VISVESVARAYA TECHNOLOGICAL UNIVERSITY

"JnanaSangama", Belgaum -590014, Karnataka.

LAB REPORT
on

BIG DATA ANALYTICS
(20CS6PEBDA)

Submitted by

PRATHIKSHA KAMATH(1BM19CS118)

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 **PRATHIKSHA KAMATH(1BM19CS118),** 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 Choudhury
Assistant Professor

Department of CSE
BMSCE, Bengaluru

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

`

Index Sheet

Course Outcome

CO 1	Apply the concept of NoSQL, Hadoop or Spark for a given task
CO 2	Analyze the Big Data and obtain insight using data analytics mechanisms.
CO 3	Design and implement Big data applications by applying NoSQL, Hadoop or Spark

LAB PROGRAM 1: MongoDB- CRUD Demonstration

1) Using MongoDB
i) Create a database for Students and Create a Student Collection (_id,Name, USN, Semester, Dept_Name, CGPA, Hobbies(Set)).
use student2;

db.createCollection("Student");

ii) Insert required documents to the collection.
> db.Student.insert({_id:1,Name: "Arun", sem:"V",dept: "CSE",CGPA: 8.2,hobbies: ['cycling','swimming']});
WriteResult({ "nInserted" : 1 })
> db.Student.insert({_id:2,Name: "Ananya", sem:"VII",dept: "ECE",CGPA: 6.8,hobbies: ['knitting','reading novels']});
WriteResult({ "nInserted" : 1 })
> db.Student.insert({_id:3,Name: "Bhuvan", sem:"III",dept: "ME",CGPA: 8.8,hobbies: ['chess','collecting coins']});
WriteResult({ "nInserted" : 1 })
> db.Student.insert({_id:4,Name: "Ajay", sem:"VII",dept: "CSE",CGPA: 9.1,hobbies: ['playing','reading novels']});
WriteResult({ "nInserted" : 1 })
> db.Student.insert({_id:5,Name: "Colin", sem:"V",dept: "CSE",CGPA: 7.1,hobbies: ['playing','watching TV']});
WriteResult({ "nInserted" : 1 })

```
> db.Student.find();
{ "_id" : 1, "Name" : "Arun", "sem" : "V", "dept" : "CSE", "CGPA" : 8.2, "hobbies" : [ "cycling", "swimming" ] }
{ "_id" : 2, "Name" : "Ananya", "sem" : "VII", "dept" : "ECE", "CGPA" : 6.8, "hobbies" : [ "knitting", "reading novels" ] }
{ "_id" : 3, "Name" : "Bhuvan", "sem" : "III", "dept" : "ME", "CGPA" : 8.8, "hobbies" : [ "chess", "collecting coins" ] }
{ "_id" : 4, "Name" : "Ajay", "sem" : "VII", "dept" : "CSE", "CGPA" : 9.1, "hobbies" : [ "playing", "reading novels" ] }
{ "_id" : 5, "Name" : "Colin", "sem" : "V", "dept" : "CSE", "CGPA" : 7.1, "hobbies" : [ "playing", "watching TV" ] }
>
```

iii) First Filter on “Dept_Name:CSE” and then group it on “Semester” and compute the Average CPGA for that semester and filter those documents where the “Avg_CPGA” is greater than 7.5.
>
db.Student.aggregate({$match:{dept:"CSE"}},{$group:{_id:"$sem",AverageCGPA:{$avg:"$CGPA"}}},{$match:{AverageCGPA:{$gt:7.5}}});
{ "_id" : "VII", "AverageCGPA" : 9.1 }
{ "_id" : "V", "AverageCGPA" : 7.6499999999999995 }

```
> db.Student.aggregate({$match:{dept:"CSE"}},{$group:{_id:"$sem",AverageCGPA:{$avg:"$CGPA"}}},{$match:{AverageCGPA:{$gt:7.5}}});
{ "_id" : "V", "AverageCGPA" : 7.6499999999999995 }
{ "_id" : "VII", "AverageCGPA" : 9.1 }
```

iv) Insert the document for “Bhuvan” 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 to “Skating”) 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,Name:"Bhuvan"},{$set:{ Hobbies:"Skating"}},{upsert:true}
);
WriteResult({ "nMatched" : 1, "nUpserted" : 0, "nModified" : 1 })
```

v)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({},{name:1,sem: 1, _id:0});
{ "sem" : "V" }
{ "sem" : "VII" }
{ "sem" : "III" }
{ "sem" : "VII" }
{ "sem" : "V" }
```

vi) To find those documents where the Grade is set to ‘VII’

```
> db.Student.find({sem:{$eq:"VII"}});
{ "_id" : 2, "Name" : "Ananya", "sem" : "VII", "dept" : "ECE", "CGPA" : 6.8, "hobbies" : [ "knitting", "reading novels" ] }
{ "_id" : 4, "Name" : "Ajay", "sem" : "VII", "dept" : "CSE", "CGPA" : 9.1, "hobbies" : [ "playing", "reading novels" ] }
```

vii)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']}});
{ "_id" : 3, "Name" : "Bhuvan", "sem" : "III", "dept" : "ME", "CGPA" : 8.8, "hobbies" : [ "chess", "collecting coins" ], "Hobbies" : "Skating" }
```

viii)To find documents from the Students collection where the StudName begins with “B”

```
> db.Student.find({Name: /^B/});
{ "_id" : 3, "Name" : "Bhuvan", "sem" : "III", "dept" : "ME", "CGPA" : 8.8, "hobbies" : [ "chess", "collecting coins" ], "Hobbies" : "Skating" }
```

ix) To find the number of documents in the Students collection.

```
> db.Student.count();
5
```

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

```
> db.Student.find().sort({Name: -1});
{ "_id" : 5, "Name" : "Colin", "sem" : "V", "dept" : "CSE", "CGPA" : 7.1, "hobbies" : [ "playing", "watching TV" ] }
{ "_id" : 3, "Name" : "Bhuvan", "sem" : "III", "dept" : "ME", "CGPA" : 8.8, "hobbies" : [ "chess", "collecting coins" ], "Hobbies" : "Skating" }
{ "_id" : 1, "Name" : "Arun", "sem" : "V", "dept" : "CSE", "CGPA" : 8.2, "hobbies" : [ "cycling", "swimming" ] }
{ "_id" : 2, "Name" : "Ananya", "sem" : "VII", "dept" : "ECE", "CGPA" : 6.8, "hobbies" : [ "knitting", "reading novels" ] }
{ "_id" : 4, "Name" : "Ajay", "sem" : "VII", "dept" : "CSE", "CGPA" : 9.1, "hobbies" : [ "playing", "reading novels" ] }
```

xi) Command used to export MongoDB JSON documents from “Student” Collection into the “Students” database into a CSV file “Output.txt”.

```
> mongoexport --host localhost --db studentDB --collection Student --csv --out /Downloads/student.txt -fields "Name","sem"
uncaught exception: SyntaxError: unexpected token: identifier :
@(shell):1:14
```

LAB PROGRAM 2: Employee database using Cassandra

Program 1. Perform the following DB operations using Cassandra.

```
bmsce@bmsce-Precision-T1700:~/cassandra/apache-cassandra-3.11.0/bin$ cqlsh
Connected to Test Cluster at 127.0.0.1:9042.
[cqlsh 5.0.1 | Cassandra 3.11.4 | CQL spec 3.4.4 | Native protocol v4]
Use HELP for help.
```

1. Create a key space by name Employee

```
cqlsh> create keyspace Employee with REPLICATION ={
   ... 'class':'SimpleStrategy','replication_factor':1
   ... };

cqlsh> use Employee;
cqlsh:employee> describe keyspaces;

students      system_auth  system_distributed  system_traces
system_schema  system      employee
```

```
cqlsh> describe keyspace employee;

CREATE KEYSPACE employee WITH replication = {'class': 'SimpleStrategy', 'replication_factor': '1'}  AND durable_writes = true;
```

2. Create a column family by name Employee-Info with attributes Emp_Id Primary Key, Emp_Name, Designation, Date_of_Joining, Salary, Dept_Name

```
cqlsh:employee> CREATE TABLE Employee_Info(
          ... emp_id int PRIMARY KEY,
          ... emp_name text,
          ... designation text,
          ... date_of_joining timestamp,
          ... salary double,
          ... dept_name text
          ... );
cqlsh:employee> describe tables

employee_info
```

```
cqlsh:employee> describe table 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 double
) WITH 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';
```

3. Insert the values into the table in batch

```
cqlsh:employee> BEGIN BATCH
        ... insert into employee_info(emp_id,emp_name,designation,date_of_joining,salary,dept_name)
        ... values(1,'Arun','Technical head','2020-03-01',50000,'Technical')
        ... insert into employee_info(emp_id,emp_name,designation,date_of_joining,salary,dept_name)
        ... values(2,'Ajay','HR manager','2020-06-11',60000,'HR')
        ... insert into employee_info(emp_id,emp_name,designation,date_of_joining,salary,dept_name)
        ... values(3,'Riya','Editor','2022-01-11',22000,'Markrting')
        ... insert into employee_info(emp_id,emp_name,designation,date_of_joining,salary,dept_name)
        ... values(4,'Kshma','Software Engineer','2021-05-11',35000,'Technical')
        ... insert into employee_info(emp_id,emp_name,designation,date_of_joining,salary,dept_name)
        ... values(5,'Ram','HR employee','2021-02-11',25000,'HR')
        ... APPLY BATCH;

cqlsh:employee> select * from employee_info;


 emp_id | date_of_joining                 | dept_name | designation       | emp_name | salary
--------+---------------------------------+-----------+-------------------+----------+--------
      5 | 2021-02-10 18:30:00.000000+0000 |        HR |       HR employee |      Ram |  25000
      1 | 2020-02-29 18:30:00.000000+0000 | Technical |    Technical head |     Arun |  50000
      2 | 2020-06-10 18:30:00.000000+0000 |        HR |        HR manager |     Ajay |  60000
      4 | 2021-05-10 18:30:00.000000+0000 | Technical | Software Engineer |    Kshma |  35000
      3 | 2022-01-10 18:30:00.000000+0000 | Markrting |            Editor |     Riya |  22000
```

```
cqlsh:employee> select * from employee_info
           ... ;

 emp_id | date_of_joining                 | dept_name | designation       | emp_name | salary
--------+---------------------------------+-----------+-------------------+----------+--------
      5 | 2021-02-10 18:30:00.000000+0000 |        HR |       HR employee |      Ram |  25000
      1 | 2020-02-29 18:30:00.000000+0000 | Technical |    Technical head |     Arun |  50000
      2 | 2020-06-10 18:30:00.000000+0000 |        HR |        HR manager |     Ajay |  60000
      4 | 2021-05-10 18:30:00.000000+0000 | Technical | Software Engineer |    Kshma |  35000
      3 | 2022-01-10 18:30:00.000000+0000 | Markrting |            Editor |     Riya |  22000

(5 rows)
```

4. Update Employee name and Department of Emp-Id 3

cqlsh:employee> UPDATE employee_info SET emp_name = 'Raj' , dept_name = 'Sales' where emp_id = 3;

cqlsh:employee> select * from employee_info;

```
 emp_id | date_of_joining                 | dept_name | designation       | emp_name | salary
--------+---------------------------------+-----------+-------------------+----------+--------
      5 | 2021-02-10 18:30:00.000000+0000 |        HR |       HR employee |      Ram |  25000
      1 | 2020-02-29 18:30:00.000000+0000 | Technical |    Technical head |     Arun |  50000
      2 | 2020-06-10 18:30:00.000000+0000 |        HR |        HR manager |     Ajay |  60000
      4 | 2021-05-10 18:30:00.000000+0000 | Technical | Software Engineer |    Kshma |  35000
      3 | 2022-01-10 18:30:00.000000+0000 |     Sales |            Editor |      Raj |  22000
```

```
cqlsh:employee> select * from employee_info;

 emp_id | date_of_joining                 | dept_name | designation       | emp_name | salary
--------+---------------------------------+-----------+-------------------+----------+--------
      5 | 2021-02-10 18:30:00.000000+0000 |        HR |       HR employee |      Ram |  25000
      1 | 2020-02-29 18:30:00.000000+0000 | Technical |    Technical head |     Arun |  50000
      2 | 2020-06-10 18:30:00.000000+0000 |        HR |        HR manager |     Ajay |  60000
      4 | 2021-05-10 18:30:00.000000+0000 | Technical | Software Engineer |    Kshma |  35000
      3 | 2022-01-10 18:30:00.000000+0000 |     Sales |            Editor |      Raj |  22000

(5 rows)
cqlsh:employee>
```

5. Sort the details of Employee records based on salary

```
CREATE TABLE emp(
        ...  emp_id int,
        ... salary double,
        ... emp_name text,
        ... PRIMARY KEY(emp_id,salary));
```

