**CSC 455: Database Processing for Large-Scale Analytics**

**Assignment 5**

**Due 5:30PM, Tuesday, November 17th.**

**Supplemental reading:** Python for Data Analytics, Chapter 7

1. In this assignment we are going to work with a larger collection of tweets (10,000) and available here:

http://rasinsrv07.cstcis.cti.depaul.edu/CSC455/Assignment5.txt

**NOTE**: There are 10000 tweets, but you need to load only the first 7000 tweets from the file. 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. At the bottom of the page you can find python code that will let you skip over badly formed tweets.

* 1. Create a SQL table for the user entry. It should contain the following attributes “id”, “name”, “screen\_name”, “description” and “friends\_count” and modify your SQL table from Assignment 4 to include “user\_id” which will be a foreign key referencing the user table.

* 1. Write python code that is going to read 7000 (only 7000) tweets from the Assignment5.txt file from the web and populate both of your tables. It doesn’t matter if you load exactly 7000 tweets or “7000 less the bad tweets”, but do not load the entire file.

You should use urllib.request to read data directly from the given link (as discussed in class)

###Python 3.x

import urllib.request as urllib

response = urllib.urlopen("http://rasinsrv07.cstcis.cti.depaul.edu/CSC455/Assignment5.txt")

str\_response = response.readline().decode("utf8")

tDict = json.loads(str\_response)

tDict['id']

### if you are using Python 2.x, this should work

import urllib2

wfd = urllib2.urlopen('http://rasinsrv07.cstcis.cti.depaul.edu/CSC455/Assignment5.txt')

data = wfd.readline() # To read one line

You can use JSON for tweet parsing; be sure to record tweets that fail parsing in a separate file – at least a few will fail. You can gracefully catch JSON errors using the following code:

for tweet in allTweets: # a version of this is also in lecture8-week9.py file in D2L Content

try:

tDict = json.loads(str\_response)

except ValueError:

# Handle the problematic tweet, which in your case would require writing it to another file

print (tweet) #this is just printing for example’s sake

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

Code Submission: csc455-HW5-Q1\_ab.py

dict['user']['id'] # user’s ID

1. *My table name is tweetuser*
   1. Write a SQL query to do the following: Find the user (“id” and “name”) with the highest “friends\_count” in the database

**SELECT id, name FROM tweetuser**

**WHERE friends\_count = (SELECT MAX(friends\_count) FROM tweetuser);**

* 1. Write python code that is going to perform the same computation (find the user with the highest “friend\_count”)

Code Submission: csc455-HW5-Q2-b.py

1. Using python, perform word frequency analysis and identify the top-3 most frequent terms in the text of the tweets. It is entirely 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 Assignment5.txt file (reading tweets directly from the file). Here is some sample code that shows how to perform word frequency analysis for one string:

from operator import itemgetter

text = "This is a frequency count test . The most frequent word will be the word test . Shifting periods to avoid complexity of confusing test and test. in this test sentence"

words = text.split(' ')

dCount = {}

for word in words:

if word not in dCount.keys():

dCount[word] = 0

dCount[word] = dCount[word]+1

countKeys = dCount.keys()

countVals = dCount.values()

# We could use dCount.items(), but we want the values to come first and the keys to be second

countPairs = zip(countVals, countKeys)

sorted\_countPairs = sorted(countPairs, key=itemgetter(0), reverse=True)

# The words are now sorted by descending frequency

print (sorted\_countPairs)

Code Submission: csc455-HW5-Q3.py

1. **OPTIONAL**: There are several optional problems which will provide a few extra points towards your grade. You can get full credit for this assignment by doing Parts 1-3 only, this part is only for extra credit:

Using the ZooKeeper database:

* 1. [1pt] Write a SQL query that finds all animals without a zookeeper assignment using NOT EXISTS with a correlated nested sub-query.

SELECT Animal.AName, AID FROM ANIMAL

WHERE NOT EXISTS

(SELECT ANIMALID

FROM HANDLES

WHERE Animal.AID = Handles.ANIMALID);

* 1. [2.5pts] Write a trigger using PL/SQL in Oracle that will ensure that TimeToFeed defaults to at least 0.25 (i.e. if TimeToFeed is less than 0.25, reset it to the value of 0.25)
  2. [2pts] Write a regular expression for identifying Social Security Numbers in the text.

SELECT \* FROM EMPLOYEE

WHERE REGEXP\_LIKE (SSN, '^[0-9]{3}-[0-9]{2}-[0-9]{4}$');