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

In [3]:
```
df = pd.read_excel(r"C:\Users\AISWARYA\Downloads\crimedata.xlsx")
```

Scanning the data

In [4]:
```
# The number of rows and columnns in the dataset; returns a tuple
df.shape
```

Out[4]: (549999, 22)

In [7]:
```
# S ummary Statistics of the data
df.describe()
```

Out[7]:

	ID	Date	Beat	District	Ward	Con
count	5.499990e+05	549999	549999.000000	549999.000000	213674.000000	215093.
mean	5.748380e+06	2009-10-05 02:55:46.740996096	1200.364957	11.339984	23.064659	36.
min	6.340000e+02	2001-01-01 00:00:00	111.000000	1.000000	1.000000	1.
25%	1.477682e+06	2001-04-21 02:04:25.500000	621.000000	6.000000	10.000000	22.
50%	1.651346e+06	2001-08-04 19:00:00	1113.000000	10.000000	23.000000	32.
75%	1.327660e+07	2023-11-07 23:00:00	1813.000000	17.000000	34.000000	55.
max	1.348547e+07	2024-05-30 00:00:00	2535.000000	31.000000	50.000000	77.
std	5.669909e+06	NaN	712.157469	7.047229	13.876843	21.

In [9]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 549999 entries, 0 to 549998
Data columns (total 22 columns):
 #   Column                Non-Null Count   Dtype
---  ------                --------------   -----
 0   ID                    549999 non-null  int64
 1   Case Number           549999 non-null  object
 2   Date                  549999 non-null  datetime64[ns]
 3   Block                 549999 non-null  object
 4   IUCR                  549999 non-null  object
 5   Primary Type          549999 non-null  object
 6   Description           549999 non-null  object
 7   Location Description  548326 non-null  object
 8   Arrest                549999 non-null  bool
 9   Domestic              549999 non-null  bool
 10  Beat                  549999 non-null  int64
 11  District              549999 non-null  int64
 12  Ward                  213674 non-null  float64
 13  Community Area        215093 non-null  float64
 14  FBI Code              549999 non-null  object
 15  X Coordinate          543087 non-null  float64
 16  Y Coordinate          543087 non-null  float64
 17  Year                  549999 non-null  int64
 18  Updated On            549999 non-null  datetime64[ns]
 19  Latitude              543087 non-null  float64
 20  Longitude             543087 non-null  float64
 21  Location              543087 non-null  object
dtypes: bool(2), datetime64[ns](2), float64(6), int64(4), object(8)
memory usage: 85.0+ MB
```

In [11]:
```
# First 5 row of our dataset
df.head()
```

Out[11]:

	ID	Case Number	Date	Block	IUCR	Primary Type	Description	Location Description	Arres
0	5741943	HN549294	2007-08-25 09:22:18	074XX N ROGERS AVE	560	ASSAULT	SIMPLE	OTHER	False
1	25953	JE240540	2021-05-24 15:06:00	020XX N LARAMIE AVE	110	HOMICIDE	FIRST DEGREE MURDER	STREET	True
2	26038	JE279849	2021-06-26 09:24:00	062XX N MC CORMICK RD	110	HOMICIDE	FIRST DEGREE MURDER	PARKING LOT	True
3	13279676	JG507211	2023-11-09 07:30:00	019XX W BYRON ST	620	BURGLARY	UNLAWFUL ENTRY	APARTMENT	False
4	13274752	JG501049	2023-11-12 07:59:00	086XX S COTTAGE GROVE AVE	454	BATTERY	AGGRAVATED P.O. - HANDS, FISTS, FEET, NO / MIN...	SMALL RETAIL STORE	True

5 rows × 22 columns

What are out Features?

In [13]:
```
# The names of the features
print("The names of the features :\n",list(df.columns))
```

```
The names of the features :
 ['ID', 'Case Number', 'Date', 'Block', 'IUCR', 'Primary Type', 'Description', 'Lo
cation Description', 'Arrest', 'Domestic', 'Beat', 'District', 'Ward', 'Community
Area', 'FBI Code', 'X Coordinate', 'Y Coordinate', 'Year', 'Updated On', 'Latitud
e', 'Longitude', 'Location']
```

Number of Distinct Crimes in the city of Chicago in 2024

In [15]:
```
# Number of distinct crimes in the city in 2024
crimes = df['Primary Type'].unique()
print("The Number of distinct crimes in Chicago in the year 2023:", len(crimes))
print()
print("The Distinct Crimes are:\n",crimes)
```

```
The Number of distinct crimes in Chicago in the year 2023: 34

The Distinct Crimes are:
 ['ASSAULT' 'HOMICIDE' 'BURGLARY' 'BATTERY' 'THEFT' 'CRIMINAL DAMAGE'
 'DECEPTIVE PRACTICE' 'CRIMINAL SEXUAL ASSAULT'
 'OFFENSE INVOLVING CHILDREN' 'MOTOR VEHICLE THEFT' 'ROBBERY'
 'SEX OFFENSE' 'OTHER OFFENSE' 'WEAPONS VIOLATION' 'STALKING'
 'CRIMINAL TRESPASS' 'PROSTITUTION' 'ARSON' 'NARCOTICS' 'KIDNAPPING'
 'CONCEALED CARRY LICENSE VIOLATION' 'INTERFERENCE WITH PUBLIC OFFICER'
 'PUBLIC PEACE VIOLATION' 'OBSCENITY' 'LIQUOR LAW VIOLATION'
 'INTIMIDATION' 'HUMAN TRAFFICKING' 'GAMBLING' 'CRIM SEXUAL ASSAULT'
 'OTHER NARCOTIC VIOLATION' 'NON-CRIMINAL' 'PUBLIC INDECENCY' 'RITUALISM'
 'DOMESTIC VIOLENCE']
```

Dealing With Missing Values

In [17]:
```
# What are the total missing values in the dataset?
print("Number of Missing Values in the whole dataset:",  df.isna().sum().sum())
```

```
Number of Missing Values in the whole dataset: 707464
```

In [19]:
```
# Let's count number of null entries per feature
missing_values = list(df.isna().sum())
# missing values is a list of the number of missing values in each column

cols = list(df.columns)
col_final = []
for i in range(len(cols)):
    if (missing_values[i] == 0):
        cols[i]="Others"
d = dict(zip(cols, missing_values)) # making a dicionary for the missing values

print("Number of Missing Values per feature >>")
missing_vals = pd.DataFrame(d, index=["Missing Values"]) # Making a custom datafr
missing_vals.head()
```

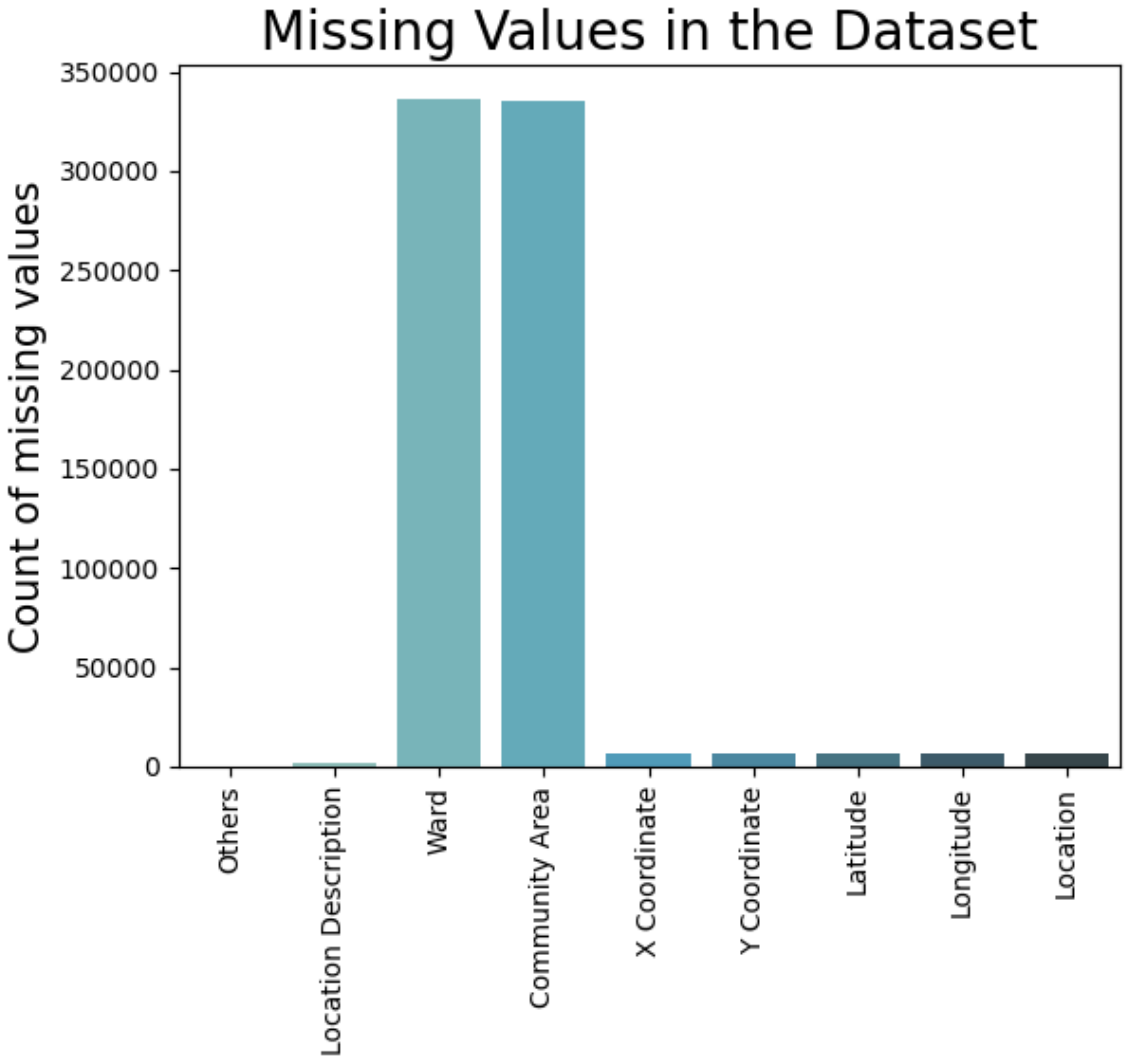
```
Number of Missing Values per feature >>
```

Out[19]:

	Others	Location Description	Ward	Community Area	X Coordinate	Y Coordinate	Latitude	Longitude	Loca
Missing Values	0	1673	336325	334906	6912	6912	6912	6912	

In [21]:
```
# Plotting the missing values in the dataset
x = list(d.keys())
y = list(d.values())
sns.barplot(x=x, y=y, palette="GnBu_d")
plt.xticks(rotation=90)
plt.title("Missing Values in the Dataset", fontdict = {'fontsize': 20})
plt.ylabel("Count of missing values", fontdict={'fontsize': 15})
plt.show()
```

```
C:\Users\AISWARYA\AppData\Local\Temp\ipykernel_16352\1072046772.py:4: FutureWarnin
g:

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

  sns.barplot(x=x, y=y, palette="GnBu_d")
```



In [23]:
```
# The simplest cleaning technique here would be to drop all the rows with here wo
df = df.dropna()
```

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Index: 207594 entries, 1 to 549701
Data columns (total 22 columns):
 #   Column                Non-Null Count   Dtype         
---  ------                --------------   -----         
 0   ID                    207594 non-null  int64         
 1   Case Number           207594 non-null  object        
 2   Date                  207594 non-null  datetime64[ns]
 3   Block                 207594 non-null  object        
 4   IUCR                  207594 non-null  object        
 5   Primary Type          207594 non-null  object        
 6   Description           207594 non-null  object        
 7   Location Description  207594 non-null  object        
 8   Arrest                207594 non-null  bool          
 9   Domestic              207594 non-null  bool          
 10  Beat                  207594 non-null  int64         
 11  District              207594 non-null  int64         
 12  Ward                  207594 non-null  float64       
 13  Community Area        207594 non-null  float64       
 14  FBI Code              207594 non-null  object        
 15  X Coordinate          207594 non-null  float64       
 16  Y Coordinate          207594 non-null  float64       
 17  Year                  207594 non-null  int64         
 18  Updated On            207594 non-null  datetime64[ns]
 19  Latitude              207594 non-null  float64       
 20  Longitude             207594 non-null  float64       
 21  Location              207594 non-null  object        
dtypes: bool(2), datetime64[ns](2), float64(6), int64(4), object(8)
memory usage: 33.7+ MB
```

In [25]:
```
# How much of the data has been retained after this removal?
print(round(262960/265698*100,2),"percentage of the data has been retained.")
```

```
98.97 percentage of the data has been retained.
```

In [27]:
```
# Continuous Variables
cont = df._get_numeric_data().columns
print("The continuous variables are:",list(cont))
```

```
The continuous variables are: ['ID', 'Arrest', 'Domestic', 'Beat', 'District', 'Wa
rd', 'Community Area', 'X Coordinate', 'Y Coordinate', 'Year', 'Latitude', 'Longit
ude']
```

In [29]:
```
# Categorical Variables
print("The categorical variables are:",list(set(df.columns)- set(cont)))
```

```
The categorical variables are: ['Location', 'Case Number', 'IUCR', 'Date', 'Update
d On', 'FBI Code', 'Primary Type', 'Location Description', 'Block', 'Description']
```

