```
import tweepy #Library required for Twitter API
from tweepy.auth import OAuthHandler

import datetime, time
!pip install wget
import wget

import csv, re
import logging

import pandas as pd
import sqlite3

from pytz import timezone
```

```
Looking in indexes: https://pypi.org/simple, https://us-python.pkg.dev/colab-wheels/public/simple/
Collecting wget
  Downloading wget-3.2.zip (10 kB)
Building wheels for collected packages: wget
  Building wheel for wget (setup.py) ... done
  Created wheel for wget: filename=wget-3.2-py3-none-any.whl size=9674 sha256=8ea6625d0b31e88d36c942ed5e8541852273c4c73d5b406c04e9a
  Stored in directory: /root/.cache/pip/wheels/bd/a8/c3/3cf2c14a1837a4e04bd98631724e81f33f462d86a1d895fae0
Successfully built wget
Installing collected packages: wget
Successfully installed wget-3.2
```

```
!pip install pip

import pip
package ='tweepy' #Just replace the package name with any package to install it.
pip.main(['install',package])
```

```
Looking in indexes: https://pypi.org/simple, https://us-python.pkg.dev/colab-wheels/public/simple/
Requirement already satisfied: pip in /usr/local/lib/python3.7/dist-packages (21.1.3)
WARNING: pip is being invoked by an old script wrapper. This will fail in a future version of pip.
Please see https://github.com/pypa/pip/issues/5599 for advice on fixing the underlying issue.
To avoid this problem you can invoke Python with '-m pip' instead of running pip directly.
Looking in indexes: https://pypi.org/simple, https://us-python.pkg.dev/colab-wheels/public/simple/
Requirement already satisfied: tweepy in /usr/local/lib/python3.7/dist-packages (3.10.0)
Requirement already satisfied: six>=1.10.0 in /usr/local/lib/python3.7/dist-packages (from tweepy) (1.15.0)
Requirement already satisfied: requests[socks]>=2.11.1 in /usr/local/lib/python3.7/dist-packages (from tweepy) (2.23.0)
Requirement already satisfied: requests-oauthlib>=0.7.0 in /usr/local/lib/python3.7/dist-packages (from tweepy) (1.3.1)
Requirement already satisfied: oauthlib>=3.0.0 in /usr/local/lib/python3.7/dist-packages (from requests-oauthlib>=0.7.0->tweepy) (3
Requirement already satisfied: certifi>=2017.4.17 in /usr/local/lib/python3.7/dist-packages (from requests[socks]>=2.11.1->tweepy)
Requirement already satisfied: idna<3,>=2.5 in /usr/local/lib/python3.7/dist-packages (from requests[socks]>=2.11.1->tweepy) (2.10)
Requirement already satisfied: urllib3!=1.25.0,!=1.25.1,<1.26,>=1.21.1 in /usr/local/lib/python3.7/dist-packages (from requests[soc
Requirement already satisfied: chardet<4,>=3.0.2 in /usr/local/lib/python3.7/dist-packages (from requests[socks]>=2.11.1->tweepy) (
Requirement already satisfied: PySocks!=1.5.7,>=1.5.6 in /usr/local/lib/python3.7/dist-packages (from requests[socks]>=2.11.1->twee
0
```

Initializing credentials and Data Frames to hold the data.

```
consumer_key = "N9UT39YeOSUEk4DY0SCUhUste"
consumer_secret = "660kzn3kiviV5Q23TPm96F0Otth0yu2SHqSBTGrnvjuzttI97h"
access_key = "1590826381532798976-jqrLEziKmsFDF8sKZpriSGWDY04unh"
access_secret = "rOfmPQvqxkDDdOda0ivipkta2u6Z4fVphSJPUtyooAgvJ"


#Creating an empty dataframe to store the information
tweets =pd.DataFrame(columns=["id","created_at", "user", "text", "hashtags", "user_mentions", "media_url", "urls", "location", "retweetec
users = pd.DataFrame(columns=["id", "user_id", "created_at", "screen_name", "name", "url", "profile_image_url_https", "statuses_count"])

urls_df =pd.DataFrame(columns=["url", "tweet_ids"])
hashtags_df =pd.DataFrame(columns=["start_index","end_index","text", "tweet_ids"])
user_mentions_df =pd.DataFrame(columns=["id","screen_name"])
media_df =pd.DataFrame(columns=["id","url","type", "tweet_ids"])
jobs_df = pd.DataFrame(columns=["job_title", "description", "url", "poster", "posted_at", "tweet_ids"])
```

Fetching current datatime and date in the past with a difference of 14 days

```
eastern = timezone('US/Eastern')
lh = datetime.datetime.now()
last_hour = lh.astimezone(eastern) - datetime.timedelta(days=14)

#getting tweets since today
since_tweets = last_hour.strftime("%Y-%m-%d")
```

```
print(join_tweets)


auth = OAuthHandler(consumer_key, consumer_secret)
auth.set_access_token(access_key, access_secret)

api = tweepy.API(auth,wait_on_rate_limit=True)

try:
    api.verify_credentials()
    print("Authentication OK")
except:
    print("Error during authentication")
```

```
 Authentication OK
```

Downloading data from Twitter based on hashtag relevant to job recommendation.

```
api = tweepy.API(auth,wait_on_rate_limit=True)
tweets_data = [] #initialize master list to hold our ready tweets
user_data = [] #initialize master list to hold users

media_urls_data = []
media = []
urls_data = []
urls = []
hashtags_data = []
hashtags = []
user_mentions_data = []
user_mentions = []
#keywords=['#jobalert', '#JobSearch', "#job", "#hiring"]
#new_search = " OR ".join(keywords)# + "jobsearch -filter:retweets"
new_search="#jobalert"
num_tweets=100
cnt=0
job_positions = ["ENGINEER", "DEVELOPER", "OFFICER", "NURSE", "CLERK", "SUPERVISOR"]
job_data = []
temp=[]

tweet_set = tweepy.Cursor(api.search,q=new_search,count=100, #The q variable holds the hashtag
                          lang="en",
                          since=since_tweets).items()
try:
    for tweet in tweet_set:
        for position in job_positions:
            if(position in tweet.text):
                job=[]
                job.append(position)
                job.append(tweet.text.encode("utf-8"))
                cnt=cnt+1
                user=tweet.user
                user_data.append([user.id, user.created_at, user.screen_name, user.name, user.url, user.profile_image_url_https, user.sta
                if(tweet.entities.get('media',[])) :
                    for m in tweet.entities.get('media',[]):
                        media_urls_data.append([ m["id"], m["media_url"], m["url"], m["type"], tweet.id])
                        media.append(str(m["id"]))
                if(tweet.entities.get('urls',[])):
                    urlStr=[]
                    for u in tweet.entities.get('urls'):
                        urls_data.append([u["url"], tweet.id])
                        urls.append(u["url"])
                        urlStr.append(u["url"])
                    job.append("".join(urlStr))
                else:
                    job.append(None)
                if(tweet.entities.get('hashtags',[])) :
                    for ht in tweet.entities.get('hashtags'):
                        hashtags_data.append([ht["indices"][0], ht["indices"][1],ht["text"], tweet.id])
                        hashtags.append(ht["text"])
                if(tweet.entities.get('user_mentions',[])) :
                    for user in tweet.entities.get('user_mentions'):
                        user_data.append([user["id"], None,user["screen_name"], None, None, None, None])
                        user_mentions.append(str(user["id"]))
                tweets_data.append([tweet.id, tweet.created_at, tweet.user.id, tweet.text.encode("utf-8"),
                                    ", ".join(hashtags), ", ".join(user_mentions), ", ".join(media),
                                    ", ".join(urls), tweet.user.location, tweet.retweeted, tweet.retweet_count,
                                    tweet.favorite_count])
                job.append(tweet.user.name)
                job.append(tweet.created_at)
                job.append(tweet.id)
```

```
                job_data.append(job)
            continue
    except BaseException as e:
          print('BaseException,',str(e)) # print the error code obtained from twitter
          time.sleep(5)

    tweets_df = pd.DataFrame(tweets_data,
                            columns = ["tweet_id","created_at", "user_id", "text", "hashtags", "user_mentions", "media_url","url", "locatior
    tweets_df["tweet_id"]= pd.to_numeric(tweets_df["tweet_id"])
    tweets_df['created_at'] = pd.to_datetime(tweets_df['created_at'])
    tweets_df["retweeted_status"] = tweets_df['retweeted_status'].astype('bool')
    tweets_df["retweet_count"]= pd.to_numeric(tweets_df["retweet_count"])
    tweets_df["favorite_count"]= pd.to_numeric(tweets_df["favorite_count"])
    tweets_df
```

	tweet_id	created_at	user_id	text	ha
0	1591558499686707200	2022-11-12 22:28:10	1491776087684104193	b'RT @careersingov: Don\xe2\x80\x99t miss this...	
1	1591558499627958272	2022-11-12 22:28:10	1491776087684104193	b'RT @careersingov: .@SanBenitoCounty is #hiri...	
2	1591558499544092672	2022-11-12 22:28:10	1491776087684104193	b'RT @careersingov: Could this job be yours? @...	
3	1591552924198608897	2022-11-12 22:06:01	525005120	b'Could this job be yours? @SanBenitoCounty is...	
4	1591551183335952386	2022-11-12 21:59:06	40591485	b'RT @careersingov: .@SanBenitoCounty is #hiri...	
...	...	...	...	...	
91	1588587645449428997	2022-11-04 17:43:03	525005120	b'Great job! @YorkCountySCGov is #hiring a REG...	
92	1588572198071111680	2022-11-04 16:41:40	1491776087684104193	b'RT @careersingov: Don\xe2\x80\x99t miss this...	
93	1588571813826760706	2022-11-04 16:40:09	40591485	b'RT @careersingov: Don\xe2\x80\x99t miss this...	
94	1588571792028962817	2022-11-04 16:40:04	525005120	b'Don\xe2\x80\x99t miss this job opportunity! ...	
95	1588547130201505648	2022-11-04	525005120	b'.@CityofHesperia is #hiring a	

```
jobs_df = pd.DataFrame(job_data,
                        columns = ["job_title", "description", "url", "poster", "posted_at","tweet_ids"] )

jobs_df['posted_at'] = pd.to_datetime(jobs_df['posted_at'])
jobs_df
```

	job_title	description	url	poster	pos
0	NURSE	b'RT @careersingov: Don\xe2\x80\x99t miss this...	None	Jobs via Tweet	2
1	SUPERVISOR	b'RT @careersingov: .@SanBenitoCounty is #hiri...	None	Jobs via Tweet	2
2	SUPERVISOR	b'RT @careersingov: Could this job be yours? @...	None	Jobs via Tweet	2
3	SUPERVISOR	b'Could this job be yours?	[illegible]	CareersInGovernment	2

```
user_df = pd.DataFrame(user_data,
                       columns=["user_id", "created_at", "screen_name", "name", "url", "profile_image_url_https", "statuses_count"])

user_df["user_id"]= pd.to_numeric(user_df["user_id"])
user_df['created_at'] = pd.to_datetime(user_df['created_at'])
user_df["statuses_count"] = pd.to_numeric(user_df["statuses_count"])
user_df
```

	user_id	created_at	screen_name	name	
0	1491776087684104193	2022-02-10 14:08:51	jobsviatweet	Jobs via Tweet	https://t
1	525005120	NaT	careersingov	None	
2	1171080641133236225	NaT	SanBenitoCounty	None	
3	1491776087684104193	2022-02-10 14:08:51	jobsviatweet	Jobs via Tweet	https://t
4	525005120	NaT	careersingov	None	
...	...	...	...	...	
207	41657673	NaT	KansasCity	None	
208	525005120	2012-03-15 03:52:59	careersingov	CareersInGovernment	https://t.c
209	41657673	NaT	KansasCity	None	
210	525005120	2012-03-15 03:52:59	careersingov	CareersInGovernment	https://t.c

```
hashtags_df = pd.DataFrame(hashtags_data,
                           columns = ["start_index", "end_index", "text", "tweet_ids"])

hashtags_df["start_index"]= pd.to_numeric(hashtags_df["start_index"])
hashtags_df["end_index"]= pd.to_numeric(hashtags_df["end_index"])
hashtags_df
```

	start_index	end_index	text	tweet_ids
0	59	66	hiring	1591558499686707200
1	39	46	hiring	1591558499627958272
2	63	70	hiring	1591558499544092672
3	45	52	hiring	1591552924198608897
4	39	46	hiring	1591551183335952386
...	...	...	...	...
90	31	38	hiring	1588587645449428997
91	66	73	hiring	1588572198071111680
92	66	73	hiring	1588571813826760706
93	48	55	hiring	1588571792028962817
94	20	27	hiring	1588547130301595648

95 rows × 4 columns

```
media_df = pd.DataFrame(media_urls_data,
                           columns = ["id", "media_url", "url", "type", "tweet_ids"])
```

```
media_df["id"]= pd.to_numeric(media_df["id"])
media_df
```

	id	media_url	url	type	tweet_ids

```
urls_df = pd.DataFrame(urls_data,
                       columns = ["url", "tweet_ids"])

urls_df
```

	url	tweet_ids
0	https://t.co/ugdjFewaYD	1591552924198608897
1	https://t.co/hr7oUFU3H9	1591551161554837504
2	https://t.co/qSjYNC1DdD	1591549399402008579
3	https://t.co/FDOZBGpxkF	1591461323451535360
4	https://t.co/NxEwkaqhyR	1591404658106552321
...	...	...
57	https://t.co/vQN2t6QMdE	1588628155912404992
58	https://t.co/Jp8N7ZNvdI	1588605260527984641
59	https://t.co/8PmyUsGBzZ	1588587645449428997
60	https://t.co/PoI0dJybL0	1588571792028962817
61	https://t.co/l72eOtK7dj	1588547130301595648

62 rows × 2 columns

Save Data Frames to SQL database in different tables.

```
conn = sqlite3.connect('MakeMyCareer.db')
cur = conn.cursor()


create_tweets_query="CREATE TABLE tweets(tweet_id int PRIMARY KEY NOT NULL UNIQUE, \
                          created_at DATE NOT NULL, user_id int NOT NULL,text VARCHAR(255) NOT NULL,hashtags VARCHAR,user_mentions VARCHA
                          location VARCHAR(255),retweeted_status BOOL NOT NULL,retweet_count int NOT NULL,favorite_count int,\
                          FOREIGN KEY (user_mentions) REFERENCES user_mentions (VARCHAR) );"

cur.execute("DROP TABLE IF EXISTS tweets;")
cur.execute(create_tweets_query)

tweets_df.to_sql('tweets',con=conn,index=False, if_exists='replace')


create_users_query="CREATE TABLE users(user_id int PRIMARY KEY NOT NULL UNIQUE,created_at DATE NOT NULL,screen_name VARCHAR(255) NOT NULL
                          name VARCHAR(255) NOT NULL,url VARCHAR ,profile_image_url_https VARCHAR, statuses_count);"

cur.execute("DROP TABLE IF EXISTS users;")
cur.execute(create_users_query)

#user_df.drop(columns = user_df.columns[0], axis = 1, inplace= True)
user_df.to_sql('users',con=conn,index=False, if_exists='replace')


create_hashtags_query="CREATE TABLE hashtags(hashtag_id int PRIMARY KEY NOT NULL UNIQUE,text VARCHAR(255),tweet_ids VARCHAR,\
                            FOREIGN KEY (tweet_ids) REFERENCES tweets (id) );"

cur.execute("DROP TABLE IF EXISTS hashtags;")
cur.execute(create_hashtags_query)

hashtags_df.to_sql('hashtags',con=conn,index=False, if_exists='replace')


create_media_query="CREATE TABLE media_urls(media_id int PRIMARY KEY NOT NULL UNIQUE,media_url VARCHAR,url VARCHAR,\
                            type VARCHAR(255), tweet_ids VARCHAR, FOREIGN KEY (tweet_ids) REFERENCES tweets (tweet_id) );"

cur.execute("DROP TABLE IF EXISTS media_urls;")
cur.execute(create_media_query)

media_df.to_sql('media_urls',con=conn,index=False, if_exists='replace')


create_urls_query="CREATE TABLE urls(url VARCHAR PRIMARY KEY NOT NULL UNIQUE,\
                            tweet_ids VARCHAR, FOREIGN KEY (tweet_ids) REFERENCES tweets (tweet_id) );"

cur.execute("DROP TABLE IF EXISTS urls;")
```

```
cur.execute(create_urls_query)

urls_df.to_sql('urls',con=conn,index=False, if_exists='replace')

create_jobs_query="CREATE TABLE jobs(job_title VARCHAR(255), description VARCHAR,\
                           url VARCHAR PRIMARY KEY NOT NULL UNIQUE,poster VARCHAR ,posted_at VARCHAR, \
                           tweet_ids VARCHAR, FOREIGN KEY (tweet_ids) REFERENCES tweets (tweet_id) );"

cur.execute("DROP TABLE IF EXISTS jobs;")
cur.execute(create_jobs_query)

jobs_df.to_sql('jobs',con=conn,index=False, if_exists='replace')


def run_query(query):
    return pd.read_sql(query,conn)
```

SQL queries to express the below questions:

- What user posted this tweet?
- When did the user post this tweet?
- What tweets have this user posted in the past 24 hours?
- How many tweets have this user posted in the past 24 hours?
- When did this user join Twitter?
- What keywords/ hashtags are popular?
- What tweets are popular?

```
list_of_tables = ["tweets","users","hashtags","media_urls","urls","jobs"]
table=list_of_tables[0]
select_query="SELECT * FROM "+table+" LIMIT 5"


query_q1 = "SELECT DISTINCT users.user_id, users.screen_name, users.name \
               FROM users INNER JOIN tweets ON users.user_id=tweets.user_id \
              where tweets.tweet_id = '1591558499627958272' "
query_q2 = "SELECT DISTINCT tweets.created_at \
              FROM users INNER JOIN tweets ON users.user_id=tweets.user_id \
              where tweets.tweet_id = '1591558499627958272' "
query_q3 = "SELECT DISTINCT tweets.tweet_id,tweets.text,users.user_id \
              FROM tweets INNER JOIN users ON users.user_id=tweets.user_id \
              WHERE tweets.user_id=1491776087684104193 AND \
              tweets.created_at>'2022-11-11 22:06:01'"
query_q4 = "SELECT users.user_id,COUNT(tweets.tweet_id) as number_of_tweets \
              FROM tweets INNER JOIN users ON users.user_id=tweets.user_id \
              WHERE tweets.user_id=1491776087684104193 AND \
              tweets.created_at>'2022-11-11 22:06:01'"
query_q5 = "SELECT DISTINCT users.user_id,users.created_at FROM users WHERE users.user_id=1491776087684104193"
query_q6 = "SELECT DISTINCT hashtags.text, tweets.retweet_count FROM hashtags \
              INNER JOIN tweets on tweets.tweet_id=hashtags.tweet_ids \
              WHERE hashtags.retweet_count > 3"
query_q7 = "SELECT tweet_id, text, retweet_count FROM tweets ORDER BY retweet_count DESC"

run_query(query_q6)
```

	text	retweet_count
0	jobopening	4
1	hiring	4

Career Recommendation System : Use Cases

1. Use Case: User can look for opening for their target job position Description: User can look for opening for a position named "Engineer"

   Actor: User

   Precondition: User should have a valid target position name

   Steps:

   Actor action: User request for list of job openings for his target position.

   System Responses: If the positon exists, the system will return a list of job openings posted.

   Post Condition: List of job openings suggested

   Alternate Path: The user request is not correct and system throws an error

   Error: User information is incorrect

2. Use Case: User can look for openings posted by their dream company handle Description: Search for job posts posted by a particular user

   Actor: User

   Precondition: User should have a company name user is target

   Steps:

   Actor action: User request for list of job openings for his target position.

   System Responses: If the company has posted job openings, the system will return the list.

   Post Condition: List of job openings suggested

   Alternate Path: The user request is not correct and system throws an error

   Error: User information is incorrect

3. Use Case: User can look for openings posted within last 5 days and for a particular position Description: Search for job posts posted within last 5 days

   Actor: User

   Precondition: User should have a valid target position name

   Steps:

   Actor action: User request for list of job openings for his target position.

   System Responses: The system will return a list of job posts.

   Post Condition: List of job openings suggested

   Alternate Path: The user request is not correct and system throws an error

   Error: User information is incorrect

4. Use Case: User can assess which job positions are more in demand Description: Search for job posts for different job positions

   Actor: User

   Precondition: User should have a valid target position name

   Steps:

   Actor action: User request for list of job openings for his target position.

   System Responses: The system will return a count of job posts for a position.

   Post Condition: List of job openings suggested

   Alternate Path: The user request is not correct and system throws an error

   Error: User information is incorrect

5. Use Case: User can assess which companies are posting more jobs

   Description: Search for job posts for job positions by different companies

   Actor: User

   Precondition: User should have a valid target position name and target company

   Steps:

   Actor action: User request for list of job openings for his target position posted by copmany.

   System Responses: The system will return a count of job posts posted by company handle.

   Post Condition: Count of job openings suggested

   Alternate Path: The user request is not correct and system throws an error

   Error: User information is incorrect

```
#Use Case: User can look for opening for their target job position
use_case_1 = "SELECT jobs.job_title,jobs.description, jobs.poster, jobs.posted_at FROM jobs INNER JOIN  tweets ON tweets.tweet_id=jobs.tw

#Use Case: User can look for openings posted by their dream company
use_case_2 = "SELECT jobs.job_title,jobs.description, jobs.poster, jobs.posted_at \
                FROM jobs WHERE poster='CareersInGovernment'"

#Use Case: User can look for openings posted within last 5 days and for a particular position
use_case_3 = "SELECT jobs.job_title,jobs.description, jobs.poster, jobs.posted_at \
                FROM jobs WHERE job_title='NURSE' and posted_at>'2022-11-11 22:28:10'"

#Use Case: User can assess which job positions are more in demand
```

```
use_case_4 = "SELECT jobs.job_title,COUNT(jobs.job_title) as number_of_postings,jobs.description, jobs.poster, jobs.posted_at \
              FROM jobs GROUP BY job_title ORDER BY number_of_postings DESC"

#Use Case: User can assess which companies are posting more jobs
use_case_5 = "SELECT jobs.job_title,COUNT(jobs.job_title) as number_of_postings,jobs.description, jobs.poster, jobs.posted_at \
              FROM jobs GROUP BY poster"

run_query(use_case_4)
```

	job_title	number_of_postings	description	poster	posted
0	SUPERVISOR	36	b'.@CityofHesperia is #hiring a MAINTENANCE CR...	CareersInGovernment	2022 15:0
1	CLERK	20	b'ADMIN CLERK \n\nOrganization: Department of ...	Jobshaven	2022 04:3
2	OFFICER	15	b'Don\xe2\x80\x99t miss this job	CareersInGovernment	2022