PS5

AUTHOR
Attaullah Abbasi

PUBLISHED
November 9, 2024

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 - Attaullah Abbasi
   - Partner 1 (Attaullah Abbasi (attaullahabbasi))
   - Partner 2 : NA; completed Solo.
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: AA
5. "I have uploaded the names of anyone else other than my partner and I worked with on the problem set **here AA" (1 point)
6. Late coins used this pset: NA Late coins left after submission: 0:1
7. Knit your `ps5.qmd` to an 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
import time

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

```
response = requests.get(url)

# Checking the response status
if response.status_code == 200:
    soup = BeautifulSoup(response.content, 'html.parser')

    # Finding all action items on the page
    actions = soup.find_all('li', class_='usa-card card--list pep-card--minimal mobile:grid-col-12')

    # Checking if actions list is empty
    if not actions:
        print("No actions found. Check the class name in the code.")
    else:
        print(f"Found {len(actions)} actions.")

    # Initialize lists for data storage
    titles = []
    dates = []
    categories = []
    links = []
    agencies = []

    # Looping through each action and extracting data
    for action in actions:
        # Extracting the title
        title_tag = action.find('h2', class_='usa-card__heading')
        title = title_tag.get_text(strip=True) if title_tag else "No Title Found"
        titles.append(title)

        # Extracting the date
        date_tag = action.find('div', class_='font-body-sm margin-top-1')
        date = date_tag.find('span', class_='text-base-dark padding-right-105').get_text(strip=True) if
        date_tag else "No Date Found"
        dates.append(date)

        # Extracting the category
        category_tag = date_tag.find('li', class_='display-inline-block usa-tag text-no-lowercase text-base-
        darkest bg-base-lightest margin-right-1') if date_tag else None
        category = category_tag.get_text(strip=True) if category_tag else "No Category Found"
        categories.append(category)

        # Extracting the link and creating hyperlink HTML
        link_tag = title_tag.find('a') if title_tag else None
        link = f"https://oig.hhs.gov{link_tag['href']}" if link_tag else "No Link Found"
        links.append(f'<a href="{link}" target="_blank">Link</a>')  # Create hyperlink

        # Visit each link to get the agency information
        if link_tag:
            action_response = requests.get(link)
            if action_response.status_code == 200:
                action_soup = BeautifulSoup(action_response.content, 'html.parser')

                # Locating the agency information in the "Action Details" section
                agency = "Agency Info Not Found"
                details_section = action_soup.find('ul', class_='usa-list usa-list--unstyled margin-y-2')
                if details_section:
                    # Search for 'Agency:' text in the list items
                    list_items = details_section.find_all('li')
                    for item in list_items:
                        if 'Agency:' in item.get_text():
                            agency = item.get_text(strip=True).split("Agency:")[-1].strip()
                            break  # Exit after finding the agency
                agencies.append(agency)

                # To be polite to the server, add a delay
                time.sleep(1)  # 1-second delay
            else:
                agencies.append("Failed to load page")
        else:
            agencies.append("No Link Found")

    # Creating a DataFrame
    df = pd.DataFrame({
        'Title': titles,
        'Date': dates,
        'Category': categories,
```

```
        'Link': links,
        'Agency': agencies
    })

    # Convert DataFrame to HTML with custom CSS for Quarto
    styled_html = df.to_html(escape=False, index=False, border=0, classes='table table-striped',
        justify='left')

    # Adding CSS to ensure the table fits nicely in Quarto
    custom_css = """
    <style>
        .table {
            width: 100%;
            table-layout: fixed;
        }
        .table td, .table th {
            word-wrap: break-word;
            padding: 8px;
            vertical-align: top;
        }
    </style>
    """

    # Display styled HTML table with CSS in Jupyter or Quarto
    from IPython.display import display, HTML
    display(HTML(custom_css + styled_html))

else:
    print(f"Failed to retrieve data. Status code: {response.status_code}")
```

```
Found 20 actions.
```

Title	Date	Category	Link	Agency
Pharmacist and Brother Convicted of $15M Medicare, Medicaid, and Private Insurer Fraud Scheme	November 8, 2024	Criminal and Civil Actions	Link	U.S. Department of Justice
Boise Nurse Practitioner Sentenced To 48 Months For Conspiracy To Distribute Controlled Substances	November 7, 2024	Criminal and Civil Actions	Link	November 7, 2024; U.S. Attorney's Office, District of Idaho
Former Traveling Nurse Pleads Guilty To Tampering With Morphine	November 7, 2024	Criminal and Civil Actions	Link	U.S. Attorney's Office, District of Massachusetts
Former Arlington Resident Sentenced To Prison For Disclosing	November 7, 2024	Criminal and Civil Actions	Link	U.S. Attorney's Office, Eastern District of Virginia

Title	Date	Category	Link	Agency
Healthcare Records And Trying To Cover Up His Crimes				
Paroled Felon Sentenced To Six Years For Fraudulent Use Of Social Security Number And Theft Of Benefits	November 7, 2024	Criminal and Civil Actions	Link	U.S. Attorney's Office, Middle District of Florida
Former Licensed Counselor Sentenced For Defrauding Medicaid	November 6, 2024	Criminal and Civil Actions	Link	U.S. Attorney's Office, Western District of Texas
Macomb County Doctor And Pharmacist Agree To Pay $700,948 To Settle False Claims Act Allegations	November 4, 2024	Criminal and Civil Actions	Link	U.S. Attorney's Office, Eastern District of Michigan
Rocky Hill Pharmacy And Its Owners Indicted For Conspiring To Submit False Pharmacy Claims, Making False Statements, And Aggravated Identity Theft	November 4, 2024	Criminal and Civil Actions	Link	U.S. Attorney's Office, Eastern District of Tennessee
North Texas Medical Center Pays $14.2 Million To Resolve Potential False Claims Act Liability For Self-Reported Violations Of Medicare Regs, Stark Law	November 4, 2024	Criminal and Civil Actions	Link	U.S. Attorney's Office, Northern District of Texas

Title	Date	Category	Link	Agency
New England Doctor Pleads Guilty To Drug Distribution Conspiracy	November 4, 2024	Criminal and Civil Actions	Link	U.S. Department of Justice
Attorney General Alan Wilson Announces Upstate Woman Charged With Stealing Thousands Of Dollars From Vulnerable Adult	November 4, 2024	State Enforcement Agencies	Link	State of South Carolina
St. Louis County Woman Accused Of $3 Million Home Health Care Fraud	November 1, 2024	Criminal and Civil Actions	Link	U.S. Attorney's Office, Eastern District of Missouri
Lab Owner And Marketing Company Owner Both Found Guilty In Multi-Million Dollar Medicare And Medicaid Fraud Scheme	November 1, 2024	Criminal and Civil Actions	Link	U.S. Attorney's Office, Middle District of Tennessee
Compound Ingredient Supplier Medisca Inc., To Pay $21.75M To Resolve Allegations Of False And Inflated Average Wholesale Prices For Ingredients Used In Compounded Prescriptions	November 1, 2024	Criminal and Civil Actions	Link	U.S. Department of Justice
The New Mexico Department Of Justice Charges Former New Mexico State Police Officer	November 1, 2024	State Enforcement Agencies	Link	State of New Mexico

Title	Date	Category	Link	Agency
With Medicaid Fraud, Identity Theft, And Practicing Medicine Without A License				
Nashville Woman Indicted, Charged In TBI Medicaid Patient Abuse Case	November 1, 2024	State Enforcement Agencies	Link	State of Tennessee
Michael DePalma, MD and Virginia I-Spine Physicians Agreed to Pay $153,000 for Allegedly Violating the Civil Monetary Penalties Law by Submitting Claims for Services that Exceeded the Allowable Number of Services	October 31, 2024	CMP and Affirmative Exclusions	Link	Agency Info Not Found
Columbus Doctor, His Clinic Convicted of $1.5 Million Medicaid Scheme	October 31, 2024	State Enforcement Agencies	Link	Ohio
Mercy Health Youngstown Agreed to Pay $69,000 for Allegedly Violating the Civil Monetary Penalties Law by Contracting with an Excluded Individual	October 30, 2024	Fraud Self-Disclosures	Link	Agency Info Not Found
Quincy-Based Physician Group To Pay $650,000 To Resolve Allegations Of False Billing To MassHealth	October 30, 2024	State Enforcement Agencies	Link	State of Massachusetts

2. Crawling (PARTNER 1)

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

# Define the URL to scrape
url = "https://oig.hhs.gov/fraud/enforcement/"
response = requests.get(url)

if response.status_code == 200:
    soup = BeautifulSoup(response.content, 'html.parser')
    actions = soup.find_all('li', class_='usa-card card--list pep-card--minimal mobile:grid-col-12')

    # Initialize lists for data storage
    titles, dates, categories, links, agencies = [], [], [], [], []

    for action in actions:
        # Extract title
        title_tag = action.find('h2', class_='usa-card__heading')
        title = title_tag.get_text(strip=True) if title_tag else "No Title Found"
        titles.append(title)

        # Extract date
        date_tag = action.find('div', class_='font-body-sm margin-top-1')
        date = date_tag.find('span', class_='text-base-dark padding-right-105').get_text(strip=True) if
        date_tag else "No Date Found"
        dates.append(date)

        # Extract category
        category_tag = date_tag.find('li', class_='display-inline-block usa-tag text-no-lowercase text-base-
        darkest bg-base-lightest margin-right-1') if date_tag else None
        category = category_tag.get_text(strip=True) if category_tag else "No Category Found"
        categories.append(category)

        # Extract link
        link_tag = title_tag.find('a') if title_tag else None
        link = f"https://oig.hhs.gov{link_tag['href']}" if link_tag else "No Link Found"
        links.append(f'<a href="{link}" target="_blank">Link</a>')  # HTML link format

        # Visit each link to get the agency information
        agency = "Agency Info Not Found"
        if link_tag:
            action_response = requests.get(link)
            if action_response.status_code == 200:
                action_soup = BeautifulSoup(action_response.content, 'html.parser')
                details_section = action_soup.find('ul', class_='usa-list usa-list--unstyled margin-y-2')
                if details_section:
                    for item in details_section.find_all('li'):
                        if 'Agency:' in item.get_text():
                            agency = item.get_text(strip=True).split("Agency:")[-1].strip()
                            break
        agencies.append(agency)

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

    # Convert DataFrame to HTML with custom CSS
    styled_html = df.to_html(escape=False, index=False, border=0, classes='table table-striped',
        justify='left')

    # Add CSS for styling
    custom_css = """
    <style>
        .table {
            width: 100%;
            table-layout: fixed;
            font-size: 0.9em;
        }
        .table td, .table th {
            word-wrap: break-word;
```

```
            padding: 8px;
            vertical-align: top;
        }
        .table th {
            background-color: #f2f2f2;
            text-align: center;
        }
    </style>
    """

    # Display styled HTML table with CSS in Jupyter or Quarto
    from IPython.display import display, HTML
    display(HTML(custom_css + styled_html))

