KINGSTON UNIVERSITY LONDON

APPLIED DATA PROGRAMMING CI7340

COURSEWORK 1

TITLE: CRIME DATA ANALYSIS AND VISUALIZATION

GROUP 18

NAME		KU NUMBER
Karthikraj Meda Nagaraja	-	K2335162
Pratiksha Rajendran	-	K2203709
Sujay Grama Suresh Kumar	-	K2201621
Vishnuvardhan Gorantla	-	K2259314

Module Leader: Dr Farzana Rahman

GROUP MEMBERS AND PARTICIPATION

Student Name	Student ID	Group Assignment Development Contribution Percentage (%)	Optional (Mention Contribution Part/ Section)
Karthikraj Meda Nagaraja	K2335162	25%	IDA, EDA, Statistics, Visualizations & Report Forming
Pratiksha Rajendran	K2203709	25%	IDA, EDA, Statistics, Visualizations & Report Forming
Sujay Grama Suresh Kumar	K2201621	25%	IDA, EDA, Statistics, Visualizations & Report Forming
Vishnuvardhan Gorantla	K2259314	25%	IDA, EDA, Statistics, Visualizations & Report Forming

ABSTRACT

Analyzing crime statistics in Montgomery County, USA involves utilizing a comprehensive dataset derived from the National Incident-Based Reporting System (NIBRS) within the Criminal Justice Information Services (CJIS) Division Uniform Crime Reporting (UCR) Program. Covering the years 2018 to 2022, the dataset provides a detailed overview of reported crimes documented by the Montgomery County Police Department. The study aims to assess the dataset's quality, propose improvements for missing values, and employ various data-wrangling techniques. Exploratory Data Analysis (EDA) techniques, including descriptive statistics and data visualization, will be employed to address ten research questions and formulate insights into crime patterns. The report emphasizes the importance of this analysis in supporting local and national efforts to minimize specific types of crime and predict future patterns.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF ABBREVIATIONS

Abbreviation	Explanation
NIBRS	National Incident-Based Reporting System
CJIS	Criminal Justice Information Services
UCR	Uniform Crime Reporting
EDA	Exploratory Data Analysis

CHAPTER 1 INTRODUCTION

Embarking on an exploration of Montgomery County's crime statistics opens a window into the intricacies of reported incidents spanning from 2018 to 2022. The dataset, a reflection of real-world occurrences, serves as a valuable resource for uncovering patterns and trends(*COPS, 2018).* The endeavor involves delving into the nuances of the data, addressing gaps, and formulating insightful questions. Beyond the numbers, the workflow seeks to extract meaningful narratives, offering a glimpse into the dynamics of crime in the county.

1.1 PROBLEM STATEMENT

This investigation aims to decipher nuanced crime patterns in Montgomery County, providing actionable insights for evidence-based policy making. The overarching problem is to understand crime typology and frequency to effectively reduce crime rates.

1.2 PROPOSED APPROACH

The methodology involves data quality assessment, data cleaning, exploratory data analysis and advanced visualization techniques. Leveraging key columns, insights will be extracted to formulate strategies addressing and predicting crime patterns.

1.3 RESEARCH QUESTIONS

The research aims to:

1. Identify predominant crime forms.
2. Understand the evolution of crime rates from 2018 to 2022.
3. Explore temporal patterns in crime occurrences.
4. Investigate inter-correlations among crime categories.
5. Assess the efficacy of law enforcement measures.
6. Identify geographical hotspots for specific crimes.
7. Examine demographic correlations with crime rates.
8. Analyze seasonal fluctuations in crime rates.
9. Understand crime rate variations across different days.
10. Develop effective visualizations for forecasting crime patterns.

1.4 OBJECTIVES

1.4.1 Comprehensive Understanding:

- Achieve a profound understanding of the Montgomery County crime dataset, encompassing its source, structure, and the types of crimes reported.

1.4.2 Identify Data Quality Issues:

- Assess the quality of the dataset, identifying areas with missing values, data cleanliness concerns, and potential inconsistencies.

1.4.3 Data Cleaning and Improvement:

- Implement effective data cleaning strategies to address missing values and enhance the overall quality of the dataset. Explore various techniques using Python and its libraries to handle missing values, ensuring a robust foundation for analysis.

1.4.4 Develop impactful visualizations:

- Through visually compelling representations, aim to provide valuable insights accessible to both technical and non-technical audiences.

1.5 METHODOLOGY

1. **Preliminary Data Analysis and Exploration:**
   Conduct an in-depth analysis for insights.
2. **Data Cleaning and Transformation:**
   Address data quality issues using Python libraries.
3. **Exploratory Data Analysis:**
   Uncover patterns and trends through EDA techniques.
4. **Descriptive Statistics:**
   Apply statistical methods to gain insights.
5. **Data Visualization:**
   Use diverse techniques for clear representation.

1.6 EXPECTED OUTCOMES

1. **In-depth Understanding of Crime Patterns:**
   Attain a nuanced comprehension through rigorous analysis.
2. **Enhanced Dataset Quality:**
   Improve reliability by addressing missing values.
3. **Visual Representation of Findings:**
   Present results through compelling visualizations.
4. **Actionable Insights:**
   Provide recommendations for effective strategies based on identified patterns.

1.7 WORKFLOW

The structured workflow in

Figure 1 outlines the methodological journey.

Figure 1 Workflow

CHAPTER 2 DATASET

The dataset, focusing on crime statistics in Montgomery County, USA, originates from the NIBRS within the CJIS Division UCR Program(*Healthy People 2030, 2016).* Covering 2018 to 2022, it serves as a detailed repository of reported crimes by the Montgomery County Police Department(*Montgomery County, Maryland, n.d.).*

2.1 SOURCE AND FORMAT

The dataset adheres to NIBRS standards and comes in a structured CSV format, facilitating systematic exploration. Imported into the Jupyter notebook IDE, it comprises 306,094 entries and 30 variables. A df.info() examination revealed detailed column information shown in Figure 2.

```
RangeIndex: 306094 entries, 0 to 306093
Data columns (total 30 columns):
 #   Column                 Non-Null Count   Dtype
---  ------                 --------------   -----
 0   Incident ID            306094 non-null  int64
 1   Offence Code           306094 non-null  object
 2   CR Number              306094 non-null  int64
 3   Dispatch Date / Time   257065 non-null  object
 4   NIBRS Code             306094 non-null  object
 5   Victims                306094 non-null  int64
 6   Crime Name1            305822 non-null  object
 7   Crime Name2            305822 non-null  object
 8   Crime Name3            305822 non-null  object
 9   Police District Name   306000 non-null  object
 10  Block Address          279888 non-null  object
 11  City                   304818 non-null  object
 12  State                  306094 non-null  object
 13  Zip Code               302915 non-null  float64
 14  Agency                 306094 non-null  object
 15  Place                  306094 non-null  object
 16  Sector                 304564 non-null  object
 17  Beat                   304564 non-null  object
 18  PRA                    305855 non-null  object
 19  Address Number         279985 non-null  float64
 20  Street Prefix          13631 non-null   object
 21  Street Name            306093 non-null  object
 22  Street Suffix          5432 non-null    object
 23  Street Type            305755 non-null  object
 24  Start_Date_Time        306094 non-null  object
 25  End_Date_Time          144436 non-null  object
 26  Latitude               306094 non-null  float64
 27  Longitude              306094 non-null  float64
 28  Police District Number 306094 non-null  object
 29  Location               306094 non-null  object
dtypes: float64(4), int64(3), object(23)
```

Figure 2 Column information

DESCRIPTION OF DATA

With diverse data types, each column contributes uniquely to understanding crime statistics intricacies as shown in Table 1.

CATEGORY	COLUMN NAME
Crime Details	• Crime Name1 • Crime Name2 • Crime Name3
Geographical Details	• Police District Name • Block Address • City • State • Zip Code
Law Enforcement Details	• Agency • Place • Sector • Beat • PRA
Temporal and Location Coordinates	• Start_Date_Time, End_Date_Time • Latitude, Longitude • Police District Number • Location

Table 1 Column Names as per Category

This structured dataset, with multidimensional attributes, forms the basis for an in-depth exploration of crime patterns in Montgomery County.

CHAPTER 3 DATA QUALITY ASSESSEMENT

Upon analyzing the dataset, several noteworthy issues were identified, prompting the need for comprehensive data quality improvements(*Tavares, E., III., n.d)* are summarized in Table 2.

ISSUE	PROPOSED SOLUTION
Significant Null/Missing Values	Addressed through strategic measures; for instance, in cases like Longitude/Latitude, mean values specific to the relevant Zip Code will replace zero or outlier values.
Features with Unexpected Data Types	With the aid of Python data type conversions will be performed as needed, aligning features with their appropriate data types.
Incorrect Entries and Data Anomalies	Remedial actions include resolving discrepancies such as multiple city spellings, unrecorded dispatch/end times, multiple states despite being in one, and outlying Zip Codes. This ensures data accuracy and consistency.

Table 2 Issues with dataset and proposed solution

The actions undertaken to cleanse and transform the data are delineated in Table 3.

COLUMN NAME	DATA TYPE	DROP	ACTION
Latitude	float64	No	Computation is done for the mean latitude and longitude values for each Zip Code by groupby and mean functions.
Longitude	float64	No	Computation is done for the mean latitude and longitude values for each Zip Code by groupby and mean functions.
Zip Code	float64	No	Utilizing the mode function on street names replaced blank values
City	object	No	Corrected misspelled city names using a dictionary
State	object	No	Elaborated on the abbreviations to enhance clarity and comprehension
Street Name	object	No	Rectified inaccuracies in street names by using a dictionary for precise and uniform correction.
Crime Name 1	object	No	Addressed null values by relying on information from the NIBRS Code column.
Crime Name 2	object	No	Addressed null values by relying on information from the NIBRS Code column.

Crime Name 3	object	No	Addressed null values by relying on information from the NIBRS Code column.
Start_Date_Time	object	No	Converted to date_time in Python
Police District Number	object	No	Corrected inaccurate information by substituting it with accurate data
Police District Name	object	No	Corrected inconsistencies in Police District Name after correcting District Number
Street Suffix	object	Yes	Unnecessary for the analytical focus; removed for simplicity
Street Prefix	object	Yes	Unnecessary for the analytical focus; removed for simplicity
Block Address	object	Yes	The information is already available in the Address column, so it was removed.
Location	object	Yes	The combination of Latitude and Longitude, increases memory usage

Table 3 Actions undertaken to improve data quality

3.1 APPROACHES TO HANDLE MISSING DATA, PREPROCESSING, AND TRANSFORMATION

3.1.1 Data Filtering

- Records from 2017 were excluded as a part of research resulting in a dataset spanning from 2018 to 2022, increasing analysis relevance.
- A new column, "Start Year," was created for potential visualizations shown in Figure 3.

```
df['Start_Date_Time'] = pd.to_datetime(df['Start_Date_Time'])

df['Start_Year'] = (df['Start_Date_Time'].dt.year)

df_1 = df[(df['Start_Year'] >=2018)]
```

Figure 3 Data Filtering (2018-2022)

3.1.2 Handling Missing Geospatial Information

- Blank values in latitude and longitude were systematically addressed.
- Mean latitude and longitude values for each Zip Code were computed using groupby and mean functions.

- Missing values in latitude and longitude were substituted with their respective Zip Code means as shown in Figure 4.

```
# Merge the calculated value with zip code and suffix with name '_mean'

df_2 = df_1.merge(lat_lon_mean, left_on = 'Zip Code', right_index= True, suffixes = ('','_mean'), how = 'left')
```

```
# Using numpy replace all the 0 in Lat and Long with 'NaN' for easy replacement

df_2['Latitude'].replace(0,np.nan,inplace = True)
df_2['Longitude'].replace(0,np.nan,inplace = True)
```

```
df_2['Latitude'].fillna(df_2['Latitude_mean'], inplace = True)
df_2['Longitude'].fillna(df_2['Longitude_mean'],inplace = True)
```

```
df_2.drop(['Latitude_mean','Longitude_mean'], axis = 1, inplace = True)
```

```
#Before repacing 0 with mean value

(df_1['Latitude'] == 0).sum()
```

```
7085
```

```
#After repacing 0 with mean value

(df_2['Latitude'] == 0).sum()
```

```
21
```

Figure 4 Handling Missing Geospatial Information - Code Implementation

3.1.3 Refining Latitude and Longitude Data

- Despite the initial replacement, some latitude and longitude entries persisted as zeros.
- Employing the mode function, the remaining zero values were further rectified by substituting them with the most prevalent values for their respective Zip Codes.

```
df_2[(df_2['Latitude'] == 0) & (df_2['Longitude'] == 0)][['Zip Code', 'Latitude', 'Longitude']]
```

	Zip Code	Latitude	Longitude
2632	20007.0	0.0	0.0
24255	20010.0	0.0	0.0
25279	20010.0	0.0	0.0
26593	20002.0	0.0	0.0
30259	20743.0	0.0	0.0
34265	20737.0	0.0	0.0
36943	20712.0	0.0	0.0
43967	20007.0	0.0	0.0
65142	21740.0	0.0	0.0
84925	20016.0	0.0	0.0
96594	22181.0	0.0	0.0
97852	20032.0	0.0	0.0
98476	21794.0	0.0	0.0
101093	22102.0	0.0	0.0
102571	20004.0	0.0	0.0
107152	20010.0	0.0	0.0
186303	20009.0	0.0	0.0
190279	20706.0	0.0	0.0
193068	20016.0	0.0	0.0

Figure 5 Display of '0' Values Unfit for Mean Replacement

```
# Using mode function replacing the remaining values having latitude & longitude as '0'

crime_data4.loc[crime_data5['Street Name'] == 'KAYWOOD']['Latitude'].mode()
```

```
0     0.000000
1    39.096564
2    39.103965
Name: Latitude, dtype: float64
```

Figure 6 Code Snippet Utilizing Mode Function for Remaining '0' Values

- Regarding null values in the "Zip Code", instances were identified where street names lacked assigned Zip Codes.
- The numpy.where() method was employed to substitute null values with mode of the street names, ensuring accurate Zip Codes for all street names within the same vicinity as shown in Figure 7.

```
crime_data4['Zip Code'] = np.where((crime_data4['Zip Code'].isna())&(crime_data4['Street Name'] == 'FLOWER HILL'), crime_dat
crime_data4['Zip Code'] = np.where((crime_data4['Zip Code'].isna())&(crime_data4['Street Name'] == 'CLARIDGE'), crime_data4.
crime_data4['Zip Code'] = np.where((crime_data4['Zip Code'].isna())&(crime_data4['Street Name'] == 'GATE'), crime_data4.loc[
crime_data4['Zip Code'] = np.where((crime_data4['Zip Code'].isna())&(crime_data4['Street Name'] == 'PARKLAND'), crime_data4.
crime_data4['Zip Code'] = np.where((crime_data4['Zip Code'].isna())&(crime_data4['Street Name'] == 'GRANADA'), crime_data4.l
crime_data4['Zip Code'] = np.where((crime_data4['Zip Code'].isna())&(crime_data4['Street Name'] == 'TYGART'), crime_data4.lo
crime_data4['Zip Code'] = np.where((crime_data4['Zip Code'].isna())&(crime_data4['Street Name'] == 'ATHERTON'), crime_data4.
crime_data4['Zip Code'] = np.where((crime_data4['Zip Code'].isna())&(crime_data4['Street Name'] == 'COLONY'), crime_data4.lo
crime_data4['Zip Code'] = np.where((crime_data4['Zip Code'].isna())&(crime_data4['Street Name'] == 'NORTHAMPTON'), crime_dat
crime_data4['Zip Code'] = np.where((crime_data4['Zip Code'].isna())&(crime_data4['Street Name'] == 'ROCKVILLE'), crime_data4
crime_data4['Zip Code'] = np.where((crime_data4['Zip Code'].isna())&(crime_data4['Street Name'] == 'DALE'), crime_data4.loc[
crime_data4['Zip Code'] = np.where((crime_data4['Zip Code'].isna())&(crime_data4['Street Name'] == 'TRAVILAH CREST'), crime_
crime_data4['Zip Code'] = np.where((crime_data4['Zip Code'].isna())&(crime_data4['Street Name'] == 'FLOWER'), crime_data4.lo
```

Figure 7 Substituting Null Zip Codes with Mode of Zip Code within respective Street Names

3.1.4 Standardization of City Name, State, and Street Name

The use of dictionaries to standardize entries in the City, State, and Street Name columns has proven effective in ensuring consistency throughout the dataset.

3.1.4.1 City Column

- City column meticulously corrected using the replace method and dictionary, ensuring precise city name spellings shown in Figure 8.

```
city_mapping = {
    'BETHEDA': 'BETHESDA',
    'CLAEKSBURG': 'CLARKSBURG',
    'FREDRICK': 'FREDERICK',
    'GAIHTERSBURG': 'GAITHERSBURG',
    'GAITHERSBUG': 'GAITHERSBURG',
    'ROCKVIILE': 'ROCKVILLE',
    'ROCKVILLLE': 'ROCKVILLE',
    'SILVERS SPRING': 'SILVER SPRING',
    'TACOMA PARK': 'TAKOMA PARK',
    'MT AIRY': 'MOUNT AIRY'
}
crime_data1['City'].replace(city_mapping, inplace=True)
```

Figure 8 Dictionary Usage for City Name Standardization

3.1.4.2 State Column

- Abbreviations were replaced for clarity as shown in Figure 9. State names were updated to "Maryland" for rows with codes "15" and "16," based on Latitude and Longitude.
- Rows with code "0" were removed due to nulls in critical fields (Latitude, Longitude, Crime Names).
- A row with code "17" was substituted, aligning state with "Maryland" based on specific data.

```
state_mapping = {
    'MD': 'Maryland',
    'VA': 'Virginia',
    'ME': 'Maine',
    'MS': 'Mississippi',
    'ND': 'North Dakota',
    'DC': 'Washington, D.C.'
}

crime_data1['State'] = crime_data1['State'].replace(state_mapping)
```

Figure 9 Dictionary Usage for State Name Standardization

3.1.4.3 Street Name Column

- Spelling inconsistencies were rectified for the Street names utilizing a dictionary, ensuring uniformity shown in Figure 10.

```
street_mapping = {
    'ZANHORN': 'VAN HORN'
}

crime_data4['Street Name'] = crime_data4['Street Name'].replace(street_mapping)
```

Figure 10 Dictionary Usage for Street Name Standardization

3.1.5 Handling NIBRS Code and Crime Name1, Crime Name2, Crime Name 3

- A comprehensive strategy was implemented to address missing values in the NIBRS code(*UCR Program, n.d)* and related Crime Name columns.
- Initial examination revealed null values in three NIBRS codes, notably, NIBRS code 26G had null values across all Crime Name columns.
- To resolve this, specific definitions for Crime Name1, 2, and 3 related to computer-related offenses were derived and used to replace null values, as illustrated in Figure 11.

```
#Crime name 1,2 & 3 columns with the NIBRS code 26G were all null, by using an online reference we could derive all the crimename
crime_data5['Crime Name1'] = np.where(crime_data5['NIBRS Code'] == '26G', 'Crime Against Property', crime_data5['Crime Name1'])
crime_data5['Crime Name2'] = np.where(crime_data5['NIBRS Code'] == '26G', 'Fraud Offenses', crime_data5['Crime Name2'])
crime_data5['Crime Name3'] = np.where(crime_data5['NIBRS Code'] == '26G', 'HACKING/COMPUTER INVASION', crime_data5['Crime Name3'])
```

Figure 11 Replacing null values in Crime Name 1, Crime Name 2, Crime Name 3 with NIBRS Code as 26G

- For other NIBRS codes with some NaN values, a mode-based approach was employed.

- Utilizing the computed mode, all NaN values within Crime Name1, 2, and 3 columns were systematically replaced, as depicted in Figure 12 and Figure 13.

```
crime_data5['Crime Name1'] = np.where((crime_data5['NIBRS Code'] == '90Z')&(crime_data5['Crime Name1'].isna()), crime_data5.loc[
crime_data5['Crime Name2'] = np.where((crime_data5['NIBRS Code'] == '90Z')&(crime_data5['Crime Name2'].isna()), crime_data5.loc[
crime_data5['Crime Name3'] = np.where((crime_data5['NIBRS Code'] == '90Z')&(crime_data5['Crime Name3'].isna()), crime_data5.loc[
```

Figure 12 Replacing NaN values in Crime Name 1, Crime Name 2, Crime Name 3 with NIBRS Code as 90Z

```
crime_data5['Crime Name1'] = np.where((crime_data5['NIBRS Code'] == '35A')&(crime_data5['Crime Name1'].isna()), crime_data5.loc[
crime_data5['Crime Name2'] = np.where((crime_data5['NIBRS Code'] == '35A')&(crime_data5['Crime Name2'].isna()), crime_data5.loc[
crime_data5['Crime Name3'] = np.where((crime_data5['NIBRS Code'] == '35A')&(crime_data5['Crime Name3'].isna()), crime_data5.loc[
```

Figure 13 Replacing NaN values in Crime Name 1, Crime Name 2, Crime Name 3 with NIBRS Code as 35A

3.1.6 Handling Police District Number and Police District Name

- Value count of Police District Number showed inconsistencies, e.g., "8.0D" equal to "8D," totaling 94 incorrect rows (*Bowser, M. ,2015*).
- Incorrect entries were replaced using .loc property, shown in Figure 14.

```
crime_data1.loc[crime_data["Police District Number"] == "8.0D", "Police District Number"] = "8D"
crime_data1.loc[crime_data["Police District Number"] == "3.0D", "Police District Number"] = "3D"
crime_data1.loc[crime_data["Police District Number"] == "6.0D", "Police District Number"] = "6D"
crime_data1.loc[crime_data["Police District Number"] == "5.0D", "Police District Number"] = "5D"
crime_data1.loc[crime_data["Police District Number"] == "1.0D", "Police District Number"] = "1D"
crime_data1.loc[crime_data["Police District Number"] == "4.0D", "Police District Number"] = "4D"
```

Figure 14 Replacing ".0D" Police District Number with "*D" Police District Number*

- Attention shifted to Police District Name, linked with Number.
- Disparities were addressed, and nulls in the Name were replaced with correct values using np.where().

```
crime_data1['Police District Name'] = np.where(crime_data1['Police District Number'] == '8D', 'CITY OF TAKOMA PARK', crime_data1['Police District Name'])
crime_data1['Police District Name'] = np.where(crime_data1['Police District Number'] == '6D', 'MONTGOMERY VILLAGE', crime_data1['Police District Name'])
crime_data1['Police District Name'] = np.where(crime_data1['Police District Number'] == '5D', 'GERMANTOWN', crime_data1['Police District Name'])
crime_data1['Police District Name'] = np.where(crime_data1['Police District Number'] == '1D', 'ROCKVILLE', crime_data1['Police District Name'])
crime_data1['Police District Name'] = np.where(crime_data1['Police District Number'] == '4D', 'WHEATON', crime_data1['Police District Name'])
crime_data1['Police District Name'] = np.where(crime_data1['Police District Number'] == '3D', 'SILVER SPRING', crime_data1['Police District Name'])
```

Figure 15 Replacing Police District Name with Missing Values based on Police District Number

3.1.7 Further refinement of Latitude Longitude and Zip Code

- Addressed remaining zero values in latitudes and longitudes associated with specific Zip Codes.
- Systematic approach considering interrelation between Zip Codes, street names, latitudes, and longitudes.

- Utilized mode values for latitudes and longitudes based on given street name and Zip Code, as shown in Fig 16.

```
crime_data5.loc[(crime_data5['Location'] == '(0.0, 0.0)') & (crime_data5['Zip Code'] == 20007.0) & (crime_data5['Street Name
crime_data5.loc[(crime_data5['Location'] == '(0.0, 0.0)') & (crime_data5['Zip Code'] == 20010.0) & (crime_data5['Street Name
crime_data5.loc[(crime_data5['Location'] == '(0.0, 0.0)') & (crime_data5['Zip Code'] == 20002.0) & (crime_data5['Street Name
crime_data5.loc[(crime_data5['Location'] == '(0.0, 0.0)') & (crime_data5['Zip Code'] == 20743.0) & (crime_data5['Street Name
crime_data5.loc[(crime_data5['Location'] == '(0.0, 0.0)') & (crime_data5['Zip Code'] == 20737.0) & (crime_data5['Street Name
crime_data5.loc[(crime_data5['Location'] == '(0.0, 0.0)') & (crime_data5['Zip Code'] == 20712.0) & (crime_data5['Street Name
crime_data5.loc[(crime_data5['Location'] == '(0.0, 0.0)') & (crime_data5['Zip Code'] == 20007.0) & (crime_data5['Street Name
crime_data5.loc[(crime_data5['Location'] == '(0.0, 0.0)') & (crime_data5['Zip Code'] == 21740.0) & (crime_data5['Street Name
crime_data5.loc[(crime_data5['Location'] == '(0.0, 0.0)') & (crime_data5['Zip Code'] == 20016.0) & (crime_data5['Street Name
crime_data5.loc[(crime_data5['Location'] == '(0.0, 0.0)') & (crime_data5['Zip Code'] == 22181.0) & (crime_data5['Street Name
crime_data5.loc[(crime_data5['Location'] == '(0.0, 0.0)') & (crime_data5['Zip Code'] == 20032.0) & (crime_data5['Street Name
crime_data5.loc[(crime_data5['Location'] == '(0.0, 0.0)') & (crime_data5['Zip Code'] == 22102.0) & (crime_data5['Street Name
crime_data5.loc[(crime_data5['Location'] == '(0.0, 0.0)') & (crime_data5['Zip Code'] == 20009.0) & (crime_data5['Street Name
crime_data5.loc[(crime_data5['Location'] == '(0.0, 0.0)') & (crime_data5['Zip Code'] == 20016.0) & (crime_data5['Street Name
```

Figure 16 Replacing null values in Latitude and Longitude with mode values

- Review of latitudes and longitudes identified seven rows still displaying null values, indicating some Zip Codes have zero as the mode.
- These rows are subsequently dropped, and an outlier with a latitude of 90 is identified and removed for data refinement.

3.1.8 Removal of Redundant Columns

- Specific columns were removed from the dataset, as shown in Figure 17.

```
#dropping those columns which we don't need for the visualisations
crime_data6.drop(['Street Suffix', 'Street Prefix', 'Location', 'Block Address'], axis=1, inplace=True)
```

Figure 17 Dropping Unnecessary columns

- Following these steps, the crime data dataframe for Montgomery County from 2018 to 2022 comprises 221,543 rows, providing a substantial dataset for meaningful visualizations.

3.2 DATA TRANSFORMATION AND FEATURE ENGINEERING

Feature engineering is integral for extracting meaningful insights from data, and various techniques were applied in preparation for visualizations.

3.2.1 Day, Month, and Date Calculation

Utilizing the pandas dt method, the day of the week, month, and date were derived from the 'Start_Date_Time', as shown in Figure 18.

```
crime_data6['day_name'] = pd.to_datetime(crime_data6['Start_Date_Time']).dt.day_name()
crime_data6['Month'] = (crime_data6['Start_Date_Time'].dt.month)
crime_data6['Day'] = crime_data6['Start_Date_Time'].dt.day

crime_data6['Time_Hour'] = crime_data6['Start_Date_Time'].dt.hour
```

Figure 18 Deriving Day and Month from Start_Date_Time

3.2.2 Hour Extraction

The hour was extracted from the 'Start_Date_Time', providing additional details for visualization.

3.2.3 Determination of Period of the Day

Employing the 24-hour format, the dataset's hour column underwent processing to categorize periods such as morning, afternoon, evening, night, and midnight, enhancing visualization interpretability, as illustrated in Figure 19.

```
def time_def(df):
    for index, row in df.iterrows():
        if 4 < row['Time_Hour'] <= 12:
            df.loc[index, 'Time_of_the_day'] = 'Morning'
        elif 12 < row['Time_Hour'] <= 15:
            df.loc[index, 'Time_of_the_day'] = 'Afternoon'
        elif 15 < row['Time_Hour'] <= 19:
            df.loc[index, 'Time_of_the_day'] = 'Evening'
        elif 19 < row['Time_Hour'] <= 23:
            df.loc[index, 'Time_of_the_day'] = 'Night'
        elif 0 <= row['Time_Hour'] <= 4:
            df.loc[index, 'Time_of_the_day'] = 'Midnight'
    return df
```

```
crime_data6 = time_def(crime_data6)
```

Figure 19 Depicting the period of the Day

In the domain of data wrangling, these methods involve extraction (temporal data extraction), merging (geospatial data imputation), and construction of analytical datasets through imputation and standardization processes. These collectively contribute to refining the dataset for further analysis, ensuring data quality and completeness.

CHAPTER 4 DESCRIPTIVE STATISTICS

4.1 STATISTICAL METHODS

4.1.1 Basic Descriptive Statistics

- Descriptive statistics offer insights into numeric features (Incident ID, CR Number, Victims, Zip Code, Address Number, Latitude, Longitude, Start_Year, Month, Day, Time_Hour), as shown in Figure 20.
- Count indicates data completeness.
- Mean provides an average value.
- Std (standard deviation) quantifies data dispersion.
- Min and max indicate data range.
- 25%, 50%, 75% percentiles show data distribution.

```
In [3]: # Display basic descriptive statistics
        descriptive_stats = data.describe()
        descriptive_stats
```

Out[3]:

	Incident ID	CR Number	Victims	Zip Code	Address Number	Latitude	Longitude	Start_Year	Month	Day
count	2.215420e+05	2.215420e+05	221542.000000	221542.000000	2.004140e+05	221542.000000	221542.000000	221542.000000	221542.000000	221542.000000
mean	2.012778e+08	1.973929e+08	1.023752	20877.149462	8.395592e+03	39.081217	-77.115251	2019.719940	6.230268	15.526063
std	6.288905e+04	1.373460e+07	0.191766	62.993771	1.600284e+04	0.071289	0.095708	1.350194	3.397613	8.868015
min	2.011689e+08	1.202759e+07	1.000000	20001.000000	1.000000e+00	38.938410	-77.516753	2018.000000	1.000000	1.000000
25%	2.012233e+08	1.900024e+08	1.000000	20853.000000	1.600000e+03	39.022578	-77.198386	2019.000000	3.000000	8.000000
50%	2.012780e+08	2.000074e+08	1.000000	20878.000000	8.100000e+03	39.075859	-77.106564	2020.000000	6.000000	15.000000
75%	2.013320e+08	2.100194e+08	1.000000	20904.000000	1.250000e+04	39.142588	-77.032945	2021.000000	9.000000	23.000000
max	2.013872e+08	2.204211e+08	12.000000	38401.000000	2.090600e+06	39.342140	-76.905354	2022.000000	12.000000	31.000000

Figure 20 Descriptive Statistics for Numeric Columns

4.1.2 Additional Statistics for Non-Numeric Columns

- Examining categorical features (Offence Code, Dispatch Date / Time, NIBRS Code, Crime Name1, Crime Name2, Crime Name3, Police District Name, City, State, Agency) is shown in Figure 21.
- Count reveals data completeness.
- Unique shows the number of unique values.
- Top represents the most frequent value.
- Freq shows the frequency of the most frequent value.

```
In [4]: # Display additional statistics for non-numeric columns
        non_numeric_stats = data.describe(include='object')
        non_numeric_stats
```

Out[4]:

	Offence Code	Dispatch Date / Time	NIBRS Code	Crime Name1	Crime Name2	Crime Name3	Police District Name	City	State	Agency	...	Sector	Beat	PRA	Street Name	Street Type	Start
count	221542	214703	221542	221542	221542	221542	221542	221542	221542	221542	...	220037	220037	221317.0	221542	221454	
unique	576	196213	58	5	60	329	9	50	4	8	...	15	50	1529.0	7488	29	
top	2305	12/02/2019 04:39:39 PM	90Z	Crime Against Property	All Other Offenses	LARCENY - FROM AUTO	SILVER SPRING	SILVER SPRING	Maryland	MCPD	...	P	3G1	702.0	GEORGIA	RD	
freq	15913	48	56230	103315	54215	18875	46407	76284	221526	191943	...	21662	14588	3303.0	9696	55034	

4 rows × 21 columns

Figure 21 Descriptive Statistics for Non-Numeric Columns

4.1.3 Frequency Distribution

- Examining frequency of unique values in non-numeric columns as shown in Figure 22.

```
In [5]: # Display the frequency of each unique value in non-numeric columns
        value_counts = data.select_dtypes(include='object').apply(lambda x: x.value_counts()).T.stack()
        value_counts

Out[5]: Offence Code      1399          10896.0
                          2303           9972.0
                          2304           4841.0
                          2305          15913.0
                          2308           6148.0
                                          ...
        Time_of_the_day   Morning       55621.0
                          Evening       51417.0
                          Night         44512.0
                          Midnight      35442.0
                          Afternoon     34550.0
        Length: 438644, dtype: float64
```

Figure 22 Frequency of unique values in Non-Numeric columns

4.1.4 Correlation Matrix

- Correlation matrix explores relationships between numeric variables.
- Values close to 1 or -1 indicate strong positive or negative correlations, respectively.
- Values closer to 0 suggest weaker correlations.

```
In [6]: # Display correlation matrix for numeric columns
        correlation_matrix = data.corr()
        correlation_matrix
```

Out[6]:

	Incident ID	CR Number	Victims	Zip Code	Address Number	Latitude	Longitude	Start_Year	Month	Day	Time_Hour
Incident ID	1.000000	0.961522	0.012656	-0.015361	-0.000721	0.004505	-0.017153	0.970511	0.076359	-0.002008	-0.039975
CR Number	0.961522	1.000000	0.012230	-0.015435	-0.001644	0.003876	-0.016111	0.979201	-0.125452	-0.016784	-0.039772
Victims	0.012656	0.012230	1.000000	0.005548	0.002600	0.022746	-0.014238	0.013313	0.003272	0.001465	0.004312
Zip Code	-0.015361	-0.015435	0.005548	1.000000	0.000074	-0.017018	0.185977	-0.014317	0.000858	-0.002135	0.011316
Address Number	-0.000721	-0.001644	0.002600	0.000074	1.000000	0.175446	-0.124577	-0.001625	0.005273	-0.002295	0.003030
Latitude	0.004505	0.003876	0.022746	-0.017018	0.175446	1.000000	-0.687728	0.002977	0.003471	-0.001083	-0.008639
Longitude	-0.017153	-0.016111	-0.014238	0.185977	-0.124577	-0.687728	1.000000	-0.015234	-0.005742	0.000043	0.013040
Start_Year	0.970511	0.979201	0.013313	-0.014317	-0.001625	0.002977	-0.015234	1.000000	-0.139818	-0.013036	-0.030250
Month	0.076359	-0.125452	0.003272	0.000858	0.005273	0.003471	-0.005742	-0.139818	1.000000	0.000981	0.002698
Day	-0.002008	-0.016784	0.001465	-0.002135	-0.002295	-0.001083	0.000043	-0.013036	0.000981	1.000000	0.020973
Time_Hour	-0.039975	-0.039772	0.004312	0.011316	0.003030	-0.008639	0.013040	-0.030250	0.002698	0.020973	1.000000

Figure 23 Correlation Matrix

4.2 RELEVANCE AND SUITABILITY

TYPE	RELEVANCE
Temporal Trends and Patterns	Basic descriptive statistics are crucial for understanding temporal features' contribution to crime patterns.
Geospatial Analysis	Geographic features provide insights into spatial distribution of crimes. The correlation matrix helps identify relationships between geographic variables.
Crime Type Distribution	Examining frequency distribution of crime types is essential for identifying prevalent categories and understanding overall criminal activity.

Correlation Matrix	The matrix is valuable for identifying relationships between numeric variables, offering insights into spatial and temporal patterns.

Table 4 Relevance and Suitability

CHAPTER 5 EXPLORATORY DATA ANALYSIS

Exploratory Data Analysis (EDA) is a crucial step in the data analysis process, allowing us to understand the structure of dataset, identify patterns, and generate insights. Based on the provided information, EDA(*Knowledgehut Solutions Private Limited, 2011-23)* has already involved a comprehensive assessment of the dataset, including the identification and resolution of various data quality issues. Here are key points to discuss regarding EDA:

5.1 CURRENT EDA FINDINGS

5.1.1 Data Quality Assessment

- Identified issues include null values, unexpected data types, incorrect entries, and anomalies in various columns.
- Proposed solutions involve strategic measures, such as replacing missing values with mean values, data type conversion, and standardization of entries(*Medium.com, 2017).*

5.1.2 Data Transformation and Feature Engineering

- Temporal data filtering was applied to focus on recent trends.
- Missing geospatial information was addressed through systematic approaches, including mean and mode calculations.
- Standardization of city names, state names, and street names using dictionaries.
- Strategies for dealing with missing values in NIBRS codes and crime names were implemented.
- Correction of inconsistencies in police district numbers and names.

5.1.3 Visualization Techniques

- A variety of visualization techniques were chosen based on their suitability for conveying different types of information, such as pie charts, bar charts, donut charts, violin plots, radar charts, heatmaps, cluster maps, correlation matrices, scatter plots, density maps, line graphs, histograms, scatter plots with hues, subplots, etc.

Example: Density Map for Spatial Analysis

- One of the key visualizations used in this EDA process was Density Map. This technique visualizes the distribution of data across space.

Example Scenario:

- The Density Map was applied to illustrate case concentrations across the geographical area of Montgomery County.
- The Figure 24 revealed spatial patterns and concentrations, indicating areas with higher and lower crime rates.

- This visualization helped in understanding the geographical distribution of crimes and identifying potential hotspots.

