VISVESVARAYA TECHNOLOGICAL UNIVERSITY

"JnanaSangama", Belgaum -590014, Karnataka.

LAB REPORT
on

BIG DATA ANALYTICS

(29CS5PEBDA)

Submitted by

MALLIKA PRASAD (1BM19CS081)

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
APRIL-2022 to AUGUST-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 **MALLIKA PRASAD (1BM19CS081),** 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.

Pallavi GB
Assistant Professor
Department of CSE
BMSCE, Bengaluru

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

`

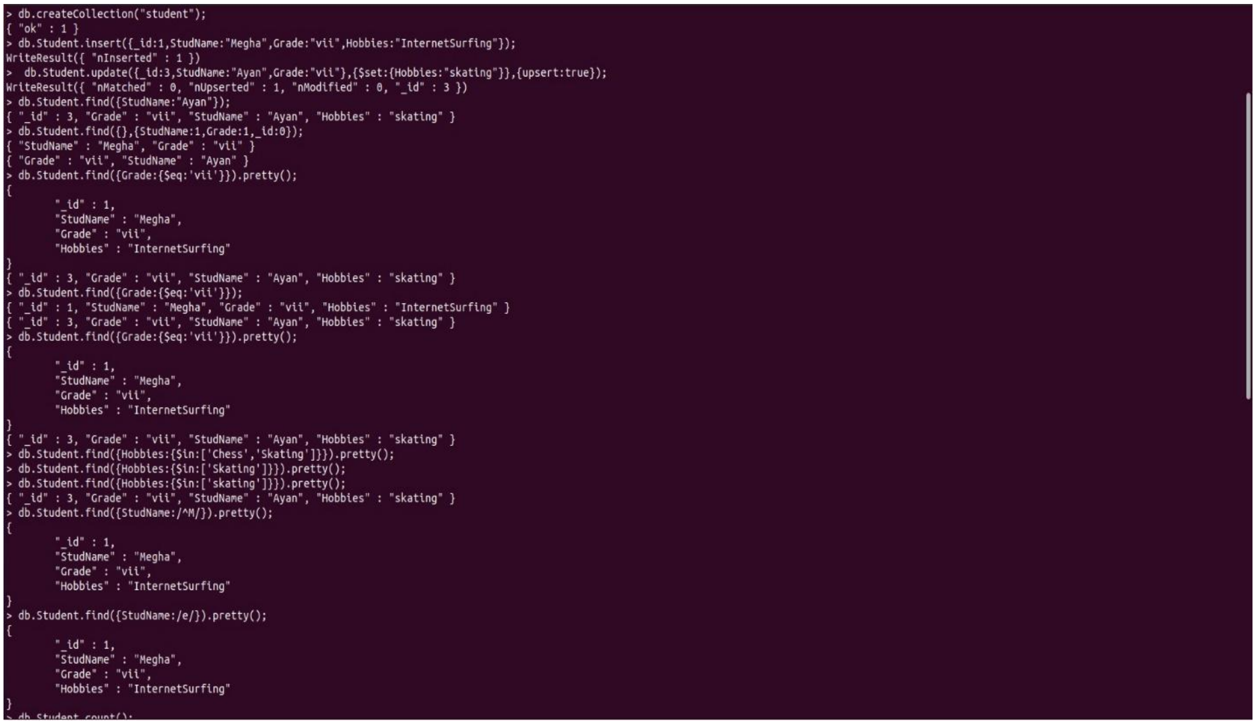
Index Sheet

Course Outcomes

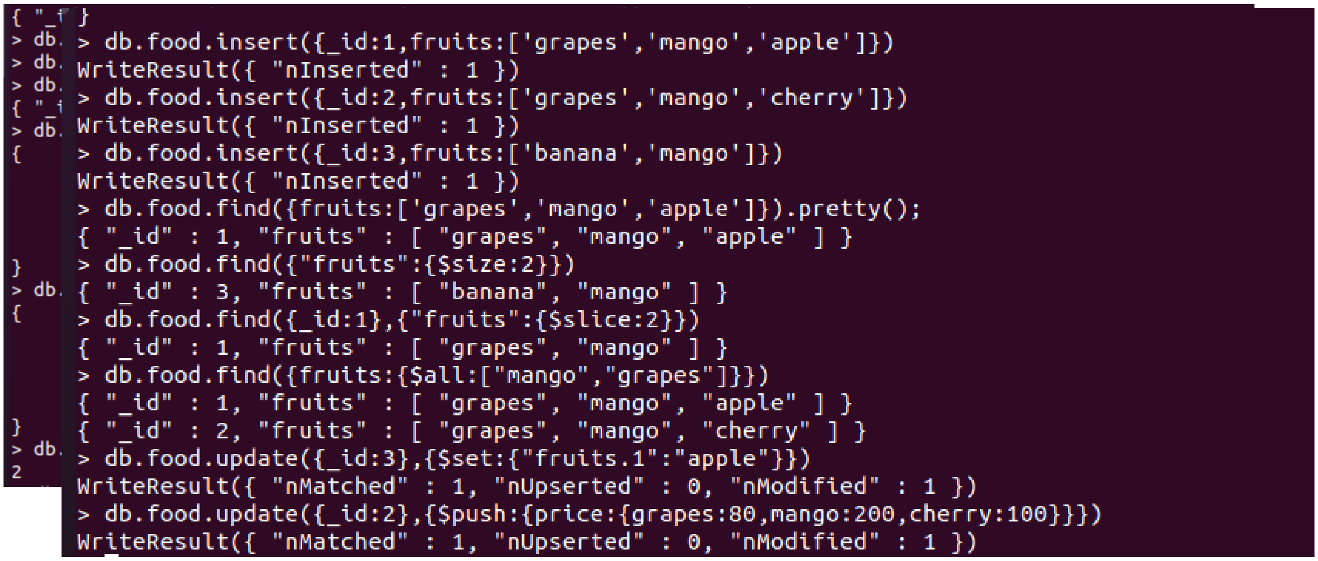
Co1	Apply the concept of NoSQL, Hadoop or Spark for a given task
C02	Analyze the Big Data and obtain insight using data analytics mechanisms.
C03	Design and implement Big data applications by applying NoSQL, Hadoop or Spark

1. MongoDB CRUD Demonstration

CRUD (CREATE, READ, UPDATE, DELETE) OPERATIONS

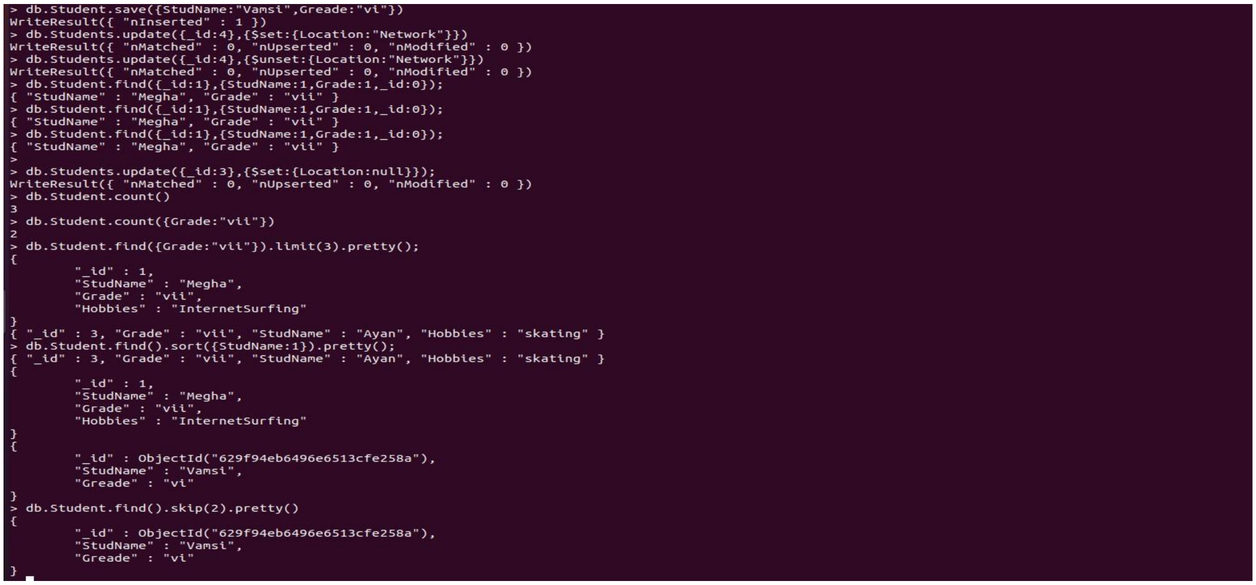


```
> db.createCollection("student");
{ "ok" : 1 }
> db.Student.insert({_id:1,StudName:"Megha",Grade:"vii",Hobbies:"InternetSurfing"});
WriteResult({ "nInserted" : 1 })
>  db.Student.update({_id:3,StudName:"Ayan",Grade:"vii"},{$set:{Hobbies:"skating"}},{upsert:true});
WriteResult({ "nMatched" : 0, "nUpserted" : 1, "nModified" : 0, "_id" : 3 })
> db.Student.find({StudName:"Ayan"});
{ "_id" : 3, "Grade" : "vii", "StudName" : "Ayan", "Hobbies" : "skating" }
> db.Student.find({},{StudName:1,Grade:1,_id:0});
{ "StudName" : "Megha", "Grade" : "vii" }
{ "Grade" : "vii", "StudName" : "Ayan" }
> db.Student.find({Grade:{$eq:'vii'}}).pretty();
{
        "_id" : 1,
        "StudName" : "Megha",
        "Grade" : "vii",
        "Hobbies" : "InternetSurfing"
}
{ "_id" : 3, "Grade" : "vii", "StudName" : "Ayan", "Hobbies" : "skating" }
> db.Student.find({Grade:{$eq:'vii'}});
{ "_id" : 1, "StudName" : "Megha", "Grade" : "vii", "Hobbies" : "InternetSurfing" }
{ "_id" : 3, "Grade" : "vii", "StudName" : "Ayan", "Hobbies" : "skating" }
> db.Student.find({Grade:{$eq:'vii'}}).pretty();
{
        "_id" : 1,
        "StudName" : "Megha",
        "Grade" : "vii",
        "Hobbies" : "InternetSurfing"
}
{ "_id" : 3, "Grade" : "vii", "StudName" : "Ayan", "Hobbies" : "skating" }
> db.Student.find({Hobbies:{$in:['Chess','Skating']}}).pretty();
> db.Student.find({Hobbies:{$in:['Skating']}}).pretty();
> db.Student.find({Hobbies:{$in:['skating']}}).pretty();
{ "_id" : 3, "Grade" : "vii", "StudName" : "Ayan", "Hobbies" : "skating" }
> db.Student.find({StudName:/^M/}).pretty();
{
        "_id" : 1,
        "StudName" : "Megha",
        "Grade" : "vii",
        "Hobbies" : "InternetSurfing"
}
> db.Student.find({StudName:/e/}).pretty();
{
        "_id" : 1,
        "StudName" : "Megha",
        "Grade" : "vii",
        "Hobbies" : "InternetSurfing"
}
> db.Student.count();
```



