Problem Set Four Submission

Kanika Shokeen

Invalid Date

Due 02/07 at 5:00PM Central.

"This submission is my work alone and complies with the 30538 integrity policy." Add your initials to indicate your agreement: Kanika Shokeen

```
import pandas as pd
import altair as alt
import time
import os
import numpy as np
from urllib.parse import urljoin
import re

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

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

Step 1: Develop initial scraper and crawler

```
# First Step: In the following code chunk I am downloading and saving the
 ↪  html file
import requests
import lxml
from bs4 import BeautifulSoup
url = "https://oig.hhs.gov/fraud/enforcement/"
myheader = {"User-Agent": "LEMMEINBot/1.0(kanikashokeen@uchicago.edu)"} # I
 ↪  have created my own bot and named it 'LEMMEINBot'
```

```
response = requests.get(url, headers = myheader) # I will use this bot to
 ↪  request the html file so the website can identify me, this is important
 ↪  for some websites such as wikipedia
time.sleep(2) # I have also added a brief time skip so the website does not
 ↪  block my bot
soup = BeautifulSoup(response.content, 'lxml') # beautiful soup will parse
 ↪  the response
soup.text[0:100]
```

```
'\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\nEnforcement Actions | Office of
Inspector General | Government Oversight | U.S.'
```

```
# Second Step: In the following code chunk I am extracting and refining the
 ↪  contents of the html file to arrive at the content of interest to me (a
 ↪  tidy dataframe made up of the attributes Title of enforcement action /
 ↪  Data / Category / Link)
# By visually inspecting the website, I was able to find that the relevant
 ↪  content starts with the <ul> tag, which stands for an unordered bullet
 ↪  list

results = soup.select_one("#results") # this code line is targetting the
 ↪  specific container, which happens to be under the results body
items = results.select("ul li") # this code line then looks for only list
 ↪  items inside of our specified container

data_rows = []

for li in items:
  a = li.select_one("a[href]")
  title = a.get_text(" ", strip = True) if a else None
  link = a["href"] if a else None
  date = (li.select_one("time") or li.select_one(".date"))
  cat = li.select_one(".category")
  data_rows.append({
    "title": title,
    "date": date.get_text(" ", strip = True) if date else None,
    "category": cat.get_text(" ", strip = True) if cat else None,
    "link": link,
  })

pd.DataFrame(data_rows)
```

	title	date	category	link
0	Houston Transplant Doctor Indicted For Making ...	None	None	/fraud/enforcement/houston-tra
1	None	None	None	None
2	MultiCare Health System to Pay Millions to Set...	None	None	/fraud/enforcement/multicare-h
3	None	None	None	None
4	Brooklyn Banker Pleads Guilty to Laundering Pr...	None	None	/fraud/enforcement/brooklyn-b
5	None	None	None	None
6	Delafield Man Sentenced to 18 Months' Imprison...	None	None	/fraud/enforcement/delafield-m
7	None	None	None	None
8	Former NFL Player Convicted for $197M Medicare...	None	None	/fraud/enforcement/former-nfl-
9	None	None	None	None
10	Attorney General Hanaway Obtains Medicaid Frau...	None	None	/fraud/enforcement/attorney-ge
11	None	None	None	None
12	AG's Office Secures Indictments Against Peabod...	None	None	/fraud/enforcement/ags-office-s
13	None	None	None	None
14	Florida Man Pleads Guilty to Conspiracy to Vio...	None	None	/fraud/enforcement/florida-man
15	None	None	None	None
16	Forefront Living Hospice Agreed to Pay $1.9 Mi...	None	None	/fraud/enforcement/forefront-li
17	None	None	None	None
18	Attorney General Jeff Jackson Announces Health...	None	None	/fraud/enforcement/attorney-ge
19	None	None	None	None
20	Yadkinville Woman Sentenced in Connection with...	None	None	/fraud/enforcement/yadkinville
21	None	None	None	None
22	Attorney General Labrador Announces Sentencing...	None	None	/fraud/enforcement/attorney-ge
23	None	None	None	None
24	Attorney General Hanaway Obtains Medicaid Frau...	None	None	/fraud/enforcement/attorney-ge
25	None	None	None	None
26	Holmes Regional Medical Center Agreed to Pay $...	None	None	/fraud/enforcement/holmes-reg
27	None	None	None	None
28	None	None	None	None
29	Slidell Chiropractor Sentenced for Health Care...	None	None	/fraud/enforcement/slidell-chir
30	None	None	None	None
31	None	None	None	None
32	Repeat Health Care Fraud Offender Sentenced fo...	None	None	/fraud/enforcement/repeat-heal
33	None	None	None	None
34	Scranton Heart Institute Agrees To Pay $48,709...	None	None	/fraud/enforcement/scranton-he
35	None	None	None	None
36	Rheumatologist Agrees To Resolve False Claims ...	None	None	/fraud/enforcement/rheumatolo
37	None	None	None	None
38	Attorney General James Uthmeier Announces Arre...	None	None	/fraud/enforcement/attorney-ge
39	None	None	None	None
40	Cordell Memorial Hospital Agreed to Pay $40,00...	None	None	/fraud/enforcement/cordell-mem

	title	date	category	link
41	None	None	None	None
42	None	None	None	None
43	1	None	None	?page=1
44	2	None	None	?page=2
45	None	None	None	None
46	536	None	None	?page=536
47	Next ›	None	None	?page=2

Running through the first iteration of this scraper, I can see that the date category is not being parsed appropriately. I think this is because the wrong tags were used in the scraper. I have shown my second attempt below.

```
cards = soup.select("li.usa-card.card--list")
rows = []

for card in cards:
    a = card.select_one("h2.usa-card__heading a[href]")
    date_el = card.select_one("div.font-body-sm span.text-base-dark")
    cat_el = card.select_one("div.font-body-sm li.usa-tag")

    rows.append({
        "title": a.get_text(strip=True) if a else None,
        "date": date_el.get_text(strip=True) if date_el else None,
        "category": cat_el.get_text(strip=True) if cat_el else None,
        "link": a["href"] if a else None
    })

pd.DataFrame(rows)
```

	title	date	category
0	Houston Transplant Doctor Indicted For Making ...	February 5, 2026	Criminal and Civil Actions
1	MultiCare Health System to Pay Millions to Set...	February 4, 2026	Criminal and Civil Actions
2	Brooklyn Banker Pleads Guilty to Laundering Pr...	February 3, 2026	COVID-19
3	Delafield Man Sentenced to 18 Months' Imprison...	February 3, 2026	Criminal and Civil Actions
4	Former NFL Player Convicted for $197M Medicare...	February 3, 2026	Criminal and Civil Actions
5	Attorney General Hanaway Obtains Medicaid Frau...	February 3, 2026	State Enforcement Agencies
6	AG's Office Secures Indictments Against Peabod...	February 2, 2026	State Enforcement Agencies
7	Florida Man Pleads Guilty to Conspiracy to Vio...	January 30, 2026	Criminal and Civil Actions
8	Forefront Living Hospice Agreed to Pay $1.9 Mi...	January 30, 2026	Fraud Self-Disclosures

	title	date	category
9	Attorney General Jeff Jackson Announces Health...	January 30, 2026	State Enforcement Agencies
10	Yadkinville Woman Sentenced in Connection with...	January 29, 2026	Criminal and Civil Actions
11	Attorney General Labrador Announces Sentencing...	January 29, 2026	State Enforcement Agencies
12	Attorney General Hanaway Obtains Medicaid Frau...	January 29, 2026	State Enforcement Agencies
13	Holmes Regional Medical Center Agreed to Pay $...	January 28, 2026	CMP and Affirmative Exclusio
14	Slidell Chiropractor Sentenced for Health Care...	January 28, 2026	COVID-19
15	Repeat Health Care Fraud Offender Sentenced fo...	January 28, 2026	Criminal and Civil Actions
16	Scranton Heart Institute Agrees To Pay $48,709...	January 28, 2026	Criminal and Civil Actions
17	Rheumatologist Agrees To Resolve False Claims ...	January 28, 2026	Criminal and Civil Actions
18	Attorney General James Uthmeier Announces Arre...	January 28, 2026	State Enforcement Agencies
19	Cordell Memorial Hospital Agreed to Pay $40,00...	January 27, 2026	CMP and Affirmative Exclusio

But now, I'm noticing there are only 19 rows here, and it is not looking through the next couple pages. I am going to try to tackle this in the next section of this assignment.

Step 2: Making the scraper dynamic

1. Turning the scraper into a function

- a. Pseudo-Code I will create three values, a start_month which will be January, a start_year which will be 2013 and a run_scraper (True/False) flag.

I will then create a new column called page, which will currently remain empty.

In the loop, I want the crawler to visit each page URL (…/enforcement/) for page 1, then (…/enforcement/?page=2) and so forth. I will use a 'while' loop, as long as the start_year >= 2013 condition is not met.

I will add a parse function, which will extract title/date/category/link from each card. Make sure there is a stop function in addition to the 'while' condition that ensures my crawler stops once the start year condition is met, (or if the page has no cards).

Add a time.sleep between pages to ensure I'm not blocked by the website.

Combine it all into a dataframe, filter again to date >= start_date, save to enforcement_actions_{year}_{month}.csv.

- b. Create Dynamic Scraper

```python
def scrape_enforcement_actions(start_year: int,
                               start_month: int,
                               run_scraper: bool = True,
                               outdir: str = ".",
                               user_agent: str =
 ↪  "LEMMEINBot/1.0(kanikashokeen@uchicago.edu)",
                               sleep_seconds: float = 1.0) -> pd.DataFrame:
    # If run_scraper=False, tries to load the saved CSV instead of scraping

    # ---- Validate year constraint ----
    if start_year < 2013:
        print("Please restrict to start_year >= 2013 (only enforcement
         ↪  actions after 2013 are listed).")
        return pd.DataFrame()

    # ---- Output filename ----
    csv_name = f"enforcement_actions_{start_year}_{start_month:02d}.csv"
    csv_path = os.path.join(outdir, csv_name)

    # ---- If not running scraper, load existing CSV ----
    if not run_scraper:
        if os.path.exists(csv_path):
            df = pd.read_csv(csv_path)
            # optional: parse date back to datetime
            if "date" in df.columns:
                df["date"] = pd.to_datetime(df["date"], errors="coerce")
            return df
        else:
            print(f"run_scraper=False but file not found: {csv_path}")
            return pd.DataFrame()

    # ---- Setup ----
    base_url = "https://oig.hhs.gov/fraud/enforcement/"
    headers = {"User-Agent": user_agent}
    start_date = pd.Timestamp(year=start_year, month=start_month, day=1)

    all_rows = []
    page = 1

    # ---- Crawl pages until stop condition met ----
    while True:
        url = base_url if page == 1 else f"{base_url}?page={page}"
        resp = requests.get(url, headers=headers, timeout=30)
```

```
        resp.raise_for_status()
        soup = BeautifulSoup(resp.content, "lxml")
        cards = soup.select("li.usa-card.card--list")

        # Stop if the page has no cards (end of pagination / unexpected page)
        if not cards:
            break
        page_dates = []

        for card in cards:
            a = card.select_one("h2.usa-card__heading a[href]")
            date_el = card.select_one("div.font-body-sm span.text-base-dark")
            cat_el = card.select_one("div.font-body-sm li.usa-tag")
            title = a.get_text(strip=True) if a else None
            link = urljoin(base_url, a["href"]) if a and a.has_attr("href")
↪   else None
            date_text = date_el.get_text(strip=True) if date_el else None
            category = cat_el.get_text(strip=True) if cat_el else None

            # Parse date into datetime for filtering/stop logic
            date_dt = pd.to_datetime(date_text, errors="coerce")

            all_rows.append({
                "title": title,
                "date": date_dt,
                "category": category,
                "link": link,
                "page": page
            })

            if pd.notna(date_dt):
                page_dates.append(date_dt)

        # Stop condition: once the OLDEST date on this page is earlier than
        ↪  the start_date,
        # we can stop crawling after this page (because later pages are even
        ↪  older).
        if page_dates:
            oldest_on_page = min(page_dates)
            if oldest_on_page < start_date:
                break

        # Next page + polite delay
```

```
        page += 1
        time.sleep(sleep_seconds)

    # ---- Build dataframe + filter to start_date ----
    df = pd.DataFrame(all_rows)

    # Keep only rows with valid dates and within range
    df = df[df["date"].notna()].copy()
    df = df[df["date"] >= start_date].copy()

    # Sort newest to oldest (optional)
    df = df.sort_values(["date", "title"], ascending=[False,
↪   True]).reset_index(drop=True)

    # Save CSV (do not commit to git)
    os.makedirs(outdir, exist_ok=True)
    df.to_csv(csv_path, index=False)

    return df
```

```
# Control flag so knitting doesn't re-scrape:
run_scraper = False  # set to False after you've generated the CSV once

# (b) since January 2024 ->
df_2024 = scrape_enforcement_actions(2024, 1, run_scraper=run_scraper)
print("Rows since Jan 2024:", len(df_2024))
if len(df_2024) > 0:
    earliest_2024 = df_2024.sort_values("date", ascending=True).iloc[0]
    print("Earliest since Jan 2024:", earliest_2024["date"].date(), "-",
     ↪  earliest_2024["title"], "-", earliest_2024["category"], "-",
     ↪  earliest_2024["link"])
```

```
Rows since Jan 2024: 1787
Earliest since Jan 2024: 2024-01-03 - Laredo Resident Admits To Impersonating
Licensed Nurse - Criminal and Civil Actions -
https://oig.hhs.gov/fraud/enforcement/laredo-resident-admits-to-impersonating-licensed-nurse/
```

I got 1788 rows, thus 1787 enforcement actions in my dataframe.

The details of the earliest action I got are:

2024-01-03 Laredo Resident Admits To Impersonating Licensed Nurse Criminal and Civil Actions https://oig.hhs.gov/fraud/enforcement/laredo-resident-admits-to-impersonating-licensed-

nurse/

- c. Test Your Code

```
run_scraper = False

# (c) since January 2022 ]
df_2022 = scrape_enforcement_actions(2022, 1, run_scraper=run_scraper)
print("Rows since Jan 2022:", len(df_2022))
if len(df_2022) > 0:
    earliest_2022 = df_2022.sort_values("date", ascending=True).iloc[0]
    print("Earliest since Jan 2022:", earliest_2022["date"].date(), "-",
     ↪  earliest_2022["title"], "-", earliest_2022["category"], "-",
     ↪  earliest_2022["link"])
```

```
Rows since Jan 2022: 3377
Earliest since Jan 2022: 2022-01-04 - Ohio home healthcare provider agrees to
pay $500,000 as part of False Claims Act settlement - Criminal and Civil
Actions -
https://oig.hhs.gov/fraud/enforcement/ohio-home-healthcare-provider-agrees-to-pay-500000-as-p
```

I got 3378 rows, thus 3377 enforcement actions in this second dataframe.

The details of the earliest action I got are:

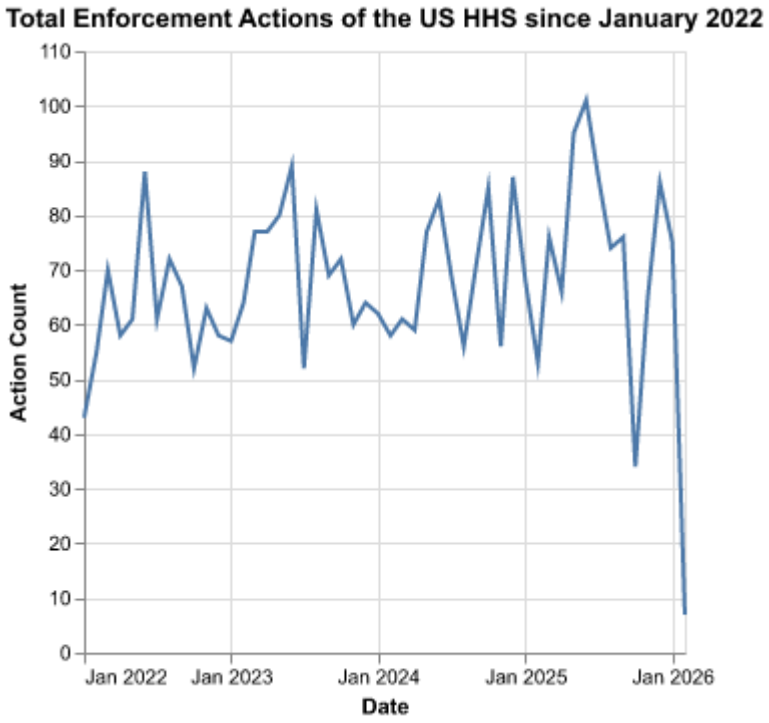
2022-01-04 Ohio home healthcare provider agrees to pay $500,000 as part of False Claims Act settlement Criminal and Civil Actions https://oig.hhs.gov/fraud/enforcement/ohio-home-healthcare-provider-agrees-to-pay-500000-as-part-of-false-claims-act-settlement/

Step 3: Plot data based on scraped data

1. Plot the number of enforcement actions over time

```
df = pd.read_csv('/Users/koniks/Desktop/GitHub
 ↪  Folder/ps4-shokeenkanika/enforcement_actions_2022_01.csv',
 ↪  parse_dates=["date"])
Chart = alt.Chart(df).mark_line().encode(
    alt.Y('count():Q', title = "Action Count"),
    alt.X('date:T', timeUnit = "yearmonth", title = "Date")
).properties(title = "Total Enforcement Actions of the US HHS since January
 ↪  2022")

Chart
```

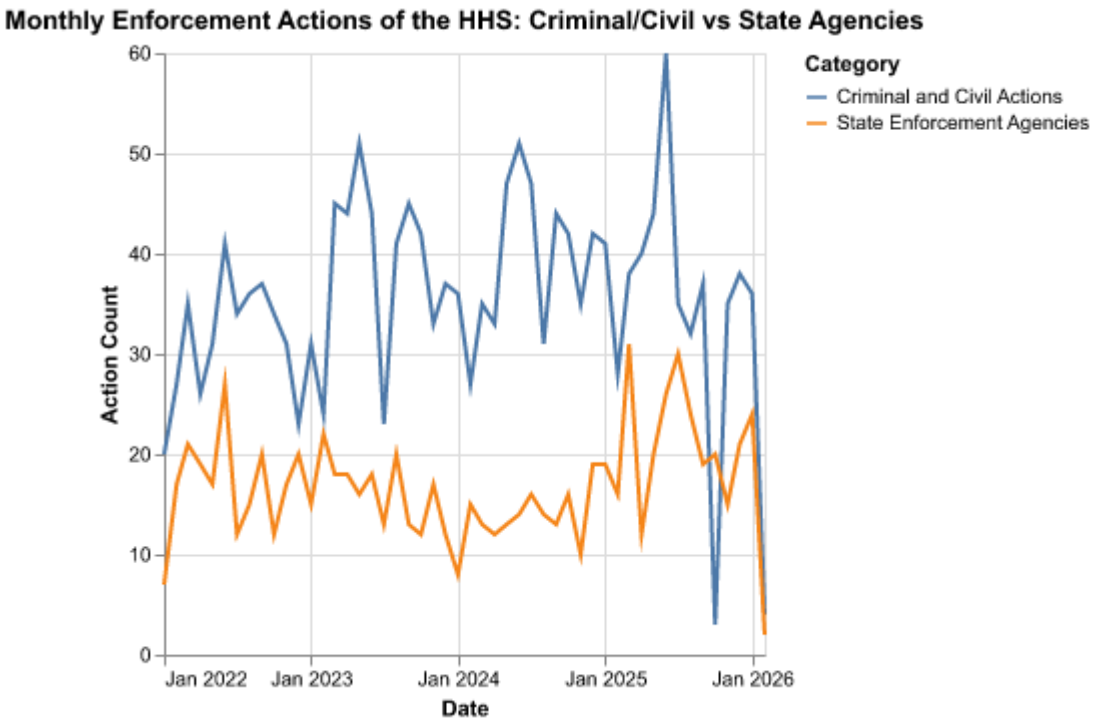


2. Plot the number of enforcement actions categorized:

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

```
chart = (
    alt.Chart(df).transform_filter(
        alt.FieldOneOfPredicate(field="category", oneOf=[
            "Criminal and Civil Actions",
            "State Enforcement Agencies"
        ])
    ).mark_line().encode(
        x=alt.X("date:T", timeUnit="yearmonth", title="Date"),
        y=alt.Y("count():Q", title="Action Count"),
        color=alt.Color("category:N", title="Category")
    ).properties(title="Monthly Enforcement Actions of the HHS:
↪   Criminal/Civil vs State Agencies")
)

chart
```



- based on five topics

```
df["title"] = df["title"].astype(str)

# Only classify within Criminal and Civil Actions
cca = df["category"].eq("Criminal and Civil Actions")

# Keyword patterns (regex). Order matters: first match wins.
TOPIC_RULES = [
    # 1) Drug Enforcement (expanded)
    ("Drug Enforcement",
     r"\b("
      ↪  r"opioid|fentanyl|heroin|oxy(codone)?|hydrocodone|percocet|vicodin|suboxone|methado
     r"controlled substance(s)?|pill mill|drug trafficking|distribution of|"
     r"illegally (prescrib|dispens)|unlawful (prescrib|dispens)|"
     r"prescription(s)?\b.*\b(traffick|fraud|illegal)|"
     r"pharmaceutic(al)?\b.*\b(diversion|distribution|wholesale)|"
     r"dea\b"
     r")\b"),

    # 2) Bribery/Corruption (expanded; keep kickback here if you want it
     ↪  treated as "corruption")
```

```
    ("Bribery/Corruption",
     r"\b("
     r"brib(e|ery)|corrupt(ion)?|extort(ion)?|gratuities|payoff|"
     r"kickback(s)?|illegal remuneration|bid rig(ging)?|"
     r"public official|official misconduct"
     r")\b"),

    # 3) Financial Fraud (expanded beyond "bank")
    ("Financial Fraud",
     r"\b("
     r"bank|banker|wire fraud|money laundering|launder(ing)?|"
     r"financial|tax\b|irs\b|mortgage|loan|lender|credit|"
     r"identity theft|embezzl(e|ement)|forg(ery|ed)|counterfeit|"
     r"securities|investment|ponzi|crypto|bitcoin|"
     r"theft of (federal|government) funds|stole federal funds|"
     r"telemarketing|marketing scheme|sweepstakes|"
     r"insurance\b|brokerage\b|premium(s)?|claims adjuster|"
     r"passport fraud|visa fraud|document fraud"
     r")\b"),

    # 4) Health Care Fraud (expanded for lots of "health system / telehealth
     ↪  / device / specialty" titles)
    ("Health Care Fraud",
     r"\b("
     r"medicare|medicaid|hhs\b|cms\b|"
     r"health care|healthcare|health system|hospital|medical center|"
     r"clinic|practice|physician(s)?|doctor(s)?|nurse(s)?|oncolog(ist|y)|"
     r"rehab(ilitation)?|physical therapy|therapy\b|"
     r"hospice|home health|nursing home|long[- ]term care|"
     r"telehealth|telemedicine|digital health|"
     r"diagnostic(s)?|laborator(y|ies)|testing\b|"
     r"medical device|device company|dme\b|durable medical|"
     r"wound\b|wound care|cancer\b|"
     r"billing|claims\b|false claims|overbill(ing)?|upcod(ing|ed)|"
     r"anti[- ]kickback|stark\b|patient\b|provider\b"
     r")\b"),
]

def classify_topic(title: str) -> str:
    t = title.lower()
    for topic, pat in TOPIC_RULES:
        if re.search(pat, t, flags=re.IGNORECASE):
            return topic
```

```
    return "Other"

df["topic"] = np.where(cca, df["title"].apply(classify_topic), np.nan)

# Quick check: topic counts inside CCA
topic_counts = df.loc[cca, "topic"].value_counts(dropna=False)
topic_counts
```

```
topic
Health Care Fraud     1089
Bribery/Corruption     237
Other                  174
Drug Enforcement       148
Financial Fraud        128
Name: count, dtype: int64
```

I ran the following code to inspect the titled still labeled 'other', and reiterated to include more keywords into each of the five buckets to arrive at the binning rule above:

df.loc[cca & (df[“topic”] == “Other”), [“date”, “title”]].head(30)

```
# Parse dates
df["date"] = pd.to_datetime(df["date"], errors="coerce")

# Keep only Criminal and Civil Actions
chart = (
    alt.Chart(df.dropna(subset=["date"]))
    .transform_filter(
        alt.datum.category == "Criminal and Civil Actions"
    )
    .mark_line()
    .encode(
        x=alt.X(
            "date:T",
            timeUnit="yearmonth",
            title="Month-Year"
        ),
        y=alt.Y(
            "count():Q",
            title="Number of Enforcement Actions"
        ),
        color=alt.Color(
            "topic:N",
```

```
            title="Topic"
        )
    ).properties(
        title="Criminal and Civil Enforcement Actions by Topic (Monthly)"
    )
)

chart
```