CRIME DATA ANALYSIS

IE6400 FOUNDATIONS DATA ANALYTICS ENGINEERING

FALL 2024

Project Report

GROUP No : 9

1) Aditya Bafna - 002320602

2) Mihir Parab - 002324347

3) Harsh Bora - 002320899

Introduction

In today's data-driven world, understanding and addressing complex societal challenges, such as crime, requires a multifaceted approach. **Exploratory Data Analysis (EDA)** serves as a critical tool in this process, offering a structured and insightful way to evaluate and analyze crime data. This report aims to explore a dataset of crime incidents to uncover patterns, trends, and potential underlying causes.

Crime is a multifaceted phenomenon influenced by various **socioeconomic, demographic, and environmental factors**. By thoroughly examining the available data, we seek to shed light on the dynamics of criminal activity and potentially uncover insights that can inform evidence-based strategies for **law enforcement, public policy**, and **community engagement**.

Through the application of rigorous statistical techniques, data visualization tools, and domain knowledge, this analysis strives to extract actionable insights from the crime data. It is crucial to approach the analysis with an open mind, acknowledging the complexity and the diverse factors influencing crime.

Ultimately, this report aims to provide stakeholders with a deeper understanding of crime patterns, enabling **evidence-based decision-making** and fostering **proactive strategies** to enhance public safety and community well-being. By delving into the nuances of crime data, we hope to contribute to ongoing efforts to create safer and more secure environments for all.

Data Preprocessing and Analysis

1. Data Acquisition:

In the data acquisition phase, we used **Python** and its powerful data manipulation library, **pandas**, to access and load the dataset for our analysis. The dataset, titled **'Crime_Data_from_2020_to_Present.csv'**, was sourced from the provided link [Crime Data from 2020 to Present]. This CSV file contains detailed crime data ranging from the year 2020 to the present.

By utilizing the `pandas` library, we imported and structured the dataset for easy manipulation and further analysis. Specifically, we used the `pd.read_csv()` function to read the CSV file, which created a **DataFrame** — a versatile tabular data structure often used in data analysis.

The resulting DataFrame, named **`crime_data`**, forms the basis for our exploratory data analysis (EDA). Through this structured data, we will uncover trends, patterns, and insights that will help us better understand the nature of crime during this period.

```
#Task: 1 Data Acquisition: Download the dataset from the provided link
and load it into your preferred data analysis tool

#Importing required libraries for the Project
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import warnings
%matplotlib inline

#loading the dataset
df=pd.read_csv('/Users/apple/DAE/FDA/Project/Crime_Data_from_2020_to_P
resent.csv')
```

2.Data Inspection::

A critical initial step in any data analysis project is to thoroughly inspect the dataset to understand its structure, content, and any potential issues. The following aspects are fundamental in this process:

Displaying the First Few Rows: One of the simplest and most effective ways to begin inspecting the dataset is by displaying the first few rows. This allows for a quick visual assessment of the data’s structure, format, and contents. Using the pandas library in Python, the head() function enables us to easily view the first five rows (by default) of the dataset.

```
df.head(10)

       DR_NO               Date Rptd                DATE OCC  TIME OCC
AREA  \
0  190326475  03/01/2020 12:00:00 AM  03/01/2020 12:00:00 AM      2130
7
```

```
1  200106753  02/09/2020 12:00:00 AM  02/08/2020 12:00:00 AM      1800
1
2  200320258  11/11/2020 12:00:00 AM  11/04/2020 12:00:00 AM      1700
3
3  200907217  05/10/2023 12:00:00 AM  03/10/2020 12:00:00 AM      2037
9
4  220614831  08/18/2022 12:00:00 AM  08/17/2020 12:00:00 AM      1200
6
5  231808869  04/04/2023 12:00:00 AM  12/01/2020 12:00:00 AM      2300
18
6  230110144  04/04/2023 12:00:00 AM  07/03/2020 12:00:00 AM       900
1
7  220314085  07/22/2022 12:00:00 AM  05/12/2020 12:00:00 AM      1110
3
8  231309864  04/28/2023 12:00:00 AM  12/09/2020 12:00:00 AM      1400
13
9  211904005  12/31/2020 12:00:00 AM  12/31/2020 12:00:00 AM      1220
19

   AREA NAME  Rpt Dist No  Part 1-2  Crm Cd  \
0   Wilshire          784         1     510
1    Central          182         1     330
2  Southwest          356         1     480
3   Van Nuys          964         1     343
4  Hollywood          666         2     354
5  Southeast         1826         2     354
6    Central          182         2     354
7  Southwest          303         2     354
8     Newton         1375         2     354
9    Mission         1974         2     624

                                Crm Cd Desc  ... Status   Status Desc
\
0                          VEHICLE - STOLEN  ...     AA  Adult Arrest

1                     BURGLARY FROM VEHICLE  ...     IC   Invest Cont

2                             BIKE - STOLEN  ...     IC   Invest Cont

3  SHOPLIFTING-GRAND THEFT ($950.01 & OVER)  ...     IC   Invest Cont

4                         THEFT OF IDENTITY  ...     IC   Invest Cont

5                         THEFT OF IDENTITY  ...     IC   Invest Cont

6                         THEFT OF IDENTITY  ...     IC   Invest Cont

7                         THEFT OF IDENTITY  ...     IC   Invest Cont

8                         THEFT OF IDENTITY  ...     IC   Invest Cont
```

```
9                  BATTERY - SIMPLE ASSAULT  ...     IC   Invest Cont


  Crm Cd 1 Crm Cd 2  Crm Cd 3 Crm Cd 4  \
0    510.0    998.0       NaN      NaN
1    330.0    998.0       NaN      NaN
2    480.0      NaN       NaN      NaN
3    343.0      NaN       NaN      NaN
4    354.0      NaN       NaN      NaN
5    354.0      NaN       NaN      NaN
6    354.0      NaN       NaN      NaN
7    354.0      NaN       NaN      NaN
8    354.0      NaN       NaN      NaN
9    624.0      NaN       NaN      NaN

                                   LOCATION Cross Street      LAT
LON
0   1900 S  LONGWOOD                     AV          NaN  34.0375 -
118.3506
1   1000 S  FLOWER                       ST          NaN  34.0444 -
118.2628
2   1400 W  37TH                         ST          NaN  34.0210 -
118.3002
3  14000    RIVERSIDE                    DR          NaN  34.1576 -
118.4387
4                           1900    TRANSIENT          NaN  34.0944 -
118.3277
5   9900    COMPTON                      AV          NaN  33.9467 -
118.2463
6   1100 S  GRAND                        AV          NaN  34.0415 -
118.2620
7   2500 S  SYCAMORE                     AV          NaN  34.0335 -
118.3537
8   1300 E  57TH                         ST          NaN  33.9911 -
118.2521
9   9000    CEDROS                       AV          NaN  34.2336 -
118.4535

[10 rows x 28 columns]
```

Reviewing Column Names and Descriptions:

"info" section is a valuable inclusion in a report because it provides a concise yet comprehensive summary of the dataset, offering insights into data types, missing values, memory usage, and data quality. It aids in efficient data exploration and supports informed decision-making during data analysis and preparation.

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 982638 entries, 0 to 982637
Data columns (total 28 columns):
 #   Column          Non-Null Count   Dtype
---  ------          --------------   -----
 0   DR_NO           982638 non-null  int64
 1   Date Rptd       982638 non-null  object
 2   DATE OCC        982638 non-null  object
 3   TIME OCC        982638 non-null  int64
 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        844193 non-null  object
 13  Vict Descent    844182 non-null  object
 14  Premis Cd       982624 non-null  float64
 15  Premis Desc     982053 non-null  object
 16  Weapon Used Cd  326167 non-null  float64
 17  Weapon Desc     326167 non-null  object
 18  Status          982637 non-null  object
 19  Status Desc     982638 non-null  object
 20  Crm Cd 1        982627 non-null  float64
 21  Crm Cd 2        68875 non-null   float64
 22  Crm Cd 3        2311 non-null    float64
 23  Crm Cd 4        64 non-null      float64
 24  LOCATION        982638 non-null  object
 25  Cross Street    151849 non-null  object
 26  LAT             982638 non-null  float64
 27  LON             982638 non-null  float64
dtypes: float64(8), int64(7), object(13)
memory usage: 209.9+ MB
```

Checking Data Types:

Understanding the data types of each column is crucial. Data types determine how the data is stored and can affect subsequent data manipulation and analysis. Pandas offers a way to check the data types of each column in the dataset:

```
df.dtypes

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

3. Data Cleaning

In the phase of identifying and handling null values, we have systematically detected and quantified missing data within our dataset. The code snippet ‘crime_data.isnull().sum()’ executed on the DataFrame 'crime_data' checks for null (or missing) values across all columns. The function isnull() identifies cells in the dataset that do not have valid values, marking them as 'True', and 'False' otherwise. The subsequent sum() function then tallies these 'True' values, effectively counting the number of null entries for each column. By executing this code, we gained valuable insights into the extent and distribution of missing data within our dataset. These findings will guide us in making informed decisions on how to address these gaps during the data cleaning and preparation phase.

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

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

```
Mocodes            145262
Vict Age                0
Vict Sex           138445
Vict Descent       138456
Premis Cd              14
Premis Desc           585
Weapon Used Cd     656471
Weapon Desc        656471
Status                  1
Status Desc             0
Crm Cd 1               11
Crm Cd 2           913763
Crm Cd 3           980327
Crm Cd 4           982574
LOCATION                0
Cross Street       830789
LAT                     0
LON                     0
dtype: int64
```

Identifying columns with object data types that have missing values. Filling missing values in object-type columns using the mode (most frequent value). This ensures that categorical data remains consistent and meaningful.

```
df['Mocodes'] = df['Mocodes'].fillna(df['Mocodes'].mode().iloc[0])
df['Vict Sex'] = df['Vict Sex'].fillna(df['Vict Sex'].mode().iloc[0])
df['Vict Descent'] = df['Vict Descent'].fillna(df['Vict 
Descent'].mode().iloc[0])
df['Weapon Desc'] = df['Weapon Desc'].fillna(df['Weapon 
Desc'].mode().iloc[0])
df['Status']=df['Status'].fillna(df['Status'].mode().iloc[0])
df['Cross Street']=df['Cross Street'].fillna(df['Cross 
Street'].mode().iloc[0])
df['Premis Desc'] = df['Premis Desc'].fillna(df['Premis 
Desc'].mode().iloc[0])
```

