30538 Problem Set 5: Web Scraping

Kohei Inagaki and Toshiyuki Kindaichi

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 (Kohei Inagaki):
   - Partner 2 (Toshiyuki Kindaichi):
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: KI and TK
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: **___** Late coins left after submission: **___**
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

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

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

Step 1: Develop initial scraper and crawler

1. Scraping (PARTNER 1)

```python
import pandas as pd
import altair as alt
import time
import requests
from bs4 import BeautifulSoup
from datetime import datetime
import geopandas as gpd
import matplotlib.pyplot as plt
```

```python
# Prepare for parsering HTML
url = 'https://oig.hhs.gov/fraud/enforcement/'
response = requests.get(url)
with open('enforcement_actions_page.html', 'r') as page:
    text = page.read()
soup = BeautifulSoup(response.text, 'lxml')
```

```python
print(response.text[:500])
```

```
<!DOCTYPE html>
<html class="no-js" lang="en">
    <head>
        <meta charset="utf-8" />
        <meta name="viewport" content="width=device-width, initial-scale=1"
        />
        <meta http-equiv="x-ua-compatible" content="ie=edge">
        <link rel="icon" type="image/ico"
        href="/static/img/favicon.c3505f52f923.ico">
```

```
<meta name="twitter:card" content="summary">
<meta name="twitter:title" content="Enforcement Actions">
<meta name="twi
```

By inspecting the page, we found title, date, category, and ling is included in the following HTML "[li class='usa-card…']…[/li]"

```python
# Set the list for enforcement actions
enforcement_actions = []

# Set loop to substract the data from HTML
for item in soup.find_all('li', class_='usa-card'):
    # Title and link
    title_tag = item.find('h2', class_='usa-card__heading').find('a')
    title = title_tag.get_text()
    link = 'https://oig.hhs.gov' + title_tag['href']  # Define the full link name

    # Date
    date_tag = item.find('span', class_='text-base-dark')
    date = date_tag.get_text() if date_tag else 'N/A'

    # Category
    category_tag = item.find('li', class_='usa-tag')
    category = category_tag.get_text() if category_tag else 'N/A'
    # Add to the list
    enforcement_actions.append({
        'Title': title,
        'Date': date,
        'Category': category,
        'Link': link
    })

# Display the result
df = pd.DataFrame(enforcement_actions)
print(df.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)

By checking the link, we found that the name of the agency involved is listed as ‘Agency’ in ‘Action Details’ tag, and some of the link do not have the ‘Agency.’ Then, by inspecting the HTML, we discovered

```python
for action in enforcement_actions:
    link = action['Link']
    response = requests.get(link)

    # Parse HTML with BeautifulSoup
    detail_soup = BeautifulSoup(response.text, 'lxml')

    # Search <li>  tag including 'Agency:'
    agency = 'N/A'  # set initial value as N/A
    for li in detail_soup.find_all('li'):
        if 'Agency:' in li.get_text():
            # Remove word 'Agency:' to get only the name of agency involved
            agency = li.get_text().replace('Agency:', '').strip()
            break

    # Add to the list
    action['Agency'] = agency

    # Wait a half second
    time.sleep(0.5)

df_full = pd.DataFrame(enforcement_actions)
print(df_full.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

- a. Pseudo-Code (PARTNER 2)

1. Define the function as scrape_enforcement_actions with arguments month and year.
2. If year is less than 2013, display a message to the user and end the function.
3. Prepare a list to store the data.
4. Start the loop (using a while loop):

- Generate the URL for each page and retrieve the HTML.
- Parse the HTML and retrieve elements containing enforcement actions.
- For each action, extract the required information (title, date, category, link) and add it to the list.
- If there is a next page, wait 1 second, then increment the page number.
- If there is no next page, exit the loop.

5. Convert the list to a DataFrame and save it as a CSV file.
6. Return the DataFrame.

- b. Create Dynamic Scraper (PARTNER 2)

```python
def scrape_enforcement_actions(month, year):
    # Check the appropriate year >= 2013
    if year < 2013:
        print("Enter a year greater than or equal to 2013.")
        return

    # Set base url and current date
    base_url = 'https://oig.hhs.gov/fraud/enforcement/'
    current_date = datetime.now()
    start_date = datetime(year, month, 1)  # set the start day
```

```
# List for encforcement actions
enforcement_actions_2 = []

# Condition for first page
page = 1
while True:
    # No 'page' on URL when it is first page
    if page == 1:
        url = base_url
    else:
        url = f"{base_url}?page={page}"

    response = requests.get(url)

    # set soup
    soup = BeautifulSoup(response.text, 'lxml')

    # Get the action information
    actions = soup.find_all('li', class_='usa-card')
    if not actions:
        # Stop the loop if no data available
        break

    for item in actions:
        # same process as step 1
        title_tag = item.find('h2', class_='usa-card__heading').find('a')
        title = title_tag.get_text()
        link = 'https://oig.hhs.gov' + title_tag['href']

        # Date
        date_tag = item.find('span', class_='text-base-dark')
        date_str = date_tag.get_text() if date_tag else 'N/A'
        try:
            action_date = datetime.strptime(date_str, '%B %d, %Y')
            # Attribution: Ask ChatGPT how to remove if NAs show up
        except ValueError:
            action_date = None

        # Stop crawling if the date is before the start date
        if action_date and action_date < start_date:
            return pd.DataFrame(enforcement_actions_2)
```

```python
            # category
            category_tag = item.find('li', class_='usa-tag')
            category = category_tag.get_text() if category_tag else 'N/A'

            # agency
            agency = 'N/A'
            detail_response = requests.get(link)
            detail_soup = BeautifulSoup(detail_response.text, 'lxml')
            for li in detail_soup.find_all('li'):
                if 'Agency:' in li.get_text():
                    agency = li.get_text().replace('Agency:', '').strip()
                    break

            # Add info to the list
            enforcement_actions_2.append({
                'Title': title,
                'Date': date_str,
                'Category': category,
                'Link': link,
                'Agency': agency
            })

        # One second wait
        time.sleep(1)
        page += 1

    # Save the data as dataframe
    df = pd.DataFrame(enforcement_actions_2)
    filename = f"enforcement_actions_{year}_{month:02}.csv"
    df.to_csv(filename, index=False)
    print(f"Data saved to {filename}")

    return df
```

```python
# Get the data from Jan, 2023 to current
df_2023 = scrape_enforcement_actions(1, 2023)

# The number of enforcement action
print("Number of enforcement actions:", len(df_2023))
```

```
# Details of the earliest action since January 2023
earliest_action = df_2023.iloc[-1]
print("Earliest enforcement action:")
print(earliest_action)
print(df_2023.head())
```

```
Number of enforcement actions: 1534
Earliest enforcement action:
Title       Podiatrist Pays $90,000 To Settle False Billin...
Date                                          January 3, 2023
Category                           Criminal and Civil Actions
Link        https://oig.hhs.gov/fraud/enforcement/podiatri...
Agency      U.S. Attorney’s Office, Southern District of T...
Name: 1533, dtype: object
                                               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...
```

- c. Test Partner’s Code (PARTNER 1)

```
# Get the data from Jan, 2021 to current
df_2021 = scrape_enforcement_actions(1, 2021)

# The number of enforcement action
print("Number of enforcement actions:", len(df_2021))

# Details of the earliest action since January 2021
earliest_action = df_2021.iloc[-1]
print("Earliest enforcement action:")
print(earliest_action)
```

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

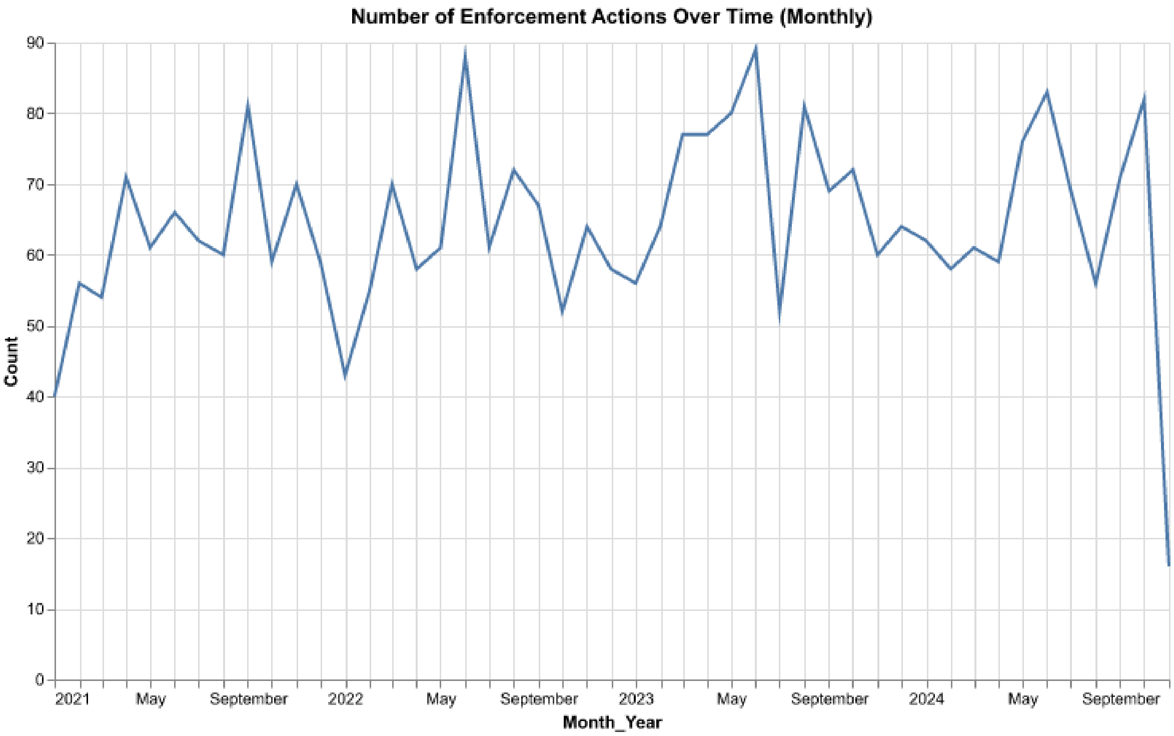
Step 3: Plot data based on scraped data

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

```
# Set month and year data
# Convert column 'Date' into datetime type and
df_2021['Date'] = pd.to_datetime(df_2021['Date'])
# Check whether there are NAs
na_count = df_2021['Date'].isna().sum()
print("Number of NA values in 'Date' column in DataFrame:", na_count)
# Create the colum for month + year
df_2021['YearMonth'] = df_2021['Date'].dt.to_period('M')

# Rearrange the data for the x-axis
# Count the number of monthly actions
monthly_counts = df_2021.groupby('YearMonth').size().reset_index(name =
 ↪ 'Count')
# Change the datatype for easy timeseries plotting
monthly_counts['YearMonth'] = monthly_counts['YearMonth'].dt.to_timestamp()
```

```
Number of NA values in 'Date' column in DataFrame: 0
```

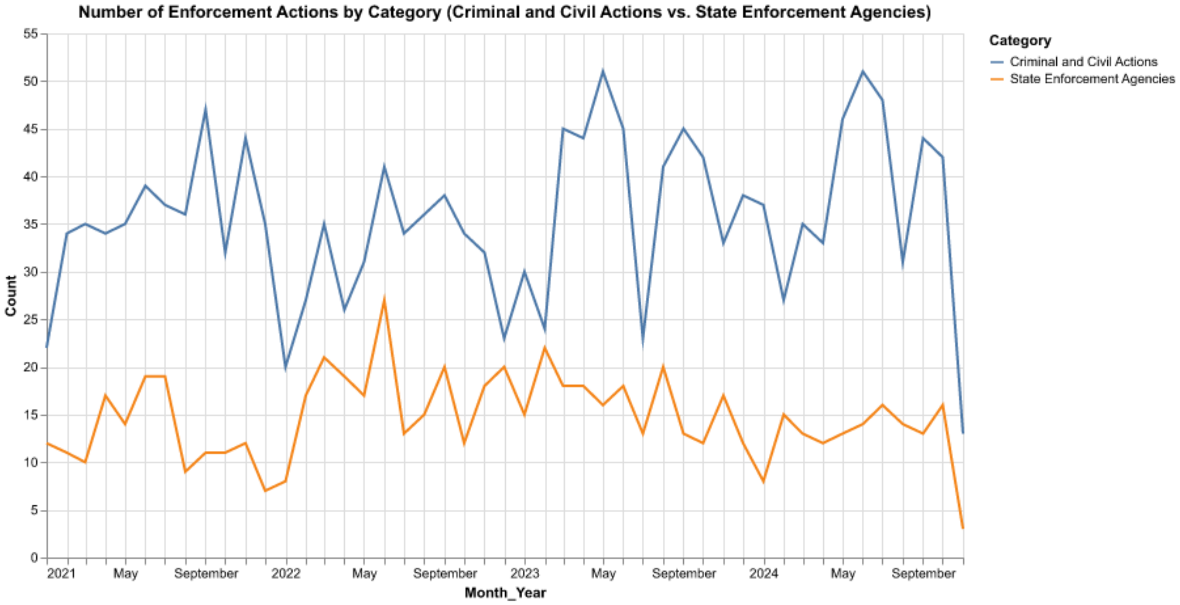
```
# Plot a line chart
line_chart_overall = alt.Chart(monthly_counts).mark_line().encode(
    x=alt.X('YearMonth:T', axis=alt.Axis(title='Month_Year',
 ↪  tickCount='month')),
    y='Count:Q'
).properties(
    title="Number of Enforcement Actions Over Time (Monthly)",
    width=700,
    height=400
)

line_chart_overall.display()
```



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

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

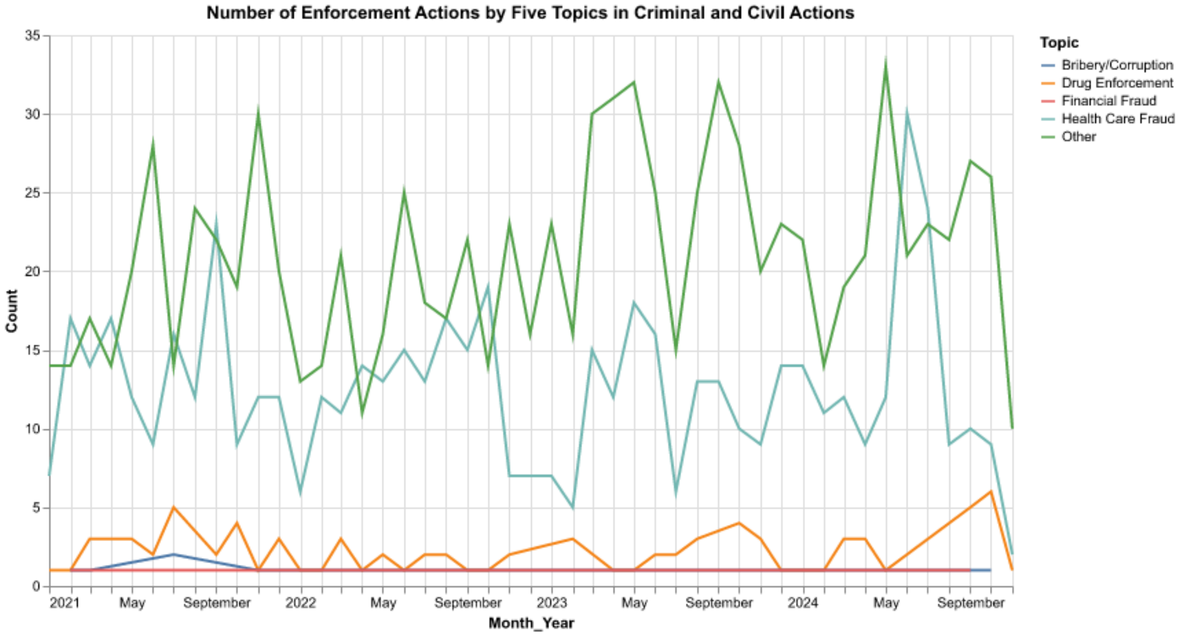
```
# Filter for the two categories; "Criminal and Civil Actions" and "State
 ↪  Enforcement Agencies"
filtered_df = df_2021[df_2021['Category'].isin(['Criminal and Civil Actions',
 ↪  'State Enforcement Agencies'])]
# Make sure the date is period type for aggregation
filtered_df['YearMonth'] = filtered_df['Date'].dt.to_period('M')

# Rearrange the data for plotting
# Group by YearMonth and Category
two_categories_counts = filtered_df.groupby(['YearMonth',
 ↪  'Category']).size().reset_index(name='Count')
# Count the occurrences and convert the date into timestamp for plotting of
 ↪  x-axis
two_categories_counts['YearMonth'] =
 ↪  two_categories_counts['YearMonth'].dt.to_timestamp()

# Plot the line chart for two categories
line_chart_category = alt.Chart(two_categories_counts).mark_line().encode(
    x=alt.X('YearMonth:T', axis=alt.Axis(title='Month_Year',
 ↪  tickCount='month')),
    y='Count:Q',
    color='Category:N',  # Different colors for each category
    tooltip=['YearMonth:T', 'Category:N', 'Count:Q']
).properties(
    title="Number of Enforcement Actions by Category (Criminal and Civil
 ↪  Actions vs. State Enforcement Agencies)",
    width=700,
    height=400
)

line_chart_category.display()
```



- based on five topics in “Criminal and Civil Actions” category: “Health Care Fraud”, “Financial Fraud”, “Drug Enforcement”, “Bribery/Corruption”, and “Other”

```
# Filter for "Criminal and Civil Actions" category
criminal_civic_df = df_2021[df_2021['Category'] == 'Criminal and Civil
 ↪  Actions']

# Define the keyword for each topic
topic_keywords = {
    'Health Care Fraud': ['health'],
    'Financial Fraud': ['financial'],
    'Drug Enforcement': ['drug'],
    'Bribery/Corruption': ['bribery', 'corruption']
}

# Assign titles to each topic based on the keywords
def assign_topic(title):
    title = title.lower()
    for topic, keywords in topic_keywords.items():
        if any(keyword in title for keyword in keywords):
            return topic
    return 'Other'

# Apply the function to create the 'Topic' column
criminal_civic_df['Topic'] = criminal_civic_df['Title'].apply(assign_topic)
```

```
# Convert 'Date' to period type for aggregation
criminal_civic_df['YearMonth'] = criminal_civic_df['Date'].dt.to_period('M')

# Group by YearMonth and Topic, then count the occurrences
topics_counts = criminal_civic_df.groupby(['YearMonth',
 ↪  'Topic']).size().reset_index(name='Count')
topics_counts['YearMonth'] = topics_counts['YearMonth'].dt.to_timestamp()  #
 ↪  Convert to timestamp for plotting

# Plot the line chart for five topics within "Criminal and Civil Actions"
line_chart_topics = alt.Chart(topics_counts).mark_line().encode(
    x=alt.X('YearMonth:T', axis=alt.Axis(title='Month_Year',
 ↪  tickCount='month')),
    y='Count:Q',
    color='Topic:N',
    tooltip=['YearMonth:T', 'Topic:N', 'Count:Q']
).properties(
    title="Number of Enforcement Actions by Five Topics in Criminal and Civil
 ↪  Actions",
    width=700,
    height=400
)

line_chart_topics.display()
```



Step 4: Create maps of enforcement activity

1. Map by State (PARTNER 1)

```
# Import shape file
states_gdf =
 ↪  gpd.read_file("C:\\Users\\sumos\\OneDrive\\   \\Harris\\2024 \\Python2\\PS\\PS5\\statedat
print(states_gdf.head())
```

```
  STATEFP   STATENS     AFFGEOID GEOID STUSPS           NAME LSAD  \
0      28  01779790  0400000US28    28     MS    Mississippi   00   
1      37  01027616  0400000US37    37     NC North Carolina   00   
2      40  01102857  0400000US40    40     OK       Oklahoma   00   
3      51  01779803  0400000US51    51     VA       Virginia   00   
4      54  01779805  0400000US54    54     WV  West Virginia   00   

          ALAND       AWATER  \
0  121533519481   3926919758   
1  125923656064  13466071395   
2  177662925723   3374587997   
3  102257717110   8528531774   
4   62266474513    489028543   
```

```
                                           geometry
0  MULTIPOLYGON (((-88.50297 30.21524, -88.49176 ...
1  MULTIPOLYGON (((-75.72681 35.93584, -75.71827 ...
2  POLYGON ((-103.00256 36.52659, -103.00219 36.6...
3  MULTIPOLYGON (((-75.74241 37.80835, -75.74151 ...
4  POLYGON ((-82.6432 38.16909, -82.643 38.16956,...
```

```
# Clean the name of states in the df_2021
# Subtract actions by state agency
state_agency_df = df_2021[df_2021['Agency'].str.contains("State of",
 ↪  na=False)].copy()

# Delete "Stat of" to get only the name of state
state_agency_df['State'] = state_agency_df['Agency'].str.replace("State of ",
 ↪  "", regex=False)

# Count the number of actions by state
state_counts =
 ↪  state_agency_df['State'].value_counts().reset_index(name='Count')
print(state_counts.head())
```

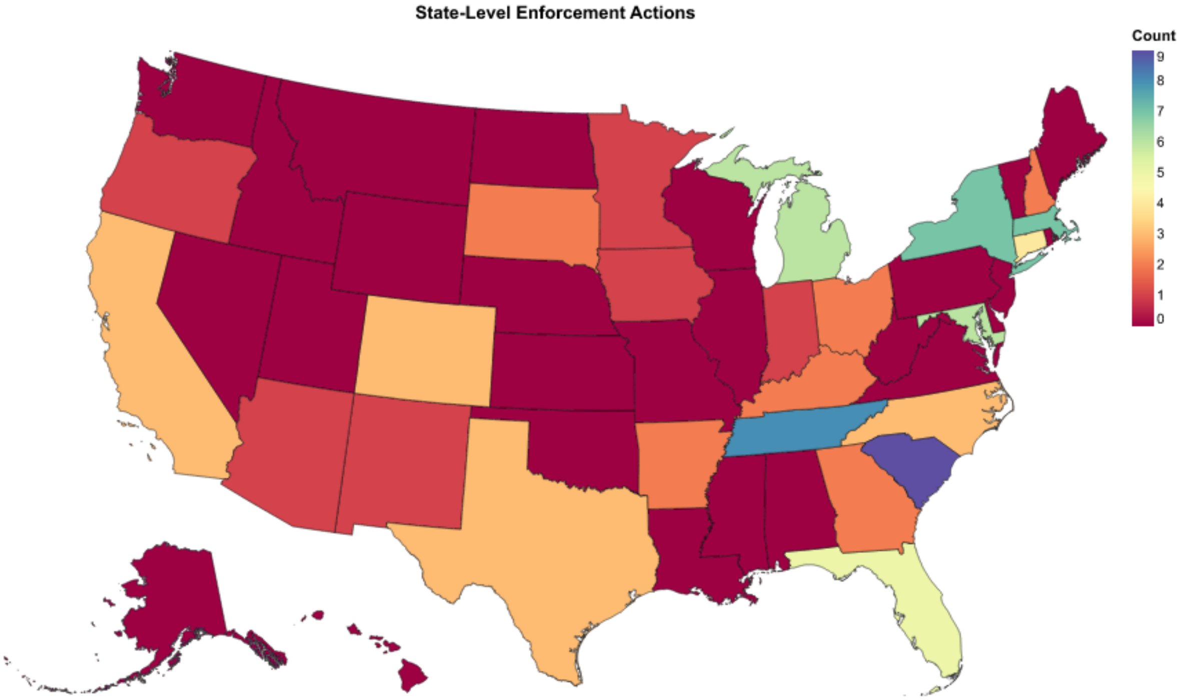
```
            State  Count
0  South Carolina      9
1       Tennessee      8
2        New York      7
3   Massachusetts      7
4        Michigan      6
```

```
# Merge the GeoDataFrame and enforcement action DF

# Merge state_gdf and state_counts with name
merged_state_gdf = states_gdf.merge(state_counts, left_on='NAME',
 ↪  right_on='State', how='left')

# Replace Na meaning no enforcement actions in that state  with 0
merged_state_gdf['Count'] = merged_state_gdf['Count'].fillna(0)
merged_state_gdf['Count'] = merged_state_gdf['Count'].astype(int)
print(merged_state_gdf[['NAME', 'Count']].head())
```

```
             NAME  Count
0     Mississippi      0
```

```
1  North Carolina      3
2        Oklahoma      0
3        Virginia      0
4   West Virginia      0
```

```
# Plot choropleth

state_choropleth = alt.Chart(merged_state_gdf).mark_geoshape(
    stroke='black',
    strokeWidth=0.5
).encode(
    color=alt.Color('Count:Q',
                    scale=alt.Scale(scheme='spectral'),  # Attribution; Ask
 ↪  ChatGPT color variation
                    legend=alt.Legend(tickCount=10)),
    tooltip=[alt.Tooltip('NAME:N', title='State'), alt.Tooltip('Count:Q',
 ↪  title='Enforcement Actions')]
).properties(
    title='State-Level Enforcement Actions',
    width=800,
    height=500
).project('albersUsa') # Attribution; Ask ChatGPT how to depict the whole US
 ↪  including Alaska and Hawaii

state_choropleth.display()
```



2. Map by District (PARTNER 2)

```
# Import shape file

districts_gdf =
 ↪  gpd.read_file("C:\\Users\\sumos\\OneDrive\\   \\Harris\\2024 \\Python2\\PS\\PS5\\district
print(districts_gdf.head())
```

```
  statefp                     judicial_d         aland        awater     state  
  \
0      21  Western District of Kentucky  4.970555e+10  1.651516e+09  Kentucky   
1      21  Eastern District of Kentucky  5.257394e+10  7.238213e+08  Kentucky   
2      18  Southern District of Indiana  5.824517e+10  5.941176e+08   Indiana   
3      01    Middle District of Alabama  3.412673e+10  5.472423e+08   Alabama   
4      01  Southern District of Alabama  6.235882e+10  3.052681e+09   Alabama   

          chief_judg          nominating  term_as_ch  shape_leng  \
0      Greg N. Stivers   Barack Obama (D)      2018.0   16.200585   
1         Danny Reeves  George W. Bush (R)      2019.0   13.514251   
```

```
2  Jane Magnus-Stinson    Barack Obama (D)      2016.0   14.956126
3    Emily Coody Marks    Donald Trump (R)      2019.0   10.235799
4        Kristi DuBose  George W. Bush (R)      2017.0   12.976906

   shape_area abbr district_n    shape__are    shape__len  \
0    5.216899  KYW          6  8.123902e+10  1.964255e+06
1    5.451047  KYE          6  8.547129e+10  1.654681e+06
2    6.137433  INS          7  9.818187e+10  1.887626e+06
3    3.858442  ALM         11  5.645450e+10  1.236201e+06
4    3.278871  ALS         11  4.772733e+10  1.567095e+06

                                            geometry
0  MULTIPOLYGON (((-89.48248 36.50214, -89.48543 ...
1  POLYGON ((-84.62012 39.07346, -84.60793 39.073...
2  POLYGON ((-85.86281 40.46476, -85.86212 40.406...
3  POLYGON ((-85.33828 33.49471, -85.33396 33.492...
4  MULTIPOLYGON (((-88.08682 30.25987, -88.07676 ...
```

```
# Clean the name of district in the df_2021

# Filter for the "District" in Agency column
district_level_df = df_2021[df_2021['Agency'].str.contains("District",
 ↪  na=False)].copy()
```

At first, we crease the subset to check the unique names in the 'Agency' column.

```
# Check the unique rows
unique_districts = district_level_df.groupby('Agency').first().reset_index()

print(unique_districts.head())
```

```
                                              Agency  \
0  2021; U.S. Attorney's Office, Northern Distric...
1   Attorney's Office, Northern District of Illinois
2          Attorney's Office, District of New Jersey
3        Attorney's Office, District of Rhode Island
4      Attorney's Office, Northern District of Texas

                                               Title        Date  \
0  Florida Counseling Center Owner And Provider S... 2021-11-19
1  Illinois Nurse Charged With Tampering With Mor... 2023-12-21
2  Monmouth County Doctor Charged with Accepting ... 2021-11-04
```

```
3  Providence Man Sentenced, Faces Deportation fo... 2022-03-18
4  Hospital to Pay More Than $3 Million to Settle... 2021-08-27

                  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 YearMonth
0  https://oig.hhs.gov/fraud/enforcement/florida-...   2021-11
1  https://oig.hhs.gov/fraud/enforcement/illinois...   2023-12
2  https://oig.hhs.gov/fraud/enforcement/monmouth...   2021-11
3  https://oig.hhs.gov/fraud/enforcement/providen...   2022-03
4  https://oig.hhs.gov/fraud/enforcement/hospital...   2021-08
```

We found in this dataframe that several words or pharese are added to some of the concrete district name in the ‘Agency’ column. Therefore, we need to remove these words. We checked every unnecessary word/phrase, and then define the function to remove them.

```
# Define the funtion to remove the unnecessary words

def clean_district_name(district_name):
    # Unnecessary words or phrases
    phrases_to_remove = [
        "U.S. Attorney's Office, ",
        "Attorney's Office, ",
        "Attorney’s Office, ",
        "2021; U.S.",
        "Connecticut Attorney General and U.S.",
        "Inspector General",
        "†††", "††",
        "June 28, 2024: ",
        "November 7, 2024; ",
        "U.S. Department of Justice and ",
        "U.S. ",
        "Attorney General, ",
        "AttorneyÍs Office, ",
        "Attorney’s Office ",
        "Attorney’s Office; "
    ]
    # Remove these words or phrases from district_name
    for phrase in phrases_to_remove:
        district_name = district_name.replace(phrase, "")
    # Remove the blank spaces before and after the district name
    return district_name.strip()
```

```
# Apply the function to the district_level_df for cleaning the 'Agency'
 ↪  Column
district_level_df['Cleaned_District'] =
 ↪  district_level_df['Agency'].apply(clean_district_name)

print(district_level_df[['Agency', 'Cleaned_District']].head())
```

```
                                              Agency  \
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...
5  U.S. Attorney's Office, Western District of Texas

              Cleaned_District
1            District of Idaho
2    District of Massachusetts
3  Eastern District of Virginia
4   Middle District of Florida
5    Western District of Texas
```

```
# Count the number of actions per Cleaned_District
district_counts =
 ↪  district_level_df['Cleaned_District'].value_counts().reset_index()
district_counts.columns = ['District', 'Count']
print(district_counts.head())
```

```
                       District  Count
0     District of Massachusetts     85
1        District of New Jersey     72
2     Southern District of Texas     60
3       District of Connecticut     57
4  Southern District of New York     53
```

Then, we found that within the "District" column, there are combined entries: "Southern District of Florida and Western District of Kentucky", "Southern District of Texas and Southern District of Illinois", and "Western District of Kentucky and Southern District of Florida". Each of these entries has a count of 1. By dividing them and counting the each district, we want to correct these by:

- Adding 2 counts to "Southern District of Florida" and 2 to "Western District of Kentucky"
- Adding 1 count each to "Southern District of Texas" and "Southern District of Illinois" Then, I will remove the rows containing the combined entries.

```
# Take a copy of 'district_counts' before making updates
original_district_counts = district_counts.copy()

# Manually update the counts in `district_counts`
district_counts.loc[district_counts['District'] == 'Southern District of
 ↪  Florida', 'Count'] += 2
district_counts.loc[district_counts['District'] == 'Western District of
 ↪  Kentucky', 'Count'] += 2
district_counts.loc[district_counts['District'] == 'Southern District of
 ↪  Texas', 'Count'] += 1
district_counts.loc[district_counts['District'] == 'Southern District of
 ↪  Illinois', 'Count'] += 1

# Remove rows with combined entries
combined_entries = [
    "Southern District of Florida and Western District of Kentucky",
    "Southern District of Texas and Southern District of Illinois",
    "Western District of Kentucky and Southern District of Florida"
]
district_counts =
 ↪  district_counts[~district_counts['District'].isin(combined_entries)]

# Confirm the final counts match between `original_district_counts` and
 ↪  `district_counts`
for district in ["Southern District of Florida", "Western District of
 ↪  Kentucky", "Southern District of Texas", "Southern District of
 ↪  Illinois"]:
    original_count =
 ↪  original_district_counts.loc[original_district_counts['District'] ==
 ↪  district, 'Count'].values[0]
    updated_count = district_counts.loc[district_counts['District'] ==
 ↪  district, 'Count'].values[0]
```

```
    print(f"{district} - Original Count: {original_count}, Updated Count:
     ↪  {updated_count}")

for entry in combined_entries:
    if entry in district_counts['District'].values:
        print(f"{entry} is found in District column.")
    else:
        print(f"{entry} is not found in District column.")
```

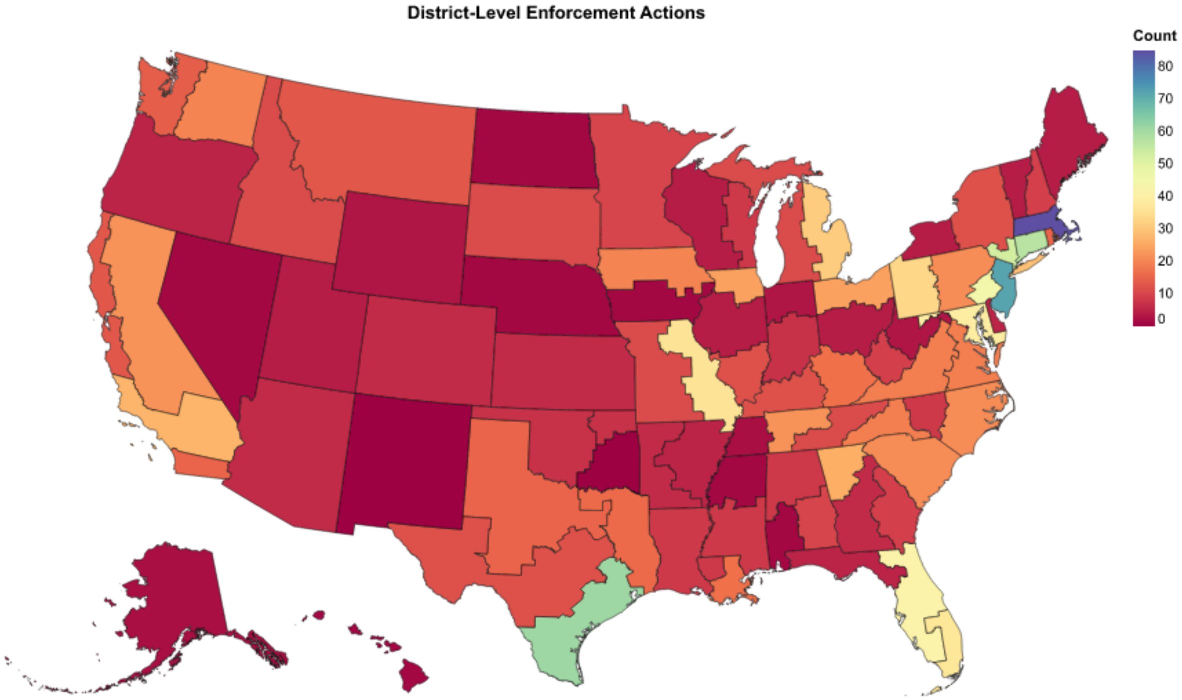
```
Southern District of Florida - Original Count: 35, Updated Count: 37
Western District of Kentucky - Original Count: 10, Updated Count: 12
Southern District of Texas - Original Count: 60, Updated Count: 61
Southern District of Illinois - Original Count: 11, Updated Count: 12
Southern District of Florida and Western District of Kentucky is not found in
District column.
Southern District of Texas and Southern District of Illinois is not found in
District column.
Western District of Kentucky and Southern District of Florida is not found in
District column.
```

Now that We can get the cleaned district, let's merge.

```
# Merge districts_gdf and district_counts by 'judicial_d' and 'District'
merged_districts_gdf = districts_gdf.merge(district_counts,
 ↪  left_on='judicial_d', right_on='District', how='left')

# Replace Na meaning no enforcement actions in that district  with 0
merged_districts_gdf['Count'] =
 ↪  merged_districts_gdf['Count'].fillna(0).astype(int)

print(merged_districts_gdf[['judicial_d', 'District', 'Count']].head())
```

```
                     judicial_d                      District  Count
0  Western District of Kentucky  Western District of Kentucky     12
1  Eastern District of Kentucky  Eastern District of Kentucky     17
2  Southern District of Indiana  Southern District of Indiana      7
3    Middle District of Alabama    Middle District of Alabama     10
4  Southern District of Alabama  Southern District of Alabama      1
```

```
# Plot choropleth
district_choropleth = alt.Chart(merged_districts_gdf).mark_geoshape(
    stroke='black',
    strokeWidth=0.5
).encode(
    color=alt.Color('Count:Q',
                    scale=alt.Scale(scheme='spectral'),
                    legend=alt.Legend(tickCount=10)),
    tooltip=[alt.Tooltip('judicial_d:N', title='District'),
 ↪  alt.Tooltip('Count:Q', title='Enforcement Actions')]
).properties(
    title='District-Level Enforcement Actions',
    width=800,
    height=500
).project('albersUsa')

district_choropleth.display()
```



Extra Credit

1. Merge zip code shapefile with population

```
# Import zip shape file
zip_gdf =
 ↪  gpd.read_file("C:\\Users\\sumos\\OneDrive\\   \\Harris\\2024 \\Python2\\PS\\PS4\\data\\gz
print(zip_gdf.head())

# Import population csv
population_df =
 ↪  pd.read_csv("C:\\Users\\sumos\\OneDrive\\   \\Harris\\2024 \\Python2\\PS\\PS5\\populatio
print(population_df.head())
```

```
           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
0  POLYGON ((-72.62734 42.16203, -72.62764 42.162...
1  POLYGON ((-72.95393 42.34379, -72.95385 42.343...
2  POLYGON ((-72.68286 42.37002, -72.68287 42.369...
3  POLYGON ((-72.39529 42.18476, -72.39653 42.183...
4  MULTIPOLYGON (((-72.39191 42.08066, -72.39077 ...
           GEO_ID                  NAME   P1_001N  Unnamed: 3
0       Geography  Geographic Area Name   !!Total         NaN
1  860Z200US00601           ZCTA5 00601     17242         NaN
2  860Z200US00602           ZCTA5 00602     37548         NaN
3  860Z200US00603           ZCTA5 00603     49804         NaN
4  860Z200US00606           ZCTA5 00606      5009         NaN
```

Looking at the population_df, we want to skip first row. In addition, we want to remove 'ZCTA5' in 'Geographic Area Name.'

```
# Skip the first row (index 0)
population_df =
 ↪  pd.read_csv("C:\\Users\\sumos\\OneDrive\\   \\Harris\\2024 \\Python2\\PS\\PS5\\populatio
 ↪  skiprows = 1)
# Replace 'ZCTA5' with '' in the column 'Geographic Area Name' and remove
 ↪  blank before zip code
```

```
population_df['ZIP'] = population_df['Geographic Area
 ↪  Name'].str.replace('ZCTA5', '', regex=False).str.strip()

print(population_df.head())
```

```
        Geography Geographic Area Name   !!Total  Unnamed: 3   ZIP
0  860Z200US00601          ZCTA5 00601     17242         NaN  00601
1  860Z200US00602          ZCTA5 00602     37548         NaN  00602
2  860Z200US00603          ZCTA5 00603     49804         NaN  00603
3  860Z200US00606          ZCTA5 00606      5009         NaN  00606
4  860Z200US00610          ZCTA5 00610     25731         NaN  00610
```

```
# Make sure the same datatype for merge
zip_gdf['ZCTA5'] = zip_gdf['ZCTA5'].astype(str)
population_df['ZIP'] = population_df['ZIP'].astype(str)

# Merge
merged_zip_population_gdf = zip_gdf.merge(population_df, left_on='ZCTA5',
 ↪  right_on='ZIP', how='left')
print(merged_zip_population_gdf.head())
```

```
           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       Geography  \
0  POLYGON ((-72.62734 42.16203, -72.62764 42.162...  860Z200US01040
1  POLYGON ((-72.95393 42.34379, -72.95385 42.343...  860Z200US01050
2  POLYGON ((-72.68286 42.37002, -72.68287 42.369...  860Z200US01053
3  POLYGON ((-72.39529 42.18476, -72.39653 42.183...  860Z200US01056
4  MULTIPOLYGON (((-72.39191 42.08066, -72.39077 ...  860Z200US01057

  Geographic Area Name   !!Total  Unnamed: 3   ZIP
0          ZCTA5 01040   38238.0         NaN  01040
1          ZCTA5 01050    2467.0         NaN  01050
2          ZCTA5 01053    2031.0         NaN  01053
3          ZCTA5 01056   21002.0         NaN  01056
4          ZCTA5 01057    8152.0         NaN  01057
```

2. Conduct spatial join

```
# Ensure both GeoDataFrames use the same CRS (coordinate reference system)
merged_zip_population_gdf =
↪  merged_zip_population_gdf.to_crs(districts_gdf.crs)

# Spatial join between zip population gdf and district gdf
sjoin_zip_districts_gdf = gpd.sjoin(merged_zip_population_gdf, districts_gdf,
↪  how='inner', predicate='intersects')

# Check the column name
print(sjoin_zip_districts_gdf.columns)
# print the result of spatial join for necessary info
print(sjoin_zip_districts_gdf[['ZCTA5', 'geometry', ' !!Total',
↪  'judicial_d']].head())
```

```
Index(['GEO_ID', 'ZCTA5', 'NAME', 'LSAD', 'CENSUSAREA', 'geometry',
       'Geography', 'Geographic Area Name', ' !!Total', 'Unnamed: 3', 'ZIP',
       'index_right', 'statefp', 'judicial_d', 'aland', 'awater', 'state',
       'chief_judg', 'nominating', 'term_as_ch', 'shape_leng', 'shape_area',
       'abbr', 'district_n', 'shape__are', 'shape__len'],
      dtype='object')
   ZCTA5                                           geometry   !!Total  \
0  01040  POLYGON ((-72.62734 42.16203, -72.62764 42.162...   38238.0
1  01050  POLYGON ((-72.95393 42.34379, -72.95385 42.343...    2467.0
2  01053  POLYGON ((-72.68286 42.37002, -72.68287 42.369...    2031.0
3  01056  POLYGON ((-72.39529 42.18476, -72.39653 42.183...   21002.0
4  01057  MULTIPOLYGON (((-72.39191 42.08066, -72.39077 ...    8152.0

                  judicial_d
0  District of Massachusetts
1  District of Massachusetts
2  District of Massachusetts
3  District of Massachusetts
4    District of Connecticut
```

```
# Convert the name of ' !!Total' to 'Population' for easy understanding
sjoin_zip_districts_gdf = sjoin_zip_districts_gdf.rename(columns={' !!Total':
↪  'Population'})
sjoin_zip_districts_gdf['Population'] =
↪  sjoin_zip_districts_gdf['Population'].fillna(0).astype(int)
```

```
print(sjoin_zip_districts_gdf[['Population', 'judicial_d']].head())
```

```
   Population                 judicial_d
0       38238  District of Massachusetts
1        2467  District of Massachusetts
2        2031  District of Massachusetts
3       21002  District of Massachusetts
4        8152    District of Connecticut
```

```
# Aggregate population data by district
district_population =
 ↪  sjoin_zip_districts_gdf.groupby('judicial_d')['Population'].sum().reset_index()
district_population.columns = ['District', 'Population']

print(district_population.head())
```

```
                         District  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
```

3. Map the action ratio in each district

At first, we calculate the action per capita.

```
# Merge the district_counts df which has the number of actions per district
 ↪  used in step3 and the population data per district
district_data = district_counts.merge(district_population,
 ↪  left_on='District', right_on='District', how='left')

# Check the Na values
na_count = district_data['Population'].isna().sum()
print(f"Number of NaN values in 'Population' column: {na_count}")

# Replace Na (meaning no population data in that district) with 0
district_data['Population'] =
 ↪  district_data['Population'].fillna(0).astype(int)
```

```
# Calculate the enforcement actions on a per-capita
district_data['Actions_Per_Capita'] = district_data.apply(
    lambda row: row['Count'] / row['Population'] if row['Population'] != 0
     ↪  else float('nan'),
    axis=1
) # Replace 0 in Population with NaN to avoid division by zero
```

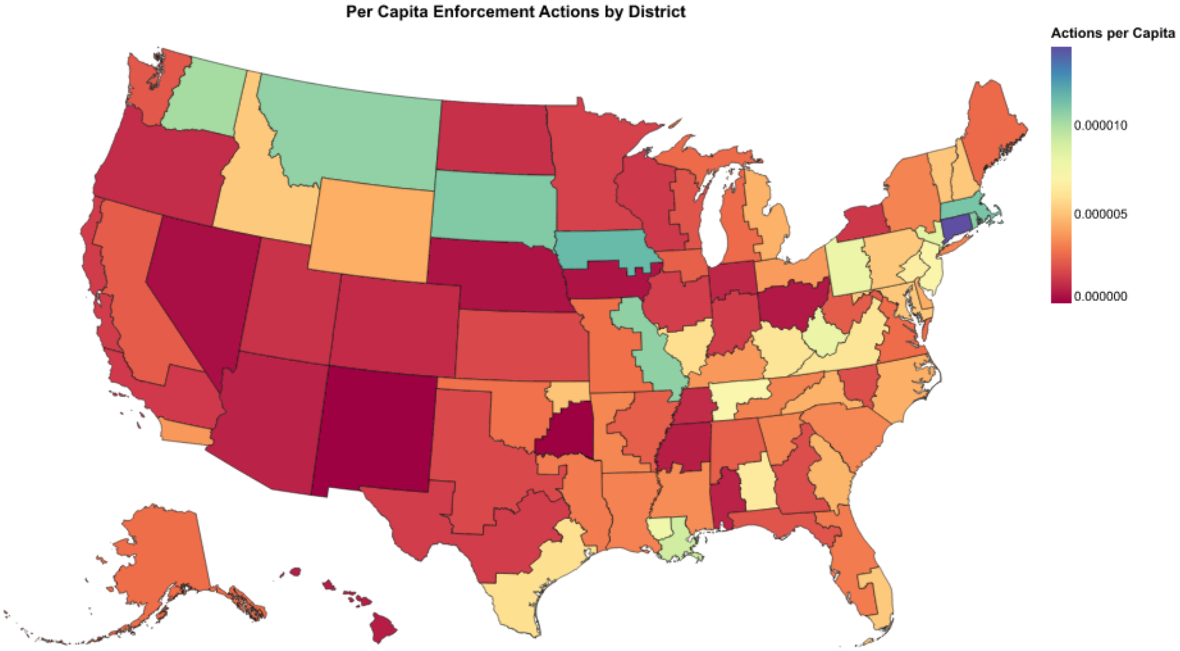
```
Number of NaN values in 'Population' column: 5
```

Then, create the GeoDataFrame for plotting.

```
# Merge
action_percapita_gdf = districts_gdf.merge(district_data,
 ↪  left_on='judicial_d', right_on='District', how='left')

# Replace Na with 0
action_percapita_gdf['Actions_Per_Capita'] =
 ↪  action_percapita_gdf['Actions_Per_Capita'].fillna(0)
```

Finally, plot the choropleth map

```
# Plot choropleth
choropleth_per_capita = alt.Chart(action_percapita_gdf).mark_geoshape(
    stroke='black',
    strokeWidth=0.5
).encode(
    color=alt.Color('Actions_Per_Capita:Q',
                    scale=alt.Scale(scheme='spectral'),
                    legend=alt.Legend(title="Actions per Capita")),
    tooltip=[alt.Tooltip('judicial_d:N', title='District'),
             alt.Tooltip('Actions_Per_Capita:Q', title='Actions per Capita')]
).properties(
    title='Per Capita Enforcement Actions by District',
    width=800,
    height=500
).project('albersUsa')

choropleth_per_capita.display()
```


Per Capita Enforcement Actions by District
Actions per Capita
0.000010
0.000005
0.000000