Experiment No: Group_C_03 **Date:** 05/08/2024

Name: Kushal Kishor Shankhapal **Subject:** ADBMS LAB

Roll No: 61

Aim:

Execute at least 10 queries on any suitable MongoDB database that demonstrates following:

- $ where queries
- Cursors (Limits, skips, sorts, advanced query options)
- Database commands

Objectives:

1. To learn SQL DCL, DDL commands.
2. To Learn ER diagram and its features.

Commands:

1. sudo service mongod start: Start MongoDB Server

```
it@IT-LL-14:~$ sudo service mongod start
[sudo] password for it:
```

2. mongosh: Connect to MongoDB Shell

```
it@IT-LL-14:~$ mongosh
Current Mongosh Log ID:	66b0a2f5c061469d42149f47
Connecting to:		mongodb://127.0.0.1:27017/?
directConnection=true&serverSelectionTimeoutMS=2000&appName=mongosh+2.2.10
Using MongoDB:		7.0.12
Using Mongosh:		2.2.10
mongosh 2.2.15 is available for download: https://www.mongodb.com/try/download/shell

For mongosh info see: https://docs.mongodb.com/mongodb-shell/

------
   The server generated these startup warnings when booting
   2024-08-05T15:13:26.964+05:30: Using the XFS filesystem is strongly recommended with the
WiredTiger storage engine. See http://dochub.mongodb.org/core/prodnotes-filesystem
```

```
  2024-08-05T15:13:31.150+05:30: Access control is not enabled for the database. Read and write access
to data and configuration is unrestricted
  2024-08-05T15:13:31.150+05:30: vm.max_map_count is too low
------
```

3. use: Create a Database

```
test> use collegeDB
switched to db collegeDB
collegeDB> // Create a collection named 'students' and insert sample documents
```

4. insertMany(): Create a Collection and Insert Sample Data

```
collegeDB> db.students.insertMany([
...     { name: "Alice", age: 22, major: "Computer Science", credits: 30, debits: 30 },
...     { name: "Bob", age: 23, major: "Mathematics", credits: 25, debdebits: 20 },
...     { name: "Charlie", age: 24, major: "Physics", credits: 35, debdebits: 35 },
...     { name: "David", age: 22, major: "Computer Science", credits: 40, debits: 20 },
...     { name: "Eve", age: 21, major: "Biology", credits: 45, debits: 45 }
... ])
{
  acknowledged: true,
  insertedIds: {
    '0': ObjectId('66b0a305c061469d42149f48'),
    '1': ObjectId('66b0a305c061469d42149f49'),
    '2': ObjectId('66b0a305c061469d42149f4a'),
    '3': ObjectId('66b0a305c061469d42149f4b'),
    '4': ObjectId('66b0a305c061469d42149f4c')
  }
}
```

5. find(): Find students where credits equal debits using $where

```
collegeDB> db.students.find({ $where: "this.credits == this.debits" })
[
  {
    _id: ObjectId('66b0a305c061469d42149f48'),
    name: 'Alice',
    age: 22,
    major: 'Computer Science',
    credits: 30,
    debits: 30
```

```
  },
  {
    _id: ObjectId('66b0a305c061469d42149f4a'),
    name: 'Charlie',
    age: 24,
    major: 'Physics',
    credits: 35,
    debits: 35
  },
  {
    _id: ObjectId('66b0a305c061469d42149f4c'),
    name: 'Eve',
    age: 21,
    major: 'Biology',
    credits: 45,
    debits: 45
  }
]
```

6. limit(): Limit the number of results to 2

```
collegeDB> db.students.find().limit(2)
[
  {
    _id: ObjectId('66b0a305c061469d42149f48'),
    name: 'Alice',
    age: 22,
    major: 'Computer Science',
    credits: 30,
    debits: 30
  },
  {
    _id: ObjectId('66b0a305c061469d42149f49'),
    name: 'Bob',
    age: 23,
    major: 'Mathematics',
    credits: 25,
    debits: 20
  }
]
```

