VISVESVARAYA TECHNOLOGICAL UNIVERSITY

"JnanaSangama", Belgaum -590014, Karnataka.

LAB REPORT
on

BIG DATA ANALYTICS
(20CS6PEBDA)

Submitted by

SNEHA SRIVASTAVA (1BM19CS158)

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 **SNEHA SRIVASTAVA (1BM19CS158),** 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. Shyamala G
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	4-9
2	Perform the following DB operations using Cassandra-Employee	10-13
3	Perform the following DB operations using Cassandra-Library	14-15
4	Hadoop Installation	16
5	Execution of HDFS Commands	17-18
6	MapReduce Program using NCDC dataset	19-24
7	Top 'n' maximum occurrence of words	25-28
8	Hadoop Map Reduce program to combine information from the users file	29-34
9	Scala Shell	35
10	Using RDD and FlatMap	36-37

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:

I. CREATE DATABASE IN MONGODB.

use myDB;

Confirm the existence of your database

db;

To list all databases

show dbs;

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

1. To create a collection by the name "Student".

db.createCollection("Student");

2. To drop a collection by the name "Student".

db.Student.drop();

3. Create a collection by the name "Students" and store the following data in it.

db.Student.insert({_id:1,StudName:"MichelleJacintha",Grade:"VII",Hobbies:"InternetS

urfing"});

4. Insert the document for "AryanDavid" in to the Students collection only if it does not already exist in the collection. However, if it is already present in the collection, then update the document with new values. (Update his hobbies from "Skating" to "Chess"). Use "Update else insert" (if there is an existing document, it will attempt to update it, if there is no existing document then it will insert it).

db.Student.update({_id:3,StudName:"AryanDavid",Grade:"VII"},{$set:{Hobbies:" Skatin

g"}},{upsert:true});

5. FIND METHOD

A. To search for documents from the "Students" collection based on certain search criteria.

db.Student.find({StudName:"Aryan David"});

B. To display only the StudName and Grade from all the documents of the Students collection. The identifier_id should be suppressed and NOT displayed.

db.Student.find({},{StudName:1,Grade:1,_id:0});

C. To find those documents where the Grade is set to 'VII'

db.Student.find({Grade:{$eq:'VII'}}).pretty();

D. To find those documents from the Students collection where the Hobbies is set to either 'Chess' or is set to 'Skating'.

db.Student.find({Hobbies :{ $in: ['Chess','Skating']}}).pretty ();

E. To find documents from the Students collection where the StudName begins with "M".

db.Student.find({StudName:/^M/}).pretty();

F. To find documents from the Students collection where the StudNamehas an "e" in any position.

db.Student.find({StudName:/e/}).pretty();

G. To find the number of documents in the Students collection.

db.Student.count();

H. To sort the documents from the Students collection in the descending order of StudName.

db.Student.find().sort({StudName:-1}).pretty();

III. Import data from a CSV file

Given a CSV file "sample.txt" in the D:drive, import the file into the MongoDB collection, "SampleJSON". The collection is in the database "test".

mongoimport --db Student --collection airlines --type csv –headerline --file /home/hduser/Desktop/airline.csv

IV. Export data to a CSV file

mongoexport --host localhost --db Student --collection airlines --csv /home/hduser/Desktop/output.txt –fields "Year","Quarter"

V. Save Method :

db.Students.save({StudName:"Vamsi", Grade:"VI"})

VI. Add a new field to existing Document:

db.Students.update({_id:4},{$set:{Location:"Network"}})

VII. Remove the field in an existing Document

db.Students.update({_id:4},{$unset:{Location:"Network"}})

VIII. Finding Document based on search criteria suppressing few fields

db.Student.find({_id:1},{StudName:1,Grade:1,_id:0});

To find those documents where the Grade is not set to 'VII'

db.Student.find({Grade:{$ne:'VII'}}).pretty();

To find documents from the Students collection where the StudName ends with s.

db.Student.find({StudName:/s$/}).pretty();

IX. to set a particular field value to NULL

db.Students.update({_id:3},{$set:{Location:null}})

X. Count the number of documents in Student Collections

db.Students.count()

XI. Count the number of documents in Student Collections with grade :VII

db.Students.count({Grade:”VII”})

Retrieve first 3 documents

db.Students.find({Grade:”VII”}).limit(3).pretty();

Sort the document in Ascending order

db.Students.find().sort({StudName:1}).pretty();

for descending order:

db.Students.find().sort({StudName:-1}).pretty();

to Skip the 1st two documents from the Students Collections

db.Students.find().skip(2).pretty()

XII. Create a collection by name “food” and add to each document add a “fruits” array

db.food.insert({ _id:1, fruits:[‘grapes’,’mango’,’apple’] })

db.food.insert({ _id:2, fruits:[‘grapes’,’mango’,’cherry’] })

db.food.insert({ _id:3, fruits:[‘banana’,’mango’] })

To find those documents from the “food” collection which has the “fruits array”

constitute of “grapes”, “mango” and “apple”.

db.food.find ({fruits: [‘grapes’,’mango’,’apple’] }). pretty().

To find in “fruits” array having “mango” in the first index position.

db.food.find ({‘fruits.1’:’grapes’})

To find those documents from the “food” collection where the size of the array is two.

db.food.find ({“fruits”: {$size:2}})

To find the document with a particular id and display the first two elements from the

array "fruits"

db.food.find({_id:1},{"fruits":{$slice:2}})

To find all the documents from the food collection which have elements mango and

grapes in the array "fruits"

db.food.find({fruits:{$all:["mango","grapes"]}})

Update on Array:

Using particular id replace the element present in the 1 st index position of the fruits

array with apple

db.food.update({_id:3},{$set:{'fruits.1':'apple'}})

Insert new key value pairs in the fruits array

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

XII. Aggregate Function :

Create a collection Customers with fields custID, AcctBal, AcctType.

