Bases de datos NoSQL

Hugo Gómez Sabucedo
hugogomezsabucedo@gmail.com

Máster Big Data, Data Science & Inteligencia Artificial

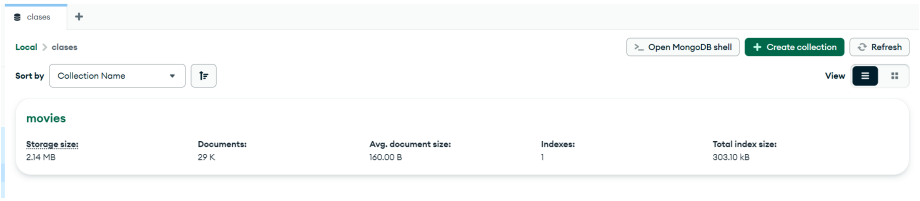
Curso 2024-2025

Universidad Complutense de Madrid

1. Ejercicios

A continuación se incluyen las capturas con las querys y los resultados de las ejecuciones de las consultas de los diferentes ejercicios, así como una breve explicación de las mismas (si es necesario). He incluido también el archivo `JavaScript` con todas las consultas al final del archivo y como adjunto.

0. **Importación del fichero en una colección llamada movies**: se ha creado la colección movies dentro de la base de datos *clases* e importado los datos usando la funcionalidad "Add data" de MongoDBCompass, importando los datos directamente desde el archivo JSON.



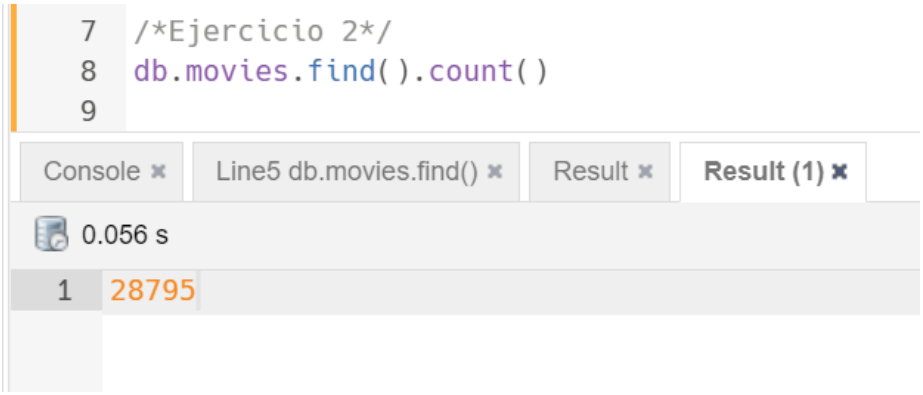
1. **Analizar con find la colección**.

```
4  /*Ejercicio 1*/
5  db.movies.find()
6
```

movies 0.124 s 28.795 Docs

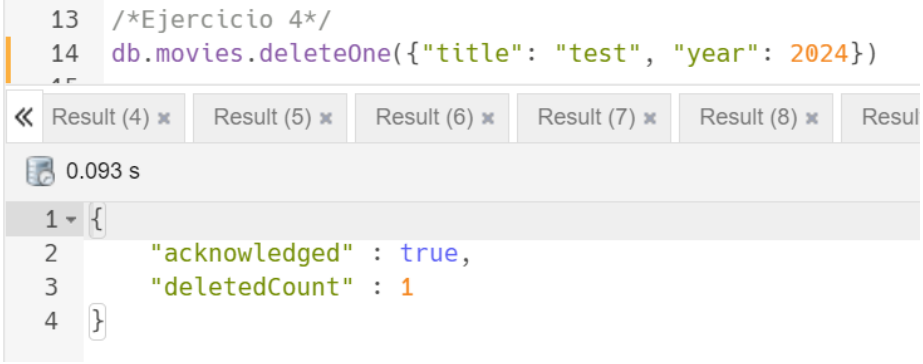
```
1  /* 1 createdAt:1/12/2024 17:35:16*/
2  {
3      "_id" : ObjectId("674c9044944caa30e978feb1"),
4      "title" : "Caught",
5      "year" : 1900,
6      "cast" : [ ],
7      "genres" : [ ]
8  },
9
10 /* 2 createdAt:1/12/2024 17:35:16*/
11 {
12     "_id" : ObjectId("674c9044944caa30e978feb2"),
13     "title" : "After Dark in Central Park",
14     "year" : 1900,
15     "cast" : [ ],
16     "genres" : [ ]
17 },
18
19 /* 3 createdAt:1/12/2024 17:35:16*/
20 {
21     "_id" : ObjectId("674c9044944caa30e978feb3"),
22     "title" : "Buffalo Bill's Wild West Parad",
23     "year" : 1900,
24     "cast" : [ ],
25     "genres" : [ ]
26 },
```

2. **Contar cuantos documentos (películas) tiene cargado**.



3. **Insertar una película**. Se crea una película *test* para insertar.

4. **Borrar la película insertada en el punto anterior**. Como la película se llama *text* y es del año 2024 (más adelante se comprobará que no hay ninguna película de este año), se borra en base a esto. Borrar en base al id no es efectivo, ya que si se reejecuta la query este cambiará.



