VISVESVARAYA TECHNOLOGICAL UNIVERSITY

“JnanaSangama”, Belgaum -590014, Karnataka.

LAB REPORT
on

BIG DATA ANALYTICS
(20CS6PEBDA)

Submitted by

R Tarun Kumar (1BM19CS229)

in partial fulfillment for the award of the degree of
BACHELOR OF ENGINEERING
in
COMPUTER SCIENCE AND ENGINEERING

B.M.S. COLLEGE OF ENGINEERING
(Autonomous Institution under VTU)
BENGALURU-560019
May-2022 to July-2022

B. M. S. College of Engineering,

Bull Temple Road, Bangalore 560019

(Affiliated To Visvesvaraya Technological University, Belgaum)

Department of Computer Science and Engineering

CERTIFICATE

This is to certify that the Lab work entitled "**BIG DATA ANALYTICS**" carried out by **R Tarun Kumar(1BM19CS229),** who is bonafide student of **B. M. S. College of Engineering.** It is in partial fulfillment for the award of **Bachelor of Engineering in Computer Science and Engineering** of the Visvesvaraya Technological University, Belgaum during the year 2022. The Lab report has been approved as it satisfies the academic requirements in respect of a **BIG DATA ANALYTICS - (20CS6PEBDA)**work prescribed for the said degree.

ANTARA ROY CHOUDURY	**Dr. Jyothi S Nayak**
Assistant Professor	Professor and Head
Department of CSE	Department of CSE
BMSCE, Bengaluru	BMSCE, Bengaluru

`

Index Sheet

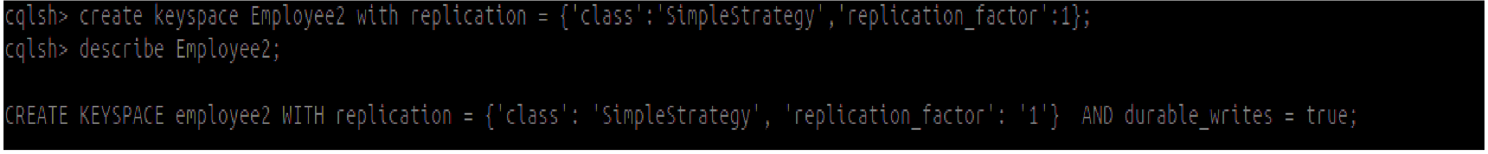
Course Outcome

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

Cassandra Lab Program 1: -

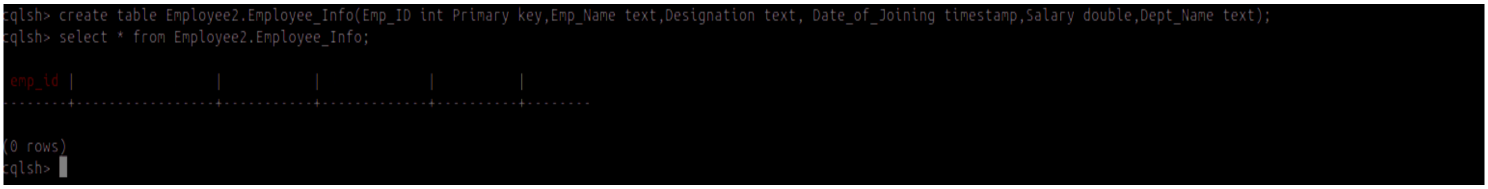
Perform the following DB operations using Cassandra.

1. Create a key space by name Employee



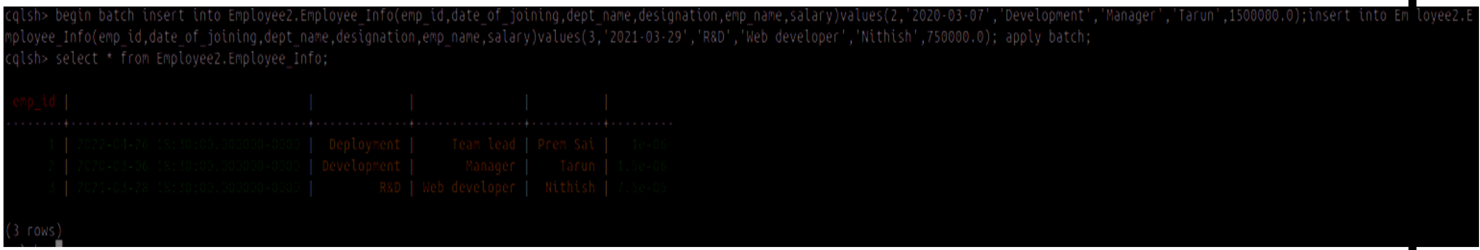
```
cqlsh> create keyspace Employee2 with replication = {'class':'SimpleStrategy','replication_factor':1};
cqlsh> describe Employee2;

CREATE KEYSPACE employee2 WITH replication = {'class': 'SimpleStrategy', 'replication_factor': '1'}  AND durable_writes = true;
```

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



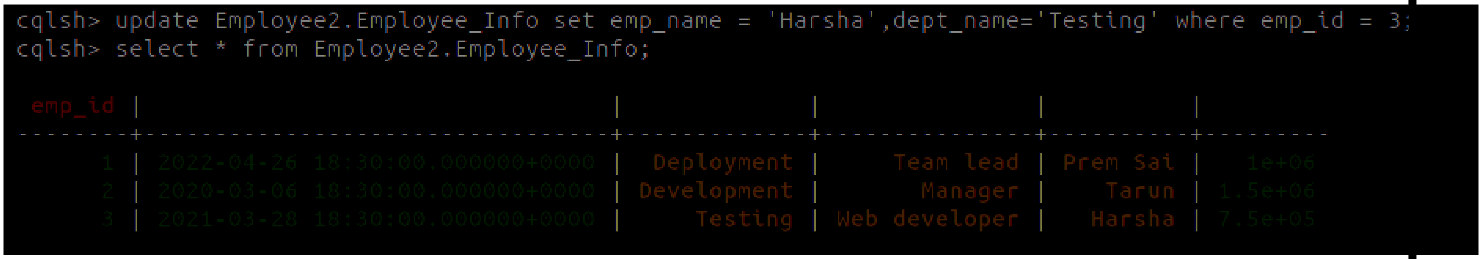
```
cqlsh> create table Employee2.Employee_Info(Emp_ID int Primary key,Emp_Name text,Designation text, Date_of_Joining timestamp,Salary double,Dept_Name text);
cqlsh> select * from Employee2.Employee_Info;

 emp_id |                 |           |             |          |
--------+-----------------+-----------+-------------+----------+--------

(0 rows)
cqlsh> 
```

3. Insert the values into the table in batch



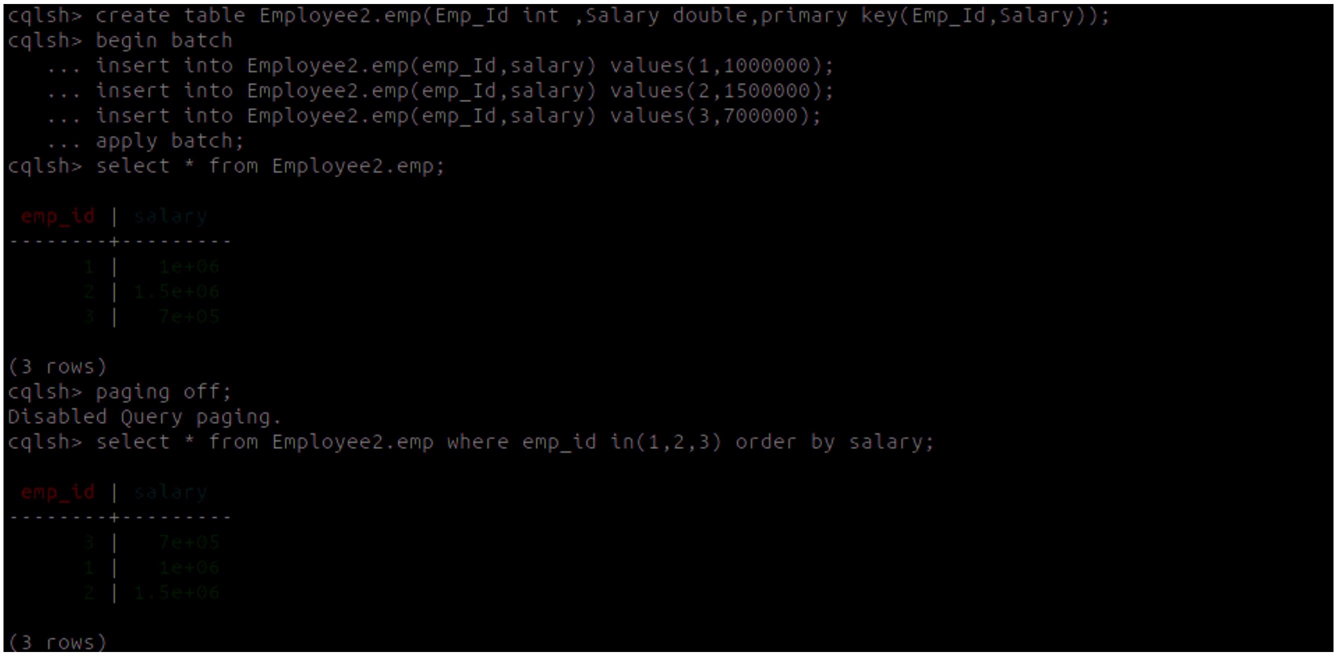
```
cqlsh> begin batch insert into Employee2.Employee_Info(emp_id,date_of_joining,dept_name,designation,emp_name,salary)values(2,'2020-03-07','Development','Manager','Tarun',1500000.0);insert into Em loyee2.E
mployee_Info(emp_id,date_of_joining,dept_name,designation,emp_name,salary)values(3,'2021-03-29','R&D','Web developer','Nithish',750000.0); apply batch;
cqlsh> select * from Employee2.Employee_Info;

 emp_id |                                 |             |               |          |
--------+---------------------------------+-------------+---------------+----------+---------
      1 | 2022-04-26 18:30:00.000000+0000 |  Deployment |     Team lead | Prem Sai |    1e+06
      2 | 2020-03-06 18:30:00.000000+0000 | Development |       Manager |    Tarun | 1.5e+06
      3 | 2021-03-28 18:30:00.000000+0000 |         R&D | Web developer |  Nithish | 7.5e+05

(3 rows)
```

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



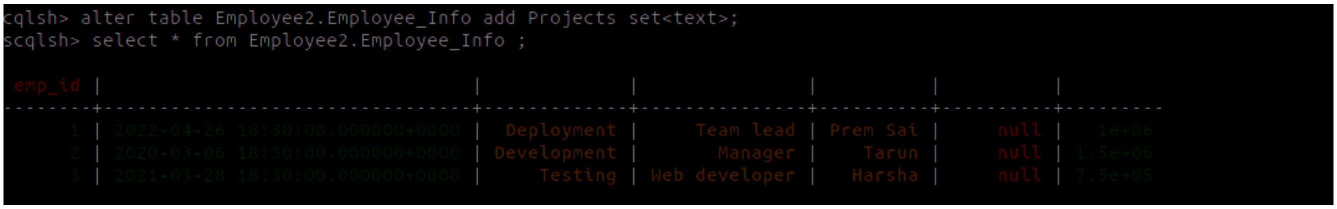
```
cqlsh> update Employee2.Employee_Info set emp_name = 'Harsha',dept_name='Testing' where emp_id = 3;
cqlsh> select * from Employee2.Employee_Info;

 emp_id |                                 |             |               |          |
--------+---------------------------------+-------------+---------------+----------+---------
      1 | 2022-04-26 18:30:00.000000+0000 |  Deployment |     Team lead | Prem Sai |    1e+06
      2 | 2020-03-06 18:30:00.000000+0000 | Development |       Manager |    Tarun | 1.5e+06
      3 | 2021-03-28 18:30:00.000000+0000 |     Testing | Web developer |   Harsha | 7.5e+05
```

5. Sort the details of Employee records based on salary



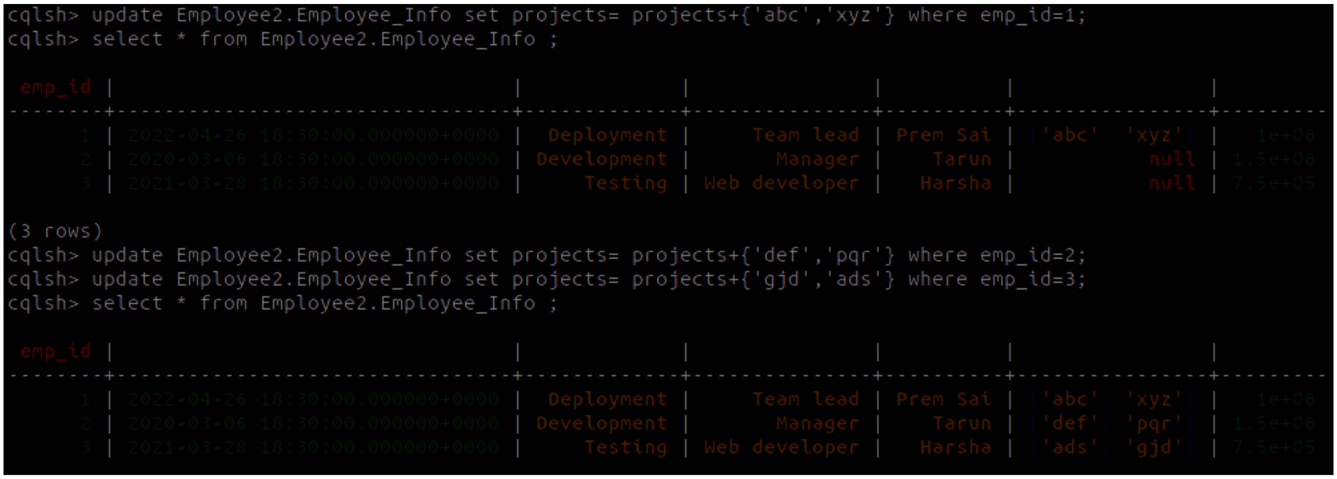
```
cqlsh> create table Employee2.emp(Emp_Id int ,Salary double,primary key(Emp_Id,Salary));
cqlsh> begin batch
   ... insert into Employee2.emp(emp_Id,salary) values(1,1000000);
   ... insert into Employee2.emp(emp_Id,salary) values(2,1500000);
   ... insert into Employee2.emp(emp_Id,salary) values(3,700000);
   ... apply batch;
cqlsh> select * from Employee2.emp;

 emp_id | salary
--------+---------
      1 |   1e+06
      2 | 1.5e+06
      3 |   7e+05

(3 rows)
cqlsh> paging off;
Disabled Query paging.
cqlsh> select * from Employee2.emp where emp_id in(1,2,3) order by salary;

 emp_id | salary
--------+---------
      3 |   7e+05
      1 |   1e+06
      2 | 1.5e+06

(3 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.



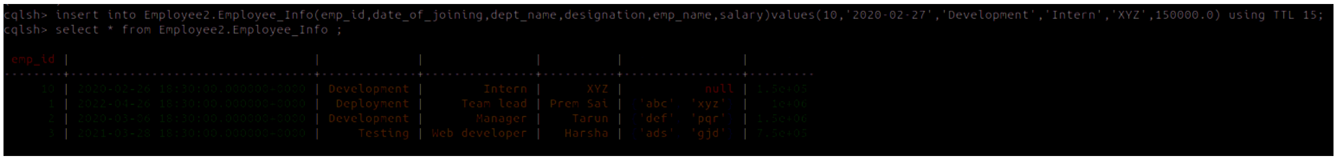
```
cqlsh> alter table Employee2.Employee_Info add Projects set<text>;
scqlsh> select * from Employee2.Employee_Info ;

 emp_id |                                 |             |                |          |          |
--------+---------------------------------+-------------+----------------+----------+----------+---------
      1 | 2022-04-26 18:30:00.000000+0000 |  Deployment |      Team lead | Prem Sai |     null |   1e+06
      2 | 2020-03-06 18:30:00.000000+0000 | Development |        Manager |    Tarun |     null | 1.5e+06
      3 | 2021-03-28 18:30:00.000000+0000 |     Testing | Web developer |   Harsha |     null | 7.5e+05
```

7. Update the altered table to add project names.

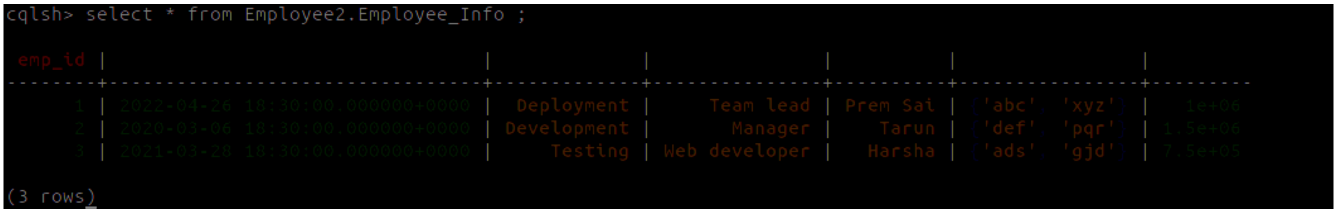


