

# VISVESVARAYA TECHNOLOGICAL UNIVERSITY

“JnanaSangama”, Belgaum -590014, Karnataka.



## LAB REPORT on

## Big Data Analytics

*Submitted by*

**Gagan D A (1BM21CS063)**

*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 **Gagan D A (1BM21CS063)**, 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.

**Sonika Sharma D**  
Assistant Professor

Department of CSE  
BMSCE, Bengaluru

**Dr. Jyothi S Nayak**  
Professor and Head  
Department of CSE  
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.

## 1. Perform the following DB operations using Cassandra

1. Create a keyspace by name Employee

```
cqlsh:library> CREATE KEYSPACE Employee WITH REPLICATION = { 'class' : 'SimpleStrategy', 'replication_factor' : 1 };
cqlsh:library>
```

2. Create a column family by name Employee-Info with attributes Emp\_Id Primary Key, Emp\_Name, Designation, Date\_of\_Joining, Salary, Dept\_Name

```
cqlsh:employee>
cqlsh:employee> CREATE TABLE Employee_Info (
...     Emp_Id int PRIMARY KEY,
...     Emp_Name text,
...     Designation text,
...     Date_of_Joining date,
...     Salary decimal,
...     Dept_Name text
... );
```

3. Insert the values into the table in batch

```
cqlsh:employee> BEGIN BATCH
... INSERT INTO Employee_Info (Emp_Id, Emp_Name, Designation, Date_of_Joining, Salary, Dept_Name)
... VALUES (101, 'John Doe', 'Manager', '2023-01-01', 50000, 'HR');
... INSERT INTO Employee_Info (Emp_Id, Emp_Name, Designation, Date_of_Joining, Salary, Dept_Name)
... VALUES (121, 'Jane Smith', 'Developer', '2023-02-01', 60000, 'IT');
... APPLY BATCH;
```

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

```
cqlsh:employee> UPDATE Employee_Info SET Emp_Name = 'Jane Johnson', Dept_Name = 'Engineering' WHERE Emp_Id = 121;
cqlsh:employee> SELECT * FROM Employee_Info;
```

emp_id	date_of_joining	dept_name	designation	emp_name	salary
121	2023-02-01	Engineering	Developer	Jane Johnson	60000
101	2023-01-01	HR	Manager	John Doe	50000

(2 rows)

5. Sort the details of Employee records based on salary

```
cqlsh:employee> paging off
Disabled Query paging.
cqlsh:employee> SELECT * FROM Employee_Info WHERE Emp_Id IN (121,101) ORDER BY Salary ALLOW FILTERING;
```

emp_id	salary	date_of_joining	dept_name	designation	emp_name
101	50000	2023-01-01	HR	Manager	John Doe
121	60000	2023-02-01	IT	Developer	Jane Smith

(2 rows)

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> UPDATE Employee_Info SET Projects = {'ProjectA', 'ProjectB'} WHERE Emp_Id = 101 and salary=50000;
cqlsh:employee> UPDATE Employee_Info SET Projects = {'ProjectC'} WHERE Emp_Id = 121 and salary=60000;
cqlsh:employee> select * from Employee_Info;
```

emp_id	salary	date_of_joining	dept_name	designation	emp_name	projects
121	60000	2023-02-01	IT	Developer	Jane Smith	{'ProjectC'}
101	50000	2023-01-01	HR	Manager	John Doe	{'ProjectA', 'ProjectB'}

(2 rows)

7. Update the altered table to add project names.

```
cqlsh:employee> UPDATE Employee_Info SET Projects = {'ProjectA', 'ProjectB'} WHERE Emp_Id = 101 and salary=50000;
cqlsh:employee> UPDATE Employee_Info SET Projects = {'ProjectC'} WHERE Emp_Id = 121 and salary=60000;
cqlsh:employee> select * from Employee_Info;
```

emp_id	salary	date_of_joining	dept_name	designation	emp_name	projects
121	60000	2023-02-01	IT	Developer	Jane Smith	{'ProjectC'}
101	50000	2023-01-01	HR	Manager	John Doe	{'ProjectA', 'ProjectB'}

(2 rows)

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

```
cqlsh:employee> INSERT INTO Employee_Info (Emp_Id, Emp_Name, Designation, Date_of_Joining, Salary, Dept_Name) VALUES (102, 'Jane Smith', 'Developer', '2022-06-03', 60000, 'IT') USING TTL 15;
cqlsh:employee> select ttl(Emp_Name) from Employee_Info where Emp_id=102;
```

ttl(emp_name)
14

(1 rows)

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

### 1. Create a keyspace by name Library

```
cqlsh> CREATE KEYSPACE Library WITH REPLICATION = { 'class' : 'SimpleStrategy', 'replication_factor' : 1 };
cqlsh> show keyspaces;
Improper show command.
cqlsh> use Library;
cqlsh:library> |
```

### 2. 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

```
cqlsh:library> CREATE TABLE Library_Info (Stud_Id int PRIMARY KEY,Counter_value counter,Stud_Name text,Book_Name text,Book_Id text,Date_of_issue timestamp);
InvalidRequest: Error from server: code=2290 [Invalid query] message="Cannot mix counter and non counter columns in the same table"
cqlsh:library> CREATE TABLE Library_Info (
... Stud_Id int PRIMARY KEY,
... Stud_Name text,
... Book_Name text,
... Book_Id text,
... Date_of_issue timestamp
... );
cqlsh:library> CREATE TABLE Library_Counters (
... Stud_Id int PRIMARY KEY,
... Counter_value counter
... );
cqlsh:library>
```

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

