



U N I V E R S I D A D
COMPLUTENSE
M A D R I D

MÁSTER DATA SCIENCE, BIG DATA & BUSINESS ANALYTICS

BASE DE DATOS NO SQL

ENTREGABLE:

TAREA.

PROFESOR:

ÁLVARO BRAVO ACOSTA

ALUMNA:

NAIRA CARRUCCIO VILLADA

ÍNDICE

EJERCICIOS

DEL 1 AL 10 **3**

DEL 11 AL 23 **9**

QUERY LIBRE DEL 24 AL 26 **24**

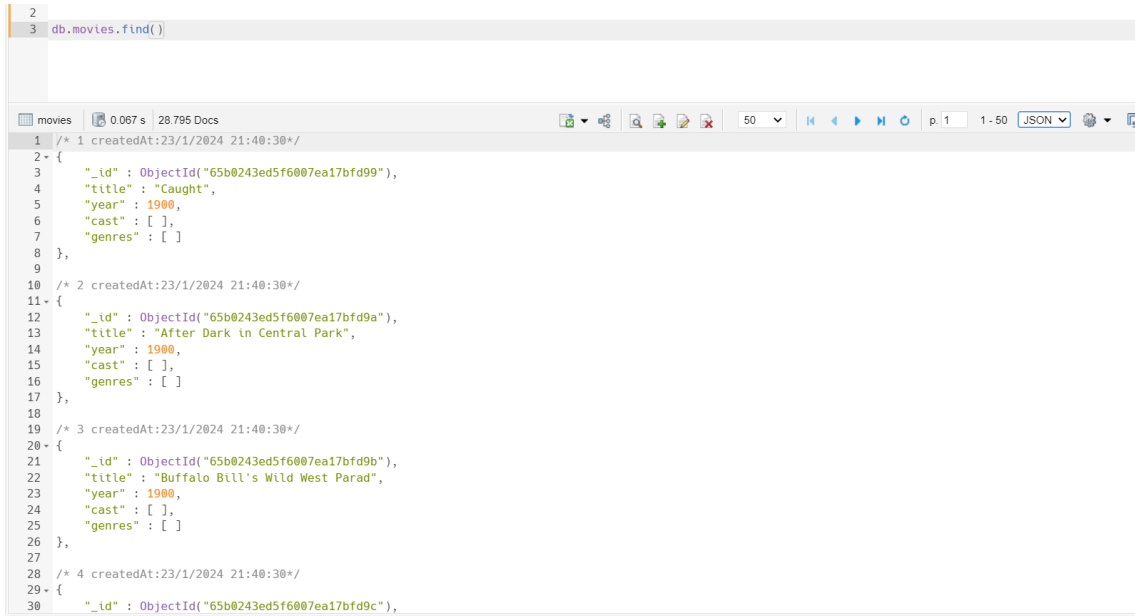
EJERCICIOS :

DEL 1 AL 10

0. Realizar la importación del json en una colección llamada movies.
1. Analizar con find la colección. Se muestra una salida de ejemplo, el resultado es erróneo.

db.movies.find()

```
2
3 db.movies.find()
```



```
1 /* 1 createdAt:23/1/2024 21:40:30*/
2 {
3   "_id" : ObjectId("65b0243ed5f6007ea17bfd99"),
4   "title" : "Caught",
5   "year" : 1900,
6   "cast" : [ ],
7   "genres" : [ ]
8 },
9
10 /* 2 createdAt:23/1/2024 21:40:30*/
11 {
12   "_id" : ObjectId("65b0243ed5f6007ea17bfd9a"),
13   "title" : "After Dark in Central Park",
14   "year" : 1900,
15   "cast" : [ ],
16   "genres" : [ ]
17 },
18
19 /* 3 createdAt:23/1/2024 21:40:30*/
20 {
21   "_id" : ObjectId("65b0243ed5f6007ea17bfd9b"),
22   "title" : "Buffalo Bill's Wild West Parad",
23   "year" : 1900,
24   "cast" : [ ],
25   "genres" : [ ]
26 },
27
28 /* 4 createdAt:23/1/2024 21:40:30*/
29 {
30   "_id" : ObjectId("65b0243ed5f6007ea17bfd9c"),
```

2. Contar cuántos documentos (películas) tiene cargado. Se muestra una salida de ejemplo, el resultado es erróneo.

db.movies.find().count()

```

4 //2)
5 db.movies.find().count()

```

0.036 s

1	28795
---	-------

3. Insertar una película. Se muestra una salida de ejemplo, el resultado es erróneo

```

var nueva_movie = { "title": "Twilight",
  "year": 2008,
  "cast": ["Kristen Stewart", "Robert Pattinson", "Taylor Lautner"],
  "genres": ["Romance", "Fantasy"]
}

db.movies.insertOne(nueva_movie)

db.movies.find()

```

```

7 var nueva_movie = { "title": "Twilight",
8   "year": 2008,
9   "cast": ["Kristen Stewart", "Robert Pattinson", "Taylor Lautner"],
10  "genres": ["Romance", "Fantasy"]
11 }
12 db.movies.insertOne(nueva_movie)
13 db.movies.find()
14
15 //4)
16 var nueva_movie = { "title": "Twilight",

```

Line3 db.movies.find() ✕ Line5 db.movies.find().cou... ✕ Line12 db.movies.insertOne(... ✕ Line13 db.movies.find() ✕ Line21 db.movies.deleteMany... ✕ Result ✕

0.043 s

Key	Value	Type
(1)	{ acknowledged: true, insertedId: ObjectId("65b24d9a1a152faf0cb00a19") }	Object
acknowledged	true	Bool
insertedId	65b24d9a1a152faf0cb00a19	ObjectId

4. Borrar la película insertada en el punto anterior (en el 3). Se muestra una salida de ejemplo, el resultado es erróneo.

```

var nueva_movie = { "title": "Twilight",
  "year": 2008,
  "cast": ["Kristen Stewart", "Robert Pattinson", "Taylor Lautner"],
  "genres": ["Romance", "Fantasy"]
}

```

db.movies.deleteMany(nueva_movie)

db.movies.find(nueva_movie)

```
15 //4)
16 var nueva_movie = { "title" : "Twilight",
17   "year" : 2008,
18   "cast" : ["Kristen Stewart", "Robert Pattinson", "Taylor Lautner" ],
19   "genres" : ["Romance", "Fantasy"]
20 }
21 db.movies.deleteMany(nueva_movie)
22 db.movies.find(nueva_movie)
23
24 //5)
```

