VISVESVARAYA TECHNOLOGICAL UNIVERSITY

"JnanaSangama", Belgaum -590014, Karnataka.

LAB REPORT on

BIG DATA ANALYTICS
(20CS6PEBDA)

Submitted by

DHRUVA M (1BM19CS049)

in partial fulfillment for the award of the degree of

BACHELOR OF ENGINEERING

in

COMPUTER SCIENCE AND ENGINEERING

B.M.S. COLLEGE OF ENGINEERING

BENGALURU-560019

(Autonomous Institution under VTU)

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 **DHRUVA M(1BM19CS049),** 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.

Antara Roy Choudury
Assistant Professor
Department of CSE
BMSCE, Bengaluru

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

`

Index Sheet

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

Lab 1

```
cqlsh> create keyspace "Employee1" with replication={'class':'SimpleStrategy','replication_factor':1};
cqlsh> describe keyspaces

library_db   system_auth  lab3               emp1          employees
employee138  system       "Employee"         employee
"Employee1"  library      system_schema      system_traces
university   sushma       system_distributed  university1

cqlsh> USE "Employee";
cqlsh:Employee> create table employee_info( Emp_Id int PRIMARY KEY, Emp_Name text, Designation text, Date_Of_joining timestamp, Salary int, Dept_Name text);
cqlsh:Employee> describe employee_info;

CREATE TABLE "Employee".employee_info (
    emp_id int PRIMARY KEY,
    date_of_joining timestamp,
    dept_name text,
    designation text,
    emp_name text,
    salary int
) WITH bloom_filter_fp_chance = 0.01
    AND caching = {'keys': 'ALL', 'rows_per_partition': 'NONE'}
    AND comment = ''
    AND compaction = {'class': 'org.apache.cassandra.db.compaction.SizeTieredCompactionStrategy', 'max_threshold': '32', 'min_threshold': '4'}
    AND compression = {'chunk_length_in_kb': '64', 'class': 'org.apache.cassandra.io.compress.LZ4Compressor'}
    AND crc_check_chance = 1.0
    AND dclocal_read_repair_chance = 0.1
    AND default_time_to_live = 0
    AND gc_grace_seconds = 864000
    AND max_index_interval = 2048
    AND memtable_flush_period_in_ms = 0
    AND min_index_interval = 128
    AND read_repair_chance = 0.0
    AND speculative_retry = '99PERCENTILE';
```

```
cqlsh:Employee> BEGIN BATCH
            ... INSERT INTO employee_info(Emp_Id,Emp_Name,Designation,Date_Of_joining,Salary,Dept_Name) VALUES(121,'Rose','Software Developer','2021-03-16',80000,'IT')
            ... INSERT INTO employee_info(Emp_Id,Emp_Name,Designation,Date_Of_joining,Salary,Dept_Name) VALUES(122,'Dhruva','Software Tester','2020-04-16',70000,'IT')
            ... NSERT INTO employee_info(Emp_Id,Emp_Name,Designation,Date_Of_joining,Salary,Dept_Name) VALUES(123,'John','Manager','2020-05-25',65000,'Sales')
            ... APPLY BATCH;
SyntaxException: line 4:0 mismatched input 'NSERT' expecting K_APPLY (....'Software Tester', '2020-04-16',70000,'IT')[NSERT]...)
cqlsh:Employee> BEGIN BATCH
            ... INSERT INTO employee_info(Emp_Id,Emp_Name,Designation,Date_Of_joining,Salary,Dept_Name) VALUES(121,'Rose','Software Developer','2021-03-16',80000,'IT')
            ... INSERT INTO employee_info(Emp_Id,Emp_Name,Designation,Date_Of_joining,Salary,Dept_Name) VALUES(122,'Dhruva','Software Tester','2020-04-16',70000,'IT')
            ... INSERT INTO employee_info(Emp_Id,Emp_Name,Designation,Date_Of_joining,Salary,Dept_Name) VALUES(123,'John','Manager','2020-05-25',65000,'Sales')
            ... APPLY BATCH;
cqlsh:Employee> SELECT * FROM employee_info;

 emp_id | date_of_joining                 | dept_name | designation        | emp_name | salary
--------+---------------------------------+-----------+--------------------+----------+--------
    123 | 2020-05-24 18:30:00.000000+0000 |     Sales |            Manager |     John |  65000
    122 | 2020-04-15 18:30:00.000000+0000 |        IT |    Software Tester |   Dhruva |  70000
    121 | 2021-03-15 18:30:00.000000+0000 |        IT | Software Developer |     Rose |  80000
```

```
(3 rows)
cqlsh:Employee> UPDATE employee_info SET Emp_Name='Rosy', Dept_Name='Software'  WHERE Emp_Id=121;
cqlsh:Employee> SELECT * FROM employee_info;

 emp_id | date_of_joining                 | dept_name | designation        | emp_name | salary
