---------- DESCRIPCIÓN ----------

Trabajaremos con una base de datos que contiene colecciones relacionadas con una aplicación de entretenimiento cinematográfico:

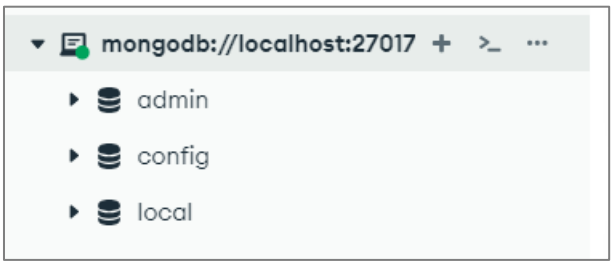
- users: Almacena información de usuarios/as, incluyendo nombres, emails y contraseñas cifradas.
- theatres: Contiene datos de cines, como ID, ubicación (dirección y coordenadas geográficas).
- sesiones: Guarda sesiones de usuario, incluyendo ID de usuario y tokens JWT para la autenticación.
- movies: Incluye detalles de películas, como trama, géneros, duración, elenco, comentarios, año de lanzamiento, directores, clasificación y premios.
- comments: Almacena comentarios de usuarios/as sobre películas, con información del autor/a del comentario, ID de la película, texto del comentario y la fecha.

Realizarás algunas consultas que te pide el cliente/a, quien está midiendo si serás capaz o no de hacerte cargo de la parte analítica del proyecto vinculado con su base de datos.

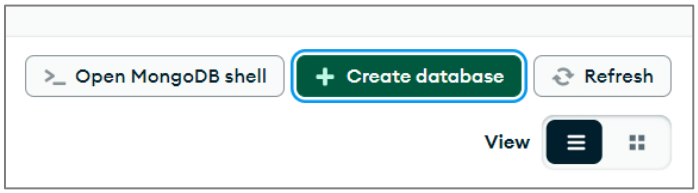
---------- NIVEL I ----------

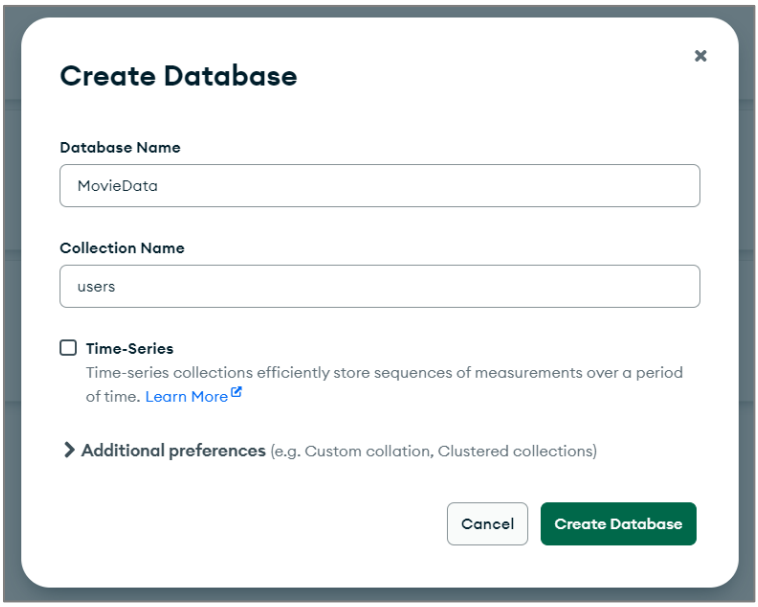
Crea una base de datos con MongoDB utilizando como colecciones los archivos adjuntos.

Para comenzar con la ejercitación de este módulo, primero instalé **MongoDB** en mi ordenador. Luego, configuré mi servidor local para poder trabajar con el programa de manera adecuada.