Now group on "custID" and compute the sum of "AccBal".

db.Customers.aggregate ({$group : { _id : "$custID",TotAccBal : {$sum:" $AccBal"} } });

Match on AcctType:"S" then group on "CustID" and compute the sum of "AccBal".

db.Customers.aggregate ({$match:{AcctType:"S"}},{$group : { _id : "$custID",TotAccBal :

{$sum:"$AccBal"} } });

Match on AcctType:”S” then group on “CustID” and compute the sum of “AccBal” and

total balance greater than 1200.

db.Customers.aggregate ({$match:{AcctType:”S”}},{$group : { _id : “$custID”,TotAccBal :

{$sum:”$AccBal”} } }, {$match:{TotAccBal:{$gt:1200}}});

2. Perform the following DB operations using Cassandra.

1.Create a keyspace by name Employee

CREATE KEYSPACE employee123 WITH REPLICATION = {'class':'SimpleStrategy','replication_factor':1};

2. Create a column family by name

Employee-Info with attributes

Emp_Id Primary Key, Emp_Name,

Designation, Date_of_Joining, Salary,

Dept_Name

CREATE TABLE EMPLOYEEINFO(EMPID INT PRIMARY KEY, EMPNAME TEXT, DESIGNATION TEXT, DATEOFJOINING TIMESTAMP, SALARY DOUBLE, DEPTNAME TEXT);

3. Insert the values into the table in batch

Begin Batch

INSERT INTO EMPLOYEEINFO (EMPID, EMPNAME, DESIGNATION, DATEOFJOINING, SALARY, DEPTNAME) VALUES(1,'ABHISHEK','ASSISTANT MANAGER', '2010-04-26', 75000, 'MARKETING')

INSERT INTO EMPLOYEEINFO (EMPID, EMPNAME, DESIGNATION, DATEOFJOINING, SALARY, DEPTNAME) VALUES(2,'BHASKAR','ASSISTANT MANAGER', '2010-04-26', 75000, 'MARKETING')

INSERT INTO EMPLOYEEINFO (EMPID, EMPNAME, DESIGNATION, DATEOFJOINING, SALARY, DEPTNAME) VALUES(3,'CHIRAG','ASSISTANT MANAGER', '2010-04-26', 75000, 'MARKETING')

INSERT INTO EMPLOYEEINFO (EMPID, EMPNAME, DESIGNATION, DATEOFJOINING, SALARY, DEPTNAME)VALUES(4,'DHANUSH','ASSISTANT MANAGER', '2010-04-26', 75000, 'MARKETING')

```
INSERT INTO EMPLOYEEINFO (EMPID, EMPNAME, DESIGNATION,
DATEOFJOINING, SALARY, DEPTNAME) VALUES(5,'ESHAAN','ASSISTANT
MANAGER', '2010-04-26', 85000, 'TECHNICAL')

INSERT INTO EMPLOYEEINFO (EMPID, EMPNAME, DESIGNATION,
DATEOFJOINING, SALARY, DEPTNAME) VALUES(6,'FARAH','MANAGER',
'2010-04-26', 95000, 'TECHNICAL')

INSERT INTO EMPLOYEEINFO (EMPID, EMPNAME, DESIGNATION,
DATEOFJOINING, SALARY, DEPTNAME) VALUES(7,'GEMMA','MANAGER',
'2010-04-26', 95000, 'PR')

INSERT INTO EMPLOYEEINFO (EMPID, EMPNAME, DESIGNATION,
DATEOFJOINING, SALARY, DEPTNAME)VALUES(121,'HARRY','REGIONAL
MANAGER', '2010-04-26', 99000, 'MANAGEMENT')

APPLY BATCH;

SELECT * FROM EMPLOYEEINFO;
```

```
empid | dateofjoining                | deptname  | designation       | empname  | salary
-------+-------------------------------+-----------+-------------------+----------+--------
    5 | 2010-04-25 18:30:00.000000+0000 | TECHNICAL | ASSISTANT MANAGER |
ESHAAN |  85000
    1 | 2010-04-25 18:30:00.000000+0000 | MARKETING | ASSISTANT MANAGER |
ABHISHEK |  75000
    2 | 2010-04-25 18:30:00.000000+0000 | MARKETING | ASSISTANT MANAGER |
BHASKAR |  75000
    4 | 2010-04-25 18:30:00.000000+0000 | MARKETING | ASSISTANT MANAGER |
DHANUSH |  75000
  121 | 2010-04-25 18:30:00.000000+0000 | MANAGEMENT |  REGIONAL MANAGER |
HARRY |  99000
    7 | 2010-04-25 18:30:00.000000+0000 |        PR |          MANAGER |    GEMMA |  95000
    6 | 2010-04-25 18:30:00.000000+0000 | TECHNICAL |          MANAGER |    FARAH |
95000
    3 | 2010-04-25 18:30:00.000000+0000 | MARKETING | ASSISTANT MANAGER |
CHIRAG |  75000
```

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

UPDATE EMPLOYEEINFO SET EMPNAME='HARISH', DEPTNAME='PR' WHERE EMPID=121;

5. Sort the details of Employee records based on salary

SELECT * FROM EMPLOYEE_IN WHERE EMP_ID IN(1,2,3,4) ORDER BY SALARY DESC ALLOW FILTERING;

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.

