Christian Craig
DSC 450 - Final

# lcDSC 450: Database Processing for Large-Scale Analytics
## Take-home Final

## Part 1

We will use a full day worth of tweets as an input (there are total of 4.4M tweets in this file, but we will intentionally use fewer tweets):
http://rasinsrv07.cstcis.cti.depaul.edu/CSC455/OneDayOfTweets.txt

Create a third table, now incorporating the Geo table (in addition to tweet and user tables that you already have) and extend your schema accordingly. You do not need to use ALTER TABLE, you can simply recreate your schema with new columns.
You will need to generate an ID for the Geo table primary key (you may use any value or reasonable combination of values as long as it is unique) for that table and link it to the Tweet table (foreign key should be in the Tweet table). In addition to the primary key column, the geo table should have the "type", "longitude" and "latitude" columns.
**NOTE:** there is no key called longitude or latitude in the tweet dictionary. Instead, the geo dictionary has a single 2-value tuple which are the coordinates (longitude, latitude).

```
'''
import urllib.request
import json
import sqlite3
Geo_Table= '''
    CREATE TABLE Geo (
    geo_id UNIQUE,
    type VARCHAR2(150),
    latitude VARCHAR2(150),
    longitude VARCHAR2(150),

    Constraint Geo_PK
        PRIMARY Key(geo_id)
 );
'''
User_Table = '''
    CREATE TABLE User (
    id   VARCHAR2(25),
    name     VARCHAR2(50),
    screen_name VARCHAR2(50),
    description VARCHAR2(75),
    friend_count NUMBER,
    Constraint user_PK
        PRIMARY KEY(id)

);
'''
Tweet_Table = '''
    CREATE TABLE Tweet (
    created_at VARCHAR2(25),
    tweet_id    VARCHAR2(25),
    text     VARCHAR2(280),
    source VARCHAR2(100),
    in_reply_to_user_id VARCHAR2(30),
    in_reply_to_screen_name VARCHAR2(30),
    in_reply_to_status_id VARCHAR2 (30),
    retweet_count NUMBER,
    contributors VARCHAR2(15),
    user_id    VARCHAR2(25),
    geo_id NULL,

    Constraint Tweets_PK
        PRIMARY KEY(tweet_id)

    Constraint Tweets_FK1
        FOREIGN KEY (user_id)
            REFERENCES User(id)

    Constraint Tweets_FK2
        FOREIGN KEY (geo_id)
            REFERENCES geo(geo_id)
);
'''

```

Christian Craig
DSC 450 - Final

Execute the following tasks with 50,000 tweets, 100,000 tweets, and 500,000 tweets. Collect and plot the runtimes in part 1-e.

a. Use python to download tweets from the web and save to a local text file (not into a database yet, just to a text file).

**NOTE**: Do not call read() or readlines(). That command will attempt to read the entire file which is too much data.

```
#50k
webFD = urllib.request.urlopen('http://rasinsrv07.cstcis.cti.depaul.edu/CSC455/OneDayOfTweets.txt'
fd = open('FinalTweets50k.txt', 'w', encoding='utf-8')
start1a50k = time.time()
for i in range(50000):
    line = webFD.readline().decode("utf8")
    fd.write(str(line))
end1a50k = time.time()
t50k = ("Elapsed time for 50,000 tweets is: " + str(end1a50k - start1a50k)+" seconds")
print(t50k) #15 seconds
fd.close()


#100k
webFD = urllib.request.urlopen('http://rasinsrv07.cstcis.cti.depaul.edu/CSC455/OneDayOfTweets.txt'
fd = open('FinalTweets100k.txt', 'w', encoding='utf-8')
start1a100k = time.time()
for i in range(100000):
    line = webFD.readline().decode("utf8")
    fd.write(str(line))
end1a100k = time.time()
t100k = ("Elapsed time for 100,000 tweets is: " + str(end1a100k - start1a100k)+" seconds")
print(t100k)#29 seconds
fd.close()

#500k
webFD = urllib.request.urlopen('http://rasinsrv07.cstcis.cti.depaul.edu/CSC455/OneDayOfTweets.txt'
fd = open('FinalTweets500k.txt', 'w', encoding='utf-8')
start1a500k = time.time()
for i in range(500000):
    line = webFD.readline().decode("utf8")
    fd.write(str(line))
end1a500k = time.time()
t500k = ("Elapsed time for 500,000 tweets is: " + str(end1a500k - start1a500k)+" seconds")
print(t500k)#142 seconds

webFD.close()
fd.close()
```

```
    ...:
    ...: end = time.time()
    ...: t50k = ("Elapsed time for 50,000 tweets
is: " + str(end - start)+" seconds")
    ...: print(t50k) #14 seconds
    ...: fd.close()
Elapsed time for 50,000 tweets is:
14.536603927612305 seconds
```

```
    ...:
    ...: end = time.time()
    ...: t100k = ("Elapsed time for 100,000
tweets is: " + str(end - start)+" seconds")
    ...: print(t100k)#27 seconds
    ...: fd.close()
Elapsed time for 100,000 tweets is:
29.01426672935486 seconds
```

```
    ...: for i in range(500000):
    ...:     line =
webFD.readline().decode("utf8")
    ...:     fd.write(str(line))
    ...:
    ...: end = time.time()
    ...: t500k = ("Elapsed time for 500,000
tweets is: " + str(end - start)+" seconds")
    ...: print(t500k)#145 seconds
Elapsed time for 500,000 tweets is:
142.28349232673645 seconds
```

