---VISVESVARAYA TECHNOLOGICAL UNIVERSITY

"JnanaSangama", Belgaum -590014, Karnataka.

LAB REPORT
on

BIG DATA ANALYTICS (20CS6PEBDA)

Submitted by

ASHWINI (1BM20CS402)

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

<u>CERTIFICATE</u>

This is to certify that the Lab work entitled "**BIG DATA ANALYTICS**" carried out by **ASHWINI (1BM20CS402),** 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.

Name of the Lab-Incharge	
PALLAVI	
Designation	Assistant Professor
Department of CSE	Department of CSE
BMSCE, Bengaluru	BMSCE, Bengaluru

`

Index Sheet

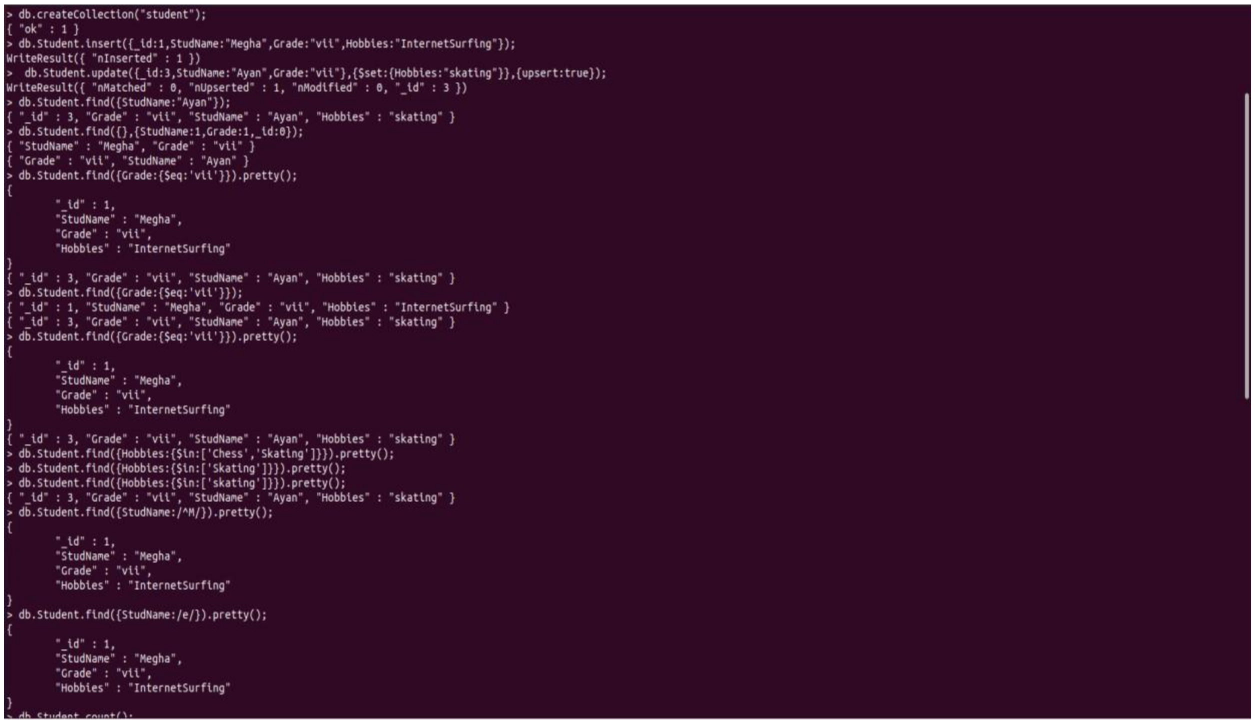
Course Outcome

