Máster Universitario en Big Data y Ciencia de Datos

Actividad 1

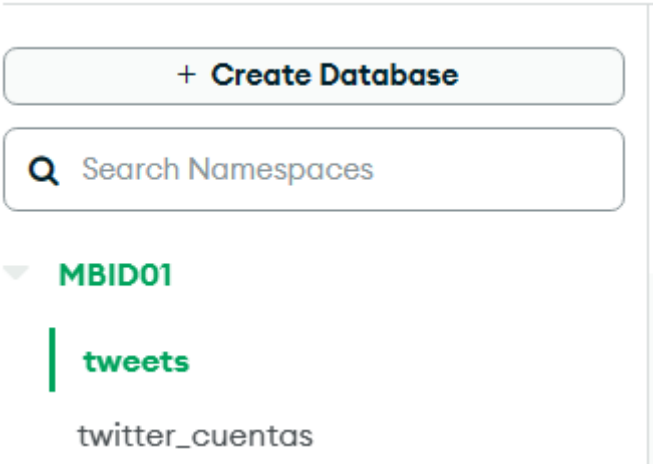
Alumno: Corral Pazos, Manuel José

Edición Octubre 2023 a 06/11/2023

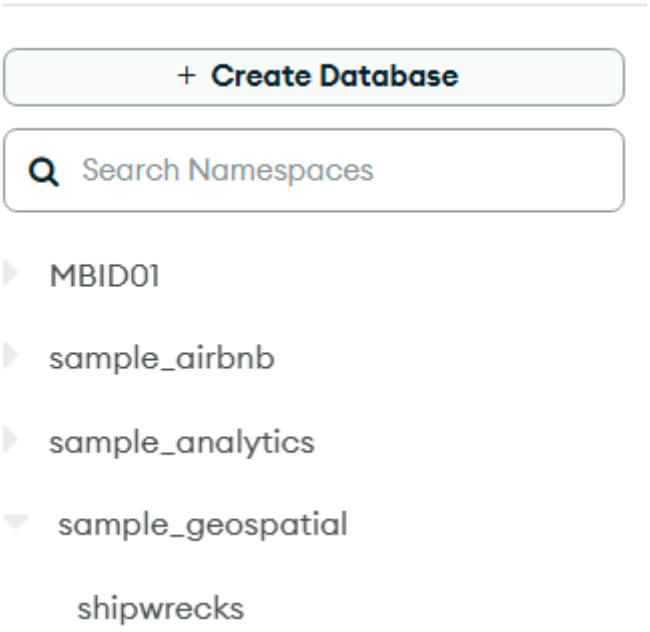
Índice

1. Configuración realizada

En primer lugar, la base de datos creada para realizar esta actividad se llama MBID01. A partir de los datos descargados del aula virtual, importamos los JSON en dos colecciones distintas a las que llamaremos “tweets” y “twitter_cuentas”, siendo el resultado de importar tweets_Actividad_R.json y twitter_Actividad_Cuentas_R.json respectivamente.



Además, se importan las colecciones de ejemplo de MongoDB Atlas para utilizar la “sample_geospatial” en el último chart.



2. Consultas

En esta sección veremos las diferentes consultas propuestas en el apartado 4 de la tarea, estudiando el código relativo a cada una de ellas. En cada una de las consultas, se incluyen los comentarios que las explican. Antes de eso, instalamos pymongo, importamos las librerías necesarias, y realizamos la conexión a la base de datos:

```
pip install pymongo


###### Importación de los paquetes de python necesarios  ######

import json
import pandas as pd
import pymongo

###### Configurar la base de datos Mongo y las colecciones ######

#Establecimiento Conexión a MongoDB Atlas

dbStringConnection = "mongodb+srv://mcorralp:4444333221@mbid01.g65x9ad.mongodb.net/?retryWrites=true&w=majority"


client = pymongo.MongoClient(dbStringConnection)
```

Consulta 1: En la colección de twitters, tener los campos y tweets enviados, cargar los datos correspondientes mediante consulta mongodb + código Python.

```
#Consulta 4.1 -- En la colección de twitters, tener los campos amigos y tweets enviados,
#cargar los datos correspondientes mediante consulta mongodb + código python.

###### Impresión del número de tweets en la base de datos por cuenta. ######

print ("Tweets en la BD MBID01, colección tweets: ", client["MBID01"].tweets.count_documents({}), "\n")

print ("Tweets por cuenta")
for org in client["MBID01"].tweets.aggregate([
    {"$group":{"_id":"$user.screen_name", "sum":{"$sum":1}}}
    ]):
    print ("   ", org['_id'], org['sum'])
    ##Añadimos en la cuenta que tenga de Twitter_handle el mismo id de org, el número de tweets que envió en el campo
    # ya existente number_of_tweets_in_db
    client["MBID01"].twitter_cuentas.update_one({"Twitter_handle":org['_id']}, {"$set" : {"number_of_tweets_in_db":org['sum']}})


print ("Amigos por cuenta")
for org in client["MBID01"].tweets.aggregate([
    {
        '$group': {
            '_id': '$user.screen_name',
            'amigos': {
                '$avg': '$user.friends_count'
            }
        }
    }, {
        '$sort': {
            'amigos': -1
        }
    }
]):
    print ("   ", org['_id'], org['amigos'])
    #Añadimos en la cuenta que tenga de Twitter_handle el mismo id de org, un atributo nuevo llamado "average_number_of_friends"
    #que contenga la media de friends por tweet de esa cuenta
    client["MBID01"].twitter_cuentas.update_one({"Twitter_handle":org['_id']}, {"$set" : {"average_number_of_friends":org['amigos']}})
```

En este caso, se ha decidido tomar la media de “user.friends_count” de los tweets de la cuenta para obtener los amigos de cada una de las cuentas.

Consulta 2: En la colección de tweets, calcular la antigüedad del tweet en función de la fecha actual considerando antigüedad 0 el día de hoy y sumando +1 por cada día transcurrido. Nuevo campo se llamará Frescura.

```
from datetime import datetime

#Tomamos la fecha actual.
ahora = datetime.now()

print ("Insertando frescura de los tweets")
for tweet in client["MBID01"].tweets.find():
    #Formateamos correctamente la fecha que está en la BD, y obtenemos la diferencia entre hoy y dicha fecha.
    #Ésta será la frescura del tweet.
    fechaTweet = datetime.strptime(tweet["created_at"],'%a %b %d %H:%M:%S +0000 %Y')
    dif = ahora - fechaTweet;

    client["MBID01"].tweets.update_one({"_id":tweet["_id"]}, {"$set" : {"frescura": dif.days}})
```

Consulta 3: En la colección de tweets, calcular la antigüedad del tweet relativa con la fecha de creación de la cuenta. Considerando antigüedad 0 si fue enviado el mismo día de creación de la cuenta y sumando +1 por cada día transcurrido desde entonces en función de la fecha del tweet. Nuevo campo se llamará Madurez.

```
from datetime import datetime


print ("Insertando madurez de los tweets")
for tweet in client["MBID01"].tweets.find():
    #Formateamos correctamente la fecha de cuándo fue creado el tweet.
    fechaTweet = datetime.strptime(tweet["created_at"],'%a %b %d %H:%M:%S +0000 %Y')
    #Formateamos correctamente la fecha de cuándo se creó el usuario asociado al tweet.
    fechaCuenta = datetime.strptime(tweet["user"]["created_at"],'%a %b %d %H:%M:%S +0000 %Y')
    #Obtenemos la diferencia entre ambas fechas, y esta será la madurez del tweet.
    dif = fechaTweet - fechaCuenta;

    client["MBID01"].tweets.update_one({"_id":tweet["_id"]}, {"$set" : {"madurez": dif.days}})
```

Se entrega junto con este documento un archivo de Python con los códigos aquí mostrados.

3. Charts

En este apartado obtendremos los charts pedidos en el punto 5 de la actividad tras haber hecho las consultas e inserciones del apartado anterior. Así, campos como el número de amigos, la frescura o la madurez de los tweets ya están insertados en la base de datos.

Chart 1: Cuenta de Twitter con mayor cantidad de tweets enviados.

En este caso, utilizamos la colección de cuentas de twitter y el tipo de chart Top Item, pues solo queremos obtener la cuenta que más tweets ha enviado. Para ello, ordenamos por el campo “number_of_tweets”, y ponemos como Display el campo de “Twitter_handle”, para obtener el nombre de la cuenta con mayor número de tweets. En este caso, hay varias cuentas empatadas en el top con 1000 tweets, pero sólo mostramos una.

Chart 2: Cuenta de twitter con mayor cantidad de amigos

En este apartado, calculamos en número de amigos de una cuenta como realizamos en la consulta 1 del apartado anterior. Utilizamos la misma colección y gráfico que en el chart anterior, ordenando por “average_number_of_friends”.

Chart 3: Número total de tweets

Utilizando la colección de tweets, realizamos una agregación sobre el "_id" de los tweets, realizando una cuenta de cuántos ids diferentes hay. De esta forma, con el tipo de chart Number, ya obtenemos el número total de tweets.

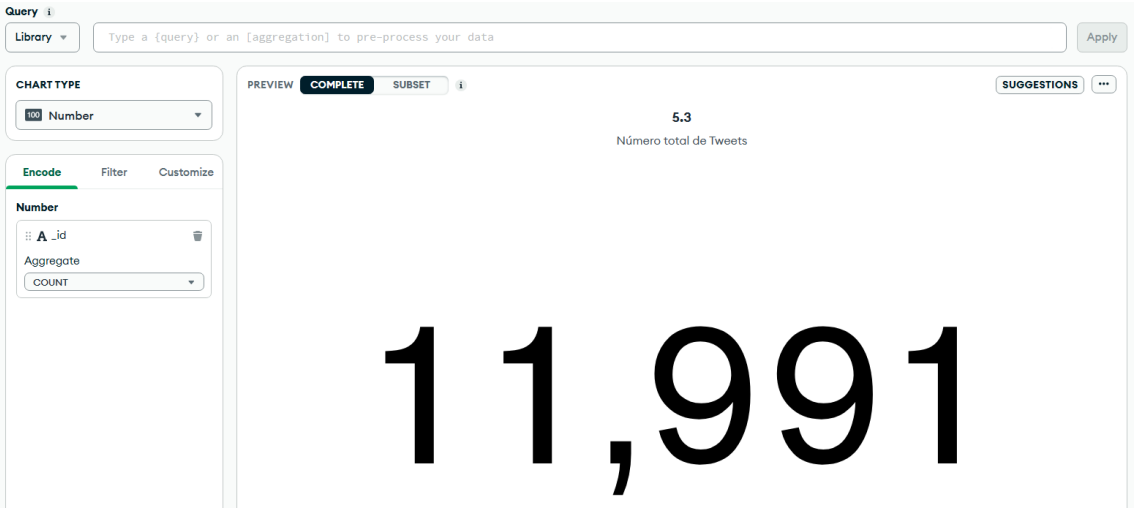


Chart 4: Número total de tweets por cuenta

Para este apartado, veo ilustrativo utilizar una tabla que nos indique el número de tweets por cada cuenta de Twitter. Para ello, agrupamos por nombre de la cuenta, y agregamos los "_id" de los tweets que tienen como "user.name" a esa cuenta.

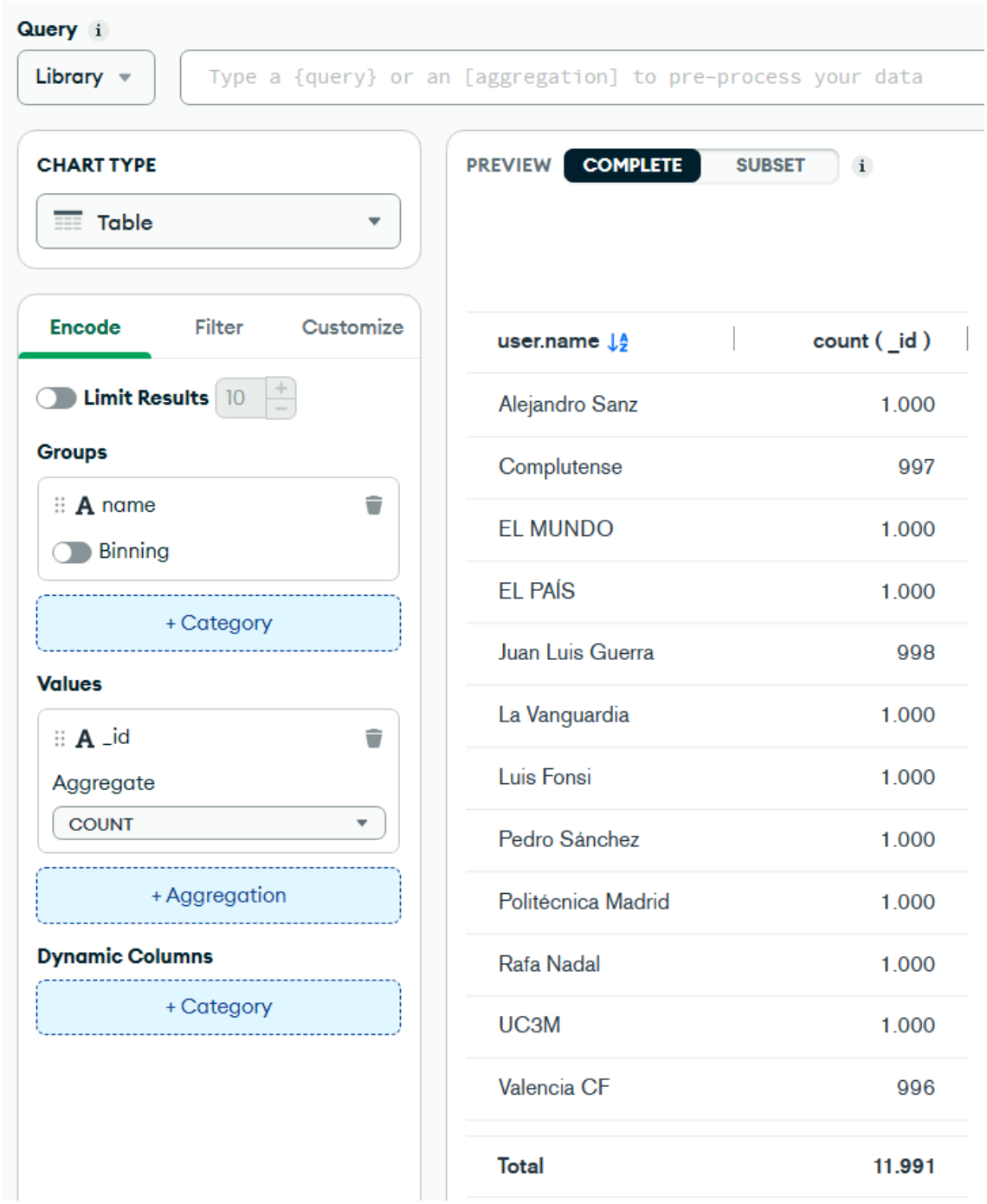


Chart 5: Cuentas de Twitter versus amigos ordenados de mayor a menor

Utilizamos la colección de twitter_cuentas y Colored Column como tipo de gráfico. De esta forma, ponemos Twitter_handle en el eje X para ver las diferentes cuentas de twitter, y average_number_of_friends en el eje Y para ver el número de amigos de cada una. El eje X lo tenemos ordenado por valor, y de esta forma obtenemos lo pedido en el enunciado.

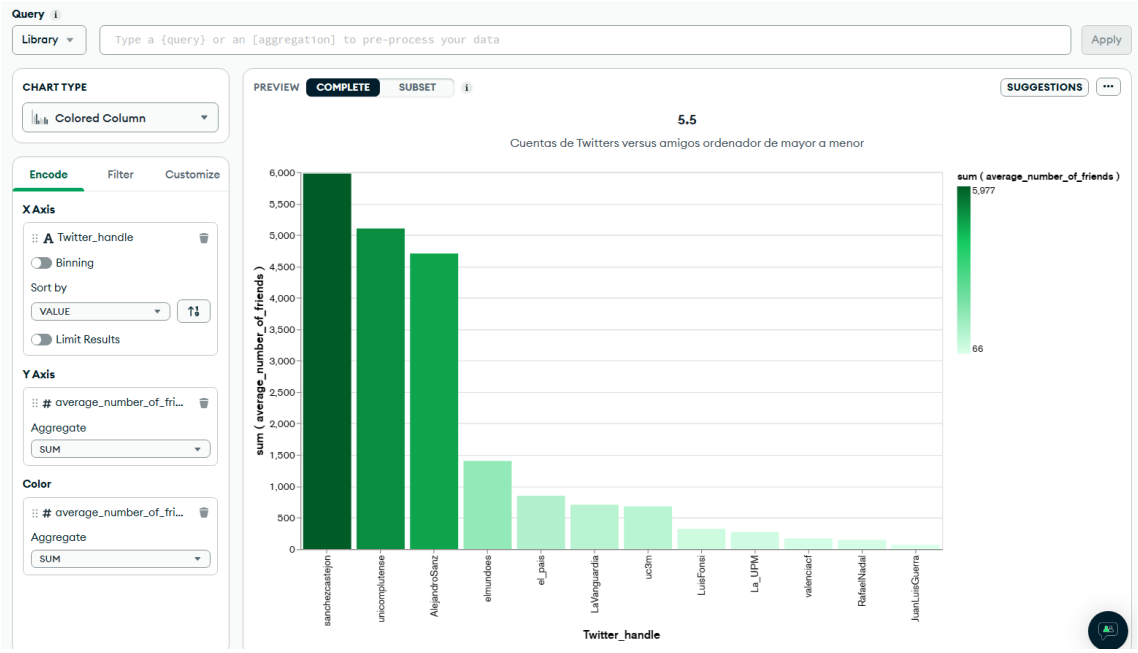


Chart 6: Cuentas de Twitter versus tweets enviados ordenado de mayor a menor

En este apartado, utilizamos la colección de cuentas de twitter. Al igual que en el apartado anterior, utilizamos Colored Column pero esta vez con number_of_tweets_in_db en el eje Y. en este caso, como los valores son muy próximos entre ellos, establezco el mínimo del eje Y en 970 para poder ver la diferencia de tweets entre las diferentes cuentas.

Chart 7: Tweets versus día semana

Utilizamos la colección de tweets y gráfico de Colored Column. En el eje X utilizamos el campo de created_at, haciendo conversión a tipo fecha y activando el Binning por Día de la semana. En el eje Y, agregamos por _id, y obtenemos la gráfica pedida.

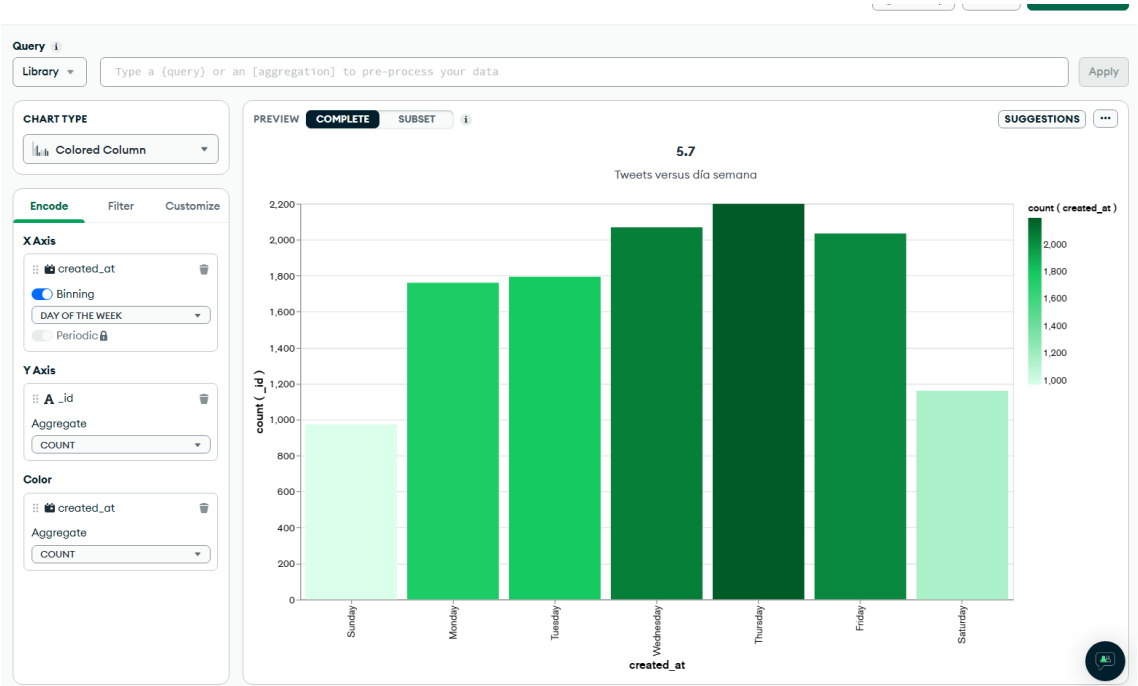


Chart 8: Ordenar los Tweets según su Madurez.

Para este apartado, utilizo un tipo de gráfico Stacked Combo, para poder ver las diferencias entre los tweets. Como se intenta visualizar la madurez para cada uno de los tweets, habiendo más de diez mil en la base de datos, es imposible visualizar correctamente el valor de cada uno de los tweets. De hecho, en Charts se indica que lo máximo es 5000, y se ha sobrepasado dicho valor. Aunque utilicemos otro tipo de gráficos, la visualización tampoco es posible para tanta cantidad de datos a mostrar.

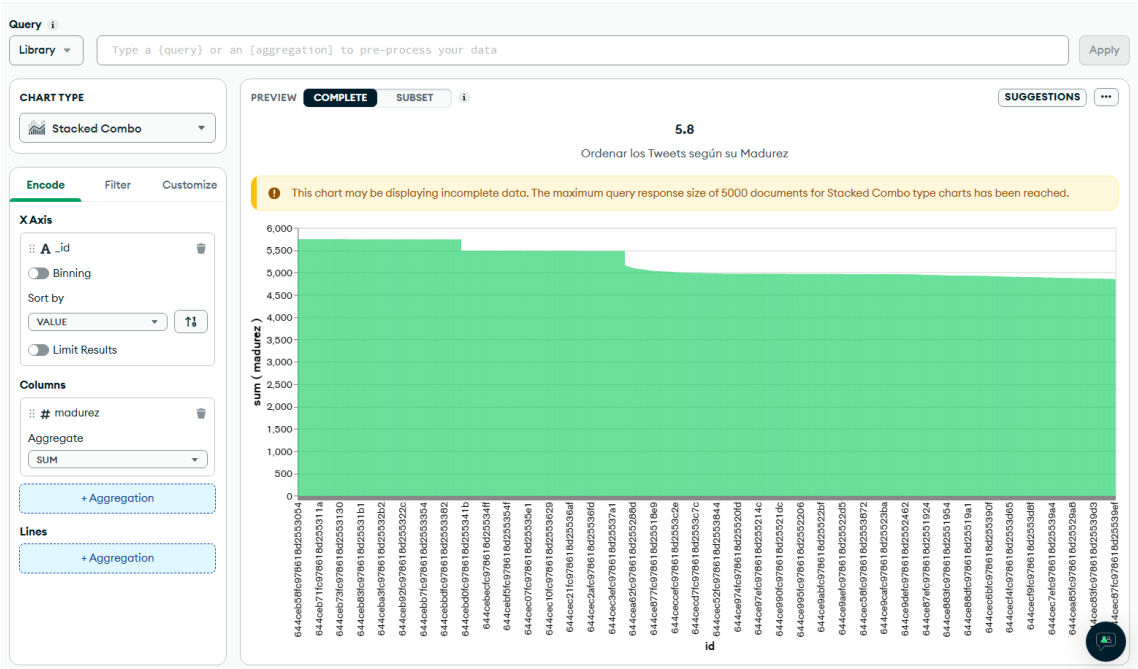


Chart 8.2: Cuenta más madura.

Utilizando la colección de tweets, empleamos un gráfico Colored Bar, agregando por madurez en el eje X y utilizando el campo user.name del tweet, ordenando por la media de madurez. De esta forma, obtenemos las cuentas ordenadas por la media de madurez de los tweets de la cuenta de mayor a menor, y la cuenta más madura es EL PAÍS.

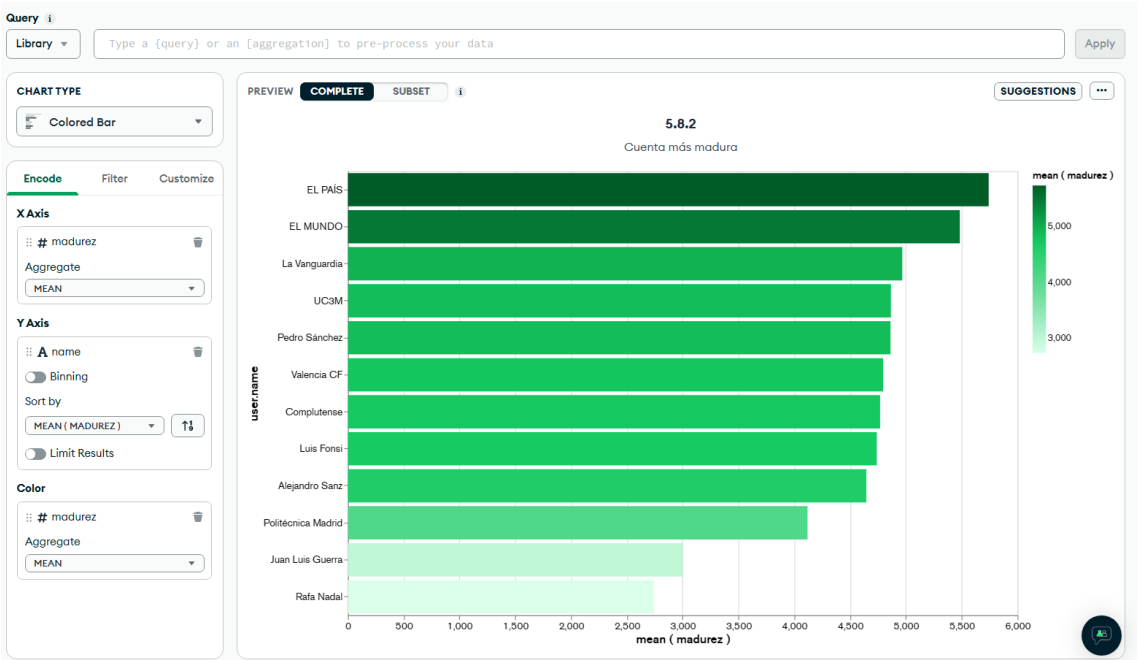


Chart 9: Ordenar los Tweets según su Frescura

Al igual que en el apartado anterior, tenemos la misma limitación en cuanto a la representación de demasiados valores en un chart. Misma configuración que antes, pero utilizando la frescura del tweet.

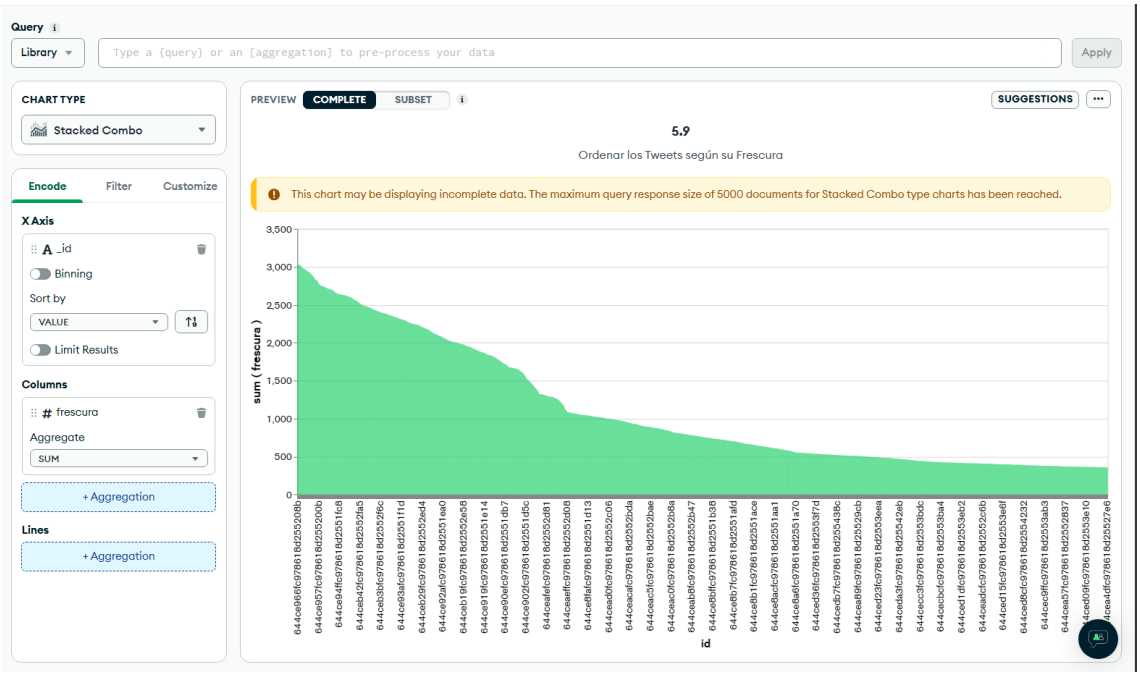


Chart 9.2:

Misma configuración que en el gráfico 8.2, pero utilizando la media de frescura de los tweets. En este caso, la cuenta más fresca es Juan Luis Guerra.

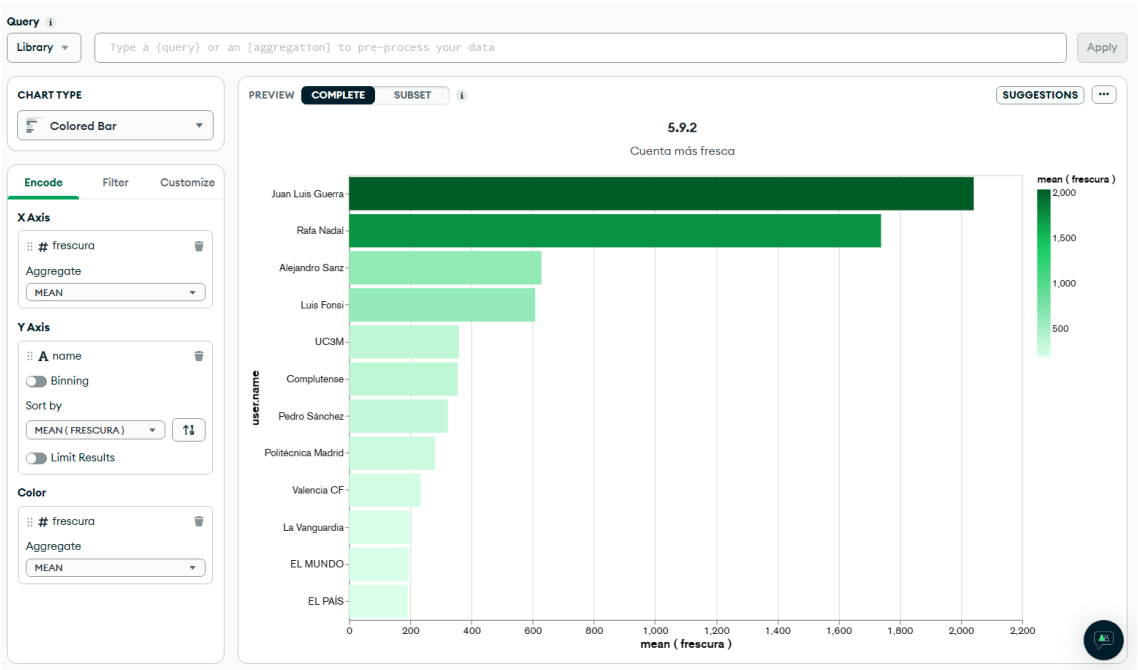
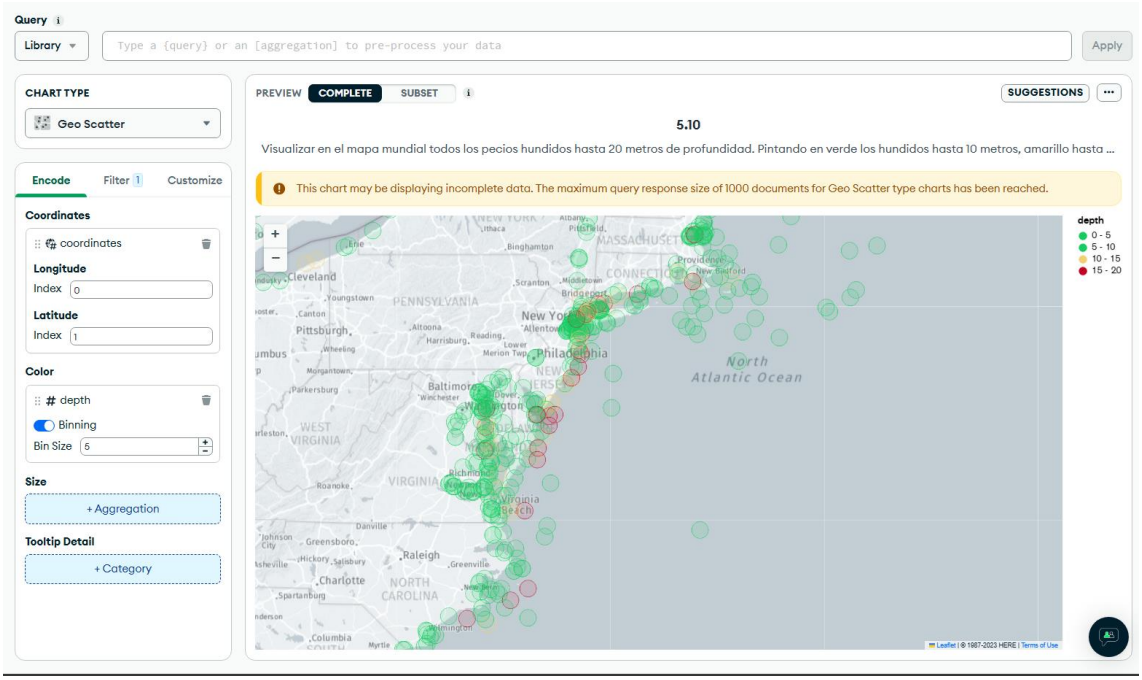
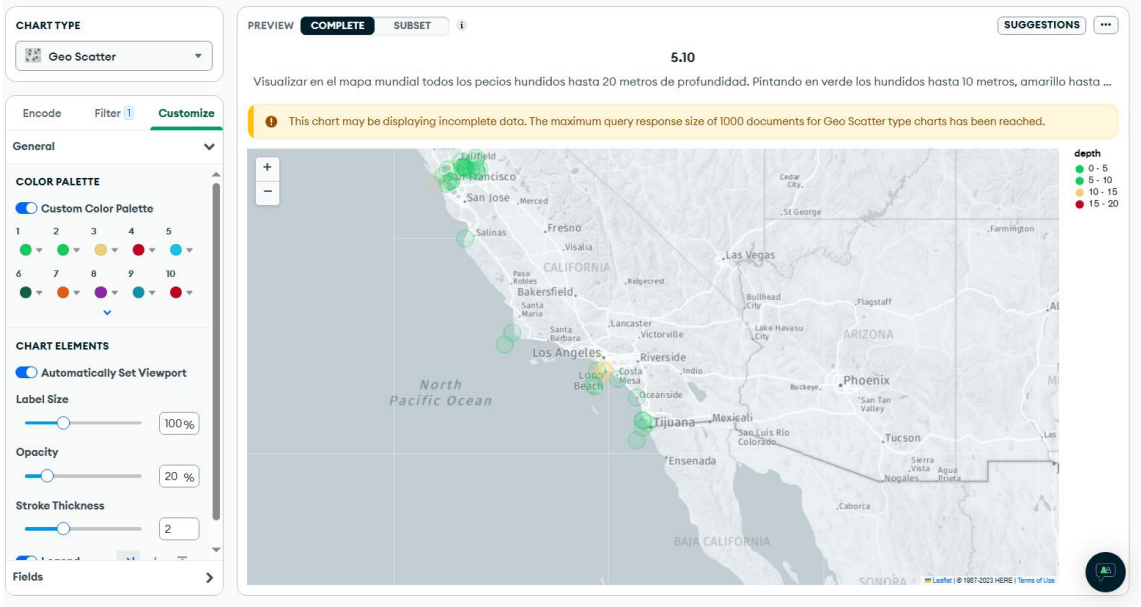
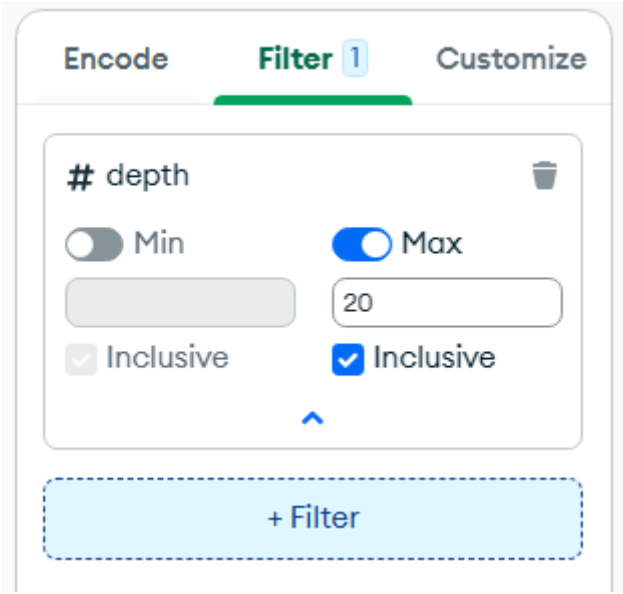


Chart 10: Visualizar en el mapa mundial todos los pecios hundidos hasta 20 metros de profundidad.

Para el último chart, utilizamos la colección shipwrecks del sample_geospatial de MongoDB Atlas. Utilizamos el tipo de chart Geo Scatter, indicando las coordenadas de los datos, y coloreando en función de su profundidad, “Depth”, poniendo como máximo 20 en el campo de filtros. También se indica un Binning con tamaño 5, y de esta forma se pintan los valores como se pide en el enunciado.






Encode
Filter 1
Customize
depth
Min
Max
20
Inclusive
Inclusive
+ Filter