**MongoDB Lab Manual Exercises**

**Exercise 1: Basic CRUD Operations**

**Objective:** Learn how to perform basic CRUD operations (Create, Read, Update, Delete) in MongoDB.

**Task:**

1. **Insert** a document into a collection called students. The document should have fields: name, age, and course.
2. **Retrieve** all documents from the students collection.
3. **Update** a student's age by increasing it by 1 for the student with name John Doe.
4. **Delete** the document of the student with name Jane Doe.

**Solution:**

1. **Insert:**

db.students.insertOne({

name: "John Doe",

age: 22,

course: "Computer Science"

});

db.students.insertOne({

name: "Jane Doe",

age: 20,

course: "Mathematics"

});

1. **Retrieve:**

db.students.find();

1. **Update:**

db.students.updateOne(

{ name: "John Doe" },

{ $inc: { age: 1 } }

);

1. **Delete:**

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

**Exercise 2: Query Filtering and Projection**

**Objective:** Learn to filter and project fields from documents.

**Task:**

1. **Retrieve** all students who are older than 21 years.
2. **Retrieve** only the name and course of students older than 21 years.

**Solution:**

1. **Filter:**

db.students.find({ age: { $gt: 21 } });

1. **Filter and Project:**

db.students.find(

{ age: { $gt: 21 } },

{ name: 1, course: 1, \_id: 0 }

);

**Exercise 3: Using Aggregation**

**Objective:** Learn to use MongoDB's aggregation framework.

**Task:**

1. **Count** the total number of students in the students collection.
2. **Group** students by course and get the total number of students per course.

**Solution:**

1. **Count:**

db.students.countDocuments();

1. **Group by Course:**

db.students.aggregate([

{ $group: { \_id: "$course", totalStudents: { $sum: 1 } } }

]);

**Exercise 4: Indexing and Performance**

**Objective:** Learn to create and use indexes for optimizing queries.

**Task:**

1. **Create an index** on the name field of the students collection.
2. **Use** the index to retrieve a student by their name.

**Solution:**

1. **Create Index:**

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

1. **Use the Index to Query:**

db.students.find({ name: "John Doe" });

**Exercise 5: Working with Embedded Documents**

**Objective:** Learn to work with embedded documents in MongoDB.

**Task:**

1. **Create a document** in the students collection with an embedded address field. The address should contain street, city, and zipcode fields.
2. **Retrieve** the students who live in the city "New York".

**Solution:**

1. **Create Document with Embedded Address:**

db.students.insertOne({

name: "John Doe",

age: 22,

course: "Computer Science",

address: {

street: "123 Elm St",

city: "New York",

zipcode: "10001"

}

});

1. **Retrieve Students from New York:**

db.students.find({ "address.city": "New York" });

**Exercise 6: Use of $and, $or Conditions**

**Objective:** Learn to combine multiple conditions using $and and $or.

**Task:**

1. **Retrieve** students who are either older than 21 years or enrolled in "Computer Science".
2. **Retrieve** students who are older than 21 years **and** enrolled in "Computer Science".

**Solution:**

1. **Using $or:**

db.students.find({

$or: [

{ age: { $gt: 21 } },

{ course: "Computer Science" }

]

});

1. **Using $and:**

db.students.find({

$and: [

{ age: { $gt: 21 } },

{ course: "Computer Science" }

]

});