ALTER TABLE EMPLOYEEINFO ADD PROJECTS LIST<TEXT>;

7. Update the altered table to add project names.

UPDATE EMPLOYEEINFO SET PROJECTS=['FACEBOOK','SNAPCHAT'] WHERE EMPID=1;

UPDATE EMPLOYEEINFO SET PROJECTS=['FACEBOOK','SNAPCHAT'] WHERE EMPID=7;

UPDATE EMPLOYEEINFO SET PROJECTS=['PINTEREST','INSTAGRAM'] WHERE EMPID=121;

UPDATE EMPLOYEEINFO SET PROJECTS=['PINTEREST','INSTAGRAM'] WHERE EMPID=4;

UPDATE EMPLOYEEINFO SET PROJECTS=['YOUTUBE','SPOTIFY'] WHERE EMPID=2;

UPDATE EMPLOYEEINFO SET PROJECTS=['YOUTUBE','SPOTIFY'] WHERE EMPID=3;

UPDATE EMPLOYEEINFO SET PROJECTS=['YOUTUBE','SPOTIFY'] WHERE EMPID=6;

UPDATE EMPLOYEEINFO SET PROJECTS=['TWITTER','REDDIT'] WHERE EMPID=5;

SELECT * FROM EMPLOYEEINFO;

```
empid | dateofjoining                | deptname  | designation       | empname  | projects                |
salary
-------+-------------------------------+-----------+-------------------+----------+-------------------------
+--------
5     | 2010-04-25 18:30:00.000000+0000 | TECHNICAL | ASSISTANT MANAGER |
ESHAAN |      ['TWITTER', 'REDDIT'] |  85000
1     | 2010-04-25 18:30:00.000000+0000 | MARKETING | ASSISTANT MANAGER |
ABHISHEK |   ['FACEBOOK', 'SNAPCHAT'] |  75000
2     | 2010-04-25 18:30:00.000000+0000 | MARKETING | ASSISTANT MANAGER |
BHASKAR |     ['YOUTUBE', 'SPOTIFY'] |  75000
4     | 2010-04-25 18:30:00.000000+0000 | MARKETING | ASSISTANT MANAGER |
DHANUSH | ['PINTEREST', 'INSTAGRAM'] |  75000
121   | 2010-04-25 18:30:00.000000+0000 |        PR |  REGIONAL MANAGER |   HARISH |
['PINTEREST', 'INSTAGRAM'] |  99000
7     | 2010-04-25 18:30:00.000000+0000 |        PR |           MANAGER |    GEMMA |
['FACEBOOK', 'SNAPCHAT'] |  95000
6     | 2010-04-25 18:30:00.000000+0000 | TECHNICAL |           MANAGER |    FARAH |
['YOUTUBE', 'SPOTIFY'] |  95000
3     | 2010-04-25 18:30:00.000000+0000 | MARKETING | ASSISTANT MANAGER |
CHIRAG |     ['YOUTUBE', 'SPOTIFY'] |  75000
```

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

SELECT TTL (EMPNAME) FROM EMPLOYEEINFO WHERE EMP_ID IN (1,2,3,4,5,6,7);

3. Perform the following DB operations using Cassandra.

1.Create a keyspace by name Library

CREATE KEYSPACE Library 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 LIBRARY_INFO_4 (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

UPDATE LIBRARY_INFO_4 SET COUNTER_VALUE+1 WHERE STUD_ID=121 AND STUD_NAME='SNEHA' AND BOOK_NAME='BDA' AND BOOK_ID=110 AND DATE_OF_ISSUE='2022-04-01';

UPDATE LIBRARY_INFO_4 SET COUNTER_VALUE+1 WHERE STUD_ID=122 AND STUD_NAME='RAHUL' AND BOOK_NAME='OOMD' AND BOOK_ID=111 AND DATE_OF_ISSUE='2022-07-03';

UPDATE LIBRARY_INFO_4 SET COUNTER_VALUE+1 WHERE STUD_ID=123 AND STUD_NAME='RITIKA' AND BOOK_NAME='ML' AND BOOK_ID=112 AND DATE_OF_ISSUE='2022-02-21';

UPDATE LIBRARY_INFO_4 SET COUNTER_VALUE+1 WHERE STUD_ID=124 AND STUD_NAME='ISHA' AND BOOK_NAME='AI' AND BOOK_ID=113 AND DATE_OF_ISSUE='2022-09-02';

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

SELECT * FROM LIBRARY_INFO_4;

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

SELECT * FROM LIBRARY_INFO_4 WHERE STUD_ID=112;

6. Export the created column to a csv file.

COPY LIBRARY_INFO_4 (STUD_ID, STUD_NAME, BOOK_NAME, BOOK_ID, DATE_OF_ISSUE, COUNTER_VALUE) TO 'C:\Users\Admin\OneDrive\Desktop\BDA Lab\data.csv';

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

COPY LIBRARY_INFO_4 (STUD_ID, STUD_NAME, BOOK_NAME, BOOK_ID, DATE_OF_ISSUE, COUNTER_VALUE) FROM 'C:\Users\Admin\OneDrive\Desktop\BDA Lab\data.csv';

