APUNTES MONGODB - ABEL RIOS - 22.04.2022

* Documentación para la instalación:
https://www.mongodb.com/docs/v4.4/tutorial/install-mongodb-on-windows/

Al iniciar, tenemos que arrancar el servicio en una consola, abriendo el archivo mongod.exe de la carpeta bin, también le decimos el directorio de la carpeta donde vamos a almacenar nuestras bases de datos:

Comando: "C:\ProgramFiles\MongoDB\Server\5.0\bin\mongod.exe" --dbpath="c:\data\db"

Una vez arrancado el servicio, podemos abrir una segunda consola y ejecutar el mongo.exe para empezar a operar con las bases de datos:

Comando: "C:\Program Files\MongoDB\Server\5.0\bin\mongo.exe"

Para familiarizarnos un poco con la semántica, en MongoDB no hablamos de “tablas”, hablamos de “colecciones o collections”.

- **Comandos Básicos:**

- db , te muestra la base de datos actual en uso (test por defecto)

- show dbs , muestra las bases de datos disponibles

- use < database >, cambia a otra base datos (la crea si no existe)

- show collections , muestra las colecciones de la base de datos

- db.myCollection.insertOne ({ x: 1 }), inserta un documento en la colección (si la colección no existe la crea vacía)

- db.myCollection.find (), lista todos los documentos de la colección myCollection

- **Insertando Datos (uno/muchos):**

- db.contenidos.insertOne ({ …})

- db.contenidos.insertMany ([{ … }, { …}, {...}])

Donde ‘contenidos’ es la collection que estamos utilizando ahora.

- **Ejercicios**

EJ: Obtener el listado de ventas con nombre de usuario y el tipo de pago realizado (si la colección ventas no tiene tipo de pago, incluirlo)

```
> db.ventas.find({},{_id:0, "usuario.nombre":true, "usuario.tipoPago": true})
{ "usuario" : { "nombre" : "Jose", "tipoPago" : "Paypal" } }
{ "usuario" : { "nombre" : "Jose", "tipoPago" : "Paypal" } }
{ "usuario" : { "nombre" : "Pedro" } }
{ "usuario" : { "nombre" : "Maria", "tipoPago" : "Bizum" } }
{ "usuario" : { "nombre" : "Jose", "tipoPago" : "Paypal" } }
{ "usuario" : { "nombre" : "Claudia", "tipoPago" : "Paypal" } }
{ "usuario" : { "nombre" : "Jose", "tipoPago" : "Paypal" } }
{ "usuario" : { "nombre" : "Maria", "tipoPago" : "Bizum" } }
{ "usuario" : { "nombre" : "Claudia", "tipoPago" : "Paypal" } }
{ "usuario" : { "nombre" : "Maria", "tipoPago" : "Bizum" } }
{ "usuario" : { "nombre" : "Jose", "tipoPago" : "Paypal" } }
{ "usuario" : { "nombre" : "Pedro" } }
{ "usuario" : { "nombre" : "Claudia", "tipoPago" : "Paypal" } }
>
```

