Analysis of Traffic Violations and Accidents in Montgomery County, Maryland

Louis Bailey

Data:

The dataset being used is *Traffic Violations* from data.gov. The dataset contains information from all electronic traffic violations issued in Montgomery County, Maryland from 2012 to 2024. It has 1.94 million rows, 43 columns, and contains features like ‘Date Of Stop’, ‘Location’, ‘Belts’, ‘Personal Injury’, and ‘Alcohol’.

Goal:

The goal of this project is to analyze the data for key insights which can be used to create recommendations on how to reduce automobile related accidents and fatalities in the county.

The Machine:

My local machine has a 7th generation i5 processor with 8GB of RAM and 2 cores. Because this not a small dataset, this notebook is being executed though a Google Cloud virtual machine with 8vCPUs, 30GB of RAM and AMD Rome Processors. This VM is a substantial upgrade from my local machine.

libraries

In [1]:
```
import requests
import json

import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns

import folium
import geopandas as gpd
```

1 | Import Data

API call

In [2]:
```
with open('data.gov_api.json') as f:
    keys         = json.load(f)
    data_gov_api = keys['data.gov_api']
```

In [3]:
```
serviceurl  = 'https://data.montgomerycountymd.gov/api/views/4mse-ku6q/rows.json?accessType=DOWNLOAD'
apikey      = '&apikey='+ data_gov_api
```

response = requests.get(serviceurl+apikey)
if response.status_code == 200: json_data = response.json() columns = [column['name'] for column in json_data['meta']['view']['columns']] rows = json_data['data'] df = pd.DataFrame(rows, columns=columns)

save as csv

df.to_csv('montgomery_county_data.csv', index=False)

pandas dataframe

In [4]:
```
df = pd.read_csv('montgomery_county_data.csv', low_memory=False)
df.head(3)
```

Out[4]:

	sid	id	position	created_at	created_meta	updated_at	updated_meta	meta	SeqID	Date Of Stop	...	DL State	
0	row-4yyf_hbzs-xeyq	00000000-0000-0000-0334-5320A443365E	0	1699524615	NaN	1699697416	NaN	{ }	345bbc5d-d407-45b4-a51f-2b4adacf9622	2023-11-08T00:00:00	...	MD	
1	row-6byx-qh5g~53st	00000000-0000-0000-DDF0-479325FAE3D3	0	1699524615	NaN	1699697416	NaN	{ }	9ad17d0d-6783-445a-88ef-fb452e4385c6	2023-11-08T00:00:00	...	XX	Ur
2	row-cben-jf7z.t337	00000000-0000-0000-E29C-D2B55A4C6A5F	0	1699524615	NaN	1699697416	NaN	{ }	9ad17d0d-6783-445a-88ef-fb452e4385c6	2023-11-08T00:00:00	...	XX	Ur

3 rows × 58 columns

2 | Clean Data

drop unwanted columns

In [5]:

```
df.columns
```

```
Out[5]: Index(['sid', 'id', 'position', 'created_at', 'created_meta', 'updated_at',
               'updated_meta', 'meta', 'SeqID', 'Date Of Stop', 'Time Of Stop',
               'Agency', 'SubAgency', 'Description', 'Location', 'Latitude',
               'Longitude', 'Accident', 'Belts', 'Personal Injury', 'Property Damage',
               'Fatal', 'Commercial License', 'HAZMAT', 'Commercial Vehicle',
               'Alcohol', 'Work Zone', 'Search Conducted', 'Search Disposition',
               'Search Outcome', 'Search Reason', 'Search Reason For Stop',
               'Search Type', 'Search Arrest Reason', 'State', 'VehicleType', 'Year',
               'Make', 'Model', 'Color', 'Violation Type', 'Charge', 'Article',
               'Contributed To Accident', 'Race', 'Gender', 'Driver City',
               'Driver State', 'DL State', 'Arrest Type', 'Geolocation',
               'Council Districts', 'Councils', 'Communities', 'Zip Codes',
               'Municipalities', 'Council Districts_from_i23j_3mj8',
               'Council Districts 7'],
              dtype='object')
```

```
In [6]: print('Number of columns:', len(df.columns))
```

```
Number of columns: 58
```

```
In [7]: df = df.drop(['sid', 'id', 'position', 'created_at', 'created_meta', 'updated_at',
                      'updated_meta', 'meta', 'SeqID', 'Agency','Description','Location', 'Race',
                       'Geolocation', 'Council Districts', 'Councils', 'Communities',
                      'Municipalities', 'Council Districts_from_i23j_3mj8', 'Council Districts 7',
                       'Search Reason For Stop', 'Search Type', 'Zip Codes', 'Search Outcome',
                      'Search Arrest Reason', 'Search Conducted', 'Search Disposition', 'Search Reason', 'Article',
                      'Year', 'Make', 'Model', 'Color'], axis=1)
```

```
In [8]: df.columns
```

```
Out[8]: Index(['Date Of Stop', 'Time Of Stop', 'SubAgency', 'Latitude', 'Longitude',
               'Accident', 'Belts', 'Personal Injury', 'Property Damage', 'Fatal',
               'Commercial License', 'HAZMAT', 'Commercial Vehicle', 'Alcohol',
               'Work Zone', 'State', 'VehicleType', 'Violation Type', 'Charge',
               'Contributed To Accident', 'Gender', 'Driver City', 'Driver State',
               'DL State', 'Arrest Type'],
              dtype='object')
```

```
In [9]: print('Number of columns:', len(df.columns))
```

```
Number of columns: 25
```

shape, missing and duplicates

```
In [10]: missing = df.isna().sum()
```

```
In [11]: print(f'shape:{df.shape}\n')
         print(f'missing:\n{missing[missing>0]}\n')
         print(f'duplicates:{df.duplicated().sum()}')
```

```
shape:(1939773, 25)

missing:
State             59
Driver City      483
Driver State      11
DL State         929
dtype: int64

duplicates:11050
```

```
In [12]: df = df.dropna()
         df = df.drop_duplicates()
```

```
In [13]: print(f'shape:{df.shape}')
         print(f'missing:{df.isna().sum().sum()}')
         print(f'duplicates:{df.duplicated().sum()}')
```

```
shape:(1927268, 25)
missing:0
duplicates:0
```

data types

```
In [14]: df.dtypes
```

Out[14]:
```
Date Of Stop                object
Time Of Stop                object
SubAgency                   object
Latitude                   float64
Longitude                  float64
Accident                    object
Belts                       object
Personal Injury             object
Property Damage             object
Fatal                       object
Commercial License          object
HAZMAT                      object
Commercial Vehicle          object
Alcohol                     object
Work Zone                   object
State                       object
VehicleType                 object
Violation Type              object
Charge                      object
Contributed To Accident       bool
Gender                      object
Driver City                 object
Driver State                object
DL State                    object
Arrest Type                 object
dtype: object
```

