PS4

Faizan R.

2025-02-05

Due 02/07 at 5:00PM Central.

This submission is my work alone and complies with the 30538 integrity policy. Add your initials to indicate your agreement: **___**

Github Classroom Assignment Setup and Submission Instructions

1. **Accepting and Setting up the PS4 Assignment Repository**
    - Each student must individually accept the repository for the problem set from Github Classroom ("ps4") – https://classroom.github.com/a/hWhtcHqH
        - You will be prompted to select your cnetid from the list in order to link your Github account to your cnetid.
        - If you can't find your cnetid in the link above, click "continue to next step" and accept the assignment, then add your name, cnetid, and Github account to this Google Sheet and we will manually link it: https://rb.gy/9u7fb6
    - If you authenticated and linked your Github account to your device, you should be able to clone your PS4 assignment repository locally.
    - Contents of PS4 assignment repository:
        - `ps4_template.qmd`: this is the Quarto file with the template for the problem set. You will write your answers to the problem set here.
2. **Submission Process**:
    - Knit your completed solution `ps4.qmd` as a pdf `ps4.pdf`.
        - Your submission does not need runnable code. Instead, you will tell us either what code you ran or what output you got.
    - To submit, push `ps4.qmd` and `ps4.pdf` to your PS4 assignment repository. Confirm on Github.com that your work was successfully pushed.

Grading

- You will be graded on what was last pushed to your PS4 assignment repository before the assignment deadline
- Problem sets will be graded for completion as: {missing (0%); - (incomplete, 50%); + (excellent, 100%)}
  - The percent values assigned to each problem denote how long we estimate the problem will take as a share of total time spent on the problem set, not the points they are associated with.
- In order for your submission to be considered complete, you need to push both your `ps4.qmd` and `ps4.pdf` to your repository. Submissions that do not include both files will automatically receive 50% credit.

```
import pandas as pd
import altair as alt
import requests
from bs4 import BeautifulSoup
from datetime import datetime
import time

import warnings
warnings.filterwarnings('ignore')
alt.renderers.enable("png")
```

```
RendererRegistry.enable('png')
```

Step 1: Develop initial scraper and crawler

```

link_here = "https://oig.hhs.gov/fraud/enforcement/"
resp = requests.get(link_here)

page = resp.text
soup_obj = BeautifulSoup(page, "html.parser")

all_cards = soup_obj.find_all("li", class_="usa-card")

t_list = []
d_list = []
c_list = []
l_list = []

for x in all_cards:
    h = x.find("h2", class_="usa-card__heading")

    if h is not None:
        a = h.find("a")
        t = a.text.strip()
        l = "https://oig.hhs.gov" + a.get("href")

        d = None
        d_tag = x.find("span", class_="text-base-dark")
        if d_tag:
            d = d_tag.text.strip()
        cat = None
        cat_tag = x.find("li", class_="display-inline-block")
        if cat_tag:
            cat = cat_tag.text.strip()

        t_list.append(t)
        d_list.append(d)
        c_list.append(cat)
        l_list.append(l)

out_df = pd.DataFrame({
    "title": t_list,
    "date": d_list,
    "category": c_list,
    "link": l_list
})

print(out_df.head())
```

```
                                               title              date  \
0  Houston Transplant Doctor Indicted For Making ...  February 5, 2026
1  MultiCare Health System to Pay Millions to Set...  February 4, 2026
2  Brooklyn Banker Pleads Guilty to Laundering Pr...  February 3, 2026
3  Delafield Man Sentenced to 18 Months’ Imprison...  February 3, 2026
4  Former NFL Player Convicted for $197M Medicare...  February 3, 2026

                     category  \
0  Criminal and Civil Actions
1  Criminal and Civil Actions
2                    COVID-19
3  Criminal and Civil Actions
4  Criminal and Civil Actions

                                                link
0  https://oig.hhs.gov/fraud/enforcement/houston-...
1  https://oig.hhs.gov/fraud/enforcement/multicar...
2  https://oig.hhs.gov/fraud/enforcement/brooklyn...
3  https://oig.hhs.gov/fraud/enforcement/delafiel...
4  https://oig.hhs.gov/fraud/enforcement/former-n...
```

Step 2: Making the scraper dynamic

1. Turning the scraper into a function

- a. Pseudo-Code
- b. Create Dynamic Scraper

```python
RUN_SCRAPER = False


def get_actions_since(m, y):
    if y < 2013:
        print("Year has to be 2013 or newer (site doesn't go earlier).")
        return None

    start_dt = datetime(y, m, 1)

    titles = []
    dates = []
    cats = []
    urls = []

    p = 1
    still = True

    while still:
        page_url = f"https://oig.hhs.gov/fraud/enforcement/?page={p}"
        r = requests.get(page_url)
        s = BeautifulSoup(r.text, "html.parser")

        stuff = s.find_all("li", class_="usa-card")

        if stuff is None or len(stuff) == 0:
            break

        for item in stuff:
            h2 = item.find("h2", class_="usa-card__heading")
            if not h2:
                continue

            a = h2.find("a")
            if not a:
                continue
            t = a.get_text(strip=True)
            u = "https://oig.hhs.gov" + a.get("href")

            d_txt = None
            d_span = item.find("span", class_="text-base-dark")
            if d_span:
                d_txt = d_span.get_text(strip=True)

            cat_txt = None
            cat_li = item.find("li", class_="display-inline-block")
            if cat_li:
                cat_txt = cat_li.get_text(strip=True)

            if d_txt:
                try:
                    d_real = datetime.strptime(d_txt, "%B %d, %Y")
                    if d_real < start_dt:
                        still = False
                        break
                except:
                    pass

            titles.append(t)
            dates.append(d_txt)
            cats.append(cat_txt)
            urls.append(u)

        p += 1
        time.sleep(1)

    df = pd.DataFrame({
        "title": titles,
        "date": dates,
        "category": cats,
        "link": urls
    })

    out_name = "enforcement_actions_year_month.csv"
    df.to_csv(out_name, index=False)

    print("saved:", out_name, "rows:", len(df))
    return df
```

```
####
# #  for January 2022
if RUN_SCRAPER:
    df_2022 = scrape_enforcement_actions(1, 2022)
    print(f"\nTotal enforcement actions since Jan 2022: {len(df_2022)}")
    print(f"\nEarliest action scraped:")
    print(df_2022.tail(1))
```

Total enforcement actions: 1,787 Earliest date scraped: January 3, 2024 Earliest action(s): Former Nurse Aide Indicted In Death Of Clarksville Woman — State Enforcement Agencies

- c. Test Your Code

```
if RUN_SCRAPER:
    df_2022 = get_actions_since(1, 2022)
    print("count since Jan 2022:", len(df_2022))
    print("earliest row (oldest):")
    print(df_2022.tail(1))
```

Since January 2022 Total enforcement actions: 3,377 Earliest date scraped: January 4, 2022 Earliest action(s): "Integrated Pain Management Medical Group Agreed to Pay $10,000 for Allegedly Violating the Civil Monetary Penalties Law by Employing Excluded Individuals" - Category: Fraud Self-Disclosures

Step 3: Plot data based on scraped data

1. Plot the number of enforcement actions over time

```
df = pd.read_csv("enforcement_actions_2022_1.csv")
df["date"] = pd.to_datetime(df["date"], format="%B %d, %Y", errors="coerce")
df["year_month"] = df["date"].dt.to_period("M").dt.to_timestamp()
```

```
cnt = df.groupby("year_month").size()
cnt = cnt.reset_index()
cnt.columns = ["year_month", "count"]

chart1 = alt.Chart(cnt).mark_line(point=True).encode(
```