

**title**

author

Invalid Date

**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): Guillermina Marto - gmarto
  - Partner 2 (name and cnet ID): Alejandra Silva - aosilva
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: **\*\*GM\*\* \*\*\_\_\*\***
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

```

import pandas as pd
import altair as alt
import time

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

```

```

RendererRegistry.enable('png')

```

## Step 1: Develop initial scraper and crawler

### 1. Scraping (PARTNER 1)

```

import requests
from bs4 import BeautifulSoup

# URL of the HHS OIG Enforcement Actions page
url = "https://oig.hhs.gov/fraud/enforcement/"

# Send a GET request to the page
response = requests.get(url)
response.raise_for_status()

# Parse the HTML with BeautifulSoup
soup = BeautifulSoup(response.text, "html.parser")

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

# Find all enforcement action items
for item in soup.select("li.usa-card"):
    # Extract the title
    title_element = item.select_one("h2.usa-card__heading a")
    if title_element:
        titles.append(title_element.get_text(strip=True))
        links.append("https://oig.hhs.gov" + title_element["href"])

```

```

else:
    titles.append(None)
    links.append(None)

# Extract the date
date_element = item.select_one("span.text-base-dark")
dates.append(date_element.get_text(strip=True) if date_element else None)

# Extract the category
category_element = item.select_one("ul li.usa-tag")
categories.append(category_element.get_text(strip=True) if
↪ category_element else None)

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

# Save the DataFrame to a CSV file
data.to_csv("hhs_oig_enforcement_actions.csv", index=False)

# Display the DataFrame
print(data.head())

```

	Title	Date \
0	Former Arlington Resident Sentenced To Prison ...	November 7, 2024
1	Paroled Felon Sentenced To Six Years For Fraud...	November 7, 2024
2	Former Licensed Counselor Sentenced For Defrau...	November 6, 2024
3	Macomb County Doctor And Pharmacist Agree To P...	November 4, 2024
4	Rocky Hill Pharmacy And Its Owners Indicted Fo...	November 4, 2024

	Category \
0	Criminal and Civil Actions
1	Criminal and Civil Actions
2	Criminal and Civil Actions
3	Criminal and Civil Actions
4	Criminal and Civil Actions

Link

```

0 https://oig.hhs.gov/fraud/enforcement/former-a...
1 https://oig.hhs.gov/fraud/enforcement/paroled-...
2 https://oig.hhs.gov/fraud/enforcement/former-l...
3 https://oig.hhs.gov/fraud/enforcement/macomb-c...
4 https://oig.hhs.gov/fraud/enforcement/rocky-hi...

```

## 2. Crawling (PARTNER 1)

```

import time

# Initialize an empty list to store agency names
agencies = []

# Loop through each link in the DataFrame to extract agency information
for link in data["Link"]:
    action_response = requests.get(link)
    action_response.raise_for_status()
    action_soup = BeautifulSoup(action_response.text, "html.parser")

    # Extract agency information - usually the second <li> element
    agency_element = action_soup.select("ul.usa-list--unstyled li")
    if len(agency_element) > 1:
        agency_text = agency_element[1].get_text(strip=True)
        agency_name = agency_text.split(":", 1)[-1].strip() # Extract text
        ↪ after "Agency:"
        agencies.append(agency_name)
    else:
        agencies.append(None)

    # Optional: Pause to avoid too many rapid requests
    time.sleep(1)

# Update the DataFrame with the new 'Agency' column
data["Agency"] = agencies

# Print the first few rows to verify
print(data.head())

```

	Title	Date \
0	Former Arlington Resident Sentenced To Prison ...	November 7, 2024
1	Paroled Felon Sentenced To Six Years For Fraud...	November 7, 2024

2	Former Licensed Counselor Sentenced For Defrau...	November 6, 2024
3	Macomb County Doctor And Pharmacist Agree To P...	November 4, 2024
4	Rocky Hill Pharmacy And Its Owners Indicted Fo...	November 4, 2024

	Category \
0	Criminal and Civil Actions
1	Criminal and Civil Actions
2	Criminal and Civil Actions
3	Criminal and Civil Actions
4	Criminal and Civil Actions

	Link \
0	<a href="https://oig.hhs.gov/fraud/enforcement/former-a...">https://oig.hhs.gov/fraud/enforcement/former-a...</a>
1	<a href="https://oig.hhs.gov/fraud/enforcement/paroled-...">https://oig.hhs.gov/fraud/enforcement/paroled-...</a>
2	<a href="https://oig.hhs.gov/fraud/enforcement/former-l...">https://oig.hhs.gov/fraud/enforcement/former-l...</a>
3	<a href="https://oig.hhs.gov/fraud/enforcement/macomb-c...">https://oig.hhs.gov/fraud/enforcement/macomb-c...</a>
4	<a href="https://oig.hhs.gov/fraud/enforcement/rocky-hi...">https://oig.hhs.gov/fraud/enforcement/rocky-hi...</a>

	Agency
0	U.S. Attorney's Office, Eastern District of Vi...
1	U.S. Attorney's Office, Middle District of Flo...
2	U.S. Attorney's Office, Western District of Texas
3	U.S. Attorney's Office, Eastern District of Mi...
4	U.S. Attorney's Office, Eastern District of Te...

## Step 2: Making the scraper dynamic

### 1. Turning the scraper into a function

- a. Pseudo-Code (PARTNER 2)

1. Define the Function `scrape_enforcement_actions(start_month, start_year)`:

- Inputs:
  - `start_month`: The month to start scraping from (1 to 12).
  - `start_year`: The year to start scraping from (should be  $\geq 2013$ ).

2. Initial Validity Check:

- If `start_year` is less than 2013, **print a message**: “Please enter a year  $\geq 2013$ , as data before 2013 is not available.”
- **Exit the function** if the condition is not met.

3. Setup:

- Define the **base URL** for the HHS OIG enforcement actions page.
  - **Initialize lists** to store scraped data for each action (e.g., **titles**, **dates**, **categories**, **links**).
  - Set **page = 1** to start from the first page of the site.
  - Get the **current date** to compare against as the end condition.
4. **Loop Through Pages Until Date Condition is Met:**
- Start a **while** loop:
    - **Construct URL** with the current **page** number.
    - **Request the page** content and parse it with BeautifulSoup.
    - **Find all enforcement action items** on the page (e.g., by selecting specific HTML elements).
    - **Check for End of Pages:**
      - \* If no items are found (end of content), **break** the loop.
5. **Extract Data for Each Action:**
- For each item on the page:
    - Extract and **store the title**, **link**, **date**, and **category**.
    - **Parse the date** to check if it's within the specified **start\_year** and **start\_month**.
      - \* If the action's date is earlier than the specified date, **break** the loop to stop scraping.
6. **Append Data to Lists:**
- Append the extracted data for title, date, category, and link to the respective lists.
7. **Move to the Next Page:**
- Increment **page** by 1 to load the next set of actions.
  - Use **time.sleep(1)** to pause between requests to avoid overloading the server.
8. **Create a DataFrame and Save to CSV:**
- After exiting the loop, **create a DataFrame** from the lists.
  - **Save** the DataFrame to a CSV file with a name format like **enforcement\_actions\_<year>\_<month>.csv**.
9. **End Function:**
- Print a confirmation message that data has been saved.

This structure uses a **while** loop to handle pagination dynamically and ensures the scraper stops early if it encounters data before the specified start date.

- b. Create Dynamic Scraper (PARTNER 2) The earliest was on January 3, 2023 Category Criminal and Civil Actions Link <https://oig.hhs.gov/fraud/enforcement/podiatry...> Agency U.S. Attorney's Office, Southern District of T...

```

from datetime import datetime
import requests
from bs4 import BeautifulSoup
import pandas as pd

def scrape_enforcement_actions(start_month, start_year):
    start_date = datetime(start_year, start_month, 1)

    # Check if the input year is valid (>= 2013)
    if start_year < 2013:
        print("Please enter a year >= 2013, as data before 2013 is not
        ↪ available.")
        return

    base_url = "https://oig.hhs.gov/fraud/enforcement/"
    all_titles, all_dates, all_categories, all_links, all_agencies = [], [],
    ↪ [], [], []
    page = 1
    stop_scraping = False

    while not stop_scraping:
        response = requests.get(f"{base_url}?page={page}")
        response.raise_for_status()
        soup = BeautifulSoup(response.text, "html.parser")

        for item in soup.select("li.usa-card"):
            title_element = item.select_one("h2.usa-card__heading a")
            title = title_element.get_text(strip=True) if title_element else
            ↪ None
            link = "https://oig.hhs.gov" + title_element["href"] if
            ↪ title_element else None

            date_element = item.select_one("span.text-base-dark")
            date_text = date_element.get_text(strip=True) if date_element
            ↪ else None
            date_obj = datetime.strptime(date_text, "%B %d, %Y") if date_text
            ↪ else None

            # Stop if the date is before start_date
            if date_obj and date_obj < start_date:
                stop_scraping = True
                break

```

```

        category_element = item.select_one("ul li.usa-tag")
        category = category_element.get_text(strip=True) if
↪ category_element else None

        all_titles.append(title)
        all_dates.append(date_text)
        all_categories.append(category)
        all_links.append(link)

        # Optional: Pause to avoid too many rapid requests
        time.sleep(0.05)

    page += 1

# Extract agency information from each link
for link in all_links:
    action_response = requests.get(link)
    action_response.raise_for_status()
    action_soup = BeautifulSoup(action_response.text, "html.parser")

    agency_element = action_soup.select("ul.usa-list--unstyled li")
    if len(agency_element) > 1:
        agency_text = agency_element[1].get_text(strip=True)
        all_agencies.append(agency_text.split(":", 1)[-1].strip())
    else:
        all_agencies.append(None)

# Create DataFrame
data = pd.DataFrame({
    "Title": all_titles,
    "Date": all_dates,
    "Category": all_categories,
    "Link": all_links,
    "Agency": all_agencies
})

# Save the DataFrame to a CSV file
filename = f"enforcement_actions_{start_year}_{start_month}.csv"
data.to_csv(filename, index=False)
print(f"Data saved to {filename}")

```



```

    return data

# Example usage:
data = scrape_enforcement_actions(1, 2023)
#print(data.head())

print(f"Total enforcement actions collected: {len(data)}")
data["Date"] = pd.to_datetime(data["Date"], errors="coerce")
earliest_action = data.sort_values("Date").iloc[0]
print("Earliest enforcement action collected:")
print(earliest_action)

```

```

Data saved to enforcement_actions_2023_1.csv
Total enforcement actions collected: 1510
Earliest enforcement action collected:
Title      Podiatrist Pays $90,000 To Settle False Billin...
Date              2023-01-03 00:00:00
Category              Criminal and Civil Actions
Link      https://oig.hhs.gov/fraud/enforcement/podiatr...
Agency      U.S. Attorney's Office, Southern District of T...
Name: 1509, dtype: object

```

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

```

# Collect data from January 2021 to today
data = scrape_enforcement_actions(1, 2021)

# Check the number of enforcement actions collected

print(f"Total enforcement actions collected: {len(data)}")

# Display the earliest enforcement action by date
data["Date"] = pd.to_datetime(data["Date"], errors="coerce")
earliest_action = data.sort_values("Date").iloc[0]
print("Earliest enforcement action collected:")
print(earliest_action)

```

```

Data saved to enforcement_actions_2021_1.csv
Total enforcement actions collected: 2998
Earliest enforcement action collected:
Title      The United States And Tennessee Resolve Claims...
Date              2021-01-04 00:00:00

```

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: 2997, dtype: object

### Step 3: Plot data based on scraped data

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

```
import altair as alt

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

# Convert 'Date' column to datetime format to enable time-based operations
data["Date"] = pd.to_datetime(data["Date"], errors="coerce")

# Drop rows with NaT in Date (if any) after conversion
data = data.dropna(subset=["Date"])

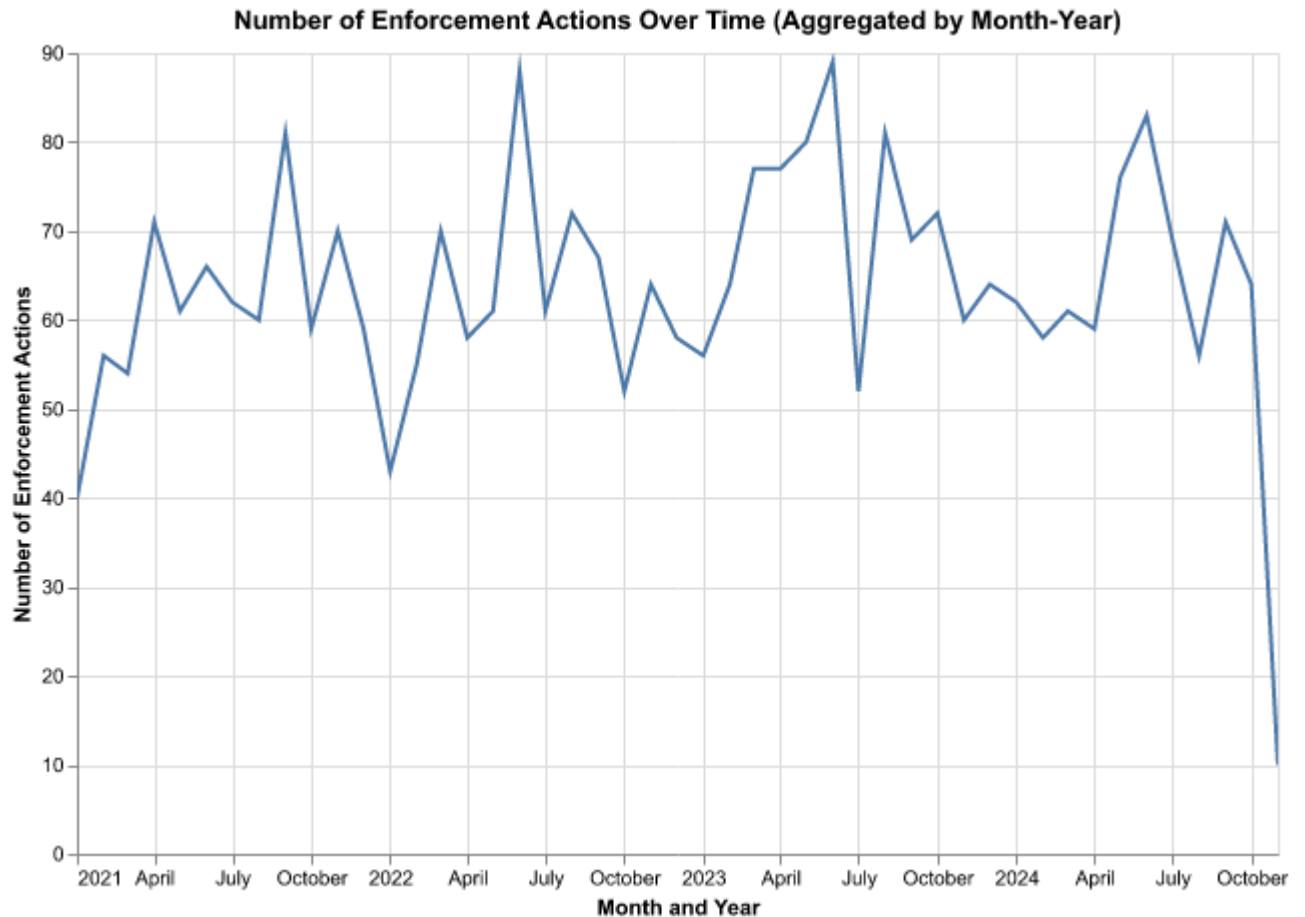
# Extract month-year for aggregation
data["Year_Month"] = data["Date"].dt.to_period("M")

# Group by month-year and count the number of actions
monthly_counts = data.groupby("Year_Month").size().reset_index(name="Count")

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

# Plot the line chart using Altair
chart = alt.Chart(monthly_counts).mark_line().encode(
    x=alt.X("Year_Month:T", title="Month and Year"),
    y=alt.Y("Count:Q", title="Number of Enforcement Actions"),
    tooltip=["Year_Month:T", "Count:Q"]
).properties(
    title="Number of Enforcement Actions Over Time (Aggregated by
↪ Month-Year)",
    width=600,
    height=400
).interactive()

chart
```



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

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

```
# Filter for specific categories
categories_of_interest = ["Criminal and Civil Actions", "State Enforcement
↵ Agencies"]
filtered_data = data[data["Category"].isin(categories_of_interest)]

# Group by month-year and category, then count the number of actions
monthly_category_counts = (
    filtered_data.groupby(["Year_Month", "Category"])
    .size()
```

```

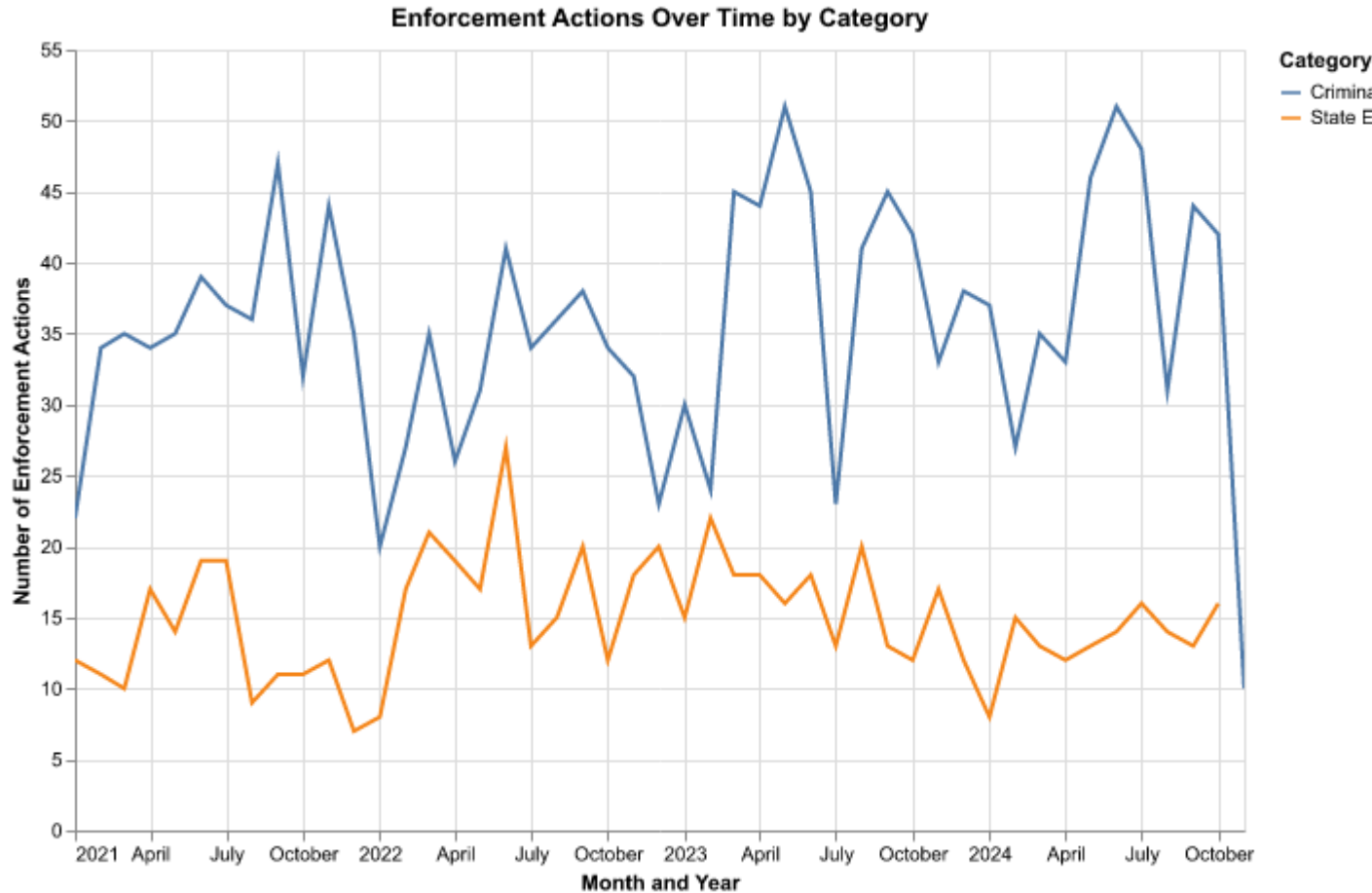
        .reset_index(name="Count")
    )

    # Convert Year_Month to datetime for plotting
    monthly_category_counts["Year_Month"] =
        ↪ monthly_category_counts["Year_Month"].dt.to_timestamp()

    # Plot the line chart by category
    category_chart = alt.Chart(monthly_category_counts).mark_line().encode(
        x=alt.X("Year_Month:T", title="Month and Year"),
        y=alt.Y("Count:Q", title="Number of Enforcement Actions"),
        color="Category",
        tooltip=["Year_Month:T", "Category", "Count"]
    ).properties(
        title="Enforcement Actions Over Time by Category",
        width=600,
        height=400
    ).interactive()

    category_chart

```



- based on five topics

```
# Define five specific topics to categorize actions
topics_of_interest = [
    "Criminal and Civil Actions",
    "State Enforcement Agencies",
    "Fraud Self-Disclosures",
    "Grant and Contractor Fraud Self-Disclosures",
    "Medicare Fraud Strike Force"
]

# Filter data for these topics
topic_filtered_data = data[data["Category"].isin(topics_of_interest)]

# Group by month-year and topic, then count the actions
monthly_topic_counts = (
```

```

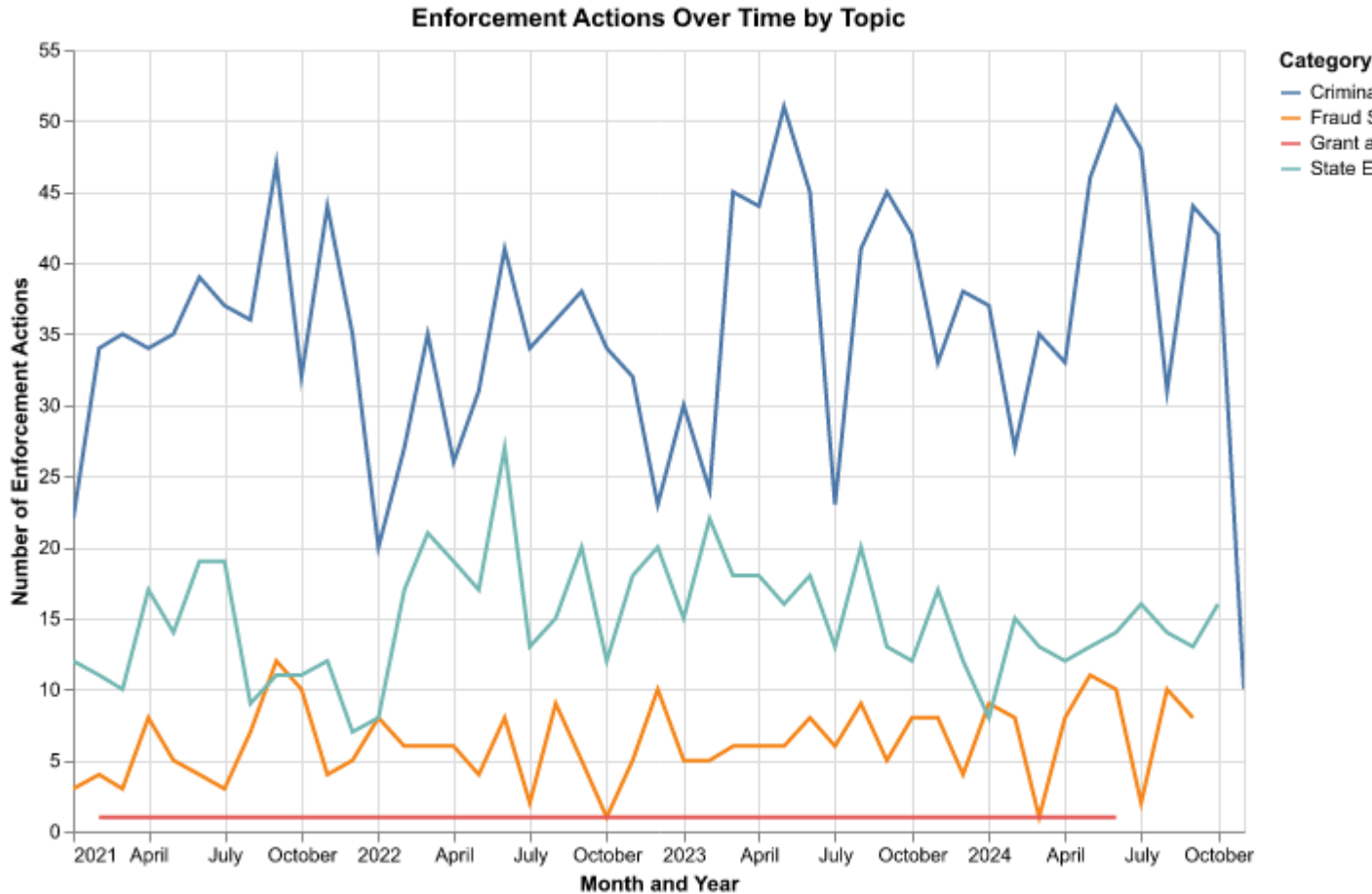
    topic_filtered_data.groupby(["Year_Month", "Category"])
        .size()
        .reset_index(name="Count")
)

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

# Plot the line chart by topic
topic_chart = alt.Chart(monthly_topic_counts).mark_line().encode(
    x=alt.X("Year_Month:T", title="Month and Year"),
    y=alt.Y("Count:Q", title="Number of Enforcement Actions"),
    color="Category",
    tooltip=["Year_Month:T", "Category", "Count"]
).properties(
    title="Enforcement Actions Over Time by Topic",
    width=600,
    height=400
).interactive()

topic_chart

```



#### Step 4: Create maps of enforcement activity

1. Map by State (PARTNER 1)
2. Map by District (PARTNER 2)

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

# Load the enforcement actions data
data = pd.read_csv("enforcement_actions_2021_1.csv")

# Filter for US Attorney District-level agencies by looking for "District" in
→ the Agency column
```

```

district_data = data[data["Agency"].str.contains("District",
↪ na=False)].copy()

# Keep only the last three words in the District column of enforcement data
district_data["District"] = district_data["Agency"].apply(lambda x: '
↪ '.join(x.split()[-3:]))

# Count the number of enforcement actions per district
district_counts = district_data["District"].value_counts().reset_index()
district_counts.columns = ["District", "Count"]

# Update the path to the .shp file with either double backslashes or a raw
↪ string
shapefile_path = r'US Attorney Districts Shapefile
↪ simplified_20241105\geo_export_863694bc-2e85-424f-96a2-1598bf583e4c.shp'

# Load the shapefile using GeoPandas
district_shapefile = gpd.read_file(shapefile_path)

# Remove the first word from each entry in the judicial_d column
district_shapefile["cleaned_judicial_d"] =
↪ district_shapefile["judicial_d"].apply(lambda x: ' '.join(x.split()[1:]))

# Merge the district counts with the shapefile data on 'cleaned_judicial_d'
↪ and 'District'
district_map_data = district_shapefile.merge(district_counts,
↪ left_on="cleaned_judicial_d", right_on="District", how="left")

# Fill NaN values in the 'Count' column with 0 for districts with no actions
district_map_data["Count"].fillna(0, inplace=True)

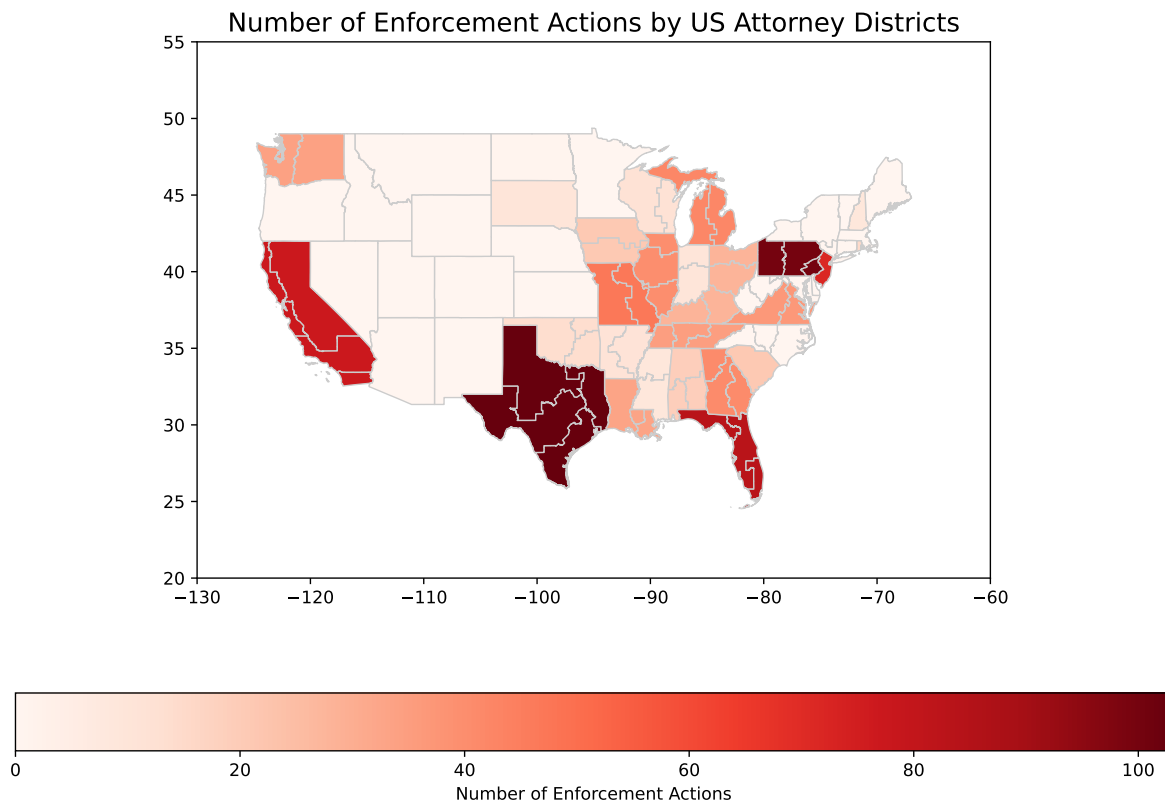
# Plot the choropleth map with focus on continental U.S.
fig, ax = plt.subplots(1, 1, figsize=(12, 8))
district_map_data.plot(column="Count", cmap="Reds", linewidth=0.8, ax=ax,
↪ edgecolor="0.8", legend=True,
                        legend_kwds={"label": "Number of Enforcement Actions",
                                      "orientation": "horizontal"})

# Set axis limits to focus on the continental U.S.
ax.set_xlim([-130, -60]) # Longitude range for continental U.S.
ax.set_ylim([20, 55])    # Latitude range for continental U.S.

```



```
ax.set_title("Number of Enforcement Actions by US Attorney Districts",  
            ↪  fontsize=15)  
plt.show()
```



### Extra Credit

1. Merge zip code shapefile with population
2. Conduct spatial join
3. Map the action ratio in each district