



<> Code

Issues

Pull requests

Actions

Projects

Wiki

Security



main ▾

ud2-practica-1-mongo-db-Dansarasix-DML / Readme.md



Dansarasix-DML change imgs

8122223 · 8 months ago



244 lines (171 loc) · 8.18 KB

Preview

Code

Blame



Raw



Big Data Aplicado

UD2 - Procesado y Presentación de Datos Almacenados

Práctica 1 MongoDB

Recuerda que para hacer la prácticas puedes optar por cualquiera de la las 3 opciones.

- Instalarla en tu máquina local.
- Usar [Atlas MongoDB](#).
- Crear un contenedor docker.

1. Crear una base de datos MongoDB llamada "db_pract1", así como una colección con nombre "Hosteleria" en la que importes el contenido del archivo dataset_restaurantes.json

Creamos la base de datos en nuestro cluster en Mongo Atlas usando Compass.

Create Database ✕

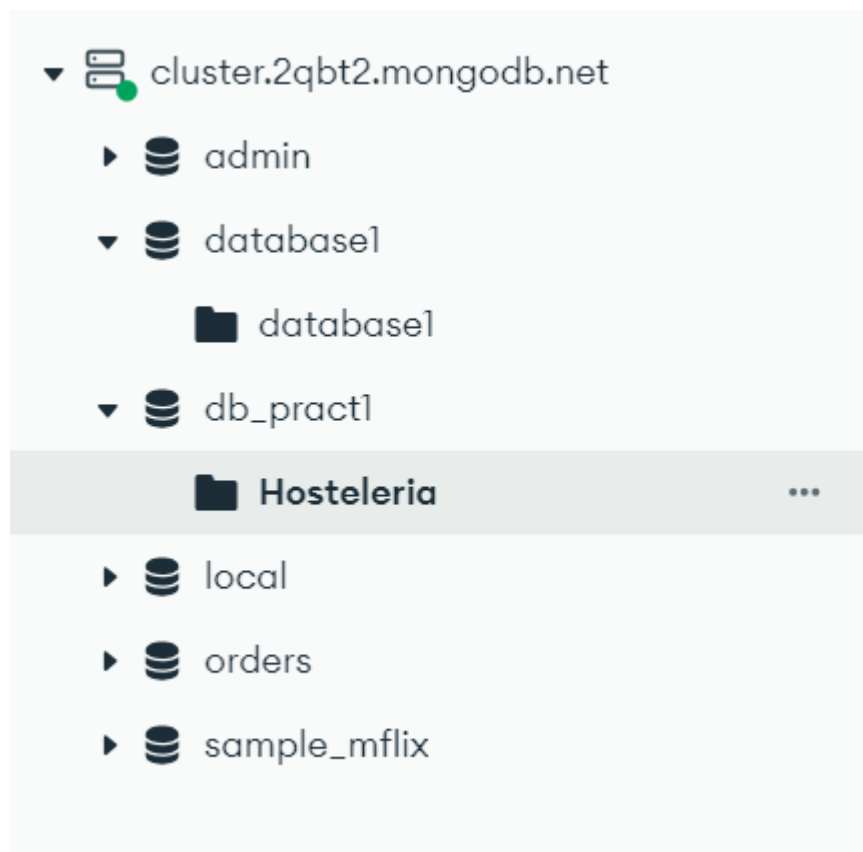
Database Name

Collection Name

☐ **Time-Series**
Time-series collections efficiently store sequences of measurements over a period of time. [Learn More](#)

➤ **Additional preferences** (e.g. Custom collation, Capped, Clustered collections)

Y en un momento se crea nuestra base de datos con nuestra colección



Ahora importamos la data del json.

[+ ADD DATA ▾](#)[EXPORT DATA ▾](#)[UPDATE](#)[DELETE](#)[Import JSON or CSV file](#)[Insert document](#)

This collection has no d

It only takes a few seconds to import data from a file.

[Import Data](#)

Import

To collection db_pract1.Hosteleria

Import file: dataset_restaurants.json

Options

☐ Stop on errors

[Cancel](#)[Import](#)

Importing dataset_restaurants.json...



14000 documents written.

[STOP](#)

Y ya aparece la data en nuestra base de datos.