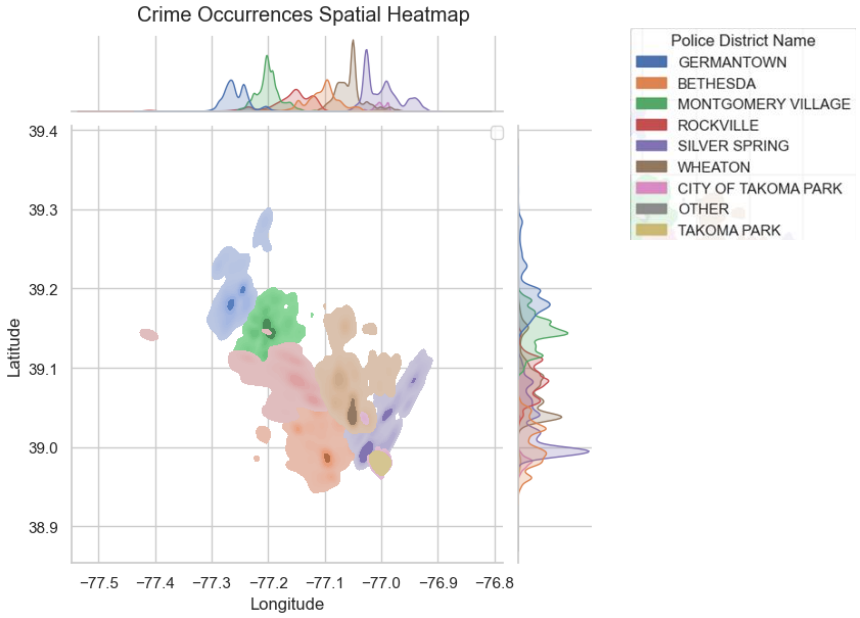


Figure 24 Density Map depicting Latitude, Longitude and Police District Name

5.2 FURTHER DATA CLEANING OR TRANSFORMATION

5.2.1 Remaining Data Quality Concerns

- While the applied methods have significantly improved data quality, there might be additional considerations based on the results of visualizations or further analyses.
- Checking for outliers or anomalies in the data, especially in geospatial information and temporal patterns.

5.2.2 Temporal Analysis

- Consider to explore temporal patterns in more detail, such as seasonal variations.
- Evaluate the effectiveness of the temporal filtering and assess if any additional adjustments are needed.

5.2.3 Geospatial Analysis

- Validate the accuracy of geospatial information, especially after the handling missing values.
- Explore spatial patterns and correlations, considering the impact of geographical factors on crime rates.

Example Analysis:

- Consider to compare the Density Map results with other geospatial visualizations, like scatter plots or cluster maps, to validate the consistency.

- This example highlights how a specific visualization technique, in this case, the Density Map, contributes to the EDA by providing insights into spatial patterns. It acts as a tool to identify potential areas for investigation and validates the effectiveness of geospatial transformation.

5.2.4 Feature Engineering

- Assessed the effectiveness of derived features like day, month, date, hour, and period of the day in capturing meaningful patterns.
- Explore additional features that might enhance the understanding of crime trends.

5.3 POSSIBLE APPROACHES

5.3.1 Iterative Data Cleaning

- Review visualizations and patterns to identify any residual issues or outliers that may require further cleaning.
- Iterate on data cleaning steps based on insights gained from visualizations.

5.3.2 Advanced Analysis

- Consider more advanced statistical analyses or machine learning models to uncover hidden patterns or relationships in the data.
- Evaluate the need for predictive modelling based on the characteristics of the dataset.

5.3.3 Collaborative Validation

- Engage with domain experts or stakeholders to validate the findings and ensure that the data transformations align with the real-world context.

CHAPTER 6 DATA VISUALIZATION

Data visualization is the graphical representation of information and data, using visual elements such as charts, graphs, maps, and dashboards.

Data visualizations are generated by transforming raw data into visual formats using tools like Tableau, Excel, or Python libraries. The process involves selecting the appropriate visualization type based on the data's nature.

6.1 ADVANTAGES OF VISUALIZATIONS

1. **Enhances Understanding:** Simplifies complex datasets for easy comprehension of trends and patterns.

2. **Facilitates Analysis:** Identifies outliers and supports effective data analysis.

3. **Supports Decision-Making:** Aids decision-makers in making informed choices.

4. **Communicates Insights:** Effectively conveys findings to both technical and non-technical audiences.

5. **Identifies Trends and Patterns:** Reveals correlations and trends within data.

6. **Enhances Memorability:** More memorable than raw data, aiding information retention.

7. **Supports Storytelling:** Guides audiences through a narrative supported by data.

8. **Enables Data Exploration:** Interactive visualizations allow real-time data exploration.

Data visualizations play a crucial role in transforming complex data into accessible insights, fostering understanding, and supporting decision-making across various industries and fields.

6.2 TYPES OF VISUALIZATIONS

Visualization Type	Description	Used (Yes/No)	Reason for Use/ Not Use
Pie Chart	Circular statistical graphic that is divided into slices to illustrate numerical proportions	Yes	Used for displaying percentage distributions and conveying a quick overview of data distribution.
Bar Chart	Represents data with rectangular bars, where the lengths are proportional to the values they represent	Yes	Used for providing a granular perspective on data and displaying counts in different categories.
Donut Chart	Ring-shaped chart with a blank centre, representing data with a central area that can contain additional information	Yes	Used to detail percentage distributions within sectors, adding layers of information to a chart.
Violin Plot	Combines aspects of a box plot and a kernel density plot, displaying the distribution of quantitative data across different categories	Yes	Used for illustrating data distribution, especially in terms of crime types and their temporal correlation.
Radar Chart	Graphical method of displaying multivariate data in the form of a two-dimensional chart of three or more quantitative variables	Yes	Used for visually outlining the distribution of data across multiple variables, revealing patterns.
Heatmap	Data visualization technique that shows the magnitude of a phenomenon as colour in two dimensions	Yes	Used for revealing patterns and changes in data, especially in the context of Police District Number.
Cluster Map	Displays dendrograms revealing relationships and grouping similar items	Yes	Used for revealing relationships and grouping similar items within a dataset.
Correlation Matrix	Visual representation of the correlation between different variables	Yes	Used for quantifying the strength and direction of relationships between features in a dataset.

Scatter Plot	Displays values for two variables in a dataset as points on a graph.	Yes	Used for providing a visual representation of case distribution patterns and revealing spatial concentrations.
Density Map	Visualizes the distribution of data across space, often used to show the concentration of a variable in geographical space	Yes	Used for illustrating case concentrations across a geographical area and revealing spatial patterns.
Line Graph	Displays information as a series of data points called 'markers' connected by straight line segments	Yes	Used for visualizing temporal trends over time, especially when looking at monthly crime trends.
Histogram	Represents the distribution of a dataset by dividing the data into intervals and displaying the count of observations in each interval	Yes	Used for providing an overview of data distribution but found limited in revealing detailed patterns.
Scatterplot with Hues	Similar to a regular scatter plot but includes a hue dimension, providing additional information	Yes	Used for assessing the overall spatial crime distribution with additional information about different times.
Subplots	Multiple smaller plots within a single plot, allowing for the examination of different aspects of the data	Yes	Used for conducting detailed analysis within subplots, isolating each time of the day.
Box Plot	Displays the distribution of data based on a five-number summary: minimum, first quartile, median, third quartile, and maximum	No	Not used as it does not provide additional insights beyond what is already covered by the Violin Plot.
Bubble Chart	A scatter plot in which a third dimension of the data is shown through the size of markers	No	Not used as it adds complexity without significantly enhancing the understanding of the data distribution.

Table 5 Visualization Types

6.3 RESEARCH QUESTIONS

- The question number column in both Table 6 and Table 7 is aligned, facilitating seamless cross-reference between the two tables.

Q. No	Questions
1	Explain Pie Chart & Bar Chart showing Crime Name 1 distribution across places from 2018-2022. How is Crime against Property detailed in Bar Graph (Crime Name 1 count) & Donut Chart (percentage distribution within sectors)?
2	Describe the trend in victim counts from 2018 to 2022 using the bar chart . What insights does the violin plot provide about crime types and their temporal correlation on victim counts?
3	How does the radar chart depicting crime across street types complement the 2022 bar graph ? What additional insights are gained, especially concerning crime counts on roads and avenues?
4	Summarize notable trends in the crime heatmap and correlation map. Focus on Police District Number changes from 2018 to 2022.
5	Explain insights from the scatter plot (latitude, longitude, police district names) and density map. What do these visuals reveal about case distribution patterns and how different police districts handle incidents across the city?
6	Briefly describe the trends in the overall crime rate in Montgomery County from 2018 to 2022. How did the pandemic influence crime rates, specifically crime against property?
7	Discuss the distribution of crimes among cities in Montgomery County, emphasizing the difference between urban and rural areas.
8	How does the stacked bar chart reveal distinct patterns in different types of crimes each month, especially during the holiday season in June and July?
9	Summarize trends in crime occurrences over the five-year period (2018-2022) based on monthly data, emphasizing recurring patterns.
10	Explain the spatial and temporal distribution of crime in Poolesville, Montgomery County, focusing on town centre vs. countryside areas and insights into crime rates along highways near motels and gas stations.
11	Describe how crimes are distributed across various settings (vehicles, homes, ATMs, banks, etc.) and the insights this provides into overall crime patterns.
12	Explore the correlation of different crime names within police districts, identifying the leading police district and insights from the top ten Crime 2 values and highest occurring Crime 3 values.

Table 6 Research Questions

6.4 USE CASE SCENARIOS

Q No.	Statistical Method	Visualizations	Figure Number
1.	Descriptive Statistics, Percentage Distribution	Pie Chart, Bar Chart, Donut Chart	Figure 25, Figure 26
2.	Trend Analysis, Descriptive statistics	Bar Chart, Violin Plot	Figure 27, Figure 28
3.	Descriptive statistics, Radar Chart	Radar Chart, Bar Graph	Figure 29, Figure 30
4.	Descriptive statistics, Heatmap, Cluster Map, Correlation Map	Heatmap, Cluster Map, Correlation Map	Figure 31, Figure 32
5.	Spatial Analysis, Descriptive statistics	Scatter Plot, Density Map	Figure 33, Figure 34
6.	Trend Analysis, Comparative Analysis	Line Graph, Bar Graph	Figure 35, Figure 36
7.	Comparative Analysis, Spatial Analysis	Plot with Diverse Biomes, Involvement of Police Districts	Figure 37, Figure 38
8.	Time Series Analysis, Comparative Analysis	Stacked Bar Chart	Figure 39, Figure 40
9.	Time Series Analysis	Histogram, Line Graph	Figure 41, Figure 42, Figure 43
10.	Spatial and Temporal Analysis	Scatterplot with Hues, Subplots	Figure 44, Figure 45
11.	Descriptive statistics	Histogram	Figure 46
12.	Comparative Analysis	Count Plot, Pie Chart	Figure 47, Figure 48, Figure 49

Table 7 Use Case Scenarios

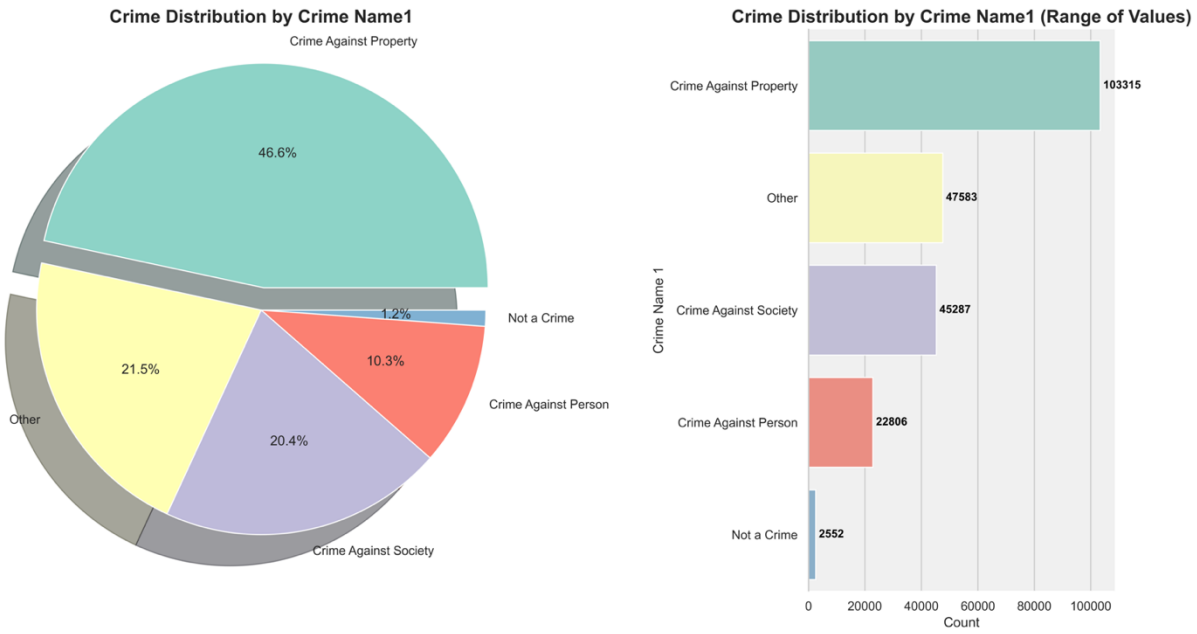


Figure 25 Pie Chart and Bar Graph depicting Crime Name 1

Figure 25 displays the percentage distribution of Crime Name 1 across different locations from 2018 to 2022 in a Pie Chart(*Datacamp, 2016).* Simultaneously, the Bar Chart details the precise crime counts for Crime Name 1 in each location, providing a comprehensive view of incident prevalence. These visuals offer insights into the overall crime landscape during the specified timeframe.

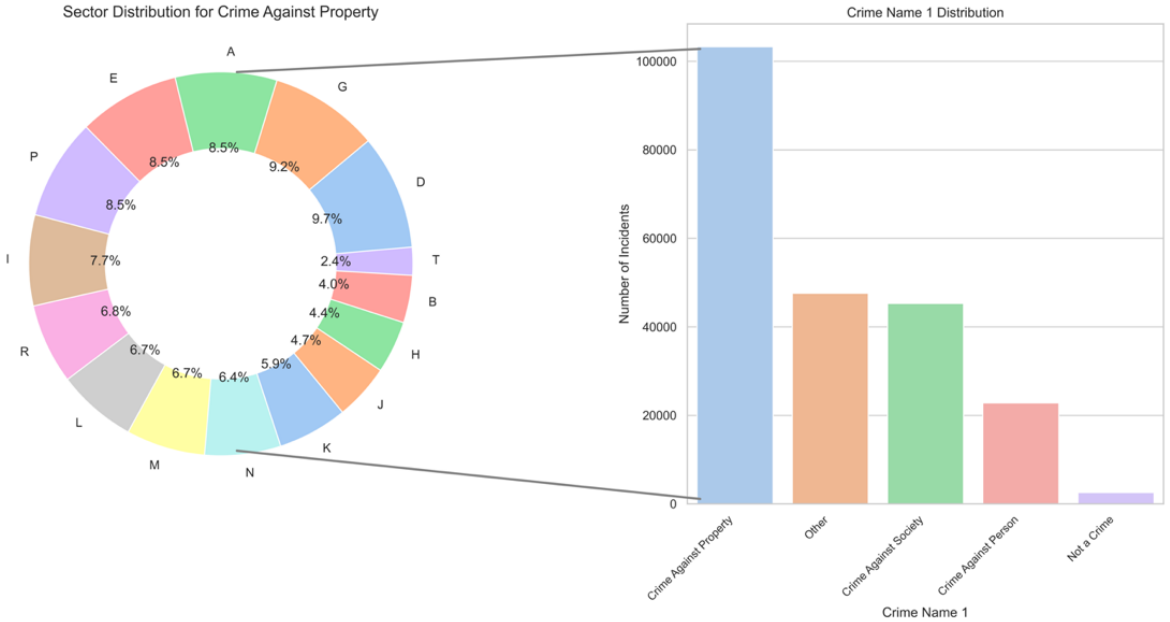


Figure 26 Bar Graph and Donut chart depicting Crime Name 1 and Sector

In Figure 26 bar chart illustrates a noticeable decline in the count of victims from 2018 to 2022, with the highest count recorded in 2018. Conversely, the violin plot reveals a distinct pattern in the number of victims across different days of the week. Surprisingly, Wednesday stands out as the day with consistently higher victim counts over the years, suggesting a potential correlation between mid-week days and increased incidents. The combination of these visualizations prompts a closer examination of both the annual trend and the day-wise variations in victim counts to better understand the underlying factors contributing to these patterns.

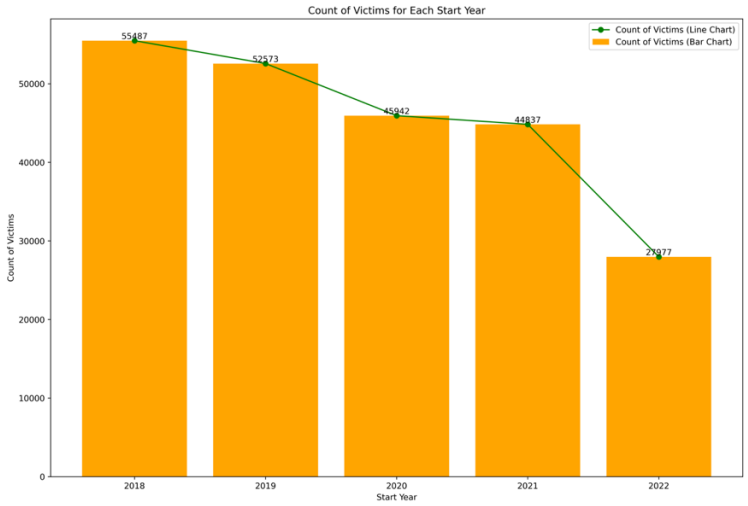


Figure 27 Bar Graph depicting Victims and Case Start Year

Figure 27 bar chart depicts a notable decline in victim counts from 2018 to 2022, with the peak occurring in 2018. This decline signifies a trend in reduced victimization over the four-year period.

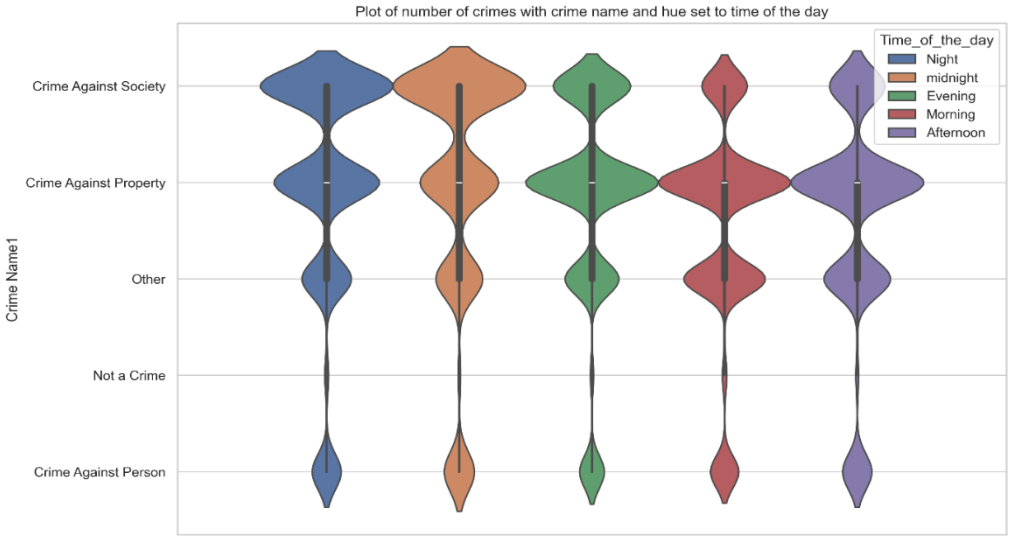


Figure 28 Violin Chart depicting Crime Name 1 and Time of a day

Figure 28 violin plot(*Hunter J, 2012*) reveals patterns in crime types and their temporal correlation with victim counts, emphasizing crime against society at night and midnight and crime against property in the morning and evening. These insights are vital for targeted strategies in victim support, crime prevention, and public safety initiatives & also shows that crime against society is more at night and midnight, while property crime occurs in morning.

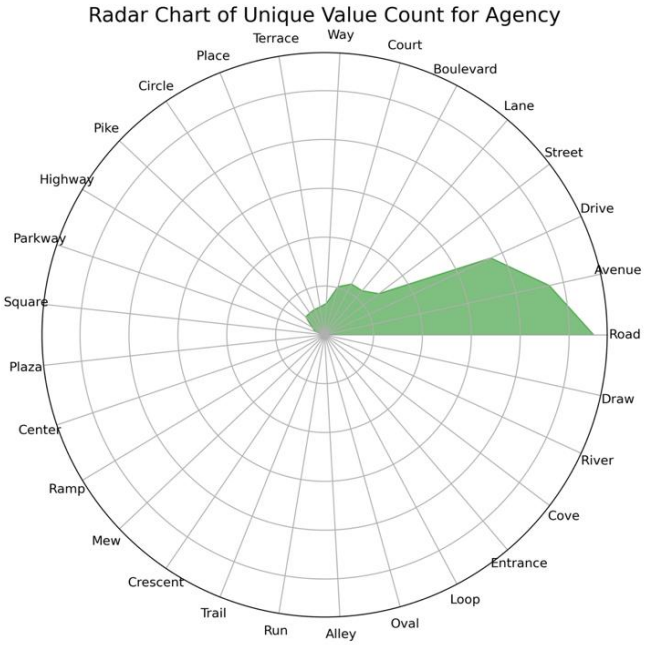


Figure 29 Radar Chart depicting Street Types

Figure 29 radar chart illustrates the crime distribution across street types from 2018 to 2022, emphasizing roads and avenues. The chart suggests potential patterns influenced by factors like population density, visibility, lighting, and security measures. It provides a comprehensive overview of crime frequency in various street environments over the specified time period.

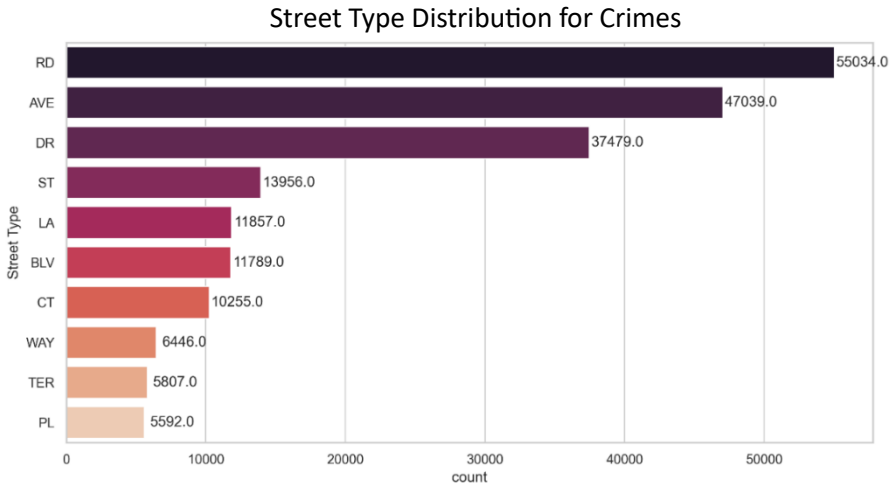


Figure 30 Bar Chart depicting Street Types

Figure 30 bar graph, displaying 2022 crime counts for the top ten locations, highlights stark differences like 55,034 crimes on roads versus 5,592 in Plazas. This underscores the urgency in addressing high incidents on roads and avenues, raising questions about the effectiveness of security measures, especially in low-crime areas. These graphs advocate for targeted strategies and a deeper understanding of dynamics influencing crime rates in different street types for effective prevention.

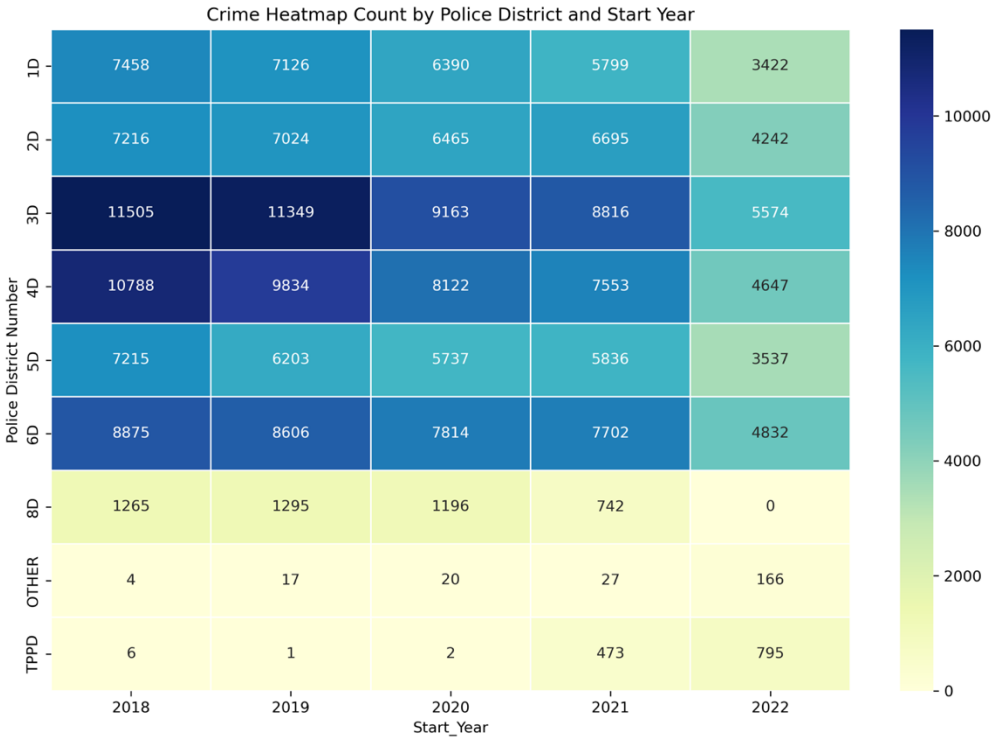


Figure 31 Heatmap depicting Police District Number and Case Start Year

Figure 31 crime heatmap reveals a notable decline in reported crimes in Police District 3D from 2018 to 2022. In 2018, it had the highest crime number, while TPPD experienced an increase. Police District 8D reported zero crimes in 2022, signalling a positive trend. This visual offers immediate insight into changes in crime rates across police districts over the specified time frame(*Python Data Science Handbook, n.d*).

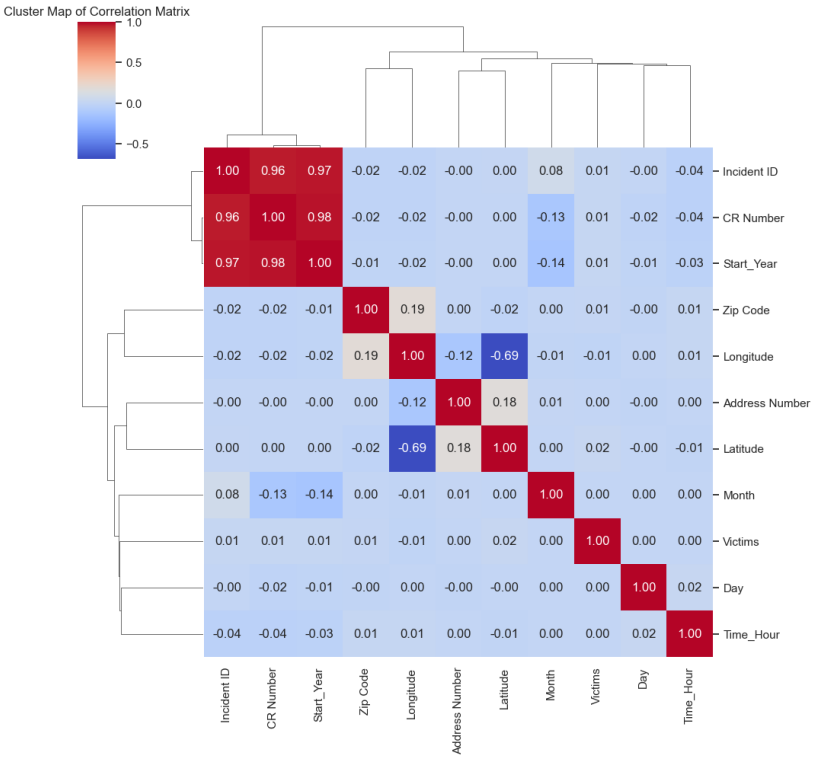


Figure 32 Cluster Map of Correlation Matrix

Figure 32 cluster and correlation maps offer insights into our dataset structure. The cluster map generates dendrograms, revealing relationships, while the correlation map quantifies feature relationships, with 1 as perfect positive, -1 as inverse, and 0 as no correlation. Together, these visuals provide a concise yet comprehensive understanding of inherent patterns and interconnections in our data.

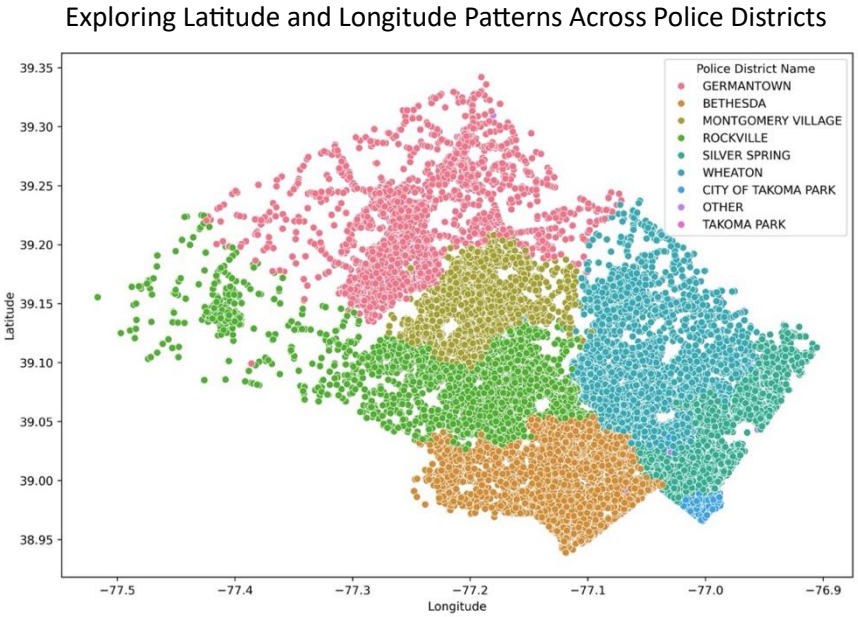


Figure 33 Cluster Map depicting Latitude, Longitude and Police District Name

Figure 33 scatter plot(*Stackoverflow.com, n.d)* visually reveals case distribution patterns by police district, highlighting clusters and insights into area-specific incidents. Latitude and longitude pinpoint geographical concentrations, aiding the identification of high-incidence zones. This tool is crucial for understanding case spatial distribution and formulating targeted law enforcement strategies.

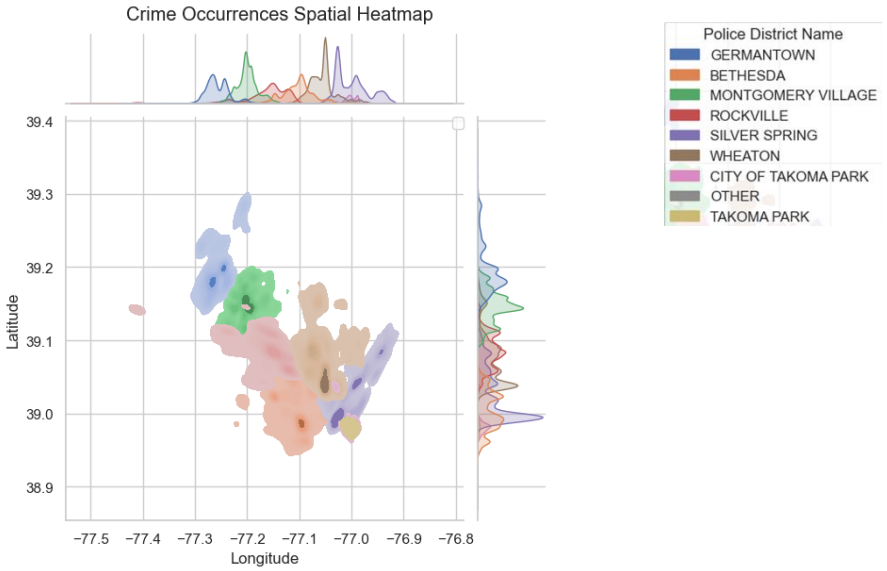


Figure 34 Density Map depicting Latitude, Longitude and Police District Name

Figure 34 density map(*Python Graph Gallery, n.d.)* complements the scatter plot, vividly illustrating case concentrations. Higher concentrations in city centre areas suggest increased law enforcement demands, while rural districts like Poolesville and Damascus show lower visibility due to fewer cases. This disparity provides a nuanced understanding of urban-rural variations, aiding resource allocation and strategic planning for case distribution patterns.

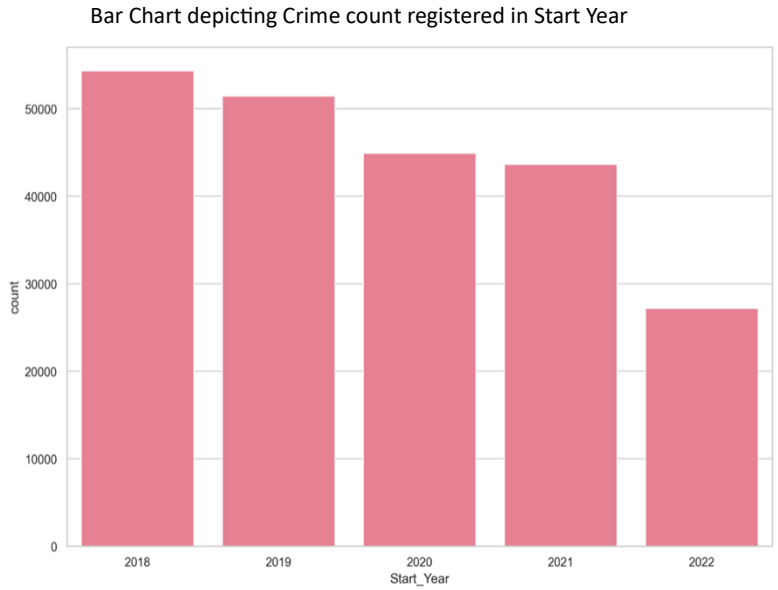


Figure 35 Bar Chart depicting Crime count registered in Start Year

Figure 35 2018 had the highest crime rate, followed by a decline in 2019. Significant reductions in 2020 and 2021, attributed to the pandemic, align with societal changes. Note: 2022 data (ending in August) isn't considered for a conclusive assessment

Count Plot depicting Start Year crime count with Crime Name 1

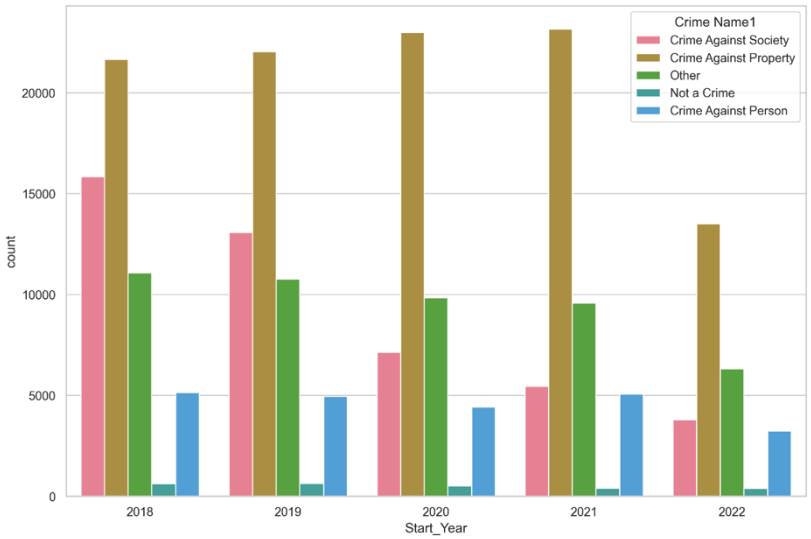


Figure 36 Count Plot depicting Start Year crime count with Crime Name 1

Figure 36 highlights a persistent all-time high in property crimes, despite an overall rate decrease in society crime in 2020 and 2021. This surge during the pandemic is attributed to panic-driven shoplifting for essentials, leading to increased incidents in supermarkets. Despite the overall rate decline, this category saw a concerning rise due to changing societal dynamics.

