Below are the MongoDB CRUD (Create, Read, Update, Delete) operations using a `student` collection as an example. I'll cover each operation in detail with appropriate MongoDB queries, including the `save` method and logical operators.

1. Create Operation (Insert Document)

To insert a new student record into the `students` collection, you can use the `insertOne()` or `insertMany()` method. The `save()` method can also be used, though it is deprecated in favor of `insertOne()` and `insertMany()`.

Query using `insertOne()`:

javascript

```
db.students.insertOne({
  name: "John Doe",
  age: 21,
  major: "Computer Science",
  gpa: 3.8,
  enrolled: true
});
```

Query using `insertMany()`:

javascript

Copy code

```
db.students.insertMany([
  {
    name: "Jane Smith",
    age: 22,
    major: "Mathematics",
    gpa: 3.6,
    enrolled: true
  },
  {
    name: "Sam Brown",
    age: 23,
    major: "Physics",
    gpa: 3.9,
    enrolled: false
```

```
  }
]);
```

Query using `save()` (deprecated but still works):

javascript
Copy code

```
db.students.save({
  name: "Emily White",
  age: 20,
  major: "Biology",
  gpa: 3.7,
  enrolled: true
});
```

2. Read Operation (Query Documents)

You can query the `students` collection using various criteria. The `find()` method is commonly used for this purpose.

Query to find all students:

javascript
Copy code

```
db.students.find({});
```

Query to find students with specific criteria (e.g., age > 21):

javascript
Copy code

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

Query to find students by major:

javascript
Copy code

```
db.students.find({ major: "Computer Science" });
```

Query using logical operators (e.g., `$or` to find students with GPA > 3.8 or enrolled):

javascript
Copy code

```
db.students.find({
  $or: [
    { gpa: { $gt: 3.8 } },
    { enrolled: true }
  ]
});
```

Query to find a specific student (e.g., by name):

javascript
Copy code

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

Query to find students with a GPA between 3.5 and 4.0:

javascript
Copy code

```
db.students.find({
  gpa: { $gte: 3.5, $lte: 4.0 }
});
```

3. Update Operation (Modify Documents)

MongoDB provides methods like `updateOne()`, `updateMany()`, and `replaceOne()` to modify documents.

Update a single student's record (e.g., update GPA):

javascript
Copy code

```
db.students.updateOne(
  { name: "John Doe" },  // Query condition
  { $set: { gpa: 3.9 } } // Update operation
);
```

Update multiple students' records (e.g., change major for all Computer Science students):

javascript
Copy code

```
db.students.updateMany(
  { major: "Computer Science" },  // Query condition
```

```
  { $set: { major: "Software Engineering" } } // Update operation
);
```

Update or insert a student using `upsert`:

javascript
Copy code

```
db.students.updateOne(
  { name: "Alice Green" },  // Query condition
  { $set: { age: 24, major: "Chemistry", gpa: 3.4 } },  // Update operation
  { upsert: true }  // If no matching document, insert new one
);
```

Replace a student's document (e.g., completely replace John's record):

javascript
Copy code

```
db.students.replaceOne(
  { name: "John Doe" },  // Query condition
  { name: "John Doe", age: 22, major: "Mathematics", gpa: 3.7, enrolled: true }  // New document
);
```

4. Delete Operation (Remove Documents)

To delete documents, use methods like `deleteOne()`, `deleteMany()`, or `remove()`.

Delete a single student:

javascript
Copy code

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

Delete multiple students based on a condition (e.g., delete all unenrolled students):

javascript
Copy code

```
db.students.deleteMany({ enrolled: false });
```

Delete all students:

javascript
Copy code
```
db.students.deleteMany({});
```

5. Logical Operators

Logical operators like `$and`, `$or`, `$not`, `$nor`, etc., can be used to combine multiple conditions in a query.

Example with `$and` (students who are enrolled and have a GPA above 3.5):

javascript
Copy code
```
db.students.find({
  $and: [
    { enrolled: true },
    { gpa: { $gt: 3.5 } }
  ]
});
```

Example with `$or` (students who are either enrolled or have a GPA greater than 3.9):

javascript
Copy code
```
db.students.find({
  $or: [
    { enrolled: true },
    { gpa: { $gt: 3.9 } }
  ]
});
```

Example with `$not` (students who do not have a GPA greater than 3.5):

javascript
Copy code
```
db.students.find({
  gpa: { $not: { $gt: 3.5 } }
});
```

Example with `$in` (students whose major is either "Computer Science" or "Physics"):

javascript
Copy code

```
db.students.find({
  major: { $in: ["Computer Science", "Physics"] }
});
```

Summary of MongoDB CRUD Operations

- **Create**: `insertOne()`, `insertMany()`, `save()`
- **Read**: `find()`, `findOne()`
- **Update**: `updateOne()`, `updateMany()`, `replaceOne()`
- **Delete**: `deleteOne()`, `deleteMany()`
- **Logical Operators**: `$and`, `$or`, `$not`, `$in`, `$gte`, `$lte`, `$gt`, `$lt`

This set of queries should help you get started with performing CRUD operations on a MongoDB `students` collection.