CRIME DATA ANALYSIS

IE6400 FOUNDATIONS DATA ANALYTICS ENGINEERING

FALL 2023

Project Report

Group Number 5:

Pooja Laxmi Sankarakameswaran(002278676)

Rishwanth Reddy Yadamakanti(002697773)

Sruthi Gandla(0028515998)

Tejesvani Muppara Vijayaram(002299364)

Naresh Gajula(002648708)

Introduction

A multifaceted strategy is necessary to comprehend and manage complex societal challenges like crime in today's data-driven environment. An essential tool in this effort is exploratory data analysis (EDA), which provides a methodical and perceptive prism through which to evaluate and analyze crime data. This report sets out to analyze a dataset that contains crime episodes in order to find patterns, trends, and underlying causes.

Numerous socioeconomic, demographic, and environmental factors influence crime, which is a complex phenomena. We hope to shed light on the dynamics of criminal activity and possibly unearth insights that can guide evidence-based tactics for law enforcement, public policy, and community engagement by carefully examining the details of the data that is accessible.

Through a rigorous application of statistical techniques, visualization tools, and domain knowledge, this analysis aims to extract actionable insights from the available crime data. It is imperative to approach this task with an open mind, recognizing that crime is a complex, multifaceted phenomenon influenced by a myriad of factors.

This report endeavors to equip stakeholders with a nuanced understanding of crime patterns, fostering evidence-based decision-making and proactive strategies to enhance public safety and community well-being. By delving into the intricacies of crime data, we seek to contribute to the ongoing efforts to create safer, more secure environments for all.

Data Preprocessing and Analysis

1. Data Acquisition:

In the data acquisition phase, we utilized the Python programming language and its data manipulation library, pandas, to access and load the dataset for our analysis. The dataset, titled 'Crime_Data_from_2020_to_Present.csv', was obtained from the provided link [Crime Data from 2020 to Present]. This CSV file contains the crime data spanning from the year 2020 to the present. By employing the pandas library, we imported and structured the dataset in a way that facilitates its manipulation and analysis. The **'pd.read_csv'** function was used to read the contents of the CSV file and create a DataFrame, a tabular data structure commonly used in data analysis. This DataFrame, named **'crime_data'**, serves as the foundation for our subsequent exploratory data analysis.

```
import pandas as pd

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

2. Data Inspection:

In any data analysis project, one of the initial steps is to thoroughly inspect the dataset to understand its structure and content. This process involves the following key aspects:

Displaying the First Few Rows:

The first and most straightforward step is to display the initial rows of the dataset. This allows us to visually inspect the data and gain insights into its format and contents. The Pandas library provides a convenient function called **head**() to facilitate this. Here's an example of how it's used:

```
# Displaying Reading the first 10 rows of the dataset

crime_data.head(10)
```

Result:

	DR_NO	Date Rptd	DATE OCC	TIME OCC	AREA	AREA NAME	Rpt Dist No	Part 1-2	Crm Cd	Crm Cd Desc	...	Status	Status Desc	Crm Cd 1	Crm Cd 2	Crm Cd 3	Crm Cd 4	LOCATION	Cross Street	LAT	LON
0	10304468	01/08/2020 12:00:00 AM	01/08/2020 12:00:00 AM	2230	3	Southwest	377	2	624	BATTERY - SIMPLE ASSAULT	...	AO	Adult Other	624.0	NaN	NaN	NaN	1100 W 39TH PL	NaN	34.0141	-118.2978
1	190101086	01/02/2020 12:00:00 AM	01/01/2020 12:00:00 AM	330	1	Central	163	2	624	BATTERY - SIMPLE ASSAULT	...	IC	Invest Cont	624.0	NaN	NaN	NaN	700 S HILL ST	NaN	34.0459	-118.2545
2	200110444	04/14/2020 12:00:00 AM	02/13/2020 12:00:00 AM	1200	1	Central	155	2	845	SEX OFFENDER REGISTRANT OUT OF COMPLIANCE	...	AA	Adult Arrest	845.0	NaN	NaN	NaN	200 E 6TH ST	NaN	34.0448	-118.2474
3	191501505	01/01/2020 12:00:00 AM	01/01/2020 12:00:00 AM	1730	15	N Hollywood	1543	2	745	VANDALISM - MISDEAMEANOR ($399 OR UNDER)	...	IC	Invest Cont	745.0	998.0	NaN	NaN	5400 CORTEEN PL	NaN	34.1685	-118.4019
4	191921269	01/01/2020 12:00:00 AM	01/01/2020 12:00:00 AM	415	19	Mission	1998	2	740	VANDALISM - FELONY ($400 & OVER, ALL CHURCH VA...	...	IC	Invest Cont	740.0	NaN	NaN	NaN	14400 TITUS ST	NaN	34.2198	-118.4468
5	200100501	01/02/2020 12:00:00 AM	01/01/2020 12:00:00 AM	30	1	Central	163	1	121	RAPE, FORCIBLE	...	IC	Invest Cont	121.0	998.0	NaN	NaN	700 S BROADWAY	NaN	34.0452	-118.2534
6	200100502	01/02/2020 12:00:00 AM	01/02/2020 12:00:00 AM	1315	1	Central	161	1	442	SHOPLIFTING - PETTY THEFT ($950 & UNDER)	...	IC	Invest Cont	442.0	998.0	NaN	NaN	700 S FIGUEROA ST	NaN	34.0483	-118.2631
7	200100504	01/04/2020 12:00:00 AM	01/04/2020 12:00:00 AM	40	1	Central	155	2	946	OTHER MISCELLANEOUS CRIME	...	IC	Invest Cont	946.0	998.0	NaN	NaN	200 E 6TH ST	NaN	34.0448	-118.2474
8	200100507	01/04/2020 12:00:00 AM	01/04/2020 12:00:00 AM	200	1	Central	101	1	341	THEFT-GRAND ($950.01 & OVER)EXCPT,GUNS,FOWL,LI...	...	IC	Invest Cont	341.0	998.0	NaN	NaN	700 BERNARD ST	NaN	34.0677	-118.2398
9	201710201	06/19/2020 12:00:00 AM	05/26/2020 12:00:00 AM	1925	17	Devonshire	1708	1	341	THEFT-GRAND ($950.01 & OVER)EXCPT,GUNS,FOWL,LI...	...	AO	Adult Other	341.0	NaN	NaN	NaN	11900 BALBOA BL	NaN	34.2864	-118.5021

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:

```
# Checking the data types

crime_data.dtypes
```

Result:

```
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
```

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.

```
# Analysing basic information of every column

crime_data.info()
```

Result:

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 825212 entries, 0 to 825211
Data columns (total 28 columns):
 #   Column          Non-Null Count   Dtype  
---  ------          --------------   -----  
 0   DR_NO           825212 non-null  int64  
 1   Date Rptd       825212 non-null  object 
 2   DATE OCC        825212 non-null  object 
 3   TIME OCC        825212 non-null  int64  
 4   AREA            825212 non-null  int64  
 5   AREA NAME       825212 non-null  object 
 6   Rpt Dist No     825212 non-null  int64  
 7   Part 1-2        825212 non-null  int64  
 8   Crm Cd          825212 non-null  int64  
 9   Crm Cd Desc     825212 non-null  object 
 10  Mocodes         711064 non-null  object 
 11  Vict Age        825212 non-null  int64  
 12  Vict Sex        716683 non-null  object 
 13  Vict Descent    716675 non-null  object 
 14  Premis Cd       825202 non-null  float64
 15  Premis Desc     824724 non-null  object 
 16  Weapon Used Cd  287714 non-null  float64
 17  Weapon Desc     287714 non-null  object 
 18  Status          825212 non-null  object 
 19  Status Desc     825212 non-null  object 
 20  Crm Cd 1        825202 non-null  float64
 21  Crm Cd 2        60707 non-null   float64
 22  Crm Cd 3        2039 non-null    float64
 23  Crm Cd 4        61 non-null      float64
 24  LOCATION        825212 non-null  object 
 25  Cross Street    131869 non-null  object 
 26  LAT             825212 non-null  float64
 27  LON             825212 non-null  float64
dtypes: float64(8), int64(7), object(13)
memory usage: 176.3+ MB
```

Column names are often self-explanatory, but when working with complex datasets, it's helpful to review any available descriptions or metadata associated with the columns. This information can provide context and help in understanding the meaning of each column.

```
# Displaying column names and descriptions

column_df = pd.DataFrame()
column_df['Column Names'] = list(crime_data.columns)
column_df
```

Result:

	Column Names
0	DR_NO
1	Date Rptd
2	DATE OCC
3	TIME OCC
4	AREA
5	AREA NAME
6	Rpt Dist No
7	Part 1-2
8	Crm Cd
9	Crm Cd Desc
10	Mocodes
11	Vict Age
12	Vict Sex
13	Vict Descent
14	Premis Cd
15	Premis Desc
16	Weapon Used Cd
17	Weapon Desc
18	Status
19	Status Desc
20	Crm Cd 1
21	Crm Cd 2
22	Crm Cd 3
23	Crm Cd 4
24	LOCATION
25	Cross Street
26	LAT
27	LON

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.

```
# Identifying and handling null values

crime_data.isnull().sum()
```

Result:

```
DR_NO                 0
Date Rptd             0
DATE OCC              0
TIME OCC              0
AREA                  0
AREA NAME             0
Rpt Dist No           0
Part 1-2              0
Crm Cd                0
Crm Cd Desc           0
Mocodes          114148
Vict Age              0
Vict Sex         108529
Vict Descent     108537
Premis Cd            10
Premis Desc         488
Weapon Used Cd   537498
Weapon Desc      537498
Status                0
Status Desc           0
Crm Cd 1             10
Crm Cd 2         764505
Crm Cd 3         823173
Crm Cd 4         825151
LOCATION              0
Cross Street     693343
LAT                   0
LON                   0
dtype: int64
```

Common Value Analysis for Handling Missing Data:

Next, we conducted an analysis to identify the most frequently occurring values within specific fields of our dataset. This information is invaluable for devising a strategy to handle missing values in a manner that preserves the integrity of the dataset. By conducting this analysis, we gained a deeper understanding of the prevalent attributes within these specific fields. This information will inform our strategy for handling missing values, ensuring that any imputation or substitution methods are applied judiciously, with consideration for the most common scenarios observed in the dataset. The provided code snippet identifies the most common values in key fields of the dataset.The fields include Mocodes, Vict Sex, Vict Descent, Weapon Used, and Premises information.

```
# Find the most commonly occuring values in all fields to analyze how to handle missing values

print("Common Mocodes :\n" + str(crime_data['Mocodes'].value_counts()[:3]) + "\n")
print("Common Vict Sex :\n" + str(crime_data['Vict Sex'].value_counts()[:3]) + "\n")
print("Common Vict Descent :\n" + str(crime_data['Vict Descent'].value_counts()[:3]) + "\n")
print("Common Weapon Used Cd :\n" + str(crime_data['Weapon Used Cd'].value_counts().iloc[:3]) + "\n")
print("Common Weapon Desc :\n" + str(crime_data['Weapon Desc'].value_counts()[:3]) + "\n")
print("Common Premis Cd :\n" + str(crime_data['Premis Cd'].value_counts().iloc[:3]) + "\n")
print("Common Premis Desc :\n" + str(crime_data['Premis Desc'].value_counts()[:3]) + "\n")
```

Result

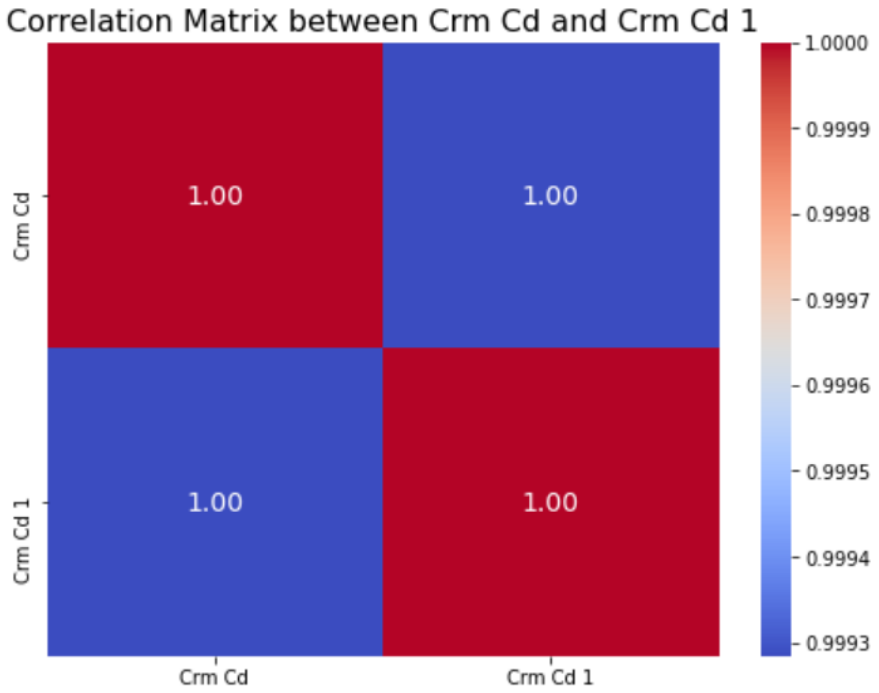
```
Common Mocodes :
0344         33674
0329         18384
1822 0344    10147
Name: Mocodes, dtype: int64

Common Vict Sex :
M    340664
F    303878
X     72050
Name: Vict Sex, dtype: int64

Common Vict Descent :
H    253152
W    168138
B    117571
Name: Vict Descent, dtype: int64

Common Weapon Used Cd :
400.0    154175
500.0     30541
511.0     21085
Name: Weapon Used Cd, dtype: int64

Common Weapon Desc :
STRONG-ARM (HANDS, FIST, FEET OR BODILY FORCE)    154175
UNKNOWN WEAPON/OTHER WEAPON                        30541
VERBAL THREAT                                      21085
Name: Weapon Desc, dtype: int64

Common Premis Cd :
101.0    208794
501.0    139820
502.0    101362
Name: Premis Cd, dtype: int64

Common Premis Desc :
STREET                                       208794
SINGLE FAMILY DWELLING                       139820
MULTI-UNIT DWELLING (APARTMENT, DUPLEX, ETC)    101362
Name: Premis Desc, dtype: int64
```

Correlation Analysis and Null Value Handling:

We examined the relationship between "Crm Cd" and "Crm Cd 1" in the dataset, two fields that likely represent related aspects of crime incidents. The correlation matrix and heatmap revealed a high positive correlation between these fields, indicating that they contain very similar information that moves in the same direction. The heatmap's values were close to 1, signifying a strong positive relationship between "Crm Cd" and "Crm Cd 1."

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

correlation_matrix = crime_data[['Crm Cd', 'Crm Cd 1']].corr()

# Plotting the correlation matrix using seaborn
plt.figure(figsize=(8, 6))
sns.heatmap(correlation_matrix, annot=True, cmap='coolwarm', fmt='.2f', annot_kws={"size": 14})
plt.title('Correlation Matrix between Crm Cd and Crm Cd 1', fontsize=16)
plt.show()
```

Result:



Handling Null Values:

We identified fields, such as Mocodes', 'Weapon Used Cd', 'Weapon Desc',Vict Sex', 'Vict Descent', 'Premis Cd', 'Premis Desc' with missing values which were filled with appropriate placeholders to indicate absence or unspecified information. Fields "Crm Cd 2," "Crm Cd 3," and "Crm Cd 4," had a substantial amount of missing data, so we made the informed decision to remove these columns from the dataset. By eliminating these columns, we streamlined the dataset, enhanced data quality, and focused on more informative variables for further analysis.

```
#Handling Missing values
crime_data['Mocodes'].fillna(0,inplace=True)
crime_data['Weapon Used Cd'].fillna("0",inplace=True)
crime_data['Weapon Desc'].fillna("No Weapon Used",inplace=True)

crime_data.drop(columns = ['Crm Cd 1','Crm Cd 2','Crm Cd 3', 'Crm Cd 4'], inplace = True)
# The Columns have been dropped and therefore aren't dropped again to avoid errors

crime_data['Vict Sex'].fillna("-",inplace=True)
crime_data['Vict Descent'].fillna("-",inplace=True)
crime_data['Premis Cd'].fillna(0.0, inplace=True)
crime_data['Premis Desc'].fillna('NO DESC',inplace=True)
crime_data['Cross Street'].fillna("NO LOCATION RECORDED",inplace=True)

missing_data = crime_data.isnull().sum()
print(missing_data)
```

Result:

```
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
LOCATION          0
Cross Street      0
LAT               0
LON               0
dtype: int64
```

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 "**crime_data.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.

```
# Dropping duplicate rows

crime_data.drop_duplicates().head()
```

Result:

	DR_NO	Date Rptd	DATE OCC	TIME OCC	AREA	AREA NAME	Rpt Dist No	Part 1-2	Crm Cd	Crm Cd Desc	...	Premis Cd	Premis Desc	Weapon Used Cd	Weapon Desc	Status	Status Desc	LOCATION	Cross Street	LAT	LON
0	10304468	01/08/2020 12:00:00 AM	01/08/2020 12:00:00 AM	2230	3	Southwest	377	2	624	BATTERY - SIMPLE ASSAULT	...	501.0	SINGLE FAMILY DWELLING	400.0	STRONG-ARM (HANDS, FIST, FEET OR BODILY FORCE)	AO	Adult Other	1100 W 39TH PL	NO LOCATION RECORDED	34.0141	-118.2978
1	190101086	01/02/2020 12:00:00 AM	01/01/2020 12:00:00 AM	330	1	Central	163	2	624	BATTERY - SIMPLE ASSAULT	...	102.0	SIDEWALK	500.0	UNKNOWN WEAPON/OTHER WEAPON	IC	Invest Cont	700 S HILL ST	NO LOCATION RECORDED	34.0459	-118.2545
2	200110444	04/14/2020 12:00:00 AM	02/13/2020 12:00:00 AM	1200	1	Central	155	2	845	SEX OFFENDER REGISTRANT OUT OF COMPLIANCE	...	726.0	POLICE FACILITY	0	No Weapon Used	AA	Adult Arrest	200 E 6TH ST	NO LOCATION RECORDED	34.0448	-118.2474
3	191501505	01/01/2020 12:00:00 AM	01/01/2020 12:00:00 AM	1730	15	N Hollywood	1543	2	745	VANDALISM - MISDEAMEANOR ($399 OR UNDER)	...	502.0	MULTI-UNIT DWELLING (APARTMENT, DUPLEX, ETC)	0	No Weapon Used	IC	Invest Cont	5400 CORTEEN PL	NO LOCATION RECORDED	34.1685	-118.4019
4	191921269	01/01/2020 12:00:00 AM	01/01/2020 12:00:00 AM	415	19	Mission	1998	2	740	VANDALISM - FELONY ($400 & OVER, ALL CHURCH VA...	...	409.0	BEAUTY SUPPLY STORE	0	No Weapon Used	IC	Invest Cont	14400 TITUS ST	NO LOCATION RECORDED	34.2198	-118.4468

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.

```
# Converting object datatype to date and time datatypes
# This should be executed only once since datatype errors may arise once it is executed

from datetime import datetime as datetime

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

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

# Convert the 'TimeInt' column to a time format
for time_item in time_occ_list:
    time_list.append( (datetime.strptime(str(time_item), "%H%M").time()).strftime("%H:%M") )

crime_data['TIME OCC'] = time_list

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

Result:

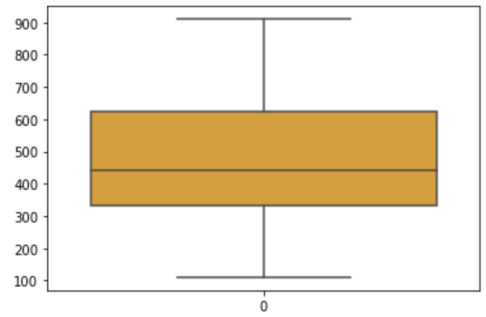
	Date Rptd	DATE OCC	TIME OCC
0	2020-01-08	2020-01-08	22:30
1	2020-01-02	2020-01-01	03:30
2	2020-04-14	2020-02-13	12:00
3	2020-01-01	2020-01-01	17:30
4	2020-01-01	2020-01-01	04:15
...	...	...	...
825207	2023-01-27	2023-01-26	18:00
825208	2023-03-22	2023-03-22	10:00
825209	2023-04-12	2023-04-12	16:30
825210	2023-07-02	2023-07-01	00:01
825211	2023-03-05	2023-03-05	09:00

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.

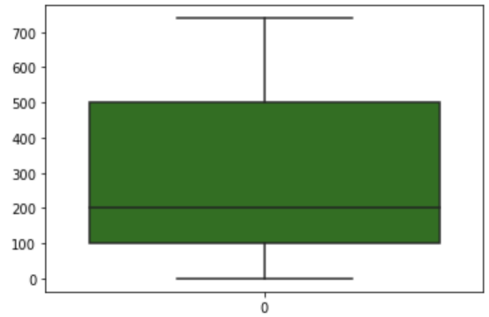
```
# Finding the outliers using Z-score and imputing them with the median values of each field

import numpy as np
from scipy.stats import zscore
import seaborn as sns
import matplotlib.pyplot as plt

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

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

# Using a boxplot to show the distribution
sns.boxplot(data = crime_data['Crm Cd'], color = 'orange')
plt.show()
```

Result:



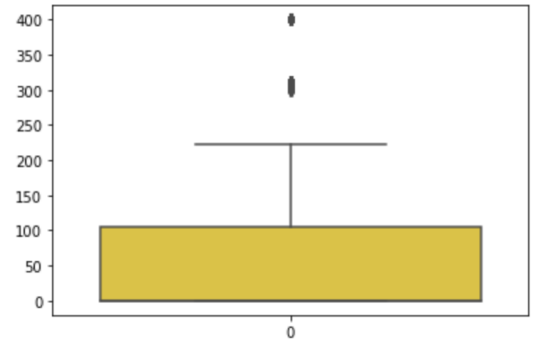

```
z_scores = zscore(crime_data['Premis Cd'])
outliers = (np.abs(z_scores) > 2)

crime_data['Premis Cd'] = np.where(outliers, crime_data['Premis Cd'].median(), crime_data['Premis Cd'])

# Using a boxplot to show the distribution
sns.boxplot(data = crime_data['Premis Cd'], color = 'green')
plt.show()
```

