**Insert Documents (insertOne, insertMany)**

Insert a single document into the "users" collection:

db.users.insertOne({ name: "Alice", age: 25, city: "New York" });

Insert multiple documents into the "users" collection:

db.users.insertMany([

{ name: "Bob", age: 30, city: "Los Angeles" },

{ name: "Charlie", age: 35, city: "Chicago" }

]);

Insert a product with an array field:

db.products.insertOne({ name: "Laptop", price: 1200, tags: ["electronics", "computer"] });

Insert a document with an embedded document:

db.employees.insertOne({ name: "John", address: { city: "Boston", zip: "02118" } });

Insert a document with a timestamp:

db.logs.insertOne({ event: "login", timestamp: new Date() });

**Find Documents (find, findOne)**

Retrieve all documents in a collection:

db.users.find();

Find a single document where name is "Alice":

db.users.findOne({ name: "Alice" });

Find users older than 30:

db.users.find({ age: { $gt: 30 } });

Find users from "Los Angeles":

db.users.find({ city: "Los Angeles" });

Find users with age between 25 and 40:

db.users.find({ age: { $gte: 25, $lte: 40 } });

Find users whose names start with "A":

db.users.find({ name: /^A/ });

Find users with either age 25 or 30:

db.users.find({ age: { $in: [25, 30] } });

Find users with an embedded document matching criteria:

db.employees.find({ "address.city": "Boston" });

Find users who have the field "city":

db.users.find({ city: { $exists: true } });

Find users whose name is not "Charlie":

db.users.find({ name: { $ne: "Charlie" } });

**Update Documents (updateOne, updateMany, replaceOne)**

Update a user's age where name is "Alice":

db.users.updateOne({ name: "Alice" }, { $set: { age: 26 } });

Increase the price of all products by 10%:

db.products.updateMany({}, { $mul: { price: 1.1 } });

Add a new field "status" to all documents:

db.users.updateMany({}, { $set: { status: "active" } });

Remove the field "city" from all documents:

db.users.updateMany({}, { $unset: { city: "" } });

Rename the field "age" to "years\_old":

db.users.updateMany({}, { $rename: { age: "years\_old" } });

Increment a user's login count by 1:

db.users.updateOne({ name: "Alice" }, { $inc: { login\_count: 1 } });

Update or insert (upsert) a document if not found:

db.users.updateOne({ name: "David" }, { $set: { age: 40 } }, { upsert: true });

Replace an entire document:

db.users.replaceOne({ name: "Charlie" }, { name: "Charlie", age: 40, city: "Miami" });

Add a new item to an array field:

db.users.updateOne({ name: "Alice" }, { $push: { hobbies: "reading" } });

Remove an item from an array field:

db.users.updateOne({ name: "Alice" }, { $pull: { hobbies: "reading" } });

**Delete Documents (deleteOne, deleteMany)**

Delete a user by name:

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

Delete all users older than 35:

db.users.deleteMany({ age: { $gt: 35 } });

Delete all users from "Los Angeles":

db.users.deleteMany({ city: "Los Angeles" });

Delete all documents in a collection:

db.users.deleteMany({});

Delete a document where a field exists:

db.users.deleteMany({ city: { $exists: true } });

**Sorting and Limiting Results**

Find users and sort by age in ascending order:

db.users.find().sort({ age: 1 });

Find users and sort by name in descending order:

db.users.find().sort({ name: -1 });

Limit results to 5 users:

db.users.find().limit(5);

Skip the first 2 results and return the next 5:

db.users.find().skip(2).limit(5);

**Aggregation Operations**

Count users older than 30:

db.users.countDocuments({ age: { $gt: 30 } });

Group users by city and count them:

db.users.aggregate([

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

]);

Calculate the average age of users:

db.users.aggregate([

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

]);

Find the highest priced product:

db.products.aggregate([

{ $sort: { price: -1 } },

{ $limit: 1 }

]);

**Working with Arrays**

Find users who have "reading" as a hobby:

db.users.find({ hobbies: "reading" });

Find users with at least two hobbies:

db.users.find({ hobbies: { $size: 2 } });

Add multiple values to an array:

db.users.updateOne({ name: "Alice" }, { $addToSet: { hobbies: { $each: ["swimming", "dancing"] } } });

**Indexing and Performance**

Create an index on the "name" field:

db.users.createIndex({ name: 1 });

Create a unique index on the "email" field:

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

**Transactions**

Start a transaction (for multi-document operations):

const session = db.getMongo().startSession();

session.startTransaction();

try {

db.users.updateOne({ name: "Alice" }, { $set: { age: 27 } }, { session });

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

session.commitTransaction();

} catch (error) {

session.abortTransaction();

} finally {

session.endSession();

}