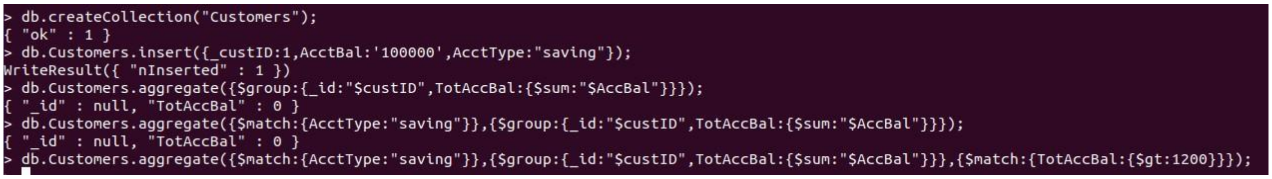
```
}
> db.food.insert({_id:1,fruits:['grapes','mango','apple']})
WriteResult({ "nInserted" : 1 })
> db.food.insert({_id:2,fruits:['grapes','mango','cherry']})
WriteResult({ "nInserted" : 1 })
> db.food.insert({_id:3,fruits:['banana','mango']})
WriteResult({ "nInserted" : 1 })
> db.food.find({fruits:['grapes','mango','apple']}).pretty();
{ "_id" : 1, "fruits" : [ "grapes", "mango", "apple" ] }
> db.food.find({"fruits":{$size:2}})
{ "_id" : 3, "fruits" : [ "banana", "mango" ] }
> db.food.find({_id:1},{"fruits":{$slice:2}})
{ "_id" : 1, "fruits" : [ "grapes", "mango" ] }
> db.food.find({fruits:{$all:["mango","grapes"]}})
{ "_id" : 1, "fruits" : [ "grapes", "mango", "apple" ] }
{ "_id" : 2, "fruits" : [ "grapes", "mango", "cherry" ] }
> db.food.update({_id:3},{$set:{"fruits.1":"apple"}})
WriteResult({ "nMatched" : 1, "nUpserted" : 0, "nModified" : 1 })
> db.food.update({_id:2},{$push:{price:{grapes:80,mango:200,cherry:100}}})
WriteResult({ "nMatched" : 1, "nUpserted" : 0, "nModified" : 1 })
```

Save method



```
> db.Student.save({StudName:"Vamsi",Greade:"vi"})
WriteResult({ "nInserted" : 1 })
> db.Students.update({_id:4},{$set:{Location:"Network"}})
WriteResult({ "nMatched" : 0, "nUpserted" : 0, "nModified" : 0 })
> db.Students.update({_id:4},{$unset:{Location:"Network"}})
WriteResult({ "nMatched" : 0, "nUpserted" : 0, "nModified" : 0 })
> db.Student.find({_id:1},{StudName:1,Grade:1,_id:0});
{ "StudName" : "Megha", "Grade" : "vii" }
> db.Student.find({_id:1},{StudName:1,Grade:1,_id:0});
{ "StudName" : "Megha", "Grade" : "vii" }
> db.Student.find({_id:1},{StudName:1,Grade:1,_id:0});
{ "StudName" : "Megha", "Grade" : "vii" }
>
> db.Students.update({_id:3},{$set:{Location:null}});
WriteResult({ "nMatched" : 0, "nUpserted" : 0, "nModified" : 0 })
> db.Student.count()
3
> db.Student.count({Grade:"vii"})
2
> db.Student.find({Grade:"vii"}).limit(3).pretty();
{
        "_id" : 1,
        "StudName" : "Megha",
        "Grade" : "vii",
        "Hobbies" : "InternetSurfing"
}
{ "_id" : 3, "Grade" : "vii", "StudName" : "Ayan", "Hobbies" : "skating" }
> db.Student.find().sort({StudName:1}).pretty();
{ "_id" : 3, "Grade" : "vii", "StudName" : "Ayan", "Hobbies" : "skating" }
{
        "_id" : 1,
        "StudName" : "Megha",
        "Grade" : "vii",
        "Hobbies" : "InternetSurfing"
}
{
        "_id" : ObjectId("629f94eb6496e6513cfe258a"),
        "StudName" : "Vamsi",
        "Greade" : "vi"
}
> db.Student.find().skip(2).pretty()
{
        "_id" : ObjectId("629f94eb6496e6513cfe258a"),
        "StudName" : "Vamsi",
        "Greade" : "vi"
}
```

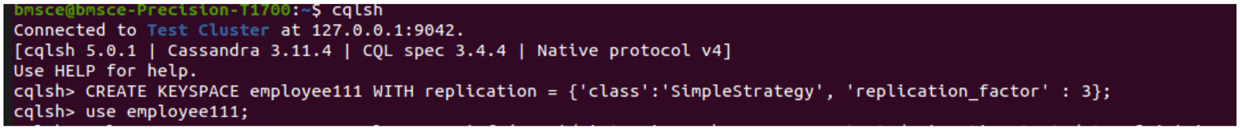
Aggregate function



```
> db.createCollection("Customers");
{ "ok" : 1 }
> db.Customers.insert({_custID:1,AcctBal:'100000',AcctType:"saving"});
WriteResult({ "nInserted" : 1 })
> db.Customers.aggregate({$group:{_id:"$custID",TotAccBal:{$sum:"$AccBal"}}});
{ "_id" : null, "TotAccBal" : 0 }
> db.Customers.aggregate({$match:{AcctType:"saving"}},{$group:{_id:"$custID",TotAccBal:{$sum:"$AccBal"}}});
{ "_id" : null, "TotAccBal" : 0 }
> db.Customers.aggregate({$match:{AcctType:"saving"}},{$group:{_id:"$custID",TotAccBal:{$sum:"$AccBal"}}},{$match:{TotAccBal:{$gt:1200}}});
```

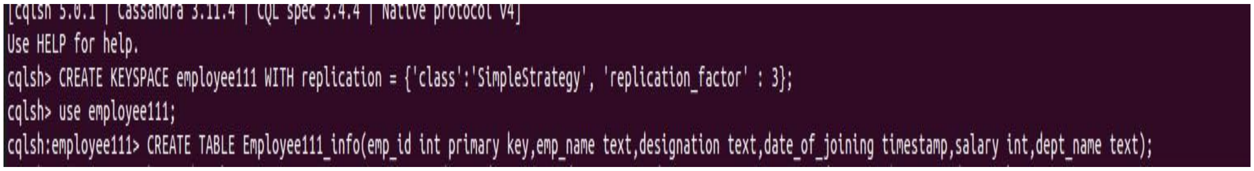
2. Perform the following DB operations using Cassandra. Create a keyspace by name Employee.

