dataset loading

```
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns

df=pd.read_csv('Crime_Data_from_2020_to_Present.csv')
df
```

```
            DR_NO                Date Rptd                 DATE OCC
TIME OCC  \
0       190326475  03/01/2020 12:00:00 AM  03/01/2020 12:00:00 AM
2130
1       200106753  02/09/2020 12:00:00 AM  02/08/2020 12:00:00 AM
1800
2       200320258  11/11/2020 12:00:00 AM  11/04/2020 12:00:00 AM
1700
3       200907217  05/10/2023 12:00:00 AM  03/10/2020 12:00:00 AM
2037
4       220614831  08/18/2022 12:00:00 AM  08/17/2020 12:00:00 AM
1200
...           ...                      ...                       ...
...
974472  240710284  07/24/2024 12:00:00 AM  07/23/2024 12:00:00 AM
1400
974473  240104953  01/15/2024 12:00:00 AM  01/15/2024 12:00:00 AM
100
974474  241711348  07/19/2024 12:00:00 AM  07/19/2024 12:00:00 AM
757
974475  240309674  04/24/2024 12:00:00 AM  04/24/2024 12:00:00 AM
1500
974476  240910892  08/13/2024 12:00:00 AM  08/12/2024 12:00:00 AM
2300

        AREA   AREA NAME  Rpt Dist No  Part 1-2  Crm Cd  \
0          7    Wilshire          784         1     510
1          1     Central          182         1     330
2          3   Southwest          356         1     480
3          9    Van Nuys          964         1     343
4          6   Hollywood          666         2     354
...      ...         ...          ...       ...     ...
974472     7    Wilshire          788         1     510
974473     1     Central          101         2     745
974474    17  Devonshire         1751         2     888
974475     3   Southwest          358         1     230
974476     9    Van Nuys          914         1     510

                                      Crm Cd Desc  ... Status  \
0                                VEHICLE - STOLEN  ...     AA
```

```
1                                BURGLARY FROM VEHICLE  ...     IC
2                                          BIKE - STOLEN  ...     IC
3            SHOPLIFTING-GRAND THEFT ($950.01 & OVER)  ...     IC
4                                       THEFT OF IDENTITY  ...     IC
...                                                       ...  ...    ...
974472                                   VEHICLE - STOLEN  ...     IC
974473       VANDALISM - MISDEAMEANOR ($399 OR UNDER)  ...     IC
974474                                        TRESPASSING  ...     IC
974475  ASSAULT WITH DEADLY WEAPON, AGGRAVATED ASSAULT  ...     IC
974476                                   VEHICLE - STOLEN  ...     IC

         Status Desc Crm Cd 1 Crm Cd 2  Crm Cd 3 Crm Cd 4  \
0       Adult Arrest    510.0    998.0       NaN      NaN
1        Invest Cont    330.0    998.0       NaN      NaN
2        Invest Cont    480.0      NaN       NaN      NaN
3        Invest Cont    343.0      NaN       NaN      NaN
4        Invest Cont    354.0      NaN       NaN      NaN
...              ...      ...      ...       ...      ...
974472   Invest Cont    510.0      NaN       NaN      NaN
974473   Invest Cont    745.0      NaN       NaN      NaN
974474   Invest Cont    888.0      NaN       NaN      NaN
974475   Invest Cont    230.0      NaN       NaN      NaN
974476   Invest Cont    510.0      NaN       NaN      NaN

                                          LOCATION  \
0        1900 S  LONGWOOD                     AV
1        1000 S  FLOWER                       ST
2        1400 W  37TH                         ST
3       14000    RIVERSIDE                    DR
4                             1900    TRANSIENT
...                                        ...
974472   4000 W  23RD                         ST
974473   1300 W  SUNSET                       BL
974474  10000    OLD DEPOT PLAZA              RD
974475           FLOWER                       ST
974476   6900    VESPER                       AV

                           Cross Street      LAT       LON
0                                   NaN  34.0375 -118.3506
1                                   NaN  34.0444 -118.2628
2                                   NaN  34.0210 -118.3002
3                                   NaN  34.1576 -118.4387
4                                   NaN  34.0944 -118.3277
...                                 ...      ...       ...
974472                              NaN  34.0362 -118.3284
974473                              NaN  34.0685 -118.2460
974474                              NaN  34.2500 -118.5990
974475  JEFFERSON                    BL  34.0215 -118.2868
974476                              NaN  34.1961 -118.4510
```

```
[974477 rows x 28 columns]
```

```
df.describe()
```

```
              DR_NO       TIME OCC           AREA    Rpt Dist No  \
count  9.744770e+05  974477.000000  974477.000000  974477.000000
mean   2.195654e+08    1338.771108      10.705048    1116.939579
std    1.285558e+07     651.717033       6.106287     610.729787
min    8.170000e+02       1.000000       1.000000     101.000000
25%    2.106058e+08     900.000000       5.000000     589.000000
50%    2.208087e+08    1420.000000      11.000000    1141.000000
75%    2.309064e+08    1900.000000      16.000000    1617.000000
max    2.499253e+08    2359.000000      21.000000    2199.000000

            Part 1-2         Crm Cd       Vict Age      Premis Cd  \
count  974477.000000  974477.000000  974477.000000  974463.000000
mean        1.405166     500.748719      29.168748     306.098905
std         0.490924     206.374691      21.954094     218.651133
min         1.000000     110.000000      -4.000000     101.000000
25%         1.000000     331.000000       0.000000     101.000000
50%         1.000000     442.000000      30.000000     203.000000
75%         2.000000     626.000000      44.000000     501.000000
max         2.000000     956.000000     120.000000     976.000000

       Weapon Used Cd       Crm Cd 1      Crm Cd 2     Crm Cd 3   Crm
Cd 4  \
count   325782.000000  974466.000000  68760.000000  2309.000000
64.00000
mean       363.797727     500.501656    958.139311   984.192724
991.21875
std        123.664671     206.171549    110.261645    51.506344
27.06985
min        101.000000     110.000000    210.000000   310.000000
821.00000
25%        311.000000     331.000000    998.000000   998.000000
998.00000
50%        400.000000     442.000000    998.000000   998.000000
998.00000
75%        400.000000     626.000000    998.000000   998.000000
998.00000
max        516.000000     956.000000    999.000000   999.000000
999.00000

                 LAT            LON
count  974477.000000  974477.000000
mean       33.995066    -118.079971
std         1.643523       5.696584
min         0.000000    -118.667600
25%        34.014600    -118.430600
```

```
50%        34.058900    -118.322500
75%        34.164900    -118.273900
max        34.334300       0.000000
```

```
df.isnull().sum()
```

```
DR_NO                  0
Date Rptd              0
DATE OCC               0
TIME OCC               0
AREA                   0
AREA NAME              0
Rpt Dist No            0
Part 1-2               0
Crm Cd                 0
Crm Cd Desc            0
Mocodes           142776
Vict Age               0
Vict Sex          136003
Vict Descent      136013
Premis Cd             14
Premis Desc          584
Weapon Used Cd    648695
Weapon Desc       648695
Status                 1
Status Desc            0
Crm Cd 1              11
Crm Cd 2          905717
Crm Cd 3          972168
Crm Cd 4          974413
LOCATION               0
Cross Street      823461
LAT                    0
LON                    0
dtype: int64
```

```
df.dtypes
```

```
DR_NO              int64
Date Rptd         object
DATE OCC          object
TIME OCC           int64
AREA               int64
AREA NAME         object
Rpt Dist No        int64
Part 1-2           int64
Crm Cd             int64
Crm Cd Desc       object
Mocodes           object
Vict Age           int64
```

```
Vict Sex           object
Vict Descent       object
Premis Cd         float64
Premis Desc        object
Weapon Used Cd    float64
Weapon Desc        object
Status             object
Status Desc        object
Crm Cd 1          float64
Crm Cd 2          float64
Crm Cd 3          float64
Crm Cd 4          float64
LOCATION           object
Cross Street       object
LAT               float64
LON               float64
dtype: object
```

Data Cleaning

```
df['Mocodes']=df['Mocodes'].fillna(0)

df['Vict Sex'] = df['Vict Sex'].fillna('NA')

df['Vict Descent']=df['Vict Descent'].fillna('Not Available')

df['Premis Cd']=df['Premis Cd'].fillna(0)

df['Premis Desc']=df['Premis Desc'].fillna('Not Available')

df['Weapon Used Cd'] = df['Weapon Used Cd'].fillna(0)

df['Weapon Desc'] = df['Weapon Desc'].fillna('Not Available')

df.drop(['Crm Cd 1', 'Crm Cd 2','Crm Cd 3','Crm Cd 4'], axis=1,
inplace=True)

df['Cross Street'] = df['Cross Street'].fillna('Not Available')

df.isnull().sum()
```

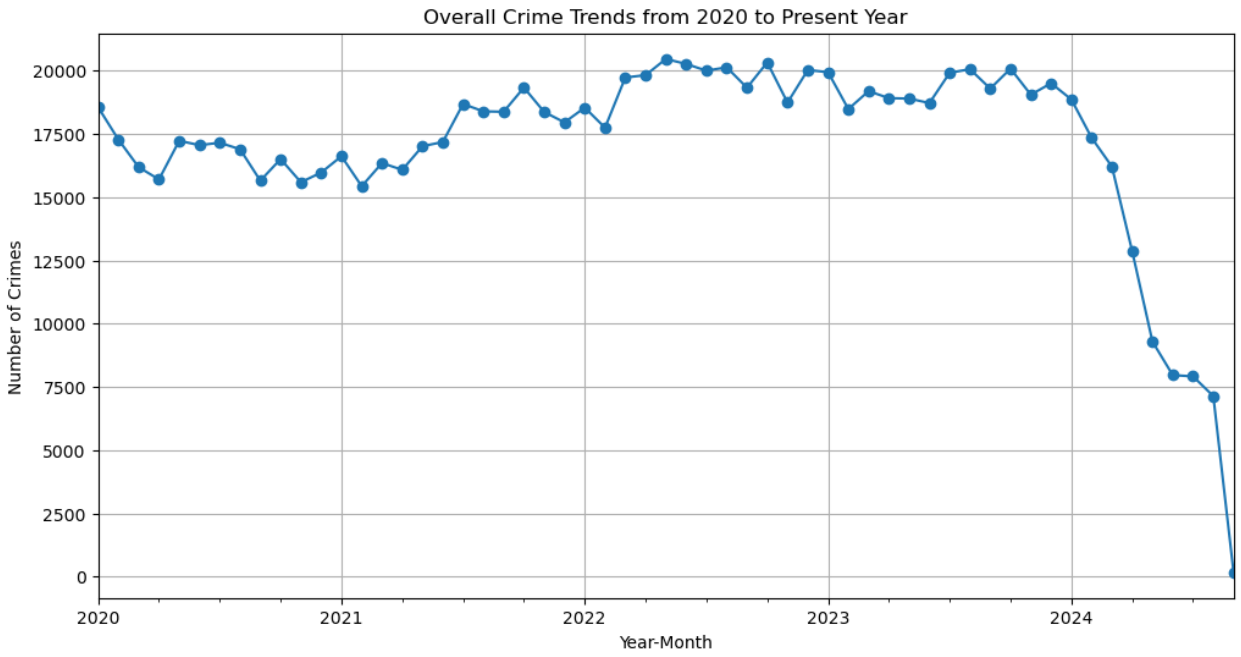
```
DR_NO             0
Date Rptd         0
DATE OCC          0
TIME OCC          0
AREA              0
AREA NAME         0
Rpt Dist No       0
Part 1-2          0
Crm Cd            0
Crm Cd Desc       0
```

```
Mocodes            0
Vict Age           0
Vict Sex           0
Vict Descent       0
Premis Cd          0
Premis Desc        0
Weapon Used Cd     0
Weapon Desc        0
Status             1
Status Desc        0
LOCATION           0
Cross Street       0
LAT                0
LON                0
dtype: int64
```

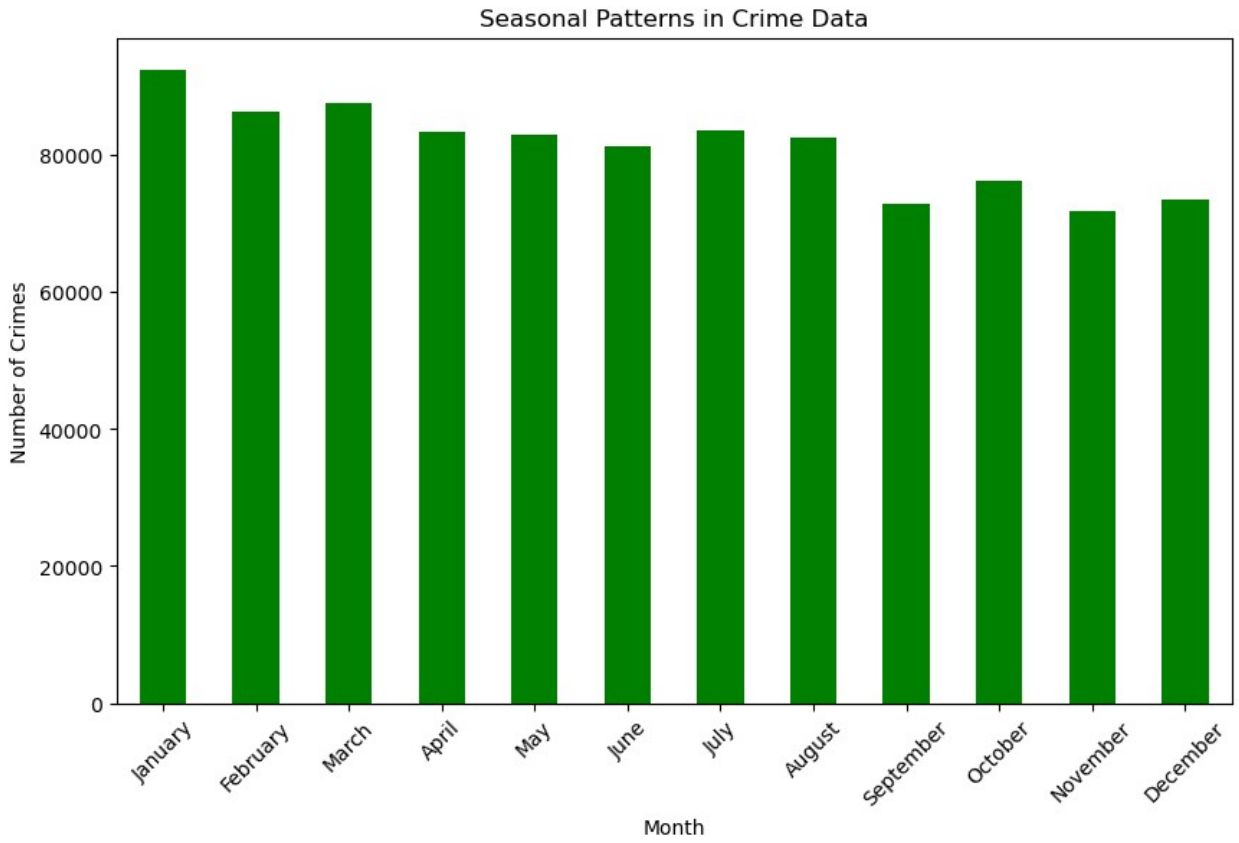
Exploratory Data Analysis

```
#1
date_columns = ['Date Rptd', 'DATE OCC']
for col in date_columns:
    df[col] = pd.to_datetime(df[col])

df['Year-Month'] = df['DATE OCC'].dt.to_period('M')

crime_counts = df.groupby('Year-Month').size()

plt.figure(figsize=(12, 6))
crime_counts.plot(kind='line', marker='o')
plt.title('Overall Crime Trends from 2020 to Present Year')
plt.xlabel('Year-Month')
plt.ylabel('Number of Crimes')
plt.grid(True)
plt.show()
```

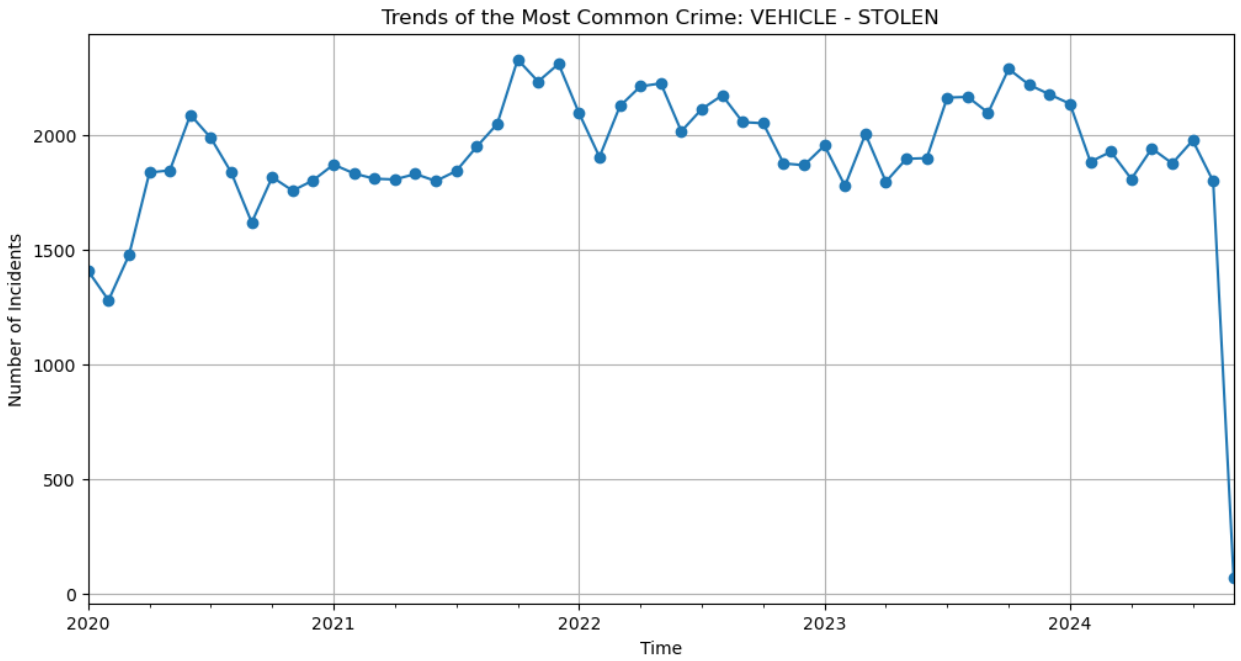
```
/var/folders/cb/2vhpyx_s6_g2z5hhvqfx7tjh0000gn/T/
ipykernel_52015/4036914501.py:4: UserWarning: Could not infer format,
so each element will be parsed individually, falling back to
`dateutil`. To ensure parsing is consistent and as-expected, please
specify a format.
  df[col] = pd.to_datetime(df[col])
/var/folders/cb/2vhpyx_s6_g2z5hhvqfx7tjh0000gn/T/ipykernel_52015/40369
14501.py:4: UserWarning: Could not infer format, so each element will
be parsed individually, falling back to `dateutil`. To ensure parsing
is consistent and as-expected, please specify a format.
  df[col] = pd.to_datetime(df[col])
```



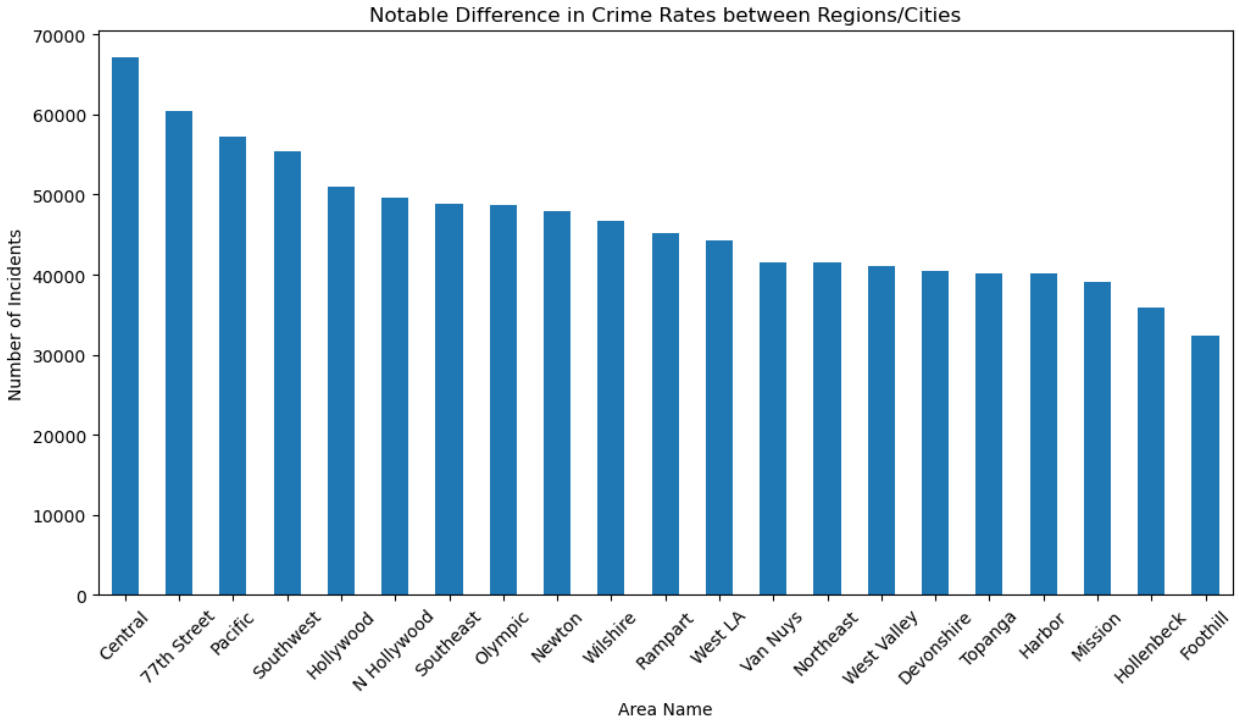

```
#2
date_columns = ['Date Rptd', 'DATE OCC']
for col in date_columns:
    df[col] = pd.to_datetime(df[col])

df['Month'] = df['DATE OCC'].dt.month

monthly_crime_counts = df.groupby('Month').size()

plt.figure(figsize=(10, 6))
monthly_crime_counts.plot(kind='bar', color = 'green')
plt.title('Seasonal Patterns in Crime Data')
plt.xlabel('Month')
plt.ylabel('Number of Crimes')
plt.xticks(range(0, 12), ['January', 'February', 'March', 'April',
'May', 'June', 'July', 'August', 'September', 'October', 'November',
'December'], rotation=45)
plt.show()
```




```
#3
df['DATE OCC'] = pd.to_datetime(df['DATE OCC'])

most_common_crime = df['Crm Cd Desc'].mode().values[0]

crime_data_most_common = df[df['Crm Cd Desc'] == most_common_crime]
crime_data_most_common =
crime_data_most_common.groupby(crime_data_most_common['DATE
OCC'].dt.to_period('M')).size()

plt.figure(figsize=(12, 6))
crime_data_most_common.plot(kind='line', marker='o')
plt.title(f'Trends of the Most Common Crime: {most_common_crime}')
plt.xlabel('Time')
plt.ylabel('Number of Incidents')
plt.grid(True)
plt.show()
```

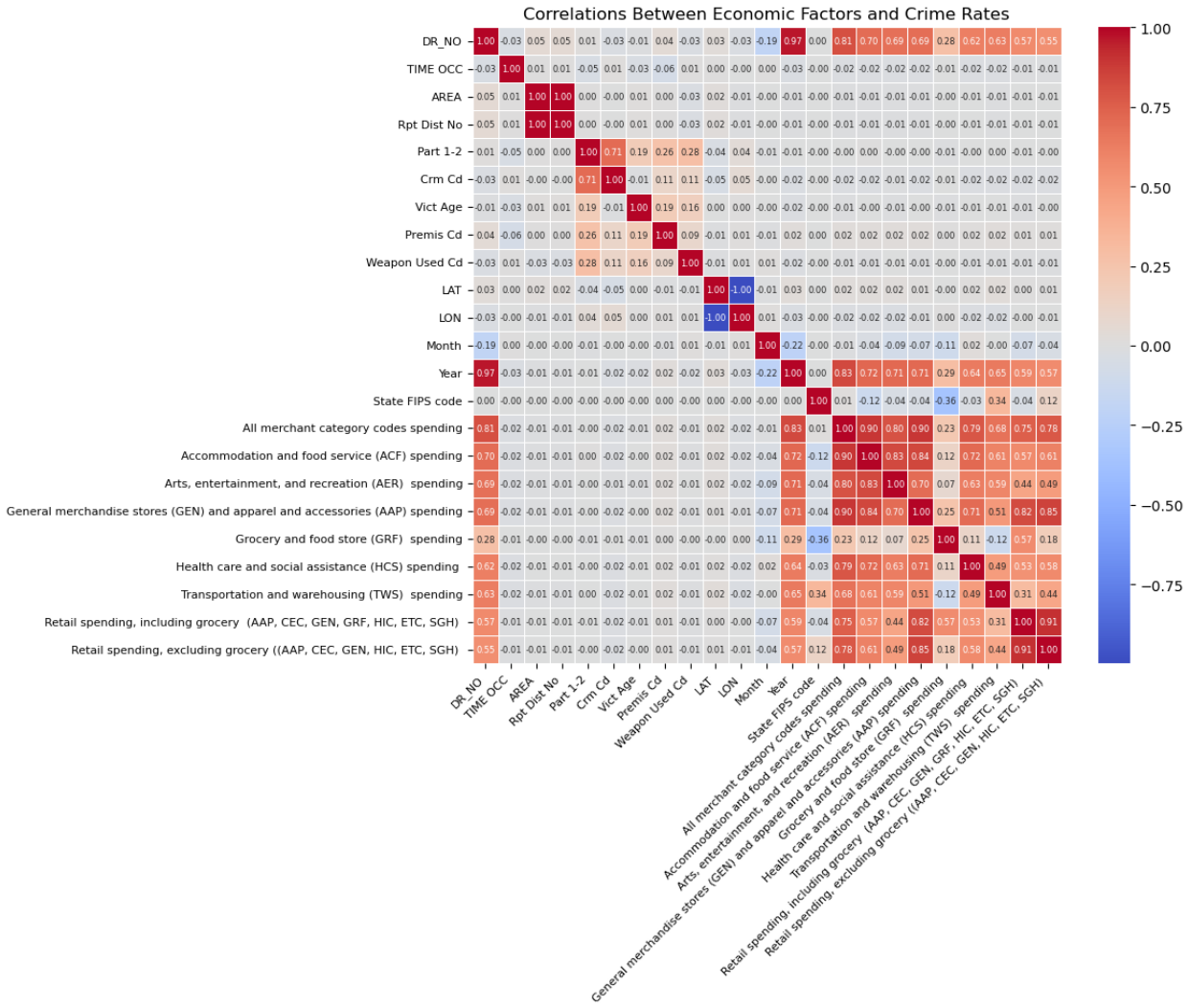



```
#4
crime_by_area = df['AREA NAME'].value_counts()

plt.figure(figsize=(12, 6))
crime_by_area.plot(kind='bar')
plt.title('Notable Difference in Crime Rates between Regions/Cities')
plt.xlabel('Area Name')
plt.ylabel('Number of Incidents')
plt.xticks(rotation=45)
plt.show()
```




```
#5
data = pd.read_csv('Percent_Change_in_Consumer_Spending.csv')

economic_data = pd.DataFrame(data)

economic_data.columns

Index(['State FIPS code', 'Date', 'All merchant category codes
spending',
       'Accommodation and food service (ACF) spending',
       'Arts, entertainment, and recreation (AER)  spending',
       'General merchandise stores (GEN) and apparel and accessories
(AAP) spending',
       'Grocery and food store (GRF)  spending',
       'Health care and social assistance (HCS) spending ',
       'Transportation and warehousing (TWS)  spending',
       'Retail spending, including grocery  (AAP, CEC, GEN, GRF, HIC,
ETC, SGH) ',
       'Retail spending, excluding grocery ((AAP, CEC, GEN, HIC, ETC,
SGH) '],
      dtype='object')


# Ensure both 'DATE OCC' and 'Date' columns are in datetime format
df['DATE OCC'] = pd.to_datetime(df['DATE OCC'])
economic_data['Date'] = pd.to_datetime(economic_data['Date'])

# Now perform the merge
```

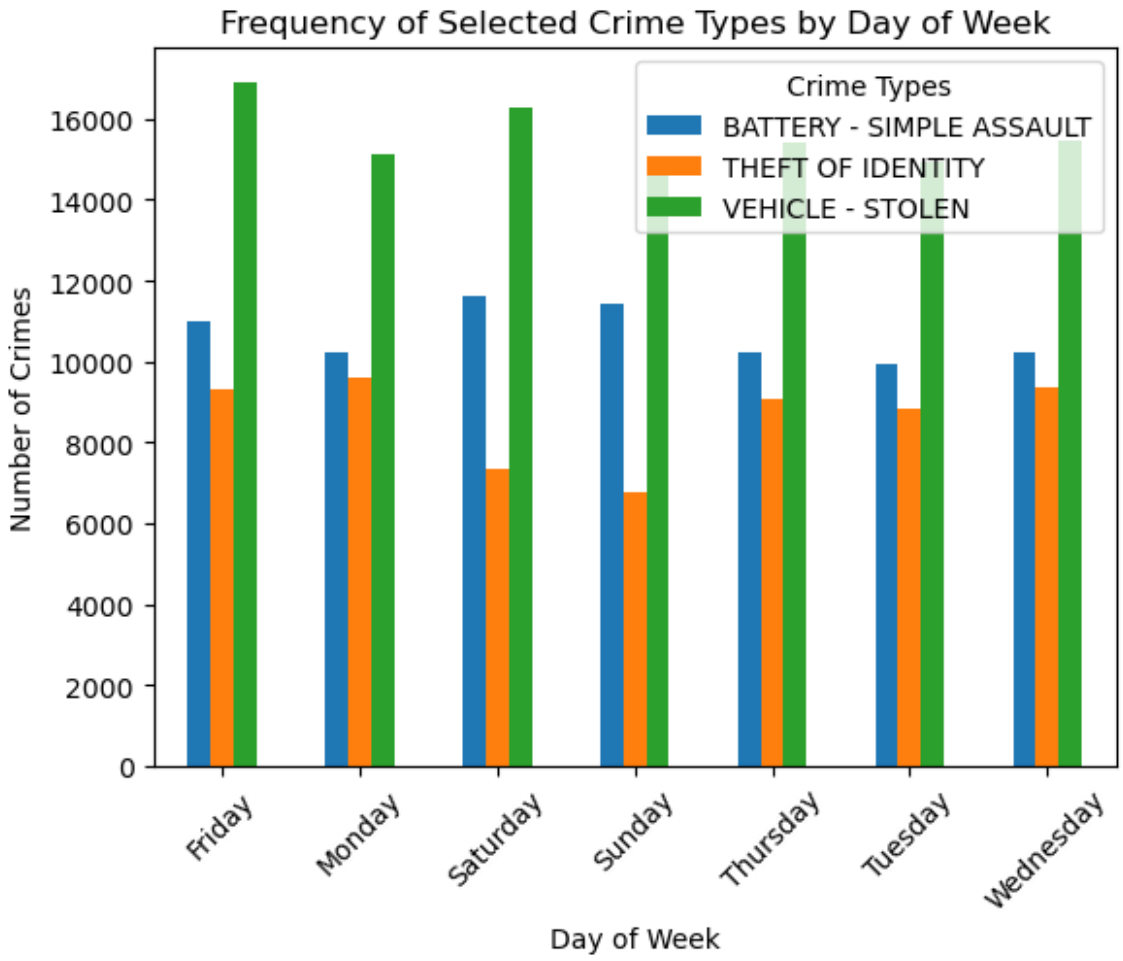
```
merged_data = pd.merge(df, economic_data, left_on='DATE OCC',
right_on='Date')

# Calculate the correlation matrix
correlation_matrix = merged_data.corr(numeric_only=True)
plt.figure(figsize=(12, 10))

# Plot the heatmap
sns.heatmap(correlation_matrix,
            annot=True,
            cmap='coolwarm',
            fmt=".2f",
            linewidths=.5,
            annot_kws={"size": 6})  # Adjust annotation font size

# Rotate x and y axis labels to prevent overlap
plt.xticks(rotation=45, ha='right', fontsize=8)  # Rotate and adjust
x-axis label font size
plt.yticks(rotation=0, fontsize=8)  # Adjust y-axis label font size

# Set title and layout
plt.title('Correlations Between Economic Factors and Crime Rates',
fontsize=12)
plt.tight_layout()
plt.show()
```




```
#6
df['Day of Week'] = df['DATE OCC'].dt.day_name()

selected_crime_types = ['THEFT OF IDENTITY', 'BATTERY - SIMPLE
ASSAULT', 'VEHICLE - STOLEN']

selected_crime_data = df[df['Crm Cd Desc'].isin(selected_crime_types)]

day_of_week_crime_patterns = selected_crime_data.groupby(['Day of
Week', 'Crm Cd Desc']).size().unstack()

plt.figure(figsize=(18, 6))
day_of_week_crime_patterns.plot(kind='bar')
plt.title('Frequency of Selected Crime Types by Day of Week')
plt.xlabel('Day of Week')
plt.ylabel('Number of Crimes')
plt.xticks(rotation=45)
```

```
plt.legend(title='Crime Types', loc='upper right')
plt.show()
```

```
<Figure size 1800x600 with 0 Axes>
```

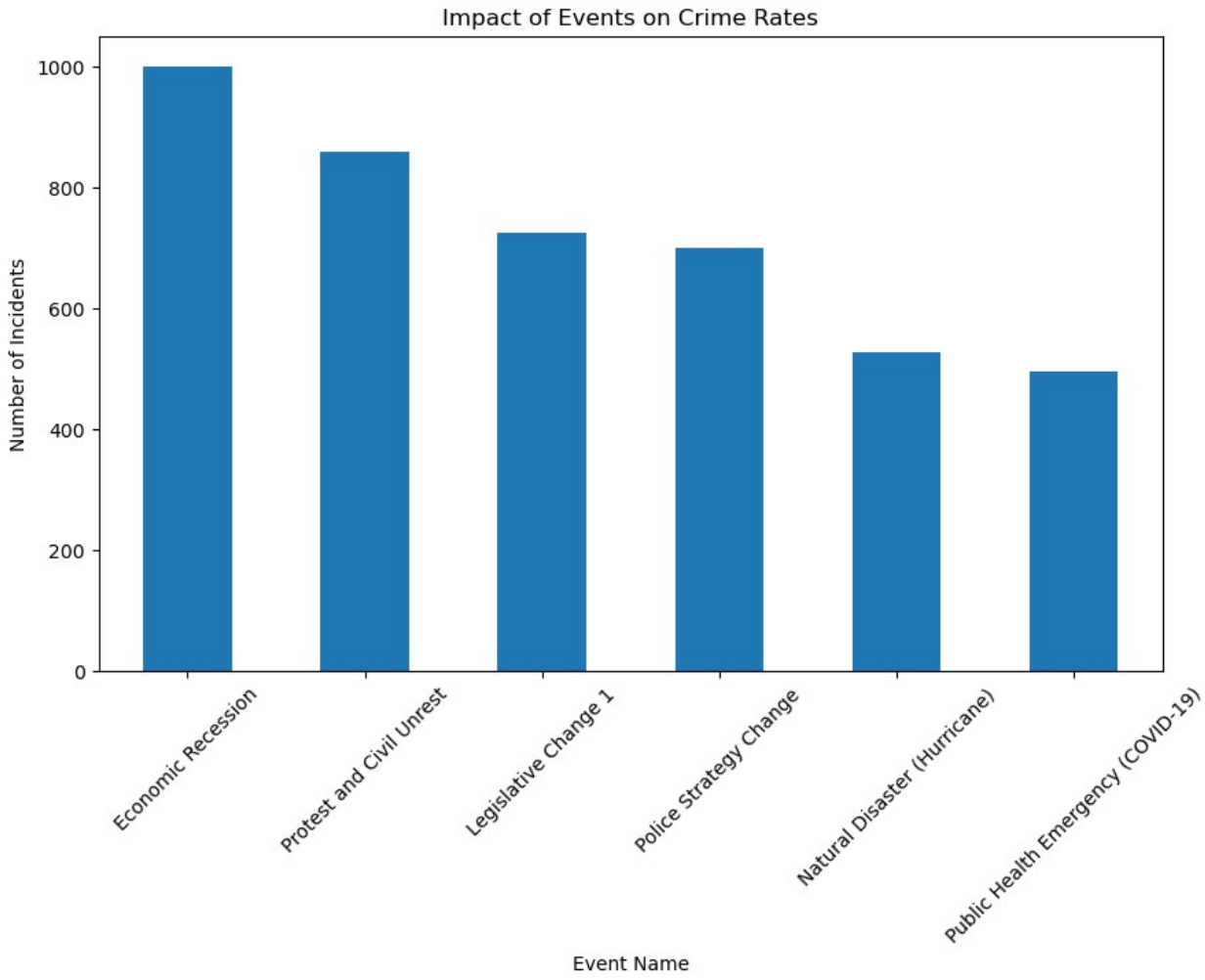



```
#7
#contextual data
data = {
    'Event Name': [
        'Legislative Change 1',
        'Economic Recession',
        'Protest and Civil Unrest',
        'Police Strategy Change',
        'Natural Disaster (Hurricane)',
        'Public Health Emergency (COVID-19)'
    ],
    'Start Date': [
        '2024-01-01',
        '2022-01-01',
        '2020-06-01',
```

```
        '2023-09-15',
        '2020-08-24',
        '2021-03-15'
    ],
    'End Date': [
        '2024-03-31',
        '2023-02-26',
        '2020-06-07',
        '2023-10-30',
        '2020-08-31',
        '2022-12-31'
    ]
}

contextual_data = pd.DataFrame(data)

contextual_data
```

	Event Name	Start Date	End Date
0	Legislative Change 1	2024-01-01	2024-03-31
1	Economic Recession	2022-01-01	2023-02-26
2	Protest and Civil Unrest	2020-06-01	2020-06-07
3	Police Strategy Change	2023-09-15	2023-10-30
4	Natural Disaster (Hurricane)	2020-08-24	2020-08-31
5	Public Health Emergency (COVID-19)	2021-03-15	2022-12-31

```
df['DATE OCC'] = pd.to_datetime(df['DATE OCC'])
contextual_data['Start Date'] = pd.to_datetime(contextual_data['Start
Date'])
contextual_data['End Date'] = pd.to_datetime(contextual_data['End
Date'])

merged_data = pd.merge(df, contextual_data, how='left', left_on='DATE
OCC', right_on='Start Date')

crime_data_within_event = merged_data[(merged_data['DATE OCC'] >=
merged_data['Start Date']) & (merged_data['DATE OCC'] <=
merged_data['End Date'])]

crime_counts = crime_data_within_event['Event Name'].value_counts()

plt.figure(figsize=(10, 6))
crime_counts.plot(kind='bar')
plt.title('Impact of Events on Crime Rates')
plt.xlabel('Event Name')
plt.ylabel('Number of Incidents')
plt.xticks(rotation=45)
plt.show()
```

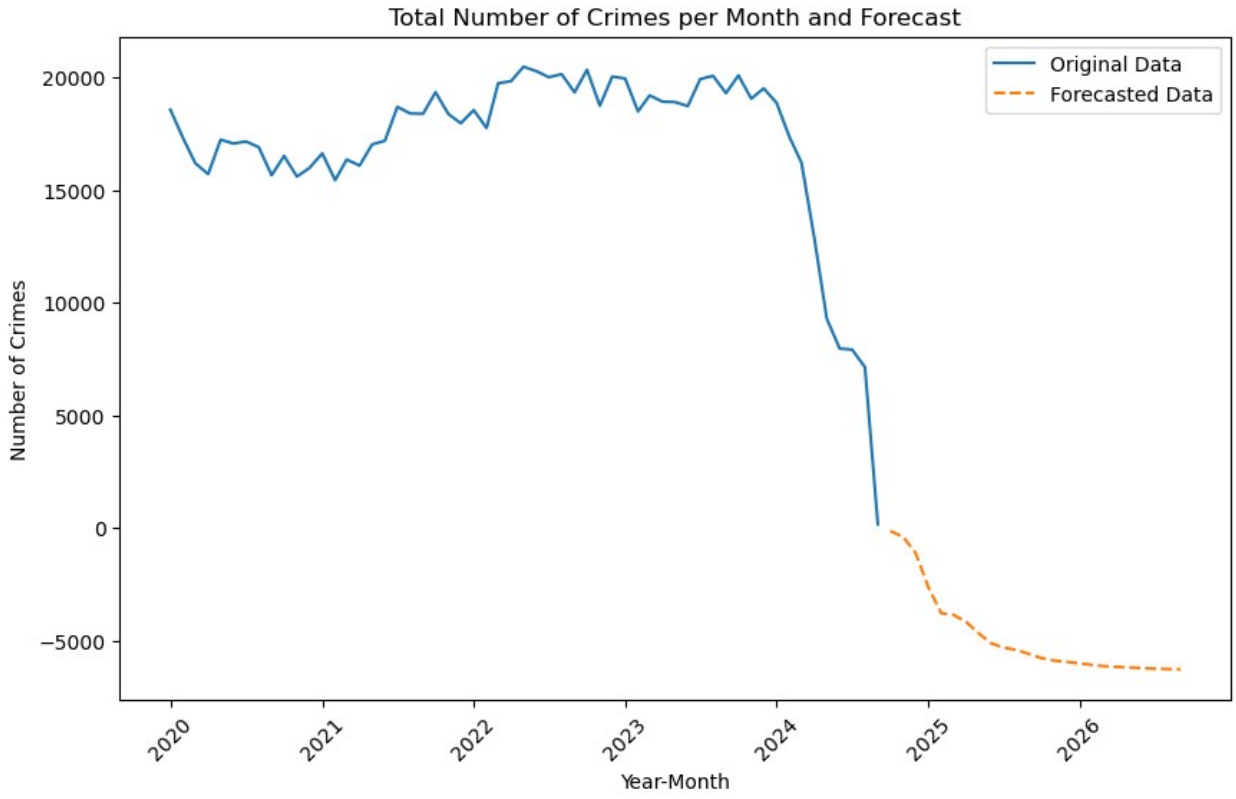


Advanced Analysis FUTURE PREDICTION

```
from statsmodels.tsa.arima.model import ARIMA

date_columns = ['Date Rptd', 'DATE OCC']
for col in date_columns:
    df[col] = pd.to_datetime(df[col])

df['Year-Month'] = df['DATE OCC'].dt.to_period('M')
total_crimes_per_month = df['Year-Month'].value_counts().sort_index()

total_crimes_per_month.index =
total_crimes_per_month.index.to_timestamp()

model = ARIMA(total_crimes_per_month, order=(5,1,0))
model_fit = model.fit()

forecast = model_fit.forecast(steps=24)

plt.figure(figsize=(10, 6))
```

```
plt.plot(total_crimes_per_month, label='Original Data')
plt.plot(forecast, label='Forecasted Data', linestyle='--')
plt.title('Total Number of Crimes per Month and Forecast')
plt.xlabel('Year-Month')
plt.ylabel('Number of Crimes')
plt.legend()
plt.xticks(rotation=45)
plt.show()
```

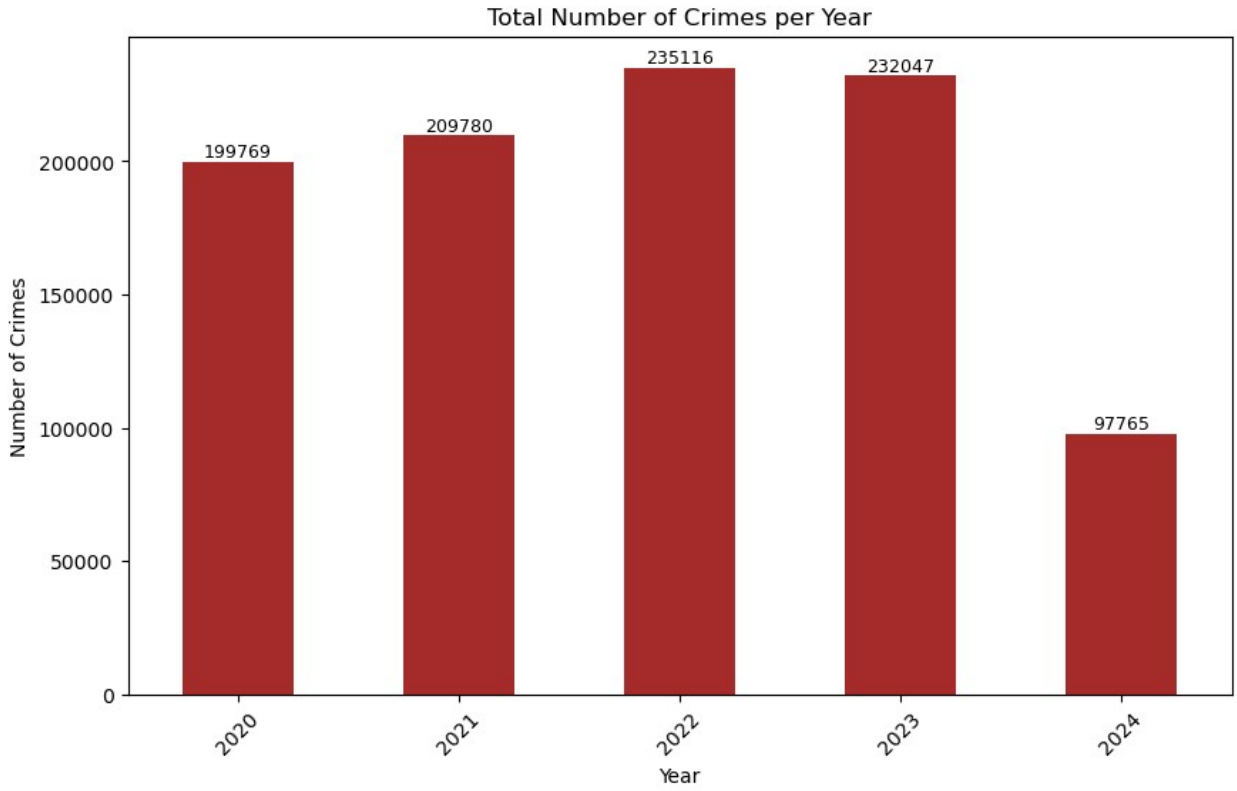
```
/opt/anaconda3/lib/python3.12/site-packages/statsmodels/tsa/
statespace/sarimax.py:966: UserWarning: Non-stationary starting
autoregressive parameters found. Using zeros as starting parameters.
  warn('Non-stationary starting autoregressive parameters'
```



Overall Crime Trends

```
date_columns = ['Date Rptd', 'DATE OCC']
for col in date_columns:
    df[col] = pd.to_datetime(df[col])

df['Year'] = df['DATE OCC'].dt.year
total_crimes_per_year = df['Year'].value_counts().sort_index()

plt.figure(figsize=(10, 6))
bars = total_crimes_per_year.plot(kind='bar', color='brown')
```

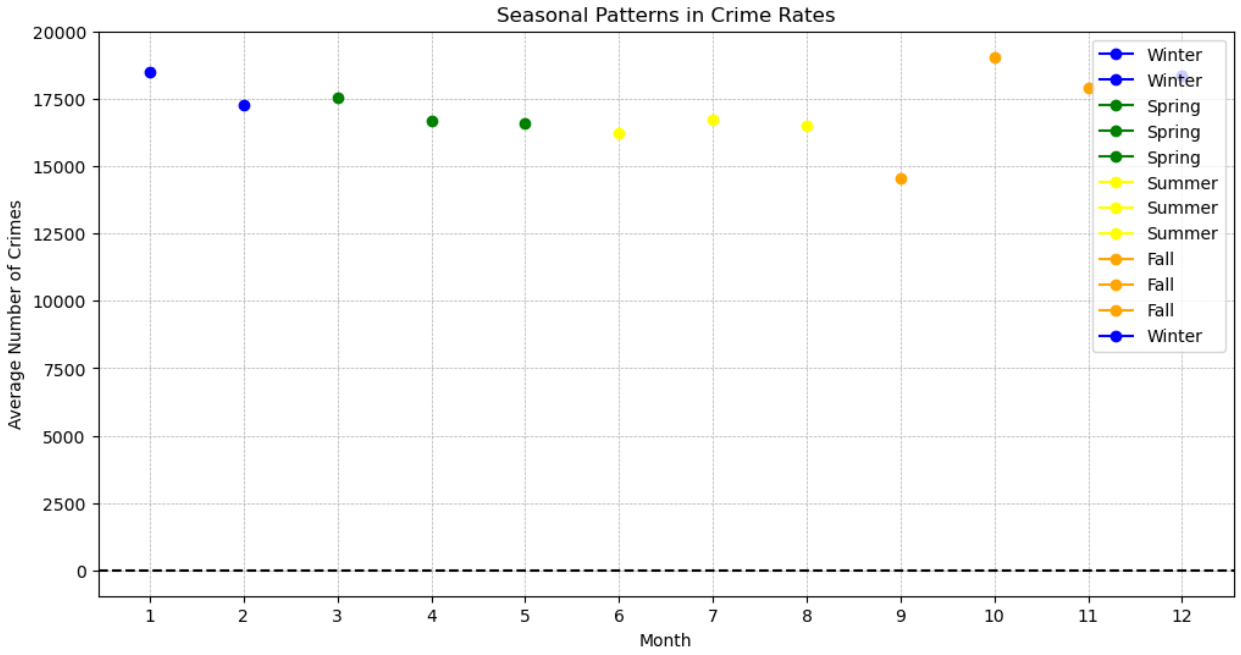
```
plt.title('Total Number of Crimes per Year')
plt.xlabel('Year')
plt.ylabel('Number of Crimes')
plt.xticks(rotation=45)

for bar in bars.patches:
    yval = bar.get_height()
    plt.annotate(f'{int(yval)}', (bar.get_x() + bar.get_width() / 2,
yval),
                 ha='center', va='bottom', fontsize=9)

plt.show()
```



Seasonal Patterns:

```
df['DATE OCC'] = pd.to_datetime(df['DATE OCC'])

df['Year'] = df['DATE OCC'].dt.year
df['Month'] = df['DATE OCC'].dt.month

monthly_avg_crime_counts = df.groupby(['Year',
'Month']).size().groupby('Month').mean()
season_colors = {
    'Winter': 'blue',
    'Spring': 'green',
```

```
    'Summer': 'yellow',
    'Fall': 'orange'
}

months = monthly_avg_crime_counts.index.map(str)
values = monthly_avg_crime_counts.values

plt.figure(figsize=(12, 6))

for i, month in enumerate(months):
    season = ''
    if month in ['12', '1', '2']:
        season = 'Winter'
    elif month in ['3', '4', '5']:
        season = 'Spring'
    elif month in ['6', '7', '8']:
        season = 'Summer'
    else:
        season = 'Fall'

    plt.plot(month, values[i], marker='o',
color=season_colors[season], label=season if
months[:i+1].tolist().count(month) == 1 else "")

plt.title('Seasonal Patterns in Crime Rates')
plt.xlabel('Month')
plt.ylabel('Average Number of Crimes')
plt.grid(True, which='both', linestyle='--', linewidth=0.5)
plt.axhline(y=0, color='black', linestyle='--')
plt.legend(loc='upper right')
plt.show()
```



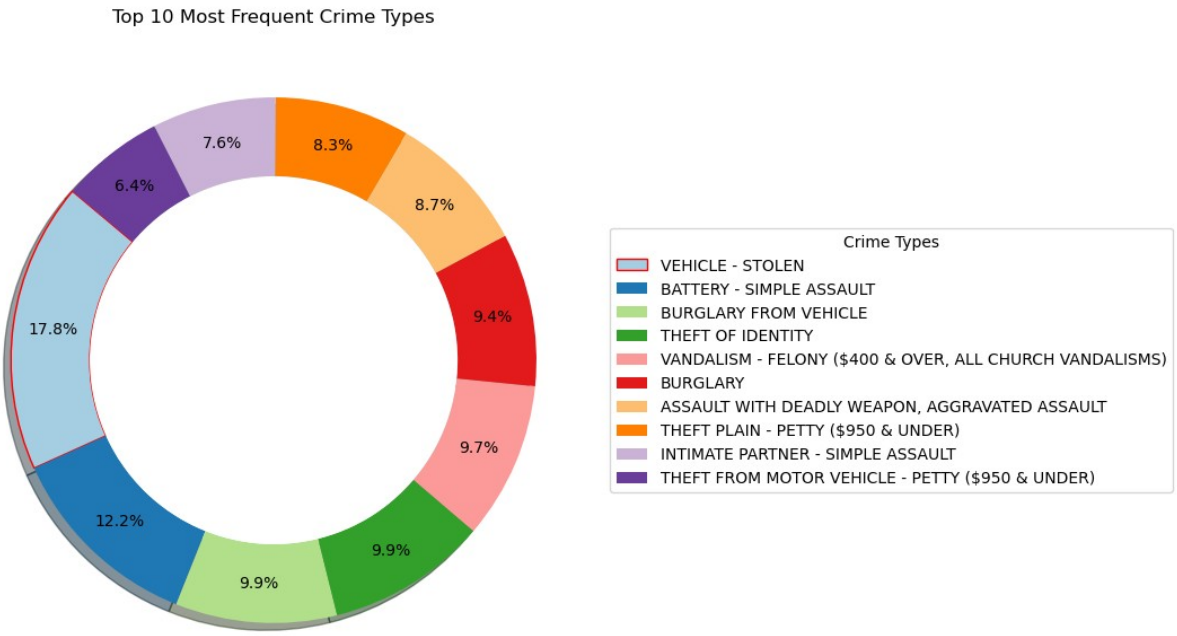
Most Common Crime Type:

```
crime_counts = df['Crm Cd Desc'].value_counts()
top_10_crimes = crime_counts.head(10)
most_common_crime_type = df['Crm Cd Desc'].value_counts().idxmax()
print(f"The most common crime type is: {most_common_crime_type}")
```

```
The most common crime type is: VEHICLE - STOLEN
```

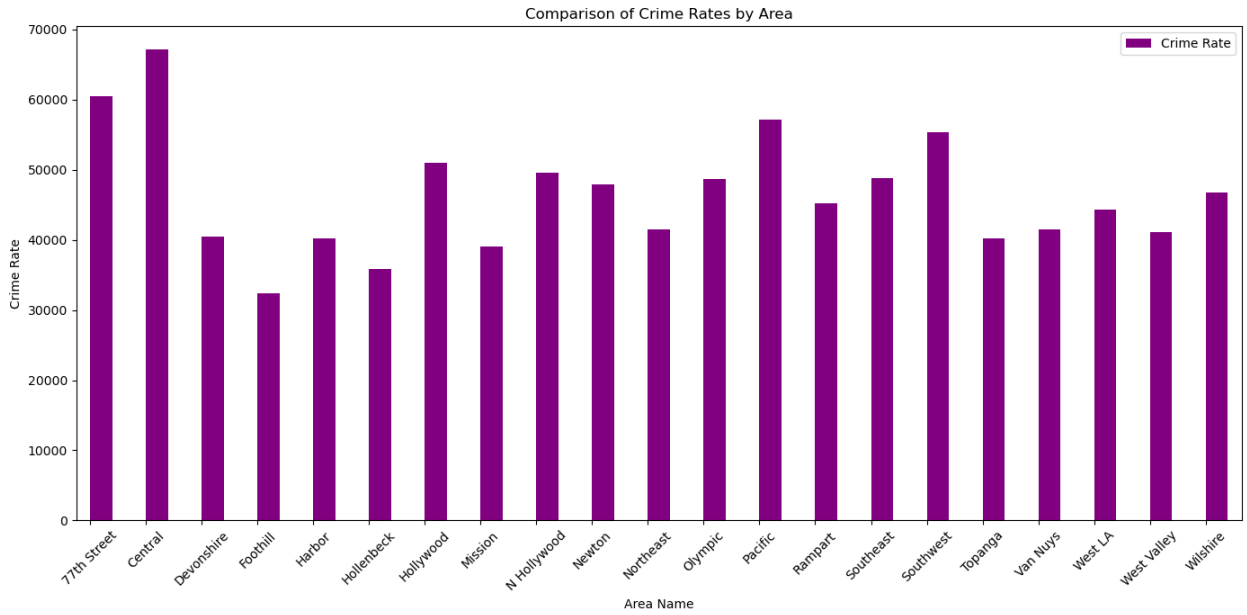
```
colors = plt.cm.Paired(range(len(top_10_crimes)))

plt.figure(figsize=(10, 7))
wedges, texts, autotexts = plt.pie(top_10_crimes, colors=colors,
labels=top_10_crimes.index,
                                   autopct='%1.1f%%', shadow=True,
startangle=140,
                                   wedgeprops=dict(width=0.3),
pctdistance=0.85)

for t in texts:
    t.set_visible(False)

wedges[top_10_crimes.index.get_loc(most_common_crime_type)].set_edgeco
lor('red')

centre_circle = plt.Circle((0,0),0.70,fc='white')
fig = plt.gcf()
fig.gca().add_artist(centre_circle)

plt.legend(wedges, top_10_crimes.index,
```

```
          title="Crime Types",
          loc="center left",
          bbox_to_anchor=(1, 0, 0.5, 1))

plt.title('Top 10 Most Frequent Crime Types')
plt.tight_layout()

plt.show()
```



Regional Differences:

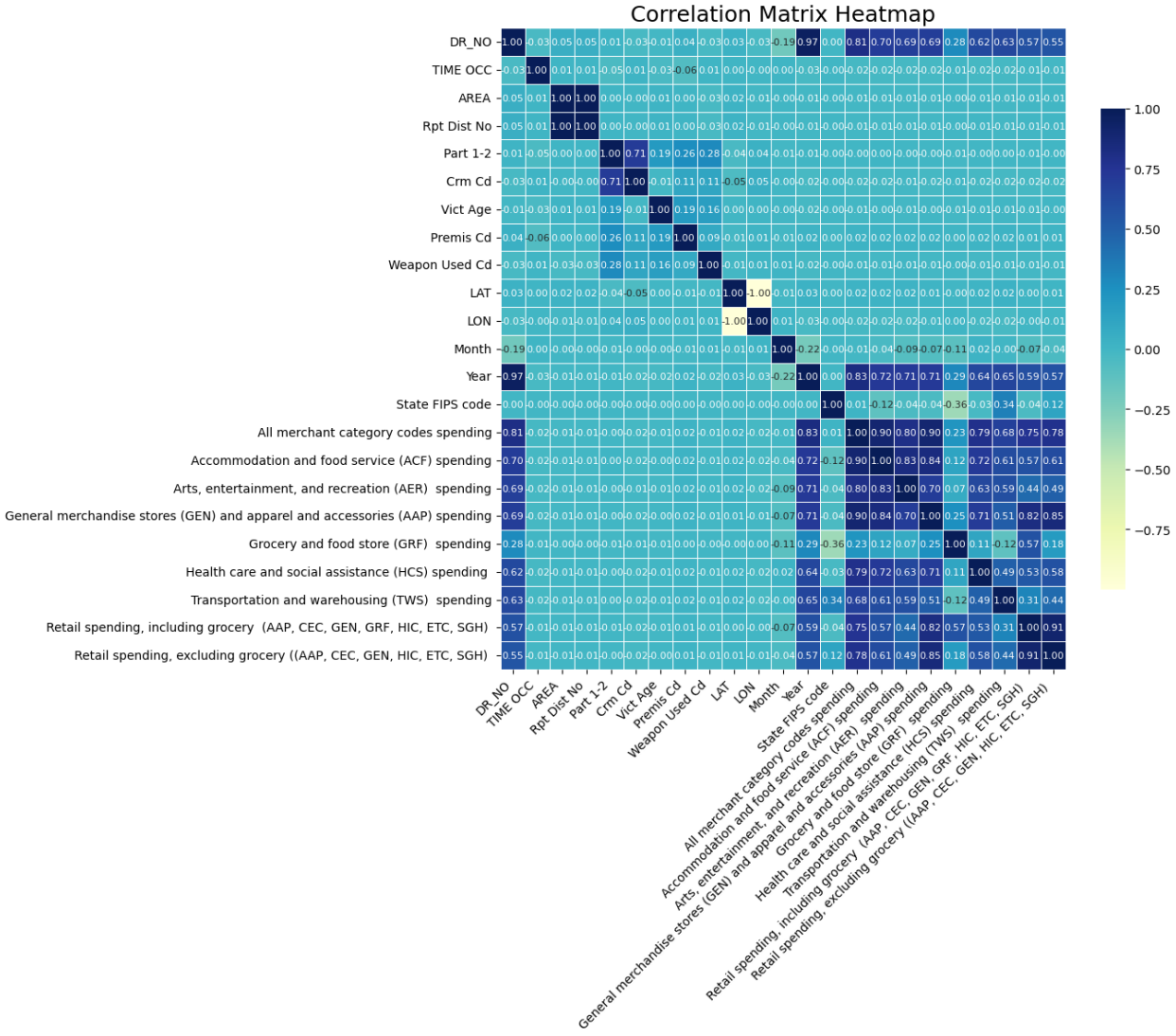
```
grouped_data = df.groupby('AREA NAME')

crime_stats_by_area = grouped_data['Crm Cd'].describe()

plt.figure(figsize=(14, 7))

crime_stats_by_area['count'].plot(kind='bar', color='purple',
position=0, width=0.4, label='Crime Rate')

plt.title('Comparison of Crime Rates by Area')
plt.xlabel('Area Name')
plt.ylabel('Crime Rate')
plt.legend()
plt.xticks(rotation=45)
plt.tight_layout()
plt.show()
```



Correlation with Economic Factors:

```
# If your column names are long, you could rename them to shorter
names for visualization
shortened_column_names = {
    'Long Column Name 1': 'Short 1',
    'Long Column Name 2': 'Short 2',
    # Add more mappings here
}

# Optionally rename the columns for visualization purposes (remove if
not needed)
correlation_matrix =
correlation_matrix.rename(columns=shortened_column_names,
index=shortened_column_names)

# Adjust figure size
plt.figure(figsize=(14, 12))

# Plot the heatmap with better label management
sns.heatmap(correlation_matrix,
            annot=True,
            cmap='YlGnBu',
            center=0,
            linewidths=.5,
            fmt=".2f",
            annot_kws={"size": 8},  # Smaller annotation size for
better readability
            cbar_kws={"shrink": 0.75})  # Adjust color bar size

# Adjust title and label size
```

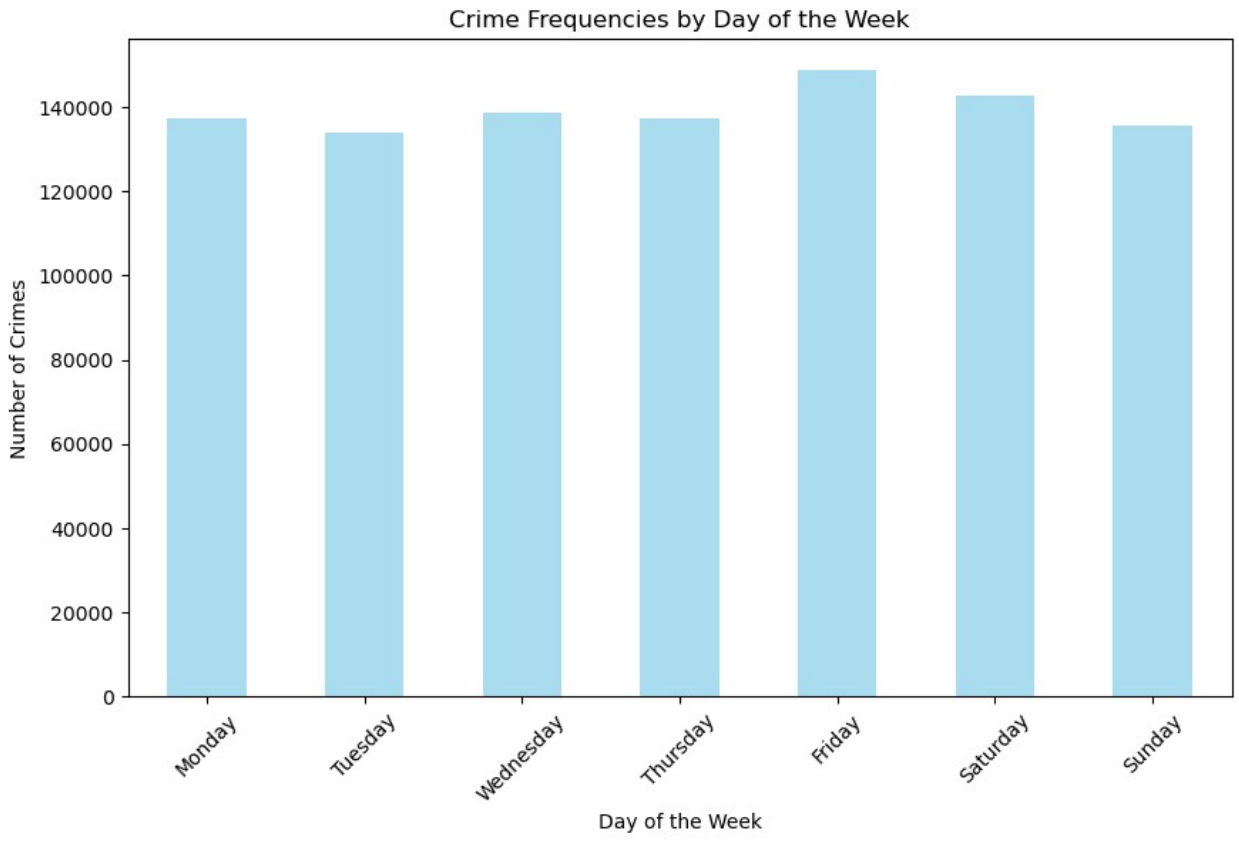
```
plt.title('Correlation Matrix Heatmap', fontsize=18)
plt.xticks(rotation=45, ha='right', fontsize=10)  # Rotate and align x-axis labels diagonally
plt.yticks(rotation=0, fontsize=10)  # Keep y-axis labels horizontal

# Ensure everything fits in the figure
plt.tight_layout()

# Show the heatmap
plt.show()
```



Day of the Week Analysis:

```
df['DATE OCC'] = pd.to_datetime(df['DATE OCC'])

df['Day of Week'] = df['DATE OCC'].dt.day_name()
```

```
crime_counts_by_day = df['Day of Week'].value_counts()

plt.figure(figsize=(10, 6))
crime_counts_by_day = crime_counts_by_day.reindex(['Monday',
'Tuesday', 'Wednesday', 'Thursday', 'Friday', 'Saturday', 'Sunday'])
crime_counts_by_day.plot(kind='bar', color='skyblue', alpha=0.7)
plt.title('Crime Frequencies by Day of the Week')
plt.xlabel('Day of the Week')
plt.ylabel('Number of Crimes')
plt.xticks(rotation=45)
plt.show()
```



Impact of Major Events:

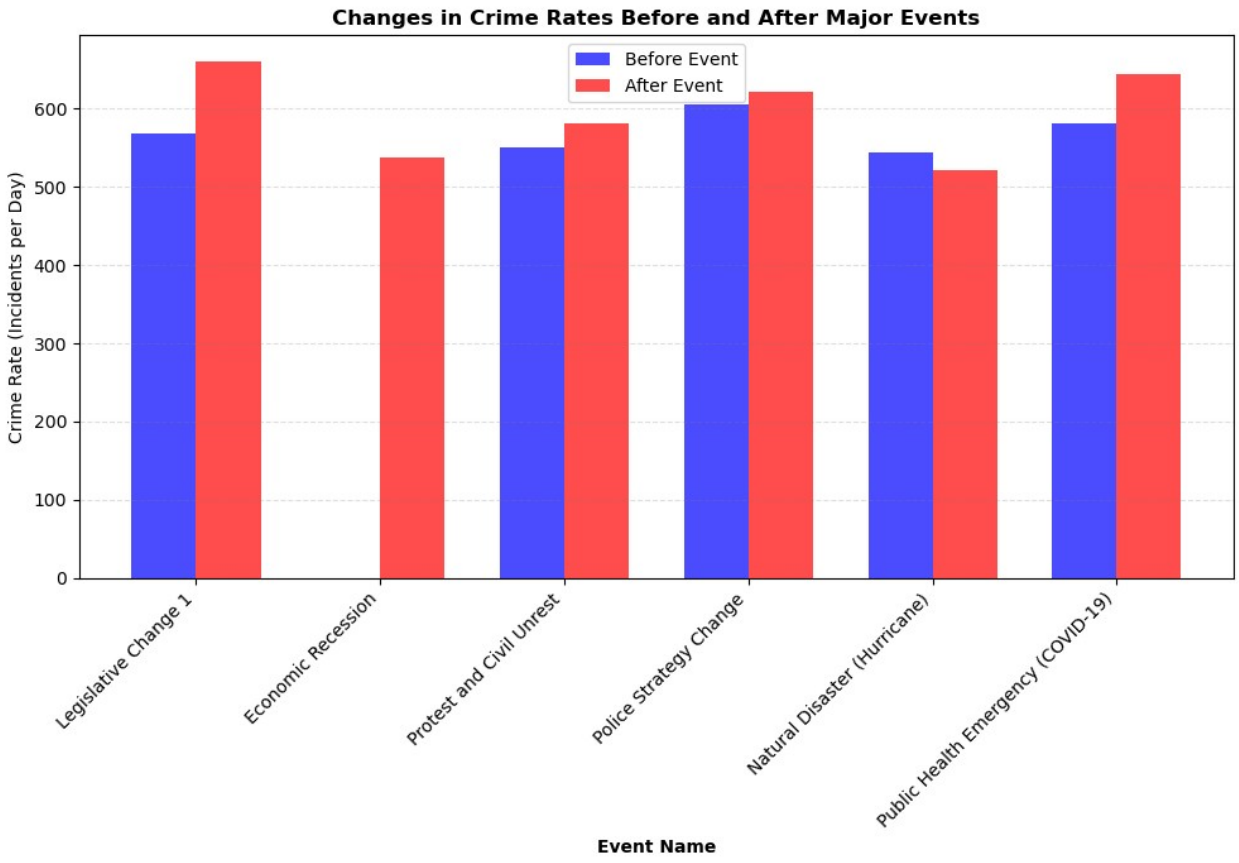
```
contextual_data['Start Date'] = pd.to_datetime(contextual_data['Start
Date'])
contextual_data['End Date'] = pd.to_datetime(contextual_data['End
Date'])

results = pd.DataFrame(columns=['Event Name', 'Crime Rate Before',
'Crime Rate After'])

for index, row in contextual_data.iterrows():
```

```
    event_name = row['Event Name']
    start_date = row['Start Date']
    end_date = row['End Date']

    crimes_before = df[(df['DATE OCC'] >= (start_date -
pd.DateOffset(days=30))) & (df['DATE OCC'] < start_date)]

    crimes_after = df[(df['DATE OCC'] > end_date) & (df['DATE OCC'] <=
(end_date + pd.DateOffset(days=30)))]

    crime_rate_before = round(len(crimes_before) / 30)
    crime_rate_after = round(len(crimes_after) / 30)

    new_row = {'Event Name': event_name, 'Crime Rate Before':
crime_rate_before, 'Crime Rate After': crime_rate_after}
    results = pd.concat([results, pd.DataFrame([new_row])],
ignore_index=True)
results
```

```
                          Event Name Crime Rate Before Crime Rate
After
0                 Legislative Change 1                569
661
1                   Economic Recession                  0
537
2            Protest and Civil Unrest                551
581
3              Police Strategy Change                605
622
4        Natural Disaster (Hurricane)                544
522
5  Public Health Emergency (COVID-19)                582
645
```

```
events = results['Event Name'].tolist()
crime_rate_before = results['Crime Rate Before'].tolist()
crime_rate_after = results['Crime Rate After'].tolist()

fig, ax = plt.subplots(figsize=(10, 7))

bar_width = 0.35
index = range(len(events))

before_bars = ax.bar(index, crime_rate_before, bar_width,
label='Before Event', color='blue', alpha=0.7)
after_bars = ax.bar([i + bar_width for i in index], crime_rate_after,
bar_width, label='After Event', color='red', alpha=0.7)

ax.set_title('Changes in Crime Rates Before and After Major Events',
fontweight='bold')
```

```
ax.set_xlabel('Event Name', fontweight='bold')
ax.set_ylabel('Crime Rate (Incidents per Day)')
ax.set_xticks([i + bar_width / 2 for i in index])
ax.set_xticklabels(events, rotation=45, ha='right')
ax.legend()

ax.yaxis.grid(True, linestyle='--', which='major', color='grey',
alpha=0.25)

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

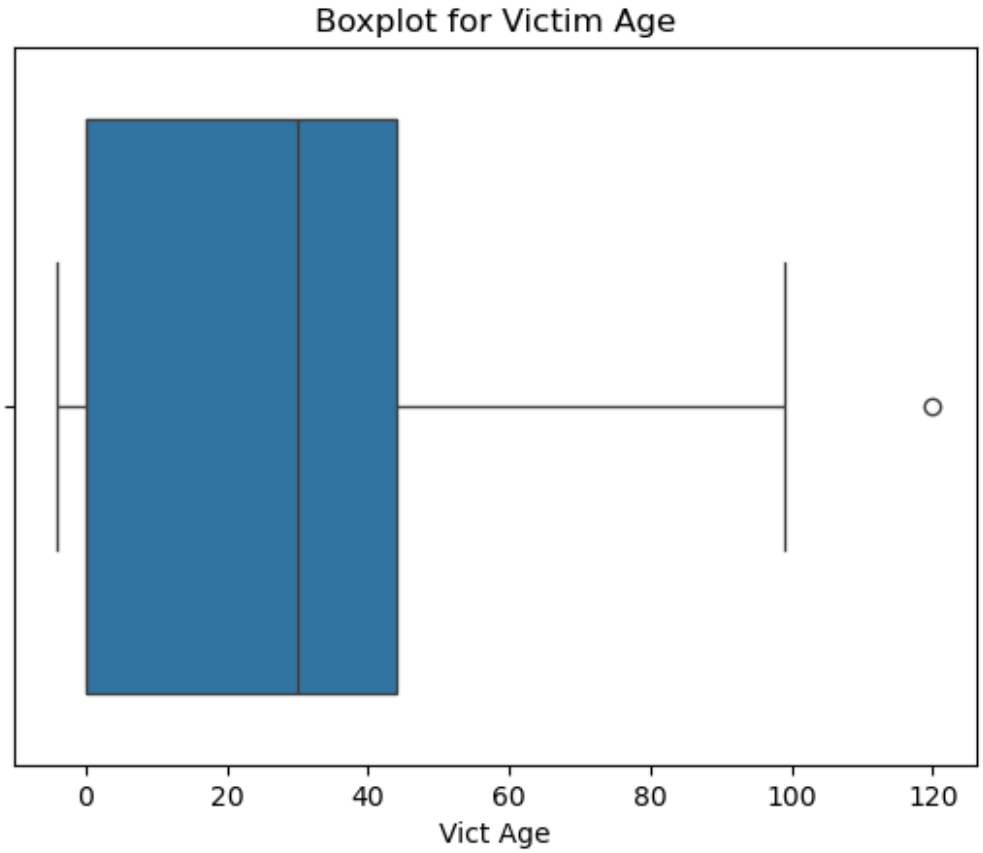


Outliers and Anomalies

```
from scipy.stats import zscore

z_scores = zscore(df['Vict Age'])

outliers = (z_scores > 3) | (z_scores < -3)

print(df['Vict Age'][outliers])
```

```
2587      99
2652      99
2719      96
4472      99
17628     99
          ..
962481    99
966998    99
967291    99
968207    99
974414    97
Name: Vict Age, Length: 591, dtype: int64
```

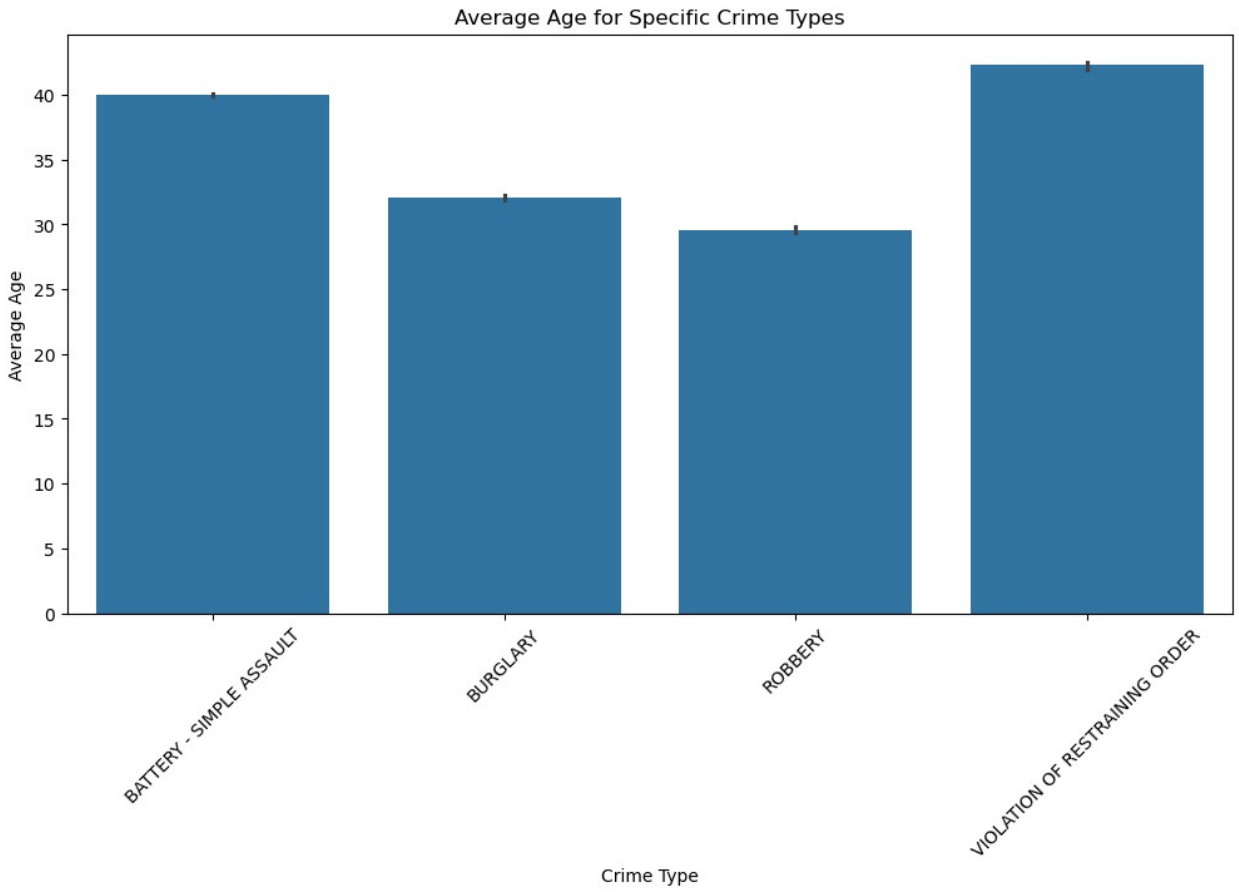
```
sns.boxplot(x=df['Vict Age'])
plt.title('Boxplot for Victim Age')
plt.show()
```

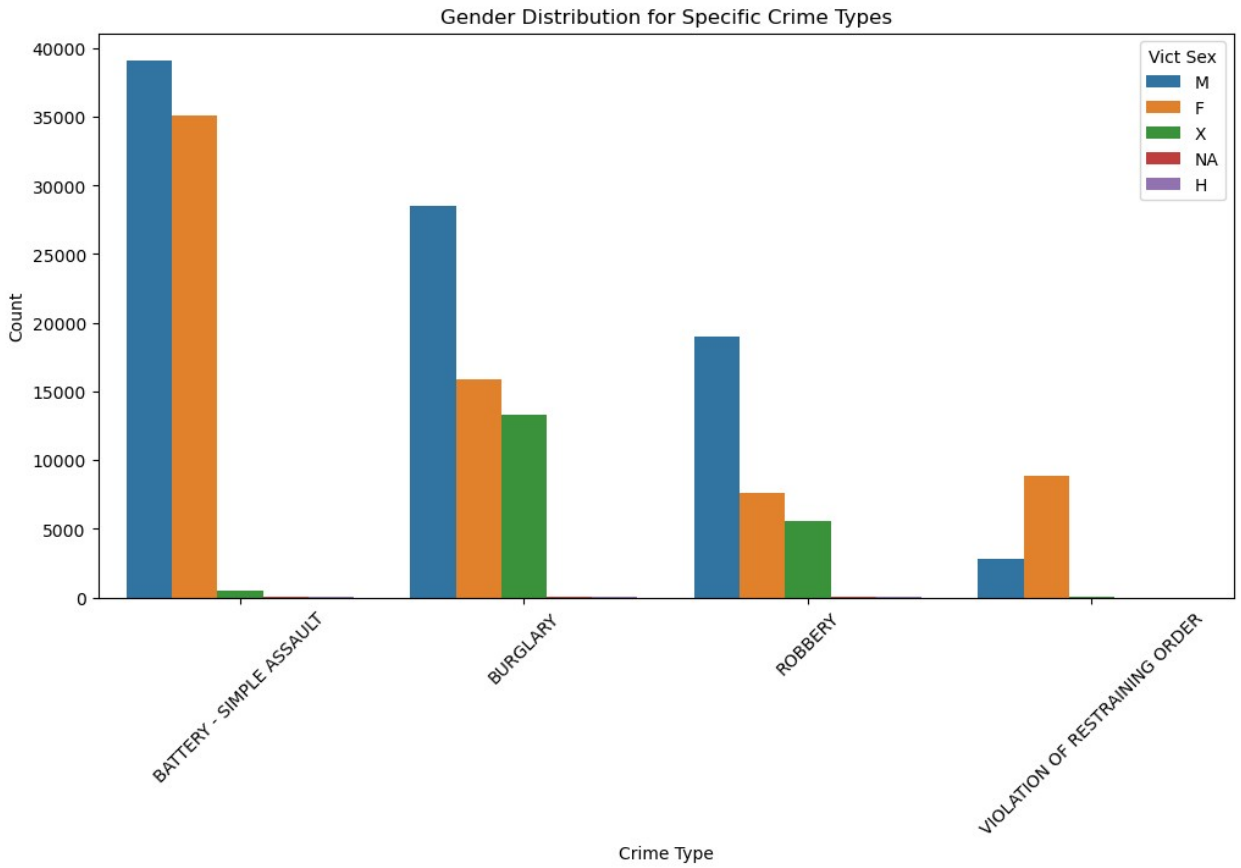


Demographic Factors

```
specific_crime_types = ['BATTERY - SIMPLE ASSAULT', 'ROBBERY',
'BURGLARY', 'VIOLATION OF RESTRAINING ORDER']

filtered_data = df[df['Crm Cd Desc'].isin(specific_crime_types)]
```

```
plt.figure(figsize=(12, 6))
sns.barplot(data=filtered_data, x='Crm Cd Desc', y='Vict Age')
plt.title('Average Age for Specific Crime Types')
plt.xlabel('Crime Type')
plt.ylabel('Average Age')
plt.xticks(rotation=45)
plt.show()

plt.figure(figsize=(12, 6))
sns.countplot(data=filtered_data, x='Crm Cd Desc', hue='Vict Sex')
plt.title('Gender Distribution for Specific Crime Types')
plt.xlabel('Crime Type')
plt.ylabel('Count')
plt.xticks(rotation=45)
plt.legend(title='Vict Sex')
plt.show()
```





Predicting Future Trends:

```
from statsmodels.tsa.arima.model import ARIMA

date_columns = ['Date Rptd', 'DATE OCC']
for col in date_columns:
    df[col] = pd.to_datetime(df[col])

df['Year-Month'] = df['DATE OCC'].dt.to_period('M')

total_crimes_per_month = df['Year-Month'].value_counts().sort_index()

total_crimes_per_month.index =
total_crimes_per_month.index.to_timestamp()

# ARIMA model
model = ARIMA(total_crimes_per_month, order=(5,1,0))
model_fit = model.fit()

# Forecast future crime trends
forecast = model_fit.forecast(steps=24)

plt.figure(figsize=(10, 6))
plt.plot(total_crimes_per_month, label='Original Data')
```

```
plt.plot(forecast, label='Forecasted Data', linestyle='--')
plt.title('Total Number of Crimes per Month and Forecast')
plt.xlabel('Year-Month')
plt.ylabel('Number of Crimes')
plt.legend()
plt.xticks(rotation=45)
plt.show()
```

```
/opt/anaconda3/lib/python3.12/site-packages/statsmodels/tsa/
statespace/sarimax.py:966: UserWarning: Non-stationary starting
autoregressive parameters found. Using zeros as starting parameters.
  warn('Non-stationary starting autoregressive parameters'
```