DSC 450: Database Processing for Large-Scale Analytics

Take-home Final

Student Name: Ronaldlee Ejalu

Part 1

We will use one full day worth of tweets as our input (there are total of 4.4M tweets in this file, but we will intentionally use fewer tweets to run this final):
http://dbgroup.cdm.depaul.edu/DSC450/OneDayOfTweets.txt
Execute the following tasks with 50,000 tweets and 500,000 tweets.

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). This is as simple as it sounds, all you need is a for-loop that reads lines and writes them into a file, just don't forget to add '\n' at the end so they are, in fact, on separate lines.

   NOTE: Do not call read() or readlines(). That command will attempt to read the entire file which is too much data. Clicking on the link in the browser would cause the same problem.

```
# Author: Ronaldlee Ejalu
# Course DSC 450
# 1a
"""
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).
This is as simple as it sounds, all you need is a for-loop that
reads lines and writes them into a file, just don't forget to add
'\n' at the end so they are, in fact, on separate lines.
NOTE: Do not call read() or readlines().
That command will attempt to read the entire file which is too much
data. Clicking on the link in the browser would cause the same prob
"""


import urllib.request
import json
import os
import csv
import time

os.chdir('C:/Users/rejalu1/OneDrive - Henry Ford Health
System/DSC450/Assignments/FinalExamTakeHome')

tweetdata =
"""http://dbgroup.cdm.depaul.edu/DSC450/OneDayOfTweets.txt"""
startTime = time.time()
# start time of processing the file in web

webFD = urllib.request.urlopen(tweetdata)
```

```
# csvf = open('OneDayOfTweets.csv', 'w', newline = '\n', encoding =
'utf-8')
csvf = open('OneDayOfTweets.csv', 'wb')

for i in range(500000):
    if i % 5000 == 0: # Print a message every 500th tweet read
        print ("Processed " + str(i) + " tweets")
    try:
        itemResponse = webFD.readline()
# read one line at a time
        # strItemResponse = itemResponse.decode('utf-8')
# decode the line that comes back from the web into a string.
        csvf.write(itemResponse)

    except Exception:
        continue

csvf.close()
# close the file
# csve.close()
# close the error file
endTime = time.time()
# end time of processing of writing the tweets data to a file.
print('The processing of the tweets data took %s seconds' %(endTime-
startTime))
print('The number of operations per second is %s seconds'
%(500000/(endTime-startTime)))
```

The screenshot of the run time when processing 500,000 tweets:

C: > Users > rejalu1 > OneDrive - Henry Ford Health System > DSC450 > Assignments > FinalExamTakeHome > final1a.py > ...

```
21  startTime = time.time()                                    # start time of processing the file i
22
23  webFD = urllib.request.urlopen(tweetdata)
24  # csvf = open('OneDayOfTweets.csv', 'w', newline = '\n', encoding = 'utf-8')
25  csvf = open('OneDayOfTweets.csv', 'wb')
26
27  for i in range(500000):
28      if i % 5000 == 0: # Print a message every 500th tweet read
29          print ("Processed " + str(i) + " tweets")
30      try:
31          itemResponse = webFD.readline()                    # read one line at a time
```

PROBLEMS 27 OUTPUT DEBUG CONSOLE TERMINAL JUPYTER Python

```
Processed 460000 tweets
Processed 465000 tweets
Processed 470000 tweets
Processed 475000 tweets
Processed 480000 tweets
Processed 485000 tweets
Processed 490000 tweets
Processed 495000 tweets
The processing of the tweets data took 2503.7961547374725 seconds
The number of operations per second is 199.69676806713758 seconds
PS C:\Users\rejalu1>
```

The screenshot of the run time when processing 50,000 tweets:

C: > Users > rejalu1 > OneDrive - Henry Ford Health System > DSC450 > Assignments > FinalExamTakeHome > final1a.py > ...

```python
startTime = time.time()                                                    # start time of processing the file i

webFD = urllib.request.urlopen(tweetdata)
# csvf = open('OneDayOfTweets.csv', 'w', newline = '\n', encoding = 'utf-8')
csvf = open('OneDayOfTweets.csv', 'wb')

for i in range(50000):
    if i % 5000 == 0: # Print a message every 500th tweet read
        print ("Processed " + str(i) + " tweets")
    try:
        itemResponse = webFD.readline()                                    # read one line at a time
        # strItemResponse = itemResponse.decode('utf-8')                     # decode the line that comes back f
        csvf.write(itemResponse)

    except Exception:
```

PROBLEMS 27 OUTPUT DEBUG CONSOLE TERMINAL JUPYTER Python

```
Processed 35000 tweets
Processed 40000 tweets
Processed 45000 tweets
The processing of the tweets data took 288.94251561164856 seconds
The number of operations per second is 1730.4480060387582 seconds
PS C:\Users\rejalu1>
```

b. For *text*, *in_reply_to_user_id* and *in_reply_to_screenname* in Tweet table and for *screen_name* in User table, find the length of the longest string in the file in 1-a and compare it to your data type size (you only have to change your table if the data type turns out to be too short). You only need to do 1-b for 500,000 tweets file and you do not need to use SQLite database.

```
# Author Ronaldlee Ejalu
# Course DSC 450
# 1b
import pandas as pd
import os
import csv
import time
import json
# 1b

fileName = 'C:/Users/rejalu1/OneDrive -
 Henry Ford Health System/DSC450/Assignments/FinalExamTakeHome/OneDayOfTweets.csv
'


def extractLine():
    """Reading the file in chunks"""
    with open(fileName, 'rb') as f:
        for item in f:
            yield item

startTime = time.time()
chunkSize = 500000
generatedLines = extractLine()                              # invoke a helper ext
ractLine
screenNameDict = {}
chunk = [i for i, j in zip(generatedLines, range(chunkSize))]

inReplytoUserIdL = [] # hold in_reply_to_user_id id values

inReplyToScreenNameL = [] # hold in_reply_to_screen_name values


for i in range(500000):
    if i % 5000 == 0: # Print a message every 500th tweet read
        print ('Processed ' + str(i) + ' tweets')
    fileDict = json.loads(chunk[i].decode('utf-
8')) # using decode() and loads to convert each item to a dictionary
    if fileDict['user']['screen_name'] not in screenNameDict.values():
        screenNameDict[i] = fileDict['user']['screen_name']

    if fileDict['in_reply_to_user_id'] not in inReplytoUserIdL:
        inReplytoUserIdL.append(fileDict['in_reply_to_user_id'])

    if fileDict['in_reply_to_screen_name'] not in inReplyToScreenNameL:
        inReplyToScreenNameL.append(fileDict['in_reply_to_screen_name'])
```

The screenshot of the output after running the above script:

C: > Users > rejalu1 > OneDrive - Henry Ford Health System > DSC450 > Assignments > FinalExamTakeHome > Final.py > ...

```python
        screenNameDict[i] = fileDict['user']['screen_name']

    if fileDict['in_reply_to_user_id'] not in inReplytoUserIdL:
        inReplytoUserIdL.append(fileDict['in_reply_to_user_id'])

    if fileDict['in_reply_to_screen_name'] not in inReplyToScreenNameL:
        inReplyToScreenNameL.append(fileDict['in_reply_to_screen_name'])

# print(screenNameDict)       # for debugging purposes

screenNameDf = pd.DataFrame(screenNameDict.values(), columns=['ScreenName'])                    # generate the ScreenName DataFrame
inReplytoUserIdDf = pd.DataFrame(inReplytoUserIdL, columns=['in_reply_to_user_id'])              # generate the inReplytoUserIdDf
inReplyToScreenNameDf = pd.DataFrame(inReplyToScreenNameL, columns=['in_reply_to_screen_name'])

# print(screenNameDf.head(10))
print('The length of the longest string in the file for the in_reply_to_user_id column is %s'%(inReplytoUserIdDf.in_reply_to_user_id.fillna(method='ffill')
print('The length of the longest string in the file for the in_reply_to_screen_name column is %s'%(inReplyToScreenNameDf.in_reply_to_screen_name.fillna(metl
print('The length of the longest string in the file for the ScreenName column is %s'%(screenNameDf.ScreenName.fillna(method='ffill').astype(str).str.len().r
endTime = time.time()
print('The processing of the tweets data took %s seconds' %(endTime-startTime))
```

PROBLEMS 5 OUTPUT DEBUG CONSOLE TERMINAL JUPYTER Pytho

```
Processed 460000 tweets
Processed 465000 tweets
Processed 470000 tweets
Processed 475000 tweets
Processed 480000 tweets
Processed 485000 tweets
Processed 490000 tweets
Processed 495000 tweets
The length of the longest string in the file for the in_reply_to_user_id column is 12
The length of the longest string in the file for the in_reply_to_screen_name column is 15
The length of the longest string in the file for the ScreenName column is 15
The processing of the tweets data took 6834.749411582947 seconds
```

c. Repeat what you did in part 1-a, but instead of saving tweets to the file, populate the 3-table schema that you previously 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.

NOTE: If your schema contains a foreign key in the Geo table or relies on TweetID as the primary key for the Geo table, you should fix your schema. Geo entries should be identified based on the location they represent. There should **not** be any "blank" Geo entries such as (ID, None, None, None).

Below is the Python Code:

```python
# Author: Ronaldlee Ejalu
# Course DSC 540
# Final Take home exam
# reading a collection of tweets data
# extending the schema by adding Geo table
# also, added a foreign key relationship between Geo and Tweets.
# Part 1 c

import urllib.request
import json
import re
import sqlite3
import os
import csv
import time

os.chdir('C:/Users/rejalu1/OneDrive -
 Henry Ford Health System/DSC450/Assignments/FinalExamTakeHome')

tweetdata = """http://dbgroup.cdm.depaul.edu/DSC450/OneDayOfTweets.txt"""


createTbl1 = """
CREATE TABLE UserTweets
(
    Id VARCHAR2(100),
    name VARCHAR2(100),
    screen_name VARCHAR2(15),
    description VARCHAR2(500),
    friends_count NUMBER,
    CONSTRAINT UserTweets_PK Primary Key(Id)
);
"""

createTbl2 = """
CREATE TABLE Tweets
(   CREATED_AT DATE,
    ID VARCHAR2(100),
    TEXT VARCHAR2(300),
    SOURCE VARCHAR2(100),
    IN_REPLY_TO_USER_ID VARCHAR2(12),
    IN_REPLY_TO_SCREEN_NAME VARCHAR2(15),
    IN_REPLY_TO_STATUS_ID VARCHAR2(100),
    RETWEET_COUNT NUMBER,
    CONTRIBUTORS VARCHAR2(100),
```

A screen shot showing the run time of 500,000 tweets being processed when reading the tweet data from the given web URL:

C: > Users > rejalu1 > OneDrive - Henry Ford Health System > DSC450 > Assignments > FinalExamTakeHome > Final1cver1.py > ...

```python
# Author: Ronaldlee Ejalu
# Course DSC 540
# Final Take home exam
# reading a collection of tweets data
# extending the schema by adding Geo table
# also, added a foreign key relationship between Geo and Tweets.
# Part 1 c

import urllib.request
import json
import re
import sqlite3
import os
import csv
import time
```

PROBLEMS 27 OUTPUT DEBUG CONSOLE TERMINAL JUPYTER Python

```
Processed 495000 tweets
The number of records in the userTweets table are [(447304,)]
The number of records in the Tweets table are [(500000,)]
The number of records in the Geo table are [(11849,)]
The processing of the tweets data took 3012.9960594177246 seconds
PS C:\Users\rejalu1>
```

A screenshot showing the runtime of 50,000 tweets being processed when reading the tweet data from the given web URL:

C: > Users > rejalu1 > OneDrive - Henry Ford Health System > DSC450 > Assignments > FinalExamTakeHome > Final1cver1.py > ...

```python
# Author: Ronaldlee Ejalu
# Course DSC 540
# Final Take home exam
# reading a collection of tweets data
# extending the schema by adding Geo table
# also, added a foreign key relationship between Geo and Tweets.
# Part 1 c

import urllib.request
import json
import re
import sqlite3
```

PROBLEMS 27 OUTPUT DEBUG CONSOLE TERMINAL JUPYTER Python

```
Processed 30000 tweets
Processed 35000 tweets
Processed 40000 tweets
Processed 45000 tweets
The number of records in the userTweets table are [(48371,)]
The number of records in the Tweets table are [(50000,)]
The number of records in the Geo table are [(1114,)]
The processing of the tweets data took 279.7146816253662 seconds
The number of operations per second is 1787.5357742918598 seconds
PS C:\Users\rejalu1>
```

d. Use your locally saved tweet file to repeat the database population step from part-c. That is, load the tweets into the 3-table database using your saved file with tweets. This is the **same** code as in 1-c, but reading tweets from your file, not from the web.

```python
# Author: Ronaldlee Ejalu
# Course DSC 540
# Take Home Exam
# reading a collection of tweets data from the local file system
# extending the schema by adding Geo table
# also, added a foreign key relationship between Geo and Tweets.
# Part 1 d
import urllib.request
import json
import re
import sqlite3
import os
import csv
import time

os.chdir('C:/Users/rejalu1/OneDrive -
 Henry Ford Health System/DSC450/Assignments/FinalExamTakeHome')

fileName = 'C:/Users/rejalu1/OneDrive -
 Henry Ford Health System/DSC450/Assignments/FinalExamTakeHome/OneDayOfTweets.csv
'

createTbl1 = """
CREATE TABLE UserTweets
(
    Id VARCHAR2(100),
    name VARCHAR2(100),
    screen_name VARCHAR2(15),
    description VARCHAR2(500),
    friends_count NUMBER,
    CONSTRAINT UserTweets_PK Primary Key(Id)
);
"""

createTbl2 = """
CREATE TABLE Tweets
(   CREATED_AT DATE,
    ID VARCHAR2(100),
    TEXT VARCHAR2(300),
    SOURCE VARCHAR2(100),
    IN_REPLY_TO_USER_ID VARCHAR2(12),
    IN_REPLY_TO_SCREEN_NAME VARCHAR2(15),
    IN_REPLY_TO_STATUS_ID VARCHAR2(100),
    RETWEET_COUNT NUMBER,
    CONTRIBUTORS VARCHAR2(100),
```

A screen shot showing the run time of 500,000 tweets being processed when reading the tweet data from the local file system:

C: > Users > rejalu1 > OneDrive - Henry Ford Health System > DSC450 > Assignments > FinalExamTakeHome > Final1d.py > ...

```python
# Author: Ronaldlee Ejalu
# Course DSC 540
# Take Home Exam
# reading a collection of tweets data from the local file system
# extending the schema by adding Geo table
# also, added a foreign key relationship between Geo and Tweets.
# Part 1 d
import urllib.request
import json
import re
import sqlite3
import os
import csv
import time
```