In [15]:
```python
df['Date Of Stop'] = pd.to_datetime(df['Date Of Stop'])
df['Time Of Stop'] = pd.to_datetime(df['Time Of Stop'], format='%H:%M:%S').dt.time
```

scanning for bad entries

In [16]:
```python
for n,col in enumerate(df.columns):
    print(f'{df.columns[n]}: {df[col].unique()}\n')
```

```
Date Of Stop: <DatetimeArray>
['2023-11-08 00:00:00', '2023-11-07 00:00:00', '2023-11-06 00:00:00',
 '2023-04-30 00:00:00', '2023-04-29 00:00:00', '2023-05-01 00:00:00',
 '2023-05-02 00:00:00', '2023-07-18 00:00:00', '2023-07-19 00:00:00',
 '2023-07-15 00:00:00',
 ...
 '2020-04-21 00:00:00', '2021-06-20 00:00:00', '2020-03-29 00:00:00',
 '2020-03-30 00:00:00', '2022-08-11 00:00:00', '2022-08-19 00:00:00',
 '2022-08-18 00:00:00', '2020-03-25 00:00:00', '2022-08-17 00:00:00',
 '2022-08-15 00:00:00']
Length: 4590, dtype: datetime64[ns]

Time Of Stop: [datetime.time(8, 44) datetime.time(21, 30) datetime.time(9, 52) ...
 datetime.time(4, 34) datetime.time(5, 17) datetime.time(5, 22)]

SubAgency: ['3rd District, Silver Spring' '2nd District, Bethesda'
 '4th District, Wheaton' '1st District, Rockville'
 '5th District, Germantown' 'Headquarters and Special Operations'
 '6th District, Gaithersburg / Montgomery Village' 'W15' 'S15']

Latitude: [39.055453   39.08393467 39.1183055  ... 39.058412   39.09042867
 38.99496433]

Longitude: [-76.960461   -77.077504   -77.1956775  ... -76.96733283 -77.0491015
 -77.04323433]

Accident: ['No' 'Yes']

Belts: ['No' 'Yes']

Personal Injury: ['No' 'Yes']

Property Damage: ['No' 'Yes']

Fatal: ['No' 'Yes']

Commercial License: ['No' 'Yes']

HAZMAT: ['No' 'Yes']

Commercial Vehicle: ['No' 'Yes']

Alcohol: ['No' 'Yes']

Work Zone: ['No' 'Yes']
```

```
State: ['MD' 'XX' 'DC' 'VA' 'NY' 'PA' 'TX' 'IL' 'NJ' 'NC' 'CA' 'TN' 'US' 'FL'
 'AR' 'MA' 'DE' 'GA' 'WY' 'IN' 'WV' 'OR' 'MO' 'NV' 'LA' 'MS' 'SC' 'WA'
 'ME' 'OH' 'AZ' 'UT' 'KY' 'VT' 'NH' 'NM' 'MI' 'MN' 'OK' 'KS' 'AK' 'CT'
 'WI' 'ND' 'AL' 'VI' 'RI' 'ID' 'NB' 'IA' 'PE' 'CO' 'NE' 'SD' 'NU' 'ON'
 'MT' 'QC' 'HI' 'BC' 'MB' 'AB' 'PR' 'PQ' 'NF' 'NS' 'MH' 'AS' 'GU' 'IT'
 'YT' 'SK']

VehicleType: ['02 - Automobile' '05 - Light Duty Truck' '01 - Motorcycle' '19 - Moped'
 '28 - Other' '06 - Heavy Duty Truck' '03 - Station Wagon'
 '07 - Truck/Road Tractor' '08 - Recreational Vehicle' '12 - School Bus'
 '25 - Utility Trailer' '29 - Unknown' '20 - Commercial Rig'
 '10 - Transit Bus' '27 - Farm Equipment' '21 - Tandem Trailer'
 '11 - Cross Country Bus' '16 - Fire(Non-Emerg)' '04 - Limousine'
 '09 - Farm Vehicle' '18 - Police Vehicle' '23 - Travel/Home Trailer'
 '15 - Fire(Emerg)' '14 - Ambulance(Non-Emerg)' '13 - Ambulance(Emerg)'
 '26 - Boat Trailer' '18 - Police(Non-Emerg)' '22 - Mobile Home'
 '24 - Camper' '13 - Ambulance' '15 - Fire Vehicle' '14 - Ambulance'
 '17 - Police(Emerg)']

Violation Type: ['Citation' 'Warning' 'ESERO' 'SERO']

Charge: ['16-112(c)' '16-101(a1)' '21-402(a)' ... '64' '7-705(b11)' '21-1412(c)']

Contributed To Accident: [False  True]

Gender: ['M' 'F' 'U']

Driver City: ['FORT WASHINGTON' 'SILVER SPRING' 'MKIDDLETOWN' ... 'FORREST HILL'
 'LANNHAM SEABROOK' 'SUITELAND']

Driver State: ['MD' 'DC' 'NY' 'TX' 'VA' 'PA' 'MA' 'CA' 'TN' 'FL' 'WV' 'SC' 'DE' 'ND'
 'NV' 'NE' 'UT' 'NC' 'GA' 'WA' 'MO' 'NM' 'OH' 'IN' 'NJ' 'IL' 'KY' 'VT'
 'NH' 'MI' 'AZ' 'MS' 'MT' 'CO' 'XX' 'CT' 'SD' 'OR' 'AL' 'LA' 'VI' 'MN'
 'ME' 'HI' 'IA' 'WI' 'RI' 'KS' 'OK' 'ON' 'AR' 'NB' 'AK' 'WY' 'QC' 'PR'
 'US' 'MB' 'ID' 'AB' 'NF' 'GU' 'SK' 'BC' 'PQ' 'PE' 'NS' 'IT']

DL State: ['MD' 'XX' 'DC' 'NY' 'TX' 'VA' 'PA' 'MS' 'IL' 'MA' 'CA' 'TN' 'FL' 'CT'
 'WV' 'IT' 'NJ' 'NV' 'NE' 'UT' 'NC' 'DE' 'GA' 'MN' 'IA' 'WA' 'SC' 'MO'
 'NM' 'OH' 'IN' 'KY' 'VT' 'NH' 'HI' 'PE' 'ND' 'MI' 'AZ' 'CO' 'LA' 'AL'
 'MT' 'SD' 'OR' 'ME' 'MB' 'VI' 'US' 'AB' 'KS' 'WI' 'AK' 'RI' 'OK' 'ON'
 'PR' 'ID' 'AR' 'NB' 'BC' 'WY' 'QC' 'SK' 'MH' 'PQ' 'AS' 'NF' 'GU' 'NS'
 'YT' 'NU']

Arrest Type: ['A - Marked Patrol' 'B - Unmarked Patrol' 'R - Unmarked Laser'
'Q - Marked Laser' 'L - Motorcycle' 'M - Marked (Off-Duty)'
'G - Marked Moving Radar (Stationary)' 'I - Marked Moving Radar (Moving)'
'E - Marked Stationary Radar' 'F - Unmarked Stationary Radar'
'O - Foot Patrol' 'H - Unmarked Moving Radar (Stationary)'
'S - License Plate Recognition' 'P - Mounted Patrol'
'N - Unmarked (Off-Duty)' 'C - Marked VASCAR'
'J - Unmarked Moving Radar (Moving)' 'D - Unmarked VASCAR'
'K - Aircraft Assist']
```

> Driver State and DL State appear to contain US states, US territories and Canadian territories. 'XX' is likely a placeholder or unknown, and 'IT' could mean Italy.

---

3 | Exploratory Analysis

number of incidents by year

In [17]:

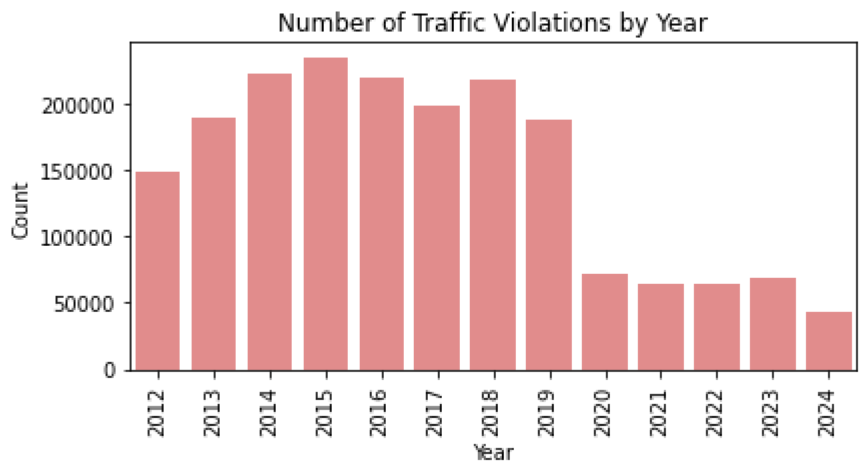
```
df['Year']     = df['Date Of Stop'].dt.year
yearly_counts = df['Year'].value_counts().sort_index()

accidents_yes_df = df[df['Accident'] == 'Yes']
injury_yes_df    = df[df['Personal Injury'] == 'Yes']
damage_yes_df    = df[df['Property Damage'] == 'Yes']
fatal_yes_df     = df[df['Fatal'] == 'Yes']
alcohol_yes_df   = df[df['Alcohol'] == 'Yes']

accidents_yes_counts = accidents_yes_df['Year'].value_counts().sort_index()
injury_yes_counts    = injury_yes_df['Year'].value_counts().sort_index()
damage_yes_counts    = damage_yes_df['Year'].value_counts().sort_index()
fatal_yes_counts     = fatal_yes_df['Year'].value_counts().sort_index()
alcohol_yes_counts   = alcohol_yes_df['Year'].value_counts().sort_index()
```

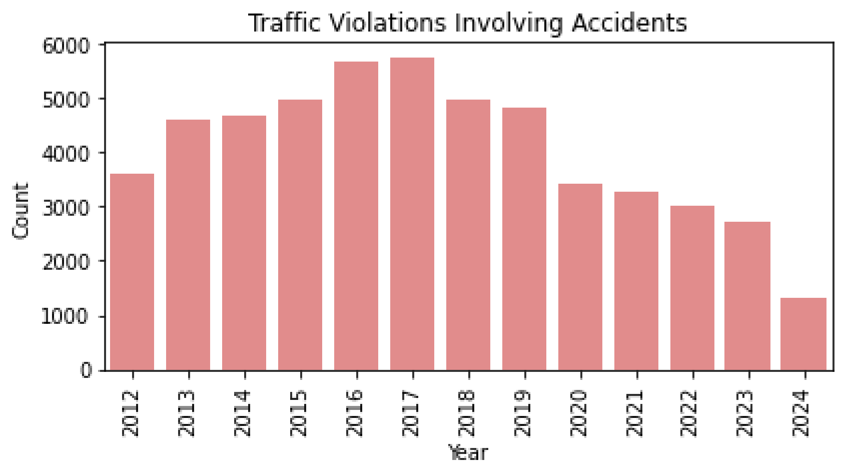
In [18]:

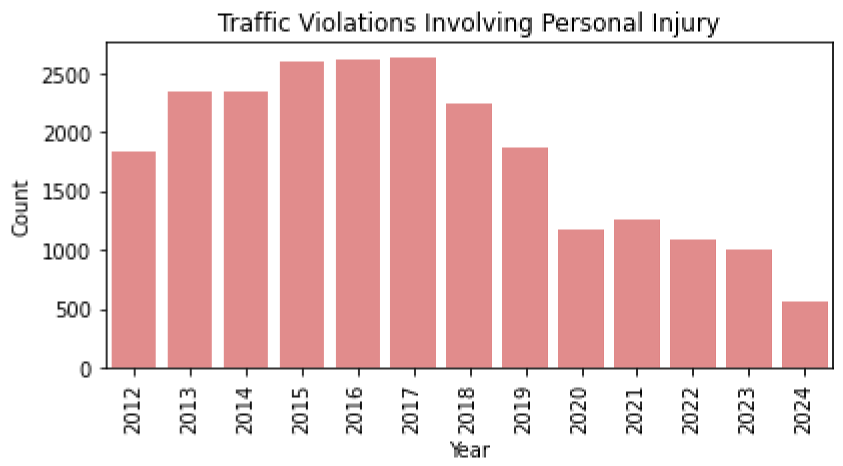
```
data = [yearly_counts, accidents_yes_counts, injury_yes_counts, damage_yes_counts, fatal_yes_counts, alcohol_yes_counts
```

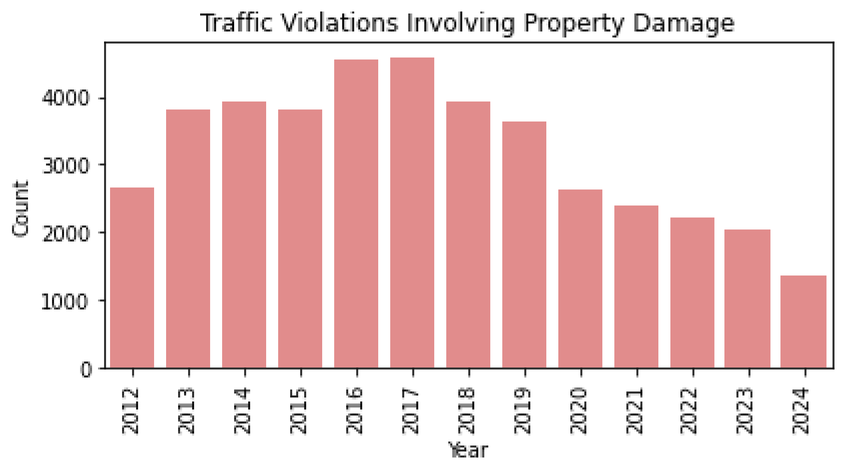
```
titles = ['Number of Traffic Violations by Year','Traffic Violations Involving Accidents',
          'Traffic Violations Involving Personal Injury','Traffic Violations Involving Property Damage',
          'Traffic Violations Involving Fatality','Traffic Violations Involving Alcohol']
```

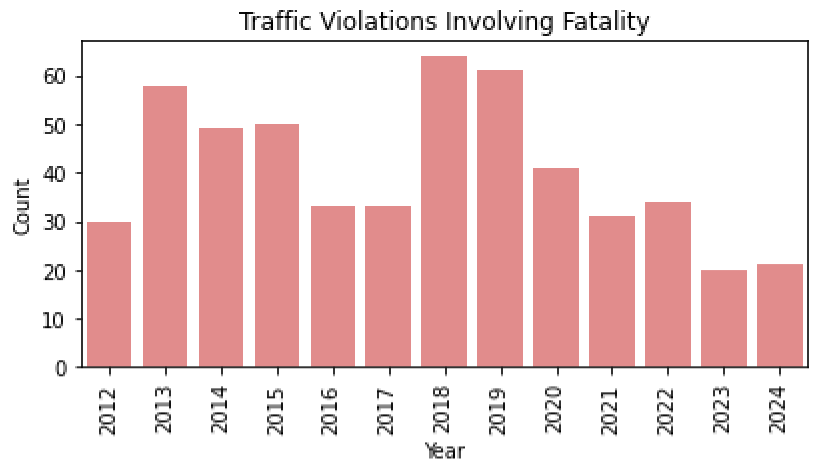
In [57]:
```
fig, axes = plt.subplots(3, 2, figsize=(12, 10))
axes = axes.flatten()

for i, count_data in enumerate(data):
    sns.barplot(x=count_data.index, y=count_data.values, color='lightcoral', ax=axes[i])
    axes[i].set_xlabel('Year')
    axes[i].set_ylabel('Count')
    axes[i].set_title(titles[i])
    axes[i].tick_params(axis='x', rotation=90)

plt.tight_layout()
plt.show()
```

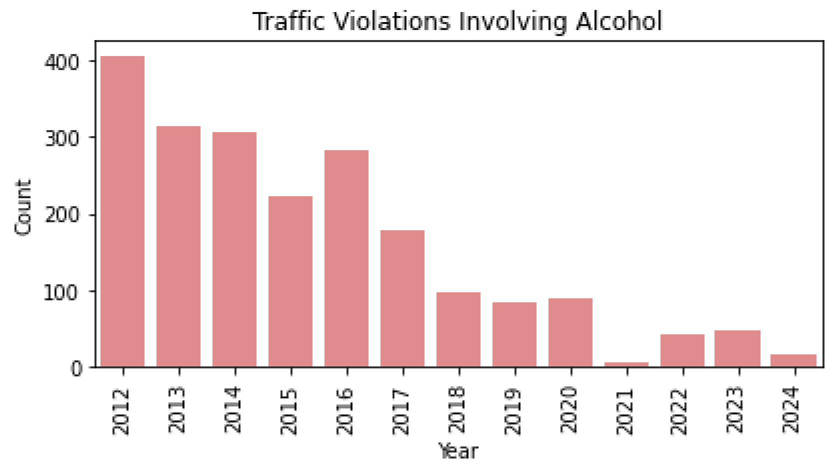

Number of Traffic Violations by Year
Count
0
50000
100000
150000
200000
Year
2012
2013
2014
2015
2016
2017
2018
2019
2020
2021
2022
2023
2024


Traffic Violations Involving Accidents
Count
0
1000
2000
3000
4000
5000
6000
Year
2012
2013
2014
2015
2016
2017
2018
2019
2020
2021
2022
2023
2024


Traffic Violations Involving Personal Injury
Count
0
500
1000
1500
2000
2500
Year
2012
2013
2014
2015
2016
2017
2018
2019
2020
2021
2022
2023
2024


Traffic Violations Involving Property Damage
Count
0
1000
2000
3000
4000
Year
2012
2013
2014
2015
2016
2017
2018
2019
2020
2021
2022
2023
2024


Traffic Violations Involving Fatality
Count
0
10
20
30
40
50
60
Year
2012
2013
2014
2015
2016
2017
2018
2019
2020
2021
2022
2023
2024

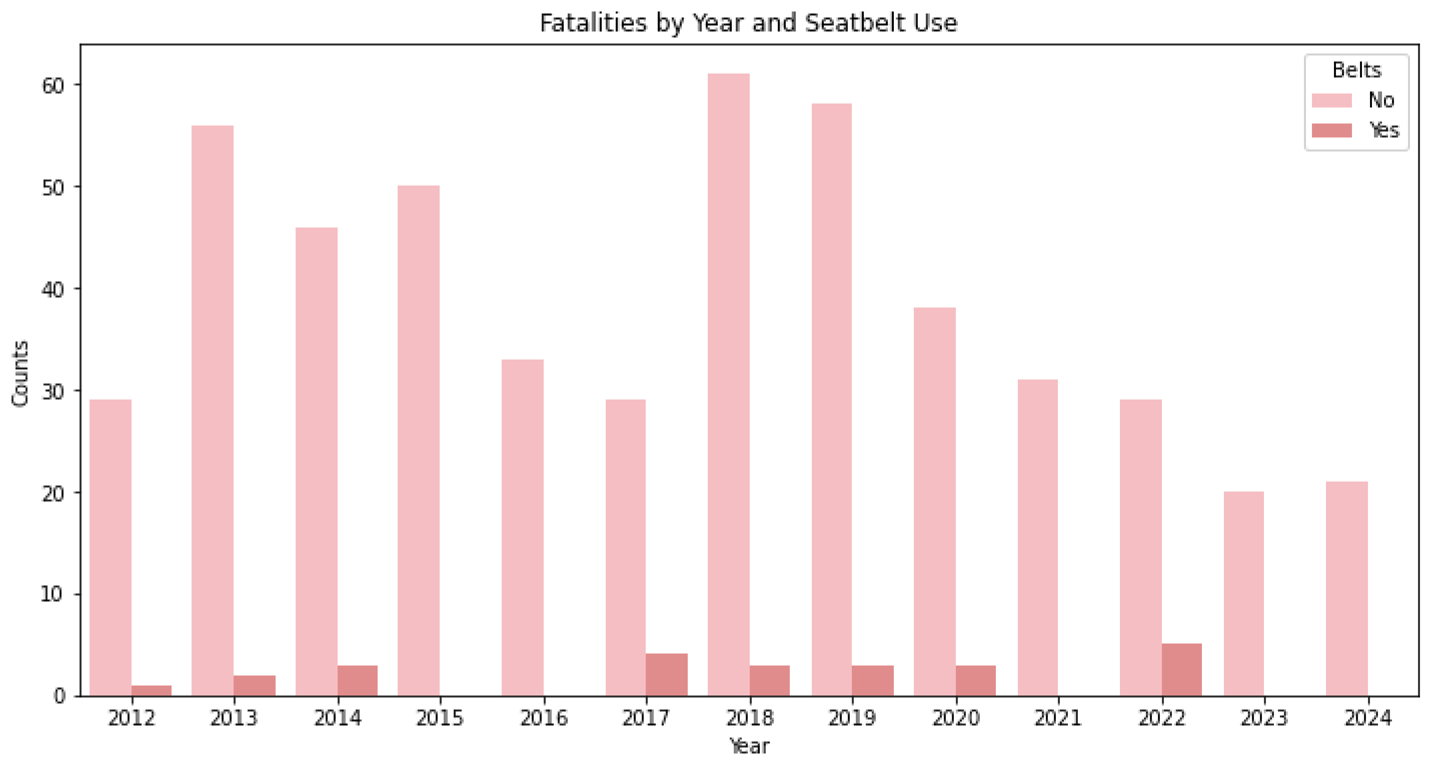

Traffic Violations Involving Alcohol
Count
0
100
200
300
400
Year
2012
2013
2014
2015
2016
2017
2018
2019
2020
2021
2022
2023
2024

> The number of traffic violations appear to have dropped dramatically since 2019 and stayed down. Because of this, it is not surprising that the occurences of traffic violations involving personal injuries, fatalities and so on are down. Notably, traffic violations involving alcohol are very low from 2021-2024.
>
> From what I could find, a possible reason for this decrease is the county's adoption of Vision Zero. The program includes various measures such as increased safety education, enhanced traffic enforcement, and improvements in road infrastructure. Additionally, there was an expansion of automated enforcement, including speed and red-light cameras. (https://montgomerycountymd.gov/visionzero/)
>
> 2024 is not complete. The data on this year goes up to July 7th (about half way through the year). The number of traffic violations being as high as they are could indicate a rise in violations relative to the past 4 years.

fatalities by year and seatbelt use

In [54]:
```
belts_fatal_df = fatal_yes_df.groupby(['Belts','Year']).Fatal.size().reset_index(name='Counts').sort_values(by=['Year']
                                                                                                            ascending=Tru
```

In [63]:
```
palette = {'No':'lightpink', 'Yes':'lightcoral'}

plt.figure(figsize=(12,6))
sns.barplot(data=belts_fatal_df,x='Year', y=belts_fatal_df.Counts, palette=palette, hue='Belts')
plt.title('Fatalities by Year and Seatbelt Use')
plt.show()
```



> This plot shows the importance of wearing a seatbelt. For almost every fatality from 2012 to 2024, the person was not wearing a seatbelt.

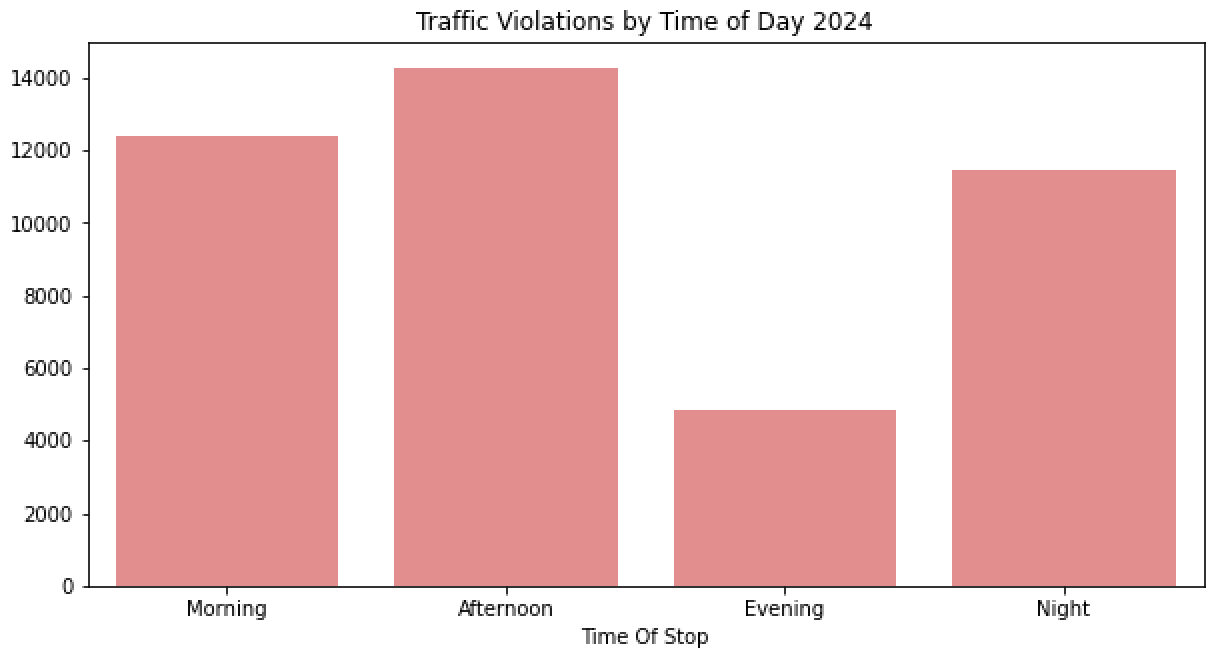
traffic violations by time of day 2024

In [25]:
```
df_2024 = df[df.Year==2024].copy()
```

In [26]:
```
night     = (pd.to_datetime('21:00:00').time(), pd.to_datetime('05:00:00').time())
morning   = (pd.to_datetime('05:00:00').time(), pd.to_datetime('12:00:00').time())
afternoon = (pd.to_datetime('12:00:00').time(), pd.to_datetime('18:00:00').time())
evening   = (pd.to_datetime('18:00:00').time(), pd.to_datetime('20:59:59').time())

df_2024['Time Of Stop'] = np.select(
    [
        (df_2024['Time Of Stop'] >= night[0]) | (df_2024['Time Of Stop'] < night[1]),
        (df_2024['Time Of Stop'] >= morning[0]) & (df_2024['Time Of Stop'] < morning[1]),
        (df_2024['Time Of Stop'] >= afternoon[0]) & (df_2024['Time Of Stop'] < afternoon[1]),
        (df_2024['Time Of Stop'] >= evening[0]) & (df_2024['Time Of Stop'] <= evening[1])
    ],
    ['Night', 'Morning', 'Afternoon', 'Evening']
)
```

In [27]:
```
time_of_stop_counts = df_2024['Time Of Stop'].value_counts()
time_order          = ['Morning', 'Afternoon', 'Evening', 'Night']
time_of_stop_counts = time_of_stop_counts.reindex(time_order)
```

In [61]:
```
plt.figure(figsize=(10,5))
sns.barplot(x=time_of_stop_counts.keys(), y=time_of_stop_counts.values, color='lightcoral')
plt.title('Traffic Violations by Time of Day 2024')
plt.show()
```

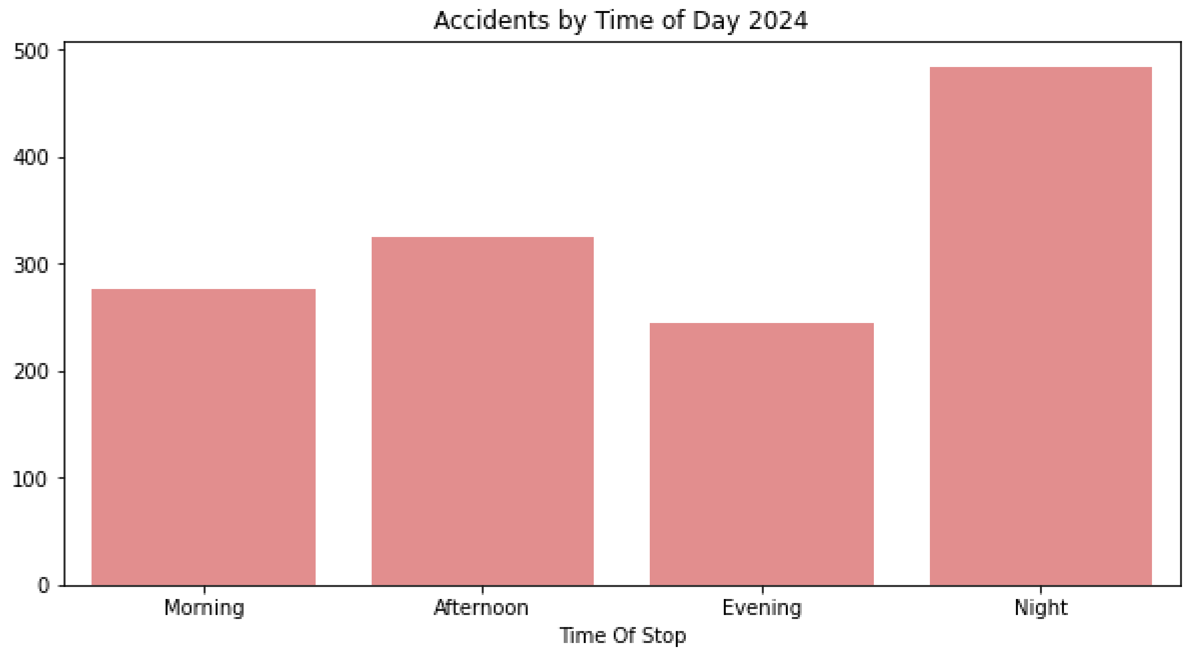


> Traffic violations appear to be highest in the afternoon.

accidents by time of day 2024

```
In [29]: accidents_yes_df_2024  = df_2024[df_2024.Accident=='Yes']
         accidents_by_time = accidents_yes_df_2024.groupby('Time Of Stop').Accident.size().reindex(time_order)
```

```
In [60]: plt.figure(figsize=(10,5))
         sns.barplot(x=accidents_by_time.keys(), y=accidents_by_time.values, color='lightcoral')
         plt.title('Accidents by Time of Day 2024')
         plt.show()
```



> Accidents appear to be highest at night.

---

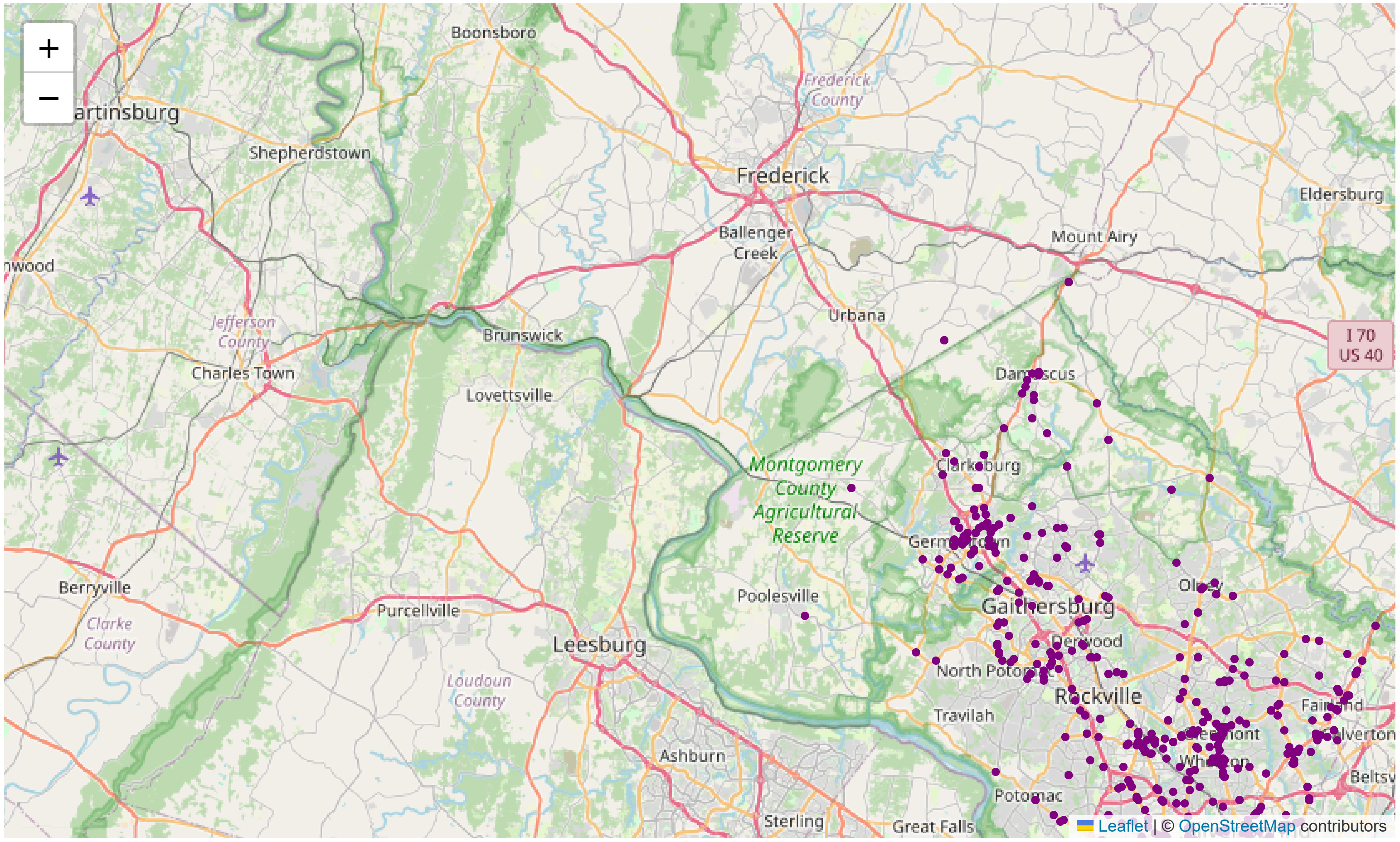
number of accidents by area 2024

In [31]:
```
accidents_2024_df = df[(df.Accident=='Yes')&(df.Year==2024)&(df.Latitude!=0)&(df.Longitude!=0)]
```

In [32]:
```
center_lat, center_lon = accidents_2024_df['Latitude'].mean(), accidents_2024_df['Longitude'].mean()
m = folium.Map(location=[center_lat, center_lon], zoom_start=10)
```

In [33]:
```
for _, row in accidents_2024_df.iterrows():
    folium.CircleMarker(
        location=[row['Latitude'], row['Longitude']],
        radius=0.5,
        color='purple',
        fill=True,
        fill_color='purple',
    ).add_to(m)

m
```

Out[33]:



> In 2024, there appears to be clusters in:

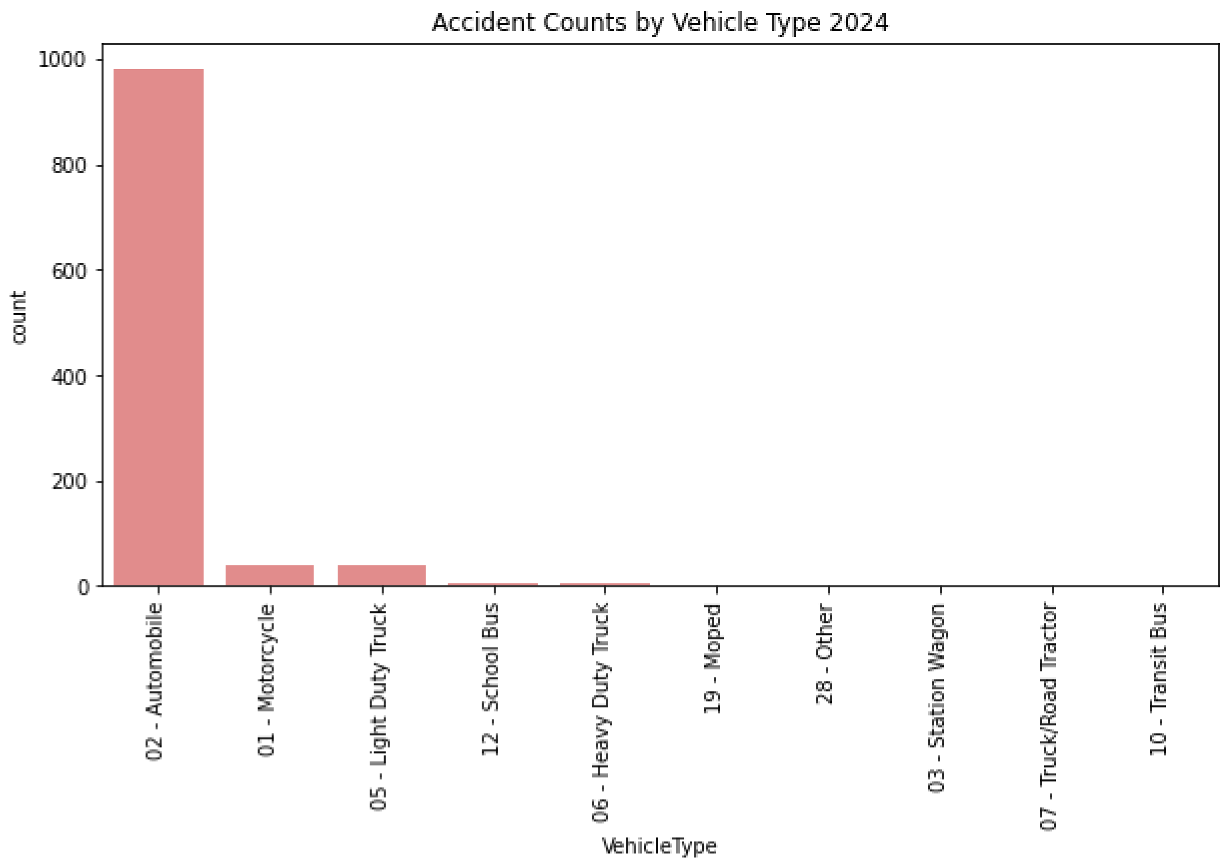
- Wheaton on University Blvd West & on Georgia Avenue near WTOP Radio Towers
- Glenmont around the 4th district police station
- Silver Spring on Georgia Avenue Northwest near AFI Silver Theater & Cultural Center
- East Bethesda around the Woodmont Rugby garage that's near the 2nd district police station
- North Bethesda down Rockville Pike near Bank of America
- GermanTown around the Germantown police station and also around the Germantown Emergency station

---

accident counts by vehicle type 2024

```
In [34]: accident_vehicles_2024    = accidents_2024_df.groupby('VehicleType').Accident.value_counts().sort_values(ascending=Fals
         accident_vehicles_2024_df = accident_vehicles_2024.reset_index(name='count')
         accident_vehicles_2024_df = accident_vehicles_2024_df[['VehicleType', 'count']]
```

```
In [59]: plt.figure(figsize=(10,5))
         sns.barplot(x=accident_vehicles_2024_df['VehicleType'], y=accident_vehicles_2024_df['count'], color='lightcoral')
         plt.title('Accident Counts by Vehicle Type 2024')
         plt.xticks(rotation=90)
         plt.show()
```



> The vast majority of accidents have involved automobiles.

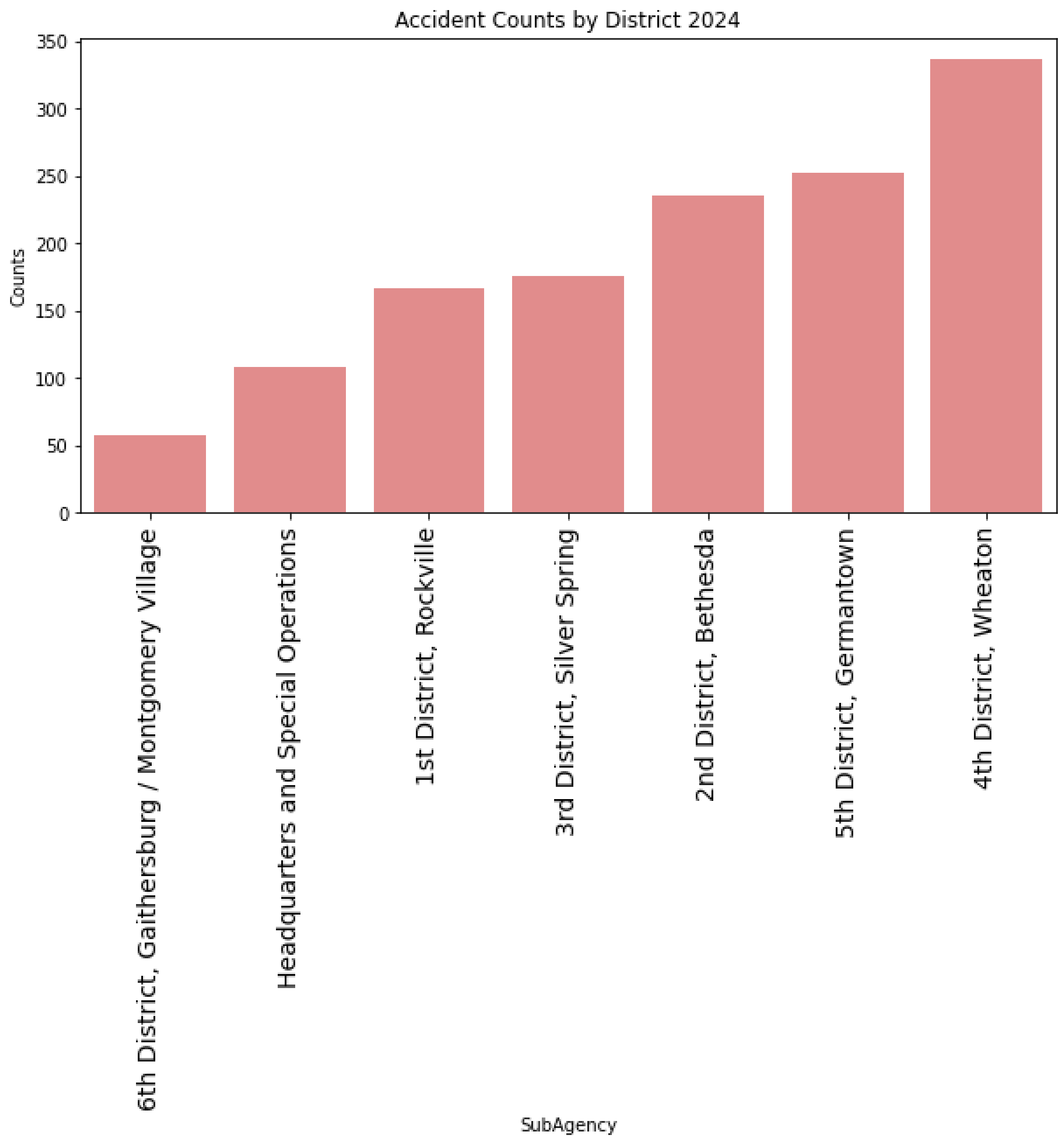
---

accident counts by district 2024

```
In [38]: accidents_yes_2024 = df[(df.Accident=='Yes')&(df.Year==2024)]
```

```
In [41]: accidents_by_district_2024 = accidents_yes_2024.groupby(['SubAgency']).Accident.size().reset_index(name='Counts').sort_
                                                                                                                ascending=True)
```

```
In [58]: plt.figure(figsize=(10,5))
         sns.barplot(data=accidents_by_district_2024,x='SubAgency', y='Counts', color='lightcoral')
         plt.title('Accident Counts by District 2024')
         plt.xticks(rotation=90, size=14)
         plt.show()
```



> The 4th district, Wheaton, has had the most accidents in 2024.

4 | Conclusion

Based on the analysis of traffic violations in Montgomery County, Maryland, here are some recommendations for policymakers:

- Enhanced Safety Measures in High-Risk Areas:

  > As listed above, certain areas in Wheaton, Glenmont, Silver Spring, East Bethesda, North Bethesda, and Germantown have been identified as high-accident areas. Implement targeted safety measures in these locations, such as increasing police presence, improving road signage, enhancing street lighting, and possibly installing additional traffic signals.

- Focus on Nighttime Accidents:

  > Accidents are most frequent at night, suggesting a need for increased enforcement during these hours. Consider extending the hours of active patrols and implementing nighttime speed limits or checkpoints to deter dangerous driving behaviors.

- Promotion of Seatbelt Use:

  > The data shows a significant correlation between fatalities and the lack of seatbelt use. Initiatives to increase seatbelt use could include educational campaigns, increased enforcement of seatbelt laws, and public service announcements highlighting the dangers of not wearing seatbelts.

- Vehicle-Specific Safety Campaigns:

  > As automobiles are the most involved in accidents, tailor safety campaigns specifically for automobile drivers. Emphasize safe driving practices and the importance of regular vehicle maintenance.

By focusing on these areas, Montgomery County can continue to improve road safety and reduce traffic violations, injuries, and fatalities.