VISVESVARAYA TECHNOLOGICAL UNIVERSITY
"JnanaSangama", Belgaum -590014, Karnataka.

LAB REPORT
on

BIG DATA ANALYTICS (20CS6PEBDA)

Submitted by

MANIKANTH LAKSHMAN SHETTY (1BM19CS082)

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 **MANIKANTH LAKSHMAN SHETTY(1BM19CS082),** 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 Big data
analytics - (20CS6PEBDA) work prescribed for the said degree.

Name of the Lab-In charge	Prof. Pallavi G B
Designation	Assistant Professor
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

LAB 1:

.CREATE DATABASE IN MONGODB.

```
> 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”); **=>** *sql equivalent*
CREATE TABLE STUDENT(…);

```
{ "ok" : 1 }
```

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:"InternetSurfing"});
```

```
WriteResult({ "nInserted" : 1 })
```

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:"Skating"}},{upsert:true});
```

```
WriteResult({ "nMatched" : 0, "nUpserted" : 1, "nModified" : 0, "_id" : 3 })
```

5.FIND METHOD

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

```
db.Student.find({StudName:"AryanDavid"});
({cond..},{columns.. column:1, columnname:0}   )
```

```
{ "_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});

{ "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:'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: ['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();

```
{
   "_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();

```
{
   "_id" : 1,
   "StudName" : "MichelleJacintha",
   "Grade" : "VII",
   "Hobbies" : "InternetSurfing"
}
```

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

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();

```
{
    "_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
```

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.Student.save({StudName:”Vamsi”, Grade:”VI”})
```

```
WriteResult({ "nInserted" : 1 })
```

VI. **Add a new field to existing Document:**

```
db.Student.update({_id:ObjectId("625695cc7d129fb98b44c8a1")},
{$set:{Location:"Network"}})
```

```
WriteResult({ "nMatched" : 1, "nUpserted" : 0, "nModified" : 1 })
```

VII. **Remove the field in an existing Document**

```
db.Student.update({_id:ObjectId("625695cc7d129fb98b44c8a1")},
{$unset:{Location:”Network”}})
```

```
WriteResult({ "nMatched" : 1, "nUpserted" : 0, "nModified" : 1 })
```

VIII. **Finding Document based on search criteria suppressing few fields**

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

```
{ "StudName" : "MichelleJacintha", "Grade" : "VII" }
```

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

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

```
{
    "_id" : ObjectId("625695cc7d129fb98b44c8a1"),
    "StudName" : "Vamsi",
    "Grade" : "VI"
}
```

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

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

```
{
    "_id" : 1,
    "StudName" : "MichelleJacintha",
    "Grade" : "VII",
    "Hobbies" : "InternetSurfing"
}
```

IX. to set a particular field value to NULL

```
db.Student.update({_id:3},{$set:{Location:null}})
```

```
WriteResult({ "nMatched" : 1, "nUpserted" : 0, "nModified" : 1 })
```

X. **Count the number of documents in Student Collections**

```
db.Student.count()
```

```
3
```

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

```
db.Student.count({Grade:”VII”})
```

```
2
```

retrieve first 3 documents

```
db.Student.find({Grade:”VII”}).limit(1).pretty();
```

```
{
 "_id" : 1,
 "StudName" : "MichelleJacintha",
 "Grade" : "VII",
 "Hobbies" : "InternetSurfing"
}
```

Sort the document in Ascending order

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

```
{
   "_id" : 3,
   "Grade" : "VII",
   "StudName" : "AryanDavid",
   "Hobbies" : "Skating",
   "Location" : null
}
{
   "_id" : 1,
   "StudName" : "MichelleJacintha",
   "Grade" : "VII",
   "Hobbies" : "InternetSurfing"
}
{
   "_id" : ObjectId("625695cc7d129fb98b44c8a1"),
   "StudName" : "Vamsi",
   "Grade" : "VI"
}
```

Note:
for desending order :

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

to Skip the 1st two documents from the Students Collections

```
db.Student.find().skip(2).pretty()
```

```
{
    "_id" : ObjectId("625695cc7d129fb98b44c8a1"),
    "StudName" : "Vamsi",
    "Grade" : "VI"
}
```

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'] } )
```

```
{ "_id" : 1, "fruits" : [ "grapes", "mango", "apple" ] }
{ "_id" : 2, "fruits" : [ "grapes", "mango", "cherry" ] }
{ "_id" : 3, "fruits" : [ "banana", "mango" ] }
```

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();
```

```
{ "_id" : 1, "fruits" : [ "grapes", "mango", "apple" ] }
```

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

```
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}} )
```

```
{ "_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}})
```

```
{ "_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”]}})
```

```
{ "_id" : 1, "fruits" : [ "grapes", "mango", "apple" ] }
{ "_id" : 2, "fruits" : [ "grapes", "mango", "cherry" ] }
```

update on Array:
using particular id replace the element present in the 1st index position of the fruits array with apple

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

```
WriteResult({ "nMatched" : 1, "nUpserted" : 0, "nModified" : 1 })
```

insert new key value pairs in the fruits array

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

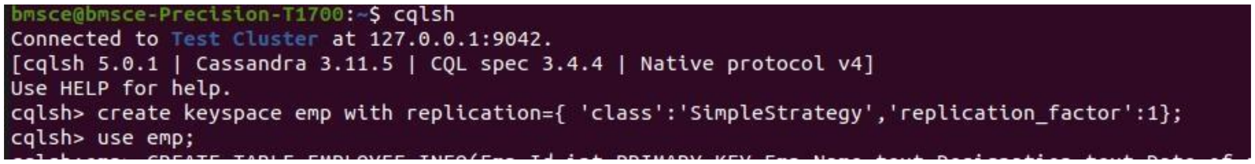
```
{ "_id" : 1, "fruits" : [ "grapes", "mango", "apple" ] }
{ "_id" : 2, "fruits" : [ "grapes", "mango", "cherry" ], "price" : [ {
"grapes" : 80, "mango" : 200, "cherry" : 100 } ] }
{ "_id" : 3, "fruits" : [ "banana", "apple" ] }
```

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

LAB 2:

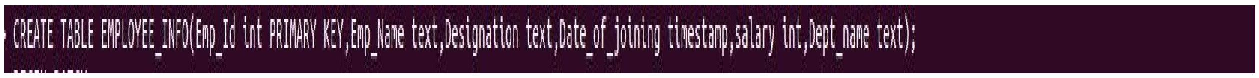
Perform the following DB operations using Cassandra.

1. Create a key space by name Employee



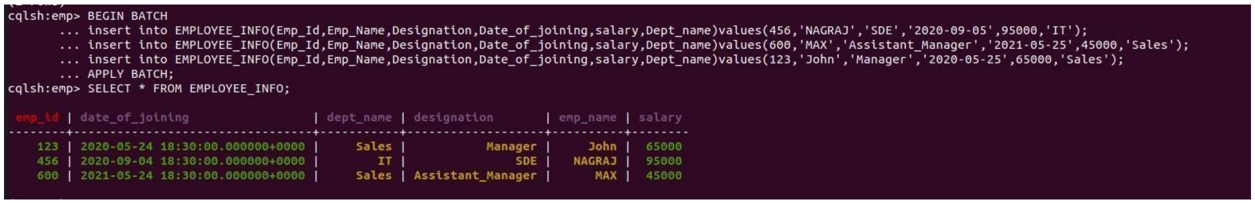
```
bmsce@bmsce-Precision-T1700:~$ cqlsh
Connected to Test Cluster at 127.0.0.1:9042.
[cqlsh 5.0.1 | Cassandra 3.11.5 | CQL spec 3.4.4 | Native protocol v4]
Use HELP for help.
cqlsh> create keyspace emp with replication={ 'class':'SimpleStrategy','replication_factor':1};
cqlsh> use emp;
```

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



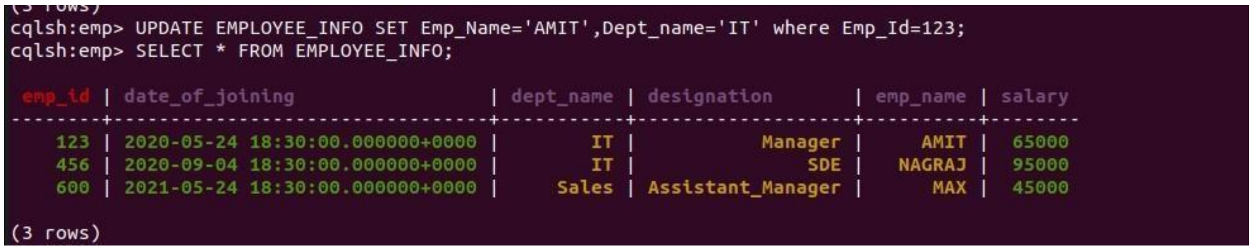
```
CREATE TABLE EMPLOYEE_INFO(Emp_Id int PRIMARY KEY,Emp_Name text,Designation text,Date_of_joining timestamp,salary int,Dept_name text);
```

