VISVESVARAYA TECHNOLOGICAL UNIVERSITY

"JnanaSangama", Belgaum -590014, Karnataka.

LAB REPORT
on

BIG DATA ANALYTICS
(20CS6PEBDA)

Submitted by

RAHUL PRAKASHA (1BM18CS078)

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**" was carried out by **RAHUL PRAKASHA (1BM18CS078),** who is bona fide 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 the course **BIG DATA ANALYTICS (20CS6PEBDA)** work prescribed for the said degree.

Name of the Lab-In charge
Designation
Department of CSE
BMSCE, Bengaluru

ANTARA ROY CHOUDHURY
Assistant Professor
Department of CSE
BMSCE, Bengaluru

`

INDEX SHEET

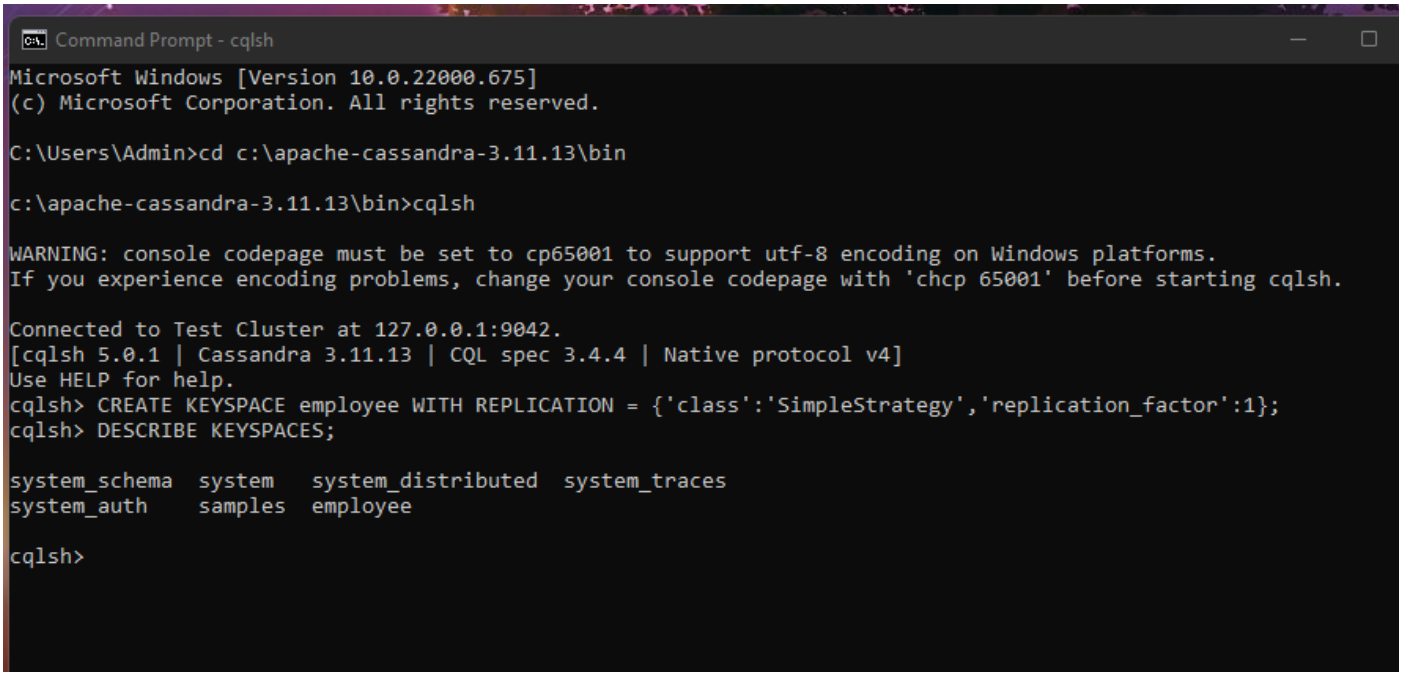
Course Outcome

CO1	Apply the concept of NoSQL, Hadoop or Spark for a given task
CO2	Analyze the Big Data and obtain insight using data analytics mechanisms.
CO3	Design and implement Big data applications by applying NoSQL, Hadoop or Spark

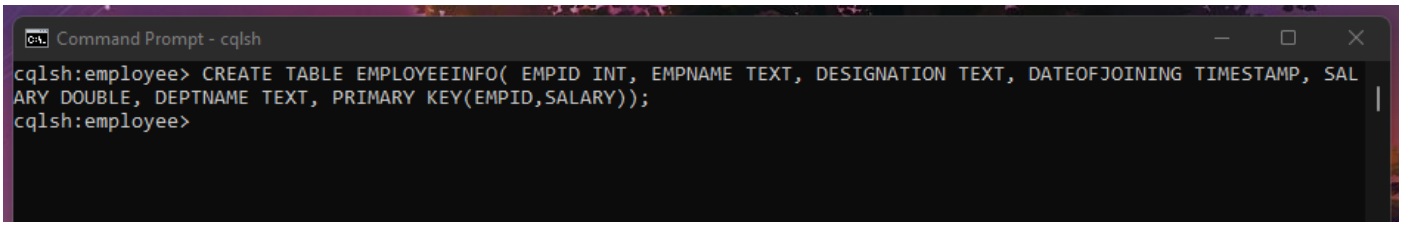
1. Cassandra Lab Program 1: -

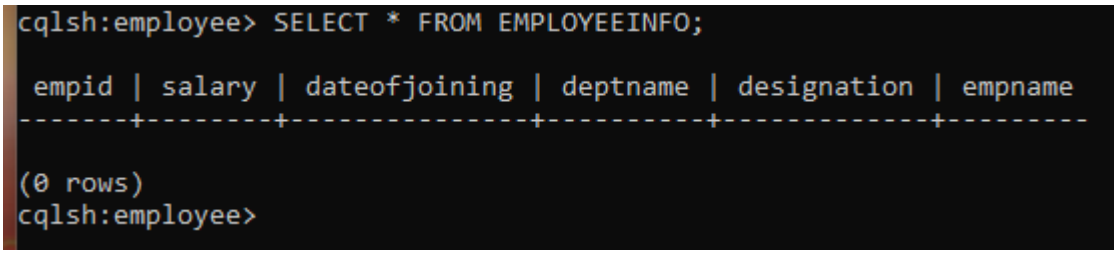
Perform the following DB operations using Cassandra.

1. Create a key space by name Employee



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





3. Insert the values into the table in batch

```
Command Prompt - cqlsh
cqlsh:employee> BEGIN BATCH
           ... INSERT INTO EMPLOYEEINFO (EMPID, EMPNAME, DESIGNATION, DATEOFJOINING, SALARY, DEPTNAME)
           ... VALUES(1,'LOKESH','ASSISTANT MANAGER', '2005-04-6', 50000, 'MARKETING')
           ... INSERT INTO EMPLOYEEINFO (EMPID, EMPNAME, DESIGNATION, DATEOFJOINING, SALARY, DEPTNAME)
           ... VALUES(2,'DHEERAJ','ASSISTANT MANAGER', '2013-11-10', 30000, 'LOGISTICS')
           ... INSERT INTO EMPLOYEEINFO (EMPID, EMPNAME, DESIGNATION, DATEOFJOINING, SALARY, DEPTNAME)
           ... VALUES(3,'CHIRAG','ASSISTANT MANAGER', '2011-07-1', 115000, 'SALES')
           ... INSERT INTO EMPLOYEEINFO (EMPID, EMPNAME, DESIGNATION, DATEOFJOINING, SALARY, DEPTNAME)
           ... VALUES(4,'DHANUSH','ASSISTANT MANAGER', '2010-04-26', 75000, 'MARKETING')
           ...  INSERT INTO EMPLOYEEINFO (EMPID, EMPNAME, DESIGNATION, DATEOFJOINING, SALARY, DEPTNAME)
           ... VALUES(5,'ESHA','ASSISTANT MANAGER', '2010-04-26', 85000, 'TECHNICAL')
           ... INSERT INTO EMPLOYEEINFO (EMPID, EMPNAME, DESIGNATION, DATEOFJOINING, SALARY, DEPTNAME)
           ... VALUES(6,'FARHAN','MANAGER', '2010-04-26', 95000, 'TECHNICAL')
           ... INSERT INTO EMPLOYEEINFO (EMPID, EMPNAME, DESIGNATION, DATEOFJOINING, SALARY, DEPTNAME)
           ... VALUES(7,'JIMMY','MANAGER', '2010-04-26', 95000, 'PR')
           ... INSERT INTO EMPLOYEEINFO (EMPID, EMPNAME, DESIGNATION, DATEOFJOINING, SALARY, DEPTNAME)
           ... VALUES(121,'HARRY','REGIONAL MANAGER', '2010-04-26', 99000, 'MANAGEMENT')
           ... APPLY BATCH;
```

```
cqlsh:employee>  SELECT * FROM EMPLOYEEINFO;

 empid | salary   | dateofjoining                   | deptname   | designation       | empname
-------+----------+---------------------------------+------------+-------------------+---------
     5 |    85000 | 2010-04-25 18:30:00.000000+0000 |  TECHNICAL | ASSISTANT MANAGER |    ESHA
     1 |    50000 | 2005-04-05 18:30:00.000000+0000 |  MARKETING | ASSISTANT MANAGER |  LOKESH
     2 |    30000 | 2013-11-09 18:30:00.000000+0000 |  LOGISTICS | ASSISTANT MANAGER | DHEERAJ
     4 |    75000 | 2010-04-25 18:30:00.000000+0000 |  MARKETING | ASSISTANT MANAGER | DHANUSH
   121 |    99000 | 2010-04-25 18:30:00.000000+0000 | MANAGEMENT |  REGIONAL MANAGER |   HARRY
     7 |    95000 | 2010-04-25 18:30:00.000000+0000 |         PR |           MANAGER |   JIMMY
     6 |    95000 | 2010-04-25 18:30:00.000000+0000 |  TECHNICAL |           MANAGER |  FARHAN
     3 | 1.15e+05 | 2011-06-30 18:30:00.000000+0000 |      SALES | ASSISTANT MANAGER |  CHIRAG

(8 rows)
cqlsh:employee>
```

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

```
cqlsh:employee> UPDATE EMPLOYEEINFO SET EMPNAME='HARRY', DEPTNAME='MANAGEMENT' WHERE EMPID=121 AND SALARY=99000;
cqlsh:employee>  SELECT * FROM EMPLOYEEINFO;

 empid | salary   | dateofjoining                   | deptname   | designation       | empname
-------+----------+---------------------------------+------------+-------------------+---------
     5 |    85000 | 2010-04-25 18:30:00.000000+0000 |  TECHNICAL | ASSISTANT MANAGER |    ESHA
     1 |    50000 | 2005-04-05 18:30:00.000000+0000 |  MARKETING | ASSISTANT MANAGER |  LOKESH
     2 |    30000 | 2013-11-09 18:30:00.000000+0000 |  LOGISTICS | ASSISTANT MANAGER | DHEERAJ
     4 |    75000 | 2010-04-25 18:30:00.000000+0000 |  MARKETING | ASSISTANT MANAGER | DHANUSH
   121 |    99000 | 2010-04-25 18:30:00.000000+0000 | MANAGEMENT |  REGIONAL MANAGER |   HARRY
     7 |    95000 | 2010-04-25 18:30:00.000000+0000 |         PR |           MANAGER |   JIMMY
     6 |    95000 | 2010-04-25 18:30:00.000000+0000 |  TECHNICAL |           MANAGER |  FARHAN
     3 | 1.15e+05 | 2011-06-30 18:30:00.000000+0000 |      SALES | ASSISTANT MANAGER |  CHIRAG

(8 rows)
cqlsh:employee>
```

5. Sort the details of Employee records based on salary (Note:- cql>PAGING OFF)

```
cqlsh:employee> select * from EMPLOYEEINFO where empid IN(1,2,3,4,5,6,7) ORDER BY salary DESC allow filtering;

 empid | salary   | dateofjoining                   | deptname  | designation       | empname
-------+----------+---------------------------------+-----------+-------------------+---------
     3 | 1.15e+05 | 2011-06-30 18:30:00.000000+0000 |     SALES | ASSISTANT MANAGER |  CHIRAG
     6 |    95000 | 2010-04-25 18:30:00.000000+0000 | TECHNICAL |           MANAGER |  FARHAN
     7 |    95000 | 2010-04-25 18:30:00.000000+0000 |        PR |           MANAGER |   JIMMY
     5 |    85000 | 2010-04-25 18:30:00.000000+0000 | TECHNICAL | ASSISTANT MANAGER |    ESHA
     4 |    75000 | 2010-04-25 18:30:00.000000+0000 | MARKETING | ASSISTANT MANAGER | DHANUSH
     1 |    50000 | 2005-04-05 18:30:00.000000+0000 | MARKETING | ASSISTANT MANAGER |  LOKESH
     2 |    30000 | 2013-11-09 18:30:00.000000+0000 | LOGISTICS | ASSISTANT MANAGER | DHEERAJ

(7 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.

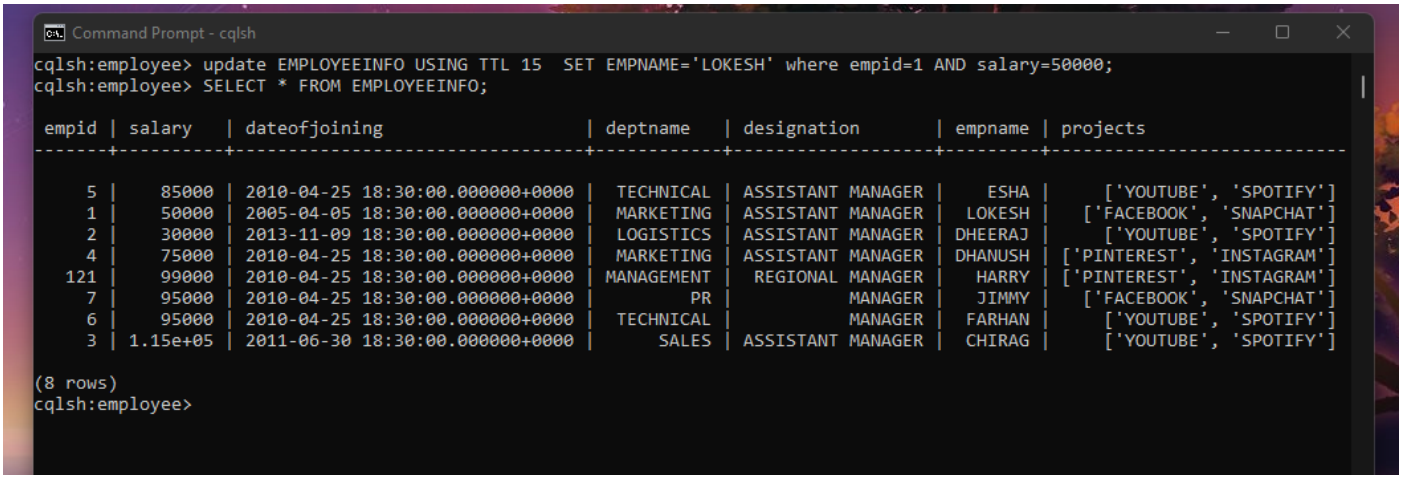
```
(7 rows)
cqlsh:employee> ALTER TABLE EMPLOYEEINFO ADD PROJECTS LIST<TEXT>;
cqlsh:employee> SELECT * FROM EMPLOYEEINFO;

 empid | salary   | dateofjoining                   | deptname    | designation       | empname | projects
-------+----------+---------------------------------+-------------+-------------------+---------+----------
     5 |    85000 | 2010-04-25 18:30:00.000000+0000 |   TECHNICAL | ASSISTANT MANAGER |    ESHA |     null
     1 |    50000 | 2005-04-05 18:30:00.000000+0000 |   MARKETING | ASSISTANT MANAGER |  LOKESH |     null
     2 |    30000 | 2013-11-09 18:30:00.000000+0000 |   LOGISTICS | ASSISTANT MANAGER | DHEERAJ |     null
     4 |    75000 | 2010-04-25 18:30:00.000000+0000 |   MARKETING | ASSISTANT MANAGER | DHANUSH |     null
   121 |    99000 | 2010-04-25 18:30:00.000000+0000 | MANAGEMENT |  REGIONAL MANAGER |   HARRY |     null
     7 |    95000 | 2010-04-25 18:30:00.000000+0000 |          PR |           MANAGER |   JIMMY |     null
     6 |    95000 | 2010-04-25 18:30:00.000000+0000 |   TECHNICAL |           MANAGER |  FARHAN |     null
     3 | 1.15e+05 | 2011-06-30 18:30:00.000000+0000 |       SALES | ASSISTANT MANAGER |  CHIRAG |     null

(8 rows)
cqlsh:employee>
```

7. Update the altered table to add project names.

