Scraping Daily Mail Online News Articles and their Comments

```
!pip install selenium
```

```
Collecting selenium
  Downloading selenium-4.29.0-py3-none-any.whl.metadata (7.1 kB)
Requirement already satisfied: urllib3<3,>=1.26 in /usr/local/lib/python3.11/dist-packages (from urllib3[socks]<3,>=1.26->selenium) (2.3
Collecting trio~=0.17 (from selenium)
  Downloading trio-0.29.0-py3-none-any.whl.metadata (8.5 kB)
Collecting trio-websocket~=0.9 (from selenium)
  Downloading trio_websocket-0.12.2-py3-none-any.whl.metadata (5.1 kB)
Requirement already satisfied: certifi>=2021.10.8 in /usr/local/lib/python3.11/dist-packages (from selenium) (2025.1.31)
Requirement already satisfied: typing_extensions~=4.9 in /usr/local/lib/python3.11/dist-packages (from selenium) (4.12.2)
Requirement already satisfied: websocket-client~=1.8 in /usr/local/lib/python3.11/dist-packages (from selenium) (1.8.0)
Requirement already satisfied: attrs>=23.2.0 in /usr/local/lib/python3.11/dist-packages (from trio~=0.17->selenium) (25.2.0)
Requirement already satisfied: sortedcontainers in /usr/local/lib/python3.11/dist-packages (from trio~=0.17->selenium) (2.4.0)
Requirement already satisfied: idna in /usr/local/lib/python3.11/dist-packages (from trio~=0.17->selenium) (2.10)
Collecting outcome (from trio~=0.17->selenium)
  Downloading outcome-1.3.0.post0-py2.py3-none-any.whl.metadata (2.6 kB)
Requirement already satisfied: sniffio>=1.3.0 in /usr/local/lib/python3.11/dist-packages (from trio~=0.17->selenium) (1.3.1)
Collecting wsproto>=0.14 (from trio-websocket~=0.9->selenium)
  Downloading wsproto-1.2.0-py3-none-any.whl.metadata (5.6 kB)
Requirement already satisfied: pysocks!=1.5.7,<2.0,>=1.5.6 in /usr/local/lib/python3.11/dist-packages (from urllib3[socks]<3,>=1.26->sel
Requirement already satisfied: h11<1,>=0.9.0 in /usr/local/lib/python3.11/dist-packages (from wsproto>=0.14->trio-websocket~=0.9->seleni
Downloading selenium-4.29.0-py3-none-any.whl (9.5 MB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 9.5/9.5 MB 62.5 MB/s eta 0:00:00
Downloading trio-0.29.0-py3-none-any.whl (492 kB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 492.9/492.9 kB 26.6 MB/s eta 0:00:00
Downloading trio_websocket-0.12.2-py3-none-any.whl (21 kB)
Downloading outcome-1.3.0.post0-py2.py3-none-any.whl (10 kB)
Downloading wsproto-1.2.0-py3-none-any.whl (24 kB)
Installing collected packages: wsproto, outcome, trio, trio-websocket, selenium
Successfully installed outcome-1.3.0.post0 selenium-4.29.0 trio-0.29.0 trio-websocket-0.12.2 wsproto-1.2.0
```

```
import requests
from bs4 import BeautifulSoup
import json
import re
import csv
import time

from selenium import webdriver
from selenium.webdriver.chrome.options import Options
from selenium.common.exceptions import TimeoutException, WebDriverException

# Base url of Daily Mail News website
base_url = "https://www.dailymail.co.uk"
# Search url for news articles
search_url = f"{base_url}/home/search.html"

headers = {
    "accept": "text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/apng,*/*;q=0.8,application/signed-exchange;v
    "accept-encoding": "gzip, deflate, br, zstd",
    "accept-language": "en-US,en;q=0.9",
    "cache-control": "max-age=0, no-cache, no-store",
    "cookie": (
        "dm_clientsegment=c; _ga=GA1.1.920993499.1742001073; blaize_session=325103a1-fa7c-477a-8231-f1655c15a09c; "
        "blaize_tracking_id=fe50a3a4-8328-487e-ad82-793f022b8192; _gcl_au=1.1.818330132.1742001073; DM_SitId845=1; "
        "DM_SitId845SecId11530=1; mol.ads.visits=0; mol.ads.visitsExpire=Tue, 15 Apr 2025 01:11:14 GMT; usprivacy=1YNN; "
        "uuid=75ec4cd0-8cbb-402f-8f2d-11f2ff1ac591; _li_dcdm_c=.dailymail.co.uk; _lc2_fpi=50b86ba02c46--01jpbnnw4gjcwatc875nwjrvzp; "
        "_lc2_fpi_js=50b86ba02c46--01jpbnnw4gjcwatc875nwjrvzp; _pubcid=81af5535-b661-48a4-947a-071f809a73e1; "
        "pbjs-unifiedid=%7B%22TDID%22%3A%22e60b74ba-f4cc-4377-9a51-df5cf9d838fa%22%2C%22TDID_LOOKUP%22%3A%22TRUE%22%2C%22TDID_CREATED_AT%22%3
        "pbjs-unifiedid_cst=3yxgLFoszg%3D%3D; _lc2_fpi_meta=%7B%22w%22%3A1742001076503%7D; _lr_env_src_ats=false; "
        "__idcontext=eyJjb29raWVJRCI6IjJ1S2RDelNpdGxXZFFqWVFCVHgxSEU0RFhoSyIsImRldmljZUlEIjoiMnVLZEN3VkJidzFKZWlhaU92VmE5ejV5dHRSIiwiaXYiOiIi
        "panoramaId_expiry=1742605876095; _cc_id=b44859ccddc1206841a20a8c20ed8c9b; panoramaId=22e5322e385bfd50d3c32c04cab8185ca02cebd701bec27
        "_pubcid=81af5535-b661-48a4-947a-071f809a73e1; JSESSIONID=9F3E07D52294AFF427B1E00064A2AFCC; DM_SitId845SecId4626=1; "
        "permutive-id=06ec1b79-cef4-4121-a8dd-effe63f75698; cnx_userId=2-d1839001a5b64fdeb6e7fa952cd0ce70; _iiq_ab_map=%7B%2295%22%3A%22A%22%
        "_au_1d=AU1D-0100-001742001141-4TO7KK6Q-KGYW; _iiq_fdata=%7B%22pcid%22%3A%229cb73a89-acbc-60e6-231a-8a992b190ce1%22%2C%22pcidDate%22%
        "_au_1d=AU1D-0100-001742001141-4TO7KK6Q-KGYW; DM_SitId845SecId4637=1; _li_ss=CrYBCgYI-QEQpRoKBgj3ARClGgoFCAoQpRoKBgjdARClGgoGCPgBEKUa
    ),
    "priority": "u=0, i",
    "referer": "https://www.dailymail.co.uk/home/search.html?offset=0&size=50&sel=site&searchPhrase=tesla&sort=recent&channel=news&type=artic
    "sec-ch-ua": '"Not(A:Brand";v="99", "Google Chrome";v="133", "Chromium";v="133"',
    "sec-ch-ua-mobile": "?0",
    "sec-ch-ua-platform": '"Windows"',
```

```
        "sec-fetch-dest": "document",
        "sec-fetch-mode": "navigate",
        "sec-fetch-site": "same-origin",
        "upgrade-insecure-requests": "1",
        "user-agent": "Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/133.0.0.0 Safari/537.36"
    }

    # List to store all the fetched news links
    all_news_links = []
    # List to hold rows for CSV output
    csv_rows = []

    # --- Part 1: Scrape search results using Requests and Beautiful Soup ---
    with requests.Session() as session:
        for offset in range(0, 151, 50):
            params = {
                "offset": offset,
                "size": "50",
                "sel": "site",
                "searchPhrase": "tesla",
                "sort": "recent",
                "channel": "news",
                "type": "article",
                "topic": "Tesla",
                "days": "last90days"
            }
            try:
                response = session.get(search_url, headers=headers, params=params, timeout=10)
                response.raise_for_status()
            except requests.exceptions.RequestException as e:
                print(f"Error fetching search results at offset {offset}")
                continue

            html = response.text
            soup = BeautifulSoup(html, "html.parser")
            for container in soup.find_all("div", class_="sch-result"):
                title_section = container.find("h3", class_="sch-res-title")
                if title_section:
                    a_tag = title_section.find("a", href=True)
                    if a_tag:
                        all_news_links.append(a_tag["href"])

    # --- Part 2: Set up Selenium for article pages ---
    chrome_options = Options()
    chrome_options.add_argument("--headless")
    chrome_options.add_argument("--no-sandbox")
    chrome_options.add_argument("--disable-dev-shm-usage")

    try:
        driver = webdriver.Chrome(options=chrome_options)
        # Optional: set a page load timeout
        driver.set_page_load_timeout(30)
    except WebDriverException as e:
        print(f"Error creating Selenium WebDriver")
        exit(1)

    # --- Part 3: Process each article page with Selenium ---
    with requests.Session() as session:
        for link in all_news_links:
            full_url = link if link.startswith("http") else base_url + link
            print("\nFetching article details from:", full_url)
            try:
                driver.get(full_url)
                # Allow some extra time for dynamic content to load
                time.sleep(2)
                article_html = driver.page_source
            except TimeoutException as e:
                print(f"Timeout loading page {full_url}")
                continue
            except Exception as e:
                print(f"Error loading page {full_url}")
                continue

            article_soup = BeautifulSoup(article_html, "html.parser")

            # Extract JSON-LD details for article metadata
            ld_script = article_soup.find("script", type="application/ld+json")
```

```
            if ld_script:
                try:
                    ld_data = json.loads(ld_script.string.strip())
                except Exception as e:
                    print(f"Error parsing JSON-LD from {full_url}")
                    continue
                date_published = ld_data.get("datePublished", "N/A")
                author = ld_data.get("author")
                if isinstance(author, dict):
                    author_name = author.get("name", "N/A")
                elif isinstance(author, list):
                    author_name = ", ".join(item.get("name", "N/A") for item in author)
                else:
                    author_name = "N/A"
            else:
                date_published = "N/A"
                author_name = "N/A"

            # Extract article content
            article_body_div = article_soup.find("div", itemprop="articleBody")
            paragraphs = []
            if article_body_div:
                for p in article_body_div.find_all("p"):
                    paragraphs.append(p.get_text(strip=True))
            article_text = " ".join(paragraphs)

            csv_rows.append({
                "type": "news",
                "content": article_text,
                "createdAt": date_published,
                "Author Name": author_name,
                "url": full_url
            })

            # --- Fetch comments ---
            match = re.search(r'article-(\d+)', full_url)
            if match:
                article_id = match.group(1)
                comments_url = f"{base_url}/reader-comments/p/asset/readcomments/{article_id}?max=999&offset=0&order=desc"
                comments_headers = {
                    "User-Agent": headers["user-agent"],
                    "Accept": "application/json, text/plain, */*",
                    "Referer": full_url,
                    "sec-ch-ua": headers["sec-ch-ua"],
                    "sec-ch-ua-mobile": headers["sec-ch-ua-mobile"],
                    "sec-ch-ua-platform": headers["sec-ch-ua-platform"]
                }
                try:
                    comments_response = session.get(comments_url, headers=comments_headers, timeout=10)
                    comments_response.raise_for_status()
                    comments_data = comments_response.json()
                except requests.exceptions.RequestException as e:
                    print(f"Error fetching comments from {comments_url}")
                    continue

                # Process and save comments
                def process_comment(comment, indent=0):
                    user_alias = comment.get("userAlias", "")
                    message = comment.get("message", "")
                    date_created = comment.get("dateCreated", "")
                    csv_rows.append({
                        "type": "comment",
                        "content": message,
                        "createdAt": date_created,
                        "Author Name": user_alias,
                        "url": full_url
                    })
                    replies = comment.get("replies", {}).get("comments", [])
                    for reply in replies:
                        process_comment(reply, indent=indent+4)

                for comment in comments_data.get("payload", {}).get("page", []):
                    process_comment(comment)
            else:
                print("No article ID found in URL for comments fetching.")

    # --- Cleanup Selenium ---
    try:
```

```
    try:
        driver.quit()
    except Exception as e:
        print(f"Error quitting WebDriver")

    # --- Part 4: Save CSV ---
    csv_filename = "output.csv"
    try:
        with open(csv_filename, mode="w", newline="", encoding="utf-8") as csvfile:
            fieldnames = ["type", "content", "createdAt", "Author Name", "url"]
            writer = csv.DictWriter(csvfile, fieldnames=fieldnames)
            writer.writeheader()
            for row in csv_rows:
                writer.writerow(row)
        print(f"\nData has been saved to {csv_filename}")
    except Exception as e:
        print(f"Error writing CSV file")
```

```
Fetching article details from: https://www.dailymail.co.uk/news/article-14479011/Jennifer-Saunders-Adrian-Edmondsons-Absolutely-Fabul

Fetching article details from: https://www.dailymail.co.uk/news/article-14501799/elon-musk-mars-spacex-starship-optimus-2026.html

Fetching article details from: https://www.dailymail.co.uk/news/article-14501189/Met-Police-thousands-protesters-descend-London-weeke

Fetching article details from: https://www.dailymail.co.uk/news/article-14500525/mark-kelly-senator-selling-tesla-elon-musk.html

Fetching article details from: https://www.dailymail.co.uk/news/article-14497841/James-Murdoch-Tesla.html

Fetching article details from: https://www.dailymail.co.uk/news/article-14498041/Thief-smashed-Tesla-stole-1-350-Stone-Island-jacket-

Fetching article details from: https://www.dailymail.co.uk/news/article-14497375/elon-musk-tesla-letter-donald-trump-tariffs-stock-pr

Fetching article details from: https://www.dailymail.co.uk/news/article-14493705/Tesla-records-cyclist-smashes-window-steal-1000-Ston
Timeout loading page https://www.dailymail.co.uk/news/article-14493705/Tesla-records-cyclist-smashes-window-steal-1000-Stone-Island-j

Fetching article details from: https://www.dailymail.co.uk/news/article-14492077/Trumps-brand-new-red-Tesla-recalled-revealed-wholl-a

Fetching article details from: https://www.dailymail.co.uk/news/article-14490375/Elon-Musks-Optimus-robot-targeted-Just-Stop-Oil.html

Fetching article details from: https://www.dailymail.co.uk/news/article-14483677/Elon-Musks-DOGE-approval-rating-revealed-voters-pred

Fetching article details from: https://www.dailymail.co.uk/news/article-14488877/Elon-Musk-trans-daughter-Vivian-Wilson-IVF-shock-cla

Fetching article details from: https://www.dailymail.co.uk/news/article-14488561/Trump-recession-stock-market-trade-war-mexico-canada

Fetching article details from: https://www.dailymail.co.uk/news/article-14488025/donald-trump-tesla-domestic-terrorism-elon-musk.html
Timeout loading page https://www.dailymail.co.uk/news/article-14488025/donald-trump-tesla-domestic-terrorism-elon-musk.html

Fetching article details from: https://www.dailymail.co.uk/news/article-14487443/Trump-tests-brand-new-Teslas-White-House-lawn-backs-

Fetching article details from: https://www.dailymail.co.uk/news/article-14487043/lara-trump-gushes-elon-musk-reveals-kissing-feet.htm
Timeout loading page https://www.dailymail.co.uk/news/article-14487043/lara-trump-gushes-elon-musk-reveals-kissing-feet.html

Fetching article details from: https://www.dailymail.co.uk/debate/article-14485697/Elon-Musk-liberals-Telsa-Cybertrucks.html
Timeout loading page https://www.dailymail.co.uk/debate/article-14485697/Elon-Musk-liberals-Telsa-Cybertrucks.html

Fetching article details from: https://www.dailymail.co.uk/news/article-14483177/Elon-Musk-tears-George-Soros-accuses-Reid-Hoffman-Ep

Fetching article details from: https://www.dailymail.co.uk/news/article-14484697/Donald-Trump-buy-brand-new-Tesla-support-Elon-Musk-b
Timeout loading page https://www.dailymail.co.uk/news/article-14484697/Donald-Trump-buy-brand-new-Tesla-support-Elon-Musk-baby.html

Fetching article details from: https://www.dailymail.co.uk/news/article-14481765/Anti-Musk-protestors-attack-Tesla-showrooms-Molotov-

Fetching article details from: https://www.dailymail.co.uk/news/article-14478697/snl-skit-elon-musk-dr-evil.html

Fetching article details from: https://www.dailymail.co.uk/news/article-14471951/Trump-picks-sides-MAGA-power-struggle-Elon-Musk-Cabi
Timeout loading page https://www.dailymail.co.uk/news/article-14471951/Trump-picks-sides-MAGA-power-struggle-Elon-Musk-Cabinet.html

Fetching article details from: https://www.dailymail.co.uk/news/article-14466507/The-brutal-truth-Anthony-Albaneses-failing-masterpla
Timeout loading page https://www.dailymail.co.uk/news/article-14466507/The-brutal-truth-Anthony-Albaneses-failing-masterplan-drive-Au

Fetching article details from: https://www.dailymail.co.uk/news/article-14469485/elon-musk-daylight-savings-poll-furious-debate-ameri

Fetching article details from: https://www.dailymail.co.uk/news/article-14468785/elon-musk-gesture-cancer-dj-daniel-target-msnbc-nico
```

Sentiment Analysis of Scraped Articles

```
pip install vader
```

```
Collecting vader
  Downloading vader-0.0.3-py3-none-any.whl.metadata (2.3 kB)
Requirement already satisfied: scikit-learn in /usr/local/lib/python3.11/dist-packages (from vader) (1.6.1)
Requirement already satisfied: numpy in /usr/local/lib/python3.11/dist-packages (from vader) (1.26.4)
Requirement already satisfied: scipy in /usr/local/lib/python3.11/dist-packages (from vader) (1.14.1)
Collecting sonopy (from vader)
  Downloading sonopy-0.1.2.tar.gz (3.3 kB)
  Preparing metadata (setup.py) ... done
Requirement already satisfied: joblib>=1.2.0 in /usr/local/lib/python3.11/dist-packages (from scikit-learn->vader) (1.4.2)
Requirement already satisfied: threadpoolctl>=3.1.0 in /usr/local/lib/python3.11/dist-packages (from scikit-learn->vader) (3.5.0)
Downloading vader-0.0.3-py3-none-any.whl (4.8 MB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 4.8/4.8 MB 41.0 MB/s eta 0:00:00
Building wheels for collected packages: sonopy
  Building wheel for sonopy (setup.py) ... done
  Created wheel for sonopy: filename=sonopy-0.1.2-py3-none-any.whl size=2851 sha256=ac044f4593dba074f9b09723df9453c801ad55a00c7a8da9c515
  Stored in directory: /root/.cache/pip/wheels/6e/02/0d/df138747348c15908c1fb09493064ead497e16e187e3094d71
Successfully built sonopy
Installing collected packages: sonopy, vader
Successfully installed sonopy-0.1.2 vader-0.0.3
```

```
import os
import glob
import pandas as pd
import nltk
import shutil
from nltk.sentiment.vader import SentimentIntensityAnalyzer

# To download the VADER lexicon
nltk.download('vader_lexicon')

# Input folder path containing the CSV or XLSX files
input_folder = '/content/Articles_content'

# Output folder for saving processed output CSV files
output_folder = '/content/Articles_content_with_sentiment'
os.makedirs(output_folder, exist_ok=True)

# Initialize the VADER sentiment analyzer
sid = SentimentIntensityAnalyzer()

# Text data preprocessing (stripping extra whitespace)
def data_preprocessing(text):
    if isinstance(text, str):
        return text.strip()
    return text

# Get all the CSV and XLSX files in the input folder
csv_files = glob.glob(os.path.join(input_folder, "*.csv"))
xlsx_files = glob.glob(os.path.join(input_folder, "*.xlsx"))
files = csv_files + xlsx_files

# Iterate over each file for performing sentiment analysis
for file in files:
    filename = os.path.basename(file)
    name, ext = os.path.splitext(filename)

    if ext.lower() == ".csv":
        df = pd.read_csv(file)
    elif ext.lower() in [".xlsx", ".xls"]:
        df = pd.read_excel(file)
    else:
        continue  # Skip files of other formats

    # Calculate the compound sentiment score for each row's "content"
    df['sentiment'] = df['content'].apply(
        lambda text: sid.polarity_scores(data_preprocessing(text))['compound']
        if isinstance(text, str) else None
    )

    print(f"Sentiment analysis for {filename}:")
    print(df[['content', 'sentiment']].head())

    # Save the updated DataFrame to a CSV file in output folder
    output_file = os.path.join(output_folder, f"{name}_with_sentiment.csv")
    df.to_csv(output_file, index=False)
```

```
    print(f"Output saved to {output_file}")

# Compress to zip and download output folder (running on local)
zip_filename = os.path.basename(os.path.normpath(output_folder)) + '.zip'
shutil.make_archive(os.path.splitext(zip_filename)[0], 'zip', output_folder)
print(f"\nOutput folder compressed to {zip_filename}")

# If running in google colab
try:
    from google.colab import files
    files.download(zip_filename)
except ImportError:
    print("Not running in Google Colab. Please manually download the ZIP file from your file system.")
```

```
[nltk_data] Downloading package vader_lexicon to /root/nltk_data...
[nltk_data]   Package vader_lexicon is already up-to-date!
Sentiment analysis for DailyMail_Articles.csv:
                                             content  sentiment
0  Salty people with TDS don't know new cars from...     0.0000
1  I’m sure you are happy to pay 100k for a car w...     0.8910
2  He certainly didn’t instill confidence for the...     0.6908
3  The most successful businessman on the planet,...    -0.6915
4  Plus SpaceX was canceled today 30 minutes befo...     0.0000
Output saved to /content/Articles_content_with_sentiment/DailyMail_Articles_with_sentiment.csv
Sentiment analysis for reddit_tslamotors_elonmusk_Jan1_Mar12.xlsx:
                                             content  sentiment
0  Yes, not in the U.S.. It used to be a Toyota C...     0.4019
1  I think this is many people's want, would make...     0.5719
2        Is it limited by the pack or the inverters?    -0.2263
3  I'm 6'4" and 225lbs, drove both, and prefer th...     0.8658
4  He literally described every hard working emig...    -0.2023
Output saved to /content/Articles_content_with_sentiment/reddit_tslamotors_elonmusk_Jan1_Mar12_with_sentiment.csv
Sentiment analysis for reddit_tsla_Jan1_Mar12.xlsx:
                                             content  sentiment
0  The delivery number has been revised downward ...     0.6705
1  "I'll buy when it hits $100"\n\nVibes 6 months...     0.0000
2  If they lose their EV credit AND lower the pri...    -0.1779
3                      Youâ€™re delusional. No clue    -0.2960
4                                Why not just hold?     0.0000
Output saved to /content/Articles_content_with_sentiment/reddit_tsla_Jan1_Mar12_with_sentiment.csv

Output folder compressed to Articles_content_with_sentiment.zip
```

YouTube

```
pip install youtube_transcript_api
```

```
Collecting youtube_transcript_api
  Downloading youtube_transcript_api-1.0.1-py3-none-any.whl.metadata (22 kB)
Requirement already satisfied: defusedxml<0.8.0,>=0.7.1 in /usr/local/lib/python3.11/dist-packages (from youtube_transcript_api) (0.7.1)
Requirement already satisfied: requests in /usr/local/lib/python3.11/dist-packages (from youtube_transcript_api) (2.32.3)
Requirement already satisfied: charset-normalizer<4,>=2 in /usr/local/lib/python3.11/dist-packages (from requests->youtube_transcript_ap
Requirement already satisfied: idna<4,>=2.5 in /usr/local/lib/python3.11/dist-packages (from requests->youtube_transcript_api) (3.10)
Requirement already satisfied: urllib3<3,>=1.21.1 in /usr/local/lib/python3.11/dist-packages (from requests->youtube_transcript_api) (2.
Requirement already satisfied: certifi>=2017.4.17 in /usr/local/lib/python3.11/dist-packages (from requests->youtube_transcript_api) (20
Downloading youtube_transcript_api-1.0.1-py3-none-any.whl (1.9 MB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 1.9/1.9 MB 22.7 MB/s eta 0:00:00
Installing collected packages: youtube_transcript_api
Successfully installed youtube_transcript_api-1.0.1
```

Scraping News URLs via Youtube API

```
import datetime
import pandas as pd
import time
from googleapiclient.discovery import build
from googleapiclient.errors import HttpError

# YouTube Data API key
api_key = "AIzaSyCzduWRq-77kXXbsUt8WCUOZPONfo1UPC0"

# YouTube API client to interact with data
youtube = build('youtube', 'v3', developerKey=api_key, cache_discovery=False)
```

```
# Start and End Date range
start_date = datetime.date(2025, 1, 1)
end_date = datetime.date(2025, 3, 12)
delta = datetime.timedelta(days=1)

# List to store rows of data
rows = []

current_date = start_date
while current_date <= end_date:
    # Define the time window for the current day
    published_after = current_date.isoformat() + "T00:00:00Z"
    next_date = current_date + delta
    published_before = next_date.isoformat() + "T00:00:00Z"

    try:
        # Perform a search query for "Tesla news" to obtain videos with captions enabled (to fetch transcripts)
        request = youtube.search().list(
            q="Tesla news",
            part="snippet",
            type="video",
            videoCaption="closedCaption",
            maxResults=1,
            order="date",
            publishedAfter=published_after,
            publishedBefore=published_before
        )
        response = request.execute()
        # To avoid rate limits
        time.sleep(1)

        if response.get('items'):
            video_id = response['items'][0]['id']['videoId']
            video_url = f"https://www.youtube.com/watch?v={video_id}"
            print(f"{current_date}: {video_url}")
        else:
            video_url = ""
            print(f"{current_date}: No video found")
    except HttpError as e:
        print(f"HttpError on {current_date}: {e}")
        video_url = ""
        # Longer wait time if an error occurs
        time.sleep(10)

    # Append the result for the current day
    rows.append({"date": current_date.isoformat(), "url": video_url})
    current_date = next_date

# Save the DataFrame to a CSV file
df = pd.DataFrame(rows)
output_csv = "youtubeVideo_links.csv"
df.to_csv(output_csv, index=False)
print(f"\nCSV file saved as {output_csv}")

# To download the CSV file from Google Colab
try:
    from google.colab import files
    files.download(output_csv)
except ImportError:
    print("Not running in Colab. Please download the file manually.")
```

```
2025-01-01: https://www.youtube.com/watch?v=Q4HnFkmRDCw
2025-01-02: https://www.youtube.com/watch?v=mroFnhe6olo
2025-01-03: https://www.youtube.com/watch?v=bQyx0F2bT9A
2025-01-04: https://www.youtube.com/watch?v=JqxtqfjvBMA
2025-01-05: https://www.youtube.com/watch?v=y1jrt5jKApI
2025-01-06: https://www.youtube.com/watch?v=3Brp8xn1ko4
2025-01-07: https://www.youtube.com/watch?v=ukkP8GtN6pk
2025-01-08: https://www.youtube.com/watch?v=SBVNzZjoI50
2025-01-09: https://www.youtube.com/watch?v=Q4Gh2n2HBmc
2025-01-10: https://www.youtube.com/watch?v=xjv5uNwmHbI
2025-01-11: https://www.youtube.com/watch?v=GgdKgb5Le1Y
2025-01-12: https://www.youtube.com/watch?v=E9K7v4V2ZSc
2025-01-13: https://www.youtube.com/watch?v=-n0qGeBwpTo
2025-01-14: https://www.youtube.com/watch?v=nDTvv65b8Ys
2025-01-15: https://www.youtube.com/watch?v=TJ2T_ITlexo
2025-01-16: https://www.youtube.com/watch?v=ekGzJgNQrTM
2025-01-17: https://www.youtube.com/watch?v=45FccKLoNzc
2025-01-18: https://www.youtube.com/watch?v=cxNE0hqAfcs
2025-01-19: https://www.youtube.com/watch?v=OF-QagxsCcU
2025-01-20: https://www.youtube.com/watch?v=3dRVBKm2BH0
2025-01-21: https://www.youtube.com/watch?v=5rnzskmZVB8
2025-01-22: https://www.youtube.com/watch?v=yTz2sIv_v68
2025-01-23: https://www.youtube.com/watch?v=T9UvsMEQotM
2025-01-24: https://www.youtube.com/watch?v=1iG8ir-ztdo
2025-01-25: https://www.youtube.com/watch?v=bvYQTf28zXQ
2025-01-26: No video found
2025-01-27: https://www.youtube.com/watch?v=9mWzibZwocg
2025-01-28: https://www.youtube.com/watch?v=uouhMdlE2rg
2025-01-29: https://www.youtube.com/watch?v=MAXyygpUor0
2025-01-30: https://www.youtube.com/watch?v=_ztSlF_J6rA
2025-01-31: https://www.youtube.com/watch?v=sOzepEHLR80
2025-02-01: https://www.youtube.com/watch?v=dN9ZGNfkZSg
2025-02-02: https://www.youtube.com/watch?v=0xLewVeDpUQ
2025-02-03: https://www.youtube.com/watch?v=pQBKoWfclEM
2025-02-04: https://www.youtube.com/watch?v=UDM_x6Snq-c
2025-02-05: https://www.youtube.com/watch?v=wvWKb52cZ28
2025-02-06: https://www.youtube.com/watch?v=LIP0-oLhAQQ
2025-02-07: https://www.youtube.com/watch?v=kh1p4GDDXbI
2025-02-08: https://www.youtube.com/watch?v=jJf18gsxFFk
2025-02-09: https://www.youtube.com/watch?v=oEWYw3JUfME
2025-02-10: https://www.youtube.com/watch?v=SpgQ-GWItiE
2025-02-11: https://www.youtube.com/watch?v=WWNJtjqgURU
2025-02-12: https://www.youtube.com/watch?v=9UWAlOvFEbI
2025-02-13: https://www.youtube.com/watch?v=Yv9gzhoT59E
2025-02-14: https://www.youtube.com/watch?v=2QKwZyK56nA
2025-02-15: https://www.youtube.com/watch?v=0immYy4mVzQ
2025-02-16: https://www.youtube.com/watch?v=SRzKCm0hmjU
2025-02-17: https://www.youtube.com/watch?v=vSyRtyk6Cxo
2025-02-18: https://www.youtube.com/watch?v=iMlPHsa5XdU
2025-02-19: https://www.youtube.com/watch?v=dkdUfSzdCDM
2025-02-20: https://www.youtube.com/watch?v=nGRSgXObUc8
2025-02-21: https://www.youtube.com/watch?v=8vI-5zzNkP0
2025-02-22: https://www.youtube.com/watch?v=w21TLQOIThk
2025-02-23: https://www.youtube.com/watch?v=QA-laFwTCoc
2025-02-24: https://www.youtube.com/watch?v=z40rbOLUi-w
2025-02-25: https://www.youtube.com/watch?v=rLgWLXzzM0Q
2025-02-26: https://www.youtube.com/watch?v=44UsjFNQt4Q
2025-02-27: https://www.youtube.com/watch?v=W_aUPoFkULs
2025-02-28: https://www.youtube.com/watch?v=gut_Gys7Pbo
2025-03-01: https://www.youtube.com/watch?v=Kx2DG4YtbF4
2025-03-02: https://www.youtube.com/watch?v=iHBf8Vb8bmY
2025-03-03: https://www.youtube.com/watch?v=Pus_mbDP4Zo
2025-03-04: https://www.youtube.com/watch?v=cKHi4cFXAc0
2025-03-05: https://www.youtube.com/watch?v=CWze2xQaqcw
2025-03-06: https://www.youtube.com/watch?v=Y9xwZRV12Nw
2025-03-07: https://www.youtube.com/watch?v=Ccpr1TOzzI0
2025-03-08: https://www.youtube.com/watch?v=UfyPtvZGHAI
2025-03-09: https://www.youtube.com/watch?v=BI84E354Yc8
2025-03-10: https://www.youtube.com/watch?v=AwoBakHU2vQ
2025-03-11: https://www.youtube.com/watch?v=QYU0n9F7pHk
2025-03-12: https://www.youtube.com/watch?v=_fgZ-idXUeY

CSV file saved as youtubeVideo_links.csv
```

Scraping Transcript and Comments of Youtube News Videos on Tesla

```
pip install googletrans==4.0.0-rc1
```

```
Collecting googletrans==4.0.0-rc1
  Downloading googletrans-4.0.0rc1.tar.gz (20 kB)
  Preparing metadata (setup.py) ... done
Collecting httpx==0.13.3 (from googletrans==4.0.0-rc1)
  Downloading httpx-0.13.3-py3-none-any.whl.metadata (25 kB)
Requirement already satisfied: certifi in /usr/local/lib/python3.11/dist-packages (from httpx==0.13.3->googletrans==4.0.0-rc1) (2025.
Collecting hstspreload (from httpx==0.13.3->googletrans==4.0.0-rc1)
  Downloading hstspreload-2025.1.1-py3-none-any.whl.metadata (2.1 kB)
Requirement already satisfied: sniffio in /usr/local/lib/python3.11/dist-packages (from httpx==0.13.3->googletrans==4.0.0-rc1) (1.3.1
Collecting chardet==3.* (from httpx==0.13.3->googletrans==4.0.0-rc1)
  Downloading chardet-3.0.4-py2.py3-none-any.whl.metadata (3.2 kB)
Collecting idna==2.* (from httpx==0.13.3->googletrans==4.0.0-rc1)
  Downloading idna-2.10-py2.py3-none-any.whl.metadata (9.1 kB)
Collecting rfc3986<2,>=1.3 (from httpx==0.13.3->googletrans==4.0.0-rc1)
  Downloading rfc3986-1.5.0-py2.py3-none-any.whl.metadata (6.5 kB)
Collecting httpcore==0.9.* (from httpx==0.13.3->googletrans==4.0.0-rc1)
  Downloading httpcore-0.9.1-py3-none-any.whl.metadata (4.6 kB)
Collecting h11<0.10,>=0.8 (from httpcore==0.9.*->httpx==0.13.3->googletrans==4.0.0-rc1)
  Downloading h11-0.9.0-py2.py3-none-any.whl.metadata (8.1 kB)
Collecting h2==3.* (from httpcore==0.9.*->httpx==0.13.3->googletrans==4.0.0-rc1)
  Downloading h2-3.2.0-py2.py3-none-any.whl.metadata (32 kB)
Collecting hyperframe<6,>=5.2.0 (from h2==3.*->httpcore==0.9.*->httpx==0.13.3->googletrans==4.0.0-rc1)
  Downloading hyperframe-5.2.0-py2.py3-none-any.whl.metadata (7.2 kB)
Collecting hpack<4,>=3.0 (from h2==3.*->httpcore==0.9.*->httpx==0.13.3->googletrans==4.0.0-rc1)
  Downloading hpack-3.0.0-py2.py3-none-any.whl.metadata (7.0 kB)
Downloading httpx-0.13.3-py3-none-any.whl (55 kB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 55.1/55.1 kB 3.7 MB/s eta 0:00:00
Downloading chardet-3.0.4-py2.py3-none-any.whl (133 kB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 133.4/133.4 kB 6.9 MB/s eta 0:00:00
Downloading httpcore-0.9.1-py3-none-any.whl (42 kB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 42.6/42.6 kB 3.0 MB/s eta 0:00:00
Downloading idna-2.10-py2.py3-none-any.whl (58 kB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 58.8/58.8 kB 4.7 MB/s eta 0:00:00
Downloading h2-3.2.0-py2.py3-none-any.whl (65 kB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 65.0/65.0 kB 5.6 MB/s eta 0:00:00
Downloading rfc3986-1.5.0-py2.py3-none-any.whl (31 kB)
Downloading hstspreload-2025.1.1-py3-none-any.whl (1.3 MB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 1.3/1.3 MB 33.8 MB/s eta 0:00:00
Downloading h11-0.9.0-py2.py3-none-any.whl (53 kB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 53.6/53.6 kB 4.4 MB/s eta 0:00:00
Downloading hpack-3.0.0-py2.py3-none-any.whl (38 kB)
Downloading hyperframe-5.2.0-py2.py3-none-any.whl (12 kB)
Building wheels for collected packages: googletrans
  Building wheel for googletrans (setup.py) ... done
  Created wheel for googletrans: filename=googletrans-4.0.0rc1-py3-none-any.whl size=17397 sha256=6374b77ec8348d440ab082a575257366c5e
  Stored in directory: /root/.cache/pip/wheels/39/17/6f/66a045ea3d168826074691b4b787b8f324d3f646d755443fda
Successfully built googletrans
Installing collected packages: rfc3986, hyperframe, hpack, h11, chardet, idna, hstspreload, h2, httpcore, httpx, googletrans
  Attempting uninstall: hyperframe
    Found existing installation: hyperframe 6.1.0
    Uninstalling hyperframe-6.1.0:
      Successfully uninstalled hyperframe-6.1.0
  Attempting uninstall: hpack
    Found existing installation: hpack 4.1.0
    Uninstalling hpack-4.1.0:
      Successfully uninstalled hpack-4.1.0
  Attempting uninstall: h11
    Found existing installation: h11 0.14.0
    Uninstalling h11-0.14.0:
      Successfully uninstalled h11-0.14.0
  Attempting uninstall: chardet
    Found existing installation: chardet 5.2.0
    Uninstalling chardet-5.2.0:
      Successfully uninstalled chardet-5.2.0
  Attempting uninstall: idna
    Found existing installation: idna 3.10
    Uninstalling idna-3.10:
      Successfully uninstalled idna-3.10
  Attempting uninstall: h2
    Found existing installation: h2 4.2.0
    Uninstalling h2-4.2.0:
      Successfully uninstalled h2-4.2.0
  Attempting uninstall: httpcore
    Found existing installation: httpcore 1.0.7
    Uninstalling httpcore-1.0.7:
      Successfully uninstalled httpcore-1.0.7
  Attempting uninstall: httpx
    Found existing installation: httpx 0.28.1
    Uninstalling httpx-0.28.1:
      Successfully uninstalled httpx-0.28.1
ERROR: pip's dependency resolver does not currently take into account all the packages that are installed. This behaviour is the sour
langsmith 0.3.13 requires httpx<1,>=0.23.0, but you have httpx 0.13.3 which is incompatible.
google-genai 1.4.0 requires httpx<1.0.0dev,>=0.28.1, but you have httpx 0.13.3 which is incompatible.
openai 1.61.1 requires httpx<1,>=0.23.0, but you have httpx 0.13.3 which is incompatible.
```

```
Successfully installed chardet-3.0.4 googletrans-4.0.0rc1 h11-0.9.0 h2-3.2.0 hpack-3.0.0 hstspreload-2025.1.1 httpcore-0.9.1 httpx-0.
WARNING: The following packages were previously imported in this runtime:
  [chardet,idna]
You must restart the runtime in order to use newly installed versions.
```

RESTART SESSION

```
import re
import csv
import requests
import matplotlib.pyplot as plt
import pandas as pd
import time
import os
import shutil

from youtube_transcript_api import YouTubeTranscriptApi
from googleapiclient.discovery import build
from googleapiclient.errors import HttpError
from googletrans import Translator  # Added for translation

# Extract video ID from various YouTube URL formats
def extract_video_id(url):
    regex = r"(?:v=|\/)([0-9A-Za-z_-]{11}).*"
    match = re.search(regex, url)
    if match:
        return match.group(1)
    else:
        raise ValueError("Invalid YouTube URL format.")

# Retrieve the transcript for a video with translation fallback
def get_transcript(video_id):
    # Try to fetch the English transcript
    try:
        transcript_list = YouTubeTranscriptApi.get_transcript(video_id, languages=['en'])
        transcript_text = " ".join([entry['text'] for entry in transcript_list])
        return transcript_text
    except Exception as e:
        print(f"English transcript not available for video {video_id}: {e}")

    # Try Spanish transcript and translate to English
    try:
        transcript_list = YouTubeTranscriptApi.get_transcript(video_id, languages=['es-419'])
        transcript_text = " ".join([entry['text'] for entry in transcript_list])
        print("Spanish transcript retrieved. Translating to English...")
        translator = Translator()
        translated = translator.translate(transcript_text, src='es', dest='en')
        return translated.text
    except Exception as e2:
        print(f"Spanish transcript not available for video {video_id}: {e2}")
        return None

# Retrieve video comments with individual author and timestamp details
def get_video_comments(video_id, api_key):
    youtube = build('youtube', 'v3', developerKey=api_key, cache_discovery=False)
    comments = []  # List to store comment details
    next_page_token = None

    while True:
        try:
            request = youtube.commentThreads().list(
                part='snippet',
                videoId=video_id,
                textFormat='plainText',
                maxResults=100,
                pageToken=next_page_token
            )
            response = request.execute()
            time.sleep(1)
        except HttpError as e:
            error_msg = str(e)
            if "commentsDisabled" in error_msg:
                print(f"Comments are disabled for video {video_id}.")
                return []
            else:
                print(f"Error fetching comments for {video_id}: {e}")
```

```
                time.sleep(10)
                break

            for item in response.get('items', []):
                snippet = item['snippet']['topLevelComment']['snippet']
                comment_data = {
                    'text': snippet.get('textDisplay', ''),
                    'author': snippet.get('authorDisplayName', 'Unknown'),
                    'timestamp': snippet.get('publishedAt', None)
                }
                comments.append(comment_data)

            next_page_token = response.get('nextPageToken')
            if not next_page_token:
                break

        return comments

    # Retrieve basic video details (author and publication date)
    def get_video_details(video_id, api_key):
        youtube = build('youtube', 'v3', developerKey=api_key, cache_discovery=False)
        try:
            request = youtube.videos().list(
                part='snippet',
                id=video_id
            )
            response = request.execute()
            time.sleep(1)
        except HttpError as e:
            print(f"Error fetching details for {video_id}: {e}")
            time.sleep(10)
            return None

        items = response.get('items', [])
        if not items:
            return None
        snippet = items[0]['snippet']
        video_details = {
            'author': snippet.get('channelTitle', 'Unknown'),
            'createdAt': snippet.get('publishedAt', None)
        }
        return video_details

    # Process a video: extract transcript, comments, and metadata; then return rows
    def process_video(video_url, api_key):
        print(f"\nProcessing video: {video_url}")
        video_id = extract_video_id(video_url)
        print(f"Video ID: {video_id}")

        transcript_text = get_transcript(video_id)
        if transcript_text:
            print("Transcript retrieved.")
        else:
            print("No transcript available.")

        comments = get_video_comments(video_id, api_key)
        print(f"Number of comments retrieved: {len(comments)}")

        video_details = get_video_details(video_id, api_key)
        if video_details:
            video_author = video_details.get('author', 'Unknown')
            video_published_at = video_details.get('createdAt', None)
        else:
            video_author = 'Unknown'
            video_published_at = None

        rows = []
        # Add transcript as a "news" type row
        if transcript_text:
            rows.append({
                'type': 'news',
                'content': transcript_text,
                'timestamp': video_published_at,
                'author': video_author,
                'post_url': video_url,
                'platform': 'YouTube'
            })
```

```
        # Add each comment as a "comment" type row using its own author and timestamp
        for comment in comments:
            rows.append({
                'type': 'comment',
                'content': comment['text'],
                'timestamp': comment['timestamp'],
                'author': comment['author'],
                'post_url': video_url,
                'platform': 'YouTube'
            })

        return rows

    # Analyze YouTube video URLs and merge all data into one final CSV
    def analyze_yt_videos():
        youtube_urls_df = pd.read_csv("youtubeVideo_links.csv")
        youtube_urls = youtube_urls_df['url'].dropna().tolist()

        api_key = "AIzaSyCzduWRq-77kXXbsUt8WCUOZPONfo1UPC0"

        summary_rows = []

        for url in youtube_urls:
            video_rows = process_video(url, api_key)
            summary_rows.extend(video_rows)
            time.sleep(1)

        # Create final DataFrame with specified column order
        summary_df = pd.DataFrame(summary_rows)
        final_columns = ['type', 'content', 'timestamp', 'author', 'post_url', 'platform']
        summary_df = summary_df[final_columns]
        final_csv = "videos_details_summary.csv"
        summary_df.to_csv(final_csv, index=False)
        print(f"Overall summary saved to {final_csv}")
        return final_csv

    # Search for a channel using the YouTube Data API
    def search_channel():
        url = "https://www.googleapis.com/youtube/v3/search"
        params = {
            "part": "snippet",
            "q": "@preyashyadav",
            "type": "channel",
            "key": "AIzaSyCzduWRq-77kXXbsUt8WCUOZPONfo1UPC0"
        }
        try:
            response = requests.get(url, params=params)
            time.sleep(1)
            data = response.json()
            print(data)
        except Exception as e:
            print(f"Error during channel search: {e}")

    # Obtain details about a specific channel using the YouTube Data API
    def get_channel_details():
        channel_id = "UCkDii4wjOVlQAsK1a3kxN1A"
        url = "https://www.googleapis.com/youtube/v3/channels"
        params = {
            "part": "snippet",
            "id": channel_id,
            "key": "AIzaSyCzduWRq-77kXXbsUt8WCUOZPONfo1UPC0"
        }
        try:
            response = requests.get(url, params=params)
            time.sleep(1)
            channel_data = response.json()
            print(channel_data)
        except Exception as e:
            print(f"Error fetching channel details: {e}")

    # Run video analysis and perform queries
    final_csv = analyze_yt_videos()
    search_channel()
    get_channel_details()

    # Download the final CSV file (for example, in Google Colab)
```

```
try:
    from google.colab import files
    files.download(final_csv)
except ImportError:
    print("File download is not available in this environment. Please manually download the CSV file.")
```