

## SPRINT 5 (MONGODB) MARINE FERNANDEZ

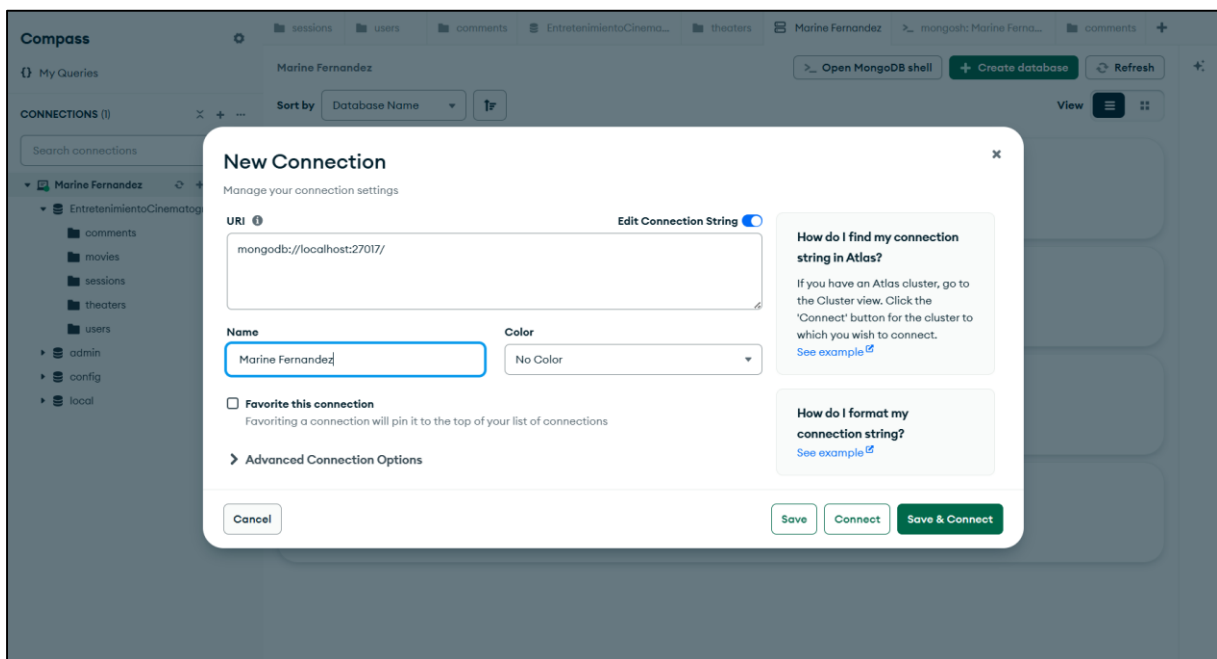
Trabajaremos con una base de datos que contiene colecciones relacionadas con una aplicación de entretenimiento cinematográfico:

- users: Almacena información de usuarios/se, incluyendo nombres, emails y contraseñas cifradas.
- theaters: Contiene datos de cines, como ID, ubicación (dirección y coordenadas geográficas).
- sessions: 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/se sobre películas, con información del autor/a del comentario, ID de la película, texto del comentario y la fecha.

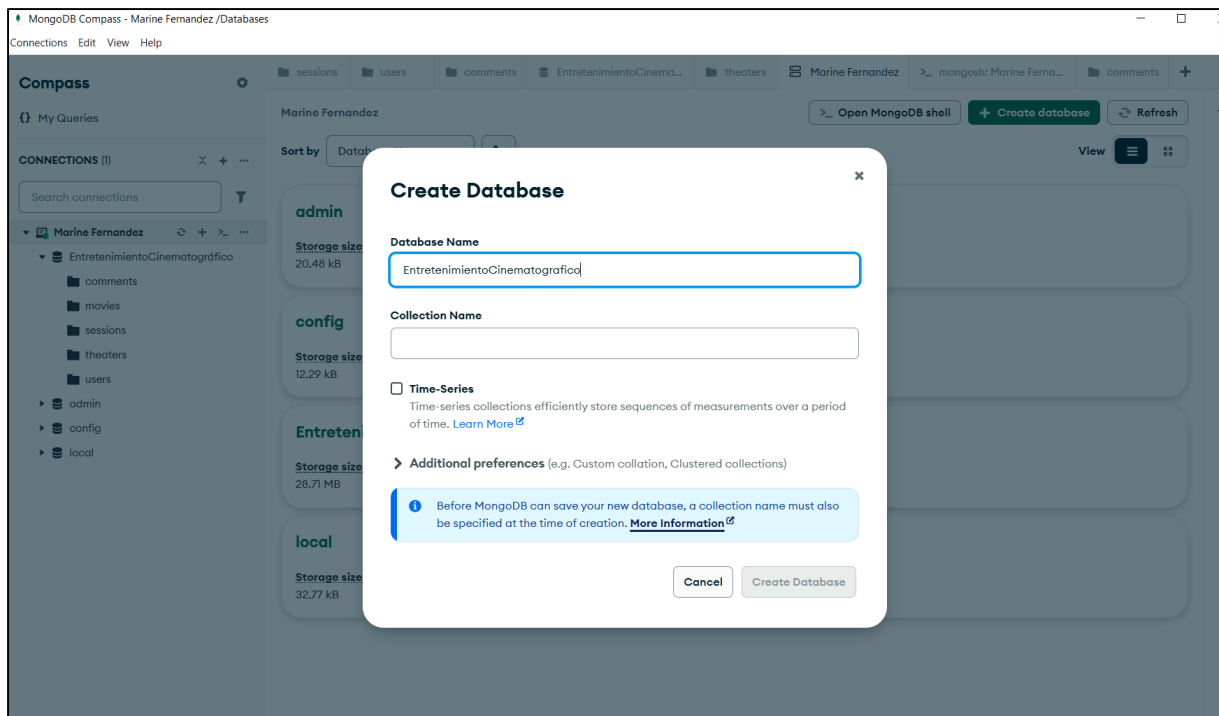
Llevarás a cabo algunas consultas que te pide el cliente/a, el cual está midiendo si serás capaz o no de hacerte cargo de la parte analítica del proyecto vinculado con su base de datos.

