MongoDB Notes

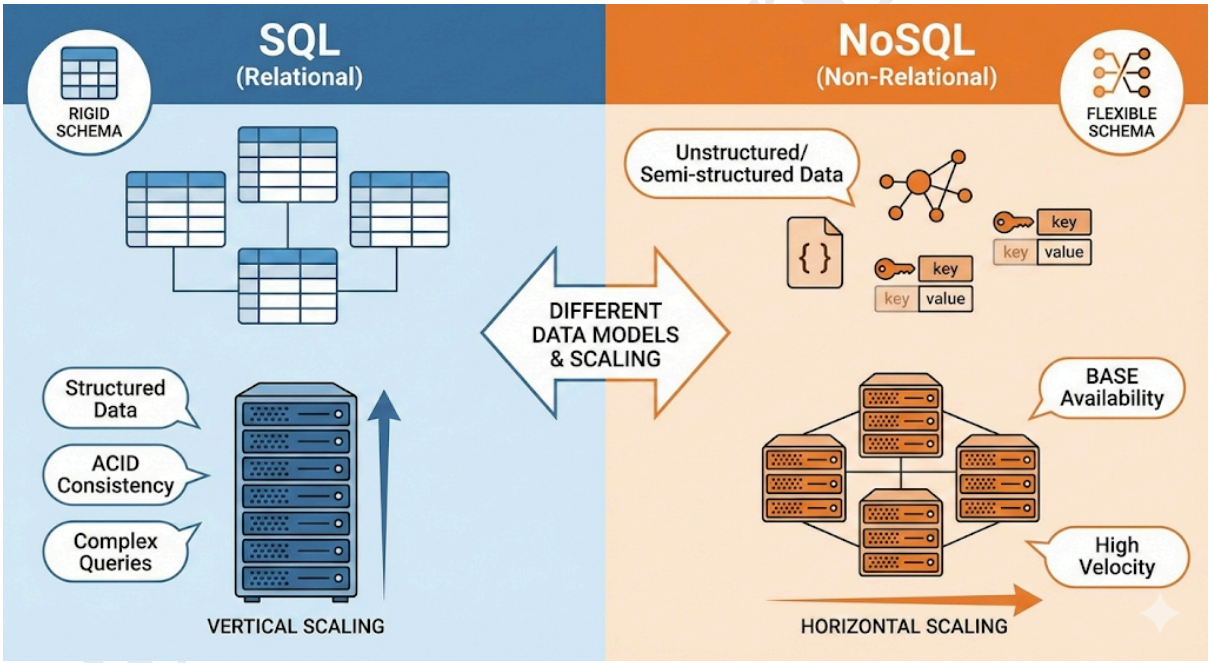
-Mo Fahim Raj

Introduction to MongoDB:

MongoDB is a product and company that builds it :

- The names comes from the work homogenous
- No SQL Document Database
- Dynamic Schemas
- High Performance
- Scalability
- High Availability
- Rich Query Capabilities
- Geospatial and Text Search
- CrossPlatform Compatibility
- Easy Integration

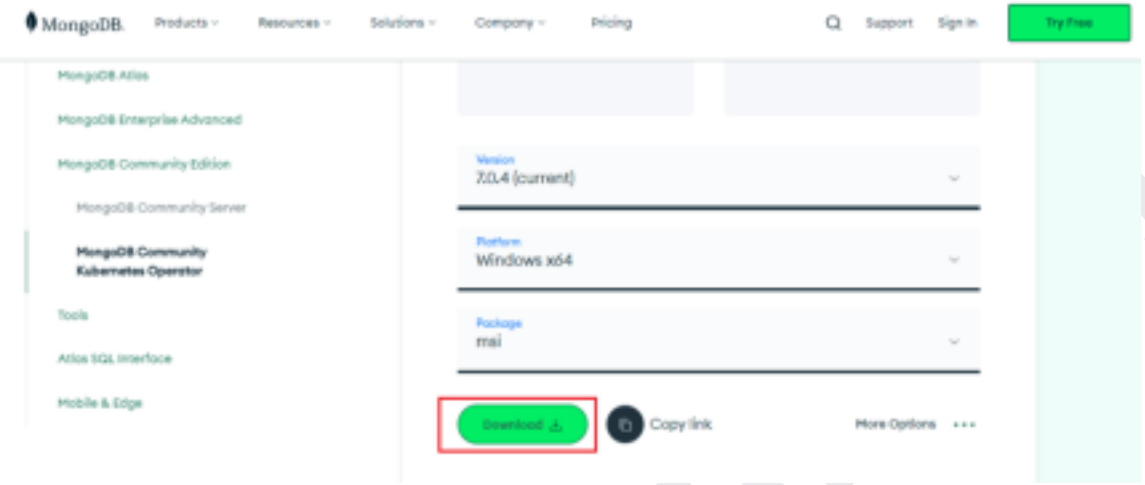
SQL V/S NoSQL



Installation of MongoDB(Steps by Steps process)

