VISVESVARAYA TECHNOLOGICAL UNIVERSITY

"JnanaSangama", Belgaum -590014, Karnataka.

LAB REPORT
on

BIG DATA ANALYTICS
(20CS6PEBDA)

Submitted by

M Vamshi Krishna (1BM19CS080)

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 **M Vamshi Krishna(1BM19CS080),** who is a 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.

Dr.Pallavi G B
Assistant Professor
Department of CSE
BMSCE, Bengaluru

Dr. Jyothi S Nayak
Professor and Head
Department of CSE
BMSCE, Bengaluru

`

Index Sheet

Course Outcome

CO 1	Apply the concept of NoSQL, Hadoop or Spark for a given task
CO 2	Analyze the Big Data and obtain insight using data analytics mechanisms.
CO 3	Design and implement Big data applications by applying NoSQL, Hadoop or Spark

BDA LAB 1

```
bmsce@bmsce-OptiPlex-3060:~$ mongo
MongoDB shell version v4.0.28
connecting to: mongodb://127.0.0.1:27017/?gssapiServiceName=mongodb
Implicit session: session { "id" : UUID("c2e3109b-0341-483b-ba3a-f9fb3b1aed87") }
MongoDB server version: 4.0.28
Server has startup warnings:
2022-04-11T14:03:08.254+0530 I STORAGE  [initandlisten]
2022-04-11T14:03:08.254+0530 I STORAGE  [initandlisten] ** WARNING: Using the XFS filesystem is strongly recommended with the WiredTiger storage engine
2022-04-11T14:03:08.254+0530 I STORAGE  [initandlisten] **          See http://dochub.mongodb.org/core/prodnotes-filesystem
2022-04-11T14:03:10.024+0530 I CONTROL  [initandlisten]
2022-04-11T14:03:10.024+0530 I CONTROL  [initandlisten] ** WARNING: Access control is not enabled for the database.
2022-04-11T14:03:10.024+0530 I CONTROL  [initandlisten] **          Read and write access to data and configuration is unrestricted.
2022-04-11T14:03:10.024+0530 I CONTROL  [initandlisten]
---
Enable MongoDB's free cloud-based monitoring service, which will then receive and display
metrics about your deployment (disk utilization, CPU, operation statistics, etc).

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

To enable free monitoring, run the following command: db.enableFreeMonitoring()
To permanently disable this reminder, run the following command: db.disableFreeMonitoring()
---

> db;
test
> use lab1DB;
switched to db lab1DB
> db;
lab1DB
```

```
> show dbs;
admin   0.000GB
config  0.000GB
local   0.000GB
myDB    0.000GB
> db.createCollection("Student");
{ "ok" : 1 }
> db.Student.drop();
true
> db.getCollectionNames()
[ ]
> db.createCollection("Student");
{ "ok" : 1 }
> db.Student.insert({_id:1, StudName:"Jeevan", Grade:"VI",Hobbies:"InternetSurfing"});
WriteResult({ "nInserted" : 1 })
> db.Student.insert({_id:2, StudName:"Vamsi", Grade:"VI", Hobbies:["Watching Movies", "Reading Novels", "Drugs"]})
WriteResult({ "nInserted" : 1 })
> db.Student.find({});
{ "_id" : 1, "StudName" : "Jeevan", "Grade" : "VI", "Hobbies" : "InternetSurfing" }
{ "_id" : 2, "StudName" : "Vamsi", "Grade" : "VI", "Hobbies" : [ "Watching Movies", "Reading Novels", "Drugs" ] }
```

```
> db.Student.find({}).pretty();
{
        "_id" : 1,
        "StudName" : "Jeevan",
        "Grade" : "VI",
        "Hobbies" : "InternetSurfing"
}
{
        "_id" : 2,
        "StudName" : "Vamsi",
        "Grade" : "VI",
        "Hobbies" : [
                "Watching Movies",
                "Reading Novels",
                "Drugs"
        ]
}
> db.Student.insert({_id:3, StudName:"Sharat", Grade:"V", Hobbies:"Reading"});
WriteResult({ "nInserted" : 1 })
> db.Student.find({});
{ "_id" : 1, "StudName" : "Jeevan", "Grade" : "VI", "Hobbies" : "InternetSurfing" }
{ "_id" : 2, "StudName" : "Vamsi", "Grade" : "VI", "Hobbies" : [ "Watching Movies", "Reading Novels", "Drugs" ] }
{ "_id" : 3, "StudName" : "Sharat", "Grade" : "V", "Hobbies" : "Reading" }
```

```
> db.Student.update({_id:5, StudName:"Kusum", Grade:"VI"}, {$set:{Hobbies: ["Golf", "Sea Shell Collection"]}}, {upsert:true});
WriteResult({ "nMatched" : 0, "nUpserted" : 1, "nModified" : 0, "_id" : 5 })
> db.Student.find({}).pretty();
{
        "_id" : 1,
        "StudName" : "Jeevan",
        "Grade" : "VI",
        "Hobbies" : "InternetSurfing"
}
{
        "_id" : 2,
        "StudName" : "Vamsi",
        "Grade" : "VI",
        "Hobbies" : [
                "Watching Movies",
                "Reading Novels",
                "Drugs"
        ]
}
{ "_id" : 3, "StudName" : "Sharat", "Grade" : "V", "Hobbies" : "Reading" }
{
        "_id" : 5,
        "Grade" : "VI",
        "StudName" : "Kusum",
        "Hobbies" : [
                "Golf",
                "Sea Shell Collection"
        ]
}
> db.Student.find({StudName:"Kusum"});
{ "_id" : 5, "Grade" : "VI", "StudName" : "Kusum", "Hobbies" : [ "Golf", "Sea Shell Collection" ] }
```

```
> db.Student.find({StudName:"Kusum"}, {StudName:1, Grade:1}).pretty();
{ "_id" : 5, "Grade" : "VI", "StudName" : "Kusum" }
> db.Student.count();
4
> db.Student.find({}).sort({StudName:1});
{ "_id" : 1, "StudName" : "Jeevan", "Grade" : "VI", "Hobbies" : "InternetSurfing" }
{ "_id" : 5, "Grade" : "VI", "StudName" : "Kusum", "Hobbies" : [ "Golf", "Sea Shell Collection" ] }
{ "_id" : 3, "StudName" : "Sharat", "Grade" : "V", "Hobbies" : "Reading" }
{ "_id" : 2, "StudName" : "Vamsi", "Grade" : "VI", "Hobbies" : [ "Watching Movies", "Reading Novels", "Drugs" ] }
> db.Student.find({}).sort({StudName:-1});
{ "_id" : 2, "StudName" : "Vamsi", "Grade" : "VI", "Hobbies" : [ "Watching Movies", "Reading Novels", "Drugs" ] }
{ "_id" : 3, "StudName" : "Sharat", "Grade" : "V", "Hobbies" : "Reading" }
{ "_id" : 5, "Grade" : "VI", "StudName" : "Kusum", "Hobbies" : [ "Golf", "Sea Shell Collection" ] }
{ "_id" : 1, "StudName" : "Jeevan", "Grade" : "VI", "Hobbies" : "InternetSurfing" }
> db.Student.update({_id:5}, {$set:{Location:"NIHMANS"}});
WriteResult({ "nMatched" : 1, "nUpserted" : 0, "nModified" : 1 })
> db.Student.find({StudName:"Kusum"}).pretty();
{
        "_id" : 5,
        "Grade" : "VI",
        "StudName" : "Kusum",
        "Hobbies" : [
                "Golf",
                "Sea Shell Collection"
        ],
        "Location" : "NIHMANS"
}
```

```
> db.Student.find({}).skip(2).pretty();
{ "_id" : 3, "StudName" : "Sharat", "Grade" : "V", "Hobbies" : "Reading" }
{
        "_id" : 5,
        "Grade" : "VI",
        "StudName" : "Kusum",
        "Hobbies" : [
                "Golf",
                "Sea Shell Collection"
        ],
        "Location" : "NIHMANS"
}
> db.Student.find().skip(2).pretty();
{ "_id" : 3, "StudName" : "Sharat", "Grade" : "V", "Hobbies" : "Reading" }
{
        "_id" : 5,
        "Grade" : "VI",
        "StudName" : "Kusum",
        "Hobbies" : [
                "Golf",
                "Sea Shell Collection"
        ],
        "Location" : "NIHMANS"
}
> db.createCollection("food")
{ "ok" : 1 }
> db.food.insert({_id:1,fruits:['avacado','dragon fruit']})
WriteResult({ "nInserted" : 1 })
```

```
> db.food.insert({_id:1,fruits:['avacado','dragon fruit']})
WriteResult({ "nInserted" : 1 })
> db.food.insert({_id:2,fruits:['strawberry','dragon fruit']})
WriteResult({ "nInserted" : 1 })
> db.food.find({'fruits.1':'avacado'}).pretty()
> db.food.find().pretty()
{ "_id" : 1, "fruits" : [ "avacado", "dragon fruit" ] }
{ "_id" : 2, "fruits" : [ "strawberry", "dragon fruit" ] }
> db.food.find({'fruits.1':"avacado"}).pretty()
> db.food.find({'fruits.1':"avacado"})
> db.food.find({'fruits.0':"avacado"})
{ "_id" : 1, "fruits" : [ "avacado", "dragon fruit" ] }
> db.food.find({'fruits.0':"avacado"}).pretty()
{ "_id" : 1, "fruits" : [ "avacado", "dragon fruit" ] }
> db.food.find({'fruits.0':"avacado"}).pretty();
{ "_id" : 1, "fruits" : [ "avacado", "dragon fruit" ] }
> db.food.find({'fruits.0':{$size:2}}).pretty();
> db.food.find({'fruits':{$size:2}})
{ "_id" : 1, "fruits" : [ "avacado", "dragon fruit" ] }
{ "_id" : 2, "fruits" : [ "strawberry", "dragon fruit" ] }
> db.food.find({_id:2},{'fruits':{$slice:2}});
{ "_id" : 2, "fruits" : [ "strawberry", "dragon fruit" ] }
> db.food.find({_id:2},{'fruits':{$slice:1}});
{ "_id" : 2, "fruits" : [ "strawberry" ] }
> db.food.find({fruits:{$all:["avacado"]}})
{ "_id" : 1, "fruits" : [ "avacado", "dragon fruit" ] }
> db.food.find({fruits:{$all:["avacado","dragon fruit"]}})
{ "_id" : 1, "fruits" : [ "avacado", "dragon fruit" ] }
> db.food.find({fruits:{$all:["dragon fruit"]}})
{ "_id" : 1, "fruits" : [ "avacado", "dragon fruit" ] }
{ "_id" : 2, "fruits" : [ "strawberry", "dragon fruit" ] }
```

```
> show collections;
Student
customer
food
> db.customer.aggregate({$match:{AcctType:"FD"}},{$group:{_id:"$custID",TotalAccBal:{$sum:"$AcctBal"}}})
{ "_id" : 2, "TotalAccBal" : 20000000 }
{ "_id" : 1, "TotalAccBal" : 10000000 }
> db.customer.find()
{ "_id" : ObjectId("6253f945d7ce1043c6d5c8cc"), "custID" : 1, "AcctBal" : 10000000, "AcctType" : "FD" }
{ "_id" : ObjectId("6253f963d7ce1043c6d5c8cd"), "custID" : 2, "AcctBal" : 20000000, "AcctType" : "FD" }
{ "_id" : ObjectId("6253f973d7ce1043c6d5c8ce"), "custID" : 3, "AcctBal" : 30000000, "AcctType" : "RD" }
> db.customer.aggregate({$match:{AcctType:"FD"}},{$group:{_id:"$custID",TotalAccBal:{$sum:"$AcctBal"}}},{$match:{TotAccBal:{$gt:10000000}}});
> db.customer.aggregate({$match:{AcctType:"FD"}},{$group:{_id:"$custID",TotalAccBal:{$sum:"$AcctBal"}}},{$match:{TotalAccBal:{$gt:10000000}}});
{ "_id" : 2, "TotalAccBal" : 20000000 }
> quit()
```