Result:




```
crime_data['Weapon Used Cd'] = crime_data['Weapon Used Cd'].astype(float)
z_scores = zscore(crime_data['Weapon Used Cd'])
outliers = (np.abs(z_scores) > 2)

crime_data['Weapon Used Cd'] = np.where(outliers, crime_data['Weapon Used Cd'].median(), crime_data['Weapon Used Cd'])

# Using a boxplot to show the distribution
sns.boxplot(data = crime_data['Weapon Used Cd'], color = 'gold')
plt.show()
```

Result:



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:

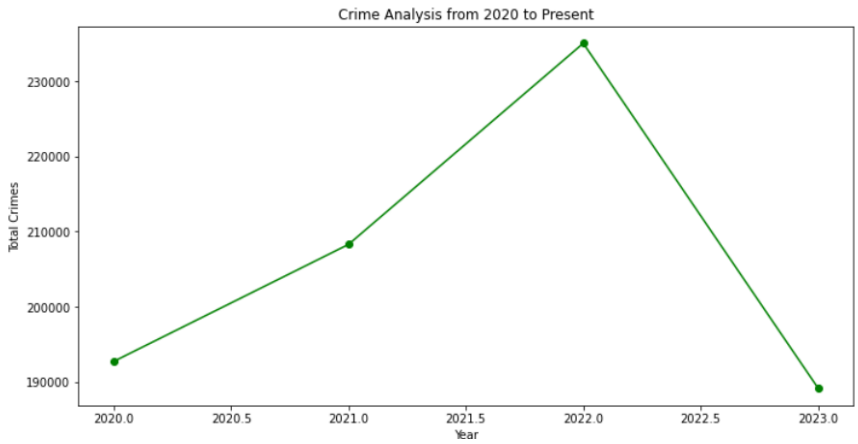
We employed the "**groupby()**" function to group the data based on the "Date Rptd" field, which represents the date when crimes were reported. By using "**crime_data['Date Rptd'].dt.year**," we extracted the reporting year from each date, enabling us to organize the data chronologically by year.
We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

```
# Task - 4.1

# Grouping data by year and count the number of crimes per year
Yearly_Analysis = crime_data['Date Rptd'].groupby(crime_data['Date Rptd'].dt.year).count()

# Yearly_Analysis = crm_data['year'].value_counts().sort_index()

# Creating a time series plot
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.grid(True)
plt.show()
```

Result:

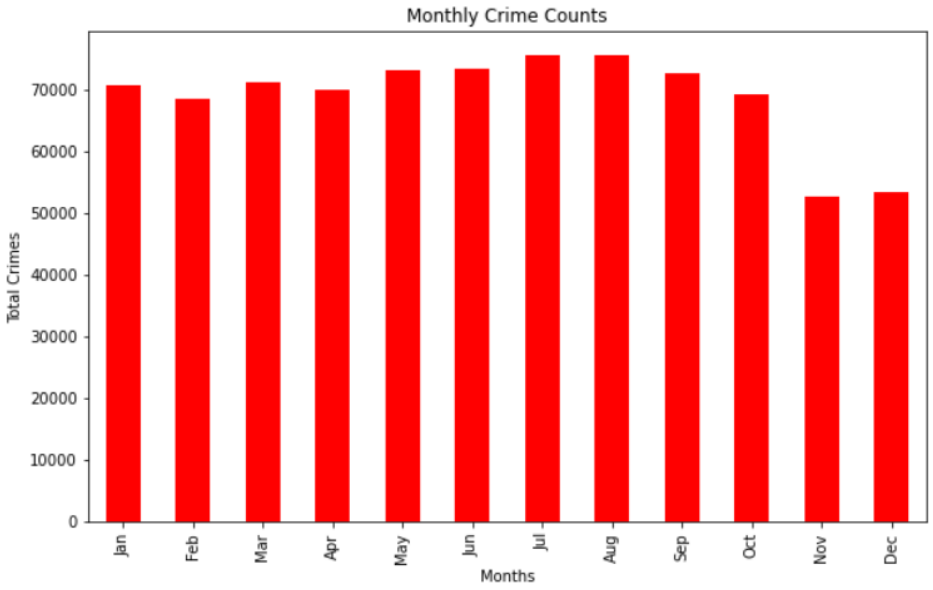


Analysis:
We will be able to see how crime rates have evolved over time by looking at the resultant time series plot, which will show the general *crime patterns from 2020 to the present. The plot shows a* ***significant increase in crime rates in 2022.***

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.

```
# Task - 4.2

# Extracting the month from the "Date Rptd" column
crime_data['Month'] = crime_data['Date Rptd'].dt.month

# Counting the no.of crimes
Counts_Monthly = crime_data['Month'].value_counts().sort_index()

#Initialising the Size of the graph
plt.figure(figsize=(10, 6))

# Creating a bar plot
Counts_Monthly.plot(kind='bar', color='Red')
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:

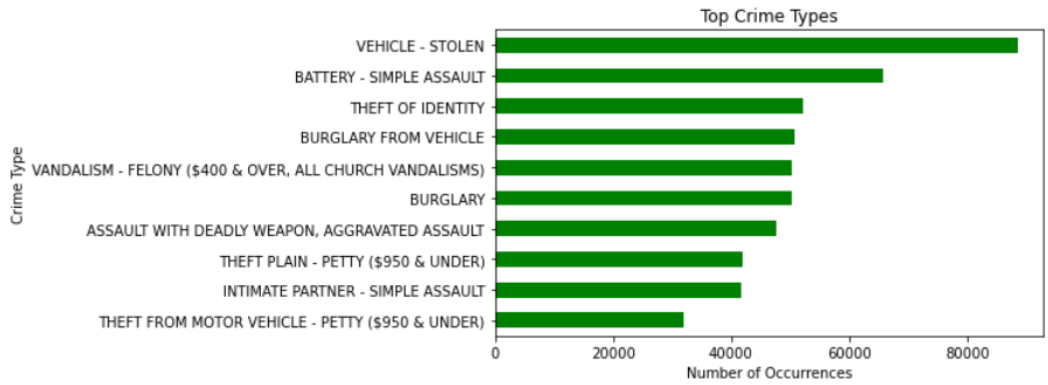


Analysis:
*The bar plot suggests that months **July and August record the highest crimes whereas November and December are significantly lower**, suggesting a drop in crime rates during holiday season.*

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.

```
# Task -4.3

#Initialising the Size of the graph
fig, axes = plt.subplots(figsize=(7, 4))

#Counting the total types of crimes and plotting pie chart
# crm_data['Crm Cd Desc'].value_counts().iloc[:25].sort_values().plot(kind="pie", title = "Top Crime Types")

#Counting the total types of crimes and plotting bar graph
crime_data['Crm Cd Desc'].value_counts().iloc[:10].sort_values().plot(kind="barh", title = "Top Crime Types", color='green')
plt.xlabel('Number of Occurrences')
plt.ylabel('Crime Type')

plt.show()
```

Results:



Analysis:
*The crime type that has the largest accounts of being reported are **Stolen Vehicle with more than 80,000** 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:

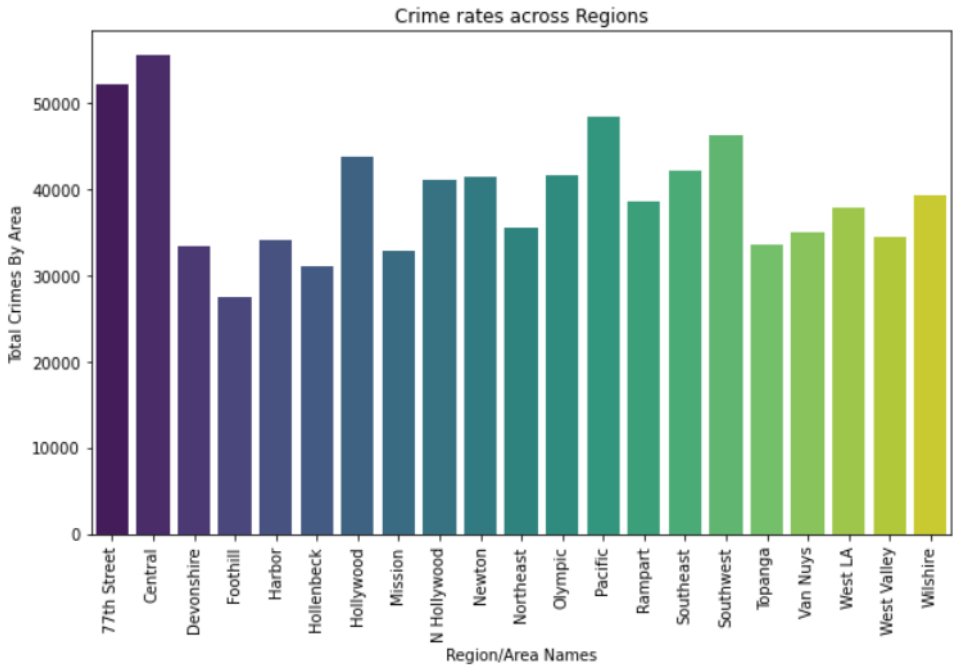
Analysis by Areas: The dataset was analyzed by grouping it based on the "AREA NAME" field, which delineates the distinct regions or areas where crimes are reported.We employed the "count()" function to calculate the number of recorded crime incidents in each area.

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.

```
# Task -4.4

# Area_crimes = crm_data['AREA NAME'].value_counts().sort_index()
Area_crimes = crime_data['AREA NAME'].groupby(crime_data['AREA NAME']).count()

#Initialising the Size of the graph
plt.figure(figsize=(10, 6))

# Creating a bar plot

# Area_crimes.plot(kind='bar')
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()
```

Result:

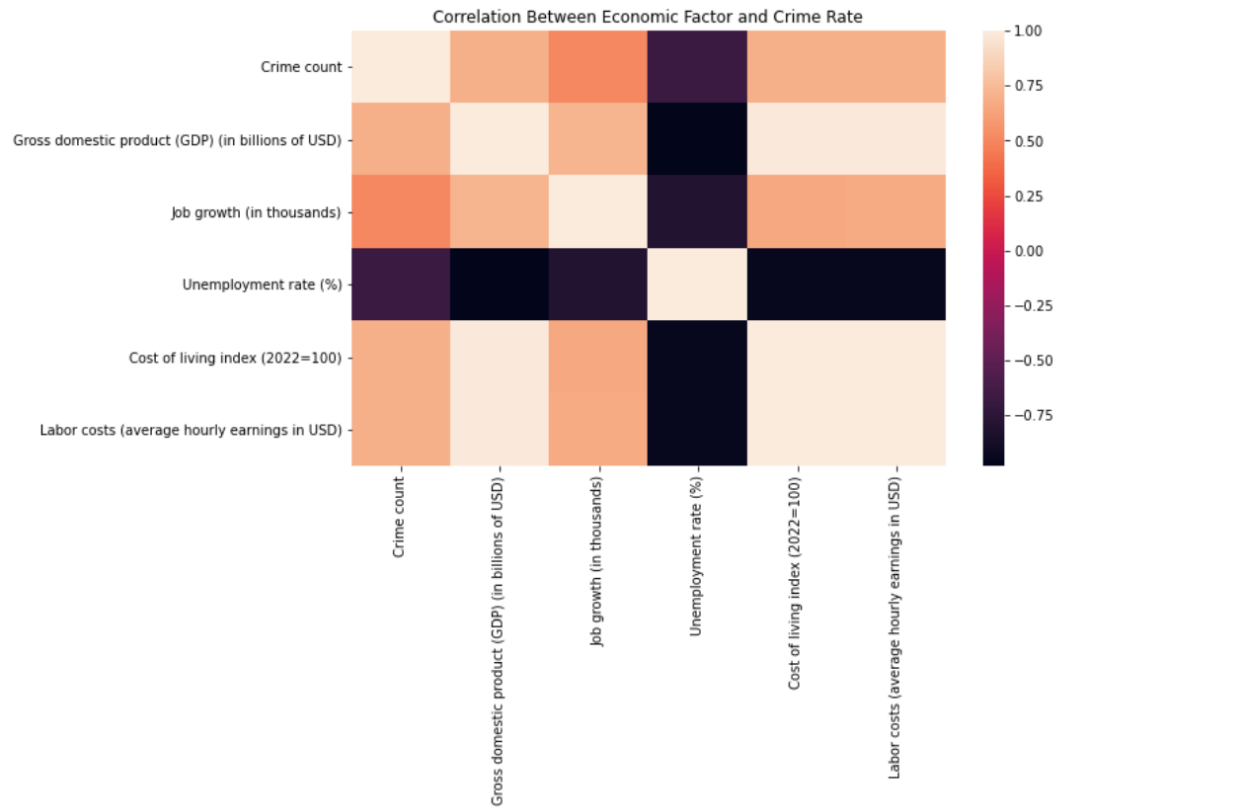


Analysis:
*The **central region** has the most number of total crimes recorded from the bar plot .*

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.

```
# Task -4.5

crm_dt = crime_data.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_Ecnmy_ds = pd.read_csv('Losangels Economic Dataset.csv')
LA_Ecnmy_ds['Crime count']=crm_dt['DR_NO'].reset_index()['DR_NO']

# print(LA_Ecnmy_ds)

correlation=LA_Ecnmy_ds[['Crime count','Gross domestic product (GDP) (in billions of USD)','Job growth (in thousands)','Unemployment rate (%)','Cost of living
plt.figure(figsize=(10, 6))
sns.heatmap(data=correlation)
plt.title('Correlation Between Economic Factor and Crime Rate')
plt.show()
```

Result:



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:

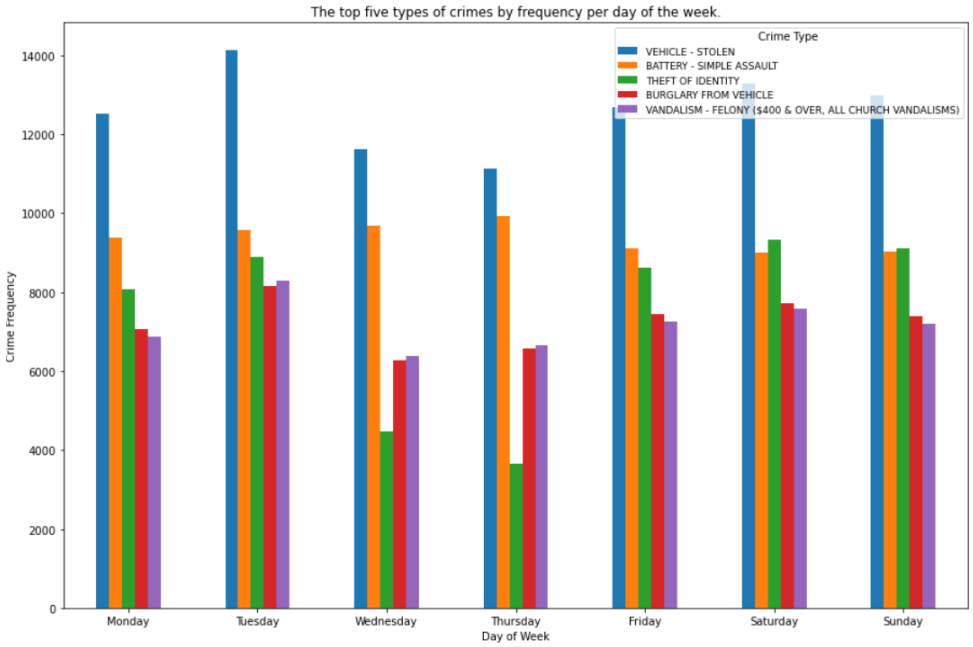
Day of the Week Extraction: We initiated the analysis by extracting the day of the week from the "Date Rptd" column. This allowed us to categorize each reported crime based on the day of the week.

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.

```
# Task 4.6

# Extract the day of the week from 'Date Rptd'
crime_data['Day_of_Week'] = pd.to_datetime(crime_data['Date Rptd']).dt.day_name()

# Group the data by 'Day_of_Week' and 'Crm Cd Desc' and count the occurrences
crime_type_day_of_week = crime_data.groupby(['Day_of_Week', 'Crm Cd Desc']).size().unstack(fill_value=0)

# Get the top N crime types
top_N = 5
top_crime_types = crime_data['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.legend(fontsize='small')
plt.xticks(range(0, 7), ['Monday', 'Tuesday', 'Wednesday', 'Thursday', 'Friday', 'Saturday', 'Sunday'], rotation=0)
plt.show()
```

Result:



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.

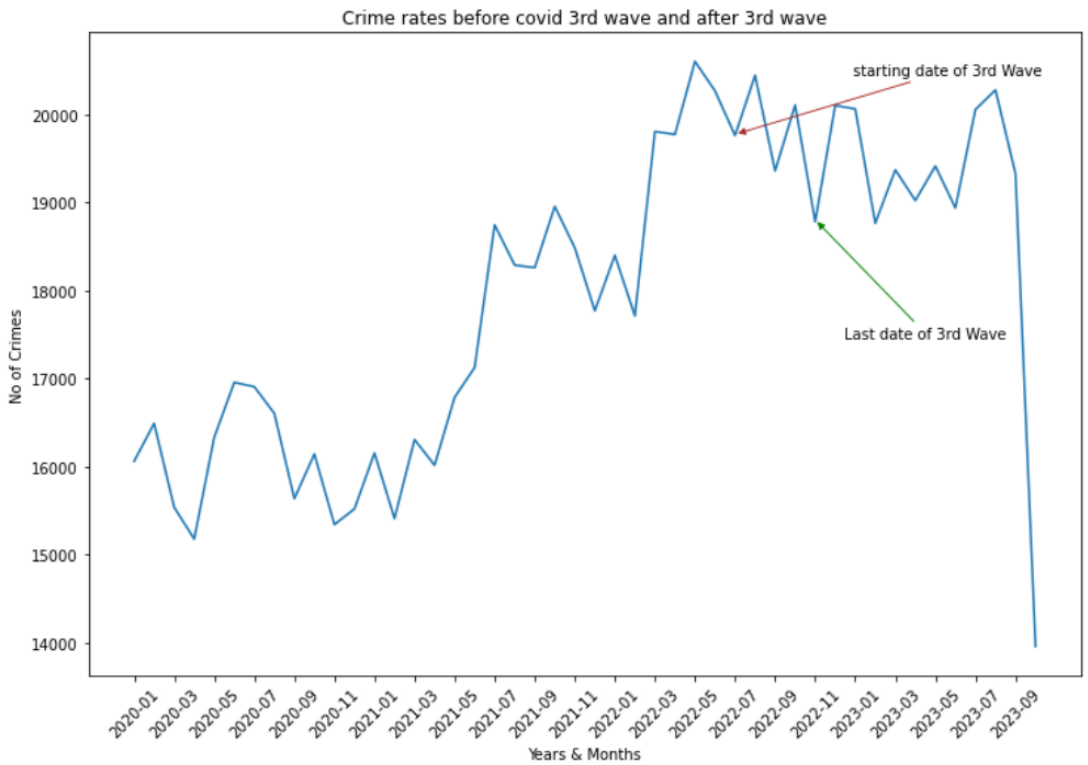
Monthly Crime Counts: Next, we counted the number of reported crimes for each month over multiple years. This allowed us to observe the temporal patterns and trends in crime rates.

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.

```
# Task 4.7

crime_data['Date Rptd'] = pd.to_datetime(crime_data['Date Rptd'])

month_count = crime_data.groupby([crime_data['Date Rptd'].dt.year, crime_data['Date Rptd'].dt.month]).size()

plt.figure(figsize=(12,8))
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)

strt_3rd_wave_date = '2022-07'
strt_label = 'starting date of 3rd Wave'

last_3rd_wave_date = '2022-11'
last_label = 'Last date of 3rd Wave'

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)
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)


plt.xlabel('Years & Months')
plt.ylabel('No of Crimes')
plt.title('Crime rates before covid 3rd wave and after 3rd wave')
```

Result:



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.*

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.

```
# Set 'DATE OCC' column as the index
#crime_data.set_index('DATE OCC', inplace=True)

# Sort the DataFrame by the index (date)
crime_data.sort_index(inplace=True)

# Check the data
#print(crime_data.head())

# Resample data to monthly frequency and count the number of crimes
crime_monthly = crime_data.resample('M').size()

# Check the resampled data
#print(crime_monthly.head())
```

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.

```
from statsmodels.tsa.arima.model import ARIMA

# Fit an ARIMA(1,1,1) model to the crime data
model = ARIMA(crime_monthly, order=(1,1,1))
results = model.fit()

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

Result:

```
                               SARIMAX Results
==============================================================================
Dep. Variable:                      y   No. Observations:                   46
Model:                 ARIMA(1, 1, 1)   Log Likelihood                -383.903
Date:                Thu, 02 Nov 2023   AIC                            773.806
Time:                        15:15:07   BIC                            779.226
Sample:                    01-31-2020   HQIC                           775.827
                         - 10-31-2023
Covariance Type:                  opg
==============================================================================
                 coef    std err          z      P>|z|      [0.025      0.975]
------------------------------------------------------------------------------
ar.L1          0.4003      1.054      0.380      0.704      -1.666       2.467
ma.L1         -0.5393      0.947     -0.569      0.569      -2.396       1.317
sigma2      1.562e+06    2.3e+05      6.783      0.000    1.11e+06    2.01e+06
===================================================================================
Ljung-Box (L1) (Q):                   0.01   Jarque-Bera (JB):               344.81
Prob(Q):                              0.92   Prob(JB):                         0.00
Heteroskedasticity (H):               6.29   Skew:                            -2.75
Prob(H) (two-sided):                  0.00   Kurtosis:                        15.39
===================================================================================
```

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.

```python
# Forecast the next 12 months (1 year)
forecast_steps = 12
forecast = results.get_forecast(steps=forecast_steps)

# Get the forecasted values and their confidence intervals
forecast_values = forecast.predicted_mean
confidence_intervals = forecast.conf_int()

# Print the forecasted values and confidence intervals
print("Forecasted Crime Counts:")
print(forecast_values)
print("\nConfidence Intervals:")
print(confidence_intervals)
```

Results:

```
Forecasted Crime Counts:
2023-11-30    13031.228128
2023-12-31    13404.816382
2024-01-31    13554.370579
2024-02-29    13614.239854
2024-03-31    13638.206617
2024-04-30    13647.800950
2024-05-31    13651.641737
2024-06-30    13653.179274
2024-07-31    13653.794778
2024-08-31    13654.041175
2024-09-30    13654.139813
2024-10-31    13654.179299
Freq: M, Name: predicted_mean, dtype: float64

Confidence Intervals:
                lower y       upper y
2023-11-30  10581.533990  15480.922266
2023-12-31  10172.219514  16637.413249
2024-01-31   9767.295694  17341.445464
2024-02-29   9369.024017  17859.455690
2024-03-31   8988.642395  18287.770840
2024-04-30   8629.639350  18665.962550
2024-05-31   8291.395480  19011.887994
2024-06-30   7971.871768  19334.486780
2024-07-31   7668.800390  19638.789165
2024-08-31   7380.109769  19927.972581
2024-09-30   7104.030320  20204.249305
2024-10-31   6839.084437  20469.274161
```

***Crime Data Visualization and Forecasting*:**

The provided code generates a visual representation of crime data using a line plot. It includes:

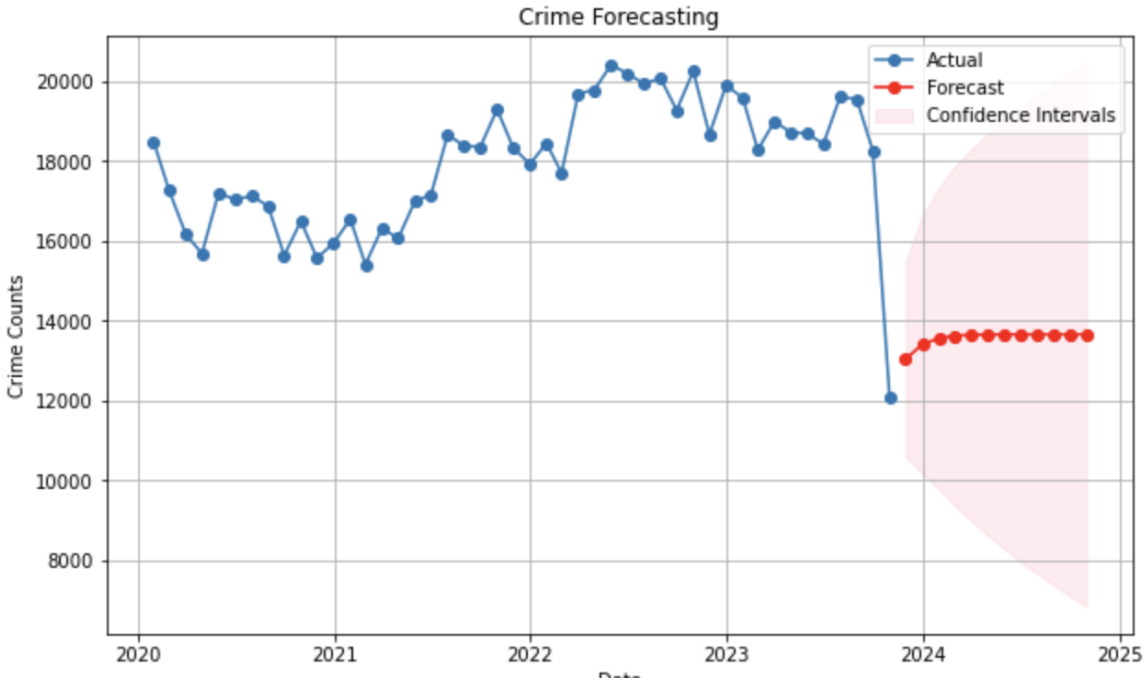
Actual Data: Displayed in blue dots, representing observed crime counts over time.

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.

```python
# Plot the actual (observed) data
plt.figure(figsize=(10, 6))
plt.plot(crime_monthly.index, crime_monthly.values, label='Actual', marker='o')

# Plot the forecasted values
plt.plot(forecast_values.index, forecast_values.values, color='red', label='Forecast', marker='o')

# Plot the confidence intervals
plt.fill_between(confidence_intervals.index, confidence_intervals.iloc[:, 0], confidence_intervals.iloc[:, 1], color='pink', alpha=0.3, label='Confidence Intervals')

# Customize the plot
plt.title('Crime Forecasting')
plt.xlabel('Date')
plt.ylabel('Crime Counts')
plt.legend()
plt.grid(True)

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

Results:



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.

```
# Calculate summary statistics for age and gender
print(crime_data[['Vict Age']].describe())
print("\n")
print(crime_data[['Vict Sex']].describe())
print("\n")

# Count the occurrences of each crime type
crime_counts = crime_data['Crm Cd Desc'].value_counts()
print(crime_counts)

top_crime_counts = crime_counts[:5]
```

Result:

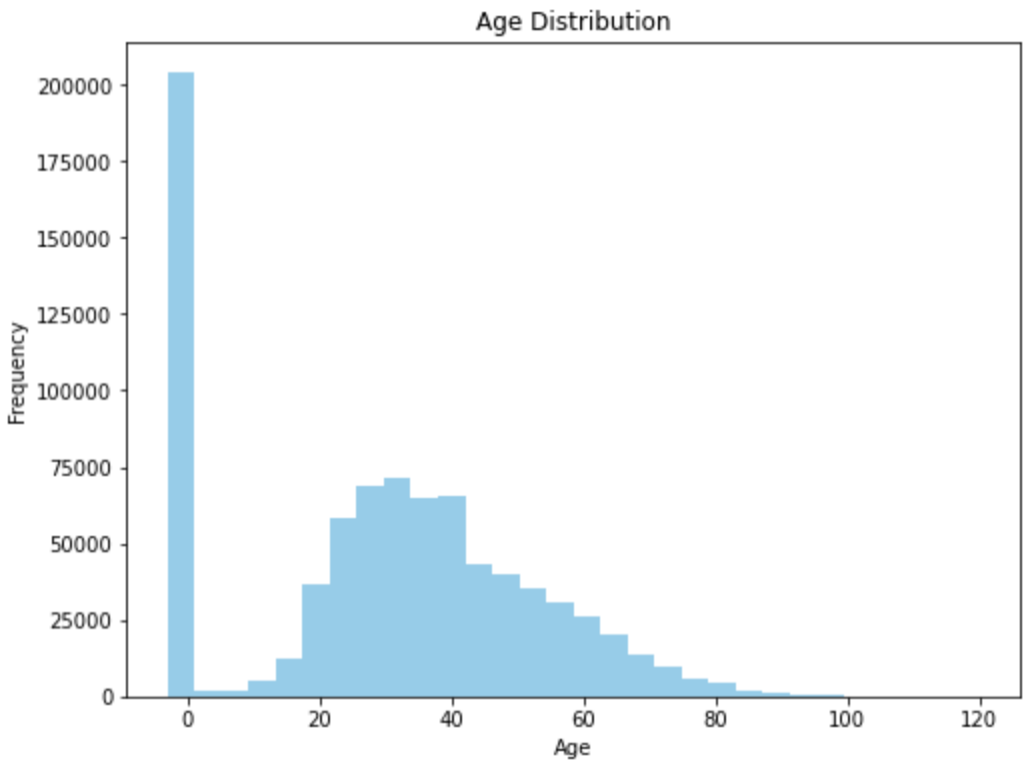
```
            Vict Age
count  825212.000000
mean       29.797181
std        21.777954
min        -3.000000
25%         7.000000
50%        31.000000
75%        45.000000
max       120.000000


