VISVESVARAYA TECHNOLOGICAL UNIVERSITY

"JnanaSangama", Belgaum -590014, Karnataka.

LAB REPORT
on

Big Data and Analytics Lab

Submitted by

Dhruva S (1BM21CS057)

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 Lab" carried out by **Dhruva S(1BM21CS057),** 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.
Assistant Professor
BMSCE, Bengaluru

Dr. Jyothi S Nayak
Professor and Head, CSE, BMSCE, Bengaluru

`

Index Sheet

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")

```
Atlas atlas-axcx6s-shard-0 [primary] lab3> db.createCollection("blogPosts")
{ ok: 1 }
Atlas atlas-axcx6s-shard-0 [primary] lab3> show collections
blogPosts
Atlas atlas-axcx6s-shard-0 [primary] lab3> db.blogPosts.insertOne({
...     id: 1,
...     title: "Sample Title",
...     comments: [
...         { user: "User1", text: "Comment1" }
...     ]
... })
{
  acknowledged: true,
  insertedId: ObjectId("660bc8a8c0cf4e885fcbb2e3")
}
```

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)

```
}
Atlas atlas-57yq38-shard-0 [primary] test> db.blogPosts.find()
[
  {
    _id: ObjectId('660bcdd48adf462691a26f41'),
    id: 1,
    title: 'Sample Title',
    comments: [ { user: 'User1', text: 'Comment1' } ]
  }
]
```

```
Atlas atlas-axcx6s-shard-0 [primary] lab3> db.blogPosts.find()
[
  {
    _id: ObjectId("660bc8a8c0cf4e885fcbb2e3"),
    id: 1,
    title: 'Sample Title',
    comments: [ { user: 'User1', text: 'Comment1' } ]
  },
  {
    _id: ObjectId("660bc98ac0cf4e885fcbb2e4"),
    id: 2,
    title: 'Title2',
    comments: [ { user: 'User2', text: 'Comment2' } ]
  },
  {
    _id: ObjectId("660bc9cdc0cf4e885fcbb2e5"),
    id: 3,
    title: 'Title3',
    comments: [ { user: 'User3', text: 'Comment3' } ]
  },
  {
    _id: ObjectId("660bc9dec0cf4e885fcbb2e6"),
    id: 4,
    title: 'Title4',
    comments: [ { user: 'User4', text: 'Comment4' } ]
  }
```

2. Display second element

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

```
Atlas atlas-axcx6s-shard-0 [primary] lab3> db.blogPosts.find().skip(1).limit(1)
[
  {
    _id: ObjectId("660bc98ac0cf4e885fcbb2e4"),
    id: 2,
    title: 'Title2',
    comments: [ { user: 'User2', text: 'Comment2' } ]
  }
]
```

3. Display size of the array

```
db.blogPosts.aggregate([
  { $project: { commentsCount: { $size: "$comments" } } }
])
```

```
Atlas atlas-axcx6s-shard-0 [primary] lab3> db.blogPosts.aggregate([
...     { $project: { commentsCount: { $size: "$comments" } } }
... ])
[
  { _id: ObjectId("660bc8a8c0cf4e885fcbb2e3"), commentsCount: 1 },
  { _id: ObjectId("660bc98ac0cf4e885fcbb2e4"), commentsCount: 1 },
  { _id: ObjectId("660bc9cdc0cf4e885fcbb2e5"), commentsCount: 1 },
  { _id: ObjectId("660bc9dec0cf4e885fcbb2e6"), commentsCount: 2 }
]
```

4. Display first two elements of the array

```
db.blogPosts.aggregate([
  { $project: { firstTwoComments: { $slice: ["$comments", 2] } } }
])
```

```
Atlas atlas-axcx6s-shard-0 [primary] lab3> db.blogPosts.aggregate([
...     { $project: { firstTwoComments: { $slice: ["$comments", 2] } } }
... ])
[
  {
    _id: ObjectId("660bc8a8c0cf4e885fcbb2e3"),
    firstTwoComments: [ { user: 'User1', text: 'Comment1' } ]
  },
  {
    _id: ObjectId("660bc98ac0cf4e885fcbb2e4"),
    firstTwoComments: [ { user: 'User2', text: 'Comment2' } ]
  },
  {
    _id: ObjectId("660bc9cdc0cf4e885fcbb2e5"),
    firstTwoComments: [ { user: 'User3', text: 'Comment3' } ]
  },
  {
    _id: ObjectId("660bc9dec0cf4e885fcbb2e6"),
    firstTwoComments: [
      { user: 'User4', text: 'Comment4' },
      [ { user: 'NewUser', text: 'NewComment' } ]
    ]
  }
]
```

