NOSQL

Diego Olalla Carrión
53957557V

Ejercicios

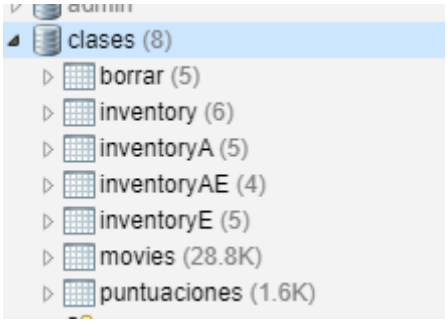
0. Realizar la importación del json en una colección llamada “movies”.

La query realizada es :

use clases

show collections

El resultado es:

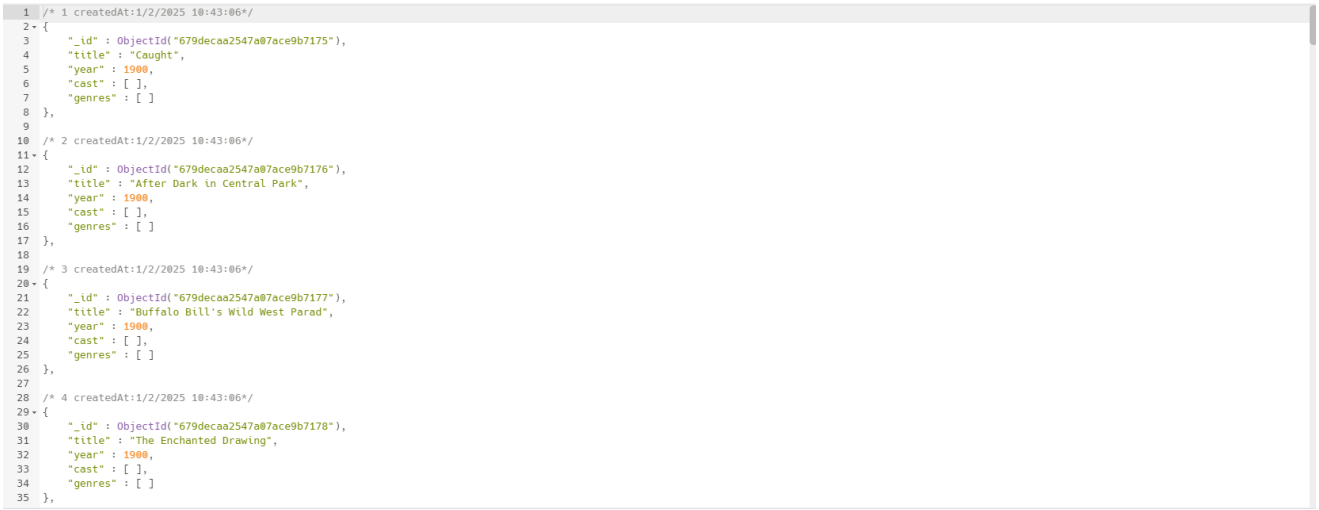


1. **Analizar con find la colección**

La query ejecutada es:

db.movies.find()

Nos arroja como resultado en formato Json, los siguientes resultados:

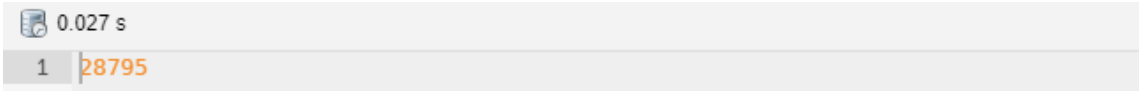


2. **Contar cuántos documentos (películas) tiene cargado.**

La query ejecutada es:

db.movies.count()

Después de realizar la query, el resultado obtenido es:



3. **Insertar una película.**

La query ejecutada es:

db.movies.insertOne({title: "Everything Everywhere All at Once",year: 2022, cast: ["Michelle Yeoh", "Ke Huy Quan", "Stephanie Hsu", "Jamie Lee Curtis"],genres: ["Comedy"],})

db.movies.find({title: "Everything Everywhere All at Once"})

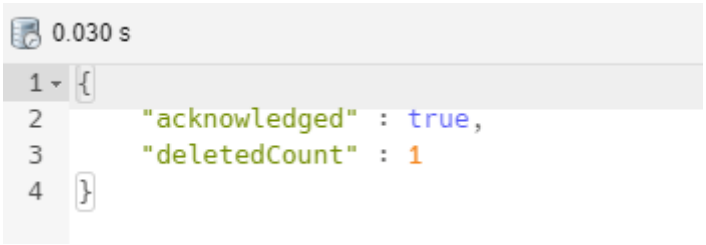
Los resultados obtenidos:

4. **Borrar la película insertada en el punto anterior (en el 3).** La query ejecutada es:

var query= {title:"Everything Everywhere All at Once"}

db.movies.deleteOne(query)

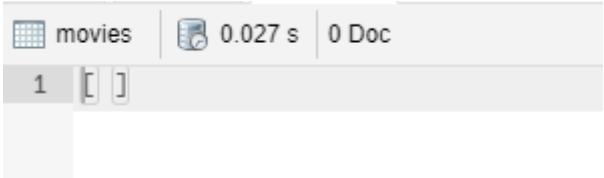
Después de realizar la query, el resultado obtenido es:



Después de ejecutarlo, revisamos si la película sigue en la base de datos:

db.movies.find({title:"Everything Everywhere All at Once"})

Como resultado:



Si el resultado es vacío ([]), significa que la película fue eliminada con éxito.

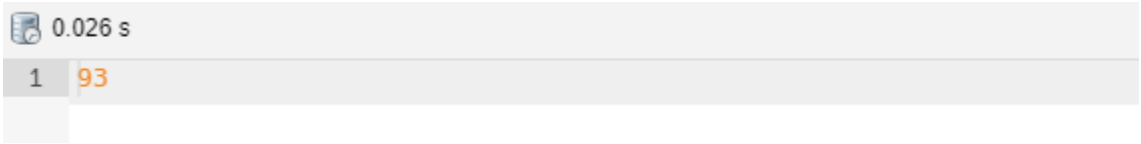
5. **Contar cuantas películas tienen actores (cast) que se llaman “and”.**

La query ejecutada es:

var query = {cast:"and"}

db.movies.find(query).count()

Después de realizar la query, el resultado obtenido es:

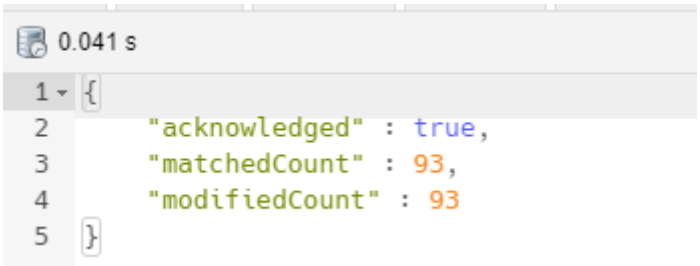


6. **Actualizar los documentos cuyo actor (cast) tenga por error el valor “and” como si realmente fuera un actor. Para ello, se debe sacar únicamente ese valor del array cast. Por lo tanto, no se debe eliminar ni el documento (película) ni su array cast con el resto de actores.**

La query ejecutada es:

db.movies.updateMany({},{$pull: {cast:{$in: ["and"]}}})

Después de realizar la query, el resultado obtenido es:



Comprobamos que no se ha eliminado ni el documento ni su array cast con el resto de actores:

db.movies.find({ title: "Inception" })

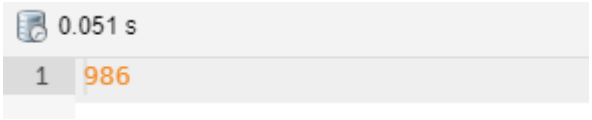
y su resultado es:

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

La query ejecutada es:

db.movies.find({ cast: { $size: 0 } }).count()

Después de realizar la query, el resultado obtenido es:



Este código cuenta cuántas películas no tienen actores en cast. Si el número devuelto es mayor que 0, significa que hay películas sin actores en la base de datos.

8. **Actualizar TODOS los documentos (películas) que tengan el array cast vacío, añadiendo un nuevo elemento dentro del array con valor Undefined. Cuidado! El tipo de cast debe seguir siendo un array. El array debe ser así -> ["Undefined"].**

La query ejecutada es:

db.movies.updateMany({cast: {$size: 0}}, {$set: {cast: ["Undefined"]}})

Después de realizar la query, el resultado obtenido es:

0.044 s

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

Comprobamos que se ha añadido este valor al array de Cast, ejecutando la query db.movies.find({ title: "Caught" })

Debería de mostrar el valor de Undefined ya que no tenía ningún actor.

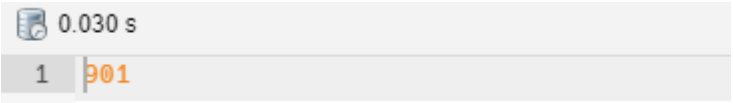
```
/* 1 createdAt:1/2/2025 11:05:59*/
{
    "_id" : ObjectId("679df2072547a07ace9be1f1"),
    "title" : "Caught",
    "year" : 1900,
    "cast" : [ "Undefined" ],
    "genres" : [ ]
},
```

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

La query ejecutada es:

db.movies.find({genres: {$size: 0}}).count()

Después de realizar la query, el resultado obtenido es:



10. Actualizar TODOS los documentos (películas) que tengan el array genres vacío, añadiendo un nuevo elemento dentro del array con valor Undefined.

La query ejecutada es:

db.movies.updateMany({ genres: { $size: 0 } }, { $set: { genres: ["Undefined"]}})

Después de realizar la query, el resultado obtenido es:

0.033 s

Key	Value	Type
(1)	{ acknowledged : true, matchedCount : 901, modifiedCount : 901 }	Object
acknowledged	true	Bool
matchedCount	901	Int32
modifiedCount	901	Int32

Y comprobamos también con la película Caught que tampoco tenía género, ejecutando la query db.movies.find({ title: "Caught" })

y muestra como resultado:

movies | 0.026 s | 3 Docs

```
/* 1 createdAt:1/2/2025 11:05:59*/
{
    "_id" : ObjectId("679df2072547a07ace9be1f1"),
    "title" : "Caught",
    "year" : 1900,
    "cast" : [ "Undefined" ],
    "genres" : [ "Undefined" ]
},

```

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

La query ejecutada es:

db.movies.aggregate([{ $sort: { year: -1 } }, { $project: { year: 1, _id: 0 }}])

Después de realizar la query, el resultado obtenido es:

	year
1	2018
2	2018
3	2018
4	2018
5	2018
6	2018
7	2018
8	2018
9	2018

El año mas reciente de movies.json es de 2018.

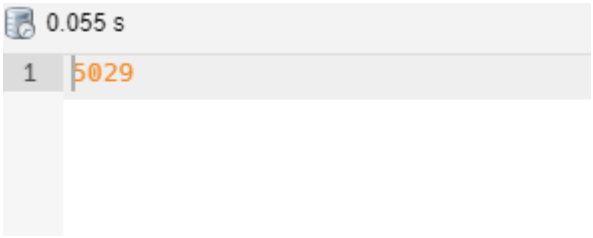
12. Contar cuántas películas han salido en los últimos 20 años. Debe hacerse desde el último año que se tienen registradas películas en la colección, mostrando el resultado total de esos años.

Como el año más reciente es 2018, a este le restamos 20 años y como resultado se obtiene que se debe contar las películas comprendidas entre 1998 y 2018.

La query ejecutada es:

db.movies.aggregate().match({year:{$gte:1998}}).project({year:1}).sort({year:-1}).count()

Después de realizar la query, el resultado obtenido es:

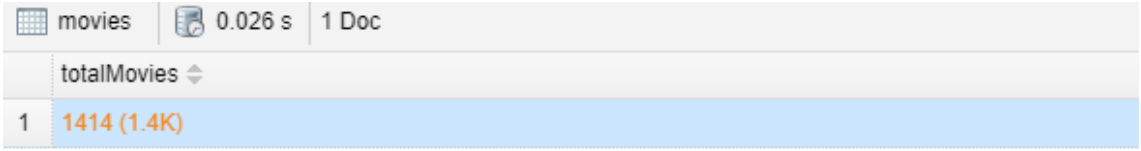


13. **Contar cuántas películas han salido en la década de los 60 (del 60 al 69 incluidos).**

La query ejecutada es:

db.movies.aggregate([{ $match: { year: { $gte: 1960, $lte: 1969 } } }, { $count: "totalMovies" }])

Después de realizar la query, el resultado obtenido es:



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

La query ejecutada es:

db.movies.aggregate([

{ $group: { _id: "$year", count: { $sum: 1 } } }

{ $sort: { count: -1 } }

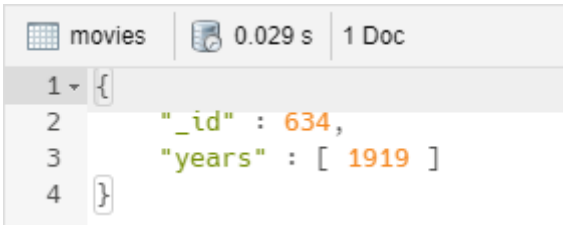
{ $group: { _id: "$count", years: { $push: "$_id" } } }

{ $sort: { _id: -1 } }

{ $limit: 1 }

])

Después de realizar la query, el resultado obtenido es:

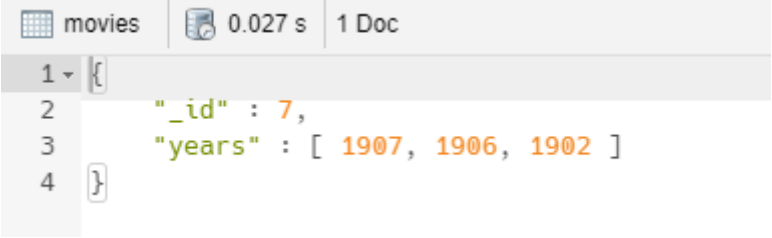


15. **Mostrar el año u años con menos películas mostrando el número de películas de ese año. Revisar si varios años pueden compartir tener el menor número de películas.**

La query ejecutada es

```
db.movies.aggregate([
  { $group: { _id: "$year", count: { $sum: 1 } } }
  { $sort: { count: 1 } }
  { $group: { _id: "$count", years: { $push: "$_id" } } }
  { $sort: { _id: 1 } }
  { $limit: 1 }
])
```

Después de realizar la query, el resultado obtenido es:

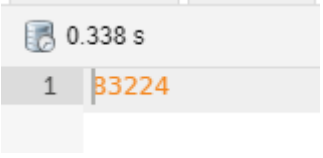


16. **Guardar en nueva colección llamada “actors” realizando la fase $unwind por actor. Después, contar cuantos documentos existen en la nueva colección.**

La query ejecutada es:

```
db.movies.aggregate([
{$unwind: "$cast"},
{$project: {_id:0,actor:"$cast",title:1,year:1, genres:"$genres"}},
{$out: "actors"}
])
db.actors.countDocuments()
```

Después de realizar la query, el resultado obtenido es:



17. **Sobre actors (nueva colección), mostrar la lista con los 5 actores que han participado en más películas mostrando el número de películas en las que ha participado.**

La query ejecutada es:

```
db.actors.aggregate([
  { $match: { actor: { $exists: true, $ne: "", $ne: "Undefined" } } }
  { $group: { _id: "$actor", totalMovies: { $sum: 1 } } }
  { $sort: { totalMovies: -1 } }
  { $limit: 5 }
])
```

Después de realizar la query, el resultado obtenido es:

actors | 0.059 s | 5 Docs

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

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

La query ejecutada es:

```
db.actors.aggregate([
  {
    $group: {
      _id: { title: "$title", year: "$year" }
      totalActors: { $sum: 1 }
    }
  },
  { $sort: { totalActors: -1 } }
  { $limit: 5 }
])
```

Después de realizar la query, el resultado obtenido es:

actors | 0.091 s | 5 Docs

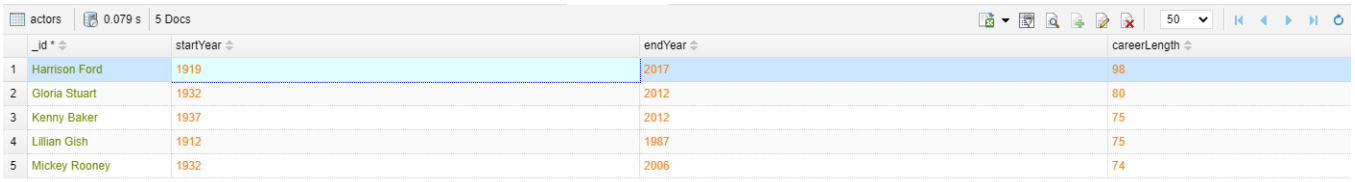
	_id		
	title	year	totalActors
1	The Twilight Saga: Breaking Dawn - Part 2	2012	35
2	Anchorman 2: The Legend Continues	2013	33
3	Cars 2	2011	32
4	Avengers: Infinity War	2018	29
5	Grown Ups 2	2013	28

19. Sobre actors (nueva colección), mostrar los 5 actores cuya carrera haya sido la más larga.

La query ejecutada es:

```
db.actors.aggregate([
    { $match: { actor: { $exists: true, $ne: "", $ne: "Undefined" } } }
    { $group: { _id: "$actor", startYear: { $min: "$year" }, endYear: { $max: "$year" } } }
    { $project: { startYear: 1, endYear: 1, careerLength: { $subtract: ["$endYear", "$startYear"] } } }
    { $sort: { careerLength: -1 } }
    { $limit: 5 }
])
```

Después de realizar la query, el resultado obtenido es:



actors | 0.079 s | 5 Docs

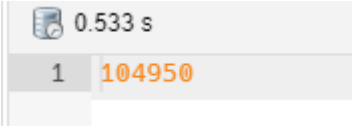
	_id *	startYear	endYear	careerLength
1	Harrison Ford	1919	2017	98
2	Gloria Stuart	1932	2012	80
3	Kenny Baker	1937	2012	75
4	Lillian Gish	1912	1987	75
5	Mickey Rooney	1932	2006	74

20. Sobre actors (nueva colección), Guardar en nueva colección llamada “genres” realizando la fase $unwind por genres

La query ejecutada es:

```
db.actors.aggregate([{ $match: { genres: { $exists: true, $ne: [] } } },{ $unwind: "$genres" },{ $project: { _id: 0, actor: 1, title: 1, year: 1, genre: "$genres" } },
    { $out: "genres" }])
db.genres.countDocuments()
```

Después de realizar la query, el resultado obtenido es:

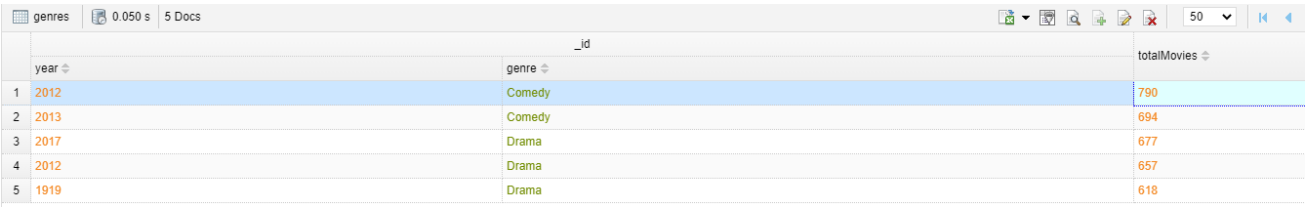


21. **Sobre genres (nueva colección), mostrar los 5 documentos agrupados por "Año y Género" que más número de películas diferentes tienen mostrando el número total de películas.**

La query ejecutada es:

```
db.genres.aggregate([
    { $group: { _id: { year: "$year", genre: "$genre" }, totalMovies: { $sum: 1 } } }
    { $sort: { totalMovies: -1 } }
    { $limit: 5 }
])
```

Después de realizar la query, el resultado obtenido es:



	_id		totalMovies
	year	genre	
1	2012	Comedy	790
2	2013	Comedy	694
3	2017	Drama	677
4	2012	Drama	657
5	1919	Drama	618

22. **Sobre genres (nueva colección), mostrar los 5 actores y los géneros en los que han participado con más número de géneros diferentes, se debe mostrar el número de géneros diferentes que ha interpretado**

La query ejecutada es:

```
db.genres.aggregate([
    { $match: { actor: { $exists: true, $ne: "", $ne: "Undefined" } } }
    { $group: { _id: { actor: "$actor", genre: "$genre" } } }
    { $group: { _id: "$_id.actor", numGeneros: { $sum: 1 }, generos: { $addToSet: "$_id.genre" } } }
    { $sort: { numGeneros: -1 } }
    { $limit: 5 }
])
```

Después de realizar la query, el resultado obtenido es:

genres | 0.242 s | 5 Docs

	_id *	numGeneros	generos
1	Dennis Quaid	20	Array[20]
2	Michael Caine	19	Array[19]
3	James Mason	19	Array[19]
4	Gene Hackman	18	Array[18]
5	James Coburn	18	Array[18]


```
/* 1 */
{
    "_id" : "Dennis Quaid",
    "numGeneros" : 20,
    "generos" : [
        "Science Fiction",
        "Sports",
        "Animated",
        "Drama",
        "Western",
        "Adventure",
        "Disaster",
        "Satire",
        "Family",
        "Musical",
        "Fantasy",
        "Comedy",
        "Biography",
        "Dance",
        "Romance",
        "Horror",
        "Thriller",
        "Crime",
        "Suspense",
        "Action"
    ]
},
```


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

La query ejecutada es:

db.genres.aggregate([

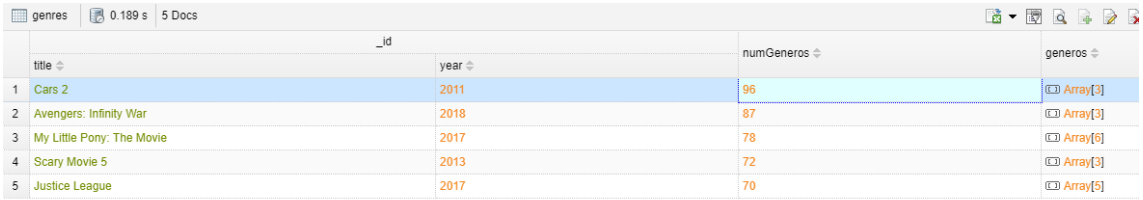
{ $group: { _id: { title: "$title", year: "$year" }, numGeneros: { $sum: 1 }, generos: { $addToSet: "$genre" } } }

{ $sort: { numGeneros: -1 } }

{ $limit: 5 }

])

Después de realizar la query, el resultado obtenido es:



	_id		numGeneros	generos
	title	year		
1	Cars 2	2011	96	Array[3]
2	Avengers: Infinity War	2018	87	Array[3]
3	My Little Pony: The Movie	2017	78	Array[6]
4	Scary Movie 5	2013	72	Array[3]
5	Justice League	2017	70	Array[5]

genres | 0.189 s | 5 Docs

```
/* 1 */
{
    "_id" : {
        "title" : "Cars 2",
        "year" : 2011
    },
    "numGeneros" : 96,
    "generos" : [ "Family", "Spy", "Animated" ]
},

/* 2 */
{
    "_id" : {
        "title" : "Avengers: Infinity War",
        "year" : 2018
    },
    "numGeneros" : 87,
    "generos" : [ "Adventure", "Superhero", "Action" ]
},

/* 3 */
{
    "_id" : {
        "title" : "My Little Pony: The Movie",
        "year" : 2017
    },
    "numGeneros" : 78,
    "generos" : [
        "Animated",
        "Fantasy",
        "Musical",
        "Adventure",
        "Comedy",
```

24. Identificar los géneros con mayor evolución en popularidad a lo largo del tiempo

Determinar qué géneros han tenido el mayor crecimiento en número de películas a lo largo de los años, mostrando su evolución y el periodo de mayor auge y mostrar los 5 género que más evolución han tenido.

La query ejecutada es:

db.genres.aggregate([

{ $group: { _id: { year: "$year", genre: "$genre" }, totalMovies: { $sum: 1 } } }

{ $sort: { "_id.genre": 1, "_id.year": 1 } }

{ $group: { _id: "$_id.genre", yearlyCounts: { $push: { year: "$_id.year", count: "$totalMovies" } }, totalMovies: { $sum: "$totalMovies" } } }

{ $project: { genre: "$_id", yearlyCounts: 1, totalMovies: 1, firstYear: { $arrayElemAt: ["$yearlyCounts.year", 0] }, lastYear: { $arrayElemAt: ["$yearlyCounts.year", -1] } } }

{ $sort: { totalMovies: -1 } }

{ $limit: 5 }

])

Después de realizar la query, el resultado obtenido es:


genres | 0.057 s | 5 Docs

	_id *	yearlyCounts	totalMovies	genre	firstYear	lastYear
1	Drama	Array[112]	26.230 (26.2K)	Drama	1906	2018
2	Comedy	Array[116]	23.307 (23.3K)	Comedy	1900	2018
3	Western	Array[106]	7292 (7.3K)	Western	1903	2018
4	Crime	Array[108]	4781 (4.8K)	Crime	1906	2018
5	Action	Array[96]	4675 (4.7K)	Action	1906	2018

25. Encontrar el primer año en el que se registró cada género

Identificar el año más antiguo en el que aparece cada género en la base de datos.

La query ejecutada es:

db.genres.aggregate([

{ $match: { genre: { $exists: true, $ne: "", $ne: "Undefined" }, year: { $exists: true } } }

{ $group: { _id: "$genre", firstYear: { $min: "$year" } } }

{ $sort: { firstYear: 1 } }

])

Después de realizar la query, el resultado obtenido es:

genres | 0.054 s | 41 Docs

	_id *	firstYear
1	Short	1900
2	Documentary	1900
3	Comedy	1900
4	Western	1903
5	Romance	1905
6	Adventure	1905
7	Action	1906
8	Drama	1906
9	Crime	1906
10	Animated	1906
11	Biography	1907
12	Historical	1907
13	Horror	1908
14	Fantasy	1908
15	Sports	1911
16	Silent	1911
17	Thriller	1911
18	War	1912
19	Mystery	1914
20	Spy	1917
21	Erotic	1917
22	Family	1923
23	Science Fiction	1923
24	Musical	1928
25	Sport	1929
26	Noir	1931
27	Suspense	1939
28	Live Action	1941
29	Disaster	1957
30	Teen	1960

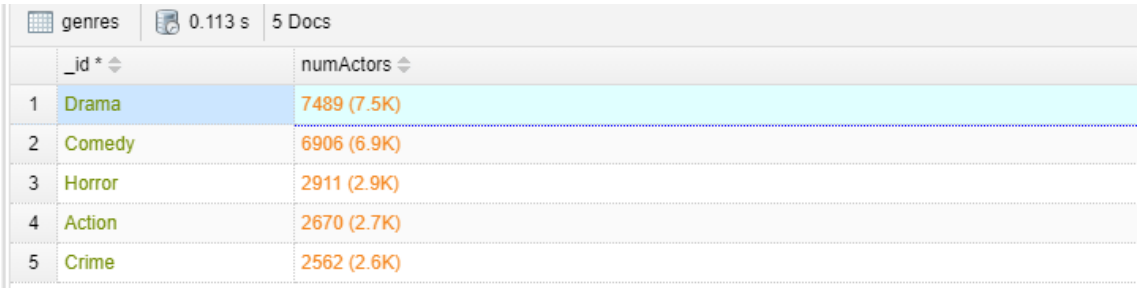
26. Identificar los géneros con mayor cantidad de actores únicos.

Encontrar los géneros cinematográficos que han contado con la mayor diversidad de actores a lo largo del tiempo.

La query ejecutada es:

```
db.genres.aggregate([
    { $match: { genre: { $exists: true, $ne: "", $ne: "Undefined" }, actor: { $exists: true, $ne: "", $ne: "Undefined" } } }
    { $group: { _id: "$genre", uniqueActors: { $addToSet: "$actor" } } }
    { $project: { numActors: { $size: "$uniqueActors" } } }
    { $sort: { numActors: -1 } }
    { $limit: 5 }
])
```

Después de realizar la query, el resultado obtenido es:



genres | 0.113 s | 5 Docs

	_id *	numActors
1	Drama	7489 (7.5K)
2	Comedy	6906 (6.9K)
3	Horror	2911 (2.9K)
4	Action	2670 (2.7K)
5	Crime	2562 (2.6K)