UNIR - Universidad Internacional de la Rioja

Ciudad de México

**ACTIVIDAD 1:**

**Uso de MongoDB**

Alumno: Cuenca Roa, Leonard Jose

Grupo: 1001

**ÍNDICE**

Carga de Datos 4

Evidencia carga de datos 4

Explora las colecciones 5

Identifica todas las distintas categorías (categories) de la colección books 5

Identifica los distintos estados (status) de la colección books 6

Describe brevemente qué arroja la siguiente consulta 6

Utiliza la condición de la consulta anterior para recuperar aquellos libros que posean exactamente 2 autores y que estén publicados 7

Describe brevemente qué ocurre si a la consulta del punto anterior le añades al final la siguiente instrucción 7

Qué ocurre si también le añades lo siguiente 8

Consulta la colección 1 9

Sobre la colección books realiza las siguientes consultas 9

¿Cuál es el tamaño de la colección (en bytes)? 9

¿Cuántos libros tiene la colección? 10

¿Cuántos libros tienen 200 o más páginas? 10

¿Cuántos libros tienen entre 300 y 600 páginas? [300, 600] 10

¿Cuántos libros tienen 0 páginas y cuántos no? 10

¿Cuántos libros han sido publicados y cuántos no? 12

Consulta la colección 2 13

¿Cuál es el tamaño de la colección (en bytes)? 13

¿Cuántas compañías tiene la colección? 13

¿Cuántas compañías se fundaron en los años 1996, 1997, 2001 y 2005 respectivamente? 14

Lista las compañías que se dedican a «web» o «mobile» y recupera: nombre, descripción, número de empleados, e-mail, año, mes y día de su fundación. 15

Lista las compañías que se dedican a videojuegos y muéstralas en orden descendente según el año en que fueron fundadas. 16

Recupera el nombre, la URL, el usuario de Twitter y el número de empleados de las compañías fundadas entre los años 2001 y 2005 incluidos, que cuenten con 500 o más empleados y que se dediquen a los videojuegos o a la música. 17

Lista las empresas que cuentan con única y exclusivamente 2 oficinas en la ciudad de San Francisco. 18

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. 20

Carga de Datos

De la carga de datos, deseo resaltar lo que pude aprender y comprender qué el objetivo es organizar y almacenar datos sobre libros y compañías en una base de datos NoSQL, utilizando MongoDB y su herramienta de importación.

Resaltando la satisfacción y la ejecución de las siguientes actividades:

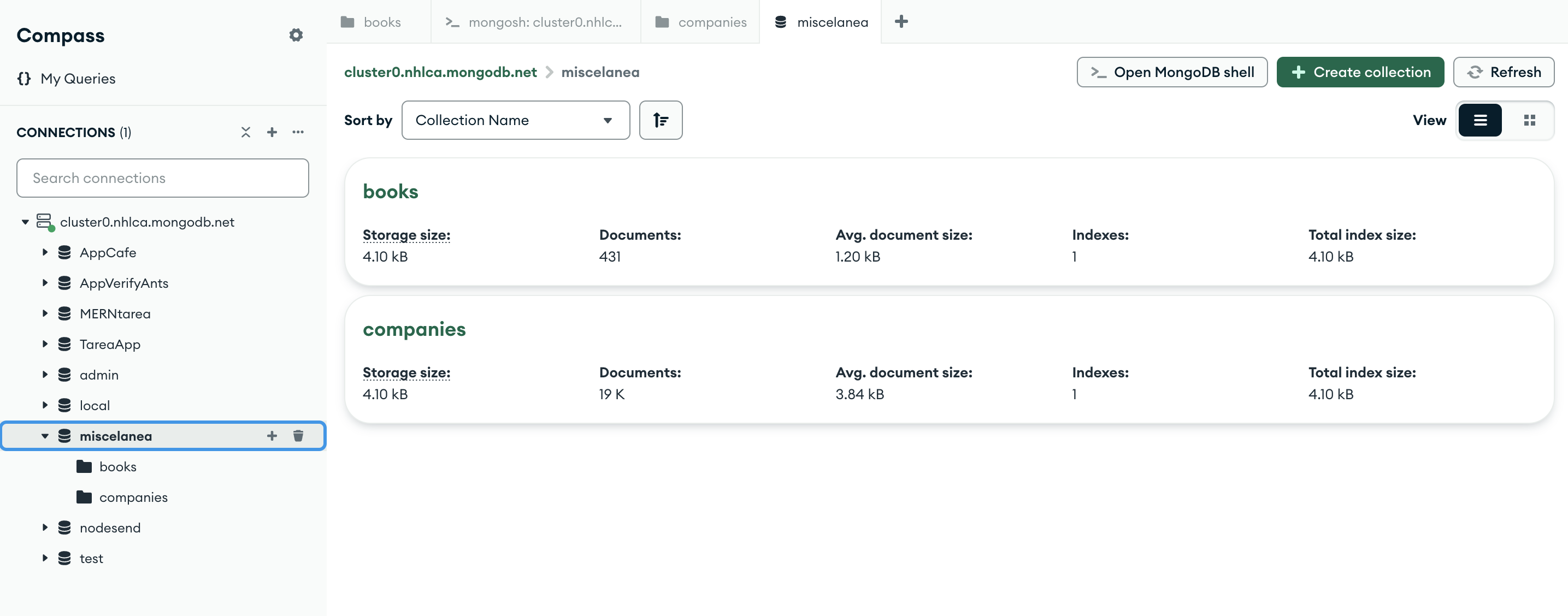
**Creación de una base de datos**: Se debe crear una nueva base de datos con el nombre "miscelanea".

**Creación de colecciones**: Dentro de la base de datos, se creó las dos colecciones: "books" y "companies".

**Importación de datos**: Los datos se cargó desde archivos JSON a las colecciones correspondientes utilizando la herramienta.

**Herramienta a utilizar**: Se adoptó una estrategia híbrida que integra la línea de comandos de MongoDB con las capacidades visuales de MongoDB Compass. Esta metodología fue seleccionada para llevar a cabo la importación y validación de datos, aprovechando las herramientas de última generación ofrecidas por Atlas. Al combinar la precisión de la línea de comandos con la facilidad de uso de la interfaz gráfica, se logró un proceso más eficiente y controlado.

Evidencia carga de datos



Evidencia de base de datos y sus colecciones creadas.

Explora las colecciones

Identifica todas las distintas categorías (categories) de la colección books

Para resolver este ejercicio, se implementó una serie de funciones que ofrece MongoDB. En primer lugar, identifico la colección con la que quiero trabajar utilizando **getCollection**(), especificando la colección **books**. Luego, empleo el método **aggregate**(), que permite realizar operaciones complejas sobre una colección. Luego, utilizo el operador $**unwind** para descomponer el campo '**categories**' de cada documento en registros individuales. Una vez hecho esto, procedo a agrupar los resultados utilizando el operador $**group** para agrupar por el campo deseado utilizando la propiedad $**addToSet** ya que, toma el valor del campo **categories** de cada documento y lo agrega a un nuevo conjunto **(set),** con el propósito de que solo agrega un valor al conjunto si este aún no existe en caso contrario si un mismo valor de categoría aparece en múltiples documentos, solo se incluirá una vez en el conjunto final y para finalizar, utilizo el operador $**project** para seleccionar los campos que quiero incluir en el resultado final, ocultando aquellos que no son necesarios.

| Código | Evidencia |
| --- | --- |
| db.getCollection('books').aggregate([  { $unwind: '$categories' },  { $group: {\_id: null, categories: { $addToSet: '$categories' }}},  { $project: { \_id: 0, categories: 1 } }  ]); |  |
|
|
|

Identifica los distintos estados (status) de la colección books

En este ejercicio, identificamos tres posibles soluciones para obtener registros únicos a partir de una colección. Utilizando el operador **distinct**: Este operador nos permite listar todos los campos de una colección, eliminando los duplicados y mostrando únicamente los valores únicos. Consultando directamente la colección de manera similar al primer caso, podemos obtener los mismos resultados accediendo directamente a la colección utilizando **db.getCollection()** otra opción es agrupando los resultados con **$group,** para agrupar los datos por un campo específico para este caso el **estatus** y obtener valores únicos, podemos emplear el operador $**group** dentro de una etapa de **agregación.**

| Código | Evidencia |
| --- | --- |
| //Opción Uno  db.books.distinct("status")  //Opción Dos  db.getCollection('books').distinct("status")  //Opción Tres  db.getCollection('books').aggregate(  [{ $group: { \_id: '$status' } }]  ); |  |
|
|
|

Describe brevemente qué arroja la siguiente consulta

Esta consulta busca en la colección **books** todos los libros cuyo campo **longDescription** comience por una letra que esté entre la letra "A" y la letra "B" sin incluir la letra "B". De todos los **books** que cumplan esta condición, solo se mostrarán los campos **title** y **longDescription.**

| Código | Evidencia |
| --- | --- |
| db.getCollection('books').find({longDescription: {$gte: "A", $lt: "B"}}, {title: 1, longDescription: 1}) | Captura de pantalla 2024-11-15 a la(s) 0.33.30.png |
|
|
|

Utiliza la condición de la consulta anterior para recuperar aquellos libros que posean exactamente 2 autores y que estén publicados

Muestra solo los campos: title, longDescription, status y authors.

| Código | Evidencia |
| --- | --- |
| db.getCollection('books').find({  status: "PUBLISH",  authors: { $size: 2 }  }, {  title: 1,  longDescription: 1,  status: 1,  authors: 1  }) |  |
|
|
|

Describe brevemente qué ocurre si a la consulta del punto anterior le añades al final la siguiente instrucción

.toArray()

Al añadirle la propiedad este genera un arreglo de objetos con todos los resultados, en cambio la consulta anterior genera un cursor para iterar sobre los resultados, esta pequeña diferencia depende de la necesidades que se desees trabajar, por ejemplo si necesitas procesar los resultados de forma incremental ó cuando tienes un gran número de resultados y quieres evitar cargarlos todos a la memoria a la vez ó cuando quieres iterar sobre los resultados varias veces **usas el cursor** pero cuando necesitas trabajar con todos los resultados a la vez o cuando quieres pasar los resultados a otra función o biblioteca debes usar **un arreglo de objetos**.

| Código | Evidencia |
| --- | --- |
| db.getCollection('books').find({  status: "PUBLISH",  authors: { $size: 2 }  }, {  title: 1,  longDescription: 1,  status: 1,  authors: 1  }).toArray() |  |
|
|
|

Qué ocurre si también le añades lo siguiente

.forEach(function(valor, indice, array){print("Titulo: " + valor.title + "Autho 1: " + valor.authors[0] + " Author 2: " + valor.authors[1] + " Registro No. " + indice);}).

¿Para qué crees que sería útil esto último?

Al convertir una colección en un arreglo mediante toArray(), podemos aprovechar métodos de iteración para construir listas personalizadas. En este caso, ordenaremos los resultados por título, autores y un índice generado durante la iteración. Esta técnica es versátil, ya que permite generar salidas formateadas en consola, realizar cálculos y validar datos de diversas maneras.

| Código | Evidencia |
| --- | --- |
| db.getCollection('books').find({  status: "PUBLISH",  authors: { $size: 2 }  }, {  title: 1,  longDescription: 1,  status: 1,  authors: 1  }).toArray().forEach(function(valor, indice, array){print("Titulo: " + valor.title + "Autho 1: " + valor.authors[0] + " Author 2: " + valor.authors[1] + " Registro No. " + indice);}) |  |
|
|
|

Consulta la colección 1

Sobre la colección books realiza las siguientes consultas

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

| Código | Evidencia |
| --- | --- |
| //Total de 517474 bytes  db.getCollection('books').dataSize() |  |
|
|
|

¿Cuántos libros tiene la colección?

| Código | Evidencia |
| --- | --- |
| //Total de 431 libros  db.getCollection('books').count() |  |
|
|
|

¿Cuántos libros tienen 200 o más páginas?

| Código | Evidencia |
| --- | --- |
| //Total 264 libros con mayor o igual a 200 páginas  db.getCollection('books').find({pageCount:{$gte:200}}).count() |  |
|
|
|

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

| Código | Evidencia |
| --- | --- |
| //Total 215 libros  db.getCollection('books').find({pageCount:{$gte:300, $lte:600}}).count() |  |
|
|
|

¿Cuántos libros tienen 0 páginas y cuántos no?

**Nota**: Con el objetivo de ilustrar este ejercicio, mostraré diversos ejemplos. Estos ejemplos me permitirán determinar, por un lado, la cantidad de libros sin páginas y, por otro, la cantidad de libros con páginas. Además, emplearé la función de agregación para obtener ambos resultados en una única operación. Finalmente, presentaré los resultados de cada ejemplo para su comprobación

| Código | Evidencia |
| --- | --- |
| // Libro 0 Páginas total de 166  db.getCollection('books').find({pageCount:0}).count()  // Libros Diferente a 0 Páginas total 265  db.getCollection('books').find({pageCount:{$ne:0}}).count()  //Consulta usando agregación  db.getCollection('books').aggregate(  [  {  $group: {  \_id: {  hasPages: {  $cond: [  { $eq: ['$pageCount', 0] },  'Libros 0 páginas',  'Libros con páginas'  ]  }  },  count: { $sum: 1 }  }  }  ]  ); |  |
|
|
|

¿Cuántos libros han sido publicados y cuántos no?

**Nota**: Mostraré diversos ejemplos para hallar el resultado, estos ejemplos me permitirán determinar, por un lado, la cantidad de libros que **sí** se han podido publicar, por otro, la cantidad de libros que **no** se han podido publicar. Además, emplearé la función de agregación para obtener ambos resultados en una única operación.

| Código | Evidencia |
| --- | --- |
| // Libros SI publicados total de 363  db.getCollection('books').find({status:'PUBLISH'}).count()  // Libros SI publicados total de 68  db.getCollection('books').find({status:{$ne:'PUBLISH'}}).count()  //Consulta usando agregación  db.getCollection('books').aggregate(  [  {  $group: {  \_id: {  hasPublish: {  $cond: [  { $eq: ['$status', 'PUBLISH'] },  'Total Libros SI Publicados',  'Total Libros NO Publicados'  ]  }  },  count: { $sum: 1 }  }  }  ]  ); |  |
|
|
|

Consulta la colección 2

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

| Código | Evidencia |
| --- | --- |
| // total 72236994 Bytes  db.getCollection('companies').dataSize() |  |
|
|
|

¿Cuántas compañías tiene la colección?

| Código | Evidencia |
| --- | --- |
| // total de registros en comapanies 18801  db.getCollection('companies').count() |  |
|
|
|

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

**Nota**: Para este ejercicio usaremos el método de agregación en combinación de match y group para mostrar los datos en una sola consulta.

| Código | Evidencia |
| --- | --- |
| //  db.getCollection('companies').aggregate(  [  {  $match: {  founded\_year: {  $in: [1996, 1997, 2001, 2005]  }  }  },  {  $group: {  \_id: '$founded\_year',  count: { $sum:1 }  }  }  , { $sort: { \_id: 1 } }  ]) |  |
|
|
|

Lista las compañías que se dedican a «web» o «mobile» y recupera: nombre, descripción, número de empleados, e-mail, año, mes y día de su fundación.

| Código | Evidencia |
| --- | --- |
| //  db.getCollection('companies').aggregate(  [  {  $match: {  category\_code: {  $in: ['web', 'mobile']  }  }  },  {  $project: {  \_id: 0,  name: 1,  description: 1,  number\_of\_employees: 1,  email\_address: 1,  founded\_year: 1,  founded\_month: 1,  founded\_day: 1  }  }  , { $sort: { name: 1 } }  ]) |  |
|
|
|

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

| Código | Evidencia |
| --- | --- |
| //  db.getCollection('companies').aggregate(  [  {  $match: {  category\_code: {  $in: ['games\_video']  }  }  },  {  $project: {  \_id: 0,  name: 1,  description: 1,  number\_of\_employees: 1,  email\_address: 1,  founded\_year: 1,  founded\_month: 1,  founded\_day: 1  }  }  , { $sort: { founded\_year: -1 } }  ]) |  |
|
|
|

¿Cuántas compañías tienen 600 o más empleados?

| Código | Evidencia |
| --- | --- |
| // Total de compañías con mas de 600 empleados  db.getCollection('companies').find({number\_of\_employees:{$gte:600}}).count(); |  |
|
|
|

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

| Código | Evidencia |
| --- | --- |
| // Resultado  db.getCollection('companies').aggregate(  [  {  $match: {  founded\_year: { $gte: 2001, $lte: 2005 },  number\_of\_employees: { $gte: 500 },  category\_code: {  $in: ['games\_video', 'music']  }  }  },  {  $project: {  name: 1,  homepage\_url: 1,  twitter\_username: 1,  number\_of\_employees: 1  }  }  , { $sort: { name: 1 } }  ]  ); |  |
|
|
|

¿Alguna empresa se dedica a videojuegos y a la música a la vez?

**Nota**: No se encontró datos para este ejercicio

| Código | Evidencia |
| --- | --- |
| // Resultado  db.getCollection('companies').aggregate(  [  {  $match: {  founded\_year: { $gte: 2001, $lte: 2005 },  number\_of\_employees: { $gte: 500 },  $and: [ { category\_code: "Music" }, { category\_code: "Game" } ]  }  },  {  $project: {  name: 1,  homepage\_url: 1,  twitter\_username: 1,  number\_of\_employees: 1  }  }  , { $sort: { name: 1 } }  ]  ); |  |
|
|
|

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

| Código | Evidencia |
| --- | --- |
| // Resultado  db.getCollection('companies').aggregate(  [  {  $match: { 'offices.city': 'San Francisco' }  },  {  $project: {  name: 1,  offices: {  $filter: {  input: '$offices',  as: 'office',  cond: {  $eq: [  '$$office.city',  'San Francisco'  ]  }  }  }  }  },  { $match: { offices: { $size: 2 } } }  ]  ); |  |
|
|
|

**Descripción de la consulta**:

Con el objetivo de resolver esta consulta, implementé una estrategia de agregación y proyección, al explorar diferentes enfoques para resolver esta consulta, descubrí que la combinación de los operadores de agregación $**match** y $**filter** en MongoDB era la más efectiva.

A través de varias pruebas, comprendí cómo utilizar $**filter** para iterar sobre el arreglo de oficinas y crear un nuevo arreglo con los elementos que cumplen una determinada condición. Este proceso me permitió no solo obtener los resultados esperados sino también profundizar en mis conocimientos sobre el manejo de datos estructurados.

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.

| Código | Evidencia |
| --- | --- |
| // Resultado  db.getCollection('companies').aggregate([  {  $match:{  category\_code:'games\_video',  'acquisition.price\_amount': {  $gte: 10000000  },  'acquisition.acquired\_year': 2007,  'offices.city': 'Culver City'  }  }  ,  {  $project:{  \_id:0,  name:1,  'acquisition.acquired\_month': 1,  'acquisition.acquired\_day': 1  }  }  ,  {$sort:{name:1}}  ]) |  |
|
|
|