**VISVESVARAYA TECHNOLOGICAL UNIVERSITY**

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



# LAB REPORT

**on**

Big Data Analytics

***Submitted by***

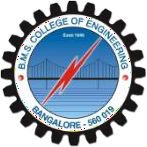
# Shashank D K(1BM21CS197)

***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 “Big Data Analytics” carried out by **Shashank D K (1BM21CS197),** 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- (22CS6PCBDA)** work prescribed for the said degree.

**Prof. Ramya K 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 | Cassandra DB Operations (Employee) | 1 |
| 2 | Cassandra DB Operations (Library) | 3 |
| 3 | MongoDB – CRUD Demonstration | 5 |
| 4 | Installing Hadoop | 9 |
| 5 | Execution of HDFS Commands | 9 |
| 6 | WordCount Program on Hadoop | 11 |
| 7 | Map Reduce Program on Weather Data | 15 |
| 8 | Map Reduce Program to Sort the Content | 21 |

**Course Outcome**

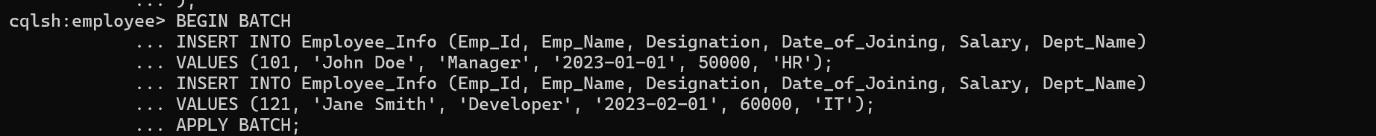
|  |  |
| --- | --- |
| **CO1** | Apply the concepts of NoSQL, Hadoop, Spark for a given task. |
| **CO2** | Analyse data analytic techniques for a given problem. |
| **CO3** | Conduct experiments using data analytics mechanisms for a given problem. |

# Perform the following DB operations using Cassandra

* 1. Create a keyspace by name Employee



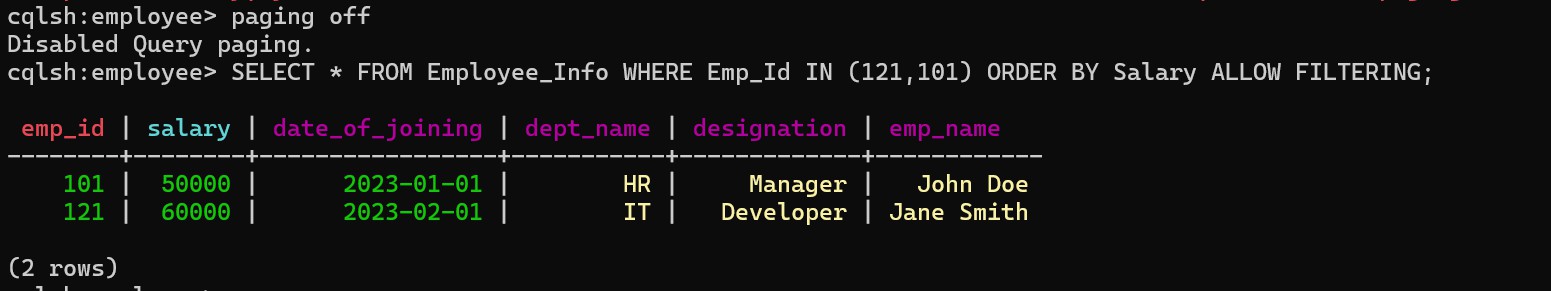
* 1. Create a column family by name Employee-Info with attributes Emp\_Id Primary Key, Emp\_Name, Designation, Date\_of\_Joining, Salary, Dept\_Name
  2. Insert the values into the table in batch



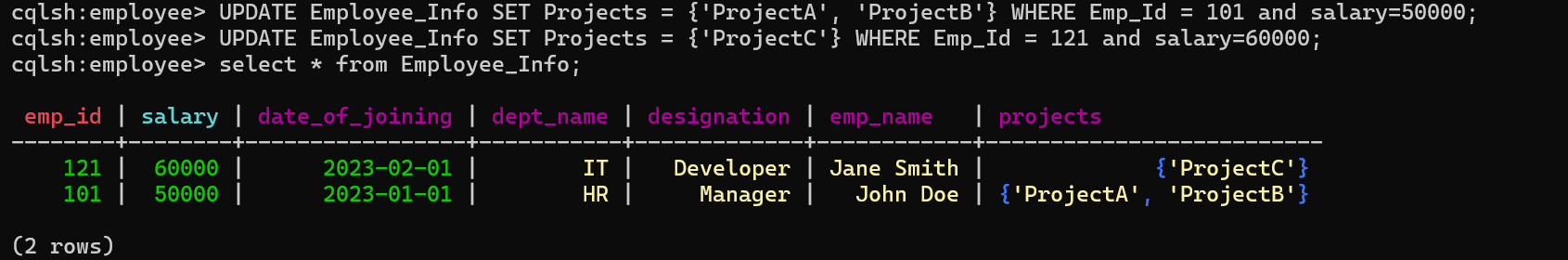
* 1. Update Employee name and Department of Emp-Id 121



