper Code:

import java.io.IOException;

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 WCMapper extends MapReduceBase implements Mapper<LongWritable,

Text, Text,

IntWritable> {

public void map(LongWritable key, Text value, OutputCollector<Text,

IntWritable> output, Reporter rep) throws IOException

{

String line = value.toString();

for (String word : line.split(" "))

{

if (word.length() > 0)

{

output.collect(new Text(word), new IntWritable(1));

} } } }

Reducer Code:

// Importing libraries

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 WCReducer extends MapReduceBase implements Reducer<Text,

IntWritable, Text, IntWritable> {

// Reduce function

public void reduce(Text key, Iterator<IntWritable> value,

OutputCollector<Text, IntWritable> output,

Reporter rep) throws IOException

{

int count = 0;

// Counting the frequency of each words

while (value.hasNext())

{

IntWritable i = value.next();

count += i.get();

}

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

} }

Driver Code: You have to copy paste this program into the WCDriver Java Class file.

// Importing libraries

import java.io.IOException;

import org.apache.hadoop.conf.Configured;

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.util.Tool;

import org.apache.hadoop.util.ToolRunner;

public class WCDriver extends Configured implements Tool {

public int run(String args[]) throws IOException

{

if (args.length < 2)

{

System.out.println("Please give valid inputs");

return -1;

}

JobConf conf = new JobConf(WCDriver.class);

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

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

conf.setMapperClass(WCMapper.class);

conf.setReducerClass(WCReducer.class);

conf.setMapOutputKeyClass(Text.class);

conf.setMapOutputValueClass(IntWritable.class);

conf.setOutputKeyClass(Text.class);

conf.setOutputValueClass(IntWritable.class);

JobClient.runJob(conf);

return 0;

}

// Main Method

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

{

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

System.out.println(exitCode);

}

}

**Program 7**

**From the following link extract the weather data https://github.com/tomwhite/hadoop-book/tree/master/input/ncdc/all**

**Create a Map Reduce program to**

1. **find average temperature for each year from NCDC data set.**

**AverageDriver**

package temp;

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.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**

package temp;

import java.io.IOException;

import org.apache.hadoop.io.IntWritable;

import org.apache.hadoop.io.LongWritable;

import org.apache.hadoop.io.Text;

import org.apache.hadoop.mapreduce.Mapper;

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

public static final int MISSING = 9999;

public void map(LongWritable key, Text value, Mapper<LongWritable, Text, Text,

IntWritable>.Context context) throws IOException, InterruptedException {

int temperature;

String line = value.toString();

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

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 != 9999 && quality.matches("[01459]"))

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

}

}

AverageReducer

package temp;

import java.io.IOException;

import org.apache.hadoop.io.IntWritable;

import org.apache.hadoop.io.Text;

import org.apache.hadoop.mapreduce.Reducer;

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

public void reduce(Text key, Iterable<IntWritable> values, Reducer<Text, IntWritable,

Text, IntWritable>.Context context) throws IOException, InterruptedException {

int max\_temp = 0;

int count = 0;

