Práctica 4: Agregación

0. Iniciar el servidor (cd; rm -rf datos; mkdir datos; mongod -dbpath datos)
1. Bajar del campus el fichero user.json
2. Desde un terminal de linux teclear mongoimport --db test --collection user --file user.json
   Deben importarse 3000 registros.

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

En esta práctica vamos a utilizar las operación de agregación $group. En particular, queremos consultas para:

1) Contar el número de usuarios verificados y no verificados (clave *verified*). La salida será de la forma:

{ "_id" : "true", "total" : 51 }
{ "_id" : "false", "total" : 2949 }

Solución
db.user.aggregate([{$group:{_id:"$verified",total:{$sum:1}}}])

2) La misma consulta pero en lugar del total de usuarios queremos conocer la media de seguidores (followers) para usuarios verificados y para usuarios no verificados:

{ "_id" : "true", "media" : 385419.07843137253 }
{ "_id" : "false", "media" : 1592.3058663953882 }

Solución
db.user.aggregate([{$group:{_id:"$verified",media:{$avg:"$followers"}}}])

3) Ahora queremos conocer el número medio de seguidores de todos los usuarios en el dataset

Solución
db.user.aggregate([{$group:{_id:null,media:{$avg:"$followers"}}}])

4) Ahora queremos relacionar el número medio de menciones (mentions) agrupados según el número total de tweets originales que ha emitido el usuario (clave *original* de la clave *tweets*), ordenado por número de tweets.

Nota: para ordenar añadir un paso de agregación $sort como segundo paso del pipeline

Debe obtenerse algo como:
{ "_id" : 0, "menciones" : 1.4010791366906474 }
{ "_id" : 1, "menciones" : 1.490909090909091 }
{ "_id" : 2, "menciones" : 0.823045267489712 }
….

Solución

```
db.user.aggregate([{$group:{_id:"$tweets.original",menciones:{$avg:"$mentions"}}},{$sort:{_id:1
}}])
```

5) Queremos saber cuantos usuarios tienen en la clave location un valor distinto de ""
La salida esperada es 1939

Solución

```
db.user.count({location:{$ne:""}})
```

6) Para agrupar, por ejemplo por el año de creación del usuario podemos usar _id:{$year:"$created_at"}. Utilizar esta pista para mostrar un listado ordenado de número de usuarios por año de creación de su cuenta. También debe mostrarse el número medio de tweets totals emitidos por los usuarios con ese año de creación (clave *all* de la clave *tweets*). Debe salir algo del estilo:

```
{ "_id" : 2006, "total" : 1, "media" : 31513 }
{ "_id" : 2007, "total" : 20, "media" : 37829.55 }
{ "_id" : 2008, "total" : 104, "media" : 41137.25961538462 }
….
```

Solución

```
db.user.aggregate([{$group:{_id:{$year:"$created_at"},
                            total:{$sum:1},
                            media:{$avg:"$tweets.all"}}},
                   {$sort:{_id:1}}])
```

7) Refinar la consulta anterior, para que en lugar de año nos aparezca desglosado por año,mes (y ordenado por año, mes). Por ejemplo, para el año 2007 y mes 2 nos saldrán 2 usuarios y una media de 131956 tweets

Solución

```
db.user.aggregate([{$group:{_id:{year:{$year:"$created_at"},month:{$month:"$created_at"}},
                    total:{$sum:1}, media:{$avg:"$tweets.all"}}},{$sort:{_id:1}}])
```

8) [2 puntos] Queremos saber el valor medio, mínimo y máximo de la clave hpref (parte de la clave opinion) para los intervalos de la clave total (dentro de la clave tweets): [0,25), [25,50), [50,100), [100,200), [200, inf]. La salida esperada es:

```
{ "_id" : 0, "media" : 0.07732043632564295 }
{ "_id" : 25, "media" : 0.11719976500577525 }
{ "_id" : 50, "media" : 0.14747790843417374 }
{ "_id" : 100, "media" : 0.15528629782796752 }
{ "_id" : "Other", "media" : 0.18734856940744338 }
```

Curiosidad: La clave hpref de la clave opinion indica la preferencia hacia Donald Trump (hpref>0) o hacia H. Clinton (hpref<0). Esta consulta indica que los twitteros más activos apoyaron más a Trump, o dicho de otra forma que losque hacían campaña por Trump eran más activos.

Solución

```
db.user.aggregate( [
  {
   $bucket: {
     groupBy: "$tweets.total",
     boundaries: [ 0,25,50,100,200 ],
     default: "Other",
     output: {
       "media": { $avg: "$opinion.hpref" }
     }
   }
  }
 ])
```

10) Queremos conocer la media de la expresión opinion.tp/opinion.hp para cada valor de tweets.total, con la condición de que para aquelos documentos en los que opinion.hp==0 el valor opinion.tp/opinion.hp se sustituya por 1. Los valores debemos mostrarse ordenados por el valor tweets.total. La salida esperada:

```
{ "_id" : 4, "media" : 0.7103505843071787 }
{ "_id" : 5, "media" : 0.7470085470085469 }
{ "_id" : 6, "media" : 0.7780934343434343 }
{ "_id" : 7, "media" : 0.771658615136876 }
{ "_id" : 8, "media" : 0.9417818740399386 }
{ "_id" : 9, "media" : 1.2975757575757574 }
…..
```

Solución

```
db.user.aggregate( [
  {
   $group:{_id:"$tweets.total",
         media: {$avg:{$cond: { if: { $gt: [ "$opinion.hp", 0 ] },
                                then: {$divide:["$opinion.tp","$opinion.hp"]},
                                else: 1 }}}
        }
},
 {$sort:{"_id":1} }
 ])
```