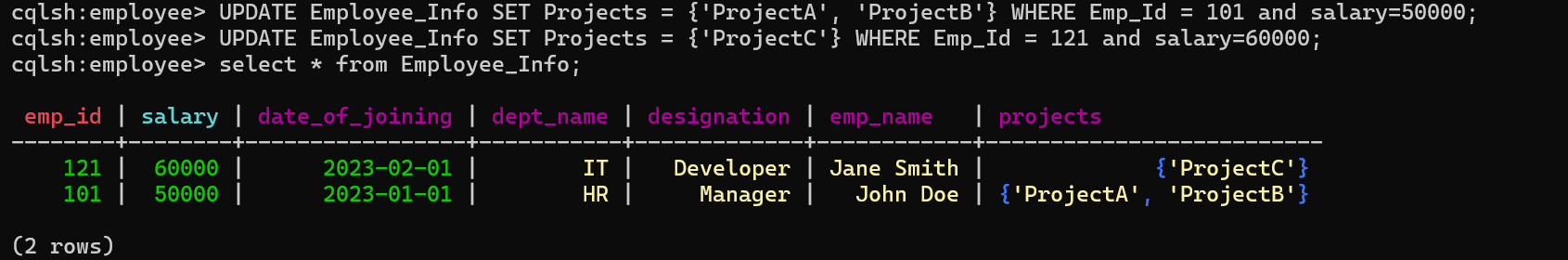
* 1. Sort the details of Employee records based on salary



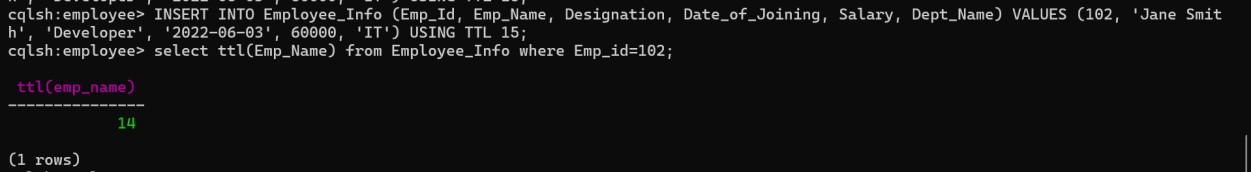
* 1. Alter the schema of the table Employee\_Info to add a column Projects which stores a set of Projects done by the corresponding Employee.



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

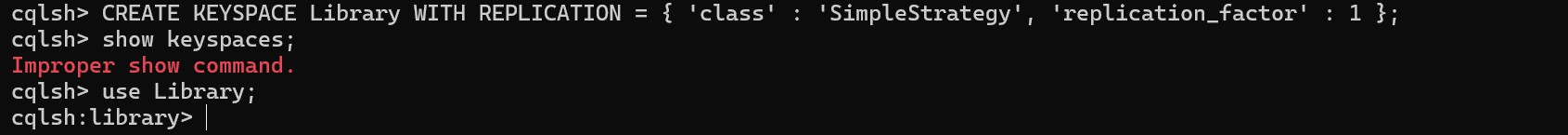


* 1. Create a TTL of 15 seconds to display the values of Employees.

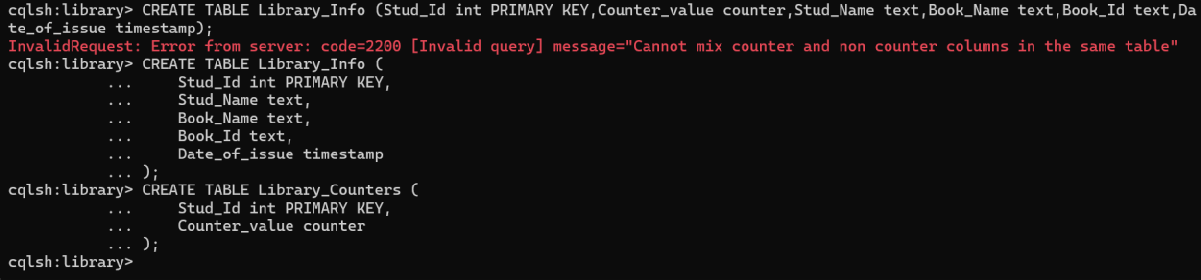


# Perform the following DB operations using Cassandra.

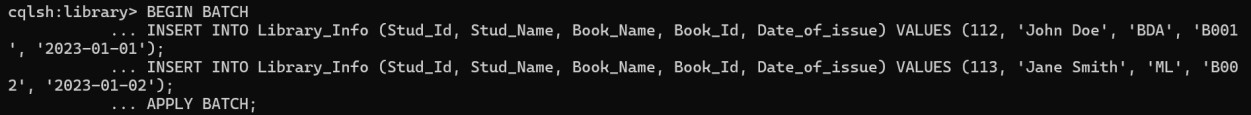
* 1. Create a keyspace by name Library



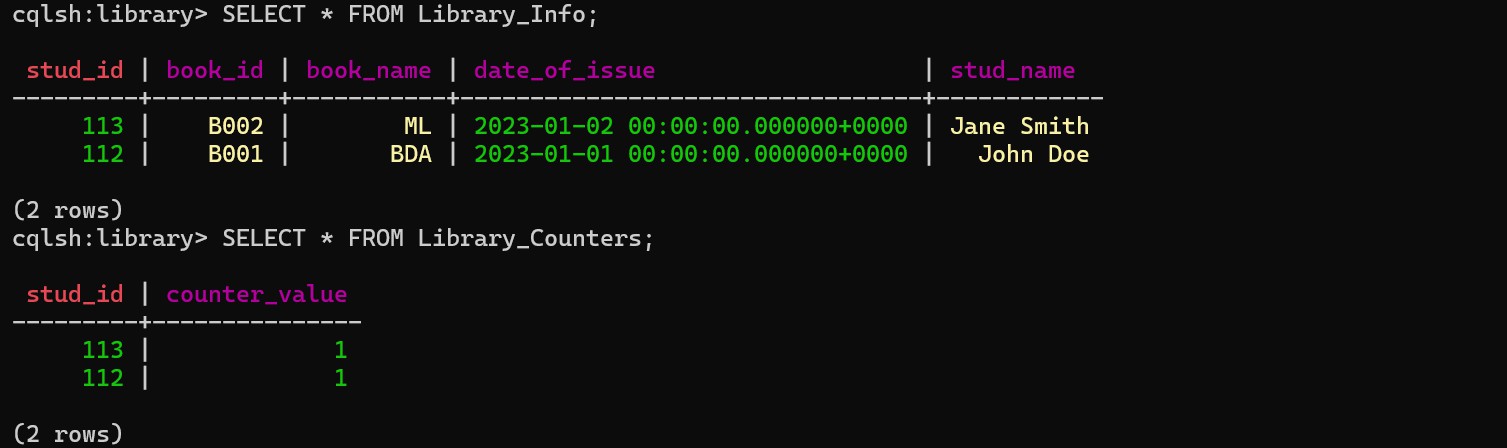
* 1. Create a column family by name Library-Info with attributes Stud\_Id Primary Key, Counter\_value of type Counter, Stud\_Name, Book-Name, Book-Id, Date\_of\_issue



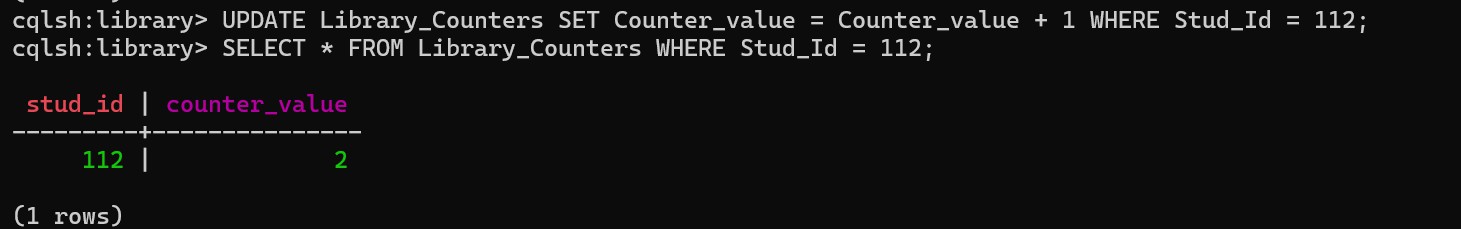
* 1. Insert the values into the table in batch



