**1. Find the Most Frequent Author in a Books Collection**

Given a collection of books, each book has multiple authors. Write a query to find the author who has written the most books.

**Collection Example (books):**

{

"\_id": 1,

"title": "Book One",

"authors": ["Author A", "Author B"]

},

{

"\_id": 2,

"title": "Book Two",

"authors": ["Author A", "Author C"]

},

{

"\_id": 3,

"title": "Book Three",

"authors": ["Author B", "Author C"]

}

#### Solution:

We can use the aggregation pipeline to unwind the authors, group them, and then sort to find the most frequent author.

db.books.aggregate([

{ $unwind: "$authors" }, // Unwind authors array

{ $group: { \_id: "$authors", count: { $sum: 1 } } }, // Group by author and count

{ $sort: { count: -1 } }, // Sort by count in descending order

{ $limit: 1 } // Limit to the most frequent author

]);

**Explanation:**

* $unwind: Deconstructs the authors array.
* $group: Groups by author name and counts the occurrences.
* $sort: Sorts by the count in descending order.
* $limit: Limits the result to the top author.

### 2. ****Find Books Issued in the Last Month****

Given a collection of books and issues, find all books that were issued within the last 30 days.

#### Collection Example (issues):

{

"\_id": 1,

"book\_id": 101,

"issue\_date": ISODate("2024-09-05")

},

{

"\_id": 2,

"book\_id": 102,

"issue\_date": ISODate("2024-08-15")

}

#### Solution:

We can filter based on the issue\_date field using the $gte operator and new Date() to get the current date.

db.issues.find({

issue\_date: {

$gte: new Date(new Date().setDate(new Date().getDate() - 30)) // Date 30 days ago

}

});

**Explanation:**

* new Date(): Retrieves the current date.
* $gte: Ensures the issue\_date is greater than or equal to 30 days ago.

### 3. ****Top N Most Popular Books by Number of Issues****

Find the top N most issued books from the issues collection.

#### Collection Example (issues):

{

"\_id": 1,

"book\_id": 101,

"issue\_date": ISODate("2024-09-05")

},

{

"\_id": 2,

"book\_id": 101,

"issue\_date": ISODate("2024-09-10")

},

{

"\_id": 3,

"book\_id": 102,

"issue\_date": ISODate("2024-08-15")

}

#### Solution:

Use aggregation to count the number of issues for each book and sort them to get the most popular books.

db.issues.aggregate([

{ $group: { \_id: "$book\_id", issue\_count: { $sum: 1 } } }, // Group by book\_id, count issues

{ $sort: { issue\_count: -1 } }, // Sort by issue count descending

{ $limit: 5 // Limit to top 5 books (for example)

]);

**Explanation:**

* $group: Groups issues by book\_id and counts the number of issues per book.
* $sort: Sorts the results by issue count in descending order.
* $limit: Limits the result to the top N books (in this case, 5).

### 4. ****Find All Members Who Have Not Issued Any Books****

Find all members from the members collection who have not issued any books from the issues collection.

#### Collection Example (members):

#### {

#### "\_id": 1,

#### "name": "John Doe",

#### "member\_id": 1001

#### },

#### {

#### "\_id": 2,

#### "name": "Jane Smith",

#### "member\_id": 1002

#### }

#### Collection Example (issues):

#### {

#### "\_id": 1,

#### "book\_id": 101,

#### "cardNo": 1001,

#### "issue\_date": ISODate("2024-09-05")

#### }

#### Solution:

We can use the $lookup operator to join the members and issues collections and then filter members without any issues.

db.members.aggregate([

{

$lookup: {

from: "issues", // Join with 'issues' collection

localField: "member\_id", // Field from 'members'

foreignField: "cardNo", // Field from 'issues'

as: "issues" // Output array of related issues

}

},

{ $match: { issues: { $size: 0 } } } // Filter members with no issues

]);

**Explanation:**

* $lookup: Performs a left outer join between the members and issues collections.
* $match: Filters the result to include only members with no issues ($size: 0).

**5. Calculate Average Issue Duration for Each Book**

Given a collection of issues with issue\_date and return\_date, calculate the average duration a book was issued for.

**Collection Example (issues):**

**{**

**"\_id": 1,**

**"book\_id": 101,**

**"issue\_date": ISODate("2024-09-05"),**

**"return\_date": ISODate("2024-09-15")**

**},**

**{**

**"\_id": 2,**

**"book\_id": 102,**

**"issue\_date": ISODate("2024-08-15"),**

**"return\_date": ISODate("2024-08-30")**

**}**

**Solution:**

We can calculate the duration (difference between return\_date and issue\_date) and then compute the average.

db.issues.aggregate([

{

$project: {

book\_id: 1,

duration: {

$subtract: ["$return\_date", "$issue\_date"] // Calculate duration in milliseconds

}

}

},

{

$group: {

\_id: "$book\_id",

avg\_duration: { $avg: "$duration" } // Calculate average duration

}

},

{

$project: {

avg\_duration\_days: {

$divide: ["$avg\_duration", 1000 \* 60 \* 60 \* 24] // Convert milliseconds to days

}

}

}

]);

**Explanation:**

* $subtract: Calculates the duration by subtracting issue\_date from return\_date.
* $avg: Computes the average duration for each book\_id.
* $divide: Converts the duration from milliseconds to days.

**6. Find Books with Multiple Authors and Sort by Most Authors**

Find all books that have multiple authors and sort the results by the number of authors in descending order.

**Solution:**

We can use $size to count the number of authors in the array, and then filter and sort the result.

db.books.aggregate([

{

$project: {

title: 1,

authors: 1,

author\_count: { $size: "$authors" } // Calculate the number of authors

}

},

{ $match: { author\_count: { $gt: 1 } } }, // Filter books with more than one author

{ $sort: { author\_count: -1 } } // Sort by the number of authors in descending order

]);

**Explanation:**

* $size: Calculates the number of elements in the authors array.
* $match: Filters books with more than one author.
* $sort: Sorts the results by the number of authors in descending order.