--------+---------------------------------+-----------+--------------------+----------+--------
    123 | 2020-05-24 18:30:00.000000+0000 |     Sales |            Manager |     John |  65000
    122 | 2020-04-15 18:30:00.000000+0000 |        IT |    Software Tester |   Dhruva |  70000
    121 | 2021-03-15 18:30:00.000000+0000 |  Software | Software Developer |     Rosy |  80000

(3 rows)
cqlsh:Employee> ALTER TABLE employee_info
            ... ADD projects set<text>;
cqlsh:Employee> SELECT * FROM employee_info;

 emp_id | date_of_joining                 | dept_name | designation        | emp_name | projects | salary
--------+---------------------------------+-----------+--------------------+----------+----------+--------
    123 | 2020-05-24 18:30:00.000000+0000 |     Sales |            Manager |     John |     null |  65000
    122 | 2020-04-15 18:30:00.000000+0000 |        IT |    Software Tester |   Dhruva |     null |  70000
    121 | 2021-03-15 18:30:00.000000+0000 |  Software | Software Developer |     Rosy |     null |  80000
```

```
(3 rows)
cqlsh:Employee> UPDATE employee_info SET projects={'sales improvement proj','ad management sys'} WHERE Emp_ID=123;
cqlsh:Employee> UPDATE employee_info SET projects={'company website','Employee management app'} WHERE Emp_ID=121;
cqlsh:Employee> UPDATE employee_info SET projects={'company website testing'} WHERE Emp_ID=122;
cqlsh:Employee> SELECT * FROM employee_info;

 emp_id | date_of_joining                 | dept_name | designation        | emp_name | projects                                                | salary
--------+---------------------------------+-----------+--------------------+----------+---------------------------------------------------------+--------
    123 | 2020-05-24 18:30:00.000000+0000 |     Sales |            Manager |     John | {'ad management sys', 'sales improvement proj'} |  65000
    122 | 2020-04-15 18:30:00.000000+0000 |        IT |    Software Tester |   Dhruva |                                 {'company website testing'} |  70000
    121 | 2021-03-15 18:30:00.000000+0000 |  Software | Software Developer |     Rosy |  {'Employee management app', 'company website'} |  80000
```

```
(3 rows)
cqlsh:Employee> BEGIN BATCH
           ... INSERT INTO employee_info(Emp_Id,Emp_Name,Designation,Date_Of_joining,Salary,Dept_Name,projects) VALUES(124,'Joe','Intern','2021-03-20',25000,'IT',{'LMS'}) USING TTL 15
           ... APPLY BATCH;
cqlsh:Employee> SELECT * FROM employee_info;

 emp_id | date_of_joining                 | dept_name | designation        | emp_name | projects                                                | salary
--------+---------------------------------+-----------+--------------------+----------+---------------------------------------------------------+--------
    123 | 2020-05-24 18:30:00.000000+0000 |     Sales |            Manager |     John | {'ad management sys', 'sales improvement proj'} |  65000
    122 | 2020-04-15 18:30:00.000000+0000 |        IT |    Software Tester |   Dhruva |                                 {'company website testing'} |  70000
    121 | 2021-03-15 18:30:00.000000+0000 |  Software | Software Developer |     Rosy |  {'Employee management app', 'company website'} |  80000
    124 | 2021-03-19 18:30:00.000000+0000 |        IT |             Intern |      Joe |                                                     {'LMS'} |  25000

(4 rows)
cqlsh:Employee> SELECT * FROM employee_info;

 emp_id | date_of_joining                 | dept_name | designation        | emp_name | projects                                                | salary
--------+---------------------------------+-----------+--------------------+----------+---------------------------------------------------------+--------
    123 | 2020-05-24 18:30:00.000000+0000 |     Sales |            Manager |     John | {'ad management sys', 'sales improvement proj'} |  65000
    122 | 2020-04-15 18:30:00.000000+0000 |        IT |    Software Tester |   Dhruva |                                 {'company website testing'} |  70000
    121 | 2021-03-15 18:30:00.000000+0000 |  Software | Software Developer |     Rosy |  {'Employee management app', 'company website'} |  80000
```

