PS5 Submission

Pei-Chin Lu and Sohyun Lim

Invalid Date

Due 11/9 at 5:00PM Central. Worth 100 points + 10 points extra credit.

Submission Steps (10 pts)

1. This problem set is a paired problem set.
2. Play paper, scissors, rock to determine who goes first. Call that person *Partner 1*.
    - Partner 1 (name and cnet ID): Pei-Chin Lu and peichin
    - Partner 2 (name and cnet ID): Sohyun Lim and shlim
3. Partner 1 will accept the `ps5` and then share the link it creates with their partner. You can only share it with one partner so you will not be able to change it after your partner has accepted.
4. "This submission is our work alone and complies with the 30538 integrity policy." Add your initials to indicate your agreement: **PL SL**
5. "I have uploaded the names of anyone else other than my partner and I worked with on the problem set **here**" (1 point)
6. Late coins used this pset: **0** Late coins left after submission: **4**
7. Knit your `ps5.qmd` to a PDF file to make `ps5.pdf`,
    - The PDF should not be more than 25 pages. Use `head()` and re-size figures when appropriate.
8. (Partner 1): push `ps5.qmd` and `ps5.pdf` to your GitHub repo.
9. (Partner 1): submit `ps5.pdf` via Gradescope. Add your partner on Gradescope.
10. (Partner 1): tag your submission in Gradescope.

```
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

1. Scraping (PARTNER 1)

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

# URL of the page to scrape
url = "https://oig.hhs.gov/fraud/enforcement/"

# Send a request to fetch the HTML content
response = requests.get(url)
soup = BeautifulSoup(response.text, "html.parser")

# Initialize lists to store the scraped data
titles = []
dates = []
categories = []
links = []

# Find all enforcement action cards on the page
actions = soup.select("li.usa-card")

# Extract information from each enforcement action
for action in actions:
    # Extract the title
    title_tag = action.find("h2", class_="usa-card__heading")
    title = title_tag.a.text.strip() if title_tag and title_tag.a else "N/A"
```

```
    # Extract the link
    link = "https://oig.hhs.gov" + title_tag.a["href"] if title_tag and
↪   title_tag.a and 'href' in title_tag.a.attrs else "N/A"

    # Extract the date
    date_tag = action.find("span", class_="text-base-dark")
    date = date_tag.text.strip() if date_tag else "N/A"

    # Extract the category
    category_tag = action.find("ul", class_="display-inline")
    category = category_tag.find("li").text.strip() if category_tag and
↪   category_tag.find("li") else "N/A"

    # Append data to lists
    titles.append(title)
    dates.append(date)
    categories.append(category)
    links.append(link)

# Create a DataFrame
df = pd.DataFrame({
    'Title': titles,
    'Date': dates,
    'Category': categories,
    'Link': links
})

# Display the first few rows
print(df.head())
```

```
                                               Title              Date  \
0  Pharmacist and Brother Convicted of $15M Medic...  November 8, 2024
1  Boise Nurse Practitioner Sentenced To 48 Month...  November 7, 2024
2  Former Traveling Nurse Pleads Guilty To Tamper...  November 7, 2024
3  Former Arlington Resident Sentenced To Prison ...  November 7, 2024
4  Paroled Felon Sentenced To Six Years For Fraud...  November 7, 2024

                    Category  \
0  Criminal and Civil Actions
1  Criminal and Civil Actions
2  Criminal and Civil Actions
3  Criminal and Civil Actions
```

```
4  Criminal and Civil Actions

                                                  Link
0  https://oig.hhs.gov/fraud/enforcement/pharmaci...
1  https://oig.hhs.gov/fraud/enforcement/boise-nu...
2  https://oig.hhs.gov/fraud/enforcement/former-t...
3  https://oig.hhs.gov/fraud/enforcement/former-a...
4  https://oig.hhs.gov/fraud/enforcement/paroled-...
```

2. Crawling (PARTNER 1)

```python
import requests
from bs4 import BeautifulSoup
import pandas as pd
import time

# Base URL for the main page
base_url = "https://oig.hhs.gov/fraud/enforcement"
response = requests.get(base_url)
soup = BeautifulSoup(response.text, "html.parser")

# Initialize data storage
data = []

# Step 1: Extract each enforcement action's information from the main page
for action in soup.select("li.usa-card"):  # Adjust this selector based on
 ↪  main page structure
    # Extract title
    title_tag = action.find("h2", class_="usa-card__heading")
    title = title_tag.get_text(strip=True) if title_tag else "N/A"

    # Extract date
    date_tag = action.find("span", class_="text-base-dark")
    date = date_tag.get_text(strip=True) if date_tag else "N/A"

    # Extract category
    category_tag = action.find("ul", class_="display-inline")
    category = category_tag.get_text(strip=True) if category_tag else "N/A"

    # Extract the link to the detail page
    link_tag = title_tag.find("a") if title_tag else None
```

```
    link = "https://oig.hhs.gov" + link_tag["href"] if link_tag else "N/A"

    # Step 2: Visit the detail page and extract the agency name
    agency = "N/A"
    if link != "N/A":
        detail_response = requests.get(link)
        detail_soup = BeautifulSoup(detail_response.text, "html.parser")

        # Find the element containing the agency name
        agency_span = detail_soup.find("span", text="Agency:")
        if agency_span:
            agency_li = agency_span.find_parent("li")
            if agency_li:
                agency = agency_li.get_text(strip=True).replace("Agency:",
↪   "").strip()
            else:
                print(f"Agency <li> tag not found on {link}")
        else:
            print(f"'Agency:' label not found on {link}")

    # Store the data in the list
    data.append({
        "Title": title,
        "Date": date,
        "Category": category,
        "Agency": agency,
        "Link": link
    })

# Create a DataFrame and display the results
df = pd.DataFrame(data)
print(df.head())
```

```
'Agency:' label not found on
https://oig.hhs.gov/fraud/enforcement/michael-depalma-md-and-virginia-i-spine-physicians-agre
'Agency:' label not found on
https://oig.hhs.gov/fraud/enforcement/mercy-health-youngstown-agreed-to-pay-69000-for-allege
                                               Title             Date  \
0  Pharmacist and Brother Convicted of $15M Medic...  November 8, 2024
1  Boise Nurse Practitioner Sentenced To 48 Month...  November 7, 2024
2  Former Traveling Nurse Pleads Guilty To Tamper...  November 7, 2024
3  Former Arlington Resident Sentenced To Prison ...  November 7, 2024
```

```
4  Paroled Felon Sentenced To Six Years For Fraud...  November 7, 2024

                     Category  \
0  Criminal and Civil Actions
1  Criminal and Civil Actions
2  Criminal and Civil Actions
3  Criminal and Civil Actions
4  Criminal and Civil Actions

                                              Agency  \
0                          U.S. Department of Justice
1  November 7, 2024; U.S. Attorney's Office, Dist...
2  U.S. Attorney's Office, District of Massachusetts
3  U.S. Attorney's Office, Eastern District of Vi...
4  U.S. Attorney's Office, Middle District of Flo...

                                                Link
0  https://oig.hhs.gov/fraud/enforcement/pharmaci...
1  https://oig.hhs.gov/fraud/enforcement/boise-nu...
2  https://oig.hhs.gov/fraud/enforcement/former-t...
3  https://oig.hhs.gov/fraud/enforcement/former-a...
4  https://oig.hhs.gov/fraud/enforcement/paroled-...
```

Step 2: Making the scraper dynamic

1. Turning the scraper into a function

- a. Pseudo-Code (PARTNER 2)

I. "dynamic_scraper" function

1. Function Start:
    - Define a function dynamic_scraper that takes month and year as input parameters.
2. Check Year Condition:
    - If year is less than 2013, print a message indicating that data is only available from 2013 onwards and exit the function.
3. Initialize Variables:
    - Define the base URL for the HHS OIG enforcement actions page.
    - Create a start_date variable with the given year and month as the starting point for scraping.

- Initialize an empty list data to store the enforcement action details.
- Set page to 1 to start scraping from the first page.

4. Loop Through Each Page:
    - Repeat the following steps for each page until there are no more enforcement actions to process.
    - Here, a while loop is used to scrape data from each page until there are no more pages left, in order to continue scraping as long as there is data on each page and stops automatically when no data is found.
        - We use while True to create an infinite loop initially and break out of the loop when we reach the last page or find no further data.
        - Page Number Increment: We start from the first page and increase the page number by 1 with each iteration to move to the next page.
        - Break Condition: The loop stops if no more enforcement actions are found on a page or if only data older than the specified start date is available.
5. Request Page:
    - Construct the page URL using the base URL and the current page number.
    - Send a GET request to fetch the page content and parse it with BeautifulSoup.
6. Find All Enforcement Actions:
    - Search for all enforcement action elements (li.usa-card) on the page.
    - If no actions are found, print a message indicating the end of available data or a possible change in page structure, then exit the loop.
7. Process Each Enforcement Action:
    - For each action found on the page, extract the following details:
        - Title: Locate the title tag and extract the title text.
        - Link: Get the link to the detailed page of the action.
        - Date: Locate the date tag, extract the date text, and parse it as a datetime object. If the date is older than start_date, print a message and stop further scraping by saving and returning the data collected so far.
        - Category: Extract the category information from the specified list tag.
8. Visit Detail Page for Agency Name:
    - If the action has a valid link, follow the link to the detailed page.
    - Parse the detail page with BeautifulSoup.
    - Find the "Agency:" span tag and retrieve the agency name from the corresponding li element.
9. Store Action Details:
    - Append a dictionary containing the action's title, date, category, agency, and link to the data list.

10. Progress Update:
    - Every 20 pages, print the number of actions collected so far and display the last few entries in data as a preview.
11. Next Page:
    - Increment the page number by 1 and add a 1-second delay before making the next page request.
12. Save Data to CSV:
    - Once all pages are processed or scraping is stopped due to the date condition, call save_data_to_csv with the data collected, month, and year.

II. "save_data_to_csv" function

1. Convert to DataFrame:
    - Convert the collected data list into a DataFrame.
2. Save as CSV:
    - Define the csv_filename based on the given year and month.
    - Try saving the DataFrame to a CSV file with utf-8-sig encoding.
    - If successful, print a confirmation message; if there is an error, print the error message.
3. Return DataFrame:
    - Return the DataFrame for further use or inspection.

- b. Create Dynamic Scraper (PARTNER 2)

1) The number of enforcement actions we get in the final dataframe : 1510
2) The date and details of the earliest enforcement action it scraped

- Date : January 3, 2023
- Title : Podiatrist Pays $90,000 To Settle False Billing Allegations
- Category : Criminal and Civil Actions
- Agency : U.S. Attorney's Office, Southern District of Texas
- Link : https://oig.hhs.gov/fraud/enforcement/podiatrist-pays-90000-to-settle-false-billing-allegations/

```
import requests
from bs4 import BeautifulSoup
import pandas as pd
import time
```

```
from datetime import datetime

def dynamic_scraper(month, year):
    print("1")
    # Step 1: Check the Year
    if year < 2013:
        print("Data is only available for 2013 and later. Please set the year
        ↪  to 2013 or higher.")
        return None

    # Step 2: Initialize Variables
    base_url = "https://oig.hhs.gov/fraud/enforcement/"
    start_date = datetime(year, month, 1)
    data = []
    page = 1  # Start on the first page
    print("2")

    # Step 3: Loop Through Each Page
    while True:
        print("3")
        # Generate the URL for the current page
        url = f"{base_url}?page={page}"
        response = requests.get(url)
        soup = BeautifulSoup(response.text, "html.parser")

        # Step 4: Find All Enforcement Actions on the Page
        actions = soup.select("li.usa-card")
        print("4")

        # If no actions are found, exit the loop
        if not actions:
            print("No more enforcement actions found or page structure has
            ↪  changed.")
            break

        # Step 5: Extract Details for Each Action on the Page
        for action in actions:
            print(f"{page}page")
            # Extract title
            title_tag = action.find("h2", class_="usa-card__heading")
            title = title_tag.a.text.strip() if title_tag and title_tag.a
↪   else "N/A"
```

```
            # Extract link
            link = "https://oig.hhs.gov" + title_tag.a["href"] if title_tag
↪  and title_tag.a and 'href' in title_tag.a.attrs else "N/A"

            # Extract date and check if it is within the required range
            date_tag = action.find("span", class_="text-base-dark")
            date_str = date_tag.text.strip() if date_tag else "N/A"
            date_obj = datetime.strptime(date_str, "%B %d, %Y") if date_str
↪  != "N/A" else None

            # Stop scraping if we've reached a date earlier than the start
             ↪  date
            if date_obj and date_obj < start_date:
                print("Reached actions older than the start date.")
                return save_data_to_csv(data, month, year)  # Save data to
                 ↪  CSV and return DataFrame

            # Extract category
            category_tag = action.find("ul", class_="display-inline")
            category = category_tag.find("li").text.strip() if category_tag
↪  and category_tag.find("li") else "N/A"

            # Extract agency name from the detail page
            agency = "N/A"
            if link != "N/A":
                detail_response = requests.get(link)
                detail_soup = BeautifulSoup(detail_response.text,
↪  "html.parser")

                # Find the agency name (adjust selector if needed)
                agency_span = detail_soup.find("span", text="Agency:")
                if agency_span:
                    agency_li = agency_span.find_parent("li")
                    if agency_li:
                        agency =
↪  agency_li.get_text(strip=True).replace("Agency:", "").strip()

            # Append the action's data to our list
            data.append({
                "Title": title,
                "Date": date_str,
```

```
                "Category": category,
                "Agency": agency,
                "Link": link
            })

        # Print progress every 20 pages
        if page % 20 == 0:
            print(f"Page {page}: Collected {len(data)} enforcement actions so
             ↪ far.")
            print(pd.DataFrame(data).tail())  # Show the last few entries to
             ↪ preview the data

        # Move to the next page
        page += 1
        time.sleep(1)  # Delay before fetching the next page

    # Final save if end of pages is reached without date check stopping the
     ↪ process
    return save_data_to_csv(data, month, year)

def save_data_to_csv(data, month, year):
    # Step 6: Convert Data to DataFrame and Save as CSV
    df = pd.DataFrame(data)
    csv_filename = f"enforcement_actions_{year}_{month}.csv"
    try:
        df.to_csv(csv_filename, index=False, encoding='utf-8-sig')
        print(f"Data successfully saved to {csv_filename}")
    except Exception as e:
        print(f"Error saving CSV file: {e}")
    return df

# Example Usage
# Collect data from January 2023
if __name__ == "__main__":
    df_2023 = dynamic_scraper(1, 2023)
    if df_2023 is not None and not df_2023.empty:
        print("Number of enforcement actions since January 2023:",
         ↪ len(df_2023))
        # print("Earliest enforcement action:", df_2023.iloc[-1])
    else:
        print("No enforcement actions found for the specified period.")
```

- c. Test Partner's Code (PARTNER 1)

1) The number of enforcement actions we get in the final dataframe : 2998
2) The date and details of the earliest enforcement action it scraped

- Date : January 4, 2021
- Title : The United States And Tennessee Resolve Claims With Three Providers For False Claims Act Liability Relating To 'P-Stim' Devices For A Total Of $1.72 Million
- Category : Criminal and Civil Actions
- Agency : U.S. Attorney's Office, Middle District of Tennessee
- Link : https://oig.hhs.gov/fraud/enforcement/the-united-states-and-tennessee-resolve-claims-with-three-providers-for-false-claims-act-liability-relating-to-p-stim-devices-for-a-total-of-172-million/

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

def dynamic_scraper(month, year):
    print("1")
    # Step 1: Check the Year
    if year < 2013:
        print("Data is only available for 2013 and later. Please set the year
        ↪  to 2013 or higher.")
        return None

    # Step 2: Initialize Variables
    base_url = "https://oig.hhs.gov/fraud/enforcement/"
    start_date = datetime(year, month, 1)
    data = []
    page = 1  # Start on the first page
    print("2")

    # Step 3: Loop Through Each Page
    while True:
        print("3")
        # Generate the URL for the current page
        url = f"{base_url}?page={page}"
        response = requests.get(url)
        soup = BeautifulSoup(response.text, "html.parser")
```

```
        # Step 4: Find All Enforcement Actions on the Page
        actions = soup.select("li.usa-card")
        print("4")

        # If no actions are found, exit the loop
        if not actions:
            print("No more enforcement actions found or page structure has
             ↪  changed.")
            break

        # Step 5: Extract Details for Each Action on the Page
        for action in actions:
            print(f"{page}page")
            # Extract title
            title_tag = action.find("h2", class_="usa-card__heading")
            title = title_tag.a.text.strip() if title_tag and title_tag.a
↪   else "N/A"

            # Extract link
            link = "https://oig.hhs.gov" + title_tag.a["href"] if title_tag
↪   and title_tag.a and 'href' in title_tag.a.attrs else "N/A"

            # Extract date and check if it is within the required range
            date_tag = action.find("span", class_="text-base-dark")
            date_str = date_tag.text.strip() if date_tag else "N/A"
            date_obj = datetime.strptime(date_str, "%B %d, %Y") if date_str
↪   != "N/A" else None

            # Stop scraping if we've reached a date earlier than the start
             ↪  date
            if date_obj and date_obj < start_date:
                print("Reached actions older than the start date.")
                return save_data_to_csv(data, month, year)  # Save data to
                 ↪  CSV and return DataFrame

            # Extract category
            category_tag = action.find("ul", class_="display-inline")
            category = category_tag.find("li").text.strip() if category_tag
↪   and category_tag.find("li") else "N/A"

            # Extract agency name from the detail page
            agency = "N/A"
```

```
            if link != "N/A":
                detail_response = requests.get(link)
                detail_soup = BeautifulSoup(detail_response.text,
↪  "html.parser")

                # Find the agency name (adjust selector if needed)
                agency_span = detail_soup.find("span", text="Agency:")
                if agency_span:
                    agency_li = agency_span.find_parent("li")
                    if agency_li:
                        agency =
↪  agency_li.get_text(strip=True).replace("Agency:", "").strip()

            # Append the action's data to our list
            data.append({
                "Title": title,
                "Date": date_str,
                "Category": category,
                "Agency": agency,
                "Link": link
            })

        # Print progress every 20 pages
        if page % 20 == 0:
            print(f"Page {page}: Collected {len(data)} enforcement actions so
             ↪  far.")
            print(pd.DataFrame(data).tail())  # Show the last few entries to
             ↪  preview the data

        # Move to the next page
        page += 1
        time.sleep(1)  # Delay before fetching the next page

    # Final save if end of pages is reached without date check stopping the
     ↪  process
    return save_data_to_csv(data, month, year)

def save_data_to_csv(data, month, year):
    # Step 6: Convert Data to DataFrame and Save as CSV
    df = pd.DataFrame(data)
    csv_filename = f"enforcement_actions_{year}_{month}.csv"
    try:
```

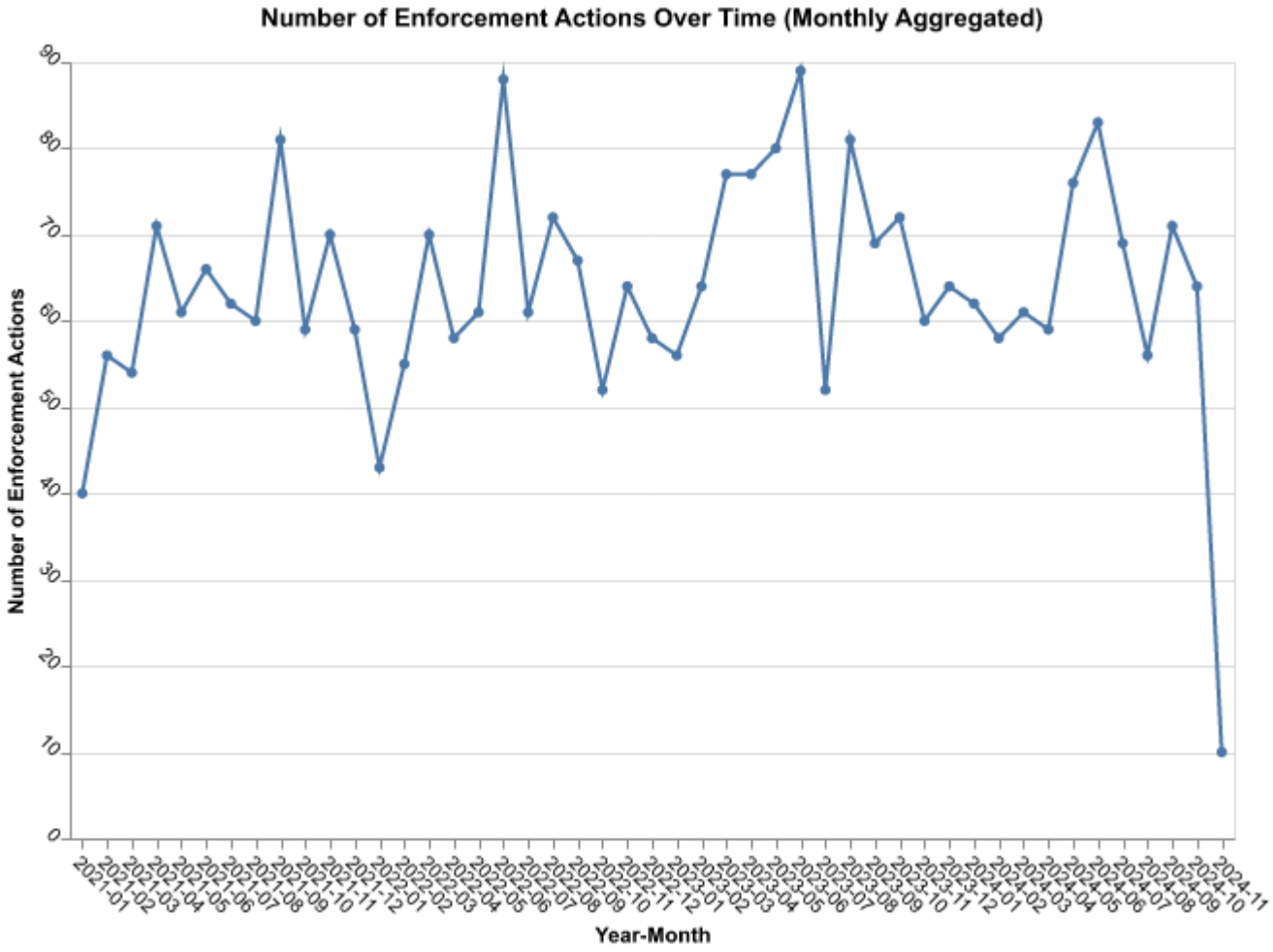
```
        df.to_csv(csv_filename, index=False, encoding='utf-8-sig')
        print(f"Data successfully saved to {csv_filename}")
    except Exception as e:
        print(f"Error saving CSV file: {e}")
    return df

# Example Usage
# Collect data from January 2021
if __name__ == "__main__":
    df_2021 = dynamic_scraper(1, 2021)
    if df_2021 is not None and not df_2021.empty:
        print("Number of enforcement actions since January 2021:",
         ↪  len(df_2021))
        print("Earliest enforcement action:", df_2021.iloc[-1])
    else:
        print("No enforcement actions found for the specified period.")
```

Step 3: Plot data based on scraped data

1. Plot the number of enforcement actions over time (PARTNER 2)

```
import pandas as pd
import altair as alt

# Load the data from the CSV file
file_path =
 ↪  r"/Users/sohyunlim/Desktop/python-ps5-shilm/enforcement_actions_2021_1.csv"
df_q3 = pd.read_csv(file_path)

# Ensure the 'Date' column is in datetime format
df_q3['Date'] = pd.to_datetime(df_q3['Date'], errors='coerce')

# Create 'Year_Month' column formatted as YYYY-MM for monthly aggregation
df_q3['Year_Month'] = df_q3['Date'].dt.to_period('M').astype(str)

# Plot the line chart using Altair with transform_aggregate and groupby
chart_1 = alt.Chart(df_q3).transform_aggregate(
    count='count()',
    groupby=['Year_Month']
).mark_line(point=True).encode(
```

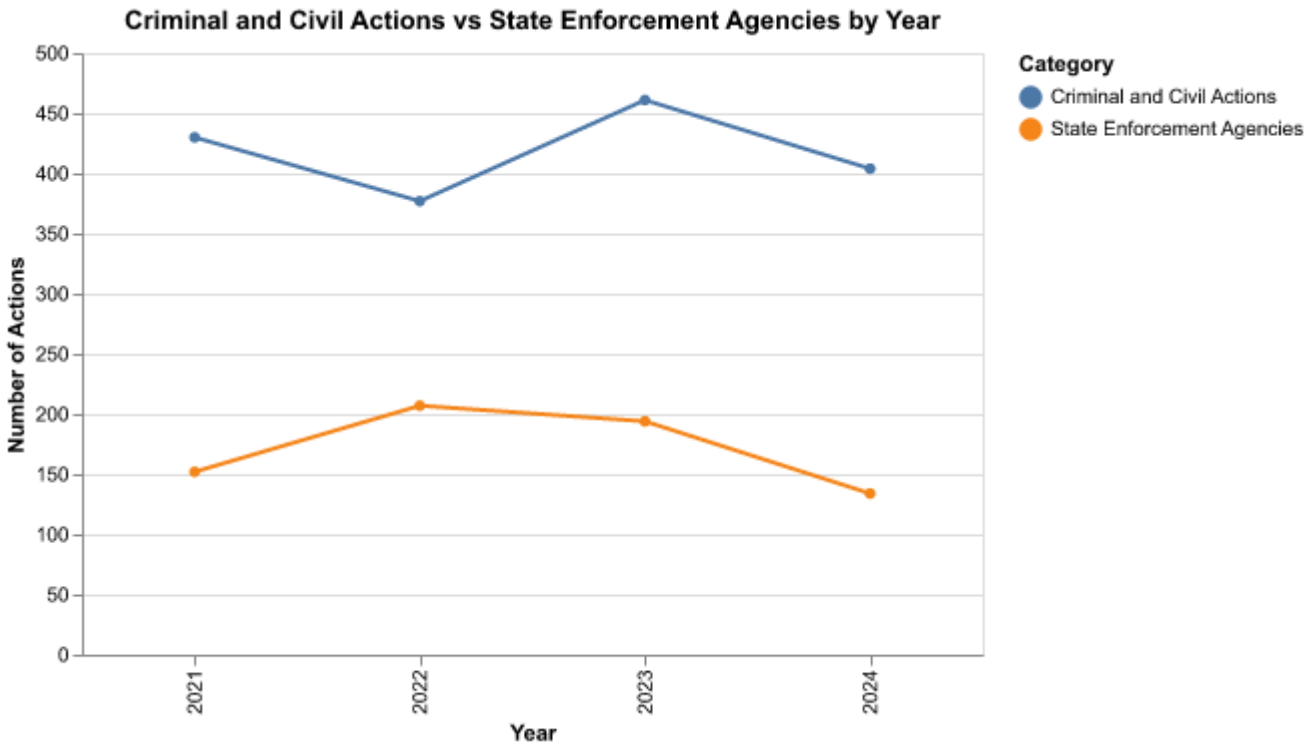
```
    alt.X('Year_Month:O', title='Year-Month'),
    alt.Y('count:Q', title='Number of Enforcement Actions')
).properties(
    title='Number of Enforcement Actions Over Time (Monthly Aggregated)',
    width=600,
    height=400
).configure_axis(
    labelAngle=45
)

chart_1
```



2. Plot the number of enforcement actions categorized: (PARTNER 1)

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

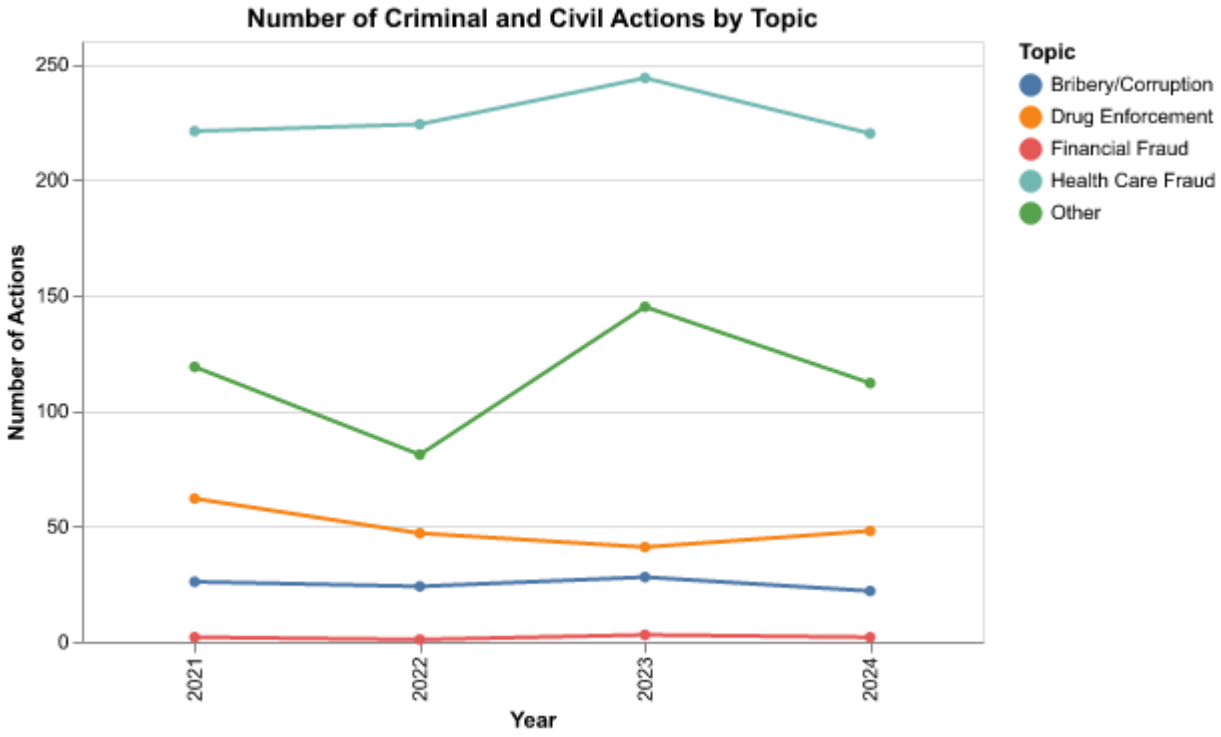
```
import pandas as pd
import altair as alt

# Load the data from the CSV file
file_path =
 ↪  r"/Users/sohyunlim/Desktop/python-ps5-shilm/enforcement_actions_2021_1.csv"
df = pd.read_csv(file_path)

# Convert the 'Date' column to datetime format and extract the year
df['Date'] = pd.to_datetime(df['Date'], errors='coerce')
df['Year'] = df['Date'].dt.year

# Calculate the count of "Criminal and Civil Actions" vs "State Enforcement
 ↪  Agencies" by year
category_counts = df.groupby(['Year',
 ↪  'Category']).size().reset_index(name='Count')

# Filter for specific categories: "Criminal and Civil Actions" and "State
 ↪  Enforcement Agencies"
filtered_categories =
 ↪  category_counts[category_counts['Category'].isin(['Criminal and Civil
 ↪  Actions', 'State Enforcement Agencies'])]

# Chart 1: Line chart comparing "Criminal and Civil Actions" vs "State
 ↪  Enforcement Agencies"
chart_2 = alt.Chart(filtered_categories).mark_line(point=True).encode(
    x=alt.X('Year:O', title='Year'),
    y=alt.Y('Count:Q', title='Number of Actions'),
    color=alt.Color('Category:N', title='Category'),
    tooltip=['Year', 'Category', 'Count']
).properties(
    title='Criminal and Civil Actions vs State Enforcement Agencies by Year',
    width=450,
    height=300
)

chart_2
```



- based on five topics

```
import pandas as pd
import altair as alt

# Load the data from the CSV file
file_path =
 ↪  r"/Users/sohyunlim/Desktop/python-ps5-shilm/enforcement_actions_2021_1.csv"
df = pd.read_csv(file_path)

# Convert the 'Date' column to datetime format and extract the year
df['Date'] = pd.to_datetime(df['Date'], errors='coerce')
df['Year'] = df['Date'].dt.year

# Define a function to assign topics based on multiple keywords within the
 ↪  "Criminal and Civil Actions" category
def assign_topic(title):
    title = title.lower()  # Convert title to lowercase for keyword matching
    if any(keyword in title for keyword in ['health', 'medicare', 'medicaid',
     ↪  'hospital', 'clinic']):
        return 'Health Care Fraud'
```

```
    elif any(keyword in title for keyword in ['bank', 'financial', 'loan',
     ↪  'securities', 'investment']):
        return 'Financial Fraud'
    elif any(keyword in title for keyword in ['drug', 'opioid', 'narcotics',
     ↪  'substance', 'pharmacy']):
        return 'Drug Enforcement'
    elif any(keyword in title for keyword in ['bribery', 'corruption',
     ↪  'kickback', 'bribe']):
        return 'Bribery/Corruption'
    else:
        return 'Other'

# Apply the topic assignment only for "Criminal and Civil Actions" category
df['Topic'] = df.apply(lambda x: assign_topic(x['Title']) if x['Category'] ==
 ↪  'Criminal and Civil Actions' else None, axis=1)

# Filter the data to include only the "Criminal and Civil Actions" category
criminal_actions_df = df[df['Category'] == 'Criminal and Civil Actions']

# Calculate the count of each topic within "Criminal and Civil Actions" by
 ↪  year
topic_counts = criminal_actions_df.groupby(['Year',
 ↪  'Topic']).size().reset_index(name='Count')

# Create an Altair line chart for topics within "Criminal and Civil Actions"
chart_3 = alt.Chart(topic_counts).mark_line(point=True).encode(
    x=alt.X('Year:O', title='Year'),
    y=alt.Y('Count:Q', title='Number of Actions'),
    color=alt.Color('Topic:N', title='Topic'),
    tooltip=['Year', 'Topic', 'Count']
).properties(
    title='Number of Criminal and Civil Actions by Topic',
    width=450,
    height=300
)

chart_3
```

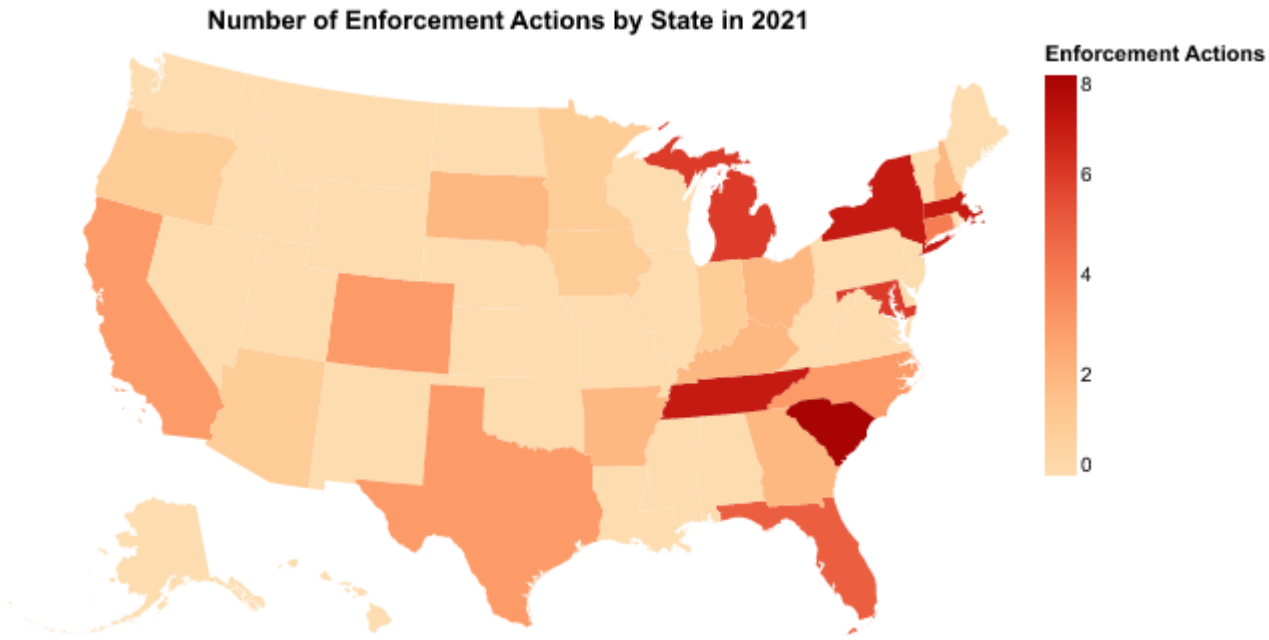


Step 4: Create maps of enforcement activity

1. Map by State (PARTNER 1)

```
import pandas as pd
import geopandas as gpd
import altair as alt
import json

# Step 1: Load the enforcement data
file_path =
 ↪  r"/Users/sohyunlim/Desktop/python-ps5-shilm/enforcement_actions_2021_1.csv"
df = pd.read_csv(file_path)

# Step 2: Filter for state-level enforcement actions
state_actions = df[df['Agency'].str.contains("State of", na=False)].copy()

# Step 3: Clean up the 'State' names by removing "State of " prefix
state_actions['State'] = state_actions['Agency'].str.replace("State of ",
 ↪  "").str.strip()
```

```
# Step 4: Count the number of enforcement actions per state
state_counts = state_actions['State'].value_counts().reset_index()
state_counts.columns = ['State', 'Enforcement_Actions']

# Step 5: Load the state boundaries shapefile
state_shapefile_path =
 ↪  r"/Users/sohyunlim/Desktop/python-ps5-shilm/cb_2018_us_state_500k/cb_2018_us_state_500k.s
states = gpd.read_file(state_shapefile_path)

# Step 6: Merge the state shapefile with the enforcement data
merged_states = states.merge(state_counts, left_on="NAME", right_on="State",
 ↪  how="left")
merged_states['Enforcement_Actions'] =
 ↪  merged_states['Enforcement_Actions'].fillna(0)

# Step 7: Convert the GeoDataFrame to GeoJSON format
merged_states_json = json.loads(merged_states.to_json())
merged_states_data = alt.Data(values=merged_states_json['features'])

# Step 8: Create an Altair choropleth map
map_chart = alt.Chart(merged_states_data).mark_geoshape().encode(
    color=alt.Color('properties.Enforcement_Actions:Q', title='Enforcement
 ↪  Actions',
                    scale=alt.Scale(scheme='orangered')),
    tooltip=[
        alt.Tooltip('properties.State:N', title='State'),
        alt.Tooltip('properties.Enforcement_Actions:Q', title='Enforcement
 ↪  Actions')
    ]
).properties(
    title="Number of Enforcement Actions by State in 2021",
    width=500,
    height=300
).project(
    type='albersUsa'  # Projection for continental US
)

map_chart
```



2. Map by District (PARTNER 2)

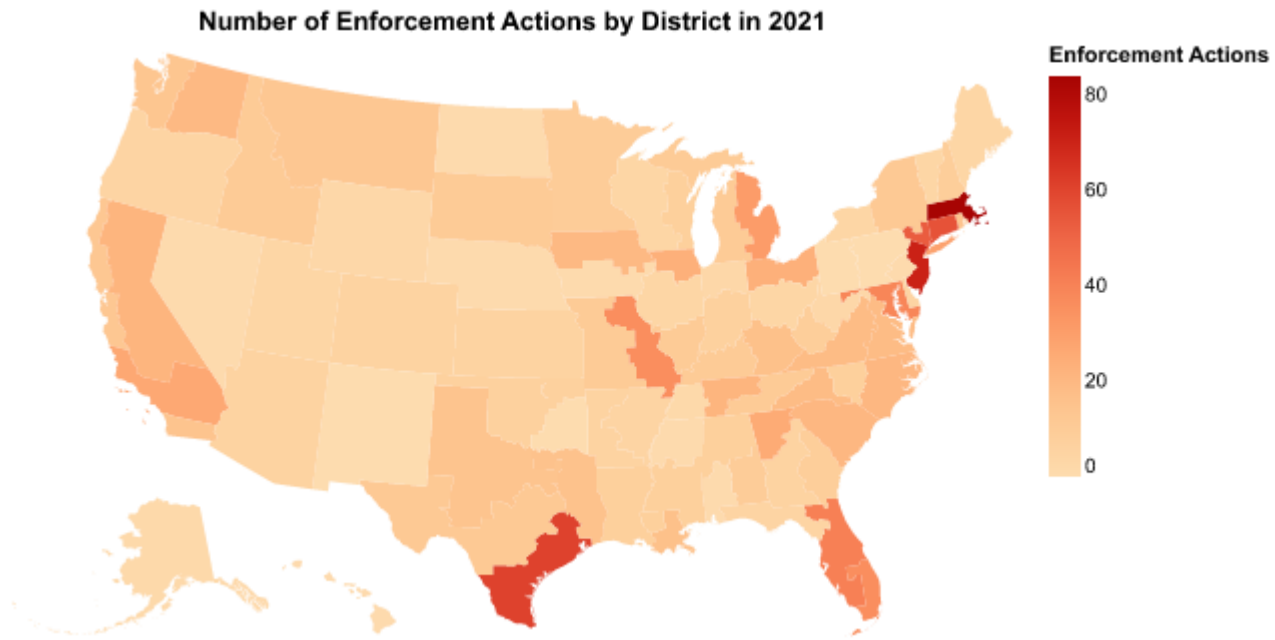
```
# First, clean the df_q3 dataset.

import pandas as pd

# Step 1: Extract district names that contain "District of ..." with any
 ↪  preceding text
# Store the result directly in a new DataFrame, df_q3_district
df_q3_district = df_q3.copy()
df_q3_district['District'] =
 ↪  df_q3_district['Agency'].str.extract(r'([A-Za-z\s]+District of
 ↪  [A-Za-z\s]+)')

# Step 2: Remove "s Office" from the extracted District names
df_q3_district['District'] = df_q3_district['District'].str.replace(r"s
 ↪  Office", "", regex=True).str.strip()

# Fix misspelled name of District
df_q3_district['District'] = df_q3_district['District'].str.replace('District
 ↪  of Idaho Boise', 'District of Idaho', regex=False)
df_q3_district['District'] = df_q3_district['District'].str.replace('District
 ↪  of Pennsylvani', 'District of Pennsylvania', regex=False)
```

```
df_q3_district['District'] = df_q3_district['District'].str.replace('Southern
 ↪  District of Florida and U', 'Southern District of Florida', regex=False)
df_q3_district['District'] = df_q3_district['District'].str.replace('Southern
 ↪  District of Texas and U', 'Southern District of Texas', regex=False)
df_q3_district['District'] = df_q3_district['District'].str.replace('Western
 ↪  District of Kentucky and U', 'Western District of Kentucky', regex=False)

# Step 3: Drop rows where 'District' is NaN
df_q3_district = df_q3_district.dropna(subset=['District'])

# Display the first few rows of the new DataFrame to verify
df_q3_district[['Agency', 'District']].head()
```

	Agency	District
0	U.S. Attorney's Office, Eastern District of Vi...	Eastern District of Virginia
1	U.S. Attorney's Office, Middle District of Flo...	Middle District of Florida
2	U.S. Attorney's Office, Western District of Texas	Western District of Texas
3	U.S. Attorney's Office, Eastern District of Mi...	Eastern District of Michigan
4	U.S. Attorney's Office, Eastern District of Te...	Eastern District of Tennessee

```
# Next, create a map.

import geopandas as gpd
import pandas as pd
import matplotlib.pyplot as plt
from mpl_toolkits.axes_grid1.inset_locator import inset_axes

# Load the shapefile
shapefile_path = r"/Users/sohyunlim/Desktop/python-ps5-shilm/US Attorney
 ↪  Districts Shapefile
 ↪  simplified_20241107/geo_export_615ad4c8-8694-4e4a-b9e8-13417c441590.shp"
districts_gdf = gpd.read_file(shapefile_path)

district_counts = df_q3_district['District'].value_counts().reset_index()
district_counts.columns = ['District', 'Count']

# Merge the enforcement data with shapefile data on district names
merged_gdf = districts_gdf.merge(district_counts, left_on='judicial_d',
 ↪  right_on='District', how='left')
merged_gdf['Count'] = merged_gdf['Count'].fillna(0)  # Fill missing values
 ↪  with 0
```

```
# Convert the GeoDataFrame to GeoJSON format
merged_gdf_json = json.loads(merged_gdf.to_json())
merged_gdf_data = alt.Data(values=merged_gdf_json['features'])

# Create an Altair choropleth map
map_chart_district = alt.Chart(merged_gdf_data).mark_geoshape().encode(
    color=alt.Color('properties.Count:Q', title='Enforcement Actions',
                    scale=alt.Scale(scheme='orangered')),
    tooltip=[
        alt.Tooltip('properties.District:N', title='State'),
        alt.Tooltip('properties.Count:Q', title='Enforcement Actions')
    ]
).properties(
    title="Number of Enforcement Actions by District in 2021",
    width=500,
    height=300
).project(
    type='albersUsa'  # Projection for continental US
)

map_chart_district
```



Extra Credit

1. Merge zip code shapefile with population

```
import geopandas as gpd
import pandas as pd

# Load the shapefile
shapefile_path_zipcode =
 ↪  "/Users/sohyunlim/Desktop/python-ps5-shilm/gz_2010_us_860_00_500k/gz_2010_us_860_00_500k
zipcode = gpd.read_file(shapefile_path_zipcode)

# Load the csv file
file_path_ppl =
 ↪  r"/Users/sohyunlim/Desktop/python-ps5-shilm/DECENNIALDHC2020.P1_2024-11-08T093912/DECENNI
ppl = pd.read_csv(file_path_ppl, header=0)
ppl = ppl.drop(0).reset_index(drop=True)
ppl = ppl.loc[:, ~ppl.columns.str.contains('Unnamed: 3')]

# Merge zipcode shp with ppl csv
zipcode_ppl = zipcode.merge(ppl, left_on='GEO_ID', right_on='GEO_ID')
```

2. Conduct spatial join

3. Map the action ratio in each district