```
BEGIN BATCH
... insert into emp(emp_id,emp_name,salary) values(1,'Prema',25000)
... insert into emp(emp_id,emp_name,salary) values(2,'Pooja',35000)
... insert into emp(emp_id,emp_name,salary) values(3,'Arun',25000)
... insert into emp(emp_id,emp_name,salary) values(4,'Ajay',50000)
... insert into emp(emp_id,emp_name,salary) values(5,'Bob',100000)
... APPLY BATCH;
```

PAGING OFF;
select * from emp where emp_id in(1,2,3,4,5) order by salary;

```
 emp_id | salary | emp_name
--------+--------+----------
      1 |  25000 |    Prema
      3 |  25000 |     Arun
      2 |  35000 |    Pooja
      4 |  50000 |     Ajay
      5 |  1e+05 |      Bob
```

```
cqlsh:employee> paging off;
Disabled Query paging.
cqlsh:employee> select * from emp where emp_id in (1,2,3,4,5) order by salary;

 emp_id | salary | emp_name
--------+--------+----------
      1 |  25000 |    Prema
      3 |  25000 |     Arun
      2 |  35000 |    Pooja
      4 |  50000 |     Ajay
      5 |  1e+05 |      Bob

(5 rows)
cqlsh:employee>
```

6. Alter the schema of the table Employee_Info to add a column Projects which stores a set of Projects done by the corresponding Employee.

```
cqlsh:employee> alter table employee_info
          ... add project text;
cqlsh:employee> select * from employee_info;
```

```
 emp_id | date_of_joining                | dept_name | designation       | emp_name | project | salary
--------+--------------------------------+-----------+-------------------+----------+---------+--------
      5 | 2021-02-10 18:30:00.000000+0000 |        HR |       HR employee |      Ram |    null |  25000
      1 | 2020-02-29 18:30:00.000000+0000 | Technical |    Technical head |     Arun |    null |  50000
      2 | 2020-06-10 18:30:00.000000+0000 |        HR |        HR manager |     Ajay |    null |  60000
```

```
     4 | 2021-05-10 18:30:00.000000+0000 | Technical | Software Engineer |    Kshma |    null |  35000
     3 | 2022-01-10 18:30:00.000000+0000 |     Sales |           Editor |      Raj |    null |  22000

(5 rows)
```

7. Update the altered table to add project names.

```
cqlsh:employee> begin batch
          ... update employee_info set project = 'xyz' where emp_id = 3
          ... update employee_info set project = 'pqr' where emp_id = 5
          ... update employee_info set project = 'pqr' where emp_id = 2
          ... update employee_info set project = 'abc' where emp_id = 1
          ... update employee_info set project = 'abc' where emp_id = 4
          ... apply batch;
cqlsh:employee> select * from employee_info;

 emp_id | date_of_joining                 | dept_name | designation       | emp_name | project | salary
--------+---------------------------------+-----------+-------------------+----------+---------+--------
     5 | 2021-02-10 18:30:00.000000+0000 |        HR |       HR employee |      Ram |     pqr |  25000
     1 | 2020-02-29 18:30:00.000000+0000 | Technical |    Technical head |     Arun |     abc |  50000
     2 | 2020-06-10 18:30:00.000000+0000 |        HR |        HR manager |     Ajay |     pqr |  60000
     4 | 2021-05-10 18:30:00.000000+0000 | Technical | Software Engineer |    Kshma |     abc |  35000
     3 | 2022-01-10 18:30:00.000000+0000 |     Sales |           Editor |      Raj |     xyz |  22000

(5 rows)
```

```
cqlsh:employee> select * from employee_info;

 emp_id | date_of_joining                 | dept_name | designation       | emp_name | project | salary
--------+---------------------------------+-----------+-------------------+----------+---------+--------
      5 | 2021-02-10 18:30:00.000000+0000 |        HR |       HR employee |      Ram |     pqr |  25000
      1 | 2020-02-29 18:30:00.000000+0000 | Technical |    Technical head |     Arun |     abc |  50000
      2 | 2020-06-10 18:30:00.000000+0000 |        HR |        HR manager |     Ajay |     pqr |  60000
      4 | 2021-05-10 18:30:00.000000+0000 | Technical | Software Engineer |    Kshma |     abc |  35000
      3 | 2022-01-10 18:30:00.000000+0000 |     Sales |           Editor |      Raj |     xyz |  22000

(5 rows)
cqlsh:employee>
```

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

```
cqlsh:employee> insert into employee_info(emp_id,
date_of_joining,dept_name,designation,emp_name,project,salary)  values(6, '2021-02-28','HR','HR
employee','Anvi','xyz',20000) using TTL 15;
cqlsh:employee> select TTL(emp_name) from employee_info;

 ttl(emp_name)
---------------
```

null
null
null
null
5
null

(6 rows)

```
cqlsh:employee> select TTL(emp_name) from employee_info;

 ttl(emp_name)
---------------
          null
          null
          null
          null
             5
          null

(6 rows)
```

LAB PROGRAM 3: Library database using Cassandra

1 Create a key space by name Library
create keyspace library with replication={
... 'class':'SimpleStrategy','replication_factor':1
... };

```
cqlsh> describe keyspace library;

CREATE KEYSPACE library WITH replication = {'class': 'SimpleStrategy', 'replication_factor': '1'}  AND durable_writes = true;
```

use library;

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(
... 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> 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';
```

3. Insert the values into the table in batch
cqlsh:library> update library_info set counter_value=counter_value+1 where stud_id=1 and stud_name = 'Raj' and book_name='BDA' and book_id=200 and date_of_issue='2022-04-30';
cqlsh:library> update library_info set counter_value=counter_value+1 where stud_id=2 and stud_name = 'Ravi' and book_name='ADA' and book_id=100 and date_of_issue='2022-04-30';

cqlsh:library> update library_info set counter_value=counter_value+1 where stud_id=1 and stud_name = 'Raj' and book_name='BDA' and book_id=200 and date_of_issue='2022-05-30';
cqlsh:library> select * from library_info;

```
cqlsh:library> select * from library_info;

 stud_id | stud_name | book_name | book_id | date_of_issue                   | counter_value
---------+-----------+-----------+---------+---------------------------------+---------------
       1 |       Raj |       BDA |     200 | 2022-04-29 18:30:00.000000+0000 |             1
       1 |       Raj |       BDA |     200 | 2022-05-29 18:30:00.000000+0000 |             1
       2 |      Ravi |       ADA |     100 | 2022-04-29 18:30:00.000000+0000 |             1

(3 rows)
```

