vedrunapro
Vall Terrassa

Eric Salazar Moreira

Práctica 2: MongoDB CRUD II

Objetivos

Esta actividad te va a permitir dominar las principales funcionalidades de la base de datos MongoDB y sepas utilizarla en diferentes contextos, desde la etapa de carga de datos hasta la construcción de consultas. Para la carga de datos, utilizarás las funciones de Mongo para importar datos. Las consultas las crearás sobre los documentos después de tenerlos disponibles en la base de datos.

Descripción

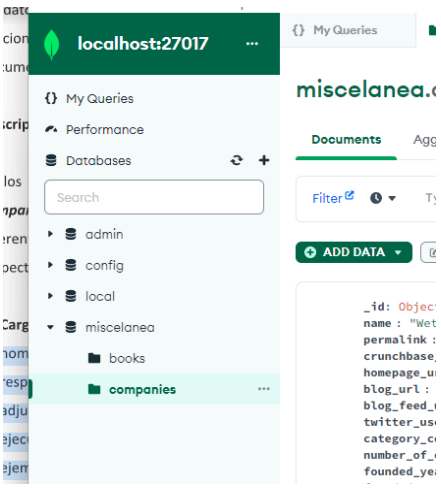
En los ficheros de datos adjuntos (fichero 1: act-2-books.json y fichero 2: act-2-companies.json) tienes un conjunto de documentos con información sobre libros de diferentes temáticas y compañías ubicadas en diferentes partes del mundo, respectivamente. Sobre ambos ficheros se pide lo siguiente:

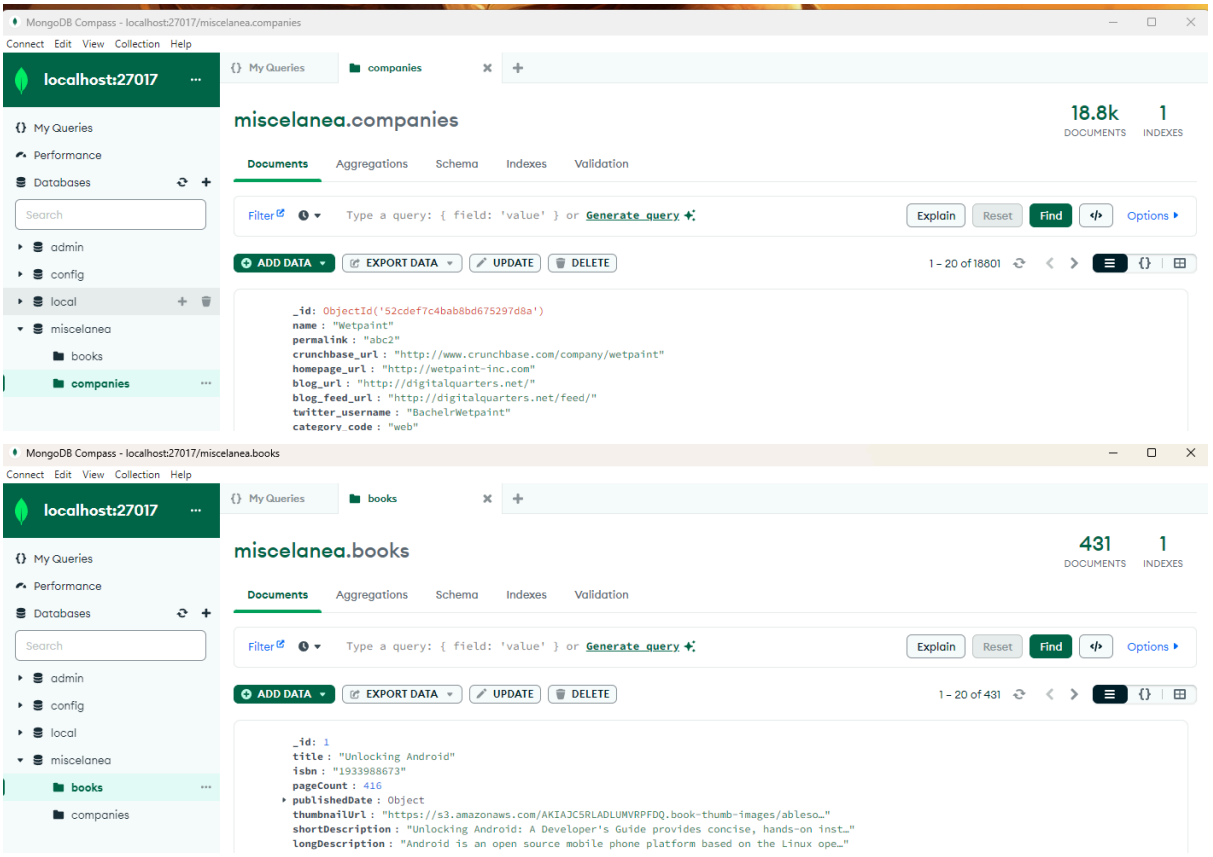
Ejercicios

Carga de datos

Crea una base de datos llamada «miscelanea» (sin acento en el nombre) y en ella crea dos colecciones llamadas books y companies respectivamente. Dichas colecciones contendrán los documentos de los ficheros adjuntos. Los ficheros deben ser importados en MongoDB utilizando el ejecutable «mongoimport.exe» desde la línea de comandos. Aquí tienes un ejemplo de cómo hacerlo: `mongoimport --verbose --db miscelanea – collection books --file c:\...\act-2-books.json`

