Assignment 4- CSC455
Sarah Cummings

Part 1— OR function in python

```
import numpy as np

def ORFunction(matrix1, matrix2):
    if matrix1.shape == matrix2.shape:
        return np.logical_or(matrix1, matrix2)
    else:
        print('Error: Matrices aren't compatible!')
```

———————————————————————————————————————-

Testing in shell:

```
matrix1 = np.array([[True, False], [False, False]])
matrix2 = np.array([[False, True], [True, False]])

>>>>ORFunction(matrix1,matrix2)
array([[ True,  True],
       [ True, False]], dtype=bool
```

Part 2— A. Create a table for twitter data/ B. Populate with data from file
See attached python file, which is copied below:

```
import sqlite3
import json
```

```
#Connect to database
conn=sqlite3.connect('csc455.db')

#Request a cursor from the database
c=conn.cursor()

#Create the table
TwitterTable= '''CREATE TABLE Twitter
(
created_at VARCHAR(50),
id_str NUMBER(50),
text VARCHAR(160),
source VARCHAR(100),
in_reply_to_user_id VARCHAR(25),
in_reply_to_screen_name VARCHAR(25),
in_reply_to_status_id VARCHAR(25),
retweet_count NUMBER(5),
contributors  VARCHAR(25),
CONSTRAINT TwitterPK
        Primary Key(id_str)
);'''

#Drop tables if they exists
c.execute("DROP TABLE IF EXISTS Twitter")

#Create the tables
c.execute(TwitterTable)

#Open and read file
fd = open('/Users/sarahcummings/Documents/csc455/Assignment4.txt', 'r', encoding='utf8')

#split file on end of tweet deliminator and creates strings for each line
tweetList = fd.readline().split('EndOfTweet')
```

```
fd.close()

for tweet in tweetList:
    decoded_line = json.loads(tweet)
    insertvalues2 = (decoded_line.get(u'created_at'), decoded_line.get(u'id_str'), decoded_line.get(u'text'),
decoded_line.get(u'source'),  decoded_line.get(u'in_reply_to_user_id'), decoded_line.get(u'in_reply_to_screen_name'),
decoded_line.get(u'in_reply_to_status_id'), decoded_line.get(u'retweet_count'), decoded_line.get(u'contributors'))
    c.execute('INSERT INTO Twitter VALUES (?,?,?,?,?,?,?,?,?);', insertvalues2)

conn.commit()
conn.close()
```

Part 3. Write Queries for the following:

a. Count the number of iPhone users (based on “source” attribute)

```
SELECT Count(*) FROM Twitter
WHERE source LIKE ‘%iphone%’
```

b. Create a view that contains only tweets from users who are not replying ("in_reply_to_user_id" is NULL)

```
CREATE VIEW NotReply AS
SELECT * FROM Twitter
WHERE in_reply_to_user_id IS NULL;
```

c. Select tweets that have a "retweet_count" higher than the average "retweet_count" from the tweets in the view in part b

```
SELECT * FROM Twitter
WHERE retweet_count > (SELECT Avg(retweet_count) FROM NotReply);
```

d.Create a view that contains only id_str, text and source from each tweet that has a retweet_count of at least 5

CREATE VIEW FivePlusRTs AS
SELECT id_str, text, source FROM Twitter
WHERE “retweet_count” >= 5;

e.Use the view from part-d to find how many tweets have a retweet_count of at least 5

SELECT Count(*) FROM FivePlusRTs

f.Write python code to compute the answer from 3-e without using SQL, i.e., write code that is going to read data from the input file and answer the same question (find how many tweets have a “retweet_count” of at least 5).

```
fd = open('/Users/sarahcummings/Documents/csc455/Assignment4.txt', 'r', encoding='utf8')
count= 0
tweetList = fd.readline().split('EndOfTweet')
for tweet in tweetList:
        if '"retweet_count":0' in tweet:
            count= count
        if '"retweet_count":1' in tweet:
            count= count
        if '"retweet_count":2' in tweet:
            count= count
        if '"retweet_count":3' in tweet:
            count= count
        elif '"retweet_count":4' in tweet:
            count= count
        else:
           count= count+1
print(count)
```

running this code gives an answer of 175

Part 4. Write python to take table as param, and write corresponding insert statements for each row to a file.

```
import sqlite3
conn = sqlite3.connect('csc455_HW4.db')
c = conn.cursor()

f= open('inserts.txt','w')

def generateInsert(table):
    allRows = c.execute("SELECT * FROM %s;" % table).fetchall()
    for row in allRows:
        string= str("Insert INTO " + table + " VALUES" + str(row))
        f.write(string)
        f.close()

conn.commit()
conn.close()
```