

Proyecto Final NoSQL: Ejercicios MongoDB

Máster Big Data, Data Science & Inteligencia Artificial

Proyecto realizado por: Alejandro Borrego Megías

Fecha: 17 de Noviembre de 2023

Correo: alejbormeg@gmail.com

Índice

- [Introducción a MongoDB](#)
 - [Prerrequisitos](#)
 - [Clientes Utilizados](#)
 - [Ejercicios](#)
-

Introducción a MongoDB

MongoDB es un sistema de gestión de bases de datos NoSQL que se ha vuelto ampliamente popular en el ámbito del desarrollo de aplicaciones modernas. A diferencia de las bases de datos relacionales tradicionales, MongoDB utiliza un modelo de datos flexible basado en documentos JSON, lo que permite una escalabilidad horizontal eficiente y una fácil adaptación a los cambios en los esquemas de datos.

Prerrequisitos

Antes de comenzar con este proyecto, es necesario instalar MongoDB como un servicio de red en su máquina local con sistema operativo Windows. A continuación, se proporcionan los pasos para realizar la instalación:

1. **Descargar MongoDB:** Acceda al [sitio web oficial de MongoDB](#) y descargue la versión Community de MongoDB para Windows.
2. **Instalación:** Siga las instrucciones de instalación proporcionadas durante el proceso de instalación. Asegúrese de seleccionar la opción para instalar MongoDB como un servicio de red.
3. **Configuración:** Después de la instalación, es posible que necesite configurar algunas opciones según sus preferencias. Asegúrese de que el servicio de MongoDB esté iniciado y en ejecución.

Clientes Utilizados

En este proyecto, se utilizarán dos clientes populares para interactuar con la base de datos MongoDB: Compass y NoSQLBoosterForMongoDB.

1. **MongoDB Compass:** Compass es una interfaz gráfica de usuario que facilita la exploración y manipulación de datos en MongoDB. Proporciona herramientas visuales intuitivas para realizar consultas, analizar el rendimiento y diseñar esquemas de datos.

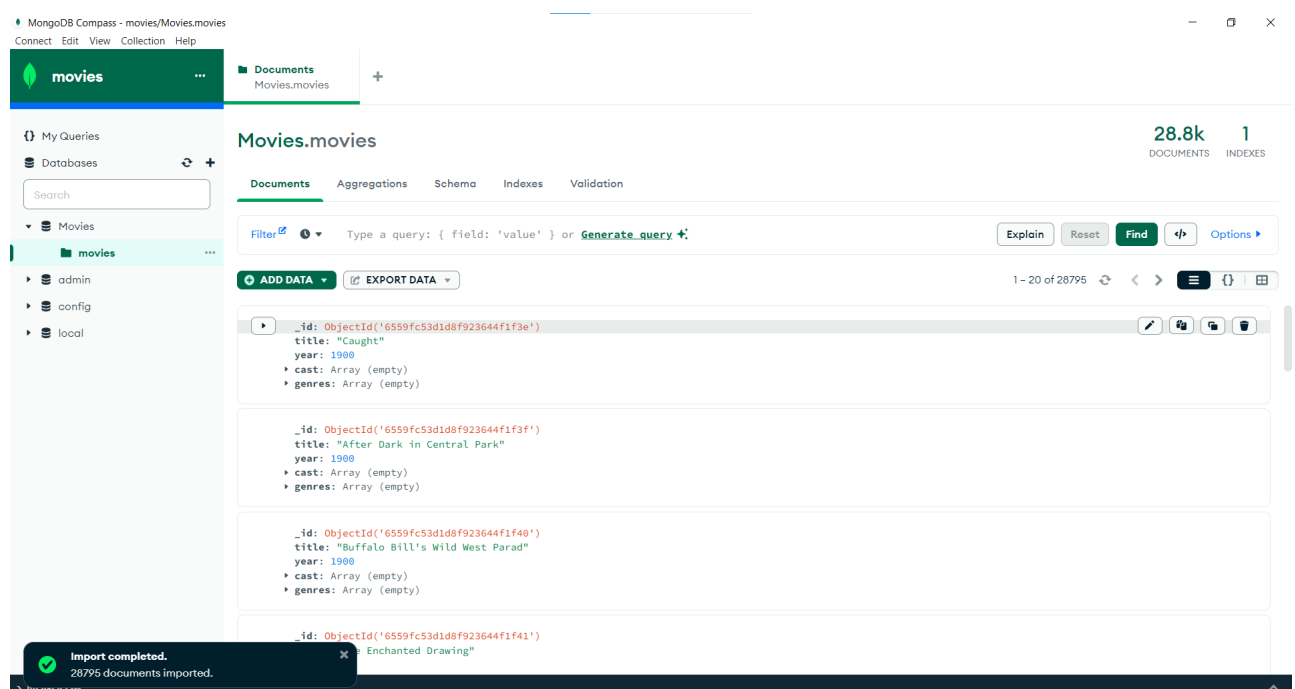
2. **NoSQLBoosterForMongoDB:** NoSQLBooster es otra herramienta poderosa para trabajar con MongoDB. Ofrece características avanzadas como autocompletar consultas, un editor de consultas integrado y una interfaz de usuario amigable que agiliza el desarrollo y la administración de bases de datos MongoDB.

Ejercicios

A continuación, se presentan los ejercicios que explorarán las capacidades de MongoDB en el contexto de NoSQL. Para el primero usaremos el Cliente *MongoDB Compass* y para el resto de ejercicios el cliente *NoSQLBoosterForMongoDB* por tener una interfaz más amigable para las consultas.

0. Realizar la importación del json en una colección llamada "movies"

El Dataset usado para este ejercicio se encuentra en la carpeta `./Database` en formato `.json`. Para importar el *Dataset* usamos el cliente MongoDB Compass:



Como podemos observar se ha importado correctamente.

1. Analizar con find la colección.

La query de este ejercicio es la siguiente:

```
// Ejercicio 1
db.movies.find()
```

Con ella podemos ver como en total se han insertado 28795 Documentos en la colección *movies*. Además, vemos como el JSON que representa dichos documentos presenta la siguiente estructura (mostramos el primer ejemplo de la colección):

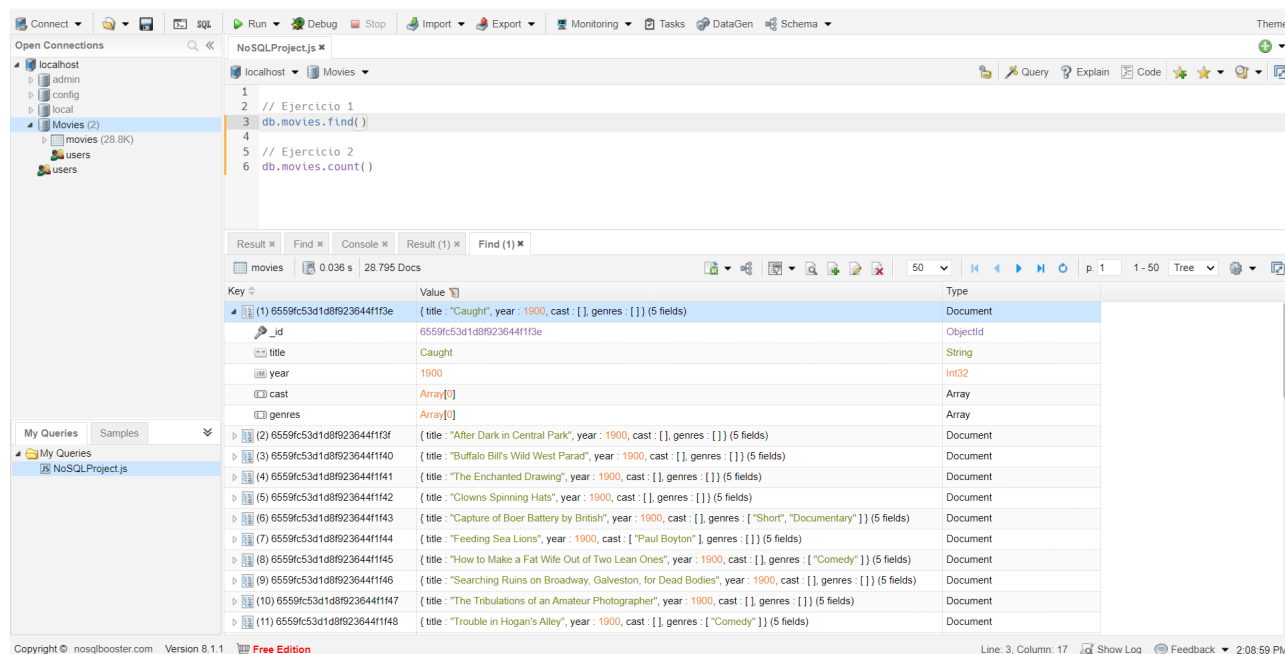
```
{
  "_id" : ObjectId("6559fc53d1d8f923644f1f3e"),
```

```
"title" : "Caught",
"year" : 1900,
"cast" : [ ],
"genres" : [ ]
},
```

Vemos que tiene las siguientes propiedades:

- *title*: es de tipo *string* representando el título de la película.
- *year*: es de tipo *int32* y representa el año de estreno de la película.
- **cast*: es de tipo *array* y aunque está vacío se deduce que se incluirán aquí los actores que conforman el casting de la película y quizá información relativa a ellos.
- *genres*: es de tipo *array* y de nuevo en este ejemplo está vacío, pero representará los géneros cinematográficos a los que pertenece dicha película.

A continuación se muestra una captura de pantalla con el resultado obtenido:



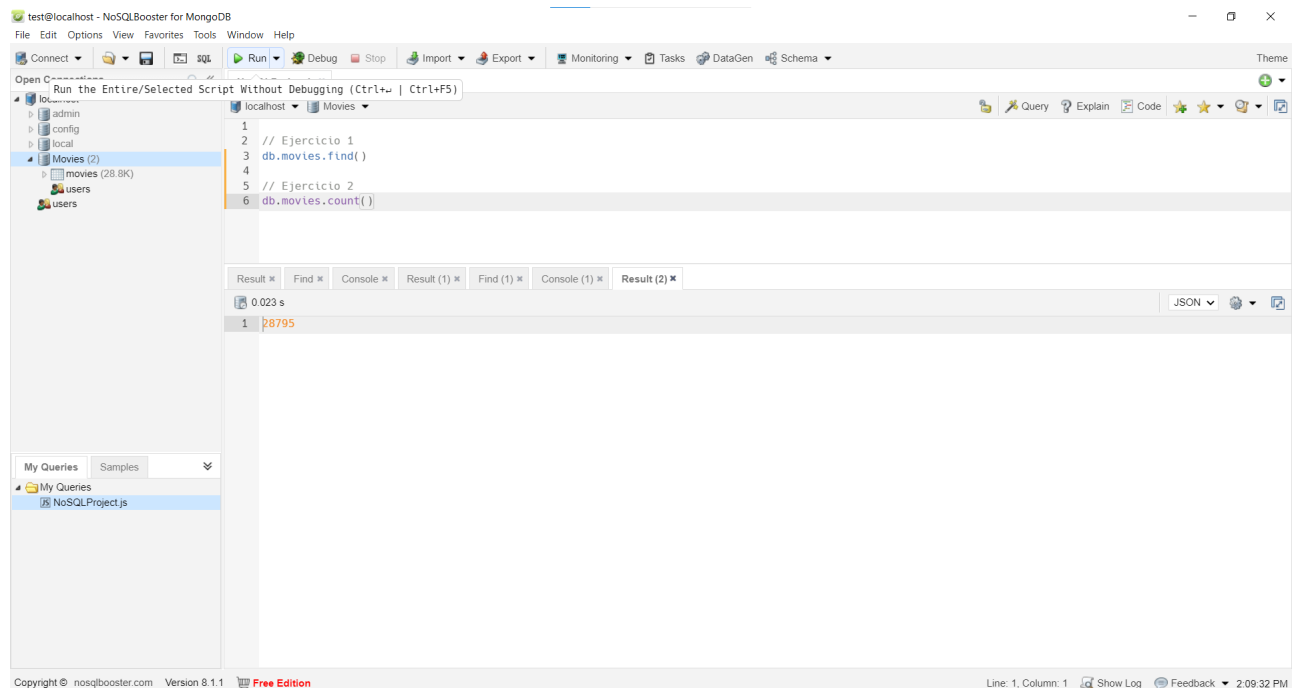
2. Contar cuántos documentos (películas) tiene cargado.

La query de este ejercicio es la siguiente:

```
// Ejercicio 2
db.movies.count()
```

El resultado obtenido es que tiene un total de 28795 documentos la colección.

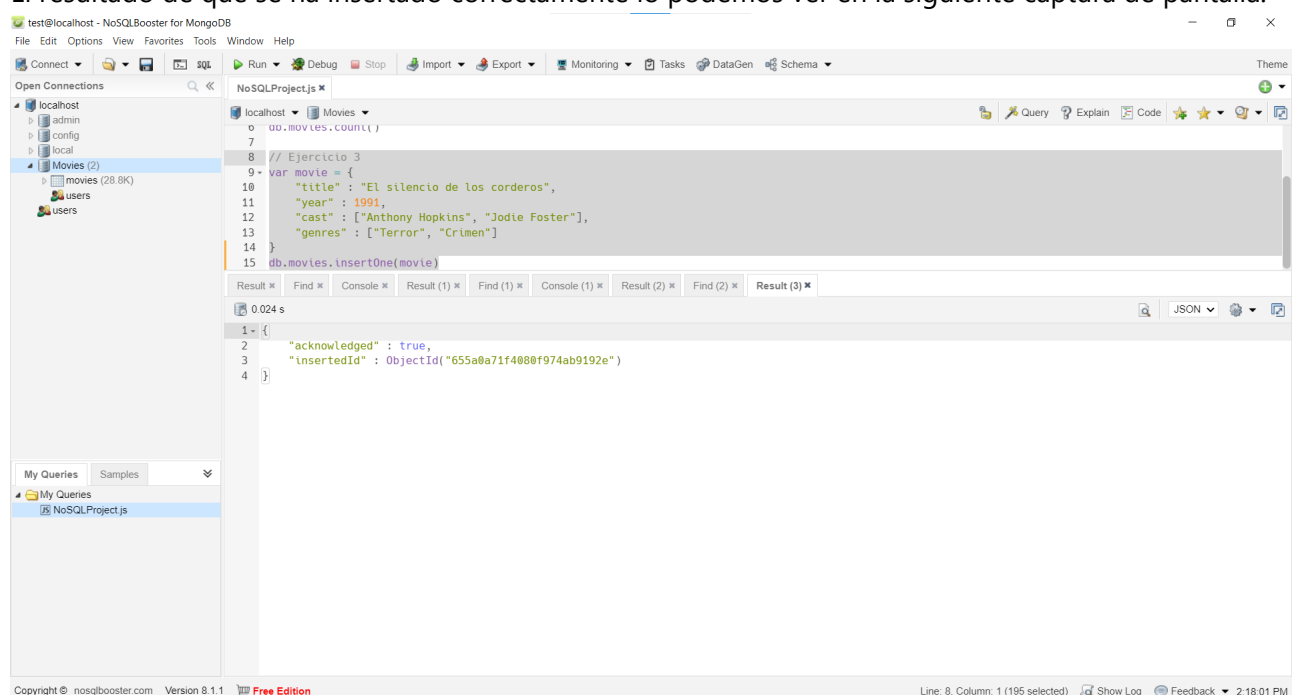
A continuación se muestra una captura de pantalla con el resultado obtenido:



3. Insertar una película. La query de este ejercicio es la siguiente:

```
// Ejercicio 3
var movie = {
  "title" : "El silencio de los corderos",
  "year" : 1991,
  "cast" : ["Anthony Hopkins", "Jodie Foster"],
  "genres" : ["Terror", "Crimen"]
}
db.movies.insertOne(movie)
```

El resultado de que se ha insertado correctamente lo podemos ver en la siguiente captura de pantalla:



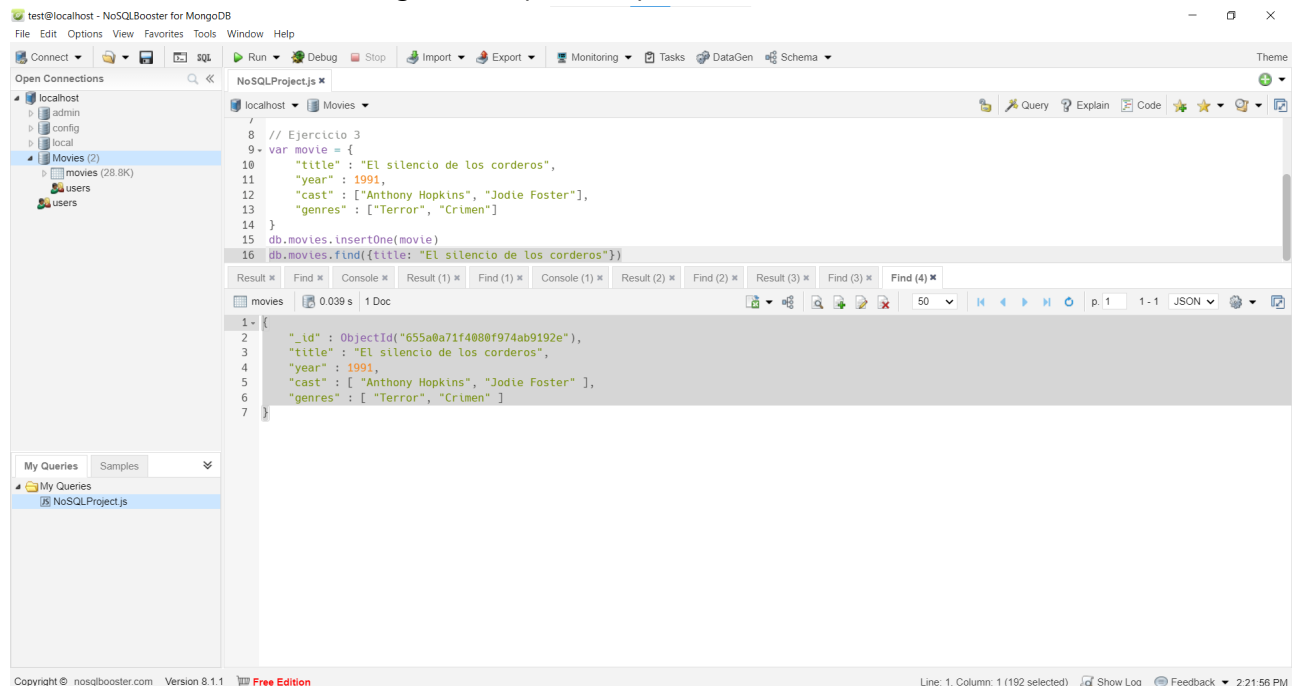
Para comprobar que verdaderamente se ha insertado hacemos una query para buscarla:

```
db.movies.find({title: "El silencio de los corderos"})
```

Obteniendo el siguiente resultado:

```
{
  "_id" : ObjectId("655a0a71f4080f974ab9192e"),
  "title" : "El silencio de los corderos",
  "year" : 1991,
  "cast" : [ "Anthony Hopkins", "Jodie Foster" ],
  "genres" : [ "Terror", "Crimen" ]
}
```

Podemos verlo también en la siguiente captura de pantalla:

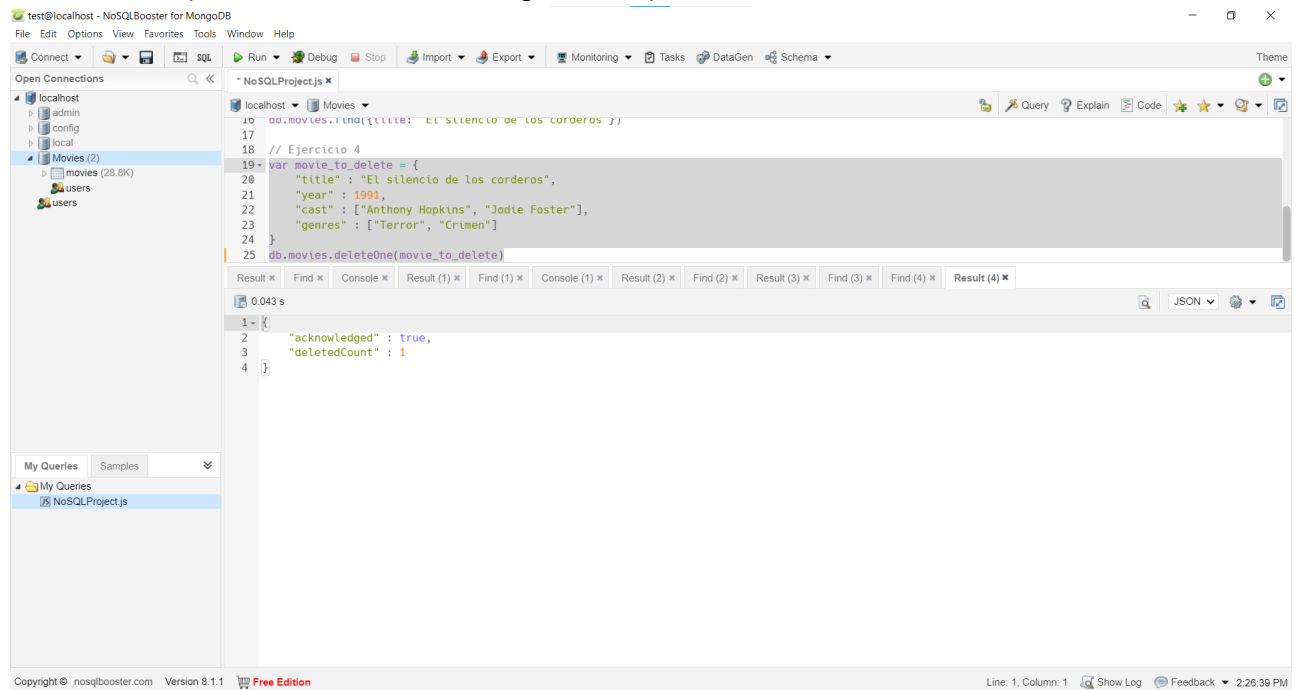


4. Borrar la película insertada en el punto anterior (en el 3).

La query de este ejercicio es la siguiente:

```
var movie_to_delete = {
  "title": "El silencio de los corderos",
  "year": 1991,
  "cast": ["Anthony Hopkins", "Jodie Foster"],
  "genres": ["Terror", "Crimen"]
}
db.movies.deleteOne(movie_to_delete)
```

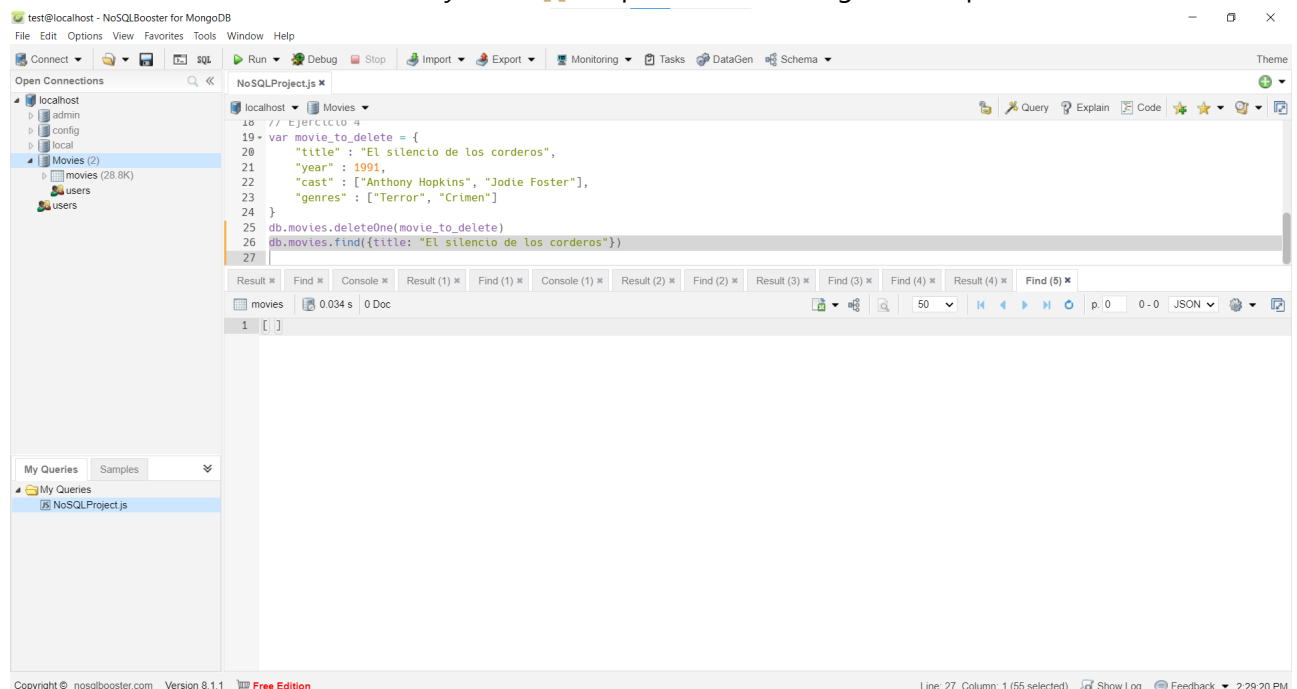
Podemos ver que se ha eliminado en la siguiente captura:



Para comprobarlo, ejecutamos la siguiente query:

```
db.movies.find({title: "El silencio de los corderos"})
```

Obteniendo el resultado de un `array` vacío `[]`. Se puede ver en la siguiente captura:

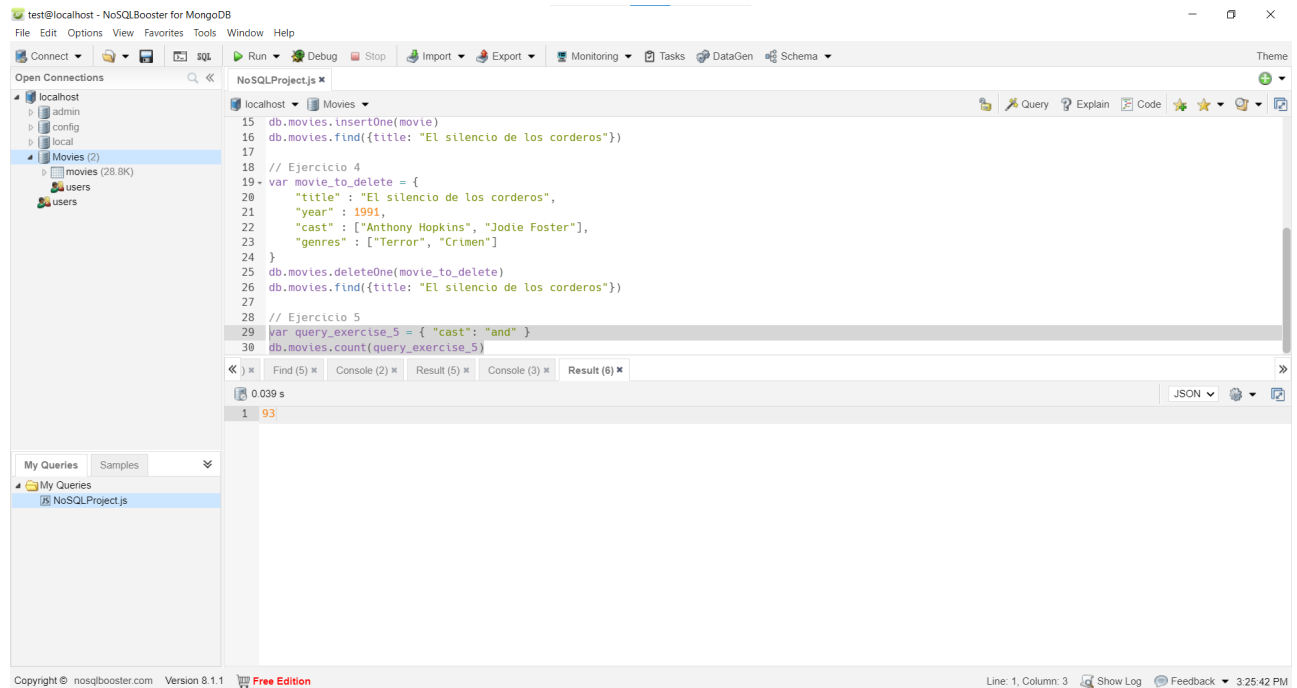


5. Contar cuantas películas tienen actores (cast) que se llaman "and". Estos nombres de actores están por ERROR

La query de este ejercicio es la siguiente:

```
var query_exercise_5 = { "cast": "and" }  
db.movies.count(query_exercise_5)
```

Con esta query estamos filtrando los documentos por aquellos que contienen al menos un "and" en el array del casting (cast) y haciendo un conteo de los documentos totales que cumplen esta condición, obteniendo un total de 93. Esto puede verse en la siguiente captura:

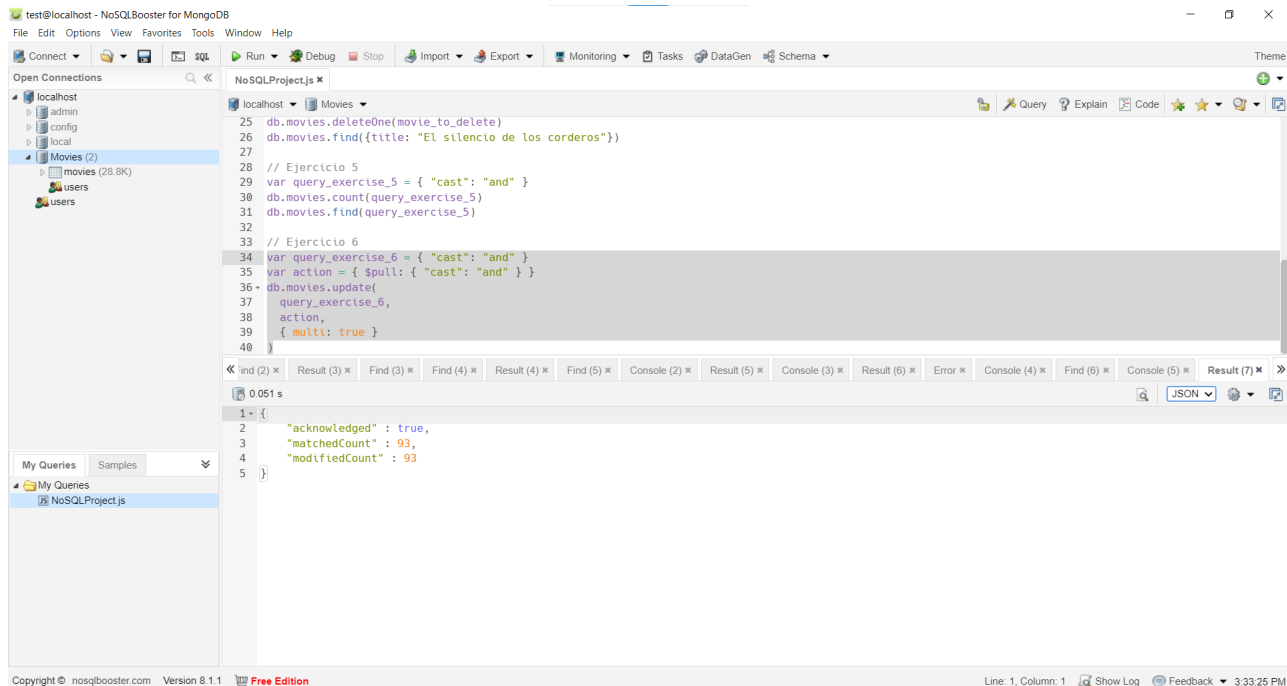


6. **Actualizar los documentos cuyo actor (cast) tenga por error el valor "and" como si realmente fuera un actor. Para ello, se debe sacar únicamente ese valor del array cast. Por lo tanto, no se debe eliminar ni el documento (película) ni su array cast con el resto de actores.**

La query de este ejercicio es la siguiente:

```
var query_exercise_6 = { "cast": "and" }  
var action = { $pull: { "cast": "and" } }  
db.movies.updateMany(  
  query_exercise_6,  
  action  
)
```

En esta query, primero establecemos el criterio de la búsqueda con el primer parámetro de la función `updateMany` (en este caso los documentos que contienen "and" entre sus elementos del array `cast`). En segundo lugar, establecemos la operación a realizar (usamos la acción `$pull` de MongoDB para eliminar el valor "and" del array). Como vemos en la siguiente captura se han modificado los 93 valores anteriores.



```
test@localhost - NoSQLBooster for MongoDB
File Edit Options View Favorites Tools Window Help

Connect  Open Connections  NoSQLProject.js
localhost  localhost  Movies
  admin  config  local
  Movies (2)
    movies (28.8K)
    users
    users

25 db.movies.deleteOne(movie_to_delete);
26 db.movies.find({title: "El silencio de los corderos"});
27
28 // Ejercicio 5
29 var query_exercise_5 = { "cast": "and" };
30 db.movies.count(query_exercise_5);
31 db.movies.find(query_exercise_5);
32
33 // Ejercicio 6
34 var query_exercise_6 = { "cast": "and" };
35 var action = { "$pull": { "cast": "and" } };
36 db.movies.update(
37   query_exercise_6,
38   action,
39   { multi: true }
40 );

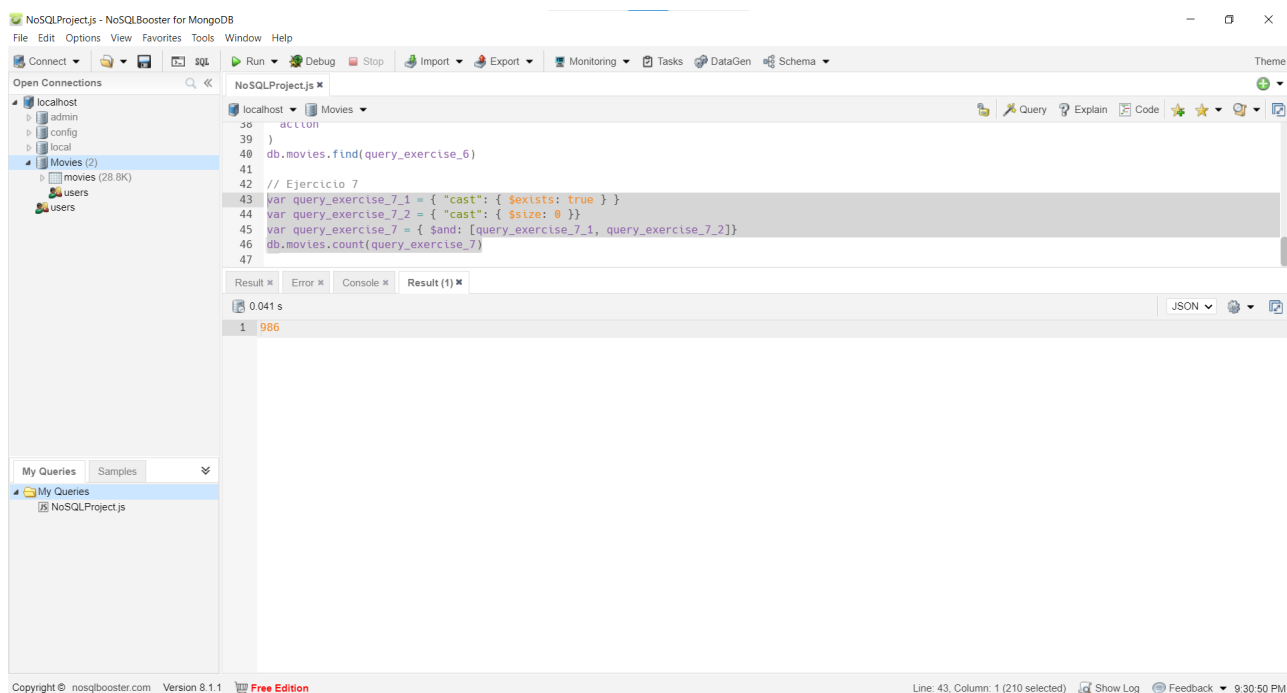
0.051 s
1 {
2   "acknowledged" : true,
3   "matchedCount" : 93,
4   "modifiedCount" : 93
5 }
```

7. Contar cuantos documentos (películas) tienen el array 'cast' vacío.

La query de este ejercicio es la siguiente:

```
var query_exercise_7_1 = { "cast": { $exists: true } }
var query_exercise_7_2 = { "cast": { $size: 0 }}
var query_exercise_7 = { $and: [query_exercise_7_1, query_exercise_7_2]}
db.movies.count(query_exercise_7)
```

Con esta query el resultado obtenido es de 986 documentos con el array vacío. Podemos ver el resultado en la siguiente captura:



```
NoSQLProject.js - NoSQLBooster for MongoDB
File Edit Options View Favorites Tools Window Help

Connect  Open Connections  NoSQLProject.js
localhost  localhost  Movies
  admin  config  local
  Movies (2)
    movies (28.8K)
    users
    users

38 action
39 }
40 db.movies.find(query_exercise_6);
41
42 // Ejercicio 7
43 var query_exercise_7_1 = { "cast": { $exists: true } };
44 var query_exercise_7_2 = { "cast": { $size: 0 } };
45 var query_exercise_7 = { "$and": [query_exercise_7_1, query_exercise_7_2] };
46 db.movies.count(query_exercise_7);
47

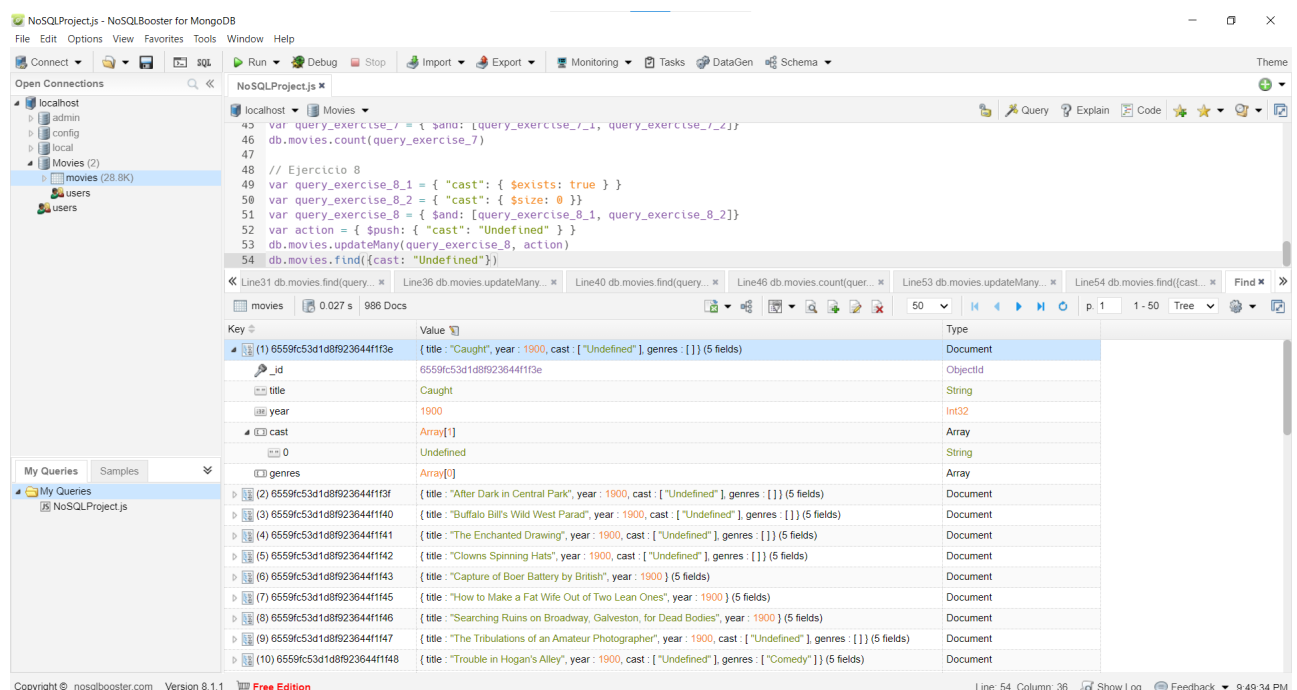
Result  Error  Console  Result (1)
0.041 s
1 986
```


8. Actualizar TODOS los documentos (películas) que tengan el array cast vacío, añadiendo un nuevo elemento dentro del array con valor Undefined. Cuidado! El tipo de cast debe seguir siendo un array. El array debe ser así -> ["Undefined"].

La query de este ejercicio es la siguiente:

```
var query_exercise_8_1 = { "cast": { $exists: true } }
var query_exercise_8_2 = { "cast": { $size: 0 } }
var query_exercise_8 = { $and: [query_exercise_8_1, query_exercise_8_2]}
var action = { $push: { "cast": "Undefined" } }
db.movies.updateMany(query_exercise_8, action)
db.movies.find({cast: "Undefined"})
```

Con esta query el resultado obtenido es que se actualizan 986 documentos con el array vacío poniendo el valor "Undefined". Podemos ver el resultado en la siguiente captura:



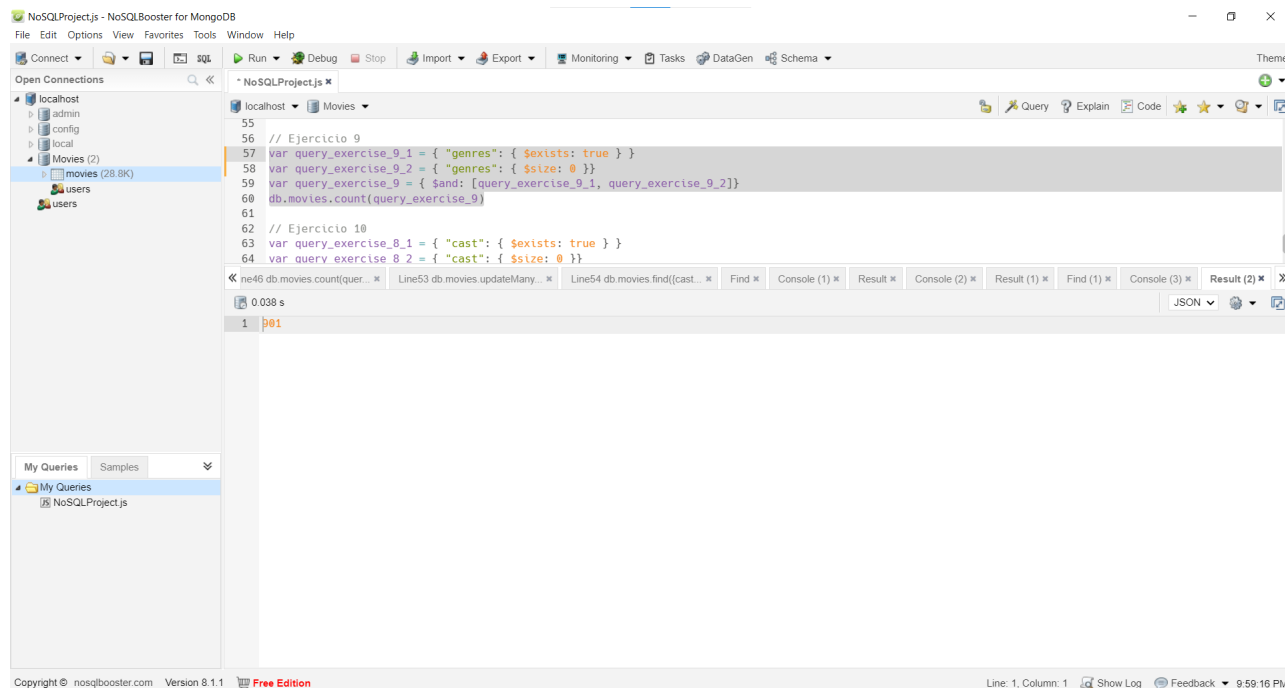
Como se aprecia en la captura, se mantiene el tipo de dato array para cast.

9. Contar cuantos documentos (películas) tienen el array genres vacío.

La query de este ejercicio es la siguiente:

```
var query_exercise_9_1 = { "genres": { $exists: true } }
var query_exercise_9_2 = { "genres": { $size: 0 } }
var query_exercise_9 = { $and: [query_exercise_9_1, query_exercise_9_2]}
db.movies.count(query_exercise_9)
```

Con esta query el resultado obtenido es de 901 documentos con el array vacío. Podemos ver el resultado en la siguiente captura:



10. **Actualizar TODOS los documentos (películas) que tengan el array genres vacío, añadiendo un nuevo elemento dentro del array con valor Undefined. Cuidado! El tipo de genres debe seguir siendo un array.**

La query de este ejercicio es la siguiente:

```
var query_exercise_10_1 = { "genres": { $exists: true } }
var query_exercise_10_2 = { "genres": { $size: 0 }}
var query_exercise_10 = { $and: [query_exercise_10_1, query_exercise_10_2]}
var action = { $push: { "genres": "Undefined" } }
db.movies.updateMany(query_exercise_10, action)
db.movies.find({genres: "Undefined"})
```

Con esta query el resultado obtenido es que se actualizan 901 documentos con el array vacío poniendo el valor "Undefined". Podemos ver el resultado en la siguiente captura:

The screenshot shows the NoSQLBooster for MongoDB interface. The code editor displays a JavaScript query that updates the 'genres' field of a document in the 'movies' collection. The query is as follows:

```

var query_exercise_9 = { $and: [query_exercise_9_1, query_exercise_9_2] }
db.movies.count(query_exercise_9)

// Ejercicio 10
var query_exercise_10_1 = { "genres": { $exists: true } }
var query_exercise_10_2 = { "genres": { $size: 0 } }
var query_exercise_10 = { $and: [query_exercise_10_1, query_exercise_10_2] }
var action = { $push: { "genres": "Undefined" } }
db.movies.updateMany(query_exercise_10, action)
db.movies.find({genres: "Undefined"})

```

The results pane shows a list of documents from the 'movies' collection. The first document is highlighted:

Key	Value	Type
(1) 6559fc53d1d8f923644f1f3e	{ title: "Caught", year: 1900, cast: ["Undefined"], genres: ["Undefined"] } (5 fields)	Document
_id	6559fc53d1d8f923644f1f3e	ObjectId
title	Caught	String
year	1900	Int32
cast	Array[1]	Array
0	Undefined	String
genres	Array[1]	Array
0	Undefined	String

The interface also shows a sidebar with 'My Queries' and 'Samples' tabs, and a bottom status bar indicating 'Line: 63, Column: 1 (321 selected)'.

Como se aprecia en la captura, se mantiene el tipo de dato array para *genres*.

11. **Mostrar el año más reciente / actual que tenemos sobre todas las películas** La query de este ejercicio es la siguiente:

```

db.movies.find({}, {year: true, _id:false})
.sort({year: -1})
.limit(1)

```

En primer lugar obtenemos todos los documentos de la colección con el filtro {}, en segundo lugar aplicamos la proyección {year: true, _id:false} que hace que solo se muestre el campo año y además (porque se muestra siempre por defecto) eliminamos el campo _id para obtener un resultado como el pedido.

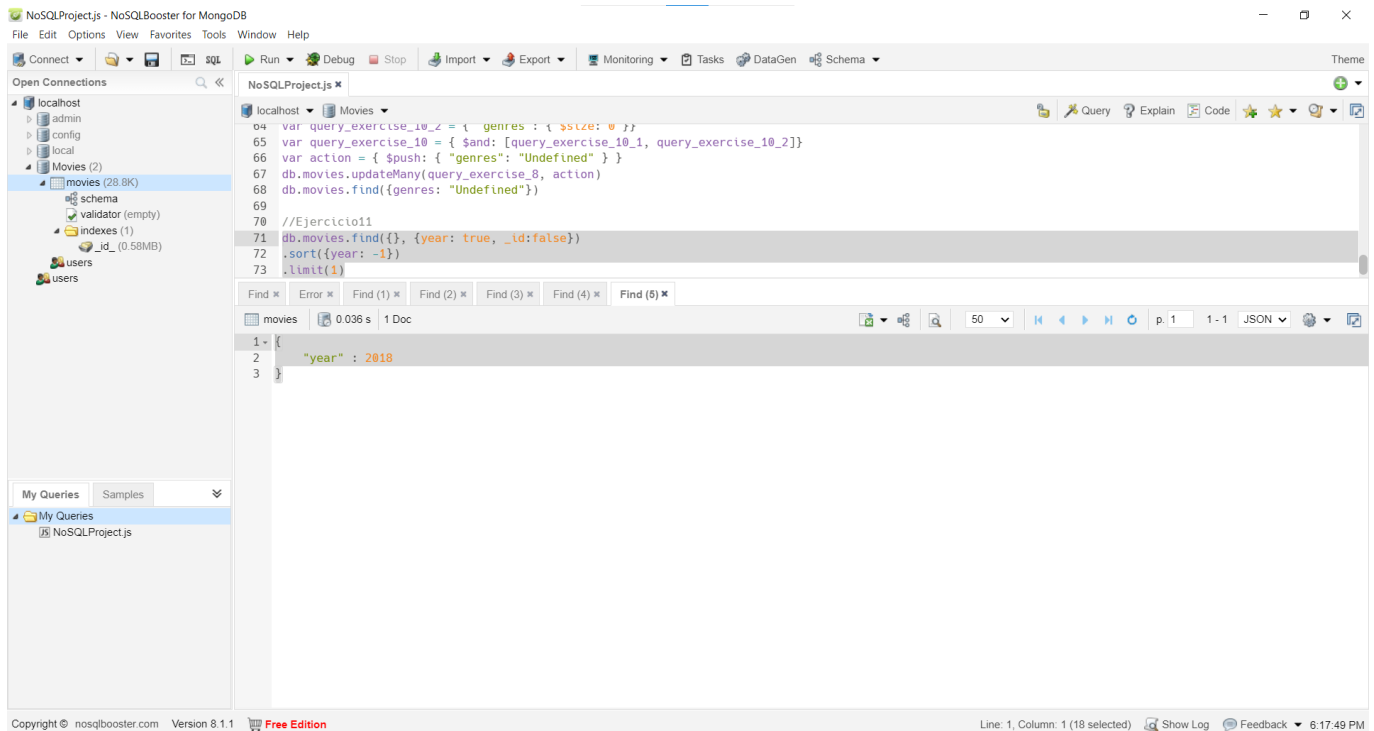
Con esta query el resultado obtenido es el siguiente:

```

{
  "year" : 2018
}

```

Lo que indica que el año más reciente es 2018. A continuación se proporciona una captura de pantalla con la query y el resultado:



12. **Contar cuántas películas han salido en los últimos 20 años. Debe hacerse desde el último año que se tienen registradas películas en la colección, mostrando el resultado total de esos años. Se debe hacer con el Framework de Agregación.** El código de este ejercicio es el siguiente:

```
var maxYearDocument = db.movies.find({}, { year: true, _id: false }).sort({ year:
-1 }).limit(1)
var maxYear
maxYearDocument.forEach(function (doc) {
  maxYear = doc.year;
})
var minYear = maxYear - 20
var query_1 = {"year": { $gte: minYear}}
var query_2 = {"year": { $lte: maxYear}}
var query_3 = {$and: [query_1, query_2]}

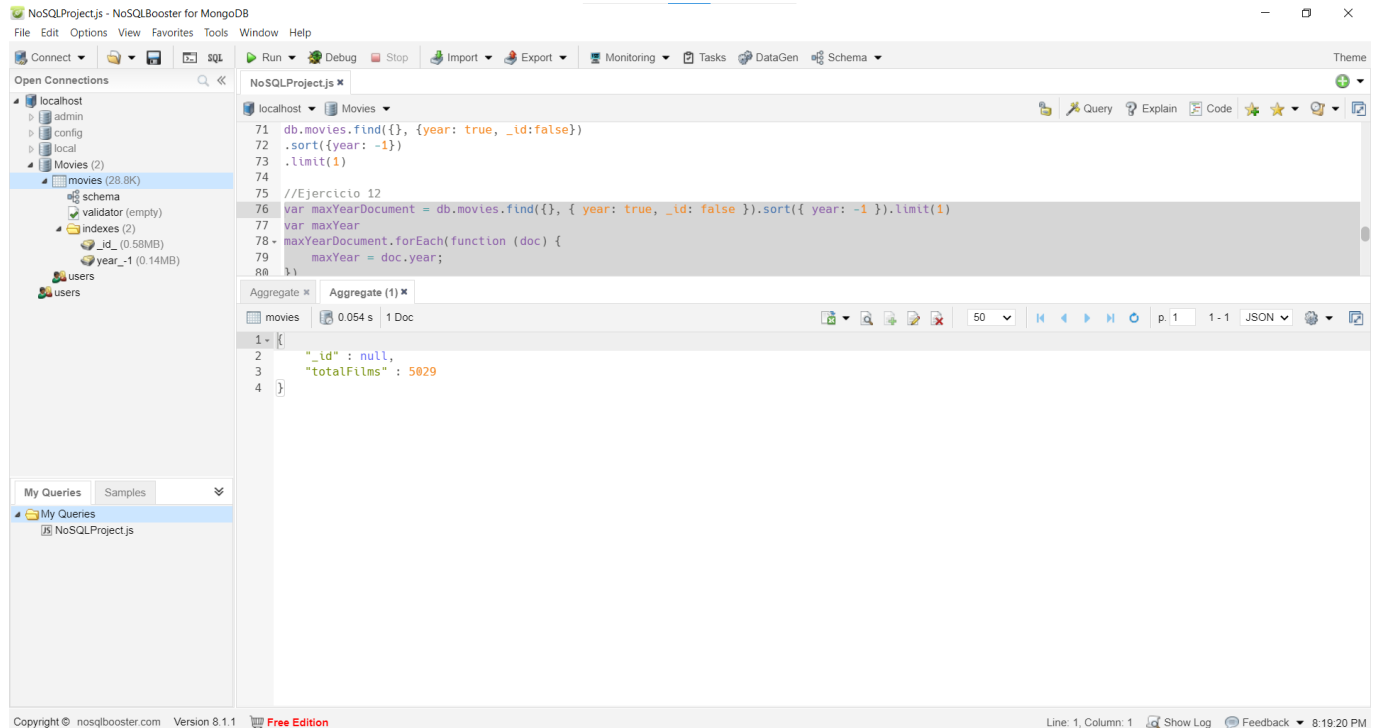
db.movies.aggregate([
  {$match: query_3},
  {$group: {_id: 0, totalFilms: { $sum: 1 }}}
])
```

En este ejercicio, primero tratamos de calcular dinámicamente el año máximo de la base de datos, para ello realizamos la query del ejercicio anterior, y teniendo en cuenta que *javascript* devuelve un *promise* como resultado de la búsqueda, para acceder al valor necesitamos ejecutar un *foreach* para setear el valor de la variable que representa el año máximo. En segundo lugar seteamos la variable que representa el año mínimo de la búsqueda (20 años menos) y realizamos la agregación. En primer lugar hacemos un match de los documentos con la fecha comprendida entre los valores deseados y luego un agrupamiento de los mismos, mandamos todos los documentos al mismo grupo (*_id : 0*) y sumamos la cantidad total de documentos.

Con esta query el resultado obtenido es el siguiente:

```
{
  "_id" : null,
  "totalFilms" : 5029
}
```

La captura de pantalla del ejercicio es esta:



13. Contar cuántas películas han salido en la década de los 60 (del 60 al 69 incluidos). Se debe hacer con el Framework de Agregación El código de este ejercicio es el siguiente:

```
var query_1 = {"year": { $gte: 1960 }}
var query_2 = {"year": { $lte: 1969 }}
var query_3 = {$and: [query_1, query_2]}

db.movies.aggregate([
  {$match: query_3},
  {$group: {_id: 0, totalFilms: { $sum: 1 }}}
])
```

En este realizamos algo similar al anterior, con la diferencia de que esta vez conocemos el rango de fechas sin necesidad de calcularlo dinámicamente. Por ello únicamente ejecutamos el pipeline de agregación anterior con el nuevo rango de fechas.

Con esta query el resultado obtenido es el siguiente:

```
{
  "_id" : null,
  "totalFilms" : 1414
}
```

La captura de pantalla del ejercicio es esta:

