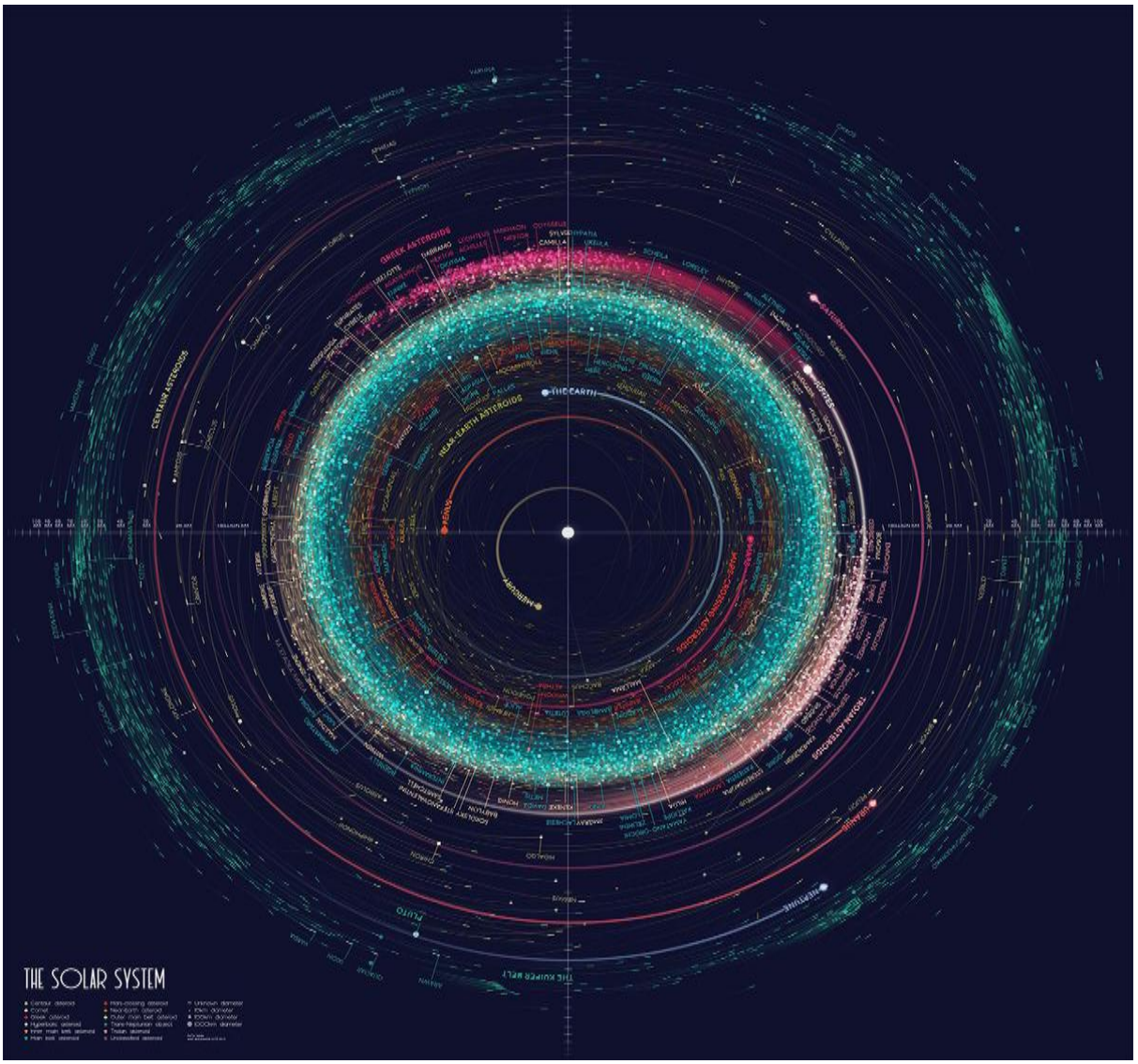
Proyecto Final de la 1º evaluación



Nombre:José Miguel
Apellidos:Rodríguez Muñoz
Curso:1ºAsir A
Fecha:13/11/2021

Explicación de consultas:

```
db.planetas.find({$and:[{tamaño:{$ne:"normal"}},
{velocidaddeescape:{$gte:1}}, {atmósfera:{$eq:["N2","CH4"
]}}]}).pretty()
```

La realización de esta consulta tiene el fin de encontrar los planetas que cumplan con una velocidad de escape mayor o igual a 1,que tengan un tamaño enano, y que contengan en su atmósfera los gases N2 y CH4.
En el inicio de la consulta utilizamos $and para que la resolución cumpla con todo los requisitos que hayamos escrito.
Usando $ne seleccionamos los documentos donde el valor de tamaño no es igual al especificado (“normal”).
Luego, utilizando $gte obtenemos los documentos donde el valor de velocidad de escape es mayor o igual a 1.
Para terminar se usa $eq para comparar documentos donde el valor de atmósfera es igual al (“N2”,”CH4”).El .pretty() se usa para que devuelva los datos en un formato que sea más fácil de leer.

```
db.planetas.find({$and:[{"estudio.satélites":{$regex:/[0-9$]/}},{cercad
elsol:{$exists: true}},{tamaño:{$not:{$eq:"gigante"}}}]}).pretty()
```

El fin de esta consulta es encontrar los planetas que hayan sido visitados por varios modelos de satélites y varias veces ,que estén cercanos al sol y sean de tamaño similar al de la tierra.
Al principio de la consulta utilizamos $and para que el resultado cumpla con todo los requisitos que hayamos redactado.
En esta consulta accedemos al campo dirección mediante ese punto entre estudio y satélites.
A continuación,usamos $regex para encontrar cualquier carácter entre corchetes que sea un dígito.Y así poder encontrar los satélites que tengan diferentes modelos y que los haya visitado más de una vez.
Luego ,tenemos el operador $exist para hacer coincidir todos los documentos en el que el campo cerca del sol es true.
Y por último, $not + $eq para seleccionar que el tamaño no sea gigante.
El .pretty() se usa para que devuelva los datos en un formato que sea más fácil de leer.

```
db.planetas.find({$and:[{date:{$lte:new Date("1975-01-01")} },{$nor:[
{acercamiento:"Estados Unidos"}, {acercamiento:"Italia"}]}]}).pretty()
```

El objetivo de esta consulta son los planetas estudiados por China,y que dichos planetas se hubieran documentado antes del 1975-01-02.
Primero, usamos $and para que el resultado cumpla todo los requisitos que hayamos determinado.
A continuación ,se usa $lte para encontrar los documentos donde el valor de date es menor o igual a la fecha de 1975-01-01.
Por último, utilizamos $nor para seleccionar los documentos que no sean igual a todas las expresiones (“Estados Unidos”,”Italia”).El .pretty() se usa para que devuelva los datos en un formato que sea más fácil de leer.

```
db.planetas.find({$and:[{ "estudio.satélites": { $size: 1 } }
,{$nor:[{atmósfera:["N2","CH4"]},{tiposdeplanetas:["fluido","helado"]}]
},{lunas:{$eq:0}}]}).pretty()
```

Con esta consulta se pretende encontrar un planeta, el cual tenga una atmósfera no esté compuesta por N2 y CH4,sino H2O y O2,ni tampoco que sea helado y fluido,que no tenga lunas,y que haya sido descubierto por un solo satélite.
Se usa $and ,el cual ya está explicado varias veces en los anteriores apartados.
En esta consulta accedemos al campo dirección mediante ese punto entre estudio y satélites.
Posteriormente,usamos $size para hacer coincidir la matriz con el número de elemento especificado en este caso sería 1 para encontrar,los que solamente tengan un satélite.
Usamos $nor para obtener los documentos que no sean igual a todas las expresiones dichas (“N2”,”CH4”)(“fluido”,”helado”).
Y para finalizar tenemos un $eq ,que recoge documentos donde el valor lunas es igual al valor 0.El .pretty() se usa para que devuelva los datos en un formato que sea más fácil de leer.

```
 db.planetas.find ( { órbita: {$all:[{
"$elemMatch":{cercadelcinturóndeasteroides: "Si", distancia: { $lt:
30000 } } }] }})
```

En esta consulta,queremos los satélites naturales que estén cerca de un cinturón de asteroides,y que estén a una distancia menor a 30000 km.
Primero usamos $all para seleccionar documentos donde el valor de órbita es una matriz cuyos elementos coinciden con los $elemMatch.Despúes usamos $elemMatch para solo coincidir con aquellos documentos en los que el resultado contenga el elemento cercadelcinturóndeasteroides igual a"Si",y por último en el campo distancia buscará que sea menor a 30000 con el operador $lt.

```
db.planetas.find({$or:[{lunas:{$gt:2}}],tamaño:{$eq:"enano"}}).pretty()
```

Esta consulta trata de buscar planetas enanos que tengan más de dos lunas.
En un principio tenemos el operador $or, que seleccionará todos los documentos de la colección planetas donde el valor del campo lunas es mayor que 2 **o** el valor del campo tamaño es igual 10.El .pretty() se usa para que devuelva los datos en un formato que sea más fácil de leer.

```
db.planetas.find({},{_id:0,"atmósfera":{$slice:1}}).pretty()
```

El fin de esta consulta ,es obtener el primer gas ,por el cuál está compuesta la atmósfera de cada planeta de nuestro sistema planetario.
Usamos _id:0 ,para que todos los documentos aparezcan con todos los campos,ya que la proyección $slice excluye el id del campo.
Luego usamos $slice para que devuelva el número de elemento especificado,en este caso es 1.El .pretty() se usa para que devuelva los datos en un formato que sea más fácil de leer.

```
db.planetas.find({$nor:[{$jsonSchema:{required:["órbita"],
properties:{satélitesnaturales:{bsonType:"string"},distancia:{bsonType:
"double"},cercadelcinturódenasteroides:{bsonType:"bool"}}}}]}).pretty()
```

Esta consulta pretende hallar los planetas que no tienen ninguna luna en su órbita.
Se usa $jsonSchema(coincide con los documentos que satisfacen el esquema JSON especificado.) + $nor para buscar todos los documentos que no satisfacen el esquema,en este caso que no tenga el campo órbita,para que aparezcan solo los planetas que no tengan lunas en la órbita.El .pretty() se usa para que devuelva los datos en un formato que sea más fácil de leer.

```
db.planetas.find({descubrimiento:{$not:{$regex:/^P./i}
},acercamiento:{$eq:["España"]}}).pretty()
```

Esta consulta pretende hallar un planeta que haya sido descubierto en la actualidad ,y que la documentación lo haya realizado España.
En la siguiente consulta utilizamos el operador $not junto a $regex hace que seleccione documentos donde el valor del campo descubrimiento no comience por la letra p,usando ^ para decir que comienza por tal letra,el . para indicar que sigue habiendo cualquier carácter después excepto nueva línea,con la i hacemos que sea indiferente la mayúscula o minúscula.
Y para finalizar usamos $eq,para que el valor del campo acercamiento sea igual a España.El .pretty() se usa para que devuelva los datos en un formato que sea más fácil de leer.

```
db.planetas.find( { "estudio.desarrolladores": { $in:[
/^E.A$/,/^JA.A$/] } } )
```

En esta consulta se quieren hallar las agencias de satélites que sean europea y japonesa.

En esta consulta accedemos al campo dirección mediante ese punto entre estudio y satélites.A continuación,usaremos el operador $in que selecciona todos los documentos de la colección donde el campo desarrolladores contiene una cadena que comienza con Eo JA debido a la expresión regular ^,y termina en A con la expresión regular $.El .pretty() se usa para que devuelva los datos en un formato que sea más fácil de leer.

Implementación nueva

```
db.planetas.aggregate([{$sort:{satélites:1}},{$group:{_id:"idAux",
primero:{$first:'$estudio.satélites'},ultimo:{$last:'$estudio.satélites
'}}}])
```

Este operador se ha utilizado para realizar una operación en los datos agrupados para que devuelva un único resultado.Ente caso para queremos que nos devuelva los valores de satélites del primer documento y del último documento de nuestra base de dato.Ya que se quiere saber los satélites del primer planeta y si el último tiene alguno.

Primero tenemos el operador $sort el cual ordena todos los documentos de entrada y los devuelve a la canalización en orden ordenado.

El campo satélites equivale a 1 para qué $sort haga que orden de los documentos sea de manera ascendente.

A continuación,$group agrupara los documentos de entrada por la _id expresión especificada,que en este caso será una id Auxiliar.

Luego,usamos $first para devolver el valor que resulta de aplicar una expresión al primer documento de un grupo de documentos.En este caso accedemos al campo dirección mediante ese punto entre estudio y satélites.

Y por último,$last que realiza la inversa de $first devuelve el valor que resulta de aplicar una expresión al último documento de un grupo de documentos.