Group - Number: 3

Team Members:

- Yen Nga Le - 0824817
- Tehsin Shaikh - 0831234
- Srilakshmi Gummadidala - 0803509
- Vinod Soloman Santhakumar - 0821990

1. Load libraries

In [1]:
```
# Import all the libraries
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
```

2. Load Data

In [2]:
```
df = pd.read_csv('Major_Crime_Indicators_Open_Data.csv')
df.head()
```

Out[2]:

	X	Y	OBJECTID	EVENT_UNIQUE_ID	REPORT_DATE	OCC_DATE	REPORT_YI
0	-8.837009e+06	5.414638e+06	1	GO-20141265238	2014/01/01 05:00:00+00	2014/01/01 05:00:00+00	2
1	-8.832733e+06	5.419701e+06	2	GO-20141259834	2014/01/01 05:00:00+00	2014/01/01 05:00:00+00	2
2	-8.836444e+06	5.410819e+06	3	GO-20141262027	2014/01/01 05:00:00+00	2014/01/01 05:00:00+00	2
3	-8.836897e+06	5.412101e+06	4	GO-20141259951	2014/01/01 05:00:00+00	2014/01/01 05:00:00+00	2
4	-8.851435e+06	5.422186e+06	5	GO-20141261561	2014/01/01 05:00:00+00	2014/01/01 05:00:00+00	2

5 rows × 31 columns

3. Exploratory Data Analysis

3.1 Understand Column Labels

- **EVENT_UNIQUE_ID:** Offence Number
- **REPORT_DATE:** Date Offence was Reported (time is displayed in UTC format when downloaded as a CSV)
- **OCC_DATE:** Date Offence Occurred (time is displayed in UTC format when downloaded as a CSV)
- **REPORT_YEAR:** Year Offence was Reported
- **REPORT_MONTH:** Month Offence was Reported
- **REPORT_DAY:** Day of the Month Offence was Reported
- **REPORT_DOY:** Day of the Year Offence was Reported
- **REPORT_DOW:** Day of the Week Offence was Reported
- **REPORT_HOUR:** Hour Offence was Reported
- **OCC_YEAR:** Year Offence Occurred
- **OCC_MONTH:** Month Offence Occurred
- **OCC_DAY:** Day of the Month Offence Occurred
- **OCC_DOY:** Day of the Year Offence Occurred
- **OCC_DOW:** Day of the Week Offence Occurred
- **OCC_HOUR:** Hour Offence Occurred
- **DIVISION:** Police Division where Offence Occurred
- **LOCATION_TYPE:** Location Type of Offence
- **PREMISES_TYPE:** Premises Type of Offence
- **UCR_CODE:** UCR Code for Offence
- **UCR_EXT:** UCR Extension for Offence
- **OFFENCE:** Title of Offence
- **MCI_CATEGORY:** MCI Category of Occurrence
- **HOOD_158:** Identifier of Neighbourhood using City of Toronto's new 158 neighbourhood structure
- **NEIGHBOURHOOD_158:** Name of Neighbourhood using City of Toronto's new 158 neighbourhood structure
- **HOOD_140:** Identifier of Neighbourhood using City of Toronto's old 140 neighbourhood structure
- **NEIGHBOURHOOD_140:** Name of Neighbourhood using City of Toronto's old 140 neighbourhood structure
- **LONG_WGS84:** Longitude Coordinates (Offset to nearest intersection)
- **LAT_WGS84:** Latitude Coordinates (Offset to nearest intersection)
- **X** and **Y** are the same actual longitude and Latitude values

In [3]:
```
#Display the summary of the DataFrame df, including the number of non-null values, the
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 360187 entries, 0 to 360186
Data columns (total 31 columns):
 #   Column              Non-Null Count   Dtype  
---  ------              --------------   -----  
 0   X                   360187 non-null  float64
 1   Y                   360187 non-null  float64
 2   OBJECTID            360187 non-null  int64  
 3   EVENT_UNIQUE_ID     360187 non-null  object 
 4   REPORT_DATE         360187 non-null  object 
 5   OCC_DATE            360187 non-null  object 
 6   REPORT_YEAR         360187 non-null  int64  
 7   REPORT_MONTH        360187 non-null  object 
 8   REPORT_DAY          360187 non-null  int64  
 9   REPORT_DOY          360187 non-null  int64  
 10  REPORT_DOW          360187 non-null  object 
 11  REPORT_HOUR         360187 non-null  int64  
 12  OCC_YEAR            360076 non-null  float64
 13  OCC_MONTH           360076 non-null  object 
 14  OCC_DAY             360076 non-null  float64
 15  OCC_DOY             360076 non-null  float64
 16  OCC_DOW             360076 non-null  object 
 17  OCC_HOUR            360187 non-null  int64  
 18  DIVISION            360187 non-null  object 
 19  LOCATION_TYPE       360187 non-null  object 
 20  PREMISES_TYPE       360187 non-null  object 
 21  UCR_CODE            360187 non-null  int64  
 22  UCR_EXT             360187 non-null  int64  
 23  OFFENCE             360187 non-null  object 
 24  MCI_CATEGORY        360187 non-null  object 
 25  HOOD_158            360187 non-null  object 
 26  NEIGHBOURHOOD_158   360187 non-null  object 
 27  HOOD_140            360187 non-null  object 
 28  NEIGHBOURHOOD_140   360187 non-null  object 
 29  LONG_WGS84          360187 non-null  float64
 30  LAT_WGS84           360187 non-null  float64
dtypes: float64(7), int64(8), object(16)
memory usage: 85.2+ MB
```

In [4]:
```
# Checking unique values
df.nunique()
```

Out[4]:
```
X                    23535
Y                    23534
OBJECTID            360187
EVENT_UNIQUE_ID     313796
REPORT_DATE           3560
OCC_DATE              4035
REPORT_YEAR             10
REPORT_MONTH            12
REPORT_DAY              31
REPORT_DOY             366
REPORT_DOW               7
REPORT_HOUR             24
OCC_YEAR                24
OCC_MONTH               12
OCC_DAY                 31
OCC_DOY                366
OCC_DOW                  7
OCC_HOUR                24
DIVISION                18
LOCATION_TYPE           54
PREMISES_TYPE            7
UCR_CODE                22
UCR_EXT                 16
OFFENCE                 51
MCI_CATEGORY             5
HOOD_158               159
NEIGHBOURHOOD_158      159
HOOD_140               141
NEIGHBOURHOOD_140      141
LONG_WGS84           23537
LAT_WGS84            23537
dtype: int64
```

3.2 Handle Missing Values Duplicate Values

In [5]:
```
#Display the number of missing values for each column of the DataFrame df
df.isnull().sum()
```

Out[5]:

```
X                     0
Y                     0
OBJECTID              0
EVENT_UNIQUE_ID       0
REPORT_DATE           0
OCC_DATE              0
REPORT_YEAR           0
REPORT_MONTH          0
REPORT_DAY            0
REPORT_DOY            0
REPORT_DOW            0
REPORT_HOUR           0
OCC_YEAR            111
OCC_MONTH           111
OCC_DAY             111
OCC_DOY             111
OCC_DOW             111
OCC_HOUR              0
DIVISION              0
LOCATION_TYPE         0
PREMISES_TYPE         0
UCR_CODE              0
UCR_EXT               0
OFFENCE               0
MCI_CATEGORY          0
HOOD_158              0
NEIGHBOURHOOD_158     0
HOOD_140              0
NEIGHBOURHOOD_140     0
LONG_WGS84            0
LAT_WGS84             0
dtype: int64
```

In [6]:

```
# dropping null values to ensure complete and reliable data for analysis
df = df.dropna()
df.isnull().sum()
```

Out[6]:
```
X                    0
Y                    0
OBJECTID             0
EVENT_UNIQUE_ID      0
REPORT_DATE          0
OCC_DATE             0
REPORT_YEAR          0
REPORT_MONTH         0
REPORT_DAY           0
REPORT_DOY           0
REPORT_DOW           0
REPORT_HOUR          0
OCC_YEAR             0
OCC_MONTH            0
OCC_DAY              0
OCC_DOY              0
OCC_DOW              0
OCC_HOUR             0
DIVISION             0
LOCATION_TYPE        0
PREMISES_TYPE        0
UCR_CODE             0
UCR_EXT              0
OFFENCE              0
MCI_CATEGORY         0
HOOD_158             0
NEIGHBOURHOOD_158    0
HOOD_140             0
NEIGHBOURHOOD_140    0
LONG_WGS84           0
LAT_WGS84            0
dtype: int64
```

In [7]:
```
# Dropping X and Y columns which are the same actual Longitude and Latitude values
df = df.drop(columns = ['X', 'Y'], axis = 1)
```

In [8]:
```
#Check for duplicates
duplicates = df.duplicated()
count_duplicates = duplicates.sum()

print(duplicates)
print("Number of duplicate rows:", count_duplicates)
```

```
0         False
1         False
2         False
3         False
4         False
          ...
360182    False
360183    False
360184    False
360185    False
360186    False
Length: 360076, dtype: bool
Number of duplicate rows: 0
```

3.3 Statistical Analysis

In [9]:
```
print ("Rows     : " ,df.shape[0])
print ("Columns  : " ,df.shape[1])
print ("\nFeatures : \n" ,df.columns.tolist())
print ("\nUnique values :  \n",df.nunique())
```

```
Rows     :  360076
Columns  :  29

Features :
 ['OBJECTID', 'EVENT_UNIQUE_ID', 'REPORT_DATE', 'OCC_DATE', 'REPORT_YEAR', 'REPORT_MO
NTH', 'REPORT_DAY', 'REPORT_DOY', 'REPORT_DOW', 'REPORT_HOUR', 'OCC_YEAR', 'OCC_MONT
H', 'OCC_DAY', 'OCC_DOY', 'OCC_DOW', 'OCC_HOUR', 'DIVISION', 'LOCATION_TYPE', 'PREMIS
ES_TYPE', 'UCR_CODE', 'UCR_EXT', 'OFFENCE', 'MCI_CATEGORY', 'HOOD_158', 'NEIGHBOURHOO
D_158', 'HOOD_140', 'NEIGHBOURHOOD_140', 'LONG_WGS84', 'LAT_WGS84']

Unique values :
 OBJECTID             360076
EVENT_UNIQUE_ID      313708
REPORT_DATE            3560
OCC_DATE               3975
REPORT_YEAR              10
REPORT_MONTH             12
REPORT_DAY               31
REPORT_DOY              366
REPORT_DOW                7
REPORT_HOUR              24
OCC_YEAR                 24
OCC_MONTH                12
OCC_DAY                  31
OCC_DOY                 366
OCC_DOW                   7
OCC_HOUR                 24
DIVISION                 18
LOCATION_TYPE            54
PREMISES_TYPE             7
UCR_CODE                 22
UCR_EXT                  16
OFFENCE                  51
MCI_CATEGORY              5
HOOD_158                159
NEIGHBOURHOOD_158       159
HOOD_140                141
NEIGHBOURHOOD_140       141
LONG_WGS84            23536
LAT_WGS84             23536
dtype: int64
```

In [10]:
```
# save the cleaned dataset for further analysis
df.to_csv('MCI_clean.csv', index = False)
```

In [11]:
```
# Top five rows as columns for better column readability
df.head().T
```

Out[11]:

	0	1	2	3	4
OBJECTID	1	2	3	4	5
EVENT_UNIQUE_ID	GO-20141265238	GO-20141259834	GO-20141262027	GO-20141259951	GO-20141261561
REPORT_DATE	2014/01/01 05:00:00+00	2014/01/01 05:00:00+00	2014/01/01 05:00:00+00	2014/01/01 05:00:00+00	2014/01/01 05:00:00+00
OCC_DATE	2014/01/01 05:00:00+00	2014/01/01 05:00:00+00	2014/01/01 05:00:00+00	2014/01/01 05:00:00+00	2014/01/01 05:00:00+00
REPORT_YEAR	2014	2014	2014	2014	2014
REPORT_MONTH	January	January	January	January	January
REPORT_DAY	1	1	1	1	1
REPORT_DOY	1	1	1	1	1
REPORT_DOW	Wednesday	Wednesday	Wednesday	Wednesday	Wednesday
REPORT_HOUR	0	0	10	3	8
OCC_YEAR	2014.0	2014.0	2014.0	2014.0	2014.0
OCC_MONTH	January	January	January	January	January
OCC_DAY	1.0	1.0	1.0	1.0	1.0
OCC_DOY	1.0	1.0	1.0	1.0	1.0
OCC_DOW	Wednesday	Wednesday	Wednesday	Wednesday	Wednesday
OCC_HOUR	0	0	10	0	8
DIVISION	D53	D53	D52	D52	D31
LOCATION_TYPE	Ttc Subway Station	Bar / Restaurant	Streets, Roads, Highways (Bicycle Path, Privat...	Streets, Roads, Highways (Bicycle Path, Privat...	Commercial Dwelling Unit (Hotel, Motel, B & B,...
PREMISES_TYPE	Transit	Commercial	Outside	Outside	Commercial
UCR_CODE	1430	1420	1430	1460	1420
UCR_EXT	100	100	100	100	100
OFFENCE	Assault	Assault With Weapon	Assault	Assault Peace Officer	Assault With Weapon
MCI_CATEGORY	Assault	Assault	Assault	Assault	Assault
HOOD_158	98	55	166	170	154
NEIGHBOURHOOD_158	Rosedale-Moore Park	Thorncliffe Park	St Lawrence-East Bayfront-The Islands	Yonge-Bay Corridor	Oakdale-Beverley Heights
HOOD_140	98	55	77	76	26
NEIGHBOURHOOD_140	Rosedale-Moore Park (98)	Thorncliffe Park (55)	Waterfront Communities-The Island (77)	Bay Street Corridor (76)	Downsview-Roding-CFB (26)

	0	1	2	3	4
LONG_WGS84	-79.384206	-79.345795	-79.379131	-79.3832	-79.513797
LAT_WGS84	43.670798	43.703684	43.645981	43.654313	43.719824

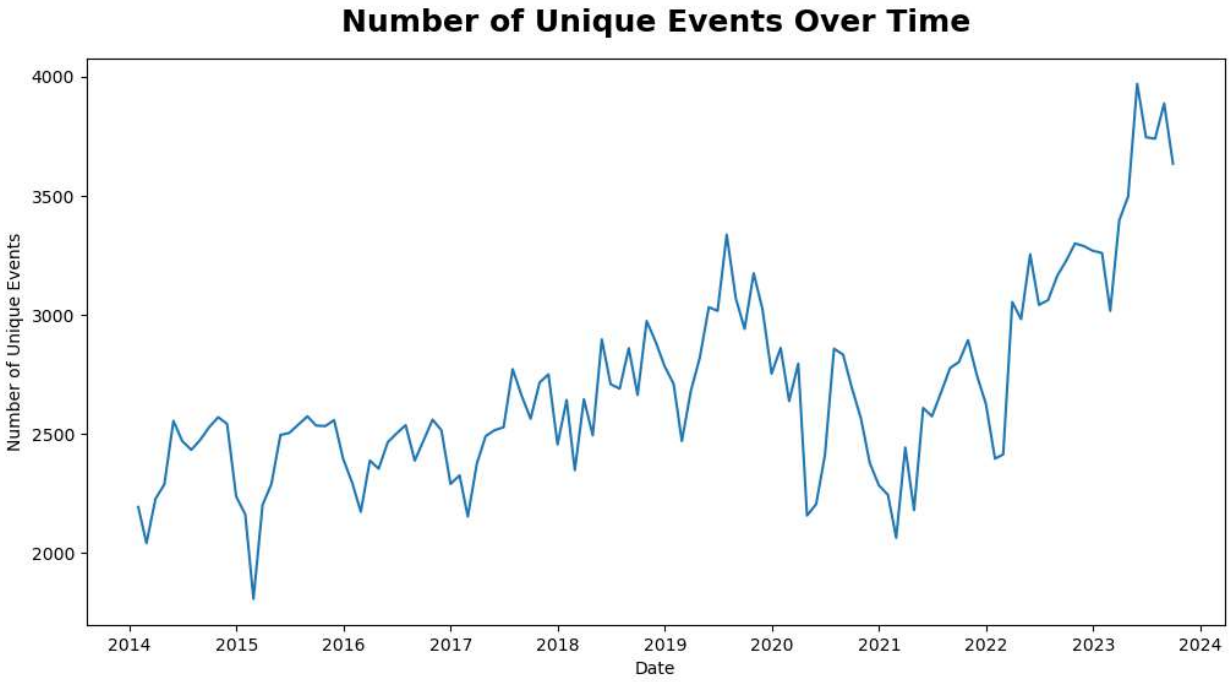
In [12]:
```
df.describe()
```

Out[12]:

	OBJECTID	REPORT_YEAR	REPORT_DAY	REPORT_DOY	REPORT_HOUR	OCC_YEAR	3
count	360076.000000	360076.000000	360076.000000	360076.000000	360076.000000	360076.000000	
mean	180104.761109	2018.653465	15.745373	182.532788	12.730424	2018.595308	
std	103973.537642	2.843301	8.766424	102.371963	6.476869	2.880038	
min	1.000000	2014.000000	1.000000	1.000000	0.000000	2000.000000	
25%	90063.750000	2016.000000	8.000000	96.000000	8.000000	2016.000000	
50%	180115.500000	2019.000000	16.000000	183.500000	13.000000	2019.000000	
75%	270149.250000	2021.000000	23.000000	268.000000	18.000000	2021.000000	
max	360187.000000	2023.000000	31.000000	366.000000	23.000000	2023.000000	

3.4 Column Level Analysis

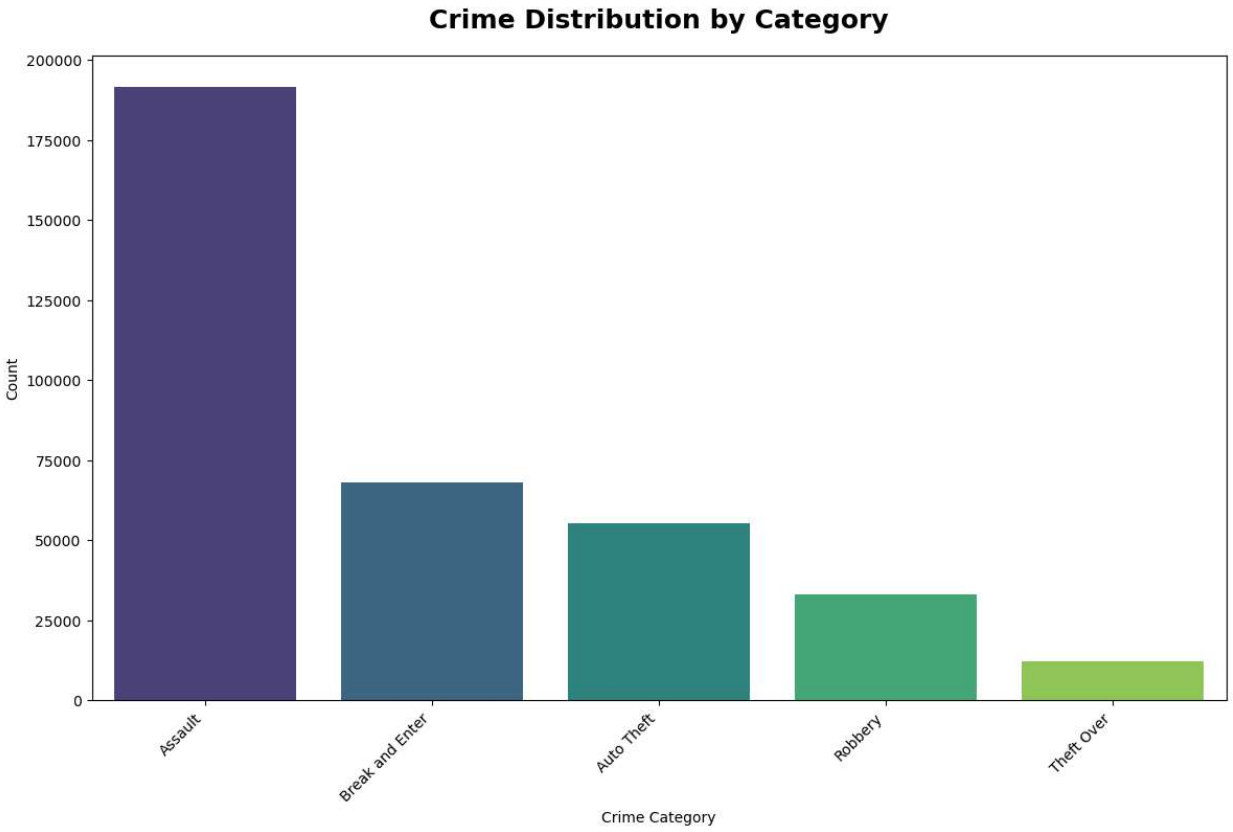
In [13]:
```
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns

# Convert date columns to datetime objects for better analysis
df['REPORT_DATE'] = pd.to_datetime(df['REPORT_DATE'])
df['OCC_DATE'] = pd.to_datetime(df['OCC_DATE'])


# Count the number of unique events per month
monthly_counts = df.resample('M', on='REPORT_DATE')['EVENT_UNIQUE_ID'].nunique()

# Plot the number of unique events over time
plt.figure(figsize=(12, 6))
sns.lineplot(x=monthly_counts.index, y=monthly_counts.values)
plt.title('Number of Unique Events Over Time',fontsize=18, fontweight='bold', y=1.03)
plt.xlabel('Date')
plt.ylabel('Number of Unique Events')
plt.show()
```



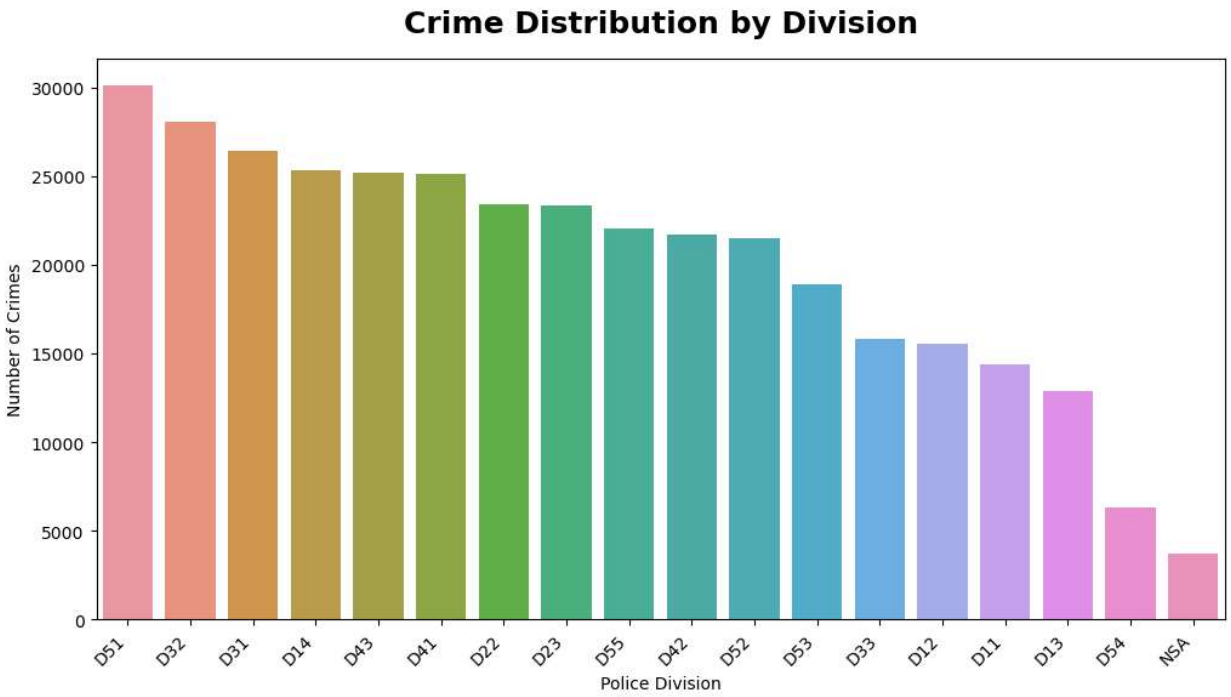
In [14]:
```
# count of unique values in the 'MCI_CATEGORY' column
df.MCI_CATEGORY.value_counts()
```

Out[14]:
```
Assault            191724
Break and Enter     67884
Auto Theft          55334
Robbery             33061
Theft Over          12073
Name: MCI_CATEGORY, dtype: int64
```

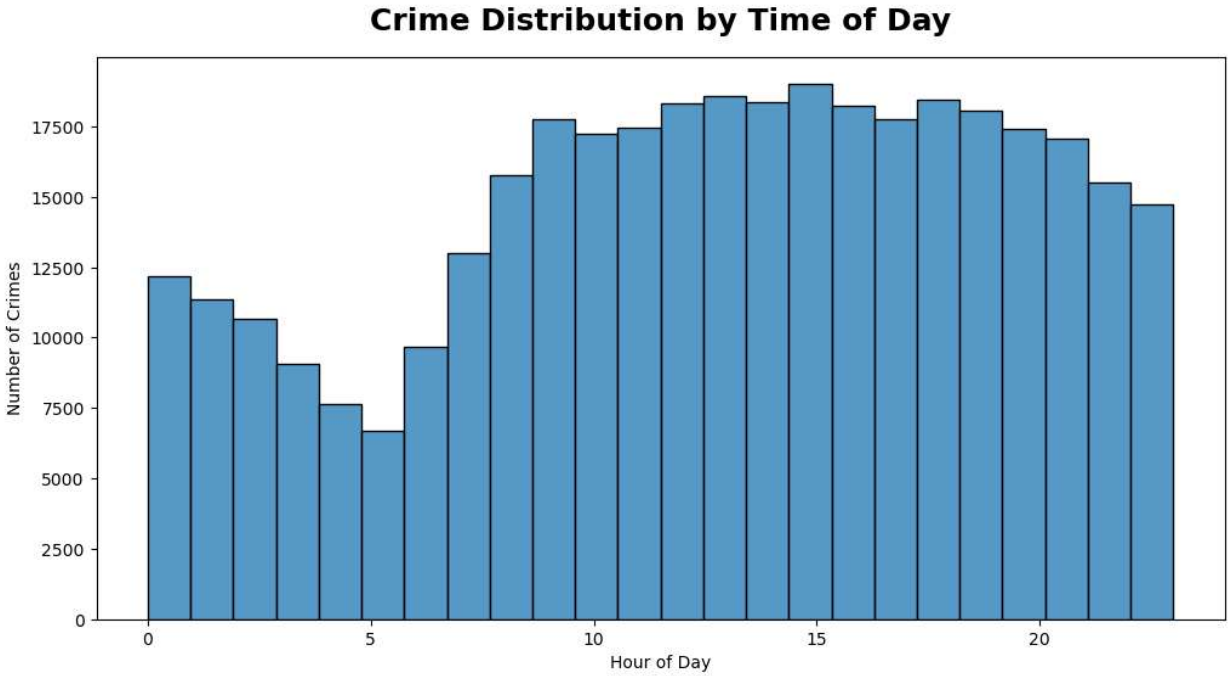
In [15]:
```
import seaborn as sns
import matplotlib.pyplot as plt

# Plot the distribution of crimes by category
plt.figure(figsize=(14, 8))
sns.countplot(x='MCI_CATEGORY', data=df, order=df['MCI_CATEGORY'].value_counts().index
plt.title('Crime Distribution by Category',fontsize=18, fontweight='bold', y=1.03)
plt.xlabel('Crime Category')
plt.ylabel('Count')
plt.xticks(rotation=45, ha='right')  # Rotate x-axis labels for better visibility
plt.show()
```



In [16]:
```
import seaborn as sns
import matplotlib.pyplot as plt

# Bar plot for crime distribution by division
plt.figure(figsize=(12, 6))
sns.countplot(x='DIVISION', data=df, order=df['DIVISION'].value_counts().index)
plt.title('Crime Distribution by Division',fontsize=18, fontweight='bold', y=1.03)
plt.xlabel('Police Division')
plt.ylabel('Number of Crimes')
plt.xticks(rotation=45, ha='right')  # Rotate x-axis labels for better readability
plt.show()
```

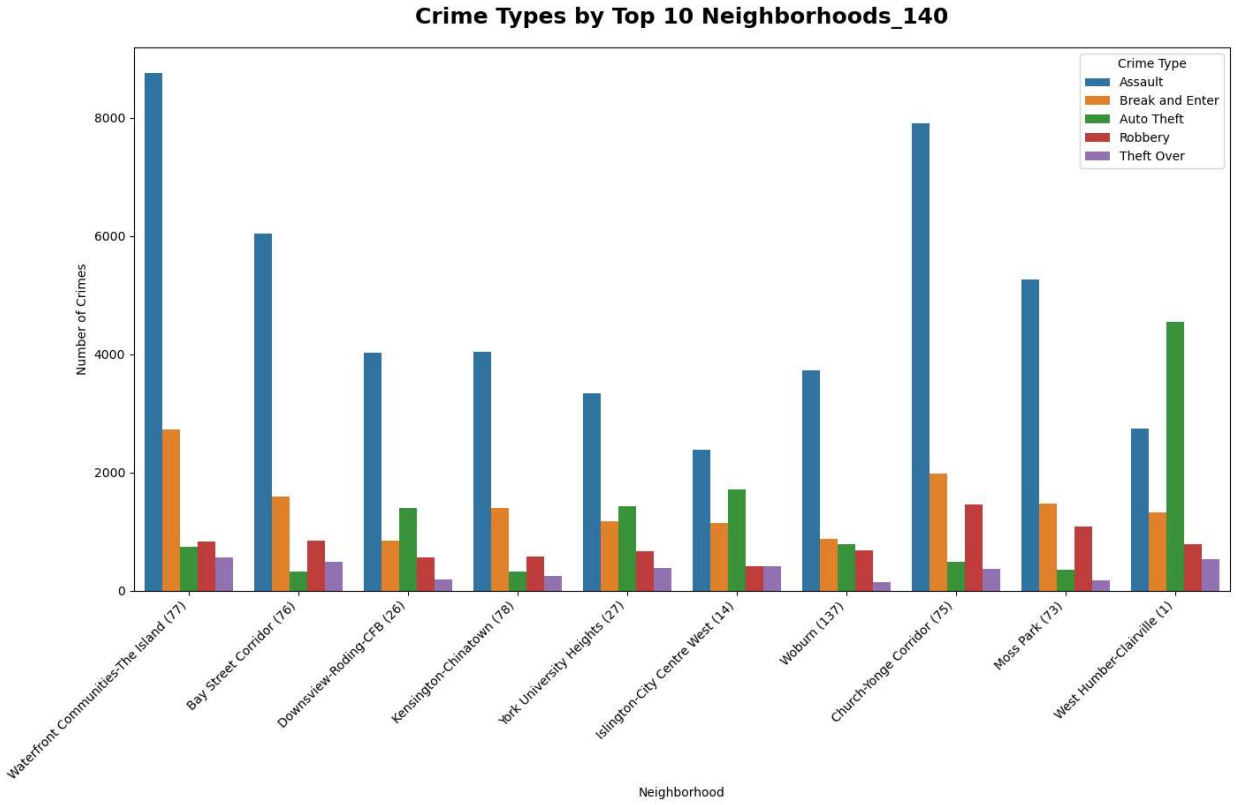


In [17]:
```
import seaborn as sns
import matplotlib.pyplot as plt

# Histogram for crime distribution by hour
plt.figure(figsize=(12, 6))
sns.histplot(df['REPORT_HOUR'], bins=24, kde=False)
plt.title('Crime Distribution by Time of Day',fontsize=18, fontweight='bold', y=1.03)
plt.xlabel('Hour of Day')
plt.ylabel('Number of Crimes')
plt.show()
```



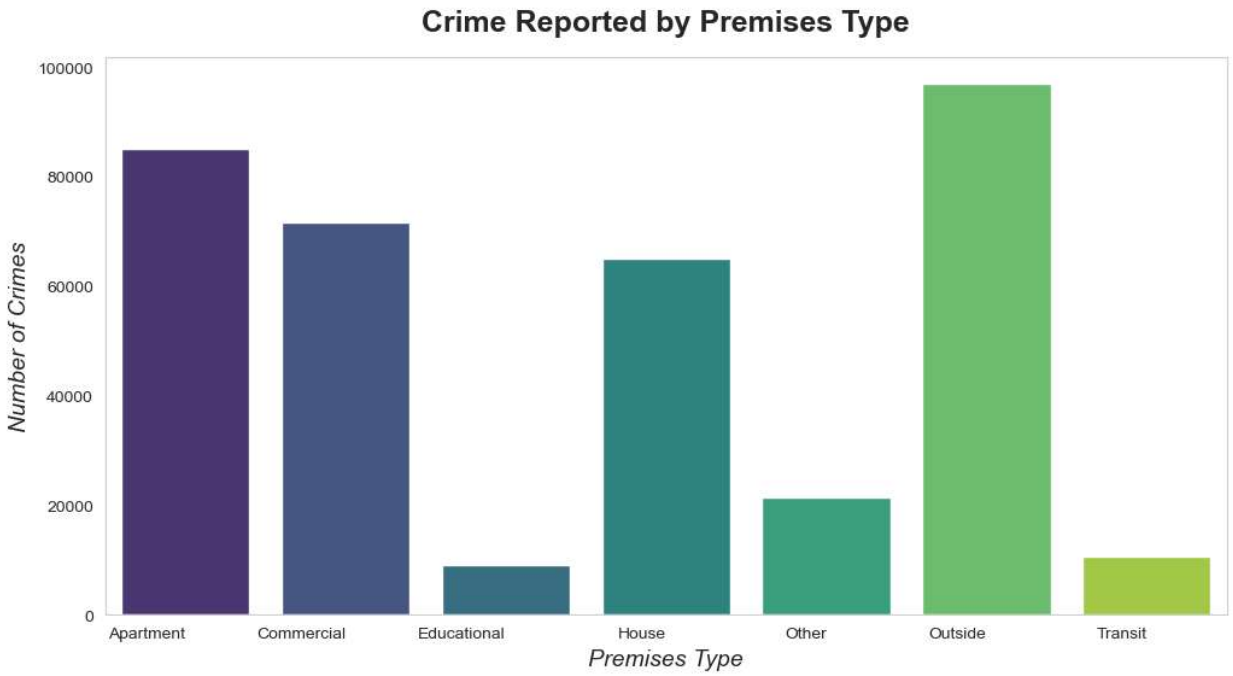
In [18]:
```
import matplotlib.pyplot as plt
import seaborn as sns

# Get the top 10 neighborhoods based on the number of crimes
```

```
top_neighborhoods = df['NEIGHBOURHOOD_140'].value_counts().nlargest(10).index

# Filter the dataset for the top 10 neighborhoods
df_top_neighborhoods = df[df['NEIGHBOURHOOD_140'].isin(top_neighborhoods)]

# Plot the count of each crime category in the top 10 neighborhoods
plt.figure(figsize=(16, 8))
sns.countplot(x='NEIGHBOURHOOD_140', hue='MCI_CATEGORY', data=df_top_neighborhoods)
plt.title('Crime Types by Top 10 Neighborhoods_140',fontsize=18, fontweight='bold', y=
plt.xlabel('Neighborhood')
plt.ylabel('Number of Crimes')
plt.xticks(rotation=45, ha='right')  # Rotate x-axis labels for better readability
plt.legend(title='Crime Type', bbox_to_anchor=(1, 1))
plt.show()
```



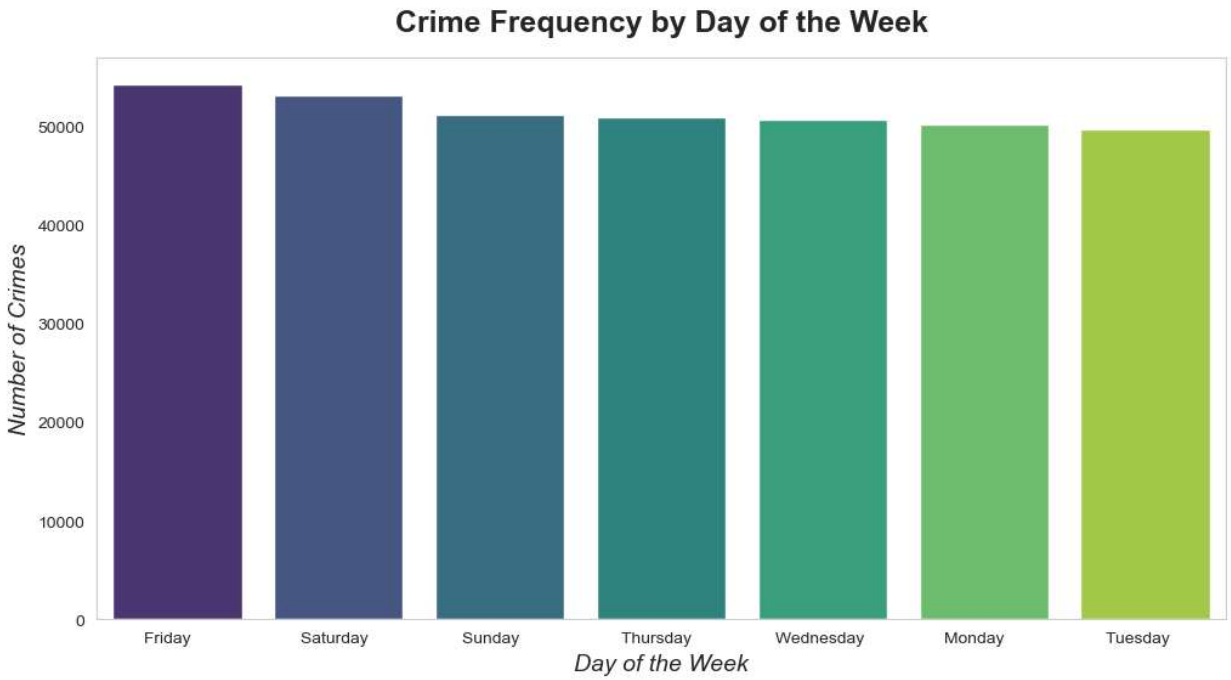
In [19]:

```
# Group by premises type and count the number of crimes
df_premises = df.groupby('PREMISES_TYPE').size().reset_index(name='count')
df_premises = df_premises.rename(columns={'X': 'Count'})

# Bar graph for Premises type distribution
sns.set_style('whitegrid')
plt.figure(figsize=(12,6))
ax = sns.barplot(x='PREMISES_TYPE', y='count', data=df_premises, palette='viridis')
ax.set_xticklabels(ax.get_xticklabels(), ha='right')
ax.set_title('Crime Reported by Premises Type',fontsize=18, fontweight='bold', y=1.03)
ax.set_xlabel('Premises Type',fontsize=14, fontstyle='italic')
ax.set_ylabel('Number of Crimes',fontsize=14, fontstyle='italic')
ax.grid(False)
plt.show()
```



In [20]:
```
#Check the count of occurrences during each day of the week
df_dayofweek = df.groupby('OCC_DOW').size().reset_index(name='count')
df_dayofweek = df_dayofweek.rename(columns={'X': 'count'})
df_dayofweek = df_dayofweek.sort_values(by='count', ascending=False)
print(df_dayofweek)

#Plot the count of occurrences during the week
sns.set_style('whitegrid')
plt.figure(figsize=(12,6))
ax = sns.barplot(x='OCC_DOW', y='count', data=df_dayofweek, palette='viridis')
ax.set_title('Crime Frequency by Day of the Week', fontsize=18, fontweight='bold', y=1
ax.set_xlabel('Day of the Week',fontsize=14, fontstyle='italic')
ax.set_ylabel('Number of Crimes',fontsize=14, fontstyle='italic')
ax.grid(False)
plt.show()
```

```
      OCC_DOW  count
0      Friday  54236
2    Saturday  53109
3      Sunday  51238
4    Thursday  50904
6   Wednesday  50732
1      Monday  50153
5     Tuesday  49704
```



In [21]:
```python
import seaborn as sns
import matplotlib.pyplot as plt

# Select numerical columns for correlation analysis
numerical_cols = df.select_dtypes(include='number')

# Calculate correlation matrix
corr_matrix = numerical_cols.corr()

# Create a heatmap
plt.figure(figsize=(12, 8))
sns.heatmap(corr_matrix, annot=True, cmap='coolwarm', fmt='.2f', linewidths=0.5)
plt.title('Correlation Matrix',fontsize=18, fontweight='bold', y=1.03)
plt.show()
```

Correlation Matrix