PROJECT QUESTION AND TASK ASSIGNED

Project Brief

Crime in Chicago has been worrisome to the Chicago PD. Reported crimes have been tracked on a daily basis since 2001 and have been provided in the project data file. The Chicago PD would like to drastically reduce the spate of voilent crimes reported in the city. Being effective involves knowing crime patterns and where they are likely to occur.
It also involves equipping the Police Department appropriately. They have recruited you to conduct full data analytics and uncover insights from the data that can be used to effectively prepare for and respond to crimes. They are interested in glleaning any insights that can help them determine What type of crimes to prepare for, Where these crimes are most likely to occur, What days of the week and periods to expect these crimes.

Task

Conduct a complete data analytics study and from your analytics, advise the Chicago PD accordingly.

CAPSTONE PROJECT

PROJECT ON: - CHICAGO CRIME DATA ANALYSIS

NAME: OLUWASEGUN AJAYI

CLASS: DATA ANALYTICS COHORT 12

USERNAME: oajayi_c12a

In [1]:
```
### TABLE OF CONTENT

#### Introduction
```

```
#### Importation of libraries and packages
#### Loading of dataset from directory
#### Inspection/Investigation/Scanning/Examining of the data.
#### Data Pre-Processing /Filtering.
#### Checkout for missing values and treat
#### Checkout for duplicates and treat.
#### Renaming labels of features
#### Sampling
#### Unpacking the date feature to add more features
#### Data Exploration and
#### Data Visualization with Obserservation and Inferences (using Seaborn, heatmap, pairplot, plotly)
#### Discoveries
#### Conclusion
#### Recommendation
#### Appendix
```

In [2]:
```
#   IMPORTATION OF LIBRARIES/PACKAGES THAT WILL BE NEEDED

import numpy as np
import pandas as pd
# importing visualization libraries
import seaborn as sns
import matplotlib.pyplot as plt
%matplotlib inline
from datetime import datetime

from jupyter_dash import JupyterDash

import dash
from dash import dcc
from dash import html
```

In [3]:
```
# LOADING THE DATASET INTO JUPYTER NOTEBOOK.

fileName = "crime_data_Proj1.CSV"

filePath ="C:/Users/OLUWATRIUMPH/Documents/Myworkspace/"
df = pd.read_csv(filePath + fileName)
```

In [4]:
```
### INSPECTION / SCANNING / EXAMINING THE GIVEN DATASET FOR ANALYSIS


df.head()
```

```
# This will automatically display the top (5) samples of the dataset.
```

Out[4]:

	Unnamed: 0	ID	Case Number	Date	Block	IUCR	Primary Type	Description	Location Description	Arrest	...	Ward	Commu A
0	0	6407111	HP485721	07/26/2008 02:30:00 PM	085XX S MUSKEGON AVE	1320	CRIMINAL DAMAGE	TO VEHICLE	STREET	False	...	10.0	
1	1	11398199	JB372830	07/31/2018 10:57:00 AM	092XX S ELLIS AVE	143C	WEAPONS VIOLATION	UNLAWFUL POSS AMMUNITION	POOL ROOM	True	...	8.0	
2	2	5488785	HN308568	04/27/2007 10:30:00 AM	062XX N TRIPP AVE	0610	BURGLARY	FORCIBLE ENTRY	RESIDENCE	True	...	39.0	
3	3	11389116	JB361368	07/23/2018 08:55:00 AM	0000X N KEELER AVE	0560	ASSAULT	SIMPLE	NURSING HOME/RETIREMENT HOME	False	...	28.0	
4	4	12420431	JE297624	07/11/2021 06:40:00 AM	016XX W HARRISON ST	051A	ASSAULT	AGGRAVATED - HANDGUN	PARKING LOT / GARAGE (NON RESIDENTIAL)	False	...	27.0	

5 rows × 23 columns

In [5]:
```
#                  DATA OBSERVATIONS AND INFERENCE FROM INSPECTION/SCANNING/EXAMINATION

# 1. From the above dataset cells I observed that there are 23 features (columns) and
# The importance of this Step is to help me find out if I will need to change the data type
# to a more appropriate format for my analysis.

#           NOTE
# The Date column will likely be changed to python's datatime.datetime format to:
# 1. Extract the month,
# 2. Period (time) and Hour
# 3. Day of the week and Daytype information.

# It was also observed that some features were duplicated and
# some features will be dropped as they will not be required in my final dataframe for analysis.
```

In [6]:
```
df.tail()
```

```
# This will display the last (5) samples of the dataset.
# This previews that there are Over 2 million samples (rows) in the dataset.
```

Out[6]:

	Unnamed: 0	ID	Case Number	Date	Block	IUCR	Primary Type	Description	Location Description	Arrest	...	War
2278721	2278721	10716043	HZ474139	10/14/2016 02:35:00 PM	006XX N CLARK ST	0560	ASSAULT	SIMPLE	CONVENIENCE STORE	True	...	42.
2278722	2278722	1740109	G546340	09/11/2001 10:20:00 PM	052XX W LAKE ST	0460	BATTERY	SIMPLE	RESIDENCE	False	...	Nal
2278723	2278723	4737434	HM342705	05/10/2006 07:49:00 PM	007XX E OAKWOOD BLVD	0560	ASSAULT	SIMPLE	APARTMENT	False	...	4.
2278724	2278724	11122832	JA476827	10/18/2017 10:30:00 PM	002XX W ERIE ST	1360	CRIMINAL TRESPASS	TO VEHICLE	PARKING LOT/GARAGE(NON.RESID.)	False	...	42.
2278725	2278725	3409804	HK420105	06/09/2004 08:19:28 PM	016XX N ROCKWELL ST	1811	NARCOTICS	POSS: CANNABIS 30GMS OR LESS	STREET	True	...	1.

5 rows × 23 columns

In [7]:

```
#         INTERPRETATION OF RESULT
# THIS SHOWS THE STRUCTURE (THE NUMBER OF FEATURES = 23 AND SAMPLES = 2,278,726 ), WITH LABELS OF THE DATA,
# This dataset has the following data type: bool(2), float64(7), int64(4), object(10),
# also the memory usage (size of the data) (369.4+ MB).

# Therefore, the data is not entirely numeric. Hence, there might be need to encode before modeling.

df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 2278726 entries, 0 to 2278725
Data columns (total 23 columns):
 #   Column                Dtype  
---  ------                -----  
 0   Unnamed: 0            int64  
 1   ID                    int64  
 2   Case Number           object 
 3   Date                  object 
 4   Block                 object 
 5   IUCR                  object 
 6   Primary Type          object 
 7   Description           object 
 8   Location Description  object 
 9   Arrest                bool   
 10  Domestic              bool   
 11  Beat                  int64  
 12  District              float64
 13  Ward                  float64
 14  Community Area        float64
 15  FBI Code              object 
 16  X Coordinate          float64
 17  Y Coordinate          float64
 18  Year                  int64  
 19  Updated On            object 
 20  Latitude              float64
 21  Longitude             float64
 22  Location              object 
dtypes: bool(2), float64(7), int64(4), object(10)
memory usage: 369.4+ MB
```

In [8]:
```
#         INTERPRETATION OF RESULT
# This specifically reveals the total samples and features of the data.
# 2,278,726 samples and 23 features respectively.


df.shape
```

Out[8]:
```
(2278726, 23)
```

In [9]:
```
#        INTERPRETATION OF RESULT
### This shows the summary statistics and distribution of the imported data
# Also, this will only return features that are only numeric.
```

```
### THIS SHOWS THE DATA DISTRIBUTION
### 1. The scale across the various features in the data are not identical and are wide apart,
### then, the data will need rescaling becausee they are wide apart if you are modelling.
### 2. The measure of dispersion in the data is large or not (the standard deviation column... measure of error in the
### 3. The mean and 50th percentile are close and not so far apart
### 4. Also, having considered the minimum and maximum values

# The data is poorly behaved after careful inspection of the statistical features
# Hence, the Chicago crime data requires a RESTANDARDIZATION of the data before usage,
# Having considered the features: MEAN, 50TH PERCENTILES, STANDARD DEVIATION, MINIMUM and MAXIMUM values.

# This reveals that it is not well behaved and will require rescaling to normallize the data before usage.


df.describe()
```

Out[9]:

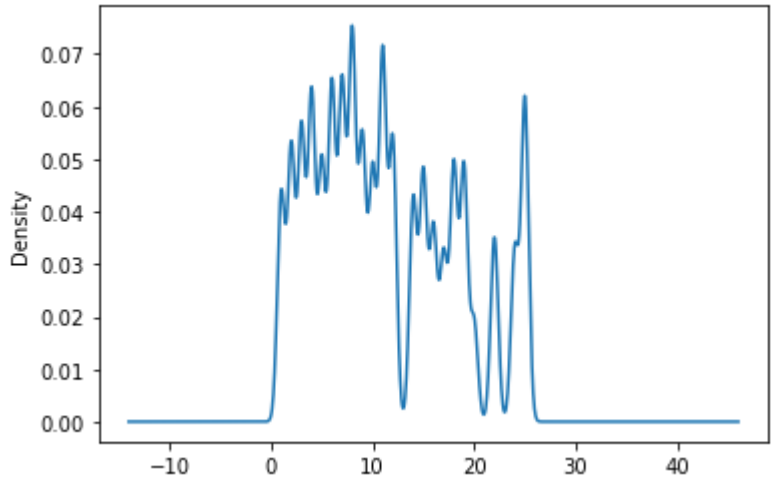
	Unnamed: 0	ID	Beat	District	Ward	Community Area	X Coordinate	Y Coordinate	Year	
count	2.278726e+06	2.278726e+06	2.278726e+06	2.278714e+06	2.094031e+06	2.094459e+06	2.254741e+06	2.254741e+06	2.278726e+06	2.2
mean	1.139362e+06	6.882068e+06	1.186442e+03	1.129072e+01	2.272764e+01	3.752140e+01	1.164569e+06	1.885747e+06	2.009638e+03	4.1
std	6.578117e+05	3.419168e+06	7.026836e+02	6.946692e+00	1.383464e+01	2.153282e+01	1.673955e+04	3.209855e+04	6.019724e+00	8.8
min	0.000000e+00	6.370000e+02	1.110000e+02	1.000000e+00	1.000000e+00	0.000000e+00	0.000000e+00	0.000000e+00	2.001000e+03	3.6
25%	5.696812e+05	3.716076e+06	6.210000e+02	6.000000e+00	1.000000e+01	2.300000e+01	1.152948e+06	1.859053e+06	2.004000e+03	4.1
50%	1.139362e+06	6.885990e+06	1.034000e+03	1.000000e+01	2.300000e+01	3.200000e+01	1.166060e+06	1.890673e+06	2.009000e+03	4.1
75%	1.709044e+06	9.887568e+06	1.731000e+03	1.700000e+01	3.400000e+01	5.700000e+01	1.176365e+06	1.909219e+06	2.014000e+03	4.1
max	2.278725e+06	1.278199e+07	2.535000e+03	3.100000e+01	5.000000e+01	7.700000e+01	1.205119e+06	1.951622e+06	2.022000e+03	4.2

In [10]:

```
# CHECKING THE DENSITY PLOT OF THE "District" DISTRIBUTION

df["District"].plot(kind="density")
```

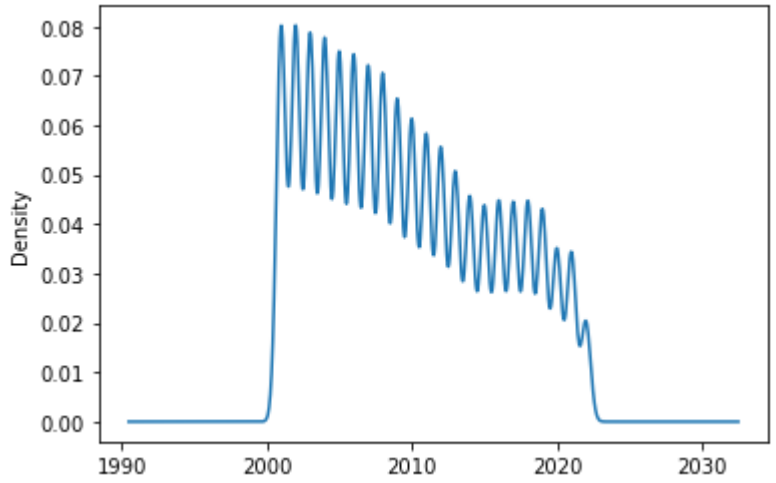
Out[10]:

```
<AxesSubplot:ylabel='Density'>
```



In [11]:
```
# CHECKING THE DENSITY PLOT OF THE "Year" DISTRIBUTION

df["Year"].plot(kind="density")
```

Out[11]:
```
<AxesSubplot:ylabel='Density'>
```



In [12]:
```
######

#df.plot(kind='density', subplots=True,layout= (4,5), sharex=False)
#plt.figure(figsize=(4,5))
#plt.show()
```

In [13]:
```
# CHECKING FOR THE UNIQUE DATA CHARACTERS FEATURES TO SEE IF THERE ARE MISTAKES

df.nunique()
```

Out[13]:
```
Unnamed: 0              2278726
ID                      2278726
Case Number             2278678
Date                    1343546
Block                     54347
IUCR                        392
Primary Type                 35
Description                 515
Location Description        198
Arrest                        2
Domestic                      2
Beat                        304
District                     23
Ward                         50
Community Area               78
FBI Code                     26
X Coordinate              73335
Y Coordinate             122589
Year                         22
Updated On                 4461
Latitude                 544919
Longitude                544655
Location                 545376
dtype: int64
```

In [14]:
```
# CHECKING TO SEE IF THERE ARE MISSING VALUES ACROSS ALL FEATURES IN THE DATA

df.isna().sum()

# THE RESULT SHOWS THAT THE DATA HAS SOME MISSING VALUES.
```

Out[14]:
```
Unnamed: 0                  0
ID                          0
Case Number                 1
Date                        0
Block                       0
IUCR                        0
Primary Type                0
Description                 0
Location Description     2877
Arrest                      0
Domestic                    0
Beat                        0
District                   12
Ward                   184695
Community Area         184267
FBI Code                    0
X Coordinate            23985
Y Coordinate            23985
Year                        0
Updated On                  0
Latitude                23985
Longitude               23985
Location                23985
dtype: int64
```

In [15]:
```
# This confirms the sum total number of missing values from the entire dataset
    # about 20% of the data are missing, therefore, it is not advisible to drop them
# because we might lose data representation but considering the perculiarity of each feature
# (such as: Location, date, day etc) then I will have to drop all.

df.isna().sum().sum()
```

Out[15]:
```
491777
```

In [16]:
```
# TO CHECK FOR DUPLICATES IN THE IMPORTED DATA "df"


df.duplicated()
```

Out[16]:
```
0          False
1          False
2          False
3          False
4          False
           ...
2278721    False
2278722    False
2278723    False
2278724    False
2278725    False
Length: 2278726, dtype: bool
```

In [17]:
```
# TO CHECK THE SUM TOTAL OF ALL DUPLICATED SAMPLE (ROW) VALUES

df.duplicated().sum()
```

Out[17]:
```
0
```

In [18]:
```
#                                NOTE:
# Having examined the dataset features, there is a need for me to drop some features
# that are not required to answer this project questions

# There is a need to remove or drop the duplicated column with "Unnamed: 0"
```

In [19]:
```
# To show labels of each features

df.columns
```

Out[19]:
```
Index(['Unnamed: 0', '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'],
      dtype='object')
```

In [20]:
```
#    FEATURES LIST TO DROP

# The following features (columns) will be dropped before proceeding to deal with missing values
#   ['Unnamed: 0', 'ID', 'Case Number', 'IUCR', 'Beat', 'Ward', 'Community Area', 'FBI Code',
#     'X Coordinate','Y Coordinate', 'Latitude', 'Longitude','Location']
```