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" }] } })

```
Atlas atlas-axcx6s-shard-0 [primary] lab3> db.blogPosts.updateOne( { id: 4 }, { $set: { "comments.1": [{ user: "NewUser"
, text: "NewComment" }] } } )
{
  acknowledged: true,
  insertedId: null,
  matchedCount: 1,
  modifiedCount: 1,
  upsertedCount: 0
}
```

7

```
Atlas atlas-axcx6s-shard-0 [primary] lab3> db.blogPosts.find()
[
  {
    _id: ObjectId("660bc8a8c0cf4e885fcbb2e3"),
    id: 1,
    title: 'Sample Title',
    comments: [ { user: 'User1', text: 'Comment1' } ]
  },
  {
    _id: ObjectId("660bc98ac0cf4e885fcbb2e4"),
    id: 2,
    title: 'Title2',
    comments: [ { user: 'User2', text: 'Comment2' } ]
  },
  {
    _id: ObjectId("660bc9cdc0cf4e885fcbb2e5"),
    id: 3,
    title: 'Title3',
    comments: [ { user: 'User3', text: 'Comment3' } ]
  },
  {
    _id: ObjectId("660bc9dec0cf4e885fcbb2e6"),
    id: 4,
    title: 'Title4',
    comments: [
      { user: 'User4', text: 'Comment4' },
      [ { user: 'NewUser', text: 'NewComment' } ]
    ]
  }
```

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);
```

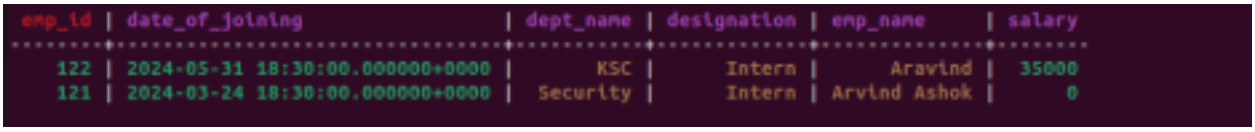
3. Insert the values into the table in batch

