#------ telegram bot and google trends param ------

key = os.environ['API\_KEY'] #This is telegrom bot api key

chat\_id = -776250187 # This is telegram chat room id.

search\_word = ['t-shirt','tshirt','t shirt','shirt','tee','merch','hoodie','sweatshirt','sleeve','apparel','pullover','costume','wearing','tee shirt','sweater','wear','hat','phone case','mug','poster','canvas','tumbler','mouse pad','t-shirt','tshirt','t shirt','shirt','tee','merch','hoodie','sweatshirt','sleeve','apparel','pullover','costume','wearing','tee shirt','sweater','wear','hat','phone case','mug','poster','canvas','tumbler','mouse pad']

cat=0

timeframe='today 1-m' #This is time for searching trends

geo='US' #This is geography for searching trends

gprop=''

sec\_ret = 600 #This is the time schedule bot send next trends in second

#------ google trends logic ------

pytrends = TrendReq(hl='en-US')

#get related queries

def rel\_queries(temp):

print(f'{str(datetime.datetime.now())}---{temp}')

strdt = ''

try:

pytrends.build\_payload(temp, cat, timeframe, geo, gprop)

except Exception as e:

print('=> BUILD FALSE!')

return f'Pytrend could not build {temp} due to: \n{e}'

for sw in temp:

data = pytrends.related\_queries()

data\_time = datetime.datetime.now()

data\_time = str(data\_time)

strdt += f'\n🛑Keyword: {sw} - {data\_time}\n\n'

try:

#print("---next---")

#print(data[sw]['rising'])

#print("-Num:")

#print(len(data[sw]['rising'].index))

head\_num = int(len(data[sw]['rising'].index))

count = 0

while count < head\_num:

strdt += '💹 '+str(data[sw]['rising']['value'][count])+' - '+str(data[sw]['rising']['query'][count])+'\n'

count+=1

except:

strdt +=f'Sorry, {sw} is not having any rising now!\n'

print('=> DONE!')

return strdt

#get find exact string from telegram callback

def find\_at(msg):

str\_output = ''

index = 0

for text in msg:

if '#' in text:

index = msg.index(text)

for x in range(index,len(msg)):

if '#' in msg[x]:

str\_output += msg[x][1:]

str\_output += " "

else:

str\_output += msg[x]

str\_output += " "

str\_output = str\_output.rstrip()

return str\_output

#------ schedule logic ------

def schedule\_checker():

while True:

schedule.run\_pending()

sleep(1)

def function\_to\_run():

sw\_num = len(search\_word)

if sw\_num >0:

count = 0

temp = []

for sw in search\_word:

if count == 5:

count = 0

try:

bot.send\_message(chat\_id, rel\_queries(temp))

except Exception as e:

bot.send\_message(chat\_id, f'Something when wrong while searching trending!\n{e}')

temp.clear()

temp.insert(0,sw)

count+=1

count = 0

print(f'last temp = {temp}')

if sw\_num%5 > 0:

try:

bot.send\_message(chat\_id, rel\_queries(temp))

except Exception as e:

bot.send\_message(chat\_id, f'Something when wrong while searching trending!\n{e}')

else:

bot.send\_message(chat\_id, "Please set keyword first!\nMore information please type /help")

#------- telegram bot logic ------

bot = telebot.TeleBot(key)

def bot\_run():

#set command start for bot

@bot.message\_handler(commands=['search'])

def alice\_answer(message):

sw\_num = len(search\_word)

if sw\_num >0:

count = 0

temp = []

for sw in search\_word:

if count == 5:

count = 0

try:

bot.send\_message(chat\_id, rel\_queries(temp))

except Exception as e:

bot.send\_message(chat\_id, f'Something when wrong while searching trending!\n{e}')

temp.clear()

temp.insert(0,sw)

count+=1

count = 0

print(f'last temp = {temp}')

if sw\_num%5 > 0:

try:

bot.send\_message(chat\_id, rel\_queries(temp))

except Exception as e:

bot.send\_message(chat\_id, f'Something when wrong while searching trending!\n{e}')

else:

bot.send\_message(chat\_id, "Please set keyword first!\nMore information please type /help")

#set command help for bot

@bot.message\_handler(commands=['help'])

def send\_help(message):

bot.reply\_to(message, f'All command:\n-/search\n-/remove\n-/current\n-/time\n\n Use #<<search keyword>> for begin searching!')

#set command stop for bot

@bot.message\_handler(commands=['remove'])

def remove\_search(message):

if len(search\_word) >0:

bot.reply\_to(message, f'Removed {len(search\_word)} keyword {search\_word}!')

search\_word.clear()

else:

bot.reply\_to(message, f'Nothing to remove here!\nPlease enter a @<<search keyword>> to start!')

#set command start for bot

@bot.message\_handler(commands=['current'])

def this\_key\_word(message):

if len(search\_word) >0:

bot.reply\_to(message, f'Current {len(search\_word)} keywords is {search\_word}!')

else:

bot.reply\_to(message, f'You did not set any keyword!')

#set command time for bot

@bot.message\_handler(commands=['time'])

def set\_time(message):

bot.reply\_to(message, f'Trending will update each {sec\_ret} sec!')

#get search key word for bot

@bot.message\_handler(func= lambda msg: msg.text is not None and '#' in msg.text)

def at\_answer(message):

texts = message.text.split()

at\_text = find\_at(texts)

bot.reply\_to(message, f'I ordered new trending \"{at\_text}"\" for you in \"{geo}\" with time \"{timeframe}\"!')

search\_word.insert(0,at\_text)

#alice\_answer(message)

bot.infinity\_polling(timeout=10, long\_polling\_timeout = 5)

#------ main logic ------

if \_\_name\_\_ == "\_\_main\_\_":

# Create the job in schedule.

schedule.every(sec\_ret).seconds.do(function\_to\_run)

# Spin up a thread to run the schedule check so it doesn't block your bot.

# This will take the function schedule\_checker which will check every second

# to see if the scheduled job needs to be ran.

Thread(target=schedule\_checker).start()

Thread(target=bot\_run).start()

# And then of course, start your server.