EXPERIMENT NO: 1

INSTALLATION OF MONGODB IN WINDOWS ENVIRONMENT

AIM:

To install MongoDB Community Server in a Windows operating system and verify the successful installation by running the MongoDB server (mongod) and MongoDB shell (mongosh).

PROCEDURE:

1. Download MongoDB

1. Open a web browser and visit the official MongoDB download page:
   https://www.mongodb.com/try/download/community

2. Select the following options:

   - Version: Latest (Recommended)
   - Platform: Windows
   - Package: MSI Installer

3. Click **Download** to get the installer.

2. Install MongoDB using MSI Installer

1. Locate the downloaded **.msi** file and double-click it.

2. In the setup wizard:

   - Click **Next**
   - Accept the license agreement
   - Choose **Complete** installation

3. Ensure the following checkbox is selected:

   - **Install MongoDB as a Windows Service**

4. Choose the default service settings (recommended).

5. Click **Install**, and wait for the installation to finish.

3. Install MongoDB Tools (Optional but Recommended)

- During installation, enable:

- **MongoDB Compass** (GUI tool)
- This provides a visual interface to view databases.

4. Verify MongoDB Installation

Using Command Prompt

- Open Command Prompt
- Run the command:
- mongod
- The message **“Waiting for connections on port 27017”** indicates MongoDB is successfully running.

5. Verify MongoDB Shell (mongosh)

- Open a new Command Prompt
- Run:
- mongosh
- If the MongoDB shell opens, the installation is successful.

6. Verify Using MongoDB Compass

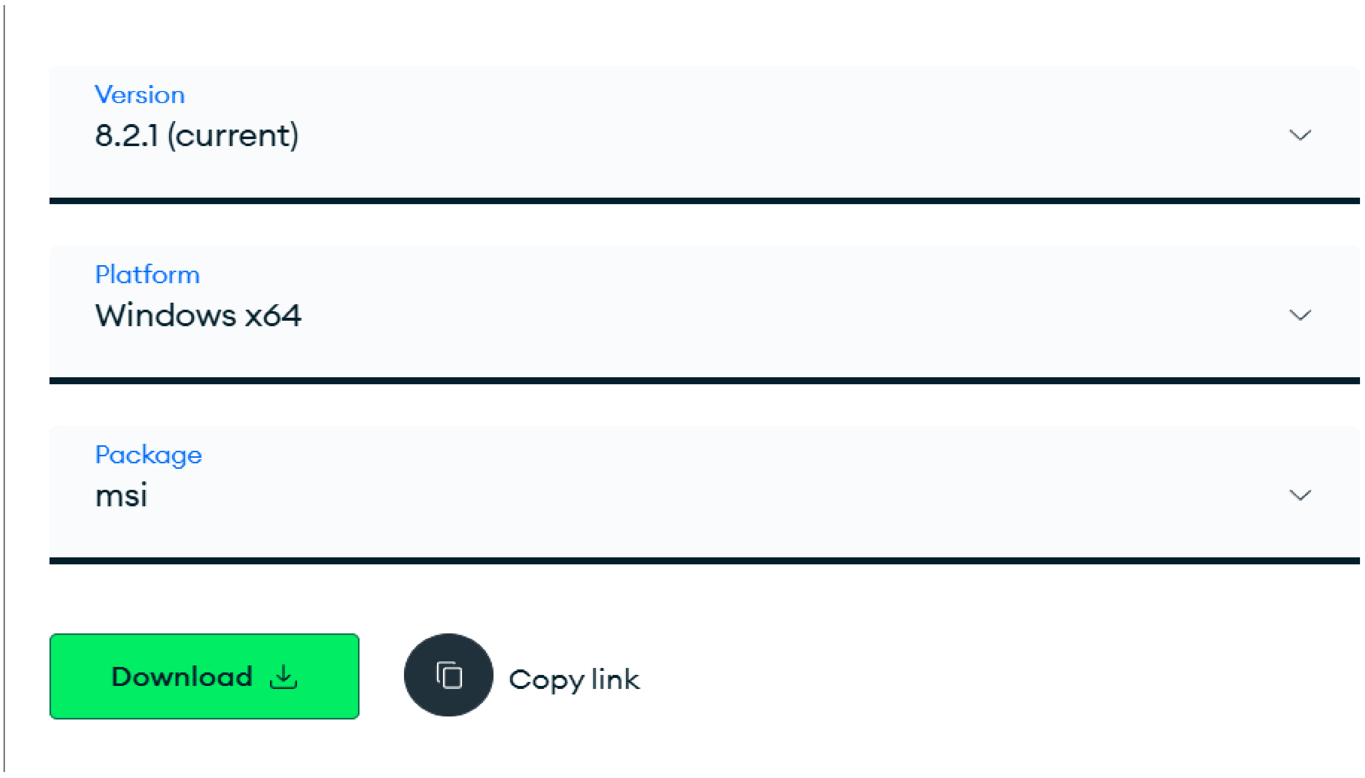
- Open MongoDB Compass
- Connect using the default URI:
- mongodb://localhost:27017
- If Compass connects successfully, MongoDB is fully functional.

RESULT:

MongoDB Community Server was successfully installed and configured in the Windows environment. The installation was verified using the **mongod**, **mongosh**, and **MongoDB Compass** tools.

SCREENSHOTS:

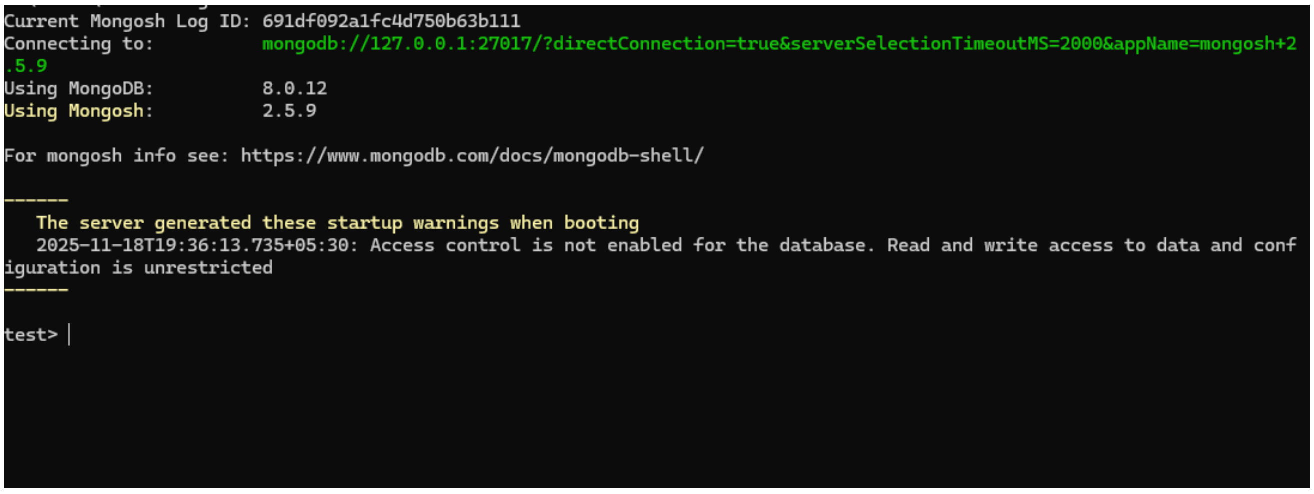
Downloading MongoDB Community Server from the official MongoDB website.


Version
8.2.1 (current)

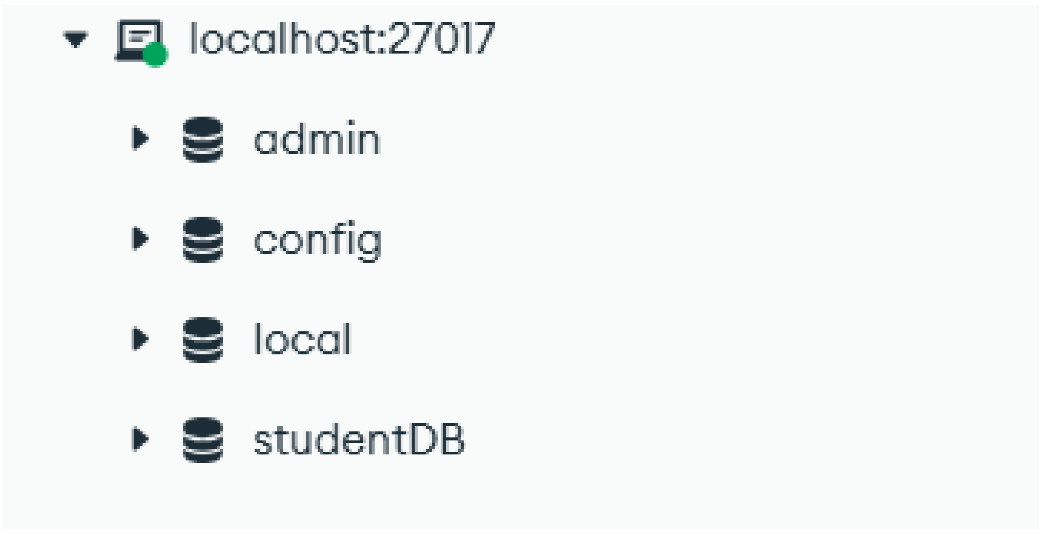
Platform
Windows x64

Package
msi

Running the MongoDB shell (mongosh) to verify client functionality.


```
Current Mongosh Log ID: 691df092a1fc4d750b63b111
Connecting to:          mongodb://127.0.0.1:27017/?directConnection=true&serverSelectionTimeoutMS=2000&appName=mongosh+2
.5.9
Using MongoDB:          8.0.12
Using Mongosh:          2.5.9

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

------
   The server generated these startup warnings when booting
   2025-11-18T19:36:13.735+05:30: Access control is not enabled for the database. Read and write access to data and conf
iguration is unrestricted
------

test> |
```

Databases inside MongoDB Compass after successful connection.



EXPERIMTENT NO: 2

INTRODUCTION TO MONGODB: BASIC DATABASE AND COLLECTION OPERATIONS

PROCEDURE:

Setting Up the MongoDB Shell

- Open your terminal or command prompt.
- Start the MongoDB shell (mongosh).

Creating the Database (student)

- Switch to the desired database named student using the use command.
- Verify the currently selected database using the db command.

Creating a Collection and Inserting Documents

- Insert a Single Document
- Insert Multiple Documents

Displaying (Querying) Documents

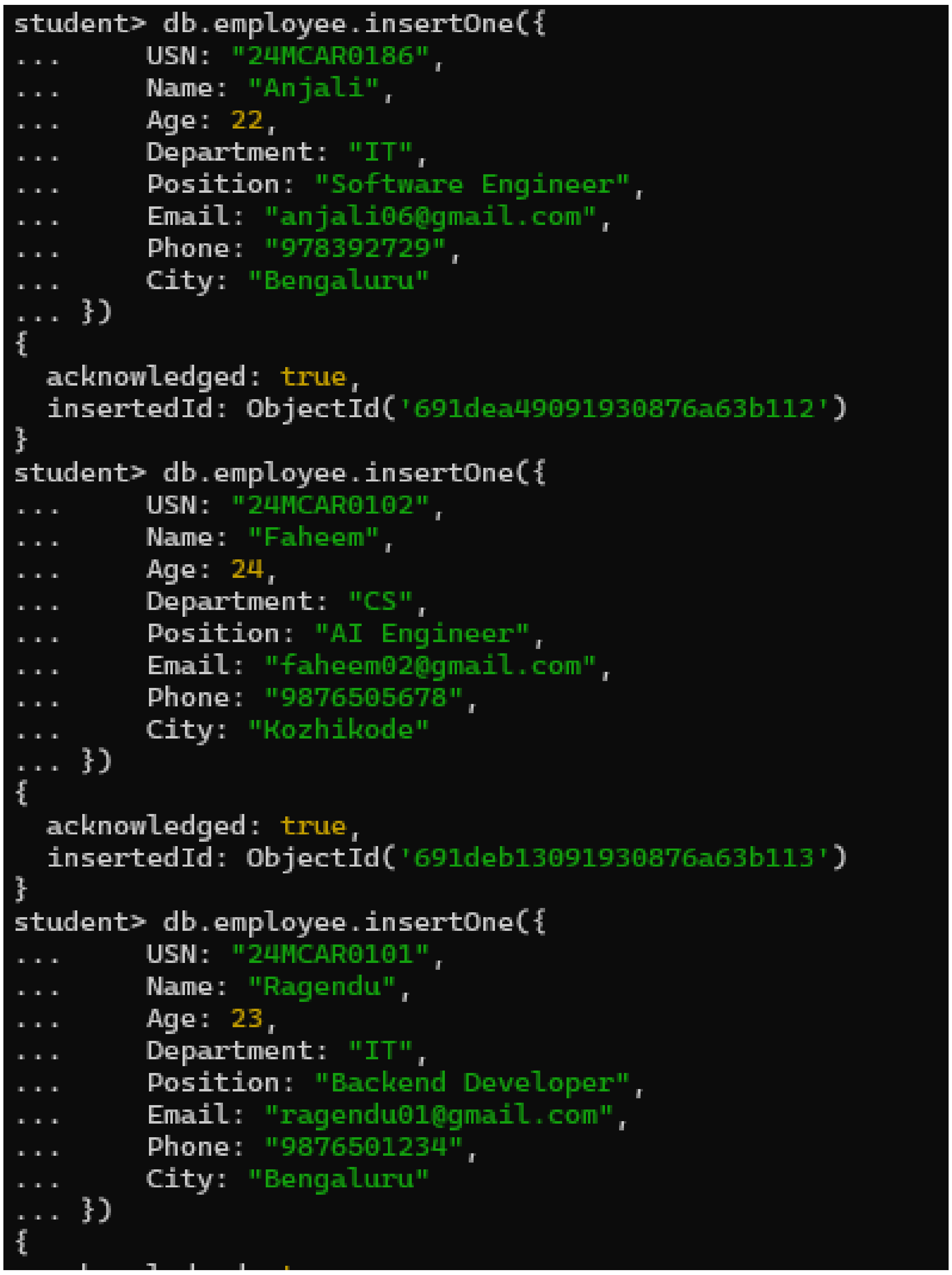
- Display All Documents
- Display All Documents in a Pretty Format
- Query for Specific Documents

Clean-Up Operations (Dropping)

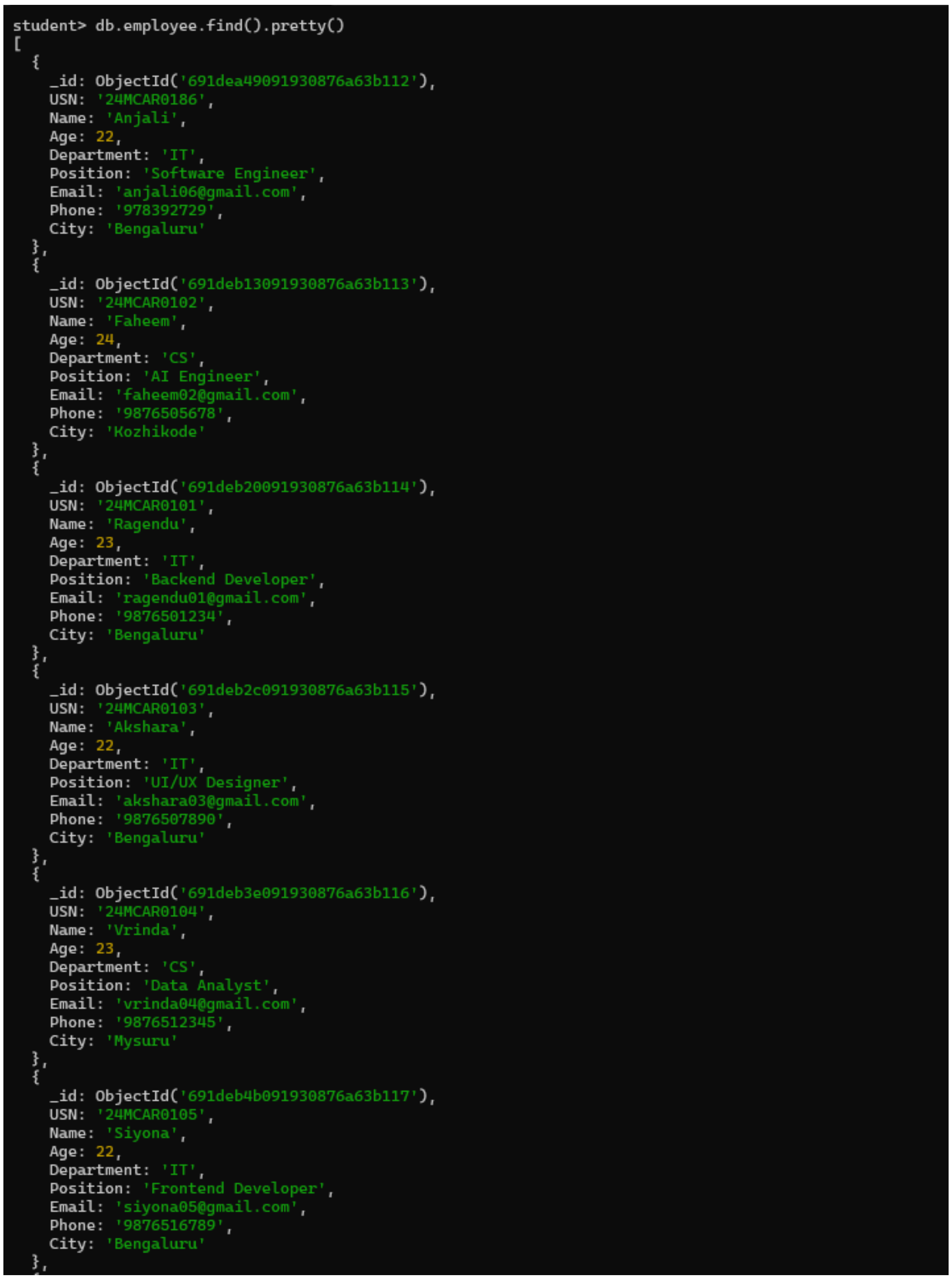
- Drop the Collection
- Drop the Database

Exit the MongoDB shell.

SCREENSHOTS:



```
student> db.employee.insertOne({
...      USN: "24MCAR0186",
...      Name: "Anjali",
...      Age: 22,
...      Department: "IT",
...      Position: "Software Engineer",
...      Email: "anjali06@gmail.com",
...      Phone: "978392729",
...      City: "Bengaluru"
... })
{
  acknowledged: true,
  insertedId: ObjectId('691dea49091930876a63b112')
}
student> db.employee.insertOne({
...      USN: "24MCAR0102",
...      Name: "Faheem",
...      Age: 24,
...      Department: "CS",
...      Position: "AI Engineer",
...      Email: "faheem02@gmail.com",
...      Phone: "9876505678",
...      City: "Kozhikode"
... })
{
  acknowledged: true,
  insertedId: ObjectId('691deb13091930876a63b113')
}
student> db.employee.insertOne({
...      USN: "24MCAR0101",
...      Name: "Ragendu",
...      Age: 23,
...      Department: "IT",
...      Position: "Backend Developer",
...      Email: "ragendu01@gmail.com",
...      Phone: "9876501234",
...      City: "Bengaluru"
... })
{
```



```
student> db.employee.find().pretty()
[
  {
    _id: ObjectId('691dea49091930876a63b112'),
    USN: '24MCAR0186',
    Name: 'Anjali',
    Age: 22,
    Department: 'IT',
    Position: 'Software Engineer',
    Email: 'anjali06@gmail.com',
    Phone: '978392729',
    City: 'Bengaluru'
  },
  {
    _id: ObjectId('691deb13091930876a63b113'),
    USN: '24MCAR0102',
    Name: 'Faheem',
    Age: 24,
    Department: 'CS',
    Position: 'AI Engineer',
    Email: 'faheem02@gmail.com',
    Phone: '9876505678',
    City: 'Kozhikode'
  },
  {
    _id: ObjectId('691deb20091930876a63b114'),
    USN: '24MCAR0101',
    Name: 'Ragendu',
    Age: 23,
    Department: 'IT',
    Position: 'Backend Developer',
    Email: 'ragendu01@gmail.com',
    Phone: '9876501234',
    City: 'Bengaluru'
  },
  {
    _id: ObjectId('691deb2c091930876a63b115'),
    USN: '24MCAR0103',
    Name: 'Akshara',
    Age: 22,
    Department: 'IT',
    Position: 'UI/UX Designer',
    Email: 'akshara03@gmail.com',
    Phone: '9876507890',
    City: 'Bengaluru'
  },
  {
    _id: ObjectId('691deb3e091930876a63b116'),
    USN: '24MCAR0104',
    Name: 'Vrinda',
    Age: 23,
    Department: 'CS',
    Position: 'Data Analyst',
    Email: 'vrinda04@gmail.com',
    Phone: '9876512345',
    City: 'Mysuru'
  },
  {
    _id: ObjectId('691deb4b091930876a63b117'),
    USN: '24MCAR0105',
    Name: 'Siyona',
    Age: 22,
    Department: 'IT',
    Position: 'Frontend Developer',
    Email: 'siyona05@gmail.com',
    Phone: '9876516789',
    City: 'Bengaluru'
  },
```

EXPERIMENT NO:3

CRUD OPERATIONS IN MONGODB

AIM

To perform basic CRUD (Create, Read, Update, Delete) operations on MongoDB collections. We'll use the student database and students collection from the previous experiments.

PROCEDURE:

1. Setup and Create (C)

- create the database by executing use student.
- Insert **single document**.
- Insert **multiple documents**

2. Read (R)

- Display **all documents**
- Display all documents in a **pretty (formatted) format**
- Query for **specific documents**
- Retrieve and display only the **first document** that matches a specific criterion

3. Update (U)

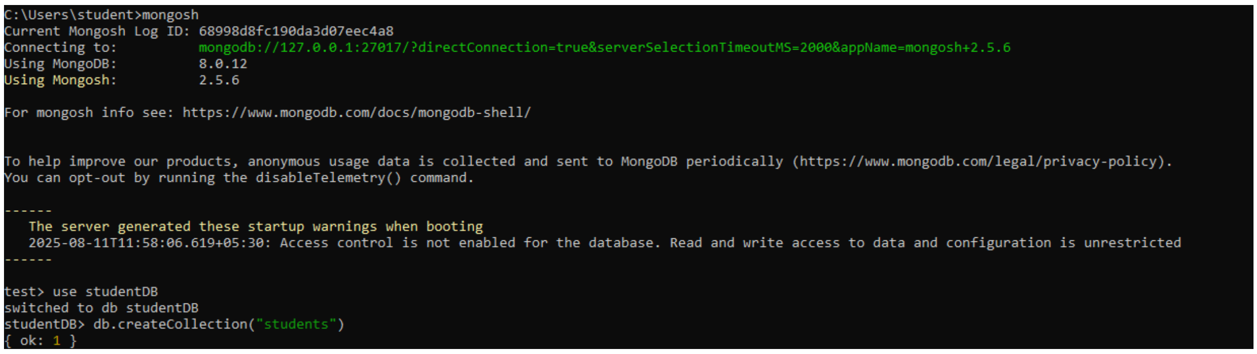
- Update a single document

4. Delete (D)

- Delete a single document
- Delete multiple documents

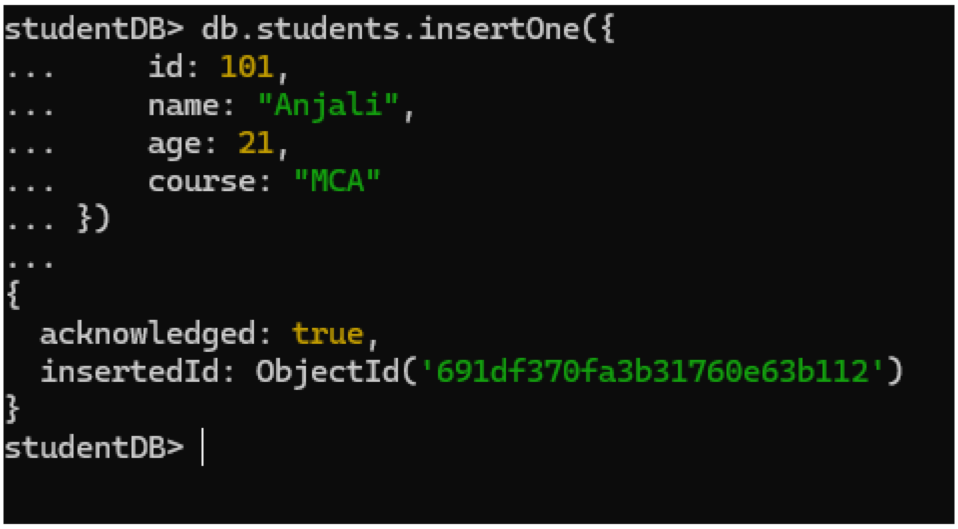
SCREEN SHOTS:

Create the database student.



