

Emilio Tornos Iniesta - Tasca S5.01. Consultes amb MongoDB

Treballarem amb una base de dades que conté col·leccions relacionades amb una aplicació d'entreteniment cinematogràfic:

- **users:** Emmagatzema informació d'usuaris/es, incloent-hi noms, emails i contrasenyes xifrades.
- **theatres:** Conté dades de cinemes, com ID, ubicació (direcció i coordenades geogràfiques).
- **sessions:** Guarda sessions d'usuari, incloent-hi ID d'usuari i tokens JWT per a l'autenticació.
- **movies:** Inclou detalls de pel·lícules, com a trama, gèneres, durada, elenc, comentaris, any de llançament, directors, classificació i premis.
- **comments:** Emmagatzema comentaris d'usuaris/es sobre pel·lícules, amb informació de l'autor/a del comentari, ID de la pel·lícula, text del comentari i la data.

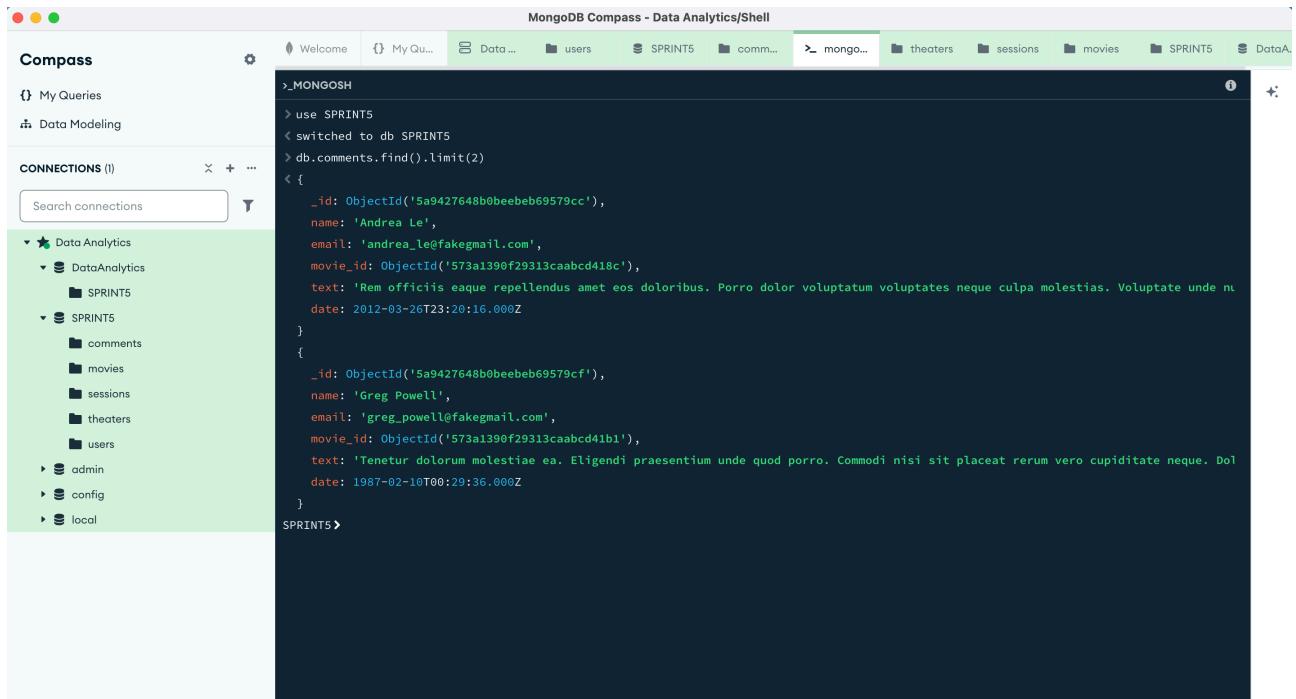
Duràs a terme algunes consultes que et demana el client/a, el qual està mesurant si seràs capaç o no de fer-te càrrec de la part analítica del projecte vinculat amb la seva base de dades.

NIVELL 1

Crea una base de dades amb MongoDB utilitzant com a col·leccions els arxius adjunts.

Exercici 1

- Mostra els 2 primers comentaris que hi ha en la base de dades.



The screenshot shows the MongoDB Compass interface. On the left, there's a sidebar with 'Compass' settings, 'My Queries', 'Data Modeling', and a 'CONNECTIONS ()' section. Below that is a tree view of databases: 'Data Analytics' (selected), 'SPRINT5' (expanded), containing 'comments', 'movies', 'sessions', 'theaters', and 'users'. Other databases listed are 'admin', 'config', and 'local'. The main area is titled 'MongoDB Compass - Data Analytics/Shell' and shows the mongo shell command line. The command `db.comments.find().limit(2)` is run, and the results are displayed as two documents. The first document is for 'Andrea Le', and the second is for 'Greg Powell'.

```
>_MONGOSH
> use SPRINT5
< switched to db SPRINT5
> db.comments.find().limit(2)
< {
    "_id": ObjectId('5a9427648b0beebe69579cc'),
    "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 n",
    "date": 2012-03-26T23:20:16.000Z
}
{
    "_id": ObjectId('5a9427648b0beebe69579cf'),
    "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. Dol",
    "date": 1987-02-10T00:29:36.000Z
}
SPRINT5>
```

BREVE EXPLICACIÓN;

- En **MONGODB** las colecciones serían la equivalencia a las **tablas** en **SQL** y los documentos serían la equivalencia a las **filas** en **SQL**.
- En mi caso la **base de datos** que utilizo para resolver los ejercicios se llama **SPRINT5** y contiene las **5 colecciones** del ejercicio.
- “Ignorar la base de datos **DataAnalytics**, la he estado usando antes para explorar y profundizar haciendo pruebas con **MONGODB**”.

