MongoDB Operator Exercise

Author: Đặng Kim Thi

Instructions

Perform the following tasks using MongoDB operators:

1. Create a Database and Collection

- **Database Name**: `Student`
- **Collection Name**: `Stud_mark`

2. Insert the Following Documents

```
[
    {
        "name": "Adam",
        "gender": "M",
        "subjects": ["Java", "C", "Python"],
        "marks": [89, 78, 90],
        "average": 85.6
    },
    {
        "name": "Franklin",
        "gender": "M",
        "subjects": ["C", "VB", "Python"],
        "marks": [78, 85, 89],
        "average": 84
    },
    {
        "name": "Michael",
        "gender": "M",
        "subjects": ["Java", "PHP"],
        "marks": [88, 89],
        "average": 88.5
    },
    {
        "name": "Amelia",
        "gender": "F",
        "subjects": ["Ruby", "C++"],
        "marks": [86, 87],
        "average": 86.5
    }
]
```

3. Perform the Following Queries

3.1 Equality and Range Queries

1. Find only the documents where the `average` value is equal to 84.
2. Find only the documents where the `average` value is greater than 85.
3. View only the documents where the `average` is greater than or equal to 87 and less than or equal to 90.

3.2 Array Queries

4. Display only the documents where the `subjects` array contains either `Java` or `C++`.
5. View all the documents where the `subjects` array has the value `Java`.
6. Display only the documents where the first element in the `marks` array is less than 80.
7. Display the details of the student named `Adam` where the `marks` array has only the first element and the second element.

3.3 Updating Fields

8. Add a new date field `Date_of_exam` which shows the current date only for the student named `Amelia`.
9. Increase the `average` value by 2 for the student named `Franklin`.
10. Rename the field `Date_of_exam` to `Examination_date`.

MongoDB Commands for Reference

Insert Data

```
db.Stud_mark.insertMany([
    { name: "Adam", gender: "M", subjects: ["Java", "C", "Python"], marks: [89, 78, 90], average: 85.6 },
    { name: "Franklin", gender: "M", subjects: ["C", "VB", "Python"], marks: [78, 85, 89], average: 84 },
    { name: "Michael", gender: "M", subjects: ["Java", "PHP"], marks: [88, 89], average: 88.5 },
    { name: "Amelia", gender: "F", subjects: ["Ruby", "C++"], marks: [86, 87], average: 86.5 }
]);
```

Queries

1. **Equality Query:**

   ```
   db.Stud_mark.find({ average: 84 });
   ```

2. **Range Query:**

```
db.Stud_mark.find({ average: { $gt: 85 } });
db.Stud_mark.find({ average: { $gte: 87, $lte: 90 } });
```

3. **Array Query:**

```
db.Stud_mark.find({ subjects: { $in: ["Java", "C++"] } });
db.Stud_mark.find({ "subjects.0": "Java" });
db.Stud_mark.find({ "marks.0": { $lt: 80 } });
```

4. **Specific Array Elements:**

```
db.Stud_mark.find({ name: "Adam" }, { name: 1, marks: { $slice: 2 }
});
```

Updates

1. **Add Date Field:**

```
db.Stud_mark.updateOne(
    { name: "Amelia" },
    { $set: { Date_of_exam: new Date() } }
);
```

2. **Increase Average:**

```
db.Stud_mark.updateOne(
    { name: "Franklin" },
    { $inc: { average: 2 } }
);
```

3. **Rename Field:**

```
db.Stud_mark.updateMany(
    {},
    { $rename: { "Date_of_exam": "Examination_date" } }
);
```

Additional Exercises

Exercise 1: Employee Records

1. Create a Database and Collection

- **Database Name**: `Company`
- **Collection Name**: `Employee`

2. Insert the Following Documents

```
[
    {
        "name": "Alice",
        "department": "HR",
        "skills": ["Communication", "Recruitment"],
        "salary": 50000,
        "experience": 5
    },
    {
        "name": "Bob",
        "department": "IT",
        "skills": ["Java", "Python"],
        "salary": 70000,
        "experience": 8
    },
    {
        "name": "Charlie",
        "department": "Finance",
        "skills": ["Accounting", "Excel"],
        "salary": 60000,
        "experience": 6
    }
]
```

3. Perform the Following Queries

1. Find employees with a salary greater than 60000.
2. Display employees with `Python` as one of their skills.
3. Update the experience of `Alice` to 6 years.
4. Rename the `salary` field to `annual_salary`.

Exercise 2: Library Management

1. Create a Database and Collection

- **Database Name**: `Library`
- **Collection Name**: `Books`

2. Insert the Following Documents

```
[
    {
        "title": "To Kill a Mockingbird",
        "author": "Harper Lee",
        "genres": ["Fiction", "Classic"],
        "copies": 5,
        "borrowed": 3
    },
    {
        "title": "1984",
        "author": "George Orwell",
        "genres": ["Fiction", "Dystopian"],
        "copies": 8,
        "borrowed": 6
    },
    {
        "title": "The Great Gatsby",
        "author": "F. Scott Fitzgerald",
        "genres": ["Fiction", "Classic"],
        "copies": 3,
        "borrowed": 1
    }
]
```

3. Perform the Following Queries

1. Find books with `borrowed` count less than 5.
2. Display books of the genre `Classic`.
3. Add a new field `available_copies` for all books (calculated as `copies - borrowed`).
4. Update the `author` field of `1984` to `Eric Arthur Blair`.

Exercise 3: Online Store

1. Create a Database and Collection

- **Database Name**: `ECommerce`
- **Collection Name**: `Products`

2. Insert the Following Documents

```
[
    {
        "product": "Laptop",
        "brand": "Dell",
        "price": 1200,
        "stock": 15,
        "ratings": [5, 4, 4, 5, 5]
    },
    {
        "product": "Smartphone",
        "brand": "Samsung",
        "price": 800,
        "stock": 30,
        "ratings": [4, 4, 5, 3, 4]
    },
    {
        "product": "Headphones",
        "brand": "Sony",
        "price": 150,
        "stock": 50,
        "ratings": [5, 5, 4, 5, 4]
    }
]
```

3. Perform the Following Queries

1. Find products priced above 500.
2. Display products with an average rating greater than 4.5.
3. Reduce the stock of `Laptop` by 2 units.
4. Add a new field `on_sale` and set it to `true` for products with price less than 200.