### NIVEL 1

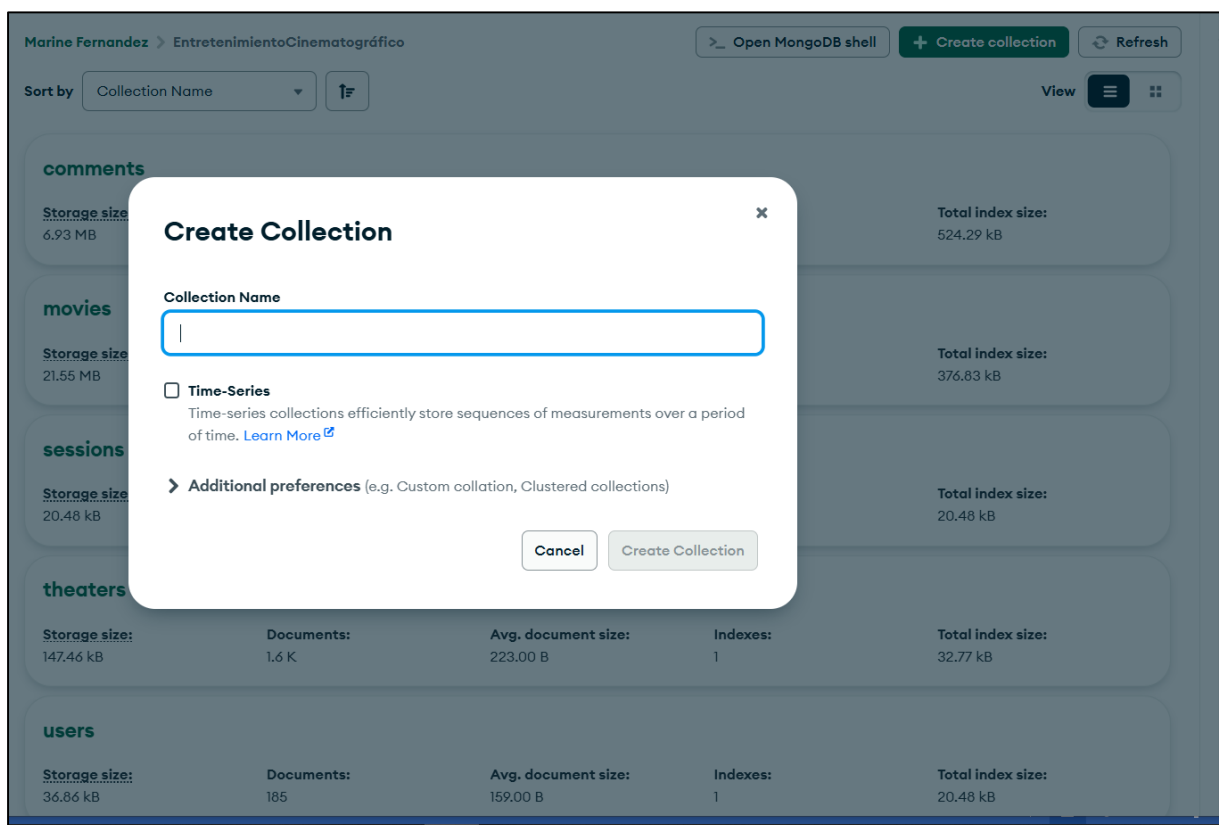
Creas una base de datos con MongoDB utilizando como colecciones los archivos adjuntos.



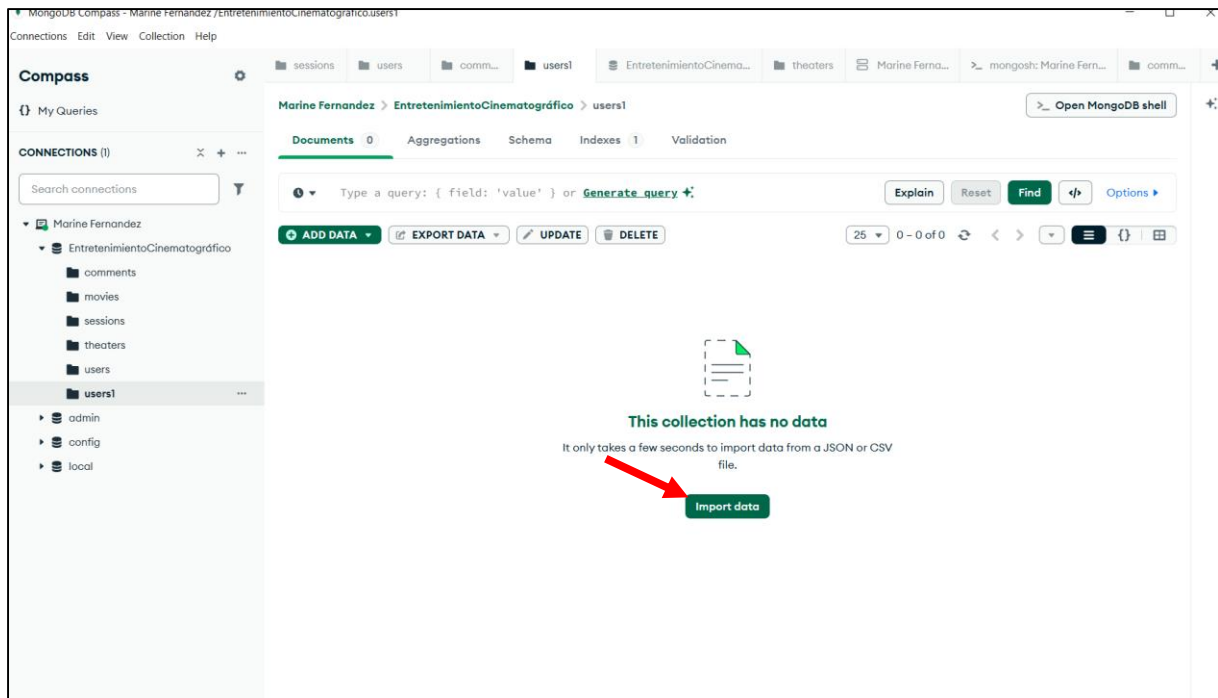
Primero creamos una Connection en la cual crearemos la base de datos.



