**DSC 450: Database Processing for Large-Scale Analytics**

**Assignment 4**

**Due Wednesday, March 13th 11PM.**

**Part 1. Python (15 points)**

We are going to work with a small extract of tweets (about 200 of them), available on D2L as Tweet1\_Assignment4.txt

Create a SQL table to contain the following few attributes of a tweet:

"created\_at", "id\_str", "text", "source", "in\_reply\_to\_user\_id", “in\_reply\_to\_screen\_name”, “in\_reply\_to\_status\_id”, "retweet\_count", “contributors”. Please assign reasonable data types to each attribute (e.g., VARCHAR(10000) is a bad idea).

Use SQLite for this assignment.

**NOTE 1**: I do not recommend trying to copy-paste this text, because there is absolutely no knowing what might come out from paste on your system. You should download the Tweets1\_Assignment4.txt file.

**NOTE 2**: The input data is separated by a string “EndOfTweet” which serves as a delimiter. The text itself consists of a single line, so using readlines() will still only give you one row which needs to be split by the tweet delimiter.

**NOTE 3:** Do not forget that you can use json.loads(OneTweetString.decode('utf8')) to parse a tweet entity into a python dictionary. Don’t forget to add import json to your code.

**NOTE 4:** Make sure your python code reads through the file and loads the data properly (including NULLs).

Write the following SQL queries:

1. Count the number of iPhone users and android users. (based on “source” attribute)
2. Count the number of tweets with users who are not replying ("in\_reply\_to\_user\_id" is NULL).

Submit your Part1.py Python file. No need to submit JSON libraries.

**Part 2. Regex (15 points)**

The Chicago.txt is the Wikipedia entry on Chicago available from <https://en.wikipedia.org/wiki/Chicago>. Chicago.txt is a bit outdated but follows the general heading section. Use Python regular expression library to find the following patterns in Chicago.txt:

1. Count the number of sections in this text. Note sections begin with ‘[edit]’.
2. In the Crime section, report all the years mentioned in sorted (ascending) order.
3. Find all county names used in this Wikipedia article.

Submit your Part2.py Python code.

**Part 3.** **Comprehensive** **(25 points)**

Now use a larger collection of tweets. Download Tweet2\_Assignment4.txt. The tweets are all on separate lines, but some of the tweets are intentionally damaged and will not parse properly. You will need to store these tweets in a separate “error” file.

Write python code that is going to read and load the Tweet2\_Assignment4.txt. file. For tweets that could not parse, simply store them in Assignment4\_errors.txt file

Note1. : Gracefully catch JSON errors.

Note2: As discussed in class, you can access the contents of the user dictionary after it was parsed by json like this:

dict['user'] # user dictionary

dict['user']['id'] # user’s ID

For tweets that parsed, create a new SQL table for the user dictionary. It should contain the following attributes “id”, “name”, “screen\_name”, “description” and “friends\_count”. Modify your SQL table from Part 1 to include “user\_id” which will be a foreign key referencing the user table.

and populate both of your tables (Tweet table from Assignment4 and User table from this assignment), with data from Tweet2\_Assignment4.txt.

* Write and execute a SQL query to do the following: Find the user (“id” and “name”) with the highest “friend\_count” in the database
* Write python code that is going to perform the same computation (find the user with the highest “friend\_count”)
* Write and execute SQL query to do the following: Find the tweets without associated user id entry.
* Write python code that is going to perform the same computation as 3-c. (Requires Python DataFrames)
* Using Python, identify the top-3 most frequent terms (i.e. words separated by ‘ ‘) in the text of the tweets. It is up to you whether you prefer to use the contents of the loaded database (reading tweets from SQLite, which contains fewer tweets) or the contents of the original Tweet2\_Assignment4.txt file (reading tweets directly from the file).

Submit Part3.py Python code.

**Part 4.** **SQLAlchemy (10 points)**

Perform PART1 using SQLALchemy. So the table within SQLite should be created using SQLAlchemy and data inserted into the table using pre-compiled statements from SQLAlchemy.

Then use SQLAlchemy to write the queries. You must determine how to filter NULLs in SQLAlchemy.

Submit Part4.py. No need to submit supporting libraries.