4. Display the details of the table created and increase the value of the counter
cqlsh:library> update library_info set counter_value=counter_value+1 where stud_id=1 and stud_name = 'Raj' and book_name='BDA' and book_id=200 and date_of_issue='2022-04-30';
cqlsh:library> select * from library_info;

```
 stud_id | stud_name | book_name | book_id | date_of_issue                   | counter_value
---------+-----------+-----------+---------+---------------------------------+---------------
       1 |       Raj |       BDA |     200 | 2022-04-29 18:30:00.000000+0000 |             2
       1 |       Raj |       BDA |     200 | 2022-05-29 18:30:00.000000+0000 |             1
       2 |      Ravi |       ADA |     100 | 2022-04-29 18:30:00.000000+0000 |             1
```

```
cqlsh:library> select * from library_info;

 stud_id | stud_name | book_name | book_id | date_of_issue                   | counter_value
---------+-----------+-----------+---------+---------------------------------+---------------
       1 |       Raj |       BDA |     200 | 2022-04-29 18:30:00.000000+0000 |             2
       1 |       Raj |       BDA |     200 | 2022-05-29 18:30:00.000000+0000 |             1
       2 |      Ravi |       ADA |     100 | 2022-04-29 18:30:00.000000+0000 |             1

(3 rows)
```

5. Write a query to show that a student with id 1 has taken a book “BDA” 2 times.
cqlsh:library> select counter_value from library_info where stud_id = 1;

```
 counter_value
---------------
             2
             1
```

```
cqlsh:library> select counter_value from library_info where stud_id = 1;

 counter_value
---------------
             2
             1

(2 rows)
```

6. Export the created column to a csv file

```
cqlsh:lab2_library> copy library_info(stud_id,stud_name,book_id,date_of_issue,counter_value)to 'lib.csv';
Using 7 child processes

Starting copy of lab2_library.library_info with columns [stud_id, stud_name, book_id, date_of_issue, counter_v
alue].
Processed: 2 rows; Rate:       9 rows/s; Avg. rate:       9 rows/s
2 rows exported to 1 files in 0.250 seconds.
```

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

cqlsh:library>truncate library_info;

cqlsh:library>copy library_info(stud_id,stud_name,book_id,date_of_issue,counter_value) from 'lib.csv';

LAB PROGRAM 4: Screenshot of Hadoop installed

```
prathiksha@PRATHIKSHA:~/hadoop/hadoop-3.3.0$ ssh localhost
Welcome to Ubuntu 20.04 LTS (GNU/Linux 5.10.16.3-microsoft-standard-WSL2 x86_64)

 * Documentation:  https://help.ubuntu.com
 * Management:     https://landscape.canonical.com
 * Support:        https://ubuntu.com/advantage

  System information as of Tue Jul 12 08:11:43 IST 2022

  System load:  0.02               Processes:             11
  Usage of /:   1.1% of 250.98GB   Users logged in:       0
  Memory usage: 4%                 IPv4 address for eth0: 172.18.170.77
  Swap usage:   0%


290 updates can be installed immediately.
176 of these updates are security updates.
To see these additional updates run: apt list --upgradable


The list of available updates is more than a week old.
To check for new updates run: sudo apt update

Last login: Tue Jun 21 13:20:28 2022 from 127.0.0.1
prathiksha@PRATHIKSHA:~$ sbin/start-dfs.sh
-bash: sbin/start-dfs.sh: No such file or directory
prathiksha@PRATHIKSHA:~$ cd ~/hadoop/hadoop-3.3.0/
prathiksha@PRATHIKSHA:~/hadoop/hadoop-3.3.0$ sbin/start-dfs.sh

Starting namenodes on [localhost]
Starting datanodes
Starting secondary namenodes [PRATHIKSHA]
prathiksha@PRATHIKSHA:~/hadoop/hadoop-3.3.0$
prathiksha@PRATHIKSHA:~/hadoop/hadoop-3.3.0$ jps
545 DataNode
818 SecondaryNameNode
996 Jps
378 NameNode
prathiksha@PRATHIKSHA:~/hadoop/hadoop-3.3.0$
```

LAB PROGRAM 5: Execution of HDFS Commands for interaction with Hadoop Environment.

```
bmsce@bmsce-Precision-T1700:~$ sudo su hduser
[sudo] password for bmsce:
hduser@bmsce-Precision-T1700:/home/bmsce$ start-all.sh
This script is Deprecated. Instead use start-dfs.sh and start-yarn.sh
Starting namenodes on [localhost]
hduser@localhost's password:
localhost: namenode running as process 6691. Stop it first.
hduser@localhost's password:
localhost: datanode running as process 6951. Stop it first.
Starting secondary namenodes [0.0.0.0]
hduser@0.0.0.0's password:
0.0.0.0: secondarynamenode running as process 7329. Stop it first.
starting yarn daemons
resourcemanager running as process 7490. Stop it first.
hduser@localhost's password:
localhost: nodemanager running as process 8817. Stop it first.
hduser@bmsce-Precision-T1700:/home/bmsce$ jps
7329 SecondaryNameNode
8817 NodeManager
7490 ResourceManager
6691 NameNode
6951 DataNode
10188 Jps

hduser@bmsce-Precision-T1700:~$ hdfs dfs -mkdir prathiksha
hduser@bmsce-Precision-T1700:/home/bmsce$ hdfs dfs -ls /
Found 3 items
drwxr-xr-x   - hduser supergroup          0 2022-05-31 09:42 /prathiksha
drwxrwxr-x   - hduser supergroup          0 2019-08-01 16:19 /tmp
drwxr-xr-x   - hduser supergroup          0 2019-08-01 16:03 /user

hduser@bmsce-Precision-T1700:/$ cd ~/Desktop
hduser@bmsce-Precision-T1700:~/Desktop$ vi abc.txt
hduser@bmsce-Precision-T1700:~/Desktop$ cd ..
hduser@bmsce-Precision-T1700:~$ hdfs dfs -put ~/Desktop/abc.txt /prathiksha/first.txt
hduser@bmsce-Precision-T1700:~$ hdfs dfs -ls /prathiksha
Found 1 items
-rw-r--r--   1 hduser supergroup         13 2022-05-31 10:01 /prathiksha/first.txt


hduser@bmsce-Precision-T1700:~$ vi ~/Desktop/welcome.txt
```

```
hduser@bmsce-Precision-T1700:~$ hdfs dfs -copyFromLocal ~/Desktop/welcome.txt
/prathiksha/welcome.txt
hduser@bmsce-Precision-T1700:~$ hdfs dfs -ls /prathiksha
Found 2 items
-rw-r--r--   1 hduser supergroup         13 2022-05-31 10:01 /prathiksha/first.txt
-rw-r--r--   1 hduser supergroup         24 2022-05-31 10:06 /prathiksha/welcome.txt


duser@bmsce-Precision-T1700:~$ hdfs dfs -get /prathiksha/welcome.txt ~/Downloads/first.txt
hduser@bmsce-Precision-T1700:~$ cat ~/Downloads/first.txt
hi hello how you doing?

hduser@bmsce-Precision-T1700:~$ hdfs dfs -copyToLocal /prathiksha/first.txt ~/Downloads/123.txt
hduser@bmsce-Precision-T1700:~$ cat ~/Downloads/123.txt
abc def ghi
hduser@bmsce-Precision-T1700:~$ hdfs dfs -cat /prathiksha/first.txt
abc def ghi

hduser@bmsce-Precision-T1700:~$ hdfs dfs -mkdir /ABC
hduser@bmsce-Precision-T1700:~$ hdfs dfs -cp /prathiksha /ABC
hduser@bmsce-Precision-T1700:~$ hdfs dfs -ls /ABC
Found 1 items
drwxr-xr-x   - hduser supergroup          0 2022-05-31 10:16 /ABC/prathiksha
hduser@bmsce-Precision-T1700:~$ hdfs dfs -cp /prathiksha /DEF
hduser@bmsce-Precision-T1700:~$ hdfs dfs -ls /DEF
Found 2 items
-rw-r--r--   1 hduser supergroup         13 2022-05-31 10:17 /DEF/first.txt
-rw-r--r--   1 hduser supergroup         24 2022-05-31 10:17 /DEF/welcome.txt

hduser@bmsce-Precision-T1700:~$ hdfs dfs -mv /prathiksha /GHI
hduser@bmsce-Precision-T1700:~$ hdfs dfs -ls /
Found 5 items
drwxr-xr-x   - hduser supergroup          0 2022-05-31 10:16 /ABC
drwxr-xr-x   - hduser supergroup          0 2022-05-31 10:17 /DEF
drwxr-xr-x   - hduser supergroup          0 2022-05-31 10:06 /GHI
drwxrwxr-x   - hduser supergroup          0 2019-08-01 16:19 /tmp
drwxr-xr-x   - hduser supergroup          0 2019-08-01 16:03 /user
```