En mi caso he hecho la carga de datos utilizando MongoDB Compass, puesto que mi versión de MongoDB no contiene la función "mongoimport":





Tanto la base de datos como las colecciones las he creado manualmente y los datos los he importado con la opción de Add Data > Import JSON/CSV

Exploración de colecciones

Visualiza los datos de ambas colecciones y explora los campos que contienen. De cada colección extrae un documento donde se muestren dichos campos y visualízalo de forma «bonita». Copia ambos documentos y añadelos a la memoria de entrega.

Para poder visualizar un documento de cada una de las colecciones a modo de exploración para saber los campos que contienen he utilizado los comandos:

```
db.companies.find().limit(1).pretty()
db.books.find().limit(1).pretty()
```

Estos dos comandos muestran un documento de cada una de las colecciones y la función “pretty” intenta mostrarlos de forma más legible, el problema es que algunos campos obtienen cadenas de carácteres muy largas, y en consola no se pueden visualizar de forma “bonita” para presentarlos. Para poder copiar una captura en la que se vea algo más claro he recurrido a MongoDB Compass, donde simplemente entrando en la colección se pueden ver documentos de ejemplo mucho más claros, aunque la información es exactamente la misma que la que obtenemos con los comandos.

Documento de ejemplo de la colección “books”:

```
_id: 1
title : "Unlocking Android"
isbn : "1933988673"
pageCount : 416
publishedDate : Object
  $date : "2009-04-01T00:00:00.000-0700"
thumbnailUrl : "https://s3.amazonaws.com/AKIAJC5RLADLUMVRPFDQ.book-thumb-images/ableso…"
shortDescription : "Unlocking Android: A Developer's Guide provides concise, hands-on inst…"
longDescription : "Android is an open source mobile phone platform based on the Linux ope…"
status : "PUBLISH"
authors : Array (3)
  0: "W. Frank Ableson"
  1: "Charlie Collins"
  2: "Robi Sen"
categories : Array (2)
  0: "Open Source"
  1: "Mobile"
```


Documento de ejemplo de la colección “companies”:


```
_id: ObjectId('52cdef7c4bab8bd675297d8c')
name : "Zoho"
permalink : "abc4"
crunchbase_url : "http://www.crunchbase.com/company/zoho"
homepage_url : "http://zoho.com"
blog_url : "http://blogs.zoho.com/"
blog_feed_url : "http://blogs.zoho.com/feed"
twitter_username : "zoho"
category_code : "software"
number_of_employees : 1600
founded_year : 2005
founded_month : 9
founded_day : 15
deadpooled_year : 3
tag_list : "zoho, officesuite, spreadsheet, writer, projects, sheet, crm, show, cr…"
alias_list : ""
email_address : "info@zohocorp.com"
phone_number : "1-888-204-3539"
description : "Online Business Apps Suite"
created_at : "Fri May 25 19:30:28 UTC 2007"
updated_at : "Wed Oct 30 00:07:05 UTC 2013"
overview : "<p>Zoho offers a suite of Business, Collaboration &amp;amp; Productivity a…"
image : Object
products : Array (3)
relationships : Array (18)
competitions : Array (26)
providerships : Array (empty)
total_money_raised : "$0"
funding_rounds : Array (empty)
investments : Array (empty)
acquisition : null
acquisitions : Array (empty)
offices : Array (1)
milestones : Array (3)
video_embeds : Array (3)
screenshots : Array (empty)
external_links : Array (6)
partners : Array (empty)
```


Además, desde la consola, extrae la siguiente información:

a. Identifica todas las distintas categorías de la colección books.

Para poder identificar las categorías únicas en el apartado “categories” he utilizado el comando: `db.books.distinct(“categories”)`.
Este comando devuelve todos los valores únicos que hay en el campo “categories” de la colección “books”. En este caso son los siguientes:

```
miscelanea> db.books.distinct("categories")
[
  '',
  '.NET',
  'Algorithmic Art',
  'Business',
  'Client Server',
  'Client-Server',
  'Computer Graph',
  'Computer Graphics',
  'In Action',
  'Internet',
  'Java',
  'Microsoft',
  'Microsoft .NET',
  'Microsoft/.NET',
  'Miscella',
  'Miscellaneous',
  'Mobile',
  'Mobile Technology',
  'Networking',
  'Next Generation Databases',
  'Object-Oriented Programming',
  'Object-Technology Programming',
  'Open Source',
  'P',
  'PHP',
  'Perl',
  'PowerBuilder',
  'Programming',
  'Python',
  'S',
  'SOA',
  'Software Development',
  'Software Engineering',
  'Theory',
  'Web Development',
  'XML',
  'internet',
  'java'
]
miscelanea>
```

b. Identifica los distintos estados (status) de la colección books.

