**9.MongoDB Queries:**

**Design and Develop MongoDB Queries using CRUD operations. (Use CRUD operations,**

**SAVE method, logical operators etc.).**

**Create**

// Insert a single document

db.users.insertOne({

name: "John Doe",

age: 28,

email: "john@example.com",

address: { city: "New York", zipcode: 10001 }

});

// Insert multiple documents

db.users.insertMany([

{

name: "Jane Doe",

age: 24,

email: "jane@example.com",

address: { city: "Los Angeles", zipcode: 90001 }

},

{

name: "Sam Smith",

age: 30,

email: "sam@example.com",

address: { city: "Chicago", zipcode: 60601 }

}

]);

**read**

// Retrieve all users

db.users.find();

// Retrieve a single user based on name

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

// Retrieve users older than 25

db.users.find({ age: { $gt: 25 } });

// Using logical operators (OR): Retrieve users from either New York or Los Angeles

db.users.find({

$or: [

{ "address.city": "New York" },

{ "address.city": "Los Angeles" }

]

});

**Update**

// Update a single user's age

db.users.updateOne({ name: "John Doe" }, { $set: { age: 29 } });

// Update multiple users: Increase age by 1 for users older than 25

db.users.updateMany({ age: { $gt: 25 } }, { $inc: { age: 1 } });

// Using save() to update or insert (upsert)

db.users.save({

\_id: ObjectId("60d5c479b9a6f2d1d4c0e6a2"), // This will update the document if the \_id exists

name: "John Doe",

age: 30,

email: "john@example.com",

address: { city: "New York", zipcode: 10001 }

});

**Delete**

// Delete a single user

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

// Delete multiple users older than 30

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

**logical**

// Find users between 25 and 30 years old

db.users.find({

$and: [

{ age: { $gt: 25 } },

{ age: { $lt: 30 } }

]

});

// Find users not living in New York

db.users.find({ "address.city": { $not: { $eq: "New York" } } });

**theory**

**1. Create:**

Purpose: Insert documents (records) into the users collection.

Code Explanation:

Single Document Insertion: db.users.insertOne({...}) inserts one document with fields such as name, age, email, and a nested address object containing city and zipcode.

Multiple Documents Insertion: db.users.insertMany([...]) allows inserting multiple documents in a single command, saving time and reducing the number of separate insert operations.

**2. Read:**

Purpose: Retrieve documents from the users collection using different queries.

Code Explanation:

Retrieve All: db.users.find() retrieves all documents in the users collection.

Retrieve by Criteria:

db.users.findOne({ name: "John Doe" }) finds a single document where the name field is "John Doe".

db.users.find({ age: { $gt: 25 } }) retrieves users older than 25 by using the $gt (greater than) operator.

Logical OR Operator: db.users.find({ $or: [...] }) uses $or to find users in either "New York" or "Los Angeles". This is helpful for flexible queries that match one of several conditions.

**3. Update:**

Purpose: Modify existing documents based on certain conditions.

Code Explanation:

Single Document Update: db.users.updateOne({...}) updates one document matching the given criteria (in this case, updating John Doe’s age to 29).

Multiple Document Update: db.users.updateMany({...}) applies the update to multiple documents that meet the condition. The $inc operator is used to increment the age field by 1 for users older than 25.

Upsert Operation: db.users.save({...}) is a method to insert or update a document based on the presence of an \_id. If a document with the specified \_id exists, it updates that document; otherwise, it inserts it as a new document.

**4. Delete:**

Purpose: Remove documents from the users collection.

Code Explanation:

Single Document Deletion: db.users.deleteOne({...}) deletes one document matching the given criteria (e.g., a user with the name "Jane Doe").

Multiple Document Deletion: db.users.deleteMany({...}) removes all documents that meet a specific condition (e.g., deleting users older than 30).

**5. Logical Queries:**

Purpose: Query documents based on complex conditions combining multiple criteria.

Code Explanation:

AND Query: db.users.find({ $and: [...] }) retrieves users between 25 and 30 years of age. The $and operator is used to match both conditions simultaneously.

NOT Query: db.users.find({ "address.city": { $not: { $eq: "New York" } } }) finds users whose city is not "New York". $not negates the specified condition, returning documents that do not match it.