Line3 db.movies.find() ✖ Line5 db.movies.find().cou... ✖ Line12 db.movies.insertOne(... ✖ Line13 db.movies.find() ✖ Line21 db.movies.deleteMany... ✖ Result ✖ Result (1) ✖

0.071 s

y	Value	Type
(1)	{ acknowledged : true, deletedCount : 1 }	Object
acknowledged	true	Bool
deletedCount	1	Int32

```
15 //4)
16 var nueva_movie = { "title" : "Twilight",
17   "year" : 2008,
18   "cast" : ["Kristen Stewart", "Robert Pattinson", "Taylor Lautner" ],
19   "genres" : ["Romance", "Fantasy"]
20 }
21 db.movies.deleteMany(nueva_movie)
22 db.movies.find(nueva_movie)
23
24 //5)
```

Line3 db.movies.find() ✖ Line5 db.movies.find().cou... ✖ Line12 db.movies.insertOne(... ✖ Line13 db.movies.find() ✖ Line21 db.movies.deleteMany... ✖ Line22 db.movies.find(nueva... ✖

io to line 22 movies 0 Doc

No records found

5. Contar cuantas películas tienen actores (cast) que se llaman “and”. Estos nombres de actores están por ERROR. Se muestra una salida de ejemplo, el resultado es erróneo

var cast1 = {"cast" : "and" }

db.movies.count(cast1)

```
24 //5)
25 var cast1 = {"cast" : "and" }
26 db.movies.count(cast1)
27
28
29
30
```

Result (2) ✖ Error (1) ✖ Result (3) ✖ Console (1) ✖ Result (4) ✖

0.063 s

1	93
---	----

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. Se muestra una salida de ejemplo, el resultado es erróneo.

```
var query = {}
```

```
var castand = {$pull: {"cast": "and"}}
```

```
db.movies.updateMany(query, castand)
```

```
db.movies.find()
```

```
28 //6)
29 var query = {}
30 var castand = {$pull: {"cast": "and"}}
31 db.movies.updateMany(query, castand)
32 db.movies.find()
33
34
35
36
37
```

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

0.572 s

Key	Value	Type
(1)	{ acknowledged: true, matchedCount: 28795, modifiedCount: 93 }	Object
acknowledged	true	Bool
matchedCount	28.795 (28.8K)	Int32
modifiedCount	93	Int32

7. Contar cuantos documentos (películas) tienen el array ‘cast’ vacío. Se muestra una salida de ejemplo, el resultado es erróneo.

```
var cast2 = {"cast": [] }
```

```
db.movies.count(cast2)
```

```
34
35 //7)
36
37 var cast2 = {"cast": [] }
38 db.movies.count(cast2)
39
40
```

Find x Console x Result x Find (1) x Console (1) x Result (1) x

0.038 s

1 986

JSON

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"]. Se muestra una salida de ejemplo, el resultado es erróneo.

```
var query = { "cast" : [] }
```

```
var operacion = { $addToSet: { "cast" : "Undefined" } }
```

```
db.movies.updateMany(query, operacion)
```

```
db.movies.find()
```

```
39
40 //8)
41 var query = { "cast" : [] }
42 var operacion = { $addToSet: { "cast" : "Undefined" } }
43 db.movies.updateMany(query, operacion)
44 db.movies.find()
45
46
47
```

es.find(nueva... ✕ Line26 db.movies.count(cast... ✕ Line32 db.movies.updateMany... ✕ Line33 db.movies.find() ✕ Line38 db.movies.count(cast... ✕ Line43 db.movies.updateMany... ✕ Line44 db.movies.find() ✕

Go to line 43

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

```
40 //8)
41 var query = { "cast" : [] }
42 var operacion = { $addToSet: { "cast" : "Undefined" } }
43 db.movies.updateMany(query, operacion)
44 db.movies.find()
45
46
47
```

es.find(nueva... ✕ Line26 db.movies.count(cast... ✕ Line32 db.movies.updateMany... ✕ Line33 db.movies.find() ✕

Go to line 44 movies Fetch Count

```
1 /* 1 createdAt:25/1/2024 17:12:33*/
2 {
3   "_id" : ObjectId("65b2887117e0db2ad4538239"),
4   "title" : "Caught",
5   "year" : 1900,
6   "cast" : [ "Undefined" ],
7   "genres" : [ ]
8 },
9
10 /* 2 createdAt:25/1/2024 17:12:33*/
11 {
12   "_id" : ObjectId("65b2887117e0db2ad453823a"),
```

9. Contar cuantos documentos (películas) tienen el array genres vacío. Se muestra una salida de ejemplo, el resultado es erróneo

```
var genres1 = {"genres": [] }  
db.movies.count(genres1)
```

```
46 //9)  
47 var genres1 = {"genres": [] }  
48 db.movies.count(genres1)  
49  
50  
51
```

Console x Result x

0.069 s

1 901

10. Actualizar TODOS los documentos (películas) que tengan el array genres vacío, añadiendo un nuevo elemento dentro del array con valor Undefined. Cuidado! El tipo de genres debe seguir siendo un array. El array debe ser así -> ["Undefined"]. Se muestra una salida de ejemplo, el resultado es erróneo

```
var genres1 = {"genres": [] }  
  
var operacion = { $addToSet: {"genres": "Undefined" } }  
  
db.movies.updateMany(genres1, operacion)  
  
db.movies.find()
```

```
50 //10)  
51 var genres1 = {"genres": [] }  
52 var operacion = { $addToSet: {"genres": "Undefined" } }  
53 db.movies.updateMany(genres1, operacion)  
54 db.movies.find()  
55  
56  
57  
58
```

Console x Result x Result (1) x

0.145 s

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


```

50 //10)
51 var genres1 = {"genres" : [] }
52 var operacion = {$addToSet: {"genres" : "Undefined" }}
53 db.movies.updateMany(genres1, operacion)
54 db.movies.find()
55
56

```

Console x Result x Result (1) x Find x

movies 0.038 s 28.795 Docs

```

1 /* 1 createdAt:25/1/2024 17:12:33*/
2 {
3   "_id" : ObjectId("65b2887117e0db2ad4538239"),
4   "title" : "Caught",
5   "year" : 1900,
6   "cast" : [ "Undefined" ],
7   "genres" : [ "Undefined" ]
8 },
9
10 /* 2 createdAt:25/1/2024 17:12:33*/
11 {
12   "_id" : ObjectId("65b2887117e0db2ad453823a"),
13   "title" : "After Dark in Central Park",
14   "year" : 1900,
15   "cast" : [ "Undefined" ],
16   "genres" : [ "Undefined" ]
17 },
18

