**MongoDB compass**

MongoDB Compass is a graphical user interface (GUI) for MongoDB that allows you to:  
✔️ Connect to a MongoDB database  
✔️ Create, read, update, and delete (CRUD) documents  
✔️ Visualize and analyze your data  
✔️ Run queries without writing complex shell commands

**Creating a Database and Collection**

1. Click on **"Create Database"** (top-left corner).
2. Enter:
   * **Database Name:** studentDB
   * **Collection Name:** students
3. Click **Create Database**.

**Insert a Document**

1. Open the students collection.
2. Click **"Insert Document"**.
3. Add the following JSON:

{

"name": "John Doe",

"age": 22,

"course": "Computer Science"

}

1. Click **Insert**.

When you connect to MongoDB, you will see some default databases like:

1️. **admin** – Used for administrative tasks like authentication and server configuration.  
2️. **config** – Stores metadata about sharded clusters (used in advanced setups).  
3️. l**ocal** – Stores information specific to the local server, including replication data.

1️. **Indexes in MongoDB Compass** – Learn how indexes improve query performance.  
2️. **Aggregation Pipeline** – Use Compass to analyze and process data.  
3️. **Import & Export Data** – Learn how to move data between Compass and files.  
4️. **Schema Validation** – Enforce rules on what data can be inserted.

**Indexes in MongoDB Compass**

**Why?** Indexes improve query speed by allowing MongoDB to find data faster.

**Task: Create an Index on the name field**

1. Open studentDB → students collection.
2. Click **Indexes** → **Create Index**.
3. Add a field:
   * **Field Name:** name
   * **Type:** Ascending (1)
4. Click **Create Index**.

🔹 **Check:** Try searching { "name": "John Doe" } and see if the query runs faster!

**Aggregation Pipeline**

**Why?** Aggregation helps in data processing, filtering, and transformation.

**Task: Find students older than 21**

1. Go to the **Aggregation** tab in students.
2. Click **Add Stage** → Select $match.
3. Enter the query: { "age": { "$gt": 21 } }
4. Click **Run**.

🔹 **Check:** Do you see students older than 21?

**Common MongoDB Comparison Operators**

| **Operator** | **Description** | **Example Query** |
| --- | --- | --- |
| $eq | Equal to | { "age": { "$eq": 21 } } |
| $ne | Not equal to | { "age": { "$ne": 21 } } |
| $gt | Greater than | { "age": { "$gt": 21 } } |
| $gte | Greater than or equal to | { "age": { "$gte": 21 } } |
| $lt | Less than | { "age": { "$lt": 21 } } |
| $lte | Less than or equal to | { "age": { "$lte": 21 } } |
| $in | Matches any value in an array | { "age": { "$in": [20, 21, 22] } } |
| $nin | Does NOT match any value in an array | { "age": { "$nin": [20, 21, 22] } } |

$match is a **stage** in the Aggregation Pipeline that filters documents **based on a condition**—similar to the find() query in MongoDB.

**Other Frequently Used Aggregation Stages**

| **Stage** | **Description** | **Example** |
| --- | --- | --- |
| $match | Filters documents | { "age": { "$gt": 21 } } |
| $project | Selects specific fields | { "name": 1, "age": 1, "\_id": 0 } |
| $group | Groups data, often with aggregation functions like $sum, $avg | { "\_id": "$course", "count": { "$sum": 1 } } |
| $sort | Sorts documents | { "age": -1 } (Descending) |
| $limit | Limits the number of results | 5 (Shows 5 documents) |
| $skip | Skips a certain number of documents | 5 (Skips first 5 docs) |
| $unwind | Breaks arrays into separate documents | { "path": "$subjects" } |

**Schema Validation**

**Why?** Ensures only valid data gets inserted.

**Task: Add Schema Validation to Restrict age to Numbers**

1. Open students collection.
2. Click **Schema Validation** → **Add Rule**.
3. Paste this rule:

{

"properties": {

"age": { "bsonType": "int" }

}

}

1. Click **Save & Enforce**.

🔹 **Check:** Try inserting { "name": "Alice", "age": "twenty" }—it should **fail**

!

**MongoDB shell(Mongosh)**

**1️ Basic Commands**

| **Command** | **Description** |
| --- | --- |
| show dbs | Lists all databases. |
| use “databaseName” | Switches to (or creates) a database. |
| db | Displays the current database. |
| show collections | Lists all collections in the current database. |

**2. CRUD Operations (Create, Read, Update, Delete)**

**📌 Insert Documents**

**# Insert one document**

**db.students.insertOne({ name: "Alice", age: 20, course: "Computer Science" })**

**# Insert multiple documents**

**db.students.insertMany([**

**{ name: "Bob", age: 22, course: "Math" },**

**{ name: "Charlie", age: 21, course: "Physics" },**

**{ name: "David", age: 23, course: "Biology" }**

**])**

📌 Read (Find) Documents

# Find all documents

db.students.find().pretty()

# Find a student named "Charlie"

db.students.find({ name: "Charlie" }).pretty()

# Find students older than 21

db.students.find({ age: { $gt: 21 } }).pretty()

# Find students taking "Math" or "Physics"

db.students.find({ course: { $in: ["Math", "Physics"] } }).pretty()

📌 Update Documents

# Update one document (change Bob's age to 23)

db.students.updateOne(

{ name: "Bob" },

{ $set: { age: 23 } }

)

# Update multiple documents (increase all students' age by 1)

db.students.updateMany(

{},

{ $inc: { age: 1 } }

)

📌 Delete Documents

# Delete one document (remove student named "David")

db.students.deleteOne({ name: "David" })

# Delete multiple documents (remove all students older than 25)

db.students.deleteMany({ age: { $gt: 25 } })

**Additional Useful Commands**

| **Command** | **Description** |
| --- | --- |
| db.dropDatabase() | Deletes the current database. |
| db.collectionName.drop() | Deletes a collection. |
| db.students.countDocuments() | Counts the number of documents in a collection. |

**MongoDB playground**

// ✅ Connect to the database

use("schoolDB");

// ✅ Show databases & collections

show dbs;

db.getCollectionNames();

// ✅ Create a collection and insert one document

db.students.insertOne({ name: "Alice", age: 20, course: "CS" });

// ✅ Insert multiple documents

db.students.insertMany([{ name: "Bob", age: 22, course: "Math" }, { name: "Charlie", age: 21, course: "Physics" }]);

// ✅ Read documents (find all)

db.students.find();

// ✅ Find specific documents

db.students.find({ name: "Alice" });

db.students.find({ age: { $gt: 21 } });

// ✅ Update one document

db.students.updateOne({ name: "Alice" }, { $set: { age: 21 } });

// ✅ Update multiple documents

db.students.updateMany({ age: { $lt: 21 } }, { $set: { course: "Updated Course" } });

// ✅ Delete one document

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

// ✅ Delete multiple documents

db.students.deleteMany({ age: { $gt: 25 } });

// ✅ Delete a collection

db.students.drop();

// ✅ Delete the database

db.dropDatabase();