MongoDB Update & Delete Operations

Databases/Collections Used

Database	Collection
db	movies

📘 MongoDB Command Summary with Explanation

📋 Databases/Collections Used

Database	Collection
db	movies

🔍 1. Basic Read Commands

🔢 Count All Documents

```
db.movies.countDocuments()
```

Returns the total number of documents in the movies collection.

Utility Commands

🔁 Show Databases

```
show dbs
```

📂 Use/Select Database

```
use moviesDB
```

📁 Show Collections in Current DB

```
show collections
```

📄 Find All Documents

```
db.movies.find()
```

Returns all documents.

📄 Find First Document

```
db.movies.findOne()
```

Returns the first document in the collection.

🎯 2. Filtering Documents

Find by Field Value

```
db.movies.find({ year: 1998 })
```

Returns all documents where `year` is 1998.

Count Matching Documents

```
db.movies.find({ year: 1998 }).count()
```

⚙️ 3. Comparison Operators

Operator	Description
`$eq`	Equal
`$ne`	Not Equal
`$lt`	Less Than
`$lte`	Less Than or Equal To

$gt	Greater Than
$gte	Greater Than or Equal To

Example:

```
db.movies.find({ year: { $gt: 2000 } })
```

🧮 4. Logical Operators

AND Condition

```
db.movies.find({ type: "series", year: 2004 })
```

or

```
db.movies.find({ $and: [ { type: "series" }, { year: 2004 } ] })
```

OR Condition

```
db.movies.find({ $or: [ { type: "series" }, { year: 2004 } ] })
```

🎭 5. Array Query Operators

$in — Match Any Value in Array

```
db.movies.find({ cast: { $in: ["Tom Cruise"] } })
```

$nin — Exclude Values

```
db.movies.find({ cast: { $nin: ["Tom Cruise"] } })
```

$and with $in

```
db.movies.find({
  $and: [
    { cast: { $in: ["Tom Cruise"] } },
    { cast: { $in: ["Rebecca Ferguson"] } }
  ]
```

```
})
```

`$all` — Match All Elements

```
db.movies.find({ cast: { $all: ["Tom Cruise", "Richard Masur"] } })
```

Complex Logic

```
db.movies.find({
  $or: [
    { cast: { $in: ["Rebecca Ferguson"] } },
    {
      $and: [
        { cast: { $in: ["Tom Cruise"] } },
        { year: 2004 }
      ]
    }
  ]
})
```

🔗 6. Query Nested Fields

```
db.movies.find({ "imdb.rating": { $gt: 9 } })
```

Access nested fields using `"field.subfield"` format.

🧾 7. Projection (Select Specific Fields)

Include Specific Fields

```
db.movies.findOne(
  { "imdb.rating": { $gt: 9 } },
  { title: 1, "imdb.rating": 1 }
```

```
)
```

Exclude Specific Fields

```
db.movies.findOne(
  { "imdb.rating": { $gt: 9 } },
  { title: 0, "imdb.rating": 0 }
)
```

📚 8. Pagination

```
db.movies.find({}, { title: 1 }).limit(3).skip(3)
```

Skips the first 3 and returns the next 3 documents.

🔄 9. Update Commands

updateOne()

```
db.movies.updateOne(
  { _id: ObjectId("573a1396f29313caabce4a9a") },
  { $set: { "imdb.rating": 8.4 } }
)
```

updateMany()

```
db.movies.updateMany(
  { year: { $gt: 1990 } },
  { $set: { status: "updated" } }
)
```

$inc — Increment Field

```
db.movies.updateMany(
  { year: { $gt: 1990 } },
  { $inc: { "imdb.rating": 1 } })
```

🛠️ 10. Update Operators

Operator	Description
`$set`	Set or add a new field
`$inc`	Increment a numeric field
`$unset`	Remove a field
`$push`	Add value to array
`$pop`	Remove first (`-1`) or last (`1`) from array
`$pull`	Remove specific value from array
`$pullAll`	Remove multiple values from array
`$addToSet`	Add value to array only if it doesn't exist

❌ 11. Delete Commands

deleteOne()

```
db.movies.deleteOne({ title: "The Godfather" })
```

deleteMany()

```
db.movies.deleteMany({ year: { $lt: 1980 } })
```

🔁 12. Replace Document

replaceOne()

```
db.movies.replaceOne(
  { title: "Old Movie" },
  { title: "New Movie", year: 2024, imdb: { rating: 9.5 } })
```

Yes! Besides the **basic CRUD operations** you've already covered (Create, Read, Update, Delete), MongoDB supports many more powerful and commonly used commands. Here's a categorized list of **additional important MongoDB commands** you should know:

🧱 13. Insert Commands

➕ Insert One Document

```
db.movies.insertOne({
  title: "Oppenheimer",
  year: 2023,
  type: "movie",
  imdb: { rating: 8.9 }
})
```

➕ Insert Multiple Documents

```
db.movies.insertMany([
  { title: "Barbie", year: 2023 },
  { title: "Tenet", year: 2020 }
])
```

🧠 14. Aggregation Framework (Powerful Data Processing)

🔍 Simple Aggregation Example

```
db.movies.aggregate([
  { $match: { year: 2023 } },
  { $group: { _id: "$type", total: { $sum: 1 } } }
])
```

- `$match`: Filters documents (like `find`).
- `$group`: Groups by a field and calculates total count.

📊 15. Sorting, Limiting, Skipping

🔽 Sort Results

```
db.movies.find().sort({ year: -1 })  // Descending
```

🔼 Ascending Sort

```
db.movies.find().sort({ year: 1 })  // Ascending
```

🎚️ Limit & Skip (Pagination)

```
db.movies.find().skip(10).limit(5)
```

🔍 16. Indexing for Performance

📈 Create Index

```
db.movies.createIndex({ title: 1 })
```

🔎 View Indexes

```
db.movies.getIndexes()
```

❌ Drop Index

```
db.movies.dropIndex("title_1")
```

🔠 17. Text Search

🆕 Create Text Index

```
db.movies.createIndex({ title: "text", plot: "text" })
```

🔍 Search Using Text

```
db.movies.find({ $text: { $search: "mission impossible" } })
```

🛡️ 18. User Management (Admin Only)

👤 Create User

```
db.createUser({
  user: "admin",
  pwd: "password123",
  roles: [ { role: "readWrite", db: "moviesDB" } ]
})
```

🧾 Show Current Users

```
db.getUsers()
```