```
Command Prompt - cqlsh
cqlsh:employee> UPDATE EMPLOYEEINFO SET PROJECTS=['FACEBOOK','SNAPCHAT'] WHERE EMPID=1 AND SALARY=50000;
cqlsh:employee> UPDATE EMPLOYEEINFO SET PROJECTS=['FACEBOOK','SNAPCHAT'] WHERE EMPID=7 AND SALARY=95000;
cqlsh:employee>  UPDATE EMPLOYEEINFO SET PROJECTS=['PINTEREST','INSTAGRAM'] WHERE EMPID=121 AND SALARY=99000;
cqlsh:employee> UPDATE EMPLOYEEINFO SET PROJECTS=['PINTEREST','INSTAGRAM'] WHERE EMPID=4 AND SALARY=75000;
cqlsh:employee> UPDATE EMPLOYEEINFO SET PROJECTS=['YOUTUBE','SPOTIFY'] WHERE EMPID=2 AND SALARY=30000;
cqlsh:employee> UPDATE EMPLOYEEINFO SET PROJECTS=['YOUTUBE','SPOTIFY'] WHERE EMPID=3 AND SALARY=115000;
cqlsh:employee> UPDATE EMPLOYEEINFO SET PROJECTS=['YOUTUBE','SPOTIFY'] WHERE EMPID=6 AND SALARY=95000;
cqlsh:employee> UPDATE EMPLOYEEINFO SET PROJECTS=['YOUTUBE','SPOTIFY'] WHERE EMPID=5 AND SALARY=85000;
cqlsh:employee> SELECT * FROM EMPLOYEEINFO;

 empid | salary   | dateofjoining                   | deptname    | designation       | empname | projects
-------+----------+---------------------------------+-------------+-------------------+---------+----------------------------
     5 |    85000 | 2010-04-25 18:30:00.000000+0000 |   TECHNICAL | ASSISTANT MANAGER |    ESHA |     ['YOUTUBE', 'SPOTIFY']
     1 |    50000 | 2005-04-05 18:30:00.000000+0000 |   MARKETING | ASSISTANT MANAGER |  LOKESH |   ['FACEBOOK', 'SNAPCHAT']
     2 |    30000 | 2013-11-09 18:30:00.000000+0000 |   LOGISTICS | ASSISTANT MANAGER | DHEERAJ |     ['YOUTUBE', 'SPOTIFY']
     4 |    75000 | 2010-04-25 18:30:00.000000+0000 |   MARKETING | ASSISTANT MANAGER | DHANUSH | ['PINTEREST', 'INSTAGRAM']
   121 |    99000 | 2010-04-25 18:30:00.000000+0000 | MANAGEMENT |  REGIONAL MANAGER |   HARRY | ['PINTEREST', 'INSTAGRAM']
     7 |    95000 | 2010-04-25 18:30:00.000000+0000 |          PR |           MANAGER |   JIMMY |   ['FACEBOOK', 'SNAPCHAT']
     6 |    95000 | 2010-04-25 18:30:00.000000+0000 |   TECHNICAL |           MANAGER |  FARHAN |     ['YOUTUBE', 'SPOTIFY']
     3 | 1.15e+05 | 2011-06-30 18:30:00.000000+0000 |       SALES | ASSISTANT MANAGER |  CHIRAG |     ['YOUTUBE', 'SPOTIFY']

(8 rows)
cqlsh:employee>
```

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

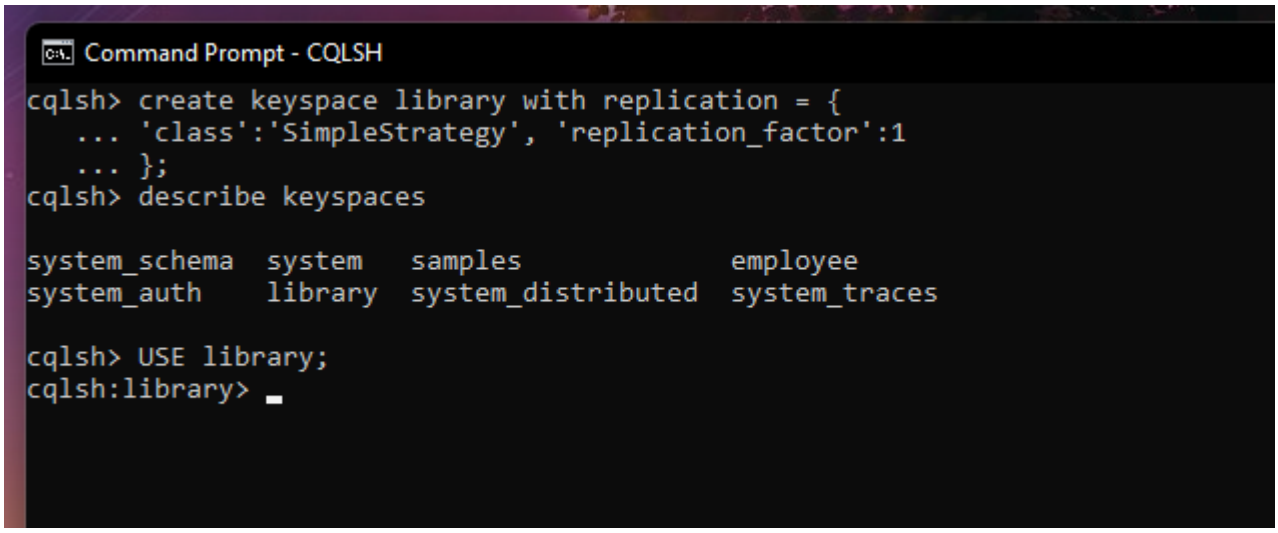
//BEFORE 15 seconds



2. Cassandra Lab Program 2: -

Perform the following DB operations using Cassandra.

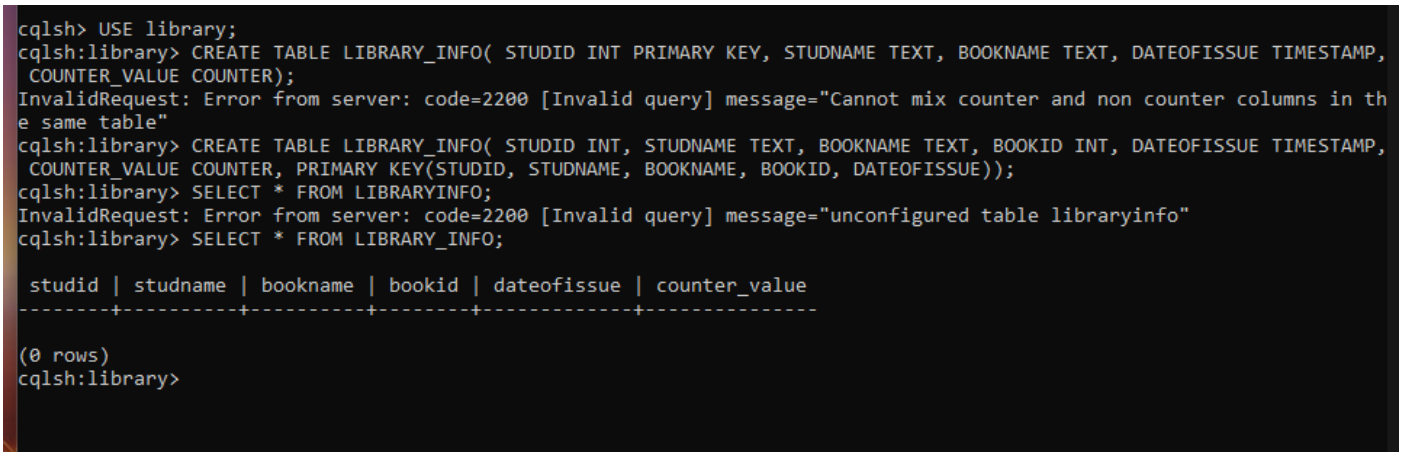
1.Create a key space by name Library



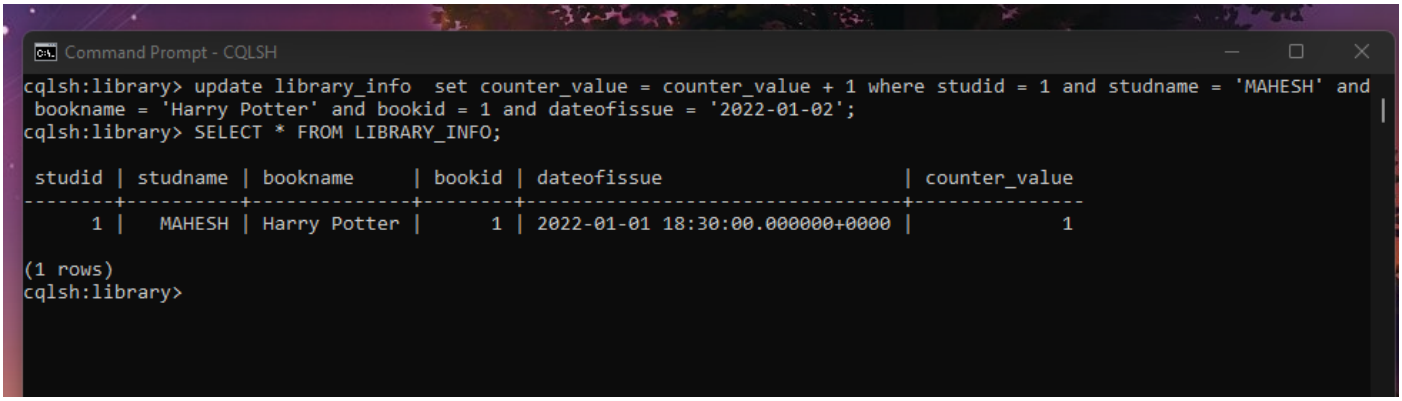
2. Create a column family by name Library-Info with attributes Stud_Id Primary Key,

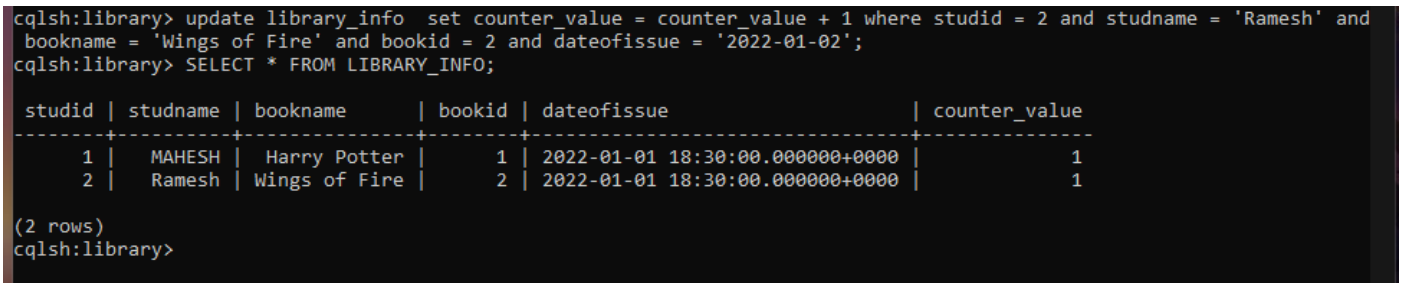
Counter_value of type Counter,

Stud_Name, Book-Name, Book-Id, Date_of_issue

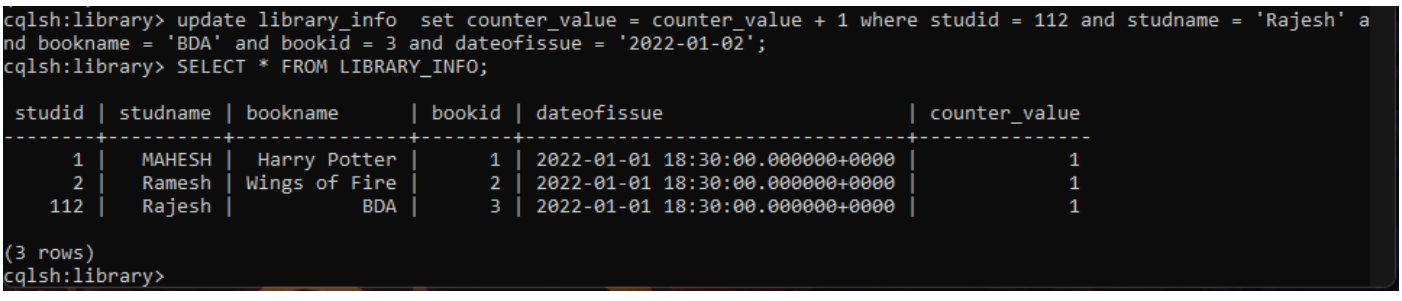


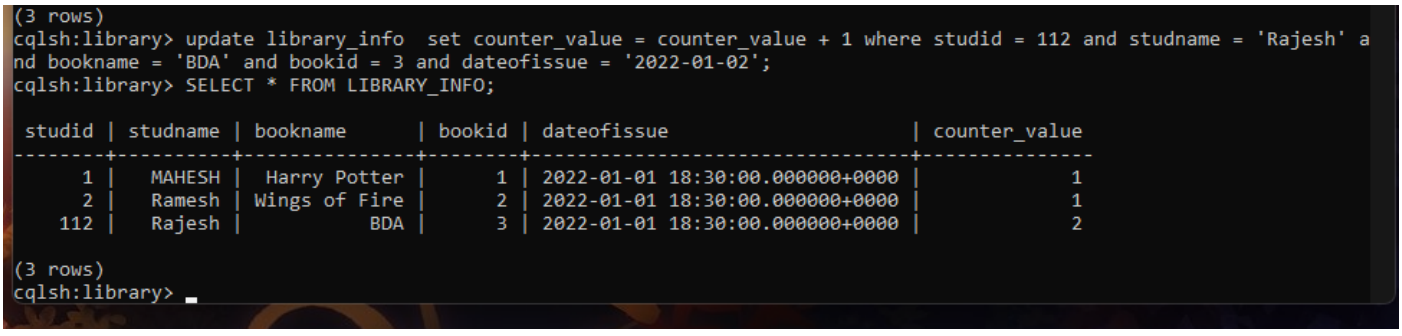
3.Insert the values into the table in batch





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





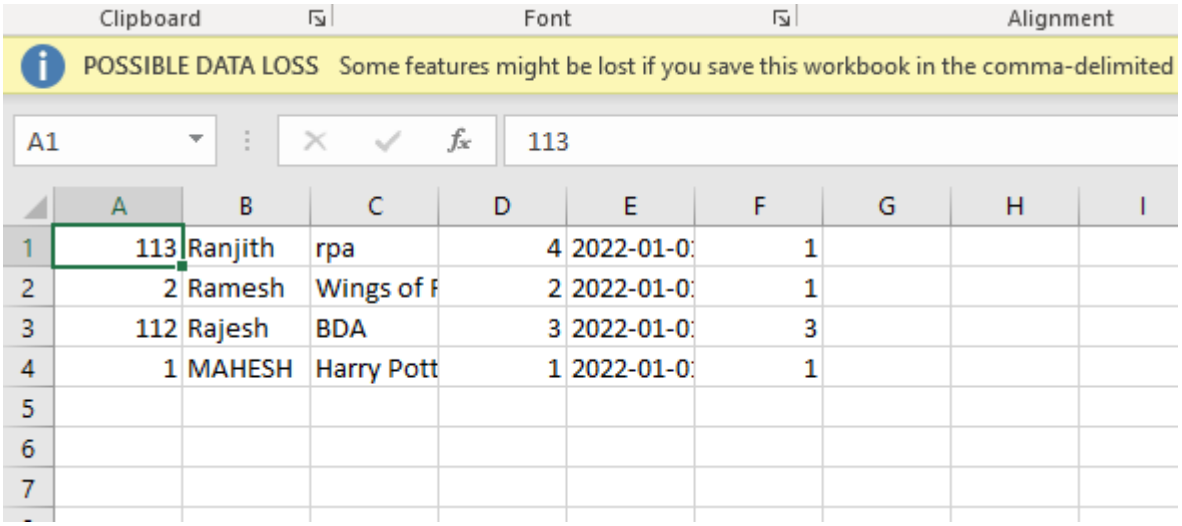
```
 studid | studname | bookname      | bookid | dateofissue                     | counter_value
--------+----------+---------------+--------+---------------------------------+---------------
    113 |  Ranjith |           rpa |      4 | 2022-01-01 18:30:00.000000+0000 |             1
      1 |   MAHESH |  Harry Potter |      1 | 2022-01-01 18:30:00.000000+0000 |             1
      2 |   Ramesh | Wings of Fire |      2 | 2022-01-01 18:30:00.000000+0000 |             1
    112 |   Rajesh |           BDA |      3 | 2022-01-01 18:30:00.000000+0000 |             3