LAB PROGRAM 6:

From the following link extract the weather data
https://github.com/tomwhite/hadoop-book/tree/master/input/ncdc/all
Create a Map Reduce program to
a) find average temperature for each year from NCDC data set.
b) find the mean max temperature for every month

```
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);
  }
}
```

AverageMapper

```
package temp;

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

public class AverageMapper extends Mapper<LongWritable, Text, Text, IntWritable> {
  public static final int MISSING = 9999;

  public void map(LongWritable key, Text value, Mapper<LongWritable, Text, Text,
IntWritable>.Context context) throws IOException, InterruptedException {
```

```
    int temperature;
    String line = value.toString();
    String year = line.substring(15, 19);
    if (line.charAt(87) == '+') {
      temperature = Integer.parseInt(line.substring(88, 92));
    } else {
      temperature = Integer.parseInt(line.substring(87, 92));
    }
    String quality = line.substring(92, 93);
    if (temperature != 9999 && quality.matches("[01459]"))
      context.write(new Text(year), new IntWritable(temperature));
  }
}
```

AverageReducer

```
package temp;

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

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

SCREENSHOTS:

```
C:\hadoop-3.3.0\sbin>hadoop jar C:\avgtemp.jar temp.AverageDriver /input_dir/temp.txt /avgtemp_outputdir
2021-05-15 14:52:50,635 INFO client.DefaultNoHARMFailoverProxyProvider: Connecting to ResourceManager at /0.0.0.0:8032
2021-05-15 14:52:51,005 WARN mapreduce.JobResourceUploader: Hadoop command-line option parsing not performed. Implement the Tool interface and execute your application with ToolRunner to remedy this.
2021-05-15 14:52:51,111 INFO mapreduce.JobResourceUploader: Disabling Erasure Coding for path: /tmp/hadoop-yarn/staging/Anusree/.staging/job_1621060230696_0005
2021-05-15 14:52:51,735 INFO input.FileInputFormat: Total input files to process : 1
2021-05-15 14:52:52,751 INFO mapreduce.JobSubmitter: number of splits:1
2021-05-15 14:52:53,073 INFO mapreduce.JobSubmitter: Submitting tokens for job: job_1621060230696_0005
2021-05-15 14:52:53,073 INFO mapreduce.JobSubmitter: Executing with tokens: []
2021-05-15 14:52:53,237 INFO conf.Configuration: resource-types.xml not found
2021-05-15 14:52:53,238 INFO resource.ResourceUtils: Unable to find 'resource-types.xml'.
2021-05-15 14:52:53,312 INFO impl.YarnClientImpl: Submitted application application_1621060230696_0005
2021-05-15 14:52:53,352 INFO mapreduce.Job: The url to track the job: http://LAPTOP-JG329ESD:8088/proxy/application_1621060230696_0005/
2021-05-15 14:52:53,353 INFO mapreduce.Job: Running job: job_1621060230696_0005
2021-05-15 14:53:06,640 INFO mapreduce.Job: Job job_1621060230696_0005 running in uber mode : false
2021-05-15 14:53:06,643 INFO mapreduce.Job:  map 0% reduce 0%
2021-05-15 14:53:12,758 INFO mapreduce.Job:  map 100% reduce 0%
2021-05-15 14:53:19,860 INFO mapreduce.Job:  map 100% reduce 100%
2021-05-15 14:53:25,967 INFO mapreduce.Job: Job job_1621060230696_0005 completed successfully
2021-05-15 14:53:26,096 INFO mapreduce.Job: Counters: 54
        File System Counters
                FILE: Number of bytes read=72210
                FILE: Number of bytes written=674341
                FILE: Number of read operations=0
                FILE: Number of large read operations=0
                FILE: Number of write operations=0
                HDFS: Number of bytes read=894860
                HDFS: Number of bytes written=8
                HDFS: Number of read operations=8
                HDFS: Number of large read operations=0
                HDFS: Number of write operations=2
                HDFS: Number of bytes read erasure-coded=0
        Job Counters
                Launched map tasks=1
                Launched reduce tasks=1
                Data-local map tasks=1
                Total time spent by all maps in occupied slots (ms)=3782
```

```
C:\hadoop-3.3.0\sbin>hdfs dfs -ls /avgtemp_outputdir
Found 2 items
-rw-r--r--   1 Anusree supergroup          0 2021-05-15 14:53 /avgtemp_outputdir/_SUCCESS
-rw-r--r--   1 Anusree supergroup          8 2021-05-15 14:53 /avgtemp_outputdir/part-r-00000

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

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

b) find the mean max temperature for every month

MeanMax
MeanMaxDriver.class

```
package meanmax;

import org.apache.hadoop.fs.Path;
import org.apache.hadoop.io.IntWritable;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapreduce.Job;
import org.apache.hadoop.mapreduce.lib.input.FileInputFormat;
import org.apache.hadoop.mapreduce.lib.output.FileOutputFormat;

public class MeanMaxDriver {
  public static void main(String[] args) throws Exception {
    if (args.length != 2) {
      System.err.println("Please Enter the input and output parameters");
      System.exit(-1);
    }
    Job job = new Job();
    job.setJarByClass(MeanMaxDriver.class);
    job.setJobName("Max temperature");
    FileInputFormat.addInputPath(job, new Path(args[0]));
    FileOutputFormat.setOutputPath(job, new Path(args[1]));
    job.setMapperClass(MeanMaxMapper.class);
    job.setReducerClass(MeanMaxReducer.class);
    job.setOutputKeyClass(Text.class);
    job.setOutputValueClass(IntWritable.class);
    System.exit(job.waitForCompletion(true) ? 0 : 1);
  }
}
```

MeanMaxMapper.class

```
package meanmax;

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

public class MeanMaxMapper extends Mapper<LongWritable, Text, Text, IntWritable> {
```

```
  public static final int MISSING = 9999;

  public void map(LongWritable key, Text value, Mapper<LongWritable, Text, Text,
IntWritable>.Context context) throws IOException, InterruptedException {
    int temperature;
    String line = value.toString();
    String month = line.substring(19, 21);
    if (line.charAt(87) == '+') {
      temperature = Integer.parseInt(line.substring(88, 92));
    } else {
      temperature = Integer.parseInt(line.substring(87, 92));
    }
    String quality = line.substring(92, 93);
    if (temperature != 9999 && quality.matches("[01459]"))
      context.write(new Text(month), new IntWritable(temperature));
  }
}
```

MeanMaxReducer.class

```
package meanmax;

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

public class MeanMaxReducer extends Reducer<Text, IntWritable, Text, IntWritable> {
  public void reduce(Text key, Iterable<IntWritable> values, Reducer<Text, IntWritable, Text,
IntWritable>.Context context) throws IOException, InterruptedException {
    int max_temp = 0;
    int total_temp = 0;
    int count = 0;
    int days = 0;
    for (IntWritable value : values) {
      int temp = value.get();
      if (temp > max_temp)
        max_temp = temp;
      count++;
      if (count == 3) {
        total_temp += max_temp;
        max_temp = 0;
        count = 0;
        days++;
      }
    }
    context.write(key, new IntWritable(total_temp / days));
  }
}
```

```
C:\hadoop-3.3.0\sbin>hadoop jar C:\meanmax.jar meanmax.MeanMaxDriver /input_dir/temp.txt /meanmax_output
2021-05-21 20:28:05,250 INFO client.DefaultNoHARMFailoverProxyProvider: Connecting to ResourceManager at /0.0.0.0:8032
2021-05-21 20:28:06,662 WARN mapreduce.JobResourceUploader: Hadoop command-line option parsing not performed. Implement the Tool interface and execute your application with ToolRunner to remedy this.
2021-05-21 20:28:06,916 INFO mapreduce.JobResourceUploader: Disabling Erasure Coding for path: /tmp/hadoop-yarn/staging/Anusree/.staging/job_1621608943095_0001
2021-05-21 20:28:08,426 INFO input.FileInputFormat: Total input files to process : 1
2021-05-21 20:28:09,107 INFO mapreduce.JobSubmitter: number of splits:1
2021-05-21 20:28:09,741 INFO mapreduce.JobSubmitter: Submitting tokens for job: job_1621608943095_0001
2021-05-21 20:28:09,741 INFO mapreduce.JobSubmitter: Executing with tokens: []
2021-05-21 20:28:10,029 INFO conf.Configuration: resource-types.xml not found
2021-05-21 20:28:10,030 INFO resource.ResourceUtils: Unable to find 'resource-types.xml'.
2021-05-21 20:28:10,676 INFO impl.YarnClientImpl: Submitted application application_1621608943095_0001
2021-05-21 20:28:11,005 INFO mapreduce.Job: The url to track the job: http://LAPTOP-JG329ESD:8088/proxy/application_1621608943095_0001/
2021-05-21 20:28:11,006 INFO mapreduce.Job: Running job: job_1621608943095_0001
2021-05-21 20:28:29,385 INFO mapreduce.Job: Job job_1621608943095_0001 running in uber mode : false
2021-05-21 20:28:29,389 INFO mapreduce.Job:  map 0% reduce 0%
2021-05-21 20:28:40,664 INFO mapreduce.Job:  map 100% reduce 0%
2021-05-21 20:28:50,832 INFO mapreduce.Job:  map 100% reduce 100%
2021-05-21 20:28:58,965 INFO mapreduce.Job: Job job_1621608943095_0001 completed successfully
2021-05-21 20:28:59,178 INFO mapreduce.Job: Counters: 54
        File System Counters
                FILE: Number of bytes read=59082
                FILE: Number of bytes written=648091
                FILE: Number of read operations=0
                FILE: Number of large read operations=0
                FILE: Number of write operations=0
                HDFS: Number of bytes read=894860
                HDFS: Number of bytes written=74
                HDFS: Number of read operations=8
                HDFS: Number of large read operations=0
                HDFS: Number of write operations=2
                HDFS: Number of bytes read erasure-coded=0
        Job Counters
                Launched map tasks=1
                Launched reduce tasks=1
                Data-local map tasks=1
                Total time spent by all maps in occupied slots (ms)=8077
                Total time spent by all reduces in occupied slots (ms)=7511
                Total time spent by all map tasks (ms)=8077
                Total time spent by all reduce tasks (ms)=7511
                Total vcore-milliseconds taken by all map tasks=8077
                Total vcore-milliseconds taken by all reduce tasks=7511
                Total megabyte-milliseconds taken by all map tasks=8270848
                Total megabyte-milliseconds taken by all reduce tasks=7691264
```

```
C:\hadoop-3.3.0\sbin>hdfs dfs -cat /meanmax_output/*
01      4
02      0
03      7
04      44
05      100
06      168
07      219
08      198
09      141
10      100
11      19
12      3

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

LAB PROGRAM 7:

For a given Text file, Create a Map Reduce program to sort the content in an alphabetic order listing only top 10 maximum occurrences of words.

Driver-TopN.class

