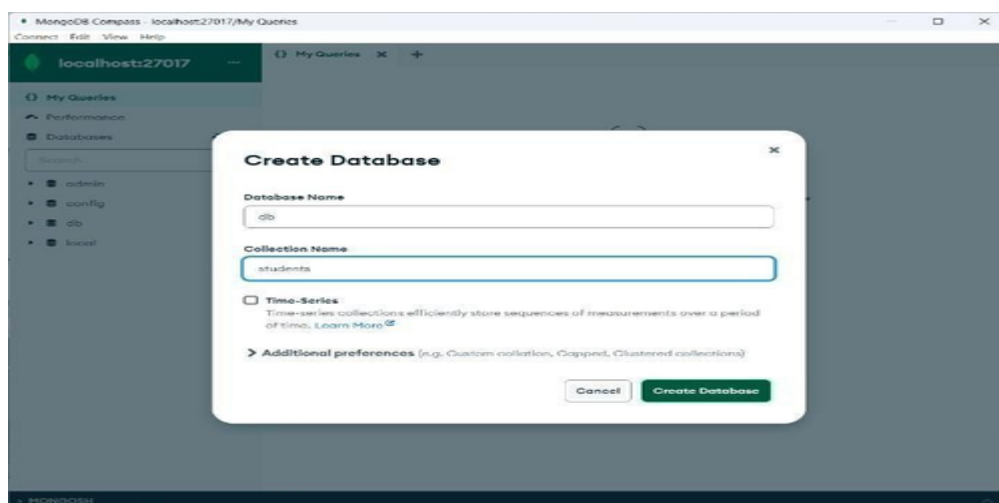
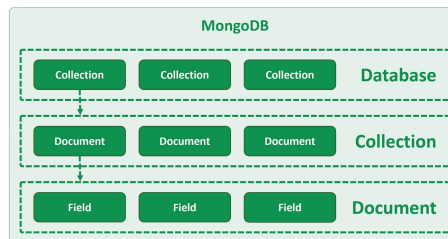


# MongoDB:Data base, Collection & Documents

## DATA BASE :

**Database:** A container that holds multiple collections of related data.

**Collection:** Similar to a table in arelational database,a collection stores documents.



## How to Import data in MongoDB Compass?

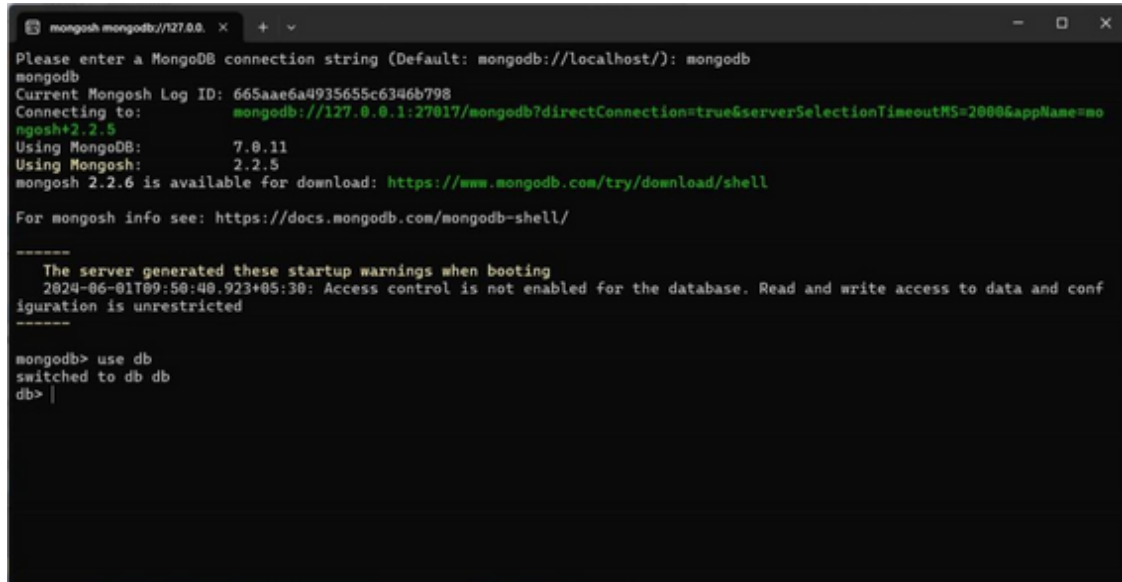
In Compass, it has always been quite easy to import data – from JSON and CSV files

- Click on Import Button
- Select the exported JSON or CSV file you want to import.
- Select the connection names you want to import. Click Import.

After connections, Few commands to test:

## Switching Databases:

By default, the db points to the "test" database when you start the Mongo shell. To switch to a different database named "db", you can use the use command followed by the database name



```
mongosh mongodb://127.0.0.1:27017
Please enter a MongoDB connection string (Default: mongodb://localhost/): mongodb
mongodb
Current Mongosh Log ID: 665aae6a4935655c6346b798
Connecting to:  mongodb://127.0.0.1:27017/mongodb?directConnection=true&serverSelectionTimeoutMS=2000&appName=mongosh+2.2.5
Using MongoDB: 7.0.11
Using Mongosh: 2.2.5
mongosh 2.2.6 is available for download: https://www.mongodb.com/try/download/shell

For mongosh info see: https://docs.mongodb.com/mongosh-shell/

-----
The server generated these startup warnings when booting
2024-06-01T09:50:40.923+05:30: Access control is not enabled for the database. Read and write access to data and configuration is unrestricted
-----

mongodb> use db
switched to db db
db> |
```

## Accessing Collections in the Current Database:

The db variable acts as a reference point for accessing collections within the current database you're connected to.

## Show Collections:

```
mongosh mongodb://127.0.0.1:27017/
Please enter a MongoDB connection string (Default: mongodb://localhost/): mongodb
mongodb
Current Mongosh Log ID: 665aae6a4935655c6346b798
Connecting to:      mongodb://127.0.0.1:27017/mongodb?directConnection=true&serverSelectionTimeoutMS=2000&appName=mongosh+2.2.5
Using MongoDB:      7.0.11
Using Mongosh:       2.2.5
mongosh 2.2.6 is available for download: https://www.mongodb.com/try/download/shell

For mongosh info see: https://docs.mongodb.com/mongosh-shell/

-----
The server generated these startup warnings when booting
2024-06-01T09:50:40.923+05:30: Access control is not enabled for the database. Read and write access to data and configuration is unrestricted
-----

mongodb> use db
switched to db db
db> show collections
students
db> |
```

// Insert a record to collection. Create collection if not exist.

```
}
db> db.color.insertOne({"color":"red"})
{
  acknowledged: true,
  insertedId: ObjectId('6661c03b0b2b725bae46b79a')
}
db> |
```

Documents:

In MongoDB, documents are the fundamental unit of data storage. They act like containers holding information in a flexible schema. Here's a breakdown of what documents are in MongoDB

### Example:

```
_id: ObjectId('6650add5ec3192109fec31a')
name: "Student 948"
age: 19
courses: "['English', 'Computer Science', 'Physics', 'Mathematics']"
gpa: 3.44
home_city: "City 2"
blood_group: "O+"
is_hotel_resident: true
```

- **\_id** is a special field that uniquely identifies each document by default.
- Other fields like **name**, **age**, and **courses** hold specific data about the document.
- Nested documents like **home\_city** can be used to group related information.

### Benefits:

- The flexible schema of documents allows you to store data without rigid table structures like in relational databases.
- Documents can efficiently represent complex data hierarchies.

### *Documents can be represented as JSON*

In MongoDB, documents are the fundamental unit of data storage. Each document represents a record and uses a JSON-like structure. This structure allows for storing various data types within a document and enables easy manipulation and querying of data.

So, while JSON is not exclusive to documents, it serves as a powerful and widely used way to represent them, especially in NoSQL databases like MongoDB.

## Collections:

In MongoDB, collections act like containers that hold documents, which are essentially JSON-like structures that store your actual data. You can think of collections as similar to tables in relational databases, but with some key differences.

Examples:

- \_id o name o email
- orders

## Few Commands to test after connections

Command	Expected Output	Notes
show dbs	admin 40.00 KiB	All Databases are shown
use db	switched to db db	Connect and use db
show collections	Students	Show all tables
db.foo.insert({"bar": "baz"})		Insert a record to collection. Create Collection if not exists

Command	Notes
db.foo.batchInsert([{"_id" : 0}, {"_id" : 1}, {"_id" : 2}])	Insert more than one document
db.foo.find()	Print all rows
db.foo.remove()	Remove foo table

## Data base:

- MongoDB groups collections into databases.
- A single instance of MongoDB can host several databases, each grouping together zero or more collections.
- A database has its own permissions, and each database is stored in separate files on disk.

## Datatype:

MongoDB stores data using a format called BSON (Binary JSON), which extends JSON with additional data types. Here's a rundown of the common data types you'll encounter in MongoDB

### Basic Types:

- **String:** The most common type, used for textual data. It must be valid UTF-8.
- **Integer:** Stores whole numbers, can be 32-bit or 64-bit depending on your server configuration.
- **Double:** Stores floating-point numbers with higher precision.
- **Boolean:** Represents true or false values.

### Complex Types:

- **Array:** Stores an ordered list of values of various data types.
- **Object:** Used for embedding documents within documents, helpful for representing hierarchical data.
- **Date:** Stores dates and times as milliseconds since the Unix epoch.
- **ObjectId:** A unique 12-byte identifier automatically generated for each document when inserted.

### Other Data Types:

- **Null:** Represents missing or undefined values.
- **Binary:** Stores raw binary data like images or files.
- **Code:** Allows storing JavaScript code within a document for server-side execution.
- **Symbol:** Similar to strings but less common, often used for specific data types in some programming languages.
- **Timestamp:** Represents a specific point in time with nanosecond precision.
- **Decimal128:** Stores high-precision decimal numbers

### Now, we can write queries in the mongo Shell.

Once you are in the MongoDB shell, create the database in MongoDB by typing this command:

>use database\_name// to create a new DB or to connect to already existed DB

>db // to check currently connected DB

>show dbs // to list all the DBs

The DB madavi is created; is not present in the list of all the databases. This is because a database is not created until you save a document in it.



Note: If the database name you mentioned is already present then this command will connect you to the database. However if the database doesn't exist then this will create the database with the given name and connect you to it.

- Now we are creating a collection Student and inserting a document in it.

```
>db.student.insert({name: "sree", age: 30, address:"vijayawada"})
```

- You can now see that the database "madavi" is created.

To Drop the DataBase ,

- `show dbs` // list all Dbs
- `use databse_name` //switch to the DB that needs to be dropped
- Example:

```
>use madavi
```

```
>db.dropDatabase()
```

- `show dbs` // to show the list of DBs after deletion

## OUTPUT

```
Command Prompt - mongo

To enable free monitoring, run the following command: db.enableFreeMonitoring()
To permanently disable this reminder, run the following command: db.disableFreeMonitoring()
---
> show dbs
admin    0.000GB
config   0.000GB
local    0.000GB
> use madavi
switched to db madavi
> db
madavi
> show dbs
admin    0.000GB
config   0.000GB
local    0.000GB
> db.student.insert({name:"sai",branch:"CSD",city:"Vijayawada"})
WriteResult({ "nInserted" : 1 })
> show dbs
admin    0.000GB
config   0.000GB
local    0.000GB
madavi    0.000GB
> db
madavi
> db.dropDatabase();
{ "ok" : 1 }
> show dbs
admin    0.000GB
config   0.000GB
local    0.000GB
>
```

Creating the Collection in MongoDB on the fly.

- The cool thing about MongoDB is that you need not to create collection before you insert document in it. With a single command you can insert a document in the collection and the MongoDB creates that collection on the fly.
- SYNTAX:


`db.collection_name.insert({key:value, key:value...})`

- EXAMPLE:

`db.student.insert({rollno:"20X41A0441",name:"durga",age:18,city:"Vijayawada"})`

- SYNTAX: `db.collection_name.find()`
- To check whether the collection is created successfully, use the following command.
- **show collections** // This command shows the list of all the collections in the currently selected database.

## Output:



```
Command Prompt - mongo
> show dbs
admin    0.000GB
config  0.000GB
local    0.000GB
> db.student.insert({rollno:"20x41A0501",name:"sree",age:19,city:"Vijayawada"})
WriteResult({ "nInserted" : 1 })
> db.student.find()
{ "_id" : ObjectId("625656a98fb133b113147087"), "rollno" : "20x41A0501", "name" : "sree",
"age" : 19, "city" : "Vijayawada" }
> db
madavi
> show dbs
admin    0.000GB
config  0.000GB
local    0.000GB
madavi   0.000GB
> show collections
student
>
```

## SYNOPSIS:

>db.collection\_name.drop() // to drop the collection

>db.dropDatabase() // to drop the current Database

>db // to show the current Database name

- show dbs // to show all the list of DBs

>show collections // to show the collection name

>db.collection\_name.find() // to find the documents in the collection

- use databasename // to create the new DB

>db.createCollection(name,options) // to create new collection

## Drop a Database :

In MongoDB, databases hold collections of documents. On a single MongoDB server, we can run multiple databases. when you install MongoDB some databases are automatically generated to use. many

times you need to delete some database when the database is no longer used.

**db.dropDatabase()** the command is used to drop an existing database. This command will delete the currently selected database. If you have not selected any database, then it will delete the default „test“ database.

**Syntax :**

**Example :** The below screenshot is showing the use of “db.dropDatabase()” command for a newly created database name userDB.

```
> use userDB
switched to db userDB
> db.dropDatabase()
{ "ok" : 1 }
```

**How to delete a database which is not currently used :**

You can check currently selected database, using the command “**db**“. Then you can use “**show dbs**” command for checking the list of Databases. Then select the database you want to delete using command “**use databasename**“.

**Example :**

In below example we were using a database name userDB and we want to delete a different database name adminDB. so first we are going to select adminDB database then we will delete this database.

```
> db                                     check the current database
userDB
> show dbs                             check list of database to verify database
UserDB  0.000GB
admin   0.000GB
adminDB 0.000GB
config  0.000GB
local   0.000GB
                                exist or deleted already
> use adminDB                         select the database you
switched to db adminDB
                                want to delete
> db
adminDB
> db.dropDatabase()                   drop the database
{ "dropped" : "adminDB", "ok" : 1 }
```

## References:

<https://docs.mongodb.com/manual/core/databases-and-collections/#databases>

```
db.dropDatabase()
```

Then execute **db.dropDatabase()** command to drop an existing database.