Problem Set 5

Sarah Morrison

2024-11-11

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): Sarah Morrison morrisons
    - Partner 2 (name and cnet ID): NA
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: **SM** **___**
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: **2** Late coins left after submission: **0**
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
from datetime import datetime
import numpy as np

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

import datetime
```

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, 'lxml')
```

```
# ChatGPT Question: use beautiful soup to extract a ul html block
table = soup.find('ul', class_='usa-card-group padding-y-0')
li = table.find_all('li')
print(li[0])
```

```
<li class="usa-card card--list pep-card--minimal mobile:grid-col-12">
<div class="usa-card__container">
<header class="usa-card__header">
<h2 class="usa-card__heading">
<a
href="/fraud/enforcement/pharmacist-and-brother-convicted-of-15m-medicare-medicaid-and-private-
and Brother Convicted of $15M Medicare, Medicaid, and Private Insurer Fraud
Scheme</a>
</h2>
<div class="font-body-sm margin-top-1">
<span class="text-base-dark padding-right-105">November 8, 2024</span><ul
class="display-inline add-list-reset"><li class="display-inline-block usa-tag
text-no-lowercase text-base-darkest bg-base-lightest margin-right-1">Criminal and
Civil Actions</li></ul>
</div>
</header>
</div>
</li>
```

```
#ChatGPT Question: code a dataset that has each action's header, date, category,
 ↪  and href link
actions_data = []

# Loop through each relevant <li> element
for l in li:
    header_tag = l.find("h2")
    date_tag = l.find("span")
    link_tag = l.find("a")
    category_tag = l.find("ul").find("li") if l.find("ul") else None

    # Skip entries where all fields are None
    if not (header_tag or date_tag or link_tag or category_tag):
        continue

    # Append non-empty rows to the list
    actions_data.append({
        "Header": header_tag.get_text(strip=True) if header_tag else None,
        "Date": date_tag.get_text(strip=True) if date_tag else None,
        "Category": category_tag.get_text(strip=True) if category_tag else None,
        "Link": f"https://oig.hhs.gov{link_tag['href']}" if link_tag and
         ↪  link_tag.has_attr('href') else None
    })

# Convert to a DataFrame
enforcement_actions = pd.DataFrame(actions_data)

# Display DataFrame to
enforcement_actions.head()
```

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

Second Page

```
url = 'https://oig.hhs.gov/fraud/enforcement/?page=2'
response = requests.get(url)
soup = BeautifulSoup(response.content, 'lxml')
```

```
# ChatGPT Question: use beautiful soup to extract a ul html block
table = soup.find('ul', class_='usa-card-group padding-y-0')
li = table.find_all('li')
print(li[0])
```

```
<li class="usa-card card--list pep-card--minimal mobile:grid-col-12">
<div class="usa-card__container">
<header class="usa-card__header">
<h2 class="usa-card__heading">
<a
href="/fraud/enforcement/boise-health-care-company-and-former-therapist-resolve-fraudulent-psyc
Health Care Company And Former Therapist Resolve Fraudulent Psychotherapy Billing
Allegations Involving Vulnerable Refugee Population</a>
</h2>
<div class="font-body-sm margin-top-1">
<span class="text-base-dark padding-right-105">October 29, 2024</span><ul
class="display-inline add-list-reset"><li class="display-inline-block usa-tag
text-no-lowercase text-base-darkest bg-base-lightest margin-right-1">Criminal and
Civil Actions</li></ul>
</div>
</header>
</div>
</li>
```

```
#ChatGPT Question: code a dataset that has each action's header, date, category,
 ↪  and href link
actions_data = []

# Loop through each relevant <li> element
for l in li:
    header_tag = l.find("h2")
    date_tag = l.find("span")
    link_tag = l.find("a")
    category_tag = l.find("ul").find("li") if l.find("ul") else None

    # Skip entries where all fields are None
    if not (header_tag or date_tag or link_tag or category_tag):
        continue

    # Append non-empty rows to the list
    actions_data.append({
        "Header": header_tag.get_text(strip=True) if header_tag else None,
        "Date": date_tag.get_text(strip=True) if date_tag else None,
        "Category": category_tag.get_text(strip=True) if category_tag else None,
        "Link": f"https://oig.hhs.gov{link_tag['href']}" if link_tag and
         ↪  link_tag.has_attr('href') else None
    })
```

```
# Convert to a DataFrame
enforcement_actions = pd.DataFrame(actions_data)

# Display DataFrame to
enforcement_actions.head()
```

	Header	Date	Category	Link
0	Boise Health Care Company And Former Therapist...	October 29, 2024	Criminal and Civil Actions	http
1	Drug Dealer Sentenced To Nine Years In Prison ...	October 29, 2024	Criminal and Civil Actions	http
2	Former Sioux City Plastic Surgeon Agrees To Pa...	October 29, 2024	Criminal and Civil Actions	http
3	Medical Billing Company Owner Pleads Guilty To...	October 29, 2024	Criminal and Civil Actions	http
4	Doctor Sentenced For $54M Medicare Fraud Scheme	October 29, 2024	Criminal and Civil Actions	http

2. Crawling (PARTNER 1)

```
# Practice with first link to find the right information
url_macomb =
 ↪  'https://oig.hhs.gov/fraud/enforcement/macomb-county-doctor-and-pharmacist-agree-to-pay-700
response_macomb = requests.get(url_macomb)
soup_macomb = BeautifulSoup(response_macomb.content, 'lxml')
table_macomb = soup_macomb.find('ul', class_='usa-list usa-list--unstyled
 ↪  margin-y-2')
li_macomb = table_macomb.find_all('li')
print(li_macomb)
```

```
[<li><span class="padding-right-2 text-base">Date:</span>November 4, 2024</li>,
<li><span class="padding-right-2 text-base">Agency:</span>U.S. Attorney's Office,
Eastern District of Michigan</li>, <li>
<span class="padding-right-1 text-base">Enforcement Types:</span>
<ul class="usa-list usa-list--unstyled margin-top-0 display-inline-block">
<li class="display-inline">
                                        Criminal and Civil Actions
                                    </li>
</ul>
</li>, <li class="display-inline">
                                        Criminal and Civil Actions
                                    </li>]
```

```
# Used ChatGPT to help get rid of the Agency: part
def get_agency(link):
    try:
        response = requests.get(link)
```

```
        page_soup = BeautifulSoup(response.content, "lxml")
        second_li = page_soup.find("ul", class_='usa-list usa-list--unstyled
↪  margin-y-2').find_all('li')[1]
        # Extract text after 'Agency:'
        agency_text = second_li.get_text(strip=True)
        # Remove 'Agency:' part if it exists
        if agency_text.startswith('Agency:'):
            agency_text = agency_text[len('Agency:'):].strip()
        return agency_text
    except Exception as e:
        print(f"Error fetching {link}: {e}")
        return None

# Apply the function to each Link in the DataFrame
enforcement_actions['Agency'] = enforcement_actions['Link'].apply(get_agency)

# Display the updated DataFrame
enforcement_actions.head()
```

	Header	Date	Category	Link
0	Boise Health Care Company And Former Therapist...	October 29, 2024	Criminal and Civil Actions	http
1	Drug Dealer Sentenced To Nine Years In Prison ...	October 29, 2024	Criminal and Civil Actions	http
2	Former Sioux City Plastic Surgeon Agrees To Pa...	October 29, 2024	Criminal and Civil Actions	http
3	Medical Billing Company Owner Pleads Guilty To...	October 29, 2024	Criminal and Civil Actions	http
4	Doctor Sentenced For $54M Medicare Fraud Scheme	October 29, 2024	Criminal and Civil Actions	http

```
enforcement_actions['Date'] = pd.to_datetime(enforcement_actions['Date'])
enforcement_actions.head(
)
```

	Header	Date	Category	Link
0	Boise Health Care Company And Former Therapist...	2024-10-29	Criminal and Civil Actions	https://oig
1	Drug Dealer Sentenced To Nine Years In Prison ...	2024-10-29	Criminal and Civil Actions	https://oig
2	Former Sioux City Plastic Surgeon Agrees To Pa...	2024-10-29	Criminal and Civil Actions	https://oig
3	Medical Billing Company Owner Pleads Guilty To...	2024-10-29	Criminal and Civil Actions	https://oig
4	Doctor Sentenced For $54M Medicare Fraud Scheme	2024-10-29	Criminal and Civil Actions	https://oig

```
# ChatGPT Question: use beautiful soup to extract a ul html block
table = soup.find('ul', class_='usa-card-group padding-y-0')
li = table.find_all('li')
print(li[0])
```

```
<li class="usa-card card--list pep-card--minimal mobile:grid-col-12">
<div class="usa-card__container">
<header class="usa-card__header">
<h2 class="usa-card__heading">
<a
href="/fraud/enforcement/boise-health-care-company-and-former-therapist-resolve-fraudulent-psy
Health Care Company And Former Therapist Resolve Fraudulent Psychotherapy Billing
Allegations Involving Vulnerable Refugee Population</a>
</h2>
<div class="font-body-sm margin-top-1">
<span class="text-base-dark padding-right-105">October 29, 2024</span><ul
class="display-inline add-list-reset"><li class="display-inline-block usa-tag
text-no-lowercase text-base-darkest bg-base-lightest margin-right-1">Criminal and
Civil Actions</li></ul>
</div>
</header>
</div>
</li>
```

```
#ChatGPT Question: code a dataset that has each action's header, date, category,
 ↪  and href link
actions_data = []

# Loop through each relevant <li> element
for l in li:
    header_tag = l.find("h2")
    date_tag = l.find("span")
    link_tag = l.find("a")
    category_tag = l.find("ul").find("li") if l.find("ul") else None

    # Skip entries where all fields are None


    # Append non-empty rows to the list
    actions_data.append({
        "Header": header_tag.get_text(strip=True) if header_tag else None,
        "Date": date_tag.get_text(strip=True) if date_tag else None,
        "Category": category_tag.get_text(strip=True) if category_tag else None,
        "Link": f"https://oig.hhs.gov{link_tag['href']}" if link_tag and
         ↪  link_tag.has_attr('href') else None
    })

# Convert to a DataFrame
enforcement_actions = pd.DataFrame(actions_data)

# Display DataFrame to
enforcement_actions.head()
```

	Header	Date	Category	Link
0	Boise Health Care Company And Former Therapist...	October 29, 2024	Criminal and Civil Actions	http
1	None	None	None	Non
2	Drug Dealer Sentenced To Nine Years In Prison ...	October 29, 2024	Criminal and Civil Actions	http
3	None	None	None	Non
4	Former Sioux City Plastic Surgeon Agrees To Pa...	October 29, 2024	Criminal and Civil Actions	http

Step 2: Making the scraper dynamic

1. Turning the scraper into a function

- a. Pseudo-Code (PARTNER 2) def scrape_enforcement_actions(year, month): if year < 2013: print("Please enter a year from 2013 onward.") return None

  start_date = datetime(year, month, 1) base_url = "https://oig.hhs.gov" all_data = []

  page_num = 1 while True: url = base_url + "/fraud/enforcement/?page=" + str(page_num) response = requests.get(url) soup = BeautifulSoup(response.content, 'lxml')

```
# Locate enforcement actions on the page
actions_list = soup.find('ul', class_='usa-card-group padding-y-0')
if not actions_list:
    break  # Stop if there are no actions on this page

actions = actions_list.find_all('li', class_="usa-card")  # Get all 'li'
elements
if not actions:
    break

for action in actions:
    # Find the span element for the date
    span_element = action.find('span', class_='text-base-dark
    padding-right-105')
    if not span_element or not span_element.text:
        print("No valid span element found for date in action:", action)
        continue

    try:
        date_text = span_element.text.strip()
        action_date = datetime.strptime(date_text, "%B %d, %Y")
    except Exception as e:
        print(f"Error parsing date: {e}")
        continue

    # Collect other action details
    header = action.find("h2").text.strip()
```

```
        category = action.find("ul").find("li").text.strip() if
        action.find("ul") else "N/A"
        link = action.find("a")['href']

        # Ensure the link has the full URL scheme
        if link.startswith("/"):
            link = base_url + link

        # Visit each link to get agency name
        action_response = requests.get(link)
        action_soup = BeautifulSoup(action_response.content, 'lxml')
        agency = action_soup.find("ul", class_='usa-list usa-list--unstyled
        margin-y-2')
        if agency:
            agency_text = agency.find_all('li')[1].text.strip() if
            len(agency.find_all('li')) > 1 else "N/A"
            if agency_text.startswith('Agency:'):
                agency_text = agency_text[len('Agency:'):].strip()
        else:
            agency_text = "N/A"

        all_data.append([header, action_date.strftime("%Y-%m-%d"), category,
        link, agency_text])

    # Stop loop if last action date on page is before the start date
    if action_date < start_date:
        break

    time.sleep(1)  # Delay to avoid server blocks
    page_num += 1
```

Create a DataFrame and save as CSV

df = pd.DataFrame(all_data, columns=[“Title”, “Date”, “Category”, “Link”, “Agency”]) df.to_csv(f”enforcement_actions_{year}_{month}.csv”, index=False) return df

b. Create Dynamic Scraper (PARTNER 2)

```
# Worked with ChatGPT to fix all my bugs
def scrape_enforcement_actions(year, month):
    if year < 2013:
        print("Please enter a year from 2013 onward.")
        return None

    start_date = datetime(year, month, 1)
    base_url = "https://oig.hhs.gov"
```

```
    all_data = []

    page_num = 1
    while True:
        url = base_url + "/fraud/enforcement/?page=" + str(page_num)
        response = requests.get(url)
        soup = BeautifulSoup(response.content, 'lxml')

        # Locate enforcement actions on the page
        actions_list = soup.find('ul', class_='usa-card-group padding-y-0')
        if not actions_list:
            break  # Stop if there are no actions on this page

        actions = actions_list.find_all('li', class_="usa-card")  # Get all 'li'
↪   elements
        if not actions:
            break

        for action in actions:
            # Find the span element for the date
            span_element = action.find('span', class_='text-base-dark
↪   padding-right-105')
            if not span_element or not span_element.text:
                print("No valid span element found for date in action:", action)
                continue

            try:
                date_text = span_element.text.strip()
                action_date = datetime.strptime(date_text, "%B %d, %Y")
            except Exception as e:
                print(f"Error parsing date: {e}")
                continue

            # Collect other action details
            header = action.find("h2").text.strip()
            category = action.find("ul").find("li").text.strip() if
↪   action.find("ul") else "N/A"
            link = action.find("a")['href']

            # Ensure the link has the full URL scheme
            if link.startswith("/"):
                link = base_url + link

            # Visit each link to get agency name
            action_response = requests.get(link)
            action_soup = BeautifulSoup(action_response.content, 'lxml')
            agency = action_soup.find("ul", class_='usa-list usa-list--unstyled
↪   margin-y-2')
```

```
            if agency:
                agency_text = agency.find_all('li')[1].text.strip() if
↪   len(agency.find_all('li')) > 1 else "N/A"
                if agency_text.startswith('Agency:'):
                    agency_text = agency_text[len('Agency:'):].strip()
            else:
                agency_text = "N/A"

            all_data.append([header, action_date.strftime("%Y-%m-%d"), category,
↪   link, agency_text])

        # Stop loop if last action date on page is before the start date
        if action_date < start_date:
            break

        time.sleep(1)  # Delay to avoid server blocks
        page_num += 1

    # Create a DataFrame and save as CSV
    df = pd.DataFrame(all_data, columns=["Title", "Date", "Category", "Link",
↪   "Agency"])
    df.to_csv(f"enforcement_actions_{year}_{month}.csv", index=False)
    return df

# Run the scraper from January 2023
df_jan2023 = scrape_enforcement_actions(2023, 1)
print(df_jan2023.head())
```

```
df_jan2023 = df_jan2023.drop(df_jan2023.index[1534:1540])
```

```
import geopandas as gpd
filepath =
↪   "/Users/sarahmorrison/Downloads/GitHub/problem-set-5-sarah/enforcement_actions_2023_1.csv"
enforcement_actions_2023_01 = gpd.read_file(filepath)
```

c. Test Partner's Code (PARTNER 1)

```
enforcement_actions_2021_01 = scrape_enforcement_actions(2021, 1)
```

```
filepath =
↪   "/Users/sarahmorrison/Downloads/GitHub/problem-set-5-sarah/enforcement_actions_2021_1.csv"
enforcement_actions_2021_01 = gpd.read_file(filepath)
```

```
enforcement_actions_2021_01 =
 ↪  enforcement_actions_2021_01.drop(enforcement_actions_2021_01.index[3022:3040])
```

Step 3: Plot data based on scraped data

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

```
enforcement_actions_2021_01['Date'] =
 ↪  pd.to_datetime(enforcement_actions_2021_01['Date'])
enforcement_actions_2021_01['month_year'] =
 ↪  enforcement_actions_2021_01['Date'].dt.to_period('M')
grouped_enforcement_actions =
 ↪  enforcement_actions_2021_01.groupby('month_year').size().reset_index(name='count')
print(grouped_enforcement_actions['month_year'].dtype)

grouped_enforcement_actions['month_year'] =
 ↪  grouped_enforcement_actions['month_year'].dt.to_timestamp()
grouped_enforcement_actions['month_year'].head()
```

```
period[M]

0   2021-01-01
1   2021-02-01
2   2021-03-01
3   2021-04-01
4   2021-05-01
Name: month_year, dtype: datetime64[ns]
```

```
grouped_month_chart = alt.Chart(grouped_enforcement_actions, title="Number of
 ↪  Enforcement Actions per Month 2021-2024").mark_line().encode(
  x=alt.X('month_year:T', title="Date", axis=alt.Axis(format='%m %Y')),
  y=alt.Y('count:Q', title='Number of Enforcement Actions')
).properties(
  width=500
)
alt.renderers.enable('default')

grouped_month_chart
```

```
alt.Chart(...)
```

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

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

```
enforcement_actions_2021_01['Category'].unique()

grouped_by_actions = enforcement_actions_2021_01.groupby(['Category',
 ↪  'month_year']).size().reset_index(name='count')
grouped_by_actions['action_type'] = grouped_by_actions['Category'].apply(
    lambda x: 'State Enforcement Agencies' if 'State' in str(x) else 'Criminal
     ↪  and Civil Actions'
)
grouped_by_action_type = grouped_by_actions.groupby(['action_type',
 ↪  'month_year']).sum().reset_index()
grouped_by_action_type['month_year'] =
 ↪  grouped_by_action_type['month_year'].dt.to_timestamp()

grouped_action_chart = alt.Chart(grouped_by_action_type, title="Number of State
 ↪  Enforcement Agency Actions and Criminal and Civil Actions per Month
 ↪  2021-2024").mark_line().encode(
  alt.X('month_year:T', title="Date", axis=alt.Axis(format='%m %Y')),
  alt.Y('count:Q', title='Number of Enforcement Actions'),
  alt.Color('action_type:N', title='Action Type')
).properties(
  width=500
)
alt.renderers.enable('default')

grouped_action_chart
```

```
alt.Chart(...)
```

- based on five topics

```
conditions = [

 ↪  enforcement_actions_2021_01['Title'].str.contains('bank|financial|embezzlement|pricing
 ↪  data|fund|remuneration|theft|dollar|billing|scheme|financial fraud',
 ↪  case=False, na=False),

 ↪  enforcement_actions_2021_01['Title'].str.contains('drug|pill|prescrib|substance|morphine|o
 ↪  case=False, na=False),

 ↪  enforcement_actions_2021_01['Title'].str.contains('bribery|corruption|fabricat|kickback|br
 ↪  case=False, na=False),
    enforcement_actions_2021_01['Title'].str.contains('exclude|medicaid|dr.|false
 ↪  claim|submitting claim|treat|examination|health
 ↪  care|medicare|medication|healthcare fraud', case=False, na=False)
```

```
]

# Define the corresponding values for the new column
choices = ['Financial Fraud', 'Drug Enforcement', 'Bribery/Corruption', 'Health
 ↪  Care Fraud']

# Create the new column based on the conditions
enforcement_actions_2021_01['fraud'] = np.select(conditions, choices,
 ↪  default='Other')

grouped_by_fraud = enforcement_actions_2021_01.groupby(['fraud',
 ↪  'month_year']).size().reset_index(name='count')
grouped_by_fraud['month_year'] = grouped_by_fraud['month_year'].dt.to_timestamp()

grouped_fraud_chart = alt.Chart(grouped_by_fraud, title="Number of Enforcement
 ↪  Actions among Five Types of Criminal and Civil Action per Month
 ↪  2021-2024").mark_line().encode(
  alt.X('month_year:T', title="Date", axis=alt.Axis(format='%m %Y')),
  alt.Y('count:Q', title='Number of Enforcement Actions'),
  alt.Color('fraud:N', title='Type of Fraud')
).properties(
  width=500
)
alt.renderers.enable('default')

grouped_fraud_chart
```

```
alt.Chart(...)
```

Step 4: Create maps of enforcement activity

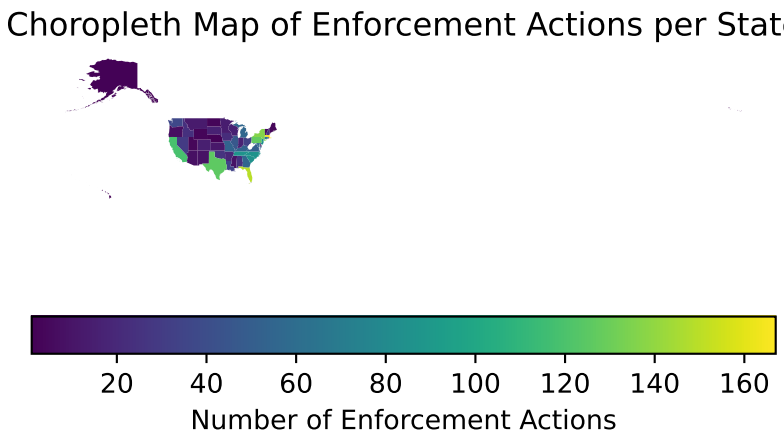
1. Map by State (PARTNER 1)

```
states_list = [
    "Alabama", "Alaska", "Arizona", "Arkansas", "California",
    "Colorado", "Connecticut", "Delaware", "Florida", "Georgia",
    "Hawaii", "Idaho", "Illinois", "Indiana", "Iowa", "Kansas",
    "Kentucky", "Louisiana", "Maine", "Maryland", "Massachusetts",
    "Michigan", "Minnesota", "Mississippi", "Missouri", "Montana",
    "Nebraska", "Nevada", "New Hampshire", "New Jersey", "New Mexico",
    "New York", "North Carolina", "North Dakota", "Ohio", "Oklahoma",
    "Oregon", "Pennsylvania", "Rhode Island", "South Carolina",
    "South Dakota", "Tennessee", "Texas", "Utah", "Vermont",
    "Virginia", "Washington", "West Virginia", "Wisconsin", "Wyoming"
]
```

```
# Function to extract state name if it's in the list of states
def extract_state(agency_info):
    for state in states_list:
        if state.lower() in agency_info.lower():
            return state
    return "Unknown"  # If no state found

# Apply the function to extract states from the 'agency_info' column
enforcement_actions_2021_01['state_cleaned'] =
 ↪  enforcement_actions_2021_01['Agency'].apply(extract_state)
```

```
state_counts =
 ↪  enforcement_actions_2021_01.groupby('state_cleaned').size().reset_index(name='action_count
```

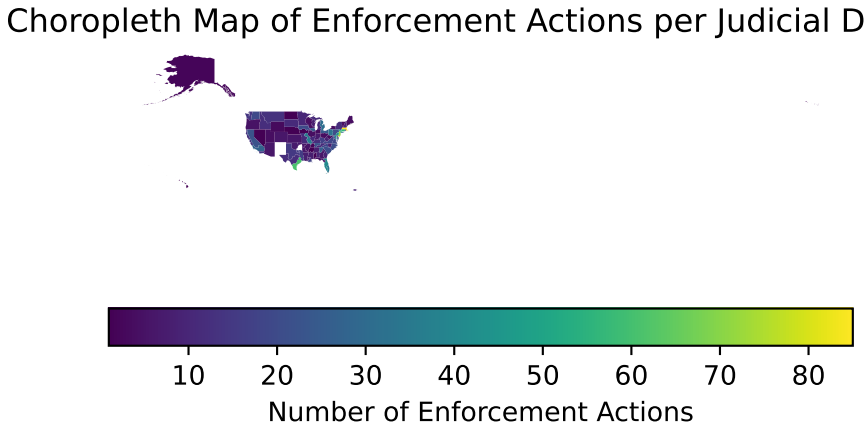
```
zip_shapefile = gpd.read_file("cb_2018_us_state_500k.zip")
zip_shapefile.head

merged_states = zip_shapefile.merge(state_counts, left_on='NAME',
 ↪  right_on='state_cleaned', how='left')

merged_states = merged_states.dropna(subset=['state_cleaned'])

import matplotlib.pyplot as plt

fig, ax = plt.subplots(1, 1, figsize=(5, 5))

merged_states.plot(column='action_count', ax=ax, legend=True,
         legend_kwds={'label': "Number of Enforcement Actions", 'orientation':
 ↪  "horizontal"})
ax.set_title('Choropleth Map of Enforcement Actions per State')
ax.set_axis_off()
plt.show()
```



2. Map by District (PARTNER 2)

```
district_counts =
 ↪  enforcement_actions_2021_01.groupby('Agency').size().reset_index(name='action_count')

district_shapefile = gpd.read_file('US Attorney Districts Shapefile
 ↪  simplified_20241111.zip')

district_shapefile.head()

district_levels = district_shapefile['judicial_d'].to_list()

def find_matching_district(text):
    for district in district_levels:
        if district in text:
            return district
    return None

# Apply the function to the DataFrame
district_counts['judicial_d'] =
 ↪  district_counts['Agency'].apply(find_matching_district)

judicial_d_counts = district_counts.groupby('judicial_d').sum().reset_index()

merged_districts = district_shapefile.merge(judicial_d_counts, on='judicial_d',
 ↪  how='left')

fig, ax = plt.subplots(1, 1, figsize=(5, 5))

merged_districts.plot(column='action_count', ax=ax, legend=True,
         legend_kwds={'label': "Number of Enforcement Actions", 'orientation':
 ↪  "horizontal"})
```

```
ax.set_title('Choropleth Map of Enforcement Actions per Judicial District')
ax.set_axis_off()
plt.show()
```



Extra Credit

1. **Merge zip code shapefile with population**

2. **Conduct spatial join**

3. **Map the action ratio in each district**