Homework Assignment #4

(Capturing and Wrangling Twitter Data)

Your Data Science career at Netflicks.com continues to flourish! Your boss wants you to be in charge of a new effort to understand what's happening in the world - the mood of the crowd, which is of course Netflicks.com''s customer base - by monitoring social media. Given your other recent successes, you're a bigger-than-ever blip on his radar screen now. Since you've delivered on all of his previous assignments, this week brings yet another: Your boss wants you to explore the tools and techniques for social media data gathering and analytics in the Python ecosystem, and again his thought is, given your previous successes: "Hey, how hard can that be…?"

Your next assignment is to become something of an expert on acquiring, saving, and wrangling large volumes of social media data using Twitter APIs and Tweets as an example.

Get Ready...

In addition to PostgreSQL (again) and the mainstream components of Anaconda, you will be making use of the Twitter developer APIs and a popular supporting Python library called Tweepy (http://www.tweepy.org/) for this assignment. Grab Tweepy if you don't already have it and read up on it a bit. You will also be using some of the techniques covered in class for doing non-query things with PostgreSQL, so make sure that you are comfortable with those. Lastly, to use Tweepy, you will need to have a Twitter account - so it's time to get yourself one if you don't already have one. (Since all of the CS170A course projects will likely include a social media component, consider this another investment in project preparation as well - so even if getting a Twitter account hasn't been high on your to-do list in the past, it's time for you to join the leaders of countries all over the world and start Tweeting!)

Get (Data) Set...

For this assignment you are going to be acquiring, archiving, and analyzing Tweets. Because your boss envisions doing this for a LOT of Tweets, you're going to do so the scalable way - you will be storing and manipulating your Tweets in PostgreSQL, using Python and Tweepy just as a means to that end. To make the assignment more fun (as well as quicker to write), we're going to give you a little flexibility - think of 2-4 keywords that might make Tweets interesting to you, and play around with Twitter's interactive search interface (https://twitter.com/search-home) to see what those words turn up (e.g., https://twitter.com/search?q=trump%20mueller&src=typd). Iterate a bit until you find a search query that yields Tweets whose results you would like to analyze. This is also a great way to get familiar with the query syntax for Twitter searches, so feel free to continue exploring the search capabilities this way. More info about Twitter's search operators can be found here:
https://developer.twitter.com/en/docs/tweets/rules-and-filtering/overview/standard-operators.html

Go...!

It's time to get to work. Tackle each of the following questions and problems using Tweepy to access the Twitter APIs and using PostgreSQL as your social data storage and query platform:

1. Design a schema, and a set of CREATE TABLE statements, for storing the data that you are going to gather. Tweets contain a number of different kinds of information, both at the top level and nested within, including information about a given Tweet (e.g., its text), information about the Tweet's content (e.g., hashtags), static info about the Tweet's user (e.g., their screen name), and dynamic info about the user (e.g., how many times they have Tweeted). (See https://developer.twitter.com/en/docs/tweets/data-dictionary). You will be capturing your Tweet data in a set of 1NF tables by collecting the Tweets with Tweepy and then "splitting them" up by writing their pieces into these tables. Create a collection of interrelated tables to hold at least the following information:
    a. Tweet information, including created_at, id, text, in_reply_to_status_id, and in_reply_to_screen_name. Also include a raw_tweet field somewhere in your design that captures the original Tweet (in case you realize later on that there's more information that you'd like to get from the Tweets).
    b. Static user information, including their id, name, screen_name, description, and created_at.
    c. Dynamic user information, including their statuses_count, followers_count, and friends_count.
    d. Tweet entity information, specifically hashtags.
2. Write a small Python program that uses your search query to gather 10,000 Tweets and stores them in your PostgreSQL tables. Start by testing your program on a small set of Tweets (say 10-100 or so) and then, once everything seems to be working, run it "at scale" to gather your 10,000 Tweets. Beware of Twitter rate limiting as you do this - your program will need to be respectful of that and will have to run for awhile, so you will need to get this step of the assignment done 1-2 days before the deadline to leave time for the last part of this problem! (Seriously!)
3. Working (only) with SQL and PostgreSQL, do some simple analysis of your collection of Tweets - record your queries and their results (e.g., by creating a script file that you can run again when you are done exploring your data):
    a. How many unique users does your Tweet collection have?
    b. What is the average number of Tweets per user in your collection?
    c. What fraction of your Tweets are (i) original Tweets vs. (ii) reply Tweets?
    d. Who are your top 20 Tweeters (user id, user name, and screen name) by Tweet count, and what are their Tweet counts, how long have they been users, and what are their last known statuses and followers counts?
    e. How many unique hashtags does your Tweet collection have?
    f. What is the average number of hashtags per Tweet?
    g. What are the top 20 hashtags and what are the Tweet counts for each?

What To Turn In

When you have finished your assignment you should use EEE to turn in a PDF file showing all of your work - including your database setup DDL, your Python program, and your SQL queries and results.