

# **PROGRAMA DE CIENCIAS DE LOS DATOS**

**Curso: Big Data**

**PROYECTO FINAL:**

**Streaming de tendencias con #hashtags en Twitter.**

**Profesor: MSc. Felipe Meza Obando**

**Alumnos:**

**Lester Salazar Viales.**

**Randal Salazar Viales.**

## Arquitectura

Para este proyecto final se va a utilizar una arquitectura Cliente – Servidor la cual consiste en:

- Se crea un socket el cual va a estar conectado al api de Twitter y que además estará extrayendo los mensajes de Twitter.
- El socket también creará un canal para que se conecte la aplicación de spark que se encargará de hacer los filtros del twitter.
- Se crea una aplicación cliente para que consulte al socket que extrae los twitter, esto con el fin de realizar el filtrado de los hashtag y revisar cuales son las tendencias en dicha red social.

## Librerías para Twitter que vamos a utilizar con Conda

Debido a la utilización de Conda para el desarrollo de la aplicación para realizar streaming se decidió que se iba a requerir de la librería Tweepy, por lo que se tuvo que instalar en el ambiente de trabajo.

## Instalación de Tweepy en Conda

Para instalar Tweepy debemos de ir a la siguiente dirección web:

<https://anaconda.org/conda-forge/tweepy>

Donde se nos mostrará una página similar a la siguiente:

The screenshot shows the Conda Forge package page for 'tweepy' version 3.8.0. The header includes the package name and version, along with navigation icons (home, code, docs, star) and a badge count of 18. Below the header, there's a description: 'An easy-to-use Python library for accessing the Twitter API'. A tabbed interface shows 'Conda' as the selected tab, with other tabs for 'Files', 'Labels', and 'Badges'. The main content area lists package details: License (MIT), Home (http://www.tweepy.org), Development (https://github.com/tweepy/tweepy), Documentation (http://tweepy.readthedocs.io), 84775 total downloads, and Last upload (6 months and 24 days ago).

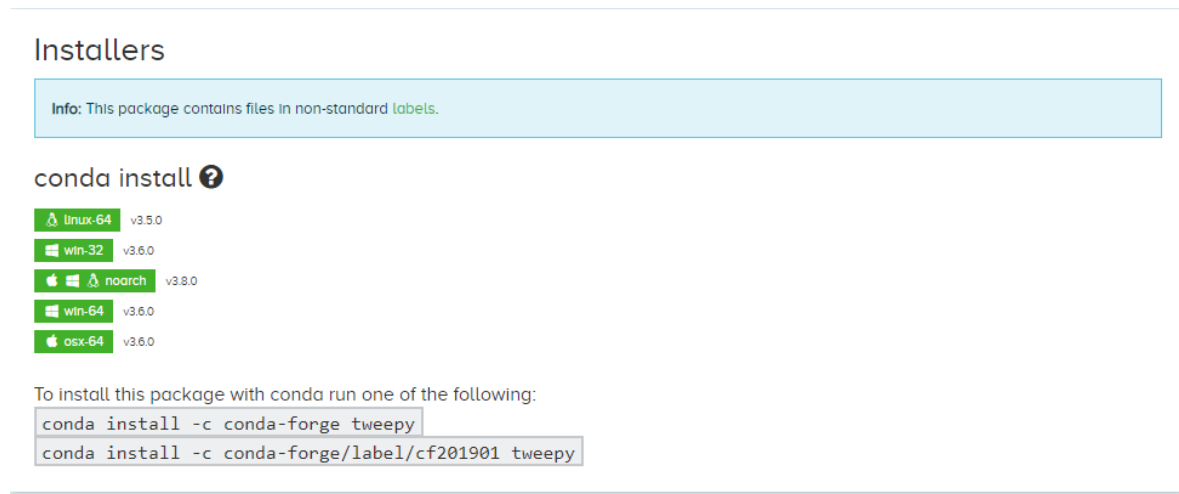
conda-forge / packages / tweepy 3.8.0

An easy-to-use Python library for accessing the Twitter API

Conda	Files	Labels	Badges
<p>License: MIT</p> <p>Home: <a href="http://www.tweepy.org">http://www.tweepy.org</a></p> <p>Development: <a href="https://github.com/tweepy/tweepy">https://github.com/tweepy/tweepy</a></p> <p>Documentation: <a href="http://tweepy.readthedocs.io">http://tweepy.readthedocs.io</a></p> <p>84775 total downloads</p> <p>Last upload: 6 months and 24 days ago</p>			

En dicha página buscamos el apartado donde se encuentran los pasos para instalar la librería tweepy.

Ahí se nos indican los comandos que se pueden utilizar para instalar tweepy en Conda.



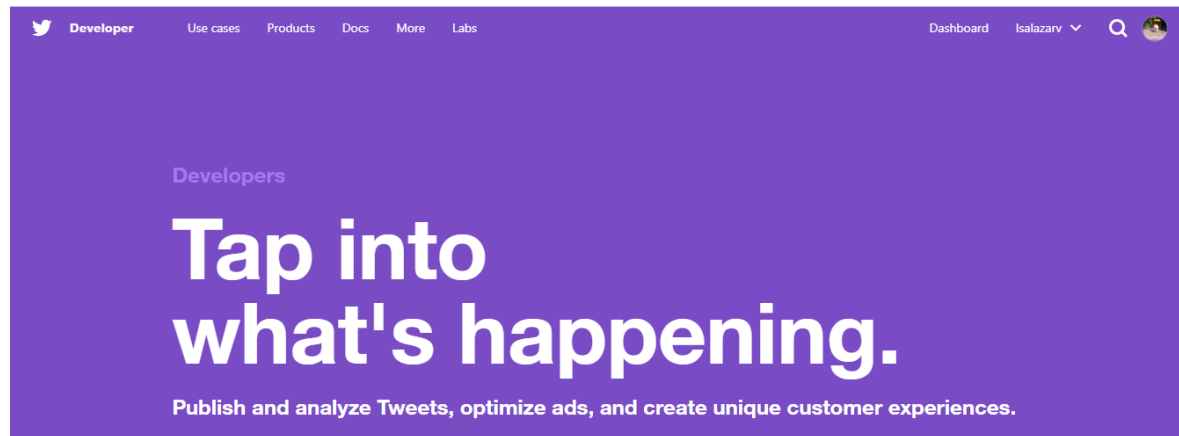
The screenshot shows the 'Installers' section for the 'tweepy' package on the Conda Forge website. At the top, there is a light blue box with the text: 'Info: This package contains files in non-standard labels.' Below this, the command 'conda install ?' is displayed. A list of operating systems and architectures is shown, each with a corresponding icon and version number: 'linux-64 v3.5.0', 'win-32 v3.6.0', 'noarch v3.8.0', 'win-64 v3.6.0', and 'osx-64 v3.6.0'. Below the list, a text prompt says: 'To install this package with conda run one of the following:'. Two code blocks are provided: 'conda install -c conda-forge tweepy' and 'conda install -c conda-forge/label/cf201901 tweepy'.

## Api Twitter

Para el caso de nuestro proyecto decidimos utilizar el API de Twitter. Por lo que decimos ingresar al sitio para developers de Twitter en la siguiente dirección web.

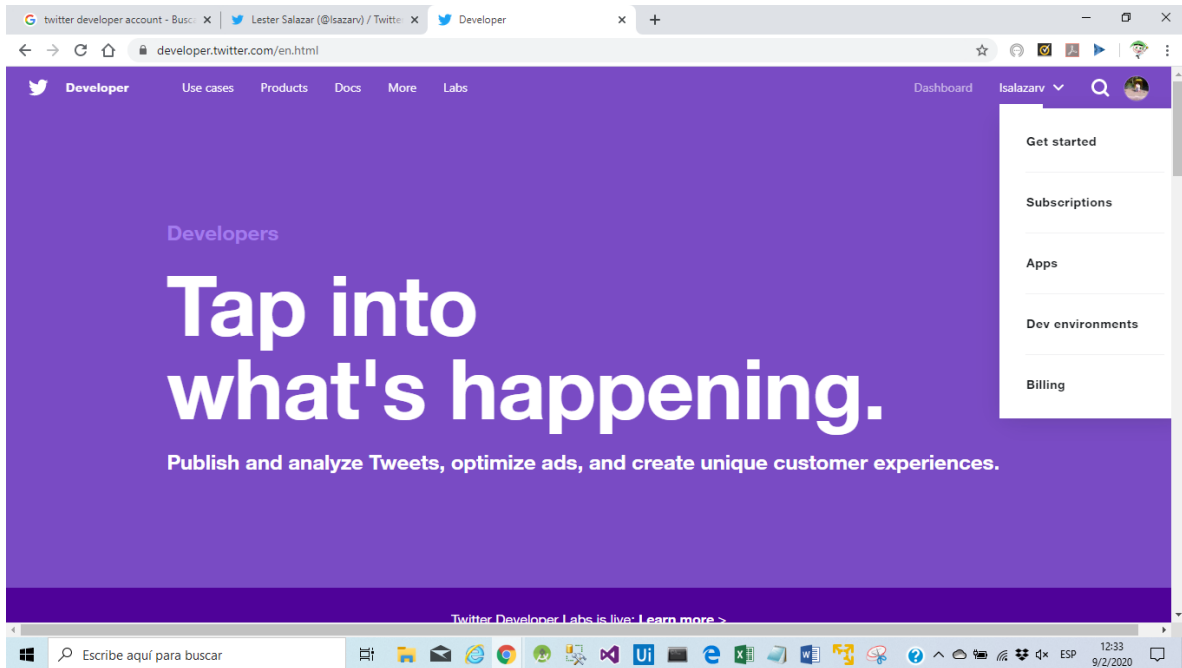
<https://developer.twitter.com/>

Lo que nos lleva a una página similar a la siguiente.



The screenshot shows the Twitter Developer website. The header is purple with the Twitter logo, 'Developer' text, and navigation links: 'Use cases', 'Products', 'Docs', 'More', and 'Labs'. On the right, there are links for 'Dashboard', 'Isalazary', a dropdown arrow, a search icon, and a user profile picture. The main content area has a purple background with the text 'Developers' in small white letters, followed by 'Tap into what's happening.' in large white letters. At the bottom, it says 'Publish and analyze Tweets, optimize ads, and create unique customer experiences.'

Una vez en dicha pantalla hacemos click sobre el nombre de usuario, lo que nos despliega un menú como el que se en la imagen, en dicho menú seleccionamos la opción de App para crear la app que vamos a utilizar del API.



Una vez presionado la opción de App nos aparecerá una pantalla similar a la siguiente donde se nos da la opción para crear la app que vamos a utilizar.



Se presiona el botón que dice “Create an app” que nos lleva a la pantalla donde procederemos a crear nuestra app de twitter.

Apps > [Create an app](#)

**Understanding apps**

**What is an app?**

Think of a Twitter app as a gateway. It provides a set of authentication keys and permission settings needed for making requests to most Twitter APIs.

**Why register an app?**

**Which products require an API key?**

**App details**

The following app details will be visible to app users and are required to generate the API keys needed to authenticate Twitter developer products.

**App name (required)** ⓘ

Maximum characters: **32**

**Application description (required)**

Share a description of your app. This description will be visible to users so this is a good place to tell them what your app does.

Please be detailed.

**Organization website URL**

**Tell us how this app will be used** (required)

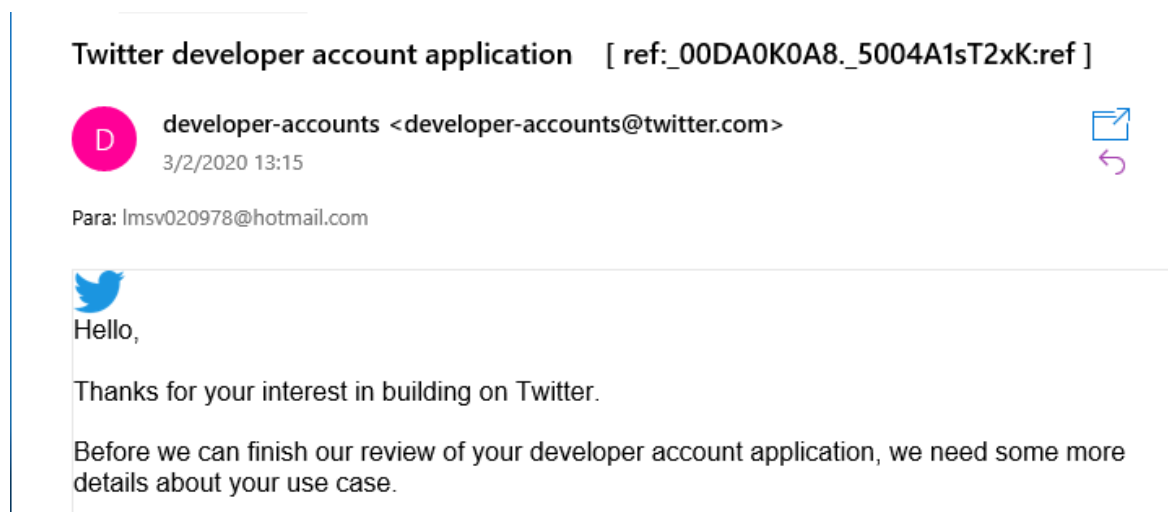
This field is only visible to Twitter employees. Help us understand how your app will be used. What will it enable you and your customers to do?

Please be detailed.

Minimum characters: **100**

En dicha pantalla se llenan los datos tales como el nombre de la app y el uso que se le va a dar. Una vez que los datos han sido ingresados se procede a presionar el botón que dice “Create”, lo que envía un correo a la cuenta que se utilizó para crear la cuenta de Twitter.

El correo es muy similar al de la imagen siguiente, en caso de que requieran más explicación del uso de la cuenta de desarrollo, se deben de responder unas preguntas que envían de la empresa.



Una vez que las preguntas son respondidas y validadas por el personal de twitter nos envían un correo donde nos indican que la cuenta fue aprobada y que ya puede ser utilizada.

## Account Application Approved



Twitter Developer Accounts <developer-accounts@twitter.com>

3/2/2020 16:26



Para: Lester Salazar



Developer

Account Application Approved

## Your Twitter developer account application has been approved!

Thanks for applying for access. We've completed our review of your application, and are excited to share that your request has been approved.

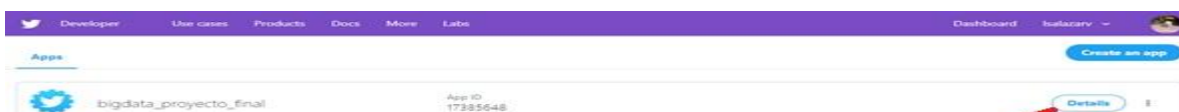
Sign in to your [developer account](#) to get started.

Thanks for building on Twitter!

### Generación Token/Keys

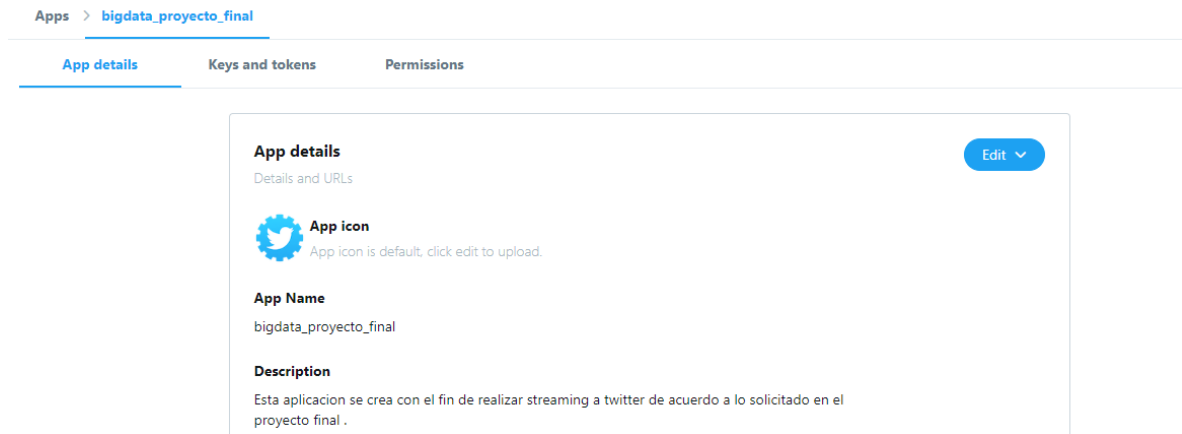
Una vez creada la aplicación se procede a crear los tokens y los keys que se van a utilizar en la aplicación básica que va a realizar streaming de twitter.