else:
    print(f"Failed to retrieve data. Status code: {response.status_code}")
```

Title	Date	Category	Link	Agency
Pharmacist and Brother Convicted of $15M Medicare, Medicaid, and Private Insurer Fraud Scheme	November 8, 2024	Criminal and Civil Actions	Link	U.S. Department of Justice
Boise Nurse Practitioner Sentenced To 48 Months For Conspiracy To Distribute Controlled Substances	November 7, 2024	Criminal and Civil Actions	Link	November 7, 2024; U.S. Attorney's Office, District of Idaho
Former Traveling Nurse Pleads Guilty To Tampering With Morphine	November 7, 2024	Criminal and Civil Actions	Link	U.S. Attorney's Office, District of Massachusetts
Former Arlington Resident Sentenced To Prison For Disclosing Healthcare Records And Trying To Cover Up His Crimes	November 7, 2024	Criminal and Civil Actions	Link	U.S. Attorney's Office, Eastern District of Virginia
Paroled Felon Sentenced To Six Years For Fraudulent Use Of Social Security	November 7, 2024	Criminal and Civil Actions	Link	U.S. Attorney's Office, Middle District of Florida

Title	Date	Category	Link	Agency
Number And Theft Of Benefits				
Former Licensed Counselor Sentenced For Defrauding Medicaid	November 6, 2024	Criminal and Civil Actions	Link	U.S. Attorney's Office, Western District of Texas
Macomb County Doctor And Pharmacist Agree To Pay $700,948 To Settle False Claims Act Allegations	November 4, 2024	Criminal and Civil Actions	Link	U.S. Attorney's Office, Eastern District of Michigan
Rocky Hill Pharmacy And Its Owners Indicted For Conspiring To Submit False Pharmacy Claims, Making False Statements, And Aggravated Identity Theft	November 4, 2024	Criminal and Civil Actions	Link	U.S. Attorney's Office, Eastern District of Tennessee
North Texas Medical Center Pays $14.2 Million To Resolve Potential False Claims Act Liability For Self-Reported Violations Of Medicare Regs, Stark Law	November 4, 2024	Criminal and Civil Actions	Link	U.S. Attorney's Office, Northern District of Texas
New England Doctor Pleads Guilty To Drug Distribution Conspiracy	November 4, 2024	Criminal and Civil Actions	Link	U.S. Department of Justice
Attorney General Alan Wilson Announces Upstate Woman Charged With Stealing	November 4, 2024	State Enforcement Agencies	Link	State of South Carolina

Title	Date	Category	Link	Agency
Thousands Of Dollars From Vulnerable Adult				
St. Louis County Woman Accused Of $3 Million Home Health Care Fraud	November 1, 2024	Criminal and Civil Actions	Link	U.S. Attorney's Office, Eastern District of Missouri
Lab Owner And Marketing Company Owner Both Found Guilty In Multi-Million Dollar Medicare And Medicaid Fraud Scheme	November 1, 2024	Criminal and Civil Actions	Link	U.S. Attorney's Office, Middle District of Tennessee
Compound Ingredient Supplier Medisca Inc., To Pay $21.75M To Resolve Allegations Of False And Inflated Average Wholesale Prices For Ingredients Used In Compounded Prescriptions	November 1, 2024	Criminal and Civil Actions	Link	U.S. Department of Justice
The New Mexico Department Of Justice Charges Former New Mexico State Police Officer With Medicaid Fraud, Identity Theft, And Practicing Medicine Without A License	November 1, 2024	State Enforcement Agencies	Link	State of New Mexico
Nashville Woman Indicted, Charged In TBI Medicaid Patient Abuse Case	November 1, 2024	State Enforcement Agencies	Link	State of Tennessee

Title	Date	Category	Link	Agency
Michael DePalma, MD and Virginia I-Spine Physicians Agreed to Pay $153,000 for Allegedly Violating the Civil Monetary Penalties Law by Submitting Claims for Services that Exceeded the Allowable Number of Services	October 31, 2024	CMP and Affirmative Exclusions	Link	Agency Info Not Found
Columbus Doctor, His Clinic Convicted of $1.5 Million Medicaid Scheme	October 31, 2024	State Enforcement Agencies	Link	Ohio
Mercy Health Youngstown Agreed to Pay $69,000 for Allegedly Violating the Civil Monetary Penalties Law by Contracting with an Excluded Individual	October 30, 2024	Fraud Self-Disclosures	Link	Agency Info Not Found
Quincy-Based Physician Group To Pay $650,000 To Resolve Allegations Of False Billing To MassHealth	October 30, 2024	State Enforcement Agencies	Link	State of Massachusetts

Step 2: Making the scraper dynamic

1. Turning the scraper into a function

- a. Pseudo-Code (PARTNER 2)

Pseudo-Code for `scrape_enforcement_actions(month, year)`

1. **Input Validation**:

    - Check if `year >= 2013`. If `year < 2013`, print a message to the user: `"Please provide a year >= 2013 as only enforcement actions after 2013 are available."`
    - If the year is valid, proceed to the next step.
2. **Setup Variables**:
    - Define the base URL for the enforcement actions page.
    - Create an empty list to store the extracted data for each action (title, date, category, link, and agency).
3. **Loop through the Pages**:
    - While the current date is on or after the specified start month and year:
        - Construct the URL dynamically to include the page number (if applicable).
        - Send a request to the server and parse the page with BeautifulSoup.
        - Wait for 1 second between requests to avoid server-side blocking.
        - If the page contains enforcement actions:
            - For each action item:
                - Extract the **title**.
                - Extract the **date**. Stop if the date is earlier than the specified `month` and `year`.
                - Extract the **category**.
                - Extract the **link** and visit it to get the **agency**.
                - Append the data (title, date, category, link, and agency) to the list.
        - If the page has no more actions (or there's a date cutoff), break the loop.
4. **Create and Save the DataFrame**:
    - Convert the list of data to a pandas DataFrame.
    - Save the DataFrame as a CSV file named `enforcement_actions_<year>_<month>.csv`.
5. **Return** the DataFrame or a confirmation message.

Discussion of Loops

- **While Loop**: The function will use a `while` loop to continue scraping pages until:
    - The date in the scraped data goes beyond the current date.
    - The dates in the enforcement actions are before the start date specified (to stop scraping earlier records).
- **For Loop**: Inside each page, a `for` loop will iterate over the enforcement actions listed, extracting data for each item.

This structure ensures that we dynamically scrape data starting from the user-specified date and save it in the specified format, all while respecting server limitations with a 1-second delay between requests.

- b. Create Dynamic Scraper (PARTNER 2)

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

async def fetch_agency(link, session):
    agency = "Not Available"
    try:
        async with session.get(link) as response:
```

```
            if response.status == 200:
                page_content = await response.text()
                action_soup = BeautifulSoup(page_content, 'html.parser')
                details_section = action_soup.find('ul', class_='usa-list usa-list--unstyled margin-y-2')
                if details_section:
                    list_items = details_section.find_all('li')
                    for item in list_items:
                        if 'Agency:' in item.get_text():
                            agency = item.get_text(strip=True).split("Agency:")[-1].strip()
                            break
    except Exception as e:
        print(f"Failed to fetch agency for link: {link}. Error: {e}")
    return agency

async def scrape_enforcement_actions(start_year, start_month, fetch_agency_details=False):
    # Check the input year
    if start_year < 2013:
        print("Please enter a year >= 2013, as only enforcement actions after 2013 are listed.")
        return None

    # Base URL
    base_url = "https://oig.hhs.gov/fraud/enforcement/"

    # Initialize lists to store data
    titles, dates, categories, links, agencies = [], [], [], [], []
    page = 1
    all_data_scraped = False  # Flag to indicate if scraping is complete

    # Create an aiohttp session if fetching agency details
    async with aiohttp.ClientSession() as session:
        # Iterate through pages until reaching the target start date or end of pages
        while not all_data_scraped:
            # Construct the URL for each page
            url = f"{base_url}?page={page}"
            async with session.get(url) as response:
                if response.status != 200:
                    print(f"Failed to retrieve page {page}. Status code: {response.status}")
                    break

                # Parse the page content
                page_content = await response.text()
                soup = BeautifulSoup(page_content, 'html.parser')

                # Find all action items on the page
                actions = soup.find_all('li', class_='usa-card card--list pep-card--minimal mobile:grid-col-
    12')
                if not actions:
                    print(f"No more actions found. Ending scraping at page {page}.")
                    break

                for action in actions:
                    # Extract title
                    title_tag = action.find('h2', class_='usa-card__heading')
                    title = title_tag.get_text(strip=True) if title_tag else "No Title Found"
                    titles.append(title)

                    # Extract date
                    date_tag = action.find('div', class_='font-body-sm margin-top-1')
                    date = date_tag.find('span', class_='text-base-dark padding-right-
    105').get_text(strip=True) if date_tag else "No Date Found"
                    dates.append(date)

                    # Check if the date is within the desired range (Jan 2023 onwards)
                    date_obj = pd.to_datetime(date, errors='coerce')
                    if date_obj and (date_obj < pd.Timestamp(f"{start_year}-{start_month}-01")):
                        all_data_scraped = True  # Stop further scraping if date is too old
                        break

                    # Extract category
                    category_tag = date_tag.find('li', class_='display-inline-block usa-tag text-no-lowercase
    text-base-darkest bg-base-lightest margin-right-1') if date_tag else None
                    category = category_tag.get_text(strip=True) if category_tag else "No Category Found"
                    categories.append(category)

                    # Extract link
                    link_tag = title_tag.find('a') if title_tag else None
```

```
                    link = f"https://oig.hhs.gov{link_tag['href']}" if link_tag else "No Link Found"
                    links.append(link)

                    # Append agency info if needed, or add placeholder
                    if fetch_agency_details and link_tag:
                        agency = await fetch_agency(link, session)
                    else:
                        agency = "Not Retrieved"
                    agencies.append(agency)

                # Increment the page counter and add a delay for the next request
                page += 1
                await asyncio.sleep(1)  # Wait between pages

    # Check if all lists are of the same length, else print error message
    min_len = min(len(titles), len(dates), len(categories), len(links), len(agencies))
    titles, dates, categories, links, agencies = titles[:min_len], dates[:min_len], categories[:min_len],
        links[:min_len], agencies[:min_len]

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

    # Save DataFrame to CSV
    filename = f"enforcement_actions_{start_year}_{start_month}.csv"
    df.to_csv(filename, index=False)
    print(f"Data saved to {filename}")

    # Display summary of results
    print(f"Total enforcement actions scraped: {len(df)}")
    if not df.empty:
        earliest_action = df.iloc[-1]
        print("Earliest Enforcement Action:")
        print(f"Title: {earliest_action['Title']}")
        print(f"Date: {earliest_action['Date']}")
        print(f"Category: {earliest_action['Category']}")
        print(f"Link: {earliest_action['Link']}")
        print(f"Agency: {earliest_action['Agency']}")

# Example usage in a Jupyter Notebook or interactive environment:
await scrape_enforcement_actions(2023, 1, fetch_agency_details=True)
```

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

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

# Async function to fetch agency details
async def fetch_agency(link, session):
    agency = "Not Available"
    try:
        async with session.get(link) as response:
            if response.status == 200:
                page_content = await response.text()
                action_soup = BeautifulSoup(page_content, 'html.parser')
                details_section = action_soup.find('ul', class_='usa-list usa-list--unstyled margin-y-2')
                if details_section:
                    list_items = details_section.find_all('li')
                    for item in list_items:
                        if 'Agency:' in item.get_text():
                            agency = item.get_text(strip=True).split("Agency:")[-1].strip()
                            break
    except Exception as e:
        print(f"Failed to fetch agency for link: {link}. Error: {e}")
    return agency
```

```
# Main async function to scrape enforcement actions
async def scrape_enforcement_actions(start_year, start_month, fetch_agency_details=True):
    # Validate year input
    if start_year < 2013:
        print("Please enter a year >= 2013, as only enforcement actions after 2013 are listed.")
        return None

    # Base URL
    base_url = "https://oig.hhs.gov/fraud/enforcement/"

    # Initialize lists to store data
    titles, dates, categories, links, agencies = [], [], [], [], []
    page = 1
    all_data_scraped = False  # Flag to indicate if scraping is complete

    # Create an aiohttp session
    async with aiohttp.ClientSession() as session:
        while not all_data_scraped:
            # Construct URL for the page
            url = f"{base_url}?page={page}"
            async with session.get(url) as response:
                if response.status != 200:
                    print(f"Failed to retrieve page {page}. Status code: {response.status}")
                    break

                # Parse the page content
                page_content = await response.text()
                soup = BeautifulSoup(page_content, 'html.parser')