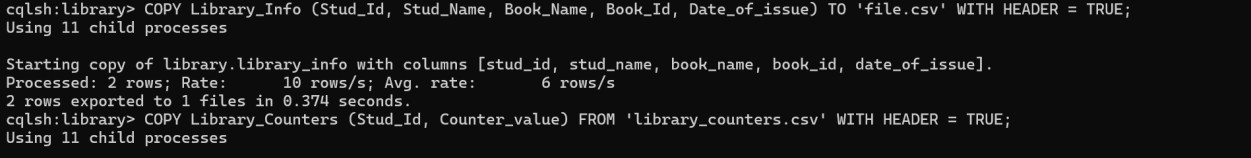
* 1. Display the details of the table created and increase the value of the counter



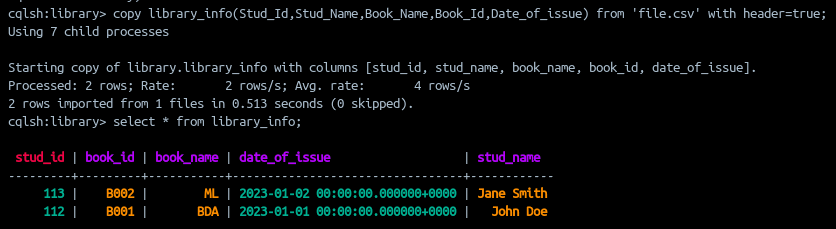
* 1. Write a query to show that a student with id 112 has taken a book “BDA” 2 times.



* 1. Export the created column to a csv file



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



# MongoDB- CRUD Demonstration

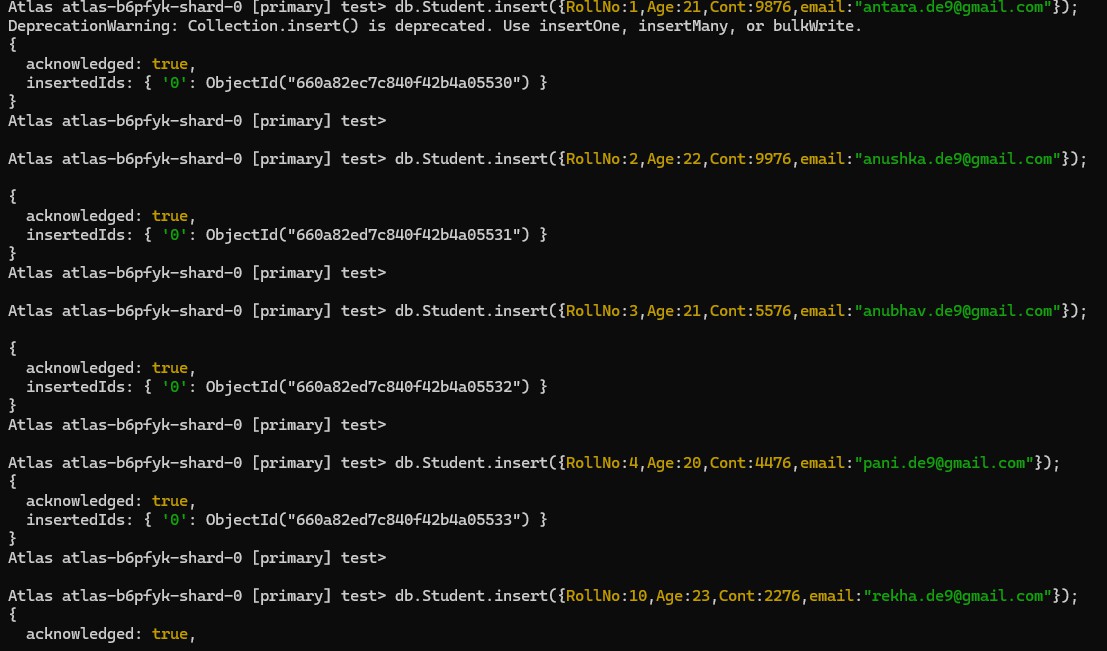
SETUP:

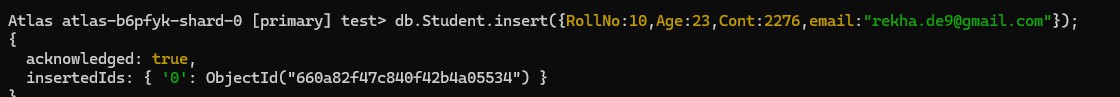


* 1. Create a database “Student” with the following attributes Rollno, Age, ContactNo, Email-Id.

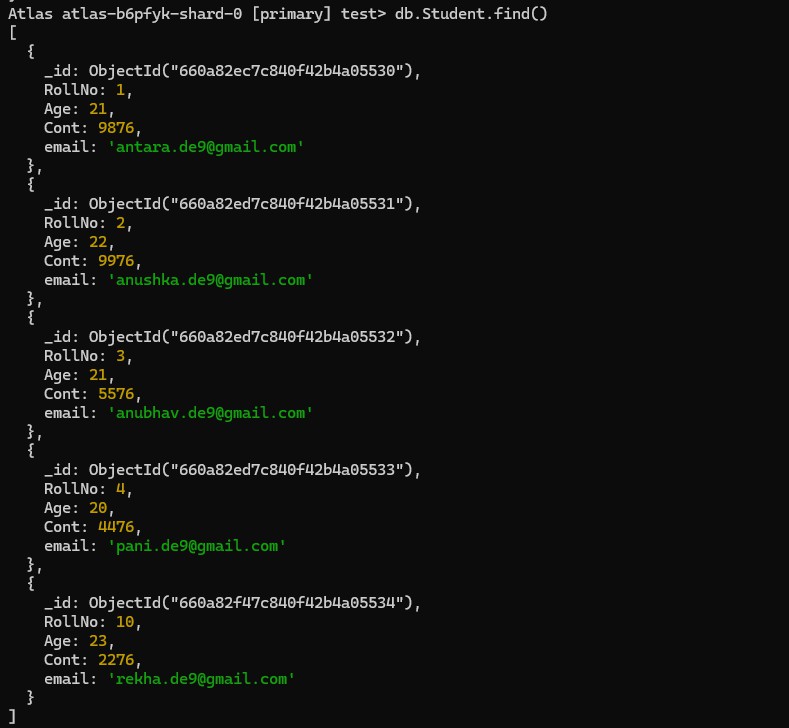


* 1. Insert appropriate values(at least 5)

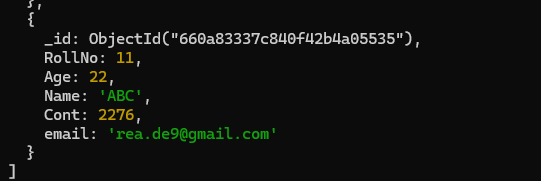




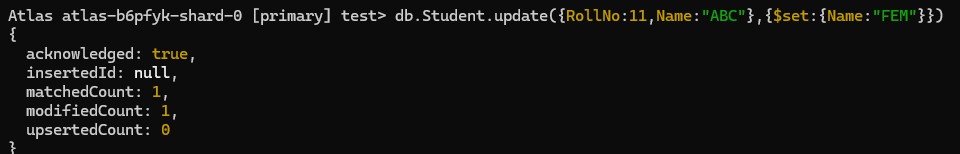
* 1. View the data



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



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



* 1. Drop the table

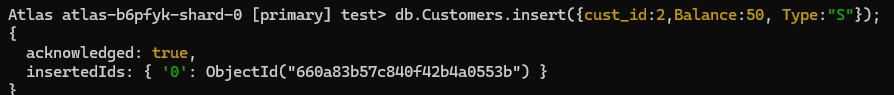


1. Create a collection by name Customers with the following attributes. Cust\_id, Acc\_Bal, Acc\_Type

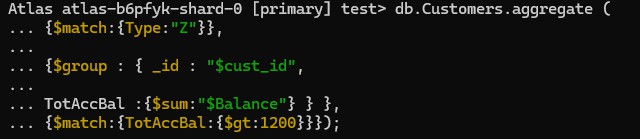


1. Insert at least 5 values into the table

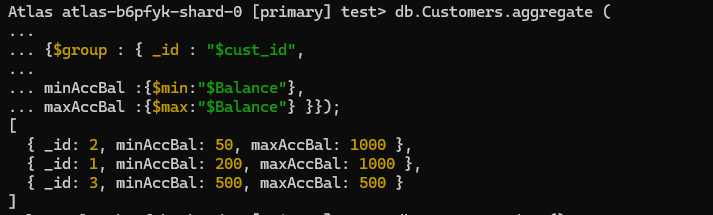




1. Write a query to display those records whose total account balance is greater than 1200 of account type ‘Z’ for each customer\_id.



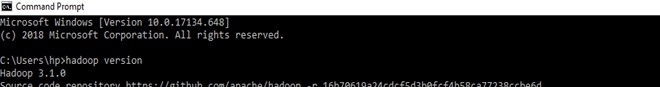
1. Determine Minimum and Maximum account balance for each customer\_id.