Check for available plot styles

In [31]:
```
# use plt.style.available to view all possible styles
```

In [33]:
```
plt.show()  # Uncomment this to display the plot
```

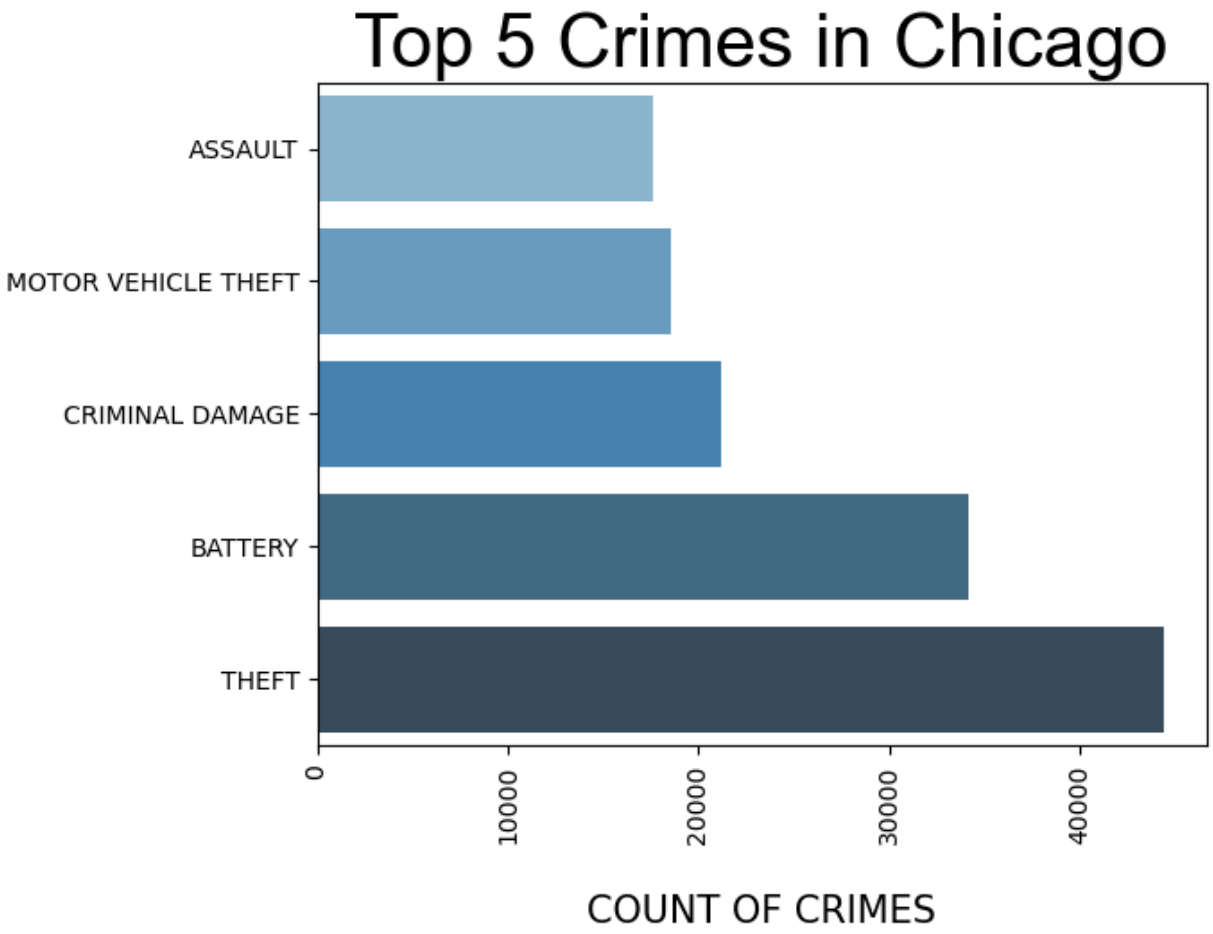
In [35]:
```
pip install seaborn
```

```
Requirement already satisfied: seaborn in c:\users\aiswarya\anaconda3\lib\site-pac
kages (0.13.2)
Requirement already satisfied: numpy!=1.24.0,>=1.20 in c:\users\aiswarya\anaconda3
\lib\site-packages (from seaborn) (1.26.4)
Requirement already satisfied: pandas>=1.2 in c:\users\aiswarya\anaconda3\lib\site
-packages (from seaborn) (2.2.2)
Requirement already satisfied: matplotlib!=3.6.1,>=3.4 in c:\users\aiswarya\anacon
da3\lib\site-packages (from seaborn) (3.9.2)
Requirement already satisfied: contourpy>=1.0.1 in c:\users\aiswarya\anaconda3\lib
\site-packages (from matplotlib!=3.6.1,>=3.4->seaborn) (1.2.0)
Requirement already satisfied: cycler>=0.10 in c:\users\aiswarya\anaconda3\lib\sit
e-packages (from matplotlib!=3.6.1,>=3.4->seaborn) (0.11.0)
Requirement already satisfied: fonttools>=4.22.0 in c:\users\aiswarya\anaconda3\li
b\site-packages (from matplotlib!=3.6.1,>=3.4->seaborn) (4.51.0)
Requirement already satisfied: kiwisolver>=1.3.1 in c:\users\aiswarya\anaconda3\li
b\site-packages (from matplotlib!=3.6.1,>=3.4->seaborn) (1.4.4)
Requirement already satisfied: packaging>=20.0 in c:\users\aiswarya\anaconda3\lib
\site-packages (from matplotlib!=3.6.1,>=3.4->seaborn) (24.1)
Requirement already satisfied: pillow>=8 in c:\users\aiswarya\anaconda3\lib\site-p
ackages (from matplotlib!=3.6.1,>=3.4->seaborn) (10.4.0)
Requirement already satisfied: pyparsing>=2.3.1 in c:\users\aiswarya\anaconda3\lib
\site-packages (from matplotlib!=3.6.1,>=3.4->seaborn) (3.1.2)
Requirement already satisfied: python-dateutil>=2.7 in c:\users\aiswarya\anaconda3
\lib\site-packages (from matplotlib!=3.6.1,>=3.4->seaborn) (2.9.0.post0)
Requirement already satisfied: pytz>=2020.1 in c:\users\aiswarya\anaconda3\lib\sit
e-packages (from pandas>=1.2->seaborn) (2024.1)
Requirement already satisfied: tzdata>=2022.7 in c:\users\aiswarya\anaconda3\lib\s
ite-packages (from pandas>=1.2->seaborn) (2023.3)
Requirement already satisfied: six>=1.5 in c:\users\aiswarya\anaconda3\lib\site-pa
ckages (from python-dateutil>=2.7->matplotlib!=3.6.1,>=3.4->seaborn) (1.16.0)
Note: you may need to restart the kernel to use updated packages.
```

In [37]:
```
import seaborn as sns
import matplotlib.pyplot as plt
```

In [39]:
```
# Import Seaborn and Matplotlib
import seaborn as sns
import matplotlib.pyplot as plt

# Filter out the Top 5 crimes
top_5_crimes = df['Primary Type'].value_counts().sort_values(ascending=False).hea

# Group by 'Primary Type' and count 'ID'
temp = df.groupby('Primary Type', as_index=False).agg({"ID": "count"})
temp = temp.sort_values(by=['ID'], ascending=False).head()
temp = temp.sort_values(by='ID', ascending=True)

# Create the bar plot using Seaborn (Seaborn will automatically handle the stylin
sns.barplot(x='ID', y='Primary Type', data=temp, palette="Blues_d")

# Add aesthetic appeal to the plot
plt.title("Top 5 Crimes in Chicago", fontdict={'fontsize': 30, 'fontname': 'Arial
plt.xlabel("\nCOUNT OF CRIMES", fontdict={'fontsize': 15})
plt.ylabel("")
plt.xticks(rotation=90)

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

```
C:\Users\AISWARYA\AppData\Local\Temp\ipykernel_16352\1190518901.py:14: FutureWarni
ng:

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

  sns.barplot(x='ID', y='Primary Type', data=temp, palette="Blues_d")
```



In [41]:
```
# Doing a bit of df manipulation for using bokeh
temp.head()
temp.columns=['Crime','Number']
temp.index=[0,1,2,3,4]
temp['co-ordinates']=[1,2,3,4,5]
temp.head()
```

Out[41]:

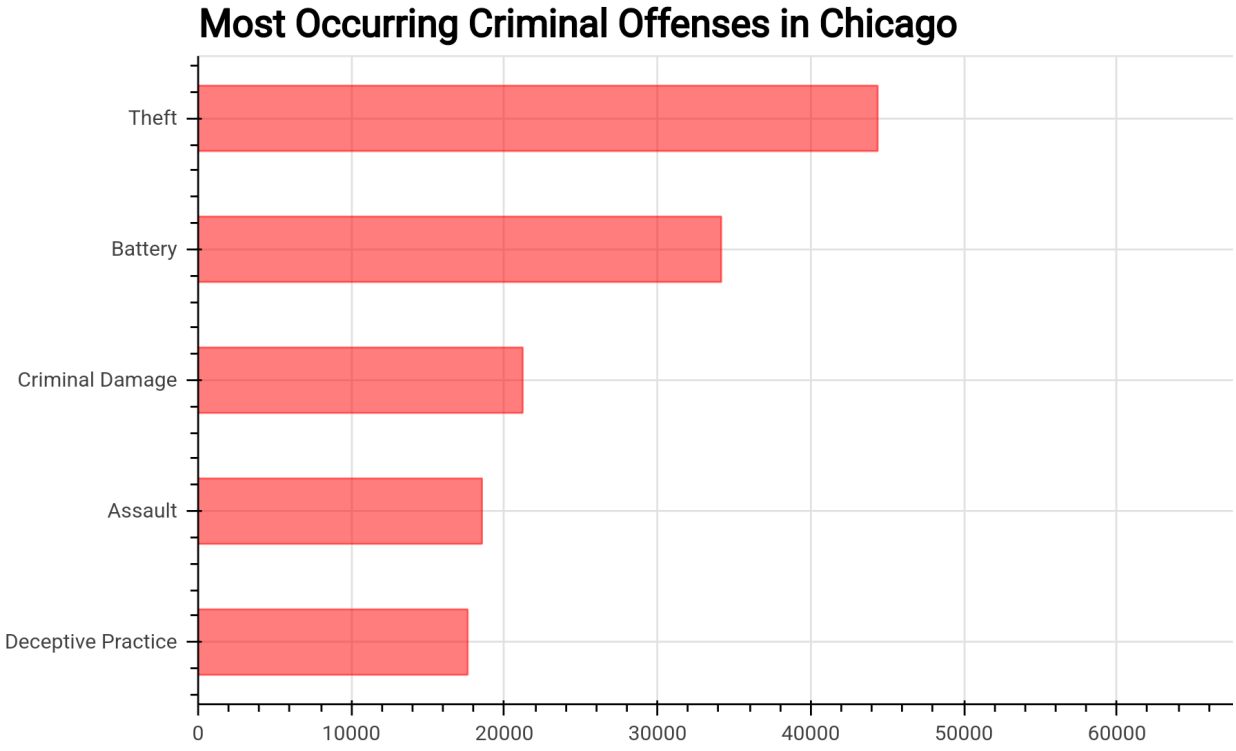
	Crime	Number	co-ordinates
0	ASSAULT	17601	1
1	MOTOR VEHICLE THEFT	18539	2
2	CRIMINAL DAMAGE	21198	3
3	BATTERY	34162	4
4	THEFT	44378	5

In [43]:
```
from bokeh.plotting import figure
from bokeh.io import show, output_notebook
from bokeh.models import ColumnDataSource, HoverTool
```

```
# Inspect the 'temp' DataFrame to confirm the column names
print(temp.columns)

# Create a ColumnDataSource from 'temp' DataFrame
temp_cds = ColumnDataSource(temp)

# Create the figure (use correct axis label attributes)
fig1 = figure(width=700, height=400, title="Most Occurring Criminal Offenses in C
              x_range=(0, 70000))

# Adjust the title styling
fig1.title.align = "left"
fig1.title.text_color = "black"
fig1.title.text_font_size = "20px"

# Create the horizontal bar plot with correct column names for y and right
fig1.hbar(y='co-ordinates', right='Number', source=temp_cds, left=0, color='red',

# Override y-axis labels (ensure that the indices correspond to your data)
fig1.yaxis.major_label_overrides = {5: 'Theft', 4: 'Battery', 3: 'Criminal Damage
                                    1: 'Deceptive Practice'}

# Adding hover tool for interactivity (ensure correct column names in tooltips)
tooltips = [
    ('Number of Crimes', '@Number'),
]

fig1.add_tools(HoverTool(tooltips=tooltips))

# Output the plot in the notebook
output_notebook()
show(fig1)
```

```
Index(['Crime', 'Number', 'co-ordinates'], dtype='object')
```

BokehJS 3.6.0 successfully loaded.



In [45]:
```python
# Assuming df['Date'][20] is a Pandas Timestamp object
t = df['Date'][20]
print("Original timestamp:", t)

# Convert the timestamp to a string with the desired format (e.g., 'YYYY-MM-DD HH
t_str = t.strftime('%Y-%m-%d %I:%M:%S %p')  # %I for 12-hour format, %p for AM/PM
print("Formatted string:", t_str)

# Now slice the string into the date and time parts
s1 = t_str[:11]  # Date part: 'YYYY-MM-DD'
print("Date part:", s1)

s2 = t_str[11:]  # Time part: 'HH:MM:SS AM/PM'
print("Time part:", s2)

# Extract hour, minute, second, and the AM/PM part
hr = s2[:2]   # Hour part (12-hour format)
mins = s2[3:5]  # Minute part
sec = s2[6:8]  # Second part
time_frame = s2[9:]  # AM/PM part

print("Hour:", hr, "Minutes:", mins, "Seconds:", sec, "Time frame:", time_frame)

# Adjust the hour based on AM/PM
if time_frame == 'PM':
    if int(hr) != 12:
        hr = str(int(hr) + 12)  # Convert PM hours to 24-hour format
else:
    if int(hr) == 12:
        hr = '00'  # Convert 12 AM to 00 (midnight)

print("Adjusted hour:", hr, "Minutes:", mins, "Seconds:", sec)
```

```
Original timestamp: 2023-11-10 17:30:00
Formatted string: 2023-11-10 05:30:00 PM
Date part: 2023-11-10
Time part: 05:30:00 PM
Hour: 05 Minutes: 30 Seconds: 00 Time frame: PM
Adjusted hour: 17 Minutes: 30 Seconds: 00
```

In [53]:
```python
from datetime import datetime

# Example inputs (replace these with your actual inputs)
s1 = '2024-12-29'  # Date part: 'YYYY-MM-DD'
hr = '04'  # Hour part (string)
mins = '45'  # Minutes part (string)
sec = '15'  # Seconds part (string)

# Step 1: Correct slicing of date part (s1)
month = s1[5:7]   # Extract month (characters 5-7)
date = s1[8:10]   # Extract day (characters 8-10)
year = s1[:4]     # Extract year (first 4 characters)

# Convert time parts (hr, mins, sec) from string to integer
hr = int(hr)
mins = int(mins)
sec = int(sec)

# Step 2: Create a datetime object
try:
    final_date = datetime(int(year), int(month), int(date), hr, mins, sec)
    print(final_date)  # Output the final datetime object
except ValueError as e:
    print(f"Error creating datetime: {e}")
```

```
2024-12-29 04:45:15
```

In [55]:
```python
# Time Conversion Function
def time_convert(date_time):
    s1 = date_time[:11]
    s2 = date_time[11:]

    month = s1[:2]
    date = s1[3:5]
    year = s1[6:10]

    hr = s2[:2]
    mins = s2[3:5]
    sec = s2[6:8]
    time_frame = s2[9:]
    if(time_frame == 'PM'):
        if (int(hr) != 12):
            hr = str(int(hr) + 12)
    else:
        if(int(hr) == 12):
            hr = '00'

    final_date = datetime(int(year), int(month), int(date), int(hr), int(mins), i
    return final_date
```

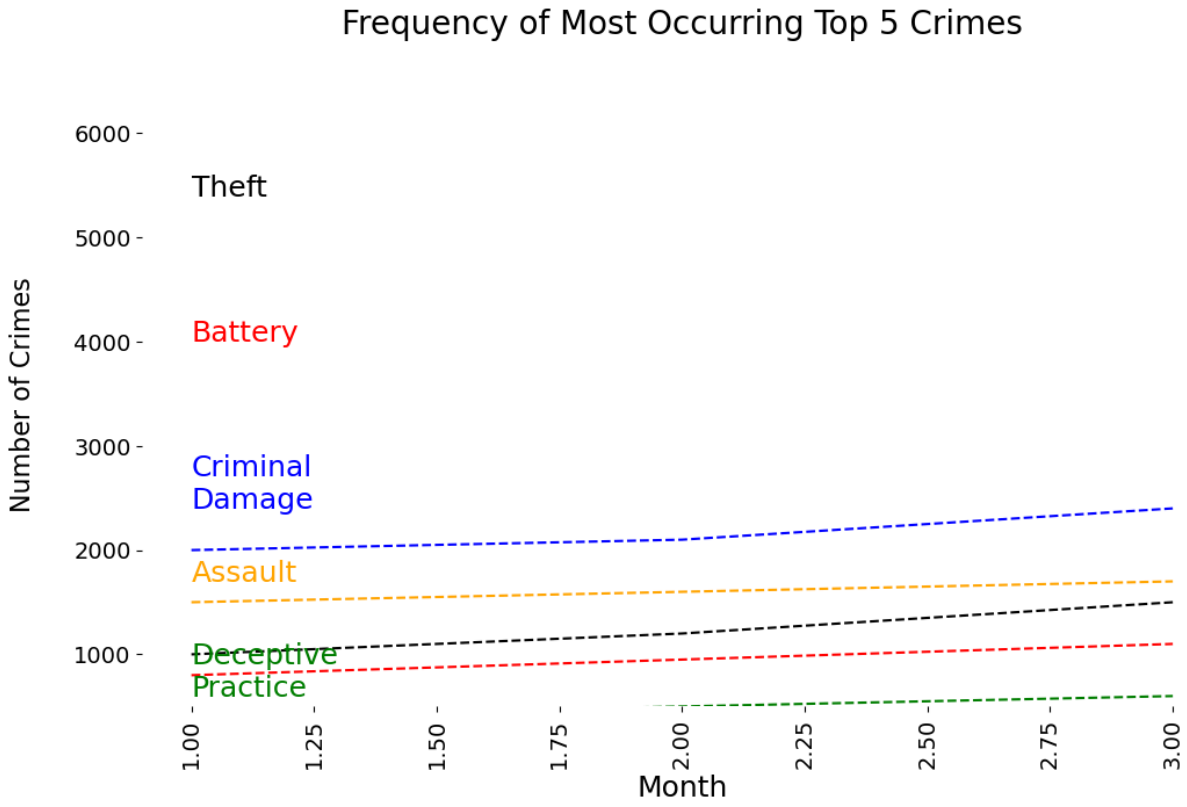
In [57]:
```python
def month(x):
    return x.strftime("%B")
```

