Day 4 – MongoDB Queries

**1. Starting MongoDB & Switching Database**

**Definition**: MongoDB databases are containers for collections. You must switch to or create a DB before performing operations.

mongod

mongosh

use school

**2. Create Collection & Insert Data**

**Definition**: Collections are like tables in SQL. Documents are JSON-like records.

db.students.insertOne({ name: "John", age: 18, grade: "A" })

db.students.insertMany([

{ name: "Ravi", age: 19, grade: "B" },

{ name: "Priya", age: 20, grade: "A" },

{ name: "Amit", age: 21, grade: "C" }

])

**3. Read / Query Data**

**Definition**: Use .find() to retrieve documents.

db.students.find()

db.students.find().pretty()

db.students.find({ grade: "A" })

**4. Query Operators**

**Definition**: Operators refine searches using conditions.

db.students.find({ age: { $gt: 18 } }) // Greater than 18

db.students.find({ age: { $gte: 18, $lte: 21 } }) // Between 18 and 21

db.students.find({ grade: { $in: ["A", "B"] } }) // A or B

**5. Update Operations**

**Definition**: Used to modify existing documents.

db.students.updateOne({ name: "Ravi" }, { $set: { grade: "A+" } })

db.students.updateMany({ grade: "C" }, { $set: { grade: "B" } })

**6. Delete Operations**

**Definition**: Removes documents from a collection.

db.students.deleteOne({ name: "Amit" })

db.students.deleteMany({ grade: "B" })

**7. Sorting and Limiting**

**Definition**: Organize results and restrict result count.

db.students.find().sort({ age: 1 }) // Ascending

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

db.students.find().limit(2) // Only 2 results

**8. Projection**

**Definition**: Choose which fields to return.

db.students.find({}, { name: 1, age: 1, \_id: 0 }) // Only name and age

**9. Aggregation Framework**

**Definition**: Performs calculations like sum, avg, group.

// Count by grade

db.students.aggregate([

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

])

// Average age

db.students.aggregate([

{ $group: { \_id: null, averageAge: { $avg: "$age" } } }

])

**10. Indexing**

**Definition**: Speeds up query performance.

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

db.students.getIndexes() // Show all indexes

**11. Embedded Documents and Arrays**

**Definition**: Nest documents or use arrays inside one document.

// Embedded document

db.students.insertOne({

name: "Sita",

age: 22,

address: { city: "Delhi", zip: "110001" }

})

// Query embedded field

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

// Array

db.students.insertOne({

name: "Ram",

subjects: ["Math", "English", "Science"]

})

// Query array

db.students.find({ subjects: "Math" })

**12. Count Documents**

**Definition**: Get number of documents in a collection.

db.students.countDocuments({})

db.students.countDocuments({ grade: "A" })

**13. Rename Fields**

**Definition**: Modify field names in all documents.

db.students.updateMany({}, { $rename: { "grade": "performance" } })

**14. Drop Collection or Database**

**Definition**: Completely remove a collection or DB.

db.students.drop() // Drop collection

db.dropDatabase() // Drop entire database

**15. Backup and Restore (System Commands)**

**Definition**: Save/restore database externally.

mongodump --db=school --out=C:\backup\school

mongorestore --db=school C:\backup\school

**16. Exit Mongo Shell**

exit

* **Case Study – Student Management System**

👉[Click Here](https://drive.google.com/file/d/1vS67-Rqt9q4MOgyjtwW9abYdiBtLV0Sy/view?usp=drive_link)