EJERCICIOS INDICES

Ejercicio 1º (indices sobre campos de subdocumentos)

Importar la coleccion clientes y realiza la siguiente consulta

```
db.Clientes.find({"Name.Last Name" : "Johnston"},
          {"_id" :0,"Name.First Name":1, "Name.Last Name" :1})
          .sort({"Name.Last Name":1}).explain("executionStats")
```

Crear un indice para el Last Name

Repetir la consulta anterior y comprobar como desciende el tiempo de ejecución

Comprueba los indices que hay creados

Borrar el indice de Last Name

Crear un nuevo indice que incluya tanto el Last Name como el First Name (indice compuesto)

Repetir la consulta anterior y comprobar como desciende el tiempo de ejecución. En este caso tarda menos porque ni siquiera es necesario consultar la colección, pues toda la información necesaria esta en el propio indice. Esta situacion se denomina "Covered Query"

Ejercicio 2º (La importancia del orden en los indices compuestos)

Realizar esta consulta:

```
db.Clientes.find().sort({"Name.Last Name":1,"Name.First Name":-1})
```

A pesar de que tenemos un indice para Last Name y First Name, no se puede ejecutar la consulta porque se supera el limite de 32MB para operaciones sort

Borrar todos los indices y crear el siguiente

```
db.Clientes.createIndex( { "Name.Last Name" : 1,"Name.First Name" : -1 })
```

Comprobar que ahora se puede realizar tanto la consulta anterior como esta:

```
db.Clientes.find().sort({"Name.Last Name":-1,"Name.First Name":1})
```

Borrar todos los indices y crear el siguiente

```
db.Clientes.createIndex({"Name.First Name":-1,"Name.Last Name":1})
```

¿Se podrá realizar la consulta anterior?

Indices multiclave: Indexan automáticamente todos los elementos de un array

Comprobar que la siguiente consulta no utiliza ningún indice:

```
db.personas.find({Asignaturas:"Fisica"}).explain("executionStats")
```

Crear el siguiente indice

```
db.personas.createIndex({"Asignaturas": 1})
```

Verificar que ahora la misma consulta anterior sí utiliza este indice

Crear el siguiente indice

```
db.personas.createIndex({Nominas: 1})
```

Comprobar que la siguiente consulta no lo aprovecha

```
db.personas.find({"Nominas.Mes": "Octubre"})
```

Pero ésta sí:

```
db.personas.find({"Nominas":
{"Mes":Octubre","Cantidad":1850.90}).explain("executionStats")
```

Para que pudiera usar un indice tendríamos que crearlo, por ejemplo, así:

```
db.personas.createIndex({"Nominas.Mes":1,"Nominas.Cantidad":-1})
```

¿Por qué?

Ejercicio 3º:

Añadir a los 3 profesores de personas un campo GeoJSON de tipo Point llamado ubicación, con las coordenadas de Madrid, Santander y Cádiz

Buscar todos los profesores cuya ubicación este a menos de 300km de Córdoba

Ejercicio 4º: Búsqueda de restaurantes por geometrías y por proximidad

Importar los datasets de **barrios** y **restaurantes.** Observar que los restaurantes estan ubicados por el campo **location.** Y que los barrios lo estan por el campo

geometry

Supongamos que estamos en las coordenadas [-73.856077, 40.848447]. Averiguar el barrio al que corresponden estar coordenadas.

Averiguar todos los restaurantes que hay en un barrio anterior.

Averiguar que restaurantes están a menos de 1000 metros de la posición anterior. Mostrar primero sin ordenar y posteriormente ordenados por proximidad.