Loche Rémy ESI5E

Compte rendu

Objectif : L'objectif du projet est de créer un pipeline streaming avec pour finalité de créer des graphes pour afficher les données recueillies.

Afin de réaliser ce pipeline, nous avons utilisé une stack composé de 4 technologies à savoir Apache Spark (Avec Spark streaming et Spark SQL), Apache Kafka, Une base de données MySQL et Apache superset pour la visualisation.

L'objectif du projet est donc de recueillir des données en temps réel via l'API de twitter, les envoyer sur le topic kafka que l'on va créer afin que notre Spark streaming récupère les données du topic, les traite et les insère dans la base de donnée MySQL pour que Superset puisse les lire et afficher le ou les graphes que l'on créera par la suite.

Nous avons commencé par installer les machines virtuelles une pour Spark, une pour Kafka, une pour exécuter nos scripts python qui produit la donnée et une pour la base de donnée et superset. Nous avons réalisé celà à l'aide de VirtualBox et Vagrant. (Le script utilisé dans le vagrantfile génère la configuration pour nous à savoir la gestion des ports, l'installation de Python, Jupyter, Spark, MySQL, Superset et Java pour la machine spark et Kafka.

Une fois les machines créées nous nous sommes connecté aux machines en utilisant la commande vagrant ssh.

Sur la machine Kafka, le broker et le zookeeper étant déjà instanciés grâce au fichier start.sh dans la configuration du Vagrantfile. Nous avons donc simplement créé le topic qui nous servira à connecter le producer et le consumer puis envoyer quelques données de test.

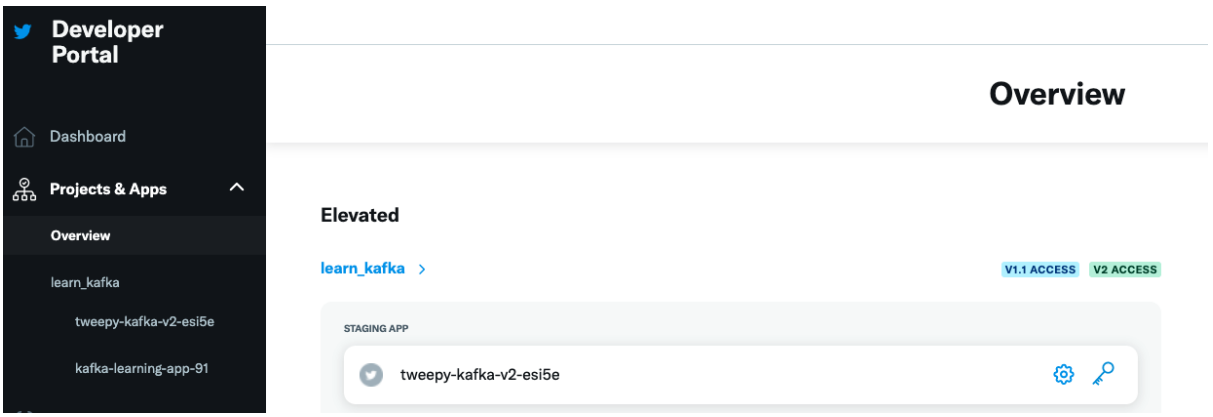
```
vagrant@vagrant:~/kafka/bin$ ./kafka-topics.sh --create --topic tweets-topic --bootstrap-server 192.168.33.13:9092
Created topic tweets-topic.
```

On vérifie qu'il est bien créé

```
vagrant@vagrant:~/kafka/bin$ ./kafka-topics.sh --list --bootstrap-server 192.168.33.13:9092
tweets-topic
```

Nous avons créé le script permettant de produire de la donnée dans kafka en récupérant les tweets de l'API Twitter

L'idée d'origine était de récupérer la géolocalisation des tweets pour étudier dans quels pays ils étaient le plus envoyés etc…
Pour cela je me suis d'abord intéressé à l'objet "location" du tweet. Mais cet élément peut être inséré par les utilisateurs librement et certains ne l'utilisent pas pour donner leur ville ou leur pays. Ensuite j'ai essayé avec l'objet "place" mais mon compte étant au rang "elevated"



Je n'ai pas le niveau le plus élevé et ne peux pas avoir ces informations.