BDA- Lab 2

1) Using MongoDB
i) Create a database for Students and Create a Student Collection (_id,Name, USN, Semester, Dept_Name, CGPA, Hobbies(Set)).
ii) Insert required documents to the collection.
iii) First Filter on “Dept_Name:CSE” and then group it on “Semester” and compute the Average CPGA for that semester and filter those documents where the “Avg_CPGA” is greater than 7.5.
iv) Command used to export MongoDB JSON documents from “Student” Collection into the “Students” database into a CSV file “Output.txt”.

```
bmsce@bmsce-Precision-T1700:~$ mongo
MongoDB shell version v3.6.8
connecting to: mongodb://127.0.0.1:27017
Implicit session: session { "id" : UUID("4419b91e-5b22-4f43-a52c-ac40a0bf73a6") }
MongoDB server version: 3.6.8
Server has startup warnings:
2022-04-20T19:31:53.425+0530 I STORAGE  [initandlisten]
2022-04-20T19:31:53.426+0530 I STORAGE  [initandlisten] ** WARNING: Using the XFS filesystem is strongly recommended with the WiredTiger storage engine
2022-04-20T19:31:53.426+0530 I STORAGE  [initandlisten] **          See http://dochub.mongodb.org/core/prodnotes-filesystem
2022-04-20T19:31:58.891+0530 I CONTROL  [initandlisten]
2022-04-20T19:31:58.891+0530 I CONTROL  [initandlisten] ** WARNING: Access control is not enabled for the database.
2022-04-20T19:31:58.891+0530 I CONTROL  [initandlisten] **          Read and write access to data and configuration is unrestricted.
2022-04-20T19:31:58.891+0530 I CONTROL  [initandlisten]
> use Niharika_db
switched to db Niharika_db
```

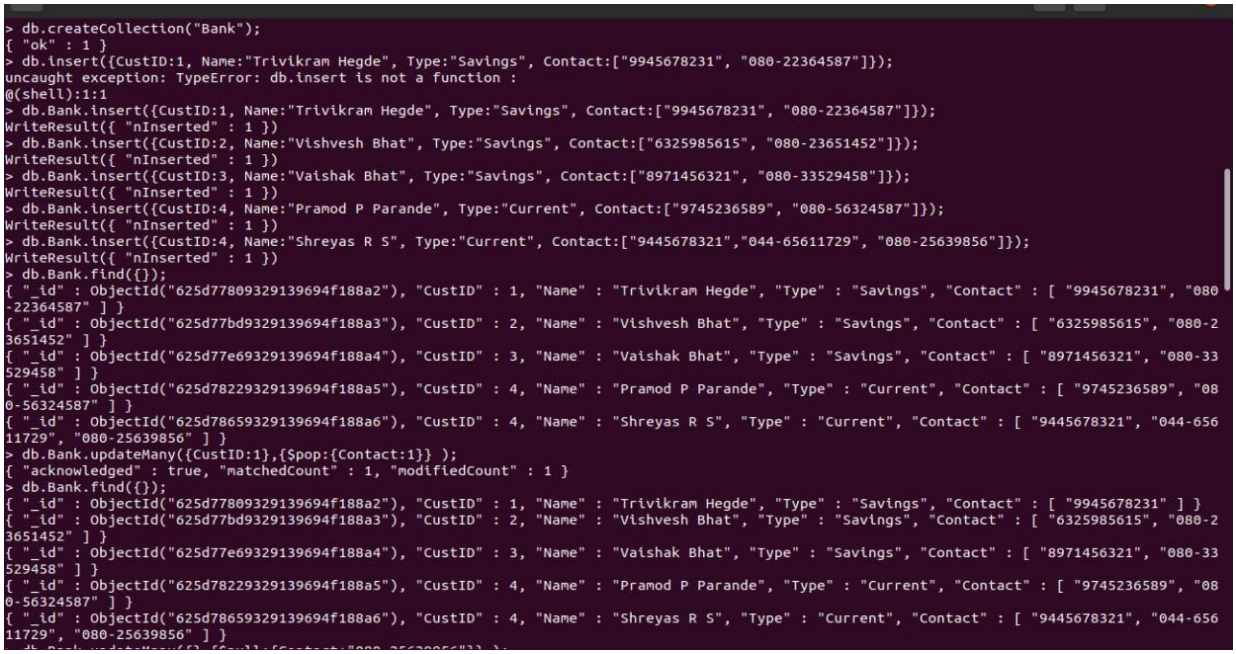
```
> db.Student.insert({_id:1,Name:"Aravind",USN:"1BM19CS001",Sem:6,Dept_name:"CSE",CGPA:"9.6",Hobbies:"Badminton"});
WriteResult({ "nInserted" : 1 })
> db.Student.insert({_id:2,Name:"Aman",USN:"1BM19EC002",Sem:7,Dept_name:"ECE",CGPA:"9.1",Hobbies:"Swimming"});
WriteResult({ "nInserted" : 1 })
> db.Student.insert({_id:3,Name:"Latha",USN:"1BM19CS003",Sem:6,Dept_name:"CSE",CGPA:"8.1",Hobbies:"Reading"});
WriteResult({ "nInserted" : 1 })
> db.Student.insert({_id:4,Name:"Sam",USN:"1BM19CS004",Sem:6,Dept_name:"CSE",CGPA:"6.5",Hobbies:"Cycling"});
WriteResult({ "nInserted" : 1 })
> db.Student.insert({_id:5,Name:"Suman",USN:"1BM19CS005",Sem:5,Dept_name:"CSE",CGPA:"7.6",Hobbies:"Cycling"});
WriteResult({ "nInserted" : 1 })
```

```
> db.Student.update({_id:1},{$set:{CGPA:9.6}})
WriteResult({ "nMatched" : 1, "nUpserted" : 0, "nModified" : 1 })
> db.Student.update({_id:2},{$set:{CGPA:9.1}})
WriteResult({ "nMatched" : 1, "nUpserted" : 0, "nModified" : 1 })
> db.Student.update({_id:3},{$set:{CGPA:8.1}})
WriteResult({ "nMatched" : 1, "nUpserted" : 0, "nModified" : 1 })
> db.Student.update({_id:4},{$set:{CGPA:6.5}})
WriteResult({ "nMatched" : 1, "nUpserted" : 0, "nModified" : 1 })
> db.Student.update({_id:5},{$set:{CGPA:8.6}})
WriteResult({ "nMatched" : 1, "nUpserted" : 0, "nModified" : 1 })
> db.Student.aggregate({$match:{Dept_name:"CSE"}},{$group:{_id:"$Sem",AvgCGPA:{$avg:"$CGPA"}}},{$match:{AvgCGPA:{$gt:7.5}}});
{ "_id" : 5, "AvgCGPA" : 8.6 }
{ "_id" : 6, "AvgCGPA" : 7.866666666666667 }
bmsce@bmsce-Precision-T1700:~$ mongoexport --host localhost --db Niharika_db --collection Student --csv --out /home/bmsce/Desktop/output.txt --fields "_id","Name","USN","Sem","Dept-name","CGPA","Hobbies";
2022-04-20T15:04:30.836+0530    csv flag is deprecated; please use --type=csv instead
2022-04-20T15:04:30.836+0530    connected to: localhost
2022-04-20T15:04:30.836+0530    exported 5 records
```

```
Open                                                    output.txt
                                                        ~/Desktop
_id,Name,USN,Sem,Dept-name,CGPA,Hobbies
1,Aravind,1BM19CS001,6,,9,Badminton
2,Aman,1BM19EC002,7,,9.1,Swimming
3,Latha,1BM19CS003,6,,8.1,Reading
4,Sam,1BM19CS004,6,,6.5,Cycling
5,Suman,1BM19CS005,5,,8.6,Cycling
```