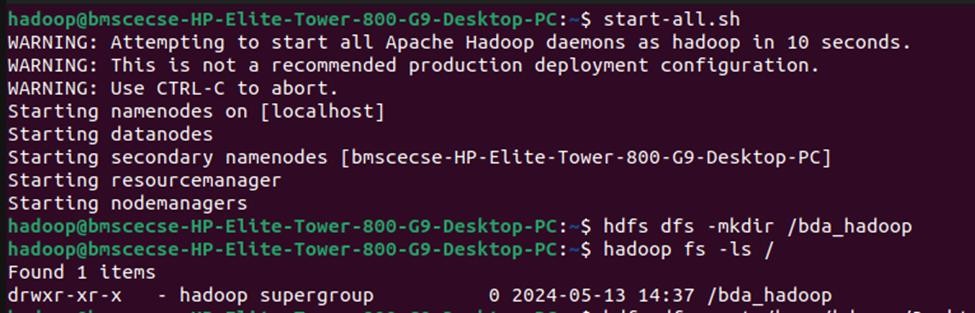
1. Drop the table



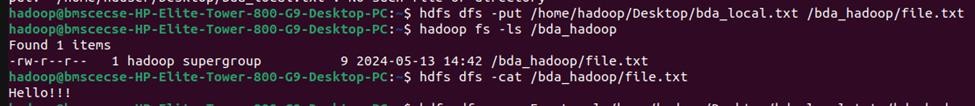
# Screenshot of Hadoop installed



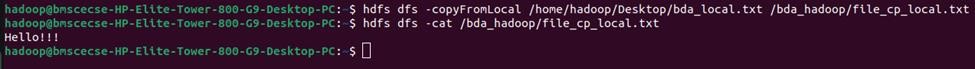
1. **Execution of HDFS Commands for interaction with Hadoop Environment. (Minimum 10 commands to be executed)**
   1. mkdir 2.ls



1. put



1. copyFromLocal



1. get





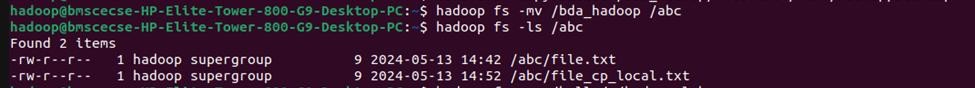
1. copyToLocal



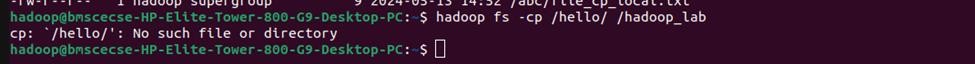
1. cat



1. mv



1. cp



# Implement WordCount Program on Hadoop framework

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&lt;LongWritable, Text, Text,

IntWritable&gt; {

// Map function

public void map(LongWritable key, Text value, OutputCollector&lt;Text, IntWritable&gt; output, Reporter rep) throws IOException

{

String line = value.toString();

// Splitting the line on spaces

for (String word : line.split(&quot; &quot;))

{

if (word.length() &gt; 0)

{

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

} } } }

Reducer Code: You have to copy paste this program into the WCReducer Java Class file

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

// Reduce function

public void reduce(Text key, Iterator&lt;IntWritable&gt; value, OutputCollector&lt;Text, IntWritable&gt; 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 &lt; 2)

{

System.out.println(&quot;Please give valid inputs&quot;); 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

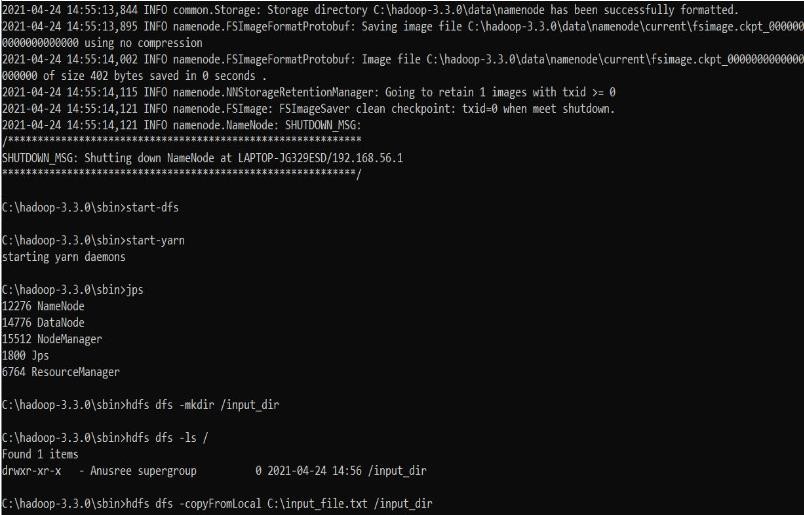
public static void main(String args[]) throws Exception

