

Task 3: Aggregating Movie Data

Objective:

Understand and practice the basics of aggregation in MongoDB, including using the aggregation pipeline to match, group, and sort movie data.

Prerequisites:

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

Concepts:

1. Introduction to Aggregation:

- Aggregation operations process data records and return computed results.
- Aggregation operations group values from multiple documents together and can perform a variety of operations on the grouped data to return a single result.
- MongoDB provides the aggregate method to use the aggregation framework.

2. Creating the Aggregation Pipeline:

Match Stage:

- The match stage filters the documents to pass only the documents that match the specified condition(s) to the next pipeline stage.

Example:

JavaScript:

```
const { MongoClient } = require('mongodb');

async function matchStage() {
  const uri = 'mongodb://localhost:27017'; // Replace with your MongoDB URI
  const client = new MongoClient(uri, { useNewUrlParser: true,
    useUnifiedTopology: true });

  try {
    await client.connect();
    const database = client.db('MoviesDB');
    const collection = database.collection('Movies');

    // Match movies with popularity greater than 50
    const match = await collection.aggregate([
      { $match: { popularity: { $gt: 50 } } }
    ]).toArray();
    console.log('Matched Movies:', match);
  } finally {
    await client.close();
  }
}
```

```

    }
  }

  matchStage().catch(console.dir);

```

MongoDB Compass:

- Open MongoDB Compass.
- Connect to your MongoDB instance.
- Select your database and collection.
- Click on the `Aggregation` tab.
- Add a new stage and select `$match`.
- Enter the following condition:

```
{ "popularity": { "$gt": 50 } }
```

- Click on the `Add Stage` button.

Group Stage:

- The group stage groups input documents by the specified `_id` expression and for each distinct grouping, outputs a document.
- The output documents can also contain computed fields that hold the values of some accumulator expressions.

Example:

JavaScript:

```

const group = await collection.aggregate([
  { $match: { popularity: { $gt: 50 } } },
  { $unwind: "$genre" },
  { $group: { _id: "$genre", averagePopularity: { $avg: "$popularity" } } }
]).toArray();
console.log('Grouped Movies:', group);

```

MongoDB Compass:

- Add a new stage and select `$group`.
- Enter the following condition:

```
{ "_id": "$genre", "averagePopularity": { "$avg": "$popularity" } }
```

- Click on the `Add Stage` button.

Sort Stage:

- The sort stage sorts all input documents and returns them to the pipeline in sorted order.

Example:

JavaScript:

```
const sorted = await collection.aggregate([
  { $match: { popularity: { $gt: 50 } } },
  { $unwind: "$genre" },
  { $group: { _id: "$genre", averagePopularity: { $avg: "$popularity" } } },
  { $sort: { averagePopularity: -1 } }
]).toArray();
console.log('Sorted Movies:', sorted);
```

MongoDB Compass:

- Add a new stage and select `$sort`.
- Enter the following condition:

```
{ "averagePopularity": -1 }
```

- Click on the Add Stage button.



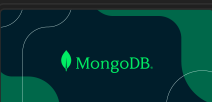
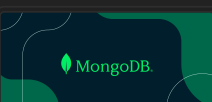
Instructions:

Perform the following queries:

1. Match movies with a popularity greater than 50.
2. Group movies by genre and calculate the average popularity.
3. Sort the results by the average popularity in descending order.
4. Extend the pipeline to also include the count of movies in each genre.

Resources:

- **Documentation:**

-  **Aggregation Operations**
Aggregation operations process multiple documents and return computed — results...
<https://docs.mongodb.com/manual/aggregation/>
-  **\$match (aggregation)**
Filters the documents to pass only the documents that match the — specified condit...
<https://docs.mongodb.com/manual/reference/operator/aggregation/match/>
-  **\$group (aggregation)**
The \$group stage separates documents into groups according to a — "group key". T...
<https://docs.mongodb.com/manual/reference/operator/aggregation/group/>
-  **\$sort (aggregation)**
Sorts all input documents and returns them to the pipeline in sorted — order.
<https://docs.mongodb.com/manual/reference/operator/aggregation/sort/>

- **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/M121/about>
- 
YouTube **Hitesh Choudhary** #mongo
Hitesh Choudhary 19:59min 54,499 Views 1,679 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:

In the VSCode terminal, add your changes to git:

```
git add index.js package.json
```

Commit your changes with a meaningful message:

```
git commit -m "Completed task 3"
```

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

# commit, and push your changes
git commit -m "Completed task 3"
git push origin main

# Create a pull request on GitHub
```