1.Create a key space by name Employee



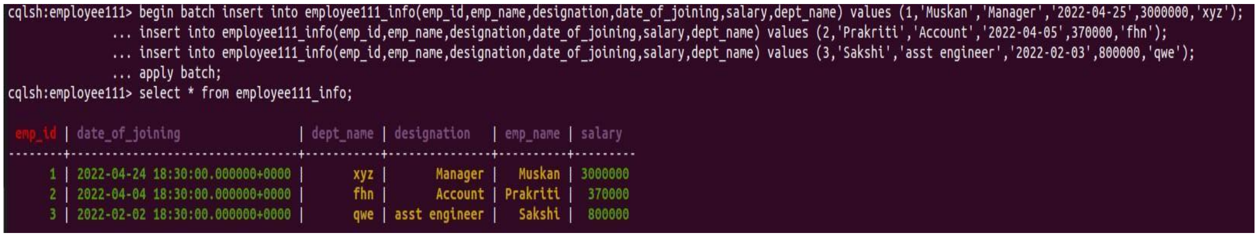
```
bmsce@bmsce-Precision-T1700:~$ cqlsh
Connected to Test Cluster at 127.0.0.1:9042.
[cqlsh 5.0.1 | Cassandra 3.11.4 | CQL spec 3.4.4 | Native protocol v4]
Use HELP for help.
cqlsh> CREATE KEYSPACE employee111 WITH replication = {'class':'SimpleStrategy', 'replication_factor' : 3};
cqlsh> use employee111;
```

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



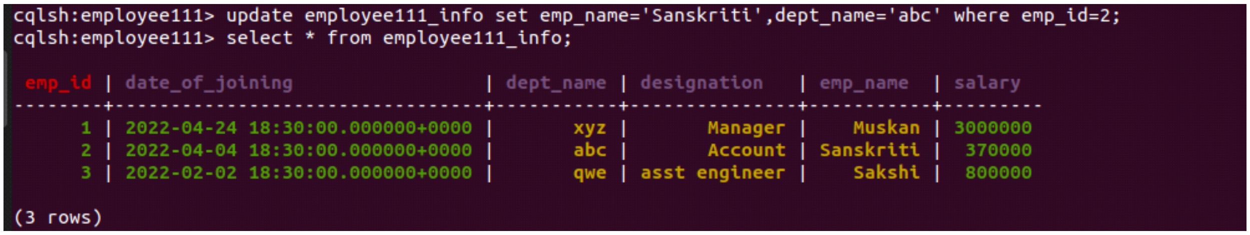
```
[cqlsh 5.0.1 | Cassandra 3.11.4 | CQL spec 3.4.4 | Native protocol v4]
Use HELP for help.
cqlsh> CREATE KEYSPACE employee111 WITH replication = {'class':'SimpleStrategy', 'replication_factor' : 3};
cqlsh> use employee111;
cqlsh:employee111> CREATE TABLE Employee111_info(emp_id int primary key,emp_name text,designation text,date_of_joining timestamp,salary int,dept_name text);
```

3.Insert the values into the table in batch



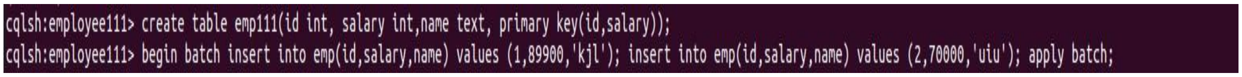
```
cqlsh:employee111> begin batch insert into employee111_info(emp_id,emp_name,designation,date_of_joining,salary,dept_name) values (1,'Muskan','Manager','2022-04-25',3000000,'xyz');
               ... insert into employee111_info(emp_id,emp_name,designation,date_of_joining,salary,dept_name) values (2,'Prakriti','Account','2022-04-05',370000,'fhn');
               ... insert into employee111_info(emp_id,emp_name,designation,date_of_joining,salary,dept_name) values (3,'Sakshi','asst engineer','2022-02-03',800000,'qwe');
               ... apply batch;
cqlsh:employee111> select * from employee111_info;

 emp_id | date_of_joining                 | dept_name | designation   | emp_name | salary
--------+---------------------------------+-----------+---------------+----------+---------
      1 | 2022-04-24 18:30:00.000000+0000 |       xyz |       Manager |   Muskan | 3000000
      2 | 2022-04-04 18:30:00.000000+0000 |       fhn |       Account | Prakriti |  370000
      3 | 2022-02-02 18:30:00.000000+0000 |       qwe | asst engineer |   Sakshi |  800000
```

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

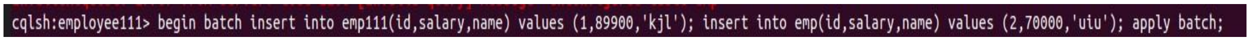


```
cqlsh:employee111> update employee111_info set emp_name='Sanskriti',dept_name='abc' where emp_id=2;
cqlsh:employee111> select * from employee111_info;

 emp_id | date_of_joining                 | dept_name | designation   | emp_name  | salary
--------+---------------------------------+-----------+---------------+-----------+---------
      1 | 2022-04-24 18:30:00.000000+0000 |       xyz |       Manager |    Muskan | 3000000
      2 | 2022-04-04 18:30:00.000000+0000 |       abc |       Account | Sanskriti |  370000
      3 | 2022-02-02 18:30:00.000000+0000 |       qwe | asst engineer |    Sakshi |  800000

(3 rows)
```

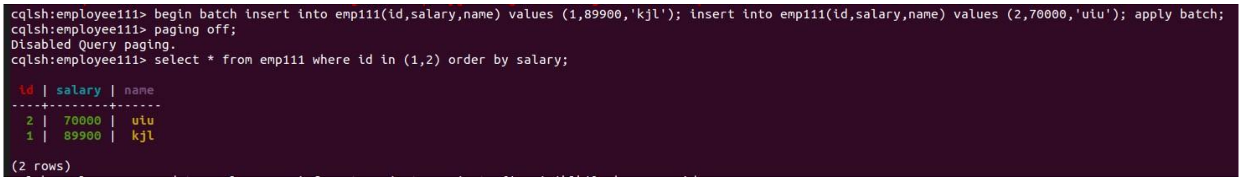
5. Sort the details of Employee records based on salary



```
cqlsh:employee111> create table emp111(id int, salary int,name text, primary key(id,salary));
cqlsh:employee111> begin batch insert into emp(id,salary,name) values (1,89900,'kjl'); insert into emp(id,salary,name) values (2,70000,'uiu'); apply batch;
```

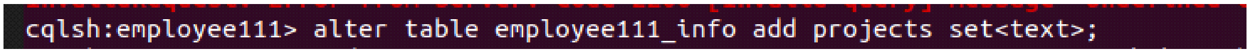


```
cqlsh:employee111> begin batch insert into emp111(id,salary,name) values (1,89900,'kjl'); insert into emp(id,salary,name) values (2,70000,'uiu'); apply batch;
```



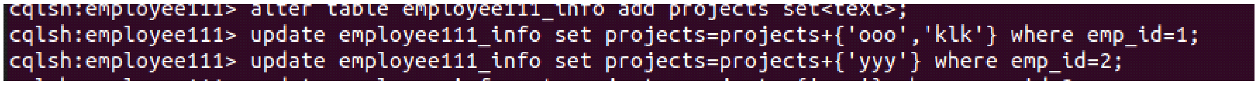
```
cqlsh:employee111> begin batch insert into emp111(id,salary,name) values (1,89900,'kjl'); insert into emp111(id,salary,name) values (2,70000,'uiu'); apply batch;
cqlsh:employee111> paging off;
Disabled Query paging.
cqlsh:employee111> select * from emp111 where id in (1,2) order by salary;

 id | salary | name
----+--------+------
  2 |  70000 |  uiu
  1 |  89900 |  kjl

(2 rows)
```

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.

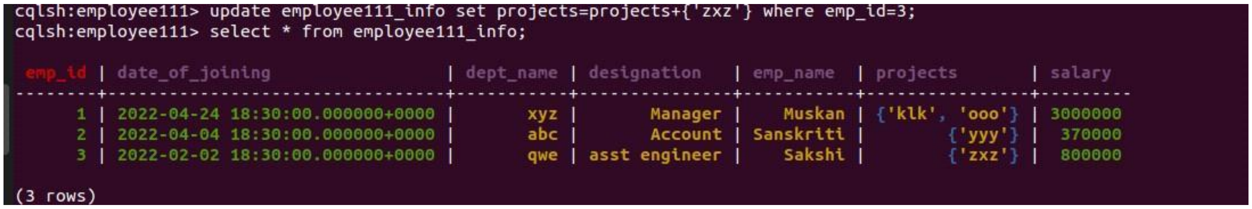


```
cqlsh:employee111> alter table employee111_info add projects set<text>;
```

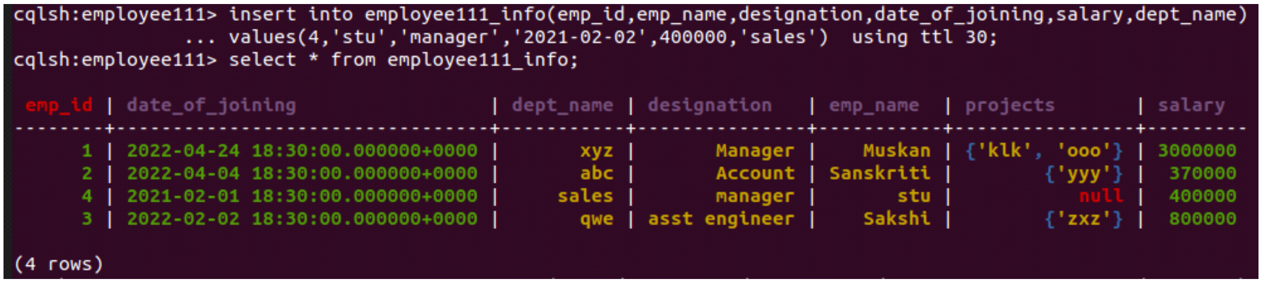
7. Update the altered table to add project names.



