1.

use universityDB  
  
db.students.insertMany([

{ roll: 101, name: "Arun", dept: "CSE", attendance: 90 },

{ roll: 102, name: "Meera", dept: "ECE", attendance: 87 },

{ roll: 103, name: "Vikram", dept: "MECH", attendance: 82 },

{ roll: 104, name: "Divya", dept: "EEE", attendance: 89 },

{ roll: 105, name: "Ravi", dept: "CIVIL", attendance: 85 }

])

db.students.find().pretty()

db.students.updateOne(

{ roll: 101 },

{ $set: { attendance: 92 } }

)

db.students.deleteOne({ roll: 102 })

2.

use companyDB

db.employees.insertMany([

{ name: "Asha", dept: "Sales", salary: 50000 },

{ name: "Rahul", dept: "Tech", salary: 70000 },

{ name: "Neha", dept: "Tech", salary: 65000 },

{ name: "Sunil", dept: "Marketing", salary: 48000 },

{ name: "Divya", dept: "Sales", salary: 52000 },

{ name: "Ajay", dept: "HR", salary: 40000 }

])  
  
db.employees.find({ dept: "Tech", salary: { $gt: 60000 } })

db.employees.find({}, { \_id: 0, name: 1, dept: 1 })

db.employees.find({ dept: { $in: ["Sales", "Marketing"] } })

3.

// app.js

const { MongoClient } = require("mongodb");

const uri = "mongodb://localhost:27017"; // MongoDB URI

const client = new MongoClient(uri); // Create MongoClient instance

async function run() {

try {

// Connect to MongoDB

await client.connect();

// Access database and collection

const db = client.db("employeeDB");

const users = db.collection("users");

// Insert one document

await users.insertOne({ name: "Nina", role: "Admin" });

// Fetch all users

const allUsers = await users.find().toArray();

console.log("All Users:", allUsers);

} catch (error) {

console.error("Error:", error);

} finally {

// Close the connection

await client.close();

}

}

// Run the function

run();

4.

use blogApp

db.blogs.insertMany([

{

title: "Learning MongoDB",

author: "Ravi",

content: "MongoDB is a NoSQL database that stores data in JSON-like documents.",

tags: ["database", "NoSQL", "MongoDB"],

createdAt: new Date()

},

{

title: "Getting Started with Express",

author: "Asha",

content: "Express.js is a minimal and flexible Node.js web framework.",

tags: ["Node.js", "Express", "Backend"],

createdAt: new Date()

},

{

title: "Understanding REST APIs",

author: "Karan",

content: "REST APIs allow communication between client and server using HTTP.",

tags: ["API", "REST", "Web"],

createdAt: new Date()

}

])

db.blogs.updateOne(

{ title: "Understanding REST APIs" },

{ $set: { title: "REST APIs Explained" } }

)

db.blogs.deleteOne({ author: "Ravi" })