Task – 12

Mongodb CRUD Project

Objective:

To study MongoDB and understand how to perform basic CRUD operations such as Insert, Read, Update, and Delete using MongoDB shell.

1. Start MongoDB Shell

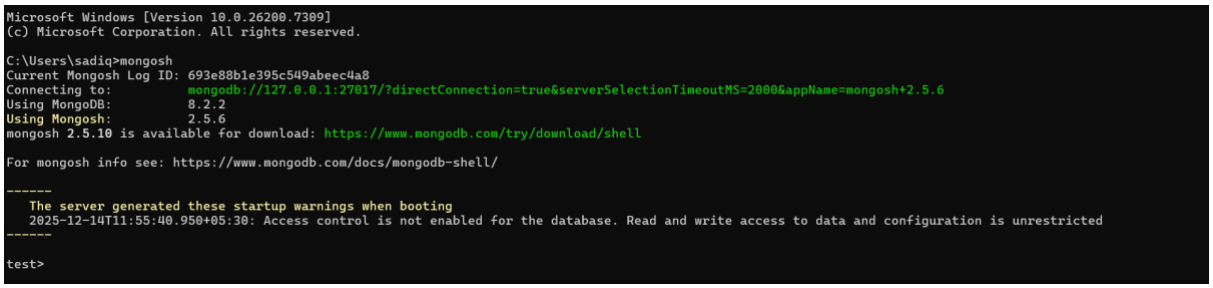
Open your terminal or command prompt and type:

mongosh

You should see a prompt like:

test>

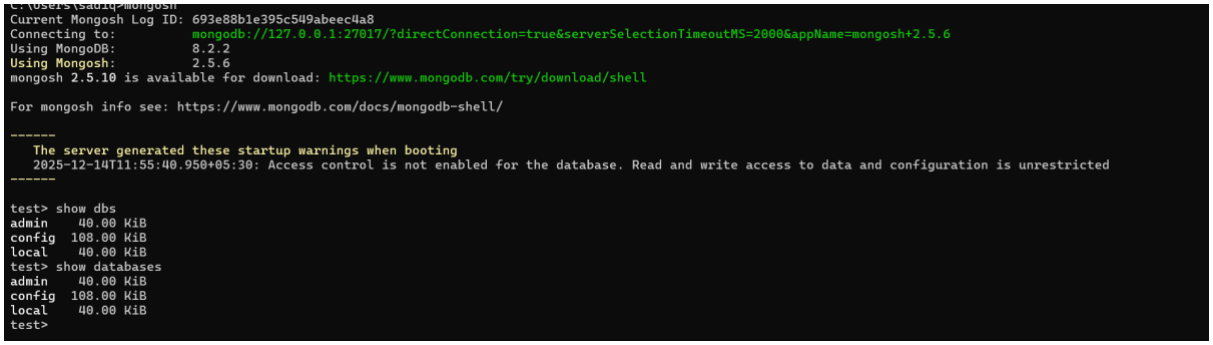
test is the default database.



b. Display Existing Databases

The following command is used to display all existing databases available in MongoDB

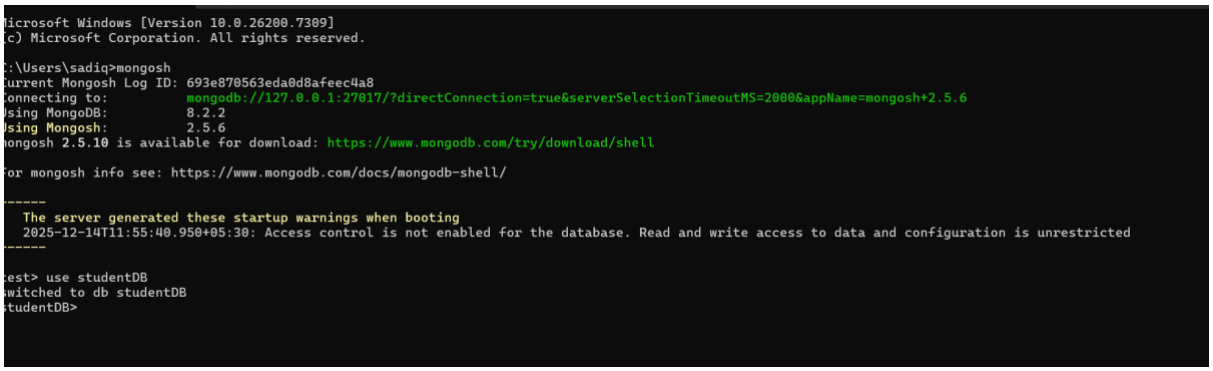
test> show dbs or show databases



2. Database & Collection Setup

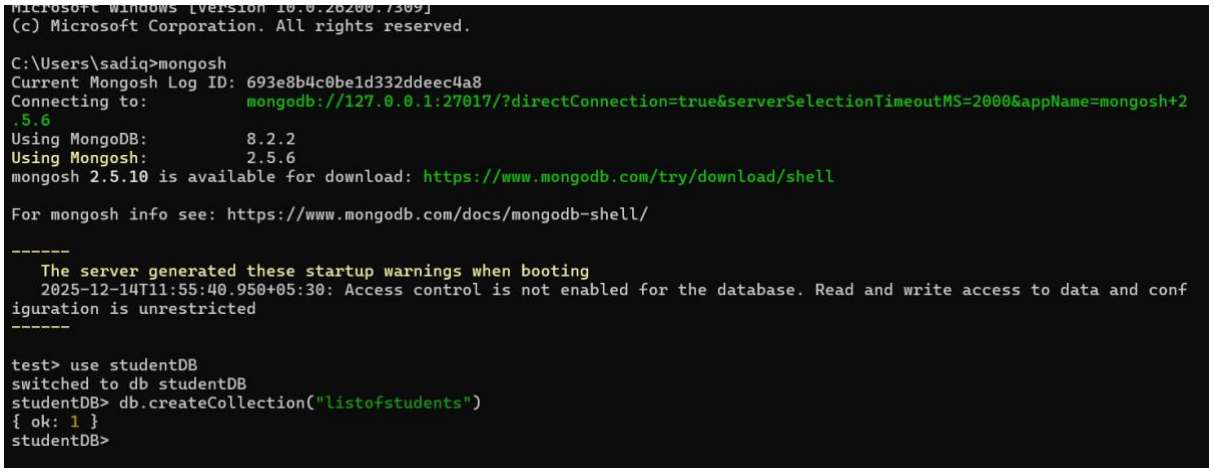
a. Create a new database: use studentDB

MongoDB will create the database when you first insert data into it.



b. Create a collection:

db.createCollection("listofstudents")



3. Insert Operations

a. InsertOne document:

db.listofstudents.insertOne({name: "Thahseen", age: 22, course: "MERN Stack",status: "enrolled"})

b. InsertMany document:

```
db.listofstudents.insertMany([
   { name: "Bob", age: 23, course: "Data Science", status: "enrolled" },
   { name: "Charlie", age: 21, course: "MERN Stack", status: "completed" },
   { name: "Diana", age: 24, course: "AI/ML", status: "enrolled" }
])
```

```
studentDB> db.listofstudents.insertMany([
...     { name: "Bob", age: 23, course: "Data Science", status: "enrolled" },
...     { name: "Charlie", age: 21, course: "MERN Stack", status: "completed" },
...     { name: "Diana", age: 24, course: "AI/ML", status: "enrolled" }
... ])
...
{
  acknowledged: true,
  insertedIds: {
    '0': ObjectId('693e90538a26397e27eec4a9'),
    '1': ObjectId('693e90538a26397e27eec4aa'),
    '2': ObjectId('693e90538a26397e27eec4ab')
  }
}
studentDB>
```

4. Read Operations

```
studentDB> db.listofstudents.find()
[
  {
    _id: ObjectId('693e8c8d0be1d332ddeec4a9'),
    name: 'Thahseen',
    age: 22,
    course: 'MERN Stack',
    status: 'enrolled'
  },
  {
    _id: ObjectId('693e90538a26397e27eec4a9'),
    name: 'Bob',
    age: 23,
    course: 'Data Science',
    status: 'enrolled'
  },
  {
    _id: ObjectId('693e90538a26397e27eec4aa'),
    name: 'Charlie',
    age: 21,
    course: 'MERN Stack',
    status: 'completed'
  },
  {
    _id: ObjectId('693e90538a26397e27eec4ab'),
    name: 'Diana',
    age: 24,
    course: 'AI/ML',
    status: 'enrolled'
  }
]
studentDB>
```

b. Fetch students in MERN Stack:

db.listofstudents.find({ course: "MERN Stack" })

```
studentDB> db.listofstudents.find({ course: "MERN Stack" })
[
  {
    _id: ObjectId('693e8c8d0be1d332ddeec4a9'),
    name: 'Thahseen',
    age: 22,
    course: 'MERN Stack',
    status: 'enrolled'
  },
  {
    _id: ObjectId('693e90538a26397e27eec4aa'),
    name: 'Charlie',
    age: 21,
    course: 'MERN Stack',
    status: 'completed'
  }
]
```

C. Pretty print:

db.listofstudents.find().pretty()

```
studentDB> db.listofstudents.find().pretty()
[
  {
    _id: ObjectId('693e8c8d0be1d332ddeec4a9'),
    name: 'Thahseen',
    age: 22,
    course: 'MERN Stack',
    status: 'enrolled'
  },
  {
    _id: ObjectId('693e90538a26397e27eec4a9'),
    name: 'Bob',
    age: 23,
    course: 'Data Science',
    status: 'enrolled'
  },
  {
    _id: ObjectId('693e90538a26397e27eec4aa'),
    name: 'Charlie',
    age: 21,
    course: 'MERN Stack',
    status: 'completed'
  },
  {
    _id: ObjectId('693e90538a26397e27eec4ab'),
    name: 'Diana',
    age: 24,
    course: 'AI/ML',
    status: 'enrolled'
  }
]
studentDB>
```

5. Update Operations

a. Update a single document:

db.listofstudents.updateOne(

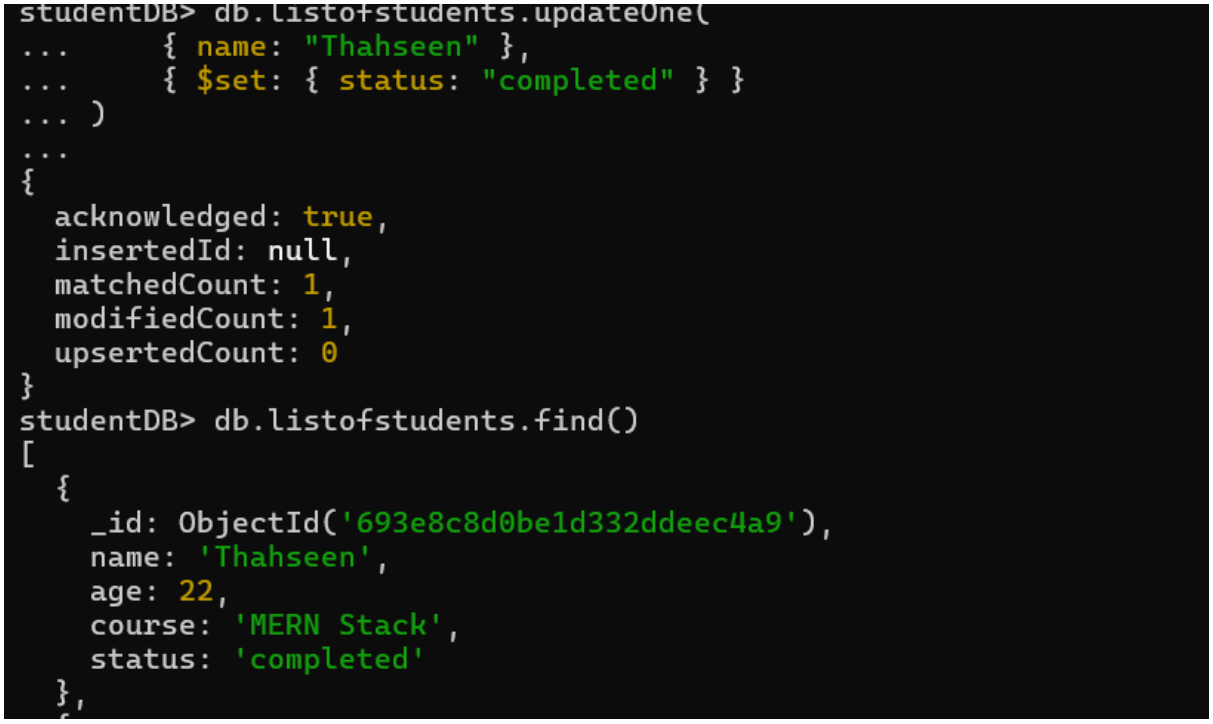
{ name: "Thahseen" },

{ $set: { status: "completed" } }

)

And

use db.listofstudents.find() To check it updated



b. Update multiple documents:

db.listofstudents.updateMany(

{ course: "MERN Stack" },

{ $set: { status: " enrolled" } }

)

And

To check whether it updated in Database use db.listofstudents.find()

```
studentDB> db.listofstudents.updateMany(
...     { course: "MERN Stack" },
...     { $set: { status: " enrolled" } }
... )
...
{
  acknowledged: true,
  insertedId: null,
  matchedCount: 2,
  modifiedCount: 2,
  upsertedCount: 0
}
studentDB> db.listofstudents.find()
[
  {
    _id: ObjectId('693e8c8d0be1d332ddeec4a9'),
    name: 'Thahseen',
    age: 22,
    course: 'MERN Stack',
    status: ' enrolled'
  },
  {
    _id: ObjectId('693e90538a26397e27eec4a9'),
    name: 'Bob',
    age: 23,
    course: 'Data Science',
    status: 'enrolled'
  },
  {
    _id: ObjectId('693e90538a26397e27eec4aa'),
    name: 'Charlie',
    age: 21,
    course: 'MERN Stack',
    status: ' enrolled'
  },
  {
    _id: ObjectId('693e90538a26397e27eec4ab'),
    name: 'Diana',
    age: 24,
    course: 'AI/ML',
    status: 'enrolled'
```

6. Delete Operations

a. Delete one document:

db.listofstudents.deleteOne({ name: "Diana" })

```
studentDB> db.listofstudents.deleteOne({ name: "Diana" })
{ acknowledged: true, deletedCount: 1 }
studentDB>
```

To check it delete in databases use db.listofstudents.find()

```
studentDB> db.listofstudents.deleteOne({ name: "Diana" })
{ acknowledged: true, deletedCount: 1 }
studentDB> db.listofstudents.find()
[
  {
    _id: ObjectId('693e8c8d0be1d332ddeec4a9'),
    name: 'Thahseen',
    age: 22,
    course: 'MERN Stack',
    status: ' enrolled'
  },
  {
    _id: ObjectId('693e90538a26397e27eec4a9'),
    name: 'Bob',
    age: 23,
    course: 'Data Science',
    status: 'enrolled'
  },
  {
    _id: ObjectId('693e90538a26397e27eec4aa'),
    name: 'Charlie',
    age: 21,
    course: 'MERN Stack',
    status: ' enrolled'
  }
]
studentDB>
```

b. Delete all documents:

db.listofstudents.deleteMany({})

```
studentDB> db.listofstudents.deleteMany({})
{ acknowledged: true, deletedCount: 3 }
studentDB>
```

AND to check it delete all data from particular collection **listofstudents** in **studentDB database. Use** db.listofstudents.find()

```
studentDB> db.listofstudents.deleteMany({})
{ acknowledged: true, deletedCount: 3 }
studentDB> db.listofstudents.find()

studentDB>
```

7. Query Operators

a. To find students using only with age (gt = greater than > 22)

db.listofstudents.find({ age: { $gt: 22 } }) // age > 22

```
studentDB> db.listofstudents.find({ age: { $gt: 22 } })
[
  {
    _id: ObjectId('693e98918a26397e27eec4ad'),
    name: 'Bob',
    age: 23,
    course: 'Data Science',
    status: 'enrolled'
  },
  {
    _id: ObjectId('693e98918a26397e27eec4af'),
    name: 'Diana',
    age: 24,
    course: 'AI/ML',
    status: 'enrolled'
  }
]
studentDB>
```

b. Fetch students enrolled in MERN Stack or AI/ML

db.listofstudents.find({ course: { $in: ["MERN Stack", "AI/ML"] } })

```
studentDB> db.listofstudents.find({
...     course: { $in: ["MERN Stack", "AI/ML"] }
... })
...
[
  {
    _id: ObjectId('693e987a8a26397e27eec4ac'),
    name: 'Thahseen',
    age: 22,
    course: 'MERN Stack',
    status: 'enrolled'
  },
  {
    _id: ObjectId('693e98918a26397e27eec4ae'),
    name: 'Charlie',
    age: 21,
    course: 'MERN Stack',
    status: 'completed'
  },
  {
    _id: ObjectId('693e98918a26397e27eec4af'),
    name: 'Diana',
    age: 24,
    course: 'AI/ML',
    status: 'enrolled'
  }
]
studentDB>
```

c. Fetch students with MERN Stack course OR completed status

```
db.listofstudents.find({
  $or: [
    { course: "MERN Stack" },
    { status: "completed" }
  ]
})
```

```
studentDB> db.listofstudents.find({
...     $or: [
...         { course: "MERN Stack" },
...         { status: "completed" }
...     ]
... })
...
[
  {
    _id: ObjectId('693e987a8a26397e27eec4ac'),
    name: 'Thahseen',
    age: 22,
    course: 'MERN Stack',
    status: 'enrolled'
  },
  {
    _id: ObjectId('693e98918a26397e27eec4ae'),
    name: 'Charlie',
    age: 21,
    course: 'MERN Stack',
    status: 'completed'
  },
  {
    _id: ObjectId('693e9bea8a26397e27eec4b0'),
    name: 'Nisma',
    age: 20,
    course: 'Data Science',
    status: 'completed'
  }
]
studentDB>
```

d. Query: Fetch students where status field exists

db.listofstudents.find({ status: { $exists: true } })

```
studentDB> db.listofstudents.find({ status: { $exists: true } })
[
  {
    _id: ObjectId('693e987a8a26397e27eec4ac'),
    name: 'Thahseen',
    age: 22,
    course: 'MERN Stack',
    status: 'enrolled'
  },
  {
    _id: ObjectId('693e98918a26397e27eec4ad'),
    name: 'Bob',
    age: 23,
    course: 'Data Science',
    status: 'enrolled'
  },
  {
    _id: ObjectId('693e98918a26397e27eec4ae'),
    name: 'Charlie',
    age: 21,
    course: 'MERN Stack',
    status: 'completed'
  },
  {
    _id: ObjectId('693e98918a26397e27eec4af'),
    name: 'Diana',
    age: 24,
    course: 'AI/ML',
    status: 'enrolled'
  },
  {
    _id: ObjectId('693e9bea8a26397e27eec4b0'),
    name: 'Nisma',
    age: 20,
    course: 'Data Science',
    status: 'completed'
  }
]
studentDB>
```

8. Use Case : Library System

Create a small use case like a Library System or E-Commerce product listing and apply insert, update, and search operations.

a . create database & collection in mongodb:

Create a new database: use libraryDB and

db.createCollection("books")

```
------
   The server generated these startup warnings when booting
   2025-12-14T11:55:40.950+05:30: Access control is not enabled for the database. Read and write access to data and conf
iguration is unrestricted
------

test> use libraryDB
switched to db libraryDB
libraryDB> db.createCollection("books")
{ ok: 1 }
libraryDB>
```

b. Insert Operations

Insert one book:

db.books.insertOne({title: "MongoDB Basics", author: "John Smith", copies: 5, available: true, genre: "Database”})

Insert multiple books:

db.books.insertMany([

{ title: "JavaScript Guide", author: "Jane Doe", copies: 3, available: true, genre: "Programming" },

{ title: "Data Science 101", author: "Alice Brown", copies: 2, available: false, genre: "Data Science" },

{ title: "AI for Beginners", author: "Bob Green", copies: 4, available: true, genre: "AI/ML" }

])

```
libraryDB> db.books.insertOne({title: "MongoDB Basics", author: "John Smith", copies: 5, available: true, genre: "Database" })
{
  acknowledged: true,
  insertedId: ObjectId('693ea09373422fa76feec4a9')
}
libraryDB> db.books.insertMany([
...     { title: "JavaScript Guide", author: "Jane Doe", copies: 3, available: true, genre: "Programming" },
...     { title: "Data Science 101", author: "Alice Brown", copies: 2, available: false, genre: "Data Science" },
...     { title: "AI for Beginners", author: "Bob Green", copies: 4, available: true, genre: "AI/ML" }
... ])
...
{
  acknowledged: true,
  insertedIds: {
    '0': ObjectId('693ea0aa73422fa76feec4aa'),
    '1': ObjectId('693ea0aa73422fa76feec4ab'),
    '2': ObjectId('693ea0aa73422fa76feec4ac')
  }
}
libraryDB>
```

c. Read/Search Operations

Fetch all books:

db.books.find()

```
libraryDB> db.books.find()
[
  {
    _id: ObjectId('693ea09373422fa76feec4a9'),
    title: 'MongoDB Basics',
    author: 'John Smith',
    copies: 5,
    available: true,
    genre: 'Database'
  },
  {
    _id: ObjectId('693ea0aa73422fa76feec4aa'),
    title: 'JavaScript Guide',
    author: 'Jane Doe',
    copies: 3,
    available: true,
    genre: 'Programming'
  },
  {
    _id: ObjectId('693ea0aa73422fa76feec4ab'),
    title: 'Data Science 101',
    author: 'Alice Brown',
    copies: 2,
    available: false,
    genre: 'Data Science'
  },
  {
    _id: ObjectId('693ea0aa73422fa76feec4ac'),
    title: 'AI for Beginners',
    author: 'Bob Green',
    copies: 4,
    available: true,
    genre: 'AI/ML'
  }
]
libraryDB>
```

Fetch books that are available:

db.books.find({ available: true })

```
libraryDB> db.books.find({ available: true })
[
  {
    _id: ObjectId('693ea09373422fa76feec4a9'),
    title: 'MongoDB Basics',
    author: 'John Smith',
    copies: 5,
    available: true,
    genre: 'Database'
  },
  {
    _id: ObjectId('693ea0aa73422fa76feec4aa'),
    title: 'JavaScript Guide',
    author: 'Jane Doe',
    copies: 3,
    available: true,
    genre: 'Programming'
  },
  {
    _id: ObjectId('693ea0aa73422fa76feec4ac'),
    title: 'AI for Beginners',
    author: 'Bob Green',
    copies: 4,
    available: true,
    genre: 'AI/ML'
  }
]
libraryDB>
```

Fetch books by author OR available:

```
db.books.find({
  $or: [
    { author: "Alice Brown" },
    { available: true }
  ]
})
```

```
libraryDB> db.books.find({
...     $or: [
...         { author: "Alice Brown" },
...         { available: true }
...     ]
... })
...
[
  {
    _id: ObjectId('693ea09373422fa76feec4a9'),
    title: 'MongoDB Basics',
    author: 'John Smith',
    copies: 5,
    available: true,
    genre: 'Database'
  },
  {
    _id: ObjectId('693ea0aa73422fa76feec4aa'),
    title: 'JavaScript Guide',
    author: 'Jane Doe',
    copies: 3,
    available: true,
    genre: 'Programming'
  },
  {
    _id: ObjectId('693ea0aa73422fa76feec4ab'),
    title: 'Data Science 101',
    author: 'Alice Brown',
    copies: 2,
    available: false,
    genre: 'Data Science'
  },
  {
    _id: ObjectId('693ea0aa73422fa76feec4ac'),
    title: 'AI for Beginners',
    author: 'Bob Green',
    copies: 4,
    available: true,
    genre: 'AI/ML'
  }
]
```

d. Update Operations

Update a single book (borrowed, so available = false):

db.books.updateOne({ title: "MongoDB Basics" }, { $set: { available: false } })

```
libraryDB> db.books.updateOne( { title: "MongoDB Basics" }, { $set: { available: false } })
{
  acknowledged: true,
  insertedId: null,
  matchedCount: 1,
  modifiedCount: 1,
  upsertedCount: 0
}
libraryDB>
```

