Los Angeles - Crime Analysis

Project Report 1

Group Number 17

1. Janani Karthikeyan (002830003)
2. Milan Gurumurthy (002833029)
3. Prathyusha Adhikam (002835277)
4. Saathvika Kethineni (002893814)
5. Shreyas Sreenivas (002825934)

Submitted to: Si

varit Sultornsanee

Submitted Date:

November 3rd, 2023

In [1]:
```
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
```

In [2]:
```
#Loading the dataset
df = pd.read_csv(r"D:\Documents\Prof_Docs\FDA\Projects\Project 1\Crime_Data_from_2020_to_Present.csv")
```

In [3]:
```
#displaying fthe datset
pd.set_option('display.max_columns', None)
df.head(5)
```

Out[3]:

	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
0	10304468	01/08/2020 12:00:00 AM	01/08/2020 12:00:00 AM	2230	3	Southwest	377	2	624	BATTERY - SIMPLE ASSAULT	0444 0913	36	F	B	501.(
1	190101086	01/02/2020 12:00:00 AM	01/01/2020 12:00:00 AM	330	1	Central	163	2	624	BATTERY - SIMPLE ASSAULT	0416 1822 1414	25	M	H	102.(
2	200110444	04/14/2020 12:00:00 AM	02/13/2020 12:00:00 AM	1200	1	Central	155	2	845	SEX OFFENDER REGISTRANT OUT OF COMPLIANCE	1501	0	X	X	726.(
3	191501505	01/01/2020 12:00:00 AM	01/01/2020 12:00:00 AM	1730	15	N Hollywood	1543	2	745	VANDALISM - MISDEAMEANOR ($399 OR UNDER)	0329 1402	76	F	W	502.(
4	191921269	01/01/2020 12:00:00 AM	01/01/2020 12:00:00 AM	415	19	Mission	1998	2	740	VANDALISM - FELONY ($400 & OVER, ALL CHURCH VA...	0329	31	X	X	409.(

In [4]:
```
#Checking the datatypes
df.dtypes
```

Out[4]:
```
DR_NO              int64
Date Rptd         object
DATE OCC          object
TIME OCC           int64
AREA               int64
AREA NAME         object
Rpt Dist No        int64
Part 1-2           int64
Crm Cd             int64
Crm Cd Desc       object
Mocodes           object
Vict Age           int64
Vict Sex          object
Vict Descent      object
Premis Cd        float64
Premis Desc       object
Weapon Used Cd   float64
Weapon Desc       object
Status            object
```

In [5]:
```
#reviewing the columns
df.columns
```

Out[5]:
```
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 [6]:
```
#Checking for missing values
df.isnull().sum()
```

Out[6]:
```
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         114148
Vict Age             0
Vict Sex        108529
Vict Descent    108537
Premis Cd           10
Premis Desc        488
Weapon Used Cd  537498
Weapon Desc     537498
Status               0
```

In [7]:
```
#Checking for duplicated rows
df.duplicated().sum()
```

Out[7]:
```
0
```

In [8]:
```
#Dropping values which are not needed
df.drop(['DR_NO', 'AREA', 'Mocodes', 'Premis Cd', 'Weapon Used Cd', 'Crm Cd 1', 'Weapon Desc', 'Status',
```

In [9]:
```
#Adding X to NaN values as X defines unknown
df["Vict Sex"].fillna("X")
df["Vict Descent"].fillna("X")
```

Out[9]:
```
0         B
1         H
2         X
3         W
4         X
         ..
825207    H
825208    H
825209    B
825210    H
825211    H
Name: Vict Descent, Length: 825212, dtype: object
```

In [10]:
```
#dropping all remaining missing values from the rows
df.dropna(inplace=True)
```

In [11]:
```
df.isnull().sum()
```

Out[11]:
```
Date Rptd       0
DATE OCC        0
TIME OCC        0
AREA NAME       0
Rpt Dist No     0
Part 1-2        0
Crm Cd          0
Crm Cd Desc     0
Vict Age        0
Vict Sex        0
Vict Descent    0
Premis Desc     0
Status Desc     0
LAT             0
LON             0
dtype: int64
```

In [12]:

```
#Mapping the descent
descent_mapping = {
    'A': 'Other Asian',
    'B': 'Black',
    'C': 'Chinese',
    'D': 'Cambodian',
    'F': 'Filipino',
    'G': 'Guamanian',
    'H': 'Hispanic/Latin/Mexican',
    'I': 'American Indian/Alaskan Native',
    'J': 'Japanese',
    'K': 'Korean',
    'L': 'Laotian',
    'O': 'Other',
    'P': 'Pacific Islander',
    'S': 'Samoan',
    'U': 'Hawaiian',
    'V': 'Vietnamese',
    'W': 'White',
    'X': 'Unknown',
    '-': 'Unknown',
    'Z': 'Asian Indian'
}

df['Vict Descent'].replace(descent_mapping, inplace = True)
```

In [13]:
```
#checking the highest number of victam descent
df['Vict Descent'].value_counts()
```

Out[13]:
```
Hispanic/Latin/Mexican            253094
White                             168047
Black                             117524
Unknown                            79364
Other                              65339
Other Asian                        18050
Korean                              4391
Filipino                            3435
Chinese                             3167
Japanese                            1145
Vietnamese                           851
American Indian/Alaskan Native       772
Asian Indian                         412
Pacific Islander                     219
Hawaiian                             167
Cambodian                             62
Guamanian                             58
Laotian                               50
Samoan                                46
Name: Vict Descent, dtype: int64
```

In [14]:
```
#grouping the victim into age category
groups = [
    (df['Vict Age'] <= 12),
    (df['Vict Age'] >= 13) & (df['Vict Age'] < 18),
    (df['Vict Age'] >= 18) & (df['Vict Age'] < 65),
    (df['Vict Age'] >= 65)
]

labels = ['Child', 'Teen', 'Adult', 'Old']

# create new column 'Age Group'
df['Age Group'] = np.select(groups, labels)
```

In [15]:
```
#Counting the top 5 places of attack
top5_plcaes_of_Attack = df['Premis Desc'].value_counts().head(5)
top5_plcaes_of_Attack
```

Out[15]:
```
SINGLE FAMILY DWELLING                          139736
STREET                                          125890
MULTI-UNIT DWELLING (APARTMENT, DUPLEX, ETC)    101289
PARKING LOT                                      43636
OTHER BUSINESS                                   37095
Name: Premis Desc, dtype: int64
```

In [16]:
```
#checking the type of crimes where 1 is serious and 2 is misdemeanors
df['Part 1-2'].value_counts()
```

Out[16]:
```
1    376708
2    339485
Name: Part 1-2, dtype: int64
```

In [17]:
```
#Converting the dates to date format
df['DATE OCC'] = pd.to_datetime(df['DATE OCC'])
df['Date Rptd'] = pd.to_datetime(df['Date Rptd'])
df['YEAR'] = df['DATE OCC'].dt.year
df['MONTH'] = df['DATE OCC'].dt.month
df['DAY'] = df['DATE OCC'].dt.day
```

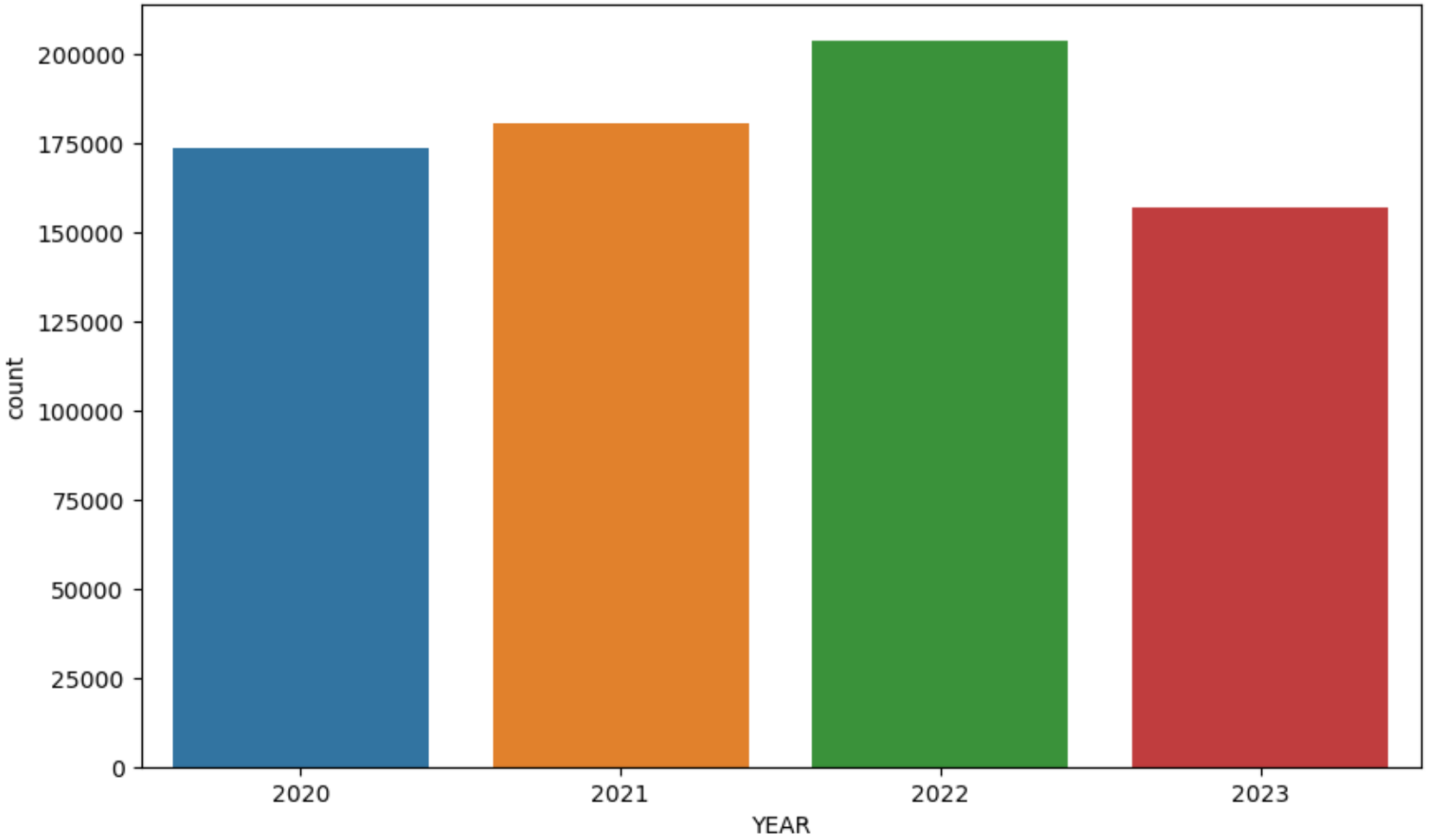
In [18]:
```
1 df.dtypes
```

Out[18]:
```
Date Rptd       datetime64[ns]
DATE OCC        datetime64[ns]
TIME OCC                 int64
AREA NAME               object
Rpt Dist No              int64
Part 1-2                 int64
Crm Cd                   int64
Crm Cd Desc             object
Vict Age                 int64
Vict Sex                object
Vict Descent            object
Premis Desc             object
Status Desc             object
LAT                    float64
LON                    float64
Age Group               object
YEAR                     int64
MONTH                    int64
DAY                      int64
dtype: object
```

1. Overall Crime Trends:

total number of crimes per year to visualize the trends.

In [19]:
```
1 plt.figure(figsize=(10,6))
2 sns.countplot(x='YEAR',data=df)
3 plt.show()
4 print('The total number of crimes in 2020 is: 173866','\nThe total number of crimes in 2021 is: 180939','
```




```
The total number of crimes in 2020 is: 173866
The total number of crimes in 2021 is: 180939
The total number of crimes in 2022 is: 204033
The total number of crimes in 2023 is: 153331
```

2. Seasonal Patterns:

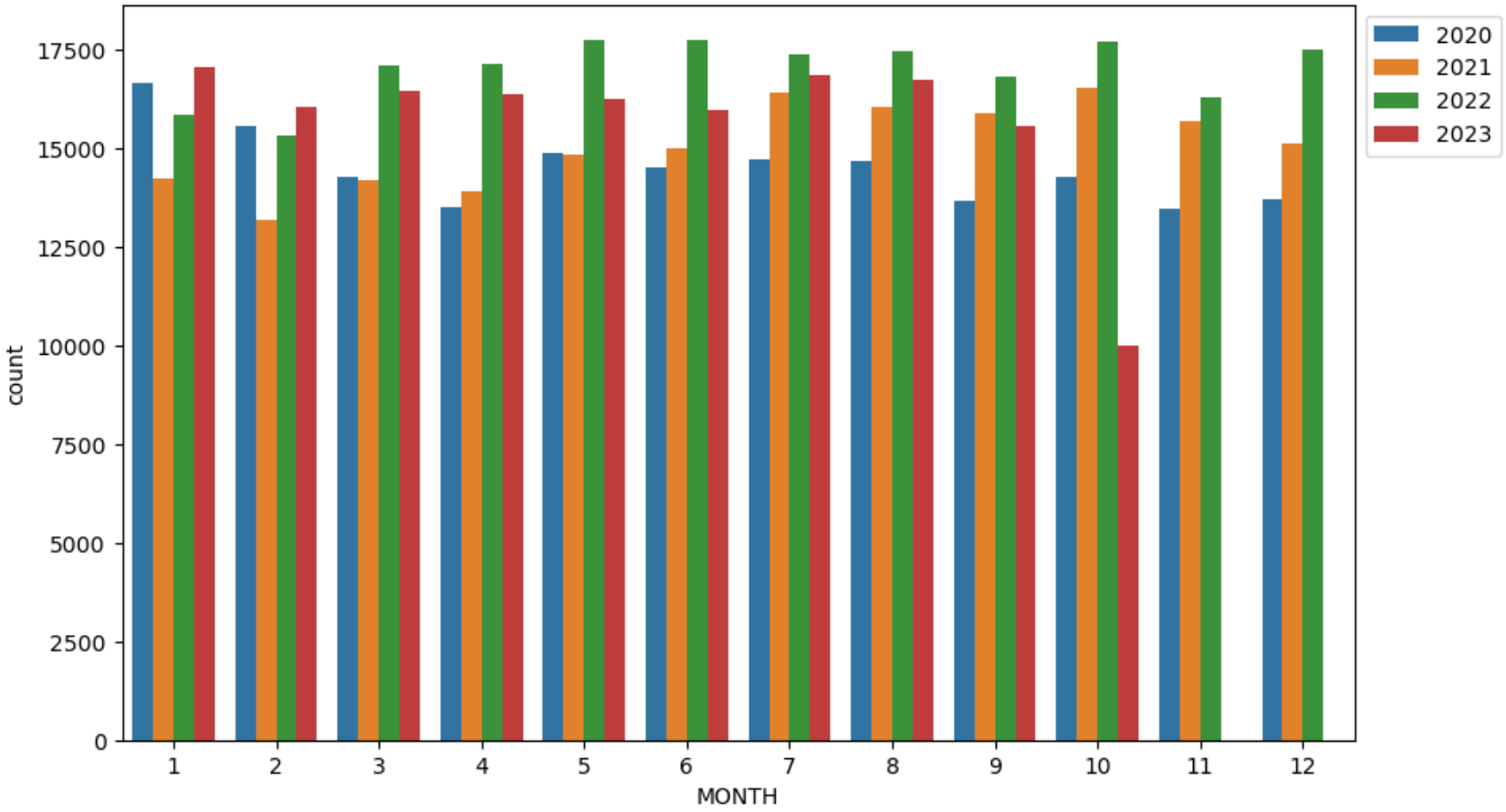
The average number of crimes per month over the years.

In [20]:
```
1 df.columns
```

Out[20]:
```
Index(['Date Rptd', 'DATE OCC', 'TIME OCC', 'AREA NAME', 'Rpt Dist No',
       'Part 1-2', 'Crm Cd', 'Crm Cd Desc', 'Vict Age', 'Vict Sex',
       'Vict Descent', 'Premis Desc', 'Status Desc', 'LAT', 'LON', 'Age Group',
       'YEAR', 'MONTH', 'DAY'],
      dtype='object')
```

In [21]:
```
1 #Grouping crime by month
2 crime_by_month = df.groupby(['MONTH'])
```

In [22]:
```
plt.figure(figsize=(10,6))
sns.countplot(x='MONTH',data=df,hue='YEAR')
plt.legend(loc=0,bbox_to_anchor=(1, 1))
plt.show()
```



3. Most Common Crime Type:

In [23]:
```
crime_type = df['Crm Cd Desc'].value_counts()
crime_type
```

Out[23]:
```
BATTERY - SIMPLE ASSAULT                                    65689
THEFT OF IDENTITY                                           52117
BURGLARY FROM VEHICLE                                       50589
VANDALISM - FELONY ($400 & OVER, ALL CHURCH VANDALISMS)     50240
BURGLARY                                                    49916
                                                            ...
THEFT, COIN MACHINE - ATTEMPT                                   5
FIREARMS RESTRAINING ORDER (FIREARMS RO)                        4
FAILURE TO DISPERSE                                             3
DISHONEST EMPLOYEE ATTEMPTED THEFT                              2
INCITING A RIOT                                                 1
Name: Crm Cd Desc, Length: 137, dtype: int64
```

In [24]:
```
print('The most common crime type is:',crime_type.head(1))
```

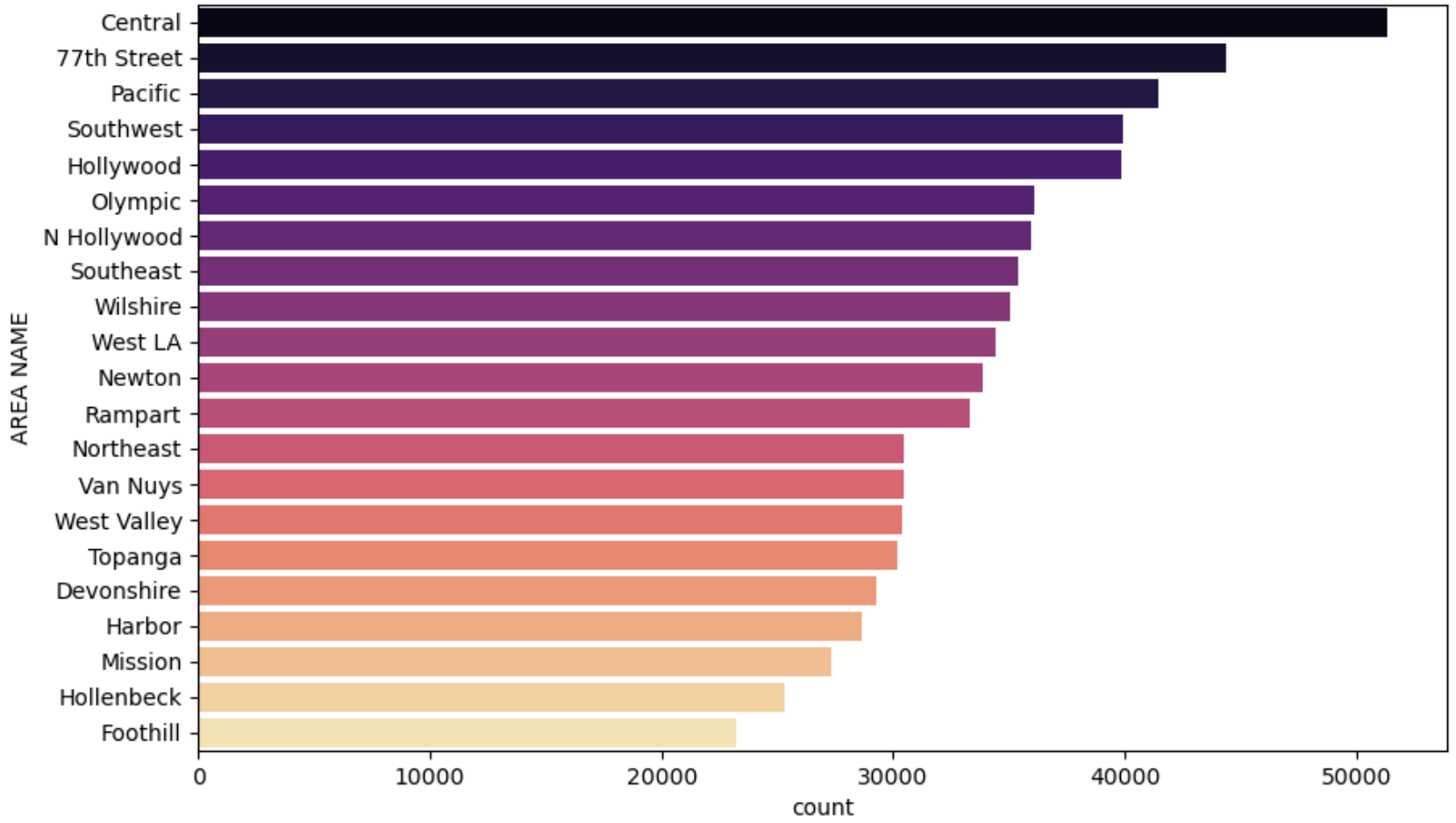
```
The most common crime type is: BATTERY - SIMPLE ASSAULT    65689
Name: Crm Cd Desc, dtype: int64
```

4. Regional Differences:

In [25]:
```
1 df['AREA NAME'].value_counts()
```

Out[25]:
```
Central         51339
77th Street     44358
Pacific         41424
Southwest       39890
Hollywood       39873
Olympic         36078
N Hollywood     35942
Southeast       35398
Wilshire        35057
West LA         34425
Newton          33884
Rampart         33333
Northeast       30479
Van Nuys        30476
West Valley     30364
Topanga         30146
Devonshire      29285
Harbor          28619
Mission         27322
Hollenbeck      25308
Foothill        23193
Name: AREA NAME, dtype: int64
```

In [26]:
```
plt.figure(figsize=(10,6))
sns.countplot(y='AREA NAME',data=df,order=df['AREA NAME'].value_counts().index,palette='magma')
plt.show()
```



In [27]:
```
df['YEAR'].value_counts()
```

Out[27]:
```
2022    204068
2021    180951
2020    173872
2023    157302
Name: YEAR, dtype: int64
```

5. Correlation with Economic Factors:

In [28]:
```
#loading dataset from (https://fred.stlouisfed.org/series/GDP)
df_economic = pd.read_csv(r"D:\Documents\Prof_Docs\FDA\Projects\Project 1\GDP.csv")
df_economic.isnull().sum()
```

Out[28]:
```
DATE    0
GDP     0
dtype: int64
```

In [39]:
```
#Filtwering the data toi have years specific to crime
df_economic['DATE'] = pd.to_datetime(df_economic['DATE'])
df_economic = df_economic[df_economic['DATE'] >= '2020-01-01']
df_economic = df_economic.reset_index(drop=True)
df_economic.to_csv('filtered_data.csv', index=False)
```

In [40]:
```
#checking filtered data
filtered = pd.read_csv('filtered_data.csv')
filtered['DATE'].head()
```

Out[40]:
```
0    2020-01-01
1    2020-04-01
2    2020-07-01
3    2020-10-01
4    2021-01-01
Name: DATE, dtype: object
```

In [41]:
```
#converting the data to datetime datatype
filtered['DATE'] = pd.to_datetime(filtered['DATE'])
filtered['YEAR'] = filtered['DATE'].dt.year
filtered['MONTH'] = filtered['DATE'].dt.month
```

In [42]:
```
#checking filtered data
filtered.head()
```

Out[42]:

	DATE	GDP	YEAR	MONTH
0	2020-01-01	21706.513	2020	1
1	2020-04-01	19913.143	2020	4
2	2020-07-01	21647.640	2020	7
3	2020-10-01	22024.502	2020	10
4	2021-01-01	22600.185	2021	1

In [43]:

```
#merging the data
data = pd.merge(merged_data, filtered, on=['YEAR','MONTH'], how='inner')
data
```

Out[43]:

	Date Rptd	DATE OCC	TIME OCC	AREA NAME	Rpt Dist No	Part 1-2	Crm Cd	Crm Cd Desc	Vict Age	Vict Sex	Vict Descent	Premis Desc	Status Desc	LAT	
0	2020-01-08	2020-01-08	2230	Southwest	377	2	624	BATTERY - SIMPLE ASSAULT	36	F	Black	SINGLE FAMILY DWELLING	Adult Other	34.0141	-1
1	2020-01-08	2020-01-08	2230	Southwest	377	2	624	BATTERY - SIMPLE ASSAULT	36	F	Black	SINGLE FAMILY DWELLING	Adult Other	34.0141	-1
2	2020-01-08	2020-01-08	2230	Southwest	377	2	624	BATTERY - SIMPLE ASSAULT	36	F	Black	SINGLE FAMILY DWELLING	Adult Other	34.0141	-1
3	2020-01-08	2020-01-08	2230	Southwest	377	2	624	BATTERY - SIMPLE ASSAULT	36	F	Black	SINGLE FAMILY DWELLING	Adult Other	34.0141	-1
4	2020-01-08	2020-01-08	2230	Southwest	377	2	624	BATTERY - SIMPLE ASSAULT	36	F	Black	SINGLE FAMILY DWELLING	Adult Other	34.0141	-1
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	
2285119	2022-04-04	2022-04-02	1	Rampart	241	2	354	THEFT OF IDENTITY	35	F	Hispanic/Latin/Mexican	MULTI-UNIT DWELLING (APARTMENT, DUPLEX, ETC)	Invest Cont	34.0649	-1
2285120	2022-04-04	2022-04-02	1	Rampart	241	2	354	THEFT OF IDENTITY	35	F	Hispanic/Latin/Mexican	MULTI-UNIT DWELLING (APARTMENT, DUPLEX, ETC)	Invest Cont	34.0649	-1
2285121	2022-04-04	2022-04-02	1	Rampart	241	2	354	THEFT OF IDENTITY	35	F	Hispanic/Latin/Mexican	MULTI-UNIT DWELLING (APARTMENT, DUPLEX, ETC)	Invest Cont	34.0649	-1
2285122	2022-04-04	2022-04-02	1	Rampart	241	2	354	THEFT OF IDENTITY	35	F	Hispanic/Latin/Mexican	MULTI-UNIT DWELLING (APARTMENT, DUPLEX, ETC)	Invest Cont	34.0649	-1

	Date Rptd	DATE OCC	TIME OCC	AREA NAME	Rpt Dist No	Part 1-2	Crm Cd	Crm Cd Desc	Vict Age	Vict Sex	Vict Descent	Premis Desc	Status Desc	LAT	
2285123	2022-04-04	2022-04-02	1	Rampart	241	2	354	THEFT OF IDENTITY	35	F	Hispanic/Latin/Mexican	MULTI-UNIT DWELLING (APARTMENT, DUPLEX, ETC)	Invest Cont	34.0649	-1

2285124 rows × 22 columns

In [63]:
```
1 data.columns
```

Out[63]:
```
Index(['Date Rptd', 'DATE OCC', 'TIME OCC', 'AREA NAME', 'Rpt Dist No',
       'Part 1-2', 'Crm Cd', 'Crm Cd Desc', 'Vict Age', 'Vict Sex',
       'Vict Descent', 'Premis Desc', 'Status Desc', 'LAT', 'LON', 'Age Group',
       'YEAR', 'MONTH', 'DAY', 'Major event/policy', 'DATE', 'GDP'],
      dtype='object')
```

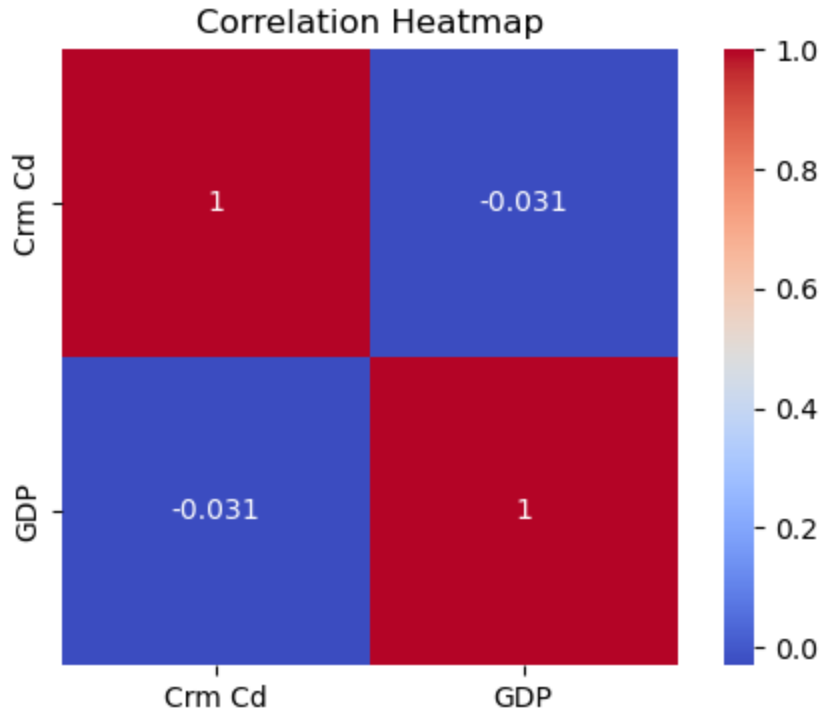
In [45]:
```
1 data.corr()
```

```
C:\Users\shrey\AppData\Local\Temp\ipykernel_36768\2627137660.py:1: FutureWarning: The default value of numer
ic_only in DataFrame.corr is deprecated. In a future version, it will default to False. Select only valid co
lumns or specify the value of numeric_only to silence this warning.
  data.corr()
```

Out[45]:

	TIME OCC	Rpt Dist No	Part 1-2	Crm Cd	Vict Age	LAT	LON	YEAR	MONTH	DAY	GDP
TIME OCC	1.000000	0.003346	-0.038128	0.016433	0.000413	0.001088	-0.000933	-0.014811	-0.001093	0.041691	-0.015940
Rpt Dist No	0.003346	1.000000	0.018444	0.004636	0.031864	0.017187	-0.005497	-0.010229	-0.008498	-0.006391	-0.013007
Part 1-2	-0.038128	0.018444	1.000000	0.737325	0.038973	-0.028528	0.028869	-0.008042	0.005218	-0.047525	-0.005908
Crm Cd	0.016433	0.004636	0.737325	1.000000	-0.017767	-0.045951	0.046294	-0.029907	0.000314	0.014749	-0.030659
Vict Age	0.000413	0.031864	0.038973	-0.017767	1.000000	0.008929	-0.007880	-0.017303	0.001447	-0.009017	-0.015587
LAT	0.001088	0.017187	-0.028528	-0.045951	0.008929	1.000000	-0.998825	0.040979	-0.010343	-0.001308	0.041608
LON	-0.000933	-0.005497	0.028869	0.046294	-0.007880	-0.998825	1.000000	-0.040631	0.009994	0.001810	-0.041263
YEAR	-0.014811	-0.010229	-0.008042	-0.029907	-0.017303	0.040979	-0.040631	1.000000	-0.108775	-0.017944	0.950888
MONTH	-0.001093	-0.008498	0.005218	0.000314	0.001447	-0.010343	0.009994	-0.108775	1.000000	-0.001743	0.126781
DAY	0.041691	-0.006391	-0.047525	0.014749	-0.009017	-0.001308	0.001810	-0.017944	-0.001743	1.000000	-0.018756
GDP	-0.015940	-0.013007	-0.005908	-0.030659	-0.015587	0.041608	-0.041263	0.950888	0.126781	-0.018756	1.000000

In [46]:
```
columns_to_correlate = ['Crm Cd', 'GDP']
correlation_matrix = data[columns_to_correlate].corr()
plt.figure(figsize=(6, 4))
sns.heatmap(correlation_matrix, annot=True, cmap='coolwarm', square=True)
plt.title('Correlation Heatmap')
plt.show()
```



We have a negative correlation here, Thus we can conclude that as the GDP increases the crime rates tends to drop.

6. Day of the Week Analysis:

In [47]:

```
df['Date'] = pd.to_datetime(df[['YEAR', 'MONTH', 'DAY']])

sorted_df = df.sort_values(by='Date')

sorted_days = sorted_df['DAY'].value_counts().sort_index()

print(sorted_days)
df['DayOfWeek'] = df['Date'].dt.day_name()

print(df[['Date', 'DayOfWeek']])
```

```
1     34115
2     27768
3     26666
4     23551
5     23583
6     23227
7     22902
8     23252
9     22568
10    23458
11    22648
12    22968
13    22907
14    23086
15    24209
16    22962
17    22846
18    22922
19    22610
20    23224
21    22954
22    22748
23    22803
24    22185
25    22330
26    22132
27    22156
28    22707
29    20802
30    20811
31    13093
Name: DAY, dtype: int64
             Date  DayOfWeek
0      2020-01-08  Wednesday
1      2020-01-01  Wednesday
2      2020-02-13   Thursday
3      2020-01-01  Wednesday
4      2020-01-01  Wednesday
...           ...        ...
825207 2023-01-26   Thursday
825208 2023-03-22  Wednesday
825209 2023-04-12  Wednesday
825210 2023-07-01   Saturday
```

```
825211 2023-03-05     Sunday

[716193 rows x 2 columns]
```

In [48]:

```python
#grouping the crimes per day of week
crime_frequencies = df['DayOfWeek'].value_counts()

total_crimes = crime_frequencies.sum()
average_crimes_per_day = total_crimes / len(crime_frequencies)
max_crimes_day = crime_frequencies.idxmax()
min_crimes_day = crime_frequencies.idxmin()

print("Crime Frequencies by Day of the Week:")
print(crime_frequencies)
print("\nTotal Crimes:", total_crimes)
print("Average Crimes per Day:", average_crimes_per_day)
print("Day with the Most Crimes:", max_crimes_day)
print("Day with the Fewest Crimes:", min_crimes_day)
```

```
Crime Frequencies by Day of the Week:
Friday       108658
Saturday     104765
Wednesday    101676
Monday       101557
Thursday     100906
Sunday       100359
Tuesday       98272
Name: DayOfWeek, dtype: int64

Total Crimes: 716193
Average Crimes per Day: 102313.28571428571
Day with the Most Crimes: Friday
Day with the Fewest Crimes: Tuesday
```

In [49]:

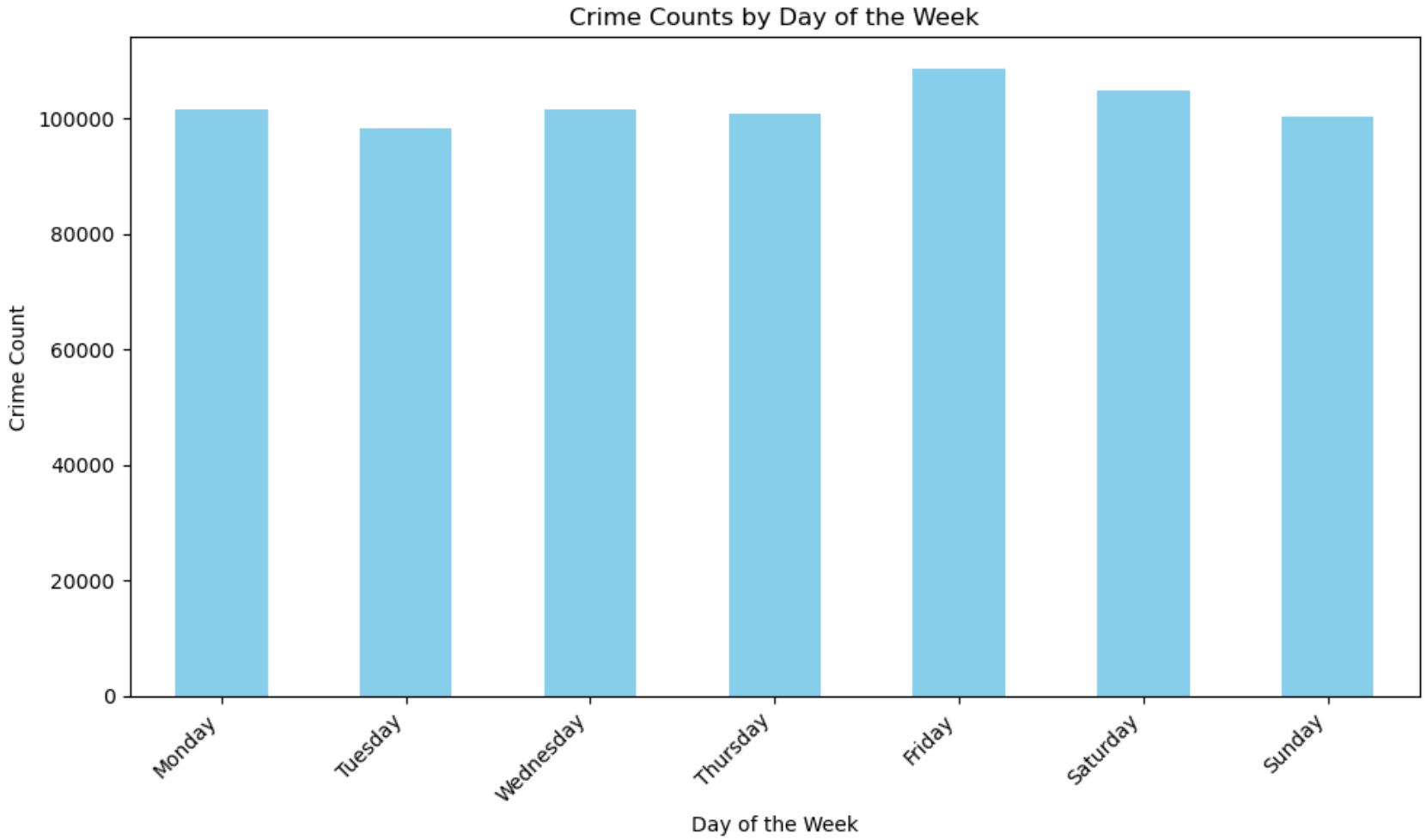
```
#sorting data from 1-31
df['Date'] = pd.to_datetime(df[['YEAR', 'MONTH', 'DAY']])

sorted_df = df.sort_values(by='Date')

sorted_days = sorted_df['DAY'].value_counts().sort_index()

print(sorted_days)
df['DayOfWeek'] = df['Date'].dt.day_name()

print(df[['Date', 'DayOfWeek']])
```

```
1     34115
2     27768
3     26666
4     23551
5     23583
6     23227
7     22902
8     23252
9     22568
10    23458
11    22648
12    22968
13    22907
14    23086
15    24209
16    22962
17    22846
18    22922
19    22610
20    23224
21    22954
22    22748
23    22803
24    22185
25    22330
26    22132
27    22156
28    22707
29    20802
30    20811
31    13093
Name: DAY, dtype: int64
             Date  DayOfWeek
0      2020-01-08  Wednesday
1      2020-01-01  Wednesday
2      2020-02-13   Thursday
3      2020-01-01  Wednesday
4      2020-01-01  Wednesday
...           ...        ...
825207 2023-01-26   Thursday
825208 2023-03-22  Wednesday
825209 2023-04-12  Wednesday
825210 2023-07-01   Saturday
```

```
825211 2023-03-05     Sunday

[716193 rows x 2 columns]
```

In [50]:

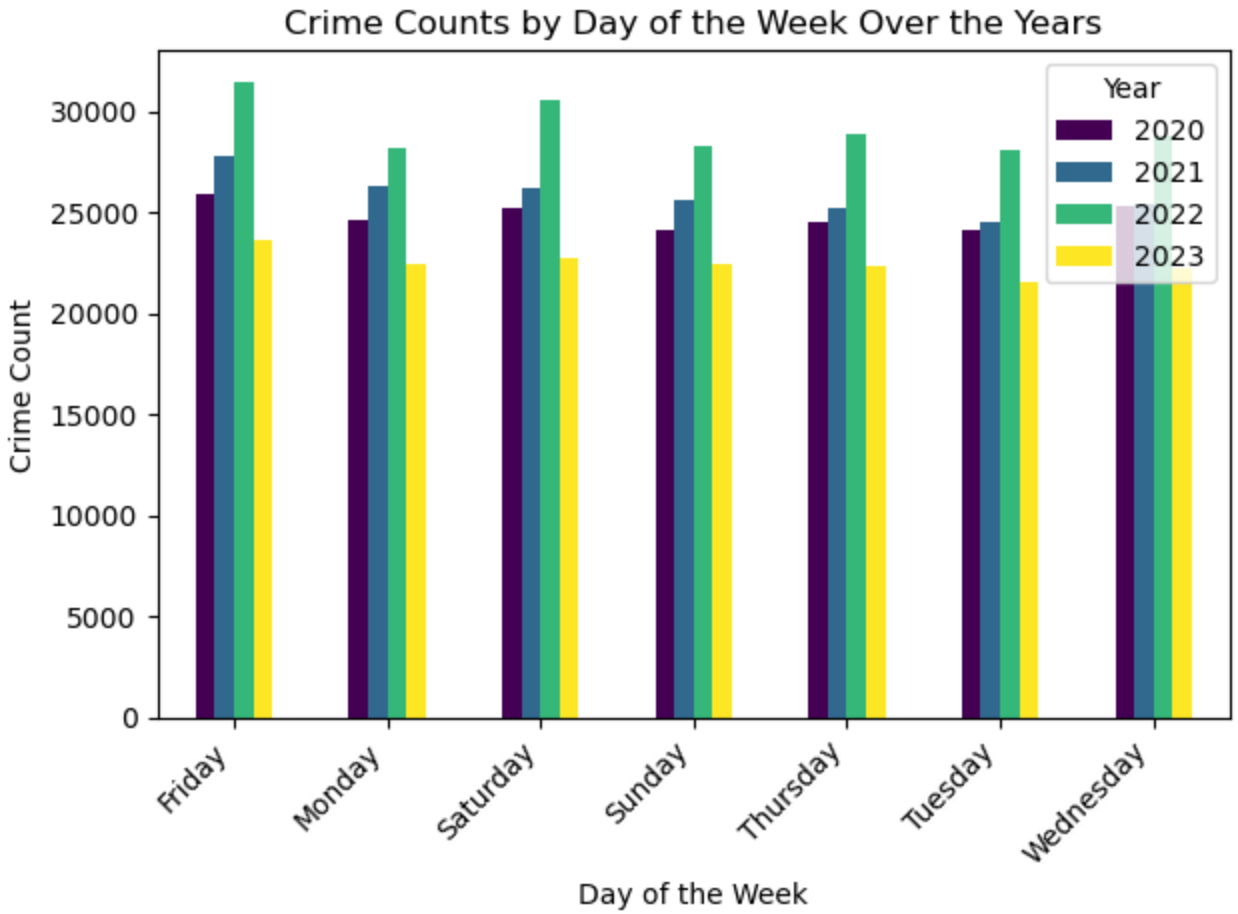
```
#Crime per day
crime_counts_by_day_of_week = df['DayOfWeek'].value_counts()

days_of_week_order = ['Monday', 'Tuesday', 'Wednesday', 'Thursday', 'Friday', 'Saturday', 'Sunday']
crime_counts_by_day_of_week = crime_counts_by_day_of_week.reindex(days_of_week_order)

plt.figure(figsize=(10, 6))
crime_counts_by_day_of_week.plot(kind='bar', color='skyblue')
plt.title('Crime Counts by Day of the Week')
plt.xlabel('Day of the Week')
plt.ylabel('Crime Count')
plt.xticks(rotation=45, ha='right')
plt.tight_layout()
plt.show()
```


Crime Counts by Day of the Week
Crime Count
0
20000
40000
60000
80000
100000
Monday
Tuesday
Wednesday
Thursday
Friday
Saturday
Sunday
Day of the Week

In [51]:

```
##Crime per day
df['Date'] = pd.to_datetime(df[['YEAR', 'MONTH', 'DAY']])

crime_counts_by_year_and_day = df.groupby(['YEAR', 'DayOfWeek'])['Date'].count()

crime_counts_by_year_and_day = crime_counts_by_year_and_day.unstack('YEAR')

years = range(df['YEAR'].min(), df['YEAR'].max() + 1)
crime_counts_by_year_and_day = crime_counts_by_year_and_day[years]

total_crimes_by_year = crime_counts_by_year_and_day.sum()
average_crimes_per_year = total_crimes_by_year.mean()
max_crimes_year = total_crimes_by_year.idxmax()
min_crimes_year = total_crimes_by_year.idxmin()

plt.figure(figsize=(12, 6))
crime_counts_by_year_and_day.plot(kind='bar', color=plt.cm.viridis(np.linspace(0, 1, len(years))))
plt.title('Crime Counts by Day of the Week Over the Years')
plt.xlabel('Day of the Week')
plt.ylabel('Crime Count')
plt.xticks(rotation=45, ha='right')
plt.legend(title='Year', loc='upper right')
plt.tight_layout()
plt.show()

print("Total Crimes by Year:")
print(total_crimes_by_year)
print("\nAverage Crimes per Year:", average_crimes_per_year)
print("Year with the Most Crimes:", max_crimes_year)
print("Year with the Fewest Crimes:", min_crimes_year)
```

```
<Figure size 1200x600 with 0 Axes>
```




```
Total Crimes by Year:
YEAR
2020    173872
2021    180951
2022    204068
2023    157302
dtype: int64

Average Crimes per Year: 179048.25
Year with the Most Crimes: 2022
Year with the Fewest Crimes: 2023
```

7. Impact of Major Events:

In [34]:

```
#Merging the major events data
df_ME = pd.read_csv(r"D:\Documents\Prof_Docs\FDA\Projects\Project 1\Major events file.csv")
merged_data = pd.merge(df, df_ME, on='YEAR', how='inner')
merged_data
```

Out[34]:

	Date Rptd	DATE OCC	TIME OCC	AREA NAME	Rpt Dist No	Part 1-2	Crm Cd	Crm Cd Desc	Vict Age	Vict Sex	Vict Descent	Premis Desc	Status Desc	LAT	LON	A Gro
0	2020-01-08	2020-01-08	2230	Southwest	377	2	624	BATTERY - SIMPLE ASSAULT	36	F	Black	SINGLE FAMILY DWELLING	Adult Other	34.0141	-118.2978	A
1	2020-01-08	2020-01-08	2230	Southwest	377	2	624	BATTERY - SIMPLE ASSAULT	36	F	Black	SINGLE FAMILY DWELLING	Adult Other	34.0141	-118.2978	A
2	2020-01-08	2020-01-08	2230	Southwest	377	2	624	BATTERY - SIMPLE ASSAULT	36	F	Black	SINGLE FAMILY DWELLING	Adult Other	34.0141	-118.2978	A
3	2020-01-08	2020-01-08	2230	Southwest	377	2	624	BATTERY - SIMPLE ASSAULT	36	F	Black	SINGLE FAMILY DWELLING	Adult Other	34.0141	-118.2978	A
4	2020-01-08	2020-01-08	2230	Southwest	377	2	624	BATTERY - SIMPLE ASSAULT	36	F	Black	SINGLE FAMILY DWELLING	Adult Other	34.0141	-118.2978	A
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	
6847321	2022-12-25	2022-12-25	1300	West Valley	1065	2	626	INTIMATE PARTNER - SIMPLE ASSAULT	33	M	Black	MULTI-UNIT DWELLING (APARTMENT, DUPLEX, ETC)	Adult Other	34.1667	-118.5244	A
6847322	2022-12-25	2022-12-25	1300	West Valley	1065	2	626	INTIMATE PARTNER - SIMPLE ASSAULT	33	M	Black	MULTI-UNIT DWELLING (APARTMENT, DUPLEX, ETC)	Adult Other	34.1667	-118.5244	A
6847323	2022-12-25	2022-12-25	1300	West Valley	1065	2	626	INTIMATE PARTNER - SIMPLE ASSAULT	33	M	Black	MULTI-UNIT DWELLING (APARTMENT, DUPLEX, ETC)	Adult Other	34.1667	-118.5244	A
6847324	2022-12-25	2022-12-25	1300	West Valley	1065	2	626	INTIMATE PARTNER - SIMPLE ASSAULT	33	M	Black	MULTI-UNIT DWELLING (APARTMENT, DUPLEX, ETC)	Adult Other	34.1667	-118.5244	A

	Date Rptd	DATE OCC	TIME OCC	AREA NAME	Rpt Dist No	Part 1-2	Crm Cd	Crm Cd Desc	Vict Age	Vict Sex	Vict Descent	Premis Desc	Status Desc	LAT	LON	A Gro
6847325	2022-12-25	2022-12-25	1300	West Valley	1065	2	626	INTIMATE PARTNER - SIMPLE ASSAULT	33	M	Black	MULTI-UNIT DWELLING (APARTMENT, DUPLEX, ETC)	Adult Other	34.1667	-118.5244	A

6847326 rows × 20 columns

In [35]:
```
1 merged_data.head()
```

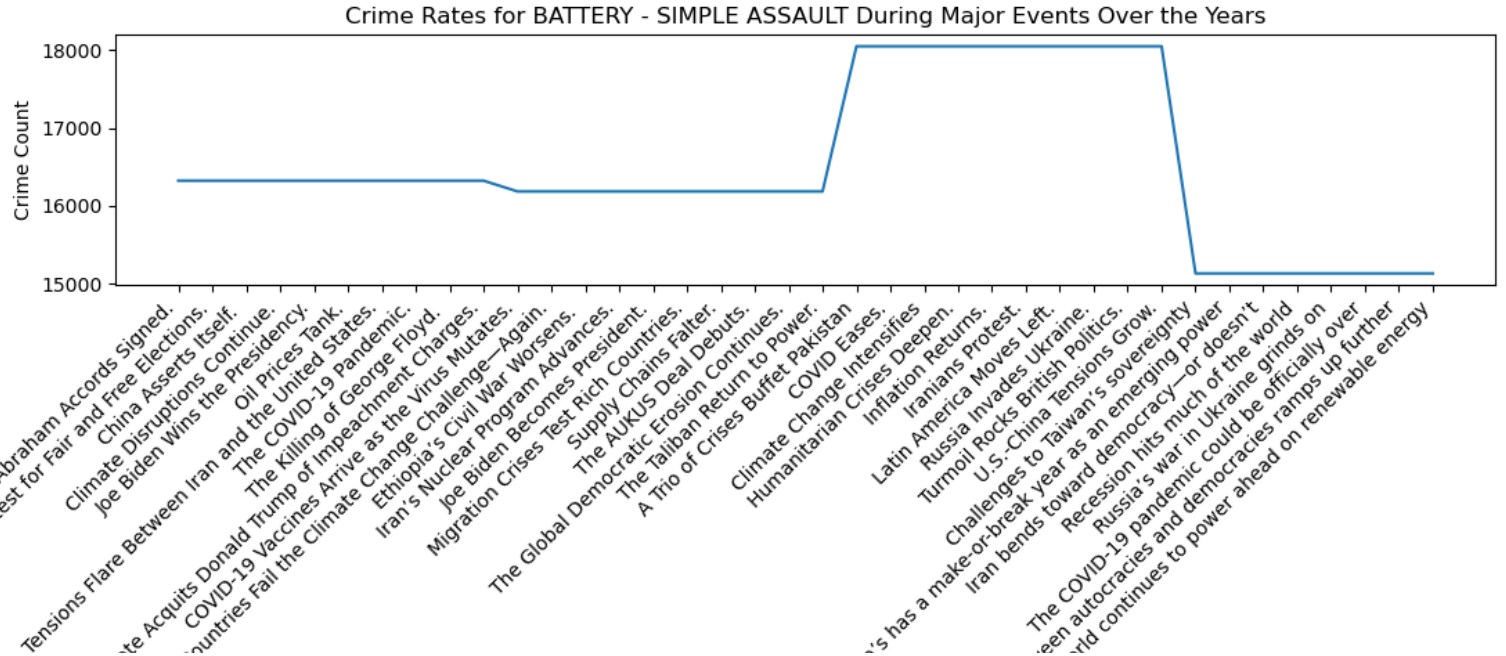
Out[35]:

	Date Rptd	DATE OCC	TIME OCC	AREA NAME	Rpt Dist No	Part 1-2	Crm Cd	Crm Cd Desc	Vict Age	Vict Sex	Vict Descent	Premis Desc	Status Desc	LAT	LON	Age Group	YEAR
0	2020-01-08	2020-01-08	2230	Southwest	377	2	624	BATTERY - SIMPLE ASSAULT	36	F	Black	SINGLE FAMILY DWELLING	Adult Other	34.0141	-118.2978	Adult	2020
1	2020-01-08	2020-01-08	2230	Southwest	377	2	624	BATTERY - SIMPLE ASSAULT	36	F	Black	SINGLE FAMILY DWELLING	Adult Other	34.0141	-118.2978	Adult	2020
2	2020-01-08	2020-01-08	2230	Southwest	377	2	624	BATTERY - SIMPLE ASSAULT	36	F	Black	SINGLE FAMILY DWELLING	Adult Other	34.0141	-118.2978	Adult	2020
3	2020-01-08	2020-01-08	2230	Southwest	377	2	624	BATTERY - SIMPLE ASSAULT	36	F	Black	SINGLE FAMILY DWELLING	Adult Other	34.0141	-118.2978	Adult	2020
4	2020-01-08	2020-01-08	2230	Southwest	377	2	624	BATTERY - SIMPLE ASSAULT	36	F	Black	SINGLE FAMILY DWELLING	Adult Other	34.0141	-118.2978	Adult	2020

In [36]:
```
1 crime_count_by_year_event = merged_data.groupby(['YEAR', 'Crm Cd Desc', 'Major event/policy']).size().res
```

In [37]:

```
#Plotting crime rates of per type    per major event
unique_crime_types = merged_data['Crm Cd Desc'].unique()

for crime_type in unique_crime_types:
    plt.figure(figsize=(12, 6))
    plt.title(f'Crime Rates for {crime_type} During Major Events Over the Years')

    crime_data_to_plot = crime_count_by_year_event[crime_count_by_year_event['Crm Cd Desc'] == crime_type

    sns.lineplot(data=crime_data_to_plot, x='Major event/policy', y='Count')

    plt.xlabel('Major Event/Policy')
    plt.ylabel('Crime Count')
    plt.xticks(rotation=45, ha='right')

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



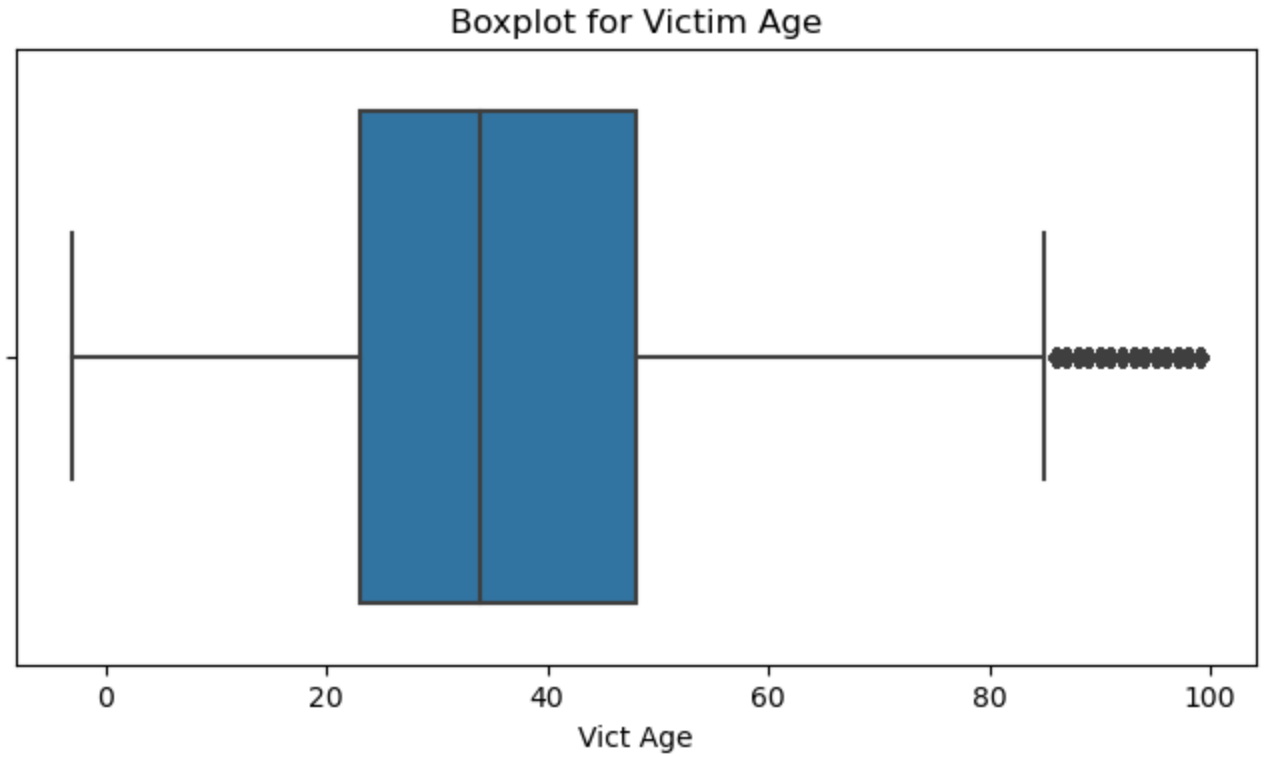
In [38]:

```
unique_crime_types = crime_count_by_year_event['Crm Cd Desc'].unique()

average_rates = {}
percentage_changes = {}

for crime_type in unique_crime_types:
    crime_data = crime_count_by_year_event[crime_count_by_year_event['Crm Cd Desc'] == crime_type]

    for major_event in crime_data['Major event/policy'].unique():
        subset_data = crime_data[crime_data['Major event/policy'] == major_event]

        crime_rates_before = subset_data[subset_data['YEAR'] < 2023]['Count']
        crime_rates_after = subset_data[subset_data['YEAR'] >= 2023]['Count']

        average_rate_before = crime_rates_before.mean()
        average_rate_after = crime_rates_after.mean()

        percentage_change = ((average_rate_after - average_rate_before) / average_rate_before) * 100

        average_rates[(crime_type, major_event)] = (average_rate_before, average_rate_after)
        percentage_changes[(crime_type, major_event)] = percentage_change

for (crime_type, major_event), (average_rate_before, average_rate_after) in average_rates.items():
    print(f'Crime Type: {crime_type}, Major Event: {major_event}')
    print(f'Average Crime Rate Before: {average_rate_before:.2f}')
    print(f'Average Crime Rate After: {average_rate_after:.2f}')
    print(f'Percentage Change in Crime Rates: {percentage_changes[(crime_type, major_event)]:.2f}%')
    print()
```

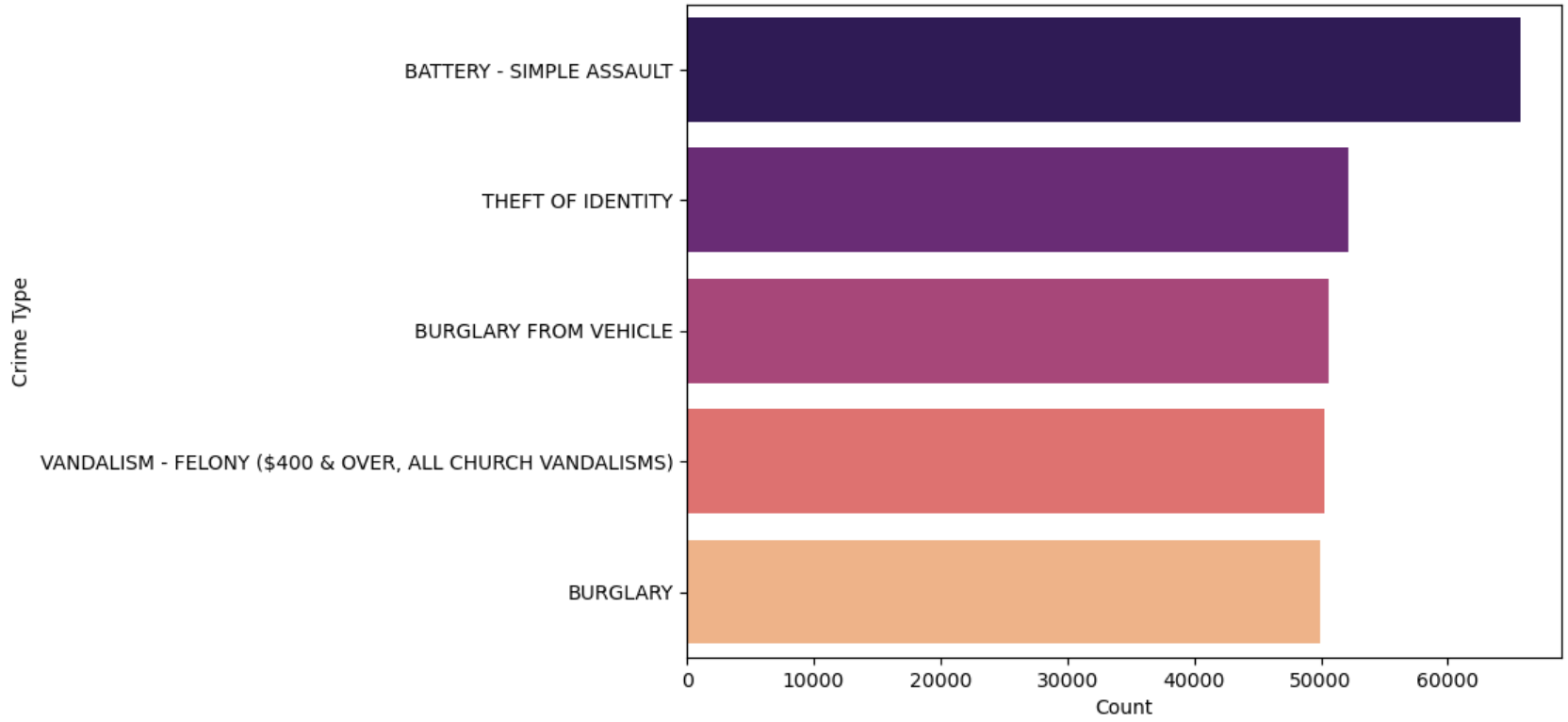
```
Crime Type: ARSON, Major Event: Abraham Accords Signed.
Average Crime Rate Before: 664.00
Average Crime Rate After: nan
Percentage Change in Crime Rates: nan%

Crime Type: ARSON, Major Event: Belarusians Protest for Fair and Free Elections.
Average Crime Rate Before: 664.00
Average Crime Rate After: nan
Percentage Change in Crime Rates: nan%

Crime Type: ARSON, Major Event: China Asserts Itself.
Average Crime Rate Before: 664.00
Average Crime Rate After: nan
Percentage Change in Crime Rates: nan%

Crime Type: ARSON, Major Event: Climate Disruptions Continue.
Average Crime Rate Before: 664.00
Average Crime Rate After: nan
Percentage Change in Crime Rates: nan%
```

8. Outliers and Anomalies:

In [52]:
```
# Checking for outliers and patterns
plt.figure(figsize=(8, 4))
sns.boxplot(x=df['Vict Age'])
plt.title('Boxplot for Victim Age')
plt.show()
```



In [53]:
```
plt.figure(figsize=(8, 6))
sns.countplot(y='Crm Cd Desc',data=df,order=df['Crm Cd Desc'].value_counts().head(5).index,palette='magma
plt.xlabel('Count')
plt.ylabel('Crime Type')
plt.show()
```

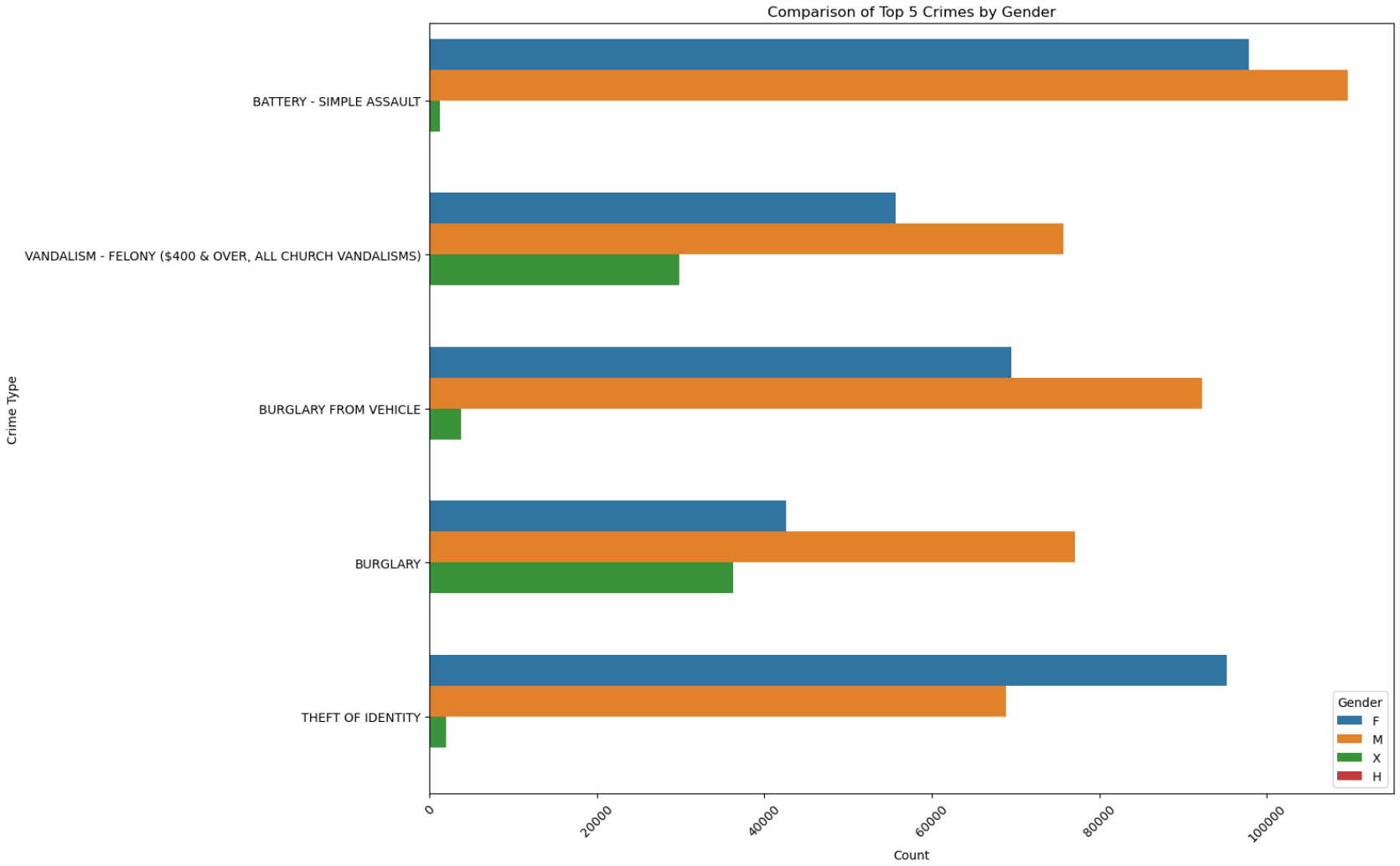


From the above two visuals, we can conclude that there are very few victims over the age of 85 due to the fact that the highest number of crimes are assault,theft of identity, burglary , vandalism , burglary. Where the old people are not involved when such crimes take place.

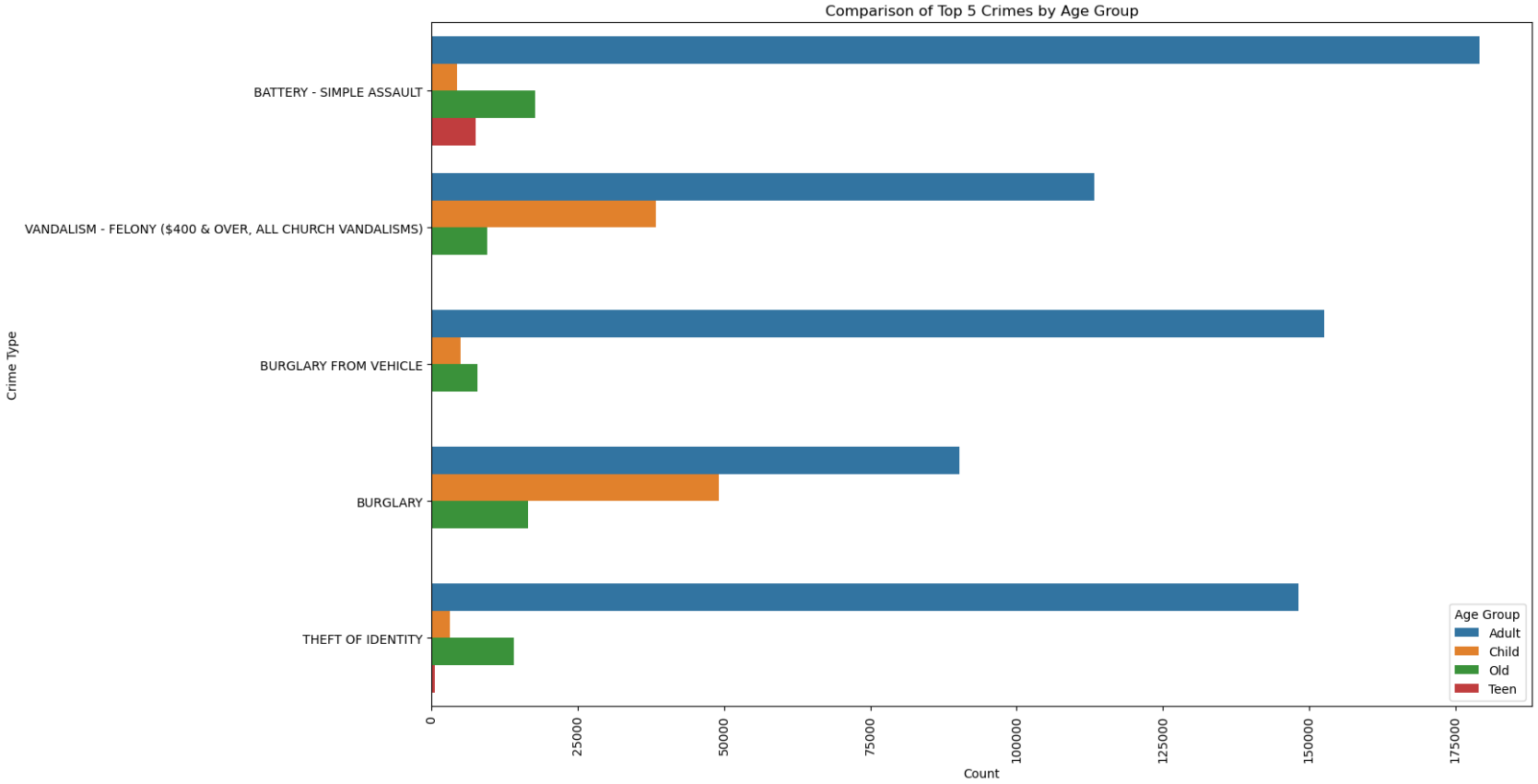
9. Demographic Factors:

In [54]:

```
# Factor 1(Victim Gender)
top_5_crime_types = data['Crm Cd Desc'].value_counts().head(5).index
filtered_df = data[data['Crm Cd Desc'].isin(top_5_crime_types)]

plt.figure(figsize=(16, 10))
sns.countplot(y='Crm Cd Desc', data=filtered_df, hue='Vict Sex')
plt.title('Comparison of Top 5 Crimes by Gender')
plt.ylabel('Crime Type')
plt.xlabel('Count')
plt.xticks(rotation=45)
plt.legend(title='Gender')
plt.tight_layout()
plt.show()
```

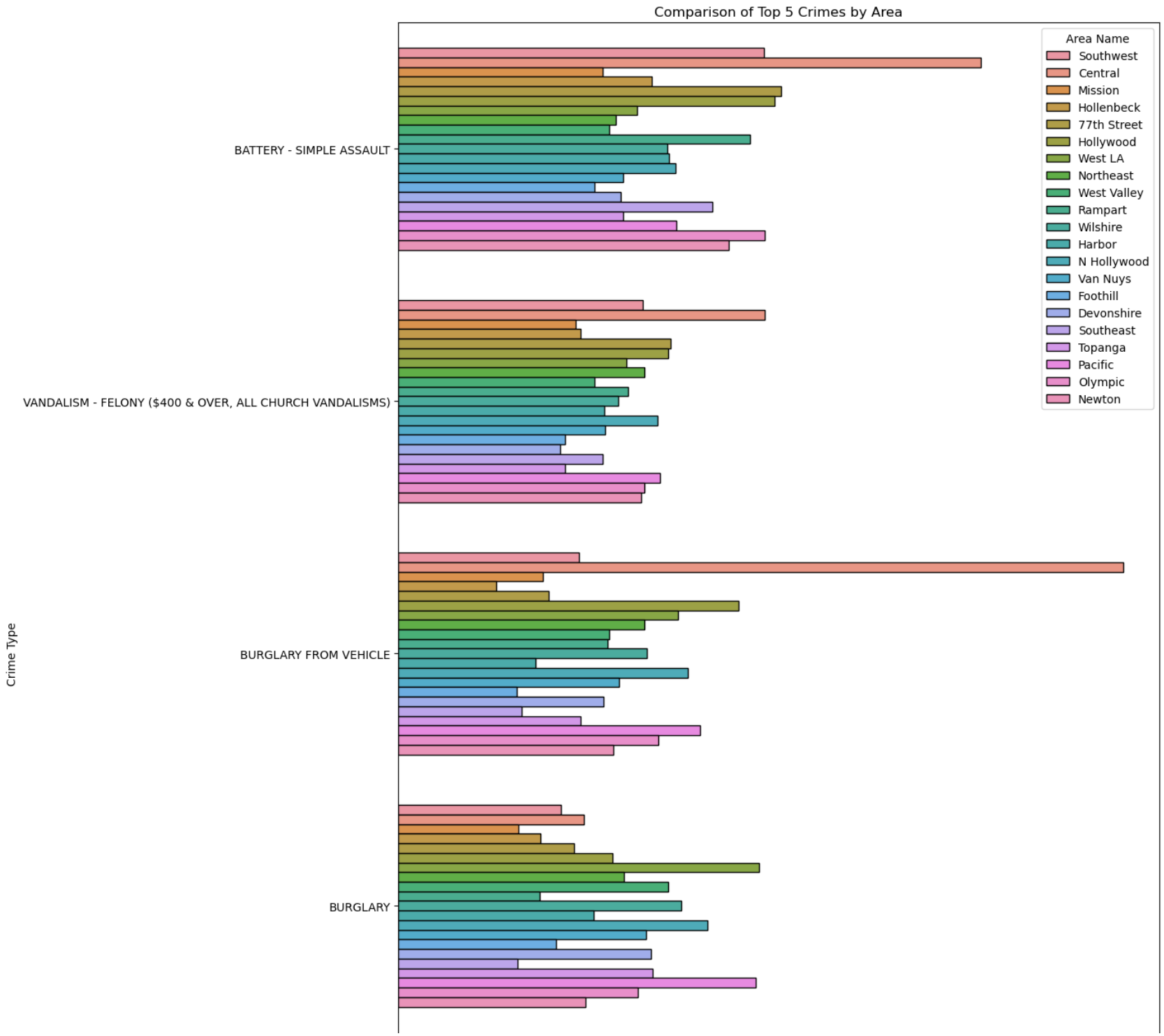

Comparison of Top 5 Crimes by Gender
BATTERY - SIMPLE ASSAULT
VANDALISM - FELONY ($400 & OVER, ALL CHURCH VANDALISMS)
BURGLARY FROM VEHICLE
BURGLARY
THEFT OF IDENTITY
Crime Type
0
20000
40000
60000
80000
100000
Count
Gender
F
M
X
H

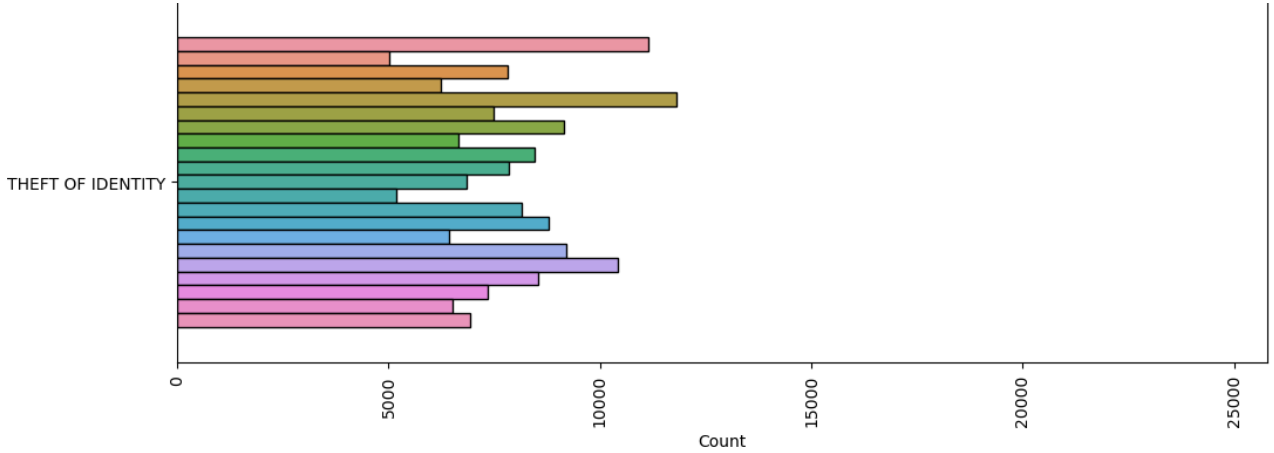
In [55]:
```
# Factor 2(Victim Age Group)
top_5_crime_types = data['Crm Cd Desc'].value_counts().head(5).index
filtered_df = data[data['Crm Cd Desc'].isin(top_5_crime_types)]

plt.figure(figsize=(16, 10))
sns.countplot(y='Crm Cd Desc', data=filtered_df, hue='Age Group')
plt.title('Comparison of Top 5 Crimes by Age Group')
plt.ylabel('Crime Type')
plt.xlabel('Count')
plt.xticks(rotation=90)
plt.legend(title='Age Group')
plt.show()
```



In [56]:

```
# Factor 3(Area)
top_5_crime_types = data['Crm Cd Desc'].value_counts().head(5).index
filtered_df = data[data['Crm Cd Desc'].isin(top_5_crime_types)]

plt.figure(figsize=(12, 20))
sns.countplot(y='Crm Cd Desc', data=filtered_df, hue='AREA NAME', edgecolor='black')
plt.title('Comparison of Top 5 Crimes by Area')
plt.ylabel('Crime Type')
plt.xlabel('Count')
plt.xticks(rotation=90)
plt.legend(title='Area Name')
plt.show()
```


Comparison of Top 5 Crimes by Area
Area Name
Southwest
Central
Mission
Hollenbeck
77th Street
Hollywood
West LA
Northeast
West Valley
Rampart
Wilshire
Harbor
N Hollywood
Van Nuys
Foothill
Devonshire
Southeast
Topanga
Pacific
Olympic
Newton
BATTERY - SIMPLE ASSAULT
VANDALISM - FELONY ($400 & OVER, ALL CHURCH VANDALISMS)
BURGLARY FROM VEHICLE
BURGLARY
Crime Type



10. Predicting Future Trends:

In [57]:
```
1 print(total_crimes_by_year)
```

```
YEAR
2020    173872
2021    180951
2022    204068
2023    157302
dtype: int64
```

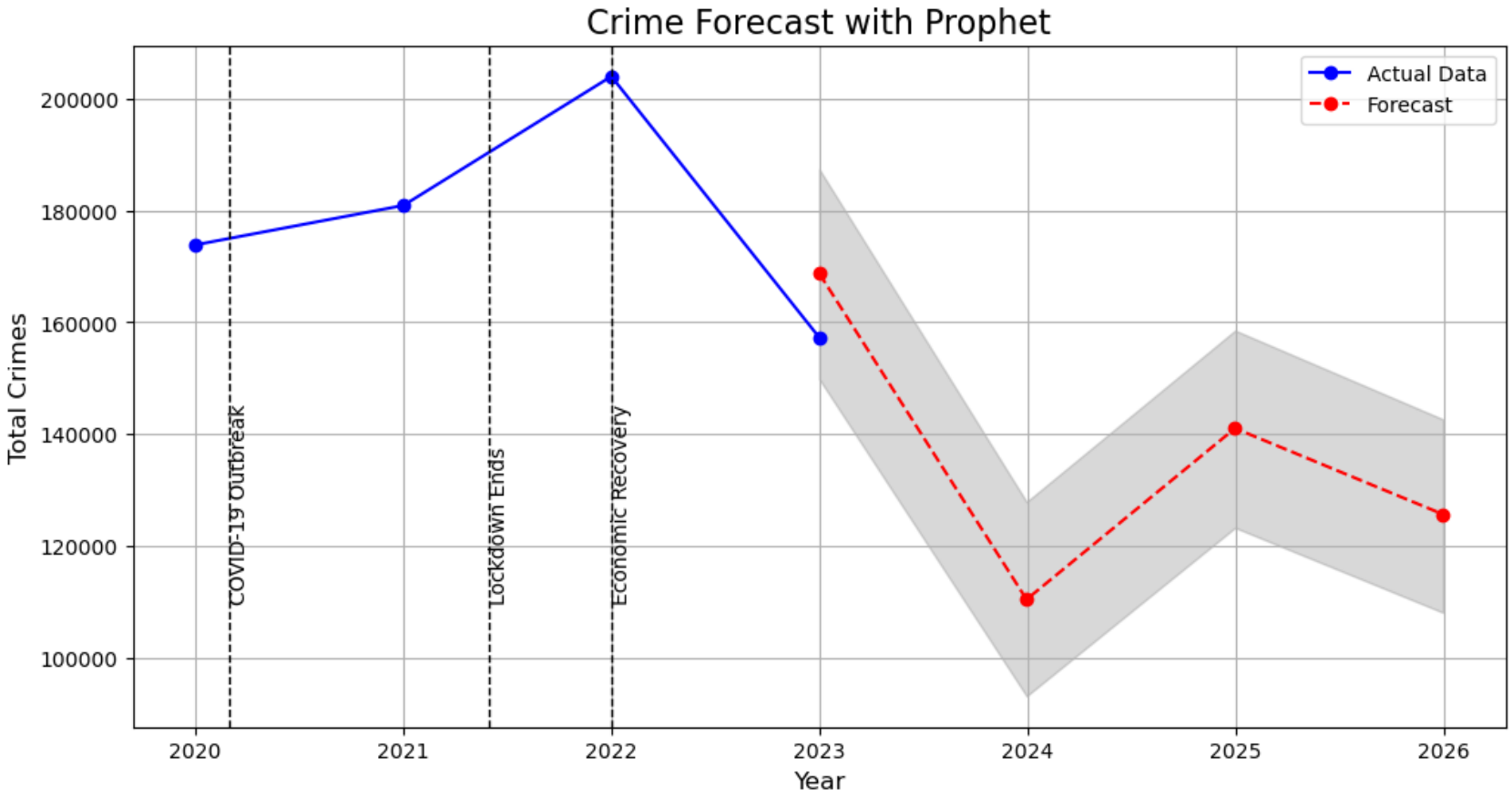
In [59]:

```
from prophet import Prophet
from sklearn.metrics import mean_absolute_error, mean_squared_error
import math

#creating a new dataframe using total_crimes_by_year
data1 = pd.DataFrame({
    'ds': pd.to_datetime(['2020-01-01', '2021-01-01', '2022-01-01', '2023-01-01']),
    'y': [173872, 180951, 204068, 157302]
})

model = Prophet()

model.fit(data1)

future = model.make_future_dataframe(periods=3, freq='Y')

forecast = model.predict(future)

forecasted_data = forecast[['ds', 'yhat', 'yhat_lower', 'yhat_upper']].tail(4)

print("Forecasted Crime Data for the Next 3 Years:")
print(forecasted_data)
print("\nds: This column represents the date for each forecasted year.\n In this case, the dates correspo
print("\nyhat: This column represents the forecasted value for the total number of crimes.\n It's the cen
print("\nyhat_lower: This column represents the lower bound of the forecasted value.\n It provides a lowe
print("\nyhat_upper: This column represents the upper bound of the forecasted value.\n It provides an upp

fig, ax = plt.subplots(figsize=(12, 6))

plt.plot(data1['ds'], data1['y'], label='Actual Data', color='b', marker='o')

plt.plot(forecasted_data['ds'], forecasted_data['yhat'], label='Forecast', color='r', linestyle='--', mar

plt.fill_between(forecasted_data['ds'], forecasted_data['yhat_lower'], forecasted_data['yhat_upper'], alp

observations = {
    'COVID-19 Outbreak': '2020-03-01',
    'Lockdown Ends': '2021-06-01',
    'Economic Recovery': '2022-01-01'
}

for label, date in observations.items():
    plt.axvline(pd.to_datetime(date), color='k', linestyle='--', linewidth=1)
```

```
    plt.text(pd.to_datetime(date), forecasted_data['yhat'].min(), label, rotation=90)

plt.title("Crime Forecast with Prophet", fontsize=16)
plt.xlabel("Year", fontsize=12)
plt.ylabel("Total Crimes", fontsize=12)
plt.legend()

mae = mean_absolute_error(data1['y'][:-1], forecasted_data['yhat'][:-1])
mse = mean_squared_error(data1['y'][:-1], forecasted_data['yhat'][:-1])
rmse = math.sqrt(mse)
mape = (abs((data1['y'][:-1] - forecasted_data['yhat'][:-1]) / data1['y'][:-1])).mean() * 100

print(f"Mean Absolute Error (MAE): {mae:.2f}")
print(f"Mean Squared Error (MSE): {mse:.2f}")
print(f"Root Mean Squared Error (RMSE): {rmse:.2f}")
print(f"Mean Absolute Percentage Error (MAPE): {mape:.2f}%")

plt.grid()
plt.show()
```

```
23:16:50 - cmdstanpy - INFO - Chain [1] start processing
23:16:50 - cmdstanpy - INFO - Chain [1] done processing
```

```
Forecasted Crime Data for the Next 3 Years:
          ds           yhat     yhat_lower     yhat_upper
3 2023-01-01  168771.107706  149838.687487  187442.904036
4 2023-12-31  110365.413912   93070.970137  127858.126917
5 2024-12-31  141017.220740  123227.518377  158558.503610
6 2025-12-31  125625.084130  108029.804393  142647.714029

ds: This column represents the date for each forecasted year.
 In this case, the dates correspond to the end of each year (December 31st).

yhat: This column represents the forecasted value for the total number of crimes.
 It's the central estimate of the forecasted value.

yhat_lower: This column represents the lower bound of the forecasted value.
 It provides a lower estimate of the expected value. Values are not expected to fall below this lower bound.

yhat_upper: This column represents the upper bound of the forecasted value.
 It provides an upper estimate of the expected value. Values are not expected to exceed this upper bound.

Mean Absolute Error (MAE): 46245.75
Mean Squared Error (MSE): 2994581610.29
Root Mean Squared Error (RMSE): 54722.77
Mean Absolute Percentage Error (MAPE): nan%
```


Crime Forecast with Prophet
Actual Data
Forecast
COVID-19 Outbreak
Lockdown Ends
Economic Recovery
Total Crimes
Year
200000
180000
160000
140000
120000
100000
2020
2021
2022
2023
2024
2025
2026