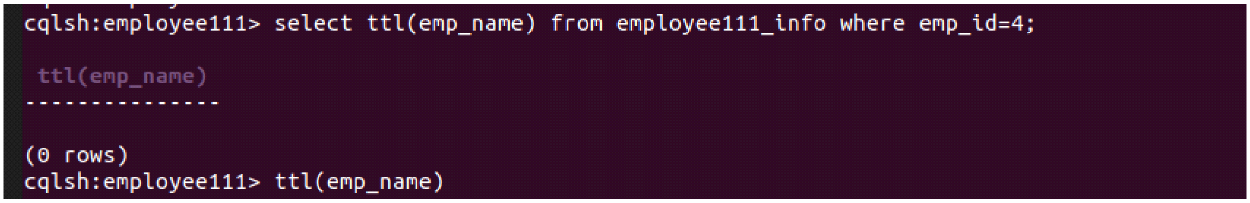
```
cqlsh:employee111> alter table employee111_info add projects set<text>;
cqlsh:employee111> update employee111_info set projects=projects+{'ooo','klk'} where emp_id=1;
cqlsh:employee111> update employee111_info set projects=projects+{'yyy'} where emp_id=2;
```



```
cqlsh:employee111> update employee111_info set projects=projects+{'zxz'} where emp_id=3;
cqlsh:employee111> select * from employee111_info;

 emp_id | date_of_joining                 | dept_name | designation    | emp_name  | projects       | salary
--------+---------------------------------+-----------+----------------+-----------+----------------+---------
      1 | 2022-04-24 18:30:00.000000+0000 |       xyz |        Manager |    Muskan | {'klk', 'ooo'} | 3000000
      2 | 2022-04-04 18:30:00.000000+0000 |       abc |        Account | Sanskriti |        {'yyy'} |  370000
      3 | 2022-02-02 18:30:00.000000+0000 |       qwe | asst engineer  |    Sakshi |        {'zxz'} |  800000

(3 rows)
```



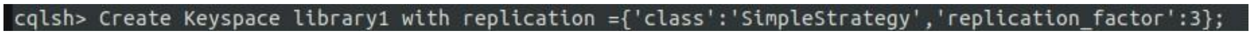
```
cqlsh:employee111> insert into employee111_info(emp_id,emp_name,designation,date_of_joining,salary,dept_name)
               ... values(4,'stu','manager','2021-02-02',400000,'sales')  using ttl 30;
cqlsh:employee111> select * from employee111_info;

 emp_id | date_of_joining                 | dept_name | designation    | emp_name  | projects       | salary
--------+---------------------------------+-----------+----------------+-----------+----------------+---------
      1 | 2022-04-24 18:30:00.000000+0000 |       xyz |        Manager |    Muskan | {'klk', 'ooo'} | 3000000
      2 | 2022-04-04 18:30:00.000000+0000 |       abc |        Account | Sanskriti |        {'yyy'} |  370000
      4 | 2021-02-01 18:30:00.000000+0000 |     sales |        manager |       stu |           null |  400000
      3 | 2022-02-02 18:30:00.000000+0000 |       qwe | asst engineer  |    Sakshi |        {'zxz'} |  800000

(4 rows)
```



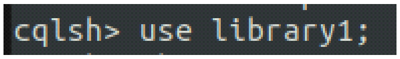
```
cqlsh:employee111> select ttl(emp_name) from employee111_info where emp_id=4;

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

(0 rows)
cqlsh:employee111> ttl(emp_name)
```

3. Perform the following DB operations using Cassandra.

1. Create a key space by name Library



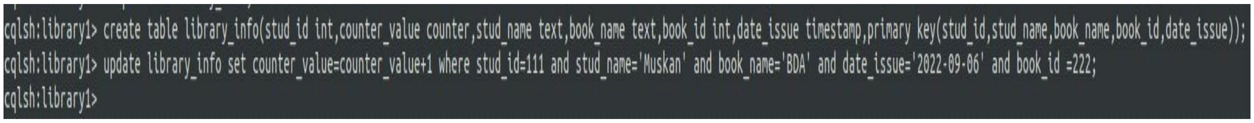
```
cqlsh> Create Keyspace library1 with replication ={'class':'SimpleStrategy','replication_factor':3};
```



```
cqlsh> use library1;
```

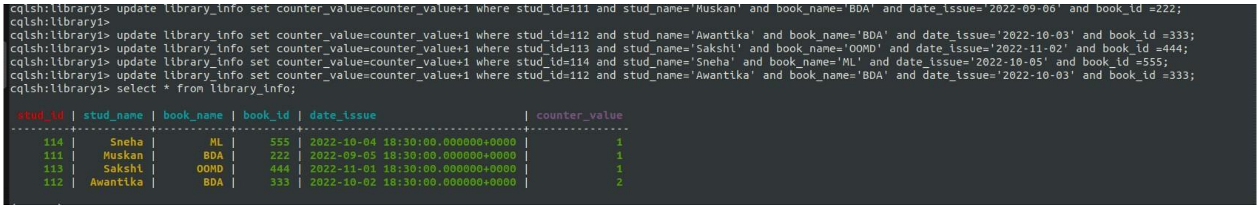
2. Create a column family by name Library-Info with attributes Stud_Id Primary

Key, Counter_value of type Counter, Stud_Name, Book-Name, Book-Id, Date_of_issue



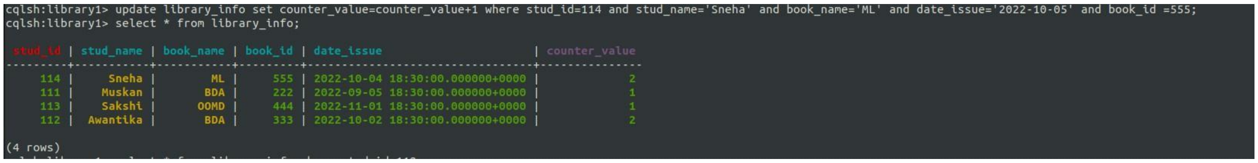
```
cqlsh:library1> create table library_info(stud_id int,counter_value counter,stud_name text,book_name text,book_id int,date_issue timestamp,primary key(stud_id,stud_name,book_name,book_id,date_issue));
cqlsh:library1> update library_info set counter_value=counter_value+1 where stud_id=111 and stud_name='Muskan' and book_name='BDA' and date_issue='2022-09-06' and book_id =222;
cqlsh:library1>
```

3. Insert the values into the table in batch



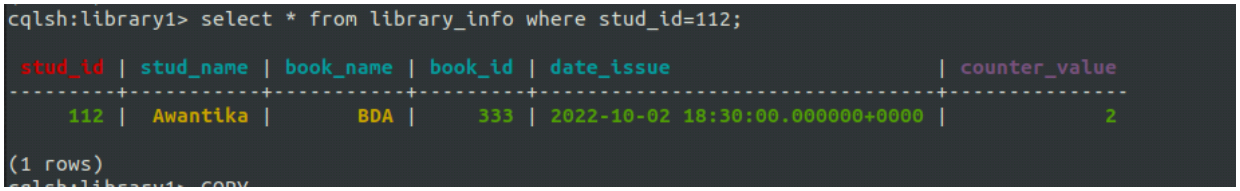
```
cqlsh:library1> update library_info set counter_value=counter_value+1 where stud_id=111 and stud_name='Muskan' and book_name='BDA' and date_issue='2022-09-06' and book_id =222;
cqlsh:library1>
cqlsh:library1> update library_info set counter_value=counter_value+1 where stud_id=112 and stud_name='Awantika' and book_name='BDA' and date_issue='2022-10-03' and book_id =333;
cqlsh:library1> update library_info set counter_value=counter_value+1 where stud_id=113 and stud_name='Sakshi' and book_name='OOMD' and date_issue='2022-11-02' and book_id =444;
cqlsh:library1> update library_info set counter_value=counter_value+1 where stud_id=114 and stud_name='Sneha' and book_name='ML' and date_issue='2022-10-05' and book_id =555;
cqlsh:library1> update library_info set counter_value=counter_value+1 where stud_id=112 and stud_name='Awantika' and book_name='BDA' and date_issue='2022-10-03' and book_id =333;
cqlsh:library1> select * from library_info;

 stud_id | stud_name | book_name | book_id | date_issue                      | counter_value
---------+-----------+-----------+---------+---------------------------------+---------------
     114 |     Sneha |        ML |     555 | 2022-10-04 18:30:00.000000+0000 |             1
     111 |    Muskan |       BDA |     222 | 2022-09-05 18:30:00.000000+0000 |             1
     113 |    Sakshi |      OOMD |     444 | 2022-11-01 18:30:00.000000+0000 |             1
     112 |  Awantika |       BDA |     333 | 2022-10-02 18:30:00.000000+0000 |             2
```

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



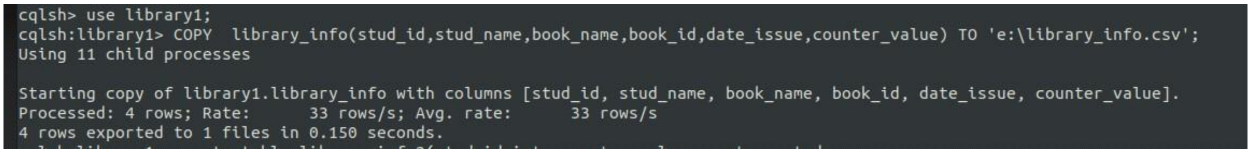
```
cqlsh:library1> update library_info set counter_value=counter_value+1 where stud_id=114 and stud_name='Sneha' and book_name='ML' and date_issue='2022-10-05' and book_id =555;
cqlsh:library1> select * from library_info;

 stud_id | stud_name | book_name | book_id | date_issue                      | counter_value
---------+-----------+-----------+---------+---------------------------------+---------------
     114 |     Sneha |        ML |     555 | 2022-10-04 18:30:00.000000+0000 |             2
     111 |    Muskan |       BDA |     222 | 2022-09-05 18:30:00.000000+0000 |             1
     113 |    Sakshi |      OOMD |     444 | 2022-11-01 18:30:00.000000+0000 |             1
     112 |  Awantika |       BDA |     333 | 2022-10-02 18:30:00.000000+0000 |             2

(4 rows)
```