3. Insert the values into the table in batch



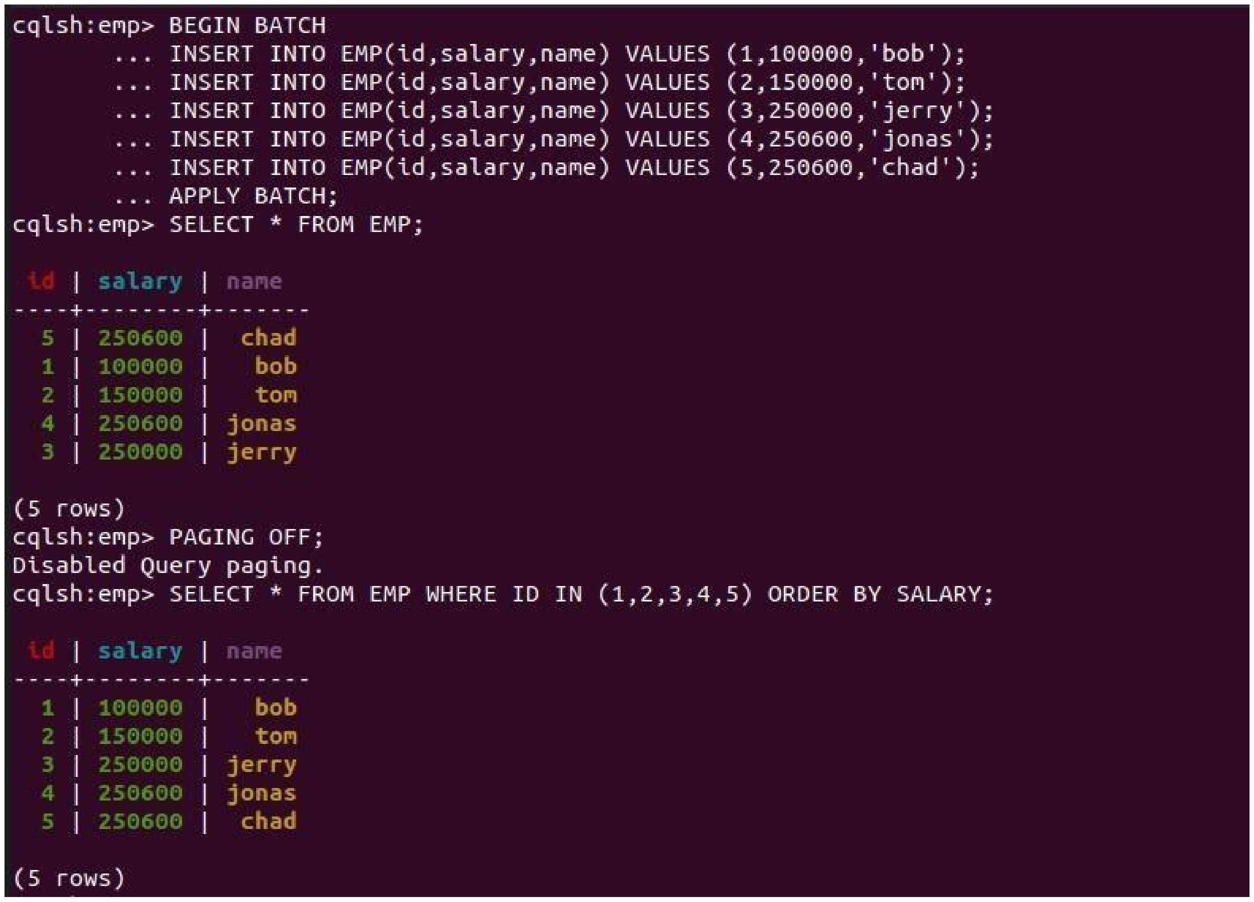
```
cqlsh:emp> BEGIN BATCH
       ... insert into EMPLOYEE_INFO(Emp_Id,Emp_Name,Designation,Date_of_joining,salary,Dept_name)values(456,'NAGRAJ','SDE','2020-09-05',95000,'IT');
       ... insert into EMPLOYEE_INFO(Emp_Id,Emp_Name,Designation,Date_of_joining,salary,Dept_name)values(600,'MAX','Assistant_Manager','2021-05-25',45000,'Sales');
       ... insert into EMPLOYEE_INFO(Emp_Id,Emp_Name,Designation,Date_of_joining,salary,Dept_name)values(123,'John','Manager','2020-05-25',65000,'Sales');
       ... APPLY BATCH;
cqlsh:emp> SELECT * FROM EMPLOYEE_INFO;

 emp_id | date_of_joining                 | dept_name | designation       | emp_name | salary
--------+---------------------------------+-----------+-------------------+----------+--------
    123 | 2020-05-24 18:30:00.000000+0000 |     Sales |           Manager |     John |  65000
    456 | 2020-09-04 18:30:00.000000+0000 |        IT |               SDE |   NAGRAJ |  95000
    600 | 2021-05-24 18:30:00.000000+0000 |     Sales | Assistant_Manager |      MAX |  45000
```

