**VISVESVARAYA TECHNOLOGICAL UNIVERSITY**

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



**LAB REPORT**

**on**

**Big Data Analytics (22CS6PEBDA)**

***Submitted by:***

**Neha Bhaskar Kamath (1BM21CS113)**

**Under the Guidance of**

**Sneha P**

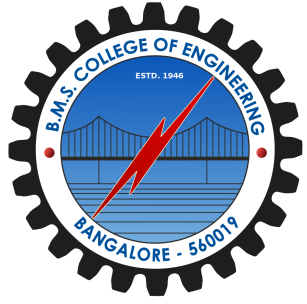
**Assistant Professor, BMSCE**

***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**

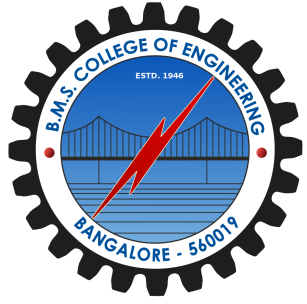
**March 2024 - June 2024**

**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 “**Big Data Analytics**” carried out by **Neha Bhaskar Kamath (1BM21CS113),** 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 **Big Data Analytics - (22CS6PEBDA)** work prescribed for the said degree.

**Sneha P**              **Dr. Jyothi S Nayak**

Associate Professor Professor and Head

Department of CSE Department of CSE

BMSCE, Bengaluru BMSCE, Bengaluru

**Table Of Contents**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **S.No.** | **Experiment Title** | | | **Page No** |
| **1** | **Course Outcomes** | | |  |
| **2** | **Experiments** | | |  |
|  | **2.1** | **Experiment - 1** | | **1** |
| **2.1.1** | **Question:**  **Perform the following DB operations using Cassandra.**   * Create a keyspace by name Employee * Create a column family by name, Employee-Info with attributes Emp\_Id Primary Key, Emp\_Name, Designation, Date\_of\_Joining, Salary, Dept\_Name * Insert the values into the table in batch * Update Employee name and Department of Emp-Id 121 * Sort the details of Employee records based on salary * Alter the schema of the table Employee\_Info to add a column Projects which stores a * set of Projects done by the corresponding Employee. * Update the altered table to add project names. * Create a TTL of 15 seconds to display the values of Employees. |
| **2.1.2** | **Code with Output** |
| **2.2** | **Experiment - 2** | | **3** |
| **2.2.1** | **Question:**  **Perform the following DB operations using Cassandra:**   * Create a keyspace by name Library * 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 * Insert the values into the table in batch * Display the details of the table created and increase the value of the counter * Write a query to show that a student with id 112 has taken a book “BDA” 2 times. * Export the created column to a csv file * Import a given csv dataset from local file system into Cassandra column family**.** |
| **2.2.2** | **Code with Output** |
| **2.3** | **Experiment - 3** | | **5** |
| **2.3.1** | **Question:**  MongoDB- CRUD Demonstration. |
| **2.3.2** | **Code with Output** |
| **2.4** | **Experiment - 4** | | **9** |
| **2.4.1** | **Question:**  Hadoop Installation Screenshot |
| **2.4.2** | **Screenshot** |
| **2.5** | **Experiment - 5** | | **10** |
| **2.5.1** | **Question:**  Execution of HDFS Commands for interaction with Hadoop Environment. (Minimum 10 commands to be executed) |
| **2.5.2** | **Code with Output** |
| **2.6** | **Experiment - 6** | | **12** |
| **2.6.1** | **Question:**  Implement WordCount Program on Hadoop framework. |
| **2.6.2** | **Code with Output** |
| **2.7** | **Experiment - 7** | | **14** |
| **2.7.1** | **Question:**  **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. 2. Find the mean max temperature for every month. |
| **2.7.2** | **Code with Output** |
| **2.8** | **Experiment - 8** | | **19** |
| **2.8.1** | **Question:**  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. |
| **2.8.2** | **Code with Output** |

1. **Course Outcomes**

**CO1:** Apply the concepts of NoSQL, Hadoop, Spark for a given task

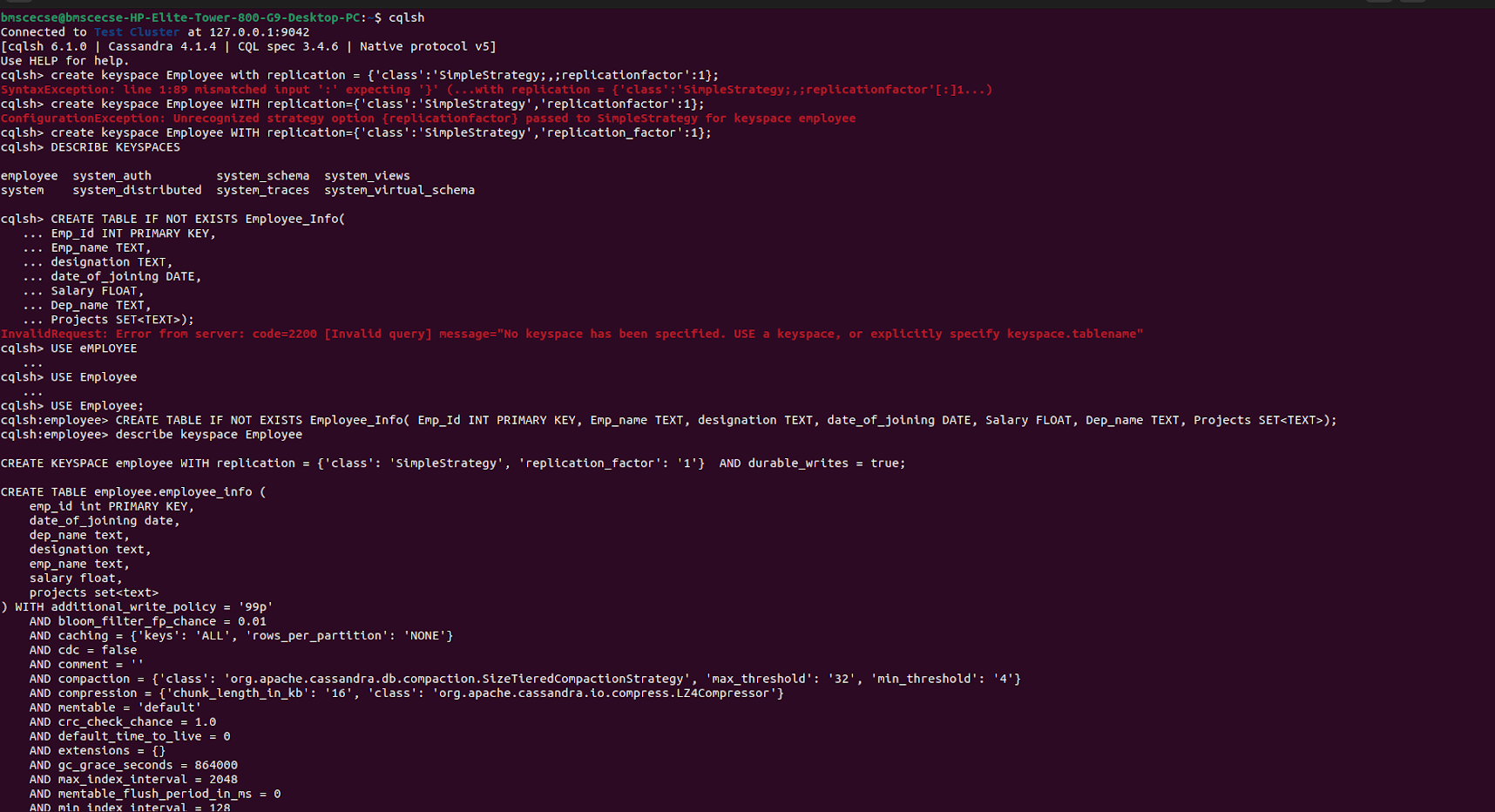
**CO2:** Analyse data analytic techniques for a given problem .

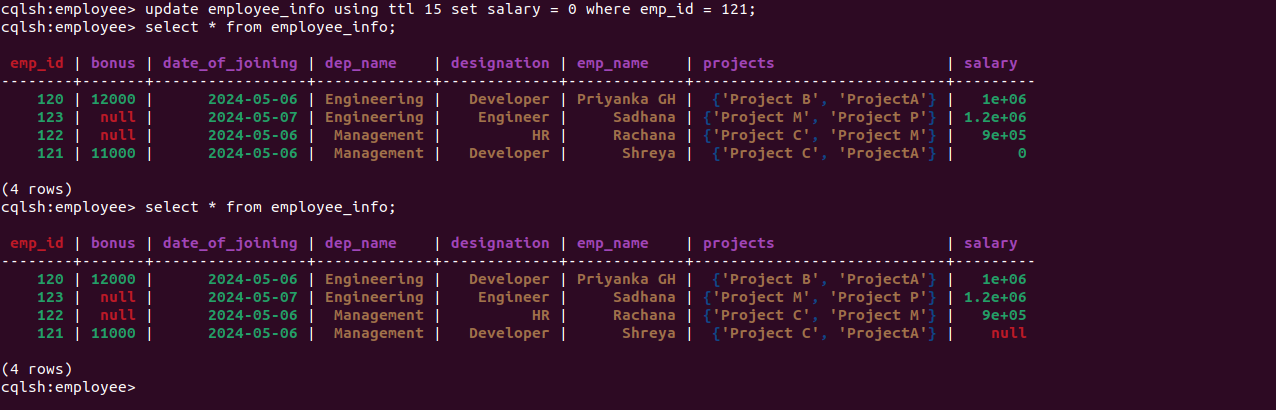
**CO3:** Conduct experiments using data analytics mechanisms for a given problem.