```
df['Month'] = df['Date'].apply(month)
```

In [59]:
```
# Frequency of the most occuring crimes over the year 2023

theft_dict ={} # dictionary
battery_dict = {}
crim_dam = {}
assault = {}
dec_prac = {}

months = df["Month"].unique()
for month in months :
    theft_dict[month]=0
    battery_dict[month]=0
    crim_dam[month]=0
    assault[month]=0
    dec_prac[month]=0

for elem in df[df["Primary Type"]=="THEFT"]["Month"]:
    if elem in theft_dict.keys():
        theft_dict[elem] += 1

for elem in df[df["Primary Type"]=="BATTERY"]["Month"]:
    if elem in battery_dict.keys():
        battery_dict[elem] += 1

for elem in df[df["Primary Type"]=="CRIMINAL DAMAGE"]["Month"]:
    if elem in crim_dam.keys():
        crim_dam[elem] += 1

for elem in df[df["Primary Type"]=="ASSAULT"]["Month"]:
    if elem in assault.keys():
        assault[elem] += 1

for elem in df[df["Primary Type"]=="DECEPTIVE PRACTICE"]["Month"]:
    if elem in dec_prac.keys():
        dec_prac[elem] += 1


# Let's order the above dictionaries for proper plotting
months=['January','February','March','April','May','June','July','August','Septem
theft_list = [(k,theft_dict[k]) for k in months]
battery_list = [(k,battery_dict[k]) for k in months]
crim_dam_list = [(k,crim_dam[k]) for k in months]
assault_list = [(k,assault[k]) for k in months]
dec_prac_list = [(k,dec_prac[k]) for k in months]
```

In [61]:
```
import matplotlib.pyplot as plt

# Example data (replace with your actual lists)
theft_list = [(1, 1000), (2, 1200), (3, 1500)]
battery_list = [(1, 800), (2, 950), (3, 1100)]
crim_dam_list = [(1, 2000), (2, 2100), (3, 2400)]
assault_list = [(1, 1500), (2, 1600), (3, 1700)]
dec_prac_list = [(1, 400), (2, 500), (3, 600)]

# You can either specify a style that works, or just omit the line
# plt.style.use('seaborn-dark')  # Comment this out or replace with a valid style
```

```
fig, ax = plt.subplots(figsize=(12,7))

ax.spines["top"].set_visible(False)
ax.spines["bottom"].set_visible(False)
ax.spines["right"].set_visible(False)
ax.spines["left"].set_visible(False)

ax.get_xaxis().tick_bottom()
ax.get_yaxis().tick_left()

plt.xticks(fontsize=14)
plt.yticks(fontsize=14)

plt.ylim(500, 6500)

x = [z[0] for z in theft_list]
y = [z[1] for z in theft_list]
ax.plot(x, y, color="black")
ax.lines[0].set_linestyle("--")

x = [z[0] for z in battery_list]
y = [z[1] for z in battery_list]
ax.plot(x, y, color="red")
ax.lines[1].set_linestyle("--")

x = [z[0] for z in crim_dam_list]
y = [z[1] for z in crim_dam_list]
ax.plot(x, y, color="blue")
ax.lines[2].set_linestyle("--")

x = [z[0] for z in assault_list]
y = [z[1] for z in assault_list]
ax.plot(x, y, color="orange")
ax.lines[3].set_linestyle("--")

x = [z[0] for z in dec_prac_list]
y = [z[1] for z in dec_prac_list]
ax.plot(x, y, color="green")
ax.lines[4].set_linestyle("--")

for tick in ax.get_xticklabels():
    tick.set_rotation(90)

plt.text(1, 5400, "Theft", fontsize=18)
plt.text(1, 4000, "Battery", fontsize=18, color="red")
plt.text(1, 2400, "Criminal\nDamage", fontsize=18, color="blue")
plt.text(1, 1700, "Assault", fontsize=18, color="orange")
plt.text(1, 600, "Deceptive\nPractice", fontsize=18, color="green")

ax.set_title("Frequency of Most Occurring Top 5 Crimes\n", fontsize=20)
ax.set_xlabel("Month", fontsize=18)
ax.set_ylabel("Number of Crimes\n", fontsize=16)

plt.show()
```



In [63]:
```
# Dataframes for each crime (Will be used in the further parts of our analysis)

theft_df = df[df['Primary Type']=='THEFT']
battery_df = df[df['Primary Type']=='BATTERY']
crim_dam_df = df[df['Primary Type']=='CRIMINAL DAMAGE']
assault_df = df[df['Primary Type']=='ASSAULT']
dec_prac_df = df[df['Primary Type']=='DECEPTIVE PRACTICE']
```

2.Arrests and the state of Chicago

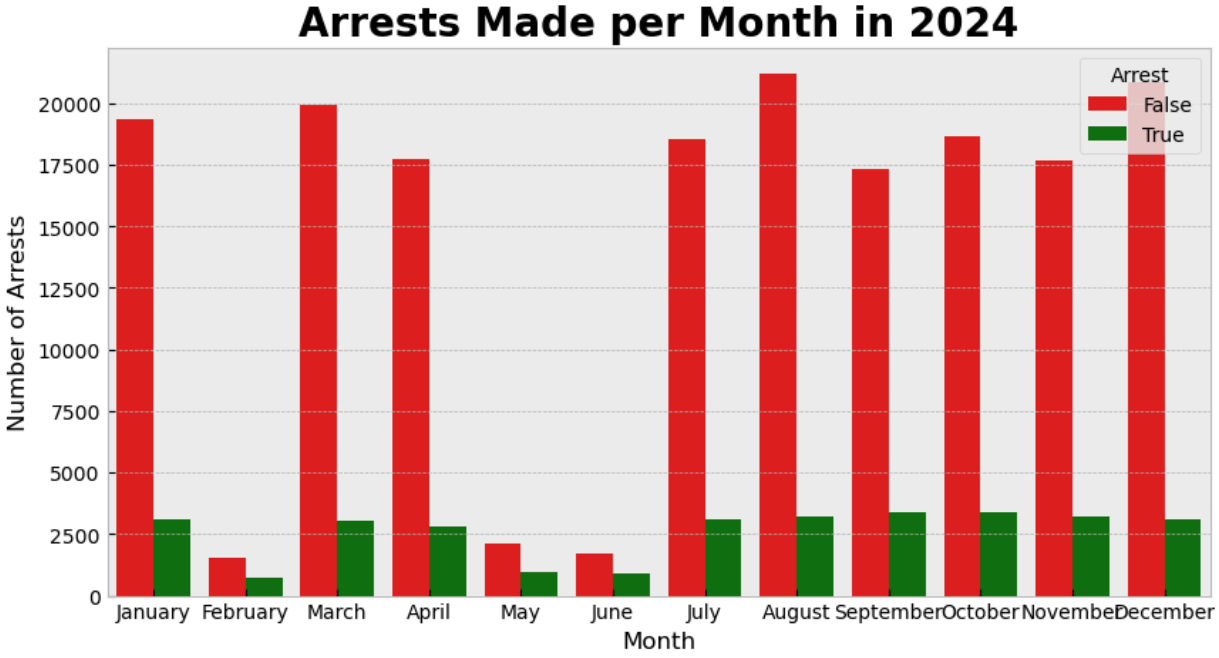
In [65]:
```
import pandas as pd

# Example DataFrame setup (replace with your actual df)
# df = pd.DataFrame({'Arrest': [True, False, True, False, True, False]})

# Check for value counts in the 'Arrest' column
l = df["Arrest"].value_counts(dropna=False)

# If 'True' and 'False' are not present, you can handle this gracefully
false = l.get(False, 0)  # If 'False' is missing, set to 0
true = l.get(True, 0)    # If 'True' is missing, set to 0

# Calculating the percentage of no arrests (False)
if false + true > 0:
    no_arrest_percentage = (false / (false + true)) * 100
    print(f"Percentage of no arrests of all reported crimes: {no_arrest_percentag
else:
    print("No data available for arrest status.")
```

```
Percentage of no arrests of all reported crimes: 85.1007254544929%
```

Distribution of arrests across the months

In [67]:
```
# How are arrests spread out across the months
plt.style.use('bmh')

fig, ax = plt.subplots(figsize=(10, 5))
ax = sns.countplot(x="Month",
                   hue='Arrest',
                   data=df[['Month','Arrest']],
                   palette=['Red', 'Green'])
months = ['January','February','March','April','May','June','July',\
             'August','September','October','November','December']

ax.set(title='Arrests Made per Month in 2024', xlabel='Month', ylabel='Number of A
plt.title('Arrests Made per Month in 2024', fontdict={'fontsize': 20, 'color': 'b
plt.show()
```

```
C:\Users\AISWARYA\AppData\Local\Temp\ipykernel_16352\2014628264.py:12: UserWarnin
g: set_ticklabels() should only be used with a fixed number of ticks, i.e. after s
et_ticks() or using a FixedLocator.
  ax.set(title='Arrests Made per Month in 2024', xlabel='Month', ylabel='Number of
Arrests', xticklabels=months)
```



How do arrests very among the type of crime?

In [69]:
```
# let's look at the pandas groupby function
#arrest_crime = df.groupby(['Primary Type', 'Arrest'])['ID'].count()
arrest_crime = df.groupby(['Primary Type', 'Arrest']).agg({'Arrest':"count"})
arrest_crime.columns = ["Count"]
# arrest_crime
```

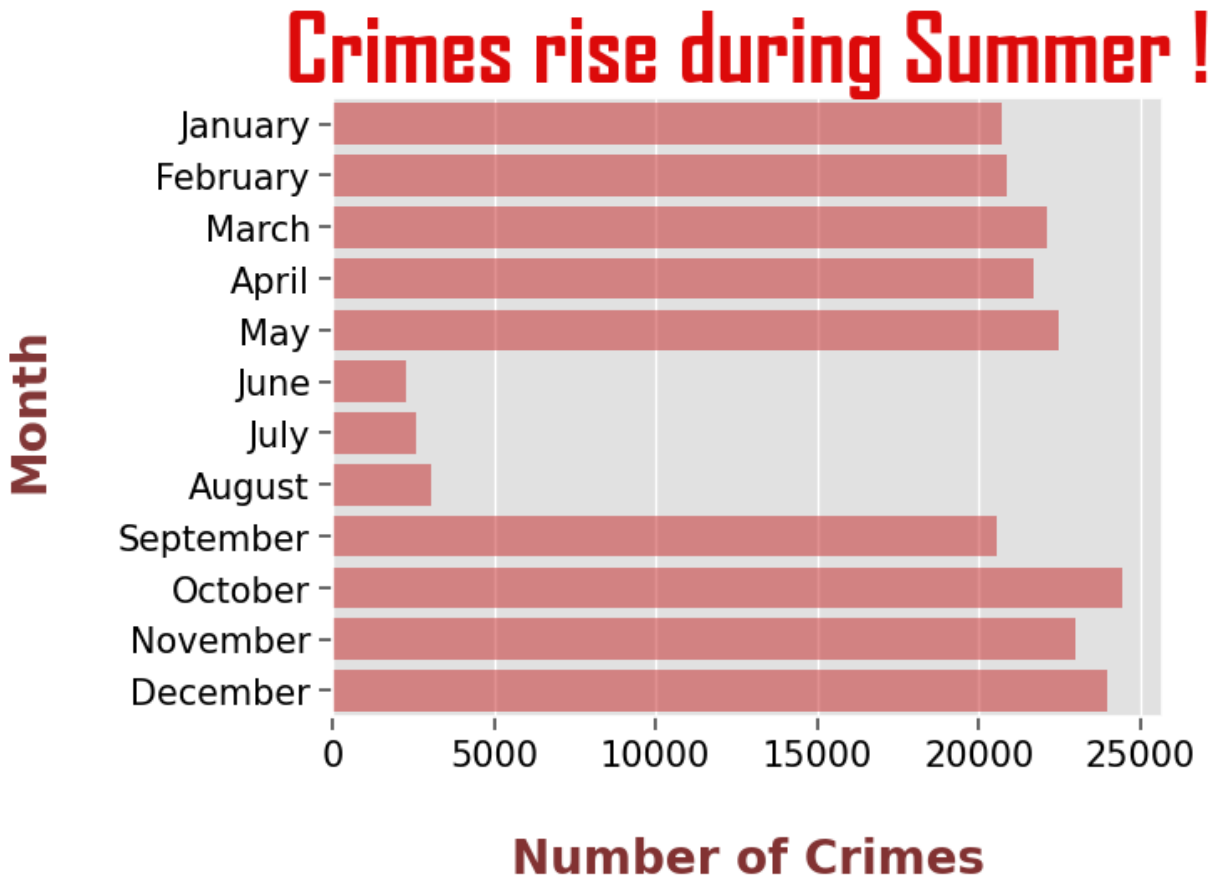
3. Crime vs Time

Distribution of crimes across the months

In [71]:
```
# Set plot style
plt.style.use('ggplot')
```

```
sns.set_context('notebook')

# Code to plot
sns.countplot(y='Month', data=df, palette=["#DF0D0D"], order=['January', 'Februar

# Aesthetic appeal of the plot
plt.title("Crimes rise during Summer !", fontdict={'fontsize': 40, 'color': '#DF0
plt.ylabel("Month\n", fontdict={'fontsize': 20}, weight="bold", color="#833636")
plt.xlabel("\nNumber of Crimes", fontdict={'fontsize': 20}, weight="bold", color=

plt.xticks(fontsize=15,color='black')
plt.yticks(fontsize=15, color='black')
plt.show()
```

```
C:\Users\AISWARYA\AppData\Local\Temp\ipykernel_16352\3270041557.py:6: FutureWarnin
g:

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

  sns.countplot(y='Month', data=df, palette=["#DF0D0D"], order=['January', 'Februa
ry', 'March', 'April', 'May', 'June', "July", 'August', 'September', 'October', 'N
ovember', 'December'], alpha=0.5)
C:\Users\AISWARYA\AppData\Local\Temp\ipykernel_16352\3270041557.py:6: UserWarning:
The palette list has fewer values (1) than needed (12) and will cycle, which may p
roduce an uninterpretable plot.
  sns.countplot(y='Month', data=df, palette=["#DF0D0D"], order=['January', 'Februa
ry', 'March', 'April', 'May', 'June', "July", 'August', 'September', 'October', 'N
ovember', 'December'], alpha=0.5)
```



In [73]:
```
def hour(x):
    return x.strftime("%H")
df['Hour_Day'] = df['Date'].apply(hour)
```

What are the most unsafest hours?

In [75]:
```
import matplotlib.pyplot as plt
print(plt.style.available)
```

```
['Solarize_Light2', '_classic_test_patch', '_mpl-gallery', '_mpl-gallery-nogrid',
'bmh', 'classic', 'dark_background', 'fast', 'fivethirtyeight', 'ggplot', 'grayscale', 'seaborn-v0_8', 'seaborn-v0_8-bright', 'seaborn-v0_8-colorblind', 'seaborn-v0_8-dark', 'seaborn-v0_8-dark-palette', 'seaborn-v0_8-darkgrid', 'seaborn-v0_8-deep', 'seaborn-v0_8-muted', 'seaborn-v0_8-notebook', 'seaborn-v0_8-paper', 'seaborn-v0_8-pastel', 'seaborn-v0_8-poster', 'seaborn-v0_8-talk', 'seaborn-v0_8-ticks', 'seaborn-v0_8-white', 'seaborn-v0_8-whitegrid', 'tableau-colorblind10']
```

In [77]:
```
pip install seaborn
```