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



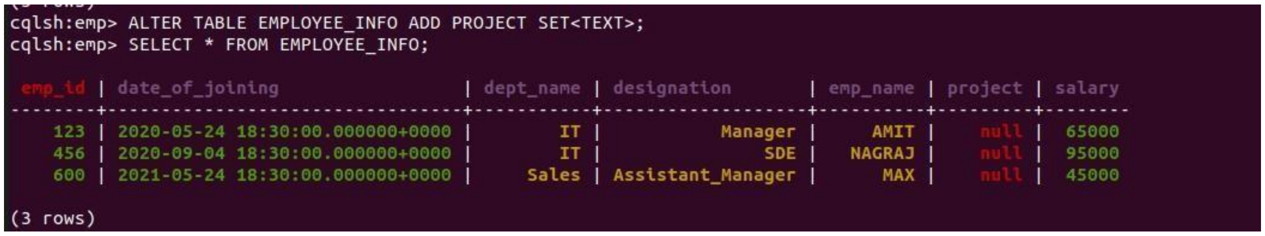
```
cqlsh:emp> UPDATE EMPLOYEE_INFO SET Emp_Name='AMIT',Dept_name='IT' where Emp_Id=123;
cqlsh:emp> SELECT * FROM EMPLOYEE_INFO;

 emp_id | date_of_joining                 | dept_name | designation       | emp_name | salary
--------+---------------------------------+-----------+-------------------+----------+--------
    123 | 2020-05-24 18:30:00.000000+0000 |        IT |           Manager |     AMIT |  65000
    456 | 2020-09-04 18:30:00.000000+0000 |        IT |               SDE |   NAGRAJ |  95000
    600 | 2021-05-24 18:30:00.000000+0000 |     Sales | Assistant_Manager |      MAX |  45000

(3 rows)
```

5. Sort the details of Employee records based on salary



```
cqlsh:emp> BEGIN BATCH
       ... INSERT INTO EMP(id,salary,name) VALUES (1,100000,'bob');
       ... INSERT INTO EMP(id,salary,name) VALUES (2,150000,'tom');
       ... INSERT INTO EMP(id,salary,name) VALUES (3,250000,'jerry');
       ... INSERT INTO EMP(id,salary,name) VALUES (4,250600,'jonas');
       ... INSERT INTO EMP(id,salary,name) VALUES (5,250600,'chad');
       ... APPLY BATCH;
cqlsh:emp> SELECT * FROM EMP;

 id | salary | name
----+--------+-------
  5 | 250600 |  chad
  1 | 100000 |   bob
  2 | 150000 |   tom
  4 | 250600 | jonas
  3 | 250000 | jerry

(5 rows)
cqlsh:emp> PAGING OFF;
Disabled Query paging.
cqlsh:emp> SELECT * FROM EMP WHERE ID IN (1,2,3,4,5) ORDER BY SALARY;

 id | salary | name
----+--------+-------
  1 | 100000 |   bob
  2 | 150000 |   tom
  3 | 250000 | jerry
  4 | 250600 | jonas
  5 | 250600 |  chad

(5 rows)
```

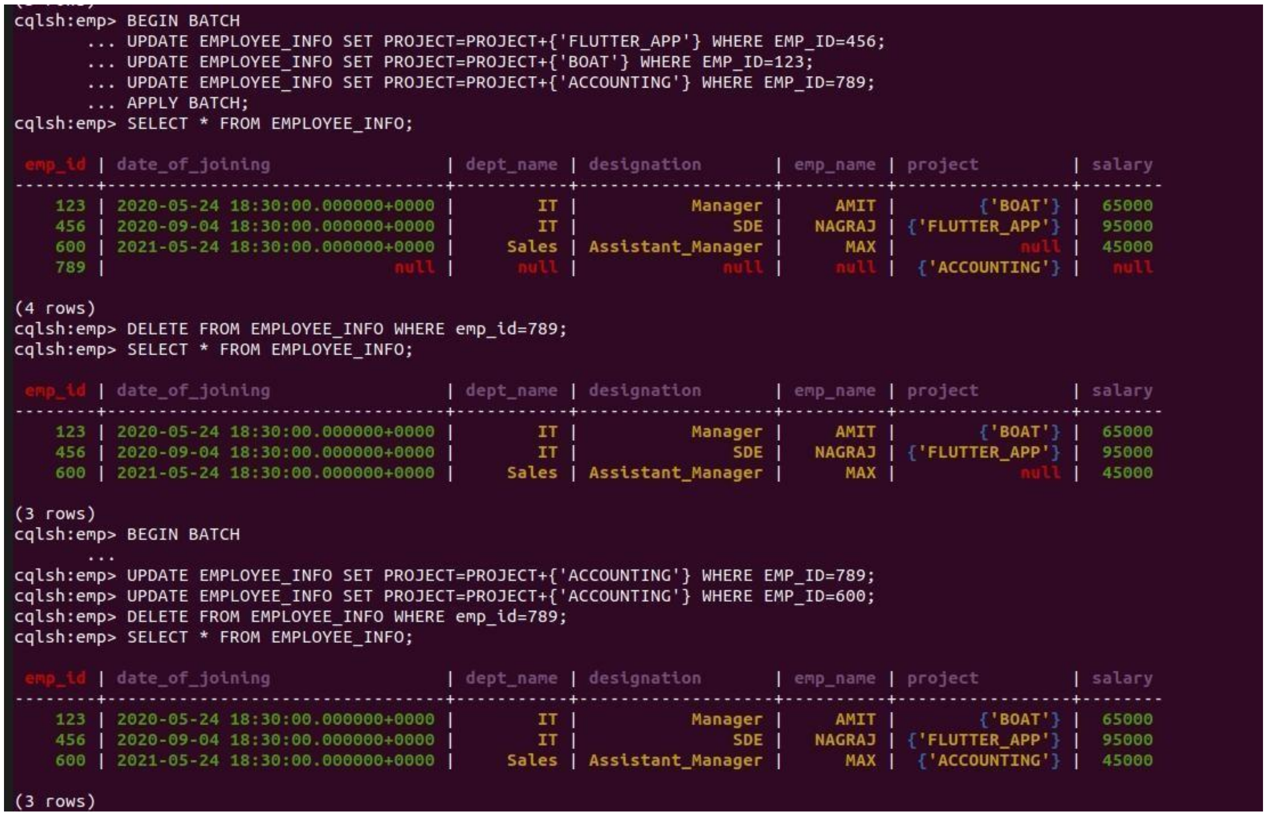
## 6.	Alter the schema of the table Employee_Info to add a column Projects which stores a set of Projects done by the corresponding



```
cqlsh:emp> ALTER TABLE EMPLOYEE_INFO ADD PROJECT SET<TEXT>;
cqlsh:emp> SELECT * FROM EMPLOYEE_INFO;

 emp_id | date_of_joining                 | dept_name | designation       | emp_name | project | salary
--------+---------------------------------+-----------+-------------------+----------+---------+--------
    123 | 2020-05-24 18:30:00.000000+0000 |        IT |           Manager |     AMIT |    null |  65000
    456 | 2020-09-04 18:30:00.000000+0000 |        IT |               SDE |   NAGRAJ |    null |  95000
    600 | 2021-05-24 18:30:00.000000+0000 |     Sales | Assistant_Manager |      MAX |    null |  45000

(3 rows)
```

