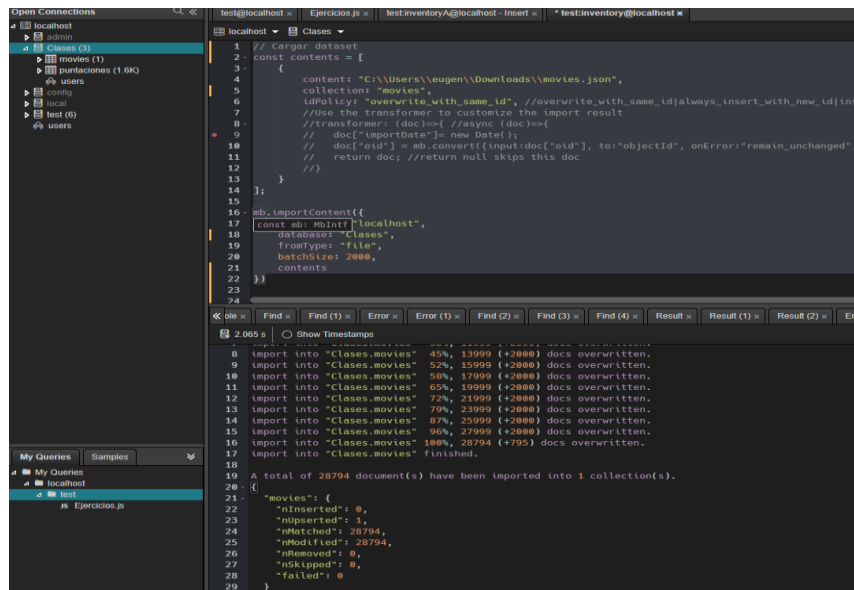


## 0. Realizar la importación del json en una colección llamada movies



The screenshot shows the MongoDB Compass interface. On the left, the 'Classes' collection is expanded, showing the 'movies' sub-collection with 1 document. The main editor displays a JavaScript script for importing a JSON file. The script defines a collection named 'movies' and uses the 'importContent' method to load data from a local file. The console output at the bottom shows the progress of the import, indicating that 28,794 documents were successfully imported into the 'movies' collection.

```
1 // Cargar dataset
2 const contents = [
3   {
4     content: "C:\\Users\\eugen\\Downloads\\movies.json",
5     collection: "movies",
6     idPolicy: "overwrite_with_same_id", //overwrite_with_same_id|always_insert_with_new_id|insert_with_new_id
7     //Use the transformer to customize the import result
8     //transformer: (doc) => { //async (doc) => {
9       doc["importDate"] = new Date();
10      // doc["old"] = mb.convert({input: doc["old"], to: "objectId", onError: "remain_unchanged"},
11      // return doc; //return null skips this doc
12    }
13  }
14 ];
15
16 mb.importContent({
17   const mb: MongoClient | MongoClient,
18   uri: string,
19   fromType: "file",
20   batchSize: 2000,
21   contents
22 })
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65
66
67
68
69
70
71
72
73
74
75
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99
100
```