LAB 2

```
cqlsh> CREATE KEYSPACE "Library" WITH REPLICATION = { 'class':'SimpleStrategy', 'replication_factor':1};
cqlsh> USE "Library";

cqlsh:Library> CREATE TABLE library_info (
   ...   stud_id int,
   ...   stud_name text,
   ...   book_name text,
   ...   book_id int,
   ...   date_of_issue timestamp,
   ...   counter_value counter,
   ...   PRIMARY KEY (stud_id, stud_name, book_name, book_id, date_of_issue)
   ... );

cqlsh:Library> DESCRIBE KEYSPACES;

"Library"  system_auth        system_schema  system_views
system     system_distributed  system_traces  system_virtual_schema

cqlsh:Library> DESCRIBE TABLE Library_Info;

CREATE TABLE "Library".library_info (
stud_id int,
stud_name text,
book_name text,
book_id int,
date_of_issue timestamp,
counter_value counter,
PRIMARY KEY (stud_id, stud_name, book_name, book_id, date_of_issue)
) WITH CLUSTERING ORDER BY (stud_name ASC, book_name ASC, book_id ASC, date_of_issue ASC)
AND additional_write_policy = '99p'
AND bloom_filter_fp_chance = 0.01
```

```
   AND caching = {'keys': 'ALL', 'rows_per_partition': 'NONE'}
   AND cdc = false
   AND comment = ''
   AND compaction = {'class': 'org.apache.cassandra.db.compaction.SizeTieredCompactionStrategy', 'max_threshold':
'32',          '   min_threshold': '4'}
   AND compression = {'chunk_length_in_kb': '16', 'class': 'org.apache.cassandra.io.compress.LZ4Compressor'}
   AND crc_check_chance = 1.0
   AND default_time_to_live = 0
   AND extensions = {}
   AND gc_grace_seconds = 864000
   AND max_index_interval = 2048
   AND memtable_flush_period_in_ms = 0
   AND min_index_interval = 128
   AND read_repair = 'BLOCKING'
   AND speculative_retry = '99p';

cqlsh:Library> UPDATE  Library_Info  SET Counter_value=Counter_value+1 where Stud_Id=1 and Stud_Name='Dhruva'
and Book_name='BDA' and Book_id=111 and Date_Of_Issue='2021-03-15';
cqlsh:Library> UPDATE  Library_Info  SET Counter_value=Counter_value+1 where Stud_Id=2 and Stud_Name='Priya' and
Book_name='OOMD' and Book_id=112 and Date_Of_Issue='2021-02-12';
cqlsh:Library> UPDATE  Library_Info  SET Counter_value=Counter_value+1 where Stud_Id=112 and Stud_Name='Aswin'
and Book_name='BDA' and Book_id=1123 and Date_Of_Issue='2021-01-18';
cqlsh:Library> SELECT * FROM Library_Info;
 stud_id | stud_name | book_name | book_id | date_of_issue                | counter_value
---------+-----------+-----------+---------+---------------------------------+---------------
     1 |    Dhruva |       BDA |     111 | 2021-03-15 00:00:00.000000+0000 |             1
     2 |     Priya |      OOMD |     112 | 2021-02-12 00:00:00.000000+0000 |             1
   112 |     Aswin |       BDA |    1123 | 2021-01-18 00:00:00.000000+0000 |             1

(3 rows)

cqlsh:Library> UPDATE  Library_Info  SET Counter_value=Counter_value+1 where Stud_Id=112 and Stud_Name='Aswin'
and Book_name='BDA' and Book_id=1123 and Date_Of_Issue='2021-01-18';
cqlsh:Library> SELECT * FROM Library_Info;
 stud_id | stud_name | book_name | book_id | date_of_issue                | counter_value
```

```
---------+-----------+-----------+---------+---------------------------------+---------------
      1 |    Dhruva |       BDA |     111 | 2021-03-15 00:00:00.000000+0000 |           1
      2 |    Priya  |      OOMD |     112 | 2021-02-12 00:00:00.000000+0000 |           1
    112 |    Aswin  |       BDA |    1123 | 2021-01-18 00:00:00.000000+0000 |           2

(3 rows)

   cqlsh:Library> COPY Library_Info(Stud_Id,Stud_Name,Book_Name,Book_Id,Date_Of_Issue,Counter_value) TO
   'g:\libraryInfo.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].
cqlshlib.copyutil.ExportProcess.write_rows_to_csv(): writing row
cqlshlib.copyutil.ExportProcess.write_rows_to_csv(): writing row
cqlshlib.copyutil.ExportProcess.write_rows_to_csv(): writing row
Processed: 3 rows; Rate:      6 rows/s; Avg. rate:      6 rows/s
3 rows exported to 1 files in 0.484 seconds.
cqlsh:Library> CREATE TABLE Library_Info_Import( 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));
cqlsh:Library> COPY Library_Info_Import(Stud_Id,Stud_Name,Book_Name,Book_Id,Date_Of_Issue,Counter_value) FROM
'g:\libraryInfo.csv';
Using 1 child processes

Starting copy of Library.library_info_import with columns [stud_id, stud_name, book_name, book_id, date_of_issue,
counter_value].
Processed: 3 rows; Rate:      6 rows/s; Avg. rate:      8 rows/s
3 rows imported from 1 files in 0.365 seconds (0 skipped).
cqlsh:Library> SELECT * FROM Library_Info_Import;

 stud_id | stud_name | book_name | book_id | date_of_issue                | counter_value
---------+-----------+-----------+---------+---------------------------------+---------------
      1 |   Dhruva  |       BDA |     111 | 2021-03-15 00:00:00.000000+0000 |           1
      2 |    Priya  |      OOMD |     112 | 2021-02-12 00:00:00.000000+0000 |           1
    112 |    Aswin  |       BDA |    1123 | 2021-01-18 00:00:00.000000+0000 |           2
```

