|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Experiment - 07** | | | | |
| **Date of Performance:** | **19/3/25** | | | |
| **Date of Submission:** | **9/4/25** | | | |
| Program Execution/  formation /  correction/  ethical practices (06) | Timely Submission (01) | Viva (03) | Experiment Total (10) | Sign with Date |
|  |  |  |  |  |

**EXPERIMENT - 07**

**1.1 Aim:** MongoDB Installation And CRUD Operations.

**1.2 Course Outcome:**Apply MongoDB for frontend and backend connectivity using REST API.

**1.3 Learning Objectives:**

**1.4 Requirement**: MongoDB Server, MongoDB Compass

**1.5 Related Theory:**

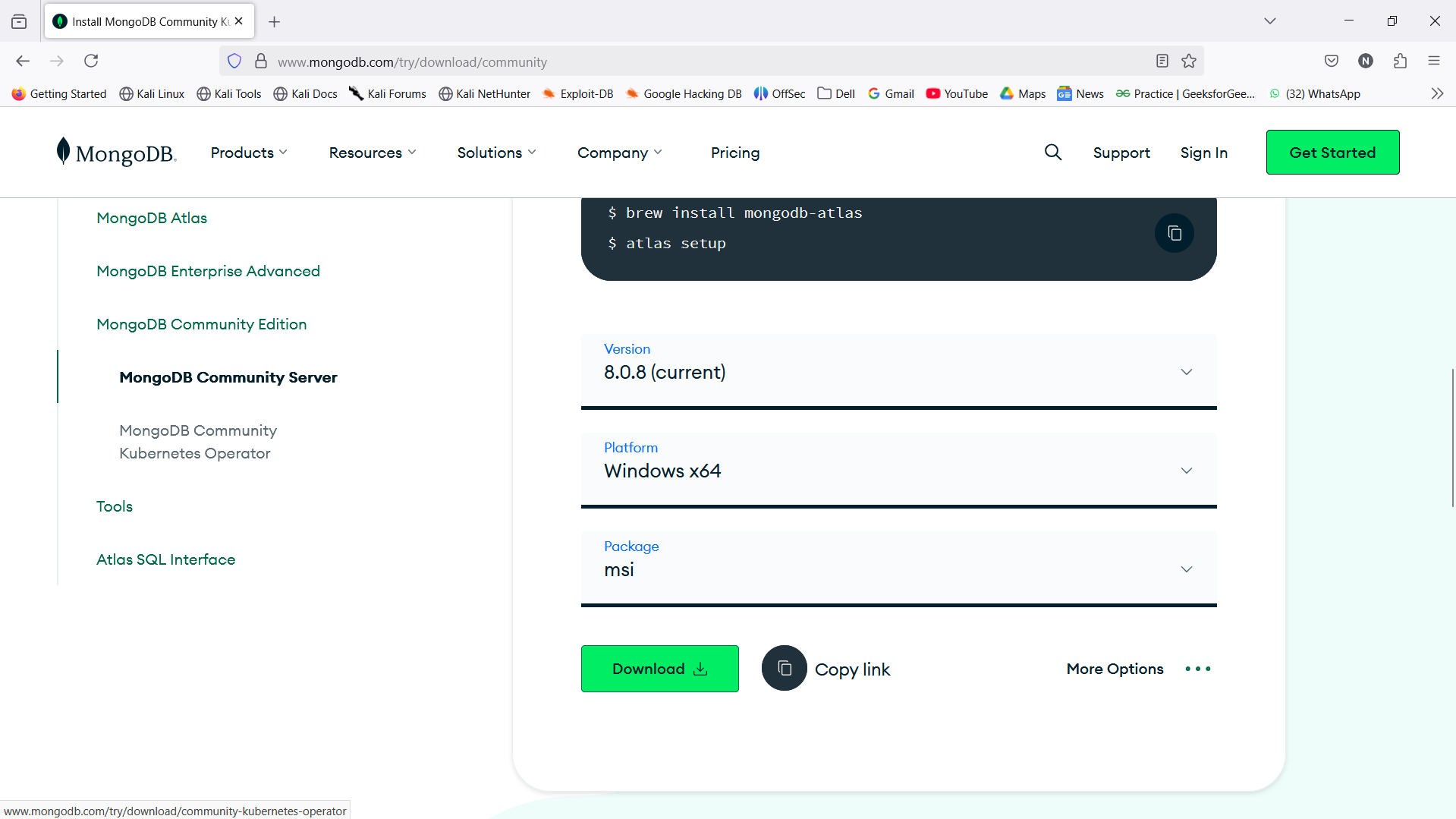
1. **What is MongoDB?**

**MongoDB** is a **NoSQL**, **document-oriented** database that stores data in **JSON-like BSON** (Binary JSON) format. Unlike traditional relational databases (RDBMS) that use tables and rows, MongoDB uses **collections and documents**.

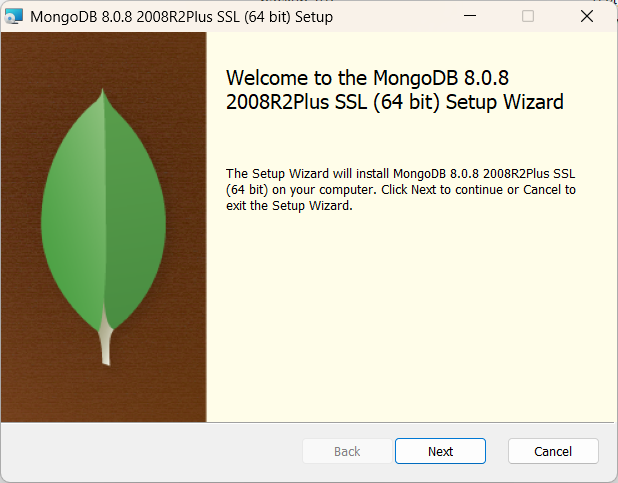
1. **Key Concepts in MongoDB**
   1. DataBase- A container for collections. Analogous to a database in RDBMS.
   2. Collection - A group of MongoDB documents, similar to a table in relational databases.
   3. Document - A record in a collection, stored in BSON format (binary-encoded JSON).
   4. Field - A key-value pair in a document (similar to a column in SQL).
   5. \_id - A unique identifier for each document (automatically generated if not provided)
2. **CRUD Operations in MongoDB**
   1. CRUD stands for **Create, Read, Update, Delete** — the four basic operations for persistent storage.
   2. MongoDB provides these operations via:
      1. Mongo Shell
      2. MongoDB Drivers (Node.js, Python, Java, etc.)
      3. Mongoose (ODM for Node.js)
      4. MongoDB Compass (GUI tool)
   3. Details of the Operations
      1. CREATE - Insert a new document into a collection.
      2. READ - Retrieve documents from a collection.
      3. UPDATE - Modify existing documents.
      4. DELETE - Remove documents from a collection.
3. **Schema Design Considerations**
   1. MongoDB is **schema-less**, but **schema design** is still important.
   2. Embed data when you have one-to-few relationships.
   3. Reference data for one-to-many or many-to-many relationships.