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

1. Insert three documents
2. Use Arrays(Use Pull and Pop operation)
3. Use Index
4. Use Cursors
5. Updation



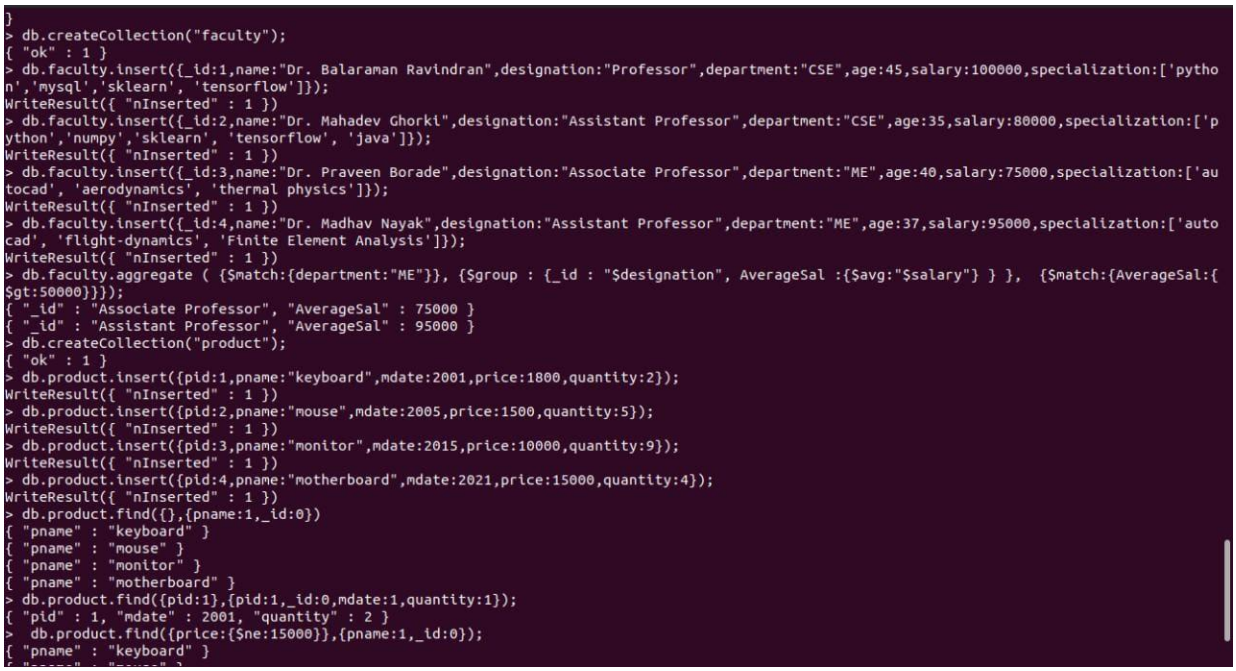
```
> db.createCollection("Bank");
{ "ok" : 1 }
> db.insert({CustID:1, Name:"Trivikram Hegde", Type:"Savings", Contact:["9945678231", "080-22364587"]});
uncaught exception: TypeError: db.insert is not a function :
@(shell):1:1
> db.Bank.insert({CustID:1, Name:"Trivikram Hegde", Type:"Savings", Contact:["9945678231", "080-22364587"]});
WriteResult({ "nInserted" : 1 })
> db.Bank.insert({CustID:2, Name:"Vishvesh Bhat", Type:"Savings", Contact:["6325985615", "080-23651452"]});
WriteResult({ "nInserted" : 1 })
> db.Bank.insert({CustID:3, Name:"Vaishak Bhat", Type:"Savings", Contact:["8971456321", "080-33529458"]});
WriteResult({ "nInserted" : 1 })
> db.Bank.insert({CustID:4, Name:"Pramod P Parande", Type:"Current", Contact:["9745236589", "080-56324587"]});
WriteResult({ "nInserted" : 1 })
> db.Bank.insert({CustID:4, Name:"Shreyas R S", Type:"Current", Contact:["9445678321","044-65611729", "080-25639856"]});
WriteResult({ "nInserted" : 1 })
> db.Bank.find({});
{ "_id" : ObjectId("625d77809329139694f188a2"), "CustID" : 1, "Name" : "Trivikram Hegde", "Type" : "Savings", "Contact" : [ "9945678231", "080
-22364587" ] }
{ "_id" : ObjectId("625d77bd9329139694f188a3"), "CustID" : 2, "Name" : "Vishvesh Bhat", "Type" : "Savings", "Contact" : [ "6325985615", "080-2
3651452" ] }
{ "_id" : ObjectId("625d77e69329139694f188a4"), "CustID" : 3, "Name" : "Vaishak Bhat", "Type" : "Savings", "Contact" : [ "8971456321", "080-33
529458" ] }
{ "_id" : ObjectId("625d78229329139694f188a5"), "CustID" : 4, "Name" : "Pramod P Parande", "Type" : "Current", "Contact" : [ "9745236589", "08
0-56324587" ] }
{ "_id" : ObjectId("625d78659329139694f188a6"), "CustID" : 4, "Name" : "Shreyas R S", "Type" : "Current", "Contact" : [ "9445678321", "044-656
11729", "080-25639856" ] }
> db.Bank.updateMany({CustID:1},{$pop:{Contact:1}} );
{ "acknowledged" : true, "matchedCount" : 1, "modifiedCount" : 1 }
> db.Bank.find({});
{ "_id" : ObjectId("625d77809329139694f188a2"), "CustID" : 1, "Name" : "Trivikram Hegde", "Type" : "Savings", "Contact" : [ "9945678231" ] }
{ "_id" : ObjectId("625d77bd9329139694f188a3"), "CustID" : 2, "Name" : "Vishvesh Bhat", "Type" : "Savings", "Contact" : [ "6325985615", "080-2
3651452" ] }
{ "_id" : ObjectId("625d77e69329139694f188a4"), "CustID" : 3, "Name" : "Vaishak Bhat", "Type" : "Savings", "Contact" : [ "8971456321", "080-33
529458" ] }
{ "_id" : ObjectId("625d78229329139694f188a5"), "CustID" : 4, "Name" : "Pramod P Parande", "Type" : "Current", "Contact" : [ "9745236589", "08
0-56324587" ] }
{ "_id" : ObjectId("625d78659329139694f188a6"), "CustID" : 4, "Name" : "Shreyas R S", "Type" : "Current", "Contact" : [ "9445678321", "044-656
11729", "080-25639856" ] }
```

```
{ "_id" : ObjectId("625d78659329139694f188a6"), "CustID" : 4, "Name" : "Shreyas R S", "Type" : "Current", "Contact" : [ "9445678321", "044-656
11729", "080-25639856" ] }
> db.Bank.updateMany({},{$pull:{Contact:"080-25639856"}} );
{ "acknowledged" : true, "matchedCount" : 5, "modifiedCount" : 1 }
> db.Bank.find({});
{ "_id" : ObjectId("625d77809329139694f188a2"), "CustID" : 1, "Name" : "Trivikram Hegde", "Type" : "Savings", "Contact" : [ "9945678231" ] }
{ "_id" : ObjectId("625d77bd9329139694f188a3"), "CustID" : 2, "Name" : "Vishvesh Bhat", "Type" : "Savings", "Contact" : [ "6325985615", "080-2
3651452" ] }
{ "_id" : ObjectId("625d77e69329139694f188a4"), "CustID" : 3, "Name" : "Vaishak Bhat", "Type" : "Savings", "Contact" : [ "8971456321", "080-33
529458" ] }
{ "_id" : ObjectId("625d78229329139694f188a5"), "CustID" : 4, "Name" : "Pramod P Parande", "Type" : "Current", "Contact" : [ "9745236589", "08
0-56324587" ] }
{ "_id" : ObjectId("625d78659329139694f188a6"), "CustID" : 4, "Name" : "Shreyas R S", "Type" : "Current", "Contact" : [ "9445678321", "044-656
11729" ] }
> db.Bank.createIndex({Name:1, Type:1},{name:});
uncaught exception: SyntaxError: expected expression, got '}' :
@(shell):1:43
> db.Bank.createIndex({Name:1, Type:1},{name:"Find current account holders"});
{
        "createdCollectionAutomatically" : false,
        "numIndexesBefore" : 1,
        "numIndexesAfter" : 2,
        "ok" : 1
}
> db.Bank.find({});
{ "_id" : ObjectId("625d77809329139694f188a2"), "CustID" : 1, "Name" : "Trivikram Hegde", "Type" : "Savings", "Contact" : [ "9945678231" ] }
{ "_id" : ObjectId("625d77bd9329139694f188a3"), "CustID" : 2, "Name" : "Vishvesh Bhat", "Type" : "Savings", "Contact" : [ "6325985615", "080-2
3651452" ] }
{ "_id" : ObjectId("625d77e69329139694f188a4"), "CustID" : 3, "Name" : "Vaishak Bhat", "Type" : "Savings", "Contact" : [ "8971456321", "080-33
529458" ] }
{ "_id" : ObjectId("625d78229329139694f188a5"), "CustID" : 4, "Name" : "Pramod P Parande", "Type" : "Current", "Contact" : [ "9745236589", "08
0-56324587" ] }
{ "_id" : ObjectId("625d78659329139694f188a6"), "CustID" : 4, "Name" : "Shreyas R S", "Type" : "Current", "Contact" : [ "9445678321", "044-656
11729" ] }
> db.Bank.getIndexes()
[
        {
                "v" : 2,
```

```
@(shell):1:20
> db.Bank.update({_id:625d78659329139694f188a6}, {$set: {CustID:5}},{upsert:true});
uncaught exception: SyntaxError: identifier starts immediately after numeric literal :
@(shell):1:20
> db.Bank.update({_id:"625d78659329139694f188a6"}, {$set: {CustID:5}},{upsert:true});
WriteResult({
        "nMatched" : 0,
        "nUpserted" : 1,
        "nModified" : 0,
        "_id" : "625d78659329139694f188a6"
})
> db.Bank.find({});
{ "_id" : ObjectId("625d77809329139694f188a2"), "CustID" : 1, "Name" : "Trivikram Hegde", "Type" : "Savings", "Contact" : [ "9945678231" ] }
{ "_id" : ObjectId("625d77bd9329139694f188a3"), "CustID" : 2, "Name" : "Vishvesh Bhat", "Type" : "Savings", "Contact" : [ "6325985615", "080-2
3651452" ] }
{ "_id" : ObjectId("625d77e69329139694f188a4"), "CustID" : 3, "Name" : "Vaishak Bhat", "Type" : "Savings", "Contact" : [ "8971456321", "080-33
529458" ] }
{ "_id" : ObjectId("625d78229329139694f188a5"), "CustID" : 4, "Name" : "Pramod P Parande", "Type" : "Current", "Contact" : [ "9745236589", "08
0-56324587" ] }
{ "_id" : ObjectId("625d78659329139694f188a6"), "CustID" : 4, "Name" : "Shreyas R S", "Type" : "Current", "Contact" : [ "9445678321", "044-656
11729" ] }
{ "_id" : "625d78659329139694f188a6", "CustID" : 5 }
> db.Bank.update({_id:"625d78659329139694f188a6", CustID:5}, {$set: {Name:"Sumantha K S", Type:"Savings", Contact:["9856321478","011-65897458"
]}},{upsert:true});
WriteResult({ "nMatched" : 1, "nUpserted" : 0, "nModified" : 1 })
> db.Bank.find({});
{ "_id" : ObjectId("625d77809329139694f188a2"), "CustID" : 1, "Name" : "Trivikram Hegde", "Type" : "Savings", "Contact" : [ "9945678231" ] }
{ "_id" : ObjectId("625d77bd9329139694f188a3"), "CustID" : 2, "Name" : "Vishvesh Bhat", "Type" : "Savings", "Contact" : [ "6325985615", "080-2
3651452" ] }
{ "_id" : ObjectId("625d77e69329139694f188a4"), "CustID" : 3, "Name" : "Vaishak Bhat", "Type" : "Savings", "Contact" : [ "8971456321", "080-33
529458" ] }
{ "_id" : ObjectId("625d78229329139694f188a5"), "CustID" : 4, "Name" : "Pramod P Parande", "Type" : "Current", "Contact" : [ "9745236589", "08
0-56324587" ] }
{ "_id" : ObjectId("625d78659329139694f188a6"), "CustID" : 4, "Name" : "Shreyas R S", "Type" : "Current", "Contact" : [ "9445678321", "044-656
11729" ] }
{ "_id" : "625d78659329139694f188a6", "CustID" : 5, "Contact" : [ "9856321478", "011-65897458" ], "Name" : "Sumantha K S", "Type" : "Savings"
}
>
```

1) Using MongoDB,
i) Create a database for Faculty and Create a Faculty Collection(Faculty_id, Name, Designation ,Department, Age, Salary, Specialization(Set)).
ii) Insert required documents to the collection.
iii) First Filter on “Dept_Name:MECH” and then group it on “Designation” and compute the Average Salary for that Designation and filter those documents where the “Avg_Sal” is greater than 650000.
iv) Demonstrate usage of import and export commands

Write MongoDB queries for the following:
1) To display only the product name from all the documents of the product collection.
2) To display only the Product ID, ExpiryDate as well as the quantity from the document of the product collection where the _id column is 1.
3) To find those documents where the price is not set to 15000.
4) To find those documents from the Product collection where the quantity is set to 9 and the product name is set to ‘monitor’.
5) To find documents from the Product collection where the Product name ends in ‘d’.



```
}
> db.createCollection("faculty");
{ "ok" : 1 }
> db.faculty.insert({_id:1,name:"Dr. Balaraman Ravindran",designation:"Professor",department:"CSE",age:45,salary:100000,specialization:['pytho
n','mysql','sklearn', 'tensorflow']});
WriteResult({ "nInserted" : 1 })
> db.faculty.insert({_id:2,name:"Dr. Mahadev Ghorki",designation:"Assistant Professor",department:"CSE",age:35,salary:80000,specialization:['p
ython','numpy','sklearn', 'tensorflow', 'java']});
WriteResult({ "nInserted" : 1 })
> db.faculty.insert({_id:3,name:"Dr. Praveen Borade",designation:"Associate Professor",department:"ME",age:40,salary:75000,specialization:['au
tocad', 'aerodynamics', 'thermal physics']});
WriteResult({ "nInserted" : 1 })
> db.faculty.insert({_id:4,name:"Dr. Madhav Nayak",designation:"Assistant Professor",department:"ME",age:37,salary:95000,specialization:['auto
cad', 'flight-dynamics', 'Finite Element Analysis']});
WriteResult({ "nInserted" : 1 })
> db.faculty.aggregate ( {$match:{department:"ME"}}, {$group : {_id : "$designation", AverageSal :{$avg:"$salary"} } },  {$match:{AverageSal:{
$gt:50000}}});
{ "_id" : "Associate Professor", "AverageSal" : 75000 }
{ "_id" : "Assistant Professor", "AverageSal" : 95000 }
> db.createCollection("product");
{ "ok" : 1 }
> db.product.insert({pid:1,pname:"keyboard",mdate:2001,price:1800,quantity:2});
WriteResult({ "nInserted" : 1 })
> db.product.insert({pid:2,pname:"mouse",mdate:2005,price:1500,quantity:5});
WriteResult({ "nInserted" : 1 })
> db.product.insert({pid:3,pname:"monitor",mdate:2015,price:10000,quantity:9});
WriteResult({ "nInserted" : 1 })
> db.product.insert({pid:4,pname:"motherboard",mdate:2021,price:15000,quantity:4});
WriteResult({ "nInserted" : 1 })
> db.product.find({},{pname:1,_id:0})
{ "pname" : "keyboard" }
{ "pname" : "mouse" }
{ "pname" : "monitor" }
{ "pname" : "motherboard" }
> db.product.find({pid:1},{pid:1,_id:0,mdate:1,quantity:1});
{ "pid" : 1, "mdate" : 2001, "quantity" : 2 }
>  db.product.find({price:{$ne:15000}},{pname:1,_id:0});
{ "pname" : "keyboard" }
```

3)Create a mongodb collection Hospital. Demonstrate the following by choosing fields of your choice.

