DATA ACQUISITION

In [1]:

```python
import pandas as pd
import warnings

# Settings the warnings to be ignored
warnings.filterwarnings('ignore')

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

DATA INSPECTION

In [2]:

```python
#FEW ROWS
data.head(10)
```

Out[2]:

	DR_NO	Date Rptd	DATE OCC	TIME OCC	AREA	AREA NAME	Rpt Dist No	Part 1-2	Crm Cd
0	211507896	04/11/2021 12:00:00 AM	11/07/2020 12:00:00 AM	845	15	N Hollywood	1502	2	354
1	201516622	10/21/2020 12:00:00 AM	10/18/2020 12:00:00 AM	1845	15	N Hollywood	1521	1	230
2	240913563	12/10/2024 12:00:00 AM	10/30/2020 12:00:00 AM	1240	9	Van Nuys	933	2	354
3	210704711	12/24/2020 12:00:00 AM	12/24/2020 12:00:00 AM	1310	7	Wilshire	782	1	331
4	201418201	10/03/2020 12:00:00 AM	09/29/2020 12:00:00 AM	1830	14	Pacific	1454	1	420
5	240412063	12/11/2024 12:00:00 AM	11/11/2020 12:00:00 AM	1210	4	Hollenbeck	429	2	354
6	240317069	12/16/2024 12:00:00 AM	04/16/2020 12:00:00 AM	1350	3	Southwest	396	2	354
7	201115217	10/29/2020 12:00:00 AM	07/07/2020 12:00:00 AM	1400	11	Northeast	1133	2	812
8	241708596	04/20/2024 12:00:00 AM	03/02/2020 12:00:00 AM	1200	17	Devonshire	1729	2	354
9	242113813	12/18/2024 12:00:00 AM	09/01/2020 12:00:00 AM	900	21	Topanga	2196	2	354

10 rows × 28 columns

In [3]:

```
# Check data types
print(data.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          float64
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 [4]:

```
#description
data.describe(include='all')
```

Out[4]:

	DR_NO	Date Rptd	DATE OCC	TIME OCC	AREA	AR NA
count	5.769100e+04	57691	57691	57691.000000	57691.000000	576
unique	NaN	1388	366	NaN	NaN	
top	NaN	12/31/2020 12:00:00 AM	01/01/2020 12:00:00 AM	NaN	NaN	Paci
freq	NaN	247	433	NaN	NaN	77
mean	2.030014e+08	NaN	NaN	1347.112323	16.430188	N
std	4.659494e+06	NaN	NaN	645.966961	3.478736	N
min	1.903265e+08	NaN	NaN	1.000000	1.000000	N
25%	2.015102e+08	NaN	NaN	927.500000	14.000000	N
50%	2.018061e+08	NaN	NaN	1425.000000	17.000000	N
75%	2.020104e+08	NaN	NaN	1900.000000	19.000000	N
max	2.514041e+08	NaN	NaN	2359.000000	21.000000	N

11 rows × 28 columns

In [5]:

```
#Review Column names
data.columns.tolist()
```

```
Out[5]: ['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']
```

```
In [6]: #data information
        data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 57691 entries, 0 to 57690
Data columns (total 28 columns):
 #   Column          Non-Null Count  Dtype  
---  ------          --------------  -----  
 0   DR_NO           57691 non-null  int64  
 1   Date Rptd       57691 non-null  object 
 2   DATE OCC        57691 non-null  object 
 3   TIME OCC        57691 non-null  int64  
 4   AREA            57691 non-null  int64  
 5   AREA NAME       57691 non-null  object 
 6   Rpt Dist No     57691 non-null  int64  
 7   Part 1-2        57691 non-null  int64  
 8   Crm Cd          57691 non-null  int64  
 9   Crm Cd Desc     57691 non-null  object 
 10  Mocodes         49362 non-null  object 
 11  Vict Age        57690 non-null  float64
 12  Vict Sex        49880 non-null  object 
 13  Vict Descent    49880 non-null  object 
 14  Premis Cd       57689 non-null  float64
 15  Premis Desc     57670 non-null  object 
 16  Weapon Used Cd  19908 non-null  float64
 17  Weapon Desc     19908 non-null  object 
 18  Status          57690 non-null  object 
 19  Status Desc     57690 non-null  object 
 20  Crm Cd 1        57690 non-null  float64
 21  Crm Cd 2        4748 non-null   float64
 22  Crm Cd 3        183 non-null    float64
 23  Crm Cd 4        4 non-null      float64
 24  LOCATION        57690 non-null  object 
 25  Cross Street    8994 non-null   object 
 26  LAT             57690 non-null  float64
 27  LON             57690 non-null  float64
dtypes: float64(9), int64(6), object(13)
memory usage: 12.3+ MB
```

DATA CLEANING

In [7]:

```python
# Check for missing values
print(data.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            8329
Vict Age              1
Vict Sex           7811
Vict Descent       7811
Premis Cd             2
Premis Desc          21
Weapon Used Cd    37783
Weapon Desc       37783
Status                1
Status Desc           1
Crm Cd 1              1
Crm Cd 2          52943
Crm Cd 3          57508
Crm Cd 4          57687
LOCATION              1
Cross Street      48697
LAT                   1
LON                   1
dtype: int64
```

In [8]:

```python
# Handling missing data
data['Mocodes'].fillna('Not Specified', inplace=True)
data['Vict Sex'].fillna('Unknown', inplace=True)
data['Vict Descent'].fillna('Unknown', inplace=True)
data['Cross Street'].fillna('Unknown', inplace=True)
data['Weapon Used Cd'].fillna(0, inplace=True)
data['Weapon Desc'].fillna('No Weapon', inplace=True)
data['Crm Cd 1'].fillna(0, inplace=True)
data['Crm Cd 2'].fillna(0, inplace=True)
data['Crm Cd 3'].fillna(0, inplace=True)
data['Crm Cd 4'].fillna(0, inplace=True)
data['Premis Cd'].fillna(0, inplace=True)
data['Premis Desc'].fillna('Unknown', inplace=True)
# Status - fill with most common
data['Status'].fillna(data['Status'].mode()[0], inplace=True)
```

In [9]:

```python
print(data.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          1
Vict Sex          0
Vict Descent      0
Premis Cd         0
Premis Desc       0
Weapon Used Cd    0
Weapon Desc       0
Status            0
Status Desc       1
Crm Cd 1          0
Crm Cd 2          0
Crm Cd 3          0
Crm Cd 4          0
LOCATION          1
Cross Street      0
LAT               1
LON               1
dtype: int64
```

In [10]:

```
# Check for duplicates
print(data.duplicated().sum())
```

```
0
```

In [11]:

```
#Convert Data types
data['Date Rptd'] = pd.to_datetime(data['Date Rptd'], format='%m/%d/%Y %I:%M
data['DATE OCC'] = pd.to_datetime(data['DATE OCC'], format='%m/%d/%Y %I:%M:%
```

In [12]:

```
# Extract time features
data['Year'] = data['DATE OCC'].dt.year
data['Month'] = data['DATE OCC'].dt.month
data['Day'] = data['DATE OCC'].dt.day
data['DayOfWeek'] = data['DATE OCC'].dt.dayofweek
data['DayName'] = data['DATE OCC'].dt.day_name()
data['MonthName'] = data['DATE OCC'].dt.month_name()
```

In [13]:

```
#Checking the converted datatypes
data.dtypes
```

Out[13]:

	0
DR_NO	int64
Date Rptd	datetime64[ns]
DATE OCC	datetime64[ns]
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	float64
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
Year	int32
Month	int32
Day	int32
DayOfWeek	int32
DayName	object

	0
MonthName	object

dtype: object

In [14]:

```
#Check for outliers
data.describe()
```

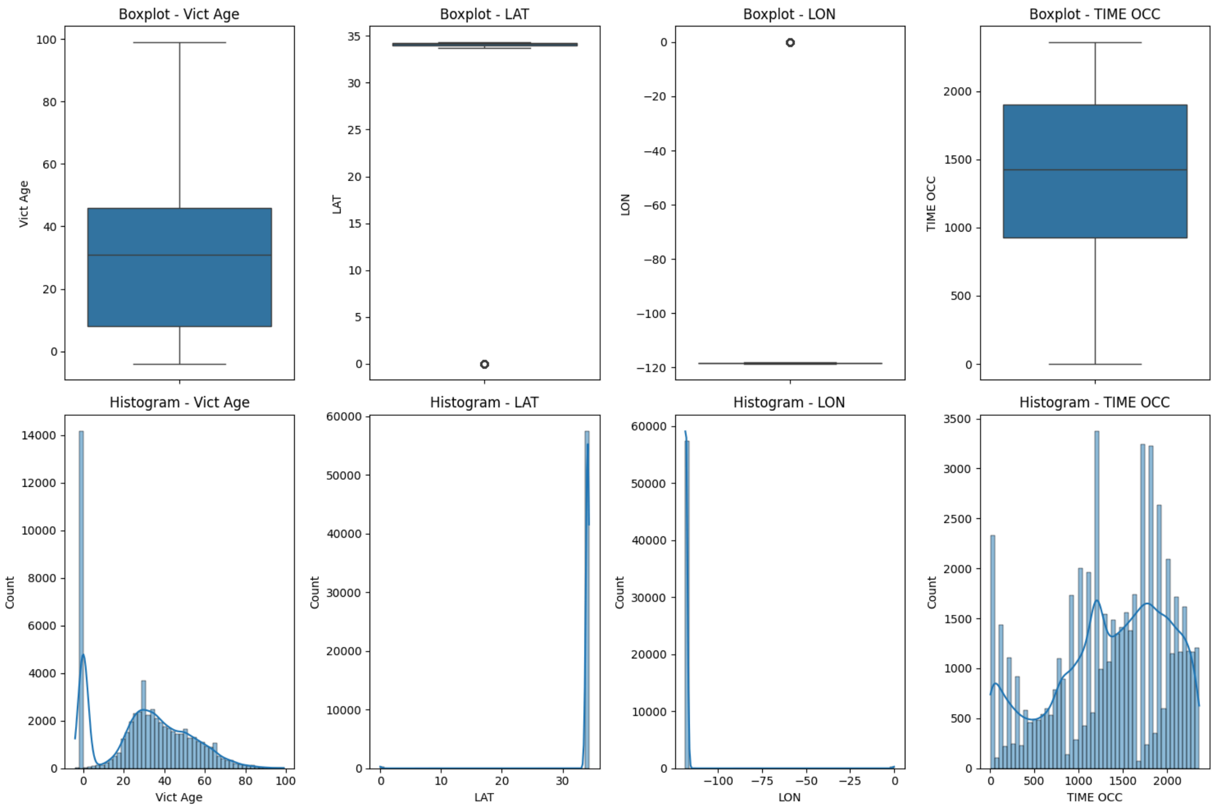
Out[14]:

	DR_NO	Date Rptd	DATE OCC	TIME OCC	
count	5.769100e+04	57691	57691	57691.000000	57
mean	2.030014e+08	2020-08-12 11:30:59.000537088	2020-07-05 05:51:13.207259136	1347.112323	
min	1.903265e+08	2020-01-01 00:00:00	2020-01-01 00:00:00	1.000000	
25%	2.015102e+08	2020-04-20 00:00:00	2020-04-06 00:00:00	927.500000	
50%	2.018061e+08	2020-07-22 00:00:00	2020-07-06 00:00:00	1425.000000	
75%	2.020104e+08	2020-10-27 00:00:00	2020-10-05 00:00:00	1900.000000	
max	2.514041e+08	2024-12-23 00:00:00	2020-12-31 00:00:00	2359.000000	
std	4.659494e+06	NaN	NaN	645.966961	

8 rows × 21 columns

In [15]:

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

# List of columns to check for outliers
columns_to_check = ['Vict Age', 'LAT', 'LON', 'TIME OCC']

# Plotting the distributions and boxplots for these columns
plt.figure(figsize=(15, 10))

for idx, col in enumerate(columns_to_check, 1):
    plt.subplot(2, len(columns_to_check), idx)
    sns.boxplot(data[col])
    plt.title(f"Boxplot - {col}")

    plt.subplot(2, len(columns_to_check), idx + len(columns_to_check))
    sns.histplot(data[col], bins=50, kde=True)
    plt.title(f"Histogram - {col}")

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



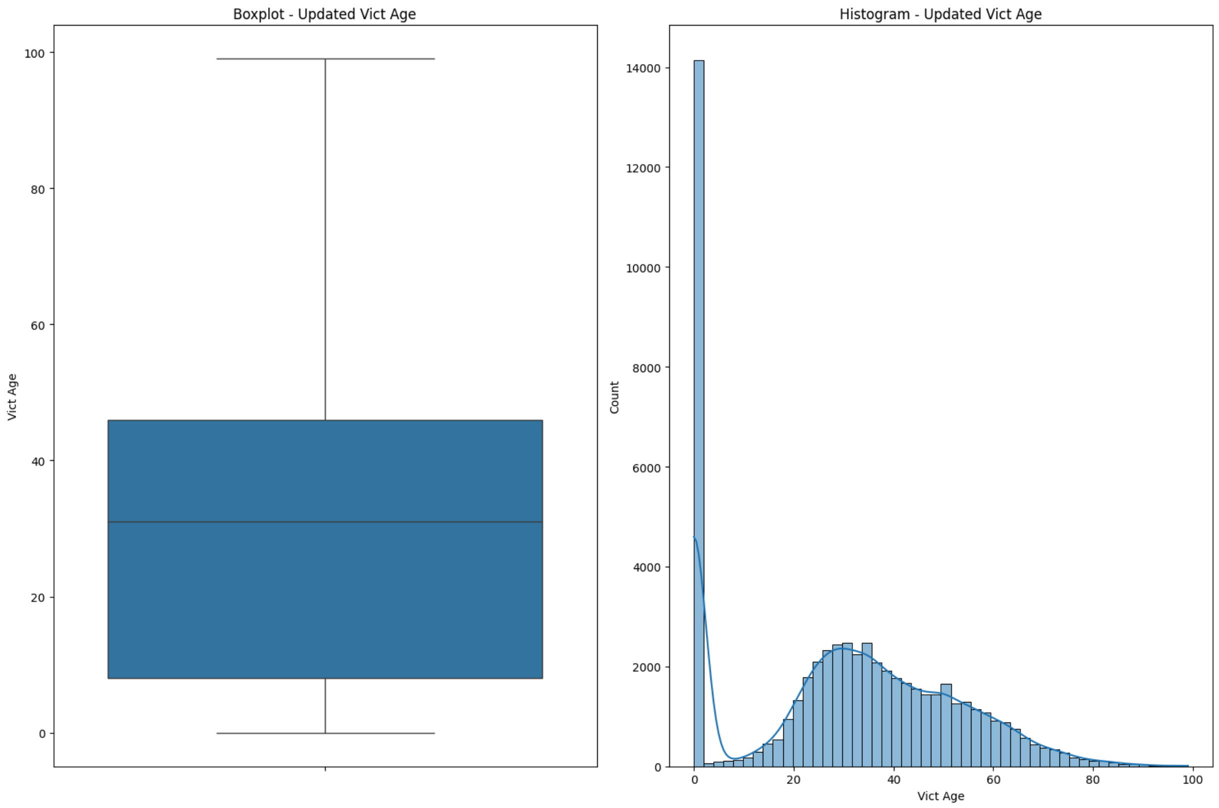
In [16]:

```
# Handling negative values in Vict Age
print(f"Age range before: {data['Vict Age'].min()} to {data['Vict Age'].max(
data = data[(data['Vict Age'] >= 0) & (data['Vict Age'] <= 120)]
print(f"Age range after: {data['Vict Age'].min()} to {data['Vict Age'].max()
```

```
Age range before: -4.0 to 99.0
Age range after: 0.0 to 99.0
```

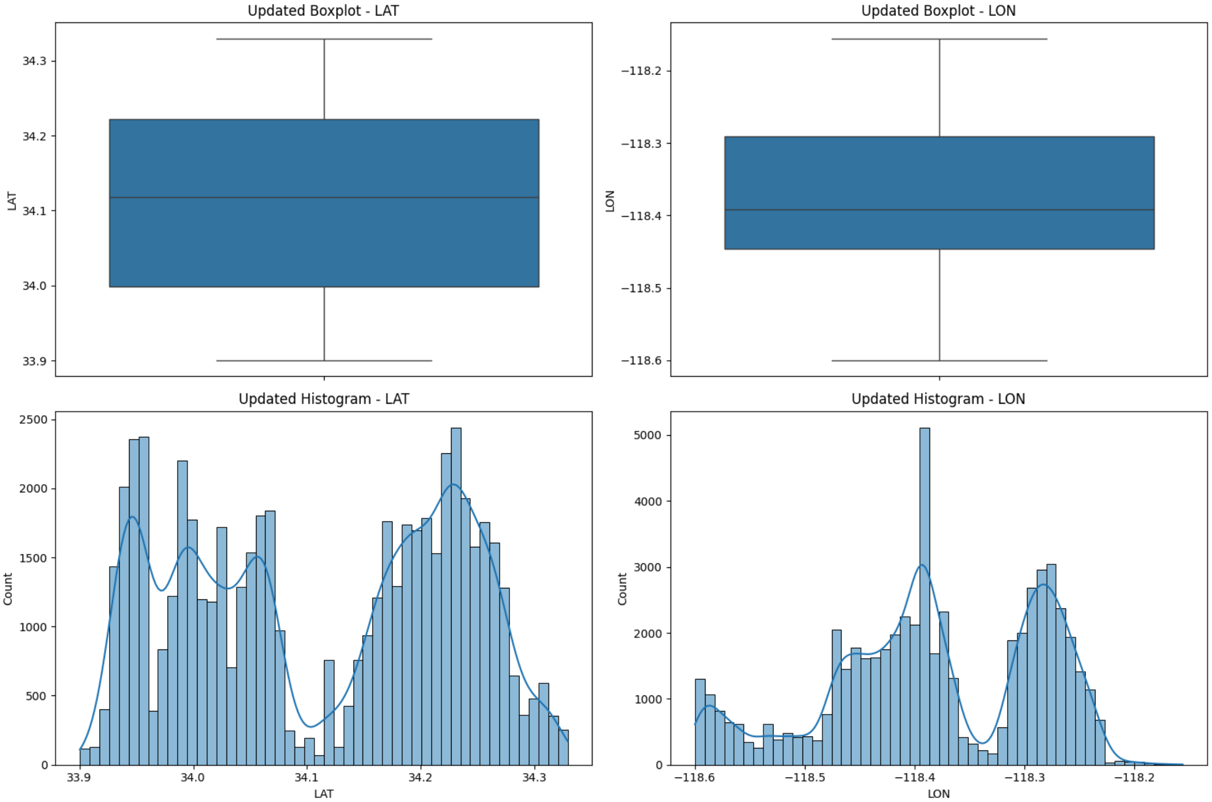
In [17]:

```
# Visualization after dealing with outlier
plt.figure(figsize=(15, 10))

plt.subplot(1, 2, 1)
sns.boxplot(data['Vict Age'])
plt.title("Boxplot - Updated Vict Age")

plt.subplot(1, 2, 2)
sns.histplot(data['Vict Age'], bins=50, kde=True)
plt.title("Histogram - Updated Vict Age")

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



In [18]:

```
# Calculate the median LAT and LOG for outlier correction
median_lat = data['LAT'].median()
median_lon = data['LON'].median()

data['LAT'] = data['LAT'].apply(lambda x: median_lat if x < 33.9 or x > 34.5
data['LON'] = data['LON'].apply(lambda x: median_lon if x < -118.6 or x > -1

columns_to_check = ['LAT', 'LON']

plt.figure(figsize=(15, 10))

for idx, col in enumerate(columns_to_check, 1):
    plt.subplot(2, len(columns_to_check), idx)
    sns.boxplot(data[col])
    plt.title(f"Updated Boxplot - {col}")

    plt.subplot(2, len(columns_to_check), idx + len(columns_to_check))
    sns.histplot(data[col], bins=50, kde=True)
    plt.title(f"Updated Histogram - {col}")

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



In [19]:

```
# Check now after handling outliers
data.describe()
```

Out[19]:

	DR_NO	Date Rptd	DATE OCC	TIME OCC	
count	5.766100e+04	57661	57661	57661.000000	57
mean	2.029897e+08	2020-08-12 02:16:04.853193984	2020-07-05 06:07:36.641404160	1347.279686	
min	1.903265e+08	2020-01-01 00:00:00	2020-01-01 00:00:00	1.000000	
25%	2.015102e+08	2020-04-20 00:00:00	2020-04-06 00:00:00	929.000000	
50%	2.018061e+08	2020-07-22 00:00:00	2020-07-06 00:00:00	1426.000000	
75%	2.020104e+08	2020-10-27 00:00:00	2020-10-05 00:00:00	1900.000000	
max	2.514041e+08	2024-12-23 00:00:00	2020-12-31 00:00:00	2359.000000	
std	4.626854e+06	NaN	NaN	645.935760	

8 rows × 21 columns

In [20]:

```
#Standardizing and Normalizing numerical data
from sklearn.preprocessing import StandardScaler, MinMaxScaler

# Initialize the scalers
```

```
standard_scaler = StandardScaler()
minmax_scaler = MinMaxScaler()

# Standardize 'Vict Age'
data['Vict Age'] = standard_scaler.fit_transform(data[['Vict Age']])

# Normalize 'TIME OCC', 'LAT', and 'LON'
data['TIME OCC'] = minmax_scaler.fit_transform(data[['TIME OCC']])
data['LAT'] = minmax_scaler.fit_transform(data[['LAT']])
data['LON'] = minmax_scaler.fit_transform(data[['LON']])

print(data[['Vict Age', 'TIME OCC', 'LAT', 'LON']].head())
```

```
   Vict Age  TIME OCC       LAT       LON
0  0.043679  0.357930  0.727506  0.429601
1  0.089449  0.782019  0.696970  0.404596
2 -0.002090  0.525445  0.662937  0.335661
3  0.775987  0.555131  0.311422  0.507321
4  1.508295  0.775657  0.188811  0.371480
```

In [21]:

```
#Encode Area Name alone using LabelEncoder
from sklearn.preprocessing import LabelEncoder

# Columns for one-hot encoding
one_hot_columns = ['AREA NAME', 'Crm Cd Desc', 'Vict Sex', 'Vict Descent', '

# Columns for label encoding
label_columns = ['Mocodes', 'LOCATION', 'Cross Street']

# Perform one-hot encoding
data_encoded = pd.get_dummies(data, columns=one_hot_columns, drop_first=True

# Perform label encoding
label_encoders = {}
for col in label_columns:
    le = LabelEncoder()
    data_encoded[col] = le.fit_transform(data_encoded[col])
    label_encoders[col] = le
print(data_encoded.head())
```

```
       DR_NO  Date Rptd   DATE OCC  TIME OCC  AREA  Rpt Dist No  Part 1-2  \
0  211507896 2021-04-11 2020-11-07  0.357930    15         1502         2
1  201516622 2020-10-21 2020-10-18  0.782019    15         1521         1
2  240913563 2024-12-10 2020-10-30  0.525445     9          933         2
3  210704711 2020-12-24 2020-12-24  0.555131     7          782         1
4  201418201 2020-10-03 2020-09-29  0.775657    14         1454         1

   Crm Cd  Mocodes  Vict Age  ...  Weapon Desc_VEHICLE  \
0     354     5060  0.043679  ...                False
1     230     6159  0.089449  ...                False
2     354     5060 -0.002090  ...                False
3     331     2703  0.775987  ...                False
4     420    14550  1.508295  ...                False

   Weapon Desc_VERBAL THREAT  Status_AO  Status_IC  Status_JA  Status_JO  \
0                      False      False       True      False      False
1                      False      False       True      False      False
2                      False      False       True      False      False
3                      False      False       True      False      False
4                      False      False       True      False      False

   Status Desc_Adult Other  Status Desc_Invest Cont  Status Desc_Juv Arrest
\
0                    False                     True                   False
1                    False                     True                   False
2                    False                     True                   False
3                    False                     True                   False
4                    False                     True                   False

   Status Desc_Juv Other
0                  False
1                  False
2                  False
3                  False
4                  False

[5 rows x 496 columns]
```

```
In [22]: print(data.head)
```

```
<bound method NDFrame.head of            DR_NO  Date Rptd   DATE OCC  TIME O
CC  AREA    AREA NAME  \
0      211507896 2021-04-11 2020-11-07  0.357930    15  N Hollywood
1      201516622 2020-10-21 2020-10-18  0.782019    15  N Hollywood
2      240913563 2024-12-10 2020-10-30  0.525445     9     Van Nuys
3      210704711 2020-12-24 2020-12-24  0.555131     7     Wilshire
4      201418201 2020-10-03 2020-09-29  0.775657    14      Pacific
...          ...        ...        ...       ...   ...          ...
57685  201410525 2020-05-06 2020-04-20  0.508482    14      Pacific
57686  201715734 2020-11-25 2020-11-25  0.678117    17   Devonshire
57687  201415109 2020-08-07 2020-08-07  0.478796    14      Pacific
57688  201406736 2020-02-16 2020-02-16  0.483885    14      Pacific
57689  201320688 2020-12-01 2020-12-01  0.559372    13       Newton

       Rpt Dist No  Part 1-2  Crm Cd  \
0             1502         2     354
1             1521         1     230
2              933         2     354
3              782         1     331
4             1454         1     420
...            ...       ...     ...
57685         1488         2     649
57686         1709         1     510
57687         1431         2     624
57688         1453         1     341
57689         1381         1     210

                                             Crm Cd Desc  ...  \
0                                      THEFT OF IDENTITY  ...
1         ASSAULT WITH DEADLY WEAPON, AGGRAVATED ASSAULT  ...
2                                      THEFT OF IDENTITY  ...
3      THEFT FROM MOTOR VEHICLE - GRAND ($950.01 AND ...  ...
4        THEFT FROM MOTOR VEHICLE - PETTY ($950 & UNDER)  ...
...                                                  ...  ...
57685                   DOCUMENT FORGERY / STOLEN FELONY  ...
57686                                   VEHICLE - STOLEN  ...
57687                           BATTERY - SIMPLE ASSAULT  ...
57688  THEFT-GRAND ($950.01 & OVER)EXCPT,GUNS,FOWL,LI...  ...
57689                                            ROBBERY  ...

                                       LOCATION      Cross Street       LAT
\
0       7800    BEEMAN                       AV           Unknown  0.727506
1               ATOLL                        AV          N  GAULT  0.696970
2      14600    SYLVAN                       ST           Unknown  0.662937
3       6000    COMEY                        AV           Unknown  0.311422
4                       4700    LA VILLA MARINA           Unknown  0.188811
...                                         ...               ...       ...
57685   8600    BELFORD                      AV           Unknown  0.138928
57686  15700    MIDWOOD                      DR           Unknown  0.888811
57687                                      17TH  OCEAN FRONT WALK  0.201166
57688   4200    GLENCOE                      AV           Unknown  0.206294
57689           59TH                         ST          BROADWAY  0.199767

            LON  Year Month  Day DayOfWeek    DayName  MonthName
0      0.429601  2020    11    7         5   Saturday   November
```

```
1      0.404596  2020    10   18         6     Sunday    October
2      0.335661  2020    10   30         4     Friday    October
3      0.507321  2020    12   24         3   Thursday   December
4      0.371480  2020     9   29         1    Tuesday  September
...         ...   ...   ...  ...       ...        ...        ...
57685  0.486596  2020     4   20         0     Monday      April
57686  0.287452  2020    11   25         2  Wednesday   November
57687  0.286326  2020     8    7         4     Friday     August
57688  0.356387  2020     2   16         6     Sunday   February
57689  0.724487  2020    12    1         1    Tuesday   December

[57661 rows x 34 columns]>
```

EXPLORATORY DATA ANALYSIS

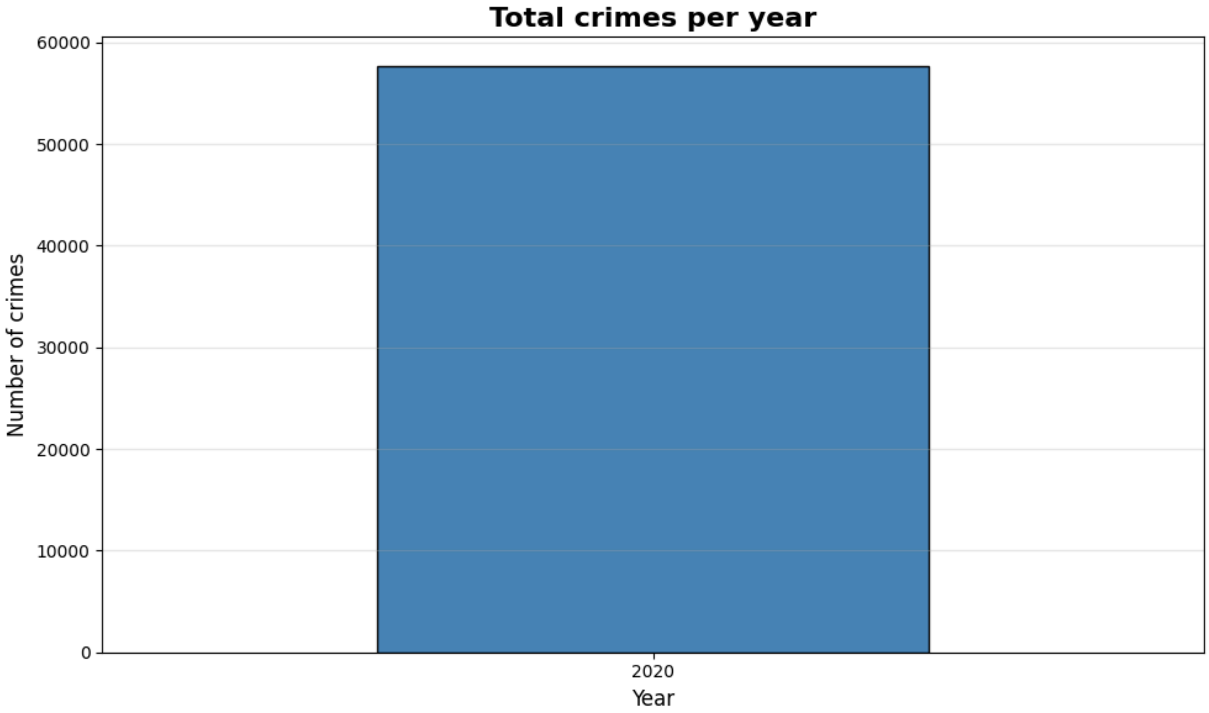
In [23]:

```python
import matplotlib.pyplot as plt
import seaborn as sns
```