```
cqlsh> update Employee2.Employee_Info set projects= projects+{'abc','xyz'} where emp_id=1;
cqlsh> select * from Employee2.Employee_Info ;

 emp_id |                                 |             |                |          |                |
--------+---------------------------------+-------------+----------------+----------+----------------+---------
      1 | 2022-04-26 18:30:00.000000+0000 |  Deployment |      Team lead | Prem Sai | {'abc', 'xyz'} |   1e+06
      2 | 2020-03-06 18:30:00.000000+0000 | Development |        Manager |    Tarun |           null | 1.5e+06
      3 | 2021-03-28 18:30:00.000000+0000 |     Testing | Web developer |   Harsha |           null | 7.5e+05

(3 rows)
cqlsh> update Employee2.Employee_Info set projects= projects+{'def','pqr'} where emp_id=2;
cqlsh> update Employee2.Employee_Info set projects= projects+{'gjd','ads'} where emp_id=3;
cqlsh> select * from Employee2.Employee_Info ;

 emp_id |                                 |             |                |          |                |
--------+---------------------------------+-------------+----------------+----------+----------------+---------
      1 | 2022-04-26 18:30:00.000000+0000 |  Deployment |      Team lead | Prem Sai | {'abc', 'xyz'} |   1e+06
      2 | 2020-03-06 18:30:00.000000+0000 | Development |        Manager |    Tarun | {'def', 'pqr'} | 1.5e+06
      3 | 2021-03-28 18:30:00.000000+0000 |     Testing | Web developer |   Harsha | {'ads', 'gjd'} | 7.5e+05
```

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



```
cqlsh> insert into Employee2.Employee_Info(emp_id,date_of_joining,dept_name,designation,emp_name,salary)values(10,'2020-02-27','Development','Intern','XYZ',150000.0) using TTL 15;
cqlsh> select * from Employee2.Employee_Info ;

 emp_id |                                 |             |                |          |                |
--------+---------------------------------+-------------+----------------+----------+----------------+---------
     10 | 2020-02-26 18:30:00.000000+0000 | Development |         Intern |      XYZ |           null | 1.5e+05
      1 | 2022-04-26 18:30:00.000000+0000 |  Deployment |      Team lead | Prem Sai | {'abc', 'xyz'} |   1e+06
      2 | 2020-03-06 18:30:00.000000+0000 | Development |        Manager |    Tarun | {'def', 'pqr'} | 1.5e+06
      3 | 2021-03-28 18:30:00.000000+0000 |     Testing | Web developer |   Harsha | {'ads', 'gjd'} | 7.5e+05
```

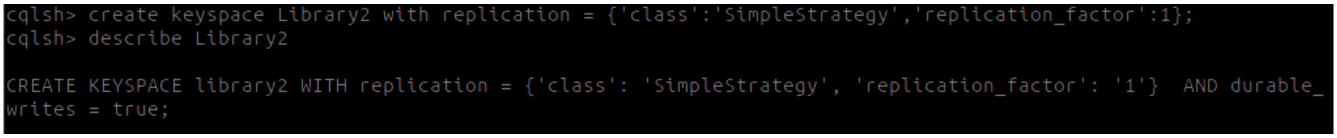


```
cqlsh> select * from Employee2.Employee_Info ;

 emp_id |                                 |             |               |          |                |
--------+---------------------------------+-------------+---------------+----------+----------------+---------
      1 | 2022-04-26 18:30:00.000000+0000 |  Deployment |     Team lead | Prem Sai | {'abc', 'xyz'} |   1e+06
      2 | 2020-03-06 18:30:00.000000+0000 | Development |       Manager |    Tarun | {'def', 'pqr'} | 1.5e+06
      3 | 2021-03-28 18:30:00.000000+0000 |     Testing | Web developer |   Harsha | {'ads', 'gjd'} | 7.5e+05

(3 rows)
```

Cassandra Lab Program 2: -

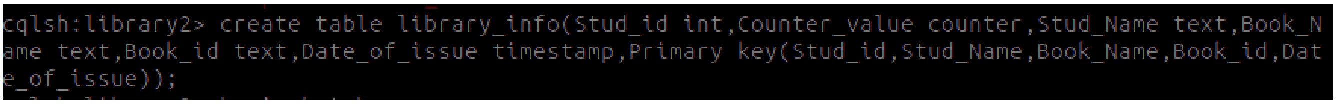
Perform the following DB operations using Cassandra.

1. Create a key space by name Library



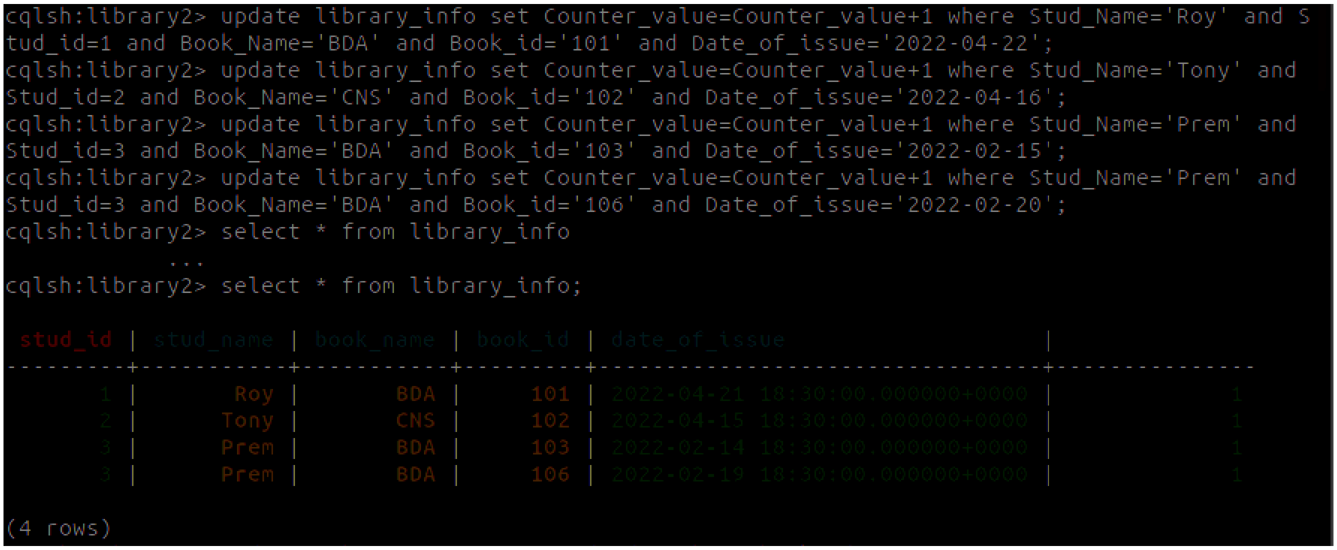
```
cqlsh> create keyspace Library2 with replication = {'class':'SimpleStrategy','replication_factor':1};
cqlsh> describe Library2

CREATE KEYSPACE library2 WITH replication = {'class': 'SimpleStrategy', 'replication_factor': '1'}  AND durable_
writes = true;
```

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



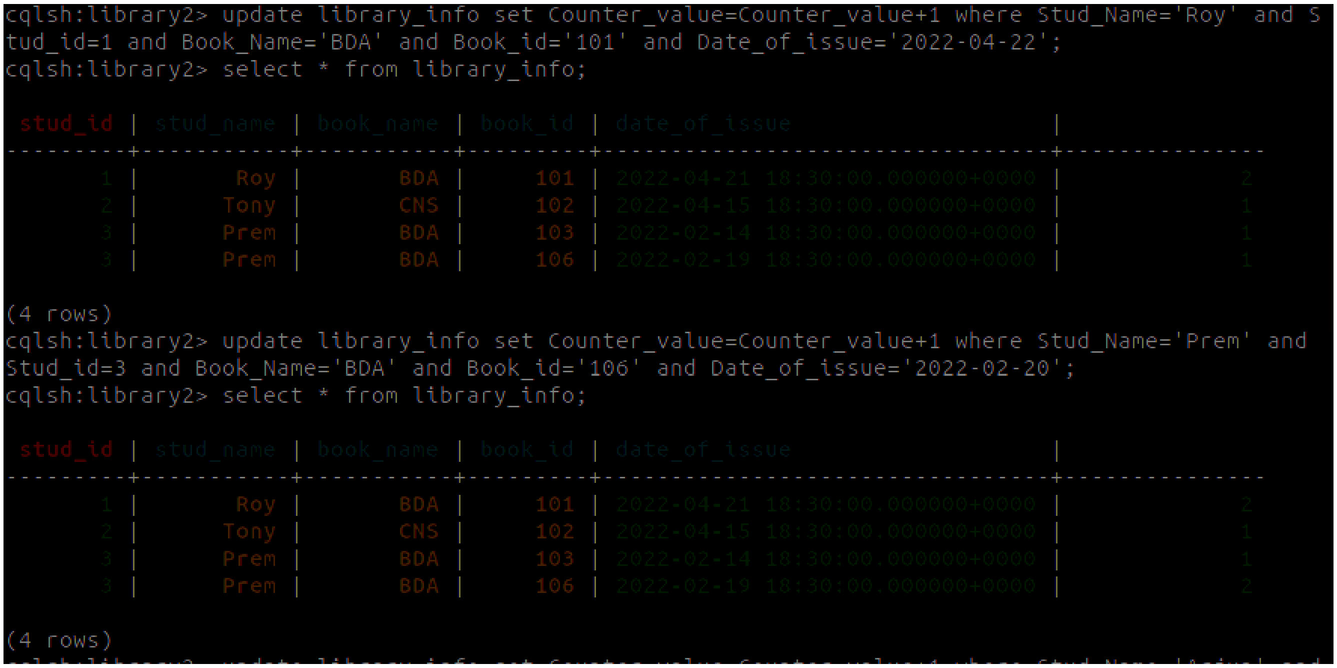
```
cqlsh:library2> create table library_info(Stud_id int,Counter_value counter,Stud_Name text,Book_N
ame text,Book_id text,Date_of_issue timestamp,Primary key(Stud_id,Stud_Name,Book_Name,Book_id,Dat
e_of_issue));
```

3. Insert the values into the table in batch



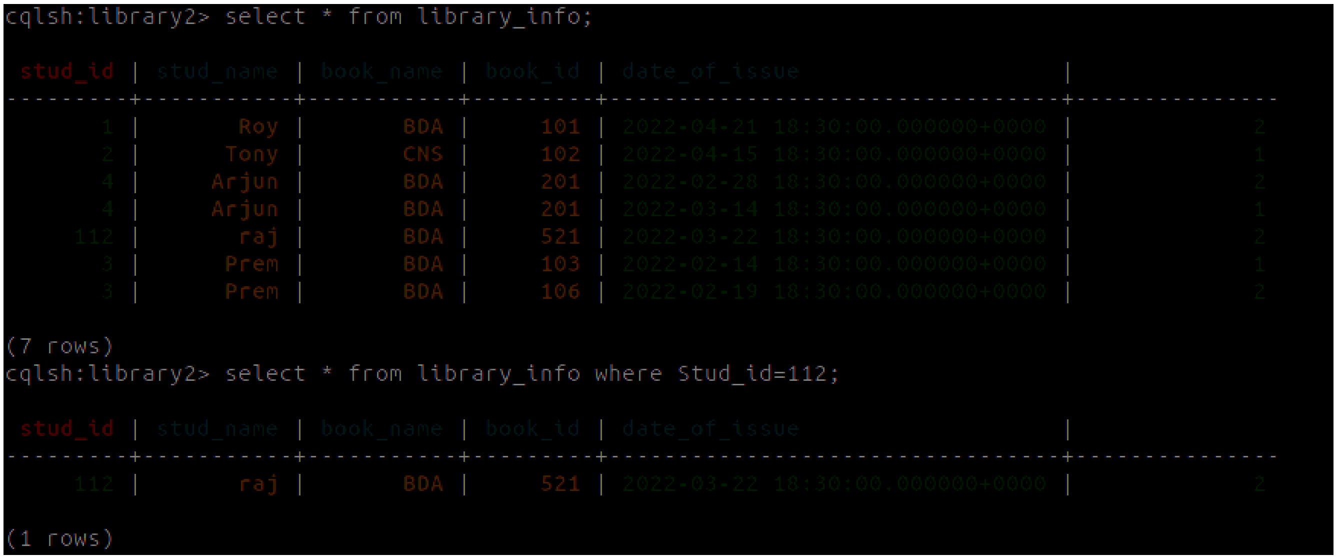
```
cqlsh:library2> update library_info set Counter_value=Counter_value+1 where Stud_Name='Roy' and S
tud_id=1 and Book_Name='BDA' and Book_id='101' and Date_of_issue='2022-04-22';
cqlsh:library2> update library_info set Counter_value=Counter_value+1 where Stud_Name='Tony' and
Stud_id=2 and Book_Name='CNS' and Book_id='102' and Date_of_issue='2022-04-16';
cqlsh:library2> update library_info set Counter_value=Counter_value+1 where Stud_Name='Prem' and
Stud_id=3 and Book_Name='BDA' and Book_id='103' and Date_of_issue='2022-02-15';
cqlsh:library2> update library_info set Counter_value=Counter_value+1 where Stud_Name='Prem' and
Stud_id=3 and Book_Name='BDA' and Book_id='106' and Date_of_issue='2022-02-20';
cqlsh:library2> select * from library_info
           ...
cqlsh:library2> select * from library_info;

 stud_id | stud_name | book_name | book_id | date_of_issue                   |
---------+-----------+-----------+---------+---------------------------------+---------------
       1 |       Roy |       BDA |     101 | 2022-04-21 18:30:00.000000+0000 |             1
       2 |      Tony |       CNS |     102 | 2022-04-15 18:30:00.000000+0000 |             1
       3 |      Prem |       BDA |     103 | 2022-02-14 18:30:00.000000+0000 |             1
       3 |      Prem |       BDA |     106 | 2022-02-19 18:30:00.000000+0000 |             1

(4 rows)
```

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



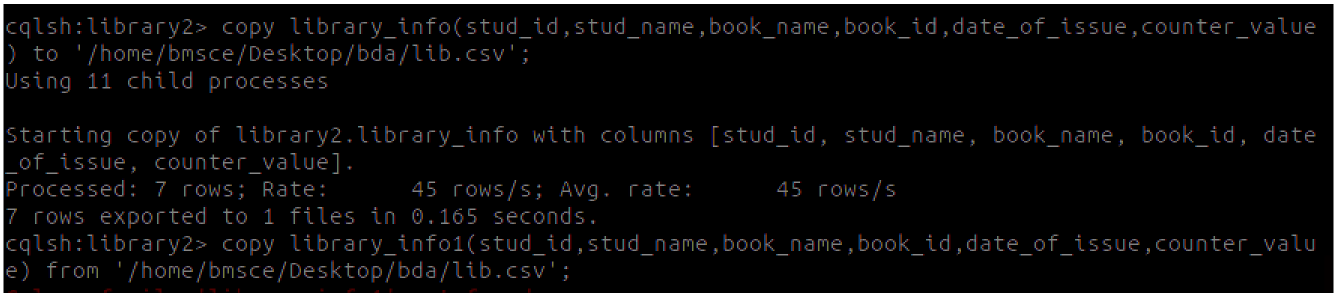
```
cqlsh:library2> update library_info set Counter_value=Counter_value+1 where Stud_Name='Roy' and S
tud_id=1 and Book_Name='BDA' and Book_id='101' and Date_of_issue='2022-04-22';
cqlsh:library2> select * from library_info;

 stud_id | stud_name | book_name | book_id | date_of_issue                   |
---------+-----------+-----------+---------+---------------------------------+---------------
       1 |       Roy |       BDA |     101 | 2022-04-21 18:30:00.000000+0000 |             2
       2 |      Tony |       CNS |     102 | 2022-04-15 18:30:00.000000+0000 |             1
       3 |      Prem |       BDA |     103 | 2022-02-14 18:30:00.000000+0000 |             1
       3 |      Prem |       BDA |     106 | 2022-02-19 18:30:00.000000+0000 |             1

(4 rows)
cqlsh:library2> update library_info set Counter_value=Counter_value+1 where Stud_Name='Prem' and
Stud_id=3 and Book_Name='BDA' and Book_id='106' and Date_of_issue='2022-02-20';
cqlsh:library2> select * from library_info;

 stud_id | stud_name | book_name | book_id | date_of_issue                   |
---------+-----------+-----------+---------+---------------------------------+---------------
       1 |       Roy |       BDA |     101 | 2022-04-21 18:30:00.000000+0000 |             2
       2 |      Tony |       CNS |     102 | 2022-04-15 18:30:00.000000+0000 |             1
       3 |      Prem |       BDA |     103 | 2022-02-14 18:30:00.000000+0000 |             1
       3 |      Prem |       BDA |     106 | 2022-02-19 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.