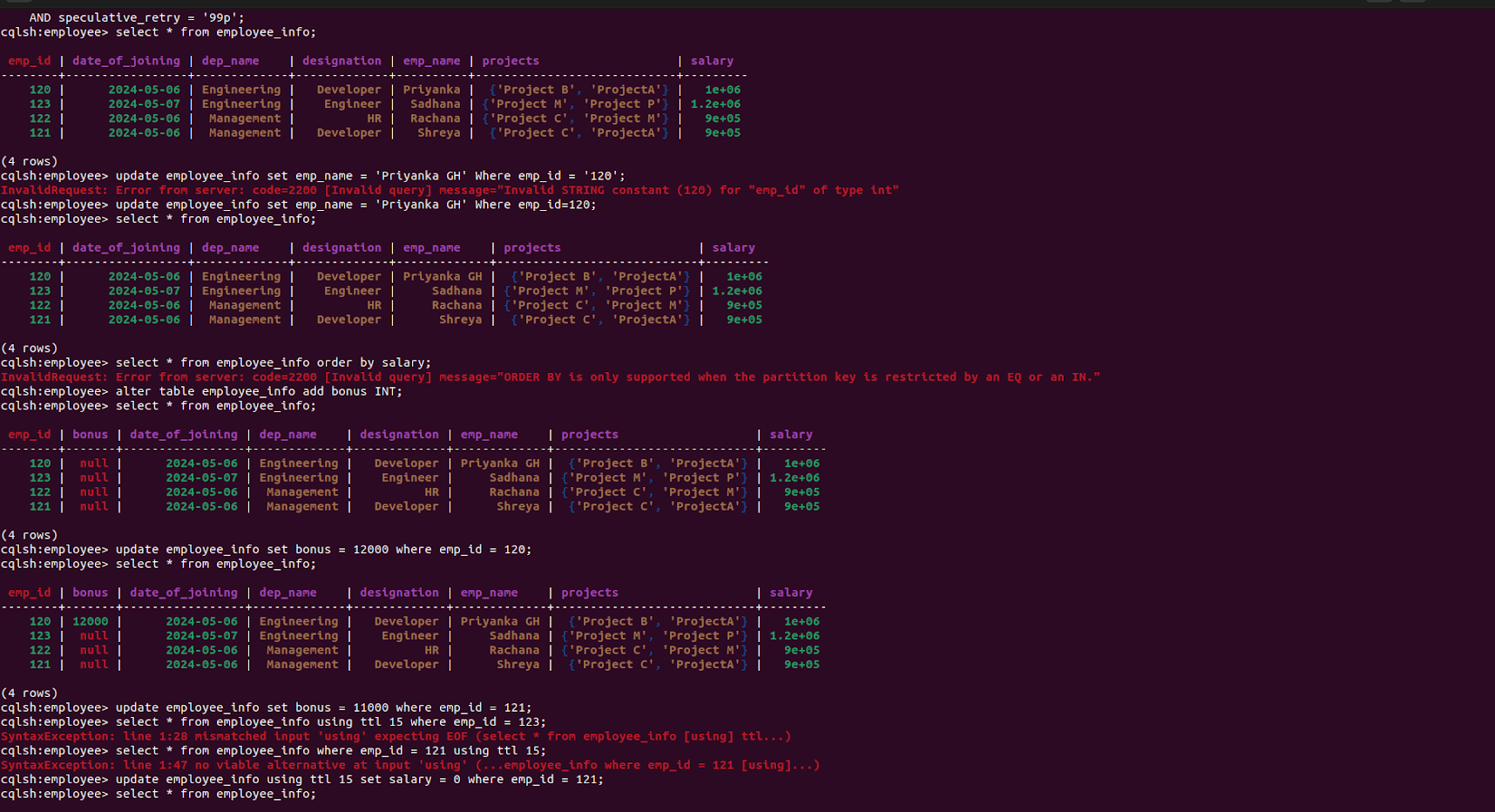
1. **Experiments**
   1. **Experiment - 1**
      1. **Question:**

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

* Create a keyspace by name Employee
* Create a column family by name, Employee-Info with attributes Emp\_Id Primary Key, Emp\_Name, Designation, Date\_of\_Joining, Salary, Dept\_Name
* Insert the values into the table in batch
* Update Employee name and Department of Emp-Id 121
* Sort the details of Employee records based on salary
* Alter the schema of the table Employee\_Info to add a column Projects which stores a
* set of Projects done by the corresponding Employee.
* Update the altered table to add project names.
* Create a TTL of 15 seconds to display the values of Employees.
  + 1. **Code with Output:**



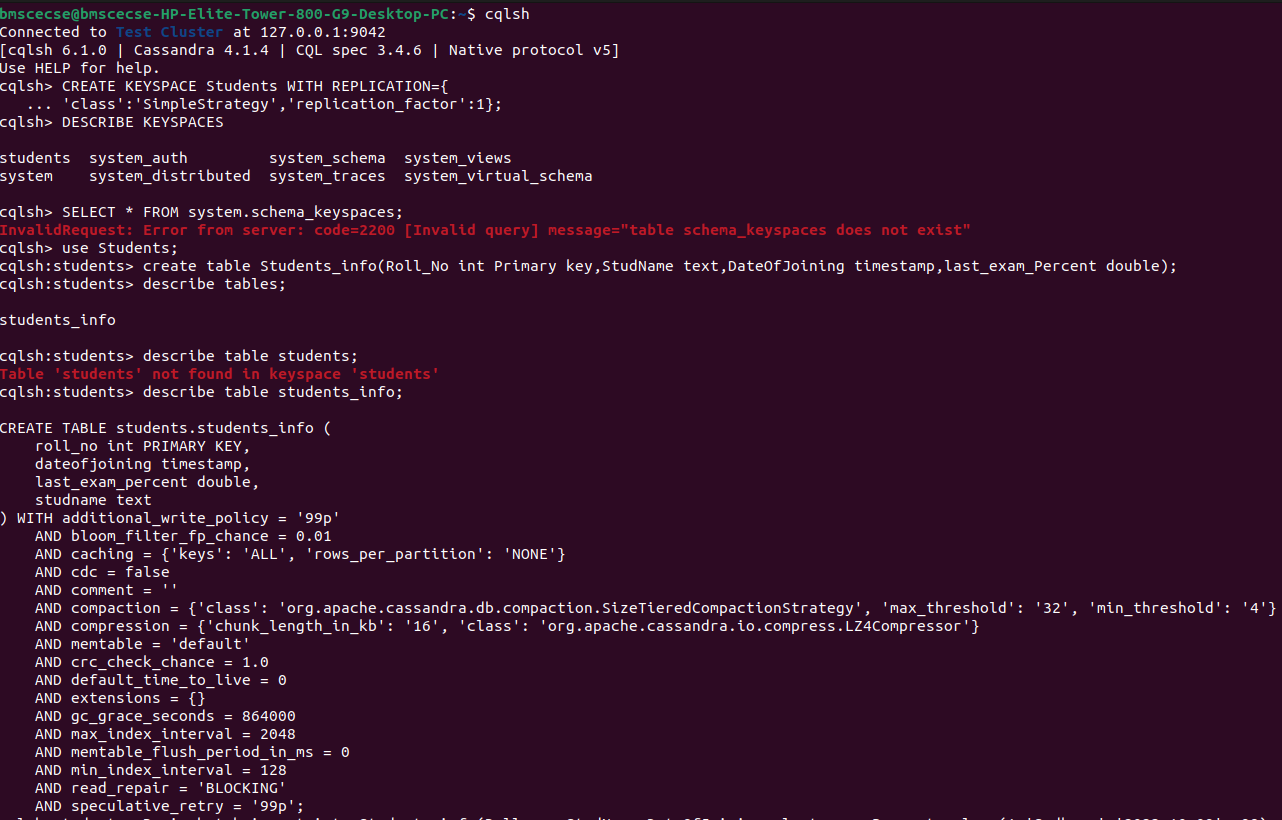


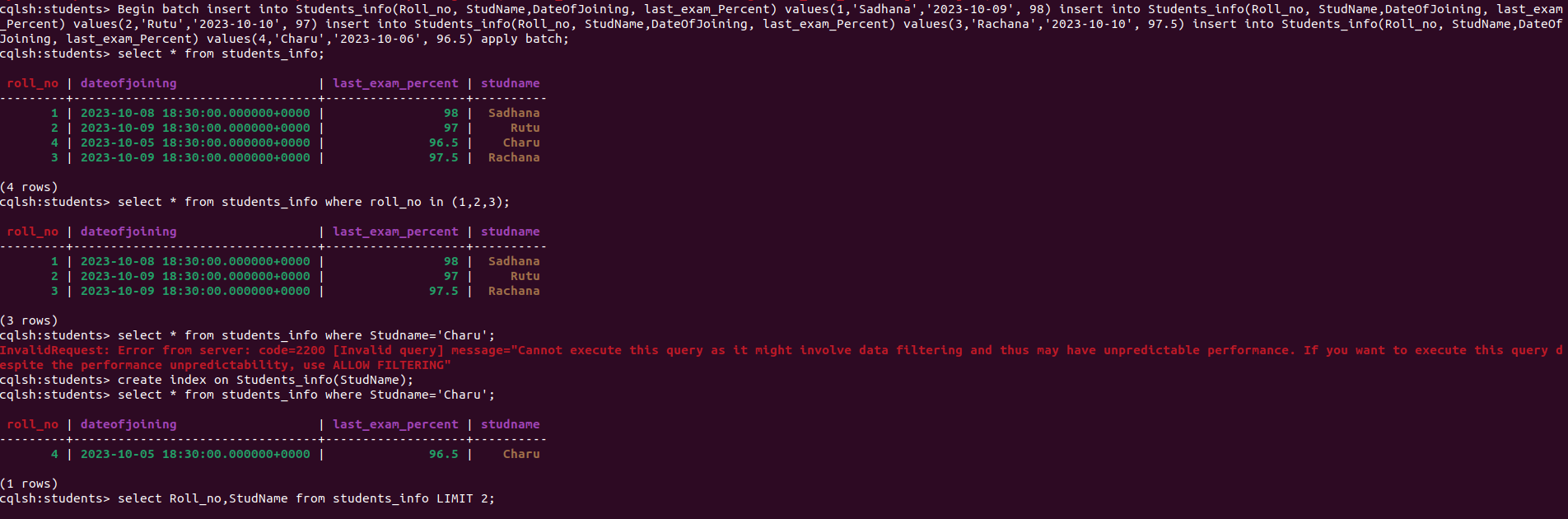


* 1. **Experiment - 2**
     1. **Question:**

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

* Create a keyspace by name Library
* 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
* Insert the values into the table in batch
* Display the details of the table created and increase the value of the counter
* Write a query to show that a student with id 112 has taken a book “BDA” 2 times.
* Export the created column to a csv file
* Import a given csv dataset from local file system into Cassandra column family**.**
  + 1. **Code with Output:**







* 1. **Experiment - 3**
     1. **Question:**

MongoDB - CRUD Demonstration.

* + 1. **Code with Output:**

1. **Create a database “Student” with the following attributes Rollno, Name , Age, ContactNo, Email-Id, grade, hobby:**

use Students

1. **Insert 5 appropriate values according to the below queries.**