Para poder ver los valores únicos que hay en el campo “status” he utilizado el comando: `db.books.distinct("status").`
Este comando devuelve los posibles estados que pueden tener los registros de la colección “books”. En este caso, solamente “MEAP” y “PUBLISH”

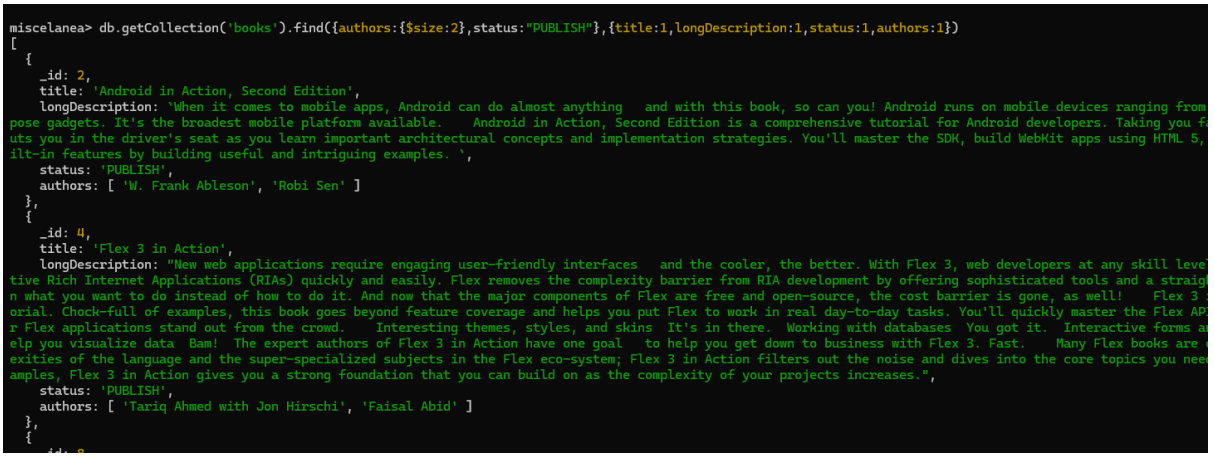
```
]
miscelanea> db.books.distinct("status")
[ 'MEAP', 'PUBLISH' ]
```

c. Describe brevemente qué arroja la siguiente consulta:
`db.getCollection(‘books’).find({longDescription:{$gte:”A”, $lt:”B”}},{title: 1, longDescription:1})`

Esta consulta busca los documentos en la colección “books” donde el valor del campo “longDescription” esté dentro del rango de “A” (incluido) hasta justo antes de “B”, de los documentos que cumplen esta condición muestra el título y la descripción larga.

d. Utiliza la condición de la consulta anterior para recuperar aquellos libros que posean exactamente dos autos y que estén publicados. Muestra sólo los campos: title, longDescripcion, status y authors.

Para poder realizar la consulta he utilizado la misma estructura que en el comando anterior, poniendo la condición de los dos autores y el status publicado en un find y en el resto añadiendo un ":1" para visualizar los campos pedidos. El comando resultante es : `db.getCollection('books').find({authors:{$size:2},status:"PUBLISH"},{title:1,longDescription:1,status:1,authors:1})`

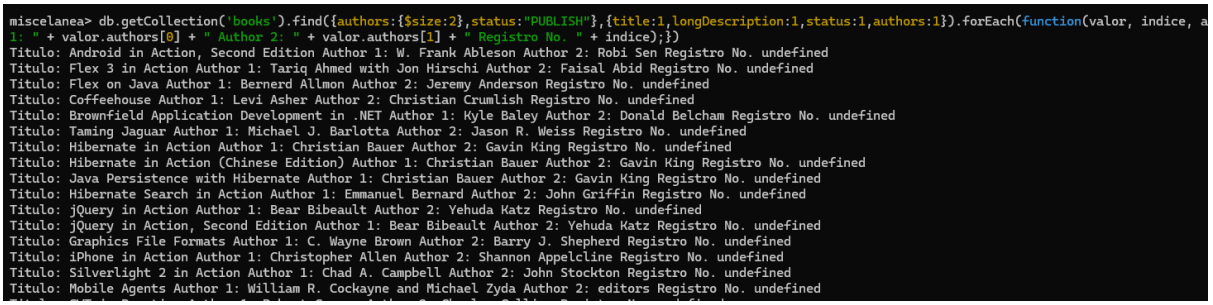


e. Describe brevemente qué ocurre si a la consulta del punto anterior le añades al final la siguiente instrucción: .toArray()