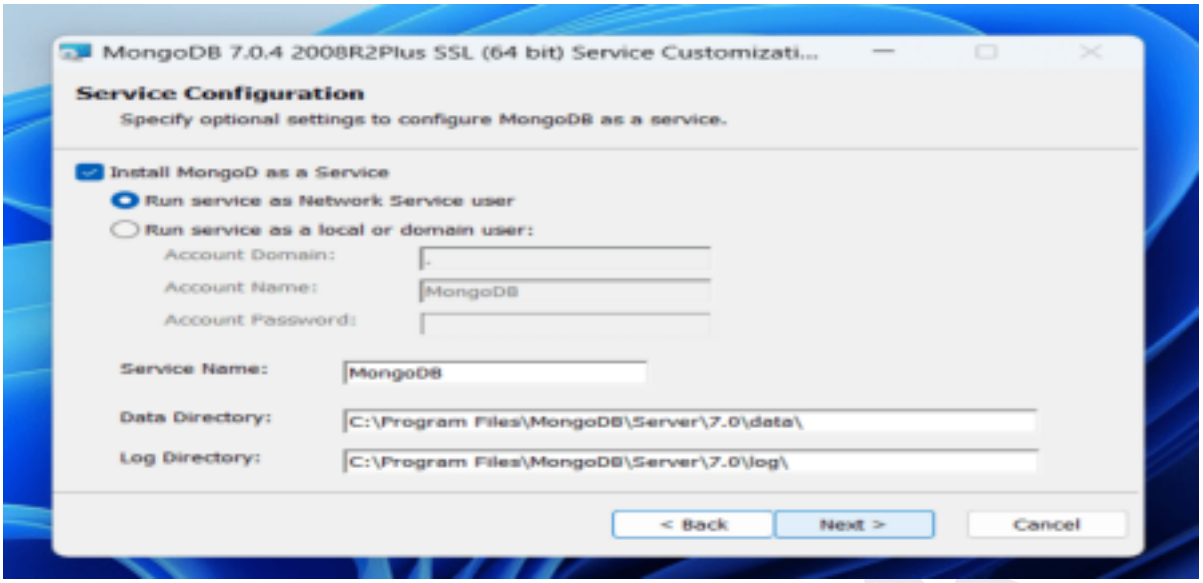
To install MongoDB on Windows, first, download the MongoDB server and then install the MongoDB shell. The Steps below explain the installation process in detail and provide the required resources for the smooth download and install MongoDB.

Step 1: Go to the MongoDB Download Center to download the MongoDB Community Server.

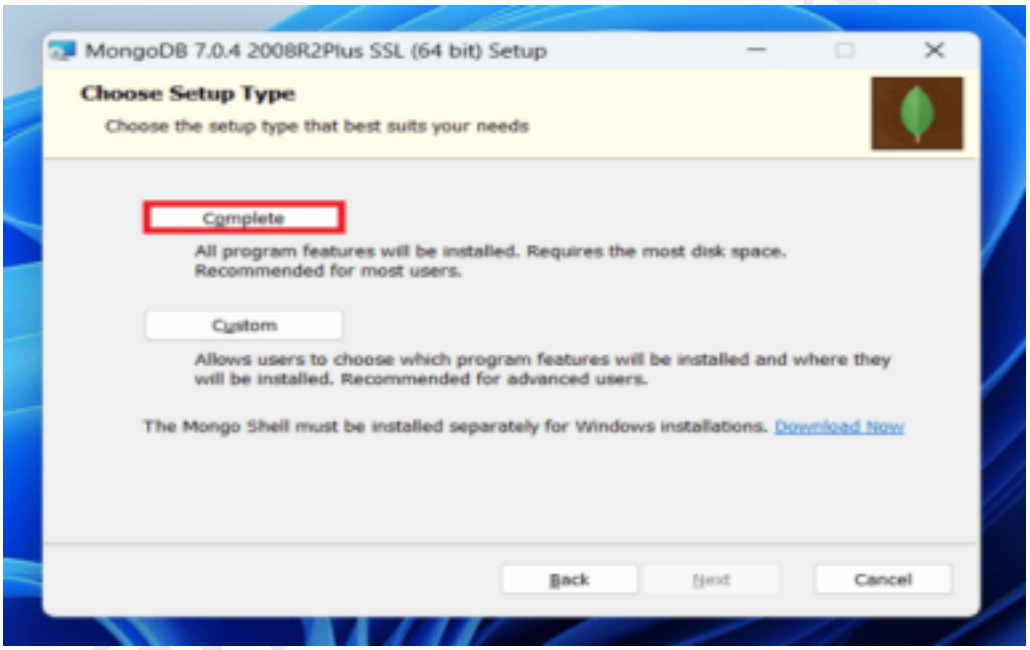


Step 2: When the download is complete open the msi file and click the *next button* in the startup screen:

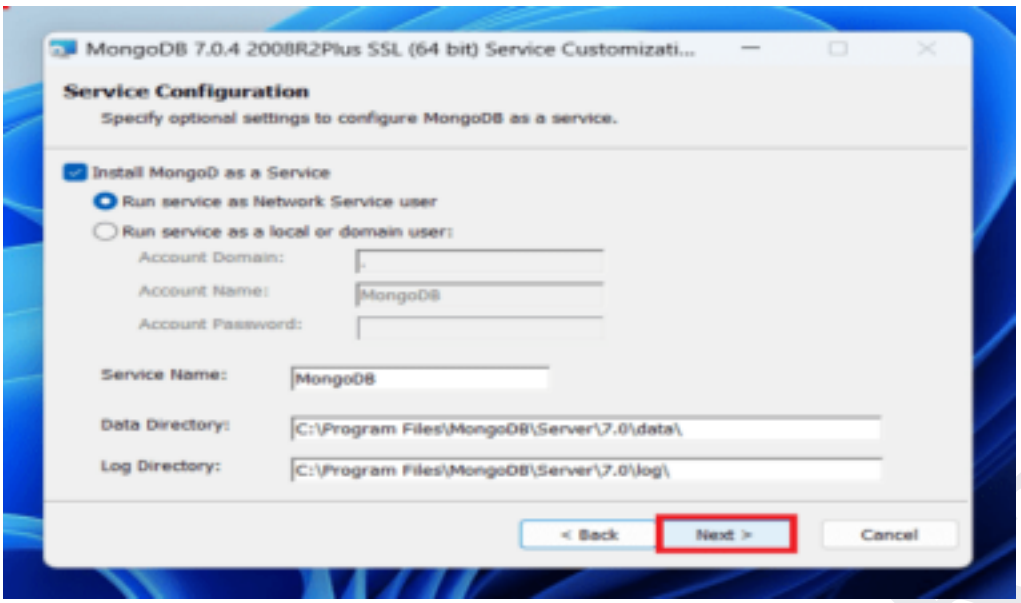
Step 3: Now accept the **End-User License Agreement** and click the next button:



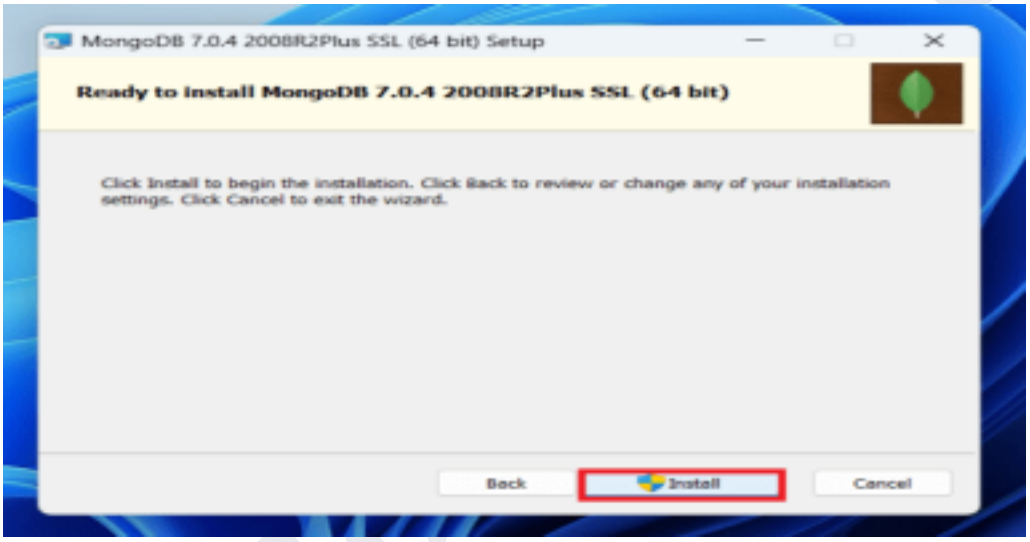
Step 4: Now select the **complete option** to install all the program features. Here, if you can want to install only selected program features and want to select the location of the installation, then use the **Custom option***:*



