

# VISVESVARAYA TECHNOLOGICAL UNIVERSITY

“JnanaSangama”, Belgaum -590014, Karnataka.



## LAB REPORT

on

## BIG DATA ANALYTICS

*Submitted by*

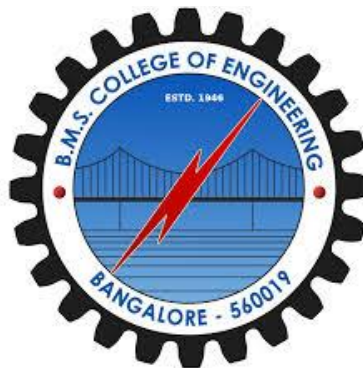
**V KENNY PHILIP**  
**(1BM21CS232)**

*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 “LAB COURSE **BIG DATA ANALYTICS**” carried out by **V KENNY PHILIP (1BM21CS232)**, 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 - (22CS6PEBDA)** work prescribed for the said degree.

**Dr. Selva Kumar S**

Assistant Professor

Department of CSE

BMSCE, Bengaluru

**Dr. Jyothi S Nayak**

Professor and Head

Department of CSE

BMSCE, Bengaluru

## Index Sheet

<b>Sl. No.</b>	<b>Experiment Title</b>	<b>Page No.</b>
<b>1</b>	<b>Perform the following DB operations using Cassandra.</b> <b>1. Create a keyspace by name Employee</b> <b>2. Create a column family by name Employee-Info with attributes Emp_Id Primary Key, Emp_Name, Designation, Date_of_Joining, Salary, Dept_Name</b> <b>3. Insert the values into the table in batch</b> <b>4. Update Employee name and Department of Emp-Id 121</b> <b>5. Sort the details of Employee records based on salary</b> <b>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.</b> <b>7. Update the altered table to add project names.</b> <b>8. Create a TTL of 15 seconds to display the values of Employees.</b>	<b>1 - 3</b>
<b>2</b>	<b>Perform the following DB operations using Cassandra.</b> <b>1. Create a keyspace by name Library</b> <b>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</b> <b>3. Insert the values into the table in batch</b> <b>4. Display the details of the table created and increase the value of the counter</b> <b>5. Write a query to show that a student with id 112 has taken a book “BDA” 2 times.</b> <b>6. Export the created column to a csv file</b> <b>7. Import a given csv dataset from local file system into Cassandra column family</b>	<b>4 - 6</b>
<b>3</b>	<b>MongoDB- CRUD Demonstration</b>	<b>7 - 9</b>

<b>4</b>	<b>Screenshot of Hadoop installed</b>	<b>10</b>
<b>5</b>	<b>Execution of HDFS Commands for interaction with Hadoop Environment. (Minimum 10 commands to be executed)</b>	<b>11 – 12</b>
<b>6</b>	<b>Implement WordCount Program on Hadoop framework</b>	<b>13 - 15</b>
<b>7</b>	<b>From the following link extract the weather data</b> <a href="https://github.com/tomwhite/hadoop-book/tree/master/input/ncdc/all">https://github.com/tomwhite/hadoop-book/tree/master/input/ncdc/all</a> <b>Create a Map Reduce program to</b> a) <b>find average temperature for each year from NCDC data set.</b> b) <b>find the mean max temperature for every month</b>	<b>16 - 23</b>
<b>8</b>	<b>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.</b>	<b>24 - 29</b>

## **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.

## **Program 1**

**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**
- 3. Insert the values into the table in batch**
- 4. Update Employee name and Department of Emp-Id 121**
- 5. Sort the details of Employee records based on 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.**
- 7. Update the altered table to add project names.**
- 8. Create a TTL of 15 seconds to display the values of Employees.**

1. Create a keyspace by name Employee

```
CREATE KEYSPACE Employee WITH replication = {'class': 'SimpleStrategy',  
      'replication_factor': 1};
```

2. Create a column family by name Employee-Info

```
CREATE TABLE Employee.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

```
BEGIN BATCH  
INSERT INTO Employee.Employee_Info (Emp_Id, Emp_Name, Designation,  
      Date_of_Joining, Salary, Dept_Name) VALUES (121, 'John Doe', 'Software Engineer',  
      '2022-01-15', 70000.00, 'IT');
```

