VIT®

Vellore Institute of Technology

(Deemed to be University under section 3 of UGC Act, 1956)

STREAMING TWITTER DATA INTO MySQL DATABASE

PROJECT REPORT

GROUP MEMBERS:

1. **SATWIK TANWAR (20BCE2945)**
2. **ABHAY NIMMAGADDA (20BCE2327)**
3. **SHIVA RAMAKRISHNA Y (20BCE2864)**
4. **DHYANA SAI N (20BCE2956)**
5. **VARANASI PHANI BALA MOUNIKA (20BCE2963)**

SUBMITTED TO: Prof. Jyotismita Chaki

COURSE NAME: Database Management Systems

COURSE CODE: CSE2004

SLOT: L57+L58

ABSTRACT

It’s undeniable that social media has now become an extremely important platform to aid businesses to expand their online reach. Marketers have gradually become increasingly conscious of the relevance of social media platforms such as Facebook, Twitter, and LinkedIn in business growth via word-of-mouth marketing over the years. The main aim of our project is to stream twitter data into MySQL database. We can stream data directly into a MySQL database using Python and an API. We'll use the Twitter API to look for tweets and users we are interested in and import them directly into our database. The data will be available for further examination at any moment once we've completed this.

Table of Contents:

1. INTRODUCTION:

1.1. How did this project evolve?

If we recollect the purpose of social media, we can say that it is to bring people together from all over the world. It's understandable why businesses have rushed to social media, since they've realised it's a terrific location for communication, whether it's good news, terrible news, fake news, or an international disaster. As a result, it would be preferable if we could store data from Twitter in a MySQL database.

1.2. Why is this helpful?

We'll use the Twitter API to get data about the followers of an account and feed it directly into our database. This data can be used to analyse their audience and increase their reach. We will store data like the username of the person who posted the tweet, the time it was posted, the tweet, the number of retweets it received, number of likes and so on. After that, the data will be available for analysis at any time. Companies will benefit greatly from this since they will be able to see how people are reacting.

2. BACKGROUND

2.1. Previous Works

a) MOA-TweetReader: Real-Time Analysis in Twitter Streaming Data

- Author: Albert Bifet, Geoffery Holmes, and Bernhard Pfahringer University Of Waikato, Hamilton, New Zealand
- Tweets are preprocesssed and converted MOA-TweetReader to vectors of attributes or machine learning instances. The second component of the system is a learner trained with several instances, that is able to

predict the class label of incoming unlabeled instances. Finally, a change detector monitors the predictions, and outputs an alarm signal when change is detected.

- Application: detecting changes in term frequencies and performing real time sentiment analysis.

b) Automating Corpora Generation with Semantic Cleaning and Tagging of Tweets for Multi-dimensional Social Media Analytics

- Author: Nazura Javed Research Scholar Bangalore University Bangalore, India and Muralidhara B.L. Bangalore University Bangalore, India.
- The 'processTweet' algorithm which automates the process of collecting and cleaning the tweets was coded using Python.
- Unprocessed as well as the cleaned, processed tweets were stored in MySQL database.
- PyEnchant, a spell checking library for Python was used for checking the spelling of English words and for splitting the joined words.
- Jellyfish, a Python library, was used for doing approximate and phonetic matching using the metaphone algorithm. A customized slang/abbreviation dictionary was used for replacing the standard slangs and abbreviations. The experiment was conducted by filtering English tweets relating to Indian politics and Prime Minister 'Narendra Modi'.

c) Twitter Data Analysis Using Spark

- Author: Yash Bopardikar, California State University, Sacramento
- Set spark worker instances to 4, which initiates 4 worker instances are soon as the spark cluster is started. The Executor instances are set to 5 and each worker is allocated 8 GB of memory i.e. the spark cluster can use the total amount of memory available on the system. The spark Executor and Worker cores are set to 6 and 10 which are virtual cores.
- Use twitter4j library which is embedded into TwitterUtils library for getting specific fields out of the data.

