

# **VISVESVARAYA TECHNOLOGICAL UNIVERSITY**

“JnanaSangama”, Belgaum -590014, Karnataka.



**LAB REPORT**  
**on**

## **BIG DATA ANALYTICS** **(20CS6PEBDA)**

*Submitted by*

**MOHAMMED IBRAHIM RAHIL S (1BM19CS090)**

*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 **Mohammed Ibrahim Rahil S (1BM19CS090)**, 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 **Big data Analytics - (20CS6PEBDA)** work prescribed for the said degree.

**Prof. Pallavi G B**  
Assistant Professor  
Department of CSE  
BMSCE, Bengaluru

## Index Sheet

Sl. No.	Experiment Title	Page No.
1	MongoDB CRUD Operations	4
2	MongoDB Operations	7
3	Cassandra Lab 1	10
4	Cassandra Lab 2	11
5	Screenshot of Hadoop	12
6	Execution of HDFS	13
7	MapReduce to find mean/max temperature	15
8	List top 10 max occurrence	19
9	Demonstrate Join Operation	21
10	Print word count on scala	24
11	Using RDD and FlatMap	25

## 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 Operations

## I. CREATE DATABASE IN MONGODB

**>use rahilDB**

switched to db rahilDB

## II. CRUD (CREATE, READ, UPDATE, DELETE) OPERATIONS

**>db.createCollection("Student");**

{ "ok" : 1 }

**>db.Student.insert({\_id:1,name:"Rahil",grade:9});**

WriteResult({ "nInserted" : 1 })

**>db.Student.update({\_id:6,name:"qwert"},{\$set:{grade:4}},{upsert:true});**

WriteResult({ "nMatched" : 0, "nUpserted" : 1, "nModified" : 0, "\_id" : 6 })

**>db.Student.find();**

{ "\_id" : 1, "name" : "Rahil", "grade" : 9 }

{ "\_id" : 2, "name" : "Abc", "grade" : 10 }

{ "\_id" : 3, "name" : "Mno", "grade" : 5 }

{ "\_id" : 4, "name" : "Pqr", "grade" : 8 }

**> show collections;**

Student

## III. Save Method

**> db.Student.save({name:"zzz",\_id:10,grade:8});**

WriteResult({ "nMatched" : 0, "nUpserted" : 1, "nModified" : 0, "\_id" : 10 })

## IV. COUNT

**> db.Student.count();**

6

**> db.Student.count({grade:9});**

1

## V FIND

**> db.Student.find({grade:{\$lt:5}},{name:1,grade:1,\_id:0});**

```
{ "grade" : 2, "name" : "qwert" }
```

**> db.Student.find({name:{\$in:["Rahil","Abc","Mno"]}}, {name:1,grade:1,\_id:0});**

```
{ "name" : "Rahil", "grade" : 9 }
```

```
{ "name" : "Abc", "grade" : 10 }
```

```
{ "name" : "Mno", "grade" : 5 }
```

**> db.Student.find({name:/^S/},{name:1,grade:1,\_id:0});**

```
{ "name" : "Rahil", "grade" : 9 }
```

**> db.Student.find({name:/.b/},{name:1,grade:1,\_id:0});**

```
{ "name" : "Abc", "grade" : 10 }
```

**> db.Student.find().sort({name:1});**

```
{ "_id" : 2, "name" : "Abc", "grade" : 10 }
```

```
{ "_id" : 3, "name" : "Mno", "grade" : 5 }
```

```
{ "_id" : 4, "name" : "Pqr", "grade" : 8 }
```

```
{ "_id" : 1, "name" : "Saffan", "grade" : 9 }
```

```
{ "_id" : 7, "name" : "kkk", "grade" : 6 }
```

```
{ "_id" : 6, "grade" : 2, "name" : "qwert" }
```

**> db.Student.find().sort({name:1,grade:-1});**

```
{ "_id" : 2, "name" : "Abc", "grade" : 10 }
```

```
{ "_id" : 3, "name" : "Mno", "grade" : 5 }
```

```
{ "_id" : 4, "name" : "Pqr", "grade" : 8 }
```

```
{ "_id" : 1, "name" : "Saffan", "grade" : 9 }
```

```
{ "_id" : 7, "name" : "kkk", "grade" : 6 }
```

```
{ "_id" : 6, "grade" : 2, "name" : "qwert" }
```

**> db.Student.find({grade:8}).limit(3);**

```
{ "_id" : 4, "name" : "Pqr", "grade" : 8 }
```

```
{ "_id" : 10, "name" : "zzz", "grade" : 8 }
```

```
> db.Student.find().skip(2);
```

```
{ "_id" : 3, "name" : "Mno", "grade" : 5 }
```

```
{ "_id" : 4, "name" : "Pqr", "grade" : 8 }
```

```
{ "_id" : 6, "grade" : 2, "name" : "qwert" }
```

```
{ "_id" : 7, "name" : "kkk", "grade" : 6 }
```

```
{ "_id" : 10, "name" : "zzz", "grade" : 8 }
```

## VI. AGGREGATE FUNCTIONS

```
> db.faculty.aggregate ( {$match:{department:"mech"}}, {$group : { _id :  
"$designation", AverageSal :{$avg:"$salary"} } },  
{$match:{AverageSal:{$gt:50000}}});
```

```
{ "_id" : " associate prof", "AverageSal" : 85000 }
```

```
{ "_id" : "assistant prof", "AverageSal" : 70000 }
```

## VII. ARRAYS

```
> db.food.insert({_id:1,fruits:['apple','mango']});
```

```
WriteResult({ "nInserted" : 1 })
```

```
> db.food.find({fruits:['pineapple','mango','orange']});
```

```
{ "_id" : 3, "fruits" : [ "pineapple", "mango", "orange" ] }
```

```
> db.food.find({fruits:{$all:['pineapple']}});
```

```
{ "_id" : 2, "fruits" : [ "pineapple", "mango", "grapes" ] }
```

```
{ "_id" : 3, "fruits" : [ "pineapple", "mango", "orange" ] }
```

```
> db.food.update({_id:2},{ $set:{'fruits.1':'apple'}});
```

```
WriteResult({ "nMatched" : 1, "nUpserted" : 0, "nModified" : 1 })
```

```
> db.food.update({_id:2},{ $push:{price:{grapes:80,mango:200,cherry:100}}} );
```

```
WriteResult({ "nMatched" : 1, "nUpserted" : 0, "nModified" : 1 })
```

## 2. MongoDB Operations

### 1) Faculty DB

i) Create a database for Faculty and Create a Faculty Collection(Faculty\_id, Name, Designation ,Department, Age, Salary, Specialization(Set)).

>use Faculty

> db.createCollection("faculty")

ii) Insert required documents to the collection.

> db.faculty.insert({\_id:1,name:"abc",designation:"assistant prof",department:"mech",age:31,salary:90000,specialization:['python','mysql','autocad']});

iii) First Filter on “Dept\_Name:MECH” and then group it on “Designation” and compute the Average Salary for that Designation and filter those documents where the “Avg\_Sal” is greater than 650000.

> db.faculty.aggregate ( {\$match:{department:"mech"}}, {\$group : {\_id : "\$designation", AverageSal :{\$avg:"\$salary"} } }, {\$match:{AverageSal:{\$gt:50000}}});

```
{ "_id" : " associate prof", "AverageSal" : 85000 }
```

```
{ "_id" : "assistant prof", "AverageSal" : 70000 }
```

2) Consider a table “Product” with the following columns:

Product \_id

ProductName

ManufacturingDate

Price

Quantity

Write MongoDB queries for the following:

> use Products switched to db Products

> db.createCollection("product");

```
{ "ok" : 1 }
```

>

```
db.product.insert({pid:1,pname:"keyboard",mdate:2001,price:1800,quantity:2})
;
```

```
WriteResult({ "nInserted" : 1 })
```

**i)To display only the product name from all the documents of the product collection.**

```
> db.product.find({}, {pname:1, _id:0});
```

```
{ "pname" : "keyboard" }
```

```
{ "pname" : "mouse" }
```

```
{ "pname" : "motherboard" }
```

**ii)To display only the Product ID, ExpiryDate as well as the quantity from the document of the product collection where the \_id column is 1.**

```
> db.product.find({pid:1},
```

```
{pid:1, _id:0, mdate:1, quantity:1});
```

```
{ "pid" : 1, "mdate" : 2001, "quantity" : 2 }
```

**iii) To find those documents where the price is not set to 45000.**

```
> db.product.find({price:{$ne:45000}}, {pname:1, _id:0});
```

```
{ "pname" : "keyboard" }
```

```
{ "pname" : "mouse" }
```

```
{ "pname" : "motherboard" }
```

**iv)To find those documents from the Product collection where the quantity is set to 30 and the product name is set to 'LEDTV'.**

```
> db.product.find({$and:[{quantity:{$eq:30}}, {pname:{$eq:"LED
TV"}}]}, {pname:1, _id:0})8
```

```
{ "pname" : "LED TV" }
```

**v)To find documents from the Product collection where the Product name ends in 'r'.**

```
> db.product.find({pname:/d$/}, {pname:1, quantity:1, _id:0})
```



```
{ "pname" : "keyboard", "quantity" : 2 }
```

```
{ "pname" : "motherboard", "quantity" : 150 }
```

**3) Create a mongodb collection Hospital. Demonstrate the following by choosing fields of your choice.**

**> use Hospital switched to db Hospital**

**> db.createCollection("hospital");**

```
{ "ok" : 1 }
```

**> db.hospital.insert({\_id:1,name:"xyz",diseases:["diabetes","high bp","fever"]});**

```
WriteResult({ "nInserted" : 1 })
```

**1. Insert three documents**

**> db.hospital.updateMany({},{\$pull:{diseases:"fever"}});**

```
{ "acknowledged" : true, "matchedCount" : 3, "modifiedCount" : 2 }
```

**2. Use Arrays(Use Pull and Pop operation)**

**> db.hospital.updateOne({\_id:1},{\$pop:{diseases:-1}});**

```
{ "acknowledged" : true, "matchedCount" : 1, "modifiedCount" : 1 }
```

**3. Use Index**

**> db.hospital.find({"diseases.2":"nausea"});**

```
{ "_id" : 3, "name" : "mno", "diseases" : [ "covid", "sarscov", "nausea" ] }
```

**4. Use Cursors**

**> db.hospital.find({}).count();**

```
3
```

**> db.hospital.find({}).limit(2);**

```
{ "_id" : 1, "name" : "xyz", "diseases" : [ "high bp" ] } { "_id" : 2, "name" : "abc", "diseases" : [ "typhoid", "cholera" ] }
```

**> db.hospital.find({}).size();**

```
3
```

**5. Updation**

**> db.hospital.update({\_id:3},{\$set:{'diseases.1':'sarscov'}});**

```
WriteResult({ "nMatched" : 1, "nUpserted" : 0, "nModified" : 1 })
```

### 3. Cassandra Lab 1

#### 1. Create a key space by name Employee

```
cqlsh:saf> create keyspace Employee with  
replication={'class':'SimpleStrategy','replication_factor':1}; cqlsh:saf> use  
Employee ;
```

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

```
cqlsh:employee> create table empInfo( emp_id int PRIMARY KEY, emp_name  
text,desig text,dpj timestamp,salary int,dept_name text );
```

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

```
cqlsh:employee> insert into  
empInfo(emp_id,emp_name,desig,dpj,salary,dept_name) values( 1, 'saffan',  
'sde', '2022-05-05', 200000, 'cse' );
```

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

```
cqlsh:employee> update empInfo set emp_name='zzz',dept_name='ie'where  
emp_id=2;
```

#### 5. Sort the details of Employee records based on salary

```
.cqlsh:employee> select * from emp_Info where emp_id in (1,2,3) order by  
salary;
```

#### 6. Alter the schema of the table Employee\_Info to add a column Projects;which stores a set of Projects done by the corresponding Employee.

```
cqlsh:employee> alter table empInfo add project set
```

#### 7. Update the altered table to add project names.

```
cqlsh:employee> update empInfo set project={'reactJs','MI'} where emp_id=1;
```

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

```
cqlsh:employee> insert into  
empInfo(emp_id,emp_name,desig,dpj,salary,dept_name) values( 5, 'wxy', 'sde',  
'2022-02-05', 250000, 'cse' ) using ttl 30; cqlsh:employee> select ttl(emp_name)  
from empInfo;
```

## 4. Cassandra Lab 2

### 1 Create a key space by name Library

```
CREATE keyspace library1 with replication={ 'class':'SimpleStrategy',  
'replication_factor':1 };
```

### 2. Create a column family by name Library-Info with attributes Stud\_Id Primary Key,Counter\_value of type Counter,Stud\_Name, Book-Name, Book-Id, Date\_of\_issue

```
CREATE TABLE lib.libinfo1 ( s_id int, sname text, book text, bid int, doi  
timestamp, counter_val counter, PRIMARY KEY (s_id, sname, book, bid, doi) );
```

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

```
update libinfo set counter_val=counter_val+1 where s_id=1 and sname='saf'  
and book='harry potter1' and bid=1 and doi='2022-05-05';
```

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

```
cqlsh:lib> update libinfo set counter_val=counter_val+1 where s_id=1 and  
sname='saf' and book='harry potter1'; cqlsh:lib> select * from libinfo;
```

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

```
cqlsh:lib> select counter_val from libinfo where s_id=1 and sname='saf' and  
book='harry potter1';
```

```
counter_val
```

```
2
```

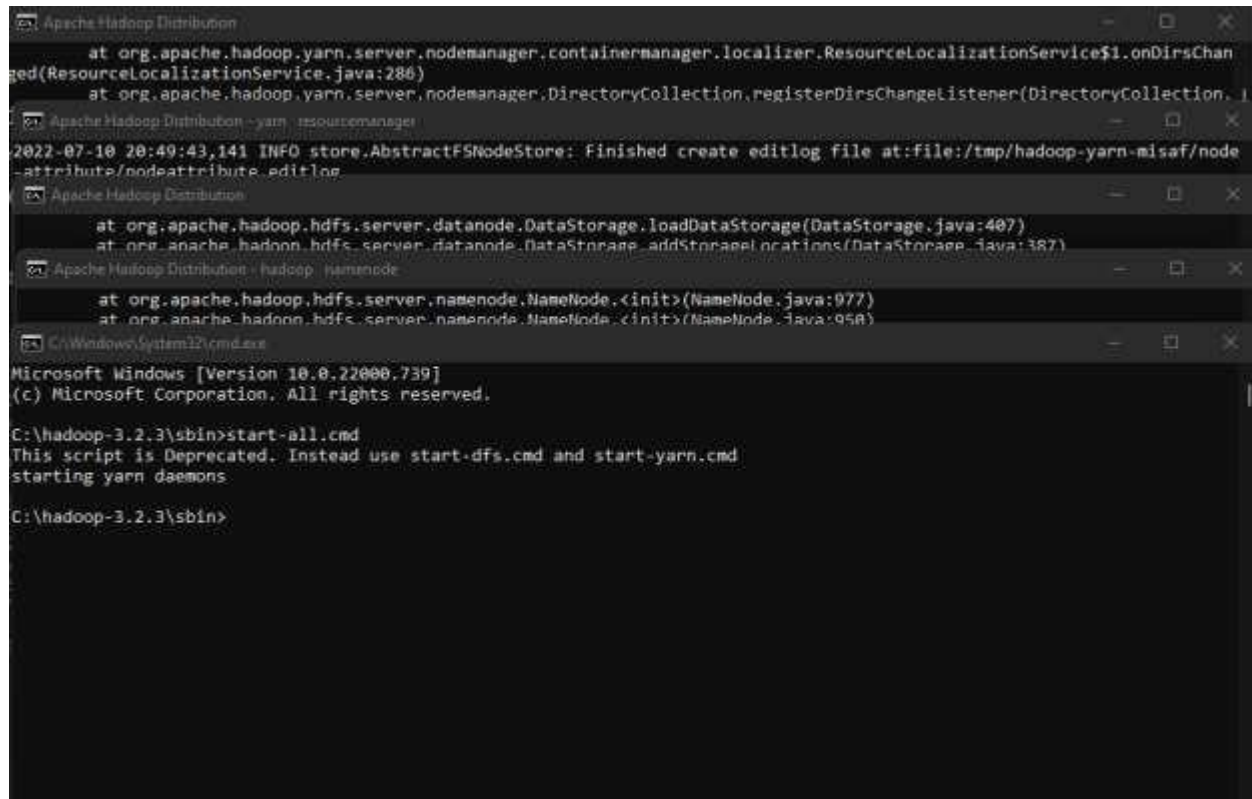
### 6. Export the created column to a csv file

```
COPY libinfo(s_id,sname,book,bid,doi,counter_val) TO 'data1.csv' WITH HEADER  
= TRUE;
```

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

```
COPY libinfo(s_id,sname,book,bid,doi) FROM 'libdata.csv' WITH HEADER =  
TRUE;
```

## 5. Screenshot of Hadoop Installation



The screenshot displays a series of overlapping windows from a Hadoop installation. The top window, titled 'Apache Hadoop Distribution', shows Java stack traces for 'ResourceLocalizationService' and 'DirectoryCollection'. Below it, a window titled 'Apache Hadoop Distribution - yarn - resourcemanager' shows a log message: '2022-07-10 20:49:43,141 INFO store.AbstractFSNodeStore: Finished create editlog file at:file:/tmp/hadoop-yarn-misaf/node-attribute/nodeattribute.editlog'. Another window titled 'Apache Hadoop Distribution' shows stack traces for 'DataStorage.loadDataStorage' and 'DataStorage.addStorageLocations'. Below that, a window titled 'Apache Hadoop Distribution - hadoop - namenode' shows stack traces for 'NameNode.<init>'. The bottom window is a Windows command prompt titled 'C:\Windows\System32\cmd.exe', showing the execution of 'C:\hadoop-3.2.3\sbin>start-all.cmd'. The prompt displays a deprecation warning: 'This script is Deprecated. Instead use start-dfs.cmd and start-yarn.cmd starting yarn daemons' and then shows the prompt 'C:\hadoop-3.2.3\sbin>'.

```
at org.apache.hadoop.yarn.server.nodemanager.containermanager.localizer.ResourceLocalizationService$1.onDirChan
ged(ResourceLocalizationService.java:286)
at org.apache.hadoop.yarn.server.nodemanager.DirectoryCollection.registerDirChangeListener(DirectoryCollection, 1
Apache Hadoop Distribution - yarn - resourcemanager
2022-07-10 20:49:43,141 INFO store.AbstractFSNodeStore: Finished create editlog file at:file:/tmp/hadoop-yarn-misaf/node
-attribute/nodeattribute.editlog
Apache Hadoop Distribution
at org.apache.hadoop.hdfs.server.datanode.DataStorage.loadDataStorage(DataStorage.java:407)
at org.apache.hadoop.hdfs.server.datanode.DataStorage.addStorageLocations(DataStorage.java:387)
Apache Hadoop Distribution - hadoop - namenode
at org.apache.hadoop.hdfs.server.namenode.NameNode.<init>(NameNode.java:977)
at org.apache.hadoop.hdfs.server.namenode.NameNode.<init>(NameNode.java:958)
C:\Windows\System32\cmd.exe
Microsoft Windows [Version 10.0.22000.739]
(c) Microsoft Corporation. All rights reserved.

C:\hadoop-3.2.3\sbin>start-all.cmd
This script is Deprecated. Instead use start-dfs.cmd and start-yarn.cmd
starting yarn daemons
C:\hadoop-3.2.3\sbin>
```

## 6. HDFS Commands

```
>C:\hadoop_new\sbin>hdfs dfs -mkdir /lab1
```

```
>C:\hadoop_new\sbin>hdfs dfs -ls /
```

Found 2 items

drwxr-xr-x - Admin supergroup 0 2022-06-19 14:47 /lab1

drwxr-xr-x - Admin supergroup 0 2022-06-19 14:46 /sample

```
>C:\hadoop_new\sbin>hdfs dfs -copyFromLocal E:\Desktop\sample.txt \lab1
```

```
>C:\hadoop_new\sbin>hdfs dfs -ls
```

/lab1Found 1 items

-rw-r--r-- 1 Admin supergroup 19 2022-06-19 14:51 /lab1/sample.txt

```
>C:\hadoop_new\sbin>hdfs dfs -cat
```

\lab1\sample.txtsample text for lab

//create a folder sample in desktop

```
>C:\hadoop_new\sbin>hdfs dfs -get \lab1\sample.txt E:\Desktop\sample
```

```
>C:\hadoop_new\sbin>hdfs dfs -put E:\Desktop\samplefolder \lab1
```

```
>C:\hadoop_new\sbin>hdfs dfs -ls
```

/lab1Found 2 items

-rw-r--r-- 1 Admin supergroup 19 2022-06-19 14:51 /lab1/sample.txt

drwxr-xr-x - Admin supergroup 0 2022-06-19 14:58

/lab1/samplefolder

```
C:\hadoop_new\sbin>hdfs dfs -mv /sample /lab1
```

**C:\hadoop\_new\sbin>hdfs dfs -ls**

**/lab1Found 3 items**

drwxr-xr-x - Admin supergroup 0 2022-06-19 14:50 /lab1/sample

-rw-r--r-- 1 Admin supergroup 19 2022-06-19 14:51 /lab1/sample.txt

drwxr-xr-x - Admin supergroup 0 2022-06-19 14:58

/lab1/samplefolder**C:\hadoop\_new\sbin>hdfs dfs -cp**

**/lab1/sample /**

**C:\hadoop\_new\sbin>hdfs dfs -ls /**

Found 2 items

drwxr-xr-x - Admin supergroup 0 2022-06-19 15:00 /lab1

drwxr-xr-x - Admin supergroup 0 2022-06-19 15:01 /sample

**C:\hadoop\_new\sbin>hdfs dfs -rm**

/lab1/sample.txtDeleted /lab1/sample.txt

**C:\hadoop\_new\sbin>hdfs dfs -ls**

**/lab1Found 2 items**

drwxr-xr-x - Admin supergroup 0 2022-06-19 14:50 /lab1/sample

drwxr-xr-x - Admin supergroup 0 2022-06-19 14:58 /lab1/samplefolder

**C:\hadoop\_new\sbin>hdfs dfs -copyFromLocal E:\Desktop\sample.txt**

**\lab1 C:\hadoop\_new\sbin>hdfs dfs -copyToLocal \lab1\sample.txt**

**E:\Desktop\sample1.txt**

## 7. Mean/Max temperature of weather data

### Driver class:

```
package temperatureMax;

import org.apache.hadoop.io.*;
import org.apache.hadoop.fs.*;
import org.apache.hadoop.mapreduce.*;
import org.apache.hadoop.mapreduce.lib.input.FileInputFormat;
import org.apache.hadoop.mapreduce.lib.output.FileOutputFormat;

public class TempDriver
{
    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(TempDriver.class);
        job.setJobName("Max temperature");
        FileInputFormat.addInputPath(job,new Path(args[0]));
        FileOutputFormat.setOutputPath(job,new Path (args[1]));
        job.setMapperClass(TempMapper.class);
        job.setReducerClass(TempReducer.class);
        job.setOutputKeyClass(Text.class);
        job.setOutputValueClass(IntWritable.class);
        System.exit(job.waitForCompletion(true)?0:1);
    }
}
```

```
}
```

### Mapper Class

```
package temperatureMax;
```

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

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

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

```
public class TempMapper extends Mapper <LongWritable, Text, Text, IntWritable>
```

```
{
```

```
    public static final int MISSING = 9999;
```

```
    public void map(LongWritable key, Text value, Context context) throws IOException,
        InterruptedException
```

```
    {
```

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

```
        String month = line.substring(19,21);
```

```
        int temperature;
```

```
        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 != MISSING && quality.matches("[01459]"))
```

```
            context.write(new Text(month),new IntWritable(temperature));
```

```
    }
```

```
}
```



### Reducer class

```
package temperatureMax;

import org.apache.hadoop.io.IntWritable;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapreduce.*;
import java.io.IOException;

public class TempReducer extends Reducer <Text, IntWritable,Text, IntWritable>
{
    public void reduce(Text key, Iterable<IntWritable> values, Context context) throws
    IOException,InterruptedException
    {
        int max_temp = 0;
        for (IntWritable value : values)
        {
            if(max_temp<value.get()) {
                max_temp = value.get();
            }
        }
        context.write(key, new IntWritable(max_temp));
    }
}
```

### Output:

```
hduser@lab-VirtualBox:/home/lab$ hadoop dfs -cat /tempmax/part-r-00000
DEPRECATED: Use of this script to execute hdfs command is deprecated.
Instead use the hdfs command for it.

WARNING: An illegal reflective access operation has occurred
WARNING: Illegal reflective access by org.apache.hadoop.security.authentication
.util.KerberosUtil (file:/usr/local/hadoop/share/hadoop/common/lib/hadoop-auth-
2.6.0.jar) to method sun.security.krb5.Config.getInstance()
WARNING: Please consider reporting this to the maintainers of org.apache.hadoop
.security.authentication.util.KerberosUtil
WARNING: Use --illegal-access=warn to enable warnings of further illegal reflec
tive access operations
WARNING: All illegal access operations will be denied in a future release
21/05/10 16:08:48 WARN util.NativeCodeLoader: Unable to load native-hadoop libr
ary for your platform... using builtin-java classes where applicable
03      111
05      22
```

```
hduser@lab-VirtualBox:/home/lab$ hadoop dfs -ls /tempmax
DEPRECATED: Use of this script to execute hdfs command is deprecated.
Instead use the hdfs command for it.

WARNING: An illegal reflective access operation has occurred
WARNING: Illegal reflective access by org.apache.hadoop.security.authentication
.util.KerberosUtil (file:/usr/local/hadoop/share/hadoop/common/lib/hadoop-auth-
2.6.0.jar) to method sun.security.krb5.Config.getInstance()
WARNING: Please consider reporting this to the maintainers of org.apache.hadoop
.security.authentication.util.KerberosUtil
WARNING: Use --illegal-access=warn to enable warnings of further illegal reflec
tive access operations
WARNING: All illegal access operations will be denied in a future release
21/05/10 16:08:23 WARN util.NativeCodeLoader: Unable to load native-hadoop libr
ary for your platform... using builtin-java classes where applicable
```

## 8. Word Occurences

```
hadoop@Lab-VirtualBox: /home/lab/hadoop-2.6.0/share/hadoop/mapreduce$ hadoop jar
hadoop-mapreduce-examples-2.6.0.jar wordcount /input /firstExampleOut
21/04/26 15:19:29 WARN util.NativeCodeLoader: Unable to load native-hadoop libr
ary for your platform... using builtin-java classes where applicable
21/04/26 15:19:31 INFO Configuration.deprecation: session.id is deprecated. Ins
tead, use dfs.metrics.session-id
21/04/26 15:19:31 INFO jvm.JvmMetrics: Initializing JVM Metrics with processNam
e=JobTracker, sessionId=
21/04/26 15:19:32 INFO input.FileInputFormat: Total input paths to process : 1
21/04/26 15:19:32 INFO mapreduce.JobSubmitter: number of splits:1
21/04/26 15:19:33 INFO mapreduce.JobSubmitter: Submitting tokens for job: job_
local1167934544_0001
21/04/26 15:19:33 INFO mapreduce.Job: The url to track the job: http://localhos
t:8080/
21/04/26 15:19:33 INFO mapreduce.Job: Running job: job_local1167934544_0001
21/04/26 15:19:33 INFO mapred.LocalJobRunner: OutputCommitter set in config nul
l
21/04/26 15:19:33 INFO mapred.LocalJobRunner: OutputCommitter is org.apache.had
oop.mapreduce.lib.output.FileOutputCommitter
21/04/26 15:19:34 INFO mapreduce.Job: Job job_local1167934544_0001 running in u
ber mode : false
21/04/26 15:19:34 INFO mapreduce.Job: map 0% reduce 0%
21/04/26 15:19:34 INFO mapred.LocalJobRunner: Waiting for map tasks
21/04/26 15:19:34 INFO mapred.LocalJobRunner: Starting task: attempt_local11679
34544_0001_m_000000_0
21/04/26 15:19:34 INFO mapred.Task: Using ResourceCalculatorProcessTree : [ ]
21/04/26 15:19:34 INFO mapred.MapTask: Processing split: hdfs://localhost:54310
/input/input.txt:0+44
21/04/26 15:19:35 INFO mapred.MapTask: Input split: hdfs://localhost:54310/inpu
t/input.txt:0+44
```

```
21/04/26 15:19:37 INFO mapreduce.Job: Map 100% Reduce 100%
21/04/26 15:19:37 INFO mapreduce.Job: Job job_local1167934544_0001 completed
ccessfully
21/04/26 15:19:37 INFO mapreduce.Job: Counters: 38
  File System Counters
    FILE: Number of bytes read=541122
    FILE: Number of bytes written=1047373
    FILE: Number of read operations=0
    FILE: Number of large read operations=0
    FILE: Number of write operations=0
    HDFS: Number of bytes read=88
    HDFS: Number of bytes written=35
    HDFS: Number of read operations=15
    HDFS: Number of large read operations=0
    HDFS: Number of write operations=4
  Map-Reduce Framework
    Map input records=3
    Map output records=9
    Map output bytes=80
    Map output materialized bytes=61
    Input split bytes=103
    Combine input records=9
    Combine output records=5
    Reduce input groups=5
    Reduce shuffle bytes=61
    Reduce input records=5
    Reduce output records=5
    Spilled Records=10
    Shuffled Maps =1
    Failed Shuffles=0
```



```
Physical memory (bytes) snapshot=0
Virtual memory (bytes) snapshot=0
Total committed heap usage (bytes)=340787200

Shuffle Errors
BAD_ID=0
CONNECTION=0
IO_ERROR=0
WRONG_LENGTH=0
WRONG_MAP=0
WRONG_REDUCE=0

File Input Format Counters
Bytes Read=44
File Output Format Counters
Bytes Written=35

hduser@lab-VirtualBox: /home/lab/hadoop-2.6.0/share/hadoop/mapreduce$ hadoop dfs
-ls /firstExampleOut
DEPRECATED: Use of this script to execute hdfs command is deprecated.
Instead use the hdfs command for it.

21/04/26 15:20:15 WARN util.NativeCodeLoader: Unable to load native-hadoop libr
ary for your platform... using builtin-java classes where applicable
Found 2 items
-rw-r--r--  1 hduser supergroup          0 2021-04-26 15:19 /firstExampleOut/_
SUCCESS
-rw-r--r--  1 hduser supergroup        35 2021-04-26 15:19 /firstExampleOut/p
art-r-00000
hduser@lab-VirtualBox: /home/lab/hadoop-2.6.0/share/hadoop/mapreduce$ hadoop dfs
-cat /firstExampleOut/part-r-00000
DEPRECATED: Use of this script to execute hdfs command is deprecated.
```

```
Bytes Read=44
File Output Format Counters
Bytes Written=35

hduser@lab-VirtualBox: /home/lab/hadoop-2.6.0/share/hadoop/mapreduce$ hadoop dfs
-ls /firstExampleOut
DEPRECATED: Use of this script to execute hdfs command is deprecated.
Instead use the hdfs command for it.

21/04/26 15:20:15 WARN util.NativeCodeLoader: Unable to load native-hadoop libr
ary for your platform... using builtin-java classes where applicable
Found 2 items
-rw-r--r--  1 hduser supergroup          0 2021-04-26 15:19 /firstExampleOut/_
SUCCESS
-rw-r--r--  1 hduser supergroup        35 2021-04-26 15:19 /firstExampleOut/p
art-r-00000
hduser@lab-VirtualBox: /home/lab/hadoop-2.6.0/share/hadoop/mapreduce$ hadoop dfs
-cat /firstExampleOut/part-r-00000
DEPRECATED: Use of this script to execute hdfs command is deprecated.
Instead use the hdfs command for it.

21/04/26 15:22:01 WARN util.NativeCodeLoader: Unable to load native-hadoop libr
ary for your platform... using builtin-java classes where applicable
bear      2
car       3
deer      1
deer      1
river     2
hduser@lab-VirtualBox: /home/lab/hadoop-2.6.0/share/hadoop/mapreduce$ hadoop dfs
-cat /firstExampleOut/part-r-00000
```

## 9. Use of Join

### Driver Class

```
package MapReduceJoin;

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,
DeptNameMapper.class);
        MultipleInputs.addInputPath(conf, BInputPath, TextInputFormat.class,
DeptEmpStrengthMapper.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);
    }
}

```

## Mapper Class

```

package MapReduceJoin;

import java.io.IOException;

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

public class DeptNameMapper 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], "0"), new Text(SingleNodeData[1]));
    }
}

```

## Reducer Class

```
package MapReduceJoin;

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);
        }
    }
}
```

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

```
1 object HelloWorld {  
2   def main(args: Array[String]) {  
3     println("Hello world")  
4   }  
5 }
```

Hello world

HelloWorld.Scala (~)

File Edit View Search Tools Documents Help

⊕ 📁 💾 ↶ ↷ ✂ 📄 📋 🔍 ✎

HelloWorld.Scala x

```
object HelloWorld  
{  
  def main(args: Array[String])  
  {  
    //This is a Hello World function in Scala  
    println("Hello World!")  
  }  
}
```

```
scala> val sorted=ListMap(counts.collect.sortWith(_._2 > _._2):_*)  
[Stage 0:>] (0 + 1) / 1  
  
sorted: scala.collection.immutable.ListMap[String,Int] = ListMap(test -> 6, Hel  
lo -> 6, Hadoop -> 3, is -> 2, This -> 2, world -> 1, Test -> 1, a -> 1, to ->  
1, see -> 1, bye -> 1, file -> 1, the -> 1)  
  
scala> println(sorted)  
ListMap(test -> 6, Hello -> 6, Hadoop -> 3, is -> 2, This -> 2, world -> 1, Tes  
t -> 1, a -> 1, to -> 1, see -> 1, bye -> 1, file -> 1, the -> 1)  
  
scala> for((k,v)<-sorted)  
  {  
    if(v>4)  
    {  
      print(k+",")  
      print(v)  
      println()  
    }  
  }  
test,6  
Hello,6
```



## 11. Using RDD and FlatMap

### Code:

```
>val text = sc.textFile("abc.txt")  
  
  >val counts = text.flatMap(line => line.split(" ")).map(word => (word,1)).reduceByKey(_+_)  
  counts.collect  
  
>val greaterThan4=counts.filter(x=>x._2>4);  
  
>greaterThan4.collect().foreach(println)
```

### Input file:

Hello Hello World Hello Hello Xyz Xyz Xyz Hello World Hello Xyz World World Xyz Hello World

### Output:

Hello 7

World 5

Xyz 5