7. skip(): Skip the first 2 documents and return the next set

```
collegeDB> db.students.find().skip(2)
[
  {
    _id: ObjectId('66b0a305c061469d42149f4a'),
    name: 'Charlie',
    age: 24,
    major: 'Physics',
    credits: 35,
    debits: 35
  },
  {
    _id: ObjectId('66b0a305c061469d42149f4b'),
    name: 'David',
    age: 22,
    major: 'Computer Science',
    credits: 40,
    debits: 20
  },
  {
    _id: ObjectId('66b0a305c061469d42149f4c'),
    name: 'Eve',
    age: 21,
    major: 'Biology',
    credits: 45,
    debits: 45
  }
]
```

8. sort(): Sort students by age in descending order

```
collegeDB> db.students.find().sort({ age: -1 })
[
  {
    _id: ObjectId('66b0a305c061469d42149f4a'),
    name: 'Charlie',
    age: 24,
    major: 'Physics',
    credits: 35,
    debits: 35
```

```
  },
  {
    _id: ObjectId('66b0a305c061469d42149f49'),
    name: 'Bob',
    age: 23,
    major: 'Mathematics',
    credits: 25,
    debits: 20
  },
  {
    _id: ObjectId('66b0a305c061469d42149f48'),
    name: 'Alice',
    age: 22,
    major: 'Computer Science',
    credits: 30,
    debits: 30
  },
  {
    _id: ObjectId('66b0a305c061469d42149f4b'),
    name: 'David',
    age: 22,
```

9. sort(): Find students older than 22, sort by age, and limit to 2 results

```
collegeDB> db.students.find({ age: { $gt: 22 } }).sort({ age: 1 }).limit(2)
[
  {
    _id: ObjectId('66b0a305c061469d42149f49'),
    name: 'Bob',
    age: 23,
    major: 'Mathematics',
    credits: 25,
    debits: 20
  },
  {
    _id: ObjectId('66b0a305c061469d42149f4a'),
    name: 'Charlie',
    age: 24,
    major: 'Physics',
    credits: 35,
```

```
    debits: 35
  }
]
```

10. countDocuments(): Count the number of students in the collection

```
collegeDB> db.students.countDocuments()
5
```

11. distinct(): Get distinct majors from the collection

```
collegeDB> db.students.distinct("major")
[ 'Biology', 'Computer Science', 'Mathematics', 'Physics' ]
```

12. aggregate(): Group by major and get the count of students in each major

```
collegeDB> db.students.aggregate([
...     { $group: { _id: "$major", count: { $sum: 1 } } }
... ])
[
  { _id: 'Biology', count: 1 },
  { _id: 'Computer Science', count: 2 },
  { _id: 'Physics', count: 1 },
  { _id: 'Mathematics', count: 1 }
]
```

13. mapReduce(): Map-Reduce to calculate total credits for each major

```
collegeDB> db.students.mapReduce(
...     function() { emit(this.major, this.credits); },
...     function(key, values) { return Array.sum(values); },
...     { out: "total_credits_per_major" }
... )
DeprecationWarning: Collection.mapReduce() is deprecated. Use an aggregation instead.
See https://docs.mongodb.com/manual/core/map-reduce for details.
{ result: 'total_credits_per_major', ok: 1 }
```

14. runCommand(): Run the 'dbStats' command to get statistics about the database

```
collegeDB> db.runCommand({ dbStats: 1 })
{
  db: 'collegeDB',
  collections: Long('2'),
  views: Long('0'),
```

```
  objects: Long('9'),
  avgObjSize: 70.66666666666667,
  dataSize: 636,
  storageSize: 40960,
  indexes: Long('2'),
  indexSize: 40960,
  totalSize: 81920,
  scaleFactor: Long('1'),
  fsUsedSize: 15607996416,
  fsTotalSize: 38652256256,
  ok: 1
}
```

16: quit(): Exit the shell.

```
collegeDB> quit()
it@IT-LL-14:~$
```