VISVESVARAYA TECHNOLOGICAL UNIVERSITY

"JnanaSangama", Belgaum -590014, Karnataka.

LAB REPORT
on

BIG DATA ANALYTICS (20CS6PEBDA)

Submitted by

MITHIL RAJ(1BM18CS086)

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 **MITHIL RAJ(1BM18CS086),** 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.

Nameof the Lab-Incharge	**ANATAR ROY CHOUDHURY**
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 MONGODB (CRUD OPERATION):-

I. CREATE DATABASE IN MONGODB

use myDB;

Confirm the existence of your database

Db;

To list all databases

Show dbs;

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

1. To create a collection by the name “Student”. Let us take a look at the collection list prior to the creation of the new collection “Student”.

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

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

   db.Student.drop();

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

db.Student.insert({_id:1,StudName:"MichelleJacintha",Grade:"VII",Hobbies:"InternetSurfing"});

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});

5. FIND METHOD

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

III. Import data from a CSV file

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

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

IV. Export data to a CSV file

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

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

V. Save Method :

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

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

VI. Add a new field to existing Document:

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

VII. Remove the field in an existing Document

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

VIII. Finding Document based on search criteria suppressing few fields

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

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

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

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

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

IX. to set a particular field value to NULL

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

X. Count the number of documents in Student Collections

db.Students.count()

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 1st two documents from the Students Collections db.Students.find().skip(2).pretty()

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'] })

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().

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}})

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

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

db.food.find ({“fruits”: {$size:2}})

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}})

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”]}})

update on Array:

array with apple

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

insert new key value pairs in the fruits array

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

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

XIII. 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}}});

Assignment:

Creation of Cursor:

Create Collection “Alphabets”

Insert Documents with fields “_id” and “alphabet”

use cursor to iterate through the “Alphabets” Collection.

NAME:MITHIL RAJ

USN:1BM19CS086

BDA LAB-1

LAB 2 MONGO DB (CRUD OPERATIONS):-

MONGO DB

1) Using MongoDB

i) Create a database for Students and Create a Student Collection

(_id,Name, USN,Semester, Dept_Name, CGPA, Hobbies(Set)).

```
> use Students
switched to db Students
```

ii) Insert required documents to the collection.

```
➢ db.Student.insert({Studname:"MITHIL RAJ",USN:"1BM19CS086",Semester:
  "VII",Dept_name:"Computer
  Science",CGPA:9.6,Hobbies:["Sleep","eat"]});
  WriteResult({ "nInserted" : 1 })

> db.Student.insert({Studname:"NITHIN",USN:"1BM19CS106",Semester:
  "VI",Dept_name:"Computer
  Science",CGPA:8.6,Hobbies:["Sleep","eat"]});
  WriteResult({ "nInserted" : 1 })

> db.Student.insert({Studname:"Hailey",USN:"1BM19CS015",Semester
  :"VIII",Dept_name:"Computer
  Science",CGPA:7.4,Hobbies:["Sleep","eat","repeat"]});
  WriteResult({ "nInserted" : 1 })
```

iii) First Filter on “Dept_Name:CSE” and then group it on “Semester” and compute the Average CPGA for that semester and filter those documents where the “Avg_CPGA” is greater than 7.5.

```
> db.Student.aggregate({$match:{Dept_name:"Computer
Science"}},{$group:{_id:"$Semester",AvgCGPA:{$avg:"$CGPA"}}},{$m
atch:{AvgCGPA:{$gt:7.5}}});

{ "_id" : "VIII", "AvgCGPA" : 8.6 }
{ "_id" : "VII", "AvgCGPA" : 8.533333333333333 }
{ "_id" : "VI", "AvgCGPA" : 8.266666666666667 }
```