Identifying columns with numerical data types that have missing values. Filling missing values in numerical columns using the median (middle value). This helps maintain central tendency without being affected by outliers.

```
df['Premis Cd']=df['Premis Cd'].fillna(df['Premis Cd'].median())
df['Weapon Used Cd']=df['Weapon Used Cd'].fillna(df['Weapon Used 
Cd'].median())
df['Crm Cd 1']=df['Crm Cd 1'].fillna(df['Crm Cd 1'].median())

df.isnull().sum()

DR_NO                   0
Date Rptd               0
```

```
DATE OCC                 0
TIME OCC                 0
AREA                     0
AREA NAME                0
Rpt Dist No              0
Part 1-2                 0
Crm Cd                   0
Crm Cd Desc              0
Mocodes                  0
Vict Age                 0
Vict Sex                 0
Vict Descent             0
Premis Cd                0
Premis Desc              0
Weapon Used Cd           0
Weapon Desc              0
Status                   0
Status Desc              0
Crm Cd 1                 0
Crm Cd 2            913763
Crm Cd 3            980327
Crm Cd 4            982574
LOCATION                 0
Cross Street             0
LAT                      0
LON                      0
dtype: int64
```

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

Duplicate Row Removal:

In the data analysis process, it is vital to ensure that the dataset contains unique and non-repetitive information. Duplicate rows in a dataset can skew analysis results and potentially lead to inaccurate insights. To address this concern, we applied the following data cleaning step: The code "df.drop_duplicates()" was executed to eliminate duplicate rows from the dataset. Duplicate rows are rows that have identical values across all fields. The resulting dataset contains only unique records, ensuring that each row represents distinct information, enhancing data accuracy and analytical validity. The "head()" function was used to display the first few rows of the cleaned dataset, offering a glimpse of the data's structure and content after removing duplicates.

```
df.drop_duplicates().head(10)
```

```
       DR_NO                Date Rptd                DATE OCC  TIME OCC 
AREA  \
0  190326475  03/01/2020 12:00:00 AM  03/01/2020 12:00:00 AM      2130 
7
```

```
1  200106753  02/09/2020 12:00:00 AM  02/08/2020 12:00:00 AM      1800
1
2  200320258  11/11/2020 12:00:00 AM  11/04/2020 12:00:00 AM      1700
3
3  200907217  05/10/2023 12:00:00 AM  03/10/2020 12:00:00 AM      2037
9
4  220614831  08/18/2022 12:00:00 AM  08/17/2020 12:00:00 AM      1200
6
5  231808869  04/04/2023 12:00:00 AM  12/01/2020 12:00:00 AM      2300
18
6  230110144  04/04/2023 12:00:00 AM  07/03/2020 12:00:00 AM       900
1
7  220314085  07/22/2022 12:00:00 AM  05/12/2020 12:00:00 AM      1110
3
8  231309864  04/28/2023 12:00:00 AM  12/09/2020 12:00:00 AM      1400
13
9  211904005  12/31/2020 12:00:00 AM  12/31/2020 12:00:00 AM      1220
19

   AREA NAME  Rpt Dist No  Part 1-2  Crm Cd  \
0   Wilshire          784         1     510
1    Central          182         1     330
2  Southwest          356         1     480
3   Van Nuys          964         1     343
4  Hollywood          666         2     354
5  Southeast         1826         2     354
6    Central          182         2     354
7  Southwest          303         2     354
8     Newton         1375         2     354
9    Mission         1974         2     624

                                 Crm Cd Desc  ... Premis Cd  \
0                           VEHICLE - STOLEN  ...     101.0
1                      BURGLARY FROM VEHICLE  ...     128.0
2                              BIKE - STOLEN  ...     502.0
3  SHOPLIFTING-GRAND THEFT ($950.01 & OVER)  ...     405.0
4                          THEFT OF IDENTITY  ...     102.0
5                          THEFT OF IDENTITY  ...     501.0
6                          THEFT OF IDENTITY  ...     502.0
7                          THEFT OF IDENTITY  ...     248.0
8                          THEFT OF IDENTITY  ...     750.0
9                   BATTERY - SIMPLE ASSAULT  ...     502.0

                                   Premis Desc Weapon Used Cd  \
0                                       STREET          400.0
1             BUS STOP/LAYOVER (ALSO QUERY 124)          400.0
2  MULTI-UNIT DWELLING (APARTMENT, DUPLEX, ETC)          400.0
3                               CLOTHING STORE          400.0
4                                     SIDEWALK          400.0
5                       SINGLE FAMILY DWELLING          400.0
```

```
6  MULTI-UNIT DWELLING (APARTMENT, DUPLEX, ETC)          400.0
7                             CELL PHONE STORE          400.0
8                                   CYBERSPACE          400.0
9  MULTI-UNIT DWELLING (APARTMENT, DUPLEX, ETC)          400.0

                                      Weapon Desc  Status   Status
Desc  \
0  STRONG-ARM (HANDS, FIST, FEET OR BODILY FORCE)      AA  Adult
Arrest
1  STRONG-ARM (HANDS, FIST, FEET OR BODILY FORCE)      IC   Invest
Cont
2  STRONG-ARM (HANDS, FIST, FEET OR BODILY FORCE)      IC   Invest
Cont
3  STRONG-ARM (HANDS, FIST, FEET OR BODILY FORCE)      IC   Invest
Cont
4  STRONG-ARM (HANDS, FIST, FEET OR BODILY FORCE)      IC   Invest
Cont
5  STRONG-ARM (HANDS, FIST, FEET OR BODILY FORCE)      IC   Invest
Cont
6  STRONG-ARM (HANDS, FIST, FEET OR BODILY FORCE)      IC   Invest
Cont
7  STRONG-ARM (HANDS, FIST, FEET OR BODILY FORCE)      IC   Invest
Cont
8  STRONG-ARM (HANDS, FIST, FEET OR BODILY FORCE)      IC   Invest
Cont
9  STRONG-ARM (HANDS, FIST, FEET OR BODILY FORCE)      IC   Invest
Cont

                                   LOCATION Cross Street      LAT
LON
0   1900 S  LONGWOOD                     AV     BROADWAY  34.0375 -
118.3506
1   1000 S  FLOWER                       ST     BROADWAY  34.0444 -
118.2628
2   1400 W  37TH                         ST     BROADWAY  34.0210 -
118.3002
3  14000    RIVERSIDE                    DR     BROADWAY  34.1576 -
118.4387
4                          1900    TRANSIENT     BROADWAY  34.0944 -
118.3277
5   9900    COMPTON                      AV     BROADWAY  33.9467 -
118.2463
6   1100 S  GRAND                        AV     BROADWAY  34.0415 -
118.2620
7   2500 S  SYCAMORE                     AV     BROADWAY  34.0335 -
118.3537
8   1300 E  57TH                         ST     BROADWAY  33.9911 -
118.2521
9   9000    CEDROS                       AV     BROADWAY  34.2336 -
```

```
118.4535

[10 rows x 24 columns]
```

Data Type Conversion for Date and Time Fields:

Converting Date Fields:

The "Date Rptd" and "DATE OCC" fields in the dataset represent dates related to reported incidents and occurrence dates, respectively. To ensure data accuracy and enable chronological analysis, we used "pd.to_datetime()" to convert these fields from object data types to date data types. This conversion guarantees that dates are correctly recognized and processed, facilitating meaningful temporal analysis.

Converting Time Fields:

The "TIME OCC" field in the dataset represents the time of crime incidents. To standardize the time format and support precise time-related analysis, we performed a process to convert the time values. We transformed the "TIME OCC" field into a consistent time format ("HH:MM") by padding and reformatting the values. This transformation ensures uniform time values suitable for time-based analyses.

```
from datetime import datetime as datetime

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

time_occ_list = df['TIME OCC'].apply(lambda x: str(x).zfill(4))
time_list = []


for time_item in time_occ_list:
     time_list.append( (datetime.strptime(str(time_item),"%H
%M").time()).strftime("%H:%M"))

df['TIME OCC'] = time_list

df[['Date Rptd', 'DATE OCC' , 'TIME OCC']]
```

```
        Date Rptd   DATE OCC TIME OCC
0      2020-03-01 2020-03-01    21:30
1      2020-02-09 2020-02-08    18:00
2      2020-11-11 2020-11-04    17:00
3      2023-05-10 2020-03-10    20:37
4      2022-08-18 2020-08-17    12:00
...           ...        ...      ...
982633 2024-08-20 2024-08-17    23:00
982634 2024-07-24 2024-07-23    14:00
982635 2024-01-15 2024-01-15    01:00
982636 2024-04-24 2024-04-24    15:00
```

