Práctica 3: MongoDB-Agregación (y II)

0. Iniciar el servidor (cd; rm -rf datos; mkdir datos; mongod -dbpath datos)
1. Bajar del campus el fichero datos.txt
2. Copia y pega el contenido del fichero para crear las 3 colecciones: users, capturas, bichos
.

Nota:
- No olvidar poner los nombres en solucion.txt
- Copiar en solucion.txt solo consultas; **no copiar el prompt >**

La compañía de software "Problemon Go" (PG de aquí en Adelante) tiene un juego de realidad aumentada en los que los usuarios capturan bichos diversos. Cada bicho capturado tiene unos puntos.

Copiar en la solución vistas para obtener los resultados de las siguientes consultas.
Nota:

1. En cada vista se pueden utilizar las vistas anteriores como punto de partida, si no se dice lo contrario
2. Se valora la simplicidad del código
3. Para borrar una vista se usa drop: db.capturasTotal.drop()

1) Una vista *capturasTotal* que tenga, a partir de la colección capturas, un documento por cada usuario y cada bicho que ha capturado, con el id del usuario (1,2…) , el nombre de cada bicho que ha capturado y una clave *s* con la suma de los puntos de los bichos de ese nombre capturados.

Salida esperada:
```
> db.capturasTotal.find()
{ "_id" : { "user" : 1, "bicho" : "cassandro" }, "total" : 650 }
{ "_id" : { "user" : 1, "bicho" : "chorizard" }, "total" : 125 }
{ "_id" : { "user" : 1, "bicho" : "snoreless" }, "total" : 950 }
{ "_id" : { "user" : 2, "bicho" : "cassandro" }, "total" : 550 }
{ "_id" : { "user" : 2, "bicho" : "plastoise" }, "total" : 1150 }
{ "_id" : { "user" : 2, "bicho" : "snoreless" }, "total" : 1150 }
{ "_id" : { "user" : 3, "bicho" : "belcebuh" }, "total" : 50 }
{ "_id" : { "user" : 3, "bicho" : "cassandro" }, "total" : 750 }
{ "_id" : { "user" : 3, "bicho" : "chorizard" }, "total" : 430 }
{ "_id" : { "user" : 3, "bicho" : "snoreless" }, "total" : 500 }
```

2) Una consulta (**no una vista**), que, apoyándose en la anterior, muestre lo mismo pero solo cuando el total supera los 1000 puntos
Salida esperada:
```
{ "_id" : { "user" : 2, "bicho" : "plastoise" }, "total" : 1150 }
{ "_id" : { "user" : 2, "bicho" : "snoreless" }, "total" : 1150 }
```

3) Una vista *capturasImportantes*, que, no utilice ninguna vista anterior, y muestre para cada usuario solo los ids de usuarios y los bichos con sus totales cuando el total supera los 1000 puntos (es decir igual que 2 pero sin apoyarse en una vista anterior).

Salida esperada:
> db.capturasImportantes.find()
{ "_id" : { "user" : 2, "bicho" : "plastoise" }, "total" : 1150 }
{ "_id" : { "user" : 2, "bicho" : "snoreless" }, "total" : 1150 }

4) Escribir una vista *masPuntos* que muestre el _id del usuario que más puntos totales ha acumulado (el total de puntos es la suma de los puntos de todos sus bichos)

Salida esperada:
> db.masPuntos.find()
{ "_id" : 2, "total" : 2850 }

5) Una vista *bichosPorClase* que a partir de la colección *bichos* muestre para cada *clase* de bicho, el nombre de la clase y un array con el nombre de los bichos de esa clase.

Salida esperada:
> db.bichosPorClase.find()
{ "_id" : "pelma", "bichos" : ["plastoise"] }
{ "_id" : "parrilla", "bichos" : ["chorizard", "belcebuh"] }
{ "_id" : "malote", "bichos" : ["snoreless"] }
{ "_id" : "aire", "bichos" : ["cassandro"] }

6) Una vista *puntosMedio* que indique el promedio de puntos capturados entre todos los usuarios. Para ello: primero se calcula la media de puntos de cada usuario por separado, y luego se hace la media de todos estos valores.

La media debe salir 313.968253968254

7) Una vista *capturasTotalPorNombre* que tenga, para cada usuario y bicho capturado, un documento conteniendo su id (1,2…) , los datos del usuario (nombre etc) , el nombre del bicho capturado y una clave *s* con la suma de los puntos de los bichos de ese nombre capturados. (Es decir, igual que el 1 pero añadiendo el nombre del usuario)

8) Una vista *capturadores* que muestre cada nombre de bicho junto con los datos (nombres, etc) de todas las personas que los han capturado.

9) Utilizar el esquema map-reduce para calcular el problema 1. Copiar en la solución la función *map*, la función *reduce*, y la llamada a *mapreduce*.

10) Algunos bichos pueden “evolucionar” cuando se han capturado tantos que la suma de sus puntos **s** es mayor o igual al valor *pts* (colección bichos). En ese caso la evolución será un nuevo bicho con nombre de la clave “*evolucion*” (colección bichos) y puntos el valor **s**.

Escribir una vista *evoluciona* que contenga por cada bicho de un usuario que puede evolucionar un documento con el _id del usuario, el nombre *bicho* del bicho antes de evolucionar, el nombre *bichoEvolucionado* del bicho tras la evolución, y el número de puntos del bicho evolucionado. (Nota: un bicho solo puede evolucionar una vez, aunque el total de puntos doblara el valor requerido para evolucionar no generaríamos dos, solo uno).