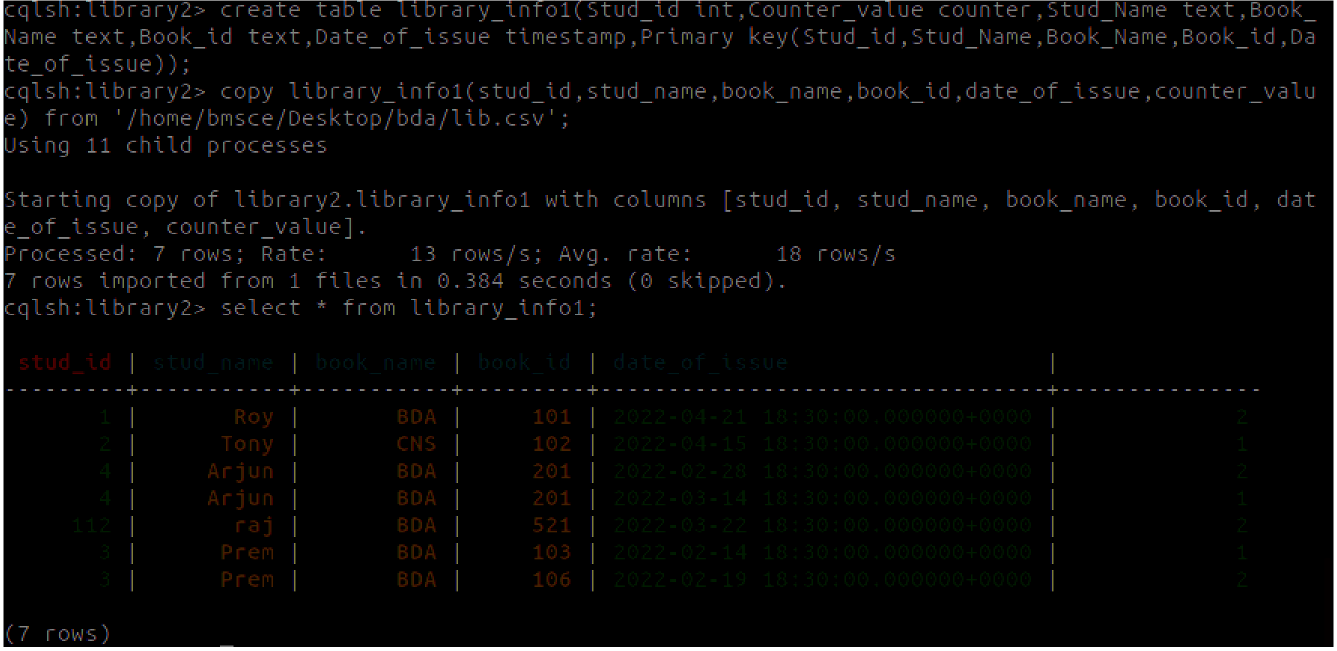
```
cqlsh:library2> select * from library_info;

 stud_id | stud_name | book_name | book_id | date_of_issue                   |
---------+-----------+-----------+---------+---------------------------------+---------------
       1 |       Roy |       BDA |     101 | 2022-04-21 18:30:00.000000+0000 |             2
       2 |      Tony |       CNS |     102 | 2022-04-15 18:30:00.000000+0000 |             1
       4 |     Arjun |       BDA |     201 | 2022-02-28 18:30:00.000000+0000 |             2
       4 |     Arjun |       BDA |     201 | 2022-03-14 18:30:00.000000+0000 |             1
     112 |       raj |       BDA |     521 | 2022-03-22 18:30:00.000000+0000 |             2
       3 |      Prem |       BDA |     103 | 2022-02-14 18:30:00.000000+0000 |             1
       3 |      Prem |       BDA |     106 | 2022-02-19 18:30:00.000000+0000 |             2

(7 rows)
cqlsh:library2> select * from library_info where Stud_id=112;

 stud_id | stud_name | book_name | book_id | date_of_issue                   |
---------+-----------+-----------+---------+---------------------------------+---------------
     112 |       raj |       BDA |     521 | 2022-03-22 18:30:00.000000+0000 |             2

(1 rows)
```

6. Export the created column to a csv file



```
cqlsh:library2> copy library_info(stud_id,stud_name,book_name,book_id,date_of_issue,counter_value
) to '/home/bmsce/Desktop/bda/lib.csv';
Using 11 child processes

Starting copy of library2.library_info with columns [stud_id, stud_name, book_name, book_id, date
_of_issue, counter_value].
Processed: 7 rows; Rate:      45 rows/s; Avg. rate:      45 rows/s
7 rows exported to 1 files in 0.165 seconds.
cqlsh:library2> copy library_info1(stud_id,stud_name,book_name,book_id,date_of_issue,counter_valu
e) from '/home/bmsce/Desktop/bda/lib.csv';
```

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



```
cqlsh:library2> create table library_info1(Stud_id int,Counter_value counter,Stud_Name text,Book_
Name text,Book_id text,Date_of_issue timestamp,Primary key(Stud_id,Stud_Name,Book_Name,Book_id,Da
te_of_issue));
cqlsh:library2> copy library_info1(stud_id,stud_name,book_name,book_id,date_of_issue,counter_valu
e) from '/home/bmsce/Desktop/bda/lib.csv';
Using 11 child processes

Starting copy of library2.library_info1 with columns [stud_id, stud_name, book_name, book_id, dat
e_of_issue, counter_value].
Processed: 7 rows; Rate:      13 rows/s; Avg. rate:      18 rows/s
7 rows imported from 1 files in 0.384 seconds (0 skipped).
cqlsh:library2> select * from library_info1;

 stud_id | stud_name | book_name | book_id | date_of_issue                   |
---------+-----------+-----------+---------+---------------------------------+---------------
       1 |       Roy |       BDA |     101 | 2022-04-21 18:30:00.000000+0000 |             2
       2 |      Tony |       CNS |     102 | 2022-04-15 18:30:00.000000+0000 |             1
       4 |     Arjun |       BDA |     201 | 2022-02-28 18:30:00.000000+0000 |             2
       4 |     Arjun |       BDA |     201 | 2022-03-14 18:30:00.000000+0000 |             1
     112 |       raj |       BDA |     521 | 2022-03-22 18:30:00.000000+0000 |             2
       3 |      Prem |       BDA |     103 | 2022-02-14 18:30:00.000000+0000 |             1
       3 |      Prem |       BDA |     106 | 2022-02-19 18:30:00.000000+0000 |             2

(7 rows)
```

MongoDB Lab Program 1 (CRUD Demonstration): -

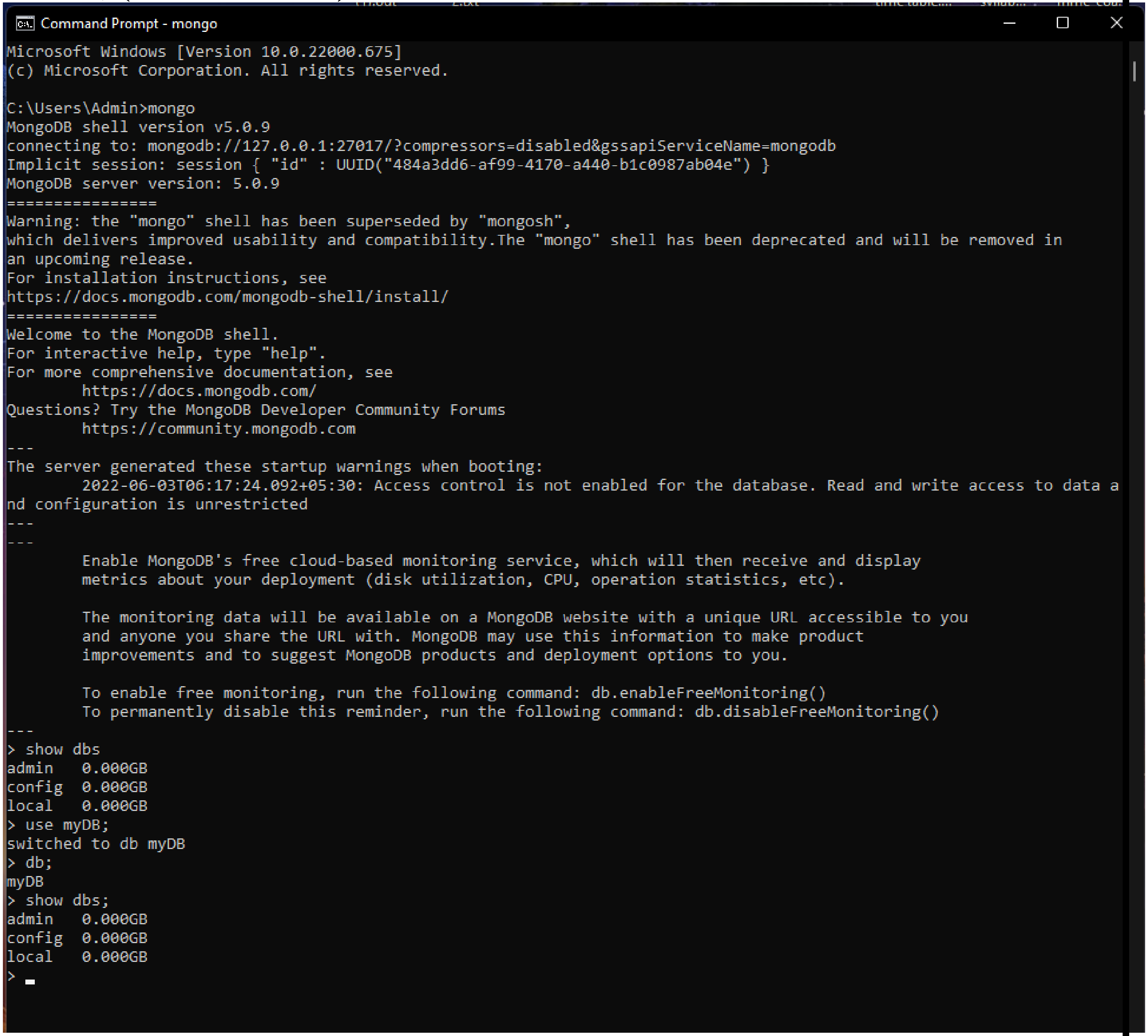
Execute the queries and upload a document with output.

I. CREATE DATABASE IN MONGODB.

use myDB;

db;   (Confirm the existence of your database)

show dbs; (To list all databases)



```
Microsoft Windows [Version 10.0.22000.675]
(c) Microsoft Corporation. All rights reserved.

C:\Users\Admin>mongo
MongoDB shell version v5.0.9
connecting to: mongodb://127.0.0.1:27017/?compressors=disabled&gssapiServiceName=mongodb
Implicit session: session { "id" : UUID("484a3dd6-af99-4170-a440-b1c0987ab04e") }
MongoDB server version: 5.0.9
================
Warning: the "mongo" shell has been superseded by "mongosh",
which delivers improved usability and compatibility.The "mongo" shell has been deprecated and will be removed in
an upcoming release.
For installation instructions, see
https://docs.mongodb.com/mongodb-shell/install/
================
Welcome to the MongoDB shell.
For interactive help, type "help".
For more comprehensive documentation, see
        https://docs.mongodb.com/
Questions? Try the MongoDB Developer Community Forums
        https://community.mongodb.com
---
The server generated these startup warnings when booting:
        2022-06-03T06:17:24.092+05:30: Access control is not enabled for the database. Read and write access to data a
nd configuration is unrestricted
---
---
        Enable MongoDB's free cloud-based monitoring service, which will then receive and display
        metrics about your deployment (disk utilization, CPU, operation statistics, etc).

        The monitoring data will be available on a MongoDB website with a unique URL accessible to you
        and anyone you share the URL with. MongoDB may use this information to make product
        improvements and to suggest MongoDB products and deployment options to you.

        To enable free monitoring, run the following command: db.enableFreeMonitoring()
        To permanently disable this reminder, run the following command: db.disableFreeMonitoring()
---
> show dbs
admin   0.000GB
config  0.000GB
local   0.000GB
> use myDB;
switched to db myDB
> db;
myDB
> show dbs;
admin   0.000GB
config  0.000GB
local   0.000GB
> _
```

