```
import tweepy
import csv
import datetime
import pandas as pd
#Import all libraries needed for analysis
#List of tweet Ids for tweet analysis
JenList = [1560704006892265472,1559274903798439936,1583115219035127811,1583867638773878791,1584350979159658497,1589789849384411137]
WalzList =[1584597890617724928 ,1585694736786096128,1589970730493808641,1590431641587286016]




#Provide API keys and validation tokens for Twitter API (Paid Only post 4/1/2023)
consumer_key ="*************************"
consumer_secret ="*************************"
access_token ="*************************"
access_secret ="*************************"

auth = tweepy.OAuthHandler(consumer_key, consumer_secret)
auth.set_access_token(access_token,access_secret)
api = tweepy.API(auth, wait_on_rate_limit=True)

#Validate login successful
if api.verify_credentials() == False:
    print("The user credentials are invalid.")
else:
    print("The user credentials are valid.")
```

```
The user credentials are valid.
```

```
import botometer
#import and log into botometer API for individual retweet analysis.
rapidapi_key = "*************************"
twitter_app_auth = {
    'consumer_key': "*************************",
    'consumer_secret': "*************************",
    'access_token': "*************************",
    'access_token_secret': "*************************",
  }
  #API call limit to 500 per day.
bom = botometer.Botometer(wait_on_ratelimit=True,
                          rapidapi_key=rapidapi_key,
                          **twitter_app_auth)


#Test-case validation to retreive data from Twitter API through tweepy.

#Parameters of tweet extraction.
user = "Tim_Walz"
start_date = pd.to_datetime("2022-10-27").date()
end_date= pd.to_datetime("2022-10-27").date()

tweets = api.user_timeline(screen_name = user, count = 200, include_rts=True, tweet_mode ='extended')

tweetid= api.get_status(1585617963876507650)


if api.verify_credentials() == False:
    print("The user credentials are invalid.")
else:
    print("The user credentials are valid.")
```

```
 The user credentials are valid.
```

```
#Test-case to extract liked tweets.
likeTweet=[]

for friend in api.friends(screen_name=user):
    likeTweet.append(friend)
print(likeTweet)
df = pd.DataFrame(likeTweet)
df.to_csv("likes1.csv")
```

```
#Prepair implementation of data extraction from API.
retweeters=[]
for tweet in tweets:
    if tweet.created_at.date() >= start_date and tweet.created_at.date() <= end_date:
       #target status id = 1585617963876507650
        print(tweet.full_text)
        print(tweet.full_text)
        retweets= api.retweets(tweet.id, count=100) #Twitter API limited to 100 calls. Not sorted.
        for retweet in retweets:
            retweeters.append(retweet.user.screen_name)
df = pd.DataFrame(retweeters)
if api.verify_credentials() == False:
    print("The user credentials are invalid.")
else:
    print("The user credentials are valid.")
```

```
Minnesota cannot be divided by dangerous, anti-democratic extremists.
Minnesota cannot be divided by dangerous, anti-democratic extremists.
The choice between your health and pocketbook should never be on the table. That's why I’ll always fight for Minnesotans to have access
The choice between your health and pocketbook should never be on the table. That's why I’ll always fight for Minnesotans to have access
I'm honored to be endorsed by more than 30 Minnesota mayors. Because coming together to hear about what's happening at the local level i
https://t.co/ggPINuyZx0
I'm honored to be endorsed by more than 30 Minnesota mayors. Because coming together to hear about what's happening at the local level i
https://t.co/ggPINuyZx0
Minnesota cannot become a test lab for extreme agendas – @PeggyFlanagan and I will not let that happen.

Help us fight back against extremism, uphold our Minnesota values, and move our state forward: https://t.co/54V9Zoh499
Minnesota cannot become a test lab for extreme agendas – @PeggyFlanagan and I will not let that happen.

Help us fight back against extremism, uphold our Minnesota values, and move our state forward: https://t.co/54V9Zoh499
What an honor! Thrilled to be endorsed by @GovJVentura.

I'm committed to being a governor for all Minnesotans, and I’ll work with anyone who’s willing to work with me to get things done.

Thank you, Jesse, for taking the unprecedented step to cast your independent vote for me! https://t.co/QABgPjFNX9
What an honor! Thrilled to be endorsed by @GovJVentura.

I'm committed to being a governor for all Minnesotans, and I’ll work with anyone who’s willing to work with me to get things done.

Thank you, Jesse, for taking the unprecedented step to cast your independent vote for me! https://t.co/QABgPjFNX9
The user credentials are valid.
```

```
#Test export to cvs file for next phase of analysis.

df.to_csv("Project1v2.csv") #Write to file successful
```

```
#Create list to store retweeters for Bot score submission.
RetweetList=[]
for i in WalzList: #List of tweet Ids for anaylsis.
    retweeter = api.retweets(i, count = 100)# API call to request retweets from tweet id list. Limit to 100 to avoid cool-down.
    RetweetList.append(i)
    for j in retweeter:
        RetweetList.append(j.user.screen_name)
    print(RetweetList) # confirm list is populated before transition to dataframe.

dfRetweet = pd.DataFrame(RetweetList)  #List placed into dataframe for cvs export.
dfRetweet.to_csv("Retweeters2.csv")
```

```
[1584597890617724928, 'bdunn300', 'samanth96173787', 'julie55443', 'GopherTheW', 'nmjohnson89', 'mnhumanrights', 'DrWalzMN', 'JodyAlforc
[1584597890617724928, 'bdunn300', 'samanth96173787', 'julie55443', 'GopherTheW', 'nmjohnson89', 'mnhumanrights', 'DrWalzMN', 'JodyAlforc
[1584597890617724928, 'bdunn300', 'samanth96173787', 'julie55443', 'GopherTheW', 'nmjohnson89', 'mnhumanrights', 'DrWalzMN', 'JodyAlforc
[1584597890617724928, 'bdunn300', 'samanth96173787', 'julie55443', 'GopherTheW', 'nmjohnson89', 'mnhumanrights', 'DrWalzMN', 'JodyAlforc
```

```
results = pd.DataFrame(columns = ["Name","Score"]) #create a name and bot score frame.
print(results)
```

```
Empty DataFrame
Columns: [Name, Score]
Index: []
```

```
#DO NOT RUN UNLESS SURE ABOUT 500 MAX LIMIT USAGE!
listName= []
listScore=[]
for screen_name, results in bom.check_accounts_in(RetweetList): #Run list of user names and check for bot score
    listName.append(screen_name) #append list to username
    listScore.append(results)#append list to bot score
    print(screen_name)#check user names and check for timeout.
```

```
1584597890617724928
bdunn300
samanth96173787
julie55443
GopherTheW
nmjohnson89
mnhumanrights
DrWalzMN
JodyAlford17
DianeRochMN
claybyington
magentanetzwerk
cj_devaan
wildmarshphoto
cbs144
pamgriffa
GSwensonSD
carsonca
OHarrysChar
Janeway12004
Skel531
ConradHarms
robgriddle
2672Local
dolly_horse
cdmorben
KarenJMerchant1
mrdunaganbailey
ConnieS53426503
EkbMary
Koogslaw
RiceCountyDFL
jenwo513
ouboomer61
AmyOnThePrairie
dandrenorton
mariedangelo22
BonneHargan
pat_thiel
Cindi1218
msbwul
IndivisibleMN58
SparkAlicia
Dirtface1
TShark4
frenfer123
LiLSilverJ
PHILCP69
BizonHornsUp
AnneLueben
purpleardvark
Chazraps
nsjww
MagentaMN
RTeachquijote
BillGoodermont
MorrisOldtom
jonimanderson
```

```
final = pd.DataFrame() #create final dataframe for export to csv.
#manipulate data to property formate data, list of list of lists.
final.insert(0,"Name","")
final.insert(1,"Score","")
final.insert(2,"Raw","")
dfls = pd.DataFrame(listScore)
#print(final)
index = 0
indexScore = 1
final["Score"] = dfls["cap"]
final["Name"] = listName
final = final.drop(labels=0, axis=0)
rawlist=[]
```

```
print(dfbot["display_scores"]["english"]['overall'])#check list formatting for export

final.to_csv("ScoresBotstestwalz2.csv") #Final Export.
```