                # Find all action items on the page
                actions = soup.find_all('li', class_='usa-card card--list pep-card--minimal mobile:grid-col-
    12')
                if not actions:
                    print(f"No more actions found. Ending scraping at page {page}.")
                    break

                for action in actions:
                    # Extract title
                    title_tag = action.find('h2', class_='usa-card__heading')
                    title = title_tag.get_text(strip=True) if title_tag else "No Title Found"
                    titles.append(title)

                    # Extract date
                    date_tag = action.find('div', class_='font-body-sm margin-top-1')
                    date = date_tag.find('span', class_='text-base-dark padding-right-
    105').get_text(strip=True) if date_tag else "No Date Found"
                    dates.append(date)

                    # Check if date is within desired range
                    date_obj = pd.to_datetime(date, errors='coerce')
                    if date_obj and (date_obj < pd.Timestamp(f"{start_year}-{start_month}-01")):
                        all_data_scraped = True
                        break

                    # Extract category
                    category_tag = date_tag.find('li', class_='display-inline-block usa-tag text-no-lowercase
    text-base-darkest bg-base-lightest margin-right-1') if date_tag else None
                    category = category_tag.get_text(strip=True) if category_tag else "No Category Found"
                    categories.append(category)

                    # Extract link
                    link_tag = title_tag.find('a') if title_tag else None
                    link = f"https://oig.hhs.gov{link_tag['href']}" if link_tag else "No Link Found"
                    links.append(link)

                    # Fetch agency info if needed
                    if fetch_agency_details and link_tag:
                        agency = await fetch_agency(link, session)
                    else:
                        agency = "Not Retrieved"
                    agencies.append(agency)

                # Increment the page counter and add a delay for the next request
                page += 1
                await asyncio.sleep(1)  # Wait between pages
```

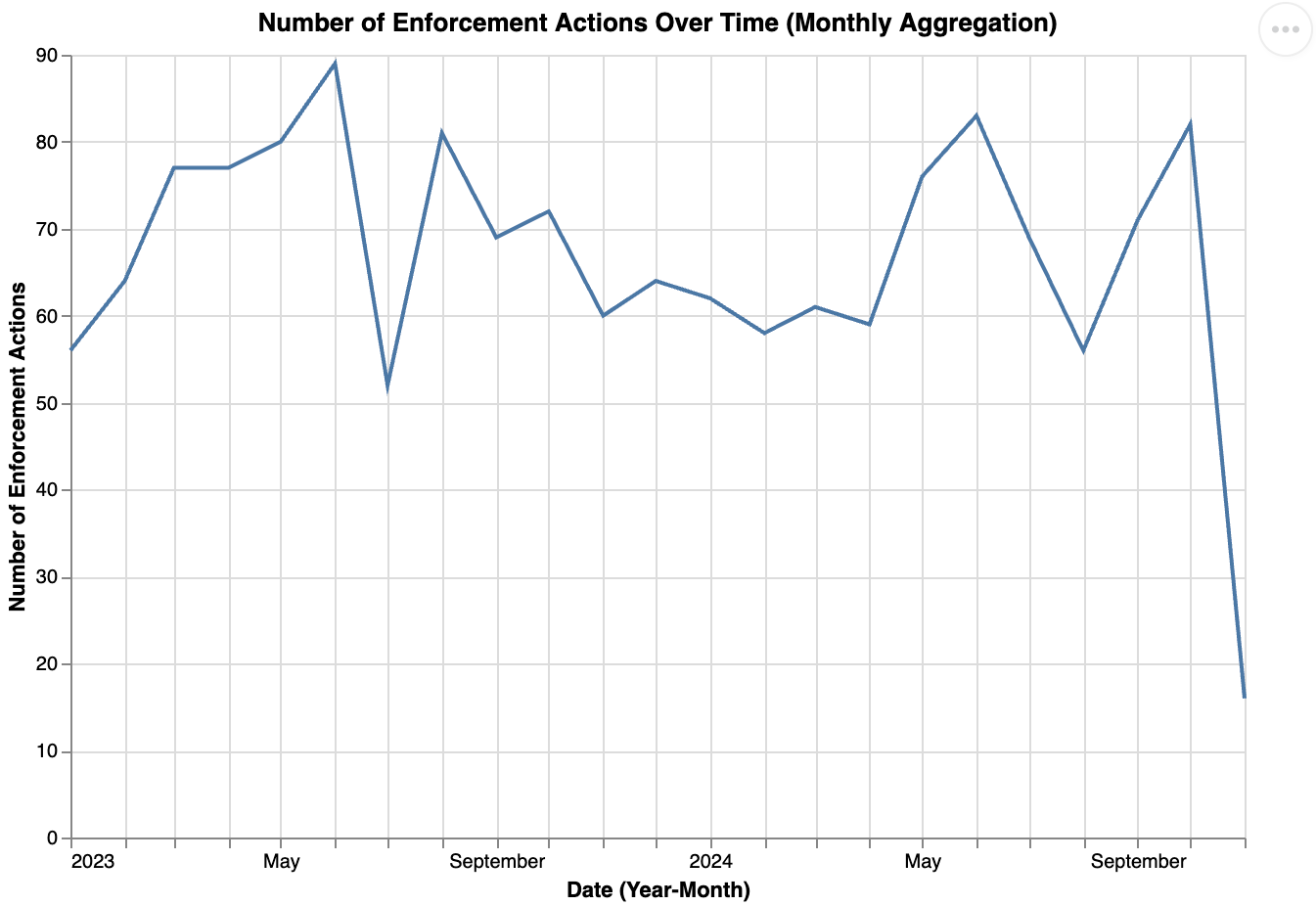
```
    # Ensure all lists are of equal length
    min_len = min(len(titles), len(dates), len(categories), len(links), len(agencies))
    titles, dates, categories, links, agencies = titles[:min_len], dates[:min_len], categories[:min_len],
        links[:min_len], agencies[:min_len]

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

    # Save DataFrame to CSV
    filename = f"enforcement_actions_{start_year}_{start_month}.csv"
    df.to_csv(filename, index=False)
    print(f"Data saved to {filename}")

    # Display summary of results
    print(f"Total enforcement actions scraped: {len(df)}")
    if not df.empty:
        earliest_action = df.iloc[-1]
        print("Earliest Enforcement Action:")
        print(f"Title: {earliest_action['Title']}")
        print(f"Date: {earliest_action['Date']}")
        print(f"Category: {earliest_action['Category']}")
        print(f"Link: {earliest_action['Link']}")
        print(f"Agency: {earliest_action['Agency']}")

# Example usage (to be run in a Jupyter Notebook or async-compatible environment):
await scrape_enforcement_actions(2023, 1, fetch_agency_details=True)
```

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
df = pd.read_csv("/Users/attaullah/Documents/PS5/enforcement_actions_2023_1_cleaned.csv")  # Update the file
        path as needed

# Convert 'Date' column to datetime format, handling any errors
df['Date'] = pd.to_datetime(df['Date'], errors='coerce')

# Drop rows with invalid dates
df = df.dropna(subset=['Date'])

# Group by year and month, then count the number of actions per month
df['YearMonth'] = df['Date'].dt.to_period('M')  # Use 'M' for monthly grouping
monthly_counts = df.groupby('YearMonth').size().reset_index(name='Enforcement_Actions')

# Convert YearMonth back to datetime for proper plotting in Altair
monthly_counts['YearMonth'] = monthly_counts['YearMonth'].dt.to_timestamp()

# Create a line chart using Altair
line_chart = alt.Chart(monthly_counts).mark_line().encode(
    x=alt.X('YearMonth:T', title='Date (Year-Month)'),
    y=alt.Y('Enforcement_Actions:Q', title='Number of Enforcement Actions')
).properties(
    title='Number of Enforcement Actions Over Time (Monthly Aggregation)',
    width=600,
    height=400
)

# Enable default renderer (useful in Jupyter Notebooks)
alt.renderers.enable("default")
```

```
# Display the chart
line_chart
```