db.students.insertMany([

{ "Rollno": 10, "Name": "John", "Age": 20, "ContactNo": "1234567890", "Email-Id": "john@example.com", "grade": "A", "hobby": "Reading" },

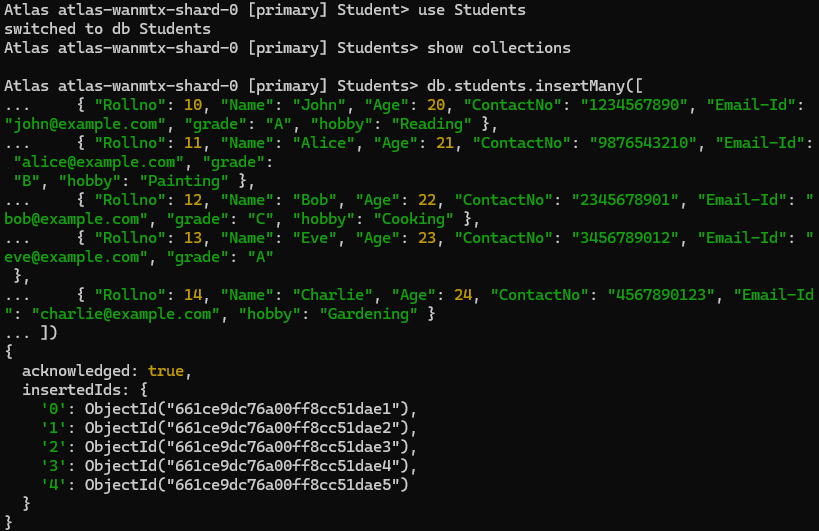
{ "Rollno": 11, "Name": "Alice", "Age": 21, "ContactNo": "9876543210", "Email-Id": "alice@example.com", "grade": "B", "hobby": "Painting" },

{ "Rollno": 12, "Name": "Bob", "Age": 22, "ContactNo": "2345678901", "Email-Id": "bob@example.com", "grade": "C", "hobby": "Cooking" },

{ "Rollno": 13, "Name": "Eve", "Age": 23, "ContactNo": "3456789012", "Email-Id": "eve@example.com", "grade": "A" },

{ "Rollno": 14, "Name": "Charlie", "Age": 24, "ContactNo": "4567890123", "Email-Id": "charlie@example.com", "hobby": "Gardening" }

])



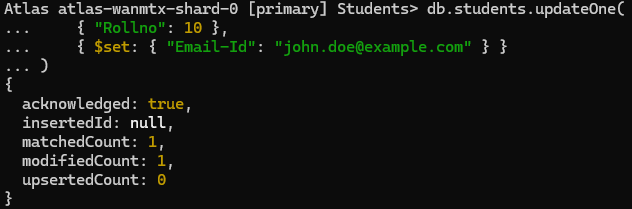
**3. Write query to update Email-Id of a student with rollno 10.**

db.students.updateOne(

{ "Rollno": 10 },

{ $set: { "Email-Id": "john.doe@example.com" } }

)



