

VISVESVARAYA TECHNOLOGICAL UNIVERSITY

“JnanaSangama”, Belgaum -590014, Karnataka.



LAB REPORT
on

BIG DATA ANALYTICS **(20CS6PEBDA)**

Submitted by

Siri Chandan Sai K (1BM19CS212)

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
May-2022 to July-2022

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 **Siri Chandan Sai K(1BM19CS212)**, 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 2022. The Lab report has been approved as it satisfies the academic requirements in respect of a **BIG DATA ANALYTICS - (20CS6PEBDA)** work prescribed for the said degree.

Dr. Pallavi G B
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	MongoDB CRUD Demonstration	4-6
2	Cassandra Employee Database	7-8
3	Cassandra Library Database	9-10
4	Hadoop installation	11
5	HDFS Commands	12
6	Create a Map Reduce program to a) find average temperature for each year from NCDC data set. b) find the mean max temperature for every month	13-19
7	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.	20-25
8	Create a Map Reduce program to demonstrating join operation	26-37
9	Program to print word count on scala shell and print "Hello world" on scala IDE	38-39
10	Using RDD and FlatMap count how many times each word appears in a file and write out a list of words whose count is strictly greater than 4 using Spark	40

Course Outcome

CO1	Apply the concept of NoSQL, Hadoop or Spark for a given task
CO2	Analyze the Big Data and obtain insight using data analytics mechanisms.
CO3	Design and implement Big data applications by applying NoSQL, Hadoop or Spark

1. MongoDB- CRUD Demonstration

CRUD (CREATE, READ, UPDATE, DELETE) OPERATIONS

```

> use mydb;
switched to db mydb
> db;
mydb
> show dbs;
admin    0.000GB
config   0.000GB
local    0.000GB
> db.createCollection("student");
{ "ok" : 1 }
> show dbs;
admin    0.000GB
config   0.000GB
local    0.000GB
mydb     0.000GB
> show collections;
student
> db.student.insert({_id:1,name:"Rohit",grade:7,Hobbies:"InternetSurfing"});
WriteResult({ "nInserted" : 1 })
> db.student.find({});
{ "_id" : 1, "name" : "Rohit", "grade" : 7, "Hobbies" : "InternetSurfing" }
> db.student.update({_id:1,name:"Rohit",grade:2,Hobbies:"InternetSurfing"},{upsert:true});
WriteResult({ "nMatched" : 0, "nUpserted" : 0, "nModified" : 0 })
> db.student.find({});
{ "_id" : 1, "name" : "Rohit", "grade" : 7, "Hobbies" : "InternetSurfing" }
> db.student.update({_id:1,name:"Rohit",Hobbies:"InternetSurfing"},{grade:2},{upsert:true});
WriteResult({ "nMatched" : 1, "nUpserted" : 0, "nModified" : 1 })
> db.student.find({});
{ "id" : 1, "grade" : 2 }

```

```

> db.student.update({_id:2,name:"Rahul",grade:12,Hobbies:"Surfing"},{upsert:true});
WriteResult({ "nMatched" : 0, "nUpserted" : 0, "nModified" : 0 })
> db.student.find({});
{ "_id" : 2, "name" : "Rahul", "grade" : 10, "Hobbies" : "Surfing" }
{ "_id" : 5, "Hobbies" : "Surfing", "grade" : 10, "name" : "Somu" }
> db.student.find({},{name:1,grade:1,_id:0});
{ "name" : "Rahul", "grade" : 10 }
{ "grade" : 10, "name" : "Somu" }
> db.student.find({name:{ $eq:"Somu" }}).pretty();
{ "_id" : 5, "Hobbies" : "Surfing", "grade" : 10, "name" : "Somu" }
> db.student.find({name:{$in:['Ram','Somu']}}).pretty();
{ "_id" : 5, "Hobbies" : "Surfing", "grade" : 10, "name" : "Somu" }
> db.student.find({name:/^R/}).pretty();
{ "_id" : 2, "name" : "Rahul", "grade" : 10, "Hobbies" : "Surfing" }
> db.student.find({name:/m/}).pretty();
{ "_id" : 5, "Hobbies" : "Surfing", "grade" : 10, "name" : "Somu" }
> db.student.find().sort({name:1}).pretty();
{ "_id" : 2, "name" : "Rahul", "grade" : 10, "Hobbies" : "Surfing" }
{ "_id" : 5, "Hobbies" : "Surfing", "grade" : 10, "name" : "Somu" }
> db.student.find().sort({name:-1}).pretty();
{ "_id" : 5, "Hobbies" : "Surfing", "grade" : 10, "name" : "Somu" }
{ "id" : 2, "name" : "Rahul", "grade" : 10, "Hobbies" : "Surfing" }