(4 rows)
```

5. Write a query to show that a student with id 112 has taken a book “BDA” 3 times.

```
Command Prompt - CQLSH
cqlsh:library> select * from library_info where studid = 112;

 studid | studname | bookname | bookid | dateofissue                     | counter_value
--------+----------+----------+--------+---------------------------------+---------------
    112 |   Rajesh |      BDA |      3 | 2022-01-01 18:30:00.000000+0000 |             3

(1 rows)
cqlsh:library>
```

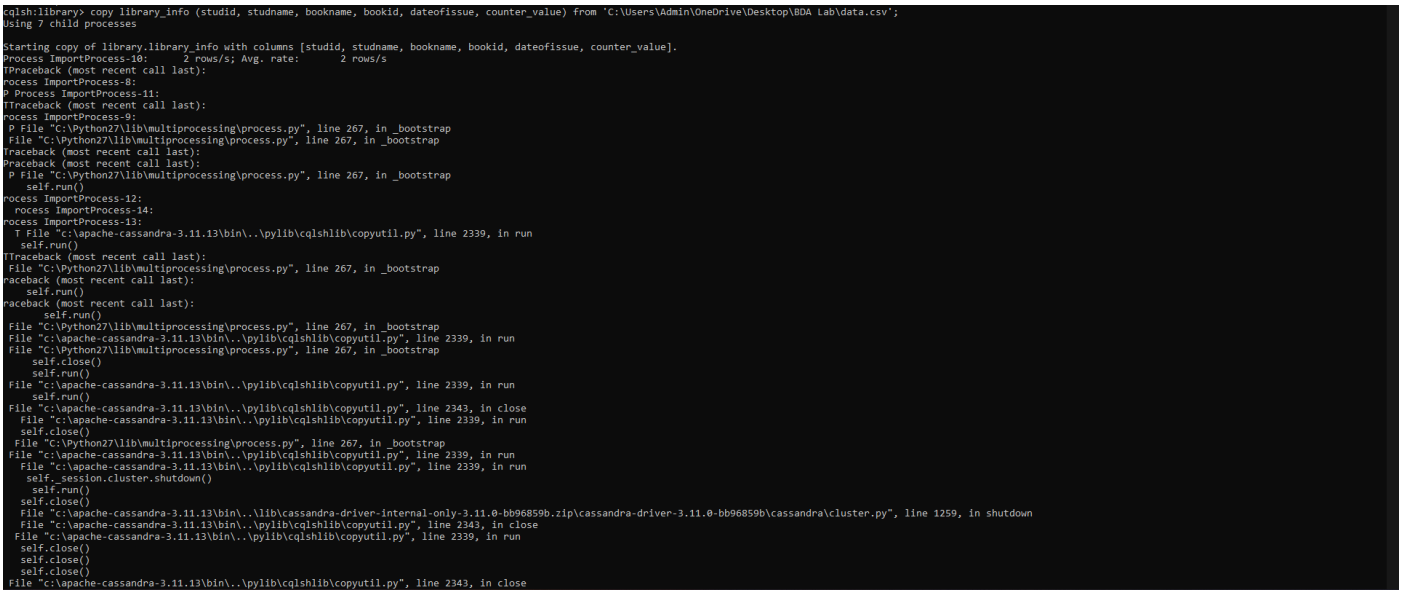
6. Export the created column to a csv file

```
cqlsh:library> copy library_info (studid, studname, bookname, bookid, dateofissue, counter_value) to 'C:\Users\Admin\O
neDrive\Desktop\BDA Lab\data.csv';
Using 7 child processes

Starting copy of library.library_info with columns [studid, studname, bookname, bookid, dateofissue, counter_value].
Processed: 4 rows; Rate:       2 rows/s; Avg. rate:       1 rows/s
4 rows exported to 1 files in 3.004 seconds.
cqlsh:library>
```



	A	B	C	D	E	F	G	H	I
1	113	Ranjith	rpa	4	2022-01-0	1			
2	2	Ramesh	Wings of F	2	2022-01-0	1			
3	112	Rajesh	BDA	3	2022-01-0	3			
4	1	MAHESH	Harry Pott	1	2022-01-0	1			
5									
6									
7									

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



```
 File "c:\apache-cassandra-3.11.13\bin\..\lib\cassandra-driver-internal-only-3.11.0-bb96859b.zip\cassandra-driver-3.11.0-bb96859b\cassandra\io\asyncorereactor.py", line 373, in close
 File "c:\apache-cassandra-3.11.13\bin\..\lib\cassandra-driver-internal-only-3.11.0-bb96859b.zip\cassandra-driver-3.11.0-bb96859b\cassandra\io\asyncorereactor.py", line 335, in create_timer
       File "c:\apache-cassandra-3.11.13\bin\..\lib\cassandra-driver-internal-only-3.11.0-bb96859b.zip\cassandra-driver-3.11.0-bb96859b\cassandra\io\asyncorereactor.py", line 373, in close
   self._connection.close()
 File "c:\apache-cassandra-3.11.13\bin\..\lib\cassandra-driver-internal-only-3.11.0-bb96859b.zip\cassandra-driver-3.11.0-bb96859b\cassandra\io\asyncorereactor.py", line 335, in create_timer
   self._connection.close()
    AsyncoreConnection.create_timer(0, partial(asyncore.dispatcher.close, self))
   File "c:\apache-cassandra-3.11.13\bin\..\lib\cassandra-driver-internal-only-3.11.0-bb96859b.zip\cassandra-driver-3.11.0-bb96859b\cassandra\io\asyncorereactor.py", line 373, in close
    cls._loop.add_timer(timer)
 File "c:\apache-cassandra-3.11.13\bin\..\lib\cassandra-driver-internal-only-3.11.0-bb96859b.zip\cassandra-driver-3.11.0-bb96859b\cassandra\io\asyncorereactor.py", line 373, in close
   cls._loop.add_timer(timer)
   AsyncoreConnection.create_timer(0, partial(asyncore.dispatcher.close, self))
AA      AsyncoreConnection.create_timer(0, partial(asyncore.dispatcher.close, self))
  ttributeError: 'NoneType' object has no attribute 'add_timer'
  File "c:\apache-cassandra-3.11.13\bin\..\lib\cassandra-driver-internal-only-3.11.0-bb96859b.zip\cassandra-driver-3.11.0-bb96859b\cassandra\io\asyncorereactor.py", line 335, in create_timer
ttributeError: 'NoneType' object has no attribute 'add_timer'
 File "c:\apache-cassandra-3.11.13\bin\..\lib\cassandra-driver-internal-only-3.11.0-bb96859b.zip\cassandra-driver-3.11.0-bb96859b\cassandra\io\asyncorereactor.py", line 335, in create_timer
 File "c:\apache-cassandra-3.11.13\bin\..\lib\cassandra-driver-internal-only-3.11.0-bb96859b.zip\cassandra-driver-3.11.0-bb96859b\cassandra\io\asyncorereactor.py", line 335, in create_timer
   AsyncoreConnection.create_timer(0, partial(asyncore.dispatcher.close, self))
       cls._loop.add_timer(timer)
   cls._loop.add_timer(timer)
A File "c:\apache-cassandra-3.11.13\bin\..\lib\cassandra-driver-internal-only-3.11.0-bb96859b.zip\cassandra-driver-3.11.0-bb96859b\cassandra\io\asyncorereactor.py", line 335, in create_timer
ttributeError: 'NoneType' object has no attribute 'add_timer'
 A   cls._loop.add_timer(timer)
   cls._loop.add_timer(timer)
ttributeError: 'NoneType' object has no attribute 'add_timer'
AAttributeError: 'NoneType' object has no attribute 'add_timer'
ttributeError: 'NoneType' object has no attribute 'add_timer'
Processed: 4 rows; Rate:       1 rows/s; Avg. rate:       2 rows/s
4 rows imported from 1 files in 2.356 seconds (0 skipped).
cqlsh:library>
```

3.a. MongoDB Lab Program 1 (CRUD Demonstration): -

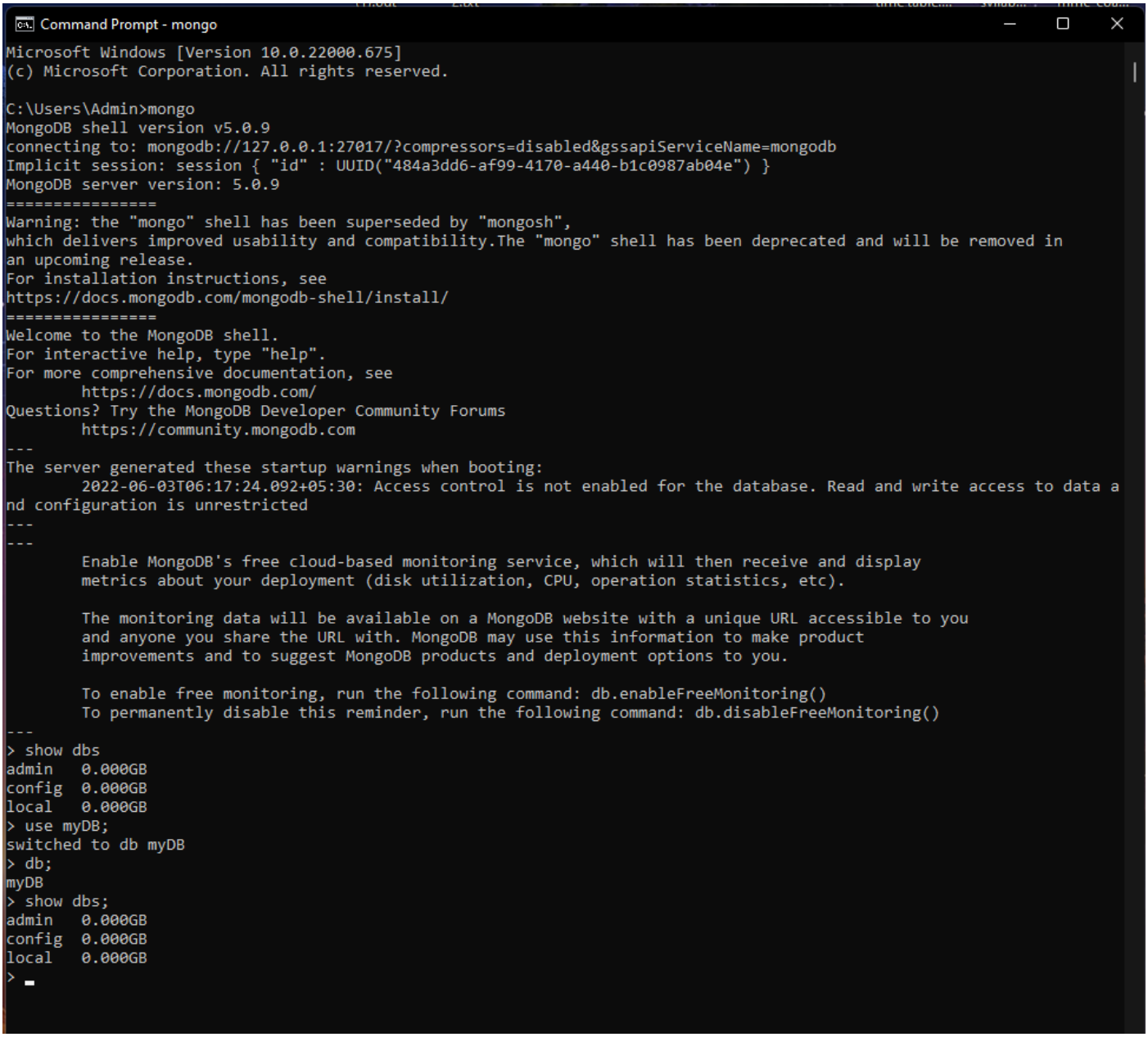
Execute the queries and upload a document with output.

I. CREATE DATABASE IN MONGODB.

use myDB;

db; (Confirm the existence of your database)

show dbs; (To list all databases)



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

1. To create a collection by the name “Student”. Let us take a look at the collection list

prior to the creation of the new collection “Student”.

db.createCollection(“Student”); => sql equivalent CREATE TABLE STUDENT(…);

2. To drop a collection by the name “Student”.

db.Student.drop();

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

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

urfing"});

4. Insert the document for “AryanDavid” in to the Students collection only if it does not

already exist in the collection. However, if it is already present in the collection, then

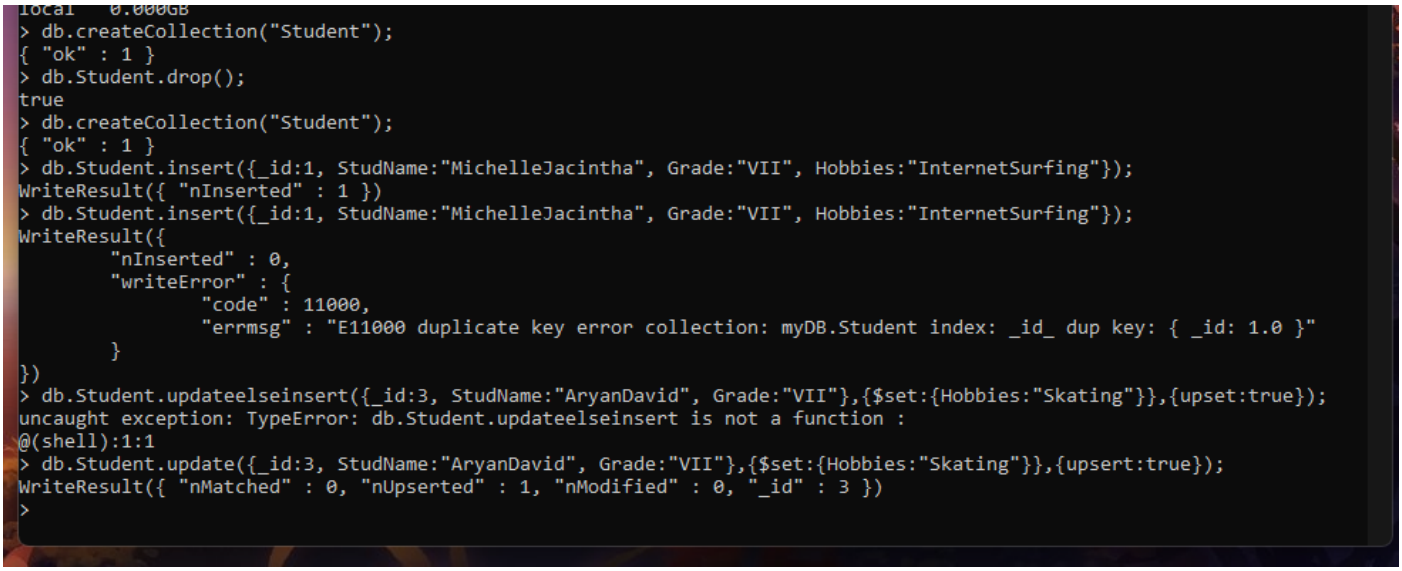
update the document with new values. (Update his Hobbies from “Skating” to “Chess”.

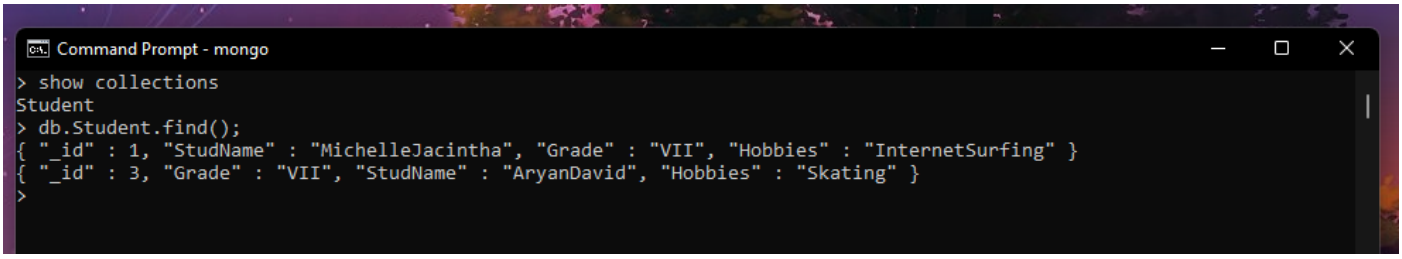
) Use “Update else insert” (if there is an existing document, it will attempt to update it,

if there is no existing document then it will insert it).

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

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



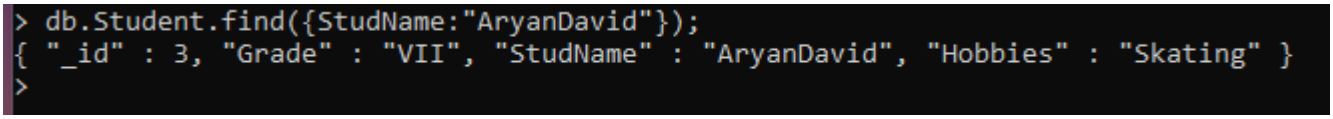


5. FIND METHOD

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

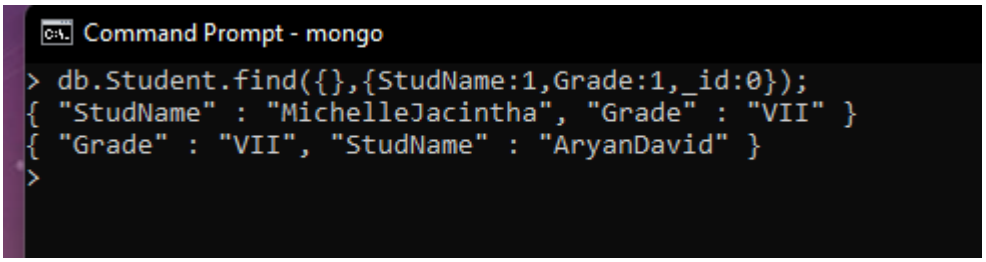
db.Student.find({StudName:"Aryan David"});

({cond..},{columns.. column:1, columnname:0})



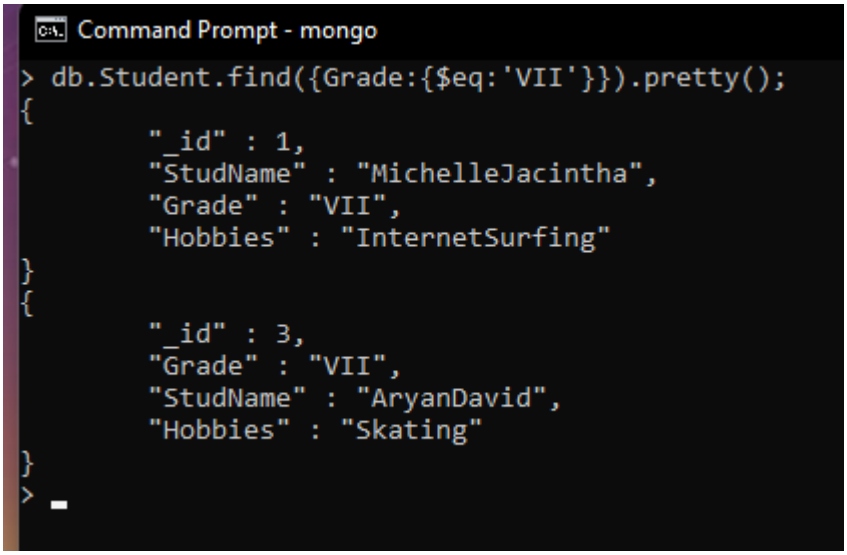
B. To display only the StudName and Grade from all the documents of the Students collection. The identifier_id should be suppressed and NOT displayed.

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



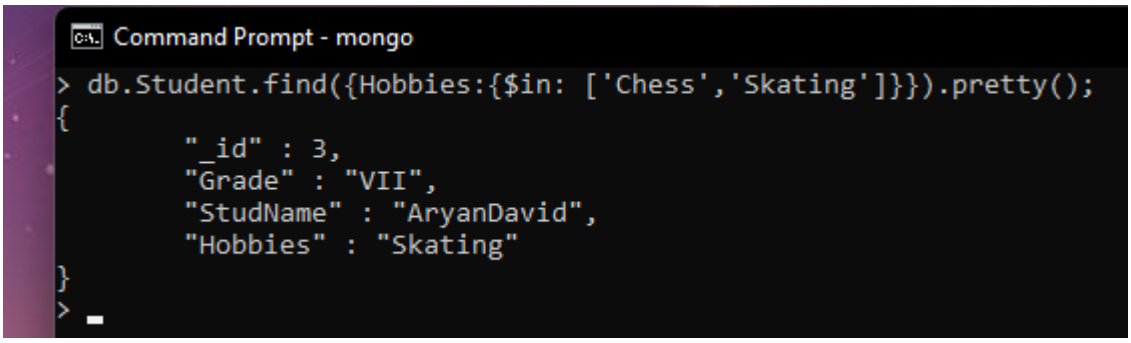
C. To find those documents where the Grade is set to ‘VII’

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



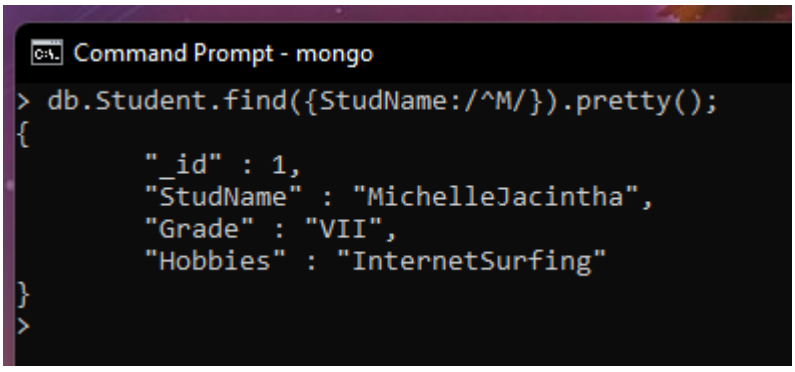
D. To find those documents from the Students collection where the Hobbies is set to either ‘Chess’ or is set to ‘Skating’.

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



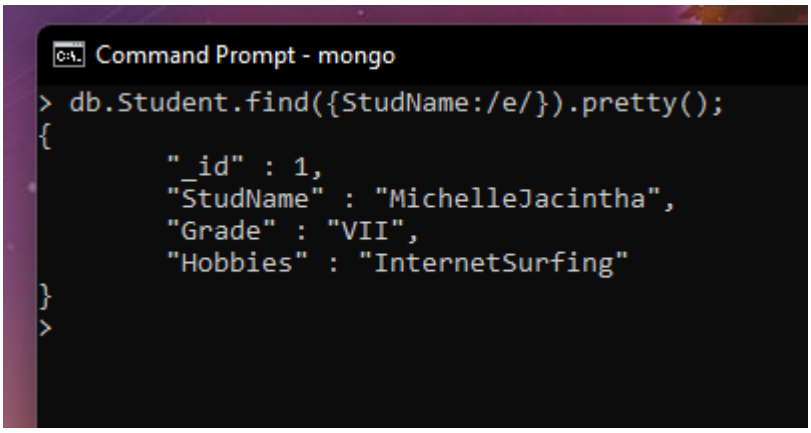
E. To find documents from the Students collection where the StudName begins with “M”.

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



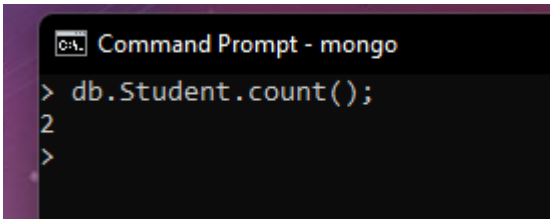
F. To find documents from the Students collection where the StudNamehas an “e” in any position.

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



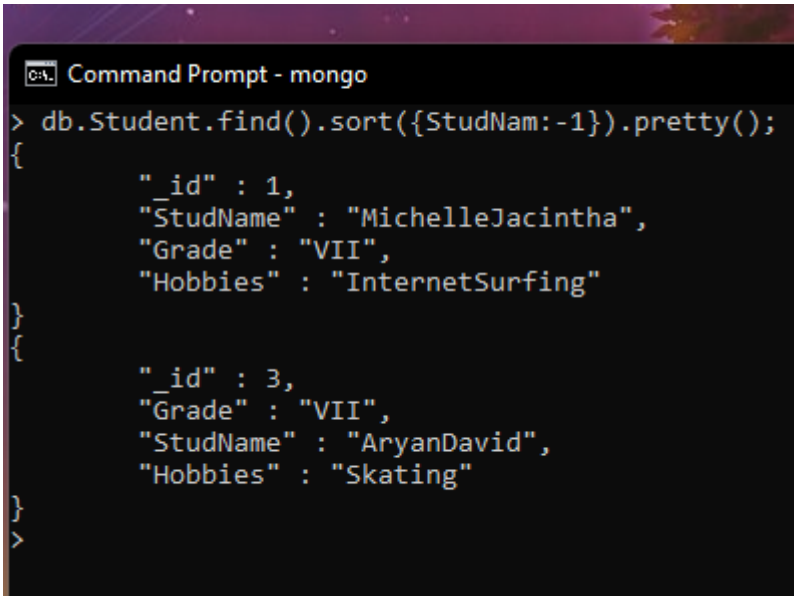
G. To find the number of documents in the Students collection.

db.Student.count();



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

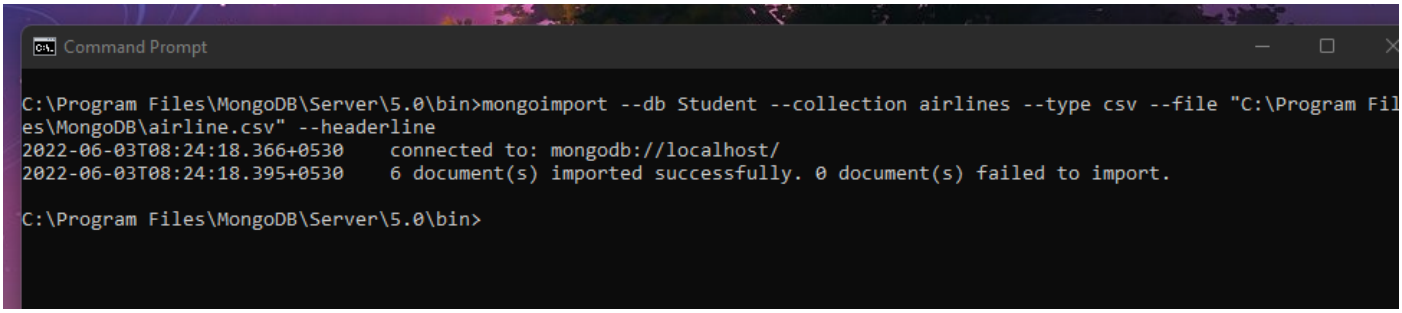
db.Student.find().sort({StudName:-1}).pretty();



III. Import data from a CSV file

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

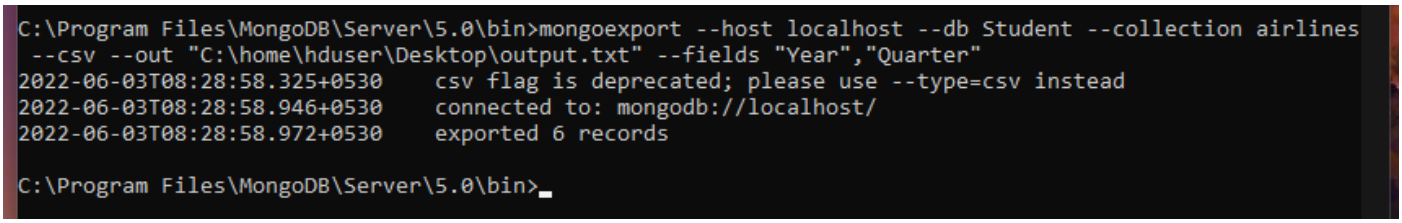
mongoimport --db Student --collection airlines --type csv –headerline --file /home/hduser/Desktop/airline.csv



IV. Export data to a CSV file

This command used at the command prompt exports MongoDB JSON documents from “Customers” collection in the “test” database into a CSV file “Output.txt” in the D:drive.

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



V. Save Method :

Save() method will insert a new document, if the document with the _id does not exist. If it exists it will replace the exisiting document.

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

```
switched to db student
> db.Students.save({StudName:"Vamsi",Grade:"VII"})
WriteResult({ "nInserted" : 1 })
>
```

VI. Add a new field to existing Document:

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

```
> db.Students.update({_id:4},{$set:{Location:"Network"}})
WriteResult({ "nMatched" : 0, "nUpserted" : 0, "nModified" : 0 })
>
```

VII. Remove the field in an existing Document

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

```
Command Prompt - mongo
> db.Students.update({_id:4},{$unset:{Location:"Network"}})
WriteResult({ "nMatched" : 0, "nUpserted" : 0, "nModified" : 0 })
>
```

VIII. Finding Document based on search criteria suppressing few fields

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

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

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

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

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

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

```
> db.Student.find({Grade:{$ne:'VII'}}).pretty();
> db.Student.find({StudName:/s$/}).pretty();
>
```

IX. to set a particular field value to NULL

```
> db.Students.update({_id:3},{$set:{Location:null}})
WriteResult({ "nMatched" : 0, "nUpserted" : 0, "nModified" : 0 })
>
```

X Count the number of documents in Student Collections

```
> db.Student.count()
0
>
```

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

db.Students.count({Grade:”VII”})

retrieve first 3 documents

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

Sort the document in Ascending order

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

Note:

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

to Skip the 1 st two documents from the Students Collections

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

```
> db.Students.find().sort({StudName:1}).pretty();
{
        "_id" : ObjectId("629979944de3211e43081306"),
        "StudName" : "Vamsi",
        "Grade" : "VII"
}
>
```

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

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

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

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

```
Command Prompt - mongo
> 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 })
>
```

To find those documents from the “food” collection which has the “fruits array” constitute of “grapes”, “mango” and “apple”.

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

```
> db.food.find({fruits:['grapes','mango','apple']}).pretty()
{ "_id" : 1, "fruits" : [ "grapes", "mango", "apple" ] }
>
```

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

db.food.find ({'fruits.1':'grapes'})

```
> db.food.find({'fruits.1':'grapes'})
>
```

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

db.food.find ({“fruits”: {$size:2}})

```
> db.food.find ( {"fruits": {$size:2}} )
{ "_id" : 3, "fruits" : [ "banana", "mango" ] }
>
```

To find the document with a particular id and display the first two elements from the array “fruits”

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

```
> db.food.find({_id:1},{"fruits":{$slice:2}})
{ "_id" : 1, "fruits" : [ "grapes", "mango" ] }
>
```

To find all the documets from the food collection which have elements mango and grapes in the array “fruits”

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

```
> db.food.find({fruits:{$all:["mango","grapes"]}})
{ "_id" : 1, "fruits" : [ "grapes", "mango", "apple" ] }
{ "_id" : 2, "fruits" : [ "grapes", "mango", "cherry" ] }
>
```

update on Array:

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

array with apple

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

insert new key value pairs in the fruits array

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

```
> 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 })
>
```

Note: perform query operations using - pop, addToSet, pullAll and pull

XII. Aggregate Function :

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

Now group on “custID” and compute the sum of “AccBal”.

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

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

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

{$sum:”$AccBal”} } });

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

total balance greater than 1200.

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

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

```
WriteResult({ "nMatched" : 1, "nUpserted" : 0, "nModified" : 1 })
> db.Customers.aggregate ( {$group : { _id : "$custID",TotAccBal : {$sum:"$AccBal"} } } );
> db.Customers.aggregate ( {$match:{AcctType:”S”}},{$group : { _id : “$custID”,TotAccBal :
... {$sum:”$AccBal”} } } );
uncaught exception: SyntaxError: illegal character :
@(shell):1:43
> db.Customers.aggregate ( {$match:{AcctType:"S"}},{$group : { _id :"$custID",TotAccBal :{$sum:"$AccBal
"} } } );
> db.Customers.aggregate ( {$match:{AcctType:"S"}},{$group : { _id : "$custID",TotAccBal :{$sum:"$AccBa
l"} } }, {$match:{TotAccBal:{$gt:1200}}});
>
```

3.b.MongoDB Lab Program 2 (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)).

ii) Insert required documents to the collection.

iii) First Filter on “Dept_Name:CSE” and then group it on “Semester” and

compute the Average CPGA for that semester and flter those documents where the “Avg_CPGA” is greater than 7.5.

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

```
> db.createCollection("Student");
{ "ok" : 1 }
```

```
> db.Student.insert({_id:1,name:"ananya",USN:"1BM19CS095",Sem:6,Dept_Name:"CSE",CGPA:"8.1",Hobbies:"Badminton"});
WriteResult({ "nInserted" : 1 })
> db.Student.insert({_id:2,name:"bharath",USN:"1BM19CS002",Sem:6,Dept_Name:"CSE",CGPA:"8.3",Hobbies:"Swimming"});
WriteResult({ "nInserted" : 1 })
> db.Student.insert({_id:3,name:"chandana",USN:"1BM19CS006",Sem:6,Dept_Name:"CSE",CGPA:"7.1",Hobbies:"Cycling"});
WriteResult({ "nInserted" : 1 })
> db.Student.insert({_id:4,name:"hrithik",USN:"1BM19CS010",Sem:6,Dept_Name:"CSE",CGPA:"8.6",Hobbies:"Reading"});
WriteResult({ "nInserted" : 1 })
> db.Student.insert({_id:5,name:"kanika",USN:"1BM19CS090",Sem:6,Dept_Name:"CSE",CGPA:"9.2",Hobbies:"Cycling"});
WriteResult({ "nInserted" : 1 })
```

```
> db.Student.update({_id:1},{$set:{CGPA:9.0}})
WriteResult({ "nMatched" : 1, "nUpserted" : 0, "nModified" : 1 })
> db.Student.update({_id:2},{$set:{CGPA:9.1}})
WriteResult({ "nMatched" : 1, "nUpserted" : 0, "nModified" : 1 })
> db.Student.update({_id:3},{$set:{CGPA:8.1}})
WriteResult({ "nMatched" : 1, "nUpserted" : 0, "nModified" : 1 })
> db.Student.update({_id:4},{$set:{CGPA:6.5}})
WriteResult({ "nMatched" : 1, "nUpserted" : 0, "nModified" : 1 })
> db.Student.update({_id:5},{$set:{CGPA:8.6}})
WriteResult({ "nMatched" : 1, "nUpserted" : 0, "nModified" : 1 })
> db.students.aggregate({$match:{Dept_Name:"CSE"}},{$group:{_id:"$Sem",AvgCGPA:{$avg:"$CGPA"}}},{$match:{AvgCGPA:{$gt:7.5}}});
> db.Student.aggregate({$match:{Dept_Name:"CSE"}},{$group:{_id:"$Sem",AvgCGPA:{$avg:"$CGPA"}}},{$match:{AvgCGPA:{$gt:7.5}}});
{ "_id" : 6, "AvgCGPA" : 8.26 }
```

```
bmsce@bmsce-Precision-T1700:~$ mongoexport --host localhost --db nayana_db --collection Student --csv --out /home/bmsce/Desktop/output.txt
--fields "_id","Name","USN","Sem","Dept_Name","CGPA","Hobbies"
2022-04-20T15:13:53.933+0530    csv flag is deprecated; please use --type=csv instead
2022-04-20T15:13:53.935+0530    connected to: localhost
2022-04-20T15:13:53.935+0530    exported 5 records
```

```
_id,Name,USN,Sem,Dept_Name,CGPA,Hobbies
1,,1BM19CS095,6,CSE,9,Badminton
2,,1BM19CS002,6,CSE,9.1,Swimming
3,,1BM19CS006,6,CSE,8.1,Cycling
4,,1BM19CS010,6,CSE,6.5,Reading
5,,1BM19CS090,6,CSE,8.6,Cycling
```

2)Create a mongodb collection Bank. Demonstrate the following by choosing felds of your choice.

1. Insert three documents
2. Use Arrays(Use Pull and Pop operation)
3. Use Index
4. Use Cursors
5. Updation

```
> db.createCollection("Bank");
{ "ok" : 1 }
> db.insert({CustID:1, Name:"Trivikram Hegde", Type:"Savings", Contact:["9945678231", "080-22364587"]});
uncaught exception: TypeError: db.insert is not a function :
@(shell):1:1
> db.Bank.insert({CustID:1, Name:"Trivikram Hegde", Type:"Savings", Contact:["9945678231", "080-22364587"]});
WriteResult({ "nInserted" : 1 })
> db.Bank.insert({CustID:2, Name:"Vishvesh Bhat", Type:"Savings", Contact:["6325985615", "080-23651452"]});
WriteResult({ "nInserted" : 1 })
> db.Bank.insert({CustID:3, Name:"Vaishak Bhat", Type:"Savings", Contact:["8971456321", "080-33529458"]});
WriteResult({ "nInserted" : 1 })
> db.Bank.insert({CustID:4, Name:"Pramod P Parande", Type:"Current", Contact:["9745236589", "080-56324587"]});
WriteResult({ "nInserted" : 1 })
> db.Bank.insert({CustID:4, Name:"Shreyas R S", Type:"Current", Contact:["9445678321","044-65611729", "080-25639856"]});
WriteResult({ "nInserted" : 1 })
> db.Bank.find({});
{ "_id" : ObjectId("625d77809329139694f188a2"), "CustID" : 1, "Name" : "Trivikram Hegde", "Type" : "Savings", "Contact" : [ "9945678231", "080
-22364587" ] }
{ "_id" : ObjectId("625d77bd9329139694f188a3"), "CustID" : 2, "Name" : "Vishvesh Bhat", "Type" : "Savings", "Contact" : [ "6325985615", "080-2
3651452" ] }
{ "_id" : ObjectId("625d77e69329139694f188a4"), "CustID" : 3, "Name" : "Vaishak Bhat", "Type" : "Savings", "Contact" : [ "8971456321", "080-33
529458" ] }
{ "_id" : ObjectId("625d78229329139694f188a5"), "CustID" : 4, "Name" : "Pramod P Parande", "Type" : "Current", "Contact" : [ "9745236589", "08
0-56324587" ] }
{ "_id" : ObjectId("625d78659329139694f188a6"), "CustID" : 4, "Name" : "Shreyas R S", "Type" : "Current", "Contact" : [ "9445678321", "044-656
11729", "080-25639856" ] }
> db.Bank.updateMany({CustID:1},{$pop:{Contact:1}} );
{ "acknowledged" : true, "matchedCount" : 1, "modifiedCount" : 1 }
> db.Bank.find({});
{ "_id" : ObjectId("625d77809329139694f188a2"), "CustID" : 1, "Name" : "Trivikram Hegde", "Type" : "Savings", "Contact" : [ "9945678231" ] }
{ "_id" : ObjectId("625d77bd9329139694f188a3"), "CustID" : 2, "Name" : "Vishvesh Bhat", "Type" : "Savings", "Contact" : [ "6325985615", "080-2
3651452" ] }
{ "_id" : ObjectId("625d77e69329139694f188a4"), "CustID" : 3, "Name" : "Vaishak Bhat", "Type" : "Savings", "Contact" : [ "8971456321", "080-33
529458" ] }
{ "_id" : ObjectId("625d78229329139694f188a5"), "CustID" : 4, "Name" : "Pramod P Parande", "Type" : "Current", "Contact" : [ "9745236589", "08
0-56324587" ] }
{ "_id" : ObjectId("625d78659329139694f188a6"), "CustID" : 4, "Name" : "Shreyas R S", "Type" : "Current", "Contact" : [ "9445678321", "044-656
11729", "080-25639856" ] }
```

```
{ "_id" : ObjectId("625d78659329139694f188a6"), "CustID" : 4, "Name" : "Shreyas R S", "Type" : "Current", "Contact" : [ "9445678321", "044-656
11729", "080-25639856" ] }
> db.Bank.updateMany({},{$pull:{Contact:"080-25639856"}} );
{ "acknowledged" : true, "matchedCount" : 5, "modifiedCount" : 1 }
> db.Bank.find({});
{ "_id" : ObjectId("625d77809329139694f188a2"), "CustID" : 1, "Name" : "Trivikram Hegde", "Type" : "Savings", "Contact" : [ "9945678231" ] }
{ "_id" : ObjectId("625d77bd9329139694f188a3"), "CustID" : 2, "Name" : "Vishvesh Bhat", "Type" : "Savings", "Contact" : [ "6325985615", "080-2
3651452" ] }
{ "_id" : ObjectId("625d77e69329139694f188a4"), "CustID" : 3, "Name" : "Vaishak Bhat", "Type" : "Savings", "Contact" : [ "8971456321", "080-33
529458" ] }
{ "_id" : ObjectId("625d78229329139694f188a5"), "CustID" : 4, "Name" : "Pramod P Parande", "Type" : "Current", "Contact" : [ "9745236589", "08
0-56324587" ] }
{ "_id" : ObjectId("625d78659329139694f188a6"), "CustID" : 4, "Name" : "Shreyas R S", "Type" : "Current", "Contact" : [ "9445678321", "044-656
11729" ] }
> db.Bank.createIndex({Name:1, Type:1},{name:});
uncaught exception: SyntaxError: expected expression, got '}' :
@(shell):1:43
> db.Bank.createIndex({Name:1, Type:1},{name:"Find current account holders"});
{
        "createdCollectionAutomatically" : false,
        "numIndexesBefore" : 1,
        "numIndexesAfter" : 2,
        "ok" : 1
}
> db.Bank.find({});
{ "_id" : ObjectId("625d77809329139694f188a2"), "CustID" : 1, "Name" : "Trivikram Hegde", "Type" : "Savings", "Contact" : [ "9945678231" ] }
{ "_id" : ObjectId("625d77bd9329139694f188a3"), "CustID" : 2, "Name" : "Vishvesh Bhat", "Type" : "Savings", "Contact" : [ "6325985615", "080-2
3651452" ] }
{ "_id" : ObjectId("625d77e69329139694f188a4"), "CustID" : 3, "Name" : "Vaishak Bhat", "Type" : "Savings", "Contact" : [ "8971456321", "080-33
529458" ] }
{ "_id" : ObjectId("625d78229329139694f188a5"), "CustID" : 4, "Name" : "Pramod P Parande", "Type" : "Current", "Contact" : [ "9745236589", "08
0-56324587" ] }
{ "_id" : ObjectId("625d78659329139694f188a6"), "CustID" : 4, "Name" : "Shreyas R S", "Type" : "Current", "Contact" : [ "9445678321", "044-656
11729" ] }
> db.Bank.getIndexes()
[
        {
                "v" : 2,
```

```
@(shell):1:20
> db.Bank.update({_id:625d78659329139694f188a6}, {$set: {CustID:5}},{upsert:true});
uncaught exception: SyntaxError: identifier starts immediately after numeric literal :
@(shell):1:20
> db.Bank.update({_id:"625d78659329139694f188a6"}, {$set: {CustID:5}},{upsert:true});
WriteResult({
        "nMatched" : 0,
        "nUpserted" : 1,
        "nModified" : 0,
        "_id" : "625d78659329139694f188a6"
})
> db.Bank.find({});
{ "_id" : ObjectId("625d77809329139694f188a2"), "CustID" : 1, "Name" : "Trivikram Hegde", "Type" : "Savings", "Contact" : [ "9945678231" ] }
{ "_id" : ObjectId("625d77bd9329139694f188a3"), "CustID" : 2, "Name" : "Vishvesh Bhat", "Type" : "Savings", "Contact" : [ "6325985615", "080-2
3651452" ] }
{ "_id" : ObjectId("625d77e69329139694f188a4"), "CustID" : 3, "Name" : "Vaishak Bhat", "Type" : "Savings", "Contact" : [ "8971456321", "080-33
529458" ] }
{ "_id" : ObjectId("625d78229329139694f188a5"), "CustID" : 4, "Name" : "Pramod P Parande", "Type" : "Current", "Contact" : [ "9745236589", "08
0-56324587" ] }
{ "_id" : ObjectId("625d78659329139694f188a6"), "CustID" : 4, "Name" : "Shreyas R S", "Type" : "Current", "Contact" : [ "9445678321", "044-656
11729" ] }
{ "_id" : "625d78659329139694f188a6", "CustID" : 5 }
> db.Bank.update({_id:"625d78659329139694f188a6", CustID:5}, {$set: {Name:"Sumantha K S", Type:"Savings", Contact:["9856321478","011-65897458"
]}},{upsert:true});
WriteResult({ "nMatched" : 1, "nUpserted" : 0, "nModified" : 1 })
> db.Bank.find({});
{ "_id" : ObjectId("625d77809329139694f188a2"), "CustID" : 1, "Name" : "Trivikram Hegde", "Type" : "Savings", "Contact" : [ "9945678231" ] }
{ "_id" : ObjectId("625d77bd9329139694f188a3"), "CustID" : 2, "Name" : "Vishvesh Bhat", "Type" : "Savings", "Contact" : [ "6325985615", "080-2
3651452" ] }
{ "_id" : ObjectId("625d77e69329139694f188a4"), "CustID" : 3, "Name" : "Vaishak Bhat", "Type" : "Savings", "Contact" : [ "8971456321", "080-33
529458" ] }
{ "_id" : ObjectId("625d78229329139694f188a5"), "CustID" : 4, "Name" : "Pramod P Parande", "Type" : "Current", "Contact" : [ "9745236589", "08
0-56324587" ] }
{ "_id" : ObjectId("625d78659329139694f188a6"), "CustID" : 4, "Name" : "Shreyas R S", "Type" : "Current", "Contact" : [ "9445678321", "044-656
11729" ] }
{ "_id" : "625d78659329139694f188a6", "CustID" : 5, "Contact" : [ "9856321478", "011-65897458" ], "Name" : "Sumantha K S", "Type" : "Savings"
}
>
```

1) Using MongoDB,

i) Create a database for Faculty and Create a Faculty Collection(Faculty_id, Name, Designation ,Department, Age, Salary, Specialization(Set)).
ii) Insert required documents to the collection.

iii) First Filter on “Dept_Name:MECH” and then group it on “Designation” and

compute the Average Salary for that Designation and flter those documents where the “Avg_Sal” is greater than 650000. iv) Demonstrate usage of import and export commands

Write MongoDB queries for the following:

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

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

3)To fnd those documents where the price is not set to 15000.

4)To fnd those documents from the Product collection where the quantity is set to 9 and the product name is set to ‘monitor’.

5)To fnd documents from the Product collection where the Product name ends in ‘d’.

```
}
> db.createCollection("faculty");
{ "ok" : 1 }
> db.faculty.insert({_id:1,name:"Dr. Balaraman Ravindran",designation:"Professor",department:"CSE",age:45,salary:100000,specialization:['pytho
n','mysql','sklearn', 'tensorflow']});
WriteResult({ "nInserted" : 1 })
> db.faculty.insert({_id:2,name:"Dr. Mahadev Ghorki",designation:"Assistant Professor",department:"CSE",age:35,salary:80000,specialization:['p
ython','numpy','sklearn', 'tensorflow', 'java']});
WriteResult({ "nInserted" : 1 })
> db.faculty.insert({_id:3,name:"Dr. Praveen Borade",designation:"Associate Professor",department:"ME",age:40,salary:75000,specialization:['au
tocad', 'aerodynamics', 'thermal physics']});
WriteResult({ "nInserted" : 1 })
> db.faculty.insert({_id:4,name:"Dr. Madhav Nayak",designation:"Assistant Professor",department:"ME",age:37,salary:95000,specialization:['auto
cad', 'flight-dynamics', 'Finite Element Analysis']});
WriteResult({ "nInserted" : 1 })
> db.faculty.aggregate ( {$match:{department:"ME"}}, {$group : {_id : "$designation", AverageSal :{$avg:"$salary"} } },  {$match:{AverageSal:{
$gt:50000}}});
{ "_id" : "Associate Professor", "AverageSal" : 75000 }
{ "_id" : "Assistant Professor", "AverageSal" : 95000 }
> db.createCollection("product");
{ "ok" : 1 }
> db.product.insert({pid:1,pname:"keyboard",mdate:2001,price:1800,quantity:2});
WriteResult({ "nInserted" : 1 })
> db.product.insert({pid:2,pname:"mouse",mdate:2005,price:1500,quantity:5});
WriteResult({ "nInserted" : 1 })
> db.product.insert({pid:3,pname:"monitor",mdate:2015,price:10000,quantity:9});
WriteResult({ "nInserted" : 1 })
> db.product.insert({pid:4,pname:"motherboard",mdate:2021,price:15000,quantity:4});
WriteResult({ "nInserted" : 1 })
> db.product.find({},{pname:1,_id:0})
{ "pname" : "keyboard" }
{ "pname" : "mouse" }
{ "pname" : "monitor" }
{ "pname" : "motherboard" }
> db.product.find({pid:1},{pid:1,_id:0,mdate:1,quantity:1});
{ "pid" : 1, "mdate" : 2001, "quantity" : 2 }
>  db.product.find({price:{$ne:15000}},{pname:1,_id:0});
{ "pname" : "keyboard" }
```

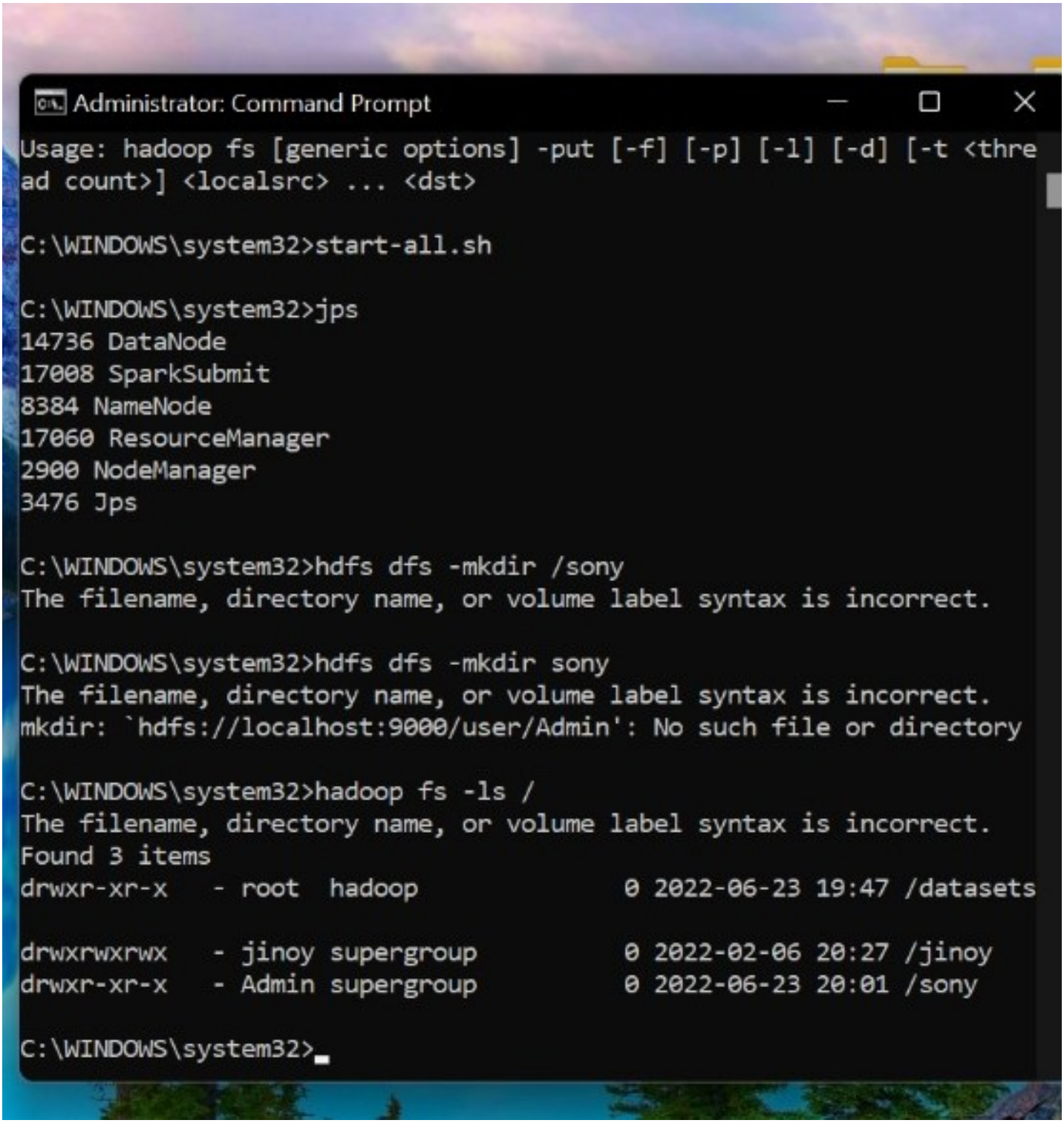
3)Create a mongodb collection Hospital. Demonstrate the following by choosing felds of

choice.

1. Insert three documents
2. Use Arrays(Use Pull and Pop operation)
3. Use Index
4. Use Cursors
5. Updation

```
{ "pname" : "motherboard" }
> db.product.find({pid:1},{pid:1,_id:0,mdate:1,quantity:1});
{ "pid" : 1, "mdate" : 2001, "quantity" : 2 }
>  db.product.find({price:{$ne:15000}},{pname:1,_id:0});
{ "pname" : "keyboard" }
{ "pname" : "mouse" }
{ "pname" : "monitor" }
> db.product.find({$and:[{quantity:{$eq:9}},{pname:{$eq:"monitor"}}]},{pname:1,_id:0})
{ "pname" : "monitor" }
> db.product.find({pname:/d$/},{pname:1,quantity:1,_id:0})
{ "pname" : "keyboard", "quantity" : 2 }
{ "pname" : "motherboard", "quantity" : 4 }
> db.createCollection("hospital");
{ "ok" : 1 }
> db.hospital.insert({_id:1, Name: "Anshuman Agarwal", age:23, diseases:["fever", "diarrhoea", "wheezing", "gastritis"]});
WriteResult({ "nInserted" : 1 })
> db.hospital.insert({_id:2, Name: "Pinky Chaubey", age:35, diseases:["fever","nausea", "food infection", "indigestion", "kidney stones"]});
WriteResult({ "nInserted" : 1 })
> db.hospital.insert({_id:3, Name: "Amresh Chowpati", age:63, diseases:["hyperglycemia", "diabetes mellitus", "food poisoning", "cold"]});
WriteResult({ "nInserted" : 1 })
> db.hospital.updateMany({},{$pull:{diseases:"fever"}});
{ "acknowledged" : true, "matchedCount" : 3, "modifiedCount" : 2 }
> db.hospital.updateOne({_id:1},{$pop:{diseases:-1}});
{ "acknowledged" : true, "matchedCount" : 1, "modifiedCount" : 1 }
> db.hospital.find({"diseases.2":"nausea"});
> db.hospital.find({"diseases.1":"nausea"});
> d.hospital.find({});
uncaught exception: ReferenceError: d is not defined :
@(shell):1:1
> db.hospital.find({});
{ "_id" : 1, "Name" : "Anshuman Agarwal", "age" : 23, "diseases" : [ "wheezing", "gastritis" ] }
{ "_id" : 2, "Name" : "Pinky Chaubey", "age" : 35, "diseases" : [ "nausea", "food infection", "indigestion", "kidney stones" ] }
{ "_id" : 3, "Name" : "Amresh Chowpati", "age" : 63, "diseases" : [ "hyperglycemia", "diabetes mellitus", "food poisoning", "cold" ] }
> db.hospital.find({"diseases.0":"nausea"});
{ "_id" : 2, "Name" : "Pinky Chaubey", "age" : 35, "diseases" : [ "nausea", "food infection", "indigestion", "kidney stones" ] }
> db.hospital.update({_id:3},{$set:{'diseases.1':'sarscov'}});
WriteResult({ "nMatched" : 1, "nUpserted" : 0, "nModified" : 1 })
>
```

4. Hadoop Installation(screenshot)



5. Hadoop Commands

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

C:\hadoop_new\sbin>hdfs dfs -ls /
Found 2 items
drwxr-xr-x   - Admin supergroup          0 2021-04-19 14:47 /lab1
drwxr-xr-x   - Admin supergroup          0 2021-04-19 14:46 /sample

// create a file sample.txt in desktop
C:\hadoop_new\sbin>hdfs dfs -copyFromLocal E:\Desktop\sample.txt \lab1

C:\hadoop_new\sbin>hdfs dfs -ls /lab1
Found 1 items
-rw-r--r--   1 Admin supergroup         19 2021-04-19 14:51 /lab1/sample.txt

C:\hadoop_new\sbin>hdfs dfs -cat \lab1\sample.txt
sample text for lab

//create a folder sample in desktop
C:\hadoop_new\sbin>hdfs dfs -get \lab1\sample.txt E:\Desktop\sample

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

C:\hadoop_new\sbin>hdfs dfs -ls /lab1
Found 2 items
-rw-r--r--   1 Admin supergroup         19 2021-04-19 14:51 /lab1/sample.txt
drwxr-xr-x   - Admin supergroup          0 2021-04-19 14:58 /lab1/samplefolder

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

C:\hadoop_new\sbin>hdfs dfs -ls /lab1
Found 3 items
drwxr-xr-x   - Admin supergroup          0 2021-04-19 14:50 /lab1/sample
-rw-r--r--   1 Admin supergroup         19 2021-04-19 14:51 /lab1/sample.txt
drwxr-xr-x   - Admin supergroup          0 2021-04-19 14:58 /lab1/samplefolder

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

C:\hadoop_new\sbin>hdfs dfs -ls /
Found 2 items
drwxr-xr-x   - Admin supergroup          0 2021-04-19 15:00 /lab1
drwxr-xr-x   - Admin supergroup          0 2021-04-19 15:01 /sample
```

```
C:\hadoop_new\sbin>hdfs dfs -rm /lab1/sample.txt
Deleted /lab1/sample.txt

C:\hadoop_new\sbin>hdfs dfs -ls /lab1
Found 2 items
drwxr-xr-x   - Admin supergroup          0 2021-04-19 14:50 /lab1/sample
drwxr-xr-x   - Admin supergroup          0 2021-04-19 14:58 /lab1/samplefolder

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

C:\hadoop_new\sbin>hdfs dfs -copyToLocal \lab1\sample.txt E:\Desktop\sample1.txt
```

---

6. Hadoop Programs: Word Count

Create Three Java Classes into the project. Name them WCDriver(having the main function), WCMapper, WCReducer.

```
// Importing libraries
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.mapred.MapReduceBase;
import org.apache.hadoop.mapred.Mapper;
import org.apache.hadoop.mapred.OutputCollector;
import org.apache.hadoop.mapred.Reporter;

public class WCMapper extends MapReduceBase implements Mapper<LongWritable,Text, Text,
IntWritable> {

	// Map function
	public void map(LongWritable key, Text value, OutputCollector<Text,
				IntWritable> output, Reporter rep) throws IOException
	{

		String line = value.toString();

		// Splitting the line on spaces
		for (String word : line.split(" "))
		{
			if (word.length() > 0)
			{
				output.collect(new Text(word), new IntWritable(1));
			}
		}
	}
}
```

Reducer Code: You have to copy paste this program into the WCReducer Java Class file.

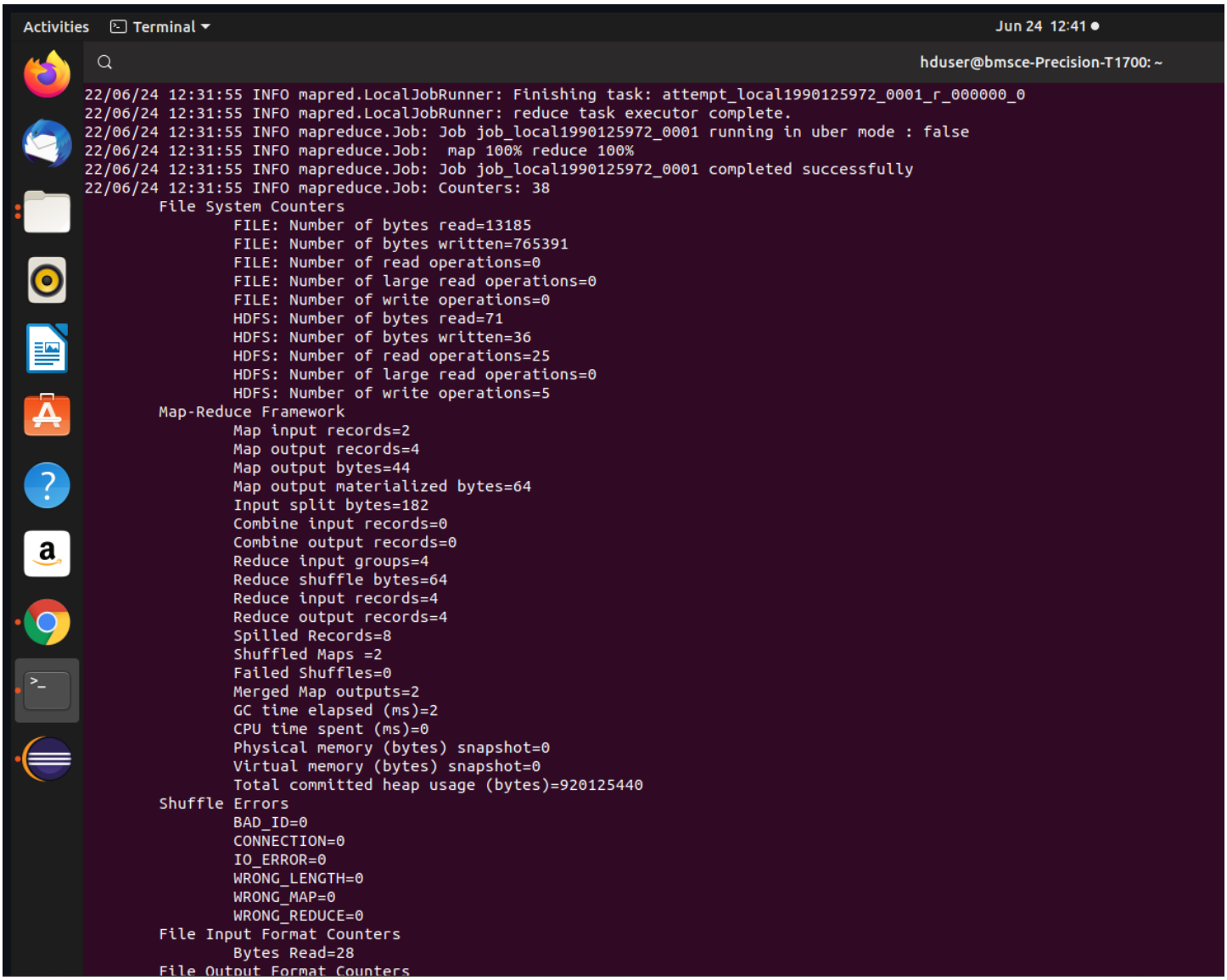
```
 // Importing libraries
import java.io.IOException;
import java.util.Iterator;
import org.apache.hadoop.io.IntWritable;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapred.MapReduceBase;
import org.apache.hadoop.mapred.OutputCollector;
import org.apache.hadoop.mapred.Reducer;
import org.apache.hadoop.mapred.Reporter;
```

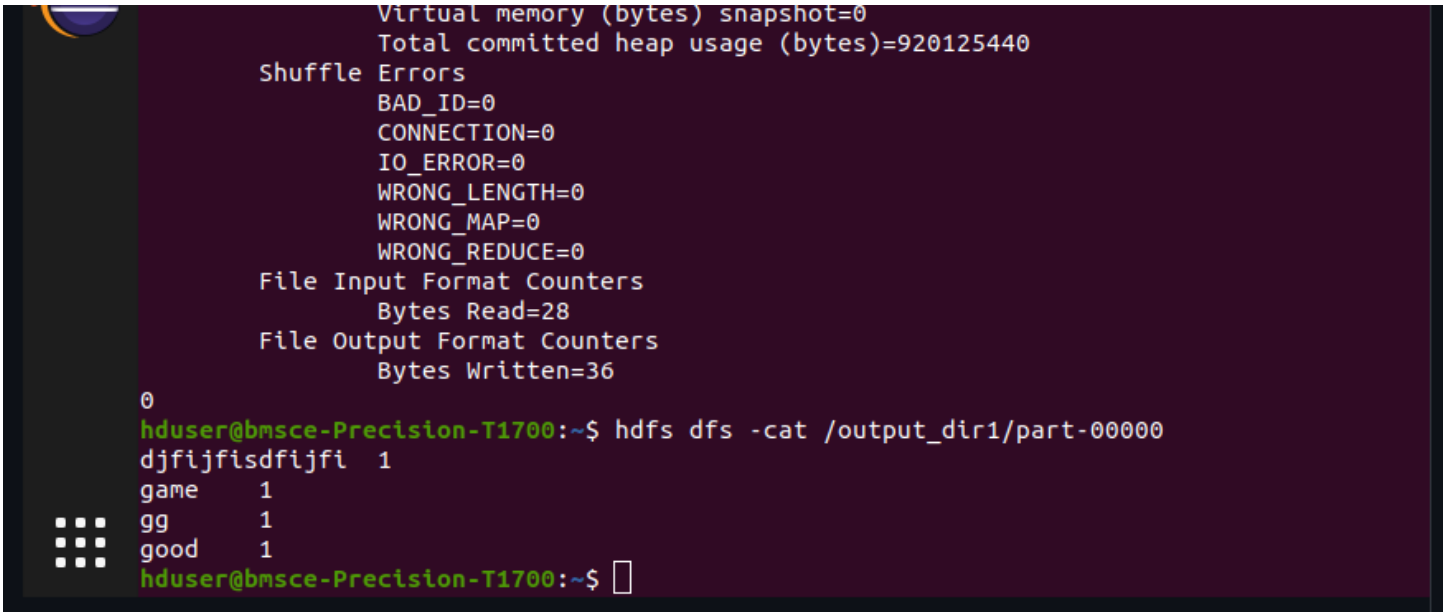
```
public class WCReducer extends MapReduceBase implements Reducer<Text,IntWritable, Text,
IntWritable> {

	// Reduce function
	public void reduce(Text key, Iterator<IntWritable> value,
				OutputCollector<Text, IntWritable> output,
						Reporter rep) throws IOException
	{

		int count = 0;

		// Counting the frequency of each words
		while (value.hasNext())
		{
			IntWritable i = value.next();
			count += i.get();
		}

		output.collect(key, new IntWritable(count));
	}
}
```

Driver Code: You have to copy paste this program into the WCDriver Java Class file.

```
// Importing libraries
import java.io.IOException;
import org.apache.hadoop.conf.Configured;
import org.apache.hadoop.fs.Path;
import org.apache.hadoop.io.IntWritable;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapred.FileInputFormat;
import org.apache.hadoop.mapred.FileOutputFormat;
import org.apache.hadoop.mapred.JobClient;
import org.apache.hadoop.mapred.JobConf;
import org.apache.hadoop.util.Tool;
import org.apache.hadoop.util.ToolRunner;

public class WCDriver extends Configured implements Tool {

	public int run(String args[]) throws IOException
	{
		if (args.length < 2)
		{
			System.out.println("Please give valid inputs");
			return -1;
		}

		JobConf conf = new JobConf(WCDriver.class);
		FileInputFormat.setInputPaths(conf, new Path(args[0]));
		FileOutputFormat.setOutputPath(conf, new Path(args[1]));
```

```
                conf.setMapperClass(WCMapper.class);
                conf.setReducerClass(WCReducer.class);
                conf.setMapOutputKeyClass(Text.class);
                conf.setMapOutputValueClass(IntWritable.class);
                conf.setOutputKeyClass(Text.class);
                conf.setOutputValueClass(IntWritable.class);
                JobClient.runJob(conf);
                return 0;
        }

        // Main Method
        public static void main(String args[]) throws Exception
        {
                int exitCode = ToolRunner.run(new WCDriver(), args);
                System.out.println(exitCode);
        }
}
```





7. Hadoop Programs : Top N

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>
```

8. Hadoop Programs : Avg Temperature and

A. Avg temp:

AverageDriver

```
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));
  }
```

}

```
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. Mean max temp:

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
        Map-Reduce Framework
```

```
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>
```

9. Hadoop Programs : Join

```
// JoinDriver.java
import org.apache.hadoop.conf.Configured;
import org.apache.hadoop.fs.Path;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapred.*;
import org.apache.hadoop.mapred.lib.MultipleInputs;
import org.apache.hadoop.util.*;

public class JoinDriver extends Configured implements Tool {

public static class KeyPartitioner implements Partitioner<TextPair, Text> {
@Override
public void configure(JobConf job) {}

@Override
public int getPartition(TextPair key, Text value, int numPartitions) {
return (key.getFirst().hashCode() & Integer.MAX_VALUE) %
numPartitions;
}
}

@Override

public int run(String[] args) throws Exception {
```

```
if (args.length != 3) {
System.out.println("Usage: <Department Emp Strength input>

<Department Name input> <output>");
return -1;
}

JobConf conf = new JobConf(getConf(), getClass());

conf.setJobName("Join 'Department Emp Strength input' with 'Department Name
input'");

Path AInputPath = new Path(args[0]);
Path BInputPath = new Path(args[1]);
Path outputPath = new Path(args[2]);

MultipleInputs.addInputPath(conf, AInputPath, TextInputFormat.class,

Posts.class);

MultipleInputs.addInputPath(conf, BInputPath, TextInputFormat.class,

User.class);
```

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

conf.setPartitionerClass(KeyPartitioner.class);

conf.setOutputValueGroupingComparator(TextPair.FirstComparator.class);

conf.setMapOutputKeyClass(TextPair.class);

conf.setReducerClass(JoinReducer.class);

conf.setOutputKeyClass(Text.class);

JobClient.runJob(conf);

return 0;
}

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

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

// JoinReducer.java
```

```
import java.io.IOException;
import java.util.Iterator;

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

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

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

throws IOException
{

Text nodeId = new Text(values.next());
while (values.hasNext()) {

Text node = values.next();
Text outValue = new Text(nodeId.toString() + "\t\t" + node.toString());
output.collect(key.getFirst(), outValue);
}
}
```

```
}

// User.java
import java.io.IOException;
import java.util.Iterator;
import org.apache.hadoop.conf.Configuration;
import org.apache.hadoop.fs.FSDataInputStream;
import org.apache.hadoop.fs.FSDataOutputStream;
import org.apache.hadoop.fs.FileSystem;
import org.apache.hadoop.fs.Path;
import org.apache.hadoop.io.LongWritable;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapred.*;

import org.apache.hadoop.io.IntWritable;

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

@Override
public void map(LongWritable key, Text value, OutputCollector<TextPair, Text>
output,
Reporter reporter)
```

```
throws IOException

{

String valueString = value.toString();

String[] SingleNodeData = valueString.split("\t");
output.collect(new TextPair(SingleNodeData[0], "1"), new

Text(SingleNodeData[1]));
}
}

//Posts.java
import java.io.IOException;

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

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

@Override
```

```
public void map(LongWritable key, Text value, OutputCollector<TextPair, Text>
output,
Reporter reporter)
throws IOException
{
String valueString = value.toString();
String[] SingleNodeData = valueString.split("\t");
output.collect(new TextPair(SingleNodeData[3], "0"), new

Text(SingleNodeData[9]));
}
}

// TextPair.java
import java.io.*;

import org.apache.hadoop.io.*;

public class TextPair implements WritableComparable<TextPair> {

private Text first;
private Text second;

public TextPair() {
set(new Text(), new Text());
```

```
}

public TextPair(String first, String second) {
set(new Text(first), new Text(second));
}

public TextPair(Text first, Text second) {
set(first, second);
}

public void set(Text first, Text second) {
this.first = first;
this.second = second;
}

public Text getFirst() {
return first;
}

public Text getSecond() {
return second;
}

@Override
public void write(DataOutput out) throws IOException {
```

```
first.write(out);
second.write(out);
}

@Override
public void readFields(DataInput in) throws IOException {
first.readFields(in);
second.readFields(in);
}

@Override
public int hashCode() {
return first.hashCode() * 163 + second.hashCode();
}

@Override
public boolean equals(Object o) {
if (o instanceof TextPair) {
TextPair tp = (TextPair) o;
return first.equals(tp.first) && second.equals(tp.second);
}
return false;
}

@Override
```

```
public String toString() {
return first + "\t" + second;
}

@Override
public int compareTo(TextPair tp) {
int cmp = first.compareTo(tp.first);
if (cmp != 0) {
return cmp;
}
return second.compareTo(tp.second);
}
// ^^ TextPair

// vv TextPairComparator
public static class Comparator extends WritableComparator {

private static final Text.Comparator TEXT_COMPARATOR = new Text.Comparator();

public Comparator() {
super(TextPair.class);
}

@Override
public int compare(byte[] b1, int s1, int l1,
```

```
byte[] b2, int s2, int l2) {

try {
int firstL1 = WritableUtils.decodeVIntSize(b1[s1]) + readVInt(b1, s1);
int firstL2 = WritableUtils.decodeVIntSize(b2[s2]) + readVInt(b2, s2);
int cmp = TEXT_COMPARATOR.compare(b1, s1, firstL1, b2, s2, firstL2);
if (cmp != 0) {
return cmp;
}
return TEXT_COMPARATOR.compare(b1, s1 + firstL1, l1 - firstL1,

b2, s2 + firstL2, l2 - firstL2);
} catch (IOException e) {
throw new IllegalArgumentException(e);
}
}
}

static {
WritableComparator.define(TextPair.class, new Comparator());
}
public static class FirstComparator extends WritableComparator {

private static final Text.Comparator TEXT_COMPARATOR = new Text.Comparator();
```

```
public FirstComparator() {
super(TextPair.class);
}

@Override
public int compare(byte[] b1, int s1, int l1,
byte[] b2, int s2, int l2) {

try {
int firstL1 = WritableUtils.decodeVIntSize(b1[s1]) + readVInt(b1, s1);
int firstL2 = WritableUtils.decodeVIntSize(b2[s2]) + readVInt(b2, s2);
return TEXT_COMPARATOR.compare(b1, s1, firstL1, b2, s2, firstL2);
} catch (IOException e) {
throw new IllegalArgumentException(e);
}
}

@Override
public int compare(WritableComparable a, WritableComparable b) {
if (a instanceof TextPair && b instanceof TextPair) {
return ((TextPair) a).first.compareTo(((TextPair) b).first);
}
return super.compare(a, b);
}
} }
```

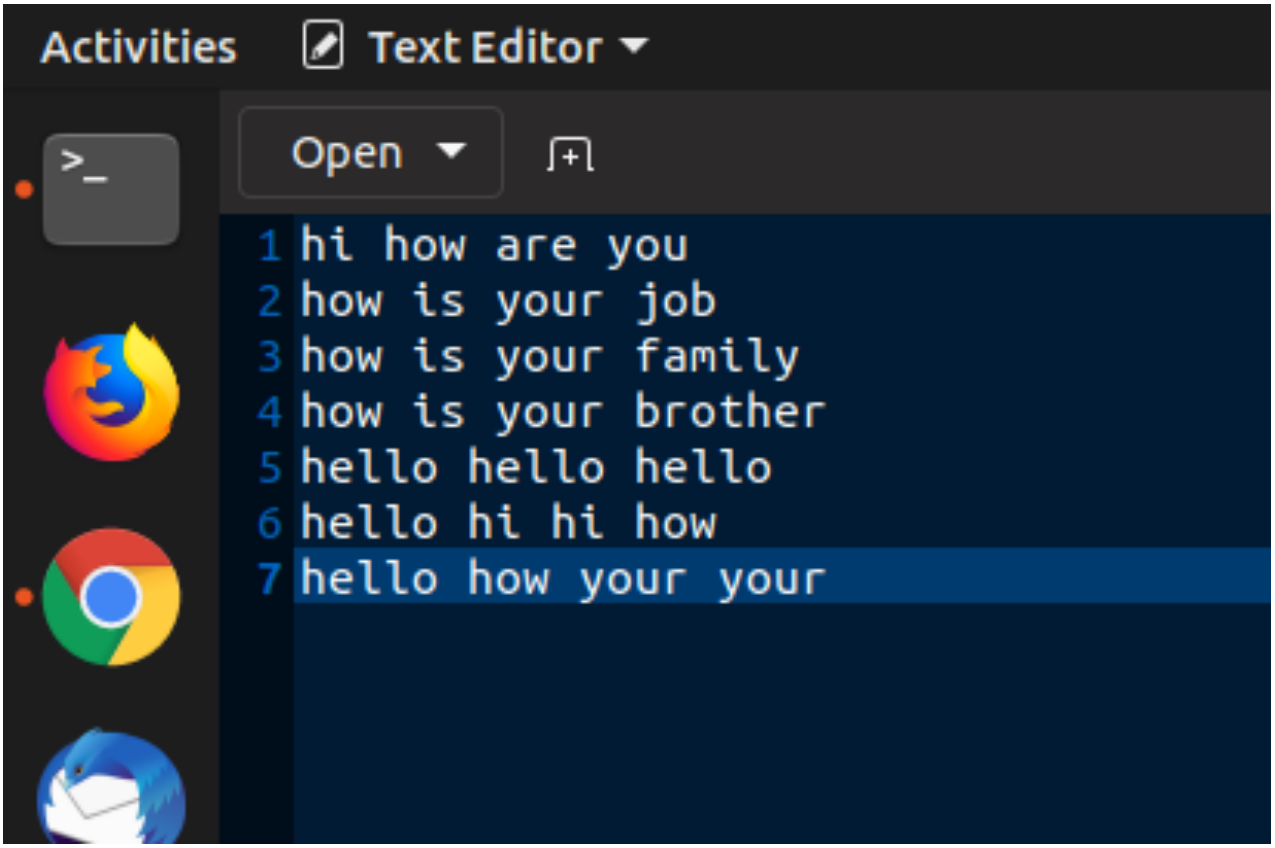
```
C:\hadoop-3.3.0\sbin>hdfs dfs -cat /input_dir/sampleposts.tsv
"2312"  "Feedback on Audio Quality"     "cs101 production audio"        "100005361"     "<p>We are looking for feedback on the audio in our videos. Tell us what you think and try to be as <em>specific</em> as po
ssible.</p>"    "question"      "\N"    "\N"    "2012-02-23 00:28:02.321344+00" "2"     ""      "\N"    "20139145"      "2014-01-14 17:18:35.613939+00" "2960"  "\N"    "\N"    "524"   "f"
"2014856"       ""      "cs101 "        "100022094"     "<p>I also would like to know the answer to this question. An 'open exam' sounds great, but on the other hand it also seems pretty easy to cheat now: solut
ions have been posted and anybody only interested in a certificate wouldn't have much of a problem getting the highest distinction. So where is the catch??</p>"       "answer"        "2014706"       "2014706"
        "2012-07-01 10:32:36.302782+00" "0"     ""      "\N"    "100022094"     "2012-07-01 10:32:36.302782+00" "2020501"       "\N"    "\N"    "0"     "f"
"2004004"       ""      "cs101 "        "100018705"     "<p>But then why even the new variable q? Why not just modify the variable p?</p>"      "comment"       "2003997"       "2003993"       "2012-05-03 21:07:5
2.028935+00"    "2"     ""      "\N"    "100018705"     "2012-05-03 21:07:52.028935+00" "2005150"       "\N"    "\N"    "0"     "f"


C:\hadoop-3.3.0\sbin>hdfs dfs -cat /input_dir/sampleusers.tsv
"100006402"     "18"    "0"     "0"     "0"
"100022094"     "6354"  "4"     "12"    "50"
"100018705"     "76"    "0"     "3"     "4"
"100005361"     "36134" "73"    "220"   "333"
```

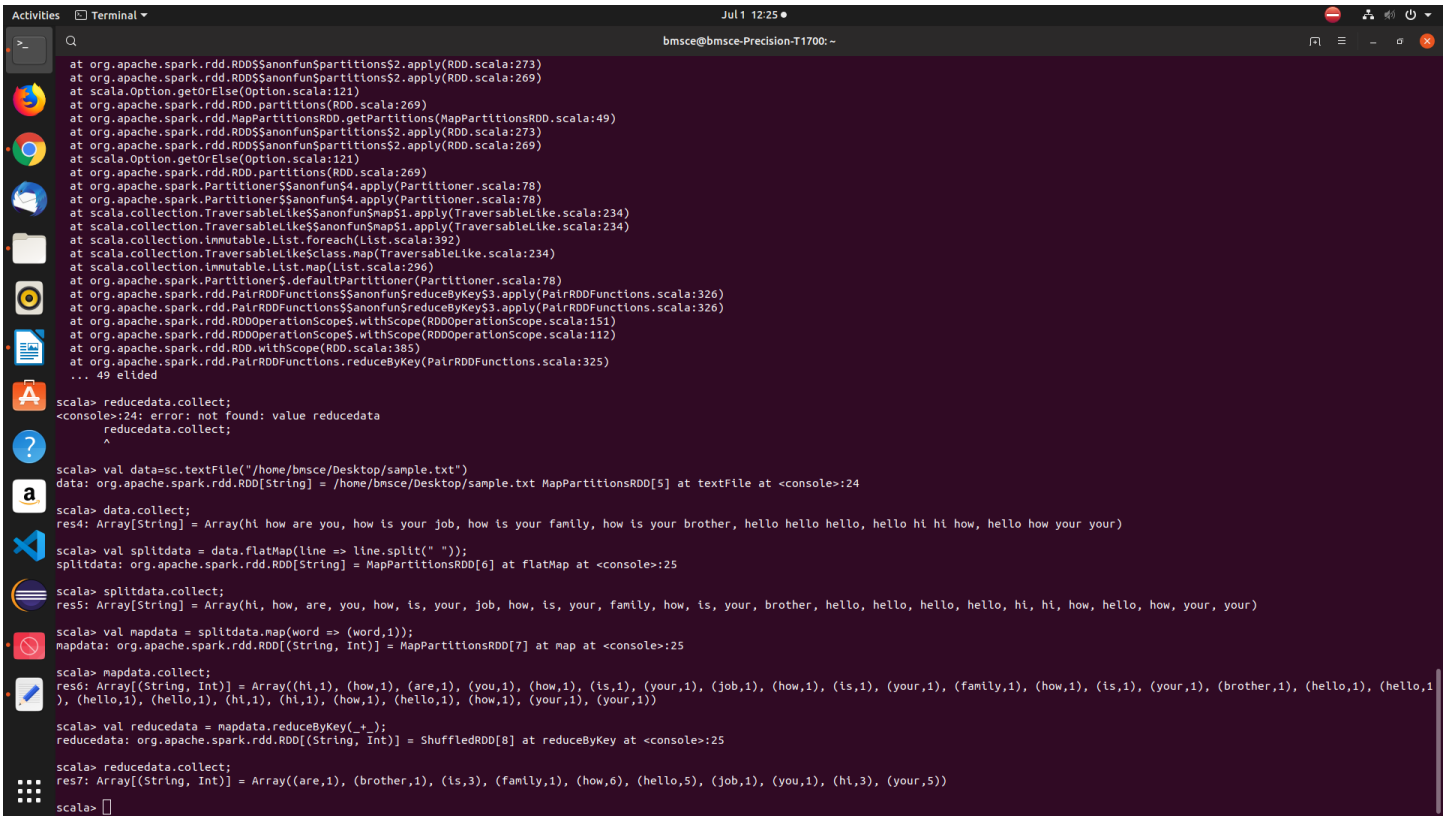
LAB8/Department_Employee_join_example/DeptName.txt

```
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"
```

10.Scala Programs: Word Count and hello world

```
val data=sc.textFile("sparkdata.txt")
data.collect;
val splitdata = data.flatMap(line => line.split(" "));
splitdata.collect;
val mapdata = splitdata.map(word => (word,1));
mapdata.collect;
val reducedata = mapdata.reduceByKey(_+_);
reducedata.collect;
```





$scala

scala> println("Hello World!");
Hello World!

11.Scala Programs: Word Count Greater Than 4

```
val textFile = sc.textFile("/home/admin/Desktop/wc.txt")
val counts = textFile.flatMap(line => line.split(" ")).map(word => (word, 1)).reduceByKey(_ + _)
import scala.collection.immutable.ListMap
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()
     }
}
```

```
scala> val filerdd = sc.textFile("input.txt");
filerdd: org.apache.spark.rdd.RDD[String] = input.txt MapPartitionsRDD[13] at textFile at <console>:24

scala> val counts = filerdd.flatMap(line=>line.split(" ")).map(word=>(word,1)).reduceByKey(_+_);
counts: org.apache.spark.rdd.RDD[(String, Int)] = ShuffledRDD[16] at reduceByKey at <console>:24

scala> import scala.collection.immutable.ListMap
import scala.collection.immutable.ListMap

scala> val sorted = ListMap(counts.collect.sortWith(_._2 > _._2): _*);
sorted: scala.collection.immutable.ListMap[String,Int] = ListMap(im -> 2, is -> 1, here -> 1, there -> 1
, better -> 1, khushil -> 1, lets -> 1, spark -> 1, run -> 1, hadoop -> 1, hi -> 1, to -> 1, see -> 1, w
hich -> 1, and -> 1)

scala> println(sorted);
ListMap(im -> 2, is -> 1, here -> 1, there -> 1, better -> 1, khushil -> 1, lets -> 1, spark -> 1, run -
> 1, hadoop -> 1, hi -> 1, to -> 1, see -> 1, which -> 1, and -> 1)

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

scala> for((k,v)<-sorted)
     | {
     | println(k+",")
     | println(v)
     | println()
     | }
im,
2

is,
1

here,
1

there,
1

better,
1

khushil,
1

lets,
1

spark,
1
```