In [24]:

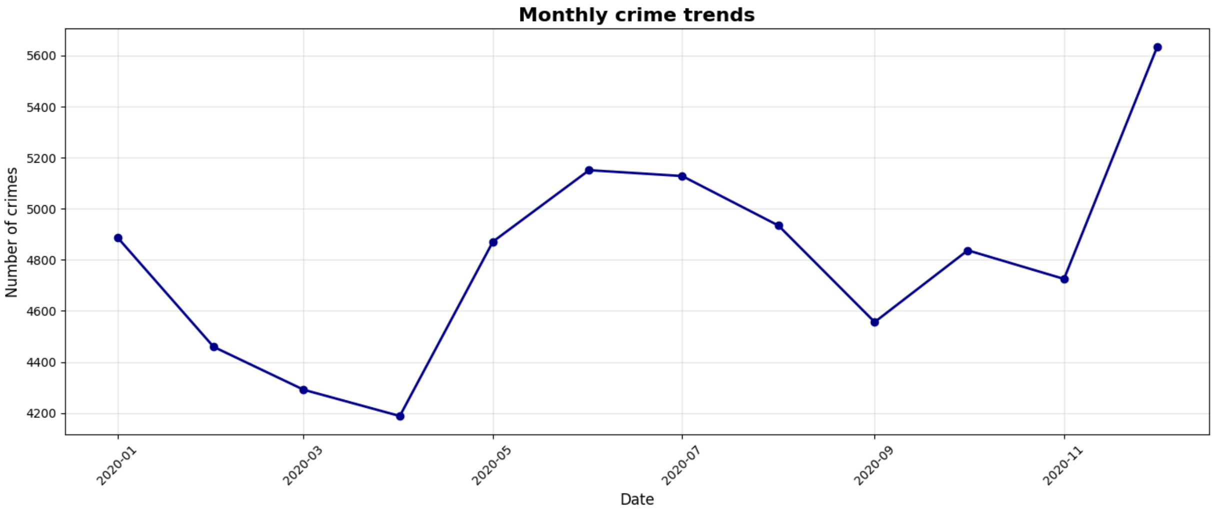
```python
# Overall crime trends from 2020 to present year
print("\nOverall crime trends")
crimes_per_year = data['Year'].value_counts().sort_index()
print("\nCrimes per year:")
print(crimes_per_year)

plt.figure(figsize=(10, 6))
crimes_per_year.plot(kind='bar', color='steelblue', edgecolor='black')
plt.title('Total crimes per year', fontsize=16, fontweight='bold')
plt.xlabel('Year', fontsize=12)
plt.ylabel('Number of crimes', fontsize=12)
plt.xticks(rotation=0)
plt.grid(axis='y', alpha=0.3)
plt.tight_layout()
plt.savefig('crime_trends_yearly.png', dpi=300)
plt.show()
```

```
Overall crime trends

Crimes per year:
Year
2020    57661
Name: count, dtype: int64
```



In [25]:

```python
# Overall monthly crime trends

monthly_crimes = data.groupby(data['DATE OCC'].dt.to_period('M')).size()
monthly_crimes.index = monthly_crimes.index.to_timestamp()

plt.figure(figsize=(14, 6))
plt.plot(monthly_crimes.index, monthly_crimes.values, linewidth=2, color='da
plt.title('Monthly crime trends', fontsize=16, fontweight='bold')
plt.xlabel('Date', fontsize=12)
plt.ylabel('Number of crimes', fontsize=12)
plt.grid(True, alpha=0.3)
plt.xticks(rotation=45)
plt.tight_layout()
plt.show()
```

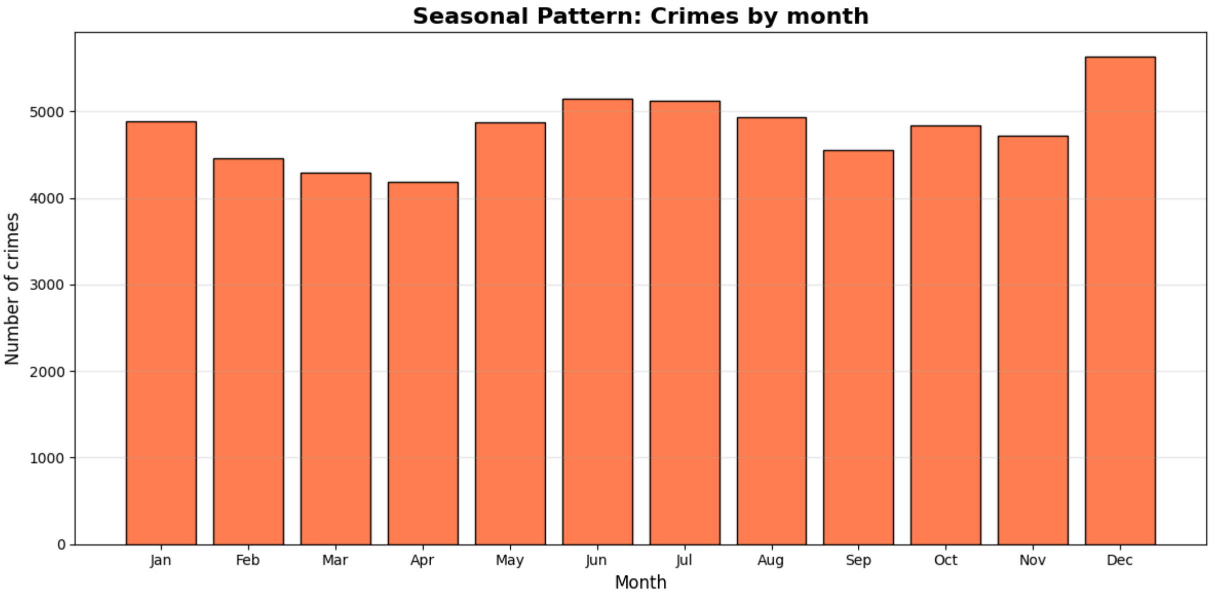


In [26]:

```python
# Analyze and Visualize Seasonal data
print("\nSeasonal Patterns")
```

```
monthly_avg = data.groupby('Month').size()
month_names = ['Jan', 'Feb', 'Mar', 'Apr', 'May', 'Jun', 'Jul', 'Aug', 'Sep'

print(monthly_avg)

plt.figure(figsize=(12, 6))
plt.bar(range(1, 13), monthly_avg.values, color='coral', edgecolor='black')
plt.title('Seasonal Pattern: Crimes by month', fontsize=16, fontweight='bold
plt.xlabel('Month', fontsize=12)
plt.ylabel('Number of crimes', fontsize=12)
plt.xticks(range(1, 13), month_names)
plt.grid(axis='y', alpha=0.3)
plt.tight_layout()
plt.show()
```

```
Seasonal Patterns
Month
1     4888
2     4459
3     4291
4     4188
5     4871
6     5151
7     5128
8     4934
9     4556
10    4837
11    4725
12    5633
dtype: int64
```

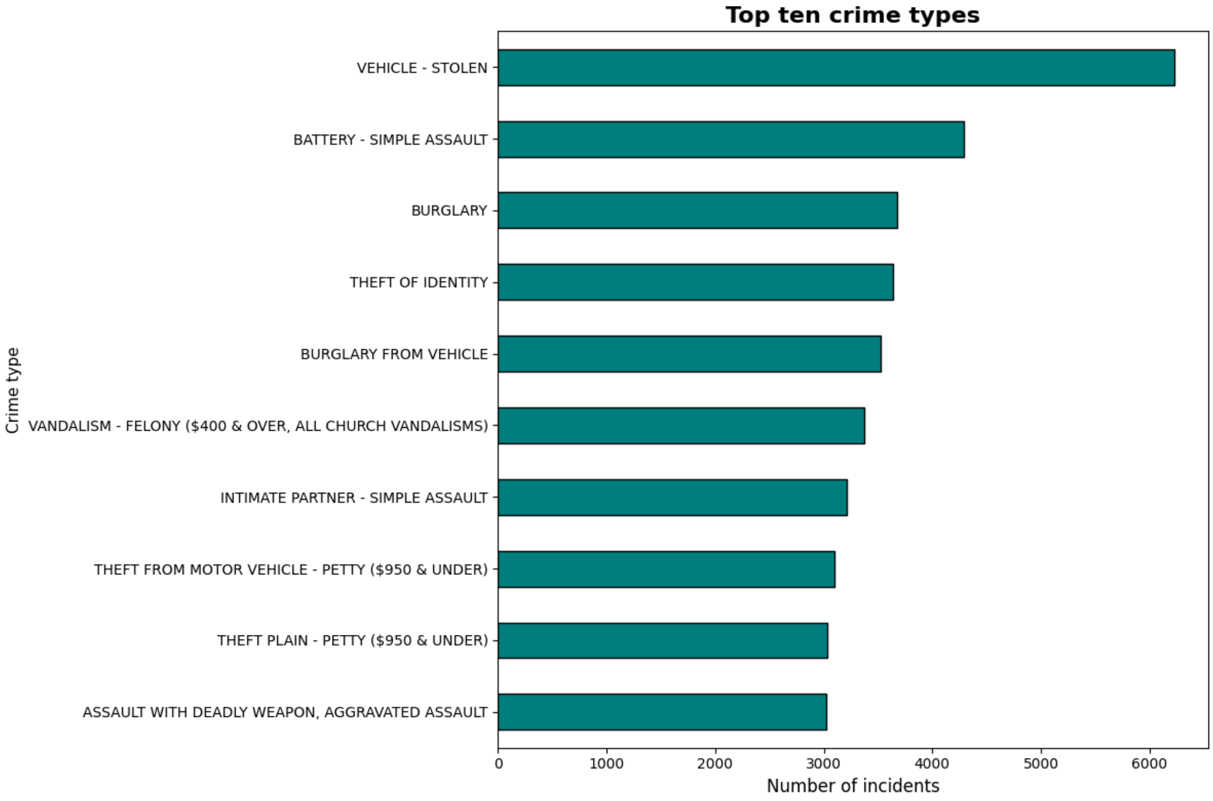


In [27]:

```
#Most common type of crimes and its trends over the time
print("\nMost common crime type and trends")

top_crimes = data['Crm Cd Desc'].value_counts().head(10)
print("\nTop ten crime types:")
print(top_crimes)
```

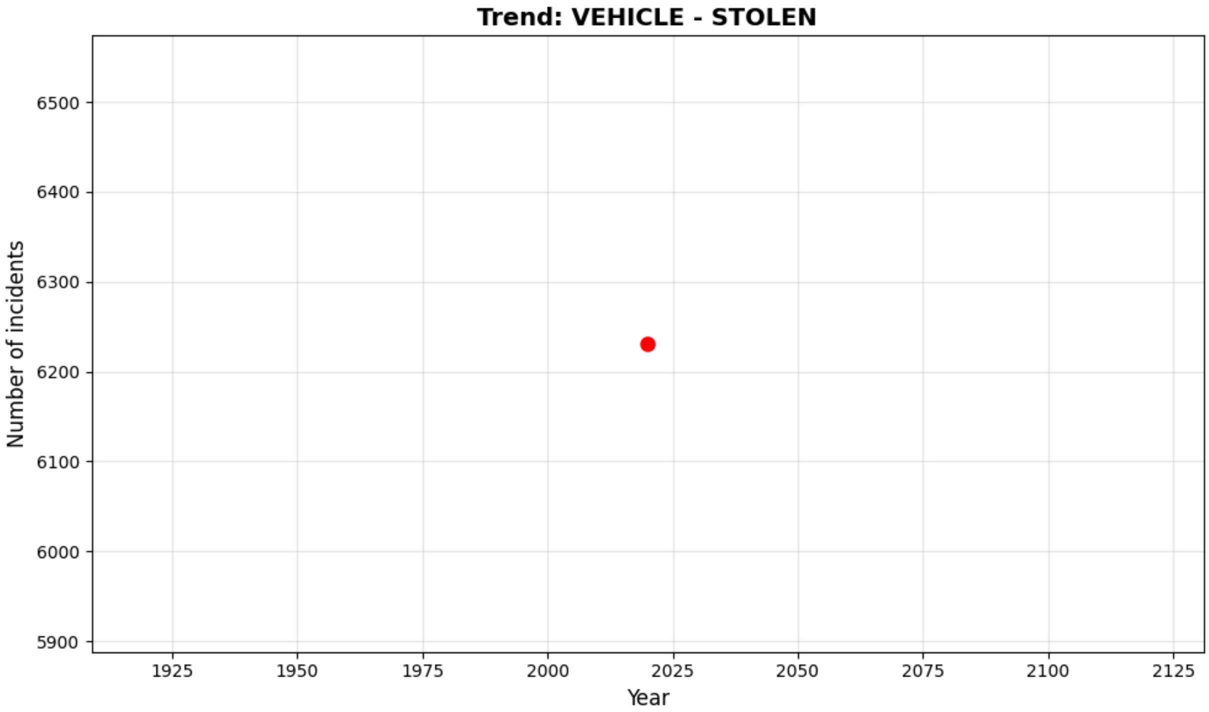
```
plt.figure(figsize=(12, 8))
top_crimes.plot(kind='barh', color='teal', edgecolor='black')
plt.title('Top ten crime types', fontsize=16, fontweight='bold')
plt.xlabel('Number of incidents', fontsize=12)
plt.ylabel('Crime type', fontsize=11)
plt.gca().invert_yaxis()
plt.tight_layout()
plt.show()
```

```
Most common crime type and trends

Top ten crime types:
Crm Cd Desc
VEHICLE - STOLEN                                           6231
BATTERY - SIMPLE ASSAULT                                   4290
BURGLARY                                                   3675
THEFT OF IDENTITY                                          3635
BURGLARY FROM VEHICLE                                      3526
VANDALISM - FELONY ($400 & OVER, ALL CHURCH VANDALISMS)    3369
INTIMATE PARTNER - SIMPLE ASSAULT                          3213
THEFT FROM MOTOR VEHICLE - PETTY ($950 & UNDER)            3103
THEFT PLAIN - PETTY ($950 & UNDER)                         3030
ASSAULT WITH DEADLY WEAPON, AGGRAVATED ASSAULT             3027
Name: count, dtype: int64
```



In [28]:

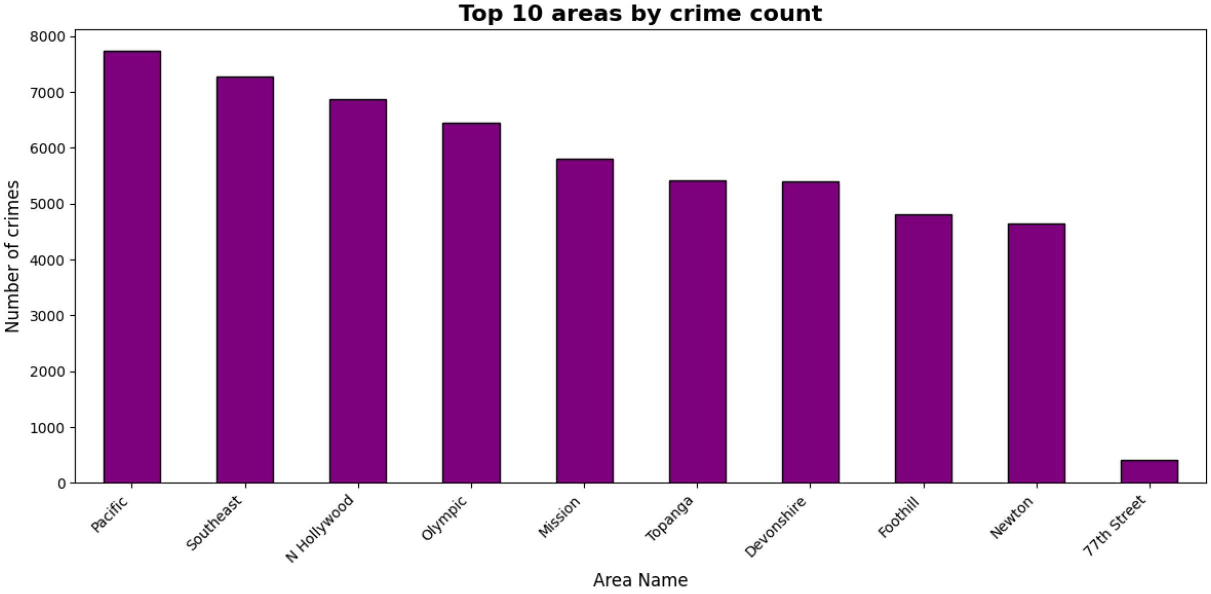
```
# Trend of the most common crime
most_common = top_crimes.index[0]
crime_trend = data[data['Crm Cd Desc'] == most_common].groupby('Year').size(

plt.figure(figsize=(10, 6))
crime_trend.plot(kind='line', marker='o', color='red', linewidth=2, markersi
plt.title(f'Trend: {most_common}', fontsize=14, fontweight='bold')
plt.xlabel('Year', fontsize=12)
plt.ylabel('Number of incidents', fontsize=12)
plt.grid(True, alpha=0.3)
plt.tight_layout()
plt.savefig('most_common_crime_trend.png', dpi=300)
plt.show()
```



In [29]:

```
# Most notable difference in crime rate between regions and cities
print("\nRegional differences")

area_crimes = data['AREA NAME'].value_counts()
print("\nTop 10 areas:")
print(area_crimes.head(10))

plt.figure(figsize=(12, 6))
area_crimes.head(10).plot(kind='bar', color='purple', edgecolor='black')
plt.title('Top 10 areas by crime count', fontsize=16, fontweight='bold')
plt.xlabel('Area Name', fontsize=12)
plt.ylabel('Number of crimes', fontsize=12)
plt.xticks(rotation=45, ha='right')
plt.tight_layout()
plt.savefig('regional_differences.png', dpi=300)
plt.show()
```

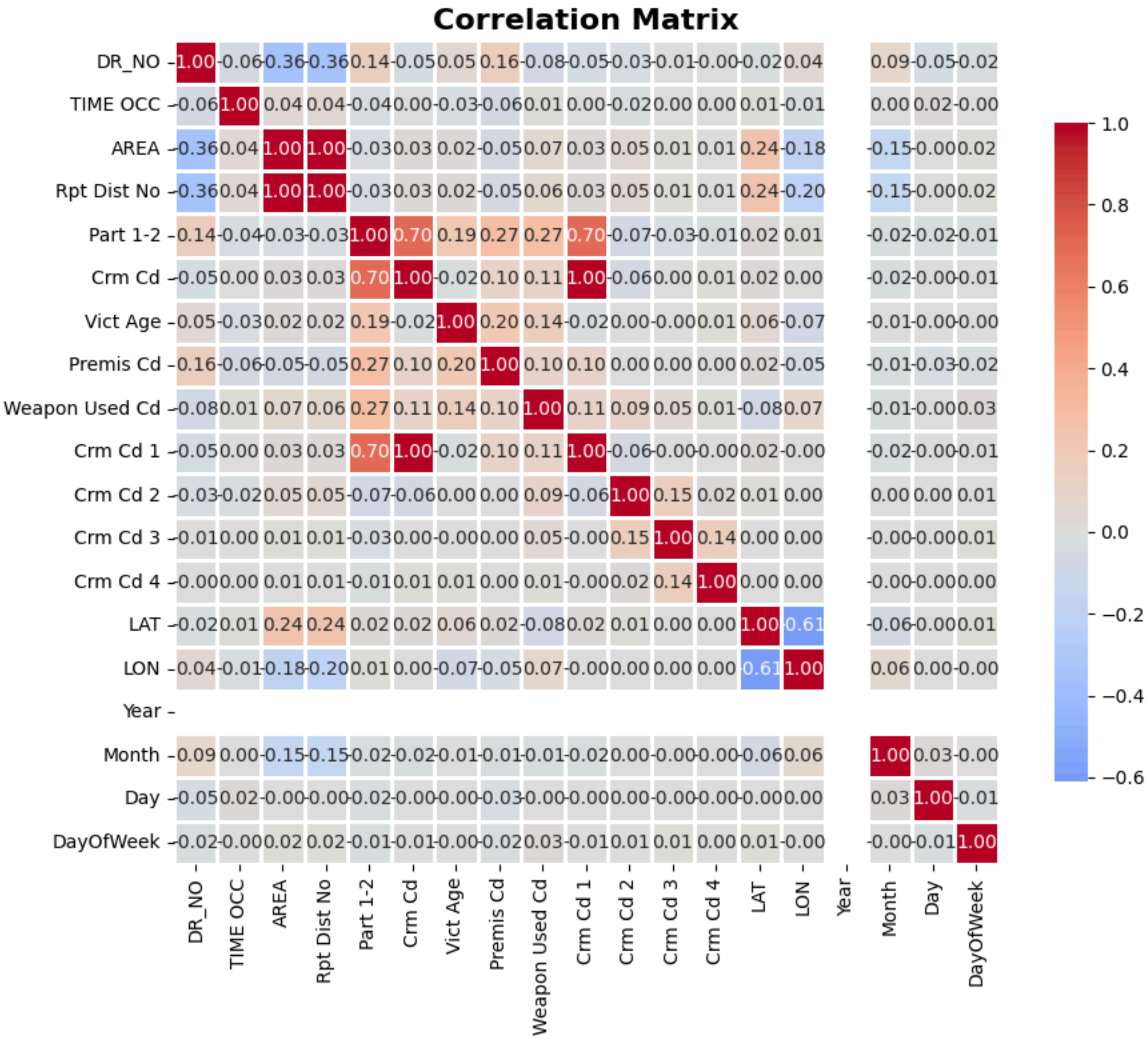
```
Regional differences

Top 10 areas:
AREA NAME
Pacific        7732
Southeast      7283
N Hollywood    6874
Olympic        6452
Mission        5805
Topanga        5423
Devonshire     5406
Foothill       4819
Newton         4644
77th Street     417
Name: count, dtype: int64
```



In [30]:

```
# Explore correlation between economic factor and crime rate
import numpy as np
print("\nCorrelation with Economic Factors")
print("Economic data not available in this dataset.")
print("External economic data needed for this analysis.")

# Correlation with available numerical data
numeric_data = data.select_dtypes(include=[np.number])
correlation = numeric_data.corr()

plt.figure(figsize=(10, 8))
sns.heatmap(correlation, annot=True, fmt='.2f', cmap='coolwarm', center=0,
            square=True, linewidths=1, cbar_kws={"shrink": 0.8})
plt.title('Correlation Matrix', fontsize=16, fontweight='bold')
plt.tight_layout()
plt.savefig('correlation.png', dpi=300)
plt.show()
```

```
Correlation with Economic Factors
Economic data not available in this dataset.
External economic data needed for this analysis.
```

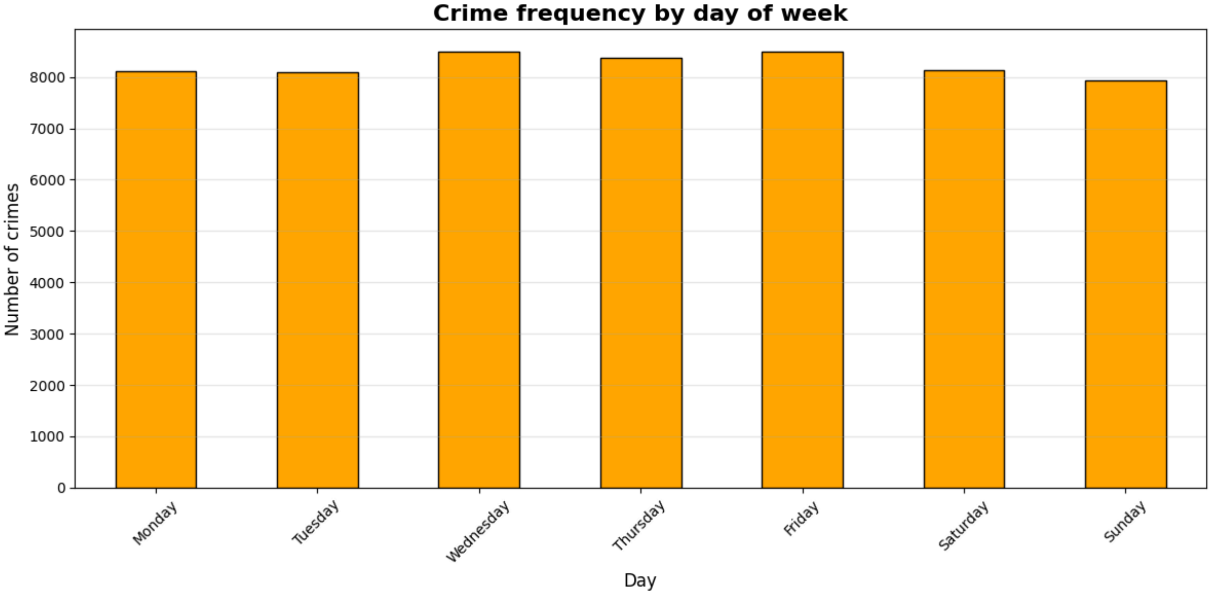


	DR_NO	TIME OCC	AREA	Rpt Dist No	Part 1-2	Crm Cd	Vict Age	Premis Cd	Weapon Used Cd	Crm Cd 1	Crm Cd 2	Crm Cd 3	Crm Cd 4	LAT	LON	Year	Month	Day	DayOfWeek
DR_NO	1.00	-0.06	-0.36	-0.36	0.14	-0.05	0.05	0.16	-0.08	-0.05	-0.03	-0.01	-0.00	-0.02	0.04		0.09	-0.05	-0.02
TIME OCC	-0.06	1.00	0.04	0.04	-0.04	0.00	-0.03	-0.06	0.01	0.00	-0.02	0.00	0.00	0.01	-0.01		0.00	0.02	-0.00
AREA	-0.36	0.04	1.00	1.00	-0.03	0.03	0.02	-0.05	0.07	0.03	0.05	0.01	0.01	0.24	-0.18		-0.15	-0.00	0.02
Rpt Dist No	-0.36	0.04	1.00	1.00	-0.03	0.03	0.02	-0.05	0.06	0.03	0.05	0.01	0.01	0.24	-0.20		-0.15	-0.00	0.02
Part 1-2	0.14	-0.04	-0.03	-0.03	1.00	0.70	0.19	0.27	0.27	0.70	-0.07	-0.03	-0.01	0.02	0.01		-0.02	-0.02	-0.01
Crm Cd	-0.05	0.00	0.03	0.03	0.70	1.00	-0.02	0.10	0.11	1.00	-0.06	0.00	0.01	0.02	0.00		-0.02	-0.00	-0.01
Vict Age	0.05	-0.03	0.02	0.02	0.19	-0.02	1.00	0.20	0.14	-0.02	0.00	-0.00	0.01	0.06	-0.07		-0.01	-0.00	-0.00
Premis Cd	0.16	-0.06	-0.05	-0.05	0.27	0.10	0.20	1.00	0.10	0.10	0.00	0.00	0.00	0.02	-0.05		-0.01	-0.03	-0.02
Weapon Used Cd	-0.08	0.01	0.07	0.06	0.27	0.11	0.14	0.10	1.00	0.11	0.09	0.05	0.01	-0.08	0.07		-0.01	-0.00	0.03
Crm Cd 1	-0.05	0.00	0.03	0.03	0.70	1.00	-0.02	0.10	0.11	1.00	-0.06	-0.00	-0.00	0.02	-0.00		-0.02	-0.00	-0.01
Crm Cd 2	-0.03	-0.02	0.05	0.05	-0.07	-0.06	0.00	0.00	0.09	-0.06	1.00	0.15	0.02	0.01	0.00		0.00	0.00	0.01
Crm Cd 3	-0.01	0.00	0.01	0.01	-0.03	0.00	-0.00	0.00	0.05	-0.00	0.15	1.00	0.14	0.00	0.00		-0.00	-0.00	0.01
Crm Cd 4	-0.00	0.00	0.01	0.01	-0.01	0.01	0.01	0.00	0.01	-0.00	0.02	0.14	1.00	0.00	0.00		-0.00	-0.00	0.00
LAT	-0.02	0.01	0.24	0.24	0.02	0.02	0.06	0.02	-0.08	0.02	0.01	0.00	0.00	1.00	-0.61		-0.06	-0.00	0.01
LON	0.04	-0.01	-0.18	-0.20	0.01	0.00	-0.07	-0.05	0.07	-0.00	0.00	0.00	0.00	0.61	1.00		0.06	0.00	-0.00
Year																			
Month	0.09	0.00	-0.15	-0.15	-0.02	-0.02	-0.01	-0.01	-0.01	-0.02	0.00	-0.00	-0.00	-0.06	0.06		1.00	0.03	-0.00
Day	-0.05	0.02	-0.00	-0.00	-0.02	-0.00	-0.00	-0.03	-0.00	-0.00	0.00	-0.00	-0.00	-0.00	0.00		0.03	1.00	-0.01
DayOfWeek	-0.02	-0.00	0.02	0.02	-0.01	-0.01	-0.00	-0.02	0.03	-0.01	0.01	0.01	0.00	0.01	-0.00		-0.00	-0.01	1.00

In [31]:

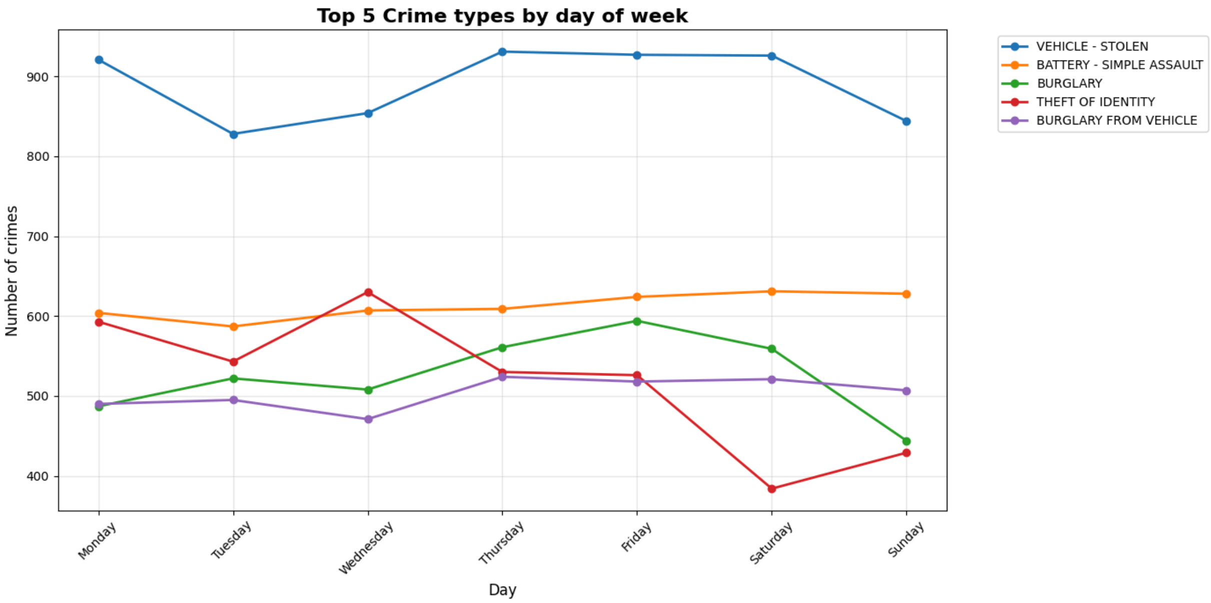
```
# Analyze the relationship between day of the week and frequency of types of
print("\Day of Week Analysis")

day_order = ['Monday', 'Tuesday', 'Wednesday', 'Thursday', 'Friday', 'Saturd
day_crimes = data['DayName'].value_counts().reindex(day_order)

print("\nCrimes by day:")
print(day_crimes)

plt.figure(figsize=(12, 6))
day_crimes.plot(kind='bar', color='orange', edgecolor='black')
plt.title('Crime frequency by day of week', fontsize=16, fontweight='bold')
plt.xlabel('Day', fontsize=12)
plt.ylabel('Number of crimes', fontsize=12)
plt.xticks(rotation=45)
plt.grid(axis='y', alpha=0.3)
plt.tight_layout()
plt.show()
```

```
\Day of Week Analysis

Crimes by day:
DayName
Monday       8105
Tuesday      8102
Wednesday    8502
Thursday     8380
Friday       8503
Saturday     8139
Sunday       7930
Name: count, dtype: int64
```



In [32]:

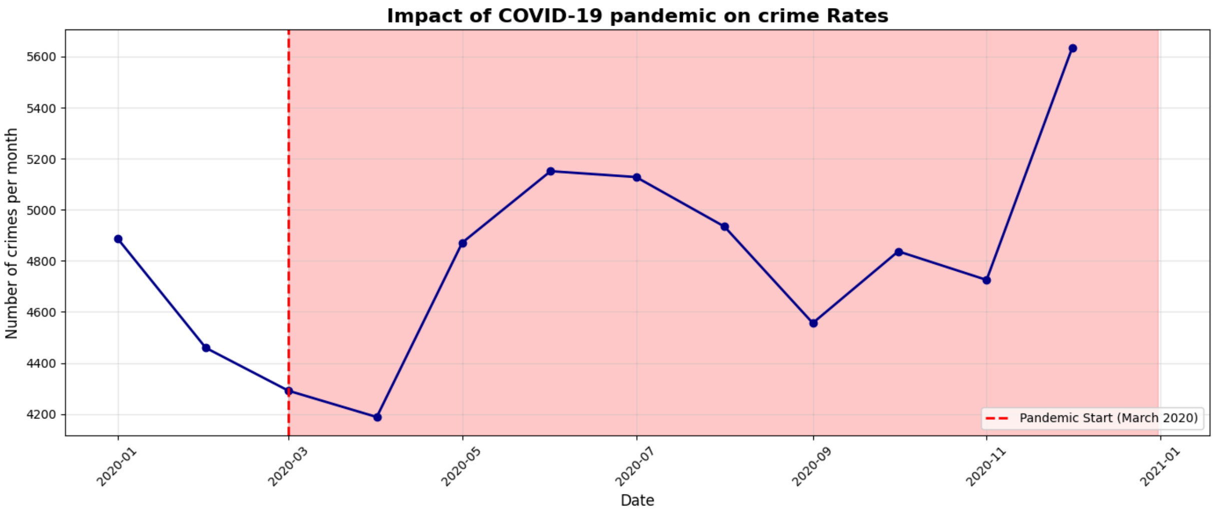
```
# Crime types by day of the week
top_5_crimes = data['Crm Cd Desc'].value_counts().head(5).index

plt.figure(figsize=(14, 7))
for crime in top_5_crimes:
    crime_by_day = data[data['Crm Cd Desc'] == crime]['DayName'].value_count
    plt.plot(day_order, crime_by_day.values, marker='o', linewidth=2, label=

plt.title('Top 5 Crime types by day of week', fontsize=16, fontweight='bold'
plt.xlabel('Day', fontsize=12)
plt.ylabel('Number of crimes', fontsize=12)
plt.legend(bbox_to_anchor=(1.05, 1), loc='upper left')
plt.xticks(rotation=45)
plt.grid(True, alpha=0.3)
plt.tight_layout()
plt.show()
```



In [33]:

```
# Impact of significant event or policy changes on crime rate
print("\Impact of Significant Events")

# COVID-19 pandemic impact
pandemic_start = pd.to_datetime('2020-03-01')

monthly_trend = data.groupby(data['DATE OCC'].dt.to_period('M')).size()
monthly_trend.index = monthly_trend.index.to_timestamp()

plt.figure(figsize=(14, 6))
plt.plot(monthly_trend.index, monthly_trend.values, linewidth=2, color='dark
plt.axvline(x=pandemic_start, color='red', linestyle='--', linewidth=2, labe
plt.axvspan(pandemic_start, pd.to_datetime('2020-12-31'), alpha=0.2, color='
plt.title('Impact of COVID-19 pandemic on crime Rates', fontsize=16, fontwei
plt.xlabel('Date', fontsize=12)
plt.ylabel('Number of crimes per month', fontsize=12)
plt.legend()
plt.grid(True, alpha=0.3)
plt.xticks(rotation=45)
plt.tight_layout()
plt.show()
```

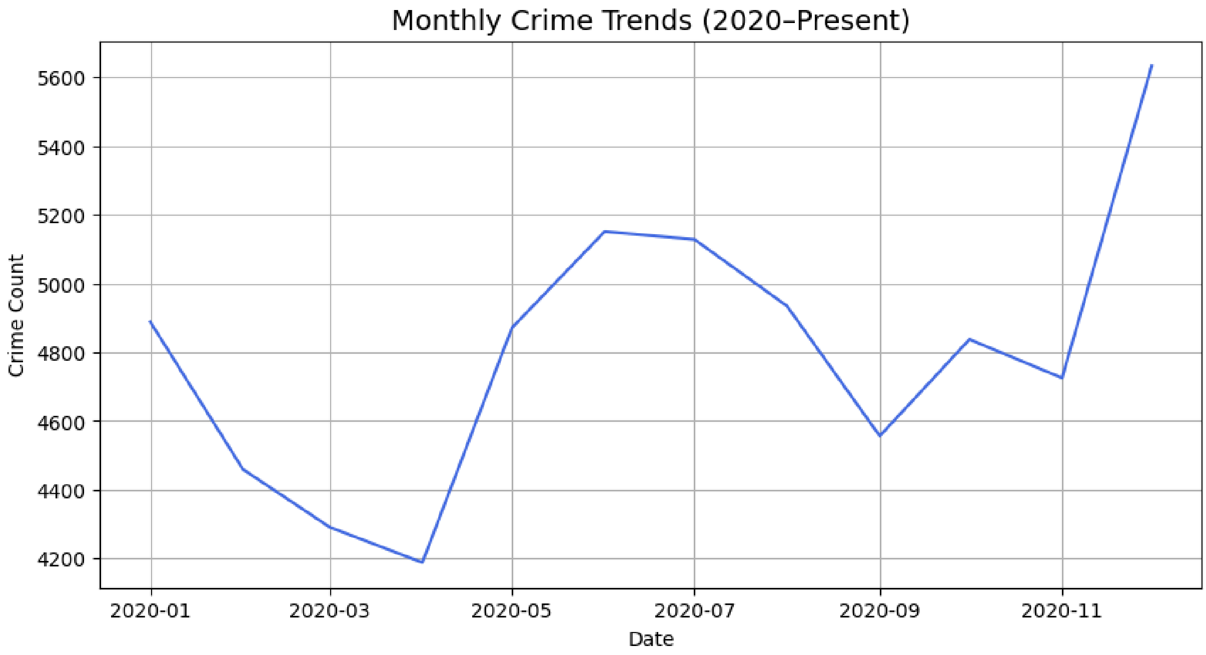
```
\Impact of Significant Events
```

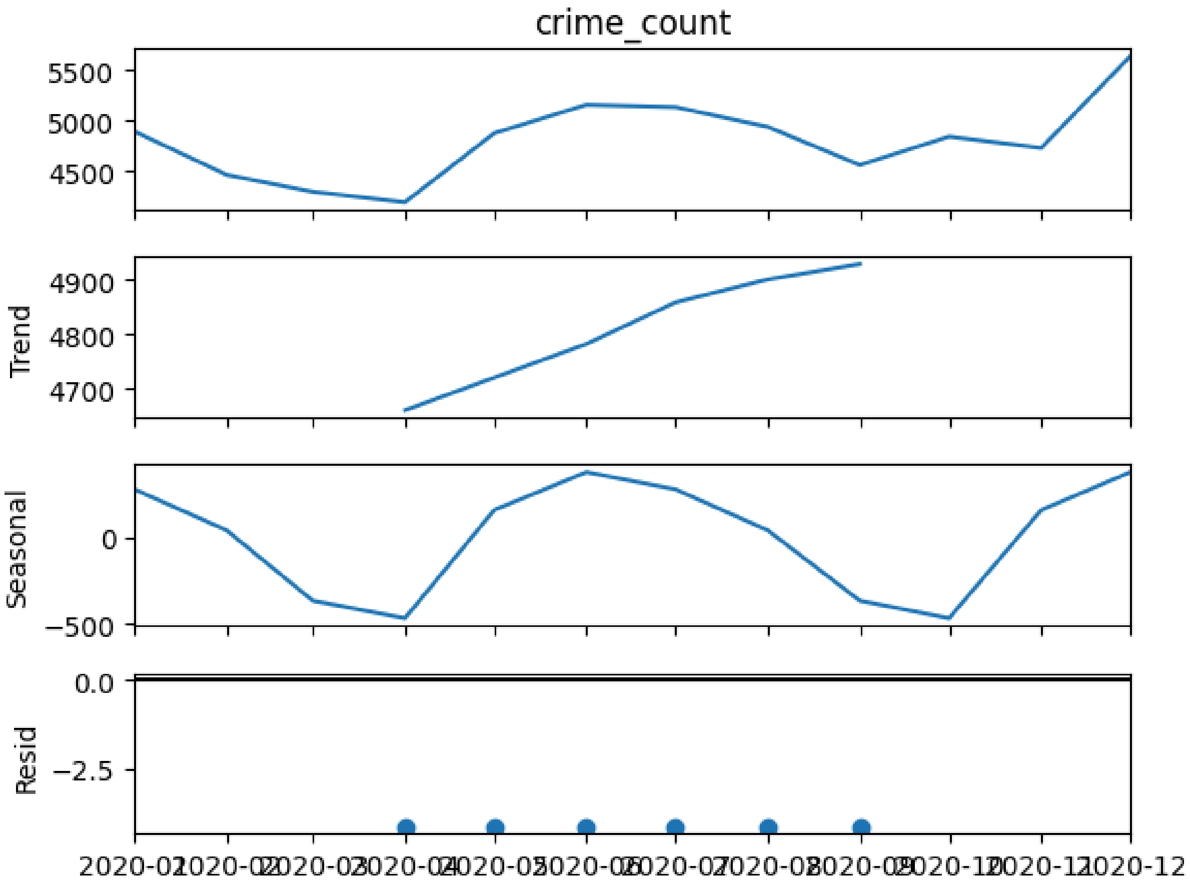


In [36]:

```
# TIME SERIES ANALYSIS & FORECASTING
#Import required libraries
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from statsmodels.tsa.seasonal import seasonal_decompose
from statsmodels.tsa.stattools import adfuller
from statsmodels.tsa.arima.model import ARIMA

df = data.copy()

# Data Preprocessing
df['DATE OCC'] = pd.to_datetime(df['DATE OCC'], errors='coerce')
df = df.dropna(subset=['DATE OCC'])
# Group by month to get total crimes per month
df['year_month'] = df['DATE OCC'].dt.to_period('M').dt.to_timestamp()
monthly_crime = df.groupby('year_month').size().rename('crime_count').reset_
monthly_crime = monthly_crime.sort_values('year_month').set_index('year_mont

# Trend Visualizing
plt.figure(figsize=(10, 5))
plt.plot(monthly_crime.index, monthly_crime['crime_count'], color='royalblue
plt.title("Monthly Crime Trends (2020–Present)", fontsize=14)
plt.xlabel("Date")
plt.ylabel("Crime Count")
plt.grid(True)
plt.show()

# Decompose the time series data
decomposition = seasonal_decompose(monthly_crime['crime_count'], model='addi
decomposition.plot()
plt.show()

# Statistical Testing
result = adfuller(monthly_crime['crime_count'])
print("ADF Statistic:", result[0])
print("p-value:", result[1])
#Confidence interval assumed to be 95%
if result[1] < 0.05:
    print("The series is stationary") # Good for ARIMA
```

```
else:
    print("The series is not stationary ")

#Data Modeling (ARIMA Model)
# Using ARIMA(1,1,1) as a baseline
model = ARIMA(monthly_crime['crime_count'], order=(1,1,1))
fitted_model = model.fit()
print(fitted_model.summary())

#Forecast the next 12 months
forecast_steps = 12
forecast = fitted_model.get_forecast(steps=forecast_steps)

forecast_index = pd.date_range(
    start=monthly_crime.index[-1] + pd.offsets.MonthBegin(1),
    periods=forecast_steps,
    freq='MS'
)

forecast_mean = forecast.predicted_mean
conf_int = forecast.conf_int()
lower_ci = conf_int.iloc[:, 0]
upper_ci = conf_int.iloc[:, 1]

# Plotting of Forecasting Result
plt.figure(figsize=(10, 5))
plt.plot(monthly_crime.index, monthly_crime['crime_count'], label='Historica
plt.plot(forecast_index, forecast_mean, label='Forecast', color='orange')
plt.fill_between(forecast_index, lower_ci, upper_ci, color='orange', alpha=0
plt.title("Crime Count Forecast (Next 12 Months)")
plt.xlabel("Date")
plt.ylabel("Predicted Crime Count")
plt.legend()
plt.grid(True)
plt.show()

#Storing Forecast Result
forecast_df = pd.DataFrame({
    'Date': forecast_index,
    'Forecast': forecast_mean,
    'Lower_CI': lower_ci,
    'Upper_CI': upper_ci
})

forecast_df.to_csv("crime_forecast_next_12_months.csv", index=False)

#Key insight summary
print("\n Summary of Findings:")
print(f"The average monthly crime count in the last year: {monthly_crime['cr
print(f"The forecast suggests {'an increase' if forecast_mean.mean() > month
```

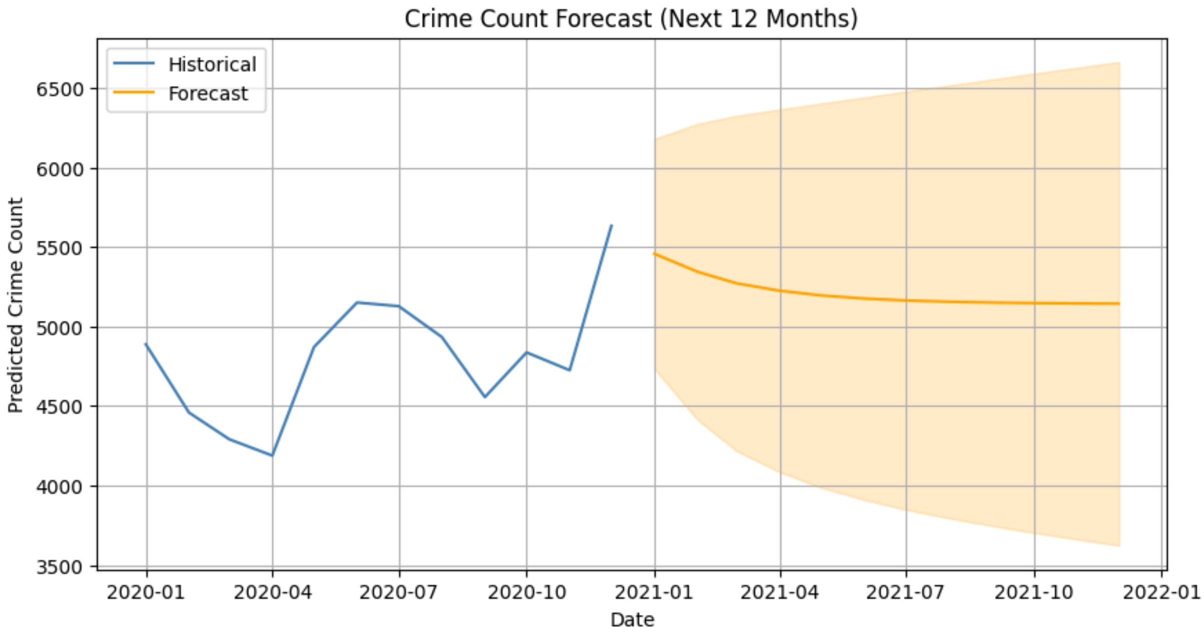




```
ADF Statistic: -0.8936112337799436
p-value: 0.7900690485093879
The series is not stationary
```

```
/usr/local/lib/python3.12/dist-packages/statsmodels/tsa/base/tsa_model.py:47
3: ValueWarning: No frequency information was provided, so inferred frequenc
y MS will be used.
  self._init_dates(dates, freq)
/usr/local/lib/python3.12/dist-packages/statsmodels/tsa/base/tsa_model.py:47
3: ValueWarning: No frequency information was provided, so inferred frequenc
y MS will be used.
  self._init_dates(dates, freq)
/usr/local/lib/python3.12/dist-packages/statsmodels/tsa/base/tsa_model.py:47
3: ValueWarning: No frequency information was provided, so inferred frequenc
y MS will be used.
  self._init_dates(dates, freq)
```

```
                               SARIMAX Results
============================================================================
==
Dep. Variable:            crime_count   No. Observations:
12
Model:                 ARIMA(1, 1, 1)   Log Likelihood                 -81.5
39
Date:                Thu, 16 Oct 2025   AIC                            169.0
78
Time:                        02:23:14   BIC                            170.2
72
Sample:                    01-01-2020   HQIC                           168.3
26
                         - 12-01-2020
Covariance Type:                  opg
============================================================================
==
                 coef    std err          z      P>|z|      [0.025      0.97
5]
----------------------------------------------------------------------------
--
ar.L1          0.6426      1.263      0.509      0.611      -1.833       3.1
18
ma.L1         -0.8360      1.215     -0.688      0.491      -3.217       1.5
45
sigma2      1.362e+05   7.19e+04      1.894      0.058   -4743.906    2.77e+
05
============================================================================
=======
Ljung-Box (L1) (Q):                   0.01   Jarque-Bera (JB):
1.26
Prob(Q):                              0.93   Prob(JB):
0.53
Heteroskedasticity (H):               1.79   Skew:
0.80
Prob(H) (two-sided):                  0.59   Kurtosis:
2.52
============================================================================
=======

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



```
 Summary of Findings:
The average monthly crime count in the last year: 4805.08
The forecast suggests an increase in future crime trends.
```