**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**

|  |  |  |
| --- | --- | --- |
| **Sl. No.** | **Experiment Title** | **Page No.** |
| **1.** | **MONGODB LAB 1(CRUD OPERATIONS)** |  |
| **2.** | **MONGODB LAB 2(CRUD OPERATIONS)** |  |
| **3.** | **CASSANDRA LAB 3 EMPLOYEE** |  |
| **4.** | **CASSANDRA LAB 4 LIBRARY** |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |

**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):-**

1. CREATE DATABASE IN MONGODB

use myDB;

Confirm the existence of your database

Db;

To list all databases

Show dbs;

1. 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:"Intern

etSurfing"});

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:"Sk

ating"}},{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();

1. 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

1. 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”

1. 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”})

1. Add a new field to existing Document:

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

1. Remove the field in an existing Document

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

1. 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();

1. to set a particular field value to NULL

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

1. Count the number of documents in Student Collections

db.Students.count()

1. 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()

1. 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

1. 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
2. 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

1. 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","marwadi","english"]})

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

1. 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" }

1. 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 }

1. 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

1. Insert the values into the table in batch
2. Update Employee name and Department of Emp-Id 121
3. 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](http://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](http://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