Creamos la base de datos y la nombramos EntretenimientoCinematografico.



Posteriormente creamos las diferentes collecciones e importamos los datos a formato JSON.



## EJERCICIO 1

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

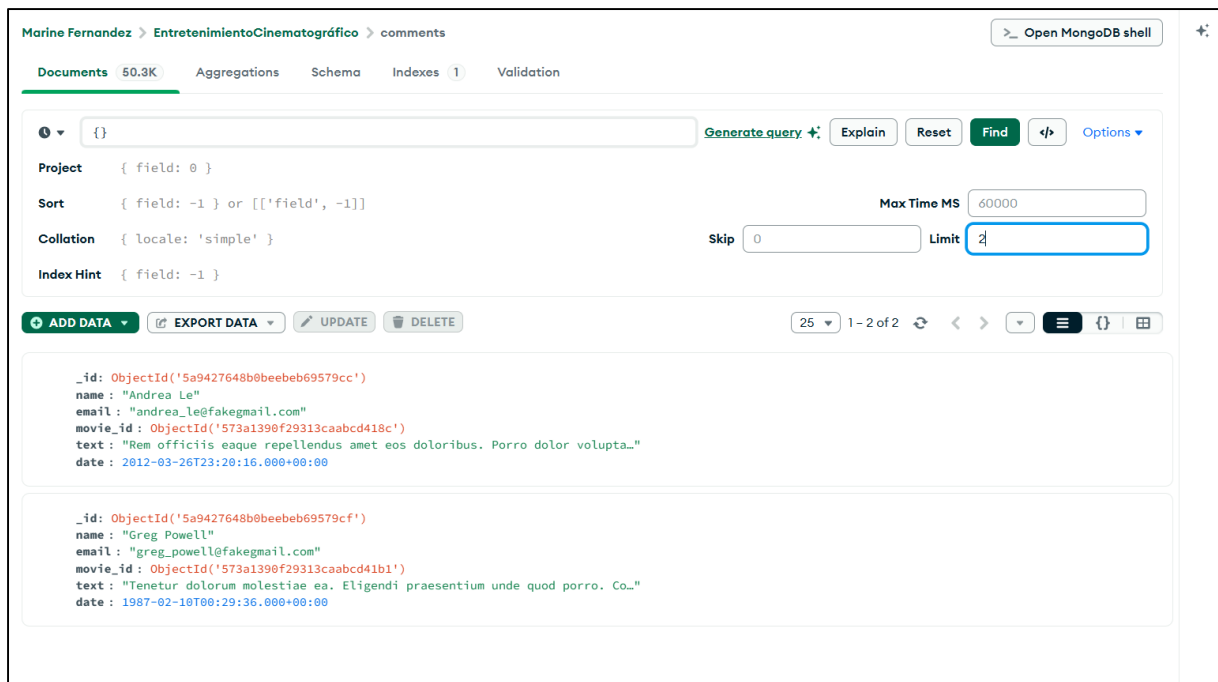
### MongoShell:

```
>_MONGOSH
> use EntretenimientoCinematográfico
< switched to db EntretenimientoCinematográfico
> db["comments"].find().limit(2).
✖ SyntaxError: Unexpected token (1:31)

[0m[31m[1m[2m[39m[90m 1 | [39m db[32m"comments"[39m][33m.[39mfind()[33m.[39mlimit([35m2[39m)[33m.[39m
[90m [39m [31m[1m[2m[39m[0m

> db["comments"].find().limit(2)
< {
  _id: ObjectId('5a9427648b0beebe69579cc'),
  name: 'Andrea Le',
  email: 'andrea_le@fakegmail.com',
  movie_id: ObjectId('573a1390f29313caabcd418c'),
  text: 'Rem officiiis eaque repellendus amet eos doloribus. Porro dolor voluptatum voluptates neque culpa molestias. Voluptate unde nulla temporibu
  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. Dolorum nihil ve
  date: 1987-02-10T00:29:36.000Z
}
```

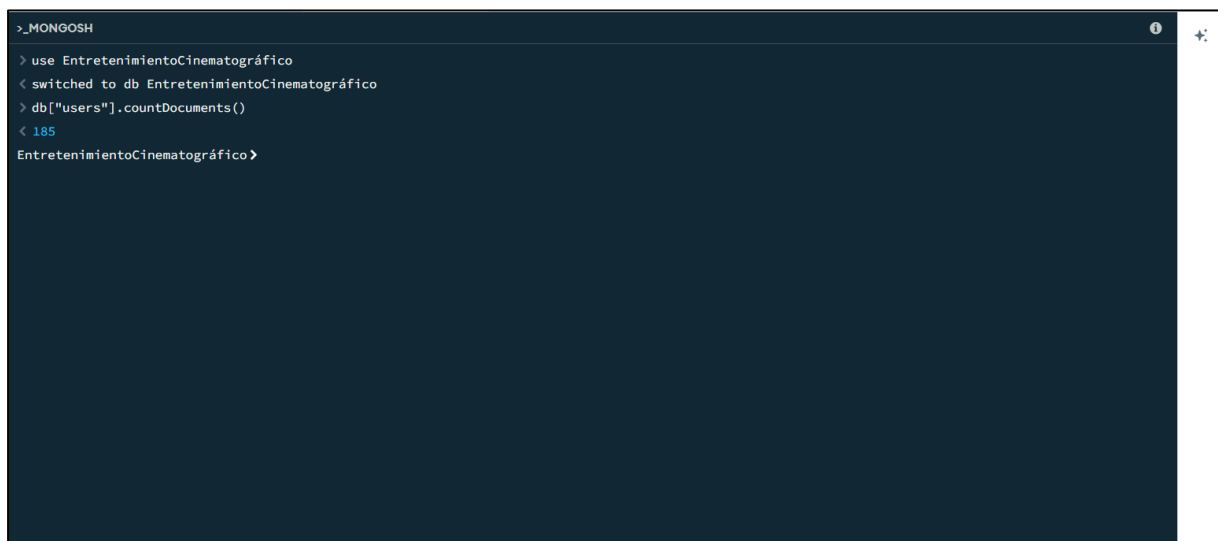
Aplicamos un find() y luego delimitamos con un limit(2).



De la misma manera en compass, no ponemos query y rellenamos el campo limit con un 2.

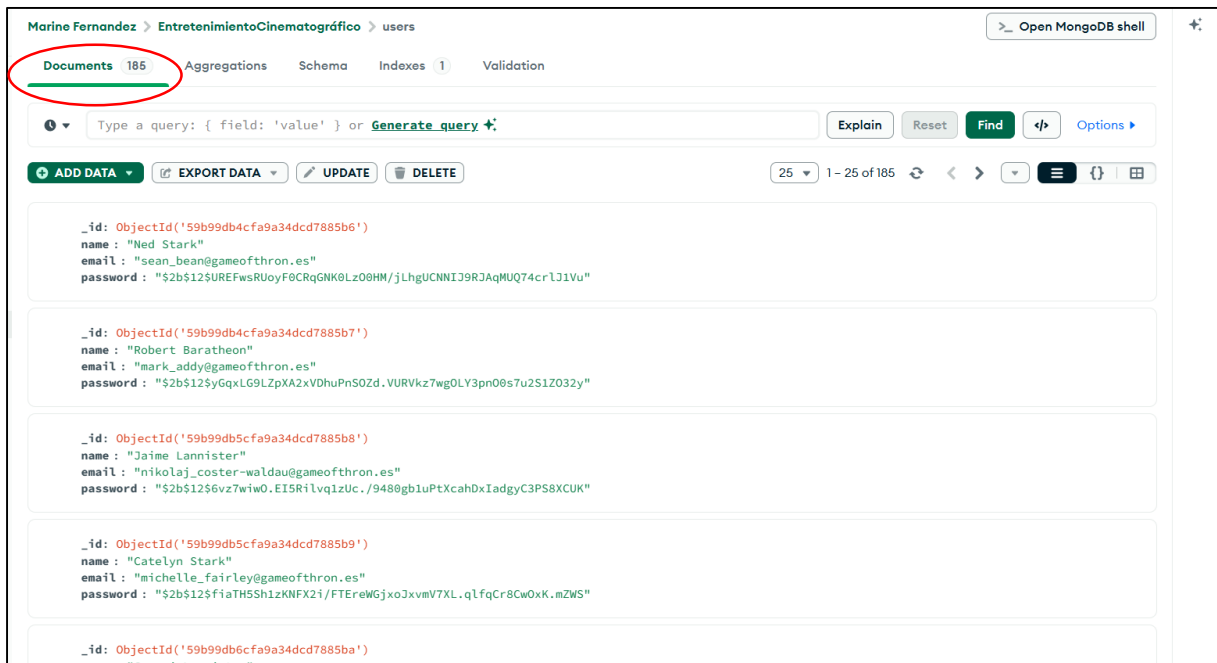
- ¿Cuántos usuarios tenemos registrados?

MongoShell:



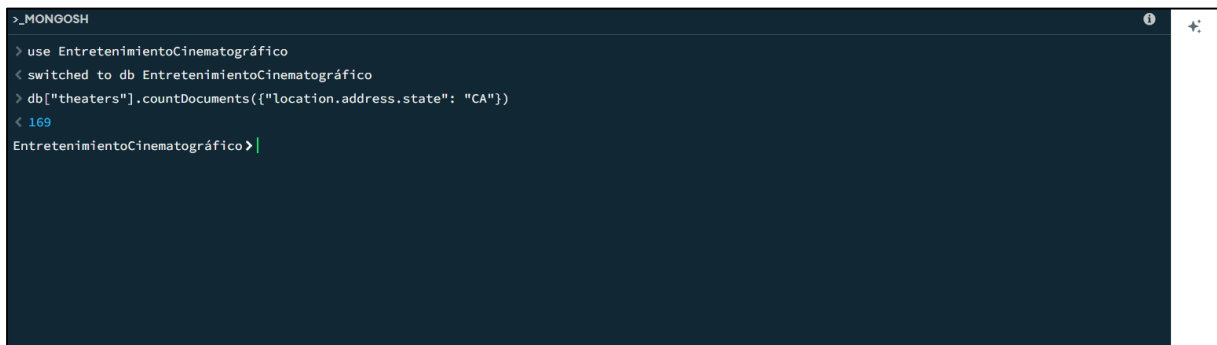
Aplicamos countDocuments() sin añadir filtros.

## Compass:



- ¿Cuántos cines existen en el estado de California?

## Mongo shell :



En este caso aplicamos countDocuments() y filtramos por el estado de California que aparece como CA.

## Aggregations pipeline en compass:

The screenshot shows the MongoDB Compass interface for a database named 'EntretenimientoCinematográfico' and a collection named 'theaters'. The 'Aggregations' tab is active, showing a pipeline with two stages:

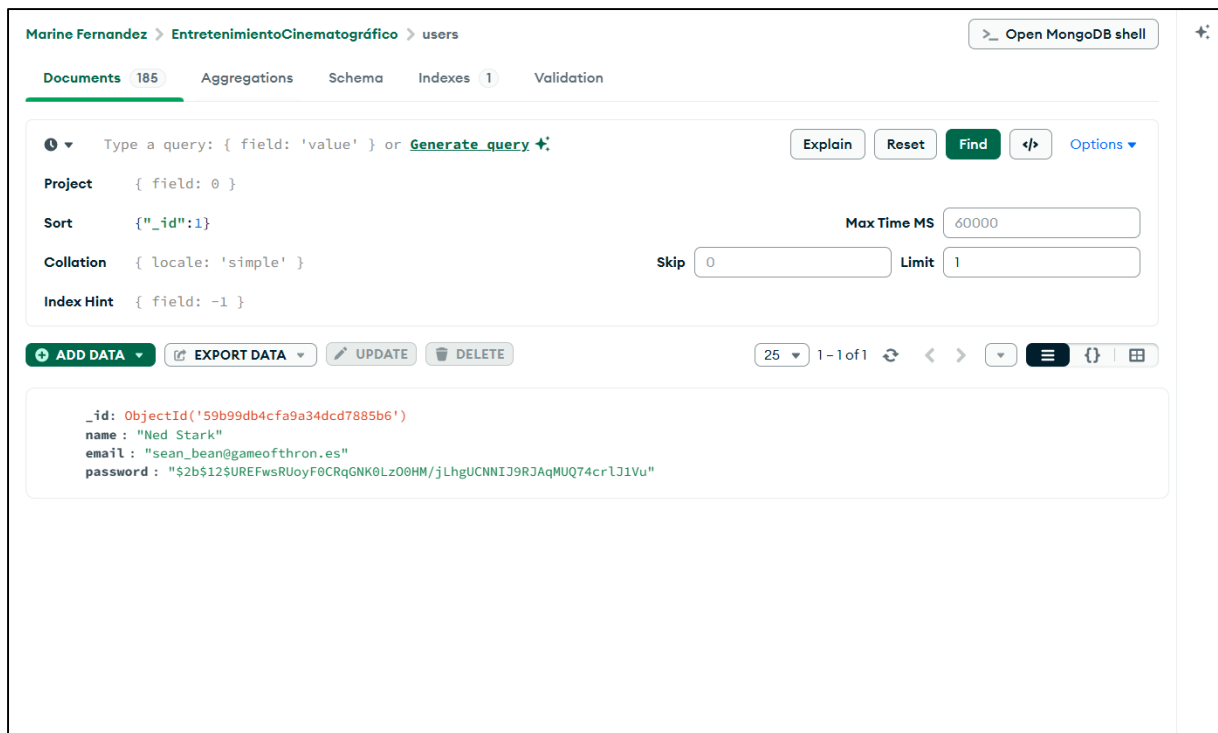
- Stage 1: \$match** with the query: `{ "location.address.state": "CA" }`. The output preview shows 10 documents, including two examples: `{ "_id": ObjectId('59a47286cfa9a3a73e51e72e'), "theaterId": 1008, "location": Object }` and `{ "_id": ObjectId('59a47286cfa9a3a73e51e734'), "theaterId": 1009, "location": Object }`.
- Stage 2: \$count** with the query: `/**  
 * Provide the field name for the count.  
 */  
'cantidad_cines_california'`. The output preview shows a single document: `{ "cantidad_cines_california": 169 }`.

Con una aggregations pipeline filtramos primero por estado con un \$match y posteriormente realizamos un \$count.

- ¿Cuál fue el primer usuario en registrarse?

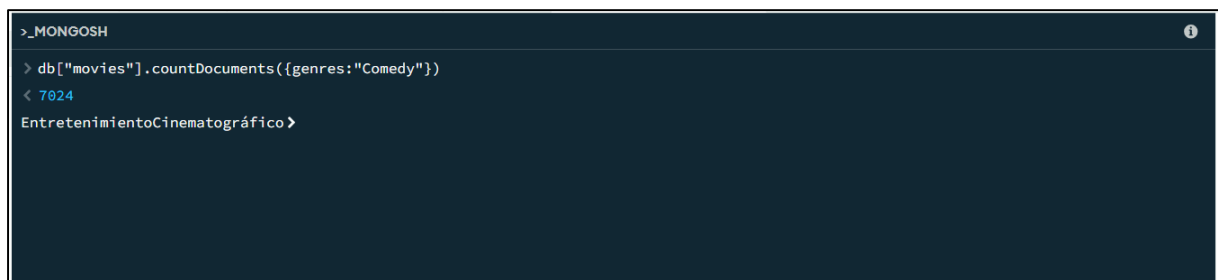
```
>_MONGOSH
> db["users"].find().sort({'_id':1}).limit(1)
< {
  _id: ObjectId('59b99db4cfa9a34dcd7885b6'),
  name: 'Ned Stark',
  email: 'sean_bean@gameofthron.es',
  password: '$2b$12$UREFwsRUoyF0CRqGNK0Lz00HM/jLhgUCNNI39RJAqMUQ74cr1J1Vu'
}
EntretenimientoCinematográfico>
```

En este caso aplicamos un find() sin filtrar y posteriormente ordenamos los usuarios con \$sort de forma ascendente (por eso le ponemos 1 y no -1), ya que en MongoDB si ordenamos por \_id se hace automaticamente por fecha de registro. Posteriormente aplicamos un limit (1) para tener al primero.



