**MongoDB Tutorial**

**What is MongoDB?**

MongoDB is a NoSQL, document-oriented database. It stores data in flexible, JSON-like documents, meaning fields can vary from document to document, and data structure can be changed over time.

**Key Concepts**

* **Document**: Basic unit of data, similar to a JSON object.
* **Collection**: Group of MongoDB documents, equivalent to a table in relational databases.
* **Database**: Holds collections.

**Installation**

**Local Installation**

* Visit: <https://www.mongodb.com/try/download/community>
* Download and follow instructions for your OS.

**Using MongoDB Atlas (Cloud)**

* Visit: <https://www.mongodb.com/cloud/atlas>
* Create a free cluster and follow prompts to set up.

**Basic Commands**

**1. Connecting to MongoDB**

bash

# Connect to local MongoDB

mongosh

# Or connect to Atlas cluster

mongosh "mongodb+srv://<username>:<password>@cluster-url"

**2. Creating/Selecting a Database**

js

use myDatabase

**3. Creating a Collection and Inserting Documents**

js

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

db.users.insertMany([{ name: "Bob", age: 30 }, { name: "Charlie", age: 35 }])

**4. Finding Documents**

js

db.users.find() // All users

db.users.find({ name: "Alice" }) // Filter by name

db.users.findOne({ age: 30 }) // Find one document

**5. Updating Documents**

js

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

db.users.updateMany({}, { $inc: { age: 1 } }) // Increment age for all

**6. Deleting Documents**

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

db.users.deleteMany({ age: { $gt: 30 } }) // Delete users older than 30

**Indexes**

db.users.createIndex({ name: 1 }) // Create index on name

**Aggregation Example**

db.users.aggregate([

{ $group: { \_id: "$age", count: { $sum: 1 } } }

])

Absolutely! Here's a handy list of **basic MongoDB shell commands** to get you started:

🗂️ Database Commands

| **Command** | **Description** |
| --- | --- |
| show dbs | Lists all databases |
| use | Switches to the specified database |
| db | Displays the current database |
| db.dropDatabase() | Deletes the current database |

📁 Collection Commands

| **Command** | **Description** |
| --- | --- |
| show collections | Lists all collections in the current database |
| db.createCollection("name") | Creates a new collection |
| db.collectionName.drop() | Deletes a collection |

📄 Document Commands

| **Command** | **Description** |
| --- | --- |
| db.collectionName.insertOne({...}) | Inserts a single document |
| db.collectionName.insertMany([{...}, {...}]) | Inserts multiple documents |
| db.collectionName.find() | Retrieves all documents |
| db.collectionName.find().pretty() | Retrieves all documents in readable format |
| db.collectionName.update({criteria}, {update}) | Updates matching documents |
| db.collectionName.remove({criteria}) | Deletes matching documents |

🔍 Query Examples

| **Command** | **Description** |
| --- | --- |
| db.collectionName.find({name: "Alice"}) | Finds documents where name is Alice |
| db.collectionName.find({age: {$gt: 25}}) | Finds documents where age is greater than 25 |
| db.collectionName.countDocuments() | Counts all documents in the collection |

🆘 Help & Info

| **Command** | **Description** |
| --- | --- |
| help | Lists shell commands |
| db.help() | Lists database-related commands |
| db.collectionName.help() | Lists collection-related commands |

For a deeper dive, check out this [MongoDB Cheat Sheet](https://www.mongodb.com/developer/products/mongodb/cheat-sheet/) or this [quick reference guide](https://blog.e-zest.com/basic-commands-for-mongodb).

Would you like examples in Python or Node.js too?

**Indexing in MongoDB** is a powerful technique that dramatically improves query performance by allowing the database to locate data without scanning every document in a collection.

🔍 What Is an Index?

* An **index** is a special data structure (typically a B-tree) that stores a subset of fields from documents in a sorted, searchable format.
* It enables MongoDB to quickly match queries, sort results, and perform range operations.

⚡ Why Use Indexes?

Without indexes:

* MongoDB performs a **collection scan**, checking every document to find matches—slow and inefficient for large datasets. With indexes:
* Queries are faster and more efficient.
* Sorting and filtering operations are optimized.
* Aggregation pipelines benefit from reduced processing time.

🛠️ How to Create an Index

Use the createIndex() method:

db.collection.createIndex({ fieldName: 1 }) // 1 for ascending, -1 for descending



**Example:**

db.users.createIndex({ username: 1 })

You can also create:

* **Compound indexes**: { firstName: 1, lastName: 1 }
* **Unique indexes**: { email: 1 }, { unique: true }
* **TTL (Time to live) indexes**: Automatically remove documents after a set time
* **Sparse indexes**: Index only documents that contain the field

🧹 Dropping Indexes

To remove an index:

db.collection.dropIndex({ fieldName: 1 })

Or drop multiple indexes using:

db.collection.dropIndexes()

💡 Pro Tips

* Use indexes on fields frequently queried or sorted.
* Avoid over-indexing—each index adds overhead to write operations.
* Use MongoDB Atlas Performance Advisor to get index recommendations.

Would you like to explore compound indexes, text search, or performance tuning next? I can also walk you through real-world scenarios or help design indexes for a specific use case.

**MongoDB Aggregation Framework** is a powerful tool for transforming, analyzing, and summarizing data directly within the database—no need to export it to external tools like Spark or Hadoop.

🔧 What Is It?

* It’s a **pipeline-based system** where documents pass through multiple **stages**, each performing a specific operation (like filtering, grouping, projecting, etc.).
* Think of it like Unix pipes: the output of one stage becomes the input for the next.

🧱 Common Aggregation Stages

| **Stage** | **Purpose** |
| --- | --- |
| $match | Filters documents (like a WHERE clause) |
| $group | Groups documents and performs aggregations |
| $project | Reshapes documents (selects fields, renames) |
| $sort | Sorts documents |
| $limit | Limits the number of documents |
| $skip | Skips a number of documents |
| $lookup | Performs joins across collections |
| $unwind | Deconstructs arrays into individual docs |

🛠️ Example: Count Users by City

db.users.aggregate([ { $group: { \_id: "$city", totalUsers: { $sum: 1 } } } ])

This groups users by city and counts how many are in each.

🚀 Why Use It?

* Efficient data processing **inside MongoDB**—no need to move data out.
* Supports **real-time analytics** on live data.
* Handles **complex transformations** with ease.

🧠 Tips for Trainers Like You

* Use $project to simplify documents before grouping.
* Combine $match and $group early to reduce pipeline load.
* Leverage $lookup for cross-collection joins in curriculum demos.
* Use $facet for multi-path analysis (e.g., summary + detailed view).

Want to explore advanced stages like $bucket, $graphLookup, or build a real-world pipeline for training logs or session analytics? I’d love to help you design one tailored to your curriculum.