```
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;
```

emp_id	date_of_joining	dept_name	designation	emp_name	salary
122	2024-05-31 18:30:00.000000+0000	KSC	Intern	Aravind	35000
121	2024-03-24 18:30:00.000000+0000	KSC	Intern	Arvind	0

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

update employeeinfo set emp_name='Arvind Ashok', dept_name='Security' where emp_id=121;



emp_id	date_of_joining	dept_name	designation	emp_name	salary
122	2024-05-31 18:30:00.000000+0000	KSC	Intern	Aravind	35000
121	2024-03-24 18:30:00.000000+0000	Security	Intern	Arvind Ashok	0

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_Name) values(1,'Khushil','2021-04-23',50000,'CSE') using TTL 15

… apply batch

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’

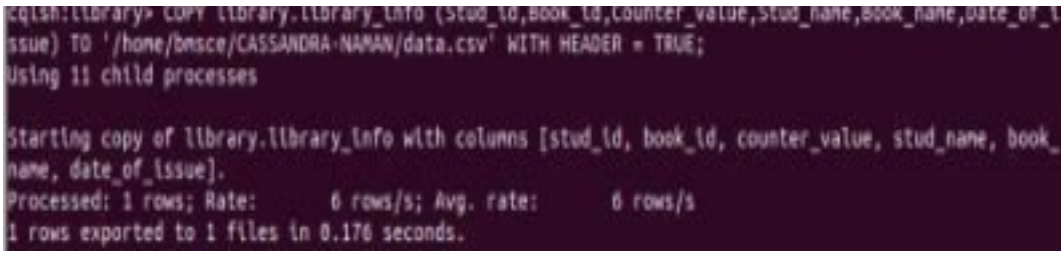
4. Display the details of the table created and increase the value of the counter

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

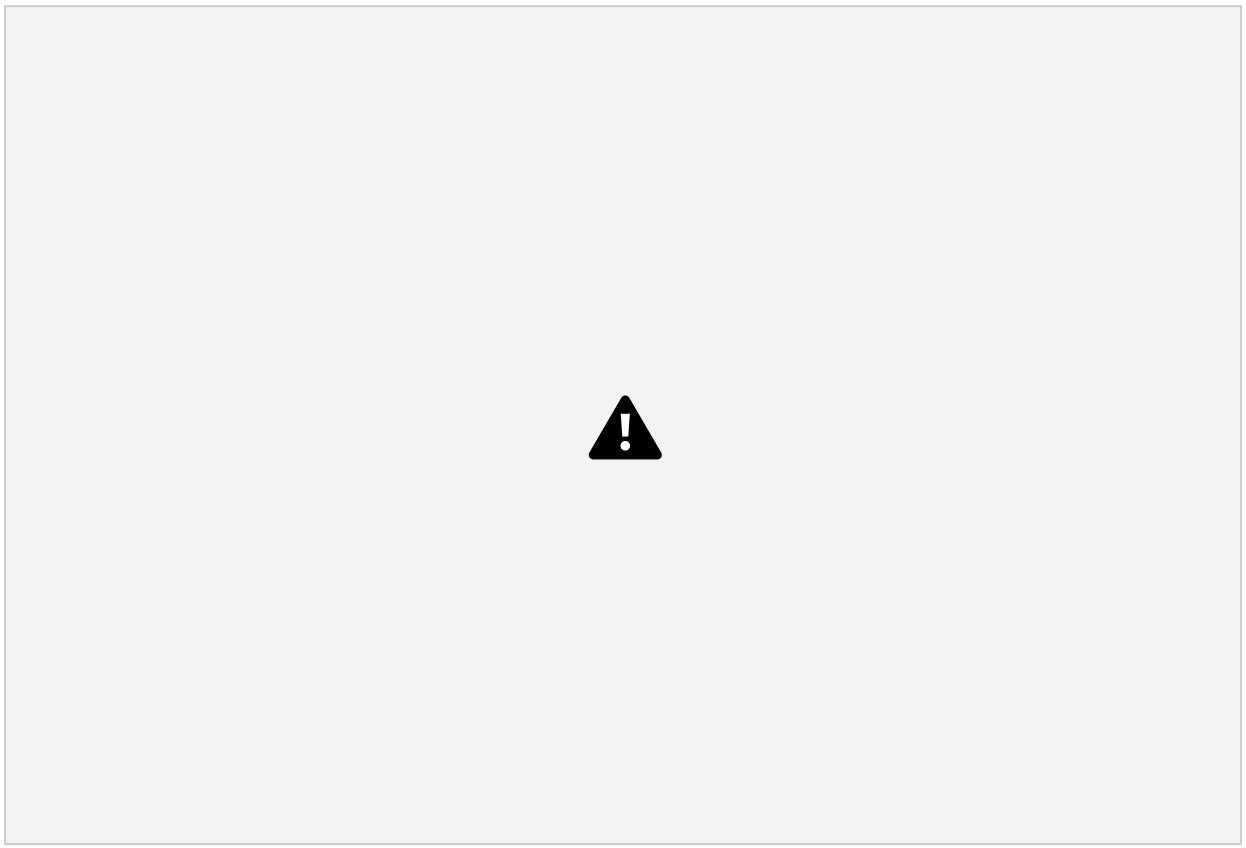
7. Import a given csv dataset from local file system into Cassandra column family

```
cqlsh:library> COPY library.library_info (Stud_Id,Book_Id,Counter_value,Stud_name,Book_name,Date_of_I
ssue) FROM '/home/bmsce/CASSANDRA-NAMAN/data.csv' WITH HEADER = TRUE;
Using 11 child processes

Starting copy of library.library_info with columns [stud_id, book_id, counter_value, stud_name, book_
name, date_of_issue].
Processed: 1 rows; Rate:       2 rows/s; Avg. rate:       3 rows/s
1 rows imported from 1 files in 0.379 seconds (0 skipped).
```

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
```

```
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
```

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)
```

{

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();

}

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);
```



```
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); }

}
```

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

```
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)
```

```
    # 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
```



```
mapper_path = 'mapper.py'
reducer_path = 'reducer.py'

# Run the MapReduce job
```

run_mapreduce(input_path, output_path, mapper_path, reducer_path) 26