- ¿Cuántas películas de comedia existen en nuestra base de datos?

Con MongoShell:



Aplicamos un `countDocuments()` filtrando por el género de las películas.

## Compass :

The screenshot shows the MongoDB Compass interface. At the top, the breadcrumb navigation is "Marine Fernandez > EntretenimientoCinematográfico > movies". Below this, there are tabs for "Documents", "Aggregations", "Schema", "Indexes", and "Validation". The "Documents" tab is active, showing a query: `{ "genres": "Comedy" }`. Below the query, there are fields for "Project", "Sort", "Collation", and "Index Hint". To the right, there are buttons for "Generate query", "Explain", "Reset", "Find", and "Options". Below these buttons, there are input fields for "Max Time MS" (set to 60000), "Skip" (set to 0), and "Limit" (set to 0). At the bottom, there are buttons for "ADD DATA", "EXPORT DATA", "UPDATE", and "DELETE". The main area displays a JSON document for a movie titled "Gertie the Dinosaur".

## Aggregations pipeline en Compass :

The screenshot shows the MongoDB Compass interface with the "Aggregations" tab active. At the top, the breadcrumb navigation is "Marine Fernandez > EntretenimientoCinematográfico > movies". Below this, there are tabs for "Documents", "Aggregations", "Schema", "Indexes", and "Validation". The "Aggregations" tab is active, showing an aggregation pipeline with two stages: "Stage 1" and "Stage 2". Stage 1 is named "\$match" and contains the query: `{ "genres": "Comedy" }`. Stage 2 is named "\$count" and contains the field: `"Cantidad_peliculas_comedia"`. To the right of the stages, there are buttons for "Generate aggregation", "Explain", "Export", "Run", and "Options". Below these buttons, there are buttons for "PREVIEW", "STAGES", "TEXT", and "WIZARD". The main area displays the output preview for each stage. Stage 1 shows a sample of 10 documents, and Stage 2 shows a sample of 1 document with the count: `Cantidad_peliculas_comedia : 7024`.

Con un aggregations pipeline procedemos a filtrar con \$match primero y luego aplicamos un \$count.



## EJERCICIO 2

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

MongoShell:

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

```
>_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 tri
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/MV5BYWY3ODE5ZWVjYmYi00NjA4LTk4ZWYtMzBhZDE5MjY0YTUxXkE5XkFqcGdeQXVyNzI4MDMyMTU@._V1_SY1000_SX677_AL
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,
}
```

Aquí usamos un `find()` para filtrar las películas que sean del 1932 y posteriormente que cumplan unas de las dos condiciones que nos dan en el enunciado usando el `$or`.

## Compass:

Marine Fernandez > EntretenimientoCinematográfico > movies Open MongoDB shell

Documents 23.5K Aggregations Schema Indexes 1 Validation

**Query:** `{ "year": 1932, "$or": [ { "genres": "Drama" }, { "languages": "French" } ] }` Generate query Explain Reset Find Options

**Project:** `{ field: 0 }`

**Sort:** `{ field: -1 } or [['field', -1]]` Max Time MS: 60000

**Collation:** `{ locale: 'simple' }` Skip: 0 Limit: 0

**Index Hint:** `{ field: -1 }`

ADD DATA EXPORT DATA UPDATE DELETE 25 1 - 18 of 18 Navigation icons

```
{
  "_id": ObjectId("573a1392f29313caabcd99a3"),
  "plot": "Junta is hated by the people in the village where she lives, especial...",
  "genres": Array(3),
  "runtime": 85,
  "cast": Array(4),
  "poster": "https://m.media-amazon.com/images/M/MV5BNTQ1NTMzMtQtODIyYS00NTAxLWE1NT...",
  "title": "The Blue Light",
  "fullplot": "Junta is hated by the people in the village where she lives, especial...",
  "languages": Array(2),
  "released": 1934-05-08T00:00:00.000+00:00,
  "directors": Array(2),
  "writers": Array(2),
  "awards": Object,
  "lastupdated": "2015-08-03 01:02:12.350000000",
  "year": 1932,
  "imdb": Object,
  "countries": Array(1),
  "type": "movie",
  "tomatoes": Object
}
```

## EJERCICIO 3

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

MongoShell :