```
cqlsh:library> BEGIN BATCH
... INSERT INTO Library_Info (Stud_Id, Stud_Name, Book_Name, Book_Id, Date_of_issue) VALUES (112, 'John Doe', 'BDA', 'B001', '2023-01-01');
... INSERT INTO Library_Info (Stud_Id, Stud_Name, Book_Name, Book_Id, Date_of_issue) VALUES (113, 'Jane Smith', 'ML', 'B002', '2023-01-02');
... APPLY BATCH;
```

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

```
cqlsh:library> SELECT * FROM Library_Info;

stud_id | book_id | book_name | date_of_issue | stud_name
-----+-----+-----+-----+-----
113 | B002 | ML | 2023-01-02 00:00:00.000000+0000 | Jane Smith
112 | B001 | BDA | 2023-01-01 00:00:00.000000+0000 | John Doe

(2 rows)
cqlsh:library> SELECT * FROM Library_Counters;

stud_id | counter_value
-----+-----
113 | 1
112 | 1

(2 rows)
```

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

```
cqlsh:library> UPDATE Library_Counters SET Counter_value = Counter_value + 1 WHERE Stud_Id = 112;
cqlsh:library> SELECT * FROM Library_Counters WHERE Stud_Id = 112;

stud_id | counter_value
-----+-----
112 | 2

(1 rows)
```

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

```
cqlsh:library> COPY Library_Info (Stud_Id, Stud_Name, Book_Name, Book_Id, Date_of_issue) TO 'file.csv' WITH HEADER = TRUE;
Using 11 child processes

Starting copy of library.library_info with columns [stud_id, stud_name, book_name, book_id, date_of_issue].
Processed: 2 rows; Rate:      10 rows/s; Avg. rate:      6 rows/s
2 rows exported to 1 files in 0.374 seconds.
cqlsh:library> COPY Library_Counters (Stud_Id, Counter_value) FROM 'library_counters.csv' WITH HEADER = TRUE;
Using 11 child processes
```

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

```
cqlsh:library> copy library_info(Stud_Id,Stud_Name,Book_Name,Book_Id,Date_of_issue) from 'file.csv' with header=true;
Using 7 child processes

Starting copy of library.library_info with columns [stud_id, stud_name, book_name, book_id, date_of_issue].
Processed: 2 rows; Rate:      2 rows/s; Avg. rate:      4 rows/s
2 rows imported from 1 files in 0.513 seconds (0 skipped).
cqlsh:library> select * from library_info;
```

stud_id	book_id	book_name	date_of_issue	stud_name
113	B002	ML	2023-01-02 00:00:00.000000+0000	Jane Smith
112	B001	BDA	2023-01-01 00:00:00.000000+0000	John Doe

### 3. MongoDB- CRUD Demonstration

#### SETUP:

```
C:\Users\student>
C:\Users\student>mongosh "mongodb+srv://cluster0.ddhftxd.mongodb.net/" --apiVersion 1 --username shravanics21
Enter password: *****
Current Mongosh Log ID: 660a82917c840f42b4a0552f
Connecting to:      mongodb+srv://<credentials>@cluster0.ddhftxd.mongodb.net/?appName=mongosh+2.0.0
Using MongoDB:      7.0.7 (API Version 1)
Using Mongosh:      2.0.0
mongosh 2.2.2 is available for download: https://www.mongodb.com/try/download/shell
For mongosh info see: https://docs.mongodb.com/mongodb-shell/
```

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

```
Atlas atlas-b6pfyk-shard-0 [primary] test> db.createCollection("Student");
{ ok: 1 }
```

2. Insert appropriate values(at least 5)

```
Atlas atlas-b6pfyk-shard-0 [primary] test> db.Student.insert({RollNo:1,Age:21,Cont:9876,email:"antara.de9@gmail.com"});
DeprecationWarning: Collection.insert() is deprecated. Use insertOne, insertMany, or bulkWrite.
{
  acknowledged: true,
  insertedIds: { '0': ObjectId("660a82ec7c840f42b4a05530") }
}
Atlas atlas-b6pfyk-shard-0 [primary] test>
Atlas atlas-b6pfyk-shard-0 [primary] test> db.Student.insert({RollNo:2,Age:22,Cont:9976,email:"anushka.de9@gmail.com"});
{
  acknowledged: true,
  insertedIds: { '0': ObjectId("660a82ed7c840f42b4a05531") }
}
Atlas atlas-b6pfyk-shard-0 [primary] test>
Atlas atlas-b6pfyk-shard-0 [primary] test> db.Student.insert({RollNo:3,Age:21,Cont:5576,email:"anubhav.de9@gmail.com"});
{
  acknowledged: true,
  insertedIds: { '0': ObjectId("660a82ed7c840f42b4a05532") }
}
Atlas atlas-b6pfyk-shard-0 [primary] test>
Atlas atlas-b6pfyk-shard-0 [primary] test> db.Student.insert({RollNo:4,Age:20,Cont:4476,email:"pani.de9@gmail.com"});
{
  acknowledged: true,
  insertedIds: { '0': ObjectId("660a82ed7c840f42b4a05533") }
}
Atlas atlas-b6pfyk-shard-0 [primary] test>
Atlas atlas-b6pfyk-shard-0 [primary] test> db.Student.insert({RollNo:10,Age:23,Cont:2276,email:"rekha.de9@gmail.com"});
{
  acknowledged: true,
```



```
Atlas atlas-b6pfyk-shard-0 [primary] test> db.Student.insert({RollNo:10, Age:23, Cont:2276, email:"rekha.de9@gmail.com"});
{
  acknowledged: true,
  insertedIds: { '0': ObjectId("660a82f47c840f42b4a05534") }
}
```

### 3. View the data

```
Atlas atlas-b6pfyk-shard-0 [primary] test> db.Student.find()
[
  {
    _id: ObjectId("660a82ec7c840f42b4a05530"),
    RollNo: 1,
    Age: 21,
    Cont: 9876,
    email: 'antara.de9@gmail.com'
  },
  {
    _id: ObjectId("660a82ed7c840f42b4a05531"),
    RollNo: 2,
    Age: 22,
    Cont: 9976,
    email: 'anushka.de9@gmail.com'
  },
  {
    _id: ObjectId("660a82ed7c840f42b4a05532"),
    RollNo: 3,
    Age: 21,
    Cont: 5576,
    email: 'anubhav.de9@gmail.com'
  },
  {
    _id: ObjectId("660a82ed7c840f42b4a05533"),
    RollNo: 4,
    Age: 20,
    Cont: 4476,
    email: 'pani.de9@gmail.com'
  },
  {
    _id: ObjectId("660a82f47c840f42b4a05534"),
    RollNo: 10,
    Age: 23,
    Cont: 2276,
    email: 'rekha.de9@gmail.com'
  }
]
```

### 4. Write query to update Email-Id of a student with rollno 10.

```
{
  _id: ObjectId("660a83337c840f42b4a05535"),
  RollNo: 11,
  Age: 22,
  Name: 'ABC',
  Cont: 2276,
  email: 'rea.de9@gmail.com'
}
```

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

```
Atlas atlas-b6pfyk-shard-0 [primary] test> db.Student.update({RollNo:11,Name:"ABC"},{$set:{Name:"FEM"}})
{
  acknowledged: true,
  insertedId: null,
  matchedCount: 1,
  modifiedCount: 1,
  upsertedCount: 0
}
```

6. Drop the table

```
Atlas atlas-b6pfyk-shard-0 [primary] test> db.Student.drop();
true
```

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

```
Atlas atlas-b6pfyk-shard-0 [primary] test> db.createCollection("Customers");
{ ok: 1 }
```

2. Insert at least 5 values into the table

```
Atlas atlas-b6pfyk-shard-0 [primary] test> db.Customers.insert({cust_id:1,Balance:200, Type:"S"});
{
  acknowledged: true,
  insertedIds: { '0': ObjectId("660a83b47c840f42b4a05536") }
}
Atlas atlas-b6pfyk-shard-0 [primary] test>
Atlas atlas-b6pfyk-shard-0 [primary] test> db.Customers.insert({cust_id:1,Balance:1000, Type:"Z"})
{
  acknowledged: true,
  insertedIds: { '0': ObjectId("660a83b47c840f42b4a05537") }
}
Atlas atlas-b6pfyk-shard-0 [primary] test>
Atlas atlas-b6pfyk-shard-0 [primary] test> db.Customers.insert({cust_id:2,Balance:100, Type:"Z"});
{
  acknowledged: true,
  insertedIds: { '0': ObjectId("660a83b47c840f42b4a05538") }
}
Atlas atlas-b6pfyk-shard-0 [primary] test>
Atlas atlas-b6pfyk-shard-0 [primary] test> db.Customers.insert({cust_id:2,Balance:1000, Type:"C"});
{
  acknowledged: true,
  insertedIds: { '0': ObjectId("660a83b57c840f42b4a05539") }
}
Atlas atlas-b6pfyk-shard-0 [primary] test>
Atlas atlas-b6pfyk-shard-0 [primary] test> db.Customers.insert({cust_id:2,Balance:500, Type:"C"});
{
  acknowledged: true,
  insertedIds: { '0': ObjectId("660a83b57c840f42b4a0553a") }
}
Atlas atlas-b6pfyk-shard-0 [primary] test>
Atlas atlas-b6pfyk-shard-0 [primary] test> db.Customers.insert({cust_id:2,Balance:50, Type:"S"});
{
  acknowledged: true,
  insertedIds: { '0': ObjectId("660a83b57c840f42b4a0553b") }
}
Atlas atlas-b6pfyk-shard-0 [primary] test>
Atlas atlas-b6pfyk-shard-0 [primary] test> db.Customers.insert({cust_id:3,Balance:500, Type:"Z"});
{
  acknowledged: true,
  insertedIds: { '0': ObjectId("660a83b77c840f42b4a0553c") }
}
```

