# Project : Crime Data Analysis for the City of Los Angeles

## Objective of the Project

In this project, I worked with a real-world dataset containing crime data from 2020 to the present. My goal was to clean and prepare the dataset for analysis, perform exploratory data analysis, and analyse crime trends, patterns, and factors influencing crime rates.

## Dataset:

I have used the crime dataset available at https://catalog.data.gov/dataset/crime-data-from-2020-to-present

In [1]:

```
# Importing necessary libraries
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
%matplotlib inline
from datetime import datetime as dt
import calendar
import seaborn as sns
```

# 1. Data Acquisition

In [2]:

```
# Loading Crime Dataset
crime_data = pd.read_csv('Crime_Data_from_2020_to_Present 2.csv')
```

```
/var/folders/wj/lvd1hjrx0vqf2hzhdxhvbz580000gn/T/ipykernel_53142/1536514335.py:2: DtypeWarning: Columns (7) have
mixed types. Specify dtype option on import or set low_memory=False.
  crime_data = pd.read_csv('Crime_Data_from_2020_to_Present 2.csv')
```

In [3]:

```
# Checking the shape of the dataframe
crime_data.shape
```

Out[3]:

```
(807377, 28)
```

## 2. Data Inspection:

In [4]:

```
# Displaying first 5 rows of the dataframe
crime_data.head()
```

Out[4]:

| | DR_NO | Date Rptd | DATE OCC | TIME OCC | AREA | AREA NAME | Rpt Dist No | Part 1-2 | Crm Cd | Crm Cd Desc | ... | Status | Status Desc | Crm Cd 1 | Crm Cd 2 | Crm Cd 3 | Crm Cd 4 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **0** | 10304468 | 1/8/20 0:00 | 1/8/20 0:00 | 2230 | 3 | Southwest | 377 | 2 | 624 | BATTERY - SIMPLE ASSAULT | ... | AO | Adult Other | 624.0 | NaN | NaN | NaN |
| **1** | 190101086 | 1/2/20 0:00 | 1/1/20 0:00 | 330 | 1 | Central | 163 | 2 | 624 | BATTERY - SIMPLE ASSAULT | ... | IC | Invest Cont | 624.0 | NaN | NaN | NaN |
| **2** | 200110444 | 4/14/20 0:00 | 2/13/20 0:00 | 1200 | 1 | Central | 155 | 2 | 845 | SEX OFFENDER REGISTRANT OUT OF COMPLIANCE | ... | AA | Adult Arrest | 845.0 | NaN | NaN | NaN |
| **3** | 191501505 | 1/1/20 0:00 | 1/1/20 0:00 | 1730 | 15 | N Hollywood | 1543 | 2 | 745 | VANDALISM - MISDEAMEANOR ($399 OR UNDER) | ... | IC | Invest Cont | 745.0 | 998.0 | NaN | NaN |
| **4** | 191921269 | 1/1/20 0:00 | 1/1/20 0:00 | 415 | 19 | Mission | 1998 | 2 | 740 | VANDALISM - FELONY ($400 & OVER, ALL CHURCH VA... | ... | IC | Invest Cont | 740.0 | NaN | NaN | NaN |

5 rows × 28 columns

In [5]:

```
# Creating Copy of the Dataframe
data_crime = crime_data.copy()
```

In [6]:

```
# Checking datatypes of the Dataframe
crime_data.dtypes
```

Out[6]:

```
DR_NO               int64
Date Rptd          object
DATE OCC           object
TIME OCC            int64
AREA                int64
AREA NAME          object
Rpt Dist No         int64
Part 1-2           object
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
```

In [7]:

```
# Printing Columns of the Dataframe
crime_data.columns
```

Out[7]:

```
Index(['DR_NO', 'Date Rptd', 'DATE OCC', 'TIME OCC', 'AREA', 'AREA NAME',
       'Rpt Dist No', 'Part 1-2', 'Crm Cd', 'Crm Cd Desc', 'Mocodes',
       'Vict Age', 'Vict Sex', 'Vict Descent', 'Premis Cd', 'Premis Desc',
       'Weapon Used Cd', 'Weapon Desc', 'Status', 'Status Desc', 'Crm Cd 1',
       'Crm Cd 2', 'Crm Cd 3', 'Crm Cd 4', 'LOCATION', 'Cross Street', 'LAT',
       'LON'],
      dtype='object')
```

In [8]:

```
# Information about the Dataframe including the index dtype and columns, non-null values and memory usage
crime_data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 807377 entries, 0 to 807376
Data columns (total 28 columns):
 #   Column          Non-Null Count   Dtype
---  ------          --------------   -----
 0   DR_NO           807377 non-null  int64
 1   Date Rptd       807377 non-null  object
 2   DATE OCC        807377 non-null  object
 3   TIME OCC        807377 non-null  int64
 4   AREA            807377 non-null  int64
 5   AREA NAME       807377 non-null  object
 6   Rpt Dist No     807377 non-null  int64
 7   Part 1-2        807377 non-null  object
 8   Crm Cd          807377 non-null  int64
 9   Crm Cd Desc     807377 non-null  object
 10  Mocodes         696010 non-null  object
 11  Vict Age        807377 non-null  int64
 12  Vict Sex        701468 non-null  object
 13  Vict Descent    701460 non-null  object
 14  Premis Cd       807368 non-null  float64
 15  Premis Desc     806901 non-null  object
 16  Weapon Used Cd  281174 non-null  float64
 17  Weapon Desc     281174 non-null  object
 18  Status          807377 non-null  object
 19  Status Desc     807377 non-null  object
 20  Crm Cd 1        807367 non-null  float64
 21  Crm Cd 2        59483 non-null   float64
 22  Crm Cd 3        1987 non-null    float64
 23  Crm Cd 4        58 non-null      float64
 24  LOCATION        807377 non-null  object
 25  Cross Street    129232 non-null  object
 26  LAT             807377 non-null  float64
 27  LON             807377 non-null  float64
dtypes: float64(8), int64(6), object(14)
memory usage: 172.5+ MB
```

## 3. Data Cleaning:

In [9]:

```
# checking the missing values in crime dataframe
crime_data.isna().sum()
```

Out[9]:

```
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            111367
Vict Age                0
Vict Sex           105909
Vict Descent       105917
Premis Cd               9
Premis Desc           476
Weapon Used Cd     526203
Weapon Desc        526203
Status                  0
Status Desc             0
Crm Cd 1               10
Crm Cd 2           747894
Crm Cd 3           805390
Crm Cd 4           807319
LOCATION                0
Cross Street       678145
LAT                     0
LON                     0
dtype: int64
```

In [10]:

```
# functions to drop/dropna columns
def drop_columns(df,columns_to_drop):
    return df.drop(columns=columns_to_drop,inplace=True)

def dropna_columns(df,columns_to_drop):
    return df.dropna(subset=columns_to_drop,inplace=True)
```

In [11]:

```
# Droping Column
column_to_dropna = ['Crm Cd 1']
dropna_columns(crime_data,column_to_dropna)
```

In [12]:

```
# handling the H and - values by adding them to X
crime_data['Vict Sex'].replace(to_replace=['H','-'],value='X',inplace=True)
crime_data['Vict Sex'].fillna('X',inplace=True)
print(crime_data['Vict Sex'].unique())
```

```
['F' 'M' 'X']
```

In [13]:

```
# Gender count of victims
crime_data['Vict Sex'].value_counts()
```

Out[13]:

```
Vict Sex
M    333678
F    297639
X    176050
Name: count, dtype: int64
```

In [14]:

```
# adding column for - weapon used or not
crime_data['Weapon Used'] = crime_data['Weapon Used Cd'].apply(lambda x: 1 if pd.notna(x) else 0)
```

In [15]:

```
# filling missing values with 'unknown' and '0'
crime_data['Premis Desc'] = crime_data['Premis Desc'].fillna('Unknown')
crime_data['Premis Cd'] = crime_data['Premis Cd'].fillna(0)
```

In [16]:

```
# replaced nan values of 'cross street' with '' and added them to Location column
crime_data['Cross Street'].fillna('',inplace=True)
crime_data['LOCATION'] = crime_data['LOCATION'] + ' ' + crime_data['Cross Street']
```

In [17]:

```
# Checking any Null values in the Dataframe
crime_data.isna().sum().sort_values(ascending=False)
```

Out[17]:

```
Crm Cd 4          807309
Crm Cd 3          805380
Crm Cd 2          747894
Weapon Used Cd    526196
Weapon Desc       526196
Mocodes           111367
Vict Descent      105916
DR_NO                  0
LON                    0
LAT                    0
Cross Street           0
LOCATION               0
Crm Cd 1               0
Status Desc            0
Status                 0
Premis Cd              0
Premis Desc            0
Date Rptd              0
Vict Sex               0
Vict Age               0
Crm Cd Desc            0
Crm Cd                 0
Part 1-2               0
Rpt Dist No            0
AREA NAME              0
AREA                   0
TIME OCC               0
DATE OCC               0
Weapon Used            0
dtype: int64
```

In [18]:

```python
# converting datetime column to correct datatype
crime_data['Date Rptd'] = pd.to_datetime(crime_data['Date Rptd'])
```

```
/var/folders/wj/lvd1hjrx0vqf2hzhdxhvbz580000gn/T/ipykernel_53142/3292367836.py:2: UserWarning: Could not infer f
ormat, so each element will be parsed individually, falling back to `dateutil`. To ensure parsing is consistent
and as-expected, please specify a format.
  crime_data['Date Rptd'] = pd.to_datetime(crime_data['Date Rptd'])
```

In [19]:

```python
# Creating copy of the Dataframe
crime_data1 = crime_data.copy()
```

In [20]:
```
# Checking shape of the Dataframe
crime_data.shape
```

Out[20]:
```
(807367, 29)
```

In [21]:
```
# converting time to hour minute format
crime_data['TIME OCC'] = crime_data['TIME OCC'].apply(lambda x: f'{x//100:02}:{x%100:02}')
```

In [22]:
```
# Combining date and time (hour, minute) of crime occured
crime_data['Date Time OCC'] = pd.to_datetime(crime_data['DATE OCC'].str.split().str[0]+' '+crime_data['TIME OCC']
```

```
/var/folders/wj/lvd1hjrx0vqf2hzhdxhvbz580000gn/T/ipykernel_53142/3137231879.py:2: UserWarning: Could not infer f
ormat, so each element will be parsed individually, falling back to `dateutil`. To ensure parsing is consistent
and as-expected, please specify a format.
  crime_data['Date Time OCC'] = pd.to_datetime(crime_data['DATE OCC'].str.split().str[0]+' '+crime_data['TIME OC
C'].astype(str))
```

In [23]:
```
# Converting the Datatype of the Columns
crime_data['Date Rptd'] = pd.to_datetime(crime_data['Date Rptd'])
crime_data['Date Time OCC'] = pd.to_datetime(crime_data['Date Time OCC'])
```

In [24]:
```
# rearranging columns
temp = crime_data.pop('Date Time OCC')
crime_data.insert(2,'Date Time OCC',temp)
```

In [25]:
```
# count of victim descents and missing values
print(crime_data['Vict Descent'].value_counts())
print('The missing values in Vict Descent column:',crime_data['Vict Descent'].isna().sum())
```

```
Vict Descent
H    247787
W    164842
B    115215
X     77446
O     64019
A     17678
K      4297
F      3360
C      3089
J      1123
V       830
I       765
Z       408
P       216
U       166
D        59
G        55
L        49
S        45
-         2
Name: count, dtype: int64
The missing values in Vict Descent column: 105916
```

In [26]:

```
# filling and replacing missing/unknown values
crime_data['Vict Descent'].fillna('X',inplace=True)
crime_data['Vict Descent'].replace(to_replace='-',value='X',inplace=True)
```

In [27]:

```
# handling semantic errors in data
crime_data = crime_data[crime_data['Vict Age'].between(0, 100)]
```

In [28]:

```
# dropping irrelevant columns (relevant information already extracted from the columns or are irrelevant)
column_to_drop=['Crm Cd 2','Crm Cd 3','Crm Cd 4','AREA','Part 1-2','Cross Street','DATE OCC','TIME OCC',
                'Weapon Used Cd','Weapon Desc','Mocodes']
drop_columns(crime_data,column_to_drop)
```

In [29]:

```
# the cleaned dataset
print('The nan values in dataset:\n',crime_data.isna().sum())
print('\nThe shape of cleaned dataset:\n',crime_data.shape)
```

```
The nan values in dataset:
 DR_NO             0
Date Rptd        0
Date Time OCC    0
AREA NAME        0
Rpt Dist No      0
Crm Cd           0
Crm Cd Desc      0
Vict Age         0
Vict Sex         0
Vict Descent     0
Premis Cd        0
Premis Desc      0
Status           0
Status Desc      0
Crm Cd 1         0
LOCATION         0
LAT              0
LON              0
Weapon Used      0
dtype: int64

The shape of cleaned dataset:
 (807298, 19)
```

In [30]:

```
# Displaying first 5 rows of the Dataframe
crime_data.head()
```

Out[30]:

| | DR_NO | Date Rptd | Date Time OCC | AREA NAME | Rpt Dist No | Crm Cd | Crm Cd Desc | Vict Age | Vict Sex | Vict Descent | Premis Cd | Premis Desc | Status | Status Desc | Crm Cd 1 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **0** | 10304468 | 2020-01-08 | 2020-01-08 22:30:00 | Southwest | 377 | 624 | BATTERY - SIMPLE ASSAULT | 36 | F | B | 501.0 | SINGLE FAMILY DWELLING | AO | Adult Other | 624.0 |
| **1** | 190101086 | 2020-01-02 | 2020-01-01 03:30:00 | Central | 163 | 624 | BATTERY - SIMPLE ASSAULT | 25 | M | H | 102.0 | SIDEWALK | IC | Invest Cont | 624.0 |
| **2** | 200110444 | 2020-04-14 | 2020-02-13 12:00:00 | Central | 155 | 845 | SEX OFFENDER REGISTRANT OUT OF COMPLIANCE | 0 | X | X | 726.0 | POLICE FACILITY | AA | Adult Arrest | 845.0 |
| **3** | 191501505 | 2020-01-01 | 2020-01-01 17:30:00 | N Hollywood | 1543 | 745 | VANDALISM - MISDEAMEANOR ($399 OR UNDER) | 76 | F | W | 502.0 | MULTI-UNIT DWELLING (APARTMENT, DUPLEX, ETC) | IC | Invest Cont | 745.0 |
| **4** | 191921269 | 2020-01-01 | 2020-01-01 04:15:00 | Mission | 1998 | 740 | VANDALISM - FELONY ($400 & OVER, ALL CHURCH VA... | 31 | X | X | 409.0 | BEAUTY SUPPLY STORE | IC | Invest Cont | 740.0 |

In [31]:

```python
# Creating columns for year, month and year_month
crime_data['Year'] = crime_data['Date Time OCC'].dt.year
crime_data['Month'] = crime_data['Date Time OCC'].dt.month
crime_data['Year_Month'] = crime_data['Date Time OCC'].dt.strftime('%Y-%m')
```

In [32]:

```python
# Displaying first 5 rows of the Dataframe
crime_data.head()
```

Out[32]:

|  | DR_NO | Date Rptd | Date Time OCC | AREA NAME | Rpt Dist No | Crm Cd | Crm Cd Desc | Vict Age | Vict Sex | Vict Descent | ... | Status | Status Desc | Crm Cd 1 | LOCATION | L |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **0** | 10304468 | 2020-01-08 | 2020-01-08 22:30:00 | Southwest | 377 | 624 | BATTERY - SIMPLE ASSAULT | 36 | F | B | ... | AO | Adult Other | 624.0 | 1100 W 39TH PL | 34.0 |
| **1** | 190101086 | 2020-01-02 | 2020-01-01 03:30:00 | Central | 163 | 624 | BATTERY - SIMPLE ASSAULT | 25 | M | H | ... | IC | Invest Cont | 624.0 | 700 S HILL ST | 34.04 |
| **2** | 200110444 | 2020-04-14 | 2020-02-13 12:00:00 | Central | 155 | 845 | SEX OFFENDER REGISTRANT OUT OF COMPLIANCE | 0 | X | X | ... | AA | Adult Arrest | 845.0 | 200 E 6TH ST | 34.04 |
| **3** | 191501505 | 2020-01-01 | 2020-01-01 17:30:00 | N Hollywood | 1543 | 745 | VANDALISM - MISDEAMEANOR ($399 OR UNDER) | 76 | F | W | ... | IC | Invest Cont | 745.0 | 5400 CORTEEN PL | 34.16 |
| **4** | 191921269 | 2020-01-01 | 2020-01-01 04:15:00 | Mission | 1998 | 740 | VANDALISM - FELONY ($400 & OVER, ALL CHURCH VA... | 31 | X | X | ... | IC | Invest Cont | 740.0 | 14400 TITUS ST | 34.21 |

5 rows × 22 columns

In [33]:

```python
# Converting Time OCC column into hours and minutes
crime_data1['TIME OCC'] = crime_data1['TIME OCC'].apply(lambda x: f'{x//100:02}:{x%100:02}')
```

In [34]:

```python
# Converting into Date time format
crime_data1['DATE OCC'] = pd.to_datetime(crime_data1['DATE OCC'])
crime_data1.head()
```

```
/var/folders/wj/lvd1hjrx0vqf2hzhdxhvbz580000gn/T/ipykernel_53142/736633326.py:2: UserWarning: Could not infer fo
rmat, so each element will be parsed individually, falling back to `dateutil`. To ensure parsing is consistent a
nd as-expected, please specify a format.
  crime_data1['DATE OCC'] = pd.to_datetime(crime_data1['DATE OCC'])
```

Out[34]:

| | DR_NO | Date Rptd | DATE OCC | TIME OCC | AREA | AREA NAME | Rpt Dist No | Part 1-2 | Crm Cd | Crm Cd Desc | ... | Status Desc | Crm Cd 1 | Crm Cd 2 | Crm Cd 3 | Crm Cd 4 | LOCATION |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **0** | 10304468 | 2020-01-08 | 2020-01-08 | 22:30 | 3 | Southwest | 377 | 2 | 624 | BATTERY - SIMPLE ASSAULT | ... | Adult Other | 624.0 | NaN | NaN | NaN | 1100 W 39TH PL |
| **1** | 190101086 | 2020-01-02 | 2020-01-01 | 03:30 | 1 | Central | 163 | 2 | 624 | BATTERY - SIMPLE ASSAULT | ... | Invest Cont | 624.0 | NaN | NaN | NaN | 700 S HILL ST |
| **2** | 200110444 | 2020-04-14 | 2020-02-13 | 12:00 | 1 | Central | 155 | 2 | 845 | SEX OFFENDER REGISTRANT OUT OF COMPLIANCE | ... | Adult Arrest | 845.0 | NaN | NaN | NaN | 200 E 6TH ST |
| **3** | 191501505 | 2020-01-01 | 2020-01-01 | 17:30 | 15 | N Hollywood | 1543 | 2 | 745 | VANDALISM - MISDEAMEANOR ($399 OR UNDER) | ... | Invest Cont | 745.0 | 998.0 | NaN | NaN | 5400 CORTEEN PL |
| **4** | 191921269 | 2020-01-01 | 2020-01-01 | 04:15 | 19 | Mission | 1998 | 2 | 740 | VANDALISM - FELONY ($400 & OVER, ALL CHURCH VA... | ... | Invest Cont | 740.0 | NaN | NaN | NaN | 14400 TITUS ST |

5 rows × 29 columns

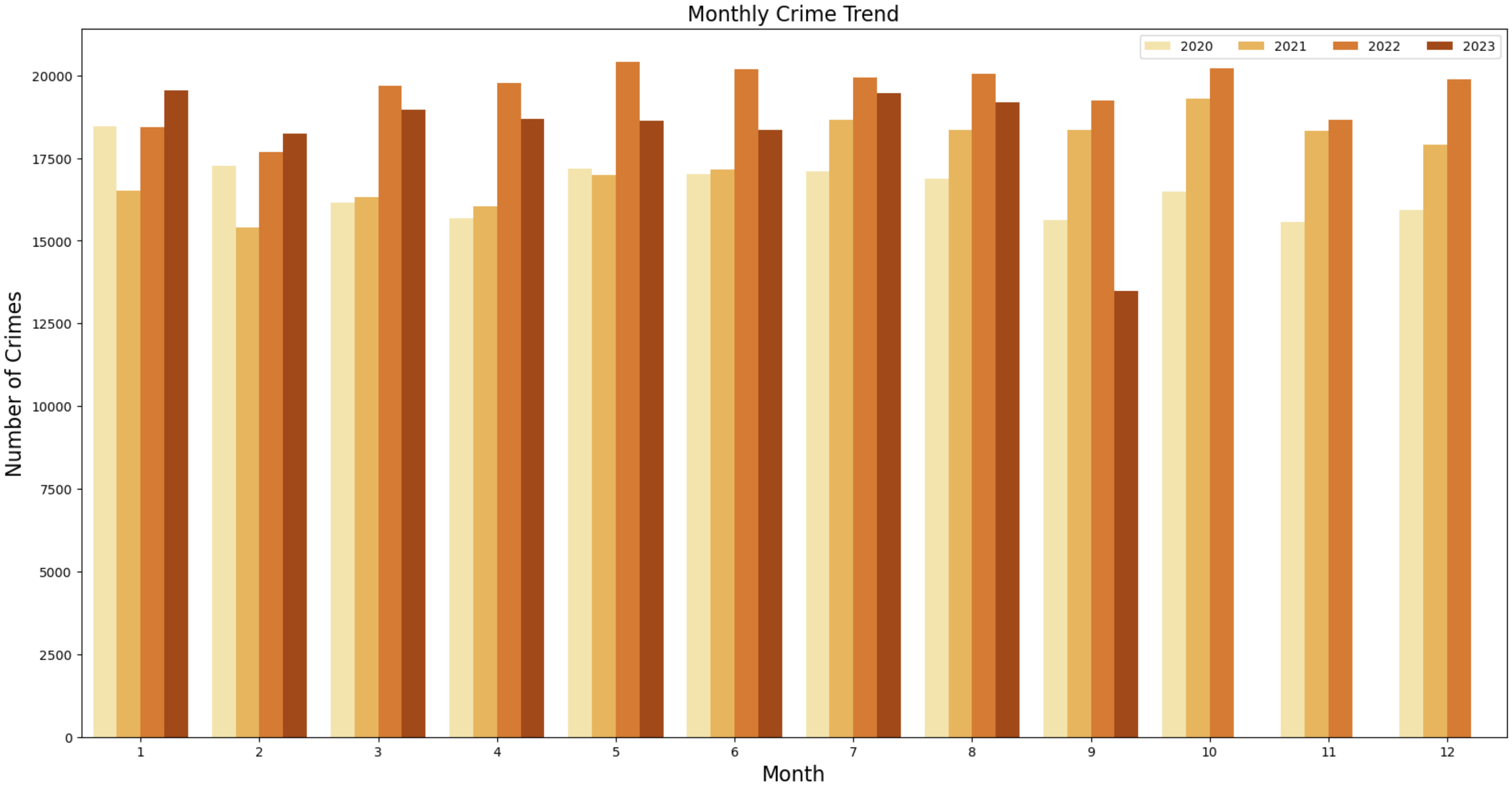
## Visualize overall crime trends from 2020 to the present year.

In [ ]:

```
# Displaying yearly crime count over the years
yearly_crime_counts = crime_data.groupby('Year').size().reset_index(name='Crime Counts')

plt.figure(figsize=(15, 8))
bars = plt.bar(yearly_crime_counts['Year'], yearly_crime_counts['Crime Counts'], color = 'Blue')
plt.ylim(0, 300000)

plt.plot(yearly_crime_counts['Year'], yearly_crime_counts['Crime Counts'], color='black', marker='o')

for x, y in zip(yearly_crime_counts['Year'], yearly_crime_counts['Crime Counts']):
    plt.text(x, y + 0.01 * 300000, f'{y}', ha='center', va='bottom')

plt.title('Crime Trends from 2020 to Present (Yearly)', fontsize = 16)
plt.xlabel('Year', fontsize = 16)
plt.ylabel('Crimes', fontsize = 16)
# plt.grid(axis='y')
plt.show()
```

In [36]:

```
# Displaying monthly crime trends
plt.figure(figsize=(20,10))
sns.countplot(data=crime_data, x='Month', hue='Year', palette='YlOrBr')
plt.title('Monthly Crime Trend',fontsize = 16)
plt.legend(ncol = 4)
plt.xlabel('Month', fontsize = 16)
plt.ylabel('Number of Crimes', fontsize = 16)
plt.show()
```



# Analyze and visualize seasonal patterns in crime data.

In [37]:

```
# Displaying first 5 rows of the dataframe
crime_data.head()
```

Out[37]:

| | DR_NO | Date Rptd | Date Time OCC | AREA NAME | Rpt Dist No | Crm Cd | Crm Cd Desc | Vict Age | Vict Sex | Vict Descent | ... | Status | Status Desc | Crm Cd 1 | LOCATION | L |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **0** | 10304468 | 2020-01-08 | 2020-01-08 22:30:00 | Southwest | 377 | 624 | BATTERY - SIMPLE ASSAULT | 36 | F | B | ... | AO | Adult Other | 624.0 | 1100 W 39TH PL | 34.0 |
| **1** | 190101086 | 2020-01-02 | 2020-01-01 03:30:00 | Central | 163 | 624 | BATTERY - SIMPLE ASSAULT | 25 | M | H | ... | IC | Invest Cont | 624.0 | 700 S HILL ST | 34.04 |
| **2** | 200110444 | 2020-04-14 | 2020-02-13 12:00:00 | Central | 155 | 845 | SEX OFFENDER REGISTRANT OUT OF COMPLIANCE | 0 | X | X | ... | AA | Adult Arrest | 845.0 | 200 E 6TH ST | 34.04 |
| **3** | 191501505 | 2020-01-01 | 2020-01-01 17:30:00 | N Hollywood | 1543 | 745 | VANDALISM - MISDEAMEANOR ($399 OR UNDER) | 76 | F | W | ... | IC | Invest Cont | 745.0 | 5400 CORTEEN PL | 34.16 |
| **4** | 191921269 | 2020-01-01 | 2020-01-01 04:15:00 | Mission | 1998 | 740 | VANDALISM - FELONY ($400 & OVER, ALL CHURCH VA... | 31 | X | X | ... | IC | Invest Cont | 740.0 | 14400 TITUS ST | 34.21 |

5 rows × 22 columns

In [38]:

```
# Grouping the monthly average crime
monthly_avg = crime_data.groupby(crime_data['Month']).size()
```

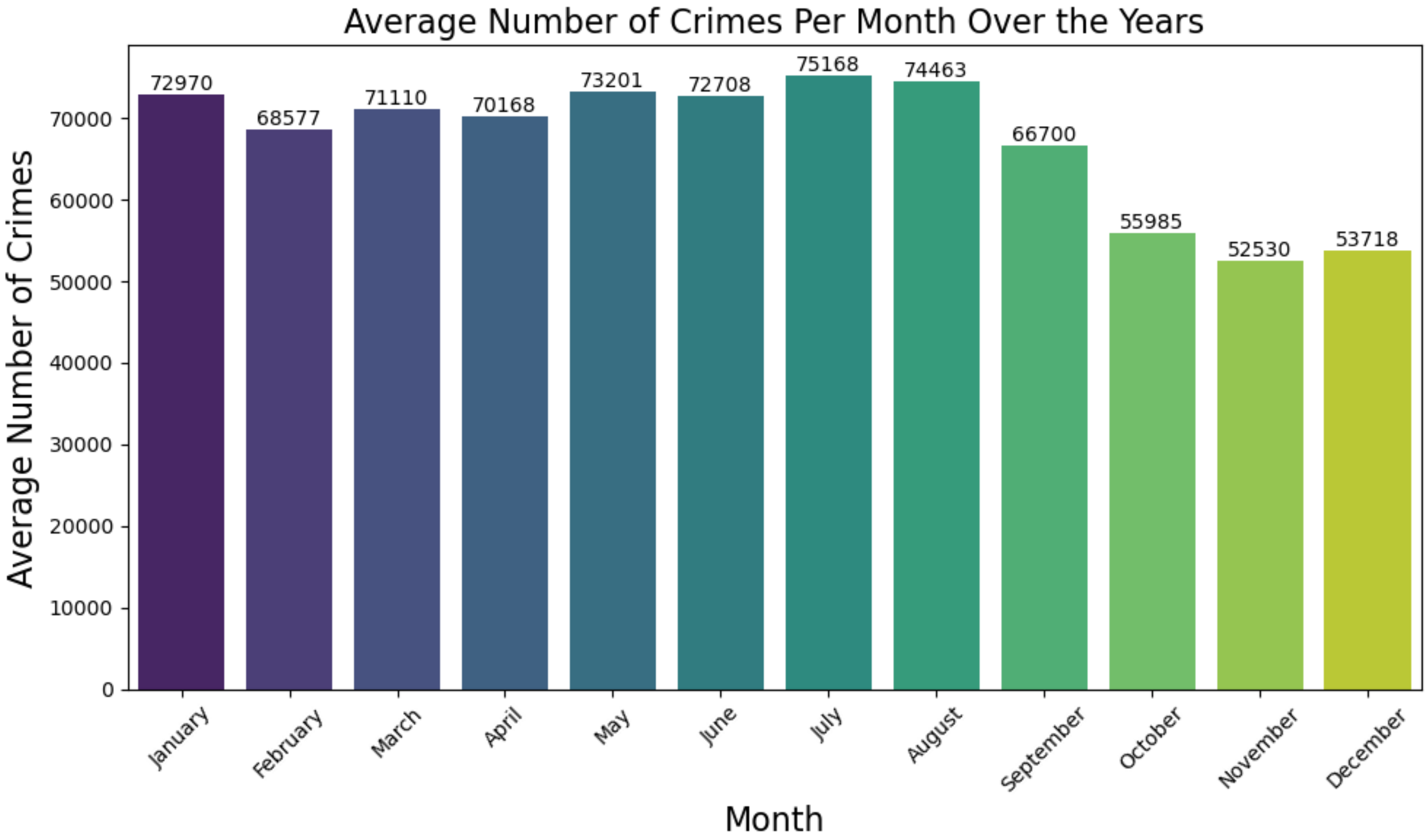
In [39]:

```
# Displaying Descriptive Statistics for Monthly crime
print("Descriptive Statistics for Monthly Crimes:")
print(monthly_avg.describe())
```

```
Descriptive Statistics for Monthly Crimes:
count       12.000000
mean     67274.833333
std       8338.348756
min      52530.000000
25%      64021.250000
50%      70639.000000
75%      73027.750000
max      75168.000000
dtype: float64
```

In [40]:

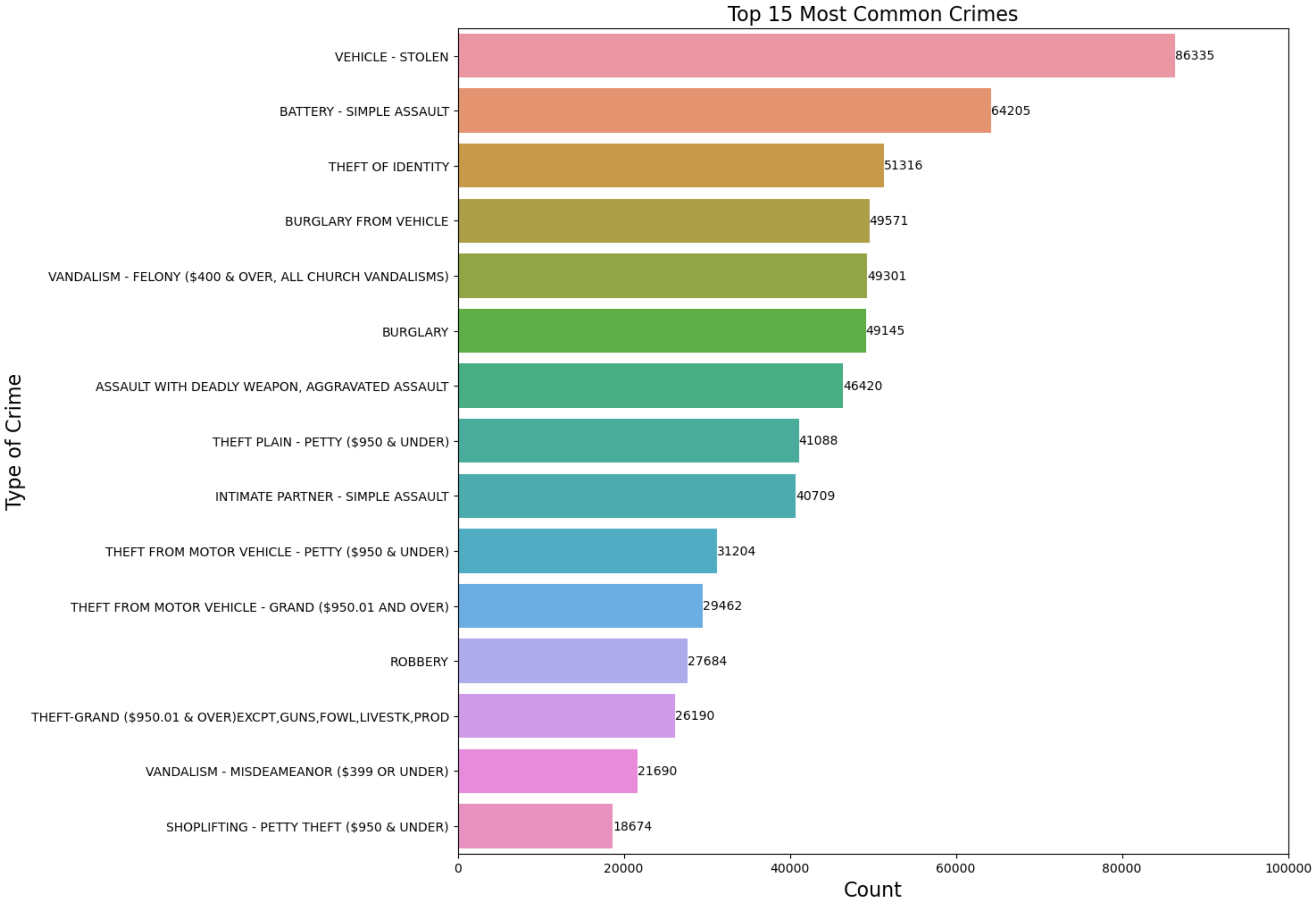
```
# Displaying Average number of crimes per month over the years
plt.figure(figsize=(10, 6))
ax = sns.barplot(data=monthly_avg.reset_index(), x='Month', y=0, palette='viridis')
plt.title('Average Number of Crimes Per Month Over the Years', fontsize = 16)
plt.ylabel('Average Number of Crimes', fontsize = 16)
plt.xlabel('Month', fontsize = 16)
plt.xticks(ticks=range(12), labels=['January', 'February', 'March', 'April', 'May', 'June',
                                    'July', 'August', 'September', 'October', 'November', 'December'], rotation=4
for p in ax.patches:
    ax.annotate(f'{int(p.get_height())}', (p.get_x() + p.get_width() / 2., p.get_height()), ha='center', va='cent

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



Identify the most common type of crime and its trends over time.

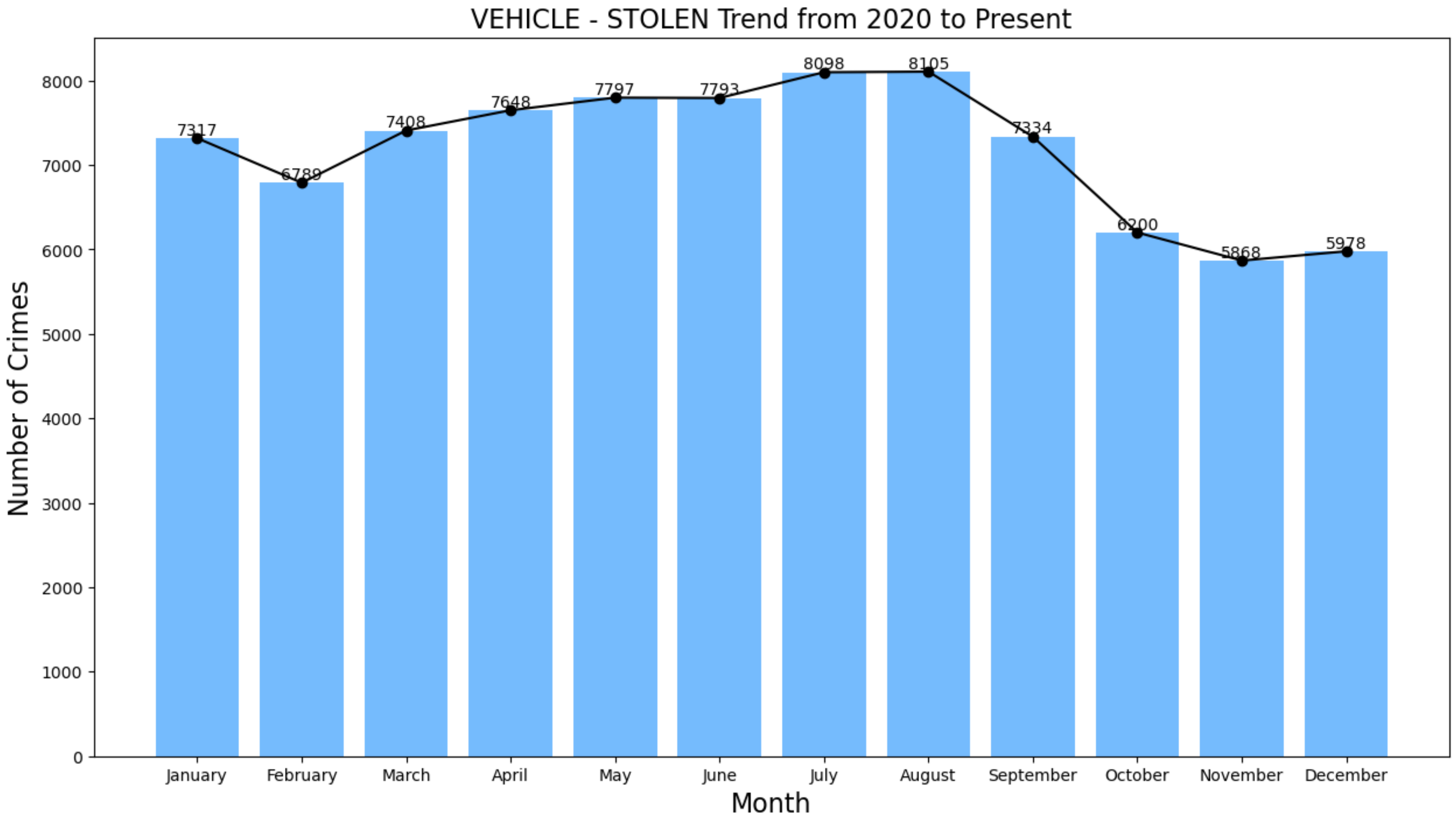
In [41]:

```
# Displaying most common type of crimes
plt.figure(figsize=(12, 12))

top_crimes = crime_data['Crm Cd Desc'].value_counts().nlargest(15)

crime_plot = sns.countplot(data=crime_data, y='Crm Cd Desc',order=top_crimes.index)

plt.title('Top 15 Most Common Crimes', fontsize = 16)
plt.xlabel('Count', fontsize = 16)
plt.ylabel('Type of Crime', fontsize = 16)
plt.xlim(0, 100000)

for bar in crime_plot.patches:
    plt.text(
        bar.get_width(),
        bar.get_y() + bar.get_height() / 2,
        f'{int(bar.get_width())}',
        va='center'
    )
plt.show()
```


Top 15 Most Common Crimes
Type of Crime
VEHICLE - STOLEN
86335
BATTERY - SIMPLE ASSAULT
64205
THEFT OF IDENTITY
51316
BURGLARY FROM VEHICLE
49571
VANDALISM - FELONY ($400 & OVER, ALL CHURCH VANDALISMS)
49301
BURGLARY
49145
ASSAULT WITH DEADLY WEAPON, AGGRAVATED ASSAULT
46420
THEFT PLAIN - PETTY ($950 & UNDER)
41088
INTIMATE PARTNER - SIMPLE ASSAULT
40709
THEFT FROM MOTOR VEHICLE - PETTY ($950 & UNDER)
31204
THEFT FROM MOTOR VEHICLE - GRAND ($950.01 AND OVER)
29462
ROBBERY
27684
THEFT-GRAND ($950.01 & OVER)EXCPT,GUNS,FOWL,LIVESTK,PROD
26190
VANDALISM - MISDEAMEANOR ($399 OR UNDER)
21690
SHOPLIFTING - PETTY THEFT ($950 & UNDER)
18674
0
20000
40000
60000
80000
100000
Count

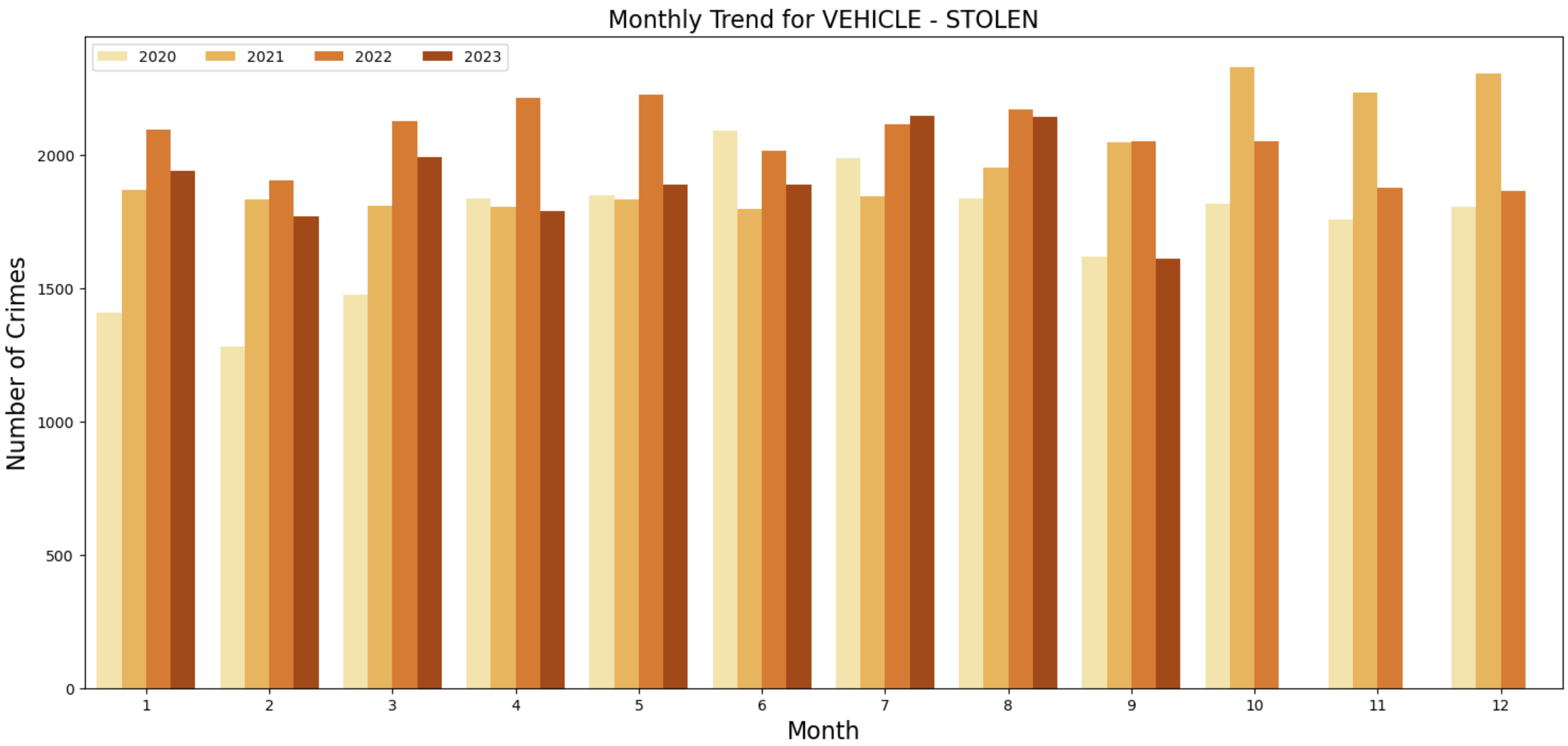
In [42]:

```python
# Displaying Top crime trend over the years
top_crime = crime_data['Crm Cd Desc'].value_counts().index[0]

filtered_data = crime_data[crime_data['Crm Cd Desc'] == top_crime]

monthly_trend = filtered_data.groupby('Month').size().reset_index(name='Counts')

monthly_trend.sort_values('Month', inplace=True)

monthly_trend['Month'] = monthly_trend['Month'].apply(lambda x: calendar.month_name[x])

plt.figure(figsize=(15, 8))
bars = plt.bar(monthly_trend['Month'], monthly_trend['Counts'], color='xkcd:sky blue')
plt.plot(monthly_trend['Month'], monthly_trend['Counts'], color='black', marker='o')

for x, y in zip(monthly_trend['Month'], monthly_trend['Counts']):
    plt.text(x, y, f'{y}', ha='center', va='bottom')

plt.title(f'{top_crime} Trend from 2020 to Present', fontsize = 16)
plt.xlabel('Month', fontsize = 16)
plt.ylabel('Number of Crimes', fontsize = 16)
plt.show()
```


VEHICLE - STOLEN Trend from 2020 to Present
Number of Crimes
Month
0
1000
2000
3000
4000
5000
6000
7000
8000
January
February
March
April
May
June
July
August
September
October
November
December
7317
6789
7408
7648
7797
7793
8098
8105
7334
6200
5868
5978

In [43]:

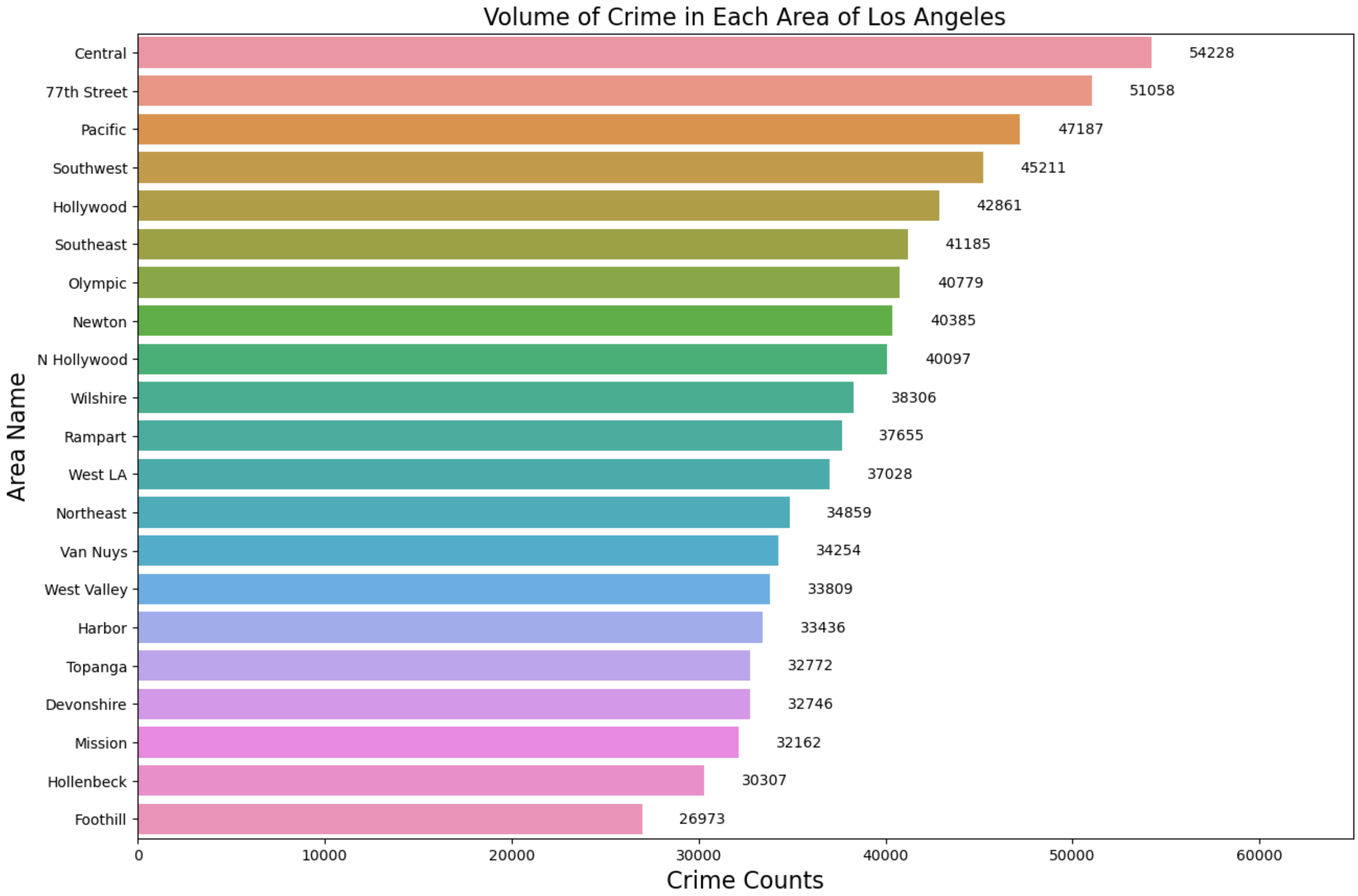
```
#Monthly Trend for Vehicle - Stolen
top_crime = crime_data['Crm Cd Desc'].value_counts().index[0]
filtered_data = crime_data[crime_data['Crm Cd Desc'] == top_crime]
plt.figure(figsize=(18, 8))
sns.countplot(data=filtered_data, x='Month', hue='Year', palette='YlOrBr')
plt.title(f'Monthly Trend for {top_crime}', fontsize = 16)
plt.legend(ncol = 4)
plt.xlabel('Month', fontsize = 16)
plt.ylabel('Number of Crimes', fontsize = 16)
plt.show()
```



## Investigate if there are any notable differences in crime rates between regions or cities

In [44]:

```
# Displaying any notable differences in crime rates between regions or cities
plt.figure(figsize=(15, 10))

order = crime_data['AREA NAME'].value_counts().index
sns.countplot(y='AREA NAME', data=crime_data, order=order)

plt.xlabel('Crime Counts', fontsize = 16)
plt.ylabel('Area Name', fontsize = 16)
plt.title('Volume of Crime in Each Area of Los Angeles', fontsize = 16)
plt.xlim(0,65000)

# Adding the count at the end of each bar
for bar in plt.gca().patches:
    plt.text(bar.get_width() + 2000,  # Adding 2000 to move the text slightly right
             bar.get_y() + bar.get_height()/2,
             str(int(bar.get_width())),
             va="center")

plt.show()
```



In [45]:

```
# Calculating Crime statistics based on area name
statistics = crime_data['AREA NAME'].value_counts().describe()
print(statistics)
```

```
count       21.000000
mean     38442.761905
std       6895.542763
min      26973.000000
25%      33436.000000
50%      37655.000000
75%      41185.000000
max      54228.000000
Name: count, dtype: float64
```

## Explore correlations between economic factors (if available) and crime rates

In [46]:

```
# Loading Monthly Inflation Data
infla = pd.read_csv("Inflation_rate.csv")
```

**Reference:**

https://data.bls.gov/pdq/SurveyOutputServlet - for Unemployment Data

https://www.bls.gov/regions/mid-atlantic/data/consumerpriceindexhistorical_us_table.htm - for Inflation Rate

In [47]:

```
# Displaying first 5 rows of the dataframe
infla.head()
```

Out[47]:

| | Date | Inflation Rate | Unemployment rate |
|---|---|---|---|
| **0** | 2020-01 | 0.2 | 4.9 |
| **1** | 2020-02 | 0.1 | 4.9 |
| **2** | 2020-03 | -0.4 | 6.7 |
| **3** | 2020-04 | -0.8 | 17.0 |
| **4** | 2020-05 | -0.1 | 19.0 |

In [48]:

```python
# Creating a new dataframe with monthly crime counts
df = crime_data.copy()
df['month-year'] = df['Date Time OCC'].dt.strftime('%m-%Y')
df = df.groupby('month-year')['Crm Cd'].mean()
df_1 = df.to_frame().reset_index()
df_1 = df_1.rename(columns={'month-year':'Date','Crm Cd':'Crime'})
df_1['Date'] = pd.to_datetime(df_1['Date'])
df_1['days_in_month'] = df_1['Date'].dt.days_in_month
df_1.head()
```

```
/var/folders/wj/lvd1hjrx0vqf2hzhdxhvbz580000gn/T/ipykernel_53142/1104235384.py:7: UserWarning: Could not infer f
ormat, so each element will be parsed individually, falling back to `dateutil`. To ensure parsing is consistent
and as-expected, please specify a format.
  df_1['Date'] = pd.to_datetime(df_1['Date'])
```

Out[48]:

| | Date | Crime | days_in_month |
|---|---|---|---|
| **0** | 2020-01-01 | 510.458290 | 31 |
| **1** | 2021-01-01 | 504.323570 | 31 |
| **2** | 2022-01-01 | 493.799446 | 31 |
| **3** | 2023-01-01 | 488.039237 | 31 |
| **4** | 2020-02-01 | 513.959019 | 29 |

In [49]:

```python
 # Creating a Dataset with Montly crimes from Jan 2020 and adding each month inflation to it.
df_1['Unemployment'] = infla['Unemployment rate']
df_1['Inflation'] = infla['Inflation Rate']

df_1.head()
```

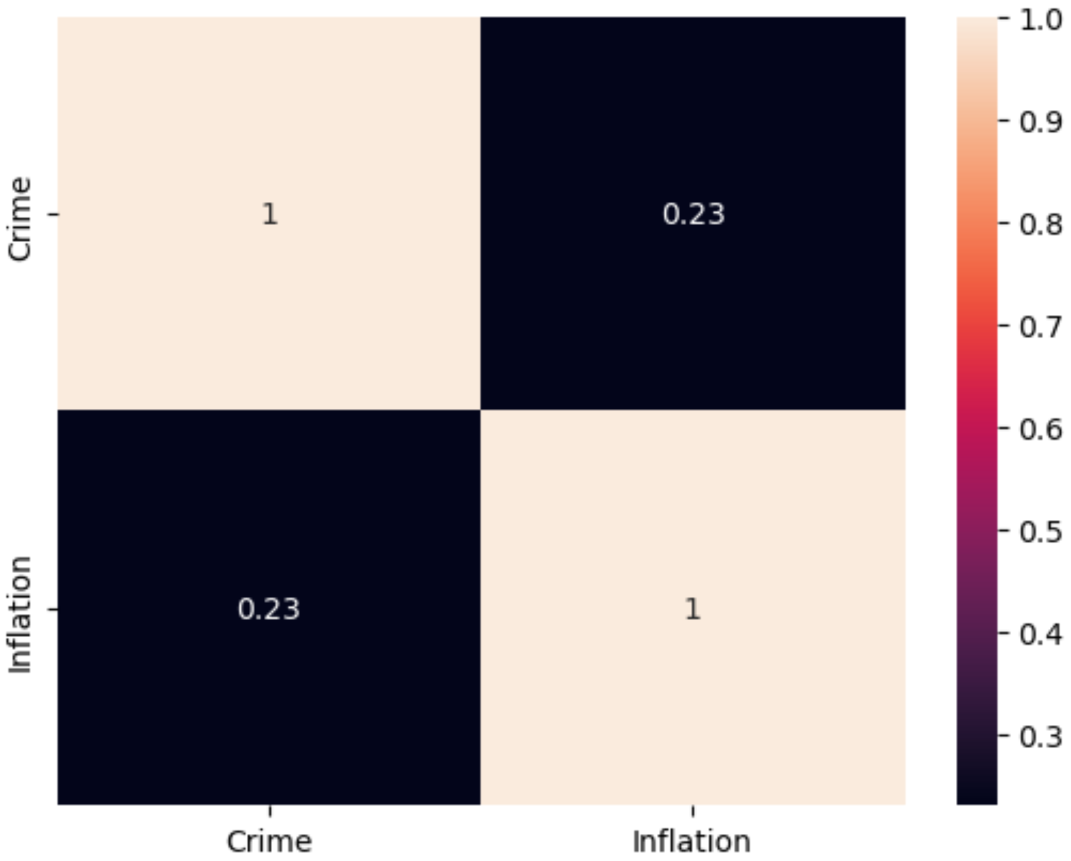
Out[49]:

|   | Date | Crime | days_in_month | Unemployment | Inflation |
|---|---|---|---|---|---|
| **0** | 2020-01-01 | 510.458290 | 31 | 4.9 | 0.2 |
| **1** | 2021-01-01 | 504.323570 | 31 | 4.9 | 0.1 |
| **2** | 2022-01-01 | 493.799446 | 31 | 6.7 | -0.4 |
| **3** | 2023-01-01 | 488.039237 | 31 | 17.0 | -0.8 |
| **4** | 2020-02-01 | 513.959019 | 29 | 19.0 | -0.1 |

In [50]:

```
# Calculating the Correlation between Crime and Inflation
corr_matrix = df_1[['Crime', 'Inflation']].astype(float).corr()
print(corr_matrix)
sns.heatmap(corr_matrix, annot=True)
plt.show()
```

```
              Crime  Inflation
Crime      1.000000   0.230833
Inflation  0.230833   1.000000
```

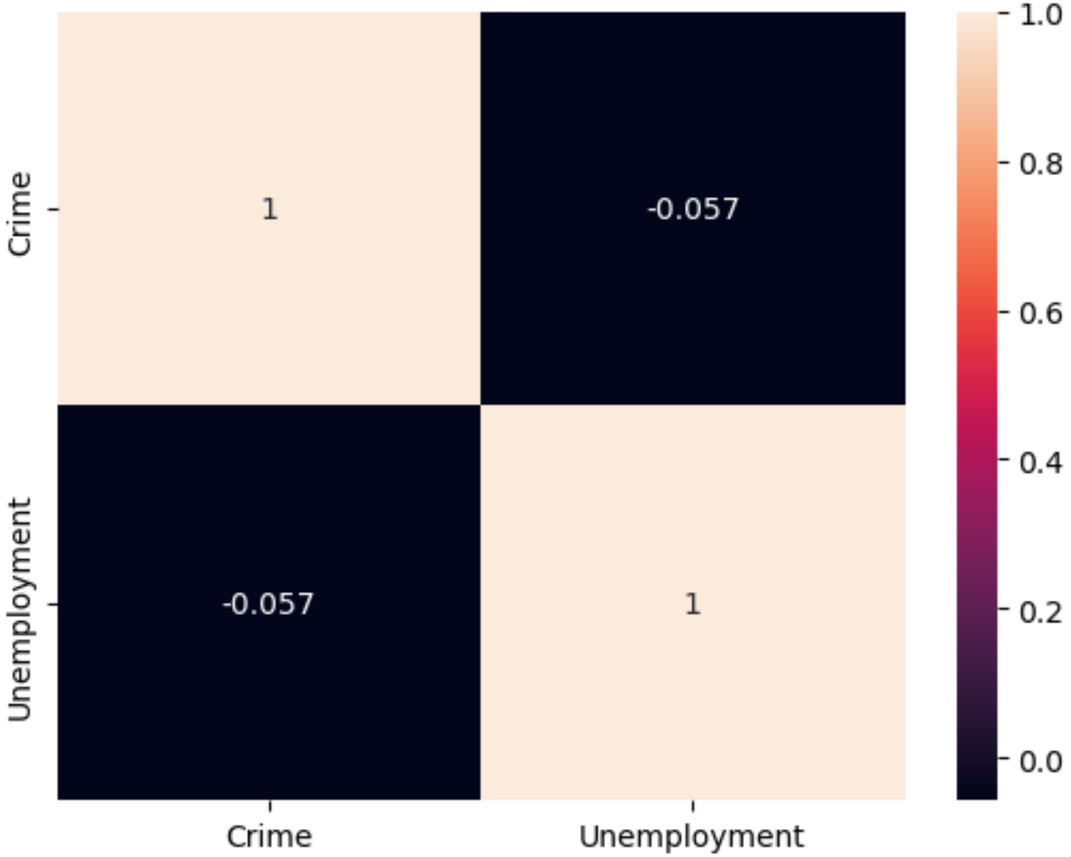


As we can observe from the above heatmap there is weak Positive Correlation (0.23) between Inflation rate and Crime rate.

In [51]:

```
# Calculating the Correlation between Crime and Unemployment
corr_matrix = df_1[['Crime', 'Unemployment']].astype(float).corr()
print(corr_matrix)
sns.heatmap(corr_matrix, annot=True)
plt.show()
```

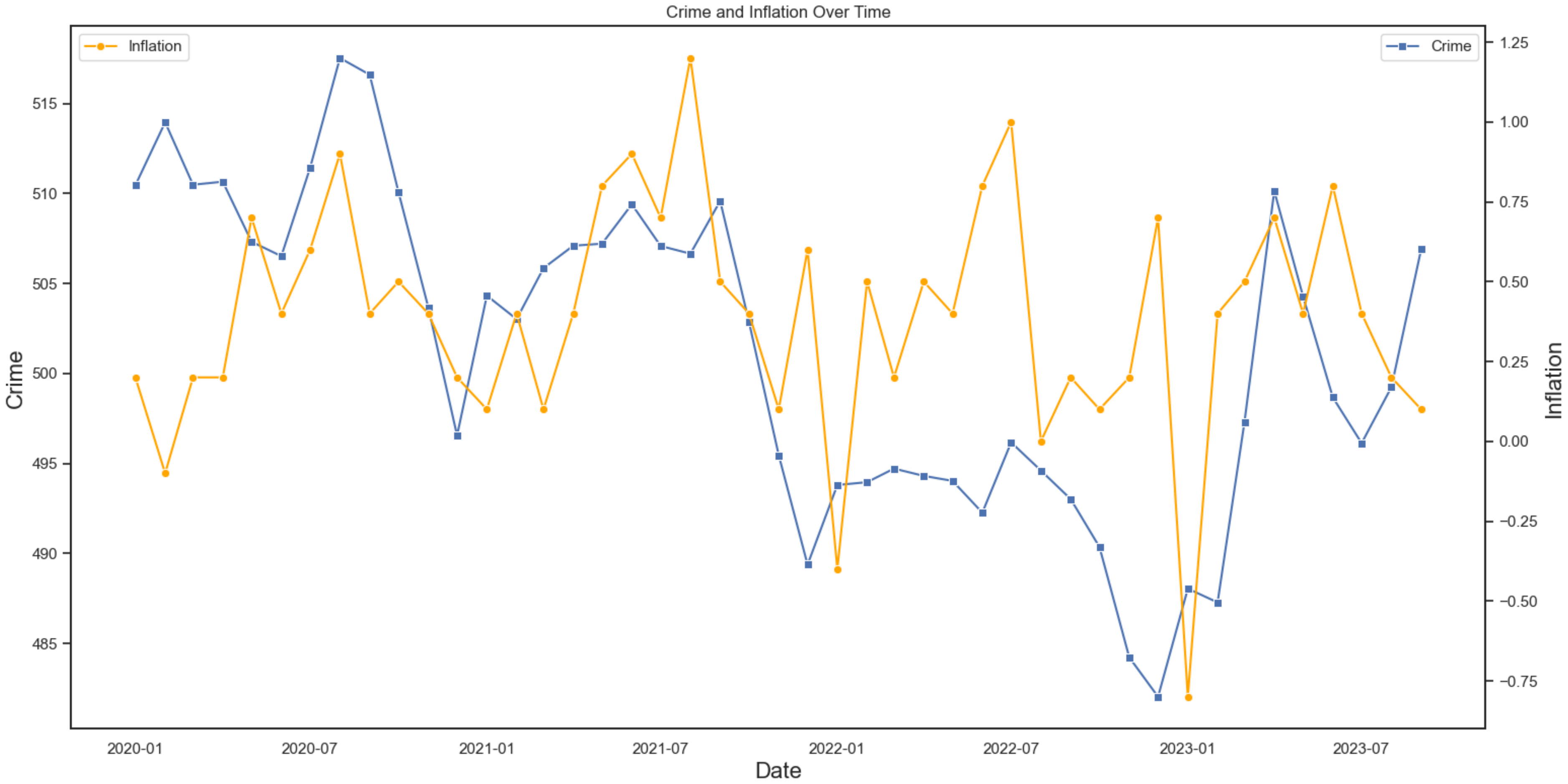
```
                 Crime  Unemployment
Crime         1.000000     -0.057009
Unemployment -0.057009      1.000000
```



As we can observe from the above heatmap there is no Correlation (-0.057) between Unemployment rate and Crime rate.

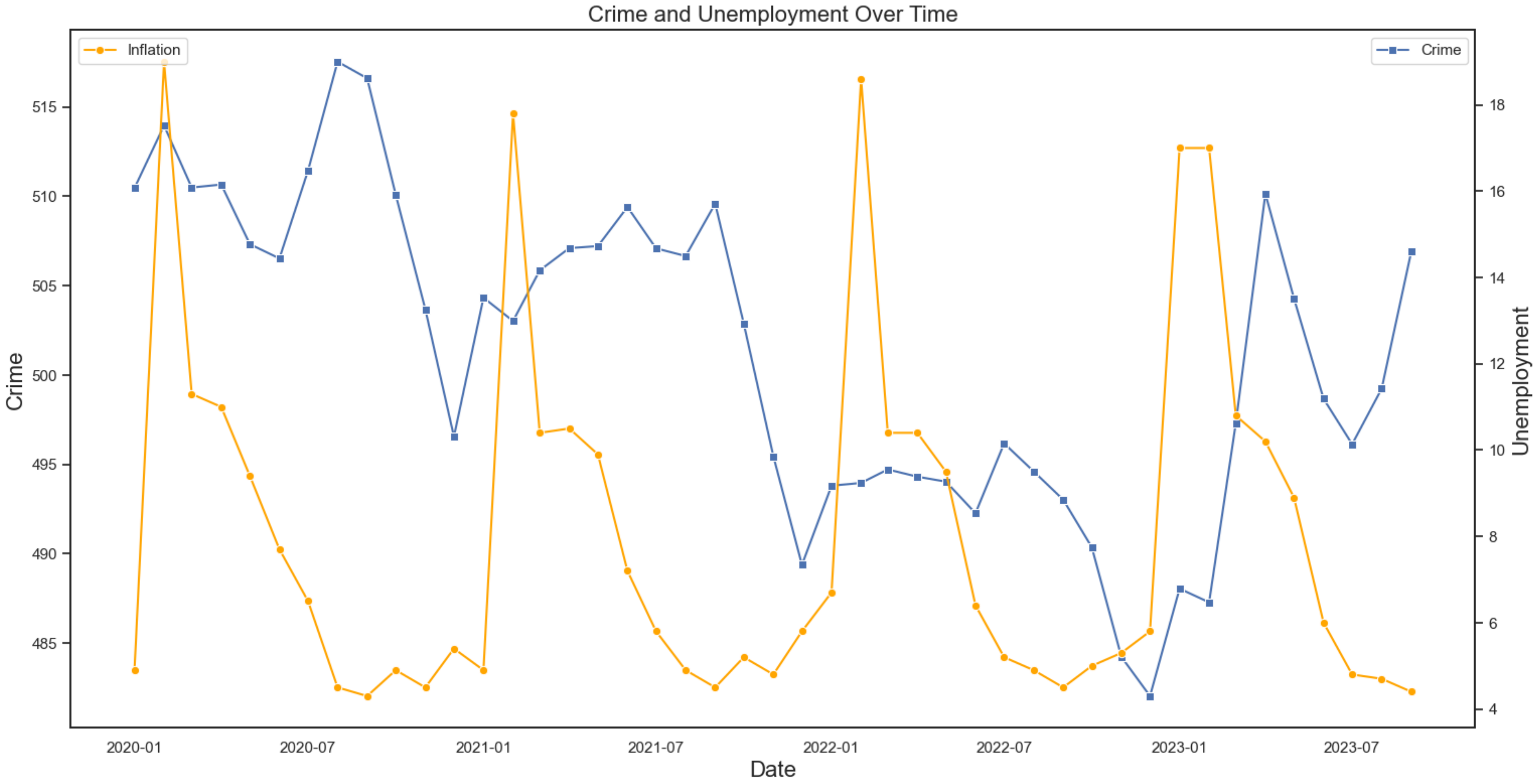
In [52]:

```python
# Plotting the crime trend and Inflation rate over the years
import seaborn as sns
import matplotlib.pyplot as plt
import pandas as pd

# Set the plotting style
sns.set(style="white")

# Create the first plot
plt.figure(figsize=(18, 9))  # Adjust the figure size if needed
ax1 = sns.lineplot(x='Date', y='Crime', data=df_1, marker='s', label='Crime')

# Create the second plot with a secondary y-axis
ax2 = ax1.twinx()
sns.lineplot(x='Date', y='Inflation', data=df_1, ax=ax2, marker='o', color='orange', label='Inflation')

# Set labels and title
ax1.set_xlabel('Date', fontsize = 16)
ax1.set_ylabel('Crime', fontsize = 16)
ax2.set_ylabel('Inflation', fontsize = 16)
plt.title('Crime and Inflation Over Time')
plt.legend(loc = 'upper left')


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


Crime and Inflation Over Time
Inflation
Crime
Crime
Inflation
Date
515
510
505
500
495
490
485
1.25
1.00
0.75
0.50
0.25
0.00
−0.25
−0.50
−0.75
2020-01
2020-07
2021-01
2021-07
2022-01
2022-07
2023-01
2023-07

In [53]:

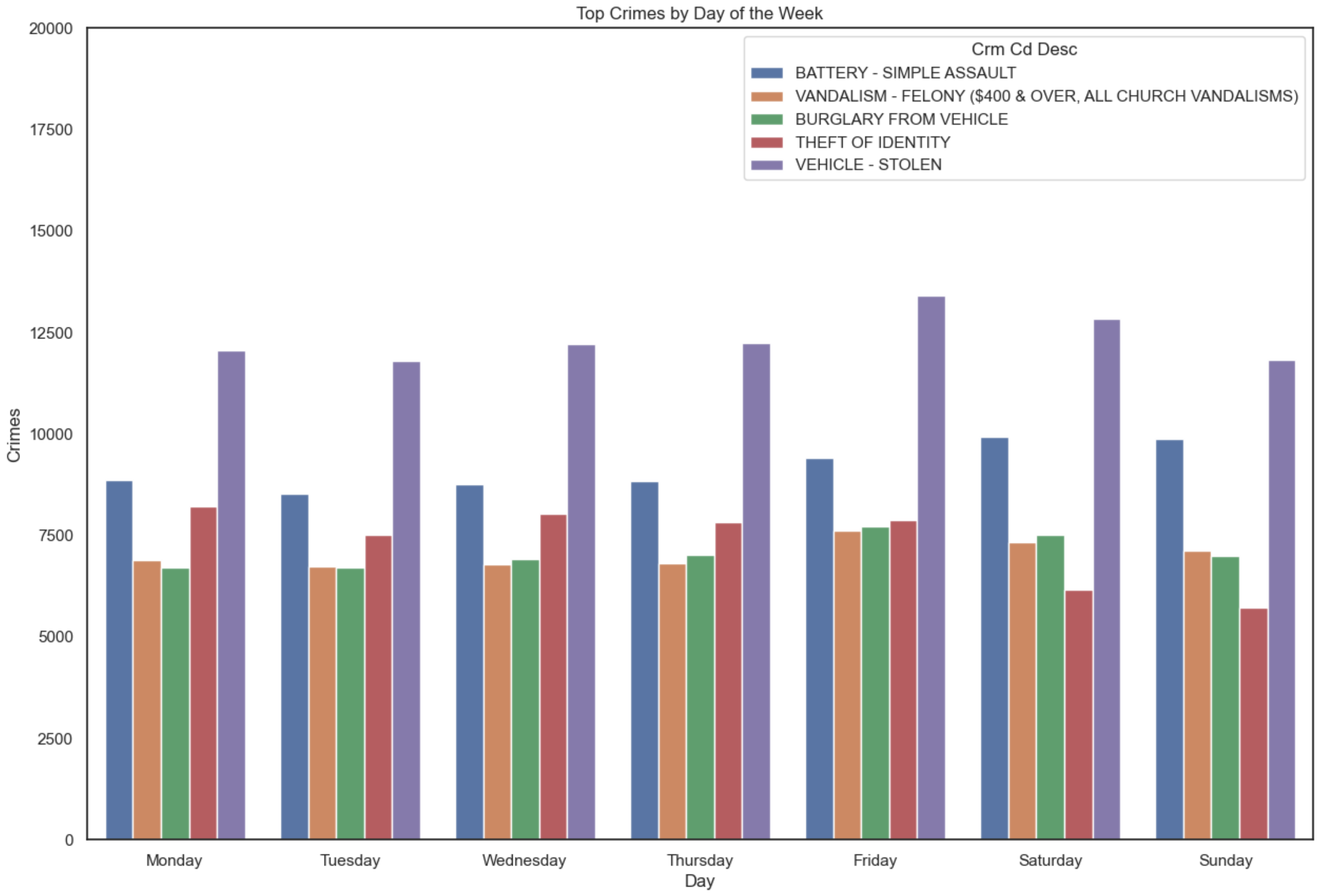
```
# Plotting Crime and Unemployment rate over the years
import seaborn as sns
import matplotlib.pyplot as plt
import pandas as pd

# Set the plotting style
sns.set(style="white")

# Create the first plot
plt.figure(figsize=(18, 9))  # Adjust the figure size if needed
ax1 = sns.lineplot(x='Date', y='Crime', data=df_1, marker='s', label='Crime')

# Create the second plot with a secondary y-axis
ax2 = ax1.twinx()
sns.lineplot(x='Date', y='Unemployment', data=df_1, ax=ax2, marker='o', color='orange', label='Inflation')

# Set labels and title
ax1.set_xlabel('Date',fontsize = 16)
ax1.set_ylabel('Crime',fontsize = 16)
ax2.set_ylabel('Unemployment',fontsize = 16)
plt.title('Crime and Unemployment Over Time',fontsize = 16)
plt.legend(loc = 'upper left')


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



Analyze the relationship between the day of the week and the frequency of certain types of crimes.

In [54]:

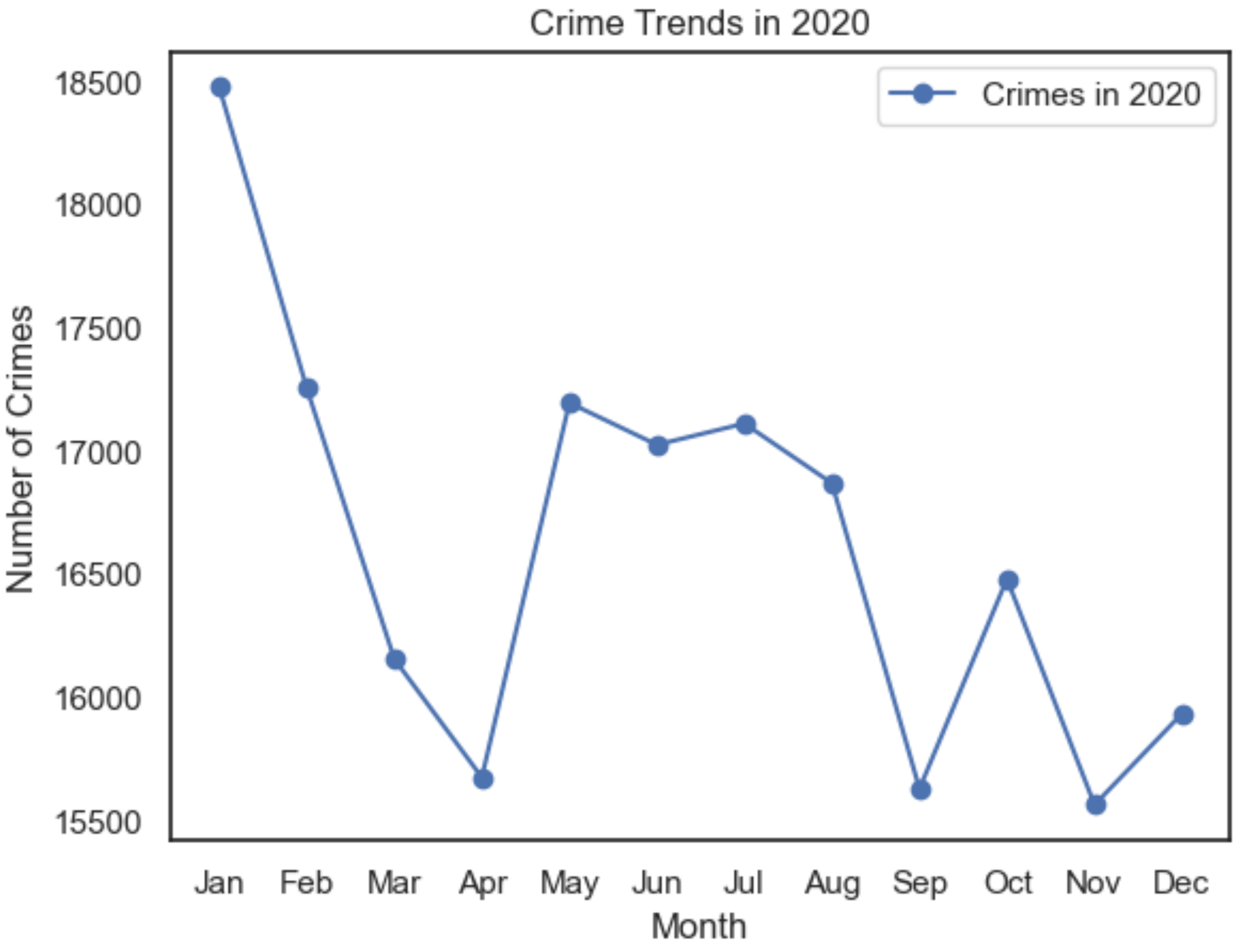
```
# Displaying relationship between the day of the week and the frequency of certain types of crimes.
crime_data['Date Time OCC'] = pd.to_datetime(crime_data['Date Time OCC'])
crime_data['DayOfWeek'] = crime_data['Date Time OCC'].dt.day_name()
top_crimes = crime_data['Crm Cd Desc'].value_counts().index[:5]
filtered_df = crime_data[crime_data['Crm Cd Desc'].isin(top_crimes)]

order_days = ['Monday', 'Tuesday', 'Wednesday', 'Thursday', 'Friday', 'Saturday', 'Sunday']

plt.figure(figsize=(15, 10))
sns.countplot(x='DayOfWeek', hue='Crm Cd Desc', data=filtered_df, order=order_days)
plt.title('Top Crimes by Day of the Week')
plt.xlabel('Day')
plt.ylabel('Crimes')
plt.ylim(0, 20000)
plt.show()
```

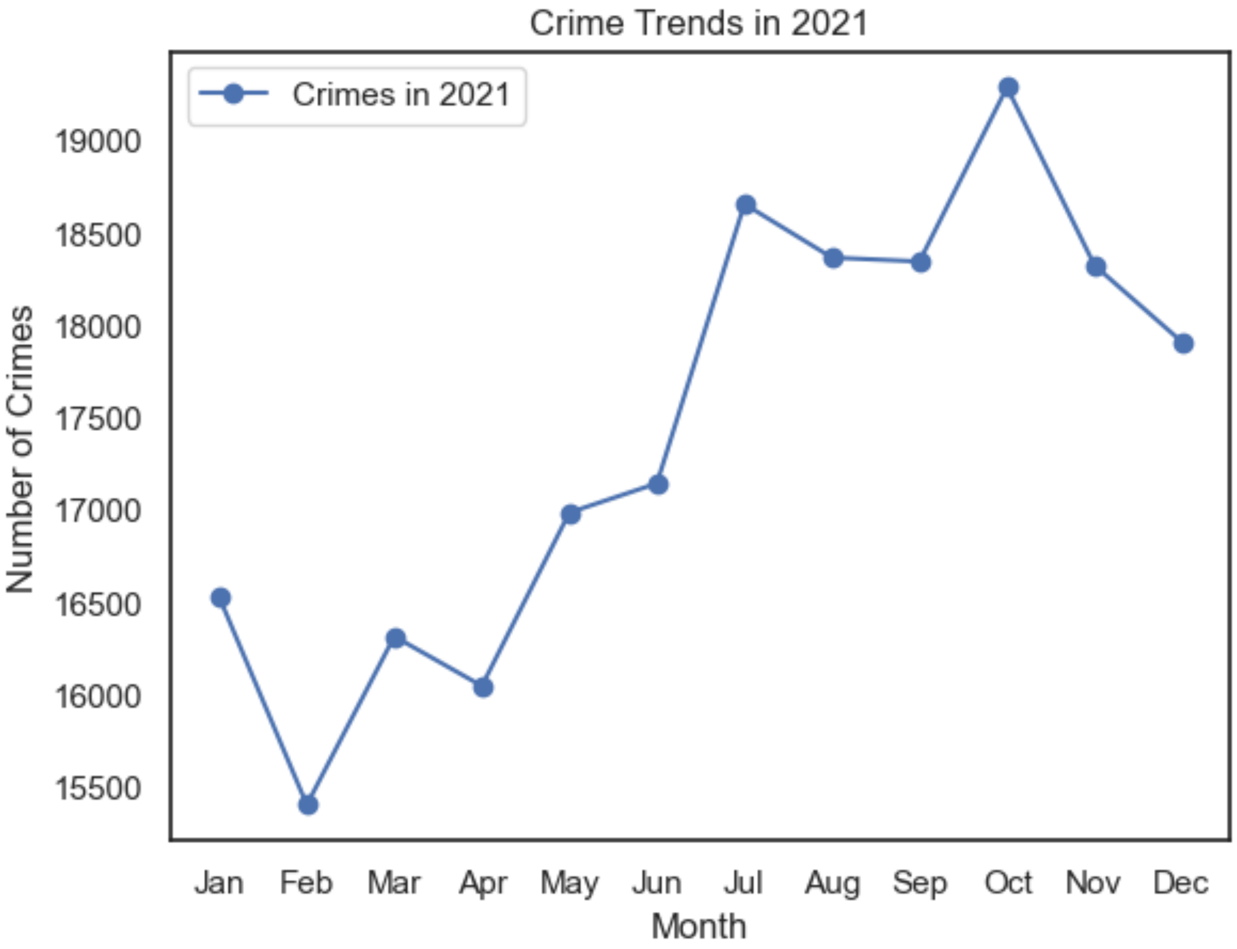

Top Crimes by Day of the Week
Crm Cd Desc
BATTERY - SIMPLE ASSAULT
VANDALISM - FELONY ($400 & OVER, ALL CHURCH VANDALISMS)
BURGLARY FROM VEHICLE
THEFT OF IDENTITY
VEHICLE - STOLEN
Crimes
20000
17500
15000
12500
10000
7500
5000
2500
0
Monday
Tuesday
Wednesday
Thursday
Friday
Saturday
Sunday
Day

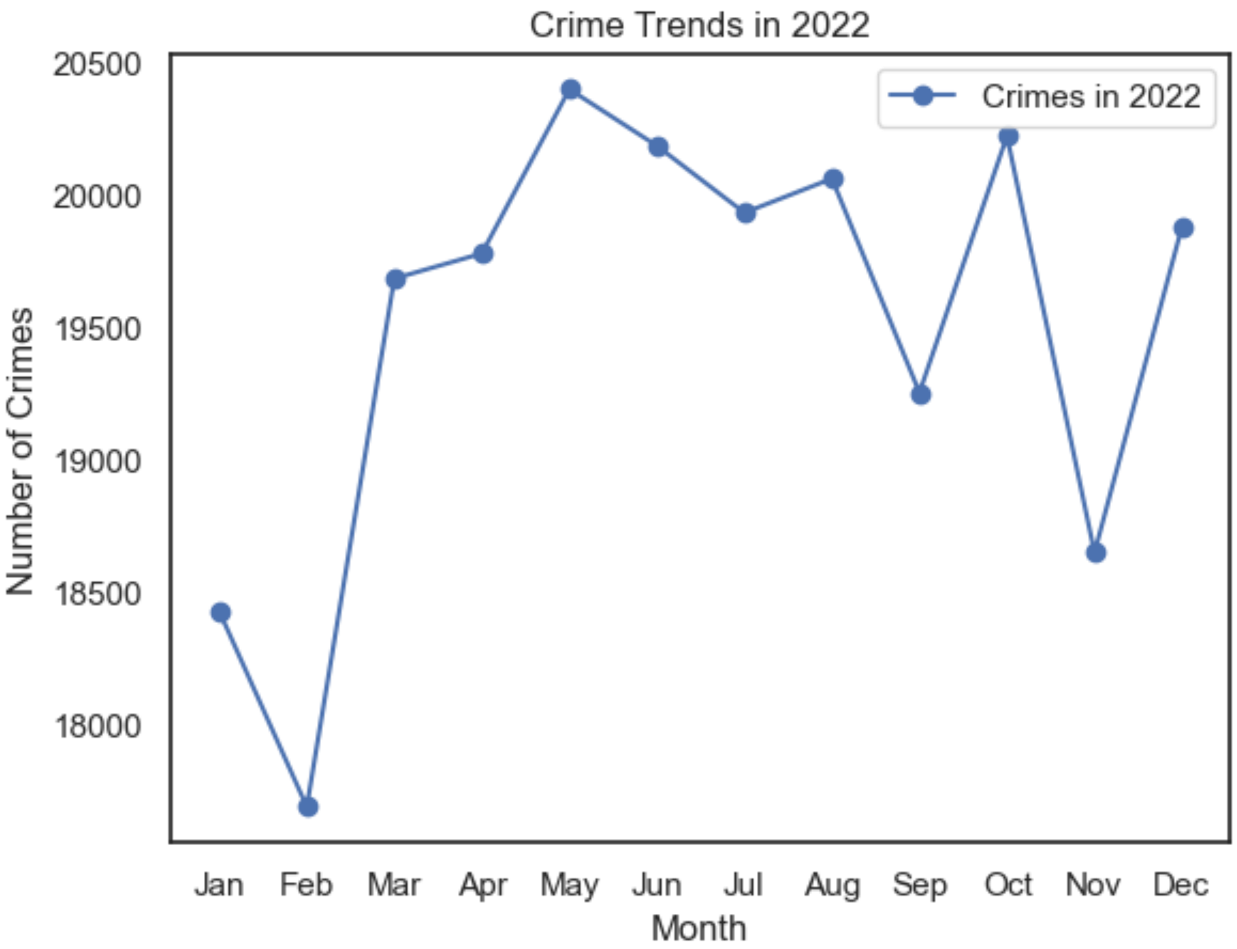
## Impact of Major Events

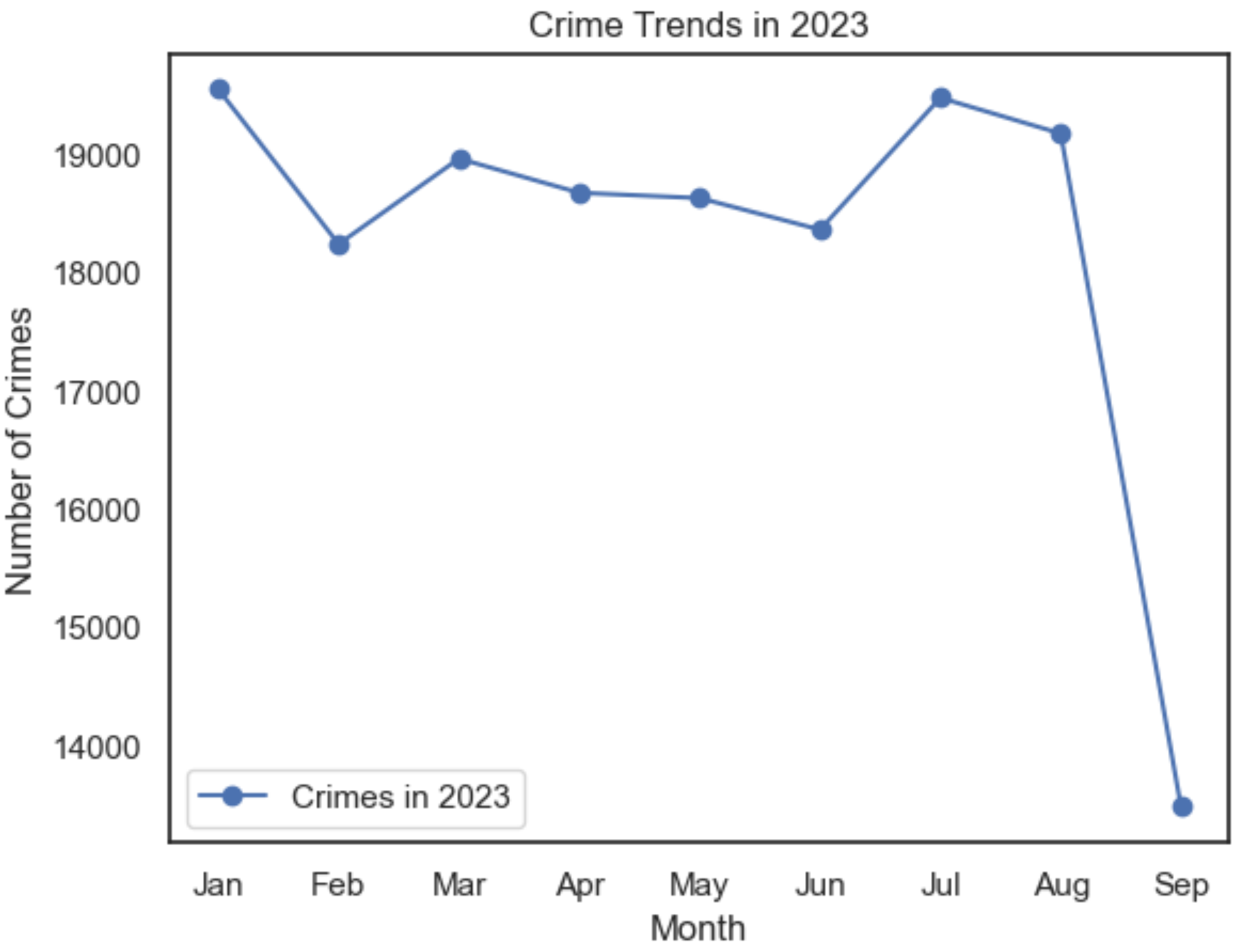
In [55]:

```
# Displaying Crime rates year wise
import pandas as pd
import matplotlib.pyplot as plt

# Assuming df is your DataFrame with the 'Year' and 'Month' columns

# Group the data by year and month, then count the number of crimes for each year and month
yearly_monthly_crime_counts = crime_data.groupby(['Year', 'Month']).size().unstack()

# Create a line chart for each year
for year in yearly_monthly_crime_counts.index:
    monthly_data = yearly_monthly_crime_counts.loc[year]
    # Replace numerical month values with month names
    monthly_data.index = [pd.Timestamp(2000, m, 1).strftime('%b') for m in monthly_data.index]

    plt.figure()  # Create a new figure for each year
    plt.plot(monthly_data.index, monthly_data.values, marker='o', linestyle='-', label=f'Crimes in {year}')
    plt.title(f'Crime Trends in {year}')
    plt.xlabel('Month')
    plt.ylabel('Number of Crimes')
    plt.legend()

plt.show()  # Display all the individual line charts
```


Crime Trends in 2020
Crimes in 2020
Number of Crimes
18500
18000
17500
17000
16500
16000
15500
Jan
Feb
Mar
Apr
May
Jun
Jul
Aug
Sep
Oct
Nov
Dec
Month


Crime Trends in 2021
Crimes in 2021
Number of Crimes
19000
18500
18000
17500
17000
16500
16000
15500
Jan
Feb
Mar
Apr
May
Jun
Jul
Aug
Sep
Oct
Nov
Dec
Month


Crime Trends in 2022
Crimes in 2022
Number of Crimes
20500
20000
19500
19000
18500
18000
Jan
Feb
Mar
Apr
May
Jun
Jul
Aug
Sep
Oct
Nov
Dec
Month



```
In [56]: # Displying month wise crime counts over the years
         yearly_monthly_crime_counts
```

```
Out[56]: Month       1        2        3        4        5        6        7        8        9       10       11       12
          Year
          2020 18473.0  17252.0  16154.0  15674.0  17193.0  17023.0  17109.0  16867.0  15628.0  16479.0  15563.0  15933.0
          2021 16525.0  15402.0  16313.0  16045.0  16983.0  17141.0  18653.0  18363.0  18342.0  19285.0  18321.0  17906.0
          2022 18424.0  17685.0  19682.0  19777.0  20398.0  20184.0  19931.0  20060.0  19251.0  20221.0  18646.0  19879.0
          2023 19548.0  18238.0  18961.0  18672.0  18627.0  18360.0  19475.0  19173.0  13479.0      NaN      NaN      NaN
```

```
In [57]: # Displaying first 5 rows od the dataframe
         crime_data1.head()
```

Out[57]:

| | DR_NO | Date Rptd | DATE OCC | TIME OCC | AREA | AREA NAME | Rpt Dist No | Part 1-2 | Crm Cd | Crm Cd Desc | ... | Status Desc | Crm Cd 1 | Crm Cd 2 | Crm Cd 3 | Crm Cd 4 | LOCATION |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **0** | 10304468 | 2020-01-08 | 2020-01-08 | 22:30 | 3 | Southwest | 377 | 2 | 624 | BATTERY - SIMPLE ASSAULT | ... | Adult Other | 624.0 | NaN | NaN | NaN | 1100 W 39TH PL |
| **1** | 190101086 | 2020-01-02 | 2020-01-01 | 03:30 | 1 | Central | 163 | 2 | 624 | BATTERY - SIMPLE ASSAULT | ... | Invest Cont | 624.0 | NaN | NaN | NaN | 700 S HILL ST |
| **2** | 200110444 | 2020-04-14 | 2020-02-13 | 12:00 | 1 | Central | 155 | 2 | 845 | SEX OFFENDER REGISTRANT OUT OF COMPLIANCE | ... | Adult Arrest | 845.0 | NaN | NaN | NaN | 200 E 6TH ST |
| **3** | 191501505 | 2020-01-01 | 2020-01-01 | 17:30 | 15 | N Hollywood | 1543 | 2 | 745 | VANDALISM - MISDEAMEANOR ($399 OR UNDER) | ... | Invest Cont | 745.0 | 998.0 | NaN | NaN | 5400 CORTEEN PL |
| **4** | 191921269 | 2020-01-01 | 2020-01-01 | 04:15 | 19 | Mission | 1998 | 2 | 740 | VANDALISM - FELONY ($400 & OVER, ALL CHURCH VA... | ... | Invest Cont | 740.0 | NaN | NaN | NaN | 14400 TITUS ST |

5 rows × 29 columns

In [58]:

```
# Filtering the dataframe based on specific events
us_election = crime_data1.loc[(crime_data1['DATE OCC'] >= '2020-10-01') & (crime_data1['DATE OCC'] < '2021-02-28'
us_election.head()
```

Out[58]:

|  | DR_NO | Date Rptd | DATE OCC | TIME OCC | AREA | AREA NAME | Rpt Dist No | Part 1-2 | Crm Cd | Crm Cd Desc | ... | Status Desc | Crm Cd 1 | Crm Cd 2 | Crm Cd 3 | Crm Cd 4 | LOCATION | Cross Stree |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **15** | 211916029 | 2021-11-26 | 2020-11-30 | 07:30 | 19 | Mission | 1916 | 2 | 626 | INTIMATE PARTNER - SIMPLE ASSAULT | ... | Invest Cont | 626.0 | NaN | NaN | NaN | 14200 BERG ST |  |
| **16** | 201116159 | 2020-11-29 | 2020-11-28 | 20:18 | 11 | Northeast | 1124 | 2 | 626 | INTIMATE PARTNER - SIMPLE ASSAULT | ... | Adult Other | 626.0 | NaN | NaN | NaN | 3200 W AVENUE 32 |  |
| **18** | 210916801 | 2021-11-22 | 2020-11-19 | 12:00 | 9 | Van Nuys | 932 | 2 | 354 | THEFT OF IDENTITY | ... | Invest Cont | 354.0 | NaN | NaN | NaN | 14700 FRIAR ST |  |
| **23** | 211014725 | 2021-11-18 | 2020-12-28 | 00:01 | 10 | West Valley | 1045 | 2 | 354 | THEFT OF IDENTITY | ... | Invest Cont | 354.0 | NaN | NaN | NaN | 5700 ENFIELD AV |  |
| **25** | 211015072 | 2021-11-29 | 2020-11-01 | 12:00 | 10 | West Valley | 1043 | 2 | 354 | THEFT OF IDENTITY | ... | Invest Cont | 354.0 | NaN | NaN | NaN | 18600 COLLINS ST |  |

5 rows × 29 columns

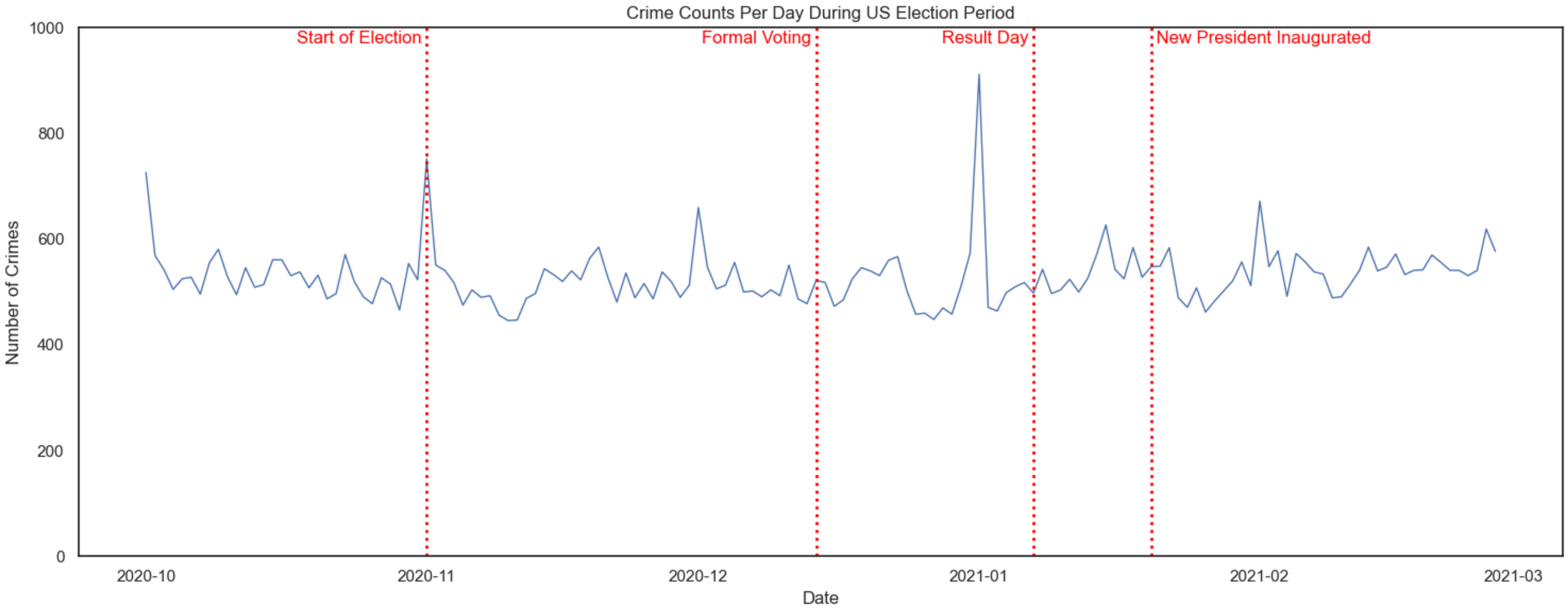
In [59]:

```
# Calculating value counts for the specific events
us_election_count = us_election['DATE OCC'].dt.date.value_counts().sort_index()
us_election_count
```

```
Out[59]: DATE OCC
         2020-10-01    726
         2020-10-02    569
         2020-10-03    542
         2020-10-04    505
         2020-10-05    525
                      ...
         2021-02-23    541
         2021-02-24    531
         2021-02-25    541
         2021-02-26    619
         2021-02-27    577
         Name: count, Length: 150, dtype: int64
```

In [60]:

```
# Plotting the specific events
import matplotlib.pyplot as plt
import pandas as pd

fig = plt.figure(figsize=(15,6))
ax = fig.add_subplot(1, 1, 1)

ax.plot(us_election_count, linestyle='-', linewidth=1)

ax.set_title('Crime Counts Per Day During US Election Period')
ax.set_xlabel('Date')
ax.set_ylabel('Number of Crimes')

ax.set_ylim(bottom=0, top=1000)

election_start = pd.to_datetime('2020-11-01')
formal_voting = pd.to_datetime('2020-12-14')
result_day = pd.to_datetime('2021-01-07')
inauguration_day = pd.to_datetime('2021-01-20')

ax.axvline(x=election_start, color='red', linestyle=':', linewidth=2)
ax.axvline(x=formal_voting, color='red', linestyle=':', linewidth=2)
ax.axvline(x=result_day, color='red', linestyle=':', linewidth=2)
ax.axvline(x=inauguration_day, color='red', linestyle=':', linewidth=2)

ax.annotate('Start of Election ', xy=(election_start, 950), xytext=(election_start, 970),
             ha='right', color='red')

ax.annotate('Formal Voting ', xy=(formal_voting, 950), xytext=(formal_voting, 970),
             ha='right', color='red')

ax.annotate('Result Day ', xy=(result_day, 950), xytext=(result_day, 970),
             ha='right', color='red')

ax.annotate(' New President Inaugurated', xy=(inauguration_day, 950), xytext=(inauguration_day, 970),
             ha='left', color='red')

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



In [61]:

```
# Filtering the dataframe based on specific events
George_Floyd_Protests = crime_data1.loc[(crime_data1['DATE OCC'] >= '2020-04-15') & (crime_data1['DATE OCC'] < '2
George_Floyd_Protests.head()
```

Out[61]:

| | DR_NO | Date Rptd | DATE OCC | TIME OCC | AREA | AREA NAME | Rpt Dist No | Part 1-2 | Crm Cd | Crm Cd Desc | ... | Status Desc | Crm Cd 1 | Crm Cd 2 | Crm Cd 3 | Cr C |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **9** | 201710201 | 2020-06-19 | 2020-05-26 | 19:25 | 17 | Devonshire | 1708 | 1 | 341 | THEFT-GRAND ($950.01 & OVER)EXCPT,GUNS,FOWL,LI... | ... | Adult Other | 341.0 | NaN | NaN | Na |
| **31** | 210816993 | 2021-11-24 | 2020-04-28 | 01:00 | 8 | West LA | 859 | 2 | 354 | THEFT OF IDENTITY | ... | Invest Cont | 354.0 | NaN | NaN | Na |
| **32** | 201111990 | 2020-07-29 | 2020-07-29 | 20:20 | 11 | Northeast | 1124 | 2 | 626 | INTIMATE PARTNER - SIMPLE ASSAULT | ... | Adult Other | 626.0 | NaN | NaN | Na |
| **39** | 200810920 | 2020-06-20 | 2020-06-20 | 10:00 | 8 | West LA | 841 | 2 | 624 | BATTERY - SIMPLE ASSAULT | ... | Adult Other | 624.0 | NaN | NaN | Na |
| **52** | 201112187 | 2020-08-04 | 2020-07-16 | 19:00 | 11 | Northeast | 1144 | 2 | 662 | BUNCO, GRAND THEFT | ... | Invest Cont | 662.0 | NaN | NaN | Na |

5 rows × 29 columns

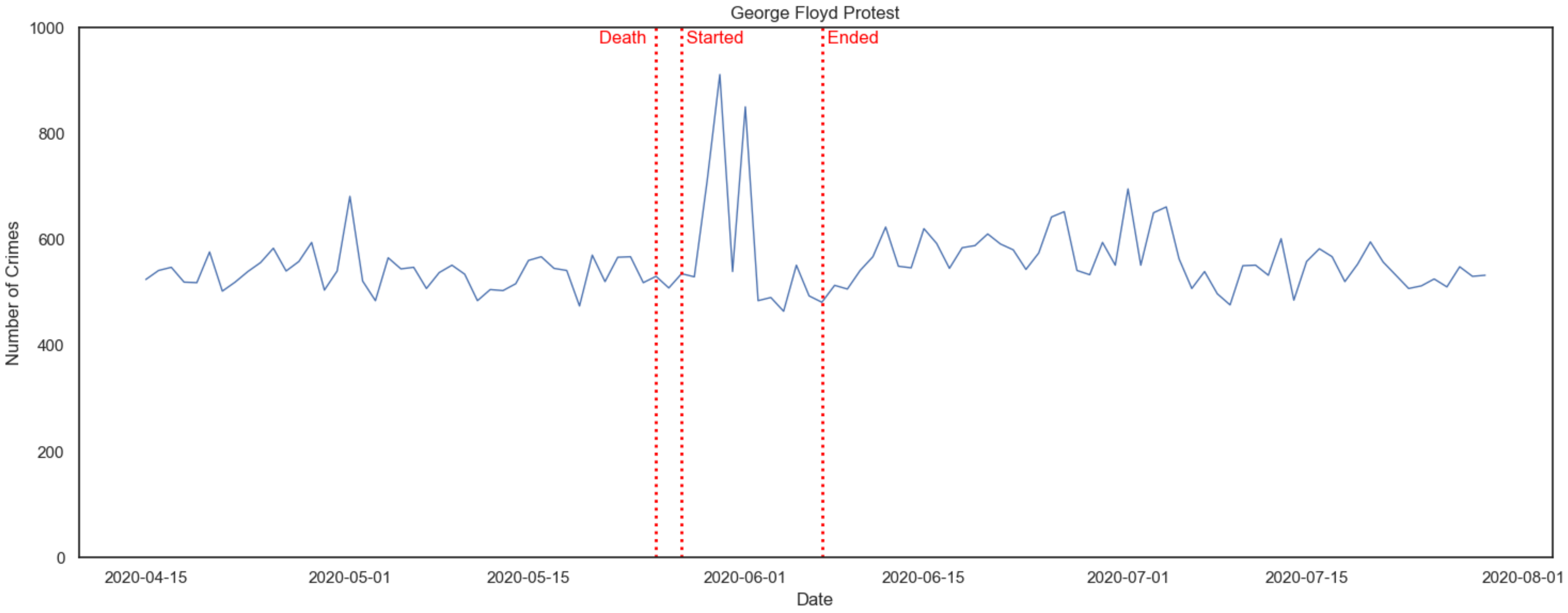
In [62]:

```
# Calculating value counts for the specific events
George_Floyd_Protests_count = George_Floyd_Protests['DATE OCC'].dt.date.value_counts().sort_index()
George_Floyd_Protests_count
```

```
Out[62]: DATE OCC
         2020-04-15    525
         2020-04-16    542
         2020-04-17    548
         2020-04-18    520
         2020-04-19    519
                      ...
         2020-07-25    526
         2020-07-26    511
         2020-07-27    549
         2020-07-28    531
         2020-07-29    533
         Name: count, Length: 106, dtype: int64
```

In [63]:

```python
# Plotting the specific events
import matplotlib.pyplot as plt
import pandas as pd

fig = plt.figure(figsize=(15,6))
ax = fig.add_subplot(1, 1, 1)

ax.plot(George_Floyd_Protests_count, linestyle='-', linewidth=1)

ax.set_title('George Floyd Protest')
ax.set_xlabel('Date')
ax.set_ylabel('Number of Crimes')

ax.set_ylim(bottom=0, top=1000)

Death = pd.to_datetime('2020-05-25')
Protests_started = pd.to_datetime('2020-05-27')
Protests_ended= pd.to_datetime('2020-06-07')


ax.axvline(x=Death, color='red', linestyle=':', linewidth=2)
ax.axvline(x=Protests_started, color='red', linestyle=':', linewidth=2)
ax.axvline(x=Protests_ended, color='red', linestyle=':', linewidth=2)

ax.annotate('Death  ', xy=(Death, 950), xytext=(Death, 970),
             ha='right', color='red')

ax.annotate(' Started ', xy=(Protests_started, 950), xytext=(Protests_started, 970),
             ha='left', color='red')

ax.annotate(' Ended ', xy=(Protests_ended, 950), xytext=(Protests_ended, 970),
             ha='left', color='red')

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



In [64]:

```
# Filtering the dataframe based on specific events
Min_wages = crime_data1.loc[(crime_data1['DATE OCC'] >= '2022-04-01') & (crime_data1['DATE OCC'] < '2022-07-30')]
Min_wages.head()
```

Out[64]:

| | DR_NO | Date Rptd | DATE OCC | TIME OCC | AREA | AREA NAME | Rpt Dist No | Part 1-2 | Crm Cd | Crm Cd Desc | ... | Status Desc | Crm Cd 1 | Crm Cd 2 | Crm Cd 3 | Crm Cd 4 | LOCAT |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **408700** | 221809626 | 2022-04-23 | 2022-04-22 | 21:00 | 18 | Southeast | 1862 | 1 | 510 | VEHICLE - STOLEN | ... | Invest Cont | 510.0 | NaN | NaN | NaN | SAN PE<br>ST 120 |
| **408701** | 221711121 | 2022-07-10 | 2022-07-10 | 02:00 | 17 | Devonshire | 1781 | 1 | 230 | ASSAULT WITH DEADLY WEAPON, AGGRAVATED ASSAULT | ... | Invest Cont | 230.0 | NaN | NaN | NaN | 8800 |
| **408706** | 221508072 | 2022-04-10 | 2022-04-10 | 06:50 | 15 | N Hollywood | 1539 | 2 | 740 | VANDALISM - FELONY ($400 & OVER, ALL CHURCH VA... | ... | Invest Cont | 740.0 | NaN | NaN | NaN | 1<br>BURE |
| **408711** | 220613315 | 2022-07-17 | 2022-07-17 | 14:15 | 6 | Hollywood | 668 | 2 | 624 | BATTERY - SIMPLE ASSAULT | ... | Invest Cont | 624.0 | NaN | NaN | NaN | 5<br>FERNW |
| **408712** | 221909398 | 2022-05-12 | 2022-05-10 | 19:30 | 19 | Mission | 1961 | 2 | 626 | INTIMATE PARTNER - SIMPLE ASSAULT | ... | Adult Other | 626.0 | NaN | NaN | NaN | 91<br>SEPULV |

5 rows × 29 columns

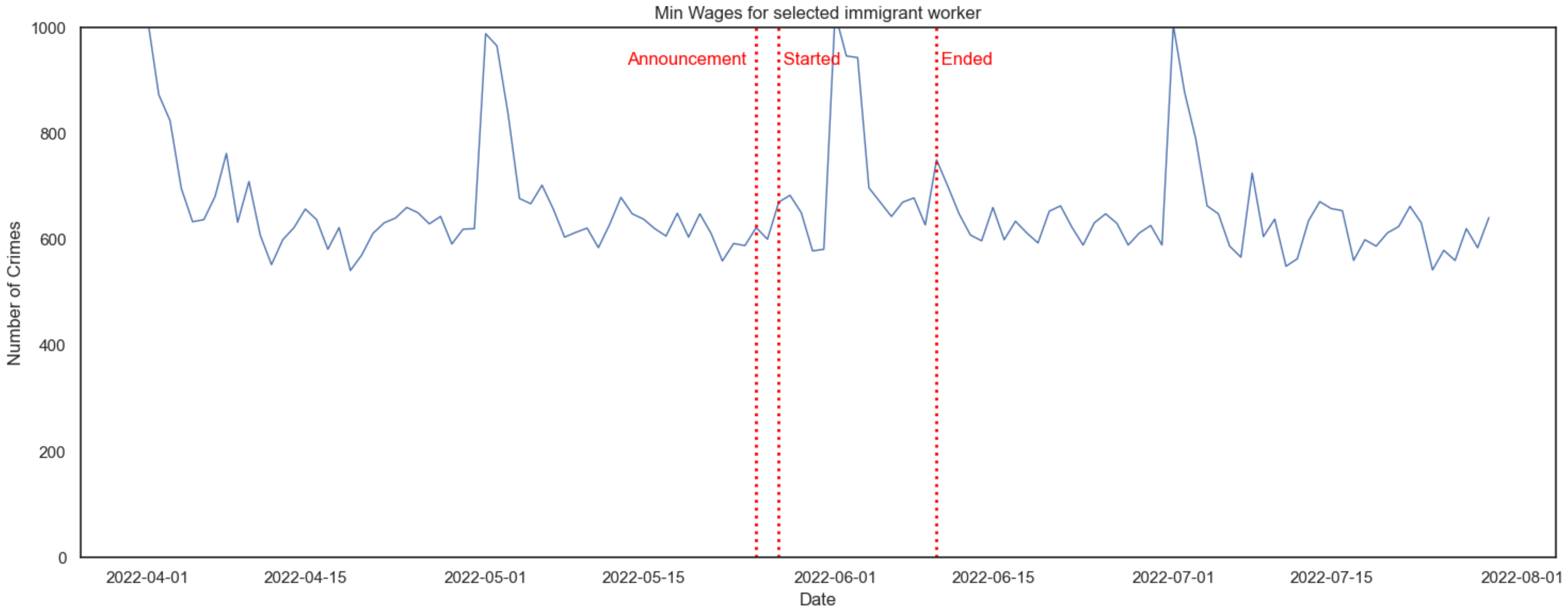
In [65]:

```
# Calculating value counts for the specific events
Min_wages_count = Min_wages['DATE OCC'].dt.date.value_counts().sort_index()
Min_wages_count
```

```
Out[65]: DATE OCC
         2022-04-01    1018
         2022-04-02     874
         2022-04-03     825
         2022-04-04     697
         2022-04-05     634
                       ...
         2022-07-25     580
         2022-07-26     561
         2022-07-27     621
         2022-07-28     585
         2022-07-29     642
         Name: count, Length: 120, dtype: int64
```

In [66]:

```python
# Plotting the specific events
import matplotlib.pyplot as plt
import pandas as pd

fig = plt.figure(figsize=(15,6))
ax = fig.add_subplot(1, 1, 1)

ax.plot(Min_wages_count, linestyle='-', linewidth=1)

ax.set_title('Min Wages for selected immigrant worker')
ax.set_xlabel('Date')
ax.set_ylabel('Number of Crimes')

ax.set_ylim(bottom=0, top=1000)

Announcement = pd.to_datetime('2022-05-25')
Protests_started = pd.to_datetime('2022-05-27')
Protests_ended= pd.to_datetime('2022-06-10')


ax.axvline(x=Announcement, color='red', linestyle=':', linewidth=2)
ax.axvline(x=Protests_started, color='red', linestyle=':', linewidth=2)
ax.axvline(x=Protests_ended, color='red', linestyle=':', linewidth=2)

ax.annotate('Announcement  ', xy=(Announcement, 300), xytext=(Announcement, 930),
             ha='right', color='red')

ax.annotate(' Started ', xy=(Protests_started, 930), xytext=(Protests_started, 930),
             ha='left', color='red')

ax.annotate(' Ended ', xy=(Protests_ended, 930), xytext=(Protests_ended, 930),
             ha='left', color='red')

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



In [67]:

```
# Filtering the dataframe based on specific events
Covid = crime_data1.loc[(crime_data1['DATE OCC'] >= '2020-01-01') & (crime_data1['DATE OCC'] < '2022-06-01')]
Covid.head()
```

Out[67]:

| | DR_NO | Date Rptd | DATE OCC | TIME OCC | AREA | AREA NAME | Rpt Dist No | Part 1-2 | Crm Cd | Crm Cd Desc | ... | Status Desc | Crm Cd 1 | Crm Cd 2 | Crm Cd 3 | Crm Cd 4 | LOCATION |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **0** | 10304468 | 2020-01-08 | 2020-01-08 | 22:30 | 3 | Southwest | 377 | 2 | 624 | BATTERY - SIMPLE ASSAULT | ... | Adult Other | 624.0 | NaN | NaN | NaN | 1100 W 39TH PL |
| **1** | 190101086 | 2020-01-02 | 2020-01-01 | 03:30 | 1 | Central | 163 | 2 | 624 | BATTERY - SIMPLE ASSAULT | ... | Invest Cont | 624.0 | NaN | NaN | NaN | 700 S HILL ST |
| **2** | 200110444 | 2020-04-14 | 2020-02-13 | 12:00 | 1 | Central | 155 | 2 | 845 | SEX OFFENDER REGISTRANT OUT OF COMPLIANCE | ... | Adult Arrest | 845.0 | NaN | NaN | NaN | 200 E 6TH ST |
| **3** | 191501505 | 2020-01-01 | 2020-01-01 | 17:30 | 15 | N Hollywood | 1543 | 2 | 745 | VANDALISM - MISDEAMEANOR ($399 OR UNDER) | ... | Invest Cont | 745.0 | 998.0 | NaN | NaN | 5400 CORTEEN PL |
| **4** | 191921269 | 2020-01-01 | 2020-01-01 | 04:15 | 19 | Mission | 1998 | 2 | 740 | VANDALISM - FELONY ($400 & OVER, ALL CHURCH VA... | ... | Invest Cont | 740.0 | NaN | NaN | NaN | 14400 TITUS ST |

5 rows × 29 columns

In [68]:

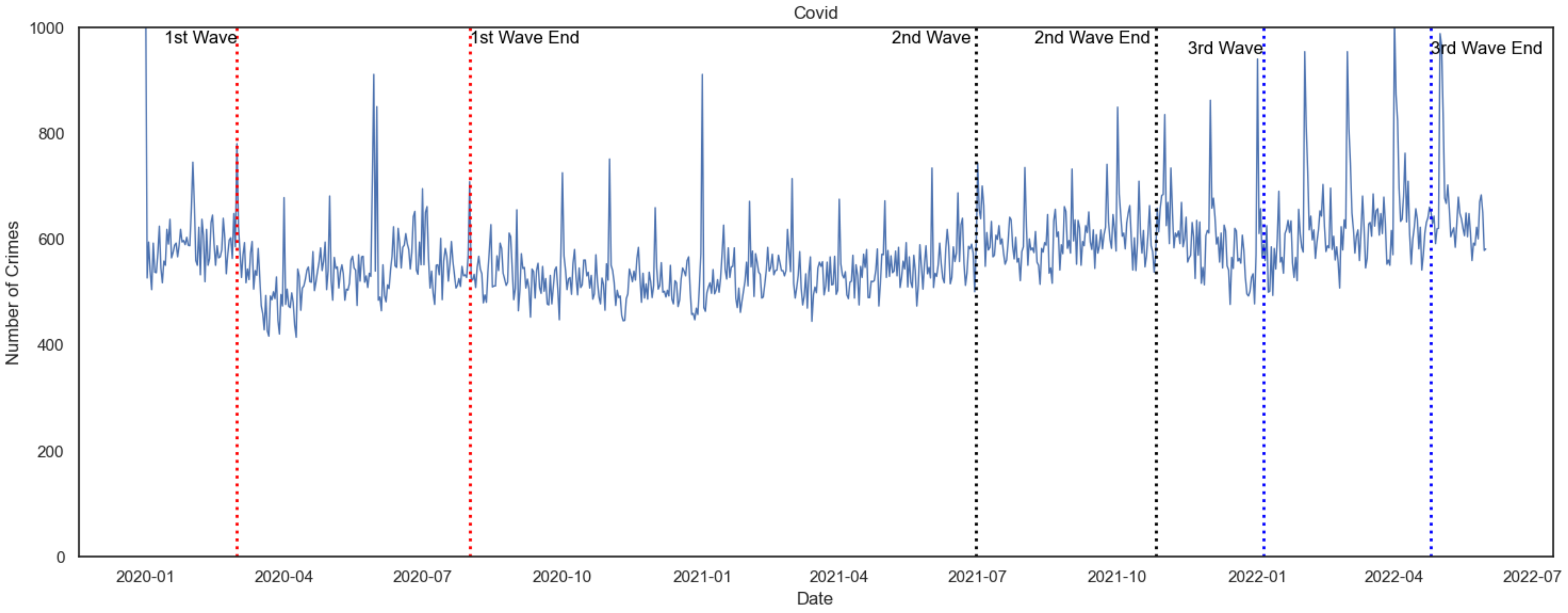
```
# Calculating value counts for the specific events
Covid_count = Covid['DATE OCC'].dt.date.value_counts().sort_index()
Covid_count
```

Out[68]:

```
DATE OCC
2020-01-01    1103
2020-01-02     527
2020-01-03     595
2020-01-04     540
2020-01-05     505
              ...
2022-05-27     671
2022-05-28     684
2022-05-29     651
2022-05-30     579
2022-05-31     582
Name: count, Length: 882, dtype: int64
```

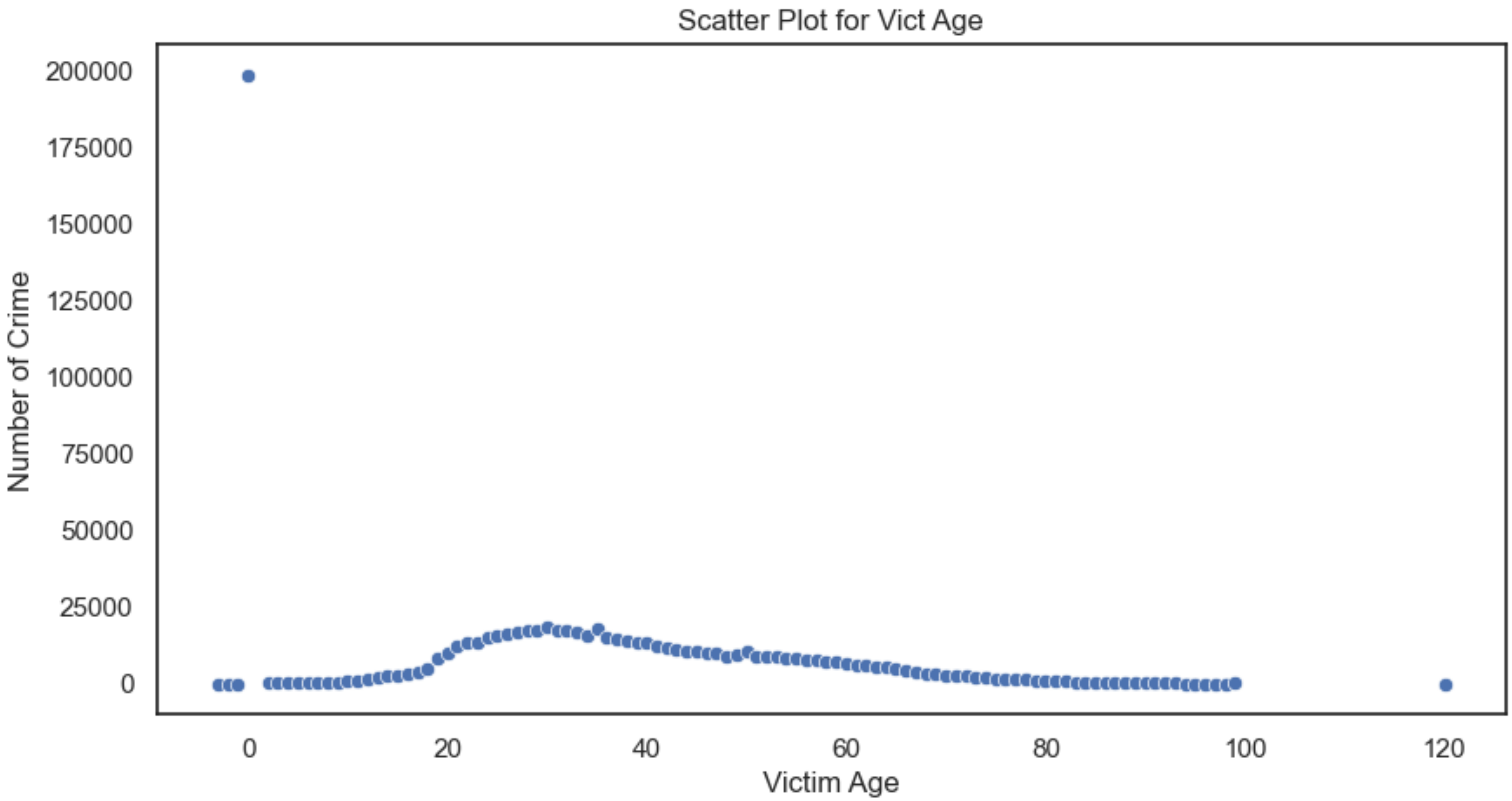
In [69]:

```
# Plotting the specific events
import matplotlib.pyplot as plt
import pandas as pd

fig = plt.figure(figsize=(15,6))
ax = fig.add_subplot(1, 1, 1)

ax.plot(Covid_count, linestyle='-', linewidth=1)

ax.set_title('Covid')
ax.set_xlabel('Date')
ax.set_ylabel('Number of Crimes')

ax.set_ylim(bottom=0, top=1000)

First_wave = pd.to_datetime('2020-03-01')
FirstWave_ended = pd.to_datetime('2020-08-01')
Second_wave= pd.to_datetime('2021-06-30')
Secondwave_ended= pd.to_datetime('2021-10-26')
Third_wave= pd.to_datetime('2022-01-05')
Thirdwave_ended= pd.to_datetime('2022-04-25')


ax.axvline(x=First_wave, color='red', linestyle=':', linewidth=2)
ax.axvline(x=FirstWave_ended, color='red', linestyle=':', linewidth=2)
ax.axvline(x=Second_wave, color='black', linestyle=':', linewidth=2)
ax.axvline(x=Secondwave_ended, color='black', linestyle=':', linewidth=2)
ax.axvline(x=Third_wave, color='blue', linestyle=':', linewidth=2)
ax.axvline(x=Thirdwave_ended, color='blue', linestyle=':', linewidth=2)

ax.annotate('1st Wave', xy=(First_wave, 950), xytext=(First_wave, 970),
             ha='right', color='black')

ax.annotate('1st Wave End', xy=(FirstWave_ended, 950), xytext=(FirstWave_ended, 970),
             ha='left', color='black')

ax.annotate('2nd Wave ', xy=(Second_wave, 950), xytext=(Second_wave, 970),
             ha='right', color='black')
ax.annotate('2nd Wave End ', xy=(Secondwave_ended, 950), xytext=(Secondwave_ended, 970),
             ha='right', color='black')

ax.annotate('3rd Wave', xy=(Third_wave, 950), xytext=(Third_wave, 950),
             ha='right', color='black')

ax.annotate('3rd Wave End', xy=(Thirdwave_ended, 950), xytext=(Thirdwave_ended, 950),
             ha='left', color='black')

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



# Outliers and Anomalies

In [70]:
```
#Scatter Plot for Vict Age
age_wise = data_crime.groupby('Vict Age')['Crm Cd'].count()

plt.figure(figsize=(10, 5))
sns.scatterplot(x=age_wise.index, y=age_wise.values)
plt.title('Scatter Plot for Vict Age')
plt.xlabel('Victim Age')
plt.ylabel('Number of Crime')
plt.show()
```



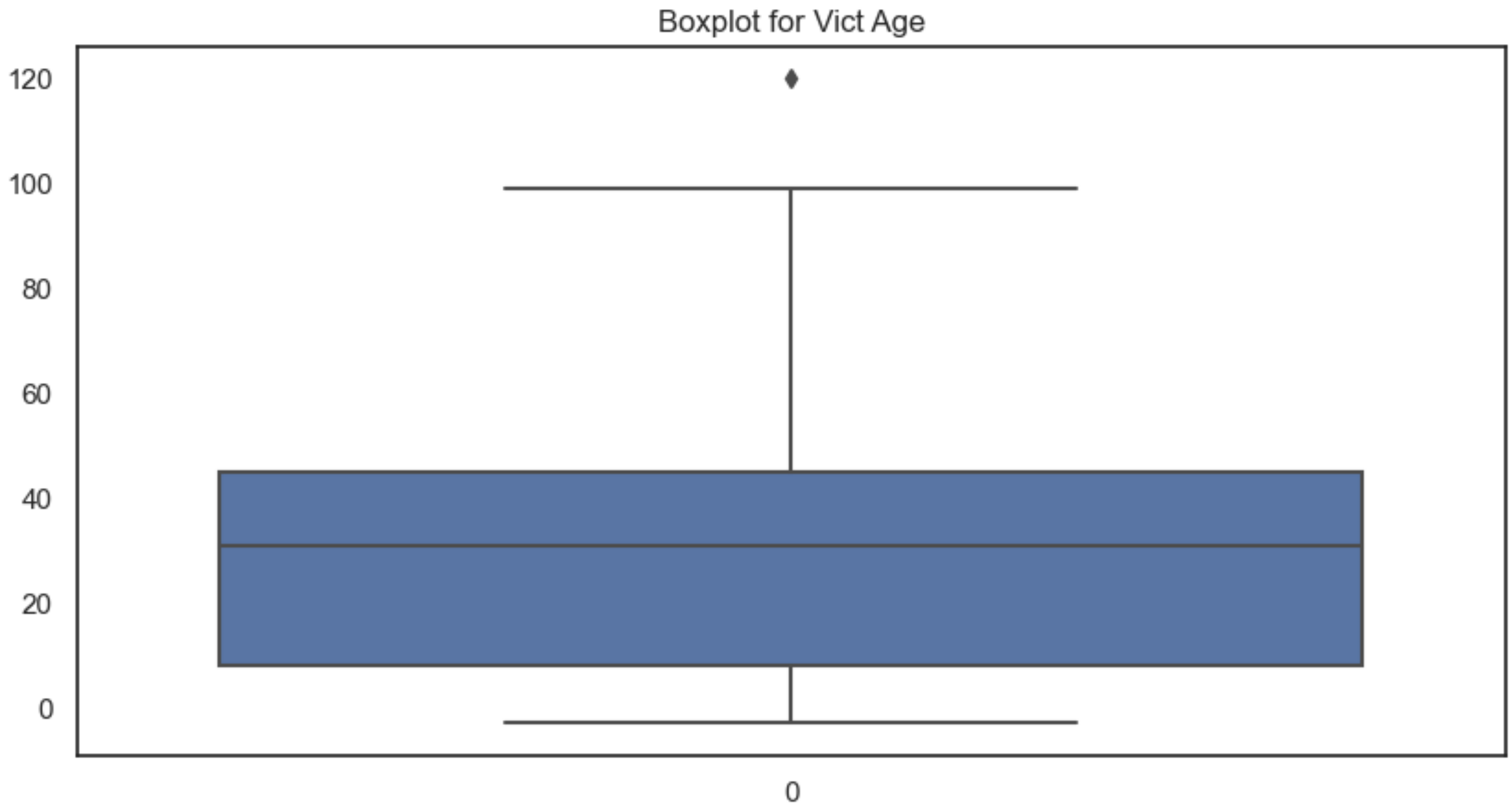
```
In [71]: age_wise
```

Out[71]:

```
Vict Age
-3          1
-2         12
-1         55
 0     198667
 2        359
        ...  
 96        84
 97        60
 98        64
 99       304
 120        1
Name: Crm Cd, Length: 103, dtype: int64
```

In [72]:

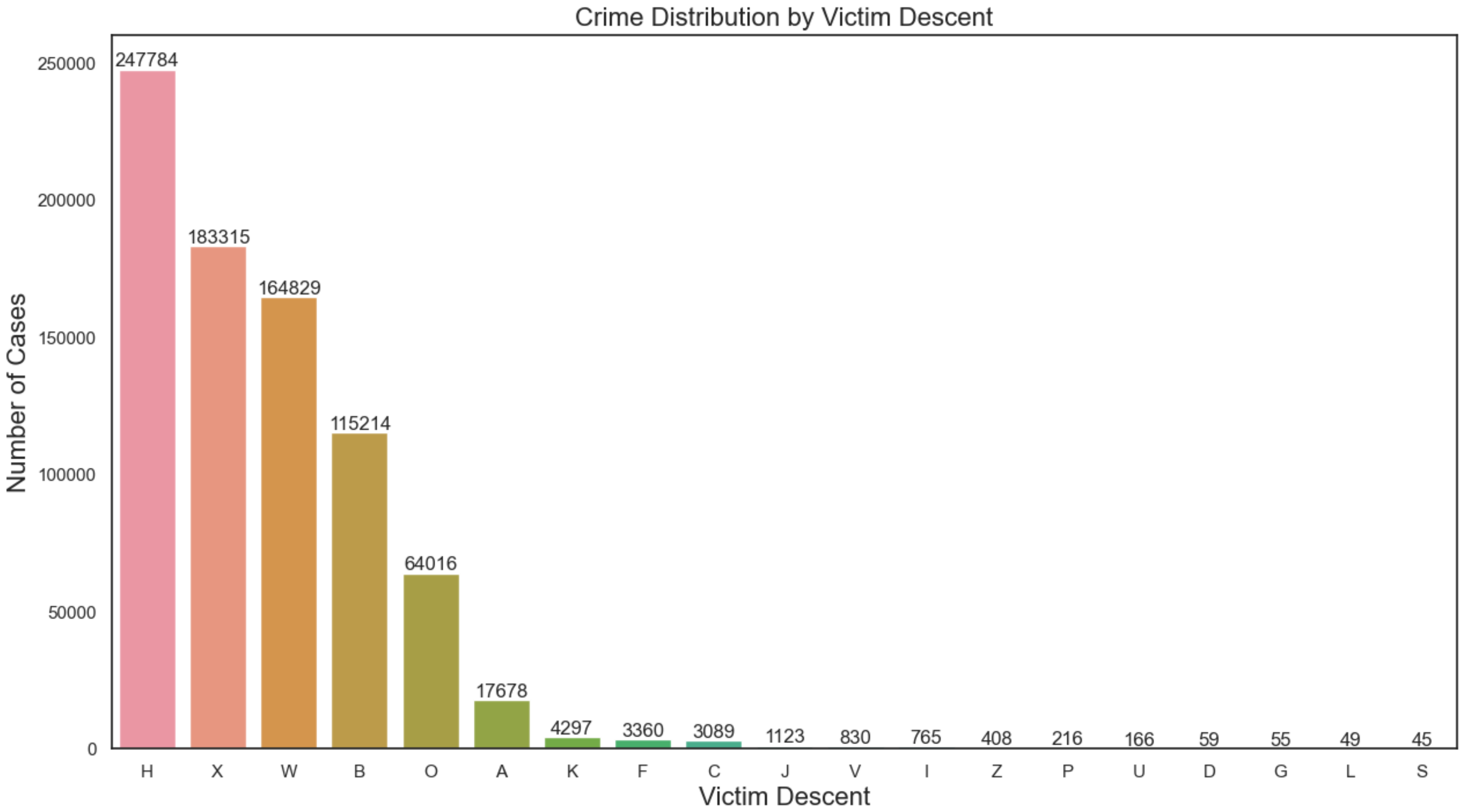
```
#Boxplot for Vict Age
plt.figure(figsize=(10, 5))
sns.boxplot(data_crime['Vict Age'])
plt.title('Boxplot for Vict Age')
plt.show()
```



## Demographic Factors

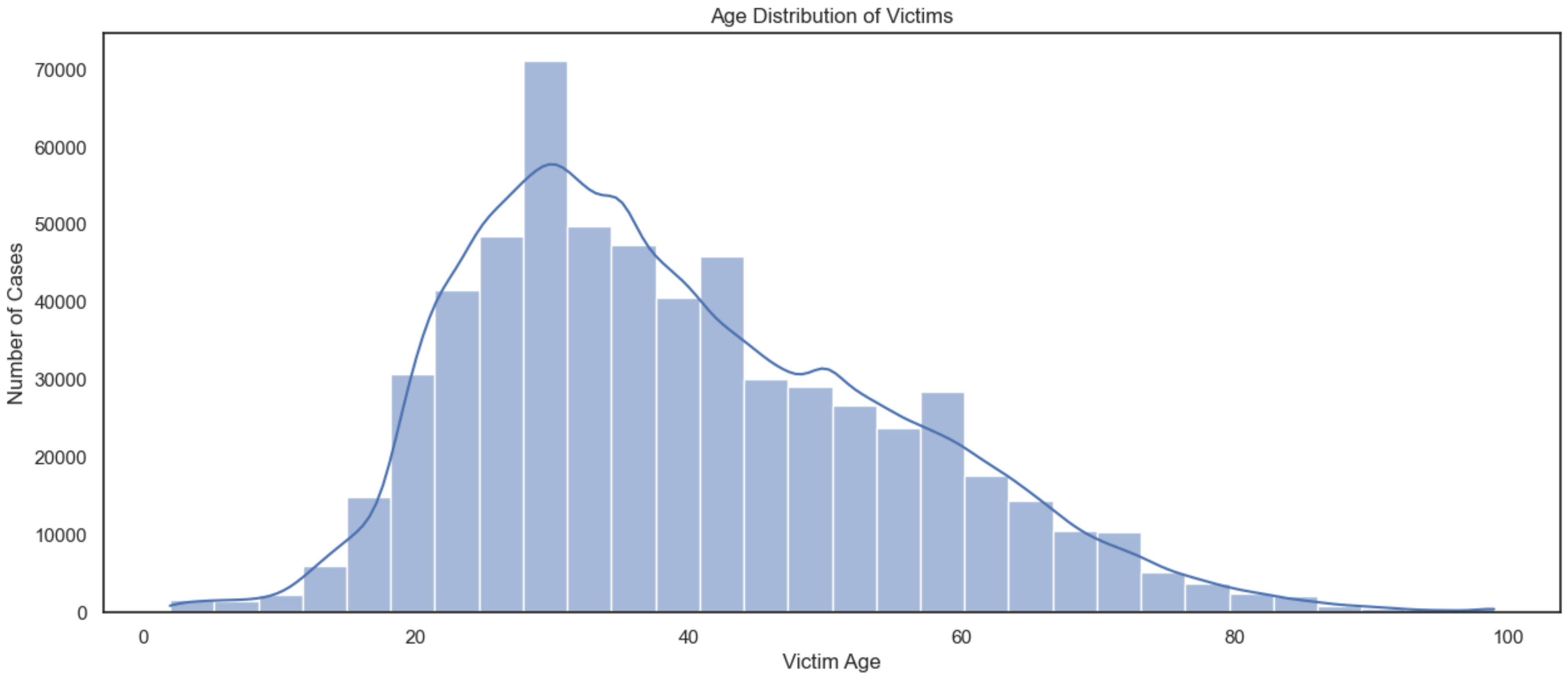
In [73]:

```
# Plotting Crime distribution by Victim Descent
plt.figure(figsize=(15, 8))
ax = sns.countplot(x='Vict Descent', data=crime_data, order=crime_data['Vict Descent'].value_counts().index)
plt.title('Crime Distribution by Victim Descent', fontsize = 16)
plt.ylabel('Number of Cases', fontsize = 16)
plt.xlabel('Victim Descent',fontsize = 16)

for p in ax.patches:
    ax.annotate(f'{int(p.get_height())}', (p.get_x() + p.get_width() / 2., p.get_height()), ha='center', va='cent

plt.show()
```



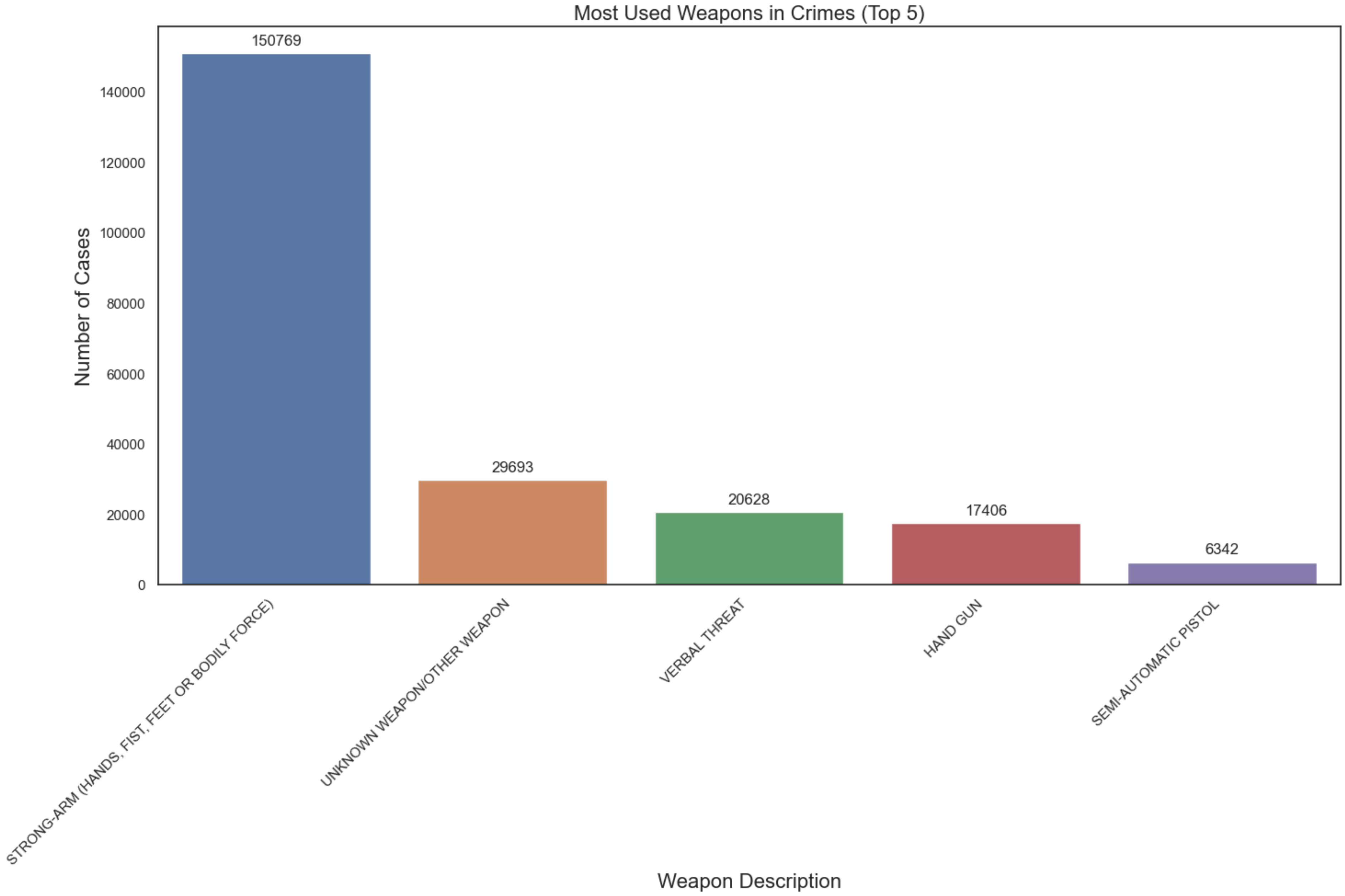
In [74]:

```
# Plotting age distribution of Victims
vict_age_data = crime_data[crime_data['Vict Age'].between(1,100)]

plt.figure(figsize=(15, 6))
ax = sns.histplot(vict_age_data['Vict Age'], bins=30, kde=True)
plt.title('Age Distribution of Victims')
plt.ylabel('Number of Cases')
plt.xlabel('Victim Age')
plt.show()
```



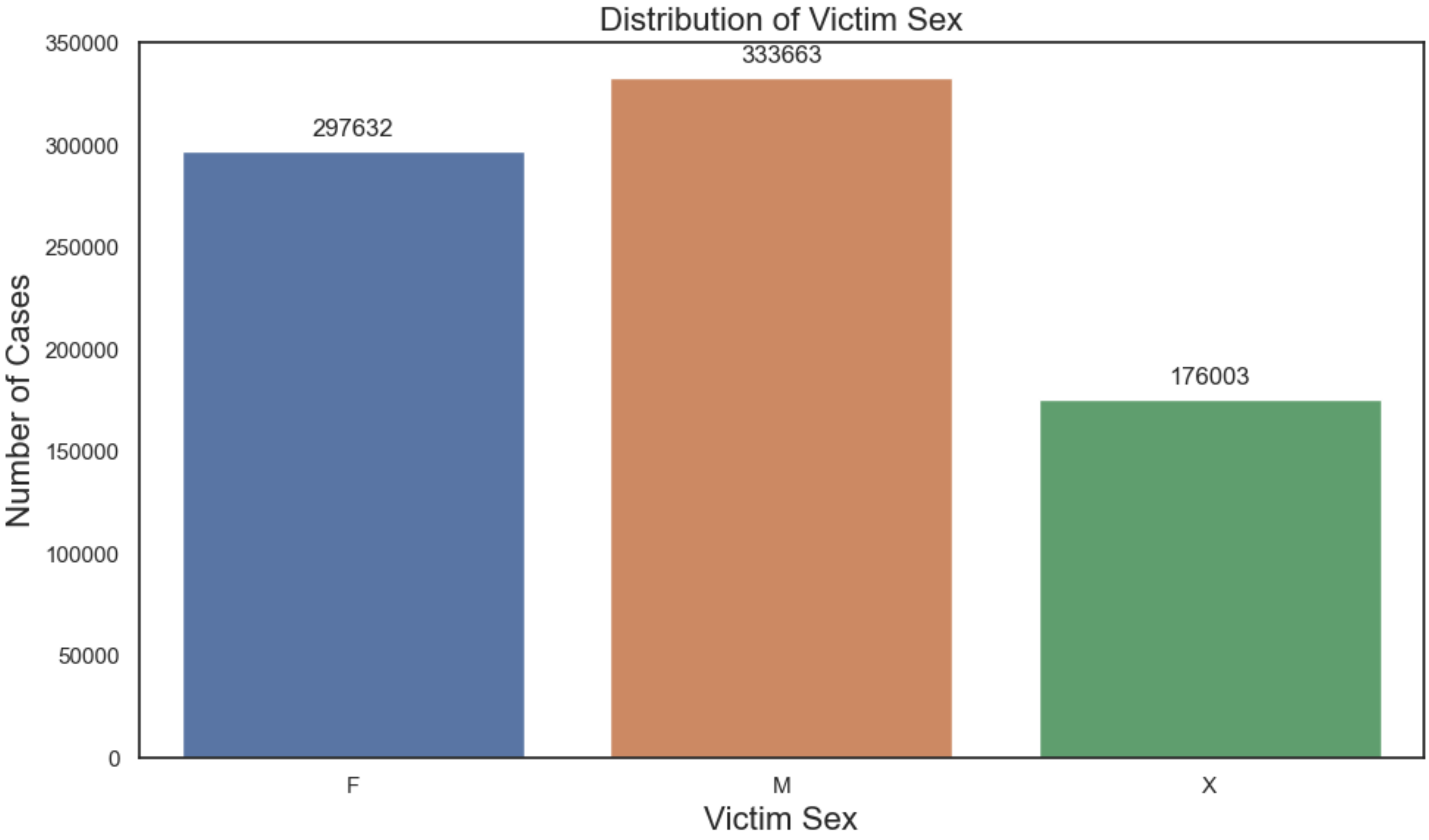
In [75]:

```
# Plotting most used Weapons in Crimes
plt.figure(figsize=(15, 10))
top_5_weapons = data_crime['Weapon Desc'].value_counts().index[:5]
ax = sns.countplot(x='Weapon Desc', data=data_crime, order=top_5_weapons)
plt.title('Most Used Weapons in Crimes (Top 5)', fontsize = 16)
plt.xlabel('Weapon Description', fontsize = 16)
plt.ylabel('Number of Cases', fontsize = 16)

plt.xticks(rotation=45, ha='right')

for p in ax.patches:
    ax.annotate(f'{int(p.get_height())}', (p.get_x() + p.get_width() / 2., p.get_height()), ha='center', va='cent

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


Most Used Weapons in Crimes (Top 5)
Number of Cases
0
20000
40000
60000
80000
100000
120000
140000
150769
29693
20628
17406
6342
STRONG-ARM (HANDS, FIST, FEET OR BODILY FORCE)
UNKNOWN WEAPON/OTHER WEAPON
VERBAL THREAT
HAND GUN
SEMI-AUTOMATIC PISTOL
Weapon Description

In [76]:

```
# Plotting Distribution of Victim Sex
plt.figure(figsize=(10, 6))
ax = sns.countplot(x='Vict Sex', data =crime_data)
plt.title('Distribution of Victim Sex',fontsize = 16)
plt.xlabel('Victim Sex',fontsize = 16)
plt.ylabel('Number of Cases',fontsize = 16)

for p in ax.patches:
    ax.annotate(f'{int(p.get_height())}', (p.get_x() + p.get_width() / 2., p.get_height()), ha='center', va='cent

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



## Advanced Analysis

In [77]:

```python
# Creating a copy of the crime data
df = crime_data.copy()
```

In [78]:

```python
# Converting month-year column into the month year format
df['month-year'] = df['Date Time OCC'].dt.strftime('%m-%Y')
df.head(3)
```

Out[78]:

| | DR_NO | Date Rptd | Date Time OCC | AREA NAME | Rpt Dist No | Crm Cd | Crm Cd Desc | Vict Age | Vict Sex | Vict Descent | ... | Crm Cd 1 | LOCATION | LAT | LON | Wea U |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **0** | 10304468 | 2020-01-08 | 2020-01-08 22:30:00 | Southwest | 377 | 624 | BATTERY - SIMPLE ASSAULT | 36 | F | B | ... | 624.0 | 1100 W 39TH PL | 34.0141 | -118.2978 | |
| **1** | 190101086 | 2020-01-02 | 2020-01-01 03:30:00 | Central | 163 | 624 | BATTERY - SIMPLE ASSAULT | 25 | M | H | ... | 624.0 | 700 S HILL ST | 34.0459 | -118.2545 | |
| **2** | 200110444 | 2020-04-14 | 2020-02-13 12:00:00 | Central | 155 | 845 | SEX OFFENDER REGISTRANT OUT OF COMPLIANCE | 0 | X | X | ... | 845.0 | 200 E 6TH ST | 34.0448 | -118.2474 | |

3 rows × 24 columns

In [79]:

```python
# Grouping the crimes by month and year
df = df.groupby('month-year')['Crm Cd'].count()
```

In [80]:

```python
# renaming columns and converting to appropriate data types
df_1 = df.to_frame().reset_index()
df_1 = df_1.rename(columns={'month-year':'Date','Crm Cd':'Crime'})
df_1['Date'] = pd.to_datetime(df_1['Date'])

# adding new column for number of days in month
df_1['days_in_month'] = df_1['Date'].dt.days_in_month
df_1.head()
```

```
/var/folders/wj/lvd1hjrx0vqf2hzhdxhvbz580000gn/T/ipykernel_53142/790939933.py:4: UserWarning: Could not infer fo
rmat, so each element will be parsed individually, falling back to `dateutil`. To ensure parsing is consistent a
nd as-expected, please specify a format.
  df_1['Date'] = pd.to_datetime(df_1['Date'])
```

Out[80]:

| | Date | Crime | days_in_month |
|---|---|---|---|
| **0** | 2020-01-01 | 18473 | 31 |
| **1** | 2021-01-01 | 16525 | 31 |
| **2** | 2022-01-01 | 18424 | 31 |
| **3** | 2023-01-01 | 19548 | 31 |
| **4** | 2020-02-01 | 17252 | 29 |

In [81]:

```python
# Normalising the number of crime by dividing it by number of days in respective months
df_1['Crime'] = round((df_1['Crime']/df_1['days_in_month']))
df_1.drop(columns=['days_in_month'],inplace=True)
df_1.head()
```

Out[81]:

| | Date | Crime |
|---|---|---|
| **0** | 2020-01-01 | 596.0 |
| **1** | 2021-01-01 | 533.0 |
| **2** | 2022-01-01 | 594.0 |
| **3** | 2023-01-01 | 631.0 |
| **4** | 2020-02-01 | 595.0 |

In [82]:

```python
# Sorting values of the dataframe based on date
df_1 = df_1.sort_values(by='Date').reset_index(drop=True)
```

In [83]:

```python
# reseting the Index of the Dataframe
df_1.set_index('Date',inplace=True)
```

In [84]:

```
df_1 = df_1[:-1]
df_1.shape
```

Out[84]:

```
(44, 1)
```

In [85]:

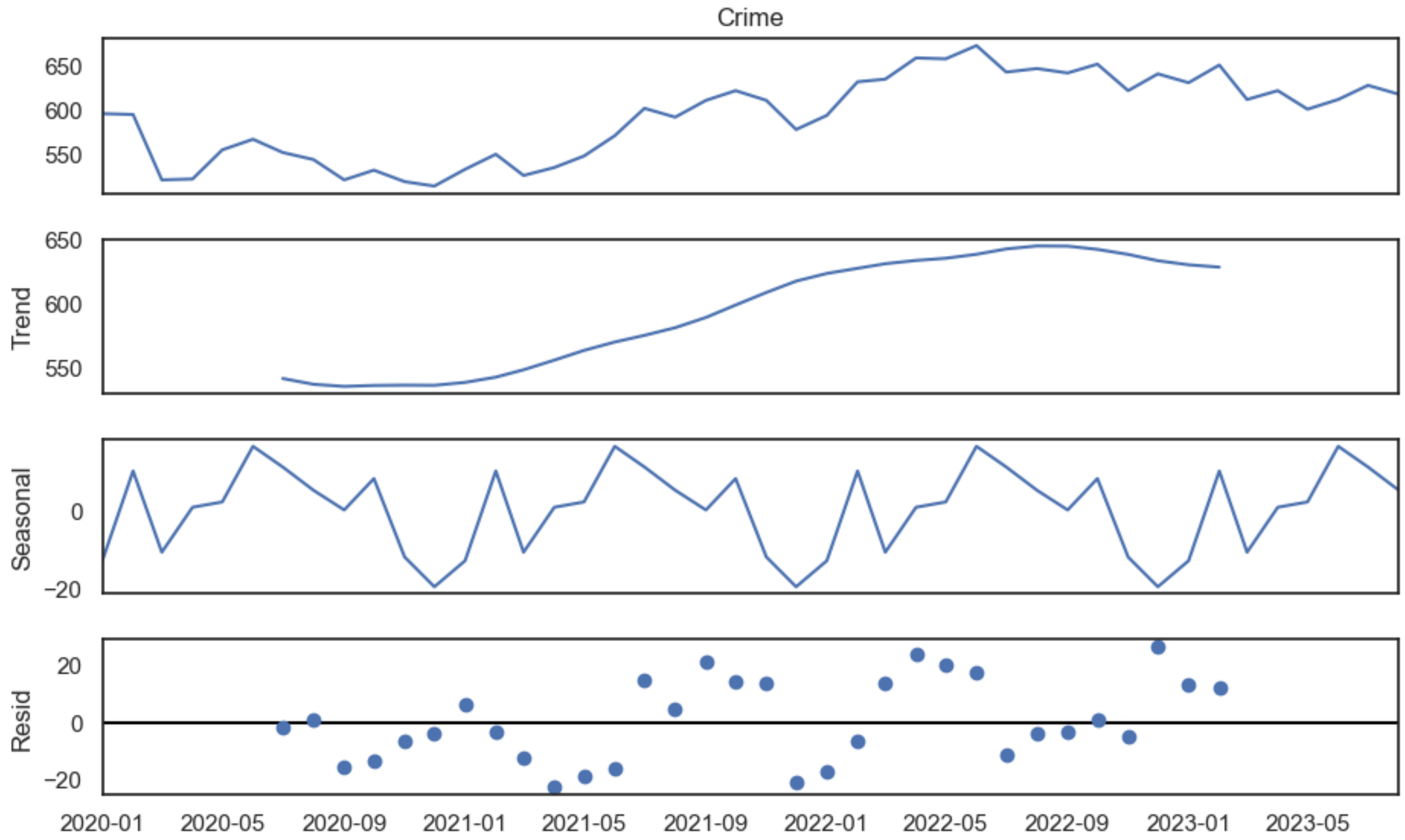
```
# Importing stats libraries
from statsmodels.tsa.seasonal import seasonal_decompose
from statsmodels.tsa.stattools import adfuller
from statsmodels.graphics.tsaplots import plot_acf,plot_pacf
from statsmodels.tsa.arima_model import ARIMA
from statsmodels.tsa.statespace.sarimax import SARIMAX
import warnings
warnings.filterwarnings('ignore')
```

In [86]:

```
# decomposing the data to analyse the trends, seasonality and residuals if any
result = seasonal_decompose(df_1['Crime'],model = 'additive',period = 12)
```

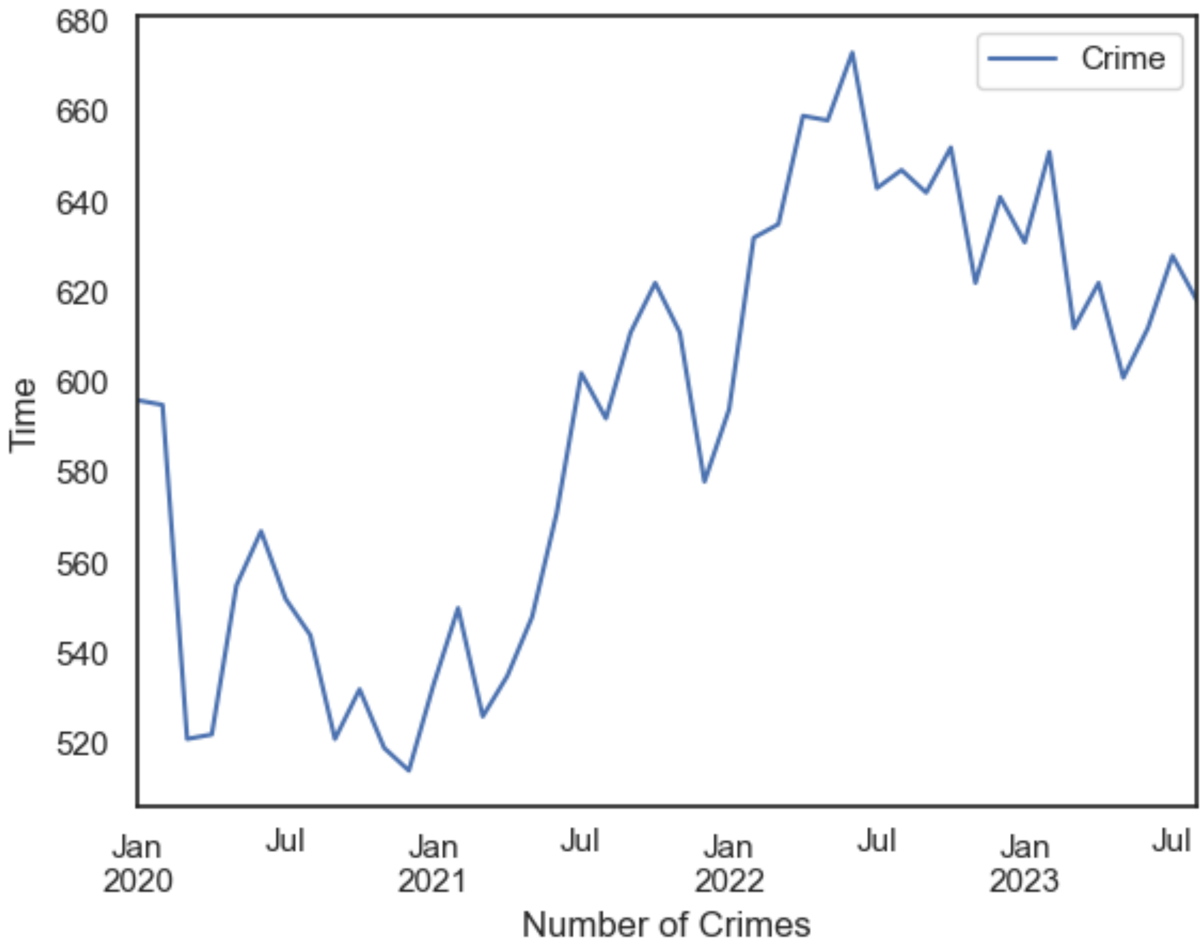
In [87]:

```
# visualising decomposition
ax = result.plot()
ax.set_size_inches(10,6)
```



1) The first subplot on the very top shows the plot for the original data with no decomposition. 2) The second subplot shows a clear smooth trend pattern in the data. This is clear evidence of a non-constant mean. 3) The third subplot shows the decomposed seasonality pattern in the data. 4) The last subplot shows the noise or residual component in the time series data

```
In [88]: # Fluctuations in number of crime over time
         df_1.plot()
         plt.xlabel('Number of Crimes')
         plt.ylabel('Time')
         plt.show()
```



From the figures above we can observe that the data violates the stationarity assumption of constant mean, constant variance and constant covariance.

In [89]:

```
# Implementing the Augmented Dicky Fuller test to test the stationarity of data
ADF = adfuller(df_1['Crime'])
```

In [90]:

```
def ADF_CHECK(time_series):
    result = adfuller(time_series)
    print("Augmented Dicky-Fuller Test")
    labels = ["ADF Test statistic","p-value","# of lags","Num of observations used"]

    for val,lab in zip(result,labels):
        print(lab + ":" + str(val))

    alpha = 0.05
    if result[1]<=alpha:
        print("Reject null hypothesis")
        print("Data does not have a unit root and is stationery")
    else:
        print("Fail to reject null hypothesis")
        print("Data has a unit root and it is non-stationery")
```

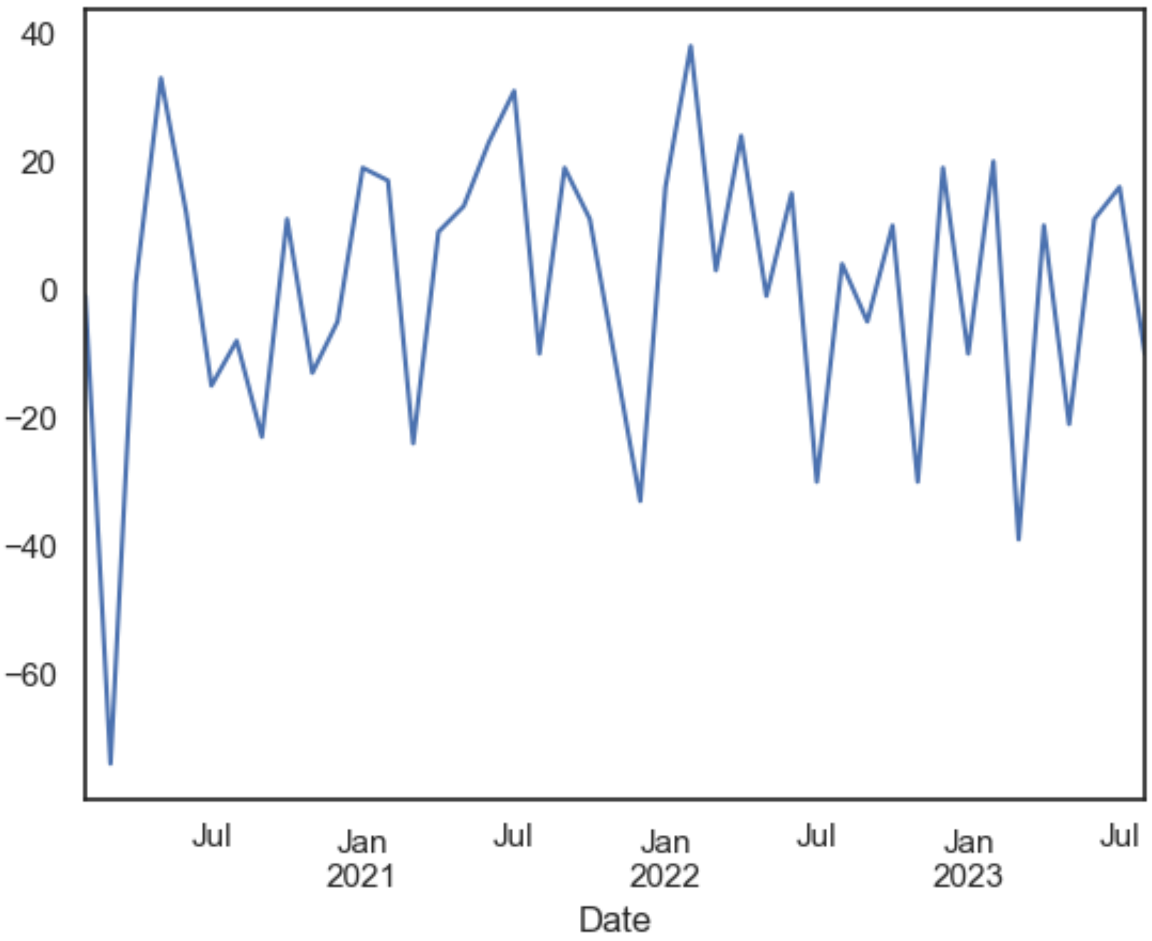
In [91]:

```
ADF_CHECK(df_1['Crime'])
```

```
Augmented Dicky-Fuller Test
ADF Test statistic:-1.4768186370337697
p-value:0.5449460798531898
# of lags:0
Num of observations used:43
Fail to reject null hypothesis
Data has a unit root and it is non-stationery
```

In [92]:

```
# difference the data and check for stationarity again
df_1["Difference_1"] = df_1['Crime']-df_1['Crime'].shift(1)
df_1.dropna(inplace=True)
df_1["Difference_1"].plot()
plt.show();
```



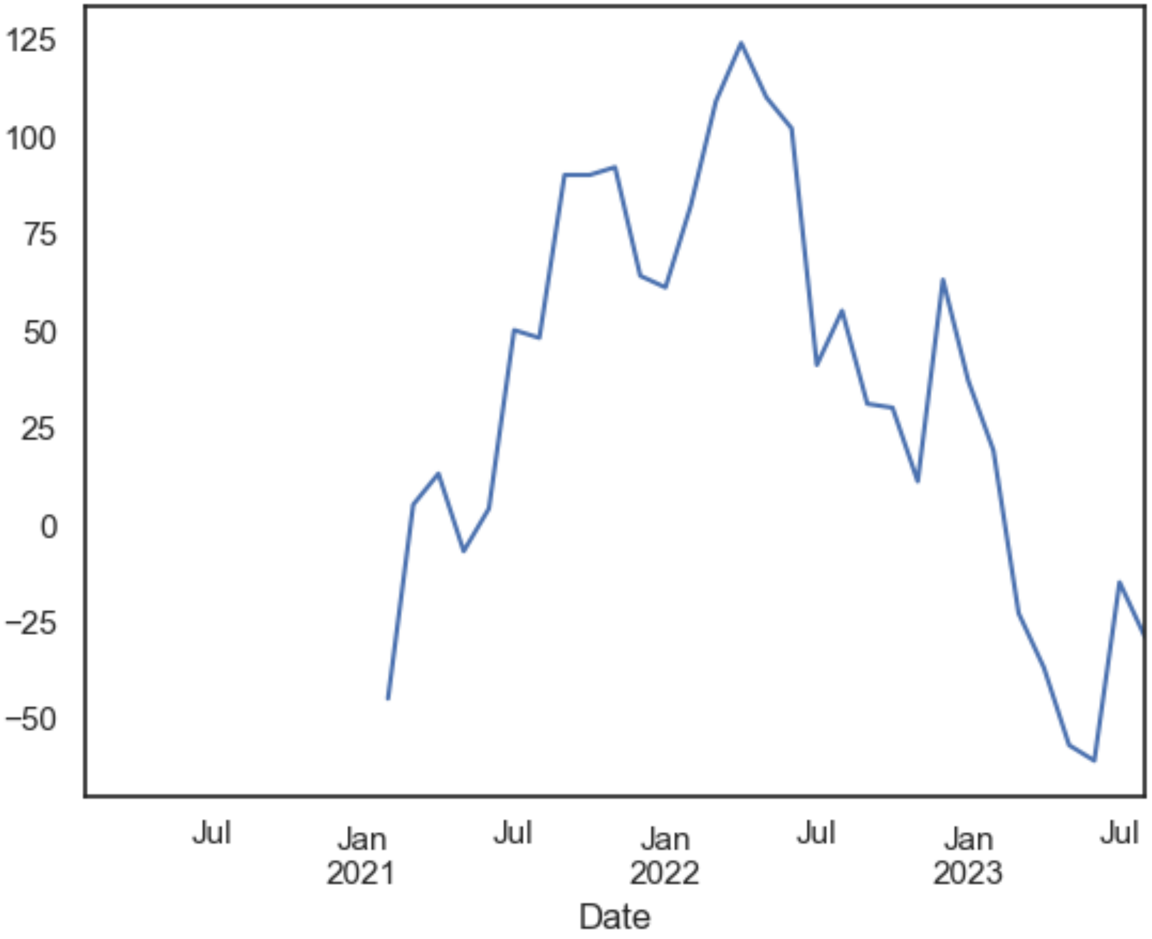
```
In [93]: ADF_CHECK(df_1["Difference_1"])
```

```
Augmented Dicky-Fuller Test
ADF Test statistic:-7.094152425938138
p-value:4.332824012200676e-10
# of lags:0
Num of observations used:42
Reject null hypothesis
Data does not have a unit root and is stationery
```

Data is Stationary now

```
In [94]: df_1["Seasonal Difference"] = df_1['Crime']-df_1['Crime'].shift(12)
         df_1["Seasonal Difference"].plot()
```

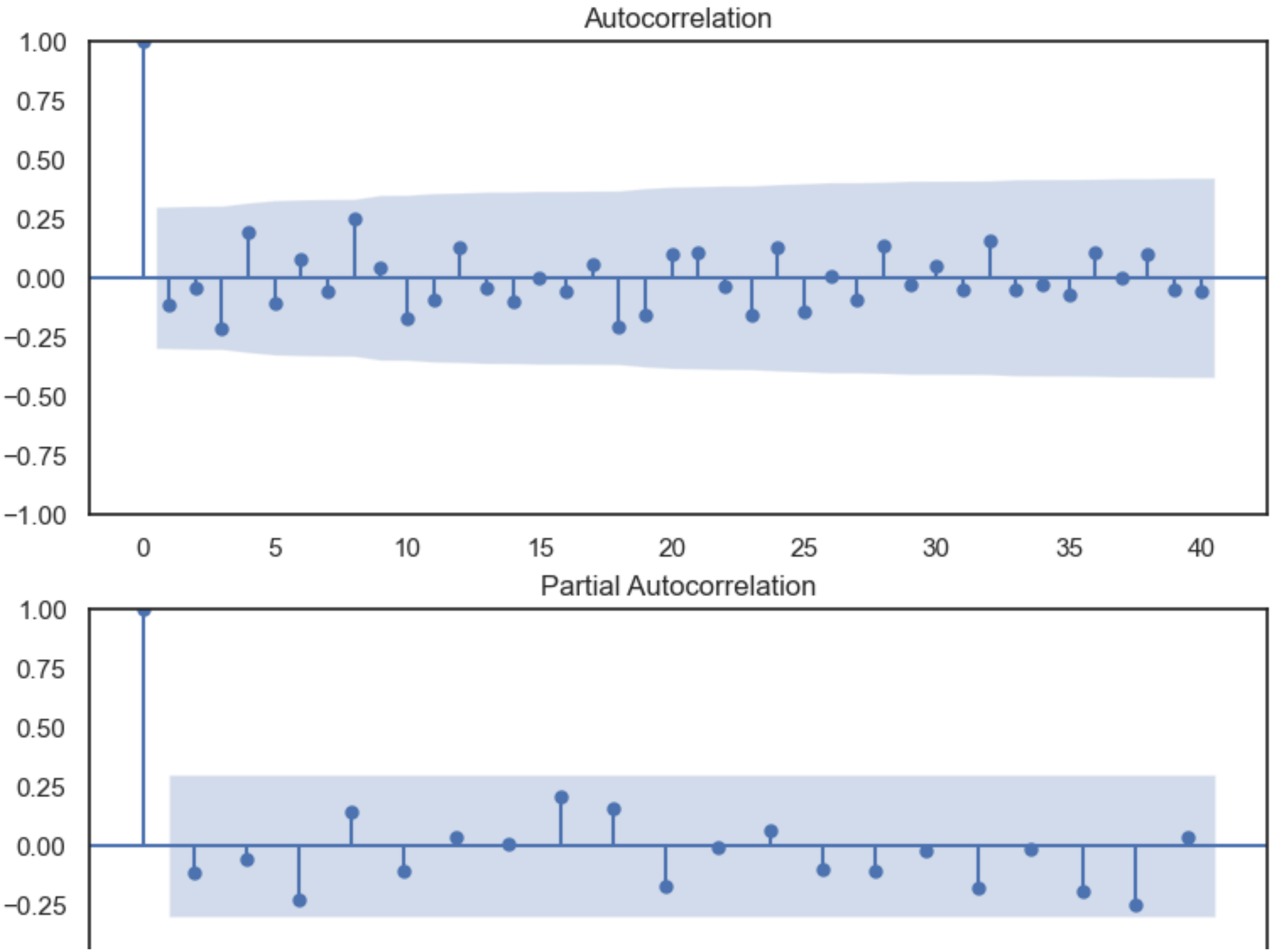
```
Out[94]: <Axes: xlabel='Date'>
```

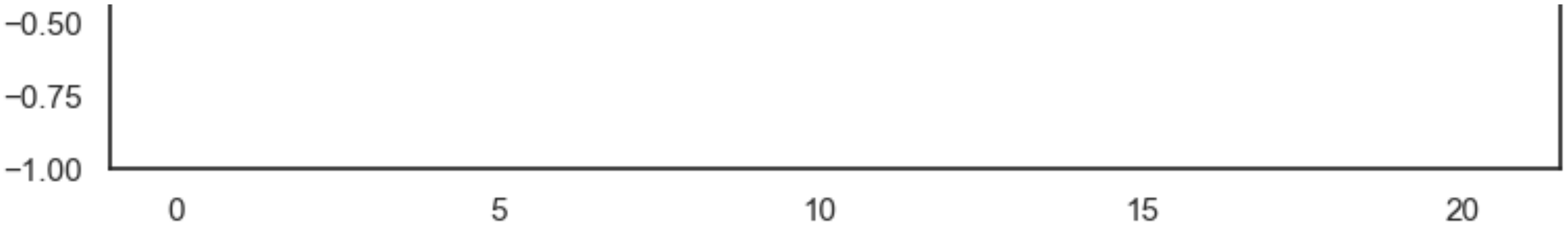


We will look at the Autocorrelation function (ACF) plot and the Partial Autocorrelation Function (PACF) plots to get get a sense of which lags are significant.

In [95]:

```
fig, (ax1, ax2) = plt.subplots(2,1, figsize=(9,8))

ACF = plot_acf(df_1["Difference_1"],lags=40, alpha=0.05, ax=ax1)
PACF = plot_pacf(df_1["Difference_1"],lags=20,alpha=0.05,ax=ax2)
plt.show()
```





"When data have a trend, the autocorrelations for small lags tend to be large and positive because observations nearby in time are also nearby in size. So the ACF of trended time series tend to have positive values that slowly decrease as the lags increase. When data are seasonal, the autocorrelations will be larger for the seasonal lags (at multiples of the seasonal frequency) than for other lags." -- https://otexts.com/fpp2/autocorrelation.html

Implementing the Seasonal ARIMA time series model to forecast the future crimes.

In [96]:
```
SARIMA_model = SARIMAX(df_1['Crime'],order = (1,1,1),seasonal_order = (1,1,1,12))
```

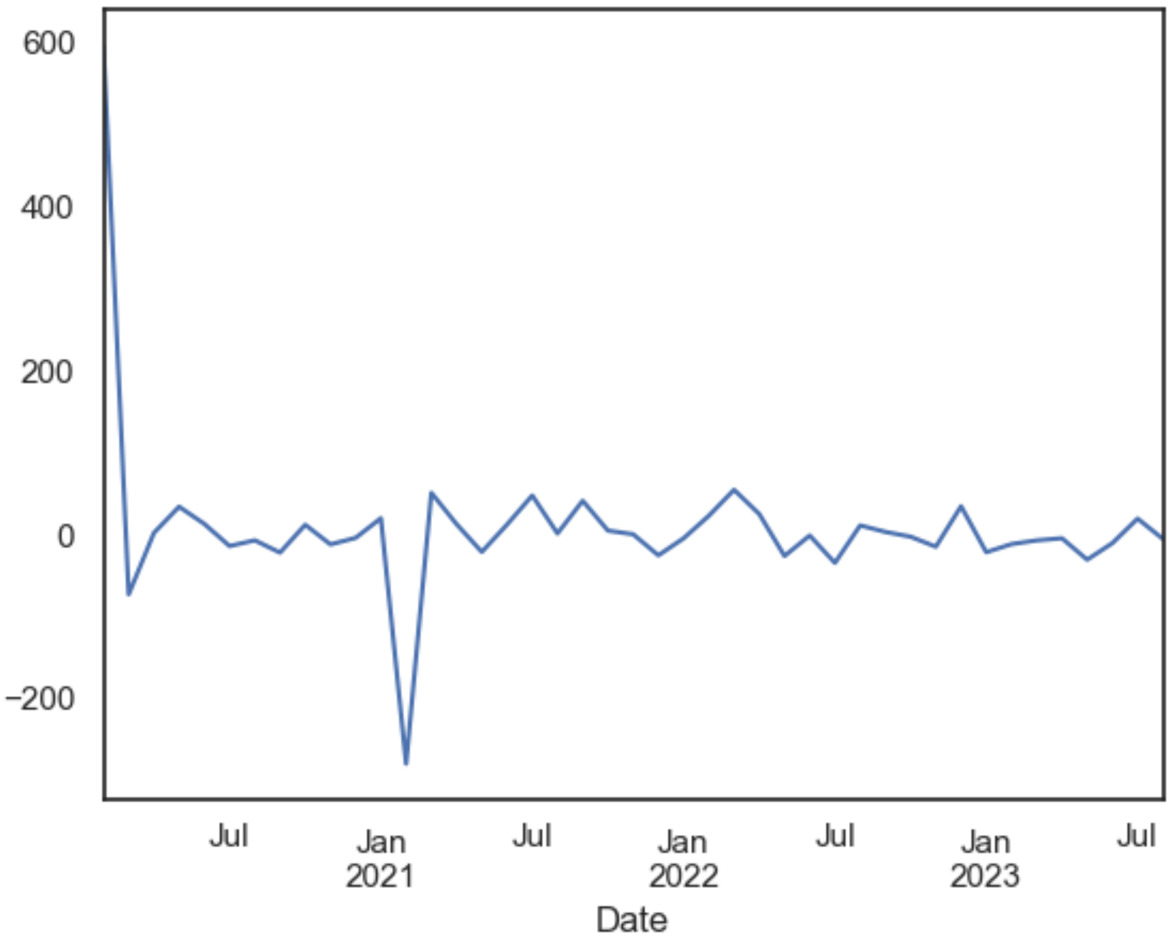
In [97]:
```
result = SARIMA_model.fit(disp=0)
```

In [98]:
```
# reults
print(result.summary())
```

```
                                     SARIMAX Results                                      
==========================================================================================
Dep. Variable:                              Crime   No. Observations:                   43
Model:             SARIMAX(1, 1, 1)x(1, 1, 1, 12)   Log Likelihood                -137.846
Date:                            Thu, 11 Jan 2024   AIC                            285.692
Time:                                    22:22:43   BIC                            292.698
Sample:                                02-01-2020   HQIC                           287.934
                                     - 08-01-2023                                         
Covariance Type:                              opg                                         
==============================================================================
                 coef    std err          z      P>|z|      [0.025      0.975]
------------------------------------------------------------------------------
ar.L1         -0.8471      0.995     -0.852      0.394      -2.797       1.102
ma.L1          0.7741      1.132      0.684      0.494      -1.445       2.994
ar.S.L12      -0.2485      0.562     -0.442      0.658      -1.349       0.852
ma.S.L12      -0.4670      0.908     -0.515      0.607      -2.246       1.312
sigma2       468.0477    199.320      2.348      0.019      77.387     858.708
===================================================================================
Ljung-Box (L1) (Q):                   0.26   Jarque-Bera (JB):                 1.31
Prob(Q):                              0.61   Prob(JB):                         0.52
Heteroskedasticity (H):               0.69   Skew:                             0.46
Prob(H) (two-sided):                  0.57   Kurtosis:                         2.53
===================================================================================

Warnings:
[1] Covariance matrix calculated using the outer product of gradients (complex-step).
```

In [99]:

```
# graphical representation of residuals (goodness of fit)
result.resid.plot()
plt.show();
```



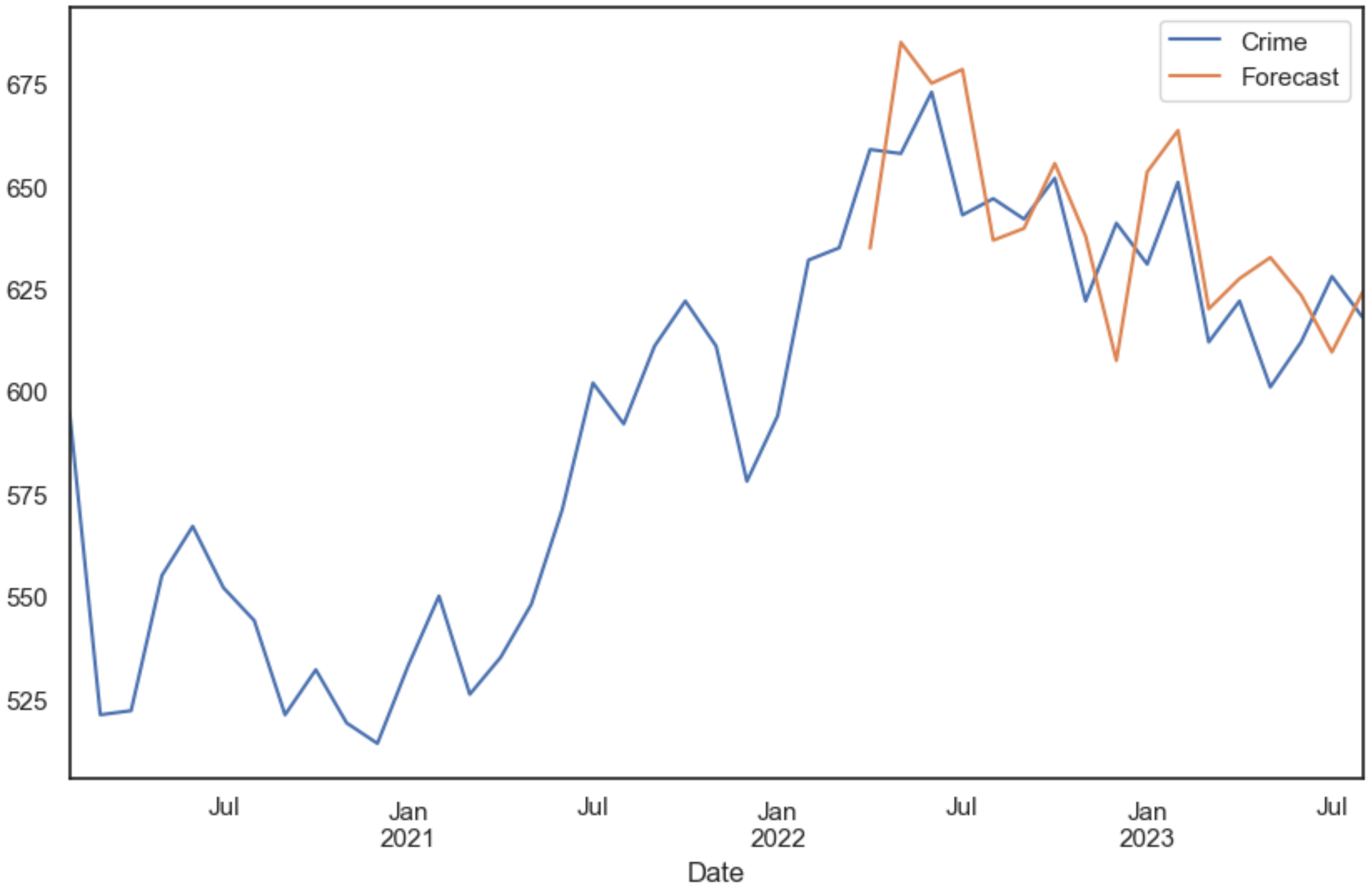
Backtesting - Visualising how the model is performing over the past data

In [100…

```
predicted = result.predict(start=17, end=44)
df_1["Forecast"] = result.predict(start=26, end=44)
df_1[["Crime","Forecast"]].plot(figsize = (10,6))
#Orange one is the one predicted by the sarima model
```

Out[100]:

```
<Axes: xlabel='Date'>
```



As we can see, the model fits really well for the past data

## Generating the future dates in order to forecast the number of crimes in future

In [101…

```
from pandas.tseries.offsets import DateOffset
```

In [102…

```
future_dates = [df_1.index[-1] + DateOffset(months = x) for x in range(1,10)]
```

In [103…

```
future_data = pd.DataFrame(index = future_dates,columns = df_1.columns)
```

In [104…

```
final_data = pd.concat([df_1,future_data])
final_data.head()
```

Out[104]:

| | Crime | Difference_1 | Seasonal Difference | Forecast |
|---|---|---|---|---|
| **2020-02-01** | 595.0 | -1.0 | NaN | NaN |
| **2020-03-01** | 521.0 | -74.0 | NaN | NaN |
| **2020-04-01** | 522.0 | 1.0 | NaN | NaN |
| **2020-05-01** | 555.0 | 33.0 | NaN | NaN |
| **2020-06-01** | 567.0 | 12.0 | NaN | NaN |

In [105…

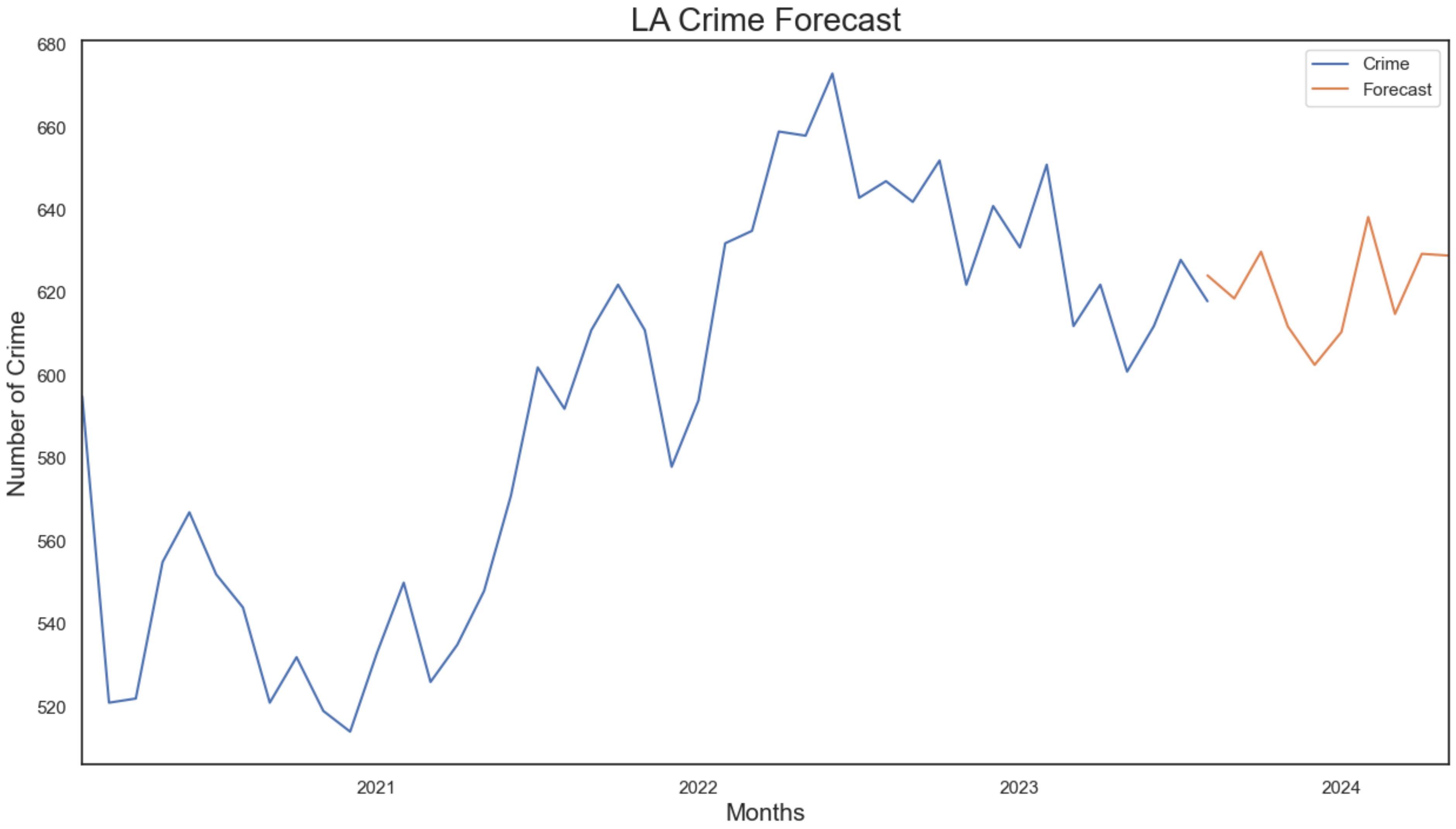
```
# forecasting
final_data["Forecast"] = result.predict(start = 42,end = 57)
final_data["Forecast"][44]
```

Out[105]:

```
629.9570444430585
```

In [106…

```
# visualising the number of crimes, that are forecasted by our model
final_data["Crime"].plot()
final_data["Forecast"].plot(figsize = (15,8))
plt.title("LA Crime Forecast",size = 20)
plt.xlabel("Months",size=15)
plt.ylabel("Number of Crime",size=15)
plt.legend();
```



In [107…

```
from sklearn.metrics import mean_squared_error
from math import sqrt

actual = df_1['Crime'][17:-1]
predicted = predicted[:-3]

rmse = sqrt(mean_squared_error(actual,predicted))
print('RMSE of SARIMA model:',round(rmse,2))
```

```
RMSE of SARIMA model: 23.84
```

## Prophet time series forecasting

In [110…

```python
from prophet import Prophet
```

In [111…

```python
df = crime_data.copy()
```

In [112…

```python
df.head()
```

Out[112]:

| | DR_NO | Date Rptd | Date Time OCC | AREA NAME | Rpt Dist No | Crm Cd | Crm Cd Desc | Vict Age | Vict Sex | Vict Descent | ... | Status Desc | Crm Cd 1 | LOCATION | LAT | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **0** | 10304468 | 2020-01-08 | 2020-01-08 22:30:00 | Southwest | 377 | 624 | BATTERY - SIMPLE ASSAULT | 36 | F | B | ... | Adult Other | 624.0 | 1100 W 39TH PL | 34.0141 | -1 |
| **1** | 190101086 | 2020-01-02 | 2020-01-01 03:30:00 | Central | 163 | 624 | BATTERY - SIMPLE ASSAULT | 25 | M | H | ... | Invest Cont | 624.0 | 700 S HILL ST | 34.0459 | -11 |
| **2** | 200110444 | 2020-04-14 | 2020-02-13 12:00:00 | Central | 155 | 845 | SEX OFFENDER REGISTRANT OUT OF COMPLIANCE | 0 | X | X | ... | Adult Arrest | 845.0 | 200 E 6TH ST | 34.0448 | -1 |
| **3** | 191501505 | 2020-01-01 | 2020-01-01 17:30:00 | N Hollywood | 1543 | 745 | VANDALISM - MISDEAMEANOR ($399 OR UNDER) | 76 | F | W | ... | Invest Cont | 745.0 | 5400 CORTEEN PL | 34.1685 | -1 |
| **4** | 191921269 | 2020-01-01 | 2020-01-01 04:15:00 | Mission | 1998 | 740 | VANDALISM - FELONY ($400 & OVER, ALL CHURCH VA... | 31 | X | X | ... | Invest Cont | 740.0 | 14400 TITUS ST | 34.2198 | -11 |

5 rows × 23 columns

In [113…

```
df['month-year'] = df['Date Time OCC'].dt.strftime('%m-%Y')
df.head(3)
```

Out[113]:

| | DR_NO | Date Rptd | Date Time OCC | AREA NAME | Rpt Dist No | Crm Cd | Crm Cd Desc | Vict Age | Vict Sex | Vict Descent | ... | Crm Cd 1 | LOCATION | LAT | LON | Wea U |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **0** | 10304468 | 2020-01-08 | 2020-01-08 22:30:00 | Southwest | 377 | 624 | BATTERY - SIMPLE ASSAULT | 36 | F | B | ... | 624.0 | 1100 W 39TH PL | 34.0141 | -118.2978 | |
| **1** | 190101086 | 2020-01-02 | 2020-01-01 03:30:00 | Central | 163 | 624 | BATTERY - SIMPLE ASSAULT | 25 | M | H | ... | 624.0 | 700 S HILL ST | 34.0459 | -118.2545 | |
| **2** | 200110444 | 2020-04-14 | 2020-02-13 12:00:00 | Central | 155 | 845 | SEX OFFENDER REGISTRANT OUT OF COMPLIANCE | 0 | X | X | ... | 845.0 | 200 E 6TH ST | 34.0448 | -118.2474 | |

3 rows × 24 columns

In [114…

```
df = df.groupby('month-year')['Crm Cd'].count()
```

In [115…

```
df_2 = df.to_frame().reset_index()
df_2 = df_2.rename(columns={'month-year':'Date','Crm Cd':'Crime'})
df_2['Date'] = pd.to_datetime(df_2['Date'])
df_2['days_in_month'] = df_2['Date'].dt.days_in_month
df_2.head()
```

Out[115]:

|   | Date | Crime | days_in_month |
|---|---|---|---|
| **0** | 2020-01-01 | 18473 | 31 |
| **1** | 2021-01-01 | 16525 | 31 |
| **2** | 2022-01-01 | 18424 | 31 |
| **3** | 2023-01-01 | 19548 | 31 |
| **4** | 2020-02-01 | 17252 | 29 |

In [116…

```
df_2['Crime'] = round((df_2['Crime']/df_2['days_in_month']))
df_2.drop(columns=['days_in_month'],inplace=True)
df_2.tail()
```

Out[116]:

|   | Date | Crime |
|---|---|---|
| **40** | 2021-11-01 | 611.0 |
| **41** | 2022-11-01 | 622.0 |
| **42** | 2020-12-01 | 514.0 |
| **43** | 2021-12-01 | 578.0 |
| **44** | 2022-12-01 | 641.0 |

In [117…

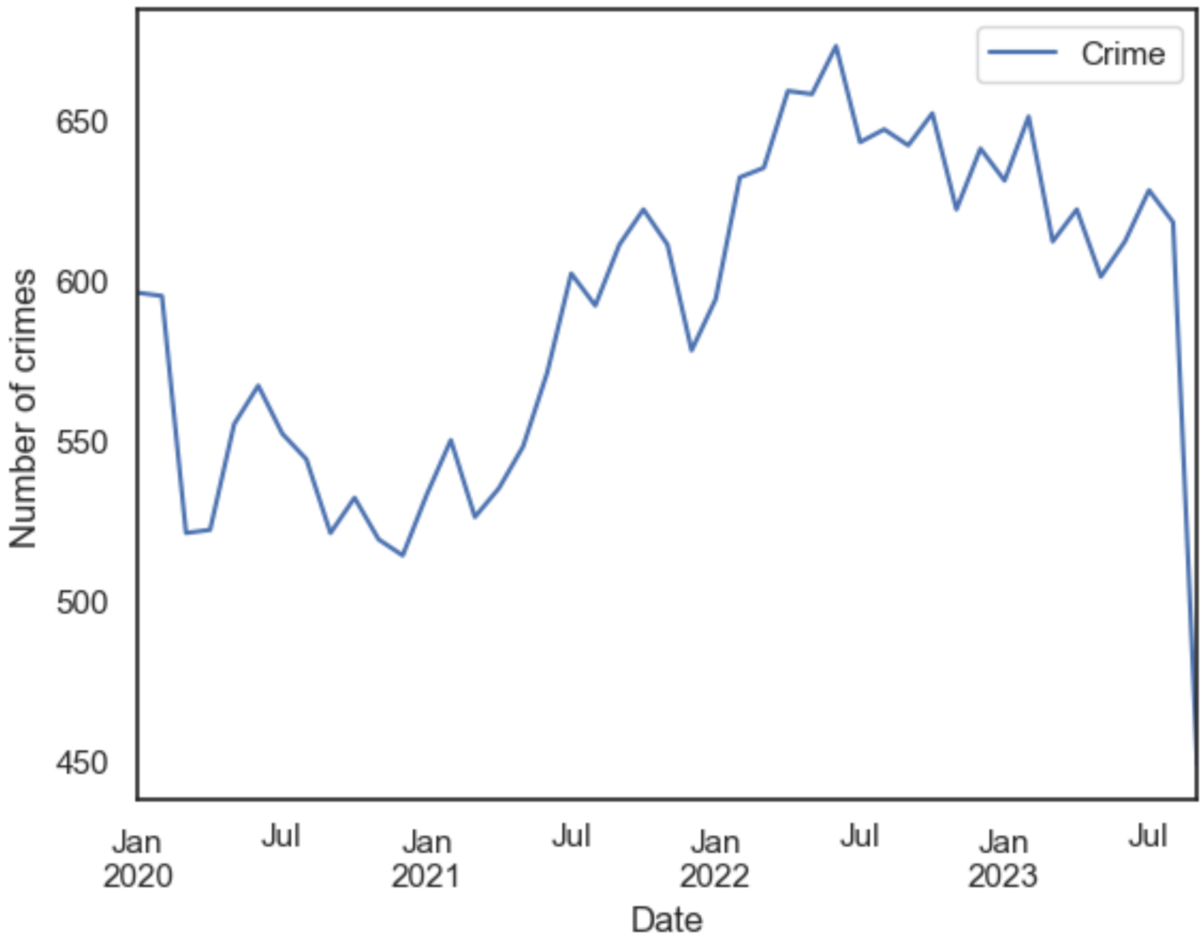
```
df_2 = df_2.sort_values(by='Date').reset_index(drop=True)
```

In [118…

```
df_2.set_index('Date',inplace=True)
```

In [119…

```
df_2.plot()
plt.ylabel('Number of crimes');
```



In [120…

```
df_2.reset_index(inplace=True)
```

In [121…

```
df_2.rename(columns={'Date':'ds','Crime':'y'},inplace=True)
```

In [122…

```
# Implementing Prophet time series forecasting
model = Prophet()
model.fit(df_2)
```

```
22:24:05 - cmdstanpy - INFO - Chain [1] start processing
22:24:05 - cmdstanpy - INFO - Chain [1] done processing
```

Out[122]:

```
<prophet.forecaster.Prophet at 0x2f6dd1390>
```

In [123…

```
# generating the future dates
future_date = model.make_future_dataframe(periods=10,freq='m')
```

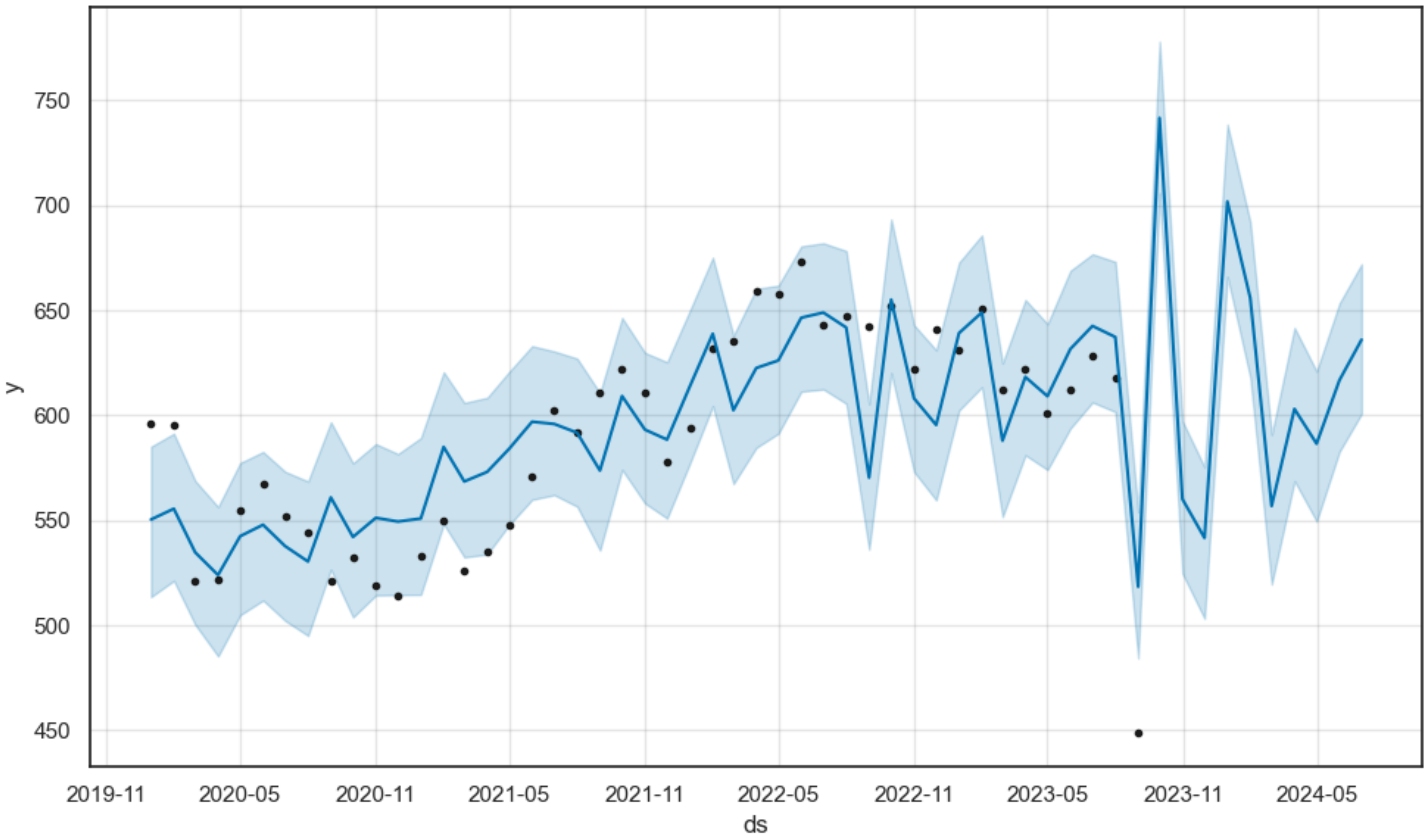
In [124…

```
# predicting for future dates
prediction = model.predict(future_date)
```

In [125…

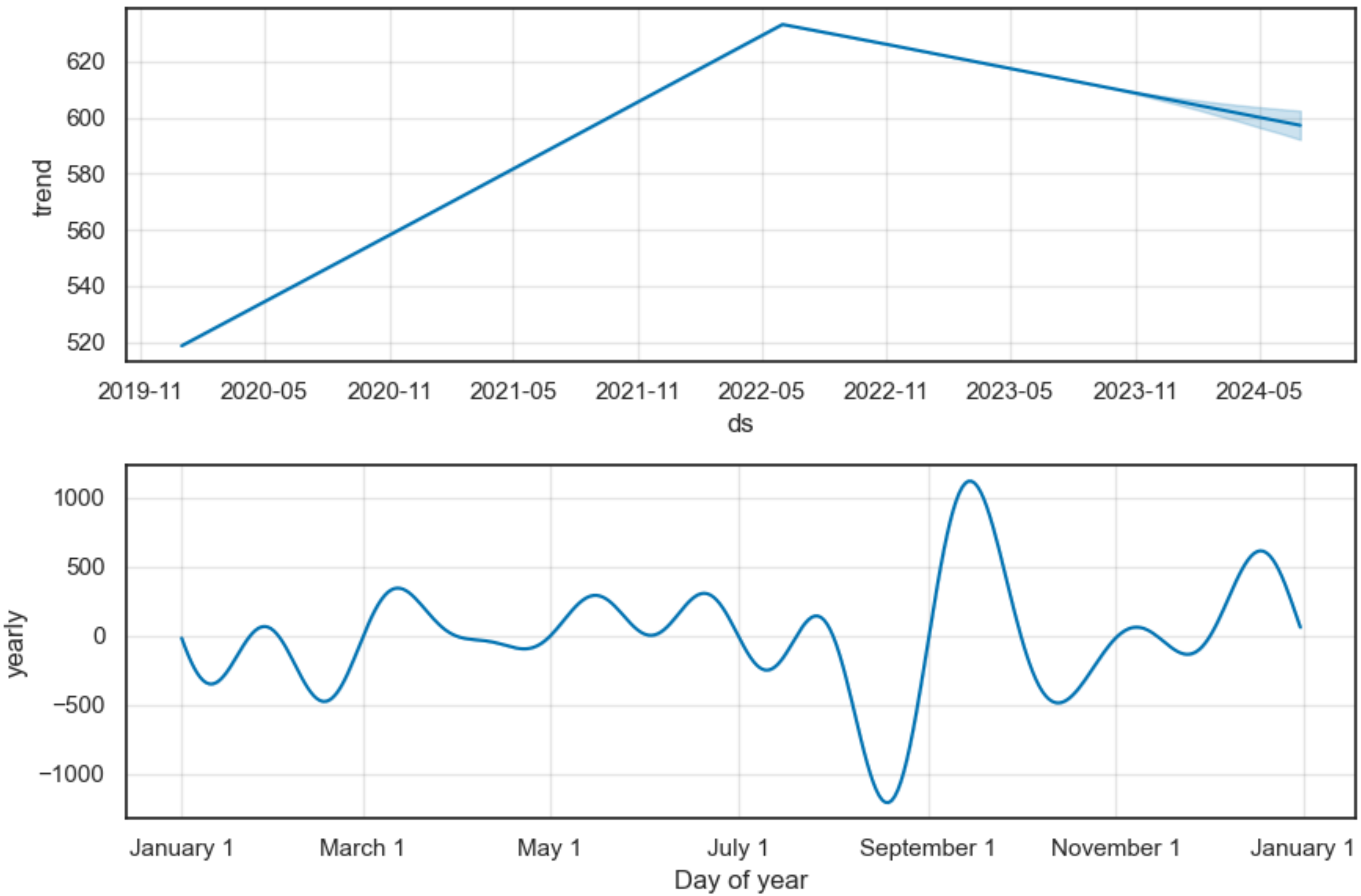
```
# visualising the predicted number of crime
plt.figure(figsize=(6,4))
model.plot(prediction)
plt.show();
```

```
<Figure size 600x400 with 0 Axes>
```



In [126…

```
# plotting predicted components
model.plot_components(prediction)
plt.show()
```


trend
620
600
580
560
540
520
2019-11
2020-05
2020-11
2021-05
2021-11
2022-05
2022-11
2023-05
2023-11
2024-05
ds
yearly
1000
500
0
−500
−1000
January 1
March 1
May 1
July 1
September 1
November 1
January 1
Day of year

In [127…

```python
from prophet.diagnostics import cross_validation, performance_metrics

# Perform cross-validation with initial 365 days for the first training data and the cut-off for every 180 days.
cutoffs = pd.date_range(start='2020-02-01', end='2022-12-01', freq='1MS')
df_cv = cross_validation(model, horizon='90 days', cutoffs=cutoffs)

# Calculate evaluation metrics
res = performance_metrics(df_cv)
res
```

```
Seasonality has period of 365.25 days which is larger than initial window. Consider increasing initial.
  0%|          | 0/35 [00:00<?, ?it/s]
22:24:06 - cmdstanpy - INFO - Chain [1] start processing
22:24:07 - cmdstanpy - INFO - Chain [1] done processing
22:24:07 - cmdstanpy - INFO - Chain [1] start processing
22:24:07 - cmdstanpy - INFO - Chain [1] done processing
22:24:07 - cmdstanpy - INFO - Chain [1] start processing
22:24:07 - cmdstanpy - INFO - Chain [1] done processing
22:24:07 - cmdstanpy - INFO - Chain [1] start processing
22:24:07 - cmdstanpy - INFO - Chain [1] done processing
22:24:07 - cmdstanpy - INFO - Chain [1] start processing
22:24:08 - cmdstanpy - INFO - Chain [1] done processing
22:24:08 - cmdstanpy - INFO - Chain [1] start processing
22:24:10 - cmdstanpy - INFO - Chain [1] done processing
22:24:10 - cmdstanpy - INFO - Chain [1] start processing
22:24:10 - cmdstanpy - INFO - Chain [1] done processing
22:24:10 - cmdstanpy - INFO - Chain [1] start processing
22:24:17 - cmdstanpy - INFO - Chain [1] done processing
22:24:17 - cmdstanpy - INFO - Chain [1] start processing
22:24:17 - cmdstanpy - INFO - Chain [1] done processing
22:24:17 - cmdstanpy - INFO - Chain [1] start processing
22:24:17 - cmdstanpy - INFO - Chain [1] done processing
22:24:17 - cmdstanpy - INFO - Chain [1] start processing
22:24:19 - cmdstanpy - INFO - Chain [1] done processing
22:24:19 - cmdstanpy - INFO - Chain [1] start processing
22:24:24 - cmdstanpy - INFO - Chain [1] done processing
22:24:24 - cmdstanpy - INFO - Chain [1] start processing
22:24:32 - cmdstanpy - INFO - Chain [1] done processing
22:24:32 - cmdstanpy - INFO - Chain [1] start processing
22:24:40 - cmdstanpy - INFO - Chain [1] done processing
```

```
22:24:40 - cmdstanpy - INFO - Chain [1] start processing
22:24:48 - cmdstanpy - INFO - Chain [1] done processing
22:24:48 - cmdstanpy - INFO - Chain [1] start processing
22:24:52 - cmdstanpy - INFO - Chain [1] done processing
22:24:52 - cmdstanpy - INFO - Chain [1] start processing
22:25:03 - cmdstanpy - INFO - Chain [1] done processing
22:25:03 - cmdstanpy - INFO - Chain [1] start processing
22:25:15 - cmdstanpy - INFO - Chain [1] done processing
22:25:15 - cmdstanpy - INFO - Chain [1] start processing
22:25:25 - cmdstanpy - INFO - Chain [1] done processing
22:25:25 - cmdstanpy - INFO - Chain [1] start processing
22:25:35 - cmdstanpy - INFO - Chain [1] done processing
22:25:35 - cmdstanpy - INFO - Chain [1] start processing
22:25:46 - cmdstanpy - INFO - Chain [1] done processing
22:25:46 - cmdstanpy - INFO - Chain [1] start processing
22:25:57 - cmdstanpy - INFO - Chain [1] done processing
22:25:58 - cmdstanpy - INFO - Chain [1] start processing
22:26:09 - cmdstanpy - INFO - Chain [1] done processing
22:26:09 - cmdstanpy - INFO - Chain [1] start processing
22:26:21 - cmdstanpy - INFO - Chain [1] done processing
22:26:21 - cmdstanpy - INFO - Chain [1] start processing
22:26:21 - cmdstanpy - INFO - Chain [1] done processing
22:26:22 - cmdstanpy - INFO - Chain [1] start processing
22:26:22 - cmdstanpy - INFO - Chain [1] done processing
22:26:22 - cmdstanpy - INFO - Chain [1] start processing
22:26:22 - cmdstanpy - INFO - Chain [1] done processing
22:26:22 - cmdstanpy - INFO - Chain [1] start processing
22:26:23 - cmdstanpy - INFO - Chain [1] done processing
22:26:23 - cmdstanpy - INFO - Chain [1] start processing
22:26:23 - cmdstanpy - INFO - Chain [1] done processing
22:26:23 - cmdstanpy - INFO - Chain [1] start processing
22:26:23 - cmdstanpy - INFO - Chain [1] done processing
22:26:23 - cmdstanpy - INFO - Chain [1] start processing
22:26:23 - cmdstanpy - INFO - Chain [1] done processing
22:26:23 - cmdstanpy - INFO - Chain [1] start processing
22:26:24 - cmdstanpy - INFO - Chain [1] done processing
22:26:24 - cmdstanpy - INFO - Chain [1] start processing
22:26:24 - cmdstanpy - INFO - Chain [1] done processing
22:26:24 - cmdstanpy - INFO - Chain [1] start processing
22:26:24 - cmdstanpy - INFO - Chain [1] done processing
22:26:24 - cmdstanpy - INFO - Chain [1] start processing
```

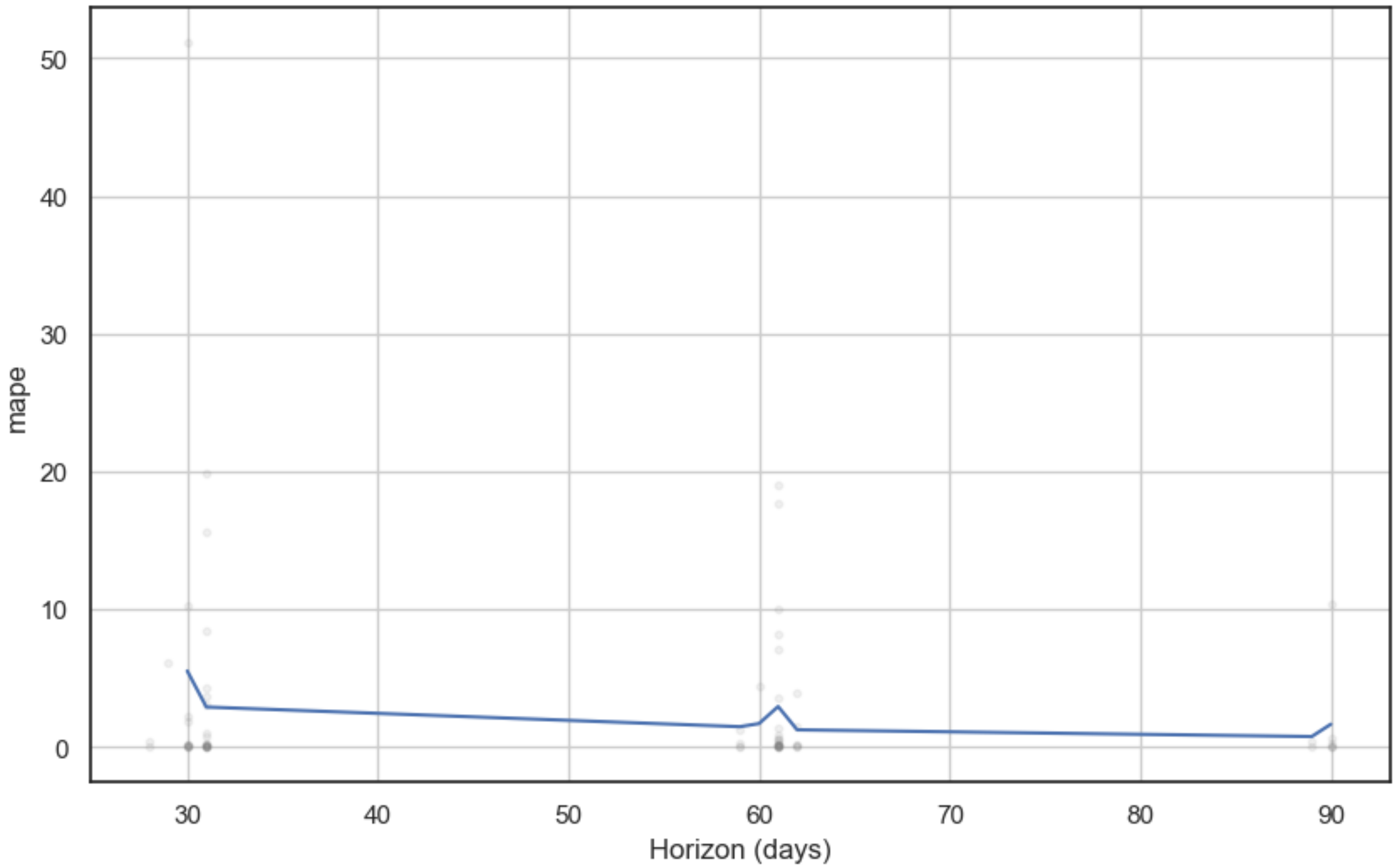
```
22:26:24 - cmdstanpy - INFO - Chain [1] done processing
```

Out[127]:

| | horizon | mse | rmse | mae | mape | mdape | smape | coverage |
|---|---|---|---|---|---|---|---|---|
| **0** | 30 days | 6.024793e+07 | 7761.954022 | 2874.054895 | 5.522660 | 0.133480 | 0.624363 | 0.0 |
| **1** | 31 days | 1.137902e+07 | 3373.279876 | 1586.868098 | 2.904921 | 0.137641 | 0.629735 | 0.0 |
| **2** | 59 days | 4.945284e+06 | 2223.799525 | 805.313447 | 1.479315 | 0.796189 | 0.428217 | 0.0 |
| **3** | 60 days | 4.099586e+06 | 2024.743471 | 912.400584 | 1.703757 | 0.796189 | 0.535731 | 0.0 |
| **4** | 61 days | 1.033630e+07 | 3215.011466 | 1560.173643 | 2.943175 | 0.188097 | 0.776290 | 0.0 |
| **5** | 62 days | 2.186576e+06 | 1478.707444 | 667.602638 | 1.250737 | 0.111741 | 0.632595 | 0.0 |
| **6** | 89 days | 5.993151e+05 | 774.154463 | 411.207334 | 0.763372 | 0.111741 | 0.496947 | 0.0 |
| **7** | 90 days | 4.711529e+06 | 2170.605720 | 919.051664 | 1.656187 | 0.091759 | 0.447047 | 0.0 |

In [128…

```
from prophet.plot import plot_cross_validation_metric
#choose between 'mse', 'rmse', 'mae', 'mape', 'coverage'

plot_cross_validation_metric(df_cv, metric= 'mape')
plt.show()
```



```
In [ ]:
```