```
Requirement already satisfied: seaborn in c:\users\aiswarya\anaconda3\lib\site-packages (0.13.2)
Requirement already satisfied: numpy!=1.24.0,>=1.20 in c:\users\aiswarya\anaconda3\lib\site-packages (from seaborn) (1.26.4)
Requirement already satisfied: pandas>=1.2 in c:\users\aiswarya\anaconda3\lib\site-packages (from seaborn) (2.2.2)
Requirement already satisfied: matplotlib!=3.6.1,>=3.4 in c:\users\aiswarya\anaconda3\lib\site-packages (from seaborn) (3.9.2)
Requirement already satisfied: contourpy>=1.0.1 in c:\users\aiswarya\anaconda3\lib\site-packages (from matplotlib!=3.6.1,>=3.4->seaborn) (1.2.0)
Requirement already satisfied: cycler>=0.10 in c:\users\aiswarya\anaconda3\lib\site-packages (from matplotlib!=3.6.1,>=3.4->seaborn) (0.11.0)
Requirement already satisfied: fonttools>=4.22.0 in c:\users\aiswarya\anaconda3\lib\site-packages (from matplotlib!=3.6.1,>=3.4->seaborn) (4.51.0)
Requirement already satisfied: kiwisolver>=1.3.1 in c:\users\aiswarya\anaconda3\lib\site-packages (from matplotlib!=3.6.1,>=3.4->seaborn) (1.4.4)
Requirement already satisfied: packaging>=20.0 in c:\users\aiswarya\anaconda3\lib\site-packages (from matplotlib!=3.6.1,>=3.4->seaborn) (24.1)
Requirement already satisfied: pillow>=8 in c:\users\aiswarya\anaconda3\lib\site-packages (from matplotlib!=3.6.1,>=3.4->seaborn) (10.4.0)
Requirement already satisfied: pyparsing>=2.3.1 in c:\users\aiswarya\anaconda3\lib\site-packages (from matplotlib!=3.6.1,>=3.4->seaborn) (3.1.2)
Requirement already satisfied: python-dateutil>=2.7 in c:\users\aiswarya\anaconda3\lib\site-packages (from matplotlib!=3.6.1,>=3.4->seaborn) (2.9.0.post0)
Requirement already satisfied: pytz>=2020.1 in c:\users\aiswarya\anaconda3\lib\site-packages (from pandas>=1.2->seaborn) (2024.1)
Requirement already satisfied: tzdata>=2022.7 in c:\users\aiswarya\anaconda3\lib\site-packages (from pandas>=1.2->seaborn) (2023.3)
Requirement already satisfied: six>=1.5 in c:\users\aiswarya\anaconda3\lib\site-packages (from python-dateutil>=2.7->matplotlib!=3.6.1,>=3.4->seaborn) (1.16.0)
Note: you may need to restart the kernel to use updated packages.
```

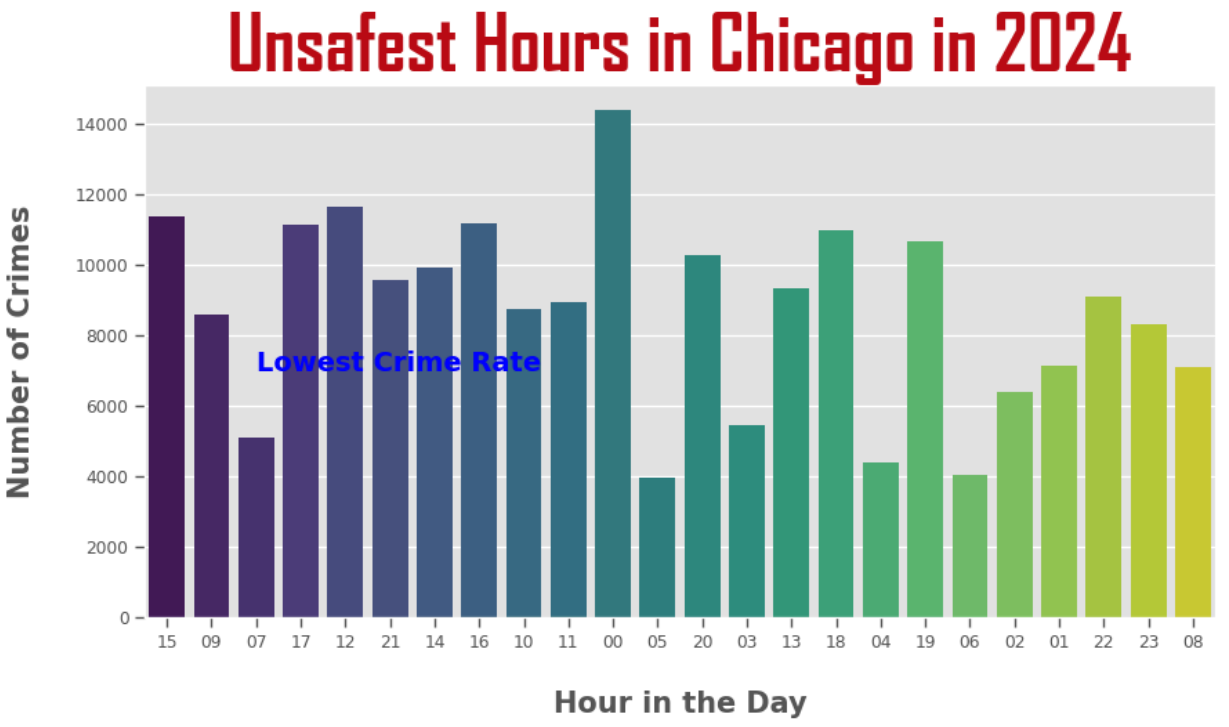
In [79]:
```
plt.style.use('ggplot')  # Or any other valid style
```

In [81]:
```
import seaborn as sns
sns.set_context("paper")
```

In [83]:
```
import matplotlib.pyplot as plt
import seaborn as sns
```

```
# Set a default plot style
plt.style.use('ggplot')  # Or any other available style
sns.set_context('paper')

# Create the figure and axis
fig, ax = plt.subplots(figsize=(10, 5))

# Plot the count plot
sns.countplot(x='Hour_Day', data=df, palette="viridis")

# Add title and labels
plt.title("Unsafest Hours in Chicago in 2024",
          fontdict={'fontsize': 40, 'color': '#bb0e14', 'fontname': 'Agency FB'},
plt.xlabel("\nHour in the Day", fontdict={'fontsize': 15}, weight='bold')
plt.ylabel("Number of Crimes\n", fontdict={'fontsize': 15}, weight="bold")

# Add text to the plot
plt.text(2, 7000, 'Lowest Crime Rate', fontdict={'fontsize': 14, 'color':"blue" }

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

```
C:\Users\AISWARYA\AppData\Local\Temp\ipykernel_16352\1373713228.py:12: FutureWarni
ng:

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

  sns.countplot(x='Hour_Day', data=df, palette="viridis")
```

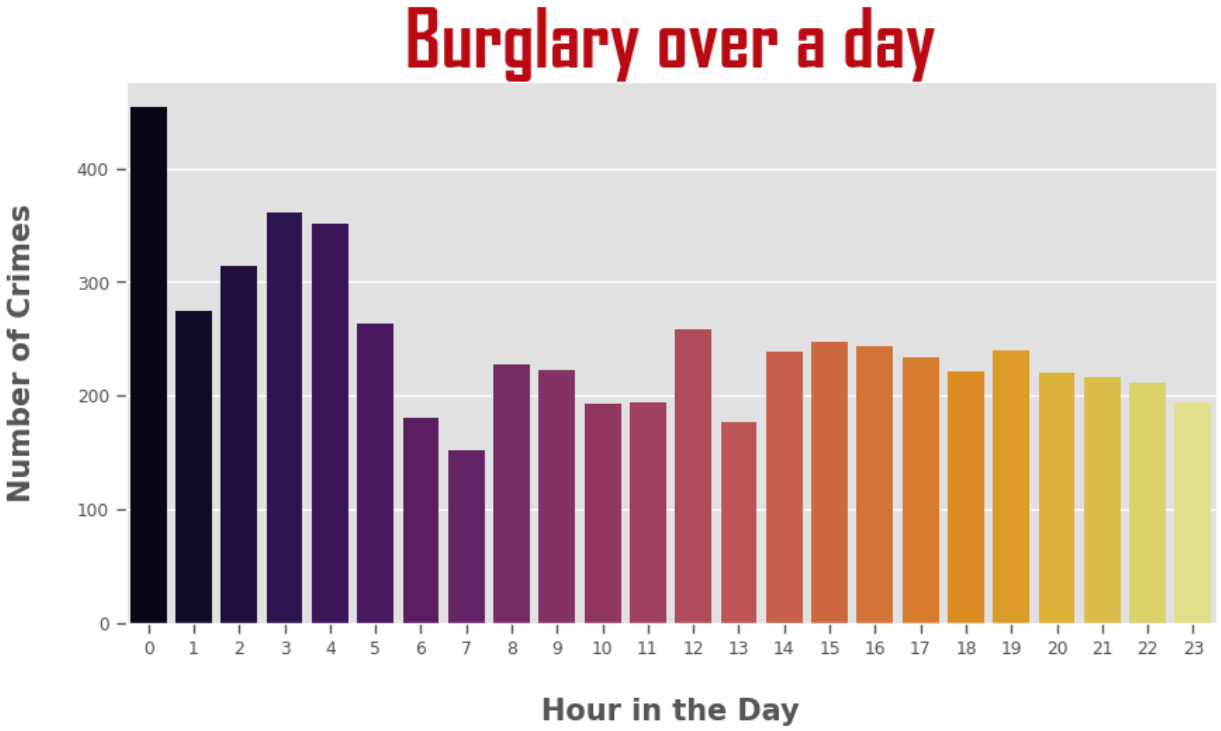


Is your house safe from a burglary during the day?

In [85]:
```
# analyse only for burglary
burglary_df = df[df['Primary Type']=='BURGLARY']
hours = [int(x) for x in list(burglary_df['Hour_Day'].unique())]
```

```
hours = sorted(hours)
# print(hours)

bur_cri = list(burglary_df['Hour_Day'].value_counts().sort_index())
# print(bur_cri)

fig, ax = plt.subplots(figsize=(10, 5))
sns.barplot(x=hours, y=bur_cri, palette='inferno')

# Aesthetic appeal
plt.title("Burglary over a day", fontdict={'fontsize': 40, 'color': '#bb0e14','fo
plt.xlabel("\nHour in the Day", fontdict={'fontsize': 15}, weight='bold')
plt.ylabel("Number of Crimes\n", fontdict={'fontsize': 15}, weight="bold")

# show plot
plt.show()
```

```
C:\Users\AISWARYA\AppData\Local\Temp\ipykernel_16352\1473634055.py:11: FutureWarni
ng:

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

  sns.barplot(x=hours, y=bur_cri, palette='inferno')
```

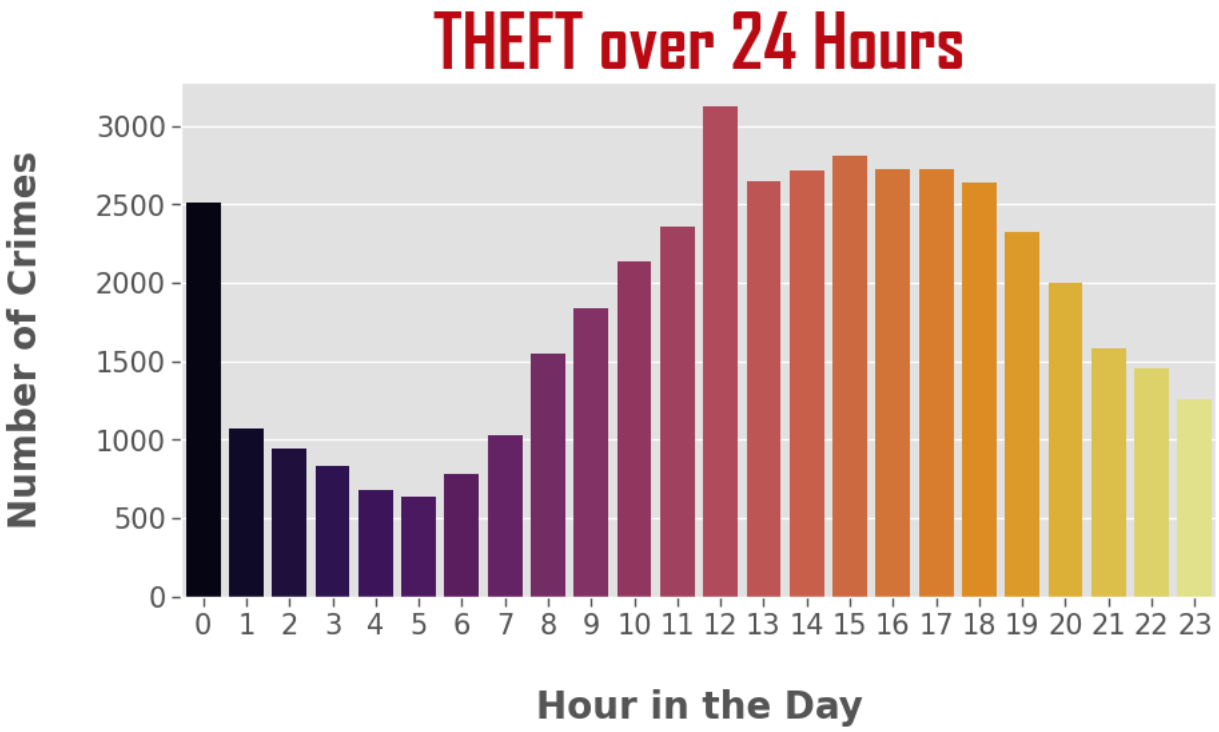


Visualize a Crime's Pattern for 24 hours in 2024

In [87]:
```
#sns.set_style('darkgrid')
plt.style.use('ggplot')
# An analysis of the 24 hour pattern for crimes
c = input("Enter the crime you wish to see the 24 hour pattern off >> ")

crime_df = df[df['Primary Type']==c.upper()]
hours = [int(x) for x in list(crime_df['Hour_Day'].unique())]
hours = sorted(hours)
# print(hours)
```

```
h_cri = list(crime_df['Hour_Day'].value_counts().sort_index())
# print(bur_cri)

fig, ax = plt.subplots(figsize=(10, 5))
sns.barplot(x=hours, y=h_cri, palette='inferno')

# Aesthetic appeal
tit = c.upper()+" over 24 Hours"
plt.title(tit, fontdict={'fontsize': 40, 'color': '#bb0e14','fontname':'Agency FB
plt.xlabel("\nHour in the Day", fontdict={'fontsize': 20}, weight='bold')
plt.ylabel("Number of Crimes\n", fontdict={'fontsize': 20}, weight="bold")
plt.yticks(fontsize=15)
plt.xticks(fontsize=15)

# show plot
plt.show()
```

```
C:\Users\AISWARYA\AppData\Local\Temp\ipykernel_16352\4275540825.py:15: FutureWarni
ng:

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

  sns.barplot(x=hours, y=h_cri, palette='inferno')
```