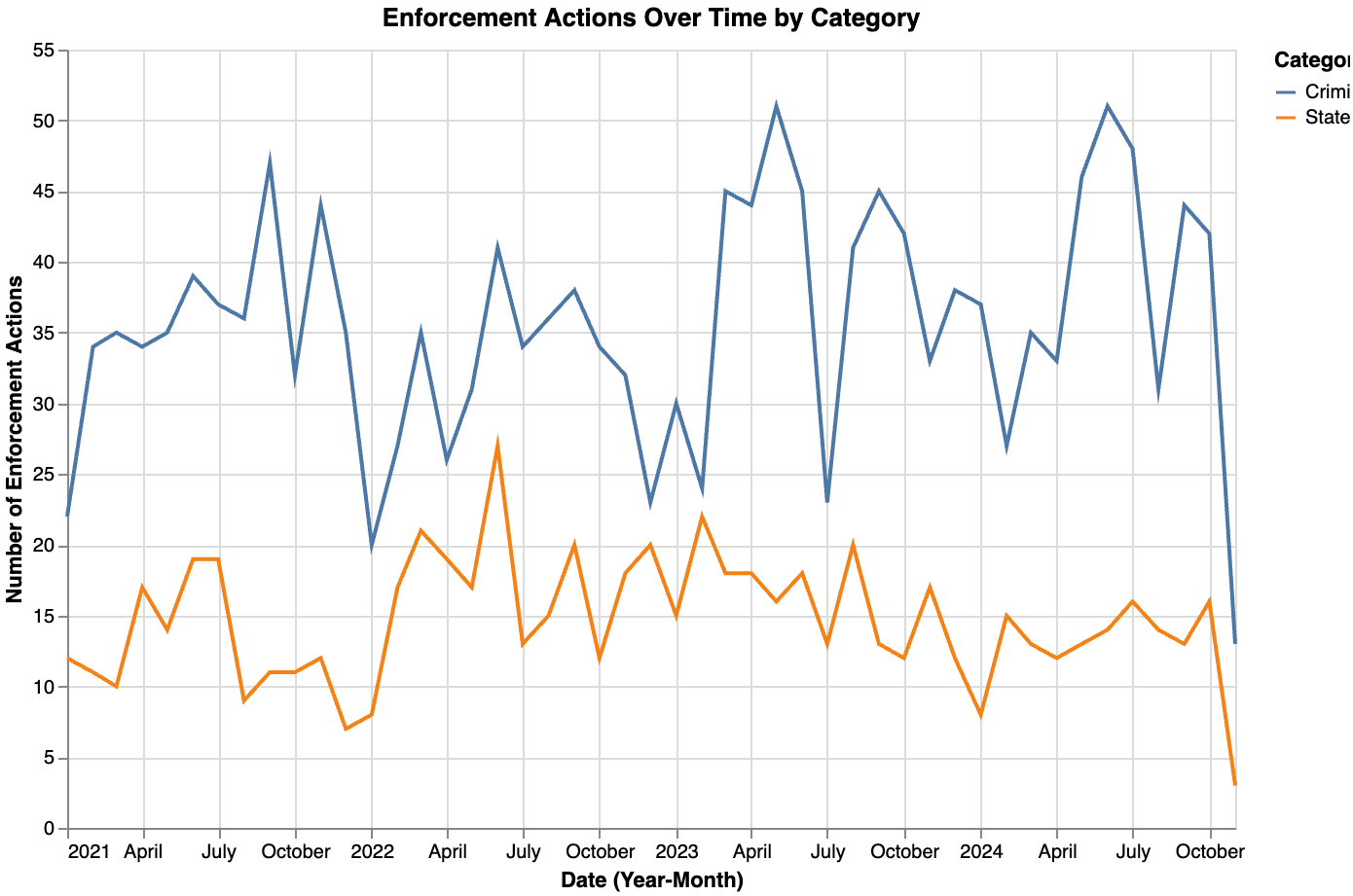


2. Plot the number of enforcement actions categorized: (PARTNER 1)

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

```
import pandas as pd
import altair as alt

# Load the data
df = pd.read_csv("enforcement_actions_2021_1.csv")

# Convert 'Date' to datetime and drop rows with invalid dates
df['Date'] = pd.to_datetime(df['Date'], errors='coerce')
df = df.dropna(subset=['Date'])

# Filter for "Criminal and Civil Actions" and "State Enforcement Agencies" categories
df_filtered = df[df['Category'].isin(['Criminal and Civil Actions', 'State Enforcement Agencies'])]

# Group by year and month, and count the number of actions for each category
df_filtered['YearMonth'] = df_filtered['Date'].dt.to_period('M').astype(str)
monthly_counts = df_filtered.groupby(['YearMonth',
        'Category']).size().reset_index(name='Enforcement_Actions')

# Convert YearMonth back to datetime for plotting
monthly_counts['YearMonth'] = pd.to_datetime(monthly_counts['YearMonth'], format='%Y-%m')

# Plot the data using Altair
line_chart = alt.Chart(monthly_counts).mark_line().encode(
    x=alt.X('YearMonth:T', title='Date (Year-Month)'),
    y=alt.Y('Enforcement_Actions:Q', title='Number of Enforcement Actions'),
    color=alt.Color('Category:N', title='Category')
).properties(
    title='Enforcement Actions Over Time by Category',
```

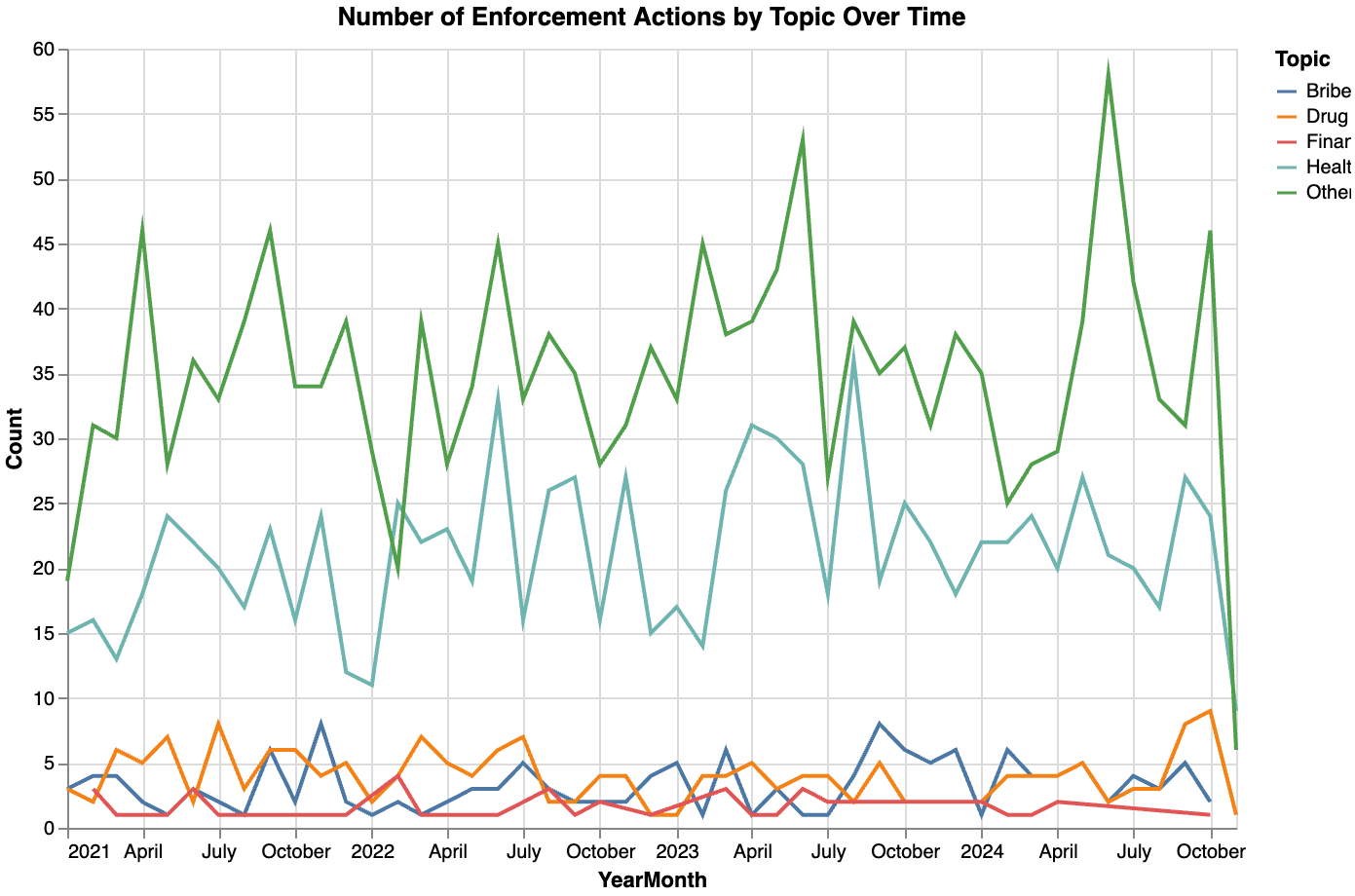
```
    width=600,
    height=400
)

line_chart.show()
```



- based on five topics

```
import pandas as pd
import altair as alt

# Load your data from the CSV file created in Step 2 (replace with actual file path if needed)
df = pd.read_csv("enforcement_actions_2021_1.csv")

# Convert Date to datetime format and extract month-year for aggregation
df['Date'] = pd.to_datetime(df['Date'], errors='coerce')
df['YearMonth'] = df['Date'].dt.to_period('M')

# Define a function to categorize topics
def categorize_topic(title):
    title = title.lower()
    if any(keyword in title for keyword in ["medicare", "medicaid", "hospital", "pharmacy", "doctor"]):
        return "Health Care Fraud"
    elif any(keyword in title for keyword in ["bank", "financial", "investment", "money laundering"]):
        return "Financial Fraud"
    elif any(keyword in title for keyword in ["drug", "narcotics", "opioid", "prescription"]):
        return "Drug Enforcement"
    elif any(keyword in title for keyword in ["bribery", "corruption", "kickback"]):
        return "Bribery/Corruption"
    else:
        return "Other"

# Apply categorization to each title
df['Topic'] = df['Title'].apply(categorize_topic)
```

```
# Group by YearMonth and Topic, then count occurrences
monthly_topic_counts = df.groupby(['YearMonth', 'Topic']).size().reset_index(name='Count')

# Convert YearMonth back to datetime for Altair compatibility
monthly_topic_counts['YearMonth'] = monthly_topic_counts['YearMonth'].dt.to_timestamp()

# Plotting the line chart with Altair
chart = alt.Chart(monthly_topic_counts).mark_line().encode(
    x='YearMonth:T',
    y='Count:Q',
    color='Topic:N'
).properties(
    title="Number of Enforcement Actions by Topic Over Time",
    width=600,
    height=400
)

chart
```

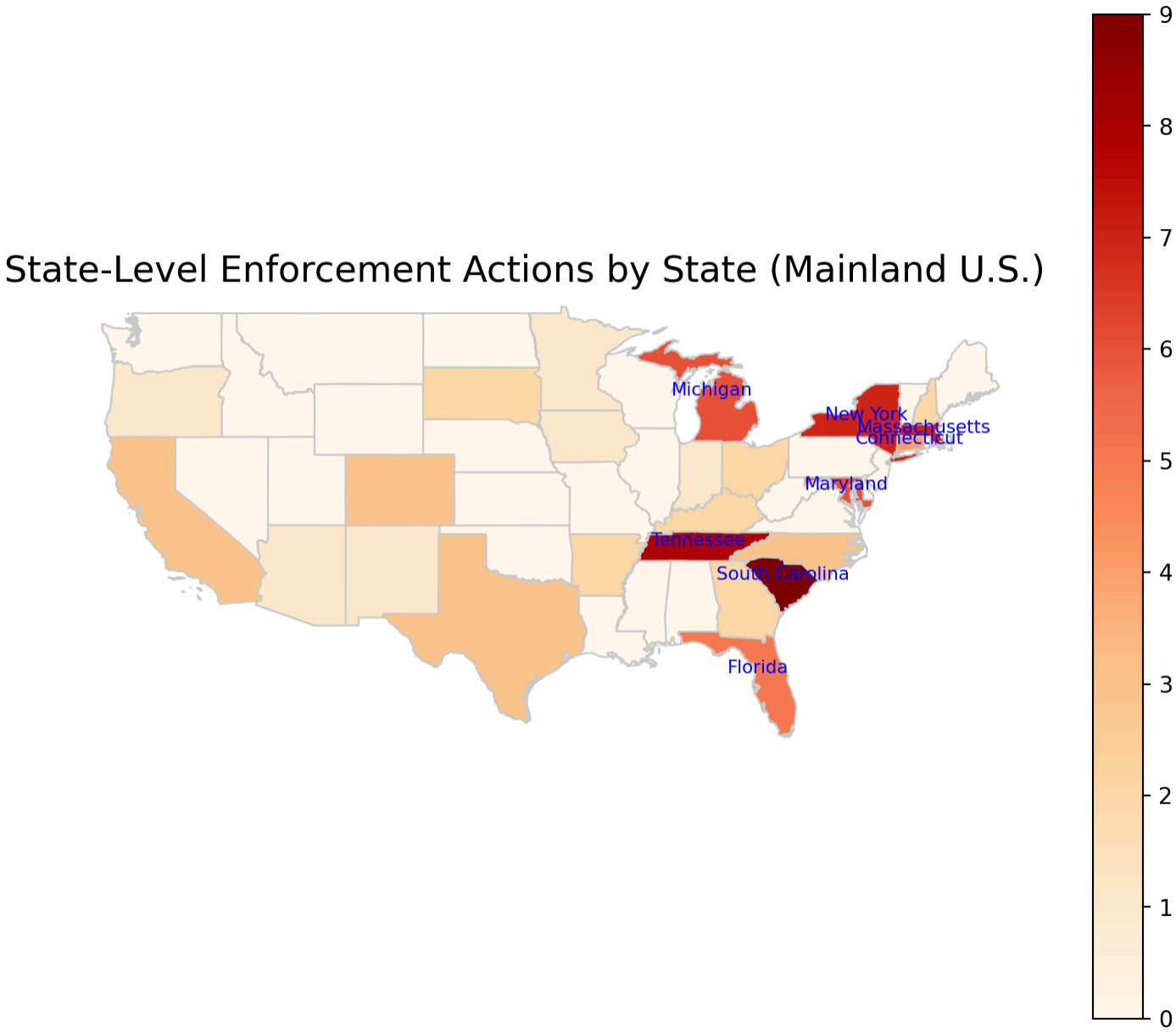


Step 4: Create maps of enforcement activity

1. Map by State (PARTNER 1)

```
import geopandas as gpd
import pandas as pd
import matplotlib.pyplot as plt
import re

# Define paths
state_shapefile_path = "/Users/attaullah/Documents/PS5/cb_2018_us_state_500k/cb_2018_us_state_500k.shp"
```

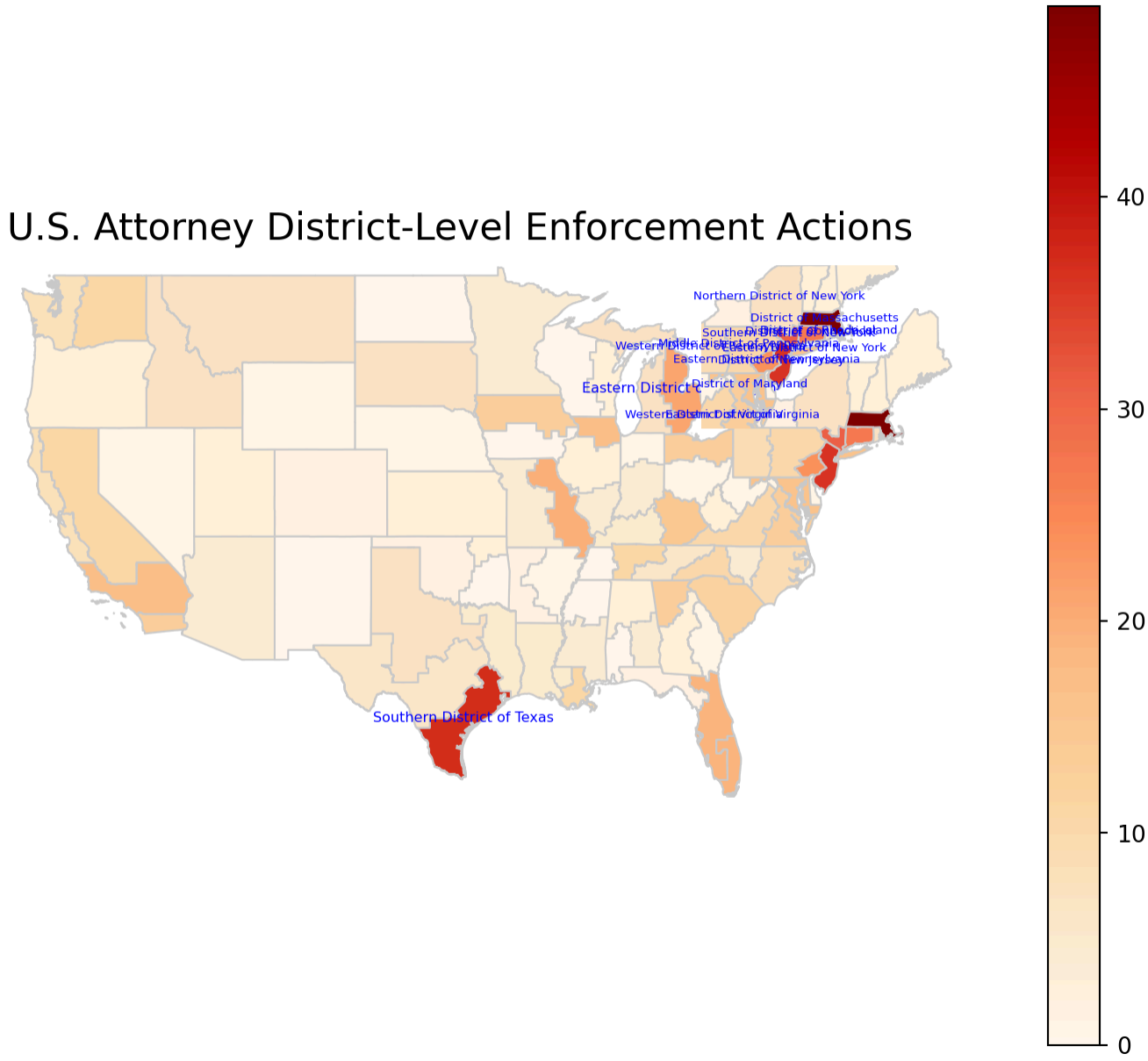
```
csv_path = "/Users/attaullah/Documents/PS5/enforcement_actions_2023_1_cleaned.csv"

# Load the state shapefile and enforcement actions data
states_gdf = gpd.read_file(state_shapefile_path)
actions_df = pd.read_csv(csv_path)

# Extract and clean state names from the 'Agency' column
def extract_state(agency_name):
    match = re.search(r"State of ([A-Za-z\s]+)", agency_name)
    return match.group(1).strip() if match else None

actions_df['State'] = actions_df['Agency'].apply(extract_state)

# Group actions by state
state_actions = actions_df.dropna(subset=['State']).groupby('State').size().reset_index(name='ActionCount')

# Rename and merge to ensure compatibility
states_gdf = states_gdf.rename(columns={"NAME": "State"})
merged_gdf = states_gdf.merge(state_actions, on="State", how="left").fillna({'ActionCount': 0})

# Plotting the choropleth map
fig, ax = plt.subplots(1, 1, figsize=(10, 8))
merged_gdf.plot(column="ActionCount", cmap="OrRd", linewidth=0.8, ax=ax, edgecolor="0.8", legend=True)

# Customize map appearance
plt.title("State-Level Enforcement Actions by State (Mainland U.S.)", fontsize=16)
plt.axis("off")
ax.set_xlim(-130, -65)
ax.set_ylim(25, 50)

# Label states with higher action counts
for x, y, label, count in zip(merged_gdf.geometry.centroid.x, merged_gdf.geometry.centroid.y,
        merged_gdf["State"], merged_gdf["ActionCount"]):
    if count > 3:
        ax.text(x, y, label, fontsize=8, ha="center", color="blue")

plt.show()
```



2. Map by District (PARTNER 2)

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

# Updated path to the shapefile and CSV data
shapefile_path = "/Users/attaullah/Documents/PS5/US Attorney Districts Shapefile
        simplified_20241108/geo_export_695e4db7-129b-4351-bfc9-31ef5c1e3499.shp"
csv_path = "/Users/attaullah/Documents/PS5/enforcement_actions_2023_1_cleaned.csv"

# Load the shapefile and CSV data
districts_gdf = gpd.read_file(shapefile_path)
actions_df = pd.read_csv(csv_path)

# Extract and clean district names in the 'Agency' column in actions_df
def extract_district(agency_name):
    match = re.search(r"(District of [A-Za-z\s]+|[A-Za-z]+ District of [A-Za-z\s]+)", agency_name)
    if match:
```

```
        return match.group(0).strip()
    return None

actions_df['District'] = actions_df['Agency'].apply(extract_district)
district_actions = actions_df.dropna(subset=
        ['District']).groupby('District').size().reset_index(name='ActionCount')

# Rename judicial_d to District in districts_gdf for merging
districts_gdf = districts_gdf.rename(columns={"judicial_d": "District"})

# Merge the data
merged_gdf = districts_gdf.merge(district_actions, on="District", how="left")
merged_gdf["ActionCount"] = merged_gdf["ActionCount"].fillna(0)  # Fill missing values with 0

# Main map plot
fig, ax = plt.subplots(1, 1, figsize=(10, 8))
merged_gdf.plot(column="ActionCount", cmap="OrRd", linewidth=0.8, ax=ax, edgecolor="0.8", legend=True)

# Add title and labels
plt.title("U.S. Attorney District-Level Enforcement Actions", fontsize=16)
plt.axis("off")

# Set plot limits to zoom in on mainland U.S.
ax.set_xlim(-130, -65)
ax.set_ylim(25, 50)

# Define boundaries for the Northeast region for inset
northeast_long = -80
northeast_lat = 40

# Add labels with a different threshold for the Northeast
for x, y, label, count in zip(merged_gdf.geometry.centroid.x, merged_gdf.geometry.centroid.y,
        merged_gdf["District"], merged_gdf["ActionCount"]):
    # Relaxed threshold for the main map, only high-action districts in the Northeast
    if count > 20:
        if x > northeast_long and y > northeast_lat:
            continue  # Skip labeling Northeast in main map
        else:
            ax.text(x, y, label, fontsize=6, ha="center", color="blue")

# Create an inset map for the Northeast
inset_ax = inset_axes(ax, width="30%", height="30%", loc="upper right")
merged_gdf.plot(column="ActionCount", cmap="OrRd", linewidth=0.8, ax=inset_ax, edgecolor="0.8")

# Adjust inset map view to focus on the Northeast
inset_ax.set_xlim(-80, -65)
inset_ax.set_ylim(37, 45)
inset_ax.axis("off")

# Add labels for the inset map with a lower threshold for more detail
for x, y, label, count in zip(merged_gdf.geometry.centroid.x, merged_gdf.geometry.centroid.y,
        merged_gdf["District"], merged_gdf["ActionCount"]):
    if -80 < x < -65 and 37 < y < 45 and count > 5:
        inset_ax.text(x, y, label, fontsize=5, ha="center", color="blue")

plt.show()
```

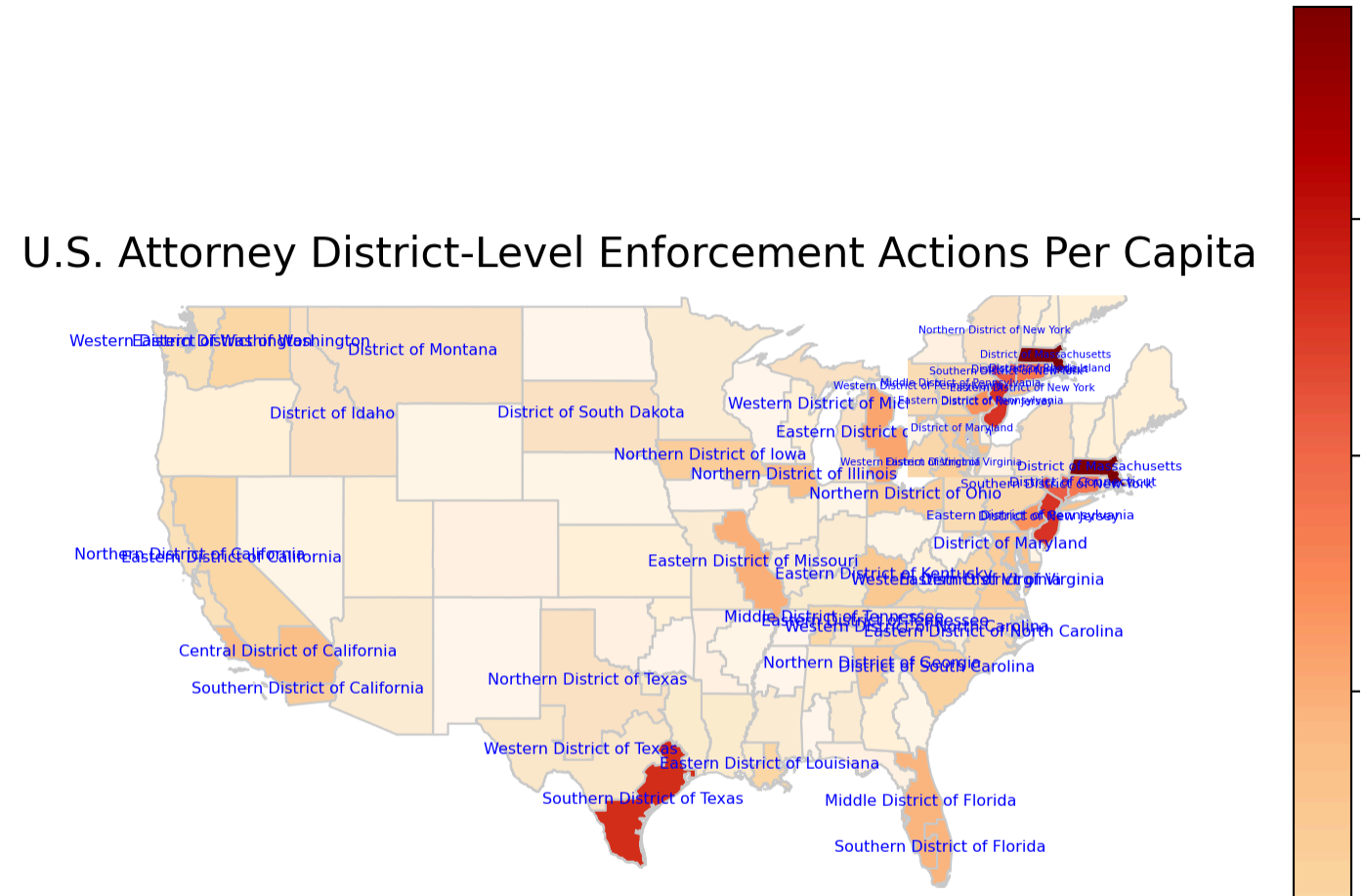


Extra Credit

1. Merge zip code shapefile with population

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

# Paths to files
districts_shapefile_path = "/Users/attaullah/Documents/PS5/US Attorney Districts Shapefile
        simplified_20241108/geo_export_695e4db7-129b-4351-bfc9-31ef5c1e3499.shp"
enforcement_data_path = "/Users/attaullah/Documents/PS5/enforcement_actions_2023_1_cleaned.csv"
population_data_path = "/Users/attaullah/Documents/PS5/DECENNIALDHC2020.P1-Data.csv"

# Load the shapefile and enforcement actions data
districts_gdf = gpd.read_file(districts_shapefile_path)
enforcement_df = pd.read_csv(enforcement_data_path)
```

```
population_df = pd.read_csv(population_data_path)

# Clean and extract zip codes in population data
population_df['ZIP'] = population_df['NAME'].str.extract(r'(\d{5})')
population_df = population_df[['ZIP', 'P1_001N']].rename(columns={'P1_001N': 'Population'})

# Clean 'Agency' field in enforcement data to extract district names
def extract_district(agency_name):
    match = re.search(r"(District of [A-Za-z\s]+|[A-Za-z]+ District of [A-Za-z\s]+)", agency_name)
    return match.group(0).strip() if match else None

enforcement_df['District'] = enforcement_df['Agency'].apply(extract_district)

# Calculate per-capita actions by district
district_actions = enforcement_df.groupby('District').size().reset_index(name='ActionCount')
population_by_district = population_df.groupby('ZIP').sum().reset_index()

# Merge data
merged_gdf = districts_gdf.merge(district_actions, left_on='judicial_d', right_on='District',
        how='left').fillna({'ActionCount': 0})
merged_gdf = merged_gdf.merge(population_by_district, left_on='district_n', right_on='ZIP', how='left')

# Fill missing population values with 1 to avoid division by zero
merged_gdf['Population'] = merged_gdf['Population'].fillna(1)  # To avoid NaNs or division by zero

# Calculate actions per capita
merged_gdf['Actions_Per_Capita'] = merged_gdf['ActionCount'] / merged_gdf['Population']

# Main map plot
fig, ax = plt.subplots(1, 1, figsize=(10, 8))
merged_gdf.plot(column="Actions_Per_Capita", cmap="OrRd", linewidth=0.8, ax=ax, edgecolor="0.8", legend=True)

# Customize map appearance
plt.title("U.S. Attorney District-Level Enforcement Actions Per Capita", fontsize=16)
plt.axis("off")
ax.set_xlim(-130, -65)
ax.set_ylim(25, 50)

# Define the boundaries for the Northeast region
northeast_long = -80
northeast_lat = 40

# Add labels with a different threshold for the Northeast
for x, y, label, count in zip(merged_gdf.geometry.centroid.x, merged_gdf.geometry.centroid.y,
        merged_gdf["District"], merged_gdf["Actions_Per_Capita"]):
    if x > northeast_long and y > northeast_lat:
        # Only show very high-count labels in the Northeast to avoid clutter
        if count > 20:  # Adjust threshold for Northeast labeling
            ax.text(x, y, label, fontsize=5, ha="center", color="blue")
    else:
        # More relaxed threshold for labels outside the Northeast
        if count > 5:  # General threshold for other areas
            ax.text(x, y, label, fontsize=6, ha="center", color="blue")

# Create an inset map for the Northeast
inset_ax = inset_axes(ax, width="30%", height="30%", loc="upper right")
merged_gdf.plot(column="Actions_Per_Capita", cmap="OrRd", linewidth=0.8, ax=inset_ax, edgecolor="0.8")

# Adjust inset map view to focus on the Northeast
inset_ax.set_xlim(-80, -65)
inset_ax.set_ylim(37, 45)
inset_ax.axis("off")

# Add labels for the inset map with a relaxed threshold
for x, y, label, count in zip(merged_gdf.geometry.centroid.x, merged_gdf.geometry.centroid.y,
        merged_gdf["District"], merged_gdf["Actions_Per_Capita"]):
    if -80 < x < -65 and 37 < y < 45:
        if count > 5:  # Lower threshold for the inset to show more details in the Northeast
            inset_ax.text(x, y, label, fontsize=4, ha="center", color="blue")

plt.show()
```



2. Conduct spatial join

3. Map the action ratio in each district