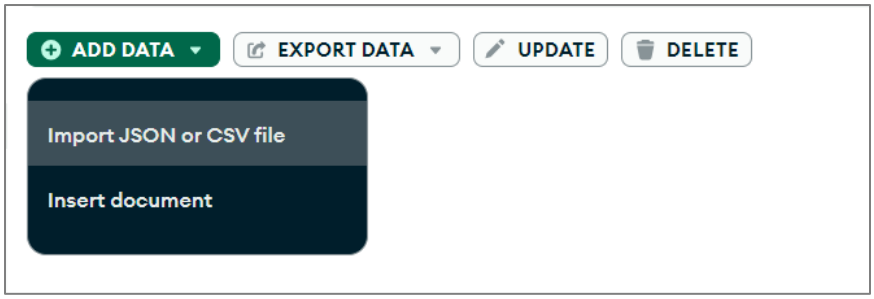


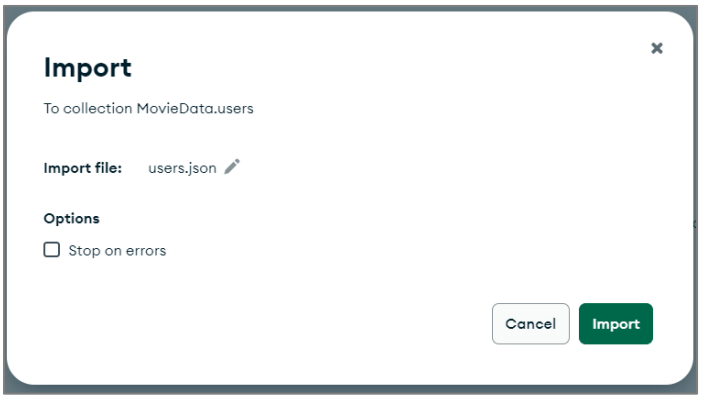
Después de establecer la conexión, seleccioné el ícono **"Create Database"** y creé la base de datos **"MovieData"**. Para poder crearla, añadí su primera colección llamada **"users"** y, finalmente, hice clic en el botón **"Create Database"**.

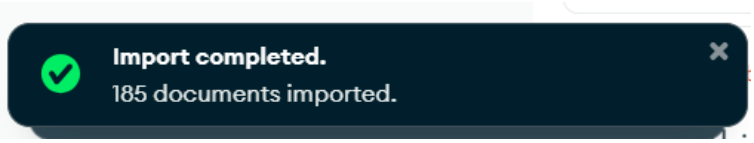




Una vez creada la colección **"users"**, procedí a cargar los datos proporcionados en el enunciado del sprint mediante un archivo JSON:

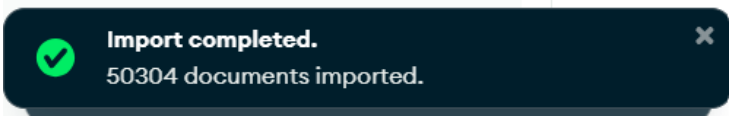




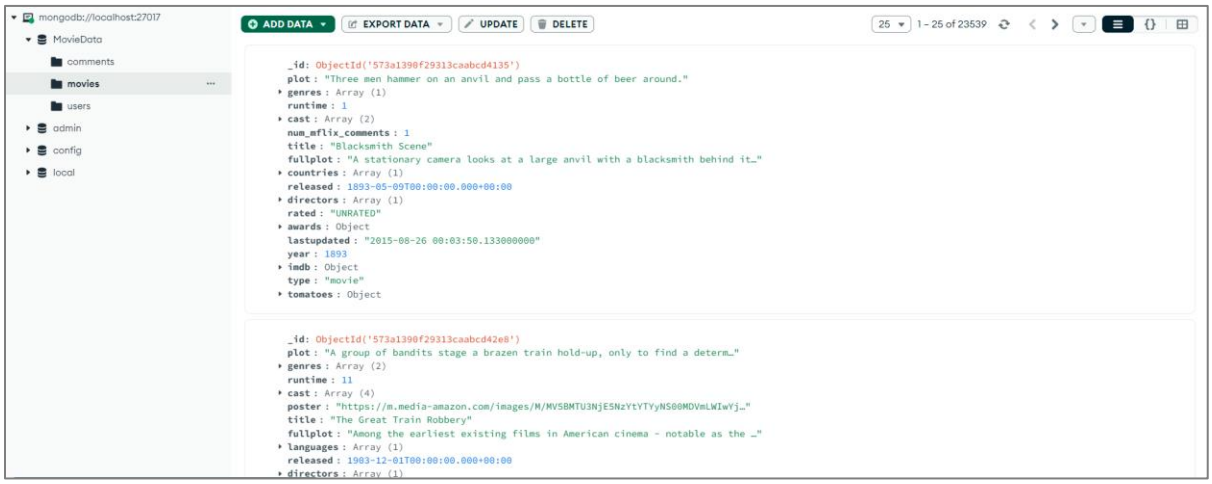


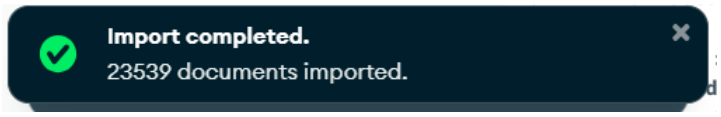
Del mismo modo, creé las demás colecciones e importé los datos correspondientes. A continuación, presento las demás colecciones creadas junto con sus datos importados:

- **Colección "comments":**

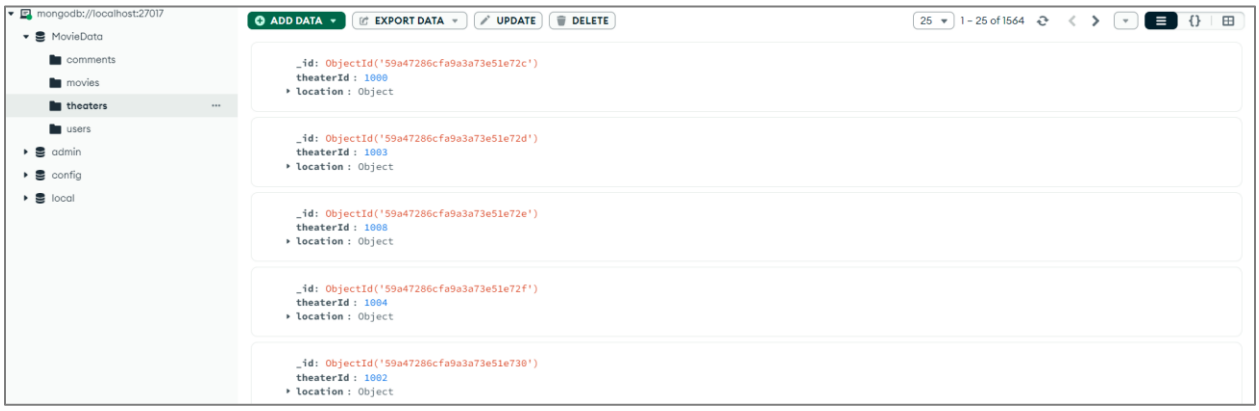


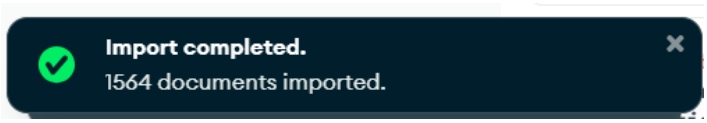
- **Colección "movies":**



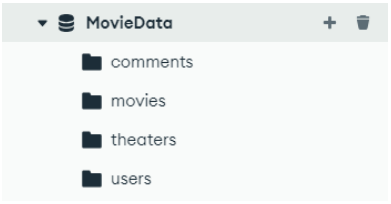


- **Colección "theatres":**



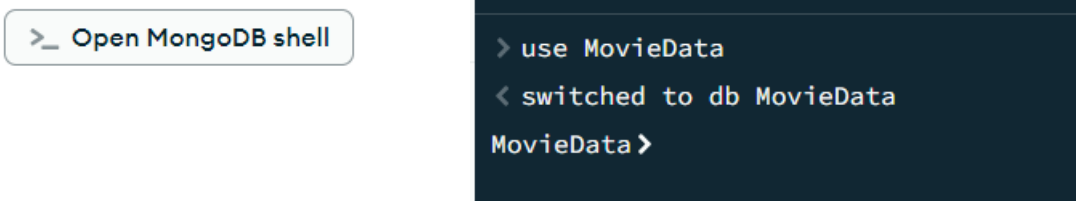


Una vez finalizada estas tareas, la base de datos **"MovieData"** quedó estructurada de la siguiente manera:



EJERCICIO 1

Antes de comenzar, es importante mencionar que para realizar estos ejercicios utilicé el **MongoDB Shell** para ejecutar las consultas necesarias:



A. Muestra los 2 primeros comentarios que aparecen en la base de datos.

```
MovieData> db.comments.find().limit(2)
```

```
> use MovieData
< switched to db MovieData
> db.comments.find().limit(2)
< {
    _id: ObjectId('5a9427648b0beebeb69579cc'),
    name: 'Andrea Le',
    email: 'andrea_le@fakegmail.com',
    movie_id: ObjectId('573a1390f29313caabcd418c'),
    text: 'Rem officiis eaque repellendus amet eos doloribus. Porro dolor voluptatum voluptates neque culpa molestias. Voluptate unde nulla temporibus ulla
    date: 2012-03-26T23:20:16.000Z
  }
  {
    _id: ObjectId('5a9427648b0beebeb69579cf'),
    name: 'Greg Powell',
    email: 'greg_powell@fakegmail.com',
    movie_id: ObjectId('573a1390f29313caabcd41b1'),
    text: 'Tenetur dolorum molestiae ea. Eligendi praesentium unde quod porro. Commodi nisi sit placeat rerum vero cupiditate neque. Dolorum nihil vero ani
    date: 1987-02-10T00:29:36.000Z
  }
```

En esta consulta se selecciona la colección comments dentro de la base de datos activa. Luego aplico el método .find() para buscar y recuperar todos los documentos de esa colección, y para restringir la salida de los dos primeros comentarios aplico .limit(2).

B. ¿Cuántos usuarios tenemos registrados?

```
MovieData> db.users.countDocuments()
```

```
> use MovieData
< switched to db MovieData
> db.users.countDocuments()
< 185
```

En esta consulta se selecciona la colección users dentro de la base de datos activa. Luego, utilizo el método .countDocuments() para contar el número total de documentos presentes en dicha colección, lo que da como resultado que existen 185 usuarios registrados.

C. ¿Cuántos cines existen en el estado de California?

```
db.theaters.countDocuments({"location.address.state": "CA"})
```

```
> db.theaters.countDocuments({"location.address.state": "CA"})
< 169
```

En esta consulta se selecciona la colección theaters de la base de datos activa. Luego, utilizo el método .countDocuments() para contar el número total de documentos presentes en dicha colección, pasando como parámetro el filtro { "location.address.state": "CA" }. Este filtro solo selecciona los registros de cines que están en California. Como resultado, se obtiene que existen 169 cines en California.

D. ¿Cuál fue el primer usuario en registrarse?

```
db.users.find().sort({_id:1}).limit(1)
```

```
> db.users.find().sort({_id:1}).limit(1)
< {
    _id: ObjectId('59b99db4cfa9a34dcd7885b6'),
    name: 'Ned Stark',
    email: 'sean_bean@gameofthron.es',
    password: '$2b$12$UREFwsRUoyF0CRqGNK0LzO0HM/jLhgUCNNIJ9RJAqMUQ74crlJ1Vu'
  }
```

Esta consulta selecciona la colección users de la base de datos activa y utilizo el método .find() para recuperar todos sus documentos. A continuación, se aplico .sort({ _id: 1 }) para ordenar los registros en forma ascendente según su identificador, posicionando al primer documento en la parte superior. Finalmente, con .limit(1) restrinjo el resultado a ese único documento, identificando al primer usuario registrado.

E. ¿Cuántas películas de comedia existen en nuestra base de datos?

```
db.movies.countDocuments({genres:"Comedy"})
```

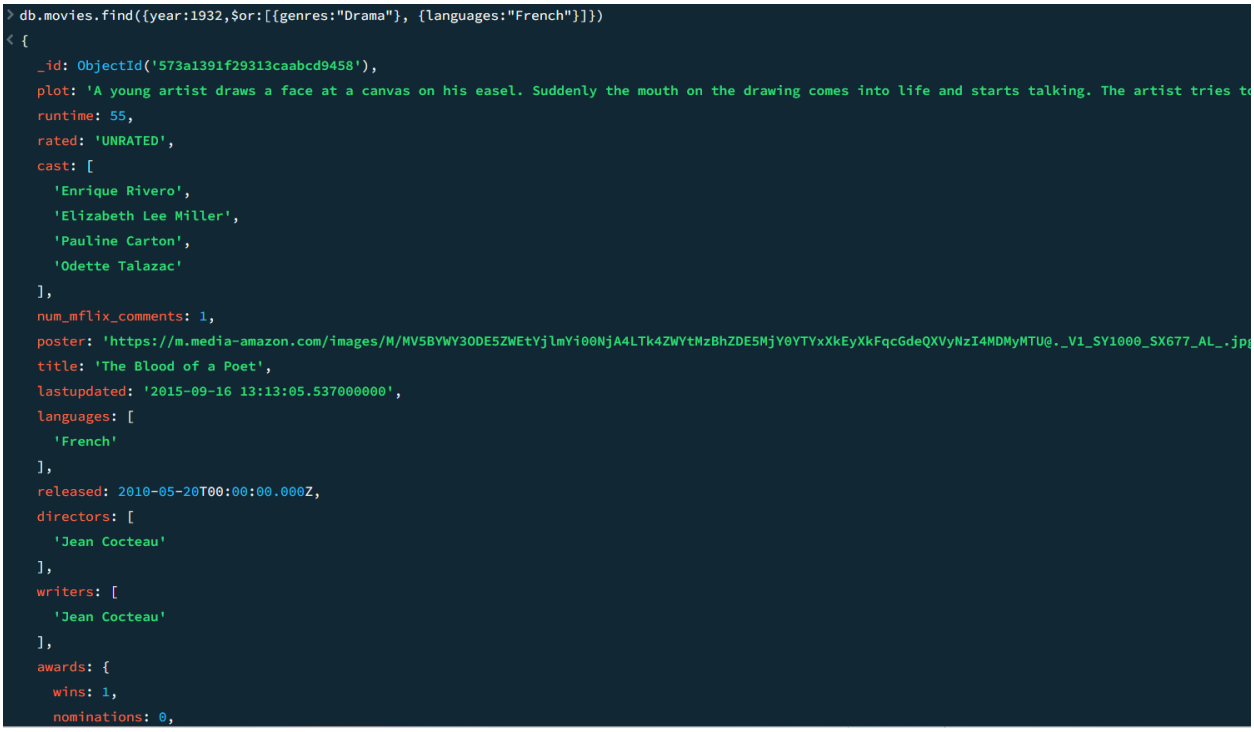
```
> db.movies.countDocuments({genres:"Comedy"})
< 7024
```

En esta consulta, se selecciona la colección movies de la base de datos activa. Luego, utilizo el método .countDocuments() para contar el número total de documentos en dicha colección, aplicando el filtro { genres: 'Comedy' }, que selecciona solo las películas con el género 'Comedy'. Como resultado, se obtiene un total de 7.024 documentos.

EJERCICIO 2

Muéstrame todos los documentos de las películas producidas en 1932, pero que el género sea drama o estén en francés.

```
db.movies.find({year:1932,$or:[{genres:"Drama"}, {languages:"French"}]})
```



```
> db.movies.find({year:1932,$or:[{genres:"Drama"}, {languages:"French"}]})
< {
    _id: ObjectId('573a1391f29313caabcd9458'),
    plot: 'A young artist draws a face at a canvas on his easel. Suddenly the mouth on the drawing comes into life and starts talking. The artist tries to
    runtime: 55,
    rated: 'UNRATED',
    cast: [
      'Enrique Rivero',
      'Elizabeth Lee Miller',
      'Pauline Carton',
      'Odette Talazac'
    ],
    num_mflix_comments: 1,
    poster: 'https://m.media-amazon.com/images/M/MV5BYWY3ODE5ZWEtYjlmYi00NjA4LTk4ZWYtMzBhZDE5MjY0YTYxXkEyXkFqcGdeQXVyNzI4MDMyMTU@._V1_SY1000_SX677_AL_.jpg
    title: 'The Blood of a Poet',
    lastupdated: '2015-09-16 13:13:05.537000000',
    languages: [
      'French'
    ],
    released: 2010-05-20T00:00:00.000Z,
    directors: [
      'Jean Cocteau'
    ],
    writers: [
      'Jean Cocteau'
    ],
    awards: {
      wins: 1,
      nominations: 0,
```

En esta consulta, se selecciona la colección movies de la base de datos activa. Luego, utilizo el método .find() para recuperar todos los documentos que cumplen con el filtro { year: 1932, $or: [{ genres: 'Drama' }, { languages: 'French' }] }, el cual selecciona únicamente las películas estrenadas en 1932 que tienen el género "Drama" o están en francés.

EJERCICIO 3

Muéstrame todos los documentos de películas estadounidenses que tengan entre 5 y 9 premios que fueron producidas entre 2012 y 2014.

```
db.movies.find({'countries':'USA', 'awards.wins':{$gte: 5, $lte: 9}, 'year':{$gte: 2012, $lte: 2014}})
```

```
> db.movies.find({'countries':'USA', 'awards.wins':{$gte: 5, $lte: 9}, 'year':{$gte: 2012, $lte: 2014}})
< {
    _id: ObjectId('573a13acf29313caabd29366'),
    fullplot: "The manager of the negative assets sector of Life magazine, Walter Mitty, has been working for sixteen years for the magazine and has a tediou
    imdb: {
      rating: 7.4,
      votes: 211230,
      id: 359950
    },
    year: 2013,
    plot: 'When his job along with that of his co-worker are threatened, Walter takes action in the real world embarking on a global journey that turns into
    genres: [
      'Adventure',
      'Comedy',
      'Drama'
    ],
    rated: 'PG',
    metacritic: 54,
    title: 'The Secret Life of Walter Mitty',
    lastupdated: '2015-08-31 00:10:51.747000000',
    languages: [
```

En esta consulta se selecciona la colección movies de la base de datos activa. Luego, utilizo el método .find() para recuperar los documentos que cumplen con tres condiciones: la película debe ser estadounidense (filtrando por el campo countries con el valor "USA"), debe haber ganado entre 5 y 9 premios (aplicando el operador de rango { $gte: 5, $lte: 9 } sobre el campo awards.wins) y debe haber sido producida entre 2012 y 2014 (usando { $gte: 2012, $lte: 2014 } en el campo year).

---------- NIVEL 2 ----------

EJERCICIO 1

Cuenta cuántos comentarios escribe un usuario que utiliza "GAMEOFTHRON.ES" como dominio de correo electrónico.

```
db.comments.countDocuments({email:{$regex: "gameofthron.es$"}})
```

```
> db.comments.countDocuments({email:{$regex: "gameofthron.es$"}})
< 22841
```

En esta consulta se selecciona la colección comments de la base de datos activa. Luego, utilizo el método .countDocuments() para contar la cantidad total de documentos que cumplen con el filtro { email: { $regex: "gameofthron.es$" } }, el cual busca correos electrónicos que terminen en "gameofthron.es" utilizando una expresión regular. Como resultado, se obtiene un total de 22.841 documentos.

EJERCICIO 2

¿Cuántos cines existen en cada código postal situados dentro del estado Washington DC (DC)?

```
db.theaters.aggregate([
  { $match: { "location.address.state": "DC" } },
  { $group: { _id: "$location.address.zipcode", totalTheaters:{$sum:1}}},
  { $project:{_id:0, zipcode:"$_id", totalTheaters:1}}])
```

```
> db.theaters.aggregate([
    { $match: { "location.address.state": "DC" } },
    { $group: { _id: "$location.address.zipcode", totalTheaters:{$sum:1}}},
    { $project:{_id:0, zipcode:"$_id", totalTheaters:1}}])
< {
    totalTheaters: 1,
    zipcode: '20016'
  }
  {
    totalTheaters: 1,
    zipcode: '20010'
  }
  {
    totalTheaters: 1,
    zipcode: '20002'
  }
```

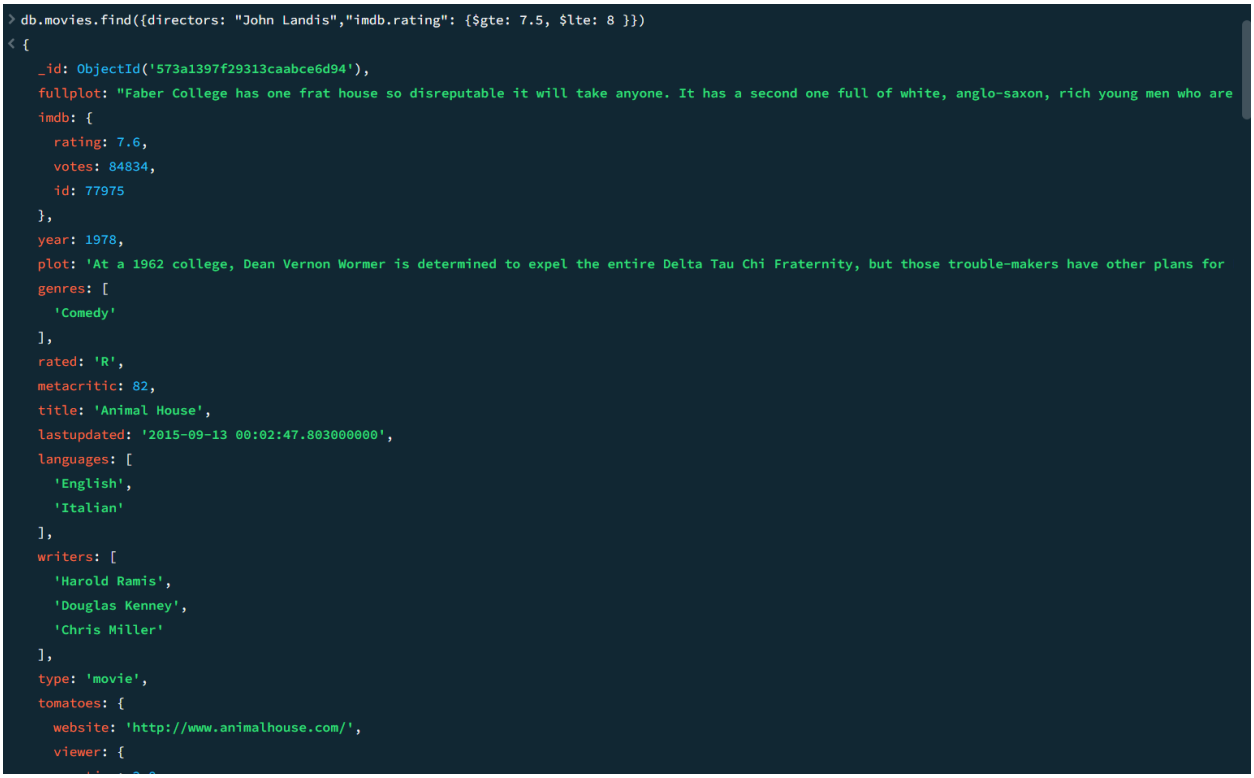
En esta consulta se selecciona la colección theaters de la base de datos activa. Luego, utilizo el método .aggregate() con tres etapas: primero, $match filtra los documentos para incluir solo aquellos donde el campo location.address.state sea "DC". Luego, $group agrupa los cines por código postal (location.address.zipcode) y cuenta la cantidad total de cines en cada uno con $sum: 1. Finalmente, $project ajusta la salida, eliminando el campo _id y renombrando _id como zipcode, manteniendo únicamente los campos zipcode y totalTheaters en el resultado.

---------- NIVEL 3 ----------

EJERCICIO 1

Encuentra todas las películas dirigidas por John Landis con una puntuación IMDb (Internet Movie Database) de entre 7,5 y 8.

```
db.movies.find({directors: "John Landis","imdb.rating": {$gte: 7.5, $lte: 8 }})
```



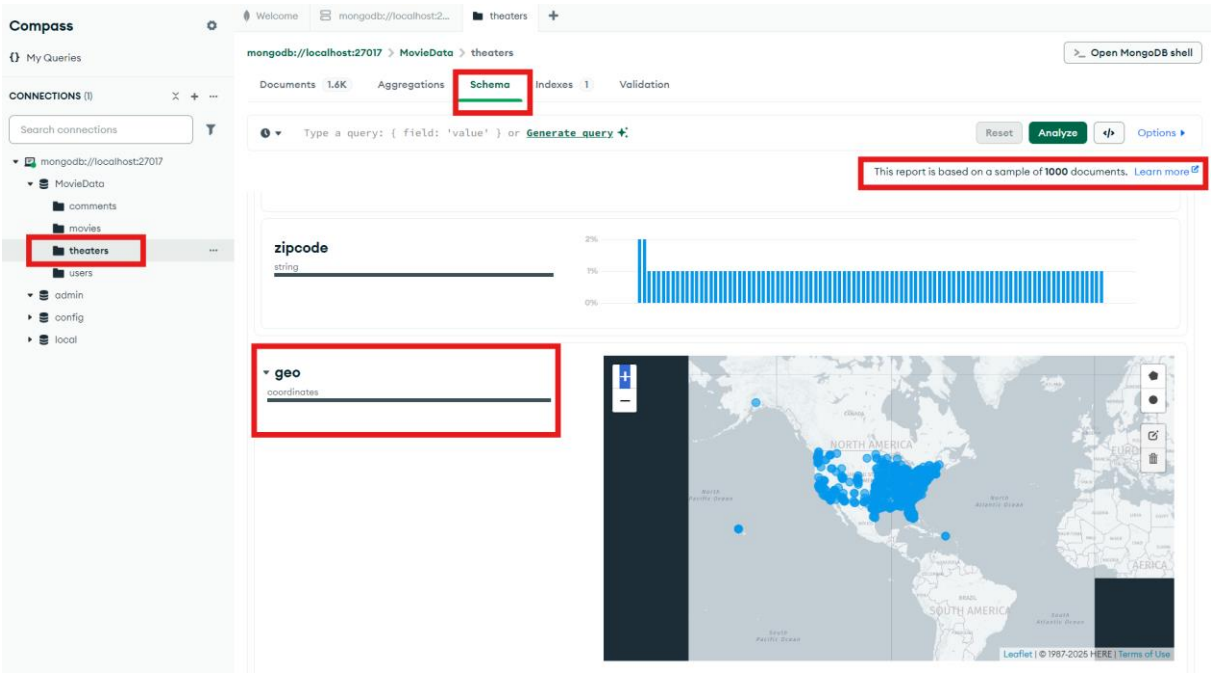
```
> db.movies.find({directors: "John Landis","imdb.rating": {$gte: 7.5, $lte: 8 }})
< {
    _id: ObjectId('573a1397f29313caabce6d94'),
    fullplot: "Faber College has one frat house so disreputable it will take anyone. It has a second one full of white, anglo-saxon, rich young men who are
    imdb: {
      rating: 7.6,
      votes: 84834,
      id: 77975
    },
    year: 1978,
    plot: 'At a 1962 college, Dean Vernon Wormer is determined to expel the entire Delta Tau Chi Fraternity, but those trouble-makers have other plans for
    genres: [
      'Comedy'
    ],
    rated: 'R',
    metacritic: 82,
    title: 'Animal House',
    lastupdated: '2015-09-13 00:02:47.803000000',
    languages: [
      'English',
      'Italian'
    ],
    writers: [
      'Harold Ramis',
      'Douglas Kenney',
      'Chris Miller'
    ],
    type: 'movie',
    tomatoes: {
      website: 'http://www.animalhouse.com/',
      viewer: {
```

En esta consulta se selecciona la colección movies de la base de datos activa. Luego, utilizo el método .find() para buscar todas las películas que cumplan dos condiciones: que el campo directors contenga el nombre "John Landis" y que la puntuación de IMDb (imdb.rating) esté dentro del rango de 7.5 a 8, utilizando los operadores $gte (mayor o igual) y $lte (menor o igual).

EJERCICIO 2

Muestra en un mapa la ubicación de todos los teatros de la base de datos.

Para realizar este análisis en MongoDB, primero debemos seleccionar la colección Theaters dentro de nuestra base de datos. Luego, accedemos a la pestaña "Schema" para examinar la estructura de los datos. Dentro del esquema, ubicamos el campo "location", y dentro de este, seleccionamos la subestructura "geo", que contiene las coordenadas geográficas de los teatros. Al realizar este análisis, podemos visualizar la ubicación de los primeros 1000 registros, lo que nos permite comprender mejor la distribución geoespacial de los teatros almacenados en la base de datos.


Compass
My Queries
CONNECTIONS (1)
Search connections
mongodb://localhost:27017
MovieData
comments
movies
theaters
users
admin
config
local
Welcome
mongodb://localhost:2...
theaters
mongodb://localhost:27017 > MovieData > theaters
Open MongoDB shell
Documents 1.6K
Aggregations
Schema
Indexes 1
Validation
Type a query: { field: 'value' } or Generate query
Reset
Analyze
Options
This report is based on a sample of 1000 documents. Learn more
zipcode
string
geo
coordinates
NORTH AMERICA
SOUTH AMERICA
Leaflet | © 1987-2025 HERE | Terms of Use