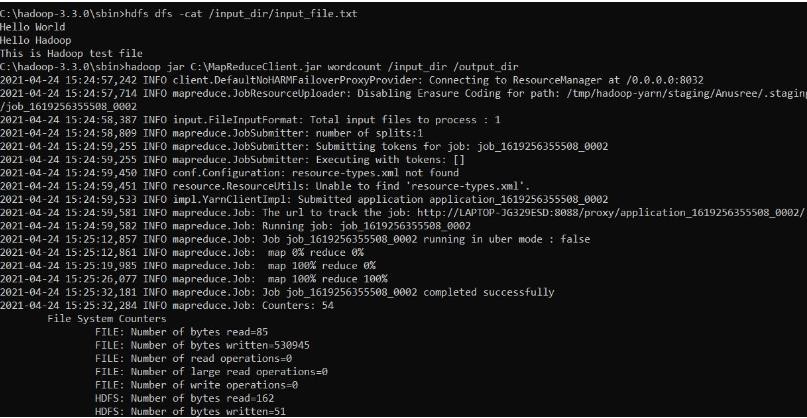
{

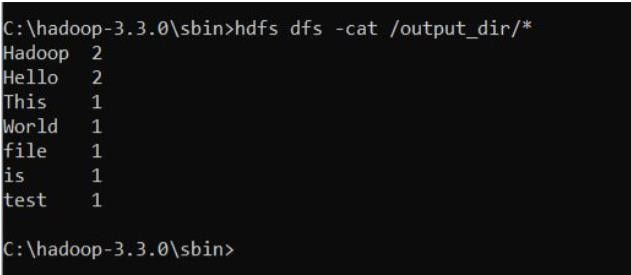
int exitCode = ToolRunner.run(new WCDriver(), args); System.out.println(exitCode);

}

} OUTPUT







1. **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&lt;LongWritable, Text, Text, IntWritable&gt; { public static final int MISSING = 9999;

public void map(LongWritable key, Text value, Mapper&lt;LongWritable, Text, Text, IntWritable&gt;.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 &amp;&amp; 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&lt;Text, IntWritable, Text, IntWritable&gt; { public void reduce(Text key, Iterable&lt;IntWritable&gt; values, Reducer&lt;Text, IntWritable, Text, IntWritable&gt;.Context context) throws IOException, InterruptedException {

int max\_temp = 0; int count = 0;

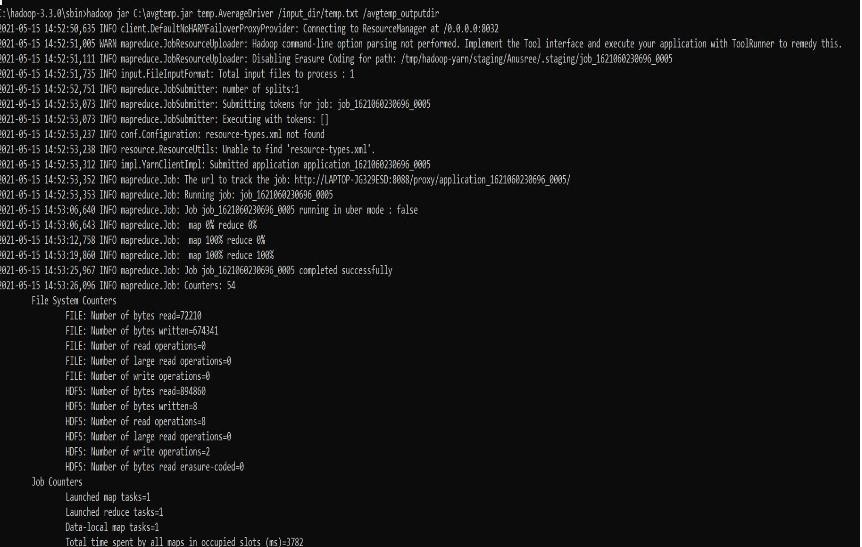
for (IntWritable value : values) { max\_temp += value.get(); count++;

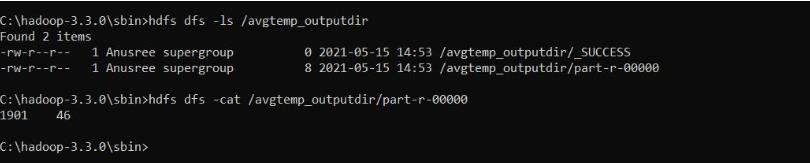
}

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

}

} OUTPUT





# 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&lt;LongWritable, Text, Text, IntWritable&gt; { public static final int MISSING = 9999;

public void map(LongWritable key, Text value, Mapper&lt;LongWritable, Text, Text, IntWritable&gt;.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 &amp;&amp; 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&lt;Text, IntWritable, Text, IntWritable&gt; { public void reduce(Text key, Iterable&lt;IntWritable&gt; values, Reducer&lt;Text, IntWritable, Text, IntWritable&gt;.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 &gt; max\_temp) max\_temp = temp; count++;

if (count == 3) { total\_temp += max\_temp; max\_temp = 0;

count = 0; days++;