- Sentiment analysis is performed using Stanford's Natural Language Processing library. It takes text input and sends this library to get the sentiment in return.

d) Public Perception of the COVID-19 Pandemic on Twitter: Sentiment Analysis and Topic Modelling Study

- Author: Sakun Boon-Itt, Yukolpat Shunkan (Originally published in JMIR Public Health and Surveillance)
- This study was conducted in two stages: (1) data collection using the Twitter streaming API to collect COVID-19 related posts in the English Language, and (2) data analysis to identify trends, keywords, and themes.
- The objective of this study was to answer questions related to themes, public concerns, and sentiments regarding the COVID-19 pandemic through social media analytics.
- The tweet database was created by specifying keywords and metadata such as language, source, data range, and location. This search used keywords and specific hashtags (#) such as *coronavirus*, *covid_19*, *2019-nCov*, and *covid-19* in the English language using the searchtweets tool.The tweets were then converted into a corpus.

2.2. Drawbacks of Previous Designs

- It is difficult to maintain the structure of data without using a DBMS.
- There is more redundancy in previous works due to use of only scripting language.
- There is no constraint as to what kind of tweets we have to extract which may increase variance while using data for machine learning algorithms.

3. METHODOLOGY:

3.1. Full Description of the Project

- Twitter is a micro blogging service that allows people to communicate with short 140 character messages. Twitter has become a platform for information dissemination and change. Twitter provides an open platform for expression, and hence mining of tweets can provide valuable insight into political scenario and societal opinion. Tweets can be collected using the REST or Streaming API.

- Although the tweet text is limited to just 140 characters, there is an associated rich metadata like the location, time zone, geographical latitude, geographical longitude, creation date, followers count and retweet count that can be captured and utilized for analysis. The retweet count is an indicator of the popularity of the tweet. The geographical parameters and location indicate the geographical origin of the tweet.

- Mining the text in conjunction with metadata can provide an insight into the different political perspectives, political trends, sentiments and opinions.

- It is required that the user has to have certain things such as a twitter account and API credentials, access to mysql database and libraries such as tweepy and mysql connector.

- We created a configuration code to extract keys such as API_Key, Access token, secret key etc.

- Then we have written an another function to essentially link the database with the details we have extracted from the previous code and create a table to store the tweets which are to be streamed into the database.

- We are using the function createTableUser to store the followers or users in general in a table with user ID as primary key, name, username, description text, account created at, location the tweet was tweeted, followers count, following people count, tweet count and even the profile image URL as the attributes.

- We have also created another function createTavleTweets to store the tweets in a table with tweetID as the primary key, userID as the foreign key, authorID, account created at, tweet text, retweet count, reply count, like count, quote count as the attributes.

- We have also used functions like filterOnFollowers to get the number of active users, ifTweetPresent to chech whether the tweet is present in our

database or not, insertTweet to insert data in tweets table, insertFollower to insert data in users table.

- We are going to use insertInDB function to retrieve the followers of the input username and then creating a table to store the details of those users.
- We are also using the function insertFilteredTweets to retrieve only the filtered tweets which pass the conditions like there have to be minimum of 1000 followers of the person who is following the userID we gave as input, and minimum tweet count as 10. We are also gathering the general public reaction to their tweets by storing the count of likes the tweet got, the number of times it was retweeted etc.
- Then we can perform whatever operation we want to execute on the dataset we collected.
- For the front-end part we have used basic HTML, CSS and flask for visual representation purposes.

3.2. Full Architecture

Code:

1. config.py

```
class Config(object):
   DEBUG = True
   TESTING = False

class DevelopmentConfig(Config):
   SECRET_KEY = "12345"

config = {
   'development': DevelopmentConfig,
   'testing': DevelopmentConfig,
   'production': DevelopmentConfig
}

#API Config
API_KEY = 'API key'
API_KEY_SECRET = 'API_KEY_SECRET'
BEARER_TOKEN = 'BEARER_TOKEN'
ACCESS_TOKEN = 'Access Token'
ACCESS_TOKEN_SECRET = 'Access Token Secret'

#Database Config
host='localhost'
```

```
user='root'
password='password'
database='TwitterDB'
```