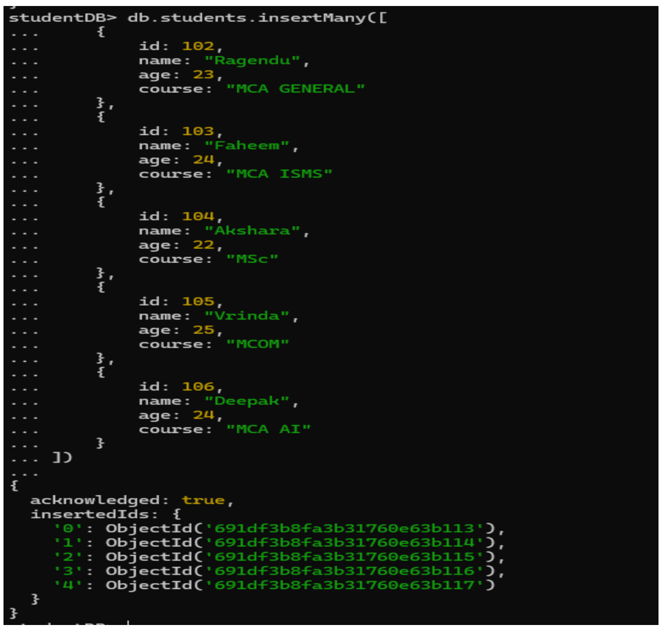
```
C:\Users\student>mongosh
Current Mongosh Log ID: 68998d8fc190da3d07eec4a8
Connecting to:          mongodb://127.0.0.1:27017/?directConnection=true&serverSelectionTimeoutMS=2000&appName=mongosh+2.5.6
Using MongoDB:          8.0.12
Using Mongosh:          2.5.6

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


To help improve our products, anonymous usage data is collected and sent to MongoDB periodically (https://www.mongodb.com/legal/privacy-policy).
You can opt-out by running the disableTelemetry() command.

------
   The server generated these startup warnings when booting
   2025-08-11T11:58:06.619+05:30: Access control is not enabled for the database. Read and write access to data and configuration is unrestricted
------

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

Insert single document.



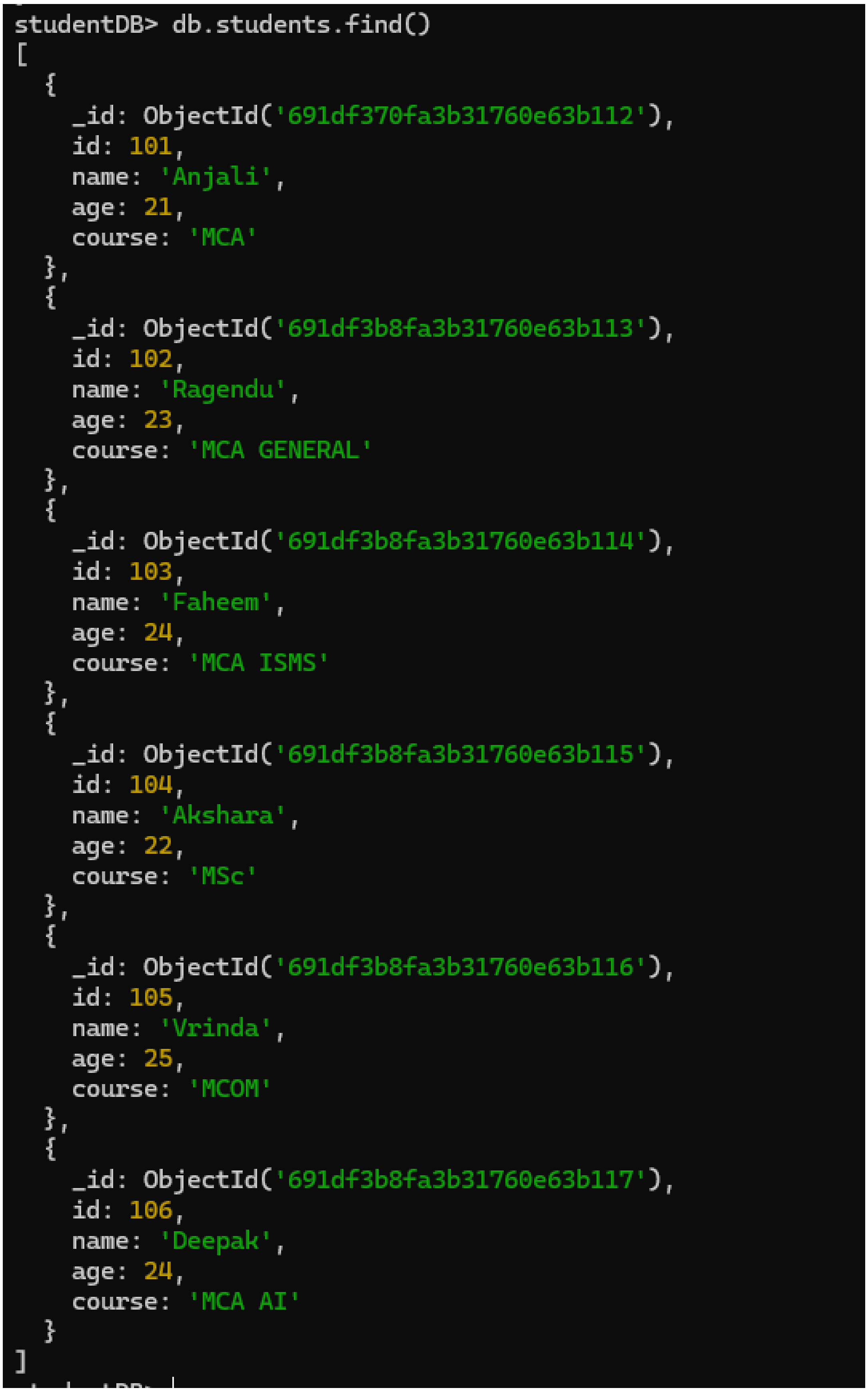
```
studentDB> db.students.insertOne({
...      id: 101,
...      name: "Anjali",
...      age: 21,
...      course: "MCA"
... })
...
{
  acknowledged: true,
  insertedId: ObjectId('691df370fa3b31760e63b112')
}
studentDB> |
```

Insert multiple documents



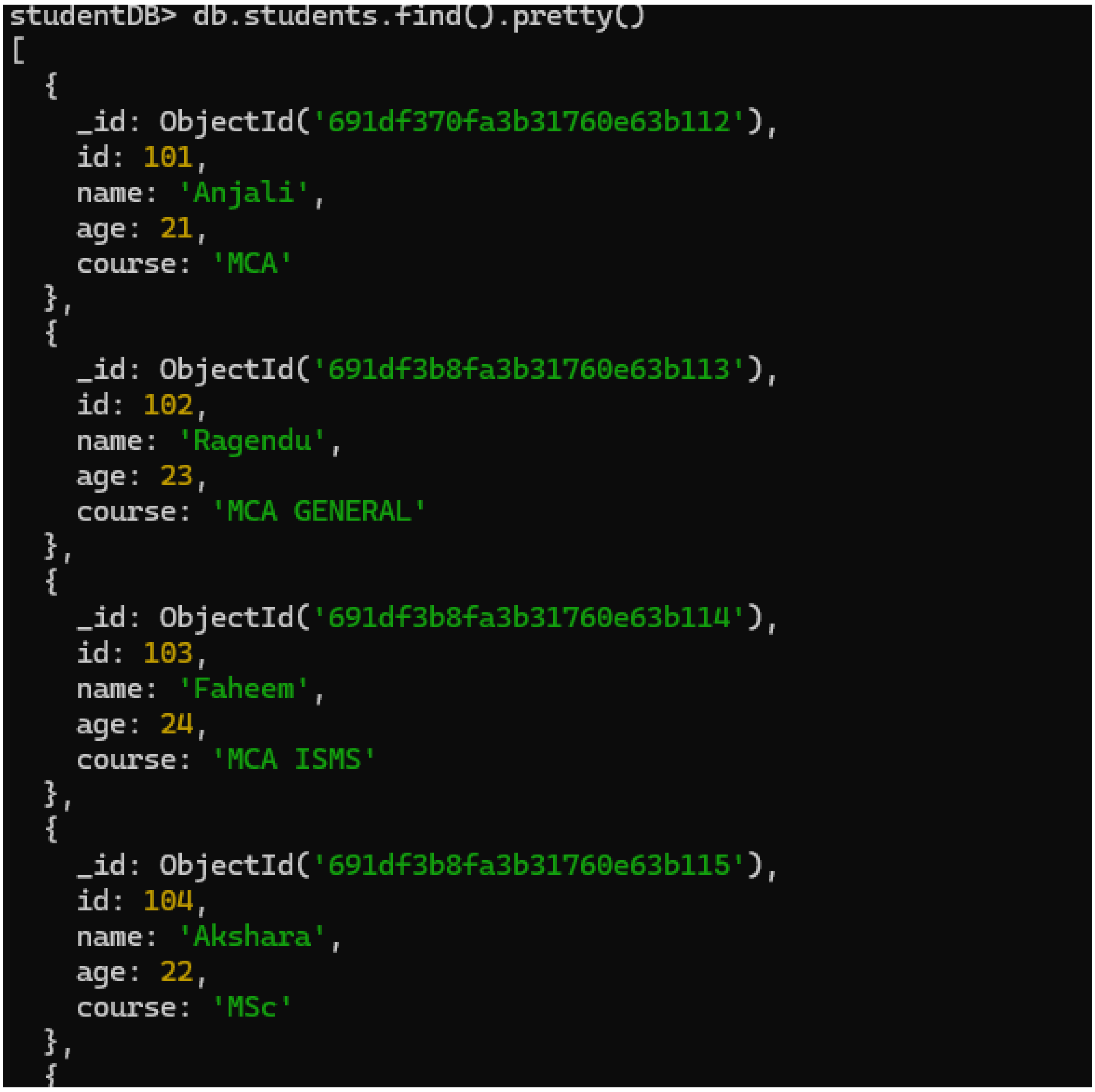
```
studentDB> db.students.insertMany([
...         {
...                 id: 102,
...                 name: "Ragendu",
...                 age: 23,
...                 course: "MCA GENERAL"
...         },
...         {
...                 id: 103,
...                 name: "Faheem",
...                 age: 24,
...                 course: "MCA ISMS"
...         },
...         {
...                 id: 104,
...                 name: "Akshara",
...                 age: 22,
...                 course: "MSc"
...         },
...         {
...                 id: 105,
...                 name: "Vrinda",
...                 age: 25,
...                 course: "MCOM"
...         },
...         {
...                 id: 106,
...                 name: "Deepak",
...                 age: 24,
...                 course: "MCA AI"
...         }
... ])
...
{
  acknowledged: true,
  insertedIds: {
    '0': ObjectId('691df3b8fa3b31760e63b113'),
    '1': ObjectId('691df3b8fa3b31760e63b114'),
    '2': ObjectId('691df3b8fa3b31760e63b115'),
    '3': ObjectId('691df3b8fa3b31760e63b116'),
    '4': ObjectId('691df3b8fa3b31760e63b117')
  }
}
```

Display all documents.



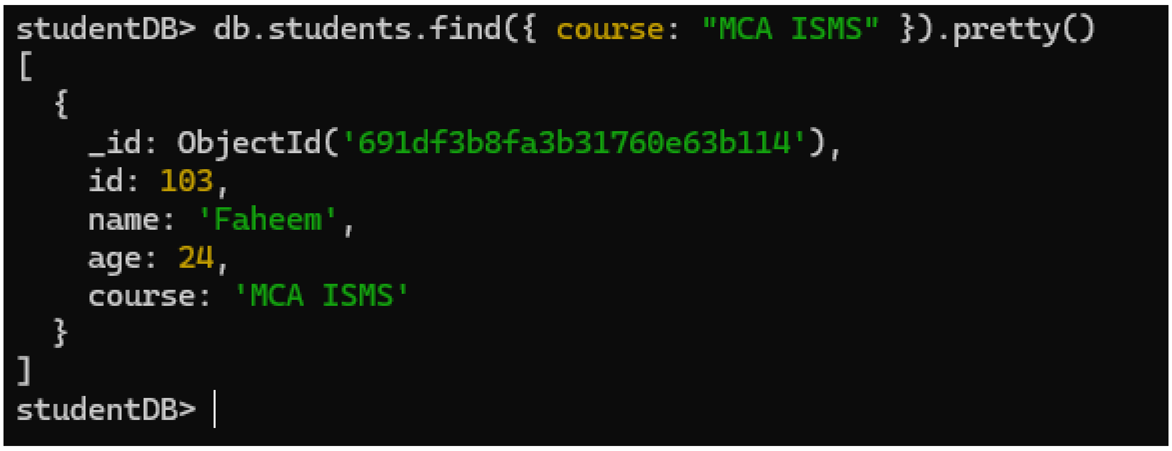
```
studentDB> db.students.find()
[
  {
    _id: ObjectId('691df370fa3b31760e63b112'),
    id: 101,
    name: 'Anjali',
    age: 21,
    course: 'MCA'
  },
  {
    _id: ObjectId('691df3b8fa3b31760e63b113'),
    id: 102,
    name: 'Ragendu',
    age: 23,
    course: 'MCA GENERAL'
  },
  {
    _id: ObjectId('691df3b8fa3b31760e63b114'),
    id: 103,
    name: 'Faheem',
    age: 24,
    course: 'MCA ISMS'
  },
  {
    _id: ObjectId('691df3b8fa3b31760e63b115'),
    id: 104,
    name: 'Akshara',
    age: 22,
    course: 'MSc'
  },
  {
    _id: ObjectId('691df3b8fa3b31760e63b116'),
    id: 105,
    name: 'Vrinda',
    age: 25,
    course: 'MCOM'
  },
  {
    _id: ObjectId('691df3b8fa3b31760e63b117'),
    id: 106,
    name: 'Deepak',
    age: 24,
    course: 'MCA AI'
  }
]
```

Display all documents in a pretty (formatted) format.



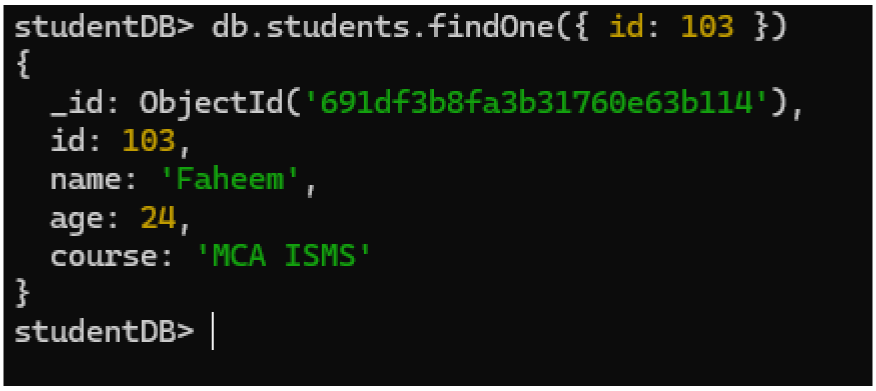
```
studentDB> db.students.find().pretty()
[
  {
    _id: ObjectId('691df370fa3b31760e63b112'),
    id: 101,
    name: 'Anjali',
    age: 21,
    course: 'MCA'
  },
  {
    _id: ObjectId('691df3b8fa3b31760e63b113'),
    id: 102,
    name: 'Ragendu',
    age: 23,
    course: 'MCA GENERAL'
  },
  {
    _id: ObjectId('691df3b8fa3b31760e63b114'),
    id: 103,
    name: 'Faheem',
    age: 24,
    course: 'MCA ISMS'
  },
  {
    _id: ObjectId('691df3b8fa3b31760e63b115'),
    id: 104,
    name: 'Akshara',
    age: 22,
    course: 'MSc'
  },
  {
```

Query for specific documents



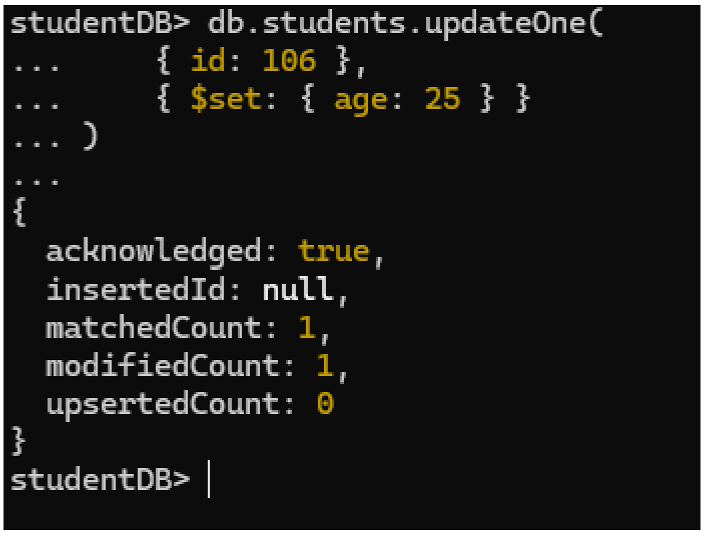
```
studentDB> db.students.find({ course: "MCA ISMS" }).pretty()
[
  {
    _id: ObjectId('691df3b8fa3b31760e63b114'),
    id: 103,
    name: 'Faheem',
    age: 24,
    course: 'MCA ISMS'
  }
]
studentDB> |
```

Retrieve and display only the first document



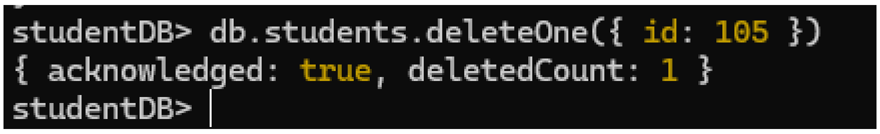
```
studentDB> db.students.findOne({ id: 103 })
{
  _id: ObjectId('691df3b8fa3b31760e63b114'),
  id: 103,
  name: 'Faheem',
  age: 24,
  course: 'MCA ISMS'
}
studentDB> |
```

Update a single document



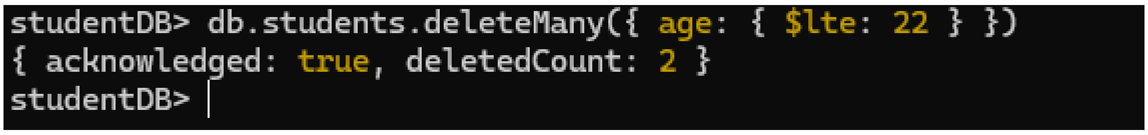
```
studentDB> db.students.updateOne(
...     { id: 106 },
...     { $set: { age: 25 } }
... )
...
{
  acknowledged: true,
  insertedId: null,
  matchedCount: 1,
  modifiedCount: 1,
  upsertedCount: 0
}
studentDB> |
```

Delete a single document



```
studentDB> db.students.deleteOne({ id: 105 })
{ acknowledged: true, deletedCount: 1 }
studentDB> |
```

Delete multiple documents



```
studentDB> db.students.deleteMany({ age: { $lte: 22 } })
{ acknowledged: true, deletedCount: 2 }
studentDB> |
```

EXPERIMENT NO: 4

CRUD OPERATIONS (UPDATE AND DELETE)

AIM:

To perform basic Update and Delete operations on a student collection in a student database. This experiment will demonstrate updateOne, updateMany, deleteOne, and deleteMany.

PROCEDURE:

Setup and Create

- Open the MongoDB shell (mongosh).
- Create the database movies.
- Insert multiple documents into the films collection using insertOne and insertMany.

Update

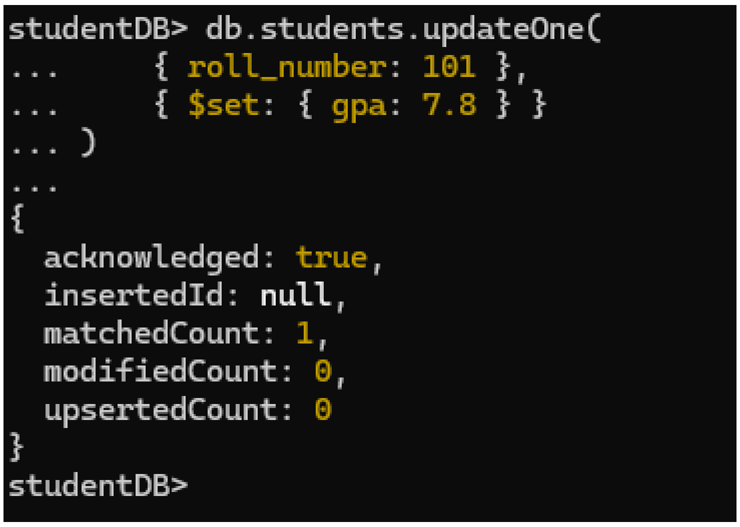
- Update a single document
- Update multiple documents (and array)

Delete

- Delete a single document
- Delete multiple documents

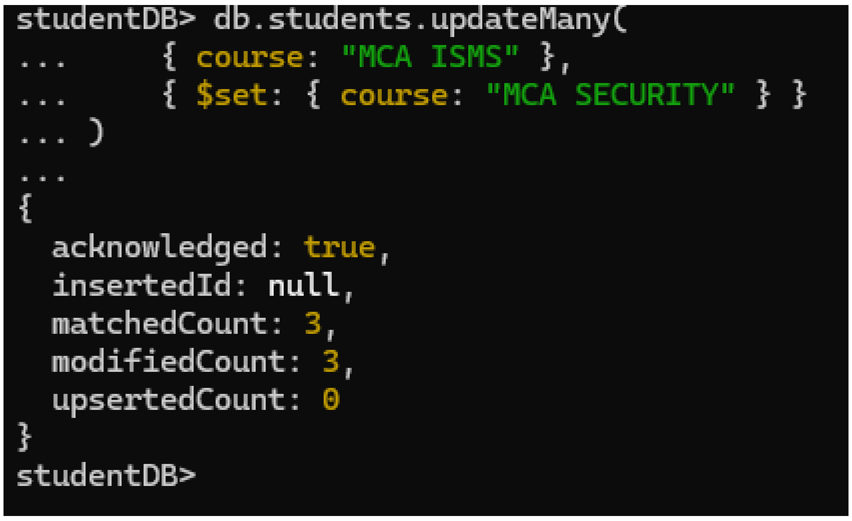
SCREENSHOTS

Update a single document



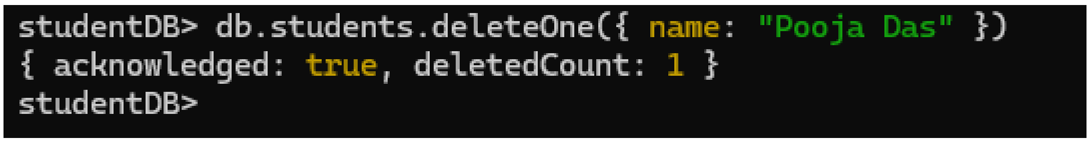
```
studentDB> db.students.updateOne(
...     { roll_number: 101 },
...     { $set: { gpa: 7.8 } }
... )
...
{
  acknowledged: true,
  insertedId: null,
  matchedCount: 1,
  modifiedCount: 0,
  upsertedCount: 0
}
studentDB>
```

Update multiple documents



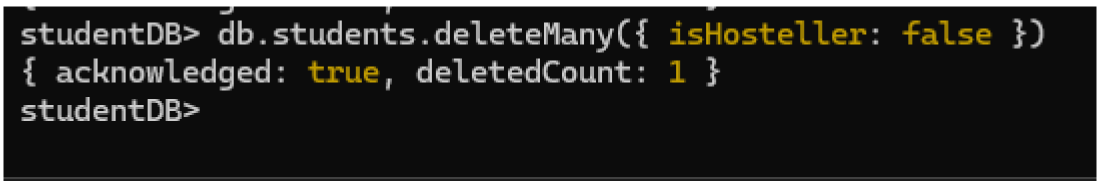
```
studentDB> db.students.updateMany(
...     { course: "MCA ISMS" },
...     { $set: { course: "MCA SECURITY" } }
... )
...
{
  acknowledged: true,
  insertedId: null,
  matchedCount: 3,
  modifiedCount: 3,
  upsertedCount: 0
}
studentDB>
```

Delete a single document



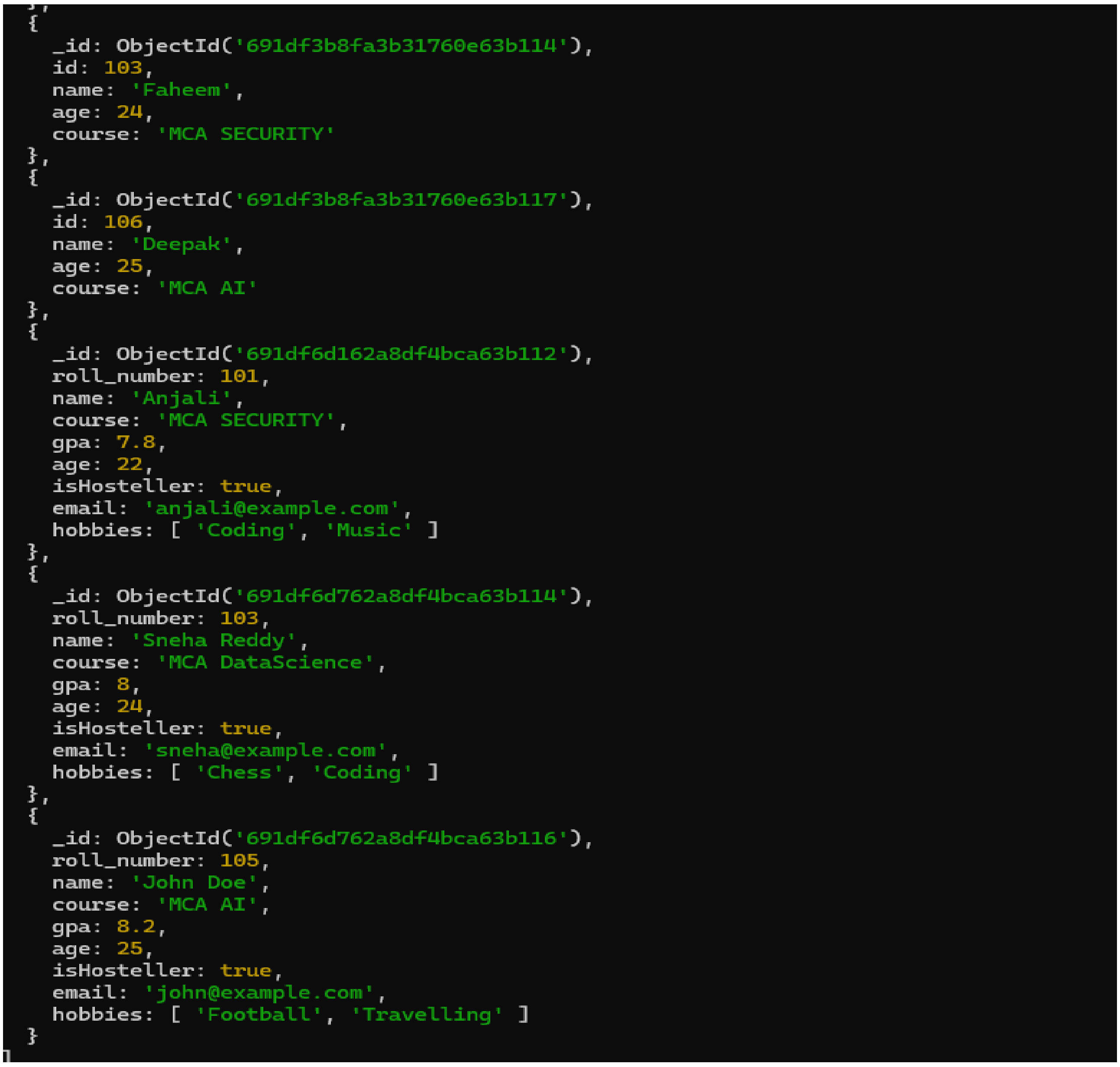
```
studentDB> db.students.deleteOne({ name: "Pooja Das" })
{ acknowledged: true, deletedCount: 1 }
studentDB>
```

Delete multiple document



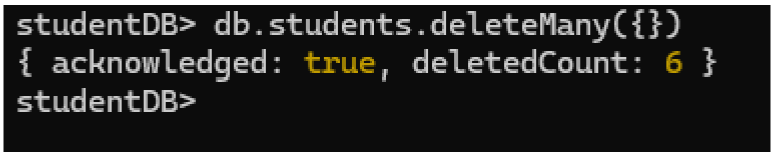
```
studentDB> db.students.deleteMany({ isHosteller: false })
{ acknowledged: true, deletedCount: 1 }
studentDB>
```

Display the remaining documents



```
  {
    _id: ObjectId('691df3b8fa3b31760e63b114'),
    id: 103,
    name: 'Faheem',
    age: 24,
    course: 'MCA SECURITY'
  },
  {
    _id: ObjectId('691df3b8fa3b31760e63b117'),
    id: 106,
    name: 'Deepak',
    age: 25,
    course: 'MCA AI'
  },
  {
    _id: ObjectId('691df6d162a8df4bca63b112'),
    roll_number: 101,
    name: 'Anjali',
    course: 'MCA SECURITY',
    gpa: 7.8,
    age: 22,
    isHosteller: true,
    email: 'anjali@example.com',
    hobbies: [ 'Coding', 'Music' ]
  },
  {
    _id: ObjectId('691df6d762a8df4bca63b114'),
    roll_number: 103,
    name: 'Sneha Reddy',
    course: 'MCA DataScience',
    gpa: 8,
    age: 24,
    isHosteller: true,
    email: 'sneha@example.com',
    hobbies: [ 'Chess', 'Coding' ]
  },
  {
    _id: ObjectId('691df6d762a8df4bca63b116'),
    roll_number: 105,
    name: 'John Doe',
    course: 'MCA AI',
    gpa: 8.2,
    age: 25,
    isHosteller: true,
    email: 'john@example.com',
    hobbies: [ 'Football', 'Travelling' ]
  }
]
```

Delete All Students



```
studentDB> db.students.deleteMany({})
{ acknowledged: true, deletedCount: 6 }
studentDB>
```

EXPERIMENT NO: 5

AIM:

To perform CRUD operations on five separate collections within a library database

PROCEDURE

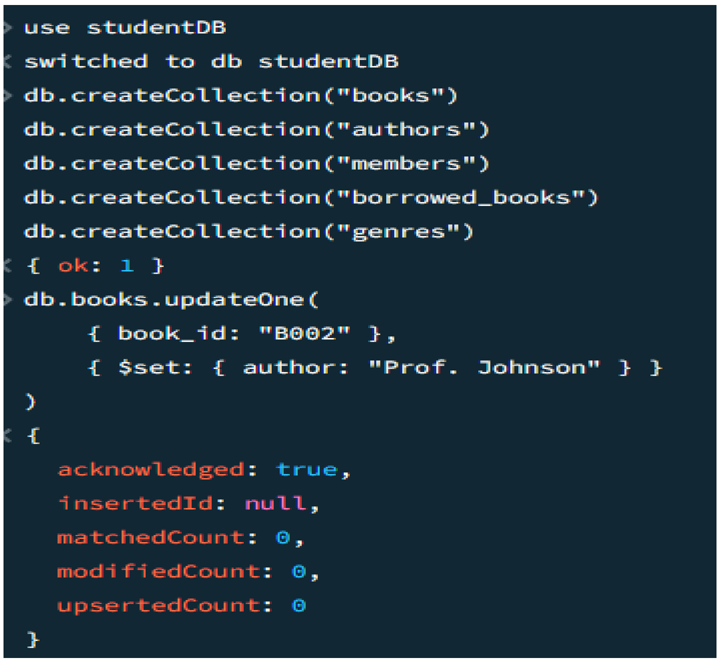
Setup and Create Collections

- Open the MongoDB shell (mongosh).
- Switch to or create the target database by executing use library_management.
- Ensure the necessary collections (books, authors, members, borrowed_books, genres) are populated with initial data (assumed to be done prior to the provided code).

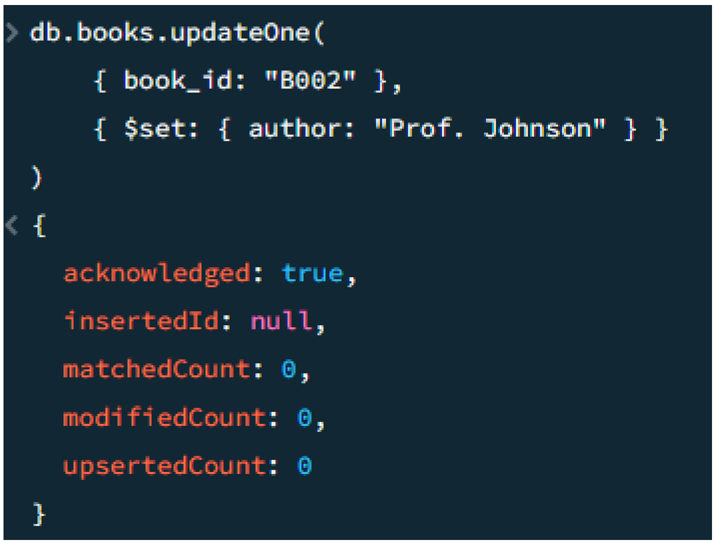
Update and Delete

- **Update the books collection**
- **Update the borrowed_books collection**
- **Update Member Status**
- **Delete Genre**

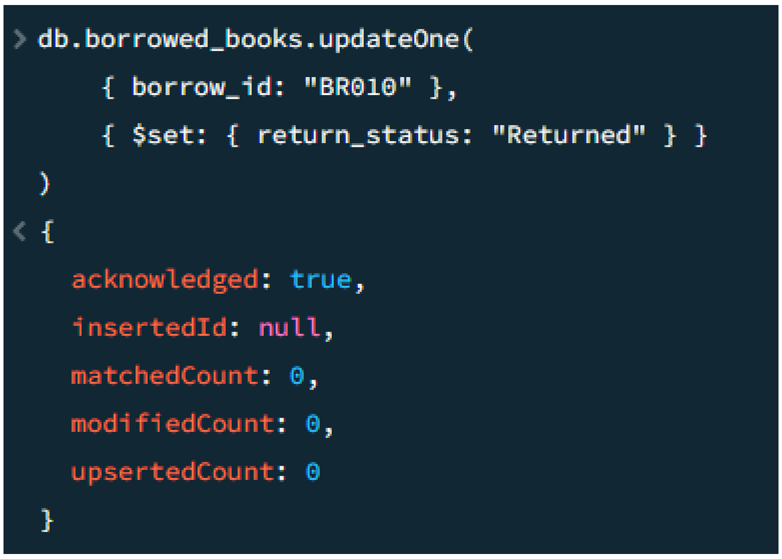
SCREENSHOTS:



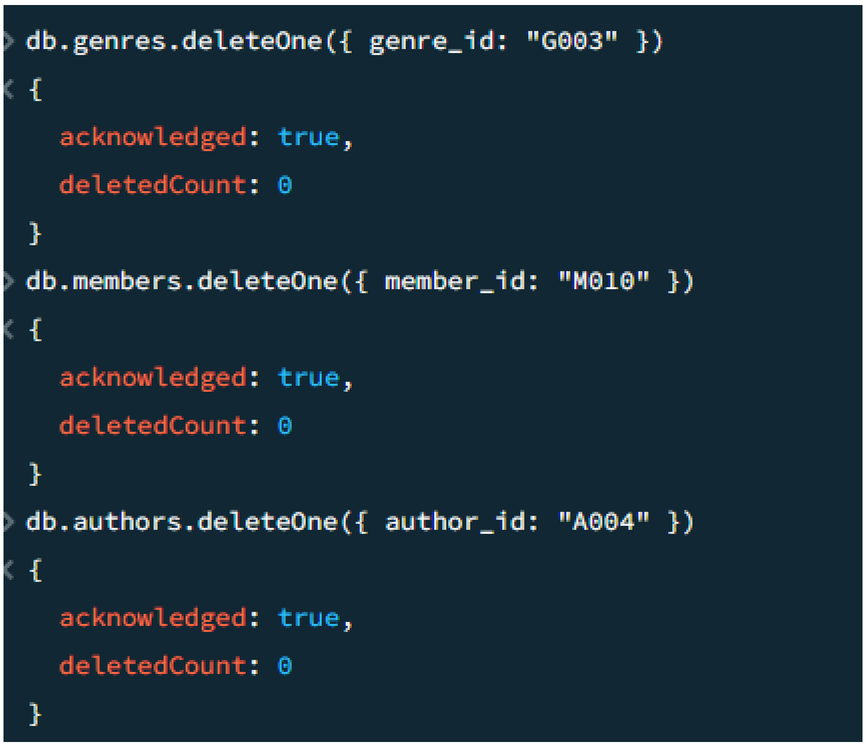
```
use studentDB
switched to db studentDB
db.createCollection("books")
db.createCollection("authors")
db.createCollection("members")
db.createCollection("borrowed_books")
db.createCollection("genres")
{ ok: 1 }
db.books.updateOne(
    { book_id: "B002" },
    { $set: { author: "Prof. Johnson" } }
)
{
  acknowledged: true,
  insertedId: null,
  matchedCount: 0,
  modifiedCount: 0,
  upsertedCount: 0
}
```



```
> db.books.updateOne(
      { book_id: "B002" },
      { $set: { author: "Prof. Johnson" } }
  )
< {
    acknowledged: true,
    insertedId: null,
    matchedCount: 0,
    modifiedCount: 0,
    upsertedCount: 0
  }
```



```
> db.borrowed_books.updateOne(
      { borrow_id: "BR010" },
      { $set: { return_status: "Returned" } }
  )
< {
    acknowledged: true,
    insertedId: null,
    matchedCount: 0,
    modifiedCount: 0,
    upsertedCount: 0
  }
```



```
> db.genres.deleteOne({ genre_id: "G003" })
< {
    acknowledged: true,
    deletedCount: 0
  }
> db.members.deleteOne({ member_id: "M010" })
< {
    acknowledged: true,
    deletedCount: 0
  }
> db.authors.deleteOne({ author_id: "A004" })
< {
    acknowledged: true,
    deletedCount: 0
  }
```

EXPERIMENT NO: 6

AIM:

To update and delete the document in mongodb by using update one and updateMany any delete one and deleteMany.

PROCEDURE:

Setup and Switch Database

- Open the MongoDB shell
- Create the database

Update Operations

- Update (updateOne)
- Batch Update (updateMany)

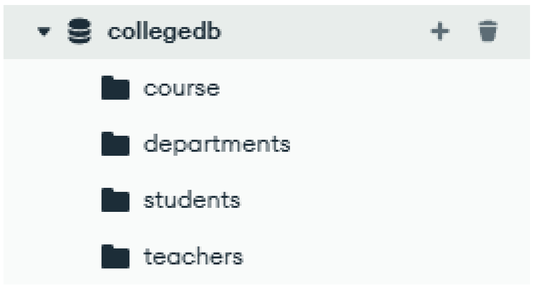
3. Delete Operations on payments

- Delete a single document
- Conditional Batch Delete.

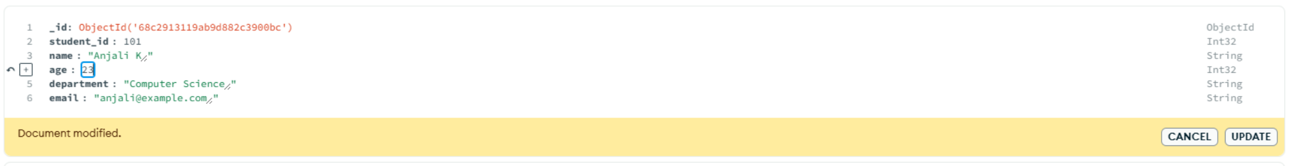
.

SCREENSHOT:

Setup and Create collection



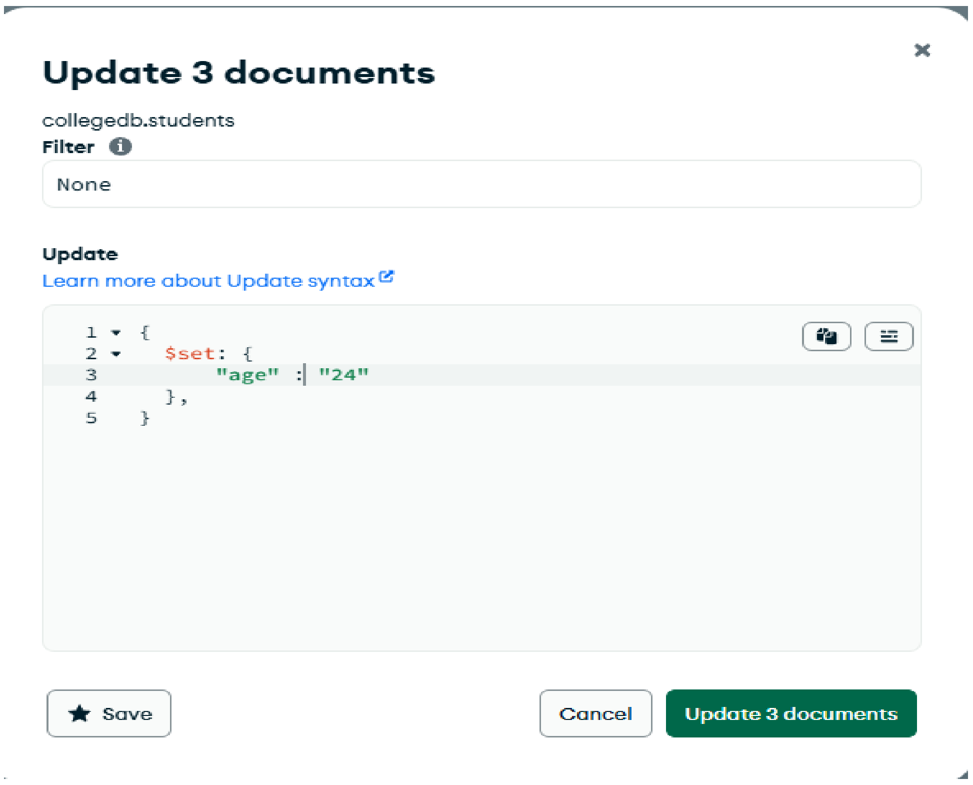
Update (updateOne)



```
_id: ObjectId('68c2913119ab9d882c3900bc')        ObjectId
student_id : 101                                  Int32
name : "Anjali K"                                 String
age : 23                                          Int32
department : "Computer Science"                   String
email : "anjali@example.com"                      String
```

Document modified.

Batch Update (updateMany)



Update 3 documents

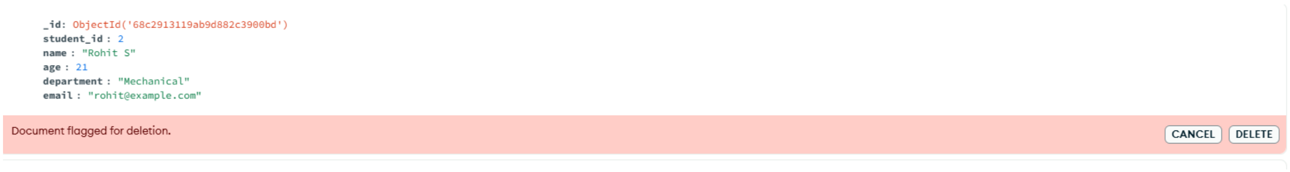
collegedb.students

Filter: None

Update

```
{
   $set: {
       "age" : "24"
   },
}
```

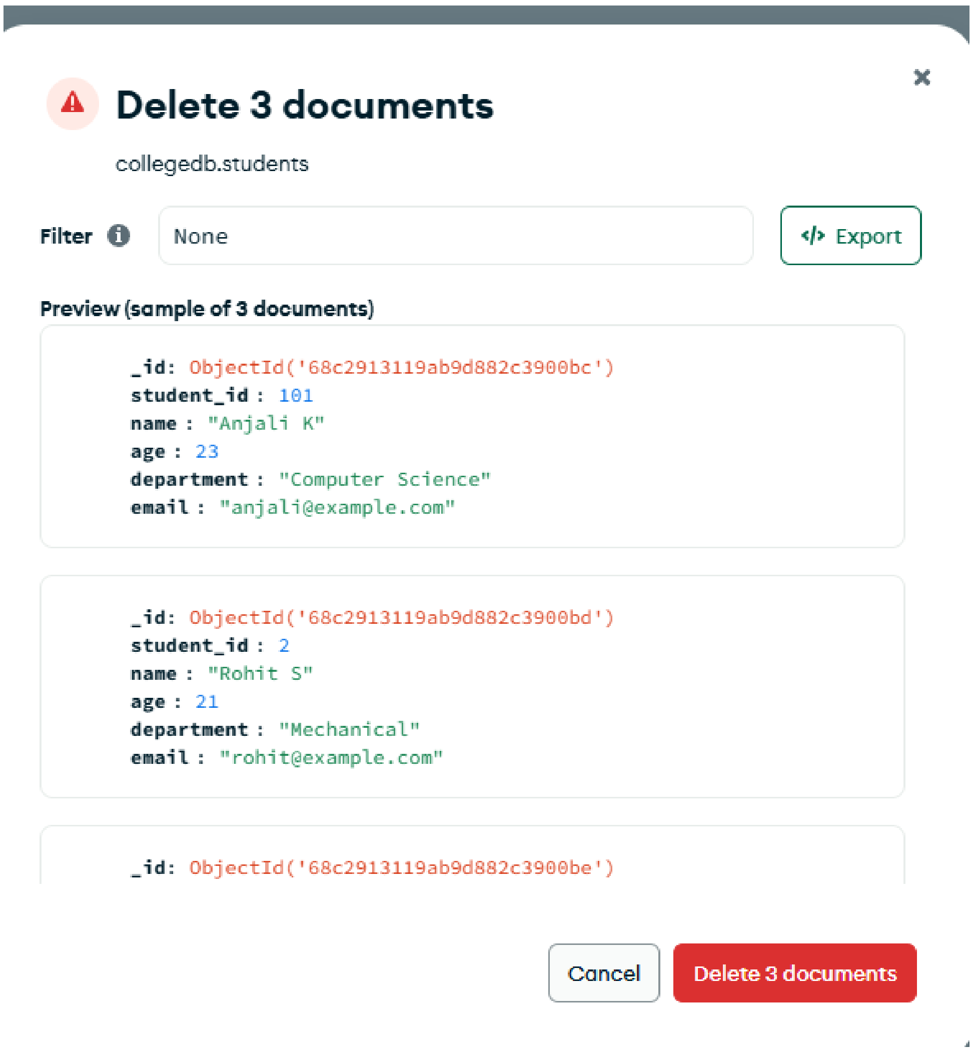
Delete a single document