**1.6 Programs & Output:**

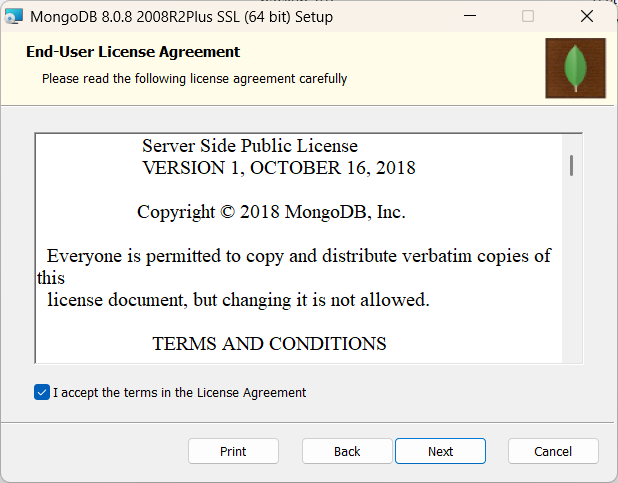
1. Step 1: Download MongoDB:
   1. Visit the official MongoDB download center: <https://www.mongodb.com/try/download/community>
   2. Select:
      1. **Version:** Latest Stable Release
      2. **OS:** Windows
      3. Package: .msi (Windows Installer)
   3. Click **Download**.



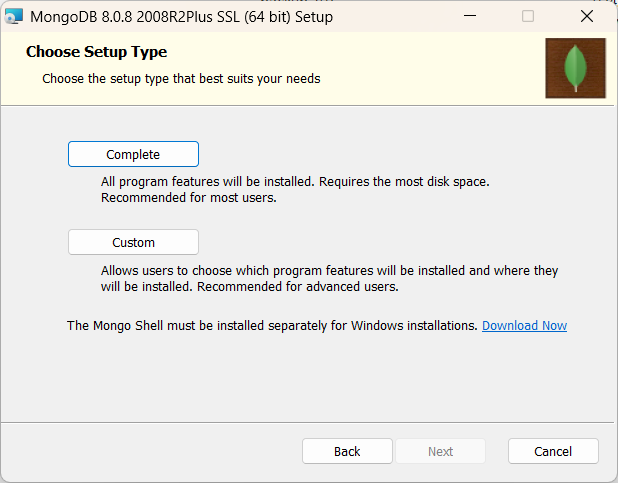
1. Step 2: Run the Installer:
   1. Run the downloaded .msi file.



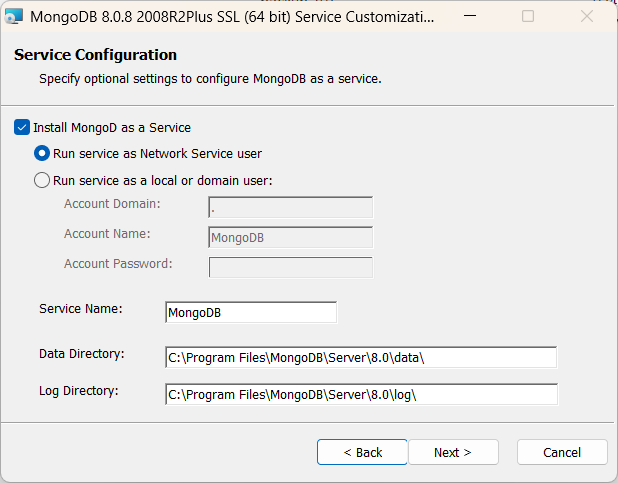
* 1. Click Next on the welcome screen.
  2. Accept the License Agreement, then click Next.



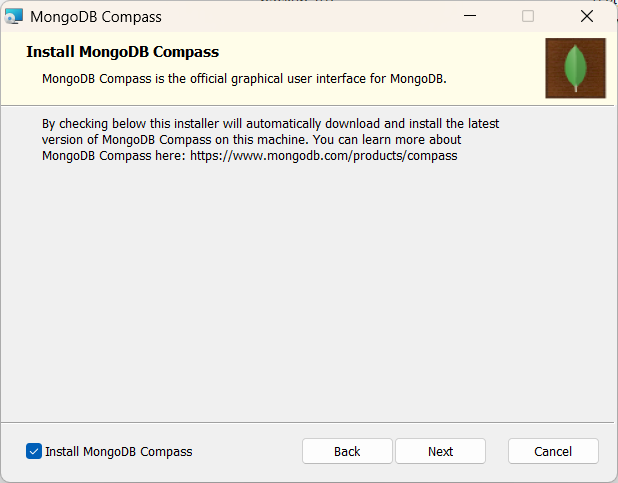
* 1. Choose Complete setup (recommended).



* 1. Ensure the following options are checked:
     1. "Install MongoDB as a Service"



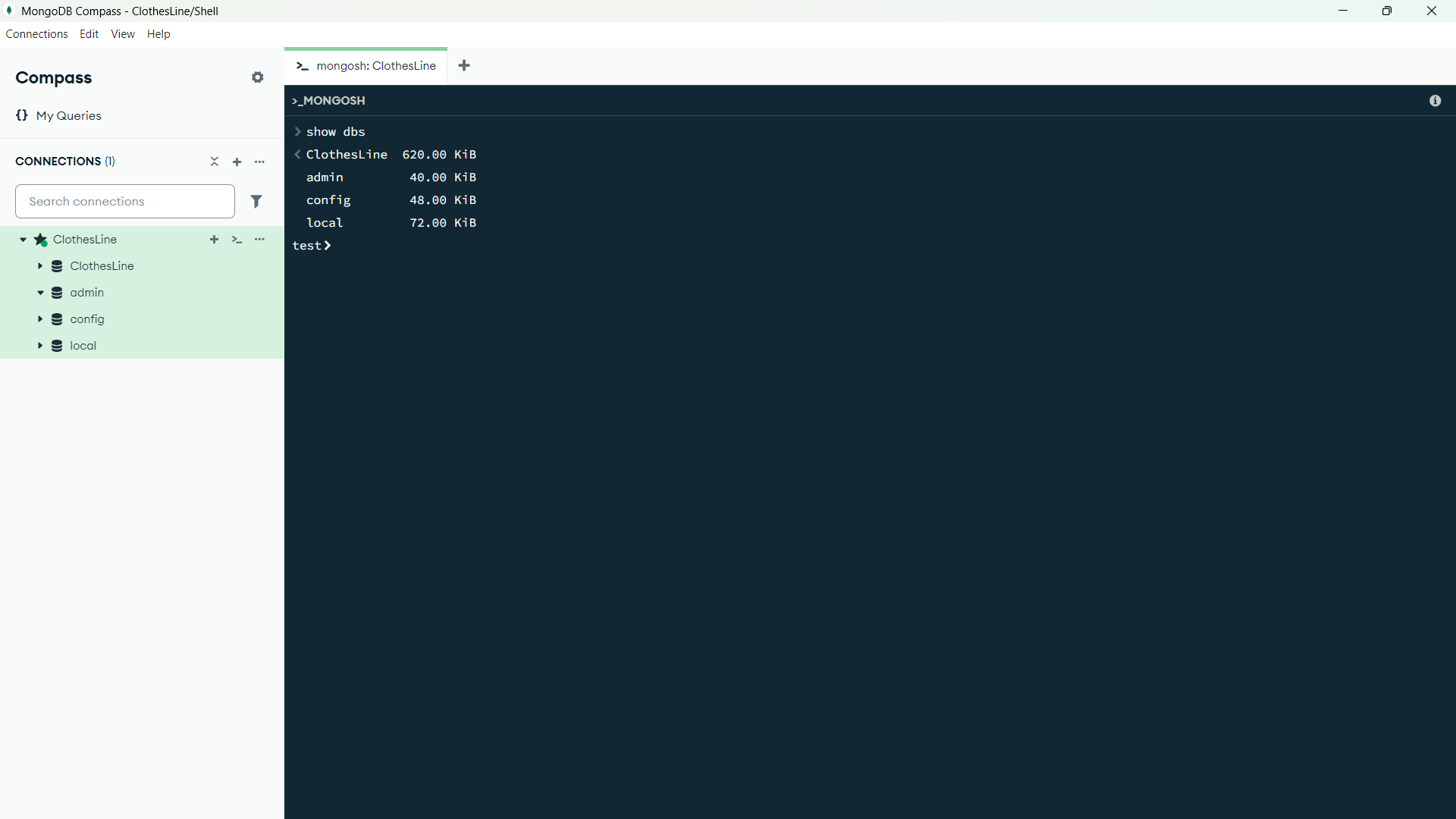
* + 1. "Run service as Network Service user"
    2. "Install MongoDB Compass" (optional)
  1. Click Next, then Install.



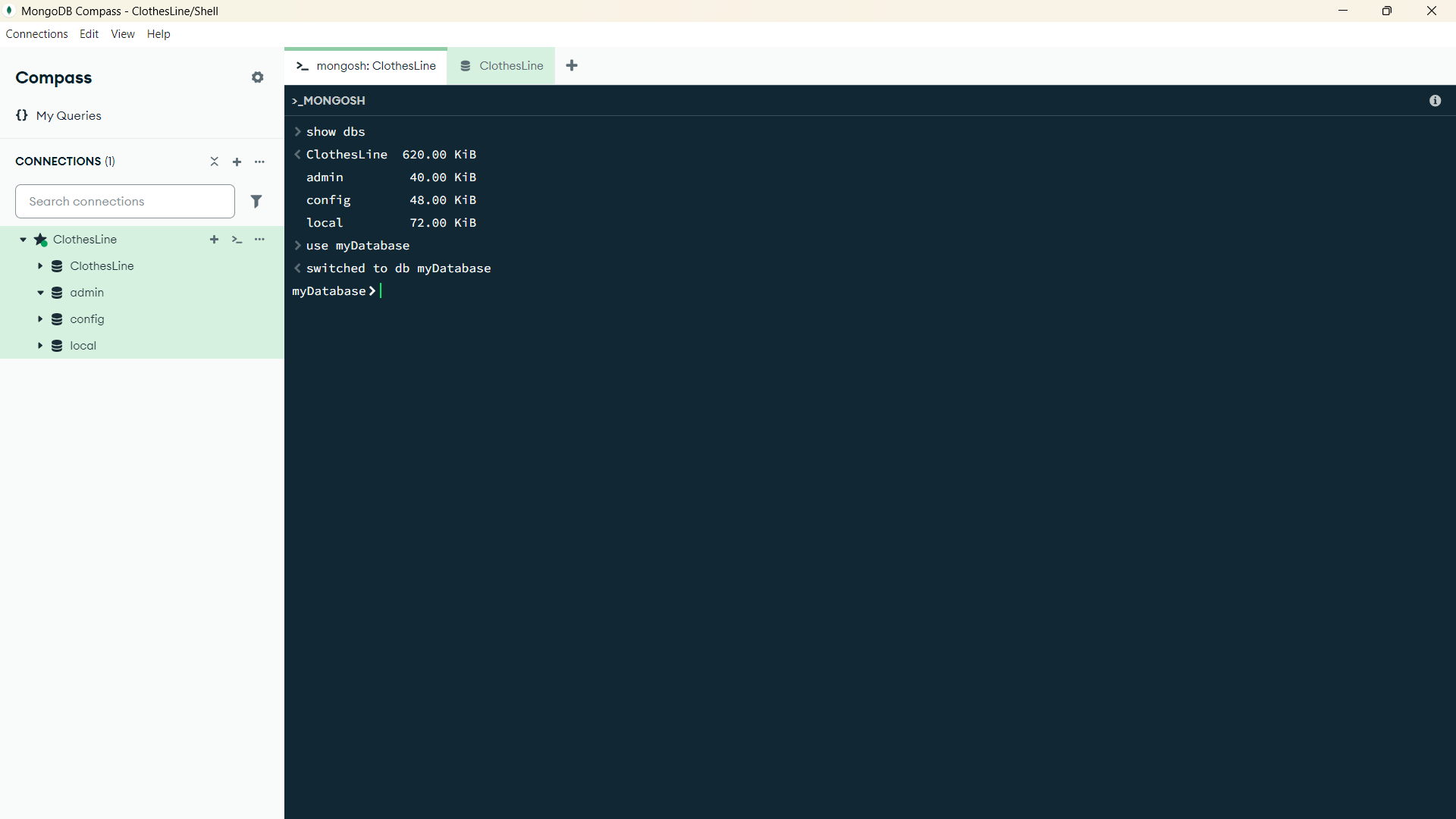
The Above step will also install the MongoDB Compass App that is the GUI version.

**MongoDB Shell Commands And CRUD operations.**

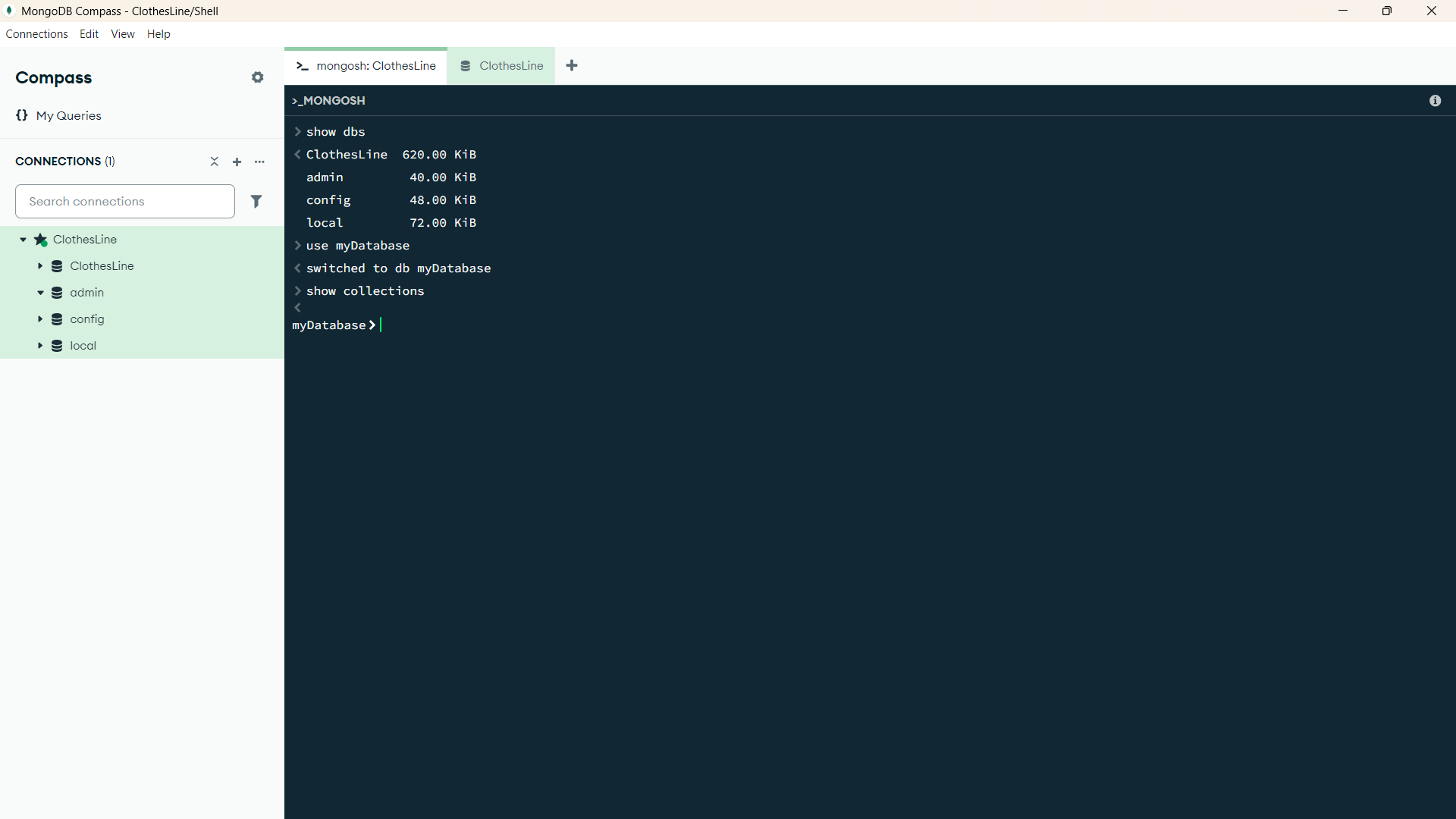
1. Show Databases: show dbs



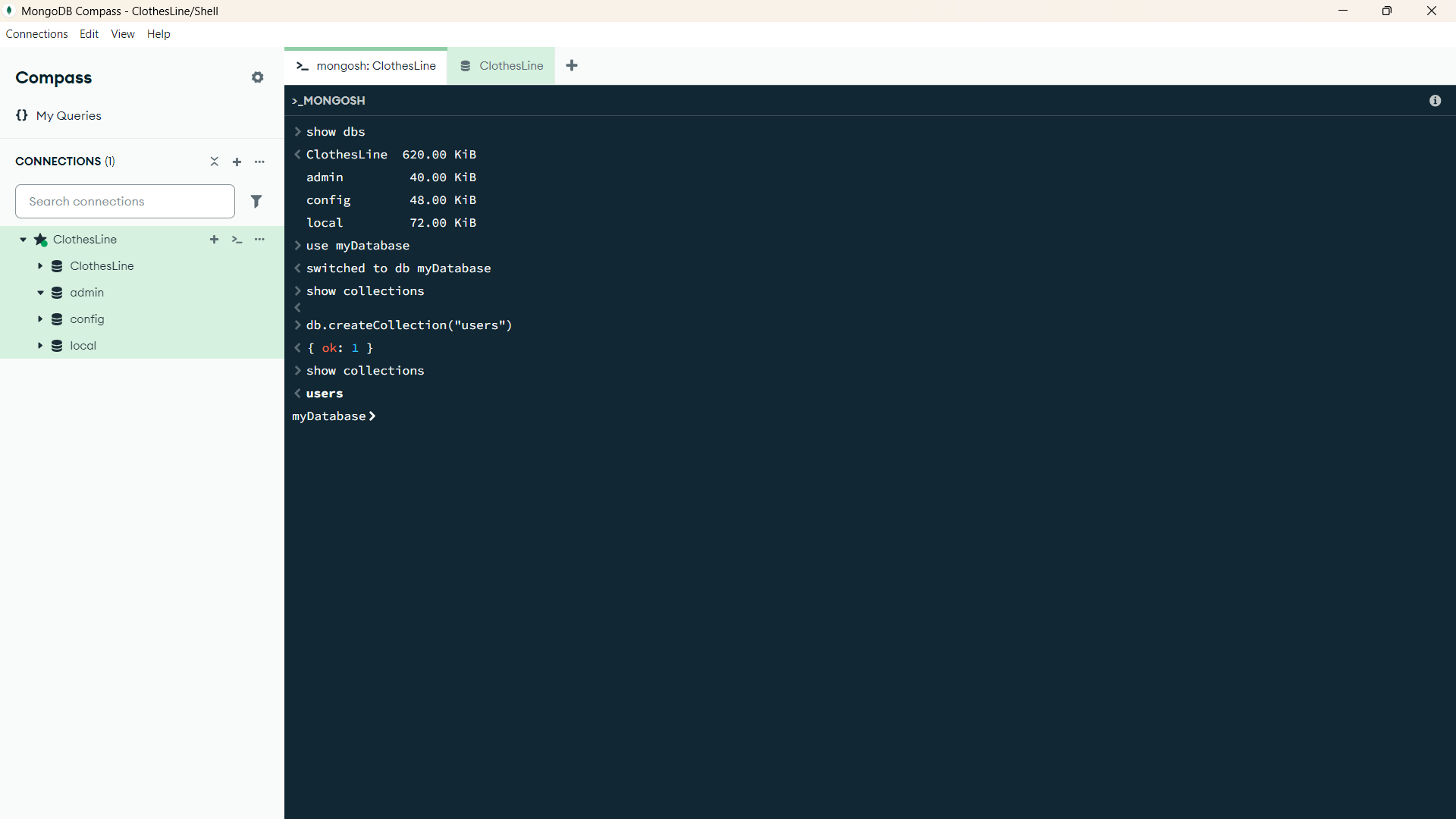
1. Create / Switch to a Database: use myDatabase
   1. If myDatabase doesn’t exist, MongoDB will **create it** when you insert the first document.



1. Show Collections: show collections



1. Create a Collection:

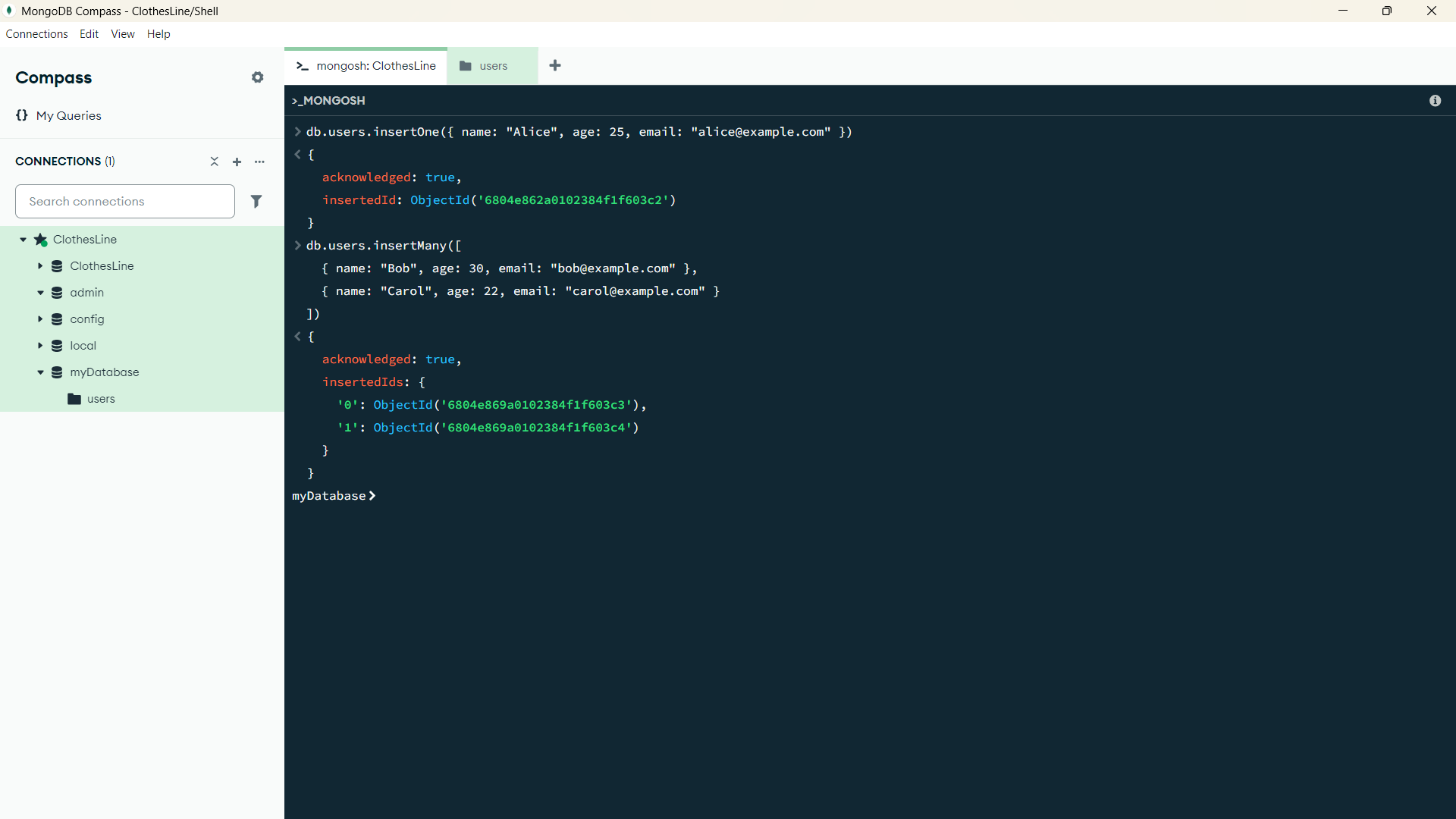


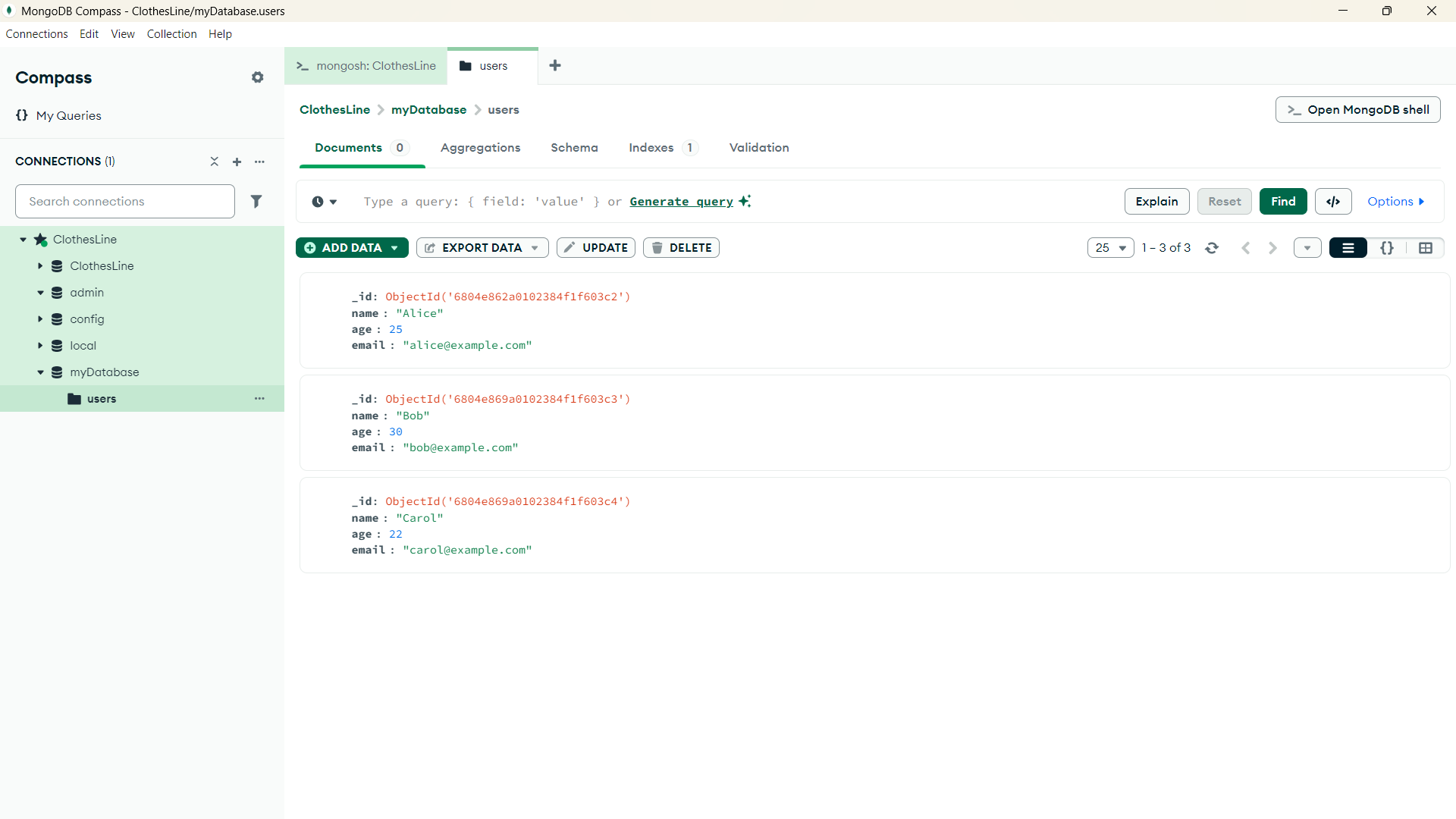
1. CRUD Operations:
   1. CREATE:
      1. Insert One Document:db.users.insertOne({ name: "Alice", age: 25, email: "alice@example.com" })
      2. Insert Multiple Documents: db.users.insertMany([

{ name: "Bob", age: 30, email: "bob@example.com" },

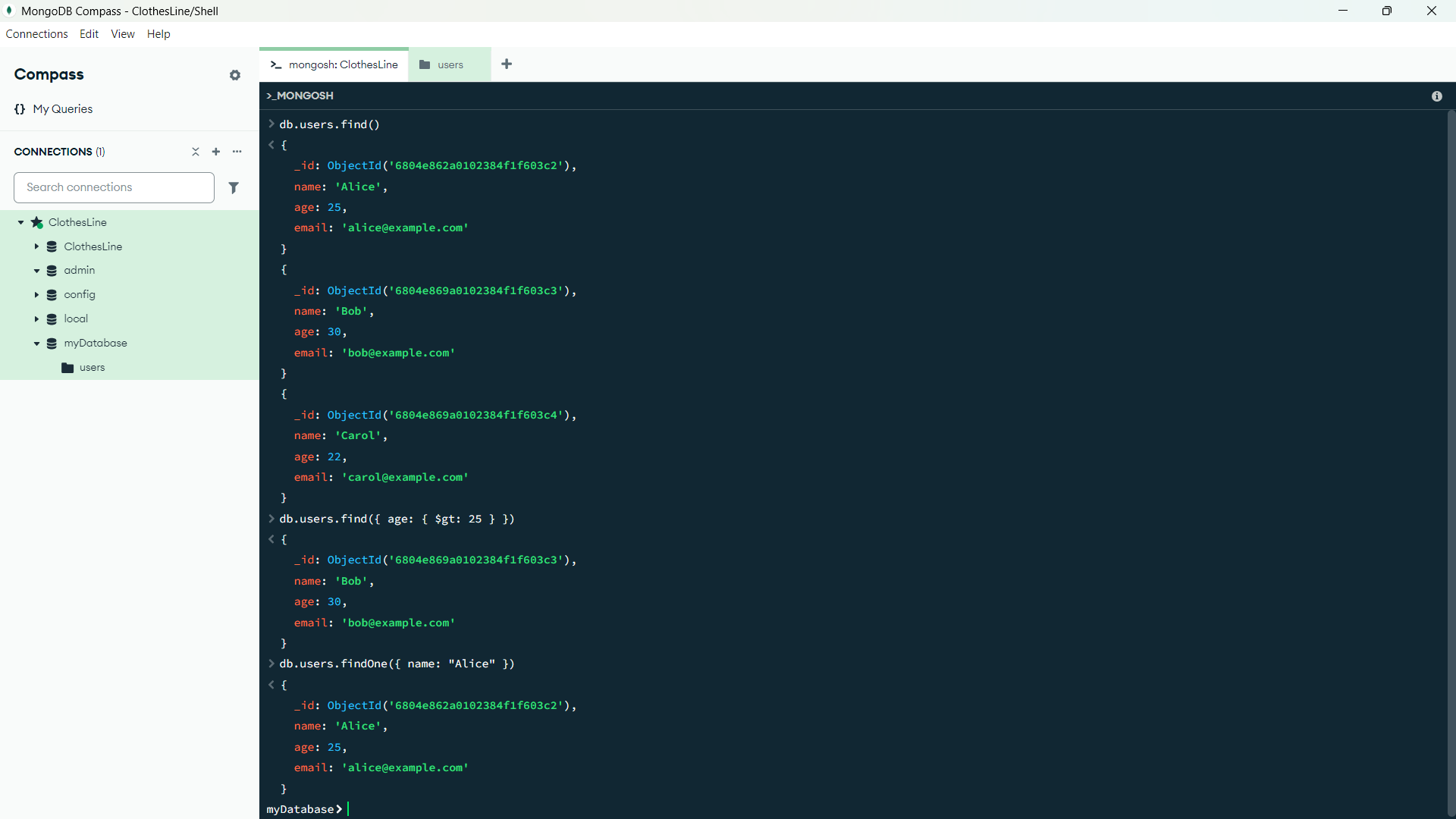
{ name: "Carol", age: 22, email: "carol@example.com" }

])





* 1. READ:
     1. Find All Documents: db.users.find()
     2. Find with Condition: db.users.find({ age: { $gt: 25 } })
     3. Find One Document: db.users.findOne({ name: "Alice" })



* 1. UPDATE:
     1. Update One Document: db.users.updateOne(

{ name: "Alice" },

{ $set: { age: 26 } }

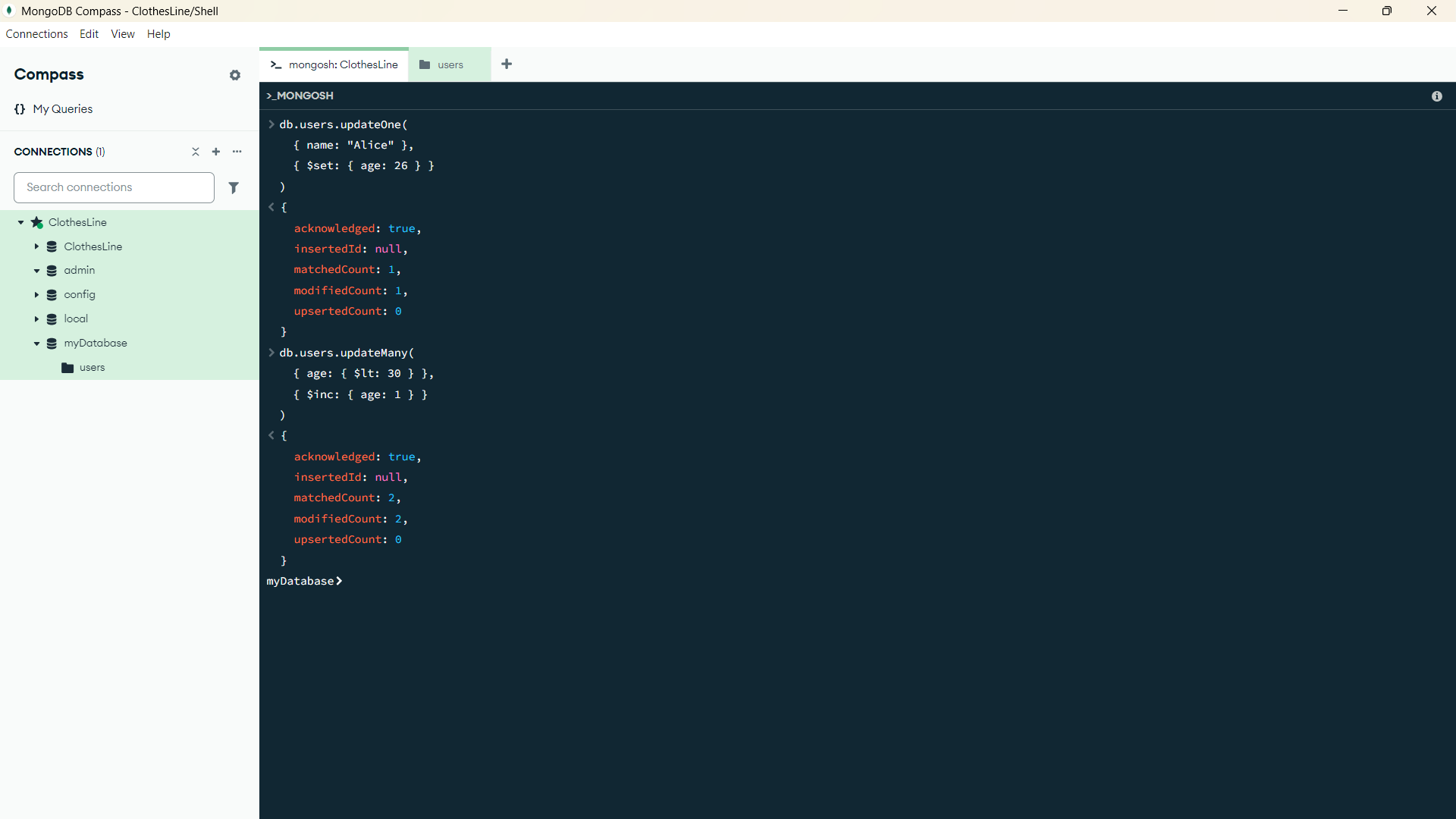
)

* + 1. Update Multiple Documents:db.users.updateMany(

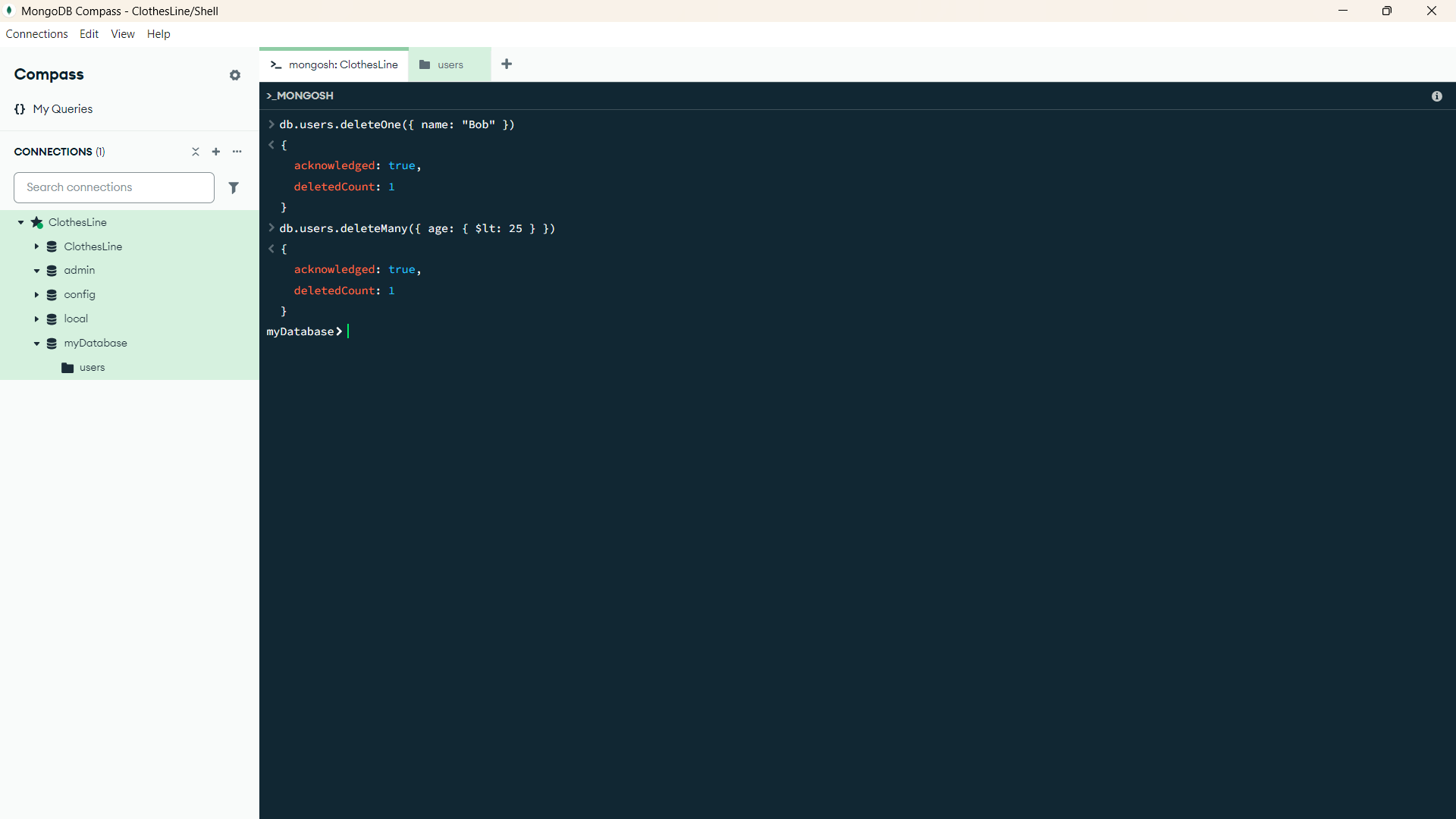
{ age: { $lt: 30 } },

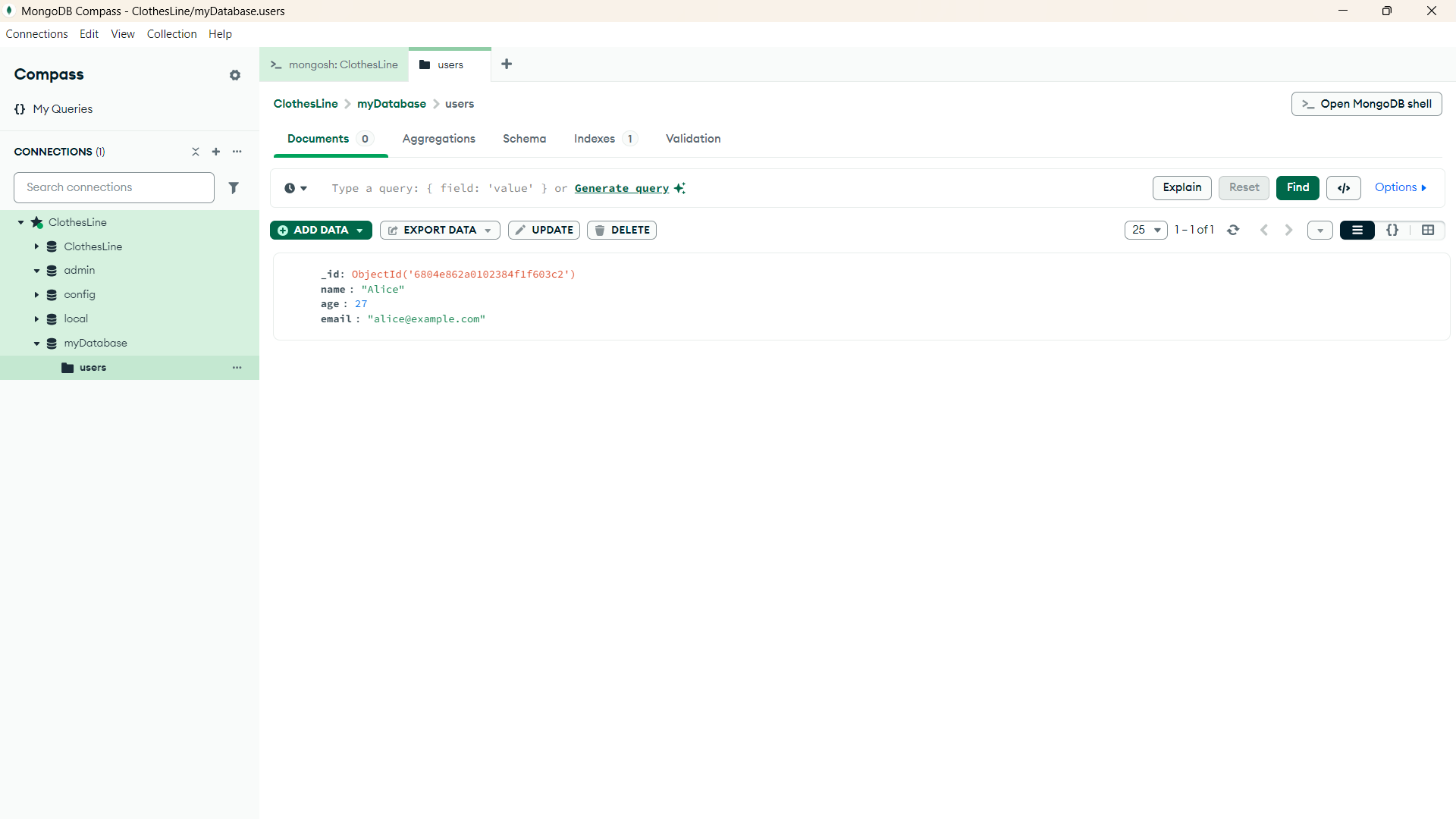
{ $inc: { age: 1 } }

)



* 1. DELETE:
     1. Delete One Document:db.users.deleteOne({ name: "Bob" })
     2. Delete Multiple Documents: db.users.deleteMany({ age: { $lt: 25 } })





**1.6 Conclusion:**

MongoDB makes working with data feel intuitive and flexible, especially when you understand its core operations. From **creating a database** to **inserting, reading, updating, and deleting documents**, MongoDB’s shell commands give you full control over your data with minimal syntax.

Whether you're building a simple app or designing a scalable backend system, mastering these basic **CRUD operations** lays a solid foundation.