Resolución:

- Mi query **db.comments.find().limit(2)** hace lo siguiente:
- **use SPRINT5:** Selecciona la base de datos de trabajo, a partir de este momento, el objeto **db** apuntará directamente a **SPRINT5** para ejecutar todas las consultas a través de los métodos.
- **db:** Hace referencia a la base de datos activa (**SPRINT5**).
- **db** en la consola, técnicamente te refieres al **objeto** que apunta a la base de datos activa en ese momento. Es decir, si escribiste **use SPRINT5**, entonces **db** efectivamente representa a esa base de datos.
- **.find():** Busca los documentos
- **.limit(2):** Le dice a MongoDB que se detenga después de encontrar los dos primeros resultados.

- Quants usuaris tenim registrats?

```

MongoDB Compass - Data Analytics/Shell

Compass
  Welcome
  My Qu...
  Data ...
  SPRINT5
  users
  mongo...
  comm...
  theaters
  sessions
  movies
  SPRINT5
  DataA.

>_MONGOSH
> use SPRINT5
< switched to db SPRINT5
> db.users.countDocuments({})
< 185
SPRINT5>

```

Connections (1)

Search connections

Data Analytics

- SPRINT5
- SPRINTS
- admin
- config
- local

Resolución:

- Mi query `db.users.countDocuments({})` hace lo siguiente:
 - **db**: Hace referencia a la base de datos activa (**SPRINT5**).
 - **.users**: Indica la colección **users**.
 - **.countDocuments({})**: Es el método que cuenta cuántos documentos existen en la colección.
 - Las llaves vacías {} indican que no estamos aplicando ningún filtro, de esta manera nos devuelve el total de usuarios de la colección.

- Quants cinemes hi ha en l'estat de Califòrnia?

```

MongoDB Compass - Data Analytics/Shell

Compass
  Welcome
  My Qu...
  Data ...
  SPRINT5
  users
  mongo...
  comm...
  theaters
  sessions
  movies
  mongo...
  SPRINT5
  DataA.

>_MONGOSH
> use SPRINT5
< switched to db SPRINT5
> db.theaters.distinct("location.address.state")
< [
    'AK', 'AL', 'AR', 'AZ', 'CA', 'CO', 'CT',
    'DC', 'DE', 'FL', 'GA', 'HI', 'IA', 'ID',
    'IL', 'IN', 'KS', 'KY', 'LA', 'MA', 'MD',
    'ME', 'MT', 'MN', 'MO', 'MS', 'MT', 'NC',
    'ND', 'NE', 'NH', 'NJ', 'NM', 'NV', 'NY',
    'OH', 'OK', 'OR', 'PA', 'PR', 'RI', 'SC',
    'SD', 'TN', 'TX', 'UT', 'VA', 'VT', 'WA',
    'WI', 'WV', 'WY'
]
> db.theaters.countDocuments({ "location.address.state": "CA" })
< 169
SPRINT5>

```

Connections (1)

Search connections

Data Analytics

- SPRINT5
- SPRINTS
- admin
- config
- local

Exploración:

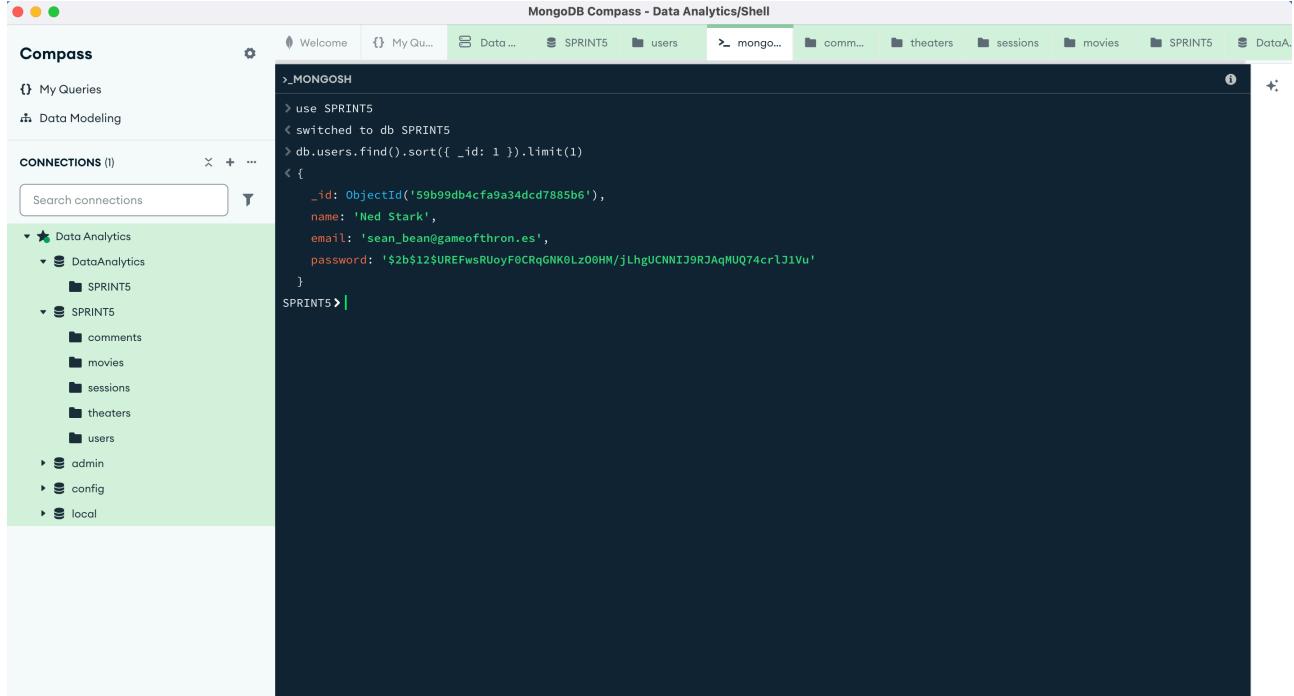
Antes de realizar el conteo, utilice el método `db.theaters.distinct("location.address.state")`.

- Este método me **devuelve una lista única** de todas las **abreviaturas** de los **estados** que **existen** en la colección **theaters**, **eliminando duplicados**.
- De esta manera he podido ver como debía filtrar el estado de California en este caso "**CA**".