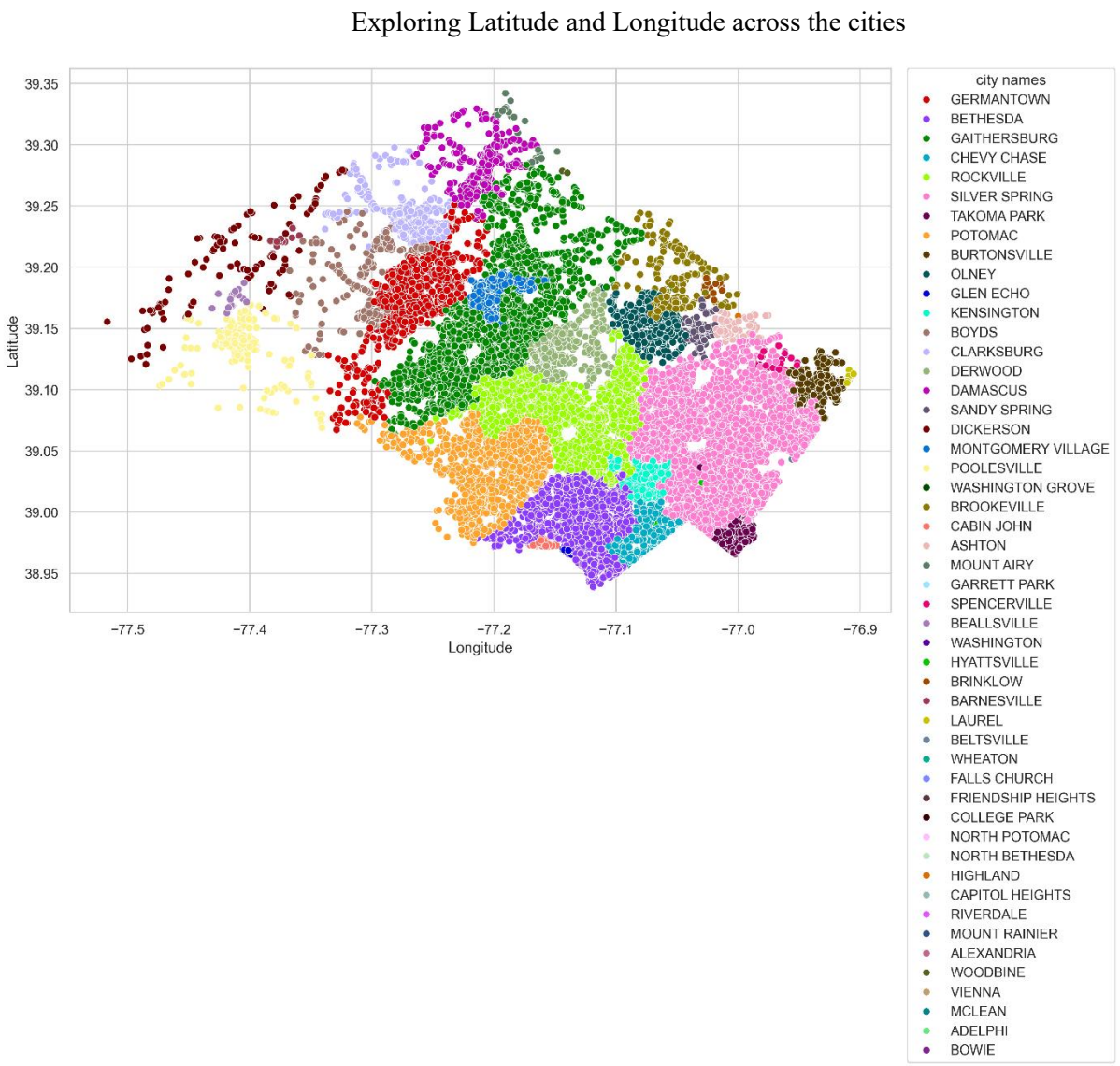


Figure 37 Scatter Plot depicting City Names with Latitude and Longitude

Figure 37 the geographical distribution of crime incidents by mapping latitude and longitude coordinates with respect to City Names. Utilize a diverse color palette with fifty distinct colors to facilitate a clearer and more nuanced interpretation of spatial patterns.

Diverse Biomes and Police District Interaction

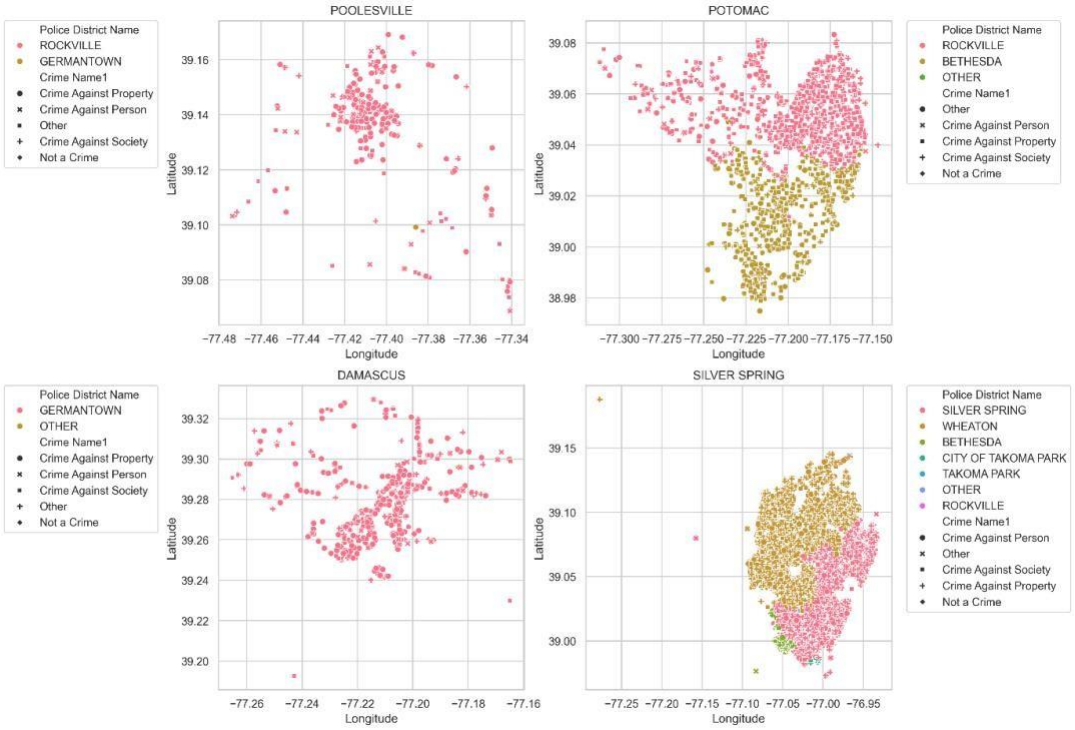


Figure 38 Plot with Diverse Biomes, Involvment of Police Districts

Figure 38 is reinforced by police district involvement in crimes. Poolesville and Damascus involve only 1 or 2 districts, emphasizing lower crime rates. Urban areas like Potomac and Silver Spring engage multiple districts, offering insights into crime distribution and the urban- rural distinction in Montgomery County.

Count Plot Representation

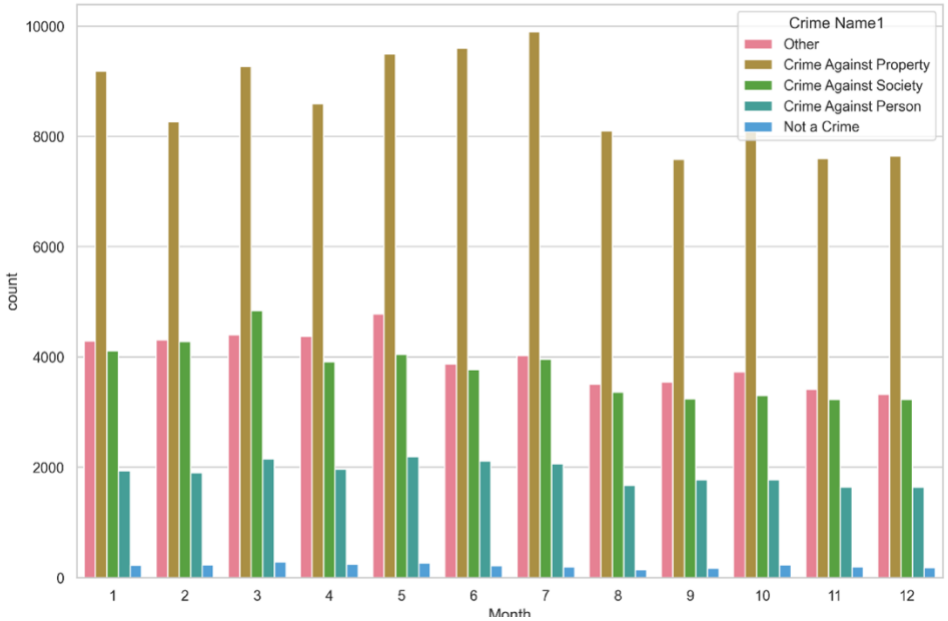


Figure 39 Count Plot Depicting Crime Count

Figure 39 illustrates monthly crime counts, noting a surge during the holiday season in June and July. This increase warrants a closer examination of contributing factors and focused attention on law enforcement strategies during the holidays.

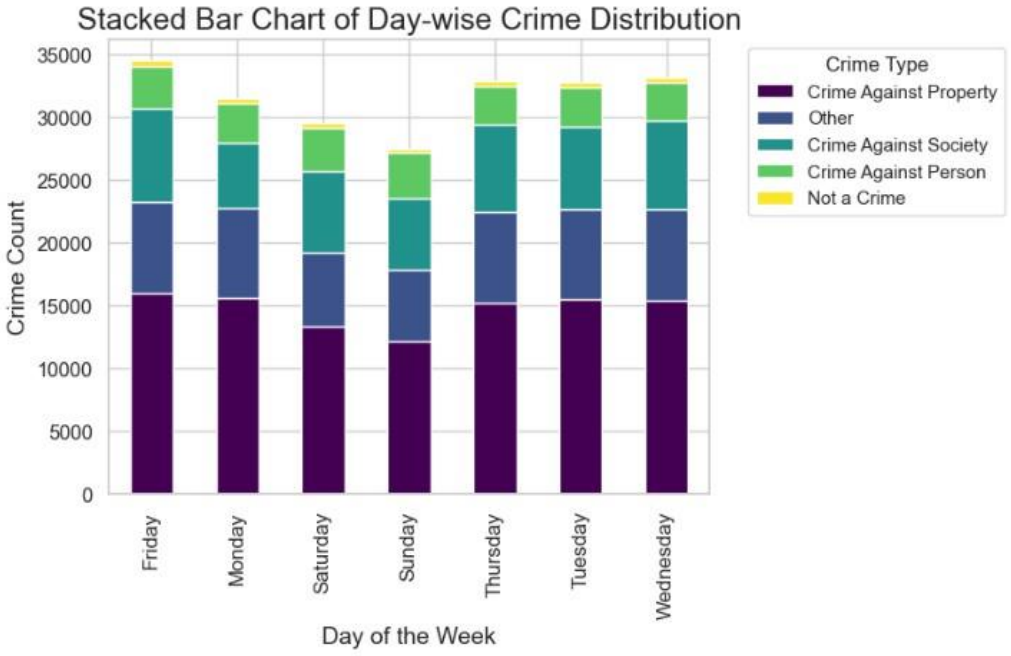


Figure 40 Stacked Bar Chart Depicting Crime Count

Figure 40 stacked bar chart effectively communicates weekly crime distribution. Crime against property consistently dominates, peaking on Fridays with about 16,000 incidents. In contrast, crime against persons has a lower count, with Sundays recording the highest frequency. This raises questions about the alignment of incidents initially classified as crimes with legal definitions, emphasizing the need for meticulous categorization. The visualization provides a comprehensive understanding of weekly variations, shedding light on distinct patterns within different crime categories.

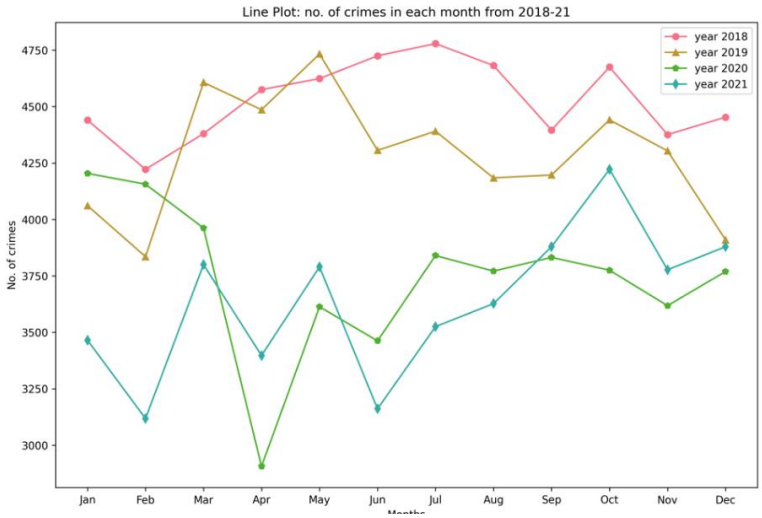


Figure 41 Line Plot depicting number of crime happened in each month during 2018-21

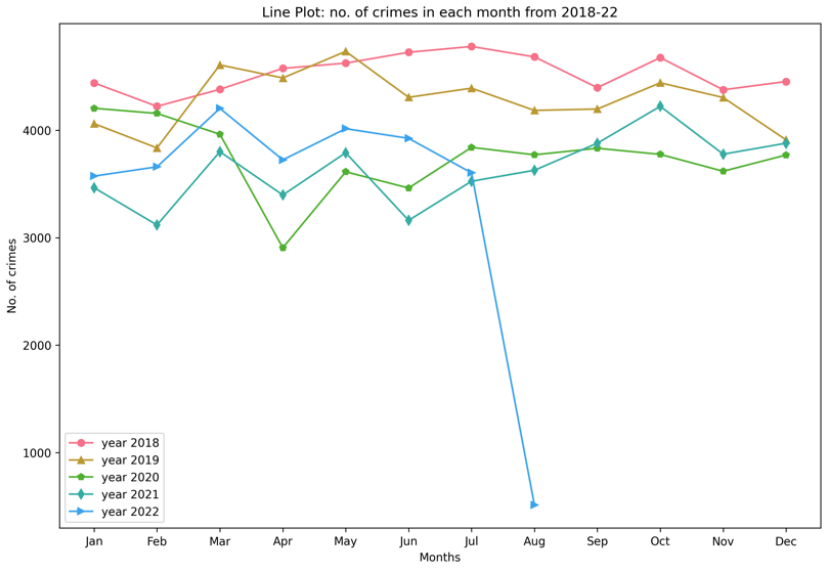


Figure 42 Line Plot depicting number of crime happenned in ecah month during 2018-22

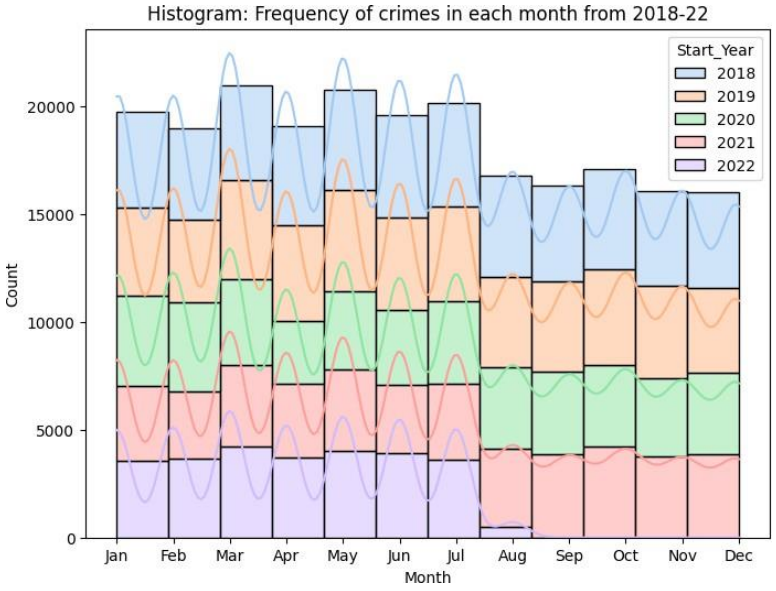


Figure 43 Histogram depicting frequency of crimes in each month during 2018-22

The histogram lacked granularity, showing consistent monthly crime counts of 2900 to 4700 over five years. To address this, a line graph revealed clearer temporal trends. March consistently saw increased crimes, except in 2020, followed by declines in April, May's slight increases, and higher October rates compared to September in most years. Overall, crime counts fluctuated between 2900 and 4700 per month. The analysis concludes recurring patterns: March consistently had increased crime, April declined, May rose slightly, and October consistently surpassed September, except in 2020. These seasonal fluctuations suggest potential trends, urging further investigation into underlying causes.

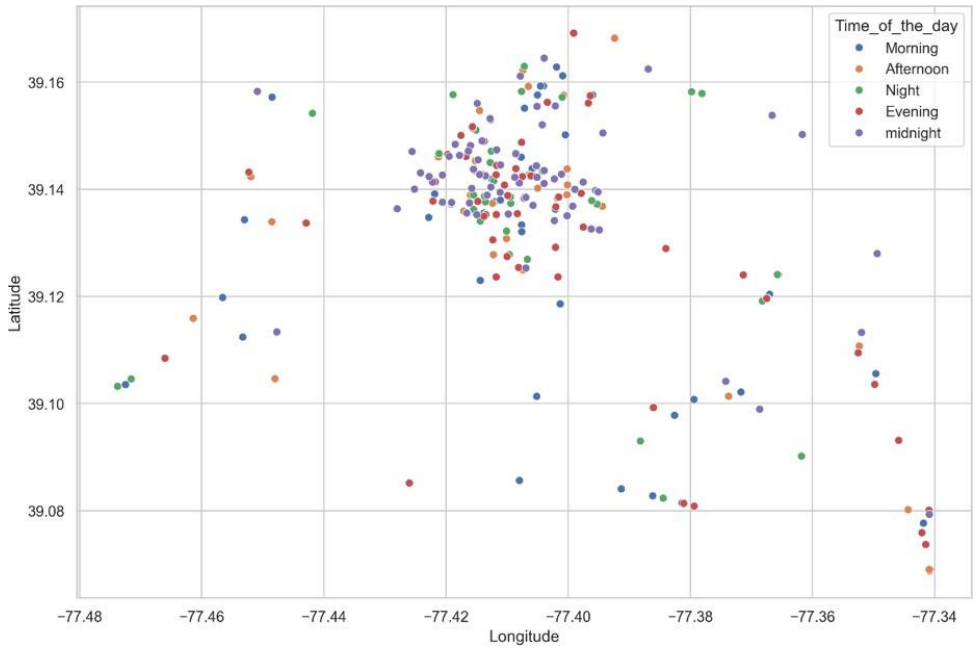


Figure 44 Scatter Plot depicting crime in Poolesville and Montgomery County

In Figure 44 research explores the interplay of spatial location, time of day, and crime in Poolesville, Montgomery County. Using a scatterplot with hues for different times was assessed for spatial crime distribution, gaining insights into temporal patterns.

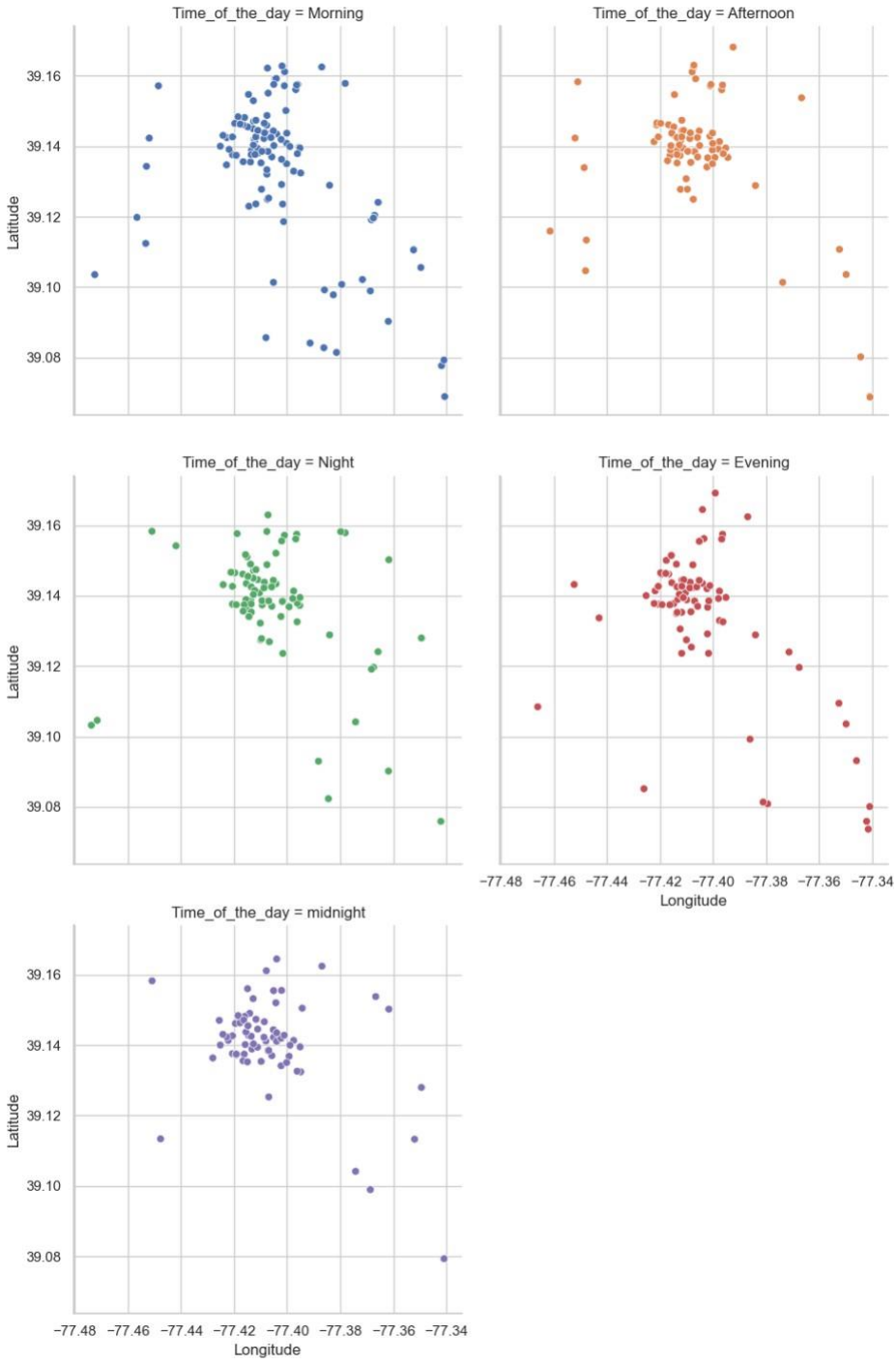


Figure 45 Subplot depicting crimes for time_of_the_day wrt Latitude and Longitude

Figure 45 detailed subplots analysis during each time of the day, revealing patterns. Morning and evening see concentrated crimes citywide, correlating with commuting. At night and midnight, crimes center in the city, suggesting reduced outskirts activity, emphasizing city center;s prominence during night incidents.

Visualizing Patterns Through Histogram analysis

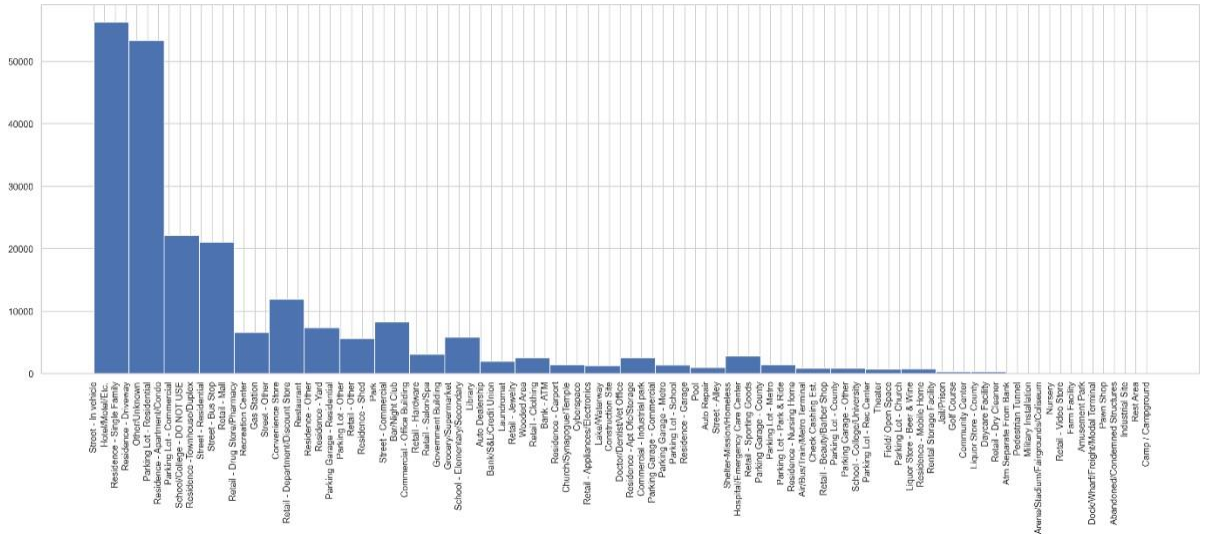


Figure 46 Histogram depicting crime distribution count

In Figure 46 research characterizes crime distribution across settings using a histogram. It visually represents crime prevalence in various locations, aiding law enforcement strategies, resource allocation, and community safety initiatives. Identifying high-incidence areas allows for targeted interventions to enhance public safety.

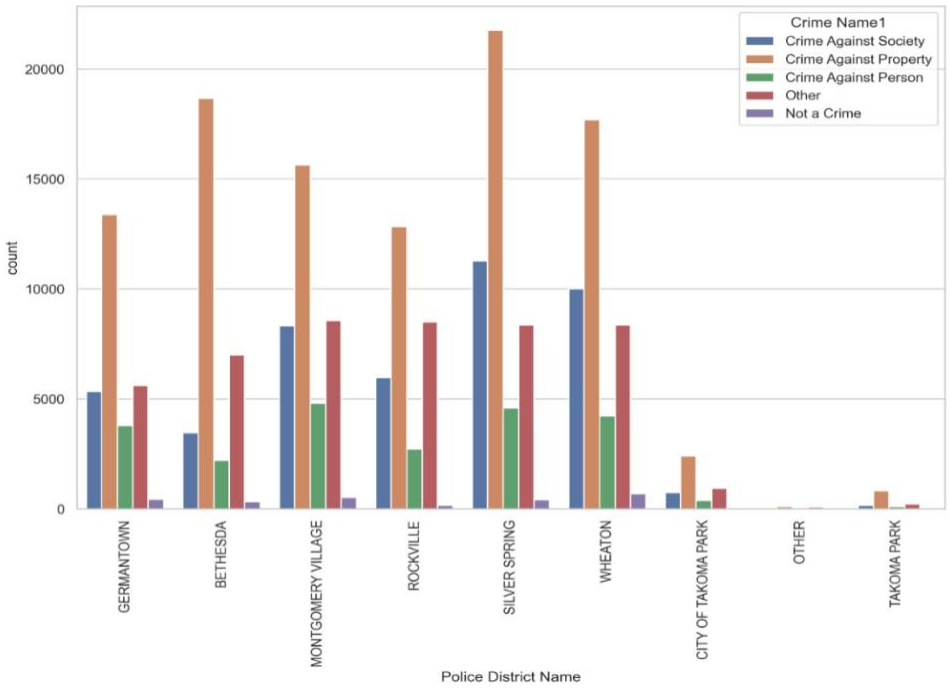


Figure 47 Count Plot depicting interplay between crime names in Police Districts

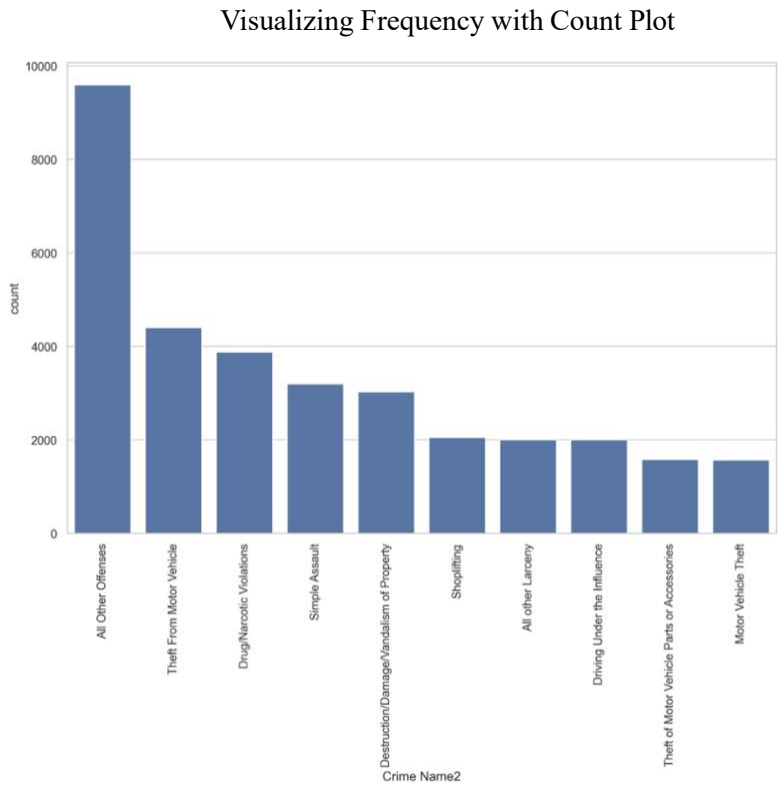


Figure 48 Count Plot depicting top ten crime values

In Figure 47 count plot, it was examined that the top ten Crime 2 values in the leading police district, offering insights into specific nuances. Figure 48 further focuses on the highest occurring Crime 3 values from the Crime 2 plot, presented in a pie chart in Figure 49.

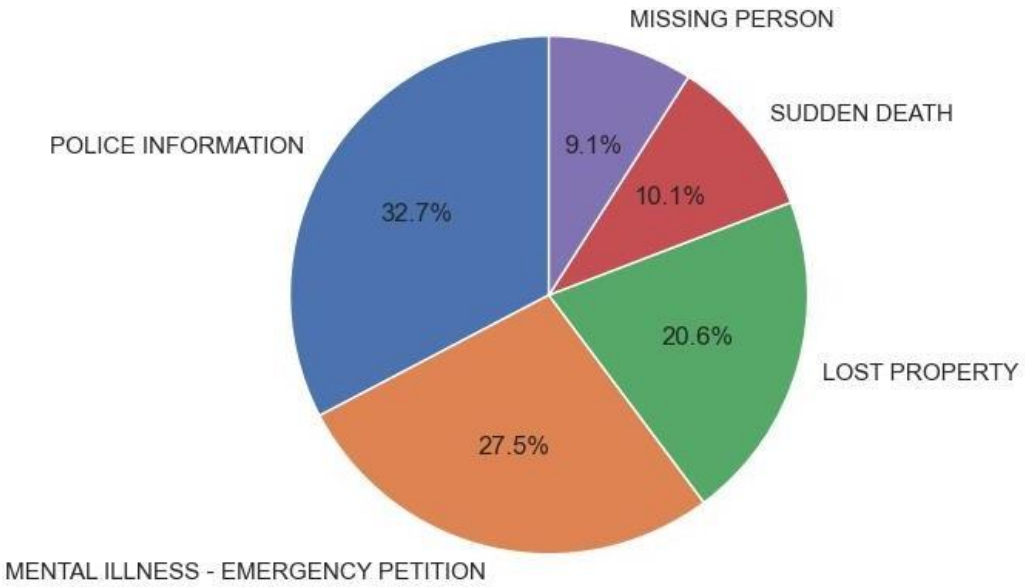


Figure 49 Pie chart depicting highest occurring Crime 3 values from the Crime 2 plot

CHAPTER 7 SUMMARY AND CONCLUSION

- Analysing Montgomery County's crime patterns, this study employs visualizations to enhance comprehension for both technical and non-technical audiences.
- Through iterative processes, including data cleaning, transformation, and visualization, the analysis demonstrates a committed effort to refine dataset understanding and ensure its reliability.
- Concluding the research, it emphasizes the indispensable role of data-driven approaches in tackling complex societal issues like crime.
- The obtained insights not only form a robust foundation for further investigations and targeted policy interventions but also contribute to collaborative efforts aimed at fortifying public safety in Montgomery County.
- The study showcases the influential role of data analytics in informing evidence-based decision-making and driving proactive solutions to intricate challenges.

CHAPTER 8 REFERENCES

[1] Bowser, M. (2015). Police Districts and police service areas [online]. Available at: https://mpdc.dc.gov/patrolservices [Accessed 18 November 2023].

[2] COPS. (2018). Introductory Guide to Crime Analysis and Mapping [Document]. Available at: https://portal.cops.usdoj.gov/resourcecenter/ric/Publications/cops-w0273-pub.pdf [Accessed 6 November 2023].

[3] Datacamp. (2016). Python Seaborn Tutorial For Beginners: Start Visualizing Data [online]. Available at: https://www.datacamp.com/tutorial/seaborn-python-tutorial [Accessed 16 November 2023].

[4] FBI Federal Bureau of Investigation. (2018). Crime in the United States [online]. Available at: https://ucr.fbi.gov/crime-in-the-u.s/2018/crime-in-the-u.s.-2018 [Accessed 1 November 2023].

[5] Healthy People 2030. (2016). Uniform Crime Reporting System (UCR) [online]. Available at: https://health.gov/healthypeople/objectives-and-data/data-sources-and-methods/data-sources/uniform-crime-reporting-system-ucr [Accessed 4 November 2023].

[6] Hunter, J. (2012). Matplotlib [online]. Available at: https://matplotlib.org/stable/tutorials/index.html [Accessed 23 November 2023].

[7] Knowledgehut Solutions Private Limited. (2011-23). Exploratory Data Analysis (EDA): Types, Tools, Process [online]. Available at: https://www.knowledgehut.com/blog/data-science/eda-data-science [Accessed 22 November 2023].

[8] Medium.com. (2017). US Crime Data Exploration and Analysis [online]. Available at: https://medium.com/@the_psy_guy/us-crime-data-exploration-and-analysis-3265ce1373b4 [Accessed 11 November 2023].

[9] Montgomery County, Maryland. (n.d.). Montgomery County Department of Police [online]. Available at: https://www.montgomerycountymd.gov/pol/crime-data.html [Accessed 15 November 2023].

[10] Python Data Science Handbook. (n.d.). Visualization with Seaborn [online]. Available at: https://jakevdp.github.io/PythonDataScienceHandbook/04.14-visualization-with-seaborn.html [Accessed 22 November 2023].

[11] Python Graph Gallery. (n.d.). Matplotlib [online]. Available at: https://python-graph-gallery.com/matplotlib/ [Accessed 17 November 2023].

[12] Stackoverflow.com. (n.d.). Scatterplot without linear fit in seaborn [online]. Available at: https://stackoverflow.com/questions/29637150/scatterplot-without-linear-fit-in-seaborn [Accessed 25 November 2023].

[13] Tavares, E., III. (n.d.). Initial and Exploratory Analysis Method [online]. Available at: https://etav.github.io/articles/ida_eda_method.html [Accessed 17 November 2023].

[14] Tuychiev, B. (n.d.). Data visualization in Python using Seaborn [online]. Available at: https://blog.logrocket.com/data-visualization-python-seaborn/ [Accessed 24 November 2023].

[15] Waskom, M. (n.d.). seaborn: statistical data visualization [online]. Available at: https://seaborn.pydata.org/ [Accessed 24 November 2023].

[16] Waskom, M. (2018). Overview of seaborn plotting functions [online]. Available at: https://seaborn.pydata.org/tutorial/function_overview.html [Accessed 22 November 2023].

[17] Montgomery County, Maryland. (2015). Montgomery County Department of Police [online]. Available at: https://www.montgomerycountymd.gov/pol/districts.html [Accessed 3 November 2023].

[18] City of Takoma Park. (2016). Takoma Park Today [online]. Available at: https://takomaparkmd.gov/about-takoma-park/takoma-park-today/#:~:text=Takoma%20Park%20lies%20within%20Montgomery,House%20of%20Representatives%20District%208 [Accessed 5 November 2023].

[19] National Center for State Courts. 2017). Wayfarer Code Tables: OFFType [online]. Available at: https://apps.ncsc.org/wayfarer1.9/codetables/OFFType.asp [Accessed 6 November 2023].

[20] US Department of Justice. (2016). NIBRS Technical Specification [online]. Available at: https://www.dps.texas.gov/sites/default/files/documents/ucr/documents/nibrs_techspec2017v3_1.pdf [Accessed 8 November 2023].

[21] UCR Program. (2017). NIBRS Offense Code [online]. Available at: https://ucr.fbi.gov/nibrs/2011/resources/nibrs-offense-codes [Accessed 8 November 2023].