In [21]:
```
# Filtering the Dataset out by dropping those features (columns) which would not be of use for my project analysis
```

```
df1 =  df[['Date', 'Block', 'Primary Type',
       'Description', 'Location Description', 'Arrest', 'Domestic',
       'District', 'Year', 'Updated On']]
```

In [22]:
```
# CHECKING TO SEE IF THE DUPLICATE COLUMN "Unnamed: 0" HAS BEEN REMOVED
# ALSO, THIS SHOWS THAT THE FEATURES NOT NEEDED HAS BEEN DROPPED.

df1.head()
```

Out[22]:

	Date	Block	Primary Type	Description	Location Description	Arrest	Domestic	District	Year	Updated On
0	07/26/2008 02:30:00 PM	085XX S MUSKEGON AVE	CRIMINAL DAMAGE	TO VEHICLE	STREET	False	False	4.0	2008	02/28/2018 03:56:25 PM
1	07/31/2018 10:57:00 AM	092XX S ELLIS AVE	WEAPONS VIOLATION	UNLAWFUL POSS AMMUNITION	POOL ROOM	True	False	4.0	2018	08/07/2018 04:02:59 PM
2	04/27/2007 10:30:00 AM	062XX N TRIPP AVE	BURGLARY	FORCIBLE ENTRY	RESIDENCE	True	False	17.0	2007	02/28/2018 03:56:25 PM
3	07/23/2018 08:55:00 AM	0000X N KEELER AVE	ASSAULT	SIMPLE	NURSING HOME/RETIREMENT HOME	False	False	11.0	2018	07/30/2018 03:52:24 PM
4	07/11/2021 06:40:00 AM	016XX W HARRISON ST	ASSAULT	AGGRAVATED - HANDGUN	PARKING LOT / GARAGE (NON RESIDENTIAL)	False	False	12.0	2021	07/18/2021 04:56:02 PM

In [23]:
```
# This reveals that the "dropped features" changes has been effected
df1.shape
```

Out[23]:
```
(2278726, 10)
```

In [24]:
```
# This displays the dataset structure before dropping and after dropping not needed features.

print('Dataset Shape before drop_duplicate : ', df.shape)
df.drop_duplicates(subset=['ID', 'Case Number'], inplace=True)
print('Dataset Shape after drop_duplicate: ', df1.shape)
```

```
Dataset Shape before drop_duplicate :  (2278726, 23)
Dataset Shape after drop_duplicate:  (2278726, 10)
```

In [25]:
```
# Checking the Nunmber of samples/records (i.e, the number of rows) and Features.
# This clarifies the changes of the dropped features.

df1.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 2278726 entries, 0 to 2278725
Data columns (total 10 columns):
 #   Column                Dtype 
---  ------                ----- 
 0   Date                  object
 1   Block                 object
 2   Primary Type          object
 3   Description           object
 4   Location Description  object
 5   Arrest                bool  
 6   Domestic              bool  
 7   District              float64
 8   Year                  int64 
 9   Updated On            object
dtypes: bool(2), float64(1), int64(1), object(6)
memory usage: 143.4+ MB
```

CLEANING OF THE DATASET

In [26]:
```
# CHECKING TO SEE IF THERE ARE MISSING VALUES ACROSS ALL FEATURES IN THE DATA (df1) AFTER DROPPING

df1.isna().sum()
```

Out[26]:
```
Date                       0
Block                      0
Primary Type               0
Description                0
Location Description    2877
Arrest                     0
Domestic                   0
District                  12
Year                       0
Updated On                 0
dtype: int64
```

In [27]:
```
# To know the total missing values in the dataset
# The missing values reduced from 491777 to 2889 after dropping.
```

```
df1.isna().sum().sum()
```

Out[27]:
```
2889
```

In [28]:
```
# Having dropped some of the dataset features not required for my analysis,
# the missing values reduced from 491777 (in df) to 2889 (in df1)
# Also, I chose to analyse the Chicago crime project location based on "district" because
# it has the least missing values aside "beat" which has no missing values.
```

In [29]:
```
#                NOTE

# I will embark on dropping the missing values in this dataset because of the delicacy of
# getting accurate results instead of padding or using mean or median methods.

# Dropping the samples (rows) will usually result in clean datasets and produce well-behaved data.
```

In [30]:
```
# How much of the data has been retained after this removal ?

print(round(2278726 / 2278725 * 100,2), "percentage of the data has been retained.")


# This is so because bulk of the features with missing values has been dropped before treating the missing values.
```

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

In [31]:
```
# To know the percentage of the data that will be lost
(2889/2278726)*100
```

Out[31]:
```
0.12678136818555633
```

In [32]:
```
# CLEANING TECHNIQUE

# The cleaning technique to use here would be to drop all the rows with atleast one missing value because
# the above result shows that I will lose around 10% of the data which is fair because i still have data representation


df2 = df1.dropna()
```

In [33]:
```
# How much of the data has been retained after this removal?

print(round(2070581  / 2278725 * 100,2), "percentage of the data has been retained.")
```

```
### the result shows percentage of the data being retained ater dropping missing values

# About 9.13% of the data was lost and it is fair without losing data representation
```

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

In [34]:
```
# To confirm that there are no more missing values in the dataset.

df2.isna().sum()
```

Out[34]:
```
Date                    0
Block                   0
Primary Type            0
Description             0
Location Description    0
Arrest                  0
Domestic                0
District                0
Year                    0
Updated On              0
dtype: int64
```

In [35]:
```
# The dataset "df2" now have zero(0) missing values

df2.isna().sum().sum()
```

Out[35]:
```
0
```

In [36]:
```
# TO CHECK THE SUM TOTAL OF ALL DUPLICATED SAMPLE (ROW) VALUES

df2.duplicated().sum()
```

Out[36]:
```
2108
```

In [37]:
```
# Dropping the duplicate

df3=df2.drop_duplicates()
```

In [38]:
```
# "df3" is void of duplicates and missing values

df3.duplicated().sum()
```

Out[38]:
```
0
```

In [39]:
```
# "df3" is void of duplicates and missing values

df3.isna().sum().sum()
```

Out[39]:
```
0
```

In [40]:
```
df3.columns
```

Out[40]:
```
Index(['Date', 'Block', 'Primary Type', 'Description', 'Location Description',
       'Arrest', 'Domestic', 'District', 'Year', 'Updated On'],
      dtype='object')
```

In [41]:
```
dfSub = df3[['Date', 'Block', 'Primary Type', 'Description', 'Location Description',
       'Arrest', 'Domestic', 'District', 'Year', 'Updated On']]
```

In [42]:
```
# CHECK FOR OUTLIERS

#### NO NEED
```

In [43]:
```
# Normalization of the data

# from sklearn.preprocessing import Normalizer

# from numpy import set_printoptions

# arrays = df.values       # CONVERTING TO ARRAYS , REMOVING HEADERS AND TURNS IT INTO LIST

# X = arrays[:,0:21]          # SEPARATING X AND Y      # TO PREDICT DAYS OF ABSENTEEISM
# Y = arrays[:,22]

# scaler =Normalizer().fit(X)
# normalizeX = scaler.transform(X)
```

NOTE

There are several columns that will help me answer the project questions.

I will unpack the 'Date' feature (column) to explore temporal patterns, 'Primary Type' and 'Location Description'

to investigate their relationship with time (month of the year, period of the day, hour of the day and daytpe).

In [44]:
```
### SLLICING A COPY OF THE DATASET

# This will help me work with just 20,000 samples records to enable me run the analysis faster.

dfSub = df3[0:20000]
```

In [45]:
```
# To view changes

dfSub.head()
```

Out[45]:

	Date	Block	Primary Type	Description	Location Description	Arrest	Domestic	District	Year	Updated On
0	07/26/2008 02:30:00 PM	085XX S MUSKEGON AVE	CRIMINAL DAMAGE	TO VEHICLE	STREET	False	False	4.0	2008	02/28/2018 03:56:25 PM
1	07/31/2018 10:57:00 AM	092XX S ELLIS AVE	WEAPONS VIOLATION	UNLAWFUL POSS AMMUNITION	POOL ROOM	True	False	4.0	2018	08/07/2018 04:02:59 PM
2	04/27/2007 10:30:00 AM	062XX N TRIPP AVE	BURGLARY	FORCIBLE ENTRY	RESIDENCE	True	False	17.0	2007	02/28/2018 03:56:25 PM
3	07/23/2018 08:55:00 AM	0000X N KEELER AVE	ASSAULT	SIMPLE	NURSING HOME/RETIREMENT HOME	False	False	11.0	2018	07/30/2018 03:52:24 PM
4	07/11/2021 06:40:00 AM	016XX W HARRISON ST	ASSAULT	AGGRAVATED - HANDGUN	PARKING LOT / GARAGE (NON RESIDENTIAL)	False	False	12.0	2021	07/18/2021 04:56:02 PM

In [46]:
```
dfSub.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 20000 entries, 0 to 20025
Data columns (total 10 columns):
 #   Column                Non-Null Count  Dtype  
---  ------                --------------  -----  
 0   Date                  20000 non-null  object 
 1   Block                 20000 non-null  object 
 2   Primary Type          20000 non-null  object 
 3   Description           20000 non-null  object 
 4   Location Description  20000 non-null  object 
 5   Arrest                20000 non-null  bool   
 6   Domestic              20000 non-null  bool   
 7   District              20000 non-null  float64
 8   Year                  20000 non-null  int64  
 9   Updated On            20000 non-null  object 
dtypes: bool(2), float64(1), int64(1), object(6)
memory usage: 1.4+ MB
```

In [47]:
```
#                   NOTE
# Having dropped the NaN (missing values), duplicated and features that will not be relevant to my analysis,
# then, I have to update the dataframe with the features that will be relevant in helping me understand
# the Chicago Crime dataset by unpacking the date feature for further analysis.
```

In [48]:
```
#   UNPACKING THE DATE: so create and add 3 more features

from dateutil.parser import parse
from datetime import datetime


tCol = dfSub.Date

List = [(datetime.ctime(parse(x[0:-3])),x[-2:]) for x in tCol]
dayList = []
monthList = []
periodList = []
for row in List:
    day = row[0][0:4]
    month = row[0][4:7]
    if row[1]=='AM':
        period = 'Morning'
    elif row[1] =='PM' and int(row[0][11:13])<4:
        period = 'Afternoon'
    elif row[1] =='PM' and int(row[0][11:13])<6:
        period = 'Evening'
    elif row[1] =='PM' and int(row[0][11:13])>5:
```

```
        period = 'Night'
    else:
        period = 'Unknown'

    dayList.append(day)
    monthList.append(month)
    periodList.append(period)

print(len(dayList), len(monthList), len(periodList))

dfSub['month'] = monthList
dfSub['day'] = dayList
dfSub['period']= periodList
dfSub.head()
```

```
20000 20000 20000
C:\Users\OLUWATRIUMPH\AppData\Local\Temp\ipykernel_7420\2998310997.py:33: SettingWithCopyWarning:
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/stable/user_guide/indexing.html#returning-a
-view-versus-a-copy
  dfSub['month'] = monthList
C:\Users\OLUWATRIUMPH\AppData\Local\Temp\ipykernel_7420\2998310997.py:34: SettingWithCopyWarning:
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/stable/user_guide/indexing.html#returning-a
-view-versus-a-copy
  dfSub['day'] = dayList
C:\Users\OLUWATRIUMPH\AppData\Local\Temp\ipykernel_7420\2998310997.py:35: SettingWithCopyWarning:
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/stable/user_guide/indexing.html#returning-a
-view-versus-a-copy
  dfSub['period']= periodList
```

Out[48]:

	Date	Block	Primary Type	Description	Location Description	Arrest	Domestic	District	Year	Updated On	month	day	per
0	07/26/2008 02:30:00 PM	085XX S MUSKEGON AVE	CRIMINAL DAMAGE	TO VEHICLE	STREET	False	False	4.0	2008	02/28/2018 03:56:25 PM	Jul	Sat	Afterno
1	07/31/2018 10:57:00 AM	092XX S ELLIS AVE	WEAPONS VIOLATION	UNLAWFUL POSS AMMUNITION	POOL ROOM	True	False	4.0	2018	08/07/2018 04:02:59 PM	Jul	Tue	Morn
2	04/27/2007 10:30:00 AM	062XX N TRIPP AVE	BURGLARY	FORCIBLE ENTRY	RESIDENCE	True	False	17.0	2007	02/28/2018 03:56:25 PM	Apr	Fri	Morn
3	07/23/2018 08:55:00 AM	0000X N KEELER AVE	ASSAULT	SIMPLE	NURSING HOME/RETIREMENT HOME	False	False	11.0	2018	07/30/2018 03:52:24 PM	Jul	Mon	Morn
4	07/11/2021 06:40:00 AM	016XX W HARRISON ST	ASSAULT	AGGRAVATED - HANDGUN	PARKING LOT / GARAGE (NON RESIDENTIAL)	False	False	12.0	2021	07/18/2021 04:56:02 PM	Jul	Sun	Morn

In [49]:
```
dfSub.day.unique()
```

Out[49]:
```
array(['Sat ', 'Tue ', 'Fri ', 'Mon ', 'Sun ', 'Wed ', 'Thu '],
      dtype=object)
```

In [50]:
```
wkList = []
for day in list(dfSub.day):
    if day in ['Sat ', 'Sun ']:
        wkList.append('Weekend')
    else:
        wkList.append('Weekday')
dfSub['dayType'] = wkList
```

```
C:\Users\OLUWATRIUMPH\AppData\Local\Temp\ipykernel_7420\86091064.py:7: SettingWithCopyWarning:
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/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  dfSub['dayType'] = wkList
```

In [51]:
```
# Adding Hour to the data features
# This will help me know the exact hour of the period of the day

datetime_column = pd.to_datetime(dfSub['Date'], format='%m/%d/%Y %I:%M:%S %p')
```

In [52]:
```
# Adding Hour to the data features

#df['month'] = datetime_column.dt.month
#df['day'] = datetime_column.dt.day
#df['weekday'] = datetime_column.dt.weekday_name.astype('category')

dfSub['Hour'] = datetime_column.dt.hour

dfSub.head()
```

```
C:\Users\OLUWATRIUMPH\AppData\Local\Temp\ipykernel_7420\3928066565.py:7: SettingWithCopyWarning:
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/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  dfSub['Hour'] = datetime_column.dt.hour
```

Out[52]:

	Date	Block	Primary Type	Description	Location Description	Arrest	Domestic	District	Year	Updated On	month	day	per
0	07/26/2008 02:30:00 PM	085XX S MUSKEGON AVE	CRIMINAL DAMAGE	TO VEHICLE	STREET	False	False	4.0	2008	02/28/2018 03:56:25 PM	Jul	Sat	Afterno
1	07/31/2018 10:57:00 AM	092XX S ELLIS AVE	WEAPONS VIOLATION	UNLAWFUL POSS AMMUNITION	POOL ROOM	True	False	4.0	2018	08/07/2018 04:02:59 PM	Jul	Tue	Morn
2	04/27/2007 10:30:00 AM	062XX N TRIPP AVE	BURGLARY	FORCIBLE ENTRY	RESIDENCE	True	False	17.0	2007	02/28/2018 03:56:25 PM	Apr	Fri	Morn
3	07/23/2018 08:55:00 AM	0000X N KEELER AVE	ASSAULT	SIMPLE	NURSING HOME/RETIREMENT HOME	False	False	11.0	2018	07/30/2018 03:52:24 PM	Jul	Mon	Morn
4	07/11/2021 06:40:00 AM	016XX W HARRISON ST	ASSAULT	AGGRAVATED - HANDGUN	PARKING LOT / GARAGE (NON RESIDENTIAL)	False	False	12.0	2021	07/18/2021 04:56:02 PM	Jul	Sun	Morn

In [53]:
```
# I created a list of dfSub.head(7)

dfSub.head(7)
```

Out[53]:

	Date	Block	Primary Type	Description	Location Description	Arrest	Domestic	District	Year	Updated On	month	day	per
0	07/26/2008 02:30:00 PM	085XX S MUSKEGON AVE	CRIMINAL DAMAGE	TO VEHICLE	STREET	False	False	4.0	2008	02/28/2018 03:56:25 PM	Jul	Sat	Afterno
1	07/31/2018 10:57:00 AM	092XX S ELLIS AVE	WEAPONS VIOLATION	UNLAWFUL POSS AMMUNITION	POOL ROOM	True	False	4.0	2018	08/07/2018 04:02:59 PM	Jul	Tue	Morn
2	04/27/2007 10:30:00 AM	062XX N TRIPP AVE	BURGLARY	FORCIBLE ENTRY	RESIDENCE	True	False	17.0	2007	02/28/2018 03:56:25 PM	Apr	Fri	Morn
3	07/23/2018 08:55:00 AM	0000X N KEELER AVE	ASSAULT	SIMPLE	NURSING HOME/RETIREMENT HOME	False	False	11.0	2018	07/30/2018 03:52:24 PM	Jul	Mon	Morn
4	07/11/2021 06:40:00 AM	016XX W HARRISON ST	ASSAULT	AGGRAVATED - HANDGUN	PARKING LOT / GARAGE (NON RESIDENTIAL)	False	False	12.0	2021	07/18/2021 04:56:02 PM	Jul	Sun	Morn
5	08/21/2001 12:00:00 AM	003XX W 28 PL	THEFT	OVER $500	STREET	False	False	2.0	2001	08/17/2015 03:03:40 PM	Aug	Tue	Morn
6	10/14/2006 10:00:00 PM	006XX S CENTRAL AVE	ROBBERY	STRONGARM - NO WEAPON	CTA PLATFORM	False	False	15.0	2006	02/28/2018 03:56:25 PM	Oct	Sat	Ni

In [54]:
```
# RENAMING OF SOME LABELS (COLUMNS) HEADERS


dfSub = dfSub.rename(columns={'Primary Type': 'Primary_Type', 'Location Description':
                              'Location_Description', 'Updated On': 'Updated_On',
                              'month': 'Month', 'day': 'Day', 'period': 'Period', 'dayType': 'DayType'})
```

In [55]:
```
# This confirms that the features has been renamed.
```

```
dfSub.columns
```

Out[55]:
```
Index(['Date', 'Block', 'Primary_Type', 'Description', 'Location_Description',
       'Arrest', 'Domestic', 'District', 'Year', 'Updated_On', 'Month', 'Day',
       'Period', 'DayType', 'Hour'],
      dtype='object')
```

In [56]:
```
#         INTERPRETATION OF RESULT
# THIS SHOWS THE STRUCTURE (THE NUMBER OF FEATURES = 15 AND SAMPLES = 20,000), WITH LABELS OF THE DATA,
# This dataset has the following data type: bool(2), float64(1), int64(2), object(10),
# also the memory usage (size of the data) (2.2+ MB).

# Therefore, the data is not entirely numeric. Hence, there might be need to encode before modeling.


dfSub.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 20000 entries, 0 to 20025
Data columns (total 15 columns):
 #   Column                Non-Null Count  Dtype
---  ------                --------------  -----
 0   Date                  20000 non-null  object
 1   Block                 20000 non-null  object
 2   Primary_Type          20000 non-null  object
 3   Description           20000 non-null  object
 4   Location_Description  20000 non-null  object
 5   Arrest                20000 non-null  bool
 6   Domestic              20000 non-null  bool
 7   District              20000 non-null  float64
 8   Year                  20000 non-null  int64
 9   Updated_On            20000 non-null  object
 10  Month                 20000 non-null  object
 11  Day                   20000 non-null  object
 12  Period                20000 non-null  object
 13  DayType               20000 non-null  object
 14  Hour                  20000 non-null  int64
dtypes: bool(2), float64(1), int64(2), object(10)
memory usage: 2.2+ MB
```

In [57]:
```
# The features has increased by 5.

dfSub.shape
```

Out[57]:
```
(20000, 15)
```

In [58]:
```
dfFinal = dfSub[['Date', 'Block', 'Primary_Type', 'Description', 'Location_Description',
       'Arrest', 'Domestic', 'District', 'Year', 'Updated_On', 'Month', 'Day',
       'Period', 'DayType', 'Hour']]
```

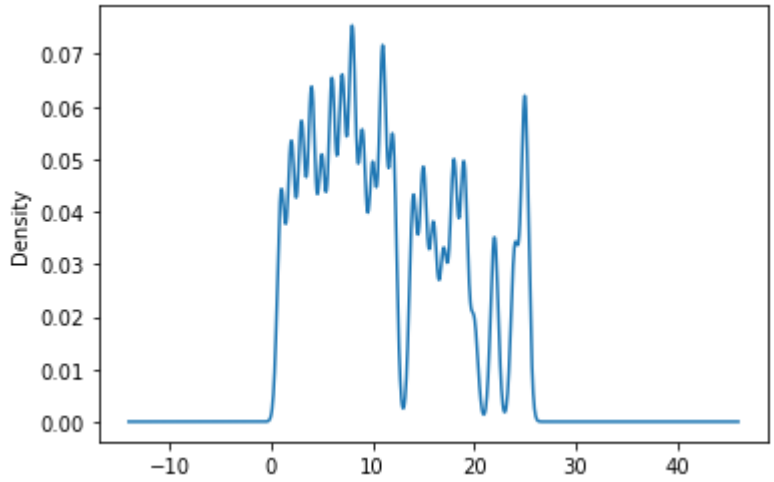
In [59]:
```
dfFinal.head()
```

Out[59]:

	Date	Block	Primary_Type	Description	Location_Description	Arrest	Domestic	District	Year	Updated_On	Month	Day
0	07/26/2008 02:30:00 PM	085XX S MUSKEGON AVE	CRIMINAL DAMAGE	TO VEHICLE	STREET	False	False	4.0	2008	02/28/2018 03:56:25 PM	Jul	Sat
1	07/31/2018 10:57:00 AM	092XX S ELLIS AVE	WEAPONS VIOLATION	UNLAWFUL POSS AMMUNITION	POOL ROOM	True	False	4.0	2018	08/07/2018 04:02:59 PM	Jul	Tue
2	04/27/2007 10:30:00 AM	062XX N TRIPP AVE	BURGLARY	FORCIBLE ENTRY	RESIDENCE	True	False	17.0	2007	02/28/2018 03:56:25 PM	Apr	Fri
3	07/23/2018 08:55:00 AM	0000X N KEELER AVE	ASSAULT	SIMPLE	NURSING HOME/RETIREMENT HOME	False	False	11.0	2018	07/30/2018 03:52:24 PM	Jul	Mon
4	07/11/2021 06:40:00 AM	016XX W HARRISON ST	ASSAULT	AGGRAVATED - HANDGUN	PARKING LOT / GARAGE (NON RESIDENTIAL)	False	False	12.0	2021	07/18/2021 04:56:02 PM	Jul	Sun

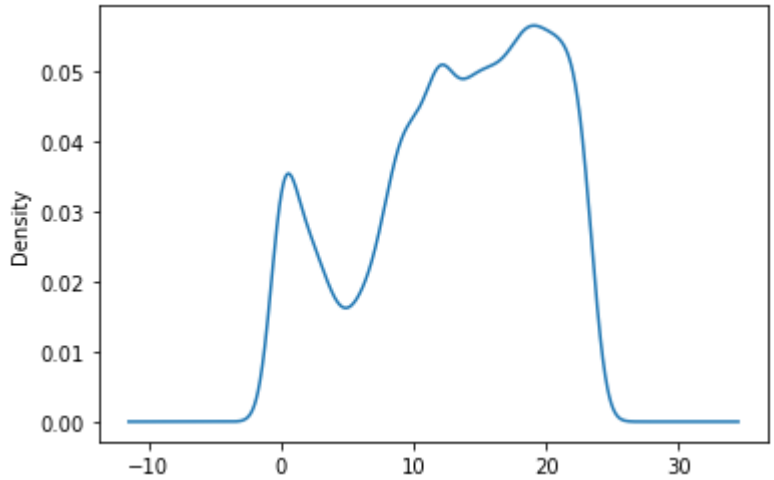
In [60]:
```
# Checking the density plot
# The data is not evenly distributed.

df["District"].plot(kind="density")

# This is multimodal
```

Out[60]:
```
<AxesSubplot:ylabel='Density'>
```

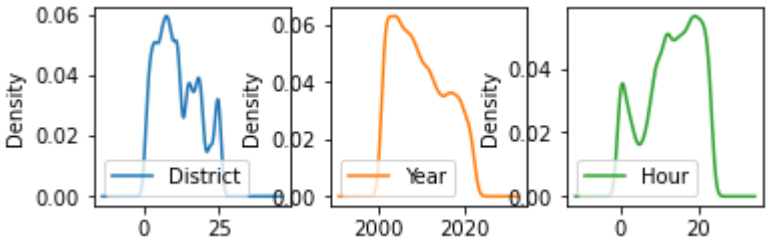


In [61]:
```
dfFinal["Hour"].plot(kind="density")
```

Out[61]:
```
<AxesSubplot:ylabel='Density'>
```



In [62]:
```
# Checking the density plot of District, Hour and Year

dfFinal.plot(kind='density', subplots=True,layout= (2,3), sharex=False)
plt.figure(figsize=(4,6))
plt.show()
```




```
<Figure size 288x432 with 0 Axes>
```

In [63]:
```
#         INTERPRETATION OF RESULT
# THIS SHOWS THE SHAPE AND STRUCTURE (With 15 FEATURES AND 20,000 SAMPLES) AND 15 LABELS OF THE NEW DATA,
# This reveals the data type: bool(2), float64(1), int64(2), object(10),
# also the memory usage reduced (size of the data) (2.2+ MB).

# Therefore, the data is still not entirely numeric. Hence, there wont be need to encode since I am not modelling.


dfFinal.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 20000 entries, 0 to 20025
Data columns (total 15 columns):
 #   Column                Non-Null Count  Dtype
---  ------                --------------  -----
 0   Date                  20000 non-null  object
 1   Block                 20000 non-null  object
 2   Primary_Type          20000 non-null  object
 3   Description           20000 non-null  object
 4   Location_Description  20000 non-null  object
 5   Arrest                20000 non-null  bool
 6   Domestic              20000 non-null  bool
 7   District              20000 non-null  float64
 8   Year                  20000 non-null  int64
 9   Updated_On            20000 non-null  object
 10  Month                 20000 non-null  object
 11  Day                   20000 non-null  object
 12  Period                20000 non-null  object
 13  DayType               20000 non-null  object
 14  Hour                  20000 non-null  int64
dtypes: bool(2), float64(1), int64(2), object(10)
memory usage: 2.2+ MB
```

In [64]:
```
dfFinal.shape
```

Out[64]:
```
(20000, 15)
```

In [65]:
```
# Number of distinct crimes in the city

dfFinal = dfFinal['Primary_Type'].unique()
print("The Number of distinct crimes in Chicago in the year(s):", len(dfFinal))
print()
print("The Distinct Crimes are :\n", dfFinal)
```

```
The Number of distinct crimes in Chicago in the year(s): 32

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

In [66]:
```
dfFinal = dfSub[['Date', 'Block', 'Primary_Type', 'Description', 'Location_Description',
       'Arrest', 'Domestic', 'District', 'Year', 'Updated_On', 'Month', 'Day',
       'Period', 'DayType', 'Hour']]
```

In [67]:
```
# Categorical variables in the dataset
catFeature = [col for col in dfFinal.columns if pd.api.types.is_string_dtype(dfFinal[col]) ]

print(catFeature)
labelDict ={}
for col in catFeature:
      labels = dfFinal[col].unique()
      labelDict[col] = labels
```

```
['Date', 'Block', 'Primary_Type', 'Description', 'Location_Description', 'Updated_On', 'Month', 'Day', 'Period', 'DayTy
pe']
```

In [68]:
```
###             Note
#'Location Description', 'Description' and 'Primary Type' columns are Categorical columns,
# I will keep the most frequent categories and then cast them to a categorical data type.
```

In [69]:
```
loc_to_change  = list(dfFinal['Location_Description'].value_counts()[15:].index)
desc_to_change = list(dfFinal['Description'].value_counts()[15:].index)
```

```
type_to_change = list(dfFinal['Primary_Type'].value_counts()[15:].index)

dfSub.loc[dfFinal['Location_Description'].isin(loc_to_change) , dfFinal.columns=='Location_Description'] = 'OTHER'
dfSub.loc[dfFinal['Description'].isin(desc_to_change) , dfFinal.columns=='Description'] = 'OTHER'
dfSub.loc[dfFinal['Primary_Type'].isin(type_to_change) , dfFinal.columns=='Primary_Type'] = 'OTHER'
```

In [70]:
```
#### Converting those columns in 'Categorical' data type

dfFinal['Primary_Type']         = pd.Categorical(dfFinal['Primary_Type'])
dfFinal['Location_Description'] = pd.Categorical(dfFinal['Location_Description'])
dfFinal['Description']          = pd.Categorical(dfFinal['Description'])
```

UNIVARIATE ANALYSIS FOR VISUALIZATION

In [71]:
```
# This reveals important insights showing most of the crimes that occurs in streets or residence of people.
```

In [72]:
```
# Top 10 crimes in Chicago

pd.value_counts(df['Primary Type'])[:10]
```

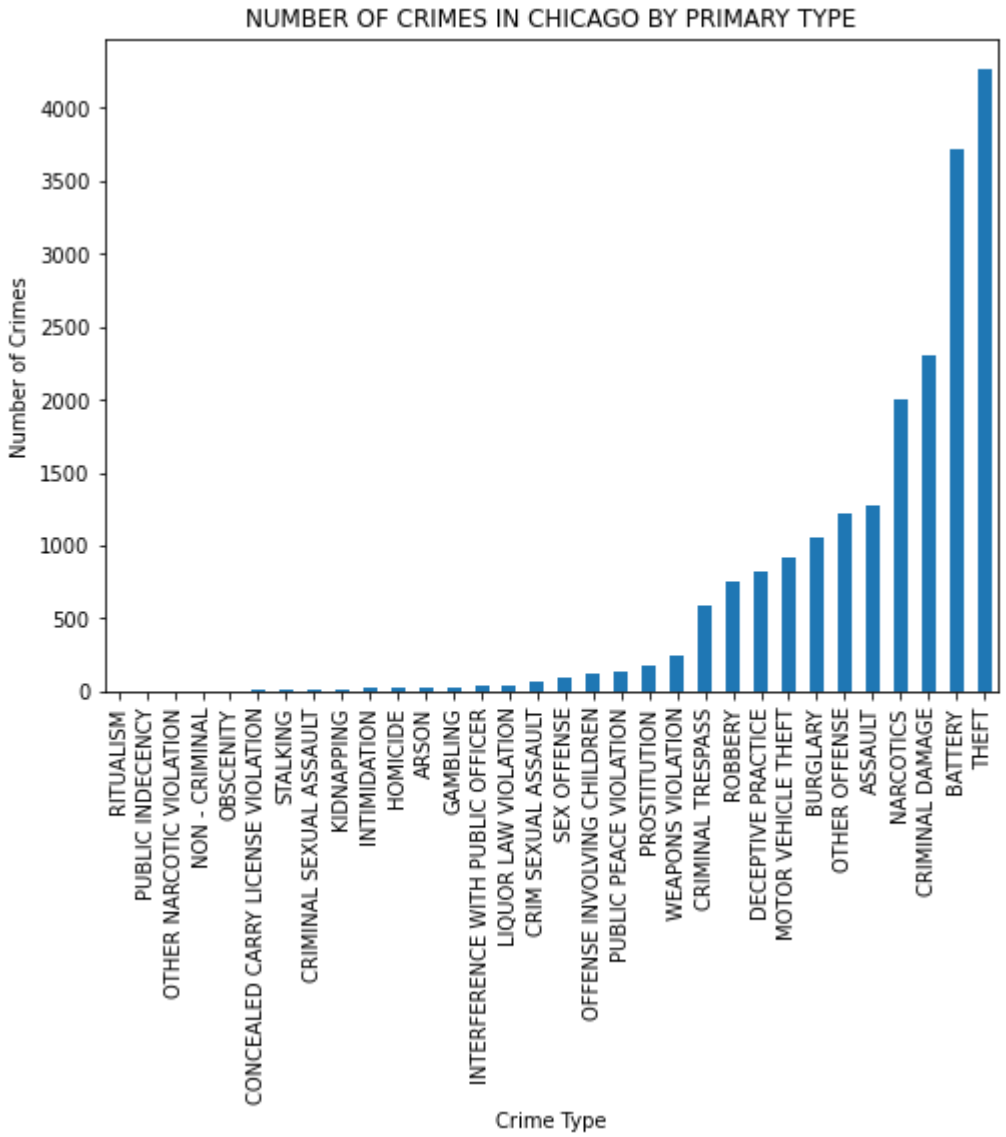
Out[72]:
```
THEFT                  479375
BATTERY                418651
CRIMINAL DAMAGE        259962
NARCOTICS              222861
ASSAULT                147666
OTHER OFFENSE          141627
BURGLARY               125163
MOTOR VEHICLE THEFT    105854
DECEPTIVE PRACTICE      99352
ROBBERY                 85695
Name: Primary Type, dtype: int64
```

In [73]:
```
dfFinal["Primary_Type"].value_counts()
```

Out[73]:
```
THEFT                                4263
BATTERY                              3718
CRIMINAL DAMAGE                      2301
NARCOTICS                            1999
ASSAULT                              1269
OTHER OFFENSE                        1220
BURGLARY                             1059
MOTOR VEHICLE THEFT                   924
DECEPTIVE PRACTICE                    820
ROBBERY                               756
CRIMINAL TRESPASS                     586
WEAPONS VIOLATION                     243
PROSTITUTION                          180
PUBLIC PEACE VIOLATION                133
OFFENSE INVOLVING CHILDREN            127
SEX OFFENSE                            91
CRIM SEXUAL ASSAULT                    67
LIQUOR LAW VIOLATION                   40
INTERFERENCE WITH PUBLIC OFFICER       35
GAMBLING                               33
ARSON                                  32
HOMICIDE                               26
INTIMIDATION                           23
KIDNAPPING                             17
CRIMINAL SEXUAL ASSAULT                14
STALKING                               11
CONCEALED CARRY LICENSE VIOLATION       6
OBSCENITY                               3
OTHER NARCOTIC VIOLATION                1
PUBLIC INDECENCY                        1
NON - CRIMINAL                          1
RITUALISM                               1
Name: Primary_Type, dtype: int64
```

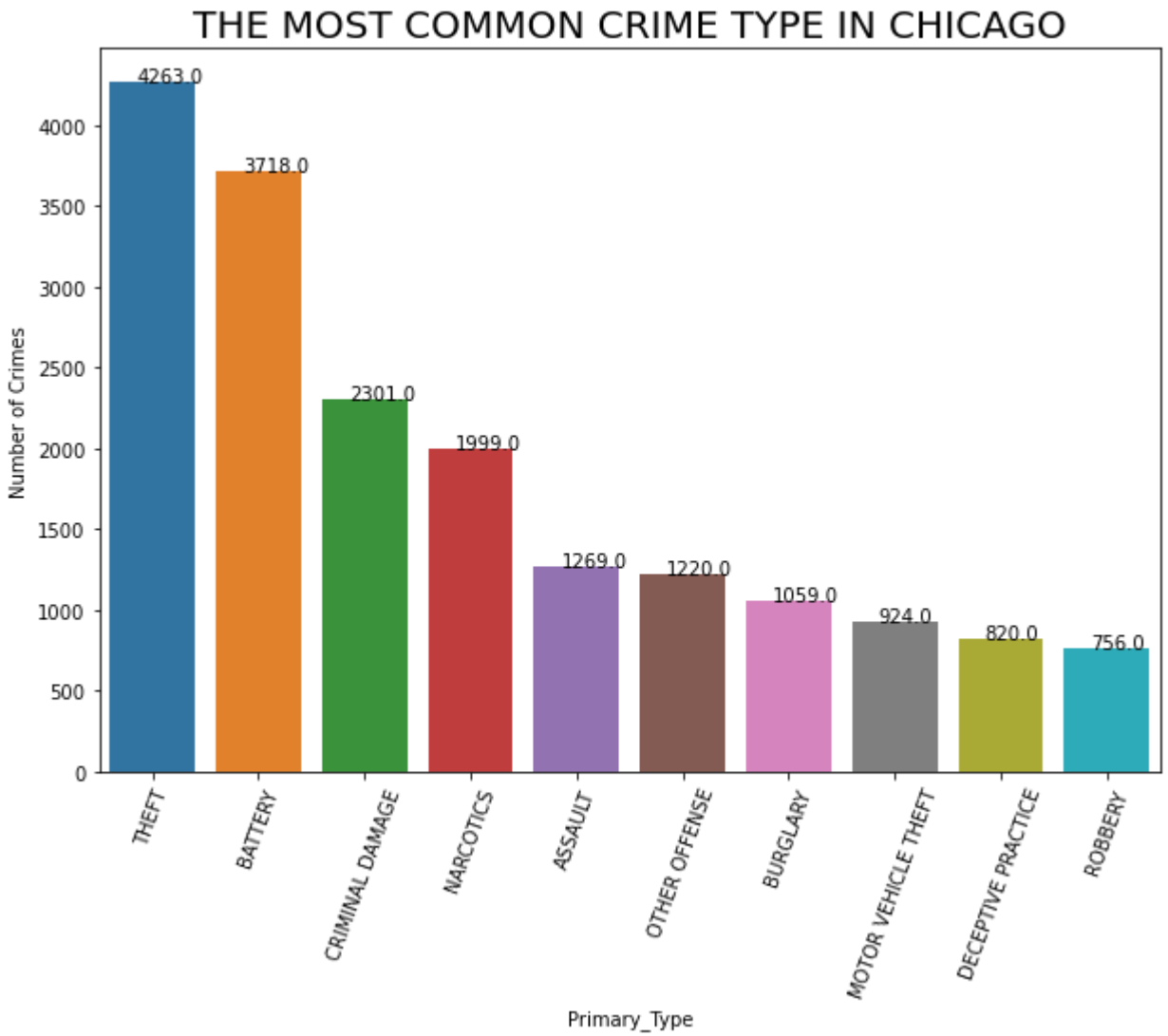
In [74]:
```
# To look at the distribution of crime by their types, which crimes are most common among
# the top 20 most frequent crime types

plt.figure(figsize=(8,6))
dfFinal.groupby([dfFinal['Primary_Type']]).size().sort_values(ascending=True).plot(kind='bar')

plt.xlabel('Crime Type')
plt.ylabel('Number of Crimes')

plt.title('NUMBER OF CRIMES IN CHICAGO BY PRIMARY TYPE')
```

```
plt.show()


# Theft cases are the most common crimes in Chicago followed by: battery,criminal damages, narcotics as shown below.
```



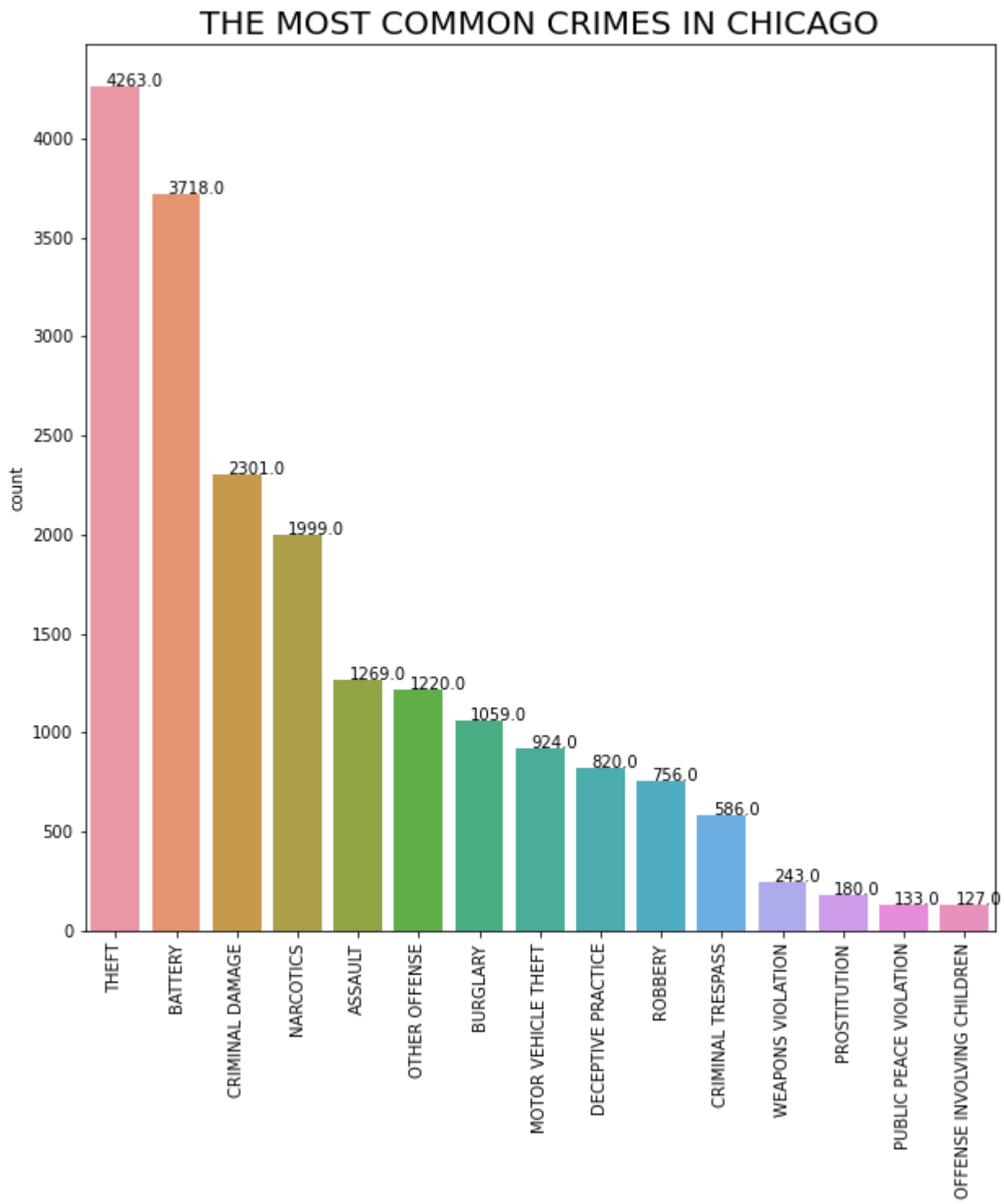
In [75]:
```
# To find the 10 most common crimes, as this will help me steer the direction of the project to
# some specific crimes at later stages of the project analysis.

plt.figure(figsize = (10, 7))
```

```
ax=sns.countplot(x= 'Primary_Type', data = dfFinal, order = dfFinal['Primary_Type'].value_counts().iloc[:10].index)
plt.ylabel('Number of Crimes')
#plt.xticks(rotation=70)
ax.set_title('THE MOST COMMON CRIME TYPE IN CHICAGO', fontsize=20)
'''for p in ax.patches:
 ax.annotate('%{:.1f}'.format(p.get_height()), (p.get_x()+0.1, p.get_height()+50))'''

for p in ax.patches:
 ax.annotate('{:.1f}'.format(p.get_height()), (p.get_x()+0.25, p.get_height()+0.01))
plt.xticks(rotation=70)

plt.show()


# Theft cases are the most common crimes in Chicago followed by: battery, criminal damages, narcotics as shown below.
# Not all crimes are the same. Some crimes types are more likely to occur than other types depending on the place and t
```



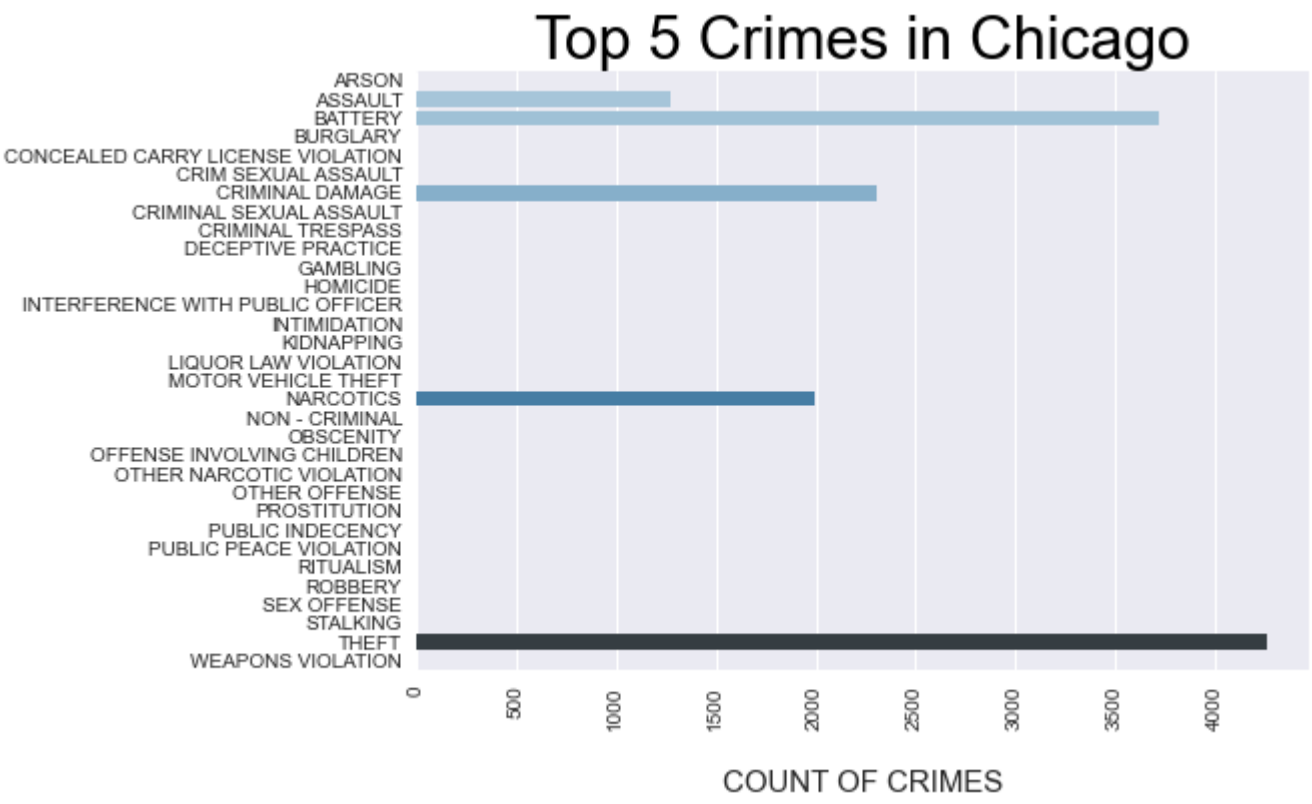
In [76]:
```
# Top 15 Crime type

plt.figure(figsize = (10, 10))
plt.xlabel('Crimes Type')
plt.ylabel('Number of Crimes')
#plt.ylabel('Frequency [%]')
ax = sns.countplot(x= 'Primary_Type', data = dfFinal, order =
             dfFinal['Primary_Type'].value_counts().iloc[:15].index)
ax.set_title('THE MOST COMMON CRIMES IN CHICAGO', fontsize=20)
'''for p in ax.patches:
```

```
 ax.annotate('%{:.1f}'.format(p.get_height()), (p.get_x()+0.1, p.get_height()+50))'''

for p in ax.patches:
 ax.annotate('{:.1f}'.format(p.get_height()), (p.get_x()+0.25, p.get_height()+0.01))
plt.xticks(rotation=90)

plt.show()
```


THE MOST COMMON CRIMES IN CHICAGO
count
0
500
1000
1500
2000
2500
3000
3500
4000
4263.0
3718.0
2301.0
1999.0
1269.0
1220.0
1059.0
924.0
820.0
756.0
586.0
243.0
180.0
133.0
127.0
THEFT
BATTERY
CRIMINAL DAMAGE
NARCOTICS
ASSAULT
OTHER OFFENSE
BURGLARY
MOTOR VEHICLE THEFT
DECEPTIVE PRACTICE
ROBBERY
CRIMINAL TRESPASS
WEAPONS VIOLATION
PROSTITUTION
PUBLIC PEACE VIOLATION
OFFENSE INVOLVING CHILDREN

Primary_Type

In [77]:
```
# Set the style of the plot first
plt.style.use('seaborn')

# Filter out the Top 5 crimes
top_5_crimes = dfFinal['Primary_Type'].value_counts().sort_values(ascending=False).head()

temp = dfFinal.groupby('Primary_Type', as_index=False).agg({"Location_Description": "count"})
temp = temp.sort_values(by=['Location_Description'], ascending=False).head()
temp = temp.sort_values(by='Location_Description', ascending=True)
sns.barplot(x='Location_Description', y='Primary_Type', data=temp, palette="Blues_d")


# Work on the aestehtic appeal of the plot
plt.title("Top 5 Crimes in Chicago", fontdict = {'fontsize': 30, 'fontname':'Arial', 'color': '#000000'})
plt.xlabel("\nCOUNT OF CRIMES", fontdict = {'fontsize': 15})
plt.ylabel("")
plt.xticks(rotation=90)
plt.show()
#plt.show()


# The Primary Type field contains the type for the crime.
# To investigate the most frequent type of crime in the Chicago. Therefore,

# Theft is the most common type of crime in the city of Chicago.
```



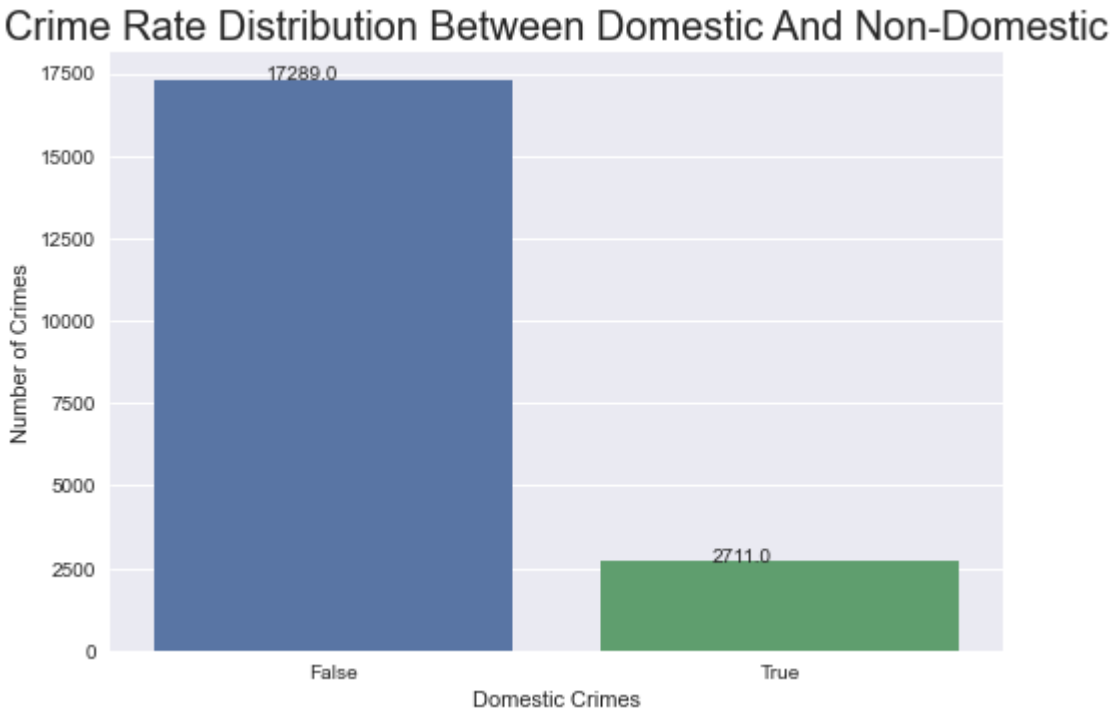
In [78]:
```
# Doing a bit of df manipulation for using bokeh
temp.head()
temp.columns=['Crime','Number']
temp.index=[0,1,2,3,4]
temp['co-ordinates']=[1,2,3,4,5]
temp.head()
```

Out[78]:

	Crime	Number	co-ordinates
0	ASSAULT	1269	1
1	NARCOTICS	1999	2
2	CRIMINAL DAMAGE	2301	3
3	BATTERY	3718	4
4	THEFT	4263	5

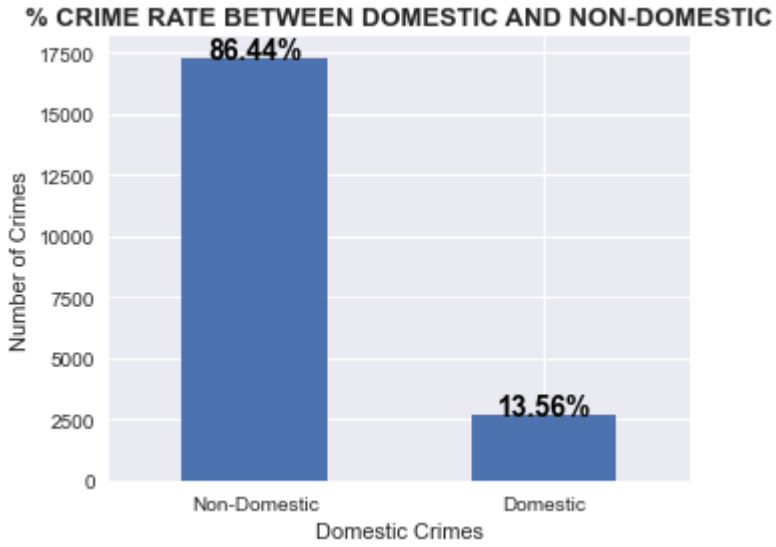
In [79]:
```
#      NOTE
# Having examined each crime type and frequency of occurrence, it is therefore important
# to know if these crimes are mostly DOMESTIC OR NON-DOMESTICS


plt.title('CRIME RATE BETWEEN DOMESTIC AND NON-DOMESTIC')
plt.xlabel('Number of Axles')
plt.ylabel('Frequency [%]')
ax = sns.countplot(x="Domestic", data=dfFinal)
ax.set_title('Crime Rate Distribution Between Domestic And Non-Domestic', fontsize=20)
'''for p in ax.patches:
 ax.annotate('%{:.1f}'.format(p.get_height()), (p.get_x()+0.1, p.get_height()+50))'''
for p in ax.patches:
 ax.annotate('{:.1f}'.format(p.get_height()), (p.get_x()+0.25, p.get_height()+0.01))

plt.ylabel('Number of Crimes')
plt.xlabel('Domestic Crimes')

dfFinal["Domestic"].value_counts()
```

Out[79]:
```
False    17289
True      2711
Name: Domestic, dtype: int64
```



In [80]:
```
print(dfFinal.Domestic.value_counts())

# dfFinal['Domestic'].value_counts().plot(kind='bar').set_title('% CRIMES BETWEEN DOMESTIC AND NON-DOMESTIC')
#Simple plot
fig, ax = plt.subplots(figsize=(5,4))
name = ["Non-Domestic", "Domestic"]
ax = dfFinal.Domestic.value_counts().plot(kind='bar')
ax.set_title("% CRIME RATE BETWEEN DOMESTIC AND NON-DOMESTIC", fontsize = 13, weight = 'bold')
ax.set_xticklabels (name, rotation = 0)
# To calculate the percentage
totals = []
for i in ax.patches:
 totals.append(i.get_height())
total = sum(totals)
for i in ax.patches:
 ax.text(i.get_x()+.09, i.get_height()-50, \
 str(round((i.get_height()/total)*100, 2))+'%', fontsize=14,
 color='black', weight = 'bold')

plt.ylabel('Number of Crimes')
plt.xlabel('Domestic Crimes')


plt.tight_layout()
```

```
False    17289
True      2711
Name: Domestic, dtype: int64
```



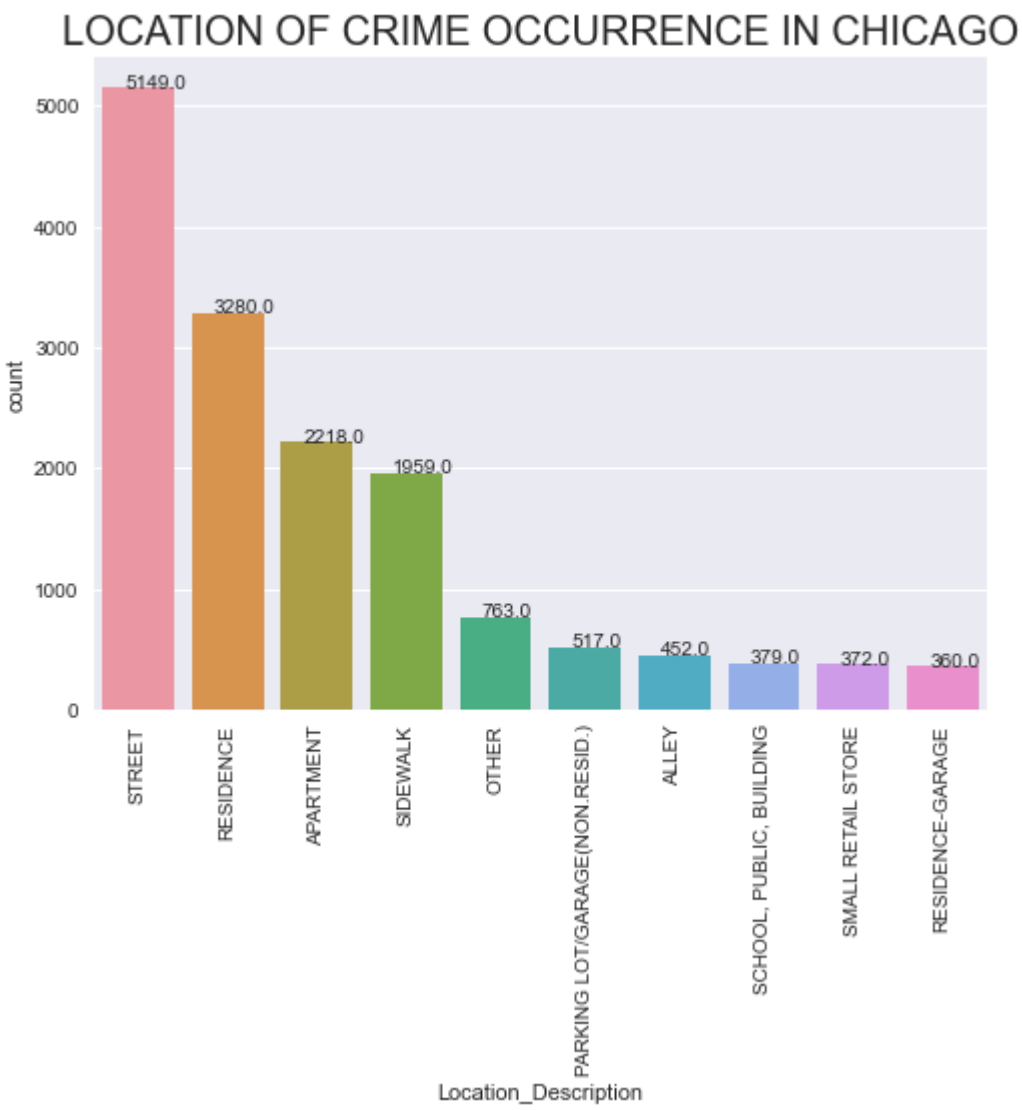
In [81]:
```
#       INFERRENCE FOR DOMESTIC CRIME
# Since most of the crimes were non-domestics (86.44%) which means that they happened outside the home or house.
# Then, the QUESTION is: WHERE EXACTLY DID THESE CRIME HAPPEN OUTSIDE ???

# If most of the crimes were non-domestic then which of the non-demestic location has the most crime rate occurrence.
# Hence, I wil proceed to visualize and analyse where (location of) the crime occurs and frequency.
```

In [82]:
```
# Top 10 location for crimes

# pd.value_counts(df['Location Description'])[:10]

dfFinal["Location_Description"].value_counts()
```

Out[82]:
```
STREET                                    5149
RESIDENCE                                 3280
APARTMENT                                 2218
SIDEWALK                                  1959
OTHER                                      763
                                          ...
CHURCH / SYNAGOGUE / PLACE OF WORSHIP        1
CTA PARKING LOT / GARAGE / OTHER PROPERTY    1
GANGWAY                                      1
PARKING LOT                                  1
YARD                                         1
Name: Location_Description, Length: 126, dtype: int64
```

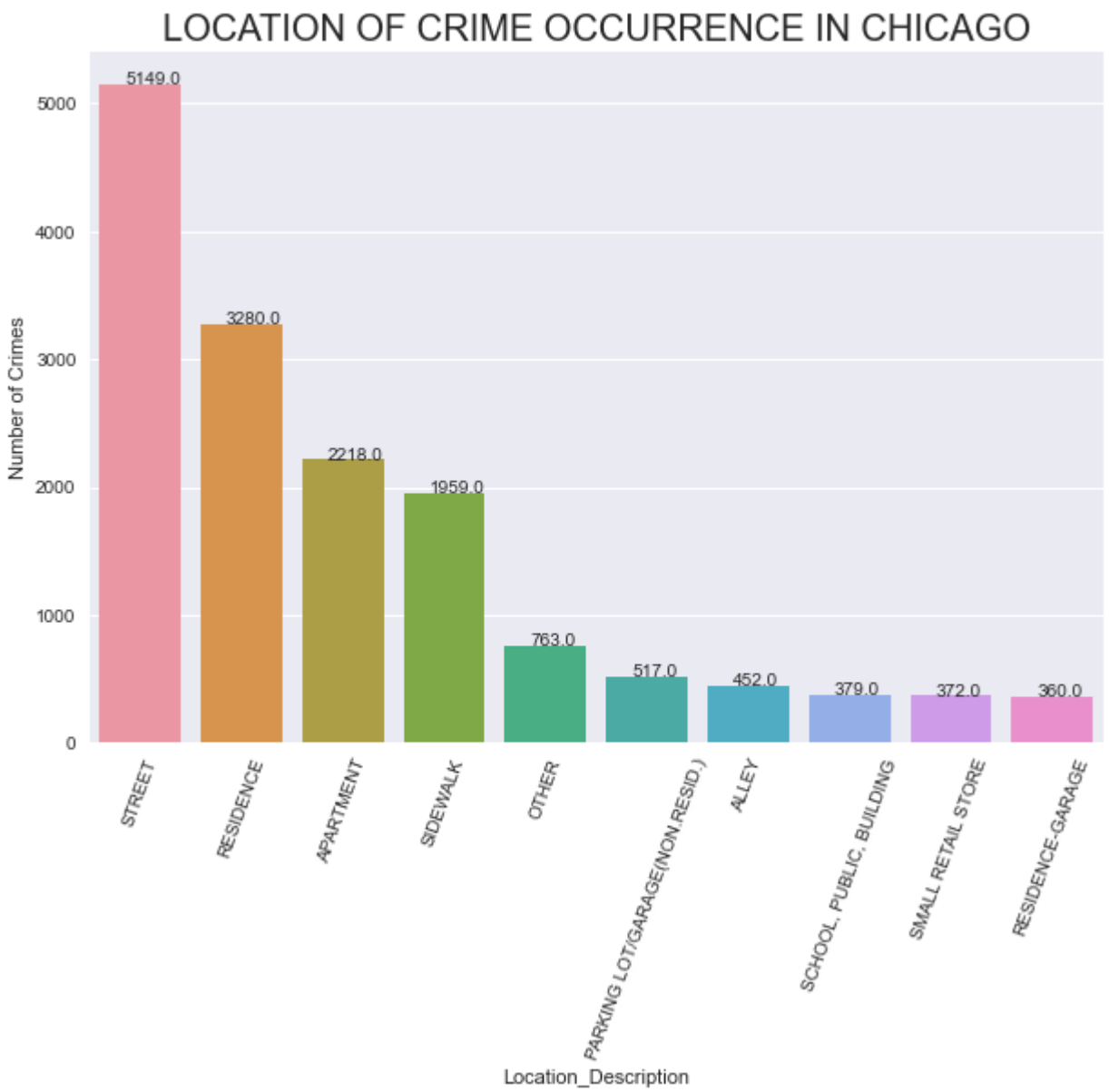
```
# Univariate
```

In [83]:
```
# This plot shows the relation between crime type and crime location.
# It indicates that most of the theft activities occur on streets.
# To find where most of these crimes usually occur chronologically as this
# could be helpful in determining where to station the police force.

plt.figure(figsize = (8, 6))
#plt.title('LOCATION OF CRIME OCCURRENCE IN CHICAGO')
plt.xlabel('Crimes Location Description')
plt.ylabel('Frequency [%]')
ax = sns.countplot(x= 'Location_Description', data = dfFinal, order =
             dfFinal['Location_Description'].value_counts().iloc[:10].index)
ax.set_title('LOCATION OF CRIME OCCURRENCE IN CHICAGO', fontsize=20)
'''for p in ax.patches:
 ax.annotate('%{:.1f}'.format(p.get_height()), (p.get_x()+0.1, p.get_height()+50))'''

for p in ax.patches:
 ax.annotate('{:.1f}'.format(p.get_height()), (p.get_x()+0.25, p.get_height()+0.01))
plt.xticks(rotation=90)

plt.show()


# Therefore, going by the chat below, it reveals that most of the crimes were commited on the street.

# To discover the number of crimes in various locations
```



In [84]:
```
# Univariate
# Distinguishing between the crimes that occur at home with those on the street (the top two locations for crimes to oc
# Another observation that can be made here is that during the later part of the day, the number of crimes on the
# street increases while the number of crimes at home decrease.
# So, maybe the crime is being carried out by the same kind of people for home and street ?


plt.figure(figsize = (10, 7))
ax=sns.countplot(x= 'Location_Description', data = dfFinal,
```

```
                order = dfFinal['Location_Description'].value_counts().iloc[:10].index)
plt.ylabel('Number of Crimes')
#plt.xticks(rotation=70)
ax.set_title('LOCATION OF CRIME OCCURRENCE IN CHICAGO', fontsize=20)
'''for p in ax.patches:
 ax.annotate('%{:.1f}'.format(p.get_height()), (p.get_x()+0.1, p.get_height()+50))'''

for p in ax.patches:
 ax.annotate('{:.1f}'.format(p.get_height()), (p.get_x()+0.25, p.get_height()+0.01))
plt.xticks(rotation=70)

plt.show()


# Street is the most frequent location for crime occurrance.
# Not all crimes are the same in all location.
# Some crimes types are more likely to occur than other types depending on the place and time
```

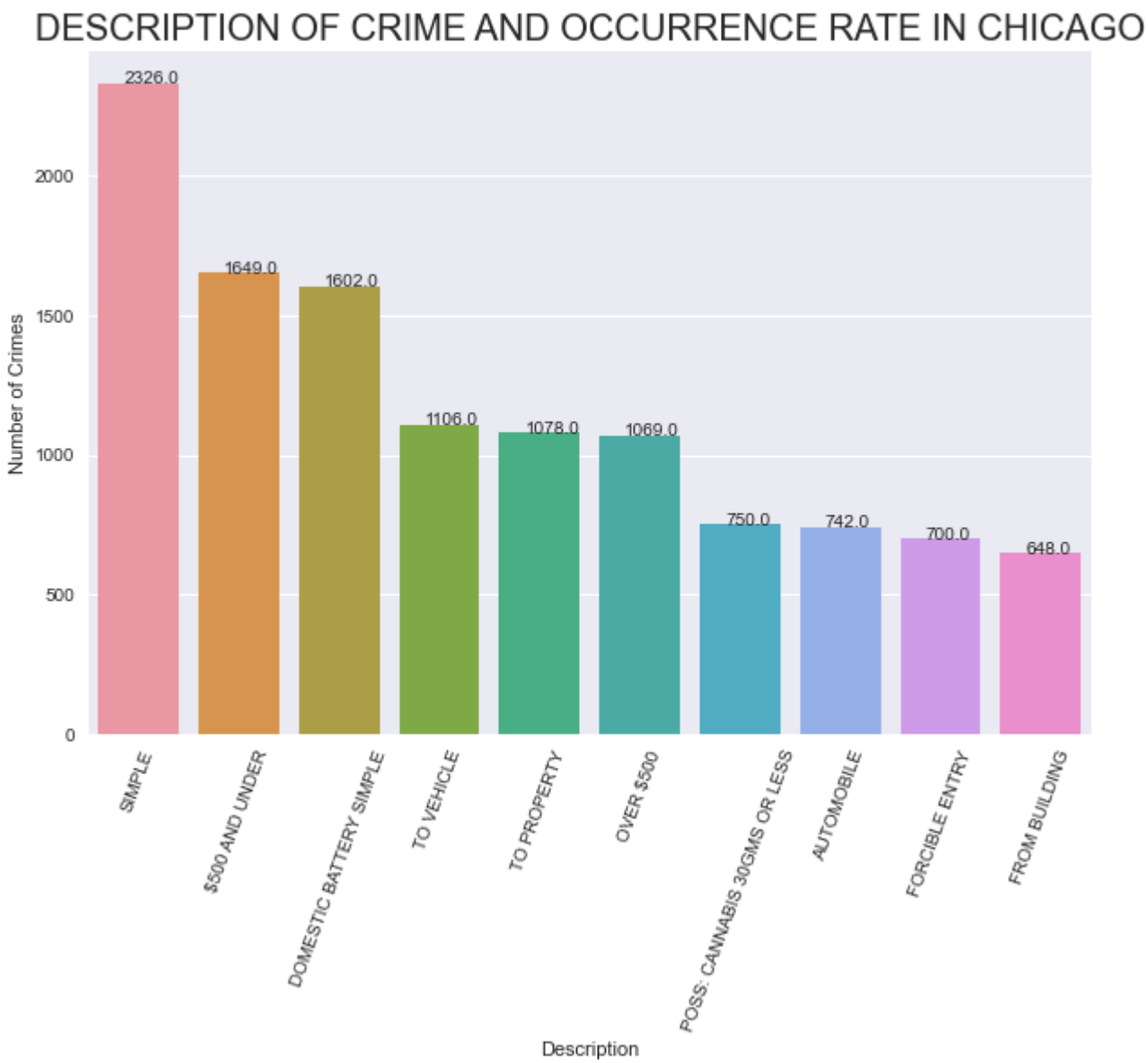


In [85]:
```
# WHAT IS THE DESCRIPTION OF THE CRIME RATE

# To know the DESCRIPTION (kind) of crimes (chronologically) that occurs in the Non-Domestic Crime as this
# could be helpful in determining what type of crime is common in a location (PREDOMINANT CRIME TYPE IN A LOCATION).
```

In [86]:
```
# Univariate
```

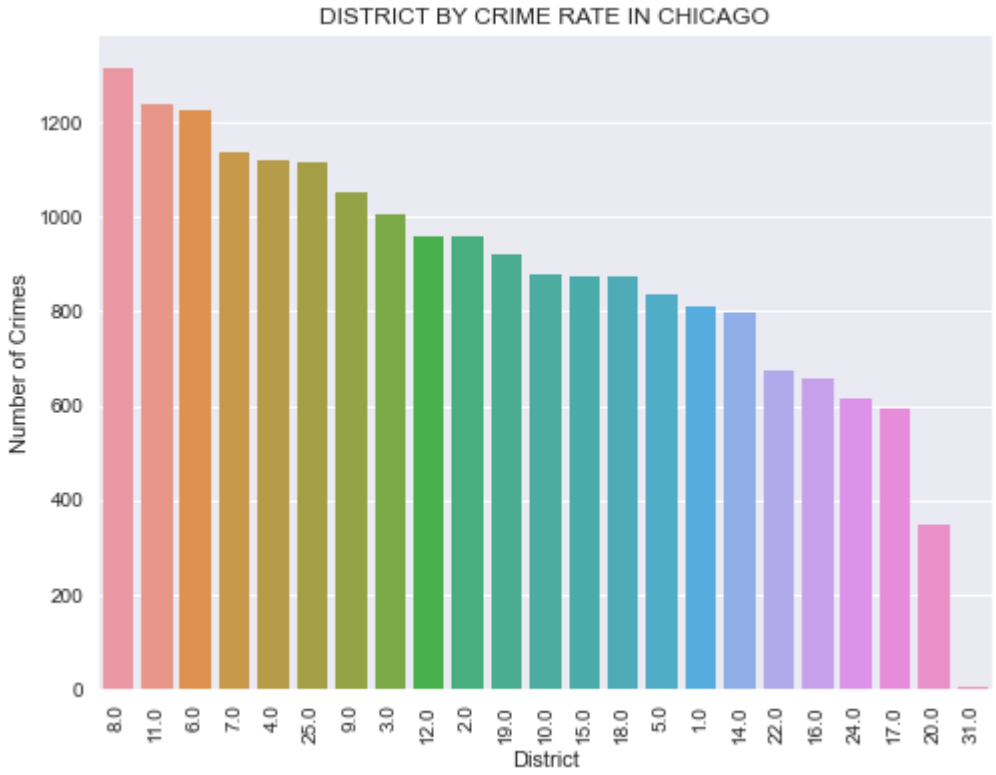
```
print(dfFinal.Description.value_counts())

plt.figure(figsize = (10, 7))
ax=sns.countplot(x= 'Description', data = dfFinal, order = dfFinal['Description'].value_counts().iloc[:10].index)
plt.ylabel('Number of Crimes')
#plt.xticks(rotation=70)
ax.set_title('DESCRIPTION OF CRIME AND OCCURRENCE RATE IN CHICAGO', fontsize=20)
'''for p in ax.patches:
 ax.annotate('%{:.1f}'.format(p.get_height()), (p.get_x()+0.1, p.get_height()+50))'''

for p in ax.patches:
 ax.annotate('{:.1f}'.format(p.get_height()), (p.get_x()+0.25, p.get_height()+0.01))
plt.xticks(rotation=70)

plt.show()
```

```
SIMPLE                                      2326
$500 AND UNDER                              1649
DOMESTIC BATTERY SIMPLE                     1602
TO VEHICLE                                  1106
TO PROPERTY                                 1078
                                            ...
ATTEMPT ARMED - KNIFE / CUTTING INSTRUMENT     1
POSSESS - PCP                                  1
POSS: LOOK-ALIKE DRUGS                         1
POSS: HEROIN(BLACK TAR)                        1
VIOLATION OF SUMMARY CLOSURE                   1
Name: Description, Length: 300, dtype: int64
```



In [87]:
```
# UNIVARIATE ANALYSIS FOR DISTRICT

# This reveals the Police district where the incident occurred and the frequency and
# This will help see how crimes differ between different places (DISTRICT) at different times

# To find the DISTRICT with/where most of these crimes usually occur chronologically as
# this could be helpful in determining where to station the police force.
```

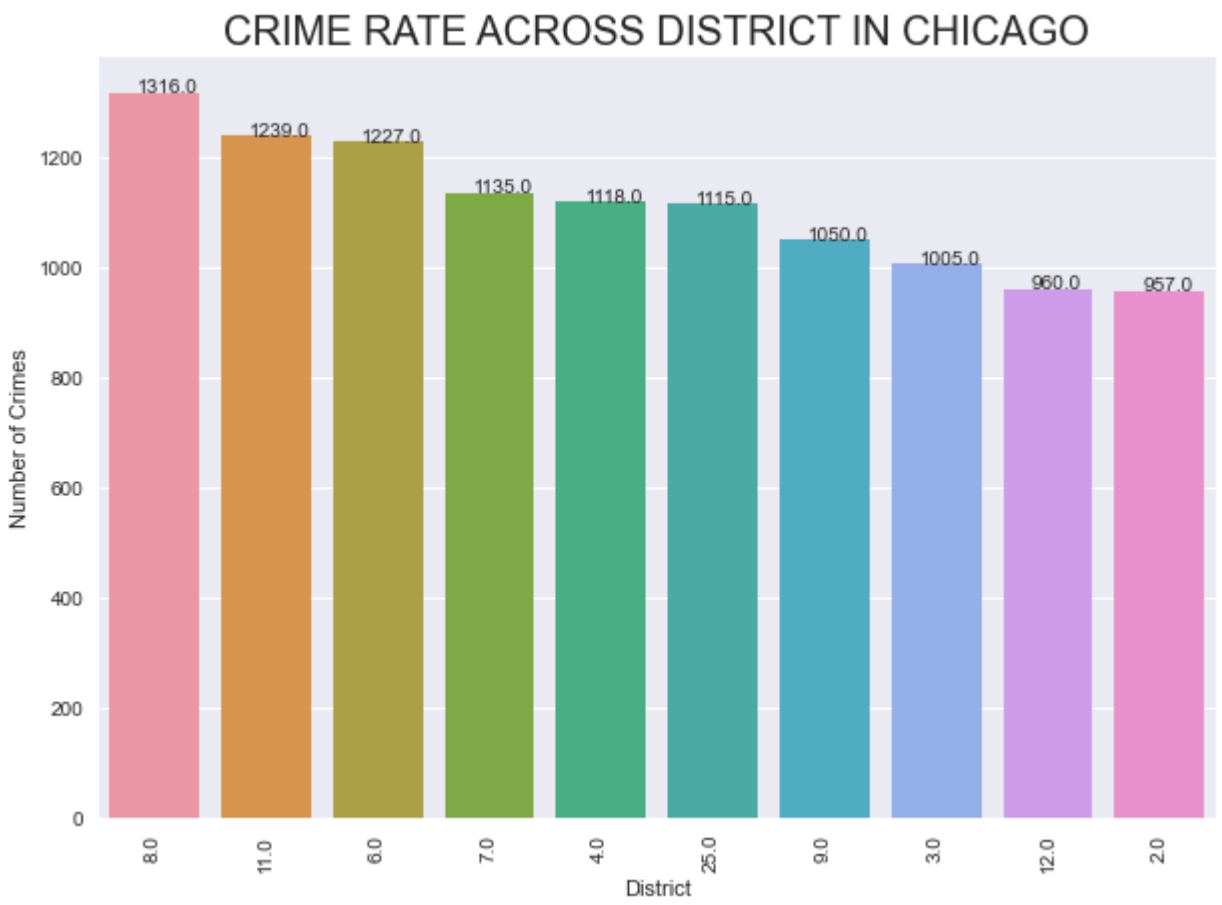
```
plt.figure(figsize = (8, 6))
sns.countplot(x= 'District', data = dfFinal, order = dfFinal['District'].value_counts().iloc[:].index)
plt.ylabel('Number of Crimes')

plt.xticks(rotation=90)
plt.title('DISTRICT BY CRIME RATE IN CHICAGO')
plt.show()

print(dfFinal.District.value_counts())

# Not all crimes are the same in all district. Some crimes types are more likely to occur
# than other types depending on the place and time.