```

DEL 11 AL 23

11. Mostrar el año más reciente / actual que tenemos sobre todas las películas. Se muestra una salida de ejemplo, el resultado es erróneo.

```
var query = {}
```

```
var proyecto = {"_id": 0, "year": 1}
```

```
db.movies.find(query, proyecto).sort({ year: -1}).limit(1)
```

```

57 //11)
58 var query = {}
59 var proyecto = {"_id": 0, "year": 1}
60 db.movies.find(query, proyecto).sort({ year: -1}).limit(1)
61
62
63

```

K d x Error x Find (1) x Find (2) x Find (3) x Find (4) x Find (5) x

movies 0.038 s 1 Doc

```

1 {
2   "year" : 2018
3 }

```

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. Se debe hacer con el Framework de Agregación. Se muestra una salida de ejemplo, el resultado es erróneo.

```
var fase1 = { $group : { "_id": "$year", "count" : { "$sum" : 1 } } }

var fase2 = { $sort: { "_id": -1 } }

var fase3 = { $limit: 20 }

var fase4 = { $group: { "_id": null, "total": { "$sum": "$count" } } }

var etapas = [fase1, fase2, fase3, fase4]

db.movies.aggregate(etapas)
```



The screenshot shows a MongoDB Shell window with the following code and output:

```
61 //12)
62
63
64 var fase1 = { $group : { "_id": "$year", "count" : { "$sum" : 1 } } }
65 var fase2 = { $sort: { "_id": -1 } }
66 var fase3 = { $limit: 20 }
67 var fase4 = { $group: { "_id": null, "total": { "$sum": "$count" } } }
68 var etapas = [fase1, fase2, fase3, fase4]
69 db.movies.aggregate(etapas)
70
```

The output shows a single document:

```
1 {
2   "_id" : null,
3   "total" : 4787
4 }
```

13. Contar cuántas películas han salido en la década de los 60 (del 60 al 69 incluidos). Se debe hacer con el Framework de Agregación. Se muestra una salida de ejemplo, el resultado es erróneo.

```
var filtro1 = { "year" : { $gte: 1960, $lte : 1969 } }

var fase1 = { $match : filtro1 }

var fase2 = { $group : { "_id": null, "count" : { "$sum" : 1 } } }

var etapas = [fase1, fase2]

db.movies.aggregate(etapas)
```

```

71 //13)
72 var filtro1 = {"year" : {$gte: 1960, $lte : 1969}}
73 var fase1 = { $match : filtro1}
74 var fase2 = { $group : {"_id": null, "count" : { "$sum" : 1} }}
75 var etapas = [fase1 ,fase2]
76 db.movies.aggregate(etapas)
77
78 //14)

```



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. Se muestra una salida de ejemplo, el resultado es erróneo

var fase1 = { \$group : {"_id": "\$year", "count" : { "\$sum" : 1} }}

var fase2 = { \$group: {"_id": "\$count", "years": { \$push : "\$_id" } } }

var fase3 = { \$sort: {"_id": -1} }

var fase4 = { \$limit: 1 }

var fase5 = { \$unwind: "\$years" }

var etapas = [fase1,fase2,fase3, fase4, fase5]

db.movies.aggregate(etapas)

```

78 //14)
79 var fase1 = { $group : {"_id": "$year", "count" : { "$sum" : 1} }}
80 var fase2 = { $group: {"_id": "$count", "years" : { $push : "$_id" } } }
81 var fase3 = { $sort: {"_id": -1} }
82 var fase4 = { $limit: 1 }
83 var fase5 = { $unwind: "$years" }
84 var etapas = [fase1,fase2,fase3, fase4, fase5]
85 db.movies.aggregate(etapas)
86

```



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. Se muestra una salida de ejemplo, el resultado es erróneo.

var fase1 = { \$group : {"_id": "\$year", "count" : { "\$sum" : 1} }}

```

var fase2 = {$group: {"_id": "$count", "years": {$push : "$_id" } } }

var fase3 = {$sort: {"_id": 1}}

var fase4 = {$limit: 1}

var fase5 = {$unwind: "$years"}

var etapas = [fase1, fase2, fase3, fase4, fase5]

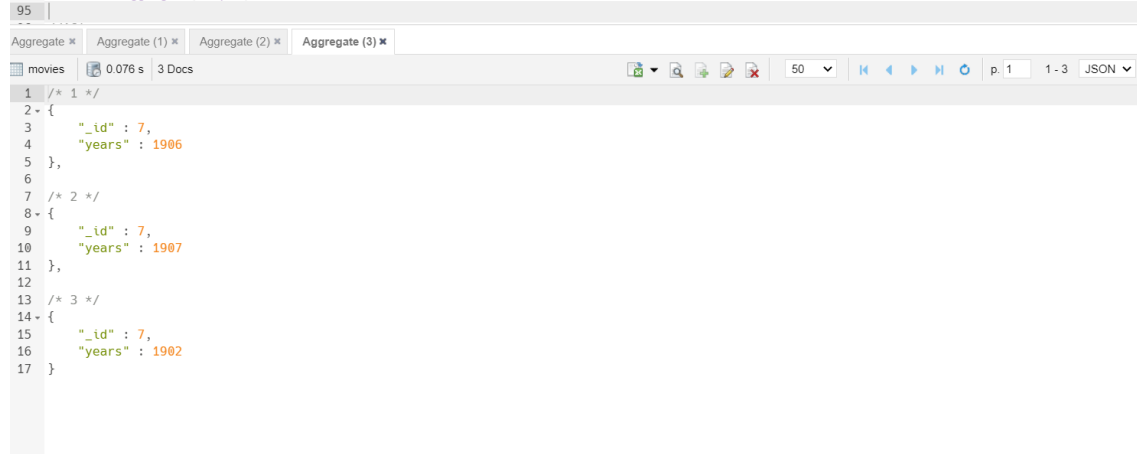
db.movies.aggregate(etapas)

```

```

87 //15)
88 var fase1 = { $group : { "_id": "$year", "count" : { "$sum" : 1 } } }
89 var fase2 = { $group: { "_id": "$count", "years" : { $push : "$_id" } } }
90 var fase3 = { $sort: { "_id": 1 } }
91 var fase4 = { $limit: 1 }
92 var fase5 = { $unwind: "$years" }
93 var etapas = [ fase1, fase2, fase3, fase4, fase5 ]
94 db.movies.aggregate(etapas)
95