Lab3

use studentdb switched

to db studentdb

db.createCollection("student_details")

{ "ok" : 1 }

db.student_details.insert({'name':'abc','rollno':1,'age':19,'contactno':9090909090,'email':'abc@l
a b.

com'})

WriteResult({ "nInserted" : 1 })

db.student_details.insert({'name':'mno','rollno':2,'age':20,'contactno':9999900000,'email':'mno
@l

ab.com'})

WriteResult({ "nInserted" : 1 })

db.student_details.insert({'name':'xyz','rollno':3,'age':21,'contactno':9999911111,'email':'xyz@
la

b .com'})

WriteResult({ "nInserted" : 1 })

db.student_details.find({})

{ "_id" : ObjectId("60a88f32ffecf7c8abe76775"), "name" : "abc", "rollno" : 1, "age" : 19,

"contactno" : 9090909090, "email" : "abc@lab.com" }

{ "_id" : ObjectId("60a88f7effecf7c8abe76776"), "name" : "mno", "rollno" : 2, "age" : 20,

"contactno" : 9999900000, "email" : "mno@lab.com" }

{ "_id" : ObjectId("60a88f8fffecf7c8abe76777"), "name" : "xyz", "rollno" : 3, "age" : 21,

"contactno" : 9999911111, "email" : "xyz@lab.com" }

```
db.student_details.update({'rollno':3},{$set:{'email':'update@lab.com'}})
WriteResult({ "nMatched" : 1, "nUpserted" : 0, "nModified" : 1 })
db.student_details.find({'rollno':3})
{ "_id" : ObjectId("60a88f8fffecf7c8abe76777"), "name" : "xyz", "rollno" : 3, "age" : 21,
"contactno" : 9999911111, "email" : "update@lab.com" }

db.student_details.update({'name':'xyz'},{$set:{'name':'pqr'}})
WriteResult({ "nMatched" : 1, "nUpserted" : 0, "nModified" : 1 })
db.student_details.find({'name':'pqr'})
{ "_id" : ObjectId("60a88f8fffecf7c8abe76777"), "name" : "pqr", "rollno" : 3, "age" : 21,
"contactno" : 9999911111, "email" : "update@lab.com" }

mongoexport --db studentdb --collection student_details --out E:\Desktop\sample.json
2021-05-22T10:43:30.687+0530    connected to: mongodb://localhost/
2021-05-22T10:43:31.026+0530    exported 3 records

db.getCollection('student_details').drop()
true

mongoimport --db studentdb --collection student_details --type=json --file=
E:\Desktop\sample.json
2021-05-22T10:46:49.898+0530    connected to: mongodb://localhost/ 2021-05-
22T10:46:50.044+0530    3 document(s) imported successfully. 0 document(s) failed to
import.

db.student_details.find({})
{ "_id" : ObjectId("60a88f8fffecf7c8abe76777"), "name" : "pqr", "rollno" : 3, "age" : 21,
"contactno" : 9999911111, "email" : "update@lab.com" }
{ "_id" : ObjectId("60a88f32ffecf7c8abe76775"), "name" : "abc", "rollno" : 1, "age" : 19,
"contactno" : 9090909090, "email" : "abc@lab.com" }
{ "_id" : ObjectId("60a88f7effecf7c8abe76776"), "name" : "mno", "rollno" : 2, "age" : 20,
"contactno" : 9999900000, "email" : "mno@lab.com" }
```

db.student_details.remove({age:{$gt:20}})

WriteResult({ "nRemoved" : 1 })

db.student_details.find({})

{ "_id" : ObjectId("60a88f32ffecf7c8abe76775"), "name" : "abc", "rollno" : 1, "age" : 19, "contactno" : 9090909090, "email" : "abc@lab.com" }

