21.7.25 Assignments

MongoDB: Installation, Architecture, Uses,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\harci>cd "C:\Program Files\MongoDB\Server\8.0\bin"

C:\Program Files\MongoDB\Server\8.0\bin>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:**

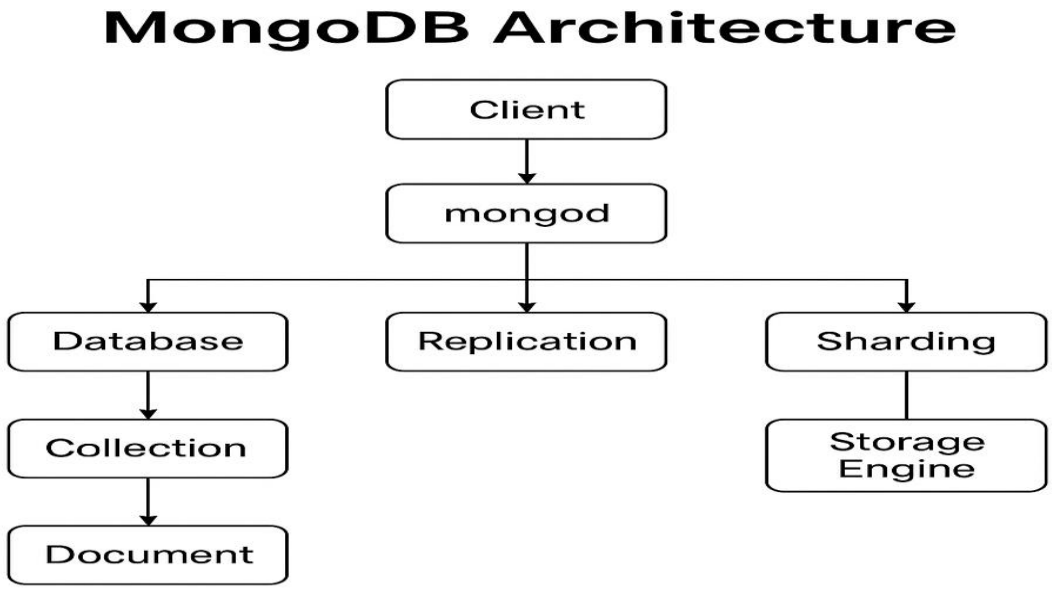
- In another command prompt:

  Mongosh

```
C:\Program Files\mongosh-2.5.6-win32-x64\mongosh-2.5.6-win32-x64\bin>mongosh
Current Mongosh Log ID: 687e03f08a147fb22deec4a8
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.co
m/legal/privacy-policy).
You can opt-out by running the disableTelemetry() command.

------
   The server generated these startup warnings when booting
   2025-07-21T12:26:49.120+05:30: Access control is not enabled for the database. Read and write access to data and conf
iguration 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 operations) to MongoDB.
- Communicates with the mongod server via drivers or the Mongo shell (mongosh).

2. mongod

- The primary MongoDB server process that handles:
  - CRUD operations (Create, Read, Update, Delete)
  - Data storage
  - Replication and sharding
- Every MongoDB server runs mongod in the background.

3. Database

- A logical container for collections.
- A MongoDB server can host multiple databases.
- Example: test, studentDB, ecommerceDB.

4. Collection

- A group of MongoDB documents, similar to a table in SQL.
- Collections do not require a predefined schema.
- Example: users, orders, products.

5. Document

- The smallest unit of data in MongoDB, stored in JSON/BSON format.
- Each document can have a different structure.
- Example: { name: "John", age: 25, city: "Chennai" }

6. Replication

- MongoDB supports replica sets to ensure high availability.
- A primary node handles writes, and secondary nodes replicate the data.
- If the primary fails, one secondary becomes the new primary.

7. Sharding

- A method for horizontal scaling.
- Data is split across multiple servers (shards) to handle huge datasets.
- Each shard handles a portion of the data.

8. Storage Engine

- The layer that interacts with the hardware to store data.
- Common engines include WiredTiger and MMAPv1.
- Manages how data is written to disk and retrieved.

Summary

MongoDB's architecture allows it to handle modern applications requiring scalability, fault tolerance, and flexible data modeling. It differs from traditional RDBMS by focusing on collections and documents rather than tables and rows.

4. Uses of MongoDB

1. **Content Management Systems (CMS)**
    - MongoDB stores flexible content like blogs, images, and media files, making it ideal for websites and publishing platforms.
2. **E-commerce Platforms**
    - Used for managing product catalogs, inventory, orders, and customer data with dynamic pricing and reviews.
3. **Mobile & Web Applications**
    - Perfect for apps with frequently changing data structures, such as messaging, delivery, or booking apps.

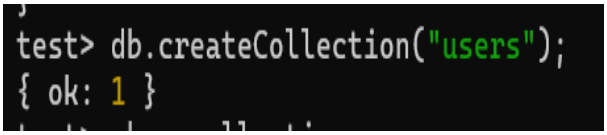
4. **Real-Time Analytics**
    - Processes data from IoT devices and applications in real-time for dashboards, monitoring, and quick decisions.
5. **Social Media Platforms**
    - Handles user profiles, posts, comments, and messages efficiently with schema flexibility.
6. **Internet of Things (IoT)**
    - Stores and analyzes sensor data from smart devices like home automation, health trackers, or connected vehicles.

COMMANDS

1. Collection Operations

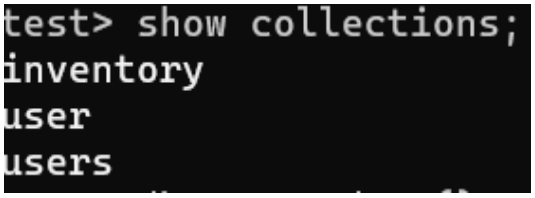
// Create a collection named "users"