```



```

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

```

16. Guardar en nueva colección llamada “actors” realizando la fase \$unwind por actor. Después, contar cuantos documentos existen en la nueva colección. Se muestra una salida de ejemplo, el resultado es erróneo.

```

var fase1 = {$unwind: "$cast"}

var fase2 = {$project: {"_id": 0}}

var fase3 = {$out: "actors"}

var etapas= [fase1, fase2, fase3]

db.movies.aggregate(etapas)

```

db.actors.count()

```
95
96 //16)
97 var fase1 = {$unwind: "$cast"}
98 var fase2 = {$project: {"_id":0}}
99 var fase3 = {$out: "actors"}
100 var etapas= [fase1,fase2,fase3]
101 db.movies.aggregate(etapas)
102
103 db.actors.count( )
104
105
...
e (8) x Console x Result (1) x Find (1) x Console (1) x Result (2) x
0.022 s
1 83224
```

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. Importante! Se necesita previamente filtrar para descartar aquellos actores llamados "Undefined". Aclarar que no se eliminan de la colección, sólo que filtramos para que no aparezcan. Se muestra una salida de ejemplo, el resultado es erróneo.

```
var fase1 = {$match : {"cast": {$ne : "Undefined"}}}

var fase2 = { $group : {"_id": "$cast", "count" : { "$sum" : 1 } } }

var fase3 = {$group: {"_id": "$count", "actors" : { $push : "$_id" } } }

var fase4 = {$sort: {"_id": -1}}

var fase5 = {$limit: 5}

var etapas= [fase1,fase2,fase3,fase4,fase5]

db.actors.aggregate(etapas)
```

```
//17)
var fase1 = {$match :{"cast": {$ne : "Undefined"}}}
var fase2 = { $group : { "_id": "$cast", "count" : { "$sum" : 1 } }}
var fase3 = {$group: { "_id": "$count", "actors" : { $push : "$_id" } } }
var fase4 = {$sort: { "_id": -1 }}
var fase5 = { $limit: 5 }
var etapas= [fase1, fase2, fase3, fase4, fase5]
db.actors.aggregate(etapas)
```

```
1  /* 1 */
2  {
3      "_id" : 190,
4      "actors" : [ "Harold Lloyd" ]
5  },
6
7  /* 2 */
8  {
9      "_id" : 142,
0      "actors" : [ "Hoot Gibson" ]
1  },
2
3  /* 3 */
4  {
5      "_id" : 136,
6      "actors" : [ "John Wayne" ]
7  },
8
9  /* 4 */
0  {
1      "_id" : 116,
2      "actors" : [ "Charles Starrett" ]
3  },
4
5  /* 5 */
6  {
7      "_id" : 103,
8      "actors" : [ "Bebe Daniels" ]
9  }
```

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. Se muestra una salida de ejemplo, el resultado es erróneo.

```
var fase1= { $group: { "_id": { "title": "$title", "year": "$year"}, "cuenta": { "$sum" : 1 }
}}}
```

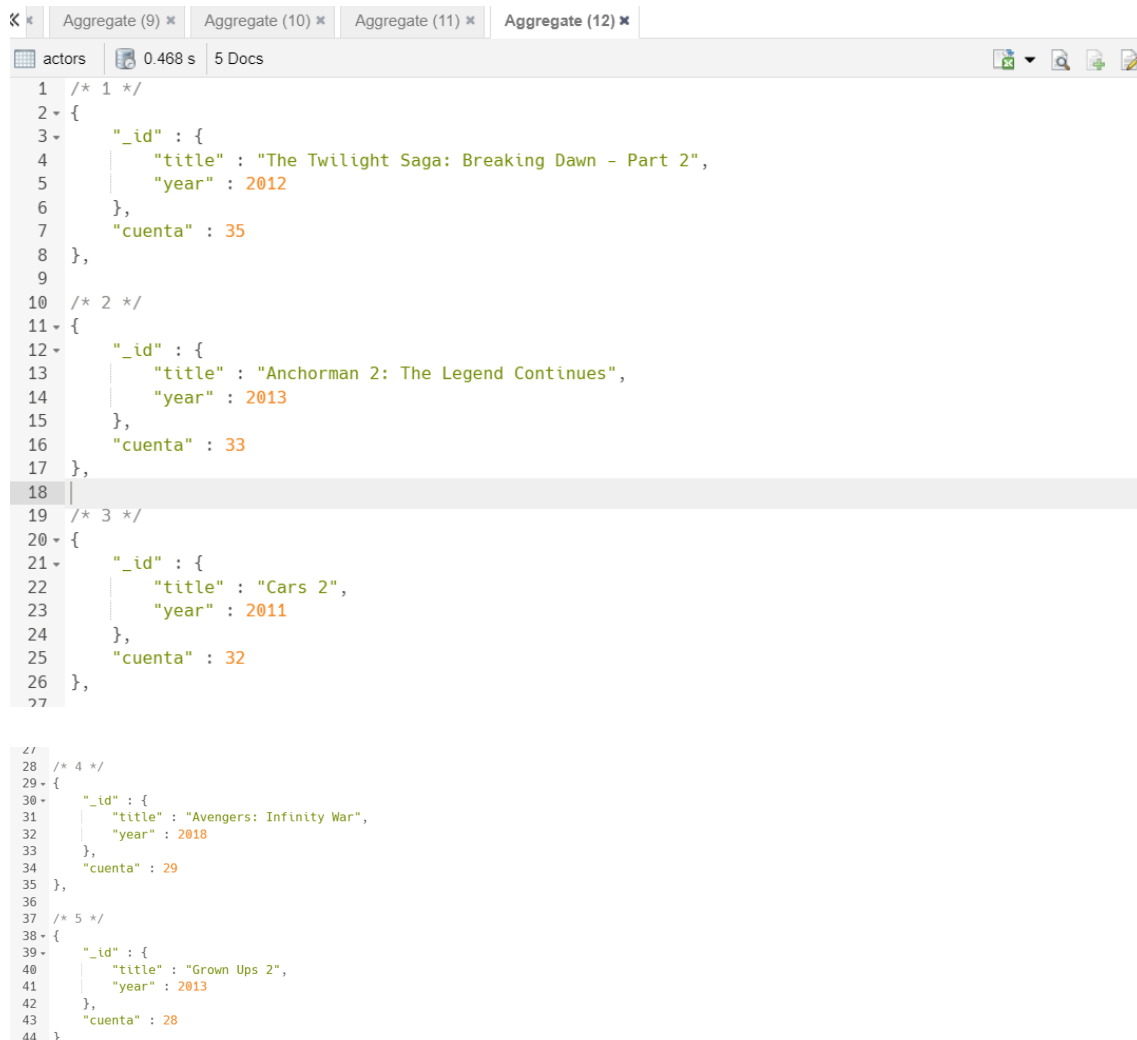
```
var fase2 = {$sort: {"cuenta": -1}}
```

```
var fase3 = {$limit:5}
```

```
var etapas= [fase1,fase2,fase3]
```

```
db.actors.aggregate(etapas)
```

```
115
114 //18)
115 var fase1= { $group: { "_id": { "title": "$title", "year": "$year"}, "cuenta": { "$sum" :1 } } }
116 var fase2 = {$sort: {"cuenta": -1}}
117 var fase3 = {$limit:5}
118 var etapas= [fase1,fase2,fase3]
119 db.actors.aggregate(etapas)
120
```



```
1 /* 1 */
2 {
3   "_id" : {
4     "title" : "The Twilight Saga: Breaking Dawn - Part 2",
5     "year" : 2012
6   },
7   "cuenta" : 35
8 },
9
10 /* 2 */
11 {
12   "_id" : {
13     "title" : "Anchorman 2: The Legend Continues",
14     "year" : 2013
15   },
16   "cuenta" : 33
17 },
18
19 /* 3 */
20 {
21   "_id" : {
22     "title" : "Cars 2",
23     "year" : 2011
24   },
25   "cuenta" : 32
26 },
27
28
29 /* 4 */
30 {
31   "_id" : {
32     "title" : "Avengers: Infinity War",
33     "year" : 2018
34   },
35   "cuenta" : 29
36 },
37
38 /* 5 */
39 {
40   "_id" : {
41     "title" : "Grown Ups 2",
42     "year" : 2013
43   },
44   "cuenta" : 28
45 }
```

19. Sobre actors (nueva colección), 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. Importante! Se necesita previamente filtrar para descartar aquellos actores llamados "Undefined". Aclarar que no se eliminan de la colección, sólo que

filtramos para que no aparezcan. Se muestra una salida de ejemplo, el resultado es erróneo

```
var fase1 = {$match : {"cast": {$ne : "Undefined"}}}
```

```
var fase2 = { $group : { "_id": "$cast", "comienza": {$min: "$year"}, "termina":  
{$max: "$year"} } }
```

```
var fase3 = {$project: { "_id": 1, "comienza": "$comienza", "termina": "$termina",  
"anos": { $subtract: [ "$termina", "$comienza" ] } } }
```

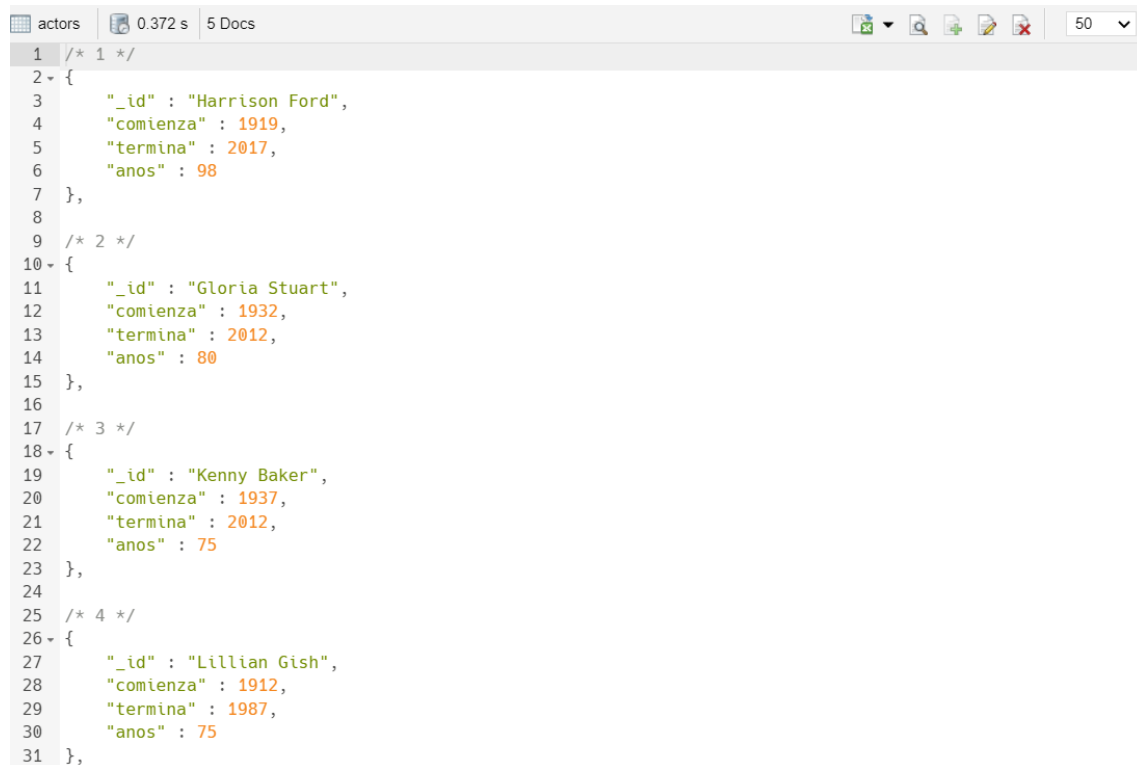
```
var fase4 = {$sort: {"anos": -1}}
```

```
var fase5 = {$limit: 5}
```

```
var etapas= [fase1,fase2,fase3,fase4,fase5]
```

```
db.actors.aggregate(etapas)
```

```
//19)  
var fase1 = {$match : {"cast": {$ne : "Undefined"}}}  
var fase2 = { $group : { "_id": "$cast", "comienza": {$min: "$year"}, "termina": {$max: "$year"} } }  
var fase3 = {$project: { "_id": 1, "comienza": "$comienza", "termina": "$termina", "anos": { $subtract: [ "$termina", "$comienza" ] } } }  
var fase4 = {$sort: {"anos": -1}}  
var fase5 = {$limit: 5}  
var etapas= [fase1,fase2,fase3,fase4,fase5]  
db.actors.aggregate(etapas)
```



```
actors 0.372 s 5 Docs  
1 /* 1 */  
2 {  
3   "_id" : "Harrison Ford",  
4   "comienza" : 1919,  
5   "termina" : 2017,  
6   "anos" : 98  
7 },  
8  
9 /* 2 */  
10 {  
11   "_id" : "Gloria Stuart",  
12   "comienza" : 1932,  
13   "termina" : 2012,  
14   "anos" : 80  
15 },  
16  
17 /* 3 */  
18 {  
19   "_id" : "Kenny Baker",  
20   "comienza" : 1937,  
21   "termina" : 2012,  
22   "anos" : 75  
23 },  
24  
25 /* 4 */  
26 {  
27   "_id" : "Lillian Gish",  
28   "comienza" : 1912,  
29   "termina" : 1987,  
30   "anos" : 75  
31 },
```



```

/* 5 */
{
  "_id" : "Angela Lansbury",
  "comienza" : 1944,
  "termina" : 2018,
  "anos" : 74
}

```

20. Sobre actors (nueva colección), Guardar en nueva colección llamada “genres” realizando la fase \$unwind por genres. Después, contar cuantos documentos existen en la nueva colección. Se muestra una salida de ejemplo, el resultado es erróneo.

```
var fase1 = {$unwind: "$genres"}
```

```
var fase2 = {$project: {"_id":0}}
```

```
var fase3 = {$out: "genres"}
```

```
var etapas= [fase1,fase2,fase3]
```

```
db.actors.aggregate(etapas)
```

```
db.genres.count()
```

```

131 //20)
132 var fase1 = {$unwind: "$genres"}
133 var fase2 = {$project: {"_id":0}}
134 var fase3 = {$out: "genres"}
135 var etapas= [fase1,fase2,fase3]
136 db.actors.aggregate(etapas)
137
138 db.genres.count( )
139
140
141

```

Aggregate x Aggregate (1) x Console x Result x
0.049 s JSON
1 104950

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. Se muestra una salida de ejemplo, el resultado es erróneo.

```
var fase1= { $group: { "_id": { "title": "$title", "year": "$year", "genre": "$genres" } } }

var fase2 = { $group: { "_id": { "year": "$_id.year", "genre": "$_id.genre", "pelis": { "$sum": 1 } } } }

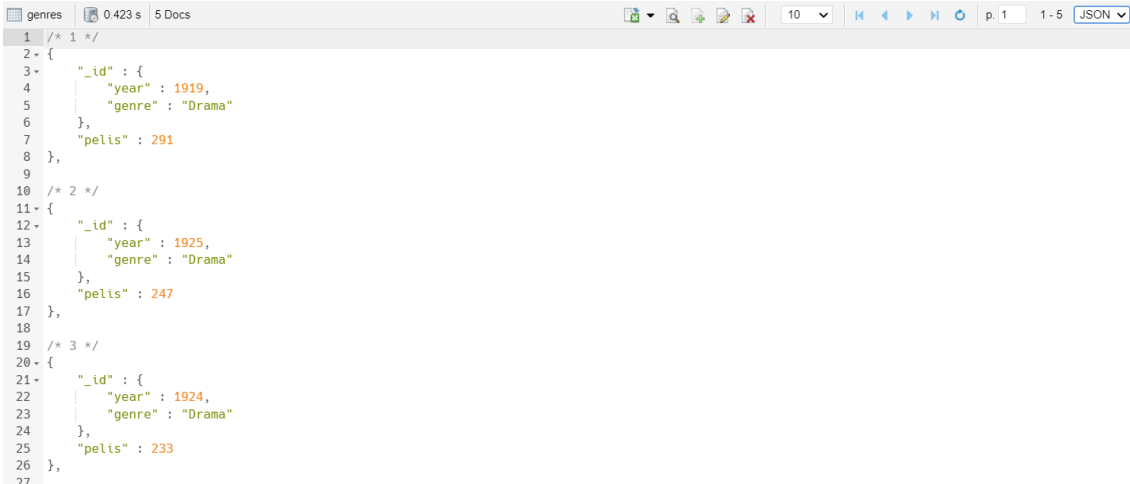
var fase3 = { $sort: { "pelis": -1 } }

var fase4 = { $limit: 5 }

var etapas= [fase1,fase2,fase3,fase4]

db.genres.aggregate(etapas)
```

```
139
140 //21)
141 var fase1= { $group: { "_id": { "title": "$title", "year": "$year", "genre": "$genres" } } }
142 var fase2 = { $group: { "_id": { "year": "$_id.year", "genre": "$_id.genre", "pelis": { "$sum": 1 } } } }
143 var fase3 = { $sort: { "pelis": -1 } }
144 var fase4 = { $limit: 5 }
145 var etapas= [fase1,fase2,fase3,fase4]
146 db.genres.aggregate(etapas)
147
```



```
genres 0.423 s 5 Docs
1 /* 1 */
2 {
3   "_id" : {
4     "year" : 1919,
5     "genre" : "Drama"
6   },
7   "pelis" : 291
8 },
9
10 /* 2 */
11 {
12   "_id" : {
13     "year" : 1925,
14     "genre" : "Drama"
15   },
16   "pelis" : 247
17 },
18
19 /* 3 */
20 {
21   "_id" : {
22     "year" : 1924,
23     "genre" : "Drama"
24   },
25   "pelis" : 233
26 },
27
```

```

7
8 /* 4 */
9 {
10   "_id" : {
11     "year" : 1919,
12     "genre" : "Comedy"
13   },
14   "pelis" : 226
15 },
16
17 /* 5 */
18 {
19   "_id" : {
20     "year" : 1922,
21     "genre" : "Drama"
22   },
23   "pelis" : 209
24 }