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



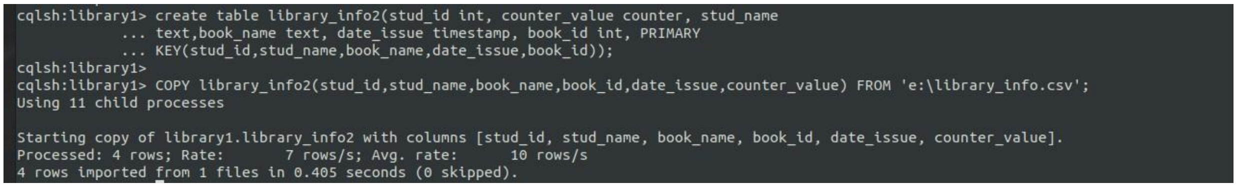
```
cqlsh:library1> select * from library_info where stud_id=112;

 stud_id | stud_name | book_name | book_id | date_issue                      | counter_value
---------+-----------+-----------+---------+---------------------------------+---------------
     112 |  Awantika |       BDA |     333 | 2022-10-02 18:30:00.000000+0000 |             2

(1 rows)
```

6. Export the created column to a csv file



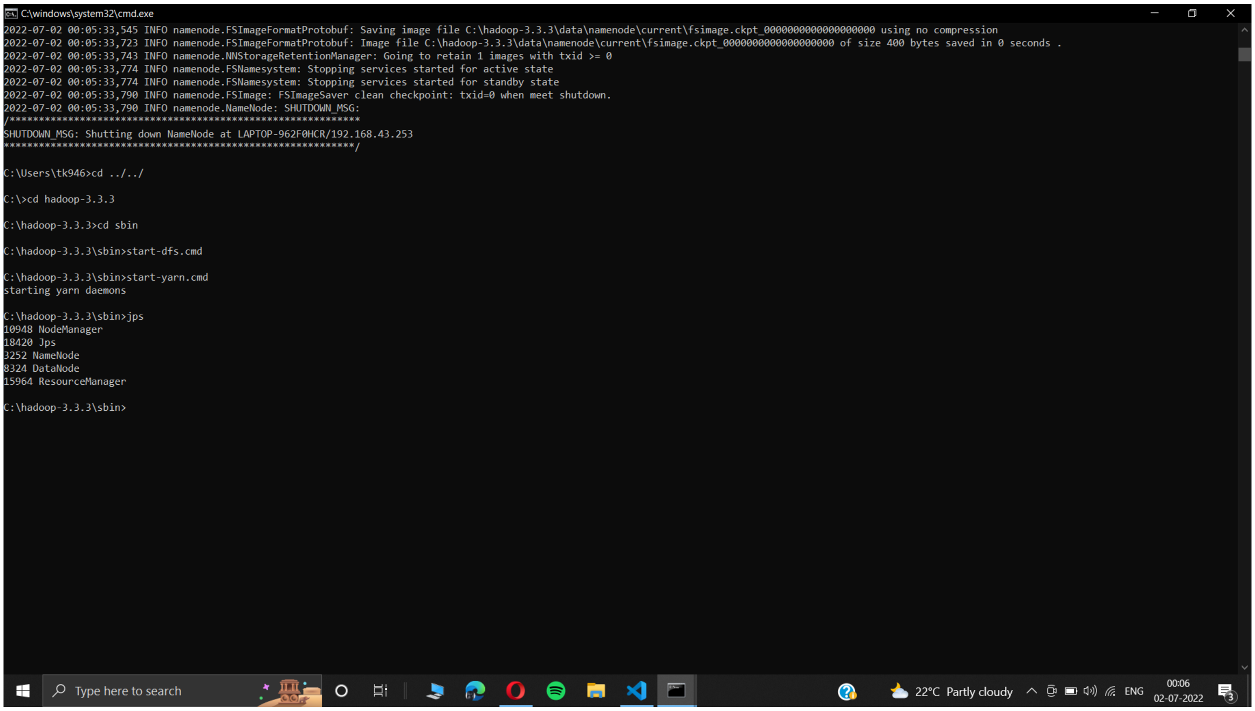
```
cqlsh> use library1;
cqlsh:library1> COPY  library_info(stud_id,stud_name,book_name,book_id,date_issue,counter_value) TO 'e:\library_info.csv';
Using 11 child processes

Starting copy of library1.library_info with columns [stud_id, stud_name, book_name, book_id, date_issue, counter_value].
Processed: 4 rows; Rate:      33 rows/s; Avg. rate:      33 rows/s
4 rows exported to 1 files in 0.150 seconds.
```

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

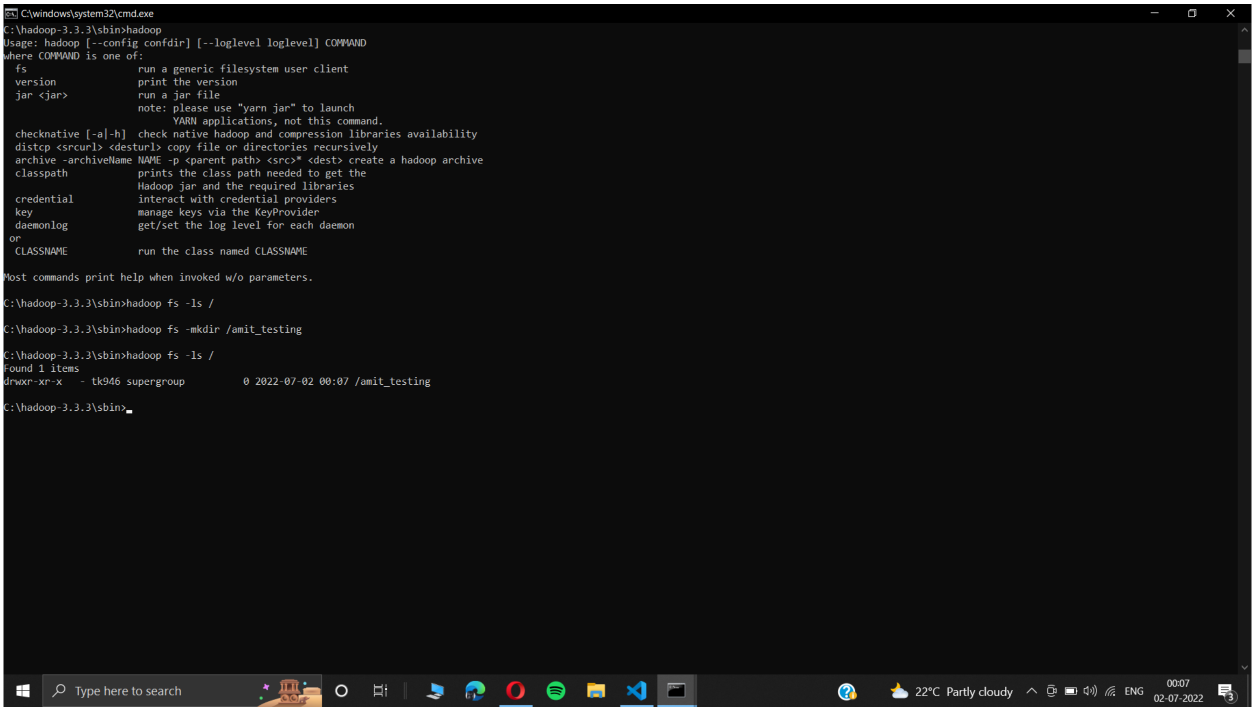


```
cqlsh:library1> create table library_info2(stud_id int, counter_value counter, stud_name
           ... text,book_name text, date_issue timestamp, book_id int, PRIMARY
           ... KEY(stud_id,stud_name,book_name,date_issue,book_id));
cqlsh:library1>
cqlsh:library1> COPY library_info2(stud_id,stud_name,book_name,book_id,date_issue,counter_value) FROM 'e:\library_info.csv';
Using 11 child processes

Starting copy of library1.library_info2 with columns [stud_id, stud_name, book_name, book_id, date_issue, counter_value].
Processed: 4 rows; Rate:       7 rows/s; Avg. rate:      10 rows/s
4 rows imported from 1 files in 0.405 seconds (0 skipped).
```

4. ScreenShots of Hadoop installations



```
2022-07-02 00:05:33,545 INFO namenode.FSImageFormatProtobuf: Saving image file C:\hadoop-3.3.3\data\namenode\current\fsimage.ckpt_0000000000000000000 using no compression
2022-07-02 00:05:33,723 INFO namenode.FSImageFormatProtobuf: Image file C:\hadoop-3.3.3\data\namenode\current\fsimage.ckpt_0000000000000000000 of size 400 bytes saved in 0 seconds .
2022-07-02 00:05:33,743 INFO namenode.NNStorageRetentionManager: Going to retain 1 images with txid >= 0
2022-07-02 00:05:33,774 INFO namenode.FSNamesystem: Stopping services started for active state
2022-07-02 00:05:33,774 INFO namenode.FSNamesystem: Stopping services started for standby state
2022-07-02 00:05:33,790 INFO namenode.FSImage: FSImageSaver clean checkpoint: txid=0 when meet shutdown.
2022-07-02 00:05:33,790 INFO namenode.NameNode: SHUTDOWN_MSG:
/************************************************************
SHUTDOWN_MSG: Shutting down NameNode at LAPTOP-962F0HCR/192.168.43.253
************************************************************/

C:\Users\tk946>cd ../../

C:\>cd hadoop-3.3.3

C:\hadoop-3.3.3>cd sbin

C:\hadoop-3.3.3\sbin>start-dfs.cmd

C:\hadoop-3.3.3\sbin>start-yarn.cmd
starting yarn daemons

C:\hadoop-3.3.3\sbin>jps
10948 NodeManager
18420 Jps
3252 NameNode
8324 DataNode
15964 ResourceManager

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



```
C:\hadoop-3.3.3\sbin>hadoop
Usage: hadoop [--config confdir] [--loglevel loglevel] COMMAND
where COMMAND is one of:
  fs                   run a generic filesystem user client
  version              print the version
  jar <jar>            run a jar file
                       note: please use "yarn jar" to launch
                             YARN applications, not this command.
  checknative [-a|-h]  check native hadoop and compression libraries availability
  distcp <srcurl> <desturl> copy file or directories recursively
  archive -archiveName NAME -p <parent path> <src>* <dest> create a hadoop archive
  classpath            prints the class path needed to get the
                       Hadoop jar and the required libraries
  credential           interact with credential providers
  key                  manage keys via the KeyProvider
  daemonlog            get/set the log level for each daemon
 or
  CLASSNAME            run the class named CLASSNAME

Most commands print help when invoked w/o parameters.

C:\hadoop-3.3.3\sbin>hadoop fs -ls /

C:\hadoop-3.3.3\sbin>hadoop fs -mkdir /amit_testing

C:\hadoop-3.3.3\sbin>hadoop fs -ls /
Found 1 items
drwxr-xr-x   - tk946 supergroup          0 2022-07-02 00:07 /amit_testing

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

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

1. mkdir

Hadoop HDFS mkdir Command Usage
mkdir
Hadoop HDFS mkdir Command Example
hdfs dfs -mkdir /abc
Hadoop HDFS mkdir Command Description
This HDFS command takes path URI's as an argument and creates directories.

2. ls

Hadoop HDFS ls Command Usage
ls
Hadoop HDFS ls Command Example
hadoop fs -ls /
Hadoop HDFS ls Commnad Description
This Hadoop HDFS ls command displays a list of the contents of a directory specified by path provided by the user, showing the names, permissions, owner, size and
modification date for each entry.

3. put

Hadoop HDFS put Command Usage
put
Hadoop HDFS put Command Example
hdfs dfs -put /home/hduser/Desktop/Welcome.txt /abc/WC.txt
Hadoop
HDFS put Command Description
This hadoop basic command copies the file or directory from the local file system to the destination within the DFS. Display the contents of the file WC.txt hdfs dfs -cat /abc/WC.txt

4. copyFromLocal

Hadoop HDFS copyFromLocal Command Usage
copyFromLocal
Hadoop HDFS copyFromLocal Command Example
hdfs dfs -put /home/hduser/Desktop/Welcome.txt /abc/WC.txt
Hadoop HDFS copyFromLocal Command Description
This hadoop shell command is similar to put command, but the source is restricted to a local file reference.
Display the contents of the file WC2.txt hdfs dfs -cat /abc/WC2.txt

5. get

Hadoop HDFS get Command Usage
get [-crc]

i. Hadoop HDFS get Command Example

```
hdfs dfs -get /abc/WC.txt /home/hduser/Downloads/WWC.txt
```

This HDFS fs command copies the file or directory in HDFS identified by the source to the local file system path identified by local destination.

ii. Hadoop HDFS get Command Example hdfs dfs -getmerge /abc/WC.txt /abc/WC2.txt
/home/hduser/Desktop/Merge.txt

This HDFS basic command retrieves all files that match to the source path entered by the user in HDFS, and creates a copy of them to one single, merged file in the local file system identified by local destination.

iii. Hadoop HDFS get Command Example hadoop fs -getfacl /abc/

This Apache Hadoop command shows the Access Control Lists (ACLs) of files and directories.

6. copyToLocal

Hadoop HDFS copyToLocal Command Usage
copyToLocal
Hadoop HDFS copyToLocal Command Example

```
hdfs dfs -copyToLocal /abc/WC.txt /home/hduser/Desktop
```

Similar to get command, only the difference is that in this the destination is restricted to a local file reference.

7. cat

Hadoop HDFS cat Command Usage
cat
Hadoop HDFS cat Command Example

```
hdfs dfs -cat /abc/WC.txt
```

This Hadoop fs shell command displays the contents of the filename on console or stdout.

8. mv

Hadoop HDFS mv Command Usage
mv
Hadoop HDFS mv Command Example

```
hadoop fs -mv /abc /FFF
hadoop fs -ls /FFF
```

This basic HDFS command moves the file or directory indicated by the source to destination, within HDFS.

9. cp

Hadoop HDFS cp Command Usage
cp
Hadoop HDFS cp Command Example

```
hadoop fs –cp /CSE/ /LLL
hadoop fs –ls /LLL
```

The cp command copies a file from one directory to another directory within the HDFS.

// start hadoop (must be in hduser)

```
$ start-all.sh
```

// creating a directory inside hadoop -mkdir

```
$ hdfs dfs -mkdir /bda_hadoop
```

// listing all content inside hadoop - ls

```
$ hadoop fs -ls /
```

// copyig files from deskop using put command - put

```
$ hdfs dfs -put /home/hduser/Desktop/bda_local.txt /bda_hadoop/file.txt
```

// cat command(listing the content of file in hadoop) -cat

```
$ hdfs dfs -cat /bda_hadoop/file.txt
```

// copying files from local reference using copyFromLocal cmd.

```
$ hdfs dfs -copyFromLocal /home/hduser/Desktop/bda_local.txt /bda_hadoop/file_cp_local.txt
```

```
$ hdfs dfs -cat /bda_hadoop/file_cp_local.txt
```

// get command

```
$ hdfs dfs -get /bda_hadoop/file.txt /home/hduser/Downloads/downloaded_file.txt
$ hdfs dfs -getmerge /bda_hadoop/file.txt /bda_hadoop/file_cp_local.txt
/home/hduser/Downloads/downloaded_file.txt

$ hadoop fs -getfacl /bda_hadoop/

# file: /bda_hadoop

# owner: hduser

# group: supergroup

user::rwx

group::r-x

other::r-x


// copyToLocal

$ hdfs dfs -copyToLocal /bda_hadoop/file.txt /home/hduser/Desktop


// mv command

$ hadoop fs -mv /bda_hadoop /abc


$ hadoop fs -ls /abc

Found 1 items

drwxr-xr-x   - hduser supergroup          0 2022-06-06 11:52 /abc/bda_hadoop


// copy

$ hadoop fs -cp /hello/ /hadoop_lab
```

6. Create a Map Reduce program to

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

b) find the mean max temperature for every month.

Dataset: https://github.com/tomwhite/hadoop-book/tree/master/input/ncdc/all

Driver code:

```
package averagetemp_amit;


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);
  }
}
```

Mapper:

```
package averagetemp_amit;
```

```java
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));

  }

}
```

Reducer:

```java
package averagetemp_amit;


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));

  }

}
```

7. Create a Map Reduce program to sort the content in an alphabetic order listing only top 10 maximum occurrences of words.

Driver:

```
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);

         }

        }

       }

      }
```

Mapper:

```
import java.io.IOException;

import java.util.StringTokenizer;

import org.apache.hadoop.io.IntWritable;

import org.apache.hadoop.io.Text;

import org.apache.hadoop.mapreduce.Mapper;


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

  private static final IntWritable one = new IntWritable(1);

  private Text word = new Text();

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


  public void map(Object key, Text value, Mapper<Object, Text, Text, IntWritable>.Context
context) throws IOException, InterruptedException {
    String cleanLine = value.toString().toLowerCase().replaceAll(this.tokens, " ");
    StringTokenizer itr = new StringTokenizer(cleanLine);
    while (itr.hasMoreTokens()) {
      this.word.set(itr.nextToken().trim());
      context.write(this.word, one);
    }
  }
}
```

Combiner:

```
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));
  }
}
```

Reducer:

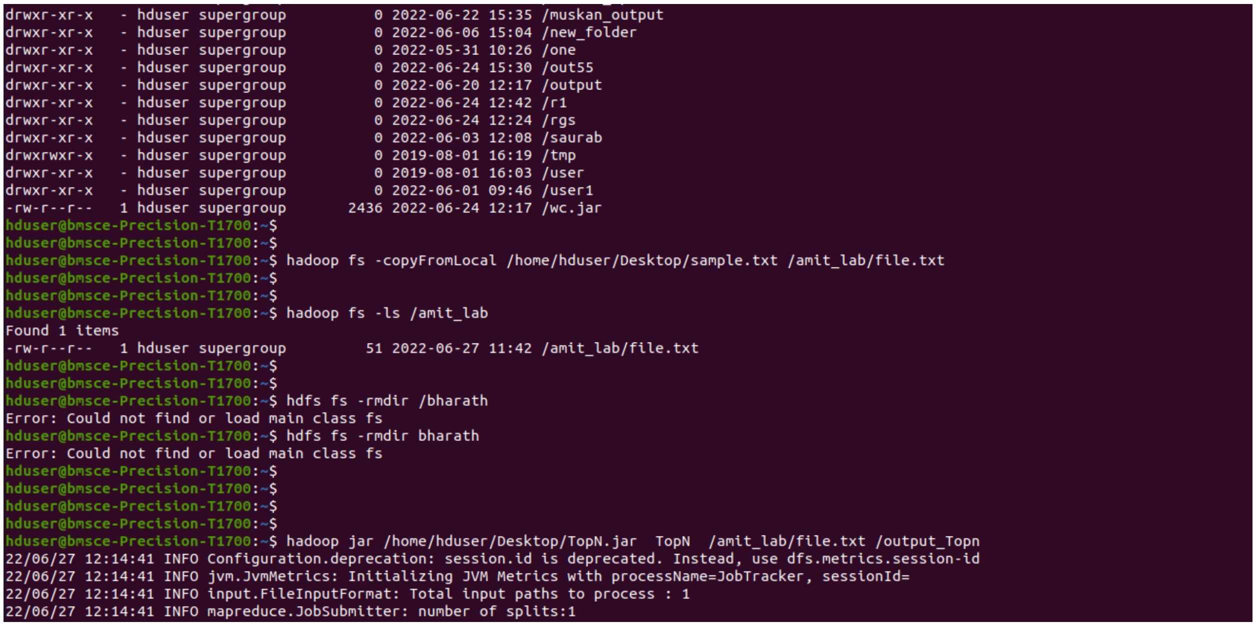
```
import java.io.IOException;
import java.util.HashMap;
import java.util.Map;
import org.apache.hadoop.io.IntWritable;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapreduce.Reducer;
import utils.MiscUtils;

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));
    }
  }
}
```