Update multiple books (e.g., set all AI/ML books as available):

db.books.updateMany({ genre: "AI/ML" }, { $set: { available: true } })

```
libraryDB> db.books.updateMany({ genre: "AI/ML" }, { $set: { available: true } })
{
  acknowledged: true,
  insertedId: null,
  matchedCount: 1,
  modifiedCount: 0,
  upsertedCount: 0
}
libraryDB>
```

Now, check it updated databases by using **db.books.find()**

```
libraryDB> db.books.find()
[
  {
    _id: ObjectId('693ea09373422fa76feec4a9'),
    title: 'MongoDB Basics',
    author: 'John Smith',
    copies: 5,
    available: false,
    genre: 'Database'
  },
  {
    _id: ObjectId('693ea0aa73422fa76feec4aa'),
    title: 'JavaScript Guide',
    author: 'Jane Doe',
    copies: 3,
    available: true,
    genre: 'Programming'
  },
  {
    _id: ObjectId('693ea0aa73422fa76feec4ab'),
    title: 'Data Science 101',
    author: 'Alice Brown',
    copies: 2,
    available: false,
    genre: 'Data Science'
  },
  {
    _id: ObjectId('693ea0aa73422fa76feec4ac'),
    title: 'AI for Beginners',
    author: 'Bob Green',
    copies: 4,
    available: true,
    genre: 'AI/ML'
  }
]
libraryDB>
```

e. Delete Operations

Delete a single book (removed from library):

db.books.deleteOne({ title: "Data Science 101" })

```
libraryDB> db.books.deleteOne({ title: "Data Science 101" })
{ acknowledged: true, deletedCount: 1 }
libraryDB>
```

Now, check it updated databases by using **db.books.find()**

```
libraryDB> db.books.deleteOne({ title: "Data Science 101" })
{ acknowledged: true, deletedCount: 1 }
libraryDB> db.books.find()
[
  {
    _id: ObjectId('693ea09373422fa76feec4a9'),
    title: 'MongoDB Basics',
    author: 'John Smith',
    copies: 5,
    available: false,
    genre: 'Database'
  },
  {
    _id: ObjectId('693ea0aa73422fa76feec4aa'),
    title: 'JavaScript Guide',
    author: 'Jane Doe',
    copies: 3,
    available: true,
    genre: 'Programming'
  },
  {
    _id: ObjectId('693ea0aa73422fa76feec4ac'),
    title: 'AI for Beginners',
    author: 'Bob Green',
    copies: 4,
    available: true,
    genre: 'AI/ML'
  }
]
libraryDB>
```

Query Operators in Use

- $in: Select multiple genres
- $or: Fetch books by author or availability

- $exists: Check if a field exists
- $gt / $lt: For numeric filters, e.g., copies > 3

Example: Find books with more than 3 copies:

db.books.find({ copies: { $gt: 3 } })

```
libraryDB> db.books.find({ copies: { $gt: 3 } })
[
  {
    _id: ObjectId('693ea09373422fa76feec4a9'),
    title: 'MongoDB Basics',
    author: 'John Smith',
    copies: 5,
    available: false,
    genre: 'Database'
  },
  {
    _id: ObjectId('693ea0aa73422fa76feec4ac'),
    title: 'AI for Beginners',
    author: 'Bob Green',
    copies: 4,
    available: true,
    genre: 'AI/ML'
  }
]
libraryDB>
```

Conclusion

This project provided practical knowledge of MongoDB by performing CRUD operations and using query operators. It helped us understand how MongoDB stores data in a flexible, document-oriented structure.

The Library System use case showed how real-world applications manage data efficiently using MongoDB. Operations like inserting, updating, and searching records demonstrated dynamic data handling. Query operators such as $in, $or, $gt, and $exists improved data retrieval. Overall, the project highlighted MongoDB's suitability for scalable modern applications.