```
package samples.topn;

import java.io.IOException;
import java.util.StringTokenizer;
import org.apache.hadoop.conf.Configuration;
import org.apache.hadoop.fs.Path;
import org.apache.hadoop.io.IntWritable;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapreduce.Job;
import org.apache.hadoop.mapreduce.Mapper;
import org.apache.hadoop.mapreduce.lib.input.FileInputFormat;
import org.apache.hadoop.mapreduce.lib.output.FileOutputFormat;
import org.apache.hadoop.util.GenericOptionsParser;

public class TopN {
  public static void main(String[] args) throws Exception {
    Configuration conf = new Configuration();
    String[] otherArgs = (new GenericOptionsParser(conf, args)).getRemainingArgs();
    if (otherArgs.length != 2) {
      System.err.println("Usage: TopN <in> <out>");
      System.exit(2);
    }
    Job job = Job.getInstance(conf);
    job.setJobName("Top N");
    job.setJarByClass(TopN.class);
    job.setMapperClass(TopNMapper.class);
    job.setReducerClass(TopNReducer.class);
    job.setOutputKeyClass(Text.class);
    job.setOutputValueClass(IntWritable.class);
    FileInputFormat.addInputPath(job, new Path(otherArgs[0]));
    FileOutputFormat.setOutputPath(job, new Path(otherArgs[1]));
    System.exit(job.waitForCompletion(true) ? 0 : 1);
  }

  public static class TopNMapper extends Mapper<Object, Text, Text, IntWritable> {
    private static final IntWritable one = new IntWritable(1);

    private Text word = new Text();

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

    public void map(Object key, Text value, Mapper<Object, Text, Text, IntWritable>.Context context)
throws IOException, InterruptedException {
```

```
      String cleanLine = value.toString().toLowerCase().replaceAll(this.tokens, " ");
      StringTokenizer itr = new StringTokenizer(cleanLine);
      while (itr.hasMoreTokens()) {
        this.word.set(itr.nextToken().trim());
        context.write(this.word, one);
      }
    }
  }
}
```

TopNCombiner.class

```
package samples.topn;

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

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

TopNMapper.class

```
package samples.topn;

import java.io.IOException;
import java.util.StringTokenizer;
import org.apache.hadoop.io.IntWritable;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapreduce.Mapper;

public class TopNMapper extends Mapper<Object, Text, Text, IntWritable> {
  private static final IntWritable one = new IntWritable(1);

  private Text word = new Text();

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

  public vo```\\id map(Object key, Text value, Mapper<Object, Text, Text, IntWritable>.Context
context) throws IOException, InterruptedException {
    String cleanLine = value.toString().toLowerCase().replaceAll(this.tokens, " ");
    StringTokenizer itr = new StringTokenizer(cleanLine);
    while (itr.hasMoreTokens()) {
```

```
      this.word.set(itr.nextToken().trim());
      context.write(this.word, one);
    }
  }
}
```

TopNReducer.class

```
package samples.topn;

import java.io.IOException;
import java.util.HashMap;
import java.util.Map;
import org.apache.hadoop.io.IntWritable;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapreduce.Reducer;
import utils.MiscUtils;

public class TopNReducer extends Reducer<Text, IntWritable, Text, IntWritable> {
  private Map<Text, IntWritable> countMap = new HashMap<>();

  public void reduce(Text key, Iterable<IntWritable> values, Reducer<Text, IntWritable, Text,
IntWritable>.Context context) throws IOException, InterruptedException {
    int sum = 0;
    for (IntWritable val : values)
      sum += val.get();
    this.countMap.put(new Text(key), new IntWritable(sum));
  }

  protected void cleanup(Reducer<Text, IntWritable, Text, IntWritable>.Context context) throws
IOException, InterruptedException {
    Map<Text, IntWritable> sortedMap = MiscUtils.sortByValues(this.countMap);
    int counter = 0;
    for (Text key : sortedMap.keySet()) {
      if (counter++ == 20)
        break;
      context.write(key, sortedMap.get(key));
    }
  }
}
```

```
C:\hadoop-3.3.0\sbin>jps
11072 DataNode
20528 Jps
5620 ResourceManager
15532 NodeManager
6140 NameNode

C:\hadoop-3.3.0\sbin>hdfs dfs -mkdir /input_dir

C:\hadoop-3.3.0\sbin>hdfs dfs -ls /
Found 1 items
drwxr-xr-x   - Anusree supergroup          0 2021-05-08 19:46 /input_dir

C:\hadoop-3.3.0\sbin>hdfs dfs -copyFromLocal C:\input.txt /input_dir

C:\hadoop-3.3.0\sbin>hdfs dfs -ls /input_dir
Found 1 items
-rw-r--r--   1 Anusree supergroup         36 2021-05-08 19:48 /input_dir/input.txt

C:\hadoop-3.3.0\sbin>hdfs dfs -cat /input_dir/input.txt
hello
world
hello
hadoop
bye
```

```
C:\hadoop-3.3.0\sbin>hadoop jar C:\sort.jar samples.topn.TopN /input_dir/input.txt /output_dir
2021-05-08 19:54:54,582 INFO client.DefaultNoHARMFailoverProxyProvider: Connecting to ResourceManager at /0.0.0.0:8032
2021-05-08 19:54:55,291 INFO mapreduce.JobResourceUploader: Disabling Erasure Coding for path: /tmp/hadoop-yarn/staging/Anusree/.staging/job_1620483374279_0001
2021-05-08 19:54:55,821 INFO input.FileInputFormat: Total input files to process : 1
2021-05-08 19:54:56,261 INFO mapreduce.JobSubmitter: number of splits:1
2021-05-08 19:54:56,552 INFO mapreduce.JobSubmitter: Submitting tokens for job: job_1620483374279_0001
2021-05-08 19:54:56,552 INFO mapreduce.JobSubmitter: Executing with tokens: []
2021-05-08 19:54:56,843 INFO conf.Configuration: resource-types.xml not found
2021-05-08 19:54:56,843 INFO resource.ResourceUtils: Unable to find 'resource-types.xml'.
2021-05-08 19:54:57,387 INFO impl.YarnClientImpl: Submitted application application_1620483374279_0001
2021-05-08 19:54:57,507 INFO mapreduce.Job: The url to track the job: http://LAPTOP-JG329ESD:8088/proxy/application_1620483374279_0001/
2021-05-08 19:54:57,508 INFO mapreduce.Job: Running job: job_1620483374279_0001
2021-05-08 19:55:13,792 INFO mapreduce.Job: Job job_1620483374279_0001 running in uber mode : false
2021-05-08 19:55:13,794 INFO mapreduce.Job:  map 0% reduce 0%
2021-05-08 19:55:20,020 INFO mapreduce.Job:  map 100% reduce 0%
2021-05-08 19:55:27,116 INFO mapreduce.Job:  map 100% reduce 100%
2021-05-08 19:55:33,199 INFO mapreduce.Job: Job job_1620483374279_0001 completed successfully
2021-05-08 19:55:33,334 INFO mapreduce.Job: Counters: 54
        File System Counters
                FILE: Number of bytes read=65
                FILE: Number of bytes written=530397
                FILE: Number of read operations=0
                FILE: Number of large read operations=0
                FILE: Number of write operations=0
                HDFS: Number of bytes read=142
                HDFS: Number of bytes written=31
                HDFS: Number of read operations=8
                HDFS: Number of large read operations=0
                HDFS: Number of write operations=2
                HDFS: Number of bytes read erasure-coded=0
```

```
C:\hadoop-3.3.0\sbin>hdfs dfs -cat /output_dir/*
hello   2
hadoop  1
world   1
bye     1

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

LAB PROGRAM 8:Create a Map Reduce program to demonstrating join operation