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

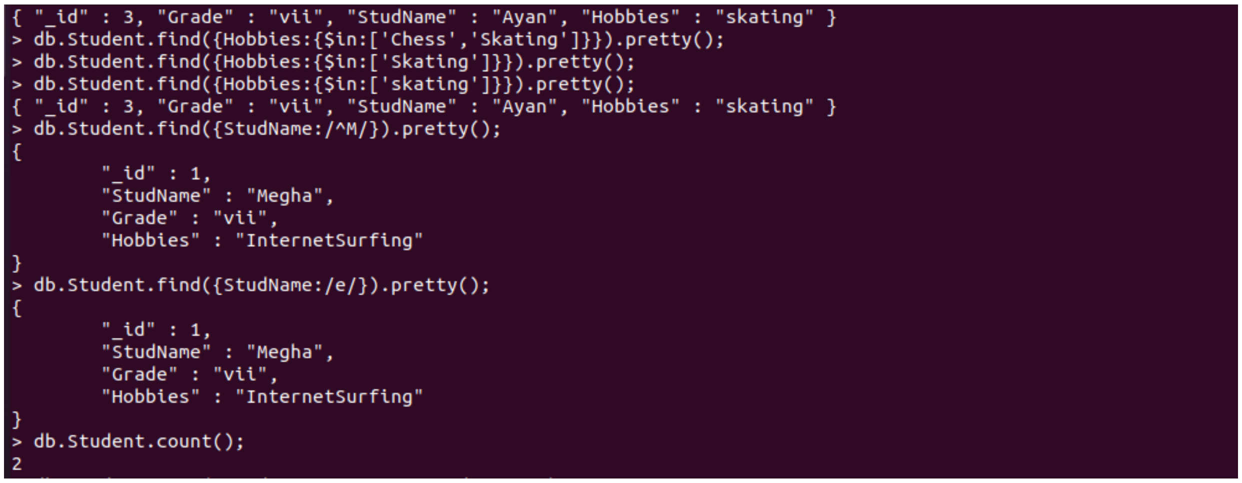
	Spark

1. MongoDB- CRUD Demonstration

CRUD (CREATE, READ, UPDATE, DELETE) OPERATIONS

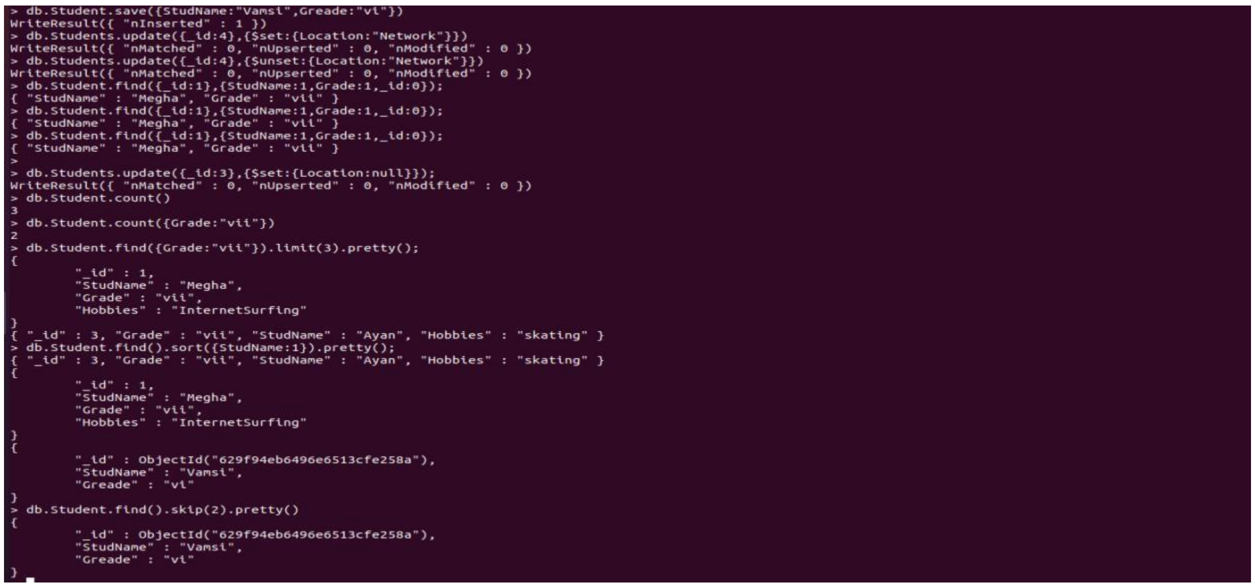


```
> db.createCollection("student");
{ "ok" : 1 }
> db.Student.insert({_id:1,StudName:"Megha",Grade:"vii",Hobbies:"InternetSurfing"});
WriteResult({ "nInserted" : 1 })
>  db.Student.update({_id:3,StudName:"Ayan",Grade:"vii"},{$set:{Hobbies:"skating"}},{upsert:true});
WriteResult({ "nMatched" : 0, "nUpserted"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

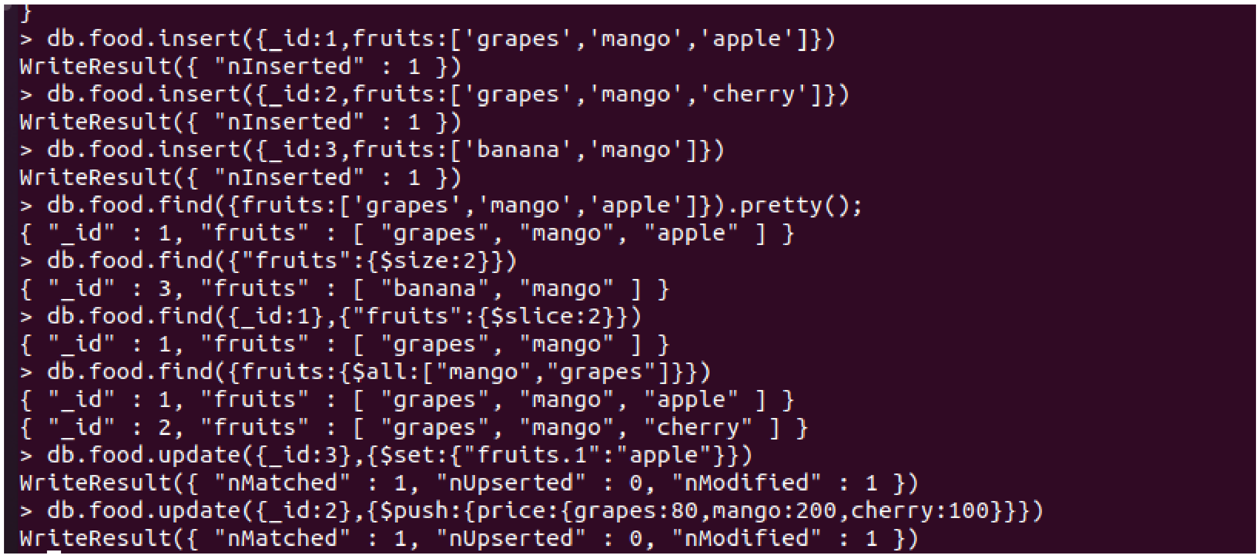


```
{ "_id" : 3, "Grade" : "vii", "StudName" : "Ayan", "Hobbies" : "skating" }
> db.Student.find({Hobbies:{$in:['Chess','Skating']}}).pretty();
> db.Student.find({Hobbies:{$in:['Skating']}}).pretty();
> db.Student.find({Hobbies:{$in:['skating']}}).pretty();
{ "_id" : 3, "Grade" : "vii", "StudName" : "Ayan", "Hobbies" : "skating" }
> db.Student.find({StudName:/^M/}).pretty();
{
        "_id" : 1,
        "StudName" : "Megha",
        "Grade" : "vii",
        "Hobbies" : "InternetSurfing"
}
> db.Student.find({StudName:/e/}).pretty();
{
        "_id" : 1,
        "StudName" : "Megha",
        "Grade" : "vii",
        "Hobbies" : "InternetSurfing"
}
> db.Student.count();
2
```

Save Method

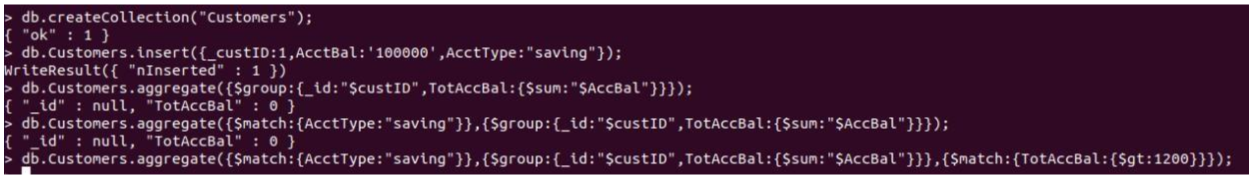