MiscUtils.java

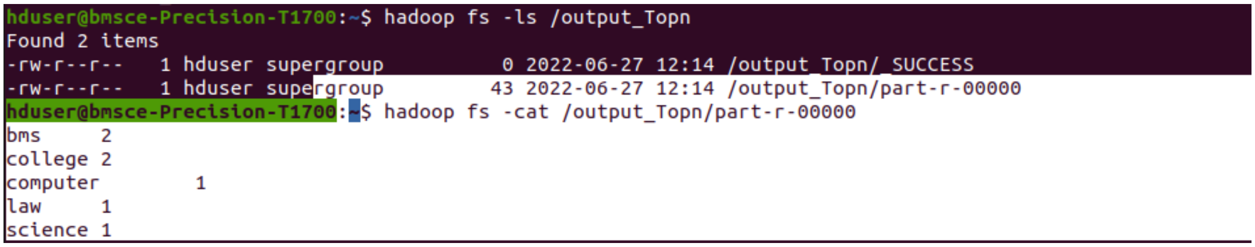
```
package utils;
import java.util.*;
public class MiscUtils {
```

```
public static <K extends Comparable, V extends Comparable> Map<K, V> sortByValues(Map<K, V> map) {
List<Map.Entry<K, V>> entries = new LinkedList<Map.Entry<K, V>>(map.entrySet());
Collections.sort(entries, new Comparator<Map.Entry<K, V>>() {
@Override
public int compare(Map.Entry<K, V> o1, Map.Entry<K, V> o2) {
return o2.getValue().compareTo(o1.getValue());
}
});
//LinkedHashMap will keep the keys in the order they are inserted

//which is currently sorted on natural ordering

Map<K, V> sortedMap = new LinkedHashMap<K, V>();
for (Map.Entry<K, V> entry : entries) {
sortedMap.put(entry.getKey(), entry.getValue());
}
return sortedMap;
}
}
```

Output:



```
drwxr-xr-x   - hduser supergroup          0 2022-06-22 15:35 /muskan_output
drwxr-xr-x   - hduser supergroup          0 2022-06-06 15:04 /new_folder
drwxr-xr-x   - hduser supergroup          0 2022-05-31 10:26 /one
drwxr-xr-x   - hduser supergroup          0 2022-06-24 15:30 /out55
drwxr-xr-x   - hduser supergroup          0 2022-06-20 12:17 /output
drwxr-xr-x   - hduser supergroup          0 2022-06-24 12:42 /r1
drwxr-xr-x   - hduser supergroup          0 2022-06-24 12:24 /rgs
drwxr-xr-x   - hduser supergroup          0 2022-06-03 12:08 /saurab
drwxrwxr-x   - hduser supergroup          0 2019-08-01 16:19 /tmp
drwxr-xr-x   - hduser supergroup          0 2019-08-01 16:03 /user
drwxr-xr-x   - hduser supergroup          0 2022-06-01 09:46 /user1
-rw-r--r--   1 hduser supergroup       2436 2022-06-24 12:17 /wc.jar
hduser@bmsce-Precision-T1700:~$
hduser@bmsce-Precision-T1700:~$
hduser@bmsce-Precision-T1700:~$ hadoop fs -copyFromLocal /home/hduser/Desktop/sample.txt /amit_lab/file.txt
hduser@bmsce-Precision-T1700:~$
hduser@bmsce-Precision-T1700:~$
hduser@bmsce-Precision-T1700:~$ hadoop fs -ls /amit_lab
Found 1 items
-rw-r--r--   1 hduser supergroup         51 2022-06-27 11:42 /amit_lab/file.txt
hduser@bmsce-Precision-T1700:~$
hduser@bmsce-Precision-T1700:~$
hduser@bmsce-Precision-T1700:~$ hdfs fs -rmdir /bharath
Error: Could not find or load main class fs
hduser@bmsce-Precision-T1700:~$ hdfs fs -rmdir bharath
Error: Could not find or load main class fs
hduser@bmsce-Precision-T1700:~$
hduser@bmsce-Precision-T1700:~$
hduser@bmsce-Precision-T1700:~$
hduser@bmsce-Precision-T1700:~$
hduser@bmsce-Precision-T1700:~$ hadoop jar /home/hduser/Desktop/TopN.jar  TopN  /amit_lab/file.txt /output_Topn
22/06/27 12:14:41 INFO Configuration.deprecation: session.id is deprecated. Instead, use dfs.metrics.session-id
22/06/27 12:14:41 INFO jvm.JvmMetrics: Initializing JVM Metrics with processName=JobTracker, sessionId=
22/06/27 12:14:41 INFO input.FileInputFormat: Total input paths to process : 1
22/06/27 12:14:41 INFO mapreduce.JobSubmitter: number of splits:1
```



```
hduser@bmsce-Precision-T1700:~$ hadoop fs -ls /output_Topn
Found 2 items
-rw-r--r--   1 hduser supergroup          0 2022-06-27 12:14 /output_Topn/_SUCCESS
-rw-r--r--   1 hduser supergroup         43 2022-06-27 12:14 /output_Topn/part-r-00000
hduser@bmsce-Precision-T1700:~$ hadoop fs -cat /output_Topn/part-r-00000
bms     2
college 2
computer        1
law     1
science 1
```

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

DeptEmpStrength.txt

Dept_ID	Total_Employee
A11	50
B12	100
C13	250

DeptName.txt

Dept_ID	Dept_Name
A11	Finance
B12	HR
C13	Manufacturing

Driver:

```
package MapReduceJoin;

import org.apache.hadoop.conf.Configured;
import org.apache.hadoop.fs.Path;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapred.*;
import org.apache.hadoop.mapred.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,
DeptNameMapper.class);
        MultipleInputs.addInputPath(conf, BInputPath, TextInputFormat.class,
DeptEmpStrengthMapper.class);
```

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

        conf.setPartitionerClass(KeyPartitioner.class);
        conf.setOutputValueGroupingComparator(TextPair.FirstComparator.class);

        conf.setMapOutputKeyClass(TextPair.class);

        conf.setReducerClass(JoinReducer.class);

        conf.setOutputKeyClass(Text.class);

        JobClient.runJob(conf);

        return 0;
    }

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

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

Mapper:

DeptEmpStrengthMapper.java

```
package MapReduceJoin;


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 DeptEmpStrengthMapper 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]));
```

```
    }
}
```

DeptNameMapper.java

```
package MapReduceJoin;

import java.io.IOException;

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

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

    @Override
    public void map(LongWritable key, Text value, OutputCollector<TextPair, Text> output,
Reporter reporter)
                throws IOException
    {
        String valueString = value.toString();
        String[] SingleNodeData = valueString.split("\t");
        output.collect(new TextPair(SingleNodeData[0], "0"), new
Text(SingleNodeData[1]));
    }
}
```

Reducer:

```
package MapReduceJoin;
```

```
import java.io.IOException;

import java.util.Iterator;


import org.apache.hadoop.io.Text;

import org.apache.hadoop.mapred.*;


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


    @Override

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

              throws IOException

    {


        Text nodeId = new Text(values.next());

        while (values.hasNext()) {

            Text node = values.next();

            Text outValue = new Text(nodeId.toString() + "\t\t" + node.toString());

            output.collect(key.getFirst(), outValue);

        }

    }

}
```

Jar link:

https://github.com/amitkumar70512/BDA_LAB/blob/main/Lab8/MapReduceJoin/MapReduceJoin.jar

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

scala program to print "Hello World".