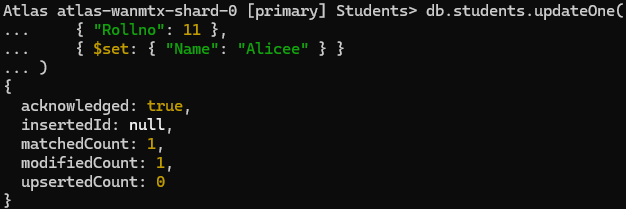
**4. Replace the student name from “Alice” to “Alicee” of rollno 11**

db.students.updateOne(

{ "Rollno": 11 },

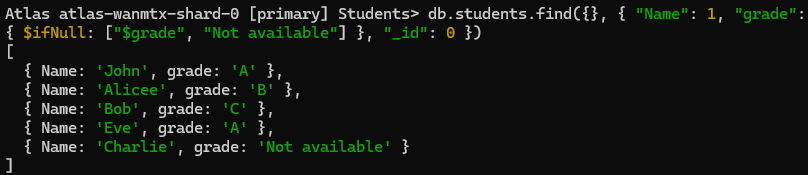
{ $set: { "Name": "Alicee" } }

)



**5. Display Student Name and grade(Add if grade is not present)where the \_id column is 1.**

db.students.find({}, { "Name": 1, "grade": { $ifNull: ["$grade", "Not available"] }, "\_id": 0 })



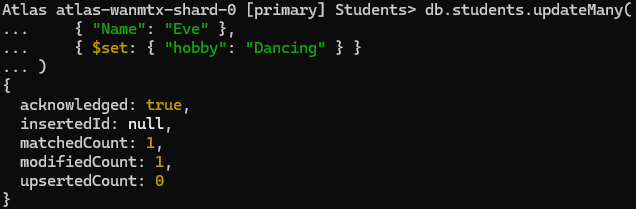
**6. Update to add hobbies**

db.students.updateMany(

{ "Name": "Eve" },

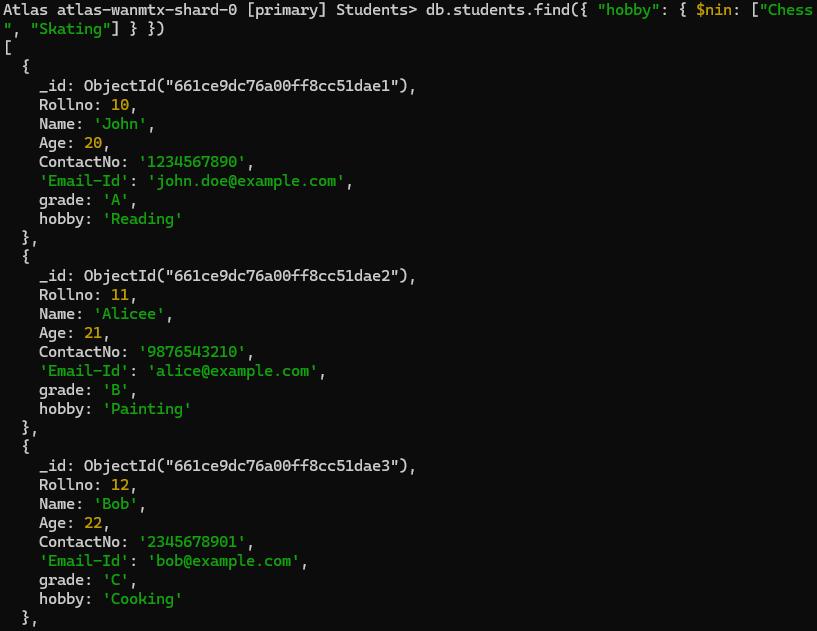
{ $set: { "hobby": "Dancing" } }

)



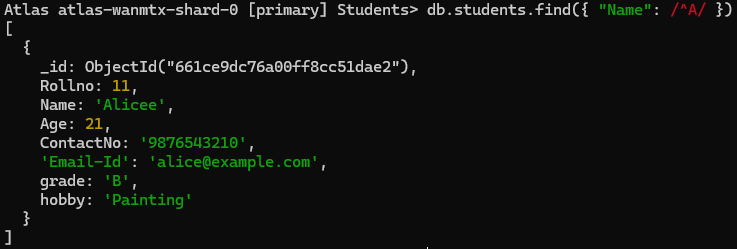
**7. Find documents where hobbies is set neither to Chess nor to Skating**

db.students.find({ "hobby": { $nin: ["Chess", "Skating"] } })



**8. Find documents whose name begins with A**

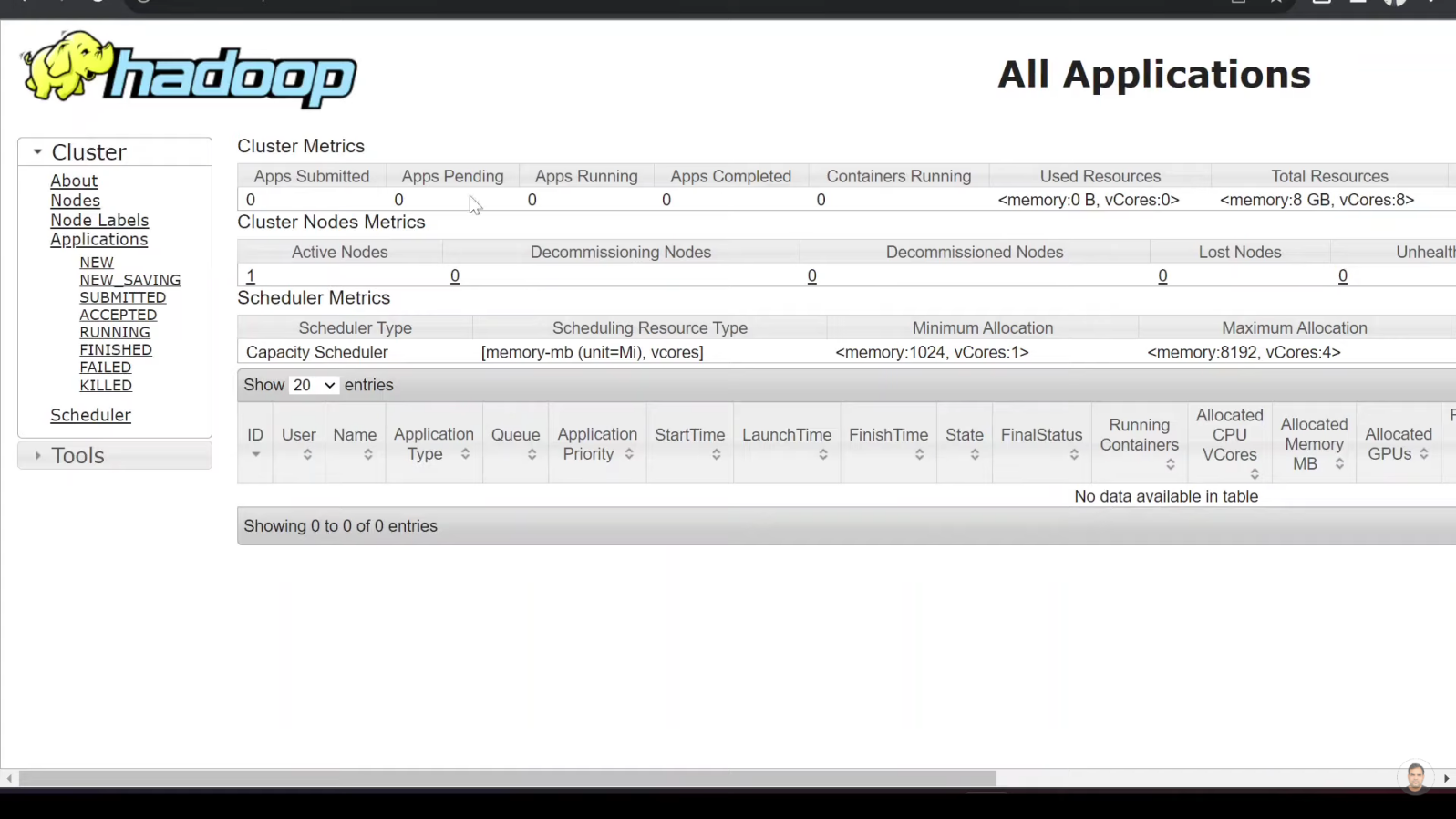
db.students.find({ "Name": /^A/ })

****

* 1. **Experiment - 4**
     1. **Question:**

Hadoop Installation Screenshot

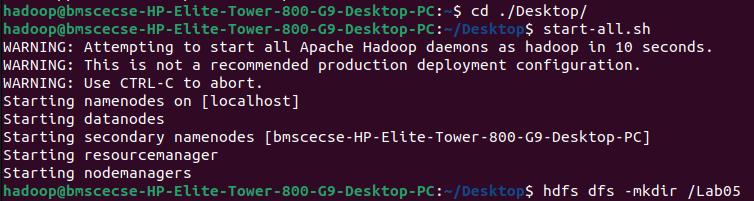
* + 1. **Screenshot:**

****

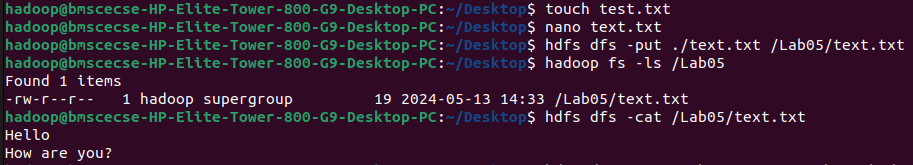
* 1. **Experiment - 5**
     1. **Question:**

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

* + 1. **Code with Output:**

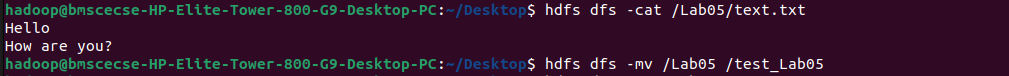
****

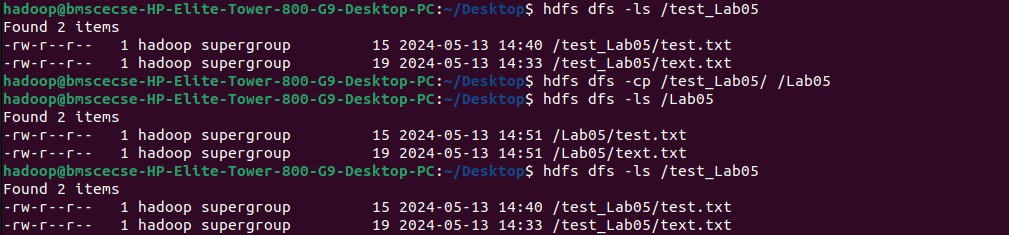
**2**

****

****

**5**

****

**`**

* 1. **Experiment - 6**
     1. **Question:**

Implement WordCount Program on Hadoop framework.