```
_id: ObjectId('68c2913119ab9d882c3900bd')
student_id : 2
name : "Rohit S"
age : 21
department : "Mechanical"
email : "rohit@example.com"
```

Document flagged for deletion.

Delete a Multiple document



Delete 3 documents

collegedb.students

Filter: None

Preview (sample of 3 documents)

```
_id: ObjectId('68c2913119ab9d882c3900bc')
student_id : 101
name : "Anjali K"
age : 23
department : "Computer Science"
email : "anjali@example.com"
```

```
_id: ObjectId('68c2913119ab9d882c3900bd')
student_id : 2
name : "Rohit S"
age : 21
department : "Mechanical"
email : "rohit@example.com"
```

```
_id: ObjectId('68c2913119ab9d882c3900be')
```

EXPERIMENT NO 7:

AIM

This experiment will demonstrate how to perform data aggregation operations on a new application-based JSON dataset using MongoDB Compass. The focus will be on the $group, $sum, $avg, and $count stages.

PROCEDURE

Setup and Data Insertion:

Create the sales_analysis database. Create two collections: products and sales. Insert sample JSON data into both collections via the **Insert Document** view in Compass.

Run Aggregation: Total Sales and Count:

Navigate to the **Aggregations** tab in the sales collection.

- **Stage 1 ($group):** Group by product_id. Use **$sum** for total_quantity_sold and **$count** for number_of_sales.

Run Aggregation: Average Price:

Create a new pipeline.

- **Stage 1 ($group):** Group by product_id. Use **$avg** to calculate the average_sale_price.

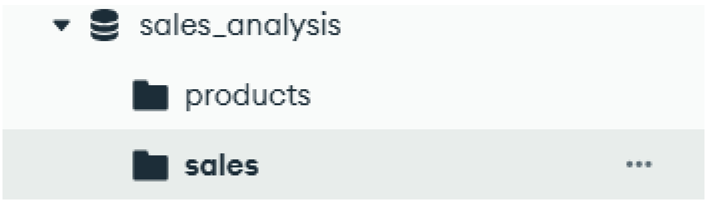
Run Aggregation:

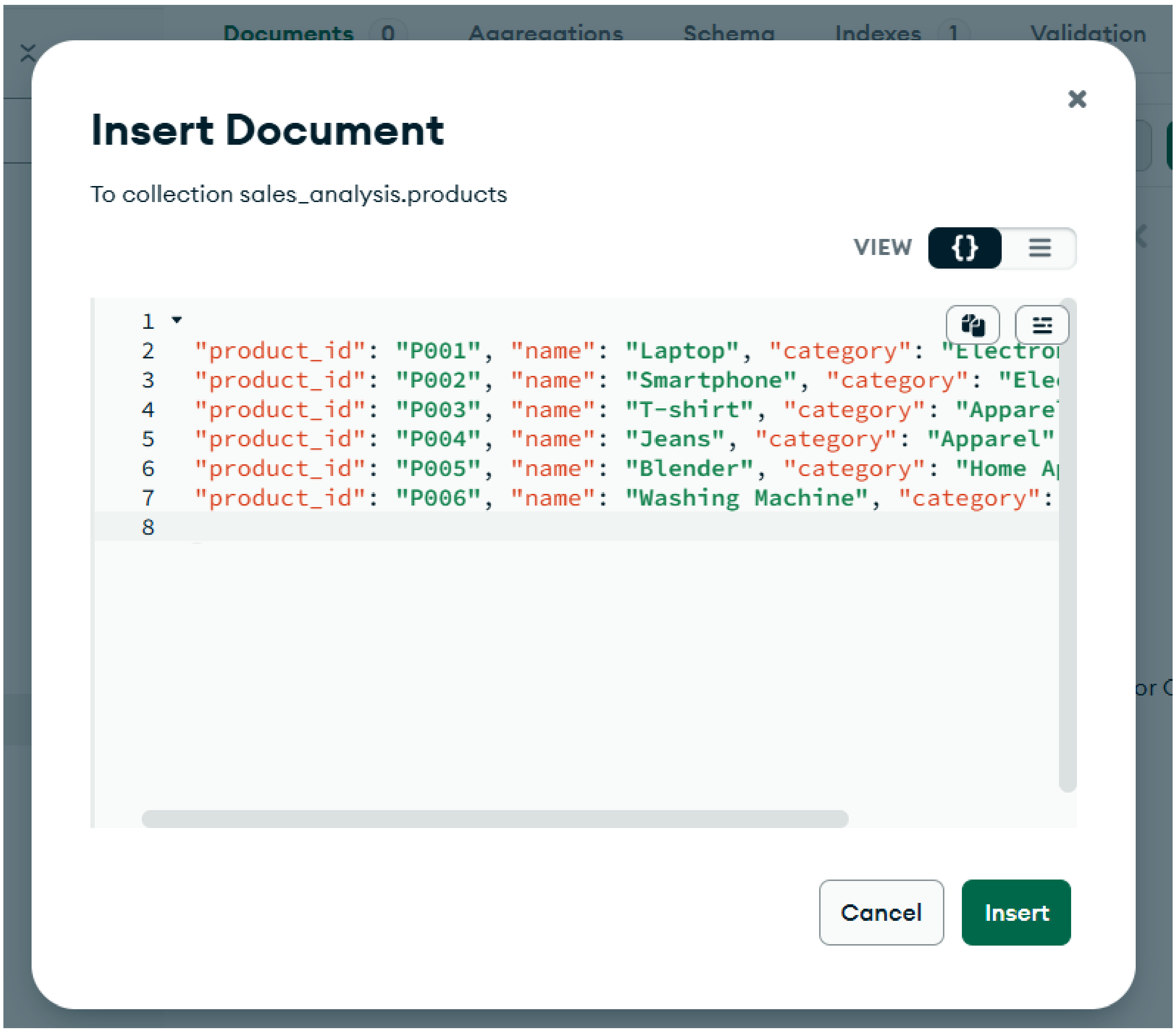
Daily Sales Count: Create a new pipeline.

- **Stage 1 ($group):** Group by date (_id: "$date"). Use **$count** for daily_sales_count.
- **Stage 2 ($sort):** Sort the results by the _id (date) in ascending order (

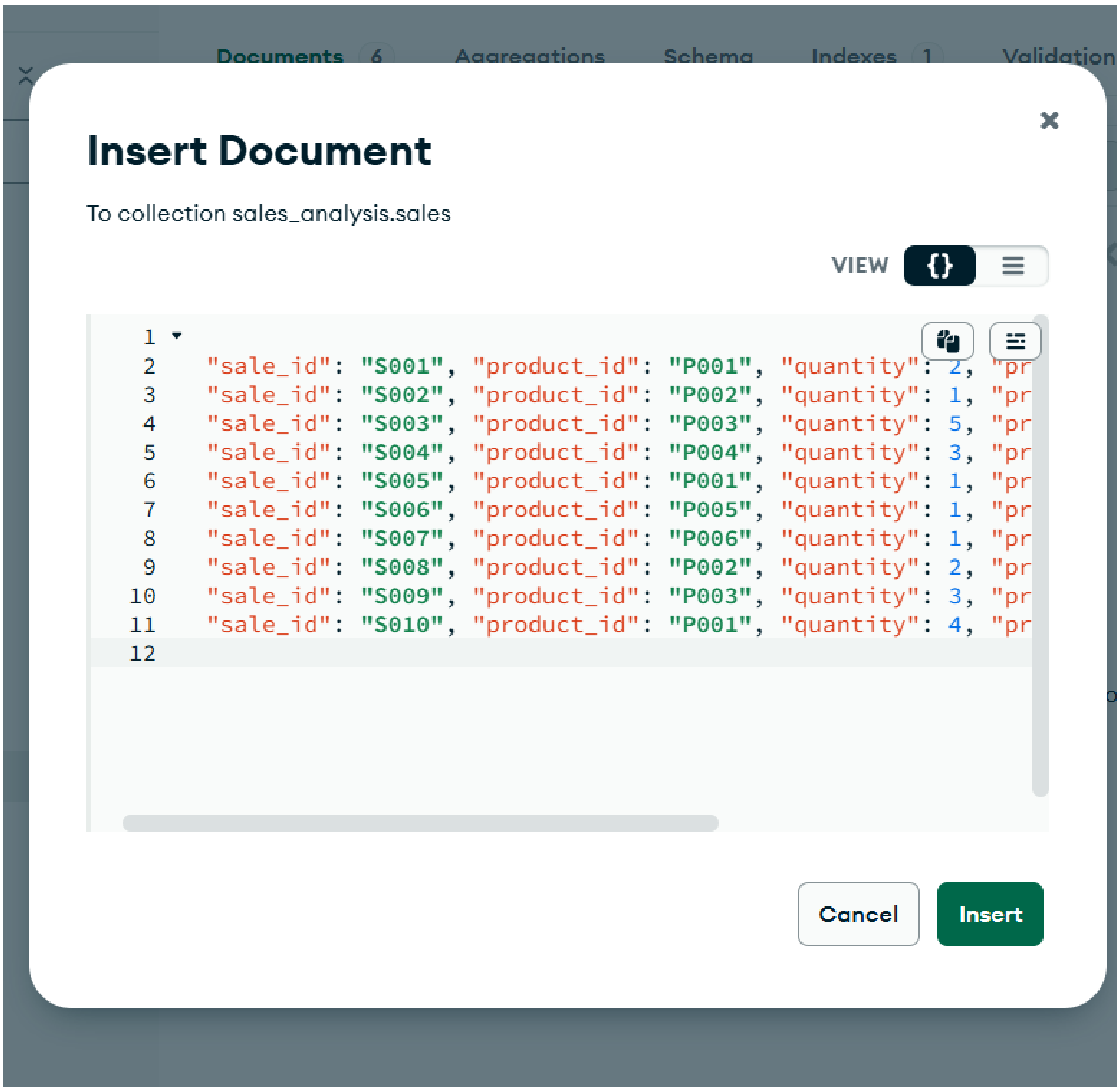
SCREENSHOTS

1. Setup and Data Insertion





```
"product_id": "P001", "name": "Laptop", "category": "Electro
"product_id": "P002", "name": "Smartphone", "category": "Ele
"product_id": "P003", "name": "T-shirt", "category": "Appare
"product_id": "P004", "name": "Jeans", "category": "Apparel"
"product_id": "P005", "name": "Blender", "category": "Home A
"product_id": "P006", "name": "Washing Machine", "category":
```



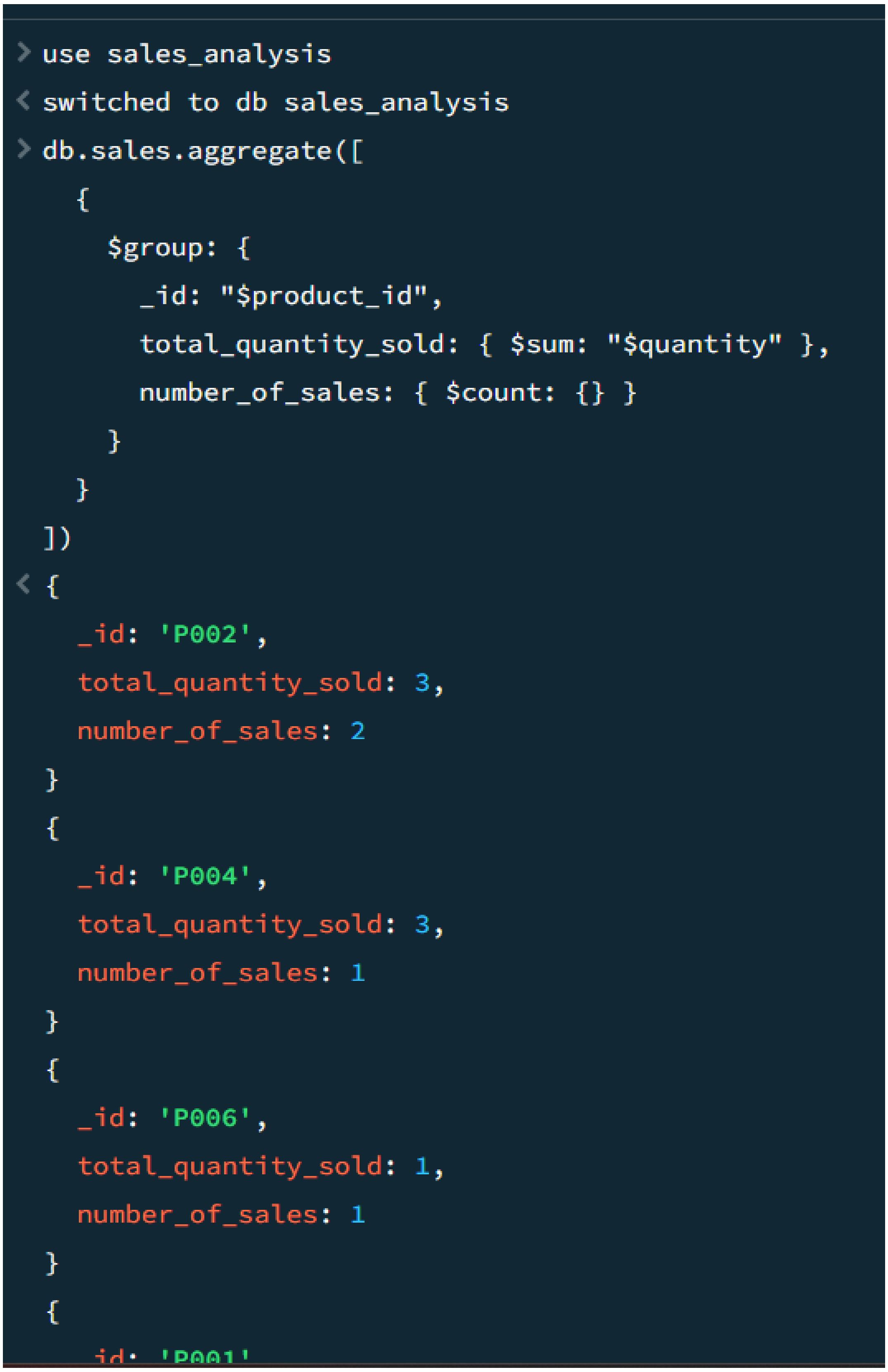
Insert Document

To collection sales_analysis.sales

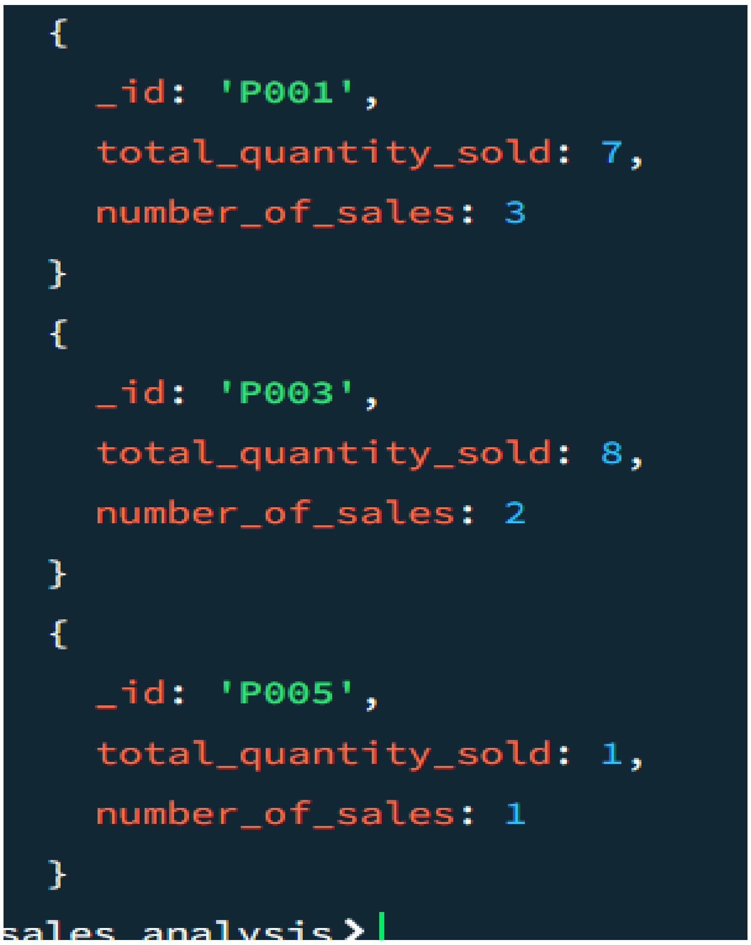
```
 "sale_id": "S001", "product_id": "P001", "quantity": 2, "pr
 "sale_id": "S002", "product_id": "P002", "quantity": 1, "pr
 "sale_id": "S003", "product_id": "P003", "quantity": 5, "pr
 "sale_id": "S004", "product_id": "P004", "quantity": 3, "pr
 "sale_id": "S005", "product_id": "P001", "quantity": 1, "pr
 "sale_id": "S006", "product_id": "P005", "quantity": 1, "pr
 "sale_id": "S007", "product_id": "P006", "quantity": 1, "pr
 "sale_id": "S008", "product_id": "P002", "quantity": 2, "pr
 "sale_id": "S009", "product_id": "P003", "quantity": 3, "pr
 "sale_id": "S010", "product_id": "P001", "quantity": 4, "pr
```

1. Using $group with $sum and $count

This pipeline groups documents by product_id and calculates the total quantity and the number of sales.

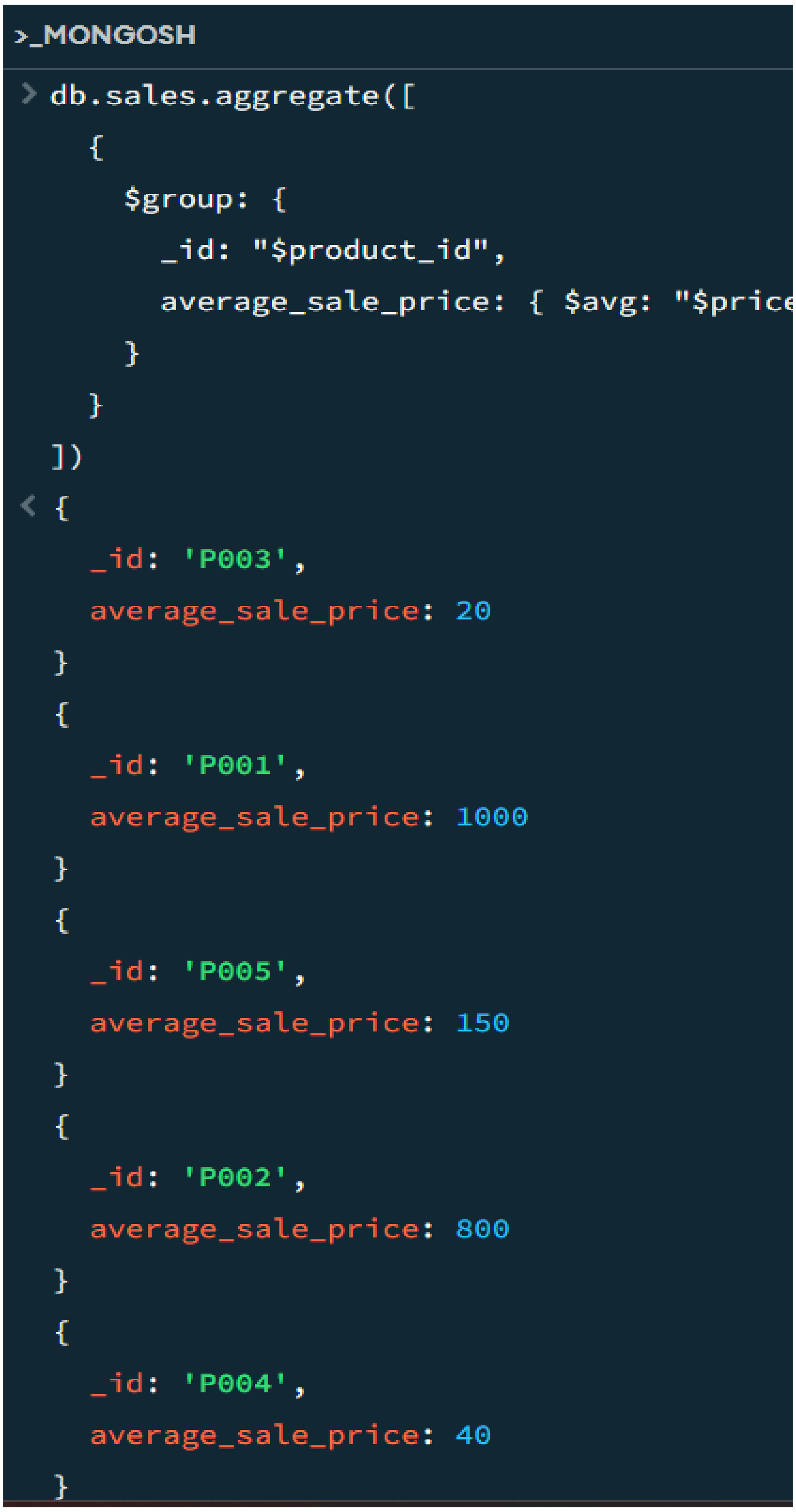


```
> use sales_analysis
< switched to db sales_analysis
> db.sales.aggregate([
    {
      $group: {
        _id: "$product_id",
        total_quantity_sold: { $sum: "$quantity" },
        number_of_sales: { $count: {} }
      }
    }
  ])
< {
    _id: 'P002',
    total_quantity_sold: 3,
    number_of_sales: 2
  }
  {
    _id: 'P004',
    total_quantity_sold: 3,
    number_of_sales: 1
  }
  {
    _id: 'P006',
    total_quantity_sold: 1,
    number_of_sales: 1
  }
  {
```

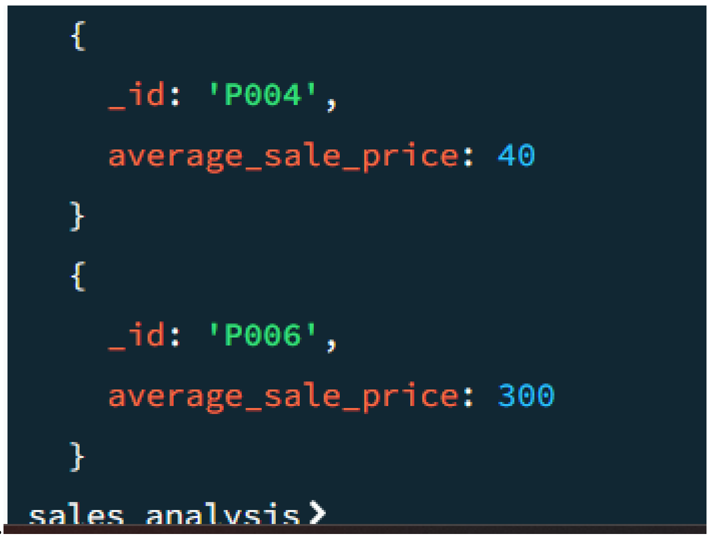

```
  {
    _id: 'P001',
    total_quantity_sold: 7,
    number_of_sales: 3
  }
  {
    _id: 'P003',
    total_quantity_sold: 8,
    number_of_sales: 2
  }
  {
    _id: 'P005',
    total_quantity_sold: 1,
    number_of_sales: 1
  }
sales_analysis>
```

2. Using $group with $avg

This pipeline calculates the average price of sales, grouping the results by product_id.



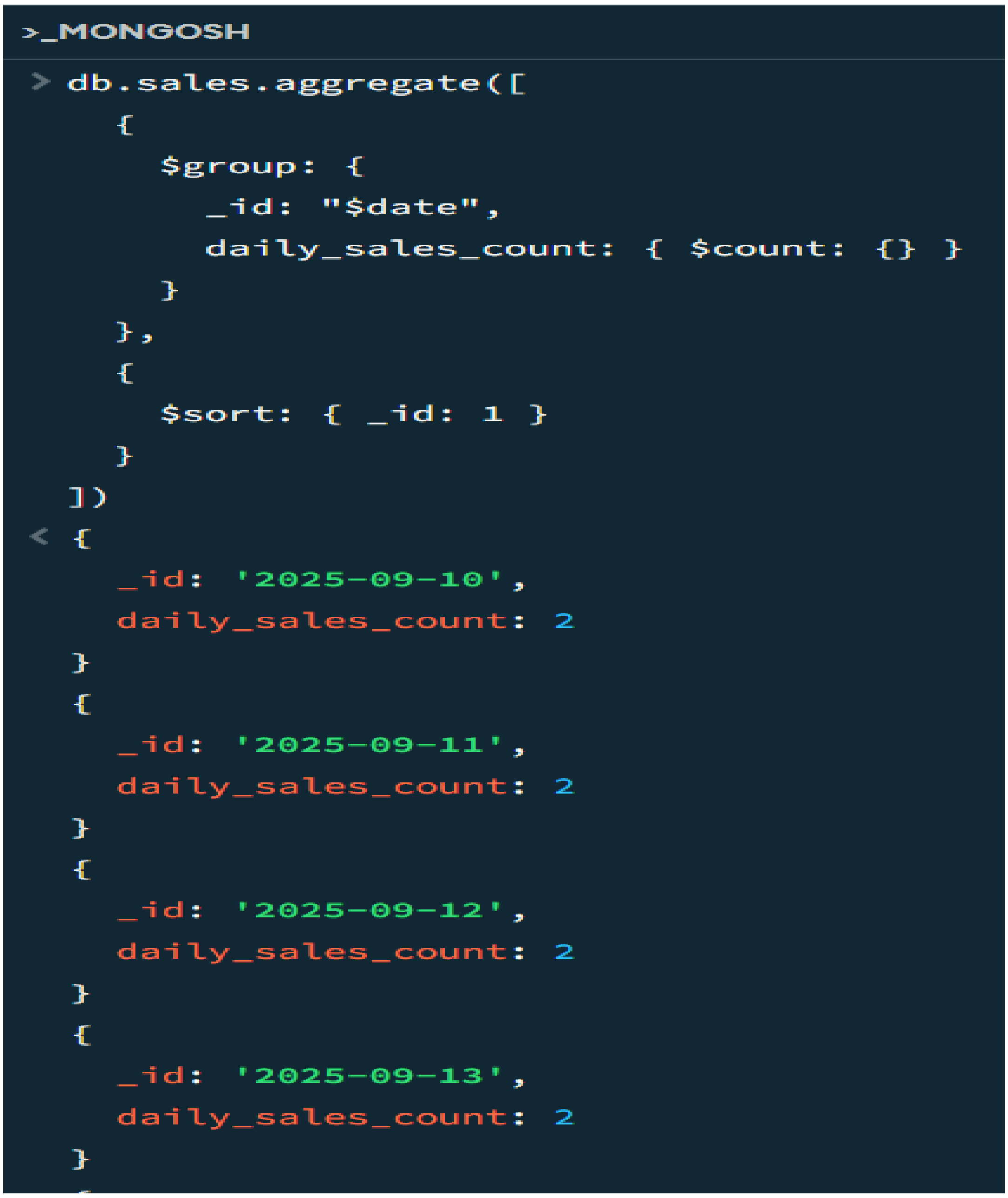
```
>_MONGOSH
> db.sales.aggregate([
    {
      $group: {
        _id: "$product_id",
        average_sale_price: { $avg: "$price
      }
    }
  ])
< {
    _id: 'P003',
    average_sale_price: 20
  }
  {
    _id: 'P001',
    average_sale_price: 1000
  }
  {
    _id: 'P005',
    average_sale_price: 150
  }
  {
    _id: 'P002',
    average_sale_price: 800
  }
  {
    _id: 'P004',
    average_sale_price: 40
  }
```



```
  {
    _id: 'P004',
    average_sale_price: 40
  }
  {
    _id: 'P006',
    average_sale_price: 300
  }
sales_analysis>
```

3. Using $group with $count (and a $sort stage)

This pipeline groups documents by date and simply counts the number of sales for each day.



```
>_MONGOSH
> db.sales.aggregate([
    {
      $group: {
        _id: "$date",
        daily_sales_count: { $count: {} }
      }
    },
    {
      $sort: { _id: 1 }
    }
  ])
< {
    _id: '2025-09-10',
    daily_sales_count: 2
  }
  {
    _id: '2025-09-11',
    daily_sales_count: 2
  }
  {
    _id: '2025-09-12',
    daily_sales_count: 2
  }
  {
    _id: '2025-09-13',
    daily_sales_count: 2
  }
```

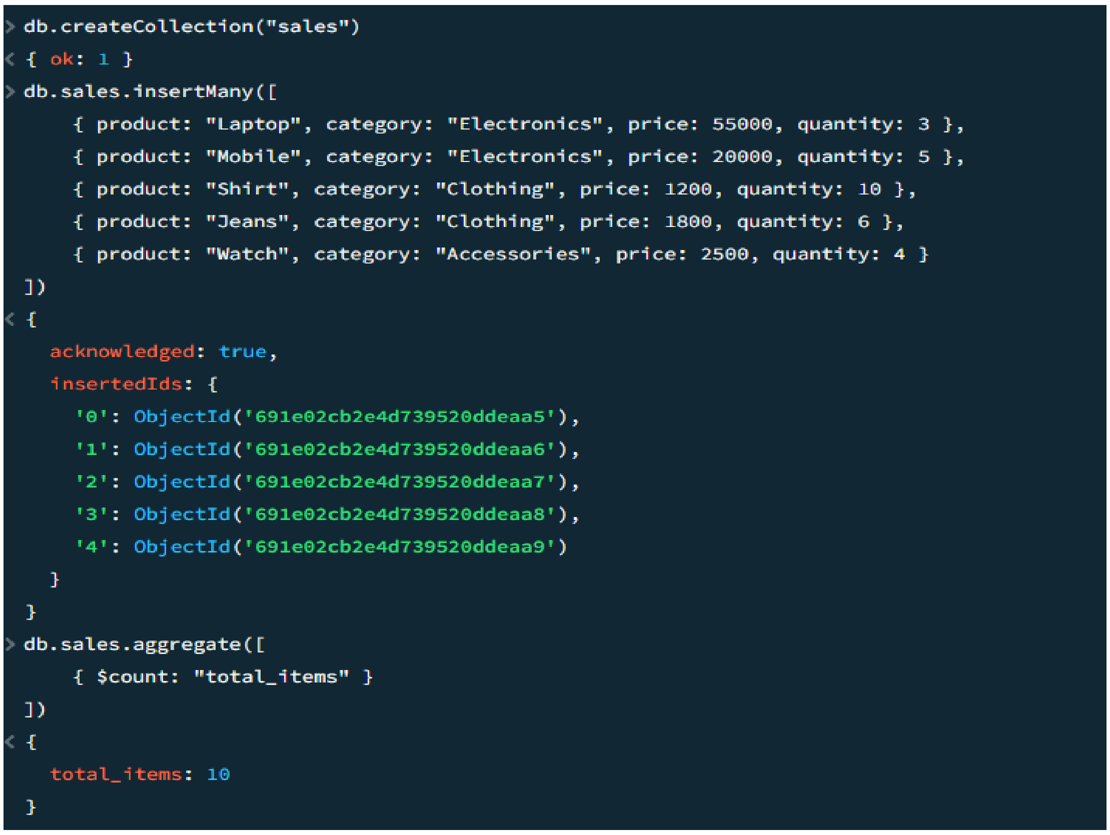
EXPERIMENT NO: 8

Create a collection to perform aggregation function

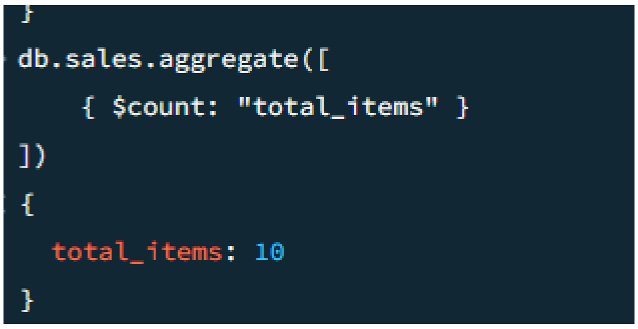
AIM:

To create a collection to perform aggregation Function

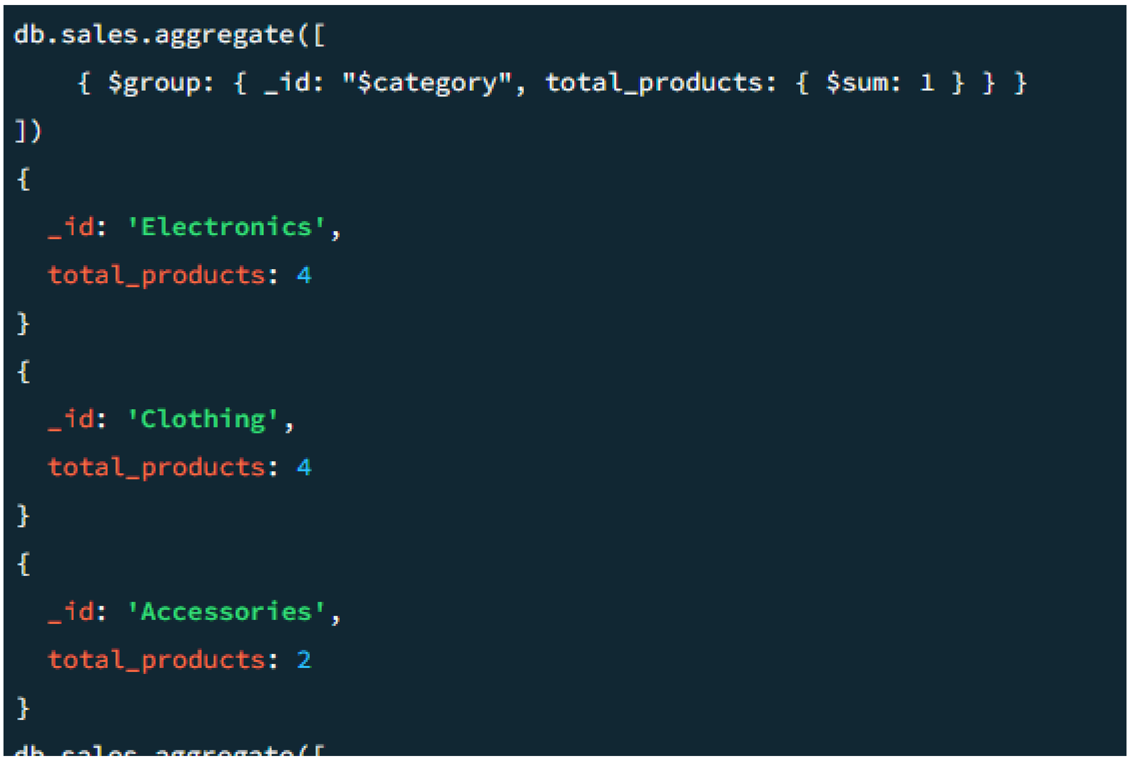
SCREENSHOTS:



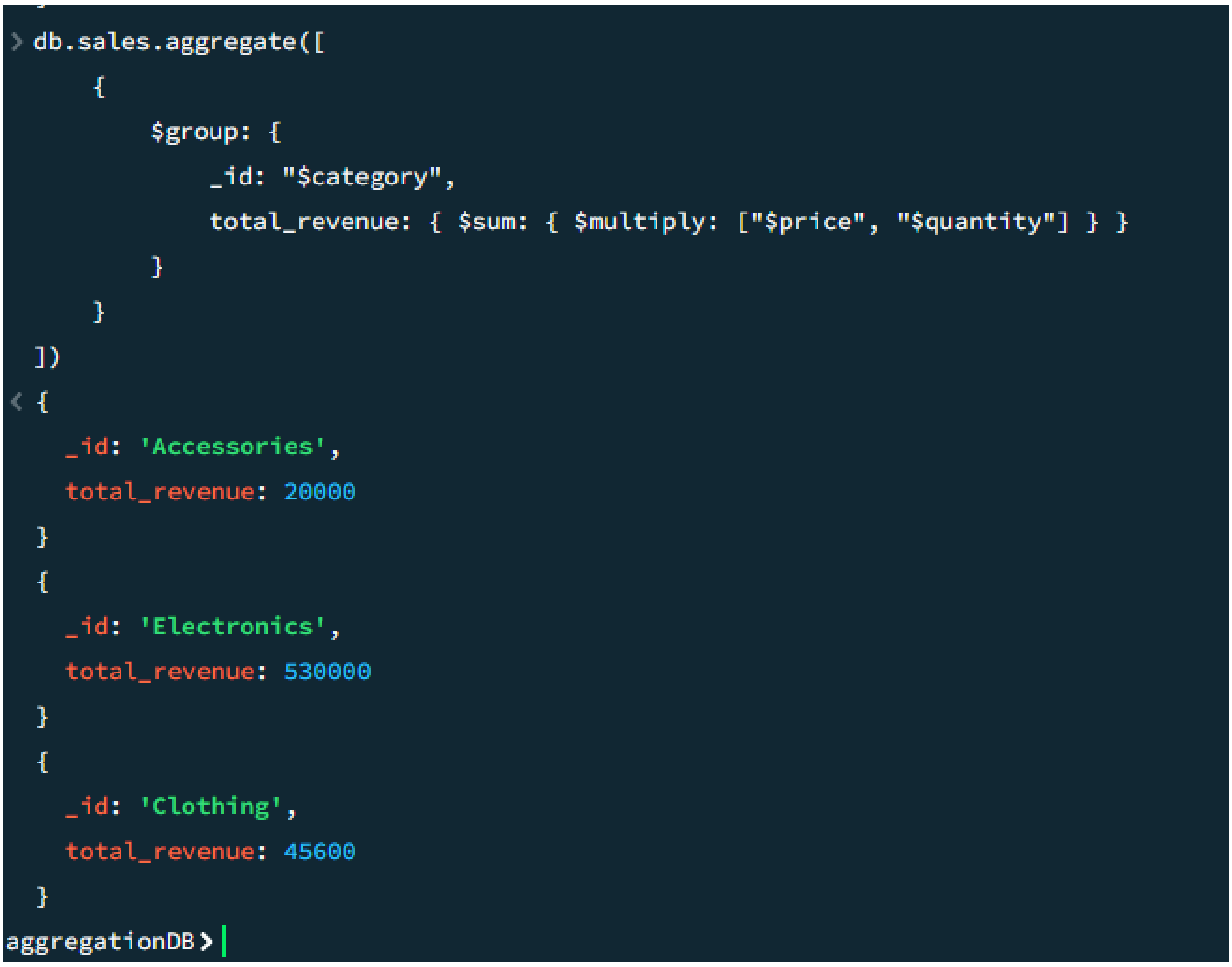
```
> db.createCollection("sales")
< { ok: 1 }
> db.sales.insertMany([
    { product: "Laptop", category: "Electronics", price: 55000, quantity: 3 },
    { product: "Mobile", category: "Electronics", price: 20000, quantity: 5 },
    { product: "Shirt", category: "Clothing", price: 1200, quantity: 10 },
    { product: "Jeans", category: "Clothing", price: 1800, quantity: 6 },
    { product: "Watch", category: "Accessories", price: 2500, quantity: 4 }
  ])
< {
    acknowledged: true,
    insertedIds: {
      '0': ObjectId('691e02cb2e4d739520ddeaa5'),
      '1': ObjectId('691e02cb2e4d739520ddeaa6'),
      '2': ObjectId('691e02cb2e4d739520ddeaa7'),
      '3': ObjectId('691e02cb2e4d739520ddeaa8'),
      '4': ObjectId('691e02cb2e4d739520ddeaa9')
    }
  }
> db.sales.aggregate([
    { $count: "total_items" }
  ])
< {
    total_items: 10
  }
```



```
}
db.sales.aggregate([
    { $count: "total_items" }
])
{
  total_items: 10
}
```



```
db.sales.aggregate([
    { $group: { _id: "$category", total_products: { $sum: 1 } } }
])
{
  _id: 'Electronics',
  total_products: 4
}
{
  _id: 'Clothing',
  total_products: 4
}
{
  _id: 'Accessories',
  total_products: 2
}
db.sales.aggregate([
```



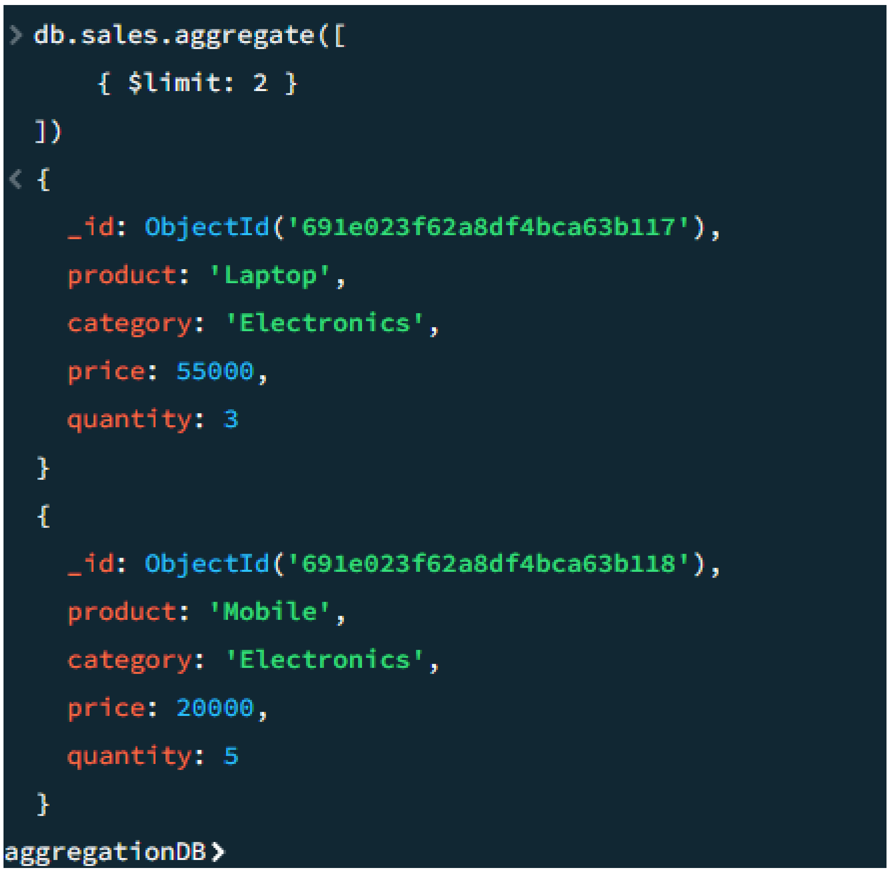
```
> db.sales.aggregate([
      {
          $group: {
              _id: "$category",
              total_revenue: { $sum: { $multiply: ["$price", "$quantity"] } }
          }
      }
  ])
< {
    _id: 'Accessories',
    total_revenue: 20000
  }
  {
    _id: 'Electronics',
    total_revenue: 530000
  }
  {
    _id: 'Clothing',
    total_revenue: 45600
  }
aggregationDB>
```

EXPERIMENT 9:

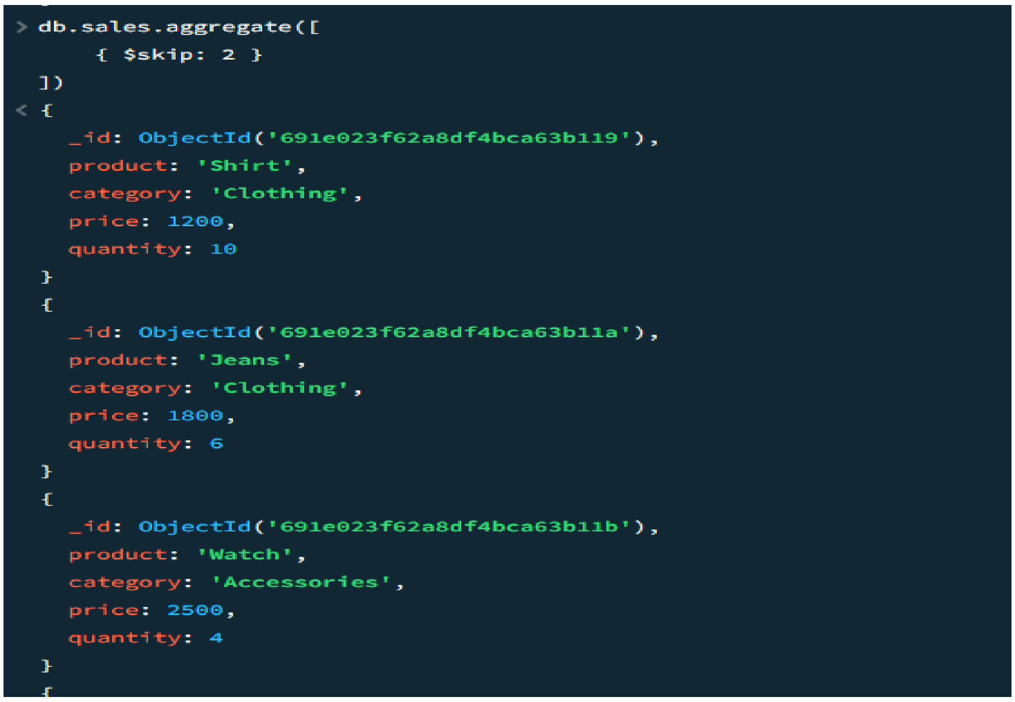
AIM:

To Demonstrate the use of aggregation, limit, skip, sorting and unwind operation in MongoDB compass using the sales and product collection from the sale database.

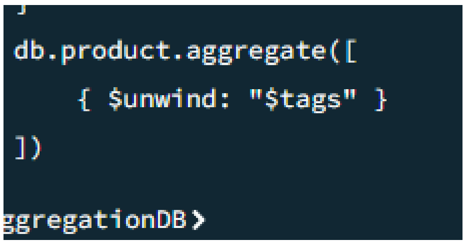
SCREENSHOTS:



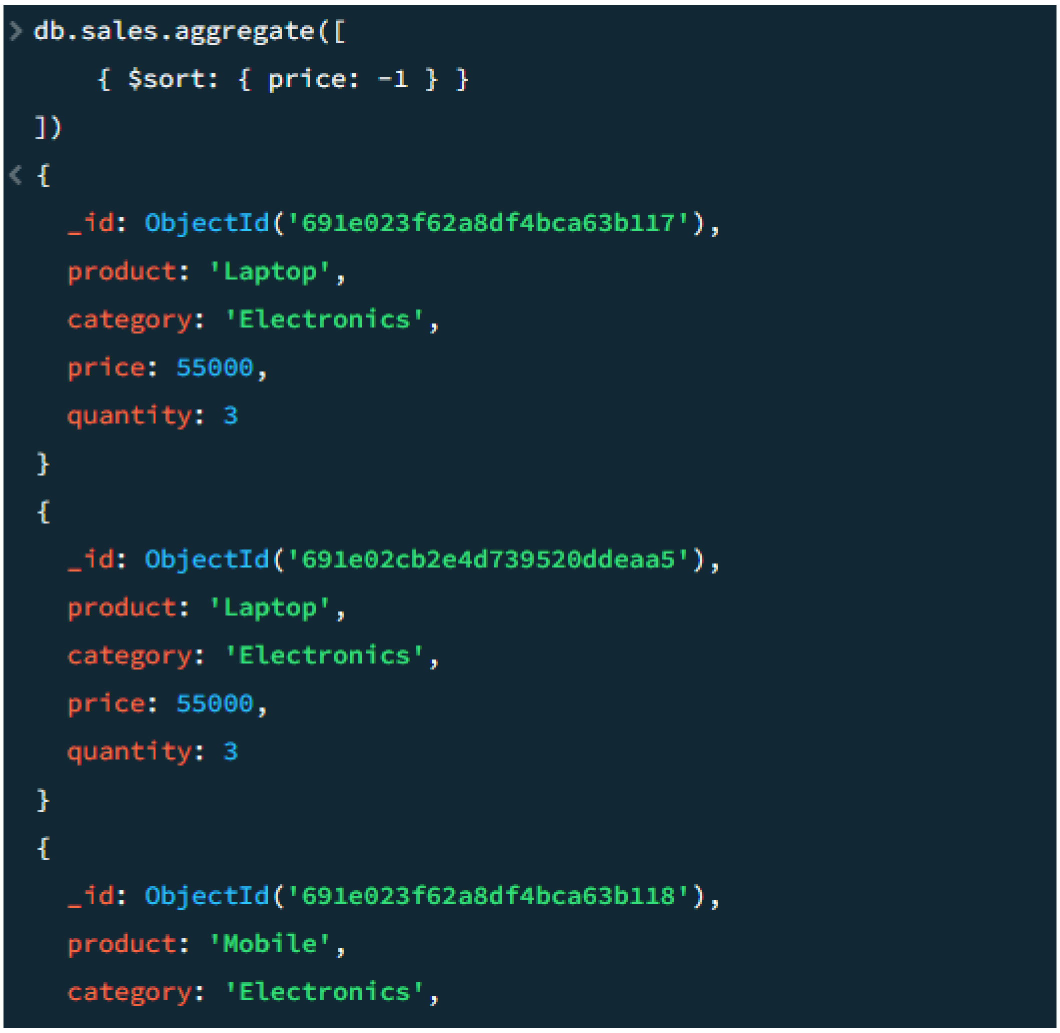
```
> db.sales.aggregate([
      { $limit: 2 }
  ])
< {
    _id: ObjectId('691e023f62a8df4bca63b117'),
    product: 'Laptop',
    category: 'Electronics',
    price: 55000,
    quantity: 3
  }
  {
    _id: ObjectId('691e023f62a8df4bca63b118'),
    product: 'Mobile',
    category: 'Electronics',
    price: 20000,
    quantity: 5
  }
aggregationDB>
```



```
> db.sales.aggregate([
      { $skip: 2 }
  ])
< {
    _id: ObjectId('691e023f62a8df4bca63b119'),
    product: 'Shirt',
    category: 'Clothing',
    price: 1200,
    quantity: 10
  }
  {
    _id: ObjectId('691e023f62a8df4bca63b11a'),
    product: 'Jeans',
    category: 'Clothing',
    price: 1800,
    quantity: 6
  }
  {
    _id: ObjectId('691e023f62a8df4bca63b11b'),
    product: 'Watch',
    category: 'Accessories',
    price: 2500,
    quantity: 4
  }
  {
```



```
db.product.aggregate([
    { $unwind: "$tags" }
])

ggregationDB>
```



```
> db.sales.aggregate([
      { $sort: { price: -1 } }
  ])
< {
    _id: ObjectId('691e023f62a8df4bca63b117'),
    product: 'Laptop',
    category: 'Electronics',
    price: 55000,
    quantity: 3
  }
  {
    _id: ObjectId('691e02cb2e4d739520ddeaa5'),
    product: 'Laptop',
    category: 'Electronics',
    price: 55000,
    quantity: 3
  }
  {
    _id: ObjectId('691e023f62a8df4bca63b118'),
    product: 'Mobile',
    category: 'Electronics',
```

EXPERIMENT 10

AIM

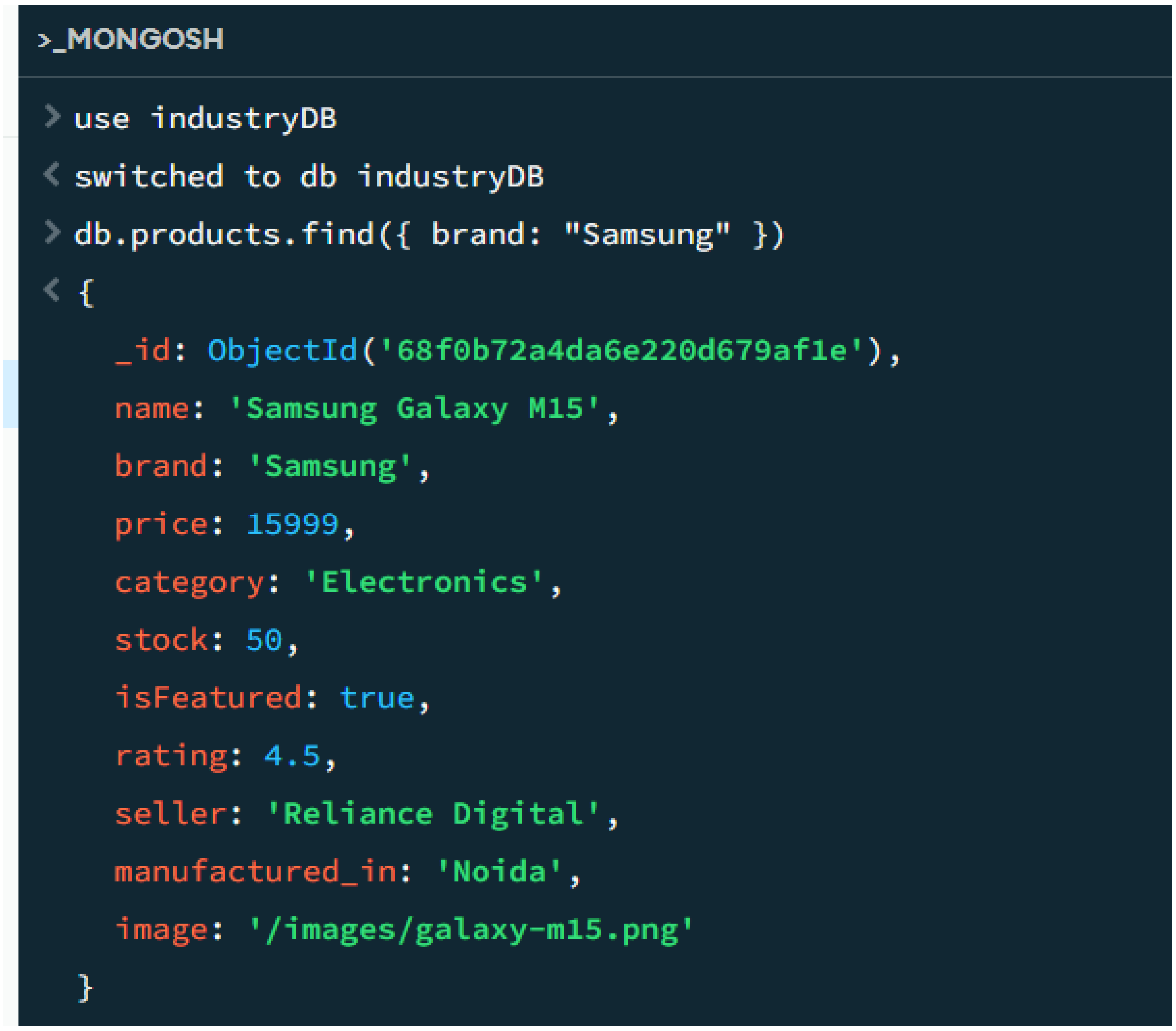
To demonstrate Projection and Filtering queries on DataBase.

PROCEDURE

Projection Queries

Find all products by the brand "Samsung"

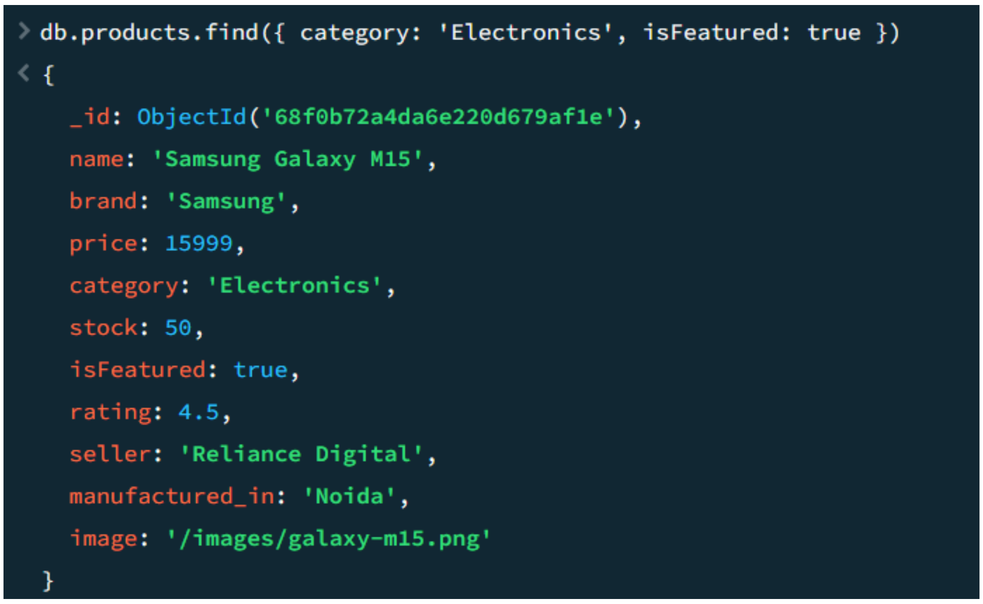
This query looks for documents where the brand field is exactly "Samsung"



```
>_MONGOSH
> use industryDB
< switched to db industryDB
> db.products.find({ brand: "Samsung" })
< {
    _id: ObjectId('68f0b72a4da6e220d679af1e'),
    name: 'Samsung Galaxy M15',
    brand: 'Samsung',
    price: 15999,
    category: 'Electronics',
    stock: 50,
    isFeatured: true,
    rating: 4.5,
    seller: 'Reliance Digital',
    manufactured_in: 'Noida',
    image: '/images/galaxy-m15.png'
  }
```

Find featured products in the 'Electronics' category

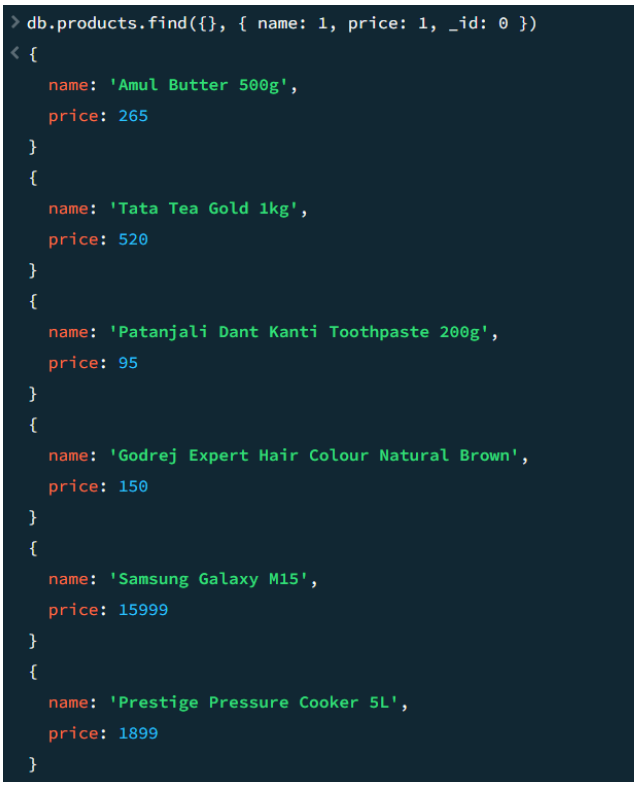
This query combines two conditions: the category must be 'Electronics' AND the isFeatured field must be true.



```
> db.products.find({ category: 'Electronics', isFeatured: true })
< {
    _id: ObjectId('68f0b72a4da6e220d679af1e'),
    name: 'Samsung Galaxy M15',
    brand: 'Samsung',
    price: 15999,
    category: 'Electronics',
    stock: 50,
    isFeatured: true,
    rating: 4.5,
    seller: 'Reliance Digital',
    manufactured_in: 'Noida',
    image: '/images/galaxy-m15.png'
  }
```

Show only the name and price of all products

This query returns all documents but only includes the name and price fields. We explicitly exclude _id with _id: 0 for a cleaner output.

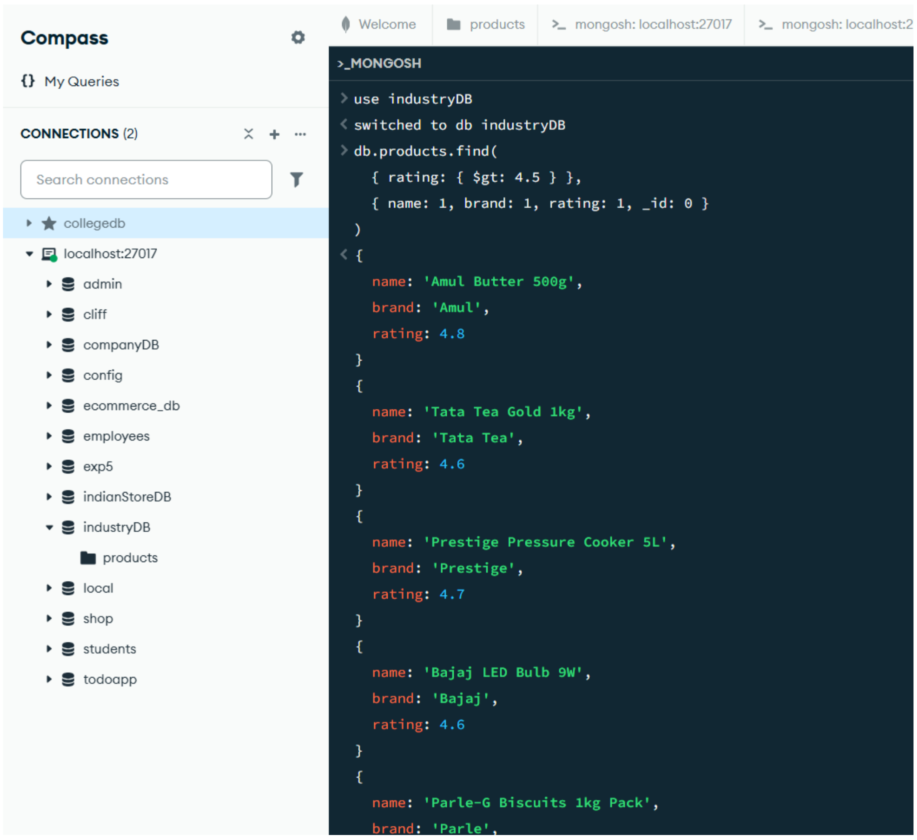


```
> db.products.find({}, { name: 1, price: 1, _id: 0 })
< {
    name: 'Amul Butter 500g',
    price: 265
  }
  {
    name: 'Tata Tea Gold 1kg',
    price: 520
  }
  {
    name: 'Patanjali Dant Kanti Toothpaste 200g',
    price: 95
  }
  {
    name: 'Godrej Expert Hair Colour Natural Brown',
    price: 150
  }
  {
    name: 'Samsung Galaxy M15',
    price: 15999
  }
  {
    name: 'Prestige Pressure Cooker 5L',
    price: 1899
  }
```

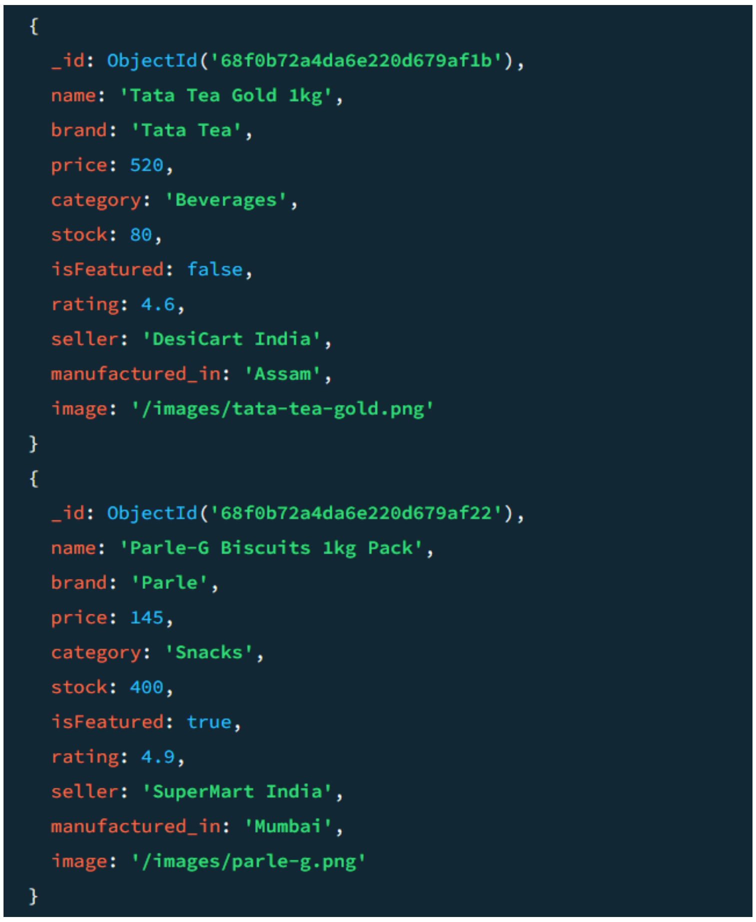
Filtering Queries

Find all products with a rating higher than 4.5, but only show their name, brand, and rating.

- **Filter:** { rating: { $gt: 4.5 } } finds the documents.
- **Projection:** { name: 1, brand: 1, rating: 1, _id: 0 } shapes the output for those found documents.



```
> use industryDB
< switched to db industryDB
> db.products.find(
    { rating: { $gt: 4.5 } },
    { name: 1, brand: 1, rating: 1, _id: 0 }
  )
< {
    name: 'Amul Butter 500g',
    brand: 'Amul',
    rating: 4.8
  }
  {
    name: 'Tata Tea Gold 1kg',
    brand: 'Tata Tea',
    rating: 4.6
  }
  {
    name: 'Prestige Pressure Cooker 5L',
    brand: 'Prestige',
    rating: 4.7
  }
  {
    name: 'Bajaj LED Bulb 9W',
    brand: 'Bajaj',
    rating: 4.6
  }
  {
    name: 'Parle-G Biscuits 1kg Pack',
    brand: 'Parle',
```



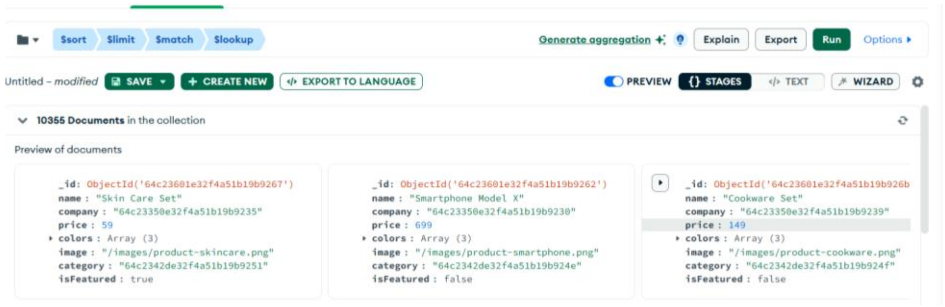
```
{
  _id: ObjectId('68f0b72a4da6e220d679af1b'),
  name: 'Tata Tea Gold 1kg',
  brand: 'Tata Tea',
  price: 520,
  category: 'Beverages',
  stock: 80,
  isFeatured: false,
  rating: 4.6,
  seller: 'DesiCart India',
  manufactured_in: 'Assam',
  image: '/images/tata-tea-gold.png'
}
{
  _id: ObjectId('68f0b72a4da6e220d679af22'),
  name: 'Parle-G Biscuits 1kg Pack',
  brand: 'Parle',
  price: 145,
  category: 'Snacks',
  stock: 400,
  isFeatured: true,
  rating: 4.9,
  seller: 'SuperMart India',
  manufactured_in: 'Mumbai',
  image: '/images/parle-g.png'
}
```

EXPERIMENT 11

AIM:

To implement the limit(), skip(), sort() methods in MongoDB

PROCEDURE:



10355 Documents in the collection

Preview of documents

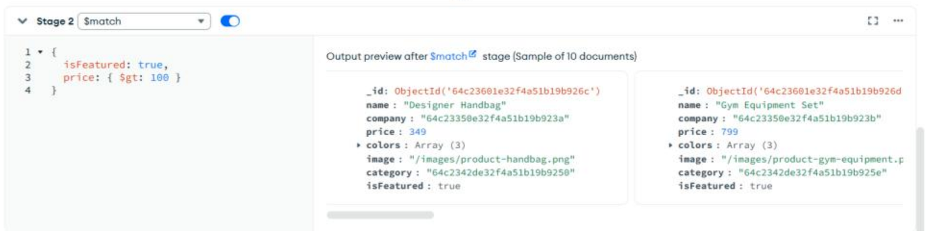
```
_id: ObjectId('64c23601e32f4a51b19b9267')
name : "Skin Care Set"
company : "64c23350e32f4a51b19b9235"
price : 59
colors : Array (3)
image : "/images/product-skincare.png"
category : "64c2342de32f4a51b19b9251"
isFeatured : true
```

```
_id: ObjectId('64c23601e32f4a51b19b9262')
name : "Smartphone Model X"
company : "64c23350e32f4a51b19b9230"
price : 699
colors : Array (3)
image : "/images/product-smartphone.png"
category : "64c2342de32f4a51b19b924e"
isFeatured : false
```

```
_id: ObjectId('64c23601e32f4a51b19b926b
name : "Cookware Set"
company : "64c23350e32f4a51b19b9239"
price : 149
colors : Array (3)
image : "/images/product-cookware.png"
category : "64c2342de32f4a51b19b924f"
isFeatured : false
```

Operation

- $match



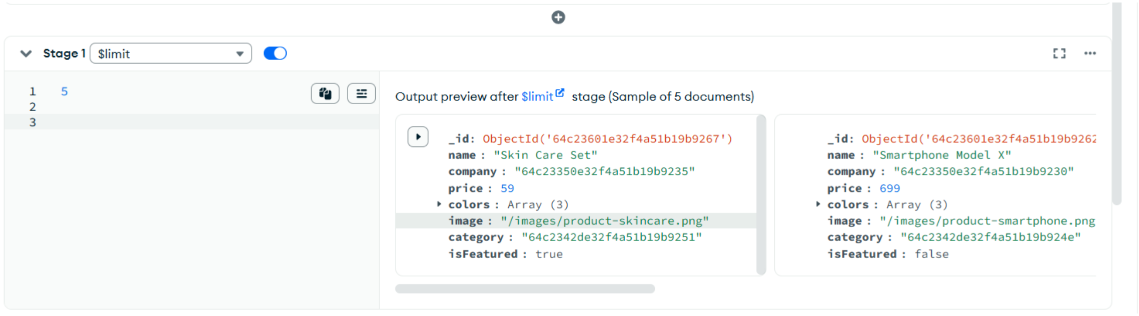
```
{
  isFeatured: true,
  price: { $gt: 100 }
}
```

Output preview after $match stage (Sample of 10 documents)

```
_id: ObjectId('64c23601e32f4a51b19b926c')
name : "Designer Handbag"
company : "64c23350e32f4a51b19b923a"
price : 349
colors : Array (3)
image : "/images/product-handbag.png"
category : "64c2342de32f4a51b19b9250"
isFeatured : true
```

```
_id: ObjectId('64c23601e32f4a51b19b926d
name : "Gym Equipment Set"
company : "64c23350e32f4a51b19b923b"
price : 799
colors : Array (3)
image : "/images/product-gym-equipment.p
category : "64c2342de32f4a51b19b925e"
isFeatured : true
```

$limit — Limit the Number of Results



```
5
```

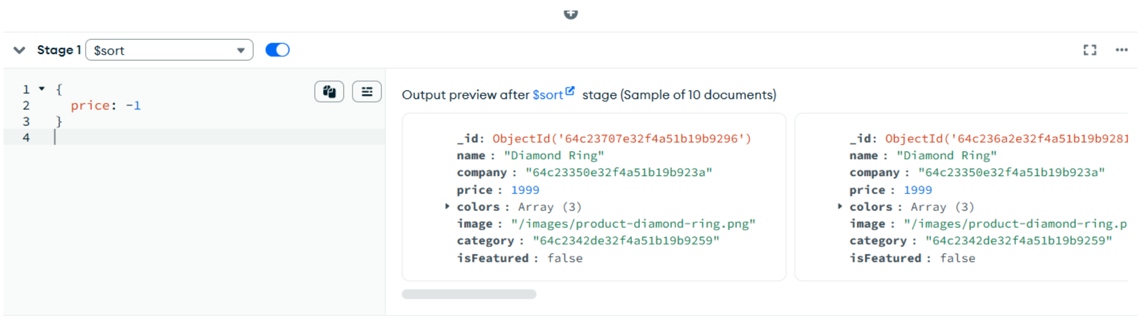
Output preview after $limit stage (Sample of 5 documents)

```
_id: ObjectId('64c23601e32f4a51b19b9267')
name : "Skin Care Set"
company : "64c23350e32f4a51b19b9235"
price : 59
colors : Array (3)
image : "/images/product-skincare.png"
category : "64c2342de32f4a51b19b9251"
isFeatured : true
```

```
_id: ObjectId('64c23601e32f4a51b19b9262
name : "Smartphone Model X"
company : "64c23350e32f4a51b19b9230"
price : 699
colors : Array (3)
image : "/images/product-smartphone.png
category : "64c2342de32f4a51b19b924e"
isFeatured : false
```

$sort — Sort Documents

Purpose: Sort the filtered products by price in descending order.



Stage 1 $sort

```
{
  price: -1
}
```

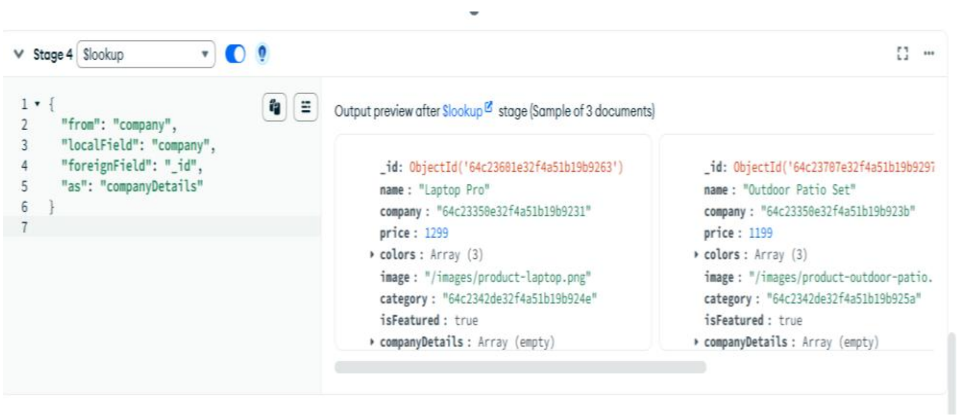
Output preview after $sort stage (Sample of 10 documents)

```
_id: ObjectId('64c23707e32f4a51b19b9296')
name : "Diamond Ring"
company : "64c23350e32f4a51b19b923a"
price : 1999
colors : Array (3)
image : "/images/product-diamond-ring.png"
category : "64c2342de32f4a51b19b9259"
isFeatured : false
```

```
_id: ObjectId('64c236a2e32f4a51b19b9281
name : "Diamond Ring"
company : "64c23350e32f4a51b19b923a"
price : 1999
colors : Array (3)
image : "/images/product-diamond-ring.p
category : "64c2342de32f4a51b19b9259"
isFeatured : false
```

$lookup — Join with Another Collection

Purpose: Join products with another collection (for example, companies) based on the company field.



Stage 4 $lookup

```
{
  "from": "company",
  "localField": "company",
  "foreignField": "_id",
  "as": "companyDetails"
}
```

Output preview after $lookup stage (Sample of 3 documents)

```
_id: ObjectId('64c23601e32f4a51b19b9263')
name : "Laptop Pro"
company : "64c23350e32f4a51b19b9231"
price : 1299
colors : Array (3)
image : "/images/product-laptop.png"
category : "64c2342de32f4a51b19b924e"
isFeatured : true
companyDetails : Array (empty)
```

```
_id: ObjectId('64c23787e32f4a51b19b9297
name : "Outdoor Patio Set"
company : "64c23350e32f4a51b19b923b"
price : 1199
colors : Array (3)
image : "/images/product-outdoor-patio.
category : "64c2342de32f4a51b19b925a"
isFeatured : true
companyDetails : Array (empty)
```

EXPERIMENT NO: 12 & 13

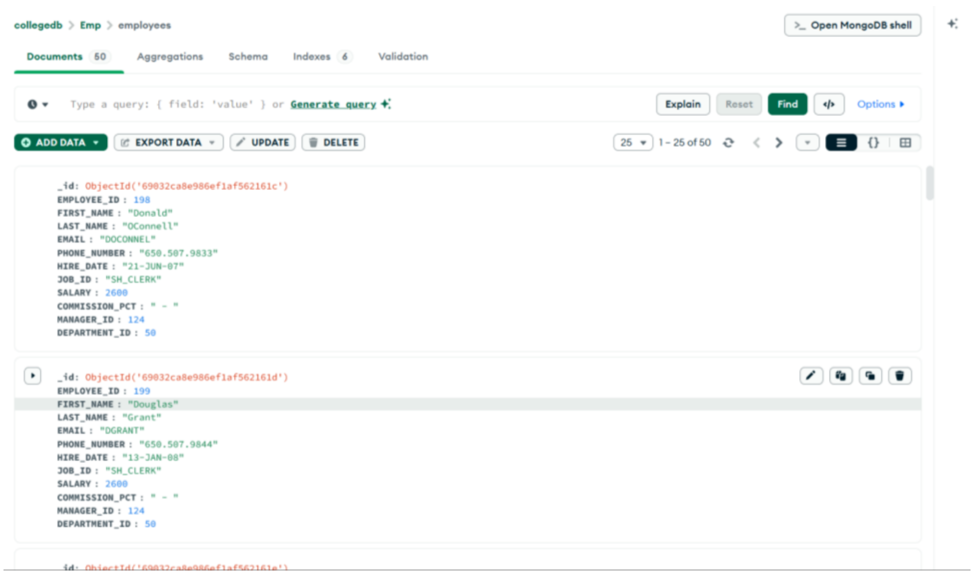
AIM

To understand how indexing improves query performance in MongoDB by creating indexes on one or more fields of a collection.

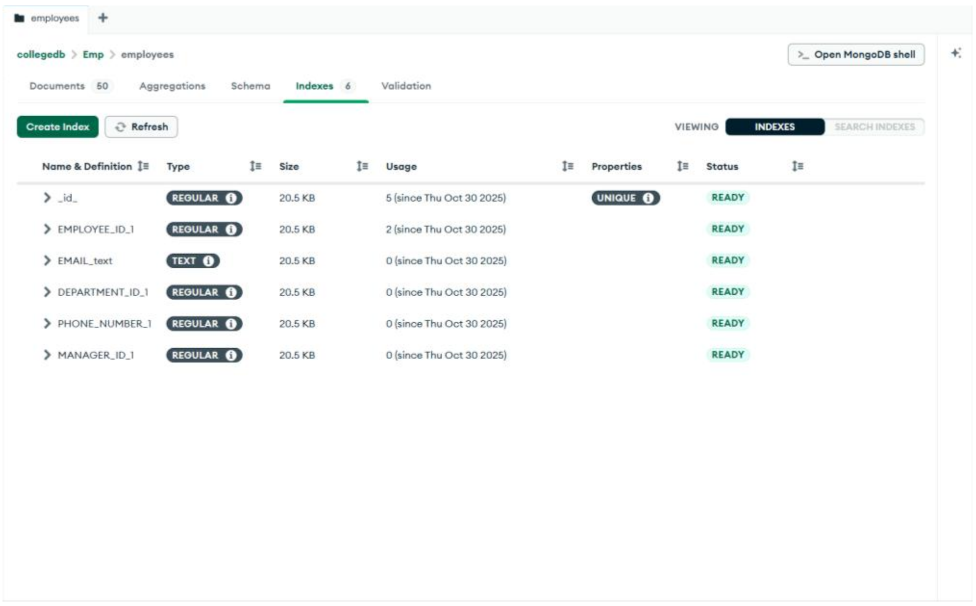
PROCEDURE

1. Open MongoDB Compass and connect to the MongoDB server.
2. Select the database collegedb → Emp → employees.
3. Go to the Documents tab to verify that data is imported correctly.
4. Click on the Indexes tab to view existing indexes (default _id_).
5. Click Create Index → add field EMPLOYEE_ID → choose Ascending (1) → click Create Index.
6. Verify the new index (EMPLOYEE_ID_1) under the Indexes tab.
7. Go back to the Documents tab and run a filter query like { "EMPLOYEE_ID": 198 } to test performance.
8. Observe that the query runs faster using the new index.

SCREENSHOTS:



```
_id: ObjectId('69032ca8e986ef1af562161c')
EMPLOYEE_ID : 198
FIRST_NAME : "Donald"
LAST_NAME : "OConnell"
EMAIL : "DOCONNEL"
PHONE_NUMBER : "650.507.9833"
HIRE_DATE : "21-JUN-07"
JOB_ID : "SH_CLERK"
SALARY : 2600
COMMISSION_PCT : " - "
MANAGER_ID : 124
DEPARTMENT_ID : 50

_id: ObjectId('69032ca8e986ef1af562161d')
EMPLOYEE_ID : 199
FIRST_NAME : "Douglas"
LAST_NAME : "Grant"
EMAIL : "DGRANT"
PHONE_NUMBER : "650.507.9844"
HIRE_DATE : "13-JAN-08"
JOB_ID : "SH_CLERK"
SALARY : 2600
COMMISSION_PCT : " - "
MANAGER_ID : 124
DEPARTMENT_ID : 50

_id: ObjectId('69032ca8e986ef1af562161e')
```



Name & Definition	Type	Size	Usage	Properties	Status
_id_	REGULAR	20.5 KB	5 (since Thu Oct 30 2025)	UNIQUE	READY
EMPLOYEE_ID_1	REGULAR	20.5 KB	2 (since Thu Oct 30 2025)		READY
EMAIL_text	TEXT	20.5 KB	0 (since Thu Oct 30 2025)		READY
DEPARTMENT_ID_1	REGULAR	20.5 KB	0 (since Thu Oct 30 2025)		READY
PHONE_NUMBER_1	REGULAR	20.5 KB	0 (since Thu Oct 30 2025)		READY
MANAGER_ID_1	REGULAR	20.5 KB	0 (since Thu Oct 30 2025)		READY

EXPERIMENT 14 & 15

MONGODB CRUD OPERATIONS USING JAVA & PHP

PART 1

AIM

To implement a Java program for performing CRUD (Create, Read, Update, and Delete) operations on a MongoDB database using the MongoDB Java driver and Maven.

PROCEDURE

1. Environment Setup

- Install **VS Code** or **Eclipse IDE** with **Java JDK (17 or above)**.
- Install **Apache Maven** and verify setup using the command mvn -version.

2. Database Configuration

- Start the MongoDB service using:
- net start MongoDB
- Open **MongoDB Compass** and connect using the URI: mongodb://localhost:27017
- Create a new database named **test** with a collection named **students**.

3. Maven Project Setup

- Create a new **Maven Project** in VS Code or Eclipse.
- Configure the pom.xml file to include the **MongoDB Java Driver** dependency:
- <dependency>
-     <groupId>org.mongodb</groupId>
-     <artifactId>mongodb-driver-sync</artifactId>
-     <version>5.6.1</version>
- </dependency>

4. Java Program Development

- Create a Java class named MongoConnect under the package com.example.
- Write Java code to:
    - Establish connection to MongoDB.
    - Perform **CREATE** operation (insert a document).
    - Perform **READ** operation (display all documents).
    - Perform **UPDATE** operation (modify a specific record).
    - Perform **DELETE** operation (remove a document).
- Use authentication URI:
  mongodb://<username>:<password>@localhost:27017/?authSource=admin

5. Execution

- Save all files and open a terminal in the project folder.
- Compile and run the program using:
- mvn compile exec:java -Dexec.mainClass="com.example.MongoConnect"
- Observe the output messages for successful database connection and CRUD operations.

6. Verification

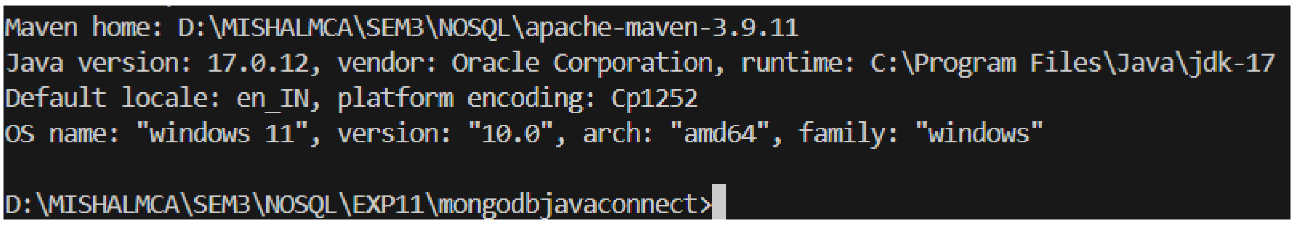
- Open **MongoDB Compass**.
- Navigate to the **test → students** collection.
- Click **Refresh** to verify the inserted, updated, or deleted documents.

7. Result

- The Java program successfully established a connection with MongoDB and performed all CRUD operations (Create, Read, Update, Delete) on the students collection.

SCREENSHOTS:

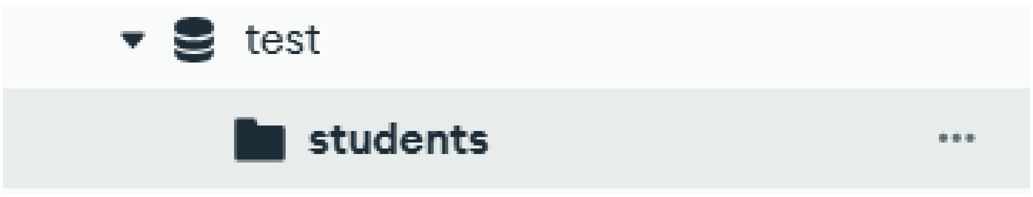
Environment Setup

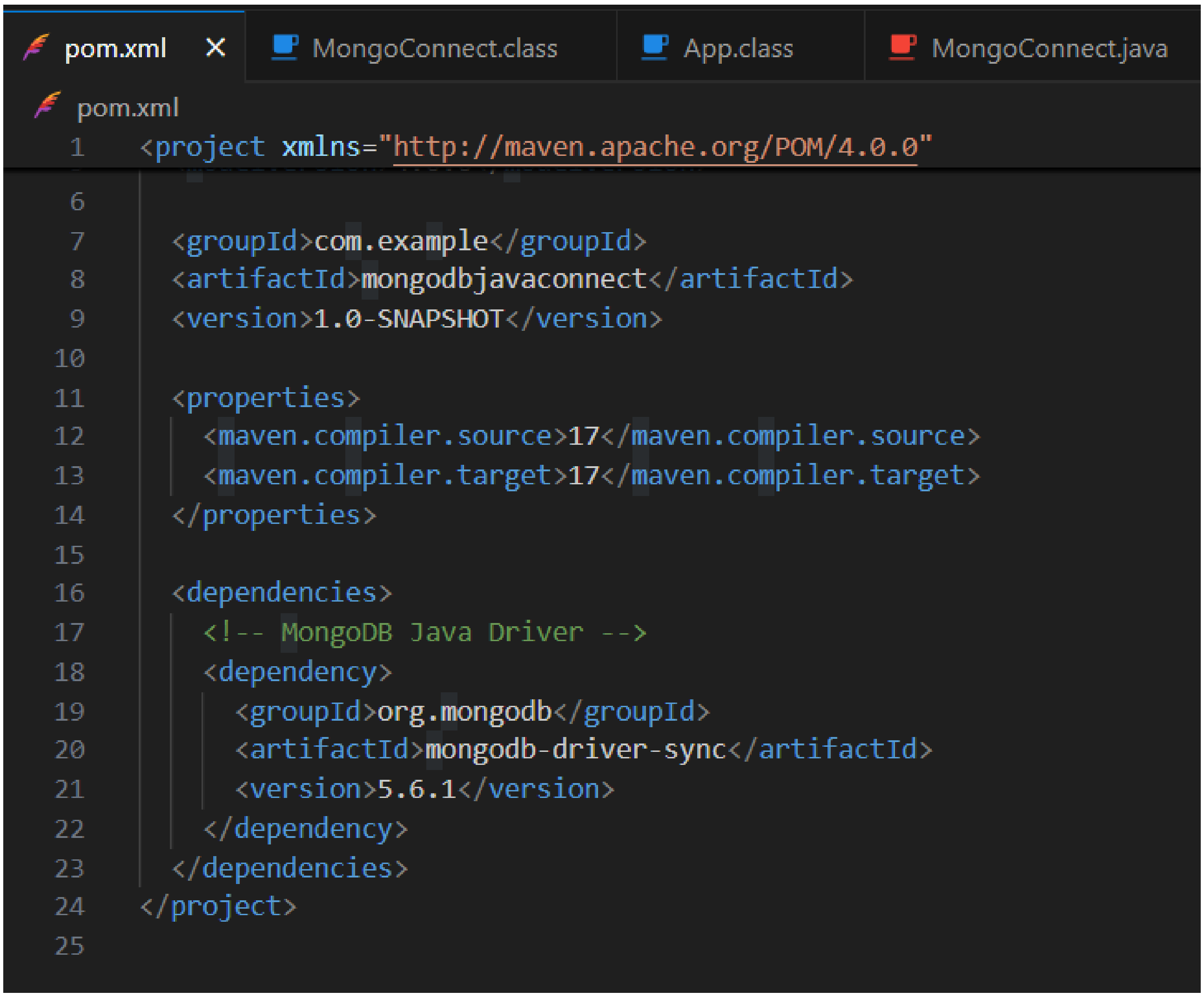


```
Maven home: D:\MISHALMCA\SEM3\NOSQL\apache-maven-3.9.11
Java version: 17.0.12, vendor: Oracle Corporation, runtime: C:\Program Files\Java\jdk-17
Default locale: en_IN, platform encoding: Cp1252
OS name: "windows 11", version: "10.0", arch: "amd64", family: "windows"

D:\MISHALMCA\SEM3\NOSQL\EXP11\mongodbjavaconnect>
```

Database Configuration

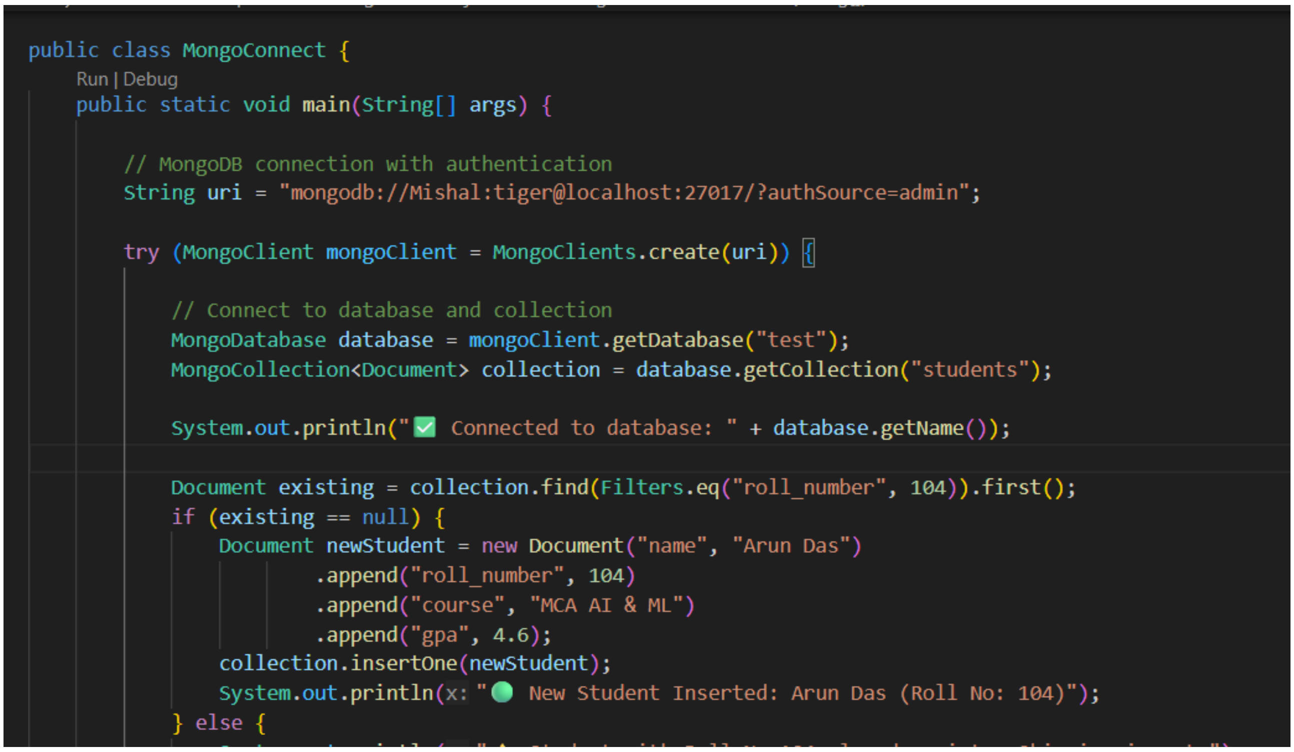
Maven Project Setup



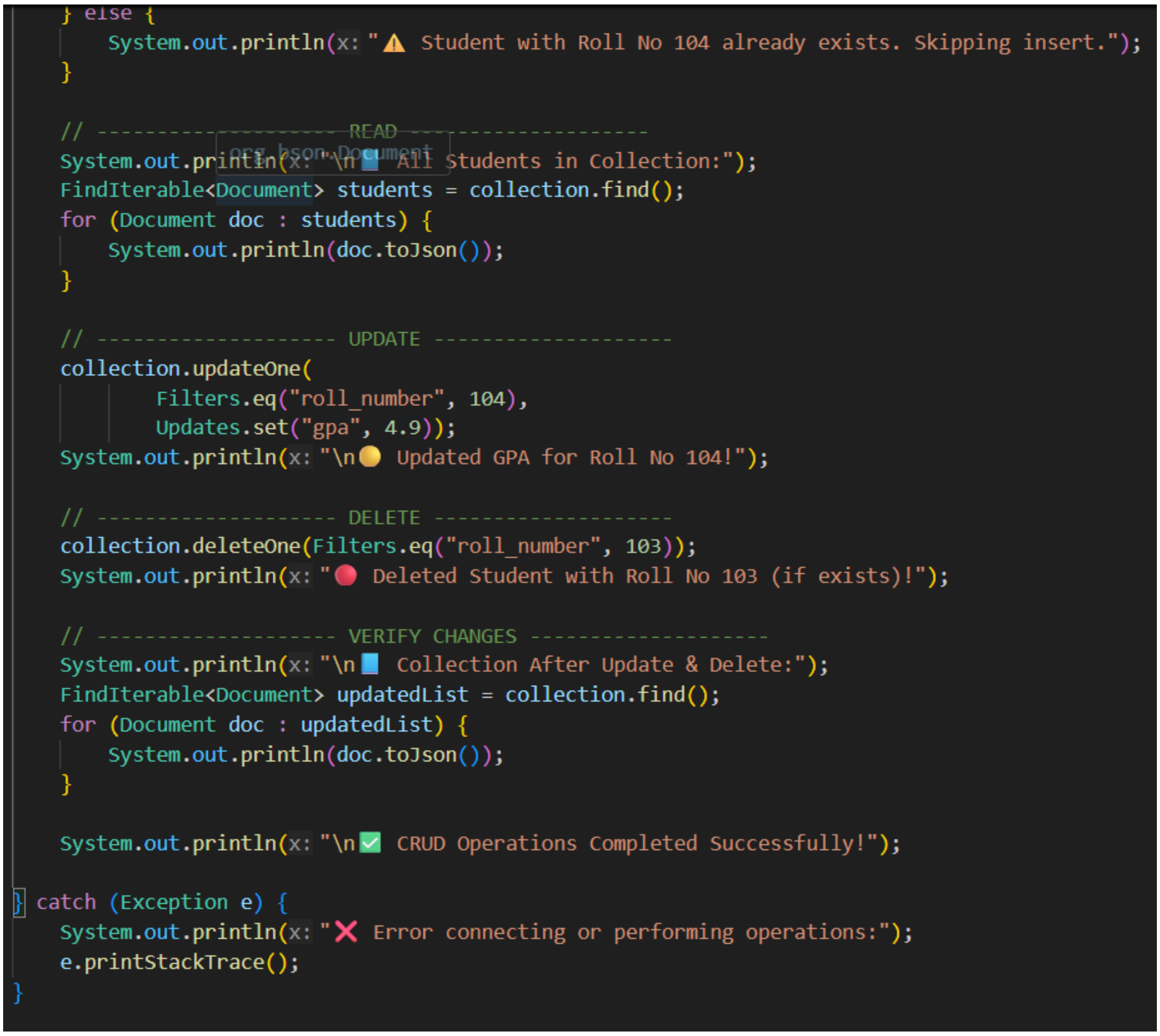


```xml
<project xmlns="http://maven.apache.org/POM/4.0.0"

  <groupId>com.example</groupId>
  <artifactId>mongodbjavaconnect</artifactId>
  <version>1.0-SNAPSHOT</version>

  <properties>
    <maven.compiler.source>17</maven.compiler.source>
    <maven.compiler.target>17</maven.compiler.target>
  </properties>

  <dependencies>
    <!-- MongoDB Java Driver -->
    <dependency>
      <groupId>org.mongodb</groupId>
      <artifactId>mongodb-driver-sync</artifactId>
      <version>5.6.1</version>
    </dependency>
  </dependencies>
</project>
```

Java Program Development

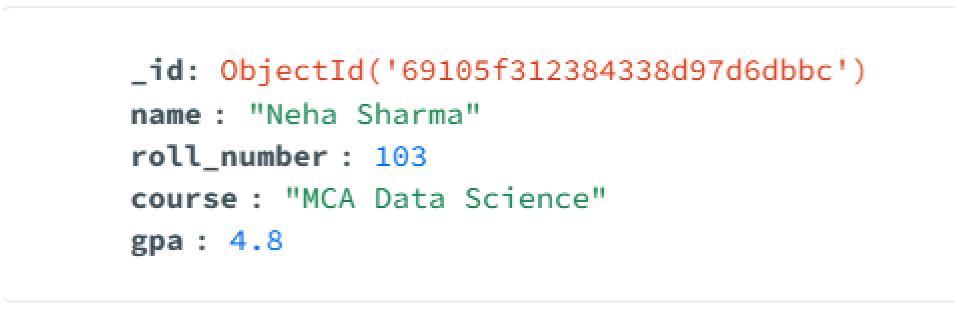


```java
public class MongoConnect {
    Run | Debug
    public static void main(String[] args) {

        // MongoDB connection with authentication
        String uri = "mongodb://Mishal:tiger@localhost:27017/?authSource=admin";

        try (MongoClient mongoClient = MongoClients.create(uri)) {

            // Connect to database and collection
            MongoDatabase database = mongoClient.getDatabase("test");
            MongoCollection<Document> collection = database.getCollection("students");

            System.out.println("✅ Connected to database: " + database.getName());

            Document existing = collection.find(Filters.eq("roll_number", 104)).first();
            if (existing == null) {
                Document newStudent = new Document("name", "Arun Das")
                        .append("roll_number", 104)
                        .append("course", "MCA AI & ML")
                        .append("gpa", 4.6);
                collection.insertOne(newStudent);
                System.out.println(x: "🟢 New Student Inserted: Arun Das (Roll No: 104)");
            } else {
```

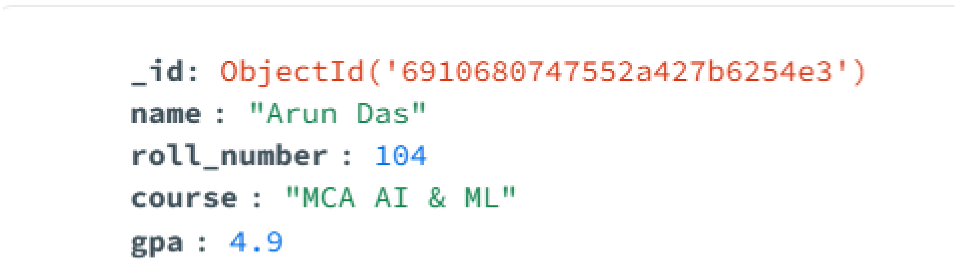


```java
    } else {
        System.out.println(x: "⚠ Student with Roll No 104 already exists. Skipping insert.");
    }

    // -------------------- READ --------------------
    System.out.println(x: "\n📘 All Students in Collection:");
    FindIterable<Document> students = collection.find();
    for (Document doc : students) {
        System.out.println(doc.toJson());
    }

    // -------------------- UPDATE --------------------
    collection.updateOne(
            Filters.eq("roll_number", 104),
            Updates.set("gpa", 4.9));
    System.out.println(x: "\n🟡 Updated GPA for Roll No 104!");

    // -------------------- DELETE --------------------
    collection.deleteOne(Filters.eq("roll_number", 103));
    System.out.println(x: "🔴 Deleted Student with Roll No 103 (if exists)!");

    // -------------------- VERIFY CHANGES --------------------
    System.out.println(x: "\n📘 Collection After Update & Delete:");
    FindIterable<Document> updatedList = collection.find();
    for (Document doc : updatedList) {
        System.out.println(doc.toJson());
    }

    System.out.println(x: "\n✅ CRUD Operations Completed Successfully!");

} catch (Exception e) {
    System.out.println(x: "❌ Error connecting or performing operations:");
    e.printStackTrace();
}
```

Verification



```
_id: ObjectId('69105f312384338d97d6dbbc')
name : "Neha Sharma"
roll_number : 103
course : "MCA Data Science"
gpa : 4.8
```



```
_id: ObjectId('6910680747552a427b6254e3')
name : "Arun Das"
roll_number : 104
course : "MCA AI & ML"
gpa : 4.9
```

PART 2 — PHP (Using XAMPP + MongoDB DLL)

AIM

To implement CRUD operations in PHP using MongoDB by configuring the MongoDB PHP extension (php_mongodb.dll), connecting PHP to MongoDB through XAMPP, and performing INSERT and READ operations on a MongoDB collection.

PROCEDURE

1. Environment Setup

1. Install **XAMPP** (PHP 8.2.12 version recommended).
2. Install **MongoDB Community Server** on Windows.
3. Install **MongoDB Compass** for GUI database visualization.
4. Verify PHP installation using:
5. php -v

2. Enabling MongoDB Support in PHP

1. Download the correct MongoDB DLL file (php_mongodb-1.19.x-8.2-ts-x64.dll from PECL).
2. Copy **php_mongodb.dll** into:
3. C:\xampp\php\ext\
4. Open:
5. C:\xampp\php\php.ini
6. Add:
7. extension=php_mongodb.dll
8. Restart **Apache** from XAMPP Control Panel.

3. Verifying MongoDB Extension

1. Create a file:
2. C:\xampp\htdocs\check_mongo.php
3. Add:
4. <?php phpinfo(); ?>

5. Open in browser:
6. http://localhost/check_mongo.php
7. Confirm the **mongodb** section appears.

4. Starting MongoDB Server

1. Open Command Prompt and run:
2. mongod
3. Ensure server starts successfully and listens on:
4. 127.0.0.1:27017

5. PHP Program Development

1. Create a file:
2. C:\xampp\htdocs\mongo_test.php
3. Write PHP code to:
   - Connect to MongoDB using MongoDB\Driver\Manager
   - Insert a document into studentDB → records collection
   - Retrieve and display all documents

6. Execution

1. Start Apache using XAMPP.
2. Open the PHP file in a browser:
3. http://localhost/mongo_test.php
4. Check the output for:
   - Successful connection message
   - Document insertion
   - Display of all records

7. Verification in MongoDB Compass

1. Open MongoDB Compass.
2. Connect using:
3. mongodb://localhost:27017

4. Expand:

5. studentDB → records

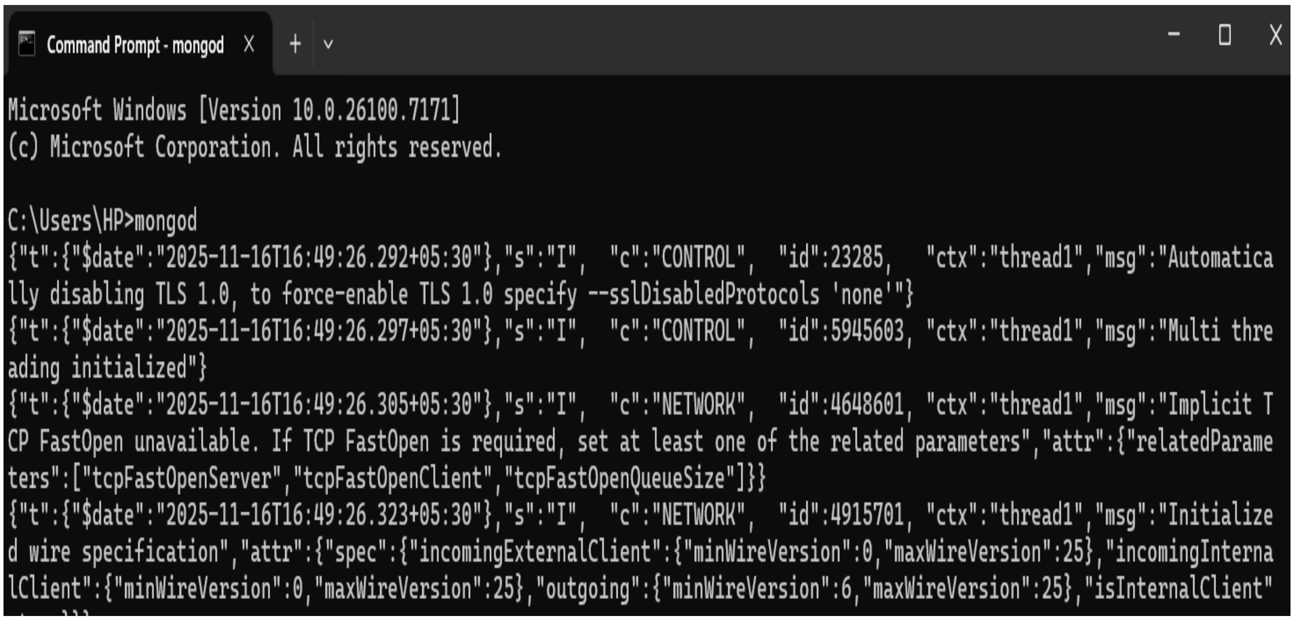
6. Click **Refresh** to confirm inserted documents appear.

8. Result

The PHP script successfully established a connection with MongoDB using the MongoDB DLL extension, performed document insertion, and displayed all records from the records collection. The operations were verified using MongoDB Compass.

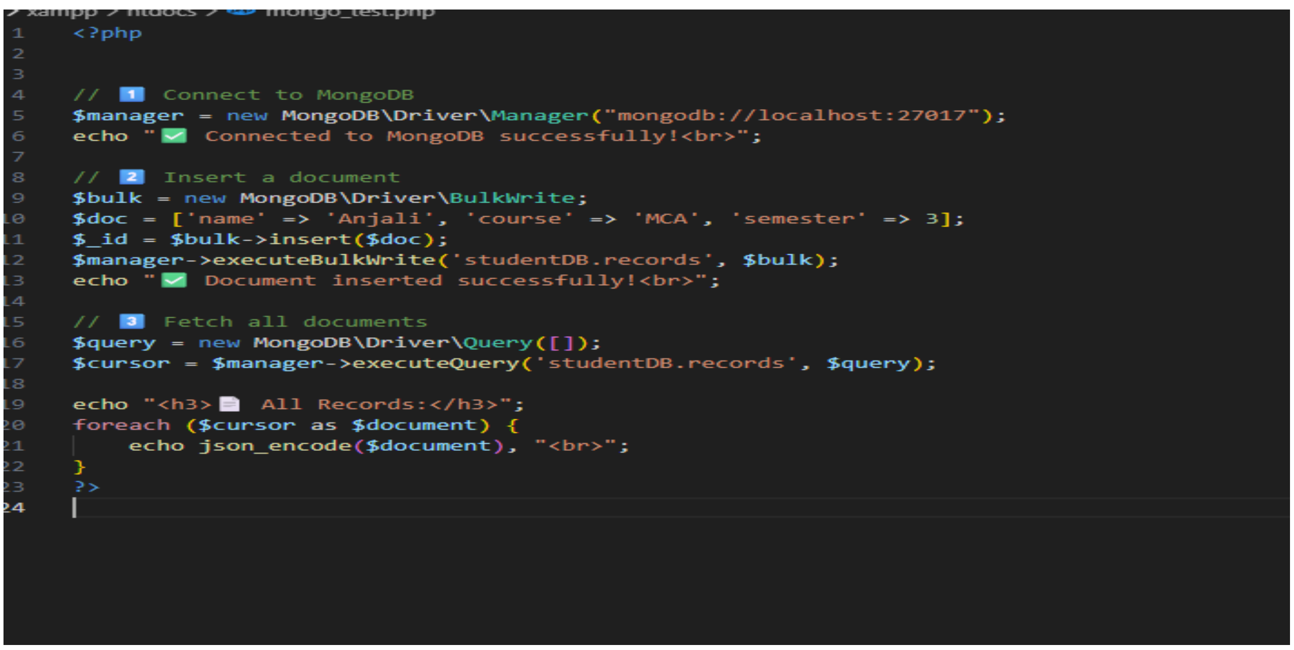
SCREENSHOTS:

MongoDB Server Running (mongod)



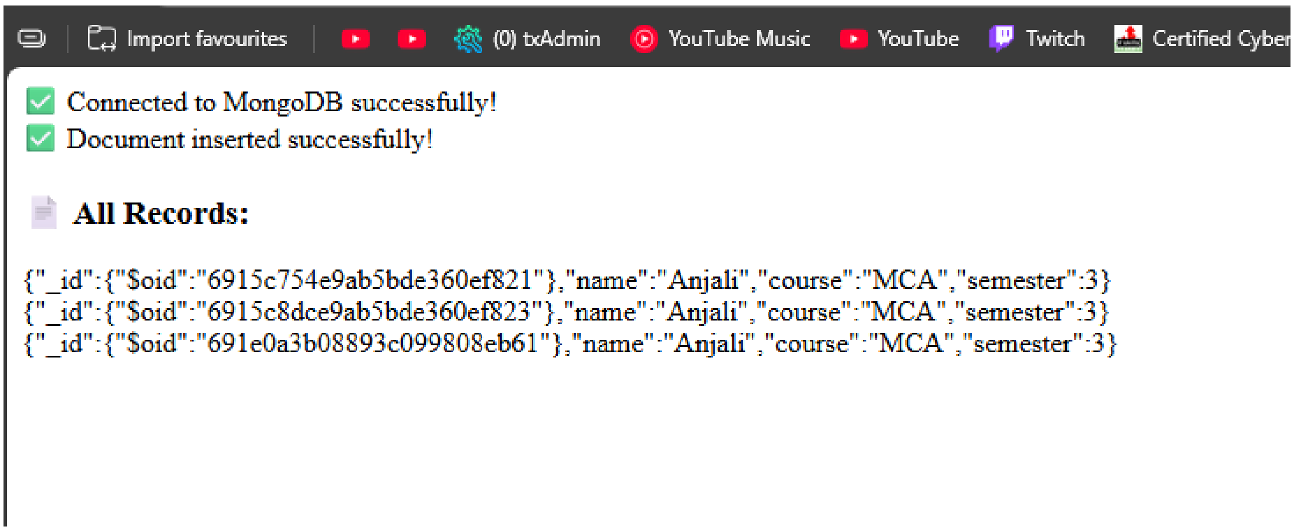
```
Command Prompt - mongod

Microsoft Windows [Version 10.0.26100.7171]
(c) Microsoft Corporation. All rights reserved.

C:\Users\HP>mongod
{"t":{"$date":"2025-11-16T16:49:26.292+05:30"},"s":"I",  "c":"CONTROL",  "id":23285,   "ctx":"thread1","msg":"Automatica
lly disabling TLS 1.0, to force-enable TLS 1.0 specify --sslDisabledProtocols 'none'"}
{"t":{"$date":"2025-11-16T16:49:26.297+05:30"},"s":"I",  "c":"CONTROL",  "id":5945603, "ctx":"thread1","msg":"Multi thre
ading initialized"}
{"t":{"$date":"2025-11-16T16:49:26.305+05:30"},"s":"I",  "c":"NETWORK",  "id":4648601, "ctx":"thread1","msg":"Implicit T
CP FastOpen unavailable. If TCP FastOpen is required, set at least one of the related parameters","attr":{"relatedParame
ters":["tcpFastOpenServer","tcpFastOpenClient","tcpFastOpenQueueSize"]}}
{"t":{"$date":"2025-11-16T16:49:26.323+05:30"},"s":"I",  "c":"NETWORK",  "id":4915701, "ctx":"thread1","msg":"Initialize
d wire specification","attr":{"spec":{"incomingExternalClient":{"minWireVersion":0,"maxWireVersion":25},"incomingInterna
lClient":{"minWireVersion":0,"maxWireVersion":25},"outgoing":{"minWireVersion":6,"maxWireVersion":25},"isInternalClient"
```

PHP Code for CRUD Operation



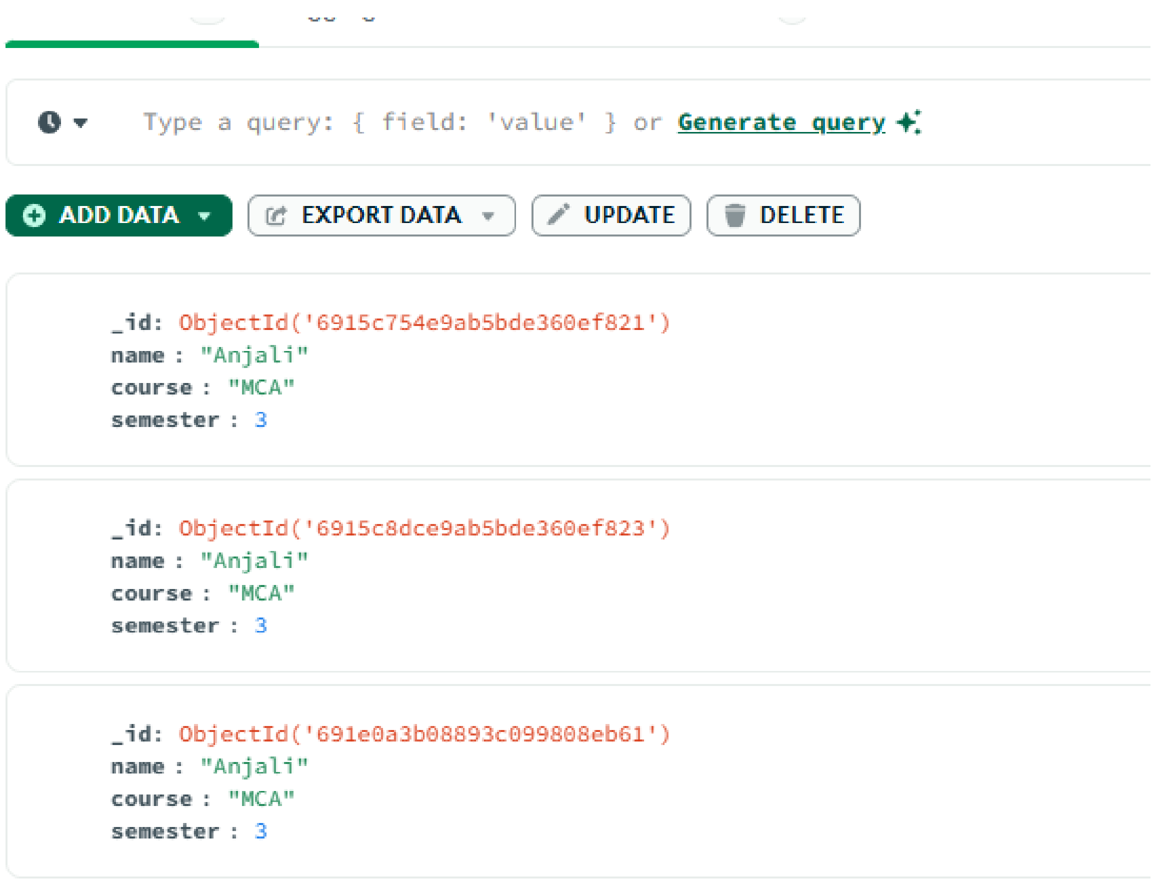
```php
> xampp > htdocs > mongo_test.php
<?php


// 1️⃣ Connect to MongoDB
$manager = new MongoDB\Driver\Manager("mongodb://localhost:27017");
echo "✅ Connected to MongoDB successfully!<br>";

// 2️⃣ Insert a document
$bulk = new MongoDB\Driver\BulkWrite;
$doc = ['name' => 'Anjali', 'course' => 'MCA', 'semester' => 3];
$_id = $bulk->insert($doc);
$manager->executeBulkWrite('studentDB.records', $bulk);
echo "✅ Document inserted successfully!<br>";

// 3️⃣ Fetch all documents
$query = new MongoDB\Driver\Query([]);
$cursor = $manager->executeQuery('studentDB.records', $query);

echo "<h3>📄 All Records:</h3>";
foreach ($cursor as $document) {
    echo json_encode($document), "<br>";
}
?>

```

Browser Output of PHP Script



```
✅ Connected to MongoDB successfully!
✅ Document inserted successfully!

📄 All Records:

{"_id":{"$oid":"6915c754e9ab5bde360ef821"},"name":"Anjali","course":"MCA","semester":3}
{"_id":{"$oid":"6915c8dce9ab5bde360ef823"},"name":"Anjali","course":"MCA","semester":3}
{"_id":{"$oid":"691e0a3b08893c099808eb61"},"name":"Anjali","course":"MCA","semester":3}
```

Compass Showing studentDB → records Collection



```
_id: ObjectId('6915c754e9ab5bde360ef821')
name : "Anjali"
course : "MCA"
semester : 3

_id: ObjectId('6915c8dce9ab5bde360ef823')
name : "Anjali"
course : "MCA"
semester : 3

_id: ObjectId('691e0a3b08893c099808eb61')
name : "Anjali"
course : "MCA"
semester : 3
```