J'ai donc décidé d'étudier uniquement les hashtags des le nombre de followers des utilisateurs.

Pour ce faire, nous avons installé kafka-python sur la machine puis nous avons écrit ce script.

python > vm > producer-streaming-twitter.py > ...

```python
from tweepy.streaming import Stream
import time
from kafka import KafkaProducer
import tweepy
import json

API_KEY = "mwqLxJ64vVtWtoy8lO8XHnwsj"
API_KEY_SECRET = "1B5xRMLMEapdFChbMUMTWZpK17Sp9J5usv4rObZrjjqXWjIthV"
BEARER_TOKEN = "AAAAAAAAAAAAAAAAAAAAAHK4bwEAAAAA3jfvvGdvNiRZjn%2By4H4tgcGUBlg%3Dnu30ECI0qb5Io7CeYTD5DHBuy98l77WkjAquBA4oGxxrHi7ifd"
ACCESS_TOKEN = "898851040451321857-swnGgmKR2BKyV6rkivheg4jJjpa4Aws"
ACCESS_TOKEN_SECRET = "7V5DkO9GR4Sy80webTtZCgGW1tpDE1UU5fereCu20xu2X"

producer = KafkaProducer(bootstrap_servers='192.168.33.13:9092')


class Listener(Stream):

    tweetsProceeded = []
    limit = 1000

    def on_data(self, raw_data):
        self.process_data(raw_data)
        return True

    def process_data(self, raw_data):
        if len(self.tweetsProceeded) == self.limit:
            print("disconnecting ... ")
            self.disconnect()
        else:
            message = json.loads(raw_data)
            if message['user']['location'] is not None:
                print(message)
                producer.send('tweets-topic', raw_data)
                self.tweetsProceeded.append(raw_data)

    def on_error(self, status_code):
        if status_code == 420:
            # returning false in on_data disconnects the stream
            return False
```

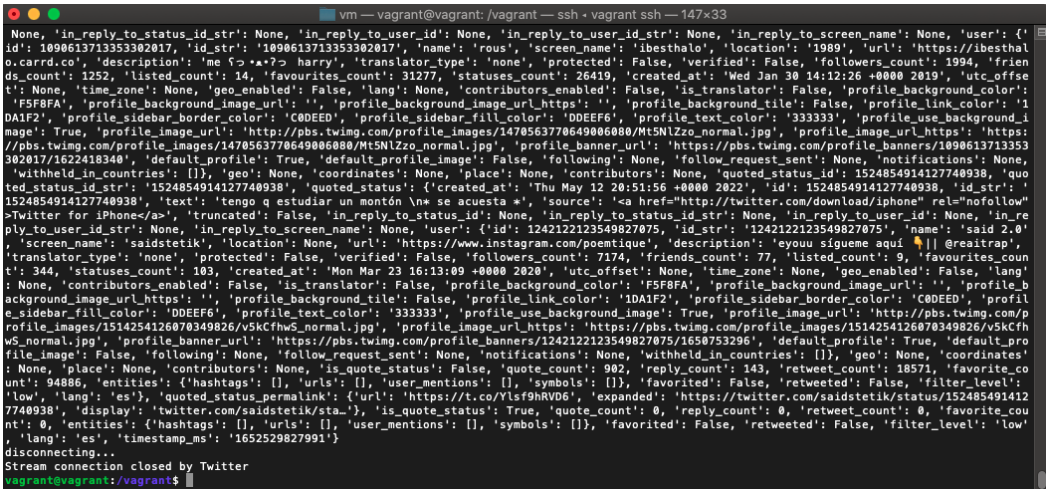
Loche Rémy ESI5E

```
41
42  # start the stream
43  if __name__ == "__main__":
44      auth = tweepy.OAuthHandler(API_KEY, API_KEY_SECRET)
45      auth.set_access_token(ACCESS_TOKEN, ACCESS_TOKEN_SECRET)
46
47      api = tweepy.API(auth)
48
49      listener = Listener(API_KEY, API_KEY_SECRET,
50                          ACCESS_TOKEN, ACCESS_TOKEN_SECRET)
51
52      keywords = ["*"]
53
54      listener.filter(track=keywords)
55
```

