[**1. Introduction to MongoDB 2**](#_qlksqf9v51yz)

[**2. Installation and Setup 3**](#_j8643e71mkz8)

[**3. Developmental Tools 4**](#_79yraojsmpxt)

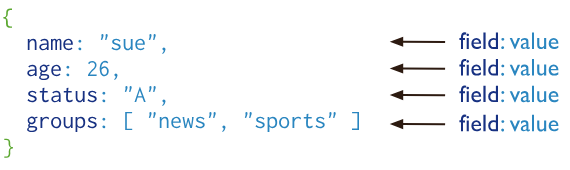
[**4. MongoDB Basics 5**](#_g64x4f7khg63)

[**5. Projects 8**](#_9irk2zcmb585)

# Introduction to MongoDB

* MongoDB is a NoSQL database that uses a document-oriented data model.
* MongoDB stores data in flexible, JSON-like documents where fields can vary from document to document.
* A record in MongoDB is a document, which is a data structure composed of field and value pairs.

**Sample :**



**SQL VS NoSQL**

| **Criteria** | **SQL (Relational DB)** | **NoSQL (Non-Relational DB)** |
| --- | --- | --- |
| Data Model | Structured data model with tables and rows. | Varied data models (document, key-value, graph, wide-column). |
| Schema | Fixed schema requiring predefined structure and data types. | Schema-less or dynamic schema allowing flexible data structures. |
| Scalability | Primarily scales vertically (more powerful server). | Designed to scale horizontally (more servers added). |
| Transactions | Strong support for ACID transactions ensuring data consistency. | Often eventual consistency. Some support ACID, but not all. |
| Complexity of Queries | Supports complex queries using SQL, ideal for deep data analysis. | Query capabilities vary; some models are more limited than SQL. |
| Data Integrity | Enforces data integrity through foreign keys and transactions. | Data integrity can be managed, but differently and less strictly. |
| Use Cases | Ideal for complex applications with many relationships like banking. | Suited for large-scale apps requiring high write speeds, flexible schema. |
| Development Model | Mature technologies with standardized languages (SQL). | Younger, diverse technologies with varied interfaces. |
| Examples | MySQL, PostgreSQL, Oracle, SQL Server. | MongoDB, Redis, Cassandra, Neo4j. |

# Installation and Setup

**2.1 Local installation:**

**Mongodb community edition software can be installed and use locally in Windows and MacOS platforms.**

**​​https://www.mongodb.com/docs/manual/installation/**

**2.2 MongoDB Atlas (Cloud) - Preferred**

* Account Creation - Go to [**MongoDB Atlas**](https://www.mongodb.com/cloud/atlas) **and create an account or log in.**
* Create Cluster - Follow the interface to create a new cluster. Atlas offers a free tier suitable for small projects and learning purposes.
* Configure Security - Set up your IP Whitelist and create database users. Ensure that access to the database is secure.

2.3 **Interactive shell**

You do not need to install anything. Click the Launch button of the in-browser Integrated Development Environment to start the tutorial.

<https://www.mongodb.com/docs/manual/tutorial/getting-started/>

# Developmental Tools

One of the following tools can be used to connect to Mongo databases.

**3.1 Compass**

MongoDB Compass is a powerful GUI for querying, aggregating, and analyzing your MongoDB data in a visual environment - <https://www.mongodb.com/docs/compass/current/#:~:text=MongoDB%20Compass%20is%20a%20powerful,Download%20Compass>

**3.2 MongoDB Shell (mongosh)**

Use the MongoDB Shell to test queries and interact with the data in your MongoDB database. - <https://www.mongodb.com/docs/mongodb-shell/#:~:text=The%20MongoDB%20Shell%2C%20mongosh%20%2C%20is,data%20in%20your%20MongoDB%20database>.

Note: - Compass comes with **embedded mongosh** and the above step can be skipped if compass is used.

**3.3 MongoDB for VSC**

MongoDB provides an extension for [Microsoft Visual Studio Code](https://code.visualstudio.com/) which lets you work with MongoDB and your data directly within your coding environment.

<https://www.mongodb.com/docs/mongodb-vscode/>

# MongoDB Basics

The following table provides a MongoDB cheat sheet, encapsulating key foundational concepts for effectively utilizing MongoDB.

| **Category** | **Command** | **Description** |
| --- | --- | --- |
| Basic Management | mongod | Start MongoDB server |
|  | mongo | Connect to MongoDB via the shell |
| Database Operations | show dbs | List all databases |
|  | use <database\_name> | Create new or switch databases |
|  | db.dropDatabase() | Drop the current database |
| Collection Operations | show collections | List all collections in the current database |
|  | db.createCollection('<name>') | Create a new collection |
|  | db.<collection>.drop() | Drop a collection |
| CRUD Operations | db.<collection>.insertOne({<doc>}) | Insert a single document into a collection |
|  | db.<collection>.insertMany([{<doc1>}, {<doc2>}, ...]) | Insert multiple documents |
|  | db.<collection>.find() | Retrieve all documents in a collection |
|  | db.<collection>.find({<query>}) | Retrieve documents matching the query |
|  | db.<collection>.updateOne({<query>}, {$set: {<field>: <value>}}) | Update first matching document |
|  | db.<collection>.updateMany({<query>}, {$set: {<field>: <value>}}) | Update all matching documents |
|  | db.<collection>.deleteOne({<query>}) | Delete the first document matching the query |
|  | db.<collection>.deleteMany({<query>}) | Delete all documents matching the query |
| Index Management | db.<collection>.createIndex({<field>: 1}) | Create an ascending index on the specified field |
|  | db.<collection>.getIndexes() | List all indexes on the collection |
|  | db.<collection>.dropIndex('<index\_name>') | Drop a specific index |
| Aggregation | db.<collection>.aggregate([{<stage>}, ...]) | Perform aggregation pipeline operations |
| Replication & Sharding | rs.initiate() | Initialize a new replica set |
|  | sh.addShard("<replica\_set>/<hostname>:<port>") | Add a shard to the cluster |
| Security | db.createUser({user:'<username>', pwd:'<password>', roles:[{role:'<role>', db:'<db>'}]}) | Create a new user |
| Performance Tuning | db.<collection>.explain("executionStats") | Explain the performance of a query |
|  | db.setProfilingLevel(1, { slowms: 100 }) | Set the profiling level to log slow queries |
| Backup and Restore | mongodump --db <database\_name> | Create a backup of the database |
|  | mongorestore --db <database\_name> <path> | Restore a database from a backup |
| Monitoring and Statistics | db.serverStatus() | Get server status and statistics |
|  | db.<collection>.stats() | Get statistics about a collection |
| Advanced Querying | db.<collection>.find({<query>}).sort({<field>: 1}).limit(10) | Query, sort, and limit the number of documents returned |
|  | db.<collection>.find({<query>}).skip(10).limit(10) | Skip the first 10 documents and return the next 10 |

# Projects