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

output

```
Hello World!
```

Word count using scala

we find and display the number of occurrences of each word.

```
$ hdfs dfs –mkdir /spark
$ hdfs dfs –put /home/amit/sparkdata.txt /spark
```

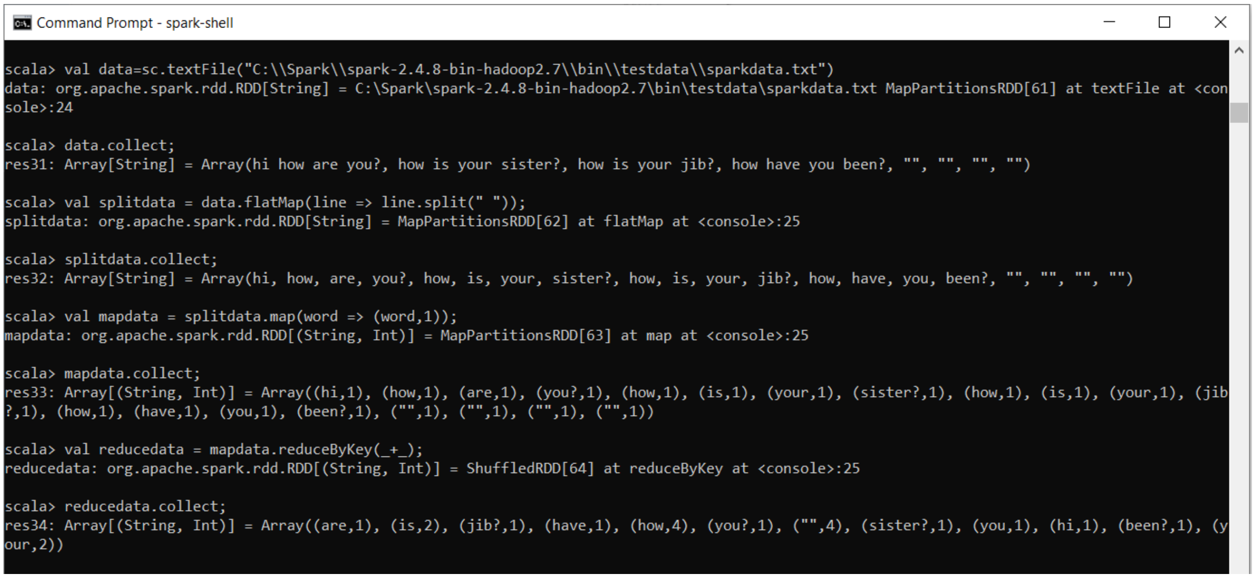
```
scala> val data=sc.textFile("sparkdata.txt") .
```

```
scala> val splitdata = data.flatMap(line => line.split(" "));
```

```
scala> splitdata.collect;
```

```
scala> val mapdata = splitdata.map(word => (word,1));
```

scala> val reducedata = mapdata.reduceByKey(+);

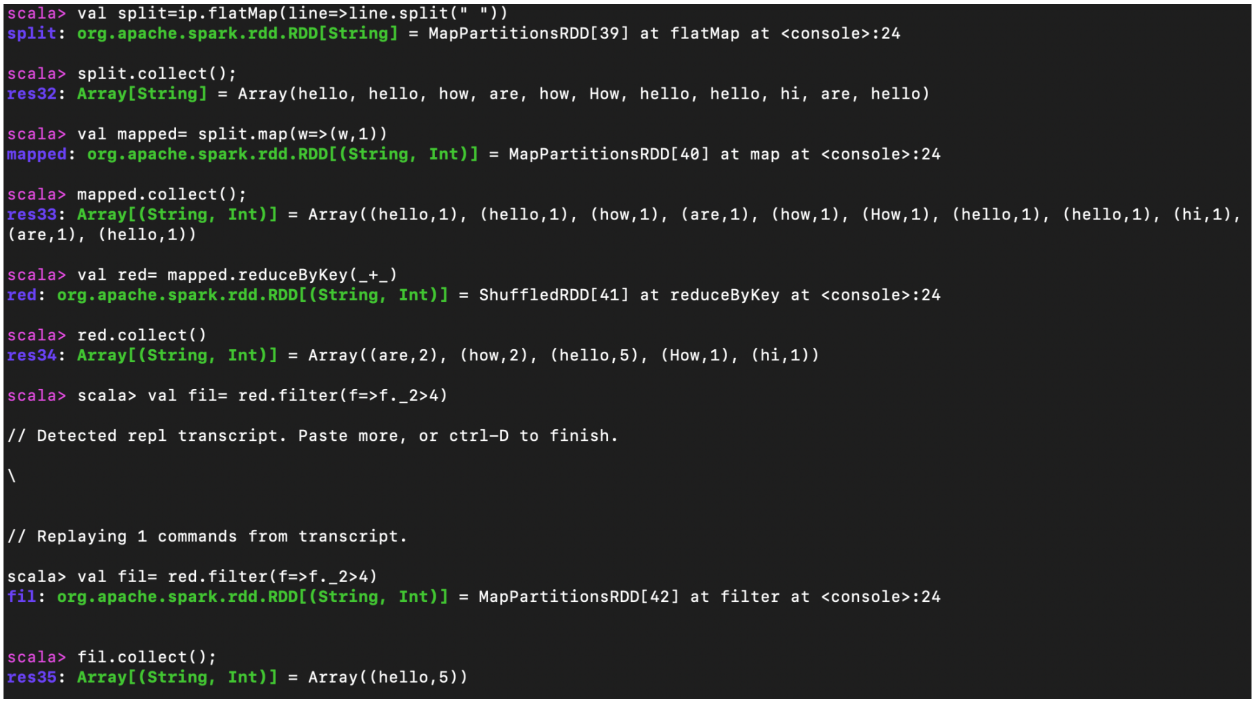


```
scala> val data=sc.textFile("C:\\Spark\\spark-2.4.8-bin-hadoop2.7\\bin\\testdata\\sparkdata.txt")
data: org.apache.spark.rdd.RDD[String] = C:\Spark\spark-2.4.8-bin-hadoop2.7\bin\testdata\sparkdata.txt MapPartitionsRDD[61] at textFile at <con
sole>:24

scala> data.collect;
res31: Array[String] = Array(hi how are you?, how is your sister?, how is your jib?, how have you been?, "", "", "", "")

scala> val splitdata = data.flatMap(line => line.split(" "));
splitdata: org.apache.spark.rdd.RDD[String] = MapPartitionsRDD[62] at flatMap at <console>:25

scala> splitdata.collect;
res32: Array[String] = Array(hi, how, are, you?, how, is, your, sister?, how, is, your, jib?, how, have, you, been?, "", "", "", "")

scala> val mapdata = splitdata.map(word => (word,1));
mapdata: org.apache.spark.rdd.RDD[(String, Int)] = MapPartitionsRDD[63] at map at <console>:25

scala> mapdata.collect;
res33: Array[(String, Int)] = Array((hi,1), (how,1), (are,1), (you?,1), (how,1), (is,1), (your,1), (sister?,1), (how,1), (is,1), (your,1), (jib
?,1), (how,1), (have,1), (you,1), (been?,1), ("",1), ("",1), ("",1), ("",1))

scala> val reducedata = mapdata.reduceByKey(_+_);
reducedata: org.apache.spark.rdd.RDD[(String, Int)] = ShuffledRDD[64] at reduceByKey at <console>:25

scala> reducedata.collect;
res34: Array[(String, Int)] = Array((are,1), (is,2), (jib?,1), (have,1), (how,4), (you?,1), ("",4), (sister?,1), (you,1), (hi,1), (been?,1), (y
our,2))
```

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

```
scala> val ip=sc.textFile("/Users/mallikaprasad/Desktop/sample1.txt");

ip: org.apache.spark.rdd.RDD[String] = /Users/mallikaprasad/Desktop/sample1.txt
MapPartitionsRDD[38] at textFile at <console>:24


scala> ip.collect();

res31: Array[String] = Array(hello hello how are how How hello hello hi are hello)


scala> val split=ip.flatMap(line=>line.split(" "))

split: org.apache.spark.rdd.RDD[String] = MapPartitionsRDD[39] at flatMap at <console>:24


scala> split.collect();

res32: Array[String] = Array(hello, hello, how, are, how, How, hello, hello, hi, are, hello)


scala> val mapped= split.map(w=>(w,1))
```

```
mapped: org.apache.spark.rdd.RDD[(String, Int)] = MapPartitionsRDD[40] at map at
<console>:24

scala> mapped.collect();

res33: Array[(String, Int)] = Array((hello,1), (hello,1), (how,1), (are,1), (how,1), (How,1), (hello,1),
(hello,1), (hi,1), (are,1), (hello,1))

scala> val red= mapped.reduceByKey(_+_)

red: org.apache.spark.rdd.RDD[(String, Int)] = ShuffledRDD[41] at reduceByKey at <console>:24

scala> red.collect()

res34: Array[(String, Int)] = Array((are,2), (how,2), (hello,5), (How,1), (hi,1))

scala> scala> val fil= red.filter(f=>f._2>4) scala> val fil= red.filter(f=>f._2>4)

fil: org.apache.spark.rdd.RDD[(String, Int)] = MapPartitionsRDD[42] at filter at <console>:24

scala> fil.collect();

res35: Array[(String, Int)] = Array((hello,5))
```



```
scala> val split=ip.flatMap(line=>line.split(" "))
split: org.apache.spark.rdd.RDD[String] = MapPartitionsRDD[39] at flatMap at <console>:24

scala> split.collect();
res32: Array[String] = Array(hello, hello, how, are, how, How, hello, hello, hi, are, hello)

scala> val mapped= split.map(w=>(w,1))
mapped: org.apache.spark.rdd.RDD[(String, Int)] = MapPartitionsRDD[40] at map at <console>:24

scala> mapped.collect();
res33: Array[(String, Int)] = Array((hello,1), (hello,1), (how,1), (are,1), (how,1), (How,1), (hello,1), (hello,1), (hi,1),
(are,1), (hello,1))

scala> val red= mapped.reduceByKey(_+_)
red: org.apache.spark.rdd.RDD[(String, Int)] = ShuffledRDD[41] at reduceByKey at <console>:24

scala> red.collect()
res34: Array[(String, Int)] = Array((are,2), (how,2), (hello,5), (How,1), (hi,1))

scala> scala> val fil= red.filter(f=>f._2>4)

// Detected repl transcript. Paste more, or ctrl-D to finish.

\


// Replaying 1 commands from transcript.

scala> val fil= red.filter(f=>f._2>4)
fil: org.apache.spark.rdd.RDD[(String, Int)] = MapPartitionsRDD[42] at filter at <console>:24


scala> fil.collect();
res35: Array[(String, Int)] = Array((hello,5))
```