* + 1. **Code with Output:**

**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: WCDriver Java Class file.**

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;

}

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

{

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

System.out.println(exitCode);

}

}

* 1. **Experiment - 7**
     1. **Question:**

**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.
2. Find the mean max temperature for every month.
   * 1. **Code with Output:**

**a) 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) {

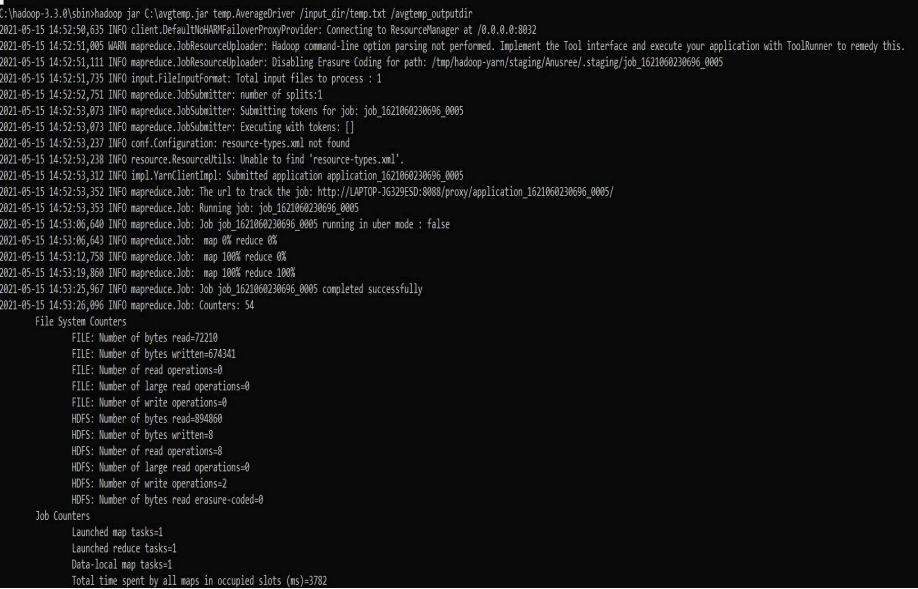
max\_temp += value.get();

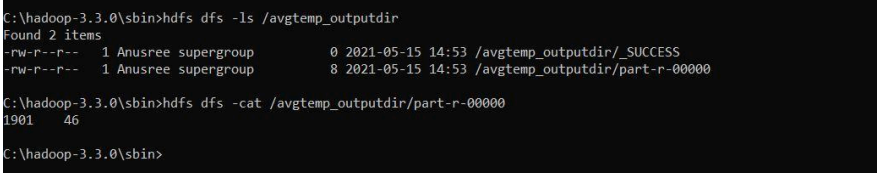
count++;

}

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

}}





**b) 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++;

}

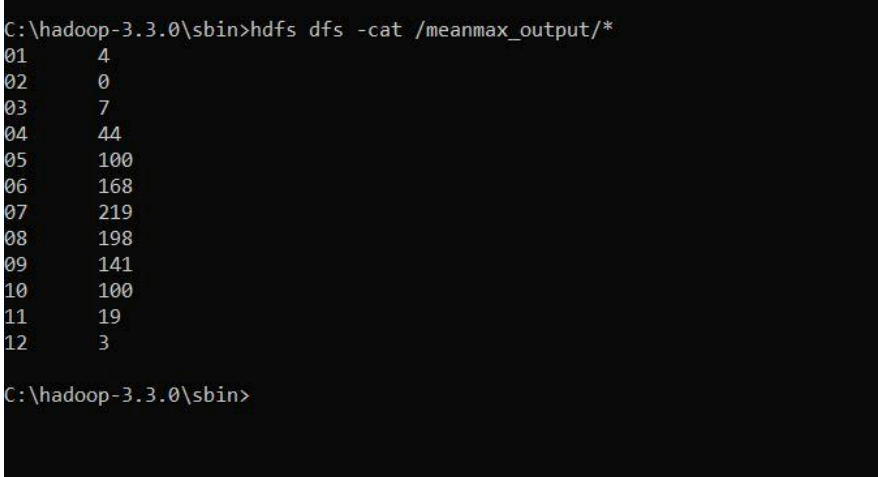
}

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

}

}





* 1. **Experiment - 8**
     1. **Question:**

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.

* + 1. **Code with Output:**

**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));

}

}

}