3. Crime vs Locations

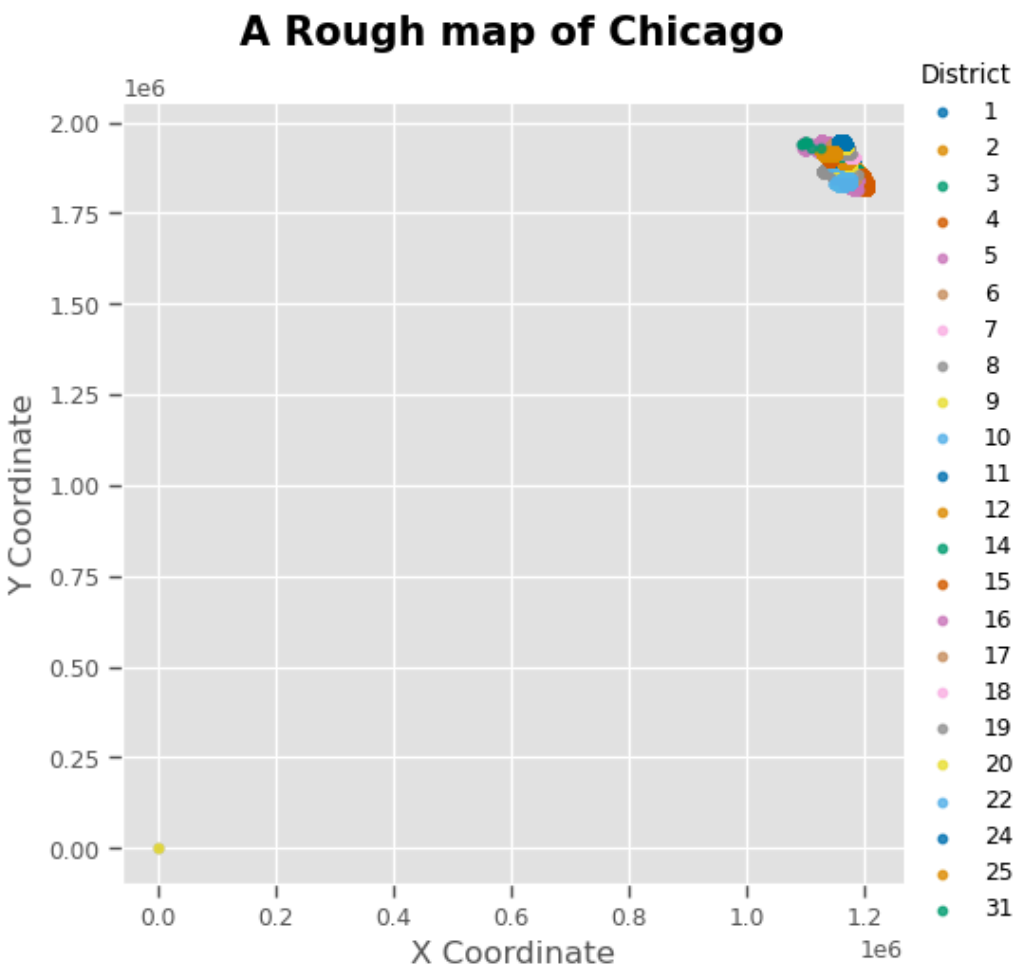
In [89]:
```
print(df.columns)
```

```
Index(['ID', 'Case Number', 'Date', 'Block', 'IUCR', 'Primary Type',
       'Description', 'Location Description', 'Arrest', 'Domestic', 'Beat',
       'District', 'Ward', 'Community Area', 'FBI Code', 'X Coordinate',
       'Y Coordinate', 'Year', 'Updated On', 'Latitude', 'Longitude',
       'Location', 'Month', 'Hour_Day'],
      dtype='object')
```

In [91]:
```
print(df[['X Coordinate', 'Y Coordinate']].isnull().sum())
```

```
X Coordinate    0
Y Coordinate    0
dtype: int64
```

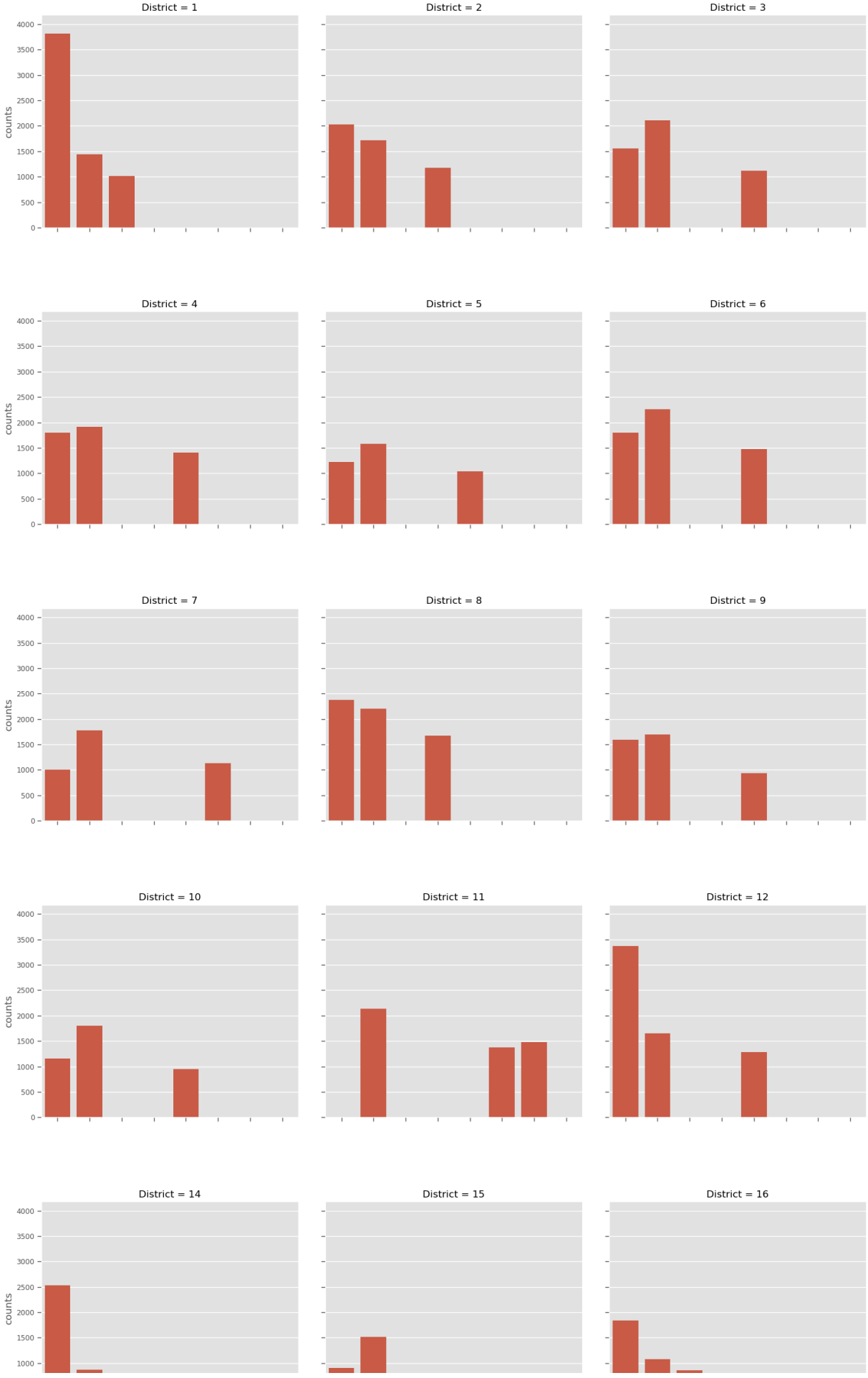
In [93]:
```
import seaborn as sns
import matplotlib.pyplot as plt

# Check if necessary columns exist
if 'X Coordinate' in df.columns and 'Y Coordinate' in df.columns:
    # Check for missing values
    if df[['X Coordinate', 'Y Coordinate']].isnull().sum().sum() == 0:
        # Plot the map with lmplot
        sns.lmplot(x='X Coordinate',  # Specify column names using 'x' and 'y'
                   y='Y Coordinate',
                   data=df,  # Pass the data argument once
                   fit_reg=False,  # No regression line
                   hue="District",  # District-based color grouping
                   palette='colorblind',  # Color palette for better visibility
                   height=5,
                   scatter_kws={"marker": "+", "s": 10})

        ax = plt.gca()  # Get current axis
        ax.set_title("A Rough map of Chicago\n", fontdict={'fontsize': 15}, weigh
        plt.show()
    else:
        print("There are missing values in the coordinates columns.")
else:
    print("X Coordinate or Y Coordinate columns not found in the dataframe.")
```

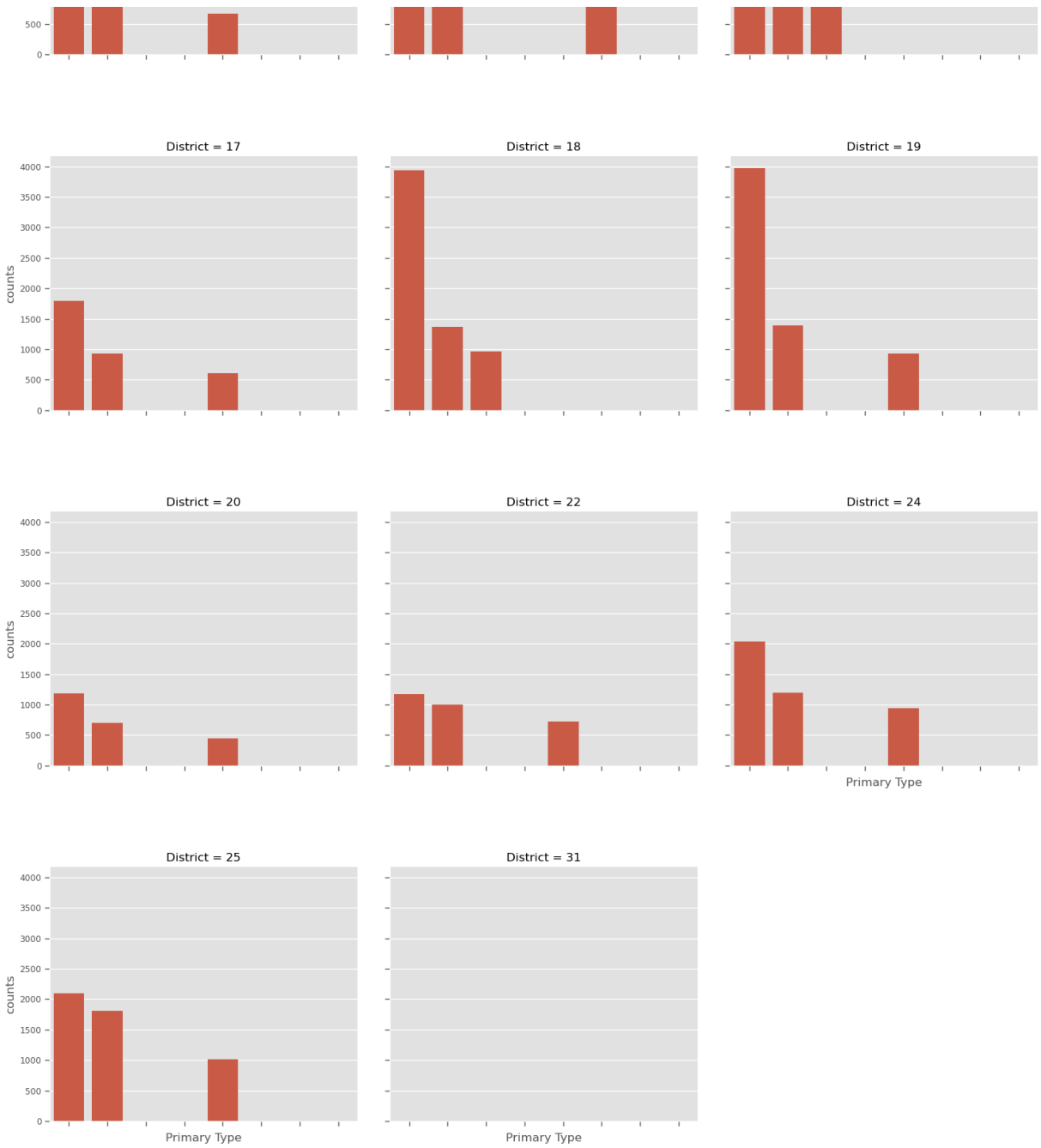


Most Common Occurences per District

In [95]:
```
import seaborn as sns
import matplotlib.pyplot as plt

# Grouping the data by 'District' and 'Primary Type', and selecting the top 3 per
top = df.groupby(['District', 'Primary Type']).size().reset_index(name='counts')
top = top.groupby('District').apply(lambda x: x.sort_values('counts', ascending=F

# Create the factor plot (catplot)
g = sns.catplot(x="Primary Type", y="counts", col="District", col_wrap=3, data=to

# Adjusting the x-axis labels' appearance
for ax in g.axes.flat:
    ax.set_xticklabels(ax.get_xticklabels(), rotation=45, ha='right')

# Adjusting the layout for better spacing
plt.subplots_adjust(hspace=0.4)
plt.show()
```

```
C:\Users\AISWARYA\AppData\Local\Temp\ipykernel_16352\2279115143.py:6: DeprecationW
arning: DataFrameGroupBy.apply operated on the grouping columns. This behavior is
deprecated, and in a future version of pandas the grouping columns will be exclude
d from the operation. Either pass `include_groups=False` to exclude the groupings
or explicitly select the grouping columns after groupby to silence this warning.
  top = top.groupby('District').apply(lambda x: x.sort_values('counts', ascending=
False).head(3)).reset_index(drop=True)
C:\Users\AISWARYA\AppData\Local\Temp\ipykernel_16352\2279115143.py:13: UserWarnin
g: set_ticklabels() should only be used with a fixed number of ticks, i.e. after s
et_ticks() or using a FixedLocator.
  ax.set_xticklabels(ax.get_xticklabels(), rotation=45, ha='right')
C:\Users\AISWARYA\AppData\Local\Temp\ipykernel_16352\2279115143.py:13: UserWarnin
g: set_ticklabels() should only be used with a fixed number of ticks, i.e. after s
et_ticks() or using a FixedLocator.
  ax.set_xticklabels(ax.get_xticklabels(), rotation=45, ha='right')
C:\Users\AISWARYA\AppData\Local\Temp\ipykernel_16352\2279115143.py:13: UserWarnin
g: set_ticklabels() should only be used with a fixed number of ticks, i.e. after s
et_ticks() or using a FixedLocator.
  ax.set_xticklabels(ax.get_xticklabels(), rotation=45, ha='right')
```


District = 1
District = 2
District = 3
District = 4
District = 5
District = 6
District = 7
District = 8
District = 9
District = 10
District = 11
District = 12
District = 14
District = 15
District = 16
counts
0
500
1000
1500
2000
2500
3000
3500
4000



In [97]:

```
a=''
a=input("enter the gang criminal activity for frequency of a crime in location ")
a=a.upper()
var=df[df['Primary Type']==a]
unique_locations = var['Location'].value_counts()
print("%%%%%%%%%%%%%%%%%%%%%%%%%%%%%% ",len(unique_locations))
#print("%%%%%%%%%%%%%%%%%%%%%%%%%%%%%% ",unique_locations)


unique_locations.index


CR_index = pd.DataFrame({"Raw_String" : unique_locations.index, "ValueCount":uniq
CR_index.index = range(len(unique_locations))
print(CR_index.head())
print("valuecount",len(CR_index['ValueCount']))
def Location_extractor(Raw_Str):
```