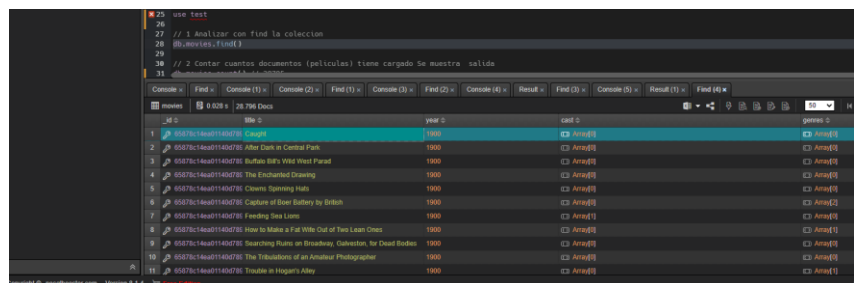
2.065 s

0 import into "Classes.movies" 45%, 13999 (+2000) docs overwritten.  
9 import into "Classes.movies" 52%, 15999 (+2000) docs overwritten.  
10 import into "Classes.movies" 58%, 17999 (+2000) docs overwritten.  
11 import into "Classes.movies" 65%, 19999 (+2000) docs overwritten.  
12 import into "Classes.movies" 72%, 21999 (+2000) docs overwritten.  
13 import into "Classes.movies" 78%, 23999 (+2000) docs overwritten.  
14 import into "Classes.movies" 87%, 25999 (+2000) docs overwritten.  
15 import into "Classes.movies" 96%, 27999 (+2000) docs overwritten.  
16 import into "Classes.movies" 100%, 28794 (+795) docs overwritten.  
17 import into "Classes.movies" finished.

A total of 28794 document(s) have been imported into 1 collection(s).

```
21 {
22   "movies": {
23     "nInserted": 0,
24     "nUpserted": 1,
25     "nMatched": 28794,
26     "nModified": 28794,
27     "nRemoved": 0,
28     "nSkipped": 0,
29     "failed": 0
30   }
31 }
```

## 1. Analizar con find la colección.

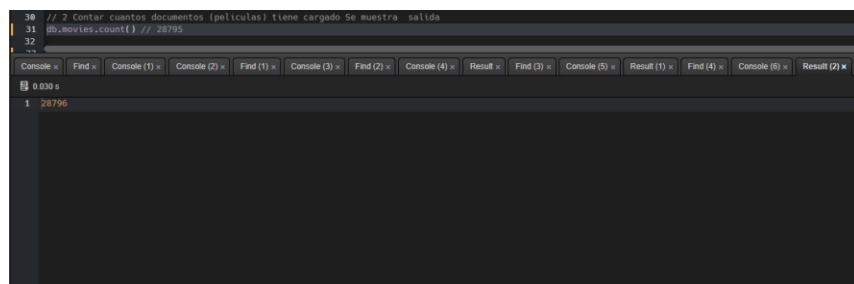


The screenshot shows the MongoDB Compass interface with a 'find' query executed on the 'movies' collection. The console output displays a list of 10 movie documents, each with fields for '\_id', 'title', 'year', 'cast', and 'genres'. The documents are sorted by year in descending order, showing movies from 1990.

```
25 use test
26 // 1 Analizar con find la coleccion
27 db.movies.find()
28
29 // 2 Contar cuantos documentos (películas) tiene cargado se muestra salida
30 db.movies.count()
31
```

_id	title	year	cast	genres
6507614e0d1140d70000000000000000	Captain	1990	Array(5)	Array(5)
6507614e0d1140d70000000000000000	After Dark in Central Park	1990	Array(5)	Array(5)
6507614e0d1140d70000000000000000	Before the Wind	1990	Array(5)	Array(5)
6507614e0d1140d70000000000000000	The Exchanged	1990	Array(5)	Array(5)
6507614e0d1140d70000000000000000	Cloves	1990	Array(5)	Array(5)
6507614e0d1140d70000000000000000	Capture of Boer Battery by British	1990	Array(5)	Array(5)
6507614e0d1140d70000000000000000	Feeding Sea Lions	1990	Array(1)	Array(5)
6507614e0d1140d70000000000000000	How to Make a Fat Wife Out of Two Lean Ones	1990	Array(5)	Array(1)
6507614e0d1140d70000000000000000	Searching Home on Broadway, Collection, for Dead Bodies	1990	Array(5)	Array(5)
6507614e0d1140d70000000000000000	The Tedium of an Amateur Photographer	1990	Array(5)	Array(5)
6507614e0d1140d70000000000000000	Trouble in Heaven's Alley	1990	Array(5)	Array(1)

## 2. Contar cuantos documentos (películas) tiene cargado se muestra salida



The screenshot shows the MongoDB Compass interface with a 'count' query executed on the 'movies' collection. The console output displays the result of the count query, which is 28795.

```
30 // 2 Contar cuantos documentos (películas) tiene cargado se muestra salida
31 db.movies.count() // 28795
32
```

0.030 s

```
1 28795
```

### 3. Insertar una película

```
33 // 3. Insertar una película.
34 db.movies.insert({"title": "Alien", "year": "1995", "cast": [], "genres": []})
35
36 //
```

Console x Find x Console (1) x Console (2) x Find (1) x Console (3) x Find (2) x Console (4) x Result x Find

0.042 s

```
1 - {
2   "acknowledged" : true,
3   "insertedId" : ObjectId("65878ca8ea01140d789afad2")
4 }
```

### 4. Eliminar la película insertada en el punto anterior

```
36 // 4. Borrar la película insertada en el punto anterior
37 db.movies.deleteOne({"title": "Alien", "year": "1995"})
```

Console x Find x Console (1) x Console (2) x Find (1) x Console (3) x Find (2) x Console (4) x Result x Find (3) x Console (5) x

0.064 s

```
1 - {
2   "acknowledged" : true,
3   "deletedCount" : 1
4 }
```

### 5. Contar cuantas películas tienen actores (cast) que se llaman "and"

```
38
39 // 5. Contar cuantas películas tienen actores (cast) que se llaman "and"
40 db.movies.find({"cast":{"$in": ["and"]}}).count()
```

Console x Result (4) x Result x

0.050 s

```
1 93
```

### 6. Actualizar los documentos cuyo actor tenga por error el valor "and"

```
42 // 6. Actualizar los documentos cuyo actor tenga por error el valor "and"
43 db.movies.update(
44   {"cast": "and"},
45   {"$pull : {cast: "and"} },
46   {"multi" : true}
47 )
```

Console x Result (4) x Result x Console (1) x Result (1) x

0.067 s

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

7. Contar cuantas películas tienen el array 'cast' vacío

```
49 // 7. Contar cuantas películas tienen el array 'cast' vacío
50 db.movies.find({cast:{ $eq:[]}}).count()
```

Console x Result (4) x Result x Console (1) x Result (1) x Find x Find (1) x Result (2) x Result (3) x

0.064 s

1 986

8. Actualizar todos los documentos que tengan el array cast vacío. Añadiendo un nuevo elemento dentro del array con valor Undefined

```
52 // 8. Actualizar todos los documentos que tengan el array cast vacío,
53 // añadiendo un nuevo elemento dentro del array con valor Undefined
54 db.movies.update(
55   {cast:{ $eq:[] } },
56   { $set : {cast : ["Undefined"] } },
57   {multi: true})
58
```

Console x Result (4) x Result x Console (1) x Result (1) x Find x Find (1) x Result (2) x Result (3) x Console (2) x Result (5) x

0.080 s

Key	Value	Type
(1)	{ acknowledged : true, matchedCount : 986, modifiedCount : 986 }	Object
acknowledged	true	Bool
matchedCount	986	Int32
modifiedCount	986	Int32

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

```
59 // 9. Contar cuantos documentos (películas) tienen el array genres vacío.
60 db.movies.find({genres:{ $eq: []}}).count()
```

(property) mongo.IDatabase.movies: mongo.\_\_mbmovies

Console x Result (4) x Result x Console (1) x Result (1) x Find x Find (1) x Result (2) x Result (3) x Console

0.052 s

1 901

10. Actualizar todos los documentos que tengan el array genres vacío, el tipo de dato debe de seguir siendo un array

```
62 // 10. Actualizar todos los comentarios que tengan el array genres vacío.
63 // El tipo de genres debe seguir siendo un array
64 db.movies.update(
65   {genres:{$eq: [] } },
66   {$set: {genres: ["Undefined"]}},
67   {multi: true})
```

Console x Result (4) x Result x Console (1) x Result (1) x Find x Find (1) x Result (2) x Result (3) x Console (2) x Result (5) x

0.097 s

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

11. Mostrar el año más reciente/ actual que tenemos sobre las películas

```
69 // 11. Mostrar el año más reciente / actual que tenemos sobre las películas.
70 db.movies.find({}, {year:1, _id:0}).sort({"year": -1}).limit(1)
```