for (IntWritable value : values) {

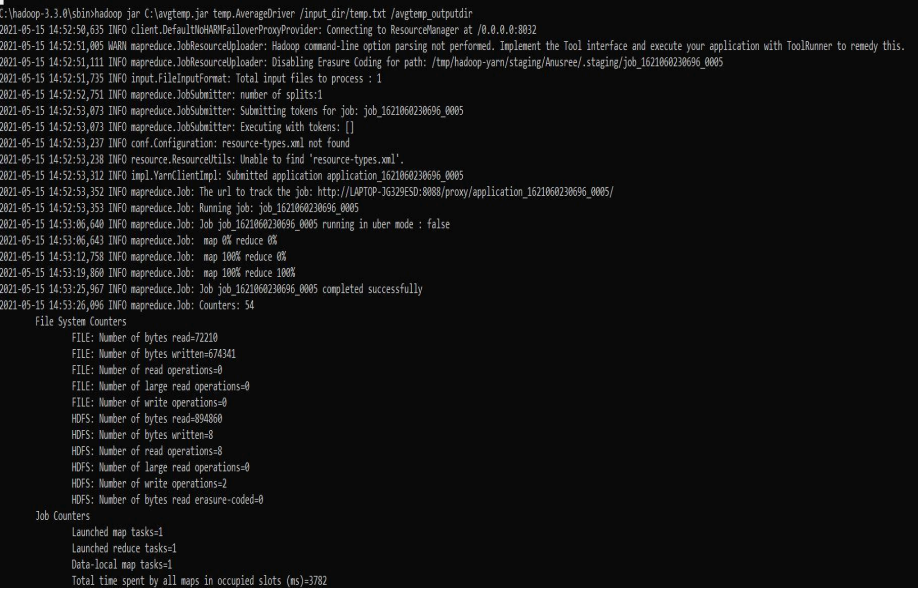
max\_temp += value.get();

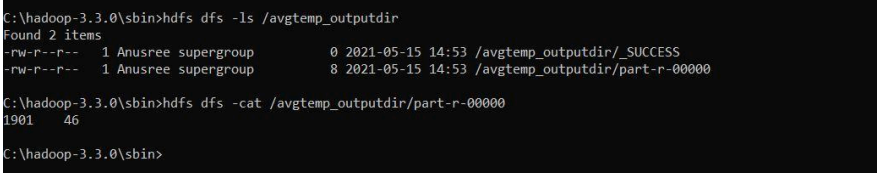
count++;

}

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

}}





1. **find the mean max temperature for every month**

**MeanMaxDriver.class**

package meanmax;

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.lib.input.FileInputFormat;

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

public class MeanMaxDriver {

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(MeanMaxDriver.class);

job.setJobName("Max temperature");

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

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

job.setMapperClass(MeanMaxMapper.class);

job.setReducerClass(MeanMaxReducer.class);

job.setOutputKeyClass(Text.class);

job.setOutputValueClass(IntWritable.class);

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

}

}

**MeanMaxMapper.class**

package meanmax;

import java.io.IOException;

import org.apache.hadoop.io.IntWritable;

import org.apache.hadoop.io.LongWritable;

import org.apache.hadoop.io.Text;

import org.apache.hadoop.mapreduce.Mapper;

public class MeanMaxMapper extends Mapper<LongWritable, Text, Text, IntWritable> {

public static final int MISSING = 9999;

public void map(LongWritable key, Text value, Mapper<LongWritable, Text, Text,

IntWritable>.Context context) throws IOException, InterruptedException {

int temperature;

String line = value.toString();

String month = line.substring(19, 21);

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 != 9999 && quality.matches("[01459]"))

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

}

}

**MeanMaxReducer.class**

package meanmax;

import java.io.IOException;

import org.apache.hadoop.io.IntWritable;

import org.apache.hadoop.io.Text;

import org.apache.hadoop.mapreduce.Reducer;

public class MeanMaxReducer extends Reducer<Text, IntWritable, Text, IntWritable> {

public void reduce(Text key, Iterable<IntWritable> values, Reducer<Text, IntWritable,

Text, IntWritable>.Context context) throws IOException, InterruptedException {

int max\_temp = 0;

int total\_temp = 0;

int count = 0;

int days = 0;

for (IntWritable value : values) {

int temp = value.get();

if (temp > max\_temp)

max\_temp = temp;

count++;

if (count == 3) {

total\_temp += max\_temp;

max\_temp = 0;

count = 0;

days++;

