VISVESVARAYA TECHNOLOGICAL UNIVERSITY

"JnanaSangama", Belgaum -590014, Karnataka.

LAB REPORT
on

Big Data and Analytics

Submitted by

Girish Kumar S K (1BM21CS068)

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 and Analytics**" carried out by **Girish Kumar S K(1BM21CS068),** 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 and Analytics - (22CS6PEBDA)** work prescribed for the said degree.

Dr. Shyamala G
Assistant Professor
Department of CSE
BMSCE, Bengaluru

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

Index Sheet

Cassandra

1. **Perform the following DB operations using Cassandra:**
   - **i.** Create a keyspace by name Employee
   - **ii.** Create a column family by name Employee-Info with attributes Emp_Id Primary Key, Emp_Name, Designation, Date_of_Joining, Salary, Dept_Name
   - **iii.** Insert the values into the table in batch
   - **iv.** Update Employee name and Department of Emp-Id 121
   - **v.** Sort the details of Employee records based on salary
   - **vi.** Alter the schema of the table Employee_Info to add a column Projects which stores a set of Projects done by the corresponding Employee
   - **vii.** Update the altered table to add project names
   - **viii.** Create a TTL of 15 seconds to display the values of Employees

Code:

```
CREATE KEYSPACE IF NOT EXISTS Employee
WITH replication = {'class': 'SimpleStrategy', 'replication_factor': '1'};



CREATE TABLE IF NOT EXISTS Employee_Info (
    Emp_Id INT,
    Emp_Name TEXT,
    Designation TEXT,
    Date_of_Joining DATE,
    Salary DECIMAL,
    Dept_Name TEXT,
    PRIMARY KEY (Dept_Name, Salary, Emp_Id)
);




BEGIN BATCH
INSERT INTO Employee_Info (Emp_Id, Emp_Name, Designation, Date_of_Joining, Salary, Dept_Name)
VALUES (101, 'John Doe', 'Manager', '2023-01-15', 50000, 'Engineering');

INSERT INTO Employee_Info (Emp_Id, Emp_Name, Designation, Date_of_Joining, Salary, Dept_Name)
VALUES (102, 'Jane Smith', 'Developer', '2023-02-20', 40000, 'Marketing');

APPLY BATCH;




UPDATE Employee_Info
SET Emp_Name = 'New Name', Dept_Name = 'New Department'
WHERE Emp_Id = 121;
```

```
SELECT * FROM Employee_Info
WHERE Dept_Name = 'Engineering'
ORDER BY Salary DESC;


ALTER TABLE Employee_Info
ADD Projects SET<TEXT>;


UPDATE Employee_Info
SET Projects = {'Project A', 'Project B'}
WHERE Emp_Id = 101;


SELECT * FROM Employee_Info
USING TTL 15;
```

Output:

```
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, Decqlcqlcqlccqlsh>cqlscqlscqlscqlsh>c
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;S
```

2. **Perform the following DB operations using Cassandra:**
   - **i.** Create a keyspace by name Library
   - **ii.** 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
   - **iii.** Insert the values into the table in batch
   - **iv.** Display the details of the table created and increase the value of the counter
   - **v.** Write a query to show that a student with id 112 has taken a book “BDA” 2 times

Code:

```
cqlsh> use library
   ... ;
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
           ... );
```

```
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', 'B00
2', '2023-01-02');
           ... APPLY BATCH;
```

```
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
```

3. Export the created column to a csv file

Code:

```
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
```

4. Import a given csv dataset from local file system into cassandra column family

Code:

```
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
```

MongoDB

5. CRUD demonstration in MongoDB

Code:

```
// Connect to the MongoDB server and select the database
use RecordsDB;
db.createCollection("records");

// Create operation: Insert multiple documents into the collection[^1^][1]
db.records.insertMany([
  { name: "Marsh", age: "6 years", species: "Dog", ownerAddress: "380 W. Fir Ave", chipped: true },
  { name: "Kitana", age: "4 years", species: "Cat", ownerAddress: "521 E. Cortland", chipped: true },
  { name: "Buddy", age: "3 years", species: "Rabbit", ownerAddress: "742 Evergreen Terrace", chipped: false },
  { name: "Max", age: "5 years", species: "Parrot", ownerAddress: "123 Sesame Street", chipped: false },
  { name: "Bella", age: "2 years", species: "Fish", ownerAddress: "221B Baker Street", chipped: false }
]);

// Read operation: Find all documents in the collection
db.records.find({});

// Update operation: Update a document's age
db.records.updateOne({ name: "Marsh" }, { $set: { age: "7 years" } });

// Delete operation: Remove a document from the collection[^2^][2]
db.records.deleteOne({ name: "Bella" });
```

