**Create (Insert) Queries**

Insert one document into a collection:

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

Insert multiple documents into a collection:

db.users.insertMany([{ name: "Bob", age: 30 }, { name: "Charlie", age: 35 }]);

Insert a document with an array field:

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

Insert a document with an embedded document:

db.orders.insertOne({ orderId: 1, customer: { name: "David", email: "david@example.com" } });

Insert a document with a unique \_id:

db.customers.insertOne({ \_id: 1001, name: "Emma", country: "USA" });

Insert a document with a timestamp:

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

Insert a document with a nested array:

db.students.insertOne({ name: "John", scores: [95, 87, 92] });

**Read (Find) Queries**

Find all documents in a collection:

db.users.find();

Find a document by a specific field:

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

Find a document using \_id:

db.customers.find({ \_id: 1001 });

Find documents with an age greater than 30:

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

Find documents with an age between 25 and 35:

db.users.find({ age: { $gte: 25, $lte: 35 } });

Find documents where a field is in a list of values:

db.users.find({ city: { $in: ["New York", "Los Angeles"] } });

Find documents where a field is not in a list of values:

db.users.find({ city: { $nin: ["Chicago", "Miami"] } });

Find documents where a field exists:

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

Find documents containing a specific value in an array:

db.products.find({ tags: "electronics" });

Find documents with an embedded document match:

db.orders.find({ "customer.name": "David" });

Find documents sorted by age in descending order:

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

Find documents with a limit of 5:

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

Find documents and exclude a specific field:

db.users.find({}, { password: 0 });

**Update Queries**

Update a single document:

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

Update multiple documents:

db.users.updateMany({ city: "New York" }, { $set: { country: "USA" } });

Increase a numeric value:

db.users.updateOne({ name: "Bob" }, { $inc: { age: 1 } });

Rename a field in a document:

db.users.updateMany({}, { $rename: { "oldField": "newField" } });

Add a new field to a document:

db.users.updateOne({ name: "Charlie" }, { $set: { status: "active" } });

Remove a field from a document:

db.users.updateOne({ name: "Emma" }, { $unset: { age: "" } });

Update an array field by adding a new element:

db.students.updateOne({ name: "John" }, { $push: { scores: 98 } });

Update an array field by removing an element:

db.students.updateOne({ name: "John" }, { $pull: { scores: 95 } });

Update multiple array elements at once:

db.students.updateMany({}, { $addToSet: { courses: "Math" } });

Update a nested document:

db.orders.updateOne({ orderId: 1 }, { $set: { "customer.email": "newemail@example.com" } });

Replace an entire document:

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

**Delete Queries**

Delete a single document:

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

Delete multiple documents:

db.users.deleteMany({ city: "New York" });

Delete all documents in a collection:

db.users.deleteMany({});

Delete documents where a field exists:

db.users.deleteMany({ phone: { $exists: false } });

Delete documents with a specific value in an array:

db.products.deleteMany({ tags: "electronics" });

Delete documents where a numeric field is greater than a value:

db.orders.deleteMany({ total: { $gt: 1000 } });

**Advanced Queries**

Find documents with a regex pattern:

db.users.find({ name: { $regex: "^A", $options: "i" } });

Find documents using a $or condition:

db.users.find({ $or: [{ city: "New York" }, { age: { $lt: 30 } }] });

Find documents using an $and condition:

db.users.find({ $and: [{ city: "New York" }, { age: { $gt: 25 } }] });

Find documents with a $not condition:

db.users.find({ age: { $not: { $gt: 30 } } });

Use aggregation to group by city:

db.users.aggregate([{ $group: { \_id: "$city", count: { $sum: 1 } } }]);

Count documents in a collection:

db.users.countDocuments();

Count documents with a condition:

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

Distinct values of a field:

db.users.distinct("city");

Create an index on a field:

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

Find documents using a text search:

db.articles.find({ $text: { $search: "MongoDB" } });

Drop a collection:

db.users.drop();

Drop a database:

db.dropDatabase();

Backup and restore data:

mongodump --db=mydb --out=/backup/

mongorestore --db=mydb /backup/mydb