

VISVESVARAYA TECHNOLOGICAL UNIVERSITY

“JnanaSangama”, Belgaum -590014, Karnataka.



LAB REPORT

on

BIG DATA ANALYTICS (20CS6PEBDA)

Submitted by

NISHCHAL NANDAGOPAL (1BM19CS105)

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 **NISHCHAL NANDAGOPAL(1BM19CS105)**, 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 **Course Title - (Course code)**work prescribed for the said degree.

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

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

Index Sheet

Sl. No.	Experiment Title	Page No.
1.	<u>MongoDB Lab Program 1 (CRUD Demonstration):</u> - Students should be classifying a dataset into one of the standard forms and apply suitable querying rules to obtain suitable results	4
2.	<u>MongoDB Lab Program 2 (CRUD Demonstration):</u> - Students should be classifying a dataset into one of the standard forms and apply suitable querying rules to obtain suitable results	12
3.	<u>Cassandra Lab Program 1:</u> - Create a Data set either structured/Semi-Structured/Unstructured from Twitter/Facebook etc. to perform various DB operations using Cassandra. (Use the Face Pager app to perform real-time streaming)	17
4.	<u>Cassandra Lab Program 2:</u> - Create a Data set either structured/Semi-Structured/Unstructured from Twitter/Facebook etc. to perform various DB operations using Cassandra. (Use the Face Pager app to perform real-time streaming)	19

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

WEEK 1

1. CREATE DATABASE IN MONGODB.

```
bmsce@bmsce-Precision-T1700:~$ mongo
MongoDB shell version v3.6.8
connecting to: mongodb://127.0.0.1:27017
Implicit session: session { "id" : UUID("feb93b57-25eb-4dd5-881e-f3dd433915db") }
MongoDB server version: 3.6.8
Server has startup warnings:
2022-04-13T19:39:14.350+0530 I STORAGE [initandlisten] ** WARNING: Using the XFS filesystem is strongly recommended with the WiredTiger storage engine
2022-04-13T19:39:14.350+0530 I STORAGE [initandlisten] ** See http://dochub.mongodb.org/core/prodnotes-filesystem
2022-04-13T19:39:18.570+0530 I CONTROL [initandlisten] ** WARNING: Access control is not enabled for the database.
2022-04-13T19:39:18.570+0530 I CONTROL [initandlisten] ** Read and write access to data and configuration is unrestricted.
2022-04-13T19:39:18.570+0530 I CONTROL [initandlisten]
> use nayana_db
switched to db nayana_db
> db
nayana_db
>
```

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

- 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");
{ "ok" : 1 }
```

- Create a collection by the name "Students" and store the following data in it.

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

```
> db.Student.insert({_id:2,StudName:"MikeHassan",Grade:"VII",Hobbies:"Swimming"});
WriteResult({ "nInserted" : 1 })
> db.Student.update({_id:3,StudName:"AryanDavid",Grade:"VII"},{$set:{Hobbies:"Skating"}},{upsert:true});
WriteResult({ "nMatched" : 0, "nUpserted" : 1, "nModified" : 0, "_id" : 3 })
> db.Student.insert({_id:4,StudName:"DuaLipa",Grade:"VII",Hobbies:"Singing"});
WriteResult({ "nInserted" : 1 })
> db.Student.insert({_id:5,StudName:"RajeshBharadwaj",Grade:"VII",Hobbies:"Badminton"});
WriteResult({ "nInserted" : 1 })
```

- FIND METHOD

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

```
> db.Student.find({StudName:"DuaLipa"});
{ "_id" : 4, "StudName" : "DuaLipa", "Grade" : "VII", "Hobbies" : "Singing" }
```

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.

C. To

```
> db.Student.find({}, {StudName:1, Grade:1, _id:0});
{ "StudName" : "MichelleJacintha", "Grade" : "VII" }
{ "StudName" : "MikeHassan", "Grade" : "VII" }
{ "Grade" : "VII", "StudName" : "AryanDavid" }
{ "StudName" : "DuaLipa", "Grade" : "VII" }
{ "StudName" : "RajeshBharadwaj", "Grade" : "VII" }
```

find

those documents where the Grade is set to 'VII'

```
> db.Student.find({Grade:{ $eq:'VII' }}).pretty();
{
  "_id" : 1,
  "StudName" : "MichelleJacintha",
  "Grade" : "VII",
  "Hobbies" : "InternetSurfing"
}
{
  "_id" : 2,
  "StudName" : "MikeHassan",
  "Grade" : "VII",
  "Hobbies" : "Swimming"
}
{
  "_id" : 3,
  "Grade" : "VII",
  "StudName" : "AryanDavid",
  "Hobbies" : "Skating"
}
{
  "_id" : 4,
  "StudName" : "DuaLipa",
  "Grade" : "VII",
  "Hobbies" : "Singing"
}
{
  "_id" : 5,
  "StudName" : "RajeshBharadwaj",
  "Grade" : "VII",
  "Hobbies" : "Badminton"
}
```

D. To find those documents from the Students collection where the Hobbies is set to either 'singing' or is set to 'Skating'.

```
> db.Student.find({Hobbies :{ $in: ['Singing','Skating']}}).pretty ();
{
  "_id" : 3,
  "Grade" : "VII",
  "StudName" : "AryanDavid",
  "Hobbies" : "Skating"
}
{
  "_id" : 4,
  "StudName" : "DuaLipa",
  "Grade" : "VII",
  "Hobbies" : "Singing"
}
```

E. To find documents from the Students collection where the StudName begins with "R".

F. To

```
> db.Student.find({StudName:/^R/}).pretty();
{
  "_id" : 5,
  "StudName" : "RajeshBharadwaj",
  "Grade" : "VII",
  "Hobbies" : "Badminton"
}
```

find

documents from the Students collection where the StudName has an "a" in any position.

```

> db.Student.find({StudName:/a/}).pretty();
{
  "_id" : 1,
  "StudName" : "MichelleJacintha",
  "Grade" : "VII",
  "Hobbies" : "InternetSurfing"
}
{
  "_id" : 2,
  "StudName" : "MikeHassan",
  "Grade" : "VII",
  "Hobbies" : "Swimming"
}
{
  "_id" : 3,
  "Grade" : "VII",
  "StudName" : "AryanDavid",
  "Hobbies" : "Skating"
}
{
  "_id" : 4,
  "StudName" : "DuaLipa",
  "Grade" : "VII",
  "Hobbies" : "Singing"
}
{
  "_id" : 5,
  "StudName" : "RajeshBharadwaj",
  "Grade" : "VII",
  "Hobbies" : "Badminton"
}

```

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

```

> db.Student.count();
5

```

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

```

> db.Student.find().sort({StudName:-1}).pretty();
{
  "_id" : 5,
  "StudName" : "RajeshBharadwaj",
  "Grade" : "VII",
  "Hobbies" : "Badminton"
}
{
  "_id" : 2,
  "StudName" : "MikeHassan",
  "Grade" : "VII",
  "Hobbies" : "Swimming"
}
{
  "_id" : 3,
  "Grade" : "VII",
  "StudName" : "AryanDavid",
  "Hobbies" : "Skating"
}
{
  "_id" : 4,
  "StudName" : "DuaLipa",
  "Grade" : "VII",
  "Hobbies" : "Singing"
}

```

3.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 existing document.

```
> db.Student.save({StudName:"Vamsi", Grade:"VI"})
WriteResult({ "nInserted" : 1 })
```

Add a new field to existing Document:

```
> db.Student.update({_id:4},{ $set:{Location:"Network"}})
WriteResult({ "nMatched" : 1, "nUpserted" : 0, "nModified" : 1 })
> db.Student.find({});
{ "_id" : 1, "StudName" : "MichelleJacintha", "Grade" : "VII", "Hobbies" : "InternetSurfing" }
{ "_id" : 2, "StudName" : "MikeHassan", "Grade" : "VII", "Hobbies" : "Swimming" }
{ "_id" : 3, "Grade" : "VII", "StudName" : "AryanDavid", "Hobbies" : "Skating" }
{ "_id" : 4, "StudName" : "DuaLipa", "Grade" : "VII", "Hobbies" : "Singing", "Location" : "Network" }
{ "_id" : 5, "StudName" : "RajeshBharadwaj", "Grade" : "VII", "Hobbies" : "Badminton" }
{ "_id" : ObjectId("62569a60a083074f5c1a00a8"), "StudName" : "Vamsi", "Grade" : "VI" }
```

Remove the field in an existing Document

```
> db.Student.update({_id:4},{ $unset:{Location:"Network"}})
WriteResult({ "nMatched" : 1, "nUpserted" : 0, "nModified" : 1 })
> db.Student.find({});
{ "_id" : 1, "StudName" : "MichelleJacintha", "Grade" : "VII", "Hobbies" : "InternetSurfing" }
{ "_id" : 2, "StudName" : "MikeHassan", "Grade" : "VII", "Hobbies" : "Swimming" }
{ "_id" : 3, "Grade" : "VII", "StudName" : "AryanDavid", "Hobbies" : "Skating" }
{ "_id" : 4, "StudName" : "DuaLipa", "Grade" : "VII", "Hobbies" : "Singing" }
{ "_id" : 5, "StudName" : "RajeshBharadwaj", "Grade" : "VII", "Hobbies" : "Badminton" }
{ "_id" : ObjectId("62569a60a083074f5c1a00a8"), "StudName" : "Vamsi", "Grade" : "VI" }
```

Finding Document based on search criteria suppressing few fields

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

To

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

```
> db.Student.find({Grade:{$ne:'VII'}}).pretty();
{
  "_id" : ObjectId("62569a60a083074f5c1a00a8"),
  "StudName" : "Vamsi",
  "Grade" : "VI"
}
```

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

```
> db.Student.find({StudName:/n$/}).pretty();
{
  "_id" : 2,
  "StudName" : "MikeHassan",
  "Grade" : "VII",
  "Hobbies" : "Swimming"
}
```


to set a particular field value to NULL

```
> db.Student.update({_id:3},{set:{Hobbies:null}})
WriteResult({ "nMatched" : 1, "nUpserted" : 0, "nModified" : 1 })
> db.Student.find({});
{ "_id" : 1, "StudName" : "MichelleJacintha", "Grade" : "VII", "Hobbies" : "InternetSurfing" }
{ "_id" : 2, "StudName" : "MikeHassan", "Grade" : "VII", "Hobbies" : "Swimming" }
{ "_id" : 3, "Grade" : "VII", "StudName" : "AryanDavid", "Hobbies" : null }
{ "_id" : 4, "StudName" : "DuaLipa", "Grade" : "VII", "Hobbies" : "Singing" }
{ "_id" : 5, "StudName" : "RajeshBharadwaj", "Grade" : "VII", "Hobbies" : "Badminton" }
{ "_id" : ObjectId("62569a60a083074f5c1a00a8"), "StudName" : "Vamsi", "Grade" : "VI" }
>
```

Count the number of documents in Student Collections

```
> db.Student.count()
6
```

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

```
> db.Student.count({Grade:"VII"})
5
```

food database using mongodb

Create a collection by name “food” and add to each document add a “fruits” array

```
> db.food.insert( { _id:1, fruits:['grapes','mango','apple'] } )
WriteResult({ "nInserted" : 1 })
> db.food.insert( { _id:2, fruits:['grapes','mango','cherry'] } )
WriteResult({ "nInserted" : 1 })
> db.food.insert( { _id:3, fruits:['banana','mango'] } )
WriteResult({ "nInserted" : 1 })
```

To find those documents from the “food” collection which has the “fruits array” constitute of “grapes”, “mango” and “apple”.

```
> db.food.find ( {fruits: ['grapes','mango','apple'] } ).pretty();
{ "_id" : 1, "fruits" : [ "grapes", "mango", "apple" ] }
```

To find all the documents from the food collection which have elements mango and grapes in the array "fruits"

```
> db.food.find({fruits:{$all:["mango","grapes"]}})
{ "_id" : 1, "fruits" : [ "grapes", "mango", "apple" ] }
{ "_id" : 2, "fruits" : [ "grapes", "mango", "cherry" ] }
```

update on Array:

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

```
> db.food.update({_id:3},{ $set:{'fruits.1':'apple'}})
WriteResult({ "nMatched" : 1, "nUpserted" : 0, "nModified" : 1 })
> db.food.find({});
{ "_id" : 1, "fruits" : [ "grapes", "mango", "apple" ] }
{ "_id" : 2, "fruits" : [ "grapes", "mango", "cherry" ] }
{ "_id" : 3, "fruits" : [ "banana", "apple" ] }
>
```

WEEK 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 CGPA for that semester and filter those documents where the "Avg_CGPA" 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".

```
"ted" : 1 })
"ted" : 1 })
"ted" : 1 })
"ted" : 1 })
"ted" : 1 })
group:{_id:"$Sem",AvgCGPA:{$avg:"$CGPA"}},{ $match:{AvgCGPA:{$gt:7.5}}});
oup:{_id:"$Sem",AvgCGPA:{$avg:"$CGPA"}},{ $match:{AvgCGPA:{$gt:7.5}}});
```

```

1 |_id,Name,USN,Sem,Dept_Name,CGPA,Hobbies
2 1,,1BM19CS095,6,CSE,9,Badminton
3 2,,1BM19CS002,6,CSE,9.1,Swimming
4 3,,1BM19CS006,6,CSE,8.1,Cycling
5 4,,1BM19CS010,6,CSE,6.5,Reading
6 5,,1BM19CS090,6,CSE,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", "080-23651452" ] }
{ "_id" : ObjectId("625d77e69329139694f188a4"), "CustID" : 3, "Name" : "Vaishak Bhat", "Type" : "Savings", "Contact" : [ "8971456321", "080-33529458" ] }
{ "_id" : ObjectId("625d78229329139694f188a5"), "CustID" : 4, "Name" : "Pramod P Parande", "Type" : "Current", "Contact" : [ "9745236589", "080-56324587" ] }
{ "_id" : ObjectId("625d78659329139694f188a6"), "CustID" : 4, "Name" : "Shreyas R S", "Type" : "Current", "Contact" : [ "9445678321", "044-65611729", "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", "080-23651452" ] }
{ "_id" : ObjectId("625d77e69329139694f188a4"), "CustID" : 3, "Name" : "Vaishak Bhat", "Type" : "Savings", "Contact" : [ "8971456321", "080-33529458" ] }
{ "_id" : ObjectId("625d78229329139694f188a5"), "CustID" : 4, "Name" : "Pramod P Parande", "Type" : "Current", "Contact" : [ "9745236589", "080-56324587" ] }
{ "_id" : ObjectId("625d78659329139694f188a6"), "CustID" : 4, "Name" : "Shreyas R S", "Type" : "Current", "Contact" : [ "9445678321", "044-65611729", "080-25639856" ] }

```

```

{ "_id" : ObjectId("625d78659329139694f188a6"), "CustID" : 4, "Name" : "Shreyas R S", "Type" : "Current", "Contact" : [ "9445678321", "044-65611729", "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", "080-23651452" ] }
{ "_id" : ObjectId("625d77e69329139694f188a4"), "CustID" : 3, "Name" : "Vaishak Bhat", "Type" : "Savings", "Contact" : [ "8971456321", "080-33529458" ] }
{ "_id" : ObjectId("625d78229329139694f188a5"), "CustID" : 4, "Name" : "Pramod P Parande", "Type" : "Current", "Contact" : [ "9745236589", "080-56324587" ] }
{ "_id" : ObjectId("625d78659329139694f188a6"), "CustID" : 4, "Name" : "Shreyas R S", "Type" : "Current", "Contact" : [ "9445678321", "044-65611729" ] }
> 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", "080-23651452" ] }
{ "_id" : ObjectId("625d77e69329139694f188a4"), "CustID" : 3, "Name" : "Vaishak Bhat", "Type" : "Savings", "Contact" : [ "8971456321", "080-33529458" ] }
{ "_id" : ObjectId("625d78229329139694f188a5"), "CustID" : 4, "Name" : "Pramod P Parande", "Type" : "Current", "Contact" : [ "9745236589", "080-56324587" ] }
{ "_id" : ObjectId("625d78659329139694f188a6"), "CustID" : 4, "Name" : "Shreyas R S", "Type" : "Current", "Contact" : [ "9445678321", "044-65611729" ] }
> db.Bank.getIndexes()
[
  {
    "v" : 2,
    "name" : "Find current account holders",
    "key" : {
      "Name" : 1,
      "Type" : 1
    },
    "ns" : "bank",
    "partialFilterExpression" : {},
    "options" : {
      "unique" : false
    },
    "autoIndex" : true
  },
  {
    "v" : 2,
    "name" : "_id_",
    "key" : {
      "_id" : 1
    },
    "ns" : "bank",
    "partialFilterExpression" : {},
    "options" : {
      "unique" : true
    },
    "autoIndex" : true
  }
]

```

```

@:(shell):1:20
> db.Bank.update({_id:625d78659329139694f188a6}, {$set: {CustID:5}}, {upsert:true});
uncaught exception: SyntaxError: identifiier starts immediately after numeric literal :
@:(shell):1:20
> db.Bank.update({_id:"625d78659329139694f188a6"}, {$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", "080-23651452" ] }
{ "_id" : ObjectId("625d77e69329139694f188a4"), "CustID" : 3, "Name" : "Vaishak Bhat", "Type" : "Savings", "Contact" : [ "8971456321", "080-33529458" ] }
{ "_id" : ObjectId("625d78229329139694f188a5"), "CustID" : 4, "Name" : "Pramod P Parande", "Type" : "Current", "Contact" : [ "9745236589", "080-56324587" ] }
{ "_id" : ObjectId("625d78659329139694f188a6"), "CustID" : 4, "Name" : "Shreyas R S", "Type" : "Current", "Contact" : [ "9445678321", "044-65611729" ] }
{ "_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", "080-23651452" ] }
{ "_id" : ObjectId("625d77e69329139694f188a4"), "CustID" : 3, "Name" : "Vaishak Bhat", "Type" : "Savings", "Contact" : [ "8971456321", "080-33529458" ] }
{ "_id" : ObjectId("625d78229329139694f188a5"), "CustID" : 4, "Name" : "Pramod P Parande", "Type" : "Current", "Contact" : [ "9745236589", "080-56324587" ] }
{ "_id" : ObjectId("625d78659329139694f188a6"), "CustID" : 4, "Name" : "Shreyas R S", "Type" : "Current", "Contact" : [ "9445678321", "044-65611729" ] }
{ "_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:['python','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:['python','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:['autocad','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:['autocad','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" }
{ "pname" : "mouse" }
```

3) Create a mongodb collection Hospital. Demonstrate the following by choosing fields of 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 })
>
```

WEEK 3

Program 1. Perform the following DB operations using Cassandra.

1. Create a key space by name Employeee

```
cqlsh> CREATE KEYSPACE Employeee WITH replication = {'class':'SimpleStrategy','replication_factor':1};
cqlsh> describe Employeee
```

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

```
cqlsh> create table Employeee.Employeee_info(Exp_ID int Primary Key,Emp_Name text,Designation text,DOJ timestamp,Salary double,Dept_Name text);
cqlsh> select * from Employeee.Employeee_info
...
...
...
...
cqlsh> select * from Employeee.Employeee_info;

exp_id | dept_name | designation | doj | emp_name | salary
-----+-----+-----+-----+-----+-----
1 | Deployment | Manager | 2021-04-30 18:30:00.000000+0000 | anagha | 1e+06
2 | Development | Web Developer | 2021-05-31 18:30:00.000000+0000 | apoorva | 1.7e+06
31 | Deployment | Manager | 2021-05-08 18:30:00.000000+0000 | rohan | 1.0087e+06
```

3. Insert the values into the table in batch

```
cqlsh> begin batch insert into Employeee.Employeee_info(exp_id,dept_name,designation,doy,emp_name,salary)values(1,'Deployment','Manager','2021-05-01','anagha',1000000.50);apply batch;
cqlsh> begin batch insert into Employeee.Employeee_info(exp_id,dept_name,designation,doy,emp_name,salary)values(2,'Development','Web Developer','2021-06-01','apoorva',1700000.50);apply batch;
cqlsh> begin batch insert into Employeee.Employeee_info(exp_id,dept_name,designation,doy,emp_name,salary)values(31,'Deployment','Manager','2021-05-09','rohan',1008700.50);apply batch;
cqlsh> select * from Employeee.Employeee_info;

exp_id | dept_name | designation | doj | emp_name | salary
-----+-----+-----+-----+-----+-----
1 | Deployment | Manager | 2021-04-30 18:30:00.000000+0000 | anagha | 1e+06
2 | Development | Web Developer | 2021-05-31 18:30:00.000000+0000 | apoorva | 1.7e+06
31 | Deployment | Manager | 2021-05-08 18:30:00.000000+0000 | rohan | 1.0087e+06
(3 rows)
```

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

```
cqlsh> update Employeee.Employeee_info SET emp_name='Kushi',dept_name='Testing' where exp_id=121;
cqlsh> select * from Employeee.Employeee_info;

exp_id | dept_name | designation | doj | emp_name | salary
-----+-----+-----+-----+-----+-----
1 | Deployment | Manager | 2021-04-30 18:30:00.000000+0000 | anagha | 1e+06
2 | Development | Web Developer | 2021-05-31 18:30:00.000000+0000 | apoorva | 1.7e+06
121 | Testing | null | null | Kushi | null
31 | Deployment | Manager | 2021-05-08 18:30:00.000000+0000 | rohan | 1.0087e+06
```

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 Employeee.Employeee_info add Projects set<text>;
cqlsh> select * from Employeee.Employeee_info;
```

exp_id	dept_name	designation	doj	emp_name	projects	salary
1	Deployment	Manager	2021-04-30 18:30:00.000000+0000	anagha	null	1e+06
2	Development	Web Developer	2021-05-31 18:30:00.000000+0000	apoorva	null	1.7e+06
121	Testing	null	null	Kushi	null	null
31	Deployment	Manager	2021-05-08 18:30:00.000000+0000	rohan	null	1.0087e+06

(4 rows)

7. Update the altered table to add project names.

```
cqlsh> update Employeee.Employeee_info set Projects=Projects+{'abc','xyz','pqr'} where exp_id=2;
cqlsh> update Employeee.Employeee_info set Projects=Projects+{'abc'} where exp_id=121;
cqlsh> select * from Employeee.Employeee_info;
```

exp_id	dept_name	designation	doj	emp_name	projects	salary
1	Deployment	Manager	2021-04-30 18:30:00.000000+0000	anagha	{'abc', 'xyz'}	1e+06
2	Development	Web Developer	2021-05-31 18:30:00.000000+0000	apoorva	{'abc', 'pqr', 'xyz'}	1.7e+06
121	Testing	null	null	Kushi	{'abc'}	null
31	Deployment	Manager	2021-05-08 18:30:00.000000+0000	rohan	null	1.0087e+06

(4 rows)

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

```
cqlsh> insert into Employeee.Employeee_info(exp_id,dept_name,designation,doj,emp_name,salary)values(21,'RAD','Intern','2022-05-01','a
atra',1034000.03) using TTL 15;
cqlsh> select * from Employeee.Employeee_info;
```

exp_id	dept_name	designation	doj	emp_name	projects	salary
1	Deployment	Manager	2021-04-30 18:30:00.000000+0000	anagha	{'abc', 'xyz'}	1e+06
2	Development	Web Developer	2021-05-31 18:30:00.000000+0000	apoorva	{'abc', 'pqr', 'xyz'}	1.7e+06
121	Testing	null	null	Kushi	{'abc'}	null
21	RAD	Intern	2022-04-30 18:30:00.000000+0000	atra	null	1.034e+06
31	Deployment	Manager	2021-05-08 18:30:00.000000+0000	rohan	null	1.0087e+06

(5 rows)

```
cqlsh> select * from Employeee.Employeee_info;
```

exp_id	dept_name	designation	doj	emp_name	projects	salary
1	Deployment	Manager	2021-04-30 18:30:00.000000+0000	anagha	{'abc', 'xyz'}	1e+06
2	Development	Web Developer	2021-05-31 18:30:00.000000+0000	apoorva	{'abc', 'pqr', 'xyz'}	1.7e+06
121	Testing	null	null	Kushi	{'abc'}	null
31	Deployment	Manager	2021-05-08 18:30:00.000000+0000	rohan	null	1.0087e+06

(4 rows)

WEEK 4

Program 2:

1 Create a key space by name Library

```
cqlsh> CREATE KEYSPACE LIBRARY1 WITH replication = {'class':'SimpleStrategy','replication_factor':3};
cqlsh> use LIBRARY1
...
cqlsh> use LIBRARY1;
cqlsh:library1> 
```

2.Create a column family by name Library-Info with attributes

Stud_Id Primary Key,

Counter_value of type Counter,

Stud_Name, Book-Name, Book-Id, Date_of_issue

```
...
cqlsh:library1> SELECT * FROM library_info;

stud_id | stud_name | book_name | date_of_issue | book_id | counter_value
-----+-----+-----+-----+-----+-----
(0 rows)
```

```
cqlsh:library1> 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));
```

3.Insert the values into the table in batch

```
cqlsh:library1> SELECT * FROM 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	2
113	AYMAN	OOMD	2020-03-31 18:30:00.000000+0000	400	2
112	SHAAN	BDA	2020-09-20 18:30:00.000000+0000	300	2

(3 rows)

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

```
cqlsh:library1> UPDATE library_info SET counter_value = counter_value + 1 WHERE stud_id = 112 and stud_name = 'SHAAN' and book_name = 'BDA' and date_of_issue = '2020-09-21'and book_id = 300;
cqlsh:library1> SELECT * FROM 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	2
113	AYMAN	OOMD	2020-03-31 18:30:00.000000+0000	400	2
112	SHAAN	BDA	2020-09-20 18:30:00.000000+0000	300	4

(3 rows)

5. Write a query to show that a student with id 113 has taken a book “OOMD” 2 times.

```
cqlsh:library1> SELECT * FROM library_info WHERE stud_id = 113;
```

stud_id	stud_name	book_name	date_of_issue	book_id	counter_value
113	AYMAN	OOMD	2020-03-31 18:30:00.000000+0000	400	2

(1 rows)

6. Export the created column to a csv file

```
cqlsh:library1> 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 library1.library_info with columns [stud_id, stud_name, book_name, book_id, date_of_issue, counter_value].
Processed: 3 rows; Rate:      21 rows/s; Avg. rate:      21 rows/s
3 rows exported to 1 files in 0.172 seconds.
cqlsh:library1>
```

Text Import - [e:\libraryInfo.csv]

Import

Character set:

Language:

From row:

Separator Options

☐ Fixed width ☒ Separated by

☒ Tab ☒ Comma ☒ Semicolon ☐ Space ☐ Other

☐ Merge delimiters ☐ Trim spaces String delimiter:

Other Options

☐ Format quoted field as text ☐ Detect special numbers

Fields

Column type:

	Standard	Standard	Standard	Standard	Standard
1	113	AYMAN	OOMD	400	2020-03-31 18:30:00.000+0000
2	111	SAM	ML	200	2020-10-10 18:30:00.000+0000
3	112	SHAAN	BDA	300	2020-09-20 18:30:00.000+0000

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

```
cqlsh:library1> create table library_info3(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:library1> COPY library_info3(stud_id,stud_name,book_name,book_id,date_of_issue,counter_value) FROM 'e:\libraryInfo.csv';
Using 11 child processes
```

```
Starting copy of library1.library_info3 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 0.403 seconds (0 skipped).
```

```
cqlsh:library1> select * from library_info3;
```

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	2
113	AYMAN	OOMD	2020-03-31 18:30:00.000000+0000	400	2
112	SHAAN	BDA	2020-09-20 18:30:00.000000+0000	300	4

(3 rows)