```
Atlas atlas-b6pfyk-shard-0 [primary] test> db.Customers.insert({cust_id:2,Balance:50, Type:"S"});
{
  acknowledged: true,
  insertedIds: { '0': ObjectId("660a83b57c840f42b4a0553b") }
}
```

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

```
Atlas atlas-b6pfyk-shard-0 [primary] test> db.Customers.aggregate (
... {$match:{Type:"Z"}},
...
... {$group : { _id : "$cust_id",
...
... TotAccBal :{$sum:"$Balance"} } },
... {$match:{TotAccBal:{$gt:1200}}});
```

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

```
Atlas atlas-b6pfyk-shard-0 [primary] test> db.Customers.aggregate (
...
... {$group : { _id : "$cust_id",
...
... minAccBal :{$min:"$Balance"},
... maxAccBal :{$max:"$Balance"} });
[
  { _id: 2, minAccBal: 50, maxAccBal: 1000 },
  { _id: 1, minAccBal: 200, maxAccBal: 1000 },
  { _id: 3, minAccBal: 500, maxAccBal: 500 }
]
```

5. Drop the table

```
Atlas atlas-b6pfyk-shard-0 [primary] test> db.Customers.drop()
true
```

## 4. Screenshot of Hadoop installed

```
Command Prompt
Microsoft Windows [Version 10.0.17134.648]
(c) 2018 Microsoft Corporation. All rights reserved.

C:\Users\hp>hadoop version
Hadoop 3.1.0
Source code repository: https://github.com/apache/hadoop - r.16b70610a34c4dcf5d3b05cf4b58ca77238cbe5d
```

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

1. mkdir
2. ls

```
hadoop@bmscscse-HP-Elite-Tower-800-G9-Desktop-PC:~$ start-all.sh
WARNING: Attempting to start all Apache Hadoop daemons as hadoop in 10 seconds.
WARNING: This is not a recommended production deployment configuration.
WARNING: Use CTRL-C to abort.
Starting namenodes on [localhost]
Starting datanodes
Starting secondary namenodes [bmscscse-HP-Elite-Tower-800-G9-Desktop-PC]
Starting resourcemanager
Starting nodemanagers
hadoop@bmscscse-HP-Elite-Tower-800-G9-Desktop-PC:~$ hdfs dfs -mkdir /bda_hadoop
hadoop@bmscscse-HP-Elite-Tower-800-G9-Desktop-PC:~$ hdfs dfs -ls /
Found 1 items
drwxr-xr-x - hadoop supergroup 0 2024-05-13 14:37 /bda_hadoop
```

3. put

```
hadoop@bmscscse-HP-Elite-Tower-800-G9-Desktop-PC:~$ hdfs dfs -put /home/hadoop/Desktop/bda_local.txt /bda_hadoop/file.txt
hadoop@bmscscse-HP-Elite-Tower-800-G9-Desktop-PC:~$ hdfs dfs -ls /bda_hadoop
Found 1 items
-rw-r--r-- 1 hadoop supergroup 9 2024-05-13 14:42 /bda_hadoop/file.txt
hadoop@bmscscse-HP-Elite-Tower-800-G9-Desktop-PC:~$ hdfs dfs -cat /bda_hadoop/file.txt
Hello!!!
```

4. copyFromLocal

```
hadoop@bmscscse-HP-Elite-Tower-800-G9-Desktop-PC:~$ hdfs dfs -copyFromLocal /home/hadoop/Desktop/bda_local.txt /bda_hadoop/file_cp_local.txt
hadoop@bmscscse-HP-Elite-Tower-800-G9-Desktop-PC:~$ hdfs dfs -cat /bda_hadoop/file_cp_local.txt
Hello!!!
hadoop@bmscscse-HP-Elite-Tower-800-G9-Desktop-PC:~$
```

5. get

```
hadoop@bmscscse-HP-Elite-Tower-800-G9-Desktop-PC:~$ hdfs dfs -get /bda_hadoop/file.txt /home/hadoop/Desktop/downloaded_file.txt
hadoop@bmscscse-HP-Elite-Tower-800-G9-Desktop-PC:~$ hdfs dfs -getmerge /bda_hadoop/file.txt /bda_hadoop/file_cp_local.txt /home/hadoop/Desktop/downloaded_file.txt
hadoop@bmscscse-HP-Elite-Tower-800-G9-Desktop-PC:~$ hdfs dfs -cat /bda_hadoop/file_cp_local.txt
Hello!!!
```

```
hadoop@bmscecse-HP-Elite-Tower-800-G9-Desktop-PC:~$ hadoop fs -getfacl /bda_hadoop/
# file: /bda_hadoop
# owner: hadoop
# group: supergroup
user::rwx
group::r-x
other::r-x
```

## 6. copyToLocal

```
hadoop@bmscecse-HP-Elite-Tower-800-G9-Desktop-PC:~$ hdfs dfs -copyToLocal /bda_hadoop/file.txt /home/hadoop/Desktop
hadoop@bmscecse-HP-Elite-Tower-800-G9-Desktop-PC:~$ hadoop fs -mv /bda_hadoop /abc
hadoop@bmscecse-HP-Elite-Tower-800-G9-Desktop-PC:~$ hadoop fs -ls /abc
Found 2 items
-rw-r--r-- 1 hadoop supergroup          9 2024-05-13 14:42 /abc/file.txt
-rw-r--r-- 1 hadoop supergroup          9 2024-05-13 14:52 /abc/file_cp_local.txt
hadoop@bmscecse-HP-Elite-Tower-800-G9-Desktop-PC:~$ hadoop fs -cp /hello/ /hadoop_lab
cp: '/hello/': No such file or directory
hadoop@bmscecse-HP-Elite-Tower-800-G9-Desktop-PC:~$
```

## 7. cat