```

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. Importante! Se necesita previamente filtrar para descartar aquellos actores llamados "Undefined". Aclarar que no se eliminan de la colección, sólo que filtramos para que no aparezcan. Se muestra una salida de ejemplo, el resultado es erróneo.

```
var fase1 = {$match : {"cast": {$ne : "Undefined"}}}
```

```
var fase2 = {$group : {"_id": {"cast": "$cast", "genres": "$genres"}}}
```

```
var fase3 = {$group: {"_id": "$_id.cast", "generos" : {$push : "$_id.genres" } } }
```

```
var fase4 = {$project: {"_id":1, "numgeneros" : {$size:"$generos"},
"generos": "$generos"}}
```

```
var fase5 = {$sort: {"numgeneros":-1}}
```

```
var fase6 = {$limit:5}
```

```
var etapas= [fase1,fase2,fase3,fase4,fase5,fase6]
```

```
db.genres.aggregate(etapas)
```

```

//22)
var fase1 = {$match : {"cast": {$ne : "Undefined"}}}
var fase2 = {$group : {"_id": {"cast": "$cast", "genres": "$genres"}}}
var fase3 = {$group: {"_id": "$_id.cast", "generos" : {$push : "$_id.genres" } } }
var fase4 = {$project: {"_id":1, "numgeneros" : {$size:"$generos"}, "generos": "$generos"}}
var fase5 = {$sort: {"numgeneros":-1}}
var fase6 = {$limit:5}
var etapas= [fase1,fase2,fase3,fase4,fase5,fase6]
db.genres.aggregate(etapas)

```

```

genres 0.316 s 5 Docs
1 /* 1 */
2 {
3   "_id" : "Dennis Quaid",
4   "numgeneros" : 20,
5   "generos" : [
6     "Horror",
7     "Biography",
8     "Thriller",
9     "Crime",
10    "Romance",
11    "Satire",
12    "Sports",
13    "Western",
14    "Adventure",
15    "Animated",
16    "Disaster",
17    "Science Fiction",
18    "Action",
19    "Dance",
20    "Musical",
21    "Drama",
22    "Family",
23    "Fantasy",
24    "Suspense",
25    "Comedy"
26  ],
27 },
28

```

```

o
9 /* 2 */
10 {
11   "_id" : "James Mason",
12   "numgeneros" : 19,
13   "generos" : [
14     "Undefined",
15     "Musical",
16     "Comedy",
17     "Biography",
18     "Western",
19     "Romance",
20     "Adventure",
21     "Thriller",
22     "War",
23     "Noir",
24     "Action",
25     "Animated",
26     "Short",
27     "Fantasy",
28     "Mystery",
29     "Suspense",
30     "Crime",
31     "Science Fiction",
32     "Drama"
33   ],
34 },
35

```

```

/* 3 */
{
  "_id" : "Michael Caine",
  "numgeneros" : 19,
  "generos" : [
    "Crime",
    "Suspense",
    "Science Fiction",
    "Disaster",
    "Comedy",
    "Romance",
    "War",
    "Thriller",
    "Undefined",
    "Drama",
    "Adventure",
    "Animated",
    "Family",
    "Mystery",
    "Action",
    "Biography",
    "Superhero",
    "Spy",
    "Horror"
  ],
},

```

```

33  /* 4 */
34  {
35      "_id" : "Gene Hackman",
36      "numgeneros" : 18,
37      "generos" : [
38          "Crime",
39          "Spy",
40          "Western",
41          "Thriller",
42          "Mystery",
43          "War",
44          "Disaster",
45          "Noir",
46          "Adventure",
47          "Animated",
48          "Action",
49          "Superhero",
50          "Sports",
51          "Biography",
52          "Drama",
53          "Comedy",
54          "Science Fiction",
55          "Suspense"
56      ]
57  },
58

```

```

/* 5 */
{
  "_id" : "Helen Mirren",
  "numgeneros" : 18,
  "generos" : [
    "Thriller",
    "Horror",
    "Mystery",
    "Spy",
    "Biography",
    "Erotic",
    "Drama",
    "Comedy",
    "Family",
    "Fantasy",
    "Science Fiction",
    "Historical",
    "Animated",
    "Action",
    "Political",
    "Crime",
    "Adventure",
    "Romance"
  ]
}

```

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. Se muestra una salida de ejemplo, el resultado es erróneo.

```
var fase1 = {$group : {"_id": {"title": "$title", "year": "$year", "genres": "$genres"}}}
```

```
var fase2 = {$group: {"_id": {"title": "$_id.title", "year": "$_id.year"}, "generos" : {$push : "$_id.genres" } } }
```

```
var fase3 = {$project: {"_id": 1, "numgeneros" : {$size: "$generos"}, "generos": "$generos"}}
```

```
var fase4 = {$sort: {"numgeneros":-1}}
```

```
var fase5 = {$limit:5}
```

```
var etapas= [fase1,fase2,fase3,fase4,fase5]
```

```
db.genres.aggregate(etapas)
```

```
//23)
var fase1 = {$group : { "_id": { "title": "$title", "year": "$year", "genres": "$genres" }}}
var fase2 = {$group: { "_id": { "title": "$_id.title", "year": "$_id.year", "genres" : {$push : "$_id.genres" } } }
var fase3 = {$project: { "_id": 1, "numgeneros" : {$size: "$genres"}, "genres": "$genres" }}
var fase4 = {$sort: {"numgeneros":-1}}
var fase5 = {$limit:5}
var etapas= [fase1,fase2,fase3,fase4,fase5]
db.genres.aggregate(etapas)
```

genres 0.437 s 5 Docs 50 p. 1 1 - 5 JSON

```
1  /* 1 */
2  {
3    "_id" : {
4      "title" : "American Made",
5      "year" : 2017
6    },
7    "numgeneros" : 7,
8    "genres" : [
9      "Historical",
10     "Action",
11     "Drama",
12     "Crime",
13     "Thriller",
14     "Biography",
15     "Comedy"
16   ]
17 },
```

```
9  /* 2 */
10 {
11   "_id" : {
12     "title" : "My Little Pony: The Movie",
13     "year" : 2017
14   },
15   "numgeneros" : 6,
16   "genres" : [
17     "Musical",
18     "Family",
19     "Adventure",
20     "Comedy",
21     "Animated",
22     "Fantasy"
23   ]
24 },
```

```
36  /* 3 */
37  {
38    "_id" : {
39      "title" : "Dunkirk",
40      "year" : 2017
41    },
42    "numgeneros" : 6,
43    "generos" : [
44      "Drama",
45      "Action",
46      "War",
47      "Thriller",
48      "Historical",
49      "Adventure"
50    ]
51  },
```

```
/* 4 */
{
  "_id" : {
    "title" : "The Dark Tower",
    "year" : 2017
  },
  "numgeneros" : 6,
  "generos" : [
    "Science Fiction",
    "Action",
    "Fantasy",
    "Adventure",
    "Western",
    "Horror"
  ]
},
```

```
/* 5 */
{
  "_id" : {
    "title" : "Wonder Woman",
    "year" : 2017
  },
  "numgeneros" : 6,
  "generos" : [
    "Drama",
    "Superhero",
    "Action",
    "Adventure",
    "War",
    "Fantasy"
  ]
}
```

