Problem Set 5

Lauren Laine and Mohamed Mohamed

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): Lauren Laine, llaine
    - Partner 2 (name and cnet ID):
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: **___** 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 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)

```python
import requests
from bs4 import BeautifulSoup
url = 'https://oig.hhs.gov/fraud/enforcement/'
response = requests.get(url)
soup = BeautifulSoup(response.text, 'lxml')
soup.text[0:50]
```

```
'\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\nEnforcement Actions | Office '
```

```python
# scrape title of the enforcement action
# used ChatGPT to figure out the the class attribute is written as class_
#Prompt why won't this code run : soup.find_all('h2', class
 ↪  ='usa-card__heading')
usa_card__heading=soup.find_all('h2', class_ ='usa-card__heading')
a_tags=[]
for tag in usa_card__heading:
  a=tag.find('a').text
  a_tags.append(a)
#check and make sure all titles were collected
print(a_tags[19])
```

```python
#scrape date
dates=[]
soup_dates=soup.find_all('span', class_='text-base-dark padding-right-105')
for tag in soup_dates:
  text=tag.text
  dates.append(text)
print(dates[0:5])
print(dates[19])
```

```
# scrape category
category=[]
soup_category=soup.find_all('li', class_="display-inline-block usa-tag text-no-lowercase text-base-darkest bg-base-lightest margin-right-1")
for tag in soup_category:
  text=tag.text
  category.append(text)
print(category[19])
```

```
#scrape link associated with the enforecment action
hrefs=[]
link_tags=[]
full_links=[]
for tag in usa_card__heading:
  link_tags.append(tag.find('a').attrs)

for link in link_tags:
  href=link.get('href')
  hrefs.append(href)

print(hrefs[19])
prefix='https://oig.hhs.gov/'
for href in hrefs:
  link= prefix+href
  full_links.append(link)

print(full_links)
```

```
#create dataframe
df=pd.DataFrame({'Title':a_tags, 'Date':dates, 'Category':category, 'Link':full_links})

df.head()
len(df)
```

2. Crawling (PARTNER 1)

```
url =
 ↪  'https://oig.hhs.gov/fraud/enforcement/washington-doctor-settles-allegations-he-submitte
response = requests.get(url)
soup = BeautifulSoup(response.text, 'lxml')
ul_tag=soup.find('ul', class_="usa-list usa-list--unstyled margin-y-2")
li_list=ul_tag.find_all('li')
print(li_list)
print(li_list[1])
#used ChatGPT to figure out how to remove the span tag.
# Prompt remove span tag and content with Beautiful Soup in Python
span_tag = li_list[1].find('span', class_='padding-right-2 text-base')
if span_tag:
    span_tag.decompose()
print(li_list[1].text)
```

```
agencies=[]
for i in range(len(full_links)):
  url = full_links[i]
  response = requests.get(url)
  soup = BeautifulSoup(response.text, 'lxml')
  ul_tag=soup.find('ul', class_="usa-list usa-list--unstyled margin-y-2")
  li_list=ul_tag.find_all('li')
  span_tag = li_list[1].find('span', class_='padding-right-2 text-base')
  if span_tag:
    span_tag.decompose()
  agency=(li_list[1].text)
  print(agency)
  agencies.append(agency)
```

```
df['Agency']=agencies
df.head()
```

```
df.to_csv('first_page_scrape.csv')
```

```
df=pd.read_csv(r"C:\Users\laine\OneDrive\Documents\GitHub\problem-set-5-lauren-and-mohamed\f
df.head()
```

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

Step 2: Making the scraper dynamic

1. Turning the scraper into a function

- a. Pseudo-Code (PARTNER 2)

1. First will check if the input year is >= 2013, then print error message if year < 2013 and return (False); otherwise, return (True).
2. Will return four empty lists to store scraped data (titles, dates, categories, links).
3. Get the current year and month.
4. Send a request to the URL so that it will return content using BeautifulSoup.
5. Get all `h2` elements with a specific class, extract text from anchor tags, to return title.
6. Get all `span` elements with a specific class, extract text, which will return dates.
7. Get all `li` elements with a specific class, extract text, which will return categories.
8. Get all `h2` elements, extract `href` attributes, prefix with base URL, which will return list of full URLs (links).
9. making_dataframe that has (titles, dates, categories, links)
10. Save the DataFrame to CSV file with a filename based on year and month.
11. Summary to print the number of records, earliest action date, and title if data exists.
12. Get and parse HTML content, locate specific (ul) and (li) tags, extract agency name.
13. Initialize data containers, scrape pages for data fields (titles, dates, categories, links) until reaching the target date, and use parallel processing for agency data.

- b. Create Dynamic Scraper (PARTNER 2)

```python
import requests
from bs4 import BeautifulSoup
from datetime import datetime
```

```python
def check_input(year):
    if year < 2013:
        print("Please enter a year >= 2013, as only enforcement actions after
         ↪  2013 are available.")
        return False
    return True
```

```python
def initialize_data_containers():
  return [], [], [], []
```

```python
def get_today_date():
    now = datetime.now()
    return now.year, now.month
```

```python
def get_page_content(url):
    response = requests.get(url)
    return BeautifulSoup(response.text, 'lxml')
```

```python
def get_titles(soup):
    titles = []
    headings = soup.find_all('h2', class_='usa-card__heading')
    for tag in headings:
        titles.append(tag.find('a').text)
    return titles
```

```python
def get_dates(soup):
    dates = []
    date_tags = soup.find_all('span', class_='text-base-dark
 ↪  padding-right-105')
    for tag in date_tags:
        dates.append(tag.text)
    return dates
```

```python
def get_categories(soup):
    categories = []
    category_tags = soup.find_all('li', class_="display-inline-block usa-tag text-no-lowercase text-base-darkest bg-base-lightest margin-right-1")
    for tag in category_tags:
        categories.append(tag.text)
    return categories
```

```python
def get_links(soup):
    full_links = []
    link_tags = [tag.find('a').attrs for tag in soup.find_all('h2', class_='usa-card__heading')]
    prefix = 'https://oig.hhs.gov/'
    for link in link_tags:
        full_links.append(prefix + link.get('href'))
    return full_links
```

```python
def making_dataframe(titles, dates, categories, links):
    return pd.DataFrame({
        'Title': titles,
        'Date': dates,
        'Category': categories,
        'Link': links
    })
```

```python
def save_to_csv(df, start_year, start_month):
    filename = f"enforcement_actions_{start_year}_{start_month:02d}.csv"
    df.to_csv(filename, index=False)
    print(f"Data saved to {filename}")
```

```python
def showing_summary(df):
    print(f"Number of enforcement actions: {len(df)}")
    if not df.empty:
        earliest_date = df['Date'].min()
        earliest_action = df[df['Date'] == earliest_date].iloc[0]
        print(f"Earliest enforcement action Date - {earliest_action['Date']}, Title - {earliest_action['Title']}")
```

```python
# used ChatGPT to debug function and learn about concurrrent.futures.
#prompt: "Is there a way to fake the function faster"

from concurrent.futures import ThreadPoolExecutor

def scrape_agency_data(url):
    response = requests.get(url)
    soup = BeautifulSoup(response.text, 'lxml')
    ul_tag = soup.find('ul', class_="usa-list usa-list--unstyled margin-y-2")

    if ul_tag:
        li_list = ul_tag.find_all('li')
        span_tag = li_list[1].find('span', class_='padding-right-2
 ↪  text-base')
        if span_tag:
            span_tag.decompose()
        return li_list[1].text.strip()  # Return the agency
    return None
```

```python
def scrape(year, month):
    date_string = f'{year}-{month+1:02d}'
    set_date = pd.to_datetime(date_string, format='%Y-%m')

    titles = []
    dates = []
    categories = []
    full_links = []
    agencies = []

    session = requests.Session()

    for i in range(250):
        base = 'https://oig.hhs.gov/fraud/enforcement/?page='
        url = f'{base}{i}'
        response = requests.get(url)
        soup = BeautifulSoup(response.text, 'lxml')

        body = soup.find('body')
        soup_dates = body.find_all('span', class_='text-base-dark
 ↪  padding-right-105')
        soup_titles = body.find_all('h2', class_='usa-card__heading')
        soup_category = body.find_all('li', class_="display-inline-block
↪   usa-tag text-no-lowercase text-base-darkest bg-base-lightest
↪   margin-right-1")

        for date_tag, title_tag, category_tag in zip(soup_dates, soup_titles,
        ↪   soup_category):
            # Dates
            date = pd.to_datetime(date_tag.text, format='%B %d, %Y')
            dates.append(date)

            # Titles
            title = title_tag.find('a').text
            titles.append(title)

            # Categories
            category = category_tag.text.strip()
            categories.append(category)

            # Links
            href = title_tag.find('a').attrs.get('href')
            full_link = f'https://oig.hhs.gov/{href}'
            full_links.append(full_link)

        if date < set_date:
            break

        time.sleep(2)

    # Use ThreadPoolExecutor to scrape agency data in parallel
    with ThreadPoolExecutor() as executor:
        agencies = list(executor.map(scrape_agency_data, full_links))

    scraped_data = pd.DataFrame({
        'Title': titles,
        'Date': dates,
        'Category': categories,
        'Link': full_links,
        'Agency': agencies
    })
    save_to_csv(scraped_data, year, month)
    return scraped_data
```

```
# Running scraper starting from January 2023
scraped = scrape(2023, 1)
```

```
scraped=pd.read_csv(r"C:\Users\laine\OneDrive\Documents\GitHub\problem-set-5-lauren-and-mohar
```

```
print(len(scraped))
print(scraped.tail(1))
```

```
1500
                                                  Title        Date  \
1499  Martin Joseph O'Brien Agreed to Be Excluded fo...  2023-01-30   

                            Category  \
1499  CMP and Affirmative Exclusions   

                                                   Link  \
1499  https://oig.hhs.gov//fraud/enforcement/martin-...   

                                                 Agency  
1499  Enforcement Types:\n\n\n                      ...  
```

The length of enforcement actions we get in our final dataframe is 1500. The earliest enforcement action it scraped in Martin Joseph O'Brien Agreed to Be Excluded fo… 2023-01-30

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

```
df_21=scrape(2021, 1 )
```

```
df_21=pd.read_csv(r"C:\Users\laine\OneDrive\Documents\GitHub\problem-set-5-lauren-and-mohamec
```

```
#print(len(df_21))
#print(df_21.tail(1))
```

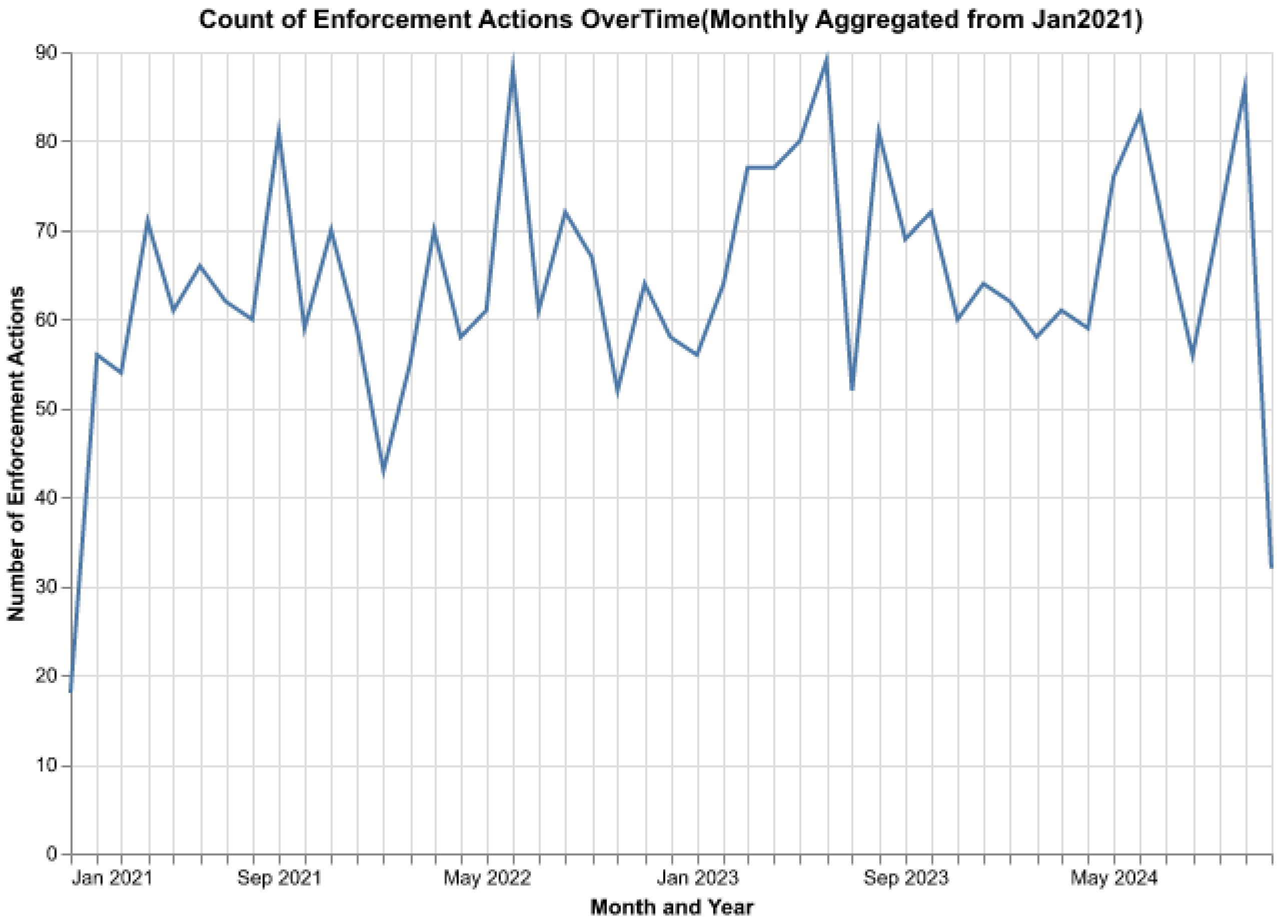
The length of the dataframe is 3020. 3019 Attorney General Becerra Announces $40 Million…
2021-01-22
Category:State Enforcement Agencies
https://oig.hhs.gov//fraud/enforcement/attorne…
Agency: California Attorney General

Step 3: Plot data based on scraped data

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

```
df_21_months=df_21
df_21_months['Date'] = pd.to_datetime(df_21_months['Date'], errors='coerce')

# Now, you can apply .dt to create 'YearMonth'
df_21_months['YearMonth'] = df_21_months['Date'].dt.to_period('M')

# Aggregating the number of enforcement actions per month
monthly_counts =
↪  df_21_months.groupby('YearMonth').size().reset_index(name='EnforcementCount')

# Changing 'YearMonth' into datetime format for compatibility with Altair.
monthly_counts['YearMonth'] = monthly_counts['YearMonth'].dt.to_timestamp()
```

```
chart_line = alt.Chart(monthly_counts).mark_line().encode(
    x=alt.X('YearMonth:T', title='Month and Year',
    axis=alt.Axis(format='%b %Y', tickCount='month')),
    y=alt.Y('EnforcementCount:Q', title='Number of Enforcement Actions'),
    tooltip=['YearMonth:T', 'EnforcementCount']
).properties(
    title='Count of Enforcement Actions OverTime(Monthly Aggregated from
↪   Jan2021)',
    width=600,
    height=400
)

# Display the chart
chart_line
```



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

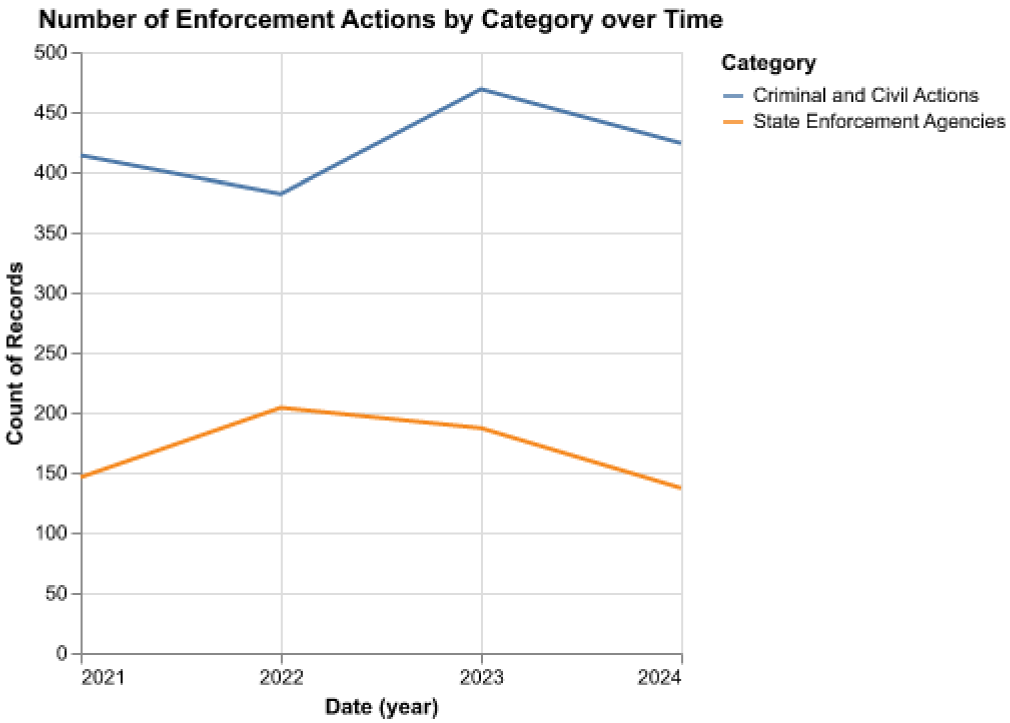
- based on “Criminal and Civil Actions” vs. “State Enforcement Agencies”

```
df_21=df_21.drop('YearMonth', axis=1)
```

```
filtered_df_21 = df_21[(df_21['Category'] == "Criminal and Civil Actions") |
↪   (df_21['Category'] == 'State Enforcement Agencies')]

filtered_df_21.head(20)
df_21.head(20)
```

	Title	Date	Category
0	Pharmacist and Brother Convicted of $15M Medic...	2024-11-08	Criminal and Civil Actions
1	Boise Nurse Practitioner Sentenced To 48 Month...	2024-11-07	Criminal and Civil Actions
2	Former Traveling Nurse Pleads Guilty To Tamper...	2024-11-07	Criminal and Civil Actions
3	Former Arlington Resident Sentenced To Prison ...	2024-11-07	Criminal and Civil Actions
4	Paroled Felon Sentenced To Six Years For Fraud...	2024-11-07	Criminal and Civil Actions
5	Former Licensed Counselor Sentenced For Defrau...	2024-11-06	Criminal and Civil Actions
6	Macomb County Doctor And Pharmacist Agree To P...	2024-11-04	Criminal and Civil Actions
7	Rocky Hill Pharmacy And Its Owners Indicted Fo...	2024-11-04	Criminal and Civil Actions
8	North Texas Medical Center Pays $14.2 Million ...	2024-11-04	Criminal and Civil Actions
9	New England Doctor Pleads Guilty To Drug Distr...	2024-11-04	Criminal and Civil Actions
10	Attorney General Alan Wilson Announces Upstate...	2024-11-04	State Enforcement Agencies
11	St. Louis County Woman Accused Of $3 Million H...	2024-11-01	Criminal and Civil Actions
12	Lab Owner And Marketing Company Owner Both Fou...	2024-11-01	Criminal and Civil Actions
13	Compound Ingredient Supplier Medisca Inc., To ...	2024-11-01	Criminal and Civil Actions
14	The New Mexico Department Of Justice Charges F...	2024-11-01	State Enforcement Agencies
15	Nashville Woman Indicted, Charged In TBI Medic...	2024-11-01	State Enforcement Agencies
16	Michael DePalma, MD and Virginia I-Spine Physi...	2024-10-31	CMP and Affirmative Exclusions
17	Columbus Doctor, His Clinic Convicted of $1.5 ...	2024-10-31	State Enforcement Agencies
18	Mercy Health Youngstown Agreed to Pay $69,000 ...	2024-10-30	Fraud Self-Disclosures
19	Quincy-Based Physician Group To Pay $650,000 T...	2024-10-30	State Enforcement Agencies

```python
actions_by_category=alt.Chart(filtered_df_21, title='Number of Enforcement Actions by Category over Time').mark_line().encode(
  alt.X('year(Date):T'),
  alt.Y('count(Title)'),
  alt.Color('Category')
)
actions_by_category
```



- based on five topics

```
crim_and_civil=df_21[df_21['Category']=='Criminal and Civil Actions']
crim_and_civil.head()
```

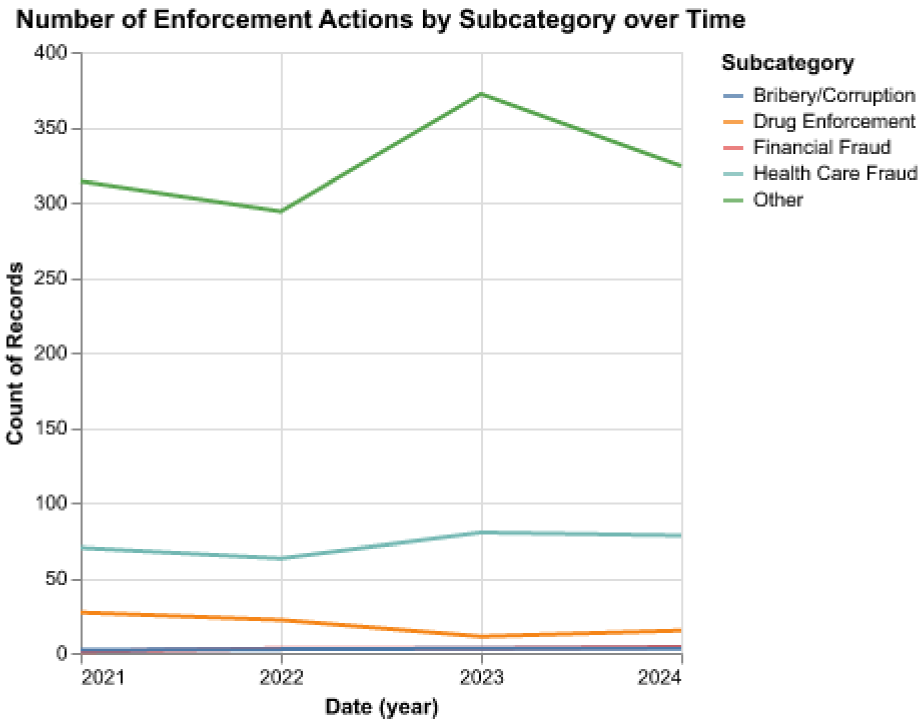
	Title	Date	Category	Link
0	Pharmacist and Brother Convicted of $15M Medic...	2024-11-08	Criminal and Civil Actions	https://o
1	Boise Nurse Practitioner Sentenced To 48 Month...	2024-11-07	Criminal and Civil Actions	https://o
2	Former Traveling Nurse Pleads Guilty To Tamper...	2024-11-07	Criminal and Civil Actions	https://o
3	Former Arlington Resident Sentenced To Prison ...	2024-11-07	Criminal and Civil Actions	https://o
4	Paroled Felon Sentenced To Six Years For Fraud...	2024-11-07	Criminal and Civil Actions	https://o

```
def assign_subcategory(title):
    if 'financial' in title.lower():
        return 'Financial Fraud'
    elif 'bank' in title.lower():
        return 'Financial Fraud'
    elif 'embezzled' in title.lower():
        return 'Financial Fraud'
    elif 'doctor' in title.lower():
        return 'Health Care Fraud'
    elif 'nurse' in title.lower():
        return 'Health Care Fraud'
    elif 'hospital' in title.lower():
        return 'Health Care Fraud'
    elif 'drug' in title.lower():
        return 'Drug Enforcement'
    elif 'possession' in title.lower():
        return 'Drug Enforcement'
    elif 'marijuana' in title.lower():
        return 'Drug Enforcement'
    elif 'bribe' in title.lower():
        return 'Bribery/Corruption'
    elif 'favor' in title.lower():
        return 'Bribery/Corruption'
    else:
        return 'Other'

crim_and_civil['Subcategory'] = crim_and_civil['Title'].apply(assign_subcategory)
```

```
#Check that there are some results in each subcategory
crim_and_civil.groupby('Subcategory').size()
```

```
Subcategory
Bribery/Corruption       8
Drug Enforcement        75
Financial Fraud         11
Health Care Fraud      291
Other                 1304
dtype: int64
```

```
actions_by_subcategory=alt.Chart(crim_and_civil, title='Number of Enforcement Actions by Subcategory over Time').mark_line().encode(
  alt.X('year(Date)'),
  alt.Y('count(Title)'),
  alt.Color('Subcategory')
)
actions_by_subcategory
```



Step 4: Create maps of enforcement activity

1. Map by State (PARTNER 1)

```python
import geopandas as gpd
census_data=gpd.read_file(r"C:\Users\laine\OneDrive\Documents\GitHub\problem-set-5-lauren-and
```

```python
enforcement_actions=df_21[df_21['Category']=='State Enforcement Agencies']
```

```python
enforcement_actions.groupby('Agency').size()
```

```
Agency
Alabama Attorney General             3
Arizona Attorney General             1
Arkansas Attorney General            3
California Attorney Genera           1
California Attorney General         28
                                    ..
```

```
Vermont Attorney General              2
Virginia Attorney General             1
Washington Attorney General           1
Washington State Attorney General     1
Wisconsin Attorney General            4
Length: 132, dtype: int64
```

```
states=['Alabama', 'Alaska', 'Arizona', 'Arkansas', 'California', 'Colorado',
 ↪  'Connecticut', 'Delaware',
'Florida', 'Georgia', 'Hawaii', 'Idaho', 'Illinois', 'Indiana', 'Iowa',
 ↪  'Kansas', 'Kentucky','Louisiana',
'Maine', 'Maryland', 'Massachusetts', 'Michigan', 'Minnesota', 'Mississippi',
 ↪  'Missouri', 'Montana', 'Nebraska',
'Nevada', 'New Hampshire', 'New Jersey', 'New Mexico', 'New York', 'North
 ↪  Carolina', 'North Dakota', 'Ohio',
'Oklahoma', 'Oregon', 'Pennsylvania', 'Rhode Island', 'South Carolina',
 ↪  'South Dakota', 'Tennessee', 'Texas',
'Utah', 'Vermont', 'Virginia', 'Washington', 'West Virginia', 'Wisconsin',
 ↪  'Wyoming']
def assign_state(agency):
    agency = agency.lower()
    for s in states:
        if s.lower() in agency:
            return s
    return 'Other'
```

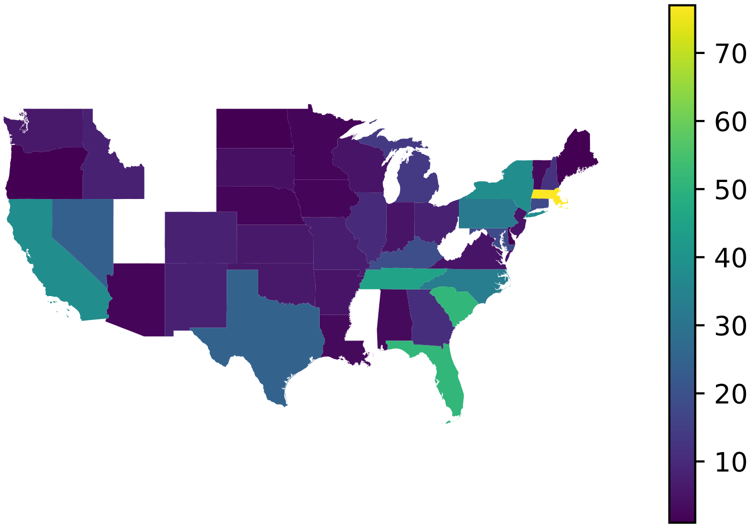
```
enforcement_actions['State']=enforcement_actions['Agency'].apply(assign_state)
enforcement_actions.head()
```

	Title	Date	Category	Link
10	Attorney General Alan Wilson Announces Upstate...	2024-11-04	State Enforcement Agencies	https:/
14	The New Mexico Department Of Justice Charges F...	2024-11-01	State Enforcement Agencies	https:/
15	Nashville Woman Indicted, Charged In TBI Medic...	2024-11-01	State Enforcement Agencies	https:/
17	Columbus Doctor, His Clinic Convicted of $1.5 ...	2024-10-31	State Enforcement Agencies	https:/
19	Quincy-Based Physician Group To Pay $650,000 T...	2024-10-30	State Enforcement Agencies	https:/

```
census_data=census_data[['NAME', 'geometry']]
merge=enforcement_actions.merge(census_data, left_on='State',
 ↪  right_on='NAME', how='left')
merge=gpd.GeoDataFrame(merge, geometry='geometry')
```

```
count=enforcement_actions.groupby('State').size()
count=count.reset_index()
count.columns=['State', 'Count']
count.head()
merged_counts=count.merge(census_data, left_on='State', right_on='NAME',
how='left')
merged_counts=gpd.GeoDataFrame(merged_counts, geometry='geometry')
```

```
plot_by_state=merged_counts.plot(column='Count', legend=True).set_axis_off()
plot_by_state
```



2. Map by District (PARTNER 2)

```
# Imports
import re
from fuzzywuzzy import process
```

```
df_21 = df_21

us_attorney_districts =
 ↪  gpd.read_file(r"C:\Users\laine\OneDrive\Documents\GitHub\problem-set-5-lauren-and-mohame
```

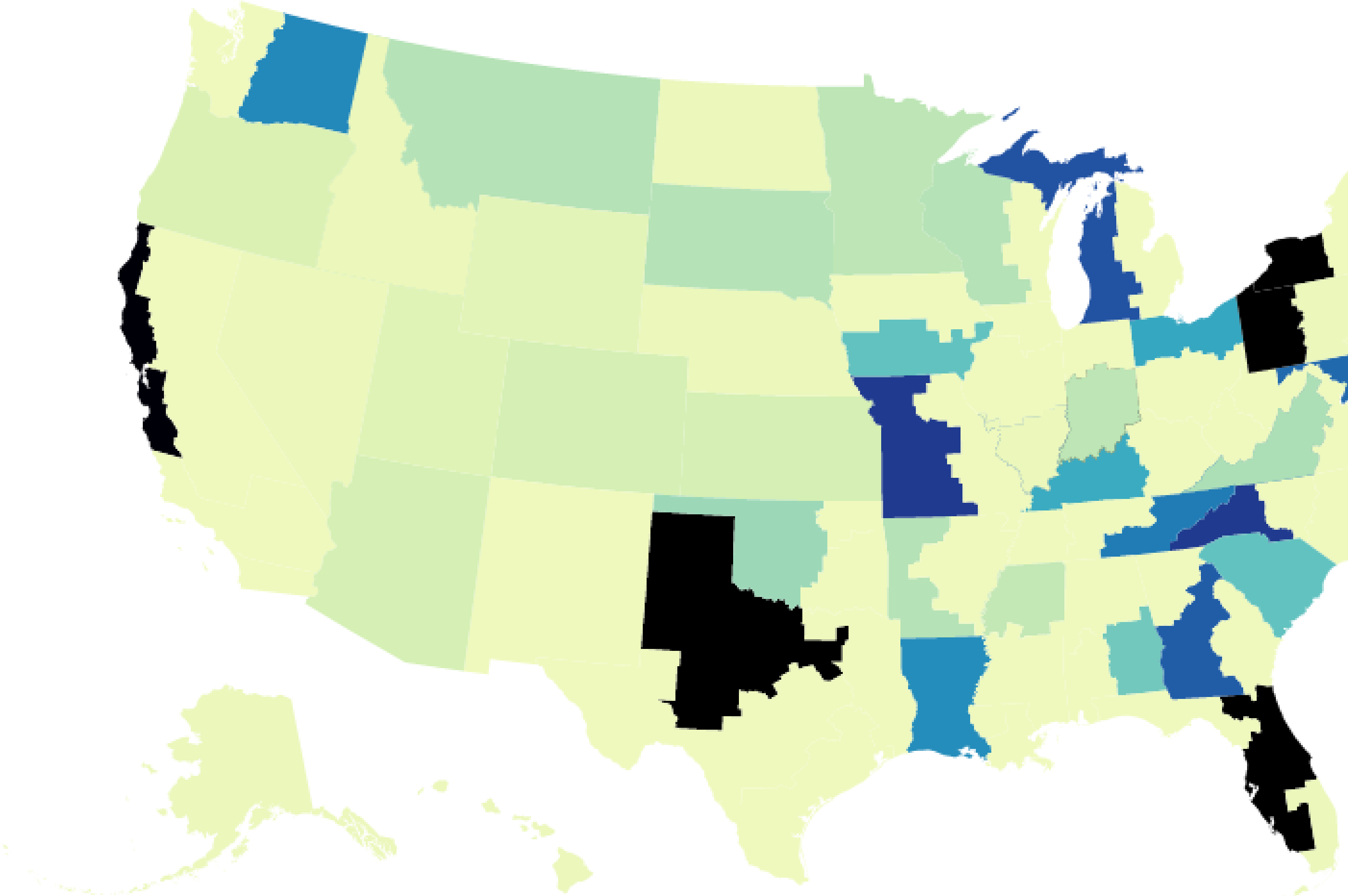
```python
enforcement_data_by_district = df_21.copy()
enforcement_data_by_district['District'] =
 ↪  enforcement_data_by_district['Agency'].str.extract(r'(District.*)')
enforcement_data_by_district['District'] =
 ↪  enforcement_data_by_district['District'].str.replace("U.S. Attorney's
 ↪  Office,", "").str.strip()

def clean_district_name(name):
    if not isinstance(name, str):
        return 'Other'
    name = name.lower()
    name = re.sub(r'u\.s\. attorney\'s office,', '', name)
    name = re.sub(r'u\.s\. department of justice and', '', name)
    name = re.sub(r'attorney\'s office,', '', name)
    name = re.sub(r'2021: u\.s\. attorney\'s office,', '', name)
    name = re.sub(r'u\.s\. ', '', name)
    name = re.sub(r'\.', '', name)
    name = re.sub(r'june 28, 2024:', '', name)
    name = re.sub(r'attorney general,', '', name)
    name = re.sub(r'district\s+of\s+|district\s+', '', name)
    name = re.sub(r'[^\w\s]', '', name)
    name = name.replace('eastern', 'east').replace('western', 'west')
    name = name.replace('northern', 'north').replace('southern', 'south')
    name = name.replace('middle', 'central')
    name = ' '.join(name.split())
    return name
```

```python
enforcement_data_by_district['cleaned_district'] =
 ↪  enforcement_data_by_district['District'].apply(clean_district_name)
us_attorney_districts['cleaned_district'] =
 ↪  us_attorney_districts['judicial_d'].apply(clean_district_name)

district_action_counts =
 ↪  enforcement_data_by_district.groupby('cleaned_district').size().reset_index(name='Count'

def match_district(district_name, choices):
    return process.extractOne(district_name, choices)[0]

district_choices = us_attorney_districts['cleaned_district'].tolist()
district_action_counts['matched_district'] =
 ↪  district_action_counts['cleaned_district'].apply(lambda x:
 ↪  match_district(x, district_choices))
```

```
merged_district_counts = us_attorney_districts.merge(district_action_counts,
left_on='cleaned_district', right_on='matched_district',
how='left')

merged_district_counts['Count'] = merged_district_counts['Count'].fillna(0)

merged_district_counts_geojson =
 ↪  merged_district_counts.to_crs(epsg=4326).__geo_interface__
```

```
alt_chart =
 ↪  alt.Chart(alt.Data(values=merged_district_counts_geojson['features'])).mark_geoshape().e
    color=alt.Color('properties.Count:Q',
                    title='Enforcement Actions',
                    scale=alt.Scale(domain=[0, 50])),
    tooltip=[
        alt.Tooltip('properties.judicial_d:N', title='District'),
        alt.Tooltip('properties.Count:Q', title='Enforcement Actions')
    ]
).properties(
    width=800,
    height=500,
    title="Enforcement Actions by U.S. Attorney District"
).project(
    type='albersUsa'
).configure_legend(
    titleFontSize=14,
    labelFontSize=12,
    symbolSize=100,
    orient='right'
)

alt_chart.show()

missing_districts = set(us_attorney_districts['cleaned_district']) -
 ↪  set(district_action_counts['cleaned_district'])
print(f"Missing Districts: {missing_districts}")
```

Enforcement Actions by U.S. Attorney District



```
Missing Districts: {'west washington', 'west tennessee', 'east texas',
'central alabama', 'south california', 'central north carolina', 'north
marianas islands', 'east arkansas', 'central florida', 'north georgia', 'west
wisconsin', 'west missouri', 'south iowa', 'north indiana', 'east new york',
'north california', 'north oklahoma', 'south florida', 'south illinois',
'central georgia', 'east california', 'south new york', 'north alabama',
'south mississippi', 'central illinois', 'west new york', 'central
pennsylvania', 'east oklahoma', 'east washington', 'east michigan', 'north
florida', 'east missouri', 'north texas', 'guam', 'west michigan', 'north
iowa', 'west north carolina', 'south indiana', 'south texas', 'west texas',
'east louisiana', 'north illinois', 'central california', 'west oklahoma',
'north mississippi', 'central louisiana', 'west pennsylvania', 'new mexico',
'south ohio', 'north ohio', 'west arkansas', 'north west virginia', 'central
tennessee', 'east tennessee', 'east virginia', 'east kentucky', 'north new
york', 'south west virginia', 'west louisiana', 'us virgin islands', 'east
north carolina', 'south georgia', 'south alabama', 'west kentucky', 'east
wisconsin', 'east pennsylvania'}
```

Extra Credit

1. Merge zip code shapefile with population

```
zipcode_geo_data=gpd.read_file(r"C:\Users\laine\OneDrive\Documents\GitHub\problem-set-5-laure
zipcode_pop_data=pd.read_csv(r"C:\Users\laine\OneDrive\Documents\GitHub\problem-set-5-lauren-
```

```
zipcode_pop_data.head()
zipcode_geo_data.head()
zipcode_geo_data['Zip Name']=zipcode_geo_data['NAME'].map(lambda x: f'ZCTA5
 ↪  {x}')
```

```
zipcode_merged=zipcode_pop_data.merge(zipcode_geo_data, left_on='NAME',
 ↪  right_on='Zip Name', how='left')
zipcode_merged=gpd.GeoDataFrame(zipcode_merged, geometry='geometry')
```

2. Conduct spatial join

```
zipcode_merged = zipcode_merged.to_crs(us_attorney_districts.crs)

district_population = gpd.sjoin(zipcode_merged, us_attorney_districts,
 ↪  how="inner", predicate="intersects")

district_population =
 ↪  district_population.groupby("judicial_d")["P1_001N"].sum().reset_index()
district_population.columns = ["judicial_d", "Population"]

merged_district_counts = merged_district_counts.merge(district_population,
 ↪  on="judicial_d", how="left")
```

```
merged_district_counts_geojson =
 ↪  merged_district_counts.to_crs(epsg=4326).__geo_interface__

alt_chart =
 ↪  alt.Chart(alt.Data(values=merged_district_counts_geojson['features'])).mark_geoshape().en
    color=alt.Color('properties.Actions_Per_Capita:Q',
                    title='Enforcement Actions per 100,000 people',
                    scale=alt.Scale(scheme='blues')),
    tooltip=[
        alt.Tooltip('properties.judicial_d:N', title='District'),
        alt.Tooltip('properties.Count:Q', title='Enforcement Actions'),
        alt.Tooltip('properties.Population:Q', title='Population',
 ↪  format=','),
        alt.Tooltip('properties.Actions_Per_Capita:Q', title='Actions per
 ↪  100,000 people', format='.2f')
    ]
).properties(
    width=800,
    height=500,
    title="Enforcement Actions per Capita by U.S. Attorney District"
).project(
    type='albersUsa'
).configure_legend(
    titleFontSize=14,
    labelFontSize=12,
    symbolSize=100,
    orient='right'
)
```

```
alt_chart.show()
```

3. Map the action ratio in each district

```
merged_district_counts["Actions_Per_Capita"] =
↪  (merged_district_counts["Count"] / merged_district_counts["Population"])
↪  * 100000

merged_district_counts["Actions_Per_Capita"] =
↪  merged_district_counts["Actions_Per_Capita"].fillna(0)
```

```
merged_district_counts_geojson =
↪  merged_district_counts.to_crs(epsg=4326).__geo_interface__

alt_chart =
↪  alt.Chart(alt.Data(values=merged_district_counts_geojson['features'])).mark_geoshape().en
    color=alt.Color('properties.Actions_Per_Capita:Q',
                    title='Enforcement Actions per 100,000 people',
                    scale=alt.Scale(scheme='blues')),
    tooltip=[
        alt.Tooltip('properties.judicial_d:N', title='District'),
        alt.Tooltip('properties.Count:Q', title='Enforcement Actions'),
        alt.Tooltip('properties.Population:Q', title='Population',
↪  format=','),
        alt.Tooltip('properties.Actions_Per_Capita:Q', title='Actions per
↪  100,000 people', format='.2f')
    ]
).properties(
    width=800,
    height=500,
    title="Enforcement Actions per Capita by U.S. Attorney District"
).project(
    type='albersUsa'
).configure_legend(
    titleFontSize=14,
    labelFontSize=12,
    symbolSize=100,
    orient='right',
    labelLimit=0
```

```
)

alt_chart.show()
```