<< x Console (2) x Result (5) x Find (2) x Result (6) x Console (3) x Result (7) x Find (3) x Find (4) x Find (5)

movies 0.041 s 1 Doc

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

12. Contar cuantas películas han salido en los últimos 20 años. Debe hacerse desde el último año que se tienen registradas las películas

```
72 // 12. Contar cuantas películas han salido en los últimos 20 años. Debe hacerse
73 // hacerse desde el último año que se tienen registradas las películas
74
75 var query = db.movies.find({}, {year:1, _id:0}).sort({"year": -1}).limit(1).toArray()[0].year
76 db.movies.count({year: {$gte: query - 19}})
77
78 // 13.
79
80
81
```

Find (22) x Find (23) x Find (24) x Error (4) x Find (25) x Error (5) x Error (6) x Find (26) x Error (7) x Result (8) x R

0.139 s

```
1 4787
```

13. Contar cuantas películas han salido en la década de los 60s del (60 al 69 incluidos). Se debe hacer con el Framework de agregación

```
78 // 13. Contar cuántas películas han salido en la década de los 60
79 // del (60 al 69 incluidos). Se debe hacer con el Framework de agregación
80
81 db.movies.find({year:{$in:[1960, 1961, 1962, 1963, 1964, 1965, 1966, 1967, 1968, 1969]}}).count()
82
83 db.movies.aggregate([{$match:{year: {$gte:1960, $lte:1969}}},
84                       {$count: "total"}])
85
86
87
88
```

Find (27) x Find (28) x Console (5) x Result (11) x Console (6) x Result (12) x Error (8) x Error (9) x Find (29) x Find (30) x

movies 0.153 s 1 Doc

```
1 - {
2   "total" : 1414
3 }
```

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

```
86 // 14. Mostrar el año con más películas mostrando el número de películas de ese año
87 // Revisar si varios años pueden compartir tener el mayor número de películas
88 db.movies.aggregate([{$group: {_id: "$year", pelis: {$sum: 1} } },
89                       {$sort: {totalMovies: -1}}]).sort({"pelis": -1})
90
91
```

Aggregate (2) x Result (13) x Aggregate (3) x Error (13) x Error (14) x Error (15) x Error (16) x Aggregate (4) x Aggregate (5) x Aggregate (6) x Aggregate (7) x

movies 0.087 s 119 Docs

```
1 /* 1 */
2 {
3   "_id" : 1919,
4   "pelis" : 634
5 },
6
7 /* 2 */
8 {
9   "_id" : 1925,
10  "pelis" : 572
11 },
12
13 /* 3 */
14 {
15   "_id" : 1936,
16   "pelis" : 504
17 }
```

15. Mostrar el año u años con menos películas mostrando el número de películas de ese año

```
92 // 15. Mostrar el año u años con menos películas mostrando el número de películas de ese año
93 db.movies.aggregate([{$group: {_id: "$year", // agrupa por year
94                             pelis: {$sum: 1} } }, // Cuenta las películas
95                             ]).sort({"pelis": 1}) // Ordena las películas
96
97
```

movies 0.207 s 119 Docs

```
1 /* 1 */
2 {
3   "_id" : 1902,
4   "pelis" : 7
5 },
6
7 /* 2 */
8 {
9   "_id" : 1906,
10  "pelis" : 7
11 },
12
13 /* 3 */
14 {
15   "_id" : 1907,
16   "pelis" : 7
17 }
```

En este caso se observa que el año 1902, 1906 y 1907 comparten el año con el menor número de películas todas teniendo únicamente 7 películas.

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

The screenshot shows a MongoDB IDE interface. On the left, a sidebar displays the database structure with collections: actors (83.2K), schema, validator (empty), indexes (1), \_id\_ (0.82MB), inventory (19), inventoryA (5), inventoryAE (4), movies (28.8K), and users. The main editor displays a JavaScript script for MongoDB aggregation and document manipulation. The script includes comments in Spanish and uses the `aggregate`, `sort`, `$group`, `$project`, `$out`, and `$unwind` operators. The bottom panel shows the execution results, indicating a successful run in 0.055 seconds with one result document.

```
85
86 // 14. Mostrar el año con más películas mostrando el número de películas de ese año
87 // Revisar si varios años pueden compartir tener el mayor número de películas
88 db.movies.aggregate([
89   $group: {
90     _id: "$year", // agrupa por year
91     pelis: { $sum: 1 }
92   }
93 ], // Cuenta las películas
94 []).sort({ "pelis": -1 }) // Ordena las películas
95
96 // 15. Mostrar el año u años con menos películas mostrando el número de películas de ese año
97 db.movies.aggregate([
98   $group: {
99     _id: "$year", // agrupa por year
100    pelis: { $sum: 1 }
101  }
102 ], // Cuenta las películas
103 []).sort({ "pelis": 1 }) // Ordena las películas
104
105
106 // 16. Guardar en una nueva colección llamada "actors" realizando la fase $unwind por actor.
107 // Después, contar cuantos documentos existen en la nueva colección
108 db.movies.aggregate([
109   { $unwind: "$cast" },
110   { $project: { _id: 0, title: 1, year: 1, cast: 1, genres: 1 } },
111   { $out: "actors" }
112 ]);
113
114 db.actors.count()
```

Aggregate x Error (1) x Error x Aggregate (1) x Console x Result x Find x Console (1) x Result (1) x

0.055 s

1 83224

\$project se utilizó para manejar los id's duplicados que surgen a partir de utilizar \$unwind, haciendo explícito `_id:0`, se le obliga a mongodb a generar nuevos id's y lidiar con el problema de id's duplicados.

17. Sobre actors, mostrar la lista con los 5 actores que han participado en más películas mostrando el número de películas en las que se ha participado. Importante, se necesita previamente filtrar para descartar aquellos actores llamados "Undefined"

```

116 // 17. Sobre actos, mostrar la lista con los 5 actores que han participado en más películas
117 // mostrando el número de películas en las que se ha participado. Importante, se necesita
118 // previamente filtrar para descartar aquellos actores llamados "Undefined". Aclarar
119 // que no se eliminan de la colección, solo que filtramos para que no aparezcan.
120
121 db.actors.aggregate([
122   {$match: {cast: { $ne: "Undefined"}}},
123   {$group: {
124     _id: "$cast",
125     cuenta: {$sum: 1}
126   }},
127   {$sort: {cuenta: -1}}]).limit(5)

```

Aggregate x Error (1) x Error x Aggregate (1) x Console x Result x Find x Console (1) x Result (1) x Aggregate (2) x

actors 0.522 s 5 Docs

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

```

129 // 18. Sobre actors, agrupar por película y año mostrando las 5 en las que más
130 // actores hayan participado, mostrando el número total de actores
131 db.actors.aggregate([
132   {$match: {cast: { $ne: "Undefined"}}},
133   {$group: {
134     _id: {title: "$title", year: "$year"},
135     cuenta: {$sum: 1}
136   }},
137   {$sort: {cuenta: -1}}]).limit(5)

```

Aggregate x Error (1) x Error x Aggregate (1) x Console x Result x Find x Console (1) x Result (1) x Aggregate (2) x Aggregate (3) x Aggregate (4) x Find (1) x Aggregate (5) x Aggregate (6) x

actors 0.780 s 5 Docs

	title	year	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

19. Sobre actors, 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. Se necesita previamente filtrar para descartar aquellos actores llamados "Undefined"



```

139 // 19. Sobre actors, mostrar los 5 actores cuya carrera haya sido la más larga.
140 // para ello, se debe mostrar cuándo comenzó su carrera, cuándo finalizó y
141 // cuántos años ha trabajado. Se necesita previamente filtrar para descartar
142 // aquellos actores llamados "Undefined"
143
144 db.actors.aggregate([
145   {$match: {cast: {$ne: "Undefined"}}},
146   {$group: {
147     _id: {cast: "$cast"},
148     comienza: {$min: "$year"},
149     termina: {$max: "$year"},
150     $project: {
151       _ids: 0,
152       cast: "$_id",
153       comienza: 1,
154       termina: 1,
155       anos: {$subtract: ["$termina", "$comienza"]},
156       {$sort: {anos: -1}}}).limit(6)
157   }
158 ])
159 db.actors.find({cast: "Harrison Ford"}).sort({year: -1})

```

comienza	termina	cast	anos
1919 (1.9K)	2017 (2.0K)	( cast : "Harrison Ford" )	98
1932 (1.9K)	2012 (2.0K)	( cast : "Gloria Stuart" )	80
1937 (1.9K)	2012 (2.0K)	( cast : "Kenny Baker" )	75
1912 (1.9K)	1987 (2.0K)	( cast : "Lillian Gish" )	75
1944 (1.9K)	2018 (2.0K)	( cast : "Angela Lansbury" )	74
1932 (1.9K)	2008 (2.0K)	( cast : "Mickey Rooney" )	74

Hay una observación aquí, el actor con mayor longevidad Harrison Ford se trata de dos personas con el mismo nombre, por lo que Harrison Ford no es realmente el actor con la carrera más larga, esto se puede ver al inspeccionar los nombres de las películas, pero en cuanto a la funcionalidad del query, se esta haciendo lo que se pide en las instrucciones.

20. Sobre actors, guardar en nueva colección llamada “genres” realizando la fase \$unwind por genres. Después, contar cuantos documentos existen en la nueva colección

```

158 // 20. Sobre actors, guardar en nueva colección llamada "genres" realizando
159 // la fase $unwind por genres. Después, contar cuanto documentos existen en la
160 // nueva colección
161
162 db.actors.aggregate([
163   {$unwind: "$genres"},
164   {$project: { _id: 0, title:1, year:1, cast:1, genres:1}},
165   {$out: "genres"}])
166 db.genres.count();
167
168
169
170

```

Aggregate (10) x Console x Result x

3.487 s

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

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

```

171 db.genres.aggregate([
172   { $group: {
173     _id: {year: "$year", genre: "$genres"},
174     uniqueTitles: {$addToSet: "$title" }},
175     { $project: {
176       _id: 0,
177       year: "$_id.year",
178       genre: "$_id.genre",
179       Numero_de_peliculas: {$size: "$uniqueTitles"}},
180       {$sort: {Numero_de_peliculas: -1}},
181       {$limit: 5}}])

```

year	genre	Numero_de_peliculas
1919	Drama	291
1925	Drama	247
1924	Drama	233
1919	Comedy	226
1922	Drama	209

22. Sobre genres, mostrar los 5 actores los género en los que han participado actores con un mayor número de géneros diferentes, se debe mostrar el número de géneros diferentes que se ha interpretado. Se necesita previamente filtrar para descartar los actores llamados "Undefined"

```

183 // 22. Sobre genres, mostrar los 5 actores y los género en lo s que han participado
184 // con más número de géneros diferentes, se debe mostrar el número de géneros diferentes
185 // que se ha interpretado. Se necesita previamente filtrar para descartar los actores
186 // llamados undefined
187 db.genres.aggregate([
188   {$match: {cast: {$ne: "Undefined"}}},
189   {$group : {
190     _id: "$cast",
191     generos: {$addToSet: "$genres" } },
192   {$project: {
193     _id: 0,
194     actor: "$_id",
195     numgeneros: {$size: "$generos"},
196     generos: 1},
197     {$sort: {numgeneros: -1} },
198     {$limit: 5}}])

```

actor	numgeneros
Dennis Quaid	20

```

1 /* 1 */
2 {
3   "generos" : [
4     "Adventure",
5     "Romance",
6     "Disaster",
7     "Satire",
8     "Crime",
9     "Comedy",
10    "Musical",
11    "Thriller",
12    "Animated",
13    "Fantasy",
14    "Sports",
15    "Horror",
16    "Drama",
17    "Science Fiction",
18    "Western",
19    "Suspense",
20    "Action",
21    "Family",
22    "Biography",
23    "Dance"
24  ],
25   "actor" : "Dennis Quaid",
26   "numgeneros" : 20
27 },
28

```

23. Sobre genres, 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

```
200 // 23. Sobre genres, mostrar las 5 películas y su año correspondiente en los que más géneros
201 // diferentes han sido catalogados, mostrando esos géneros y el número de géneros que contiene.
202 db.genres.aggregate([
203   {$group: {
204     _id: {"title": "$title", "year": "$year"}, // agrupa por título
205     genres: {$addToSet: "$genres"}}}, // agrega géneros para obtener géneros únicos
206   {$project: {
207     _id: 0,
208     title: "$_id.title",
209     year: "$_id.year",
210     numgenres: {$size: "$genres"}, // Cuenta el número de géneros únicos
211     genres: 1}}, // incluye el género de la variable anterior
212   {$sort: { numgenres: -1 }}, // Ordenar
213   {$limit: 5}])
214
215
```

Aggregate (10) x Console x Result x Find x Find (1) x Aggregate x Aggregate (1) x Error x Aggregate (2) x Aggregate (3) x

genres 1.655 s 5 Docs

```
2 - {
3   "genres" : [
4     "Thriller",
5     "Crime",
6     "Action",
7     "Comedy",
8     "Drama",
9     "Historical",
10    "Biography"
11  ],
12  "title" : "American Made",
13  "year" : 2017,
14  "numgenres" : 7
15 },
16
17 /* 2 */
18 - {
19   "genres" : [
20     "Action",
21     "Drama",
22     "Fantasy",
23     "War",
24     "Superhero",
25     "Adventure"
26  ],
27  "title" : "Wonder Woman",
28  "year" : 2017,
29  "numgenres" : 6
30 }
```

24. Query libre de agregación. Contar el número total de películas por género

```
215 // 24. Contar el número total de películas por género
216 db.genres.aggregate([
217   {$group: {
218     _id: "$genres",
219     totalMovies: {$sum: 1}}}])
220
221
```

Aggregate (10) x Console x Result x Find x Find (1) x Aggregate x Aggregate (1) x Error x Aggregate (2) x Aggregate (3) x Aggregate (4) x

genres 0.573 s 42 Docs

	_id	totalMovies
1	Found Footage	1
2	Animated	2050 (2.0K)
3	Teen	162
4	Performance	55
5	Erotic	96
6	Spy	275

25. Listar las películas y su cantidad de actores excluyendo aquellos con "Undefined"

```
230 // 26. Listar las películas y su cantidad de actores excluyendo aquellas con "Undefined"
231
232 db.movies.aggregate([
233   let db: mongo.IDatabase { $ne: "Undefined" } } },
234   { $unwind: "$cast" },
235   { $group: {
236     _id: "$title",
237     actorCount: { $sum: 1 }
238   }},
239   { $sort: { actorCount: -1 } }
240 ]);
241
```

Aggregate (10) x Console x Result x Find x Find (1) x Aggregate x Aggregate (1) x Error x Aggregate (2) x Ag

movies 1.432 s Fetch Count

	_id	actorCount
1	The Three Musketeers	37
2	The Twilight Saga: Breaking	35
3	Anchorman 2: The Legend	33
4	Cars 2	32
5	The Unspeakable Act	30
6	Avengers: Infinity War	29

26. Encontrar el número de películas producidas desde 2018 hasta 2023

```
242 // 26. Encontrar número de películas producidas desde 2018 hasta 2023
243
244 db.movies.aggregate([
245   { $match: { year: { $gte: 2018 } } }, // Asume que el año actual es 2023
246   { $group: {
247     _id: null, // Agrupa todos los documentos coincidentes
248     numberOfMovies: { $sum: 1 } // Cuenta las películas
249   }},
250   { $project: {
251     _id: 0, // Excluye el campo _id en el resultado final
252     numberOfMovies: 1 // Incluye el recuento de películas
253   }}
254 ]);
255
```

Aggregate (10) x Console x Result x Find x Find (1) x Aggregate x Aggregate (1) x Error x Agg

movies 0.398 s 1 Doc

numberOfMovies

1 236