PS5

Dizhe Xia & Qiyin Yao

2024-11-08

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): Dizhe Xia; dizhexia
    - Partner 2 (name and cnet ID): Kevin Yao; qiyin
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: **_DX_** **_QY_**(We both use late token)
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: **_1_** Late coins left after submission: **_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 geopandas as gpd
import matplotlib.pyplot as plt
from shapely.affinity import translate, scale
import requests
from shapely.geometry import shape
from bs4 import BeautifulSoup
import time
from datetime import datetime
from concurrent.futures import ThreadPoolExecutor, as_completed
from datetime import datetime
from concurrent.futures import ThreadPoolExecutor, as_completed
import random
import re
import pandas as pd
import altair as alt
import requests
from bs4 import BeautifulSoup
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)

```
# Base URL to construct full URLs for links
base_url = "https://oig.hhs.gov"
# URL of the Enforcement Actions page
url = f"{base_url}/fraud/enforcement/"

# Send a GET request to the page
response = requests.get(url)
soup = BeautifulSoup(response.content, 'html.parser')

# Lists to store scraped data
titles = []
dates = []
categories = []
links = []

# Find all enforcement action entries
action_entries = soup.find_all('li', class_='usa-card')  # Each entry container

for entry in action_entries:
    # Extract title
    title_tag = entry.find('h2', class_='usa-card__heading')
    title = title_tag.get_text(strip=True) if title_tag else "N/A"
    titles.append(title)

    # Extract date
    date_tag = entry.find('span', class_='text-base-dark padding-right-105')  # Date span class
    date = date_tag.get_text(strip=True) if date_tag else "N/A"
    dates.append(date)

    # Extract category
    category_tag = entry.find('li', class_='display-inline-block usa-tag text-no-lowercase text-base-darkest bg-base-lightest margin-right-1')
    category = category_tag.get_text(strip=True) if category_tag else "N/A"
    categories.append(category)

    # Extract link
    link_tag = title_tag.find('a') if title_tag else None
    link = f"{base_url}{link_tag['href']}" if link_tag and link_tag.has_attr('href') else "N/A"
    links.append(link)

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

# Display the first few rows of the DataFrame
print(data.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)

```
# Assuming `data` is the existing DataFrame with 'Title', 'Date', 'Category',
 ↪  and 'Link' columns

# Initialize a list to store agency names
agencies = []

# Loop through each link in the 'Link' column of the DataFrame
for link in data['Link']:
    agency = "N/A"  # Default value in case the agency is not found
    if link != "N/A":
        # Send a GET request to the detailed page
        detail_response = requests.get(link)
        detail_soup = BeautifulSoup(detail_response.content, 'html.parser')

        # Locate the "Agency:" label within its list item <li>
        agency_label = detail_soup.find('span', text='Agency:')
        if agency_label:
            # Get the parent <li> tag, then extract the entire text after
             ↪  "Agency:"
            agency_text = agency_label.parent.get_text(strip=True)
            # Remove "Agency:" from the text to leave only the agency name
            agency = agency_text.replace("Agency:", "").strip()

    # Append the extracted agency to the agencies list
    agencies.append(agency)

# Add the 'Agency' column to the DataFrame
data['Agency'] = agencies

print(data.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-...

                                              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...
```

Step 2: Making the scraper dynamic

1. Turning the scraper into a function

(1) Define the function scrape_enforcement_actions(month, year):

- Input: month (integer) and year (integer).
- Output: Saves scraped data into a CSV file named enforcement_actions_year_month.csv.

(2) Check the Year Validity:

- If year < 2013, print a message: "Data only available for enforcement actions from 2013 onwards. Please enter a year >= 2013."
- If year >= 2013, proceed with the scraping process.

(3) Set Up Required Libraries:

- Import requests, BeautifulSoup, pandas, concurrent.futures (for threading), time, datetime, and random modules.

(4) Define the Base URL and Initial URL:

- Set base_url to "https://oig.hhs.gov".
- Set enforcement_path to "/fraud/enforcement/".
- Construct the initial URL by combining base_url, enforcement_path, and the parameters for year and month to start scraping from the specified date.

(5) Initialize Empty Lists for Storing Data:

- Create lists: titles, dates, categories, and links to hold data for each enforcement action.

(6) Set Up a Loop to Scrape Data from Multiple Pages:

- Use a while True loop to keep scraping each page until no more pages are left.
  - Send a GET Request to the current page URL using requests.get.
  - Parse the page content with BeautifulSoup.
  - Extract data from each enforcement action entry:
    - Find the title, date, category, and link for each entry.
    - Append each of these to the respective lists.
  - Check if any enforcement action entry date is before January 2023:
    - If yes, break the loop.
  - If a "next page" link is available, update the page URL and use time.sleep(2) to pause to avoid blocking.
  - If no further pages are found, break the loop.

(7) Use Concurrent Threading to Fetch Additional Details (Agency):

- Create an empty list called agencies.
- Use ThreadPoolExecutor to send requests to each detailed enforcement action page concurrently.
  - For each link, send a GET request.
  - Parse the detailed page to find the "Agency:" label using BeautifulSoup.
  - Extract and store the agency data in the agencies list.

(8) Create a DataFrame from Collected Data:

- Use pandas.DataFrame to create a DataFrame with columns Title, Date, Category, Link, and Agency.

(9) Save the DataFrame to a CSV File:

- Use DataFrame.to_csv to save the DataFrame as "enforcement_actions_year_month.csv", formatted with the provided year and month.

(10) Print Confirmation:

- Print a message indicating the data has been saved to the specified CSV file.

- b. Create Dynamic Scraper (PARTNER 2)

```
# Switch to control code execution
run_scraping = False  # Set to True to run the code, False to skip it
import pandas as pd

if run_scraping:
    from datetime import datetime
    from concurrent.futures import ThreadPoolExecutor, as_completed
    import random
    import time
    import requests
    from bs4 import BeautifulSoup
```

```
    def fetch_detail_info(link):
        """Fetch agency information from detail page"""
        try:
            # Add random delay to simulate manual operation
            time.sleep(random.uniform(2, 4))

            # Send a request to the detailed page
            detail_response = requests.get(link, headers={
                "User-Agent": "Mozilla/5.0 (Windows NT 10.0; Win64; x64)
                 ↪  AppleWebKit/537.36 (KHTML, like Gecko)
                 ↪  Chrome/91.0.4472.124 Safari/537.36"
            })

            if detail_response.status_code != 200:
                print(f"Failed to fetch detail page: {link}, Status Code:
                 ↪  {detail_response.status_code}")
                return "N/A"

            detail_soup = BeautifulSoup(detail_response.content,
↪   'html.parser')

            # Find the <span> tag with the specified class and get its parent
             ↪  <li> tag's text
            agency_label = detail_soup.find('span', class_='padding-right-2
↪   text-base', text='Agency:')
            if agency_label:
                # Get the parent <li> tag's text and remove "Agency:" to
                 ↪  extract the actual agency name
                agency =
↪   agency_label.parent.get_text(strip=True).replace("Agency:", "").strip()
            else:
                # Debug: Print the link if the <span> tag is not found
                print(f"Agency label not found in: {link}")
                agency = "N/A"  # Use "N/A" if the "Agency" information is
↪   not found
        except Exception as e:
            print(f"Error fetching details for {link}: {e}")
            agency = "N/A"  # Use "N/A" in case of an error
        return agency

    def scrape_enforcement_actions(year, month):
        if year < 2013:
            print("Data only available for enforcement actions from 2013
             ↪  onwards. Please enter a year >= 2013.")
            return

        base_url = "https://oig.hhs.gov"
        enforcement_path = "/fraud/enforcement/"
        titles, dates, categories, links = [], [], [], []
```