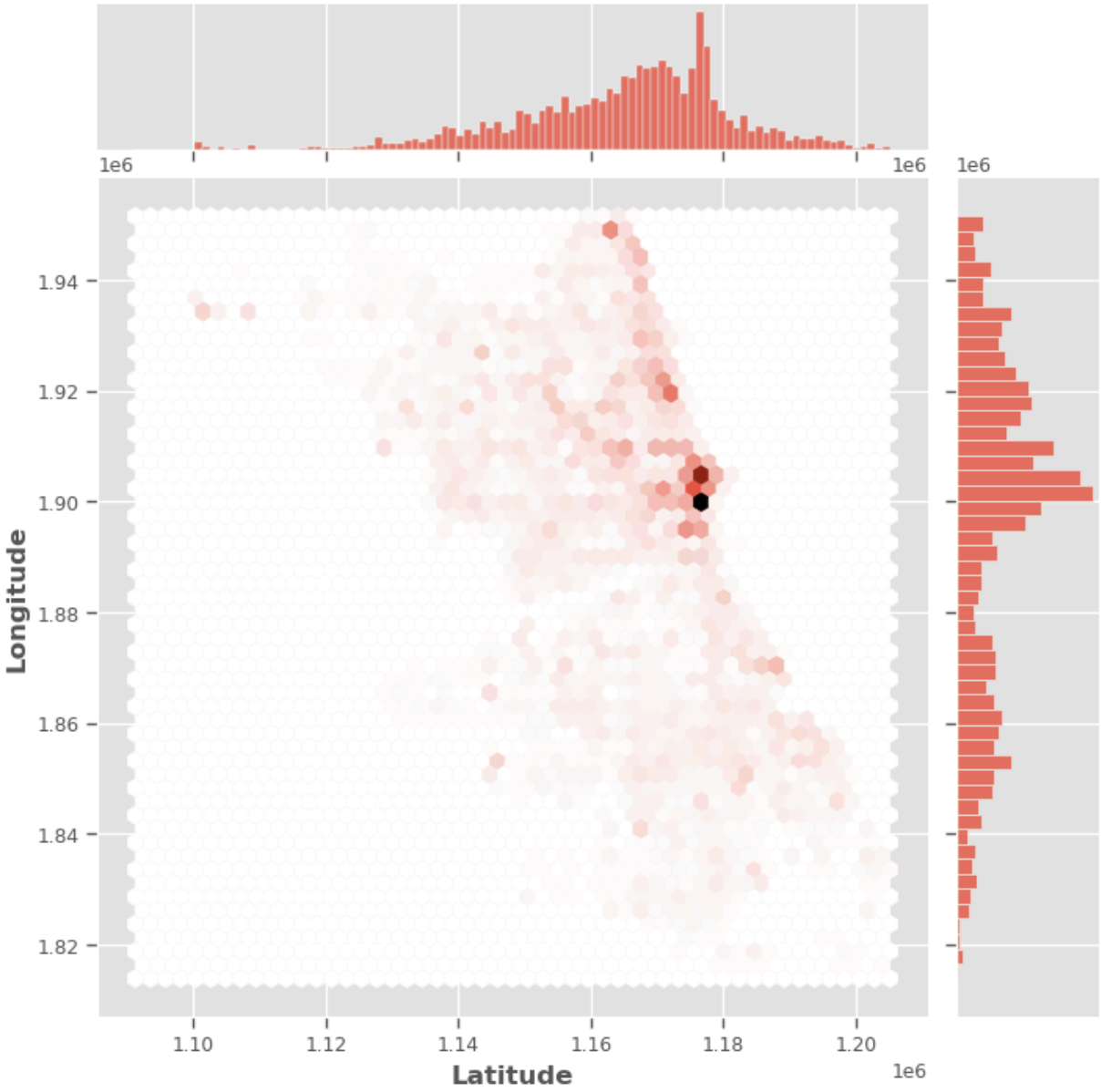
```
    preProcess = Raw_Str[1:-1].split(',')
    lat =  float(preProcess[0])
    long = float(preProcess[1])
    return (lat, long)


CR_index['LocationCoord'] = CR_index['Raw_String'].apply(Location_extractor)


CR_index  = CR_index.drop(columns=['Raw_String'], axis = 1)


#chicago_map._build_map()
#%%time

chicago_map_crime = folium.Map(location=[41.895140898, -87.624255632],
                        zoom_start=13,
                        tiles="openstreetmap")

for i in range(500):
    lat = CR_index['LocationCoord'].iloc[i][0]
    long = CR_index['LocationCoord'].iloc[i][1]
    radius = CR_index['ValueCount'].iloc[i] / 50

    if CR_index['ValueCount'].iloc[i] > 60:
        color = "#FF4500"
    else:
        color = "#008080"

    popup_text = """Latitude : {}<br>
                Longitude : {}<br>
                Criminal Incidents : {}<br>"""
    popup_text = popup_text.format(lat,
                                long,
                                CR_index['ValueCount'].iloc[i]
                                )
    folium.CircleMarker(location = [lat, long], popup= popup_text,radius = radius

#chicago_map_crime.save('crime.html')

display(chicago_map_crime)
```

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%  3280
                      Raw_String  ValueCount
0  (41.953783081, -87.915105451)          59
1  (41.868180939, -87.709271389)          50
2  (41.894945606, -87.754874977)          15
3  (41.873699424, -87.704705156)          15
4   (41.868165405, -87.62743954)          14
valuecount 3280
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
Cell In[97], line 33
     27 CR_index  = CR_index.drop(columns=['Raw_String'], axis = 1)
     30 #chicago_map._build_map()
     31 #%%time
---> 33 chicago_map_crime = folium.Map(location=[41.895140898, -87.624255632],
     34                         zoom_start=13,
     35                         tiles="openstreetmap")
     37 for i in range(500):
     38     lat = CR_index['LocationCoord'].iloc[i][0]

NameError: name 'folium' is not defined
```

In [99]:

```
# let's take in the crime as an input as always
c = input("Enter the crime you wish to see the concentration of in the city >> ")
crime_df = df[df['Primary Type']==c.upper()]

sns.jointplot(x=crime_df['X Coordinate'].values, y=crime_df['Y Coordinate'].value

plt.xlabel("Latitude", fontdict={'fontsize': 12}, weight='bold')
plt.ylabel("Longitude", fontdict={'fontsize': 12}, weight="bold")
plt.show()
```



In [101…

```
df.head()
```

Out[101…

	ID	Case Number	Date	Block	IUCR	Primary Type	Description	Location Description	Arre
1	25953	JE240540	2021-05-24 15:06:00	020XX N LARAMIE AVE	110	HOMICIDE	FIRST DEGREE MURDER	STREET	Tru
2	26038	JE279849	2021-06-26 09:24:00	062XX N MC CORMICK RD	110	HOMICIDE	FIRST DEGREE MURDER	PARKING LOT	Tru
3	13279676	JG507211	2023-11-09 07:30:00	019XX W BYRON ST	620	BURGLARY	UNLAWFUL ENTRY	APARTMENT	Fals
4	13274752	JG501049	2023-11-12 07:59:00	086XX S COTTAGE GROVE AVE	454	BATTERY	AGGRAVATED P.O. - HANDS, FISTS, FEET, NO / MIN...	SMALL RETAIL STORE	Tru
6	13203321	JG415333	2023-09-06 17:00:00	002XX N Wells st	1320	CRIMINAL DAMAGE	TO VEHICLE	PARKING LOT / GARAGE (NON RESIDENTIAL)	Fals

5 rows × 24 columns

Crime data distribution based on district

In [115…

```
from bokeh.plotting import figure, show
from bokeh.models import ColumnDataSource

# Assuming 'dis' and 'crime' are defined somewhere
tit1 = "Time Series Graph for District " + str(dis) + " and Crime " + crime + "."

# Create the figure with the correct parameters
fig5 = figure(
    height=400,  # Correct parameter name
    width=700,   # Correct parameter name
    title=tit1 + " (Hover over graph for specifics)",
    x_axis_label="Month",
    y_axis_label="Number of Crimes"
)

# Example of using ColumnDataSource, assuming 'tsg_df2' is your dataframe
tsg_cds = ColumnDataSource(tsg_df2)

# Example of setting x-axis tickers and labels (just as in your original code)
fig5.xaxis.ticker = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12]
fig5.xaxis.major_label_overrides = {1: 'Jan', 2: 'Feb', 3: 'Mar', 4: 'Apr',
                                    5: 'May', 6: 'Jun', 7: 'Jul', 8: 'Aug',
                                    9: 'Sep', 10: 'Oct', 11: 'Nov', 12: 'Dec'}

# Show the figure
show(fig5)
```

```
WARNING:bokeh.core.validation.check:W-1000 (MISSING_RENDERERS): Plot has no render
ers: figure(id='p1133', ...)
```

Time Series Graph for District 10 and Crime THEFT. (Hover over graph for specifics)

In [117…
```
# Filter the narcotics data
narc = df[df['Primary Type'] == 'NARCOTICS']

# Create a DataFrame for narcotics descriptions and their counts
narc_data = narc['Description'].value_counts().reset_index()

# Rename the columns for clarity
narc_data.columns = ['Description', 'Counts']

# Display the top few rows to verify
narc_data.head()
```

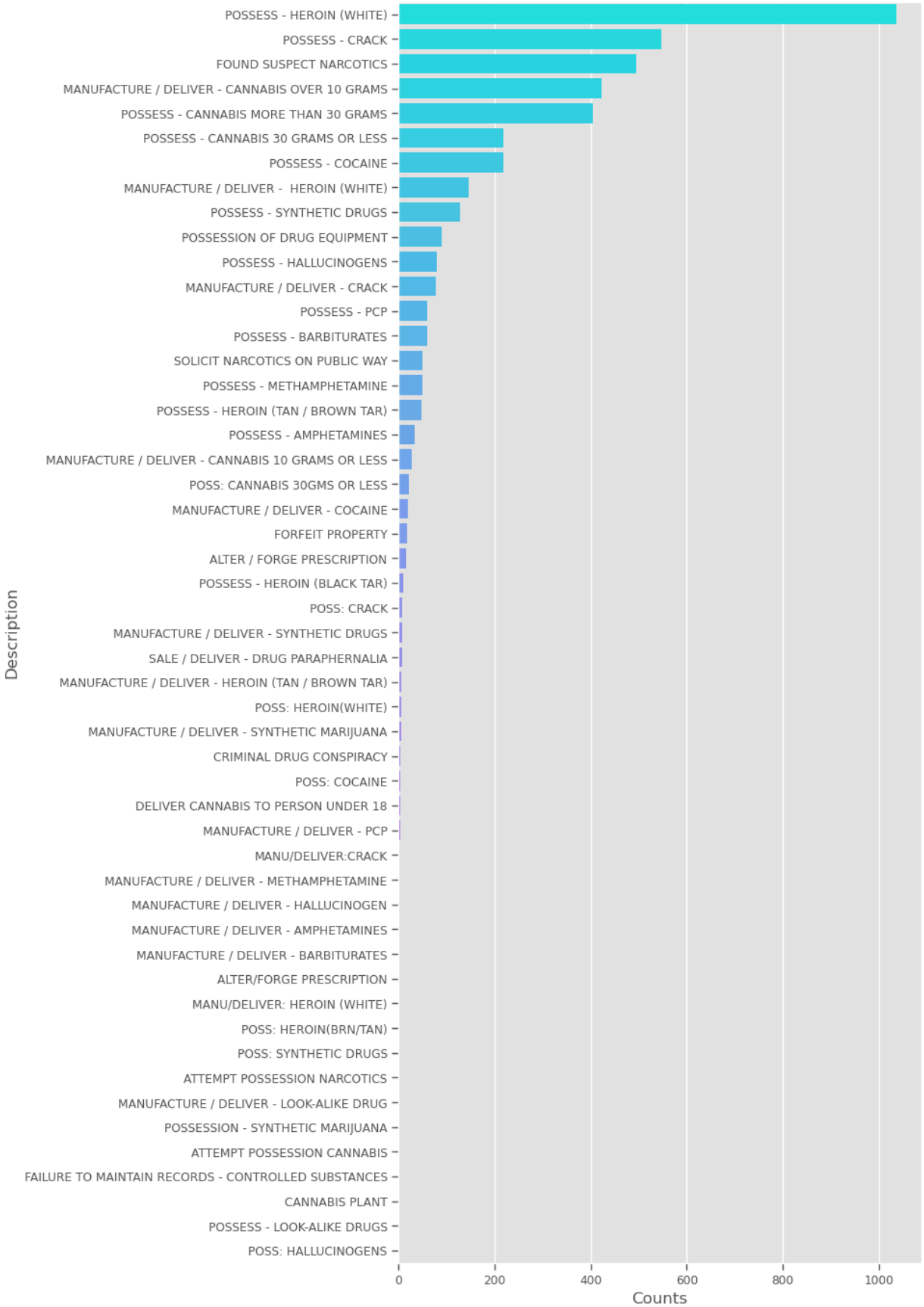
Out[117…

	Description	Counts
0	POSSESS - HEROIN (WHITE)	1036
1	POSSESS - CRACK	546
2	FOUND SUSPECT NARCOTICS	495
3	MANUFACTURE / DELIVER - CANNABIS OVER 10 GRAMS	423
4	POSSESS - CANNABIS MORE THAN 30 GRAMS	404

In [119…
```
plt.figure(figsize=(7,17))
sns.barplot(y="Description", x= "Counts", data=narc_data, palette="cool")
```

```
C:\Users\AISWARYA\AppData\Local\Temp\ipykernel_16352\8768158.py:2: FutureWarning:

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

  sns.barplot(y="Description", x= "Counts", data=narc_data, palette="cool")
```

Out[119… <Axes: xlabel='Counts', ylabel='Description'>



Sin District

In [121…
```
crimes = df['District'].unique()
crimes
```

Out[121…
```
array([25, 17, 19,  6,  1, 12,  3,  8,  7, 22,  9, 11,  4, 16, 14,  5,  2,
       10, 20, 18, 15, 24, 31], dtype=int64)
```

In [123…
```
# Filter out the Top 5 criminal districts
top_5_District = df['District'].value_counts().sort_values(ascending=False).head(
top_5_District

#plt.show()
```

Out[123…
```
District
8     14008
6     12503
12    12258
11    11767
4     11472
Name: count, dtype: int64
```

In [125…
```
from bokeh.plotting import figure
from bokeh.io import show, output_notebook
from bokeh.models import ColumnDataSource, HoverTool

# Assuming 'temp' is the DataFrame containing 'co-ordinates' and 'Number'
# Print temp DataFrame to verify its structure (for debugging)
print(temp.head())  # Uncomment this to check the structure of the 'temp' DataFra

# Create a ColumnDataSource from temp DataFrame
temp_cds = ColumnDataSource(temp)

# Create a figure with specified dimensions and labels
fig1 = figure(plot_width=700, plot_height=400, title="Most Occurring Criminal Off
              x_axis_label='Count of Crimes', y_axis_label='District number', x_r

# Format the title
fig1.title.align = "left"
fig1.title.text_color = "black"
fig1.title.text_font_size = "20px"

# Horizontal bar plot (using 'co-ordinates' for y-axis and 'Number' for crime cou
fig1.hbar(y='co-ordinates', right='Number', source=temp_cds, left=0, color='viole

# Update the y-axis labels
fig1.yaxis.major_label_overrides = {5: '11', 4: '6', 3: '8', 2: '18', 1: '1'}

# Add hover tool to display the number of crimes
tooltips = [
    ('Number of Crimes', '@Number'),
]

# Add interactivity (HoverTool)
fig1.add_tools(HoverTool(tooltips=tooltips))

# Display the plot in the notebook
output_notebook()
show(fig1)
```

```
                Crime  Number  co-ordinates
0             ASSAULT   17601             1
1  MOTOR VEHICLE THEFT   18539             2
2      CRIMINAL DAMAGE   21198             3
3             BATTERY    34162             4
4               THEFT    44378             5
```

```
---------------------------------------------------------------------------
AttributeError                            Traceback (most recent call last)
Cell In[125], line 13
     10 temp_cds = ColumnDataSource(temp)
     12 # Create a figure with specified dimensions and labels
---> 13 fig1 = figure(plot_width=700, plot_height=400, title="Most Occurring Crimi
nal Offences District-wise",
     14               x_axis_label='Count of Crimes', y_axis_label='District numbe
r', x_range=(0, 70000))
     16 # Format the title
     17 fig1.title.align = "left"

File ~\anaconda3\Lib\site-packages\bokeh\plotting\_figure.py:197, in figure.__init
__(self, *arg, **kw)
    195 for name in kw.keys():
    196     if name not in names:
--> 197         self._raise_attribute_error_with_matches(name, names | opts.proper
ties())
    199 super().__init__(*arg, **kw)
    201 self.x_range = get_range(opts.x_range)