Al añadir la instrucción ".toArray()" al final de la consulta, al ejecutarse devuelve los resultados coincidentes en forma de array de documentos, de esta forma, en caso de tener muchos documentos, por ejemplo, la manipulación es más sencilla.

f. ¿Qué ocurre si ahora también le añades lo siguiente?: `.forEach(function(valor, indice, array){print("Titulo:"+valor.title"Author 1: " +valor.authors[0] + " Author 2: " + valor.authors[1] + "Registro No. " + indice);})` ¿Para qué crees que sería útil esto último?

Ejecutando la consulta anterior añadiendo este bucle al final, obtendremos una impresión de los resultados distinta, puesto que para cada documento que se vaya a mostrar como resultado, se verá una frase como: "Título: Título resultante Author 1: Autor 1 resultante Author 2: Autor 2 resultante Registro no: Id del resultado".
Esto puede ayudar para ver información más clara y rápida de cada resultado sin tener que revisar el array de documentos. Nos ayuda a resumir la información y centrarnos en lo que nos importa saber de los documentos resultantes.

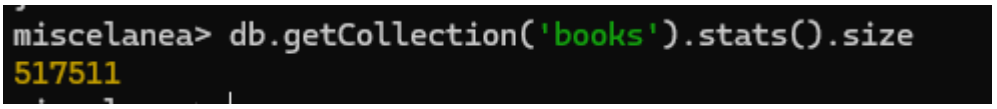


Consulta la colección 1

Sobre la colección “books” realiza las siguientes consultas:

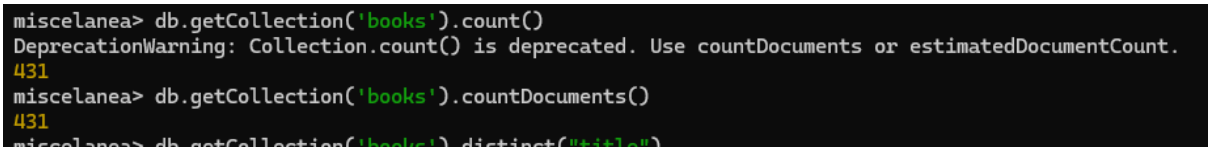
a. ¿Cuál es el tamaño de la colección (en bytes)?

Para poder saber el tamaño de la colección en bytes he utilizado el comando:
`db.getCollection('books').stats().size`

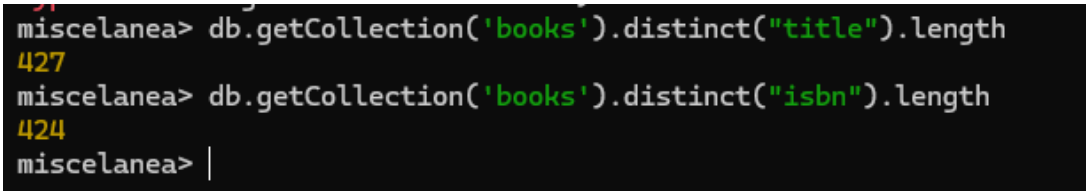


b. ¿Cuántos libros tiene la colección?

Para contar los libros que tiene la colección he utilizado el comando:
`db.getCollection('books').count()`
Puesto que la consola de MongoDB me ha dicho que era un comando obsoleto he sustituido el .count por .countDocuments:



Después he utilizado otros dos comandos (a modo de prueba), contando solamente los que tienen un título único y un ISBN único también, para ello he usado los comandos:
`db.getCollection('books').distinct("title").length`
`db.getCollection('books').distinct("isbn").length`
Y hay 4 registros de libros con mismo nombre y 7 con mismo isbn:



c. ¿Cuántos libros tienen 200 o más páginas?

Para contar los libros con 200 páginas o más he utilizado el comando:
`db.getCollection('books').count({pageCount: {$gte: 200}})`
Esto cuenta (por eso uso count) los libros con 200 páginas o más (gte) en la colección “books”:

```
miscelanea> db.getCollection('books').count({pageCount: {$gte: 200}})
264
miscelanea>
```

d. ¿Cuántos libros tienen entre 300 y 600 páginas? [300,600]

Para contar los libros que tienen entre 300 y 600 páginas incluyendo los extremos que utilizado la misma estructura que en el comando anterior pero añadiendo una condición $lte para cortar en 600 o menos. El comando completo:
`db.getCollection('books').count({pageCount: {$gte: 300, $lte: 600}})`

```
miscelanea> db.getCollection('books').count({pageCount: {$gte: 300, $lte: 600}})
215
miscelanea>
```

e. ¿Cuántos libros tienen 0 páginas y cuántos no?