}

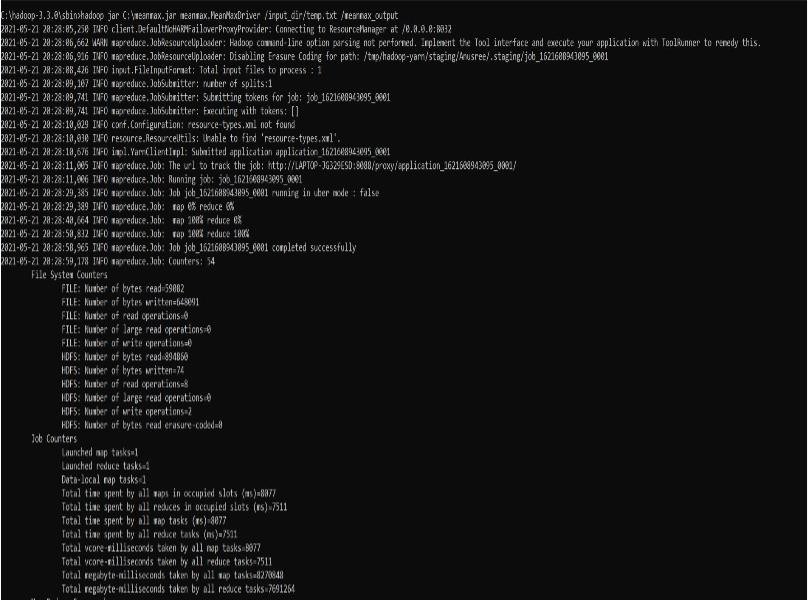
}

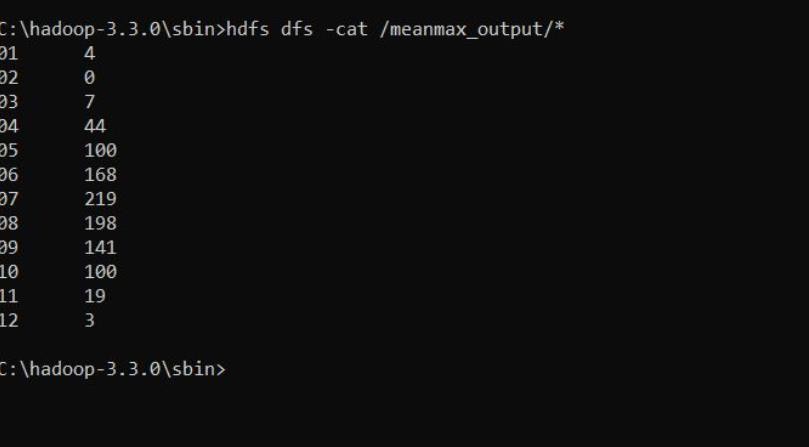
context.write(key, new IntWritable(total\_temp / days));

}

}

OUTPUT





# 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.

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&lt;Object, Text, Text, IntWritable&gt; { private static final IntWritable one = new IntWritable(1);

private Text word = new Text();

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

public void map(Object key, Text value, Mapper&lt;Object, Text, Text, IntWritable&gt;.Context context) throws IOException, InterruptedException {

String cleanLine = value.toString().toLowerCase().replaceAll(this.tokens, &quot; &quot;); 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&lt;Text, IntWritable, Text, IntWritable&gt; { public void reduce(Text key, Iterable&lt;IntWritable&gt; values, Reducer&lt;Text, IntWritable, Text, IntWritable&gt;.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&lt;Object, Text, Text, IntWritable&gt; { private static final IntWritable one = new IntWritable(1);

private Text word = new Text();

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

public void map(Object key, Text value, Mapper&lt;Object, Text, Text, IntWritable&gt;.Context context) throws IOException, InterruptedException {

String cleanLine = value.toString().toLowerCase().replaceAll(this.tokens, &quot; &quot;); 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&lt;Text, IntWritable, Text, IntWritable&gt; { private Map&lt;Text, IntWritable&gt; countMap = new HashMap&lt;&gt;();

public void reduce(Text key, Iterable&lt;IntWritable&gt; values, Reducer&lt;Text, IntWritable, Text, IntWritable&gt;.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&lt;Text, IntWritable, Text, IntWritable&gt;.Context context) throws IOException, InterruptedException {

Map&lt;Text, IntWritable&gt; sortedMap = MiscUtils.sortByValues(this.countMap); int counter = 0;

for (Text key : sortedMap.keySet()) { if (counter++ == 20)

break;

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

}

}

}

OUTPUT