# Where will be the safest place for me in Chicago ?
# District 31 has the least crime rate in chicago while district 8, 11, and 6 has the most crime.
```



```
8.0     1316
11.0    1239
6.0     1227
7.0     1135
4.0     1118
25.0    1115
9.0     1050
3.0     1005
12.0     960
2.0      957
19.0     919
10.0     876
15.0     875
18.0     875
5.0      836
1.0      808
14.0     796
22.0     675
16.0     658
24.0     615
17.0     592
20.0     348
31.0       5
Name: District, dtype: int64
```

In [88]:

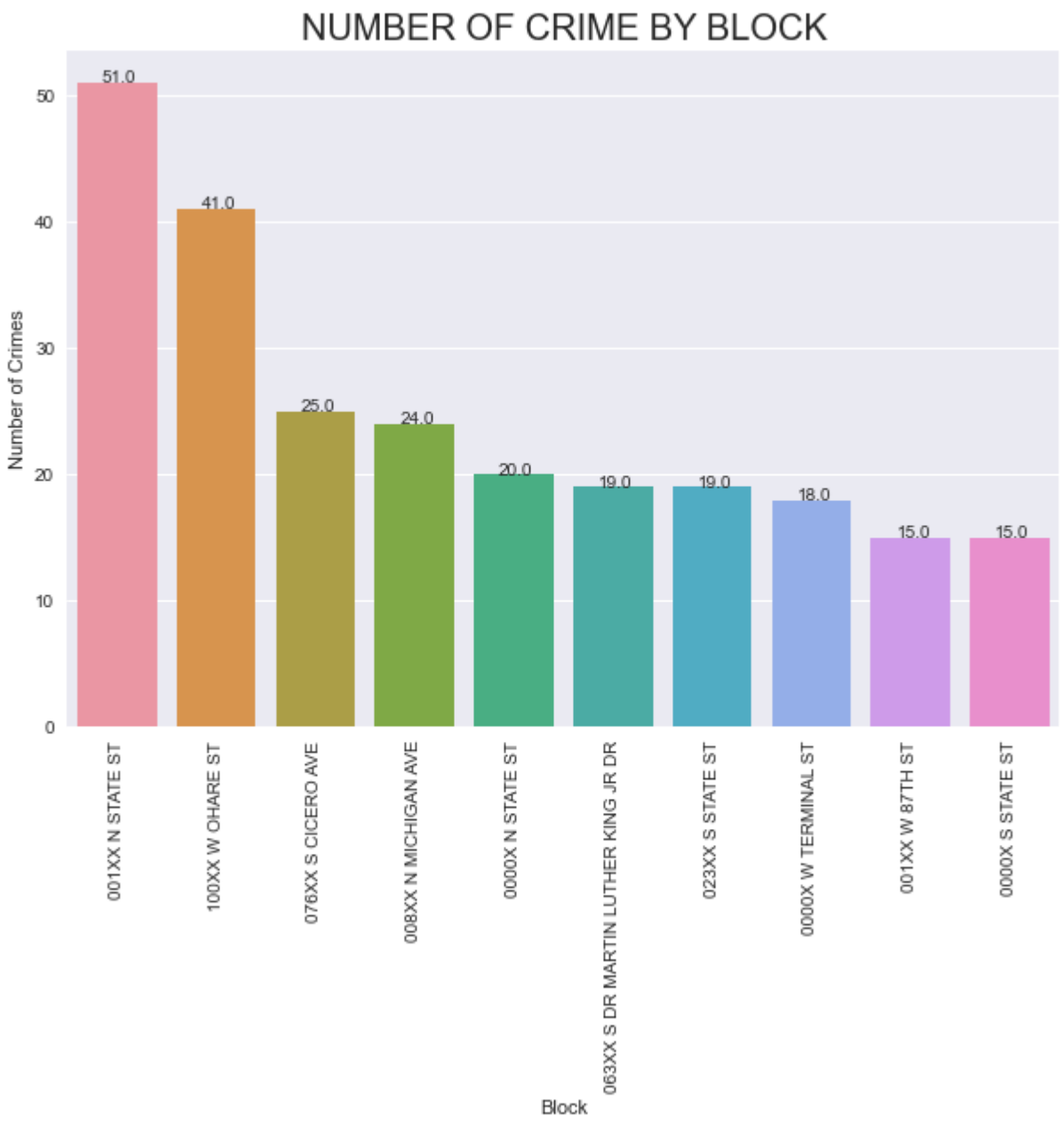
```
# Top 10 districts with most crimes

plt.figure(figsize = (10, 7))
ax=sns.countplot(x= 'District', data = dfFinal, order = dfFinal['District'].value_counts().iloc[:10].index)
plt.ylabel('Number of Crimes')
#plt.xticks(rotation=70)
ax.set_title('CRIME RATE ACROSS DISTRICT IN CHICAGO', fontsize=20)
'''for p in ax.patches:
 ax.annotate('%{:.1f}'.format(p.get_height()), (p.get_x()+0.1, p.get_height()+50))'''

for p in ax.patches:
 ax.annotate('{:.1f}'.format(p.get_height()), (p.get_x()+0.25, p.get_height()+0.01))
plt.xticks(rotation=90)

plt.show()
```



In [89]:
```
# UNIVARIATE ANALYSIS FOR BLOCK

# This shows count of crime incidents by block group and the address at which the incident occurred and the frequency.


print(dfFinal.Block.value_counts())

plt.figure(figsize = (10, 7))
ax=sns.countplot(x= 'Block', data = dfFinal, order = dfFinal['Block'].value_counts().iloc[:10].index)
plt.ylabel('Number of Crimes')
#plt.xticks(rotation=70)
ax.set_title('NUMBER OF CRIME BY BLOCK', fontsize=20)
'''for p in ax.patches:
 ax.annotate('%{:.1f}'.format(p.get_height()), (p.get_x()+0.1, p.get_height()+50))'''

for p in ax.patches:
```

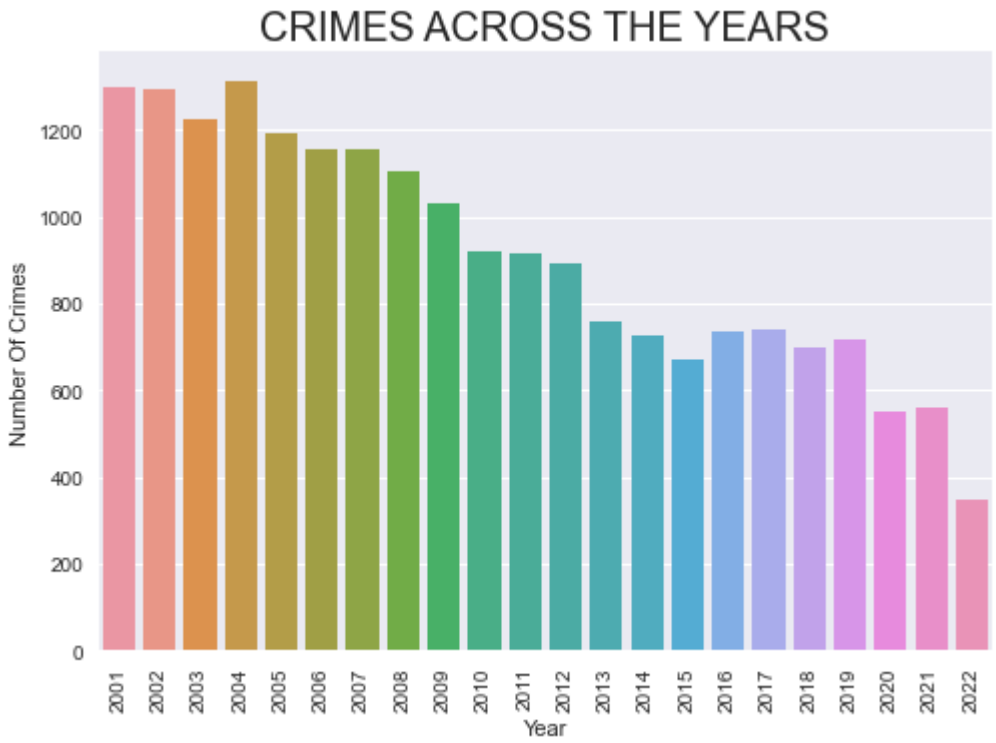
```
 ax.annotate('{:.1f}'.format(p.get_height()), (p.get_x()+0.25, p.get_height()+0.01))
plt.xticks(rotation=90)

plt.show()
```

```
001XX N STATE ST        51
100XX W OHARE ST        41
076XX S CICERO AVE      25
008XX N MICHIGAN AVE    24
0000X N STATE ST        20
                        ..
071XX S ELLIS AVE        1
066XX S ARTESIAN AV      1
041XX N MONITOR AVE      1
014XX W 49TH ST          1
004XX W OAK ST           1
Name: Block, Length: 11900, dtype: int64
```



In [90]:
```
#              NOTE

# The next thing I am going to look at is if there is a difference in the number of crimes during
# specific days of the week, Periods of the day, months, Year and Hour. Moreso,
# Are there more crimes during weekdays or weekends?
```

In [91]:
```
# ******

sns.countplot(x = "Year", data = dfFinal)
plt.ylabel('Number Of Crimes')

plt.title("CRIMES ACROSS THE YEARS", fontsize=20)

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



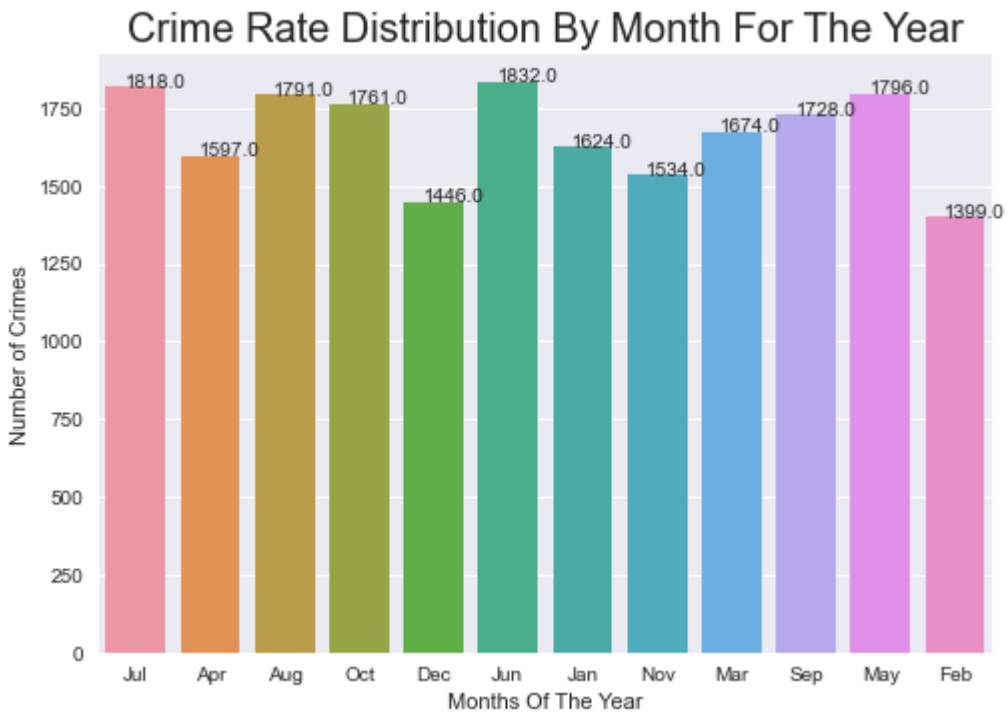
To determine the pattern of crimes by analyzing it based on: monthly, weekly, daily, periodically and hourly.

In [92]:
```
#  looking at crimes per month and see if certain months show more crimes than others.

print(dfFinal.Month.value_counts())

plt.title('CRIMES RATE ACROSS THE MONTH')
plt.xlabel('Number of Axles')
plt.ylabel('Frequency [%]')
```

```
ax = sns.countplot(x="Month", data=dfFinal)
ax.set_title('Crime Rate Distribution By Month For The Year', fontsize=20)
'''for p in ax.patches:
 ax.annotate('%{:.1f}'.format(p.get_height()), (p.get_x()+0.1, p.get_height()+50))'''
for p in ax.patches:
 ax.annotate('{:.1f}'.format(p.get_height()), (p.get_x()+0.25, p.get_height()+0.01))

plt.ylabel('Number of Crimes')
plt.xlabel('Months Of The Year')
```

```
Jun    1832
Jul    1818
May    1796
Aug    1791
Oct    1761
Sep    1728
Mar    1674
Jan    1624
Apr    1597
Nov    1534
Dec    1446
Feb    1399
Name: Month, dtype: int64
```

Out[92]:
```
Text(0.5, 0, 'Months Of The Year')
```

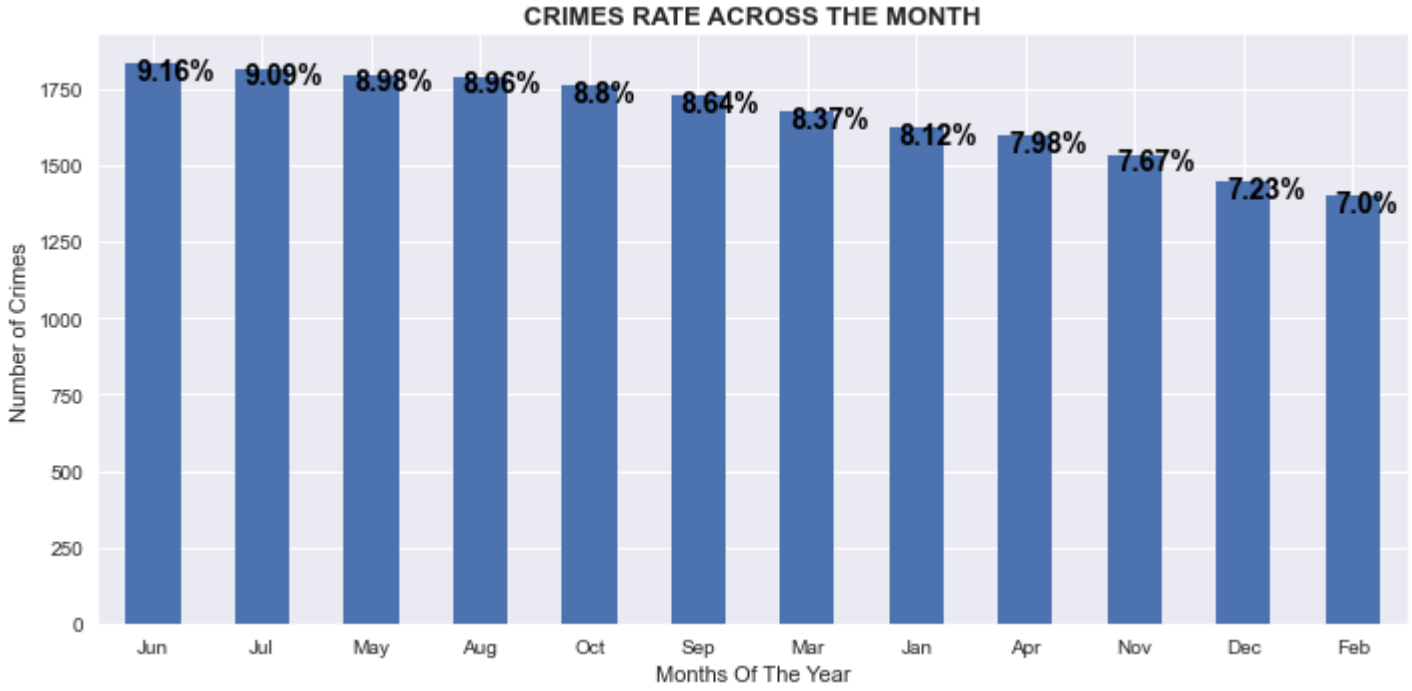


In [93]:
```
dfFinal.Month.unique()
```

Out[93]:
```
array(['Jul', 'Apr', 'Aug', 'Oct', 'Dec', 'Jun', 'Jan', 'Nov', 'Mar',
       'Sep', 'May', 'Feb'], dtype=object)
```