ADD DATA

EXPORT DATA

UPDATE

DELETE

251 - 25 of 25359

```
{
  "_id": ObjectId('67136cea8e831985a9fe7882'),
  "address": {
    "borough": "Bronx",
    "cuisine": "Bakery"
  },
  "grades": [
    {
      "name": "Morris Park Bake Shop",
      "restaurant_id": "30075445"
    }
  ]
}
```

```
{
  "_id": ObjectId('67136cea8e831985a9fe7883'),
  "address": {
    "borough": "Brooklyn",
    "cuisine": "Hamburgers"
  },
  "grades": [
    {
      "name": "Wendy'S",
      "restaurant_id": "30112340"
    }
  ]
}
```

1. Sobre la base de datos creada anteriormente, realizar las siguientes consultas:

- a. Obtén el número total de documentos que se han importado a la colección Restaurantes

Usamos la función `countDocuments()` para contar el número de documentos que se subieron.

```
db.Hosteleria.countDocuments()
```



```
> db.Hosteleria.countDocuments()
< 25359
```

- b. Mostrar los datos de todos los restaurantes de cocina española (valor "Spanish"). ¿Cuántos restaurantes de este tipo de cocina hay en la colección? ¿Cuántos restaurantes sobre el total suponen los restaurantes de cocina española?

Usamos la función `find()` junto al valor que deseamos buscar.

```
db.Hosteleria.find( { cuisine: "Spanish" } )
```



```
> db.Hosteleria.find( { cuisine: "Spanish" } )
< {
  _id: ObjectId('67136cea8e831985a9fe78ed'),
  address: {
    building: '113',
    coord: [
      -73.9979214,
      40.7371344
    ],
    street: 'West 13 Street',
    zipcode: '10011'
  },
  borough: 'Manhattan',
  cuisine: 'Spanish',
  grades: [
    {
      date: 2014-07-25T00:00:00-0007
```

Para contar estos documentos usamos el mismo comando anterior pero añadiendo el valor que deseamos como el que pusimos en `find()` .

```
db.Hosteleria.countDocuments({ cuisine: "Spanish" })
```



```
> db.Hosteleria.countDocuments({ cuisine: "Spanish" })
< 637
```

MongoDB también permite hacer cálculos, por lo que para la última pregunta podemos juntar los comandos anteriores en una división y multiplicarlo por 10.

```
(db.Hosteleria.countDocuments({ cuisine: "Spanish"
}))/db.Hosteleria.countDocuments()*100
```



```
> (db.Hosteleria.countDocuments({ cuisine: "Spanish" }))/db.Hosteleria.countDocuments()*100
< 2.5119287038132416
```

El resultado obtenido es de 2.52% del total.

- c. Mostrar los nombres de todos los restaurantes de cocina portuguesa (valor "Portuguese") que se encuentren en el barrio de "Queens".

Para sacar los restaurantes con valor "Portuguese" y en el barrio "Queens" usamos `find()` y sobre el segundo valor hay que aplicarle la condición `$eq`.

```
db.Hosteleria.find( { cuisine: "Portuguese" , borough:
{$eq:"Queens"} } )
```



```
> db.Hosteleria.find( { cuisine: "Portuguese" , borough: {$eq:"Queens"} } )
< {
  _id: ObjectId('67136cea8e831985a9fe7c1b'),
  address: {
    building: '222-05',
    coord: [
      -73.732315,
      40.720725
    ],
    street: 'Jamaica Avenue',
    zipcode: '11428'
  },
  borough: 'Queens',
  cuisine: 'Portuguese',
  grades: [
    {
      date: 2014-09-17T00:00:00.000Z,
      grade: 'A',

```

Para solo mostrar los nombre añadimos otras llaves indicando el valor que solo queremos mostrar y `_id: 0` para ignorar el resto.

```
db.Hosteleria.find(
  { cuisine: "Portuguese", borough: { $eq: "Queens" } },
  { name: 1, _id: 0 }
)
```



```
> db.Hosteleria.find(
  { cuisine: "Portuguese", borough: { $eq: "Queens" } },
  { name: 1, _id: 0 }
)
< {
  name: 'Mateus Restaurant'
}
{
  name: 'A. Churrasqueira Restaurant'
}
{
  name: 'O Lavrador Restaurant'
}
{
  name: 'South Jamaica Portuguese Sporting Club'
}
```

- o d. Mostrar el nombre y la dirección de los restaurantes que tengan una valoración mayor de 90 puntos.

Para mostrar el nombre y la dirección de los restaurantes con `score > 90` hay que hacer una agregación.

Para hacer una agregación vamos al campo de 'Aggregations' y vamos creando nuevos 'Stages' hasta llegar al resultado deseado.

Lo primero será usar `$unwind` para desenrollar los `$grades`.

The screenshot shows the MongoDB Compass interface. On the left, the aggregation pipeline is defined with a single stage: `$unwind` with the path `"$grades"`. The right panel shows the output after the `$unwind` stage, displaying a sample of 10 documents. The first document shown is:

```
{
  "_id": ObjectId("67136cea8e831985a9fe7882"),
  "address": {
    "borough": "Bronx",
    "cuisine": "Bakery",
    "grades": {
      "name": "Morris Park Bake Shop",
      "restaurant_id": "30075445"
    }
  }
}
```

Luego agrupamos por nombre y suma con `$group`.

Stage 2 `$group`

```

1  /**
2   * _id: The id of the group.
3   * fieldN: The first field name.
4   */
5  {
6    _id: "$_id",
7    totalScore: { $sum: "$grades.score" },
8    name: { $first: "$name" },
9    address: { $first: "$address" }
10 }
11

```

Output after `$group` stage (Sample of 10 documents)

```

_id: ObjectId('67136cf58e831985a9fec8c8')
totalScore: 23
name: "Louie And Chan"
address: Object

```

```

_id: ObjectId('67136cf58e831985a9fea470')
totalScore: 51
name: "Pronto"
address: Object

```

Luego filtramos las sumas mayores a 90 con `$match`.

Stage 3 `$match`

```

1  /**
2   * query: The query in MQL.
3   */
4  {
5    totalScore: { $gt: 90 }
6  }

```

Output after `$match` stage (Sample of 10 documents)

```

_id: ObjectId('67136cf08e831985a9fea470')
totalScore: 109
name: "Gmc Temaxcal Deli & Grocery"
address: Object

```

```

_id: ObjectId('67136cf58e831985a9fec8c8')
totalScore: 109
name: "Gmc Temaxcal Deli & Grocery"
address: Object

```

Y por último usamos `$project` para mostrar solo el nombre y la dirección.

Stage 4 `$project`

```

1  /**
2   * specifications: The fields to
3   * include or exclude.
4   */
5  {
6    _id: 0,
7    name: 1,
8    address: 1
9  }
10

```

Output after `$project` stage (Sample of 10 documents)

```

name: "Gmc Temaxcal Deli & Grocery"
address: Object

```

```

name: "Dagan I"
address: Object

```

Una vez hemos terminado, ejecutamos la sentencia.

☰
Sunwind
\$group
\$match
\$project
Generate aggregation
?
Explain
Export
Run
Options

Y tras la ejecución, salen los resultados de la agregación.


```

    name : "Gmc Temaxcal Deli & Grocery"
  ▼ address : Object
    building : "163"
    ▶ coord : Array (2)
    street : "Park Avenue"
    zipcode : "11205"

```

```

    name : "Dagan Pizza & Dairy Restaurant"
  ▼ address : Object
    building : "4820"
    ▶ coord : Array (2)
    street : "16 Avenue"
    zipcode : "11204"

```

```

    name : "Portobello Cafe"
  ▶ address : Object

```

```

db.getCollection('Hosteleria').aggregate(
[
  { $unwind: { path: '$grades' } },
  {
    $group: {
      _id: '$_id',
      totalScore: { $sum: '$grades.score' },
      name: { $first: '$name' },
      address: { $first: '$address' }
    }
  },
  { $match: { totalScore: { $gt: 90 } } },
  { $project: { _id: 0, name: 1, address: 1 } }
],
{ maxTimeMS: 60000, allowDiskUse: true }
);

```



- e. ¿Cuántos restaurantes de cocina española tienen una puntuación menor a 50 puntos?

Para esto hacemos otra agregación. Hacemos el primer paso y en el segundo cogemos el valor de 'cuisine' en vez de 'address'.

Stage 2 \$group

```

1  /**
2   * _id: The id of the group.
3   * fieldN: The first field name.
4   */
5  {
6    _id: "$_id",
7    totalScore: { $sum: "$grades.score" },
8    name: { $first: "$name" },
9    cuisine: { $first: "$cuisine" }
10 }

```

Output after \$group stage (Sample of 10 documents)

```

_id: ObjectId('67136cf28e831985a9feb9e8')
totalScore: 18
name: "Cloud 9"
cuisine: "American "

_id: ObjectId('67136cf08e831985a9feaa77')
totalScore: 39
name: "La Puntilla Bar & Restaurant"
cuisine: "Spanish"

```

Luego en el \$match necesitamos las 'score' menores a 50 y que 'cuisine' sea 'Spanish'.

Stage 3 \$match

```

1  /**
2   * query: The query in MQL.
3   */
4  {
5
6    totalScore: { $lt: 50 },
7    cuisine: { $eq: "Spanish" }
8  }

```

Output after \$match stage (Sample of 10 documents)

```

_id: ObjectId('67136cf08e831985a9feaa77')
totalScore: 39
name: "La Puntilla Bar & Restaurant"
cuisine: "Spanish"

_id: ObjectId('67136cf28e831985a9feb9e8')
totalScore: 18
name: "Cloud 9"
cuisine: "American "

```

Por último hacemos un \$count para contar todos los resultados.

Stage 4 \$count

```

1  /**
2   * Provide the field name for the count.
3   */
4  'cantidad'

```

Output after \$count stage (Sample of 1 document)

```

cantidad: 428

```

Ejecutamos la agregación.

ALL RESULTS

OUTPUT OPTIONS ▼

cantidad : 428

```

db.getCollection('Hosteleria').aggregate(
[
  { $unwind: { path: '$grades' } },

```



```
{
  $group: {
    _id: '$_id',
    totalScore: { $sum: '$grades.score' },
    name: { $first: '$name' },
    cuisine: { $first: '$cuisine' }
  }
},
{
  $match: {
    totalScore: { $lt: 50 },
    cuisine: { $eq: 'Spanish' }
  }
},
{ $count: 'cantidad' }
],
{ maxTimeMS: 60000, allowDiskUse: true }
);
```

- o f. Mostrar el nombre y el tipo de cocina de los cinco restaurantes con mayor puntuación.

Para hacer esta agregación dejamos las 2 etapas de la anterior y empezamos a agregar nuevas etapas acordes al enunciado.

Ordenamos los resultados con `$sort` en orden del mayor al menor.

Stage 3 `$sort`

```
1 ▾ /**
2   * Provide any number of field/order pair
3   */
4 ▾ {
5   totalScore: -1
6 }
```

Output after `$sort` stage (Sample of 10 documents)

```
_id: ObjectId('67136cf18e831985a9feaf4b')
totalScore: 254
name: "Red Chopstick"
cuisine: "Chinese"
```

Luego usamos `$limit` para sacar los 5 primeros.

Stage 4 `$limit`

```
1 ▾ /**
2   * Provide the number of documents to lim
3   */
4 5
```

Output after `$limit` stage (Sample of 5 documents)

```
_id: ObjectId('67136cf18e831985a9feaf4b')
totalScore: 254
name: "Red Chopstick"
cuisine: "Chinese"
```

Y por último hacemos `$project` para mostrar los resultados.

Stage 5 \$project

```

1  /**
2   * specifications: The fields to
3   *   include or exclude.
4   */
5  {
6    _id: 0,
7    name: 1,
8    cuisine: 1
9  }

```

Output after \$project stage (Sample of 5 documents)

```

name : "Red Chopstick"
cuisine : "Chinese"

name : "Nios Restaurant"
cuisine : "American "

name : "Nanni Restaurant"
cuisine : "Italian"

name : "Amici 36"
cuisine : "American "

name : "Cheikh Umar Futiyu Restaurant"
cuisine : "African"

```

```

db.getCollection('Hosteleria').aggregate(
[
  { $unwind: { path: '$grades' } },
  {
    $group: {
      _id: '$_id',
      totalScore: { $sum: '$grades.score' },
      name: { $first: '$name' },
      cuisine: { $first: '$cuisine' }
    }
  },
  { $sort: { totalScore: -1 } },
  { $limit: 5 },
  { $project: { _id: 0, name: 1, cuisine: 1 } }
],
{ maxTimeMS: 60000, allowDiskUse: true }
);

```

ALL RESULTS OUTPUT OPTIONS

Showing 1 - 5 count results < > VIEW

```

name : "Red Chopstick"
cuisine : "Chinese"

```

```

name : "Nios Restaurant"
cuisine : "American "

```

```

name : "Nanni Restaurant"
cuisine : "Italian"

```

```

name : "Amici 36"
cuisine : "American "

```

```

name : "Cheikh Umar Futiyu Restaurant"
cuisine : "African"

```

- o g. Mostrar el nombre y el tipo de cocina de los cinco restaurantes con menor puntuación

Simplemente cambiamos el \$sort para que sea del menor al mayor.

```

db.getCollection('Hosteleria').aggregate(
[
  { $unwind: { path: '$grades' } },

```

```
{
  $group: {
    _id: '$_id',
    totalScore: { $sum: '$grades.score' },
    name: { $first: '$name' },
    cuisine: { $first: '$cuisine' }
  }
},
{ $sort: { totalScore: 1 } },
{ $limit: 5 },
{ $project: { _id: 0, name: 1, cuisine: 1 } }
],
{ maxTimeMS: 60000, allowDiskUse: true }
);
```

ALL RESULTS OUTPUT OPTIONS ▾Showing 1 - 5 [count results](#) < > VIEW ≡ { }

name : "CaffeBene"
cuisine : "Café/Coffee/Tea"

name : "Madison Club (Bb7184)"
cuisine : "American "

name : "Centerplate-Employee Cafeteria-Jacob K Javits Convention Center"
cuisine : "Other"

name : "Papa John'S"
cuisine : "Pizza"

name : "Guchun Private Kitchen"
cuisine : "Chinese"