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”); =&gt; sql equivalent CREATE TABLE STUDENT(…);
```

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

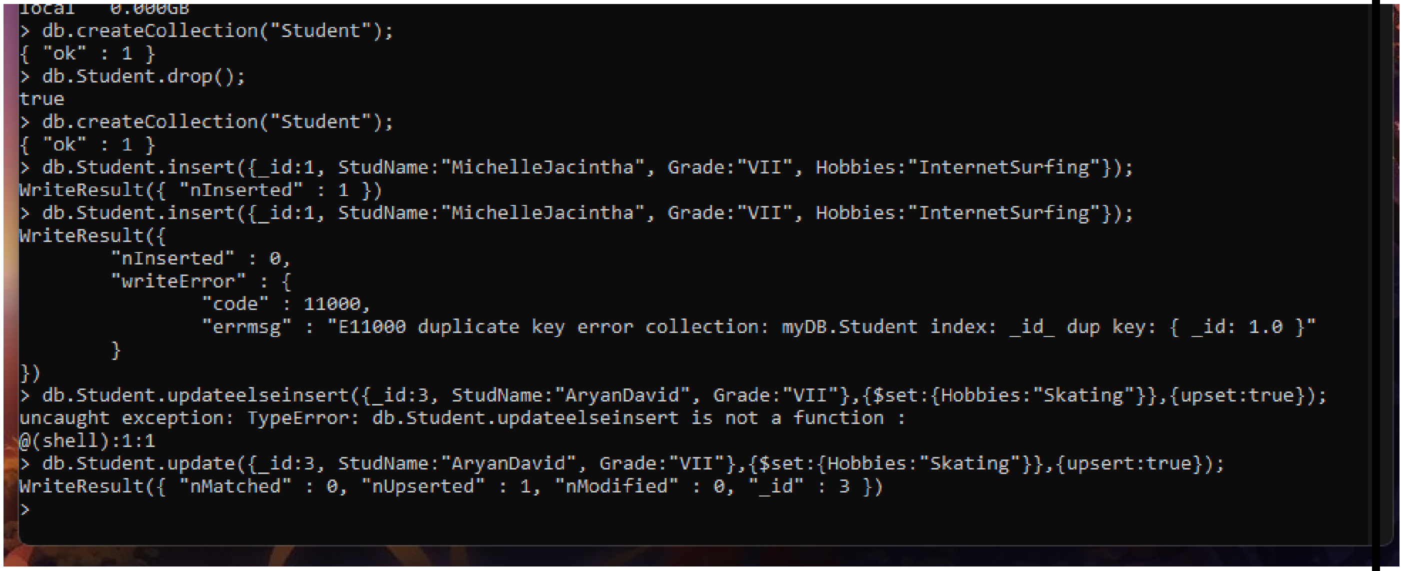
```
db.Student.drop();
```

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

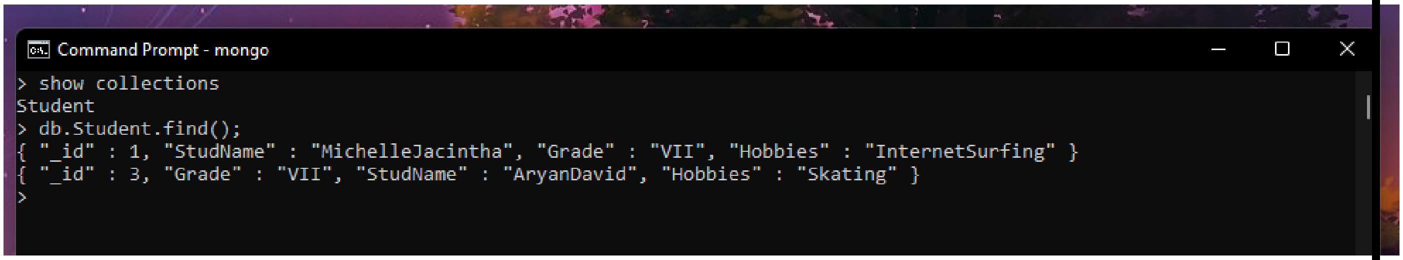
```
db.Student.insert({_id:1,StudName:&quot;MichelleJacintha&quot;,Grade:&quot;VII&quot;,Hob
bies:&quot;InternetS
urfing&quot;});
```

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:&quot;AryanDavid&quot;,Grade:&quot;VII&quot;},{$set:
{Hobbies:&quot;Skatin
g&quot;}},{upsert:true});
```



```
local   0.000GB
> db.createCollection("Student");
{ "ok" : 1 }
> db.Student.drop();
true
> db.createCollection("Student");
{ "ok" : 1 }
> db.Student.insert({_id:1, StudName:"MichelleJacintha", Grade:"VII", Hobbies:"InternetSurfing"});
WriteResult({ "nInserted" : 1 })
> db.Student.insert({_id:1, StudName:"MichelleJacintha", Grade:"VII", Hobbies:"InternetSurfing"});
WriteResult({
        "nInserted" : 0,
        "writeError" : {
                "code" : 11000,
                "errmsg" : "E11000 duplicate key error collection: myDB.Student index: _id_ dup key: { _id: 1.0 }"
        }
})
> db.Student.updateelseinsert({_id:3, StudName:"AryanDavid", Grade:"VII"},{$set:{Hobbies:"Skating"}},{upset:true});
uncaught exception: TypeError: db.Student.updateelseinsert is not a function :
@(shell):1:1
> db.Student.update({_id:3, StudName:"AryanDavid", Grade:"VII"},{$set:{Hobbies:"Skating"}},{upsert:true});
WriteResult({ "nMatched" : 0, "nUpserted" : 1, "nModified" : 0, "_id" : 3 })
>
```



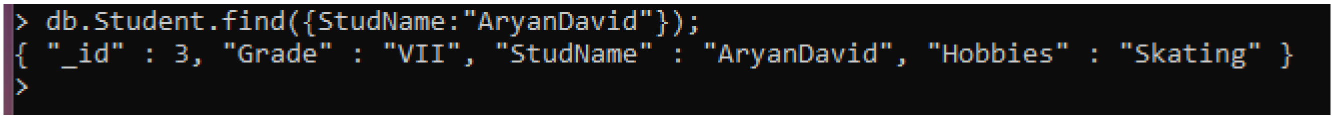
```
> show collections
Student
> db.Student.find();
{ "_id" : 1, "StudName" : "MichelleJacintha", "Grade" : "VII", "Hobbies" : "InternetSurfing" }
{ "_id" : 3, "Grade" : "VII", "StudName" : "AryanDavid", "Hobbies" : "Skating" }
>
```

5. FIND METHOD

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

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

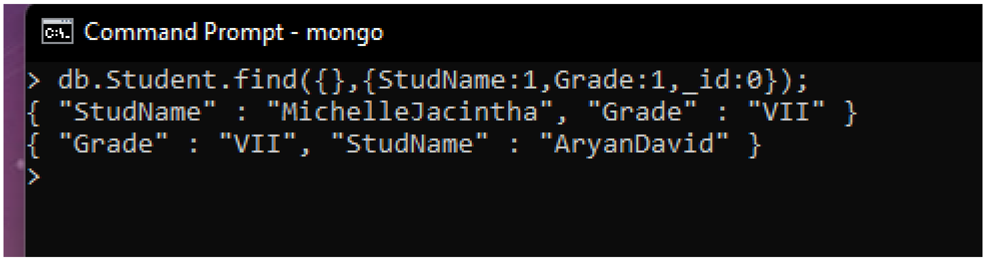
({cond..},{columns.. column:1, columnname:0})



```
> db.Student.find({StudName:"AryanDavid"});
{ "_id" : 3, "Grade" : "VII", "StudName" : "AryanDavid", "Hobbies" : "Skating" }
>
```

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.

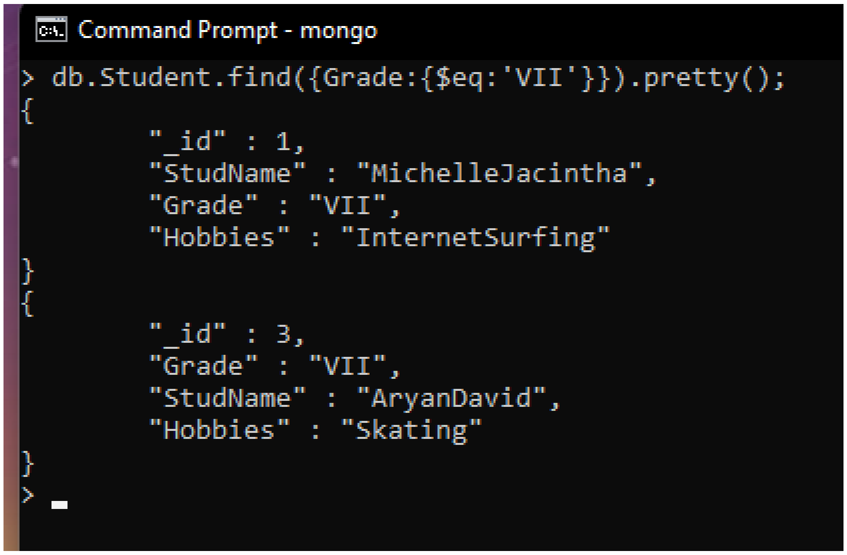
db.Student.find({},{StudName:1,Grade:1,_id:0});



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

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

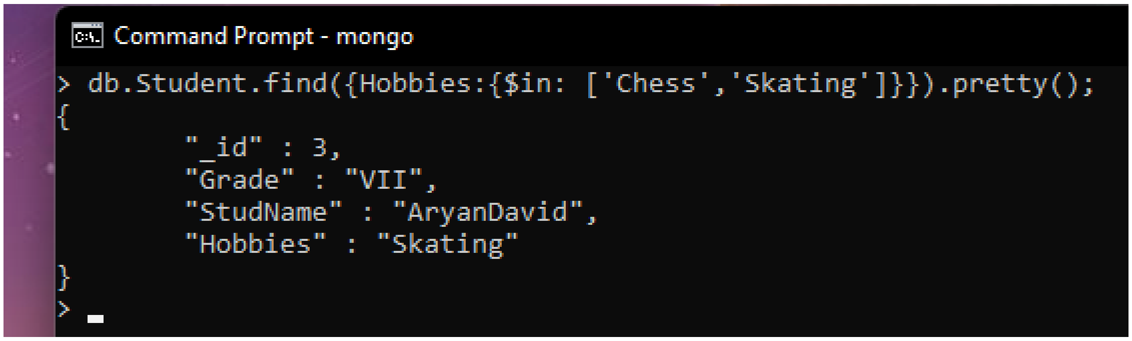
db.Student.find({Grade:{$eq:&#39;VII&#39;}}).pretty();



```
> db.Student.find({Grade:{$eq:'VII'}}).pretty();
{
        "_id" : 1,
        "StudName" : "MichelleJacintha",
        "Grade" : "VII",
        "Hobbies" : "InternetSurfing"
}
{
        "_id" : 3,
        "Grade" : "VII",
        "StudName" : "AryanDavid",
        "Hobbies" : "Skating"
}
>
```

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

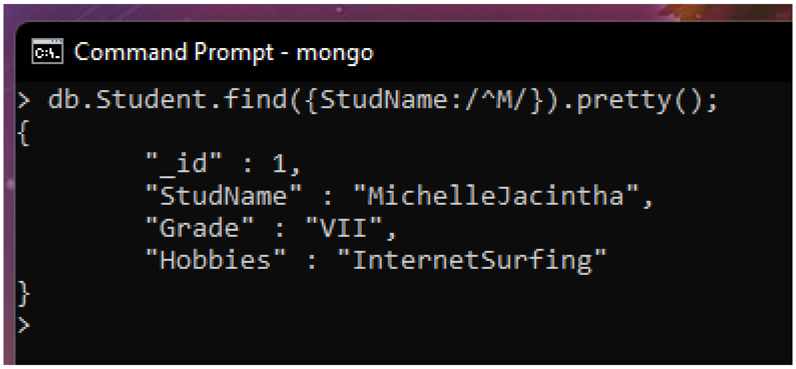
db.Student.find({Hobbies :{ $in: [&#39;Chess&#39;,&#39;Skating&#39;]}}).pretty ();



```
> db.Student.find({Hobbies:{$in: ['Chess','Skating']}}).pretty();
{
        "_id" : 3,
        "Grade" : "VII",
        "StudName" : "AryanDavid",
        "Hobbies" : "Skating"
}
>
```

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

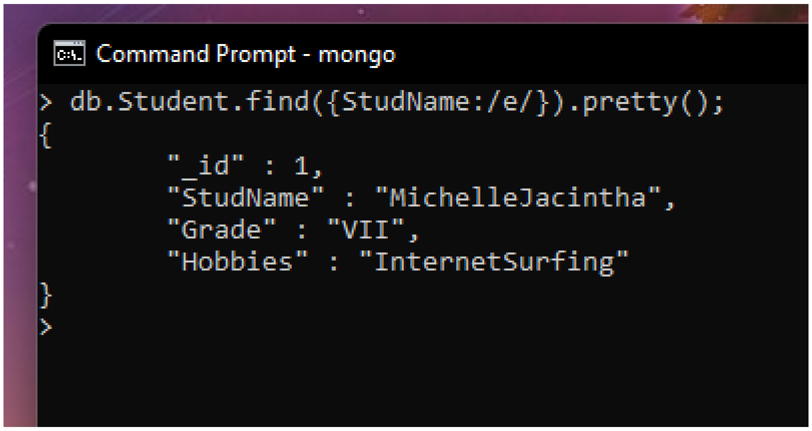
db.Student.find({StudName:/^M/}).pretty();



```
> db.Student.find({StudName:/^M/}).pretty();
{
        "_id" : 1,
        "StudName" : "MichelleJacintha",
        "Grade" : "VII",
        "Hobbies" : "InternetSurfing"
}
>
```

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

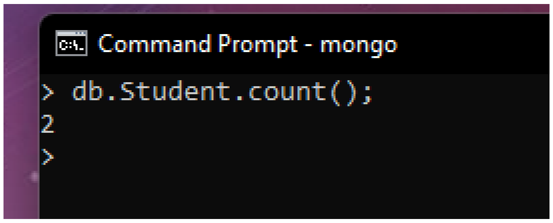
db.Student.find({StudName:/e/}).pretty();



```
> db.Student.find({StudName:/e/}).pretty();
{
        "_id" : 1,
        "StudName" : "MichelleJacintha",
        "Grade" : "VII",
        "Hobbies" : "InternetSurfing"
}
>
```

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

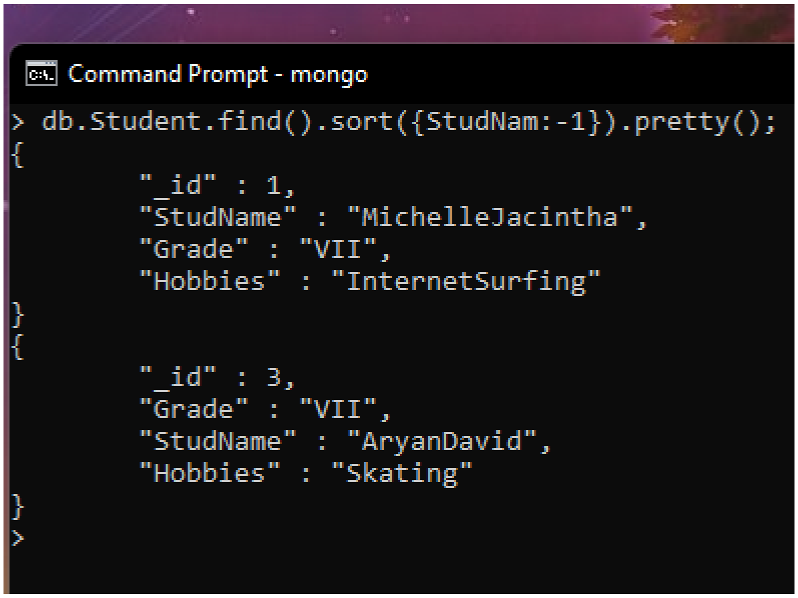
db.Student.count();



```
> db.Student.count();
2
>
```

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

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

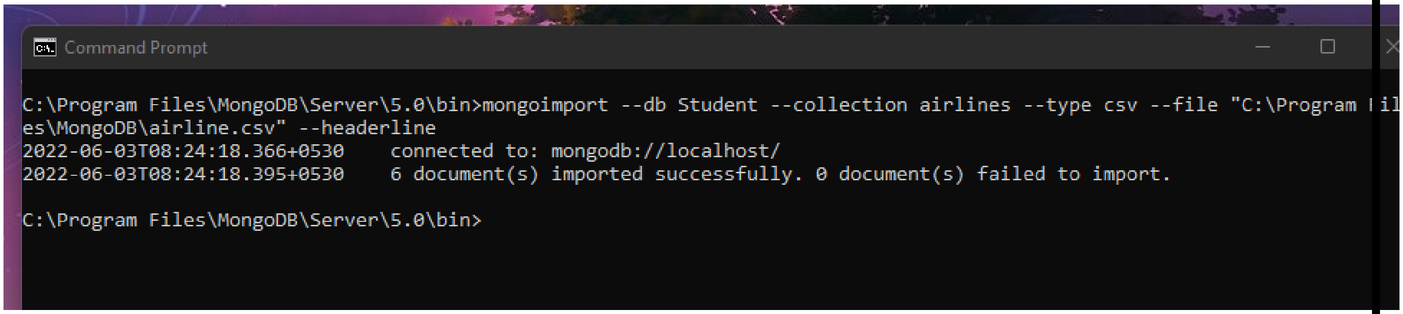


```
> db.Student.find().sort({StudNam:-1}).pretty();
{
        "_id" : 1,
        "StudName" : "MichelleJacintha",
        "Grade" : "VII",
        "Hobbies" : "InternetSurfing"
}
{
        "_id" : 3,
        "Grade" : "VII",
        "StudName" : "AryanDavid",
        "Hobbies" : "Skating"
}
>
```

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

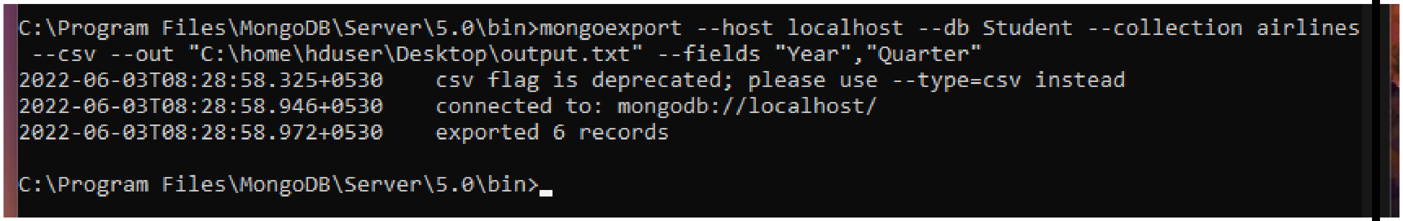


```
C:\Program Files\MongoDB\Server\5.0\bin>mongoimport --db Student --collection airlines --type csv --file "C:\Program Fil
es\MongoDB\airline.csv" --headerline
2022-06-03T08:24:18.366+0530    connected to: mongodb://localhost/
2022-06-03T08:24:18.395+0530    6 document(s) imported successfully. 0 document(s) failed to import.

C:\Program Files\MongoDB\Server\5.0\bin>
```

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”

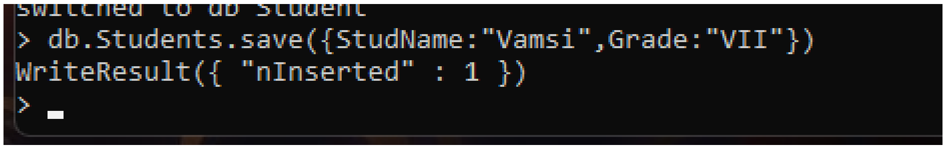


```
C:\Program Files\MongoDB\Server\5.0\bin>mongoexport --host localhost --db Student --collection airlines
 --csv --out "C:\home\hduser\Desktop\output.txt" --fields "Year","Quarter"
2022-06-03T08:28:58.325+0530    csv flag is deprecated; please use --type=csv instead
2022-06-03T08:28:58.946+0530    connected to: mongodb://localhost/
2022-06-03T08:28:58.972+0530    exported 6 records

C:\Program Files\MongoDB\Server\5.0\bin>_
```

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.