```

```
C:\Program Files\MongoDB\Server\5.0\bin>mongoimport --db mydb --collection tests --type csv --headerline --file C:\Users\DELL\Downloads\csv\testdemo.csv
2022-06-05T21:40:11.501+0530   connected to: mongodb://localhost/
2022-06-05T21:40:11.538+0530   4 document(s) imported successfully. 0 document(s) failed to import.

C:\Program Files\MongoDB\Server\5.0\bin>mongoexport --db mydb --collection tests --csv --fieldFile C:\Users\DELL\Downloads\csv\testdemo.csv --out C:\Users\DELL\Downloads\csv\d.csv
2022-06-06T18:30:03.775+0530   csv flag is deprecated; please use --type=csv instead
2022-06-06T18:30:04.462+0530   connected to: mongodb://localhost/
2022-06-06T18:30:04.610+0530   exported 4 records
```

```
> db.Faculty.aggregate({$match : {department : "CSE"} },
... {$group : {_id : "$designation",avgsal : {$avg : "$salary"} }
... },
... {$match : {avgsal : { $gt : 65000} }}});
{ "_id" : "Professor", "avgsal" : 78333.33333333333 }
>
```

2. Perform the following DB operations using Cassandra.

1. Create a key space by name Employee

```
nakhilesh@nakhilesh-VirtualBox:~/Desktop$ cqlsh
Connected to Test Cluster at 127.0.0.1:9042
[cqlsh 6.0.0 | Cassandra 4.0.3 | CQL spec 3.4.5 | Native protocol v5]
Use HELP for help.
cqlsh> CREATE KEYSPACE emp WITH replication = {'class': 'SimpleStrategy', 'replication_factor' : 3};
```

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> use emp;
cqlsh:emp> CREATE TABLE emp_info(
... emp_id int primary key,
... emp_name text,
... designation text,
... date_of_joining timestamp,
... salary int,
... dept_name text);
```

3. Insert the values into the table in batch

```
cqlsh:emp> BEGIN BATCH
... INSERT INTO emp_info(emp_id,emp_name,designation,date_of_joining,salary,dept_name) VALUES (1,'Somu','ABC','2022-06-08',25000,'BCD');
... APPLY BATCH;
cqlsh:emp> BEGIN BATCH
... INSERT INTO emp_info(emp_id,emp_name,designation,date_of_joining,salary,dept_name) VALUES (2,'Shekhar','AAA','2022-04-08',35000,'BBB');
... INSERT INTO emp_info(emp_id,emp_name,designation,date_of_joining,salary,dept_name) VALUES (3,'Akshay','CCC','2021-02-08',20000,'DDD');
... APPLY BATCH;
cqlsh:emp> SELECT * from emp_info;
```

emp_id	date_of_joining	dept_name	designation	emp_name
1	2022-06-07 18:30:00.000000+0000	BCD	ABC	Somu
2	2022-04-07 18:30:00.000000+0000	BBB	AAA	Shekhar
3	2021-02-07 18:30:00.000000+0000	DDD	CCC	Akshay

4. Update Employee name and Department of Emp-Id 2

```
(3 rows)
cqlsh:emp> update emp_info set emp_name='Harsha',dept_name='CSE' where emp_id=3;
cqlsh:emp> select * from emp_info;
```

emp_id	date_of_joining	dept_name	designation	emp_name
1	2022-06-07 18:30:00.000000+0000	BCD	ABC	Somu
2	2022-04-07 18:30:00.000000+0000	BBB	AAA	Shekhar
3	2021-02-07 18:30:00.000000+0000	CSE	CCC	Harsha

```
(3 rows)
```


5. Sort the details of Employee records based on salary

```
cqlsh:emp> create table e1_info(e_id int,e_name text,e_designation text,DOJ timestamp,salary int,e_dept_name text,primary key(e_id,salary));
cqlsh:emp> BEGIN BATCH
... INSERT INTO e1_info(e_id,e_name,e_designation,DOJ,salary,e_dept_name
) VALUES (1,'Sai','Manager','2021-04-08',55000,'EEE');
... INSERT INTO e1_info(e_id,e_name,e_designation,DOJ,salary,e_dept_name
) VALUES (2,'Raghav','GM','2021-03-08',15000,'DEF');
... APPLY BATCH;
cqlsh:emp> select * from e1_info;
```

e_id	salary	doj	e_dept_name	e_designation
1	55000	2021-04-07 18:30:00.000000+0000	EEE	Manager
2	15000	2021-03-07 18:30:00.000000+0000	DEF	GM

```
(2 rows)
cqlsh:emp> paging off;
cqlsh:emp> select * from e1_info where e_id in (1,2) order by salary;
```

e_id	salary	doj	e_dept_name	e_designation
2	15000	2021-03-07 18:30:00.000000+0000	DEF	GM
1	55000	2021-04-07 18:30:00.000000+0000	EEE	Manager

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:emp> alter table emp_info add projects set<text>;
```

7. Update the altered table to add project names

```
cqlsh:emp> alter table emp_info add projects set<text>;
cqlsh:emp> update emp_info set projects=projects+{'p1','p2'} where emp_id=1;
cqlsh:emp> select * from emp_info;
```

emp_id	date_of_joining	dept_name	designation	emp_name
1	2022-06-07 18:30:00.000000+0000	BCD	ABC	Sonu
	{'p1', 'p2'}			
2	2022-04-07 18:30:00.000000+0000	BBB	AAA	Shekhar
	null			
3	2021-02-07 18:30:00.000000+0000	CSE	CCC	Harsha
	null			

```
(3 rows)
```

3. Perform the following DB operations using Cassandra.

1. Create a key space by name Library

```
nakhilesh@nakhilesh-VirtualBox:~/Desktop$ cqlsh
Connected to Test Cluster at 127.0.0.1:9042
[cqlsh 6.0.0 | Cassandra 4.0.3 | CQL spec 3.4.5 | Native protocol v5]
Use HELP for help.
cqlsh> CREATE KEYSPACE library WITH replication ={'class' : 'SimpleStrategy', 'r
eplication_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

```
cqlsh:library> CREATE TABLE library_info(
... stud_id int,
... counter_value counter,
... stud_name text,
... book_name text,
... book_id int,
... date_of_issue timestamp,
... primary key(stud_id,stud_name,book_name,book_id,date of issue));
```

3. Insert the values into the table in batch

```
cqlsh:library> update library_info set counter_value=counter_value+1 where stud
_id=1 and stud_name='AKHIL' and book_name='BDA' and date_of_issue='2022-08-09'
and book_id=11;
cqlsh:library> update library_info set counter_value=counter_value+1 where stud
_id=1 and stud_name='BHEEM' and book_name='BBB' and date_of_issue='2022-04-09'
and book_id=12;
cqlsh:library> update library_info set counter_value=counter_value+1 where stud
_id=3 and stud_name='RAM' and book_name='CCC' and date_of_issue='2022-04-06' an
d book_id=13;
cqlsh:library> update library_info set counter_value=counter_value+1 where stud
_id=2 and stud_name='BHEEM' and book_name='BBB' and date_of_issue='2022-04-09'
and book_id=12;
cqlsh:library> select * from library_info;
```

stud_id	stud_name	book_name	book_id	date_of_issue	counter_value
1	AKHIL	BDA	11	2022-08-08 18:30:00.000000+0000	1
1	BHEEM	BBB	12	2022-04-08 18:30:00.000000+0000	1
2	BHEEM	BBB	12	2022-04-08 18:30:00.000000+0000	1
3	RAM	CCC	13	2022-04-05 18:30:00.000000+0000	1

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

```
cqlsh:library> update library_info set counter_value=counter_value+2 where stud_id=2 and stud_name='BHEEM' and book_name='BBB' and date_of_issue='2022-04-09' and book_id=12;
cqlsh:library> select * from library_info;
```

stud_id	stud_name	book_name	book_id	date_of_issue	counter_value
1	AKHIL	BDA	11	2022-08-08 18:30:00.000000+0000	1
1	BHEEM	BBB	12	2022-04-08 18:30:00.000000+0000	1
2	BHEEM	BBB	12	2022-04-08 18:30:00.000000+0000	3
3	RAM	CCC	13	2022-04-05 18:30:00.000000+0000	1

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

```
cqlsh:library> select * from library_info where stud_id=2;
```

stud_id	stud_name	book_name	book_id	date_of_issue	counter_value
2	BHEEM	BBB	12	2022-04-08 18:30:00.000000+0000	3

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,counter_value) TO 'e:\library_info.csv';
Using 1 child processes

Starting copy of library.library_info with columns [stud_id, stud_name, book_name, book_id, date_of_issue, counter_value].
Processed: 4 rows; Rate: 44 rows/s; Avg. rate: 14 rows/s
4 rows exported to 1 files in 0.309 seconds.
```

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

```
cqlsh:library> CREATE TABLE lib_info(
... stud_id int,
... counter_value counter,
... stud_name text,
... book_name text,
... date_of_issue timestamp,
... book_id int,
... primary key(stud_id,stud_name,book_name,date_of_issue,book_id));
cqlsh:library> COPY lib_info(stud_id,stud_name,book_name,book_id,date_of_issue,counter_value) TO 'e:\library_info.csv';
Using 1 child processes

Starting copy of library.lib_info with columns [stud_id, stud_name, book_name, book_id, date_of_issue, counter_value].
Processed: 0 rows; Rate: 0 rows/s; Avg. rate: 0 rows/s
0 rows exported to 1 files in 0.188 seconds.
```

4.Hadoop Installation

```
C:\hadoop-3.3.3\sbin>jps
12372 SparkSubmit
19108 NameNode
19268 ResourceManager
2596 DataNode
2740 Eclipse
5476
11164 NodeManager
14044 Jps
```

5. Execution of HDFS Commands for interaction with Hadoop Environment.

```
C:\hadoop-3.3.3\sbin>hdfs dfs -mkdir /dir
```

```
C:\hadoop-3.3.3\sbin>hdfs dfs -ls /
```

```
Found 8 items
```

```
drwxr-xr-x - DELL supergroup      0 2022-06-22 10:29 /aaa
drwxr-xr-x - DELL supergroup      0 2022-06-19 19:27 /abc
drwxr-xr-x - DELL supergroup      0 2022-07-12 22:17 /dir
drwxr-xr-x - DELL supergroup      0 2022-07-10 11:19 /input
drwxr-xr-x - DELL supergroup      0 2022-07-12 16:43 /input_dir
drwxr-xr-x - DELL supergroup      0 2022-07-12 22:17 /inputdir
drwxr-xr-x - DELL supergroup      0 2022-07-10 11:25 /out
drwx----- - DELL supergroup      0 2022-07-10 11:24 /tmp
```

```
C:\hadoop-3.3.3\sbin>
```

```
C:\hadoop-3.3.3\sbin>hdfs dfs -put C:\Users\DELL\Downloads\num.txt /dir
```

```
C:\hadoop-3.3.3\sbin>hdfs dfs -ls /dir
```

```
Found 1 items
```

```
-rw-r--r-- 1 DELL supergroup      31 2022-07-12 22:21 /dir/num.txt
```

```
C:\hadoop-3.3.3\sbin>
```

```
C:\hadoop-3.3.3\sbin>hdfs dfs -cat /dir/num.txt
```

```
1
5
7
2
3
10
11
25
6
5
```

```
C:\hadoop-3.3.3\sbin>
```

```
C:\hadoop-3.3.3\sbin>hdfs dfs -copyToLocal /dir/num.txt /C:/Users/DELL/Desktop/num
```

```
C:\hadoop-3.3.3\sbin>ls /C:/Users/DELL/Desktop/num
```

6. For the given file, Create a Map Reduce program to

a) Find the average temperature for each year from the NCDC data set.

Average Driver Class

```
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);
    }
}

job.setOutputKeyClass(Text.class);
job.setOutputValueClass(IntWritable.class);
System.exit(job.waitForCompletion(true) ? 0 : 1);
}
```

Average Mapper Class

```
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));
    }
}
```

Average Reducer class

```
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>hdfs dfs -cat /avgtemp_outputdir/part-r-00000
1901 46

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

A terminal window with a black background and white text. The first line shows a command to cat a file from HDFS. The second line shows the output of the command, which is a single line of text: "1901 46". The third line shows the prompt for the next command.

b) find the mean max temperature for every month

MeanMax driver 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);
    }
}
```

MeanMax Mapper 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));
    }
}
```

MeanMax Reducer 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>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>
```

7. For a given Text file, create a Map Reduce program to sort the content in an alphabetic order listing only top 'n' maximum occurrence of words.

TopN Driver 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);
        }
    }
}

```

TopN Combiner 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));
    }
}
```

TopN Mapper 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);
        }
    }
}
```


TopN Reducer 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>hdfs dfs -cat /output_dir/*
```

```
hello 2
```

```
hadoop 1
```

```
world 1
```

```
bye 1
```

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

8. Create a Map Reduce program to demonstrating join operation.

```
// JoinDriver.java

import org.apache.hadoop.conf.Configured;
import org.apache.hadoop.fs.Path;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapred.*;
import org.apache.hadoop.mapred.libMultipleInputs;
import org.apache.hadoop.util.*;

public class JoinDriver extends Configured implements Tool {

    public static class KeyPartitioner implements Partitioner<TextPair, Text> {

        @Override
        public void configure(JobConf job) {}

        @Override
        public int getPartition(TextPair key, Text value, int numPartitions) {
            return (key.getFirst().hashCode() & Integer.MAX_VALUE) %
                numPartitions;
        }
    }

    @Override
    public int run(String[] args) throws Exception {
```

```
if (args.length != 3) {
    System.out.println("Usage: <Department Emp Strength input>

    <Department Name input> <output>");
    return -1;
}

JobConf conf = new JobConf(getConf(), getClass());

conf.setJobName("Join 'Department Emp Strength input' with 'Department Name
input'");

Path AInputPath = new Path(args[0]);
Path BInputPath = new Path(args[1]);
Path outputPath = new Path(args[2]);

MultipleInputs.addInputPath(conf, AInputPath, TextInputFormat.class,

Posts.class);

MultipleInputs.addInputPath(conf, BInputPath, TextInputFormat.class,

User.class);

FileOutputFormat.setOutputPath(conf, outputPath);
```

```
conf.setPartitionerClass(KeyPartitioner.class);
```

```
conf.setOutputValueGroupingComparator(TextPair.FirstComparator.class);
```

```
conf.setMapOutputKeyClass(TextPair.class);
```

```
conf.setReducerClass(JoinReducer.class);
```

```
conf.setOutputKeyClass(Text.class);
```

```
JobClient.runJob(conf);
```

```
return 0;
```

```
}
```

```
public static void main(String[] args) throws Exception {
```

```
int exitCode = ToolRunner.run(new JoinDriver(), args);
```

```
System.exit(exitCode);
```

```
}
```

```
}
```

```
// JoinReducer.java
```

```
import java.io.IOException;
```

```
import java.util.Iterator;
```

```
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapred.*;

public class JoinReducer extends MapReduceBase implements Reducer<TextPair,
Text, Text,
Text> {

    @Override
    public void reduce (TextPair key, Iterator<Text> values, OutputCollector<Text,
Text>
output, Reporter reporter)

        throws IOException
    {

        Text nodeId = new Text(values.next());
        while (values.hasNext()) {

            Text node = values.next();
            Text outValue = new Text(nodeId.toString() + "\t\t" + node.toString());
            output.collect(key.getFirst(), outValue);
        }
    }
}
```

```
// User.java

import java.io.IOException;
import java.util.Iterator;
import org.apache.hadoop.conf.Configuration;
import org.apache.hadoop.fs.FSDataInputStream;
import org.apache.hadoop.fs.FSDataOutputStream;
import org.apache.hadoop.fs.FileSystem;
import org.apache.hadoop.fs.Path;
import org.apache.hadoop.io.LongWritable;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapred.*;

import org.apache.hadoop.io.IntWritable;

public class User extends MapReduceBase implements Mapper<LongWritable,
Text, TextPair,
Text> {

    @Override
    public void map(LongWritable key, Text value, OutputCollector<TextPair, Text>
output,
    Reporter reporter)

    throws IOException

    {
```

```
String valueString = value.toString();
```

```
String[] SingleNodeData = valueString.split("\t");
```

```
output.collect(new TextPair(SingleNodeData[0], "1"), new
```

```
Text(SingleNodeData[1]));
```

```
}
```

```
}
```

```
//Posts.java
```

```
import java.io.IOException;
```

```
import org.apache.hadoop.io.*;
```

```
import org.apache.hadoop.mapred.*;
```

```
public class Posts extends MapReduceBase implements Mapper<LongWritable,  
Text, TextPair,
```

```
Text> {
```

```
@Override
```

```
public void map(LongWritable key, Text value, OutputCollector<TextPair, Text>  
output,
```

```
Reporter reporter)
```

```
throws IOException
```

```
{
```



```
String valueString = value.toString();  
String[] SingleNodeData = valueString.split("\t");  
output.collect(new TextPair(SingleNodeData[3], "0"), new
```

```
Text(SingleNodeData[9]));  
}  
}
```

```
// TextPair.java
```

```
import java.io.*;
```

```
import org.apache.hadoop.io.*;
```

```
public class TextPair implements WritableComparable<TextPair> {
```

```
    private Text first;
```

```
    private Text second;
```

```
    public TextPair() {
```

```
        set(new Text(), new Text());
```

```
    }
```

```
    public TextPair(String first, String second) {
```

```
        set(new Text(first), new Text(second));
```

```
    }
```

```
public TextPair(Text first, Text second) {  
    set(first, second);  
}
```

```
public void set(Text first, Text second) {  
    this.first = first;  
    this.second = second;  
}
```

```
public Text getFirst() {  
    return first;  
}
```

```
public Text getSecond() {  
    return second;  
}
```

@Override

```
public void write(DataOutput out) throws IOException {  
    first.write(out);  
    second.write(out);  
}
```

@Override

```
public void readFields(DataInput in) throws IOException {  
    first.readFields(in);  
}
```

```
second.readFields(in);  
}
```

```
@Override  
public int hashCode() {  
    return first.hashCode() * 163 + second.hashCode();  
}
```

```
@Override  
public boolean equals(Object o) {  
    if (o instanceof TextPair) {  
        TextPair tp = (TextPair) o;  
        return first.equals(tp.first) && second.equals(tp.second);  
    }  
    return false;  
}
```

```
@Override  
public String toString() {  
    return first + "\t" + second;  
}
```

```
@Override  
public int compareTo(TextPair tp) {  
    int cmp = first.compareTo(tp.first);  
    if (cmp != 0) {
```

```

return cmp;
}
return second.compareTo(tp.second);
}
// ^^ TextPair

// vv TextPairComparator
public static class Comparator extends WritableComparator {

private static final Text.Comparator TEXT_COMPARATOR = new
Text.Comparator();

public Comparator() {
super(TextPair.class);
}

@Override
public int compare(byte[] b1, int s1, int l1,
byte[] b2, int s2, int l2) {

try {
int firstL1 = WritableUtils.decodeVIntSize(b1[s1]) + readVInt(b1, s1);
int firstL2 = WritableUtils.decodeVIntSize(b2[s2]) + readVInt(b2, s2);
int cmp = TEXT_COMPARATOR.compare(b1, s1, firstL1, b2, s2, firstL2);
if (cmp != 0) {
return cmp;

```

```

    }
    return TEXT_COMPARATOR.compare(b1, s1 + firstL1, l1 - firstL1,

    b2, s2 + firstL2, l2 - firstL2);
    } catch (IOException e) {
    throw new IllegalArgumentException(e);
    }
    }
    }

    static {
    WritableComparator.define(TextPair.class, new Comparator());
    }

    public static class FirstComparator extends WritableComparator {

    private static final Text.Comparator TEXT_COMPARATOR = new
    Text.Comparator();

    public FirstComparator() {
    super(TextPair.class);
    }

    @Override
    public int compare(byte[] b1, int s1, int l1,
    byte[] b2, int s2, int l2) {

```

```

try {
    int firstL1 = WritableUtils.decodeVIntSize(b1[s1]) + readVInt(b1, s1);
    int firstL2 = WritableUtils.decodeVIntSize(b2[s2]) + readVInt(b2, s2);
    return TEXT_COMPARATOR.compare(b1, s1, firstL1, b2, s2, firstL2);
} catch (IOException e) {
    throw new IllegalArgumentException(e);
}
}

```

@Override

```

public int compare(WritableComparable a, WritableComparable b) {
    if (a instanceof TextPair && b instanceof TextPair) {
        return ((TextPair) a).first.compareTo(((TextPair) b).first);
    }
    return super.compare(a, b);
}
}

```



```

C:\hadoop-3.3.0\sbin>hdfs dfs -cat /join8_output/part-00000
"100005361"      "2"             "36134"
"100018705"      "2"             "76"
"100022094"      "0"             "6354"

```

9. Program to print word count on scala shell and print “Hello world” on scala IDE

```
scala> println("Hello World!");  
Hello World!
```

```
val data=sc.textFile("C:\Users\DELL\Downloads\new.txt ")  
data.collect;  
val splitdata = data.flatMap(line => line.split(" "));  
splitdata.collect;  
val mapdata = splitdata.map(word => (word,1));  
mapdata.collect;  
val reducedata = mapdata.reduceByKey(_+_);  
reducedata.collect;
```

```
scala> val data=sc.textFile("C:/Users/DELL/Downloads/new.txt")  
data: org.apache.spark.rdd.RDD[String] = C:/Users/DELL/Downloads/new.txt MapPartitionsRDD[18] at textFile at <console>:24  
  
scala> data.collect;  
res14: Array[String] = Array(hi, how, hello, how, bye, how, hello, hi, how, how)  
  
scala> val splitdata = data.flatMap(line => line.split(" "));  
splitdata: org.apache.spark.rdd.RDD[String] = MapPartitionsRDD[19] at flatMap at <console>:25  
  
scala> splitdata.collect;  
res15: Array[String] = Array(hi, how, hello, how, bye, how, hello, hi, how, how)  
  
scala> val mapdata = splitdata.map(word => (word,1));  
mapdata: org.apache.spark.rdd.RDD[(String, Int)] = MapPartitionsRDD[20] at map at <console>:25  
  
scala> mapdata.collect;  
res16: Array[(String, Int)] = Array((hi,1), (how,1), (hello,1), (how,1), (bye,1), (how,1), (hello,1), (hi,1), (how,1), (how,1))  
  
scala> val reducedata = mapdata.reduceByKey(_+_);  
reducedata: org.apache.spark.rdd.RDD[(String, Int)] = ShuffledRDD[21] at reduceByKey at <console>:25  
  
scala> reducedata.collect;  
res17: Array[(String, Int)] = Array((how,5), (hello,2), (bye,1), (hi,2))  
  
scala> _
```

```
package helloWorld

object hello {
  def main (args: Array[String]) {
    println("Hello World")
  }
}
```

Problems Tasks Console

<terminated> hello\$ [Scala Application] C:\Program Files\Java\jre1.8.0_261\bin\javaw.exe (12-Jul-2022, 11:11:36 PM)

Hello World

10. Using RDD and Flat Map count how many times each word appears in a file and write out a list of words whose count is strictly greater than 4 using Spark

```
val textFile = sc.textFile("C:\\Users\\DELL\\Downloads\\new.txt ")
val counts = textFile.flatMap(line => line.split(" ")).map(word => (word,
1)).reduceByKey(_ + _)
import scala.collection.immutable.ListMap
val sorted=ListMap(counts.collect.sortWith(_._2 > _._2))
println(sorted)
for((k,v)<-sorted)
{
  if(v>4)
  {
    print(k+",")
    print(v)
    println()
  }
}
```

```
scala> val textFile = sc.textFile("C:/Users/DELL/Downloads/new.txt")
textFile: org.apache.spark.rdd.RDD[String] = C:/Users/DELL/Downloads/new.txt MapPartitionsRDD[23] at textFile at <console>:24

scala> val counts = textFile.flatMap(line => line.split(" ")).map(word => (word, 1)).reduceByKey(_ + _)
counts: org.apache.spark.rdd.RDD[(String, Int)] = ShuffledRDD[26] at reduceByKey at <console>:25

scala> import scala.collection.immutable.ListMap
import scala.collection.immutable.ListMap

scala> val sorted=ListMap(counts.collect.sortWith(_._2 > _._2):_*)// sort in descending order based on values
sorted: scala.collection.immutable.ListMap[String,Int] = ListMap(how -> 5, hello -> 2, hi -> 2, bye -> 1)

scala> println(sorted)
ListMap(how -> 5, hello -> 2, hi -> 2, bye -> 1)

scala> for((k,v)<-sorted)
| {
|   if(v>4)
|   {
|     print(k+",")
|     print(v)
|     println()
|   }
| }
how,5
```