1. Insert three documents
2. Use Arrays(Use Pull and Pop operation)
3. Use Index
4. Use Cursors
5. Updation

```
{ "pname" : "motherboard" }
> db.product.find({pid:1},{pid:1,_id:0,mdate:1,quantity:1});
{ "pid" : 1, "mdate" : 2001, "quantity" : 2 }
>  db.product.find({price:{$ne:15000}},{pname:1,_id:0});
{ "pname" : "keyboard" }
{ "pname" : "mouse" }
{ "pname" : "monitor" }
> db.product.find({$and:[{quantity:{$eq:9}},{pname:{$eq:"monitor"}}]},{pname:1,_id:0})
{ "pname" : "monitor" }
> db.product.find({pname:/d$/},{pname:1,quantity:1,_id:0})
{ "pname" : "keyboard", "quantity" : 2 }
{ "pname" : "motherboard", "quantity" : 4 }
> db.createCollection("hospital");
{ "ok" : 1 }
> db.hospital.insert({_id:1, Name: "Anshuman Agarwal", age:23, diseases:["fever", "diarrhoea", "wheezing", "gastritis"]});
WriteResult({ "nInserted" : 1 })
> db.hospital.insert({_id:2, Name: "Pinky Chaubey", age:35, diseases:["fever","nausea", "food infection", "indigestion", "kidney stones"]});
WriteResult({ "nInserted" : 1 })
> db.hospital.insert({_id:3, Name: "Amresh Chowpati", age:63, diseases:["hyperglycemia", "diabetes mellitus", "food poisoning", "cold"]});
WriteResult({ "nInserted" : 1 })
> db.hospital.updateMany({},{$pull:{diseases:"fever"}});
{ "acknowledged" : true, "matchedCount" : 3, "modifiedCount" : 2 }
> db.hospital.updateOne({_id:1},{$pop:{diseases:-1}});
{ "acknowledged" : true, "matchedCount" : 1, "modifiedCount" : 1 }
> db.hospital.find({"diseases.2":"nausea"});
> db.hospital.find({"diseases.1":"nausea"});
> d.hospital.find({});
uncaught exception: ReferenceError: d is not defined :
@(shell):1:1
> db.hospital.find({});
{ "_id" : 1, "Name" : "Anshuman Agarwal", "age" : 23, "diseases" : [ "wheezing", "gastritis" ] }
{ "_id" : 2, "Name" : "Pinky Chaubey", "age" : 35, "diseases" : [ "nausea", "food infection", "indigestion", "kidney stones" ] }
{ "_id" : 3, "Name" : "Amresh Chowpati", "age" : 63, "diseases" : [ "hyperglycemia", "diabetes mellitus", "food poisoning", "cold" ] }
> db.hospital.find({"diseases.0":"nausea"});
{ "_id" : 2, "Name" : "Pinky Chaubey", "age" : 35, "diseases" : [ "nausea", "food infection", "indigestion", "kidney stones" ] }
> db.hospital.update({_id:3},{$set:{'diseases.1':'sarscov'}});
WriteResult({ "nMatched" : 1, "nUpserted" : 0, "nModified" : 1 })
>
```

BDA LAB 3

Program 1. Perform the following DB operations using Cassandra.

1. Create a key space by name Employee

```
cqlsh> CREATE KEYSPACE Employee WITH replication = {'class': 'SimpleStrategy', 'replication_factor': 1};
cqlsh> describe keyspace
No keyspace specified and no current keyspace
cqlsh> describe Employee;
```

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

```
cqlsh> create table Employee.Employee_Info(Emp_Id int Primary Key,Emp_Name text,Designation text,Date_of_Joining timestamp,Salary dou
ble,Dept_Name text);
```

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

 emp_id | date_of_joining | dept_name | designation | emp_name | salary
--------+-----------------+-----------+-------------+----------+--------

(0 rows)
```

3. Insert the values into the table in batch

```
cqlsh> begin batch insert into Employee.Employee_Info(emp_id,date_of_joining,dept_name,designation,emp_name,salary)values(1,'2021-06-
03','Deployment','Manager','Kusum',1500000.50); apply batch;
cqlsh> select * from Employee.Employee_Info;

 emp_id | date_of_joining                 | dept_name  | designation | emp_name | salary
--------+---------------------------------+------------+-------------+----------+---------
      1 | 2021-06-03 00:00:00.000000+0000 | Deployment |     Manager |    Kusum | 1.5e+06

(1 rows)
```

```
cqlsh> begin batch
   ... insert into Employee.Employee_Info(emp_id,date_of_joining,dept_name,designation,emp_name,salary)values(2,'2020-09-03','Develop
ment','Web developer','Karan',1700000.50);
   ... insert into Employee.Employee_Info(emp_id,date_of_joining,dept_name,designation,emp_name,salary)values(121,'2019-05-03','R&D',
'Intern','Kia',2000000.50);
   ... apply batch;
cqlsh> select * from Employee.Employee_Info;

 emp_id | date_of_joining                 | dept_name   | designation   | emp_name | salary
--------+---------------------------------+-------------+---------------+----------+---------
      1 | 2021-06-03 00:00:00.000000+0000 |  Deployment |       Manager |    Kusum | 1.5e+06
      2 | 2020-09-03 00:00:00.000000+0000 | Development | Web developer |    Karan | 1.7e+06
    121 | 2019-05-03 00:00:00.000000+0000 |         R&D |        Intern |      Kia |   2e+06

(3 rows)
```

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

```
cqlsh> update Employee.Employee_Info SET emp_name='Kushi',dept_name='Testing' where emp_id=121;
cqlsh> select * from Employee.Employee_Info;

 emp_id | date_of_joining                 | dept_name   | designation    | emp_name | salary
--------+---------------------------------+-------------+----------------+----------+---------
      1 | 2021-06-03 00:00:00.000000+0000 |  Deployment |        Manager |    Kusum | 1.5e+06
      2 | 2020-09-03 00:00:00.000000+0000 | Development | Web developer |    Karan | 1.7e+06
    121 | 2019-05-03 00:00:00.000000+0000 |     Testing |         Intern |    Kushi |   2e+06

