

# Bases de datos NoSQL

HUGO GÓMEZ SABUCEDO

[hugogomezsabucedo@gmail.com](mailto:hugogomezsabucedo@gmail.com)

**Máster Big Data, Data Science & Inteligencia Artificial**

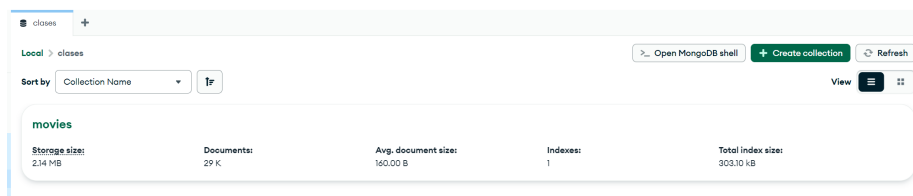
Curso 2024-2025

Universidad Complutense de Madrid

## 1. Ejercicios

A continuación se incluyen las capturas con las queries 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 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 MongoDB Compass, importando los datos directamente desde el archivo JSON.



1. Analizar con find la colección.



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

```
7 /*Ejercicio 2*/
8 db.movies.find().count()
9
```

Console x	Line5 db.movies.find() x	Result x	Result (1) x
0.056 s			
1	28795		

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

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

Result (3) x	Result (4) x	Result (5) x	Result (6) x	Result (7) x	Result (8) x	Result (9) x	Console (2) x	Result (10) x	Result (11) x
0.058 s									
1	{								
2	"acknowledged" : true,								
3	"insertedId" : ObjectId("674cafde793a77d5440e85ab")								
4	}								

**4. Borrar la película insertada en el punto anterior.** Como la película se llama *test* 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.

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

Result (4) x	Result (5) x	Result (6) x	Result (7) x	Result (8) x	Result (9) x
0.093 s					
1	{				
2	"acknowledged" : true,				
3	"deletedCount" : 1				
4	}				

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) x Result (17) x Find x Error (1) x Error (2) x

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) x Error (3) x Error (4) x Error (5) x Error (6) x Error (7) x Find (1) x Error (8) x

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
```

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
```

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()
```

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"] } })
```

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)
```

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.

```

37 /*Ejercicio 12*/
38 db.movies.aggregate([
39   { // Encontrar el año mas reciente
40     $group: {
41       _id: null,
42       maxYear: { $max: "$year" }
43     }
44   },
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 l
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   { // Añadimos un campo con el total de las películas
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 ]]);

```

Result	Aggregate	Result (1)	Aggregate (1)	Aggregate (2)	Aggregate (3)	Aggregate (4)	Aggregate (5)
movies	0.219 s	1 Doc					
1	[{						
2	"_id": "",						
3	"total": 4787						
4	}]						

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).

```
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 }
93     }
94   }
95 ])
```

Aggregate \* Aggregate (1) \* Aggregate (2) \* Aggregate (3) \*

movies 0.105 s 1 Doc

```
1 {
2   "_id" : null,
3   "total" : 1414
4 }
```



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.

```

197 /*Ejercicio 14*/
198 db.movies.aggregate([
199   { // Agrupamos por año y contamos las películas por año
200     $group: {
201       _id: "$year",
202       pelis: { $sum: 1 }
203     },
204   },
205   { // Encontramos el máximo
206     $group: {
207       _id: null,
208       maxPelis: { $max: "$pelis" },
209       years:
210       { // Guardamos los años y su total de películas
211         $push: {
212           year: "$_id",
213           pelis: "$pelis"
214         }
215       }
216     },
217   },
218   { // Obtenemos un documento por año
219     $unwind: "$years"
220   },
221   { // Filtramos por el año con el máximo de películas
222     $match: {
223       $expr: { $eq: ["$years.pelis", "$maxPelis"] }
224     }
225   },
226   { // Ajustamos el resultado
227     $project: {
228       _id: "$years.year",
229       pelis: "$years.pelis"
230     }
231   }
232 ])
233

```

« (6) \* Aggregate (30) \* Aggregate (31) \* Aggregate (32) \* Aggregate (33) \* Aggreg

movies 0.034 s 1 Doc

```

1 {
2   "_id" : 1919,
3   "pelis" : 634
4 }

```

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

