Pset 5

Ashirwad and Anand

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 (name and cnet ID): Ashirwad Wakade - ashirwad
   - Partner 2 (name and cnet ID): Anand Kshirsagar -anandkshirsagar
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: **Ashirwad Wakade and Anand Kshirsagar**
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 each** Late coins left after submission: *** 2 each *
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
import pandas as pd
# URL for scrapping
url = "https://oig.hhs.gov/fraud/enforcement/"
# Send request and parsing
response = requests.get(url)
soup = BeautifulSoup(response.text, 'lxml')
# Creating lists to hold data
titles, dates, categories, links = [], [], [], []
# Find  action entries
for item in soup.select('h2.usa-card__heading'):
    # Extracting title
    title = item.select_one('a').text.strip()
    # Extracting  and completing the link
    link = "https://oig.hhs.gov" + item.select_one('a')['href']
    # Extracting 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()
    # Extracting date
    date = item.find_next('span', class_='text-base-dark
↪   padding-right-105').text.strip()
    # Appending lists
    titles.append(title)
    links.append(link)
    categories.append(category)
    dates.append(date)
# Creating a DataFrame
df = pd.DataFrame({
    "Title": titles,
    "Date": dates,
    "Category": categories,
    "Link": links
})
# Showing head
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)

```
import requests
from bs4 import BeautifulSoup
import pandas as pd
# creating  empty list for agency names
agencies = []
# Looping  through each  action link in the df
```

```
for link in df['Link']:
    response = requests.get(link)  # Send request to the enforcement page
    soup = BeautifulSoup(response.text, 'lxml')  # Parse the page
    # Find the <h2> tag that has "Action Details"
    h2_tag = soup.find('h2', class_='font-heading-lg')
    # If <h2> there, then get the  <ul> containing the agency data
    if h2_tag:
        ul_tag = h2_tag.find_next('ul', class_='usa-list--unstyled')
        li_tags = ul_tag.find_all('li') if ul_tag else []

        # Extract  agency name from  second <li> tag only if 'Agency:' is
         ↪  present
        if len(li_tags) >= 2 and 'Agency:' in li_tags[1].text:
            agency_name = li_tags[1].text.split('Agency:')[1].strip()
        else:
            agency_name = "No agency found"
    else:
        agency_name = "No agency section found"
    # Append the agency name in list
    agencies.append(agency_name)
# Add  agency names to  dataframe
df['Agency'] = agencies

# Print the updated dataframe
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-...

                                              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. **Validate Year**:
   If year < 2013, print a reminder and exit.
2. **Set Base URL**:
   Define the URL `'https://oig.hhs.gov/fraud/enforcement/'`.
3. **Loop Through Pages**:
   - Start at `page = 1`.
   - Continue until no more enforcement actions are found.
4. **Scrape Data**:
   - Extract title, date, category, and link.
   - Visit each link to extract the agency.
5. **Rate Limiting**:
   After each page, `time.sleep(1)`.
6. **Store & Save**:
   Append data, convert to DataFrame, and save as CSV.
7. **Return DataFrame**.

b. Create Dynamic Scraper (PARTNER 2)

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

def scrape_enforcement_actions(year, month):
    if year < 2013:
        return
    base_url = "https://oig.hhs.gov/fraud/enforcement/"
    page = 1
    all_data = []
    target_date = datetime(year, month, 1)

    while True:
        url = f"{base_url}?page={page}"
        response = requests.get(url)
        soup = BeautifulSoup(response.text, 'html.parser')
        actions = soup.select('.usa-card__heading')
        if not actions:
            break
        for action in actions:
            title = action.get_text()
            link = action.find('a')['href']
            date_text = action.find_next('span',
 ↪  class_='text-base-dark').get_text()
            date = datetime.strptime(date_text, '%B %d, %Y')
            if date < target_date:
                break
            category = "Unknown Category"
            agency = "No agency 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')
                li_tags = ul_tag.find_all('li') if ul_tag else []

                if len(li_tags) >= 2 and 'Agency:' in li_tags[1].text:
                    agency = li_tags[1].text.split('Agency:')[1].strip()
                if len(li_tags) >= 3 and 'Enforcement Types:' in
                 ↪  li_tags[2].text:
                    category = li_tags[2].text.split('Enforcement
 ↪   Types:')[1].strip()
            all_data.append([title, date_text, category, link, agency])
        if date < target_date:
            break
        page += 1
        time.sleep(1)
    if all_data:
        df = pd.DataFrame(all_data, columns=["Title", "Date", "Category",
 ↪   "Link", "Agency"])
        filename = f"enforcement_actions_{year}_{month}.csv"
        df.to_csv(filename, index=False)

    return df
# Run the function for 2023
df = scrape_enforcement_actions(2023, 1)
```

Total Enforcement Actions in Final DataFrame: 1535

Earliest Enforcement Action Details:

Title: Podiatrist Pays $90,000 To Settle False Billing Allegations

Date: January 3, 2023

Category: Criminal and Civil Actions

Link: https://oig.hhs.gov/fraud/enforcement/fraud/enforcement/fraud/enforcement/podiatrist-pays-90000-to-settle-false-billing-allegations/

Agency: U.S. Attorney's Office, Southern District of Texas

c. Test Partner's Code (PARTNER 1)

```
# Running the unction for data  since January 2021
df = scrape_enforcement_actions(2021, 1)
```

Total enforcement actions in the final DataFrame: 3022

Earliest enforcement action scraped: Title: The United States And Tennessee Resolve Claims With Three Providers For False Claims Act Liability Relating To 'P-Stim' Devices For A Total Of $1.72 Million

Date: January 4, 2021

Category: Criminal and Civil Actions

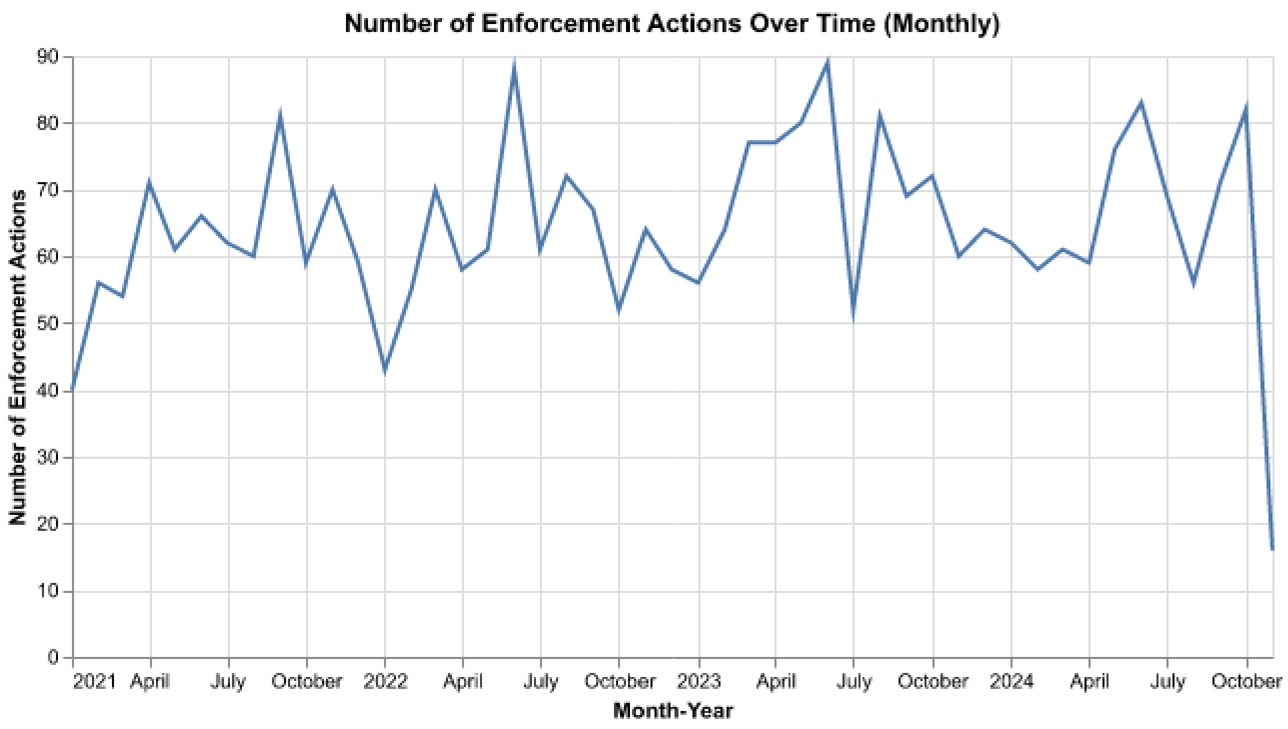
Link: https://oig.hhs.gov/fraud/enforcement/fraud/enforcement/the-united-states-and-tennessee-resolve-claims-with-three-providers-for-false-claims-act-liability-relating-to-p-stim-devices-for-a-total-of-172-million/

Agency: U.S. Attorney's Office, Middle District of Tennessee

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
df =
 ↪  pd.read_csv('/Users/afreenwala/Documents/GitHub/enforcement_actions_2021_1.csv')
# Convert to datetime format
df['Date'] = pd.to_datetime(df['Date'])
# Extract Year and Month
df['YearMonth'] = df['Date'].dt.to_period('M')
# Aggregate the data by YearMonth to get the count
monthly_counts = df.groupby('YearMonth').size().reset_index(name='Count')
monthly_counts['YearMonth'] = monthly_counts['YearMonth'].dt.to_timestamp()
 ↪  # Convert to timestamp
# Creating the line chart
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=600,
    height=300
)
# Display chart
line_chart.display()
```

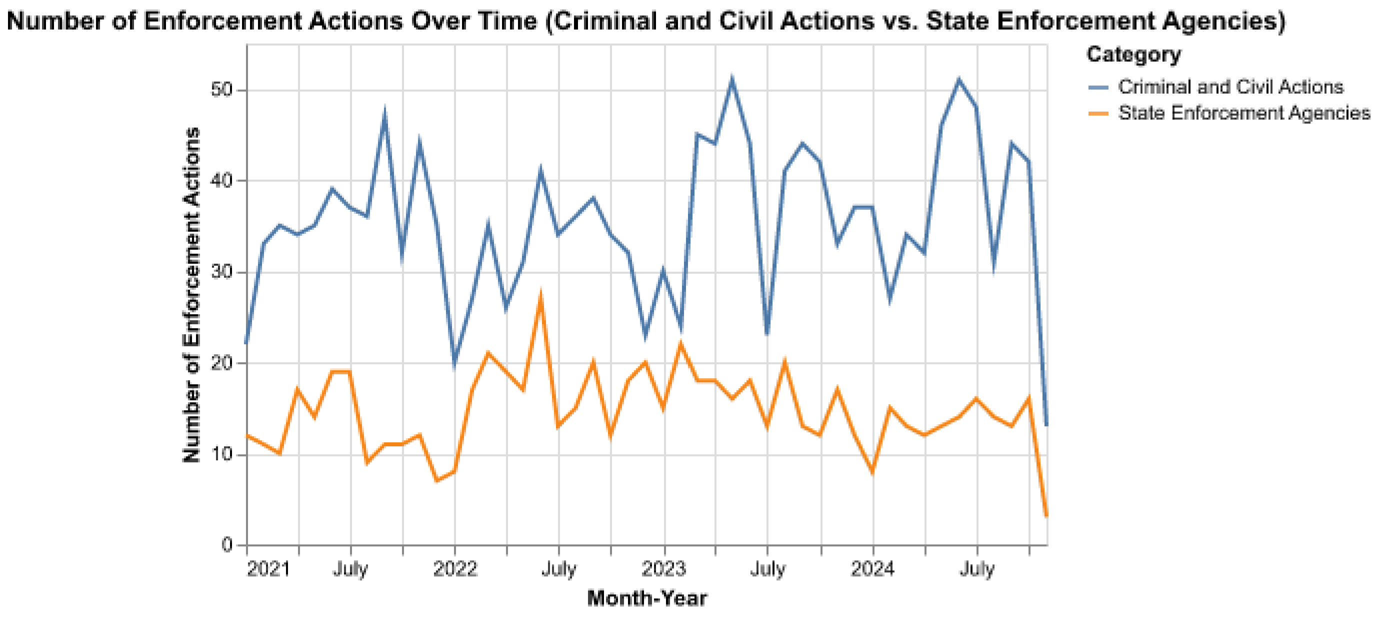


2. Plot the number of enforcement actions categorized: (PARTNER 1)

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

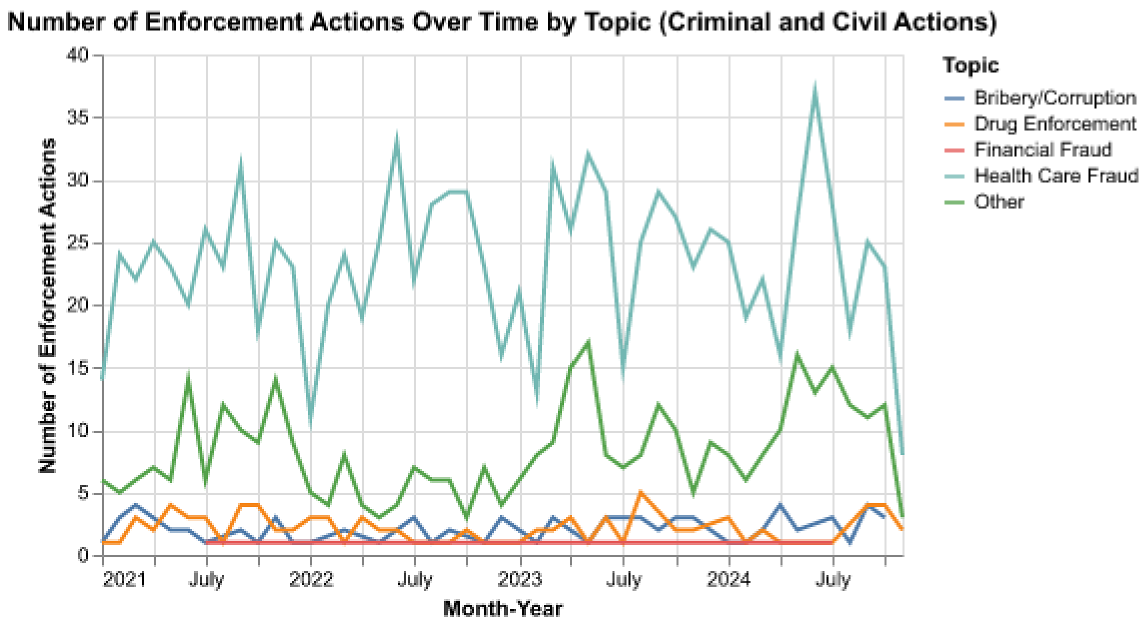
```
import pandas as pd
import altair as alt
# Loading data
df =
↪ pd.read_csv('/Users/afreenwala/Documents/GitHub/enforcement_actions_2021_1.csv')
# Convert the 'Date' column to datetime
df['Date'] = pd.to_datetime(df['Date'])
# Extract Year and Month from 'Date'
df['YearMonth'] = df['Date'].dt.to_period('M')
# Filtering  "Criminal and Civil Actions" and "State Enforcement Agencies"
↪ categories
df_filtered = df[df['Category'].isin(['Criminal and Civil Actions', 'State
↪ Enforcement Agencies'])]
# Aggregate the data by YearMonth and Category to get the count of actions
```

```
monthly_counts = df_filtered.groupby(['YearMonth',
↪  'Category']).size().reset_index(name='Count')
monthly_counts['YearMonth'] = monthly_counts['YearMonth'].dt.to_timestamp()
↪  # Convert to timestamp
# Create the line chart
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'),
    color=alt.Color('Category:N', title='Category')  # Differentiate lines by
↪  Category
).properties(
    title="Number of Enforcement Actions Over Time (Criminal and Civil
↪  Actions vs. State Enforcement Agencies)",
    width=400,
    height=250
)
# Display chart
line_chart.display()
```



Create the second line chart for topics under “Criminal and Civil Actions

```
import pandas as pd
import altair as alt
# Load the CSV data
df =
 ↪  pd.read_csv('/Users/afreenwala/Documents/GitHub/enforcement_actions_2021_1.csv')
# Convert the 'Date' column to datetime format
df['Date'] = pd.to_datetime(df['Date'])
# Extract Year and Month from 'Date'
df['YearMonth'] = df['Date'].dt.to_period('M')
# Filtering "Criminal and Civil Actions" category
df_criminal_civil = df[df['Category'] == 'Criminal and Civil Actions']
# Function to assign topic based on keywords in 'Title'
def assign_topic(title):
    title = title.lower()
    if any(keyword in title for keyword in ['health', 'medic', 'pharma',
     ↪  'care']):
        return 'Health Care Fraud'
    elif any(keyword in title for keyword in ['bank', 'financial',
     ↪  'investment', 'securities']):
        return 'Financial Fraud'
    elif any(keyword in title for keyword in ['drug', 'narcotics',
     ↪  'substance']):
        return 'Drug Enforcement'
    elif any(keyword in title for keyword in ['bribery', 'corruption',
     ↪  'kickback']):
        return 'Bribery/Corruption'
    else:
        return 'Other'
# Applying function to classify topics
df_criminal_civil['Topic'] = df_criminal_civil['Title'].apply(assign_topic)
# Aggregating  by YearMonth and Topic to get the count of actions per month
monthly_counts = df_criminal_civil.groupby(['YearMonth',
 ↪  'Topic']).size().reset_index(name='Count')
monthly_counts['YearMonth'] = monthly_counts['YearMonth'].dt.to_timestamp()
 ↪  # Convert to timestamp
# Create the line chart
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'),
    color=alt.Color('Topic:N', title='Topic')  # Differentiate lines by Topic
).properties(
    title="Number of Enforcement Actions Over Time by Topic (Criminal and
 ↪  Civil Actions)",
    width=400,
    height=250
)
# Display chart
line_chart.display()
```

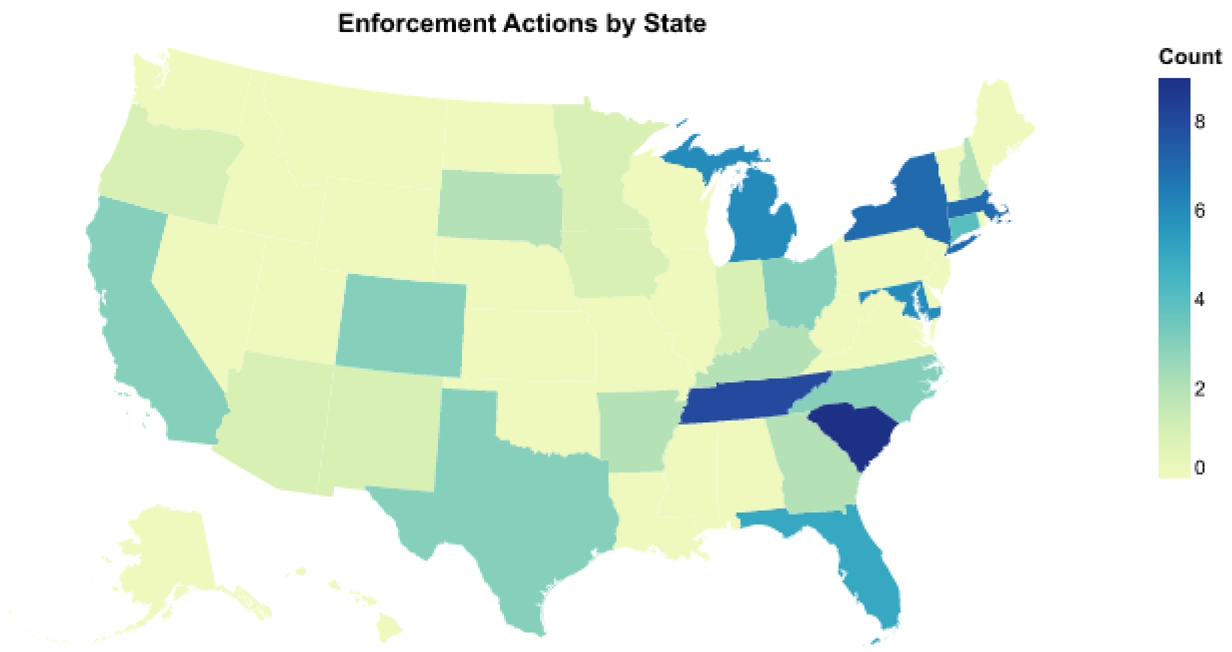


Step 4: Create maps of enforcement activity

1. Map by State (PARTNER 1)

```
import geopandas as gpd
import pandas as pd
import altair as alt
# Load the shapefile
shapefile_path =
 ↪  '/Users/afreenwala/Documents/GitHub/cb_2018_us_state_500k/cb_2018_us_state_500k.shp'
```

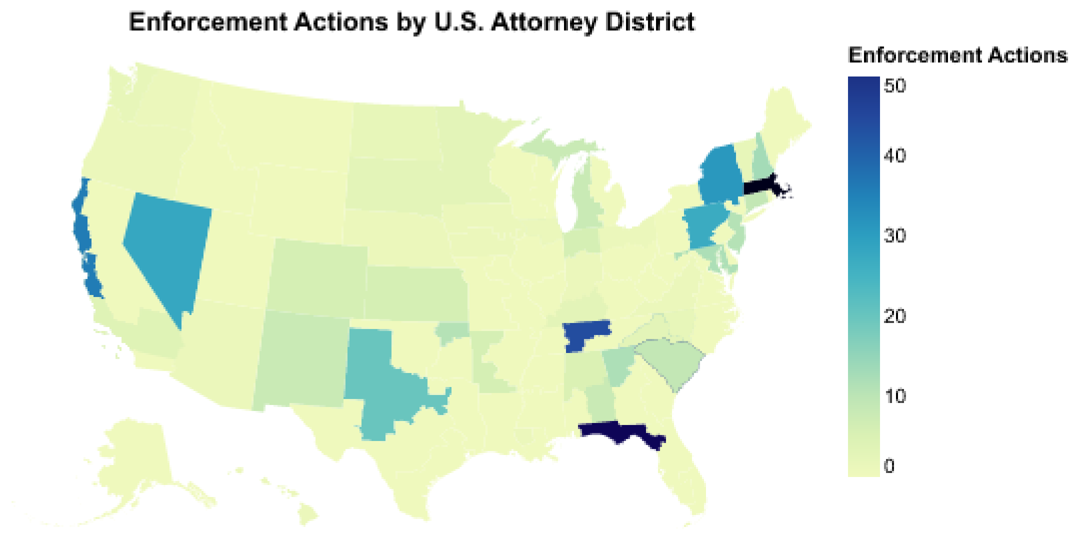
```python
gdf = gpd.read_file(shapefile_path)

# prepare your enforcement data and Filter data for 'State Enforcement
↪  Agencies'
state_df = df[df['Category'] == 'State Enforcement Agencies']
# Clean the state names from 'Agency' column to match with shapefile
def extract_state_name(agency_name):
    # Define a list of known state keywords to match
    state_keywords = ['State of', 'Attorney General']
    for keyword in state_keywords:
        if keyword in agency_name:
            return agency_name.split(keyword)[-1].strip()
    return agency_name.strip()
# Applying function to create a new 'State' column
state_df['State'] = state_df['Agency'].apply(extract_state_name)
# Count enforcement actions per state
state_counts = state_df.groupby('State').size().reset_index(name='Count')
# Merge enforcement data with shapefile data
gdf['State'] = gdf['STUSPS'].apply(lambda x: gdf.loc[gdf['STUSPS'] == x,
↪  'NAME'].values[0])
state_counts = state_counts.set_index('State')
gdf = gdf.set_index('State').join(state_counts, how='left')
gdf['Count'] = gdf['Count'].fillna(0)  # Fill missing states with 0
↪  enforcement actions

#Create the choropleth map using Altair
chart = alt.Chart(gdf).mark_geoshape().encode(
    color='Count:Q',  # Map the number of enforcement actions to color
    tooltip=['NAME:N', 'Count:Q']  # Show state name and count of enforcement
↪  actions in the tooltip
).project(
    type='albersUsa'  # Use Albers USA projection
).properties(
    title="Enforcement Actions by State",
    width=600,
    height=300
)
# Show  chart
chart.show()
```



2. Map by District (PARTNER 2)

```
import re
from fuzzywuzzy import process
import geopandas as gpd
import pandas as pd
import altair as alt

# Load shapefile for US Attorney Districts
shapefile_path = '/Users/afreenwala/Documents/GitHub/US Attorney Districts
 ↪  Shapefile
 ↪  simplified_20241109/geo_export_a617ced5-8fd4-4a45-94ca-d434014ef31f.shp'
gdf_districts = gpd.read_file(shapefile_path)

# Clean the district names for matching
def clean_district_name(name):
    if not isinstance(name, str):
        return 'Other'
    name = re.sub(r'u\.s\. attorney\'s office,', '', name.lower())
    name = re.sub(r'attorney\'s office,', '', 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')
    return ' '.join(name.split())

# Apply cleaning functions
df['cleaned_district'] = df['Agency'].apply(lambda x:
 ↪  clean_district_name(str(x)))
gdf_districts['cleaned_district'] =
 ↪  gdf_districts['judicial_d'].apply(clean_district_name)

# Count actions per district
district_action_counts = df[df['Category'] == 'State Enforcement
 ↪  Agencies'].groupby('cleaned_district').size().reset_index(name='Count')

# Match cleaned district names to US Attorney Districts
district_choices = gdf_districts['cleaned_district'].tolist()
district_action_counts['matched_district'] =
 ↪  district_action_counts['cleaned_district'].apply(lambda x:
 ↪  process.extractOne(x, district_choices)[0])

# Merge enforcement data with shapefile
merged_district_counts = gdf_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)

# Convert merged GeoDataFrame to GeoJSON
merged_district_counts_geojson =
 ↪  merged_district_counts.to_crs(epsg=4326).__geo_interface__

# Create and display Altair chart
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=400,
    height=250,
    title="Enforcement Actions by U.S. Attorney District"
).project(type='albersUsa')
```

```
alt_chart.show()

# Identify any missing districts
missing_districts = set(gdf_districts['cleaned_district']) -
 ↪  set(district_action_counts['cleaned_district'])
print(f"Missing Districts: {missing_districts}")
```



```
Missing Districts: {'east district of tennessee', 'east district of
michigan', 'north district of texas', 'district of north dakota', 'west
district of missouri', 'district of guam', 'central district of illinois',
'south district of ohio', 'east district of missouri', 'east district of
california', 'west district of michigan', 'south district of florida', 'west
district of tennessee', 'district of south carolina', 'south district of west
virginia', 'west district of oklahoma', 'east district of louisiana',
'district of nebraska', 'central district of california', 'district of
kansas', 'central district of georgia', 'central district of alabama', 'south
district of alabama', 'east district of kentucky', 'north district of
mississippi', 'south district of texas', 'district of nevada', 'east district
of pennsylvania', 'west district of arkansas', 'west district of kentucky',
'south district of mississippi', 'district of new jersey', 'district of
colorado', 'district of rhode island', 'west district of texas', 'west
district of louisiana', 'south district of new york', 'district of idaho',
'district of wyoming', 'central district of north carolina', 'district of
montana', 'north district of florida', 'district of oregon', 'west district
of washington', 'south district of california', 'north district of indiana',
'east district of wisconsin', 'district of maine', 'district of vermont',
'west district of new york', 'north district of new york', 'north district of
ohio', 'east district of virginia', 'east district of arkansas', 'central
district of tennessee', 'central district of louisiana', 'district of new
hampshire', 'district of arizona', 'east district of texas', 'east district
of oklahoma', 'district of puerto rico', 'district of north marianas
islands', 'north district of alabama', 'central district of pennsylvania',
'district of massachusetts', 'west district of pennsylvania', 'west district
of north carolina', 'district of maryland', 'west district of wisconsin',
'district of alaska', 'district of minnesota', 'district of hawaii', 'south
district of iowa', 'north district of georgia', 'east district of new york',
'district of utah', 'east district of north carolina', 'district of new
mexico', 'west district of virginia', 'north district of california',
'district of us virgin islands', 'district of delaware', 'district of
district of columbia', 'north district of oklahoma', 'south district of
illinois', 'north district of west virginia', 'district of south dakota',
'south district of georgia', 'district of connecticut', 'central district of
florida'}
```