Para generar los tokens presionamos el botón que dice "Details" en el app que creamos recientemente

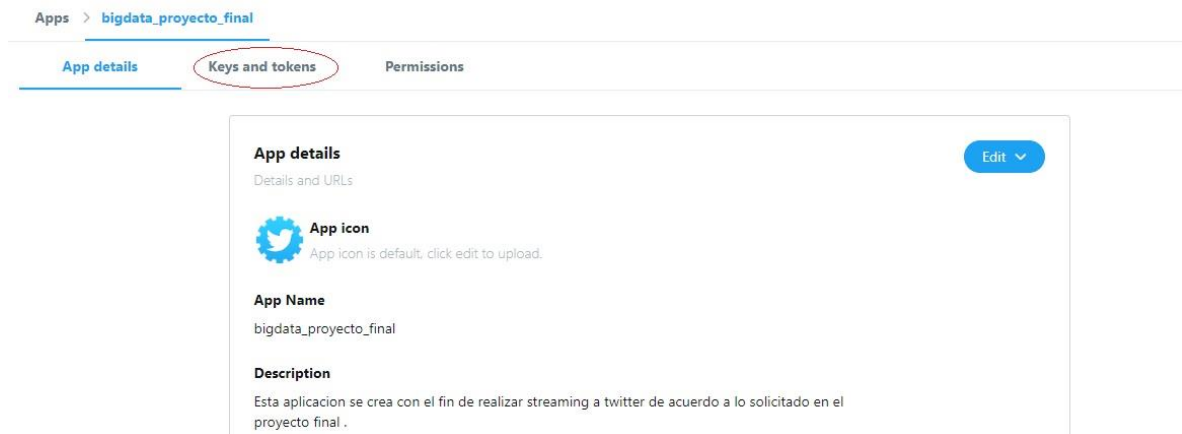


*Botón para ingresar al detalle del App para generar los token y keys*

Cuando se presiona dicho botón nos lleva a la siguiente pantalla similar a la que se muestra en la siguiente imagen



En la pantalla de detalle de la app seleccionamos el tab que dice “Keys and tokens” para generar los tokens y los keys que necesita nuestra aplicación para poder conectarse al api de twitter.



Una vez en la pantalla de Keys and tokens procedemos a generar los consumer API keys y los Access token que va a ocupar nuestra aplicación que va a estar realizando streaming a Twitter por medio de la Api.

Looking for your secret token? For security, API tokens are only displayed once. You will need to regenerate access tokens for previously authenticated apps. To learn more,

Keys, secret keys and access tokens management.

## Regenerate

**API secret key:** wBiDmKBKAd9ZTiE7P4PdMF49r0yoGIsFuXRJIW69OGAuZMyJKC

Revoke

*We only show your access token and secret when you first generate it in order to make your account more secure. You can revoke or regenerate them at any time, which will invalidate your existing tokens.*

## Consumer API keys

Regenerate

**API secret key:** wBiDmKBKAd9ZTiE7P4PdMF49r0yoGlsFuXRJIW69OGAuZMyJKC

### Access token & access token secret

Revoke

Regenerate



## Creación del SOCKET.

Para poder crear el socket primero debemos de crear una notebook en Jupyter y luego procedemos a importar las librerías a utilizar.

```
import os
import tweepy
from tweepy import OAuthHandler
from tweepy import Stream
from tweepy.streaming import StreamListener
import socket
import json
```

Una vez importado las librerías se asignan los valores de acceso de los token para la api de twitter

```
CONSUMER_KEY='gXokQgEW6dv8fSSnKTvvxL2qg'
CONSUMER_SECRET='a2jOnasE5sgnh2UfVDY1izwHuzoaOPiJGwCVFBXnLdnfkqZi7n'
ACCESS_TOKEN = '1224205344257335297-BY4usKicNvYTEPjA6uBPiGlumA32Mf'
ACCESS_TOKEN_SECRET='RdzJYaqFF2FWS6Zmnd1Jr818v35r6uTFbLxQwChKHHDY4'

auth = tweepy.OAuthHandler(CONSUMER_KEY, CONSUMER_SECRET)
auth.set_access_token(ACCESS_TOKEN, ACCESS_TOKEN_SECRET)
```

Una vez asignados los valores se crea una clase para escuchar los mensajes de Twitter, la clase es similar a la siguiente

```
class MyStreamListener(tweepy.StreamListener):

    def __init__(self, csocket):
        self.client_socket = csocket

    def on_data(self, raw_data):
        try:
            # Variable con los datos completos obtenidos desde TWITTER
            tweets = json.loads(raw_data)

            # Definición de campos del mensaje de TWITTER a enviar para Streaming
            if 'extended_tweet' not in tweets:
                text = tweets['text'].strip()
            else:
                text = tweets['extended_tweet']['full_text'].strip()

            outputMsg = {
                'created_at': tweets['created_at'],
                'text': tweets['text'],
                'userid': tweets['user']['id'],
                'username': tweets['user']['name'],
                'userlocation': tweets['user']['location'],
                'retweet_count': tweets['retweet_count'],
                'entities': [x['text'] for x in tweets['entities']['hashtags']]
            }
```