```
> db.Student.save({StudName:"Vamsi",Greade:"vi"})
WriteResult({ "nInserted" : 1 })
> db.Students.update({_id:4},{$set:{Location:"Network"}})
WriteResult({ "nMatched" : 0, "nUpserted" : 0, "nModified" : 0 })
> db.Students.update({_id:4},{$unset:{Location:"Network"}})
WriteResult({ "nMatched" : 0,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



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

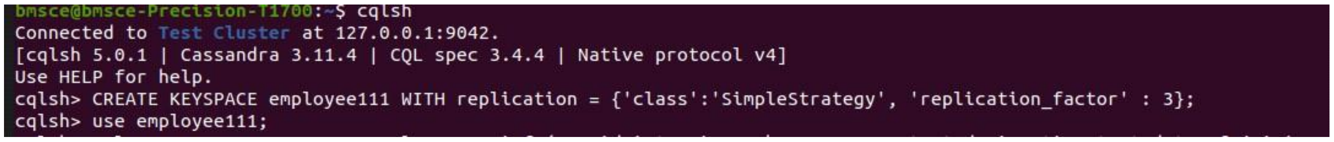
Aggregate Function



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

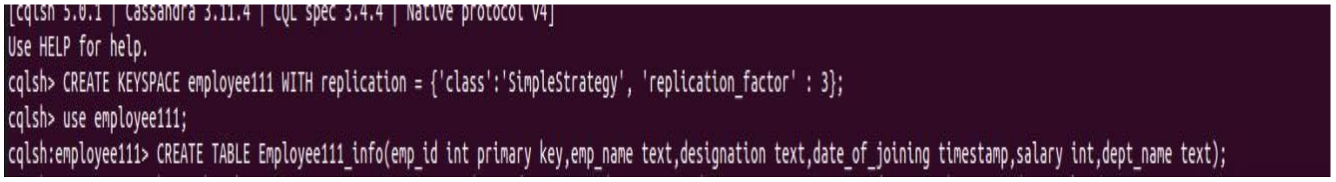
2.Perform the following DB operations using Cassandra.

1.Create a key space by name Employee



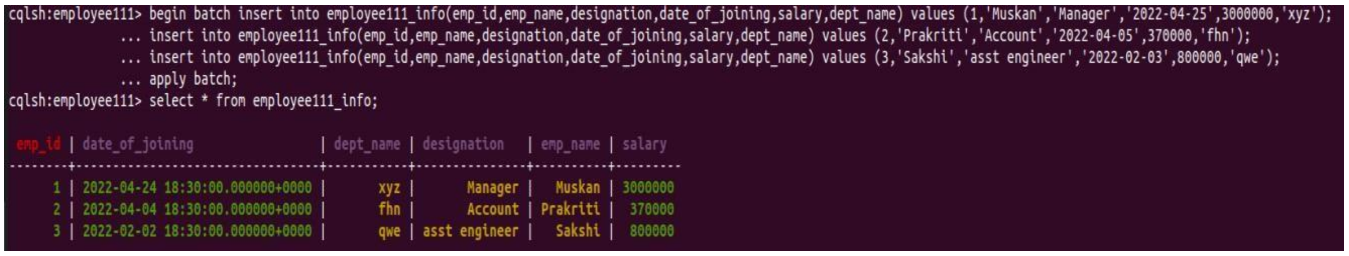
```
bmsce@bmsce-Precision-T1700:~$ cqlsh
Connected to Test Cluster at 127.0.0.1:9042.
[cqlsh 5.0.1 | Cassandra 3.11.4 | CQL spec 3.4.4 | Native protocol v4]
Use HELP for help.
cqlsh> CREATE KEYSPACE employee111 WITH replication = {'class':'SimpleStrategy', 'replication_factor' : 3};
cqlsh> use employee111;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

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



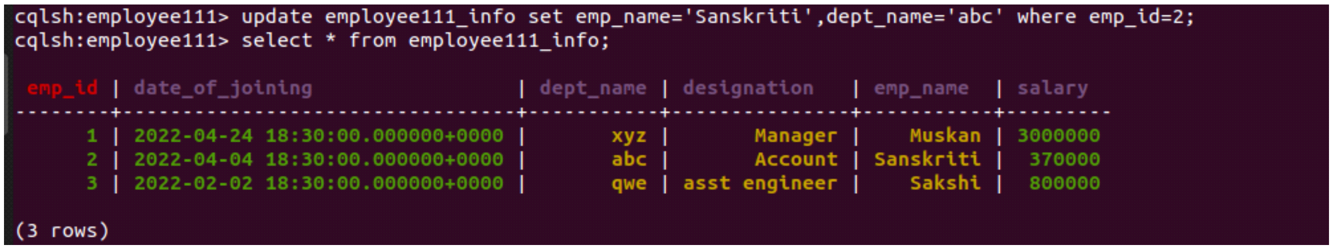
```
[cqlsh 5.0.1 | Cassandra 3.11.4 | CQL spec 3.4.4 | Native protocol v4]
Use HELP for help.
cqlsh> CREATE KEYSPACE employee111 WITH replication = {'class':'SimpleStrategy', 'replication_factor' : 3};
cqlsh> use employee111;
cqlsh:employee111> CREATE TABLE Employee111_info(emp_id int primary key,emp_name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

3.Insert the values into the table in batch



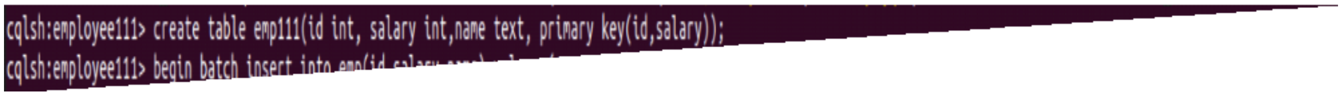
```
cqlsh:employee111> begin batch insert into employee111_info(emp_id,emp_name,designation,date_of_joining,salary,dept_name) values (1,'Muskan','Manager','2022-04-25',3000000,'xyz');
               ... insert into employee111_info(emp_id,emp_name,designation,date_of_joining,salary,dept_name) values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

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

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

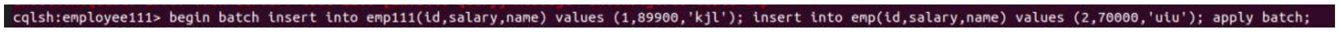


```
cqlsh:employee111> update employee111_info set emp_name='Sanskriti',dept_name='abc' where emp_id=2;
cqlsh:employee111> select * from employee111_info;

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

(3 rows)
```

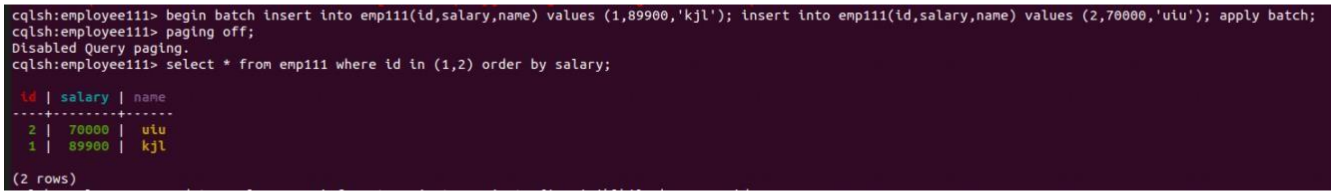
5. Sort the details of Employee records based on salary



```
cqlsh:employee111> create table emp111(id int, salary int,name text, primary key(id,salary));
cqlsh:employee111> begin batch insert into emp(id
```



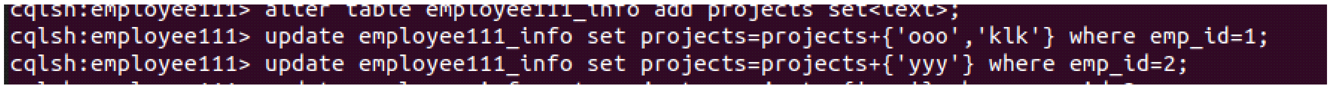
```
cqlsh:employee111> begin batch insert into emp111(id,salary,name) values (1,89900,'kjl'); insert into emp(id,salary,name) values (2,70000,'uiu'); apply batch;
```



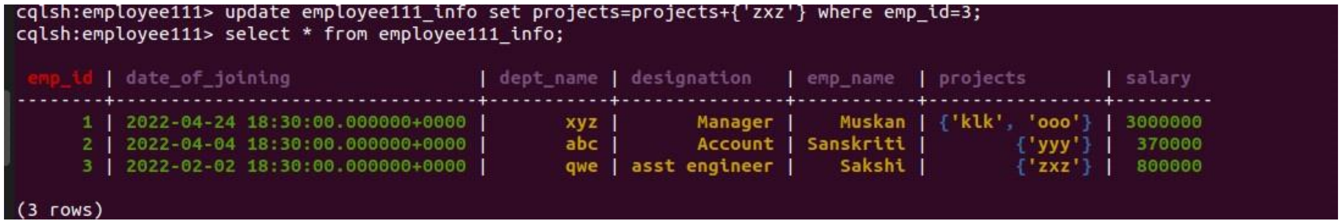
```
cqlsh:employee111> begin batch insert into emp111(id,salary,name) values (1,89900,'kjl'); insert into emp111(id,salary,name) values (2,70000,'uiu'); apply batch;
cqlsh:employee111> paging off;
Disabled Query paging.
cqlsh:employee111> select * from emp111 where id in (1,2) order by salary;

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

(2 rows)
```

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

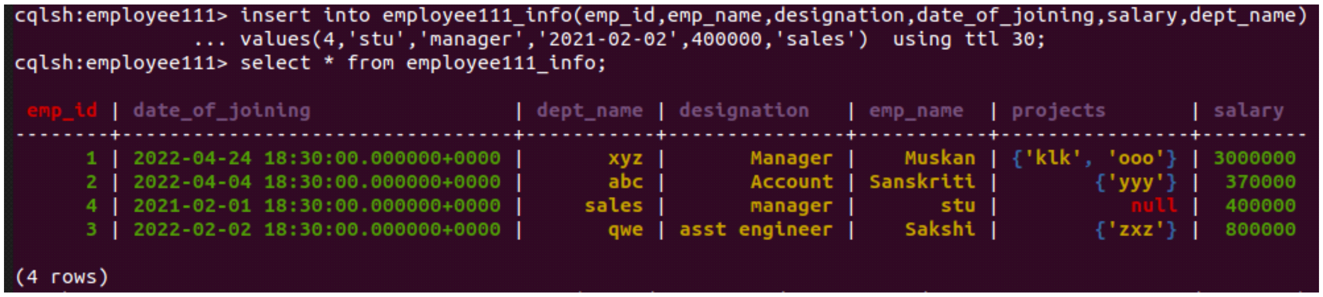
7. Update the altered table to add project names.



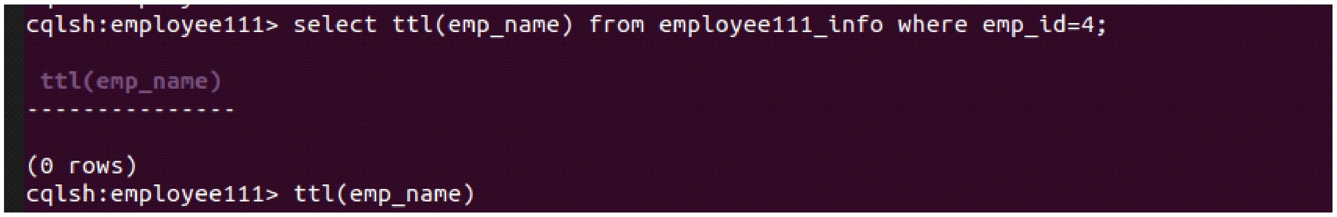
```
cqlsh:employee111> alter table employee111_info add projects set<text>;
cqlsh:employee111> update employee111_info set projects=projects+{'ooo','klk'} where emp_id=1;
cqlsh:employee111> update employee111_info set projects=projects+{'yyy'} where emp_id=2;
```



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

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

(3 rows)
```



