MongoDB installation , Architecture, Commands

1. Introduction:

MongoDB is a NoSQL database designed for high performance, scalability, and flexibility. Instead of tables and rows (used in relational databases), MongoDB stores data in collections and documents using a JSON-like format called BSON.

2. MongoDB Installation (on Windows)

Step-by-step Installation:

1. Download MongoDB:

- Visit: https://www.mongodb.com/try/download/community
- Choose Windows, select MSI Installer, and download.

2. Install MongoDB:

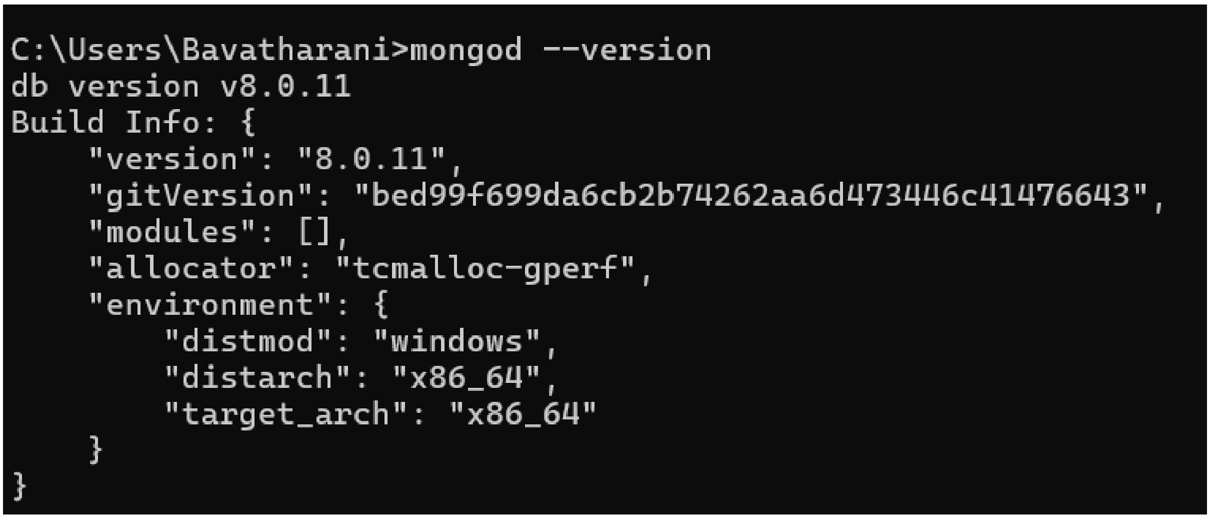
- Run the .msi installer.
- Choose Complete setup.
- Ensure the "Install MongoDB Compass" option is checked.

3. Set Environment Variables:

- Add this path to System Environment Variables > Path:
- C:\Program Files\MongoDB\Server\8.0\bin

4. Start MongoDB:

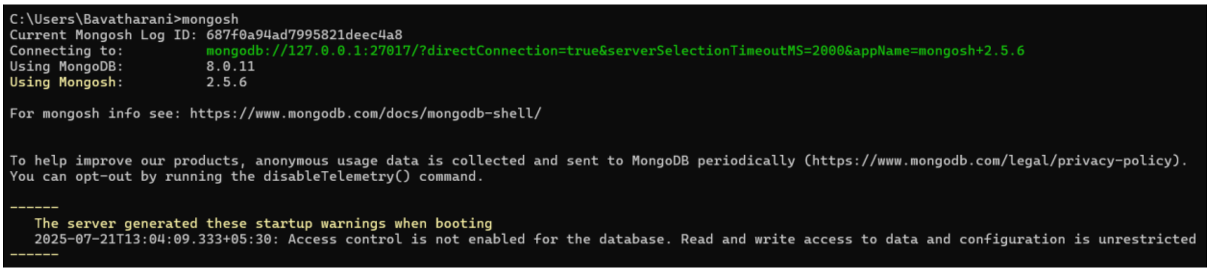
- Open Command Prompt and run:
- Mongod –version



```
C:\Users\Bavatharani>mongod --version
db version v8.0.11
Build Info: {
    "version": "8.0.11",
    "gitVersion": "bed99f699da6cb2b74262aa6d473446c41476643",
    "modules": [],
    "allocator": "tcmalloc-gperf",
    "environment": {
        "distmod": "windows",
        "distarch": "x86_64",
        "target_arch": "x86_64"
    }
}
```

