Sara Memon

AI Task Report

1. **Al Jazeera English (Qatar) Website:** https://www.aljazeera.com/

Objective: To scrape URLs, descriptions, titles and dates of all the previous and current articles available on the website Al Jazeera and it is a dynamically loaded content.

Methodology: Steps to scrape all the current and previous published articles using Python language, Selenium and BeautifulSoup drivers and generate & save the results in csv file.

Tools Used:

- Jupiter notebook
- Python Language
- Selenium driver (automates the web-browser to scroll and render dynamic website)
- BeautifulSoup (extracts and parses data from fully-loaded HTML)
- WebDriver manager (it automatically downloads and setup the ChromeDriver)

Steps learned & performed:

i. Initialize libraries and use Selenium for GUI-free fast browsing.
ii. Load the Webpage: https://www.aljazeera.com
iii. Scrolling down iteratively to trigger loading of all articles.
iv. Navigate to the pages of article for each URL.
v. Finally, scrape relevant data such as: URL, title, description, and date.
vi. Save scraped data into csv file.

Code:

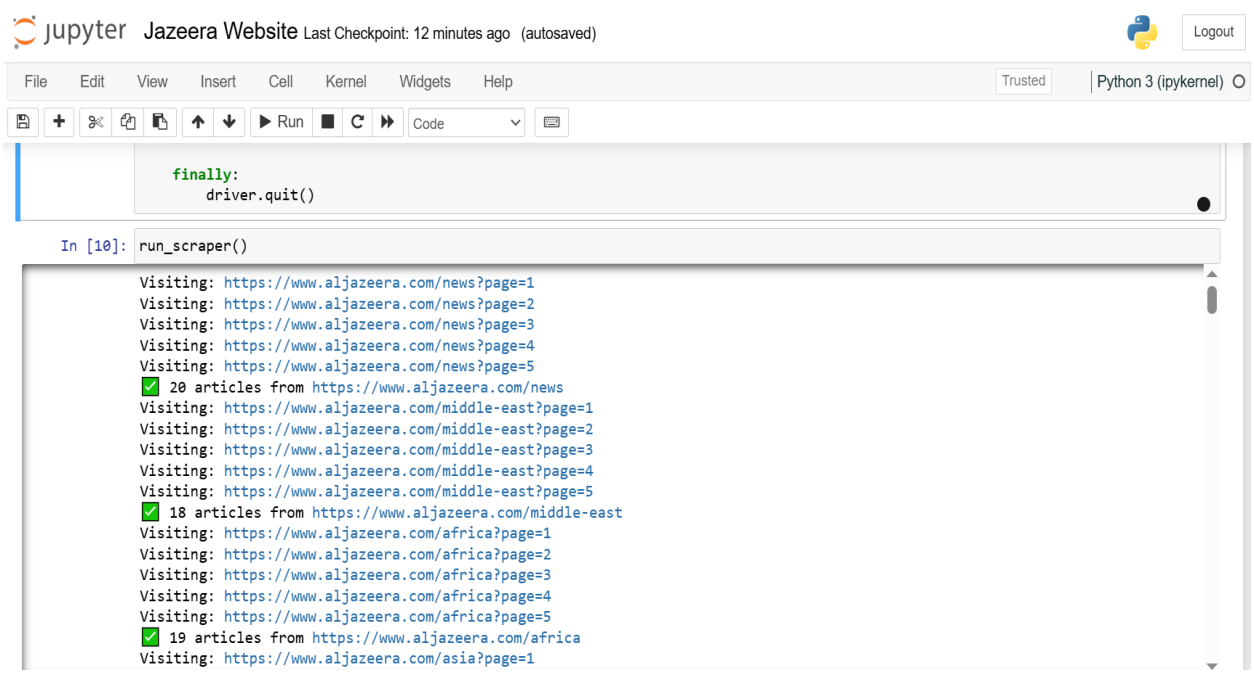
```
!pip install selenium beautifulsoup4 pandas tenacity
import time
import pandas as pd
from bs4 import BeautifulSoup
from selenium import webdriver
from selenium.webdriver.chrome.options import Options
from tenacity import retry, stop_after_attempt, wait_fixed
def init_driver():
    options = Options()
    options.add_argument("--headless")
    options.add_argument("--no-sandbox")
    options.add_argument("--disable-dev-shm-usage")
    driver = webdriver.Chrome(options=options)
    return driver
@retry(stop=stop_after_attempt(3), wait=wait_fixed(2))
def get_page_source(driver, url):
```

```
    driver.get(url)
    time.sleep(2)
    return driver.page_source
SECTIONS = [
    "https://www.aljazeera.com/news",
    "https://www.aljazeera.com/middle-east",
    "https://www.aljazeera.com/africa",
    "https://www.aljazeera.com/asia",
    "https://www.aljazeera.com/europe",
    "https://www.aljazeera.com/us-canada",
    "https://www.aljazeera.com/business-economy",
    "https://www.aljazeera.com/sports",
    "https://www.aljazeera.com/features",
    "https://www.aljazeera.com/opinions"
]
def extract_article_links(driver, section_url, max_pages=5):
    article_links = set()

    for page in range(1, max_pages + 1):
        page_url = f"{section_url}?page={page}"
        print(f"Visiting: {page_url}")

        try:
            html = get_page_source(driver, page_url)
        except Exception as e:
            print(f"Failed on {page_url}: {e}")
            continue

        soup = BeautifulSoup(html, 'html.parser')
        cards = soup.select('a.u-clickable-card__link')

        if not cards:
            break
        for tag in cards:
            href = tag.get("href")
            if href and href.startswith("/"):
                full_url = "https://www.aljazeera.com" + href
                article_links.add(full_url)

    return list(article_links)
def extract_article_data(driver, url):
    try:
        html = get_page_source(driver, url)
    except Exception as e:
        print(f"Failed to scrape {url}: {e}")
```

```
        return {
            "Title": "N/A",
            "Url": url,
            "Date": "N/A",
            "Description": "N/A"
        }

    soup = BeautifulSoup(html, "html.parser")

    # Extract title
    title_tag = soup.find("h1")
    title = title_tag.get_text(strip=True) if title_tag else "N/A"

    # Extract publication date
    date_tag = soup.find("time")
    date = date_tag.get("datetime") if date_tag else "N/A"

    # Extract article content
    body_tag = soup.find("div", class_="wysiwyg")
    if body_tag:
        paragraphs = body_tag.find_all("p")
        description = " ".join(p.get_text(strip=True).replace("\n", " ") for p in paragraphs)
    else:
        description = "N/A"

    return {
        "Title": title.strip(),
        "Url": url.strip(),
        "Date": date.strip(),
        "Description": description.strip()
    }
def save_to_csv(data, filename="aljazeera_articles_fixed.csv"):
    if not data:
        print("No data to save.")
        return

    df = pd.DataFrame(data)

    # Ensure correct column order
    expected_cols = ["Title", "Url", "Date", "Description"]
    for col in expected_cols:
        if col not in df.columns:
            df[col] = "N/A"

    df = df[expected_cols]  # enforce order
```

```
    df.to_csv(filename, index=False, encoding="utf-8-sig")
    print(f"✅ Saved {len(df)} articles to {filename}")
def run_scraper():
    driver = init_driver()
    all_links = set()

    try:
        for section in SECTIONS:
            links = extract_article_links(driver, section, max_pages=5)
            print(f"✅ {len(links)} articles from {section}")
            all_links.update(links)

        print(f"🔗 Total unique articles found: {len(all_links)}")

        data = []
        for i, url in enumerate(all_links, 1):
            print(f"[{i}/{len(all_links)}] Scraping {url}")
            article = extract_article_data(driver, url)
            if article:
                data.append(article)

        save_to_csv(data)

    finally:
        driver.quit()
run_scraper()
```

Output:



```
jcpoa
[120/126] Scraping https://www.aljazeera.com/news/2025/7/17/babies-born-in-uk-using-dna-from-three-people-to-avoid-genetic-di
sease
[121/126] Scraping https://www.aljazeera.com/news/2025/7/17/is-it-making-a-difference-absolutely-uk-celebrities-rally-for-gaz
a
[122/126] Scraping https://www.aljazeera.com/news/2025/7/18/pressure-builds-in-malaysia-to-reject-trumps-pro-israel-pick-as-a
mbassador
[123/126] Scraping https://www.aljazeera.com/news/2025/7/14/former-nigerian-president-buhari-to-be-buried-in-hometown-on-tues
day
[124/126] Scraping https://www.aljazeera.com/opinions/2025/7/9/why-the-future-of-ai-may-be-open-and-chinese
[125/126] Scraping https://www.aljazeera.com/opinions/2025/7/10/trump-didnt-start-the-war-on-the-poor-but-hes-taking-it-to-ne
w-extremes
[126/126] Scraping https://www.aljazeera.com/news/2025/7/18/trump-seeks-release-of-grand-jury-transcripts-as-epstein-uproar
✅ Saved 126 articles to aljazeera_articles_fixed.csv
```

CSV file is attached to the email.

2. **Deutsche Welle (DW) (Germany) Website:** https://www.dw.com/en
   Methodology: To scrape and generate current and previous published articles from Website: "https://www.dw.com/en" and extract URLs, title, description, and date.
   Tools used:
   - Jupiter notebook
   - Python language
   - BeautifulSoup (to extract data and parse the HTML).
   - Selenium (to load dynamic content).
   - installed and managed the chrome driver.

Steps learned & performed:

i. Download ChromeDriver and set the Path
ii. navigate to DW website
iii. Article elements wait
iv. Extracting article URLs
v. Scrape URLs, title, description, and date using RSS
vi. Generates output
vii. Save scraped data into csv file.

Code:

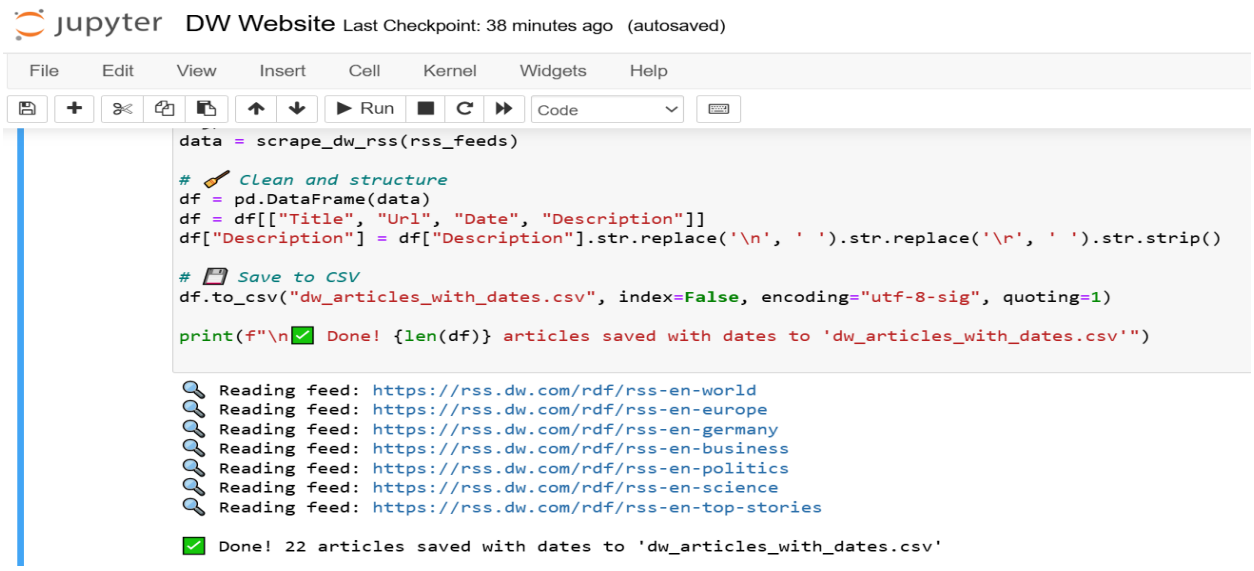
```
import feedparser
import pandas as pd
import requests
from bs4 import BeautifulSoup
from datetime import datetime

# Get full article description from URL
def get_full_description(url):
```

```
    try:
        response = requests.get(url, timeout=10)
        soup = BeautifulSoup(response.text, 'html.parser')
        paragraphs = soup.select("div.rich-text p, .bodyTxt p, .group p")
        full_text = " ".join(p.get_text(strip=True) for p in paragraphs if p.get_text(strip=True))
        return full_text
    except Exception as e:
        return f"Failed to fetch full text: {str(e)}"

#  Extract date with fallback
def extract_date(entry):
    for key in ['published', 'updated', 'dc:date']:
        if key in entry:
            try:
                return datetime(*entry.published_parsed[:6]).strftime("%Y-%m-%d")
            except Exception:
                return entry.get(key, "N/A")
    return "N/A"

def scrape_dw_rss(rss_urls):
    articles = []
    for feed_url in rss_urls:
        print(f"🔍 Reading feed: {feed_url}")
        feed = feedparser.parse(feed_url)
        for entry in feed.entries:
            title = entry.get("title", "").replace('\n', ' ').strip()
            url = entry.get("link", "").strip()
            summary = BeautifulSoup(entry.get("summary", ""), "html.parser").get_text().strip()
            date = extract_date(entry)
            full_text = get_full_description(url)

            articles.append({
                "Title": title,
                "Url": url,
                "Date": date,
                "Description": full_text if full_text else summary
            })
    return articles

# RSS Feeds
rss_feeds = [
    "https://rss.dw.com/rdf/rss-en-world",
    "https://rss.dw.com/rdf/rss-en-europe",
    "https://rss.dw.com/rdf/rss-en-germany",
    "https://rss.dw.com/rdf/rss-en-business",
```

Sara Memon

```
    "https://rss.dw.com/rdf/rss-en-politics",
    "https://rss.dw.com/rdf/rss-en-science",
    "https://rss.dw.com/rdf/rss-en-top-stories"
]
# Run
data = scrape_dw_rss(rss_feeds)

#  Clean and structure
df = pd.DataFrame(data)
df = df[["Title", "Url", "Date", "Description"]]
df["Description"] = df["Description"].str.replace('\n', ' ').str.replace('\r', ' ').str.strip()

#  Save to CSV
df.to_csv("dw_articles_with_dates.csv", index=False, encoding="utf-8-sig", quoting=1)
print(f"\n✅ Done! {len(df)} articles saved with dates to 'dw_articles_with_dates.csv'")
```

Output:



CSV file is attached to email.

Note: I have used RSS DW because simple DW main website is not designed for scraping (Even AI declared this). Tried really hard to scrape but couldn't. It showed misinterpreted output multiple times.