Resolución:

- Mi query `db.theaters.countDocuments({ "location.address.state": "CA" })` hace lo siguiente:
 - **db**: Hace referencia a la base de datos activa (**SPRINT5**).
 - **.theaters**: Indica la colección de cines (teatros) sobre la que queremos realizar la consulta.
 - **.countDocuments({ ... })**: Es el método que cuenta los documentos, esta vez sí que usaremos filtros.
 - **"location.address.state": "CA"**: Es el filtro de búsqueda, utilizo la **notación de puntos** “dentro.de.comillas” para acceder a campos anidados, **navegando desde el objeto location**, pasando por **address**, hasta llegar al campo **state**.
- El valor **“CA”** filtra los resultados para **incluir solo** los registros del estado de **California**.

• Quin va ser el primer usuari/ària en registrar-se?



```
_id: ObjectId('59b99dbacfa9a34cd7885b6'),
name: 'Ned Stark',
email: 'sean_bean@gameofthron.es',
password: '$2b$12$UREFwsUoyFOCRqGNK0Lz00HM/jLhgUCNNIJ9RJAqMUQ74crlJ1Vu'
```

Resolución:

- Mi query `db.users.find().sort({ _id: 1 }).limit(1)` hace lo siguiente:
 - **db.users.find()**: Busca documentos en la colección de **users**.
 - **.sort({ _id: 1 })**: Ordena los documentos por **_id**.
En este punto estuve buscando un campo similar a algo como **FechaCreación**, que me ayudara a ordenar pero no encontré nada parecido en la colección **users**.
 - Entonces ordeno por su identificador único, ya que los **ObjectId** en MongoDB **siempre incluyen** una **marca de tiempo**, ordenar por **_id** de forma **ascendente** (1) equivale a ordenar por fecha de creación de la más antigua a la más reciente.
 - **.limit(1)**: Limita a mostrar únicamente el primer registro de la lista ordenada, que corresponde al primer usuario creado.

- Quantes pel·lícules de comèdia hi ha en la nostra base de dades?

The screenshot shows the MongoDB Compass interface. On the left, the 'Connections' sidebar lists several databases under 'Data Analytics' (SPRINTS, comments, movies, sessions, theaters, users) and other databases like admin, config, and local. The main pane displays the MongoDB shell command history. The user has run the command `db.movies.distinct("genres")` which returns a list of genres: Action, Adventure, Animation, Biography, Comedy, Crime, Documentary, Drama, Family, Fantasy, Film-Noir, History, Horror, Music, Musical, Mystery, News, Romance, Sci-Fi, Short, Sport, Talk-Show, Thriller, War, and Western. The user then runs `db.movies.countDocuments({ genres: "Comedy" })` which returns the count 7024. The database name SPRINTS is also visible at the bottom of the command history.

```

> use SPRINTS
< switched to db SPRINTS
> db.movies.distinct("genres")
< [
    'Action',
    'Adventure',
    'Animation',
    'Biography',
    'Comedy',
    'Crime',
    'Documentary',
    'Drama',
    'Family',
    'Fantasy',
    'Film-Noir',
    'History',
    'Horror',
    'Music',
    'Musical',
    'Mystery',
    'News',
    'Romance',
    'Sci-Fi',
    'Short',
    'Sport',
    'Talk-Show',
    'Thriller',
    'War',
    'Western'
]
> db.movies.countDocuments({ genres: "Comedy" })
< 7024
SPRINTS>

```

Exploración:

Antes de realizar el conteo, utilizo el método `db.movies.distinct("genres")`.

- Este método **recorre** toda la colección **movies** y **devuelve** una **lista única** con **todos** los **géneros** existentes, **eliminando duplicados**.
- De esta manera he podido ver como debía filtrar las películas de comedia en este caso "**Comedy**" en **inglés** y con la **primera letra en mayúscula**.

Resolución:

- Mi query `db.movies.countDocuments({ genres: "Comedy" })` hace lo siguiente:

• **db.movies**: Referencia a la colección de películas en la base de datos activa.

• **.countDocuments({})**: Es el método que cuenta los documentos filtrados.

• **{ genres: "Comedy" }**: Es el filtro que he aplicado.

Aunque el campo **genres** sea un **array (lista de valores)**, **MongoDB** tiene la capacidad de buscar directamente dentro de esa lista y contabilizar todos los documentos que contengan el valor "**Comedy**".

Exercici 2

Mostra'm tots els documents de les pel·lícules produïdes en 1932, però que el gènere sigui drama o estiguin en francès.

Exploración:

- Comienzo utilizando el método **db.movies.findOne()** y **confirmo** que el **año** se registra como un **valor numérico** ya que **no** está entre **comillas**.

```
>_MONGOSH
> use SPRINTS
< switched to db SPRINTS
> db["movies"].findOne()
< {
    "_id": ObjectId("573a1390f29313caabed4135"),
    "plot": "Three men hammer on an anvil and pass a bottle of beer around.",
    "genres": [
        "Short"
    ],
    "runtime": 1,
    "cast": [
        "Charles Kaysen",
        "John Ott"
    ],
    "num_mflix_comments": 1,
    "title": "Blacksmith Scene",
    "fullplot": "A stationary camera looks at a large anvil with a blacksmith behind it and one on either side. The smith in the middle draws a heated metal rod from the fire,",
    "countries": [
        "USA"
    ],
    "released": "1893-05-09T00:00:00.000Z",
    "directors": [
        "William K.L. Dickson"
    ],
    "rated": "UNRATED",
    "awards": {
        "wins": 1,
        "nominations": 0,
        "text": "1 win."
    },
    "lastupdated": "2015-08-26 00:03:50.133000000",
    "year": 1893, ←
    "imdb": {
        "rating": 6.2,
        "votes": 1189,
        "id": 5
    },
    "type": "movie",
}
```

Luego utilizo los métodos **db.movies.distinct("genres")**, y **db.movies.distinct("languages")** para verificar las nomenclaturas exactas de los valores.

```
>_MONGOSH
> db["movies"].distinct("genres")
< [
    "Action", "Adventure", "Animation",
    "Biography", "Comedy", "Crime",
    "Documentary", "Drama", "Family",
    "Fantasy", "Film-Noir", "History",
    "Horror", "Music", "Musical",
    "Mystery", "News", "Romance",
    "Sci-Fi", "Short", "Sport",
    "Talk-Show", "Thriller", "War",
    "Western"
]
```

```
_MONGOSH
> db.movies.distinct("languages")
< [
  'Ancient (to 1453)',
  'Old',
  'Abkhazian',
  'Aboriginal',
  'Acholi',
  'Afrikaans',
  'Aïdeukrou',
  'Albanian',
  'Algonquin',
  'American Sign Language',
  'Amharic',
  'Apache languages',
  'Arabic',
  'Aramaic',
  'Arapaho',
  'Armenian',
  'Assamese',
  'Assyrian Neo-Aramaic',
  'Athapaskan languages',
  'Awadhi',
  'Aymara',
  'Azerbaijani',
  'Balinese',
  'Bambara',
  'Basque',
  'Belarusian',
  'Bengali',
  'Berber languages',
  'Bhojpuri',
  'Bosnian',
  'Brazilian Sign Language',
  'Breton',
  'British Sign Language',
  'Bulgarian',
  'Burmese',
  'Cantonese',
  'Central'
]
```

Ahora ya sé que los valores correctos son "**Drama**" y "**French**" y que el **año** es un **valor numérico**.

```
_MONGOSH
> 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 wipe it away with his hand, but fails. The mouth continues to talk, revealing the artist's darkest fears and desires. The artist becomes increasingly distressed and ends up fainting. The drawing then disappears, leaving only a blank canvas behind.',
  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/MV5BYWY3ODE5ZWEtYjlmY100NjA4LTk4ZWYtHzbZDE5MjY0YTtXKFcGdeQXVvNzI4MDMyMTU0_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,
    text: '1 win.'
  },
  year: 1932,
  imdb: {
    rating: 7.5,
    votes: 3903,
    id: 21331
  },
  countries: [
    'Spain'
  ]
}
```

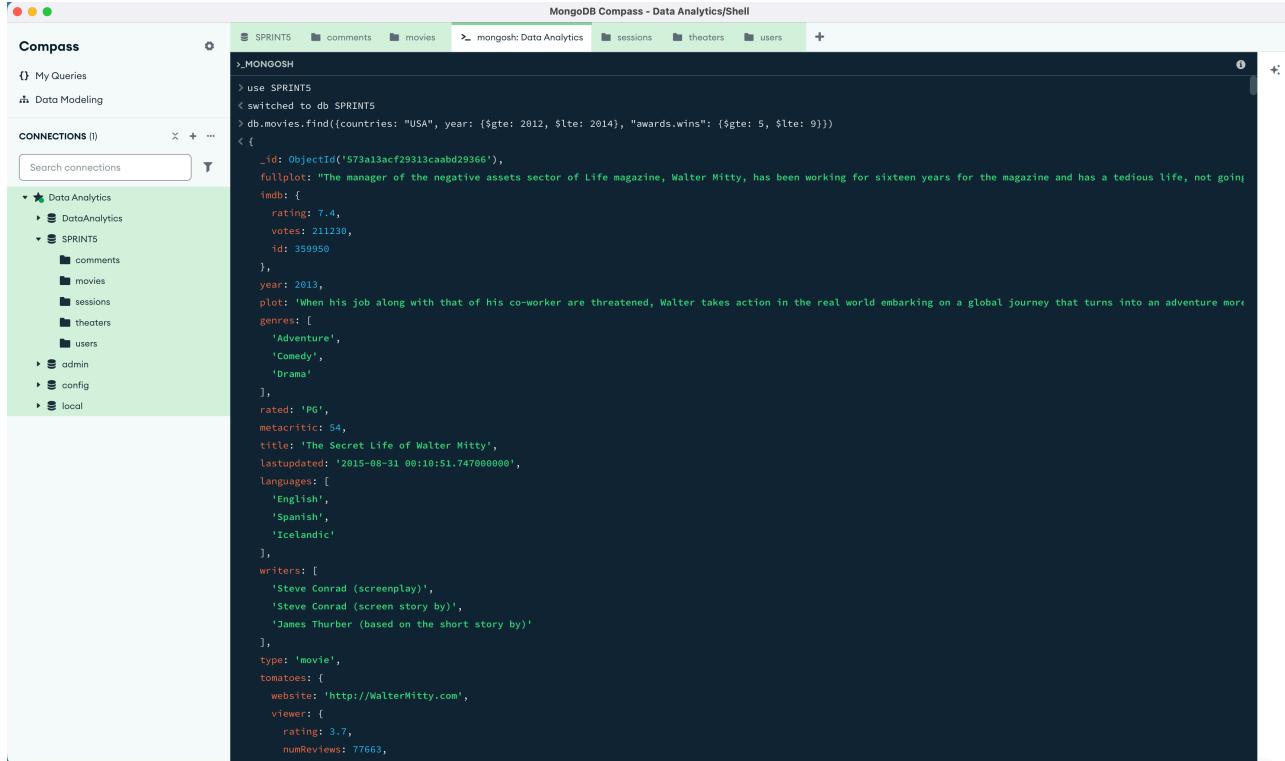
Resolución: Mi query `db.movies.find({ year: 1932, $or: [{ genres: "Drama" }, { languages: "French" }]}))` hace lo siguiente:

- **db.movies.find():** Inicia la búsqueda en la colección de películas.
- **year: 1932:** Establece una **condición obligatoria**, solo se muestran películas de ese año.
- **\$or: [...]:** Es el operador lógico que permite que se cumpla **una u otra** condición de las que aparecen en la lista.
- **[]:** Representan un **array (lista de valores)**. El operador \$or siempre los necesita para contener las diferentes opciones que queremos filtrar.
- **{ genres: "Drama" }, { languages: "French" }:** Son las dos condiciones alternativas.

El resultado de mi query **siempre devolverá**, películas de **1932** que sean **dramas** y estén en **francés**, películas de **1932** que sean **dramas**, o películas de **1932** que estén en **francés**, pero **nunca** películas de **1932** que no sean **drama ni** estén en **francés**.