5. Open Mongo Shell:

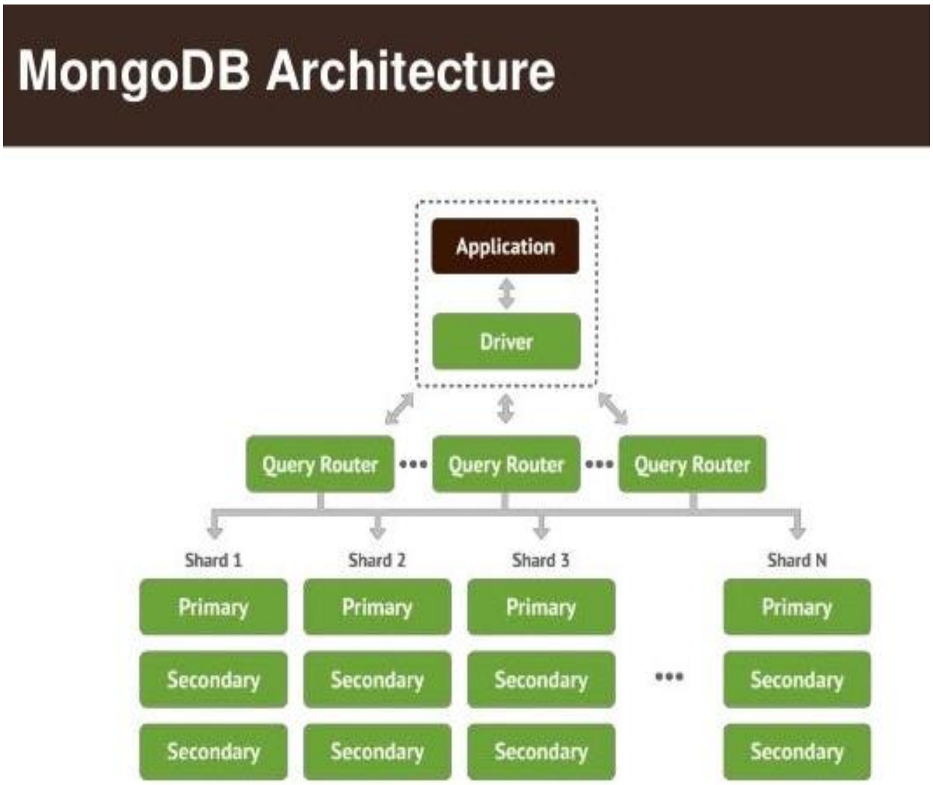
- In another command prompt:
- Mongosh



```
C:\Users\Bavatharani>mongosh
Current Mongosh Log ID: 687f0a94ad7995821deec4a8
Connecting to:          mongodb://127.0.0.1:27017/?directConnection=true&serverSelectionTimeoutMS=2000&appName=mongosh+2.5.6
Using MongoDB:          8.0.11
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
   2025-07-21T13:04:09.333+05:30: Access control is not enabled for the database. Read and write access to data and configuration is unrestricted
------
```

3. MongoDB Architecture

MongoDB is a NoSQL, document-oriented database that stores data in flexible, JSON-like documents. It is designed for high availability, scalability, and performance. MongoDB’s architecture supports both standalone and distributed deployments, including replica sets and sharded clusters.



1. Client

- The user or application that sends requests (read/write) to MongoDB.
- Communicates with the MongoDB server using drivers or mongosh.

2. mongod (MongoDB Server)

- The primary process that runs MongoDB.

- Handles CRUD operations, data storage, replication, and sharding.
- Runs in the background on your system.

3. Database

- A logical container for collections.
- Each server can host multiple databases (e.g., test, studentDB).

4. Collection

- A group of MongoDB documents (similar to a SQL table).
- Collections are schema-less and can store documents of varying structures.
- Examples: users, orders, products.

5. Document

- The smallest unit of data in MongoDB.
- Stored in JSON/BSON format.
- Example: { "name": "John", "age": 25, "city": "Chennai" }

6. Replication

- Ensures high availability via replica sets.
- One primary node handles writes; secondary nodes replicate data.
- Auto-failover: if the primary fails, a secondary is promoted.