```
hadoop@bmscecse-HP-Elite-Tower-800-G9-Desktop-PC:~$ hdfs dfs -cat /bda_hadoop/file_cp_local.txt
Hello!!!
hadoop@bmscecse-HP-Elite-Tower-800-G9-Desktop-PC:~$
```

## 8. mv

```
hadoop@bmscecse-HP-Elite-Tower-800-G9-Desktop-PC:~$ hadoop fs -mv /bda_hadoop /abc
hadoop@bmscecse-HP-Elite-Tower-800-G9-Desktop-PC:~$ hadoop fs -ls /abc
Found 2 items
-rw-r--r-- 1 hadoop supergroup          9 2024-05-13 14:42 /abc/file.txt
-rw-r--r-- 1 hadoop supergroup          9 2024-05-13 14:52 /abc/file_cp_local.txt
```

## 9. cp

```
hadoop@bmscecse-HP-Elite-Tower-800-G9-Desktop-PC:~$ hadoop fs -cp /hello/ /hadoop_lab
cp: '/hello/': No such file or directory
hadoop@bmscecse-HP-Elite-Tower-800-G9-Desktop-PC:~$
```

## 6. 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<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: 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<Text,
IntWritable, Text, IntWritable> {
```

```
IntWritable, Text, IntWritable> {
```

```
// Reduce function
```



```
public void reduce(Text key, Iterator<IntWritable> value,
```

```
OutputCollector<Text, IntWritable> output,
```

```
Reporter rep) throws IOException
```

```
{  
int count = 0;  
// Counting the frequency of each words  
while (value.hasNext())  
{  
IntWritable i = value.next();  
count += i.get();  
}  
output.collect(key, new IntWritable(count));  
} }
```

Driver Code: 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 < 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

```

2021-04-24 14:55:13,844 INFO common.Storage: Storage directory C:\hadoop-3.3.0\data\namenode has been successfully formatted.
2021-04-24 14:55:13,895 INFO namenode.FSImageFormatProtobuf: Saving image file C:\hadoop-3.3.0\data\namenode\current\fsimage.ckpt_00000000000000000000 using no compression
2021-04-24 14:55:14,002 INFO namenode.FSImageFormatProtobuf: Image file C:\hadoop-3.3.0\data\namenode\current\fsimage.ckpt_00000000000000000000 of size 402 bytes saved in 0 seconds .
2021-04-24 14:55:14,115 INFO namenode.NNStorageRetentionManager: Going to retain 1 images with txid >= 0
2021-04-24 14:55:14,121 INFO namenode.FSImage: FSImageSaver clean checkpoint: txid=0 when meet shutdown.
2021-04-24 14:55:14,121 INFO namenode.NameNode: SHUTDOWN_MSG:
/*****
SHUTDOWN_MSG: Shutting down NameNode at LAPTOP-JG329ESD/192.168.56.1
*****/

C:\hadoop-3.3.0\sbin>start-dfs

C:\hadoop-3.3.0\sbin>start-yarn
starting yarn daemons

C:\hadoop-3.3.0\sbin>jps
12276 NameNode
14776 DataNode
15512 NodeManager
1800 Jps
6764 ResourceManager

C:\hadoop-3.3.0\sbin>hdfs dfs -mkdir /input_dir

C:\hadoop-3.3.0\sbin>hdfs dfs -ls /
Found 1 items
drwxr-xr-x - Anusree supergroup 0 2021-04-24 14:56 /input_dir

C:\hadoop-3.3.0\sbin>hdfs dfs -copyFromLocal C:\input_file.txt /input_dir

```



```

C:\hadoop-3.3.0\sbin>hdfs dfs -cat /input_dir/input_file.txt
Hello World
Hello Hadoop
This is Hadoop test file
C:\hadoop-3.3.0\sbin>hadoop jar C:\MapReduceClient.jar wordcount /input_dir /output_dir
2021-04-24 15:24:57,242 INFO client.DefaultNoHARMFailoverProxyProvider: Connecting to ResourceManager at /0.0.0.0:8032
2021-04-24 15:24:57,714 INFO mapreduce.JobResourceUploader: Disabling Erasure Coding for path: /tmp/hadoop-yarn/staging/Anusree/.staging/job_1619256355508_0002
2021-04-24 15:24:58,387 INFO input.FileInputFormat: Total input files to process : 1
2021-04-24 15:24:58,809 INFO mapreduce.JobSubmitter: number of splits:1
2021-04-24 15:24:59,255 INFO mapreduce.JobSubmitter: Submitting tokens for job: job_1619256355508_0002
2021-04-24 15:24:59,255 INFO mapreduce.JobSubmitter: Executing with tokens: []
2021-04-24 15:24:59,450 INFO conf.Configuration: resource-types.xml not found
2021-04-24 15:24:59,451 INFO resource.ResourceUtils: Unable to find 'resource-types.xml'.
2021-04-24 15:24:59,533 INFO impl.YarnClientImpl: Submitted application application_1619256355508_0002
2021-04-24 15:24:59,581 INFO mapreduce.Job: The url to track the job: http://LAPTOP-JG329ESD:8088/proxy/application_1619256355508_0002/
2021-04-24 15:24:59,582 INFO mapreduce.Job: Running job: job_1619256355508_0002
2021-04-24 15:25:12,857 INFO mapreduce.Job: Job job_1619256355508_0002 running in uber mode : false
2021-04-24 15:25:12,861 INFO mapreduce.Job: map 0% reduce 0%
2021-04-24 15:25:19,985 INFO mapreduce.Job: map 100% reduce 0%
2021-04-24 15:25:26,077 INFO mapreduce.Job: map 100% reduce 100%
2021-04-24 15:25:32,181 INFO mapreduce.Job: Job job_1619256355508_0002 completed successfully
2021-04-24 15:25:32,284 INFO mapreduce.Job: Counters: 54
    File System Counters
        FILE: Number of bytes read=85
        FILE: Number of bytes written=530945
        FILE: Number of read operations=0
        FILE: Number of large read operations=0
        FILE: Number of write operations=0
        HDFS: Number of bytes read=162
        HDFS: Number of bytes written=51

```

```

C:\hadoop-3.3.0\sbin>hdfs dfs -cat /output_dir/*
Hadoop 2
Hello 2
This 1
World 1
file 1
is 1
test 1

C:\hadoop-3.3.0\sbin>

```

**7. From the following link extract the weather data**  
**[https://github.com/tomwhite/hadoop-](https://github.com/tomwhite/hadoop-Book/tree/master/input/ncdc/all)**

**Book/tree/master/input/ncdc/all**  
**Create a Map Reduce program to**

**a) find average temperature for each year from NCDC data set.**

#### **AverageDriver**

```
package temp;

import org.apache.hadoop.fs.Path;
import org.apache.hadoop.io.IntWritable;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapreduce.Job;
import org.apache.hadoop.mapreduce.lib.input.FileInputFormat;
import org.apache.hadoop.mapreduce.lib.output.FileOutputFormat;
public class AverageDriver {
    public static void main(String[] args) throws Exception
    { if (args.length != 2) {
        System.err.println("Please Enter the input and output parameters");
        System.exit(-1);
    }
    Job job = new Job();
    job.setJarByClass(AverageDriver.class);
    job.setJobName("Max temperature");
    FileInputFormat.addInputPath(job, new Path(args[0]));
    FileOutputFormat.setOutputPath(job, new Path(args[1]));
    job.setMapperClass(AverageMapper.class);
    job.setReducerClass(AverageReducer.class);
    job.setOutputKeyClass(Text.class);
    job.setOutputValueClass(IntWritable.class);
    System.exit(job.waitForCompletion(true) ? 0 : 1);
    }
}
```

#### **AverageMapper**

```
package temp;
import java.io.IOException;
import org.apache.hadoop.io.IntWritable;
import org.apache.hadoop.io.LongWritable;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapreduce.Mapper;
public class AverageMapper extends Mapper<LongWritable, Text, Text, IntWritable>
{ public static final int MISSING = 9999;
```

```

public void map(LongWritable key, Text value, Mapper<LongWritable, Text, Text,
IntWritable> context) throws IOException, InterruptedException {
    int temperature;
    String line = value.toString();
    String year = line.substring(15, 19);
    if (line.charAt(87) == '+') {
        temperature = Integer.parseInt(line.substring(88, 92));
    } else {
        temperature = Integer.parseInt(line.substring(87, 92));
    }
    String quality = line.substring(92, 93);
    if (temperature != 9999 && quality.matches("[01459]"))
        context.write(new Text(year), new IntWritable(temperature));
}
}

```

### **AverageReducer**

```

package temp;
import java.io.IOException;
import org.apache.hadoop.io.IntWritable;
import org.apache.hadoop.io.Text;

import org.apache.hadoop.mapreduce.Reducer;
public class AverageReducer extends Reducer<Text, IntWritable, Text, IntWritable>
{ public void reduce(Text key, Iterable<IntWritable> values, Reducer<Text,
IntWritable, Text, IntWritable> context) 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**

```
C:\hadoop-3.3.0\sbin>hdfs dfs -ls /avgtemp_outputdir
Found 2 items
-rw-r--r--  1 Anusree supergroup      0 2021-05-15 14:53 /avgtemp_outputdir/_SUCCESS
-rw-r--r--  1 Anusree supergroup      8 2021-05-15 14:53 /avgtemp_outputdir/part-r-00000

C:\hadoop-3.3.0\sbin>hdfs dfs -cat /avgtemp_outputdir/part-r-00000
1901    46

C:\hadoop-3.3.0\sbin>
```

## **b) find the mean max temperature for every month**

### **MeanMaxDriver.class**

```
package meanmax;
import org.apache.hadoop.fs.Path;
import org.apache.hadoop.io.IntWritable;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapreduce.Job;
import org.apache.hadoop.mapreduce.lib.input.FileInputFormat;
import org.apache.hadoop.mapreduce.lib.output.FileOutputFormat;
public class MeanMaxDriver {
public static void main(String[] args) throws Exception
{ if (args.length != 2) {
```

---

```

System.err.println("Please Enter the input and output parameters");
System.exit(-1);
}
Job job = new Job();
job.setJarByClass(MeanMaxDriver.class);
job.setJobName("Max temperature");
FileInputFormat.addInputPath(job, new Path(args[0]));
FileOutputFormat.setOutputPath(job, new Path(args[1]));
job.setMapperClass(MeanMaxMapper.class);
job.setReducerClass(MeanMaxReducer.class);
job.setOutputKeyClass(Text.class);
job.setOutputValueClass(IntWritable.class);
System.exit(job.waitForCompletion(true) ? 0 : 1);
}
}

```

### **MeanMaxMapper.class**

```

package meanmax;
import java.io.IOException;
import org.apache.hadoop.io.IntWritable;
import org.apache.hadoop.io.LongWritable;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapreduce.Mapper;
public class MeanMaxMapper extends Mapper<LongWritable, Text, Text, IntWritable> {
    public static final int MISSING = 9999;
    public void map(LongWritable key, Text value, Mapper<LongWritable, Text, Text,
    IntWritable>.Context context) throws IOException, InterruptedException {
        int temperature;
        String line = value.toString();
        String month = line.substring(19, 21);
        if (line.charAt(87) == '+') {
            temperature = Integer.parseInt(line.substring(88, 92));
        } else {
            temperature = Integer.parseInt(line.substring(87, 92));
        }
        String quality = line.substring(92, 93);
        if (temperature != 9999 && quality.matches("[01459]"))
            context.write(new Text(month), new IntWritable(temperature));
        }
    }
}

```

### **MeanMaxReducer.class**

```

package meanmax;
import java.io.IOException;
import org.apache.hadoop.io.IntWritable;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapreduce.Reducer;
public class MeanMaxReducer extends Reducer<Text, IntWritable, Text, IntWritable>
{ public void reduce(Text key, Iterable<IntWritable> values, Reducer<Text,
IntWritable, Text, IntWritable>, Context context) throws IOException, InterruptedException
{
int max_temp = 0;
int total_temp = 0;
int count = 0;
int days = 0;
for (IntWritable value : values)
{ int temp = value.get();
if (temp > max_temp)
max_temp = temp;
count++;
if (count == 3)
{ total_temp += max_temp;
max_temp = 0;
count = 0;
days++;
}
}
context.write(key, new IntWritable(total_temp / days));
}
}

```

OUTPUT

```
C:\hadoop-3.3.0\sbin>hdfs dfs -cat /meanmax_output/*
```

```
01      4  
02      0  
03      7  
04     44  
05    100  
06    168  
07    219  
08    198  
09    141  
10    100  
11     19  
12      3
```

```
C:\hadoop-3.3.0\sbin>
```

## 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.

```
package samples.topn;
import java.io.IOException;
import java.util.StringTokenizer;
import org.apache.hadoop.conf.Configuration;
import org.apache.hadoop.fs.Path;
import org.apache.hadoop.io.IntWritable;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapreduce.Job;
import org.apache.hadoop.mapreduce.Mapper;
import org.apache.hadoop.mapreduce.lib.input.FileInputFormat;
import org.apache.hadoop.mapreduce.lib.output.FileOutputFormat;
import org.apache.hadoop.util.GenericOptionsParser;
public class TopN {
    public static void main(String[] args) throws Exception
    { Configuration conf = new Configuration();
    String[] otherArgs = (new GenericOptionsParser(conf, args)).getRemainingArgs();
    if (otherArgs.length != 2) {
        System.err.println("Usage: TopN <in><out>");
        System.exit(2);
    }
    Job job = Job.getInstance(conf);
    job.setJobName("Top N");
    job.setJarByClass(TopN.class);
    job.setMapperClass(TopNMapper.class);
    job.setReducerClass(TopNReducer.class);
    job.setOutputKeyClass(Text.class);
    job.setOutputValueClass(IntWritable.class);
    FileInputFormat.addInputPath(job, new Path(otherArgs[0]));
    FileOutputFormat.setOutputPath(job, new Path(otherArgs[1]));
    System.exit(job.waitForCompletion(true) ? 0 : 1);
    }
    public static class TopNMapper extends Mapper<Object, Text, Text, IntWritable>
    { private static final IntWritable one = new IntWritable(1);
    private Text word = new Text();
    private String tokens = "[_!$#<>\\^`=\\[\\]\\\\\\*\\/\\\\\\\\\\,;,.\\-:()?!\\\"'"]
    public void map(Object key, Text value, Mapper<Object, Text, Text, IntWritable>.Context context)
    throws IOException, InterruptedException {
        String cleanLine = value.toString().toLowerCase().replaceAll(this.tokens, "&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<Text, IntWritable, Text, IntWritable>
{ public void reduce(Text key, Iterable<IntWritable> values, Reducer<Text,
IntWritable, Text, IntWritable>, Context context) throws IOException, InterruptedException
{
int sum = 0;
for (IntWritable val : values)
sum += val.get();
context.write(key, new IntWritable(sum));
}
}

```

```

TopNMapper.class
package samples.topn;
import java.io.IOException;
import java.util.StringTokenizer;
import org.apache.hadoop.io.IntWritable;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapreduce.Mapper;
public class TopNMapper extends Mapper<Object, Text, Text, IntWritable>
{ private static final IntWritable one = new IntWritable(1);
private Text word = new Text();
private String tokens = "[_!$#<>\\^=\\[\\]\\|\\*/\\\\\\\\,;,.\\-:()?!\\\"'"]
public void map(Object key, Text value, Mapper<Object, Text, Text, IntWritable>, Context context)
throws IOException, InterruptedException {
String cleanLine = value.toString().toLowerCase().replaceAll(this.tokens, &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<Text, IntWritable, Text, IntWritable>
{ private Map<Text, IntWritable> countMap = new HashMap<>();
public void reduce(Text key, Iterable<IntWritable> values, Reducer<Text, IntWritable,
Text, IntWritable>.Context context) throws IOException, InterruptedException {
int sum = 0;
for (IntWritable val : values)
sum += val.get();
this.countMap.put(new Text(key), new IntWritable(sum));
}
protected void cleanup(Reducer<Text, IntWritable, Text, IntWritable>.Context context)
throws IOException, InterruptedException {
Map<Text, IntWritable> sortedMap = MiscUtils.sortByValues(this.countMap);
int counter = 0;
for (Text key : sortedMap.keySet())
{ if (counter++ == 20)
break;
context.write(key, sortedMap.get(key));
}
}
}

```

OUTPUT

```

C:\hadoop-3.3.0\sbin>jps
11072 DataNode
20528 Jps
5620 ResourceManager
15532 NodeManager
5140 NameNode

C:\hadoop-3.3.0\sbin>hdfs dfs -mkdir /input_dir

C:\hadoop-3.3.0\sbin>hdfs dfs -ls /
Found 1 items
drwxr-xr-x   - Anusree supergroup          0 2021-05-08 19:46 /input_dir

C:\hadoop-3.3.0\sbin>hdfs dfs -copyFromLocal C:\input.txt /input_dir

C:\hadoop-3.3.0\sbin>hdfs dfs -ls /input_dir
Found 1 items
-rw-r--r--   1 Anusree supergroup        36 2021-05-08 19:48 /input_dir/input.txt

C:\hadoop-3.3.0\sbin>hdfs dfs -cat /input_dir/input.txt
hello
world
hello
hadoop
bye

```

```

C:\hadoop-3.3.0\sbin>hadoop jar C:\sort.jar samples.topn.TopN /input_dir/input.txt /output_dir
2021-05-08 19:54:54,582 INFO client.DefaultHadoopFailoverProxyProvider: Connecting to ResourceManager at /0.0.0.0:8032
2021-05-08 19:54:55,291 INFO mapreduce.JobResourceUploader: Disabling Erasure Coding for path: /tmp/hadoop-yarn/staging/Anusree/.staging/job_1620483374279_0001
2021-05-08 19:54:55,821 INFO input.FileInputFormat: Total input files to process : 1
2021-05-08 19:54:56,261 INFO mapreduce.JobSubmitter: number of splits:1
2021-05-08 19:54:56,552 INFO mapreduce.JobSubmitter: Submitting tokens for job: job_1620483374279_0001
2021-05-08 19:54:56,552 INFO mapreduce.JobSubmitter: Executing with tokens: []
2021-05-08 19:54:56,843 INFO conf.Configuration: resource-types.xml not found
2021-05-08 19:54:56,843 INFO resource.ResourceUtils: Unable to find 'resource-types.xml'.
2021-05-08 19:54:57,387 INFO impl.YarnClientImpl: Submitted application application_1620483374279_0001
2021-05-08 19:54:57,587 INFO mapreduce.Job: The url to track the job: http://LAPTOP-J6329E5D:8088/proxy/application_1620483374279_0001/
2021-05-08 19:54:57,588 INFO mapreduce.Job: Running job: job_1620483374279_0001
2021-05-08 19:55:13,792 INFO mapreduce.Job: Job job_1620483374279_0001 running in uber mode : false
2021-05-08 19:55:13,794 INFO mapreduce.Job: map 0% reduce 0%
2021-05-08 19:55:20,020 INFO mapreduce.Job: map 100% reduce 0%
2021-05-08 19:55:27,116 INFO mapreduce.Job: map 100% reduce 100%
2021-05-08 19:55:33,199 INFO mapreduce.Job: Job job_1620483374279_0001 completed successfully
2021-05-08 19:55:33,334 INFO mapreduce.Job: Counters: 54
  File System Counters
    FILE: Number of bytes read=65
    FILE: Number of bytes written=530397
    FILE: Number of read operations=0
    FILE: Number of large read operations=0
    FILE: Number of write operations=0
    HDFS: Number of bytes read=142
    HDFS: Number of bytes written=31
    HDFS: Number of read operations=8
    HDFS: Number of large read operations=0
    HDFS: Number of write operations=2
    HDFS: Number of bytes read erasure-coded=0

```

```

C:\hadoop-3.3.0\sbin>hdfs dfs -cat /output_dir/*
hello      2
hadoop     1
world      1
bye        1

C:\hadoop-3.3.0\sbin>

```