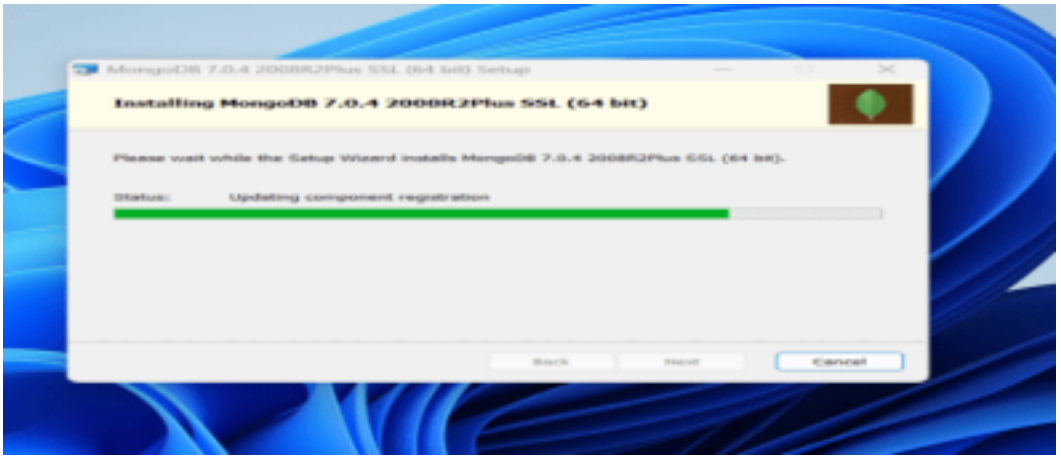
Step 5: Select "**Run service as Network Service use**r" and copy the path of the data directory. **Click Next**:



Step 6: Click the **Install button** to start the MongoDB installation process:



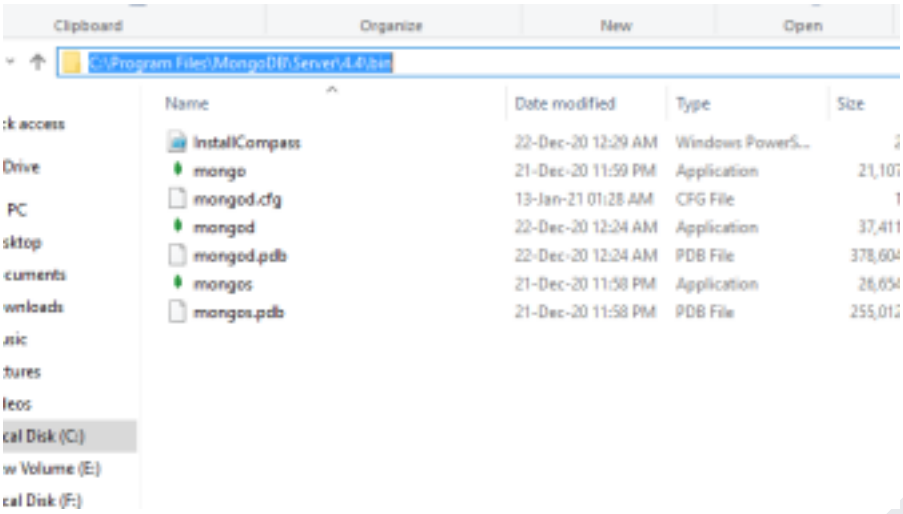
Step 7: After clicking on the install button installation of MongoDB begins:



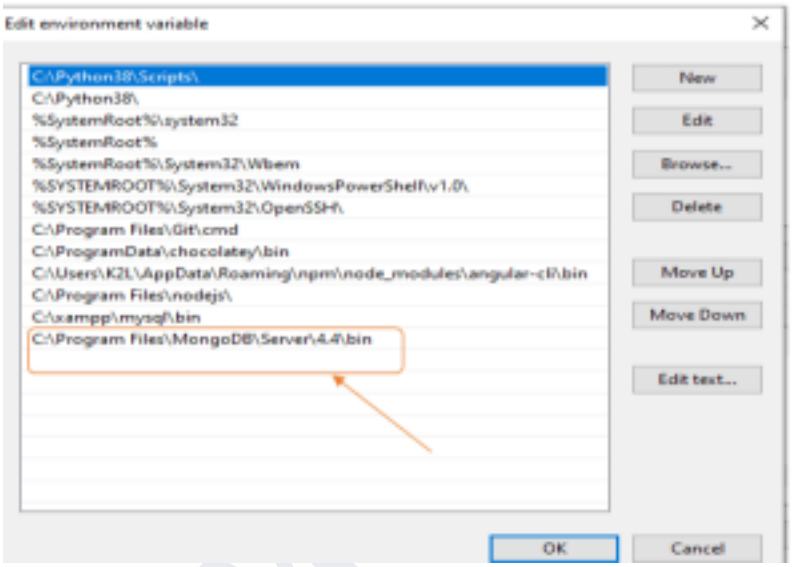
Step 8: Now click the **Finish button** to complete the MongoDB installation process:

Step 9: Now we go to the location where MongoDB installed in step 5 in your

system and copy the **bin** path:



Step 10: Now, to create an environment variable open system **properties >> Environment Variable >> System variable >> path >> Edit Environment variable** and paste the copied link to your environment system and **click Ok**:



Step 11: After setting the environment variable, we will run the MongoDB server, i.e. **mongod**. So, open the **command prompt** and run the following command:

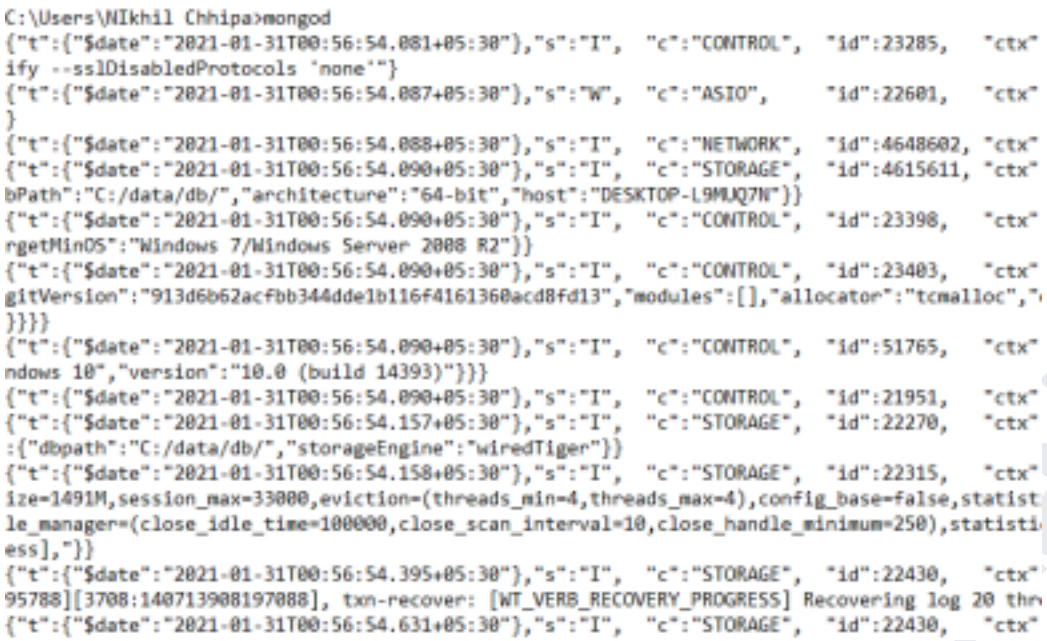
```
mongod
```

When you run this command you will get an error i.e. **C:/data/db/ not found**.

Step 12: Now, Open **C** drive and create a folder named "**data**" inside this folder create another folder named "db". After creating these folders. Again open the command prompt and run the following command:

```
mongod
```

Now, this time the MongoDB server(i.e., mongod) will run successfully.



Run mongo Shell

Step 13: Now we are going to connect our server (mongod) with the mongo shell. So, keep that mongod window and open a new command prompt window and write **mongo.** Now, our mongo shell will successfully connect to the mongod.

MongoDB Sheet

1. Create Database

MongoDB creates a database when you start using it.

use myDatabase

2. Rename Database

MongoDB **does not support direct rename**.
Workaround:

1. Create new DB
2. Copy collections
3. Drop old DB

```
use oldDB
db.copyDatabase("oldDB", "newDB")
use oldDB
db.dropDatabase()
```

3. Delete Database

Deletes the current database:

```
use myDatabase
db.dropDatabase()
```

Collections

4. Show Collections

```
show collections
```

5. Create Collection

```
db.createCollection("students")
```

6. Drop Collection

```
db.students.drop()
```

MongoDB Data Types

Type	Example
String	"name": "Fahim"
Number	"age": 22
Double	"gpa": 7.82
Boolean	"isActive": true

Array	"skills": ["Java", "Node"]
Object	"address": { city: "Mumbai", zip: 400001 }
Null	"middleName": null
ObjectId	"_id": ObjectId("...")
Date	"createdAt": ISODate()
Timestamp	Timestamp()
Binary Data	For files/images
Regular Expression	/pattern/