4. Hadoop Installation.

```
[shashi@Shashis-MacBook-Air-2 ~ % hadoop -version
ERROR: -version is not COMMAND nor fully qualified CLASSNAME.
Usage: hadoop [OPTIONS] SUBCOMMAND [SUBCOMMAND OPTIONS]
  or    hadoop [OPTIONS] CLASSNAME [CLASSNAME OPTIONS]
    where CLASSNAME is a user-provided Java class

    OPTIONS is none or any of:

--config dir                     Hadoop config directory
--debug                          turn on shell script debug mode
--help                           usage information
buildpaths                       attempt to add class files from build tree
hostnames list[,of,host,names]   hosts to use in slave mode
hosts filename                   list of hosts to use in slave mode
loglevel level                   set the log4j level for this command
workers                          turn on worker mode

    SUBCOMMAND is one of:


      Admin Commands:

daemonlog      get/set the log level for each daemon

      Client Commands:

archive        create a Hadoop archive
checknative    check native Hadoop and compression libraries availability
classpath      prints the class path needed to get the Hadoop jar and the
               required libraries
conftest       validate configuration XML files
credential     interact with credential providers
distch         distributed metadata changer
distcp         copy file or directories recursively
dtutil         operations related to delegation tokens
envvars        display computed Hadoop environment variables
fs             run a generic filesystem user client
gridmix        submit a mix of synthetic job, modeling a profiled from
               production load
jar <jar>      run a jar file. NOTE: please use "yarn jar" to launch YARN
               applications, not this command.
jnipath        prints the java.library.path
kdiag          Diagnose Kerberos Problems
kerbname       show auth_to_local principal conversion
key            manage keys via the KeyProvider
rumenfolder    scale a rumen input trace
rumentrace     convert logs into a rumen trace
s3guard        manage metadata on S3
trace          view and modify Hadoop tracing settings
version        print the version

      Daemon Commands:

kms            run KMS, the Key Management Server
registrydns    run the registry DNS server

SUBCOMMAND may print help when invoked w/o parameters or with -h.
```

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

c:\hadoop_new\sbin>hdfs dfs -mkdir /temp

c:\hadoop_new\sbin>hdfs dfs -copyFromLocal E:\Desktop\sample.txt \temp c:
\hadoop_new\sbin>hdfs dfs -ls \temp
Found 1 items

-rw-r--r-- 1 Admin supergroup 11 2021-06-11 21:12 /temp/sample.txt c:
\hadoop_new\sbin>hdfs dfs -cat \temp\sample.txt hello
world

c:\hadoop_new\sbin>hdfs dfs -get \temp\sample.txt E:\Desktop\temp c:
\hadoop_new\sbin>hdfs dfs -put E:\Desktop\temp \temp c:\hadoop_new\sbin>hdfs dfs -ls
\temp
Found 2 items

-rw-r--r-- 1 Admin supergroup 11 2021-06-11 21:12 /temp/sample.txt drwxr-xr-x -

Admin supergroup 0 2021-06-11 21:15 /temp/temp c:\hadoop_new\sbin>hdfs dfs
-mv \lab1 \temp c:\hadoop_new\sbin>hdfs dfs -ls \temp Found 3 items drwxr-xr-x
- Admin supergroup 0 2021-04-19 15:07 /temp/lab1 -rw-r--r-- 1 Admin
7
supergroup 11 2021-06-11 21:12 /temp/sample.txt drwxr-xr-x -

Admin supergroup 0 2021-06-11 21:15 /temp/temp c:
\hadoop_new\sbin>hdfs dfs -rm /temp/sample.txt Deleted
/temp/sample.txt
c:\hadoop_new\sbin>hdfs dfs -ls \temp Found 2 items drwxr-xr-x - Admin
supergroup 0 2021-04-19 15:07 /temp/lab1 drwxr-xr-x - Admin

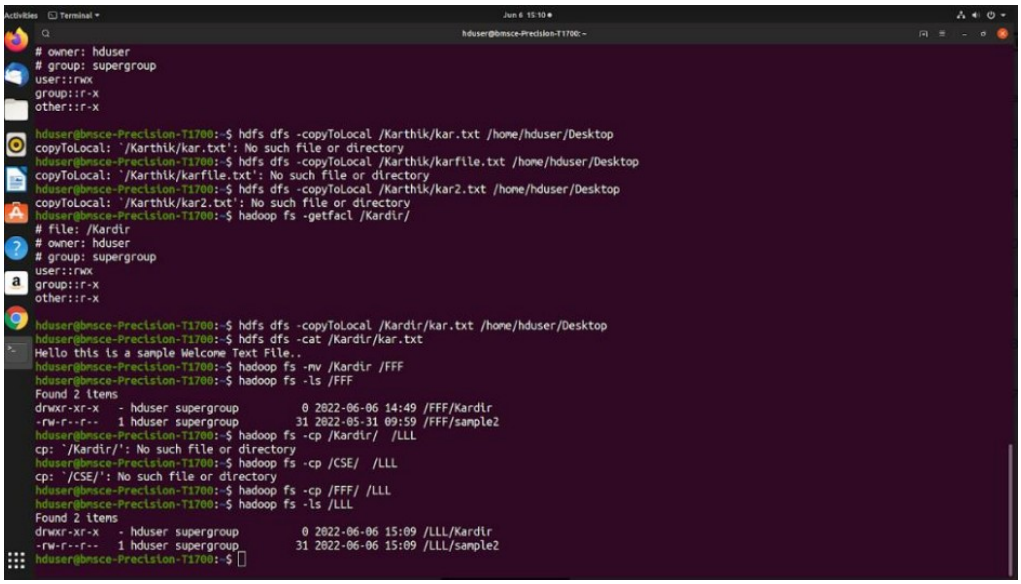
supergroup 0 2021-06-11 21:15 /temp/temp

c:\hadoop_new\sbin>hdfs dfs -copyFromLocal E:\Desktop\sample.txt \temp c:
\hadoop_new\sbin>hdfs dfs -ls \temp Found 3 items drwxr-xr-x - Admin supergroup 0
2021-04-19 15:07 /temp/lab1 -rw-r--r-- 1 Admin supergroup
11 2021-06-11 21:17 /temp/sample.txt drwxr-xr-x - Admin supergroup 0