PROBLEMS 23 OUTPUT DEBUG CONSOLE TERMINAL JUPYTER Python

```
Processed 485000 tweets
Processed 490000 tweets
Processed 495000 tweets
The number of records in the userTweets table are [(447304,)]
The number of records in the Tweets table are [(500000,)]
The number of records in the Geo table are [(11849,)]
The processing of the tweets data took 172.24891209602356 seconds
The number of operations per second is 2902.775953216268 seconds
PS C:\Users\rejalu1>
```

A screen shot showing the run time of 50,000 tweets being processed when reading the tweet data from the local file system:

C: > Users > rejalu1 > OneDrive - Henry Ford Health System > DSC450 > Assignments > FinalExamTakeHome > Final1d.py > ...

```python
# Author: Ronaldlee Ejalu
# Course DSC 540
# Take Home Exam
# reading a collection of tweets data from the local file system
# extending the schema by adding Geo table
# also, added a foreign key relationship between Geo and Tweets.
# Part 1 d
import urllib.request
import json
import re
import sqlite3
import os
import csv
```

PROBLEMS 23 OUTPUT DEBUG CONSOLE TERMINAL JUPYTER Python

```
Processed 35000 tweets
Processed 40000 tweets
Processed 45000 tweets
The number of records in the userTweets table are [(48371,)]
The number of records in the Tweets table are [(50000,)]
The number of records in the Geo table are [(1114,)]
The processing of the tweets data took 14.039840698242188 seconds
The number of operations per second is 35612.939686886966 seconds
PS C:\Users\rejalu1>
```

e. Repeat the same step with a batching size of 2000 (i.e. by inserting 2000 rows at a time with executemany instead of doing individual inserts). Since many of the tweets are missing a Geo location, its fine for the batches of Geo inserts to be smaller than 2000.

```
# Author: Ronaldlee Ejalu
# Course DSC 540
# Take home exam
# reading a collection of tweets data from a file
# and using a batch size of 2000 to perform bulk inserts
# into the sqlite database
# extending the schema by adding Geo table
# also, added a foreign key relationship between Geo and Tweets.
# Part 1 e

import urllib.request
import json
import re
import sqlite3
import os
import csv
import time

os.chdir('C:/Users/rejalu1/OneDrive -
 Henry Ford Health System/DSC450/Assignments/FinalExamTakeHome')

fileName = 'C:/Users/rejalu1/OneDrive -
 Henry Ford Health System/DSC450/Assignments/FinalExamTakeHome/OneDayOfT
'

createTbl1 = """
CREATE TABLE UserTweets
(
    Id VARCHAR2(100),
    name VARCHAR2(100),
    screen_name VARCHAR2(15),
    description VARCHAR2(500),
    friends_count NUMBER,
    CONSTRAINT UserTweets_PK Primary Key(Id)
);
"""

createTbl2 = """
CREATE TABLE Tweets
(   CREATED_AT DATE,
    ID VARCHAR2(100),
    TEXT VARCHAR2(300),
    SOURCE VARCHAR2(100),
    IN_REPLY_TO_USER_ID VARCHAR2(12),
    IN_REPLY_TO_SCREEN_NAME VARCHAR2(15),
```

A screen shot showing the run time of 500,000 tweets being processed when a batch size of 2000 records are inserted into sqlite using bulk insert:

C: > Users > rejalu1 > OneDrive - Henry Ford Health System > DSC450 > Assignments > FinalExamTakeHome > Final1e.py > ...

```python
# Author: Ronaldlee Ejalu
# Course DSC 540
# Take home exam
# reading a collection of tweets data from a file
# and using a batch size of 2000 to perform bulk inserts
# into the sqlite database
# extending the schema by adding Geo table
# also, added a foreign key relationship between Geo and Tweets.
# Part 1 e

import urllib.request
import json
import re
```

PROBLEMS 23 OUTPUT DEBUG CONSOLE TERMINAL JUPYTER Python

```
Processed 494000 tweets
Processed 496000 tweets
Processed 498000 tweets
The number of records in the userTweets table are [(447304,)]
The number of records in the Tweets table are [(500000,)]
The number of records in the Geo table are [(11849,)]
The processing of the tweets data took 173.3715558052063 seconds
The number of operations per second is 2883.9794260240765 seconds
PS C:\Users\rejalu1>
```

A screen shot showing the run time of 50,000 tweets being processed when a batch size of 2000 records are inserted into sqlite using the bulk insert operation:

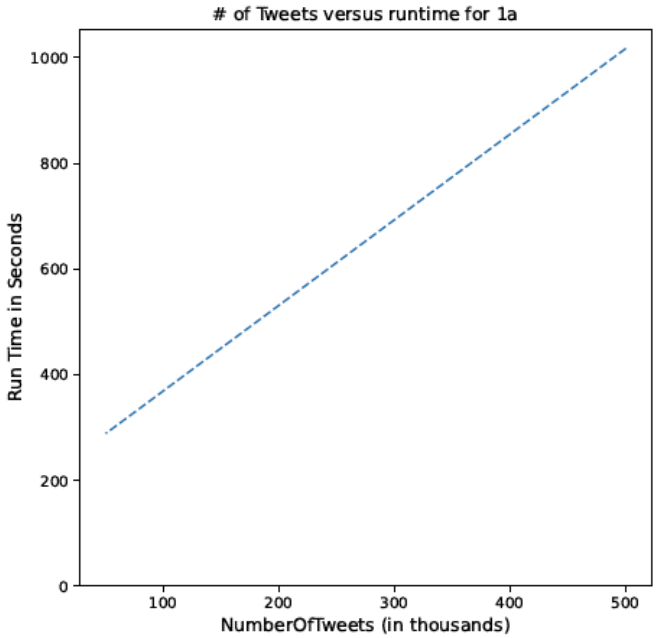
C: > Users > rejalu1 > OneDrive - Henry Ford Health System > DSC450 > Assignments > FinalExamTakeHome > Final1e.py > ...

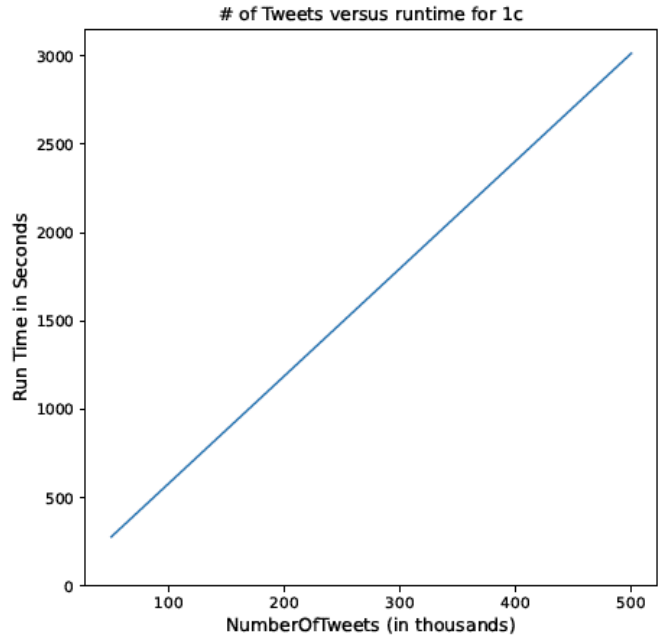
```python
# Author: Ronaldlee Ejalu
# Course DSC 540
# Take home exam
# reading a collection of tweets data from a file
# and using a batch size of 2000 to perform bulk inserts
# into the sqlite database
# extending the schema by adding Geo table
# also, added a foreign key relationship between Geo and Tweets.
# Part 1 e

import urllib.request
import json
import re
```

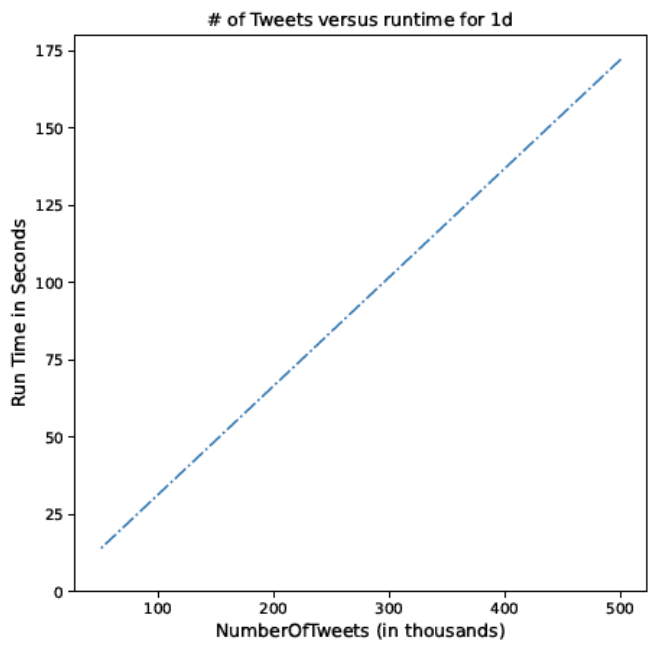
PROBLEMS 23 OUTPUT DEBUG CONSOLE TERMINAL JUPYTER Python

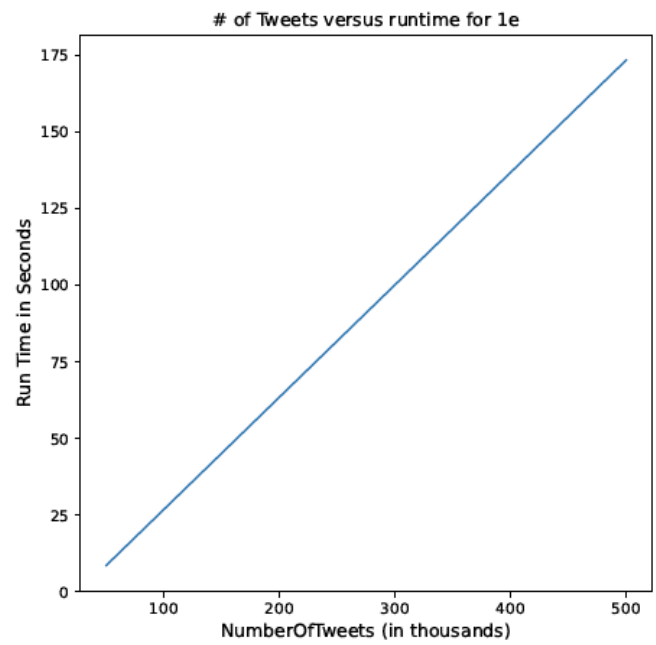
```
Processed 44000 tweets
Processed 46000 tweets
Processed 48000 tweets
The number of records in the userTweets table are [(48371,)]
The number of records in the Tweets table are [(50000,)]
The number of records in the Geo table are [(1114,)]
The processing of the tweets data took 8.637104034423828 seconds
The number of operations per second is 5788.977393432016 seconds
PS C:\Users\rejalu1>
```

f. Plot the resulting runtimes (# of tweets versus runtimes) using matplotlib for 1-a, 1-c, 1-d, and 1-e. How does the runtime compare?









The python code for 1f is below:

```
# Ronaldlee Ejalu
# DSC 450
# Final Take home exam
# plotting the resulting runtime (# of tweets versus run times)
# using matplotlib for 1-a, 1-c, 1-d, and 1-e
# 1f
import pandas as pd
import matplotlib.pyplot as plt
import os

os.chdir('C:/Users/rejalu1/OneDrive -
 Henry Ford Health System/DSC450/Assignments/FinalExamTakeHome')

# deriving the data structures for the runtimes of the number of tweets for the
# questions 1a, 1c, 1d, and 1e
dataForA = [(500, 1016.9364602565765), (50, 288.94251561164856)]
dataForC = [(500, 3012.9960594177246), (50, 279.7146816253662)]
dataForD = [(500, 172.24891209602356), (50, 14.039840698242188)]
dataForE = [(500, 173.3715558052063), (50, 8.637104034423828)]

# derive the different data frames
dfRunTimeA = pd.DataFrame(dataForA, columns=['NumberOfTweets', 'RunTimeinSecs'])
dfRunTimeC = pd.DataFrame(dataForC, columns=['NumberOfTweets', 'RunTimeinSecs'])
dfRunTimeD = pd.DataFrame(dataForD, columns=['NumberOfTweets', 'RunTimeinSecs'])
dfRunTimeE = pd.DataFrame(dataForE, columns=['NumberOfTweets', 'RunTimeinSecs'])
print(dfRunTimeE)

fig = plt.figure()                                          # create a blank figu
re

# add a varierty of sub plots to it,
# the first two parameters describe the size of the grid
# that corresponds to the sub figures being added.
# 2 by 2 means we have a grid of four different sub plots
# and we are going to use that to compare and contrast the styles of the differen
t figures.
# the last parameter refers to which of the sub plots we are adding to.
# so one refers to the top left of the first out of the four subplots in the figu
re

sp = fig.add_subplot(2, 2, 1)                                   # Add a grid of
4 subplots
fig.set_size_inches(15, 15)                                     # set the figure
 size in inches.
```

Part 2

a. Write and execute a SQL query to find the smallest longitude and latitude value for each user ID. This query does not need the User table because User ID is a foreign key in the Tweet table. E.g., something like *SELECT UserID, MIN(longitude), MIN(latitude) FROM Tweet, Geo WHERE Tweet.GeoFK = Geo.GeoID;*

```python
# Author: Ronaldlee Ejalu
# Course DSC 540
# Final Exam
# a SQL query that finds the smallest
# longitude and latitude value for each user ID
# Part 2a
import re
import sqlite3
import os
import time

os.chdir('C:/Users/rejalu1/OneDrive -
 Henry Ford Health System/DSC450/Assignments/FinalExamTakeHome')
# connect to the database file to open a connection
conn = sqlite3.connect('dsc450.db')                                    # open the c
onnection
cur = conn.cursor()                                                      # instanti
ate a cursor object
# construct a sql query and assign it to a string variable
sqlScript = """
SELECT Tweets.User_Id,
MIN(Geo.longitude),
MIN(latitude)
FROM Tweets, Geo
WHERE Tweets.GeoId = Geo.Id
GROUP BY Tweets.User_Id;
"""
userIdRes = cur.execute(sqlScript).fetchall()
print('The results of query %s are: /n %s ' %(sqlScript, userIdRes))
print('%s rows are returned' %(len(userIdRes)))
cur.close()
```

The screenshot of the python execution with the SQL script is below:

C: > Users > rejalu1 > OneDrive - Henry Ford Health System > DSC450 > Assignments > FinalExamTakeHome > final2a.py > ...

```python
# Part 2a
import re
import sqlite3
import os
import time

os.chdir('C:/Users/rejalu1/OneDrive - Henry Ford Health System/DSC450/Assignments/FinalExamTakeHome')
# connect to the database file to open a connection
conn = sqlite3.connect('dsc450.db')                                # open the connection
cur = conn.cursor()                                                 # instantiate a cursor object
# construct a sql query and assign it to a string variable
sqlScript = """
SELECT Tweets.User_Id,
MIN(Geo.longitude),
MIN(latitude)
FROM Tweets, Geo
WHERE Tweets.GeoId = Geo.Id
GROUP BY Tweets.User_Id;
"""
userIdRes = cur.execute(sqlScript).fetchall()
print('The results of query %s are: /n %s ' %(sqlScript, userIdRes))
print('%s rows are returned' %(len(userIdRes)))
```

PROBLEMS 3 OUTPUT DEBUG CONSOLE TERMINAL JUPYTER Python

```
), ('985833864', 41.617828, -85.913666), ('98644594', 39.863963, -75.004817), ('987115842', -22.741145, -43.524809), ('987184400', 40.75452, -73.99442
8), ('987231116', 51.609889, -0.220958), ('987578682', 45.344333, -75.663464), ('98762058', -6.648649, 108.531697), ('987931675', 3.058134, 101.636046
), ('987952836', 26.479096, -97.772538), ('98797026', 36.125742, -115.316038), ('987992689', 35.302686, -119.131522), ('988780549', 10.715263, 122.562
734), ('989010919', -24.174029, -65.296125), ('98919720', 39.637386, -77.710679), ('989365214', 35.922764, -83.191863), ('98948484', 37.288166, -121.7
8071), ('989695964', 43.921986, -79.455856), ('98993397', -36.727443, -73.110443), ('989935333', -31.359912, -64.33101), ('98994607', -22.91965, -43.2
39359), ('99001288', -23.448484, -46.557089), ('99006782', 29.618798, -95.408482), ('99025019', 55.810065, -4.335379), ('991080476', -22.888825, -43.3
01799), ('992093004', 44.328771, -94.481102), ('992158333', -23.758392, -53.308836), ('992756215', 36.242647, 140.291435), ('99281803', 13.863125, 100
.590266), ('99306982', -27.593717, -48.597556), ('993419628', -34.619987, -58.553191), ('99399628', -22.896852, -43.497166), ('994025173', 41.005795,
-75.181867), ('994104908', -27.578743, -48.609596), ('994115214', -34.870759, -56.02044), ('99415330', -3.075897, -59.961332), ('99421392', -12.076442
, -77.088962), ('994330746', -34.638346, -58.546498), ('994403964', 34.552007, 135.481768), ('994577408', 3.04754, 101.53223), ('99518517', 17.432231,
 102.753392), ('995309119', 53.505376, -1.363705), ('995827418', -30.039866, -51.220241), ('996083256', 18.430404, -69.999889), ('996146797', 40.85204
8, -81.790479), ('99636728', 28.595265, -81.28986), ('996382303', -34.093307, -59.02647), ('996396824', -34.613686, -58.54686), ('996703064', -8.10518
9, 111.876895), ('99670518', -6.709692, 108.556905), ('99671384', -6.87306, 107.55956), ('998179927', 35.673175, -88.855326), ('998290609', -34.61058,
 -58.447199), ('998597671', 41.918538, -87.866166), ('999543320', -34.512195, -58.515246), ('999583742', 62.028706, 129.728452)]
11405 rows are returned
```

b. Re-execute the SQL 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, it is as simple as it looks). Does the runtime scale linearly? (i.e., does it take 10X and 100X as much time?)

The screenshot of the query running 10 times is below:

C: > Users > rejalu1 > OneDrive - Henry Ford Health System > DSC450 > Assignments > FinalExamTakeHome > final2b.py > ...

```python
# Part 2b
import re
import sqlite3
import os
import time

os.chdir('C:/Users/rejalu1/OneDrive - Henry Ford Health System/DSC450/Assignments/FinalExamTakeHome')
# connect to the database file to open a connection
conn = sqlite3.connect('dsc450.db')                                # open the connection
cur = conn.cursor()                                                  # instantiate a cursor object
# construct a sql query and assign it to a string variable
sqlScript = """
SELECT Tweets.User_Id,
MIN(Geo.longitude),
MIN(latitude)
FROM Tweets, Geo
WHERE Tweets.GeoId = Geo.Id
GROUP BY Tweets.User_Id;
"""
startTime = time.time()
for i in range(10):
    cur.execute(sqlScript)

endTime = time.time()
print('The processing of the query 10 times takes %s seconds to run'%(endTime - startTime))
cur.close()
```

PROBLEMS 3 OUTPUT DEBUG CONSOLE TERMINAL JUPYTER Python

```
PS C:\Users\rejalu1\OneDrive - Henry Ford Health System\DSC450\Assignments\Week9> & C:/Users/rejalu1/.conda/envs/cmdpy37/python.exe "c:/Users/rejalu1/
OneDrive - Henry Ford Health System/DSC450/Assignments/FinalExamTakeHome/final2b.py"
The processing of the query 10 times takes 7.192025661468506 seconds to run
```

The screenshot of the query running 100 times:

C: > Users > rejalu1 > OneDrive - Henry Ford Health System > DSC450 > Assignments > FinalExamTakeHome > final2b.py > ...

```python
# longitude and latitude value for each user ID
# 10 times and 100 times
# and then measure the total run time
# Part 2b
import re
import sqlite3
import os
import time

os.chdir('C:/Users/rejalu1/OneDrive - Henry Ford Health System/DSC450/Assignments/FinalExamTakeHome')
# connect to the database file to open a connection
conn = sqlite3.connect('dsc450.db')                                # open the connection
cur = conn.cursor()                                                  # instantiate a cursor object
# construct a sql query and assign it to a string variable
sqlScript = """
SELECT Tweets.User_Id,
MIN(Geo.longitude),
MIN(latitude)
FROM Tweets, Geo
WHERE Tweets.GeoId = Geo.Id
GROUP BY Tweets.User_Id;
"""
startTime = time.time()
for i in range(100):
    cur.execute(sqlScript)

endTime = time.time()
print('The processing of the query 10 times takes %s seconds to run'%(endTime - startTime))
```

PROBLEMS 3 OUTPUT DEBUG CONSOLE TERMINAL JUPYTER Python

```
PS C:\Users\rejalu1\OneDrive - Henry Ford Health System\DSC450\Assignments\Week9> & C:/Users/rejalu1/.conda/envs/cmdpy37/python.exe "c:/Users/rejalu1/
OneDrive - Henry Ford Health System/DSC450/Assignments/FinalExamTakeHome/final2b.py"
The processing of the query 10 times takes 69.96626949310303 seconds to run
```

Compared to running it 10 times, which takes 0.1 mins, it takes 1.56 mins more to run it 100 times.

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

```python
# Author: Ronaldlee Ejalu
# Course DSC 540
# Final Exam
# Rewriting the equivalance of python logic finds the smallest
# longitude and latitude value for each user ID
# without using SQL
# Part 2c

import urllib.request
import json
import re
import sqlite3
import os
import csv
import time
import pandas as pd

os.chdir('C:/Users/rejalu1/OneDrive -
 Henry Ford Health System/DSC450/Assignments/FinalExamTakeHome')

fileName = 'C:/Users/rejalu1/OneDrive -
 Henry Ford Health System/DSC450/Assignments/FinalExamTakeHome/OneDayOfTweets.csv
'


def transformExtraneousValues(fileDictkey):
    """A function that takes a dictionary key and
    checks if the value is null, an empty string or []
    and it replaces it with None otherwise it assigns
    the actual value to a variable which is returned
    """

    valuestr = ''
    if fileDictkey =='null' or fileDictkey =='' or fileDictkey =='[]':
        valuestr = None
    else:
        valuestr = fileDictkey
    return valuestr


def extractLine():
    """Reading the file in chunks"""
    with open(fileName, 'rb') as f:
        for item in f:
            yield item
```

The screenshot of the code run time is below:

C: > Users > rejalu1 > OneDrive - Henry Ford Health System > DSC450 > Assignments > FinalExamTakeHome > final2c.py > ...

```python
# print(joinedDF.shape[0])
# perform multiple aggregations

# Group records by User_id, perform min on longitude and min on latitude
resAgg = joinedDF.groupby('User_Id', as_index=False).agg({'longitude':'min', 'latitude':min})
print('The top 20 rows are /n %s' %(resAgg.head(20)))
print('the number of records computed are %s' %(resAgg.shape[0]))
endTime = time.time()

print('The processing of the tweets data took %s seconds' %(endTime-startTime))
print('The number of operations per second is %s seconds' %(500000/(endTime-startTime)))
```

PROBLEMS 3 OUTPUT DEBUG CONSOLE TERMINAL JUPYTER

```
Processed 250000 tweets
The top 20 rows are /n      User_Id  longitude    latitude
0       3300  39.522038  -87.496588
1    2571851  38.627267  -77.365936
2    3026801  39.501035  -76.320989
3    3550481  -3.922880  119.789860
4    4507521  37.544506 -122.049999
5    5781452  -6.767315  108.519947
6    6380032  34.111404 -117.816942
7    6389212  46.273795 -119.361381
8    6769992  40.817063  -96.710171
9    7265462  34.653708 -118.210390
10   7519912  51.588476   -1.791912
11   7618792  35.638734  -80.472098
12   8203272 -34.618668  -58.670396
13   8492362  34.420459 -117.543042
14   9009042  24.728771   46.848448
15   9105582  -3.728181  -38.500570
16  10694342 -22.773269  -41.921204
17  11284242  -1.467635  -48.479942
18  11627352  34.971533  137.113988
19  11926282  25.877993  -80.329600
the number of records computed are 11844
The processing of the tweets data took 45.663657903671265 seconds
The number of operations per second is 10949.62652914849 seconds
```

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

```python
# Author: Ronaldlee Ejalu
# Course DSC 540
# Final Exam
# Rewriting the equivalance of python logic finds the smallest
# longitude and latitude value for each user ID
# without using SQL
# Part 2d

import os
import csv
import time
import pandas as pd

os.chdir('C:/Users/rejalu1/OneDrive -
 Henry Ford Health System/DSC450/Assignments/FinalExamTakeHome')

fileName = 'C:/Users/rejalu1/OneDrive -
 Henry Ford Health System/DSC450/Assignments/FinalExamTakeHome/OneDayOfTweets.csv
'


def transformExtraneousValues(fileDictkey):
    """A function that takes a dictionary key and
    checks if the value is null, an empty string or []
    and it replaces it with None otherwise it assigns
    the actual value to a variable which is returned
    """

    valuestr = ''
    if fileDictkey =='null' or fileDictkey =='' or fileDictkey =='[]':
        valuestr = None
    else:
        valuestr = fileDictkey
    return valuestr


def extractLine():
    """Reading the file in chunks"""
    with open(fileName, 'rb') as f:
        for item in f:
            yield item

chunkSize = 500000
generatedLines = extractLine()                              # invoke a helper ext
ractLine to read  the file in chunks
screenNameDict = {}
```

Executing the query in part 2c 10 times:

```python
# data columns description
# print(joinedDF.describe())
# print(joinedDF.shape[0])
# perform multiple aggregations
startTime = time.time()
# Group records by User_id, perform the minimum on longitude and minimum on latitude
for i in range(10):
    resAgg = joinedDF.groupby('User_Id', as_index=False).agg({'longitude':'min', 'latitude':min})
    # print('The top 20 rows are /n %s' %(resAgg.head(20)))
    # print('the number of records computed are %s' %(resAgg.shape[0]))
endTime = time.time()

print('The processing of the tweets data 10 times took %s seconds' %(endTime-startTime))
```

PROBLEMS 3 OUTPUT DEBUG CONSOLE TERMINAL JUPYTER Python

```
PS C:\Users\rejalu1\OneDrive - Henry Ford Health System\DSC450\Assignments\Week9> & C:/Users/rejalu1/.conda/envs/cmdpy37/python.exe "c:/Users/rejalu1/OneDrive - Henry Ford Health System/DSC450/Assignments/FinalExamTakeHome/final2d.py"
Processed 0 tweets
Processed 250000 tweets
The processing of the tweets data 10 times took 0.05804014205932617 seconds
```

Executing the query in part 2c 100 times:

```python
startTime = time.time()
# Group records by User_id, perform the minimum on longitude and minimum on latitude
for i in range(100):
    resAgg = joinedDF.groupby('User_Id', as_index=False).agg({'longitude':'min', 'latitude':min})
    # print('The top 20 rows are /n %s' %(resAgg.head(20)))
    # print('the number of records computed are %s' %(resAgg.shape[0]))
endTime = time.time()

print('The processing of the tweets data 10 times took %s seconds' %(endTime-startTime))
```

PROBLEMS 3 OUTPUT DEBUG CONSOLE TERMINAL JUPYTER Python

```
Processed 0 tweets
Processed 250000 tweets
The processing of the tweets data 10 times took 0.05804014205932617 seconds
PS C:\Users\rejalu1\OneDrive - Henry Ford Health System\DSC450\Assignments\Week9> & C:/Users/rejalu1/.conda/envs/cmdpy37/python.exe "c:/Users/rejalu1/OneDrive - Henry Ford Health System/DSC450/Assignments/FinalExamTakeHome/final2d.py"
Processed 0 tweets
Processed 250000 tweets
The processing of the tweets data 10 times took 0.6180455684661865 seconds
```

Based on the results of the time, running the query 10 times takes less time to execute when compared to running it 100 times.

e. Write the equivalent of the 2-a query in python by using regular expressions instead of json.loads(). Do not use json.loads() here. Note that you only need to find userid and geo location (if any) for each tweet, you don't need to parse the whole thing.

```python
# Author: Ronaldlee Ejalu
# Course DSC 540
# Final Exam
# Rewriting the equivalance of python logic finds the smallest
# longitude and latitude value for each user ID
# without using SQL
# Part 2e

import os
import csv
import time
import re

import pandas

os.chdir('C:/Users/rejalu1/OneDrive -
 Henry Ford Health System/DSC450/Assignments/FinalExamTakeHome')

fileName = 'C:/Users/rejalu1/OneDrive -
 Henry Ford Health System/DSC450/Assignments/FinalExamTakeHome/OneDayOfTweets.csv
'


def transformExtraneousValues(fileDictkey):
    """A function that takes a dictionary key and
    checks if the value is null, an empty string or []
    and it replaces it with None otherwise it assigns
    the actual value to a variable which is returned
    """

    valuestr = ''
    if fileDictkey =='null' or fileDictkey =='' or fileDictkey =='[]':
        valuestr = None
    else:
        valuestr = fileDictkey
    return valuestr


def extractLine():
    """Reading the file in chunks"""
    with open(fileName, 'rb') as f:
        for item in f:
            yield item

chunkSize = 500000
```

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

Processing the query in part 2-e 10 times took 32.98 seconds as shown below:

```
        )
# print(dataL)
# print(len(dataL))

# transform the list of tuples into a data frame.
df = pandas.DataFrame(dataL, columns=['User_Id', 'Type', 'longitude', 'latitude'])

startTime = time.time()
for i in range(10):
# Group records by User_id, perform min on longitude and min on latitude
    resAgg = df.groupby('User_Id', as_index=False).agg({'longitude':'min', 'latitude':min})


endTime = time.time()

print('The top 20 rows are /n %s' %(resAgg.head(20)))
print('the number of records computed are %s' %(resAgg.shape[0]))
print('The processing of the minimum longitude and latitude for each user took %s seconds' %(endTime-startTime))
```

PROBLEMS 6 OUTPUT DEBUG CONSOLE TERMINAL JUPYTER

```
14  1002643110   29.301451   -97.146381
15  1002700999   43.737907   -79.545189
16  1002748080   41.919662   -87.795174
17  1002823386   34.048536  -117.649656
18  1002866570    36.92059  -121.775173
19  1003300658   30.709846   -95.031001
the number of records computed are 14870
The processing of the minimum longitude and latitude for each user took 32.9830379486084 seconds
PS C:\Users\rejalu1\OneDrive - Henry Ford Health System\DSC450\Assignments\Week9>
```

Processing the query in part 2-e 100 times took 352.26 seconds as show below:

C: > Users > rejalu1 > OneDrive - Henry Ford Health System > DSC450 > Assignments > FinalExamTakeHome > final2e.py > ...

```python
endTime = time.time()

print('The top 20 rows are \n %s' %(resAgg.head(20)))
print('the number of records computed are %s' %(resAgg.shape[0]))
print('The processing of the minimum longitude and latitude for each user took %s seconds' %(endTime-startTime))
```

PROBLEMS 5 OUTPUT DEBUG CONSOLE TERMINAL JUPYTER Python

```
6   1001513000    -7.82601    110.39425
7   1001949656   18.479096   -69.903223
8   1002046280  -23.673729   -52.634936
9   1002108422    2.757217   101.689349
PS C:\Users\rejalu1\OneDrive - Henry Ford Health System\DSC450\Assignments\Week9> & C:/Users/rejalu1/.conda/envs/cmdpy37/python.exe "c:/Users/rejalu1/OneDrive - Henry Ford Health System/DSC450/Assignments/FinalExamTakeHome/final2e.py"
The top 20 rows are
        User_Id   longitude     latitude
0    100021456   54.631073    -5.841404
1   1000544017   34.281477  -118.454024
2   1000601762   32.291488   -83.878695
3   1000713967   33.142374  -117.086226
4   1000778952   32.820376   -96.973631
5   1001399526   21.333757  -158.062477
6   1001513000    -7.82601    110.39425
7   1001949656   18.479096   -69.903223
8   1002046280  -23.673729   -52.634936
9   1002108422    2.757217   101.689349
10  1002289843   51.639035    -1.313527
11   100233312    -6.23498    106.74735
12   100234910  -20.069479   -44.582519
13  1002421872   38.432701   -82.404734
14  1002643110   29.301451   -97.146381
15  1002700999   43.737907   -79.545189
16  1002748080   41.919662   -87.795174
17  1002823386   34.048536  -117.649656
18  1002866570    36.92059  -121.775173
19  1003300658   30.709846   -95.031001
the number of records computed are 14870
The processing of the minimum longitude and latitude for each user took 352.2630367279053 seconds
PS C:\Users\rejalu1\OneDrive - Henry Ford Health System\DSC450\Assignments\Week9>
```

Processing the query 100 times in 2e takes much time to process compared to when the query is processed 10 times.

Part 3

a. Using the database with 500,000 tweets, create a new table that corresponds to the join of all 3 tables in your database, <u>including records without a geo location</u>. This is the equivalent of a materialized view but since SQLite does not support MVs, we will use CREATE TABLE AS SELECT (instead of CREATE MATERIALIZED VIEW AS SELECT).

```
# Author: Ronaldlee Ejalu
# Course DSC 540
# Final Exam
# Using a database of 500,000 tweets
# create a new table that corresponds to the materialized view
# joining all the three tables.
# Part 3a
import sqlite3
import os
import time

os.chdir('C:/Users/rejalu1/OneDrive -
 Henry Ford Health System/DSC450/Assignments/FinalExamTakeHome')

# tweetdata = """http://dbgroup.cdm.depaul.edu/DSC450/OneDayOfTweets.txt"""
fileName = 'C:/Users/rejalu1/OneDrive -
 Henry Ford Health System/DSC450/Assignments/FinalExamTakeHome/OneDayOfTweets.csv
'

startTime = time.time()

conn = sqlite3.connect('dsc450.db')
                          # open the connection
cur = conn.cursor()
                          # instantiate a cursor object

sqlSciptView = """
CREATE TABLE TweetsMv AS
SELECT Tweets.CREATED_AT,
Tweets.ID,
Tweets.TEXT,
Tweets.SOURCE,
Tweets.IN_REPLY_TO_USER_ID,
Tweets.IN_REPLY_TO_SCREEN_NAME,
Tweets.IN_REPLY_TO_STATUS_ID,
Tweets.RETWEET_COUNT,
Tweets.CONTRIBUTORS,
Tweets.User_Id,
Tweets.GeoId,
UserTweets.name,
UserTweets.screen_name,
UserTweets.description,
UserTweets.friends_count,
Geo.Type,
Geo.longitude,
```

The screenshot of the first 10 records in the view:

C: > Users > rejalu1 > OneDrive - Henry Ford Health System > DSC450 > Assignments > FinalExamTakeHome > final3a.py > ...

```python


# Drop the tables if it exists
cur.execute('DROP TABLE IF EXISTS TweetsMv;')

# create the table
cur.execute(sqlSciptView)

mvRes = cur.execute('SELECT * FROM TweetsMv limit 10;').fetchall()
```

PROBLEMS 4 OUTPUT DEBUG CONSOLE TERMINAL JUPYTER Python

```
1/OneDrive - Henry Ford Health System/DSC450/Assignments/FinalExamTakeHome/final3a.py"
The first top ten records of the view are:
 [('Thu May 29 00:00:43 +0000 2014', '471803285746495489', 'There is no wealth but life. ~John Ruskin #wisdomink', '<a href="http://www.hootsuite.co
m" rel="nofollow">HootSuite</a>', None, None, None, 0, None, '213646047', None, 'Wisdom Ink', 'Wisdom_Ink', 'Wisdom Ink Online Magazine:  Expressing
 our #joy & #love via our articles. Celebrating diversity & the growth of #consciousness. #zen #meditation #spiritual', 4821, None, None, None), ('T
hu May 29 00:00:43 +0000 2014', '471803285738106880', 'Mucho la Plop esto, la Plop aquello, pero de los viernes es la fiesta con la gente más linda.
 \nEn las otras vienen directo de la frontera.', 'web', None, None, None, 0, None, '38950479', None, 'Yarer Sofier', 'firekites', 'Visite nuestro st
and en la planta cuarta. Gran liquidación en revólveres, cuchillos y todos los complementos de la mujer inquieta.', 99, None, None, None), ('Thu May
 29 00:00:43 +0000 2014', '471803285767462913', 'motive. When a political idea 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, 'Vera Luciani', 'rosaxonahag', 'Proud coffee advocate ; studi
es in  # all about-Devoted alcohol lover .', 261, None, None, None), ('Thu May 29 00:00:43 +0000 2014', '471803285750681600', '@im_2realbih bol!', '
<a href="http://twitter.com/download/android" rel="nofollow">Twitter for Android</a>', '290322075', 'im_2realbih', '471800733541486600', 0, None, '2
84813188', None, 'J e s s', '_SameOleSHIT', None, 683, None, None, None), ('Thu May 29 00:00:43 +0000 2014', '471803285759078401', 'A veces no enten
demos por que, 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>', None, None, None, 0, None, '146270795', None, 'rocio murillo lopez', 'gomitatica', None, 4, None, None,
None), ('Thu May 29 00:00:43 +0000 2014', '471803285742317568', 'jajajjaja hay no el vr todo por queres ver si se podia grabar lpm :#', 'web', None,
 None, None, 0, None, '1146905466', None, "Fuiste'peroperdiste.", 'NatyEmeri', 'Mi Idilo es aquel que años atras con solo una guitarra , consiguio s
u sueño en las escaleras del teatro avon.\r\nBELIEBER  ∞', 238, None, None, None), ('Thu May 29 00:00:43 +0000 2014', '471803285763280897', '@LariiN
andoFT9 kkkkkkkkkkkkkkk ai gente to aqui um bebê com uma recém-nascida e meu emocional?', 'web', '1854700621', 'LariiNandoFT9', '471802996926713860'
, 0, None, '1031294863', None, 'Mayara', 'foryoumartinez', None, 184, None, None, None), ('Thu May 29 00:00:43 +0000 2014', '471803285742288896', 'R
T @beyles66: cant wait for next year with @Harrison_BHS and @ThePurbalite', '<a href="http://twitter.com/download/android" rel="nofollow">Twitter fo
r Android</a>', None, None, None, 0, None, '329489005', None, 'Harrison_BHS', 'Harrison_BHS', 'English teacher and sponsor of the Purbalite and Lite
rary Guild. On a mission to stomp out bad music everywhere.', 105, None, None, None), ('Thu May 29 00:00:43 +0000 2014', '471803285738094592', '"@J_
Mf_Gandee26: This picture ◆◆◆◆◆◆◆◆◆◆◆◆◆◆ @PleasurePink_ @Damn_CeBadd http://t.co/mo59x1di"◆◆◆◆◆◆', '<a href="http://twitter.com/downlo
nofollow">Twitter for iPhone</a>', '1083804685', 'J_Mf_Gandee26', '471803096180752400', 0, None, '561545606', None, 'baddgirl◆', 'Damn_CeBadd', "#
TeamGet'EmBuck◆IG @_MixedShorty", 879, None, None, None), ('Thu May 29 00:00:43 +0000 2014', '471803285767462912', 'El sábado es la reu y no voy..
Segunda reunión que falto desde que entre a JBRO,mami forra.', '<a href="http://twitter.com/download/android" rel="nofollow">Twitter for Android</a>
', None, None, None, 0, None, '263210284', None, '† Stay and Believe †', 'yami_belieberJB', 'No siempre tenemos lo que queremos,aveces debemos confo
rmarnos con lo que podemos.', 320, None, None, None)]
The processing of the tweets data took 36.31770944595337 seconds
```

b. Export the contents of your table from 3-a into a new JSON file. You do not need to replicate the structure of the input and can come up with any reasonable keys for each field stored in JSON structure (e.g., you can have longitude as “longitude” key when the location is present). How does the file size compare to the original input file?

```
# Author: Ronaldlee Ejalu
# Course DSC 540
# Final Exam
# Export the contents of the Materialized view into a json structure.
# Part 3b
import json
import sqlite3
import os
import time

os.chdir('C:/Users/rejalu1/OneDrive -
 Henry Ford Health System/DSC450/Assignments/FinalExamTakeHome')

# tweetdata = """http://dbgroup.cdm.depaul.edu/DSC450/OneDayOfTweets.txt"""
fileName = 'C:/Users/rejalu1/OneDrive -
 Henry Ford Health System/DSC450/Assignments/FinalExamTakeHome/OneDayOfTweets.csv
'


startTime = time.time()

sqlScipt = """
SELECT * FROM TweetsMv;
"""
conn = sqlite3.connect('dsc450.db')
                          # open the connection
cur = conn.cursor()
                          # instantiate a cursor object

# create the table
cur.execute(sqlScipt)
rows = [rec for rec in cur]                                  # derive a list of ro
ws
rowL = []
tableColumns = [col[0] for col in cur.description]
    # derive a list of columns

for tablerow in rows:
    tDict = {}
    for tCol, tRow in zip(tableColumns, tablerow):  # using zip() which an iterat
or to conserve memory.
        tDict[tCol] = tRow
        rowL.append(tDict)
# print(rowL)
# jsonStr = json.dumps(rowL)      # derive a string representation of your table
schema as a json object
```

The json file appears much bigger when compared with the original csv file as shown below:

OneDayOfTweets.csv		8/18/2021 6:08 PM	Microsoft Excel Comma Separat...	1,541,349 KB
tableJson.json		8/20/2021 1:05 AM	JSON File	4,216,082 KB

c. Export the contents of your table from 3-a into a .csv file. How does the file size compare to the original input file and to the file in 3-b?

```
# Author: Ronaldlee Ejalu
# Course DSC 540
# Final Exam
# Export the contents of the Materialized view into a csv file.
# Part 3c
import json
import sqlite3
import os
import time
import csv

os.chdir('C:/Users/rejalu1/OneDrive -
 Henry Ford Health System/DSC450/Assignments/FinalExamTakeHome')

fileName = 'C:/Users/rejalu1/OneDrive -
 Henry Ford Health System/DSC450/Assignments/FinalExamTakeHome/OneDayOfTweets.csv
'


def writeTableRowsToFile(result, colNames):
  """Function that writes the list of tuples to a csv file"""

  # use newline='' to avoid redudant blank lines inn the file.
  with open('tweetsCsv.csv', "w", newline='', encoding= 'utf-8') as csvf:
    csvWriter = csv.writer(csvf)
    print(colNames)
    csvWriter.writerow([colNames])
    for item in result:                                # Loop through the items of
 the list
      csvWriter.writerow(item)                         # write the tuples of the l
ist to a csv file.
startTime = time.time()

sqlScipt = """
SELECT * FROM TweetsMv;
"""
conn = sqlite3.connect('dsc450.db')
                          # open the connection
cur = conn.cursor()
                          # instantiate a cursor object

# create the table
cur.execute(sqlScipt)
rows = [rec for rec in cur]                                  # derive a list of ro
ws
rowL = []
```

The screen shot of the file size is shown below:

tweetsCsv.csv		8/20/2021 1:46 AM	Microsoft Excel Comma Separat...	167,420 KB

The file size in 3c is smaller than both the original file and generated file in 3b.