Insert Documents

7. InsertOne

```
db.students.insertOne({
 name: "Fahim",
 age: 22,
 city: "Mumbai"
})
```

8. Create Index

```
db.students.createIndex({ name: 1 })    // ascending
db.students.createIndex({ age: -1 })    // descending
db.students.createIndex({ email: 1 }, { unique: true })
```

9. Insert Array

```
db.students.insertOne({
 name: "Ayaan",
 skills: ["JavaScript", "MongoDB", "React"]
})
```

10. Insert Object

```
db.students.insertOne({
```

```
 name: "Sara",
 address: {
  city: "Delhi",
  pin: 110001
 }
})
```

11. Insert Many

```
db.students.insertMany([
 { name: "John", age: 21 },
 { name: "Maria", age: 23 },
 { name: "Alex", age: 20 }
])
```

UPDATE / RENAME

1. Update One Document

```
db.students.updateOne(
  { name: "Fahim" },
  { $set: { age: 23 } }
)
```

2. Update Embedded (Nested) Document

```
db.students.updateOne(
  { name: "Fahim" },
  { $set: { "address.city": "Pune" } }
)
```

3. Update ALL Documents

```
db.students.updateMany(
  {},
  { $set: { isActive: true } }
```

```
)
```

UNSET / DELETE FIELDS

4. Unset (Delete one field from one document)

```
db.students.updateOne(
  { name: "Fahim" },
  { $unset: { age: "" } }
)
```

5. Delete One Document

```
db.students.deleteOne({ name: "Fahim" })
```

6. Delete ONE Field in ALL Documents

```
db.students.updateMany(
  {},
  { $unset: { age: "" } }
)
```

7. Delete Field in ALL Documents (Same as above)

```
db.students.updateMany(
  {},
  { $unset: { fieldName: "" } }
)
```

8. Delete MULTIPLE Fields in ALL Documents

```
db.students.updateMany(
  {},
  {
    $unset: {
      age: "",
      city: "",
      pin: ""
    }
  }
```

```
)
```

Insert Document (General Template)

```
db.students.insertOne({
  name: "Ayaan",
  age: 21,
  skills: ["MongoDB", "Node"],
  address: { city: "Delhi", pin: 110034 }
})
```

FIND DATA

9. find()

Returns multiple documents:

```
db.students.find()
```

With condition:

```
db.students.find({ age: 21 })
```

10. findOne()

```
db.students.findOne({ name: "Ayaan" })
```

11. Query Document (Simple Query)

```
db.students.find({ age: { $gt: 20 } })
```

12. Query Nested Objects

```
db.students.find({ "address.city": "Mumbai" })
```

PROJECTION (Select specific fields)

Include fields

```
db.students.find(
  {},
  { name: 1, age: 1, _id: 0 }
)
```

Exclude fields

```
db.students.find(
  {},
  { age: 0 }
)
```

UPDATE OPERATIONS

13. updateOne()

```
db.students.updateOne(
  { name: "John" },
  { $set: { age: 25 } }
)
```

14. Update Nested Object

```
db.students.updateOne(
  { name: "John" },
  { $set: { "address.pin": 400020 } }
)
```

15. UpdateMany

```
db.students.updateMany(
  { age: { $lt: 25 } },
  { $set: { category: "young" } }
)
```

Adding New Field in All Documents

```
db.students.updateMany(
  {},
  { $set: { joined: "2025" } }
)
```

UPSERT (Update + Insert)

If document **exists → update**
If **not exists → insert new**

```
db.students.updateOne(
  { name: "Sara" },
  { $set: { city: "Pune", age: 22 } },
  { upsert: true }
)
```

DELETE DOCUMENTS

1. deleteOne()

```
db.students.deleteOne({ name: "Fahim" })
```

Deletes **first matching document**.

2. deleteMany()

```
db.students.deleteMany({ age: { $lt: 20 } })
```

Deletes **all matching documents**.

3. remove() (Deprecated)

```
db.students.remove({ city: "Mumbai" })
```

Use **deleteOne / deleteMany** instead.