```
        page = 1
        while True:
            print(f"Scraping page {page}...")
            url =
↪  f"{base_url}{enforcement_path}?page={page}&year={year}&month={month}"
            response = requests.get(url, headers={
                "User-Agent": "Mozilla/5.0 (Windows NT 10.0; Win64; x64)
                ↪  AppleWebKit/537.36 (KHTML, like Gecko)
                ↪  Chrome/91.0.4472.124 Safari/537.36"
            })
            if response.status_code != 200:
                print(f"Failed to fetch main page: {url}, Status Code:
                ↪  {response.status_code}")
                break

            soup = BeautifulSoup(response.content, 'html.parser')

            action_entries = soup.find_all('li', class_='usa-card')
            if not action_entries:
                print("No more entries found, ending scrape.")
                break
            stop_scraping = False
            for entry in action_entries:
                title_tag = entry.find('h2', class_='usa-card__heading')
                title = title_tag.get_text(strip=True) if title_tag else
↪  "N/A"
                titles.append(title)

                date_tag = entry.find('span', class_='text-base-dark
↪  padding-right-105')
                date = date_tag.get_text(strip=True) if date_tag else "N/A"

                # Check if date is earlier than January 2023
                try:
                    action_date = datetime.strptime(date, "%B %d, %Y")
                    if action_date.year < 2023:
                        stop_scraping = True
                        break
                except ValueError:
                    pass
               dates.append(date)
               category_tag = entry.find('li', class_='display-inline-block
↪  usa-tag text-no-lowercase text-base-darkest bg-base-lightest
↪  margin-right-1')
               category = category_tag.get_text(strip=True) if category_tag
↪  else "N/A"
               categories.append(category)
               link_tag = title_tag.find('a') if title_tag else None
               link = f"{base_url}{link_tag['href']}" if link_tag and
↪  link_tag.has_attr('href') else "N/A"
```

```
                link = link.replace("enforcement//", "enforcement/")  #
↪  Remove double slashes if present
                links.append(link)
            if stop_scraping:
                print("Reached entries from 2022 or earlier, stopping
                 ↪  scrape.")
                break
            page += 1
            time.sleep(2)  # Adding a fixed delay between each page request
↪  to prevent blocking

        # Concurrently fetch agency information for each link
        agencies = ["N/A"] * len(links)  # Initialize with "N/A"
        with ThreadPoolExecutor(max_workers=5) as executor:
            future_to_index = {executor.submit(fetch_detail_info, links[i]):
↪  i for i in range(len(links)) if links[i] != "N/A"}
            for future in as_completed(future_to_index):
                index = future_to_index[future]
                try:
                    agency = future.result()
                    agencies[index] = agency  # Store the result in the
↪  correct index
                except Exception as exc:
                    print(f'{links[index]} generated an exception: {exc}')
                    agencies[index] = "N/A"

        # Ensure all lists have the same length
        categories.extend(["N/A"] * (len(titles) - len(categories)))
        links.extend(["N/A"] * (len(titles) - len(links)))
        agencies.extend(["N/A"] * (len(titles) - len(agencies)))

        # Debugging: Print lengths of each list to verify consistency
        print("Length of Titles:", len(titles))
        print("Length of Dates:", len(dates))
        print("Length of Categories:", len(categories))
        print("Length of Links:", len(links))
        print("Length of Agencies:", len(agencies))

        # Check if all lists have the same length before creating the
         ↪  DataFrame
        if len(titles) == len(dates) == len(categories) == len(links) ==
         ↪  len(agencies):
            data_2023 = pd.DataFrame({
                'Title': titles,
                'Date': dates,
                'Category': categories,
                'Link': links,
                'Agency': agencies
            })
```

```
            filename = f"enforcement_actions_{year}_{month}.csv"
            data_2023.to_csv(filename, index=False)
            print(f"Data saved to {filename}")

            print("Total enforcement actions:", len(data_2023))
            earliest_action = data_2023.sort_values(by='Date').iloc[0]
            print("Earliest enforcement action date:",
             ↪  earliest_action['Date'])
            print("Earliest enforcement action details:", earliest_action)
        else:
            print("Error: Lists are not of the same length.")
            print("Please check the debugging output for list lengths.")

    # Run the function for January 2023
    scrape_enforcement_actions(2023, 1)
else:
    print("CSV file already generated. Skipping code to avoid redundant data
     ↪  scraping and save time. To run the code, set 'run_scraping' to
     ↪  True.")


file_path =
↪  "/Users/xiadizhe/Documents/GitHub/problem-set-5-dylan-kevin/enforcement_actions_2023_1.csv
data = pd.read_csv(file_path)
print(data.head())
```

```
CSV file already generated. Skipping code to avoid redundant data scraping
and save time. To run the code, set 'run_scraping' to True.
                                               Title              Date  \
0  Boise Nurse Practitioner Sentenced To 48 Month...  November 7, 2024
1  Former Traveling Nurse Pleads Guilty To Tamper...  November 7, 2024
2  Former Arlington Resident Sentenced To Prison ...  November 7, 2024
3  Paroled Felon Sentenced To Six Years For Fraud...  November 7, 2024
4  Former Licensed Counselor Sentenced For Defrau...  November 6, 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/boise-nu...
1  https://oig.hhs.gov/fraud/enforcement/former-t...
2  https://oig.hhs.gov/fraud/enforcement/former-a...
3  https://oig.hhs.gov/fraud/enforcement/paroled-...
4  https://oig.hhs.gov/fraud/enforcement/former-l...
```

```
                                                    Agency
0  November 7, 2024; U.S. Attorney's Office, Dist...
                          1  U.S. Attorney's Office,
District of Massachusetts2  U.S. Attorney's Office,
Eastern District of Vi...3  U.S. Attorney's Office,
Middle District of Flo...4  U.S. Attorney's Office,
                           Western District of Texas
```

Number of enforcement actions: 1534

The earliest enforcement action it scraped:

Title: Podiatrist Pays $90,000 To Settle False Billing Allegations

Date: 3-Jan-23

Category: Criminal and Civil Actions

Link: https://oig.hhs.gov/fraud/enforcement/podiatrist-pays-90000-to-settle-false-billing-allegations/

Agency: U.S. Attorney's Office, Southern District of Texas

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

```
# Switch to control code execution
run_scraping = False  # Set to True to run the code, False to skip it
import pandas as pd

if run_scraping:
    from datetime import datetime
    from concurrent.futures import ThreadPoolExecutor, as_completed
    import random
    import time
    import requests
    from bs4 import BeautifulSoup

    def fetch_detail_info(link):
        """Fetch agency information from detail page"""
        try:
            # Add random delay to simulate manual operation
            time.sleep(random.uniform(2, 4))

            # Send a request to the detailed page
            detail_response = requests.get(link, headers={
                "User-Agent": "Mozilla/5.0 (Windows NT 10.0; Win64; x64)
                ↪ AppleWebKit/537.36 (KHTML, like Gecko)
                ↪ Chrome/91.0.4472.124 Safari/537.36"
            })

            if detail_response.status_code != 200:
                print(f"Failed to fetch detail page: {link}, Status Code:
                ↪ {detail_response.status_code}")
                return "N/A"

↪           detail_soup = BeautifulSoup(detail_response.content,
    'html.parser')
```

```
            # Find the <span> tag with the specified class and get its parent
                <li> tag's text
            agency_label = detail_soup.find('span', class_='padding-right-2
    text-base', text='Agency:')

            if agency_label:
                # Get the parent <li> tag's text and remove "Agency:" to
                    extract the actual agency name
                agency =
    agency_label.parent.get_text(strip=True).replace("Agency:", "").strip()
            else:
                # Debug: Print the link if the <span> tag is not found
                print(f"Agency label not found in: {link}")
                agency = "N/A"  # Use "N/A" if the "Agency" information is
    not found
        except Exception as e:
            print(f"Error fetching details for {link}: {e}")
            agency = "N/A"  # Use "N/A" in case of an error
        return agency
    def scrape_enforcement_actions(year, month):
        if year < 2013:
            print("Data only available for enforcement actions from 2013
                onwards. Please enter a year >= 2013.")
            return

        base_url = "https://oig.hhs.gov"
        enforcement_path = "/fraud/enforcement/"
        titles, dates, categories, links = [], [], [], []
        page = 1

        while True:
            print(f"Scraping page {page}...")
            url =
    f"{base_url}{enforcement_path}?page={page}&year={year}&month={month}"
            response = requests.get(url, headers={
                "User-Agent": "Mozilla/5.0 (Windows NT 10.0; Win64; x64)
                    AppleWebKit/537.36 (KHTML, like Gecko)
                    Chrome/91.0.4472.124 Safari/537.36"
            })
            if response.status_code != 200:
                print(f"Failed to fetch main page: {url}, Status Code:
                    {response.status_code}")
                break
            soup = BeautifulSoup(response.content, 'html.parser')

            action_entries = soup.find_all('li', class_='usa-card')
            if not action_entries:
                print("No more entries found, ending scrape.")
                break

            stop_scraping = False
            for entry in action_entries:
                title_tag = entry.find('h2', class_='usa-card__heading')
                title = title_tag.get_text(strip=True) if title_tag else
    "N/A"
                titles.append(title)
```

```
                date_tag = entry.find('span', class_='text-base-dark
↪  padding-right-105')
                date = date_tag.get_text(strip=True) if date_tag else "N/A"

                # Check if date is earlier than January 2021
                try:
                    action_date = datetime.strptime(date, "%B %d, %Y")
                    if action_date.year < 2021:
                        stop_scraping = True
                        break
                except ValueError:
                    pass

                dates.append(date)

                category_tag = entry.find('li', class_='display-inline-block
↪  usa-tag text-no-lowercase text-base-darkest bg-base-lightest
↪  margin-right-1')
                category = category_tag.get_text(strip=True) if category_tag
↪  else "N/A"
                categories.append(category)

                link_tag = title_tag.find('a') if title_tag else None
                link = f"{base_url}{link_tag['href']}" if link_tag and
↪  link_tag.has_attr('href') else "N/A"
                link = link.replace("enforcement//", "enforcement/")  #
↪  Remove double slashes if present
                links.append(link)

            if stop_scraping:
                print("Reached entries from 2020 or earlier, stopping
                ↪  scrape.")
                break

            page += 1
            time.sleep(2)  # Adding a fixed delay between each page request
  ↪  to prevent blocking

        # Concurrently fetch agency information for each link
        agencies = ["N/A"] * len(links)  # Initialize with "N/A"
        with ThreadPoolExecutor(max_workers=5) as executor:
            future_to_index = {executor.submit(fetch_detail_info, links[i]):
  ↪  i for i in range(len(links)) if links[i] != "N/A"}
            for future in as_completed(future_to_index):
                index = future_to_index[future]
                try:
                    agency = future.result()
                    agencies[index] = agency  # Store the result in the
  ↪  correct index
                except Exception as exc:
                    print(f'{links[index]} generated an exception: {exc}')
                    agencies[index] = "N/A"

        # Ensure all lists have the same length
        categories.extend(["N/A"] * (len(titles) - len(categories)))
        links.extend(["N/A"] * (len(titles) - len(links)))
        agencies.extend(["N/A"] * (len(titles) - len(agencies)))
```

```
        # Debugging: Print lengths of each list to verify consistency
        print("Length of Titles:", len(titles))
        print("Length of Dates:", len(dates))
        print("Length of Categories:", len(categories))
        print("Length of Links:", len(links))
        print("Length of Agencies:", len(agencies))

        # Check if all lists have the same length before creating the
        ↪ DataFrame
        if len(titles) == len(dates) == len(categories) == len(links) ==
        ↪ len(agencies):
            data = pd.DataFrame({
                'Title': titles,
            'Date': dates,
            'Category': categories,
            'Link': links,
            'Agency': agencies
        })

        filename = f"enforcement_actions_{year}_{month:02d}.csv"
        data.to_csv(filename, index=False)
        print(f"Data saved to {filename}")

        print("Total enforcement actions:", len(data))
        earliest_action = data.sort_values(by='Date').iloc[0]
        print("Earliest enforcement action date:",
        ↪ earliest_action['Date'])
        print("Earliest enforcement action details:", earliest_action)
    else:
        print("Error: Lists are not of the same length.")
        print("Please check the debugging output for list lengths.")

    # Loop through each month starting from January 2021 until the current
    ↪ month
    start_year = 2021
    start_month = 1
    end_year = datetime.now().year
    end_month = datetime.now().month

    current_year, current_month = start_year, start_month

    while (current_year < end_year) or (current_year == end_year and
    ↪ current_month <= end_month):
        print(f"Scraping data for {current_year}-{current_month:02d}")
        scrape_enforcement_actions(current_year, current_month)

        # Increment month and adjust year if needed
        current_month += 1
        if current_month > 12:
            current_month = 1
            current_year += 1

else:
    print("CSV files are already generated. Skipping code to avoid redundant
    ↪ data scraping and save time. To run the code, set 'run_scraping' to
    ↪ True.")
```

```
    file_path =
↪   "/Users/xiadizhe/Documents/GitHub/problem-set-5-dylan-kevin/enforcement_actions_2021_01.
    data = pd.read_csv(file_path)
    print(data.head())
```

```
CSV files are already generated. Skipping code to avoid redundant data
scraping and save time. To run the code, set 'run_scraping' to True.
                                               Title              Date  \
0  Former Arlington Resident Sentenced To Prison ...  November 7, 2024
1  Paroled Felon Sentenced To Six Years For Fraud...  November 7, 2024
2  Macomb County Doctor And Pharmacist Agree To P...  November 4, 2024
3  Rocky Hill Pharmacy And Its Owners Indicted Fo...  November 4, 2024
4  North Texas Medical Center Pays $14.2 Million ...  November 4, 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/former-a...
1  https://oig.hhs.gov/fraud/enforcement/paroled-...
2  https://oig.hhs.gov/fraud/enforcement/macomb-c...
3  https://oig.hhs.gov/fraud/enforcement/rocky-hi...
4  https://oig.hhs.gov/fraud/enforcement/north-te...

                                              Agency
0  U.S. Attorney's Office, Eastern District of Vi...
1  U.S. Attorney's Office, Middle District of Flo...
2  U.S. Attorney's Office, Eastern District of Mi...
3  U.S. Attorney's Office, Eastern District of Te...
4  U.S. Attorney's Office, Northern District of T...
```

Number of enforcement actions: 3525

The earliest enforcement action it scraped:

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

Date: 4-Jan-21

Category: Criminal and Civil Actions

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/

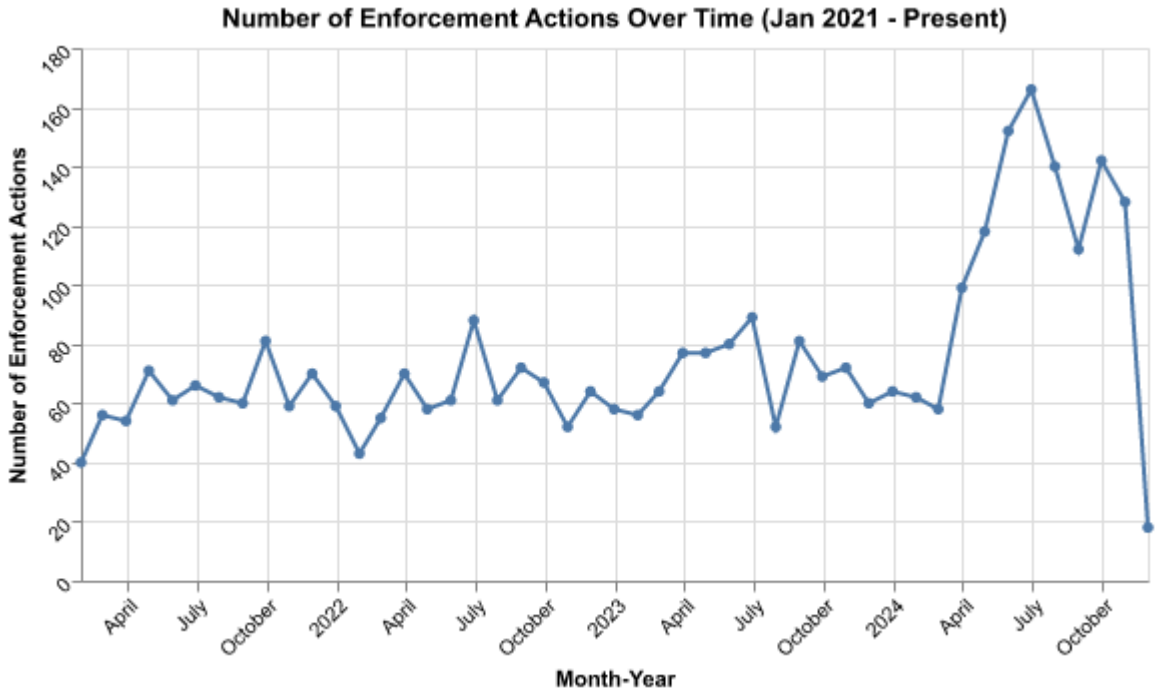
Agency: U.S. Attorney's Office, Middle District of Tennessee

Step 3: Plot data based on scraped data

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

```
# Load the data
data_path = "E:\\UChicago\\[4] 24 Autumn\\Python 2\\Problem set
↪  5\\enforcement_actions_2021_01.csv"
data_2021 = pd.read_csv(data_path)
```

```
# Parse and prepare the date column
data_2021['Date'] = pd.to_datetime(data_2021['Date'], errors='coerce')
data_2021 = data_2021.dropna(subset=['Date'])

# Aggregate the data by month
monthly_data = data_2021.resample('M',
 ↪  on='Date').size().reset_index(name='Enforcement Actions')

# Create the Altair line chart
chart = alt.Chart(monthly_data).mark_line(point=True).encode(
    x=alt.X('Date:T', title='Month-Year'),
    y=alt.Y('Enforcement Actions:Q', title='Number of Enforcement Actions'),
    tooltip=['Date:T', 'Enforcement Actions:Q']
).properties(
    title='Number of Enforcement Actions Over Time (Jan 2021 - Present)',
    width=int(800 * 2/3),
    height=int(400 * 2/3)
).configure_axis(
    labelAngle=-45
)

chart
```

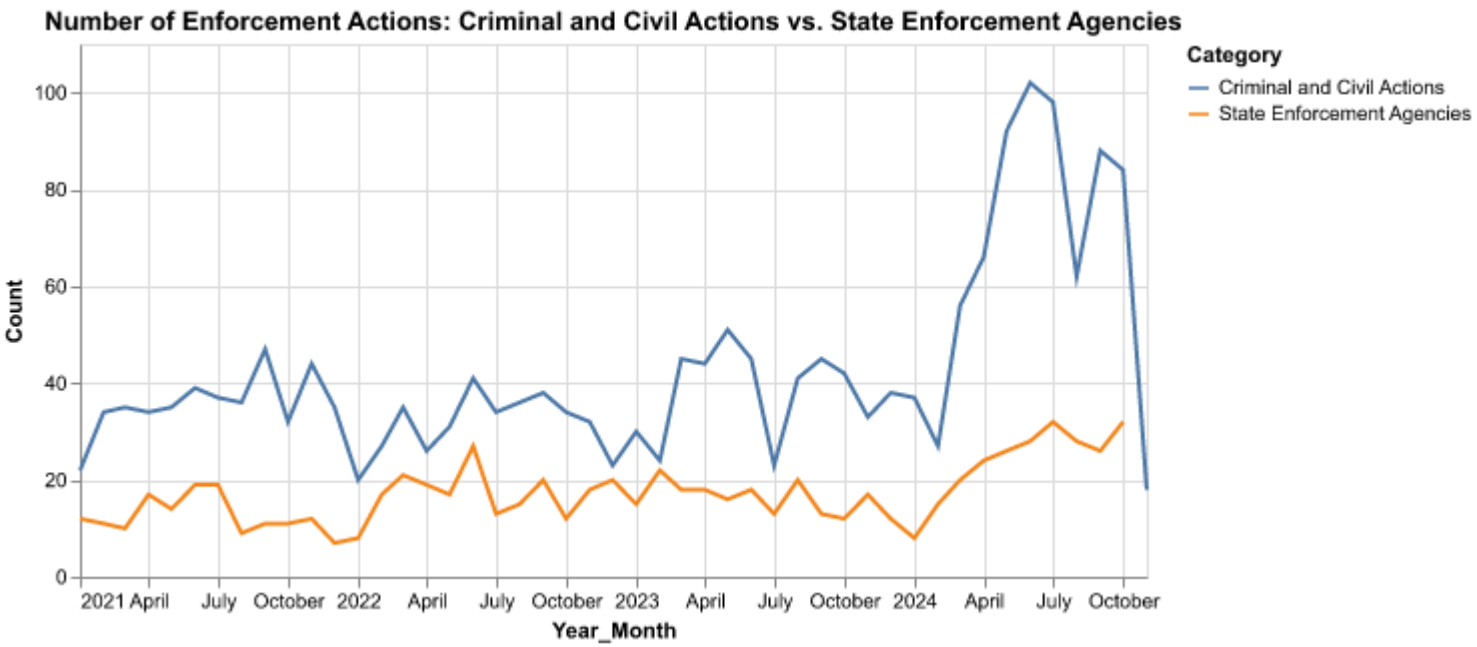


2. Plot the number of enforcement actions categorized: (PARTNER 1)

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

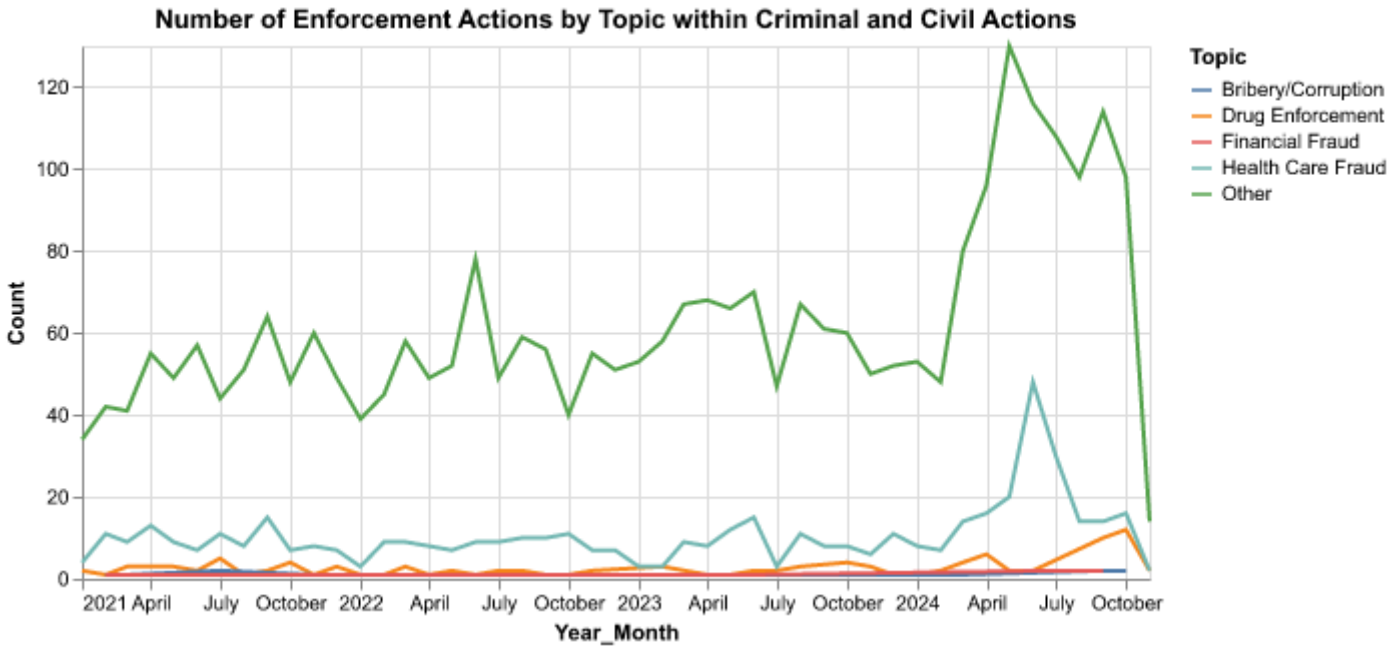
```
# Load the data
data_path = "E:\\UChicago\\[4] 24 Autumn\\Python 2\\Problem set
 ↪  5\\enforcement_actions_2021_01.csv"
data = pd.read_csv(data_path)

# Convert 'Date' column to datetime format without specifying a format
 ↪  (automatic inference)
```

```
data['Date'] = pd.to_datetime(data['Date'], errors='coerce')

# Extract year and month for aggregation
data['Year_Month'] = data['Date'].dt.to_period('M')

# Aggregating data
# Count the number of actions per month for each category
monthly_counts = data.groupby(['Year_Month',
 ↪  'Category']).size().reset_index(name='Count')

# Convert 'Year_Month' back to datetime for plotting
monthly_counts['Year_Month'] = monthly_counts['Year_Month'].dt.to_timestamp()

# Plot: "Criminal and Civil Actions" vs. "State Enforcement Agencies"
chart = alt.Chart(monthly_counts).mark_line().encode(
    x='Year_Month:T',
    y='Count:Q',
    color='Category:N'
).transform_filter(
    (alt.datum.Category == 'Criminal and Civil Actions') |
 ↪  (alt.datum.Category == 'State Enforcement Agencies')
).properties(
    title='Number of Enforcement Actions: Criminal and Civil Actions vs.
 ↪  State Enforcement Agencies',
    width=int(800 * 2/3),
    height=int(400 * 2/3)
)

# Display the chart
chart
```



- based on five topics

```
# Convert 'Date' column to datetime format without specifying a format
 ↪  (automatic inference)
data['Date'] = pd.to_datetime(data['Date'], errors='coerce')

# Extract year and month for aggregation
data['Year_Month'] = data['Date'].dt.to_period('M')

# Define topics within "Criminal and Civil Actions"
def categorize_topic(title):
    title = title.lower()
    if "health care" in title or "health care fraud" in title:
        return "Health Care Fraud"
    elif "financial" in title or "financial fraud" in title:
        return "Financial Fraud"
    elif "drug" in title:
        return "Drug Enforcement"
    elif "bribery" in title or "corruption" in title:
        return "Bribery/Corruption"
    else:
        return "Other"

# Apply topic categorization only to "Criminal and Civil Actions" category
data['Topic'] = data.apply(lambda row: categorize_topic(row['Title']) if
 ↪  row['Category'] == "Criminal and Civil Actions" else "Other", axis=1)

# Aggregating data
# Count the number of actions per month for each topic
monthly_topic_counts = data.groupby(['Year_Month',
 ↪  'Topic']).size().reset_index(name='Count')

# Convert 'Year_Month' back to datetime for plotting
monthly_topic_counts['Year_Month'] =
 ↪  monthly_topic_counts['Year_Month'].dt.to_timestamp()

# Plotting
# Filter for the five specified topics within "Criminal and Civil Actions"
chart = alt.Chart(monthly_topic_counts).mark_line().encode(
    x='Year_Month:T',
    y='Count:Q',
    color='Topic:N'
).transform_filter(
    alt.FieldOneOfPredicate(field='Topic', oneOf=["Health Care Fraud",
 ↪  "Financial Fraud", "Drug Enforcement", "Bribery/Corruption", "Other"])
).properties(
    title='Number of Enforcement Actions by Topic within Criminal and Civil
 ↪  Actions',
    width=int(800 * 2/3),
    height=int(400 * 2/3)
)

# Display the chart
chart
```

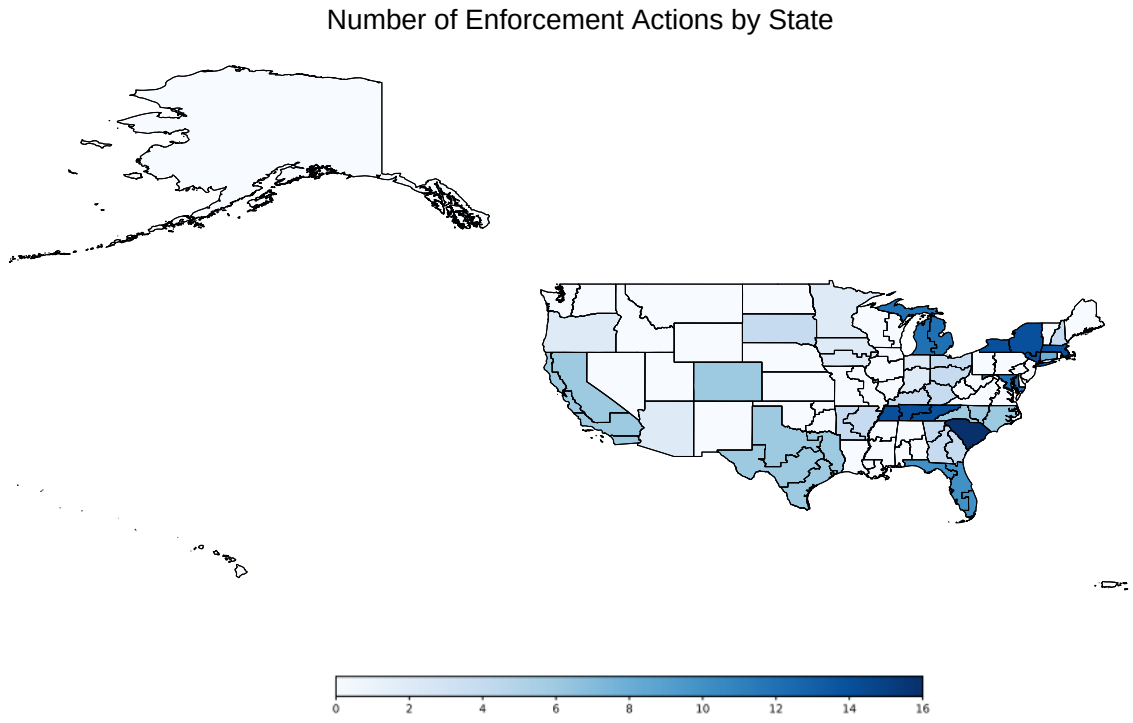


Step 4: Create maps of enforcement activity

1. Map by State (PARTNER 1)

```
# Step 1: Load the enforcement data
data_path = "E:\\UChicago\\[4] 24 Autumn\\Python 2\\Problem set
 ↪  5\\enforcement_actions_2021_01.csv"
enforcement_data = pd.read_csv(data_path)

# Step 2: Filter for state-level agencies and extract state names
state_actions =
 ↪  enforcement_data[enforcement_data['Agency'].str.contains("State of",
 ↪  na=False)]
state_actions['State'] = state_actions['Agency'].str.extract(r"State of
 ↪  ([A-Za-z\s]+)")
# Step 3: Aggregate number of actions by state
state_counts = state_actions['State'].value_counts().reset_index()
state_counts.columns = ['State', 'Action_Count']
# Step 4: Load the JSON data from the API endpoint
shape_url = 'https://data.ojp.usdoj.gov/resource/5fdt-n5ne.json'
response = requests.get(shape_url)
data = response.json()
# Step 5: Convert JSON data to a GeoDataFrame
geometries = [shape(feature["the_geom"]) for feature in data]
properties = [{key: feature[key] for key in feature if key != "the_geom"} for
 ↪  feature in data]
shape_data = gpd.GeoDataFrame(properties, geometry=geometries)
```

```
# Step 6: Merge shape data with state enforcement action counts
state_counts['State'] = state_counts['State'].str.strip()
merged_data = shape_data.merge(state_counts, left_on='state',
 ↪  right_on='State', how='left')
merged_data['Action_Count'] = merged_data['Action_Count'].fillna(0)

# Step 7: Plot the main map for all U.S. states, including Alaska and Hawaii
fig, ax = plt.subplots(1, 1, figsize=(15, 10))

# Plot all states with enhanced styles
merged_data.plot(column='Action_Count', cmap='Blues', linewidth=0.8, ax=ax,
 ↪  edgecolor='black', legend=True,
                 legend_kwds={'shrink': 0.5, 'orientation': "horizontal",
 ↪  'pad': 0.02, 'aspect': 30})
merged_data.boundary.plot(ax=ax, linewidth=0.8, color="black")

# Adjust map extent to include Alaska and Hawaii
ax.set_xlim([-180, -60])  # Extends to cover Alaska and Hawaii
ax.set_ylim([15, 72])     # Adjusted for both Hawaii and Alaska latitudes

# Customize the plot further for better visualization

ax.set_title('Number of Enforcement Actions by State', fontsize=16)
ax.set_axis_off()
plt.tight_layout()

plt.show()
```

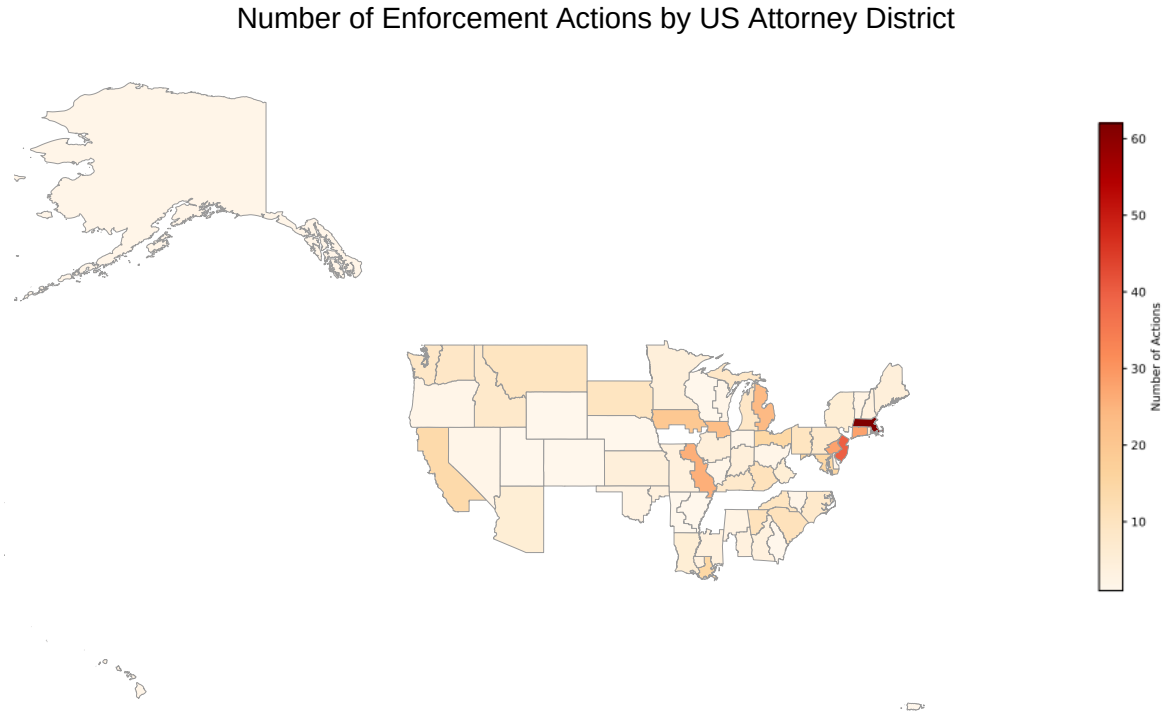


2. Map by District (PARTNER 2)

```
# Step 1: Load datasets
data_path = "E:\\UChicago\\[4] 24 Autumn\\Python 2\\Problem set
 ↪  5\\enforcement_actions_2021_01.csv"
district_shapefile_path = "E:\\UChicago\\[4] 24 Autumn\\Python 2\\Problem set
 ↪  5\\US Attorney Districts Shapefile
 ↪  simplified_20241108\\geo_export_50f98fd1-912c-4dba-9e7d-a91095ae741a.shp"

# Step 1: Load datasets
data_2021 = pd.read_csv(data_path)
gdf = gpd.read_file(district_shapefile_path)

# Step 2: Filter records for U.S. Attorney's Office agencies
us_attorney_data = data_2021[data_2021['Agency'].str.contains("U.S.
 ↪  Attorney's Office", na=False)]

# Step 3: Extract and clean district names
def extract_district_name(agency):
    match = re.search(r"U\.S\. Attorney's Office, (.+)", agency)
    if match:
        return match.group(1).replace("District of ", "").strip().lower()
    return None
us_attorney_data['Cleaned_District'] =
 ↪  us_attorney_data['Agency'].apply(extract_district_name)

# Step 4: Standardize 'judicial_d' column in the shapefile
def clean_judicial_d(judicial_d):
    judicial_d = judicial_d.lower()
    judicial_d = judicial_d.replace("district of ", "")
    judicial_d = judicial_d.replace("northern district of ", "northern ")
    judicial_d = judicial_d.replace("southern district of ", "southern ")
    judicial_d = judicial_d.replace("eastern district of ", "eastern ")
    judicial_d = judicial_d.replace("western district of ", "western ")
    return judicial_d.strip()

gdf['judicial_d'] = gdf['judicial_d'].apply(clean_judicial_d)

# Step 5: Manual mapping for unmatched district names
mapping = {
    "eastern virginia": "virginia-e",
    "western virginia": "virginia-w",
    "northern california": "california-n",
    "central california": "california-c",
    "southern california": "california-s",
    "northern texas": "texas-n",
    "southern texas": "texas-s",
    "western texas": "texas-w",
    "eastern texas": "texas-e",
    "middle florida": "florida-m",
    "southern florida": "florida-s",
    "northern florida": "florida-n",
    "southern new york": "new york-s",
    "eastern new york": "new york-e",
    "western new york": "new york-w",
    "middle tennessee": "tennessee-m",
    "eastern tennessee": "tennessee-e",
```

```
    "western tennessee": "tennessee-w",
    "columbia": "district of columbia",
    "puerto rico": "puerto rico",
    "vermont": "vermont",
    "maine": "maine",
    "new jersey": "new jersey",
    "delaware": "delaware",
    "hawaii": "hawaii",
    "alaska": "alaska",
    "rhode island": "rhode island"
}

us_attorney_data['Cleaned_District'] =
 ↪  us_attorney_data['Cleaned_District'].replace(mapping)

# Step 6: Count the number of enforcement actions per district
district_counts =
 ↪  us_attorney_data.groupby('Cleaned_District').size().reset_index(name='Action_Count')

# Step 7: Merge GeoDataFrame with enforcement action counts
merged_gdf = gdf.merge(district_counts, left_on='judicial_d',
 ↪  right_on='Cleaned_District', how='left')

# Step 8: Check the merge result
print("Number of non-null values in 'Action_Count':",
 ↪  merged_gdf['Action_Count'].notna().sum())
print("Sample of merged data:")
print(merged_gdf[['judicial_d', 'Cleaned_District',
 ↪  'Action_Count']].head(10))

# Step 9: Plot the choropleth map
fig, ax = plt.subplots(1, 1, figsize=(20, 12))

# Plotting the choropleth map
merged_gdf.plot(column='Action_Count', cmap='OrRd', linewidth=0.5, ax=ax,
 ↪  edgecolor='0.6', legend=True,
                legend_kwds={'shrink': 0.6, 'label': "Number of Actions"})

# Customize the plot
ax.set_title('Number of Enforcement Actions by US Attorney District',
 ↪  fontdict={'fontsize': 20})
ax.set_xlim([-170, -50])  # Adjust x-axis range to center the US mainland
ax.set_axis_off()

# Display the plot
plt.show()
```

```
Number of non-null values in 'Action_Count': 65
Sample of merged data:
            judicial_d    Cleaned_District  Action_Count
0     western kentucky    western kentucky           8.0
1     eastern kentucky    eastern kentucky          11.0
2     southern indiana    southern indiana           5.0
3       middle alabama      middle alabama           4.0
4     southern alabama                 NaN           NaN
5     western arkansas    western arkansas           1.0
6     eastern arkansas    eastern arkansas           1.0
7  northern california                 NaN           NaN
8   eastern california  eastern california          14.0
9   central california                 NaN           NaN
```



Extra Credit

1. Merge zip code shapefile with population

```
# File paths
shapefile_path = r"E:\UChicago\[4] 24 Autumn\Python 2\Problem set
 ↪  4\gz_2010_us_860_00_500k\gz_2010_us_860_00_500k.shp"
population_data_path = r"E:\UChicago\[4] 24 Autumn\Python 2\Problem set
 ↪  5\DECENNIALDHC2020.P1_2024-11-09T205546\DECENNIALDHC2020.P1-Data.csv"

# Load the ZIP code shapefile and population data
zip_gdf = gpd.read_file(shapefile_path)
population_df = pd.read_csv(population_data_path, skiprows=1)

# Rename columns and extract ZIP code from population data
population_df.columns = ['Geography', 'Geographic Area Name', 'Total
 ↪  Population', 'Unnamed']
```

```
population_df['ZIP'] = population_df['Geographic Area
 ↪  Name'].str.extract(r'(\d{5})')
```

```
# Convert ZIP codes to string format for consistency
zip_gdf['ZCTA5'] = zip_gdf['ZCTA5'].astype(str)
population_df['ZIP'] = population_df['ZIP'].astype(str)

# Merge the GeoDataFrame with the population data based on ZIP code
merged_gdf = zip_gdf.merge(population_df[['ZIP', 'Total Population']],
 ↪  left_on='ZCTA5', right_on='ZIP', how='left')

# Display the first few rows of the merged data
print("Merged Data Preview:")
print(merged_gdf.head())
```

```
Merged Data Preview:
          GEO_ID  ZCTA5   NAME   LSAD  CENSUSAREA  \
0  8600000US01040  01040  01040  ZCTA5      21.281
1  8600000US01050  01050  01050  ZCTA5      38.329
2  8600000US01053  01053  01053  ZCTA5       5.131
3  8600000US01056  01056  01056  ZCTA5      27.205
4  8600000US01057  01057  01057  ZCTA5      44.907

                                            geometry    ZIP  Total Population
0  POLYGON ((-72.62734 42.16203, -72.62764 42.162...  01040           38238.0
1  POLYGON ((-72.95393 42.34379, -72.95385 42.343...  01050            2467.0
2  POLYGON ((-72.68286 42.37002, -72.68287 42.369...  01053            2031.0
3  POLYGON ((-72.39529 42.18476, -72.39653 42.183...  01056           21002.0
4  MULTIPOLYGON (((-72.39191 42.08066, -72.39077 ...  01057            8152.0
```

2. Conduct spatial join

```
# File path for the district shapefile
district_shapefile_path = r"E:\UChicago\[4] 24 Autumn\Python 2\Problem set
 ↪  5\US Attorney Districts Shapefile
 ↪  simplified_20241108\geo_export_50f98fd1-912c-4dba-9e7d-a91095ae741a.shp"

# Step 1: Load the district shapefile
district_gdf = gpd.read_file(district_shapefile_path)

# Step 2: Ensure the CRS (coordinate reference system) is consistent
if merged_gdf.crs != district_gdf.crs:
    merged_gdf = merged_gdf.to_crs(district_gdf.crs)

# Step 3: Perform the spatial join between merged_gdf (ZIP code with
 ↪  population) and district_gdf
joined_gdf = gpd.sjoin(merged_gdf, district_gdf, how='inner',
 ↪  predicate='within')

# Step 4: Aggregate the total population for each district
population_by_district = joined_gdf.groupby('judicial_d')['Total
 ↪  Population'].sum().reset_index()
```

```python
# Step 5: Print the aggregated population result
pd.set_option('display.max_rows', None)
print("Total Population by District:")
print(population_by_district)
```

```
Total Population by District:
                        judicial_d  Total Population
0        Central District of California    17541476.0
1          Central District of Illinois     1836364.0
2                  District of Alaska       358718.0
3                 District of Arizona     6648991.0
4                District of Colorado     5636000.0
5             District of Connecticut     2586019.0
6                District of Delaware       320940.0
7     District of District of Columbia       330784.0
8                  District of Hawaii       443079.0
9                   District of Idaho     1573196.0
10                 District of Kansas     2507705.0
11                  District of Maine       868251.0
12               District of Maryland     4497571.0
13          District of Massachusetts     5220236.0
14              District of Minnesota     5242993.0
15                District of Montana       948114.0
16               District of Nebraska     1682745.0
17                 District of Nevada     2901681.0
18          District of New Hampshire       967455.0
19             District of New Jersey     7791345.0
20             District of New Mexico     1824625.0
21           District of North Dakota       489788.0
22                 District of Oregon     3565044.0
23            District of Puerto Rico     1966637.0
24           District of Rhode Island       541650.0
25         District of South Carolina     4013772.0
26           District of South Dakota       659867.0
27                  District of Utah     2968949.0
28               District of Vermont       490131.0
29               District of Wyoming       395222.0
30       Eastern District of Arkansas     1297889.0
31      Eastern District of California     7604030.0
32        Eastern District of Kentucky     1642124.0
33       Eastern District of Louisiana     1449042.0
34        Eastern District of Michigan     5767327.0
35        Eastern District of Missouri     2228609.0
36        Eastern District of New York     6278336.0
37  Eastern District of North Carolina     2981408.0
38        Eastern District of Oklahoma       402383.0
39    Eastern District of Pennsylvania     5017598.0
40       Eastern District of Tennessee     2006345.0
41           Eastern District of Texas     2870090.0
42        Eastern District of Virginia     4288385.0
43      Eastern District of Washington     1392086.0
44       Eastern District of Wisconsin     2413334.0
45          Middle District of Alabama       943609.0
46          Middle District of Florida    10146531.0
47          Middle District of Georgia     1380515.0
48       Middle District of Louisiana       755485.0
49  Middle District of North Carolina     2220063.0
50     Middle District of Pennsylvania     2688697.0
51        Middle District of Tennessee     2154906.0
52        Northern District of Alabama     2656122.0
53     Northern District of California     5728689.0
54        Northern District of Florida     1191409.0
55        Northern District of Georgia     6131923.0
56       Northern District of Illinois     7879150.0
57        Northern District of Indiana     1902774.0
58           Northern District of Iowa     1026968.0
59    Northern District of Mississippi       696493.0
60       Northern District of New York     3064946.0
61           Northern District of Ohio     4501996.0
62        Northern District of Oklahoma       767664.0
63          Northern District of Texas     6486306.0
64  Northern District of West Virginia       447924.0
65        Southern District of Alabama       385301.0
66     Southern District of California     2799591.0
67        Southern District of Florida     5702674.0
68        Southern District of Georgia     1094934.0
69       Southern District of Illinois       945507.0
70        Southern District of Indiana     3467235.0
71           Southern District of Iowa     1348551.0
72    Southern District of Mississippi     1244332.0
73       Southern District of New York     4772460.0
74           Southern District of Ohio     5484720.0
75          Southern District of Texas     8062133.0
76  Southern District of West Virginia       673048.0
77        Western District of Arkansas       862990.0
78        Western District of Kentucky     1551786.0
79       Western District of Louisiana     1865340.0
80        Western District of Michigan     2398302.0
81        Western District of Missouri     2802071.0
82        Western District of New York     2335397.0
83  Western District of North Carolina     2163712.0
84        Western District of Oklahoma     1954610.0
85    Western District of Pennsylvania     3395674.0
86       Western District of Tennessee     1207114.0
87           Western District of Texas     6840310.0
88        Western District of Virginia     1831829.0
89      Western District of Washington     4010453.0
90       Western District of Wisconsin     1944373.0
```

3. Map the action ratio in each district

```
# Step 1: Load the enforcement data and filter for actions since January 2021
enforcement_data_path = "E:\\UChicago\\[4] 24 Autumn\\Python 2\\Problem set
 ↪  5\\enforcement_actions_2021_01.csv"
enforcement_data = pd.read_csv(enforcement_data_path)

# Convert 'Date' column to datetime format
enforcement_data['Date'] = pd.to_datetime(enforcement_data['Date'],
 ↪  errors='coerce')

# Filter for actions since January 2021
enforcement_data = enforcement_data[enforcement_data['Date'] >= '2021-01-01']

# Step 2: Aggregate the number of enforcement actions by judicial district
district_enforcement_counts =
 ↪  enforcement_data['Agency'].value_counts().reset_index()
district_enforcement_counts.columns = ['District', 'Action_Count']

# Step 3: Merge enforcement actions with the population data by district
district_population_data =
 ↪  population_by_district.merge(district_enforcement_counts,
 ↪  left_on='judicial_d', right_on='District', how='left')
district_population_data['Action_Count'] =
 ↪  district_population_data['Action_Count'].fillna(0)  # Replace NaN with 0

# Step 4: Calculate enforcement actions per capita
district_population_data['Actions_Per_Capita'] =
 ↪  district_population_data['Action_Count'] /
 ↪  district_population_data['Total Population']

# Step 5: Merge with the district shapefile for plotting
district_shapefile_path = "E:\\UChicago\\[4] 24 Autumn\\Python 2\\Problem set
 ↪  5\\US Attorney Districts Shapefile
 ↪  simplified_20241108\\geo_export_50f98fd1-912c-4dba-9e7d-a91095ae741a.shp"
district_gdf = gpd.read_file(district_shapefile_path)

# Merge the per capita data with the district GeoDataFrame
district_map_data = district_gdf.merge(district_population_data,
 ↪  left_on='judicial_d', right_on='judicial_d', how='left')
```

```
# Step 6: Plot the map with adjusted layout and size
fig, ax = plt.subplots(1, 1, figsize=(15, 10))  # Adjust figure size

# Plot data with adjusted limits to center map and include Alaska and Hawaii
district_map_data.plot(column='Actions_Per_Capita', cmap='OrRd',
 ↪  linewidth=0.5, ax=ax, edgecolor='0.7', legend=True,
                        legend_kwds={'label': "Enforcement Actions Per
 ↪  Capita", 'orientation': "horizontal", 'shrink': 0.7})

# Set xlim and ylim to focus on main U.S. area while keeping Alaska and
 ↪  Hawaii
ax.set_xlim([-180, -50])
ax.set_ylim([15, 75])

# Add title and remove axis
ax.set_title("Enforcement Actions Per Capita by US Attorney District",
 ↪  fontsize=16)
ax.set_axis_off()

plt.tight_layout()
plt.show()
```