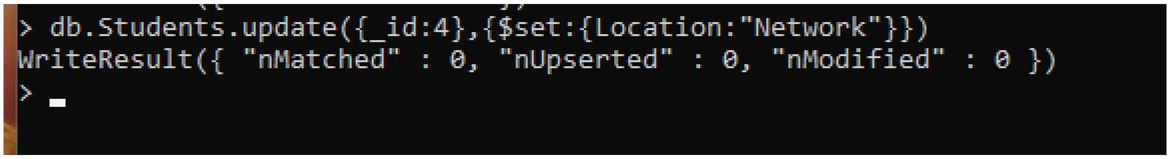
db.Students.save({StudName:”Vamsi”, Grade:”VI”})



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

VI. Add a new field to existing Document:

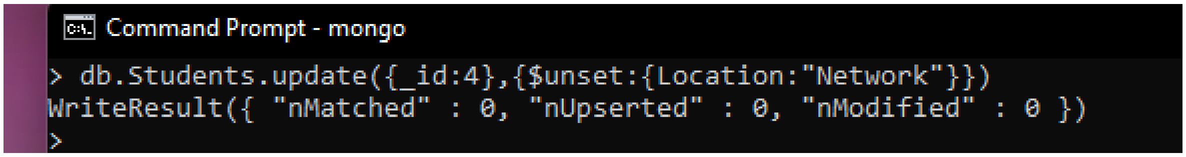
db.Students.update({_id:4},{$set:{Location:”Network”}})



```
> 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”}})



```
> 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

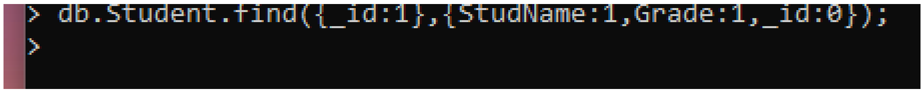
db.Student.find({_id:1},{StudName:1,Grade:1,_id:0});

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

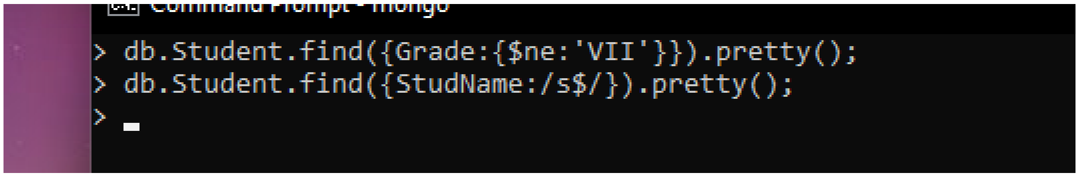
db.Student.find({Grade:{$ne:&#39;VII&#39;}}).pretty();

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

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

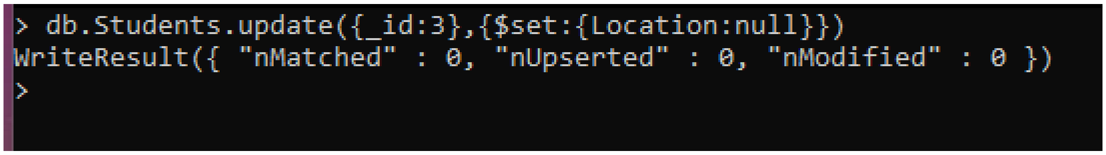


```
> db.Student.find({_id:1},{StudName:1,Grade:1,_id:0});
>
```



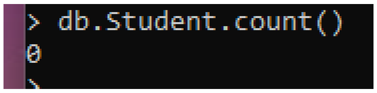
```
> db.Student.find({Grade:{$ne:'VII'}}).pretty();
> db.Student.find({StudName:/s$/}).pretty();
> _
```

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

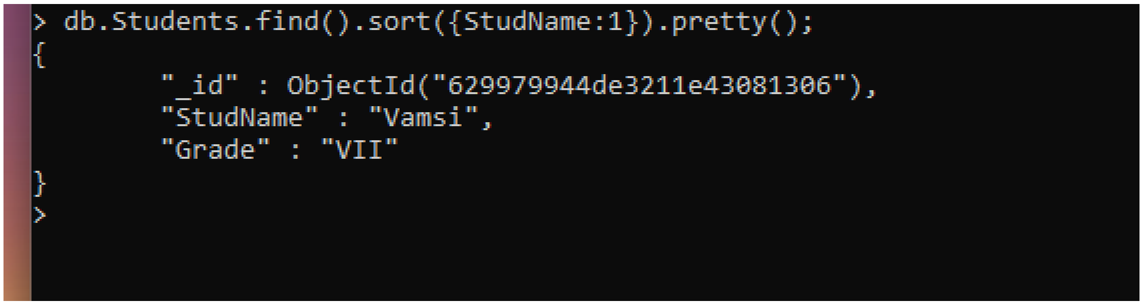
db.Students.find().sort({StudName:1}).pretty();

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()



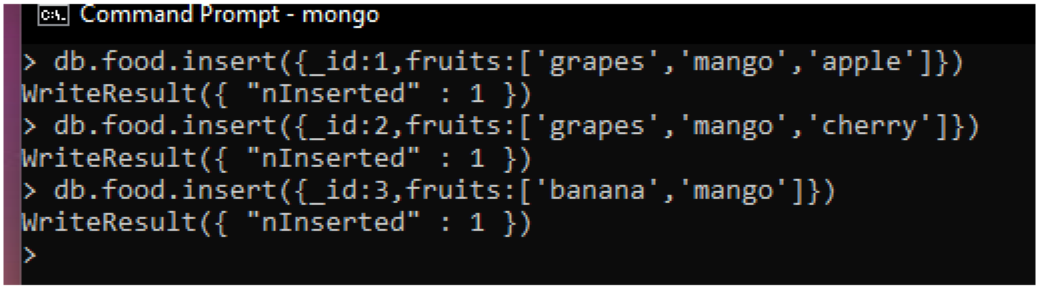
```
> db.Students.find().sort({StudName:1}).pretty();
{
        "_id" : ObjectId("629979944de3211e43081306"),
        "StudName" : "Vamsi",
        "Grade" : "VII"
}
>
```

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

db.food.insert({ _id:1, fruits:[&#39;grapes&#39;,&#39;mango&#39;,&#39;apple&#39;] })

db.food.insert({ _id:2, fruits:[&#39;grapes&#39;,&#39;mango&#39;,&#39;cherry&#39;] })

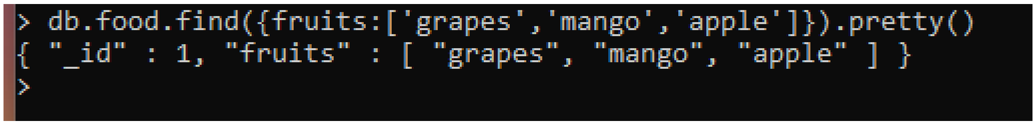
db.food.insert({ _id:3, fruits:[&#39;banana&#39;,&#39;mango&#39;] })



```
> 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”.

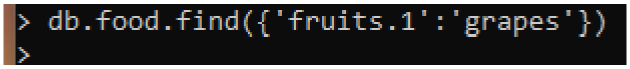
db.food.find ({fruits: [&#39;grapes&#39;,&#39;mango&#39;,&#39;apple&#39;] }). pretty().



```
> 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.

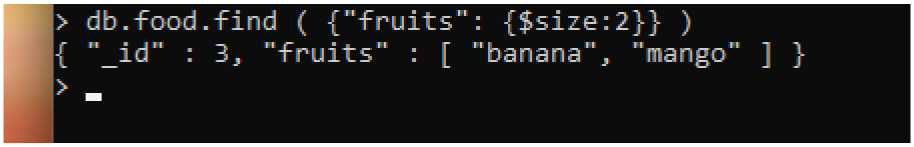
db.food.find ({&#39;fruits.1&#39;:&#39;grapes&#39;})



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

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

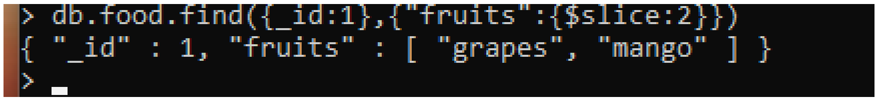
db.food.find ({“fruits”: {$size:2}})



```
> 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”

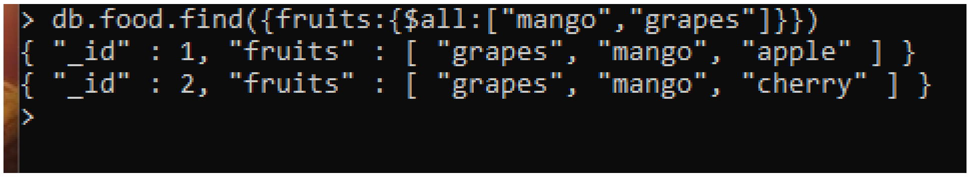
db.food.find({_id:1},{“fruits”:{$slice:2}})



```
> 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" ] }
>
```

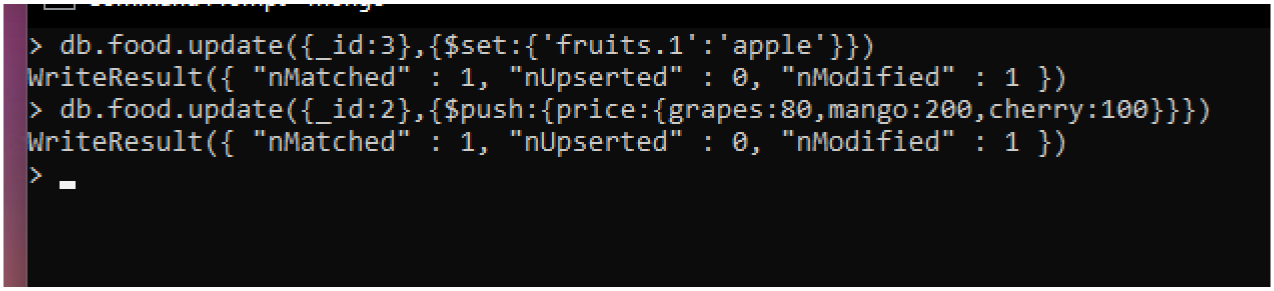
update on Array:

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:{&#39;fruits.1&#39;:&#39;apple&#39;}})

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”.

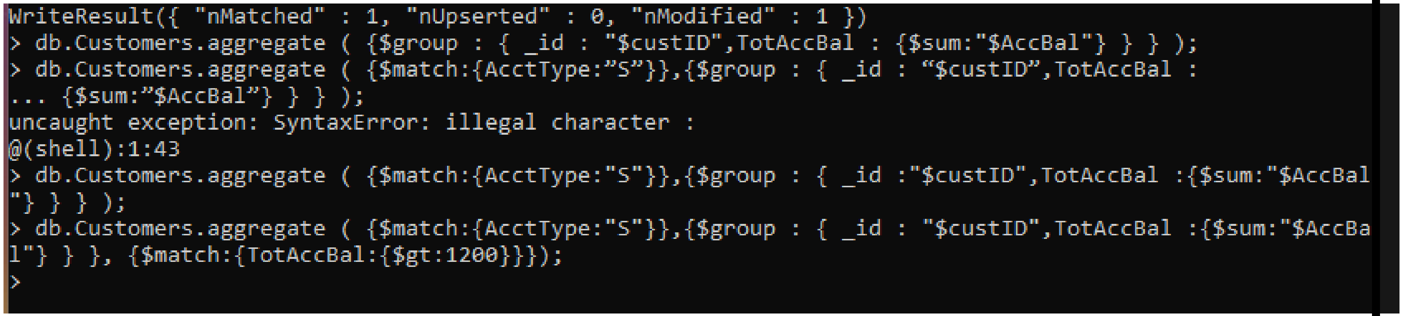
db.Customers.aggregate ({$group : { _id : “$custID”,TotAccBal : {$sum:”$AccBal”} } });

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}}});
>
```

MongoDB Lab Program 2 (CRUD Demonstration): -

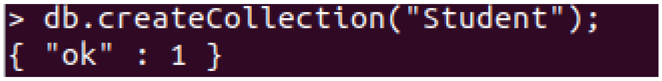
1) Using MongoDB

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.

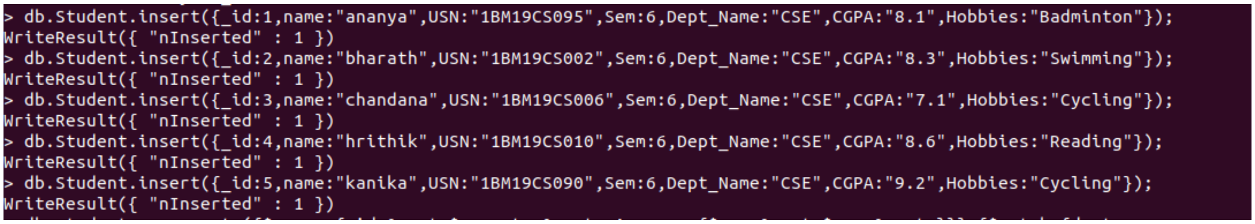
iii) First Filter on “Dept_Name:CSE” and then group it on “Semester” and

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

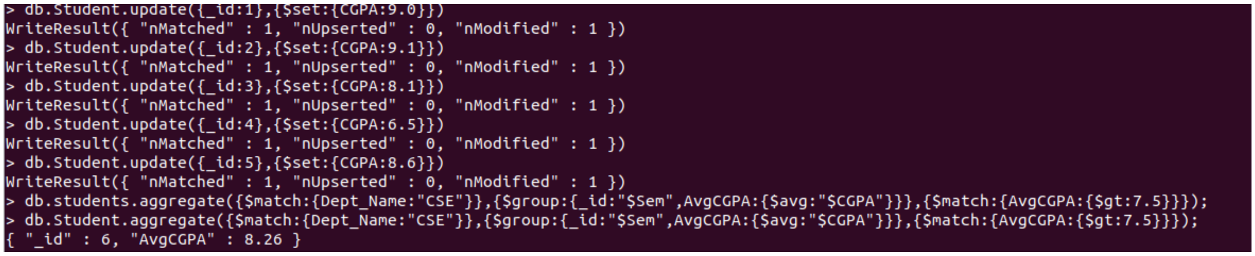
iv) Command used to export MongoDB JSON documents from “Student” Collection into the “Students” database into a CSV fle “Output.txt”.



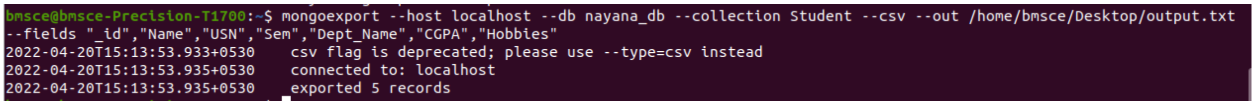
```
> db.createCollection("Student");
{ "ok" : 1 }
```



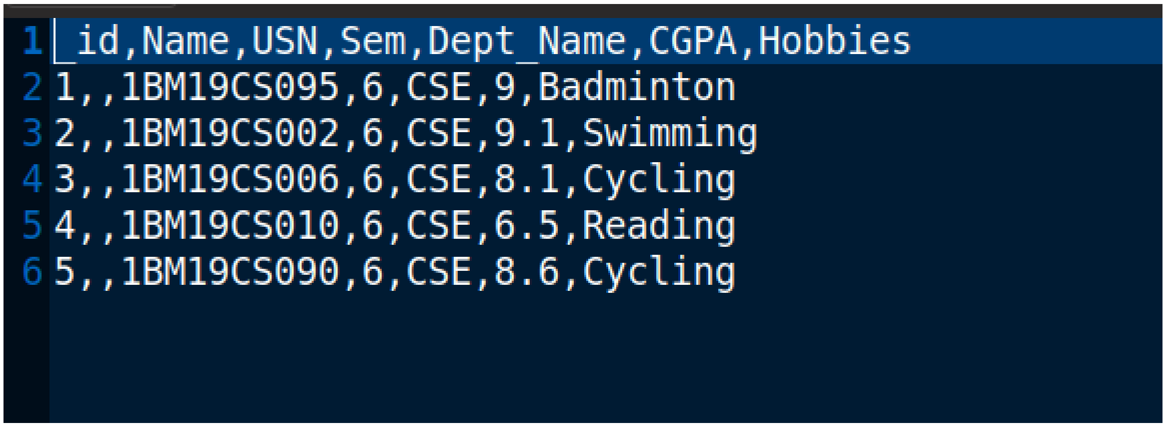
```
> db.Student.insert({_id:1,name:"ananya",USN:"1BM19CS095",Sem:6,Dept_Name:"CSE",CGPA:"8.1",Hobbies:"Badminton"});
WriteResult({ "nInserted" : 1 })
> db.Student.insert({_id:2,name:"bharath",USN:"1BM19CS002",Sem:6,Dept_Name:"CSE",CGPA:"8.3",Hobbies:"Swimming"});
WriteResult({ "nInserted" : 1 })
> db.Student.insert({_id:3,name:"chandana",USN:"1BM19CS006",Sem:6,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.

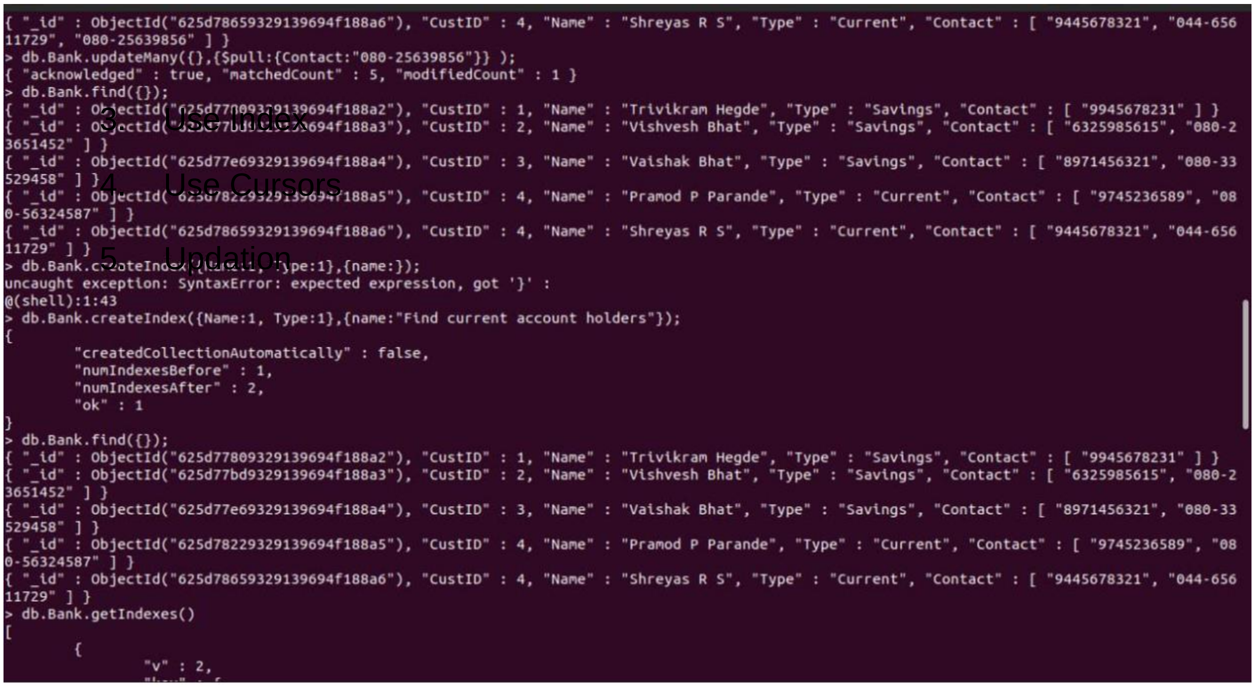
1. Insert three documents

2. Use Arrays(Use Pull and Pop operation)

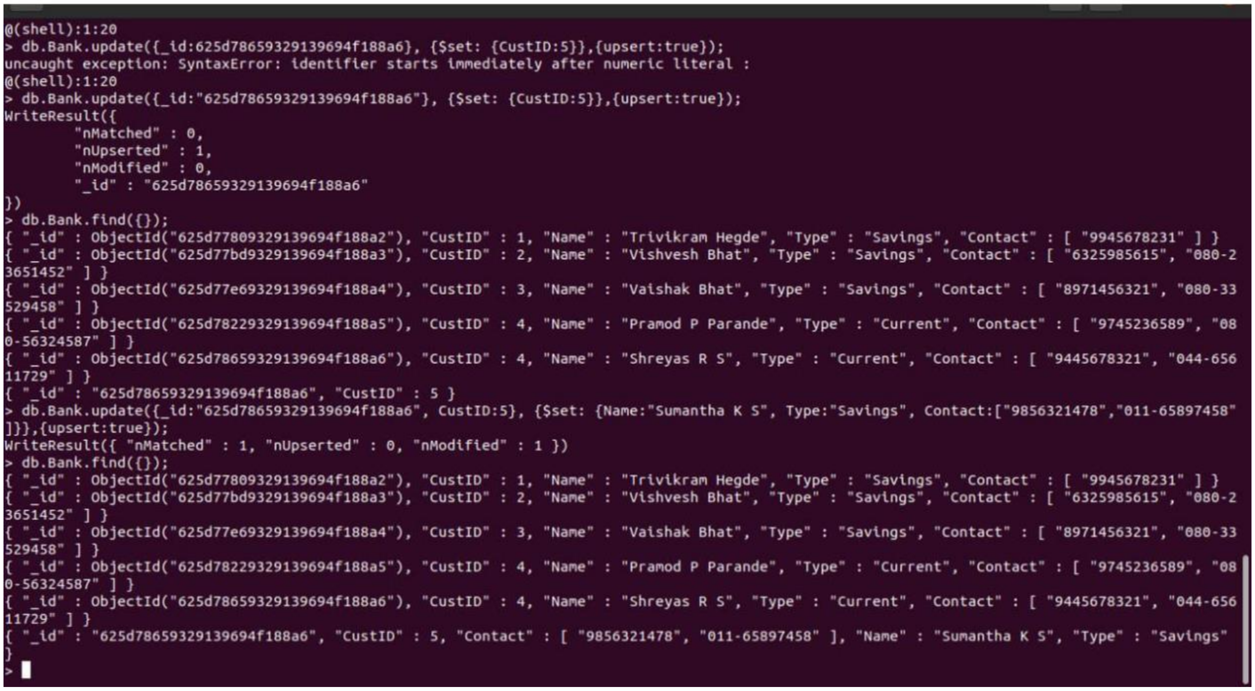
3. Use Index

4. Use Cursors

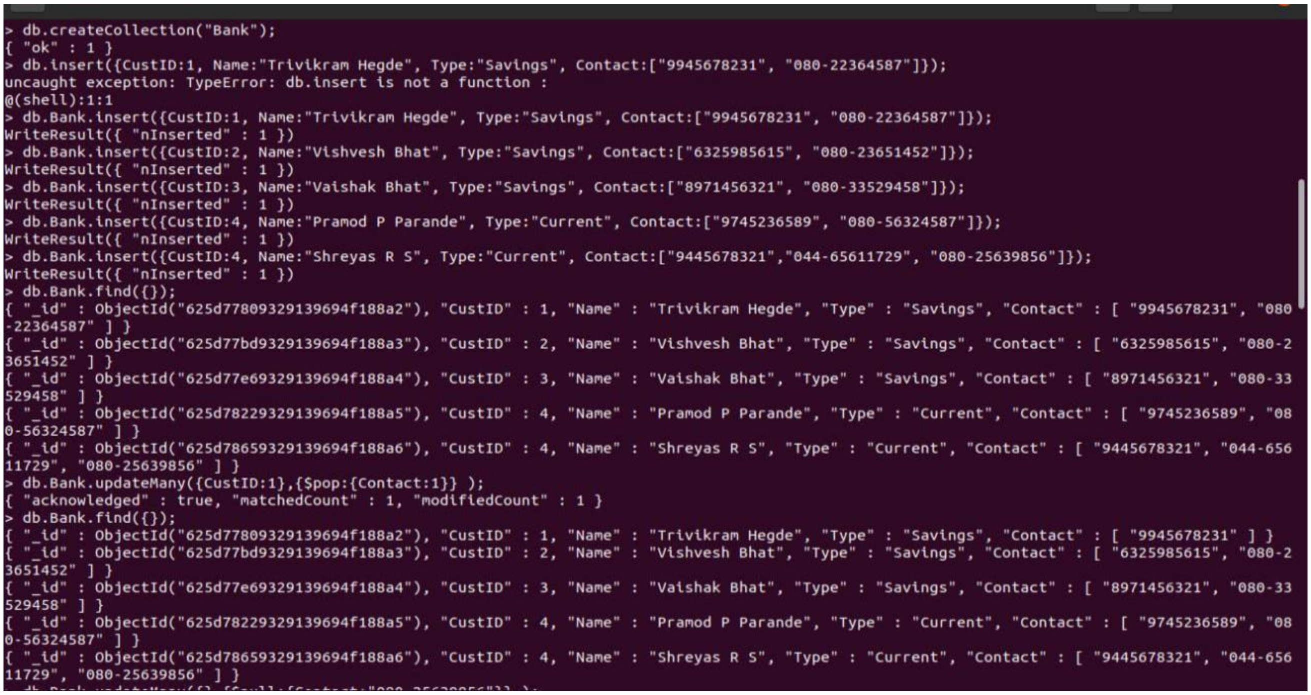
5. Updation



```
{ "_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"
}
>
```



```
> 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" ] }
```

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:

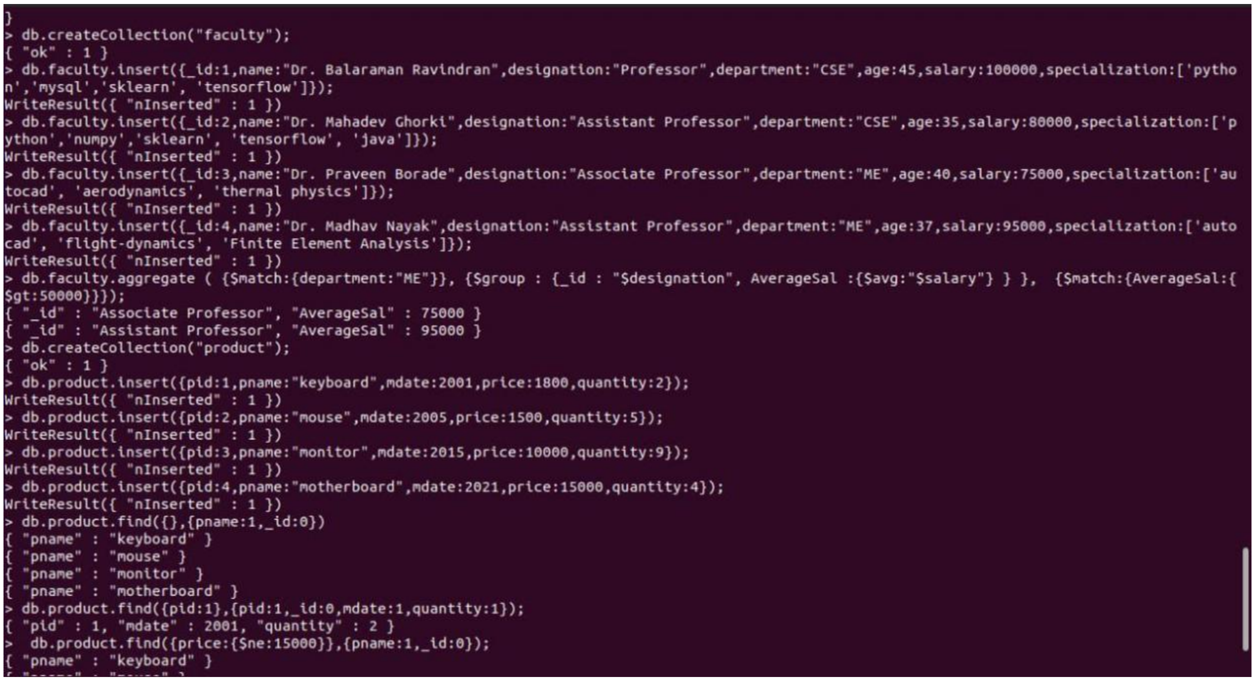
1)To display only the product name from all the documents of the product collection.

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

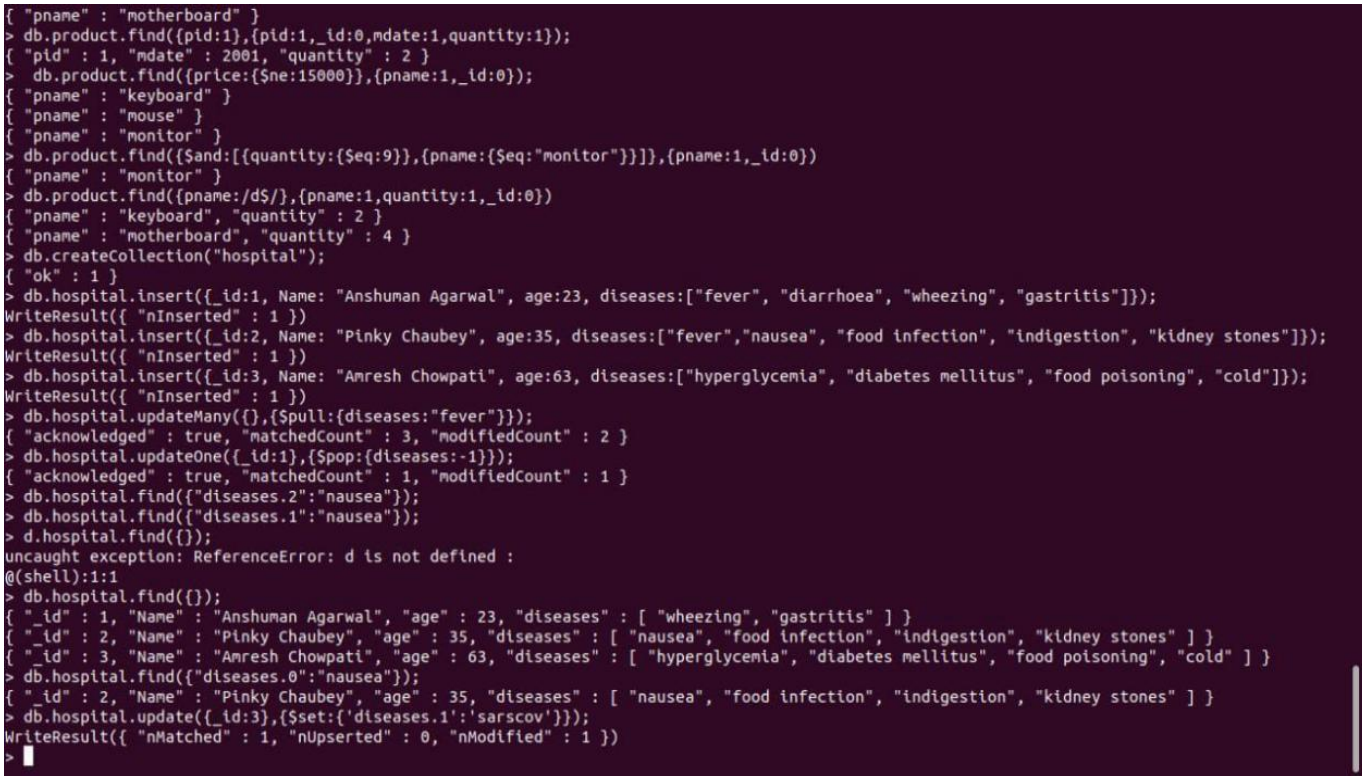
choice.

1. Insert three documents
2. Use Arrays(Use Pull and Pop operation)

3. Use Index
4. Use Cursors

   Updation

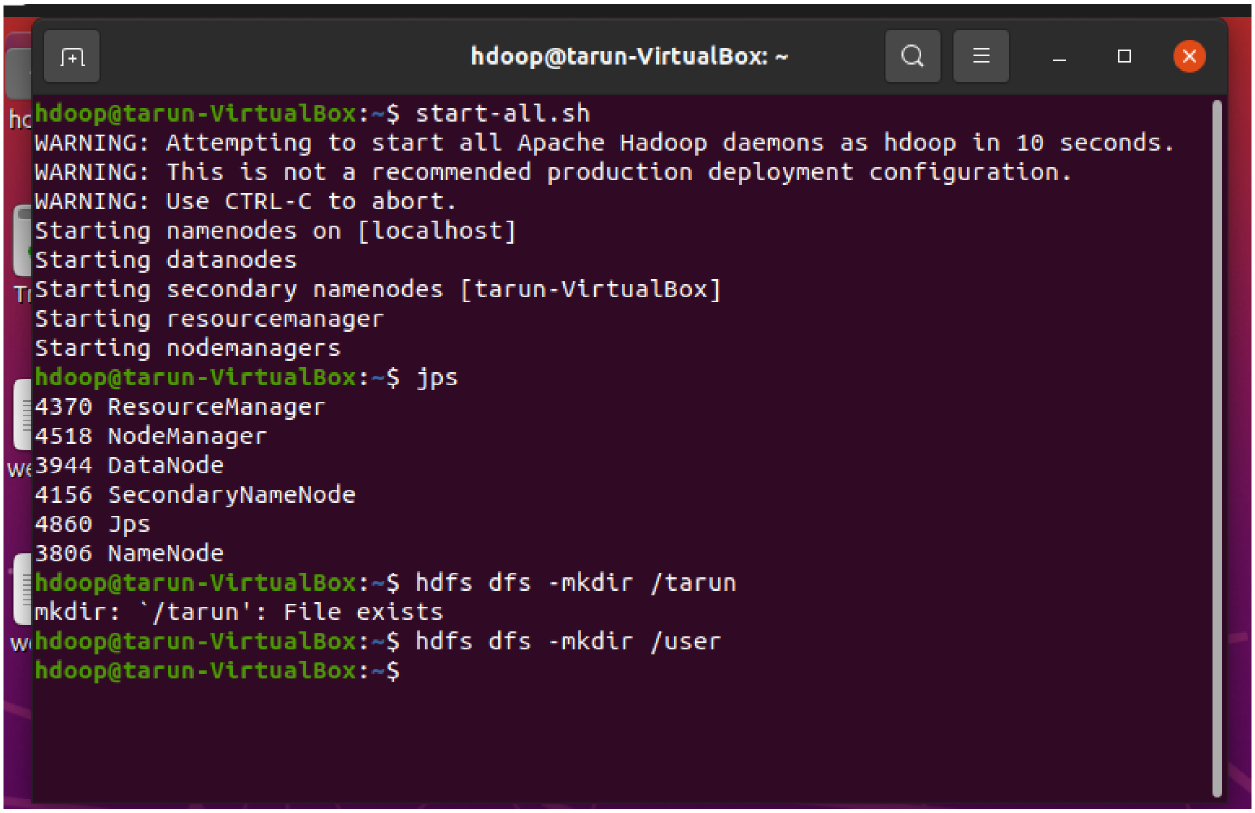
5.



```
{ "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 })
>
```

LAB4

SCREENSHOT OF HADOOP INSTALLATION



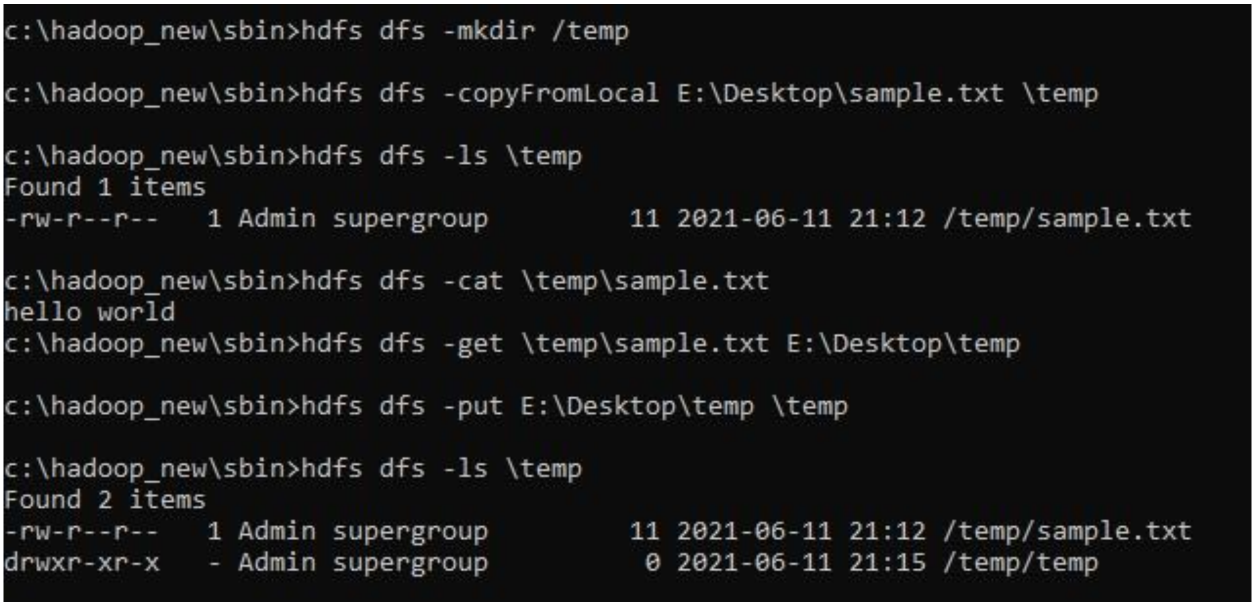
```
hdoop@tarun-VirtualBox:~$ start-all.sh
WARNING: Attempting to start all Apache Hadoop daemons as hdoop in 10 seconds.
WARNING: This is not a recommended production deployment configuration.
WARNING: Use CTRL-C to abort.
Starting namenodes on [localhost]
Starting datanodes
Starting secondary namenodes [tarun-VirtualBox]
Starting resourcemanager
Starting nodemanagers
hdoop@tarun-VirtualBox:~$ jps
4370 ResourceManager
4518 NodeManager
3944 DataNode
4156 SecondaryNameNode
4860 Jps
3806 NameNode
hdoop@tarun-VirtualBox:~$ hdfs dfs -mkdir /tarun
mkdir: `/tarun': File exists
hdoop@tarun-VirtualBox:~$ hdfs dfs -mkdir /user
hdoop@tarun-VirtualBox:~$
```

LAB 5

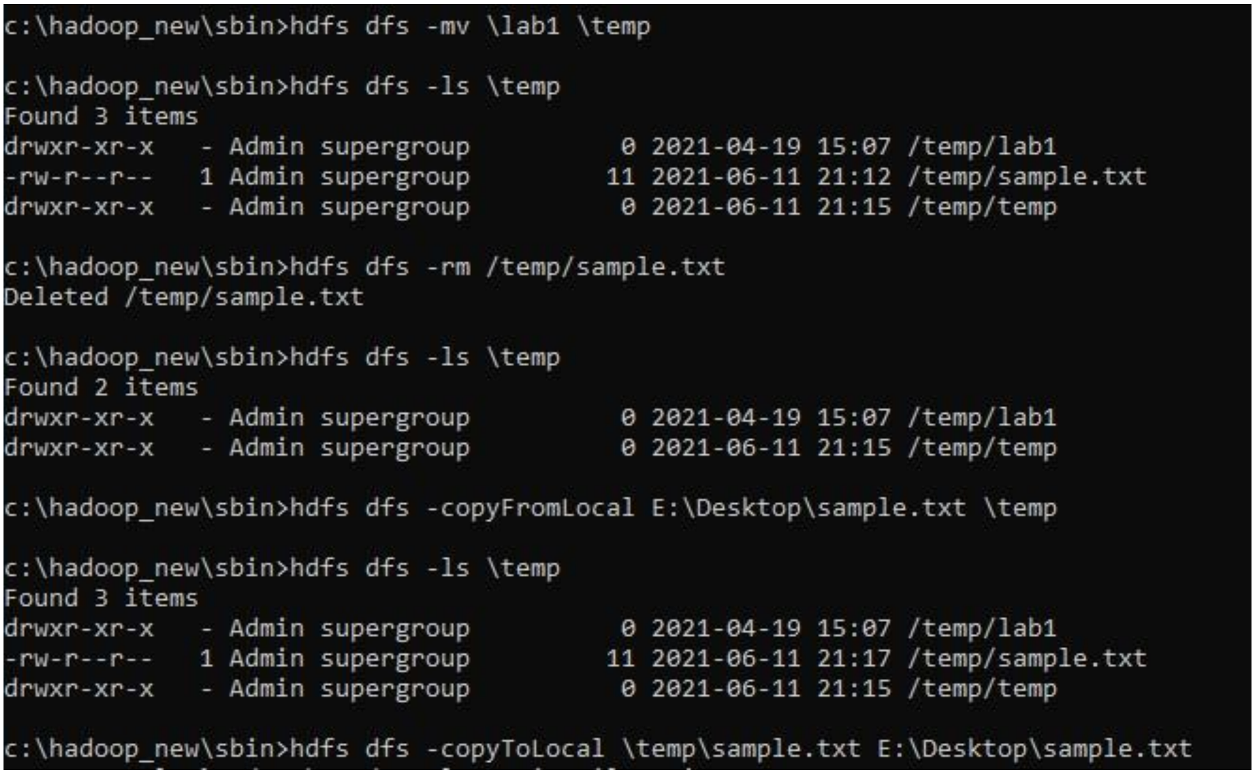
Execution of HDFS Commands for interaction with Hadoop Environment. (Minimum 10 commands to be executed)

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

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

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

c:\hadoop_new\sbin>hdfs dfs -cat \temp\sample.txt hello
world

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

c:\hadoop_new\sbin>hdfs dfs -put E:\Desktop\temp \temp

c:\hadoop_new\sbin>hdfs dfs -ls \temp
Found 2 items
-rw-r--r--   1 Admin supergroup         11 2021-06-11 21:12 /temp/sample.txt drwxr-xr-x   -
Admin supergroup          0 2021-06-11 21:15 /temp/temp

c:\hadoop_new\sbin>hdfs dfs -mv \lab1 \temp
```

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

c:\hadoop_new\sbin>hdfs dfs -rm /temp/sample.txt
Deleted /temp/sample.txt

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

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

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

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



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

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

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

c:\hadoop_new\sbin>hdfs dfs -cat \temp\sample.txt
hello world
c:\hadoop_new\sbin>hdfs dfs -get \temp\sample.txt E:\Desktop\temp

c:\hadoop_new\sbin>hdfs dfs -put E:\Desktop\temp \temp

c:\hadoop_new\sbin>hdfs dfs -ls \temp
Found 2 items
-rw-r--r--   1 Admin supergroup         11 2021-06-11 21:12 /temp/sample.txt
drwxr-xr-x   - Admin supergroup          0 2021-06-11 21:15 /temp/temp
```



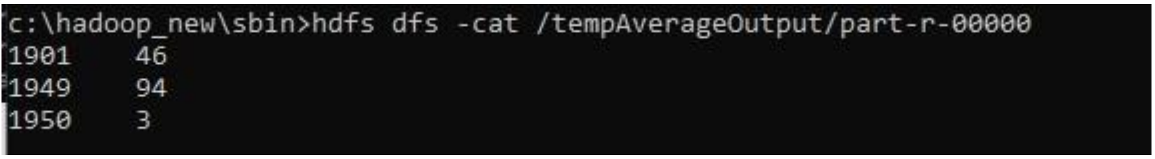
```
c:\hadoop_new\sbin>hdfs dfs -mv \lab1 \temp

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

c:\hadoop_new\sbin>hdfs dfs -rm /temp/sample.txt
Deleted /temp/sample.txt

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

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

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

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

LAB 6

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

```
// AverageDriver.java package temperature;

import org.apache.hadoop.io.*; import org.apache.hadoop.fs.*; import
org.apache.hadoop.mapreduce.*; import org.apache.hadoop.mapreduce.lib.input.FileInputFormat;
import org.apache.hadoop.mapreduce.lib.output.FileOutputFormat;

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

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

//AverageMapper.java package temperature;

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

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

public void map(LongWritable key, Text value, Context context) throws IOException,
InterruptedException
{
        String line = value.toString();   String year = line.substring(15,19);      int temperature;
        if (line.charAt(87)=='+')                 temperature = Integer.parseInt(line.substring(88, 92));
        else
                temperature = Integer.parseInt(line.substring(87, 92));   String quality =
line.substring(92, 93);   if(temperature != MISSING && quality.matches("[01459]"))
        context.write(new Text(year),new IntWritable(temperature)); }
}
```

```
//AverageReducer.java package temperature;

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

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



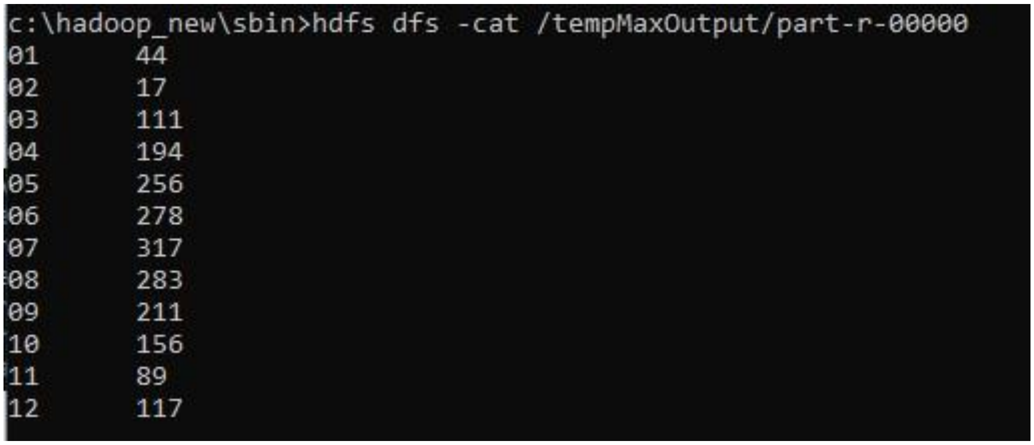
```
c:\hadoop_new\sbin>hdfs dfs -cat /tempAverageOutput/part-r-00000
1901    46
1949    94
1950    3
```

```
//TempDriver.java package
temperatureMax;


import org.apache.hadoop.io.*; import org.apache.hadoop.fs.*; import
org.apache.hadoop.mapreduce.*; import
org.apache.hadoop.mapreduce.lib.input.FileInputFormat; import
org.apache.hadoop.mapreduce.lib.output.FileOutputFormat;


public class TempDriver
{       public static void main (String[] args) throws Exception
        {
                 if (args.length != 2)
                {
```

```
                                System.err.println("Please Enter the input and output parameters");
                               System.exit(-1);
                        }
                        Job job = new Job();
job.setJarByClass(TempDriver.class);                    job.setJobName("Max
temperature");
                          FileInputFormat.addInputPath(job,new Path(args[0]));
                          FileOutputFormat.setOutputPath(job,new Path (args[1]));


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

//TempMapper.java package
temperatureMax;

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

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

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

```
            String line = value.toString();      String month = line.substring(19,21);
int temperature;            if (line.charAt(87)=='+')                         temperature =
Integer.parseInt(line.substring(88, 92));
            else
                        temperature = Integer.parseInt(line.substring(87, 92));   String
quality = line.substring(92, 93);  if(temperature != MISSING &&
quality.matches("[01459]"))                  context.write(new Text(month),new
IntWritable(temperature)); }

}


//TempReducer.java package
temperatureMax;


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


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


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

```
        }
}
```



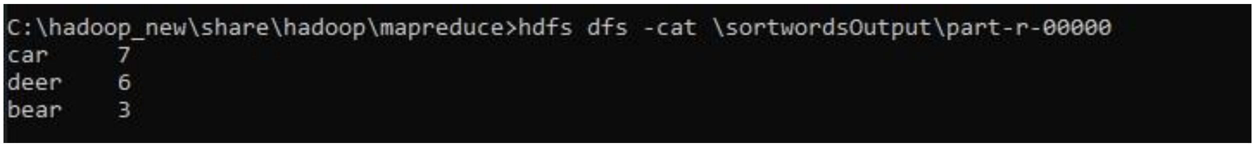
```
c:\hadoop_new\sbin>hdfs dfs -cat /tempMaxOutput/part-r-00000
01      44
02      17
03      111
04      194
05      256
06      278
07      317
08      283
09      211
10      156
11      89
12      117
```

LAB 7

For a given Text file, create a Map Reduce program to sort the content in an alphabetic order listing only top ‘n’ maximum occurrence of words.

```
// TopN.java package sortWords;

import org.apache.hadoop.conf.Configuration; import org.apache.hadoop.fs.Path; import
org.apache.hadoop.io.IntWritable; import org.apache.hadoop.io.Text; import
org.apache.hadoop.mapreduce.Job; import org.apache.hadoop.mapreduce.Mapper; import
org.apache.hadoop.mapreduce.Reducer; import
org.apache.hadoop.mapreduce.lib.input.FileInputFormat; import
org.apache.hadoop.mapreduce.lib.output.FileOutputFormat; import
org.apache.hadoop.util.GenericOptionsParser; import utils.MiscUtils;

import java.io.IOException; import java.util.*;

public class TopN {

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

  /**
   * The mapper reads one line at the time, splits it into an array of single words and emits every      *
word to the reducers with the value of 1.
   */
  public static class TopNMapper extends Mapper<Object, Text, Text, IntWritable> {

    private final static IntWritable one = new IntWritable(1);      private Text word = new Text();
    private String tokens = "[_|$#<>\\^=\\[\\]\\*/\\\\,;,.\\-:()?!\"']";

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

  /**
   * The reducer retrieves every word and puts it into a Map: if the word already exists in the    * map,
increments its value, otherwise sets it to 1.
   */
  public static class TopNReducer extends Reducer<Text, IntWritable, Text, IntWritable> {

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

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

      // computes the number of occurrences of a single word       int sum = 0;       for
(IntWritable val : values) {         sum += val.get();
      }
      // puts the number of occurrences of this word into the map.
      // We need to create another Text object because the Text instance
      // we receive is the same for all the words       countMap.put(new Text(key), new
IntWritable(sum));
    }
@Override
    protected void cleanup(Context context) throws IOException, InterruptedException {

      Map<Text, IntWritable> sortedMap = MiscUtils.sortByValues(countMap);

      int counter = 0;       for (Text key : sortedMap.keySet()) {         if (counter++ == 3) {
break;
        }
        context.write(key, sortedMap.get(key));
      }
    }
  }

  /**
   * The combiner retrieves every word and puts it into a Map: if the word already exists in the    *
map, increments its value, otherwise sets it to 1.
   */
  public static class TopNCombiner extends Reducer<Text, IntWritable, Text, IntWritable> {

    @Override
```

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

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

// MiscUtils.java package utils;

import java.util.*;

public class MiscUtils {

    /**
sorts the map by values. Taken from:
http://javarevisited.blogspot.it/2012/12/how-to-sort-hashmap-java-by-key-and-value.html
     */
    public static <K extends Comparable, V extends Comparable> Map<K, V> sortByValues(Map<K, V>
map) {
        List<Map.Entry<K, V>> entries = new LinkedList<Map.Entry<K, V>>(map.entrySet());

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

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

        //LinkedHashMap will keep the keys in the order they are inserted
        //which is currently sorted on natural ordering
        Map<K, V> sortedMap = new LinkedHashMap<K, V>();
for (Map.Entry<K, V> entry : entries) {
            sortedMap.put(entry.getKey(), entry.getValue());
        }

        return sortedMap;
    }
}
```



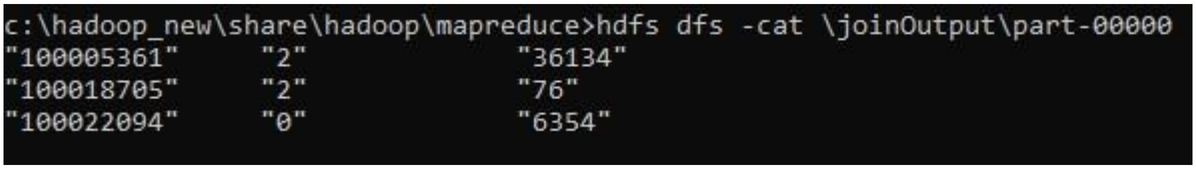
```
C:\hadoop_new\share\hadoop\mapreduce>hdfs dfs -cat \sortwordsOutput\part-r-00000
car     7
deer    6
bear    3
```

LAB 8

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

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


public class JoinDriver extends Configured implements Tool {

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

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

@Override public int run(String[] args) throws Exception {                  if (args.length != 3) {
                        System.out.println("Usage: <Department Emp Strength input>
<Department Name input> <output>");
                        return -1;
                }

                JobConf conf = new JobConf(getConf(), getClass());                  conf.setJobName("Join
'Department Emp Strength input' with 'Department Name input'");

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

                MultipleInputs.addInputPath(conf, AInputPath, TextInputFormat.class,
Posts.class);
                MultipleInputs.addInputPath(conf, BInputPath, TextInputFormat.class,
User.class);

                FileOutputFormat.setOutputPath(conf, outputPath);

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

                conf.setMapOutputKeyClass(TextPair.class);
                conf.setReducerClass(JoinReducer.class);

                conf.setOutputKeyClass(Text.class);

        JobClient.runJob(conf);

                return 0;
        }

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

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

// JoinReducer.java import java.io.IOException; import java.util.Iterator;

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

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

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

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

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

import org.apache.hadoop.io.IntWritable;

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

        @Override
```

```
 public void map(LongWritable key, Text value, OutputCollector<TextPair, Text> output, Reporter
reporter)
                                throws IOException
            {

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

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

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

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

            @Override
 public void map(LongWritable key, Text value, OutputCollector<TextPair, Text> output, Reporter
reporter)
                                throws IOException
            {
                        String valueString = value.toString();
                        String[] SingleNodeData = valueString.split("\t");                        output.collect(new
TextPair(SingleNodeData[3], "0"), new
Text(SingleNodeData[9]));
            }
}

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

import org.apache.hadoop.io.*;
public class TextPair implements WritableComparable<TextPair> {

  private Text first;   private Text second;

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

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

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

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

  public Text getSecond() {     return second;
  }

  @Override
  public void write(DataOutput out) throws IOException {     first.write(out);     second.write(out);
  }

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

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

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

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

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

  // vv TextPairComparator   public static class Comparator extends WritableComparator {

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

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

   @Override     public int compare(byte[] b1, int s1, int l1,                     byte[] b2, int s2, int l2) {
           try {
       int firstL1 = WritableUtils.decodeVIntSize(b1[s1]) + readVInt(b1, s1);        int firstL2 =
WritableUtils.decodeVIntSize(b2[s2]) + readVInt(b2, s2);        int cmp =
TEXT_COMPARATOR.compare(b1, s1, firstL1, b2, s2, firstL2);        if (cmp != 0) {          return cmp;
       }
       return TEXT_COMPARATOR.compare(b1, s1 + firstL1, l1 - firstL1,
                               b2, s2 + firstL2, l2 - firstL2);
     } catch (IOException e) {        throw new IllegalArgumentException(e);
     }
   }
  }

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

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

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

   @Override     public int compare(byte[] b1, int s1, int l1,                      byte[] b2, int s2, int l2) {
           try {
       int firstL1 = WritableUtils.decodeVIntSize(b1[s1]) + readVInt(b1, s1);        int firstL2 =
WritableUtils.decodeVIntSize(b2[s2]) + readVInt(b2, s2);        return TEXT_COMPARATOR.compare(b1,
s1, firstL1, b2, s2, firstL2);
     } catch (IOException e) {        throw new IllegalArgumentException(e);
     }
   }

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



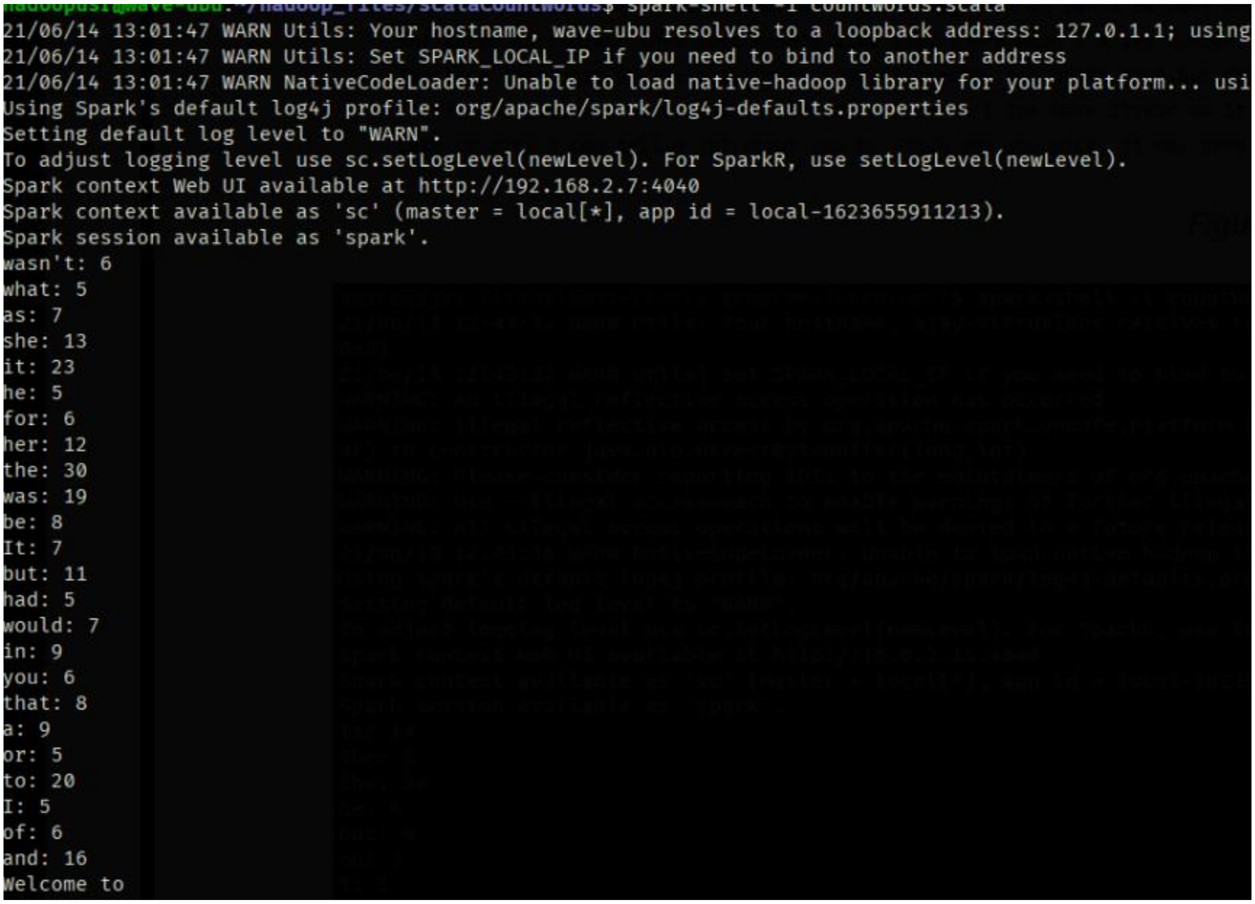
```
c:\hadoop_new\share\hadoop\mapreduce>hdfs dfs -cat \joinOutput\part-00000
"100005361"     "2"             "36134"
"100018705"     "2"             "76"
"100022094"     "0"             "6354"
```

LAB 9

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

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

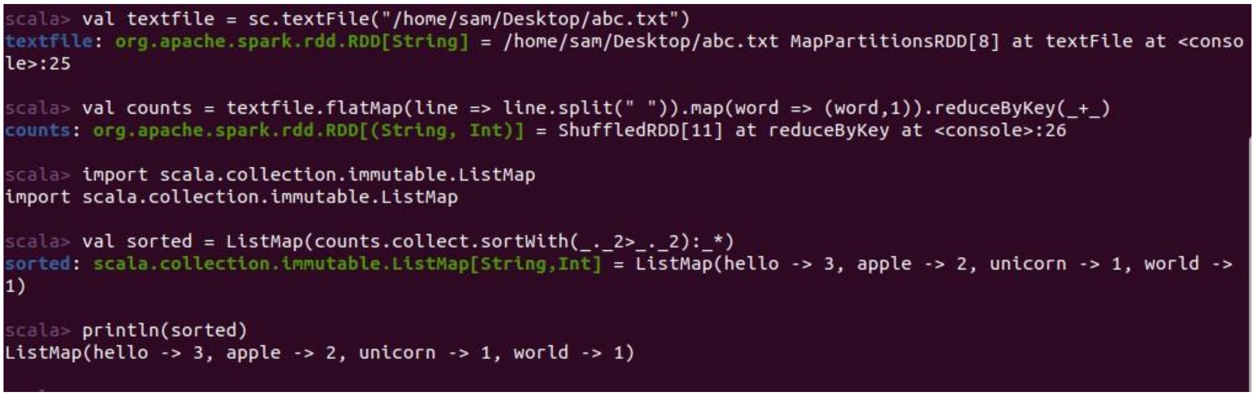
```
val data=sc.textFile("sparkdata.txt")
data.collect;
val splitdata = data.flatMap(line => line.split(" "));
splitdata.collect;
val mapdata = splitdata.map(word => (word,1));
mapdata.collect;
val reducedata = mapdata.reduceByKey(_+_);
reducedata.collect;
```



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

LAB 10

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



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

scala> val counts = textfile.flatMap(line => line.split(" ")).map(word => (word,1)).reduceByKey(_+_)
counts: org.apache.spark.rdd.RDD[(String, Int)] = ShuffledRDD[11] at reduceByKey at <console>:26

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

scala> val sorted = ListMap(counts.collect.sortWith(_._2>_._2):_*)
sorted: scala.collection.immutable.ListMap[String,Int] = ListMap(hello -> 3, apple -> 2, unicorn -> 1, world ->
1)

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