### Problem Statement 1 (CRUD)

We will create a MongoDB collection called `Student`, insert some sample data, and perform various CRUD operations (Create, Read, Update, Delete) on the collection.

---

### Step 1: Create the Collection `Student`

To create the collection and insert documents, first ensure your MongoDB server is running and open the MongoDB shell (`mongosh`).

Switch to the database where you want to create the collection:

```javascript

use school

```

Now, let’s create the `Student` collection and insert some sample documents.

```javascript

db.Student.insertMany([

{ Roll\_No: "A01", Name: "Alice", Class: "SE", Marks: 85, Address: "123 Street, City", Enrolled\_Courses: ["DBMS", "TOC", "Maths"] },

{ Roll\_No: "A02", Name: "Bob", Class: "TE", Marks: 75, Address: "456 Street, City", Enrolled\_Courses: ["DBMS", "Algorithms", "Maths"] },

{ Roll\_No: "A03", Name: "Charlie", Class: "SE", Marks: 90, Address: "789 Street, City", Enrolled\_Courses: ["TOC", "Maths"] },

{ Roll\_No: "A04", Name: "David", Class: "BE", Marks: 60, Address: "321 Street, City", Enrolled\_Courses: ["DBMS", "AI"] },

{ Roll\_No: "A05", Name: "Eve", Class: "TE", Marks: 45, Address: "654 Street, City", Enrolled\_Courses: ["Maths", "DBMS", "TOC"] },

{ Roll\_No: "A06", Name: "Frank", Class: "SE", Marks: 55, Address: "987 Street, City", Enrolled\_Courses: ["TOC", "Algorithms"] },

{ Roll\_No: "A07", Name: "Grace", Class: "TE", Marks: 80, Address: "111 Street, City", Enrolled\_Courses: ["DBMS", "AI"] },

{ Roll\_No: "A08", Name: "Hannah", Class: "SE", Marks: 35, Address: "222 Street, City", Enrolled\_Courses: ["TOC", "DBMS"] },

{ Roll\_No: "A09", Name: "Ivy", Class: "BE", Marks: 25, Address: "333 Street, City", Enrolled\_Courses: ["Maths", "DBMS"] },

{ Roll\_No: "A10", Name: "Jack", Class: "TE", Marks: 95, Address: "444 Street, City", Enrolled\_Courses: ["DBMS", "AI"] }

]);

```

---

### Step 2: Queries for CRUD Operations

#### Query 1: List the names of students who have enrolled in the course "DBMS" and "TOC"

```javascript

db.Student.find(

{ Enrolled\_Courses: { $all: ["DBMS", "TOC"] } },

{ Name: 1, \_id: 0 }

).pretty();

```

- \*\*Explanation\*\*: The `$all` operator ensures that both "DBMS" and "TOC" courses are present in the `Enrolled\_Courses` array.

#### Query 2: List the Roll numbers and class of students who have marks greater than 50 or class as "TE"

```javascript

db.Student.find(

{ $or: [ { Marks: { $gt: 50 } }, { Class: "TE" } ] },

{ Roll\_No: 1, Class: 1, \_id: 0 }

).pretty();

```

- \*\*Explanation\*\*: The `$or` operator is used to fetch students who either have marks greater than 50 or belong to class "TE".

#### Query 3: Update the entire record of roll\_no "A10"

```javascript

db.Student.updateOne(

{ Roll\_No: "A10" },

{

$set: {

Name: "John",

Class: "BE",

Marks: 80,

Address: "555 Street, City",

Enrolled\_Courses: ["AI", "ML"]

}

}

);

```

- \*\*Explanation\*\*: The `$set` operator updates the document with the new values. We update the entire record for `Roll\_No: "A10"`.

#### Query 4: Display the names of students having 3rd and 4th highest marks

To get the 3rd and 4th highest marks, we can first sort the students by marks in descending order, and then use `skip` and `limit` to fetch the appropriate records.

```javascript

db.Student.find()

.sort({ Marks: -1 })

.skip(2) // Skip the top 2 highest marks

.limit(2) // Get the next 2 records (3rd and 4th highest marks)

.project({ Name: 1, Marks: 1, \_id: 0 })

.pretty();

```

- \*\*Explanation\*\*: Sorting by `Marks: -1` sorts students in descending order of marks. Then `skip(2)` skips the top 2 records (highest and 2nd highest), and `limit(2)` fetches the 3rd and 4th highest records.

#### Query 5: Delete the records of students having marks less than 20

```javascript

db.Student.deleteMany({ Marks: { $lt: 20 } });

```

- \*\*Explanation\*\*: The `deleteMany` method deletes all documents where `Marks` is less than 20.

#### Query 6: Delete only the first record from the collection

```javascript

db.Student.deleteOne({}); // This will delete the first document as it matches the query condition (empty query matches the first document)

```

- \*\*Explanation\*\*: Since no conditions are specified, `deleteOne` will delete the first document it encounters in the collection.

---

### Summary of Queries

1. \*\*List students enrolled in DBMS and TOC\*\*: `find()` with `$all`.

2. \*\*List students with marks > 50 or class TE\*\*: `find()` with `$or`.

3. \*\*Update student record for roll\_no A10\*\*: `updateOne()` with `$set`.

4. \*\*Display 3rd and 4th highest marks\*\*: `find()` with `sort()`, `skip()`, and `limit()`.

5. \*\*Delete students with marks < 20\*\*: `deleteMany()`.

6. \*\*Delete first document\*\*: `deleteOne()` with empty filter.

---

### Running the Queries

1. \*\*Open MongoDB Shell\*\*: Run `mongosh` in your terminal.

2. \*\*Select Database\*\*: Switch to the `school` database by running:

```javascript

use school

```

3. \*\*Execute Queries\*\*: Run each of the queries listed above to perform the respective operations.

By following these steps, you will successfully create the collection, insert data, and perform the required CRUD operations in MongoDB.