Bases de datos NoSQL

Hugo Gómez Sabucedo
hugogomezsabucedo@gmail.com

Máster Big Data, Data Science & Inteligencia Artificial

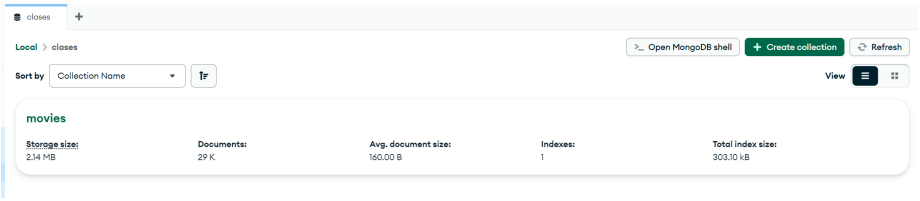
Curso 2024-2025

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.



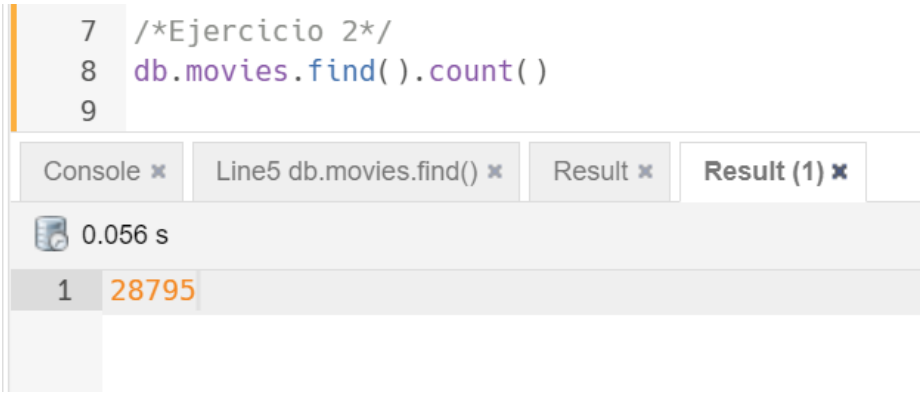
1. **Analizar con find la colección**.

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

movies 0.124 s 28.795 Docs

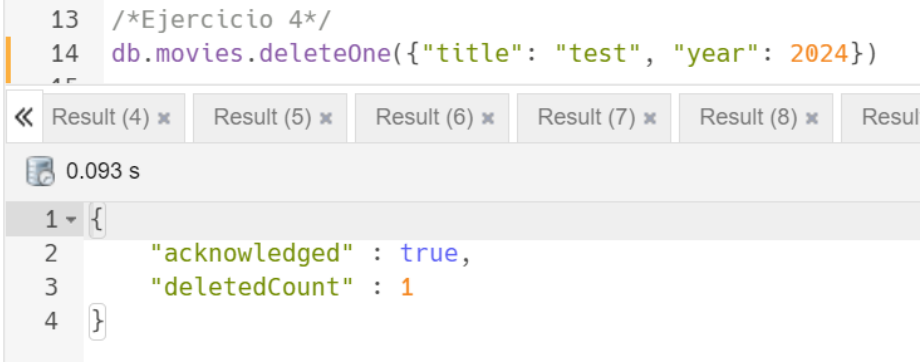
```
1  /* 1 createdAt:1/12/2024 17:35:16*/
2  {
3      "_id" : ObjectId("674c9044944caa30e978feb1"),
4      "title" : "Caught",
5      "year" : 1900,
6      "cast" : [ ],
7      "genres" : [ ]
8  },
9
10 /* 2 createdAt:1/12/2024 17:35:16*/
11 {
12     "_id" : ObjectId("674c9044944caa30e978feb2"),
13     "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".**

6. **Actualizar los documentos cuyo actor tenga el valor "and", sacando ese valor del array cast.**

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

8. **Actualizar todos los documentos que tengan el array cast vacío, añadiendo un nuevo elemento con el valor Undefined.**

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

10. **Actualizar todos los documentos que tengan el array genres vacío, añadiendo un nuevo elemento con el valor Undefined.**

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

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

0.091 s

```
1 986
```

```
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

```
{
    "acknowledged" : true,
    "matchedCount" : 986,
    "modifiedCount" : 986
}
```

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

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

0.082 s

```
1 901
```

```
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

```
{
    "acknowledged" : true,
    "matchedCount" : 901,
    "modifiedCount" : 901
}
```

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

```
/*Ejercicio 11*/
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

```
{
    "year" : 2018
}
```

12. **Contar cuantas películas han salido en los últimos 20 años, desde el último año quee se tienen películas en la colección.**

```
db.movies.aggregate([
    { // Encontrar el año mas reciente
        $group: {
            _id: null,
            maxYear: { $max: "$year" }
        }
    },
    { // Encontrar el inicio del intervalo
        $addFields: {
            minYear: { $subtract: ["$maxYear", 19] }
        }
    },
    { // Obtenemos las peliculas en el intervalo
        $lookup: {
            from: "movies",
            let: { minYear: "$minYear", maxYear: "$maxYear" }, // "Pasamos" las variables al lookup
            pipeline: [
                {
                    $match: {
                        $expr: { $and: [ { $gte: ["$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

movies | 0.159 s | 1 Doc

```
{
    "_id" : null,
    "total" : 4787
}
```

13. **Contar cuantas películas han salido en la década de los 60.**

```
/*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**.

15. **Mostrar el año/años con menos películas**.

16. **Guardar en una nueva colección llamada "actors", haciendo $unwind por actor. Contar cuantos elementos existen en la nueva colección**.

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**.

```
/*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"
        }
    }
])
```

(6) × Aggregate (30) × Aggregate (31) × Aggregate (32) × Aggregate (33) × Aggreg

movies 0.034 s 1 Doc

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

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

19. **Sobre actors, mostrar los 5 actores cuya carrera haya sido la más larga, mostrando cuando comenzó, cuando finalizó y cuántos años ha**

```
/*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
    }
]
```

trabajado.

20. **Sobre actors, guardar en una nueva colección llamada "genres" reali-**

```
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
```

zando la fase $unwind por genres. Contar cuantos elementos existen en al nueva colección.

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

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.**

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.**

```
/*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
    }
]
```

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**.

25. **Ejercicio libre. Sobre actors, mostrar el actor más popular en cada género (es decir, el que más veces ha actuado)**.

26. **Ejercicio libre. Sobre genres, mostrar el génenro más popular en cada década, mostrando su nombre y el total de películas**.

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" }
        }
    },
```

```
    { // Encontrar el inicio del intervalo
        $addFields: {
            minYear: { $subtract: ["$maxYear", 19] }
        }
    },
    { // Obtenemos las peliculas en el intervalo
        $lookup: {
            from: "movies",
            let: { minYear: "$minYear", maxYear: "$maxYear"
            }, // "Pasamos" las variables al lookup
            pipeline: [
                {
                    $match: {
                        $expr: {
                            $and: [
                                { $gte: ["$year", "$$minYear
                                "] },
                                { $lte: ["$year", "$$maxYear
                                "] }
                            ]
                        }
                    }
                }
            ],
            as: "peliculasIntervalo"
        }
    },
    { // Anhadimos un campo con el total de las peliculas
        $addFields: {
            total: { $size: "$peliculasIntervalo" }
        }
    },
    { //Seleccionamos solo los campos de interes
        $project: {
            _id: 1,
            total: 1
        }
    }
])

/*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 }
```

```
        }
    }
]);

/*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 }
        }
```

```
141     },
142     { // Encontramos el minimo
143         $group: {
144             _id: null,
145             minPelis: { $min: "$pelis" },
146             years:
147             {  // Guardamos los anhos y su total de
                peliculas
148                 $push: { year: "$_id", pelis: "$pelis" }
149             }
150         }
151     },
152     { // Obtenemos un documento por anho
153         $unwind: "$years"
154     },
155     { // Filtramos por el anho con el minimo de peliculas
156         $match: { $expr: { $eq: ["$years.pelis", "$minPelis"
            ] } }
157     },
158     { // Ajustamos el reesultado
159         $project: {
160             _id: "$years.year",
161             pelis: "$years.pelis"
162         }
163     }
164 ]);
165
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
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 ]);
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 }
```

```
236     },
237     { // Nos quedamos con el maximo
238         $limit: 1
239     }
240 ]);
241 
242 /*Ejercicio 20*/
243 db.actors.aggregate([
244     { // Hacemos el unwind
245         $unwind: "$genres"
246     },
247     { // Eliminamos el id
248         $project: { _id: 0 }
249     },
250     { // Guardamos en la coleccion genres
251         $out: "genres"
252     }
253 ]);
254 db.genres.count();
255 
256 /*Ejercicio 21*/
257 db.genres.aggregate([
258     { // Agrupamos por anho y genero y hallamos las
    peliculas distintas
259         $group: {
260             _id: { year: "$year", genre: "$genres" },
261             pelisUnicas: { $addToSet: "$title" }
262         }
263     },
264     { // Anhadimos el contador de pelis unicas
265         $project:{
266             _id: 1,
267             pelis: { $size: "$pelisUnicas" }
268         }
269     },
270     { // Ordenamos por el contador descendente
271         $sort: { pelis: -1 }
272     },
273     { // Nos quedamos con los cinco primeros
274         $limit: 5
275     }
276 ]);
277 
278 /*Ejercicio 22*/
279 db.genres.aggregate([
280     { // Eliminamos los actores "Undefined"
281         $match: {
282             cast: { $ne: "Undefined" }
283         }
284     },
```

```
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 ]);
```