{ "_id" : ObjectId("60a88f7effecf7c8abe76776"), "name" : "mno", "rollno" : 2, "age" : 20, "contactno" : 9999900000, "email" : "mno@lab.com" }

db.student_details.find({})

{ "_id" : ObjectId("60a88f32ffecf7c8abe76775"), "name" : "abc", "rollno" : 1, "age" : 19, "contactno" : 9090909090, "email" : "abc@lab.com" }

{ "_id" : ObjectId("60a88f7effecf7c8abe76776"), "name" : "mno", "rollno" : 2, "age" : 20, "contactno" : 9999900000, "email" : "mno@lab.com" }

```
> use studentdb
switched to db studentdb
> db.createCollection("student_details")
{ "ok" : 1 }
> db.student_details.insert({'name':'abc','rollno':1,'age':19,'contactno':9090909090,'email':'abc@lab.com'})
WriteResult({ "nInserted" : 1 })
> db.student_details.insert({'name':'mno','rollno':2,'age':20,'contactno':9999900000,'email':'mno@lab.com'})
WriteResult({ "nInserted" : 1 })
> db.student_details.insert({'name':'xyz','rollno':3,'age':21,'contactno':9999911111,'email':'xyz@lab.com'})
WriteResult({ "nInserted" : 1 })
> db.student_details.find({})
{ "_id" : ObjectId("60a88f32ffecf7c8abe76775"), "name" : "abc", "rollno" : 1, "age" : 19, "contactno" : 9090909090, "email" : "abc@lab.com" }
{ "_id" : ObjectId("60a88f7effecf7c8abe76776"), "name" : "mno", "rollno" : 2, "age" : 20, "contactno" : 9999900000, "email" : "mno@lab.com" }
{ "_id" : ObjectId("60a88f8fffecf7c8abe76777"), "name" : "xyz", "rollno" : 3, "age" : 21, "contactno" : 9999911111, "email" : "xyz@lab.com" }
> db.student_details.update({'rollno':3},{$set:{'email':'update@lab.com'}})
WriteResult({ "nMatched" : 1, "nUpserted" : 0, "nModified" : 1 })
> db.student_details.find({'rollno':3})
{ "_id" : ObjectId("60a88f8fffecf7c8abe76777"), "name" : "xyz", "rollno" : 3, "age" : 21, "contactno" : 9999911111, "email" : "update@lab.com" }
> db.student_details.update({'name':'xyz'},{$set:{'name':'pqr'}})
WriteResult({ "nMatched" : 1, "nUpserted" : 0, "nModified" : 1 })
> db.student_details.find({'name':'pqr'})
{ "_id" : ObjectId("60a88f8fffecf7c8abe76777"), "name" : "pqr", "rollno" : 3, "age" : 21, "contactno" : 9999911111, "email" : "update@lab.com" }
```

```
> db.student_details.find({})
{ "_id" : ObjectId("60a88f8fffecf7c8abe76777"), "name" : "pqr", "rollno" : 3, "age" : 21, "contactno" : 9999911111, "email" : "update@lab.com" }
{ "_id" : ObjectId("60a88f32ffecf7c8abe76775"), "name" : "abc", "rollno" : 1, "age" : 19, "contactno" : 9090909090, "email" : "abc@lab.com" }
{ "_id" : ObjectId("60a88f7effecf7c8abe76776"), "name" : "mno", "rollno" : 2, "age" : 20, "contactno" : 9999900000, "email" : "mno@lab.com" }
> db.student_details.remove({age:{$gt:20}})
WriteResult({ "nRemoved" : 1 })
> db.student_details.find({})
{ "_id" : ObjectId("60a88f32ffecf7c8abe76775"), "name" : "abc", "rollno" : 1, "age" : 19, "contactno" : 9090909090, "email" : "abc@lab.com" }
{ "_id" : ObjectId("60a88f7effecf7c8abe76776"), "name" : "mno", "rollno" : 2, "age" : 20, "contactno" : 9999900000, "email" : "mno@lab.com" }
```