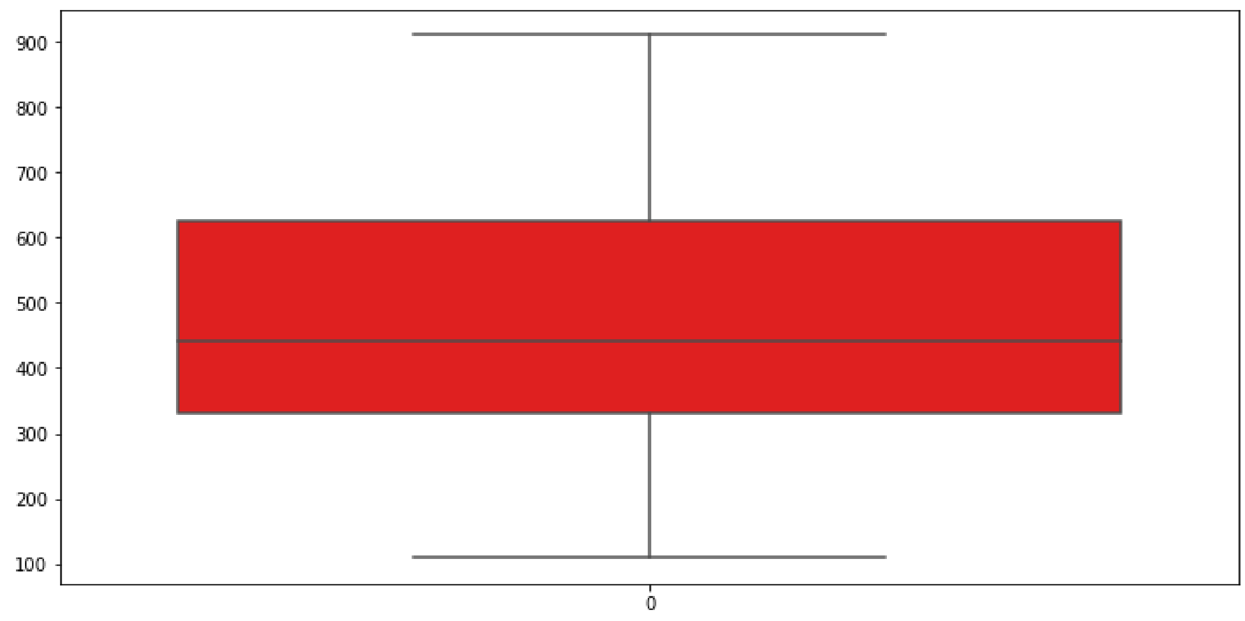
```
982637 2024-08-13 2024-08-12    23:00

[982638 rows x 3 columns]

df.head()

       DR_NO  Date Rptd   DATE OCC TIME OCC  AREA  AREA NAME  Rpt Dist
No  \
0  190326475 2020-03-01 2020-03-01    21:30     7   Wilshire
784
1  200106753 2020-02-09 2020-02-08    18:00     1    Central
182
2  200320258 2020-11-11 2020-11-04    17:00     3  Southwest
356
3  200907217 2023-05-10 2020-03-10    20:37     9   Van Nuys
964
4  220614831 2022-08-18 2020-08-17    12:00     6  Hollywood
666

   Part 1-2  Crm Cd                               Crm Cd Desc  ...
Premis Cd  \
0         1     510                          VEHICLE - STOLEN  ...
101.0
1         1     330                     BURGLARY FROM VEHICLE  ...
128.0
2         1     480                             BIKE - STOLEN  ...
502.0
3         1     343  SHOPLIFTING-GRAND THEFT ($950.01 & OVER)  ...
405.0
4         2     354                         THEFT OF IDENTITY  ...
102.0

                                    Premis Desc Weapon Used Cd  \
0                                        STREET          400.0
1             BUS STOP/LAYOVER (ALSO QUERY 124)          400.0
2  MULTI-UNIT DWELLING (APARTMENT, DUPLEX, ETC)          400.0
3                                CLOTHING STORE          400.0
4                                      SIDEWALK          400.0

                                      Weapon Desc  Status   Status
Desc  \
0  STRONG-ARM (HANDS, FIST, FEET OR BODILY FORCE)      AA  Adult
Arrest
1  STRONG-ARM (HANDS, FIST, FEET OR BODILY FORCE)      IC   Invest
Cont
2  STRONG-ARM (HANDS, FIST, FEET OR BODILY FORCE)      IC   Invest
Cont
3  STRONG-ARM (HANDS, FIST, FEET OR BODILY FORCE)      IC   Invest
Cont
4  STRONG-ARM (HANDS, FIST, FEET OR BODILY FORCE)      IC   Invest
```

```
Cont

                                   LOCATION Cross Street      LAT
LON
0   1900 S  LONGWOOD                    AV      BROADWAY  34.0375 -
118.3506
1   1000 S  FLOWER                      ST      BROADWAY  34.0444 -
118.2628
2   1400 W  37TH                        ST      BROADWAY  34.0210 -
118.3002
3  14000    RIVERSIDE                   DR      BROADWAY  34.1576 -
118.4387
4                             1900     TRANSIENT      BROADWAY  34.0944 -
118.3277

[5 rows x 24 columns]
```

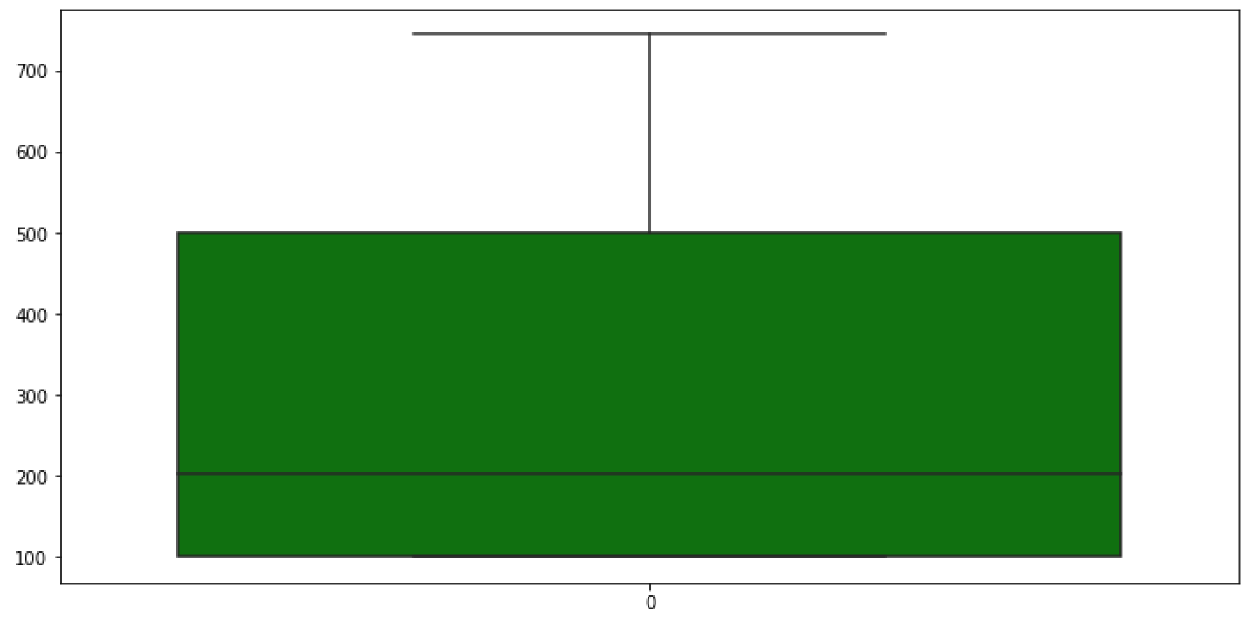
Outlier Detection and Imputation using Z-Score Method:

The Z-score method is used to identify outliers in a field. Outliers represent data points significantly deviating from the typical range of values. These outliers were then replaced with the median value of the column. This imputation approach is robust to extreme values and helps maintain the integrity of the dataset. To visually confirm the effectiveness of this treatment, we created a boxplot, which provides a graphical representation of the central tendency and spread of the values post-outlier imputation. By implementing this approach, we aimed to ensure that extreme values in the field do not unduly influence subsequent analyses. This process serves to enhance the reliability and accuracy of our data for further exploration and interpretation.

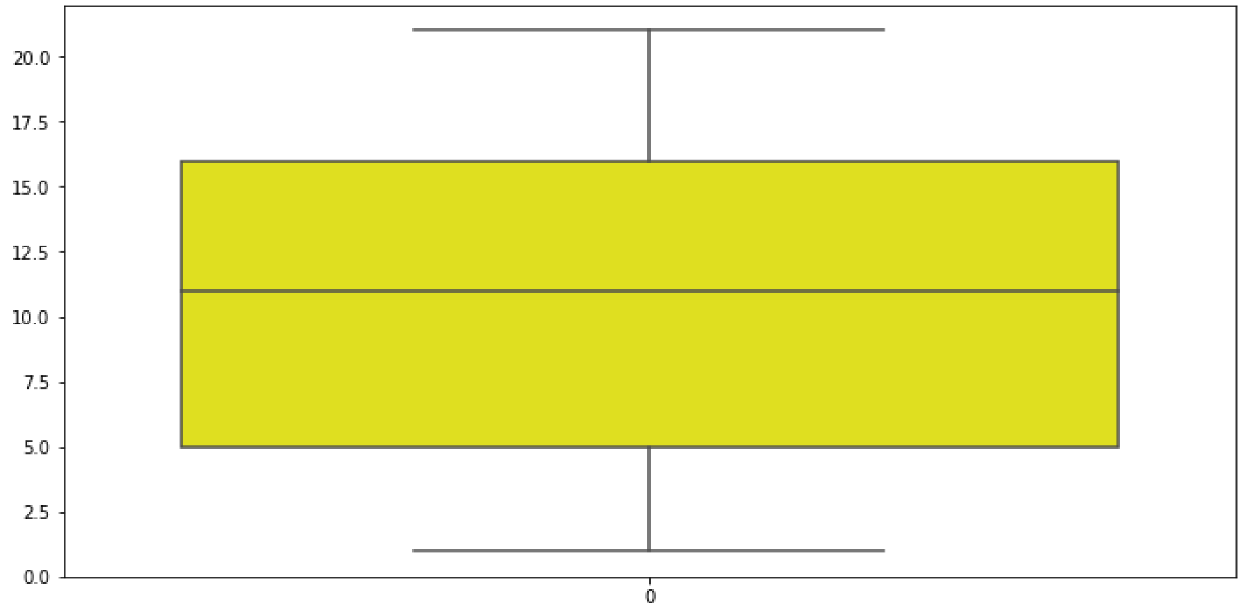
```python
from os import stat
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import scipy.stats as stats
from scipy.stats import zscore


z_scores = zscore(df['Crm Cd'])
outliers = (np.abs(z_scores) > 2)


df['Crm Cd'] = np.where(outliers, df['Crm Cd'].median(), df['Crm Cd'])



#bxplot
plt.figure(figsize =(12,6))
sns.boxplot(data = df['Crm Cd'], color = 'red')
plt.show()
```



```
z_scores=zscore(df['Premis Cd'])
outliers=(np.abs(z_scores)>2)
df['Premis Cd'] = np.where(outliers, df['Premis Cd'].median(),
df['Premis Cd'])

#bxplot
plt.figure(figsize =(12,6))
sns.boxplot(data = df['Premis Cd'], color = 'green')
plt.show()
```



```
#df['Weapon Used Cd'] = df['Weapon Used Cd'].astype(float)
#z_scores=zscore(df['Weapon Used Cd'])
#outliers=(np.abs(z_scores)>2)
#df['Weapon Used Cd'] = np.where(outliers,df['Weapon Used
Cd'].median(),df['Weapon Used Cd'])



#bxplot
plt.figure(figsize =(12,6))
sns.boxplot(data = df['AREA'], color = 'yellow')
plt.show()
```



4. Exploratory Data Analysis (EDA):

Yearly Crime Analysis:

In this section, we focused on analyzing the dataset's temporal dimension by grouping the data based on the reporting year. This approach allowed us to gain insights into the trends and variations in crime incidents over time. The key steps and results are as follows:

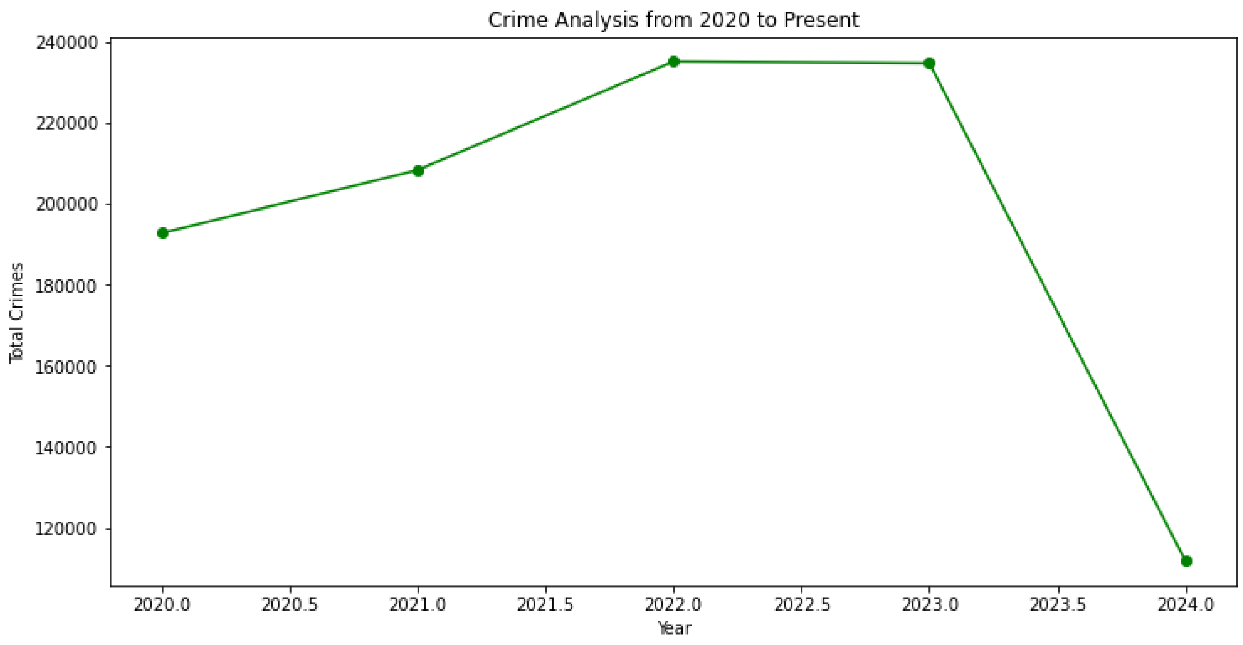
Data Grouping:

We employed the "groupby()" function to group the data based on the "Date Rptd" field, which represents the date when crimes were reported. By using "crime_data['Date Rptd'].dt.year," we extracted the reporting year from each date, enabling us to organize the data chronologically by year. We then applied the "count()" function to calculate the number of crimes reported in each year.

Time Series Plot:

To visualize the temporal trend, we created a time series plot that displays the total number of crimes reported from 2020 to the present. The plot presents a clear overview of how crime incidents have evolved over the years. The x-axis represents the years, the y-axis shows the total number of crimes reported, and each point on the line plot represents a specific year's crime count.

```
yearly_analysis = df['Date Rptd'].groupby(df['Date
Rptd'].dt.year).count()


plt.figure(figsize =(12,6))
yearly_analysis.plot(kind = 'line', marker ='o',color ='green')
plt.title('Crime Analysis from 2020 to Present')
plt.xlabel('Year')
plt.ylabel('Total Crimes')
plt.show()
```



Analysis:

We will be able to see how crime rates have evolved over time by looking at the resultant time series plot, which will show the general crime patterns from 2020 to the present. The plot shows a significant increase in crime rates in 2022 to 2023.

Monthly Crime Analysis:

In this section, we focused on understanding the dataset's temporal patterns at a monthly level by extracting and analyzing the reporting months. The following steps and results were obtained:

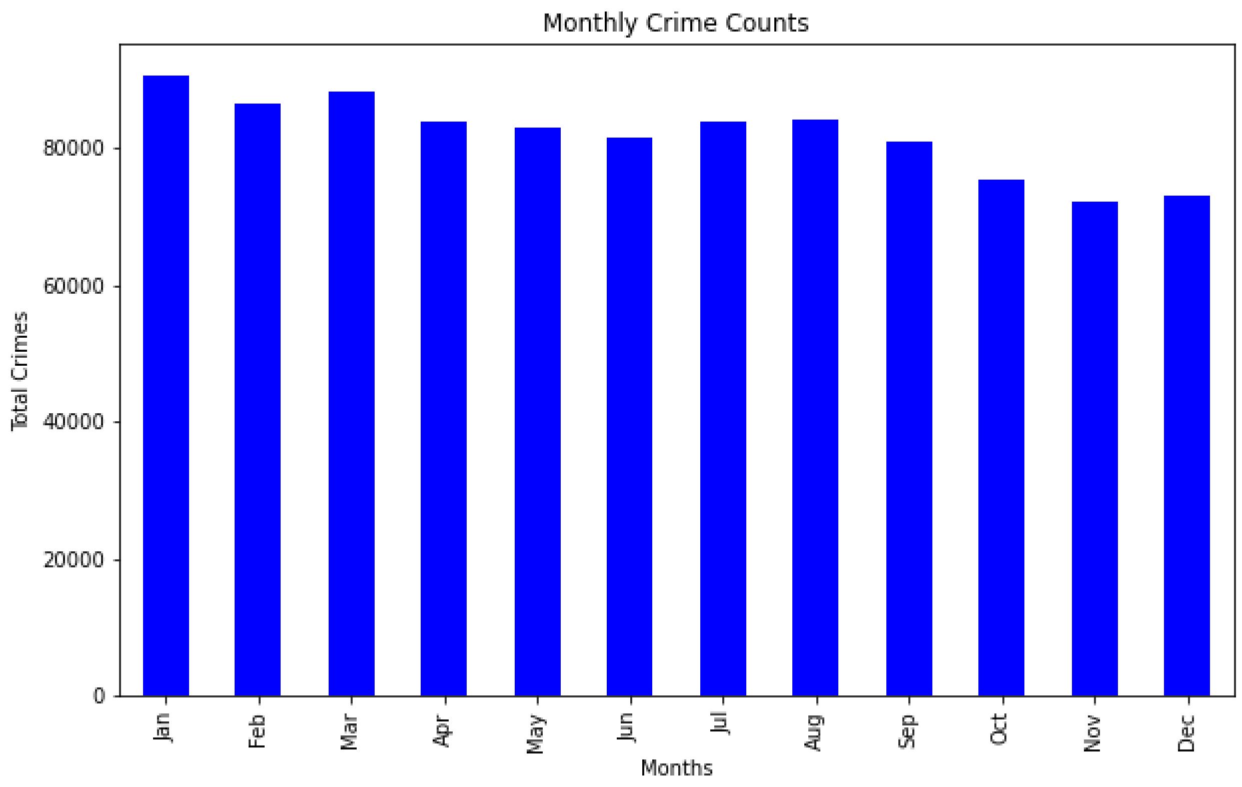
Month Extraction:

We extracted the month from the "Date Rptd" column using the "dt.month" property. This allowed us to isolate the reporting months, providing insights into monthly variations in crime reports. Monthly Crime Counts: After extracting the reporting months, we counted the number of crimes reported for each month. The resulting counts were sorted in chronological order for clear representation.

Visualization:

To visualize the monthly crime counts, we created a bar plot. Each bar in the plot represents a month, and its height corresponds to the total number of crimes reported in that month. The x-axis displays the months (January to December), while the y-axis indicates the total crime counts.

```
#extracting month from "Date Rptd"
df['Month'] = df['Date Rptd'].dt.month

#counting no. of crimes
Counts_Monthly = df['Month'].value_counts().sort_index()

#plotting the graph
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()
```



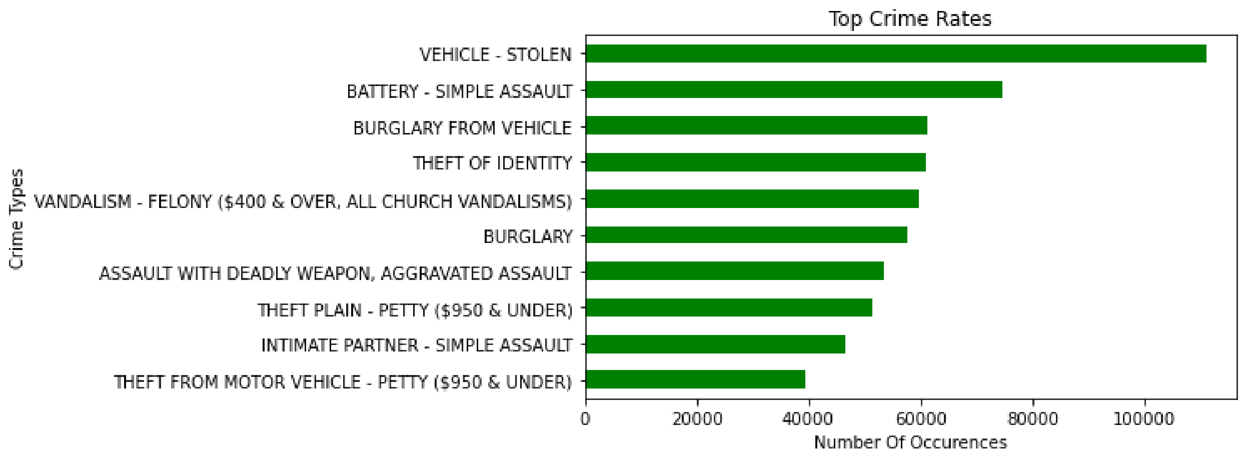
Analysis:

The bar plot suggests that months Jan and March record the highest crimes whereas November and December are significantly lower, suggesting a drop in crime rates during holiday season.

Top Crime Types Analysis:

In this section, we conducted an analysis to identify and visually represent the most frequently occurring crime types in the dataset. The essential details are as follows: Analysis of Crime Types: We utilized the "value_counts()" function to count the occurrences of various crime types in the dataset. We focused on the top 10 most prevalent crime types, sorting them in ascending order to facilitate presentation. Visual Presentation: For a clear visualization of the leading crime types, we generated a horizontal bar graph. Each horizontal bar within the graph signifies a distinct crime type, with its length indicating the frequency of occurrences. The x-axis displays the count of occurrences, while the y-axis labels the individual crime types.

```
fig,axes=plt.subplots(figsize=(7,4))
df['Crm Cd
Desc'].value_counts().iloc[:10].sort_values().plot(kind='barh',title='
Top Crime Rates',color='green')
plt.xlabel('Number Of Occurences')
plt.ylabel('Crime Types')
plt.show()
```



Analysis:

The crime type that has the largest accounts of being reported are Stolen Vehicle with more than 1,00,000 occurrences.

Crime Rates Across Regions Analysis:

In this section, we focused on examining the distribution of crime incidents across different regions or areas within the dataset. The key elements of this analysis are summarized as follows:

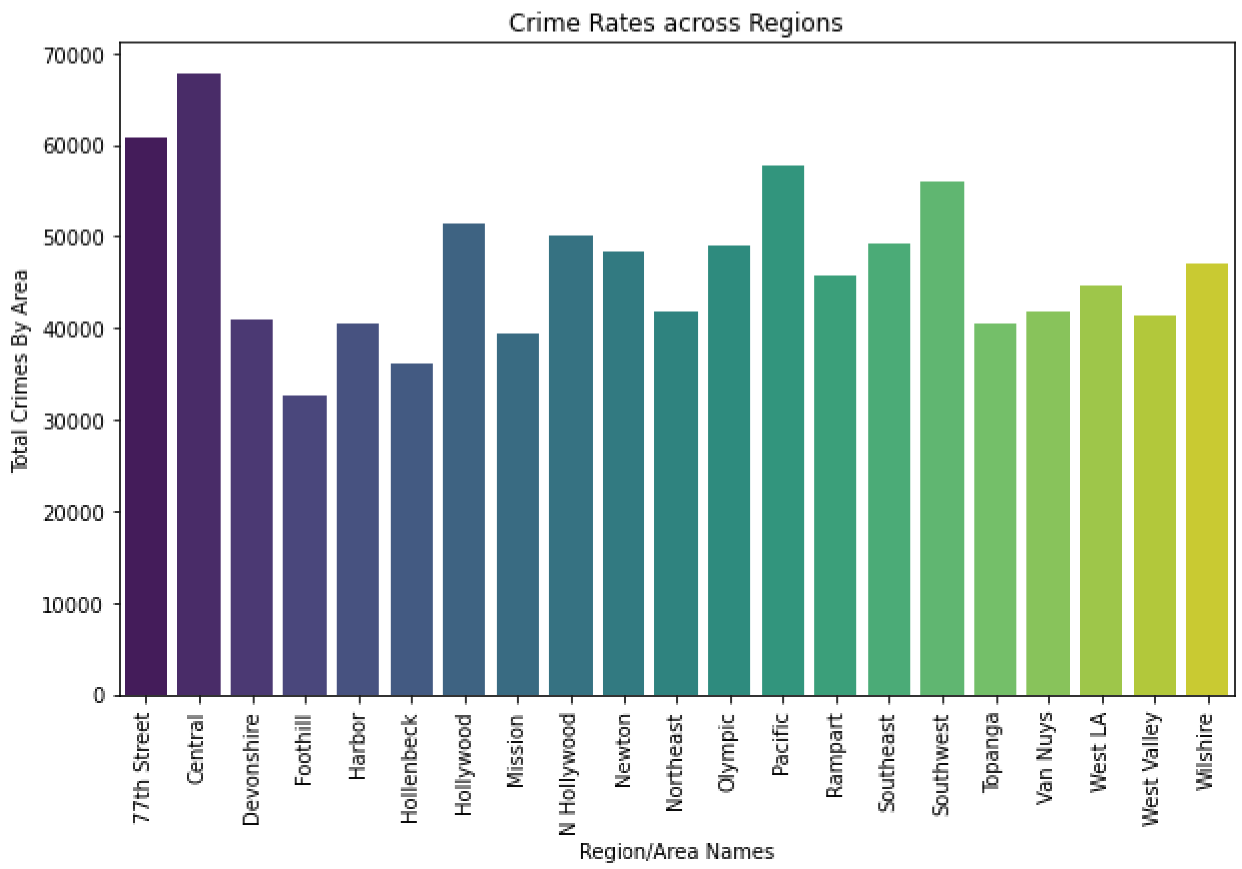
Analysis by Areas:

The dataset was analyzed by grouping it based on the "AREA NAME" field, which delineates the distinct regions or areas where crimes are reported.We employed the "count()" function to calculate the number of recorded crime incidents in each area.

Visual Presentation:

To vividly depict the distribution of crime rates across regions, we opted for a bar plot. Each bar in the plot corresponds to a unique region or area, and its height represents the total count of crimes reported within that specific area.The x-axis is labeled with the names of the regions or areas, while the y-axis quantifies the total number of reported crimes.

```
area_crimes = df['AREA NAME'].groupby(df['AREA NAME']).count()

plt.figure(figsize =(10,6))
sns.barplot(x=area_crimes.index, y= area_crimes.values, palette =
'viridis')

plt.title('Crime Rates across Regions')
plt.xlabel('Region/Area Names')
plt.ylabel('Total Crimes By Area')
plt.xticks(rotation=90)
plt.show()
```



Analysis:

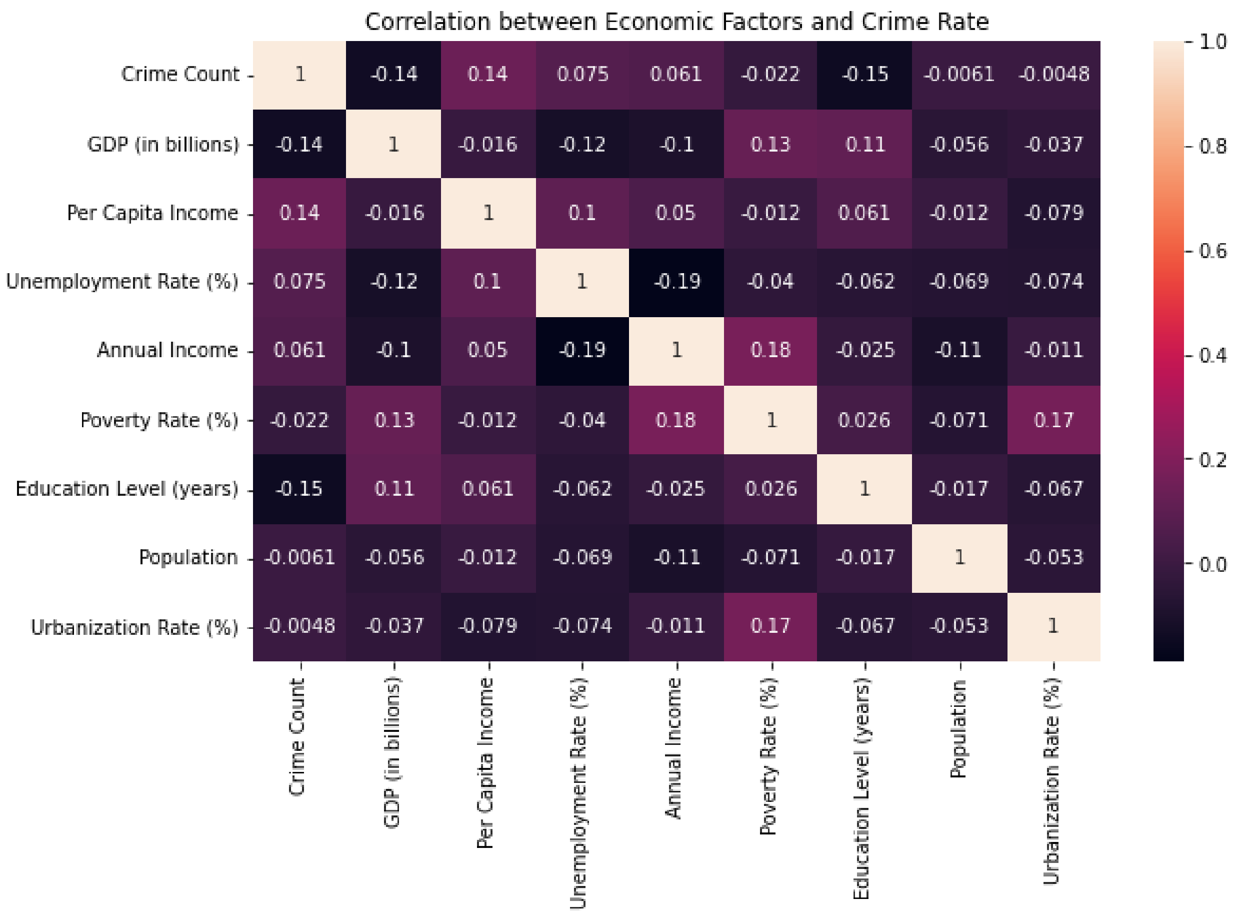
The central region has the most number of total crimes recorded from the bar plot .

Correlation Between Economic Factors and Crime Rate Analysis:

In this section, we explored the potential relationships between economic factors and crime rates. We sought to understand whether changes in economic conditions, as represented by various economic indicators, are correlated with fluctuations in crime rates. Here are the key components of this analysis: Data Aggregation: We began by aggregating and grouping the crime data based on the "Date Rptd" field. This step allowed us to examine the daily distribution of crime incidents. Economic Dataset Integration: We integrated an economic dataset, "Los Angeles Economic Dataset," which provides valuable economic indicators such as Gross Domestic Product (GDP), job growth, unemployment rate, cost of living index, tax rates, and labor costs.To combine this economic dataset with the crime data, we introduced a "Crime count" column that represents the count of crime incidents for each day. Correlation Analysis: The focus of our analysis was to explore potential correlations between economic factors and crime rates. To quantify these correlations, we calculated the correlation coefficients between the "Crime count" and various economic indicators, as mentioned earlier.The correlation matrix reveals the strength and direction of relationships between crime rates and economic variables.

```
df1=pd.read_csv('/Users/apple/DAE/FDA/Project/
Crime_Data_from_2020_to_Present.csv')
```

```
df2=pd.read_csv('/Users/apple/DAE/FDA/Project/la_economicdata.csv')
df1['Date Rptd'] = pd.to_datetime(df1['Date Rptd'])

crm_dt = df1.copy()
crm_dt = crm_dt.reset_index()
crm_dt['Month'] = crm_dt['Date Rptd'].dt.month
crm_dt['Year'] = crm_dt['Date Rptd'].dt.year
crm_dt = crm_dt.groupby(['Year', 'Month']).count()

LA_Economy_data =
pd.read_csv('/Users/apple/DAE/FDA/Project/la_economicdata.csv')
LA_Economy_data['Crime Count'] = crm_dt['DR_NO'].reset_index()
['DR_NO']


correlation = LA_Economy_data[['Crime Count', 'GDP (in billions)',
'Per Capita Income','Unemployment Rate (%)','Annual Income','Poverty
Rate (%)','Education Level (years)','Population','Urbanization Rate
(%)']].corr()
plt.figure(figsize=(10, 6))
sns.heatmap(data = correlation,annot=True)
plt.title('Correlation between Economic Factors and Crime Rate')
plt.show()
```



Day of the Week Crime Analysis:

This section of our analysis delves into the correlation between the day of the week and the frequency of specific crime types. The primary objective is to determine whether certain crimes exhibit day-of-week patterns. The analysis is structured as follows:

Day of the Week Extraction:

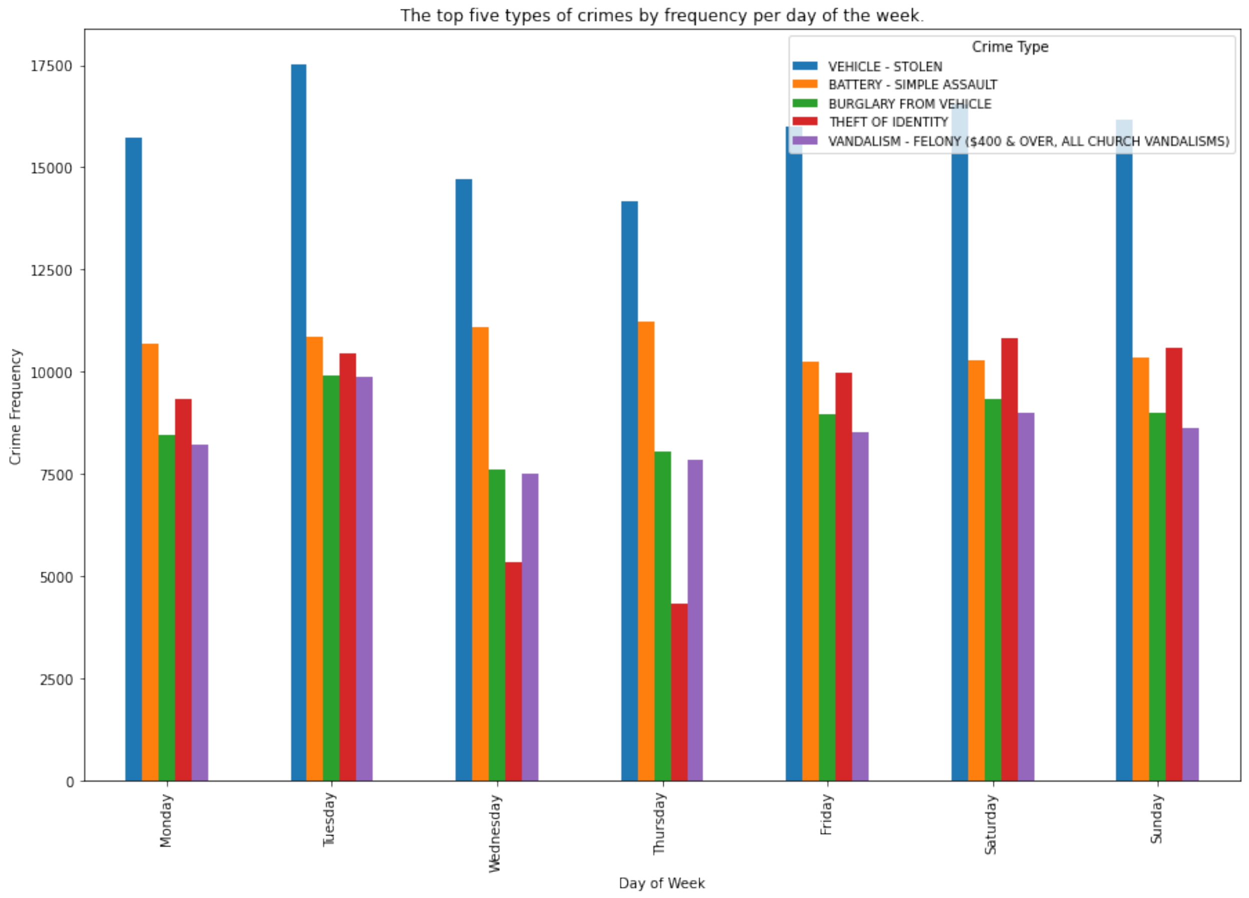
We initiated the analysis by extracting the day of the week from the "Date Rptd" column. This allowed us to categorize each reported crime based on the day of the week. Crime Type and Day-of-Week Frequencies: To gain insights into day-of-week crime patterns, we grouped the data by both the "Day_of_Week" and "Crm Cd Desc" fields. This grouping facilitated the counting of occurrences for each crime type across all days of the week.

Top Crime Types:

Given the abundance of crime types, we narrowed our focus to the top N (in this case, 5) most frequently occurring crime types. These crime types were identified based on their frequency in the dataset.

```
df[ 'Day_of_Week'] = pd.to_datetime(df['Date Rptd']) .dt.day_name()
# Group the data by 'Day_of_Week' and 'Crm Cd Descand count the
```

```
occurrences
crime_type_day_of_week = df.groupby(['Day_of_Week','Crm Cd
Desc']).size().unstack(fill_value=0)
# Get the top N crime types
top_N = 5
top_crime_types = df['Crm Cd Desc'].value_counts().head(top_N).index
crime_type_day_of_week = crime_type_day_of_week[top_crime_types]

crime_type_day_of_week.plot(kind='bar')
plt.title('The top five types of crimes by frequency per day of the
week.')
plt.xlabel ('Day of Week')
plt.ylabel ('Crime Frequency')
plt.rcParams["figure.figsize"] = (15,10)
plt.legend(title='Crime Type',loc='upper right',fontsize=9)

plt.xticks(range(0,7), ['Monday', 'Tuesday', 'Wednesday', 'Thursday',
'Friday', 'Saturday', 'Sunday'], rotation=90)
plt. show()
```



Analysis:

Tuesdays have the highest number of crimes recorded and as seen in the trends before, Stolen Vehicle crimes are recorded the most.

Temporal Analysis:

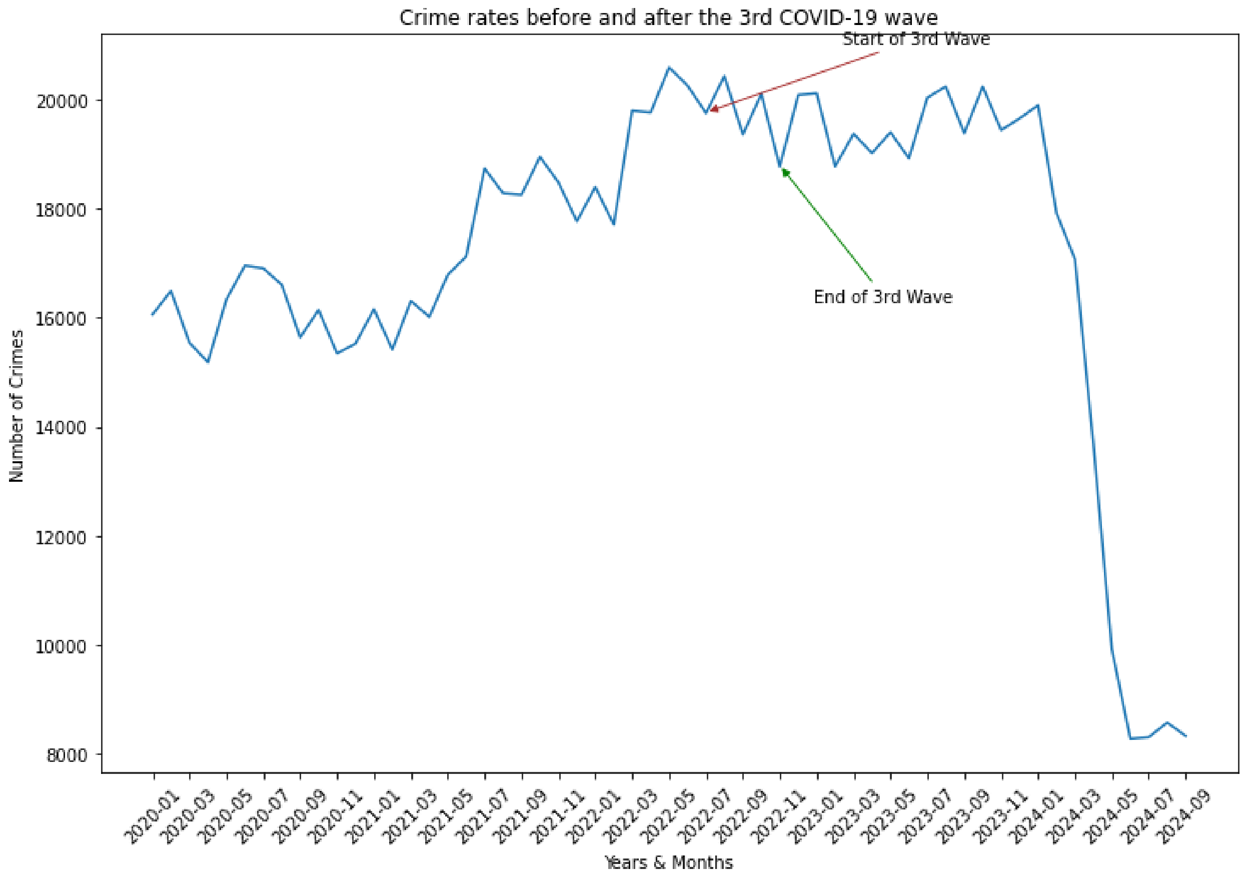
We started by converting the "Date Rptd" column to a datetime data type, which is essential for accurate temporal analysis. Monthly Crime Counts: Next, we counted the number of reported crimes for each month over multiple years. This allowed us to observe the temporal patterns and trends in crime rates. Visualization: We created a line plot to visualize the variations in crime rates over time. The x-axis represents the years and months, while the y-axis indicates the number of reported crimes.

Notable Events:

We identified significant events, specifically the "3rd Wave" of the COVID-19 pandemic, and marked the start and end dates of this wave on the plot. This provides context for understanding how crime rates correlate with major events.

```
df['Date Rptd']= pd.to_datetime(df['Date Rptd'])
month_count = df.groupby([df['Date Rptd'].dt.year, df['Date 
Rptd'].dt.month]).size()

plt.figure(figsize =(12,8))
month_count.plot(kind='line')

# Construct xtick_labels for the x-axis
xtick_labels = [f'{year}-{month:02}' for year, month in 
month_count.index]

# Print xtick_labels for debugging purposes
#print(xtick_labels)

# Plot x-ticks at every second label for better readability
plt.xticks(range(0, len(xtick_labels), 2), xtick_labels[::2], rotation
= 45)
strt_3rd_wave_date = '2022-07'  # Example start date of the 3rd wave
strt_label = 'Start of 3rd Wave'
last_3rd_wave_date = '2022-11'  # Example end date of the 3rd wave
last_label = 'End of 3rd Wave'

# Annotate start of 3rd wave
if strt_3rd_wave_date in xtick_labels:
    plt.annotate(strt_label, 
xy=(xtick_labels.index(strt_3rd_wave_date), 19770), xytext=(80, 40),
                 textcoords='offset points', 
arrowprops=dict(arrowstyle='-|>', lw=0.9, color='brown'), fontsize=10)
else:
    print(f"'{strt_3rd_wave_date}' not found in xtick_labels")
```

```
# Annotate end of 3rd wave
if last_3rd_wave_date in xtick_labels:
    plt.annotate(last_label,
xy=(xtick_labels.index(last_3rd_wave_date), 18800), xytext=(20, -80),
                 textcoords='offset points',
arrowprops=dict(arrowstyle='-|>', lw=0.9, color='green'), fontsize=10)
else:
    print(f"'{last_3rd_wave_date}' not found in xtick_labels")

plt.xlabel('Years & Months')
plt.ylabel('Number of Crimes')
plt.title('Crime rates before and after the 3rd COVID-19 wave')

plt.show()
```



Analysis:

The graph clearly shows a fluctuation in the number of crimes recorded during the period of the 3rd wave of covid. The highest or lowest crime rates in this period do not correspond to the highest or lowest ever recorded.

Distribution based on Age, Gender and Crime Type:

Age Distribution Histogram: The histogram illustrates the distribution of victim ages. It provides a visual representation of the frequency of different age groups within the dataset. The x-axis represents age, the y-axis represents frequency, and the data is displayed in a sky-blue color. The plot indicates the age distribution of crime victims.

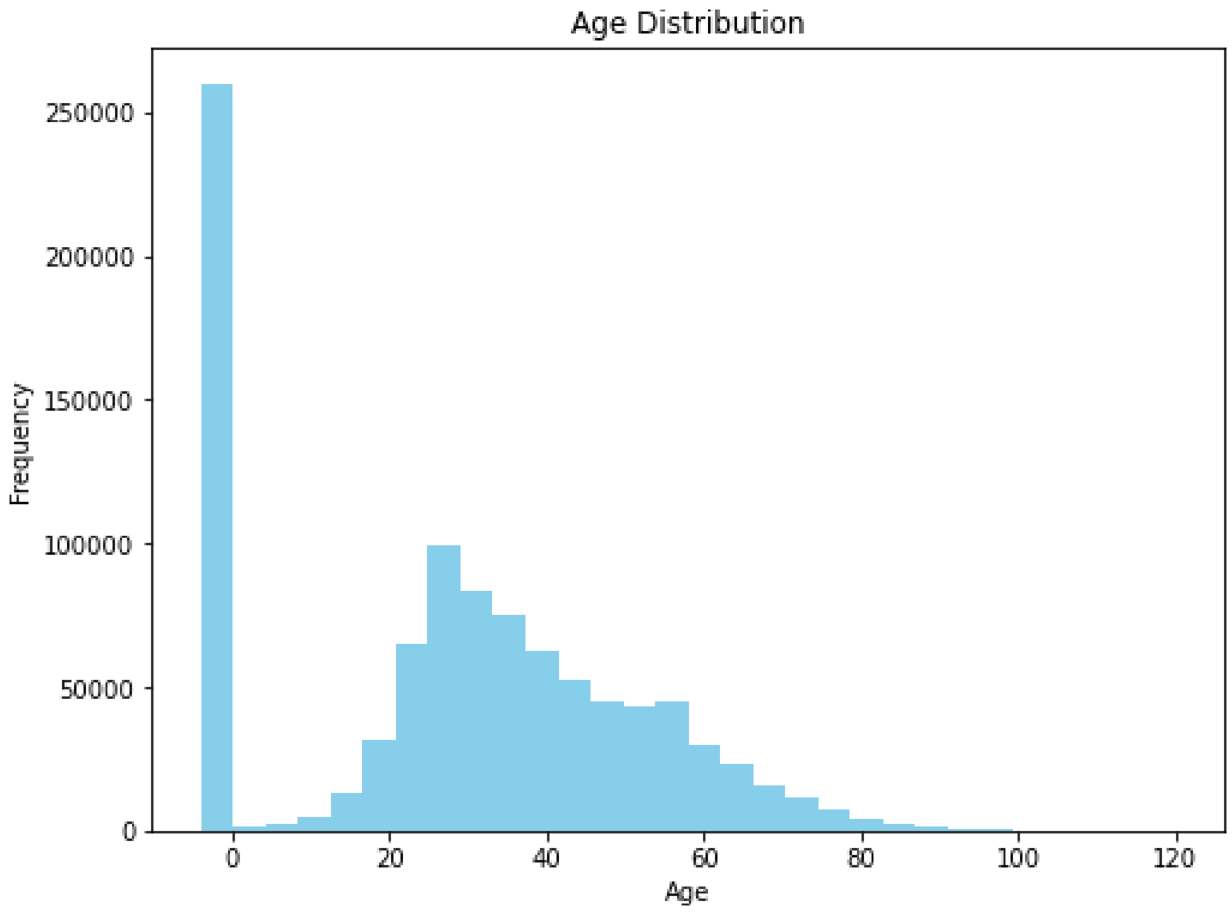
Gender Distribution Bar Chart:

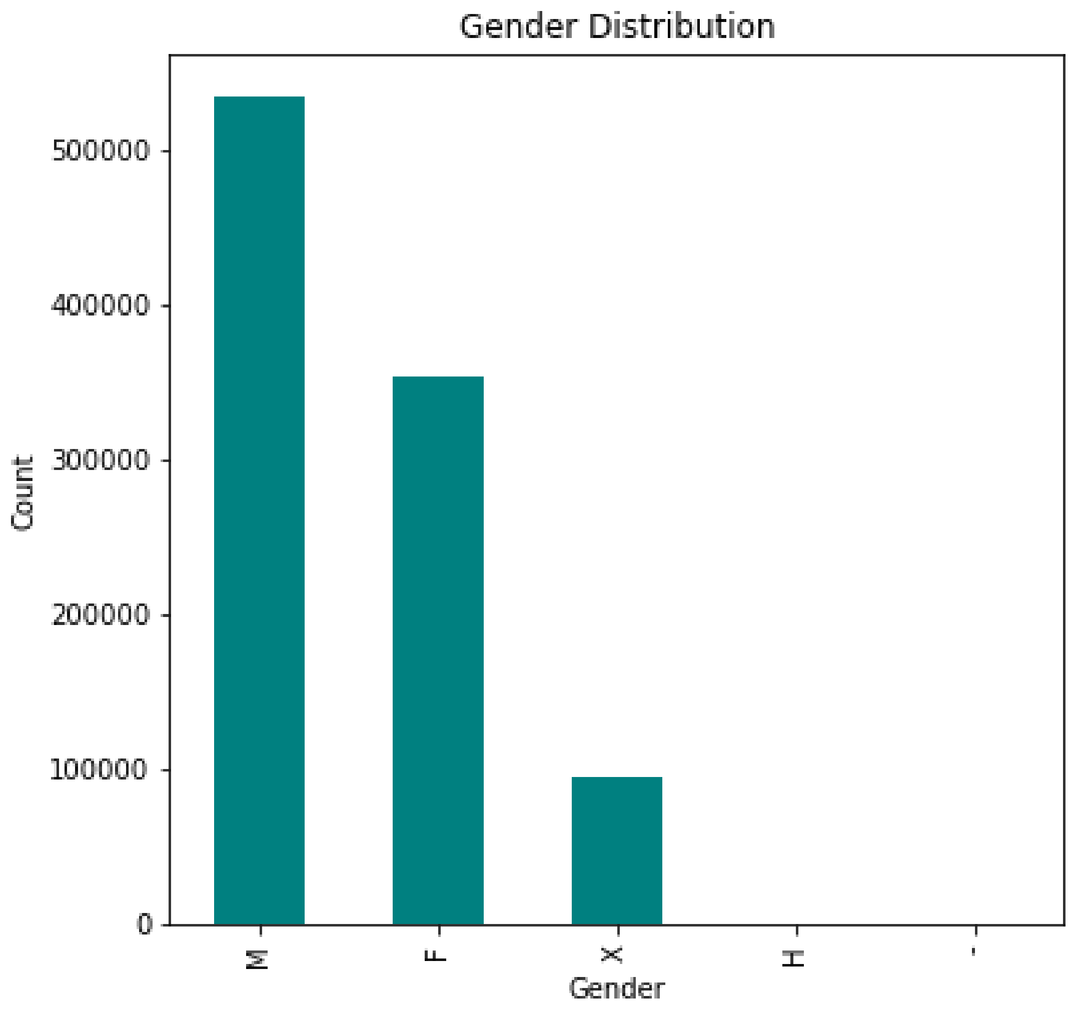
A bar chart is used to display the distribution of victim genders. It shows the count of male and female victims. The x-axis represents gender, the y-axis represents the count, and teal bars represent the male and female categories. This chart offers a visual comparison of gender distribution in the dataset.

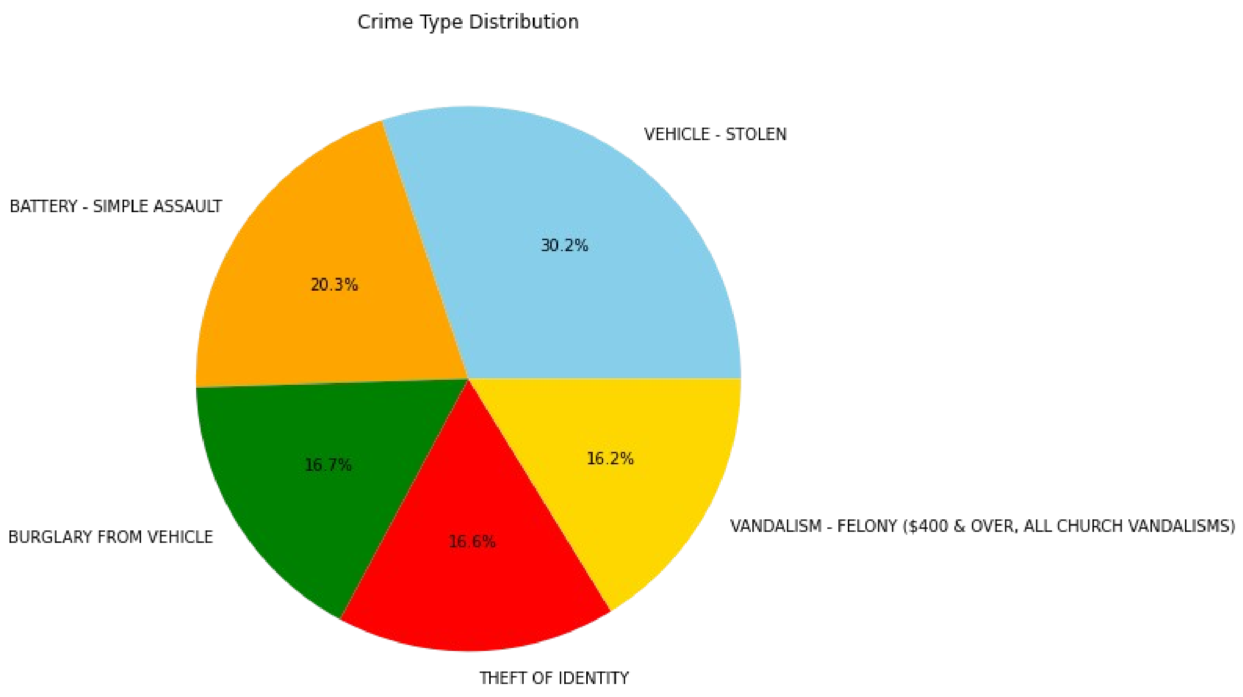
Top Crime Types Pie Chart:

A pie chart visualizes the distribution of the top 5 most frequent crime types. Each slice of the pie represents a different crime type, with colors denoting the categories. The chart provides a clear illustration of the relative prevalence of these crime types, expressed as percentages of the whole.

```python
crime_counts=df['Crm Cd Desc'].value_counts()
top_crime_counts=crime_counts[:5]

plt.figure(figsize = (8,6))
plt.hist(df['Vict Age'], bins=30 ,color= 'skyblue')
plt.title('Age Distribution')
plt.xlabel('Age')
plt.ylabel('Frequency')
plt.show()
plt.figure(figsize=(6,6))
df['Vict Sex'].value_counts().plot(kind ='bar', color ='teal')
plt.title('Gender Distribution')
plt.xlabel('Gender')
plt.ylabel('Count')
plt.show()

plt.figure(figsize=(8,8))
top_crime_counts.plot(kind ='pie', autopct ='%1.1f%%', colors =
['skyblue', 'orange', 'green', 'red', 'gold'])
plt.title('Crime Type Distribution')
plt.ylabel('')
plt.show()
```


Age Distribution
Frequency
250000
200000
150000
100000
50000
0
0
20
40
60
80
100
120
Age


Gender Distribution
Count
500000
400000
300000
200000
100000
0
M
F
X
H
-
Gender



Analysis:

The age, gender and the most commonly occurring crimes have been visualized above. They neatly outline how the demographic affects the crime rates and what the recurring types of crimes are.

5. Advanced Analysis

The dataset was systematically organized by arranging incidents chronologically based on their occurrence date. This chronological arrangement allows for a clear understanding of how events unfolded over time. Subsequently, the data was resampled to a monthly frequency, summarizing the number of reported crimes for each month. This transformation provides a broader temporal perspective, offering insights into any overarching trends or patterns in criminal activity over the specified time frame. These steps lay the groundwork for deeper temporal analysis and trend identification in the subsequent phases of our investigation.

```
df.set_index('DATE OCC',inplace=True)
df.sort_index(inplace=True)
crime=df.resample('M').size()
```

Time Series Analysis:

In the below code we have applied ARIMA(1,1,1) time series model to the resampled crime data. This model aims to capture underlying patterns and relationships within the data. The results of the model training and fitting were summarized, providing insights into its performance and the significance of its parameters. This analysis is crucial for understanding and potentially predicting future crime trends, which can inform law enforcement and policy-making efforts.

```python
import pandas as pd
from statsmodels.tsa.arima.model import ARIMA

# Fit an ARIMA(1,1,1) model to the 'Crime Count' time series
model = ARIMA(crime, order=(1, 1, 1))
results = model.fit()

# Print the model summary
print(results.summary())
```

```
                               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:                        16:17:29   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
```

```
==============================================================================
=============

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

Crime Count Forecasting for the Next 12 Months:

We utilized the ARIMA model to forecast crime counts for the next 12 months. The forecasted values, along with their confidence intervals, were computed and displayed. This information is vital for proactive planning and resource allocation in law enforcement and public safety efforts, providing an estimate of expected crime levels and the level of uncertainty associated with these predictions.

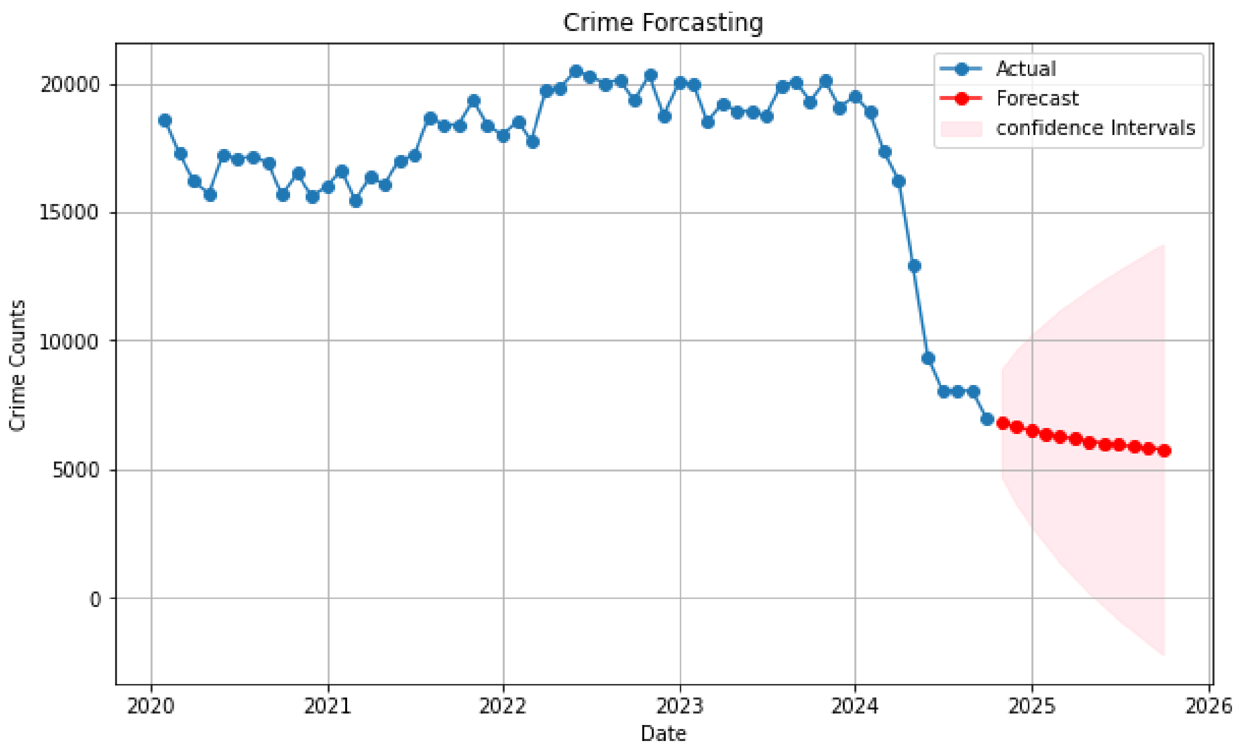
```
forecast_steps=12
forecast=results.get_forecast(steps=forecast_steps)
forecast_values=forecast.predicted_mean
confidence_intervals=forecast.conf_int()

print('Forecasted Crime Counts:')
print(forecast_values)
print('\nConfidence Intervals:')
print(confidence_intervals)

Forecasted Crime Counts:
2024-10-31    6777.453692
2024-11-30    6627.432392
2024-12-31    6493.103020
2025-01-31    6372.824231
2025-02-28    6265.126363
2025-03-31    6168.693477
2025-04-30    6082.347277
2025-05-31    6005.032717
2025-06-30    5935.805106
2025-07-31    5873.818564
2025-08-31    5818.315690
2025-09-30    5768.618306
Freq: M, Name: predicted_mean, dtype: float64

Confidence Intervals:
                lower y       upper y
2024-10-31  4678.450233   8876.457150
2024-11-30  3622.816476   9632.048307
2024-12-31  2771.881269  10214.324770
2025-01-31  2031.400356  10714.248107
2025-02-28  1364.815364  11165.437363
2025-03-31   753.179205  11584.207749
2025-04-30   185.029104  11979.665451
```

```
2025-05-31  -347.289375  12357.354810
2025-06-30  -849.251346  12720.861557
2025-07-31 -1324.975327  13072.612455
2025-08-31 -1777.679648  13414.311028
2025-09-30 -2209.954386  13747.190998
```

Crime Data Visualization and Forecasting:

The provided code generates a visual representation of crime data using a line plot. It includes: Actual Data: Displayed in blue dots, representing observed crime counts over time. Forecasted Values: Shown in red, indicating predicted crime counts based on the ARIMA model. Confidence Intervals: Depicted as a shaded area around the forecasted values, representing the range within which future values are likely to fall.

```
plt.figure(figsize=(10,6))
plt.plot(crime.index,crime.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 Forcasting')
plt.xlabel('Date')
plt.ylabel('Crime Counts')
plt.legend()
plt.grid(True)
plt.show()
```



Analysis:

The ARIMA model predicts that the crime rates are expected to fall between the confidence levels and shows the predicted trends in the upcoming year.

Victim Demographics and Crime Type Frequencies:

We conducted a descriptive analysis of age and gender variables, as well as an assessment of the frequency of different crime types. Here's the interpretation: Summary Statistics for Age and Gender: Descriptive statistics, including measures like mean, standard deviation, minimum, maximum, and quartiles, were computed for the 'Vict Age' variable. This provides insights into the distribution and central tendencies of victim ages. For 'Vict Sex', the count of unique categories (e.g., male, female) was provided along with the most frequent category.

Count of Crime Types:

We calculated the occurrence of each crime type ('Crm Cd Desc'). This analysis provides a breakdown of the frequency of different types of crimes recorded in the dataset. Top 5 Crime Types: The top 5 most frequently occurring crime types were identified based on their counts. This subset of crime types is of particular interest due to their high prevalence.

```
print(df[['Vict Age']].describe())
print('\n')
print(df[['Vict Sex']].describe())
print('\n')
print(crime_counts)
```

```
            Vict Age
count  982638.000000
mean       29.079817
std        21.970094
min        -4.000000
25%         0.000000
50%        30.000000
75%        44.000000
max       120.000000


       Vict Sex
count    982638
unique        5
top           M
freq     535214


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: Crm Cd Desc, Length: 140, dtype: int64
```

Analysis:

From the initial analyses we can observe that the average age for the criminal is almost 30 years and the Male Gender has committed most of the crimes with a frequency of 535,214.

Conclusion

The exploratory data analysis (EDA) of crime data in Los Angeles from 2020 to the present has yielded significant insights into the city's criminal activity. We identified clear temporal trends, with certain crime types showing noticeable increases during specific periods. By applying the ARIMA time series model, we were able to forecast future crime patterns, providing a valuable resource for proactive law enforcement planning. The forecasts, along with their confidence intervals, offer a degree of certainty that can assist in informed decision-making.

Overall, this EDA offers a detailed understanding of crime dynamics in Los Angeles. The insights gained can inform law enforcement strategies, optimize resource distribution, and guide policy-making efforts aimed at improving public safety and reducing crime in the city.