**VISVESVARAYA TECHNOLOGICAL UNIVERSITY**

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



**LAB REPORT**

**On**

**BIG DATA ANALYTICS**

**Submitted by**

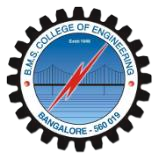
**ADITYA DUA(1BM21CS006)**

**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 to 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 **ADITYA DUA(1BM21CS006)**, 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 2023-24. The Lab report has been approved as it satisfies the academic requirements in respect of Big Data Analytics Lab - **(22CS6PCBDA )**work prescribed for the said degree.

**DR.PALLAVI GB** **Dr. Jyothi S Nayak**

Assistant Professor Professor and Head

Department of CSE Department of CSE

BMSCE, Bengaluru BMSCE, Bengaluru

# Index

|  |  |  |
| --- | --- | --- |
| **Sl.**  **No.** | **Experiment Title** | **Page No.** |
| 1 | MongoDB- CRUD Demonstration | 1 |
| 2 | Perform the following DB operations using CassandraEmployee keyspace. | 10 |
| 3 | Perform the following DB operations using Cassandra-Library keyspace. | 13 |
| 4 | Screenshot of Hadoop installed | 15 |
| 5 | Execution of HDFS Commands for interaction with Hadoop Environment. | 16 |
| 6 | Implement WordCount Program on Hadoop framework | 18 |
| 7 | From the following link extract the weather data https://github.com/tomwhite/hadoopbook/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 | 22 |
| 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. | 25 |

**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 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.**
2. **Insert appropriate values**
3. **Write query to update Email-Id of a student with rollno 10.**
4. **. Replace the student name from “ABC” to “FEM” of rollno 11**

Solution:

Atlas atlas-12eb3b-shard-0 [primary] MY\_DB> db.createCollection("Student")

;

{ ok: 1 }

Atlas atlas-12eb3b-shard-0 [primary] MY\_DB>

db.Student.insert({\_id:1,name:"Alice",rollNo:80,age:20,phNo:"9999988888",email:"alice@gmail.com"})

;

DeprecationWarning: Collection.insert() is deprecated. Use insertOne, insertMany, or bulkWrite.

{ acknowledged: true, insertedIds: { '0': 1 } } Atlas atlas-12eb3b-shard-0 [primary] MY\_DB>

db.Student.insert({\_id:2,name:"Bob",rollNo:81,age:20,phno:"8888855555",email:"bob@gmail.com"}); { acknowledged: true, insertedIds: { '0': 2 } } Atlas atlas-12eb3b-shard-0 [primary] MY\_DB>

db.Student.insert({\_id:3,name:"Cath",rollNo:82,age:21,phno:"8888877777",email:"cath@gmail.com"}); { acknowledged: true, insertedIds: { '0': 3 } }

Atlas atlas-12eb3b-shard-0 [primary] MY\_DB> db.Student.find();

[

{

\_id: 1, name: 'Alice', rollNo: 80,

age: 20, phNo: '9999988888',

email: 'alice@gmail.com'

},

{

\_id: 2, name: 'Bob', rollNo: 81, age: 20,

email: 'bob@gmail.com',

phNo: '8888855555'

},

{

\_id: 3, name: 'Cath', rollNo: 82, age: 21,

email: 'cath@gmail.com',

phNo: '8888877777'

}

]

Atlas atlas-12eb3b-shard-0 [primary] MY\_DB> db.Student.update({rollNo:80},

{$set:{email:"alice123@gmail.com"}});

DeprecationWarning: Collection.update() is deprecated. Use updateOne, updateMany, or bulkWrite.

{ acknowledged: true, insertedId: null, matchedCount: 1, modifiedCount: 1, upsertedCount: 0

}

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

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**
2. **Write a query to display those records whose total account balance is greater than 1200 of account type ‘Z’ for each customer\_id.**
3. **Determine Minimum and Maximum account balance for each customer\_i**

Atlas atlas-12eb3b-shard-0 [primary] MY\_DB> db.createCollection("

Customers");

{ ok: 1 }

Atlas atlas-12eb3b-shard-0 [primary] MY\_DB>

db.Student.insertOne({custId:1,accBal:10000,accType:"saving"});

{ acknowledged: true,

insertedId: ObjectId('660295b055dc2f3d86c4479f')

}

Atlas atlas-12eb3b-shard-0 [primary] MY\_DB>

db.Customers.insertOne({custId:1,accBal:10000,accType:"saving"});

{ acknowledged: true,

insertedId: ObjectId('660295c155dc2f3d86c447a0')

}

Atlas atlas-12eb3b-shard-0 [primary] MY\_DB>

db.Customers.insertOne({custId:2,accBal:50000,accType:"current"});

{ acknowledged: true,

insertedId: ObjectId('6602960055dc2f3d86c447a1')

}

Atlas atlas-12eb3b-shard-0 [primary] MY\_DB>

db.Customers.insertOne({custId:3,accBal:60000,accType:"current"});

{ acknowledged: true,

insertedId: ObjectId('6602960e55dc2f3d86c447a2')

}

Atlas atlas-12eb3b-shard-0 [primary] MY\_DB>

db.Customers.insertOne({custId:4,accBal:20000,accType:"savings"});

{ acknowledged: true,

insertedId: ObjectId('6602961c55dc2f3d86c447a3')

}

Atlas atlas-12eb3b-shard-0 [primary] MY\_DB>

db.Customers.insertOne({custId:5,accBal:200000,accType:"current"});

{ acknowledged: true,

insertedId: ObjectId('6602962955dc2f3d86c447a4')

}

Atlas atlas-12eb3b-shard-0 [primary] MY\_DB> db.Customers.find();

[

{

\_id: ObjectId('

660295c155dc2f3d86c447a0'),

custId: 1, accBal: 10000, accType: 'savings'

},

{

\_id: ObjectId('6602960055dc2f3d86c447a1'), custId: 2, accBal: 50000,

accType: 'current'

},

{

\_id: ObjectId('6602960e55dc2f3d86c447a2'), custId: 3, accBal: 60000,

accType: 'current'

},

{

\_id: ObjectId('6602961c55dc2f3d86c447a3'), custId: 4, accBal: 20000, accType: 'savings'

},

{

\_id: ObjectId('6602962955dc2f3d86c447a4'), custId: 5, accBal: 200000,

accType: 'current'

},

{

\_id: ObjectId('6602980955dc2f3d86c447a5'), custId: 1, accBal: 30000,

accType: 'savings'

},

{

\_id: ObjectId('6602982f55dc2f3d86c447a6'), custId: 2, accBal: 20000, accType: 'current'

}

]

Atlas atlas-12eb3b-shard-0 [primary] MY\_DB> db.Customers.aggregate({$

match:{accType:'savings'}},{$group:{\_id:"$custId",TotalAccBal:{$sum:"$accBal"}}},{$match:{TotalA ccBal:{$gt:20000}}});

[ { \_id: 1, TotalAccBal: 40000 } ]

Atlas atlas-12eb3b-shard-0 [primary] MY\_DB>

db.Customers.aggregate({$group:{\_id:"$custId",minAccBal:{$min:"$accBal"},maxAccBal:{$max:"accB al"}}});

[

{ \_id: 1, minAccBal: 10000, maxAccBal: 'accBal' },

{ \_id: 4, minAccBal: 20000, maxAccBal: 'accBal' },

{ \_id: 5, minAccBal: 200000, maxAccBal: 'accBal' },

{ \_id: 2, minAccBal: 20000, maxAccBal: 'accBal' },

{ \_id: 3, minAccBal: 60000, maxAccBal: 'accBal' }

]

Atlas atlas-12eb3b-shard-0 [primary] MY\_DB>

db.Customers.aggregate({$group:{\_id:"$custId",minAccBal:{$min:"$accBal"},maxAccBal:{$max:"$acc Bal"}}});

[

{ \_id: 3, minAccBal: 60000, maxAccBal: 60000 },

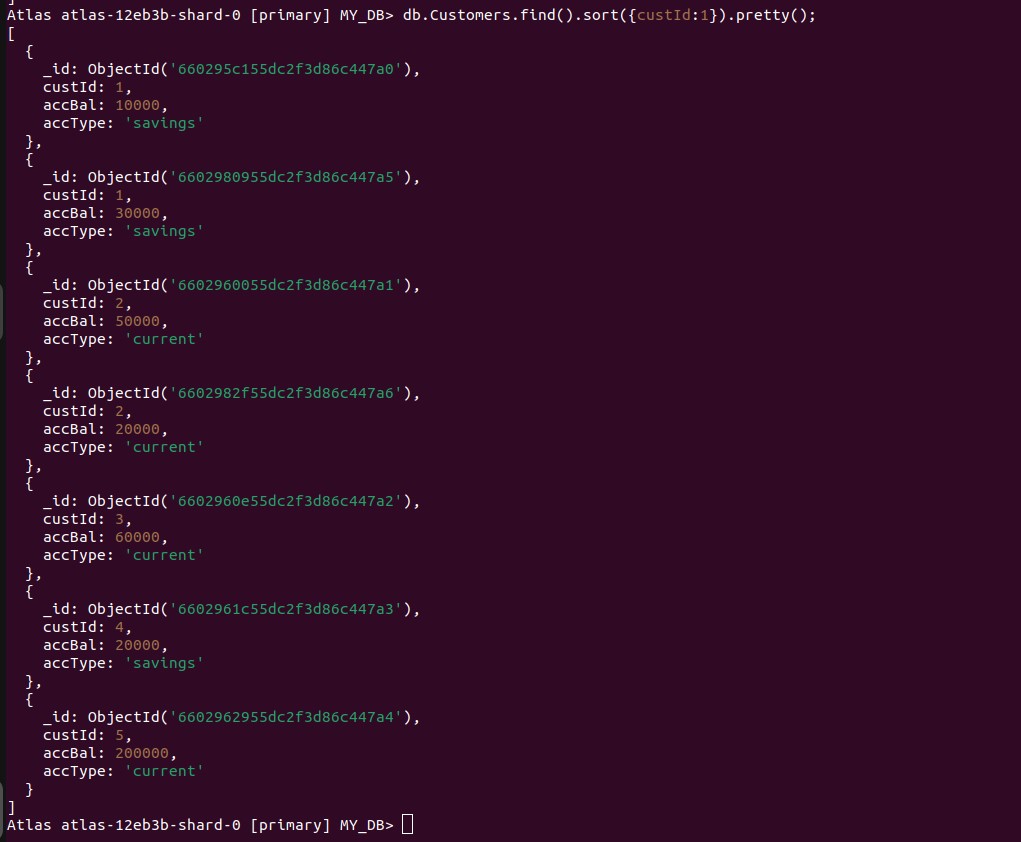
{ \_id: 4, minAccBal: 20000, maxAccBal: 20000 },

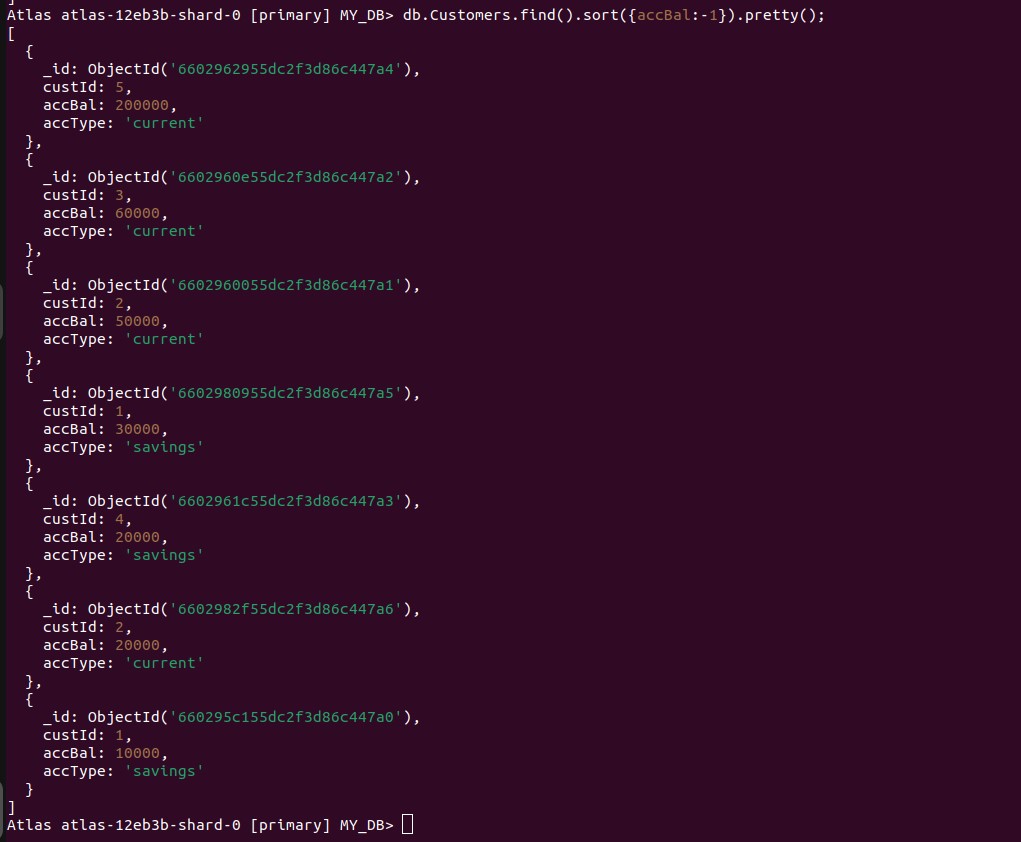
{ \_id: 5, minAccBal: 200000, maxAccBal: 200000 },

{ \_id: 2, minAccBal: 20000, maxAccBal: 50000 },

{ \_id: 1, minAccBal: 10000, maxAccBal: 30000 }

]



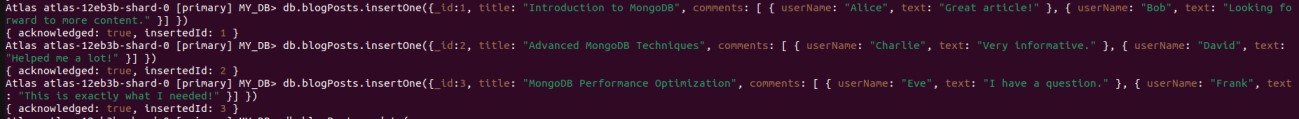


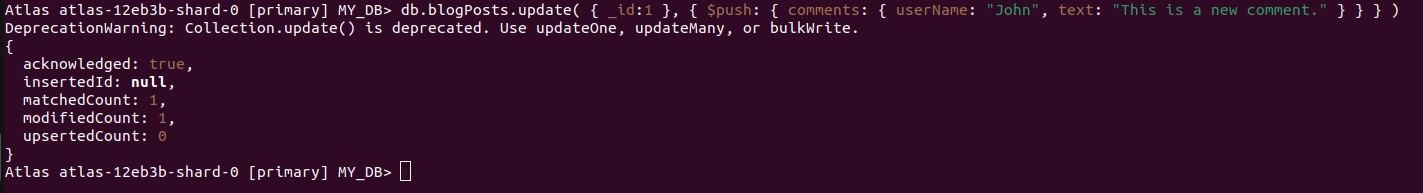
Create a collection by the name blogPosts and it has 3 fields id, title and comments.

In the collection the comments field is an array which consists of user details. Each collection consists of two user details inside the comments array- user name and text

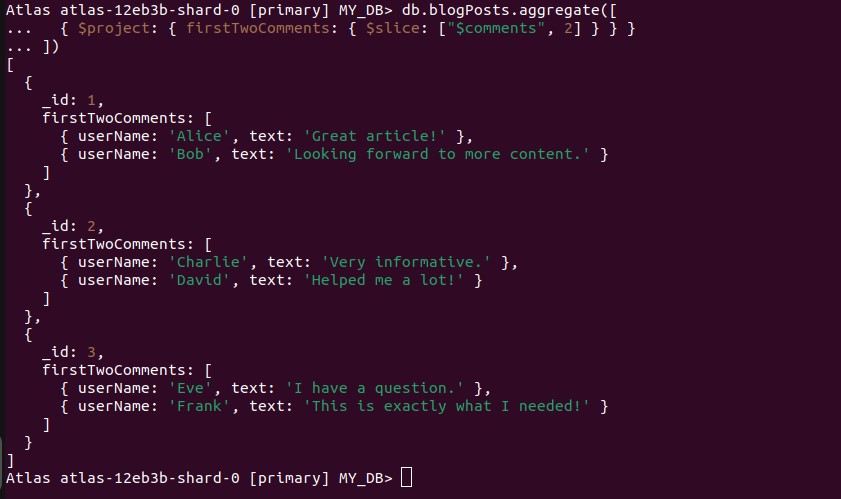
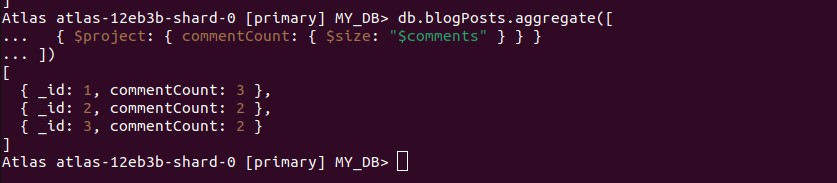
Demonstrate the following

1. Adding an element into array
2. Display second element
3. Display size of the array
4. Display first two elements of the array
5. Update the document with id 4 and replace the element present in 1st index position of the array with another array











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

1. Create a keyspace by name Employee
2. Create a column family by name

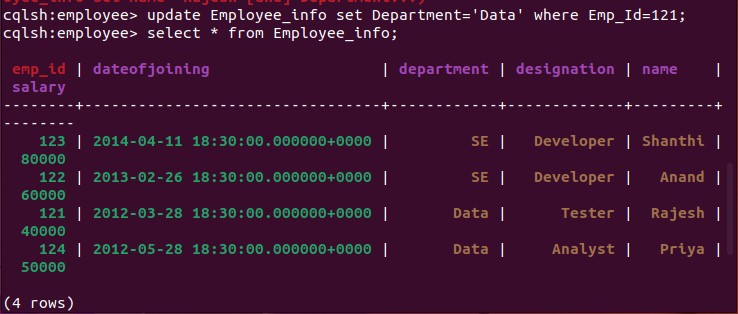
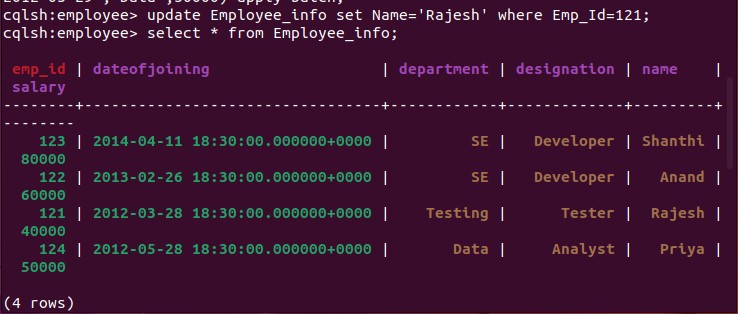
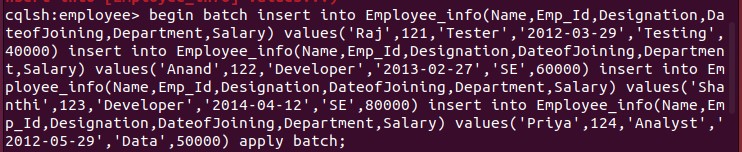
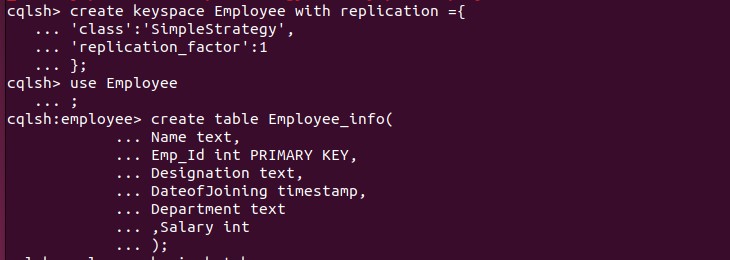
Employee-Info with attributes

Emp\_Id Primary Key, Emp\_Name,

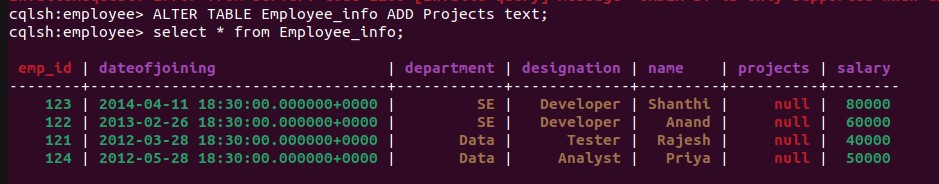
Designation, Date\_of\_Joining, Salary, Dept\_Name

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.

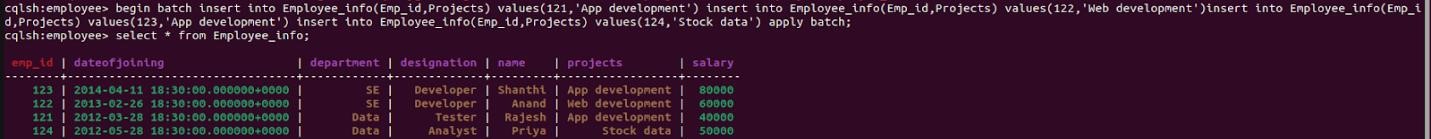
8.Create a TTL of 15 seconds to display the values of Employees.



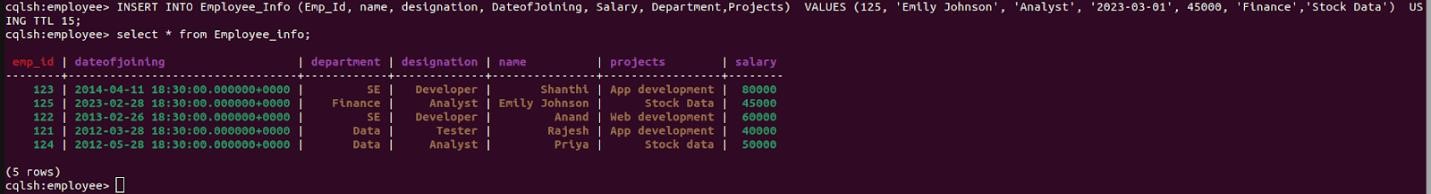
1. Alter the schema of the table Employee Info to add a column Projects which stores a set of Projects done by the corresponding Employee.



1. Update the altered table to add project names.



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



1. **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

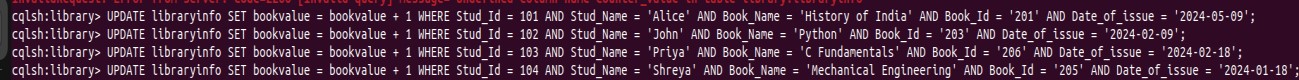
1.Create a keyspace by name Library



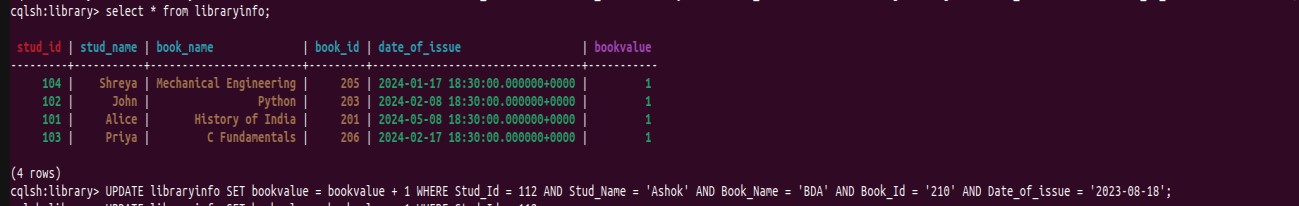
1. Create a column family by name Library-Info with attributes Stud\_Id Primary Key,Counter\_value of type Counter,Stud\_Name, Book-Name, Book-Id, Date\_of\_issue.



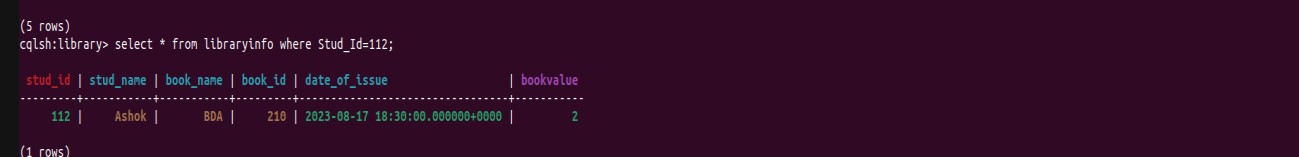
1. Insert the values into the table in batch



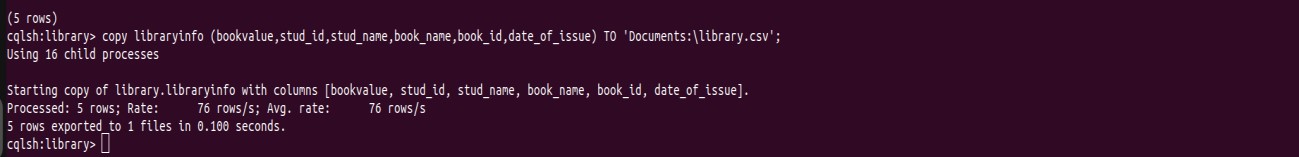
1. Display the details of the table created and increase the value of the counter



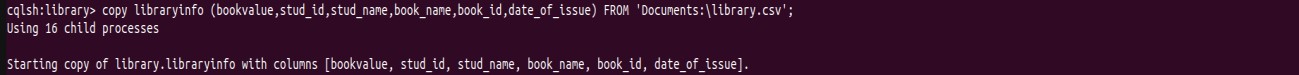
1. Write a query to show that a student with id 112 has taken a book “BDA” 2 time



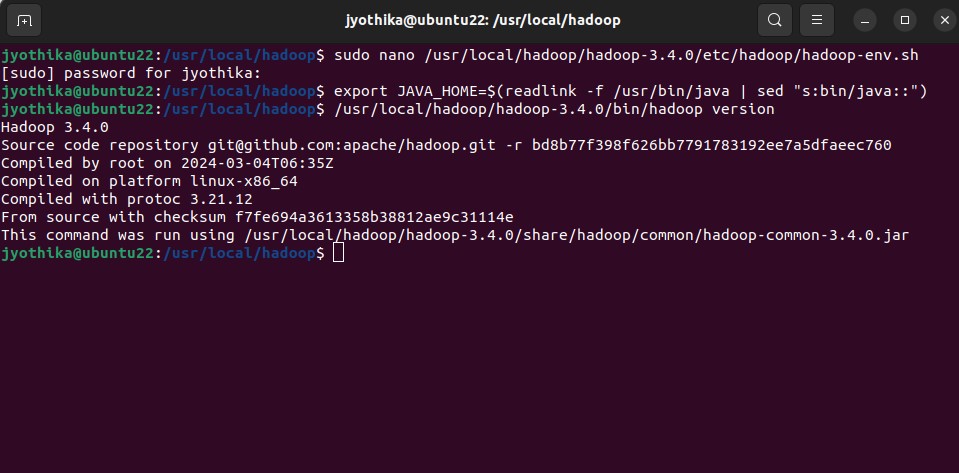
1. Export the created column to a csv file



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



**4. Screenshot of Hadoop installed**



**5 Execution of HDFS Commands for interaction with Hadoop Environment.**

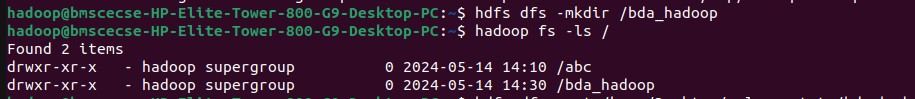
**(Minimum 10 commands to be executed)**

to start hadoop services

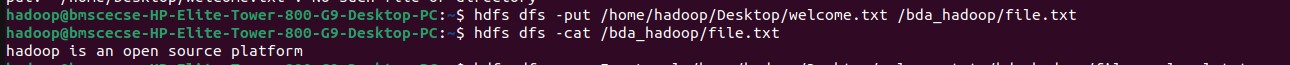
***start-all.sh jps***



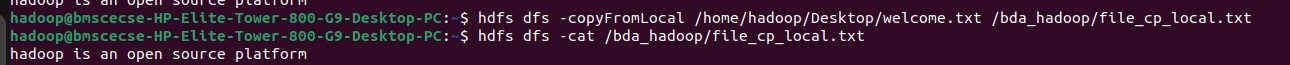
make a new directory and display the dierctory contents



use put to copy files from local to bda\_hadoop folder



use copyFromLocal to copy files from local to bda\_hadoop folder



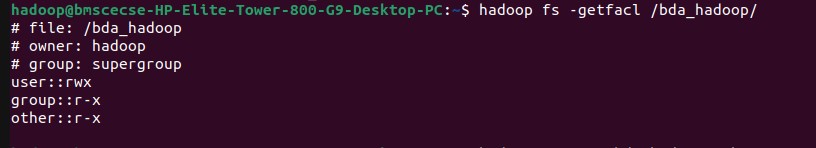
use get to copy files from hadoop folder to local



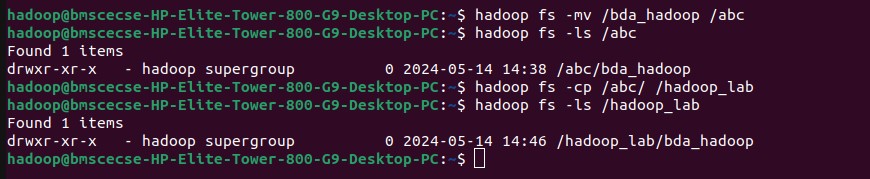
make a merged file from hadoop and store it in local desktop



use getfacl to show the access rights



move the contents of a directory to another directory in hadoop



**6. Implement Wordcount Program on Hadoop framework**

Mapper Code: You have to copy paste this program into the WCMapper Java Class file.

// Importing libraries 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

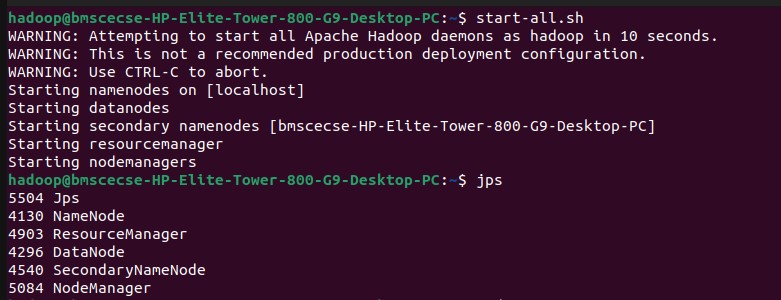
{

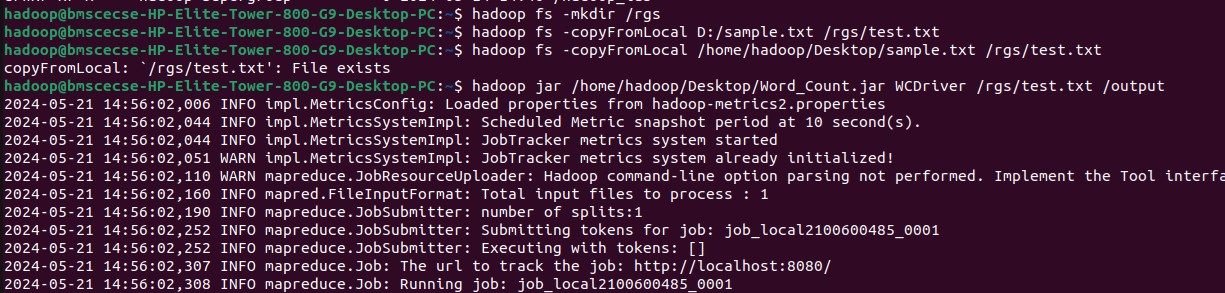
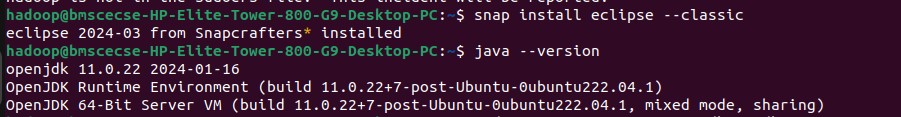
int exitCode = ToolRunner.run(new WCDriver(), args);

System.out.println(exitCode);

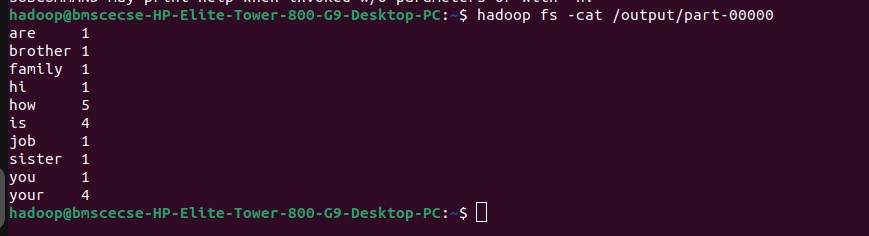
}

}





Output:



**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.**

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

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

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

Configuration conf = new Configuration(); Job job = Job.getInstance(conf, "Avg Temp"); job.setJarByClass(AvgTemp.class); job.setMapperClass(AvgTempMapper.class); job.setCombinerClass(AvgTempReducer.class); job.setReducerClass(AvgTempReducer.class); job.setOutputKeyClass(Text.class); job.setOutputValueClass(IntWritable.class); FileInputFormat.addInputPath(job,new Path(args[0]));

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

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

}

}

import java.io.\*;

import org.apache.hadoop.io.IntWritable; import org.apache.hadoop.io.Text; import org.apache.commons.lang.StringUtils;

import org.apache.hadoop.mapreduce.Mapper;

public class AvgTempMapper

extends Mapper<Object, Text, Text, IntWritable> {

public void map(Object key, Text value, Context

context)throws IOException, InterruptedException{

String[] line = value.toString().split(",");

String datePart = line[1]; String temp = line[10]; if(StringUtils.isNumeric(temp))

context.write(new Text(datePart), new IntWritable(Integer.parseInt(temp)));

}

}

import java.io.\*; import org.apache.hadoop.io.IntWritable; import org.apache.hadoop.io.Text;

import org.apache.hadoop.mapreduce.Reducer;

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

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

int sumTemps = 0; int numItems = 0;

for(IntWritable val : values) {

sumTemps += val.get();

numItems += 1;

}

context.write(key, new IntWritable(sumTemps/numItems));

}

}

Output:



**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.**

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

import org.apache.hadoop.io.LongWritable; import org.apache.hadoop.io.Text;

import org.apache.hadoop.mapreduce.Mapper;

public class top\_10\_Movies\_Mapper

extends Mapper<Object, Text, Text, LongWritable> {

private TreeMap<Long, String> tmap;

@Override

public void setup(Context context)

throws IOException, InterruptedException

{

tmap = new TreeMap<Long, String>();

}

@Override

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

{

// input data format => movie\_name

// no\_of\_views (tab separated)

// we split the input data

String[] tokens = value.toString().split("\t");

String movie\_name = tokens[0];

long no\_of\_views = Long.parseLong(tokens[1]);

// insert data into treeMap,

// we want top 10 viewed movies // so we pass no\_of\_views as key tmap.put(no\_of\_views, movie\_name);

// we remove the first key-value // if it's size increases 10 if (tmap.size() > 10) { tmap.remove(tmap.firstKey()); }

}

@Override

public void cleanup(Context context)

throws IOException, InterruptedException

{

for (Map.Entry<Long, String> entry :

tmap.entrySet()) {

long count = entry.getKey(); String name = entry.getValue();

context.write(new Text(name),

new LongWritable(count));

}

}

} import java.io.IOException; import java.util.Map; import java.util.TreeMap; import org.apache.hadoop.io.LongWritable; import org.apache.hadoop.io.Text; import org.apache.hadoop.mapreduce.Reducer;

public class top\_10\_Movies\_Reducer extends Reducer<Text, LongWritable, LongWritable,

Text> {

private TreeMap<Long, String> tmap2;

@Override

public void setup(Context context)

throws IOException, InterruptedException

{

tmap2 = new TreeMap<Long, String>();

}

@Override

public void reduce(Text key,

Iterable<LongWritable> values, Context context)

throws IOException, InterruptedException

{

// input data from mapper

// key values

// movie\_name [ count ]

String name = key.toString();

long count = 0;

for (LongWritable val : values) {

count = val.get();

}

// insert data into treeMap,

// we want top 10 viewed movies // so we pass count as key

tmap2.put(count, name);

// we remove the first key-value // if it's size increases 10 if (tmap2.size() > 10) {

tmap2.remove(tmap2.firstKey());

}

}

@Override

public void cleanup(Context context) throws IOException, InterruptedException

{

for (Map.Entry<Long, String> entry :

tmap2.entrySet()) {

long count = entry.getKey(); String name = entry.getValue(); context.write(new LongWritable(count),

new Text(name));

}

}

}

import org.apache.hadoop.conf.Configuration; import org.apache.hadoop.fs.Path; import org.apache.hadoop.io.LongWritable; 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; import org.apache.hadoop.util.GenericOptionsParser;

public class Driver {

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

{

Configuration conf = new Configuration();

String[] otherArgs

= new GenericOptionsParser(conf, args)

.getRemainingArgs();

// if less than two paths // provided will show error if (otherArgs.length & lt; 2) {

System.err.println(" Error

: please provide two paths

& quot;);

System.exit(2);

}

Job job

= Job.getInstance(conf, " top 10 & quot;); job.setJarByClass(Driver.class);

job.setMapperClass(top\_10\_Movies\_Mapper.class);

job.setReducerClass(top\_10\_Movies\_Reducer.class);

job.setMapOutputKeyClass(Text.class); job.setMapOutputValueClass(LongWritable.class);

job.setOutputKeyClass(LongWritable.class);

job.setOutputValueClass(Text.class);

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

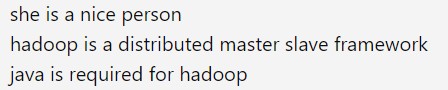
job, new Path(otherArgs[1]));

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

}

}

Input:



Output:

