PASO PREVIO

Para empezar, cargaremos los datos en compass.

Primero crearemos una base de datos llamada 'Netflix', y crearemos 5 *colecciones*: users, theaters, sessions, directos y comments.

A continuación, con *import data*, introduciremos los datos de los archivos *json* a las nuevas colecciones creadas.

Nivel 1

Ejercicio 1

a. Muestra los dos primeros comentarios que hay en la base de datos:

```
>_MONGOSH
> use Netflix;
< switched to db Netflix
> db.comments.find().limit(2);
< {
    _id: ObjectId('5a9427648b0beebeb69579cc'),
    name: 'Andrea Le',
    email: 'andrea_le@fakegmail.com',
    movie_id: ObjectId('573a1390f29313caabcd418c'),
    text: 'Rem officiis eaque repellendus amet eos doloribus. Porro dolor voluptatum voluptates neque culpa molestias. Voluptate unde nulla temporibus ullam.',
    date: 2012-03-26T23:20:16.000Z
  }
  {
    _id: ObjectId('5a9427648b0beebeb69579cf'),
    name: 'Greg Powell',
    email: 'greg_powell@fakegmail.com',
    movie_id: ObjectId('573a1390f29313caabcd41b1'),
    text: 'Tenetur dolorum molestiae ea. Eligendi praesentium unde quod porro. Commodi nisi sit placeat rerum vero cupiditate neque. Dolorum nihil vero animi.',
    date: 1987-02-10T00:29:36.000Z
  }
Netflix>
```

Primero usaremos *use Netflix*, para asegurarnos que las consultas se están realizando en la base de datos que corresponde. Nos devuelve *switched to db Netflix*, de este modo sabemos que ya estamos usando la base de datos correspondiente.

Ahora con el comando *db.comments.find()* buscamos resultados en la colección comments, y añadiendo *.limit(2)* nos devuelve solo los dos primeros resultados.

b. Cuantos usuarios hay registrados:

```
> db.users.countDocuments();
< 185
Netflix>
```

Para encontrar el número de usuarios registrados usaremos la función *.countDocuments()*, para contar los usuarios contaremos el número de entradas en la colección users, así que los especificaremos con *db.users*.

c. Cuantos cines hay en el estado de California.

```
> db.theaters.countDocuments({ "location.address.state": "CA" });
< 169
```

Si investigamos la base de datos vemos que en la colección de theaters hay un apartado en location – adress donde sale el estado donde se encuentra el cine, en nuestro caso california viene dado por las iniciales CA.

Si queremos saber el número de cines en el estado de California debemos realizar un *countDocuments()*, pero en esta ocasión vamos a introducir una condición en la búsqueda, y es que el estado sea california; lo vamos a especificar con el filtro *"location.adress.state": "CA"*. Finalmente vamos a especificar que la colección donde se realice el count sea en theaters con *db.thaters*.

d. Quien fue el primer usuario en registrarse:

```
> db.users.find().limit(1);
< {
    _id: ObjectId('59b99db4cfa9a34dcd7885b6'),
    name: 'Ned Stark',
    email: 'sean_bean@gameofthron.es',
    password: '$2b$12$UREFwsRUoyF0CRqGNK0LzO0HM/jLhgUCNNIJ9RJAqMUQ74crlJ1Vu'
  }
```

Vamos a realizar la búsqueda en users, en este caso no vamos a imponer ningún filtro, pero vamos a limitar el resultado devuelto a 1.

Suponemos que este es el primer usuario registrado ya que es la primera entrada en la colección users, pero en dicha colección no existe un campo temporal que indique cual es el primer usario en registrarse. Si existiera un campo llamado registration_date por ejemplo podríamos ordenar la colección a partir de ese campo de forma ascendente y limitar el resultado a 1 usuario, para ello utilizaríamos la función *sort()*, que toma un campo y lo ordena de forma ascendente(1) o descentdente (-1 o 0). El query quedaría de la siguiente forma:

db.users.find().sort({ registration_date: 1}).limit(1)

Este query ordenaria la colección users de forma ascendente (1) y limitaría el resultado a 1 elemento de la colección, de esta forma obtendríamos el primer usuario en registrarse en la platafomra.

e. Cuantas películas de comedia hay en la base de datos:

```
> db.movies.countDocuments({ genres: "Comedy" });
< 7024
```

Para encontrar este dato utilizaremos *.countDocuments()*, dentro de la función especificaremos que el genero se comedia mediante *genres: "Comedy"*.

Ejercicio 2

```
> db.movies.find({
    $and: [
      { year: 1932 },
      {
        $or: [
          { genres: "Drama" },
          { languages: "French" }
        ]
      }
    ]
  });
```

Para encontrar este dato utilizaremos la función *find()* en la colección movies, a la cual le aplicaremos los filtros deseados.

Tenemos 3 filtros, year = 1932 y genero = Drama o language = French. Es decir que tenemos una condición and y una or (X and (Y or Z)).

En mongodb esto se especifica de la siguiente manera; find({ $and: [{x} , {y}] }), como tenemos un or en la segunda variable y, se escribiría { $or: [{y} , {z}] }, cuando juntamos las dos nos queda lo siguiente:

find({ $and: [{x} , { $or: [{y} , {z}] }] })

Donde x = year: 1932, y = genres: Drama, z = languages: French.

De esta forma encontramos las películas producidas en el año 1932, y que son de género Dramático o de habla Francesa.

Ejercicio 3

```
Netflix> db.movies.find({
           $and: [
             { countries: "USA" },
             { year: { $gte: 2012, $lte: 2014 } },
             { "awards.wins": { $gte: 5, $lte: 9 } }
           ]
         });
```

En este caso tenemos 3 condiciones que se han de dar simultáneamente, es decir x and y and z. La particularidad de este query es que el año y los awards han de estar entre unos valores determinados, es decir tenemos un between. Como no existe un operador que sea igual a

between en mongodb tendremos que usar los operadores mayor que y menor que, que en mongodb se expressan como **$gte** (greater or equal to) **$lte** (less than or equal to).

De esta forma year: { $gte: 2012, $lte: 2014} y “award.wins”: {$gte: 5, $lte: 9}

El resto del query no es diferente a lo que hemos visto hasta ahora, ya que solo utilizamos la función *find()* y dentro de esta especificamos las condiciones con $and de la siguiente forma:

find({ $and: [{x} , {y} , {z}] })