The Data Stream

Your online streaming platform *Codecadestream* is starting to become popular and is starting to even rival Twitch. Large companies are reaching out to you for advertisement and sponsorship opportunities, but they would like to know some more information (e.g., viewership numbers). In this project, you'll retrieve information about your streaming service, including channel and user data. By filtering this data, you can use the information to find insights about the streaming platform to share with potential partners.

To accomplish this, you will:

- Connect to and view the contents of collections in a database.
- Perform simple filters on documents.
- Sort and count filtered documents.
- Use field projection to hide unnecessary information in queries.
- Perform complex filters with array data.

If you get stuck during this project or would like to see an experienced developer work through it, click **"Get Unstuck"** to see a project walkthrough video.

Tasks

12/12 Complete

Mark the tasks as complete by checking them off

MongoDB Database Inspection

1.

Your engineering team has moved the streaming service data over to MongoDB, and you'd like to take a look. View the list of databases to see which are available inside your MongoDB instance.
Hint

To view all of the databases, run `show dbs`.
2.

Looks like there is a database named `codecadestream`. That database is likely to contain important data. Connect to the `codecadestream` database to access its collections.
Hint

To connect to a database run `use <databasename>`.

Need another hint?

To connect to the `codecadestream` database, use the command `use codecadestream`.

3.

Once in the `codecadestream` database, take a look at the collections that exist. This will let you know what data is likely stored in the database.
Hint

To view all of the collections in the current database run `show collections`.

4.

You want to check out the data in the `channels` collection to see what important data points are being stored. Use the `.find()` method to ensure that documents exist in the `channels` collection. Take a look at how the data is formatted.
Hint

Remember, to show all the data in a collection, use the command: `db.<collection>.find()`.

Need another hint?

To examine all the data in the collection, use the command `db.channels.find()`

5.

You also want to now check out the data in the `users` collection to get a full picture of the database. Query the `users` collection to ensure it's populated with documents.

Be sure to examine how the data is formatted.
Hint

Remember, to show all the data in a collection, use the command: `db.<collection>.find()`.

Need another hint?

To examine all the data in the collection, use the command `db.users.find()`

Finding Important Data

6.

Your friend recently created a channel on your streaming platform, and you'd like to know how it's doing. They go by the streamer name of "Shark". Find their data in the database.
Hint

Use a query argument in the `.find()` method to filter through a collection. Use the following command format:

```
db.<collection>.find(
  {
    <field>: <value>,
    <second_field>: <value>
    ...
  }
);
```

Make sure to use the field name `streamer` and provide the correct string as its value.

Need another hint?

```
db.channels.find({streamer: "Shark"})
```

7.

Your marketing team would like to know which channels have 10,000 or more average views. Find all channels with a value of 10,000 for the `avg_views` field.
Hint

Use the `$gte` operator with the `.find()` method. Follow this format:

```
db.<collection>.find({ <field>: { $gte: <value>} })
```

Need another hint?

```
db.channels.find({ avg_views: { $gte: 10000} })
```

8.

In order to find the top channel on your platform, `.sort()` the result of the last task so that the channel with the highest average views is first (descending order).

Make sure to scroll up to where the command was entered to see the name of the top channel.

Hint

To sort the results, add the `.sort()` method to the end of the previous `.find()` method. The sort method accepts a field to sort on and a value representing descending order (`-1`) or ascending order (`1`). Here is the sample syntax:

```
db.<collection>.find(<filter>).sort({ <field>: <-1 or 1> })
```

The top channel should be `'Codecadetron Gaming'`.

Need another hint?

```
db.channels.find({ avg_views: { $gte: 10000} }).sort({avg_views: -1})
```

9.

Your marketing team doesn't need the entire long list of follower data. Retrieve just the streamer name and average views from the top channel from your earlier queries. Either add on to the existing query or filter using the top channel's name.

Hint

A field projection is the second argument to the `.find()` method. The argument should contain field-value pairs representing whether they are shown (`1`) or not (`0`). Here is an example:

```
db.<collection>.find(
  { <filter>: <value> },
  {
    <project_field_1>: <1 or 0>,
    <project_field_2>: <1 or 0>
})
```

Need another hint?

```
db.channels.find({channel_name:"Codecadetron Gaming"}, {streamer: 1, avg_views: 1})
```

Making Money

10.

You would like to know how many channels are making the popular channels money by having at least one subscriber (a user that pays for premium perks). Use `$elemMatch` and `.count()` to find out how many channels have at least one follower in the `followers` arrays who have an `is_subscribed` value of `true`.

Hint

To query with `$elemMatch` on a document containing an array, you can use the following syntax:

```
db.<collection>.find({ <filter_field>: { $elemMatch: { <array_field>: <value> }}})
```

Add `.count()` at the end to return the number of results. There should be 55 total channels.

Need another hint?

```
db.channels.find({ followers: { $elemMatch: { is_subscribed: true }}}).count()
```
```
</details>
```

11.

You have been approached by a large Esports company to determine if they would like to stream a tournament on your platform. They would like to know how many users have recently watched streams containing the tags `"esports"` and `"moba"`.
Hint

You can use `$all` to query for all the values contained in an array:

```
db.<collection>.find({ <filter_field>: { $all: [ <value_1>, <value_2> ]}})
```

Make sure to use the `users` collection and query based on the `watched` field. Add `.count()` at the end to return the number of results. There should be 7 users.

Need another hint?

```
db.users.find({ watched: { $all: [ "esports", "moba" ]}}).count()
```

12.

Congratulations on finishing the project! You were able to find valuable data about your streaming platform.

In this project, you:

- Performed queries using the `.find()` method.
- Used simple and complex filters to return specific documents.
- Removed unnecessary results from returned documents using field projection.
- Searched for values in array data using `$elemMatch` and `$all`.

Feel free to continue playing around with these collections to get even more practice.