```
db.createCollection("users");
```



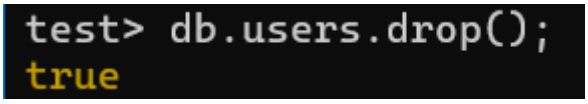
// Show all collections in the current database

```
show collections;
```



// Drop (delete) the collection named "users"

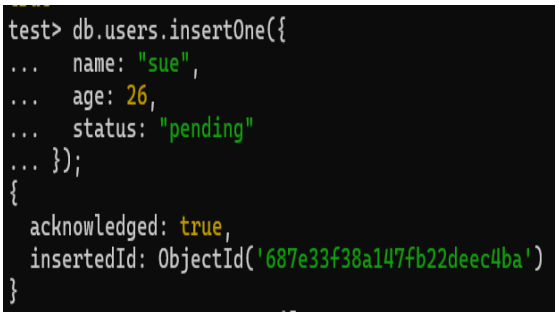
```
db.users.drop();
```



2. Create (Insert) Operations [C in CRUD]

// Insert a single document into the "users" collection

```
db.users.insertOne({
  name: "sue",
  age: 26,
  status: "pending"
});
```



// Insert multiple documents into "users"

```
db.users.insertMany([
  { name: "sue", age: 26, status: "pending" },
  { name: "niha", age: 35, status: "active" },
  { name: "preethi", age: 22, status: "inactive" },
  { name: "jhara", age: 30, status: "active" },
  { name: "durga", age: 28, status: "pending" }
]);
```

```
test> db.users.insertMany([
...   { name: "sue", age: 26, status: "pending" },
...   { name: "niha", age: 35, status: "active" },
...   { name: "preethi", age: 22, status: "inactive" },
...   { name: "jhara", age: 30, status: "active" },
...   { name: "durga", age: 28, status: "pending" }
... ]);
{
  acknowledged: true,
  insertedIds: {
    '0': ObjectId('687e34028a147fb22deec4bb'),
    '1': ObjectId('687e34028a147fb22deec4bc'),
    '2': ObjectId('687e34028a147fb22deec4bd'),
    '3': ObjectId('687e34028a147fb22deec4be'),
    '4': ObjectId('687e34028a147fb22deec4bf')
  }
}
```

// Insert multiple documents into "inventory"

```
db.inventory.insertMany([
  { item: "journal", qty: 25, tags: ["blank", "red"], size: { h: 14, w: 21 }, uom: "cm" },
  { item: "mat", qty: 85, tags: ["gray"], size: { h: 27.9, w: 35.5 }, uom: "cm" },
  { item: "mousepad", qty: 25, tags: ["gel", "blue"], size: { h: 19, w: 22.85 }, uom: "cm" }
]);
```