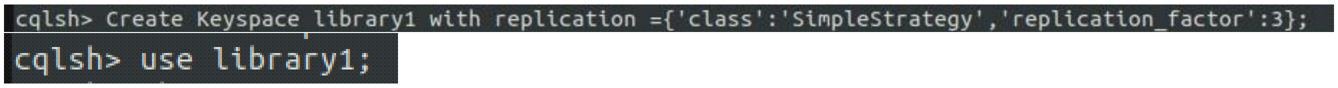
```
cqlsh:employee111> insert into employee111_info(emp_id,emp_name,designation,date_of_joining,salary,dept_name)
               ... values(4,'stu','manager','2021-02-02',400000,'sales')  using ttl 30;
cqlsh:employee111> select * from employee111_info;

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

(4 rows)
```



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

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

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

3. Perform the following DB operations using Cassandra.

1. Create a key space by name Library



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

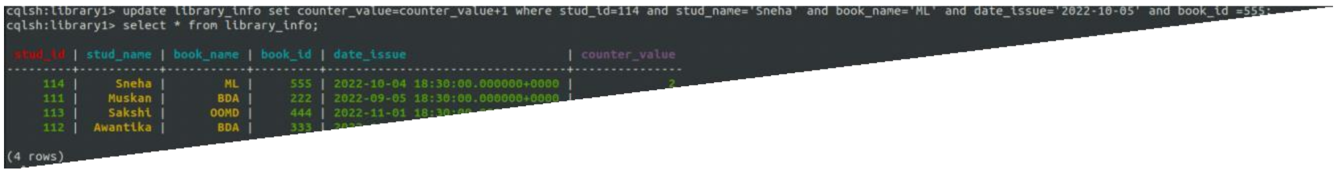
2. Create a column family by name Library-Info with attributes Stud_Id Primary Key,

Counter_value of type Counter,

Stud_Name, Book-Name, Book-Id, Date_of_issue

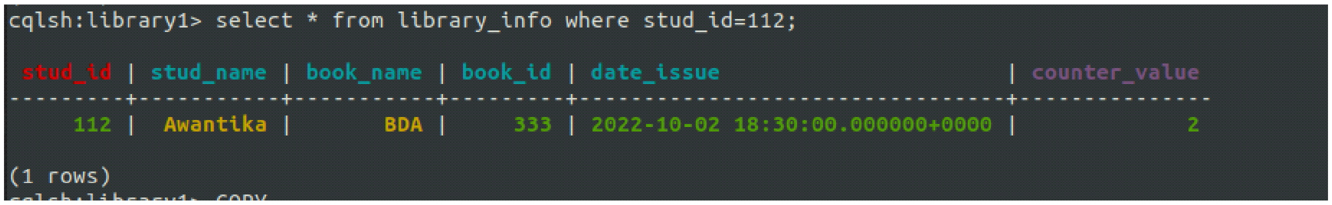
3. Insert the values into the table in batch

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



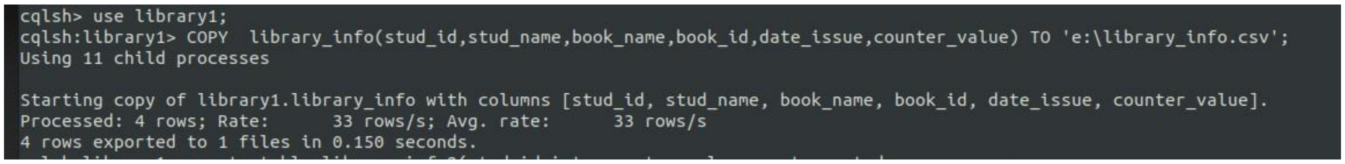
```
cqlsh:library1> update library_info set counter_value=counter_value+1 where stud_id=114 and stud_name='Sneha' and book_name='ML' and date_issue='2022-10-05' and book_id =555;
cqlsh:library1> select * from library_info;

 stud_id | stud_name | book_name | book_id | date_issue                      | counter_value
---------+-----------+-----------+---------+---------------------------------+---------------
     114 |     Sneha |        ML |     555 | 2022-10-04 18:30:00.000000+0000 |             2
     111 |    Muskan |       BDA |     222 | 2022-09-05 18:30:00.000000+0000 |
     113 |    Sakshi |      OOMD |     444 | 2022-11-01 18:30:00
     112 |  Awantika |       BDA |     333 |

(4 rows)
```

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



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

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

(1 rows)
cqlsh:library1> COPY
```

6. Export the created column to a csv file



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

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

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