

Task 4: Advanced Querying with MongoDB

Objective:

Understand and practice advanced querying techniques in MongoDB, including using comparison, logical, and element operators.

Prerequisites:

- Basic understanding of JavaScript and MongoDB.
- Node.js installation.
- MongoDB installed and running.
- A MongoDB collection with sample movie data.

Concepts:

1. Query Operators:

Comparison Operators:

- `$gt` (greater than): Matches values that are greater than the specified value.
- `$lt` (less than): Matches values that are less than the specified value.
- `$eq` (equals): Matches values that are equal to the specified value.
- `$ne` (not equals): Matches values that are not equal to the specified value.

Example:

JavaScript:

```
const comparisonQuery = await collection.find({ popularity: { $gt: 100 } }).toArray();
console.log('Movies with popularity > 100:', comparisonQuery);
```

MongoDB Compass:

- Open MongoDB Compass.
- Connect to your MongoDB instance.
- Select your database and collection.
- Click on the `Filter` tab.
- Enter the following query in the filter box:

```
{ "popularity": { "$gt": 100 } }
```

- Click on the `Find` button to execute the query.

2. Logical Operators:

- `$and`: Joins query clauses with a logical AND returns all documents that match the conditions of both clauses.
- `$or`: Joins query clauses with a logical OR returns all documents that match the conditions of either clause.

- **\$not** : Inverts the effect of a query expression and returns documents that do not match the query expression.
- **\$nor** : Joins query clauses with a logical NOR returns all documents that fail to match both clauses.

Example:

JavaScript:

```
const logicalQuery = await collection.find({
  $or: [
    { genre: 'Action' },
    { vote_average: { $gt: 7 } }
  ]
}).toArray();
console.log('Action movies or movies with vote average > 7:', logicalQuery);
```

MongoDB Compass:

- Follow the same steps to navigate to your collection in MongoDB Compass.
- Enter the following query in the filter box:

```
{
  "$or": [
    { "genre": "Action" },
    { "vote_average": { "$gt": 7 } }
  ]
}
```

- Click on the Find button to execute the query.

3. Element Operators:

- **\$exists** : Matches documents that have the specified field.
- **\$type** : Matches documents where the field is of the specified type.

Example:

JavaScript:

```
const elementQuery = await collection.find({ video: { $exists:
true } }).toArray();
console.log('Movies with video field:', elementQuery);
```

MongoDB Compass:

- Follow the same steps to navigate to your collection in MongoDB Compass.
- Enter the following query in the filter box:

```
{ "video": { "$exists": true } }
```

- Click on the Find button to execute the query.

4. Projection in Queries:

- Projection is used to specify or restrict the fields returned in the query results.
- By including specific fields in the projection, you can limit the data returned to only what is necessary.

Example:

JavaScript:

```
const projectionQuery = await collection.find({}, { projection: { title: 1,
release_date: 1 } }).toArray();
console.log('Projected Movies:', projectionQuery);
```

MongoDB Compass:

- Follow the same steps to navigate to your collection in MongoDB Compass.
- Click on the Project tab.
- Enter the following projection:

```
{
  "title": 1,
  "release_date": 1
}
```

- Click on the Find button to execute the query.

Instructions:

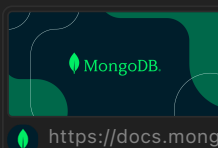
Perform the following queries:

1. Find movies that are either in the 'Science Fiction' genre or have a vote average less than 5, and also have a popularity less than 50.
2. Find movies that have an 'adult' field set to false, and either have a release date before the year 2024 or a vote count greater than 50.
3. Find movies and project only the title, genre, and vote_average fields, where the genre includes 'Horror' and the popularity is greater than 75.
4. Combine all the operators to find movies that are for adults, have a vote average greater than 6, include 'Thriller' in the genre, and project only the title, release_date, and vote_count fields.

Resources:

- **Documentation:**



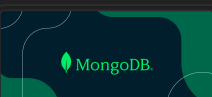
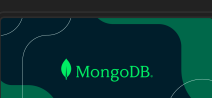
-



Query and Projection Operators

For details on a specific operator, including syntax and examples, — click on the link...

<https://docs.mongodb.com/manual/reference/operator/query/>

- 
Comparison Query Operators
 For details on a specific operator, including syntax and examples, — click on the link...
<https://docs.mongodb.com/manual/reference/operator/query-comparison/>
- 
Logical Query Operators
 Logical operators return data based on expressions that evaluate to — true or false.
<https://docs.mongodb.com/manual/reference/operator/query-logical/>
- 
Element Query Operators
 Element operators return data based on field existence or data types.
<https://docs.mongodb.com/manual/reference/operator/query-element/>
- 
db.collection.find()
 This page documents a .leafygreen-ui-rp2r6i{font-family:'Euclid Circular A','Helvetica...
<https://docs.mongodb.com/manual/reference/method/db.collection.find/#projection>

• Videos:

- 
MongoDB Courses and Trainings | MongoDB University
 Discover our MongoDB Database Management courses and begin improving your CV with Mong...
<https://university.mongodb.com/courses/M101JS/about>
- 

The video thumbnail features a dark green background with stylized white leaves. In the center, the text 'MongoDB Tutorial' is displayed in large, bold, white letters. Below it, 'Complex Queries & Operators' is written in a smaller white font. A green circle with the number '10' is visible in the bottom left corner. At the bottom of the thumbnail, there is a red play button icon and the text 'Net Ninja 7:38min 101,104 Views 1,745 Likes'.

GitHub Instructions

Open in Visual Studio Code:

After clicking on the "Open in Visual Studio Code" button from the GitHub Classroom confirmation page, Visual Studio Code (VSCode) will open the repository directly.

If prompted, select "Open" or "Allow" to open the repository in VSCode.

Complete the Task:

In VSCode, open the `index.js` file in the root directory of your repository and write your solution.

Ensure the `package.json` file is present and contains all necessary dependencies. If you need to install additional packages, use:

```
npm i
```

Run and Test Your Code:

Run your code to ensure it works correctly. Use the following command:

```
node index.js
```

Commit Your Changes:

Commit your changes with a meaningful message:

```
git commit -m "Completed task 4"
```

Push Your Changes to Your Forked Repository:

Push your changes to your forked repository:

```
git push origin main
```

Create a Pull Request:

Go to your forked repository on GitHub.

Click on the "Pull Requests" tab.

Click the "New Pull Request" button.

Ensure the base repository is the original template repository and the base branch is `main`.

Ensure the head repository is your forked repository and the compare branch is `main`.

Click "Create Pull Request".

Add a title and description for your pull request and submit it.

Summary of Commands

```
# Fork the repository on GitHub

# Clone the forked repository
git clone https://github.com/your-github-username/repository-name.git
cd repository-name

# Complete the task by editing index.js

# Run your code
node index.js

# Add, commit, and push your changes
git commit -m "Completed task 4"
git push origin main

# Create a pull request on GitHub
```