

PS5

Author

Mauricio Paniagua & Mahnoor Arif

Published

November 10, 2024

Due 11/9 at 5:00PM Central. Worth 100 points paper, scissors, rock to determine who goes first. Call that person *Partner 1*. - Partner 1: Mauricio Paniagua, mauriciop1 - Partner 2: Mahnoor Arif, mahnoorarif 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: ** __ ** __ ** 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: 1 late coin used per perosn left after submission: 2, 3 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
import pandas as pd

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

```

# Send a GET request to fetch the page content
response = requests.get(url)
soup = BeautifulSoup(response.text, 'html.parser') # Using the default parser

# Lists to hold data
titles, dates, categories, links = [], [], [], []

# Find enforcement action entries
for item in soup.select('h2.usa-card__heading'):
    # Extract title
    title = item.select_one('a').text.strip()

    # Extract and complete the link
    link = "https://oig.hhs.gov" + item.select_one('a')['href']

    # Extract category
    category = item.find_next('li', class_='display-inline-block usa-tag text-no-lowercase text-base-darkest bg-base-lightest margin-right-1').text.strip()

    # Extract date
    date = item.find_next('span', class_='text-base-dark padding-right-105').text.strip()

    # Append to lists
    titles.append(title)
    links.append(link)
    categories.append(category)
    dates.append(date)

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

# Show the first few rows
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 \	Link
0	Criminal and Civil Actions	https://oig.hhs.gov/fraud/enforcement/pharmaci...
1	Criminal and Civil Actions	https://oig.hhs.gov/fraud/enforcement/boise-nu...
2	Criminal and Civil Actions	https://oig.hhs.gov/fraud/enforcement/former-t...
3	Criminal and Civil Actions	https://oig.hhs.gov/fraud/enforcement/former-a...
4	Criminal and Civil Actions	https://oig.hhs.gov/fraud/enforcement/paroled-...

2. Crawling (PARTNER 1)

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

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

# Send a GET request to fetch the page content
response = requests.get(url)
soup = BeautifulSoup(response.text, 'html.parser') # Use 'html.parser' instead of 'lxml'

# Lists to hold data
titles, dates, categories, links, agencies = [], [], [], [], []

# Find enforcement action entries
for item in soup.select('h2.usa-card__heading'):
    # Extract title
    title = item.select_one('a').text.strip()

    # Extract and complete the link
    link = "https://oig.hhs.gov" + item.select_one('a')['href']

    # Extract category
    category = item.find_next('li', class_='display-inline-block usa-tag text-no-lowercase text-base-darkest bg-base-lightest margin-right-1').text.strip()

    # Extract date
    date = item.find_next('span', class_='text-base-dark padding-right-105').text.strip()
```

```

# Visit the link to extract the agency
enforcement_page = requests.get(link)
enforcement_soup = BeautifulSoup(enforcement_page.text, 'html.parser')

# Extract agency name (based on the example, you may need to adjust the selector)
agency_tag = enforcement_soup.select('div.content p:nth-of-type(3)')
agency_name = agency_tag[0].text.strip() if agency_tag else "Unknown"

# Append to lists
titles.append(title)
links.append(link)
categories.append(category)
dates.append(date)
agencies.append(agency_name)

# Create a DataFrame with the agency name included
df = pd.DataFrame({
    "Title": titles,
    "Date": dates,
    "Category": categories,
    "Link": links,
    "Agency": agencies
})

# Show the first few rows
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	Agency
0	https://oig.hhs.gov/fraud/enforcement/pharmaci...	Unknown
1	https://oig.hhs.gov/fraud/enforcement/boise-nu...	Unknown
2	https://oig.hhs.gov/fraud/enforcement/former-t...	Unknown
3	https://oig.hhs.gov/fraud/enforcement/former-a...	Unknown
4	https://oig.hhs.gov/fraud/enforcement/paroled-...	Unknown

Step 2: Making the scraper dynamic

1. Turning the scraper into a function

- a. Pseudo-Code (PARTNER 2)
 1. Defining a function with Year and Month as key inputs
 2. Validate Year:
If year < 2013, print a reminder and exit.
 3. Set Base URL:
Define the URL 'https://oig.hhs.gov/fraud/enforcement/'.
 4. **oop Through Pages:**
Start at page = 1.
If the year > 2013 continue until no more enforcement actions are found until the current date.
 5. Scrape Data:
 - Extract: • Date, Title of the enforcement action, Category (e.g, "Criminal and Civil Actions"), and link for each agency
 6. Rate Limiting: Include a time.sleep(1) pause after each page request to add a one-second delay between requests.
 7. Data Storage: Accumulate the scraped data, convert it into a DataFrame, and save it as a CSV file.
 8. Return the DataFrame.
- b. Create Dynamic Scraper (PARTNER 2)

b. Create Dynamic Scraper (PARTNER 2)

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

#def scrape_enforcement_actions(year, month):
#    # Check year validity
#    if year < 2013:
#        print("Please restrict the year to >= 2013.")
```

```
    return

# Base URL and setup
base_url = "https://oig.hhs.gov/fraud/enforcement/"
page = 1
all_data = []
target_date = datetime(year, month, 1)

while True:
    print(f"Scraping page {page}...")
    url = f"{base_url}?page={page}"
    response = requests.get(url)

    # Check if the response is empty or redirected
    if response.status_code != 200:
        print(f"Failed to retrieve page {page}, status code: {response.status_code}")
        break

    soup = BeautifulSoup(response.text, 'html.parser')

    # Print a sample of the HTML for debugging
    print(soup.prettify()[:1000]) # Print the first 1000 characters of the HTML

    # Select all action cards
    action_cards = soup.select('.usa-card')

    # Debugging: check if the action cards are being found
    if not action_cards:
        print("No actions found on this page. Exiting loop.")
        break
    else:
        print(f"Found {len(action_cards)} actions on page {page}.")

    # Process each action card in one pass
    for card in action_cards:
        title = card.select_one('.usa-card__heading').get_text(strip=True) if card.select_one('.usa-card__heading') else "No title found"
        link_elem = card.select_one('.usa-card__heading a')
        link = link_elem['href'] if link_elem else "No link found"
        date_elem = card.select_one('.text-base-dark')
        date_text = date_elem.get_text(strip=True) if date_elem else "No date found"

        if date_text != "No date found":
            date = datetime.strptime(date_text, '%B %d, %Y')
            if date < target_date:
```

```

        print(f"Reached actions from before {target_date.strftime('%B %Y')}. Stopping scrape.")
        break

# Initialize default values for category and agency
category = "Unknown Category"
agency = "No agency found"

# Skip action page request if link not found
if link != "No link found":
    response_action = requests.get(f"https://oig.hhs.gov{link}")
    action_soup = BeautifulSoup(response_action.text, 'html.parser')

    h2_tag = action_soup.find('h2', class_='font-heading-lg')
    if h2_tag:
        ul_tag = h2_tag.find_next('ul', class_='usa-list--unstyled')
        if ul_tag:
            li_tags = ul_tag.find_all('li')
            if len(li_tags) > 1 and 'Agency:' in li_tags[1].text:
                agency = li_tags[1].text.split('Agency:')[1].strip()
            if len(li_tags) > 2 and 'Enforcement Types:' in li_tags[2].text:
                category = li_tags[2].text.split('Enforcement Types:')[1].strip()

# Append the collected data
all_data.append([title, date_text, category, link, agency])

# Break the loop if we've hit the target date
if date_text != "No date found" and date < target_date:
    break

# Go to the next page and wait
page += 1
time.sleep(1)

# Convert collected data into DataFrame and save if data exists
if all_data:
    print(f"Collected {len(all_data)} enforcement actions in total.")
    df = pd.DataFrame(all_data, columns=["Title", "Date", "Category", "Link", "Agency"])
    filename = f"enforcement_actions_{year}_{month}.csv"
    df.to_csv(filename, index=False)
    print(f"Data saved to {filename}")
else:
    print("No data was collected.")

return df

```

```
# Run the function for debugging
df = scrape_enforcement_actions(2023, 1)
print(df.head())
```

b. Create Dynamic Scraper (PARTNER 2.. continued)

Total enforcement actions collected: 1534 Earliest enforcement action details: Title Podiatrist Pays \$90,000 To Settle False Billin... Date 2023-01-03 00:00:00 Category Criminal and Civil Actions Link /fraud/enforcement/podiatrist-pays-90000-to-se... Agency U.S. Attorney's Office, Southern District of T... Name: 1533, dtype: object

```
# Run the function for data collection since January 2021
df = scrape_enforcement_actions(2021, 1)
print(df.head())
```

#Chcking partner's code

```
import requests
from bs4 import BeautifulSoup
import pandas as pd
import time
import random
from concurrent.futures import ThreadPoolExecutor, as_completed

def scrape_enforcement_actions(year_start, month_start, year_end, month_end):
    # Step 1: Ensure that start year is >= 2013
    if year_start < 2013:
        print("Please restrict to a year >= 2013, as only enforcement actions from 2013 onwards are listed.")
        return None

    # Lists to hold data
    titles, dates, categories, links, agencies = [], [], [], [], []

    # Step 2: Define a function to fetch and scrape individual pages
    def fetch_page(url):
        response = requests.get(url)

        if response.status_code != 200:
            print(f"Error fetching URL: {url}")
            return []

        soup = BeautifulSoup(response.text, 'html.parser')
        items = soup.select('h2.usa-card__heading')

        page_data = []
```



```

for item in items:
    title = item.select_one('a').text.strip()
    link = "https://oig.hhs.gov" + item.select_one('a')['href']
    category = item.find_next('li', class_='display-inline-block usa-tag text-no-lowercase text-base-darkest bg-base-lightest margin-right-1').text.strip()
    date = item.find_next('span', class_='text-base-dark padding-right-105').text.strip()

    # Visit the link to extract the agency
    enforcement_page = requests.get(link)
    if enforcement_page.status_code != 200:
        print(f"Error fetching {link}: {enforcement_page.status_code}")
        continue

    enforcement_soup = BeautifulSoup(enforcement_page.text, 'html.parser')
    agency_tag = enforcement_soup.select('div.content p:nth-of-type(3)')
    agency_name = agency_tag[0].text.strip() if agency_tag else "Unknown"

    page_data.append((title, date, category, link, agency_name))

return page_data

# Step 3: Loop through months and years
page_num = 1
all_data = []

# Loop through months and years from start to end
for year in range(year_start, year_end + 1):
    # Calculate the start and end month for each year
    start_month = month_start if year == year_start else 1
    end_month = month_end if year == year_end else 12

    for month in range(start_month, end_month + 1):
        # Create year-month string
        year_month = f"{year}-{month:02d}"
        print(f"Scraping {year_month}...")

    # Step 4: Concurrently fetch pages for the given month and year
    with ThreadPoolExecutor(max_workers=5) as executor:
        future_to_page = {executor.submit(fetch_page, f"https://oig.hhs.gov/fraud/enforcement/?year_month={year_month}&page={page_num + i}"): i for i in range(5)}

        for future in as_completed(future_to_page):
            data = future.result()
            if data:
                all_data.extend(data)

```

```

        page_num += 5 # Skip 5 pages per loop for faster scraping

# Step 5: Create a DataFrame with the collected data
titles, dates, categories, links, agencies = zip(*all_data)
df = pd.DataFrame({
    "Title": titles,
    "Date": dates,
    "Category": categories,
    "Link": links,
    "Agency": agencies
})

# Step 6: Save the DataFrame to a .csv file
file_name = f"enforcement_actions_{year_start}-{month_start}to{year_end}-{month_end}.csv"
df.to_csv(file_name, index=False)

# Step 7: Return the DataFrame
return df

# Example usage: scrape enforcement actions from January 2021 to present (current month)
from datetime import datetime

today = datetime.today()
year_end = today.year
month_end = today.month

df = scrape_enforcement_actions(2021, 1, year_end, month_end)
if df is not None:
    print(f"Total enforcement actions scraped: {len(df)}")
    print(f"Earliest enforcement action details:\n{df.iloc[0]}")

#I got 3000 enforcement actions, and the earliest date is February 24, 2021.

```

Step 3: Plot data based on scraped data

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

```

import pandas as pd
import altair as alt

# Load the CSV data into a DataFrame
df = pd.read_csv('C:/Users/arifm/OneDrive/Documents/GitHub/Pset5/enforcement_actions_2021_1.csv')

```

```
# Convert the 'Date' column to datetime format (assuming the 'Date' column is present in the CSV)
df['Date'] = pd.to_datetime(df['Date'])

# Extract Year and Month from 'Date' for grouping
df['YearMonth'] = df['Date'].dt.to_period('M')

# Aggregate the data by YearMonth to get the count of actions per month
monthly_counts = df.groupby('YearMonth').size().reset_index(name='Count')
monthly_counts['YearMonth'] = monthly_counts['YearMonth'].dt.to_timestamp() # Convert to timestamp for Altair

# Create the line chart using Altair
line_chart = alt.Chart(monthly_counts).mark_line().encode(
    x=alt.X('YearMonth:T', title='Month-Year'),
    y=alt.Y('Count:Q', title='Number of Enforcement Actions')
).properties(
    title="Number of Enforcement Actions Over Time (Monthly)",
    width=700,
    height=400
)
# Display the chart
line_chart.display()
```



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

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

```
import pandas as pd
import altair as alt

# Assuming the 'df' DataFrame is loaded with the scraped data

# Convert all Period columns to datetime (or str) in the DataFrame
for col in df.columns:
    if pd.api.types.is_period_dtype(df[col]):
        df[col] = df[col].dt.to_timestamp() # Convert Period to datetime

# Ensure 'Date' column is in datetime format
```

```
df['Date'] = pd.to_datetime(df['Date'], errors='coerce')

# Step 1: Aggregate the data by year-month
df['Year_Month'] = df['Date'].dt.to_period('M')
df['Year_Month'] = df['Year_Month'].dt.to_timestamp() # Ensure Year_Month is in timestamp format for Altair

# Aggregate data by Year-Month
enforcement_by_month = df.groupby('Year_Month').size().reset_index(name='Num_Actions')

# Plot Number of Enforcement Actions Over Time
line_chart = alt.Chart(enforcement_by_month).mark_line().encode(
    x=alt.X('Year_Month:T', title='Year-Month'),
    y=alt.Y('Num_Actions:Q', title='Number of Actions'),
    tooltip=['Year_Month:T', 'Num_Actions:Q']
).properties(title='Number of Enforcement Actions Over Time')

line_chart # Display chart without .show()

# Step 2: Classify actions by topic based on keywords in the title
def classify_topic(title):
    title = title.lower()
    if 'health care' in title:
        return 'Health Care Fraud'
    elif 'financial' in title or 'bank' in title:
        return 'Financial Fraud'
    elif 'drug' in title:
        return 'Drug Enforcement'
    elif 'bribery' in title or 'corruption' in title:
        return 'Bribery/Corruption'
    else:
        return 'Other'

df['Topic'] = df['Title'].apply(classify_topic)

# Step 3: Classify 'Enforcement Type' based on category
def classify_enforcement_type(category):
    if 'State' in category or 'State Enforcement' in category:
        return 'State Enforcement Agencies'
    else:
        return 'Criminal and Civil Actions'

df['Enforcement_Type'] = df['Category'].apply(classify_enforcement_type)

# Aggregate by Enforcement Type and Month
```

```
enforcement_by_type_month = df.groupby(['Year_Month', 'Enforcement_Type']).size().reset_index(name='Num_Actions')

# Plot by Enforcement Type
enforcement_by_type_chart = alt.Chart(enforcement_by_type_month).mark_line().encode(
    x=alt.X('Year_Month:T', title='Year-Month'),
    y=alt.Y('Num_Actions:Q', title='Number of Actions'),
    color='Enforcement_Type:N',
    tooltip=['Year_Month:T', 'Num_Actions:Q', 'Enforcement_Type:N']
).properties(title='Enforcement Actions by Type (Criminal & Civil vs State Agencies)')

enforcement_by_type_chart # Display chart without .show()

# Step 4: Plot Number of Enforcement Actions by Topic within "Criminal and Civil Actions"

# Filter for "Criminal and Civil Actions"
df_criminal_civil = df[df['Enforcement_Type'] == 'Criminal and Civil Actions']

# Aggregate by Topic and Month
enforcement_by_topic = df_criminal_civil.groupby(['Year_Month', 'Topic']).size().reset_index(name='Num_Actions')

# Plot by Topic
topic_chart = alt.Chart(enforcement_by_topic).mark_line().encode(
    x=alt.X('Year_Month:T', title='Year-Month'),
    y=alt.Y('Num_Actions:Q', title='Number of Actions'),
    color='Topic:N',
    tooltip=['Year_Month:T', 'Num_Actions:Q', 'Topic:N']
).properties(title='Enforcement Actions by Topic in Criminal and Civil Actions')

topic_chart # Display chart without .show()
```



- based on five topics

```
import geopandas as gpd
import matplotlib.pyplot as plt
# Step 1: Load the shapefile for US Attorney Districts
# Correct the quote mark at the end of the file path
US_Attorney_path_shp="C:\\Users\\arifm\\OneDrive\\Documents\\GitHub\\Pset5\\geo_export_bb370b2c-46eb-4eb9-a2de-46b19e5bb7b1.shp"

# Read the shapefile
gdf_districts = gpd.read_file(US_Attorney_path_shp)

# Inspect the shapefile to confirm column names and district format
print("Shapefile columns:", gdf_districts.columns)
print(gdf_districts.head())

# Step 2: Filter enforcement actions to include only district-level records
district_df = df[df['Category'] == 'State Enforcement Agencies']
district_df = district_df[district_df['Agency'].str.contains('District', case=False, na=False)]

# Display filtered data to verify
print("Filtered district-level enforcement actions:")
print(district_df[['Agency', 'Category']].head())
```

```

# Step 3: Extract district name, standardize format, and count actions
district_df['District'] = district_df['Agency'].str.extract(r'(District.*)').fillna('')
district_df['District'] = district_df['District'].str.replace(r'^a-zA-Z\s', '', regex=True).str.strip()
district_counts = district_df.groupby('District').size().reset_index(name='Count')

# Display district_counts for validation
print("District counts DataFrame:")
print(district_counts.head())

# Step 4: Prepare shapefile by aligning district names for merging
# Standardize judicial_d in the GeoDataFrame for comparison
gdf_districts['judicial_d'] = gdf_districts['judicial_d'].str.replace(r'^a-zA-Z\s', '', regex=True).str.strip()

# Merge district_counts with shapefile GeoDataFrame
merged_gdf = gdf_districts.merge(district_counts, left_on='judicial_d', right_on='District', how='left')
merged_gdf['Count'] = merged_gdf['Count'].fillna(0) # Fill NaNs with 0 for missing districts

# Verify that merged_gdf has non-zero counts where appropriate
print("Merged GeoDataFrame with Count values:")
print(merged_gdf[['judicial_d', 'Count']].head(20))

# Step 5: Plot the map using geopandas
fig, ax = plt.subplots(1, 1, figsize=(12, 8))
merged_gdf.plot(column='Count', cmap='YlGnBu', linewidth=0.8, ax=ax, edgecolor='0.8', legend=True)
ax.set_title('US Attorney District-Level Enforcement Actions')
plt.axis('off') # Optional: Turn off axis for a cleaner map

plt.show()
```

Shapefile columns: Index(['statefp', 'judicial_d', 'aland', 'awater', 'state', 'chief_judg', 'nominating', 'term_as_ch', 'shape_leng', 'shape_area', 'abbr', 'district_n', 'shape__are', 'shape__len', 'geometry'], dtype='object')

	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 ...

```

Filtered district-level enforcement actions:

	Agency	\
487	District of Columbia Inspector General	
577	U.S. Attorney's Office, Eastern District of Wa...	
578	U.S. Attorney's Office, Northern District of I...	
1225	U.S. Attorney's Office, District of Columbia	
1315	U.S. Attorney's Office, Southern District of I...	

	Category
487	State Enforcement Agencies
577	State Enforcement Agencies
578	State Enforcement Agencies
1225	State Enforcement Agencies
1315	State Enforcement Agencies

District counts DataFrame:

	District	Count
0	District of Columbia	1
1	District of Columbia Inspector General	1
2	District of Illinois	1
3	District of Indiana	1
4	District of Iowa	1

Merged GeoDataFrame with Count values:

	judicial_d	Count
0	Western District of Kentucky	0.0
1	Eastern District of Kentucky	0.0
2	Southern District of Indiana	0.0
3	Middle District of Alabama	0.0
4	Southern District of Alabama	0.0
5	Western District of Arkansas	0.0
6	Eastern District of Arkansas	0.0
7	Northern District of California	0.0
8	Eastern District of California	0.0
9	Central District of California	0.0
10	District of Colorado	0.0
11	District of District of Columbia	0.0

12	Middle District of Florida	0.0
13	Northern District of Florida	0.0
14	Middle District of Georgia	0.0
15	Southern District of Georgia	0.0
16	Northern District of Georgia	0.0
17	District of Idaho	0.0
18	District of Kansas	0.0
19	Northern District of Texas	0.0



Step 4: Create maps of enforcement activity

1. Map by State (PARTNER 1)

```
import pandas as pd
import geopandas as gpd
```

```
import altair as alt

# Clean the 'Agency' column to extract state names for state-level enforcement actions
def extract_state(agency_name):
    # Check if the agency is a state-level agency (contains "State of")
    if 'State of' in agency_name:
        # Extract the state name (after "State of")
        state_name = agency_name.split("State of")[-1].strip()
        return state_name
    else:
        return None

# Add a 'State' column to the dataframe for state-level agencies
df['State'] = df['Agency'].apply(extract_state)

# Filter the dataset to only include state-level enforcement actions
state_actions = df.dropna(subset=['State'])

# Load the shapefile for US states
states_shapefile = "C:\\\\Users\\arifm\\OneDrive\\Documents\\GitHub\\Pset5\\cb_2018_us_state_500k.shp" # Replace with your actual path
states_gdf = gpd.read_file(states_shapefile)

# Merge the enforcement actions with the states shapefile on the 'State' column
merged_states = states_gdf.merge(state_actions.groupby('State').size().reset_index(name='Num_Actions'),
                                  left_on='NAME', right_on='State', how='left')

# Plot the choropleth map
state_map = alt.Chart(merged_states).mark_geoshape().encode(
    color='Num_Actions:Q',
    tooltip=['NAME:N', 'Num_Actions:Q']
).properties(
    title='Number of Enforcement Actions by State (State-Level Agencies)',
    width=800,
    height=500
)

state_map.show()
```



2. Map by District (PARTNER 2)

```
import geopandas as gpd
import matplotlib.pyplot as plt
# Step 1: Load the shapefile for US Attorney Districts
# Correct the quote mark at the end of the file path
US_Attorney_path_shp= "C:\\Users\\arifm\\OneDrive\\Documents\\GitHub\\Pset5\\geo_export_bb370b2c-46eb-4eb9-a2de-46b19e5bb7b1.shp"

# Read the shapefile
gdf_districts = gpd.read_file(US_Attorney_path_shp)
```

```
# Inspect the shapefile to confirm column names and district format
print("Shapefile columns:", gdf_districts.columns)
print(gdf_districts.head())

# Step 2: Filter enforcement actions to include only district-level records
district_df = df[df['Category'] == 'State Enforcement Agencies']
district_df = district_df[district_df['Agency'].str.contains('District', case=False, na=False)]

# Display filtered data to verify
print("Filtered district-level enforcement actions:")
print(district_df[['Agency', 'Category']].head())

# Step 3: Extract district name, standardize format, and count actions
district_df['District'] = district_df['Agency'].str.extract(r'(District.*)').fillna('')
district_df['District'] = district_df['District'].str.replace(r'^a-zA-Z\s', '', regex=True).str.strip()
district_counts = district_df.groupby('District').size().reset_index(name='Count')

# Display district_counts for validation
print("District counts DataFrame:")
print(district_counts.head())

# Step 4: Prepare shapefile by aligning district names for merging
# Standardize judicial_d in the GeoDataFrame for comparison
gdf_districts['judicial_d'] = gdf_districts['judicial_d'].str.replace(r'^a-zA-Z\s', '', regex=True).str.strip()

# Merge district_counts with shapefile GeoDataFrame
merged_gdf = gdf_districts.merge(district_counts, left_on='judicial_d', right_on='District', how='left')
merged_gdf['Count'] = merged_gdf['Count'].fillna(0) # Fill NaNs with 0 for missing districts

# Verify that merged_gdf has non-zero counts where appropriate
print("Merged GeoDataFrame with Count values:")
print(merged_gdf[['judicial_d', 'Count']].head(20))

# Step 5: Plot the map using geopandas
fig, ax = plt.subplots(1, 1, figsize=(12, 8))
merged_gdf.plot(column='Count', cmap='YlGnBu', linewidth=0.8, ax=ax, edgecolor='0.8', legend=True)
ax.set_title('US Attorney District-Level Enforcement Actions')
plt.axis('off') # Optional: Turn off axis for a cleaner map

plt.show()
```

```
Shapefile columns: Index(['statefp', 'judicial_d', 'aland', 'awater', 'state', 'chief_judg',
                           'nominating', 'term_as_ch', 'shape_leng', 'shape_area', 'abbr',
                           'district_n', 'shape__are', 'shape__len', 'geometry'],
                           dtype='object')
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 ...

```

Filtered district-level enforcement actions:

	Agency	\
487	District of Columbia Inspector General	
577	U.S. Attorney's Office, Eastern District of Wa...	
578	U.S. Attorney's Office, Northern District of I...	
1225	U.S. Attorney's Office, District of Columbia	
1315	U.S. Attorney's Office, Southern District of I...	

	Category
487	State Enforcement Agencies
577	State Enforcement Agencies
578	State Enforcement Agencies
1225	State Enforcement Agencies
1315	State Enforcement Agencies

District counts DataFrame:

	District	Count
0	District of Columbia	1
1	District of Columbia Inspector General	1
2	District of Illinois	1
3	District of Indiana	1
4	District of Iowa	1

Merged GeoDataFrame with Count values:

	judicial_d	Count
--	------------	-------

0	Western District of Kentucky	0.0
1	Eastern District of Kentucky	0.0
2	Southern District of Indiana	0.0
3	Middle District of Alabama	0.0
4	Southern District of Alabama	0.0
5	Western District of Arkansas	0.0
6	Eastern District of Arkansas	0.0
7	Northern District of California	0.0
8	Eastern District of California	0.0
9	Central District of California	0.0
10	District of Colorado	0.0
11	District of District of Columbia	0.0
12	Middle District of Florida	0.0
13	Northern District of Florida	0.0
14	Middle District of Georgia	0.0
15	Southern District of Georgia	0.0
16	Northern District of Georgia	0.0
17	District of Idaho	0.0
18	District of Kansas	0.0
19	Northern District of Texas	0.0



Extra Credit

1. Merge zip code shapefile with population

2. Conduct spatial join

3. Map the action ratio in each district