QUERY LIBRE DEL 24 AL 26

24. Sobre la colección de actores, encuentra el tercer actor con más cantidad de películas diferentes de género de drama, se debe mostrar el número de películas que ha interpretado y cuáles son.

```
var fase1 = {$match : {"cast": {$ne : "Undefined"}, "genres": "Drama"}}
var fase2 = {$group : {"_id": {"cast": "$cast", "title": "$title"}}}
var fase3 = {$group: {"_id": "$_id.cast", "pelis" : {$push : "$_id.title" } } }
var fase4 = {$project: {"_id": 1, "numpelis" : {$size: "$pelis"}, "pelis": "$pelis"}}
var fase5 = {$sort: {"numpelis": -1}}
var fase6 = {$skip: 2}
var fase7 = {$limit: 1}
var etapas= [fase1,fase2,fase3,fase4,fase5,fase6,fase7]
db.actors.aggregate(etapas)
```

En la fase 1, se filtran las películas donde el campo de actores no sea “undefined” y donde el campo géneros sea “Drama”. En la fase 2, se agrupan las películas resultantes por la combinación de los campos de actores y títulos de película.

En la fase 3, se agrupan las películas resultantes por actores y crea un array que contiene el título de las películas por cada actor. Así la cantidad de películas es diferente y no hay duplicidad.

En la fase 4, se selecciona a los actores y el array con los títulos de películas y se crea una variable que cuenta el número de películas correspondiente por cada actor.

En la fase 5, se ordenan las películas restantes en función del número de películas producida por cada actor de manera descendente.

En la fase 6, se omite los primeros 2 documentos de la secuencia, es decir, se saltarán a los dos primeros actores con mayor número de películas.

En la fase 7, se limita el resultado a solo 1 documento, obteniendo el tercer actor con mayor número de películas en las que ha participado.

```
actors 0.221 s 1 Doc
1 {
2   "_id" : "Mary Astor",
3   "numpelis" : 50,
4   "pelis" : [
5     "The World Changes",
6     "The Man with Two Faces",
7     "I Am a Thief",
8     "Any Number Can Play",
9     "Scarlet Saint",
10    "Playing with Souls",
11    "Woman Against Woman",
12    "The Lash",
13    "Man of Iron",
14    "Other Men's Women",
15    "Claudia and David",
16    "Puritan Passions",
17    "High Steppers",
18    "Inez from Hollywood",
19    "The Pace That Thrills",
20    "New Year's Eve",
21    "Dressed to Kill",
22    "Enticement",
23    "Jennie Gerhardt",
24    "Men of Chance",
25    "Three-Ring Marriage",
26    "The Woman from Hell",
27    "Red Dust",
28    "Act of Violence",
29    "Smart Woman",
30    "Young Ideas",
31    "The Bright Shawl",
32    "Trapped by Television",
33    "Upper World",
34    "Behind Office Doors",
35    "Dinky"
```

25. Sobre genres, mostrar el año con menor cantidad de géneros diferentes, mostrando esos géneros y el número de géneros que contiene. Sin contar aquellas películas sin género definido.

```
var fase1 = {$match : {"genres": {$ne : "Undefined"}}}
```

```
var fase2 = {$group : {"_id": "$year", "generos": { $addToSet: "$genres" }}}}
```

```

var fase3 = { $project: { "_id":1, "numgeneros" : { $size: "$generos"},
"generos": "$generos" }}

var fase4 = { $sort: { "numgeneros":1 }}

var fase5 = { $limit:1 }

var etapas= [fase1,fase2,fase3,fase4,fase5]

db.genres.aggregate(etapas)

```

En la fase 1, se filtran las películas donde el género no sea “Undefined”, eliminando aquellos que no están definidos. En la fase 2, se agrupan los datos por el año y agrega los valores únicos del campo de género. En la fase 3, se selecciona el año, los tipos de género de ese año y se crea la variable número de géneros que cuenta la cantidad de tipos de género por año. En la fase 4, se ordenan los datos resultantes en función de la cantidad de tipos de género por año de manera ascendente. En la fase 5, se limita el resultado a solo 1 año, lo que significa que se obtendrá el año con menor cantidad de géneros únicos.

```

180 //25)
181 var fase1 = { $match : { "genres": { $ne : "Undefined" }}}
182 var fase2 = { $group : { "_id": "$year", "generos": { $addToSet: "$genres" }}}
183 var fase3 = { $project: { "_id":1, "numgeneros" : { $size: "$generos"}, "generos": "$generos" }}
184 var fase4 = { $sort: { "numgeneros":1 }}
185 var fase5 = { $limit:1 }
186 var etapas= [fase1,fase2,fase3,fase4,fase5]
187 db.genres.aggregate(etapas)
188
189

```



```

Aggregate (12) x
genres 0.122 s 1 Doc
1- {
2  " _id" : 1993,
3  "numgeneros" : 2,
4  "generos" : [ "Documentary", "Western" ]
5 }

```

26. Contar cuántas películas han salido en el año 2015. En el resultado final, se debe omitir el campo “_id”.

```

var fase1 = { $match : { "year":2015 }}

var fase2 = { $group : { "_id": "$title", "count" : { "$sum" : 1 } }}

var fase3 = { $group: { "_id": null, "total": { "$sum": "$count" } } }

```

```
var fase4 = {$project: {"total":1,"_id":0}}
```

```
var etapas = [fase1 , fase2, fase3,fase4]
```

```
db.movies.aggregate(etapas)
```

En la fase 1, se filtran que aparezcan solo las películas realizadas el año 2015. En la fase 2, se agrupan las películas por el título y cuenta cuántos documentos hay para cada título.

En la fase 3, se agrupan todos los documentos y se suman todas las películas encontradas en la fase anterior. En la última fase, se selecciona solo el campo que contiene el total de películas encontradas en el año 2015 y elimina el campo “-id”.

```
189
190 //26)
191 var fase1 = {$match :{"year":2015}}
192 var fase2 = { $group : { "_id":"$title", "count" : { "$sum" : 1 } }}
193 var fase3 = { $group: { "_id": null, "total": { "$sum": "$count" } } }
194 var fase4 = {$project: {"total":1,"_id":0}}
195 var etapas = [fase1 , fase2, fase3,fase4]
196 db.movies.aggregate(etapas)
197
198
```



```
1 {
2   "total" : 130
3 }
```