7. Sharding

- MongoDB's method of horizontal scaling.
- Distributes data across shards (servers).
- Each shard handles part of the total dataset.

8. Storage Engine

- Manages how data is written to and read from disk.
- Common engines: WiredTiger (default), MMAPv1 (deprecated).
- Handles compression, caching, journaling, etc.

Summary

MongoDB’s architecture is built for scalability, flexibility, and fault tolerance. Unlike traditional RDBMS, it focuses on documents and collections rather than rows and tables.

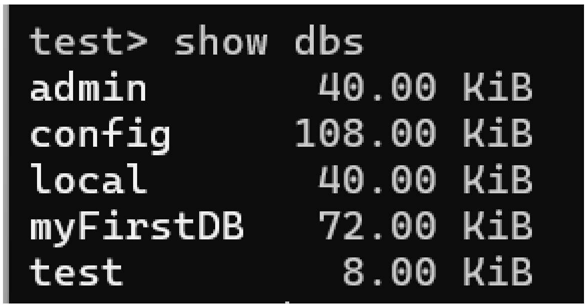
4. Uses of MongoDB

1. Content Management Systems (CMS)
    - Perfect for managing blogs, websites, or digital content.
    - Allows storing text, images, and metadata in a single document.
2. Real-Time Analytics
    - Supports fast reads and writes.
    - Used in dashboards, monitoring systems, and analytics platforms.
3. E-Commerce Platforms
    - Handles products, users, orders, inventory with flexible schema.
    - Enables product recommendation engines and customer activity tracking.
4. Social Networks
    - Used to store user profiles, feeds, messages, and media content.
    - Easily handles relationships and dynamic interactions.
5. Gaming Applications
    - Stores game states, user data, leaderboards, and in-game events.
    - Supports rapid reads/writes needed in gaming environments.
6. Internet of Things (IoT)
    - Collects and stores sensor data in real-time.
    - Scales horizontally to accommodate thousands of IoT devices.
7. Big Data Applications
    - Handles large volumes of unstructured or semi-structured data.
    - Commonly used in analytics, IoT, and real-time processing systems.

Commands

1. Create Database and Collection:

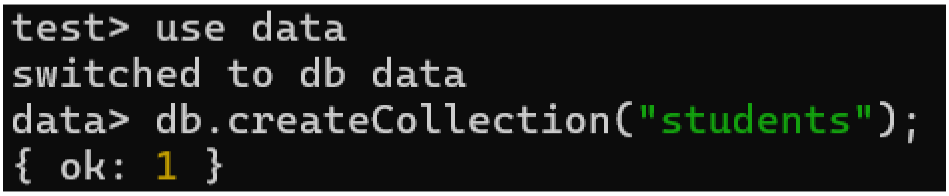
```
show dbs
```



```
test> show dbs
admin         40.00 KiB
config       108.00 KiB
local         40.00 KiB
myFirstDB     72.00 KiB
test           8.00 KiB
```

use data

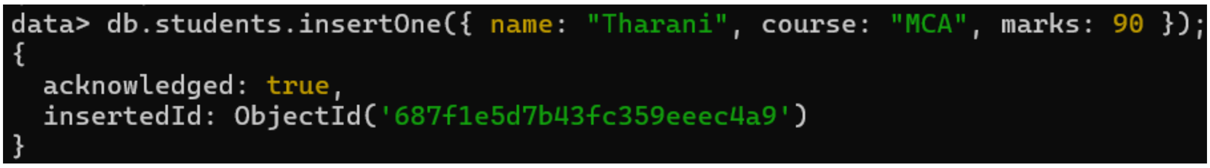
db.createCollection(“Students”);



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

2. Inserting Values:

db.students.insertOne({ name: "Tharani", course: "MCA", marks: 90 });



```
data> db.students.insertOne({ name: "Tharani", course: "MCA", marks: 90 });
{
  acknowledged: true,
  insertedId: ObjectId('687f1e5d7b43fc359eeec4a9')
}
```

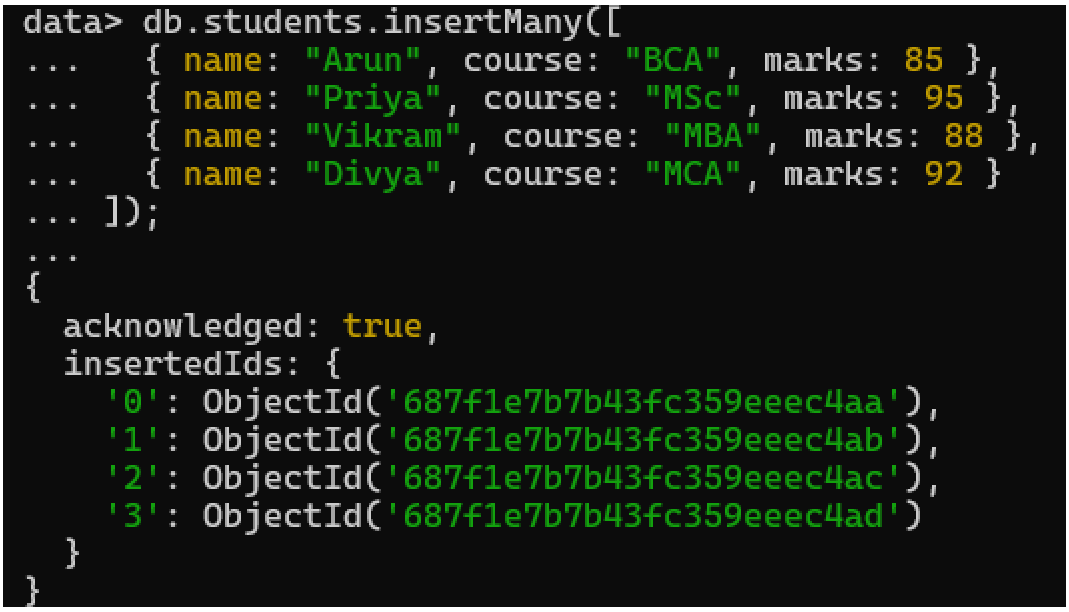
db.students.insertMany([

{ name: "Arun", course: "BCA", marks: 85 },

{ name: "Priya", course: "MSc", marks: 95 },

{ name: "Vikram", course: "MBA", marks: 88 },

{ name: "Divya", course: "MCA", marks: 92 }

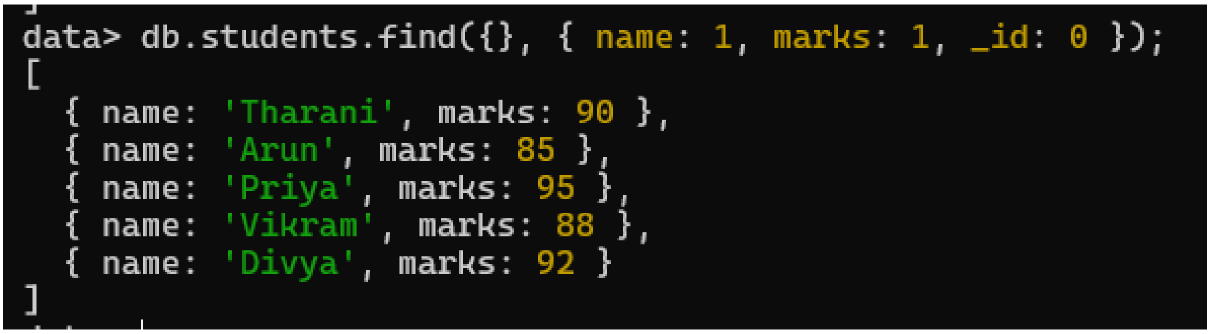


```
data> db.students.insertMany([
...   { name: "Arun", course: "BCA", marks: 85 },
...   { name: "Priya", course: "MSc", marks: 95 },
...   { name: "Vikram", course: "MBA", marks: 88 },
...   { name: "Divya", course: "MCA", marks: 92 }
... ]);
...
{
  acknowledged: true,
  insertedIds: {
    '0': ObjectId('687f1e7b7b43fc359eeec4aa'),
    '1': ObjectId('687f1e7b7b43fc359eeec4ab'),
    '2': ObjectId('687f1e7b7b43fc359eeec4ac'),
    '3': ObjectId('687f1e7b7b43fc359eeec4ad')
  }
}
```

]

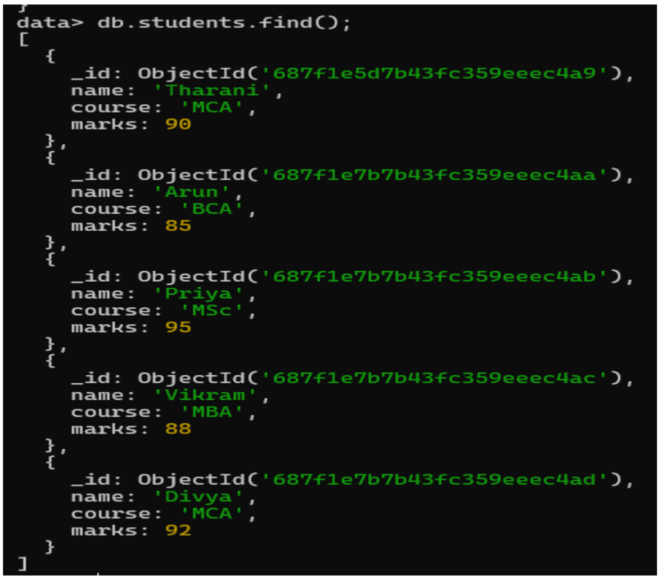
3. Read Documents:

db.students.find({}, { name: 1, marks: 1, _id: 0 });



```
data> db.students.find({}, { name: 1, marks: 1, _id: 0 });
[
  { name: 'Tharani', marks: 90 },
  { name: 'Arun', marks: 85 },
  { name: 'Priya', marks: 95 },
  { name: 'Vikram', marks: 88 },
  { name: 'Divya', marks: 92 }
]
```

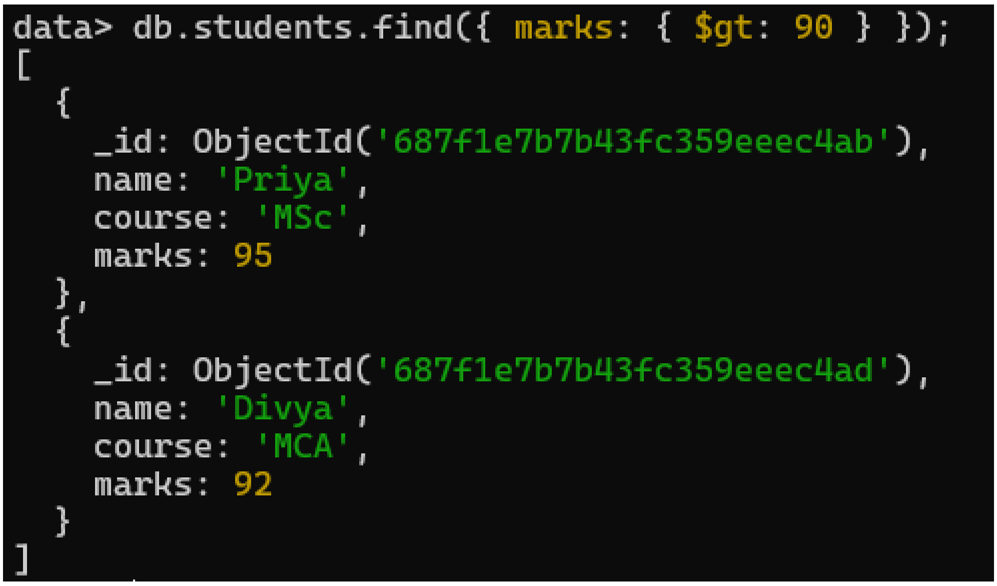
db.students.find();



```
data> db.students.find();
[
  {
    _id: ObjectId('687f1e5d7b43fc359eeec4a9'),
    name: 'Tharani',
    course: 'MCA',
    marks: 90
  },
  {
    _id: ObjectId('687f1e7b7b43fc359eeec4aa'),
    name: 'Arun',
    course: 'BCA',
    marks: 85
  },
  {
    _id: ObjectId('687f1e7b7b43fc359eeec4ab'),
    name: 'Priya',
    course: 'MSc',
    marks: 95
  },
  {
    _id: ObjectId('687f1e7b7b43fc359eeec4ac'),
    name: 'Vikram',
    course: 'MBA',
    marks: 88
  },
  {
    _id: ObjectId('687f1e7b7b43fc359eeec4ad'),
    name: 'Divya',
    course: 'MCA',
    marks: 92
  }
]
```

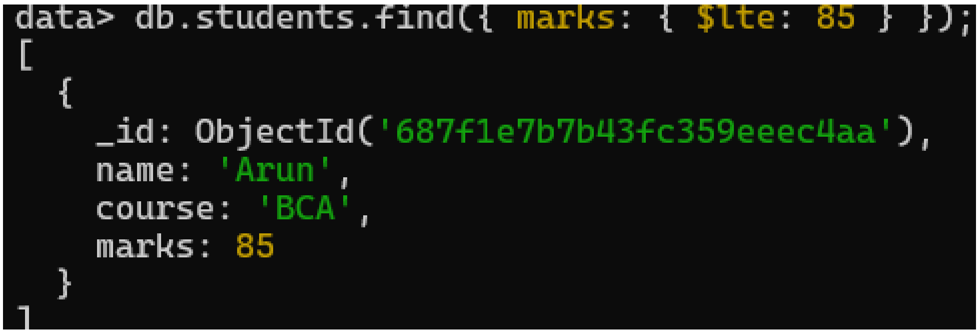
4. Filtering Conditions:

db.students.find({ marks: { $gt: 90 } });



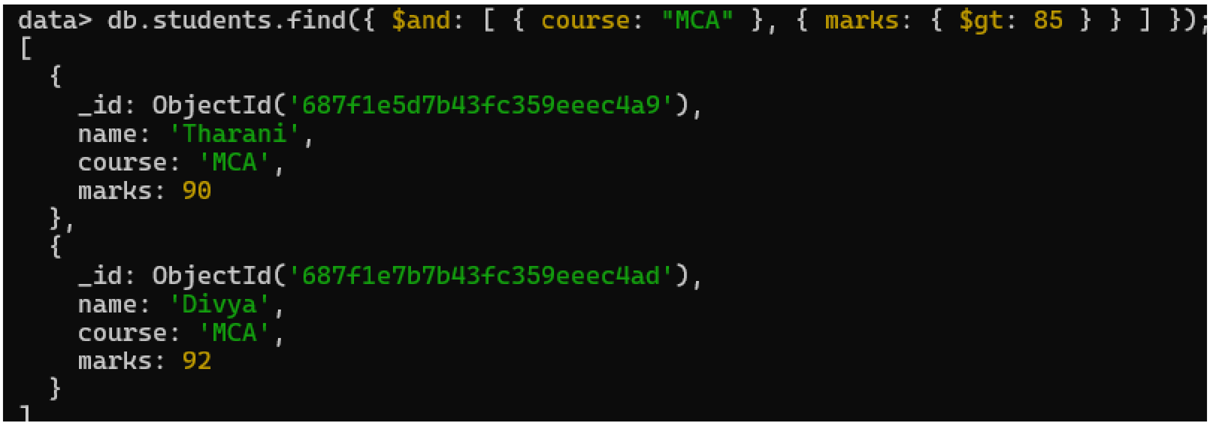
```
data> db.students.find({ marks: { $gt: 90 } });
[
  {
    _id: ObjectId('687f1e7b7b43fc359eeec4ab'),
    name: 'Priya',
    course: 'MSc',
    marks: 95
  },
  {
    _id: ObjectId('687f1e7b7b43fc359eeec4ad'),
    name: 'Divya',
    course: 'MCA',
    marks: 92
  }
]
```

db.students.find({ marks: { $lte: 85 } });



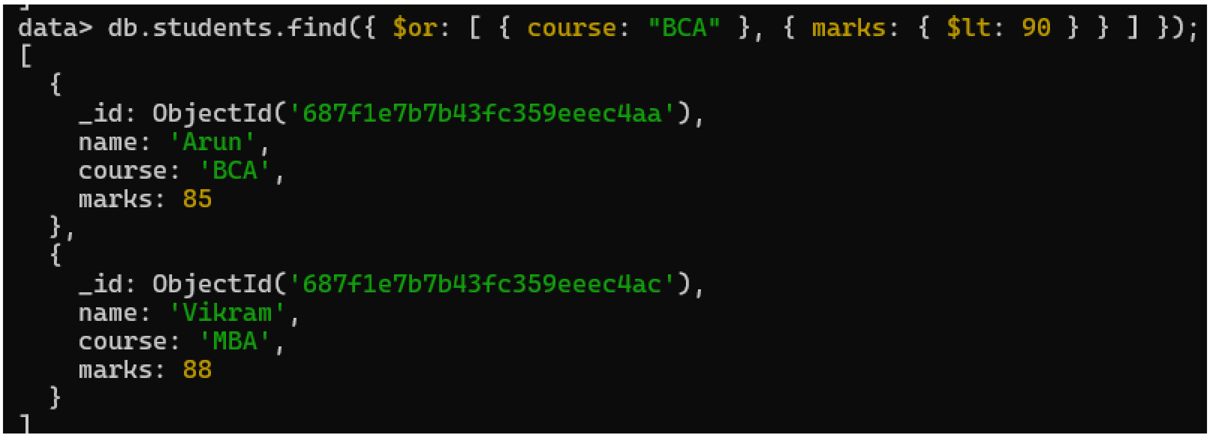
```
data> db.students.find({ marks: { $lte: 85 } });
[
  {
    _id: ObjectId('687f1e7b7b43fc359eeec4aa'),
    name: 'Arun',
    course: 'BCA',
    marks: 85
  }
]
```

db.students.find({ $and: [{ course: "MCA" }, { marks: { $gt: 85 } }] });



```
data> db.students.find({ $and: [ { course: "MCA" }, { marks: { $gt: 85 } } ] });
[
  {
    _id: ObjectId('687f1e5d7b43fc359eeec4a9'),
    name: 'Tharani',
    course: 'MCA',
    marks: 90
  },
  {
    _id: ObjectId('687f1e7b7b43fc359eeec4ad'),
    name: 'Divya',
    course: 'MCA',
    marks: 92
  }
]
```

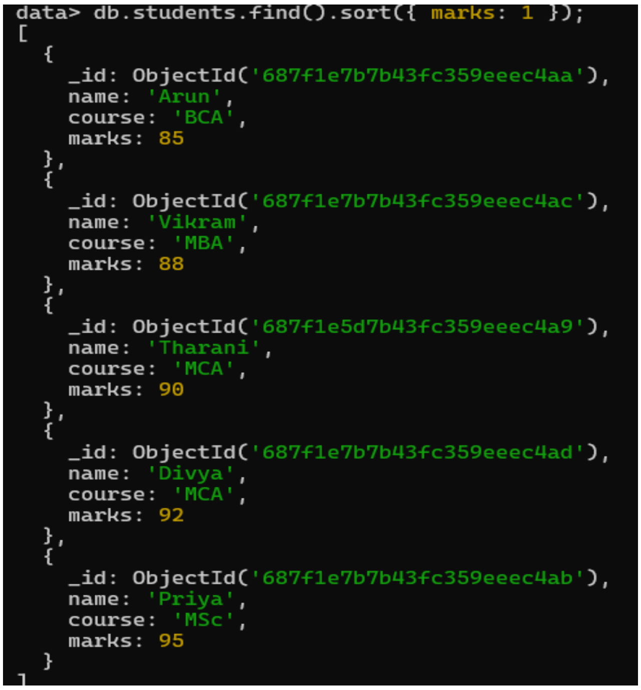
db.students.find({ $or: [{ course: "BCA" }, { marks: { $lt: 90 } }] });



```
data> db.students.find({ $or: [ { course: "BCA" }, { marks: { $lt: 90 } } ] });
[
  {
    _id: ObjectId('687f1e7b7b43fc359eeec4aa'),
    name: 'Arun',
    course: 'BCA',
    marks: 85
  },
  {
    _id: ObjectId('687f1e7b7b43fc359eeec4ac'),
    name: 'Vikram',
    course: 'MBA',
    marks: 88
  }
]
```

5.Sorting:

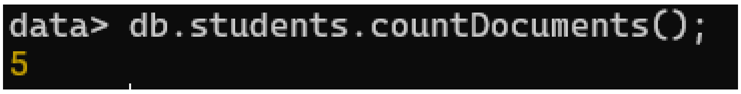
db.students.find().sort({ marks: 1 });



```
data> db.students.find().sort({ marks: 1 });
[
  {
    _id: ObjectId('687f1e7b7b43fc359eeec4aa'),
    name: 'Arun',
    course: 'BCA',
    marks: 85
  },
  {
    _id: ObjectId('687f1e7b7b43fc359eeec4ac'),
    name: 'Vikram',
    course: 'MBA',
    marks: 88
  },
  {
    _id: ObjectId('687f1e5d7b43fc359eeec4a9'),
    name: 'Tharani',
    course: 'MCA',
    marks: 90
  },
  {
    _id: ObjectId('687f1e7b7b43fc359eeec4ad'),
    name: 'Divya',
    course: 'MCA',
    marks: 92
  },
  {
    _id: ObjectId('687f1e7b7b43fc359eeec4ab'),
    name: 'Priya',
    course: 'MSc',
    marks: 95
  }
]
```

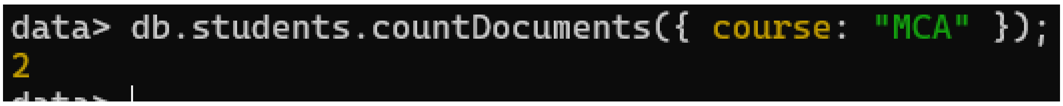
6. Counting:

db.students.countDocuments();



```
data> db.students.countDocuments();
5
```

db.students.countDocuments({ course: "MCA" });

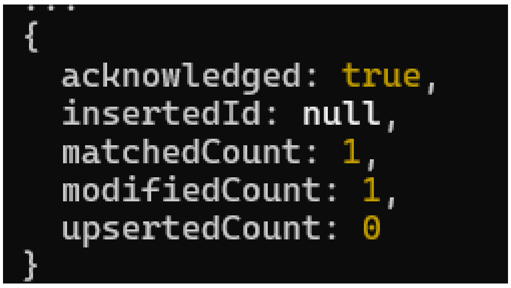


```
data> db.students.countDocuments({ course: "MCA" });
2
data>
```

7.Updating:

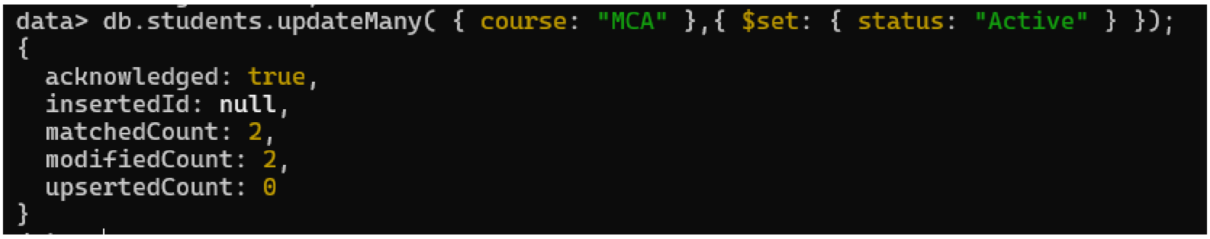
db.students.updateOne(

{ name: "Tharani" },{ $set: { marks: 95 } });



```
{
  acknowledged: true,
  insertedId: null,
  matchedCount: 1,
  modifiedCount: 1,
  upsertedCount: 0
}
```

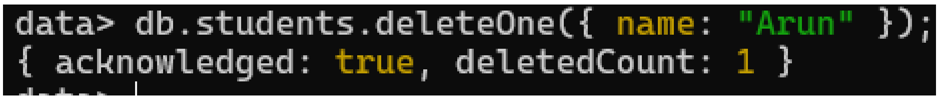
db.students.updateMany({ course: "MCA" },{ $set: { status: "Active" } });



```
data> db.students.updateMany( { course: "MCA" },{ $set: { status: "Active" } });
{
  acknowledged: true,
  insertedId: null,
  matchedCount: 2,
  modifiedCount: 2,
  upsertedCount: 0
}
```

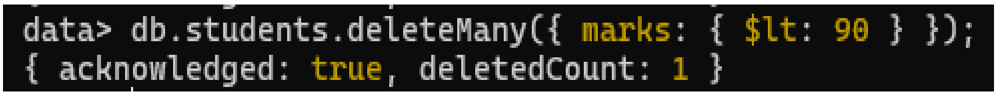
8. Delete:

db.students.deleteOne({ name: "Arun" });



```
data> db.students.deleteOne({ name: "Arun" });
{ acknowledged: true, deletedCount: 1 }
```

db.students.deleteMany({ marks: { $lt: 85 } });



```
data> db.students.deleteMany({ marks: { $lt: 90 } });
{ acknowledged: true, deletedCount: 1 }
```