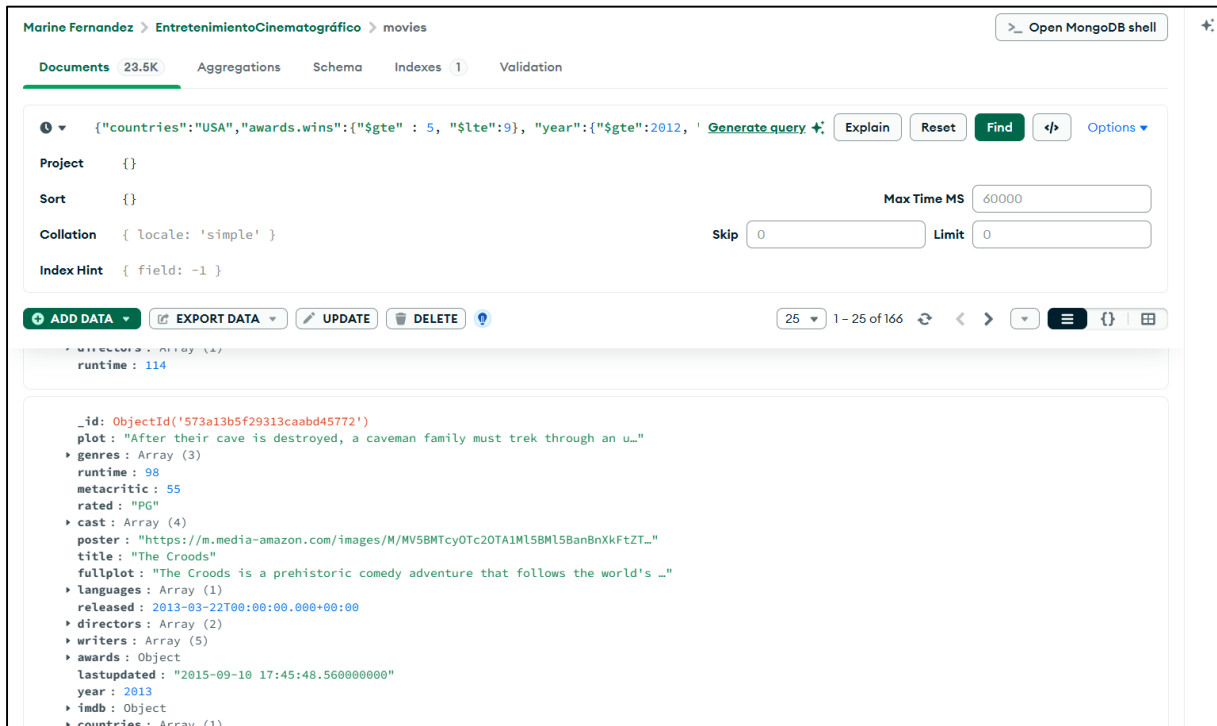
Query: `db["movies"].find({"countries":"USA","awards.wins":{"$gte" : 5, "$lte":9}, "year":{"$gte":2012, "$lte":2014}})`

```
>_MONGOSH
> 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
  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 tur
  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)'
```

Aplicamos un find donde filtramos por distintos fields, determinamos rangos para los fields premios y año gracias a \$gte (más o igual a) y \$lte(menos o igual a)

## Compass

Query:{"countries":"USA",awards.wins:{"\$gte" : 5, "\$lte":9}, year:{"\$gte":2012, "\$lte":2014}}



## NIVEL 2

### EJERCICIO 1

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

### MongoShell:



Aquí hemos usado find() para aplicar el filtro solicitado en el enunciado y hemos usado \$regex para buscar el patrón "GAMEOFTHRON.ES" y le hemos añadido la \$option i para que tome en cuenta tanto esos dominios escritos en mayúscula como en minúscula.

## Compass:

Query: `{email: {"$regex": "gameofthron.es", "$options": "i"}}`

The screenshot shows the MongoDB Compass interface. The top navigation bar includes 'Documents' (50.3K), 'Aggregations', 'Schema', 'Indexes' (1), and 'Validation'. The main query bar contains the query `{email: {"$regex": "gameofthron.es", "$options": "i"}}` and buttons for 'Generate query', 'Explain', 'Reset', 'Find', and 'Options'. Below the query bar, the 'Project' field is set to `{ field: 0 }`, 'Sort' is `{ field: -1 } or [['field', -1]]`, 'Collation' is `{ locale: 'simple' }`, and 'Index Hint' is `{ field: -1 }`. The 'Skip' field is 0 and 'Limit' is 0. The 'Max Time MS' is 60000. The document list shows three documents. The first document is highlighted and its details are shown below: `email: "aidan_gillen@gameofthron.es", movie_id: ObjectId('573a1390f29313caabdc4218'), text: "Quo deserunt ipsam ipsum. Tenetur eos nemo nam sint praesentium minus ...", date: 2001-07-13T19:25:09.000+00:00`. The second document is: `_id: ObjectId('5a9427648b0beebe69579db'), name: "Olly", email: "brenock_o'connor@gameofthron.es", movie_id: ObjectId('573a1390f29313caabdc413b'), text: "Perspiciatis sit pariatur quas. Perferendis officia harum ipsum delenit...", date: 2005-01-04T13:49:05.000+00:00`. The third document is: `_id: ObjectId('5a9427648b0beebe69579ee'), name: "Theon Greyjoy", email: "alfie_allen@gameofthron.es", movie_id: ObjectId('573a1390f29313caabdc437c'), text: "Dicta asperiores necessitatibus corporis. Quidem fugiat eius animi fug...", date: 2000-12-06T07:26:29.000+00:00`. A red circle highlights the pagination controls showing '25' and '1 - 25 of 22841'.

## Compass con aggregations pipeline:

The screenshot shows the MongoDB Compass interface with the 'Aggregations' tab selected. The top navigation bar includes 'Documents' (50.3K), 'Aggregations', 'Schema', 'Indexes' (1), and 'Validation'. The main aggregation bar contains the aggregation pipeline `[{"$match": {"email": {"$regex": "gameofthron.es", "$options": "i"}}}]` and buttons for 'Generate aggregation', 'Explain', 'Export', 'Run', and 'Options'. Below the aggregation bar, the 'Stage 1' dropdown is set to '\$match' and the 'Stage 2' dropdown is set to '\$count'. The 'Output preview after \$match stage (Sample of 10 documents)' shows three documents: `_id: ObjectId('5a9427648b0beebe69579db'), name: "Olly", email: "brenock_o'connor@gameofthron.es", movie_id: ObjectId('573a1390f29313caabdc413b'), text: "Perspiciatis sit pariatur quas. Perferendis officia harum ipsum delenit...", date: 2005-01-04T13:49:05.000+00:00`, `_id: ObjectId('5a9427648b0beebe69579de'), name: "Petyr Baelish", email: "aidan_gillen@gameofthron.es", movie_id: ObjectId('573a1390f29313caabdc41b'), text: "Rem itaque ad sit rem voluptatibus. Ad fugiat maxime illum optio iure ...", date: 1998-08-22T11:45:03.000+00:00`, and `_id: ObjectId('5a9427648b0beebe69579df'), name: "Theon Greyjoy", email: "alfie_allen@gameofthron.es", movie_id: ObjectId('573a1390f29313caabdc437c'), text: "Dicta asperiores necessitatibus corporis. Quidem fugiat eius animi fug...", date: 2000-12-06T07:26:29.000+00:00`. The 'Output preview after \$count stage (Sample of 1 document)' shows a single document: `Cantidad_usuarios_gameofthron: 22841`.