}

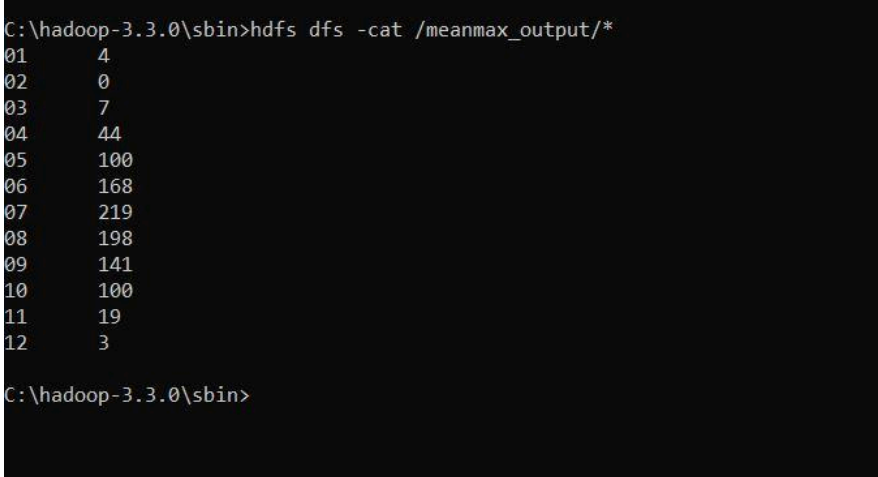
}

context.write(key, new IntWritable(total\_temp / days));

}

}





**Program 8**

**For a given Text file, create a Map Reduce program to sort the content in an alphabetic order listing only top 10 maximum occurrences of words.**

**Driver-TopN.class**

package samples.topn;

import java.io.IOException;

import java.util.StringTokenizer;

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.lib.input.FileInputFormat;

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

import org.apache.hadoop.util.GenericOptionsParser;

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.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);

}

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

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

private Text word = new Text();

private String tokens = "[\_|$#<>\\^=\\[\\]\\\*/\\\\,;,.\\-:()?!\"']";

public void map(Object key, Text value, Mapper<Object, Text, Text, IntWritable>.Context

context) throws IOException, InterruptedException {

String cleanLine = value.toString().toLowerCase().replaceAll(this.tokens, " ");

StringTokenizer itr = new StringTokenizer(cleanLine);

while (itr.hasMoreTokens()) {

this.word.set(itr.nextToken().trim());

context.write(this.word, one);

}

}

}

}

**TopNCombiner.class**

package samples.topn;

import java.io.IOException;

import org.apache.hadoop.io.IntWritable;

import org.apache.hadoop.io.Text;

import org.apache.hadoop.mapreduce.Reducer;

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

public void reduce(Text key, Iterable<IntWritable> values, Reducer<Text, IntWritable,

Text, IntWritable>.Context context) throws IOException, InterruptedException {

int sum = 0;

for (IntWritable val : values)

sum += val.get();

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

}

}

**TopNMapper.class**

package samples.topn;

import java.io.IOException;

import java.util.StringTokenizer;

import org.apache.hadoop.io.IntWritable;

import org.apache.hadoop.io.Text;

import org.apache.hadoop.mapreduce.Mapper;

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

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

private Text word = new Text();

private String tokens = "[\_|$#<>\\^=\\[\\]\\\*/\\\\,;,.\\-:()?!\"']";

public void map(Object key, Text value, Mapper<Object, Text, Text, IntWritable>.Context

context) throws IOException, InterruptedException {

String cleanLine = value.toString().toLowerCase().replaceAll(this.tokens, " ");

StringTokenizer itr = new StringTokenizer(cleanLine);

while (itr.hasMoreTokens()) {

this.word.set(itr.nextToken().trim());

context.write(this.word, one);

}

}

}

**TopNReducer.class**

package samples.topn;

import java.io.IOException;

import java.util.HashMap;

import java.util.Map;

import org.apache.hadoop.io.IntWritable;

import org.apache.hadoop.io.Text;

import org.apache.hadoop.mapreduce.Reducer;

import utils.MiscUtils;

public class TopNReducer extends Reducer<Text, IntWritable, Text, IntWritable> {

private Map<Text, IntWritable> countMap = new HashMap<>();

public void reduce(Text key, Iterable<IntWritable> values, Reducer<Text, IntWritable,