File ~\anaconda3\Lib\site-packages\bokeh\core\has_props.py:377, in HasProps._raise
_attribute_error_with_matches(self, name, properties)
    374 if not matches:
    375     matches, text = sorted(properties), "possible"
--> 377 raise AttributeError(f"unexpected attribute {name!r} to {self.__class__.__
name__}, {text} attributes are {nice_join(matches)}")

AttributeError: unexpected attribute 'plot_width' to figure, similar attributes ar
e outer_width, width or min_width
```

The Models

Classfication of Crime Type

In [127…
```
import pandas as pd
import numpy as np
from sklearn.multiclass import OneVsRestClassifier
from sklearn.ensemble import AdaBoostClassifier
from sklearn.model_selection import cross_val_score
```

In [129…
```
tc = df['Primary Type'].value_counts().sort_values(ascending=False).head()
print(tc)
```

```
Primary Type
THEFT                   44378
BATTERY                 34162
CRIMINAL DAMAGE         21198
MOTOR VEHICLE THEFT     18539
ASSAULT                 17601
Name: count, dtype: int64
```

In [131…
```
X = df[['Arrest', 'Domestic', 'Beat', 'Community Area',
                  'Latitude', 'Longitude', 'Year', 'Hour_Day']]
y = df['Primary Type']
```

In [133…
```
X = X.fillna(0)
feature_names = list(X)
target_names = list(y)
```

In [135…
```
from sklearn.model_selection import train_test_split
seed = 42
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.33, random_
```

Classifying crime Hotspots

In [137…
```
def day(x):
    return x.strftime("%A")
df['Day'] = df['Date'].apply(day)
```

In [139…
```
# Make a new dataset for the predictions
cols = ['Date', 'Block', 'Location Description', 'Domestic', 'District','Month',
new_df = df[cols]
new_df.head()
```

Out[139…

	Date	Block	Location Description	Domestic	District	Month	Primary Type
1	2021-05-24 15:06:00	020XX N LARAMIE AVE	STREET	False	25	May	HOMICIDE
2	2021-06-26 09:24:00	062XX N MC CORMICK RD	PARKING LOT	False	17	June	HOMICIDE
3	2023-11-09 07:30:00	019XX W BYRON ST	APARTMENT	False	19	November	BURGLARY
4	2023-11-12 07:59:00	086XX S COTTAGE GROVE AVE	SMALL RETAIL STORE	False	6	November	BATTERY
6	2023-09-06 17:00:00	002XX N Wells st	PARKING LOT / GARAGE (NON RESIDENTIAL)	False	1	September	CRIMINAL DAMAGE

In [141…
```
def new_hour(x):
    return int(x.strftime("%H"))
new_df['Hour'] = new_df['Date'].apply(new_hour)
```

```
C:\Users\AISWARYA\AppData\Local\Temp\ipykernel_16352\295491757.py:3: SettingWithCo
pyWarning:
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-docs/stabl
e/user_guide/indexing.html#returning-a-view-versus-a-copy
  new_df['Hour'] = new_df['Date'].apply(new_hour)
```

In [143…
```
def new_day(x):
    return int(x.strftime("%w"))
new_df['Day'] = new_df['Date'].apply(new_day)
```

```
C:\Users\AISWARYA\AppData\Local\Temp\ipykernel_16352\911806837.py:3: SettingWithCo
pyWarning:
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-docs/stabl
e/user_guide/indexing.html#returning-a-view-versus-a-copy
  new_df['Day'] = new_df['Date'].apply(new_day)
```

In [145…
```
def new_month(x):
    return int(x.strftime("%m"))
new_df['Month_num'] = new_df['Date'].apply(new_month)
```

```
C:\Users\AISWARYA\AppData\Local\Temp\ipykernel_16352\762440598.py:3: SettingWithCo
pyWarning:
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-docs/stabl
e/user_guide/indexing.html#returning-a-view-versus-a-copy
  new_df['Month_num'] = new_df['Date'].apply(new_month)
```

In [147…
```
new_df.head()
```

Out[147…

	Date	Block	Location Description	Domestic	District	Month	Primary Type	Hour	Day	Mont
1	2021-05-24 15:06:00	020XX N LARAMIE AVE	STREET	False	25	May	HOMICIDE	15	1	
2	2021-06-26 09:24:00	062XX N MC CORMICK RD	PARKING LOT	False	17	June	HOMICIDE	9	6	
3	2023-11-09 07:30:00	019XX W BYRON ST	APARTMENT	False	19	November	BURGLARY	7	4	
4	2023-11-12 07:59:00	086XX S COTTAGE GROVE AVE	SMALL RETAIL STORE	False	6	November	BATTERY	7	0	
6	2023-09-06 17:00:00	002XX N Wells st	PARKING LOT / GARAGE (NON RESIDENTIAL)	False	1	September	CRIMINAL DAMAGE	17	3	

Key

Month : 1-12 Day : 0-6 (0 is Sunday) Hour : 0-23 (24 hour format)

In [149…
```
# replacing in Domestic (Let's use Label Encoding)
new_df['Location Description'] = new_df['Location Description'].astype('category'
```

```
new_df['Domestic'] = new_df['Domestic'].astype('category')
# new_df['Primary Type'] = new_df['Primary Type'].astype('category')
# new_df.dtypes
new_df['Location_Cat'] = new_df['Location Description'].cat.codes
new_df['Domestic_Cat'] = new_df['Domestic'].cat.codes
# new_df['Crime_Cat'] = new_df['Primary Type'].cat.codes
new_df.head()
```

```
C:\Users\AISWARYA\AppData\Local\Temp\ipykernel_16352\1497165665.py:2: SettingWithC
opyWarning:
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-docs/stabl
e/user_guide/indexing.html#returning-a-view-versus-a-copy
  new_df['Location Description'] = new_df['Location Description'].astype('categor
y')
C:\Users\AISWARYA\AppData\Local\Temp\ipykernel_16352\1497165665.py:3: SettingWithC
opyWarning:
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-docs/stabl
e/user_guide/indexing.html#returning-a-view-versus-a-copy
  new_df['Domestic'] = new_df['Domestic'].astype('category')
C:\Users\AISWARYA\AppData\Local\Temp\ipykernel_16352\1497165665.py:6: SettingWithC
opyWarning:
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-docs/stabl
e/user_guide/indexing.html#returning-a-view-versus-a-copy
  new_df['Location_Cat'] = new_df['Location Description'].cat.codes
C:\Users\AISWARYA\AppData\Local\Temp\ipykernel_16352\1497165665.py:7: SettingWithC
opyWarning:
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-docs/stabl
e/user_guide/indexing.html#returning-a-view-versus-a-copy
  new_df['Domestic_Cat'] = new_df['Domestic'].cat.codes
```

Out[149…

	Date	Block	Location Description	Domestic	District	Month	Primary Type	Hour	Day	Mont
1	2021-05-24 15:06:00	020XX N LARAMIE AVE	STREET	False	25	May	HOMICIDE	15	1	
2	2021-06-26 09:24:00	062XX N MC CORMICK RD	PARKING LOT	False	17	June	HOMICIDE	9	6	
3	2023-11-09 07:30:00	019XX W BYRON ST	APARTMENT	False	19	November	BURGLARY	7	4	
4	2023-11-12 07:59:00	086XX S COTTAGE GROVE AVE	SMALL RETAIL STORE	False	6	November	BATTERY	7	0	
6	2023-09-06 17:00:00	002XX N Wells st	PARKING LOT / GARAGE (NON RESIDENTIAL)	False	1	September	CRIMINAL DAMAGE	17	3	

In [151…

```
def day_conv(x):
    return x.strftime("%a")
new_df['Day Name'] = new_df['Date'].apply(day_conv)
```

```
C:\Users\AISWARYA\AppData\Local\Temp\ipykernel_16352\309894684.py:3: SettingWithCo
pyWarning:
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-docs/stabl
e/user_guide/indexing.html#returning-a-view-versus-a-copy
  new_df['Day Name'] = new_df['Date'].apply(day_conv)
```

Visualizing With Heatmaps

In [157…

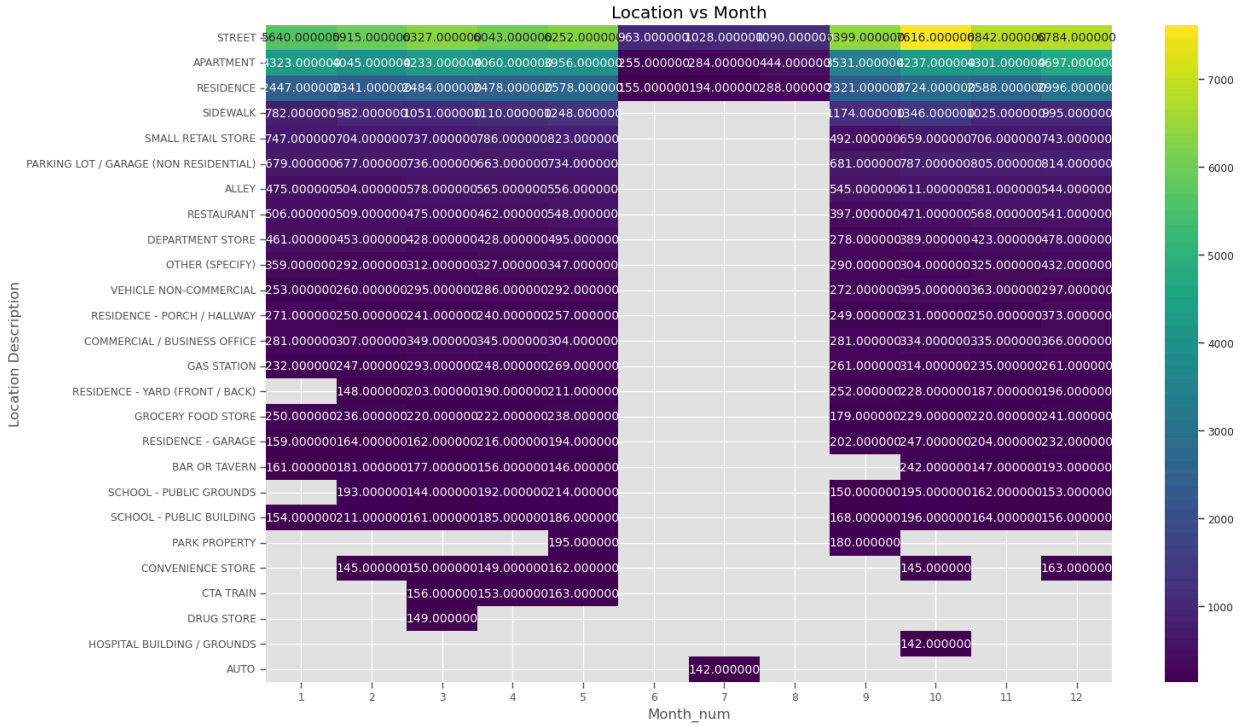
```
# Group by 'Location Description' and 'Month_num', then count the occurrences of
cri = new_df.groupby(['Location Description', 'Month_num']).size().reset_index(na
cri = cri.sort_values(by=['Primary Type', 'Month_num'], ascending=False)

# Group by 'Location Description' and 'Day', then count the occurrences of 'Prima
cri2 = new_df.groupby(['Location Description', 'Day']).size().reset_index(name='P
cri2 = cri2.sort_values(by=['Primary Type', 'Day'], ascending=False)
```

```
C:\Users\AISWARYA\AppData\Local\Temp\ipykernel_16352\1431740191.py:2: FutureWarnin
g: The default of observed=False is deprecated and will be changed to True in a fu
ture version of pandas. Pass observed=False to retain current behavior or observed
=True to adopt the future default and silence this warning.
  cri = new_df.groupby(['Location Description', 'Month_num']).size().reset_index(n
ame='Primary Type')
C:\Users\AISWARYA\AppData\Local\Temp\ipykernel_16352\1431740191.py:6: FutureWarnin
g: The default of observed=False is deprecated and will be changed to True in a fu
ture version of pandas. Pass observed=False to retain current behavior or observed
=True to adopt the future default and silence this warning.
  cri2 = new_df.groupby(['Location Description', 'Day']).size().reset_index(name
='Primary Type')
```

In [161…
```
# Group by 'Location Description' and 'Month_num', then count the occurrences of
cri = new_df.groupby(['Location Description', 'Month_num']).size().reset_index(na
cri = cri.sort_values(by=['Primary Type', 'Month_num'], ascending=False)

# Make sure to get only the first 200 rows
cri = cri.head(200)

# Pivot table for 'Location Description' and 'Month_num'
cri = cri.pivot(index='Location Description', columns='Month_num', values='Primar

# Group by 'Location Description' and 'Day', then count the occurrences of 'Prima
cri2 = new_df.groupby(['Location Description', 'Day']).size().reset_index(name='P
cri2 = cri2.sort_values(by=['Primary Type', 'Day'], ascending=False)

# Make sure to get only the first 100 rows
cri2 = cri2.head(100)

# Pivot table for 'Location Description' and 'Day'
cri2 = cri2.pivot(index='Location Description', columns='Day', values='Primary Ty
```

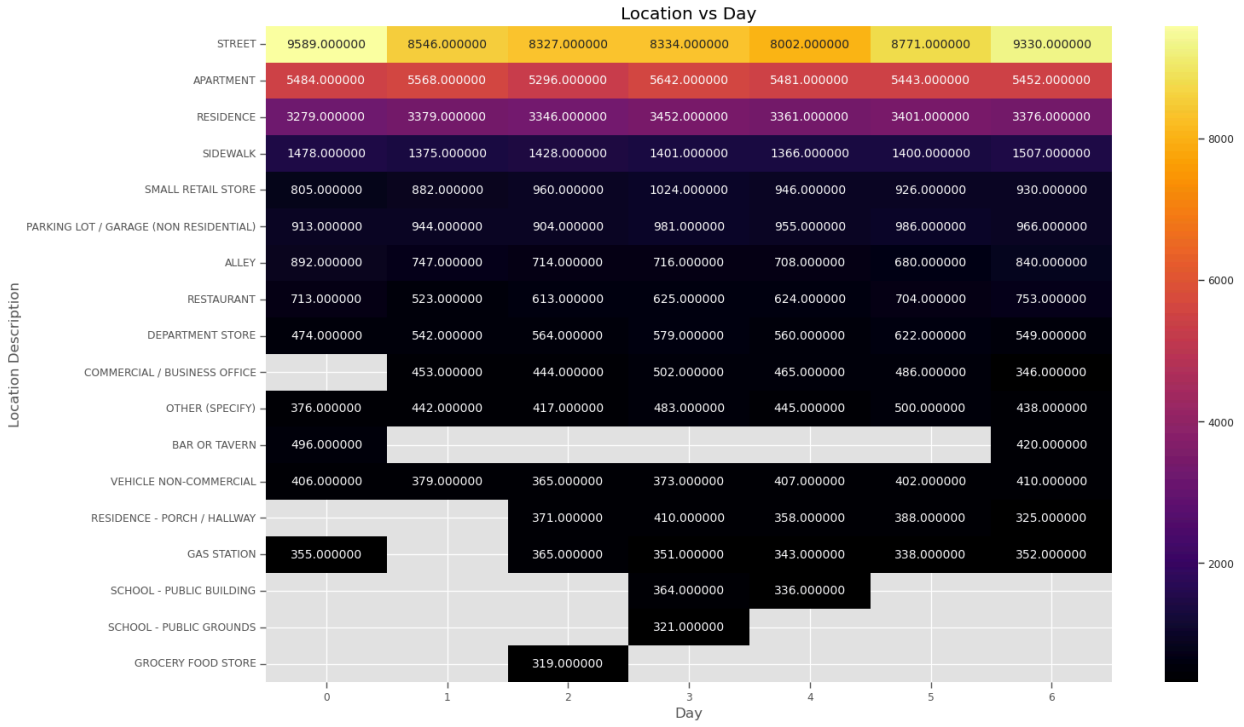
```
C:\Users\AISWARYA\AppData\Local\Temp\ipykernel_16352\2377774541.py:2: FutureWarnin
g: The default of observed=False is deprecated and will be changed to True in a fu
ture version of pandas. Pass observed=False to retain current behavior or observed
=True to adopt the future default and silence this warning.
  cri = new_df.groupby(['Location Description', 'Month_num']).size().reset_index(n
ame='Primary Type')
C:\Users\AISWARYA\AppData\Local\Temp\ipykernel_16352\2377774541.py:12: FutureWarni
ng: The default of observed=False is deprecated and will be changed to True in a f
uture version of pandas. Pass observed=False to retain current behavior or observe
d=True to adopt the future default and silence this warning.
  cri2 = new_df.groupby(['Location Description', 'Day']).size().reset_index(name
='Primary Type')
```

In [163…
```
# Plotting for Location vs Month
plt.figure(figsize = (16,10))
plt.title("Location vs Month")
with sns.axes_style("white"):
    sns.heatmap(cri, mask=cri.isnull(), cmap="viridis",annot=True,fmt="f")
```



In [165…

```
# Plotting for Location vs Day
plt.figure(figsize = (16,10))
plt.title("Location vs Day")
with sns.axes_style("white"):
    sns.heatmap(cri2, mask=cri2.isnull(), cmap="inferno",annot=True,fmt="f")
```



In [167…

```
# Gang Activity
gang_crimes = ['NARCOTICS', 'HOMICIDE', 'WEAPONS VIOLATION', 'CONCEALED CARRY LIC
gang_df = new_df[new_df['Primary Type'].isin(gang_crimes)]
gang_df.shape
```

Out[167… (28974, 13)

In [169…
```
gang_df = gang_df[gang_df['Domestic']==False] # Domestic crimes are not usually g
gang_df.shape
```

Out[169… (27770, 13)

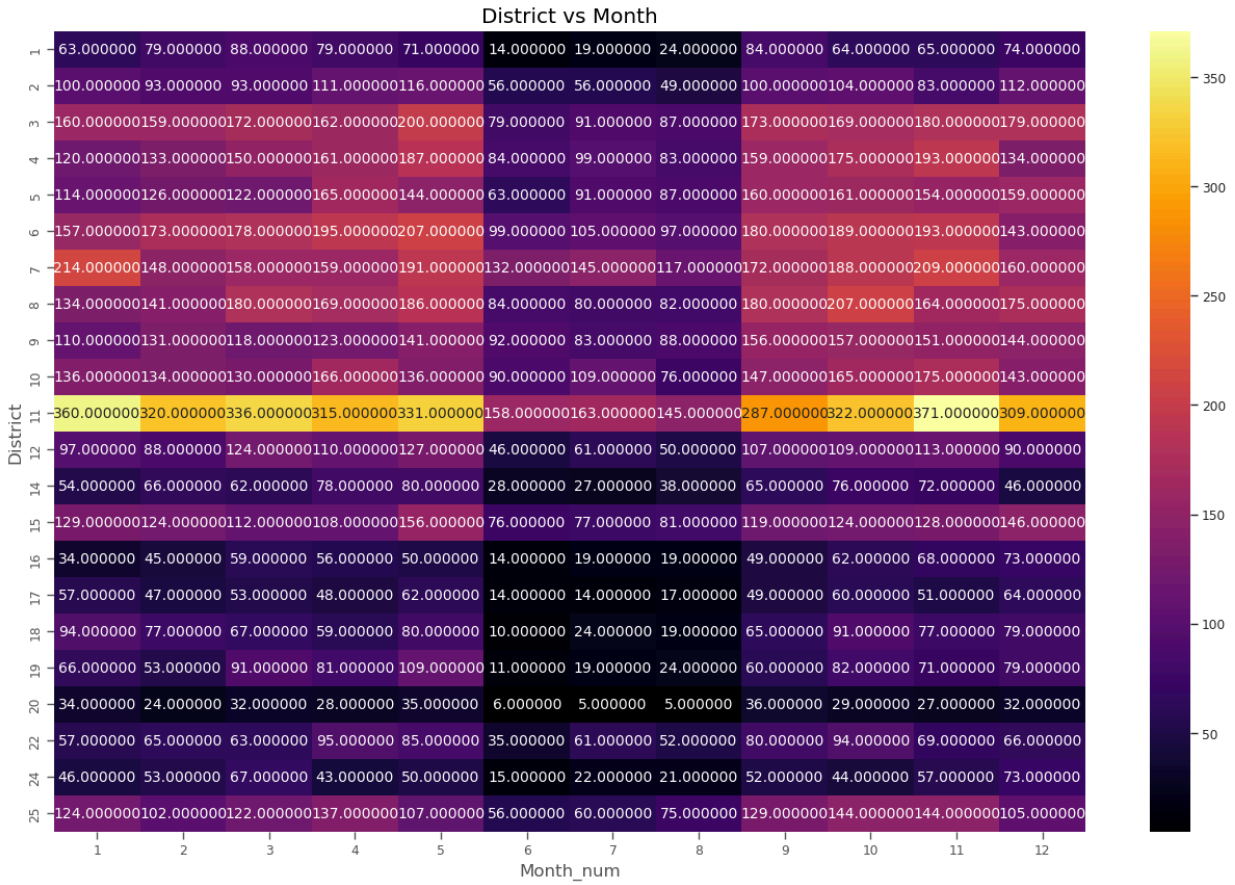
In [171…
```
del [gang_df['Date'],gang_df['Block'],gang_df['Domestic'],gang_df['Domestic_Cat']
gang_df.head()
```

Out[171…

	Location Description	District	Month	Primary Type	Hour	Day	Month_num	Location_Cat	Day Name
1	STREET	25	May	HOMICIDE	15	1	5	172	Mon
2	PARKING LOT	17	June	HOMICIDE	9	6	6	131	Sat
3	APARTMENT	19	November	BURGLARY	7	4	11	17	Thu
36	STREET	7	November	WEAPONS VIOLATION	3	4	11	172	Thu
84	CAR WASH	11	September	HOMICIDE	16	3	9	33	Wed

In [175…
```
import matplotlib.pyplot as plt
import seaborn as sns

# Group by 'District' and 'Month_num', then count the occurrences of 'Primary Typ
cri3 = gang_df.groupby(['District', 'Month_num'], as_index=False).agg({'Primary T

# Pivot the DataFrame to get 'District' as rows and 'Month_num' as columns
cri3 = cri3.pivot(index='District', columns='Month_num', values='Primary Type')

# Check if the pivot resulted in any NaN values (which could be problematic for t
print(cri3.isnull().sum())  # Check for any missing values in the pivoted data

# Plotting
plt.figure(figsize=(16, 10))
plt.title("District vs Month")

# Create a heatmap with the specified style
with sns.axes_style("white"):
    sns.heatmap(cri3, mask=cri3.isnull(), cmap="inferno", annot=True, fmt="f")

plt.show()
```

```
Month_num
1     0
2     0
3     0
4     0
5     0
6     0
7     0
8     0
9     0
10    0
11    0
12    0
dtype: int64
```



In [177…

```python
gang_df2 = new_df[new_df['Primary Type'].isin(gang_crimes)]
gang_df2 = gang_df2[gang_df2['Domestic']==False] # Domestic crimes are not usuall
del [gang_df2['Date'],gang_df2['Block'],gang_df2['Domestic'],gang_df2['Domestic_C

gang_df2['Primary Type'] = gang_df2['Primary Type'].astype('category')
gang_df2['Primary Type'] = gang_df2['Primary Type'].cat.codes

gang_df2['Location Description'] = gang_df2['Location Description'].astype('categ
gang_df2['Location Description'] = gang_df2['Location Description'].cat.codes

del [gang_df2['Day Name'], gang_df2['Month']]
print(gang_df2.head())
print("\nShape of dataset :",gang_df2.shape)
```

```
    Location Description  District  Primary Type  Hour  Day  Month_num  \
1                    172        25             2    15    1          5
2                    131        17             2     9    6          6
3                     17        19             0     7    4         11
36                   172         7             5     3    4         11
84                    33        11             2    16    3          9

    Location_Cat
1            172
2            131
3             17
36           172
84            33

Shape of dataset : (27770, 7)
```

In [179…
```
gang_df2.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Index: 27770 entries, 1 to 539516
Data columns (total 7 columns):
 #   Column                Non-Null Count  Dtype
---  ------                --------------  -----
 0   Location Description  27770 non-null  int16
 1   District              27770 non-null  int64
 2   Primary Type          27770 non-null  int8
 3   Hour                  27770 non-null  int64
 4   Day                   27770 non-null  int64
 5   Month_num             27770 non-null  int64
 6   Location_Cat          27770 non-null  int16
dtypes: int16(2), int64(4), int8(1)
memory usage: 1.2 MB
```

Using k-means clustering algorithm to cluster gang Related Crimes as per Crime Type

In [183…
```
from sklearn.cluster import KMeans
from sklearn.preprocessing import LabelEncoder
from sklearn.preprocessing import MinMaxScaler
from sklearn.metrics import accuracy_score

# Assuming 'gang_df2' is a DataFrame with your data, and 'Primary Type' is catego
X = np.array(gang_df2.drop(['Primary Type'], axis=1).astype(float))
y = np.array(gang_df2['Primary Type'])

# Scaling the features for better performance
scaler = MinMaxScaler()
X_scaled = scaler.fit_transform(X)

# Clustering the crimes into 7 clusters
kmeans = KMeans(n_clusters=7, random_state=10)
kmeans.fit(X_scaled)

# Get the predicted cluster labels
predicted_clusters = kmeans.predict(X_scaled)

# Since the cluster labels are arbitrary, you would need to map them to the true
# This step requires additional logic to find a mapping between predicted cluster
# However, as a simple approximation (not generally recommended), you can compare
# Accuracy based on cluster assignments, even though it's not ideal
```

```
accuracy = accuracy_score(y, predicted_clusters)

print("Accuracy: ", accuracy)
```

```
Accuracy:  0.14094346416996759
```

Using Supervised Machine Learning to Predict Crime Hostspots

In [189…
```
# Creating our explicit dataset
cri4 = new_df.groupby(['Month_num','Day','District','Hour'], as_index=False).agg(
cri4 = cri4.sort_values(by=['District'], ascending=False)
cri4.head()
```

Out[189…

	Month_num	Day	District	Hour	Primary Type
1545	1	2	31	9	1
32384	11	3	31	9	1
2062	1	3	31	20	2
12868	4	3	31	10	1
32905	11	4	31	6	1

In [191…
```
cri4 = cri4[['Month_num','Day','Hour','Primary Type','District']]
cri4.head()
cri4.shape
```

Out[191… (37561, 5)

In [193…
```
print(cri4['Primary Type'].max(),cri4['Primary Type'].min())
```

```
42 1
```

In [195…
```
print("Average no. of crime per month per day per district per hour :",cri4['Prim
```

```
Average no. of crime per month per day per district per hour : 4.853275354187123 .
```

In [197…
```
# Feature Engineer and create a new feature
def crime_rate_assign(x):
    if(x<=7):
        return 0
    elif(x>7 and x<=15):
        return 1
    else:
        return 2
cri4['Alarm'] = cri4['Primary Type'].apply(crime_rate_assign)
cri4 = cri4[['Month_num','Day','Hour','District','Primary Type','Alarm']]
cri4.head()
```

Out[197…

	Month_num	Day	Hour	District	Primary Type	Alarm
1545	1	2	9	31	1	0
32384	11	3	9	31	1	0
2062	1	3	20	31	2	0
12868	4	3	10	31	1	0
32905	11	4	6	31	1	0

In [199…

```
# Using Decision Trees for classification
from sklearn.tree import DecisionTreeClassifier
from sklearn.model_selection import train_test_split
from sklearn import metrics
from sklearn.metrics import confusion_matrix, classification_report
from sklearn.utils.multiclass import unique_labels

X = cri4[['Month_num', 'Day', 'Hour', 'District']] # independent
y = cri4['Alarm'] # dependent

# Let's split the dataset
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.25, random_

# print(X_train)
# print('hi')
# print(X_test)
# Creating tree
d_tree = DecisionTreeClassifier(random_state=10)
# Fitting tree
d_tree = d_tree.fit(X_train, y_train)
# Predicting !
y_pred = d_tree.predict(X_test)

# Model Evaluation
# print(y_test)
# print(y_pred)
print("Accuracy:",(metrics.accuracy_score(y_test, y_pred)*100),"\n")

# Confusion Matrix for evaluating the model
cm = pd.crosstab(y_test, y_pred, rownames=['Actual Alarm'], colnames=['Predicted 
print("\n----------Confusion Matrix-----------------------------------")
print(cm)

# Classification Report
print("\n----------Classification Report------------------------------------")
print(classification_report(y_test,y_pred))

# Unweighted Average Recall
print("\nUAR ->",((cm[0][0])/(cm[0][0]+cm[1][0]+cm[2][0])+(cm[1][1])/(cm[0][1]+cm
```

```
Accuracy: 71.99446278351613


----------Confusion Matrix------------------------------------
Predicted Alarm     0     1    2
Actual Alarm
0                5607  1223   33
1                1139  1126  116
2                  28    91   28

----------Classification Report------------------------------------
              precision    recall  f1-score   support

           0       0.83      0.82      0.82      6863
           1       0.46      0.47      0.47      2381
           2       0.16      0.19      0.17       147

    accuracy                           0.72      9391
   macro avg       0.48      0.49      0.49      9391
weighted avg       0.72      0.72      0.72      9391


UAR -> 0.49345879564510803
```

In [203…
```
from sklearn import metrics
```

In [205…
```
from sklearn.metrics import classification_report
```

In [207…
```
import joblib
```

In [209…
```
# Import necessary libraries
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import confusion_matrix, classification_report, accuracy_sco
import joblib
import pandas as pd

# Assuming 'cri4' is your DataFrame with data
X = cri4.iloc[:, 0:4].values
y = cri4.iloc[:, 5].values

# Split the data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.25, random_

# Standardize the data
scaler = StandardScaler()
X_train = scaler.fit_transform(X_train)
X_test = scaler.transform(X_test)

# Initialize the RandomForestClassifier
classifier = RandomForestClassifier(n_estimators=10, criterion='entropy', random_
classifier.fit(X_train, y_train)

# Predict on the test set
y_pred = classifier.predict(X_test)

# Print accuracy
```

```
print("Accuracy:", (accuracy_score(y_test, y_pred) * 100), "\n")

# Confusion Matrix
cm = confusion_matrix(y_test, y_pred)
print("\n----------Confusion Matrix-----------------------------------")
print(cm)

# Classification Report
print("\n----------Classification Report------------------------------------")
print(classification_report(y_test, y_pred))

# Calculate Unweighted Average Recall (UAR)
# UAR = (Recall for Class 0 + Recall for Class 1 + Recall for Class 2) / 3
recall_0 = cm[0, 0] / (cm[0, 0] + cm[0, 1] + cm[0, 2])  # Recall for Class 0
recall_1 = cm[1, 1] / (cm[1, 0] + cm[1, 1] + cm[1, 2])  # Recall for Class 1
recall_2 = cm[2, 2] / (cm[2, 0] + cm[2, 1] + cm[2, 2])  # Recall for Class 2

UAR = (recall_0 + recall_1 + recall_2) / 3
print("\nUAR ->", UAR)
```

```
Accuracy: 76.77563624747098


----------Confusion Matrix-----------------------------------
[[6126  745    7]
 [1288 1075   31]
 [  22   88    9]]

----------Classification Report------------------------------------
              precision    recall  f1-score   support

           0       0.82      0.89      0.86      6878
           1       0.56      0.45      0.50      2394
           2       0.19      0.08      0.11       119

    accuracy                           0.77      9391
   macro avg       0.53      0.47      0.49      9391
weighted avg       0.75      0.77      0.76      9391


UAR -> 0.4717784693923448
```

In [ ]: