PS5

Luyao Guo, Ruyu Zhang

2024-11-09

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): luyao Guo, luyao1
    - Partner 2 (name and cnet ID): Ruyu Zhang, ruyuzhang
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: ***R.Z L.G***
5. "I have uploaded the names of anyone else other than my partner and I worked with on the problem set **here**" (1 point)
6. Late coins used this pset: **0** Late coins left after submission: **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')
```

```
import requests
from bs4 import BeautifulSoup
from datetime import datetime
```

Step 1: Develop initial scraper and crawler

1. Scraping (PARTNER 1)

```
# Set up the URL parser and initialize lists
page_url = 'https://oig.hhs.gov/fraud/enforcement/'
page_response = requests.get(page_url)
soup = BeautifulSoup(page_response.text, 'html.parser')
category_data = []
title_data = []
date_data = []
url_links = []

# Locate the main content sections
entries = soup.find_all(
    'li', class_='usa-card card--list pep-card--minimal mobile:grid-col-12')

# Loop through each entry to collect title, date, category, and link
for entry in entries:
    title_block = entry.find('h2', class_='usa-card__heading')
    title = title_block.get_text(
        strip=True) if title_block else 'Title Not Available'

    link_tag = title_block.find('a') if title_block else None
    link_url = f"https://oig.hhs.gov{link_tag['href']
                                     }" if link_tag else 'Link Not Available'
```

```
    date_block = entry.find('span', class_='text-base-dark
↪  padding-right-105')
    date = date_block.get_text(
        strip=True) if date_block else 'Date Not Available'

    category_block = entry.find(
        'li', class_='display-inline-block usa-tag text-no-lowercase
         ↪  text-base-darkest bg-base-lightest margin-right-1')
    category = category_block.get_text(
        strip=True) if category_block else 'Category Not Available'

    # Append data to respective lists
    category_data.append(category)
    title_data.append(title)
    date_data.append(date)
    url_links.append(link_url)

# save data to DataFrame & display
enforcement_data = pd.DataFrame({
    'Enforcement Category': category_data,
    'Enforcement Title': title_data,
    'Enforcement Date': date_data,
    'Enforcement Link': url_links
})
print("Scraped DataFrame:")
print(enforcement_data.head())
```

```
Scraped DataFrame:
         Enforcement 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

                                  Enforcement Title  Enforcement 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
```

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

2. Crawling (PARTNER 1)

```python
# Initialize list to store agency names
agency_names = []

# Function to extract agency name from each individual page

def get_agency_name(link):
    response = requests.get(link)
    page_soup = BeautifulSoup(response.text, 'html.parser')
    agency_info = 'Agency Not Found'

    try:
        # Locate the "Agency:" span
        agency_label = page_soup.find(
            'span', class_='padding-right-2 text-base', string="Agency:")
        if agency_label:
            # If found, get the next sibling text (which is likely the agency
            ↪ name)
            agency_info = agency_label.find_next_sibling(text=True).strip()

    except Exception as e:
        print(f"Error occurred for link {link}: {e}")

    return agency_info


# Crawl each link in the DataFrame and get the agency name
for link in enforcement_data['Enforcement Link']:
    agency_name = get_agency_name(link)
    agency_names.append(agency_name)
    time.sleep(1)
```

```
# Add the agency names to the DataFrame &updata data frame
enforcement_data['Agency Name'] = agency_names
print("Updated DataFrame with Agency Information:")
print(enforcement_data.head())
```

```
Updated DataFrame with Agency Information:
         Enforcement 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

                                    Enforcement Title  Enforcement 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

                                     Enforcement 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 Name
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

- a. Pseudo-Code (PARTNER 2)

Input Validation: Check if the input year is 2013 or later. If the year is less than 2013, print a message and exit the function.

Initialization: Define the base URL for the enforcement actions page. Initialize lists to store data for titles, dates, categories, links, and agencies. Set the initial page number to 1. Initialize a variable keep_scraping as True to control the main loop.

Main Loop (While Loop): Loop Definition: A while loop that continues as long as keep_scraping is True. Purpose: This loop will go through each page of the enforcement actions, scraping data until there are no more entries or until the specified date range is exceeded. Inside the While Loop: Construct Page URL: Append the page number to the base URL to form the full URL for the current page. Send HTTP Request: Use requests.get() to retrieve the page content. Parse the HTML content with BeautifulSoup. Locate Enforcement Entries: Search for the main content section containing all enforcement actions on the page. If no entries are found, set keep_scraping to False and break out of the loop.

Inner Loop (For Loop): Loop Definition: A for loop that iterates over each enforcement action found in the current page's entries. Purpose: This loop extracts the details (title, date, category, link, and agency name) for each enforcement action. Inside the For Loop: Extract Date: Retrieve and parse the date of the enforcement action. If the date is older than the specified starting month and year, set keep_scraping to False and break out of the loop. Extract Title, Link, and Category: Retrieve and store the title, link, and category of the enforcement action. Extract Agency Name: Call the get_agency_name function with the link to fetch the agency name from the detailed page. Append Data to Lists: Add the extracted data to the corresponding lists (titles, dates, categories, links, agencies). End of Inner For Loop: After processing each entry, increment the page number by 1 to move to the next page and add a 1-second delay to prevent server overload.

Create DataFrame: Once the loop completes, create a DataFrame from the lists containing enforcement data.

Save Data to CSV: Save the DataFrame as a CSV file named based on the input year and month. Print a confirmation message with the file name.

Function Call Example: Call scrape_enforcement_actions with specific year and month values, such as scrape_enforcement_actions(2023, 1).

- b. Create Dynamic Scraper (PARTNER 2)

```
def scrape_enforcement_actions(year, month):
    if year < 2013:
        print("The year should be restricted to 2013 or later.")
        return

    # Set the URL and initialize lists to store data
    base_url = 'https://oig.hhs.gov/fraud/enforcement/'
    titles, dates, categories, links, agencies = [], [], [], [], []
```

```
    page = 1
    keep_scraping = True

    while keep_scraping:
        # Construct the URL with pagination
        url = f"{base_url}?page={page}"
        response = requests.get(url)
        soup = BeautifulSoup(response.text, 'html.parser')

        # Locate the main content sections for each enforcement action
        entries = soup.find_all(
            'li', class_='usa-card card--list pep-card--minimal
             ↪  mobile:grid-col-12')

        if not entries:  # Stop if no entries are found on the page
            break

        for entry in entries:
            # Extract the date and stop if we exceed the specified date range
            date_block = entry.find(
                'span', class_='text-base-dark padding-right-105')
            date_text = date_block.get_text(strip=True)
            entry_date = datetime.strptime(date_text, '%B %d, %Y')
            if entry_date.year < year or (entry_date.year == year and
             ↪  entry_date.month < month):
                keep_scraping = False
                break

            # Extract other details if within the date range
            title_block = entry.find('h2', class_='usa-card__heading')
            title = title_block.get_text(strip=True)

            link_tag = title_block.find('a')
            link_url = f"https://oig.hhs.gov{link_tag['href']}"

            category_block = entry.find(
                'li', class_='display-inline-block usa-tag text-no-lowercase
                 ↪  text-base-darkest bg-base-lightest margin-right-1')
            category = category_block.get_text(strip=True)

            # Get agency name from the linked page
            agency_name = get_agency_name(link_url)
```

```
            # Append data to lists
            dates.append(date_text)
            titles.append(title)
            links.append(link_url)
            categories.append(category)
            agencies.append(agency_name)

        # Increment page number and wait before the next request
        page += 1
        time.sleep(1)

    # Create and save DataFrame
    df_enforcement = pd.DataFrame({
        'Enforcement Date': dates,
        'Enforcement Title': titles,
        'Enforcement Category': categories,
        'Enforcement Link': links,
        'Agency Name': agencies
    })

    # Save to CSV
    file_name = f"enforcement_actions_{year}_{month}.csv"
    df_enforcement.to_csv(file_name, index=False)
    print(f"Data saved to {file_name}")


scrape_enforcement_actions(2023, 1)
```

There are 1534 enforcement actions. Date and Details of the Earliest Enforcement Action: Date: January 3, 2023 Title: Podiatrist Pays $90,000 To Settle False Billing Allegations Category: Criminal and Civil Actions Link: https://oig.hhs.gov/fraud/enforcement/podiatrist-pays-90000-to-settle-false-billing-allegations/ Agency Involved: U.S. Attorney's Office, Southern District of Texas

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

```
scrape_enforcement_actions(2021, 1)
```

```
df_enforcement = pd.read_csv('enforcement_actions_2021_1.csv')
```

```
print(f"Total number of enforcement actions: {len(df_enforcement)}")
print("Earliest enforcement action:")
print(df_enforcement.iloc[-1])
```

```
Total number of enforcement actions: 3022
Earliest enforcement action:
Enforcement Date                                          January 4, 2021
Enforcement Title       The United States And Tennessee Resolve Claims...
Enforcement Category                             Criminal and Civil Actions
Enforcement Link        https://oig.hhs.gov/fraud/enforcement/the-unit...
Agency Name             U.S. Attorney's Office, Middle District of Ten...
Name: 3021, dtype: object
```

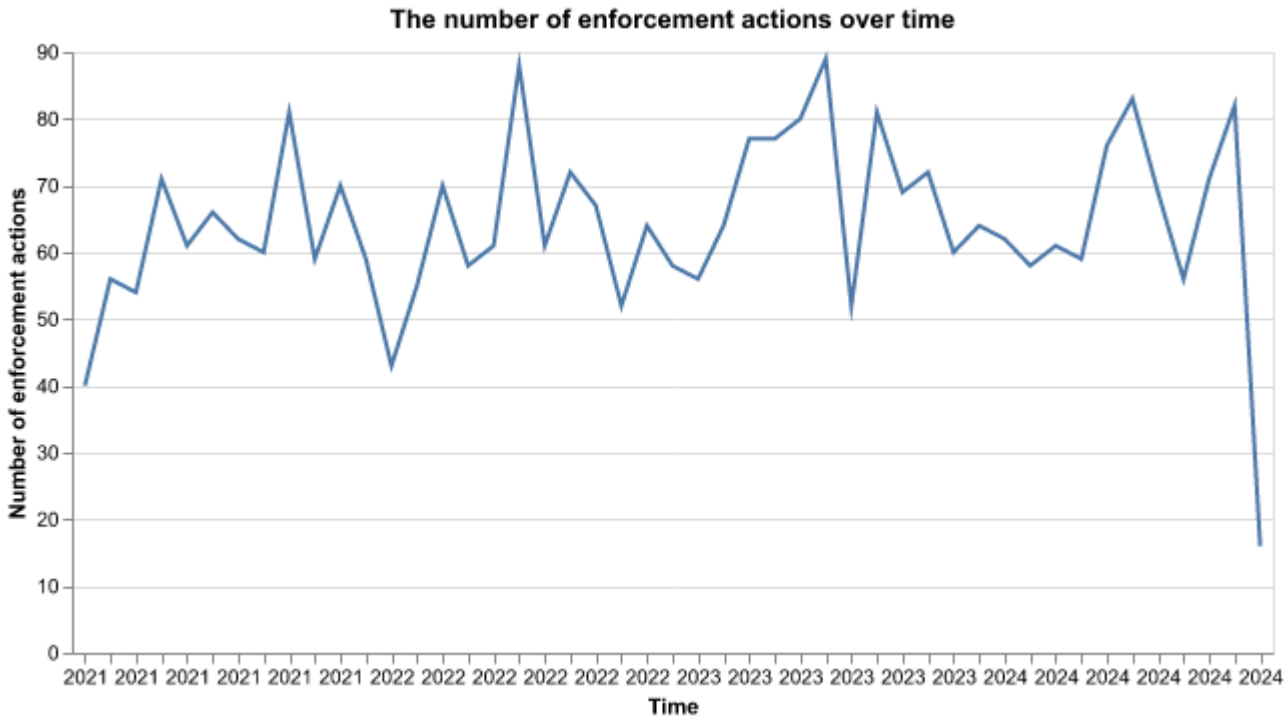
Total Number of Enforcement Actions: A total of 3022 enforcement actions were collected since January 2021. Date and Details of the Earliest Enforcement Action: Date: January 4, 2021 Title: The United States and Tennessee Resolve Claims Involving Medicaid Managed Care Fraud Scheme Category: Criminal and Civil Actions Link: https://oig.hhs.gov/fraud/enforcement/the-united-states-and-tennessee-resolve-claims-involving-medicaid-managed-care-fraud-scheme Agency Involved: 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)

```
df_enforcement_2021 = pd.read_csv('enforcement_actions_2021_1.csv')
df_enforcement_2021['Enforcement Date'] = pd.to_datetime(
    df_enforcement_2021['Enforcement Date'], errors='coerce')
df_enforcement_2021['year_month'] = df_enforcement_2021['Enforcement
 ↪  Date'].dt.to_period(
    'M')
df_enforcement_2021_monthly = df_enforcement_2021.groupby(
    ['year_month']).size().reset_index(name='num_enforcement')
df_enforcement_2021_monthly['year_month'] =
 ↪  df_enforcement_2021_monthly['year_month'].dt.to_timestamp()

line_chart_2021 = alt.Chart(df_enforcement_2021_monthly, title="The number of
 ↪  enforcement actions over time").mark_line().encode(
    alt.X("yearmonth(year_month):O", title="Time")
    .axis(
```

```
        format="%Y",
        labelAngle=0,
    ),
    alt.Y('num_enforcement:Q', title='Number of enforcement actions'),
    tooltip=['year_month', 'num_enforcement']
).properties(
    width=600,
    height=300
)

line_chart_2021
```



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

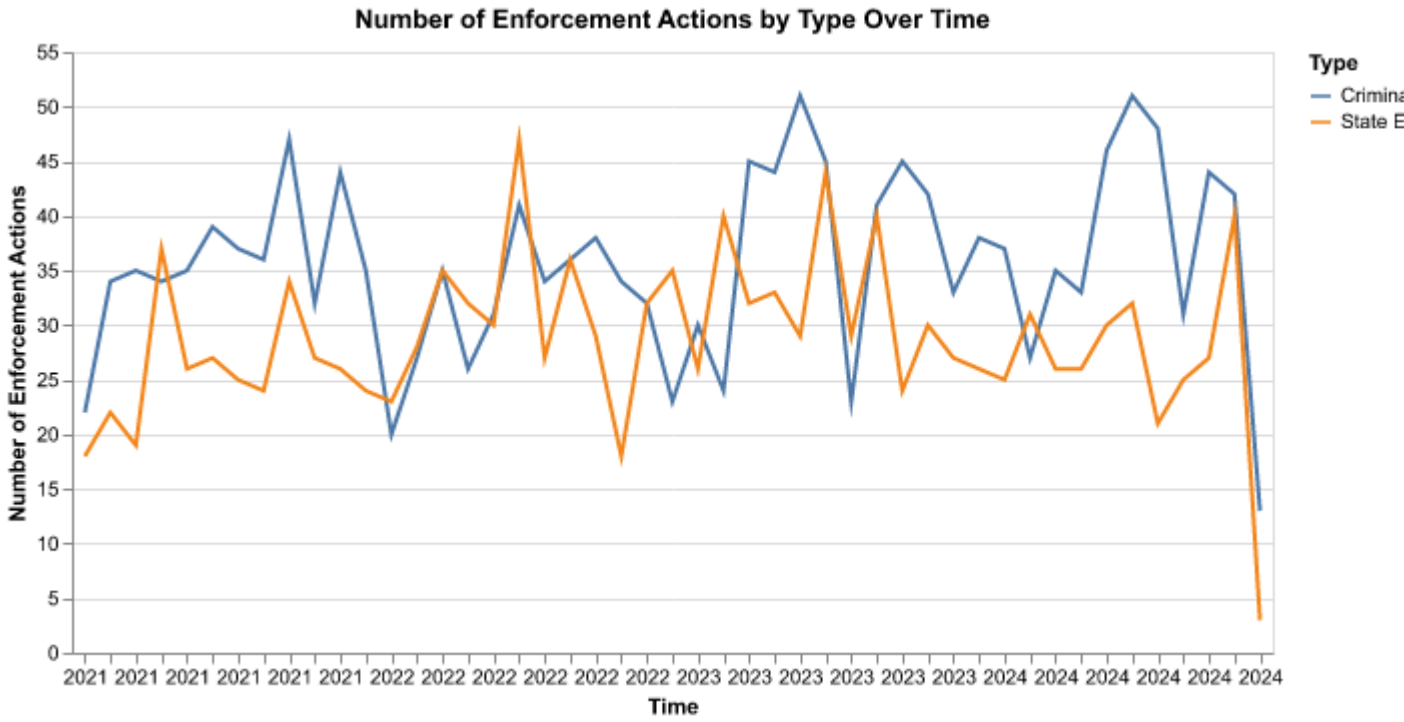
```
df_enforcement_2021['Type'] = df_enforcement_2021['Enforcement
 ↪  Category'].apply(
    lambda x: 'Criminal and Civil Actions' if 'Criminal and Civil' in x else
     ↪  'State Enforcement Agencies'
)
```

```
def assign_topic(title):
    if 'health' in title.lower() or 'medicare' in title.lower() or 'medicaid'
     ↪  in title.lower():
        return 'Health Care Fraud'
    elif 'financial' in title.lower() or 'bank' in title.lower():
        return 'Financial Fraud'
    elif 'drug' in title.lower():
        return 'Drug Enforcement'
    elif 'bribery' in title.lower() or 'corruption' in title.lower():
        return 'Bribery/Corruption'
    else:
        return 'Other'


df_enforcement_2021['Topic'] = df_enforcement_2021.apply(
    lambda row: assign_topic(
        row['Enforcement Title']) if row['Type'] == 'Criminal and Civil
 ↪  Actions' else None,
    axis=1
)
```

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

```
# Plot the line chart for "Criminal and Civil Actions" vs. "State Enforcement
 ↪  Agencies"
df_type_monthly = df_enforcement_2021.groupby(
    ['year_month', 'Type']).size().reset_index(name='num_enforcement')
df_type_monthly['year_month'] =
 ↪  df_type_monthly['year_month'].dt.to_timestamp()

type_chart = alt.Chart(df_type_monthly, title="Number of Enforcement Actions
 ↪  by Type Over Time").mark_line().encode(
    alt.X("yearmonth(year_month):O", title="Time")
    .axis(format="%Y", labelAngle=0),
    alt.Y('num_enforcement:Q', title='Number of Enforcement Actions'),
    color='Type:N',
    tooltip=['year_month', 'num_enforcement', 'Type']
).properties(
    width=600,
    height=300
)
```

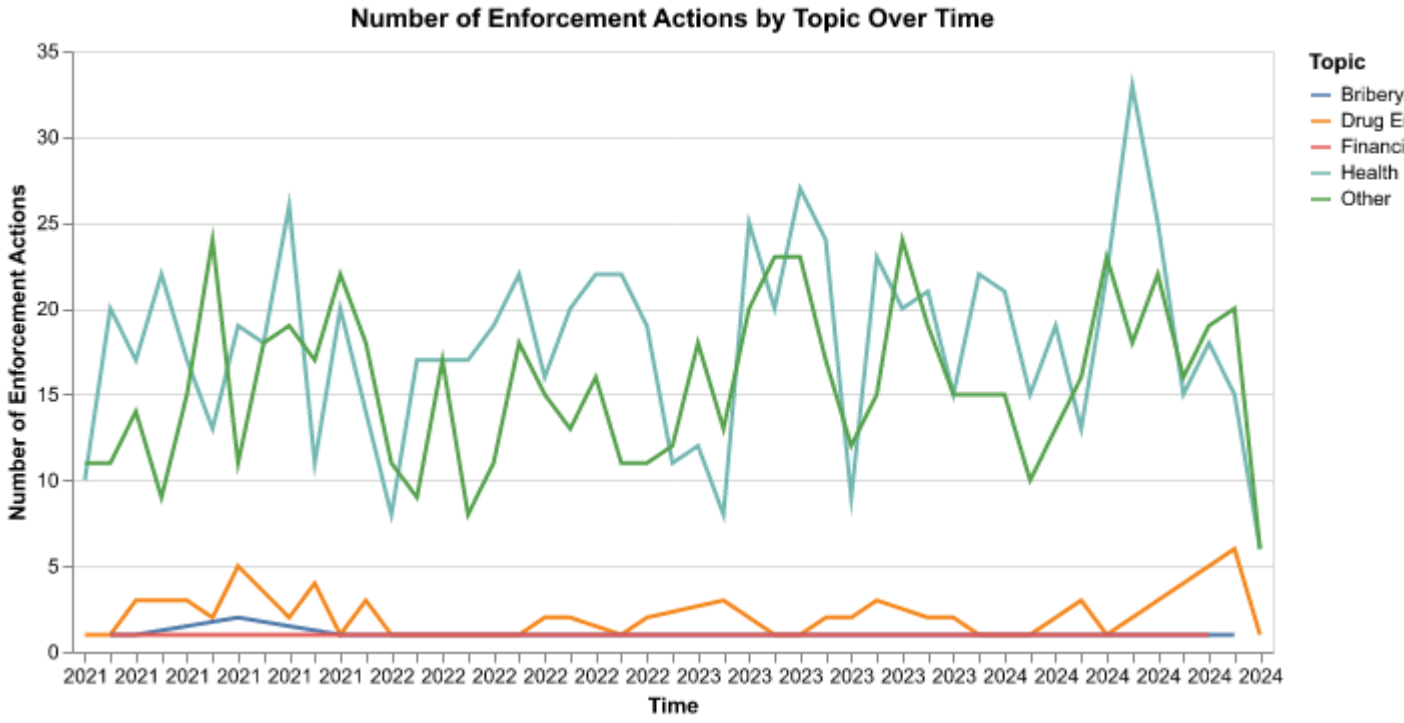
```
type_chart.display()
```



- based on five topics

```
# Plot the line chart for topics within "Criminal and Civil Actions"
df_topic_monthly = df_enforcement_2021[df_enforcement_2021['Type']
                                       == 'Criminal and Civil Actions']
df_topic_monthly = df_topic_monthly.groupby(
    ['year_month', 'Topic']).size().reset_index(name='num_enforcement')
df_topic_monthly['year_month'] =
 ↪  df_topic_monthly['year_month'].dt.to_timestamp()

topic_chart = alt.Chart(df_topic_monthly, title="Number of Enforcement
 ↪  Actions by Topic Over Time").mark_line().encode(
    alt.X("yearmonth(year_month):O", title="Time")
    .axis(format="%Y", labelAngle=0),
    alt.Y('num_enforcement:Q', title='Number of Enforcement Actions'),
    color='Topic:N',
    tooltip=['year_month', 'num_enforcement', 'Topic']
```

```
).properties(
    width=600,
    height=300
)

topic_chart.display()
```

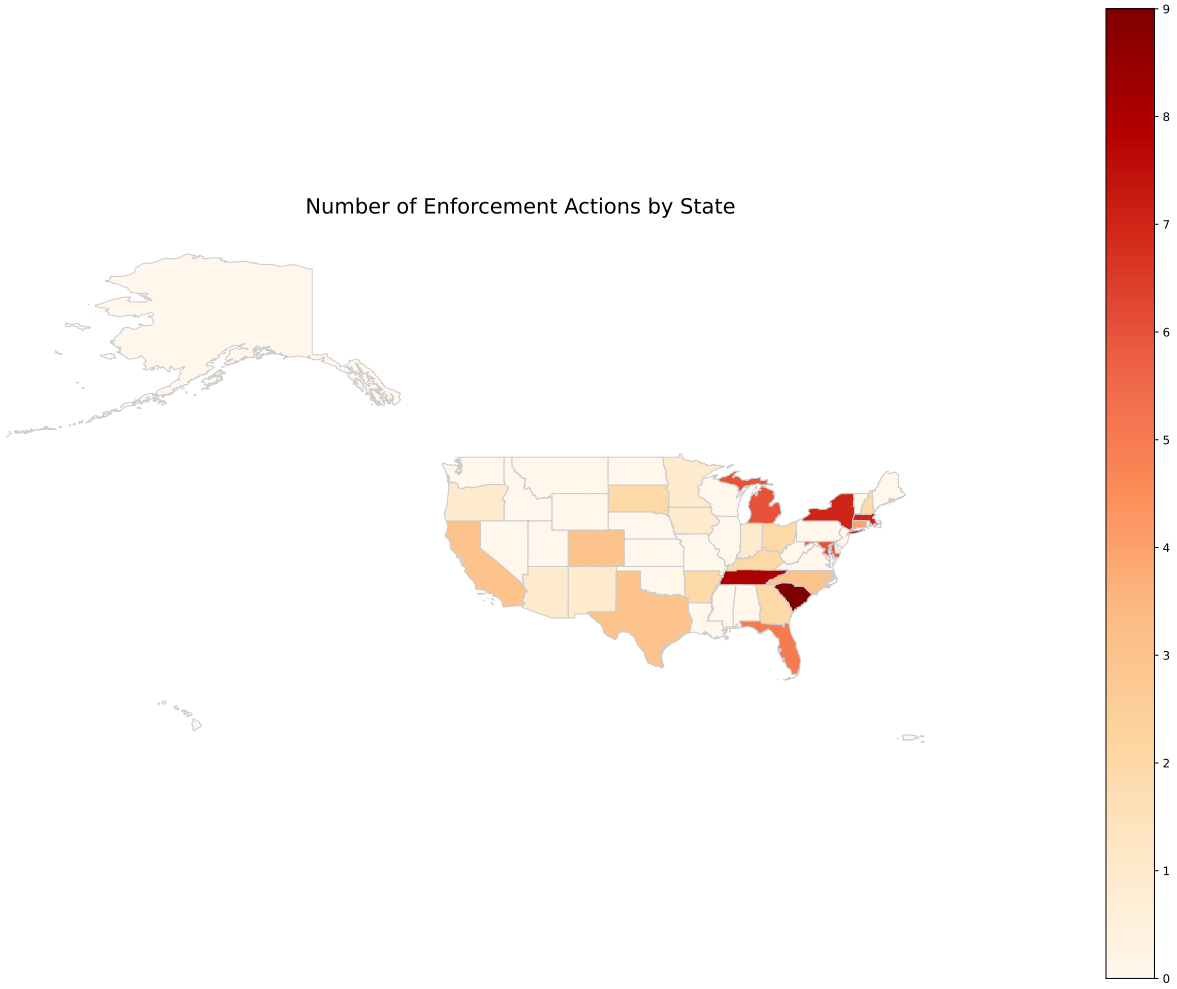


Step 4: Create maps of enforcement activity

1. Map by State (PARTNER 1)

```
import geopandas as gpd
import matplotlib.pyplot as plt

state_actions =  df_enforcement_2021[ df_enforcement_2021['Agency
 ↪  Name'].str.contains("State of", case=False, na=False)]

# Extract state names from "Agency Name" for accurate grouping
```

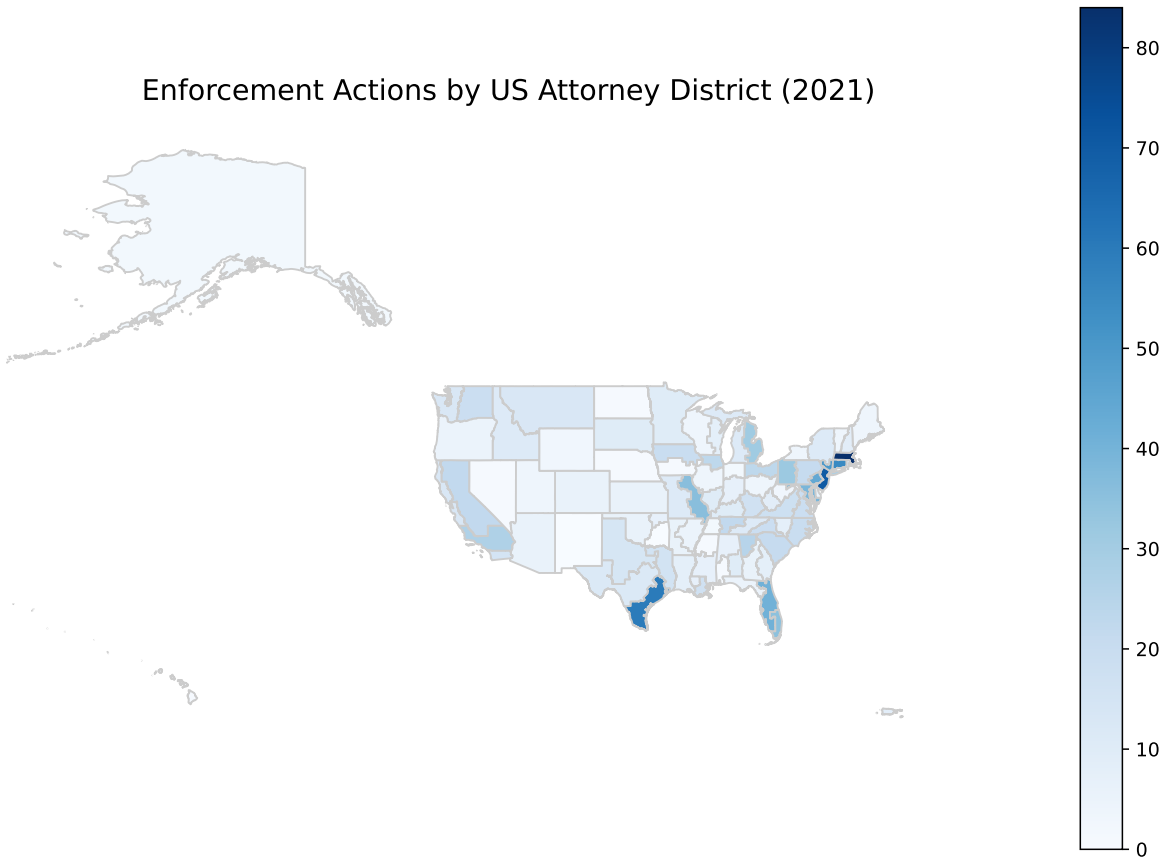
```
state_actions['State'] = state_actions['Agency Name'].str.replace("State of
 ↪  ", "", case=False, regex=True).str.strip()

# Count the number of enforcement actions per state
state_counts = state_actions['State'].value_counts().reset_index()
state_counts.columns = ['State', 'Enforcement Count']

# Load the US Census state shapefile
# Make sure to replace with your actual shapefile path
us_states = gpd.read_file('cb_2018_us_state_5m.shp')

# Merge state counts with the shapefile
us_states = us_states.merge(state_counts, how="left", left_on="NAME",
 ↪  right_on="State")
us_states['Enforcement Count'] = us_states['Enforcement Count'].fillna(0)

# Plot the map
fig, ax = plt.subplots(1, 1, figsize=(20, 15))
us_states.plot(column='Enforcement Count', cmap='OrRd', linewidth=0.8, ax=ax,
 ↪  edgecolor='0.8', legend=True)
ax.set_title('Number of Enforcement Actions by State', fontdict={'fontsize':
 ↪  '18', 'fontweight': '3'})

# Set map boundaries to display the US clearly
ax.set_xlim([-180, -50])
ax.set_ylim([15, 75])
ax.set_axis_off()

plt.show()
```



2. Map by District (PARTNER 2)

```
# Load enforcement actions data for 2021 and filter by district
df_enforcement_2021_district =
 ↪  df_enforcement_2021[df_enforcement_2021['Agency Name'].str.contains(
    "District", case=False, na=False)]

# Extract district names from the "Agency Name" column
df_enforcement_2021_district['District'] =
 ↪  df_enforcement_2021_district['Agency Name'].str.extract(
    r'U\.S\. Attorney['\'s]* Office, (.+)')[0]
```

```
# Group by district and count the number of enforcement actions per district
df_districts_count = df_enforcement_2021_district.groupby(
    ['District']).size().reset_index(name='Enforcement_count')

# Load the US Attorney District shapefile
gdf_districts = gpd.read_file(
    'geo_export_be204818-01a4-4982-a890-81531d8a7432.shp')
# Merge enforcement counts with the shapefile
map_districts = gdf_districts.merge(
    df_districts_count, left_on='judicial_d', right_on='District',
 ↪  how='left')
map_districts['Enforcement_count'] =
 ↪  map_districts['Enforcement_count'].fillna(
    0)

# plot a choropleth
fig, ax = plt.subplots(1, 1, figsize=(12, 8))
map_districts.plot(column='Enforcement_count', ax=ax,
                   cmap='Blues', edgecolor='0.8', legend=True).set_axis_off()

ax.set_xlim([-180, -50])
ax.set_ylim([15, 75])
plt.title("Enforcement Actions by US Attorney District (2021)", fontsize=15)

plt.show()
```



Extra Credit

1. Merge zip code shapefile with population

```
# Load dataframes
zip_gdf = gpd.read_file('gz_2010_us_860_00_500k.shp')
population_df = pd.read_csv('DECENNIALDHC2020.P1-Data.csv')
```

```
# Clean population data
population_df['ZCTA5'] = population_df['NAME'].str.extract(r"ZCTA5 (\d+)")

# Merge ZIP Code shapefile with population data on the ZIP Code column
zip_map = zip_gdf.merge(population_df, on='ZCTA5', how='left')

zip_map.head()
```

	GEO_ID_x	ZCTA5	NAME_x	LSAD	CENSUSAREA	geometry
0	8600000US01040	01040	01040	ZCTA5	21.281	POLYGON ((-72.62734 42.16203, -72
1	8600000US01050	01050	01050	ZCTA5	38.329	POLYGON ((-72.95393 42.34379, -72
2	8600000US01053	01053	01053	ZCTA5	5.131	POLYGON ((-72.68286 42.37002, -72
3	8600000US01056	01056	01056	ZCTA5	27.205	POLYGON ((-72.39529 42.18476, -72
4	8600000US01057	01057	01057	ZCTA5	44.907	MULTIPOLYGON (((-72.39191 42.08

2. Conduct spatial join

```
# Perform spatial join between ZIP Code shapefile and District shapefile
zip_district_join = gpd.sjoin(
    zip_map, gdf_districts, how="inner", predicate="intersects")
zip_district_join.head()

# Aggregate population to get total population in each district
district_population = zip_district_join.groupby(
    'ZCTA5')['P1_001N'].sum().reset_index()
district_population.columns = ['District', 'Total Population']

district_population.head()
```

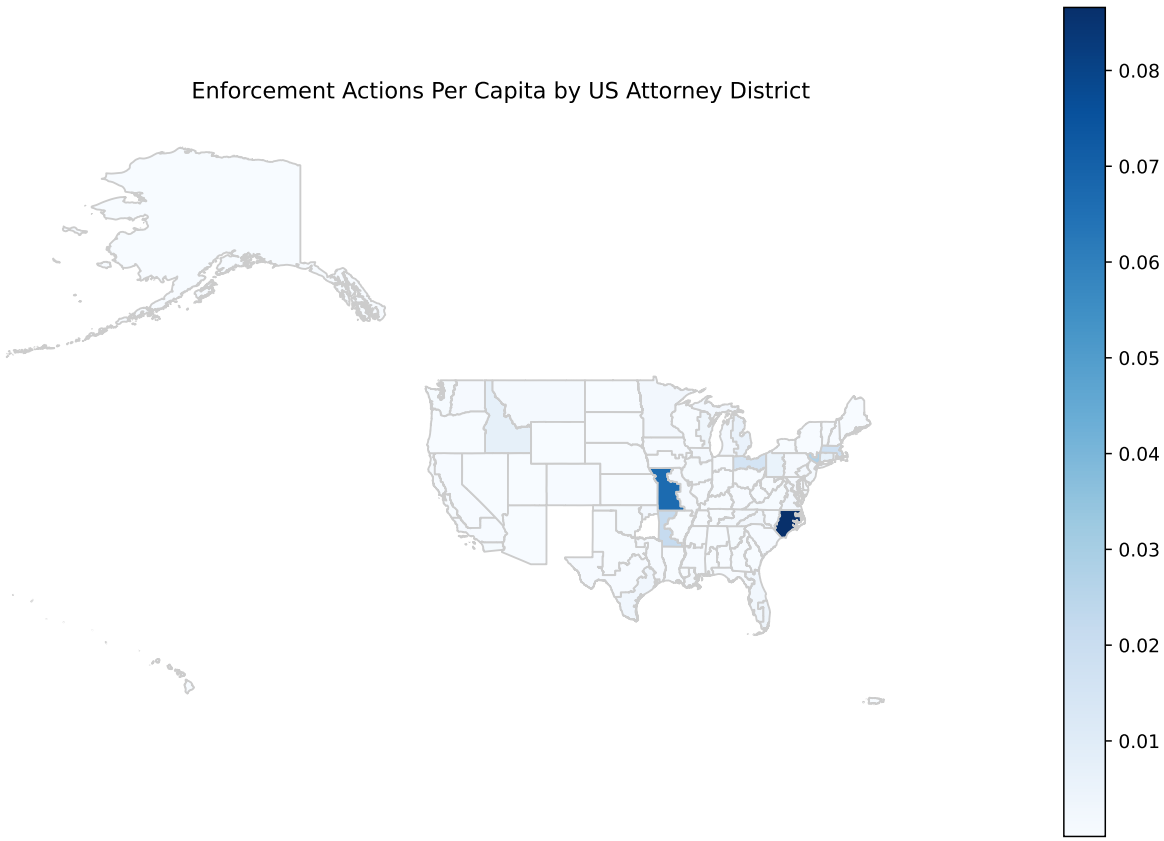
	District	Total Population
0	00601	17242
1	00602	37548
2	00603	49804
3	00606	5009
4	00610	25731

3. Map the action ratio in each district

```
# Convert district_population['Total Population'] from string to numeric
district_population['Total Population'] = pd.to_numeric(
    district_population['Total Population'], errors='coerce')

# Calculate the ratio of enforcement actions per capita
```

```
df_districts_count['Actions Per Capita'] =
 ↪  df_districts_count['Enforcement_count'] / \
    district_population['Total Population']
```

```
# Merge with district GeoDataFrame for mapping
district_map_gdf = gdf_districts.merge(
    df_districts_count, left_on='judicial_d', right_on='District',
 ↪  how='left')

# Plot the map
fig, ax = plt.subplots(1, 1, figsize=(12, 8))
district_map_gdf.plot(column='Actions Per Capita', cmap='Blues',
                      ax=ax, edgecolor='0.8', legend=True).set_axis_off()
ax.set_xlim([-180, -50])
ax.set_ylim([15, 75])
ax.set_title("Enforcement Actions Per Capita by US Attorney District")

plt.show()
```


Enforcement Actions Per Capita by US Attorney District
0.08
0.07
0.06
0.05
0.04
0.03
0.02
0.01