**VISVESVARAYA TECHNOLOGICAL UNIVERSITY “JnanaSangama”, Belgaum -590014, Karnataka.**

****

**LAB REPORT**

**on**

**Big Data and Analytics Lab**

***Submitted by***

**G Mohammed Awaiz (1BM21CS060)**

***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**

1

**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 Lab” carried out by **G Mohammed Awaiz(1BM21CS060),** who is a 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 and Analytics - (22CS6PEBDA)** work prescribed for the said degree.

**Dr. Rekha G.S. Dr. Jyothi S Nayak** Assistant Professor Professor and Head, CSE, BMSCE, Bengaluru BMSCE, Bengaluru

`

2

**Index Sheet**

|  |  |
| --- | --- |
| **Sl.**  **No.** | **Experiment Title Page No.** |
| **1.**  **2.** | **MongoDB CRUD Operations 4**  **Cassandra Employee 9** |
| **3.** | **Cassandra Library 11** |
| **4.** | **Hadoop Installation 13** |
| **5.** | **Hadoop Commands 14** |
| **6.** | **Hadoop Word Count 16** |
| **7.** | **Map Reduce Programs 20** |
| **8.** | **Map Reduce Sort 23** |

3

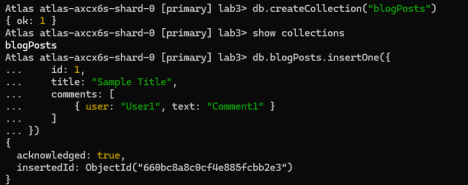
**Lab 1**

**Perform the following DB operations using MongoDB**

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

db.createCollection("blogPosts")



Demonstrate the following

**1. Adding an element into array**

db.blogPosts.insertOne({

id: 1,

title: "Sample Title",

comments: [

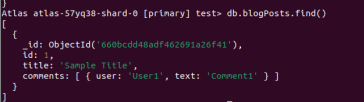
{ user: "User1", text: "Comment1" }

]

})

(Similarly, Insert 4 ids)

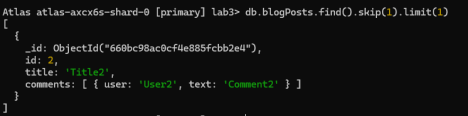
4



**2. Display second element**

db.blogPosts.find().skip(1).limit(1)

5

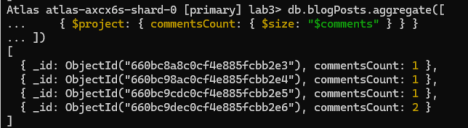


**3. Display size of the array**

db.blogPosts.aggregate([

{ $project: { commentsCount: { $size: "$comments" } } }

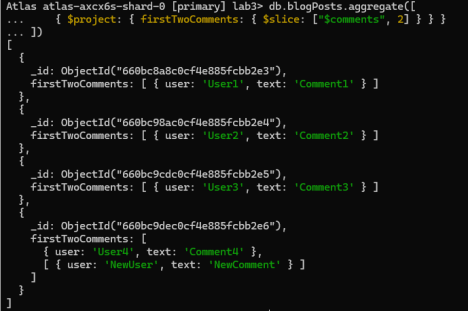
])

**4. Display first two elements of the array**

db.blogPosts.aggregate([

{ $project: { firstTwoComments: { $slice: ["$comments", 2] } } } ])

6



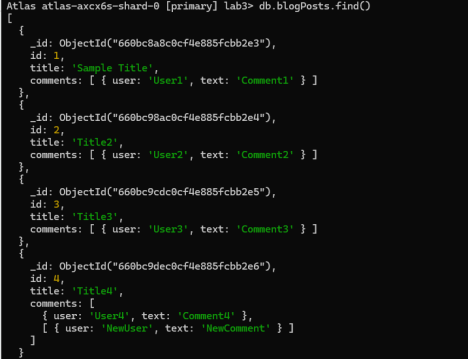
**5. Update the document with id 4 and replace the element present in 1st index position of the array with another array**

db.blogPosts.updateOne(

{ id: 4 },

{ $set: { "comments.1": [{ user: "NewUser", text: "NewComment" }] } } )

7

8

**Lab 2**

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

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

create keyspace Employee with replication = {'class':'SimpleStrategy', 'replication\_factor':1};

use 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**

create table EmployeeInfo(Emp\_Id int primary key, Emp\_Name text, Designation text, Date\_of\_Joining timestamp, Salary double, Dept\_Name text);

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

begin batch

... insert into employeeinfo (emp\_id, date\_of\_joining, dept\_name, designation, emp\_name, salary)

... values (121, '2024-03-25', 'KSC', 'Intern', 'Arvind', 0)

... insert into employeeinfo (emp\_id, date\_of\_joining, dept\_name, designation, emp\_name, salary)

... values (122, '2024-06-01', 'KSC', 'Intern', 'Aravind', 35000)

... apply batch;

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

9

update employeeinfo set emp\_name='Arvind Ashok', dept\_name='Security' where emp\_id=121;

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

cqlsh:employee> select \* from Employee\_information where emp\_id in (1,2,3) order by 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.**

cqlsh:employee> alter table employee\_info add projects set<text>;

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

cqlsh:employee> update employee\_info set

projects=projects+{'project1','project2','project3'} where emp\_id=1;

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

... insert into Employee\_Info(Emp\_id,Emp\_name,Date\_of\_Joining,Salary,Dept\_N ame) values(1,'Khushil','2021-04-23',50000,'CSE') using TTL 15

… apply batch

10

**Lab 3**

**1. Create a key space by name Library**

create keyspace Library with replication = {‘class’:’SimpleStrategy’, ‘replication\_factor’:1};

**2. Create a column family by name Library-Info with attributes Stud\_Id Primary Key, Counter\_value of type Counter,**

create table Library\_info(Stud\_id int, COunter\_value counter, Stud\_name varchar, Book\_name varchar, Book\_id int, doi date, primary key (Stud\_id, Stud\_name, Book\_id, Book\_id, doi));

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

update Library\_info set Counter\_value = Counter\_value + 1 where Stud\_id = 112 and Stud\_name = ‘Arvind’ and ‘Book\_id’=’123’ and ‘doi’=’2024-06-09’

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

update library\_info set Counter\_value = Counter\_value + 1 where Stud\_id=112 and Stud\_name=’Arvind’ and Book\_name=’abc’ and Book\_id=’123’ and doi=’2024-05-01’;

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

select counter\_value as borrow\_count form Library\_info where Stud\_if=1 and Book\_id=123

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

****11

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

****12

**Lab 4**

****13

**Lab 5**

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

hadoop@bmscecse-HP-Elite-Tower-800-G9-Desktop-PC:~$ hdfs dfs -ls /First hadoop@bmscecse-HP-Elite-Tower-800-G9-Desktop-PC:~$ hdfs dfs -put /home/hadoop/Documents/test.txt /First/test.txt

hadoop@bmscecse-HP-Elite-Tower-800-G9-Desktop-PC:~$ hdfs dfs -cat /First/test.txt Hello World!

hadoop@bmscecse-HP-Elite-Tower-800-G9-Desktop-PC:~$ hadoop fs -ls /First Found 1 items

-rw-r--r-- 1 hadoop supergroup 13 2024-05-14 14:22 /First/test.txt

hadoop@bmscecse-HP-Elite-Tower-800-G9-Desktop-PC:~$ hdfs dfs -ls /First Found 1 items

-rw-r--r-- 1 hadoop supergroup 13 2024-05-14 14:22 /First/test.txt

hadoop@bmscecse-HP-Elite-Tower-800-G9-Desktop-PC:~$ jps

7297 Jps

3860 ResourceManager

4020 NodeManager

3306 DataNode

3149 NameNode

3581 SecondaryNameNode

hadoop@bmscecse-HP-Elite-Tower-800-G9-Desktop-PC:~$ hdfs dfs -get /First/test/txt /home/hadoop/Documents/got.txt

get: `/First/test/txt': No such file or directory

hadoop@bmscecse-HP-Elite-Tower-800-G9-Desktop-PC:~$ hdfs dfs -get /First/test.txt /home/hadoop/Documents/got.txt

hadoop@bmscecse-HP-Elite-Tower-800-G9-Desktop-PC:~$ cat

home/hadoop/Documents/got.txt

cat: home/hadoop/Documents/got.txt: No such file or directory

hadoop@bmscecse-HP-Elite-Tower-800-G9-Desktop-PC:~$ cat

home/hadoop/Documents/got.txt

cat: home/hadoop/Documents/got.txt: No such file or directory

hadoop@bmscecse-HP-Elite-Tower-800-G9-Desktop-PC:~$ hdfs dfs -put /home/hadoop/Documents/test.txt /First/test1.txt

hadoop@bmscecse-HP-Elite-Tower-800-G9-Desktop-PC:~$ hdfs dfs -getmerge /First/test.txt /First/test1.txt /home/hadoop/Documents/new.txt

hadoop@bmscecse-HP-Elite-Tower-800-G9-Desktop-PC:~$ hdfs dfs -getfacl /First/ # file: /First

# owner: hadoop

# group: supergroup

user::rwx

group::r-x

14

other::r-x

hadoop@bmscecse-HP-Elite-Tower-800-G9-Desktop-PC:~$ hdfs dfs -copyToLocal /First/test1.txt /home/hadoop/Documents

hadoop@bmscecse-HP-Elite-Tower-800-G9-Desktop-PC:~$ hdfs dfs -cat /First/test1.txt Hello World!

hadoop@bmscecse-HP-Elite-Tower-800-G9-Desktop-PC:~$ hadoop fs -mv /First /FFF hadoop@bmscecse-HP-Elite-Tower-800-G9-Desktop-PC:~$ hadoop fs -ls /First /FFF ls: `/First': No such file or directory

Found 2 items

-rw-r--r-- 1 hadoop supergroup 13 2024-05-14 14:22 /FFF/test.txt

-rw-r--r-- 1 hadoop supergroup 13 2024-05-14 14:44 /FFF/test1.txt hadoop@bmscecse-HP-Elite-Tower-800-G9-Desktop-PC:~$ hadoop fs -ls /FFF Found 2 items

-rw-r--r-- 1 hadoop supergroup 13 2024-05-14 14:22 /FFF/test.txt

-rw-r--r-- 1 hadoop supergroup 13 2024-05-14 14:44 /FFF/test1.txt hadoop@bmscecse-HP-Elite-Tower-800-G9-Desktop-PC:~$ hadoop fs -ls /First ls: `/First': No such file or directory

hadoop@bmscecse-HP-Elite-Tower-800-G9-Desktop-PC:~$ hadoop fs -ls /FFF /First Found 2 items

-rw-r--r-- 1 hadoop supergroup 13 2024-05-14 14:22 /FFF/test.txt

-rw-r--r-- 1 hadoop supergroup 13 2024-05-14 14:44 /FFF/test1.txt hadoop@bmscecse-HP-Elite-Tower-800-G9-Desktop-PC:~$

hadoop@bmscecse-HP-Elite-Tower-800-G9-Desktop-PC:~$ hdfs dfs -ls /First hadoop@bmscecse-HP-Elite-Tower-800-G9-Desktop-PC:~$ hdfs dfs -put /home/hadoop/Documents/test.txt /First/test.txt

hadoop@bmscecse-HP-Elite-Tower-800-G9-Desktop-PC:~$ hdfs dfs -cat /First/test.txt Hello World!

hadoop@bmscecse-HP-Elite-Tower-800-G9-Desktop-PC:~$ hadoop fs -ls /First Found 1 items

-rw-r--r-- 1 hadoop supergroup 13 2024-05-14 14:22 /First/test.txt

hadoop@bmscecse-HP-Elite-Tower-800-G9-Desktop-PC:~$ hdfs dfs -ls /First Found 1 items

-rw-r--r-- 1 hadoop supergroup 13 2024-05-14 14:22 /First/test.txt

hadoop@bmscecse-HP-Elite-Tower-800-G9-Desktop-PC:~$ jps

7297 Jps

3860 ResourceManager

4020 NodeManager

3306 DataNode

3149 NameNode

3581 SecondaryNameNode

hadoop@bmscecse-HP-Elite-Tower-800-G9-Desktop-PC:~$ hdfs dfs -get /First/test.txt /home/hadoop/Documents/got.txt

15

**Lab 6**

Word Count

**Implement WordCount Program on Hadoop framework**

**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<LongWritable,

Text, Text, IntWritable> {

// Map function

public void map(LongWritable key, Text value, OutputCollector<Text, IntWritable> output, Reporter rep) throws IOException

{

String line = value.toString();

// Splitting the line on spaces for (String word : line.split(" "))

{

if (word.length() > 0)

{

16

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 functionpublic 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();

}

17

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

}

}

**Driver Code:**

// 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);

18

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);

}

**Lab 7**

**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**

**Driver**

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); }

}

20

**Mapper**

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));

}

}

**Reducer**

21

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));

}

}

**Lab 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.**

**Mapper.py**

import sys

# Read input from STDIN

for line in sys.stdin:

# Remove leading and trailing whitespace

line = line.strip()

# Split the line into words

words = line.split()

# Emit the word along with a count of 1

for word in words:

print(f"{word}\t1")

**Reducer.py**

import sys

from collections import defaultdict

word\_counts = defaultdict(int)

# Read input from STDIN

for line in sys.stdin:

# Remove leading and trailing whitespace

line = line.strip()

# Parse the input we got from mapper.py

word, count = line.split('\t', 1)

23

# Convert count from string to int

try:

count = int(count)

except ValueError:

continue

# Increment word count

word\_counts[word] += count

# Sort words alphabetically

sorted\_words = sorted(word\_counts.items(), key=lambda x: x[0])

# Emit the top 10 words with the highest counts

for word, count in sorted(sorted\_words, key=lambda x: -x[1])[:10]: print(f"{word}\t{count}")

**Driver.py**

import os

import subprocess

def run\_mapreduce(input\_path, output\_path, mapper\_path, reducer\_path):

# Hadoop streaming jar path - you may need to adjust this based on your Hadoop installation

hadoop\_streaming\_jar = '/usr/lib/hadoop/hadoop-streaming.jar' # Construct the Hadoop streaming command

hadoop\_command = [

'hadoop', 'jar', hadoop\_streaming\_jar,

'-input', input\_path,

'-output', output\_path,

'-mapper', mapper\_path,

'-reducer', reducer\_path,

'-combiner', reducer\_path,

'-file', mapper\_path,

'-file', reducer\_path

]

try:

# Run the Hadoop streaming command

subprocess.run(hadoop\_command, check=True)

print(f"MapReduce job completed successfully. Output is stored in {output\_path}")

except subprocess.CalledProcessError as e:

print(f"MapReduce job failed with error: {e}")

if \_\_name\_\_ == "\_\_main\_\_":

# Paths to input and output directories in HDFS

input\_path = '/path/to/input.txt'

output\_path = '/path/to/output'

# Paths to the mapper and reducer scripts

25

mapper\_path = 'mapper.py'

reducer\_path = 'reducer.py'

# Run the MapReduce job

run\_mapreduce(input\_path, output\_path, mapper\_path, reducer\_path) 26