5. **Contar cuantas películas tienen actores que se llaman "and"**.

```
16  /*Ejercicio 5*/
17  db.movies.find({ cast: "and" }).count()
18
19  /*Ejercicio 6*/
```

Result (16) × | Result (17) × | Find × | Error (1) × | Error (2) ×

0.099 s

```
1  93
```

6. **Actualizar los documentos cuyo actor tenga el valor "and", sacando ese valor del array cast**. Usamos `updateMany` porque se actualizarán varios documentos a la vez. Se busca, como antes, las que tengan "and", y se eliminan estos del array con `$pull`.

```
19  /*Ejercicio 6*/
20  db.movies.updateMany({ cast: "and" }, { $pull: { cast: "and" } })
```

2) × | Error (3) × | Error (4) × | Error (5) × | Error (6) × | Error (7) × | Find (1) × | Error (8) ×

0.115 s

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

7. **Contar cuantos documentos tienen el array "cast" vacío**.

```
22 /*Ejercicio 7*/
23 db.movies.find({ cast: [] }).count()
24
```

Error (6) × Error (7) × Find (1) × Error (8) ×

0.091 s

```
1 986
```

8. **Actualizar todos los documentos que tengan el array cast vacío, añadiendo un nuevo elemento con el valor Undefined**. Con el `updateMany`, actualizamos el campo cast al array *["Undefined"]*, para asegurarnos que se sigue manteniendo el tipo del mismo.

```
25 /*Ejercicio 8*/
26 db.movies.updateMany({ cast: [] }, {$set: {"cast": ["Undefined"]} })
27
```

(3) × Find (4) × Find (5) × Result (18) × Result (19) × Result (20) × Find (6) × Resul

0.139 s

```
1 {
2     "acknowledged" : true,
3     "matchedCount" : 986,
4     "modifiedCount" : 986
5 }
```

9. **Contar cuantos documentos tienen el array genres vacío**.

```
28 /*Ejercicio 9*/
29 db.movies.find({ genres: [] }).count()
```

Result (18) × Result (19) × Result (20) × Find (6) ×

0.082 s

```
1 901
```

10. **Actualizar todos los documentos que tengan el array genres vacío, añadiendo un nuevo elemento con el valor Undefined**. Hacemos lo mismo que en el caso del cast.

```
31 /*Ejercicio 10*/
32 db.movies.updateMany({ genres: [] }, {$set: {"genres": ["Undefined"]} })
```

Find (6) × Result (21) × Find (7) × Find (8) × Result (22) × Find (9) × Result (23) ×

0.122 s

```
1 {
2     "acknowledged" : true,
3     "matchedCount" : 901,
4     "modifiedCount" : 901
5 }
```

11. **Mostrar el año más reciente/actual que tenemos sobre todas las películas**. Buscamos solo los años, lo ordenamos de forma descendente y limitamos a 1 el output.

```
34 /*Ejercicio 11*/
35 db.movies.find({}, {year: 1, "_id": 0}).sort({year: -1}).limit(1)
```

Find (9) × Result (23) × Find (10) × Result (24) × Result (25) × Result (26) ×

movies 0.104 s 1 Doc

```
1 {
2     "year" : 2018
3 }
```

12. **Contar cuantas películas han salido en los últimos 20 años, desde el último año que se tienen películas en la colección**. Primero, buscamos el año más reciente. Luego, le restamos 20. Como las variables minYear y maxYear se definen fuera, las tenemos que pasar al pipeline usando `let`. A continuación hacemos un match y, como queremos comparar los campos del mismo documento, usamos el `$expr`. Ya que restamos 20 al año para hallar el inicio del intervalo, debemos indicar que sea *greater than* (para que no lo coja), mientras que para el fin debemos indicar *less than or equal*. Con `$size`, contamos el número de películas, y seleccionamos para el output el id y este total.

```
/*Ejercicio 12*/
db.movies.aggregate([
    { // Encontrar el año mas reciente
        $group: {
            _id: null,
            maxYear: { $max: "$year" }
        }
    },
    { // Encontrar el inicio del intervalo
        $addFields: {
            minYear: { $subtract: ["$maxYear", 20] }
        }
    },
    { // Obtenemos las peliculas en el intervalo
        $lookup: {
            from: "movies",
            let: { minYear: "$minYear", maxYear: "$maxYear" }, // "Pasamos" las variables al l
            pipeline: [
                {
                    $match: {
                        $expr: {
                            $and: [
                                { $gt: ["$year", "$$minYear"] },
                                { $lte: ["$year", "$$maxYear"] }
                            ]
                        }
                    }
                }
            ],
            as: "peliculasIntervalo"
        }
    },
    { // Añadimos un campo con el total de las películas
        $addFields: {
            total: { $size: "$peliculasIntervalo" }
        }
    },
    { //Seleccionamos solo los campos de interes
        $project: {
            _id: 1,
            total: 1
        }
    }
]);
```

Result × | Aggregate × | Result (1) × | Aggregate (1) × | Aggregate (2) × | Aggregate (3) × | Aggregate (4) × | **Aggregate (5) ×**

movies | 0.219 s | 1 Doc

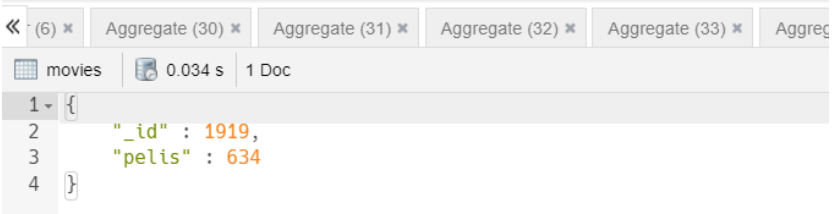
```
[{
    "_id": "",
    "total": 4787
}]
```

13. **Contar cuantas películas han salido en la década de los 60**. En este caso, como es un rango fijo, simplemente hacemos un `$match`, incluyendo los años límite del intervalo, y calculamos el total como `$sum: 1` (como 1 es siempre true, esta es una buena forma también de contar todos los documentos).

```
/*Ejercicio 13*/
db.movies.aggregate([
    { // Obtenemos las peliculas en el intervalo
        $match: {
            year: { $gte: 1960, $lte: 1969 }
        }
    },
    { // Contamos el total de peliculas
        $group: {
            _id: null,
            total: { $sum: 1 }
        }
    }
])
```

14. **Mostrar el año/años con más películas**. En primer lugar, agrupamos por año y contamos las películas de cada año. Luego, con otro `$group`, encontramos el máximo número. Nos creamos un array con todos los años y el total de películas y, adicionalmente, el máximo. Separamos "years" en documentos individuales y, sobre esto, filtramos los que tengan el total igual al máximo, devolviendo los años como id y el total. Crear un array con los años y las películas nos sirve para manejar el caso en el que haya varios años con el máximo de películas, ya que se nos devolverían dichos años.

```
/*Ejercicio 14*/
db.movies.aggregate([
    { // Agrupamos por año y contamos las películas por año
        $group: {
            _id: "$year",
            pelis: { $sum: 1 }
        }
    },
    { // Encontramos el máximo
        $group: {
            _id: null,
            maxPelis: { $max: "$pelis" },
            years:
            {  // Guardamos los años y su total de peliculas
                $push: {
                    year: "$_id",
                    pelis: "$pelis"
                }
            }
        }
    },
    { // Obtenemos un documento por año
        $unwind: "$years"
    },
    { // Filtramos por el año con el máximo de películas
        $match: {
            $expr: { $eq: ["$years.pelis", "$maxPelis"] }
        }
    },
    { // Ajustamos el reesultado
        $project: {
            _id: "$years.year",
            pelis: "$years.pelis"
        }
    }
])
```



15. **Mostrar el año/años con menos películas**. Procedemos igual que en el caso anterior, pero esta vez hallando el mínimo.

```
/*Ejercicio 15*/
db.movies.aggregate([
    { // Agrupamos por año y contamos las películas por año
        $group: {
            _id: "$year",
            pelis: { $sum: 1 }
        }
    },
    { // Encontramos el mínimo
        $group: {
            _id: null,
            minPelis: { $min: "$pelis" },
            years:
            {  // Guardamos los años y su total de peliculas
                $push: { year: "$_id", pelis: "$pelis" }
            }
        }
    },
    { // Obtenemos un documento por año
        $unwind: "$years"
    },
    { // Filtramos por el año con el mínimo de películas
        $match: { $expr: { $eq: ["$years.pelis", "$minPelis"] } }
    },
    { // Ajustamos el reesultado
        $project: {
            _id: "$years.year",
            pelis: "$years.pelis"
        }
    }
]);
```

1) × | Aggregate (1) × | Aggregate (2) × | Aggregate (3) × | Aggregate (4) × | Aggregate (5) ×

movies | 0.130 s | 3 Docs

```
[{
        "_id": 1907,
        "pelis": 7
    },
    {
        "_id": 1906,
        "pelis": 7
    },
    {
        "_id": 1902,
        "pelis": 7
    }
]
```

16. **Guardar en una nueva colección llamada "actors", haciendo $unwind por actor. Contar cuantos elementos existen en la nueva colección**. Hacemos el unwind indicado, eliminamos el campo id (para que no este no se detecte como el campo id a usar por MongoDB y nos de error por tener IDs duplicados) y guardamos en la colección indicada. Luego se hace un count para contar el número de documentos que tiene la colección.

```
166 /*Ejercicio 16*/
167 db.movies.aggregate([
168     { // Hacemos el unwind
169         $unwind: "$cast"
170     },
171     { // Eliminamos el id
172         $project: { _id: 0 }
173     },
174     { // Guardamos en la coleccion actors
175         $out: "actors"
176     }
177 ]);
178 db.actors.find().count();
179
```

ovies.aggregate(... × | Aggregate × | Find × | Result × | Result

0.068 s

```
1 83224
```

17. **Sobre actors, mostrar la lista con los 5 actores que han participado en más películas, mostrando el número de películas en las que ha participado**. Primero excluimos los actores Undefined. Agrupamos por actor, contamos sus apariciones, ordenamos por este nuevo campo de forma descendente, y nos quedamos con los cinco primeros resultados.

```
/*Ejercicio 17*/
db.actors.aggregate([
    { // Excluimos los actores llamados Undefined
        $match: {
            cast: { $ne: "Undefined" }
        }
    },
    { // Agrupamos por actor y contamos sus apariciones
        $group: {
            _id: "$cast",
            cuenta: { $sum: 1 }
        }
    },
    { // Ordenamos por el total de apariciones de forma descendente
        $sort: { cuenta: -1 }
    },
    { // Limitamos a 5 el output
        $limit: 5
    }
])
```

Result (8) × | Find (1) × | Aggregate (1) × | Aggregate (2) × | Aggregate (3) × | Aggregate (4) ×

actors | 0.235 s | 5 Docs

```
[{
        "_id": "Harold Lloyd",
        "cuenta": 190
    },
    {
        "_id": "Hoot Gibson",
        "cuenta": 142
    },
    {
        "_id": "John Wayne",
        "cuenta": 136
    },
    {
        "_id": "Charles Starrett",
        "cuenta": 116
    },
    {
        "_id": "Bebe Daniels",
        "cuenta": 103
    }
]
```

18. **Sobre actors, agrupar por película y año, mostrando las 5 en las que más actores hayan participado, mostrando el número total de actores**. Realizamos la agrupación, contamos el total de actores, ordenamos por el total de forma descendente y nos quedamos con los cinco primeros.

```
/*Ejercicio 18*/
db.actors.aggregate([
    { // Agrupamos por película y año y contamos el total de actores
        $group: {
            _id: {
                title: "$title", year: "$year"
            },
            cuenta: { $sum: 1 }
        }
    },
    { // Ordenamos por el total de actores de forma descendente
        $sort: { cuenta: -1 }
    },
    { // Limitamos a 5 el output
        $limit: 5
    }
])
```

actors 0.403 s 5 Docs


```
[{
    "_id": {
        "title": "The Twilight Saga: Breaking Dawn - Part 2",
        "year": 2012
    },
    "cuenta": 35
},
{
    "_id": {
        "title": "Anchorman 2: The Legend Continues",
        "year": 2013
    },
    "cuenta": 33
},
{
    "_id": {
        "title": "Cars 2",
        "year": 2011
    },
    "cuenta": 32
},
{
    "_id": {
        "title": "Avengers: Infinity War",
        "year": 2018
    },
    "cuenta": 29
},
{
    "_id": {
        "title": "Grown Ups 2",
        "year": 2013
    },
    "cuenta": 28
}
]
```

19. **Sobre actors, mostrar los 5 actores cuya carrera haya sido la más larga, mostrando cuando comenzó, cuando finalizó y cuántos años ha trabajado**. Se agrupa por actor y se halla la fecha de inicio y de fin como el año mínimo y máximo. Se añade un campo con los años trabajados, haciendo un `$subtract`. Se ordena por este campo y se limita a cinco el resultado.

```
/*Ejercicio 19*/
db.actors.aggregate([
    { // Excluimos los actores undefined
        $match: {
            cast: { $ne: "Undefined" }
        }
    },
    { // Agrupamos por actor y hallamos la fecha de inicio y fin
        $group: {
            _id: "$cast",
            comienza: { $min: "$year" },
            termina: { $max: "$year" }
        }
    },
    { // Añadimos un campo con los años trabajados
        $addFields: { anos: {$subtract: ["$termina", "$comienza"]} }
    },
    { // Ordenamos por la duración descendente
        $sort: { anos: -1 }
    },
    { // Nos quedamos con el máximo
        $limit: 5
    }
]);
```

Result × Aggregate ×

actors | 0.177 s | 5 Docs

	_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	Angela Lansbury	1944 (1.9K)	2018 (2.0K)	74

20. **Sobre actors, guardar en una nueva colección llamada "genres" realizando la fase $unwind por genres. Contar cuantos elementos existen en al nueva colección**. Hacemos el unwind indicado, eliminamos el campo id y guardamos en la colección.

```
244 /*Ejercicio 20*/
245 db.actors.aggregate([
246     { // Hacemos el unwind
247         $unwind: "$genres"
248     },
249     { // Eliminamos el id
250         $project: { _id: 0 }
251     },
252     { // Guardamos en la colección genres
253         $out: "genres"
254 ]);
255 db.genres.count()
```

Aggregate (15) × Aggregate (16) × Aggregate (17) × Explain ×

0.056 s

```
1 104950
```

21. **Sobre genres, mostrar los 5 documentos agrupados por Año y Género que más número de películas diferentes tienen, mostrando el total de películas**. Agrupamos por año y por género, y usamos el `$addToSet` para hallar las películas únicas (ya que un set se compone de elementos únicos, sin duplicados). Sobre este set, contamos las películas, ordenamos de forma descendente y limitamos a 5 el output.

```
/*Ejercicio 21*/
db.genres.aggregate([
    { // Agrupamos por año y género y hallamos las películas distintas
        $group: {
            _id: { year: "$year", genre: "$genres" },
            pelisUnicas: { $addToSet: "$title" }
        }
    },
    { // Añadimos el contador de pelis unicas
        $project:{
            _id: 1,
            pelis: { $size: "$pelisUnicas" }
        }
    },
    { // Ordenamos por el contador descendente
        $sort: { pelis: -1 }
    },
    { // Nos quedamos con los cinco primeros
        $limit: 5
    }
])

```

genres 0.272 s 5 Docs


```
[{
        "_id": {
            "year": 1919,
            "genre": "Drama"
        },
        "pelis": 291
    },
    {
        "_id": {
            "year": 1925,
            "genre": "Drama"
        },
        "pelis": 247
    },
    {
        "_id": {
            "year": 1924,
            "genre": "Drama"
        },
        "pelis": 233
    },
    {
        "_id": {
            "year": 1919,
            "genre": "Comedy"
        },
        "pelis": 226
    },
    {
        "_id": {
            "year": 1922,
            "genre": "Drama"
        },
        "pelis": 209
    }
]
```

22. **Sobre genres, mostrar los 5 actores y lo géneros en los que han participado con más número de géneros diferentes, mostrando el número de géneros**. Se agrupa por actor y, como antes, se añaden los géneros a un set. Se cuentan, se ordena y se limita a 5 el output.

```
/*Ejercicio 22*/
db.genres.aggregate([
    { // Eliminamos los actores "Undefined"
        $match: {
            cast: { $ne: "Undefined" }
        }
    },
    { // Agrupamos por actor y hallamos los géneros distintos
        $group: {
            _id: "$cast",
            generos: { $addToSet: "$genres" }
        }
    },
    { // Añadimos el número de géneros
        $addFields: { numgeneros: { $size: "$generos" } }
    },
    { // Ordenamos por el numero de generos descendente
        $sort: { numgeneros: -1 }
    },
    { // Nos quedamos con los cinco primeros
        $limit: 5
    }
])
```


```
[{
    "_id": "Dennis Quaid",
    "generos": [
        "Fantasy",
        "Adventure",
        "Comedy",
        "Family",
        "Science Fiction",
        "Sports",
        "Dance",
        "Drama",
        "Animated",
        "Satire",
        "Suspense",
        "Action",
        "Crime",
        "Horror",
        "Western",
        "Musical",
        "Romance",
        "Disaster",
        "Thriller",
        "Biography"
    ],
    "numgeneros": 20
},
{
    "_id": "James Mason",
    "generos": [
        "Romance",
        "Science Fiction",
        "Adventure",
        "Comedy",
        "Short",
        "Animated",
        "Suspense",
        "Drama",
        "Noir",
        "Western",
        "Musical",
        "Crime",
        "Mystery",
        "Undefined",
        "Action",
        "Biography",
        "Fantasy",
        "War",
        "Thriller"
    ],
    "numgeneros": 19
},
{
    "_id": "Michael Caine",
    "generos": [
        "Spy",
        "Adventure",
        "Comedy",
        "Family",
        "Science Fiction",
        "Drama",
        "Animated",
        "Suspense",
        "Horror",
        "Crime",
        "Undefined",
        "Action",
        "Mystery",
        "Superhero",
        "Disaster",
        "Romance",
        "War",
        "Thriller",
        "Biography"
    ],
    "numgeneros": 19
},
{
    "_id": "James Coburn",
    "generos": [
        "War",
        "Science Fiction",
        "Sports",
        "Adventure",
        "Family",
        "Spy",
        "Comedy",
        "Animated",
        "Satire",
        "Suspense",
        "Drama",
        "Western",
        "Mystery",
        "Crime",
        "Action",
        "Biography",
        "Romance",
        "Thriller"
    ],
    "numgeneros": 18
},
{
    "_id": "Lionel Barrymore",
    "generos": [
        "Adventure",
        "Comedy",
        "Science Fiction",
        "Sports",
        "Drama",
        "Historical",
        "Noir",
        "Horror",
        "Undefined",
        "Mystery",
        "Crime",
        "Western",
        "Musical",
        "Romance",
        "War",
        "Thriller",
        "Biography",
        "Fantasy"
    ],
    "numgeneros": 18
}
]
```

23. **Sobre genres, mostrar las 5 películas y su año, en los que más géneros diferentes han sido catalogados, mostrando esos géneros y el número**. Se agrupa por título y año y se hallan los géneros diferentes añadiendo a un set. Se cuenta, se ordena y se limita el output.

```
/*Ejercicio 23*/
db.genres.aggregate([
    { // Agrupamos por título y año y hallamos los géneros distintos
        $group: {
            _id: { title: "$title", year: "$year" },
            generos: { $addToSet: "$genres" }
        }
    },
    { // Añadimos el número de géneros
        $addFields: { numgeneros: { $size: "$generos" } }
    },
    { // Ordenamos por el numero de generos descendente
        $sort: { numgeneros: -1 }
    },
    { // Nos quedamos con los cinco primeros
        $limit: 5
    }
])
```


```
[{
        "_id": {
            "title": "American Made",
            "year": 2017
        },
        "generos": [
            "Drama",
            "Historical",
            "Crime",
            "Action",
            "Comedy",
            "Thriller",
            "Biography"
        ],
        "numgeneros": 7
    },
    {
        "_id": {
            "title": "The Dark Tower",
            "year": 2017
        },
        "generos": [
            "Western",
            "Horror",
            "Action",
            "Fantasy",
            "Adventure",
            "Science Fiction"
        ],
        "numgeneros": 6
    },
    {
        "_id": {
            "title": "Wonder Woman",
            "year": 2017
        },
        "generos": [
            "War",
            "Drama",
            "Superhero",
            "Adventure",
            "Fantasy",
            "Action"
        ],
        "numgeneros": 6
    },
    {
        "_id": {
            "title": "Thor: Ragnarok",
            "year": 2017
        },
        "generos": [
            "Science Fiction",
            "Comedy",
            "Adventure",
            "Action",
            "Superhero",
            "Fantasy"
        ],
        "numgeneros": 6
    },
    {
        "_id": {
            "title": "Dunkirk",
            "year": 2017
        },
        "generos": [
            "War",
            "Drama",
            "Historical",
            "Action",
            "Adventure",
            "Thriller"
        ],
        "numgeneros": 6
    }
]
```

24. **Ejercicio libre. Mostrar la películas con el reparto más grande, es decir, con el mayor número de actores. Mostrar la película, el año, los actores y el tamaño del cast**. Para este ejercicio, se añade un campo con el tamaño del cast, para ordenar de forma descendente, limitar a 1 el output y eliminar el id de la misma del output.

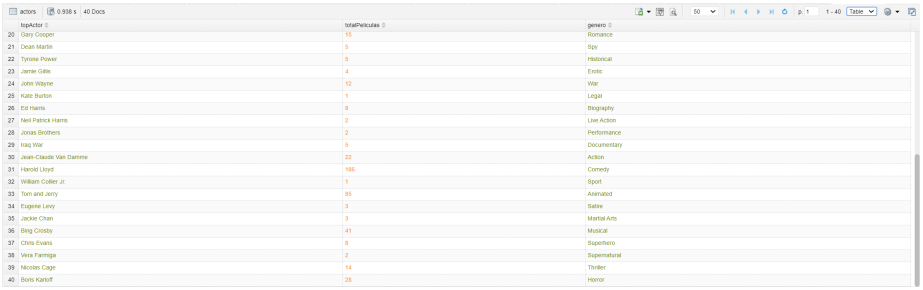
```
/*Ejercicio 24 - Película con el reparto más grande */
db.movies.aggregate([
    { // Añadimos un campo con el tamaño del cast
        $addFields: { castSize: { $size: "$cast" } }
    },
    { // ORdenamos por el tamaño del cast
        $sort: { castSize: -1 }
    },
    { // Limitamos a uno el resultado
        $limit: 1
    },
    { // Eliminamos el id del output y los géneros
        $project: { _id: 0, genres: 0 }
    }
]);
```


```
[{
    "title": "The Twilight Saga: Breaking Dawn - Part 2",
    "year": 2012,
    "cast": [
        "Kristen Stewart",
        "Robert Pattinson",
        "Taylor Lautner",
        "Nikki Reed",
        "Peter Facinelli",
        "Elizabeth Reaser",
        "Ashley Greene",
        "Kellan Lutz",
        "Jackson Rathbone",
        "Julia Jones",
        "Booboo Stewart",
        "Billy Burke",
        "Sarah Clarke",
        "MyAnna Buring",
        "Maggie Grace",
        "Casey LaBow",
        "Michael Sheen",
        "Jamie Campbell Bower",
        "Christopher Heyerdahl",
        "Chaske Spencer",
        "Christian Camargo",
        "Mía Maestro",
        "Mackenzie Foy",
        "Dakota Fanning",
        "Cameron Bright",
        "Charlie Bewley",
        "Daniel Cudmore",
        "Noel Fisher",
        "Guri Weinberg",
        "Lee Pace",
        "Joe Anderson",
        "Judi Shekoni",
        "Tracey Huggins",
        "J. D. Pardo",
        "Rami Malek"
    ],
    "castSize": 35
}]
```


25. **Ejercicio libre. Sobre actors, mostrar el actor más popular en cada género (es decir, el que más veces ha actuado)**. Primero, excluimos los actores y géneros que sean Undefined. Se hace un unwind de los géneros, para desglosar; se agrupa por género y actor y se cuentan las películas. Se ordena por género, y por el total de películas (de forma descendente), y volvemos a agrupar por género, para obtener el primer actor, que se corresponderá con el actor más popular. Se seleccionan los campos de interés.

```
/*Ejercicio 25 - Actor más popular por género */
db.actors.aggregate([
    { // Excluimos los actores y géneros Undefined
        $match: {
            $and: [ { genres: { $ne: "Undefined"} }, { cast: { $ne: "Undefined"} } ]
        }
    },
    { // Hacemos un unwind de los generos
        $unwind: "$genres"
    },
    { // Agrupamos por genero y actor y contamos
        $group: {
            _id: { genero: "$genres", actor: "$cast" },
            totalPeliculas: { $sum: 1 }
        }
    },
    { // Ordenamos por género y total de películas
        $sort: { "_id.genre": 1, totalPeliculas: -1 }
    },
    { // Agrupamos por genero y seleccionamos el actor con mas peliculas
        $group: {
            _id: "$_id.genero",
            topActor: { $first: "$_id.actor" },
            totalPeliculas: { $first: "$totalPeliculas" }
        }
    },
    {
        $project: {
            _id: 0,
            genero: "$_id",
            topActor: 1,
            totalPeliculas: 1
        }
    }
]);
```

	topActor	totalPeliculas	genero
1	Bette Davis	56	Drama
2	Adam G. Sevani	3	Dance
3	William Garwood	1	Silent
4	Cary Grant	4	Suspense
5	Kane Hodder	2	Slasher
6	Hoot Gibson	131	Western
7	Emma Watson	9	Fantasy
8	Bonnie Hunt	7	Family
9	The Three Stooges	64	Short
10	Elyse Knox	4	Sports
11	George Kennedy	4	Disaster
12	Leonard Nimoy	9	Science Fiction
13	Edward G. Robinson	14	Noir
14	Tallie Medel	2	Teen
15	Warner Oland	16	Mystery
16	George Clooney	3	Political
17	Gregor Collins	1	Independent
18	Johnny Weissmuller	29	Adventure
19	Edward G. Robinson	21	Crime
20	Gary Cooper	15	Romance



	topActor	totalPeliculas	genero
20	Gary Cooper	15	Romance
21	Dean Martin	5	Spy
22	Tyrone Power	5	Historical
23	Jamie Gillis	4	Erotic
24	John Wayne	12	War
25	Kate Burton	1	Legal
26	Ed Harris	6	Biography
27	Neil Patrick Harris	2	Live Action
28	Jonas Brothers	2	Performance
29	Iraq War	5	Documentary
30	Jean-Claude Van Damme	22	Action
31	Harold Lloyd	186	Comedy
32	William Collier Jr.	1	Sport
33	Tom and Jerry	85	Animated
34	Eugene Levy	3	Satire
35	Jackie Chan	3	Martial Arts
36	Bing Crosby	41	Musical
37	Chris Evans	8	Superhero
38	Vera Farmiga	2	Supernatural
39	Nicolas Cage	14	Thriller
40	Boris Karloff	28	Horror

26. **Ejercicio libre. Sobre genres, mostrar el génenro más popular en cada década, mostrando su nombre y el total de películas**. Primero, se exluyen los géneros Undefined. Se usa el addFields para añadir un campo con la década, que se calcula restando al año de la película el resultado de hacer el módulo (con `$mod`) 10 del año (es decir, restamos las unidades de los años, para quedarnos con las decenas). Se agrupa por década y género y se procede de forma similar al anterior: se calcula el total, se ordena por década y total, se agrupa por década para obtener el primer género, y luego se ordena el resultado por década de forma ascendente.

```
/*Ejercicio 26 - Género más popular por década */
db.genres.aggregate([
    { // Excluimos los géneros que sean Undefined
        $match: { genres: { $ne: "Undefined"} }
    },
    { // Calculamos la década, usando la función $mod
        $addFields: {
            decada: { $subtract: ["$year", { $mod: [ "$year", 10 ] } ] }
        }
    },
    { // Agrupamos por década y género y calculamos el total
        $group: {
            _id: { decada: "$decada", genero: "$genres" },
            total: { $sum: 1 }
        }
    },
    { // Ordenamos por década, de forma ascendente, y luego por el total de géneros
        $sort: { "_id.decada": 1, total: -1 }
    },
    { // Agrupamos por década para obtener el género más popular, y el conteo de películas
        $group: {
            _id: "$_id.decada",
            generoMasPopular: { $first: "$_id.genero" },
            totalPeliculas: { $first: "$total" }
        }
    },
    { // Ordenamos por década de manera ascendente
        $sort: { "_id": 1}
    }
])
```

genres 0.517 s 12 Docs

	_id	generoMasPopular	totalPeliculas
1	1900	Comedy	22
2	1910	Drama	1256 (1.3K)
3	1920	Drama	3347 (3.3K)
4	1930	Drama	4292 (4.3K)
5	1940	Comedy	2175 (2.2K)
6	1950	Drama	1815 (1.8K)
7	1960	Drama	1330 (1.3K)
8	1970	Drama	1308 (1.3K)
9	1980	Comedy	2038 (2.0K)
10	1990	Drama	2933 (2.9K)
11	2000	Comedy	2814 (2.8K)
12	2010	Comedy	4102 (4.1K)