## EJERCICIO 2:

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

MongoShell :

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

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

En este caso creamos una aggregations pipeline, donde primero filtramos los documentos por el estado DC usando \$match, posteriormente usamos \$group para agrupar los cines por el código postal y posteriormente con \$sum:1 hacemos el recuento de cines por código postal, sirve de contador.

Compass con Agregation pipeline:

Query :

```
[
  {
    $match:
      /**
       * query: The query in MQL.
       */
      {
        "location.address.state": "DC"
      }
  },
  {
    $group:
      /**
       * _id: The id of the group.
       * fieldN: The first field name.
       */
      {
        _id: "$location.address.zipcode",
        TotalCines: {
          $sum: 1
        }
      }
  }
]
```

```
}  
]
```

Marine Fernandez > EntretenimientoCinematográfico > theaters Open MongoDB shell

Documents 1.6K Aggregations Schema Indexes 1 Validation

\$match \$group Generate aggregation Explain Export Run Options

NIVEL 2 EJERC... - modified SAVE CREATE NEW EXPORT TO LANGUAGE PREVIEW STAGES TEXT WIZARD

**Stage 1** \$match Output preview after \$match stage (Sample of 3 documents)

```
1 /**  
2  * query: The query in MQL.  
3  */  
4  {  
5    "location.address.state":"DC"  
6  }
```

**Stage 2** \$group Output preview after \$group stage (Sample of 3 documents)

```
1 /**  
2  * _id: The id of the group.  
3  * fieldN: The first field name.  
4  */  
5  {  
6    _id: "$location.address.zipcode",  
7    TotalCines: {$sum:1  
8  }  
9  }
```

### 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.

MongoShell:

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

```
>_MONGOSH
> 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, 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
  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',
  ...
}
```

Aquí aplicamos `find()` para filtrar por director y por puntuación IMDb dentro del rango del enunciado usando `$gte` y `$lte`.



Compass:

Query: {"directors":"John Landis", "imdb.rating": {"\$gte": 7.5, "\$lte": 8}}

Marine Fernandez > EntretenimientoCinematográfico > movies

Documents 23.5K Aggregations Schema Indexes 1 Validation

ectors:"John Landis", "imdb.rating": {"\$gte": 7.5, "\$lte": 8}} [Generate query](#) [Explain](#) [Reset](#) [Find](#) [Options](#)

Project { field: 0 }

Sort { field: -1 } or [['field', -1]] Max Time MS 60000

Collation { locale: 'simple' } Skip 0 Limit 0

Index Hint { field: -1 }

[ADD DATA](#) [EXPORT DATA](#) [UPDATE](#) [DELETE](#) [?](#)

25 1 - 4 of 4

```
{
  "_id": ObjectId('573a1397f29313caabce6d94'),
  "fullplot": "Faber College has one frat house so disreputable it will take anyone. ...",
  "imdb": {
    "year": 1978,
    "plot": "At a 1962 college, Dean Vernon Wormer is determined to expel the entire...",
    "genres": [
      "R"
    ],
    "rated": "R",
    "metacritic": 82,
    "title": "Animal House",
    "lastupdated": "2015-09-13 00:02:47.803000000",
    "languages": [
      "English"
    ],
    "writers": [
      "Michael Butler",
      "John Landis"
    ],
    "type": "movie",
    "tomatoes": {
      "poster": "https://m.media-amazon.com/images/M/MV5BM2M2ZDA4MTYtOGRjMi00OTg5LWI1ZT...",
      "num_mflix_comments": 1,
      "released": "1978-07-28T00:00:00.000+00:00",
      "awards": {
        "countries": [
          "USA"
        ],
        "cast": [
          "John Belushi",
          "John Landis",
          "John Landis",
          "John Landis"
        ],
        "directors": [
          "John Landis"
        ]
      }
    }
  }
}
```

## EJERCICIO 2

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

Marine Fernandez > EntretenimientoCinematográfico > theaters

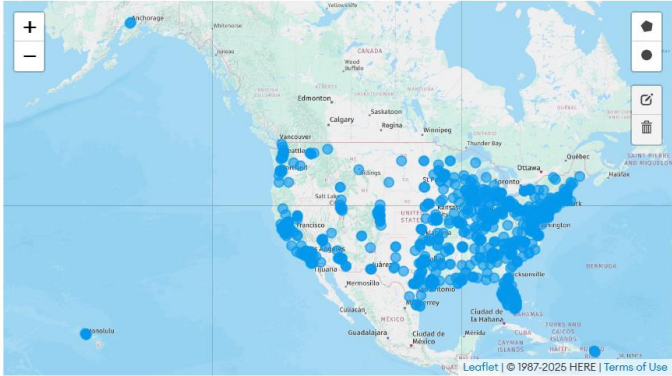
Documents 1.6K Aggregations Schema Indexes 1 Validation

Type a query: { field: 'value' } or [Generate query](#) [Reset](#) [Analyze](#) [Options](#)

[EXPORT SCHEMA](#) This report is based on a sample of 1000 documents. [Learn more](#)

**address** Document with 5 nested fields.

**geo** coordinates



Leaflet | © 1987-2025 HERE | Terms of Use

Para obtener ese mapa hemos ido a la pestaña schema de la collection theaters, hemos entrado en la parte location y ha aparecido el mapa.