In [14]:
```
# let's import all the required libraries
```

In [15]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import colorcet as cc
from matplotlib.gridspec import GridSpec
```

In [16]:
```
### let's read the CSV file
```

In [17]:
```
crime_data = pd.read_csv('Crime_Dataset_Coursework1.csv', low_memory=Fals
```

In [18]:
```
crime_data.head()          # display top 5 rows of dataset
```

Out[18]:

	Incident ID	Offence Code	CR Number	Dispatch Date / Time	NIBRS Code	Victims	Crime Name1	Crime Name
0	201202980	3550	180042096	08/23/2018 09:52:08 PM	35B	1	Crime Against Society	Drug Equipme Violatio
1	201181293	3522	180015424	03/30/2018 01:00:55 AM	35A	1	Crime Against Society	Drug/Narcot Violatio
2	201181293	3562	180015424	03/30/2018 01:00:55 AM	35A	1	Crime Against Society	Drug/Narcot Violatio
3	201193163	3520	180029476	06/14/2018 10:26:45 PM	35A	1	Crime Against Society	Drug/Narcot Violatio
4	201204355	2204	180043926	09/03/2018 12:06:54 PM	220	1	Crime Against Property	Burglary/Breaki and Enteri

5 rows × 30 columns

In [19]:
```
### Let's see the information of the dataframe
```

In [20]:
```
crime_data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 306094 entries, 0 to 306093
Data columns (total 30 columns):
 #   Column                  Non-Null Count   Dtype  
---  ------                  --------------   -----  
 0   Incident ID             306094 non-null  int64  
 1   Offence Code            306094 non-null  object 
 2   CR Number               306094 non-null  int64  
 3   Dispatch Date / Time    257065 non-null  object 
 4   NIBRS Code              306094 non-null  object 
 5   Victims                 306094 non-null  int64  
 6   Crime Name1             305822 non-null  object 
 7   Crime Name2             305822 non-null  object 
 8   Crime Name3             305822 non-null  object 
 9   Police District Name    306000 non-null  object 
 10  Block Address           279888 non-null  object 
 11  City                    304818 non-null  object 
 12  State                   306094 non-null  object 
 13  Zip Code                302915 non-null  float64
 14  Agency                  306094 non-null  object 
 15  Place                   306094 non-null  object 
 16  Sector                  304564 non-null  object 
 17  Beat                    304564 non-null  object 
 18  PRA                     305855 non-null  object 
 19  Address Number          279985 non-null  float64
 20  Street Prefix           13631 non-null   object 
 21  Street Name             306093 non-null  object 
 22  Street Suffix           5432 non-null    object 
 23  Street Type             305755 non-null  object 
 24  Start_Date_Time         306094 non-null  object 
 25  End_Date_Time           144436 non-null  object 
 26  Latitude                306094 non-null  float64
 27  Longitude               306094 non-null  float64
 28  Police District Number  306094 non-null  object 
 29  Location                306094 non-null  object 
dtypes: float64(4), int64(3), object(23)
memory usage: 70.1+ MB
```

In [21]:
```
### With the data being intially scanned in excel, we saw that there were
```

In [22]:
```
crime_data1 = crime_data.copy()                    # duplicating the crim
```

In [23]:
```
crime_data1
```

Out[23]:

	Incident ID	Offence Code	CR Number	Dispatch Date / Time	NIBRS Code	Victims	Crime Name1	Crim
0	201202980	3550	180042096	08/23/2018 09:52:08 PM	35B	1	Crime Against Society	Drug E
1	201181293	3522	180015424	03/30/2018 01:00:55 AM	35A	1	Crime Against Society	Drug
2	201181293	3562	180015424	03/30/2018 01:00:55 AM	35A	1	Crime Against Society	Drug
3	201193163	3520	180029476	06/14/2018 10:26:45 PM	35A	1	Crime Against Society	Drug
4	201204355	2204	180043926	09/03/2018 12:06:54 PM	220	1	Crime Against Property	Burglary, and
...	...	...	...	...	...	...	...	
306089	201320571	1399	210006666	02/21/2021 07:25:28 AM	13B	1	Crime Against Person	Simpl
306090	201329875	2305	210017101	05/06/2021 10:54:47 PM	23F	1	Crime Against Property	Theft Fr
306091	201329751	5016	210016952	05/05/2021 10:38:58 PM	90Z	1	Crime Against Society	
306092	201323117	9099	210009468	03/13/2021 12:07:14 PM	90Z	1	Crime Against Society	
306093	201327806	2305	210014587	04/19/2021 02:17:29 PM	23F	1	Crime Against Property	Theft Fr

306094 rows × 30 columns

In [24]:
```
crime_data1['City'].value_counts()                    # Getting the count v
```

Out[24]:
```
SILVER SPRING         105487
GAITHERSBURG           44598
ROCKVILLE              42492
GERMANTOWN             30066
BETHESDA               21240
                         ...
FRIENDSHIP HEIGHTS         1
FALLS CHURCH               1
GAITHERSBUG                1
ROCKVIILE                  1
GREENBELT                  1
Name: City, Length: 66, dtype: int64
```

In [25]:
```
# There are few spelling mistakes in the city names in the dataframe, her
city_mapping = {
    'BETHEDA': 'BETHESDA',
    'CLAEKSBURG': 'CLARKSBURG',
    'FREDRICK': 'FREDERICK',
    'GAIHTERSBURG': 'GAITHERSBURG',
    'GAITHERSBUG': 'GAITHERSBURG',
    'ROCKVIILE': 'ROCKVILLE',
    'ROCKVILLLE': 'ROCKVILLE',
    'SILVERS SPRING': 'SILVER SPRING',
    'TACOMA PARK': 'TAKOMA PARK',
    'MT AIRY': 'MOUNT AIRY'
}
```

In [26]:
```
crime_data1['City'].replace(city_mapping, inplace=True)          # Replace
```

In [27]:
```
crime_data1['City'].value_counts()
```

Out[27]:
```
SILVER SPRING         105488
GAITHERSBURG           44600
ROCKVILLE              42495
GERMANTOWN             30066
BETHESDA               21242
MONTGOMERY VILLAGE      9445
TAKOMA PARK             8370
CHEVY CHASE             6410
POTOMAC                 6336
DERWOOD                 5496
KENSINGTON              4824
OLNEY                   4679
BURTONSVILLE            3776
CLARKSBURG              3248
DAMASCUS                2552
BOYDS                   1968
BROOKEVILLE              919
POOLESVILLE              897
ASHTON                   419
SANDY SPRING             392
DICKERSON                292
CABIN JOHN               227
SPENCERVILLE             149
MOUNT AIRY                92
GLEN ECHO                 91
```

```
BRINKLOW                 72
LAUREL                   49
BARNESVILLE              48
BEALLSVILLE              43
WASHINGTON               23
BELTSVILLE               19
GARRETT PARK             16
HYATTSVILLE              16
WASHINGTON GROVE          9
HIGHLAND                  9
WOODBINE                  7
WHEATON                   6
LANHAM                    5
ADELPHI                   4
COLLEGE PARK              2
MCLEAN                    1
FREDERICK                 1
WEST FRIENDSHIP           1
VIENNA                    1
HAGERSTOWN                1
BRENTWOOD                 1
BOWIE                     1
NORTH BETHESDA            1
ALEXANDRIA                1
0                         1
MOUNT RAINIER             1
RIVERDALE                 1
CAPITOL HEIGHTS           1
NORTH POTOMAC             1
FRIENDSHIP HEIGHTS        1
FALLS CHURCH              1
GREENBELT                 1
Name: City, dtype: int64
```

In [28]:
```
crime_data1['City'].value_counts().count()              #get the count of
```

Out[28]:
```
57
```

In [29]:
```
crime_data1 = crime_data1[crime_data1['City'] != '0']    # removeing the
```

In [30]:
```
crime_data1['City'].nunique()                          #get the count of numbe
```

Out[30]:
```
56
```

In [31]:
```
crime_data1['City'].unique()                         #getting the unique citie
```

Out[31]:
```
array(['GERMANTOWN', 'BETHESDA', 'GAITHERSBURG', 'CHEVY CHASE',
       'SILVER SPRING', 'ROCKVILLE', 'TAKOMA PARK', 'POTOMAC',
       'BURTONSVILLE', 'CLARKSBURG', 'OLNEY', 'GLEN ECHO', 'KENSINGTON',
       'BOYDS', 'ASHTON', 'MONTGOMERY VILLAGE', 'DAMASCUS', 'DERWOOD',
       'BEALLSVILLE', 'HYATTSVILLE', 'SANDY SPRING', 'DICKERSON', nan,
       'POOLESVILLE', 'WASHINGTON GROVE', 'BROOKEVILLE', 'CABIN JOHN',
       'MOUNT AIRY', 'GARRETT PARK', 'SPENCERVILLE', 'WASHINGTON',
       'BARNESVILLE', 'BRINKLOW', 'LAUREL', 'BELTSVILLE', 'WHEATON',
       'FALLS CHURCH', 'FRIENDSHIP HEIGHTS', 'COLLEGE PARK',
       'NORTH POTOMAC', 'LANHAM', 'NORTH BETHESDA', 'HIGHLAND',
       'CAPITOL HEIGHTS', 'RIVERDALE', 'MOUNT RAINIER', 'ALEXANDRIA',
       'BRENTWOOD', 'ADELPHI', 'WOODBINE', 'HAGERSTOWN', 'VIENNA',
       'WEST FRIENDSHIP', 'MCLEAN', 'FREDERICK', 'BOWIE', 'GREENBELT'],
      dtype=object)
```

In [32]:
```
crime_data1.loc[crime_data1['City'].isna()]          #checking if there
```

Out[32]:

	Incident ID	Offence Code	CR Number	Dispatch Date / Time	NIBRS Code	Victims	Crime Name1	
358	201334197	2305	210022270	06/11/2021 10:19:54 AM	23F	1	Crime Against Property	Th
398	201330620	2404	210017991	05/13/2021 11:08:22 AM	240	1	Crime Against Property	
905	201332913	2305	210020757	06/01/2021 02:47:05 PM	23F	1	Crime Against Property	Th
1559	201336265	2902	210024766	06/28/2021 12:23:12 PM	290	1	Crime Against Property	Destruct
1896	201318325	2303	210004042	02/01/2021 10:55:28 AM	23C	1	Crime Against Property	
...	...	...	...	...	...	...	...	
305623	201323983	5212	210000015	01/01/2021 01:23:19 AM	520	1	Crime Against Society	
305750	201325005	3615	210011236	03/25/2021 03:38:20 PM	90C	1	Crime Against Society	
305803	201332129	2305	210019731	05/25/2021 12:11:07 PM	23F	1	Crime Against Property	Th
305854	201321030	2404	210007201	02/25/2021 08:50:02 AM	240	1	Crime Against Property	
305923	201326022	2404	210012418	04/03/2021 07:50:16 AM	240	1	Crime Against Property	

1276 rows × 30 columns

In [33]:
```
crime_data1.loc[crime_data1['City'] == 'CLARKSBURG', 'State'] = 'MD'
```

In [34]:
```
crime_data1['State'].unique()                    #getting the unique state
```

Out[34]:
```
array(['MD', 'DC', '16', 'VA', 'ND', '15', 'ME', 'MS', '0', '17'],
      dtype=object)
```

In [35]:
```
# Mapping the abbrivation with the expansion
state_mapping = {
    'MD': 'Maryland',
    'VA': 'Virginia',
    'ME': 'Maine',
    'MS': 'Mississippi',
    'ND': 'North Dakota',
    'DC': 'Washington, D.C.'
}

crime_data1['State'] = crime_data1['State'].replace(state_mapping)
```

```
C:\Users\vishn\AppData\Local\Temp\ipykernel_22508\1456719557.py:11: Setti
ngWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-do
cs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  crime_data1['State'] = crime_data1['State'].replace(state_mapping)
```

In [36]:
```
crime_data1['State'].unique()                #getting the unique state nam
```

Out[36]:
```
array(['Maryland', 'Washington, D.C.', '16', 'Virginia', 'North Dakota',
       '15', 'Maine', 'Mississippi', '0', '17'], dtype=object)
```

In [37]:
```
crime_data1['State'] = np.where((crime_data1['City'] == 'SILVER SPRING')
```

```
C:\Users\vishn\AppData\Local\Temp\ipykernel_22508\3439905036.py:1: Settin
gWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-do
cs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  crime_data1['State'] = np.where((crime_data1['City'] == 'SILVER SPRIN
G') | (crime_data1['State'] == '16') | (crime_data1['State'] == '15'), 'M
aryland', crime_data1['State'])            #replacing the state name as M
aryland for the rows which has state name as "15" or "16" or the city nam
e as"Silver Spring"
```

In [38]:
```
crime_data1['State'].unique()              #getting the unique state names
```

Out[38]:
```
array(['Maryland', 'Washington, D.C.', 'Virginia', 'North Dakota',
       'Maine'], dtype=object)
```

In [39]:
```
# crime_data1.to_csv('Updated.csv', index=False)
```

In [40]:
```
crime_data1['Police District Number'].value_counts()       #getting the v
```

Out[40]:
```
3D       64388
4D       57568
6D       52538
2D       42777
1D       41535
5D       39255
8D        6408
TPPD      1284
OTHER      246
8.0D        82
3.0D         6
6.0D         2
5.0D         2
1.0D         1
4.0D         1
Name: Police District Number, dtype: int64
```

In [41]:
```
#replacing the respective "*.0D" Police District Number to "*D" Police Di
crime_data1.loc[crime_data["Police District Number"] == "8.0D", "Police D
crime_data1.loc[crime_data["Police District Number"] == "3.0D", "Police D
crime_data1.loc[crime_data["Police District Number"] == "6.0D", "Police D
crime_data1.loc[crime_data["Police District Number"] == "5.0D", "Police D
crime_data1.loc[crime_data["Police District Number"] == "1.0D", "Police D
crime_data1.loc[crime_data["Police District Number"] == "4.0D", "Police D
```

In [42]:
```
crime_data1['Police District Number'].value_counts()            #getting
```

Out[42]:
```
3D       64394
4D       57569
6D       52540
2D       42777
1D       41536
5D       39257
8D        6490
TPPD      1284
OTHER      246
Name: Police District Number, dtype: int64
```

In [43]:
```
pd.isnull(crime_data1['Police District Name']).value_counts()    #getting
```

Out[43]:
```
False    305999
True         94
Name: Police District Name, dtype: int64
```

In [44]:
```
#replacing the respective Police District Name to missing values wrt Poli
crime_data1['Police District Name'] = np.where(crime_data1['Police Distri
crime_data1['Police District Name'] = np.where(crime_data1['Police Distri
crime_data1['Police District Name'] = np.where(crime_data1['Police Distri
crime_data1['Police District Name'] = np.where(crime_data1['Police Distri
crime_data1['Police District Name'] = np.where(crime_data1['Police Distri
crime_data1['Police District Name'] = np.where(crime_data1['Police Distri
```

```
C:\Users\vishn\AppData\Local\Temp\ipykernel_22508\1103510619.py:2: Settin
gWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead
```

```
See the caveats in the documentation: https://pandas.pydata.org/pandas-do
cs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  crime_data1['Police District Name'] = np.where(crime_data1['Police Dist
rict Number'] == '8D', 'CITY OF TAKOMA PARK', crime_data1['Police Distric
t Name'])
C:\Users\vishn\AppData\Local\Temp\ipykernel_22508\1103510619.py:3: Settin
gWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-do
cs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  crime_data1['Police District Name'] = np.where(crime_data1['Police Dist
rict Number'] == '6D', 'MONTGOMERY VILLAGE', crime_data1['Police District
Name'])
C:\Users\vishn\AppData\Local\Temp\ipykernel_22508\1103510619.py:4: Settin
gWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-do
cs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  crime_data1['Police District Name'] = np.where(crime_data1['Police Dist
rict Number'] == '5D', 'GERMANTOWN', crime_data1['Police District Name'])
C:\Users\vishn\AppData\Local\Temp\ipykernel_22508\1103510619.py:5: Settin
gWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-do
cs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  crime_data1['Police District Name'] = np.where(crime_data1['Police Dist
rict Number'] == '1D', 'ROCKVILLE', crime_data1['Police District Name'])
C:\Users\vishn\AppData\Local\Temp\ipykernel_22508\1103510619.py:6: Settin
gWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-do
cs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  crime_data1['Police District Name'] = np.where(crime_data1['Police Dist
rict Number'] == '4D', 'WHEATON', crime_data1['Police District Name'])
C:\Users\vishn\AppData\Local\Temp\ipykernel_22508\1103510619.py:7: Settin
gWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-do
cs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  crime_data1['Police District Name'] = np.where(crime_data1['Police Dist
rict Number'] == '3D', 'SILVER SPRING', crime_data1['Police District Nam
e'])
```

In [45]:
```
pd.isnull(crime_data1['Police District Name']).value_counts()        #get
```

Out[45]:
```
False    306093
Name: Police District Name, dtype: int64
```

In [46]:
```
# crime_data1.to_csv('Updated.csv', index=False)
```

In [47]:
```
# Replacing the null values of the city wrt Police District Name
crime_data1['City'] = np.where((crime_data1['Police District Name'] == 'C
crime_data1['City'] = np.where((crime_data1['Police District Name'] == 'T
crime_data1['City'] = np.where((crime_data1['Police District Name'] == 'S
crime_data1['City'] = np.where((crime_data1['Police District Name'] == 'M
crime_data1['City'] = np.where((crime_data1['Police District Name'] == 'O
crime_data1['City'] = np.where((crime_data1['Police District Name'] == 'R
```

```
C:\Users\vishn\AppData\Local\Temp\ipykernel_22508\348072495.py:2: Setting
WithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-do
cs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  crime_data1['City'] = np.where((crime_data1['Police District Name'] ==
'CITY OF TAKOMA PARK') & (crime_data1['City'].isna()), 'TAKOMA PARK', cri
me_data1['City'])
C:\Users\vishn\AppData\Local\Temp\ipykernel_22508\348072495.py:3: Setting
WithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-do
cs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  crime_data1['City'] = np.where((crime_data1['Police District Name'] ==
'TAKOMA PARK') & (crime_data1['City'].isna()), 'TAKOMA PARK', crime_data
1['City'])
C:\Users\vishn\AppData\Local\Temp\ipykernel_22508\348072495.py:4: Setting
WithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-do
cs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  crime_data1['City'] = np.where((crime_data1['Police District Name'] ==
'SILVER SPRING') & (crime_data1['City'].isna()), 'SILVER SPRING', crime_d
ata1['City'])
C:\Users\vishn\AppData\Local\Temp\ipykernel_22508\348072495.py:5: Setting
WithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-do
cs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  crime_data1['City'] = np.where((crime_data1['Police District Name'] ==
'MONTGOMERY VILLAGE') & (crime_data1['City'].isna()), 'GAITHERSBURG', cri
me_data1['City'])
C:\Users\vishn\AppData\Local\Temp\ipykernel_22508\348072495.py:6: Setting
WithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-do
```

```
cs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  crime_data1['City'] = np.where((crime_data1['Police District Name'] ==
'OTHER') & (crime_data1['City'].isna()), 'HYATTSVILLE', crime_data1['Cit
y'])
C:\Users\vishn\AppData\Local\Temp\ipykernel_22508\348072495.py:7: Setting
WithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-do
cs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  crime_data1['City'] = np.where((crime_data1['Police District Name'] ==
'ROCKVILLE') & (crime_data1['City'].isna()), 'ROCKVILLE', crime_data1['Ci
ty'])
```

In [48]:
```
# crime_data1.to_csv('Updated.csv', index=False)
```

In [49]:
```
crime_data1.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 306093 entries, 0 to 306093
Data columns (total 30 columns):
 #   Column                  Non-Null Count   Dtype
---  ------                  --------------   -----
 0   Incident ID             306093 non-null  int64
 1   Offence Code            306093 non-null  object
 2   CR Number               306093 non-null  int64
 3   Dispatch Date / Time    257064 non-null  object
 4   NIBRS Code              306093 non-null  object
 5   Victims                 306093 non-null  int64
 6   Crime Name1             305821 non-null  object
 7   Crime Name2             305821 non-null  object
 8   Crime Name3             305821 non-null  object
 9   Police District Name    306093 non-null  object
 10  Block Address           279888 non-null  object
 11  City                    306093 non-null  object
 12  State                   306093 non-null  object
 13  Zip Code                302914 non-null  float64
 14  Agency                  306093 non-null  object
 15  Place                   306093 non-null  object
 16  Sector                  304563 non-null  object
 17  Beat                    304563 non-null  object
 18  PRA                     305854 non-null  object
 19  Address Number          279985 non-null  float64
 20  Street Prefix           13631 non-null   object
 21  Street Name             306092 non-null  object
 22  Street Suffix           5432 non-null    object
 23  Street Type             305754 non-null  object
 24  Start_Date_Time         306093 non-null  object
 25  End_Date_Time           144435 non-null  object
 26  Latitude                306093 non-null  float64
 27  Longitude               306093 non-null  float64
 28  Police District Number  306093 non-null  object
 29  Location                306093 non-null  object
dtypes: float64(4), int64(3), object(23)
memory usage: 72.4+ MB
```

In [50]:
```
crime_data2 = crime_data1.copy()                                    #Getting the copy
```

In [51]:
```
crime_data2['Start_Date_Time'] = pd.to_datetime(crime_data2['Start_Date_T
```

In [52]:
```
crime_data2['Start_Year'] = (crime_data2['Start_Date_Time'].dt.year)
```

In [53]:
```
crime_data3 = crime_data2[(crime_data2['Start_Year'] >=2018)]           #ne
```

In [54]:
```
crime_data3
```

Out[54]:

	Incident ID	Offence Code	CR Number	Dispatch Date / Time	NIBRS Code	Victims	Crime Name1	Crim
0	201202980	3550	180042096	08/23/2018 09:52:08 PM	35B	1	Crime Against Society	Drug E
1	201181293	3522	180015424	03/30/2018 01:00:55 AM	35A	1	Crime Against Society	Drug
2	201181293	3562	180015424	03/30/2018 01:00:55 AM	35A	1	Crime Against Society	Drug
3	201193163	3520	180029476	06/14/2018 10:26:45 PM	35A	1	Crime Against Society	Drug
4	201204355	2204	180043926	09/03/2018 12:06:54 PM	220	1	Crime Against Property	Burglary, and
...	...	...	...	...	...	...	...	
306089	201320571	1399	210006666	02/21/2021 07:25:28 AM	13B	1	Crime Against Person	Simpl
306090	201329875	2305	210017101	05/06/2021 10:54:47 PM	23F	1	Crime Against Property	Theft Fr
306091	201329751	5016	210016952	05/05/2021 10:38:58 PM	90Z	1	Crime Against Society	
306092	201323117	9099	210009468	03/13/2021 12:07:14 PM	90Z	1	Crime Against Society	
306093	201327806	2305	210014587	04/19/2021 02:17:29 PM	23F	1	Crime Against Property	Theft Fr

221564 rows × 31 columns

In [55]:
```
# Using numpy replace all the 0 in Lat and Long with 'NaN' for easy repla
lat_zero2nan = crime_data3['Latitude'].replace(0, np.nan, inplace=True)
lon_zero2nan = crime_data3['Longitude'].replace(0, np.nan, inplace=True)
```

```
C:\Users\vishn\AppData\Local\Temp\ipykernel_22508\3322506082.py:2: Settin
gWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: https://pandas.pydata.org/pandas-do
cs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  lat_zero2nan = crime_data3['Latitude'].replace(0, np.nan, inplace=True)
C:\Users\vishn\AppData\Local\Temp\ipykernel_22508\3322506082.py:3: Settin
gWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: https://pandas.pydata.org/pandas-do
cs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  lon_zero2nan = crime_data3['Longitude'].replace(0, np.nan, inplace=Tru
e)
```

In [56]:
```
lat_lon_mean = crime_data3.groupby('Zip Code')[['Latitude','Longitude']].
```

In [57]:
```
lat_lon_mean
```

Out[57]:

Zip Code	Latitude	Longitude
20001.0	39.101450	-77.160300
20002.0	NaN	NaN
20004.0	NaN	NaN
20007.0	NaN	NaN
20009.0	NaN	NaN
...	...	...
21797.0	39.277713	-77.141888
22046.0	39.089400	-77.154100
22102.0	NaN	NaN
22181.0	NaN	NaN
22314.0	39.086040	-77.153100

79 rows × 2 columns

In [58]:
```
# Merge the calculated value with zip code and suffix with name '_mean'

crime_data4 = crime_data3.merge(lat_lon_mean, left_on = 'Zip Code', right
```

In [59]:
```
crime_data4
```

Out[59]:

	Incident ID	Offence Code	CR Number	Dispatch Date / Time	NIBRS Code	Victims	Crime Name1	Crim
0	201202980	3550	180042096	08/23/2018 09:52:08 PM	35B	1	Crime Against Society	Drug E
1	201181293	3522	180015424	03/30/2018 01:00:55 AM	35A	1	Crime Against Society	Drug
2	201181293	3562	180015424	03/30/2018 01:00:55 AM	35A	1	Crime Against Society	Drug
3	201193163	3520	180029476	06/14/2018 10:26:45 PM	35A	1	Crime Against Society	Drug
4	201204355	2204	180043926	09/03/2018 12:06:54 PM	220	1	Crime Against Property	Burglary, and
...	...	...	...	...	...	...	...	
306089	201320571	1399	210006666	02/21/2021 07:25:28 AM	13B	1	Crime Against Person	Simpl
306090	201329875	2305	210017101	05/06/2021 10:54:47 PM	23F	1	Crime Against Property	Theft Fr
306091	201329751	5016	210016952	05/05/2021 10:38:58 PM	90Z	1	Crime Against Society	
306092	201323117	9099	210009468	03/13/2021 12:07:14 PM	90Z	1	Crime Against Society	
306093	201327806	2305	210014587	04/19/2021 02:17:29 PM	23F	1	Crime Against Property	Theft Fr

221564 rows × 33 columns

In [60]:
```
(crime_data3['Latitude']).isna().sum()        #getting the sum of rows whi
```

Out[60]:
```
7084
```

In [61]:
```
#filling the Latitude, Longitude with the calculated mean
crime_data4['Latitude'].fillna(crime_data4['Latitude_mean'], inplace = Tr
crime_data4['Longitude'].fillna(crime_data4['Longitude_mean'], inplace =
```

In [62]:
```
crime_data4.drop(['Latitude_mean','Longitude_mean'], axis = 1, inplace =
```

In [63]:
```
#Before repacing 0 with mean value

(crime_data3['Latitude']).isna().sum()
```

Out[63]:
```
7084
```

In [64]:
```
#After repacing 0 with mean value

(crime_data4['Latitude']).isna().sum()
```

Out[64]:
```
21
```

In [65]:
```
#replacing the Latitude, Longitude null values with 0
crime_data4['Latitude'].fillna(0, inplace=True)
crime_data4['Longitude'].fillna(0, inplace=True)
```

In [66]:
```
# To display all the values of '0' which cannot be replaced with mean val

crime_data4[(crime_data4['Latitude'] == 0) & (crime_data4['Longitude'] ==
```

Out[66]:

	Zip Code	Latitude	Longitude
2632	20007.0	0.0	0.0
24255	20010.0	0.0	0.0
25279	20010.0	0.0	0.0
26593	20002.0	0.0	0.0
30259	20743.0	0.0	0.0
34265	20737.0	0.0	0.0
36943	20712.0	0.0	0.0
43967	20007.0	0.0	0.0
65142	21740.0	0.0	0.0
84925	20016.0	0.0	0.0
96594	22181.0	0.0	0.0
97852	20032.0	0.0	0.0
98476	21794.0	0.0	0.0
101093	22102.0	0.0	0.0
102571	20004.0	0.0	0.0
107152	20010.0	0.0	0.0
186303	20009.0	0.0	0.0
190279	20706.0	0.0	0.0
193068	20016.0	0.0	0.0
204289	20770.0	0.0	0.0
211429	20010.0	0.0	0.0

In [67]:
```
crime_data4[crime_data2['Zip Code'].isna()]    #getting the numebr of row
```

```
C:\Users\vishn\AppData\Local\Temp\ipykernel_22508\1834070977.py:1: UserWa
rning: Boolean Series key will be reindexed to match DataFrame index.
  crime_data4[crime_data2['Zip Code'].isna()]    #getting the numebr of r
ows for null values in zip code
```

Out[67]:

	Incident ID	Offence Code	CR Number	Dispatch Date / Time	NIBRS Code	Victims	Crime Name1	
5957	201234818	2305	190017054	04/12/2019 11:09:24 AM	23F	1	Crime Against Property	Tl
5959	201237151	2999	190019926	04/28/2019 03:23:59 PM	290	1	Crime Against Property	Destruc

5975	201308722	2203	200044158	11/09/2020 08:55:36 AM	220	1	Crime Against Property	Burglar
5976	201308722	2901	200044158	11/09/2020 08:55:36 AM	290	1	Crime Against Property	Destruc
5977	201309459	2304	200045092	11/15/2020 03:03:19 PM	23G	1	Crime Against Property	Theft o
5978	201311642	2305	200047747	12/04/2020 06:39:53 AM	23F	1	Crime Against Property	Tl
5979	201318750	2404	210004526	02/04/2021 06:23:00 PM	240	1	Crime Against Property	
7344	201337196	1399	210025985	07/06/2021 07:39:03 PM	13B	1	Crime Against Person	
7345	201337196	2902	210025985	07/06/2021 07:39:03 PM	290	1	Crime Against Property	Destruc
13172	201338365	2305	210027179	07/14/2021 02:50:35 PM	23F	1	Crime Against Property	Tl
18041	201340040	2304	210028904	07/26/2021 10:33:05 AM	23G	1	Crime Against Property	Theft o
29770	201366575	9021	220009013	03/02/2022 11:55:00 AM	90Z	1	Not a Crime	
49399	201375969	1315	220020290	05/12/2022 11:10:09 PM	13A	2	Crime Against Person	
85092	201248843	9021	190034524	07/20/2019 02:15:44 PM	90Z	1	Not a Crime	
87770	201254602	9199	190041477	08/30/2019 01:59:28 AM	90Z	1	Other	
91755	201252930	9101	190039457	08/18/2019 02:22:21 PM	90Z	1	Other	
93731	201327647	9107	210014396	04/17/2021 08:55:50 PM	90Z	1	Other	
95037	201327647	9021	210014396	04/17/2021 08:55:50 PM	90Z	1	Not a Crime	

102600	201327565	9113	210014304	04/17/2021 12:21:39 AM	90Z	1	Other	
109644	201386461	9113	220032921	07/31/2022 09:00:50 PM	90Z	1	Other	
196090	201197926	3562	180035857	07/19/2018 11:21:06 PM	35A	1	Crime Against Society	D
201068	201192698	3504	180019359	04/20/2018 01:56:32 PM	35A	1	Crime Against Society	D
206586	201215852	2304	180058679	11/23/2018 05:15:39 PM	23G	1	Crime Against Property	Theft o
211055	201204481	2305	180044061	09/04/2018 09:22:39 AM	23F	1	Crime Against Property	Tl
238537	201249335	2305	190035126	07/24/2019 07:52:35 AM	23F	1	Crime Against Property	Tl
251550	201252022	2602	190038409	08/12/2019 10:54:10 AM	26A	1	Crime Against Property	Pretens
259100	201283935	2304	200014872	04/03/2020 09:33:42 AM	23G	1	Crime Against Property	Theft o
260838	201281291	4199	200011861	03/10/2020 07:24:50 PM	90G	1	Crime Against Society	
261654	201279413	2299	200009491	02/26/2020 06:35:35 AM	220	1	Crime Against Property	Burglar
263393	201278603	5311	200008564	02/20/2020 06:05:14 PM	90C	1	Crime Against Society	
264274	201277226	2308	200006820	02/10/2020 05:35:46 PM	23D	1	Crime Against Property	
266139	201279343	1315	200009450	02/25/2020 08:59:22 PM	13A	1	Crime Against Person	
277450	201298178	2205	200031487	08/13/2020 08:36:09 AM	220	1	Crime Against Property	Burglar
298292	201327820	2203	210014570	04/19/2021 11:52:30 AM	220	1	Crime Against Property	Burglar

299185	201327560	1399	210014300	04/17/2021 12:03:33 AM	13B	1	Crime Against Person	
299573	201328681	2404	210015620	04/27/2021 06:04:54 AM	240	1	Crime Against Property	
299986	201327820	2901	210014570	04/19/2021 11:52:30 AM	290	1	Crime Against Property	Destruc
301346	201330754	2404	210018178	05/14/2021 03:33:17 PM	240	1	Crime Against Property	
301703	201328681	2305	210015620	04/27/2021 06:04:54 AM	23F	1	Crime Against Property	Tl
305233	201322100	4104	210007734	03/01/2021 01:15:01 PM	90G	1	Crime Against Society	

40 rows × 31 columns

In [68]:
```
#crime_data2['Zip Code'] = (crime_data2.groupby('Street Name')['Zip Code'
```

In [69]:
```
crime_data4.loc[crime_data2['Zip Code'] == 20007]
```

Out[69]:

	Incident ID	Offence Code	CR Number	Dispatch Date / Time	NIBRS Code	Victims	Crime Name1	Crime Name2
2632	201343436	2404	210033807	08/30/2021 06:41:43 AM	240	1	Crime Against Property	Motor Vehicle Theft
43967	201366630	9108	220009125	03/02/2022 10:04:25 PM	90Z	1	Other	All Other Offenses

2 rows × 31 columns

In [70]:
```
#As there was not zipcode to any of the rows with the street name ZANHORN
```

In [71]:
```
crime_data4.loc[crime_data4['Street Name'] == 'ZANHORN', 'Zip Code'] = 20
```

In [72]:
```
crime_data4.loc[crime_data4['Street Name'] == 'MAURY', 'Zip Code'] = 3840
```

In [73]:
```
crime_data4.loc[crime_data4['Street Name'] == 'SOUTH BELGRADE', 'Zip Code
```

In [74]:
```
temp_data = crime_data4[crime_data4['Zip Code'].isna()]          #gettin
```

In [75]:
```
temp_data
```

Out[75]:

	Incident ID	Offence Code	CR Number	Dispatch Date / Time	NIBRS Code	Victims	Crime Name1	
5957	201234818	2305	190017054	04/12/2019 11:09:24 AM	23F	1	Crime Against Property	Th
5979	201318750	2404	210004526	02/04/2021 06:23:00 PM	240	1	Crime Against Property	
13172	201338365	2305	210027179	07/14/2021 02:50:35 PM	23F	1	Crime Against Property	Th
18041	201340040	2304	210028904	07/26/2021 10:33:05 AM	23G	1	Crime Against Property	Theft
29770	201366575	9021	220009013	03/02/2022 11:55:00 AM	90Z	1	Not a Crime	
49399	201375969	1315	220020290	05/12/2022 11:10:09 PM	13A	2	Crime Against Person	
85092	201248843	9021	190034524	07/20/2019 02:15:44 PM	90Z	1	Not a Crime	
87770	201254602	9199	190041477	08/30/2019 01:59:28 AM	90Z	1	Other	
93731	201327647	9107	210014396	04/17/2021 08:55:50 PM	90Z	1	Other	
95037	201327647	9021	210014396	04/17/2021 08:55:50 PM	90Z	1	Not a Crime	
102600	201327565	9113	210014304	04/17/2021 12:21:39 AM	90Z	1	Other	
109644	201386461	9113	220032921	07/31/2022 09:00:50 PM	90Z	1	Other	
196090	201197926	3562	180035857	07/19/2018 11:21:06 PM	35A	1	Crime Against Society	D
201068	201192698	3504	180019359	04/20/2018 01:56:32 PM	35A	1	Crime Against Society	D
				07/24/2019			Crime	

238537	201249335	2305	190035126	07:52:35 AM	23F	1	Against Property	Th
251550	201252022	2602	190038409	08/12/2019 10:54:10 AM	26A	1	Crime Against Property	Pretens
259100	201283935	2304	200014872	04/03/2020 09:33:42 AM	23G	1	Crime Against Property	Theft
260838	201281291	4199	200011861	03/10/2020 07:24:50 PM	90G	1	Crime Against Society	
261654	201279413	2299	200009491	02/26/2020 06:35:35 AM	220	1	Crime Against Property	
263393	201278603	5311	200008564	02/20/2020 06:05:14 PM	90C	1	Crime Against Society	
264274	201277226	2308	200006820	02/10/2020 05:35:46 PM	23D	1	Crime Against Property	
266139	201279343	1315	200009450	02/25/2020 08:59:22 PM	13A	1	Crime Against Person	
277450	201298178	2205	200031487	08/13/2020 08:36:09 AM	220	1	Crime Against Property	
299185	201327560	1399	210014300	04/17/2021 12:03:33 AM	13B	1	Crime Against Person	
305233	201322100	4104	210007734	03/01/2021 01:15:01 PM	90G	1	Crime Against Society	

25 rows × 31 columns

In [76]:
```
temp_data['Street Name'].value_counts()
```

Out[76]:
```
WOODFIELD          5
GEORGIA            4
FLOWER HILL        2
CLARIDGE           1
VEIRS MILL         1
GATE               1
PARKLAND           1
GRANADA            1
TYGART             1
UNIVERSITY         1
ATHERTON           1
COLONY             1
NORTHAMPTON        1
ROCKVILLE          1
DALE               1
TRAVILAH CREST     1
FLOWER             1
Name: Street Name, dtype: int64
```

In [77]:
```
#mapping "ZANHORN" with "VAN HORN" as there is no street name with the na

street_mapping = {
    'ZANHORN': 'VAN HORN'
}

crime_data4['Street Name'] = crime_data4['Street Name'].replace(street_ma
```

In [78]:
```
crime_data4.loc[crime_data4['Zip Code'] == 20866]
```

Out[78]:

	Incident ID	Offence Code	CR Number	Dispatch Date / Time	NIBRS Code	Victims	Crime Name1	
54	201346569	9113	210037794	09/24/2021 12:12:58 PM	90Z	1	Other	
180	201341136	9199	210031007	08/09/2021 07:54:08 PM	90Z	1	Other	
199	201256295	2399	190043137	09/09/2019 09:31:02 AM	23H	1	Crime Against Property	
220	201276332	2505	200004780	01/29/2020 01:32:38 PM	250	1	Crime Against Property	(
534	201356028	5404	210049611	12/06/2021 09:23:17 PM	90D	1	Crime Against Society	Drivi
...	...	...	...	...	...	...	...	
305720	201331500	2610	210019057	05/20/2021 02:50:52 PM	26F	1	Crime Against Property	
305811	201325908	2304	210012285	04/02/2021 07:54:18 AM	23G	1	Crime Against Property	Theft
305935	201329581	2610	210016768	05/04/2021 05:22:57 PM	26F	1	Crime Against Property	
305957	201322111	2601	210008367	03/05/2021 03:36:35 PM	26A	1	Crime Against Property	Pretens
306004	201323483	2610	210009871	03/16/2021 03:56:57 PM	26F	1	Crime Against Property	

2761 rows × 31 columns

In [79]:
```
mode1 = crime_data4.loc[crime_data4['Street Name'] == 'FLOWER HILL']['Zip
print(mode1)
```