```
test> db.inventory.insertMany([
...   { item: "journal", qty: 25, tags: ["blank", "red"], size: { h: 14, w: 21 }, uom: "cm" },
...   { item: "mat", qty: 85, tags: ["gray"], size: { h: 27.9, w: 35.5 }, uom: "cm" },
...   { item: "mousepad", qty: 25, tags: ["gel", "blue"], size: { h: 19, w: 22.85 }, uom: "cm" }
... ]);
{
  acknowledged: true,
  insertedIds: {
    '0': ObjectId('687e340c8a147fb22deec4c0'),
    '1': ObjectId('687e340c8a147fb22deec4c1'),
    '2': ObjectId('687e340c8a147fb22deec4c2')
  }
}
```

3. Read (Query) Operations [R in CRUD]

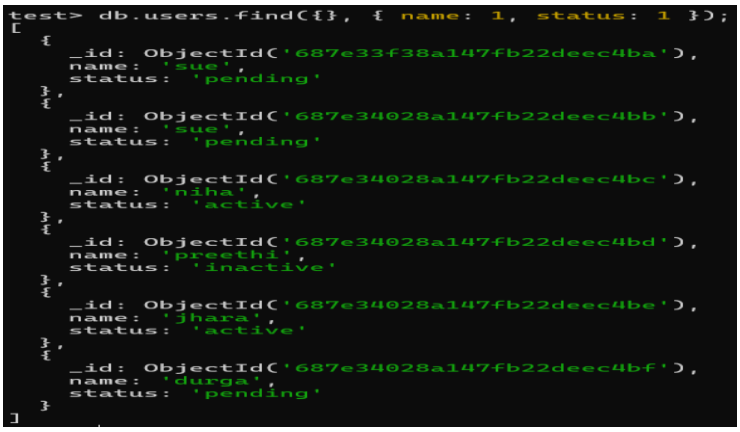
// Fetch all documents