Después de crear la clase que escucha los mensajes de twitter se procede a crear el socket que va a esperar la comunicación de la aplicación que se crea con pyspark.

```
import socket
import time

# Conexión de
#Initializing the port and host
host = 'localhost'
port = 5555
address = (host, port)

#Initializing the socket
server_socket = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
server_socket.bind(address)
server_socket.listen(5)

print(server_socket)
print("Listening for client...")

conn, address = server_socket.accept()
print("Connected to Client at " + str(address))

## Conexión con API Twitter para poder "escuchar" streams de los tweets
myStreamListener = MyStreamListener(csocket=conn)
myStream = tweepy.Stream(auth=auth, listener=myStreamListener, )

# Hashtag a buscar en los tweets:
tracklist = ["#"]
myStream.filter(track=tracklist)
```

## Creación aplicación Spark

Para crear la aplicación spark primero se van a importar las librerías

```
# Importación de Librería PySpark
import pyspark
from pyspark import SparkFiles

# Otras Librerías PySpark
import pyspark.sql.functions as F
from pyspark.sql.types import *

from pyspark.sql.functions import col, date_format, udf
from pyspark.sql.functions import explode, split, size, from_json
from pyspark.sql.types import DateType

# Librerías del Sistema Operativo
import os

# Librerías Numéricas
import pandas as pd
import numpy as np
from datetime import datetime

# Librerías de Preprocesamiento de datos Twitter
import preprocessor as p
import json
```

Una vez importadas las librerías se procede a importar el findspark para poder inicializar spark

```
import findspark

# Ruta de Apache Spark para sistema Mac/OS
findspark.init('/opt/spark')

# Ruta de Apache Spark para sistema Windows/OS
findspark.init('C:\Spark')
```

Luego creamos el SparkSession

```
from pyspark.sql import SparkSession
from pyspark.sql.functions import explode, split, size

spark = SparkSession.builder.appName("TwitterSpark").getOrCreate()
```

Una vez creado el SparkSession procedemos a crear la variable esquema para almacenar los valores

```
# Definir schema para el input data
schema = StructType([ \
    StructField("created_at", StringType()),\
    StructField("text", StringType()),\
    StructField("userid", StringType()),\
    StructField("username", StringType()),\
    StructField("userlocation", StringType()),\
    StructField("retweet_count", IntegerType()),\
    StructField("entities", ArrayType(StringType())),\
])
```

Se procede a crear el streamdataframe para almacenar los valores que se van a recibir de twitter

```
lines = spark.readStream.format("socket").option("host", "localhost").option("port", 5555).load()
```

Una vez creado el dataframe procedemos a visualizar los datos

```
# Identificación de si el DataFrame tiene datos Streaming o NO
lines.isStreaming

True
```

Realizado el punto anterior creamos el streamdataframe basado en el schema creado anteriormente.

```
df_twitter = lines.select(
    from_json('value', schema).created_at.alias('created_at'),
    from_json('value', schema).text.alias('text'),
    from_json('value', schema).userid.alias('userid'),
    from_json('value', schema).username.alias('username'),
    from_json('value', schema).userlocation.alias('userlocation'),
    from_json('value', schema).retweet_count.alias('retweet_count'),
    from_json('value', schema).entities.alias('entities')
)
```

Se crea un arreglo para almacenar el streamdataframe de los datos de twitter

```
exp_entities = df_twitter.withColumn('hashtag', explode(df_twitter.entities))
```

Visualizamos el dataframe definido anteriormente

```
# Identificación de si el DataFrame tiene datos Streaming o NO
exp_entities.isStreaming
```

Procedemos a ordenar los hashtags repetidos

```
total_tweets = exp_entities.select('hashtag', 'retweet_count', 'userlocation', 'userid') \
    .groupBy('hashtag') \
    .agg({'hashtag': 'count', 'retweet_count': 'sum'}) \
    .sort('count(hashtag)', ascending = False)
```

Una vez ordenados los datos procedemos a crear el query

```
query = total_tweets.writeStream.queryName('query_hashtag').outputMode('complete').format('memory').start()
```

Luego realizamos un describe del dataframe

```
spark.sql("Describe query_hashtag").show()
```

Para visualizar los datos procesados en el dataframe utilizamos la siguiente instrucción

```
spark.sql("select count(*) from query_hashtag ").show()
```