MONGODB OPERATORS (with Examples)

1. COMPARISON QUERY OPERATORS

$eq — Equal

```
db.students.find({ age: { $eq: 22 } })
```

$ne — Not equal

```
db.students.find({ city: { $ne: "Mumbai" } })
```

$gt — Greater than

```
db.students.find({ age: { $gt: 18 } })
```

$gte — Greater than or equal

```
db.students.find({ marks: { $gte: 90 } })
```

$lt — Less than

```
db.students.find({ age: { $lt: 30 } })
```

$lte — Less than or equal

```
db.students.find({ age: { $lte: 25 } })
```

$in — Value in array

```
db.students.find({ city: { $in: ["Delhi", "Mumbai"] } })
```

$nin — Value NOT in array

```
db.students.find({ age: { $nin: [20, 25] } })
```

2. LOGICAL QUERY OPERATORS

$and

```
db.students.find({
  $and: [{ age: { $gt: 20 } }, { city: "Delhi" }]
})
```

$or

```
db.students.find({
  $or: [{ city: "Mumbai" }, { city: "Delhi" }]
})
```

$nor (NOT OR)

```
db.students.find({
  $nor: [
    { city: "Delhi" },
    { age: 22 }
  ]
})
```

$not

```
db.students.find({
  age: { $not: { $gt: 25 } }
})
```

3. ELEMENT QUERY OPERATORS

$exists — Field present or not

```
db.students.find({ age: { $exists: true } })
```

$type — Field by data type

```
db.students.find({ age: { $type: "number" } })
```

4. EVALUATION QUERY OPERATORS

$expr — Use expressions

```
db.sales.find({
  $expr: { $gt: ["$amount", "$tax"] }
})
```

$mod — Modulus

```
db.students.find({ roll: { $mod: [2, 0] } }) // even numbers
```

$regex — Pattern search

```
db.students.find({ name: { $regex: /^A/ } })
```

$text — Text search (requires text index)

```
db.products.find({ $text: { $search: "phone" } })
```

$where — JavaScript condition

```
db.students.find({
  $where: "this.age > 20"
})
```

5. ARRAY QUERY OPERATORS

$all — Matches all array values

```
db.students.find({ skills: { $all: ["JS", "MongoDB"] } })
```

$size — Array length

```
db.students.find({ skills: { $size: 3 } })
```

$elemMatch — Match condition inside array

```
db.students.find({
```

```
  scores: { $elemMatch: { subject: "Math", marks: { $gt: 90 } } }
})
```

$comment — Add comment for debugging

```
db.students.find({ age: 21 }).comment("Age filter query")
```

FIELD UPDATE OPERATORS

$currentDate

```
db.students.updateOne({}, { $currentDate: { updatedAt: true } })
```

$inc — Increment

```
db.students.updateOne({ name: "Ayaan" }, { $inc: { marks: 5 } })
```

$max — Update only if new value is greater

```
db.students.updateOne({ name: "Ayaan" }, { $max: { marks: 90 } })
```

$mul — Multiply

```
db.products.updateOne({ id: 1 }, { $mul: { price: 1.1 } })
```

$rename — Rename field

```
db.students.updateMany({}, { $rename: { "city": "location" } })
```

$set — Add/Update field

```
db.students.updateOne({ name: "John" }, { $set: { age: 25 } })
```

$setOnInsert — Only when upsert inserts

```
db.students.updateOne(
  { name: "Sara" },
  { $set: { city: "Delhi" }, $setOnInsert: { joined: 2025 } },
  { upsert: true }
)
```

$unset — Remove field

```
db.students.updateOne({ name: "John" }, { $unset: { age: "" } })
```

ARRAY UPDATE OPERATORS

1. $ — Positional Operator

Updates first matching array element.

```
db.students.updateOne(
  { name: "Ayaan", skills: "MongoDB" },
  { $set: { "skills.$": "Mongoose" } }
)
```

2. $[ ] — All array elements

```
db.students.updateOne(
  { name: "Ayaan" },
  { $set: { "scores.$[]": 100 } }
)
```

3. $[identifier] — Filtered array update

```
db.students.updateMany(
  {},
  { $set: { "scores.$[high].bonus": true } },
  { arrayFilters: [{ "high.marks": { $gt: 90 } }] }
)
```

4. $addToSet — Add unique item

```
db.students.updateOne(
  { name: "John" },
  { $addToSet: { skills: "Node" } }
)
```

5. $pop — Remove first/last

```
db.students.updateOne({ name: "John" }, { $pop: { skills: 1 } })  // last
db.students.updateOne({ name: "John" }, { $pop: { skills: -1 } }) // first
```

6. $pull — Remove value

```
db.students.updateOne(
  { name: "John" },
  { $pull: { skills: "MongoDB" } }
```

```
)
```

7. $push — Add value

```
db.students.updateOne(
  { name: "John" },
  { $push: { skills: "React" } }
)
```

8. $pullAll — Remove multiple values

```
db.students.updateOne(
  { name: "John" },
  { $pullAll: { skills: ["MongoDB", "React"] } }
)
```

9. $each — Push multiple values

```
db.students.updateOne(
  { name: "John" },
  { $push: { skills: { $each: ["Node", "Express"] } } }
)
```