Si no le dijéramos _id:0, nos enseñará las id de cada objeto/registro:

```
> db.ventas.find({},{ "usuario.nombre":true, "usuario.tipoPago": true})
{ "_id" : ObjectId("6262c251403760c20adab52e"), "usuario" : { "nombre" : "Jose", "tipoPago" : "Paypal" } }
{ "_id" : ObjectId("6262c251403760c20adab52f"), "usuario" : { "nombre" : "Jose", "tipoPago" : "Paypal" } }
{ "_id" : ObjectId("6262c251403760c20adab530"), "usuario" : { "nombre" : "Pedro" } }
{ "_id" : ObjectId("6262c251403760c20adab531"), "usuario" : { "nombre" : "Maria", "tipoPago" : "Bizum" } }
{ "_id" : ObjectId("6262c251403760c20adab532"), "usuario" : { "nombre" : "Jose", "tipoPago" : "Paypal" } }
{ "_id" : ObjectId("6262c251403760c20adab533"), "usuario" : { "nombre" : "Claudia", "tipoPago" : "Paypal" } }
{ "_id" : ObjectId("6262c251403760c20adab534"), "usuario" : { "nombre" : "Jose", "tipoPago" : "Paypal" } }
{ "_id" : ObjectId("6262c251403760c20adab535"), "usuario" : { "nombre" : "Maria", "tipoPago" : "Bizum" } }
{ "_id" : ObjectId("6262c251403760c20adab536"), "usuario" : { "nombre" : "Claudia", "tipoPago" : "Paypal" } }
{ "_id" : ObjectId("6262c251403760c20adab537"), "usuario" : { "nombre" : "Maria", "tipoPago" : "Bizum" } }
{ "_id" : ObjectId("6262c251403760c20adab538"), "usuario" : { "nombre" : "Jose", "tipoPago" : "Paypal" } }
{ "_id" : ObjectId("6262c251403760c20adab539"), "usuario" : { "nombre" : "Pedro" } }
{ "_id" : ObjectId("6262c251403760c20adab53a"), "usuario" : { "nombre" : "Claudia", "tipoPago" : "Paypal" } }
>
```

EJ: Sacar por consola lista de productos donde aparezca su nombre y la cantidad vendida

```
>
> db.ventas.find({},{_id:0, "producto.nombre":true, "cantidad": true})
{ "producto" : { "nombre" : "Correa perros" }, "cantidad" : 2 }
{ "producto" : { "nombre" : "Correa perros" }, "cantidad" : 2 }
{ "producto" : { "nombre" : "Correa perros" }, "cantidad" : 1 }
{ "producto" : { "nombre" : "Lampara de pie" }, "cantidad" : 1 }
{ "producto" : { "nombre" : "Lampara de pie" }, "cantidad" : 4 }
{ "producto" : { "nombre" : "Lampara de pie" }, "cantidad" : 2 }
{ "producto" : { "nombre" : "Mesa" }, "cantidad" : 3 }
{ "producto" : { "nombre" : "Mesa" }, "cantidad" : 2 }
{ "producto" : { "nombre" : "Silla Gaming" }, "cantidad" : 10 }
{ "producto" : { "nombre" : "Silla Gaming" }, "cantidad" : 2 }
{ "producto" : { "nombre" : "Silla Gaming" }, "cantidad" : 3 }
{ "producto" : { "nombre" : "Televisor" }, "cantidad" : 1 }
{ "producto" : { "nombre" : "Televisor" }, "cantidad" : 2 }
>
```

Para hacer un GROUP BY en esta query usamos el comando ‘aggregate’:

```
> db.ventas.aggregate({$group:{_id:"$producto.nombre",totalVentas:{$sum:"$cantidad"}}})
{ "_id" : "Silla Gaming", "totalVentas" : 15 }
{ "_id" : "Televisor", "totalVentas" : 3 }
{ "_id" : "Lampara de pie", "totalVentas" : 7 }
{ "_id" : "Mesa", "totalVentas" : 5 }
{ "_id" : "Correa perros", "totalVentas" : 5 }
>
```

EJ: Sacar por consola Lista de productos que se muestre el nombre y el rating

```
> db.productos.find({},{_id:0,"nombre":true, "rating":true})
{ "nombre" : "Correa perros", "rating" : 4 }
{ "nombre" : "Lampara de pie", "rating" : 4 }
{ "nombre" : "Mesa", "rating" : 3 }
{ "nombre" : "Silla Gaming", "rating" : 4 }
{ "nombre" : "Televisor", "rating" : 2 }
>
```

EJ: Sacar por consola cual es el producto más vendido y nombre de usuarios han comprado dicho producto

Primero hacemos la agrupación de antes, y la ordenamos con el 'sort'

```
> db.ventas.aggregate([{$group:{_id:"$producto.nombre",totalVentas:{$sum:"$cantidad"}}},{$sort:{totalVentas:-1}}])
{ "_id" : "Silla Gaming", "totalVentas" : 15 }
{ "_id" : "Lampara de pie", "totalVentas" : 7 }
{ "_id" : "Mesa", "totalVentas" : 5 }
{ "_id" : "Correa perros", "totalVentas" : 5 }
{ "_id" : "Televisor", "totalVentas" : 3 }
```

Ahora hacemos el limit, para quedarnos con únicamente el primero (el más vendido)

```
> db.ventas.aggregate([{$group:{_id:"$producto.nombre",totalVentas:{$sum:"$cantidad"}}},{$sort:{totalVentas:-1}},{$limit:1}])
{ "_id" : "Silla Gaming", "totalVentas" : 15 }
>
```

OJO*
Al usar el aggregate, todas las funciones que usemos (group, sort, limit…) tienen que ir dentro de un array.

EJ: Sacar por consola el producto con su nombre y tipo que mayor ganancia ha generado.

```
> db.ventas.aggregate([{$group:{_id:"$producto.nombre",ganancias:{$sum:{$multiply:
["$cantidad","$precio"]}}}},{$sort:{ganancias:-1}},{$limit:1}])
{ "_id" : "Televisor", "ganancias" : 2790 }
```

Hemos hecho el 'aggregate' para agruparlos, llamamos 'ganancias' al producto de cantidad por precio, lo ordenamos por ganancias de forma descendente y nos quedamos únicamente con el primero de la lista (limit:1)

EJ: Sacar por consola el nombre de usuario que más artículos ha comprado

```
> db.ventas.aggregate([{$group:{_id:"$usuario.correo",articulosComprados:{$sum:"$cantidad"}}},
{$sort:{articulosComprados:-1}}])
{ "_id" : "clau@clau.com", "articulosComprados" : 14 }
{ "_id" : "jose@gmail.com", "articulosComprados" : 14 }
{ "_id" : "mp@hotmail.es", "articulosComprados" : 5 }
{ "_id" : "pedro@pedro.es", "articulosComprados" : 2 }
>
>
> db.ventas.aggregate([{$group:{_id:"$usuario.correo",articulosComprados:{$sum:"$cantidad"}}},
{$sort:{articulosComprados:-1}},{$limit:1}])
{ "_id" : "jose@gmail.com", "articulosComprados" : 14 }
>
```

EJ: Sacar por consola el nombre de usuario que se ha gastado más dinero.

```
> db.ventas.aggregate([{$group:{_id:"$usuario.correo",gastoTotal:{$sum:{$multiply:["$cantidad"
,"$precio"]}}}},{$sort:{gastoTotal:-1}},{$limit:1}])
{ "_id" : "clau@clau.com", "gastoTotal" : 2532.4 }
>
>
> db.ventas.aggregate([{$group:{_id:"$usuario.correo",gastoTotal:{$sum:{$multiply:["$cantidad"
,"$precio"]}}}},{$sort:{gastoTotal:-1}}])
{ "_id" : "clau@clau.com", "gastoTotal" : 2532.4 }
{ "_id" : "pedro@pedro.es", "gastoTotal" : 925.9 }
{ "_id" : "jose@gmail.com", "gastoTotal" : 670.05 }
{ "_id" : "mp@hotmail.es", "gastoTotal" : 307.2 }
>
```

Tenemos ambas opciones en la solución, usando el ‘limit’ para quedarnos sólo con el que más dinero ha gastado, y sin el ‘limit’ para ver los gastos de cada usuario.

Aquí utilizamos la función ‘concat’ para mostrar el nombre:

```
> db.ventas.aggregate([{$group:{_id:{$concat:["$usuario.nombre"," ","$usuario.apellidos"]},gastoTotal:{$sum:{$multiply:
["$cantidad","$precio"]}}}},{$sort:{gastoTotal:-1}}])
{ "_id" : "Claudia Semper", "gastoTotal" : 2532.4 }
{ "_id" : "Pedro Lopez", "gastoTotal" : 925.9 }
{ "_id" : "Jose Alcala", "gastoTotal" : 670.05 }
{ "_id" : "Maria Perez", "gastoTotal" : 307.2 }
>
```

He intentado hacer el id:usuario.correo y añadir otra columna con el nombre con el concat, pero el aggregate no te lo permite, sólo te deja dos columnas.