MongoDB is a **NoSQL** database, which means it stores data in a different way compared to traditional relational databases (like MySQL or PostgreSQL).

## **What is a Document in MongoDB?**

A **document** is the **basic unit of data** in MongoDB.

* It's a JSON-like structure (actually stored in a format called **BSON** — Binary JSON).
* Each document is like a row in a table, but it’s **flexible** — fields can vary from one document to another.
* Documents can contain:  
  + Strings, numbers, arrays, booleans, nested documents (like objects), etc.

Example of a document:

{

"name": "Alice",

"age": 25,

"city": "Bangalore",

"skills": ["Java", "MongoDB", "Spring Boot"]

}

## **What is a Collection?**

A **collection** is a **group of documents** — like a table in SQL.

* You can think of it as a folder full of JSON documents.
* All documents in a collection are typically related (e.g., all are "users" or "products"), but they **don’t need to have the same structure**.
* In SQL: **Table** = In MongoDB: **Collection**

### **Example:**

A collection named users might have these documents:

{ "name": "Alice", "age": 25 }

{ "name": "Bob", "city": "Delhi" }

{ "name": "Charlie", "age": 22, "hobbies": ["reading", "music"] }

## **What is a Database?**

Just like in any DB system, a **database** is a container for collections.

### **Structure Summary:**

MongoDB Server

├── Database: testdb

│ ├── Collection: users

│ │ ├── Document 1: { "name": "Alice", "age": 25 }

│ │ ├── Document 2: { "name": "Bob", "city": "Delhi" }

│ │ └── Document 3: { "name": "Charlie", "age": 22 }

│ └── Collection: orders

│ └── Documents: ...

## **MongoDB automatically adds \_id**

Each document **automatically gets an \_id field**, which is a unique identifier.

{

"\_id": ObjectId("6503ff0d51d9b72c93d96c90"),

"name": "Alice",

"age": 25

}

You can also provide your own \_id if needed.

db.users.insertOne({ name: "Alice", age: 25 })

* **db** This refers to the current database you’re using in the MongoDB shell. For example, if you ran use testDB, then db points to testDB.
* **users** This is the **collection** name inside the database. Collections are like tables in relational databases, but more flexible. Here, users is the collection where you want to store documents.
* **insertOne()** This is a method/function that inserts **one document** into the specified collection (users).
* **{ name: "Alice", age: 25 }** This is the **document** you are inserting. It’s a JSON-like object with two fields:  
  + name with the value "Alice" (a string)
  + age with the value 25 (a number)

### **Open the mongo shell**

Open Terminal and type:

mongosh

You should see a prompt like:

test>

### **2. Select/Create a database**

use testDB

### **3. Create/Insert documents (Create)**

Insert one document:

db.users.insertOne({ name: "Alice", age: 25 })

Insert many documents:

db.users.insertMany([

{ name: "Bob", age: 30 },

{ name: "Charlie", age: 35 }

])

### **4. Read documents (Read)**

Find all documents:

db.users.find()

Find documents with condition:

db.users.find({ age: { $gt: 28 } })

$gt = greater than

Other similar operators:

* **$lt = less than**
* **$gte = greater than or equal**
* **$lte = less than or equal**
* **$eq = equals**

Pretty print output:

db.users.find().pretty()

### **5. Update documents (Update)**

Update one document:

db.users.updateOne({ name: "Alice" }, { $set: { age: 26 } })

Update many documents:

db.users.updateMany({ age: { $lt: 30 } }, { $set: { status: "young" } })

### **6. Delete documents (Delete)**

Delete one document:

db.users.deleteOne({ name: "Charlie" })

Delete many documents:

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

### **7. Drop collection (delete all docs & collection)**

db.users.drop()

### **To switch to another database without exiting:**

Just run:

use otherDatabaseName

Example:

use testDB2

This changes your current database context to testDB2