       Vict Sex
count    825212
unique        5
top           M
freq     340664


VEHICLE - STOLEN                                          88355
BATTERY - SIMPLE ASSAULT                                  65728
THEFT OF IDENTITY                                         52136
BURGLARY FROM VEHICLE                                     50616
VANDALISM - FELONY ($400 & OVER, ALL CHURCH VANDALISMS)   50274
                                                          ...
FIREARMS EMERGENCY PROTECTIVE ORDER (FIREARMS EPO)            5
FIREARMS RESTRAINING ORDER (FIREARMS RO)                      4
FAILURE TO DISPERSE                                           3
DISHONEST EMPLOYEE ATTEMPTED THEFT                            2
INCITING A RIOT                                               1
Name: Crm Cd Desc, Length: 138, dtype: int64
```

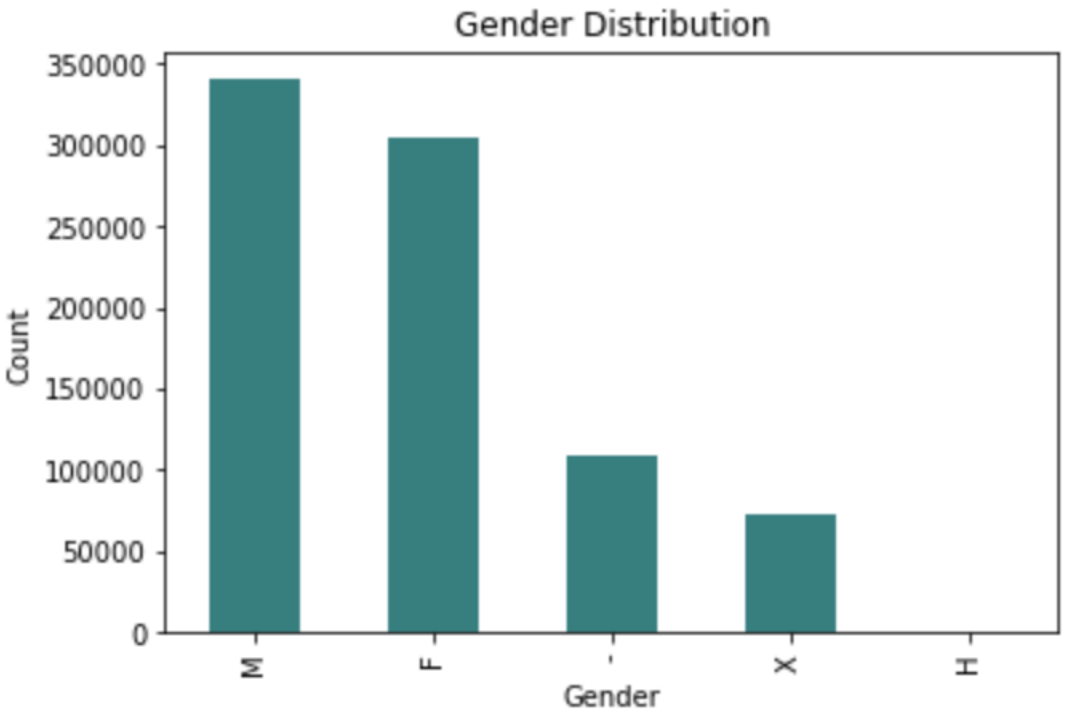
Analysis:
From the initial analyses we can observe that the ***average age for the criminal is almost 30 years*** *and the* ***Male Gender has committed most of the crimes with a frequency of 340,664.***

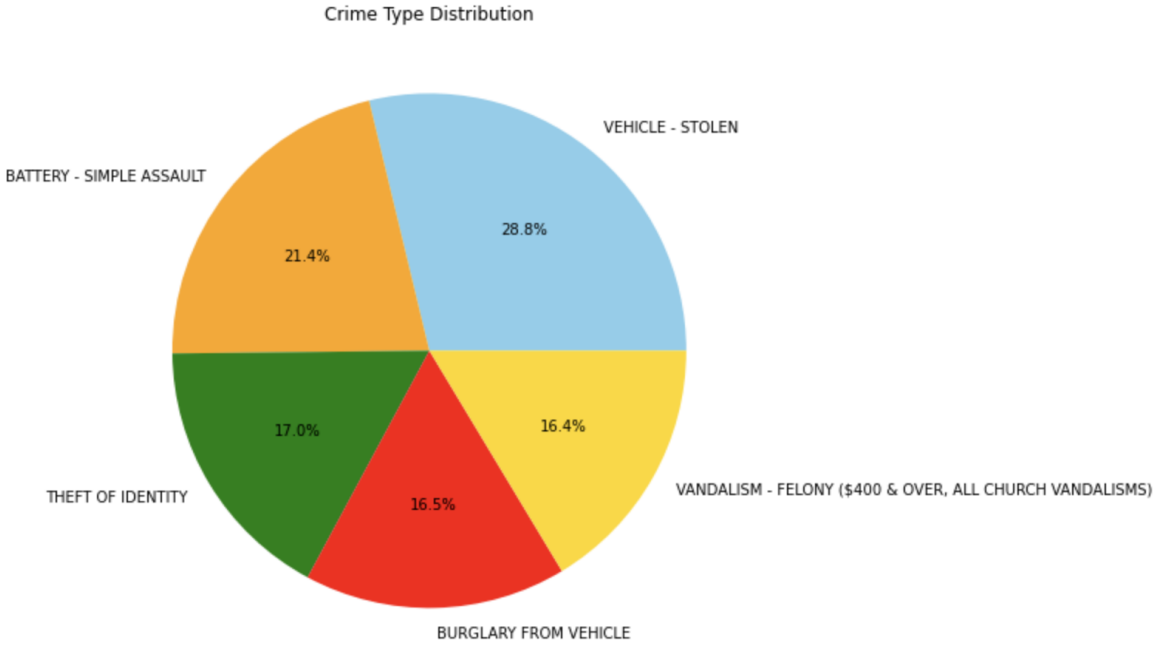
Distribution based on Age, Gender and Crime Type:
Age Distribution Histogram: The histogram illustrates the distribution of victim ages. It provides a visual representation of the frequency of different age groups within the dataset. The x-axis represents age, the y-axis represents frequency, and the 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.

```
# Histogram for age distribution
plt.figure(figsize=(8, 6))
plt.hist(crime_data['Vict Age'], bins=30, color='skyblue')
plt.title('Age Distribution')
plt.xlabel('Age')
plt.ylabel('Frequency')
plt.show()

# Bar chart for gender distribution
plt.figure(figsize=(6, 4))
crime_data['Vict Sex'].value_counts().plot(kind='bar', color='teal')
plt.title('Gender Distribution')
plt.xlabel('Gender')
plt.ylabel('Count')
plt.show()

# Pie chart for Top crime types
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()
```

Result:







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.

Conclusion

The exploratory data analysis (EDA) of the crime data from 2020 to present in Los Angeles has provided valuable insights into various aspects of criminal activities in the city. We observed distinct temporal patterns, with a notable increase in certain crime types during specific periods. The application of the ARIMA time series model allowed for forecasting future crime trends, offering a valuable tool for proactive law enforcement strategies. The generated forecasts and confidence intervals provide a level of certainty to guide decision-making.

Overall, 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.