Comme on peut le voir au moment ou nous lançons le script, les tweets sont affiché sur la console

```
vagrant@vagrant:/vagrant$ python3 producer-streaming-twitter.py
{'created_at': 'Wed May 04 16:28:31 +0000 2022', 'id': 1521889520261627904, 'id_str': '1521889520261627904', 'text': '@Suzanne_Smith From the first testament *',
 'display_text_range': [15, 41], 'source': '<a href="http://twitter.com/download/iphone" rel="nofollow">Twitter for iPhone</a>', 'truncated': False, 'in_reply_to
_status_id': 1521871374695817216, 'in_reply_to_status_id_str': '1521871374695817216', 'in_reply_to_user_id': 22652933, 'in_reply_to_user_id_str': '22652933', 'in
_reply_to_screen_name': 'NDrumark', 'user': {'id': 22652933, 'id_str': '22652933', 'name': 'Doc Tash Denmark', 'screen_name': 'NDrumark', 'location': 'UT: 33.431
212,-111.903974', 'url': 'http://culturedconvos.com', 'description': 'EdD, ATC, CPRW (she/her) #educator #AT #SocialJustice and #Equity Warrior \u2640\ufe0f#mother #wife a
nd #activesoul \u2695\ufe0f\u200d\u2640\ufe0f Belarus \u27a1\ufe0fNew Jersey \u27a1\ufe0fCalifornia', 'translator_type': 'none', 'protected': False, 'verified': False, 'followers_count': 698, 'friends
_count': 811, 'listed_count': 11, 'favourites_count': 21957, 'statuses_count': 6904, 'created_at': 'Tue Mar 03 18:04:04 +0000 2009', 'utc_offset': None, 'time_zo
ne': None, 'geo_enabled': True, 'lang': None, 'contributors_enabled': False, 'is_translator': False, 'profile_background_color': 'BADFCD', 'profile_background_im
age_url': 'http://abs.twimg.com/images/themes/theme12/bg.gif', 'profile_background_image_url_https': 'https://abs.twimg.com/images/themes/theme12/bg.gif', 'profi
le_background_tile': False, 'profile_link_color': 'FF0000', 'profile_sidebar_border_color': 'F2E195', 'profile_sidebar_fill_color': 'FFF7CC', 'profile_text_color
': '0C3E53', 'profile_use_background_image': True, 'profile_image_url': 'http://pbs.twimg.com/profile_images/1478436621120794625/JUzNHND4_normal.jpg', 'profile_i
mage_url_https': 'https://pbs.twimg.com/profile_images/1478436621120794625/JUzNHND4_normal.jpg', 'default_profile': False, 'default_profile_image': False, 'follo
wing': None, 'follow_request_sent': None, 'notifications': None, 'withheld_in_countries': []}, 'geo': None, 'coordinates': None, 'place': None, 'contributors': N
one, 'is_quote_status': False, 'quote_count': 0, 'reply_count': 0, 'retweet_count': 0, 'favorite_count': 0, 'entities': {'hashtags': [], 'urls': [], 'user_mentio
ns': [{'screen_name': 'Suzanne_Smith', 'name': 'Suzanne', 'id': 19152377, 'id_str': '19152377', 'indices': [0, 14]}], 'symbols': []}, 'favorited': False, 'retwee
ted': False, 'filter_level': 'low', 'lang': 'en', 'timestamp_ms': '1651681711627'}
```

une fois que le nombre de message à été produit, le stream se stop :



En lançant un kafka-console-consumer nous voyons bien les tweets passer aussi

```
vagrant@vagrant:~/kafka/bin$ ./kafka-console-consumer.sh --bootstrap-server 192.168.33.13:9092 --from-beginning --topic tweets-topic
{"created_at":"Wed May 04 16:28:31 +0000 2022","id":1521889520261627904,"id_str":"1521889520261627904","text":"@Suzanne_Smith From the firs
t testament *","display_text_range":[15,41],"source":"\u003ca href=\"http:\/\/twitter.com\/download\/iphone\" rel=\"nofollow\"\u003eTwitter
 for iPhone\u003c\/a\u003e","truncated":false,"in_reply_to_status_id":1521871374695817216,"in_reply_to_status_id_str":"1521871374695817216"
,"in_reply_to_user_id":22652933,"in_reply_to_user_id_str":"22652933","in_reply_to_screen_name":"NDrumark","user":{"id":22652933,"id_str":"2
2652933","name":"Doc Tash Denmark","screen_name":"NDrumark","location":"\u00dcT: 33.431212,-111.903974","url":"http:\/\/culturedconvos.com"
,"description":"EdD, ATC, CPRW (she\/her) #educator #AT #SocialJustice and #Equity Warrior \u2721\ufe0f #mother #wife and #activesoul \ud83
c\udfc3\ud83c\udffb\u200d\u2640\ufe0f Belarus \u27a1\ufe0f New Jersey \u27a1\ufe0f California","translator_type":"none","protected":false,"
verified":false,"followers_count":698,"friends_count":811,"listed_count":11,"favourites_count":21957,"statuses_count":6904,"created_at":"Tu
e Mar 03 18:04:04 +0000 2009","utc_offset":null,"time_zone":null,"geo_enabled":true,"lang":null,"contributors_enabled":false,"is_translator
":false,"profile_background_color":"BADFCD","profile_background_image_url":"http:\/\/abs.twimg.com\/images\/themes\/theme12\/bg.gif","profi
le_background_image_url_https":"https:\/\/abs.twimg.com\/images\/themes\/theme12\/bg.gif","profile_background_tile":false,"profile_link_col
or":"FF0000","profile_sidebar_border_color":"F2E195","profile_sidebar_fill_color":"FFF7CC","profile_text_color":"0C3E53","profile_use_backg
round_image":true,"profile_image_url":"http:\/\/pbs.twimg.com\/profile_images\/1478436621120794625\/JUzNHND4_normal.jpg","profile_image_url
_https":"https:\/\/pbs.twimg.com\/profile_images\/1478436621120794625\/JUzNHND4_normal.jpg","default_profile":false,"default_profile_image"
:false,"following":null,"follow_request_sent":null,"notifications":null,"withheld_in_countries":[]},"geo":null,"coordinates":null,"place":n
ull,"contributors":null,"is_quote_status":false,"quote_count":0,"reply_count":0,"retweet_count":0,"favorite_count":0,"entities":{"hashtags"
:[],"urls":[],"user_mentions":[{"screen_name":"Suzanne_Smith","name":"Suzanne","id":19152377,"id_str":"19152377","indices":[0,14]}],"symbol
s":[]},"favorited":false,"retweeted":false,"filter_level":"low","lang":"en","timestamp_ms":"1651681711627"}
```