Exercici 3

Mostra'm tots els documents de pel·lícules estatunidenques que tinguin entre 5 i 9 premis que van ser produïdes entre 2012 i 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 tedious life, not going
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 an adventure more
genres: [
  'Adventure',
  'Comedy',
  'Drama'
],
rated: 'PG',
metacritic: 54,
title: 'The Secret Life of Walter Mitty',
lastupdated: '2015-08-31 00:10:51.747000000',
languages: [
  'English',
  'Spanish',
  'Icelandic'
],
writers: [
  'Steve Conrad (screenplay)',
  'Steve Conrad (screen story by)',
  'James Thurber (based on the short story by)'
],
type: 'movie',
tomatoes: {
  website: 'http://WalterMitty.com',
  viewer: {
    rating: 3.7,
    numReviews: 77663
  }
}
```

Exploración: Para resolver este ejercicio, he analizado la estructura de la colección **movies** con el método **db.movies.findOne()** y me he fijado en esto:

- El **país** se encuentra en el campo **countries**, que es una **lista de strings**.
- Los **premios ganados** están **dentro** de un **objeto** llamado **awards**, específicamente en el **campo wins**.
- Tanto el **año (year)** como las **victorias (wins)** son **valores numéricos**, lo que nos **permite** usar **operadores de comparación**.

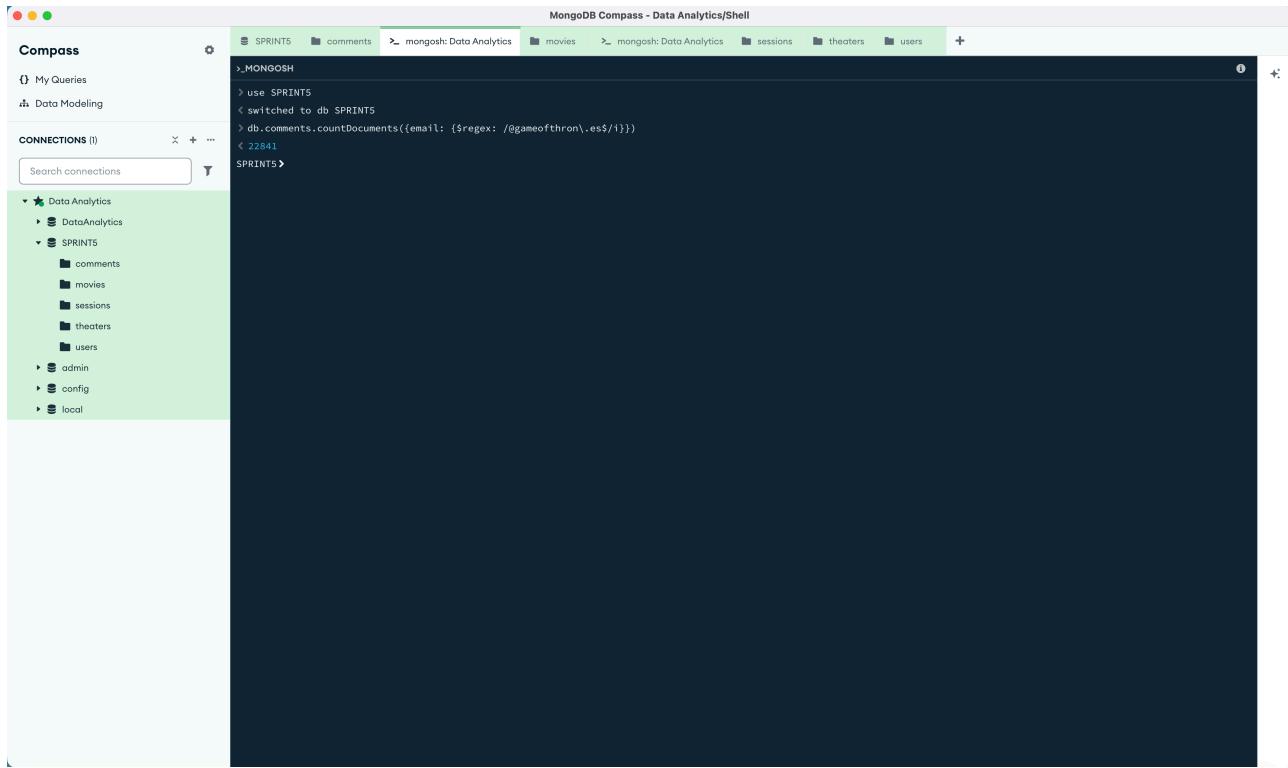
Resolución: Mi query **db.movies.find({countries: "USA", year: {\$gte: 2012, \$lte: 2014}, "awards.wins": {\$gte: 5, \$lte: 9}})** hace lo siguiente:

- **countries: "USA"**: Filtra los documentos para que solo aparezcan películas producidas en Estados Unidos.
- **year: { \$gte: 2012, \$lte: 2014 }**: Define un rango cerrado para el año, usando **\$gte** (mayor o igual a 2012) y **\$lte** (menor o igual a 2014) combinados para obtener el periodo que me solicita el ejercicio.
- **"awards.wins": { \$gte: 5, \$lte: 9 }**: Aplicamos la misma lógica de rango al campo de premios ganados. Usamos la **notación de puntos y las comillas** para **entrar** en el **objeto awards** y llegar al valor de **wins**.
- **Lógica AND**: Al colocar estas **tres condiciones separadas por comas**, MongoDB **solo** nos **devolverá** las **películas** que **cumplan todos los requisitos a la vez**.

NIVELL 2

Exercici 1

Compte quants comentaris escriu un usuari/ària que utilitza "GAMEOFTHRON.ES" com a domini de correu electrònic.



The screenshot shows the MongoDB Compass interface. On the left, the sidebar displays a tree view of databases: Data Analytics, SPRINTS, admin, config, and local. The SPRINTS database is selected, showing collections: comments, movies, sessions, theaters, and users. The main pane shows the MongoDB shell with the following command and result:

```
MongoDB Compass - Data Analytics/Shell
SPRINTS comments mongosh: Data Analytics movies mongosh: Data Analytics sessions theaters users
> use SPRINTS
< switched to db SPRINTS
> db.comments.countDocuments({email: {$regex: '/@gameofthron\.es$/i'}})
< 22841
SPRINTS >
```

Exploración: Antes de realizar el conteo, utilice `db.comments.findOne()` para verificar la estructura de los documentos.

Confirme que el campo que contiene la información de **contacto** es **email**.

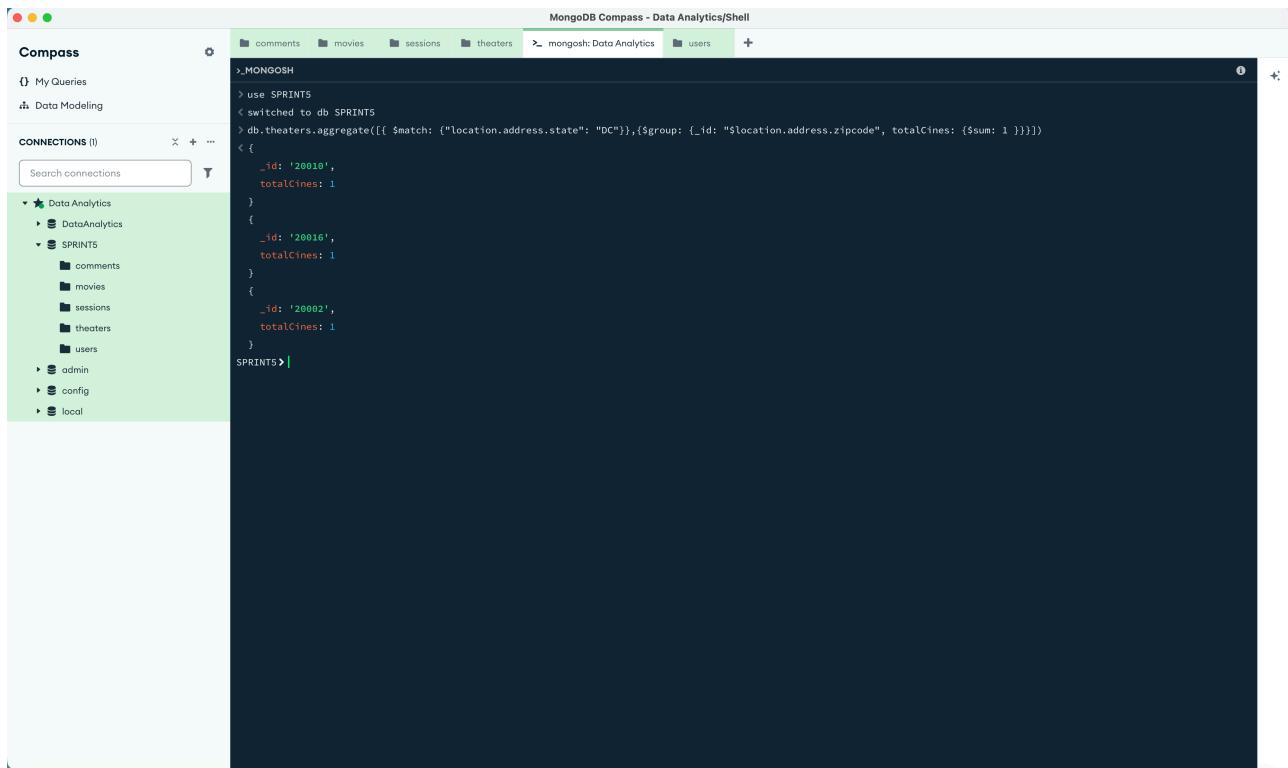
El ejercicio me pide filtrar por un dominio específico (GAMEOFTHRON.ES) y no por una dirección completa, así que uso **expresiones regulares** para **buscar coincidencias parciales dentro de la cadena de texto**.

Resolución: Mi query `db.comments.countDocuments({email: {$regex: '/@gameofthron\.es$/i'}})` hace lo siguiente:

- **db.comments.countDocuments(...):** Ejecuta el conteo de documentos en la colección de comentarios que cumplen el criterio.
- **\$regex:** Es el **operador** que **permite realizar búsquedas de patrones complejos**, en este caso lo usamos para **localizar el dominio** del correo.
- El símbolo **\$**: Dentro de la expresión regular, este **símbolo** es **importante** porque **indica** que el **patrón** debe **encontrarse al final** de la **cadena de texto**. (Esto asegura que filtramos por el dominio y no por texto que pudiera aparecer en el nombre del usuario).
- La opción **i**: Se **añade** para que la **búsqueda sea insensible a mayúsculas y minúsculas (case-insensitive)**, garantiza que se cuenten todos los registros sin importar cómo los han escrito.

Exercici 2

Quants cinemes hi ha en cada codi postal situats dins de l'estat Washington D. C. (DC)?



The screenshot shows the MongoDB Compass interface. On the left, the sidebar displays 'Data Analytics' and 'SPRINT5' collections, which contain 'comments', 'movies', 'sessions', 'theaters', and 'users'. The main panel is titled 'MongoDB Compass - Data Analytics/Shell' and contains the following command:

```
> .MONGOSH
> use SPRINTS
< switched to db SPRINTS
> db.theaters.aggregate([{"$match: {"location.address.state": "DC"}}, {"$group: {"_id: "$location.address.zipcode", "totalCines: {$sum: 1 }}}])
< [
    {
        "_id: '20010',
        "totalCines: 1
    }
    {
        "_id: '20016',
        "totalCines: 1
    }
    {
        "_id: '20082',
        "totalCines: 1
    }
]
```

Exploración: Para resolver este ejercicio, identifico que necesito procesar los datos en varias fases, por lo que opto por el **Aggregation Framework**.

Al revisar un documento con `findOne()`, veo que la **información geográfica** está **anidada dentro** de `location.address`.

Resolución: Mi query `db.theaters.aggregate([{"$match: {"location.address.state": "DC"}}, {"$group: {"_id: "$location.address.zipcode", "totalCines: {$sum: 1 }}}])`

es un **pipeline de agregación** con **dos etapas**:

- Un **pipeline** es un modelo donde los datos entran por un extremo, pasan por una serie de etapas de procesamiento y salen por el otro extremo transformados.
 - La **agregación** se refiere al proceso de procesar múltiples documentos para devolver un resultado combinado o calculado.
 - Un **Pipeline de agregación** es un marco de trabajo que permite ejecutar estas operaciones en etapas secuenciales. Los documentos de una colección pasan por una "tubería" donde cada etapa modifica los datos antes de pasarlos a la siguiente.
- **\$match:** Filtro los **documentos** para trabajar **únicamente** con los **cines situados** en el **estado de Washington D. C.** ("DC").
 - **\$group:** Agrupo los **resultados** usando el código postal (**zipcode**) como identificador único (**_id**).
 - **\$sum:** Para obtener el conteo, utilizo el acumulador `{ $sum: 1 }`, que añade una unidad por cada cine encontrado en ese código postal.
- Aunque mi query la explico en 3 partes, no son 3 etapas; El **\$sum**, que es el **conteo**, **no se considera** una "**etapa**" **independiente** en este caso, sino que es un **operador de acumulación** que vive **dentro** de la **etapa \$group**.

Conclusión: Hay **un cine por** cada **código postal** **dentro** del estado **Washington D. C. (DC)**.

NIVELL 3

Exercici 1

Troba totes les pel·lícules dirigides per John Landis amb una puntuació IMDb (Internet Movie Database) d'entre 7,5 i 8.

The screenshot shows the MongoDB Compass interface. On the left, there's a sidebar with 'Compass' at the top, followed by 'My Queries', 'Data Modeling', and 'CONNECTIONS (1)'. Under 'CONNECTIONS (1)', there's a 'Search connections' input field. Below these are sections for 'Data Analytics', 'SPRINT5', and 'admin'. The main area is titled 'MongoDB Compass - Data Analytics/Shell' and contains a terminal window. The terminal shows the following command and its output:

```
> use SPRINT5
< switched to db SPRINT5
> db.movies.findOne({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 so sanctimonious r",
    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 him.',
    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: {
            rating: 3.8,
            numReviews: 184490,
            meter: 89
        }
    },
    dvd: '1998-02-24T00:00:00Z',
}
```

Exploración: En este caso utilizo `db.movies.findOne()` para localizar la ubicación exacta de la métrica de puntuación.

Confirmo que la puntuación se encuentra en el subcampo **rating** dentro del objeto **imdb**.

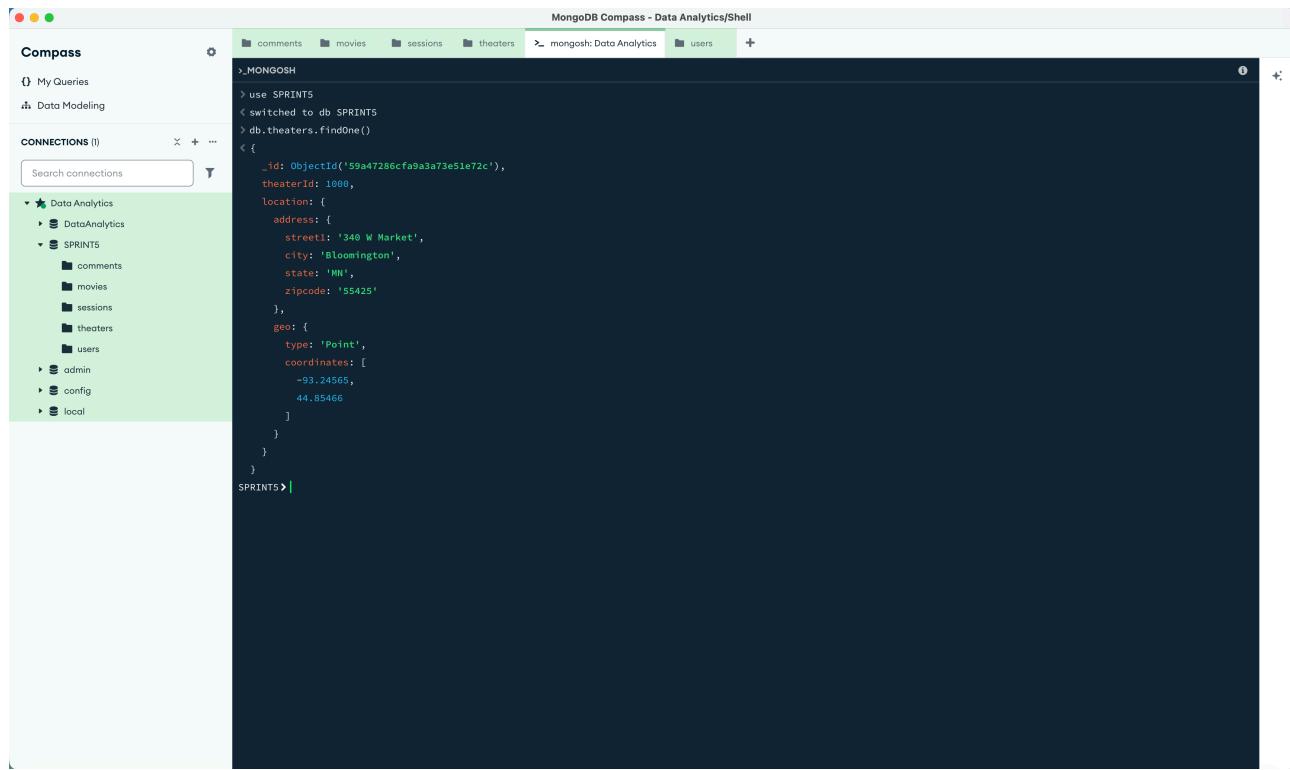
También verifico que el campo **directors** es un array (lista de valores), porque está en medio de [], lo que me permite buscar un nombre específico directamente.

Resolución: Mi query `db.movies.findOne({directors: "John Landis", "imdb.rating": {$gte: 7.5, $lte: 8}})` funciona así:

- **directors: "John Landis"**: Filtra las películas donde el director que solicita el ejercicio sea parte del equipo de dirección.
- **"imdb.rating": { \$gte: 7.5, \$lte: 8 }**: Establece un rango de puntuación. Al usar **\$gte** y **\$lte** dentro del mismo objeto, definimos un límite inferior (7.5) y uno superior (8) ambos incluidos.
- Usamos la notación de puntos y las comillas para entrar en el objeto **imdb** y llegar al valor de **rating**.

Exercici 2

Mostra en un mapa la ubicació de tots els teatres de la base de dades.



The screenshot shows the MongoDB Compass interface. On the left, the sidebar displays connections and collections. A connection named 'SPRINT5' is selected, showing its collections: comments, movies, sessions, theaters, users, admin, config, and local. The main pane shows a shell session titled 'MONGOSH'. The command `db.theaters.findOne()` is run, and the response is displayed as a JSON object representing a theater document. The document includes fields like _id, theaterId, location (with address and geo coordinates), and geoJSON coordinates (-93.24565, 44.85466).

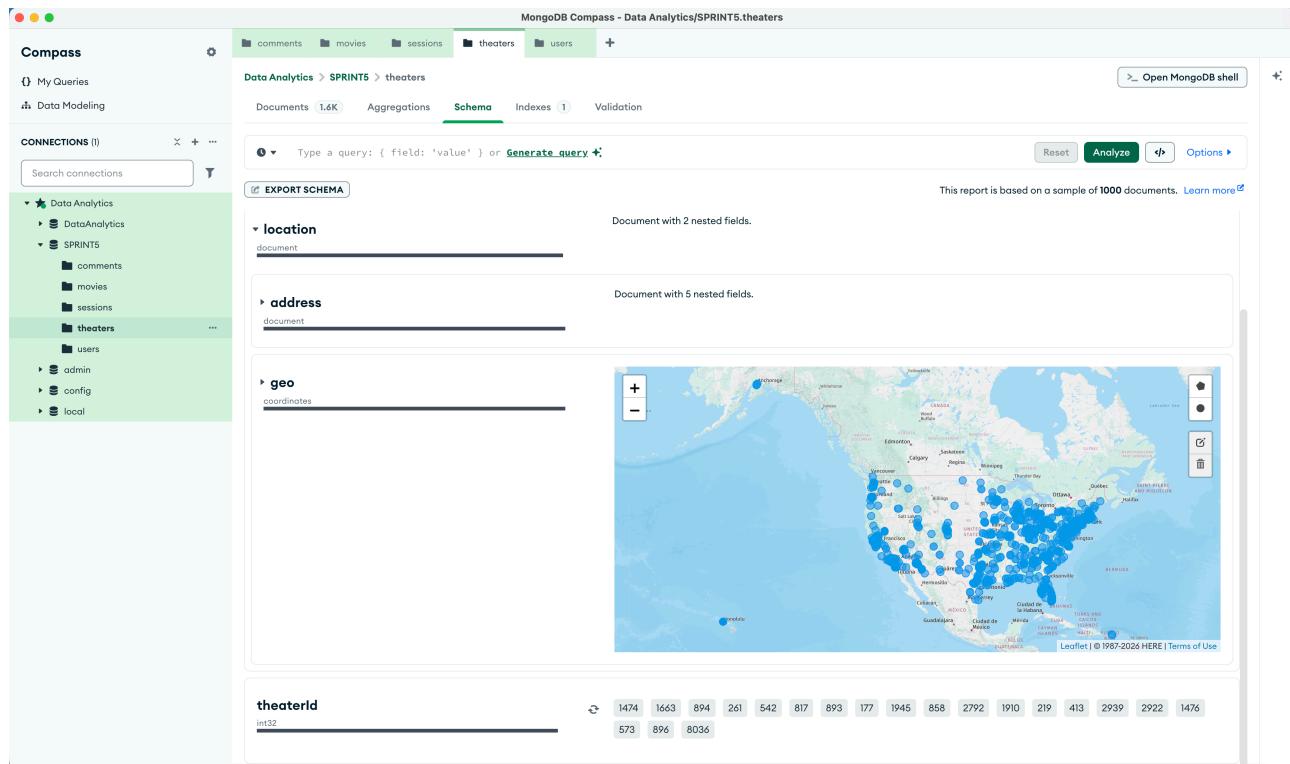
```
> use SPRINT5
< switched to db SPRINT5
> db.theaters.findOne()
< {
    "_id": ObjectId("59a47286cf9a3a73e51e72c"),
    "theaterId": 1000,
    "location": {
        "address": {
            "street": "340 W Market",
            "city": "Bloomington",
            "state": "MN",
            "zipcode": "55425"
        },
        "geo": {
            "type": "Point",
            "coordinates": [
                -93.24565,
                44.85466
            ]
        }
    }
}
SPRINT5>
```

Exploración: Antes de empezar con la visualización, utilice el comando `db.theaters.findOne()` para ver el esquema de datos.

Esta fase de exploración me permite confirmar dos aspectos clave:

- **Identificación del formato GeoJSON:** Verifico que la ubicación no es solo texto, sino que está estructurada bajo el estándar **GeoJSON** dentro del campo **location.geo**.
Esto es importante, ya que las herramientas de mapas necesitan identificar el tipo "**Point**" y las **coordenadas [longitud, latitud]** para funcionar.
- **Verificación de la jerarquía:** Al observar el resultado de `findOne()`, identifico que las coordenadas se encuentran anidadas en **location.geo.coordinates**.

Esta ruta es la que permite a **MongoDB Compass** interpretar los datos y transformarlos en puntos geográficos sobre un mapa.



Resolución: Realizo la visualización utilizando la herramienta **MongoDB Compass**:

- Utilizo la funcionalidad **Analyze Schema** sobre la colección **theaters**.
- Al detectar el campo **geoespacial location.geo.coordinates**, la interfaz genera automáticamente una **vista de mapa**.
- El resultado me muestra la **distribución geográfica** de todos los **teatros almacenados** y me permite interactuar con los puntos de datos de forma visual sobre un mapa global.