

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:

The screenshot shows the MongoDB Compass interface. On the left, the sidebar lists databases (movies, admin, config, local) and collections (Movies, movies). The main area displays the 'Movies.movies' collection with 28.8k documents and 1 index. It shows a list of documents with fields like _id, title, year, cast, and genres. A message at the bottom indicates 'Import completed. 28795 documents imported.'

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 quizás 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:

The screenshot shows the NoSQLBooster IDE interface. In the top navigation bar, 'Run' is selected. Below it, the 'Movies' collection is chosen. The code editor contains the following query:

```

1 // Ejercicio 1
2
3 db.movies.find()
4
5 // Ejercicio 2
6 db.movies.count()

```

The 'Result' tab is active, displaying the results of the 'find()' query. The results table has columns for 'Key', 'Value', and 'Type'. The first document is highlighted:

Key	Type
6559fc53d1dbf923644ff1f3e	Document
_id	Objectid
title	String
year	Int32
cast	Array
genres	Array

The 'Value' column shows the document content:

```

{
  "_id": "6559fc53d1dbf923644ff1f3e",
  "title": "Caught",
  "year": 1900,
  "cast": [],
  "genres": []
}

```

Below the table, the results of the 'count()' query are shown:

Count
28.795 Docs

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:

The screenshot shows the NoSQLBooster interface for MongoDB. In the left sidebar, under 'My Queries', there is a file named 'NoSQLProject.js'. The main area displays a script with the following code:

```

1
2 // Ejercicio 1
3 db.movies.find()
4
5 // Ejercicio 2
6 db.movies.count()

```

The 'Result' tab shows the output: 1 | 2879. This indicates that there are 2879 documents in the 'movies' collection.

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:

The screenshot shows the NoSQLBooster interface after running the insertion query. The 'Result' tab displays the response from the database:

```

1 {
  "acknowledged" : true,
  "insertedId" : ObjectId("655a0a71f4080f974ab9192e")
}

```

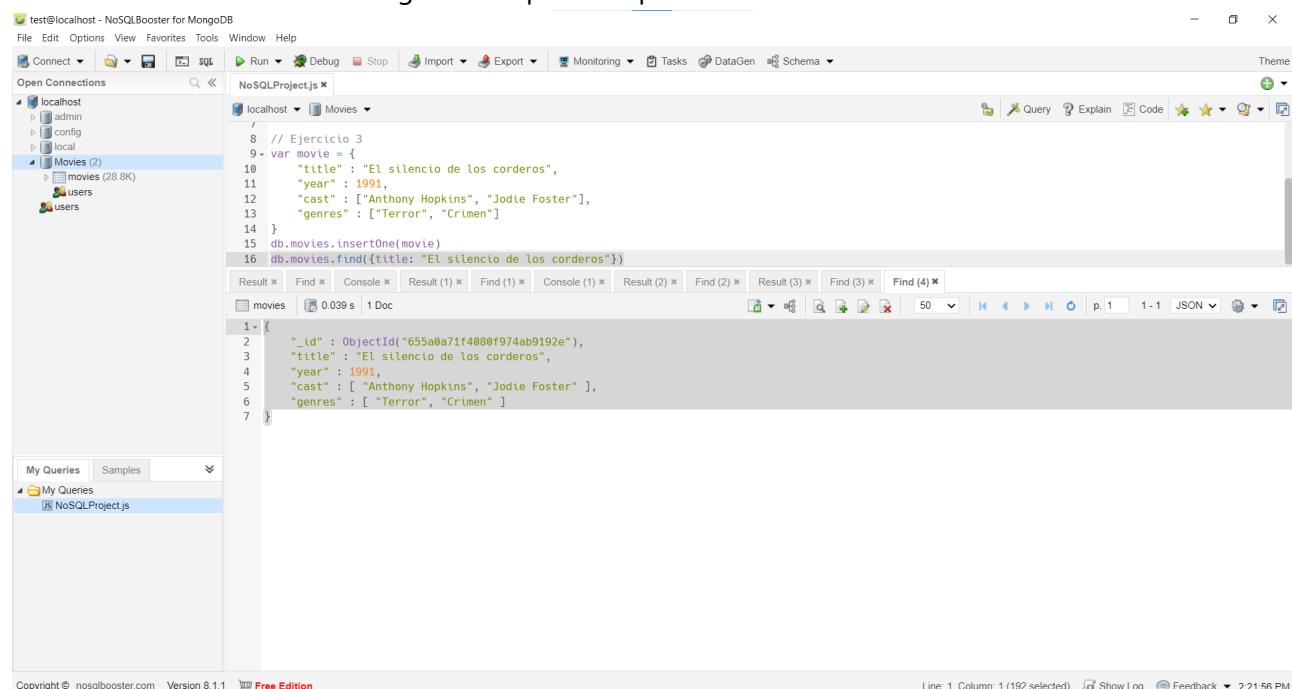
Para comprobar que verdaderamente se ha insertado hacemos una query para buscarla:

```
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:

The screenshot shows the NoSQLBooster interface for MongoDB. In the left sidebar, under 'Open Connections' for 'localhost', there is a 'Movies' database with a collection named '(2)' containing 28.8K documents. A query is being run in the main pane:

```
db.movies.findOne({title: "El silencio de los corderos"})
17
18 // Ejercicio 4
19 + var movie_to_delete = {
20   "title": "El silencio de los corderos",
21   "year": 1991,
22   "cast": ["Anthony Hopkins", "Jodie Foster"],
23   "genres": ["Terror", "Crimen"]
24 }
25 db.movies.deleteOne(movie_to_delete)
```

The result pane shows the execution time: 0.043 s and the response:

```
1 [
2   "acknowledged": true,
3   "deletedCount": 1
4 ]
```

At the bottom, it says 'Copyright © nosqlbooster.com Version 8.1.1' and 'Free Edition'.

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:

The screenshot shows the NoSQLBooster interface for MongoDB. The 'Movies' database now has 0 documents. The same query is run again:

```
db.movies.find({title: "El silencio de los corderos"})
25 db.movies.deleteOne(movie_to_delete)
26 db.movies.find({title: "El silencio de los corderos"})
27
```

The result pane shows the execution time: 0.034 s and the response:

```
movies | 0.034 s | 0 Doc
1 [ ]
```

At the bottom, it says 'Copyright © nosqlbooster.com Version 8.1.1' and 'Free Edition'.

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:

The screenshot shows the NoSQLBooster application interface for MongoDB. The left sidebar shows a tree view of databases (localhost, admin, config, local) and collections (Movies, movies). The main area displays the following code in a script named 'NoSQLProject.js':

```
15 db.movies.insertOne(movie)
16 db.movies.find({title: "El silencio de los corderos"})
17
18 // Ejercicio 4
19 var movie_to_delete = {
20   "title": "El silencio de los corderos",
21   "year": 1991,
22   "cast": ["Anthony Hopkins", "Jodie Foster"],
23   "genres": ["Terror", "Crimen"]
24 }
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)
```

The 'Result' tab shows the output: 1 93. The bottom status bar indicates 'Line: 1, Column: 3'.

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
Run Debug Stop Import Export Monitoring Tasks DataGen Schema
Open Connections
localhost Movies
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 )

```

Result (2) × Find (3) × Find (4) × Result (4) × Find (5) × Console (2) × Result (5) × Console (3) × Result (6) × Error × Console (4) × Find (6) × Console (5) × Result (7) ×

0.051 s

```

1 - {
2   "acknowledged" : true,
3   "matchedCount" : 93,
4   "modifiedCount" : 93
5 }

```

My Queries Samples

NoSQLProject.js

Copyright © nosqlbooster.com Version 8.1.1 [Free Edition](#)

Line: 1, Column: 1 Show Log Feedback 3:33:25 PM

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
Run Debug Stop Import Export Monitoring Tasks DataGen Schema
Open Connections
localhost Movies
36 ACTION
37
38 )
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

My Queries Samples

NoSQLProject.js

Copyright © nosqlbooster.com Version 8.1.1 [Free Edition](#)

Line: 43, Column: 1 (210 selected) Show Log Feedback 9:30:50 PM

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:

Key	Value	Type
6559fc53d1dbf923644ff13e	{ title: "Caught", year: 1900, cast: [] (5 fields)}	Document
_id	6559fc53d1dbf923644ff13e	ObjectID
title	Caught	String
year	1900	Int32
cast	Array[1]	Array
0	Undefined	String
genres	Array[0]	Array

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:

```

NoSQLProject.js - NoSQLBooster for MongoDB
File Edit Options View Favorites Tools Window Help
Run Debug Import Export Monitoring Tasks DataGen Schema
Open Connections <> NoSQLProject.js
localhost Movies (2)
admin config local movies (28.8K)
users users
localhost Movies
55
56 // Ejercicio 9
57 var query_exercise_9_1 = { "genres": { $exists: true } }
58 var query_exercise_9_2 = { "genres": { $size: 0 } }
59 var query_exercise_9 = { $and: [query_exercise_9_1, query_exercise_9_2] }
60 db.movies.count(query_exercise_9)
61
62 // Ejercicio 10
63 var query_exercise_10_1 = { "cast": { $exists: true } }
64 var query_exercise_10_2 = { "cast": { $size: 0 } }
65
66 db.movies.updateMany(query_exercise_10_1, { $push: { "cast": "Undefined" } })
67 db.movies.find({genres: "Undefined"})

```

My Queries Samples <>

My Queries NoSQLProject.js

Copyright © nosqlbooster.com Version 8.1.1 [Free Edition](#)

Line: 1, Column: 1 Show Log Feedback 9:59:16 PM

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 interface for MongoDB. In the top-left, there's a tree view of databases and collections: 'localhost' (admin, config, local), 'Movies (2)' (movies, users). The 'movies' collection is selected, showing 28.8K documents. The top-right has tabs for 'Query', 'Explain', 'Code', etc. Below the tabs is a results table with columns 'Key', 'Value', and 'Type'. The table contains 9 rows of movie documents, each with fields like '_id', 'title', 'year', 'cast', and 'genres'. The 'genres' field is consistently shown as an array of strings. At the bottom, there's a command line interface with a query editor containing some JavaScript code for updating movies.

Key	Type
(1) 6559fc53d1d8f923644f1f3e	Document
_id	Objectid
title	String
year	Int32
cast	Array[1]
0	String
genres	Array[1]
0	String
(2) 6559fc53d1d8f923644f1f3f	Document
(3) 6559fc53d1d8f923644f1f40	Document
(4) 6559fc53d1d8f923644f1f41	Document
(5) 6559fc53d1d8f923644f1f42	Document
(6) 6559fc53d1d8f923644f1f44	Document
(7) 6559fc53d1d8f923644f1f46	Document
(8) 6559fc53d1d8f923644f1f47	Document
(9) 6559fc53d1d8f923644f1f49	Document

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:

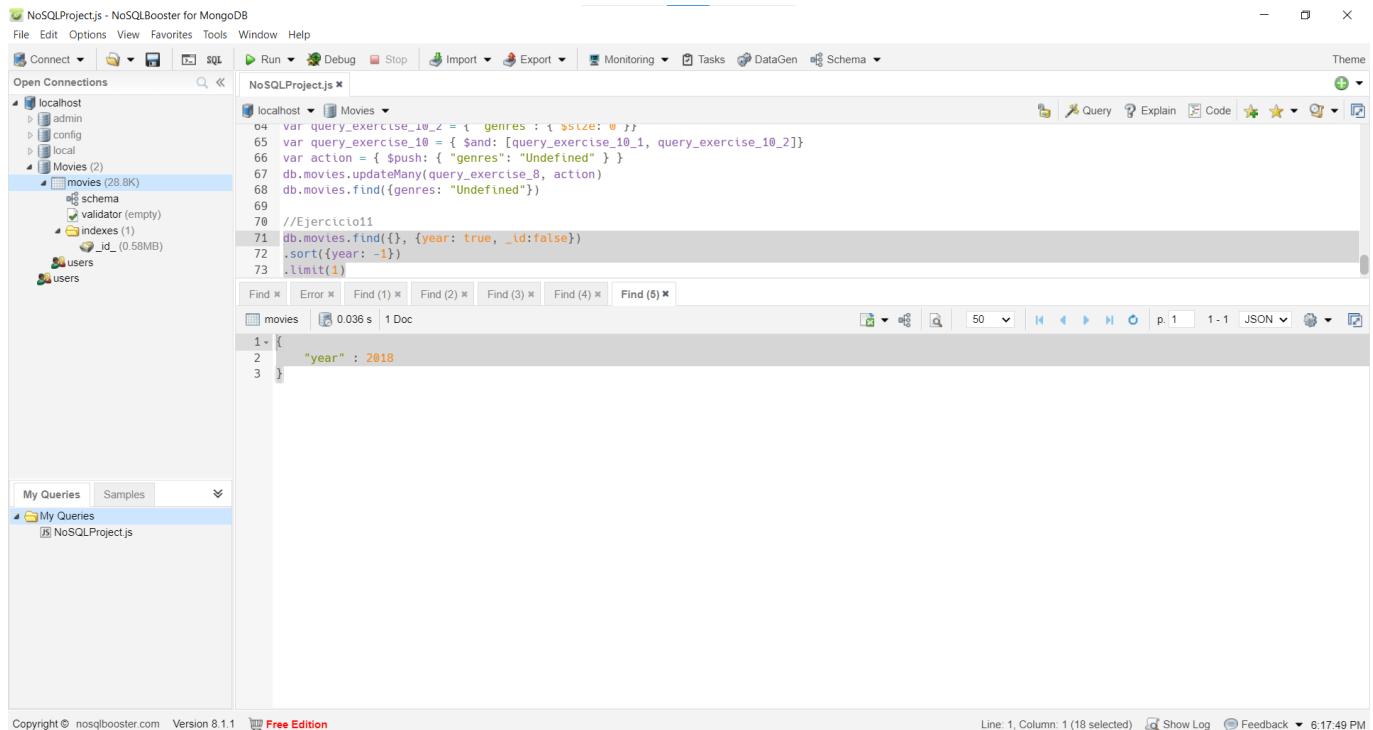
```
db.movies.find({}, {year: true, _id:false})
  .sort({year: -1})
  .limit(1)
```

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,
  "total" : 5029
}
```

La captura de pantalla del ejercicio es esta:

The screenshot shows the NoSQLBooster application interface. In the top navigation bar, 'File', 'Edit', 'Options', 'View', 'Favorites', 'Tools', 'Window', and 'Help' are visible. Below the menu is a toolbar with icons for 'Run', 'Debug', 'Stop', 'Import', 'Export', 'Monitoring', 'Tasks', 'DataGen', and 'Schema'. The main area shows a database tree on the left with 'localhost' expanded, showing 'admin', 'config', 'local', and 'Movies (4)' collections. The 'Movies' collection has sub-folders for 'actors', 'genres', and 'movies'. On the right, a code editor window titled 'NoSQLProject.js (1)' contains the following MongoDB aggregation pipeline:

```

73   .limit(1)
74
75 // Ejercicio 12
76 var maxYearDocument = db.movies.find({}, { year: true, _id: false }).sort({ year: -1 }).limit(1)
77 var maxYear
78 maxYearDocument.forEach(function (doc) {
79   maxYear = doc.year;
80 })
81 var minYear = maxYear - 20
82 var query_1 = {"year": { $gte: minYear}}
83 var query_2 = {"year": { $lte: maxYear}}
84 var query_3 = {$and: [query_1, query_2]}
85
86 db.movies.aggregate([
87   {$match: query_3},
88   {$group: { _id: null, total: { $sum: 1 } }}
89 ])
90
91 // Ejercicio 13
92 var query_1 = {"year": { $gte: 1960}}
93 var query_2 = {"year": { $lte: 1969}}

```

Below the code editor, a results pane shows the output of the last command:

```

1 - [
2   {
3     "_id" : null,
4     "total" : 5029
5   }
]

```

At the bottom of the interface, there are tabs for 'My Queries' and 'Samples', and a status bar indicating 'Copyright © nosqlbooster.com Version 8.1.1 Free Edition'.

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,
  "total" : 1414
}
```

La captura de pantalla del ejercicio es esta:

The screenshot shows the NoSQLBooster for MongoDB interface. On the left, the 'Open Connections' sidebar lists databases: 'localhost' (admin, config, local, Movies, users). The 'Movies' database is selected. In the main pane, a query editor window titled 'NoSQLProject.js (1)' contains the following code:

```

86 - db.movies.aggregate([
87 -   {$match: query_3},
88 -   {$group: {_id: null, total: { $sum: 1 }}}
89 - ])
90
91 // Ejercicio 13
92 var query_1 = {"year": { $gte: 1960}}
93 var query_2 = {"year": { $lte: 1969}}
94 var query_3 = {$and: [query_1, query_2]}
95
96 - db.movies.aggregate([
97 -   {$group: {_id: null, total: { $sum: 1 }}}
98 - ])
99
100
101 // Ejercicio 14
102 - db.movies.aggregate([
103 -   {$group: { _id: "$year", totalFilms: { $sum: 1 } } },
104 -   {$group: { _id: "$totalFilms", years: { $push: "$_id" } } },
105 -   {$sort: { _id: -1 } },
106 -   {$unwind: "$years"
107 - },
108 -   {$replaceRoot: {
109 -     newRoot: {
110 -       _id: "$years",
111 -       pelis: "$_id"
112 -     }
113 -   }
114 - },
115 -   {$sort: { _id: -1 }}
116 - ])

```

Below the code, the results pane shows a single document:

```

1 - [
2 -   "_id": null,
3 -   "total": 1414
4 - ]

```

At the bottom, the status bar indicates: Copyright © nosqlbooster.com Version 8.1.1 Free Edition Line: 1, Column: 1 Show Log Feedback 6:55:23 PM

14. Mostrar el año u años con más películas mostrando el número de películas de ese año. Revisar si varios años pueden compartir tener el mayor número de películas.

El código de este ejercicio es el siguiente:

```

db.movies.aggregate([
  { $group: { _id: "$year", totalFilms: { $sum: 1 } } },
  { $group: { _id: "$totalFilms", years: { $push: "$_id" } } },
  { $sort: { _id: -1 } },
  { $limit: 1 },
  {
    $unwind: "$years"
  },
  {
    $replaceRoot: {
      newRoot: {
        _id: "$years",
        pelis: "$_id"
      }
    }
  },
  {$sort: { _id: -1 }}
]);

```

En primer lugar agrupamos por año y contabilizamos el total de películas por año. Después agrupamos en función del total de películas y en los nuevos documentos incluimos un array de años en los que se hicieron esa cantidad de películas. Ordenamos de mayor a menor, de esta forma en el primer documento obtenemos el máximo de películas en un año así como en un array los años en que se hicieron esa cantidad de películas.

Finalmente con `$unwind` creamos un nuevo documento por cada año en el array `years`. Finalmente reestructuramos los documentos añadiendo el año como `_id` y el total de pelis como el `_id` de los documentos previos a la transformación (dónde en el `_id` estaba el total de pelis de ese año). Finalmente, ordenamos descendente según el año.

Con esta query el resultado obtenido es el siguiente:

```
{
  "_id" : 1919,
  "pelis" : 634
}
```

Siendo 1919 el año con más películas en la base de datos, con un total de 634. La captura de pantalla del ejercicio es esta:

```
db.movies.aggregate([
  { $group: { _id: "$year", totalFilms: { $sum: 1 } } },
  { $group: { _id: "$totalFilms", years: { $push: "$_id" } } },
  { $sort: { _id: -1 } },
  { $limit: 1 },
  { $unwind: "$years" },
  { $replaceRoot: {
    newRoot: {
      _id: "$years",
      pelis: "$_id"
    }
  }},
  { $sort: { _id: -1 } }
]);
//Ejercicio 15
```

movies	0.043 s 1 Doc
1	[
2	{"_id": 1919,
3	"pelis": 634
4]

15. Mostrar el año u años con más películas mostrando el número de películas de ese año. Revisar si varios años pueden compartir tener el mayor número de películas.

El código de este ejercicio es el siguiente:

```
db.movies.aggregate([
  { $group: { _id: "$year", totalFilms: { $sum: 1 } } },
  { $group: { _id: "$totalFilms", years: { $push: "$_id" } } },
  { $sort: { _id: 1 } },
  { $limit: 1 },
  { $unwind: "$years" },
  { $replaceRoot: {
```

```
        newRoot: {
            _id: "$years",
            pelis: "$_id"
        }
    },
    {$sort: { _id: -1 }}
]);

```

El razonamiento es idéntico al anterior pero ordenando los primeros grupos de menor a mayor por el total de películas en lugar de mayor a menor.

Con esta query el resultado obtenido es el siguiente:

```
/* 1 */
{
    "_id" : 1907,
    "pelis" : 7
},
/* 2 */
{
    "_id" : 1906,
    "pelis" : 7
},
/* 3 */
{
    "_id" : 1902,
    "pelis" : 7
}
```

Siendo así 1907, 1906 y 1902 los años con menos películas de la base de datos con un total de 7. La captura de pantalla del ejercicio es esta:

The screenshot shows the NoSQLBooster interface for MongoDB. On the left, the database structure is displayed with 'localhost' selected. Under 'localhost', there are databases: admin, config, local, and Movies (2). The 'Movies' database has two collections: movies (28.8K) and users. The 'movies' collection is expanded, showing its schema, validator (empty), indexes (2), and documents (3). One document is highlighted with the ID '1907'. On the right, the code editor shows a script named 'NoSQLProject.js' containing an aggregation query. Below the code editor is a results grid titled 'movies' showing three documents with IDs 1907, 1906, and 1902, each having a single field 'pelis' with value 7. At the bottom, there are tabs for 'My Queries' and 'Samples', and a status bar indicating 'Copyright © nosqlbooster.com Version 8.1.1' and 'Line: 1, Column: 1'.

```

NoSQLProject.js - NoSQLBooster for MongoDB
File Edit Options View Favorites Tools Window Help
Connect Run Debug Stop Import Export Monitoring Tasks DataGen Schema Theme
Open Connections NoSQLProject.js
localhost admin config local Movies (2)
  movies (28.8K)
    schema
    validator (empty)
    indexes (2)
      _id_ (0.58MB)
      year_-1 (0.14MB)
  users
  users
NoSQLProject.js
118   {$sort: { _id: -1 }};
119 ];
120
121 //Ejercicio 15
122 db.movies.aggregate([
123   { $group: { _id: "$year", totalFilms: { $sum: 1 } } },
124   { $group: { _id: "$totalFilms", years: { $push: "$_id" } } },
125   { $sort: { _id: 1 } },
126   { $limit: 1 },
127   {
128     $unwind: "$years"
129   },
130   {
131     $replaceRoot: {
132       newRoot: {
133         _id: "$years",
134         pelis: "$_id"
135       }
136     },
137   },
138   { $sort: { _id: -1 } }
139 ]);
140
My Queries Samples
My Queries NoSQLProject.js
movies 0.040 s 3 Docs
  _id pelis
  1 1907 7
  2 1906 7
  3 1902 7
Copyright © nosqlbooster.com Version 8.1.1 Free Edition
Line: 1, Column: 1 Show Log Feedback 7:52:38 PM

```

16. Guardar en nueva colección llamada “actors” realizando la fase \$unwind por actor. Después, contar cuantos documentos existen en la nueva colección.

El código de este ejercicio es el siguiente:

```

db.movies.aggregate([
  { $unwind: "$cast" },
  { $project: {"_id": false}},
  { $out: "actors"}
]);

db.actors.count()

```

Con esto en primer lugar realizamos un **unwind** para partir cada documento en tantos como actores haya en el array de cast. Tras esto eliminamos el **_id** de los documentos para evitar errores al crear la nueva colección. Finalmente contamos el total de actores, como se pide en la nueva colección, obteniendo 83224 documentos.

La captura de pantalla del ejercicio es esta:

The screenshot shows the NoSQLBooster application interface for MongoDB. The top menu bar includes File, Edit, Options, View, Favorites, Tools, Window, Help, and a Theme selector. The left sidebar displays the database structure under 'localhost' with the 'Movies' database selected, showing collections like 'actors' (83.2K), 'schema', 'validator (empty)', 'indexes (1)', '_id_ (0.82MB)', 'movies' (28.8K), and 'users'. The main workspace contains a query editor with the following Aggregation Pipeline:

```
126      { $limit: 1},  
127      {  
128          $unwind: "$years"  
129      },  
130      {  
131          $replaceRoot:  
132              newRoot: {  
133                  _id: "$years",  
134                  pelis: "$_id"  
135              }  
136      },  
137      {$sort: { _id: -1 }}  
138  ]);  
139  
140  
141 //Ejercicio 16  
142 db.movies.aggregate([  
143     { $unwind: "$cast" },  
144     { $project: { "cast": true, "_id": false}},  
145     { $out: "actors" }  
146 ]);  
147  
148 db.actors.count()
```

The results pane at the bottom shows the output of the last command: 'db.actors.count()' with a value of 83224.

17. Sobre actors (nueva colección), mostrar la lista con los 5 actores que han participado en más películas mostrando el número de películas en las que ha participado. Importante! Se necesita previamente filtrar para descartar aquellos actores llamados "Undefined". Aclarar que no se eliminan de la colección, sólo que filtramos para que no aparezcan.

El código de este ejercicio es el siguiente:

```
db.actors.aggregate([
  { $match: { cast: { $ne: "Undefined" } } },
  { $group: { _id: "$cast", cuenta: { $sum: 1 } } },
  { $sort: { cuenta: -1 } },
  { $limit: 5 }
]);
```

Filtramos para eliminar los actores `Undefined`. Tras esto agrupamos por actor contando las películas en las que aparece. Finalmente ordenamos de mayor a menor y nos quedamos con las primeras 5. El resultado es el siguiente:

```
/* 1 */
{
    "_id" : "Harold Lloyd",
    "cuenta" : 190
},
/* 2 */
{
    "_id" : "Hoot Gibson",
    "cuenta" : 142
```

```

},
/* 3 */
{
  "_id" : "John Wayne",
  "cuenta" : 136
},
/* 4 */
{
  "_id" : "Charles Starrett",
  "cuenta" : 116
},
/* 5 */
{
  "_id" : "Bebe Daniels",
  "cuenta" : 103
}

```

La captura de pantalla del ejercicio es esta:

The screenshot shows the NoSQLBooster application interface for MongoDB. On the left, the 'Open Connections' sidebar shows a connection to 'localhost' with databases 'admin', 'config', 'local', and 'Movies'. The 'Movies' database has two collections: 'movies' (28.8K documents) and 'users'. In the main workspace, a file named 'NoSQLProject.js' is open, containing the following MongoDB aggregation pipeline:

```

135   },
136   }
137 },
138 { $sort: { _id: -1 }
139 ];
140
141 //Ejercicio 16
142 db.movies.aggregate([
143   { $unwind: "$cast" },
144   { $project: { "_id": false } },
145   { $out: "actors" }
146 ]);
147
148 db.actors.count()
149
150 //Ejercicio 17
151 db.actors.aggregate([
152   { $match: { cast: { $ne: "Undefined" } } },
153   { $group: { _id: "$cast", cuenta: { $sum: 1 } } },
154   { $sort: { cuenta: -1 } },
155   { $limit: 5 }
156 ]);
157

```

Below the code editor, a results table titled 'actors' displays the following data:

_id	cuenta
1 Harold Lloyd	190
2 Hoot Gibson	142
3 John Wayne	136
4 Charles Starrett	116
5 Bebe Daniels	103

At the bottom of the interface, there are copyright and version information: 'Copyright © nosqlbooster.com Version 8.1.1 Free Edition' and 'Line: 151, Column: 1 (177 selected) Show Log Feedback 9:09:44 PM'.

18. Sobre actors (nueva colección), agrupar por película y año mostrando las 5 en las que más actores hayan participado, mostrando el número total de actores.

El código de este ejercicio es el siguiente:

```

db.actors.aggregate([
  { $match: { cast: { $ne: "Undefined" } } },
  { $group: { _id: { title: "$title", year: "$year" }, cuenta: { $sum: 1 } } },
  { $sort: { cuenta: -1 } },

```

```
{ $limit: 5 }  
]);
```

Filtramos para eliminar los actores `Undefined`. Tras esto agrupamos por película y año, contando las ocurrencias (que coinciden con el número de actores que participan, pues la colección surgió de hacer un `$unwind` sobre el campo `cast`). Finalmente ordenamos de mayor a menor y nos quedamos con las primeras 5 películas. El resultado es el siguiente:

```
/* 1 */  
{  
  "_id" : {  
    "title" : "The Twilight Saga: Breaking Dawn - Part 2",  
    "year" : 2012  
  },  
  "cuenta" : 35  
},  
  
/* 2 */  
{  
  "_id" : {  
    "title" : "Anchorman 2: The Legend Continues",  
    "year" : 2013  
  },  
  "cuenta" : 33  
},  
  
/* 3 */  
{  
  "_id" : {  
    "title" : "Cars 2",  
    "year" : 2011  
  },  
  "cuenta" : 32  
},  
  
/* 4 */  
{  
  "_id" : {  
    "title" : "Avengers: Infinity War",  
    "year" : 2018  
  },  
  "cuenta" : 29  
},  
  
/* 5 */  
{  
  "_id" : {  
    "title" : "Grown Ups 2",  
    "year" : 2013  
  },
```

```
"cuenta" : 28
}
```

La captura de pantalla del ejercicio es esta:

The screenshot shows the NoSQLBooster application interface for MongoDB. The top menu bar includes File, Edit, Options, View, Favorites, Tools, Window, Help, and a Theme selector. The main window has a 'Connect' dropdown, a toolbar with Run, Debug, Stop, Import, Export, Monitoring, Tasks, DataGen, Schema, and a search icon. Below the toolbar is a tree view of databases and collections: 'localhost' (admin, config, local), 'Movies (2)' (movies, users), and 'Movies' (28.8K) (actors). The query editor contains the following code:

```

NoSQLProject.js - NoSQLBooster for MongoDB
File Edit Options View Favorites Tools Window Help
Connect Run Debug Stop Import Export Monitoring Tasks DataGen Schema
NoSQLProject.js
localhost
Movies (2)
Movies (28.8K)
Movies
142 ~ db.movies.aggregate([
143   { $unwind: "$cast" },
144   { $project: { "_id": false } },
145   { $out: "actors" }
146 ]);
147
148 db.actors.count()
149
150 //Ejercicio 17
151 db.actors.aggregate([
152   { $match: { cast: { $ne: "Undefined" } } },
153   { $group: { _id: "$cast", cuenta: { $sum: 1 } } },
154   { $sort: { cuenta: -1 } },
155   { $limit: 5
156 });
157
158 // Ejercicio 18
159 db.actors.aggregate([
160   { $match: { cast: { $ne: "Undefined" } } },
161   { $group: { _id: { title: "$title", year: "$year" }, cuenta: { $sum: 1 } } },
162   { $sort: { cuenta: -1 } },
163   { $limit: 5
164 });

```

The results table shows the following data:

	_id	cuenta	
1	The Twilight Saga: Breaking Dawn - Part 2	2012	35
2	Anchorman 2: The Legend Continues	2013	33
3	Cars 2	2011	32
4	Avengers: Infinity War	2018	29
5	Grown Ups 2	2013	28

Copyright © nosqlbooster.com Version 8.1.1 Free Edition Line: 1, Column: 1 Show Log Feedback 9:19:12 PM

19. Sobre actors (nueva colección), mostrar los 5 actores cuya carrera haya sido la más larga. Para ello, se debe mostrar cuándo comenzó su carrera, cuándo finalizó y cuántos años ha trabajado. Importante! Se necesita previamente filtrar para descartar aquellos actores llamados "Undefined".

El código de este ejercicio es el siguiente:

```

db.actors.aggregate([
  {$match: { cast: { $ne: "Undefined" } }},
  {$group: { _id: "$cast", years: { $push: "$year" } }},
  {
    $project: {
      comienza: { $min: "$years" },
      termina: { $max: "$years" },
      anos: { $subtract: [{ $max: "$years" }, { $min: "$years" }] }
    }
  },
  { $sort: { anos: -1 } },
  { $limit: 5
]);

```

Filtramos para eliminar los actores **Undefined**. Tras esto agrupamos por el nombre del actor y añadimos en una lista los años de las películas ordenados de menor a mayor. Tras esto componemos el objeto resultado con una proyección. El resultado es el siguiente:

```
/* 1 */
{
  "_id" : "Harrison Ford",
  "comienza" : 1919,
  "termina" : 2017,
  "anos" : 98
},

/* 2 */
{
  "_id" : "Gloria Stuart",
  "comienza" : 1932,
  "termina" : 2012,
  "anos" : 80
},

/* 3 */
{
  "_id" : "Kenny Baker",
  "comienza" : 1937,
  "termina" : 2012,
  "anos" : 75
},

/* 4 */
{
  "_id" : "Lillian Gish",
  "comienza" : 1912,
  "termina" : 1987,
  "anos" : 75
},

/* 5 */
{
  "_id" : "Angela Lansbury",
  "comienza" : 1944,
  "termina" : 2018,
  "anos" : 74
}
```

La captura de pantalla del ejercicio es esta:

The screenshot shows the NoSQLBooster interface for MongoDB. The left sidebar lists databases (localhost, admin, config, local) and collections (Movies, actors, movies, users). The main area displays two pieces of MongoDB code. The first is a query for 'Ejercicio 18' using the `aggregate` method on the 'actors' collection. The second is a query for 'Ejercicio 19' using the `aggregate` method on the same collection. Below the queries is a table titled 'actors' showing five rows of data. The table has columns: '_id', 'comienza', 'termina', and 'anos'. The data is as follows:

	_id	comienza	termina	anos
1	Harrison Ford	1919 (1.9K)	2017 (2.0K)	98
2	Gloria Stuart	1932 (1.9K)	2012 (2.0K)	80
3	Lillian Gish	1912 (1.9K)	1987 (2.0K)	75
4	Kenny Baker	1937 (1.9K)	2012 (2.0K)	75
5	Mickey Rooney	1932 (1.9K)	2006 (2.0K)	74

20. Sobre actors (nueva colección), Guardar en nueva colección llamada “genres” realizando la fase \$unwind por genres. Después, contar cuantos documentos existen en la nueva colección. El código de este ejercicio es el siguiente:

```
db.actors.aggregate([
  { $unwind: "$genres" },
  { $project: {"_id": false}},
  { $out: "genres"}
]);

db.genres.count()
```

El resultado de esta query es una nueva colección con 104950 documentos. La captura de pantalla del ejercicio es la siguiente:

The screenshot shows the NoSQLBooster interface for MongoDB. In the top navigation bar, 'File', 'Edit', 'Options', 'View', 'Favorites', 'Tools', 'Window', and 'Help' are visible. Below the menu is a toolbar with icons for 'Run', 'Debug', 'Stop', 'Import', 'Export', 'Monitoring', 'Tasks', 'DataGen', and 'Schema'. The 'Theme' dropdown is set to 'Light'. The 'Open Connections' section shows 'localhost' with databases 'admin', 'config', 'local', and 'Movies (4)'. The 'Movies' database has collections 'actors', 'genres', 'movies', and 'users'. The main area displays a code editor with a file named 'NoSQLProject.js' containing the following MongoDB aggregation query:

```

179     ];
180
181 // Ejercicio 20
182 db.actors.aggregate([
183     { $unwind: "$genres" },
184     { $project: {"_id": false}},
185     { $out: "genres" }
186 ]);
187
188 db.genres.count()

```

The 'Result' tab is selected, showing the output of the query. It includes a header row with '0.024 s' and a data row with '1 104950'.

At the bottom left, there's a 'My Queries' section with 'My Queries' and 'NoSQLProject.js'. At the bottom right, it says 'Copyright © nosqlbooster.com Version 8.1.1 Free Edition' and 'Line: 182, Column: 1 (127 selected) Show Log Feedback 5:28:26 PM'.

21. Sobre genres (nueva colección), mostrar los 5 documentos agrupados por “Año y Género” que más número de películas diferentes tienen mostrando el número total de películas.

El código de este ejercicio es el siguiente:

```

db.genres.aggregate([
    { $match: { genres: { $ne: "Undefined" } } },
    { $group: { _id: { year: "$year", genre: "$genres" }, uniqueTitles: { $addToSet: "$title" } } },
    { $project: { pelis: { $size: "$uniqueTitles" } } },
    { $sort: { pelis: -1 } },
    { $limit: 5 }
]);

```

En primer lugar, se descartan los documentos con género **Undefined**. Tras esto agrupamos por año y género y utilizando un **\$addToSet** metemos en una propiedad los títulos de ese año y género de forma que no haya repetidos. Finalmente con una proyección contamos el total de películas, ordenamos de mayor a menor y limitamos el número de documentos a devolver a 5. El resultado obtenido es el siguiente:

```

/* 1 */
{
    "_id" : {
        "year" : 1919,
        "genre" : "Drama"
    },
    "pelis" : 291
},
/* 2 */
{

```

```
"_id" : {  
    "year" : 1925,  
    "genre" : "Drama"  
},  
"pelis" : 247  
},  
  
/* 3 */  
{  
    "_id" : {  
        "year" : 1924,  
        "genre" : "Drama"  
    },  
    "pelis" : 233  
},  
  
/* 4 */  
{  
    "_id" : {  
        "year" : 1919,  
        "genre" : "Comedy"  
    },  
    "pelis" : 226  
},  
  
/* 5 */  
{  
    "_id" : {  
        "year" : 1922,  
        "genre" : "Drama"  
    },  
    "pelis" : 209  
}
```

La captura de pantalla del ejercicio es la siguiente:

```

NoSQLProject.js - NoSQLBooster for MongoDB
File Edit Options View Favorites Tools Window Help
Connect SQL Run Debug Stop Import Export Monitoring Tasks DataGen Schema Theme
Open Connections localhost
localhost admin config local
Movies (4) actors (83.2K) genres (0.10M) movies (28.0K) users
NoSQLProject.js
localhost / Movies
181 // Ejercicio 20
182 db.actors.aggregate([
183   { $unwind: "$genres" },
184   { $project: { "_id": false } },
185   { $out: "genres" }
186 ]);
187
188 db.genres.count()
189
190 // Ejercicio 21
191 db.genres.aggregate([
192   { $match: { genres: { $ne: "Undefined" } } },
193   { $group: { _id: { year: "$year", genre: "$genres" }, uniqueTitles: { $addToSet: "$title" } } },
194   { $project: { pelis: { $size: "$uniqueTitles" } } },
195   { $sort: { pelis: -1 } },
196   { $limit: 5 }
197 ]);

< result (1) > Find Aggregate (1) Aggregate (2) Aggregate (3) Aggregate (4) Aggregate (5) Aggregate (6) Aggregate (7) Console (1) Result (2) Find (1) Aggregate (8) >
genres 0.142 s | 5 Docs
1 /* 1 */
2 {
3   "_id": {
4     "year": 1919,
5     "genre": "Drama"
6   },
7   "pelis": 291
8 },
9
10 /* 2 */
11 {
12   "_id": {
13     "year": 1925,
14     "genre": "Drama"
15   },
16   "pelis": 247
```

```

Copyright © nosqlbooster.com Version 8.1.1 [Free Edition](#)

Line: 195, Column: 30 Show Log Feedback 6:04:15 PM

**22. Sobre genres (nueva colección), mostrar los 5 actores y los géneros en los que han participado con más número de géneros diferentes, se debe mostrar el número de géneros diferentes que ha interpretado. Importante! Se necesita previamente filtrar para descartar aquellos actores llamados "Undefined".**

El código de este ejercicio es el siguiente:

```

db.genres.aggregate([
 { $match: { cast: { $ne: "Undefined" } } },
 { $match: { genres: { $ne: "Undefined" } } },
 { $group: { _id: "$cast", generos: { $addToSet: "$genres" } } },
 { $project: { numgeneros: { $size: "$generos" }, generos: 1 } },
 { $sort: { numgeneros: -1 } },
 { $limit: 5 }
]);
```

```

En primer lugar filtramos para evitar el género y actor **Undefined**. Tras esto agrupamos por actor y creamos un conjunto de géneros para cada actor sin repetidos (con el **\$addToSet**). Tras esto hacemos una proyección mostrando el conjunto de géneros y el total de elementos en el conjunto, ordenamos de mayor a menor y limitamos a 5. El resultado obtenido es el siguiente:

```

/* 1 */
{
  "_id": "Dennis Quaid",
  "generos": [
    "Fantasy",
    "Disaster",
    "Musical",
    "Science Fiction",
```

```

```
 "Adventure",
 "Satire",
 "Sports",
 "Western",
 "Action",
 "Family",
 "Thriller",
 "Animated",
 "Dance",
 "Comedy",
 "Romance",
 "Drama",
 "Biography",
 "Suspense",
 "Horror",
 "Crime"
],
 "numgeneros" : 20
},
/* 2 */
{
 "_id" : "James Mason",
 "generos" : [
 "Adventure",
 "Fantasy",
 "Musical",
 "Science Fiction",
 "War",
 "Action",
 "Western",
 "Short",
 "Thriller",
 "Animated",
 "Noir",
 "Romance",
 "Drama",
 "Comedy",
 "Biography",
 "Suspense",
 "Mystery",
 "Crime"
],
 "numgeneros" : 18
},
/* 3 */
{
 "_id" : "James Coburn",
 "generos" : [
 "Adventure",
 "Spy",
 "Science Fiction",
 "War",
 "Romantic",
 "Thriller",
 "Drama",
 "Action",
 "Western",
 "Family",
 "Comedy",
 "Musical",
 "Short",
 "Biography",
 "Mystery",
 "Horror",
 "Crime"
],
 "numgeneros" : 15
}
```

```
"Satire",
"Sports",
>Action",
"Western",
"Family",
"Thriller",
"Animated",
"Comedy",
"Romance",
"Drama",
"Biography",
"Suspense",
"Mystery",
"Crime"
],
"numgeneros" : 18
},
/* 4 */
{
 "_id" : "Gene Hackman",
 "generos" : [
 "Disaster",
 "Adventure",
 "Science Fiction",
 "Spy",
 "War",
 "Sports",
 "Action",
 "Western",
 "Thriller",
 "Superhero",
 "Animated",
 "Comedy",
 "Noir",
 "Drama",
 "Biography",
 "Suspense",
 "Mystery",
 "Crime"
],
 "numgeneros" : 18
},
/* 5 */
{
 "_id" : "Michael Caine",
 "generos" : [
 "Disaster",
 "Crime",
 "Spy",
 "Science Fiction",
 "War",
 "Action",
```

```

 "Family",
 "Thriller",
 "Superhero",
 "Animated",
 "Comedy",
 "Romance",
 "Drama",
 "Biography",
 "Suspense",
 "Mystery",
 "Horror",
 "Adventure"
],
"numgeneros" : 18
}

```

La captura de pantalla del ejercicio es la siguiente:

The screenshot shows the NoSQLBooster interface for MongoDB. On the left, the 'Open Connections' sidebar shows a connection to 'localhost' with databases 'admin', 'config', 'local', and 'Movies'. The 'Movies' database has collections 'actors', 'genres', and 'movies'. The main workspace displays an aggregation pipeline for the 'genres' collection. The pipeline consists of two stages: a '\$match' stage where genres are undefined, and an '\$group' stage that groups by year and title, adds genres to a set, and calculates the size of the genres set. This is followed by a '\$project' stage that adds a 'numgeneros' field (size of genres), sorts by numgeneros in descending order, and limits the results to 5. The results pane shows 5 documents, each containing an '\_id' (the movie ID) and a 'generos' array. One document is highlighted, showing genres like 'Fantasy', 'Disaster', 'Musical', etc.

```

db.genres.aggregate([
 { $match: { genres: { $ne: "Undefined" } } },
 { $group: { _id: { year: "$year", title: "$title" }, generos: { $addToSet: "$genres" } } },
 { $project: { numgeneros: { $size: "$generos" }, generos: true } },
 { $sort: { numgeneros: -1 } },
 { $limit: 5 }
]);

```

**23. Sobre genres (nueva colección), mostrar las 5 películas y su año correspondiente en los que más géneros diferentes han sido catalogados, mostrando esos géneros y el número de géneros que contiene.** La query del ejercicio es la siguiente:

```

db.genres.aggregate([
{ $match: { genres: { $ne: "Undefined" } } },
{ $group: { _id: { year: "$year", title: "$title" }, generos: { $addToSet: "$genres" } } },
{ $project: { numgeneros: { $size: "$generos" }, generos: true } },
{ $sort: { numgeneros: -1 } },
{ $limit: 5 }
]);

```

En primer lugar descartamos los que tienen género `Undefined`, tras esto agrupamos por título de película y año guardando en un conjunto los géneros catalogados para dicha película sin repetidos. Tras esto hacemos una proyección mostrando además el total de elementos en el conjunto. El resultado obtenido es el siguiente:

```
/* 1 */
{
 "_id" : {
 "year" : 2017,
 "title" : "American Made"
 },
 "generos" : [
 "Action",
 "Biography",
 "Historical",
 "Comedy",
 "Thriller",
 "Drama",
 "Crime"
],
 "numgeneros" : 7
},
/* 2 */
{
 "_id" : {
 "year" : 2017,
 "title" : "Thor: Ragnarok"
 },
 "generos" : [
 "Comedy",
 "Science Fiction",
 "Superhero",
 "Action",
 "Adventure",
 "Fantasy"
],
 "numgeneros" : 6
},
/* 3 */
{
 "_id" : {
 "year" : 2017,
 "title" : "My Little Pony: The Movie"
 },
 "generos" : [
 "Fantasy",
 "Animated",
 "Musical",
 "Comedy",
 "Family",
 "Adventure"
]
}
```

```
],
 "numgeneros" : 6
},

/* 4 */
{
 "_id" : {
 "year" : 2017,
 "title" : "The Dark Tower"
 },
 "generos" : [
 "Western",
 "Science Fiction",
 "Action",
 "Adventure",
 "Fantasy",
 "Horror"
],
 "numgeneros" : 6
},

/* 5 */
{
 "_id" : {
 "year" : 2017,
 "title" : "Dunkirk"
 },
 "generos" : [
 "Action",
 "Adventure",
 "Thriller",
 "Historical",
 "Drama",
 "War"
],
 "numgeneros" : 6
}
```

La captura de pantalla del ejercicio es la siguiente:

The screenshot shows the NoSQLBooster application interface. In the top navigation bar, there are tabs for File, Edit, Options, View, Favorites, Tools, Window, and Help. Below the menu, there are buttons for Connect, Run, Debug, Import, Export, Monitoring, Tasks, DataGen, Schema, and Theme. The left sidebar shows 'Open Connections' with 'localhost' selected, displaying databases like admin, config, local, and Movies (4). The main area shows a code editor with a query for Exercise 23:

```

201 { $match: { cast: { $ne: "Undefined" } } },
202 { $match: { genres: { $ne: "Undefined" } } },
203 { $group: { _id: "$cast", generos: { $addToSet: "$genres" } } },
204 { $project: { numgeneros: { $size: "$generos" }, generos: 1 } },
205 { $sort: { numgeneros: -1 } },
206 { $limit: 5 }
207];
208
209 // Ejercicio 23
210 db.genres.aggregate([
211 { $match: { genres: { $ne: "Undefined" } } },
212 { $group: { _id: { year: "$year", title: "$title" }, generos: { $addToSet: "$genres" } } },
213 { $project: { numgeneros: { $size: "$generos" }, generos: true } },
214 { $sort: { numgeneros: -1 } },
215 { $limit: 5 }
216]);
217

```

The results pane shows the output of the query, which includes fields like '\_id', 'year', 'title', 'generos', and 'numgeneros'. The results are as follows:

```

1 /* 1 */
2 {
3 "_id": {
4 "year": 2017,
5 "title": "American Made"
6 },
7 "generos": [
8 "Action",
9 "Biography",
10 "Historical",
11 "Comedy",
12 "Thriller",
13 "Drama",
14 "Crime"
15],
16 "numgeneros": 7

```

At the bottom of the interface, there are copyright information, a free edition notice, and system status.

**24. Obtén los 3 actores que más de películas de género *Drama* han realizado.** La query de este ejercicio es la siguiente:

```

db.genres.aggregate([
 { $match: { genres: "Drama", cast: { $ne: "Undefined" } } },
 { $group: { _id: "$cast", peliculas: { $addToSet: "$title" } } },
 { $project: { numpeliculas: { $size: "$peliculas" }, peliculas: 1 } },
 { $sort: { numpeliculas: -1 } },
 { $limit: 3 }
]);

```

En primer lugar hacemos match para tomar aquellas películas de género *Drama* que no tengan como actor ningún *Undefined*. En segundo lugar agrupamos por actor y creamos un set con las películas realizadas con género *Drama*. Finalmente proyectamos por el campo de total de películas y agregamos un conteo del total de películas en el set. Ordenamos de mayor a menor por este campo y tomamos los 3 primeros.

El resultado obtenido es el siguiente:

```

/* 1 */
{
 "_id": "Bette Davis",
 "peliculas": [
 "Hell's House",
 "Bordertown",
 "Beyond the Forest",
 "Jezebel",
 "The Dark Horse",
 "That Certain Woman",
 "Storm Center",
 "Where Love Has Gone",
]
}

```

```
"Mr. Skeffington",
"Front Page Woman",
"The Cabin in the Cotton",
"Bureau of Missing Persons",
"The Petrified Forest",
"20,000 Years in Sing Sing",
"All About Eve",
"The Little Foxes",
"Old Acquaintance",
"Another Man's Poison",
"Parachute Jumper",
"Of Human Bondage",
"A Stolen Life",
"The Star",
"Satan Met a Lady",
"Three on a Match",
"The Big Shakedown",
"Phone Call from a Stranger",
"The Man Who Played God",
"The Private Lives of Elizabeth and Essex",
"Wicked Stepmother",
"Payment on Demand",
"Special Agent",
"In This Our Life",
"The Girl from 10th Avenue",
"Marked Woman",
"The Old Maid",
"Jimmy the Gent",
"Watch on the Rhine",
"The Corn Is Green",
"Dark Victory",
"The Catered Affair",
"The Sisters",
"The Whales of August",
"Dangerous",
"Fashions of 1934",
"The Scapegoat",
"Kid Galahad",
"Housewife",
"Now, Voyager",
"The Great Lie",
"Waterloo Bridge",
"Way Back Home",
"Winter Meeting",
"All This, and Heaven Too",
"Bad Sister",
"Fog Over Frisco",
"The Golden Arrow"
],
"numpeliculas" : 56
},
/* 2 */
{
```

```
"_id" : "Lionel Barrymore",
"peliculas" : [
 "The Girl Who Wouldn't Work",
 "Sadie Thompson",
 "The Show",
 "The Yellow Ticket",
 "Oil and Water",
 "The Copperhead",
 "A Man of Iron",
 "Broken Lullaby",
 "Rasputin and the Empress",
 "The Little Colonel",
 "The Secret of Dr. Kildare",
 "Three Wise Fools",
 "I Am the Man",
 "Paris at Midnight",
 "The Penalty",
 "Bannerline",
 "The Voice of Bugle Ann",
 "One Man's Journey",
 "Captains Courageous",
 "Guilty Hands",
 "The Road to Glory",
 "Night Flight",
 "The Return of Peter Grimm",
 "This Side of Heaven",
 "Saratoga",
 "Young Dr. Kildare",
 "The Eternal City",
 "3 Men in White",
 "A Free Soul",
 "The Barrier",
 "Children of the Whirlwind",
 "The Lion and the Mouse",
 "Unseeing Eyes",
 "Dr. Gillespie's Criminal Case",
 "A Family Affair",
 "West of Zanzibar",
 "The Washington Masquerade",
 "Sweepings",
 "The Gorgeous Hussy",
 "Jim the Penman",
 "Meddling Women",
 "Calling Dr. Gillespie",
 "Public Hero No. 1",
 "Camille",
 "Road House",
 "Mata Hari",
 "The Wrongdoers",
 "Ah, Wilderness!",
 "Fifty-Fifty",
 "Three Friends",
 "Carolina",
 "Dark Delusion",
```

```
"Dr. Gillespie's New Assistant",
"Looking Forward",
"Calling Dr. Kildare"
],
"numpeliculas" : 55
},

/* 3 */
{
 "_id" : "Mary Astor",
 "peliculas" : [
 "Young Ideas",
 "Upper World",
 "The Sin Ship",
 "Claudia and David",
 "A Stranger in My Arms",
 "The Lash",
 "Straight from the Heart",
 "Red Dust",
 "The Price of a Party",
 "Man of Iron",
 "Puritan Passions",
 "Other Men's Women",
 "Scarlet Saint",
 "The Pace That Thrills",
 "Inez from Hollywood",
 "Dodsworth",
 "The Woman from Hell",
 "Unguarded Women",
 "Woman Against Woman",
 "Behind Office Doors",
 "Youngblood Hawke",
 "New Year's Eve",
 "Enticement",
 "Three-Ring Marriage",
 "The Lost Squadron",
 "Playing with Souls",
 "Jennie Gerhardt",
 "The Man with Two Faces",
 "Second Fiddle",
 "Viva Villa!",
 "Act of Violence",
 "Dinky",
 "Cynthia",
 "Men of Chance",
 "Any Number Can Play",
 "Forever After",
 "Dressed to Kill",
 "Smart Woman",
 "The Bright Shawl",
 "The Runaway Bride",
 "The Great Lie",
 "Blonde Fever",
 "Romance of the Underworld",
```

```

 "Trapped by Television",
 "White Shoulders",
 "The Fighting Adventurer",
 "Those We Love",
 "I Am a Thief",
 "The World Changes",
 "High Steppers"
],
 "numpeliculas" : 50
}

```

La captura de pantalla del ejercicio es la siguiente:

The screenshot shows the NoSQLBooster interface for MongoDB. The left sidebar shows the database structure with 'localhost' selected, containing 'admin', 'config', 'local', and 'Movies'. The 'Movies' collection has subcollections 'actors' (83.2K), 'genres' (0.10M), 'movies' (28.8K), and 'users'. The main area displays an aggregation query in the 'Aggregate (3)' tab:

```

db.genres.aggregate([
 { $match: { genres: { $ne: "Undefined" } } },
 { $group: { _id: { genre: "$genres", year: "$year" }, peliculas: { $addToSet: "$title" } } },
 { $project: { numpeliculas: { $size: "$peliculas" } } },
 { $sort: { numpeliculas: -1 } },
 { $limit: 3 }
])

```

The results table shows three documents:

| _id                | peliculas | numpeliculas |
|--------------------|-----------|--------------|
| 1 Bette Davis      | Array[56] | 56           |
| 2 Lionel Barrymore | Array[55] | 55           |
| 3 Mary Astor       | Array[50] | 50           |

At the bottom, the status bar indicates 'Copyright © nosqlbooster.com Version 8.1.1 Free Edition' and 'Line: 184, Column: 2 (3736 selected) Show Log Feedback 6:04:45 PM'.

**25. Obtén para cada género el año en que más películas se hicieron de dicho género (la moda).** La query de este ejercicio es la siguiente:

```

db.genres.aggregate([
 { $match: { genres: { $ne: "Undefined" } } },
 { $group: { _id: { genre: "$genres", year: "$year" }, peliculas: { $addToSet: "$title" } } },
 { $project: { numpeliculas: { $size: "$peliculas" } } },
 { $sort: { numpeliculas: -1 } },
 { $group: { _id: "$_id.genre", anoModa: { $first: "$_id.year" }, numPelisModa: { $first: "$numpeliculas" } } },
 { $project: { _id: 0, genre: "$_id", anoModa: 1, numPelisModa: 1 } },
 { $sort: { numPelisModa: -1 } }
])

```

Primero filtramos por entradas con género distinto de "Undefined". Luego agrupamos por género y año los documentos añadiendo en un set sin elementos repetidos todas las películas de ese género y año. En una

proyección, añadimos un campo para obtener el total de películas del set. Después ordenamos de mayor a menor de acuerdo con este nuevo campo. Agrupamos esta vez por género, y en el año que representa la moda usamos `$first` para obtener el año del primer documento de cada género (que como estaba ordenado en función del número de películas sabemos que es el que más películas tiene). Finalmente proyectamos el resultado para una lectura más amigable de los documentos, mostrando el género, año de la moda y número de películas. El resultado es el siguiente:

```
/*
 1
{
 "anoModa" : 1919,
 "numPelisModa" : 291,
 "genre" : "Drama"
},
/*
 2
{
 "anoModa" : 1919,
 "numPelisModa" : 226,
 "genre" : "Comedy"
},
/*
 3
{
 "anoModa" : 1925,
 "numPelisModa" : 120,
 "genre" : "Western"
},
...

```

El resultado se puede ver en la siguiente captura:

The screenshot shows the NoSQLBooster application interface. In the top navigation bar, the title is 'NoSQLProject.js - NoSQLBooster for MongoDB'. The main window displays the results of an aggregation query. The code in the editor is:

```
227 // Ejercicio 25
228 db.genres.aggregate([
229 { $match: { genres: { $ne: "Undefined" } } },
230 { $group: { _id: { genre: "$genres", year: "$year" }, pelisModa: { $addToSet: "$title" } } },
231 { $project: { numPelisModa: { $size: "$pelisModa" } } },
232 { $sort: { numPelisModa: -1 } },
233 { $group: { _id: "$_id.genre", anoModa: { $first: "$_id.year" }, numPelisModa: { $first: "$numPelisModa" } } },
234 { $project: { _id: 0, genre: "$_id", anoModa: 1, numPelisModa: 1 } },
235 { $sort: { numPelisModa: -1 } }
236])
```

The results grid shows the following data:

| anoModa     | numPelisModa | genre       |
|-------------|--------------|-------------|
| 1919 (1.9K) | 291          | Drama       |
| 1919 (1.9K) | 226          | Comedy      |
| 1925 (1.9K) | 120          | Western     |
| 1944 (1.9K) | 67           | Musical     |
| 2006 (2.0K) | 59           | Documentary |
| 2017 (2.0K) | 55           | Action      |
| 2018 (2.0K) | 55           | Thriller    |
| 1950 (1.9K) | 53           | Noir        |
| 2006 (2.0K) | 51           | Horror      |
| 1929 (1.9K) | 49           | Romance     |
| 1935 (1.9K) | 47           | Crime       |
| 1953 (2.0K) | 46           | Adventure   |

**26. Calcula el promedio de géneros por película en cada año y ordena el resultado de mayor a menor promedio.**

La query es la siguiente:

```
db.genres.aggregate([
 { $match: { genres: { $ne: "Undefined" } } },
 { $group: { _id: { year: "$year", title: "$title" }, uniqueGenres: { $addToSet: "$genres" } } },
 { $group: { _id: "$_id.year", totalMovies: { $sum: 1 }, totalGenres: { $sum: { $size: "$uniqueGenres" } } } },
 { $project: { _id: 0, year: "$_id", averageGenresPerMovie: { $divide: ["$totalGenres", "$totalMovies"] } } },
 { $sort: { averageGenresPerMovie: -1 } }
 { $limit: 5 }
])
```

En primer lugar, filtramos por aquellos documentos con género distinto a `Undefined`. Después agrupamos por película y año, añadiendo para cada pareja película, año un set con los géneros. Tras esto agrupamos por año asumiendo el total de películas y el total de géneros. Finalmente proyectamos para una salida más legible y calculamos el promedio de géneros, ordenamos por el promedio de géneros de mayora a menor y nos quedamos con los primeros 5 valores.

El resultado obtenido es el siguiente:

```
/* 1 */
{
 "year" : 2017,
 "promedioGenresPerMovie" : 2.2941176470588234
},

/* 2 */
{
 "year" : 2018,
 "promedioGenresPerMovie" : 2.01271186440678
},

/* 3 */
{
 "year" : 1906,
 "promedioGenresPerMovie" : 1.8571428571428572
},

/* 4 */
{
 "year" : 1902,
 "promedioGenresPerMovie" : 1.75
},

/* 5 */
```

```
{
 "year" : 2011,
 "promedioGenresPerMovie" : 1.7313432835820894
}
```

Como podemos ver el año con mayor número de géneros por película fue 2017 con más de 2 géneros por película de media.

La captura de pantalla de este ejercicio se muestra a continuación:

El resultado se puede ver en la siguiente captura:

The screenshot shows the NoSQLBooster interface for MongoDB. On the left, the database structure is visible with 'Movies' selected. In the center, the code editor contains an aggregation query (line 233 to 247) for exercise 26. The results pane shows a table with one row, indicating the year 2017 has a 'promedioGenresPerMovie' of 2.2941176470588234. The bottom status bar shows the line and column numbers (Line: 247, Column: 1), the version (Version 8.1.1), and a link to the free edition.

| Key                | Value                                                    | Type   |
|--------------------|----------------------------------------------------------|--------|
| 2017               | {year: 2017, promedioGenresPerMovie: 2.2941176470588234} | Object |
| 2.2941176470588234 |                                                          | Double |