2021-06-11 21:15 /temp/temp

c:\hadoop_new\sbin>hdfs dfs -copyToLocal \temp\sample.txt E:\Desktop\sample.txt

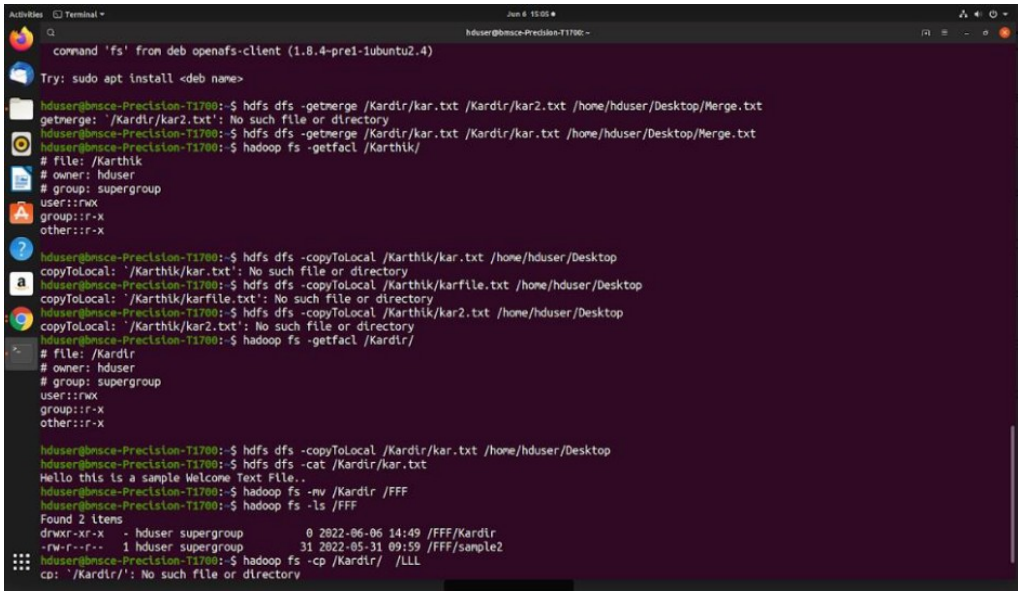
OUTPUT :



```
# owner: hduser
# group: supergroup
user::rwx
group::r-x
other::r-x

hduser@bmsce-Precision-T1700:~$ hdfs dfs -copyToLocal /Karthik/kar.txt /home/hduser/Desktop
copyToLocal: `/Karthik/kar.txt': No such file or directory
hduser@bmsce-Precision-T1700:~$ hdfs dfs -copyToLocal /Karthik/karfile.txt /home/hduser/Desktop
copyToLocal: `/Karthik/karfile.txt': No such file or directory
hduser@bmsce-Precision-T1700:~$ hdfs dfs -copyToLocal /Karthik/kar2.txt /home/hduser/Desktop
copyToLocal: `/Karthik/kar2.txt': No such file or directory
hduser@bmsce-Precision-T1700:~$ hadoop fs -getfacl /Kardir/
# file: /Kardir
# owner: hduser
# group: supergroup
user::rwx
group::r-x
other::r-x

hduser@bmsce-Precision-T1700:~$ hdfs dfs -copyToLocal /Kardir/kar.txt /home/hduser/Desktop
hduser@bmsce-Precision-T1700:~$ hdfs dfs -cat /Kardir/kar.txt
Hello this is a sample Welcome Text File..
hduser@bmsce-Precision-T1700:~$ hadoop fs -mv /Kardir /FFF
hduser@bmsce-Precision-T1700:~$ hadoop fs -ls /FFF
Found 2 items
drwxr-xr-x   - hduser supergroup          0 2022-06-06 14:49 /FFF/Kardir
-rw-r--r--   1 hduser supergroup         31 2022-05-31 09:59 /FFF/sample2
hduser@bmsce-Precision-T1700:~$ hadoop fs -cp /Kardir/  /LLL
cp: `/Kardir/': No such file or directory
hduser@bmsce-Precision-T1700:~$ hadoop fs -cp /CSE/  /LLL
cp: `/CSE/': No such file or directory
hduser@bmsce-Precision-T1700:~$ hadoop fs -cp /FFF/ /LLL
hduser@bmsce-Precision-T1700:~$ hadoop fs -ls /LLL
Found 2 items
drwxr-xr-x   - hduser supergroup          0 2022-06-06 15:09 /LLL/Kardir
-rw-r--r--   1 hduser supergroup         31 2022-06-06 15:09 /LLL/sample2
hduser@bmsce-Precision-T1700:~$
```



```
command 'fs' from deb openafs-client (1.8.4-pre1-1ubuntu2.4)

Try: sudo apt install <deb name>

hduser@bmsce-Precision-T1700:~$ hdfs dfs -getmerge /Kardir/kar.txt /Kardir/kar2.txt /home/hduser/Desktop/Merge.txt
getmerge: `/Kardir/kar2.txt': No such file or directory
hduser@bmsce-Precision-T1700:~$ hdfs dfs -getmerge /Kardir/kar.txt /Kardir/kar.txt /home/hduser/Desktop/Merge.txt
hduser@bmsce-Precision-T1700:~$ hadoop fs -getfacl /Karthik/
# file: /Karthik
# owner: hduser
# group: supergroup
user::rwx
group::r-x
other::r-x

hduser@bmsce-Precision-T1700:~$ hdfs dfs -copyToLocal /Karthik/kar.txt /home/hduser/Desktop
copyToLocal: `/Karthik/kar.txt': No such file or directory
hduser@bmsce-Precision-T1700:~$ hdfs dfs -copyToLocal /Karthik/karfile.txt /home/hduser/Desktop
copyToLocal: `/Karthik/karfile.txt': No such file or directory
hduser@bmsce-Precision-T1700:~$ hdfs dfs -copyToLocal /Karthik/kar2.txt /home/hduser/Desktop
copyToLocal: `/Karthik/kar2.txt': No such file or directory
hduser@bmsce-Precision-T1700:~$ hadoop fs -getfacl /Kardir/
# file: /Kardir
# owner: hduser
# group: supergroup
user::rwx
group::r-x
other::r-x

hduser@bmsce-Precision-T1700:~$ hdfs dfs -copyToLocal /Kardir/kar.txt /home/hduser/Desktop
hduser@bmsce-Precision-T1700:~$ hdfs dfs -cat /Kardir/kar.txt
Hello this is a sample Welcome Text File..
hduser@bmsce-Precision-T1700:~$ hadoop fs -mv /Kardir /FFF
hduser@bmsce-Precision-T1700:~$ hadoop fs -ls /FFF
Found 2 items
drwxr-xr-x   - hduser supergroup          0 2022-06-06 14:49 /FFF/Kardir
-rw-r--r--   1 hduser supergroup         31 2022-05-31 09:59 /FFF/sample2
hduser@bmsce-Precision-T1700:~$ hadoop fs -cp /Kardir/  /LLL
cp: `/Kardir/': No such file or directory
```

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

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

```
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);
  }
}

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));
  }
}
```

b) find the mean max temperature for every month

MeanMax

```
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);
  }
}


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));
  }
}

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));
  }
}

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 :

```
hduser@ubuntu:~/hadoop-3.2.1/sbin$ hdfs dfs -ls /
2021-05-10 23:17:54,055 WARN util.NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-java classes where applicable
Found 4 items
-rw-r--r--   1 hduser supergroup     888190 2021-05-06 02:21 /NCDC_Dataset
drwxr-xr-x   - hduser supergroup          0 2021-04-29 02:41 /home
drwxr-xr-x   - hduser supergroup          0 2021-04-26 02:47 /rgs
drwx------   - hduser supergroup          0 2021-04-29 02:40 /tmp
hduser@ubuntu:~/hadoop-3.2.1/sbin$ hadoop jar /home/hduser/Desktop/Temperature_up.jar /NCDC_Dataset /lab5_Output
2021-05-10 23:20:34,241 WARN util.NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-java classes where applicable
2021-05-10 23:20:35,819 INFO client.RMProxy: Connecting to ResourceManager at /127.0.0.1:8032
2021-05-10 23:20:36,826 WARN mapreduce.JobResourceUploader: Hadoop command-line option parsing not performed. Implement the Tool interface and execute your application with ToolRunner to remedy this.
2021-05-10 23:20:36,982 INFO mapreduce.JobResourceUploader: Disabling Erasure Coding for path: /tmp/hadoop-yarn/staging/hduser/.staging/job_1620713846028_0001
2021-05-10 23:20:37,352 INFO sasl.SaslDataTransferClient: SASL encryption trust check: localHostTrusted = false, remoteHostTrusted = false
2021-05-10 23:20:38,622 INFO input.FileInputFormat: Total input files to process : 1
2021-05-10 23:20:38,710 INFO sasl.SaslDataTransferClient: SASL encryption trust check: localHostTrusted = false, remoteHostTrusted = false
2021-05-10 23:20:38,765 INFO sasl.SaslDataTransferClient: SASL encryption trust check: localHostTrusted = false, remoteHostTrusted = false
2021-05-10 23:20:38,793 INFO mapreduce.JobSubmitter: number of splits:1
2021-05-10 23:20:39,091 INFO sasl.SaslDataTransferClient: SASL encryption trust check: localHostTrusted = false, remoteHostTrusted = false
2021-05-10 23:20:39,149 INFO mapreduce.JobSubmitter: Submitting tokens for job: job_1620713846028_0001
2021-05-10 23:20:39,149 INFO mapreduce.JobSubmitter: Executing with tokens: []
2021-05-10 23:20:39,572 INFO conf.Configuration: resource-types.xml not found
2021-05-10 23:20:39,573 INFO resource.ResourceUtils: Unable to find 'resource-types.xml'.
2021-05-10 23:20:40,296 INFO impl.YarnClientImpl: Submitted application application_1620713846028_0001
2021-05-10 23:20:40,376 INFO mapreduce.Job: The url to track the job: http://ubuntu:8088/proxy/application_1620713846028_0001/
2021-05-10 23:20:40,377 INFO mapreduce.Job: Running job: job_1620713846028_0001
2021-05-10 23:20:54,068 INFO mapreduce.Job: Job job_1620713846028_0001 running in uber mode : false
2021-05-10 23:20:54,072 INFO mapreduce.Job:  map 0% reduce 0%
2021-05-10 23:21:03,308 INFO mapreduce.Job:  map 100% reduce 0%
2021-05-10 23:21:13,443 INFO mapreduce.Job:  map 100% reduce 100%
2021-05-10 23:21:14,477 INFO mapreduce.Job: Job job_1620713846028_0001 completed successfully
2021-05-10 23:21:14,743 INFO mapreduce.Job: Counters: 54
        File System Counters
                FILE: Number of bytes read=72210
                FILE: Number of bytes written=594857
                FILE: Number of read operations=0
                FILE: Number of large read operations=0
                FILE: Number of write operations=0
                HDFS: Number of bytes read=888289
                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
```

```
                Bytes Written=8
hduser@ubuntu:~/hadoop-3.2.1/sbin$ hdfs dfs -ls /lab5_Output
2021-05-10 23:21:36,021 WARN util.NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-java classes where applicable
Found 2 items
-rw-r--r--   1 hduser supergroup          0 2021-05-10 23:21 /lab5_Output/_SUCCESS
-rw-r--r--   1 hduser supergroup          8 2021-05-10 23:21 /lab5_Output/part-r-00000
hduser@ubuntu:~/hadoop-3.2.1/sbin$ hdfs dfs -cat /lab5_Output/part-r-00000
2021-05-10 23:22:09,025 WARN util.NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-java classes where applicable
2021-05-10 23:22:10,985 INFO sasl.SaslDataTransferClient: SASL encryption trust check: localHostTrusted = false, remoteHostTrusted = false
1901    46
```

```
        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)=6777
                Total time spent by all reduces in occupied slots (ms)=7015
                Total time spent by all map tasks (ms)=6777
                Total time spent by all reduce tasks (ms)=7015
                Total vcore-milliseconds taken by all map tasks=6777
                Total vcore-milliseconds taken by all reduce tasks=7015
                Total megabyte-milliseconds taken by all map tasks=6939648
                Total megabyte-milliseconds taken by all reduce tasks=7183360
        Map-Reduce Framework
                Map input records=6565
                Map output records=6564
                Map output bytes=59076
                Map output materialized bytes=72210
                Input split bytes=99
                Combine input records=0
                Combine output records=0
                Reduce input groups=1
                Reduce shuffle bytes=72210
                Reduce input records=6564
                Reduce output records=1
                Spilled Records=13128
                Shuffled Maps =1
                Failed Shuffles=0
                Merged Map outputs=1
                GC time elapsed (ms)=237
                CPU time spent (ms)=3000
                Physical memory (bytes) snapshot=447746048
                Virtual memory (bytes) snapshot=5058613248
                Total committed heap usage (bytes)=400556032
                Peak Map Physical memory (bytes)=269074432
                Peak Map Virtual memory (bytes)=2524450816
                Peak Reduce Physical memory (bytes)=178671616
                Peak Reduce Virtual memory (bytes)=2534162432
        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=888190
        File Output Format Counters
                Bytes Written=8
```

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.

```
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.Reducer; import
org.apache.hadoop.mapreduce.lib.input.FileInputFormat; import
org.apache.hadoop.mapreduce.lib.output.FileOutputFormat; import
org.apache.hadoop.util.GenericOptionsParser; import utils.MiscUtils; import
java.io.IOException; import java.util.*;

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.setCombinerClass(TopNReducer.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);

}

/**

* The mapper reads one line at the time, splits it into an array of single words and emits every *

word to the reducers with the value of 1.

*/

public static class TopNMapper extends Mapper<Object, Text, Text, IntWritable> {
```

```
private final static IntWritable one = new IntWritable(1); private Text word = new Text();

private String tokens = "[_|$#<>\\^=\\[\\]\\*/\\\\,;,.\\-:()?!\'"]";

@Override

public void map(Object key, Text value, Context context) throws IOException,
InterruptedException {

String cleanLine = value.toString().toLowerCase().replaceAll(tokens, " "); StringTokenizer itr

= new StringTokenizer(cleanLine); while (itr.hasMoreTokens()) { word.set(itr.nextToken().trim());
context.write(word, one);

}

}

}

/**

* The reducer retrieves every word and puts it into a Map: if the word already exists in the *
map,

increments its value, otherwise sets it to 1.

*/

public static class TopNReducer extends Reducer<Text, IntWritable, Text, IntWritable> {

private Map<Text, IntWritable> countMap = new HashMap<>(); @Override

public void reduce(Text key, Iterable<IntWritable> values, Context context) throws IOException,

InterruptedException {

// computes the number of occurrences of a single word int sum = 0; for (IntWritable val :
values) { sum += val.get();

}

// puts the number of occurrences of this word into the map.

// We need to create another Text object because the Text instance

// we receive is the same for all the words countMap.put(new Text(key), new IntWritable(sum));

}

@Override

protected void cleanup(Context context) throws IOException, InterruptedException {
```

Map<Text, IntWritable> sortedMap =MiscUtils.sortByValues(countMap); int counter = 0; for (Text key : sortedMap.keySet()) { if (counter++ == 3) {

break;

}

context.write(key, sortedMap.get(key));

}

}

}

/**

* The combiner retrieves every word and puts it into a Map: if the word already exists in the *

map, increments its value, otherwise sets it to 1.

*/

public static class TopNCombiner extends Reducer<Text, IntWritable, Text, IntWritable> {

18

@Override

public void reduce(Text key, Iterable<IntWritable> values, Context context) throws IOException,

InterruptedException {

// computes the number of occurrences of a single word int sum = 0; for (IntWritable val : values) { sum += val.get();

}

context.write(key, new IntWritable(sum));

}

}

}

// MiscUtils.java package utils; import java.util.*;

public class MiscUtils {

/**

sorts the map by values. Taken from:

```
http://javarevisited.blogspot.it/2012/12/how-to-sort-hashmap-java-by-key-and-va lue.html

*/


public static <K extends Comparable, V extends Comparable> Map<K, V>
sortByValues(Map<K, V>

map) {

List<Map.Entry<K, V>> entries = new LinkedList<Map.Entry<K, V>>(map.entrySet());

Collections.sort(entries, new Comparator<Map.Entry<K, V>>() {

@Override public int compare(Map.Entry<K, V> o1, Map.Entry<K, V> o2) { return
o2.getValue().compareTo(o1.getValue());

}

});

//LinkedHashMap will keep the keys in the order they are inserted

//which is currently sorted on natural ordering Map<K, V> sortedMap = new
LinkedHashMap<K, V>(); for (Map.Entry<K, V> entry : entries) {

sortedMap.put(entry.getKey(), entry.getValue());

}

return sortedMap;

}

}
```

OUTPUT :

```
C:\hadoop_new\share\hadoop\mapreduce>hdfs dfs -cat \sortwordsOutput\part-r-00000
car     7
deer    6
bear    3
```

8. Create a Hadoop Map Reduce program to combine information from the users file along with Information from the posts file by using the concept of join and display user_id, Reputation and Score.

```
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.lib.MultipleInputs; 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); 21

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);

}

}

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);
}
}
}
```

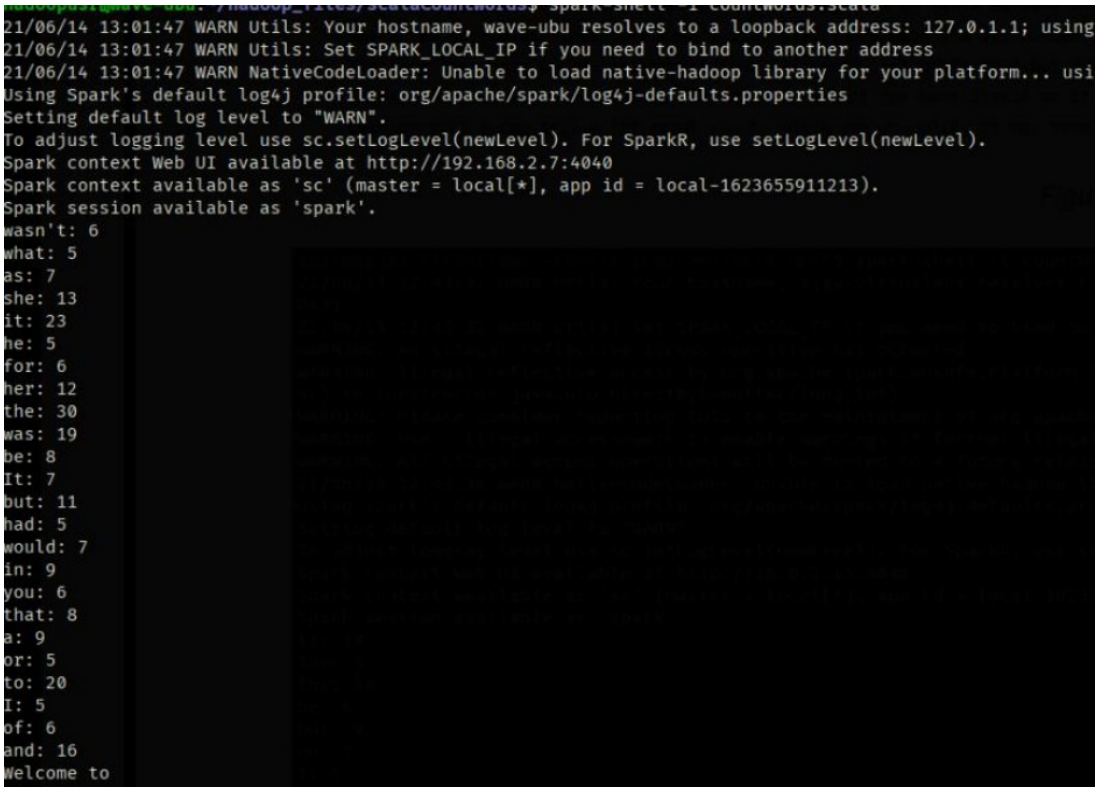
OUTPUT :

```
c:\hadoop_new\share\hadoop\mapreduce>hdfs dfs -cat \joinOutput\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("sparkdata.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;
```

OUTPUT :



```
21/06/14 13:01:47 WARN Utils: Your hostname, wave-ubu resolves to a loopback address: 127.0.1.1; using
21/06/14 13:01:47 WARN Utils: Set SPARK_LOCAL_IP if you need to bind to another address
21/06/14 13:01:47 WARN NativeCodeLoader: Unable to load native-hadoop library for your platform... usi
Using Spark's default log4j profile: org/apache/spark/log4j-defaults.properties
Setting default log level to "WARN".
To adjust logging level use sc.setLogLevel(newLevel). For SparkR, use setLogLevel(newLevel).
Spark context Web UI available at http://192.168.2.7:4040
Spark context available as 'sc' (master = local[*], app id = local-1623655911213).
Spark session available as 'spark'.
wasn't: 6
what: 5
as: 7
she: 13
it: 23
he: 5
for: 6
her: 12
the: 30
was: 19
be: 8
It: 7
but: 11
had: 5
would: 7
in: 9
you: 6
that: 8
a: 9
or: 5
to: 20
I: 5
of: 6
and: 16
Welcome to
```

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.

- **Commands and Output:**

```
scala> val textFile=sc.textFile("/home/hduser/Desktop/sample.txt");

textFile: org.apache.spark.rdd.RDD[String] = /home/hduser/Desktop/sample.txt
MapPartitionsRDD[8] at textFile at <console>:24

scala> val counts=textFile.flatMap(line=>line.split("
")).map(word=>(word,1)).reduceByKey(_=_)

<console>:25: error: reassignment to val

      val counts=textFile.flatMap(line=>line.split("
")).map(word=>(word,1)).reduceByKey(_=_)

                                                    ^

scala> val counts=textFile.flatMap(line=>line.split("
")).map(word=>(word,1)).reduceByKey(_+_)

counts: org.apache.spark.rdd.RDD[(String, Int)] = ShuffledRDD[11] 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):_*)

sorted: scala.collection.immutable.ListMap[String,Int] = Map(is -> 4, how -> 4, your -> 4,
are -> 1, brother -> 1, sister -> 1, family -> 1, ypu -> 1, job -> 1, hi -> 1,
hw -> 1)

scala> println(sorted)
```

```
Map(is -> 4, how -> 4, your -> 4, are -> 1, brother -> 1, sister -> 1, family -> 1, ypu
-> 1, job -> 1, hi -> 1, hw -> 1)


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