```
#!/usr/bin/env python2
# -*- coding: utf-8 -*-
"""
Created on Tue Jul  3 11:55:48 2018

@author: intern
"""

from threading import Thread
from queue import Queue
import twitter
from datetime import datetime
import datetime as dt
import pandas as pd
import boto
from boto.s3.key import Key
from cStringIO import StringIO
import psycopg2
import mysql.connector
from datetime import date, timedelta
import os
import re

# Authentication
with open(os.environ['HOME']+'/env.txt') as f:
  env =dict([line.rstrip('\n').split('=') for line in f])
locals().update(env)

consumer_key =
[twitter_consumer_key_z,twitter_consumer_key_ebb,twitter_consumer_key_
tlu ]
consumer_secret =
[twitter_consumer_secret_z,twitter_consumer_secret_ebb,
twitter_consumer_secret_tlu]
access_token_key =
[twitter_access_token_key_z,twitter_access_token_key_ebb,
twitter_access_token_key_tlu]
access_token_secret =
[twitter_access_token_secret_z,twitter_access_token_secret_ebb,
twitter_access_token_secret_tlu]

api_0= twitter.Api(consumer_key[0], consumer_secret[0],
access_token_key[0],
access_token_secret[0],sleep_on_rate_limit=True,tweet_mode='extended')
api_1= twitter.Api(consumer_key[1], consumer_secret[1],
access_token_key[1],
access_token_secret[1],sleep_on_rate_limit=True,tweet_mode='extended')
api_2= twitter.Api(consumer_key[2], consumer_secret[2],
access_token_key[2],
access_token_secret[2],sleep_on_rate_limit=True,tweet_mode='extended')
```

```
API=[api_0, api_1, api_2]

def get_tweets(term, date, count, api):
    term = '"'+term+'"'
    since = date
    until = datetime.strptime(date, '%Y-%m-%d') + dt.timedelta(days=1)
    until = until.strftime('%Y-%m-%d')
    final_tweets_list=[]

    for loop in range(0, 100000):
        if loop == 0:
                max_id = None
        tweets = api.GetSearch(term = term.decode("utf-8"), count =
count, max_id=max_id,since = since, until = until, lang = 'en')
        if len(tweets) < 100:
            for tweet in tweets:
                final_tweets_list.append(tweet)
            break
        else:
            tweets_temp = []
            for tweet in tweets:
                tweets_temp.append(tweet)
            final_tweets_list = final_tweets_list + tweets_temp
            max_id = tweets_temp[99].id

    results = []
    for i in range(len(final_tweets_list)):
        dict = {}
        dict['id'] = final_tweets_list[i].id
        dict['favorite_count'] = final_tweets_list[i].favorite_count
        d = final_tweets_list[i].created_at.split()[1] +' '+
final_tweets_list[i].created_at.split()[2] + ' ' +
final_tweets_list[i].created_at.split()[3] + ' ' +
final_tweets_list[i].created_at.split()[5]
        dict['date']=datetime.strptime(d, '%b %d %H:%M:%S %Y')
        dict['retweet_count']=final_tweets_list[i].retweet_count
        dict['user_screen_name']=final_tweets_list[i].user.screen_name
        dict['user_name']=final_tweets_list[i].user.name

dict['user_followers_count']=final_tweets_list[i].user.followers_count
        if final_tweets_list[i].retweeted_status:
            dict['text']="RT"+"
"+final_tweets_list[i].retweeted_status.full_text
        else:
            dict['text']=final_tweets_list[i].full_text
        hashtags_list = []
        for hash in final_tweets_list[i].hashtags:
            tag= hash.text
            hashtags_list.append(tag)
        dict['tags']= ",".join(hashtags_list)
        dict['author']= term.replace('"','')
        results.append(dict)
    results_df = pd.DataFrame(results)
    return results_df
#test=get_tweets(term="Bill
Gates",date=yesterday,count=100,api=API[0])
# Get author list

#Connect Redshift to get the latest author data
dbname = "sailthrudata"
host = "sailthru-data.cmpnfzedptft.us-east-1.redshift.amazonaws.com"
conn_string = "dbname=%s port='5439' user=%s password=%s host=%s" %
(dbname, redshift_user, redshift_password, host)
print "Connecting to database\n->%s" % (conn_string)
conn = psycopg2.connect(conn_string)
cursor = conn.cursor()

sql = """
SELECT * FROM
        iobyte_amazon_catalog_feed
    WHERE
        import_detail_id IN (SELECT
                MAX(import_detail_id)
            FROM
                iobyte_amazon_catalog_feed);
"""
cursor.execute(sql)
iacf=cursor.fetchall()
iacf=pd.DataFrame(iacf)
conn.commit()

#Connect to MySQL to get the primary isbn
cnx = mysql.connector.connect(user=mysql_user,
password=mysql_password,
                              host='orim-internal-
db01.cqbltkaqn0z7.us-east-1.rds.amazonaws.com',
                              database='openroad_internal')
cursor = cnx.cursor()

query = """
    SELECT DISTINCT
        primary_isbn13
    FROM
        title_links_feed
    WHERE
        bisac_status IN ('active' , 'not yet published')
            AND partner_title = 'N'
            AND primary_isbn13 NOT IN (SELECT
                primary_isbn13
            FROM
                title_public_domain);
"""
cursor.execute(query)
isbn = cursor.fetchall()

#Author list by joining with reference_id(iacf) & isbn
iacf=pd.DataFrame(iacf)
isbn=pd.DataFrame(isbn)
#Rename two dataframe
isbn.rename(columns={0:'primary_isbn'}, inplace=True)
iacf.rename(columns={3:'reference_id',8:'author'}, inplace=True)
merged_author = iacf.merge(isbn, left_on='reference_id',
right_on='primary_isbn',how='right')

#Refine author name
author=merged_author['author']
a=list(set(author))
a=[aa for aa in a if str(aa) !='nan']
a=pd.DataFrame(a)
a = a[0].str.split(";").apply(pd.Series, 1).stack()
s=a.str.split(', ',expand=True)
s =  s.values.tolist()

authors = []
for ss in s:
    if ss[1] == None:
        author = ss[0]
    else:
        author = ss[1] + ' ' + ss[0]
    authors.append(author)
authors_new = []
for a in authors:
        a = re.sub(' +', ' ', a)
        authors_new.append(a)

authors_new = list(set(authors_new))
authors = authors_new
cursor.close()
cnx.close()

#Define the number of processors
n = 3
def chunkIt(seq, num):
    avg = len(seq) / float(num)
    out = []
    last = 0.0
    while last < len(seq):
        out.append(seq[int(last):int(last + avg)])
        last += avg
    return out
sep_author=chunkIt(range(len(authors)),n)

Author = []
for i in range(n): # create a list with nested lists
    Author.append([])
    for j in sep_author[i]:
        Author[i].append(authors[j]) # fills nested lists with data

for i in list(reversed(range(1,3))):
    yesterday = date.today() - timedelta(i)
    yesterday_file = yesterday.strftime('%Y%m%d')
    yesterday = yesterday.strftime('%Y-%m-%d')

    # Queue
    def run(idx, *args):
        global result
        if idx == 0:
            for a in Author[0]:
                p = get_tweets(term=a, date=yesterday, count=100,
api=API[0])
                print (idx,Author[0].index(a),a)
                result.append(p)
        elif idx==1:
            for a in Author[1]:
                p = get_tweets(term=a, date=yesterday, count=100,
api=API[1])
                print (idx,Author[1].index(a),a)
                result.append(p)
        else:
            for a in Author[2]:
                p = get_tweets(term=a, date=yesterday, count=100,
api=API[2])
                print (idx,Author[2].index(a),a)
                result.append(p)

    def run_jobs(jobs, workers=1):
        q = Queue()
        def worker(idx):
            while True:
                args = q.get()
                run(idx, *args)
                q.task_done()

        for job in jobs:
            q.put(job)

        for i in range(0, workers):
            t = Thread(target=worker, args=[i])
            t.daemon = True
            t.start()

        q.join()

    #Multiprocessing
    result=list()
    if __name__ == "__main__":
        run_jobs([('Author List :',i ) for i in Author], workers = 3)

    #Transform list into dataframe
    result=pd.concat(result)
    df = result
    df = df[['date', 'id', 'text', 'tags', 'retweet_count',
'favorite_count', 'user_screen_name', 'user_name',
'user_followers_count', 'author']]

    # Suppress scientific notation
    df.id = df.id.map(lambda x: '{:.0f}'.format(float(x)))
    df.to_csv('Results/twitter_%s.csv' % yesterday_file,
encoding='utf-8', index = False) #df.to_csv('Results/twitter_%s.csv' %
yesterday_file, encoding='utf-8', index = False)

    # Set up variables for S3 API
    bucket_name = "twitter-listening"
    folder = "mongodb"
    conn = boto.connect_s3(aws_access_key_id, aws_access_key)
    bucket = conn.get_bucket(bucket_name, validate = False)

    #Upload to S3
    with open('Results/twitter_%s.csv' % yesterday_file,'rb') as d:
        temp_file = StringIO(d.read())
        temp_file.seek(0)
        upload = Key(bucket)
        upload.key = 'twitter_%s.csv' % yesterday_file
        upload.set_contents_from_string(temp_file.getvalue())

    # Connect to RedShift
    dbname = "sailthrudata"
    host = "sailthru-data.cmpnfzedptft.us-
east-1.redshift.amazonaws.com"

    conn_string = "dbname=%s port='5439' user=%s password=%s host=%s"
%(dbname, redshift_user, redshift_password, host)
    print "Connecting to database\n        ->%s" % (conn_string)
    conn = psycopg2.connect(conn_string)
    cursor = conn.cursor()

    #Creating redshift table for the first time:
    #CREATE TABLE twitter_author (date timestamp, id char(150),text
varchar(max), tags varchar(max),retweet_count
varchar(225),favorite_count varchar(225), user_screen_name
varchar(255), user_name varchar(max), user_followers_count
varchar(225), author varchar(225))

    #Truncate the existing table before new ingestion
    #cursor.execute("TRUNCATE twitter_author")
    #conn.commit()

    #ingesting to redshift"
    sql = """
    COPY twitter_author
    FROM 's3://twitter-listening/twitter_%s.csv'
    CREDENTIALS 'aws_iam_role=arn:aws:iam::822605674378:role/
DataPipelineRole'
    IGNOREHEADER 1
    EMPTYASNULL
    QUOTE '"'
    CSV
    REGION 'us-east-1';
    """ % yesterday_file
    cursor.execute(sql)
    conn.commit()

    print 'Ingested Successfully to Redshift'
```