

Task 5: Array Queries in MongoDB

Objective:

Understand and practice querying array fields in MongoDB.

Prerequisites:

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

Concepts:

1. Array Queries:

Querying Arrays:

- You can query arrays to match documents where the array field contains specific values.
- Use `$in` to match any of the values specified in an array.

Example:

JavaScript:

```
const arrayQuery = await collection.find({ genre: 'Comedy' }).toArray();
console.log('Comedy Movies:', arrayQuery);
```

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:

```
{ "genre": "Comedy" }
```

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

Using \$elemMatch:

- `$elemMatch` is used to match documents where at least one element in the array matches the specified criteria.
- This operator is useful when you need to specify multiple conditions on array elements.

Example:

JavaScript:

```
const elemMatchQuery = await collection.find({
  genre: { $elemMatch: { $in: ['Action', 'Adventure'] } }
}).toArray();
console.log('Action and Adventure Movies:', elemMatchQuery);
```

MongoDB Compass:

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

```
{ "genre": { "$elemMatch": { "$in": ["Action", "Adventure"] } } }
```

- Click on the Find button to execute the query.

Advanced Array Query:

- Combine array queries with other conditions like \$where to filter based on the array length.

Example:

JavaScript:

```
const advancedArrayQuery = await collection.find({
  $and: [
    { genre: 'Drama' },
    { $where: 'this.genre.length > 2' }
  ]
}).toArray();
console.log('Drama Movies with more than two genres:', advancedArrayQuery);
```

MongoDB Compass:

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

```
{
  "$and": [
    { "genre": "Drama" },
    { "$where": "this.genre.length > 2" }
  ]
}
```

- Click on the Find button to execute the query.

Array and Projection Combined:

- Use array queries along with projection to limit the fields returned in the result set.

Example:

JavaScript:

```
const projectionArrayQuery = await collection.find(
  { genre: 'Sci-Fi' },
  { projection: { title: 1, genre: 1, popularity: 1 } }
).toArray();
console.log('Sci-Fi Movies (projected):', projectionArrayQuery);
```

MongoDB Compass:

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

```
{ "genre": "Sci-Fi" }
```

- Click on the `Project` tab.
- Enter the following projection:

```
{
  "title": 1,
  "genre": 1,
  "popularity": 1
}
```

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

Combining Array Queries with Other Operators:

- Combine array queries with comparison, logical, and projection operators to create complex queries.

Example:

JavaScript:

```
const combinedArrayQuery = await collection.find(
  {
    $and: [
      { genre: 'Horror' },
      { popularity: { $gt: 75 } },
      { vote_average: { $lt: 5 } }
    ]
  },
  { projection: { title: 1, vote_average: 1 } }
).toArray();
console.log('Horror Movies with popularity > 75 and vote average < 5:',
combinedArrayQuery);
```

MongoDB Compass:

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

```
{
  "$and": [
    { "genre": "Horror" },
    { "popularity": { "$gt": 75 } },
    { "vote_average": { "$lt": 5 } }
  ]
}
```

- Click on the **Project** tab.
- Enter the following projection:

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

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

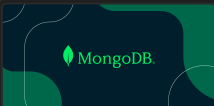

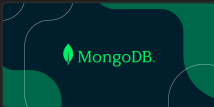



Instructions:

Perform the following queries:

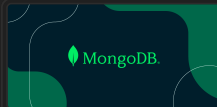
1. Find movies where the genre includes 'Family'.
2. Find movies where the genre includes both 'Fantasy' and 'Animation' using `$elemMatch`.
3. Find movies where the genre includes 'Romance' and the genre array has exactly three elements.
4. Find movies where the genre includes 'Documentary' and project only the title, release_date, and overview fields.
5. Find movies where the genre includes 'Adventure', the popularity is less than 60, and the vote average is greater than 8, and project only the title, genre, and popularity fields.

Resources:

- **Documentation:**

-  **Query an Array**
The `.leafygreen-ui-rp2r6i`{font-family:'Euclid Circular A','Helvetica Neue',Helvetica,Ar...
 <https://docs.mongodb.com/manual/tutorial/query-arrays/>
-  **Query an Array of Embedded Documents**
You can query documents in MongoDB by using the following — methods:
 <https://docs.mongodb.com/manual/tutorial/query-array-of-documents/>
-  **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/>

◦



`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:

◦



◦



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 5"
```

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 5"
git push origin main

# Create a pull request on GitHub
```