2. Querys de los ejercicios

```
use clases
db.movies

/*Ejercicio 1*/
db.movies.find();

/*Ejercicio 2*/
db.movies.find().count();

/*Ejercicio 3*/
db.movies.insertOne({"title": "test", "year": 2024, "cast":
["Actor1", "Actor2"], "genres": ["Comedy"]});

/*Ejercicio 4*/
db.movies.deleteOne({"title": "test", "year": 2024});

/*Ejercicio 5*/
db.movies.find({ cast: "and" }).count();

/*Ejercicio 6*/
db.movies.updateMany({ cast: "and" }, { $pull: { cast: "and"
 } });

/*Ejercicio 7*/
db.movies.find({ cast: [] }).count();

/*Ejercicio 8*/
db.movies.updateMany({ cast: [] }, {$set: {"cast": ["
Undefined"]} });

/*Ejercicio 9*/
db.movies.find({ genres: [] }).count();

/*Ejercicio 10*/
db.movies.updateMany({ genres: [] }, {$set: {"genres": ["
Undefined"]} });

/*Ejercicio 11*/
db.movies.find({}, {year: 1, "_id": 0}).sort({year: -1}).
limit(1);

/*Ejercicio 12*/
db.movies.aggregate([
    { // Encontrar el anho mas reciente
        $group: {
            _id: null,
            maxYear: { $max: "$year" }
        }
    },
```

```
45      { // Encontrar el inicio del intervalo
46          $addFields: {
47              minYear: { $subtract: ["$maxYear", 20] }
48          }
49      },
50      { // Obtenemos las peliculas en el intervalo
51          $lookup: {
52              from: "movies",
53              let: { minYear: "$minYear", maxYear: "$maxYear"
                }, // "Pasamos" las variables al lookup
54              pipeline: [
55                  {
56                      $match: {
57                          $expr: {
58                              $and: [
59                                  { $gt: ["$year", "$$minYear"
                                    ] },
60                                  { $lte: ["$year", "$$maxYear
                                    "] }
61                              ]
62                          }
63                      }
64                  }
65              ],
66              as: "peliculasIntervalo"
67          }
68      },
69      { // Anhadimos un campo con el total de las peliculas
70          $addFields: {
71              total: { $size: "$peliculasIntervalo" }
72          }
73      },
74      { //Seleccionamos solo los campos de interes
75          $project: {
76              _id: 1,
77              total: 1
78          }
79      }
80  ]);
81
82  /*Ejercicio 13*/
83  db.movies.aggregate([
84      { // Obtenemos las peliculas en el intervalo
85          $match: {
86              year: { $gte: 1960, $lte: 1969 }
87          }
88      },
89      { // Contamos el total de peliculas
90          $group: {
91              _id: null,
92              total: { $sum: 1 }
```

```
        }
    }
]);

/*Ejercicio 14*/
db.movies.aggregate([
    { // Agrupamos por anho y contamos las peliculas por anho
        $group: {
            _id: "$year",
            pelis: { $sum: 1 }
        }
    },
    { // Encontramos el maximo
        $group: {
            _id: null,
            maxPelis: { $max: "$pelis" },
            years:
            {  // Guardamos los anhos y su total de peliculas
                $push: {
                    year: "$_id",
                    pelis: "$pelis"
                }
            }
        }
    },
    { // Obtenemos un documento por anho
        $unwind: "$years"
    },
    { // Filtramos por el anho con el maximo de peliculas
        $match: {
            $expr: { $eq: ["$years.pelis", "$maxPelis"] }
        }
    },
    { // Ajustamos el reesultado
        $project: {
            _id: "$years.year",
            pelis: "$years.pelis"
        }
    }
]);

/*Ejercicio 15*/
db.movies.aggregate([
    { // Agrupamos por anho y contamos las peliculas por anho
        $group: {
            _id: "$year",
            pelis: { $sum: 1 }
        }
```

```
    },
    { // Encontramos el minimo
        $group: {
            _id: null,
            minPelis: { $min: "$pelis" },
            years:
            {  // Guardamos los anhos y su total de
            peliculas
                $push: { year: "$_id", pelis: "$pelis" }
            }
        }
    },
    { // Obtenemos un documento por anho
        $unwind: "$years"
    },
    { // Filtramos por el anho con el minimo de peliculas
        $match: { $expr: { $eq: ["$years.pelis", "$minPelis"
        ] } }
    },
    { // Ajustamos el reesultado
        $project: {
            _id: "$years.year",
            pelis: "$years.pelis"
        }
    }
]);

/*Ejercicio 16*/
db.movies.aggregate([
    { // Hacemos el unwind
        $unwind: "$cast"
    },
    { // Eliminamos el id
        $project: { _id: 0 }
    },
    { // Guardamos en la coleccion actors
        $out: "actors"
    }
]);
db.actors.find().count();

/*Ejercicio 17*/
db.actors.aggregate([
    { // Excluimos los actores llamados Undefined
        $match: {
            cast: { $ne: "Undefined" }
        }
    },
    { // Agrupamos por actor y contamos sus apariciones
        $group: {
            _id: "$cast",
```

```
190             cuenta: { $sum: 1 }
191         }
192     },
193     { // Ordenamos por el total de apariciones de forma
        descendente
194         $sort: { cuenta: -1 }
195     },
196     { // Limitamos a 5 el output
197         $limit: 5
198     }
199 ]);
200
201 /*Ejercicio 18*/
202 db.actors.aggregate([
203     { // Agrupamos por pelicula y anho y contamos el total
        de actores
204         $group: {
205             _id: { title: "$title", year: "$year" },
206             cuenta: { $sum: 1 }
207         }
208     },
209     { // Ordenamos por el total de actores de forma
        descendente
210         $sort: { cuenta: -1 }