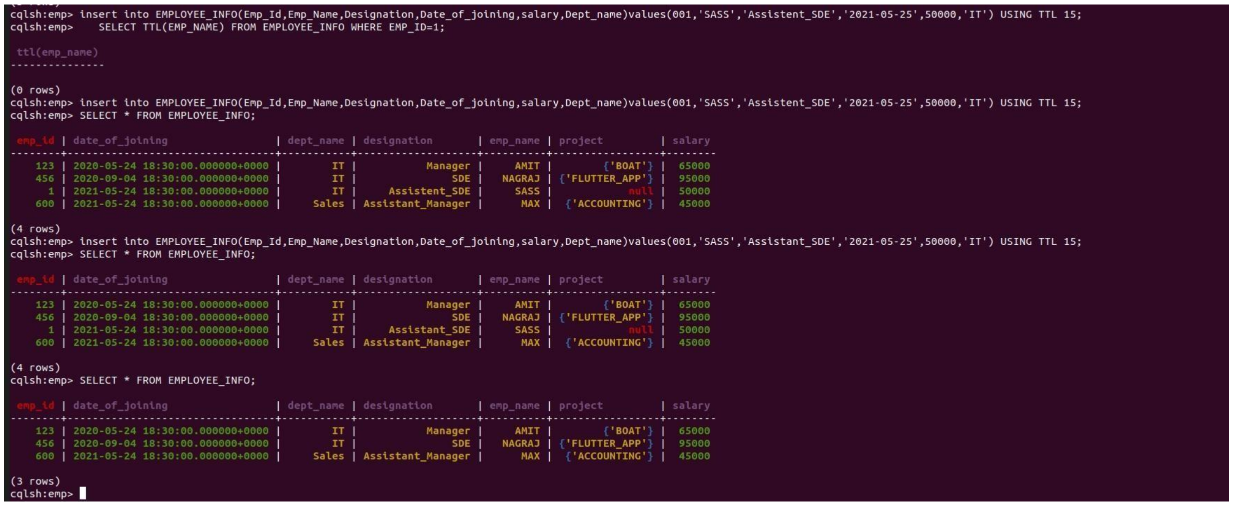
Employee.

7. Update the altered table to add project names.



```
cqlsh:emp> BEGIN BATCH
       ... UPDATE EMPLOYEE_INFO SET PROJECT=PROJECT+{'FLUTTER_APP'} WHERE EMP_ID=456;
       ... UPDATE EMPLOYEE_INFO SET PROJECT=PROJECT+{'BOAT'} WHERE EMP_ID=123;
       ... UPDATE EMPLOYEE_INFO SET PROJECT=PROJECT+{'ACCOUNTING'} WHERE EMP_ID=789;
       ... APPLY BATCH;
cqlsh:emp> SELECT * FROM EMPLOYEE_INFO;

 emp_id | date_of_joining                 | dept_name | designation       | emp_name | project           | salary
--------+---------------------------------+-----------+-------------------+----------+-------------------+--------
    123 | 2020-05-24 18:30:00.000000+0000 |        IT |           Manager |     AMIT |          {'BOAT'} |  65000
    456 | 2020-09-04 18:30:00.000000+0000 |        IT |               SDE |   NAGRAJ | {'FLUTTER_APP'} |  95000
    600 | 2021-05-24 18:30:00.000000+0000 |     Sales | Assistant_Manager |      MAX |              null |  45000
    789 |                            null |      null |              null |     null |  {'ACCOUNTING'} |   null

(4 rows)
cqlsh:emp> DELETE FROM EMPLOYEE_INFO WHERE emp_id=789;
cqlsh:emp> SELECT * FROM EMPLOYEE_INFO;

 emp_id | date_of_joining                 | dept_name | designation       | emp_name | project           | salary
--------+---------------------------------+-----------+-------------------+----------+-------------------+--------
    123 | 2020-05-24 18:30:00.000000+0000 |        IT |           Manager |     AMIT |          {'BOAT'} |  65000
    456 | 2020-09-04 18:30:00.000000+0000 |        IT |               SDE |   NAGRAJ | {'FLUTTER_APP'} |  95000
    600 | 2021-05-24 18:30:00.000000+0000 |     Sales | Assistant_Manager |      MAX |              null |  45000

(3 rows)
cqlsh:emp> BEGIN BATCH
       ...
cqlsh:emp> UPDATE EMPLOYEE_INFO SET PROJECT=PROJECT+{'ACCOUNTING'} WHERE EMP_ID=789;
cqlsh:emp> UPDATE EMPLOYEE_INFO SET PROJECT=PROJECT+{'ACCOUNTING'} WHERE EMP_ID=600;
cqlsh:emp> DELETE FROM EMPLOYEE_INFO WHERE emp_id=789;
cqlsh:emp> SELECT * FROM EMPLOYEE_INFO;

 emp_id | date_of_joining                 | dept_name | designation       | emp_name | project           | salary
--------+---------------------------------+-----------+-------------------+----------+-------------------+--------
    123 | 2020-05-24 18:30:00.000000+0000 |        IT |           Manager |     AMIT |          {'BOAT'} |  65000
    456 | 2020-09-04 18:30:00.000000+0000 |        IT |               SDE |   NAGRAJ | {'FLUTTER_APP'} |  95000
    600 | 2021-05-24 18:30:00.000000+0000 |     Sales | Assistant_Manager |      MAX |  {'ACCOUNTING'} |  45000

(3 rows)
```

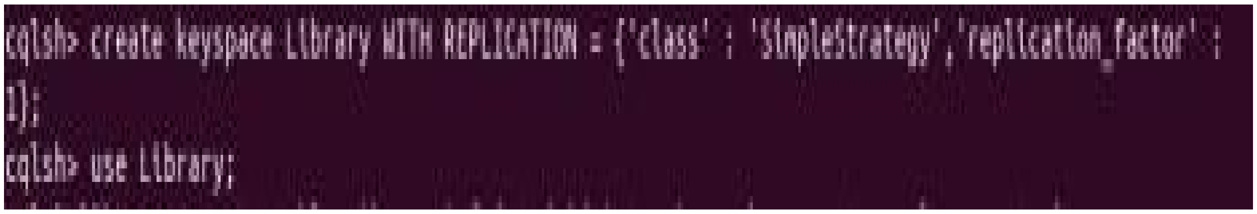
8 Create a TTL of 15 seconds to display the values of Employees.



```
cqlsh:emp> insert into EMPLOYEE_INFO(Emp_Id,Emp_Name,Designation,Date_of_joining,salary,Dept_name)values(001,'SASS','Assistent_SDE','2021-05-25',50000,'IT') USING TTL 15;
cqlsh:emp>    SELECT TTL(EMP_NAME) FROM EMPLOYEE_INFO WHERE EMP_ID=1;

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

(0 rows)
cqlsh:emp> insert into EMPLOYEE_INFO(Emp_Id,Emp_Name,Designation,Date_of_joining,salary,Dept_name)values(001,'SASS','Assistent_SDE','2021-05-25',50000,'IT') USING TTL 15;
cqlsh:emp> SELECT * FROM EMPLOYEE_INFO;

 emp_id | date_of_joining                 | dept_name | designation       | emp_name | project           | salary
--------+---------------------------------+-----------+-------------------+----------+-------------------+--------
    123 | 2020-05-24 18:30:00.000000+0000 |        IT |           Manager |     AMIT |          {'BOAT'} |  65000
    456 | 2020-09-04 18:30:00.000000+0000 |        IT |               SDE |   NAGRAJ | {'FLUTTER_APP'} |  95000
      1 | 2021-05-24 18:30:00.000000+0000 |        IT |     Assistent_SDE |     SASS |              null |  50000
    600 | 2021-05-24 18:30:00.000000+0000 |     Sales | Assistant_Manager |      MAX |  {'ACCOUNTING'} |  45000

(4 rows)
cqlsh:emp> insert into EMPLOYEE_INFO(Emp_Id,Emp_Name,Designation,Date_of_joining,salary,Dept_name)values(001,'SASS','Assistant_SDE','2021-05-25',50000,'IT') USING TTL 15;
cqlsh:emp> SELECT * FROM EMPLOYEE_INFO;

 emp_id | date_of_joining                 | dept_name | designation       | emp_name | project           | salary
--------+---------------------------------+-----------+-------------------+----------+-------------------+--------
    123 | 2020-05-24 18:30:00.000000+0000 |        IT |           Manager |     AMIT |          {'BOAT'} |  65000
    456 | 2020-09-04 18:30:00.000000+0000 |        IT |               SDE |   NAGRAJ | {'FLUTTER_APP'} |  95000
      1 | 2021-05-24 18:30:00.000000+0000 |        IT |     Assistent_SDE |     SASS |              null |  50000
    600 | 2021-05-24 18:30:00.000000+0000 |     Sales | Assistant_Manager |      MAX |  {'ACCOUNTING'} |  45000

(4 rows)
cqlsh:emp> SELECT * FROM EMPLOYEE_INFO;

 emp_id | date_of_joining                 | dept_name | designation       | emp_name | project           | salary
--------+---------------------------------+-----------+-------------------+----------+-------------------+--------
    123 | 2020-05-24 18:30:00.000000+0000 |        IT |           Manager |     AMIT |          {'BOAT'} |  65000
    456 | 2020-09-04 18:30:00.000000+0000 |        IT |               SDE |   NAGRAJ | {'FLUTTER_APP'} |  95000
    600 | 2021-05-24 18:30:00.000000+0000 |     Sales | Assistant_Manager |      MAX |  {'ACCOUNTING'} |  45000

(3 rows)
cqlsh:emp>
```