```
INSERT INTO Employee.Employee_Info (Emp_Id, Emp_Name, Designation,
Date_of_Joining, Salary, Dept_Name) VALUES (122, 'Jane Smith', 'Data Scientist', '2021-
05-20', 80000.00, 'Data Science');
```

```
INSERT INTO Employee.Employee_Info (Emp_Id, Emp_Name, Designation,
Date_of_Joining, Salary, Dept_Name) VALUES (123, 'Alice Johnson', 'Project Manager',
'2020-07-18', 90000.00, 'Management');
```

```
APPLY BATCH;
```

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

```
UPDATE Employee.Employee_Info SET Emp_Name = 'Johnathon Doe', Dept_Name =
'Software Development' WHERE Emp_Id = 121;
```

5. Sort the details of Employee records based on salary

```
CREATE INDEX ON Employee.Employee_Info (Salary);
```

6. Alter the schema of the table Employee\_Info to add a column Projects

```
ALTER TABLE Employee.Employee_Info ADD Projects set<text>;
```

7. Update the altered table to add project names

```
UPDATE Employee.Employee_Info SET Projects = {'Project A', 'Project B'} WHERE
Emp_Id = 121;
```

```
UPDATE Employee.Employee_Info SET Projects = {'Project C'} WHERE Emp_Id = 122;
```

```
UPDATE Employee.Employee_Info SET Projects = {'Project D', 'Project E'} WHERE
Emp_Id = 123;
```

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

```
INSERT INTO Employee.Employee_Info (Emp_Id, Emp_Name, Designation,
Date_of_Joining, Salary, Dept_Name) VALUES (124, 'Bob Brown', 'Analyst', '2023-01-
10', 60000.00, 'Finance') USING TTL 15;
```

```

Connected to Test Cluster at 127.0.0.1:9042
[cqlsh 6.1.0 | Cassandra 4.1.5 | CQL spec 3.4.6 | Native protocol v5]
Use HELP for help.
cqlsh> CREATE KEYSPACE Employee WITH replication = {'class': 'SimpleStrategy', 'replication_factor': 1};
cqlsh> CREATE TABLE Employee.Employee_Info (
...     Emp_Id int PRIMARY KEY,
...     Emp_Name text,
...     Designation text,
...     Date_of_Joining date,
...     Salary decimal,
...     Dept_Name text
... );
cqlsh> BEGIN BATCH
... INSERT INTO Employee.Employee_Info (Emp_Id, Emp_Name, Designation, Date_of_Joining, Salary, Dept_Name) VALUES (121, 'John Doe', '
Software Engineer', '2022-01-15', 70000.00, 'IT');
... INSERT INTO Employee.Employee_Info (Emp_Id, Emp_Name, Designation, Date_of_Joining, Salary, Dept_Name) VALUES (122, 'Jane Smith',
'Data Scientist', '2021-05-20', 80000.00, 'Data Science');
... INSERT INTO Employee.Employee_Info (Emp_Id, Emp_Name, Designation, Date_of_Joining, Salary, Dept_Name) VALUES (123, 'Alice Johnso
n', 'Project Manager', '2020-07-18', 90000.00, 'Management');
... APPLY BATCH;
cqlsh> UPDATE Employee.Employee_Info SET Emp_Name = 'Johnathon Doe', Dept_Name =
... 'Software Development' WHERE Emp_Id = 121;
cqlsh> CREATE INDEX ON Employee.Employee_Info (Salary);
cqlsh> ALTER TABLE Employee.Employee_Info ADD Projects set<text>;
cqlsh> UPDATE Employee.Employee_Info SET Projects = {'Project A', 'Project B'} WHERE Emp_Id = 121;
cqlsh> UPDATE Employee.Employee_Info SET Projects = {'Project C'} WHERE Emp_Id = 122;
cqlsh> INSERT INTO Employee.Employee_Info (Emp_Id, Emp_Name, Designation, Date_of_Joining, Salary, Dept_Name) VALUES (124, 'Bob Brown',
'Analyst', '2023-01-10', 60000.00, 'Finance') USING TTL 15;
cqlsh> SELECT * FROM Employee_Infooooo
... SELECT * FROM Employee_Info;
cqlsh> INSERT INTO Employee.Employee_Info (Emp_Id, Emp_Name, Designation, Date_of_Joining, Salary, Dept_Name) VALUES (124, 'Bob Brown',
'Analyst', '2023-01-10', 60000.00, 'Finance') USING TTL 15;

```

## Program 2

Perform the following DB operations using Cassandra.

### 1. Create a keyspace by name Library

```
CREATE KEYSPACE Library WITH replication = { 'class' : 'SimpleStrategy',  
      'replication_factor' : 3 };
```

### 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

USE 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  
);
```

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

```
BEGIN BATCH;
```

```
INSERT INTO Library_Info (Stud_Id, Counter_value, Stud_Name, Book_Name,  
    Book_Id, Date_of_issue)  
VALUES (1, 101, 'Alice Smith', 'Introduction to Algorithms', 'B001', '2024-05-01');
```

```
INSERT INTO Library_Info (Stud_Id, Counter_value, Stud_Name, Book_Name,  
    Book_Id, Date_of_issue)  
VALUES (2, 102, 'Bob Johnson', 'Clean Code', 'B002', '2024-05-02');
```

```
INSERT INTO Library_Info (Stud_Id, Counter_value, Stud_Name, Book_Name,  
    Book_Id, Date_of_issue)
```



```
VALUES (3, 103, 'Charlie Brown', 'Design Patterns', 'B003', '2024-05-03');
```

```
INSERT INTO Library_Info (Stud_Id, Counter_value, Stud_Name, Book_Name,  
Book_Id, Date_of_issue)
```

```
VALUES (4, 104, 'Diana Prince', 'The Pragmatic Programmer', 'B004', '2024-05-04');
```

```
INSERT INTO Library_Info (Stud_Id, Counter_value, Stud_Name, Book_Name,  
Book_Id, Date_of_issue)
```

```
VALUES (5, 105, 'Ethan Hunt', 'Effective Java', 'B005', '2024-05-05');
```

```
APPLY BATCH;
```

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

```
SELECT * FROM Library_Info;
```

```
UPDATE Library_Info SET Counter_value = Counter_value + 1 WHERE Stud_Id = 111;
```

```
SELECT * FROM Library_Info;
```

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

```
SELECT Stud_Name, Book_Name, Counter_value FROM Library_Info
```

```
WHERE Stud_Id = 112 AND Book_Name = 'BDA';
```

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

```
COPY Library_Info TO '/path/to/<lib_info>.csv' WITH DELIMITER = ',' QUOTE = ''
```

```
HEADER = TRUE;
```

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

```
COPY Library_Info FROM '/path/to/<filename>.csv' WITH DELIMITER = ',' QUOTE =
```

```
'''' HEADER = TRUE;
```



### **Program 3**

#### **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.

```
> use StudentDB
```

2. Insert appropriate values

```
> db.students.insertMany([
    { Rollno: 1, Age: 20, ContactNo: "1234567890", EmailId: "student1@example.com" },
    { Rollno: 2, Age: 21, ContactNo: "1234567891", EmailId: "student2@example.com" },
    { Rollno: 10, Age: 22, ContactNo: "1234567892", EmailId: "student10@example.com" },
    { Rollno: 11, Age: 23, ContactNo: "1234567893", EmailId: "student11@example.com", Name:
"ABC" }
])
```

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

```
> db.students.updateOne(
    { Rollno: 10 },
    { $set: { EmailId: "newemail10@example.com" } }
)
```

4. Replace the student’s name from “ABC” to “FEM” of rollno 11

```
> db.students.updateOne(
    { Rollno: 11, Name: "ABC" },
    { $set: { Name: "FEM" } }
)
```

```
mongosh mongodb+srv://<credentials>@cluster0.g2lpi.mongodb.net/
Atlas atlas-vf2sb5-shard-0 [primary] test> use StudentDB
switched to db StudentDB
Atlas atlas-vf2sb5-shard-0 [primary] StudentDB> db.students.insertMany([ { Rollno: 1, Age: 20, ContactNo: "1234567890", EmailId: "student1@example.com" }, { Rollno: 2, Age: 21, ContactNo: "1234567891", EmailId: "student2@example.com" }, { Rollno: 10, Age: 22, ContactNo: "1234567892", EmailId: "student10@example.com" }, { Rollno: 11, Age: 23, ContactNo: "1234567893", EmailId: "student11@example.com", Name: "ABC" } ])
{
  acknowledged: true,
  insertedIds: {
    '0': ObjectId("665cb292bb6d9bf2f0a39a10"),
    '1': ObjectId("665cb292bb6d9bf2f0a39a11"),
    '2': ObjectId("665cb292bb6d9bf2f0a39a12"),
    '3': ObjectId("665cb292bb6d9bf2f0a39a13")
  }
}
Atlas atlas-vf2sb5-shard-0 [primary] StudentDB> db.students.updateOne({ Rollno: 10 }, { $set: { EmailId: "newemail10@example.com" } })
{
  acknowledged: true,
  insertedId: null,
  modifiedCount: 1,
  upsertedCount: 0
}
Atlas atlas-vf2sb5-shard-0 [primary] StudentDB> db.students.find({Rollno:10})
[
  {
    _id: ObjectId("665cb292bb6d9bf2f0a39a12"),
    Rollno: 10,
    Age: 22,
    ContactNo: "1234567892",
    EmailId: "newemail10@example.com"
  }
]
Atlas atlas-vf2sb5-shard-0 [primary] StudentDB> db.students.updateOne({ Rollno: 11, Name: "ABC" }, { $set: { Name: "FER" } })
{
  acknowledged: true,
  insertedId: null,
  matchedCount: 1,
  modifiedCount: 1,
  upsertedCount: 0
}
Atlas atlas-vf2sb5-shard-0 [primary] StudentDB> db.students.find({Rollno:11})
[
  {
    _id: ObjectId("665cb292bb6d9bf2f0a39a13"),
    Rollno: 11,
    Age: 23,
    ContactNo: "1234567893",
    EmailId: "student11@example.com",
    Name: "FER"
  }
]
```

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

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

```
db.Customers.insertOne({
  Cust_id: 1,
  Acc_Bal: 1000,
  Acc_Type: "A"
});
```

2. Insert at least 5 values into the table

> use CustomerDB

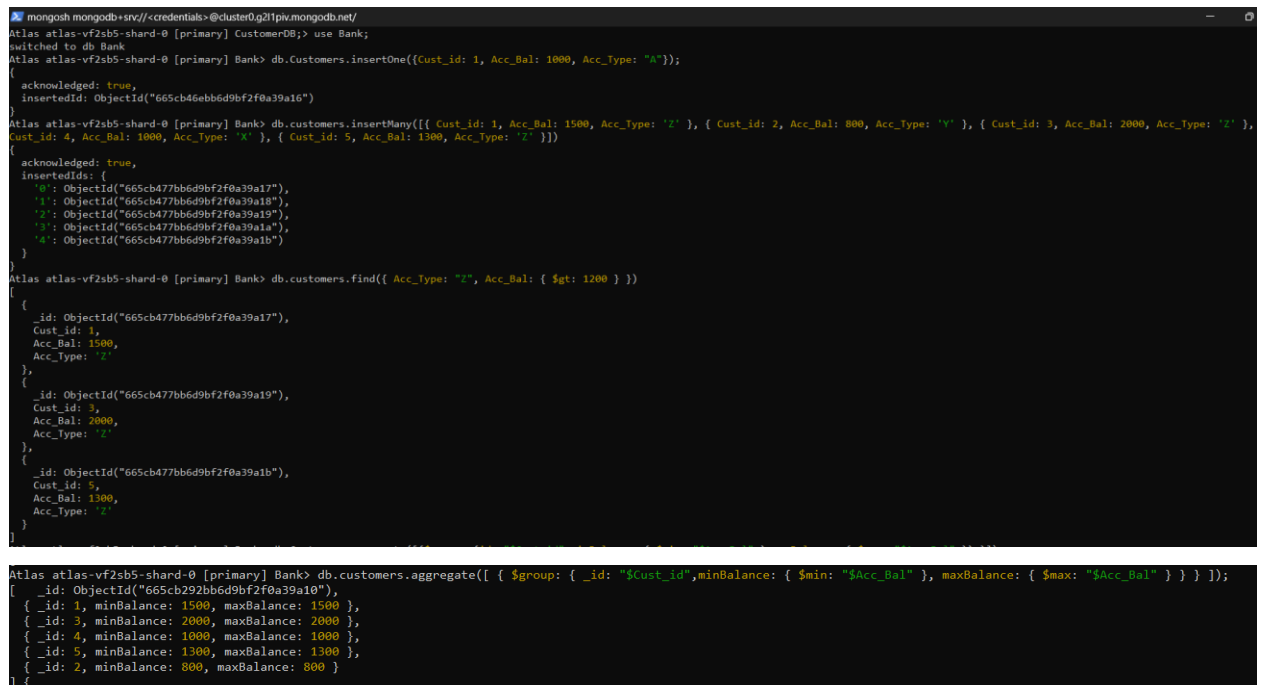
```
db.customers.insertMany([
  { Cust_id: 1, Acc_Bal: 1500, Acc_Type: 'Z' },
  { Cust_id: 2, Acc_Bal: 800, Acc_Type: 'Y' },
  { Cust_id: 3, Acc_Bal: 2000, Acc_Type: 'Z' },
  { Cust_id: 4, Acc_Bal: 1000, Acc_Type: 'X' },
  { Cust_id: 5, Acc_Bal: 1300, Acc_Type: 'Z' }
])
```

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

```
db.Customers.find({
  Acc_Type: "Z",
  Acc_Bal: { $gt: 1200 }
});
```

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

```
db.Customers.aggregate([
  {
    $group: {
      _id: "$Cust_id",
      minBalance: { $min: "$Acc_Bal" },
      maxBalance: { $max: "$Acc_Bal" }
    }
  }
]);
```



The screenshot shows a MongoDB terminal window with the following commands and results:

```
Atlas atlas-vf2sb5-shard-0 [primary] CustomerDB; use Bank;
switched to db Bank
Atlas atlas-vf2sb5-shard-0 [primary] Bank> db.Customers.insertOne({Cust_id: 1, Acc_Bal: 1500, Acc_Type: "A"});
{
  acknowledged: true,
  insertedId: ObjectId("665cb46ebb6d9bf2f0a39a16")
}
Atlas atlas-vf2sb5-shard-0 [primary] Bank> db.customers.insertMany([ { Cust_id: 1, Acc_Bal: 1500, Acc_Type: 'Z' }, { Cust_id: 2, Acc_Bal: 800, Acc_Type: 'Y' }, { Cust_id: 3, Acc_Bal: 2000, Acc_Type: 'Z' },
Cust_id: 4, Acc_Bal: 1000, Acc_Type: 'X' }, { Cust_id: 5, Acc_Bal: 1300, Acc_Type: 'Z' } ] );
{
  acknowledged: true,
  insertedIds: {
    '0': ObjectId("665cb477bb6d9bf2f0a39a17"),
    '1': ObjectId("665cb477bb6d9bf2f0a39a18"),
    '2': ObjectId("665cb477bb6d9bf2f0a39a19"),
    '3': ObjectId("665cb477bb6d9bf2f0a39a1a"),
    '4': ObjectId("665cb477bb6d9bf2f0a39a1b")
  }
}
Atlas atlas-vf2sb5-shard-0 [primary] Bank> db.customers.find({ Acc_Type: "Z", Acc_Bal: { $gt: 1200 } })
[
  {
    _id: ObjectId("665cb477bb6d9bf2f0a39a17"),
    Cust_id: 1,
    Acc_Bal: 1500,
    Acc_Type: 'Z'
  },
  {
    _id: ObjectId("665cb477bb6d9bf2f0a39a19"),
    Cust_id: 3,
    Acc_Bal: 2000,
    Acc_Type: 'Z'
  },
  {
    _id: ObjectId("665cb477bb6d9bf2f0a39a1b"),
    Cust_id: 5,
    Acc_Bal: 1300,
    Acc_Type: 'Z'
  }
]
Atlas atlas-vf2sb5-shard-0 [primary] Bank> db.customers.aggregate([ { $group: { _id: "$Cust_id", minBalance: { $min: "$Acc_Bal" }, maxBalance: { $max: "$Acc_Bal" } } } ] );
[
  {
    _id: ObjectId("665cb292bb6d9bf2f0a39a10"),
    [
      {
        _id: 1, minBalance: 1500, maxBalance: 1500
      },
      {
        _id: 3, minBalance: 2000, maxBalance: 2000
      },
      {
        _id: 4, minBalance: 1000, maxBalance: 1000
      },
      {
        _id: 5, minBalance: 1300, maxBalance: 1300
      },
      {
        _id: 2, minBalance: 800, maxBalance: 800
      }
    ]
  }
]
```

## Program 4

### Screenshot of Hadoop installed

```
Microsoft Windows [Version 10.0.22000.739]
(c) Microsoft Corporation. All rights reserved.

C:\WINDOWS\system32>start-all.cmd
This script is Deprecated. Instead use start-dfs.cmd and start-yarn.cmd
starting yarn daemons

C:\WINDOWS\system32>jps
7072 DataNode
13492 Jps
15844 ResourceManager
16196 NameNode
1388 NodeManager

C:\WINDOWS\system32>hdfs dfs -ls -R /
drwxr-xr-x  - khush supergroup          0 2022-06-27 14:09 /input
drwxr-xr-x  - khush supergroup          0 2022-06-21 09:03 /input/inputtest
-rw-r--r--  1 khush supergroup         21 2022-06-21 09:03 /input/inputtest/output.txt
-rw-r--r--  1 khush supergroup         21 2022-06-21 08:19 /input/sample.txt
-rw-r--r--  1 khush supergroup         21 2022-06-27 14:09 /input/sample2.txt
drwxr-xr-x  - khush supergroup          0 2022-06-21 13:30 /test
-rw-r--r--  1 khush supergroup         19 2022-06-21 13:30 /test/sample.txt

C:\WINDOWS\system32>hadoop version
Hadoop 3.3.3
Source code repository https://github.com/apache/hadoop.git -r d37586cbda38c338d9fe481addda5a05fb516f71
Compiled by stevel on 2022-05-09T16:36Z
Compiled with protoc 3.7.1
From source with checksum eb96dd4a797b6989ae0cdb9db6efc6
This command was run using /C:/hadoop-3.3.3/share/hadoop/common/hadoop-common-3.3.3.jar

C:\WINDOWS\system32>
```

## Program 5

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

```
hadoop@bmscecse-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 [bmscecse-HP-Elite-Tower-800-G9-Desktop-PC]
```

```
Starting resourcemanager
```

```
Starting nodemanagers
```

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

```
WARNING: Use of this script to execute dfs is deprecated.
```

```
WARNING: Attempting to execute replacement "hdfs dfs" instead.
```

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

```
mkdir: `/sadh': File exists
```

```
hadoop@bmscecse-HP-Elite-Tower-800-G9-Desktop-PC:~$ hadoop fs -ls /
```

```
Found 1 items
```

```
drwxr-xr-x - hadoop supergroup    0 2024-05-13 14:27 /sadh
```

```
hadoop@bmscecse-HP-Elite-Tower-800-G9-Desktop-PC:~$ hadoop fs -ls /sadh
```

```
hadoop@bmscecse-HP-Elite-Tower-800-G9-Desktop-PC:~$ hdfs dfs -put  
/home/hadoop/Desktop/example/Welcome.txt /sadh/WC.txt
```

```
hadoop@bmscecse-HP-Elite-Tower-800-G9-Desktop-PC:~$ hdfs dfs -cat /sadh/WC.txt
```

```
hiii
```

```
hadoop@bmscecse-HP-Elite-Tower-800-G9-Desktop-PC:~$ hdfs dfs -get /sadh/WC.txt  
/home/hadoop/Desktop/example/WWC.txt
```

```
hadoop@bmscecse-HP-Elite-Tower-800-G9-Desktop-PC:~$ hdfs dfs -get /sadh/WC.txt  
/home/hadoop/Desktop/example/WWC2.txt
```

```
hadoop@bmscecse-HP-Elite-Tower-800-G9-Desktop-PC:~$ hdfs dfs -put  
/home/hadoop/Desktop/example/Welcome.txt /sadh/WC2.txt
```

```
hadoop@bmscecse-HP-Elite-Tower-800-G9-Desktop-PC:~$ hdfs dfs -getmerge /sadh/WC.txt  
/sadh/WC2.txt /home/hadoop/Desktop/example/Merge.txt
```

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

```
hadoop@bmscecse-HP-Elite-Tower-800-G9-Desktop-PC:~$ hadoop fs -mv /sadh /WC2.txt
```

```
hadoop@bmscecse-HP-Elite-Tower-800-G9-Desktop-PC:~$ hadoop fs -ls /sadh /WC2.txt
```

```
ls: `/sadh': No such file or directory
```

```
Found 2 items
```

```
-rw-r--r-- 1 hadoop supergroup 6 2024-05-13 14:51 /WC2.txt/WC.txt
```

```
-rw-r--r-- 1 hadoop supergroup 6 2024-05-13 15:03 /WC2.txt/WC2.txt
```

```
hadoop@bmscecse-HP-Elite-Tower-800-G9-Desktop-PC:~$ hadoop fs -cp /WC2.txt/ /WC.txt
```



## **Program 6**

### **Implement WordCount Program on Hadoop framework**

Mapper Code:

```
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> {
public void map(LongWritable key, Text value, OutputCollector<Text,
IntWritable> output, Reporter rep) throws IOException
{
String line = value.toString();
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 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("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);
}
}

```

## **Program 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**

**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 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 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));
    }
}

```

```

C:\hadoop-3.3.0\sbin>hadoop jar C:\avgtemp.jar temp.AverageDriver /input_dir/temp.txt /avgtemp_outputdir
2021-05-15 14:52:50,635 INFO client.DefaultHARMFaloverProxyProvider: Connecting to ResourceManager at /0.0.0.0:8032
2021-05-15 14:52:51,005 WARN mapreduce.JobResourceUploader: Hadoop command-line option parsing not performed. Implement the Tool interface and execute your application with ToolRunner to remedy this.
2021-05-15 14:52:51,111 INFO mapreduce.JobResourceUploader: Disabling Erasure Coding for path: /tmp/hadoop-yarn/staging/Anusree/.staging/job_1621060230696_0005
2021-05-15 14:52:51,735 INFO input.FileInputFormat: Total input files to process : 1
2021-05-15 14:52:52,751 INFO mapreduce.JobSubmitter: number of splits:1
2021-05-15 14:52:53,073 INFO mapreduce.JobSubmitter: Submitting tokens for job: job_1621060230696_0005
2021-05-15 14:52:53,073 INFO mapreduce.JobSubmitter: Executing with tokens: []
2021-05-15 14:52:53,237 INFO conf.Configuration: resource-types.xml not found
2021-05-15 14:52:53,238 INFO resource.ResourceUtils: Unable to find 'resource-types.xml'.
2021-05-15 14:52:53,312 INFO impl.YarnClientImpl: Submitted application application_1621060230696_0005
2021-05-15 14:52:53,352 INFO mapreduce.Job: The url to track the job: http://LAPTOP-JG329ESD:8088/proxy/application_1621060230696_0005/
2021-05-15 14:52:53,353 INFO mapreduce.Job: Running job: job_1621060230696_0005
2021-05-15 14:53:06,640 INFO mapreduce.Job: Job job_1621060230696_0005 running in user mode : false
2021-05-15 14:53:06,643 INFO mapreduce.Job: map 0% reduce 0%
2021-05-15 14:53:12,758 INFO mapreduce.Job: map 100% reduce 0%
2021-05-15 14:53:19,060 INFO mapreduce.Job: map 100% reduce 100%
2021-05-15 14:53:25,967 INFO mapreduce.Job: Job job_1621060230696_0005 completed successfully
2021-05-15 14:53:26,096 INFO mapreduce.Job: Counters: 54
  File System Counters
    FILE: Number of bytes read=72210
    FILE: Number of bytes written=674341
    FILE: Number of read operations=0
    FILE: Number of large read operations=0
    FILE: Number of write operations=0
    HDFS: Number of bytes read=894860
    HDFS: Number of bytes written=8
    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
  Job Counters
    Launched map tasks=1
    Launched reduce tasks=1
    Data-local map tasks=1
    Total time spent by all maps in occupied slots (ms)=3782

```

```

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-000000

C:\hadoop-3.3.0\sbin>hdfs dfs -cat /avgtemp_outputdir/part-r-000000
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));

    }

}

```

```

C:\hadoop-3.3.0\sbin>hadoop jar C:\meanmax.jar meanmax.MeanMaxDriver /input_dir/temp.txt /meanmax_output
2021-05-21 20:28:05,250 INFO client.DefaultHadoopFailoverProxyProvider: Connecting to ResourceManager at /0.0.0.0:8032
2021-05-21 20:28:06,662 WARN mapreduce.JobResourceUploader: Hadoop command-line option parsing not performed. Implement the Tool interface and execute your application with ToolRunner to remedy this.
2021-05-21 20:28:06,916 INFO mapreduce.JobResourceUploader: Disabling Erasure Coding for path: /tmp/hadoop-yarn/staging/Anusree/.staging/job_1621608943095_0001
2021-05-21 20:28:08,426 INFO input.FileInputFormat: Total input files to process : 1
2021-05-21 20:28:09,107 INFO mapreduce.JobSubmitter: number of splits:1
2021-05-21 20:28:09,741 INFO mapreduce.JobSubmitter: Submitting tokens for job: job_1621608943095_0001
2021-05-21 20:28:09,741 INFO mapreduce.JobSubmitter: Executing with tokens: []
2021-05-21 20:28:10,029 INFO conf.Configuration: resource-types.xml not found
2021-05-21 20:28:10,030 INFO resource.ResourceUtils: Unable to find 'resource-types.xml'.
2021-05-21 20:28:10,676 INFO impl.YarnClientImpl: Submitted application application_1621608943095_0001
2021-05-21 20:28:11,005 INFO mapreduce.Job: The url to track the job: http://LAPTOP-3G329ESD:8088/proxy/application_1621608943095_0001/
2021-05-21 20:28:11,006 INFO mapreduce.Job: Running job: job_1621608943095_0001
2021-05-21 20:28:29,385 INFO mapreduce.Job: Job job_1621608943095_0001 running in uber mode : false
2021-05-21 20:28:29,389 INFO mapreduce.Job:  map 0% reduce 0%
2021-05-21 20:28:40,664 INFO mapreduce.Job:  map 100% reduce 0%
2021-05-21 20:28:50,832 INFO mapreduce.Job:  map 100% reduce 100%
2021-05-21 20:28:50,965 INFO mapreduce.Job: Job job_1621608943095_0001 completed successfully
2021-05-21 20:28:59,178 INFO mapreduce.Job: Counters: 54
  File System Counters
    FILE: Number of bytes read=59082
    FILE: Number of bytes written=648091
    FILE: Number of read operations=0
    FILE: Number of large read operations=0
    FILE: Number of write operations=0
    HDFS: Number of bytes read=894860
    HDFS: Number of bytes written=74
    HDFS: Number of read operations=0
    HDFS: Number of large read operations=0
    HDFS: Number of write operations=2
    HDFS: Number of bytes read erasure-coded=0
  Job Counters
    Launched map tasks=1
    Launched reduce tasks=1
    Data-local map tasks=1
    Total time spent by all maps in occupied slots (ms)=8077
    Total time spent by all reduces in occupied slots (ms)=7511
    Total time spent by all map tasks (ms)=8077
    Total time spent by all reduce tasks (ms)=7511
    Total vcore-milliseconds taken by all map tasks=8077
    Total vcore-milliseconds taken by all reduce tasks=7511
    Total megabyte-milliseconds taken by all map tasks=8270848
    Total megabyte-milliseconds taken by all reduce tasks=7691264

```

```

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>

```

## Program 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.**

### Driver-TopN.class

```
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, " ");

        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, " ");
        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));

    }

}

}

```



```

C:\hadoop-3.3.0\sbin>jps
11072 DataNode
20528 Jps
5620 ResourceManager
15532 NodeManager
6140 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.DefaultHARMPFailoverProxyProvider: 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 19:54:57,507 INFO mapreduce.Job: The url to track the job: http://LAPTOP-JG329ESD:8088/proxy/application_1620483374279_0001/
2021-05-08 19:54:57,508 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>

```