10. $position — Insert at specific index

```
db.students.updateOne(
  { name: "John" },
  { $push: { skills: { $each: ["Python"], $position: 1 } } }
)
```

11. $slice — Keep only limited array size

```
db.students.updateOne(
  { name: "John" },
  { $push: { scores: { $each: [100], $slice: 5 } } }
)
```

12. $sort — Sort array

```
db.students.updateOne(
  { name: "John" },
```

```
  { $push: { scores: { $each: [], $sort: 1 } } }
)
```

REMAINING MONGODB METHODS

1. find() & findOne()

find() → returns multiple documents

```
db.students.find({ city: "Mumbai" })
```

findOne() → returns first matching document

```
db.students.findOne({ age: 22 })
```

2. findAndModify() *(Old method—still usable)*

```
db.students.findAndModify({
  query: { name: "Fahim" },
  update: { $set: { age: 25 } },
  new: true
})
```

3. findOneAndDelete()

```
db.students.findOneAndDelete({ name: "Ayaan" })
```

4. findOneAndReplace()

```
db.students.findOneAndReplace(
  { name: "John" },
  { name: "John", age: 30, city: "Pune" }
)
```

5. findOneAndUpdate()

```
db.students.findOneAndUpdate(
  { name: "John" },
  { $set: { age: 28 } },
  { returnDocument: "after" }
```

)

6. sort()

Sort ascending (1) or descending (-1):

```
db.students.find().sort({ age: -1 })
```

7. count() (Deprecated but still works)

```
db.students.count({ city: "Mumbai" })
```

8. countDocuments() (Recommended)

```
db.students.countDocuments({ city: "Mumbai" })
```

9. limit()

Limit number of results:

```
db.students.find().limit(5)
```

10. distinct()

Find unique values:

```
db.students.distinct("city")
```

11. LENGTH (Count array size)

Find documents where array length = X

```
db.students.find({ skills: { $size: 3 } })
```

To get array length inside projection

(Not direct—needs aggregation)

```
db.students.aggregate([
  { $project: { name: 1, skillCount: { $size: "$skills" } } }
])
```

12. Find by ObjectId / _id

```
db.students.find({ _id: ObjectId("676fac24321b") })
```

13. comment()

Add a comment for debug:

```
db.students.find({ age: 21 }).comment("Checking age filter")
```

14. validate() (Check collection consistency)

```
db.runCommand({ validate: "students" })
```

15. Check Storage of a Collection

```
db.students.stats()
```

Shows:

- storage size
- total index size
- avg object size

INDEXING IN MONGODB

Index makes searching fast.

1. Create Index

```
db.students.createIndex({ name: 1 })
```

2. Unique Index

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

3. Compound Index

```
db.students.createIndex({ age: 1, city: -1 })
```

4. Text Index

```
db.products.createIndex({ name: "text", description: "text" })
```

5. Drop Index

```
db.students.dropIndex("name_1")
```

6. List All Indexes

```
db.students.getIndexes()
```

7. TTL Index (Expire documents)

```
db.logs.createIndex(
  { createdAt: 1 },
  { expireAfterSeconds: 3600 }
)
```

AGGREGATION IN MONGODB

Aggregation = processing data in pipeline stages.

Basic Structure

```
db.students.aggregate([
  { $match: { city: "Delhi" } },
  { $group: { _id: "$city", total: { $sum: 1 } } }
])
```

1. $match → Filter

```
{ $match: { age: { $gt: 20 } } }
```

2. $group → Grouping

```
{
  $group: {
    _id: "$city",
    count: { $sum: 1 }
  }
}
```

3. $sort

```
{ $sort: { age: -1 } }
```

4. $project → Select specific fields

```
{
  $project: {
```

```
    name: 1,
    city: 1,
    _id: 0
  }
}
```

5. $limit

```
{ $limit: 5 }
```

6. $skip

```
{ $skip: 10 }
```

7. $count

```
{ $count: "totalStudents" }
```

8. $lookup → JOIN

```
{
  $lookup: {
    from: "courses",
    localField: "courseId",
    foreignField: "_id",
    as: "courseDetails"
  }
}
```

9. $unwind → Break array

```
{ $unwind: "$skills" }
```

10. $addFields → Add fields

```
{ $addFields: { passed: true } }
```

11. $sum / $avg / $min / $max

```
$sum: "$marks"
$avg: "$marks"
```

```
$min: "$marks"
$max: "$marks"
```

12. Full Example

```
db.students.aggregate([
  { $match: { city: "Delhi" } },
  { $group: { _id: "$city", avgAge: { $avg: "$age" } } },
  { $sort: { avgAge: -1 } },
  { $project: { _id: 0, avgAge: 1 } }
])
```