LAB4

SCREENSHOT OF HADOOP INSTALLATION

Administrator: Command Prompt

```
Microsoft Windows [Version 10.0.19043.1766]
(c) Microsoft Corporation. All rights reserved.

C:\WINDOWS\system32>start-dfs

C:\WINDOWS\system32>start-yarn
starting yarn daemons

C:\WINDOWS\system32>jps
16520 NameNode
23544 NodeManager
9240 DataNode
11692 Eclipse
16924 Jps
20060 ResourceManager
```

LAB 5

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

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 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
```

```
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 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
```

LAB 6

For the given file, Create a Map Reduce program to
a) Find the average temperature for each year from the NCDC data set.

```
// AverageDriver.java package temperature;

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 AverageDriver
{	public static void main (String[] args) throws Exception
	{
		if (args.length != 2)
		{
			System.err.println("Please Enter the input and output parameters");
			System.exit(-1);
		}
   Job job = new Job();	job.setJarByClass(AverageDriver.class);	job.setJobName("Max
temperature");
		FileInputFormat.addInputPath(job,new Path(args[0]));
		FileOutputFormat.setOutputPath(job,new Path (args[1]));

		job.setMapperClass(AverageMapper.class);
	job.setReducerClass(AverageReducer.class);		job.setOutputKeyClass(Text.class);
	job.setOutputValueClass(IntWritable.class);
	System.exit(job.waitForCompletion(true)?0:1);
	}
}

//AverageMapper.java package temperature;

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

public class AverageMapper 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 year = line.substring(15,19);	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(year),new IntWritable(temperature)); }
}
//AverageReducer.java package temperature;

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

public class AverageReducer extends Reducer <Text, IntWritable,Text, IntWritable>
{
        public void reduce(Text key, Iterable<IntWritable> values, Context context) throws
IOException,InterruptedException
            {
                    int max_temp = 0;                    int count = 0;
                    for (IntWritable value : values)
                    {
                            max_temp += value.get();
                count+=1;
                    }
                    context.write(key, new IntWritable(max_temp/count));
            }
}
```

```
c:\hadoop_new\sbin>hdfs dfs -cat /tempAverageOutput/part-r-00000
1901    46
1949    94
1950    3
```

```
//TempDriver.java 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);
        }
}
//TempMapper.java 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)); }

}
//TempReducer.java 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));
            }
}
```

```
c:\hadoop_new\sbin>hdfs dfs -cat /tempMaxOutput/part-r-00000
01      44
02      17
03      111
04      194
05      256
06      278
07      317
08      283
09      211
10      156
11      89
12      117
```

LAB 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.java package sortWords;

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> {

      @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-value.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;
   }
}
```

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

LAB 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.

```
// 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.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);

                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);
    }
  }
}
```

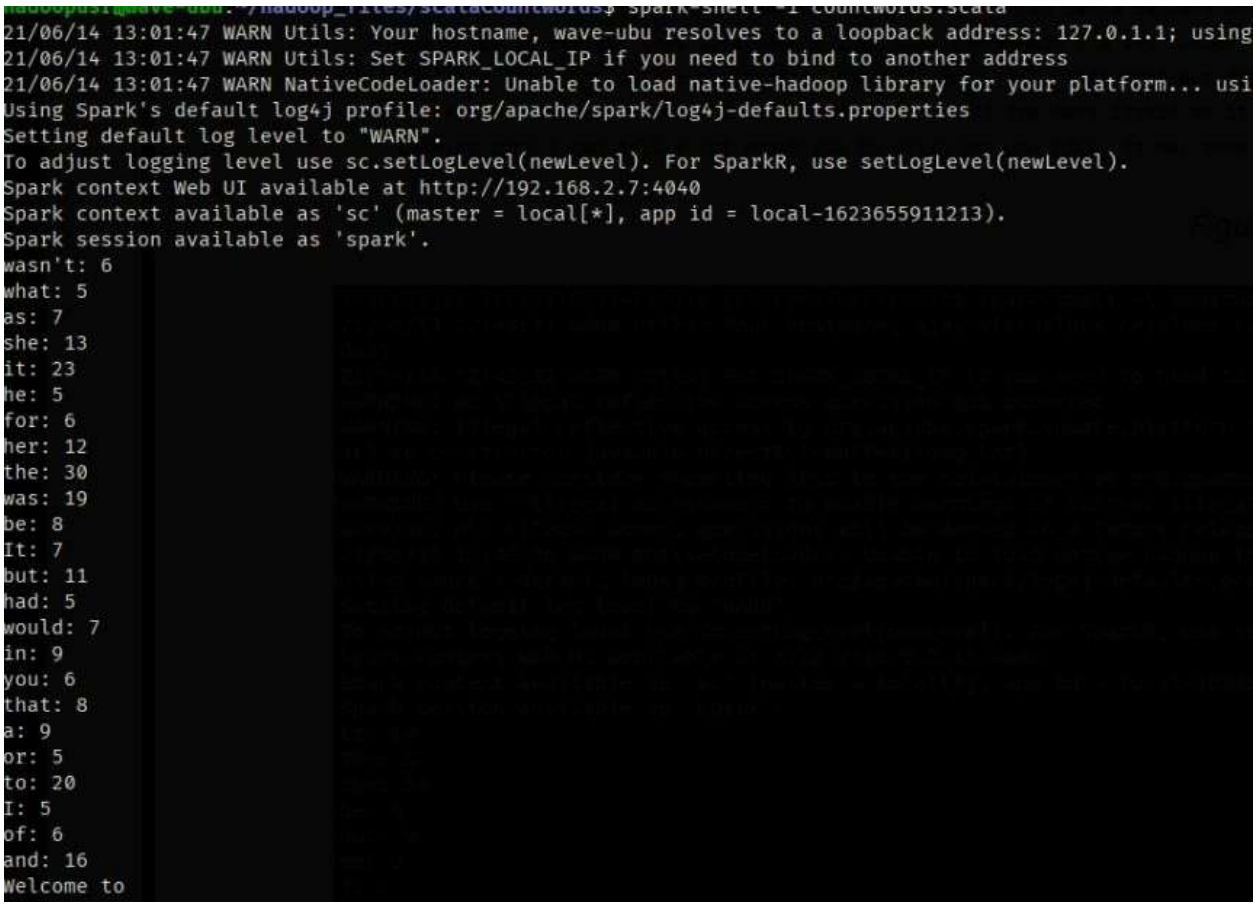
```
c:\hadoop_new\share\hadoop\mapreduce>hdfs dfs -cat \joinOutput\part-00000
"100005361"     "2"             "36134"
"100018705"     "2"             "76"
"100022094"     "0"             "6354"
```

LAB 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;
```



```
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
```

LAB 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

```
scala> val textfile = sc.textFile("/home/sam/Desktop/abc.txt")
textfile: org.apache.spark.rdd.RDD[String] = /home/sam/Desktop/abc.txt MapPartitionsRDD[8] at textFile at <conso
le>:25

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>:26

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] = ListMap(hello -> 3, apple -> 2, unicorn -> 1, world ->
1)

scala> println(sorted)
ListMap(hello -> 3, apple -> 2, unicorn -> 1, world -> 1)
```