2. mySqlConnection.py

```
from colorama import Cursor
import mysql.connector

#creating connector object to connect with database
def connect(host, user, password, database):

 conn = mysql.connector.connect(
   host=host,
   user=user,
   password=password,
   database=database
 )
 return conn

#creating table to store followers or user in general
def createTableUser(conn,tableName):
 cursor=conn.cursor()
 sql = "create table "+ str(tableName)+"""(
         userID bigint primary key,
         name varchar(100),
         username varchar(100),
         description text,
         created_at varchar(45),
         location varchar(100),
         followers_count int,
         following_count int,
         tweet_count int,
         verified boolean,
         profileImageURL varchar(255)
       );
   """
 cursor.execute(sql)
 cursor.close()

#creating table to store tweets
def createTableTweets(conn,tableName,userTableName):
 cursor=conn.cursor()
 sql = "create table "+ str(tableName)+"""(
         tweetID bigint,
         userID bigint,
         authorID bigint,
         created_at varchar(45),
         tweet text,
         retweet_count int,
         reply_count int,
         like_count int,
         quote_count int,
         PRIMARY KEY(tweetID),
```

```
        FOREIGN KEY(userID) REFERENCES """
+userTableName+"(userid));"

 cursor.execute(sql)
 cursor.close()

#inserting data in users tables
def insertFollower(conn, recordFollower, tableName):
 cursor = conn.cursor()
 query="INSERT INTO "+ tableName +"""
         (userID, name, username, description, created_at,
location, followers_count, following_count, tweet_count,
verified, profileImageURL)
         VALUES (%s,%s,%s,%s,%s,%s,%s,%s,%s,%s,%s)"""
  cursor.execute(query,recordFollower)
 conn.commit()
 cursor.close()
 #inserting data in tweets table
def insertTweet(conn, recordTweet,tableName):
 cursor = conn.cursor()
 query="INSERT INTO " + tableName +"""
         (tweetID, userID, authorID, created_at, tweet,
retweet_count, reply_count, like_count, quote_count)
         VALUES (%s,%s,%s,%s,%s,%s,%s,%s,%s)"""
  cursor.execute(query,recordTweet)
 conn.commit()
 cursor.close()
 #query to get active users
def filterOnFollowers(conn,tableName,minFollowers,minTweetCount):
 cursor=conn.cursor()
 query=("SELECT username FROM " +tableName+ " WHERE
followers_count > "+str(minFollowers) + " and tweet_count >
"+str(minTweetCount))
 cursor.execute(query)
 result = cursor.fetchall()
 cursor.close()
 return result

#function to check if a tweet is present
def ifTweetPresent(conn, tableName, tweetID):
 cursor=conn.cursor()
 query = ("SELECT tweetID from ")+tableName+(" where tweetID =
")+str(tweetID)
 cursor.execute(query)
 result=cursor.fetchall()
 cursor.close()
 if len(result)!=0: return True
 else: return False
```

3. search.py

```
#imports
import tweepy
import config
import mysqlConnection

#setting up a client to make API request to twitter API
def createClient():
   client = tweepy.Client(bearer_token=config.BEARER_TOKEN,
                          consumer_key=config.API_KEY,
                          consumer_secret=config.API_KEY_SECRET,
                          access_token=config.ACCESS_TOKEN,

access_token_secret=config.ACCESS_TOKEN_SECRET)
   return client


#Retriving followers of the input username and then creating a table to
store the details of those users
def insertInDB(client,yourUsername):

   #getting id of user from username
   id=client.get_user(username=yourUsername).data.id

   #fields that we want to extract

user_fields=['description','created_at','location','public_metrics','veri
fied','profile_image_url']

   #getting the followers of the input user

followers=client.get_users_followers(id,user_fields=user_fields,max_resul
ts=1000)

   #forming a table to store the data
   conn=mysqlConnection.connect(config.host, config.user,
config.password, config.database) #mysql connector object

   #creating the table
   tableName=yourUsername+"Followers"
   mysqlConnection.createTableUser(conn,tableName)

   #inserting data in table
   for user in followers.data:


record=(user.id,user.name,user.username,user.description,user.created_at,
user.location,user.public_metrics['followers_count'],user.public_metrics[
'following_count'],user.public_metrics['tweet_count'],user.verified,user.
profile_image_url)

       mysqlConnection.insertFollower(conn,record,tableName)


 #getting posted and liked tweets of active users
```

```
#active user are based upon minimum followers they have and the total
number of tweets they've done.

def
insertFilteredTweets(client,userName,minFollowers=1000,minTweetCount=10):

   conn = mysqlConnection.connect(config.host, config.user,
config.password, config.database)  #mysql connection object

   #getting usernames of active users
   followerTableName=userName+'Followers'

userNames=mysqlConnection.filterOnFollowers(conn,followerTableName,minFol
lowers,minTweetCount)

   #getting tweets
   for user in userNames:
       tweet_fields=['author_id','created_at','geo','public_metrics']

       #getting tweets posted by the user
       query='from:'+user[0]+' -is:retweet'
       tweetsPosted =
client.search_recent_tweets(query=query,tweet_fields=tweet_fields,
max_results=100)

       #getting tweets liked by the user
       userID=client.get_user(username=user[0]).data.id
       tweetsLiked = client.get_liked_tweets(id=userID,
tweet_fields=tweet_fields)

       if tweetsPosted.data==None or tweetsLiked.data==None: continue

       #creating table for the user
       userTweetTable = user[0]+"_"+userName+"_Tweets"

mysqlConnection.createTableTweets(conn,userTweetTable,followerTableName)

       #function to insert tweets
       def insert(tweets,tableName,userID):
           for tweet in tweets.data:

record=(tweet.id,userID,tweet.author_id,tweet.created_at,tweet.text,tweet
.public_metrics['retweet_count'],tweet.public_metrics['reply_count'],twee
t.public_metrics['like_count'],tweet.public_metrics['quote_count'])

               #checking if tweet already inserted....
               if
mysqlConnection.ifTweetPresent(conn,tableName,tweet.id): continue

               #inserting the tweet
               mysqlConnection.insertTweet(conn,record,tableName)

       insert(tweetsPosted,userTweetTable,userID)
       insert(tweetsLiked,userTweetTable,userID)
```

4. app.py

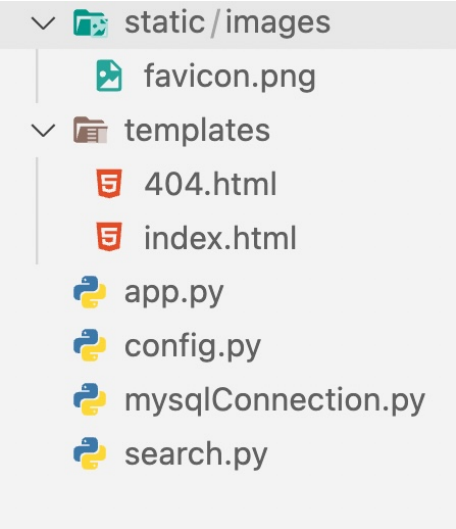
```
from flask import Flask, render_template, url_for, redirect,
request
import config
import search


def page_not_found(e):
 return render_template('404.html'), 404


app = Flask(__name__)
app.config.from_object(config.config['development'])

app.register_error_handler(404, page_not_found)



@app.route('/', methods=["GET", "POST"])
def index():
   client=search.createClient()
   if request.method == 'POST':
       userName=request.form['username']
       minFollowers=request.form['minFollowers']
       minTweetCount = request.form['minTweetCount']

       search.insertInDB(client,userName)

search.insertFilteredTweets(client,userName,minFollowers,minTweet
Count)

   return render_template('index.html', **locals())


@app.route('/retweet/<string:tweetId>/', methods=["GET", "POST"])
def reweet(tweetId):

   return redirect(url_for('index'))


if __name__ == '__main__':
   app.run(host='0.0.0.0', port='8000', debug=True)
```

5. index.html

```
<!doctype html>
<html lang="en">
 <head>
   <!-- Required meta tags -->
   <meta charset="utf-8">
   <meta name="viewport" content="width=device-width,
initial-scale=1">

   <title>Twitter Followers Analysis</title>
   <link rel="shortcut icon" type="image/x-icon" href="{{
url_for('static', filename='images/favicon.png') }}">

   <!-- Bootstrap CSS -->
   <link
href="https://cdn.jsdelivr.net/npm/bootstrap@5.1.3/dist/css/boots
trap.min.css" rel="stylesheet"
integrity="sha384-1BmE4kWBq78iYhFldvKuhfTAU6auU8tT94WrHftjDbrCEXS
U1oBoqyl2QvZ6jIW3" crossorigin="anonymous">

 </head>
 <body>

   <div class="container mt-5 mb-5">

     <h1 class="mb-3">Twitter Follower Analysis</h1>

     <form class="" action="/" method="post">

       <div class="col-sm-3">
         <label class="sr-only">Username</label>
         <div class="input-group-prepend">
           <div class="input-group-text">@</div>
         </div>
         <input type="text" class="form-control"
placeholder="Username" id="username" name="username">
       </div>
```