Text, IntWritable>.Context context) throws IOException, InterruptedException {

int sum = 0;

for (IntWritable val : values)

sum += val.get();

this.countMap.put(new Text(key), new IntWritable(sum));

}

protected void cleanup(Reducer<Text, IntWritable, Text, IntWritable>.Context context)

throws IOException, InterruptedException {

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

int counter = 0;

for (Text key : sortedMap.keySet()) {

if (counter++ == 20)

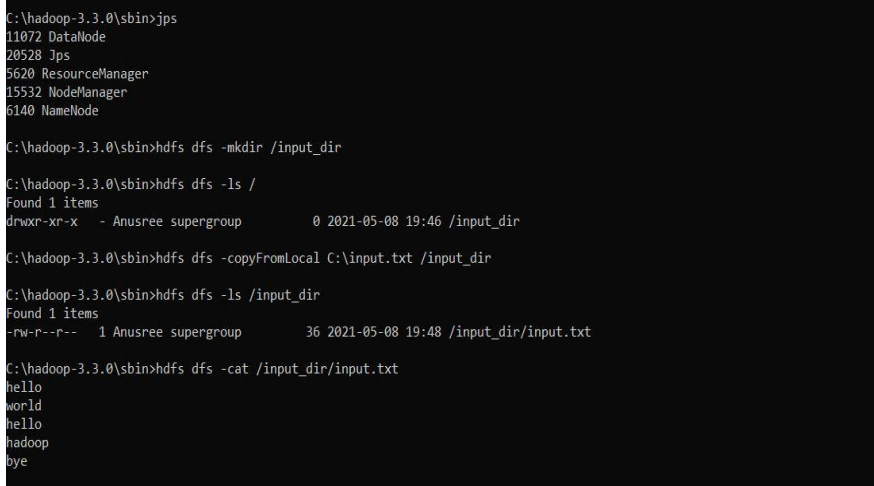
break;

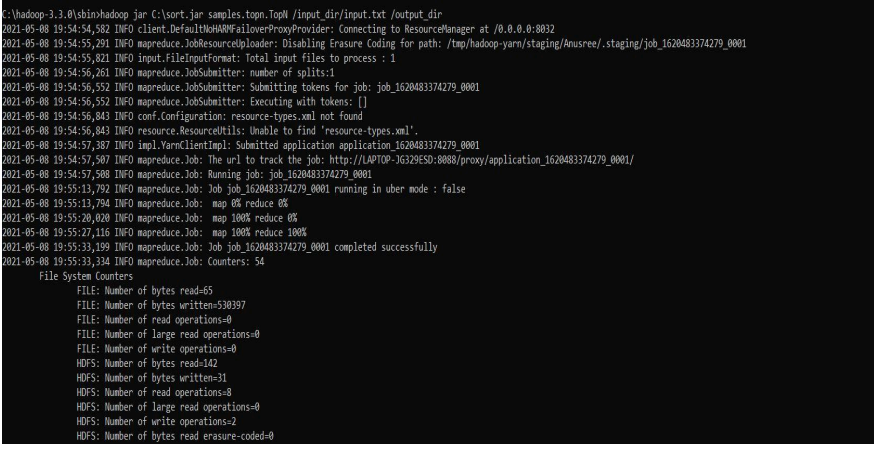
context.write(key, sortedMap.get(key));

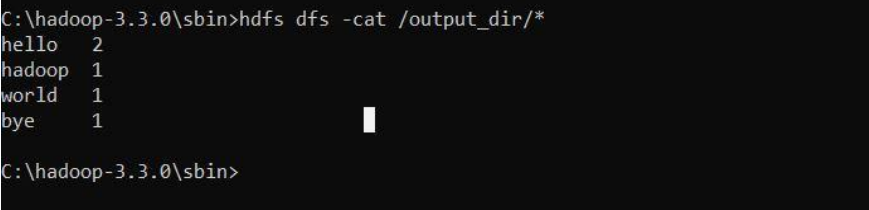
}

}

}

****

****

****