Puis nous avons créé le job Spark qui va consommer les données du topic

Loche Rémy ESI5E

```
import string
from pyspark.sql import SparkSession, Row
from pyspark.streaming import StreamingContext
from pyspark.streaming.kafka import KafkaUtils
from pyspark.sql.types import FloatType, StringType
from pyspark.sql.functions import to_date
import json
import re

emoji_pattern = re.compile("["
        u"\U0001F600-\U0001F64F"  # emoticons
        u"\U0001F300-\U0001F5FF"  # symbols & pictographs
        u"\U0001F680-\U0001F6FF"  # transport & map symbols
        u"\U0001F1E0-\U0001F1FF"  # flags (iOS)
        u"\U00002702-\U000027B0"
        u"\U000024C2-\U0001F251"
        u"\U0001f926-\U0001f937"
        u'\U00010000-\U0010ffff'
        u"\u200d"
        u"\u2640-\u2642"
        u"\u2600-\u2B55"
        u"\u23cf"
        u"\u23e9"
        u"\u231a"
        u"\u3030"
        u"\ufe0f"
                           "]+", flags=re.UNICODE)

def getHashtags(rdd_collected):
    list_hashtags = []
    for element in rdd_collected:
        print(element)
        for hashtag in element:
            if hashtag["text"] != None:
                list_hashtags.append(hashtag["text"])
    return list_hashtags
```

```
def stripTextAndRemoveEmoji(text):
    stringWithoutEmoji = emoji_pattern.sub(r'', text)
    nameIsBlank = re.search('^\s*$', stringWithoutEmoji)
    if not stringWithoutEmoji or nameIsBlank :
        return "No name"
    return stringWithoutEmoji.strip()
```

```

def process(time, rdd):
    print("========= %s =========" % str(time))
    if not rdd.isEmpty():
        print(rdd)
        locationDF = rdd.map(lambda tweet: Row(username=stripTextAndRemoveEmoji(tweet['user']['name']),
                           nb_friends=tweet['user']['friends_count'])).toDF()

        locationDF = locationDF.withColumn("username", locationDF["username"].cast(StringType())) \
                           .withColumn("nb_friends", locationDF["nb_friends"].cast(IntegerType()))

        locationDF.printSchema()

        locationDF.write.format('jdbc').options(
            url='jdbc:mysql://192.168.33.10/data',
            dbtable='users',
            user='admin',
            password='admin').mode('append').save()


        rdd_collected = rdd.map(lambda tweet: tweet["entities"]["hashtags"]).collect()

        if len(rdd_collected) >= 1:
            hashtags_list = getHashtags(rdd_collected)
            rdd_hashtags = sc.parallelize(hashtags_list)
            if rdd_hashtags.isEmpty() == False:
                hashtagsDF = rdd_hashtags.map(lambda hashtagg: Row(hashtag=hashtagg)).toDF()
                hashtagsDF = hashtagsDF.withColumn("hashtag", hashtagsDF["hashtag"].cast(StringType()))
                hashtagsDF.printSchema()
                hashtagsDF.write.format('jdbc').options(
                    url='jdbc:mysql://192.168.33.10/data',
                    dbtable='hashtags',
                    user='admin',
                    password='admin').mode('append').save()


spark = SparkSession.builder \
        .master("local[2]") \
        .appName("data") \
        .getOrCreate()

```

```
sc = spark.sparkContext
sc.setLogLevel("ERROR")
ssc = StreamingContext(sc, 10)

directKafkaStream = KafkaUtils.createDirectStream(ssc, ["tweets"], {"metadata.broker.list": "192.168.33.13:9092"})
rdd = directKafkaStream.map(lambda tweet: json.loads(tweet[1]))
rdd.foreachRDD(process)


ssc.start()
ssc.awaitTermination()
```

ensuite, sur la machine spark, nous avons installé spark-streaming-kafka : wget https://repo1.maven.org/maven2/org/apache/spark/spark-streaming-kafka-0-8-assembly_2.11/2.4.7/spark-streaming-kafka-0-8-assembly_2.11-2.4.7.jar

Puis déplacé ce jar dans les jars de spark sudo mv spark-streaming-kafka-0-8-assembly_2.11-2.4.7.jar /usr/local/spark/jars

Récupération des données sur le consumer spark en lançant le job avec la commande spark-submit

```
vagrant@vagrant:~$ /usr/local/spark/bin/spark-submit spark-consumer-tweets.py
```

Log du consumer spark : J'ai affiché les hashtags et les schémas des Dataframes pour vérifier que le job fonctionnait correctement.

```
========= 2022-05-14 10:25:30 =========
PythonRDD[2247] at RDD at PythonRDD.scala:53
root
 |-- location: string (nullable = true)
 |-- username: string (nullable = true)

[{'text': 'ปาฐ', 'indices': [132, 137]}]
[]
[{'text': 'Davido', 'indices': [91, 98]}]
[]
root
 |-- hashtag: string (nullable = true)
```

J'ai ensuite vérifier que les données s'insèrent bien dans la base de données.
Sur la machine MySQL / Superset, en accédant à la base de donnée avec les commandes

"sudo mysql"
"use data;"
"select * from users;"

```
1083 rows in set (0.01 sec)
```

et pour les hashtags

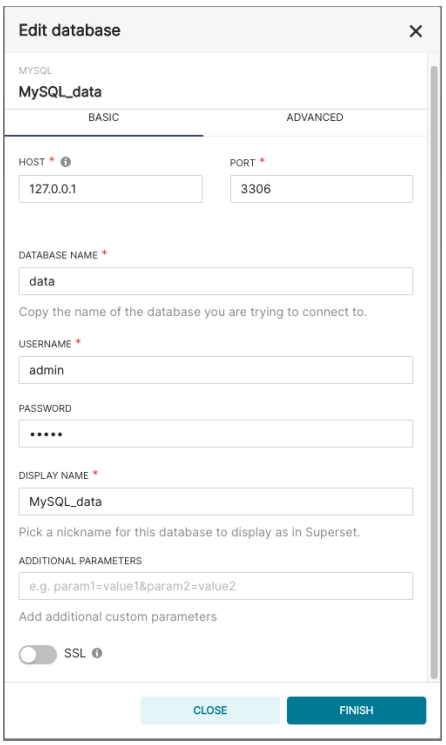
"select * from hashtags;"

```
401 rows in set (0.00 sec)
```

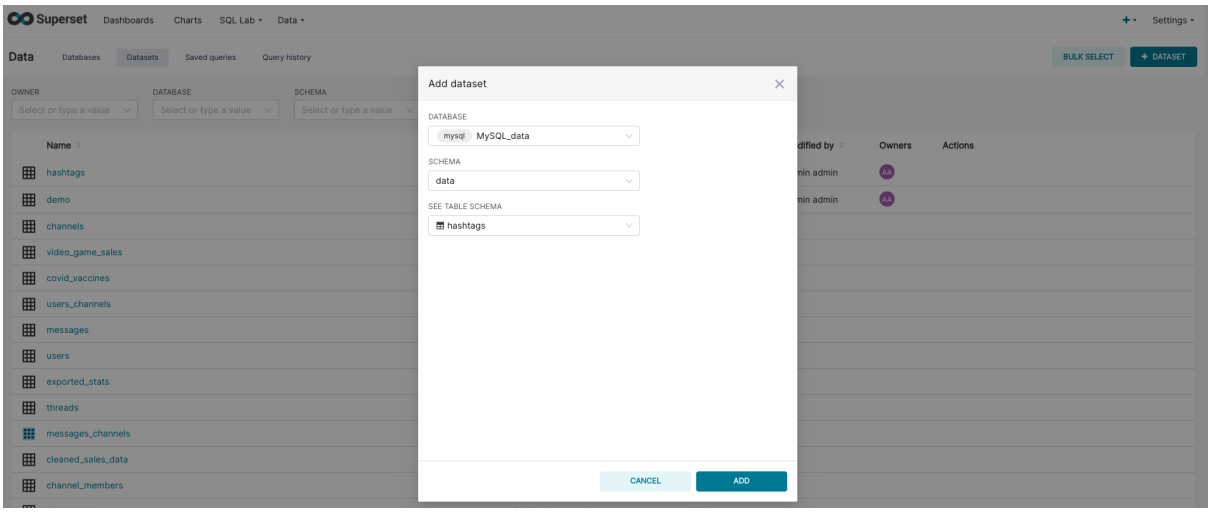
Nous allons maintenant connecter notre base de données sur superset pour pouvoir effectuer des actions sur les données.

On ajoute donc notre base de donnée :

Loche Rémy ESI5E

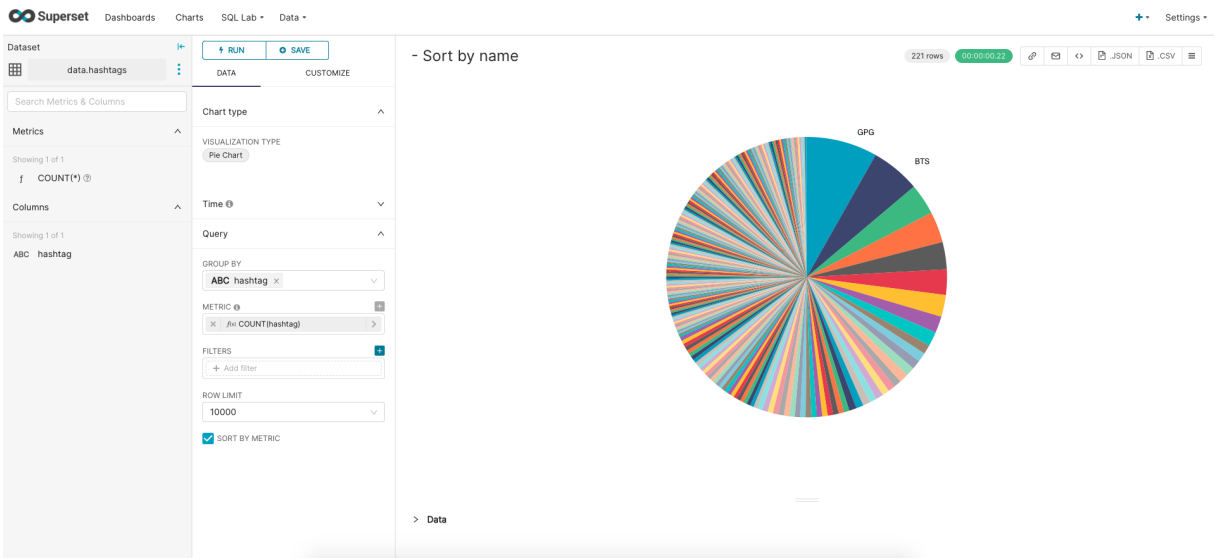


Création du dataset à partir de la table de notre base de données

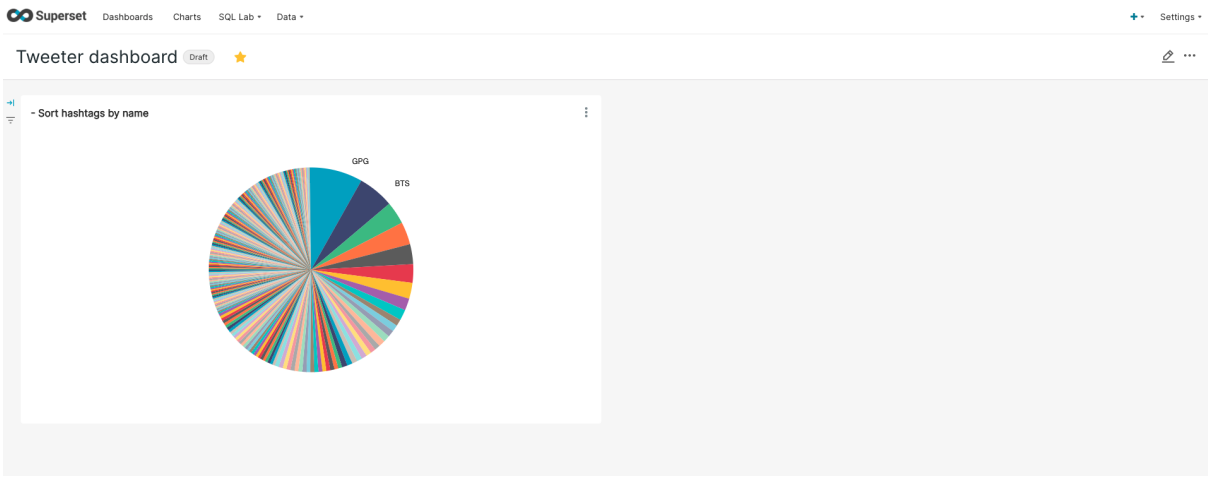


Création d’un chart en fonction du count des hashtag présents dans la base de données

Loche Rémy ESI5E



Ajout du chart à un nouveau dashboard “Tweeter dashboard”



J’ai donc créé différents graph représentant sous différentes formes les utilisateurs et le nombre d’amis ainsi que les hashtags les plus utilisés au moment où j'ai réalisé l’exercice.