211     },
212     { // Limitamos a 5 el output
213         $limit: 5
214     }
215 ]);
216
217 /*Ejercicio 19*/
218 db.actors.aggregate([
219     { // Excluimos los actores undefined
220         $match: {
221             cast: { $ne: "Undefined" }
222         }
223     },
224     { // Agrupamos por actor y hallamos la fecha de inicio y
         fin
225         $group: {
226             _id: "$cast",
227             comienza: { $min: "$year" },
228             termina: { $max: "$year" }
229         }
230     },
231     { // Anhadimos un campo con los anhos trabajados
232         $addFields: { anos: {$subtract: ["$termina", "
        $comienza"]} }
233     },
234     { // Ordenamos por la duracion descendente
235         $sort: { anos: -1 }
```

```
    },
    { // Nos quedamos con el maximo
        $limit: 5
    }
]);

/*Ejercicio 20*/
db.actors.aggregate([
    { // Hacemos el unwind
        $unwind: "$genres"
    },
    { // Eliminamos el id
        $project: { _id: 0 }
    },
    { // Guardamos en la coleccion genres
        $out: "genres"
    }
]);
db.genres.count();

/*Ejercicio 21*/
db.genres.aggregate([
    { // Agrupamos por anho y genero y hallamos las
    peliculas distintas
        $group: {
            _id: { year: "$year", genre: "$genres" },
            pelisUnicas: { $addToSet: "$title" }
        }
    },
    { // Anhadimos el contador de pelis unicas
        $project:{
            _id: 1,
            pelis: { $size: "$pelisUnicas" }
        }
    },
    { // Ordenamos por el contador descendente
        $sort: { pelis: -1 }
    },
    { // Nos quedamos con los cinco primeros
        $limit: 5
    }
]);

/*Ejercicio 22*/
db.genres.aggregate([
    { // Eliminamos los actores "Undefined"
        $match: {
            cast: { $ne: "Undefined" }
        }
    },
```

```
285     { // Agrupamos por actor y hallamos los generos
        distintos
286         $group: {
287             _id: "$cast",
288             generos: { $addToSet: "$genres" }
289         }
290     },
291     { // Anhadimos el numero de generos
292         $addFields: { numgeneros: { $size: "$generos" } }
293     },
294     { // Ordenamos por el numero de generos descendente
295         $sort: { numgeneros: -1 }
296     },
297     { // Nos quedamos con los cinco primeros
298         $limit: 5
299     }
300 ]);
301 
302 /*Ejercicio 23*/
303 db.genres.aggregate([
304     { // Agrupamos por titulo y anho y hallamos los generos
        distintos
305         $group: {
306             _id: { title: "$title", year: "$year" },
307             generos: { $addToSet: "$genres" }
308         }
309     },
310     { // Anhadimos el numero de generos
311         $addFields: { numgeneros: { $size: "$generos" } }
312     },
313     { // Ordenamos por el numero de generos descendente
314         $sort: { numgeneros: -1 }
315     },
316     { // Nos quedamos con los cinco primeros
317         $limit: 5
318     }
319 ]);
320 
321 /*Ejercicio 24 - Pelicula con el reparto mas grande */
322 db.movies.aggregate([
323     { // Anhadimos un campo con el tamanho del cast
324         $addFields: { castSize: { $size: "$cast" } }
325     },
326     { // Ordenamos por el tamanho del cast
327         $sort: { castSize: -1 }
328     },
329     { // Limitamos a uno el resultado
330         $limit: 1
331     },
332     { // Eliminamos el id del output y los generos
333         $project: { _id: 0, genres: 0 }
```

```
    }
]);


/*Ejercicio 25 - Actor mas popular por genero */
db.actors.aggregate([
    { // Excluimos los actores y generos Undefined
        $match: {
            $and: [ { genres: { $ne: "Undefined"} }, { cast:
             { $ne: "Undefined"} } ]
        }
    },
    { // Hacemos un unwind de los generos
        $unwind: "$genres"
    },
    { // Agrupamos por genero y actor y contamos
        $group: {
            _id: { genero: "$genres", actor: "$cast" },
            totalPeliculas: { $sum: 1 }
        }
    },
    { // Ordenamos por genero y total de peliculas
        $sort: { "_id.genre": 1, totalPeliculas: -1 }
    },
    { // Agrupamos por genero y seleccionamos el actor con
    mas peliculas
        $group: {
            _id: "$_id.genero",
            topActor: { $first: "$_id.actor" },
            totalPeliculas: { $first: "$totalPeliculas" }
        }
    },
    {
        $project: {
            _id: 0,
            genero: "$_id",
            topActor: 1,
            totalPeliculas: 1
        }
    }
]);

/*Ejercicio 26 - Genero mas popular por decada */
db.genres.aggregate([
    { // Excluimos los generos que sean Undefined
        $match: { genres: { $ne: "Undefined"} }
    },
    { // Calculamos la decada, usando la funcion $mod
        $addFields: {
            decada: { $subtract: ["$year", { $mod: [ "$year"
            , 10 ] } ] }
```

```
382         }
383     },
384     { // Agrupamos por decada y genero y calculamos el total
385         $group: {
386             _id: { decada: "$decada", genero: "$genres" },
387             total: { $sum: 1 }
388         }
389     },
390     { // Ordenamos por decada, de forma ascendente, y luego
    por el total de generos
391         $sort: { "_id.decada": 1, total: -1 }
392     },
393     { // Agrupamos por decada para obtener el genero mas
    popular, y el conteo de peliculas
394         $group: {
395             _id: "$_id.decada",
396             generoMasPopular: { $first: "$_id.genero" },
397             totalPeliculas: { $first: "$total" }
398         }
399     },
400     { // Ordenamos por decada de manera ascendente
401         $sort: { "_id": 1}
402     }
403 ]);
```