Los Angeles Crime Analysis and Predictive Modeling

IE6400 – FOUNDATIONS OF DATA ANALYTICS (FALL 2024)

PROJECT REPORT - 1

GROUP NUMBER 14

NIRMALKUMAR THIRUPALLIKRISHNAN KESAVAN (002334868)

MOHAMMED IBRAHIM LNU (002319064)

Introduction

Crime analysis is essential for understanding urban safety and informing law enforcement strategies. Los Angeles, a major city known for its cultural diversity and complex neighborhoods, faces a wide range of criminal activities. From gang-related incidents to property crimes, the city's crime landscape varies significantly across different areas and over time.

In this project, we aim to analyze crime patterns in Los Angeles using a dataset of reported crimes. The dataset includes detailed information on crime types, locations, and timestamps, allowing us to explore trends across neighborhoods and time periods. Our analysis will use Python for exploratory data analysis (EDA) to identify hidden patterns and correlations within the data.

We will create many visual representations to show the geographic distribution of crimes, making it easier to identify hotspots and problem areas. Additionally, time series plots will illustrate how crime rates have evolved over time, highlighting any seasonal trends or shifts. Beyond analyzing past data, we also aim to develop predictive models that estimate the likelihood of future crimes occurring in specific areas.

These insights will help law enforcement agencies make data-driven decisions, allocate resources efficiently, and improve crime prevention strategies. This project ultimately seeks to provide a comprehensive understanding of crime patterns in Los Angeles and contribute to a safer urban environment.

Task 1. Data Acquisition:

During the data collecting stage, we loaded and accessed the dataset for our study using the Python programming language and its pandas data manipulation tool. The provided link [Crime Data from 2020 to Present] yielded the dataset, which was named 'Crime_Data_from_2020_to_Present.csv'. The crime statistics in this CSV file cover the years 2020 through the present. We imported and organized the dataset using the pandas library to make manipulation and analysis easier. A DataFrame was created by reading the contents of the CSV file using the '**pd.read_csv'** function. The basis for our additional data preprocessing and exploratory data analysis is this DataFrame, which we have named **'crimedata'**.

```
import pandas as pd
pd.set_option('display.max_columns', None)

crimedata = pd.read_csv('Crime_Data.csv')
display(crimedata)
```

Task 2. Data Inspection:

One of the first tasks in any data analysis project is to carefully examine the dataset in order to comprehend its content and structure. The following are the main components of this process:

1. **Display the first few rows of the dataset:**

The starting rows of the dataset must be shown in order to begin comprehending it. This brief examination gives us an idea of the format and structure of the data. The Pandas **'head()'** function makes it simple to inspect the first few rows for a brief summary.

```
]: crimedata.head(10)
```

This will display the first 10 rows of the **'crimedata'** data frame. The results of this are attached below.

Results:

	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
0	190326475	03/01/2020 12:00:00 AM	03/01/2020 12:00:00 AM	2130	7	Wilshire	784	1	510	VEHICLE - STOLEN	NaN	0	M	O	101.0	STREET	NaN
1	200106753	02/09/2020 12:00:00 AM	02/08/2020 12:00:00 AM	1800	1	Central	182	1	330	BURGLARY FROM VEHICLE	1822 1402 0344	47	M	O	128.0	BUS STOP/LAYOVER (ALSO QUERY 124)	NaN
2	200320258	11/11/2020 12:00:00 AM	11/04/2020 12:00:00 AM	1700	3	Southwest	356	1	480	BIKE - STOLEN	0344 1251	19	X	X	502.0	MULTI-UNIT DWELLING (APARTMENT, DUPLEX, ETC)	NaN
3	200907217	05/10/2023 12:00:00 AM	03/10/2020 12:00:00 AM	2037	9	Van Nuys	964	1	343	SHOPLIFTING-GRAND THEFT ($950.01 & OVER)	0325 1501	19	M	O	405.0	CLOTHING STORE	NaN
4	220614831	08/18/2022 12:00:00 AM	08/17/2020 12:00:00 AM	1200	6	Hollywood	666	2	354	THEFT OF IDENTITY	1822 1501 0930 2004	28	M	H	102.0	SIDEWALK	NaN
5	231808869	04/04/2023 12:00:00 AM	12/01/2020 12:00:00 AM	2300	18	Southeast	1826	2	354	THEFT OF IDENTITY	1822 0100 0930 0929	41	M	H	501.0	SINGLE FAMILY DWELLING	NaN
6	230110144	04/04/2023 12:00:00 AM	07/03/2020 12:00:00 AM	900	1	Central	182	2	354	THEFT OF IDENTITY	0930 0929	25	M	H	502.0	MULTI-UNIT DWELLING (APARTMENT, DUPLEX, ETC)	NaN
7	220314085	07/22/2022 12:00:00 AM	05/12/2020 12:00:00 AM	1110	3	Southwest	303	2	354	THEFT OF IDENTITY	0100	27	F	B	248.0	CELL PHONE STORE	NaN
8	231309864	04/28/2023 12:00:00 AM	12/09/2020 12:00:00 AM	1400	13	Newton	1375	2	354	THEFT OF IDENTITY	0100	24	F	B	750.0	CYBERSPACE	NaN
9	211904005	12/31/2020 12:00:00 AM	12/31/2020 12:00:00 AM	1220	19	Mission	1974	2	624	BATTERY - SIMPLE ASSAULT	0416	26	M	H	502.0	MULTI-UNIT DWELLING (APARTMENT, DUPLEX, ETC)	400.0

2. Check the data types of each column:

Knowing each column's data type is essential. Data types dictate the data's storage and can influence further data analysis and processing. A method **'dtypes'** provided by Pandas allows you to verify the data types of every column in the dataset.

```
print(crimedata.dtypes)
```

This **'dtype'** method returns all the column names along with their data types details. The results of this code are attached below.

Results:

```
DR_NO               int64
Date Rptd          object
DATE OCC           object
TIME OCC            int64
AREA                int64
AREA NAME          object
Rpt Dist No         int64
Part 1-2            int64
Crm Cd              int64
Crm Cd Desc        object
Mocodes            object
Vict Age            int64
Vict Sex           object
Vict Descent       object
Premis Cd         float64
Premis Desc        object
Weapon Used Cd    float64
Weapon Desc        object
Status             object
Status Desc        object
Crm Cd 1          float64
Crm Cd 2          float64
Crm Cd 3          float64
Crm Cd 4          float64
LOCATION           object
Cross Street       object
LAT               float64
LON               float64
dtype: object
```

The dataset contains a variety of data types. Numerical fields like **DR_NO, TIME OCC, AREA, Rpt Dist No, Part 1-2, Crm Cd,** and **Vict Age** are stored as **integers (int64)**. Location coordinates **LAT** and **LON** as well as some other fields like **Premis Cd, Weapon Used Cd, and multiple crime code fields (Crm Cd 1, Crm Cd 2, etc.)** are stored as **floating-point numbers (float64)**. Textual data, such as **Date Rptd, DATE OCC, AREA NAME, Crm Cd Desc, Vict Sex, Vict Descent, Premis Desc, Weapon Desc,** and **LOCATION**, are stored as **objects (object)**, which is Pandas' type for string data.

3. Review column names, info and description:

A report's **"info"** part is a useful addition since it gives an easily understood but thorough overview of the dataset, including details on data kinds, missing values, memory utilization, and data quality. During data preparation and analysis, it facilitates well-informed decision-making and effective data exploration.

```
crimedata.info()
```

Result:

```
RangeIndex: 982638 entries, 0 to 982637
Data columns (total 27 columns):
 #   Column          Non-Null Count   Dtype
---  ------          --------------   -----
 0   DR_NO           982638 non-null  int64
 1   Date Rptd       982638 non-null  datetime64[ns]
 2   DATE OCC        982638 non-null  datetime64[ns]
 3   TIME OCC        982638 non-null  object
 4   AREA            982638 non-null  int64
 5   AREA NAME       982638 non-null  object
 6   Rpt Dist No     982638 non-null  int64
 7   Part 1-2        982638 non-null  int64
 8   Crm Cd          982638 non-null  int64
 9   Crm Cd Desc     982638 non-null  object
 10  Mocodes         837376 non-null  object
 11  Vict Age        982638 non-null  int64
 12  Vict Sex        982638 non-null  object
 13  Vict Descent    982638 non-null  object
 14  Premis Cd       982624 non-null  float64
 15  Premis Desc     982053 non-null  object
 16  Weapon Used Cd  982638 non-null  object
 17  Weapon Desc     982638 non-null  object
 18  Status          982637 non-null  object
 19  Status Desc     982638 non-null  object
 20  Crm Cd 1        982638 non-null  float64
 21  LOCATION        982638 non-null  object
 22  Cross Street    982638 non-null  object
 23  LAT             982638 non-null  float64
 24  LON             982638 non-null  float64
 25  Day of Week     982638 non-null  object
 26  Hour OCC        982638 non-null  int32
dtypes: datetime64[ns](2), float64(4), int32(1), int64(6), object(14)
memory usage: 198.7+ MB
```

A brief synopsis of the dataset's numerical columns is given by Pandas **'describe()'** function, which displays important statistics including **count, mean, standard deviation, minimum, maximum, and quartile ranges.** It's a helpful method for gaining a general understanding of the distribution and spread of numerical data, which aids in spotting patterns or anomalies. It gives the **frequency, highest value, count, and unique values** for **non-numerical data.**

```
crimedata.describe()
```

The results of the **'describe()'** are attached below

Results:

	DR_NO	TIME OCC	AREA	Rpt Dist No	Part 1-2	Crm Cd	Vict Age	Premis Cd	Weapon Used Cd	Crm Cd 1
count	9.826380e+05	982638.000000	982638.000000	982638.000000	982638.000000	982638.000000	982638.000000	982624.000000	326167.000000	982627.000000
mean	2.197437e+08	1338.945426	10.700277	1116.459887	1.404253	500.823555	29.079817	306.133008	363.840882	500.578668
std	1.294954e+07	651.537830	6.107808	610.893787	0.490747	206.211940	21.970094	219.053795	123.684663	206.010361
min	8.170000e+02	1.000000	1.000000	101.000000	1.000000	110.000000	-4.000000	101.000000	101.000000	110.000000
25%	2.106089e+08	900.000000	5.000000	587.000000	1.000000	331.000000	0.000000	101.000000	311.000000	331.000000
50%	2.208146e+08	1420.000000	11.000000	1141.000000	1.000000	442.000000	30.000000	203.000000	400.000000	442.000000
75%	2.309153e+08	1900.000000	16.000000	1617.000000	2.000000	626.000000	44.000000	501.000000	400.000000	626.000000
max	2.499253e+08	2359.000000	21.000000	2199.000000	2.000000	956.000000	120.000000	976.000000	516.000000	956.000000

Task 3. Data Cleaning:

1. Identifying and handling missing data:

Identifying the missing value percentage:

To deal with missing data, we use **'crimedata.isnull().sum() / len(crimedata) * 100'** to get the percentage of null values in each column. **'Sum()'** counts the number of 'True' values in each column, while **'isnull()'** looks for missing values in every column and marks them as 'True'. We calculated the proportion of missing data for each column by dividing this count by the total number of rows **'(len(crimedata))'** and then multiplying the result by **100**. This approach helps us make judgments about data cleansing by giving us a clear picture of the percentage of missing values.

```
print(crimedata.isnull().sum() / len(crimedata) * 100)
```

Results:

```
DR_NO              0.000000
Date Rptd          0.000000
DATE OCC           0.000000
TIME OCC           0.000000
AREA               0.000000
AREA NAME          0.000000
Rpt Dist No        0.000000
Part 1-2           0.000000
Crm Cd             0.000000
Crm Cd Desc        0.000000
Mocodes           14.782860
Vict Age           0.000000
Vict Sex          14.089115
Vict Descent      14.090235
Premis Cd          0.001425
Premis Desc        0.059534
Weapon Used Cd    66.807003
Weapon Desc       66.807003
Status             0.000102
Status Desc        0.000000
Crm Cd 1           0.001119
Crm Cd 2          92.990806
Crm Cd 3          99.764817
Crm Cd 4          99.993487
LOCATION           0.000000
Cross Street      84.546802
LAT                0.000000
LON                0.000000
dtype: float64
```

The table displays the percentage of missing values for each column in the dataset. Most columns, such as **DR_NO**, **Date Rptd**, **DATE OCC**, and **Crm Cd**, have no missing values (0.000000%). However, some columns, like **Mocodes** (14.78%), **Vict Sex** (14.09%), and **Weapon Used Cd** (66.87%), have a significant proportion of missing entries. The columns **Crm Cd 3** and **Crm Cd 4** have the highest percentage of missing data, at around 99%. This overview indicates which columns require attention during the data cleaning process.

Handling the missing values:

In order to deal with missing values, we used **'Unknown'** to fill in the blanks in important category fields **(such as Vict Sex, Vict Descent, and Weapon Desc)**. We first changed the **Weapon Used Cd** column to a categorical type before adding the missing data. Furthermore, we eliminated columns such as **Crm Cd 2, 3, and 4** that had too much

missing data. And we used **mean imputation** for the column **Crm Cd 1.** We'll see the process in detail for better understanding.

```
# Dropping columns
crimedata = crimedata.drop(columns=['Crm Cd 2', 'Crm Cd 3', 'Crm Cd 4'])

# Fill NaN values for categorical columns
crimedata['Vict Sex'] = crimedata['Vict Sex'].fillna('Unknown')
crimedata['Vict Descent'] = crimedata['Vict Descent'].fillna('Unknown')
crimedata['Cross Street'] = crimedata['Cross Street'].fillna('Unknown')
crimedata['Weapon Desc'] = crimedata['Weapon Desc'].fillna('Unknown')

# Convert 'Weapon Used Cd' to object type before filling NaN
crimedata['Weapon Used Cd'] = crimedata['Weapon Used Cd'].astype('object').fillna('Unknown')

crimedata['Weapon Used Cd'].fillna('Unknown', inplace=True)
```

Dropping Columns: Since the columns **Crm Cd 2, Crm Cd 3**, and **Crm Cd 4** had a large percentage of missing data (**around 99%)** and weren't essential to our research, we dropped them. Furthermore, since the **'Mocodes'** column column was ignored as it wasn't relevant to our study.

Filling NaN Values for Categorical Columns: We substituted **'Unknown'** for missing values in categorical columns such as **Vict Sex, Vict Descent, Cross Street,** and **Weapon Desc**. This guarantees that these columns can still be used for analysis while maintaining the integrity of the data.

Handling 'Weapon Used Cd': Since this column is **numerical** one, but acts as a **categorical**, we first converted it to **object** type, then filled missing values with **'Unknown'** to maintain consistency with other categorical fields.

Mean() Imputation for Crm Cd 1: Since there weren't many missing entries, we used the **mean()** to fill in the blanks in **Crm Cd 1** so that the distribution of the data as a whole wouldn't be greatly affected.

After performing all these steps to handle the missing values, we now have a dataframe that is free from missing entries. This cleaned dataframe will be used for further exploratory data analysis and forecasting. The results after handling the missing values are shown below:

Results:

```
DR_NO              0.000000
Date Rptd          0.000000
DATE OCC           0.000000
TIME OCC           0.000000
AREA               0.000000
AREA NAME          0.000000
Rpt Dist No        0.000000
Part 1-2           0.000000
Crm Cd             0.000000
Crm Cd Desc        0.000000
Mocodes           14.782860
Vict Age           0.000000
Vict Sex           0.000000
Vict Descent       0.000000
Premis Cd          0.001425
Premis Desc        0.059534
Weapon Used Cd     0.000000
Weapon Desc        0.000000
Status             0.000102
Status Desc        0.000000
Crm Cd 1           0.000000
LOCATION           0.000000
Cross Street       0.000000
LAT                0.000000
LON                0.000000
dtype: float64
```

The table displays the percentage of missing values in the updated dataset after handling most of the missing data. All columns now show **0%** missing values, except for **Mocodes**, which still has around **14.78%** missing data. This indicates that our cleaning process successfully addressed missing values in all other columns, making the dataset ready for further analysis.

2. Removing Duplicate Rows :

Pandas **'duplicated()'** method looks for duplicate rows in the dataset; if it finds any, it returns True; if not, it returns False. The number of duplicate rows in the dataset is then determined by counting the total number of True values using the **'sum()**' method. For our dataframe, the **result is 0**, meaning there are no duplicate rows in the dataset. This

ensures that our data is unique and free from redundancy, making it ready for further analysis.

Code and Result:

```
crimedata.duplicated().sum()
```

```
0
```

3. Converting Data Types:

In order to guarantee that the data is in a consistent format for analysis, filtering, and comparisons, date and time fields must be converted. Time conversions help in accurately recording and adjusting the times at which events occur, and properly typed dates allows chronological analysis.

To manage and work with date and time objects effectively, we import **'datetime'** from Python's **datetime module**. Working with time-related functions in the dataset requires this import in order to ensure proper formatting and conversion.

```
from datetime import datetime as datetime
crimedata['DATE OCC'] = pd.to_datetime(crimedata['DATE OCC'], errors='coerce')
crimedata['Date Rptd'] = pd.to_datetime(crimedata['Date Rptd'], errors='coerce')
crimedata['TIME OCC'] = crimedata['TIME OCC'].astype(str).str.zfill(4)
crimedata['TIME OCC'] = pd.to_datetime(crimedata['TIME OCC'], format='%H%M').dt.time
```

Explanations:

Date Conversion: Using **pd.to_datetime()**, the code transforms the **DATE OCC** and **Date Rptd** columns from string to an appropriate datetime format. In order to prevent conversion issues, the **errors='coerce'** option makes sure that any inaccurate date values are changed to **NaT** (Not a Time). Easy tasks like date-based filtering and extracting particular components (year, month, etc.) are made possible by this uniformity.

Time Conversion: The code first transforms the numeric values to strings for the **TIME OCC** column. It then uses **zfill(4)** to make sure that all times are expressed with four digits **(for example, '0900' for 9:00 AM)**. This text format is then transformed into a time object with the given **%H%M forma**t (hours and minutes) using **pd.to_datetime()**. Lastly, the**.dt.time** ensures that the time is correctly structured for additional analysis by extracting only the time component.

Results:

	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	De
0	190326475	2020-03-01	2020-03-01	21:30:00	7	Wilshire	784	1	510	VEHICLE - STOLEN	NaN	0	M	
1	200106753	2020-02-09	2020-02-08	18:00:00	1	Central	182	1	330	BURGLARY FROM VEHICLE	1822 1402 0344	47	M	
2	200320258	2020-11-11	2020-11-04	17:00:00	3	Southwest	356	1	480	BIKE - STOLEN	0344 1251	19	X	
3	200907217	2023-05-10	2020-03-10	20:37:00	9	Van Nuys	964	1	343	SHOPLIFTING-GRAND THEFT ($950.01 & OVER)	0325 1501	19	M	
4	220614831	2022-08-18	2020-08-17	12:00:00	6	Hollywood	666	2	354	THEFT OF IDENTITY	1822 1501 0930 2004	28	M	

A preview of the cleaned dataset is shown in the output, with important columns like **Date Rptd, DATE OCC**, and **TIME OCC**. To make chronological crime analysis easier, dates are now provided in the **YYYY-MM-DD** format, while times are shown in the **HH:MM:SS** format.

Task 4. Exploratory Data Analysis (EDA):

Finding patterns, trends, and connections in datasets is the goal of exploratory data analysis, or EDA. Prior to using more complex analytics or modeling, it aids in our comprehension of the dataset's structure and salient features. The following will be our main focus for this EDA: overall crime trends, seasonal patterns, the most common crime types, regional differences in crime, correlation analysis, examine day-of-the-week crime frequencies, and assess the impact of major events on crime rates. Additionally, we'll identify outliers and anomalies, explore demographic factors related to crime, and employ time series forecasting to predict future trends. Now we'll dive into the dataset by performing each analysis one after the other.

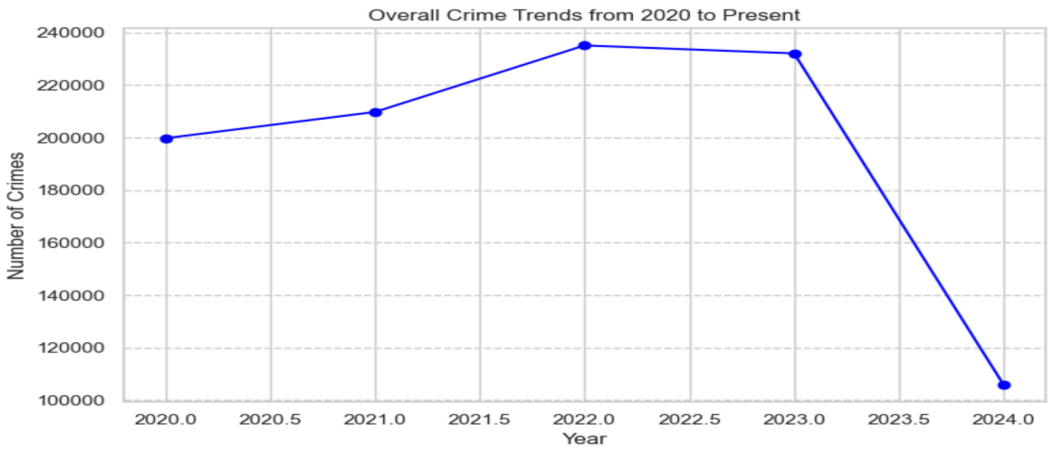
1. Overall Crime Trends:

In this study, we will plot the annual number of crimes reported in order to visualize the general trends in crime from 2020 to the present. In order to find trends or variations in crime rates over time, we group the crime data by year and calculate the incidents. This will help us understand whether crime levels have increased, decreased, or remained stable since 2020. The resulting plot will provide a clear, year-by-year overview of crime trends for further interpretation.

```
import matplotlib.pyplot as plt

crime_trends = crimedata[crimedata['DATE OCC'].dt.year >= 2020].groupby(crimedata['DATE OCC'].dt.year).size()

plt.figure(figsize=(8, 5))
plt.plot(crime_trends.index, crime_trends.values, color='blue', marker='o', linestyle='-')
plt.xlabel('Year')
plt.ylabel('Number of Crimes')
plt.title('Overall Crime Trends from 2020 to Present')
plt.grid(axis='y', linestyle='--')
plt.tight_layout()
plt.show()
```

We used **matplotlib.pyplot** to visualize crime trends from 2020 onwards. The data was filtered and grouped by year using **crime_trends = crimedata[crimedata['DATE OCC'].dt.year >= 2020].groupby(crimedata['DATE OCC'].dt.year).size()** to calculate the total crimes per year. We then created a plot with **plt.figure(figsize=(8, 5))**, and the data was plotted with **plt.plot(crime_trends.index, crime_trends.values, color='blue', marker='o', linestyle='-')**.We added Labels using **plt.xlabel('Year')** and **plt.ylabel('Number of Crimes')**, with a title set using **plt.title('Overall Crime Trends from 2020 to Present')**. Then grid lines were created using **plt.grid(axis='y', linestyle='--')**, adjusted the layout with **plt.tight_layout()**, and displayed the plot with **plt.show()**.

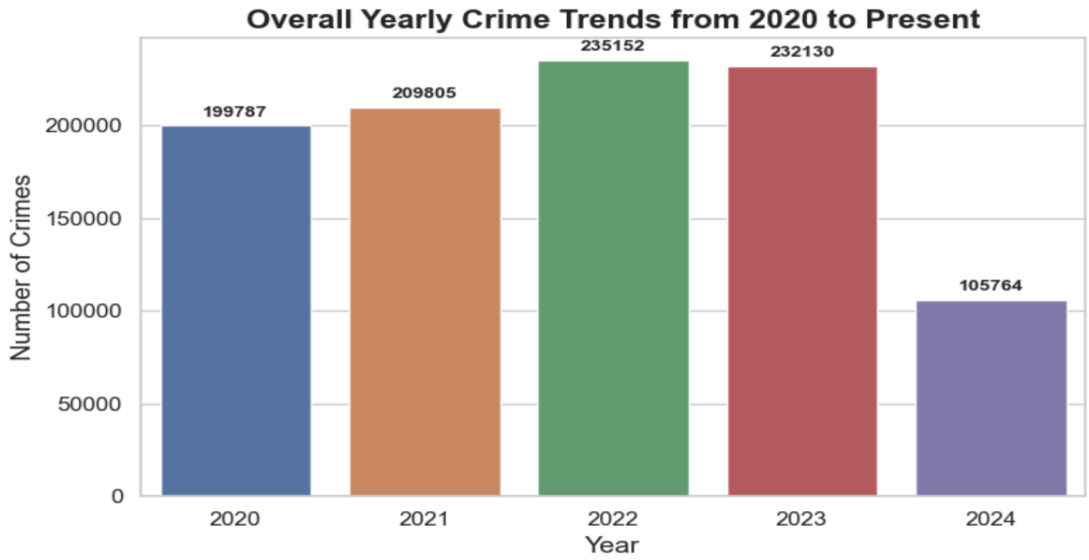
Result and Analysis:



The line plot displays the overall number of crimes reported per year and shows the trends in crime from **2020 to the present**. Prior to **reaching a peak in 2022**, there is a **minor rise** in crime from **2020 to 2021**. The crime rate stays **largely the same after 2022 until 2023**, when it starts to **noticeably drop**. Data for the current year may not yet reflect the entire year, which could explain the 2024 decline. So there have been some fluctuations in crime rates over the last few years.