```

134 /*Ejercicio 15*/
135 db.movies.aggregate([
136   { // Agrupamos por año y contamos las películas por año
137     $group: {
138       _id: "$year",
139       pelis: { $sum: 1 }
140     },
141   },
142   { // Encontramos el mínimo
143     $group: {
144       _id: null,
145       minPelis: { $min: "$pelis" },
146       years:
147       { // Guardamos los años y su total de películas
148         $push: { year: "$_id", pelis: "$pelis" }
149       }
150     },
151   },
152   { // Obtenemos un documento por año
153     $unwind: "$years"
154   },
155   { // Filtramos por el año con el mínimo de películas
156     $match: { $expr: { $eq: ["$years.pelis", "$minPelis"] } }
157   },
158   { // Ajustamos el resultado
159     $project: {
160       _id: "$years.year",
161       pelis: "$years.pelis"
162     }
163   }
164 ]);
165

```

1) x

Aggregate (1) x

Aggregate (2) x

Aggregate (3) x

Aggregate (4) x

Aggregate (5) x

movies

0.130 s

3 Docs

```

1 [{
2   "_id": 1907,
3   "pelis": 7
4 },
5 {
6   "_id": 1906,
7   "pelis": 7
8 },
9 {
10  "_id": 1902,
11  "pelis": 7
12 }
13 ]

```

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
```

	movies.aggregate(... x	Aggregate x	Find x	Result x	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.

```

180  /*Ejercicio 17*/
181  db.actors.aggregate([
182    { // Excluimos los actores llamados Undefined
183      $match: {
184        cast: { $ne: "Undefined" }
185      },
186    },
187    { // Agrupamos por actor y contamos sus apariciones
188      $group: {
189        _id: "$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  ])

```

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

actors 0.235 s 5 Docs

```

1  [ {
2    "_id": "Harold Lloyd",
3    "cuenta": 190
4  },
5  {
6    "_id": "Hoot Gibson",
7    "cuenta": 142
8  },
9  {
10   "_id": "John Wayne",
11   "cuenta": 136
12 },
13 {
14   "_id": "Charles Starrett",
15   "cuenta": 116
16 },
17 {
18   "_id": "Bebe Daniels",
19   "cuenta": 103
20 }
21 ]

```

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.

```
201 /*Ejercicio 18*/
202 db.actors.aggregate([
203   { // Agrupamos por película y año y contamos el total de actores
204     $group: {
205       _id: {
206         title: "$title", year: "$year"
207       },
208       cuenta: { $sum: 1 }
209     },
210   },
211   { // Ordenamos por el total de actores de forma descendente
212     $sort: { cuenta: -1 }
213   },
214   { // Limitamos a 5 el output
215     $limit: 5
216   }
217 ])
```

```
actors 0.403 s 5 Docs
1 - [{
2 -   "_id": {
3 -     "title": "The Twilight Saga: Breaking Dawn - Part 2",
4 -     "year": 2012
5 -   },
6 -   "cuenta": 35
7 - },
8 - {
9 -   "_id": {
10 -    "title": "Anchorman 2: The Legend Continues",
11 -    "year": 2013
12 -   },
13 -   "cuenta": 33
14 - },
15 - {
16 -   "_id": {
17 -    "title": "Cars 2",
18 -    "year": 2011
19 -   },
20 -   "cuenta": 32
21 - },
22 - {
23 -   "_id": {
24 -    "title": "Avengers: Infinity War",
25 -    "year": 2018
26 -   },
27 -   "cuenta": 29
28 - },
29 - {
30 -   "_id": {
31 -    "title": "Grown Ups 2",
32 -    "year": 2013
33 -   },
34 -   "cuenta": 28
35 - }
36 ]
```

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.

```

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   { // Añadimos un campo con los años trabajados
232     $addFields: { anos: { $subtract: ["$termina", "$comienza"] } }
233   },
234   { // Ordenamos por la duración descendente
235     $sort: { anos: -1 }
236   },
237   { // Nos quedamos con el máximo
238     $limit: 5
239   }
240 ]);
241

```

Result x

Aggregate x

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 ];
256 db.genres.count()
```

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

0.056 s

1	104950
---	--------

21. **Sobre géneros, 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.

```

255 /*Ejercicio 21*/
256 db.genres.aggregate([
257   { // Agrupamos por año y género y hallamos las películas distintas
258     $group: {
259       _id: { year: "$year", genre: "$genres" },
260       pelisUnicas: { $addToSet: "$title" }
261     }
262   },
263   { // Añadimos el contador de pelis unicas
264     $project: {
265       _id: 1,
266       pelis: { $size: "$pelisUnicas" }
267     }
268   },
269   { // Ordenamos por el contador descendente
270     $sort: { pelis: -1 }
271   },
272   { // Nos quedamos con los cinco primeros
273     $limit: 5
274   }
275 ])
276

```

```

genres 0.272 s 5 Docs
1 - [{
2   {
3     "_id": {
4       "year": 1919,
5       "genre": "Drama"
6     },
7     "pelis": 291
8   },
9   {
10    "_id": {
11      "year": 1925,
12      "genre": "Drama"
13    },
14    "pelis": 247
15  },
16  {
17    "_id": {
18      "year": 1924,
19      "genre": "Drama"
20    },
21    "pelis": 233
22  },
23  {
24    "_id": {
25      "year": 1919,
26      "genre": "Comedy"
27    },
28    "pelis": 226
29  },
30  {
31    "_id": {
32      "year": 1922,
33      "genre": "Drama"
34    },
35    "pelis": 209
36  }
}

```



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.

```

277 /*Ejercicio 22*/
278 db.genres.aggregate([
279   { // Eliminamos los actores "Undefined"
280     $match: {
281       cast: { $ne: "Undefined" }
282     },
283   },
284   { // Agrupamos por actor y hallamos los géneros distintos
285     $group: {
286       _id: "$cast",
287       genres: { $addToSet: "$genres" }
288     },
289   },
290   { // Añadimos el número de géneros
291     $addFields: { numgeneros: { $size: "$genres" } }
292   },
293   { // Ordenamos por el numero de generos descendente
294     $sort: { numgeneros: -1 }
295   },
296   { // Nos quedamos con los cinco primeros
297     $limit: 5
298   }
299 ])

```

```

1  {
2    "_id": "Dennis Quaid",
3    "genres": [
4      "Fantasy",
5      "Adventure",
6      "Comedy",
7      "Family",
8      "Science Fiction",
9      "Sports",
10     "Dance",
11     "Drama",
12     "Animated",
13     "Satire",
14     "Suspense",
15     "Action",
16     "Crime",
17     "Horror",
18     "Western",
19     "Musical",
20     "Romance",
21     "Disaster",
22     "Thriller",
23     "Biography"
24   ],
25   "numgeneros": 20
26 },
27 {
28   "_id": "James Mason",
29   "genres": [
30     "Romance",
31     "Science Fiction",
32     "Adventure",
33     "Comedy",
34     "Short",
35     "Animated",
36     "Suspense",
37     "Drama",
38     "Noir",
39     "Western",
40     "Musical",
41     "Crime",
42     "Mystery",
43     "Undefined",
44     "Action",
45     "Biography",
46     "Fantasy",
47     "War",
48     "Thriller"
49 ],
50   "numgeneros": 19
51 },
52 {
53   "_id": "Michael Caine",
54   "genres": [
55     "Spy",
56     "Adventure",
57     "Comedy",
58     "Family",
59     "Science Fiction",
60     "Drama",
61     "Animated",
62     "Suspense",
63     "Horror",
64     "Crime",
65     "Undefined",
66     "Action",
67     "Mystery",
68     "Superhero",
69     "Disaster",
70     "Romance",
71     "War",
72     "Thriller",
73     "Biography"
74   ],
75   "numgeneros": 19
76 },
77 {
78   "_id": "James Coburn",
79   "genres": [
80     "War",
81     "Science Fiction",
82     "Sports",
83     "Adventure",
84     "Family",
85     "Spy",
86     "Comedy",
87     "Animated",
88     "Satire",
89     "Suspense",
90     "Drama",
91     "Western",
92     "Mystery",
93     "Crime",
94     "Action",
95     "Biography",
96     "Romance",
97     "Thriller"
98 ],
99   "numgeneros": 18
100 },
101 {
102   "_id": "Lionel Barrymore",
103   "genres": [
104     "Adventure",
105     "Comedy",
106     "Science Fiction",
107     "Sports",
108     "Drama",
109     "Historical",
110     "Noir",
111     "Horror",
112     "Undefined",
113     "Mystery",
114     "Crime",
115     "Western",
116     "Musical",
117     "Romance",
118     "War",
119     "Thriller",
120     "Biography",
121     "Fantasy"
122 ],
123   "numgeneros": 18
124 }
125 ]

```

23. **Sobre géneros, 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.

```

301  /*Ejercicio 23*/
302  db.genres.aggregate([
303    { // Agrupamos por título y año y hallamos los géneros distintos
304      $group: {
305        _id: { title: "$title", year: "$year" },
306        genres: { $addToSet: "$genres" }
307      }
308    },
309    { // Añadimos el número de géneros
310      $addFields: { numgeneros: { $size: "$genres" } }
311    },
312    { // Ordenamos por el número de géneros descendente
313      $sort: { numgeneros: -1 }
314    },
315    { // Nos quedamos con los cinco primeros
316      $limit: 5
317    }
318  ])

```

```

1 - [{
2   {
3     "_id": {
4       "title": "American Made",
5       "year": 2017
6     },
7     "genres": [
8       "Drama",
9       "Historical",
10      "Crime",
11      "Action",
12      "Comedy",
13      "Thriller",
14      "Biography"
15    ],
16    "numgeneros": 7
17  },
18  {
19    "_id": {
20      "title": "The Dark Tower",
21      "year": 2017
22    },
23    "genres": [
24      "Western",
25      "Horror",
26      "Action",
27      "Fantasy",
28      "Adventure",
29      "Science Fiction"
30    ],
31    "numgeneros": 6
32  },
33  {
34    "_id": {
35      "title": "Wonder Woman",
36      "year": 2017
37    },
38    "genres": [
39      "War",
40      "Drama",
41      "Superhero",
42      "Adventure",
43      "Fantasy",
44      "Action"
45    ],
46    "numgeneros": 6

```

```

47  {
48    "_id": {
49      "title": "Thor: Ragnarok",
50      "year": 2017
51    },
52    "genres": [
53      "Science Fiction",
54      "Comedy",
55      "Adventure",
56      "Action",
57      "Superhero",
58      "Fantasy"
59    ],
60    "numgeneros": 6
61  },
62  {
63    "_id": {
64      "title": "Dunkirk",
65      "year": 2017
66    },
67    "genres": [
68      "War",
69      "Drama",
70      "Historical",
71      "Action",
72      "Adventure",
73      "Thriller"
74    ],
75    "numgeneros": 6
76  }
77 ]

```

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.

```

321 /*Ejercicio 24 - Película con el reparto más grande */
322 db.movies.aggregate([
323   { // Añadimos un campo con el tamaño del cast
324     $addFields: { castSize: { $size: "$cast" } }
325   },
326   { // Ordenamos por el tamaño 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 géneros
333     $project: { _id: 0, genres: 0 }
334   }
335 ]);

```

```

1- [{
2   "title": "The Twilight Saga: Breaking Dawn - Part 2",
3   "year": 2012,
4   "cast": [
5     "Kristen Stewart",
6     "Robert Pattinson",
7     "Taylor Lautner",
8     "Nikki Reed",
9     "Peter Facinelli",
10    "Elizabeth Reaser",
11    "Ashley Greene",
12    "Kellan Lutz",
13    "Jackson Rathbone",
14    "Julia Jones",
15    "Booboo Stewart",
16    "Billy Burke",
17    "Sarah Clarke",
18    "MyAnna Buring",
19    "Maggie Grace",
20    "Casey LaBow",
21    "Michael Sheen",
22    "Jamie Campbell Bower",
23    "Christopher Heyerdahl",
24    "Chaske Spencer",
25    "Christian Camargo",
26    "Mia Maestro",
27    "Mackenzie Foy",
28    "Dakota Fanning",
29    "Cameron Bright",
30    "Charlie Bewley",
31    "Daniel Cudmore",
32    "Noel Fisher",
33    "Gurti Weinberg",
34    "Lee Pace",
35    "Joe Anderson",
36    "Judi Shekoni",
37    "Tracey Huggins",
38    "J. D. Pardo",
39    "Rami Malek"
40   ],
41   "castSize": 35
42 }]

```

25. **Ejercicio libre. Sobre actores, 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.

```

337 /*Ejercicio 25 - Actor más popular por género */
338 db.actors.aggregate([
339   { // Excluimos los actores y géneros Undefined
340     $match: {
341       $and: [ { genres: { $ne: "Undefined" } }, { cast: { $ne: "Undefined" } } ]
342     },
343   },
344   { // Hacemos un unwind de los generos
345     $unwind: "$genres"
346   },
347   { // Agrupamos por genero y actor y contamos
348     $group: {
349       _id: { genero: "$genres", actor: "$cast" },
350       totalPelículas: { $sum: 1 }
351     },
352   },
353   { // Ordenamos por género y total de películas
354     $sort: { "_id.genero": 1, totalPelículas: -1 }
355   },
356   { // Agrupamos por genero y seleccionamos el actor con mas películas
357     $group: {
358       _id: "$_id.genero",
359       topActor: { $first: "$_id.actor" },
360       totalPelículas: { $first: "$totalPelículas" }
361     },
362   },
363   {
364     $project: {
365       _id: 0,
366       genero: "$_id",
367       topActor: 1,
368       totalPelículas: 1
369     }
370   }
371 ]);

```

actors	0.958 s	40 Docs				60	1	1-40	Type	
logActor		totalPelículas	genero							
1	Bette Davis	55	Drama							
2	Adam G. Sevast	3	Dance							
3	William Garwood	1	Sant							
4	Gary Gray	4	Suspense							
5	Kane Hodder	2	Slasher							
6	Hoel Gibson	121	Western							
7	Emma Watson	9	Fantasy							
8	Bernie Hunt	7	Fanty							
9	The Three Musketeers	44	Short							
10	Elyse Knox	4	Sports							
11	George Kennedy	4	Disaster							
12	Leonard Nimoy	9	Science Fiction							
13	Edward G. Robinson	14	Nue							
14	John Lecker	2	Team							
15	Warren Oland	16	Mystery							
16	George Clooney	3	Political							
17	Grigor Gukha	1	Independent							
18	Johnny Weissmuller	28	Adventure							
19	Edward G. Robinson	21	Crime							
20	Gary Cooper	15	Romance							

actors		0.958 s	40 Docs		
	totalPelículas		genero		
20	Gary Cooper	15	Romance		
21	Dean Cain	5	Sci		
22	Yvonne French	5	Historical		
23	Jesse Gils	4	Emic		
24	John Wayne	12	War		
25	Kate Burton	1	Legal		
26	Art Hanks	8	Biography		
27	Neil Patrick Harris	2	Law Action		
28	Jonas Brothers	2	Performance		
29	Itaq War	5	Documentary		
30	Jess-Charles Vele Dianna	22	Action		
31	Harold Lloyd	188	Comedy		
32	William Culler Jr	1	Sport		
33	Tom and Jerry	80	Animated		
34	Engelke Levy	2	Satire		
35	Jackie Chan	3	Martial Arts		
36	Reggie Cleveland	41	Musical		
37	Chris Evans	8	Superhero		
38	Vera Farmiga	2	Supernatural		
39	Harrison Cagle	14	Thriller		
40	Rocky Horror	28	Horror		

**26. Ejercicio libre. Sobre generes, mostrar el género más popular en cada década, mostrando su nombre y el total de películas.** Primero, se excluyen 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.

```

326 //Ejercicio 26 - Género más popular por década */
327 db.genres.aggregate([
328   { // Excluimos los géneros que sean Undefined
329     $match: { genres: { $ne: "Undefined" } }
330   },
331   { // Calculamos la década, usando la función $mod
332     $addFields: {
333       decada: { $subtract: [ "$year", { $mod: [ "$year", 10 ] } ] }
334     },
335   },
336   { // Agrupamos por década y género y calculamos el total
337     $group: {
338       _id: { decada: "$decada", genero: "$genres" },
339       total: { $sum: 1 }
340     },
341   },
342   { // Ordenamos por década, de forma ascendente, y luego por el total de géneros
343     $sort: { "_id.decada": 1, total: -1 }
344   },
345   { // Agrupamos por década para obtener el género más popular, y el conteo de películas
346     $group: {
347       _id: "$_id.decada",
348       generoMasPopular: { $first: "$_id.genero" },
349       totalPelículas: { $first: "$total" }
350     },
351   },
352   { // Ordenamos por década de manera ascendente
353     $sort: { "_id": 1 }
354   },
355 ])

```

genres	0.517 s	12 Docs
_id	generoMasPopular	totalPelículas
1	1900	Comedy
2	1910	Drama
3	1920	Drama
4	1930	Drama
5	1940	Comedy
6	1950	Drama
7	1960	Drama
8	1970	Drama
9	1980	Comedy
10	1990	Drama
11	2000	Comedy
12	2010	Comedy