

MongoDB



i wan tto learn the mongodo so give me all th esteps from an scratch

Great! Learning MongoDB from scratch is a smart move—it's a powerful NoSQL database used widely for modern web applications. Here's a complete step-by-step roadmap to learn MongoDB from scratch:

What is MongoDB?

MongoDB is a **NoSQL** (non-relational) database that stores data in **JSON-like** documents (**BSON** format). It's flexible, scalable, and great for applications with fast-changing or unstructured data.

Step-by-Step MongoDB Learning Guide

✓ Step 1: Understand the Basics of Databases

Before diving into MongoDB, be sure you understand:

- What is a database?
 - Difference between SQL (relational) and NoSQL (non-relational) databases
 - Why choose MongoDB over SQL?
- ♦ Resource: Search "SQL vs NoSQL difference" on YouTube or read articles.

✓ Step 2: Install MongoDB

1. Go to <https://www.mongodb.com/try/download/community>
 2. Choose your OS and download MongoDB Community Server
 3. Follow the installation instructions
 4. Also install **MongoDB Compass** (GUI for MongoDB)
- ♦ Optional: Use **MongoDB Atlas** for cloud-based DB (recommended for beginners).

✓ Step 3: Understand MongoDB Key Concepts

Learn these terms:

Concept	Explanation
Database	A container for collections

Concept	Explanation
Collection	Like a table in SQL, holds multiple documents
Document	JSON-like object (data is stored this way)
Field	A key-value pair inside a document
ObjectId	Unique ID for each document

♦ Example Document:

```
json

{
  "_id": ObjectId("..."),
  "name": "Alice",
  "age": 25,
  "skills": ["MongoDB", "Node.js"]
}
```

✓ Step 4: Learn MongoDB CRUD Operations

CRUD = Create, Read, Update, Delete

Use MongoDB shell, Compass, or driver (Node.js, Python, etc.)

● Create

```
js

db.users.insertOne({ name: "John", age: 30 });
```

● Read

```
js

db.users.find();
db.users.find({ age: { $gt: 25 } });
```

● Update

```
js

db.users.updateOne({ name: "John" }, { $set: { age: 31 } });
```

● Delete

```
js

db.users.deleteOne({ name: "John" });
```

✓ Step 5: Practice Using MongoDB Compass

MongoDB Compass is a GUI tool:

- Connect to local MongoDB server
- Create database and collections
- Insert, find, and update documents visually

✅ Step 6: Learn MongoDB Query Operators

Operators like `$gt`, `$lt`, `$in`, `$or`, `$and`, `$exists`, `$regex` are useful.

♦ Example:

```
js
db.products.find({ price: { $gt: 100 } });
```

✅ Step 7: Work With Data Models

- Embedding vs Referencing documents
- Normalization vs Denormalization
- Example: embed addresses in users or keep in separate collections

✅ Step 8: Aggregation Framework

This is MongoDB's powerful way to analyze and transform data.

Example:

```
js
db.sales.aggregate([
  { $group: { _id: "$category", total: { $sum: "$amount" } } }
]);
```

Learn stages: `$match`, `$group`, `$sort`, `$project`, etc.

✅ Step 9: Use MongoDB With a Programming Language

Most common:

- **Node.js + MongoDB (using Mongoose)**
- **Python + MongoDB (using PyMongo)**

Example (Node.js):

```
js
const MongoClient = require("mongodb").MongoClient;
MongoClient.connect("mongodb://localhost:27017", function(err, client) {
```

```
const db = client.db("test");
db.collection("users").insertOne({ name: "Alice" });
});
```

✅ Step 10: Build a Mini Project

Suggestions:

- User management system
- Blog system
- Task manager (like Todo App)

Use Express.js + MongoDB (Mongoose) to create a full-stack project.

✅ Step 11: Learn MongoDB Atlas (Cloud Version)

- Create a free account at <https://www.mongodb.com/cloud/atlas>
- Create a cluster
- Connect your local app or MongoDB Compass to it

✅ Step 12: Advanced Topics (Optional)

- Indexing and performance
- MongoDB transactions
- Sharding and replication
- Security and access control

Recommended Resources

- [MongoDB University \(Free Courses\)](#)
- YouTube: "MongoDB Full Course" (NetNinja, Academind, Traversy Media)
- Docs: <https://www.mongodb.com/docs/manual/>

Would you like me to give you a learning schedule (e.g., 2 weeks or 1 month plan)? Or would you like a practical mini project idea to apply what you learn?



```
_id: ObjectId('684bcc618ddb8b3ff43a2796'),
Java
name :
pri se:
_id: ObjectId('684bcd688ddb8b3ff43a2797'),
```

```
Python
name :
prise: 45000
db.users. prise:
{ $gt:
i want to find the course whose prose is 30000
```

It looks like you're trying to query a MongoDB collection to find a course where the **prise** (probably meant to be **price**) is 30,000.

Let me help you clean that up and show you how to do it correctly.

✓ Step-by-step fix:

Assuming your collection is called **courses** and your documents look like this:

```
json

{
  _id: ObjectId("684bcc618ddb8b3ff43a2796"),
  name: "Java",
  prise: 30000
},
{
  _id: ObjectId("684bcd688ddb8b3ff43a2797"),
  name: "Python",
  prise: 45000
}
```

✓ Correct MongoDB Query

To find a course with **prise** equal to **30000**, use:

```
js

db.courses.find({ prise: 30000 });
```

This will return:

```
json

{
  _id: ObjectId("684bcc618ddb8b3ff43a2796"),
  name: "Java",
  prise: 30000
}
```

! Tip:

If you meant **price** instead of **prise**, be sure your field names are spelled correctly **in both the document and the query**. MongoDB is case-sensitive and spelling-sensitive.

Would you like help fixing your database or renaming **prise** to **price** in all documents? I can show you how to do that too.



i want an list of queries to be performed on the database and collection so i will be inderstand this is in esay language

Absolutely! Here's a **simple and clear list of MongoDB queries** that will help you understand how to use MongoDB step by step. Each example assumes you're working with a collection named **courses** inside a database like **mydb**.

MongoDB Basic Queries (Easy Language)

1. Show All Databases

```
js
show dbs
```

2. Switch to a Database (or create if not exists)

```
js
use mydb
```

3. Create a Collection (like a table)

```
js
db.createCollection("courses")
```

4. Insert One Course

```
js
db.courses.insertOne({ name: "Java", prise: 30000 })
```

5. Insert Many Courses

```
js
db.courses.insertMany([
  { name: "Python", prise: 45000 },
```

```
{ name: "C++", prise: 25000 },
{ name: "JavaScript", prise: 30000 }
])
```

6. Show All Documents (Read All)

```
js

db.courses.find()
```

7. Find Course Where prise is 30000

```
js

db.courses.find({ prise: 30000 })
```

8. Find Course Where Name is "Python"

```
js

db.courses.find({ name: "Python" })
```

9. Find Courses With prise Greater Than 30000

```
js

db.courses.find({ prise: { $gt: 30000 } })
```

10. Find Courses With prise Less Than or Equal to 30000

```
js

db.courses.find({ prise: { $lte: 30000 } })
```

11. Update One Course

Change prise of Python course to 40000

```
js

db.courses.updateOne(
  { name: "Python" },
  { $set: { prise: 40000 } }
)
```

12. Update Many Courses

Change `prise` of all courses with 30000 to 35000

```
js

db.courses.updateMany(
  { prise: 30000 },
  { $set: { prise: 35000 } }
)
```

13. Delete One Course

```
js

db.courses.deleteOne({ name: "C++" })
```

14. Delete All Courses With `prise > 40000`

```
js

db.courses.deleteMany({ prise: { $gt: 40000 } })
```

15. Sort Courses by `prise` Ascending

```
js

db.courses.find().sort({ prise: 1 })
```

16. Sort Courses by `prise` Descending

```
js

db.courses.find().sort({ prise: -1 })
```

17. Count Total Courses

```
js

db.courses.countDocuments()
```

18. Find Course Name Starts with "J"


```
js
db.courses.find({ name: /^J/ })
```

● 19. Limit Results to 2 Courses

```
js
db.courses.find().limit(2)
```

● 20. Skip First 2 Courses

```
js
db.courses.find().skip(2)
```

✓ Bonus: Aggregation Example (Group by Price)

```
js
db.courses.aggregate([
  { $group: { _id: "$prise", totalCourses: { $sum: 1 } } }
])
```

Do you want me to create a printable **cheat sheet PDF** for this? I can generate one for you!



we can able to create any other Primary inn the mongo db

Great question! Let's clear this up:

? Can we create another Primary key in MongoDB?

✗ No, MongoDB allows only one primary key per document, and that is the `_id` field.

Every document in MongoDB **must** have a unique `_id` field. This `_id` field works like a **primary key** in SQL — it uniquely identifies each document in a collection.

✓ But Here's What You CAN Do:

1. Use Custom `_id` Values (instead of ObjectId)

You can assign your own value to `_id`:

```
js

db.users.insertOne({
  _id: "U001",
  name: "Alice",
  age: 25
});
```

2. Create a Unique Index on Other Fields (Acts Like a Secondary Primary Key)

If you want to prevent duplicate values in another field (e.g., email or username), you can create a **unique index**:

```
js

db.users.createIndex({ email: 1 }, { unique: true });
```

Now, MongoDB will not allow two users with the same email.

Summary:

What You Want	Can You Do It?	How?
Add second primary key	✗ No	Only <code>_id</code> is primary
Use custom <code>_id</code>	✓ Yes	Set <code>_id</code> manually
Enforce uniqueness on other fields	✓ Yes	Use <code>createIndex(..., { unique: true })</code>

Would you like an example where `_id` is custom and `email` is also unique?