Output:

```
mycompiler_mongodb>
mycompiler_mongodb> switched to db RecordsDB
RecordsDB> { ok: 1 }
RecordsDB>
RecordsDB>
RecordsDB> ..................... {
  acknowledged: true,
  insertedIds: {
    '0': ObjectId('665dfcd8dd3f30b0334ec05e'),
    '1': ObjectId('665dfcd8dd3f30b0334ec05f'),
    '2': ObjectId('665dfcd8dd3f30b0334ec060'),
    '3': ObjectId('665dfcd8dd3f30b0334ec061'),
    '4': ObjectId('665dfcd8dd3f30b0334ec062')
  }
}
RecordsDB>
RecordsDB>
RecordsDB> [
  {
    _id: ObjectId('665dfcd8dd3f30b0334ec05e'),
    name: 'Marsh',
    age: '6 years',
    species: 'Dog',
    ownerAddress: '380 W. Fir Ave',
```

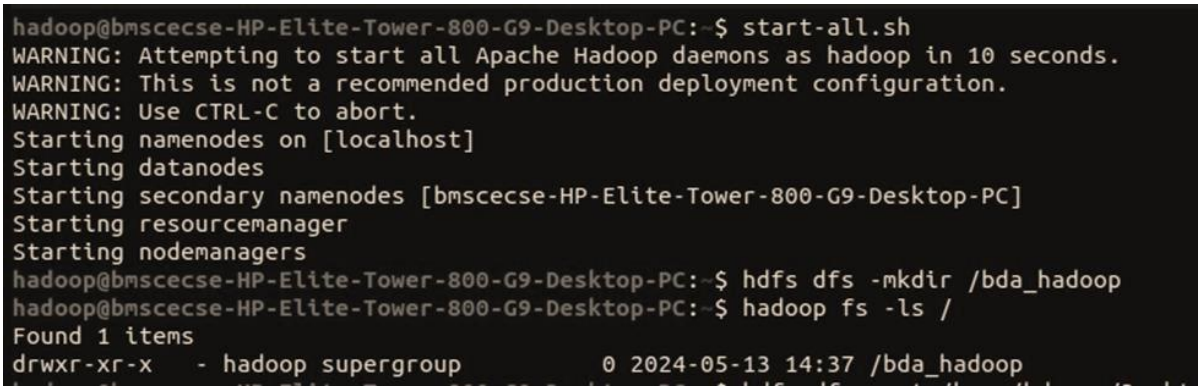
```
    chipped: true
  },
  {
    _id: ObjectId('665dfcd8dd3f30b0334ec05f'),
    name: 'Kitana',
    age: '4 years',
    species: 'Cat',
    ownerAddress: '521 E. Cortland',
    chipped: true
  },
  {
    _id: ObjectId('665dfcd8dd3f30b0334ec060'),
    name: 'Buddy',
    age: '3 years',
    species: 'Rabbit',
    ownerAddress: '742 Evergreen Terrace',
    chipped: false
  },
  {
    _id: ObjectId('665dfcd8dd3f30b0334ec061'),
    name: 'Max',
    age: '5 years',
    species: 'Parrot',
    ownerAddress: '123 Sesame Street',
    chipped: false
  },
  {
    _id: ObjectId('665dfcd8dd3f30b0334ec062'),
    name: 'Bella',
    age: '2 years',
    species: 'Fish',
    ownerAddress: '221B Baker Street',
    chipped: false
  }
]
RecordsDB>
RecordsDB>
RecordsDB> {
  acknowledged: true,
  insertedId: null,
  matchedCount: 1,
  modifiedCount: 1,
  upsertedCount: 0
}
RecordsDB>
RecordsDB>
RecordsDB> { acknowledged: true, deletedCount: 1 }
RecordsDB>

[Execution complete with exit code 0]
```