```
0    20879.0
Name: Zip Code, dtype: float64
```

In [80]:
```
# Replacig the more of zip code wrt street name where zip code is null
crime_data4['Zip Code'] = np.where((crime_data4['Zip Code'].isna())&(crim
crime_data4['Zip Code'] = np.where((crime_data4['Zip Code'].isna())&(crim
crime_data4['Zip Code'] = np.where((crime_data4['Zip Code'].isna())&(crim
crime_data4['Zip Code'] = np.where((crime_data4['Zip Code'].isna())&(crim
crime_data4['Zip Code'] = np.where((crime_data4['Zip Code'].isna())&(crim
crime_data4['Zip Code'] = np.where((crime_data4['Zip Code'].isna())&(crim
crime_data4['Zip Code'] = np.where((crime_data4['Zip Code'].isna())&(crim
crime_data4['Zip Code'] = np.where((crime_data4['Zip Code'].isna())&(crim
crime_data4['Zip Code'] = np.where((crime_data4['Zip Code'].isna())&(crim
crime_data4['Zip Code'] = np.where((crime_data4['Zip Code'].isna())&(crim
crime_data4['Zip Code'] = np.where((crime_data4['Zip Code'].isna())&(crim
crime_data4['Zip Code'] = np.where((crime_data4['Zip Code'].isna())&(crim
crime_data4['Zip Code'] = np.where((crime_data4['Zip Code'].isna())&(crim
```

In [81]:
```
crime_data4 = crime_data4.drop(crime_data4[crime_data4['Zip Code'].isna()
```

In [82]:
```
crime_data4.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 221553 entries, 0 to 306093
Data columns (total 31 columns):
 #   Column                  Non-Null Count   Dtype
---  ------                  --------------   -----
 0   Incident ID             221553 non-null  int64
 1   Offence Code            221553 non-null  object
 2   CR Number               221553 non-null  int64
 3   Dispatch Date / Time    214713 non-null  object
 4   NIBRS Code              221553 non-null  object
 5   Victims                 221553 non-null  int64
 6   Crime Name1             221294 non-null  object
 7   Crime Name2             221294 non-null  object
 8   Crime Name3             221294 non-null  object
 9   Police District Name    221553 non-null  object
 10  Block Address           200347 non-null  object
 11  City                    221553 non-null  object
 12  State                   221553 non-null  object
 13  Zip Code                221553 non-null  float64
 14  Agency                  221553 non-null  object
 15  Place                   221553 non-null  object
 16  Sector                  220042 non-null  object
 17  Beat                    220042 non-null  object
 18  PRA                     221322 non-null  object
 19  Address Number          200425 non-null  float64
 20  Street Prefix           10156 non-null   object
 21  Street Name             221552 non-null  object
 22  Street Suffix           3814 non-null    object
 23  Street Type             221464 non-null  object
 24  Start_Date_Time         221553 non-null  datetime64[ns]
 25  End_Date_Time           100546 non-null  object
 26  Latitude                221553 non-null  float64
 27  Longitude               221553 non-null  float64
 28  Police District Number  221553 non-null  object
 29  Location                221553 non-null  object
 30  Start_Year              221553 non-null  int64
dtypes: datetime64[ns](1), float64(4), int64(4), object(22)
memory usage: 54.1+ MB
```

In [83]:
```
crime_data5 = crime_data4.copy()          # duplicating the crime_data4
```

In [84]:
```
#Crime name 1,2 & 3 columns with the NIBRS code 26G were all null, by usi
crime_data5['Crime Name1'] = np.where(crime_data5['NIBRS Code'] == '26G',
crime_data5['Crime Name2'] = np.where(crime_data5['NIBRS Code'] == '26G',
crime_data5['Crime Name3'] = np.where(crime_data5['NIBRS Code'] == '26G',
```

In [85]:
```
#Crime name 1,2 & 3 columns with the NIBRS code 90Z are null, we have fil

crime_data5['Crime Name1'] = np.where((crime_data5['NIBRS Code'] == '90Z'
crime_data5['Crime Name2'] = np.where((crime_data5['NIBRS Code'] == '90Z'
crime_data5['Crime Name3'] = np.where((crime_data5['NIBRS Code'] == '90Z'
```

In [86]:
```
#Crime name 1,2 & 3 columns with the NIBRS code 35A are null, we have fil

crime_data5['Crime Name1'] = np.where((crime_data5['NIBRS Code'] == '35A'
crime_data5['Crime Name2'] = np.where((crime_data5['NIBRS Code'] == '35A'
crime_data5['Crime Name3'] = np.where((crime_data5['NIBRS Code'] == '35A'
```

In [87]:
```
crime_data5.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 221553 entries, 0 to 306093
Data columns (total 31 columns):
 #   Column                  Non-Null Count   Dtype         
---  ------                  --------------   -----         
 0   Incident ID             221553 non-null  int64         
 1   Offence Code            221553 non-null  object        
 2   CR Number               221553 non-null  int64         
 3   Dispatch Date / Time    214713 non-null  object        
 4   NIBRS Code              221553 non-null  object        
 5   Victims                 221553 non-null  int64         
 6   Crime Name1             221553 non-null  object        
 7   Crime Name2             221553 non-null  object        
 8   Crime Name3             221553 non-null  object        
 9   Police District Name    221553 non-null  object        
 10  Block Address           200347 non-null  object        
 11  City                    221553 non-null  object        
 12  State                   221553 non-null  object        
 13  Zip Code                221553 non-null  float64       
 14  Agency                  221553 non-null  object        
 15  Place                   221553 non-null  object        
 16  Sector                  220042 non-null  object        
 17  Beat                    220042 non-null  object        
 18  PRA                     221322 non-null  object        
 19  Address Number          200425 non-null  float64       
 20  Street Prefix           10156 non-null   object        
 21  Street Name             221552 non-null  object        
 22  Street Suffix           3814 non-null    object        
 23  Street Type             221464 non-null  object        
 24  Start_Date_Time         221553 non-null  datetime64[ns]
 25  End_Date_Time           100546 non-null  object        
 26  Latitude                221553 non-null  float64       
 27  Longitude               221553 non-null  float64       
 28  Police District Number  221553 non-null  object        
 29  Location                221553 non-null  object        
 30  Start_Year              221553 non-null  int64         
dtypes: datetime64[ns](1), float64(4), int64(4), object(22)
memory usage: 54.1+ MB
```

In [88]:
```
# we are getting only the rows with latitude and longitude being zero
crime_data5[(crime_data5['Latitude'] == 0.0) & (crime_data5['Longitude']
```

Out[88]:

	Zip Code	Street Name	Latitude	Longitude
2632	20007.0	MASSACHUSETTS	0.0	0.0
24255	20010.0	IRVING	0.0	0.0
25279	20010.0	OTIS	0.0	0.0
26593	20002.0	6TH	0.0	0.0
30259	20743.0	MARLBORO	0.0	0.0
34265	20737.0	BALTIMORE	0.0	0.0
36943	20712.0	KAYWOOD	0.0	0.0
43967	20007.0	WISCONSIN	0.0	0.0
65142	21740.0	SALEM	0.0	0.0
84925	20016.0	BRANDYWINE	0.0	0.0
96594	22181.0	COURTHOUSE	0.0	0.0
97852	20032.0	LIVINGSTON	0.0	0.0
98476	21794.0	NEW ROVER	0.0	0.0
101093	22102.0	INTERNATIONAL	0.0	0.0
102571	20004.0	CONSTITUTION	0.0	0.0
107152	20010.0	IRVING	0.0	0.0
186303	20009.0	CONNECTICUT	0.0	0.0
190279	20706.0	ANNAPOLIS	0.0	0.0
193068	20016.0	WISCONSIN	0.0	0.0
204289	20770.0	SPRINGHILL	0.0	0.0
211429	20010.0	IRVING	0.0	0.0

In [89]:
```
# here we are calculating the latitude and longitudes which are zero, we
crime_data5.loc[(crime_data5['Location'] == '(0.0, 0.0)') & (crime_data5[
crime_data5.loc[(crime_data5['Location'] == '(0.0, 0.0)') & (crime_data5[
crime_data5.loc[(crime_data5['Location'] == '(0.0, 0.0)') & (crime_data5[
crime_data5.loc[(crime_data5['Location'] == '(0.0, 0.0)') & (crime_data5[
crime_data5.loc[(crime_data5['Location'] == '(0.0, 0.0)') & (crime_data5[
crime_data5.loc[(crime_data5['Location'] == '(0.0, 0.0)') & (crime_data5[
crime_data5.loc[(crime_data5['Location'] == '(0.0, 0.0)') & (crime_data5[
crime_data5.loc[(crime_data5['Location'] == '(0.0, 0.0)') & (crime_data5[
crime_data5.loc[(crime_data5['Location'] == '(0.0, 0.0)') & (crime_data5[
crime_data5.loc[(crime_data5['Location'] == '(0.0, 0.0)') & (crime_data5[
crime_data5.loc[(crime_data5['Location'] == '(0.0, 0.0)') & (crime_data5[
crime_data5.loc[(crime_data5['Location'] == '(0.0, 0.0)') & (crime_data5[
crime_data5.loc[(crime_data5['Location'] == '(0.0, 0.0)') & (crime_data5[
crime_data5.loc[(crime_data5['Location'] == '(0.0, 0.0)') & (crime_data5[
```

In [90]:
```
# we are checking for those modes of latitudes and longitudes which still
crime_data5[(crime_data5['Latitude'] == 0.0) & (crime_data5['Longitude']
```

Out[90]:

	Zip Code	Street Name	Latitude	Longitude
24255	20010.0	IRVING	0.0	0.0
98476	21794.0	NEW ROVER	0.0	0.0
102571	20004.0	CONSTITUTION	0.0	0.0
107152	20010.0	IRVING	0.0	0.0
190279	20706.0	ANNAPOLIS	0.0	0.0
204289	20770.0	SPRINGHILL	0.0	0.0
211429	20010.0	IRVING	0.0	0.0

In [91]:
```
crime_data5.loc[(crime_data5['Latitude'] == 0.0) & (crime_data5['Longitud
```

Out[91]:

	Incident ID	Offence Code	CR Number	Dispatch Date / Time	NIBRS Code	Victims	Crime Name1	C Na
24255	201354380	9105	210046789	11/18/2021 10:27:56 AM	90Z	1	Other	All Offe
98476	201317755	9105	210003262	NaN	90Z	1	Other	All Offe
102571	201376386	9199	220020786	05/16/2022 09:47:52 AM	90Z	1	Other	All Offe
107152	201380105	9199	220025278	06/12/2022 09:16:12 PM	90Z	1	Other	All Offe
190279	201180843	2610	180014861	03/26/2018 05:06:22 PM	26F	1	Crime Against Property	Identity
204289	201195006	2399	180032338	06/30/2018 07:44:07 PM	23H	1	Crime Against Property	All La
211429	201206353	3532	180046489	09/16/2018 08:53:04 PM	35A	1	Crime Against Society	Drug/Na Viola

7 rows × 31 columns

In [92]:
```
crime_data5.shape
```

Out[92]:
```
(221553, 31)
```

In [93]:
```
# removing rows which have latitude and longitude as the extreme values(0

crime_data6 = crime_data5[(crime_data5['Latitude']!=0.0) & (crime_data5['

crime_data6 = crime_data6[crime_data6['Latitude']<90.0]
```

In [94]:
```
# this is the final data which is being displayed here with all the colum
crime_data6
```

Out[94]:

	Incident ID	Offence Code	CR Number	Dispatch Date / Time	NIBRS Code	Victims	Crime Name1	Crim
0	201202980	3550	180042096	08/23/2018 09:52:08 PM	35B	1	Crime Against Society	Drug E
1	201181293	3522	180015424	03/30/2018 01:00:55 AM	35A	1	Crime Against Society	Drug
2	201181293	3562	180015424	03/30/2018 01:00:55 AM	35A	1	Crime Against Society	Drug
3	201193163	3520	180029476	06/14/2018 10:26:45 PM	35A	1	Crime Against Society	Drug
4	201204355	2204	180043926	09/03/2018 12:06:54 PM	220	1	Crime Against Property	Burglary, and
...	...	...	...	...	...	...	...	
306089	201320571	1399	210006666	02/21/2021 07:25:28 AM	13B	1	Crime Against Person	Simpl
306090	201329875	2305	210017101	05/06/2021 10:54:47 PM	23F	1	Crime Against Property	Theft Fr
306091	201329751	5016	210016952	05/05/2021 10:38:58 PM	90Z	1	Crime Against Society	
306092	201323117	9099	210009468	03/13/2021 12:07:14 PM	90Z	1	Crime Against Society	
306093	201327806	2305	210014587	04/19/2021 02:17:29 PM	23F	1	Crime Against Property	Theft Fr

221542 rows × 31 columns

In [95]:
```
crime_data6.columns
```

Out[95]:
```
Index(['Incident ID', 'Offence Code', 'CR Number', 'Dispatch Date / Tim
e',
       'NIBRS Code', 'Victims', 'Crime Name1', 'Crime Name2', 'Crime Name
3',
       'Police District Name', 'Block Address', 'City', 'State', 'Zip Cod
e',
       'Agency', 'Place', 'Sector', 'Beat', 'PRA', 'Address Number',
       'Street Prefix', 'Street Name', 'Street Suffix', 'Street Type',
       'Start_Date_Time', 'End_Date_Time', 'Latitude', 'Longitude',
       'Police District Number', 'Location', 'Start_Year'],
      dtype='object')
```

In [96]:
```
#dropping those columns which we don't need for the visualisations
crime_data6.drop(['Street Suffix', 'Street Prefix', 'Location', 'Block Ad
```

In [97]:
```
# here we have calculated the day name for the particular date
crime_data6['day_name'] = pd.to_datetime(crime_data6['Start_Date_Time']).
crime_data6['Month'] = (crime_data6['Start_Date_Time'].dt.month)
crime_data6['Day'] = crime_data6['Start_Date_Time'].dt.day

crime_data6['Time_Hour'] = crime_data6['Start_Date_Time'].dt.hour
```

In [98]:
```
crime_data6.head()
```

Out[98]:

	Incident ID	Offence Code	CR Number	Dispatch Date / Time	NIBRS Code	Victims	Crime Name1	Crime Name
0	201202980	3550	180042096	08/23/2018 09:52:08 PM	35B	1	Crime Against Society	Drug Equipme Violatio
1	201181293	3522	180015424	03/30/2018 01:00:55 AM	35A	1	Crime Against Society	Drug/Narcot Violatio
2	201181293	3562	180015424	03/30/2018 01:00:55 AM	35A	1	Crime Against Society	Drug/Narcot Violatio
3	201193163	3520	180029476	06/14/2018 10:26:45 PM	35A	1	Crime Against Society	Drug/Narcot Violatio
4	201204355	2204	180043926	09/03/2018 12:06:54 PM	220	1	Crime Against Property	Burglary/Breaki and Enteri

5 rows × 31 columns

In [99]:
```
def time_def(df):
    for index, row in df.iterrows():
        if 4 < row['Time_Hour'] <= 12:
            df.loc[index, 'Time_of_the_day'] = 'Morning'
        elif 12 < row['Time_Hour'] <= 15:
            df.loc[index, 'Time_of_the_day'] = 'Afternoon'
        elif 15 < row['Time_Hour'] <= 19:
            df.loc[index, 'Time_of_the_day'] = 'Evening'
        elif 19 < row['Time_Hour'] <= 23:
            df.loc[index, 'Time_of_the_day'] = 'Night'
        elif 0 <= row['Time_Hour'] <= 4:
            df.loc[index, 'Time_of_the_day'] = 'Midnight'
    return df
```

In [100…
```
crime_data6 = time_def(crime_data6)
```

In [101…
```
crime_data6.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 221542 entries, 0 to 306093
Data columns (total 32 columns):
 #   Column                  Non-Null Count   Dtype         
---  ------                  --------------   -----         
 0   Incident ID             221542 non-null  int64         
 1   Offence Code            221542 non-null  object        
 2   CR Number               221542 non-null  int64         
 3   Dispatch Date / Time    214703 non-null  object        
 4   NIBRS Code              221542 non-null  object        
 5   Victims                 221542 non-null  int64         
 6   Crime Name1             221542 non-null  object        
 7   Crime Name2             221542 non-null  object        
 8   Crime Name3             221542 non-null  object        
 9   Police District Name    221542 non-null  object        
 10  City                    221542 non-null  object        
 11  State                   221542 non-null  object        
 12  Zip Code                221542 non-null  float64       
 13  Agency                  221542 non-null  object        
 14  Place                   221542 non-null  object        
 15  Sector                  220037 non-null  object        
 16  Beat                    220037 non-null  object        
 17  PRA                     221317 non-null  object        
 18  Address Number          200414 non-null  float64       
 19  Street Name             221542 non-null  object        
 20  Street Type             221454 non-null  object        
 21  Start_Date_Time         221542 non-null  datetime64[ns]
 22  End_Date_Time           100542 non-null  object        
 23  Latitude                221542 non-null  float64       
 24  Longitude               221542 non-null  float64       
 25  Police District Number  221542 non-null  object        
 26  Start_Year              221542 non-null  int64         
 27  day_name                221542 non-null  object        
 28  Month                   221542 non-null  int64         
 29  Day                     221542 non-null  int64         
 30  Time_Hour               221542 non-null  int64         
 31  Time_of_the_day         221542 non-null  object        
dtypes: datetime64[ns](1), float64(4), int64(7), object(20)
memory usage: 63.8+ MB
```

In [102…
```
# saving the file to use the same data for visualisations
#crime_data6.to_csv('final_data_no_changes.csv', index=False)
```

In [103…
```
crime = crime_data6.copy()
```

In [104…
```
crime.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 221542 entries, 0 to 306093
Data columns (total 32 columns):
 #   Column                  Non-Null Count   Dtype         
---  ------                  --------------   -----         
 0   Incident ID             221542 non-null  int64         
 1   Offence Code            221542 non-null  object        
 2   CR Number               221542 non-null  int64         
 3   Dispatch Date / Time    214703 non-null  object        
 4   NIBRS Code              221542 non-null  object        
 5   Victims                 221542 non-null  int64         
 6   Crime Name1             221542 non-null  object        
 7   Crime Name2             221542 non-null  object        
 8   Crime Name3             221542 non-null  object        
 9   Police District Name    221542 non-null  object        
 10  City                    221542 non-null  object        
 11  State                   221542 non-null  object        
 12  Zip Code                221542 non-null  float64       
 13  Agency                  221542 non-null  object        
 14  Place                   221542 non-null  object        
 15  Sector                  220037 non-null  object        
 16  Beat                    220037 non-null  object        
 17  PRA                     221317 non-null  object        
 18  Address Number          200414 non-null  float64       
 19  Street Name             221542 non-null  object        
 20  Street Type             221454 non-null  object        
 21  Start_Date_Time         221542 non-null  datetime64[ns]
 22  End_Date_Time           100542 non-null  object        
 23  Latitude                221542 non-null  float64       
 24  Longitude               221542 non-null  float64       
 25  Police District Number  221542 non-null  object        
 26  Start_Year              221542 non-null  int64         
 27  day_name                221542 non-null  object        
 28  Month                   221542 non-null  int64         
 29  Day                     221542 non-null  int64         
 30  Time_Hour               221542 non-null  int64         
 31  Time_of_the_day         221542 non-null  object        
dtypes: datetime64[ns](1), float64(4), int64(7), object(20)
memory usage: 63.8+ MB
```

In [105…

```
crime.head()
```

Out[105]:

	Incident ID	Offence Code	CR Number	Dispatch Date / Time	NIBRS Code	Victims	Crime Name1	Crime Nam
0	201202980	3550	180042096	08/23/2018 09:52:08 PM	35B	1	Crime Against Society	Drug Equipm Violati
1	201181293	3522	180015424	03/30/2018 01:00:55 AM	35A	1	Crime Against Society	Drug/Narc Violati
2	201181293	3562	180015424	03/30/2018 01:00:55 AM	35A	1	Crime Against Society	Drug/Narc Violati
3	201193163	3520	180029476	06/14/2018 10:26:45 PM	35A	1	Crime Against Society	Drug/Narc Violati
4	201204355	2204	180043926	09/03/2018 12:06:54 PM	220	1	Crime Against Property	Burglary/Break and Enter

5 rows × 32 columns

descriptive stats

In [106…

```
crime.describe()
```

Out[106]:

	Incident ID	CR Number	Victims	Zip Code	Address Number	
count	2.215420e+05	2.215420e+05	221542.000000	221542.000000	2.004140e+05	221
mean	2.012778e+08	1.973929e+08	1.023752	20877.149462	8.395592e+03	
std	6.288905e+04	1.373460e+07	0.191766	62.993771	1.600284e+04	
min	2.011689e+08	1.202759e+07	1.000000	20001.000000	1.000000e+00	
25%	2.012233e+08	1.900024e+08	1.000000	20853.000000	1.600000e+03	
50%	2.012780e+08	2.000074e+08	1.000000	20878.000000	8.100000e+03	
75%	2.013320e+08	2.100194e+08	1.000000	20904.000000	1.250000e+04	
max	2.013872e+08	2.204211e+08	12.000000	38401.000000	2.090600e+06	

The above table shows the statistics for the dataframe given and it includes only those columns that are integer or float type data.

the columns that are object type, we can have mode calculated for the required columns that we use in the dataset

to find the mode there are several ways using mode method from the pandas dataframe and calculating the mode. But value_counts() method is more informational which gives the count of each element repeated in the dataset and the first value will always be the mode with the count also displayed making it more grasping for the audience

In [107…
```
crime['Offence Code'].value_counts()
```

Out[107]:
```
2305    18875
9199    14151
1399    13277
2303    12274
9113    10453
        ...
6201        1
5205        1
FTAS        1
0301        1
5199        1
Name: Offence Code, Length: 334, dtype: int64
```

In [108…
```
crime['Offence Code'].mode()
```

Out[108]:
```
0    2305
Name: Offence Code, dtype: object
```

Let's see for the police district name and calculate which police district deals with majority share of the crimes

In [109…
```
crime['Police District Name'].value_counts()
```

Out[109]:
```
SILVER SPRING          46407
WHEATON                40944
MONTGOMERY VILLAGE     37826
BETHESDA               31641
ROCKVILLE              30194
GERMANTOWN             28527
CITY OF TAKOMA PARK     4498
TAKOMA PARK             1277
OTHER                    228
Name: Police District Name, dtype: int64
```

In [110…
```
crime['Police District Name'].mode()
```

Out[110]:
```
0    SILVER SPRING
Name: Police District Name, dtype: object
```

Let's explore the most important columns of the dataset the crime names 1,2 and 3

In [111…
```
crime['Crime Name1'].value_counts()
```

Out[111]:
```
Crime Against Property    103315
Other                      47582
Crime Against Society      45287
Crime Against Person       22806
Not a Crime                 2552
Name: Crime Name1, dtype: int64
```

In [112…
```
crime['Crime Name1'].mode()
```

Out[112]:
```
0    Crime Against Property
Name: Crime Name1, dtype: object
```

for crime name 2

In [113…
```
crime['Crime Name2'].value_counts()
```

Out[113]:
```
All Other Offenses                          54215
Theft From Motor Vehicle                    18875
Simple Assault                              15797
Drug/Narcotic Violations                    14839
Destruction/Damage/Vandalism of Property    13642
Shoplifting                                 12274
All other Larceny                            9951
Driving Under the Influence                  9947
Identity Theft                               7886
Theft from Building                          7862
Burglary/Breaking and Entering               6130
Theft of Motor Vehicle Parts or Accessories  5447
False Pretenses/Swindle/Confidence Game      5086
Motor Vehicle Theft                          5068
Liquor Law Violations                        4105
Credit Card/Automatic Teller Machine Fraud   3436
Aggravated Assault                           3348
Disorderly Conduct                           3020
Runaway                                      2552
Trespass of Real Property                    2550
Robbery                                      2471
Counterfeiting/Forgery                       2416
Weapon Law Violations                        1720
Drug Equipment Violations                    1190
Forcible Rape                                1054
Fondling                                      810
Family Offenses, NonViolent                   581
Forcible Sodomy                               486
Pocket/picking                                481
```

```
Forcible Fondling                                   396
Embezzlement                                        336
Extortion/Blackmail                                 317
Sexual Assault With An Object                       317
Purse-snatching                                     298
Intimidation                                        277
Arson                                               251
Pornography/Obscene Material                        241
Bad Checks                                          234
Wire Fraud                                          217
Impersonation                                       213
NOT NIBRS CODE                                      186
Not Mapped                                          182
Stolen Property Offenses                            145
Fraud Offenses                                      143
Prostitution                                        101
Murder and Nonnegligent Manslaughter                 91
Peeping Tom                                          82
Drunkenness                                          64
Kidnapping/Abduction                                 56
From Coin/Operated Machine or Device                 41
Human Trafficking, Commercial Sex Acts               29
Statuory Rape                                        27
Curfew/Loitering/Vagrancy Violations                 19
Justifiable Homicide                                 13
Welfare Fraud                                        11
Assisting or Promoting Prostitution                  11
Human Trafficking, Involuntary Servitude              2
Negligent Manslaughter                                1
Operating/Promoting/Assisting Gambling                1
Bribery                                               1
Name: Crime Name2, dtype: int64
```

In [114…
```
crime['Crime Name2'].mode()
```

Out[114]:
```
0    All Other Offenses
Name: Crime Name2, dtype: object
```

for crime name 3

In [115…
```
crime['Crime Name3'].value_counts()
```

Out[115]:
```
LARCENY - FROM AUTO                                   18875
POLICE INFORMATION                                    14219
ASSAULT - 2ND DEGREE                                  13277
LARCENY - SHOPLIFTING                                 12274
MENTAL ILLNESS - EMERGENCY PETITION                   10453
                                                      ...
SMUGGLING - CONTRABAND                                    1
NON SUPPORT                                               1
CONSERVATION - ANIMALS (DESCRIBE OFFENSE)                 1
COMM SEX OFF - PROCURE FOR PROSTITUTE (PIMPING)           1
BRIBERY (DESCRIBE OFFENSE)                                1
Name: Crime Name3, Length: 329, dtype: int64
```

In [116…
```
crime['Crime Name3'].mode()
```

Out[116]:
```
0    LARCENY - FROM AUTO
Name: Crime Name3, dtype: object
```

let's calculate for the sector beat and PRA

In [117…
```
crime['Sector'].value_counts()
```

Out[117]:
```
P    21662
A    21302
G    19743
D    17398
R    16165
I    16094
L    15940
N    15024
K    14526
E    14243
M    13503
H    10569
J    10478
B     8892
T     4498
Name: Sector, dtype: int64
```

In [118…
```
crime['Beat'].value_counts()
```

Out[118]:
```
3G1    14588
4L2     9678
3I1     9568
5N1     9171
1A3     8471
4K1     7471
6P2     7415
5M1     7264
6P3     7213
4K2     7055
6P1     7025
6R1     6693
3I2     6526
4L1     6262
3H2     5764
4J1     5648
6R2     5601
1A2     5502
2D2     5380
3G2     5140
2D4     5030
4J2     4830
3H1     4805
5N2     4713
1A1     4344
1B1     4316
2E1     4296
2E2     4115
2E3     3891
6R3     3870
2D3     3723
1B2     3700
5M3     3276
2D1     3265
1A4     2985
5M2     2963
2E4     1941
8T3     1893
8T2     1447
8T1     1158
5N3     1140
1B3      868
3G4        9
1B4        8
6P6        6
3G5        4
3G3        2
6P4        2
-PG        1
1R2        1
Name: Beat, dtype: int64
```

In [119…
```
crime['PRA'].value_counts()
```

Out[119]:
```
702     3961
321     3032
380     3012
701     2982
419     2434
        ...
762        1
629        1
48         1
651        1
46         1
Name: PRA, Length: 800, dtype: int64
```

In [120…
```
crime_2018 = crime.loc[crime['Start_Year'] == 2018]
crime_2019 = crime.loc[crime['Start_Year'] == 2019]
crime_2020 = crime.loc[crime['Start_Year'] == 2020]
crime_2021 = crime.loc[crime['Start_Year'] == 2021]
crime_2022 = crime.loc[crime['Start_Year'] == 2022]
```

In [121…
```
def year_plot(df):
    y = df['Month'].value_counts()
    y = dict(sorted(y.items()))
    return y
```

In [122…
```
y_2018 = year_plot(crime_2018)
y_2019 = year_plot(crime_2019)
y_2020 = year_plot(crime_2020)
y_2021 = year_plot(crime_2021)
y_2022 = year_plot(crime_2022)
```

Question 1a

What notable trends can be observed in the crime heatmap, specifically focusing on Police District Number, and how does it compare to other police districts in terms of overall changes from 2018 to 2022?¶ Answer The crime heatmap reveals a noteworthy pattern, particularly in Police District 3D, where there was a substantial number of crimes in 2018. However, over the years, there has been a significant decrease in reported crimes in this district. In contrast, other police districts, including TPPD, have experienced an increase in crime rates from 2018 to 2022. Additionally, it is worth mentioning that Police District 8D reported zero crimes in the year 2022.

In [123…

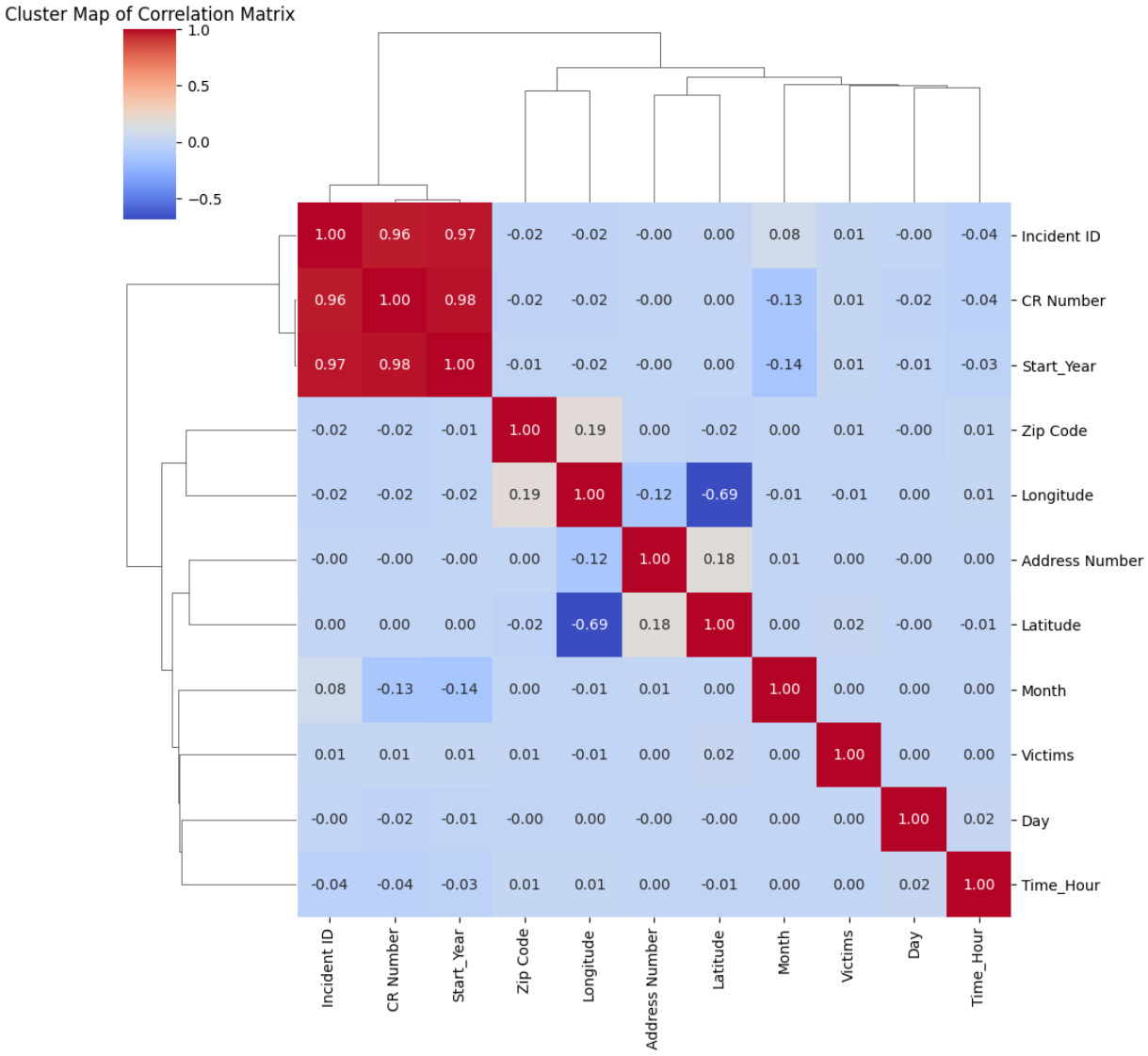
```
#2 Creating a heatmap using seaborn

heatmap_data = crime.pivot_table(index='Police District Number', columns=


plt.figure(figsize=(12, 8), dpi = 300)
sns.heatmap(heatmap_data, cmap='YlGnBu', annot=True, fmt='d', linewidths=
plt.title('Crime Heatmap Count by Police District and Start Year')
plt.show()
```

Question 1b

In [124…