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

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

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

1. Insert three documents

```
db.Bank.insert({_id:1,name:"Ramesh",state:"Gujarat",country:"India",language:["gujarati","marat
hi","english"]})

db.Bank.insert({_id:2,name:"Mahesh",state:"Gujarat",country:"India",language:["gujarati","marw
adi","english"]})

db.Bank.insert({_id:3,name:"Ghelbhai",state:"Maharashta",country:"India",language:["marathi","
marwadi","english"]})
```

2. Use Arrays(Use Pull and Pop operation)

```
db.Bank.update({_id: 1}, {$push: {language: "hindi"}})

db.Bank.update({_id: 2}, {$pull: {language: "english"}})
```

3. Use Index
4. Use Cursors
5. Updation

3) Consider a table "Students " with the following columns:
1. StudRollNo / _id
2. StudName
3. Grade
4. Hobbies
5. DOJ

Write MongoDB queries for the following:

1. To display only the students name from all the documents of the Students collection.

```
> db.Students.find({},{Studname:1,_id:0});
{ "Studname" : "mithil" }

{ "Studname" : "varun" }
{ "Studname" : "Lodi" }
{ "Studname" : "Modi" }
{ "Studname" : "Nithin" }
```

2. To display only the student name, grade as well as the identifier from the document of the Student collection where the _id column is 1.

```
> db.Students.find({_id:{$eq:ObjectId("625fd1171e24dbace73bd604")}
},{Studname:1,Grade:1,_id:1});
{ "_id" : ObjectId("625fd1171e24dbace73bd604"), "Studname" : "mithil",
"Grade" : "VII" }
```

3. To find those documents where the grade is not set to VIII.

```
> db.Students.find({Grade:{$ne:"VII"}});
{ "_id" : ObjectId("625fd11d1e24dbace73bd605"), "Studname" :
"varun", "Grade" : "VIII", "Hobbies" : [ "cricket" ], "DOJ" : "12/8/2021" }
{ "_id" : ObjectId("625fd1241e24dbace73bd606"), "Studname" :
"Lodi", "Grade" : "VIII", "Hobbies" : [ "Sleep" ], "DOJ" : "12/8/2021" }
{ "_id" : ObjectId("625fd12d1e24dbace73bd607"), "Studname" :
"Modi", "Grade" : "VI", "Hobbies" : [ "Sleep", "eat" ], "DOJ" : "12/7/2001"
}
```

4. To find those documents from the Students collection where the hobbies is set to 'cricket' and the student name is set to 'varun'.

```
> db.Student.find({Hobbies :{
$in:['cricket']},Studname:{$eq:"varun"}}).pretty ();
{
"_id" : ObjectId("625fd0771e24dbace73bd602"),
"Studname" : "varun",
"Grade" : "VIII",
"Hobbies" : [
"cricket"
],
"DOJ" : "12/8/2021"
}
```

5.To find documents from the Students collection where the student name ends in 'j'

```
> db.Student.find({Studname:/j$/}).pretty();
{
"_id" : ObjectId("625fd09b1e24dbace73bd603"),
"Studname" : "mithil",
"Grade" : "VII",
"Hobbies" : [
"cricket"
],
"DOJ" : "12/8/2021"
}
```

4) Using MongoDB,

i) Create a database for Faculty and Create a Faculty Collection(Faculty_id, Name, Designation ,Department, Age, Salary, Specialization(Set)).

```
> use faculty

switched to db faculty

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

iii) Insert required documents to the collection.

```
> db.Faculty.insert({Name:"NITHIN",Designation:"Teacher",Department:"
CSE",Age:90,Salary:40000,Specialization:["Eating","Talking","Web
dev"]});
WriteResult({ "nInserted" : 1 })

> db.Faculty.insert({Name:"KHUSHIL",Designation:"Teacher",Depart
ment:"MECH",Age:90,Salary:120000,Specialization:["Eating","Talking"
,"Web dev"]});
WriteResult({ "nInserted" : 1 })

> db.Faculty.insert({Name:"ugrasen",Designation:"Assisstant",Departm
ent:"MECH",Age:20,Salary:1000,Specialization:["Eating","Talking","We
b dev"]});
WriteResult({ "nInserted" : 1 })

>
db.Faculty.insert({Name:"JEEVAN",Designation:"Assisstant",Departmen
t:"MECH",Age:20,Salary:111000,Specialization:["Eating","Talking","We
b dev"]});
WriteResult({ "nInserted" : 1 })
```

iii) First Filter on “Dept_Name:MECH” and then group it on “Designation” and
compute the Average Salary for that Designation and filter those documents where the “Avg_Sal” is greater than 6500.

```
> db.Faculty.aggregate({$match:{Department:"MECH"}},{$group:{_id:"$
Designation",AvgSAL:{$avg:"$Salary"}}},{$match:{AvgSAL:{$gt:6500}
}});
{ "_id" : "Assisstant", "AvgSAL" : 56000 }
{ "_id" : "Teacher", "AvgSAL" : 120000 }
```

NAME:MITHIL RAJ

USN:1BM19CS086

BDA LAB-2

LAB-3 CASSANDRA EMPLOYEE QUESTION:-

1. Program 1. Perform the following DB operations using Cassandra

2. Create a key space by name Employee

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

3. Insert the values into the table in batch

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

5. Sort the details of Employee records based on salary

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

8. Update the altered table to add project names.

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

SOLUTION:-

```
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 Employee_info WITH REPLICATION =
{'class':'SimpleStrategy','replication_factor':2};
cqlsh> DESCRIBE KEYSPACES;

employee_info  system_auth  employee134  tranzmetro        employee
students       system       students1    studentinfo       system_traces
system_schema  library      tranz        system_distributed  students2
```

```
cqlsh> USE Employee_info;

cqlsh:employee_info> CREATE TABLE EMPLOYEE_INFO(Emp_id int PRIMARY KEY,
Emp_name text,Designation text,DOJ timestamp,salary int,Dept_name text);

cqlsh:employee_info> INSERT INTO EMPLOYEE-

INFO(Emp_id,Emp_name,Designation,DOJ,salary,Dept_name) VALUES (1,'manny','senior
employee','2018-06-01','1000000','CSE');

SyntaxException: line 1:20 no viable alternative at input '-' (INSERT INTO [EMPLOYEE]-...)
cqlsh:employee_info> INSERT INTO
EMPLOYEE_INFO(Emp_id,Emp_name,Designation,DOJ,salary,Dept_name) VALUES
(1,'manny','senior employee','2018-06-01','1000000','CSE');

InvalidRequest: Error from server: code=2200 [Invalid query] message="Invalid STRING
constant (1000000) for "salary" of type int"

cqlsh:employee_info> INSERT INTO
EMPLOYEE_INFO(Emp_id,Emp_name,Designation,DOJ,salary,Dept_name) VALUES
(1,'manny','senior employee','2018-06-01',1000000,'CSE');

cqlsh:employee_info> INSERT INTO
EMPLOYEE_INFO(Emp_id,Emp_name,Designation,DOJ,salary,Dept_name) VALUES
(2,'maddy','Manager','2017-04-01',100000,'ISE');

cqlsh:employee_info> INSERT INTO
EMPLOYEE_INFO(Emp_id,Emp_name,Designation,DOJ,salary,Dept_name) VALUES
(3,'nathen','junior employee','2019-01-01',200000,'EEE');
cqlsh:employee_info> SELECT * FROM EMPLOYEE_INFO;


 emp_id | dept_name | designation     | doj                             | emp_name | salary
--------+-----------+-----------------+---------------------------------+----------+---------
      1 |       CSE | senior employee | 2018-05-31 18:30:00.000000+0000 |    manny | 1000000
      2 |       ISE |         Manager | 2017-03-31 18:30:00.000000+0000 |    maddy |  100000
      3 |       EEE | junior employee | 2018-12-31 18:30:00.000000+0000 |   nathen |  200000

(3 rows)
cqlsh:employee_info> UPDATE EMPLOYEE_INFO SET Emp_name='mithil',Dept_name='EEE'
WHERE Emp_id=2;

cqlsh:employee_info> SELECT * FROM EMPLOYEE_INFO;


 emp_id | dept_name | designation     | doj                             | emp_name | salary
--------+-----------+-----------------+---------------------------------+----------+---------
      1 |       CSE | senior employee | 2018-05-31 18:30:00.000000+0000 |    manny | 1000000
      2 |       EEE |         Manager | 2017-03-31 18:30:00.000000+0000 |   mithil |  100000
      3 |       EEE | junior employee | 2018-12-31 18:30:00.000000+0000 |   nathen |  200000
```

```
(3 rows)
cqlsh:employee_info> ALTER TABLE EMPLOYEE_INFO ADD PROJECTS SET<text>;

cqlsh:employee_info> SELECT * FROM EMPLOYEE_INFO;


 emp_id | dept_name | designation     | doj                             | emp_name | projects | salary
--------+-----------+-----------------+---------------------------------+----------+----------+---------
      1 |       CSE | senior employee | 2018-05-31 18:30:00.000000+0000 |    manny |     null |
1000000
      2 |       EEE |         Manager | 2017-03-31 18:30:00.000000+0000 |   mithil |     null |  100000
      3 |       EEE | junior employee | 2018-12-31 18:30:00.000000+0000 |   nathen |     null |
 200000

(3 rows)
cqlsh:employee_info> UPDATE EMPLOYEE_INFO SET
PROJECTS=PROJECTS+{'WEBAPP','ANDROIDAPP'} WHERE Emp_id=1;

cqlsh:employee_info> UPDATE EMPLOYEE_INFO SET
PROJECTS=PROJECTS+{'WEBAPP1','ANDROIDAPP1'} WHERE Emp_id=2;

cqlsh:employee_info> UPDATE EMPLOYEE_INFO SET
PROJECTS=PROJECTS+{'WEBAPP2','ANDROIDAPP2'} WHERE Emp_id=3;

cqlsh:employee_info> UPDATE EMPLOYEE_INFO SET
PROJECTS=PROJECTS+{'WEBAPP1','ANDROIDAPP1'} WHERE Emp_id=2;

cqlsh:employee_info> SELECT * FROM EMPLOYEE-INFO;

SyntaxException: line 1:22 no viable alternative at input '-' (SELECT * FROM [EMPLOYEE]-...)
cqlsh:employee_info> SELECT * FROM EMPLOYEE_INFO;


 emp_id | dept_name | designation     | doj                             | emp_name | projects                   |
salary
--------+-----------+-----------------+---------------------------------+----------+----------------------------+---------
      1 |       CSE | senior employee | 2018-05-31 18:30:00.000000+0000 |    manny |
{'ANDROIDAPP', 'WEBAPP'} | 1000000
      2 |       EEE |         Manager | 2017-03-31 18:30:00.000000+0000 |   mithil |
{'ANDROIDAPP1', 'WEBAPP1'} |  100000
      3 |       EEE | junior employee | 2018-12-31 18:30:00.000000+0000 |   nathen |
{'ANDROIDAPP2', 'WEBAPP2'} |  200000

(3 rows)
cqlsh:employee_info> INSERT INTO
EMPLOYEE_INFO(Emp_id,Emp_name,Designation,DOJ,salary,Dept_name) VALUES
(4,'nidhi','junior1 employee','2020-02-04',300000,'ECE') USING TTL 15;

cqlsh:employee_info> SELECT * FROM EMPLOYEE_INFO;
```

```
 emp_id | dept_name | designation     | doj                             | emp_name | projects                   |
salary
--------+-----------+-----------------+---------------------------------+----------+----------------------------+---------
      1 |       CSE | senior employee | 2018-05-31 18:30:00.000000+0000 |    manny |
{'ANDROIDAPP', 'WEBAPP'} | 1000000
      2 |       EEE |         Manager | 2017-03-31 18:30:00.000000+0000 |   mithil |
{'ANDROIDAPP1', 'WEBAPP1'} |  100000
      3 |       EEE | junior employee | 2018-12-31 18:30:00.000000+0000 |   nathen |
{'ANDROIDAPP2', 'WEBAPP2'} |  200000

(3 rows)
cqlsh:employee_info> INSERT INTO
EMPLOYEE_INFO(Emp_id,Emp_name,Designation,DOJ,salary,Dept_name) VALUES
(4,'nidhi','junior1 employee','2020-02-04',300000,'ECE') USING TTL 15;

cqlsh:employee_info> SELECT * FROM EMPLOYEE_INFO;


 emp_id | dept_name | designation      | doj                             | emp_name | projects                   |
salary
--------+-----------+------------------+---------------------------------+----------+----------------------------+---------
      1 |       CSE |  senior employee | 2018-05-31 18:30:00.000000+0000 |    manny |
{'ANDROIDAPP', 'WEBAPP'} | 1000000
      2 |       EEE |          Manager | 2017-03-31 18:30:00.000000+0000 |   mithil |
{'ANDROIDAPP1', 'WEBAPP1'} |  100000
      4 |       ECE | junior1 employee | 2020-02-03 18:30:00.000000+0000 |    nidhi |
null |  300000
      3 |       EEE |  junior employee | 2018-12-31 18:30:00.000000+0000 |   nathen |
{'ANDROIDAPP2', 'WEBAPP2'} |  200000

(4 rows)
cqlsh:employee_info> CREATE TABLE EMP(id int, salary int,name text,PRIMARY
KEY(id,salary));

cqlsh:employee_info> INSERT INTO EMP(id,salary,name) VALUES (1,100000,'myth');

cqlsh:employee_info> INSERT INTO EMP(id,salary,name) VALUES (1,100000,'myth');

cqlsh:employee_info> INSERT INTO EMP(id,salary,name) values (1,100000,'myth');

cqlsh:employee_info> INSERT INTO EMP(id,salary,name) values (2,200000,'mith');

cqlsh:employee_info> INSERT INTO EMP(id,salary,name) values (3,500000,'nith');

cqlsh:employee_info> SELECT * FROM EMP WHERE ID IN (1,2,3,4) ORDER BY SALARY;
InvalidRequest: Error from server: code=2200 [Invalid query] message="Cannot page queries
with both ORDER BY and a IN restriction on the partition key; you must either remove the
ORDER BY or the IN and sort client side, or disable paging for this query"

cqlsh:employee_info> PAGING OFF;
```

```
Disabled Query paging.

cqlsh:employee_info> SELECT * FROM EMP WHERE ID IN (1,2,3,4) ORDER BY SALARY;

 id | salary | name
----+--------+------
  1 | 100000 | myth
  2 | 200000 | mith
  3 | 500000 | nith

(3 rows)
```

NAME:MITHIL RAJ

USN:1BM19CS086

BDA LAB 3 CASSANDRA

LAB 4 CASSANDRA LIBRARY:-

CASSANDRA

Perform the following DB operations using Cassandra.

Program 2:

1 Create a key space by name Library

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

3.Insert the values into the table in batch

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

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

6.Export the created column to a csv file

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

SOLUTION:-

```
bmscecse@bmscecse-HP-Pro-3330-MT:~$ cqlsh
Connected to Test Cluster at 127.0.0.1:9042
[cqlsh 6.0.0 | Cassandra 4.0.3 | CQL spec 3.4.5 | Native protocol v5]
Use HELP for help.
cqlsh> create keyspace library_info with replication =
{'class':'SimpleStrategy','replication_factor':2};
AlreadyExists: Keyspace 'library_info' already exists
cqlsh> describe keyspaces;

library_info  system_auth         system_traces
student       system_distributed  system_views
system        system_schema       system_virtual_schema
```

```
cqlsh:library_info> create table library_details(stud_id int,counter_value counter,stud_name
text,book_id int,book_name text,date_of_issue timestamp,primary
key(stud_id,stud_name,book_name,date_of_issue,book_id));

AlreadyExists: Table 'library_info.library_details' already exists

cqlsh:library_info> create table library_information(stud_id int,counter_value counter,stud_name
text,book_id int,book_name text,date_of_issue timestamp,primary
key(stud_id,stud_name,book_name,date_of_issue,book_id));

cqlsh:library_info> update library_information set counter_value = counter_value+1 where
stud_id = 111 and stud_name ='mithil' and book_name ='BDA' and date_of_issue = '2020-11-
08' and book_id = 200;

cqlsh:library_info> update library_information set counter_value = counter_value+1 where
stud_id = 112 and stud_name ='myth' and book_name ='ML' and date_of_issue = '2020-05-01'
and book_id = 300;

cqlsh:library_info> update library_information set counter_value = counter_value+1 where
stud_id = 113 and stud_name ='mith' and book_name ='OOMD' and date_of_issue = '2020-01-
01' and book_id = 400;

cqlsh:library_info> select * from library-information;

SyntaxException: line 1:25 mismatched character 'o' expecting set null

cqlsh:library_info> select * from library_information;

 stud_id | stud_name | book_name | date_of_issue                   | book_id | counter_value
---------+-----------+-----------+---------------------------------+---------+---------------
     111 |    mithil |       BDA | 2020-11-07 18:30:00.000000+0000 |     200 |             1
     113 |      mith |      OOMD | 2019-12-31 18:30:00.000000+0000 |     400 |             1
     112 |      myth |        ML | 2020-04-30 18:30:00.000000+0000 |     300 |             1


(3 rows)

cqlsh:library_info> update library_information set counter_value = counter_value+1 where
stud_id = 111 and stud_name ='mithil' and book_name ='BDA' and date_of_issue = '2020-11-
08' and book_id = 200;

cqlsh:library_info>  select * from library_information where stud_id = 111;


 stud_id | stud_name | book_name | date_of_issue                   | book_id | counter_value
---------+-----------+-----------+---------------------------------+---------+---------------
     111 |    mithil |       BDA | 2020-11-07 18:30:00.000000+0000 |     200 |             2


cqlsh:library_info> copy
```

```
library_information(stud_id,stud_name,book_id,book_name,date_of_issue,counter_value) to
'/home/bmscecse/library_information.csv';

Using 3 child processes

Starting copy of library_info.library_information with columns [stud_id, stud_name, book_id,
book_name, date_of_issue, counter_value].
Processed: 3 rows; Rate:      32 rows/s; Avg. rate:      32 rows/s
3 rows exported to 1 files in 0.097 seconds.
cqlsh:library_info> truncate library_information;
cqlsh:library_info> copy
library_information(stud_id,stud_name,book_id,book_name,date_of_issue,counter_value) from
'/home/bmscecse/library_information.csv';
Using 3 child processes

Starting copy of library_info.library_information with columns [stud_id, stud_name, book_id,
book_name, date_of_issue, counter_value].
Processed: 3 rows; Rate:       5 rows/s; Avg. rate:       7 rows/s
3 rows imported from 1 files in 0.418 seconds (0 skipped).
cqlsh:library_info> select * from library_information;


 stud_id | stud_name | book_name | date_of_issue                   | book_id | counter_value
---------+-----------+-----------+---------------------------------+---------+---------------
     111 |    mithil |       BDA | 2020-11-07 18:30:00.000000+0000 |     200 |             2
     113 |      mith |      OOMD | 2019-12-31 18:30:00.000000+0000 |     400 |             1
     112 |      myth |        ML | 2020-04-30 18:30:00.000000+0000 |     300 |             1


(3 rows)
```

NAME:MITHIL RAJ
USN:-1BM19CS086