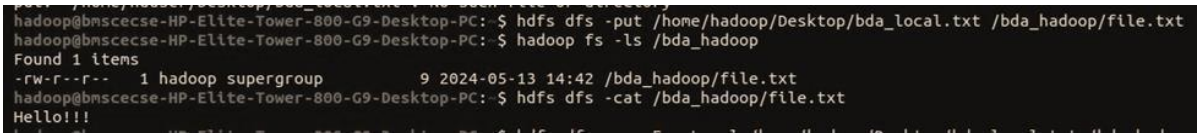
Hadoop

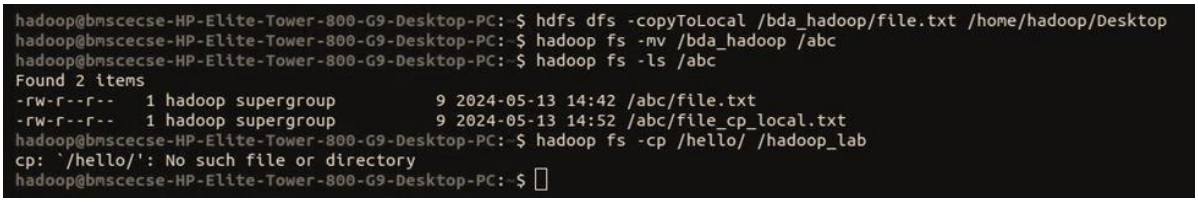
6. Execution of HDFS Commands for interaction with Hadoop Environment

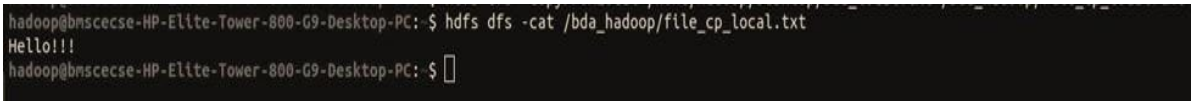
Code:

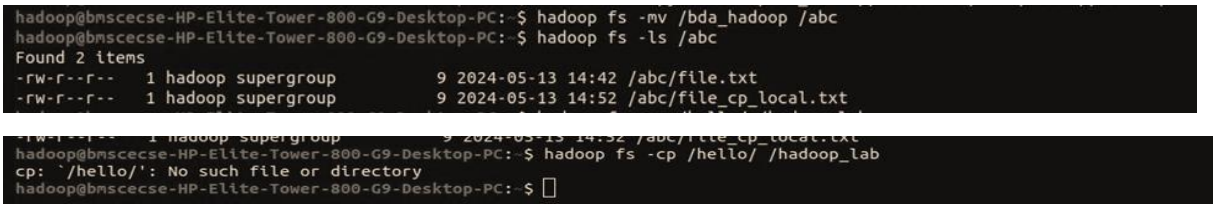
Using mkdir, ls, put, copyfromlocal, get, copytolocal cat, mv, cp











7. Implement WordCount program on Hadoop framework

Code:

```
// 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();
```

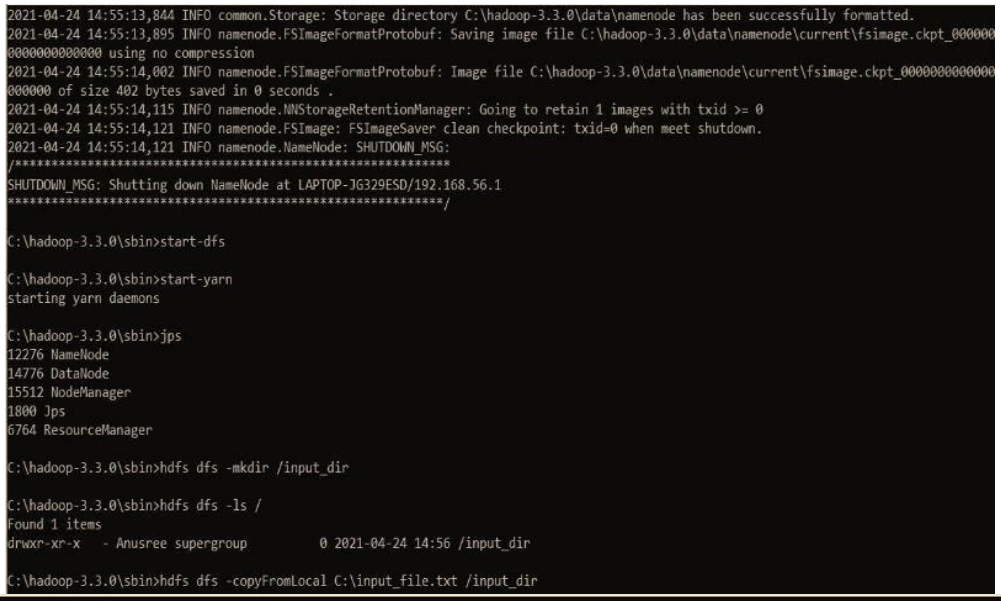
```
            // Splitting the line on spaces
            for (String word : line.split(" "))
            {
                if (word.length() > 0)
                {
                    output.collect(new Text(word), new
IntWritable(1));
                }
            }
        }
}

// 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
```

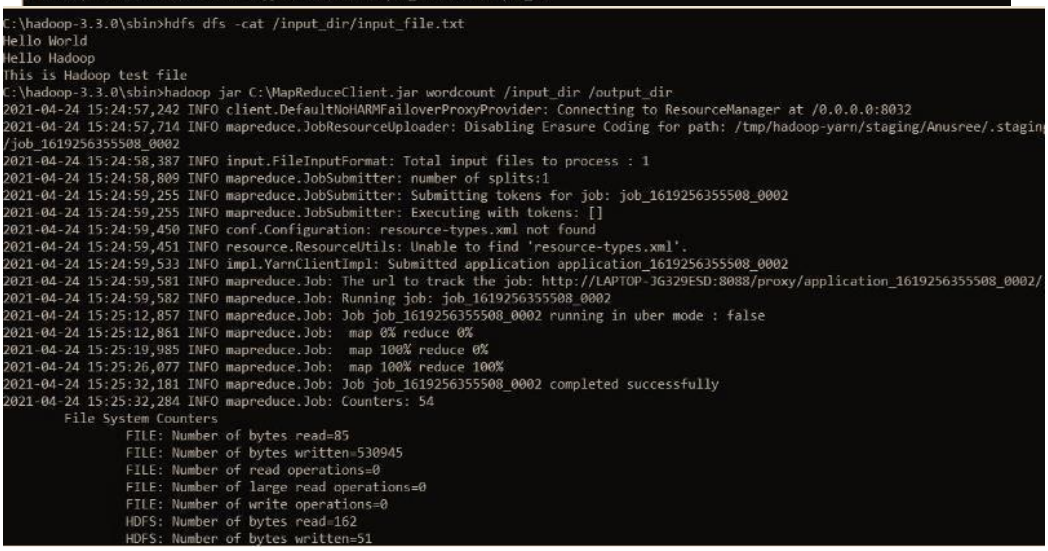
```
    public void reduce(Text key, Iterator<IntWritable> value,
                       OutputCollector<Text, IntWritable> output,
                                  Reporter rep) throws
IOException
    {

        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));
    }
}

// Importing libraries
import java.io.IOException;
import org.apache.hadoop.conf.Configured;
import org.apache.hadoop.fs.flath;
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("flease give valid inputs");
               return -1;
          }

          JobConf conf = new JobConf(WCDriver.class);
          FileInputFormat.setInputflaths(conf, new flath(args[0]));
          FileOutputFormat.setOutputflath(conf, new flath(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:





```
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>
```

8. **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:**
   i. **find average temperature for each year from NCDC data set**
   ii. **find the mean max temperature for every month**

Code:

```
// AverageDriver
package temp;

import org.apache.hadoop.fs.flath;
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 ("flease Enter the input and output parameters");
          System.exit (-1);

}

Job job = new Job ();
       job.setJarByClass (AverageDriver.class);
       job.setJobName ("Max temperature");
       FileInputFormat.addInputflath (job, new flath (args[0]));
       FileOutputFormat.setOutputflath (job, new flath (args[1]));

job.setMapperClass (AverageMapper.class);
       job.setReducerClass (AverageReducer.class);
       job.setOutputKeyClass (Text.class);
       job.setOutputValueClass (IntWritable.class);
       System.exit (job.waitForCompletion (true) o 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));
}
}
```

Output:

```
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.DefaultNoHARMFailoverProxyProvider: 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 uber 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 14:53:19,860 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-00000

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

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

9. **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**

Code:

```
// 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.flath;

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 = "[_#$ <>\\^=\\[\\]\\*/\\\\,;,.\\-
:()o!\"']";
     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.hadoopi o.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 Stri ng tokens = "[_#$ <>\\^=\\[\\]\\*/\\\\,;,.\\-
:()o!\"']";

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));}
                    }
                    }
```

Output:

```
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.DefaultNoHARMFailoverProxyProvider: 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
```