```
correlation_matrix = crime.corr()

plt.figure(figsize=(12, 10))
sns.clustermap(correlation_matrix, cmap='coolwarm', annot=True, fmt=".2f"
plt.title('Cluster Map of Correlation Matrix')
plt.show()
```

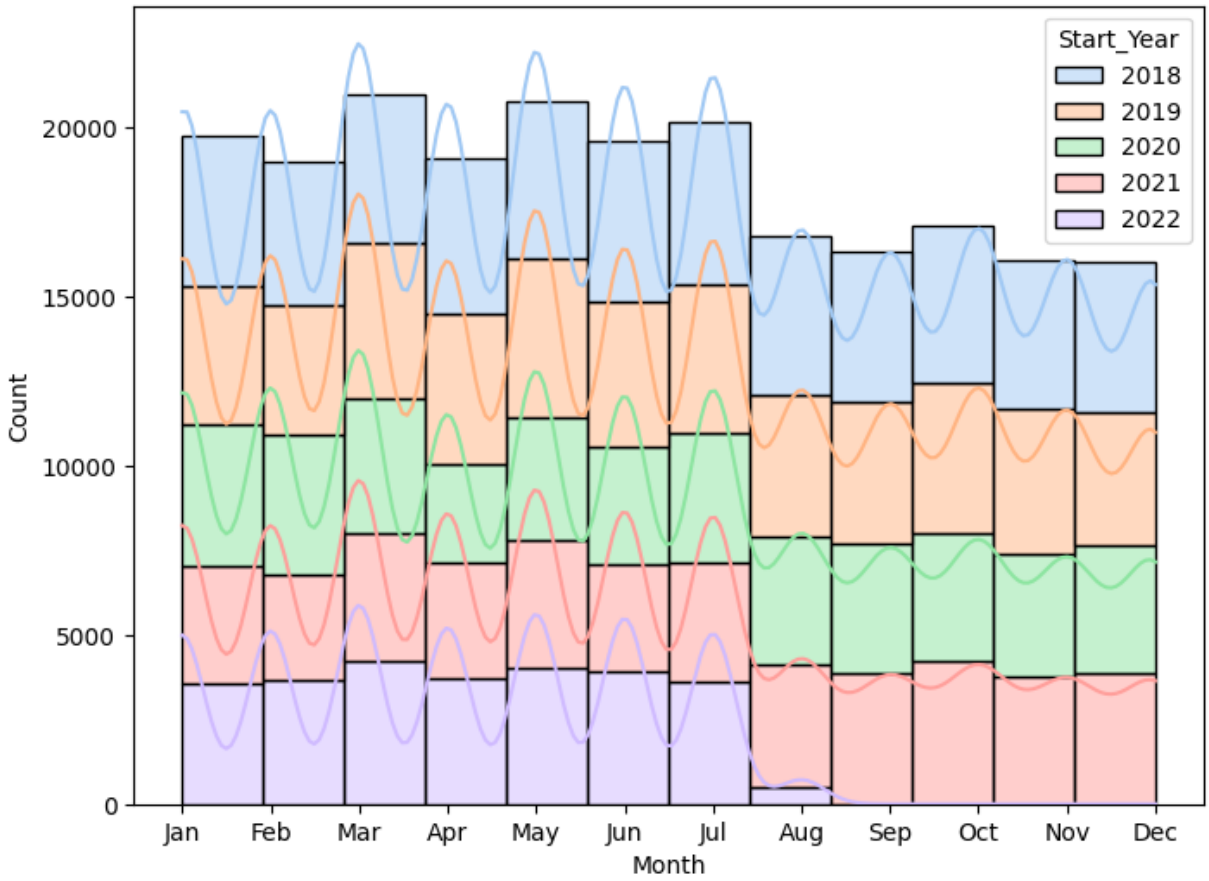
```
C:\Users\vishn\AppData\Local\Temp\ipykernel_22508\609870196.py:1: FutureW
arning: The default value of numeric_only in DataFrame.corr is deprecate
d. In a future version, it will default to False. Select only valid colum
ns or specify the value of numeric_only to silence this warning.
  correlation_matrix = crime.corr()
<Figure size 1200x1000 with 0 Axes>
```



question 2a

In [125…

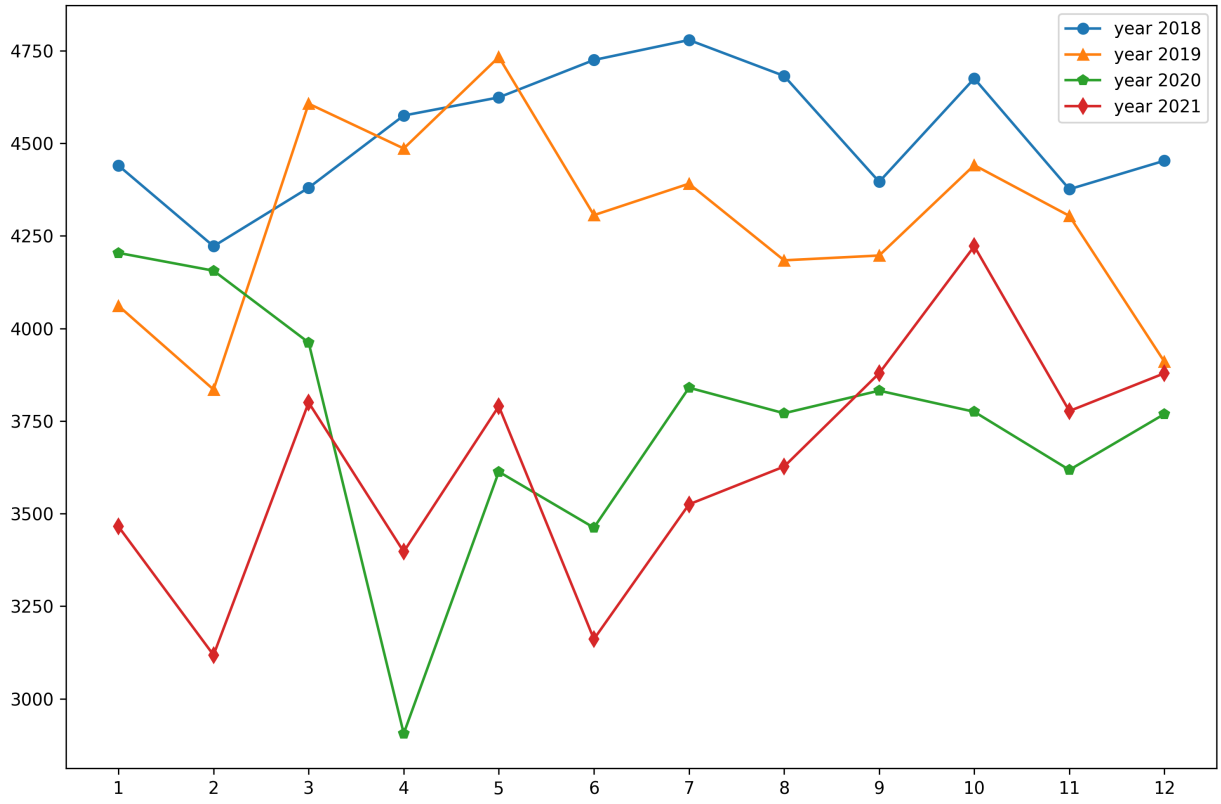
```
plt.figure(figsize=(8, 6))
sns.histplot(data = crime, x = crime['Month'], kde = True, bins=12, hue =
plt.xticks(range(1, 13), ['Jan', 'Feb', 'Mar', 'Apr', 'May', 'Jun', 'Jul'
plt.savefig('hist5.png', bbox_inches='tight')
```



question 2b

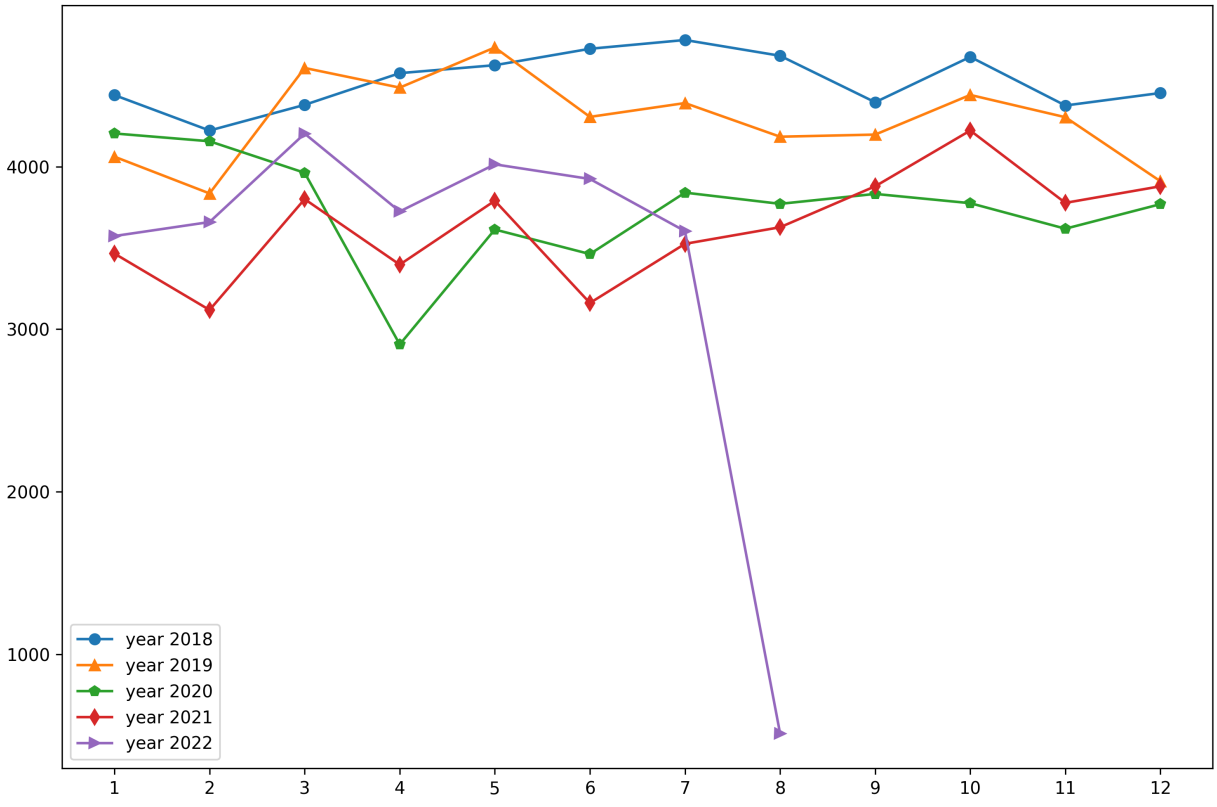
In [126…

```
plt.figure(figsize = (12,8), dpi = 300)
plt.plot(sorted(list(crime_2018['Month'].unique())), y_2018.values(), mar
plt.plot(sorted(list(crime_2019['Month'].unique())), y_2019.values(), mar
plt.plot(sorted(list(crime_2020['Month'].unique())), y_2020.values(), mar
plt.plot(sorted(list(crime_2021['Month'].unique())), y_2021.values(), mar
# plt.plot(sorted(list(crime_2018['Month'].unique())), y_2022.values(), m

plt.xticks(sorted(list(crime_2018['Month'].unique())))
plt.legend(loc = 'best')
plt.savefig('line4.png', bbox_inches='tight')
plt.show()
```



In [127…

```
plt.figure(figsize = (12,8), dpi = 300)
plt.plot(sorted(list(crime_2018['Month'].unique())), y_2018.values(), mar
plt.plot(sorted(list(crime_2019['Month'].unique())), y_2019.values(), mar
plt.plot(sorted(list(crime_2020['Month'].unique())), y_2020.values(), mar
plt.plot(sorted(list(crime_2021['Month'].unique())), y_2021.values(), mar
plt.plot(sorted(list(crime_2022['Month'].unique())), y_2022.values(), mar

plt.xticks(sorted(list(crime_2018['Month'].unique())))
plt.legend(loc = 'best')
plt.show()
```

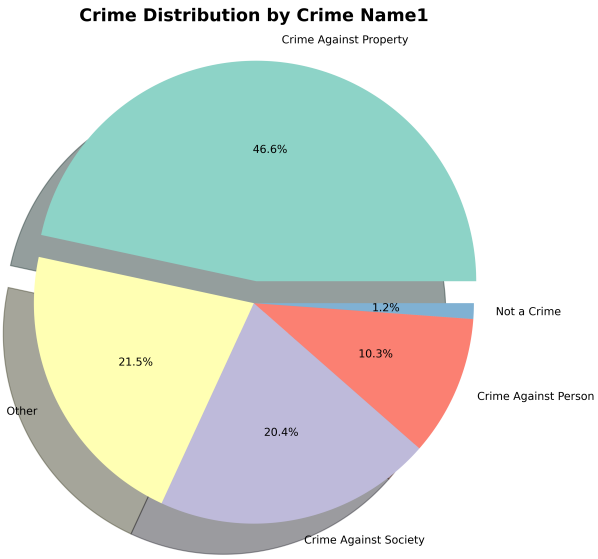

4000
3000
2000
1000
1
2
3
4
5
6
7
8
9
10
11
12
year 2018
year 2019
year 2020
year 2021
year 2022

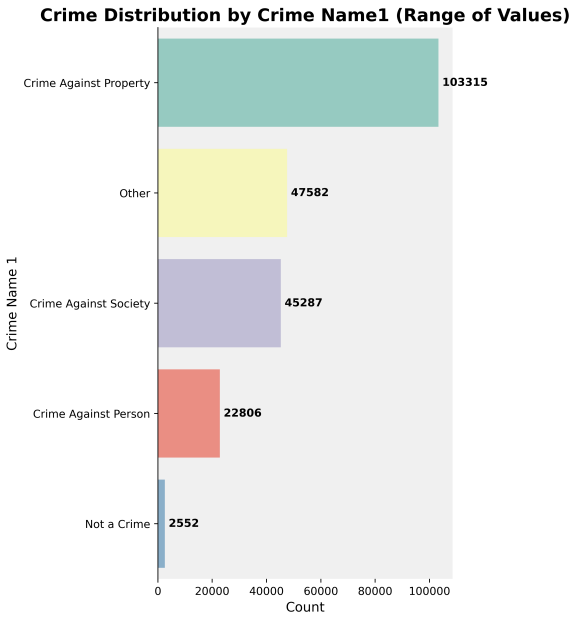
question 3a

In [128…

```
# Assuming crime_data5 is a DataFrame with 'Crime Name1' column
crime_name1_counts = crime['Crime Name1'].value_counts()

# Set up figure and axes
fig, (ax1, ax2) = plt.subplots(1, 2, figsize=(15, 8), gridspec_kw={'width

# Use a more colorful Set3 color palette
colors_set3 = sns.color_palette('Set3', n_colors=len(crime_name1_counts))

# Find the index of the largest value in crime_name1_counts
largest_index = crime_name1_counts.idxmax()

# Create an explode list with 0.1 for the largest slice (0.1 means pullin
explode = [0.1 if index == largest_index else 0 for index in crime_name1_

# Pie chart with explode
ax1.pie(x=crime_name1_counts.values, labels=crime_name1_counts.index, aut
ax1.set_title('Crime Distribution by Crime Name1', fontsize=16, fontweigh

# Bar chart
barplot = sns.barplot(x=crime_name1_counts.values, y=crime_name1_counts.i
ax2.set_title('Crime Distribution by Crime Name1 (Range of Values)', font
ax2.set_xlabel('Count', fontsize=12)
ax2.set_ylabel('Crime Name 1', fontsize=12)

# Add value counts on each bar
for index, value in enumerate(crime_name1_counts.values):
    barplot.text(value + 1, index, f' {value}', color='black', ha="left",

# Increase space between subplots
plt.subplots_adjust(wspace=0.5)

# Remove spines
sns.despine(ax=ax2, top=True, right=True, bottom=True)

# Add a background color to the bar chart
ax2.set_facecolor('#f0f0f0')

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

```
C:\Users\vishn\AppData\Local\Temp\ipykernel_22508\81637548.py:21: FutureW
arning:

Passing `palette` without assigning `hue` is deprecated and will be remov
ed in v0.14.0. Assign the `y` variable to `hue` and set `legend=False` fo
r the same effect.

  barplot = sns.barplot(x=crime_name1_counts.values, y=crime_name1_count
s.index, ax=ax2, palette=colors_set3)
```


Crime Distribution by Crime Name1
Crime Against Property
46.6%
Not a Crime
1.2%
Crime Against Person
10.3%
Crime Against Society
20.4%
Other
21.5%

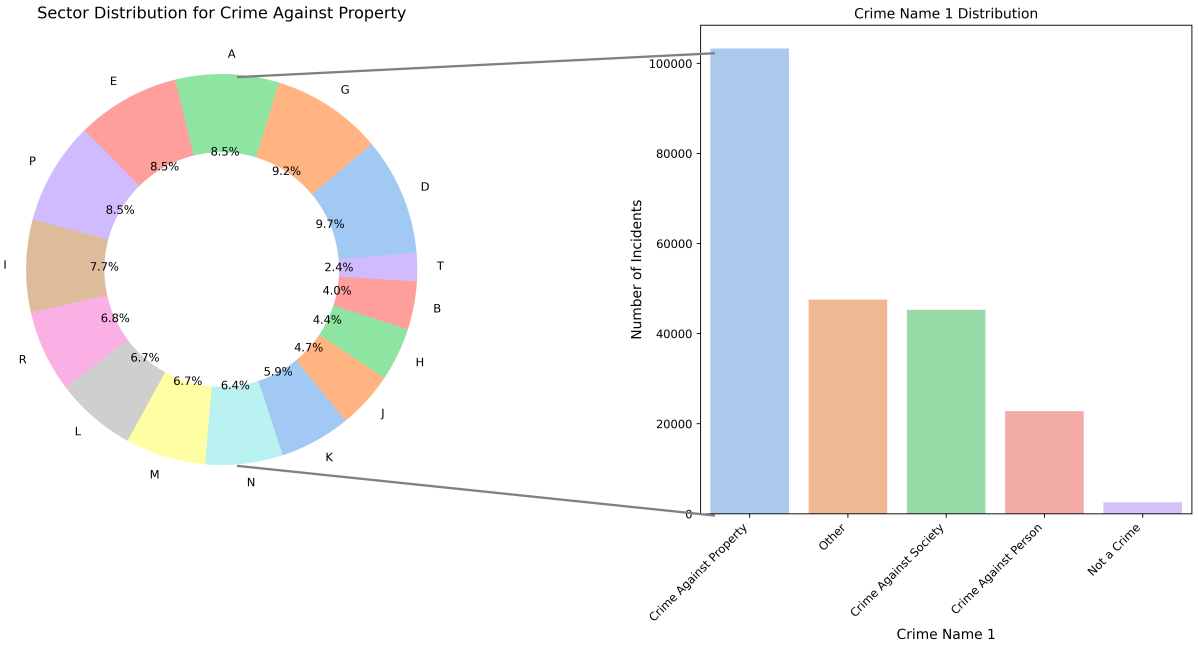

Crime Distribution by Crime Name1 (Range of Values)
Crime Against Property
103315
Other
47582
Crime Against Society
45287
Crime Against Person
22806
Not a Crime
2552
Crime Name 1
0
20000
40000
60000
80000
100000
Count

question 3b

In [129…

```
# Set a color palette for better aesthetics
sns.set_palette("husl")

# Create a grid with 1 row and 2 columns
fig = plt.figure(figsize=(18, 8), dpi = 500)
gs = GridSpec(1, 2, width_ratios=[1, 0.5])


# Bar chart for Crime Name 1
ax0 = plt.subplot(gs[1])
bar_chart = sns.countplot(x='Crime Name1', data=crime, order=crime['Crime
ax0.set_title('Crime Name 1 Distribution')
ax0.set_xticks(range(len(crime['Crime Name1'].unique())))  # Set the x-ti
ax0.set_xticklabels(crime['Crime Name1'].value_counts().index, rotation=4
ax0.set_xlabel('Crime Name 1', fontsize=12)
ax0.set_ylabel('Number of Incidents', fontsize=12)

# Find the crime with the highest frequency
highest_crime = crime['Crime Name1'].value_counts().idxmax()

# Filter the dataset for the crime with the highest frequency
highest_crime_data = crime[crime['Crime Name1'] == highest_crime]

# Count the occurrences of each sector
sector_counts = highest_crime_data['Sector'].value_counts()

# Create a DataFrame for the donut chart
sector_df = pd.DataFrame({'count': sector_counts.values}, index=sector_co

# Donut chart for Sector distribution of the highest crime
ax1 = plt.subplot(gs[0])
donut_chart = ax1.pie(sector_df['count'], labels=sector_df.index, autopct
ax1.add_artist(plt.Circle((0, 0), 0.6, fc='white'))  # Add a white circle
ax1.set_title(f'Sector Distribution for {highest_crime}', fontsize=14)

# Add fixed-size lines
line_coords1 = [(0.35, 0.67), (0.874, 0.91)]  # Adjust the coordinates as

line_coords2 = [(0.35, 0.67), (0.285, 0.21)]  # Adjust the coordinates as

line1 = plt.Line2D(line_coords1[0], line_coords1[1], color='grey', linewi
line2 = plt.Line2D(line_coords2[0], line_coords2[1], color='grey', linewi

fig.lines.append(line1)
fig.lines.append(line2)

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

```
C:\Users\vishn\AppData\Local\Temp\ipykernel_22508\492733378.py:11: Future
Warning:

Passing `palette` without assigning `hue` is deprecated and will be remov
ed in v0.14.0. Assign the `x` variable to `hue` and set `legend=False` fo
r the same effect.

  bar_chart = sns.countplot(x='Crime Name1', data=crime, order=crime['Cri
me Name1'].value_counts().index, palette="pastel", ax=ax0)
```



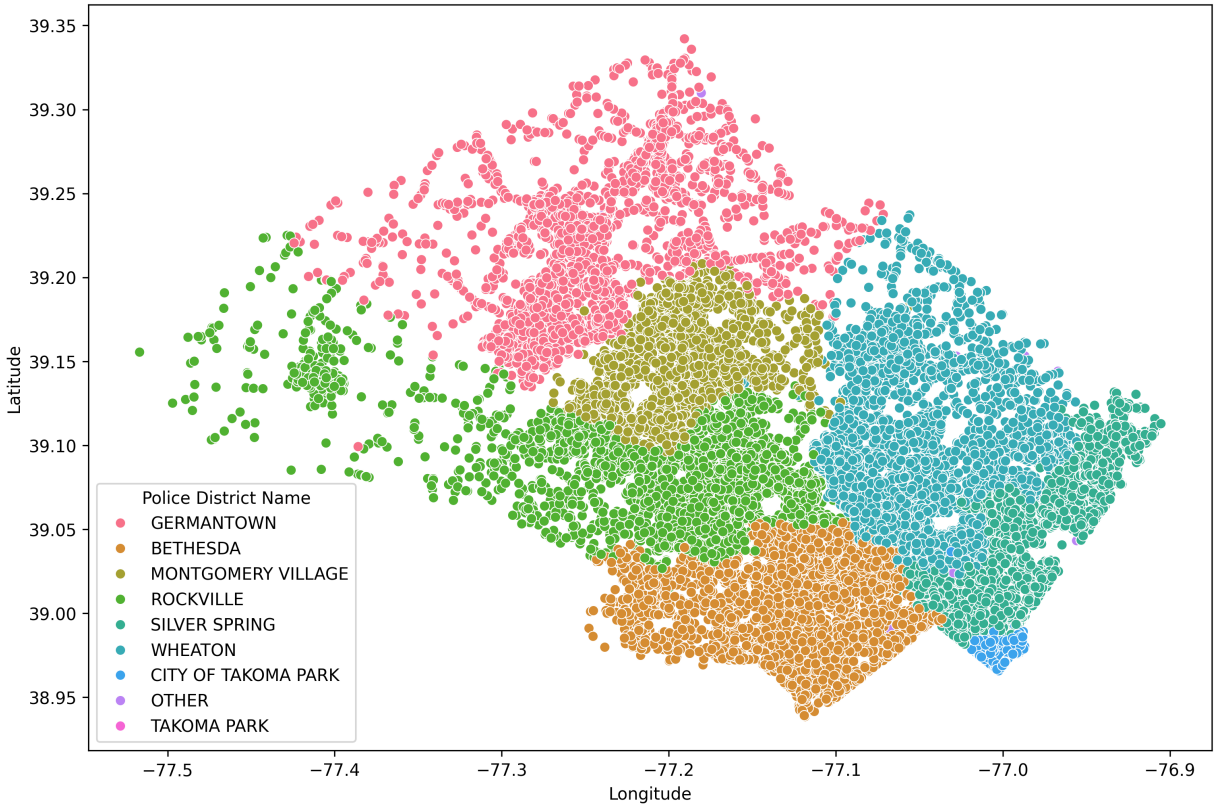
Question 4a

In [130…

```
plt.figure(figsize=(12,8), dpi=300)
sns.scatterplot(data=crime, y = crime.Latitude, x = crime.Longitude, hue=
```

Out[130]:

```
<Axes: xlabel='Longitude', ylabel='Latitude'>
```

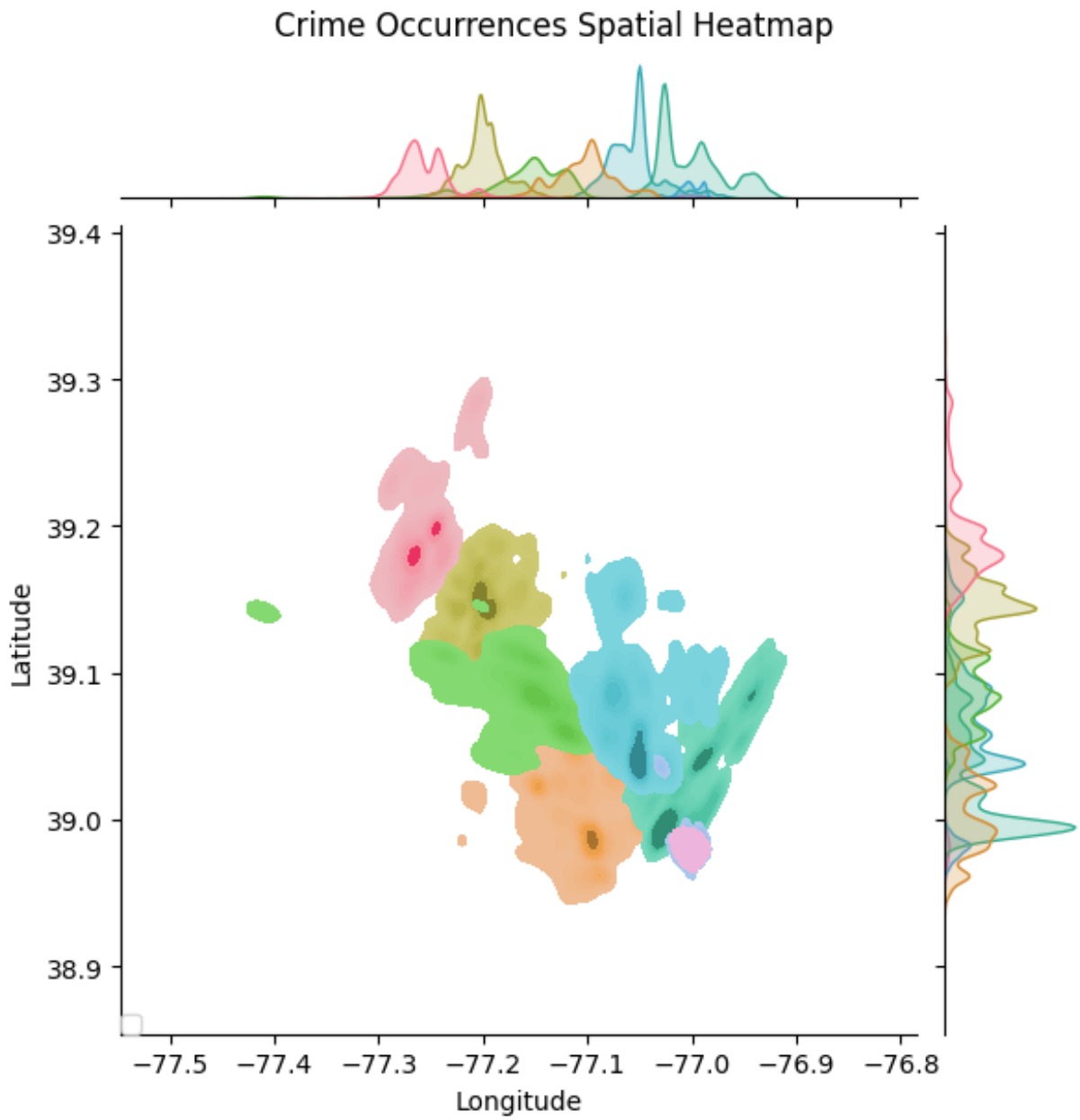


question 4b

In [132…

```
plt.figure(figsize=(12,8), dpi = 200)
sns.jointplot(x=crime['Longitude'], y=crime['Latitude'], hue = crime['Pol
plt.suptitle('Crime Occurrences Spatial Heatmap', y=1.02)
plt.legend(loc='lower left', borderaxespad=0)
plt.show()
```

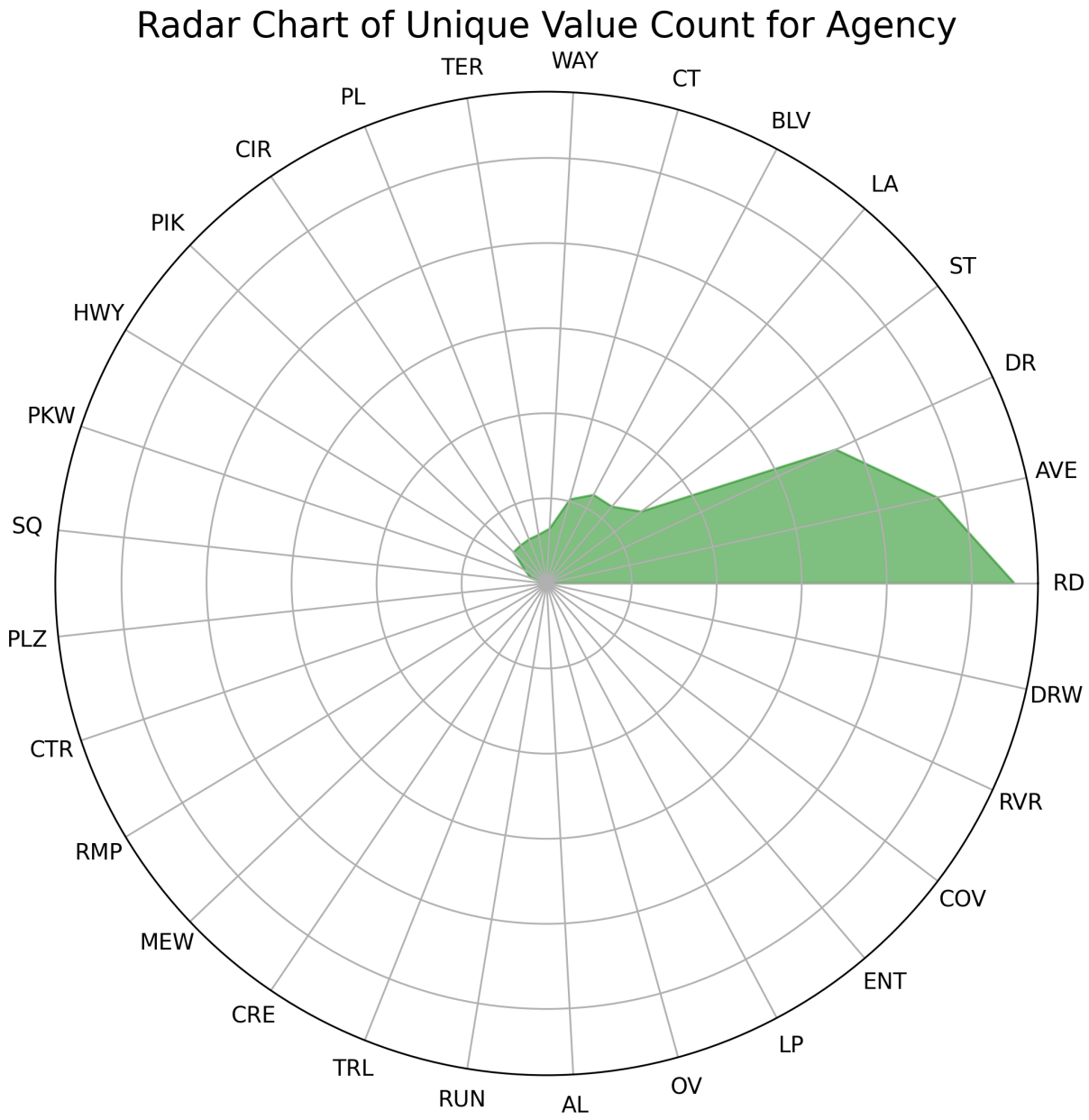
```
No artists with labels found to put in legend.  Note that artists whose l
abel start with an underscore are ignored when legend() is called with no
argument.
<Figure size 2400x1600 with 0 Axes>
```


Crime Occurrences Spatial Heatmap
39.4
39.3
39.2
39.1
39.0
38.9
Latitude
−77.5
−77.4
−77.3
−77.2
−77.1
−77.0
−76.9
−76.8
Longitude

question 5a

In [133…

```
# Count the unique values in the 'Agency' column
agency_counts = crime['Street Type'].value_counts()

# Create a DataFrame with the counts
agency_counts_df = pd.DataFrame({'Street Type': agency_counts.index, 'Cou

# Number of variables
num_vars = len(agency_counts_df)

# Compute angle of each axis
angles = np.linspace(0, 2 * np.pi, num_vars, endpoint=False).tolist()

# The plot is circular, so we need to "close the loop" and append the sta
values = agency_counts_df['Count'].tolist()
values += values[:1]
angles += angles[:1]

# Plot
fig, ax = plt.subplots(figsize=(8,8), subplot_kw=dict(polar=True), dpi =
ax.fill(angles, values, color='green', alpha=0.5)
ax.set_yticklabels([])
ax.set_xticks(angles[:-1])
ax.set_xticklabels(agency_counts_df['Street Type'])

plt.title('Radar Chart of Unique Value Count for Agency', size=16)


plt.show()
```

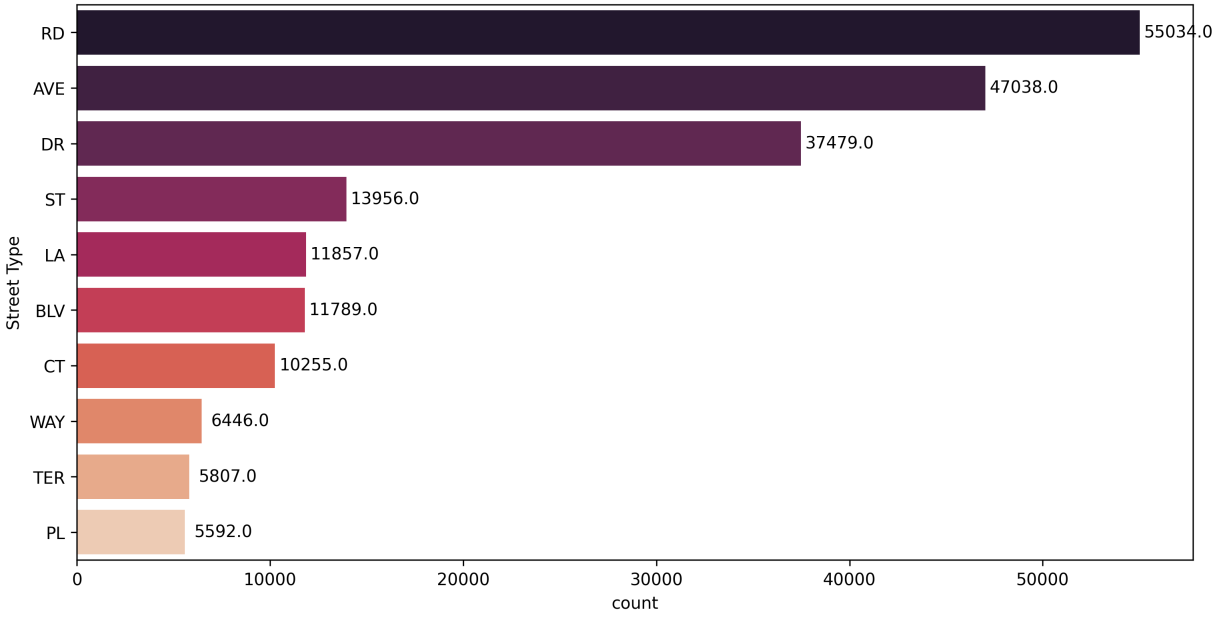


question 5b

In [134…

```
plt.figure(figsize=(12,6), dpi = 300)
# Get the top 10 values and their counts
top_values = crime['Street Type'].value_counts().head(10)

# Create a horizontal count plot for the top 10 values
sns.countplot(y="Street Type", data=crime, order=top_values.index, palett

# Display count values on the right of each bar
ax = plt.gca()
for p in ax.patches:
    ax.text(p.get_width() + 2000, p.get_y() + p.get_height() / 2, f'{p.ge

plt.show()
```

```
C:\Users\vishn\AppData\Local\Temp\ipykernel_22508\1993748988.py:6: Future
Warning:

Passing `palette` without assigning `hue` is deprecated and will be remov
ed in v0.14.0. Assign the `y` variable to `hue` and set `legend=False` fo
r the same effect.

  sns.countplot(y="Street Type", data=crime, order=top_values.index, pale
tte='rocket')
```



Question 6a

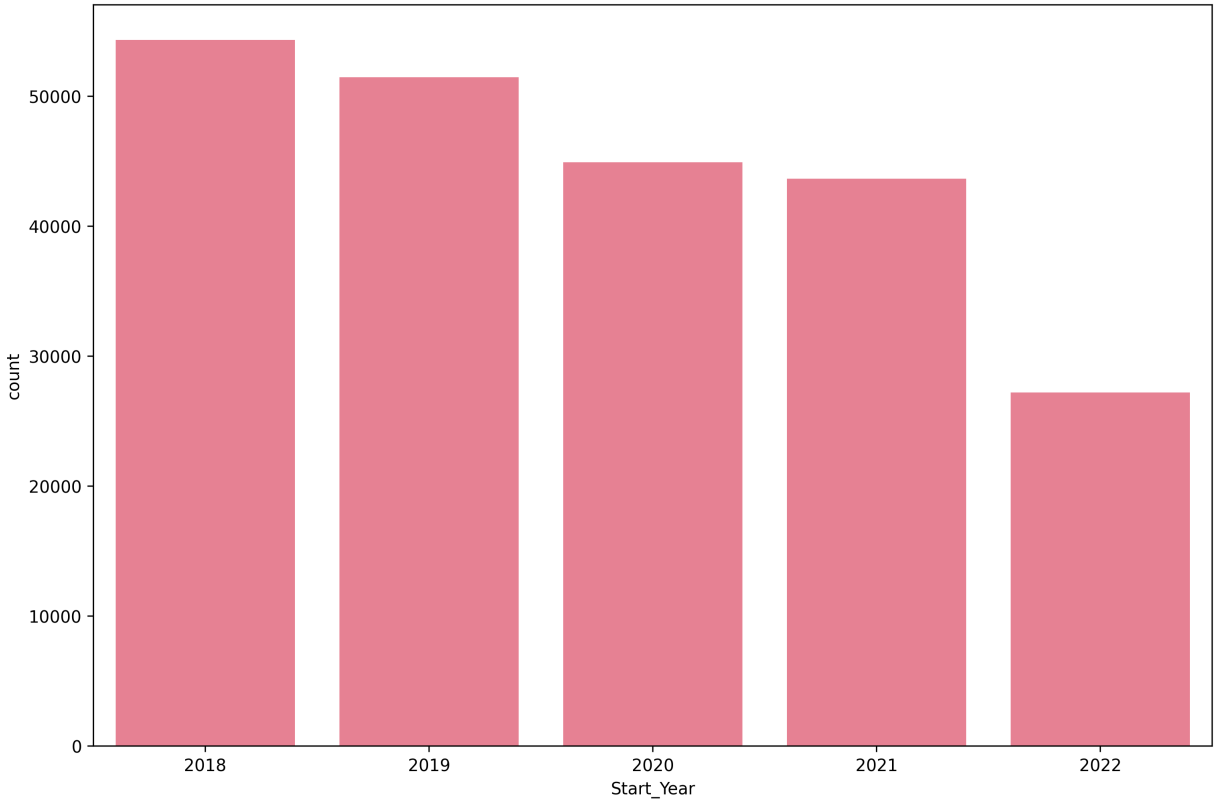
By the look of the above code cell, we do get an idea that the crime rate has decreased over years in the state of Maryland. With a countplot over the years will give us a much better visualisation that there is a decrease in the crime rate through the selected years from 2018-2022

In [135…

```
plt.figure(figsize=(12,8), dpi = 300)
sns.countplot(data = crime, x = crime['Start_Year'])
```

Out[135]:

```
<Axes: xlabel='Start_Year', ylabel='count'>
```



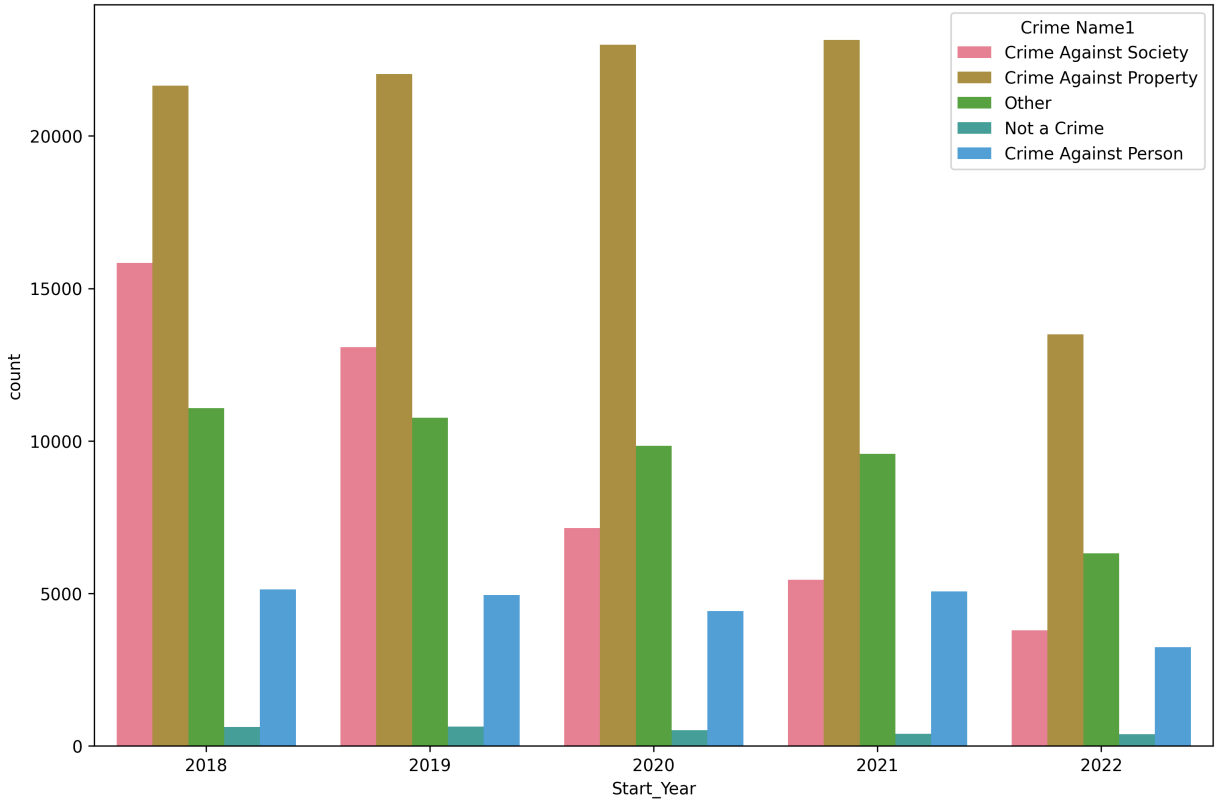
question 6b

the overall crime rate has decreased, but we want to see more deeper analysis of what type of crimes have gotten lesser over the years. With a countplot and seeting the hue to the subset of the types of crimes, which is 'Crime Name1' column in the dataset. We can see the plot below that describes the number of crimes for differetn category of crimes

The plot below shows a more deeper view into the crime rate, which depicts that the crime against society has been decresing, because of the pandemic in the recent years has made many of us home bound and there was a decrease in the crime against society. But crime against property has been at a all time high in 2021 with a visible increase from 2018, again because of the pandemic that has led rise to robbery, theft and other types of crime against the property.

In [136…
```
plt.figure(figsize=(12,8), dpi = 300)
sns.countplot(data = crime, x = crime['Start_Year'], hue=crime['Crime Nam
```

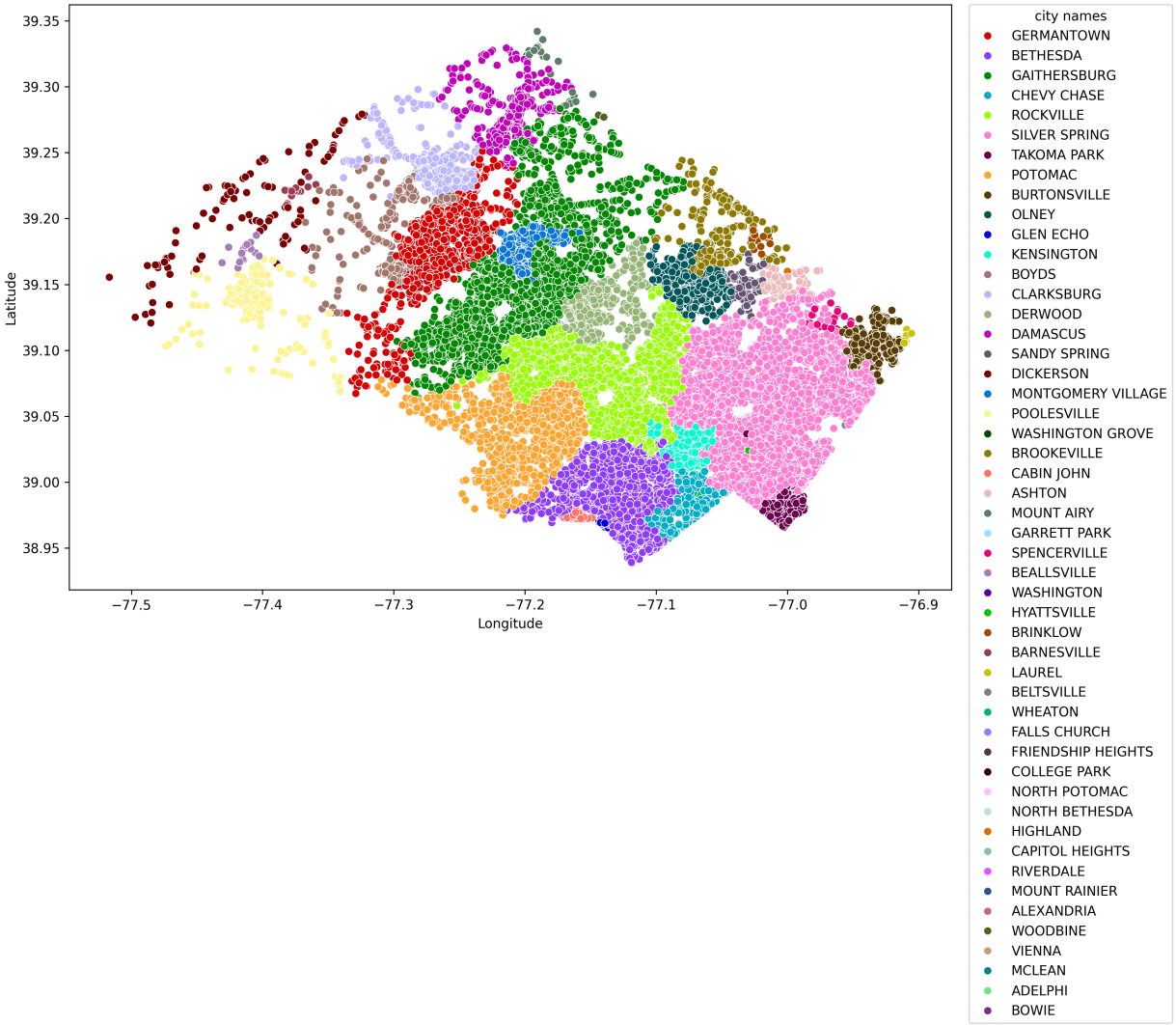
Out[136]:
```
<Axes: xlabel='Start_Year', ylabel='count'>
```



question 7a

In [137…
```
plt.figure(figsize=(12,8), dpi = 300)
#sns.rugplot(data = crime, x = crime['Longitude'], y = crime['Latitude'],
palette = sns.color_palette(cc.glasbey, n_colors=50)
sns.scatterplot(data = crime, x = crime['Longitude'], y = crime['Latitude
plt.legend(bbox_to_anchor=(1.02, 1), loc='upper left', borderaxespad=0, t
```

Out[137]:
```
<matplotlib.legend.Legend at 0x22f7a413010>
```



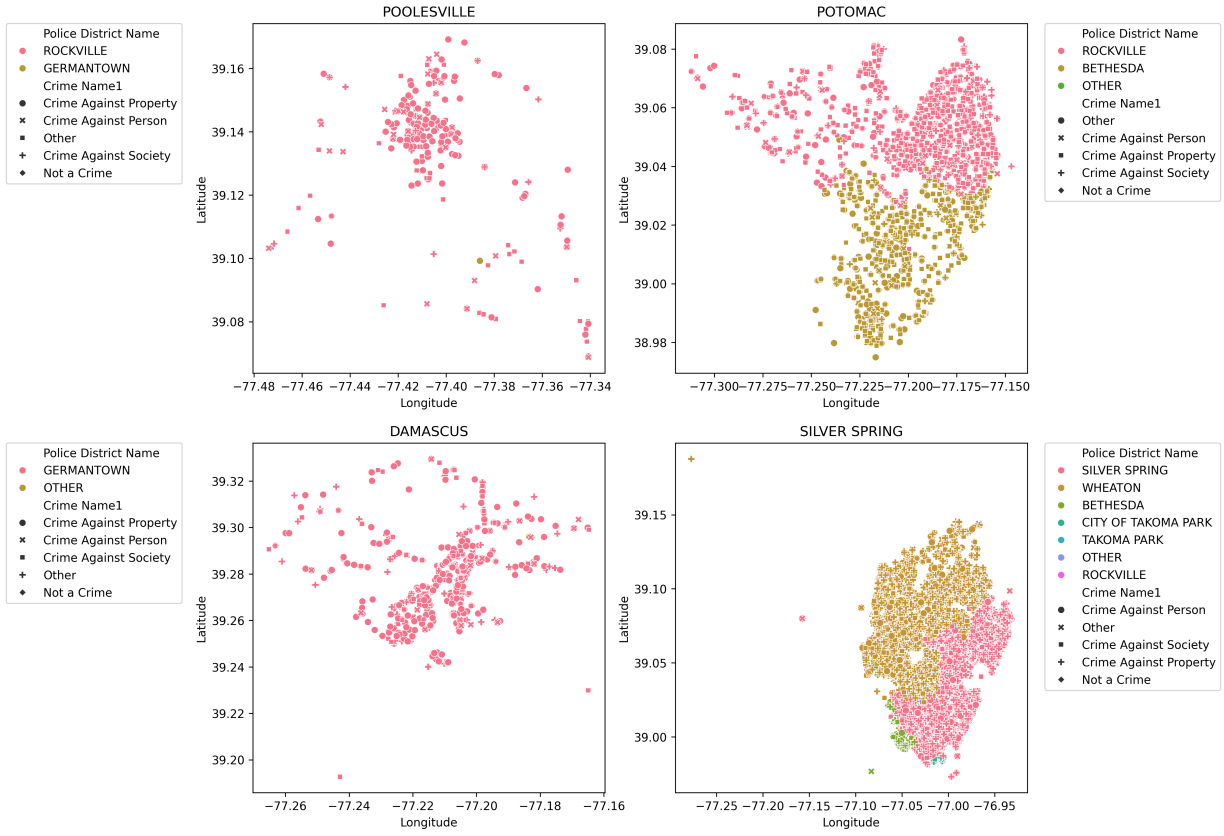
question 7b

In [138…

```
pool_crime = crime.loc[crime['City'] == 'POOLESVILLE']
potomac_crime = crime.loc[crime['City'] == 'POTOMAC']
damascus_crime = crime.loc[crime['City'] == 'DAMASCUS']
silver_crime = crime.loc[crime['City'] == 'SILVER SPRING']
```

In [139…

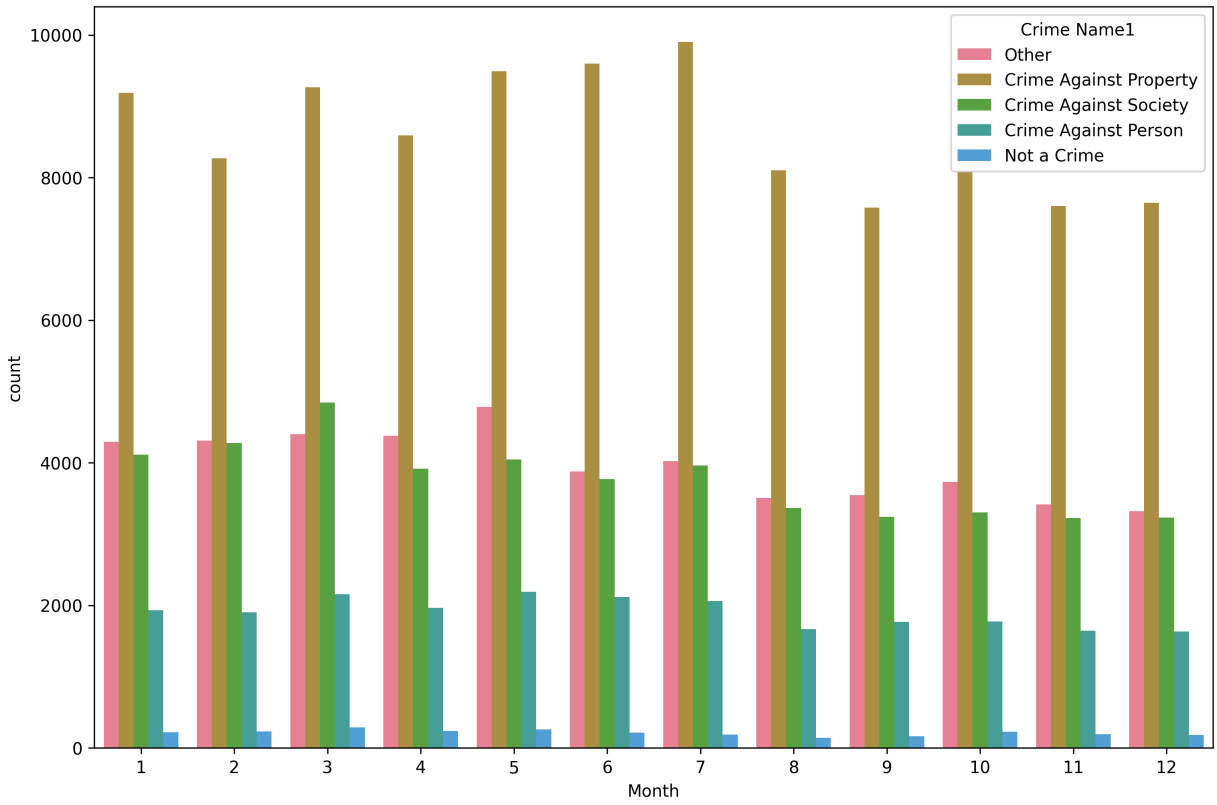
```
plt.figure(figsize=(12,12), dpi = 300)

# handles, labels = plt.gca().get_legend_handles_labels()
# plt.legend(handles, labels, loc='upper center')

plt.subplot(2,2,1)
plt.title('POOLESVILLE')
sns.scatterplot(data = pool_crime, x = pool_crime['Longitude'], y = pool_
plt.legend(bbox_to_anchor=(-0.7, 1), loc='upper left', borderaxespad=0)

plt.subplot(2,2,2)
plt.title('POTOMAC')
sns.scatterplot(data = potomac_crime, x = potomac_crime['Longitude'], y =
plt.legend(bbox_to_anchor=(1.05, 1), loc='upper left', borderaxespad=0)

plt.subplot(2,2,3)
plt.title('DAMASCUS')
sns.scatterplot(data = damascus_crime, x = damascus_crime['Longitude'], y
plt.legend(bbox_to_anchor=(-0.70, 1), loc='upper left', borderaxespad=0)

plt.subplot(2,2,4)
plt.title('SILVER SPRING')
sns.scatterplot(data = silver_crime, x = silver_crime['Longitude'], y = s
plt.legend(bbox_to_anchor=(1.05, 1), loc='upper left', borderaxespad=0)

plt.show()
```



question 8a

In [140…
```
plt.figure(figsize=(12,8), dpi = 300)
sns.countplot(data = crime, x = crime['Month'], hue = crime['Crime Name1'
```

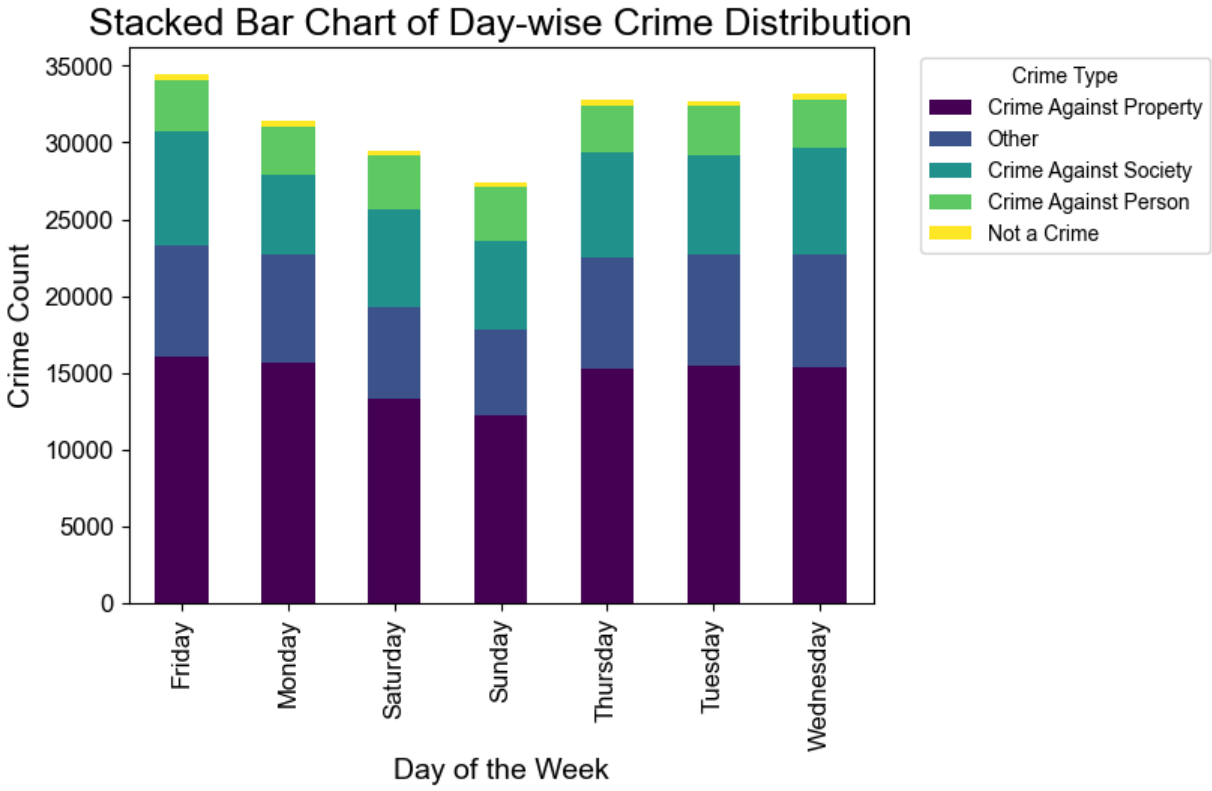
Out[140]:
```
<Axes: xlabel='Month', ylabel='count'>
```



question 8b

In [141…
```
# Color Palette
custom_palette = sns.color_palette("husl", n_colors=len(crime['Crime Name

# Stacked bar chart for crime types by day of the week
crime_day_counts = crime.groupby(['day_name', 'Crime Name1']).size().unst
crime_day_counts = crime_day_counts[crime['Crime Name1'].value_counts().i

plt.figure(figsize=(16, 8))
crime_day_counts.plot(kind='bar', stacked=True, colormap='viridis')
plt.title('Stacked Bar Chart of Day-wise Crime Distribution', fontsize=18
plt.xlabel('Day of the Week', fontsize=14)
plt.ylabel('Crime Count', fontsize=14)
plt.legend(title='Crime Type', bbox_to_anchor=(1.05, 1), loc='upper left'


sns.set(style="whitegrid")
plt.xticks(fontsize=12)
plt.yticks(fontsize=12)

plt.show()
```

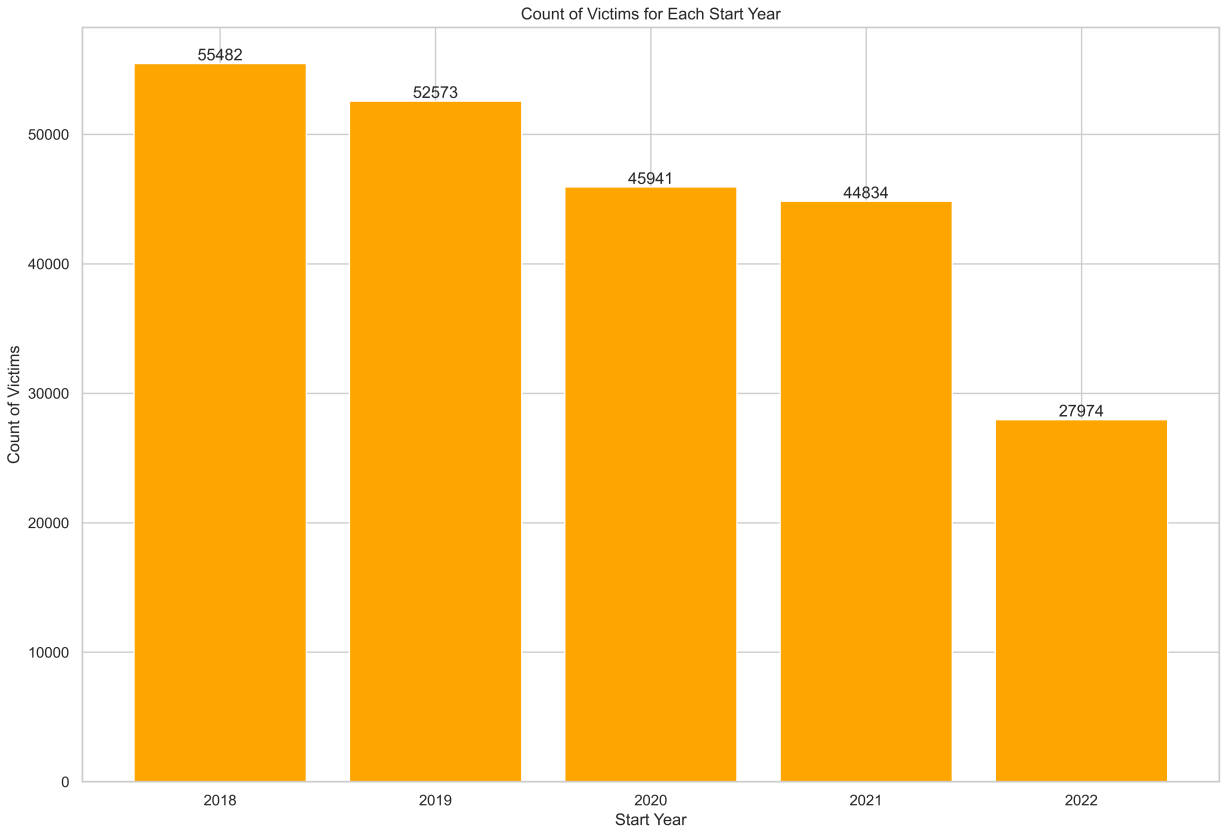
```
<Figure size 1600x800 with 0 Axes>
```



question 9a

In [142…

```
fig, ax = plt.subplots(figsize=(15, 10), dpi=500)

# Assuming 'Start_Year' and 'Victims' are columns in your 'crime_data5' D
victims_count_by_year = crime.groupby('Start_Year')['Victims'].sum()
df_victims_count = victims_count_by_year.reset_index()

# Plotting
bars = ax.bar(df_victims_count['Start_Year'], df_victims_count['Victims']

plt.xlabel('Start Year')
plt.ylabel('Count of Victims')
plt.title('Count of Victims for Each Start Year')

# Add labels on top of each bar
for bar in bars:
    yval = bar.get_height()
    plt.text(bar.get_x() + bar.get_width()/2, yval, round(yval, 1), ha='c

plt.show()
```

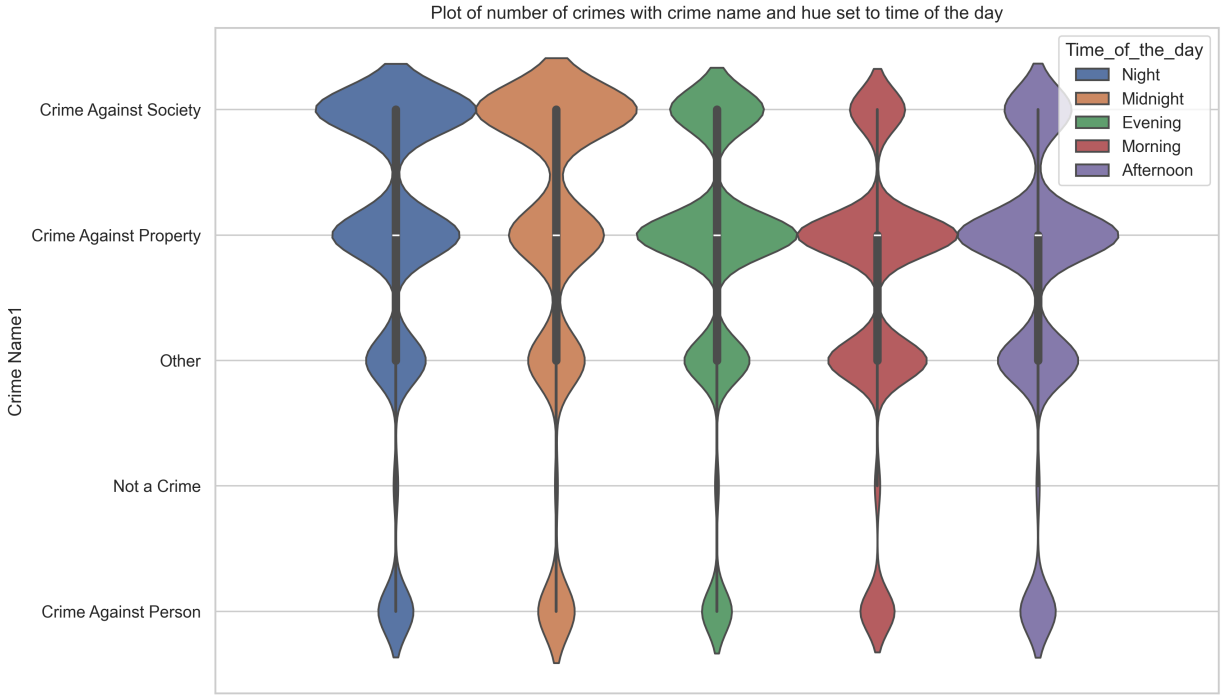


question 9b

In [143…
```
crime_2018 = crime[crime['Start_Year'] == 2018]
```

In [144…
```
plt.figure(figsize=(12,8), dpi=300)
sns.violinplot(data = crime_2018, y = crime_2018['Crime Name1'], hue = cr
plt.title('Plot of number of crimes with crime name and hue set to time o
```

Out[144]:
```
Text(0.5, 1.0, 'Plot of number of crimes with crime name and hue set to
time of the day')
```



question 10

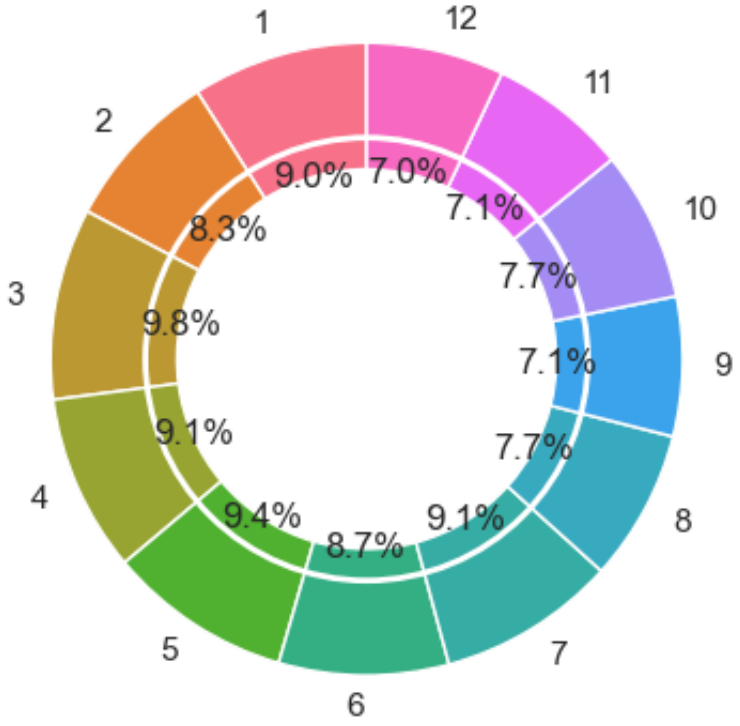
In [145…

```
inner_column = 'Time_of_the_day'
outer_column = 'Month'

crime_counts = crime.groupby([inner_column, outer_column]).size().unstack

inner_palette = sns.color_palette("husl", n_colors=len(crime_counts.colum
outer_palette = sns.color_palette("Set3", n_colors=len(crime_counts.index

plt.figure(figsize=(12, 12))
outer_circle_radius = 0.7

for outer_index, (outer_label, outer_row) in enumerate(crime_counts.iterr
    plt.pie(outer_row, labels=outer_row.index, autopct='%1.1f%%', startan
            wedgeprops=dict(width=0.4, edgecolor='w'), radius=1.0, colors

    outer_circle = plt.Circle((0, 0), outer_circle_radius, color='white',
    plt.gca().add_artist(outer_circle)

    plt.title(f'Crime Distribution by {inner_column} - {outer_column} {ou
    plt.show()
```

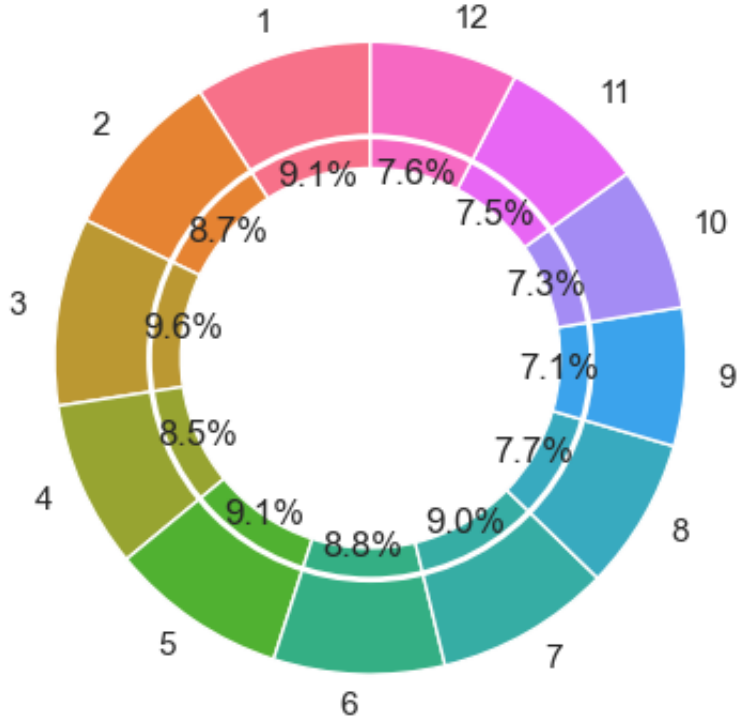
```
C:\Users\vishn\AppData\Local\Temp\ipykernel_22508\3283741658.py:16: UserW
arning: Setting the 'color' property will override the edgecolor or facec
olor properties.
  outer_circle = plt.Circle((0, 0), outer_circle_radius, color='white', e
dgecolor='black', linewidth=2, fill=False)
```

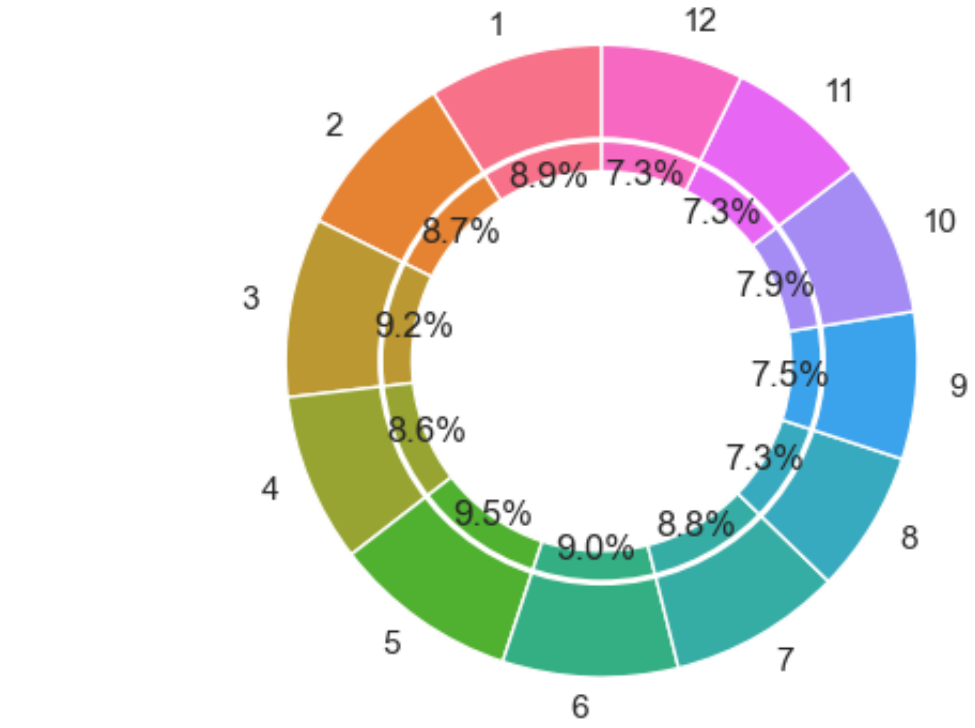

Crime Distribution by Time_of_the_day - Month Afternoon
1
12
11
2
10
3
9
4
8
5
7
6
9.4%
7.3%
7.5%
8.9%
7.9%
9.6%
7.4%
8.4%
7.2%
9.3%
8.5%
8.5%

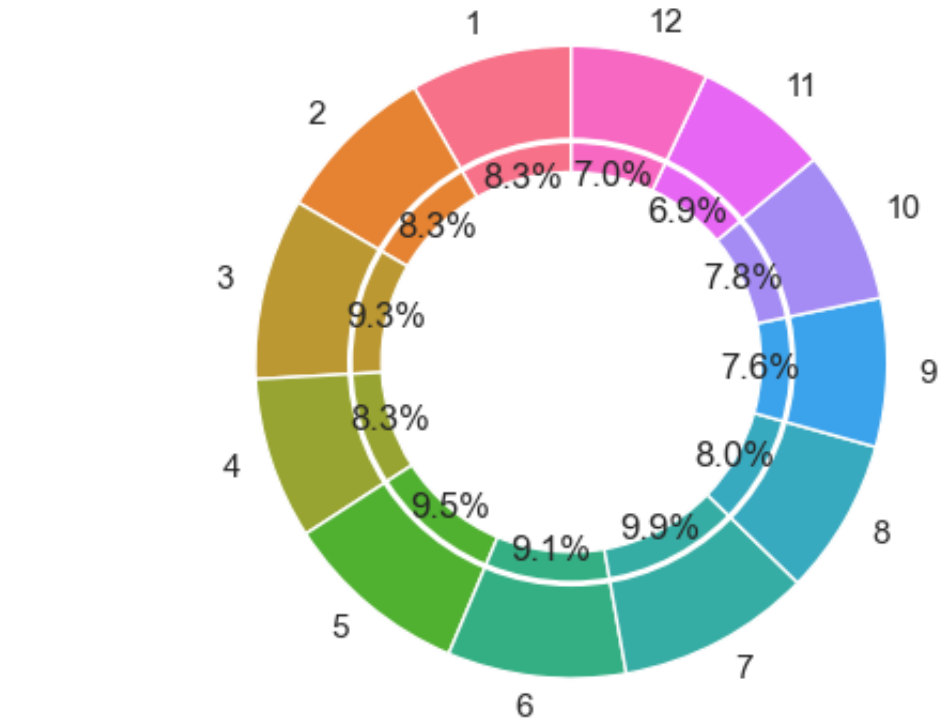
Crime Distribution by Time_of_the_day - Month Evening



Crime Distribution by Time_of_the_day - Month Midnight

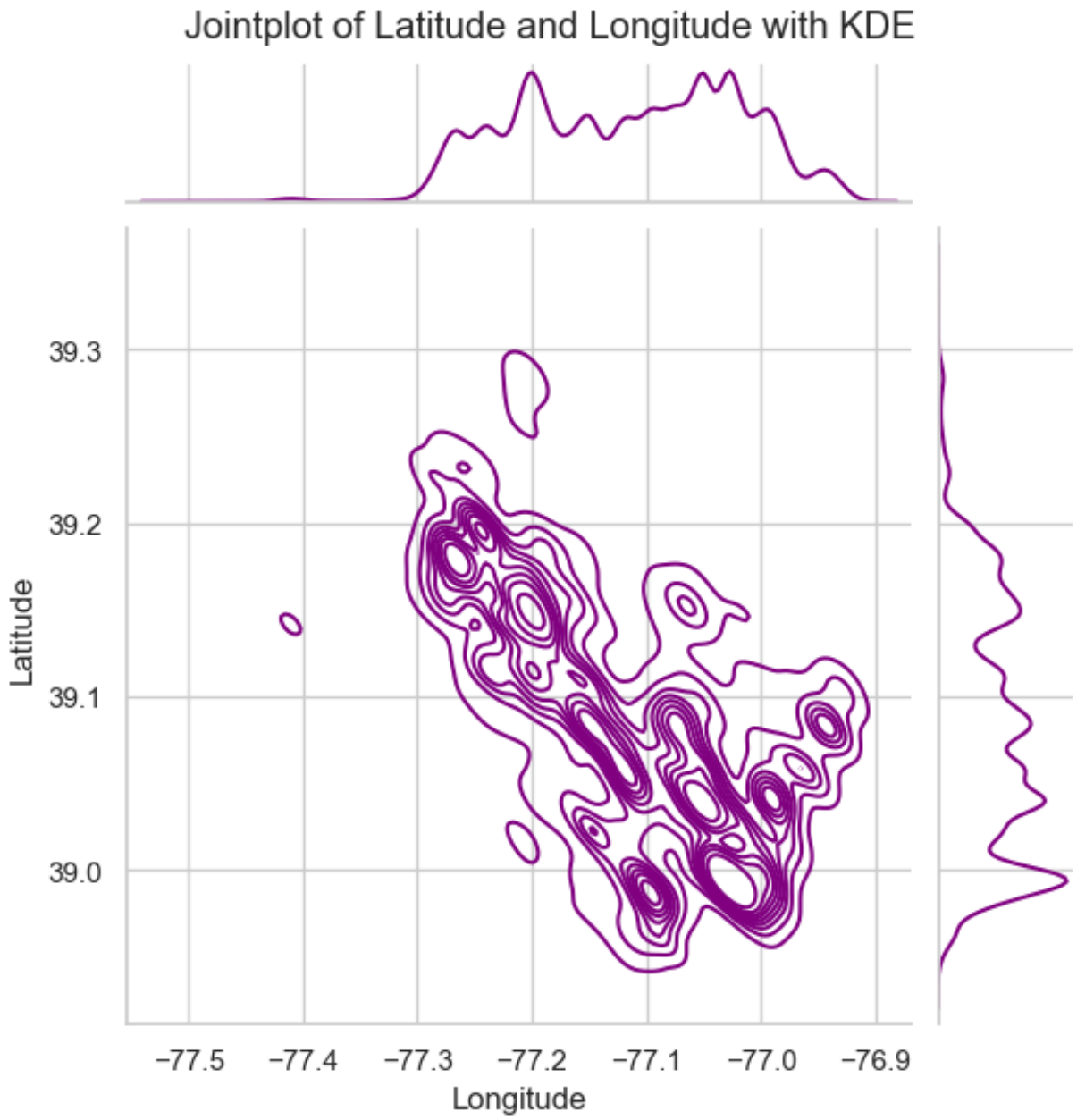






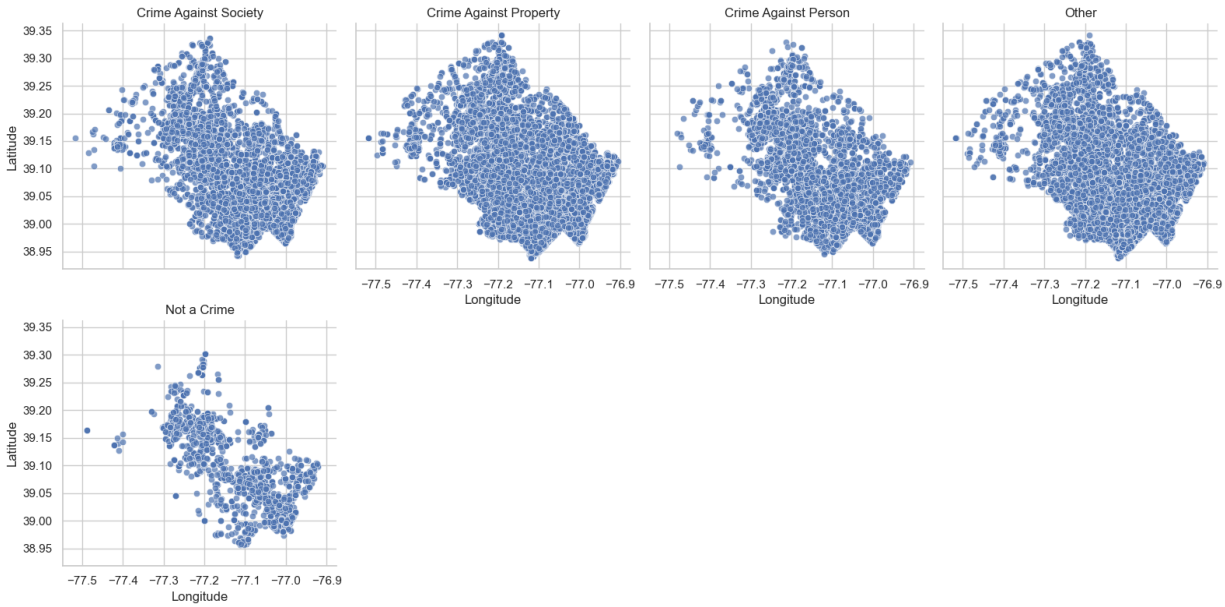
question 11a

In [146…
```
sns.jointplot(x='Longitude', y='Latitude', data=crime, kind='kde', color=
plt.suptitle('Jointplot of Latitude and Longitude with KDE', y=1.02)
plt.show()
```

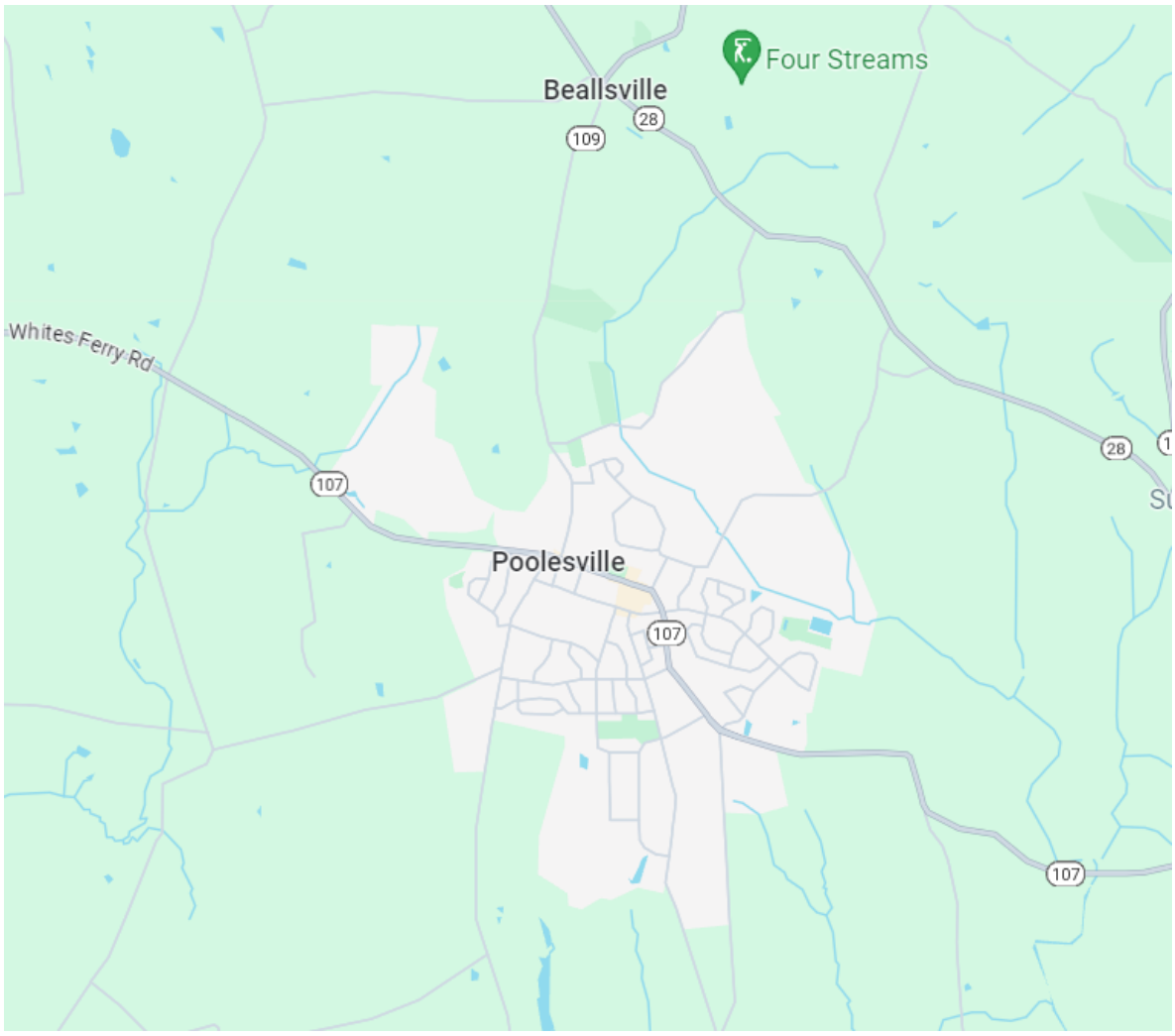


question 11b

In [147…
```
g = sns.FacetGrid(crime, col='Crime Name1', col_wrap=4, height=4)
g.map(sns.scatterplot, 'Longitude', 'Latitude', alpha=0.7)
g.set_axis_labels('Longitude', 'Latitude')
g.set_titles(col_template='{col_name}')
plt.show()
```

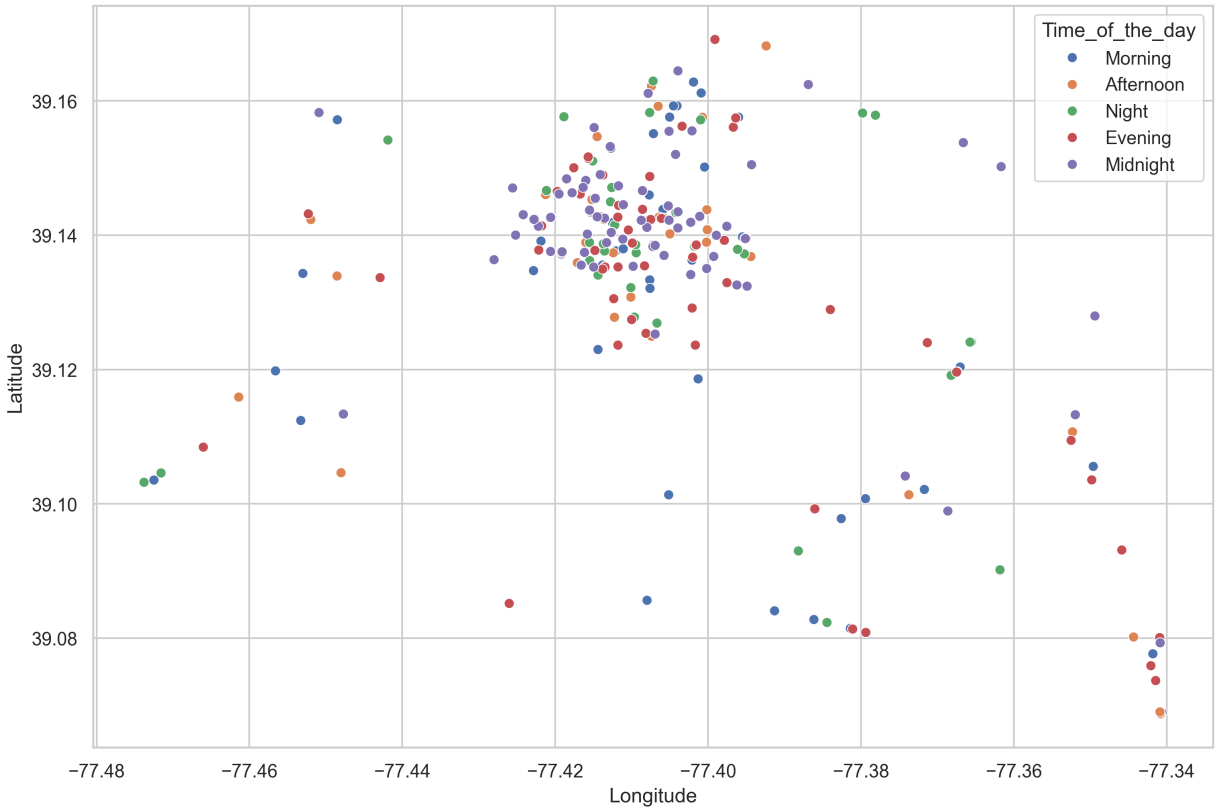


question 12a



In [149…

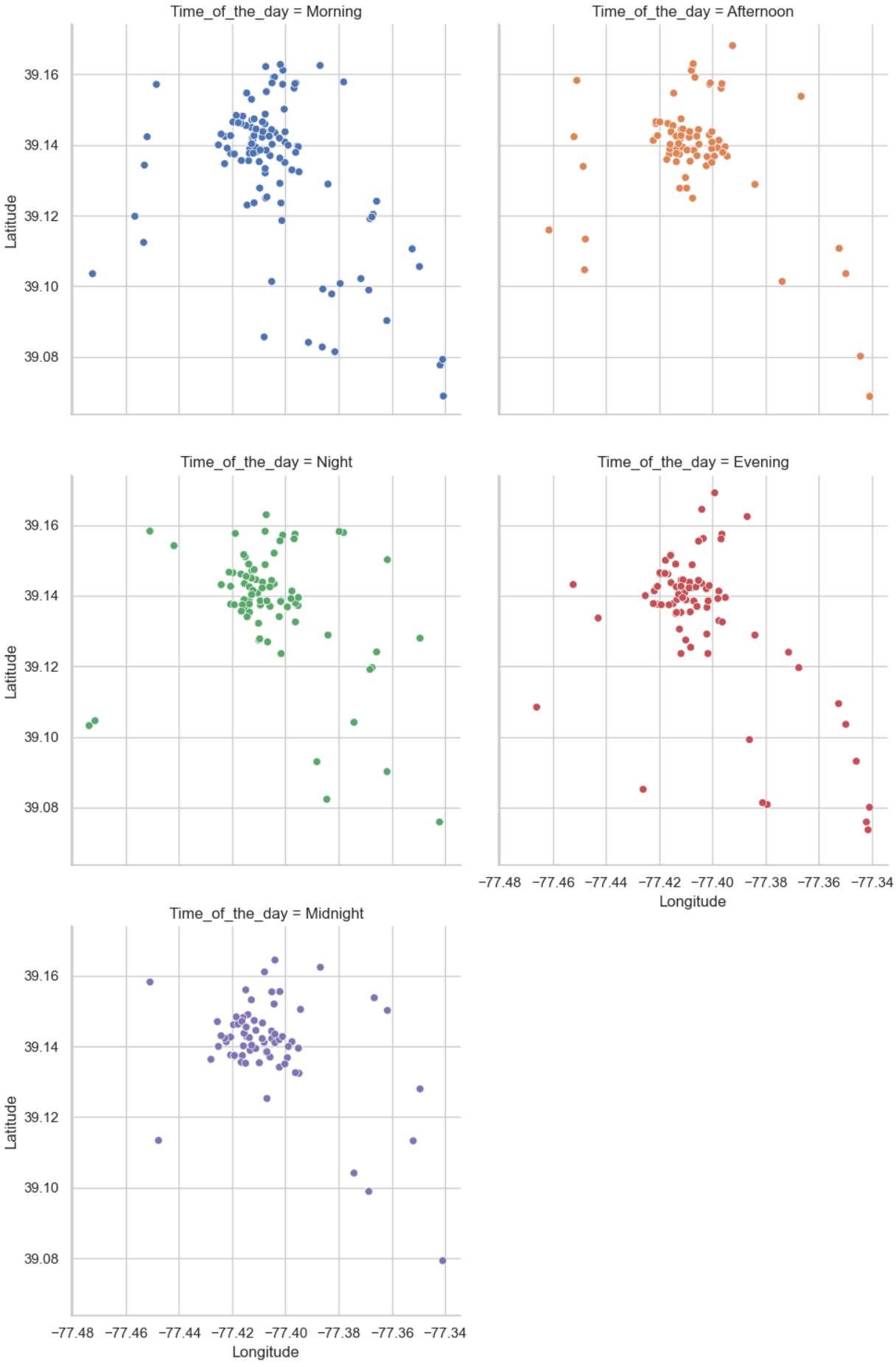
```
pool_crime = crime[crime['City'] == 'POOLESVILLE']
desired_order = ['Morning', 'Afternoon', 'Night', 'Evening', 'Midnight']
plt.figure(figsize=(12, 8), dpi=300)
sns.scatterplot(
    data=pool_crime.sort_values('Time_of_the_day', key=np.vectorize(desir
    x='Longitude',
    y='Latitude',
    hue='Time_of_the_day',
    hue_order=desired_order,
)

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



question 12b

In [150…

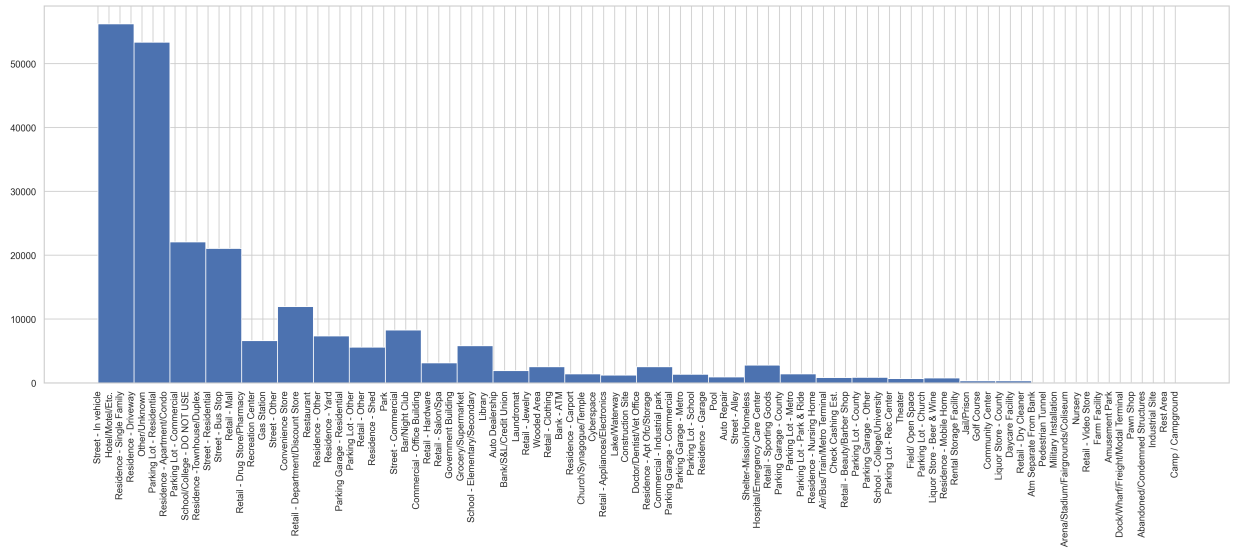
```
pool_crime = crime[crime['City'] == 'POOLESVILLE']

g = sns.relplot(
    data=pool_crime,
    x="Longitude",
    y="Latitude",
    hue='Police District Name',
    col="Time_of_the_day",
    col_wrap=2
)

num_plots = len(g.axes.flatten())

colors = sns.color_palette(n_colors=num_plots)

for i, ax in enumerate(g.axes.flat):
    current_time = pool_crime['Time_of_the_day'].unique()[i]
    current_data = pool_crime[pool_crime['Time_of_the_day'] == current_ti
    sns.scatterplot(
        data=current_data,
        x='Longitude',
        y='Latitude',
        hue='Police District Name',
        palette={dist: colors[i] for dist in current_data['Police Distric
        ax=ax
    )
    ax.get_legend().set_visible(False)

g._legend.remove()
# Show the plot
plt.show()
```


Time_of_the_day = Morning
Time_of_the_day = Afternoon
Time_of_the_day = Night
Time_of_the_day = Evening
Time_of_the_day = Midnight
Latitude
39.16
39.14
39.12
39.10
39.08
−77.48 −77.46 −77.44 −77.42 −77.40 −77.38 −77.36 −77.34
Longitude

question 13

In [151…

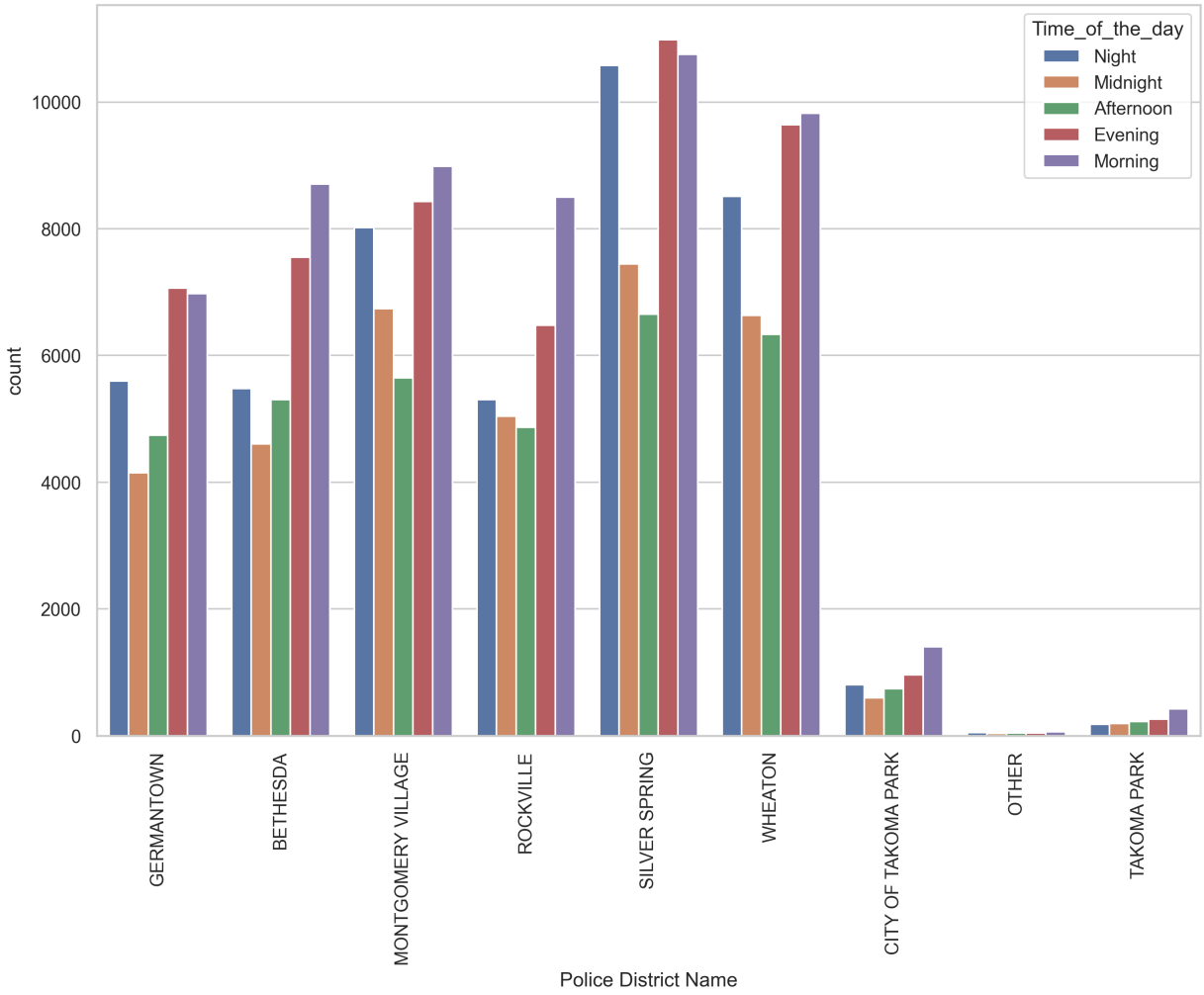
```
plt.figure(figsize=(25,8), dpi = 300)
plt.hist(crime['Place'], bins = 30)# round(len(crime['Place'])**(1/2)))
plt.xticks(rotation = 90)
plt.show()
```



question 14

In [152…

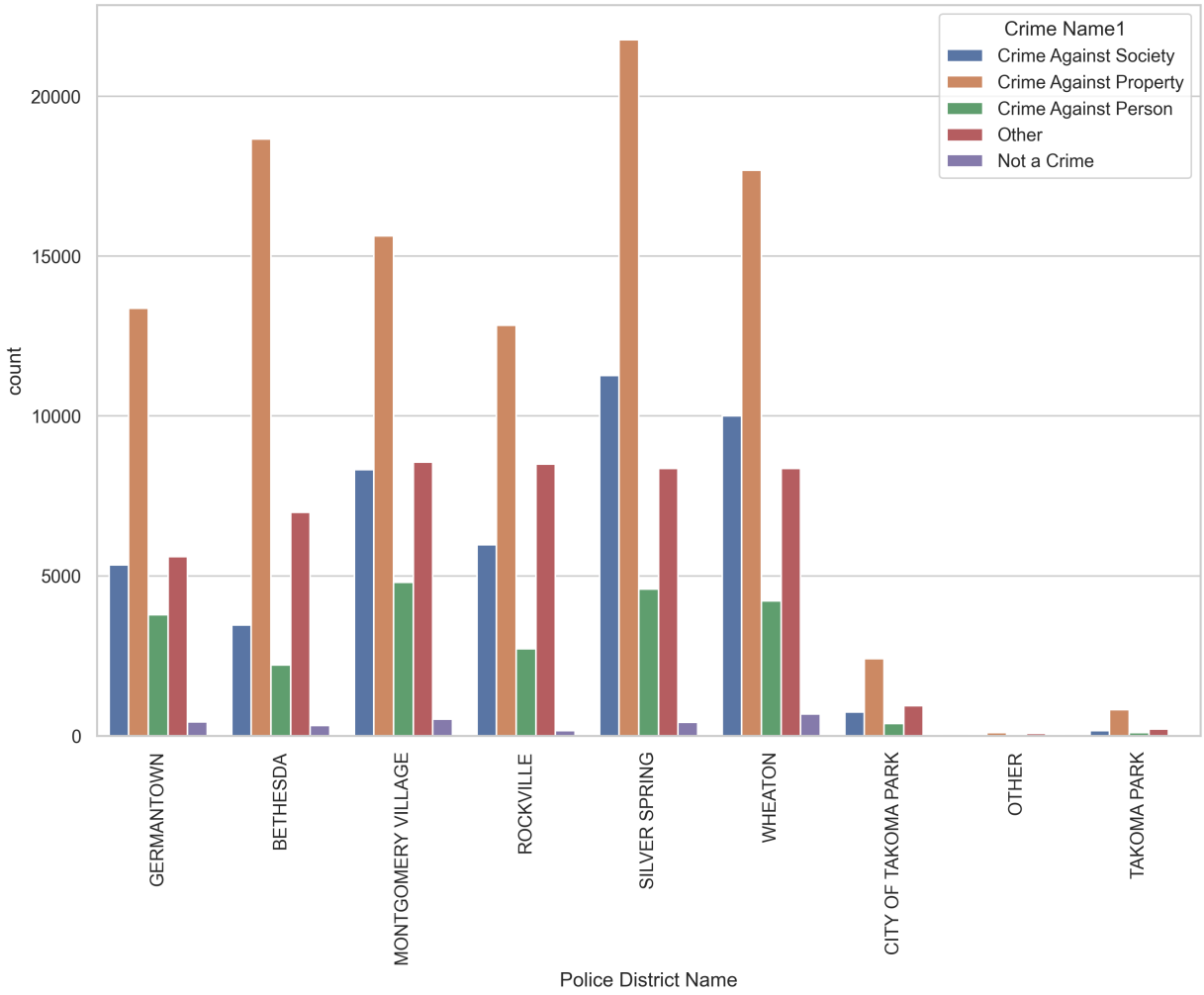
```
plt.figure(figsize=(12,8), dpi = 300)
sns.countplot(x = crime['Police District Name'], hue = crime['Time_of_the

plt.xticks(rotation = 90)
plt.show()
```



question 15a

In [153…

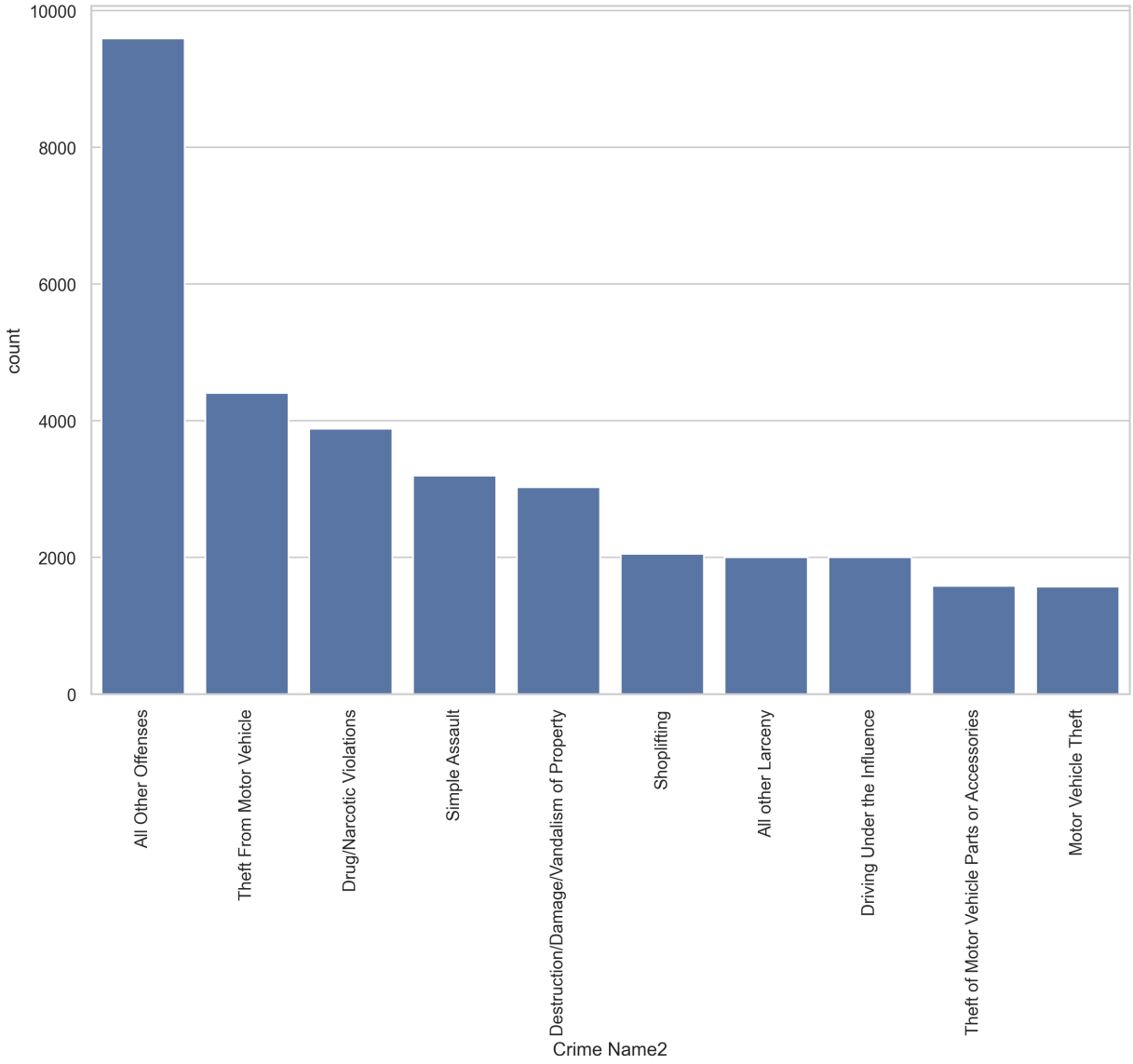
```
plt.figure(figsize=(12,8), dpi = 300)
sns.countplot(x = crime['Police District Name'], hue = crime['Crime Name1

plt.xticks(rotation = 90)
plt.show()
```



question 15b

In [154…
```
silver_crime = crime[crime['Police District Name'] == 'SILVER SPRING']
```

In [155…
```
plt.figure(figsize=(12,8), dpi = 300)
x = silver_crime['Crime Name2'].value_counts().head(10)
sns.countplot(data = silver_crime, x = silver_crime['Crime Name2'], order
plt.xticks(rotation = 90)
plt.show()
```



In [159…
```
# # Example data
# data = silver_crime

# # Create a swarm plot
# sns.swarmplot(x=silver_crime['Crime Name2'], data=data,hu)
# plt.show()
```

In [160…
```
silver_crime_all = silver_crime[silver_crime['Crime Name2'] == 'All Other
```

In [161…
```
silver_crime_all['Crime Name3'].value_counts()
```

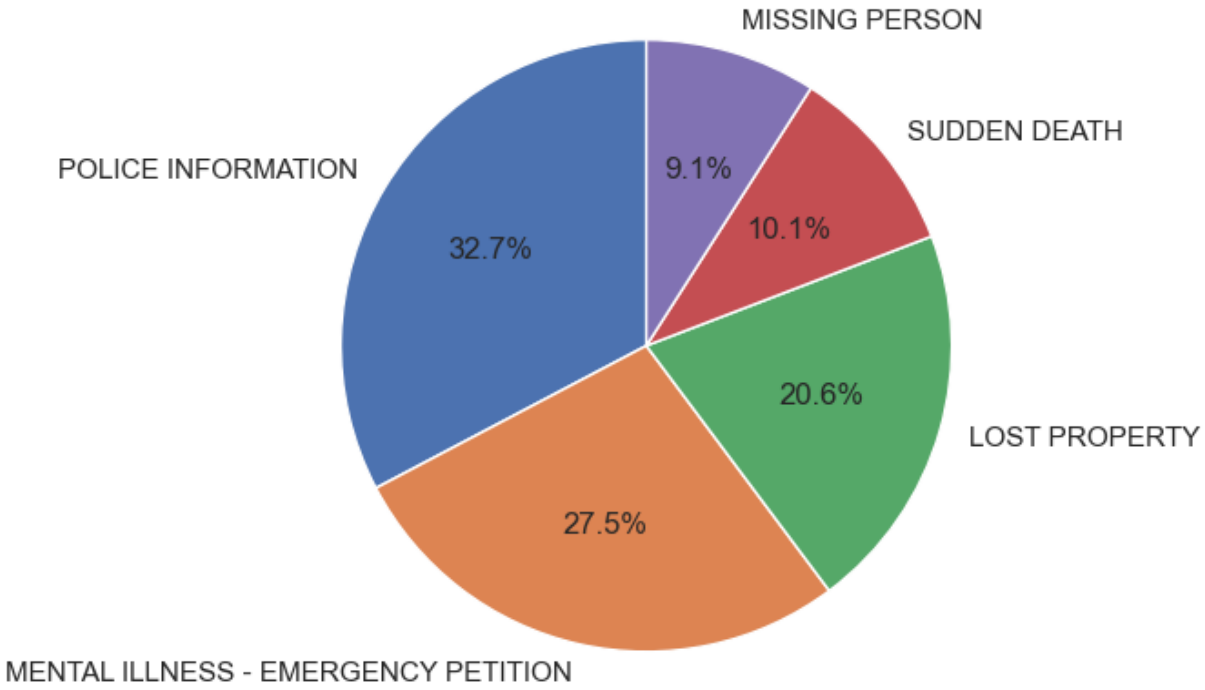
Out[161]:
```
POLICE INFORMATION                                    2353
MENTAL ILLNESS - EMERGENCY PETITION                   1980
LOST PROPERTY                                         1482
SUDDEN DEATH                                           728
MISSING PERSON                                         652
                                                      ...
SMUGGLING - CONTRABAND                                   1
OBSTRUCT POLICE - EVIDENCE - DESTROYING                  1
ESCAPE - FLIGHT TO AVOID                                 1
INVASION PRIVACY (DESCRIBE OFFENSE)                      1
FAMILY OFFENSE - CONTRIBUTING TO THE DELINQUENCY O       1
Name: Crime Name3, Length: 65, dtype: int64
```

In [162…
```
dict1 = dict(silver_crime_all['Crime Name3'].value_counts())
```

In [163…
```
silver_crime_all['Crime Name3'].value_counts().head()
```

Out[163]:
```
POLICE INFORMATION                     2353
MENTAL ILLNESS - EMERGENCY PETITION    1980
LOST PROPERTY                          1482
SUDDEN DEATH                            728
MISSING PERSON                          652
Name: Crime Name3, dtype: int64
```

In [164…
```
import matplotlib.pyplot as plt

# Sample data
labels = dict1.keys()
values = dict1.values()

# Sort the data in descending order
data = list(zip(labels, values))
data.sort(key=lambda x: x[1], reverse=True)

# Extract the top 5 values and labels
top_labels = [item[0] for item in data[:5]]
top_values = [item[1] for item in data[:5]]

# Create a pie chart for the top 5 values
plt.pie(top_values, labels=top_labels, autopct='%1.1f%%', startangle=90)

# Equal aspect ratio ensures that pie is drawn as a circle.
plt.axis('equal')

# Display the pie chart
plt.show()
```


MISSING PERSON
SUDDEN DEATH
POLICE INFORMATION
9.1%
10.1%
32.7%
20.6%
LOST PROPERTY
27.5%
MENTAL ILLNESS - EMERGENCY PETITION

In [ ]:

In [ ]: