title

author

Invalid Date

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: XYL

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 time

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

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

Step 1: Develop initial scraper and crawler

```
pip install lxml
```

```
Requirement already satisfied: lxml in
/opt/miniconda3/envs/DAP/lib/python3.13/site-packages (6.0.2)
Note: you may need to restart the kernel to use updated packages.
```

```
import re
import requests
import pandas as pd
from bs4 import BeautifulSoup
from urllib.parse import urljoin

URL = "https://oig.hhs.gov/fraud/enforcement/"
BASE = "https://oig.hhs.gov"

headers = {"User-Agent": "Mozilla/5.0"}
r = requests.get(URL, headers=headers, timeout=30)
r.raise_for_status()

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

rows = []


for a in soup.select("li h2 a"):
    title = a.get_text(strip=True)
    link = urljoin(BASE, a.get("href", ""))
```

```
    container = a.find_parent("li")
    if container is None:
        continue

    meta = container.select_one("div.font-body-sm")
    meta_text = meta.get_text(" ", strip=True) if meta else ""

    #date
    date_text = None
    m = re.search(

        ↪ r"(January|February|March|April|May|June|July|August|September|October|November|
        meta_text
    )
    if m:
        date_text = m.group(0)

    #category
    category_text = meta_text
    if date_text:
        category_text = meta_text.replace(date_text, "").strip()
    category_text = re.sub(r"\s+", " ", category_text).strip()

    rows.append({
        "title": title,
        "date": date_text,
        "category": category_text,
        "link": link
    })

df = pd.DataFrame(rows)
df.head()
```

	title	date	category	lin
0	Houston Transplant Doctor Indicted For Making ...	February 5, 2026	Criminal and Civil Actions	htt
1	MultiCare Health System to Pay Millions to Set...	February 4, 2026	Criminal and Civil Actions	htt
2	Brooklyn Banker Pleads Guilty to Laundering Pr...	February 3, 2026	COVID-19	htt
3	Delafield Man Sentenced to 18 Months' Imprison...	February 3, 2026	Criminal and Civil Actions	htt
4	Former NFL Player Convicted for $197M Medicare...	February 3, 2026	Criminal and Civil Actions	htt

Step 2: Making the scraper dynamic

1. Turning the scraper into a function

- a. Pseudo-Code If year < 2013: print reminder and return empty dataframe Set start_date = first day of year, month Initialize empty list rows = [], set page =1 While loop: Build url = "https://oig.hhs.gov/fraud/enforcement/" Downlowd the HTML, parse with BuautifulSoup Scrape title, link, meta text category = meta text minus date Convert date to datetime Keep only rows with date >= start_date, append to rows If this page has no dates >= start_date, break page += 1, sleep(1) Make dataframe, sort by date descending, save to enforcement_actions_{year}_{month}.csv, return df
- b. Create Dynamic Scraper

```
import re
import time
import requests
import pandas as pd
from bs4 import BeautifulSoup
from urllib.parse import urljoin

DATE_PAT = re.compile(

    ↪ r"(January|February|March|April|May|June|July|August|September|October|November|Decem
)

def scrape_enforcement_actions_from(month: int, year: int, *, pause_sec:
 ↪ float = 1.0) -> pd.DataFrame:
    if year < 2013:
        print("Please restrict to year >= 2013 (only actions after 2013 are
         ↪ listed).")
        return pd.DataFrame(columns=["title", "date", "category", "link"])

    start_date = pd.Timestamp(year=year, month=month, day=1)

    base = "https://oig.hhs.gov"
    headers = {"User-Agent": "Mozilla/5.0"}

    page = 1
    kept_pages = []

    while True:
        url = f"{base}/fraud/enforcement/?page={page}"
```

```
        r = requests.get(url, headers=headers, timeout=30)
        r.raise_for_status()
        soup = BeautifulSoup(r.text, "html.parser")

        links = soup.select("li h2 a")
        if not links:
            break

        page_rows = []
        for a in links:
            title = a.get_text(strip=True)
            link = urljoin(base, a.get("href", ""))

            container = a.find_parent("li")
            if container is None:
                continue

            meta = container.select_one("div.font-body-sm")
            meta_text = meta.get_text(" ", strip=True) if meta else ""

            m = DATE_PAT.search(meta_text)
            date_text = m.group(0) if m else None
            date_dt = pd.to_datetime(date_text, errors="coerce")

            # category = meta_text - date_text
            category_text = meta_text
            if date_text:
                category_text = meta_text.replace(date_text, "").strip()
            category_text = re.sub(r"\s+", " ", category_text).strip()

            page_rows.append({
                "title": title,
                "date": date_text,
                "date_dt": date_dt,
                "category": category_text,
                "link": link
            })

        page_df = pd.DataFrame(page_rows)

        if page_df.empty or page_df["date_dt"].isna().all():
            break
```

```
        keep = page_df[page_df["date_dt"] >= start_date].copy()
        kept_pages.append(keep)

        if keep.empty:
            break

        page += 1
        time.sleep(pause_sec)

    if not kept_pages:
        return pd.DataFrame(columns=["title", "date", "category", "link"])

    out = pd.concat(kept_pages, ignore_index=True)
    out = out.sort_values("date_dt",
↪   ascending=False).drop(columns=["date_dt"])
    return out
```

```
run_scraper_2024 = False

year_2024, month_2024 = 2024, 1
csv_2024 = f"enforcement_actions_{year_2024}_{month_2024:02d}.csv"

if run_scraper_2024:
    df_2024 = scrape_enforcement_actions_from(month=month_2024,
↪   year=year_2024, pause_sec=1.0)
    df_2024.to_csv(csv_2024, index=False)
else:
    df_2024 = pd.read_csv(csv_2024)

# Compute total count + earliest action
df_2024_tmp = df_2024.copy()
df_2024_tmp["date_dt"] = pd.to_datetime(df_2024_tmp["date"], errors="coerce")

n_2024 = len(df_2024_tmp)
earliest_2024 = df_2024_tmp.sort_values("date_dt", ascending=True).iloc[0]

n_2024, earliest_2024[["date", "title", "category", "link"]].to_dict()
```

```
(1787,
 {'date': 'January 3, 2024',
  'title': 'Former Nurse Aide Indicted In Death Of Clarksville Patient
  Arrested In Georgia',
```

```
  'category': 'State Enforcement Agencies',
  'link':
  'https://oig.hhs.gov/fraud/enforcement/former-nurse-aide-indicted-in-death-of-clarksville-p
```

I collected enforcement actions starting from January 2024. In total, the scraper returned 1,787 enforcement actions in the final dataframe. The earliest enforcement action in this dataset is dated January 3, 2024, titled "Former Nurse Aide Indicted In Death Of Clarksville Patient Arrested In Georgia." This action falls under the category State Enforcement Agencies, and the associated link is: https://oig.hhs.gov/fraud/enforcement/former-nurse-aide-indicted-in-death-of-clarksville-patient-arrested-in-georgia/

- c. Test Your Code

```
run_scraper_2022 = True

year_2022, month_2022 = 2022, 1
csv_2022 = f"enforcement_actions_{year_2022}_{month_2022:02d}.csv"

if run_scraper_2022:
    df_2022 = scrape_enforcement_actions_from(month=month_2022,
 ↪  year=year_2022, pause_sec=1.0)
    df_2022.to_csv(csv_2022, index=False)
else:
    df_2022 = pd.read_csv(csv_2022)

# Compute total count + earliest action
df_2022_tmp = df_2022.copy()
df_2022_tmp["date_dt"] = pd.to_datetime(df_2022_tmp["date"], errors="coerce")

n_2022 = len(df_2022_tmp)
earliest_2022 = df_2022_tmp.sort_values("date_dt", ascending=True).iloc[0]

n_2022, earliest_2022[["date", "title", "category", "link"]].to_dict()
```

```
(3377,
 {'date': 'January 4, 2022',
  'title': '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',
  'link':
  'https://oig.hhs.gov/fraud/enforcement/integrated-pain-management-medical-group-agreed-to-p
```

I collected enforcement actions starting from January 2022. In total, the scraper returned 3,377 enforcement actions in the final dataframe. The earliest enforcement action in this dataset is dated January 4, 2022, titled "Integrated Pain Management Medical Group Agreed to Pay $10,000 for Allegedly Violating the Civil Monetary Penalties Law by Employing Excluded Individuals." This action is categorized as Fraud Self-Disclosures, and the associated link is: https://oig.hhs.gov/fraud/enforcement/integrated-pain-management-medical-group-agreed-to-pay-10000-for-allegedly-violating-the-civil-monetary-penalties-law-by-employing-excluded-individuals/

Step 3: Plot data based on scraped data

1. Plot the number of enforcement actions over time

```
import pandas as pd
import altair as alt
alt.renderers.enable("default")

df = pd.read_csv("enforcement_actions_2022_01.csv")

df["date_dt"] = pd.to_datetime(df["date"], errors="coerce")

df["month"] = df["date_dt"].dt.to_period("M").dt.to_timestamp()

df.head()
```

	title	date	category	li
0	Houston Transplant Doctor Indicted For Making ...	February 5, 2026	Criminal and Civil Actions	ht
1	MultiCare Health System to Pay Millions to Set...	February 4, 2026	Criminal and Civil Actions	ht
2	Brooklyn Banker Pleads Guilty to Laundering Pr...	February 3, 2026	COVID-19	ht
3	Former NFL Player Convicted for $197M Medicare...	February 3, 2026	Criminal and Civil Actions	ht
4	Attorney General Hanaway Obtains Medicaid Frau...	February 3, 2026	State Enforcement Agencies	ht

```
monthly_counts = (
    df.groupby("month", as_index=False)
      .size()
      .rename(columns={"size": "n_actions"})
)

alt.Chart(monthly_counts).mark_line(point=True).encode(
```

```
    x=alt.X("month:T", title="Month"),
    y=alt.Y("n_actions:Q", title="Number of enforcement actions")
).properties(
    width=650,
    height=350
)
```

```
alt.Chart(...)
```

2. Plot the number of enforcement actions categorized:

- based on "Criminal and Civil Actions" vs. "State Enforcement Agencies"

```
df_two = df.copy()

df_two["category_group"] = pd.NA
df_two.loc[
    df_two["category"].str.contains("Criminal and Civil Actions", na=False),
    "category_group"
] = "Criminal and Civil Actions"

df_two.loc[
    df_two["category"].str.contains("State Enforcement Agencies", na=False),
    "category_group"
] = "State Enforcement Agencies"

df_two = df_two.dropna(subset=["category_group"])

monthly_two = (
    df_two.groupby(["month", "category_group"], as_index=False)
          .size()
          .rename(columns={"size": "n_actions"})
)

alt.Chart(monthly_two).mark_line(point=True).encode(
    x=alt.X("month:T", title="Month"),
    y=alt.Y("n_actions:Q", title="Number of enforcement actions"),
    color=alt.Color("category_group:N", title="Category")
).properties(
    width=650,
    height=350
)
```

`alt.Chart(...)`

- based on five topics

```
df_cc = df[df["category"].str.contains("Criminal and Civil Actions",
 ↪  na=False)].copy()

title_lower = df_cc["title"].str.lower()

df_cc["topic"] = "Other"

# Health Care Fraud
df_cc.loc[
    title_lower.str.contains(
        r"medicare|medicaid|health
         ↪  care|healthcare|clinic|physician|hospital|home health",
        regex=True,
        na=False
    ),
    "topic"
] = "Health Care Fraud"

# Financial Fraud
df_cc.loc[
    title_lower.str.contains(
        r"bank|financial|money laundering|wire fraud|securities|credit|loan",
        regex=True,
        na=False
    ),
    "topic"
] = "Financial Fraud"

# Drug Enforcement
df_cc.loc[
    title_lower.str.contains(
        r"drug|opioid|fentanyl|controlled substance|prescription",
        regex=True,
        na=False
    ),
    "topic"
] = "Drug Enforcement"

# Bribery / Corruption
```

```
df_cc.loc[
    title_lower.str.contains(
        r"bribery|corruption|kickback",
        regex=True,
        na=False
    ),
    "topic"
] = "Bribery/Corruption"


monthly_topic = (
    df_cc.groupby(["month", "topic"], as_index=False)
         .size()
         .rename(columns={"size": "n_actions"})
)

topic_order = [
    "Health Care Fraud",
    "Financial Fraud",
    "Drug Enforcement",
    "Bribery/Corruption",
    "Other"
]

alt.Chart(monthly_topic).mark_line(point=True).encode(
    x=alt.X("month:T", title="Month"),
    y=alt.Y("n_actions:Q", title="Number of enforcement actions"),
    color=alt.Color("topic:N", sort=topic_order, title="Topic")
).properties(
    width=650,
    height=350
)
```

```
alt.Chart(...)
```