b. Repeat what you did in part-a, but instead of saving tweets to the file, populate the 3-table schema that you created in SQLite. Be sure to execute commit and verify that the data has been successfully loaded (report loaded row counts for each of the 3 tables).
**Below format same for 100k & 500k**

```
import time
webFD = urllib.request.urlopen('http://rasinsrv07.cstcis.cti.depaul.edu/CSC455/OneDayOfTweets.txt')
start1b50k=time.time()
import json
TweetDict=[]
lstError = []
for i in range(50000):#50k or 100k or 500k
    if i % 100 == 0: # Print a message every 100th tweet read
        print ("Processed " + str(i) + " tweets")
    line = webFD.readline()
    try:
        T1 = json.loads(line.decode('utf8'))
        TweetDict.append(T1)
    except (ValueError): #prof mentioned no errors
        lstError.append(line)

lstTweet = []
lstUser = []
lstGeo = []
for clean_tweet in TweetDict:

        userDict=clean_tweet["user"]
        geoDict=clean_tweet["geo"]
        lstUser.append((userDict["id"],userDict["name"], userDict["screen_name"], userDict["description"],userDict["friends_count"]))

        if geoDict!=None:
            longitude = geoDict["coordinates"][0]
            latitude = geoDict["coordinates"][1]
            geo_id = str(longitude)+str(latitude)

            lstGeo.append((geo_id,geoDict['type'],longitude,latitude))
            lstTweet.append((clean_tweet["created_at"], clean_tweet["id_str"], clean_tweet["text"], clean_tweet["source"], clean_tweet["in_reply_to_us

        else:

            lstTweet.append((clean_tweet["created_at"], clean_tweet["id_str"], clean_tweet["text"], clean_tweet["source"], clean_tweet["in_reply_to_us

conn = sqlite3.connect('dsc450.db')
cursor = conn.cursor()
cursor.execute("Drop Table Tweet")
cursor.execute("Drop Table User")
cursor.execute("Drop Table Geo")

cursor.execute(Tweet_Table)
cursor.execute(User_Table)
cursor.execute(Geo_Table)

sqlTweet = cursor.executemany("INSERT OR IGNORE INTO Tweet VALUES (?,?,?,?,?,?,?,?,?,?,?,?)",lstTweet)
sqlUser = cursor.executemany("INSERT OR IGNORE INTO User VALUES (?,?,?,?,?)",lstUser)
sqlGeo = cursor.executemany("INSERT OR IGNORE INTO Geo VALUES (?,?,?,?)",lstGeo)

queryTweet = '''SELECT COUNT(*) FROM Tweet'''
queryUser  ='''SELECT COUNT(*) FROM User'''
queryGeo = '''SELECT COUNT(*) FROM Geo'''

countTweet = cursor.execute(queryTweet).fetchall()
print(countTweet)
countUser = cursor.execute(queryUser).fetchall()
print(countUser)
countGeo = cursor.execute(queryGeo).fetchall()
print(countGeo)

end1b50k=time.time()
t1b50k = ("Elapsed time for t1b50k is: " + str(end1b50k - start1b50k)+" seconds")
print(t1b50k)#16 seconds
```

```
Processed 49800 tweets
Processed 49900 tweets
[(49973,)]
[(48371,)]
[(1114,)]
Elapsed time for t1b50k is:
21.341416358947754 seconds
```

```
Processed 99700 tweets
Processed 99800 tweets
Processed 99900 tweets
[(99946,)]
[(95103,)]
[(3354,)]
Elapsed time for t1b100k is:
30.772485733032227 seconds
```

```
Processed 499800 tweets
Processed 499900 tweets
[(499776,)]
[(447304,)]
[(12962,)]
Elapsed time for 1b500k is:
164.46902179718018 seconds
```

c. Use your locally saved tweet file (created in part-a) to repeat the database population step from part-b. That is, load the tweets into the 3-table database using your saved file with tweets (do not use the URL to read twitter data here).

**Below format same for 100k & 500k**

```
Tweets50k = open("FinalTweets50k.txt",'r',encoding = "utf8")
import time
start1c50k=time.time()
import json
lstTweet = []
lstUser = []
lstGeo = []
rdFile = Tweets50k.readlines()#choose 50k,100k,500k text file
inecount=0
for tweet in rdFile:
    #linecount+=1
    #print(clean_tweet)
    #print(linecount)
    clean_tweet = json.loads(tweet,encoding = "utf-8")
    userDict=clean_tweet["user"]
    geoDict=clean_tweet["geo"]

    lstUser.append((userDict["id"],userDict["name"], userDict["screen_name"], userDict["description"],userDict["friends_count"]))

    if geoDict!=None:
        latitude = geoDict["coordinates"][0]
        longitude = geoDict["coordinates"][1]
        geo_id = str(longitude)+str(latitude)

        lstGeo.append((geo_id,geoDict['type'],longitude,latitude))
        lstTweet.append((clean_tweet["created_at"], str(clean_tweet["id_str"]), clean_tweet["text"], clean_tweet["source"], clean_tweet["in_reply_to_use

    else:

        lstTweet.append((clean_tweet["created_at"], str(clean_tweet["id_str"]), clean_tweet["text"], clean_tweet["source"], clean_tweet["in_reply_to_use


import sqlite3



sqlTweet = cursor.executemany("INSERT OR IGNORE INTO Tweet VALUES (?,?,?,?,?,?,?,?,?,?,?,?)",lstTweet)
sqlUser = cursor.executemany("INSERT OR IGNORE INTO User VALUES (?,?,?,?,?)",lstUser)
sqlGeo = cursor.executemany("INSERT OR IGNORE INTO Geo VALUES (?,?,?,?)",lstGeo)

queryTweet = '''SELECT COUNT(*) FROM Tweet'''
queryUser  ='''SELECT COUNT(*) FROM User'''
queryGeo = '''SELECT COUNT(*) FROM Geo'''


countTweet1 = cursor.execute(queryTweet).fetchall()
print(countTweet1)
countUser1 = cursor.execute(queryUser).fetchall()
print(countUser1)
countGeo1 = cursor.execute(queryGeo).fetchall()
print(countGeo1)

end1c50k=time.time()
t1c50k = ("Elapsed time for 1c50k is: " + str(end1c50k - start1c50k)+" seconds")
print(t1c50k)#5 seconds
```

50k

```
    ...:
    ...: countTweet =
cursor.execute(queryTweet).fetchall()
    ...: print(countTweet)
    ...: countUser =
cursor.execute(queryUser).fetchall()
    ...: print(countUser)
    ...: countGeo =
cursor.execute(queryGeo).fetchall()
    ...: print(countGeo)
    ...:
    ...: end=time.time()
    ...: t1c50k = ("Elapsed time for 1c50k is:
+ str(end - start)+" seconds")
    ...: print(t1c50k)#
[(49973,)]
[(48371,)]
[(1114,)]
Elapsed time for 1c50k is: 4.854224920272827
seconds
```

100k

```
cursor.execute(queryGeo).fetchall()
    ...: print(countGeo)
    ...:
    ...: end1c100k=time.time()
    ...: t1c100k = ("Elapsed time for
1c100k is: " + str(end1c100k - start1c100k)
+" seconds")
    ...: print(t1c100k)#19 seconds
[(99946,)]
[(95103,)]
[(2241,)]
Elapsed time for 1c100k is:
9.09029507637024 seconds

In [25]:
```

500k

```
    ...: t1c500k = ("Elapsed time for 1c500k is
" + str(end - start)+" seconds")
    ...: print(t1c500k)#13 seconds
[(499776,)]
[(447304,)]
[(11849,)]
Elapsed time for 1c500k is: 190.64131093025208
seconds
```

**Ignore the "#x seconds" copied and pasted from previous code**

d. Re-run the previous step with a batching size of 1000 (i.e. by inserting 1000 rows at a time with executemany).
How does the load runtime compare with and without batching?
**The batched run times run much quicker. From 50k to 500k – Times in seconds with batching =3,6,34 – With no Batching = 5,9,191**

```
    ...:
    ...:     linecount+=1
    ...:     if linecount<1000:
    ...:         cursor.executemany ('INSERT OR IGNORE INTO Tweet VALUES(?,?,?,?,?,?,?,?,?,?,?)
lstTweetBatch)
    ...:         cursor.executemany ('INSERT OR IGNORE INTO User VALUES(?,?,?,?,?)', lstUserBat
    ...:         cursor.executemany ('INSERT OR IGNORE INTO Geo VALUES(?,?,?,?)', lstGeoBatch)
    ...:         lstGeoBatch =[]
    ...:         lstUserBatch = [] # Reset the list of batched users
    ...:         lstTweetBatch =[] #reset
    ...:
    ...:
    ...: cursor.executemany ('INSERT OR IGNORE INTO Tweet VALUES(?,?,?,?,?,?,?,?,?,?,?)',
lstTweetBatch)
    ...: cursor.executemany ('INSERT OR IGNORE INTO User VALUES(?,?,?,?,?)', lstUserBatch)
    ...: cursor.executemany ('INSERT OR IGNORE INTO Geo VALUES(?,?,?,?)', lstGeoBatch)
    ...:
    ...:
    ...:
    ...: queryTweet = '''SELECT COUNT(*) FROM Tweet'''
    ...: queryUser  ='''SELECT COUNT(*) FROM User'''
    ...: queryGeo = '''SELECT COUNT(*) FROM Geo'''
    ...:
    ...:
    ...: countTweet = cursor.execute(queryTweet).fetchall()
    ...: print(countTweet)
    ...: countUser = cursor.execute(queryUser).fetchall()
    ...: print(countUser)
    ...: countGeo = cursor.execute(queryGeo).fetchall()
    ...: print(countGeo)
    ...: end1d50k=time.time()
    ...: t1d50k = ("Elapsed time for 1d50k is: " + str(end1d50k - start1d50k)+" seconds")
    ...: print(t1d50k)#
    ...: Tweets50k.close()
[(49973,)]
[(48371,)]
[(1114,)]
Elapsed time for 1d50k is: 3.147576332092285 seconds
```

Christian Craig
DSC 450 - Final

100k

```
    ...:         cursor.executemany ('INSERT OR IGNORE INTO Tweet VALUES(?,?,?,?,?,?,?,?,?,?,?)'
lstTweetBatch)
    ...:         cursor.executemany ('INSERT OR IGNORE INTO User VALUES(?,?,?,?,?)', lstUserBatcl
    ...:         cursor.executemany ('INSERT OR IGNORE INTO Geo VALUES(?,?,?,?)', lstGeoBatch)
    ...:         lstGeoBatch =[]
    ...:         lstUserBatch = [] # Reset the list of batched users
    ...:         lstTweetBatch =[] #reset
    ...:
    ...:
    ...: cursor.executemany ('INSERT OR IGNORE INTO Tweet VALUES(?,?,?,?,?,?,?,?,?,?,?)',
lstTweetBatch)
    ...: cursor.executemany ('INSERT OR IGNORE INTO User VALUES(?,?,?,?,?)', lstUserBatch)
    ...: cursor.executemany ('INSERT OR IGNORE INTO Geo VALUES(?,?,?,?)', lstGeoBatch)
    ...:
    ...:
    ...:
    ...: queryTweet = '''SELECT COUNT(*) FROM Tweet'''
    ...: queryUser  ='''SELECT COUNT(*) FROM User'''
    ...: queryGeo = '''SELECT COUNT(*) FROM Geo'''
    ...:
    ...:
    ...: countTweet = cursor.execute(queryTweet).fetchall()
    ...: print(countTweet)
    ...: countUser = cursor.execute(queryUser).fetchall()
    ...: print(countUser)
    ...: countGeo = cursor.execute(queryGeo).fetchall()
    ...: print(countGeo)
    ...: end1d100k=time.time()
    ...: t1d100k = ("Elapsed time for 1d100k is: " + str(end1d100k - start1d100k)+" seconds")
    ...: print(t1d100k)#
    ...: Tweets100k.close()
[(99946,)]
[(95103,)]
[(2241,)]
Elapsed time for 1d100k is: 6.437624216079712 seconds
```

Christian Craig
DSC 450 - Final

500k

```
    ...:     if tinecount<1000:
    ...:         cursor.executemany ('INSERT OR IGNORE INTO Tweet VALUES(?,?,?,?,?,?,?,?,?,?,?,?)
lstTweetBatch)
    ...:         cursor.executemany ('INSERT OR IGNORE INTO User VALUES(?,?,?,?,?)', lstUserBat
    ...:         cursor.executemany ('INSERT OR IGNORE INTO Geo VALUES(?,?,?,?)', lstGeoBatch)
    ...:         lstGeoBatch =[]
    ...:         lstUserBatch = [] # Reset the list of batched users
    ...:         lstTweetBatch =[] #reset
    ...:
    ...:
    ...: cursor.executemany ('INSERT OR IGNORE INTO Tweet VALUES(?,?,?,?,?,?,?,?,?,?,?,?)',
lstTweetBatch)
    ...: cursor.executemany ('INSERT OR IGNORE INTO User VALUES(?,?,?,?,?)', lstUserBatch)
    ...: cursor.executemany ('INSERT OR IGNORE INTO Geo VALUES(?,?,?,?)', lstGeoBatch)
    ...:
    ...:
    ...:
    ...: queryTweet = '''SELECT COUNT(*) FROM Tweet'''
    ...: queryUser  ='''SELECT COUNT(*) FROM User'''
    ...: queryGeo = '''SELECT COUNT(*) FROM Geo'''
    ...:
    ...:
    ...: countTweet = cursor.execute(queryTweet).fetchall()
    ...: print(countTweet)
    ...: countUser = cursor.execute(queryUser).fetchall()
    ...: print(countUser)
    ...: countGeo = cursor.execute(queryGeo).fetchall()
    ...: print(countGeo)
    ...: end=time.time()
    ...: t1d500k = ("Elapsed time for 1d500k is: " + str(end - start)+" seconds")
    ...: print(t1d500k)#40 seconds
    ...: Tweets500k.close()
    ...: #e
[(499776,)]
[(447304,)]
[(11849,)]
Elapsed time for 1d500k is: 34.95691418647766 seconds
```

e. Plot the resulting runtimes (# of tweets versus runtimes) using matplotlib for parts a-c.

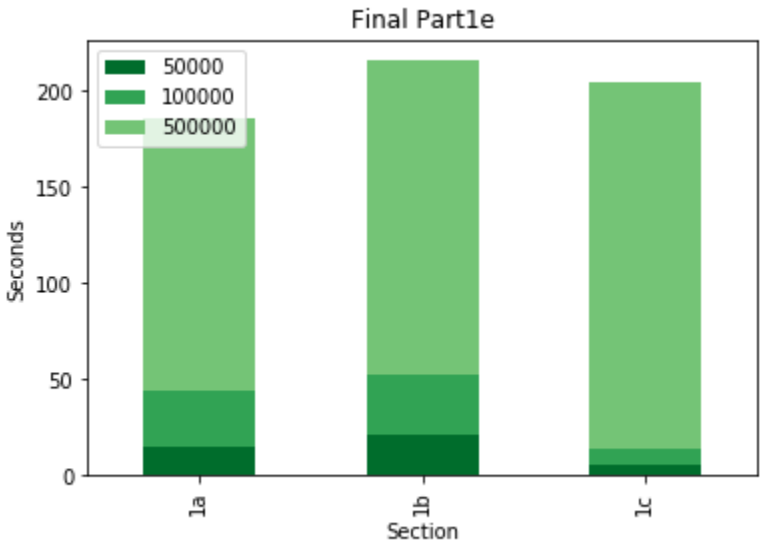
```
import matplotlib.pyplot as plt
import pandas as pd

data = {'Time (seconds)':[15,29,142,21,31,164,5,9,191],
        'Tweets (Thousands)':[50000,100000,500000,50000,100000,500000,50000,100000,500000],
        'Section':['1a','1a','1a','1b','1b','1b','1c','1c','1c']}
frame = pd.DataFrame(data)


def tweet_plot():

    pivot = frame.pivot(index = 'Section',columns ='Tweets (Thousands)',values = 'Time (seconds)' )
    pivot.loc[:,[50000,100000, 500000]].plot.bar(stacked=True, color=["#006D2C", "#31A354","#74C476"])
    plt.ylabel('Seconds')
    plt.xlabel('Section')
    plt.title('Final Part1e')
    plt.legend(loc=2)
    plt.show()

tweet_plot()
```



Christian Craig
DSC 450 - Final

# Part 2

a. Write and execute a SQL query to find the minimum longitude and latitude value for each user name.

```
In [51]: min_long_lat_user = '''SELECT name,min(longitude),min(latitude) FROM User,Geo,Tweet WHERE user
tweet.user_id AND geo.geo_id = tweet.geo_id GROUP BY user.name'''
    ...:
    ...: start2a=time.time()
    ...: lines = cursor.execute(min_long_lat_user).fetchall()
    ...: print(len(lines))
    ...: end2a=time.time()
    ...: t2a = ("Elapsed time for t2a is: " + str(end2a - start2a)+" seconds")
    ...: print(t2a)
11029
Elapsed time for t2a is: 3.2493107318878174 seconds
```

b. Re-execute the query in part 2-a 10 times and 100 times and measure the total runtime (just re-run the same exact query multiple times using a for-loop).
Does the runtime scale linearly? (i.e., does it take 10X and 100X as much time?)

**It scales linearly, because 322/31 = 10.38**

```
In [59]: import time
    ...: start = time.time()
    ...: for i in range(10):
    ...:     for geo_min in cursor.execute(min_long_lat_user).fetchall():
    ...:         # print(geo_min)
    ...:         linecount+=1
    ...:
    ...: print(linecount)
    ...: end = time.time()
    ...: t10 = ("Elapsed time for running min_long_lat_user x10 is: " + str(end - start)+" seconds")
    ...: print(t10) #53.74289107322693
198523
Elapsed time for running min_long_lat_user x10 is: 31.406943559646606 seconds

In [60]: start = time.time()
    ...: for i in range(100):
    ...:     for geo_min in cursor.execute(min_long_lat_user).fetchall():
    ...:         #print(geo_min)
    ...:         linecount+=1
    ...:
    ...: print(linecount)
    ...: end = time.time()
    ...: t100 = ("Elapsed time for running min_long_lat_user x100 is: " + str(end - start)+" seconds")
    ...: print(t100) #542.973828792572
1301423
Elapsed time for running min_long_lat_user x100 is: 322.1122658252716 seconds
```

c. Write the equivalent of the 2-a query in python (without using SQL) by reading it from the file with 500,000 tweets.

Christian Craig
DSC 450 - Final

```
In [62]: Tweets500k = open("FinalTweets500k.txt",'r',encoding = "utf8")
    ...: import json
    ...: longDict = {}
    ...: latDict = {}
    ...: newDict={}
    ...: linecount=0
    ...: rdFile = Tweets500k.readlines()#choose 500k text file
    ...: for tweet in rdFile:
    ...:     #linecount+=1
    ...:     #print(clean_tweet)
    ...:     #print(linecount)
    ...:     clean_tweet = json.loads(tweet,encoding = "utf-8")
    ...:     userDict=clean_tweet["user"]
    ...:     geoDict=clean_tweet["geo"]
    ...:     #lstUser.append((userDict["id"],userDict["name"], userDict["screen_name"],
userDict["description"],userDict["friends_count"]))
    ...:
    ...:     if geoDict != None and userDict["name"]!=None and userDict!=None:
    ...:         latitude = geoDict["coordinates"][0]
    ...:         longitude = geoDict["coordinates"][1]
    ...:
    ...:         if longitude!=None and latitude!=None:
    ...:             longDict.setdefault(userDict["name"],[]).append(longitude)
    ...:             latDict.setdefault(userDict["name"],[]).append(latitude)
    ...:
    ...: for i,j in longDict.items():
    ...:     for k,l in latDict.items():
    ...:         if i==k and i!=None and k!=None:
    ...:             newDict.setdefault(i,[]).append((min(j),min(l)))
    ...:             linecount+=1
    ...:
    ...: #print(newDict)
    ...: print(linecount)#11,036
11036
```

d. Re-execute the query in part 2-c 10 times and 100 times and measure the total runtime. Does the runtime scale linearly?

Its linear because 3914/385=10.16

```
In [74]: import time
    ...: start2d10=time.time()
    ...: for i in range(10):
    ...:     Tweets500k = open("FinalTweets500k.txt",'r',encoding = "utf8")
    ...:     import json
    ...:     longDict = {}
    ...:     latDict = {}
    ...:     newDict={}
    ...:     linecount=0
    ...:     rdFile = Tweets500k.readlines()#choose 500k text file
    ...:     for tweet in rdFile:
    ...:         #linecount+=1
    ...:         #print(clean_tweet)
    ...:         #print(linecount)
    ...:         clean_tweet = json.loads(tweet,encoding = "utf-8")
    ...:         userDict=clean_tweet["user"]
    ...:         geoDict=clean_tweet["geo"]
    ...:         #lstUser.append((userDict["id"],userDict["name"], userDict["screen_name"], userDict["
    ...:
    ...:         if geoDict != None and userDict["name"]!=None and userDict!=None:
    ...:             latitude = geoDict["coordinates"][0]
    ...:             longitude = geoDict["coordinates"][1]
    ...:
    ...:             if longitude!=None and latitude!=None:
    ...:                 longDict.setdefault(userDict["name"],[]).append(longitude)
    ...:                 latDict.setdefault(userDict["name"],[]).append(latitude)
    ...:     for i,j in longDict.items():
    ...:         for k,l in latDict.items():
    ...:             if i==k and i!=None and k!=None:
    ...:                 newDict.setdefault(i,[]).append((min(j),min(l)))
    ...:                 linecount+=1
    ...:     print(linecount)
    ...:
    ...: end2d10=time.time()
    ...: t2d10 = ("Elapsed time for running for 2dx10 is: " + str(end2d10 - start2d10)+" seconds")
    ...: print(t2d10)
11036
11036
11036

    ...:                 newDict.setdefault(i,[]).append((min(j),min(l)))
    ...:                 linecount+=1
    ...:     print(linecount)
    ...:
    ...: end2d10=time.time()
    ...: t2d10 = ("Elapsed time for running for 2dx10 is: " + str(end2d10 - start2d10)+" seconds")
    ...: print(t2d10)
11036
11036
11036
11036
11036
11036
11036
11036
11036
11036
Elapsed time for running for 2dx10 is: 385.699755191803 seconds
```

Christian Craig
DSC 450 - Final

```
In [78]: import time
    ...: start2d100=time.time()
    ...: for i in range(100):
    ...:     Tweets500k = open("FinalTweets500k.txt",'r',encoding = "utf8")
    ...:     import json
    ...:     longDict = {}
    ...:     latDict = {}
    ...:     newDict={}
    ...:     linecount=0
    ...:     rdFile = Tweets500k.readlines()#choose 500k text file
    ...:     for tweet in rdFile:
    ...:         #linecount+=1
    ...:         #print(clean_tweet)
    ...:         #print(linecount)
    ...:         clean_tweet = json.loads(tweet,encoding = "utf-8")
    ...:         userDict=clean_tweet["user"]
    ...:         geoDict=clean_tweet["geo"]
    ...:         #lstUser.append((userDict["id"],userDict["name"], userDict["screen_name"],
userDict["description"],userDict["friends_count"]))
    ...:
    ...:         if geoDict != None and userDict["name"]!=None and userDict!=None:
    ...:             latitude = geoDict["coordinates"][0]
    ...:             longitude = geoDict["coordinates"][1]
    ...:
    ...:             if longitude!=None and latitude!=None:
    ...:                 longDict.setdefault(userDict["name"],[]).append(longitude)
    ...:                 latDict.setdefault(userDict["name"],[]).append(latitude)
    ...:     for i,j in longDict.items():
    ...:         for k,l in latDict.items():
    ...:             if i==k and i!=None and k!=None:
    ...:                 newDict.setdefault(i,[]).append((min(j),min(l)))
    ...:                 linecount+=1
    ...:     print(linecount)
    ...:
    ...: end2d100=time.time()
    ...: t2d100 = ("Elapsed time for running for 2dx100 is: " + str(end2d100 - start2d100)+" seconds")
    ...: print(t2d100)
```

```
11036
11036
11036
11036
11036
11036
11036
11036
Elapsed time for running for 2dx100 is: 3914.230082988739 seconds
```

Christian Craig
DSC 450 - Final

# Part 3

a. Export the contents of the Tweet table (500k tweets) from SQLite table into a sequence of INSERT SQL statements. This is very similar to what you already did in an assignment. However, you have to resolve the problem of duplicate keys. If the key is unique, create the INSERT normally. If the key repeats, modify it to make it unique (e.g., by adding "_1" to disambiguate from the same key).
**Not sure if there is a typo in the problem. Since the primary eliminates duplicates, there are none. I added code to make unique ID's though.**

```
In [145]: import random
     ...: import string
     ...: queryTweet = '''SELECT * FROM Tweet'''
     ...: fdInsert = open('Insert500k.txt', 'w', encoding='utf-8') # no duplicates in text file
     ...: #will write code if needed unique vals, tagged on to tweet_id
     ...: for tweet in cursor.execute(queryTweet).fetchall():
     ...:     ltr=string.ascii_uppercase
     ...:     rand_ltrs = ''.join(random.choice(ltr) for i in range(6))
     ...:     primary = str(tweet[1])+'_'+rand_ltrs
     ...:     #not unique
     ...:     fdInsert.write('INSERT OR IGNORE INTO Tweet VALUES ({}{}{})
\n'.format(tweet[0],primary,tweet[2:]))
     ...:
     ...:
     ...:
     ...:
     ...: fdInsert.close()
```

b. Export all three tables (Tweet, User and Geo tables) from the largest (500k records) database into a |-separated text file – similar to csv but using "|" intead of a ",". In this part, you do not generate SQL INSERT statements, just the |-separated text data.

   i. For the Geo table, add a new column with a relative distance from a fixed point which is the location of CDM (41.878668, -87.625555). You can treat it as a Euclidean distance (although feel free to find a real distance in miles, if you wish).

```
In [140]: from math import cos, asin, sqrt, pi
     ...: CDM_latitude_pt=41.878668
     ...: CDM_longitude_pt=-87.625555
     ...: fd3bi = open('fd3bi.txt', 'w', encoding='utf-8')
     ...: for line in G:
     ...:     id=line[0]
     ...:     type=line[1]
     ...:     latitude=float(line[2])
     ...:     longitude=float(line[3])
     ...:     #below formulas from stackoverflow - https://stackoverflow.com/questions/27928/calculate-distance-between-two-latitude-longitude-points%20math%20import%20cos%2C%20asin,of%20completeness%3A%20Haversine%20on%20wiki
     ...:     p = pi/180
     ...:     a = 0.5 - cos((latitude-CDM_latitude_pt)*p)/2 + cos(CDM_latitude_pt*p) * cos(latitude*p) * (1-cos((longitude-latitude)*p))/2
     ...:     distance=12742 * asin(sqrt(a)) #6373.0*2=12742
     ...:
     ...:
     ...:
     ...:     fd3bi.write('{}|{}|{}|{}|{}\n'.format(id,type,latitude,longitude,distance))
     ...:
     ...:
     ...:
     ...: fd3bi.close()
```

```
121.04395514.670275|Point|121.043955|14.670275|5241.956356170689
110.213471-7.351872|Point|110.213471|-7.351872|4647.8019423829455
-122.22247.8487|Point|-122.222|47.8487|11119.765916660894
-77.15961738.767654|Point|-77.159617|38.767654|15158.102047558794
106.728999-6.149429|Point|106.728999|-6.149429|4862.378390269698
-1.50507852.536283|Point|-1.505078|52.536283|7249.418680835565
127.69884426.198516|Point|127.698844|26.198516|5755.343787438886
-115.17028636.109317|Point|-115.170286|36.109317|12126.249095291472
-46.288466-23.953996|Point|-46.288466|-23.953996|10049.68110423536
-51.216072-29.188215|Point|-51.216072|-29.188215|10569.123711881564
-77.35998134.771422|Point|-77.359981|34.771422|15062.428110493413
-57.612373-25.299878|Point|-57.612373|-25.299878|11464.204517156211
-43.288543-22.678829|Point|-43.288543|-22.678829|9691.655905012376
101.4513660.442102|Point|101.451366|0.442102|5221.825181628136
```

ii. For the Tweet table, add a new column with data from the User table (“screen_name”) in addition to existing columns. You do not need to modify the original table in SQLite, although you can if you wish.

```
In [142]: query3bii = '''SELECT
tweet.created_at,tweet.tweet_id,tweet.text,tweet.source,tweet.in_reply_to_user_id,tweet.in_reply_to_screen_name,tweet.in_reply_to_status_id,tweet.retweet_count,tweet.
er_id,tweet.geo_id,user.screen_name FROM Tweet,User WHERE user.id = tweet.user_id'''
     ...: fd3bii = open('fd3bii.txt', 'w', encoding='utf-8')
     ...: for tweet in cursor.execute(query3bii).fetchall():
     ...:     #defs for reference
     ...:     created_at = tweet[0]
     ...:     tweet_id = tweet[1]
     ...:     text = tweet[2]
     ...:     source = tweet[3]
     ...:     in_reply_to_user_id = tweet[4]
     ...:     in_reply_to_screen_name = tweet[5]
     ...:     in_reply_to_status_id = tweet[6]
     ...:     retweet_count = tweet[7]
     ...:     contributors = tweet[8]
     ...:     user_id = tweet[9]
     ...:     geo_id = tweet[10]
     ...:     screen_name=tweet[11]
     ...:     fd3bii.write('{}|{}|{}|{}|{}|{}|{}|{}|{}|{}|{}|{}|{}
\n'.format(created_at,tweet_id,text,source,in_reply_to_user_id,in_reply_to_screen_name,in_reply_to_status_id,retweet_count,contributors,user_id,geo_id,screen_name))
```

```
~John Ruskin #wisdomink|<a href="http://www.hootsuite.com" rel="nofollow">HootSuite</a>|None|None|None|0|None|213646047|None|Wisdom_Ink
aquello, pero de los viernes es la fiesta con la gente más linda.
|None|firekites
a finds its way into such heads,|<a href="http://eto-secret4.ru" rel="nofollow">eto prosto NEW secret</a>|None|None|None|0|None|2443526930|None|rosaxonahag
:tp://twitter.com/download/android" rel="nofollow">Twitter for Android</a>|290322075|im_2realbih|471800733541486600|0|None|284813188|None|_SameOleSHIT
, cuando, donde , como y ahora que hago, por que . Dios es perfecto y a veces... http://t.co/iCyBxph8s9|<a href="http://www.facebook.com/twitter" rel="nofollow">Facebook</a
or queres ver si se podia grabar lpm :#|web|None|None|None|0|None|1146905466|None|NatyEmeri
```

iii. For the User table file add a column that specifies how many tweets by that user are currently in the database. You do not need to modify the User table, just the contents of the exported User text file.

```
In [144]: query3biii = '''SELECT
user.id,user.name,user.name,user.screen_name,user.friend_count,COUNT(tweet.user_id) FROM USER,Tweet WHE
user.id = tweet.user_id GROUP BY user.id'''
     ...: fd3biii = open('fd3biii.txt', 'w', encoding='utf-8')
     ...: for user in cursor.execute(query3biii).fetchall():
     ...:     #defs for reference
     ...:     id = user[0]
     ...:     name = user[1]
     ...:     screen_name=user[2]
     ...:     description = user[3]
     ...:     friend_count = user[4]
     ...:     fd3biii.write('{}|{}|{}|{}|{}|{}
\n'.format(id,name,text,screen_name,description,friend_count,count))
     ...:
     ...:
     ...: fd3biii.close()
```

```
1000003303|Kanddy---(^-^)--- |Me comere un gusanito porque a mi nadie me quiere:(.|Kanddy---(^-^)--- |Kanddyxx|47
100001424|dianne((the potato))|Me comere un gusanito porque a mi nadie me quiere:(.|dianne((the potato))|horanninja|586
1000018914|kristina|Me comere un gusanito porque a mi nadie me quiere:(.|kristina|kristinaernstt|205
1000020805|MIA WARSTLER|Me comere un gusanito porque a mi nadie me quiere:(.|MIA WARSTLER|miawarstler|223
100003178|Goddy |Me comere un gusanito porque a mi nadie me quiere:(.|Goddy |gxddy|1741
100003573|Khairyfornia™|Me comere un gusanito porque a mi nadie me quiere:(.|Khairyfornia™|khairy30|1206
1000044026||Me comere un gusanito porque a mi nadie me quiere:(.||hunterismyhero|4686
1000048921|,|Me comere un gusanito porque a mi nadie me quiere:(.|,|baby_seff|433
1000057837|Lindsy Butina|Me comere un gusanito porque a mi nadie me quiere:(.|Lindsy Butina|lindsy_butina|473
1000062391|Amanda|Me comere un gusanito porque a mi nadie me quiere:(.|Amanda|ALambertsHair|576
1000071432|Silver Hedgie (Sick)|Me comere un gusanito porque a mi nadie me quiere:(.|Silver Hedgie (Sick)|NaiveHedgie|854
100008258|LENJI|Me comere un gusanito porque a mi nadie me quiere:(.|LENJI|LENJI1059|425
1000085868|Camillionaire|Me comere un gusanito porque a mi nadie me quiere:(.|Camillionaire|camsilvera|400
100008869|angel marlin|Me comere un gusanito porque a mi nadie me quiere:(.|angel marlin|angelmarlin|353
100009584|Pau!!!|Me comere un gusanito porque a mi nadie me quiere:(.|Pau!!!|PolitasM|191
1000112702|Bertha Arreola|Me comere un gusanito porque a mi nadie me quiere:(.|Bertha Arreola|baarreola|102
100012761|Lucia Moreno D.|Me comere un gusanito porque a mi nadie me quiere:(.|Lucia Moreno D.|R8nita|1086
1000129837|@ngie|Me comere un gusanito porque a mi nadie me quiere:(.|@ngie|AguiarAngelina|180
1000142653|Jenna|Me comere un gusanito porque a mi nadie me quiere:(.|Jenna|jennalabbe10|201
1000143158|TeeJay CrDwford|Me comere un gusanito porque a mi nadie me quiere:(.|TeeJay CrDwford|TeeJayCrawford|294
1000147832|Mystique|Me comere un gusanito porque a mi nadie me quiere:(.|Mystique|irisiri8|632
```