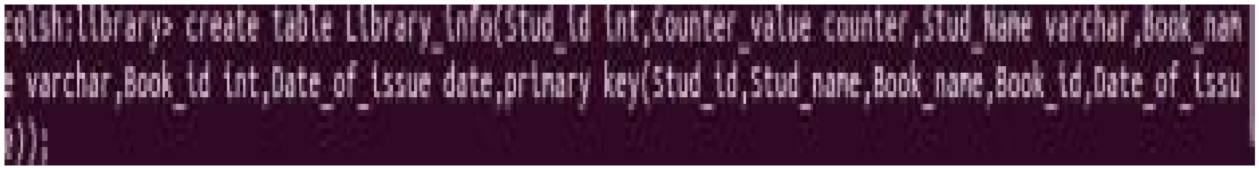
LAB 3:

1.Create a key space by name Library



```
cqlsh> create keyspace Library WITH REPLICATION = {'class' : 'SimpleStrategy','replication_factor' :
1};
cqlsh> use Library;
```

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

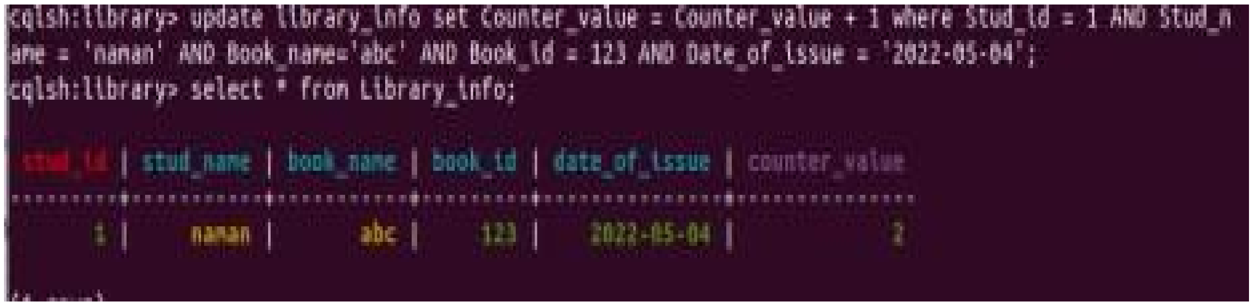


```
cqlsh:library> create table Library_info(Stud_id int,Counter_value counter,Stud_Name varchar,Book_nam
e varchar,Book_id int,Date_of_issue date,primary key(Stud_id,Stud_name,Book_name,Book_id,Date_of_issu
e));
```

3. Insert the values into the table in batch

```
[illegible]
```

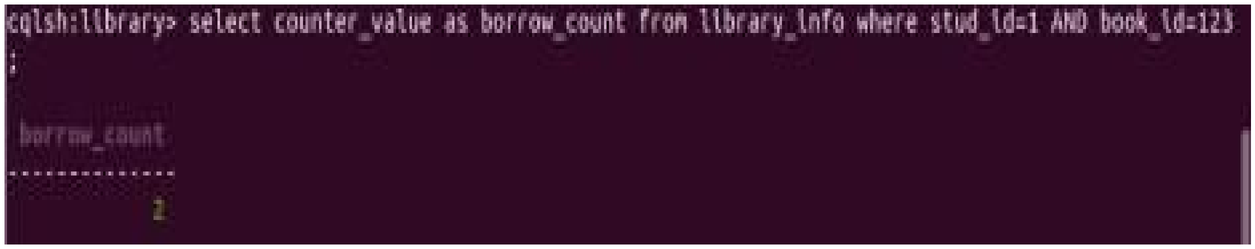
4. Display the details of the table created and increase the



```
cqlsh:library> update library_info set Counter_value = Counter_value + 1 where Stud_id = 1 AND Stud_n
ame = 'naman' AND Book_name='abc' AND Book_id = 123 AND Date_of_issue = '2022-05-04';
cqlsh:library> select * from Library_info;

 stud_id | stud_name | book_name | book_id | date_of_issue | counter_value
---------+-----------+-----------+---------+---------------+---------------
       1 |     naman |       abc |     123 |    2022-05-04 |             2
```

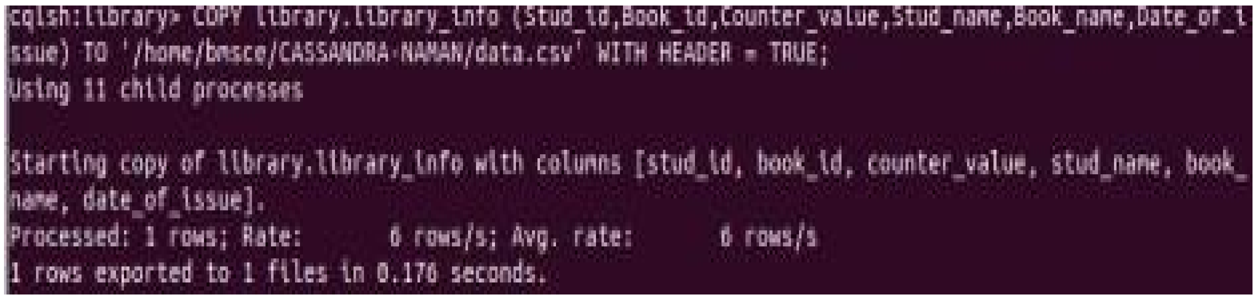
value of the counter

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



```
cqlsh:library> select counter_value as borrow_count from library_info where stud_id=1 AND book_id=123
;

 borrow_count
--------------
            2
```

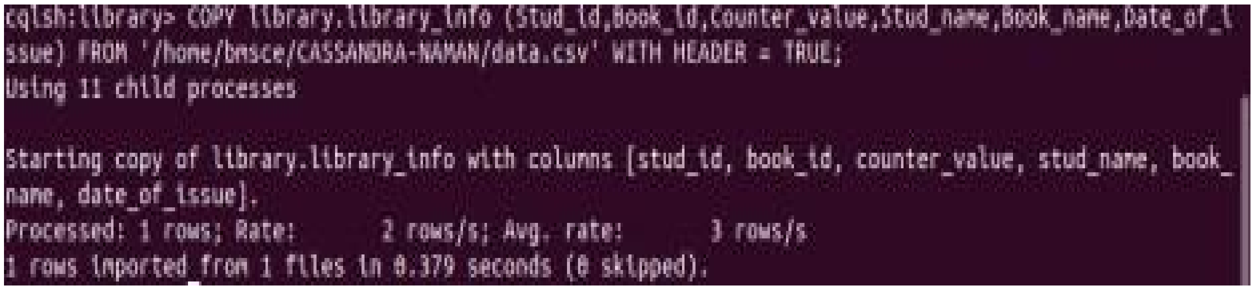
6. Export the created column to a csv file



```
cqlsh:library> COPY library.library_info (Stud_id,Book_id,Counter_value,Stud_name,Book_name,Date_of_i
ssue) TO '/home/bmsce/CASSANDRA-NAMAN/data.csv' WITH HEADER = TRUE;
Using 11 child processes

Starting copy of library.library_info with columns [stud_id, book_id, counter_value, stud_name, book_
name, date_of_issue].
Processed: 1 rows; Rate:       6 rows/s; Avg. rate:       6 rows/s
1 rows exported to 1 files in 0.176 seconds.
```

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

Cassandra column family



```
cqlsh:library> COPY library.library_info (Stud_id,Book_id,Counter_value,Stud_name,Book_name,Date_of_i
ssue) FROM '/home/bmsce/CASSANDRA-NAMAN/data.csv' WITH HEADER = TRUE;
Using 11 child processes

Starting copy of library.library_info with columns [stud_id, book_id, counter_value, stud_name, book_
name, date_of_issue].
Processed: 1 rows; Rate:       2 rows/s; Avg. rate:       3 rows/s
1 rows imported from 1 files in 0.379 seconds (0 skipped).
```