```
// JoinDriver.java
import org.apache.hadoop.conf.Configured;
import org.apache.hadoop.fs.Path;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapred.*;
import org.apache.hadoop.mapred.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);
}
}  }
```

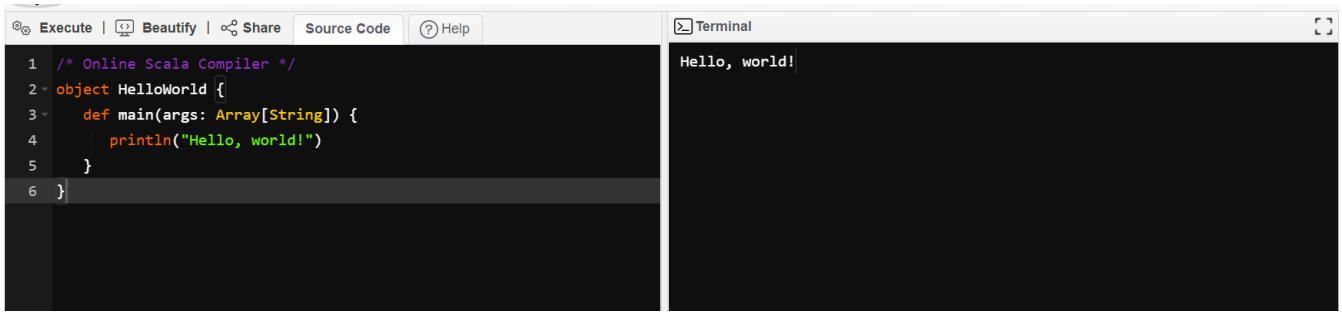
LAB8/Department_Employee_join_example/DeptName.txt

```
C:\hadoop-3.3.0\sbin>hdfs dfs -ls /join8_output/
Found 2 items
-rw-r--r--   1 Anusree supergroup          0 2021-06-13 12:16 /join8_output/_SUCCESS
-rw-r--r--   1 Anusree supergroup         71 2021-06-13 12:16 /join8_output/part-00000

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

LAB PROGRAM 9:

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



WORD COUNT

```
scala> val data= sc.textFile("scala.txt");
data: org.apache.spark.rdd.RDD[String] = scala.txt MapPartitionsRDD[1] at textFile at
<console>:24

scala> data.collect;
res1: Array[String] = Array(hello, how you doing?, are you alright)

scala> val splitdata= data.flatMap(line =>line.split(" "));
splitdata: org.apache.spark.rdd.RDD[String] = MapPartitionsRDD[2] at flatMap at
<console>:25

scala> splitdata.collect;
res2: Array[String] = Array(hello, how, you, doing?, are, you, alright)

scala> val mapdata = splitdata.map(word => (word,1));
mapdata: org.apache.spark.rdd.RDD[(String, Int)] = MapPartitionsRDD[3] at map at
<console>:25

scala> mapdata.collect;
res3: Array[(String, Int)] = Array((hello,1), (how,1), (you,1), (doing?,1), (are,1), (you,1),
(alright,1))

scala> val reducedata = mapdata.reduceByKey(_+_);
reducedata: org.apache.spark.rdd.RDD[(String, Int)] = ShuffledRDD[4] at reduceByKey at
<console>:25

scala> reducedata.collect;
res4: Array[(String, Int)] = Array((are,1), (doing?,1), (how,1), (hello,1), (you,2), (alright,1))
```

LAB PROGRAM 10:

Using RDD and FlaMap 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

```
package scalawordcount

import org.apache.spark.SparkConf
import org.apache.spark.SparkContext
import org.apache.spark.rdd.RDD.rddToPairRDDFunctions
import scala.collection.immutable.ListMap

object wordcount {
  def main (args: Array[String]) {
  val conf = new SparkConf().setAppName("WordCount").setMaster("local")
  val sc = new SparkContext(conf)
  val textFile = sc.textFile("input.txt")
  val counts = textFile.flatMap(line => line.split(" ")).map(word => (word, 1)).reduceByKey(_ + _)
  val sorted=ListMap(counts.collect.sortWith(_._2 > _._2):_*)// sort in descending order based on
values
  println(sorted)
  for((k,v)<-sorted)
  {
    if(v>4)
    {
      print(k+",")
      print(v)
      println()
    }
  }
  }
}
```

```
21/06/13 10:45:41 INFO DAGScheduler: ResultStage 1 (main at <unknown>:0) finished in 0.110 s
21/06/13 10:45:41 INFO DAGScheduler: Job 0 is finished. Cancelling potential speculative or zombie tasks for this job
21/06/13 10:45:41 INFO TaskSchedulerImpl: Killing all running tasks in stage 1: Stage finished
21/06/13 10:45:41 INFO DAGScheduler: Job 0 finished: main at <unknown>:0, took 0.823276 s
ListMap(Hello -> 6, Test -> 5, Hadoop -> 3, is -> 2, This -> 2, test -> 2, The -> 1, a -> 1, bye. -> 1, to -> 1, see -> 1, World
Hello,6
Test,5
21/06/13 10:45:41 INFO SparkContext: Invoking stop() from shutdown hook
21/06/13 10:45:41 INFO SparkUI: Stopped Spark web UI at http://LAPTOP-JG329ESD:4041
21/06/13 10:45:41 INFO MapOutputTrackerMasterEndpoint: MapOutputTrackerMasterEndpoint stopped!
21/06/13 10:45:41 INFO MemoryStore: MemoryStore cleared
21/06/13 10:45:41 INFO BlockManager: BlockManager stopped
21/06/13 10:45:41 INFO BlockManagerMaster: BlockManagerMaster stopped
21/06/13 10:45:41 INFO OutputCommitCoordinator$OutputCommitCoordinatorEndpoint: OutputCommitCoordinator stopped!
```