PS5

Ben Zhang, Xinyi Chen

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): Ben Zhang, bzhang369
    - Partner 2 (name and cnet ID): Xinyi Chen, chen61
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: XC, BZ
5. "I have uploaded the names of anyone else other than my partner and I worked with on the problem set **here**" : XC, BZ(1 point)
6. Late coins used this pset: 0 Late coins left after submission: 4
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 requests
from bs4 import BeautifulSoup

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

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

Step 1: Develop initial scraper and crawler

1. Scraping (PARTNER 1)

```
url = "https://oig.hhs.gov/fraud/enforcement/"

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

enforcement_actions = soup.find_all('div', class_='usa-card__container')

titles = []
dates = []
categories = []
links = []

for action in enforcement_actions:

    title_tag = action.find('h2', class_='usa-card__heading').find('a')
    title = title_tag.text.strip()
    link = "https://oig.hhs.gov" + title_tag['href']

    date = action.find(
        'span', class_='text-base-dark padding-right-105').text.strip()

    category_tag = action.find('ul', class_='display-inline add-list-reset')
    category = category_tag.find('li').text.strip()
```

```
    titles.append(title)
    dates.append(date)
    categories.append(category)
    links.append(link)

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

print(df.head())

# reference: removes any leading (starting) and trailing (ending) whitespaces
# https://www.programiz.com/python-programming/methods/string/strip

# ChatGpt, "how can I save my loop result into a dataframe", ask for help of
 ↪  how cna I save appened information into a dataframe.
```

```
                                               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)

```
agencies = []
for link in links:
    response = requests.get(link)
    soup = BeautifulSoup(response.content, 'html.parser')

    agency_tag = soup.find('span', class_='padding-right-2 text-base')
    if agency_tag:
        agency_text = agency_tag.find_next('li').text.strip()
        if agency_text.startswith("Agency:"):
            agency = agency_text.replace("Agency:", "").strip()
        else:
            agency = "Unknown"  # If "Agency" prefix is not present, assign
 ↪  "Unknown"
    else:
        agency = "Unknown"

    agencies.append(agency)

df['Agency'] = agencies

print(df.head())

# ChatGpt, 'Help me debug my if statement inside my for loop, there is
 ↪  error', ask help to combine if statment into for loop
```

```
                                               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

- a. Pseudo-Code (PARTNER 2)

1. Define the main function scrape_enforcement_actions(start_year, start_month)

Check if start_year is valid (>= 2013). If not, return a message indicating the restriction.

2. Create empty variables

Create empty lists for titles, dates, categories, links, and agencies.

Set start_date based on start_year and start_month.

Set page to 1 and more_pages to True.

Initialize agency_cache as an empty dictionary. (to reduce run time)

3. First Loop to Iterate Pages

set restriction of page <= 480

Build the URL for the current page and send a request to the URL and parse the content using BeautifulSoup.

4. Check if there are enforcement actions

If no enforcement actions are found, set more_pages to False and exit the loop.

5. Second Loop

Extract the date and check if it's older than start_date. If so, stop further pages.

Extract details (title, link, category, Agency).

Cache the agency information to avoid duplicate requests.

Append the extracted data to the lists.

6. Move to the next page (if not older than start_date)
7. Save data to CSV

Create a DataFrame and save it to a CSV file.

- b. Create Dynamic Scraper (PARTNER 2)

```
from datetime import datetime

def scrape_actions(start_year, start_month):
    if start_year < 2013:
        print("Data is only available for years >= 2013.")
        return None
# initial empty variables and page
    titles, dates, categories, links, agencies = [], [], [], [], []
    start_date = datetime(start_year, start_month, 1)
    page = 1
    more_pages = True
    agency_cache = {}

    while more_pages and page <= 480:
        url = f"https://oig.hhs.gov/fraud/enforcement/?page={page}"
        print(f"Scraping page {page}...")

        response = requests.get(url)
        soup = BeautifulSoup(response.content, 'html.parser')
        enforcement_actions = soup.find_all(
            'div', class_='usa-card__container')

        if not enforcement_actions:
            more_pages = False
            break

        for action in enforcement_actions:
            date_text = action.find(
                'span', class_='text-base-dark
                ↪ padding-right-105').text.strip()
```

```
            action_date = datetime.strptime(date_text, "%B %d, %Y")

            if action_date < start_date:
                more_pages = False
                break
# Write a condition here to aviod loop all 480 pages, help from ChatGpt

            title_tag = action.find('h2',
 ↪  class_='usa-card__heading').find('a')
            title = title_tag.text.strip()
            link = "https://oig.hhs.gov" + title_tag['href']
            category_tag = action.find(
                'ul', class_='display-inline add-list-reset')
            category = category_tag.find('li').text.strip()

            titles.append(title)
            dates.append(date_text)
            categories.append(category)
            links.append(link)

# Agency Caching: if an agency's page has already been accessed, the cached
 ↪  value will be used, reducing the number of detail page requests.

            if link in agency_cache:
                agency = agency_cache[link]
            else:
                agency_response = requests.get(link)
                agency_soup = BeautifulSoup(
                    agency_response.content, 'html.parser')
                agency_tag = agency_soup.find(
                    'span', class_='padding-right-2 text-base')
                if agency_tag:
                    agency_text = agency_tag.find_next('li').text.strip()
                    if agency_text.startswith("Agency:"):
                        agency = agency_text.replace("Agency:", "").strip()
                    else:
                        agency = "Unknown"  # some do not have agency, put
 ↪  unknown here
                else:
                    agency = "Unknown"
                # Cache the result to avoid future requests
                agency_cache[link] = agency
```

```
            agencies.append(agency)

        page += 1

    df = pd.DataFrame({
        'Title': titles,
        'Date': dates,
        'Category': categories,
        'Link': links,
        'Agency': agencies
    })
    df.to_csv(f"enforcement_actions_{start_year}_{
              start_month}.csv", index=False)
    print(f"Data saved to
     ↪  enforcement_actions_{start_year}_{start_month}.csv")

    return df

# ChatGpt: Can you help me reduces the time of running this loop?,
 ↪  agency_cache to store agency information by link.
# ChatGpt: Can you debug my code, I want to make the loop iterate with pages,
 ↪  and stop at the date I input.
```

```
df_2023 = scrape_actions(2023, 1)
```

```
print(f"Total enforcement actions: {len(df_2023)}")
print(f"The earliest enforcement action date is: {df_2023.tail(1)}")
```

```
Total enforcement actions: 1534
The earliest enforcement action date is:
Title            Date  \
1533  Podiatrist Pays $90,000 To Settle False Billin...  January 3, 2023

                        Category  \
1533  Criminal and Civil Actions

                                                  Link  \
1533  https://oig.hhs.gov/fraud/enforcement/podiatri...

                                                Agency
1533  U.S. Attorney's Office, Southern District of T...
```

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

```
df_2021 = scrape_actions(2021, 1)
```

```
print(f"Total enforcement actions: {len(df_2021)}")
print(f"The earliest enforcement action date is: {df_2021.tail(1)}")
```

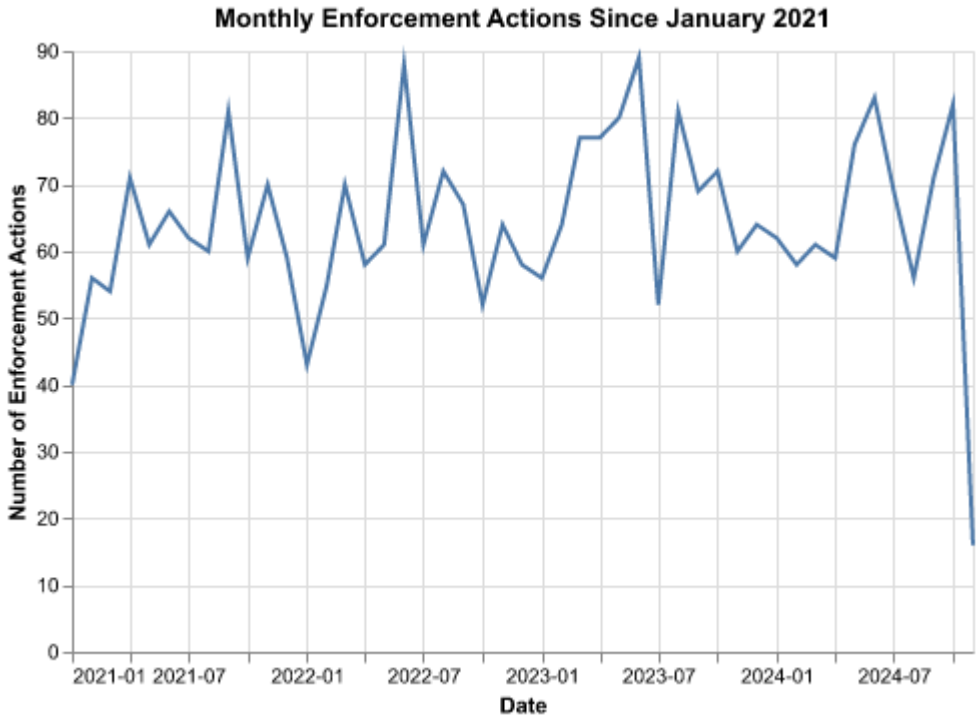
```
Total enforcement actions: 3022
The earliest enforcement action date is:
Title             Date  \
3021  The United States And Tennessee Resolve Claims...  January 4, 2021

                        Category  \
3021  Criminal and Civil Actions

                                                  Link  \
3021  https://oig.hhs.gov/fraud/enforcement/the-unit...

                                                Agency
3021  U.S. Attorney's Office, Middle District of Ten...
```

Step 3: Plot data based on scraped data

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

```
df_2021 = pd.read_csv("enforcement_actions_2021_1.csv")

df_2021['Date'] = pd.to_datetime(df_2021['Date'], format="%B %d, %Y")

df_2021['Year-Month'] = df_2021['Date'].dt.to_period('M')

monthly_counts =
 ↪  df_2021.groupby('Year-Month').size().reset_index(name='Count')
monthly_counts['Year-Month'] = monthly_counts['Year-Month'].dt.to_timestamp()

alt.Chart(monthly_counts).mark_line().encode(
    x=alt.X('Year-Month:T', title='Date', axis=alt.Axis(format='%Y-%m')),
    y=alt.Y('Count:Q', title='Number of Enforcement Actions')
).properties(
    title='Monthly Enforcement Actions Since January 2021',
```

```
    width=450,
    height=300
)

# Reference of datetime:
 ↪  #https://pandas.pydata.org/docs/reference/api/pandas.to_datetime.html
```

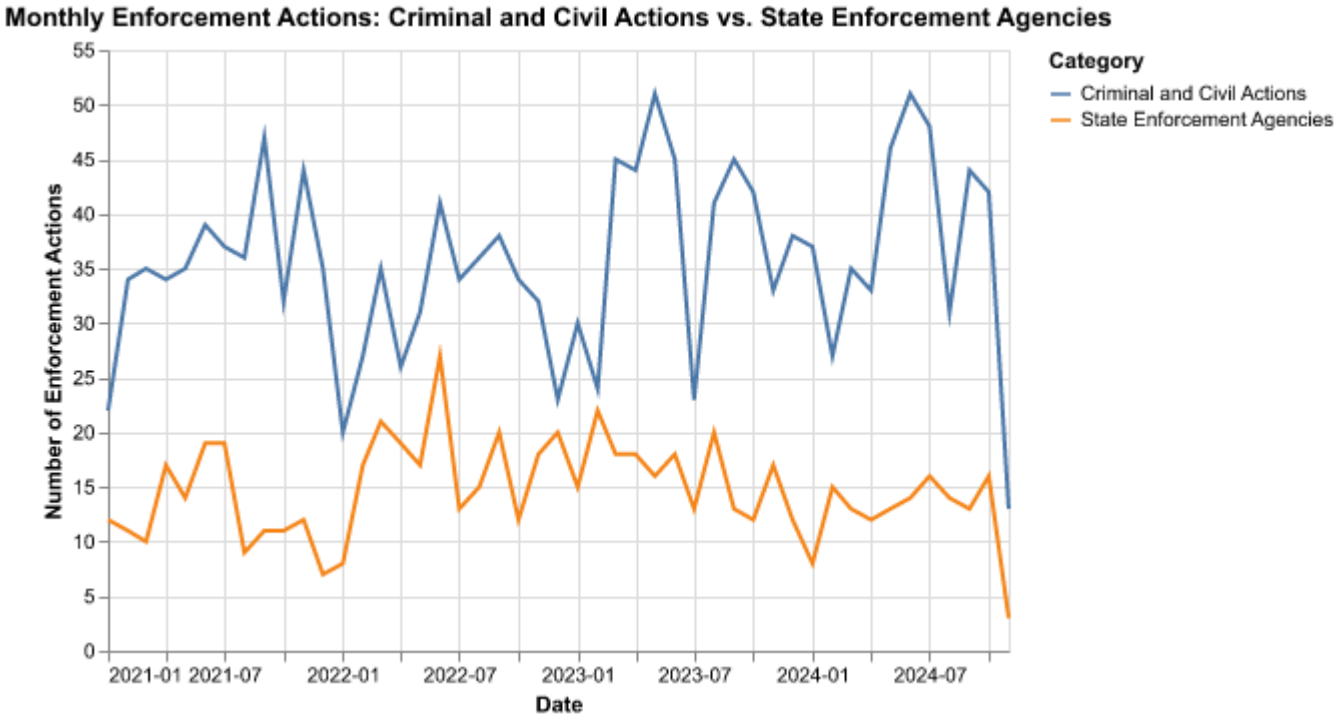


2. Plot the number of enforcement actions categorized: (PARTNER 1)

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

```
df_2021['Year-Month'] = df_2021['Date'].dt.to_period('M').dt.to_timestamp()

df_filtered = df_2021[df_2021['Category'].isin(
    ["Criminal and Civil Actions", "State Enforcement Agencies"])]

category_counts = df_filtered.groupby(
    ['Year-Month', 'Category']).size().reset_index(name='Count')
```

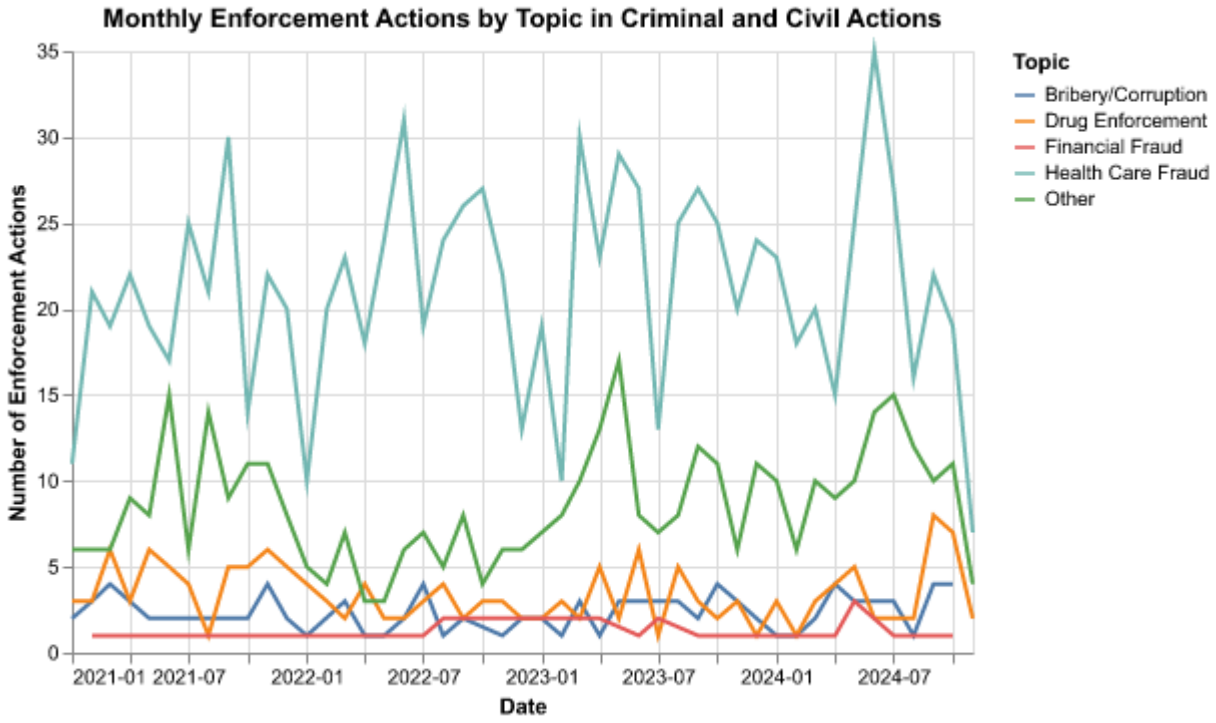
```
alt.Chart(category_counts).mark_line().encode(
    x=alt.X('Year-Month:T', title='Date', axis=alt.Axis(format='%Y-%m')),
    y=alt.Y('Count:Q', title='Number of Enforcement Actions'),
    color=alt.Color('Category:N', title='Category'),
    tooltip=['Year-Month:T', 'Count:Q', 'Category:N']
).properties(
    title='Monthly Enforcement Actions: Criminal and Civil Actions vs. State
 ↪  Enforcement Agencies',
    width=450,
    height=300
)
```



- based on five topics

```
df_criminal_civil = df_filtered[df_filtered['Category']
                                == "Criminal and Civil Actions"]


def assign_topic(title):
    title_lower = title.lower()
```

```
    if "health" in title_lower or "medi" in title_lower:
        return "Health Care Fraud"
    elif "bank" in title_lower or "financial" in title_lower or "laundering"
     ↪  in title_lower or "tax" in title_lower:
        return "Financial Fraud"
    elif "drug" in title_lower or "opioid" in title_lower or "substance" in
     ↪  title_lower:
        return "Drug Enforcement"
    elif "bribe" in title_lower or "corruption" in title_lower or "kickback"
     ↪  in title_lower:
        return "Bribery/Corruption"
    else:
        return "Other"


df_criminal_civil['Topic'] = df_criminal_civil['Title'].apply(assign_topic)

# Chatgpt, ‘can you write a template for me to search for certain words and
 ↪  make it become a new category in new column', ask help for write a
 ↪  function at beginning

topic_counts = df_criminal_civil.groupby(
    ['Year-Month', 'Topic']).size().reset_index(name='Count')

alt.Chart(topic_counts).mark_line().encode(
    x=alt.X('Year-Month:T', title='Date', axis=alt.Axis(format='%Y-%m')),
    y=alt.Y('Count:Q', title='Number of Enforcement Actions'),
    color=alt.Color('Topic:N', title='Topic')
).properties(
    title='Monthly Enforcement Actions by Topic in Criminal and Civil
 ↪  Actions',
    width=450,
    height=300
)
```

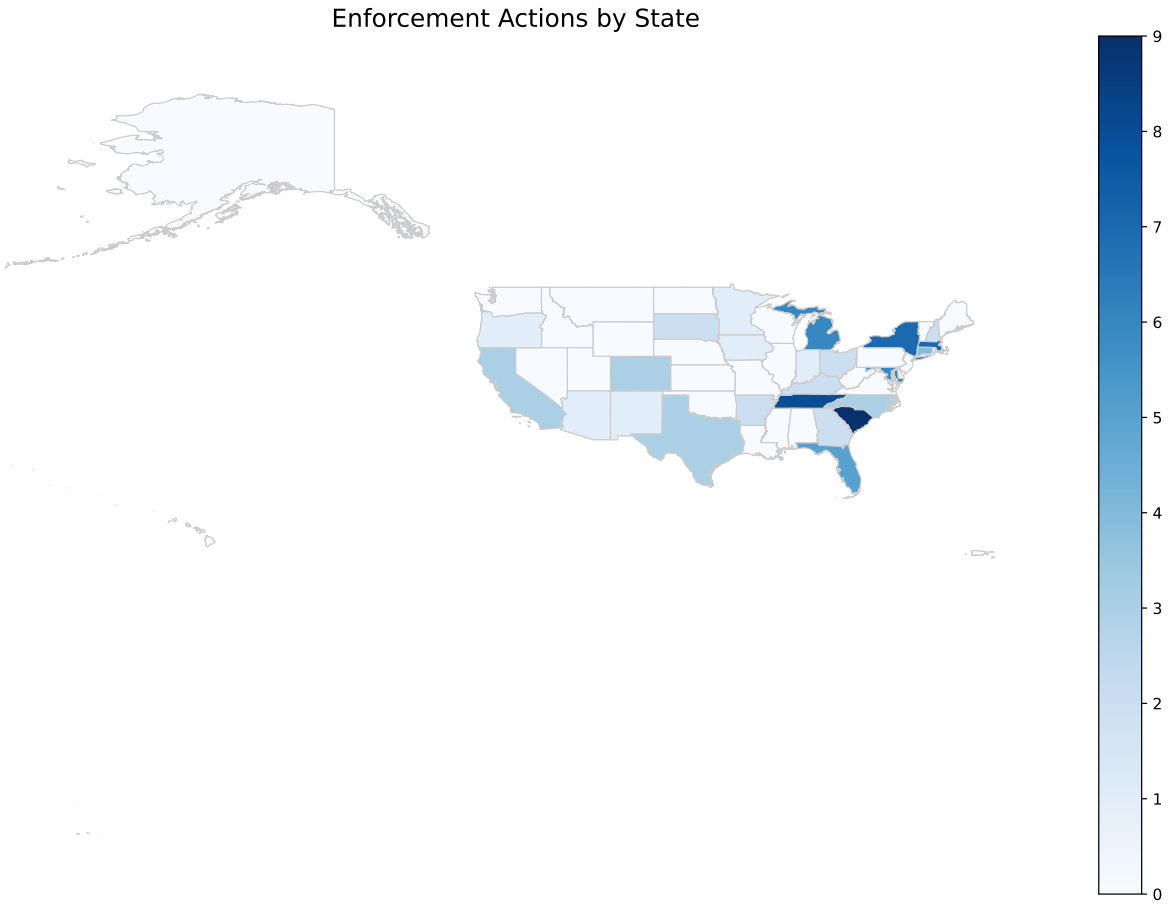


Step 4: Create maps of enforcement activity

1. Map by State (PARTNER 1)

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

state_gdf = gpd.read_file("cb_2018_us_state_500k.shp")
enforcement_data = pd.read_csv("enforcement_actions_2021_1.csv")

state_enforcement = enforcement_data[enforcement_data['Agency'].str.contains(
    "State of", na=False)]
state_enforcement['State'] = state_enforcement['Agency'].str.replace(
    "State of", "").str.strip()

state_counts = state_enforcement['State'].value_counts().reset_index()
state_counts.columns = ['State', 'EnforcementCount']

state_gdf = state_gdf.rename(columns={"NAME": "State"})
```

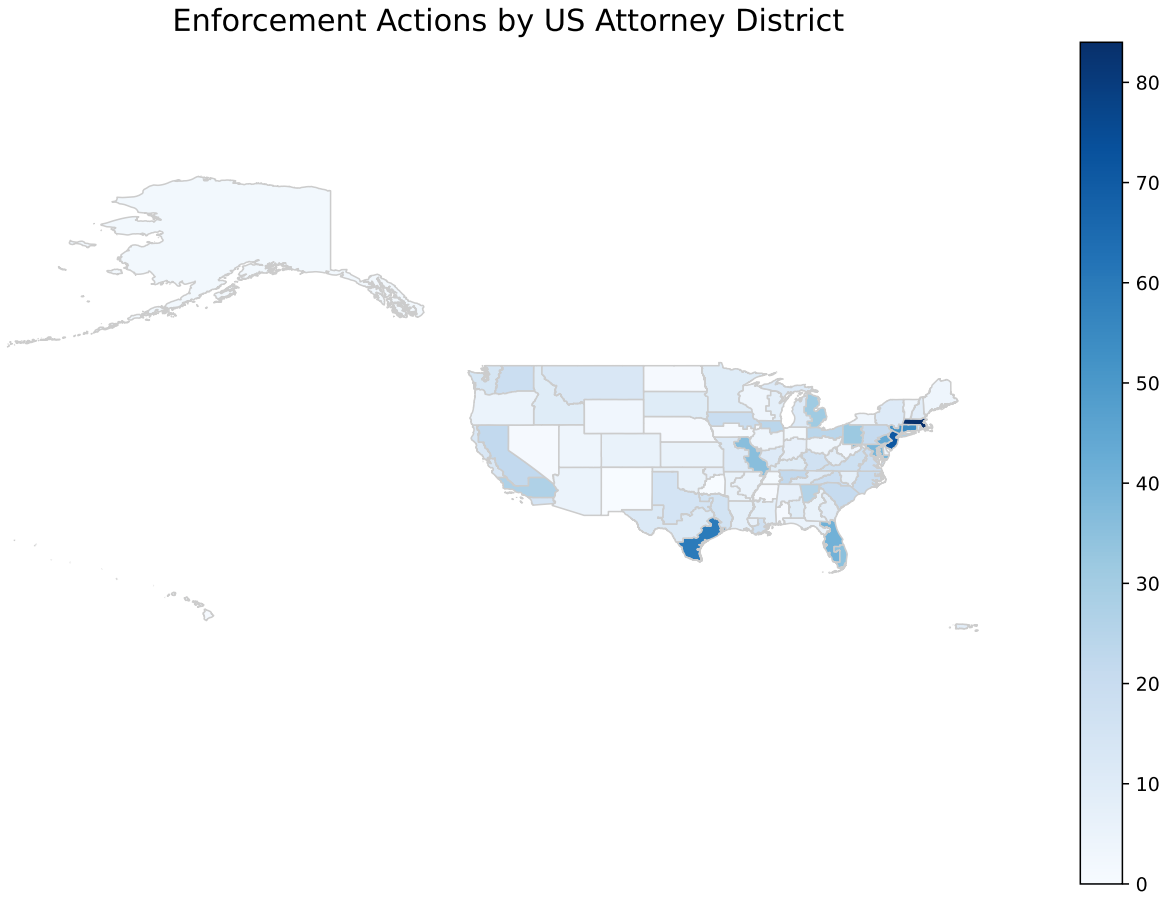
```
merged_gdf = state_gdf.merge(state_counts, on="State", how="left").fillna(0)

fig, ax = plt.subplots(1, 1, figsize=(15, 10))
merged_gdf.plot(column="EnforcementCount", cmap="Blues",
                linewidth=0.8, ax=ax, edgecolor="0.8", legend=True)
ax.set_title("Enforcement Actions by State", fontsize=16)
ax.set_axis_off()

plt.axis('equal')
plt.xlim([-180, -60])

plt.show()

# ChatGpt, "how can I make my graph bigger", ask help for adjust the size
 ↪  bigger
```



2. Map by District (PARTNER 2)

```
district_gdf = gpd.read_file(
    "geo_export_828db5ef-6c4f-412d-aed1-b5f6c9905efd.shp")

enforcement_data = pd.read_csv("enforcement_actions_2021_1.csv")

district_enforcement =
 ↪  enforcement_data[enforcement_data['Agency'].str.contains(
    "District", na=False)]

district_enforcement['District'] = district_enforcement['Agency'].str.split(
    ',').str[1].str.strip()

# Chatgpt, 'I want to extract information after comma, how can I write that',
 ↪  help for /split() function

district_enforcement['District'] =
 ↪  district_enforcement['District'].str.replace(
    r'[^\w\s,.-]', '', regex=True)

# chatgpt, 'there are some columns has some signs, how can I remove them?'
# reference:
 ↪  https://teamtreehouse.com/community/can-someone-explain-the-logic-of-replacews-to-me
district_gdf = district_gdf.rename(columns={"judicial_d": "District"})

district_counts =
 ↪  district_enforcement['District'].value_counts().reset_index()
district_counts.columns = ['District', 'EnforcementCount']

merged_district_gdf = district_gdf.merge(
    district_counts, on="District", how="left").fillna(0)

fig, ax = plt.subplots(1, 1, figsize=(12, 8))
merged_district_gdf.plot(
    column="EnforcementCount",
    cmap="Blues",
    linewidth=0.8,
    ax=ax,
    edgecolor="0.8",
    legend=True
```

```
)

ax.set_title("Enforcement Actions by US Attorney District", fontsize=16)
ax.set_axis_off()

plt.axis('equal')
plt.xlim([-180, -60])
plt.ylim([20, 55])

plt.show()
```



Extra Credit

1. Merge zip code shapefile with population

```
# read file from last PS
zip_codes = gpd.read_file("gz_2010_us_860_00_500k.shp")

population_2020 = pd.read_csv("DECENNIALDHC2020.P1-Data.csv")

population_2020 = population_2020 .drop(index=0).reset_index(drop=True)

population_2020.rename(columns={'NAME': 'ZCTA5'}, inplace=True)

population_2020['ZCTA5'] = population_2020['ZCTA5'].str.replace(
    'ZCTA5 ', '', regex=False)

population_2020 = population_2020.drop(
    columns=['Unnamed: 3'])  # remove weird column

merged_2020 = population_2020.merge(
    zip_codes, left_on='ZCTA5', right_on='ZCTA5', how='left')

print(merged_2020.head(3))
```

```
         GEO_ID_x  ZCTA5 P1_001N        GEO_ID_y   NAME   LSAD  CENSUSAREA  \
0  860Z200US00601  00601   17242  8600000US00601  00601  ZCTA5      64.348
1  860Z200US00602  00602   37548  8600000US00602  00602  ZCTA5      30.613
2  860Z200US00603  00603   49804  8600000US00603  00603  ZCTA5      31.614

                                            geometry
0  POLYGON ((-66.76905 18.13498, -66.76914 18.134...
1  POLYGON ((-67.18285 18.31303, -67.18325 18.312...
2  POLYGON ((-67.16026 18.41564, -67.1601 18.4157...
```

2. Conduct spatial join

```
merged_2020['P1_001N'] = pd.to_numeric(merged_2020['P1_001N'])
# change population to numberic to make sum works
```

```
# merged_2020 is not GeoDF, I cannot merge in the beginning, asked chatGPT
merged_2020 = gpd.GeoDataFrame(
    merged_2020, geometry='geometry', crs=zip_codes.crs)

if merged_2020.crs != district_gdf.crs:
    merged_2020 = merged_2020.to_crs(district_gdf.crs)

joined = gpd.sjoin(merged_2020, district_gdf,
                   how="left", predicate="intersects")

district_population =
 ↪  joined.groupby('District')['P1_001N'].sum().reset_index()
district_population.rename(
    columns={'P1_001N': 'Total_Population'}, inplace=True)

print(district_population)
```

```
                         District  Total_Population
0   Central District of California          19621862
1     Central District of Illinois           2684528
2               District of Alaska            707199
3              District of Arizona           7334666
4             District of Colorado           5935657
..                             ...               ...
86   Western District of Tennessee           1895710
87       Western District of Texas           8203041
88    Western District of Virginia           3067463
89  Western District of Washington           6289276
90   Western District of Wisconsin           2989929

[91 rows x 2 columns]
```

3. Map the action ratio in each district

```
merged_district_gdf = merged_district_gdf.merge(
    district_population, on='District', how='left')

merged_district_gdf['Total_Population'] = pd.to_numeric(
    merged_district_gdf['Total_Population'], errors='coerce')
```

```
merged_district_gdf['EnforcementCount'] =
 ↪  merged_district_gdf['EnforcementCount'].fillna(
    0)
merged_district_gdf['Total_Population'] =
 ↪  merged_district_gdf['Total_Population'].fillna(
    1)  # Avoid division by zero, inspired from chatGPT

merged_district_gdf['EnforcementPerCapita'] =
 ↪  merged_district_gdf['EnforcementCount'] / \
    merged_district_gdf['Total_Population']

print(merged_district_gdf[['District', 'EnforcementCount',
      'Total_Population', 'EnforcementPerCapita']].head())

fig, ax = plt.subplots(1, 1, figsize=(12, 8))
merged_district_gdf.plot(
    column='EnforcementPerCapita',
    cmap='Blues',
    linewidth=0.8,
    ax=ax,
    edgecolor='0.8',
    legend=True
)

ax.set_title(
    "Enforcement Actions per Capita by US Attorney District", fontsize=16)
ax.set_axis_off()

plt.axis('equal')
plt.xlim([-180, -60])
plt.ylim([20, 55])

plt.show()
```

```
                    District  EnforcementCount  Total_Population  \
0  Western District of Kentucky              10.0         3098801.0
1  Eastern District of Kentucky              17.0         2745347.0
2  Southern District of Indiana               7.0         4852556.0
3    Middle District of Alabama              10.0         1539168.0
4  Southern District of Alabama               1.0         1225407.0

   EnforcementPerCapita
```

0	3.227055e-06
1	6.192296e-06
2	1.442539e-06
3	6.497017e-06
4	8.160554e-07