We have also visualized crime trends using bar charts, which provide a clear comparison of crime numbers across different years. The code and output are shown below

```
import seaborn as sns

crimedata_filtered['Year'] = crimedata_filtered['DATE OCC'].dt.year

yearly_crime_trends = crimedata_filtered['Year'].value_counts().sort_index()

sns.set(style="whitegrid")

plt.figure(figsize=(8, 5))
ax = sns.barplot(x=yearly_crime_trends.index, y=yearly_crime_trends.values, hue=yearly_crime_trends.index, dodge=False, palette='deep', legend=False)
ax.set_title('Overall Yearly Crime Trends from 2020 to Present', fontsize=16, fontweight='bold')
ax.set_xlabel('Year', fontsize=14)
ax.set_ylabel('Number of Crimes', fontsize=14)
plt.xticks(fontsize=12)
plt.yticks(fontsize=12)

for i, value in enumerate(yearly_crime_trends.values):
    ax.text(i, value + 5000, str(value), ha='center', fontsize=10, fontweight='bold')

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



Crime **increased gradually between 2020 and 2022**, reaching a peak of **235,152 crimes from 2020's 199,787 offenses**. The l**ittle dip in 2023 (232,130 crimes)** after

2022 is followed by a steep drop in 2024 (105,764 crimes), which might be the result of incomplete data for the year.

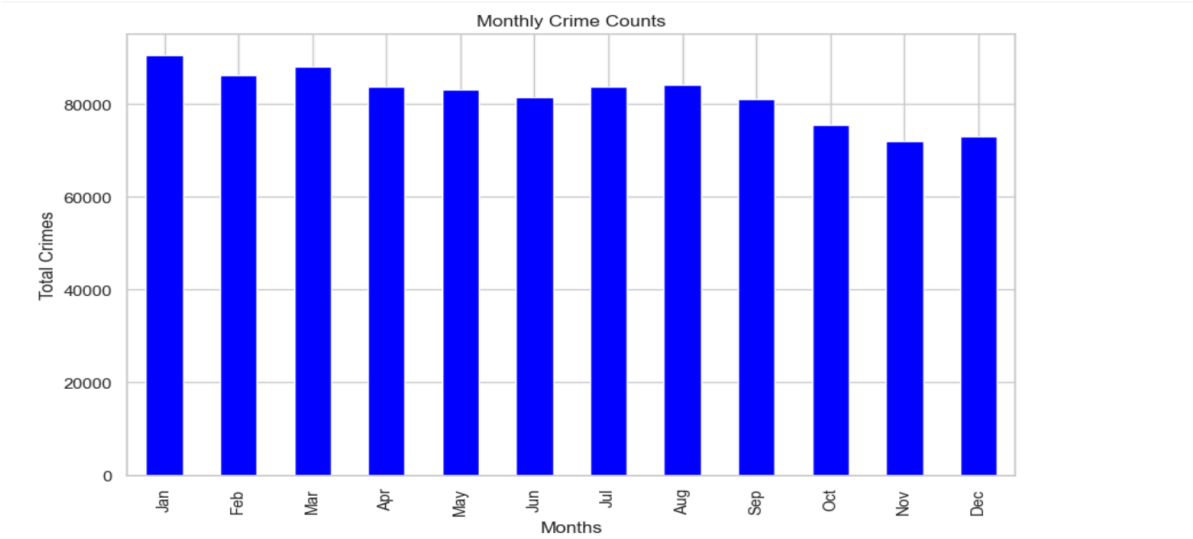
2. Seasonal Patterns in Crime data:

Our goal is to examine and display the dataset's monthly crime distribution. Using the **dt.month attribute**, we first extract the month from the **Date Rptd column**. Next, we use the **value_counts()** function to determine the number of crimes that took place in each month. **Sort_index()** is used to arrange the data in chronological order.

Next, we use a **bar plot** to depict this data, with the y-axis showing the total number of offenses and the x-axis representing the months (January through December). The plot is stylized with blue bars to reflect monthly crime counts, and the months are labeled using the **plt.xticks()** method.

```
crime_data['Month'] = crime_data['Date Rptd'].dt.month
Counts_Monthly = crime_data['Month'].value_counts().sort_index()

plt.figure(figsize=(10, 6))
Counts_Monthly.plot(kind='bar', color='blue')
plt.title('Monthly Crime Counts')
plt.xlabel('Months')
plt.ylabel('Total Crimes')
plt.xticks(range(0, 12), ['Jan', 'Feb', 'Mar', 'Apr', 'May', 'Jun', 'Jul', 'Aug', 'Sep', 'Oct', 'Nov', 'Dec'])
plt.show()
```

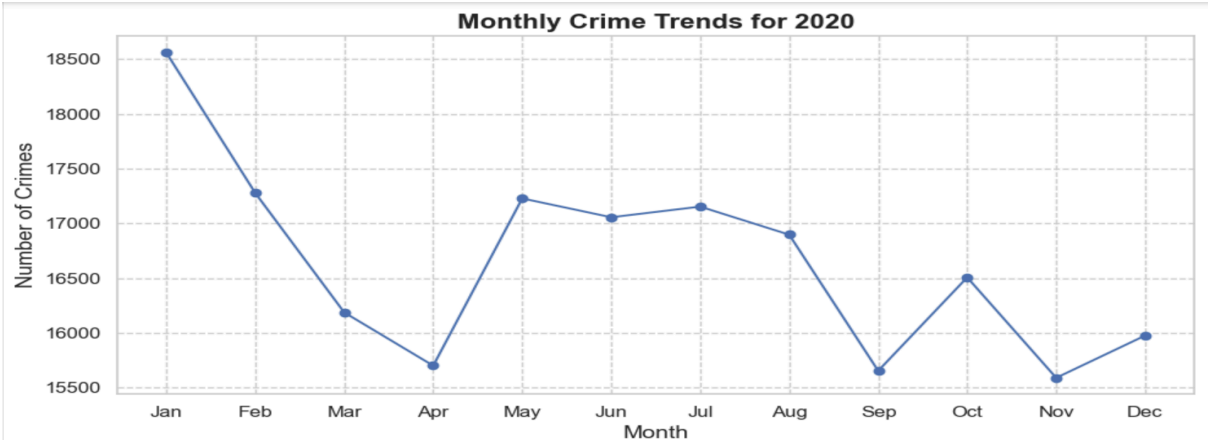
Result and Analysis:

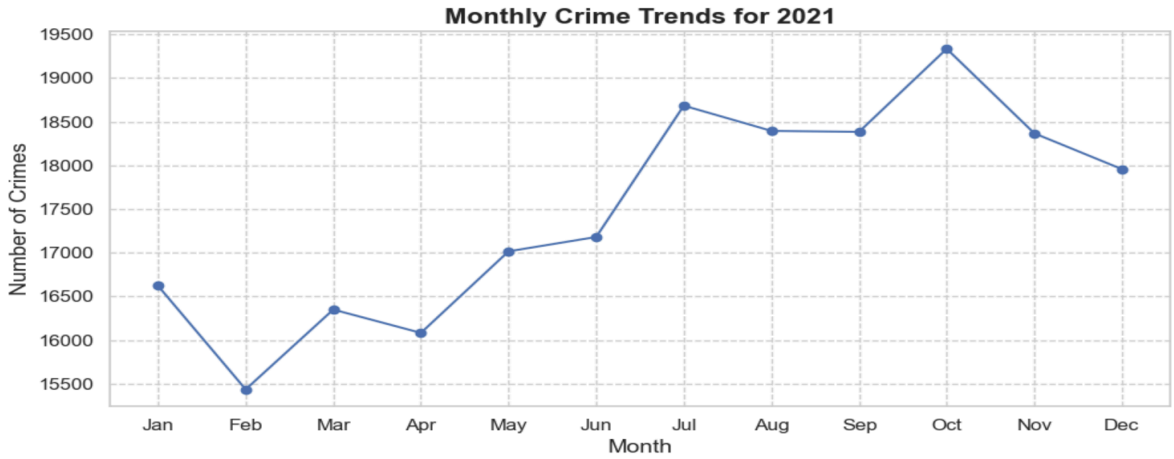


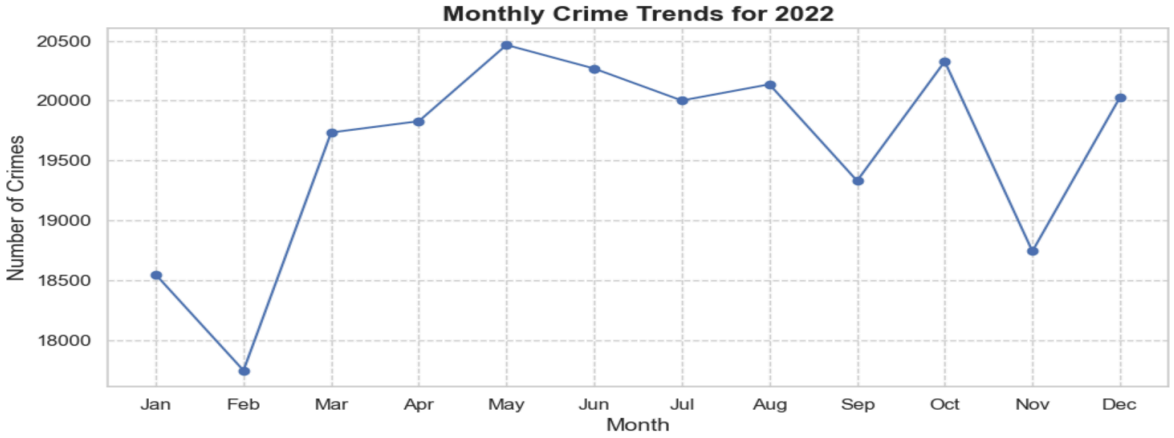
A clear comparison of crime counts throughout the year is provided by the bar chart, which shows the total number of crimes reported each month. Crime rates are **highest in January** and then **slightly lower in February** and **later months**. Although there is a **noticeable drop** beginning in **September and lasting through December**, the number of crimes is **comparatively constant** over the main part of the year, from **May to August**. According to this seasonal trend, there may be a **minor increase in crime during the first few months of the year**, followed by a **slow decrease as the year draws to a close**.

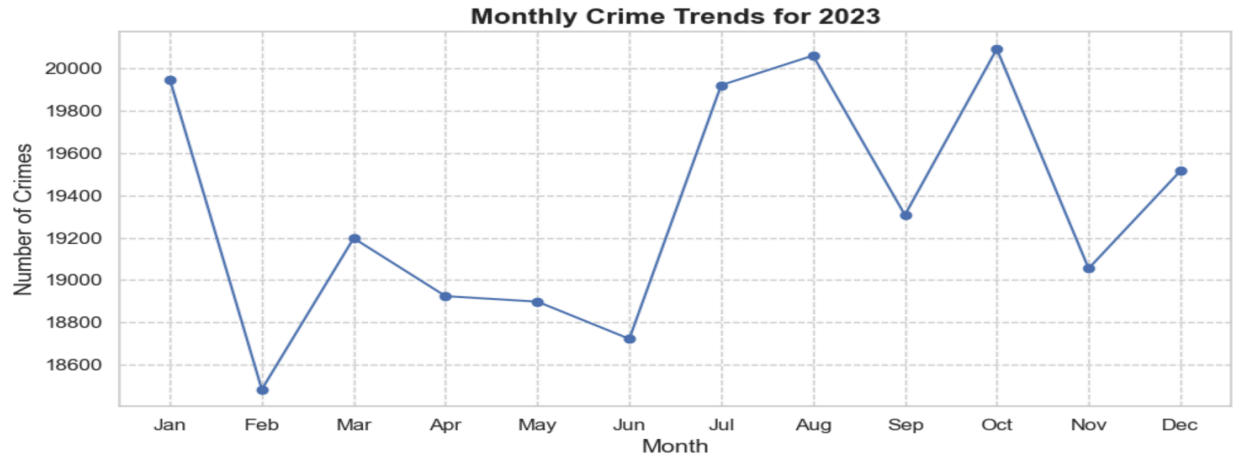
We also plotted the seasonal trends of each year separately. In this analysis, we extracted the **Year** and **Month** from the **DATE OCC** column and grouped the crime data by year and month to identify monthly crime trends. Using **crimedata_filtered.groupby(['Year', 'Month']).size()**, we calculated the total number of crimes for each month. To visualize the seasonal crime trends for each year, we created a separate **line plot** for each year from 2020 to 2024. The **Seaborn** style was applied to ensure the graphs are clean and professional-looking. Each line plot displays the monthly crime trends for a specific year, helping us observe how crime patterns vary month-to-month across different years. The code and outputs are shown below.

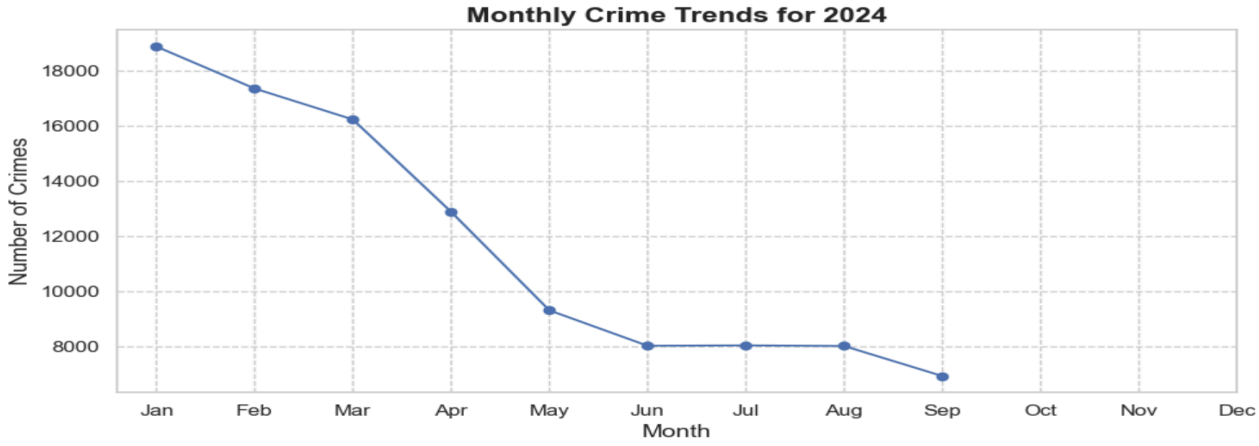
```
import seaborn as sns

crimedata_filtered['Year'] = crimedata_filtered['DATE OCC'].dt.year
crimedata_filtered['Month'] = crimedata_filtered['DATE OCC'].dt.month
monthly_crime_trends = crimedata_filtered.groupby(['Year', 'Month']).size().reset_index(name='Crime Count')

sns.set(style="whitegrid")
years = monthly_crime_trends['Year'].unique()

for year in years:
    crime_year = monthly_crime_trends[monthly_crime_trends['Year'] == year]

    plt.figure(figsize=(10, 5))
    plt.plot(crime_year['Month'], crime_year['Crime Count'], marker='o', linestyle='-', color='b')
    plt.title(f'Monthly Crime Trends for {year}', fontsize=16, fontweight='bold')
    plt.xlabel('Month', fontsize=14)
    plt.ylabel('Number of Crimes', fontsize=14)
    plt.xticks(ticks=range(1, 13), labels=['Jan', 'Feb', 'Mar', 'Apr', 'May', 'Jun', 'Jul', 'Aug', 'Sep', 'Oct', 'Nov', 'Dec'], fontsize=12)
    plt.yticks(fontsize=12)
    plt.grid(True, linestyle='--')
    plt.tight_layout()
    plt.show()
```


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


Monthly Crime Trends for 2021
Number of Crimes
19500
19000
18500
18000
17500
17000
16500
16000
15500
Jan
Feb
Mar
Apr
May
Jun
Jul
Aug
Sep
Oct
Nov
Dec
Month


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





3. Most Common Crime Type and its Trends over time:

Now, we concentrate on determining which crime categories are most frequently occurring in the dataset. We can learn more about the kinds of crimes that happen most frequently by looking at the frequency of each crime category. This will assist us in identifying criminal trends and setting priorities for more research.

```
crimefreq=crimedata['Crm Cd Desc'].value_counts()
crimefreq
```

The code **crimefreq = crimedata['Crm Cd Desc'].value_counts()** calculates how often each crime description appears in the dataset. The **.value_counts()** function arranges

the crimes in descending order, with the most frequent crimes listed first. We are focused on finding the **top 5 common crimes and its trends over time.**

Result and Analysis:

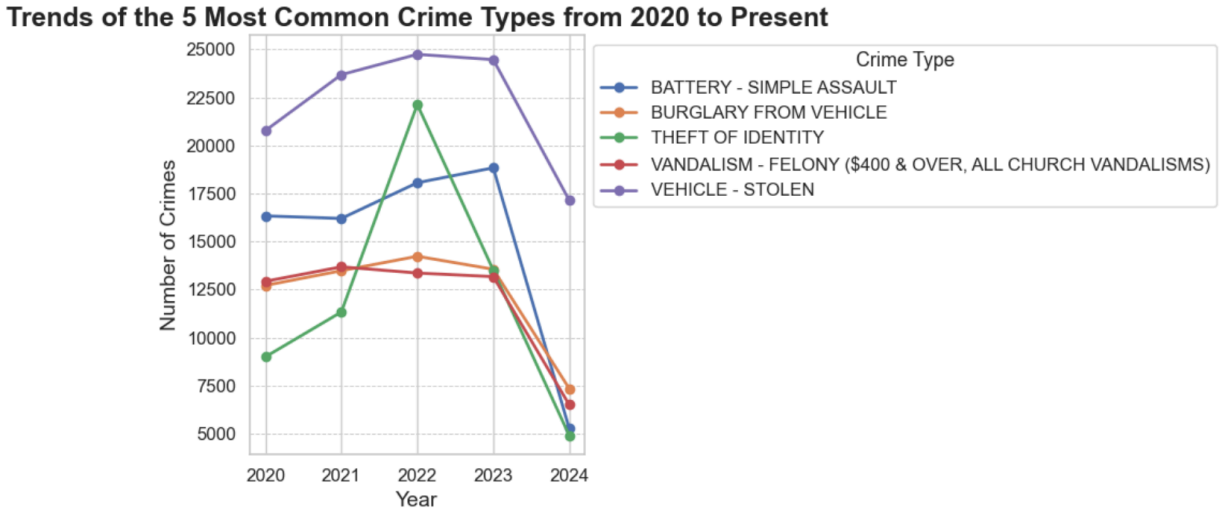
```
Crm Cd Desc
VEHICLE - STOLEN                                        110804
BATTERY - SIMPLE ASSAULT                                 74688
BURGLARY FROM VEHICLE                                    61324
THEFT OF IDENTITY                                        60867
VANDALISM - FELONY ($400 & OVER, ALL CHURCH VANDALISMS)  59639
                                                           ...
FIREARMS EMERGENCY PROTECTIVE ORDER (FIREARMS EPO)           5
FIREARMS RESTRAINING ORDER (FIREARMS RO)                     4
DISHONEST EMPLOYEE ATTEMPTED THEFT                           4
TRAIN WRECKING                                               1
DRUNK ROLL - ATTEMPT                                         1
Name: count, Length: 140, dtype: int64
```

In the output, **VEHICLE - STOLEN** is the most reported crime with **110,804** instances, followed by **BATTERY - SIMPLE ASSAULT (74,688)** and **BURGLARY FROM VEHICLE (61,324).** On the other hand, rare crimes like **TRAIN WRECKING** and **DRUNK ROLL - ATTEMPT** occur only once. This analysis highlights both the most and least common crimes in the dataset.

For a clear visualization we utilized the **seaborn librar**y to visualize the top 5 common crime types by line plots. The Code and output are shown below.

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

crimedata['DATE OCC'] = pd.to_datetime(crimedata['DATE OCC'], errors='coerce')
crimedata_filtered = crimedata[crimedata['DATE OCC'].dt.year >= 2020]
top_5_crimes = crimedata_filtered['Crm Cd Desc'].value_counts().head(5).index
top_5_crime_data = crimedata_filtered[crimedata_filtered['Crm Cd Desc'].isin(top_5_crimes)].copy()
top_5_crime_data.loc[:, 'Year'] = top_5_crime_data['DATE OCC'].dt.year
crime_trends = top_5_crime_data.groupby(['Year', 'Crm Cd Desc']).size().reset_index(name='Crime Count')
crime_trends_pivot = crime_trends.pivot(index='Year', columns='Crm Cd Desc', values='Crime Count').fillna(0)

sns.set(style="whitegrid")
plt.figure(figsize=(12, 5))
for crime_type in crime_trends_pivot.columns:
    plt.plot(crime_trends_pivot.index, crime_trends_pivot[crime_type], marker='o', linestyle='-', linewidth=2, label=crime_type)

plt.title('Trends of the 5 Most Common Crime Types from 2020 to Present', fontsize=18, weight='bold')
plt.xlabel('Year', fontsize=14)
plt.ylabel('Number of Crimes', fontsize=14)
plt.xticks(ticks=crime_trends_pivot.index, fontsize=12)
plt.yticks(fontsize=12)
plt.legend(title='Crime Type', fontsize=12, title_fontsize=13, loc='upper left', bbox_to_anchor=(1, 1))
plt.grid(axis='y', linestyle='--', linewidth=0.7)
plt.tight_layout(rect=[0, 0, 0.85, 1])
plt.show()
```

First, the **DATE OCC** column is converted to datetime using **pd.to_datetime()**, and the dataset is filtered for records after 2020. We then use **.value_counts().head(5)** to identify the top 5 crime types. The filtered data is grouped by year and crime type using **.groupby(['Year', 'Crm Cd Desc']).size()** to count the occurrences for each year. A pivot table is created using **.pivot()** to reshape the data for easy plotting. We then set the Seaborn style with **sns.set(style="whitegrid")** and create a line plot for each crime type with **plt.plot()**. The plot is customized with a title, axis labels, and a legend using **plt.title(), plt.xlabel(), and plt.legend().**



The annual trends of the five most common crime categories from 2020 to the present are illustrated in the line graph. The frequency of **vehicle- stolen** is the highest overall, reaching its **peak in 2022** before **sharply declining in 2023 and 2024**. **Battery-Simple Assault** and **Vehicle Burglary** show more consistent patterns, with **minor rises** and **falls over time**. Both **Identity Theft** and **Vandalism** exhibit **rising crime rates in 2022** before **sharply declining by 2024**.

The **significant decline in crime rates in 2024** may be due to incomplete data for the year, but **the 2022 peak points to a time of increased criminal activity** for these common crimes.

4. Regional Differences:

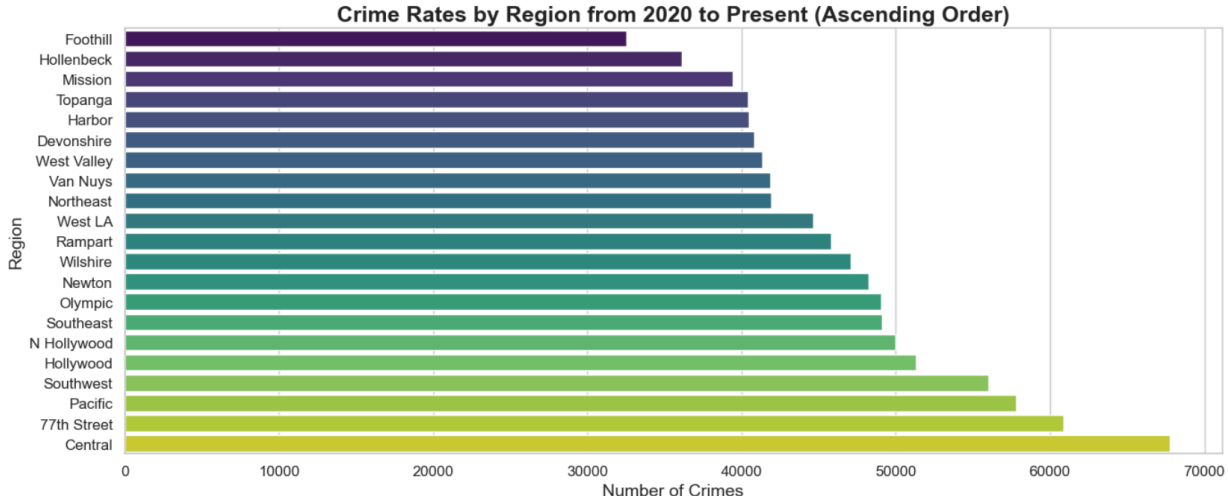
Now we focus on examining crime rates in various locations in order to identify regional differences in crime data starting from 2020. In order to evaluate crime levels across various regions and identify those with the greatest or lowest crime counts, the crime

data is grouped by region. A bar plot of the results will be used to make it simple to understand how crime activity varies by region. Additionally we'll further analyzing and visualizing the top 5 most frequent crimes within the top 5 cities or regions.

```
crimedata['DATE OCC'] = pd.to_datetime(crimedata['DATE OCC'], errors='coerce')
crimedata_filtered = crimedata[crimedata['DATE OCC'].dt.year >= 2020]

region_crime_counts = crimedata_filtered.groupby('AREA NAME').size().reset_index(name='Crime Count').sort_values(by='Crime Count', ascending=True)

sns.set(style="whitegrid")
plt.figure(figsize=(14, 6))
sns.barplot(x='Crime Count', y='AREA NAME', data=region_crime_counts, palette='viridis')
plt.title('Crime Rates by Region from 2020 to Present (Ascending Order)', fontsize=18, weight='bold')
plt.xlabel('Number of Crimes', fontsize=14)
plt.ylabel('Region', fontsize=14)
plt.xticks(fontsize=12)
plt.yticks(fontsize=12)
plt.tight_layout()
plt.show()
```

The code begins by converting the **DATE OCC** column to a datetime format using **pd.to_datetime()** and filtering the dataset for crimes occurring from 2020 onwards with **crimedata_filtered = crimedata[crimedata['DATE OCC'].dt.year >= 2020]**. We then group the data by region using **.groupby('AREA NAME').size()** to count the number of crimes in each region. The result is reset with **.reset_index()** and sorted in ascending order using **.sort_values(by='Crime Count')**. Next, we set the style to **whitegrid** with **sns.set()**, and a bar plot is created using **sns.barplot()**, where the x-axis represents the **Crime Count** and the y-axis represents the **Region**. The palette **'viridis'** is used for the color scheme, and the plot is labeled and customized using **plt.title()**, **plt.xlabel()**, and **plt.ylabel()**. Finally, **plt.show()** displays the chart, showing crime rates across different regions.

Result and Analysis:



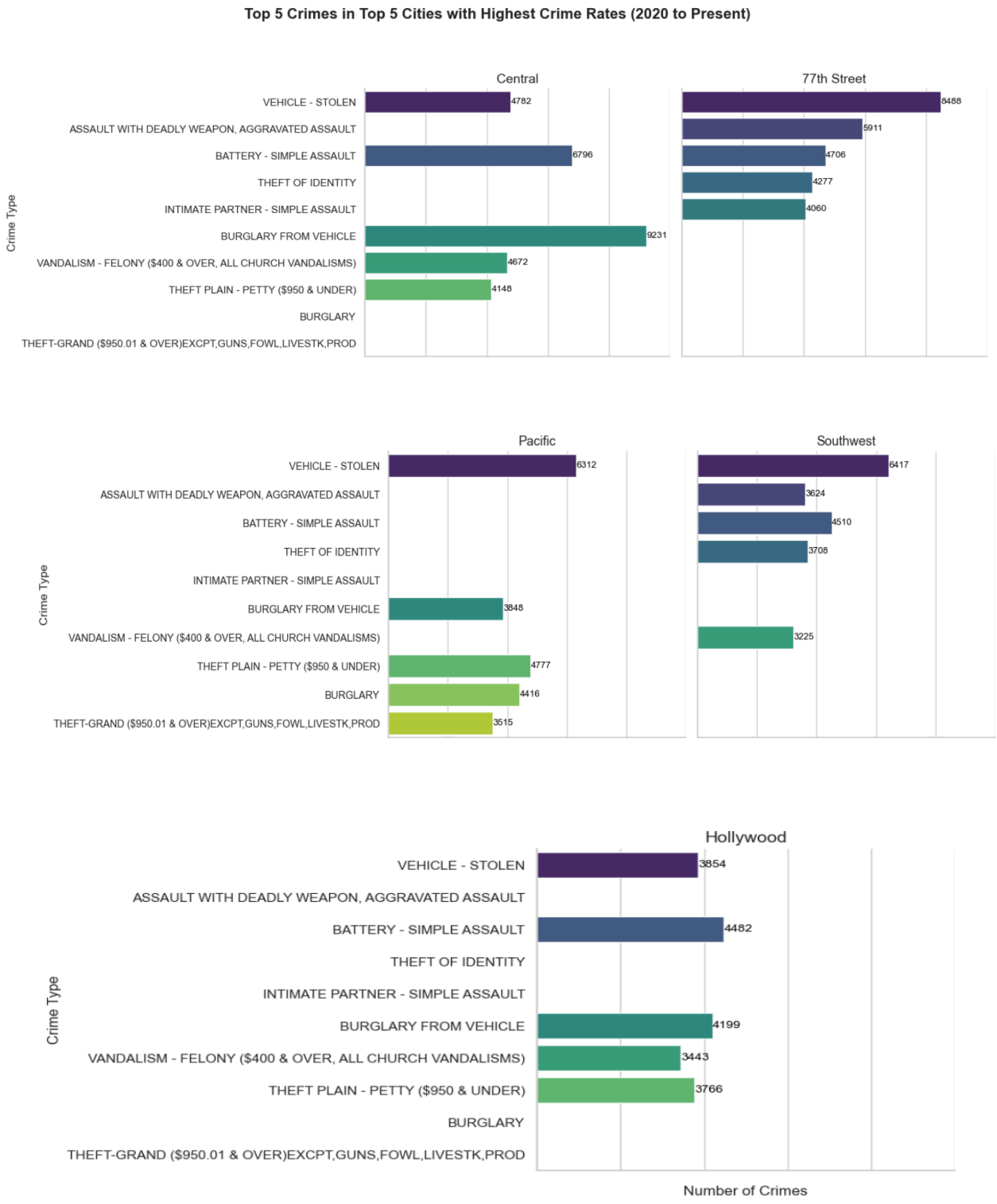
From the bar chart we can see that Central**, 77th Street, Pacific, Southwest,** and **Hollywood** are the **top 5 areas with the highest crime rates**. With around **70,000** crimes, **Central** is far ahead, followed by **77th Street** and **Pacific**, both of which have about **60,000**. **Hollywood** and **Southwest**, both of which have crime rates **above 50,000**, complete the top 5. From 2020 to the present, these regions have had the greatest rates of criminal activity.

Mission, Hollenbeck, and **Foothill** are the **three areas** with the **lowest crime rates**. Just **under 20,000** crimes are reported in **Foothill**, which is followed closely by **Hollenbeck** and **Mission**, both of which have **slightly over 20,000** crimes. Compared to the top five regions, these locations are comparatively safer since they have the lowest rates of criminal activity from 2020 to the present.

We have also analyzed and visualized the top 5 crimes in the top 5 regions with the highest crime rates from 2020 to the present. The code is shown below.

```
crimedata_filtered = crimedata[crimedata['DATE OCC'].dt.year >= 2020]
region_crime_counts = crimedata_filtered.groupby('AREA NAME').size().reset_index(name='Crime Count').sort_values(by='Crime Count', ascending=False)
top_5_regions = region_crime_counts.head(5)['AREA NAME']
top_5_regions_data = crimedata_filtered[crimedata_filtered['AREA NAME'].isin(top_5_regions)]
top_5_crimes_by_region = top_5_regions_data.groupby(['AREA NAME', 'Crm Cd Desc']).size().reset_index(name='Crime Count')
top_5_crimes_by_region = top_5_crimes_by_region.sort_values(['AREA NAME', 'Crime Count'], ascending=[True, False]).groupby('AREA NAME').head(5)
region_order = top_5_regions

sns.set(style="whitegrid")
g = sns.catplot(x='Crime Count', y='Crm Cd Desc', col='AREA NAME', data=top_5_crimes_by_region,
                kind='bar', col_wrap=2, height=5, aspect=1.5, palette='viridis', col_order=region_order)

g.fig.suptitle('Top 5 Crimes in Top 5 Cities with Highest Crime Rates (2020 to Present)', fontsize=16, weight='bold')
g.set_titles(col_template="{col_name}", size=14)
g.set_axis_labels("Number of Crimes", "Crime Type")
g.set_xticklabels(rotation=45)

for ax in g.axes.flat:
    for p in ax.patches:
        ax.text(p.get_width() + 10, p.get_y() + p.get_height() / 2, int(p.get_width()),
                ha='left', va='center', fontsize=10, color='black')

plt.tight_layout()
plt.subplots_adjust(top=0.9)
plt.show()
```

We first filter the dataset for crimes from 2020 onwards using **crimedata_filtered** and then group the data by **AREA NAME** to find the top 5 regions with the highest crime counts via **.groupby('AREA NAME').size().reset_index()**. Then, the data is filtered for these top regions using **isin(top_5_regions)**. The top crimes in each of these regions are identified with **.groupby(['AREA NAME', 'Crm Cd Desc']).size()** and sorted in descending order by **Crime Count**. We then visualize the output using **sns.catplot()**, with crime types on the x-axis and regions on the y-axis, and labels are customized using **g.set_axis_labels()**. The outputs are shown below.



Central:

Burglary from Vehicle ranks first in the Central area with 9,231 events, followed by Battery-Simple Assault with 6,796 cases. Other noteworthy crimes are Vandalism

(4,672) and Vehicle Theft (4,782). With 4,148 occurrences, Petty Theft is also commonly seen in Theft Plain. This indicates that crimes involving property, such as theft and burglary, are very common.

77th Street:

Compared to other crime categories, Vehicle Theft is the most common crime on 77th Street, with 8,488 occurrences. Additionally common are Battery-Simple Assault (5,706 instances) and Aggravated Assault (5,911 cases).

Pacific:

Vehicle Theft accounts for 6,312 instances in the Pacific area. Property crimes such as Petty Theft and Vehicle Burglary are also frequent, accounting for 3,848 and 4,777 instances, respectively. Significant violent offenses include Battery-Simple Assault (4,060) and Aggravated Assault (4,416).

Southwest:

With 6,417 occurrences, Vehicle Theft ranks first in the Southwest, much like in other regions. Battery-Simple Assault and Aggravated Assault are also common, with 4,510 and 3,624 incidents, respectively. The most frequent property-related crimes, which exhibit a pattern of theft and burglary, are Vehicle Burglaries and Petty Theft.

Hollywood:

With 4,482 incidents, Battery-Simple Assault is the most common crime in Hollywood. Vehicle Theft and Vehicle Burglary are also common, with 3,854 and 4,199 occurrences, respectively. Vandalism and Petty Theft are among the other most common crimes in this area, indicating a combination of theft-related and property damage offenses.

5. Correlational Analysis:

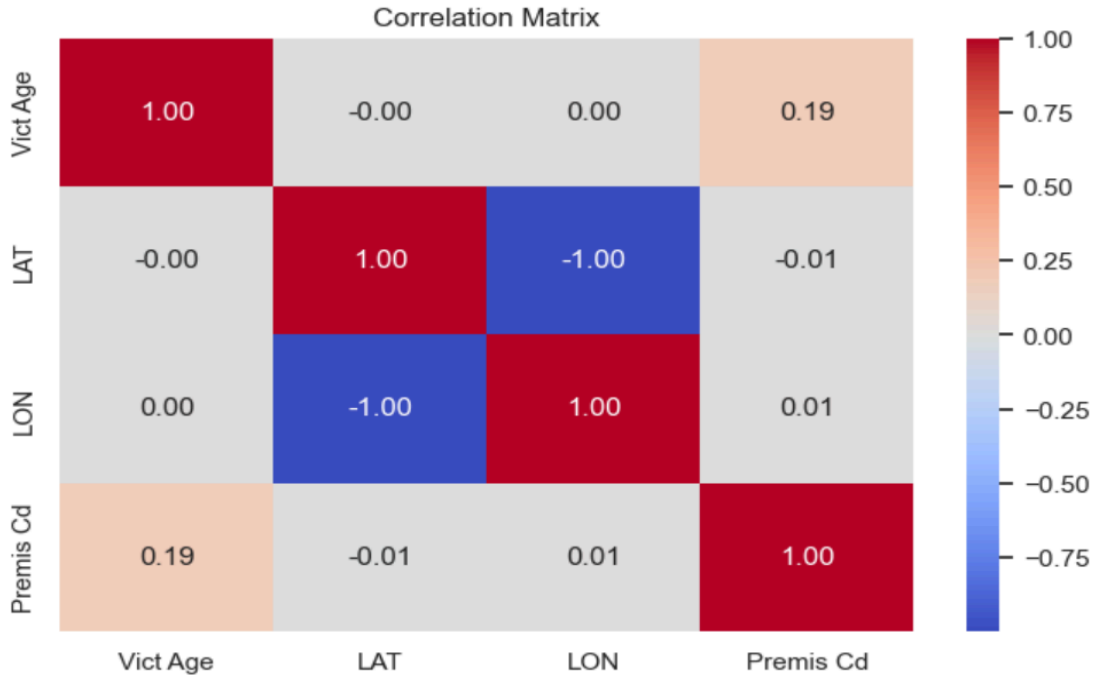
In this step, we look into the connections between our dataset's numerical columns such as victim age and geographical information, by performing a correlation analysis. We will be better able to comprehend the relationships between these variables . For simpler interpretation, we will calculate a correlation matrix and display it as a heatmap.

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

# Assuming your dataset is loaded into a DataFrame called 'crimedata'

# Select numeric columns for correlation analysis
columns_to_correlate = ['Vict Age', 'LAT', 'LON', 'Premis Cd']

# Drop rows with missing data in these columns
crimedata_clean = crimedata[columns_to_correlate].dropna()

# Calculate the correlation matrix for the selected columns
correlation_matrix = crimedata_clean.corr()

# Display the correlation matrix
print(correlation_matrix)

# Optionally, visualize the correlation matrix using a heatmap
plt.figure(figsize=(8, 5))
sns.heatmap(correlation_matrix, annot=True, cmap='coolwarm', fmt='.2f')
plt.title('Correlation Matrix')
plt.show()
```

We first select the numeric columns for correlation using **columns_to_correlate = ['Vict Age', 'LAT', 'LON', 'Premis Cd']**. Then, rows with missing data are dropped with **crimedata_clean = crimedata[columns_to_correlate].dropna()**. The correlation matrix is calculated using **.corr()** and displayed via **print(correlation_matrix)**. To make it visually appealing, we create a heatmap with **sns.heatmap(correlation_matrix, annot=True, cmap='coolwarm', fmt='.2f')**, which shows the strength of relationships between these variables. Finally, **plt.show()** is used to display the heatmap.

Result and Analysis:

	Vict Age	LAT	LON	Premis Cd
Vict Age	1.000000	-0.000486	0.001811	0.185756
LAT	-0.000486	1.000000	-0.998247	-0.005749
LON	0.001811	-0.998247	1.000000	0.007119
Premis Cd	0.185756	-0.005749	0.007119	1.000000



The correlation matrix shows the relationships between the numeric columns: **Vict Age**, **LAT**, **LON**, and **Premis Cd**. The values range from -1 to 1, where 1 indicates a perfect positive correlation, -1 indicates a perfect negative correlation, and 0 indicates no correlation.

LAT and **LON** have a perfect negative correlation (-1.00), which is expected as they represent geographical coordinates.

Vict Age shows a weak positive correlation with **Premis Cd** (0.19), suggesting a slight relationship between the victim's age and the premise code.

Other correlations, such as between **Vict Age** and **LAT/LON**, are close to zero, indicating little to no linear relationship between these variables.

Overall, the matrix reveals that most of the selected variables have **weak correlations**, except for the geographic coordinates.

6. Day of the Week Analysis:

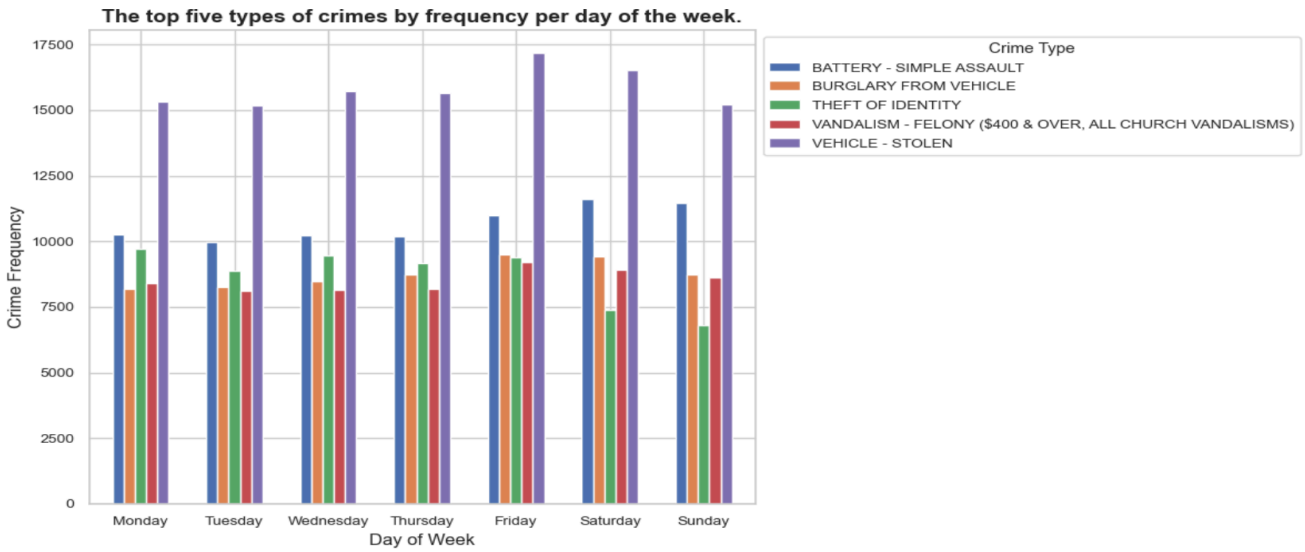
The purpose of this analysis is to determine how crime rates change by day of the week. The five most common forms of crimes are specifically being looked at along with

their weekly distribution. We will use a bar chart to visualize the data and look for any patterns after grouping it by the type of crime and the day of the week.

```
crimedata_filtered['DayOfWeek'] = crimedata_filtered['DATE OCC'].dt.day_name()
top_5_crime_types = crimedata_filtered['Crm Cd Desc'].value_counts().head(5).index
crimedata_top_5 = crimedata_filtered[crimedata_filtered['Crm Cd Desc'].isin(top_5_crime_types)]
crime_day_frequency = crimedata_top_5.groupby(['DayOfWeek', 'Crm Cd Desc']).size().reset_index(name='Count')
crime_day_pivot = crime_day_frequency.pivot(index='DayOfWeek', columns='Crm Cd Desc', values='Count').fillna(0)
days_order = ['Monday', 'Tuesday', 'Wednesday', 'Thursday', 'Friday', 'Saturday', 'Sunday']
crime_day_pivot = crime_day_pivot.reindex(days_order)

plt.figure(figsize=(12, 6))
ax = crime_day_pivot.plot(kind='bar', stacked=False, figsize=(12, 6), width=0.6)
plt.title('The top five types of crimes by frequency per day of the week.', fontsize=14, fontweight='bold')
plt.xlabel('Day of Week', fontsize=12)
plt.ylabel('Crime Frequency', fontsize=12)
plt.xticks(rotation=0, fontsize=10)
plt.yticks(fontsize=10)
plt.legend(title='Crime Type', fontsize=10, title_fontsize='11', loc='upper left', bbox_to_anchor=(1, 1))
plt.tight_layout()
plt.show()
```

We start by extracting the day of the week using **crimedata_filtered['DayOfWeek'] = crimedata_filtered['DATE OCC'].dt.day_name()**. Then, the top 5 crime types are selected using **value_counts().head(5)**, and the data is filtered to include only these crimes with **.isin(top_5_crime_types)**. After grouping by **DayOfWeek** and **Crm Cd Desc**, the data is pivoted using **crime_day_pivot = crime_day_frequency.pivot()**. We then create bar charts with **plot(kind='bar', stacked=False)**.

Result and Analysis:



On all days, **vehicle theft (purple)** is the **most common crime**, with **Friday** and **Saturday** seeing the highest rates. The **second most** frequent crime is **battery**, also known as **simple assault (blue)**, which has rather **consistent weekly rates** with minor surges on **Saturday** and **Sunday**.

Other crimes, such as **theft of identity (green), vandalism (red)**, and **burglary from a vehicle (orange),** show fewer but **steady** occurrences over the course of the day. Most crime types have a **minor increase during the weekend**, indicating that criminal activity may pick up toward the end of the week.

From this we can infer that while **assault-related crimes tend to rise on weekends**, **vehicle-related crimes are a major problem throughout the week.**

7. Impact of Major Events on Crime Rates:

This analysis looks at the impact of significant events, like the **COVID-19 epidemic**, on crime rates. In order to identify any notable changes, we are specifically examining crime data prior to, during, and following three COVID-19 waves. Visualizing crime trends over time and determining whether any patterns arise as a result of these outside occurrences is the aim of this analysis.

```
from scipy.stats import ttest_ind

crimedata['DATE OCC'] = pd.to_datetime(crimedata['DATE OCC'], errors='coerce')

wave_1_start = pd.to_datetime('2020-03-01')
wave_1_end = pd.to_datetime('2020-05-31')
wave_2_start = pd.to_datetime('2020-11-01')
wave_2_end = pd.to_datetime('2021-02-28')
wave_3_start = pd.to_datetime('2021-07-01')
wave_3_end = pd.to_datetime('2021-11-30')

month_count = crimedata.groupby([crimedata['DATE OCC'].dt.year, crimedata['DATE OCC'].dt.month]).size()

plt.figure(figsize=(10, 6))
month_count.plot(kind='line')
xtick_labels = [f"{year}-{month:02}" for year, month in month_count.index]
plt.xticks(range(0, len(xtick_labels), 2), xtick_labels[::2], rotation=45)

wave_dates = [
    (wave_1_start, 'Wave 1 Start', 'brown', (80, 40)),
    (wave_1_end, 'Wave 1 End', 'green', (-20, -30)),
    (wave_2_start, 'Wave 2 Start', 'brown', (80, 40)),
    (wave_2_end, 'Wave 2 End', 'green', (-20, -30)),
    (wave_3_start, 'Wave 3 Start', 'brown', (80, 40)),
    (wave_3_end, 'Wave 3 End', 'green', (-20, -30))
]

for date, label, color, xytext in wave_dates:
    plt.annotate(label, xy=(xtick_labels.index(f"{date.year}-{date.month:02}"), month_count[date.year, date.month]),
                 xytext=xytext, textcoords='offset points', arrowprops=dict(arrowstyle='-|>', lw=0.9, color=color), fontsize=10)
```

```
start_index = xtick_labels.index('2020-01')
end_index = xtick_labels.index('2022-01')
plt.plot([start_index, end_index], [month_count[2020, 1], month_count[2022, 1]], color='red', linestyle='--', linewidth=1.5)

green_line_end_index = end_index + 5
plt.plot([end_index, green_line_end_index], [month_count[2022, 1], month_count.iloc[end_index + 5]], color='green', linestyle='--', linewidth=1.5)
plt.xlabel('Years & Months')
plt.ylabel('No of Crimes')
plt.title('Crime rates before, during, and after COVID-19 Waves 1, 2, and 3')
plt.grid(True, linestyle='--')
plt.tight_layout()
plt.show()

before_wave_1_end = wave_1_start - pd.DateOffset(months=1)
after_wave_3_end = wave_3_end + pd.DateOffset(months=1)

before_wave_1_data = crimedata[crimedata['DATE OCC'] < before_wave_1_end].copy()
after_wave_3_data = crimedata[crimedata['DATE OCC'] > after_wave_3_end].copy()

before_after_month_count = pd.concat([
    before_wave_1_data.groupby([before_wave_1_data['DATE OCC'].dt.year, before_wave_1_data['DATE OCC'].dt.month]).size(),
    after_wave_3_data.groupby([after_wave_3_data['DATE OCC'].dt.year, after_wave_3_data['DATE OCC'].dt.month]).size()
])

plt.figure(figsize=(10, 6))
before_after_month_count.plot(kind='line', color='purple', marker='o')
```

```
plt.figure(figsize=(10, 6))
before_after_month_count.plot(kind='line', color='purple', marker='o')

xtick_labels = [f"{year}-{month:02}" for year, month in before_after_month_count.index]
plt.xticks(range(0, len(xtick_labels), 2), xtick_labels[::2], rotation=45)
plt.xlabel('Years & Months')
plt.ylabel('No of Crimes')
plt.title('Crime rates before COVID-19 Wave 1 and after Wave 3')
plt.grid(True, linestyle='--')
plt.tight_layout()
plt.show()
```

The code begins by defining the dates of **three COVID-19** waves using the **pd.to_datetime() function**. The crime data is grouped by **year and month to calculate the number of crimes per month using .groupby()**. We then create a **line plot to visualize these trends**, using **plt.annotate()** to mark key points like the start and end of each wave.

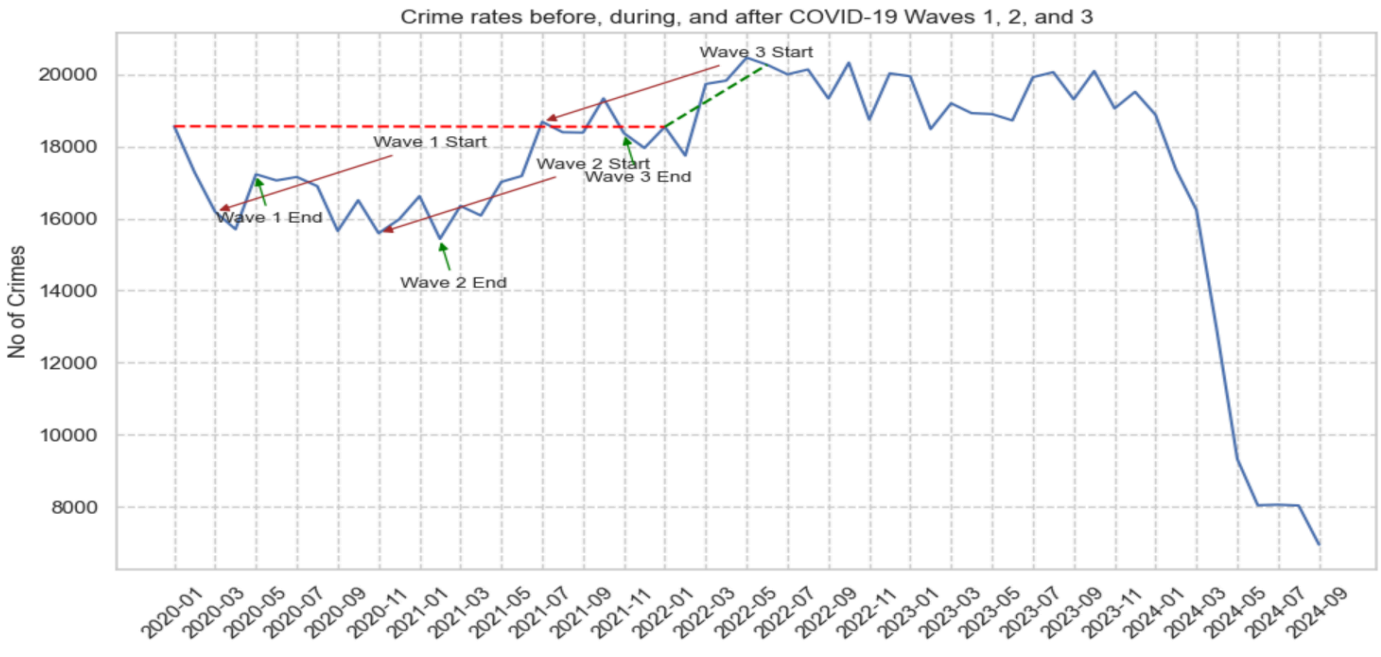
After that, we plot the **crime rates before Wave 1** and **after Wave 3** to compare the periods, showing how crime numbers fluctuate due to the pandemic. Finally, we use another **line plot** to visualize this comparison, which provides insights into how crime rates changed before and after major COVID-19 events.

To be more specific, we have also separately analyzed the most common type of crime which is **Vehicle-Stolen** and how it fluctuates and varies due to COVID-19 and its lockdown restrictions. The code for that is shown below.

```python
vehicle_stolen_data = crimedata_filtered[crimedata_filtered['Crm Cd Desc'] == 'VEHICLE - STOLEN'].copy()
vehicle_stolen_data.loc[:, 'YearMonth'] = vehicle_stolen_data['DATE OCC'].dt.to_period('M')
monthly_trend_vehicle_stolen = vehicle_stolen_data.groupby('YearMonth').size().reset_index(name='Crime Count')

plt.figure(figsize=(12, 6))
plt.plot(monthly_trend_vehicle_stolen['YearMonth'].dt.to_timestamp(), monthly_trend_vehicle_stolen['Crime Count'], marker='o', color='b')
plt.title('Monthly Trend of VEHICLE - STOLEN', fontsize=14)
plt.ylabel('Number of Crimes')
plt.xlabel('Month-Year')
plt.grid(True)

start_red_line = monthly_trend_vehicle_stolen[monthly_trend_vehicle_stolen['YearMonth'] == '2021-01'].index[0]
end_red_line = monthly_trend_vehicle_stolen[monthly_trend_vehicle_stolen['YearMonth'] == '2021-07'].index[0]
plt.plot(monthly_trend_vehicle_stolen['YearMonth'].iloc[start_red_line:end_red_line + 1].dt.to_timestamp(),
         monthly_trend_vehicle_stolen['Crime Count'].iloc[start_red_line:end_red_line + 1],
         color='red', linestyle='-', linewidth=2)

start_green_line = monthly_trend_vehicle_stolen[monthly_trend_vehicle_stolen['YearMonth'] == '2021-07'].index[0]
end_green_line = monthly_trend_vehicle_stolen[monthly_trend_vehicle_stolen['YearMonth'] == '2021-10'].index[0]
plt.plot(monthly_trend_vehicle_stolen['YearMonth'].iloc[start_green_line:end_green_line + 1].dt.to_timestamp(),
         monthly_trend_vehicle_stolen['Crime Count'].iloc[start_green_line:end_green_line + 1],
         color='green', linestyle='-', linewidth=2)

plt.xticks(rotation=45)
plt.tight_layout()
plt.show()
```

From **2020 to 2024**, we examine the **monthly trend** of **auto thefts** (designated as **"VEHICLE-STOLEN"**). First, we use the **.dt.to_period('M')** function to transform the **"DATE OCC"** column to a **month-year period** after filtering the crime data for **auto thefts**. The number of **car thefts** for each month is then determined by applying the **.groupby('YearMonth')** function on the **monthly trend**. To see the overall changes in **car thefts** over time, we use **plt.plot()** to plot this trend. We designate two time segments to draw attention to certain times: **January 2021–July 2021** (shown in **red**) and **July 2021–October 2021** (shown in **green**). To illustrate how **crime changed** during various periods, these portions are **plotted independently**.

Result and Analysis:

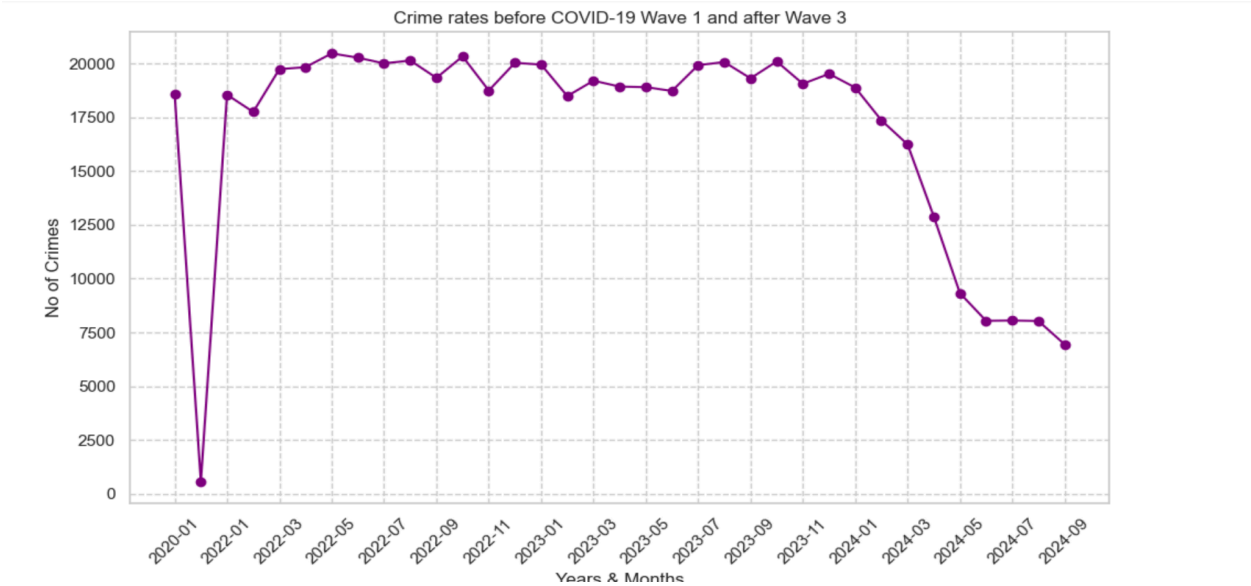


The graph focuses on the time periods **prior to, during, and following COVID-19 Waves 1, 2, and 3** and displays **crime rates from January 2020 to September 2024**.

Due to the **pandemic's start**, **strict lockdowns**, and other **social distancing measures**, crime rates **dramatically decreased** during **Wave 1 (March 2020–May 2020)**. As **limits were loosened** following the **end of Wave 1** in **May 2020**, crime rates started to **gradually increase**.

Crime rates **leveled** during **Wave 2 (November 2020–February 2021)**, with an **apparent decrease** toward the end of the wave. This pattern is consistent with the more **restrictive rules** that were implemented in the **winter**. Crime rates **progressively rose** after **Wave 2 ended** in **February 2021**.

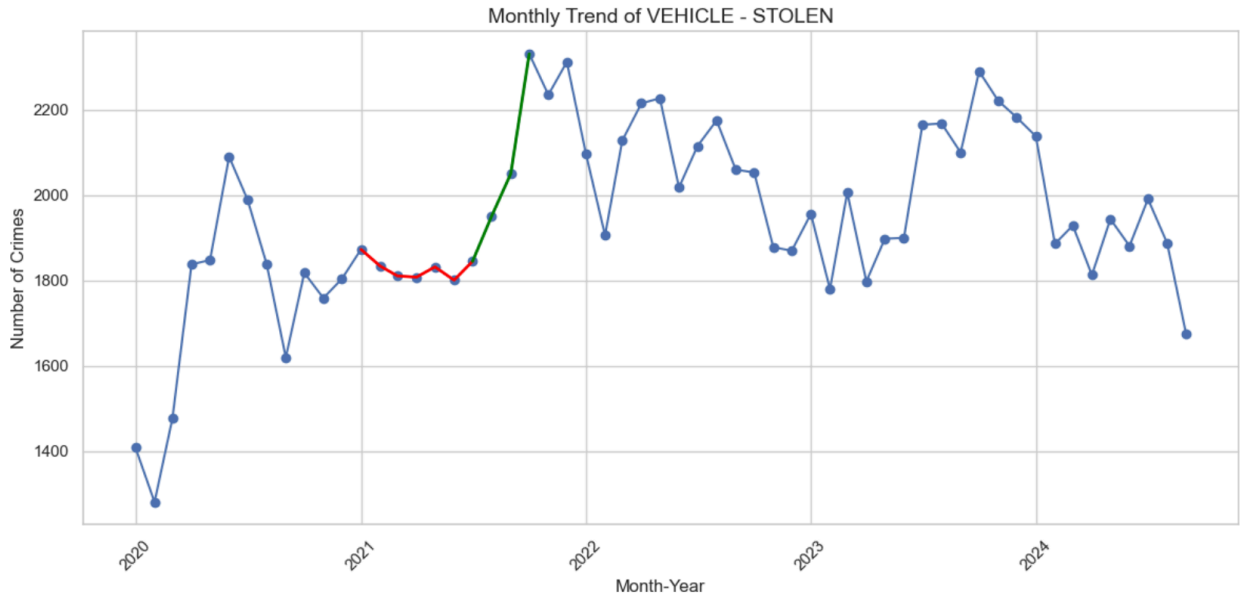
The greatest increase in crime occurred during **Wave 3 (July 2021 to November 2021)**, with rates reaching their **highest point** in **July 2021**. Crime rates stayed **comparatively high** after **Wave 3 concluded** in **November 2021** until **late 2023**, when they began to **decline sharply and steadily** in **early 2024**.



The graph shows crime rates before **COVID-19 Wave 1** and after **Wave 3**. There is a sharp drop in crime rates at the beginning of **Wave 1** in **March 2020**, coinciding with the implementation of strict lockdown measures. Crime rates gradually increased after **Wave 1** and remained relatively stable through **Wave 2** and **Wave 3**. After **Wave 3** (ending in **November 2021**), crime rates peaked but then began a steady decline, with a sharp drop starting from early **2024**.

These variations can be explained by a number of **pandemic-related measures**, including **curfews**, **lockdowns**, and **social distancing campaigns**, which decreased the incidence of criminal activity during the pandemic's heights. The subsequent **increase after the wave** may have been caused by the **relaxation of these regulations** and the **reopening of public areas**, which gave criminals additional chances to commit crimes. The **dramatic decline** in crime rates following **2023**, however, might be the result of **community-based programs** that reduced crime over time, **adjustments to police enforcement tactics**, or **socioeconomic variables**.

Now we'll look into the visual representation of how **"Vehicle-Stolen"** type of crime's rate get affected by COVID-19 lockdown.



The monthly trend of Vehicle-Stolen from 2020 to 2024 is depicted in the graph. A red line from January 2021 to July 2021, where the trend is comparatively stable but slightly falling, indicates an decrease in vehicle theft cases in early 2021. After this time, there is a notable increase, seen by the green line from July 2021 to October 2021.

The loosening of pandemic restrictions may be the cause of this surge, as increasing use of vehicles and public areas created more opportunity for theft. Following that, there is a visible dramatic dip in early 2024, although the pattern continues to fluctuate, showing both gains and decreases.

8. Advance Analysis - Predicting Future Trends(using ARIMA Analysis):

In this analysis, we are performing **future crime rate forecasting** using the **ARIMA (AutoRegressive Integrated Moving Average)** model.

The dataset was **resampled monthly** using **crime_data.resample('M').size()**, which aggregated monthly crime counts, then **arranged chronologically** by occurrence date for this study. The basis for future **ARIMA (AutoRegressive Integrated Moving Average)** crime rate predictions was established by this preparation. Using **monthly crime data**, the **ARIMA model**—which has an order of **(1, 1, 1)**—was trained. We used **results.get_forecast(steps=12)** to create a **12-step forecast** for future crime rates after fitting the model with **model.fit()**. **Plotting** of the actual crime data and predicted values was done using **plt.plot()**, and the **visualization of confidence intervals** was done using **plt.fill_between()**, which produced a shaded region that showed the **range of anticipated future crime counts**. This **prediction method** provides useful information for planning and decision-making by predicting **future crime trends** based on historical patterns.

```
from statsmodels.tsa.arima.model import ARIMA
crime_data = crimedata_filtered.set_index('DATE OCC')
crime_data.sort_index(inplace=True)
crime_monthly = crime_data.resample('M').size()
model = ARIMA(crime_monthly, order=(1, 1, 1))
results = model.fit()
print(results.summary())

forecast_steps = 12
forecast = results.get_forecast(steps=forecast_steps)
forecast_values = forecast.predicted_mean
confidence_intervals = forecast.conf_int()
print("Forecasted Crime Counts:")
print(forecast_values)
print("\nConfidence Intervals:")
print(confidence_intervals)

plt.figure(figsize=(12, 6))
plt.plot(crime_monthly.index, crime_monthly.values, label='Actual', marker='o')
plt.plot(forecast_values.index, forecast_values.values, color='red', label='Forecast', marker='o')
plt.fill_between(confidence_intervals.index,
                 confidence_intervals.iloc[:, 0],
                 confidence_intervals.iloc[:, 1],
                 color='pink', alpha=0.3, label='Confidence Intervals')
plt.title('Crime Forecasting')
plt.xlabel('Date')
plt.ylabel('Crime Counts')
plt.legend()
plt.grid(True)
plt.tight_layout()
plt.show()
```

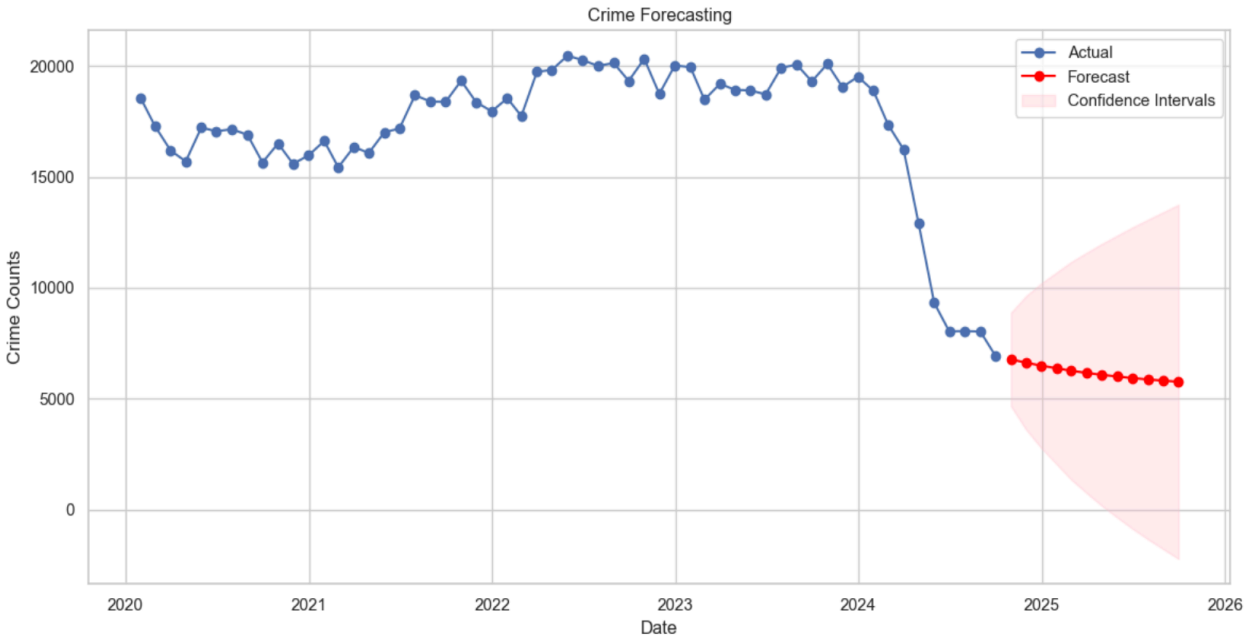
Result and Analysis:

```
                               SARIMAX Results
==============================================================================
Dep. Variable:                      y   No. Observations:                   57
Model:                 ARIMA(1, 1, 1)   Log Likelihood                -470.426
Date:                Tue, 15 Oct 2024   AIC                            946.852
Time:                        22:20:23   BIC                            952.928
Sample:                  01-31-2020   HQIC                           949.208
                       - 09-30-2024
Covariance Type:                  opg
==============================================================================
                 coef    std err          z      P>|z|      [0.025      0.975]
------------------------------------------------------------------------------
ar.L1          0.8954      0.514      1.742      0.082      -0.112       1.903
ma.L1         -0.8712      0.539     -1.615      0.106      -1.929       0.186
sigma2      1.147e+06   2.02e+05      5.671      0.000    7.51e+05    1.54e+06
===================================================================================
Ljung-Box (L1) (Q):                   0.30   Jarque-Bera (JB):                 5.44
Prob(Q):                              0.58   Prob(JB):                         0.07
Heteroskedasticity (H):               2.06   Skew:                            -0.60
Prob(H) (two-sided):                  0.12   Kurtosis:                         3.94
===================================================================================
```

The **ARIMA(1,1,1)** model summary shows a Log Likelihood of -470.426 and values for **AIC**, **BIC**, and **HQIC** (946.852, 952.928, and 949.208, respectively). These metrics help us assess the model's quality, with lower values indicating a better fit. The **coefficients** for **ar.L1** (0.8954) and **ma.L1** (-0.8712) represent the autoregressive and moving average components of the model, which indicate how past values influence future predictions.

```
Warnings:
[1] Covariance matrix calculated using the outer product of gradients (complex-step).
Forecasted Crime Counts:
2024-10-31    6777.454561
2024-11-30    6627.434062
2024-12-31    6493.105428
2025-01-31    6372.827318
2025-02-28    6265.130074
2025-03-31    6168.697761
2025-04-30    6082.352089
2025-05-31    6005.038012
2025-06-30    5935.810844
2025-07-31    5873.824708
2025-08-31    5818.322206
2025-09-30    5768.625163
Freq: ME, Name: predicted_mean, dtype: float64

Confidence Intervals:
                lower y       upper y
2024-10-31  4678.449917   8876.459205
2024-11-30  3622.816637   9632.051487
2024-12-31  2771.882024  10214.328832
2025-01-31  2031.401748  10714.252889
2025-02-28  1364.817408  11165.442740
2025-03-31   753.181902  11584.213621
2025-04-30   185.032446  11979.671731
2025-05-31  -347.285402  12357.361426
2025-06-30  -849.246758  12720.868446
2025-07-31 -1324.970145  13072.619561
2025-08-31 -1777.673890  13414.318302
2025-09-30 -2209.948072  13747.198398
```

The **forecasted crime counts** for the next 12 months, starting from **October 2024**, range from **6,777 in October 2024** to **5,768 in September 2025**, showing a steady decline. The **confidence intervals** provide a range within which these predictions are likely to fall, with a lower bound (for October 2024) of **4,678** and an upper bound of **8,876**. As time progresses, the intervals widen, reflecting increasing uncertainty in the long-term forecast.



A **decreasing trend** in crime is graphically depicted in the **prediction plot**, where the **shaded region** represents the **confidence intervals** and the **red line** represents the **predicted values**. After a **steep reduction** in early **2024**, the decline is still going strong, indicating that **low crime rates** are anticipated. **Confidence intervals** that are getting **wider** show how **uncertain** forecasts are getting further out into the future.

So finally, according to the ARIMA model, crime rates should fall between the confidence levels and exhibit the anticipated patterns for the next year.

9. Demographic Factors of the Crime data:

Our goal in this part is to show the demographic characteristics associated with the crimes in our dataset, with particular attention to age, gender, and the kinds of crimes that were committed. We're carrying out three important visualizations:

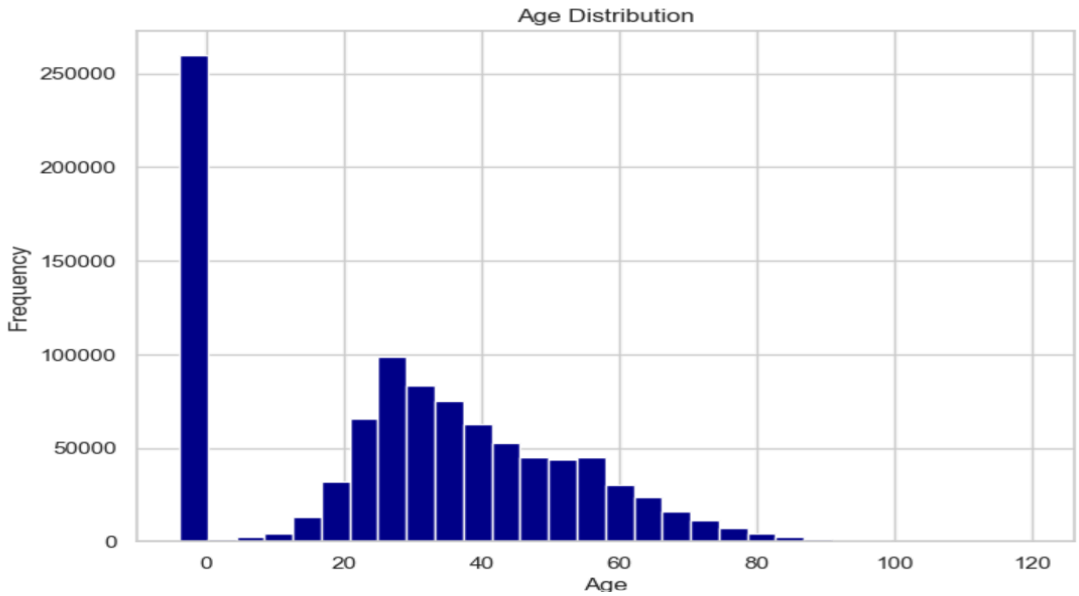
Age Distribution: To show how the victims' ages are distributed, we shall make a histogram. This enables us to determine the age ranges that are most frequently impacted by criminal activity.

Gender Distribution: The victims' gender distribution will be shown using a bar chart. This enables us to determine if the dataset shows any appreciable differences between male and female victims.

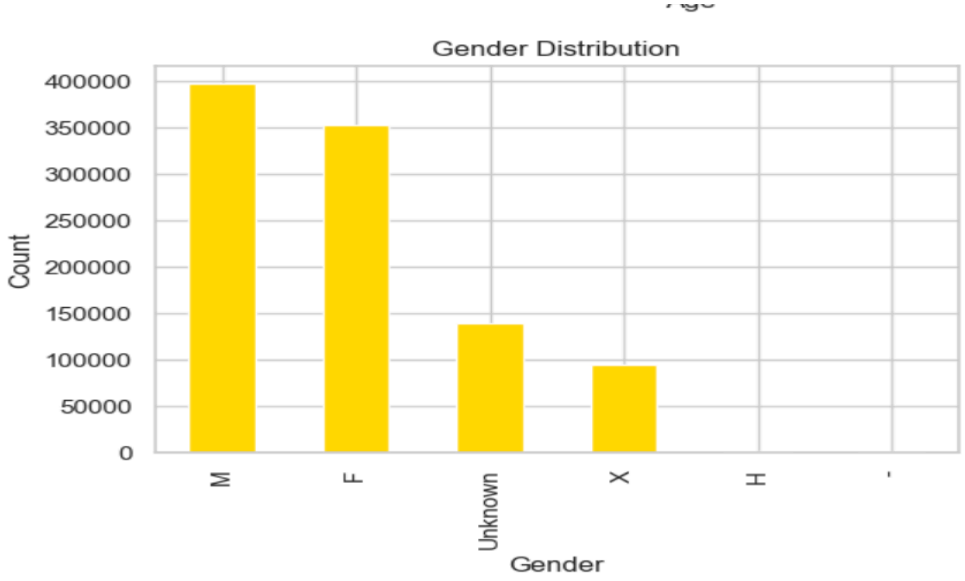
Distribution of Crime Types: Lastly, a pie chart will show the top five most common types of crimes. We can learn which crimes are more common by looking at the percentage of various crime kinds.

```
top_crime_counts = crime_data['Crm Cd Desc'].value_counts().head(5)

plt.figure(figsize=(8, 6))
plt.hist(crime_data['Vict Age'], bins=30, color='darkblue')
plt.title('Age Distribution')
plt.xlabel('Age')
plt.ylabel('Frequency')
plt.show()

plt.figure(figsize=(6, 4))
crime_data['Vict Sex'].value_counts().plot(kind='bar', color='gold')
plt.title('Gender Distribution')
plt.xlabel('Gender')
plt.ylabel('Count')
plt.show()

plt.figure(figsize=(5, 5))
top_crime_counts.plot(kind='pie', autopct='%1.1f%%', colors=['blue', 'yellow', 'orange', 'red', 'purple'])
plt.title('Crime Type Distribution')
plt.ylabel('')
plt.show()
```

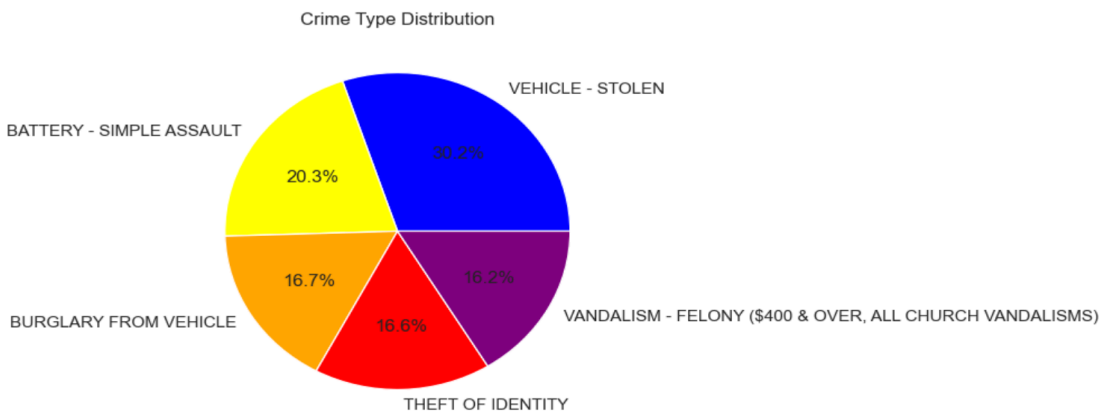
Result and Analysis:



According to the **Age Distribution plot**, a **sizable portion** of crime-related incidents include **people who are 0 years old**, which may indicate **inaccurate data** or **misclassifications**. With the **largest frequency of victims** being between the ages of **20 and 40**, the distribution leans toward **younger age groups** aside from this anomaly. The **lower part** of the distribution extends into **higher age groups**, indicating that the frequency **declines with age** and that **fewer older people** are victims of crimes.



According to the **Gender Distribution plot**, **men (M)** make up the **greatest group of crime victims**, with **women (F)** coming in second. Additionally, a sizable amount of data is marked as **Unknown** or **X**, which could indicate **non-binary gender data** or **unreported data**. Considering the fact that women make up a **significant portion of the sample**, this distribution implies that **crime victimization** is more common among **men**.



According to the **Crime Type Distribution** pie chart, the **most frequent crime type** in the dataset is **vehicle theft (30.2%)**, followed by **battery-simple assault (20.3%)**. **Theft of Identity**, **Vandalism**, and **Vehicle Burglary** each make up about **sixteen percent** of all instances. This demonstrates that **violent and property-related crimes** are both **highly widespread**, with **car theft** being one of the **most common problems**.

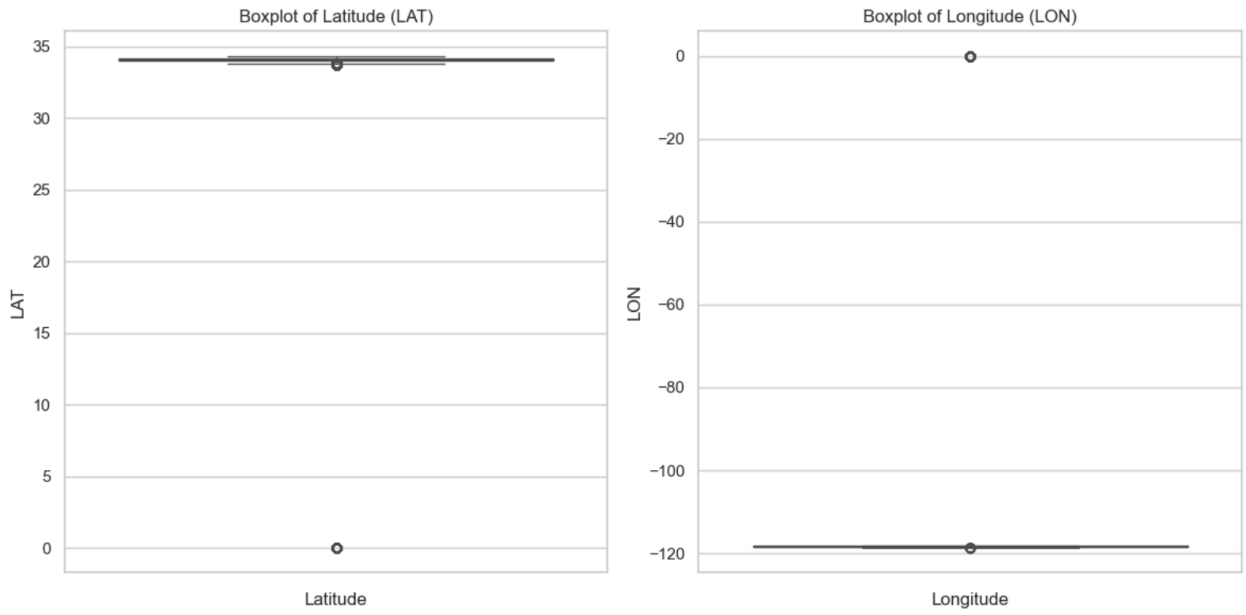
10. Outliers and Anomalies:

Now, we are going to visualize the distribution of **Latitude (LAT)** and **Longitude (LON)** values in our crime dataset using boxplots. This will help us detect any potential outliers or Anomalies in the geographical data and understand the general spread of crime locations.

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

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

# Boxplot for Latitude
plt.subplot(1, 2, 1)
sns.boxplot(y=crimedata['LAT'].dropna(), orient='v')
plt.title('Boxplot of Latitude (LAT)')
plt.xlabel('Latitude')

# Boxplot for Longitude
plt.subplot(1, 2, 2)
sns.boxplot(y=crimedata['LON'].dropna(), orient='v')
plt.title('Boxplot of Longitude (LON)')
plt.xlabel('Longitude')

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

The code creates two boxplots to represent the latitude and longitude values. Using **sns.boxplot()**, it generates separate visualizations for both the LAT and LON columns. The first subplot visualizes the **LAT** values to detect the spread and potential outliers in the dataset. The second subplot does the same for the **LON** values.

Result and Analysis:



The **LAT** box plot displays a closely clustered set of values, roughly between **30 and 35**, suggesting that the majority of the dataset's geographic locations fall within this latitude range. A small number of outliers with extremely low latitude values **could be inaccurate data.**

The values in the **LON** boxplot are primarily grouped around **-120**, which corresponds to the usual longitudinal coordinates. The existence of some extreme numbers, as shown by the outliers, **could still be a reflection of inaccurate data.**

Conclusion

Throughout this academic project, we rigorously engaged in data preprocessing and exploratory analysis with a professional approach. The dataset, centered around crime data, provided us with the opportunity to apply data cleaning techniques and ensure the reliability of the data for potential future use. We meticulously removed certain columns that were not pertinent to our focus, ensuring the dataset remained streamlined and relevant. The handling of missing data was approached with precision; we strategically imputed missing values in some columns and removed rows with null values, thereby maintaining data consistency.

Although our work is preliminary, it has laid a strong foundation for future explorations or analyses that could be conducted on this dataset. By carefully cleaning and preprocessing the data, we ensured it is in an optimal state for further academic exploration. The skills and techniques applied in this project reflect a professional approach to data management and analysis.

In conclusion, this EDA provides a comprehensive understanding of crime dynamics in Los Angeles. The findings have implications for law enforcement strategies, resource allocation, and policy-making efforts aimed at enhancing public safety and reducing criminal activities in the city