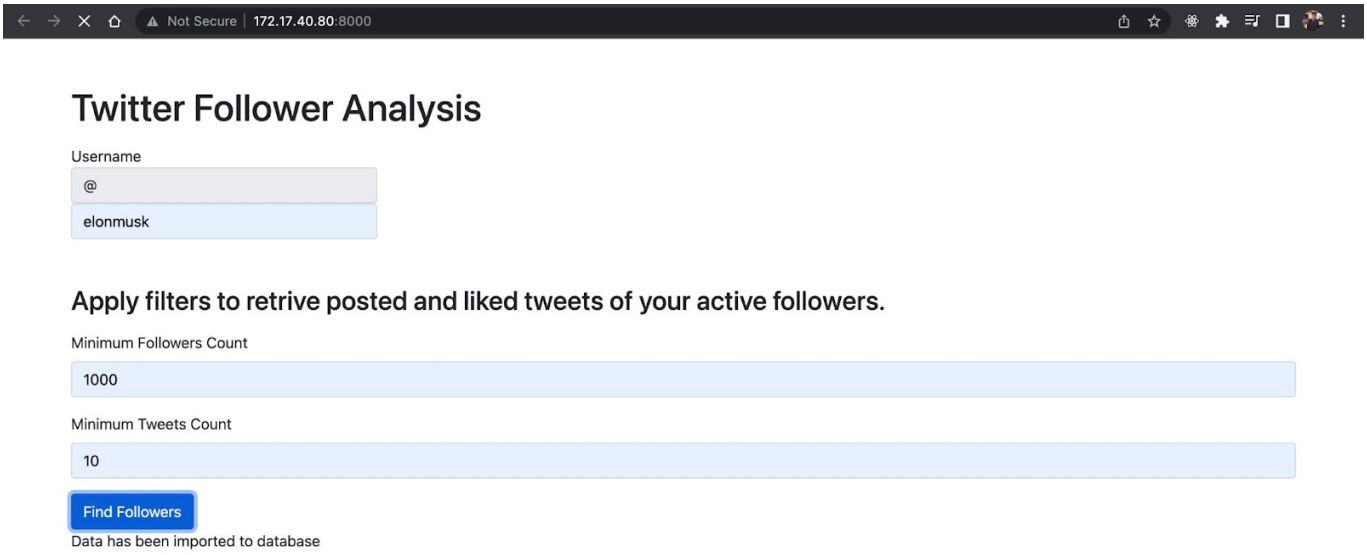
```
        <div class="mb-3">
          <br><br>
          <h3 class="mb-3">Apply filters to retrive posted and
liked tweets of your active followers.</h3>
          <label for="minFollowers" class="form-label">Minimum
Followers Count</label>
          <input type="text" class="form-control"
id="minFollowers" name="minFollowers" placeholder="1000....">
        </div>

        <div class="mb-3">
          <label for="minTweetCount" class="form-label">Minimum
Tweets Count</label>
          <input type="text" class="form-control"
id="minTweetCount" name="minTweetCount" placeholder="10.....">
        </div>

        <button type="submit" class="btn btn-primary"
onclick="result()"> Find Followers </button>
        <p id="output"></p>
        <br>
        <br>
        <script> function result() {
          document.getElementById("output").innerHTML = "Data has
been imported to database";
        }
        </script>
      </form>
    </div>

    <!-- Option 1: Bootstrap Bundle with Popper -->
    <script
src="https://cdn.jsdelivr.net/npm/bootstrap@5.1.3/dist/js/bootstr
ap.bundle.min.js"
integrity="sha384-ka7Sk0Gln4gmtz2MlQnikT1wXgYsOg+OMhuP+IlRH9sENBO
0LRn5q+8nbTov4+1p" crossorigin="anonymous"></script>
  </body>
</html>
```

File Structure:

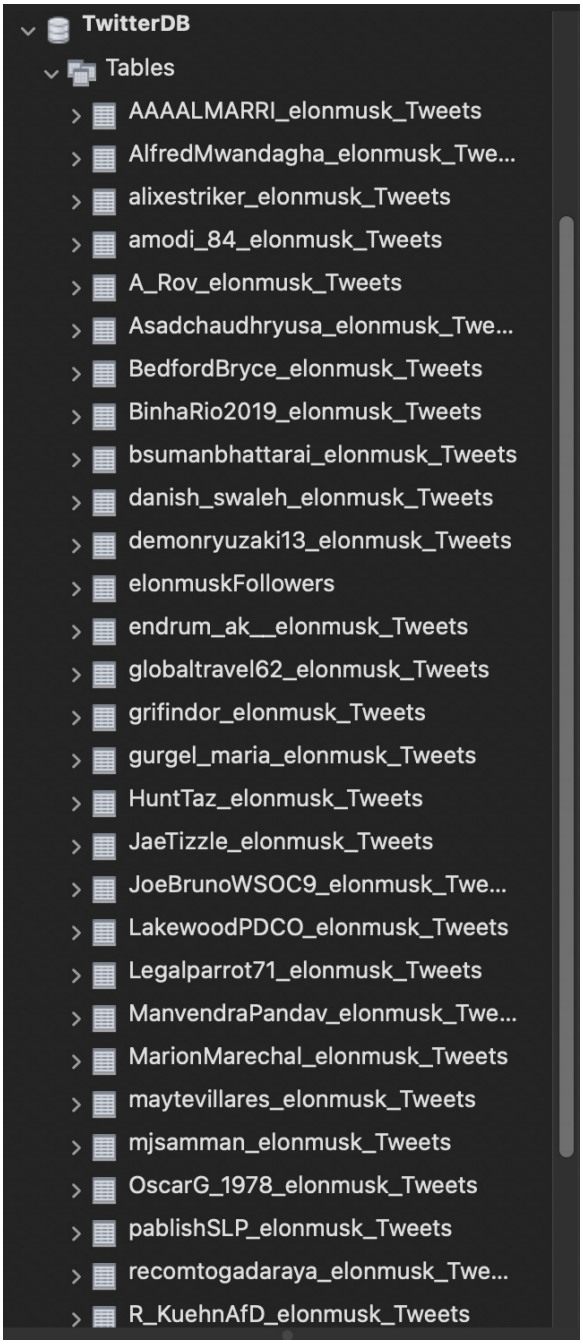


Output:

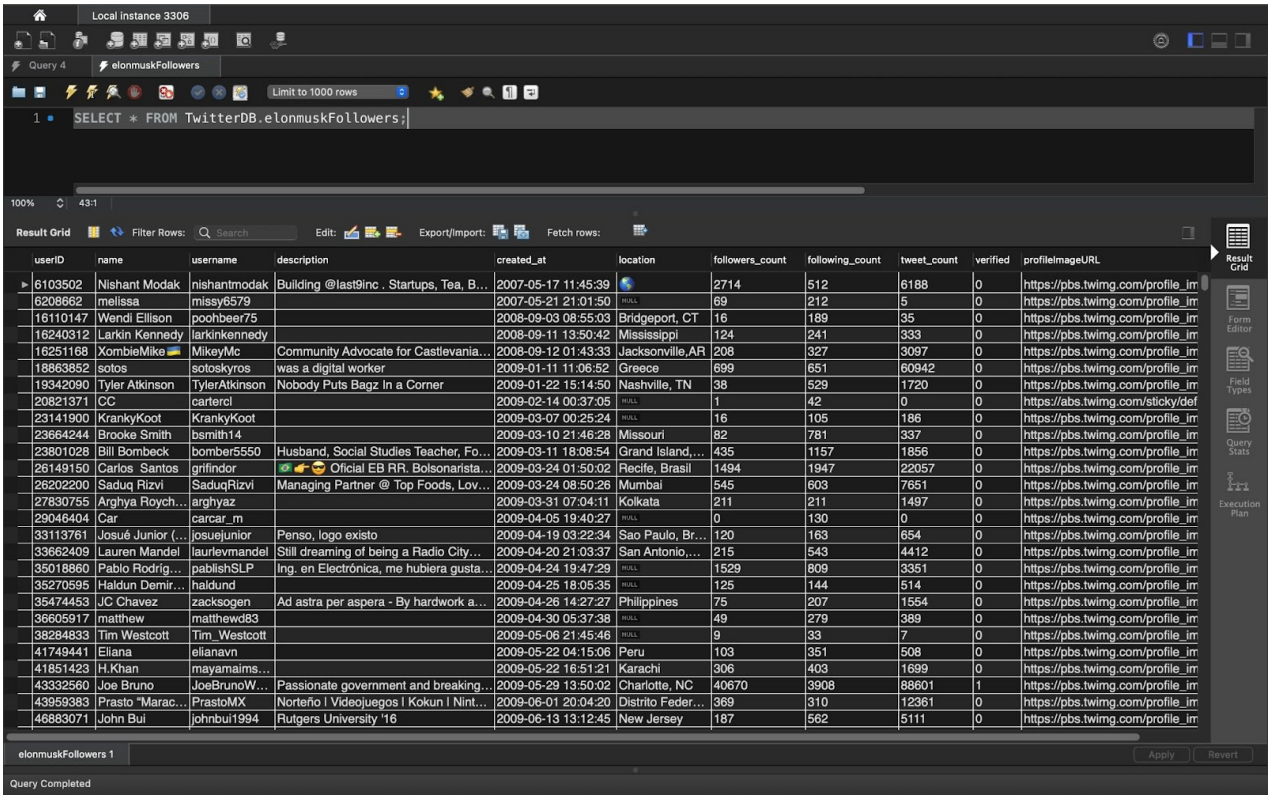
Web Interface



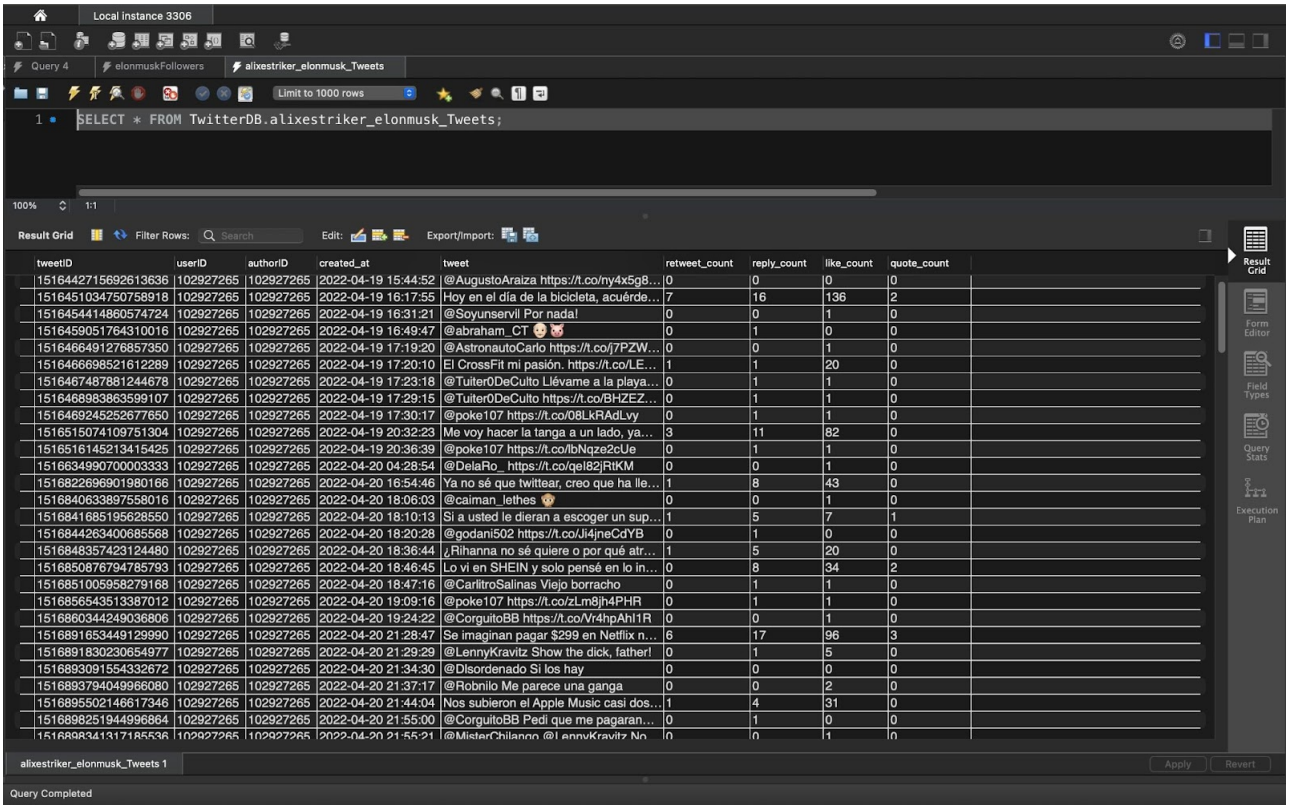
Tables:


TwitterDB
Tables
AAAALMARRI_elonmusk_Tweets
AlfredMwandagha_elonmusk_Twe...
alixestriker_elonmusk_Tweets
amodi_84_elonmusk_Tweets
A_Rov_elonmusk_Tweets
Asadchaudhryusa_elonmusk_Twe...
BedfordBryce_elonmusk_Tweets
BinhaRio2019_elonmusk_Tweets
bsumanbhattarai_elonmusk_Tweets
danish_swaleh_elonmusk_Tweets
demonryuzaki13_elonmusk_Tweets
elonmuskFollowers
endrum_ak__elonmusk_Tweets
globaltravel62_elonmusk_Tweets
grifindor_elonmusk_Tweets
gurgel_maria_elonmusk_Tweets
HuntTaz_elonmusk_Tweets
JaeTizzle_elonmusk_Tweets
JoeBrunoWSOC9_elonmusk_Twe...
LakewoodPDCO_elonmusk_Tweets
Legalparrot71_elonmusk_Tweets
ManvendraPandav_elonmusk_Twe...
MarionMarechal_elonmusk_Tweets
maytevillares_elonmusk_Tweets
mjsamman_elonmusk_Tweets
OscarG_1978_elonmusk_Tweets
pablishSLP_elonmusk_Tweets
recomtogadaraya_elonmusk_Twe...
R_KuehnAfD_elonmusk_Tweets

Followers Table Structure:



Tweets Table Structure:



RESULT ANALYSIS:

- Two types of tables are obtained, one for users(followers) and another for tweets.
- First, we retrieve a relation containing information about the followers of the input user.
- Then we retrieve the tweets posted and liked by the active followers of the input user.
- This data is extremely valuable for a social media manager because they can analyse it to form the best possible strategy for the social media handles of the given company.

CONCLUSION:

We used the Twitter API to look for tweets posted and liked by the followers of the concerned account and imported them directly into our database. We used the Tweepy library in python which made it very simple to connect to the API and begin streaming data.This has demonstrated that setting up a data pipeline using MYSQL is not difficult, especially when using the great modules that python has to offer.

REFERENCES AND CITATIONS:

Previous Works:

1. https://link.springer.com/chapter/10.1007/978-3-642-24477-3_7

   Albert Bifet, Geoffrey Holmes, Bernhard Pfranhinger ‘MOA-Tweet Reader Real-Time Analysis in Twitter Streaming Data’

2. https://publichealth.jmir.org/2020/4/e21978

   Sakun Boon-Itt, Yukolpat Shunkan (Originally published in JMIR Public Health and Surveillance) ‘Public Perception of the COVID-19 Pandemic on Twitter: Sentiment Analysis and Topic Modeling Study’

3. http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.735.9448&rep=rep1&type=pdf

Nazura Javed Research Scholar Bangalore University Bangalore, India and Muralidhara B.L. Bangalore University Bangalore, India ‘Automating Corpora Generation with Semantic Cleaning and Tagging of Tweets for Multi-dimensional Social Media Analytics’

4. https://scholarworks.calstate.edu/downloads/db78tc01m

   Yash Bopardikar, California State University, Sacramento ‘Twitter Data Analysis Using Spark’

Documentation:

1. https://docs.tweepy.org/en/stable/client.html
2. https://dev.mysql.com/doc/connector-python/en/