(3 rows)
```

5. Sort the details of Employee records based on salary

```
cqlsh> create table Employee.emp(Emp_Id int,Emp_name text,Designation text,Date_Of_Joining timestamp,Salary double,Dept_Name text,primary ke
y(Emp_Id,Salary));
cqlsh> begin batch
   ... insert into Employee.emp(emp_id,salary,date_of_joining,dept_name,designation,emp_name)values(1,1500000.50,'2021-06-03','Deployment','
Manager','Kusum');
   ... insert into Employee.emp(emp_id,salary,date_of_joining,dept_name,designation,emp_name)values(2,1100000.50,'2022-05-03','Development',
'Web Developer','Karan');
   ... insert into Employee.emp(emp_id,salary,date_of_joining,dept_name,designation,emp_name)values(121,1900000.50,'2022-05-03','R&D','Inter
n','Kia');
   ... apply batch;
cqlsh> select * from Employee.emp;

 emp_id | salary  | date_of_joining                 | dept_name   | designation    | emp_name
--------+---------+---------------------------------+-------------+----------------+----------
      1 | 1.5e+06 | 2021-06-03 00:00:00.000000+0000 |  Deployment |        Manager |    Kusum
      2 | 1.1e+06 | 2022-05-03 00:00:00.000000+0000 | Development | Web Developer |    Karan
    121 | 1.9e+06 | 2022-05-03 00:00:00.000000+0000 |         R&D |         Intern |      Kia

(3 rows)
cqlsh> paging off;
Disabled Query paging.
cqlsh> select * from Employee.emp where emp_id in (1,2,121) order by salary;

 emp_id | salary  | date_of_joining                 | dept_name   | designation    | emp_name
--------+---------+---------------------------------+-------------+----------------+----------
      2 | 1.1e+06 | 2022-05-03 00:00:00.000000+0000 | Development | Web Developer |    Karan
      1 | 1.5e+06 | 2021-06-03 00:00:00.000000+0000 |  Deployment |        Manager |    Kusum
    121 | 1.9e+06 | 2022-05-03 00:00:00.000000+0000 |         R&D |         Intern |      Kia

(3 rows)
cqlsh>
```

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

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

 emp_id | date_of_joining                 | dept_name   | designation    | emp_name | projects | salary
--------+---------------------------------+-------------+----------------+----------+----------+---------
      1 | 2021-06-03 00:00:00.000000+0000 |  Deployment |        Manager |    Kusum |     null | 1.5e+06
      2 | 2020-09-03 00:00:00.000000+0000 | Development | Web developer |    Karan |     null | 1.7e+06
    121 | 2019-05-03 00:00:00.000000+0000 |     Testing |         Intern |    Kushi |     null |   2e+06

(3 rows)
```

7. Update the altered table to add project names.

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

 emp_id | date_of_joining                 | dept_name   | designation    | emp_name | projects       | salary
--------+---------------------------------+-------------+----------------+----------+----------------+---------
      1 | 2021-06-03 00:00:00.000000+0000 |  Deployment |        Manager |    Kusum | {'abc', 'xyz'} | 1.5e+06
      2 | 2020-09-03 00:00:00.000000+0000 | Development | Web developer |    Karan |           null | 1.7e+06
    121 | 2019-05-03 00:00:00.000000+0000 |     Testing |         Intern |    Kushi |           null |   2e+06

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

 emp_id | date_of_joining                 | dept_name   | designation    | emp_name | projects                                | salary
--------+---------------------------------+-------------+----------------+----------+-----------------------------------------+---------
      1 | 2021-06-03 00:00:00.000000+0000 |  Deployment |        Manager |    Kusum |                        {'abc', 'xyz'} | 1.5e+06
      2 | 2020-09-03 00:00:00.000000+0000 | Development | Web developer |    Karan | {'def', 'lmn', 'pqr', 'tuv'} | 1.7e+06
    121 | 2019-05-03 00:00:00.000000+0000 |     Testing |         Intern |    Kushi |                                    null |   2e+06

(3 rows)

cqlsh> update Employee.Employee_Info set projects=projects+{'lab','jkl'} where emp_id=121;
cqlsh> select * from Employee.Employee_Info;

 emp_id | date_of_joining                 | dept_name   | designation    | emp_name | projects                                | salary
--------+---------------------------------+-------------+----------------+----------+-----------------------------------------+---------
      1 | 2021-06-03 00:00:00.000000+0000 |  Deployment |        Manager |    Kusum |                        {'abc', 'xyz'} | 1.5e+06
      2 | 2020-09-03 00:00:00.000000+0000 | Development | Web developer |    Karan | {'def', 'lmn', 'pqr', 'tuv'} | 1.7e+06
    121 | 2019-05-03 00:00:00.000000+0000 |     Testing |         Intern |    Kushi |                        {'lab', 'jkl'} |   2e+06

(3 rows)
```

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

```
cqlsh> insert into Employee.Employee_Info(emp_id,date_of_joining,dept_name,designation,emp_name,salary)values(11,'2019-05-05','R&D','
Intern','Kajal',1000000.50) using TTL 15;
cqlsh> select * from Employee.Employee_Info;

 emp_id | date_of_joining                 | dept_name   | designation    | emp_name | projects                                | salary
--------+---------------------------------+-------------+----------------+----------+-----------------------------------------+---------
     11 | 2019-05-05 00:00:00.000000+0000 |         R&D |         Intern |    Kajal |                                    null |   1e+06
      1 | 2021-06-03 00:00:00.000000+0000 |  Deployment |        Manager |    Kusum |                        {'abc', 'xyz'} | 1.5e+06
      2 | 2020-09-03 00:00:00.000000+0000 | Development | Web developer |    Karan | {'def', 'lmn', 'pqr', 'tuv'} | 1.7e+06
    121 | 2019-05-03 00:00:00.000000+0000 |     Testing |         Intern |    Kushi |                        {'lab', 'jkl'} |   2e+06

(4 rows)
cqlsh> select * from Employee.Employee_Info;

 emp_id | date_of_joining                 | dept_name   | designation    | emp_name | projects                                | salary
--------+---------------------------------+-------------+----------------+----------+-----------------------------------------+---------
      1 | 2021-06-03 00:00:00.000000+0000 |  Deployment |        Manager |    Kusum |                        {'abc', 'xyz'} | 1.5e+06
      2 | 2020-09-03 00:00:00.000000+0000 | Development | Web developer |    Karan | {'def', 'lmn', 'pqr', 'tuv'} | 1.7e+06
    121 | 2019-05-03 00:00:00.000000+0000 |     Testing |         Intern |    Kushi |                        {'lab', 'jkl'} |   2e+06

(3 rows)
cqlsh>
```

LAB-4

Perform the following DB operations using Cassandra:

1 Create a key space by name Library

```
cqlsh> CREATE KEYSPACE LIBRARY WITH replication = {'class':'SimpleStrategy','replication_factor':3};
cqlsh> Use LIBRARY;
cqlsh: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.

```
cqlsh:library> create table library_info(stud_id int, counter_value Counter, stud_name text,book_name text, date_of_issue timestamp, book_id int, PRIMARY KEY(stud_id,stud_name,book_name,date_of
_issue,book_id));
```

```
cqlsh:library> select * from library.library_info;

 stud_id | stud_name | book_name | date_of_issue | book_id | counter_value
---------+-----------+-----------+---------------+---------+---------------

(0 rows)
```

3. Insert the values into the table in batch

```
cqlsh:library> UPDATE library_info SET counter_value = counter_value + 1 WHERE stud_id = 111 and stud_name = 'SAM' and book_name = 'ML' and date_of_issue = '2020-10-11' and book_id = 200;
cqlsh:library> UPDATE library_info SET counter_value = counter_value + 1 WHERE stud_id = 113 and stud_name = 'SHAAN' and book_name = 'BDA' and date_of_issue = '2020-09-21' and book_id = 300;
cqlsh:library> UPDATE library_info SET counter_value = counter_value + 1 WHERE stud_id = 113 and stud_name = 'AYMAN' and book_name = 'OOMD' and date_of_issue = '2020-04-01' and book_id = 400;
```

```
cqlsh:library> select * from library.library_info;

 stud_id | stud_name | book_name | date_of_issue                   | book_id | counter_value
---------+-----------+-----------+---------------------------------+---------+---------------
     111 |       SAM |        ML | 2020-10-10 18:30:00.000000+0000 |     200 |             1
     113 |     AYMAN |      OOMD | 2020-03-31 18:30:00.000000+0000 |     400 |             1
     112 |     SHAAN |       BDA | 2020-09-20 18:30:00.000000+0000 |     300 |             1

(3 rows)
```

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

```
cqlsh:library> UPDATE library_info SET counter_value = counter_value + 1 WHERE stud_id = 112 and stud_name = 'SHAAN' and book_name = 'BDA' and date_of_issue = '2020-09-21' and book_id = 300;
```

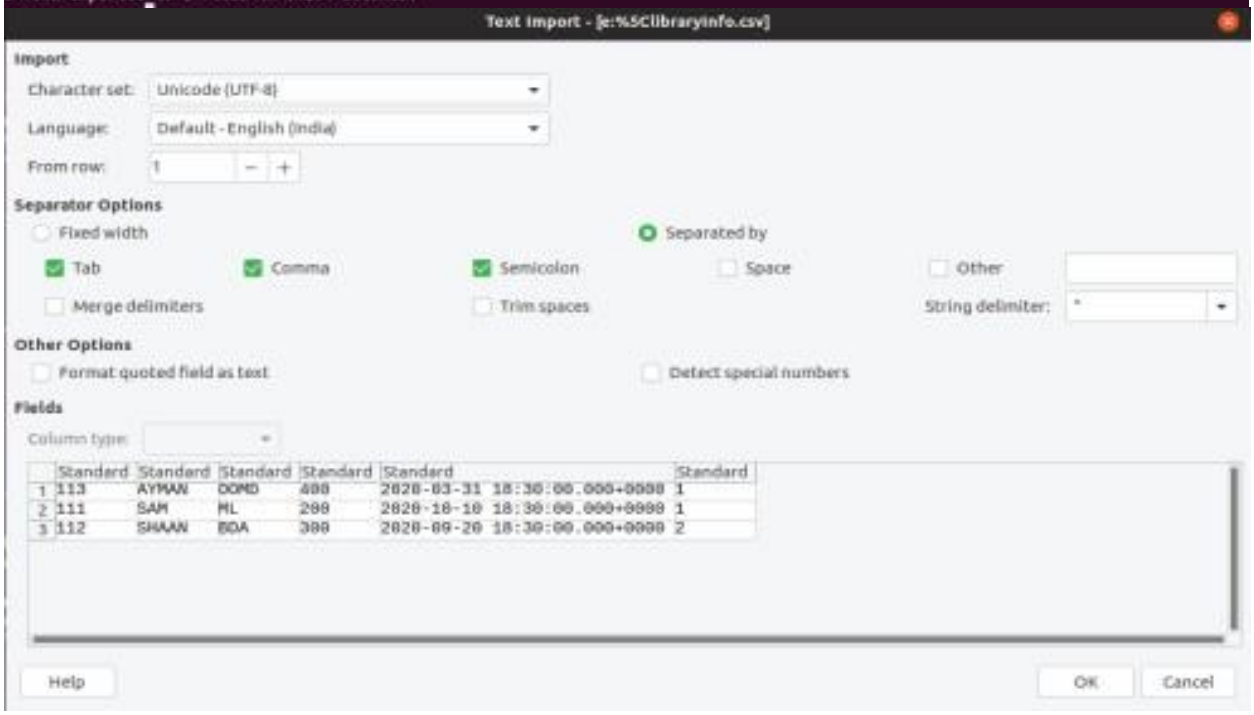
```
cqlsh:library> select * from library.library_info;

 stud_id | stud_name | book_name | date_of_issue                   | book_id | counter_value
---------+-----------+-----------+---------------------------------+---------+---------------
     111 |       SAM |        ML | 2020-10-10 18:30:00.000000+0000 |     200 |             1
     113 |     AYMAN |      OOMD | 2020-03-31 18:30:00.000000+0000 |     400 |             1
     112 |     SHAAN |       BDA | 2020-09-20 18:30:00.000000+0000 |     300 |             2

(3 rows)
```

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

```
cqlsh:library> SELECT * FROM library_info WHERE stud_id = 112;

 stud_id | stud_name | book_name | date_of_issue                   | book_id | counter_value
---------+-----------+-----------+---------------------------------+---------+---------------
     112 |     SHAAN |       BDA | 2020-09-20 18:30:00.000000+0000 |     300 |             2

(1 rows)
```

6. Export the created column to a csv file

```
cqlsh:library> COPY Library_Info(Stud_Id,Stud_Name,Book_Name,Book_Id,Date_Of_Issue,Counter_value) TO 'e:\libraryInfo.csv';
Using 11 child processes

Starting copy of library.library_info with columns [stud_id, stud_name, book_name, book_id, date_of_issue, counter_value].
Processed: 3 rows; Rate:      17 rows/s; Avg. rate:      17 rows/s
3 rows exported to 1 files in 0.204 seconds.
```



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

```
cqlsh:library> create table library_info2(stud_id int, counter_value counter, stud_name text,book_name text, date_of_issue timestamp, book_id int, PRIMARY KEY(stud_id,stud_name,book_name,date_of_issue,
book_id));
```

```
cqlsh:library> SELECT * FROM library_info2;

 stud_id | stud_name | book_name | date_of_issue | book_id | counter_value
---------+-----------+-----------+---------------+---------+---------------

(0 rows)
```

```
cqlsh:library> COPY library_info2(stud_id,stud_name,book_name,book_id,date_of_issue,counter_value) FROM 'e:\libraryInfo.csv';
Using 11 child processes

Starting copy of library.library_info2 with columns [stud_id, stud_name, book_name, book_id, date_of_issue, counter_value].
Processed: 3 rows; Rate:       5 rows/s; Avg. rate:       7 rows/s
3 rows imported from 1 files in 0.416 seconds (0 skipped).
```