```
!pip install selenium tweepy linkedin-api facebook-scraper
!apt-get update
!apt install -yq chromium-chromedriver
```

```
Unpacking snapd (2.58+22.04.1) ...
Setting up apparmor (3.0.4-2ubuntu2.2) ...
Created symlink /etc/systemd/system/sysinit.target.wants/apparmor.service → /lib/systemd/system/apparmor.service.
Setting up liblzo2-2:amd64 (2.10-2build3) ...
Setting up squashfs-tools (1:4.5-3build1) ...
Setting up udev (249.11-0ubuntu3.9) ...
invoke-rc.d: could not determine current runlevel
invoke-rc.d: policy-rc.d denied execution of start.
Setting up libfuse3-3:amd64 (3.10.5-1build1) ...
Setting up snapd (2.58+22.04.1) ...
Created symlink /etc/systemd/system/multi-user.target.wants/snapd.aa-prompt-listener.service → /lib/systemd/system/snapd.aa-prompt-li
Created symlink /etc/systemd/system/multi-user.target.wants/snapd.apparmor.service → /lib/systemd/system/snapd.apparmor.service.
Created symlink /etc/systemd/system/multi-user.target.wants/snapd.autoimport.service → /lib/systemd/system/snapd.autoimport.service.
Created symlink /etc/systemd/system/multi-user.target.wants/snapd.core-fixup.service → /lib/systemd/system/snapd.core-fixup.service.
Created symlink /etc/systemd/system/multi-user.target.wants/snapd.recovery-chooser-trigger.service → /lib/systemd/system/snapd.recove
Created symlink /etc/systemd/system/multi-user.target.wants/snapd.seeded.service → /lib/systemd/system/snapd.seeded.service.
Created symlink /etc/systemd/system/cloud-final.service.wants/snapd.seeded.service → /lib/systemd/system/snapd.seeded.service.
Unit /lib/systemd/system/snapd.seeded.service is added as a dependency to a non-existent unit cloud-final.service.
Created symlink /etc/systemd/system/multi-user.target.wants/snapd.service → /lib/systemd/system/snapd.service.
Created symlink /etc/systemd/system/timers.target.wants/snapd.snap-repair.timer → /lib/systemd/system/snapd.snap-repair.timer.
Created symlink /etc/systemd/system/sockets.target.wants/snapd.socket → /lib/systemd/system/snapd.socket.
Created symlink /etc/systemd/system/final.target.wants/snapd.system-shutdown.service → /lib/systemd/system/snapd.system-shutdown.serv
Selecting previously unselected package chromium-browser.
(Reading database ... 121268 files and directories currently installed.)
Preparing to unpack .../chromium-browser_1%3a85.0.4183.83-0ubuntu2.22.04.1_amd64.deb ...
=> Installing the chromium snap
==> Checking connectivity with the snap store
===> System doesn't have a working snapd, skipping
Unpacking chromium-browser (1:85.0.4183.83-0ubuntu2.22.04.1) ...
Selecting previously unselected package chromium-chromedriver.
Preparing to unpack .../chromium-chromedriver_1%3a85.0.4183.83-0ubuntu2.22.04.1_amd64.deb ...
Unpacking chromium-chromedriver (1:85.0.4183.83-0ubuntu2.22.04.1) ...
Selecting previously unselected package systemd-hwe-hwdb.
Preparing to unpack .../systemd-hwe-hwdb_249.11.3_all.deb ...
Unpacking systemd-hwe-hwdb (249.11.3) ...
Setting up systemd-hwe-hwdb (249.11.3) ...
Setting up chromium-browser (1:85.0.4183.83-0ubuntu2.22.04.1) ...
update-alternatives: using /usr/bin/chromium-browser to provide /usr/bin/x-www-browser (x-www-browser) in auto mode
update-alternatives: using /usr/bin/chromium-browser to provide /usr/bin/gnome-www-browser (gnome-www-browser) in auto mode
Setting up chromium-chromedriver (1:85.0.4183.83-0ubuntu2.22.04.1) ...
Processing triggers for udev (249.11-0ubuntu3.9) ...
Processing triggers for hicolor-icon-theme (0.17-2) ...
Processing triggers for libc-bin (2.35-0ubuntu3.1) ...
/sbin/ldconfig.real: /usr/local/lib/libtbbbind_2_0.so.3 is not a symbolic link

/sbin/ldconfig.real: /usr/local/lib/libtbbmalloc.so.2 is not a symbolic link

/sbin/ldconfig.real: /usr/local/lib/libtbbmalloc_proxy.so.2 is not a symbolic link

/sbin/ldconfig.real: /usr/local/lib/libtbbbind.so.3 is not a symbolic link

/sbin/ldconfig.real: /usr/local/lib/libtbbbind_2_5.so.3 is not a symbolic link

/sbin/ldconfig.real: /usr/local/lib/libtbb.so.12 is not a symbolic link

Processing triggers for man-db (2.10.2-1) ...
Processing triggers for dbus (1.12.20-2ubuntu4.1) ...
```

```
import csv
import time
import tweepy
from selenium import webdriver
from selenium.webdriver.common.keys import Keys
from selenium.common.exceptions import NoSuchElementException
from linkedin_api import Linkedin
from facebook_scraper import get_posts
import warnings
warnings.filterwarnings("ignore")


# Function to get user input for websites
def get_user_input_websites():
    websites = []
    for i in range(3):
        website = input(f"Enter website {i+1} (URL or name): ")
        websites.append(website)
```

```
    return websites


# Function to get user input for keywords
def get_user_input_keywords():
    keywords = input("Enter keywords (comma-separated): ")
    return [keyword.strip() for keyword in keywords.split(',')]


# Function to scrape LinkedIn
def scrape_linkedin(keywords):
    scraped_data = []

    # Replace with your actual LinkedIn credentials
    username = "your_username"
    password = "your_password"

    api = Linkedin(username, password)

    for keyword in keywords:
        search_results = api.search_v2(keyword)
        for result in search_results:
            post_url = f"https://www.linkedin.com/feed/update/{result['id']}/"
            user_name = result['author']['name']
            scraped_data.append((post_url, user_name))

    return scraped_data


# Function to scrape Facebook
def scrape_facebook(keywords):
    scraped_data = []

    for keyword in keywords:
        for post in get_posts(keyword, pages=5):
            post_url = f"https://www.facebook.com/{post['post_id']}/"
            user_name = post['username']
            scraped_data.append((post_url, user_name))

    return scraped_data


# Function to authenticate Twitter
def authenticate_twitter():
    # Replace with your Twitter API credentials
    consumer_key = "your_consumer_key"
    consumer_secret = "your_consumer_secret"
    access_token = "your_access_token"
    access_token_secret = "your_access_token_secret"

    auth = tweepy.OAuthHandler(consumer_key, consumer_secret)
    auth.set_access_token(access_token, access_token_secret)
    api = tweepy.API(auth)

    return api


# Function to scrape Twitter
def scrape_twitter(keywords, max_tweets=5):
    api = authenticate_twitter()
    scraped_data = []

    for keyword in keywords:
        tweets = tweepy.Cursor(api.search, q=keyword, tweet_mode="extended").items(max_tweets)
        for tweet in tweets:
            post_url = f"https://twitter.com/{tweet.user.screen_name}/status/{tweet.id}/"
            user_name = tweet.user.screen_name
            scraped_data.append((post_url, user_name))

    return scraped_data


# Function to save data to CSV
def save_to_csv(data, csv_filename):
    with open(csv_filename, 'w', newline='', encoding='utf-8') as csvfile:
        csv_writer = csv.writer(csvfile)
        csv_writer.writerow(["Post URL", "User Name"])
        csv_writer.writerows(data)
```

```
def main():
    try:
        websites = get_user_input_websites()
        keywords = get_user_input_keywords()

        scraped_data = []

        for website in websites:
            if "linkedin" in website.lower():
                scraped_data.extend(scrape_linkedin(keywords))
            elif "facebook" in website.lower():
                scraped_data.extend(scrape_facebook(keywords))
            elif "twitter" in website.lower():
                scraped_data.extend(scrape_twitter(keywords))
            else:
                print(f"Unsupported website: {website}")

        if scraped_data:
            csv_filename = "scraped_data.csv"
            save_to_csv(scraped_data, csv_filename)
            print(f"Data saved to {csv_filename}")
        else:
            print("No data to save.")

    except Exception as e:
        print(f"An error occurred: {e}")

if __name__ == "__main__":
    main()
```

```
 Enter website 1 (URL or name):
```

Colab paid products - Cancel contracts here

Executing (48s) <cell line: 28> › main() › get_user_input_websites() › raw_input() › _input_request() › select()