Para realizar esta consulta podría utilizar dos comandos sencillos, uno para contar los libros con 0 páginas y otro para contar los libros con más de 0, pero he utilizado una agregación para verlo en un mismo resultado:

```
db.getCollection('books').aggregate([
    {
        $group: {
            _id: null,
            countWithZeroPages: { $sum: { $cond: [{ $eq: ["$pageCount", 0]
}, 1, 0] } },
            countWithoutZeroPages: { $sum: { $cond: [{ $ne: ["$pageCount",
0] }, 1, 0] } }
        }
    }
])
```

En la agregación se nos muestran dos nuevos campos, countWithZeroPages y countWithoutZeroPages, los cuales salen de la suma de resultados de las condiciones de contar los libros en los que su "pageCount" equivale ($eq) a 0 y los que no equivalen a 0 ($ne).

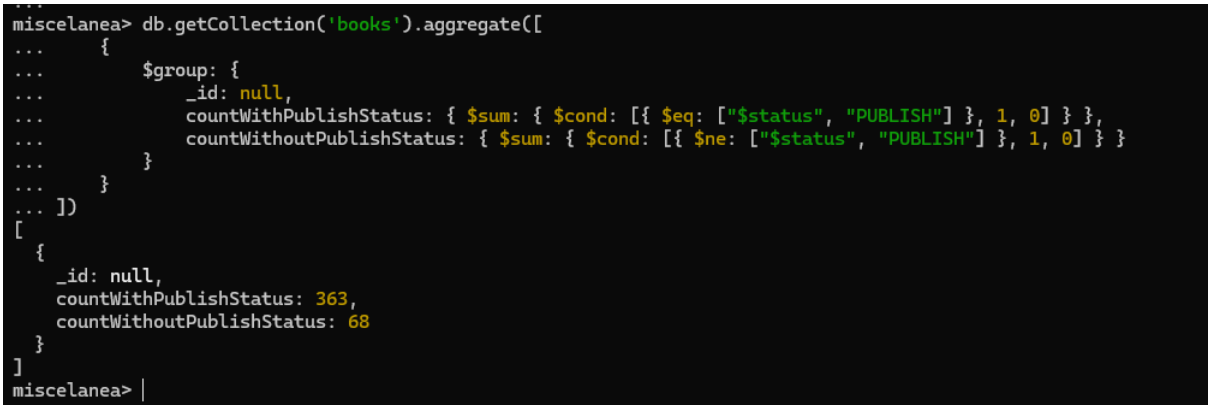
```
miscelanea> db.getCollection('books').aggregate([
...     {
...         $group: {
...             _id: null,
...             countWithZeroPages: { $sum: { $cond: [{ $eq: ["$pageCount", 0] }, 1, 0] } },
...             countWithoutZeroPages: { $sum: { $cond: [{ $ne: ["$pageCount", 0] }, 1, 0] } }
...         }
...     }
... ])
[ { _id: null, countWithZeroPages: 166, countWithoutZeroPages: 265 } ]
miscelanea>
```

f. ¿Cuántos libros han sido publicados y cuántos no?

Para realizar esta consulta también podría haber utilizado dos comandos sencillos, uno para contar los equivalentes a status: PUBLISH y otro para los que no lo son, pero he utilizado una agregación con la misma estructura que la anterior.

```
db.getCollection('books').aggregate([
    {
        $group: {
            _id: null,
            countWithPublishStatus: { $sum: { $cond: [{ $eq: ["$status",
"PUBLISH"] }, 1, 0] } },
            countWithoutPublishStatus: { $sum: { $cond: [{ $ne: ["$status",
"PUBLISH"] }, 1, 0] } }
        }
    }
])
```

Es exactamente la misma agregación pero contando los valores que equivalen a “PUBLISH” en el campo status y los que equivalen a algo distinto.



Consulta la colección 2

Sobre la colección “companies” realiza las siguientes consultas:

a. ¿Cuál es el tamaño de la colección (en bytes)?

Para poder ver el tamaño de la colección en bytes he utilizado el mismo comando que en la colección anterior:
`db.getCollection(‘companies’).stats().size`

b. ¿Cuántas compañías tiene la colección?

Para poder contar cuantas compañías tiene la colección he utilizado el comando:
`db.getCollection('companies').count()`

```
miscelanea> db.getCollection('companies').countDocuments()
18801
miscelanea>
```

También he contado cuántas compañías con nombre único hay con el comando:

```
db.getCollection('companies').distinct("name").length
```

En este caso hay unas cuantas repetidas:

```
miscelanea> db.getCollection('companies').distinct("name").length
17893
miscelanea>
```

c. ¿Cuántas compañías se fundaron en los años 1996,1997,2001 y 2005 respectivamente?

En este caso de consulta he utilizado una agregación, para no tener que ejecutar cuatro comandos distintos.

```
db.getCollection('companies').aggregate([
  {
    $facet: {
      "count_1996": [{ $match: { founded_year: 1996 } }, { $count: "count"
}],
      "count_1997": [{ $match: { founded_year: 1997 } }, { $count: "count"
}],
      "count_2001": [{ $match: { founded_year: 2001 } }, { $count: "count"
}],
      "count_2005": [{ $match: { founded_year: 2005 } }, { $count: "count"
}]
    }
  }
])
```

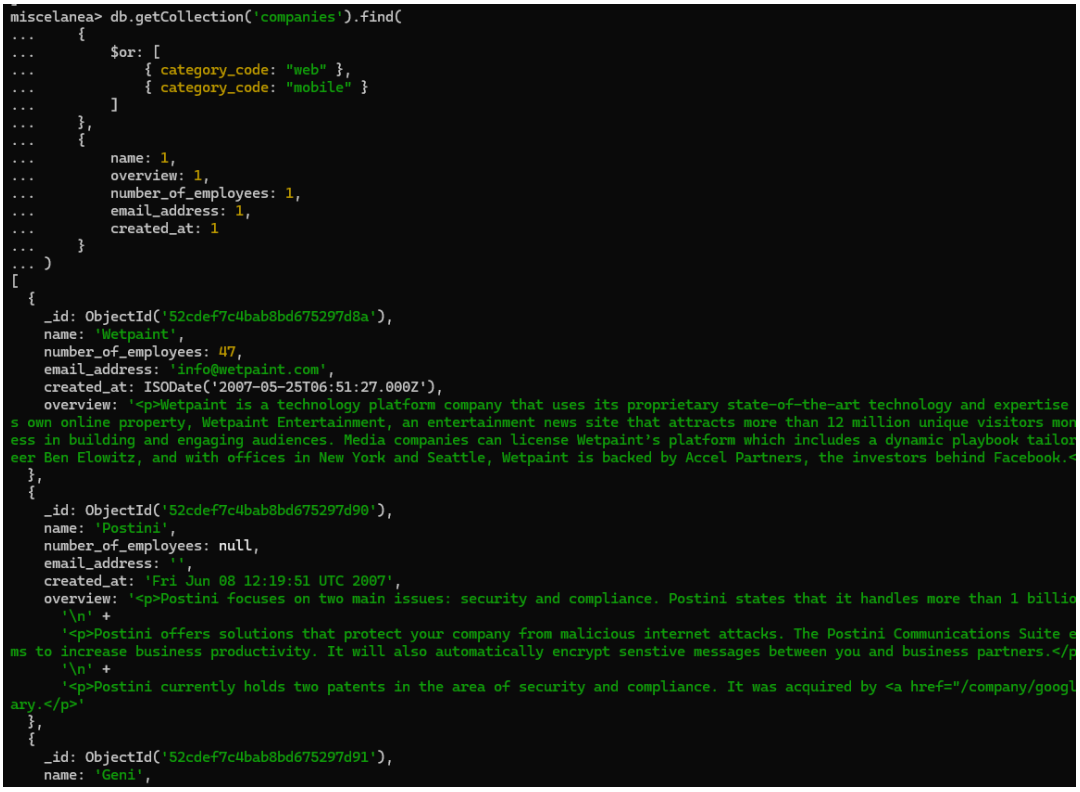
Esta agregación simplemente muestra en cuatro campos distintos el resultado del $count de los documentos que cumplen con el parámetro especificado en el $match, en este caso los años.

```
miscelanea> db.getCollection('companies').aggregate([
...   {
...     $facet: {
...       "count_1996": [{ $match: { founded_year: 1996 } }, { $count: "count" }],
...       "count_1997": [{ $match: { founded_year: 1997 } }, { $count: "count" }],
...       "count_2001": [{ $match: { founded_year: 2001 } }, { $count: "count" }],
...       "count_2005": [{ $match: { founded_year: 2005 } }, { $count: "count" }]
...     }
...   }
... ])
[
  {
    count_1996: [ { count: 216 } ],
    count_1997: [ { count: 200 } ],
    count_2001: [ { count: 464 } ],
    count_2005: [ { count: 961 } ]
  }
]
miscelanea>
```

d. Lista las compañías que se dedican a “web” o “mobile” y recupera: el nombre, descripción, número de empleados, email, año, mes y día de su fundación.

Para poder ver la información descrita dependiendo de si cumplen una especificación u otra, he utilizado un $or, para que pueda escoger entre las dos categorías y después simplemente he especificado con un :1 los campos que quería visualizar.

```
db.getCollection('companies').find(
    {
        $or: [
            { category_code: "web" },
            { category_code: "mobile" }
        ]
    },
    {
        name: 1,
        overview: 1,
        number_of_employees: 1,
        email_address: 1,
        created_at: 1
    }
)
```



e. Lista las compañías que se dedican a videojuegos y muéstralas en orden descendente según el año en que fueron fundadas.

Para esta consulta he utilizado un .find para especificar el category_code en games_video y después he utilizado un .sort con un -1 para que se muestren ordenadas del revés por año de fundación.

```
db.getCollection('companies').find({category_code:"games_video"}).sort({fou
nded_year: -1})
```

f. ¿Cuántas compañías tienen 600 o más empleados?

Para realizar esta consulta he utilizado un .count donde el valor del número de empleados sea mayor o igual ($gte) a 600:

```
db.getCollection('companies').count({number_of_employees: {$gte: 600}})
```

g. Recupera el nombre, la URL, el usuario de Twitter y el número de empleados de las compañías fundadas entre los años 2001 y 2005, ambos incluidos, que cuenten con 500 o más empleados y que se dediquen a los videojuegos o a la música. ¿Alguna compañía se dedica a los videojuegos y a la música a la vez?

Para poder visualizar todos los datos de la consulta he utilizado un find especificando el rango de los años con un $gte y $lte, el número de empleados con un $gte y las categorías con un $or, puesto que valen las dos.

```
db.getCollection('companies').find({
    founded_year: { $gte: 2001, $lte: 2005 },
    number_of_employees: { $gte: 500 },
    $or: [
        { category_code: "games_video" },
        { category_code: "music" }
    ]
}, {
    name: 1,
    homepage_url: 1,
    twitter_username: 1,
    number_of_employees: 1,
    founded_year: 1,
    _id: 0
})
```

```
miscelanea> db.getCollection('companies').find({
...     founded_year: { $gte: 2001, $lte: 2005 },
...     number_of_employees: { $gte: 500 },
...     $or: [
...         { category_code: "games_video" },
...         { category_code: "music" }
...     ]
... }, {
...     name: 1,
...     homepage_url: 1,
...     twitter_username: 1,
...     number_of_employees: 1,
...     founded_year: 1,
...     _id: 0
... })
[
  {
    name: 'GREE',
    homepage_url: 'http://www.gree-corp.com',
    twitter_username: 'gree_corp',
    number_of_employees: 700,
    founded_year: 2004
  },
  {
    name: 'Bigpoint',
    homepage_url: 'http://www.bigpoint.com',
    twitter_username: 'Bigpoint',
    number_of_employees: 500,
    founded_year: 2002
  }
]
```

Para listar las compañías que se dediquen a los videojuegos y a la música a la vez puede utilizarse una consulta find sencilla con un $and:

```
db.getCollection('companies').find({
    $and: [
        { category_code: "games_video" },
        { category_code: "music" }
    ]
})
```

El resultado es que no hay ninguna.

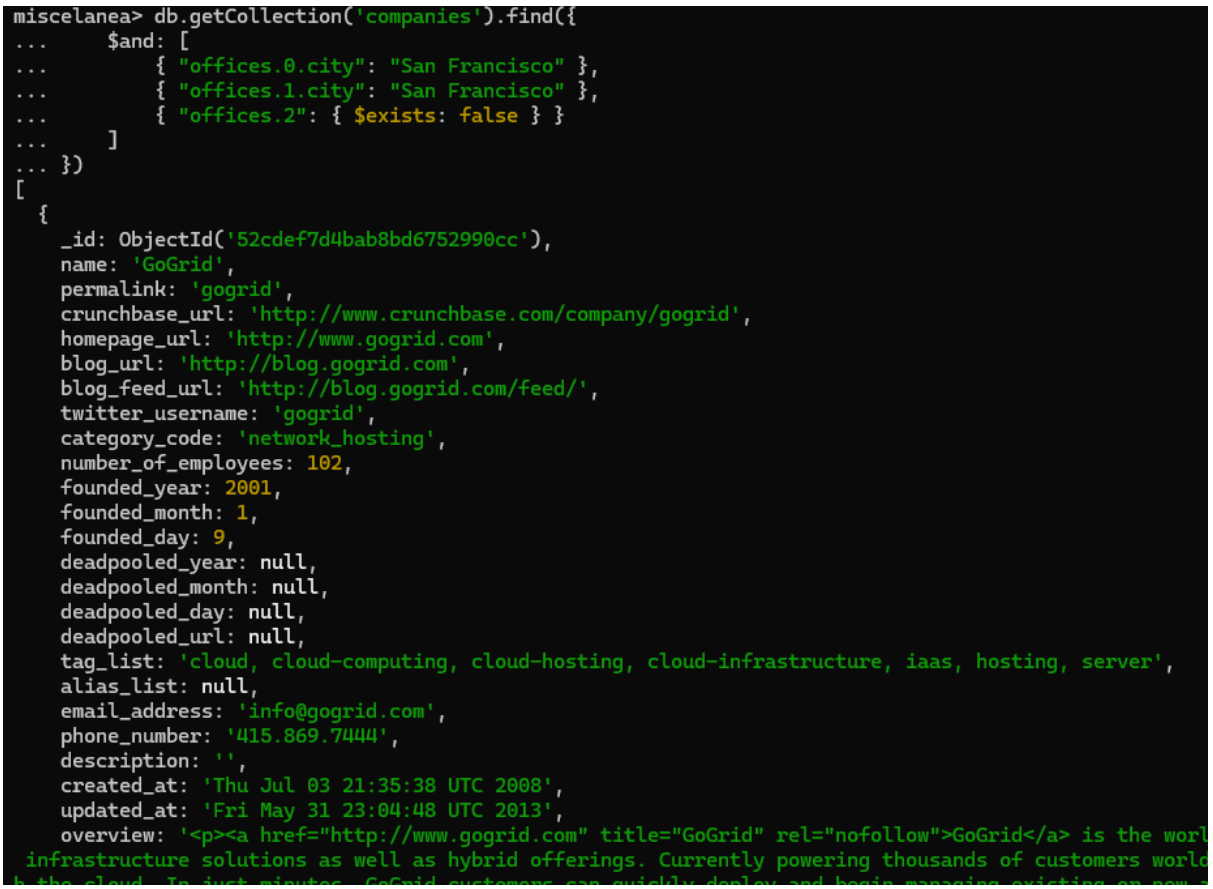
```
miscelanea> db.getCollection('companies').find({
...     $and: [
...         { category_code: "games_video" },
...         { category_code: "music" }
...     ]
... })

miscelanea>
```

h. Lista las empresas que cuentan con única y exclusivamente 2 oficinas en la ciudad de San Francisco.

Para realizar esta consulta he utilizado un $and donde el campo city de los objetos 0 y 1 del campo offices sea “San Francisco” y donde el objeto 2 (tercero) del campo offices no exista, para asegurarme de que no cuenta empresas con más de dos oficinas.

```
db.getCollection('companies').find({
    $and: [
        { "offices.0.city": "San Francisco" },
        { "offices.1.city": "San Francisco" },
        { "offices.2": { $exists: false } }
    ]
})
```



```
miscelanea> db.getCollection('companies').find({
...     $and: [
...         { "offices.0.city": "San Francisco" },
...         { "offices.1.city": "San Francisco" },
...         { "offices.2": { $exists: false } }
...     ]
... })
[
  {
    _id: ObjectId('52cdef7d4bab8bd6752990cc'),
    name: 'GoGrid',
    permalink: 'gogrid',
    crunchbase_url: 'http://www.crunchbase.com/company/gogrid',
    homepage_url: 'http://www.gogrid.com',
    blog_url: 'http://blog.gogrid.com',
    blog_feed_url: 'http://blog.gogrid.com/feed/',
    twitter_username: 'gogrid',
    category_code: 'network_hosting',
    number_of_employees: 102,
    founded_year: 2001,
    founded_month: 1,
    founded_day: 9,
    deadpooled_year: null,
    deadpooled_month: null,
    deadpooled_day: null,
    deadpooled_url: null,
    tag_list: 'cloud, cloud-computing, cloud-hosting, cloud-infrastructure, iaas, hosting, server',
    alias_list: null,
    email_address: 'info@gogrid.com',
    phone_number: '415.869.7444',
    description: '',
    created_at: 'Thu Jul 03 21:35:38 UTC 2008',
    updated_at: 'Fri May 31 23:04:48 UTC 2013',
    overview: '<p><a href="http://www.gogrid.com" title="GoGrid" rel="nofollow">GoGrid</a> is the worl
 infrastructure solutions as well as hybrid offerings. Currently powering thousands of customers worlc
```

i. Lista el nombre, el mes y día de adquisición de las empresas de videojuegos que hayan sido adquiridas en el año 2007 por un precio igual o superior a los 10 millones de dólares y que tengan oficinas en la ciudad de Culver City.

Para realizar esta consulta he utilizado un find, especificando el nombre del campo “city” del objeto offices, el “acquired_year” y el “price_amount” del objeto acquisition y la categoría de la empresa. Y después he mostrado con un :1 los campos que quería visualizar, en este caso el “acquired_month” y el “acquired_day” del objeto acquisition y el nombre de la compañía.

```
db.getCollection('companies').find({
    "offices.city": "Culver City",
    "acquisition.acquired_year": 2007,
    "acquisition.price_amount": { $gte: 10000000 },
    category_code: "games_video"
},
{
    name: 1,
    "acquisition.acquired_month": 1,
    "acquisition.acquired_day": 1,
    _id: 0
})
```

En este caso solamente una empresa cumple con todos los requisitos.