```
db.users.find();
```

```
test> db.users.find();
[
  {
    _id: ObjectId('687e33f38a147fb22deec4ba'),
    name: 'sue',
    age: 26,
    status: 'pending'
  },
  {
    _id: ObjectId('687e34028a147fb22deec4bb'),
    name: 'sue',
    age: 26,
    status: 'pending'
  },
  {
    _id: ObjectId('687e34028a147fb22deec4bc'),
    name: 'niha',
    age: 35,
    status: 'active'
  },
  {
    _id: ObjectId('687e34028a147fb22deec4bd'),
    name: 'preethi',
    age: 22,
    status: 'inactive'
  },
  {
    _id: ObjectId('687e34028a147fb22deec4be'),
    name: 'jhara',
    age: 30,
    status: 'active'
  },
  {
    _id: ObjectId('687e34028a147fb22deec4bf'),
    name: 'durga',
    age: 28,
    status: 'pending'
  }
]
```

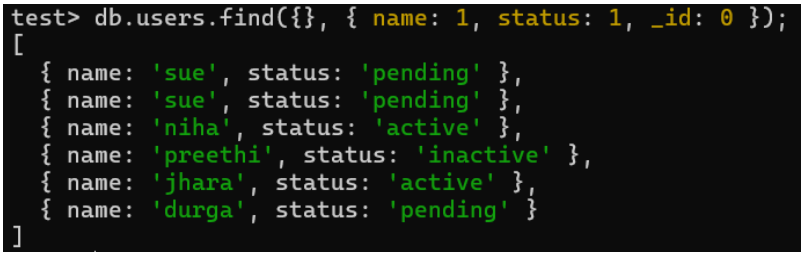
// Fetch name and status only (default includes _id)

db.users.find({}, { name: 1, status: 1 });



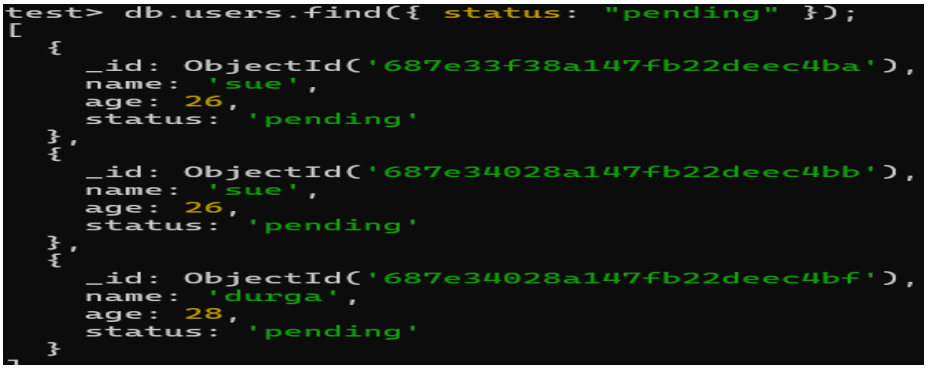
// Fetch name and status, exclude _id

db.users.find({}, { name: 1, status: 1, _id: 0 });



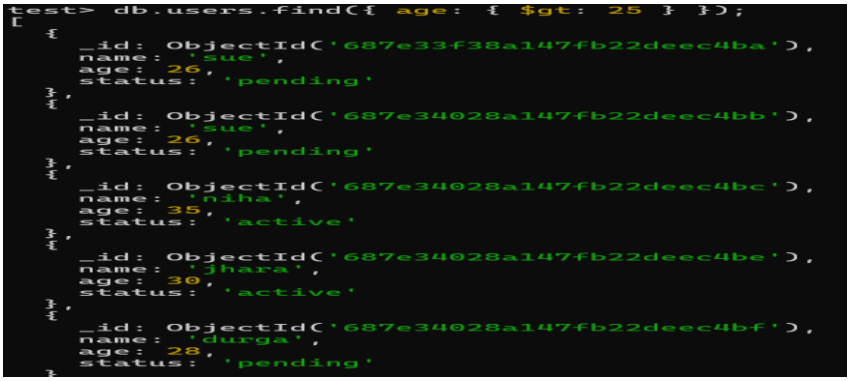
// Filter by status

db.users.find({ status: "pending" });



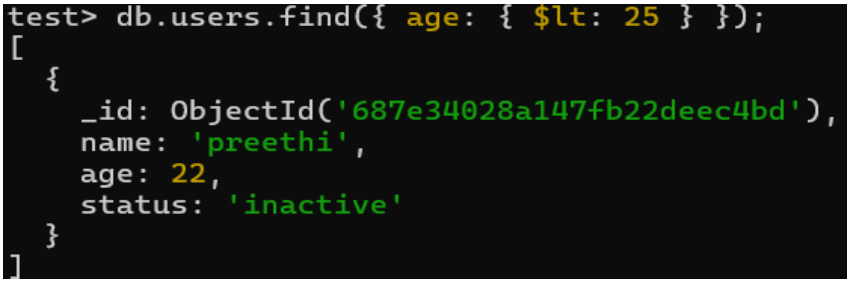
// Filter where age > 25

db.users.find({ age: { $gt: 25 } });



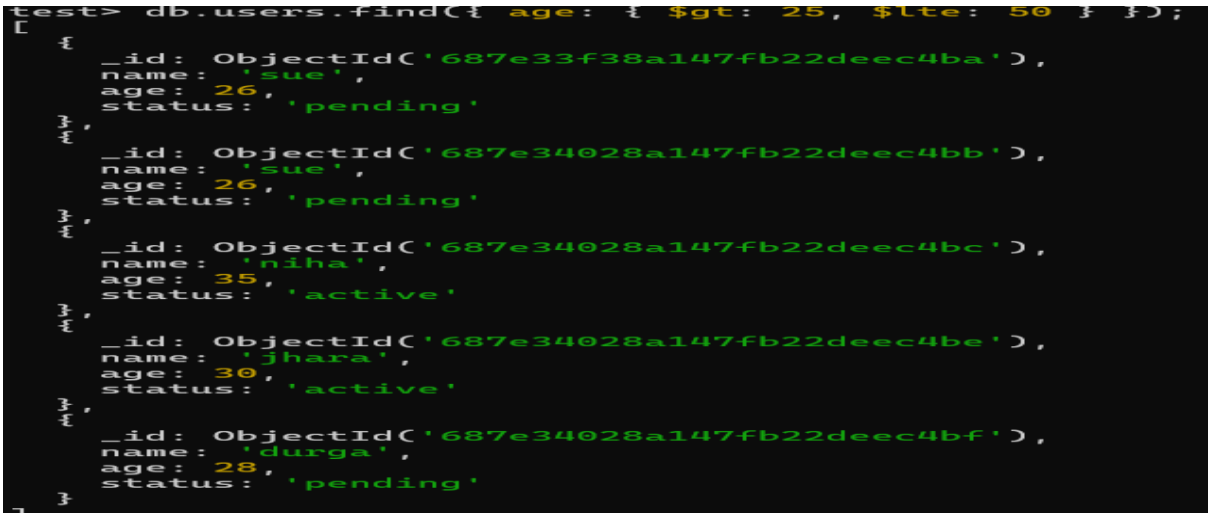
// Filter where age < 25

db.users.find({ age: { $lt: 25 } });



// Filter where age is between 25 and 50

db.users.find({ age: { $gt: 25, $lte: 50 } });



// Regex: Find user_id starting with "su" (if user_id exists)

db.users.find({ user_id: /^su/ });

db.users.find({ user_id: { $regex: /^su/ } });

```
test> db.users.find({ name: { $regex: /^su/ } });
[
  {
    _id: ObjectId('687e33f38a147fb22deec4ba'),
    name: 'sue',
    age: 26,
    status: 'pending'
  },
```

4. Filter + Projection + Logical Operators

// status not equal to "A"

db.users.find({ status: { $ne: "A" } });

```
test> db.users.find({ status: { $ne: "A" } });
[
  {
    _id: ObjectId('687e33f38a147fb22deec4ba'),
    name: 'sue',
    age: 26,
    status: 'pending'
  },
  {
    _id: ObjectId('687e34028a147fb22deec4bb'),
    name: 'sue',
    age: 26,
    status: 'pending'
  },
  {
    _id: ObjectId('687e34028a147fb22deec4bc'),
    name: 'niha',
    age: 35,
    status: 'active'
  },
  {
    _id: ObjectId('687e34028a147fb22deec4bd'),
    name: 'preethi',
    age: 22,
    status: 'inactive'
  },
  {
    _id: ObjectId('687e34028a147fb22deec4be'),
    name: 'jhara',
    age: 30,
    status: 'active'
  },
  {
    _id: ObjectId('687e34028a147fb22deec4bf'),
    name: 'durga',
    age: 28,
    status: 'pending'
  }
```

// status = "Active" AND age = 35

db.users.find({ status: "Active", age: 35 });

```
test> db.users.find({ status: "active", age: 35 });
[
  {
    _id: ObjectId('687e34028a147fb22deec4bc'),
    name: 'niha',
    age: 35,
    status: 'active'
  }
]
```

// status = "Active" OR age = 50

db.users.find({ $or: [{ status: "Active" }, { age: 50 }] });

```
test> db.users.find({ $or: [ { status: "active" }, { age: 50 } ] });
[
  {
    _id: ObjectId('687e34028a147fb22deec4bc'),
    name: 'niha',
    age: 35,
    status: 'active'
  },
  {
    _id: ObjectId('687e34028a147fb22deec4be'),
    name: 'jhara',
    age: 30,
    status: 'active'
  }
]
```

5. Sorting, Counting, Distinct

// Sort by user_id in ascending order

db.users.find({ status: "active" }).sort({ user_id: 1 });

```
test> db.users.find({ status: "active" }).sort({ name: 1 });
[
  {
    _id: ObjectId('687e34028a147fb22deec4be'),
    name: 'jhara',
    age: 30,
    status: 'active'
  },
  {
    _id: ObjectId('687e34028a147fb22deec4bc'),
    name: 'niha',
    age: 35,
    status: 'active'
  }
]
```

// Sort by user_id in descending order

db.users.find({ status: "active" }).sort({ user_id: -1 });

```
test> db.users.find({ status: "active" }).sort({ user_id: -1 });
[
  {
    _id: ObjectId('687e34028a147fb22deec4bc'),
    name: 'niha',
    age: 35,
    status: 'active'
  },
  {
    _id: ObjectId('687e34028a147fb22deec4be'),
    name: 'jhara',
    age: 30,
    status: 'active'
  }
]
```

// Count all documents

db.users.count();

db.users.find().count();

```
test> db.users.count();
... db.users.find().count();
6
test>
```

// Count documents where user_id exists

db.users.count({ user_id: { $exists: true } });

```
test> db.users.count({ user_id: { $exists: true } });
0
test>
```

// Count users with age > 30

db.users.count({ age: { $gt: 30 } });

```
test> db.users.count({ age: { $gt: 30 } });
1
```

// Show only distinct status values

db.users.distinct("status");

```
test> db.users.distinct("status");
[ 'active', 'inactive', 'pending' ]
```

6. Find Variants

// Find the first matching document

db.users.findOne();

```
test> db.users.findOne();
{
  _id: ObjectId('687e33f38a147fb22deec4ba'),
  name: 'sue',
  age: 26,
  status: 'pending'
}
```

// Limit to one result

db.users.find().limit(1);

```
test> db.users.find().limit(1);
[
  {
    _id: ObjectId('687e33f38a147fb22deec4ba'),
    name: 'sue',
    age: 26,
    status: 'pending'
  }
]
```

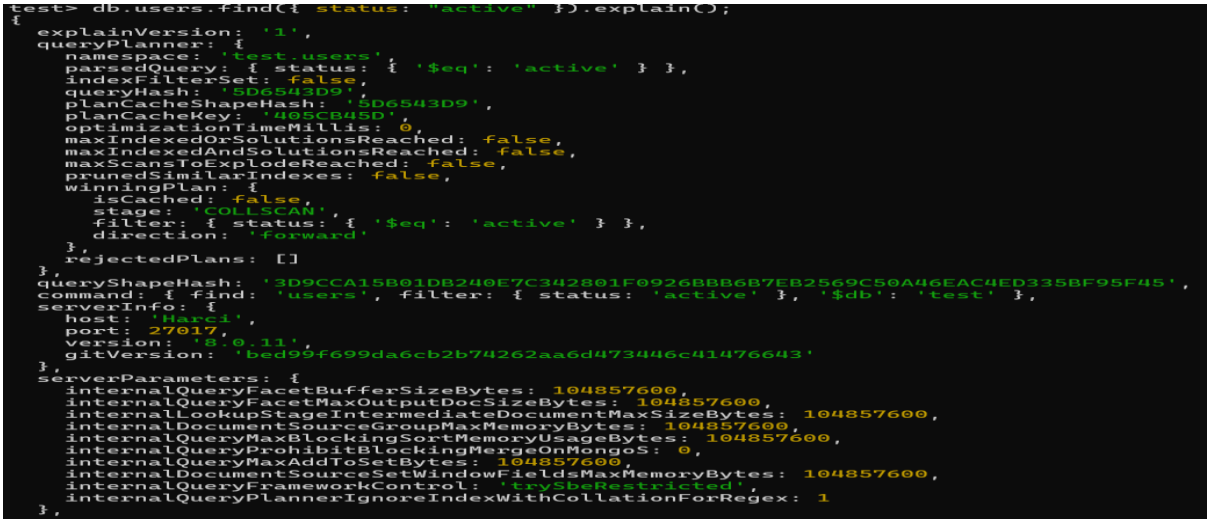
// Skip 5 and show next

db.users.find().limit(5).skip(5);

```
test> db.users.find().skip(5).limit(5);
[
  {
    _id: ObjectId('687e34028a147fb22deec4bf'),
    name: 'durga',
    age: 28,
    status: 'pending'
  }
```

// Show how MongoDB executes a query

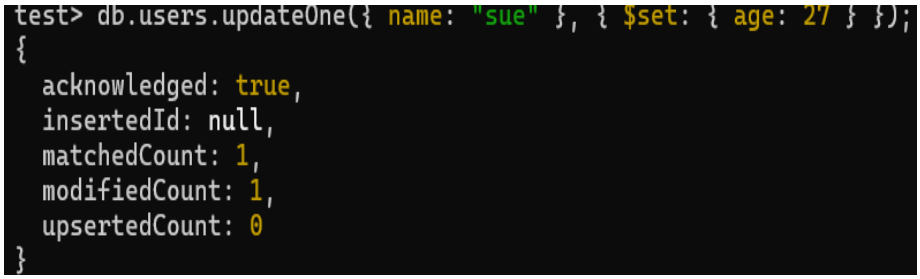
db.users.find({ status: "active" }).explain();



7. Update and Delete Operations [U and D in CRUD]

// Update a document (set new age for "sue")

db.users.updateOne({ name: "sue" }, { $set: { age: 27 } });



// Delete a document by name

db.users.deleteOne({ name: "sue" });