In [94]:
```
#  looking at %crimes per month and see if certain months show more crimes than others.


# print(dfFinal.Month.value_counts())
# dfFinal['Month'].value_counts().plot(kind='bar').set_title('CRIMES RATE ACROSS THE MONTH') #Simple plot
fig, ax = plt.subplots(figsize=(10,5))
name = ['Jun', 'Jul', 'May', 'Aug', 'Oct', 'Sep', 'Mar', 'Jan', 'Apr', 'Nov', 'Dec', 'Feb']
ax = dfFinal.Month.value_counts().plot(kind='bar')
ax.set_title("CRIMES RATE ACROSS THE MONTH", fontsize = 13, weight = 'bold')
ax.set_xticklabels (name, rotation = 0)
plt.ylabel('Number of Crimes')
plt.xlabel('Months Of The Year')

# To calculate the percentage
totals = []
for i in ax.patches:
 totals.append(i.get_height())
```

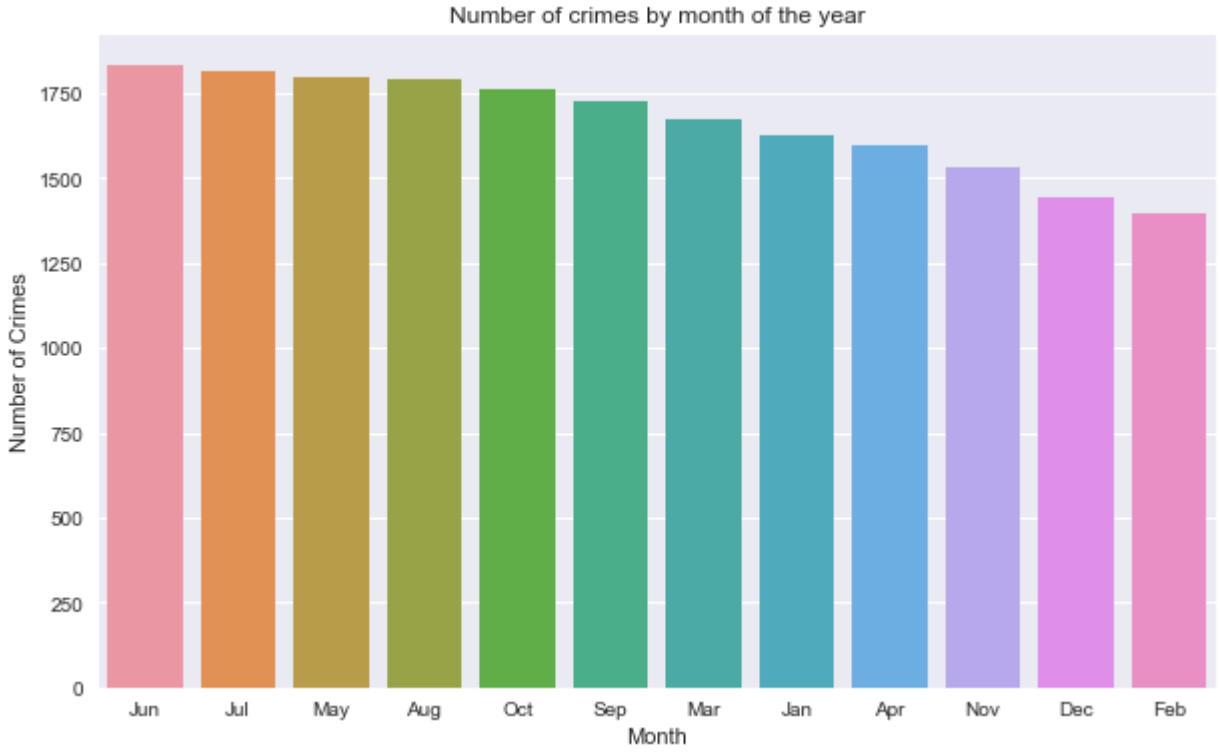
```
total = sum(totals)
for i in ax.patches:
 ax.text(i.get_x()+.09, i.get_height()-50, \
 str(round((i.get_height()/total)*100, 2))+'%', fontsize=14,
 color='black', weight = 'bold')

plt.tight_layout()
```



In [95]:
```
#           NOTE
#  There are seasonal peaks in (Summer period) June, July, May and August, and
#  seasonal drops in (Winter season) November, December, February.
```

In [96]:
```
# To show the distribution of total number of crimes by months in descending order.

# month =['Jan', 'Feb', 'Mar', 'Apr', 'May', 'Jun', 'Jul', 'Aug', 'Sep', 'Oct', 'Nov', 'Dec']

plt.figure(figsize = (10, 6))
sns.countplot(x= 'Month', data = dfFinal,
              order = dfFinal['Month'].value_counts().iloc[:12].index)
plt.title('Number of crimes by month of the year')
plt.ylabel('Number of Crimes')
```

```
plt.show()
```



In [97]:
```
#      OBSERVATION / INFERENCE
# Based on the monthly crime rates, we can see that crimes dip/decline in February and spikes in June (May to August)
# The above diagram shows the distribution of total number of crimes by months.
# “July” and “June” recorded the most number of crimes reported.
# Crimes rate seems to be at peak in summer time/period

# This result might be of help or could be useful for tourists/ travel information,
# and for the police department, this data tells them which months to be on high alert.
```

In [98]:
```
### Higher trend of crime during the summer months

### QUESTION
#### Are there attractive recreational centre in those Districts in Chicago that contributes to crime rate increase?
#### Could it be Summer break and vacations or holidays?
#### An attraction to tourist in summer which affect increase crime in summer months? Or
#### A similar trend I see across the analysis shows that crime rates spike from June – August  during the summer perio
```

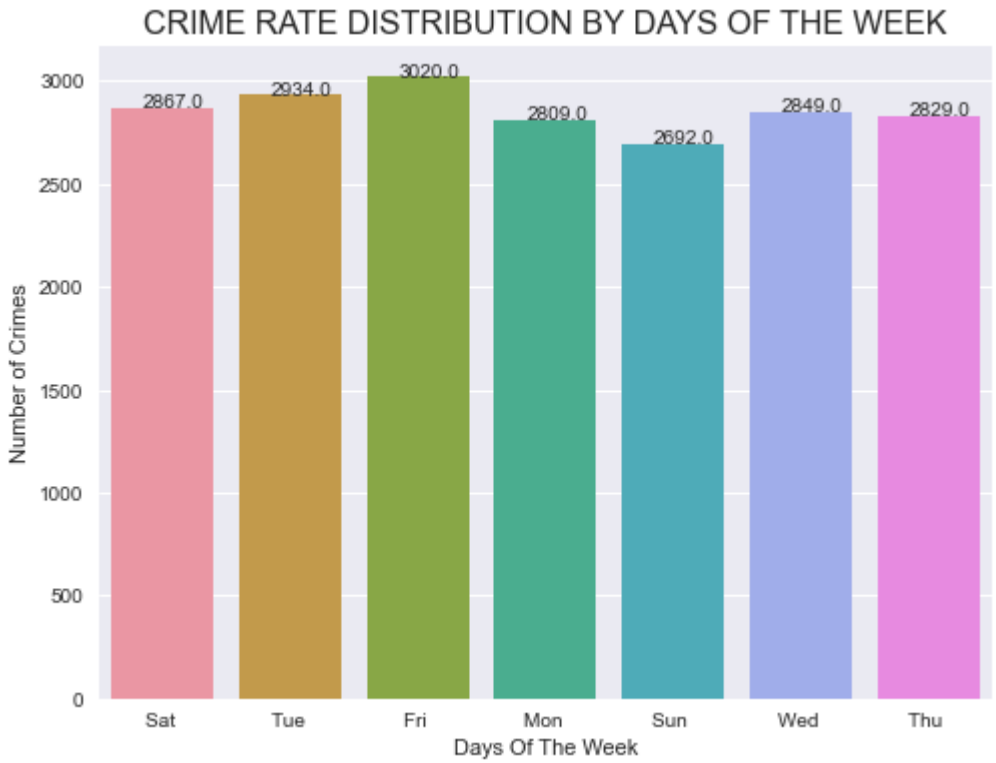
```
#### Could it be that increased temperatures of the increased daylight hours can lengthen
#### the amount of time people spend away from their homes, increasing the number of people in public and
#### the amount of time that homes are left empty, creating a greater possibility of criminal activities.
#### Let see if there are patterns for days of the week also to further enquip the PD with insights to address the crim
#### Are there is a difference in the number of crimes during specific days of the week?
```

In [99]:

```
# To analysis CRIME for days of the week

# Are there difference in the number of crimes during specific days of the week.

plt.figure(figsize = (8, 6))
plt.title('CRIME RATE ACROSS DAYS OF THE WEEK')
plt.xlabel('Number of Axles')
plt.ylabel('Frequency [%]')
ax = sns.countplot(x="Day", data=dfFinal)
ax.set_title('CRIME RATE DISTRIBUTION BY DAYS OF THE WEEK', fontsize=16)

'''for p in ax.patches:
 ax.annotate('%{:.1f}'.format(p.get_height()), (p.get_x()+0.1, p.get_height()+50))'''
for p in ax.patches:
 ax.annotate('{:.1f}'.format(p.get_height()), (p.get_x()+0.25, p.get_height()+0.01))
plt.ylabel('Number of Crimes')
plt.xlabel('Days Of The Week')

dfFinal["Day"].value_counts()
```

Out[99]:

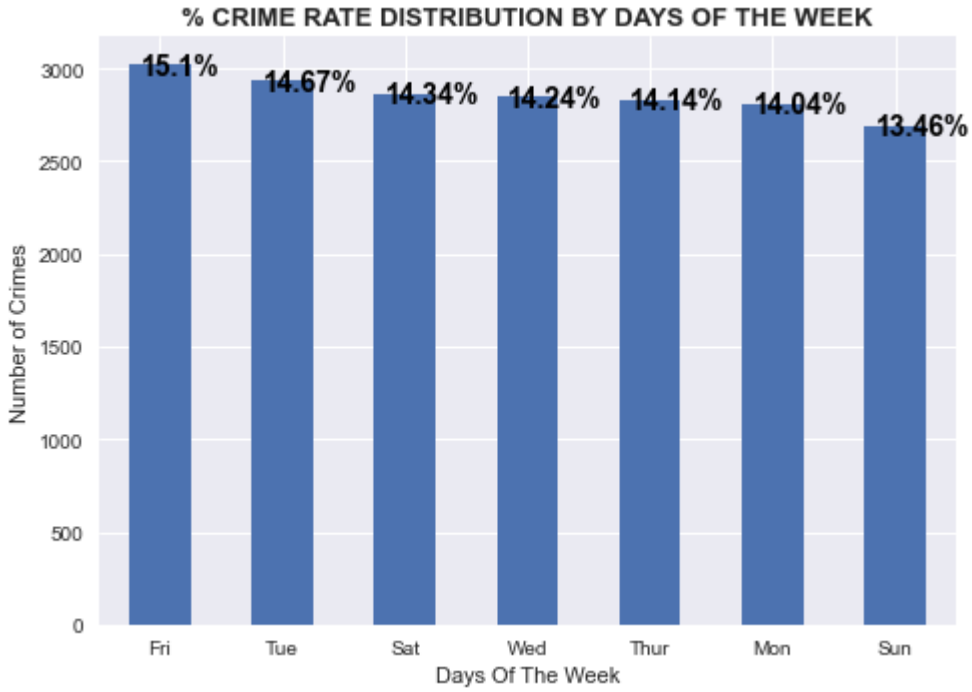
```
Fri     3020
Tue     2934
Sat     2867
Wed     2849
Thu     2829
Mon     2809
Sun     2692
Name: Day, dtype: int64
```



In [100…

```
# To analysis %Crime rate for days of the week chronologically

#print(dfFinal.Day.value_counts())

# dfFinal['Day'].value_counts().plot(kind='bar').set_title('% CRIME RATE ACROSS THE DAYS OF THE WEEK') #Simple plot
fig, ax = plt.subplots(figsize=(7,5))
name = ["Fri", "Tue", "Sat", "Wed", "Thur", "Mon", "Sun"]
ax = dfFinal.Day.value_counts().plot(kind='bar')
ax.set_title("% CRIME RATE DISTRIBUTION BY DAYS OF THE WEEK", fontsize = 13, weight = 'bold')
ax.set_xticklabels (name, rotation = 0)
# To calculate the percentage
totals = []
for i in ax.patches:
 totals.append(i.get_height())
total = sum(totals)
for i in ax.patches:
 ax.text(i.get_x()+.09, i.get_height()-50, \
 str(round((i.get_height()/total)*100, 2))+'%', fontsize=14,
 color='black', weight = 'bold')
```

```
plt.ylabel('Number of Crimes')
plt.xlabel('Days Of The Week')

plt.tight_layout()
```



In [101…

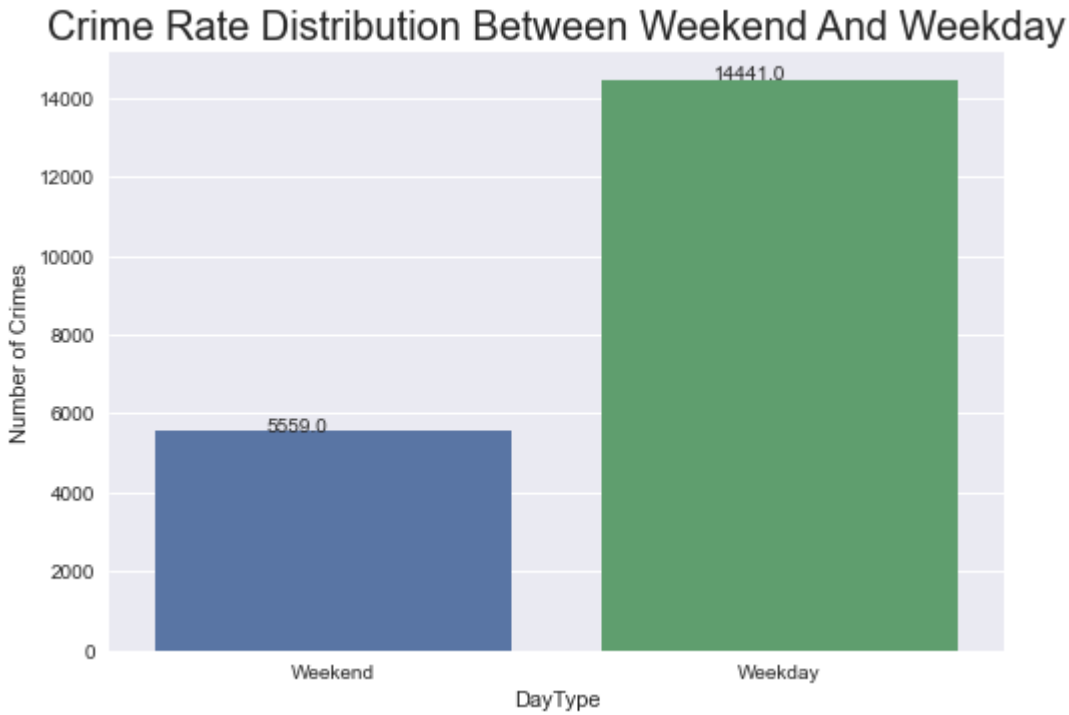
```
# The result shows that number of crimes on Friday has the highest rate though,
# slightly higher than the rest of the week: except for Sunday with the least crime rate.
```

In [102…

```
# ANALYSING CRIME RATE BETWEEN WEEKENDS AND WEEKDAYS

#  Are there more crimes during weekdays or weekends?

plt.title('CRIME RATE BETWEEN WEEKENDS AND WEEKDAYS')
plt.xlabel('Number of Axles')
plt.ylabel('Frequency [%]')
ax = sns.countplot(x="DayType", data=dfFinal)
ax.set_title('Crime Rate Distribution Between Weekend And Weekday', fontsize=20)
'''for p in ax.patches:
 ax.annotate('%{:.1f}'.format(p.get_height()), (p.get_x()+0.1, p.get_height()+50))'''
for p in ax.patches:
 ax.annotate('{:.1f}'.format(p.get_height()), (p.get_x()+0.25, p.get_height()+0.01))
plt.ylabel('Number of Crimes')
```

```
dfFinal["DayType"].value_counts()
```

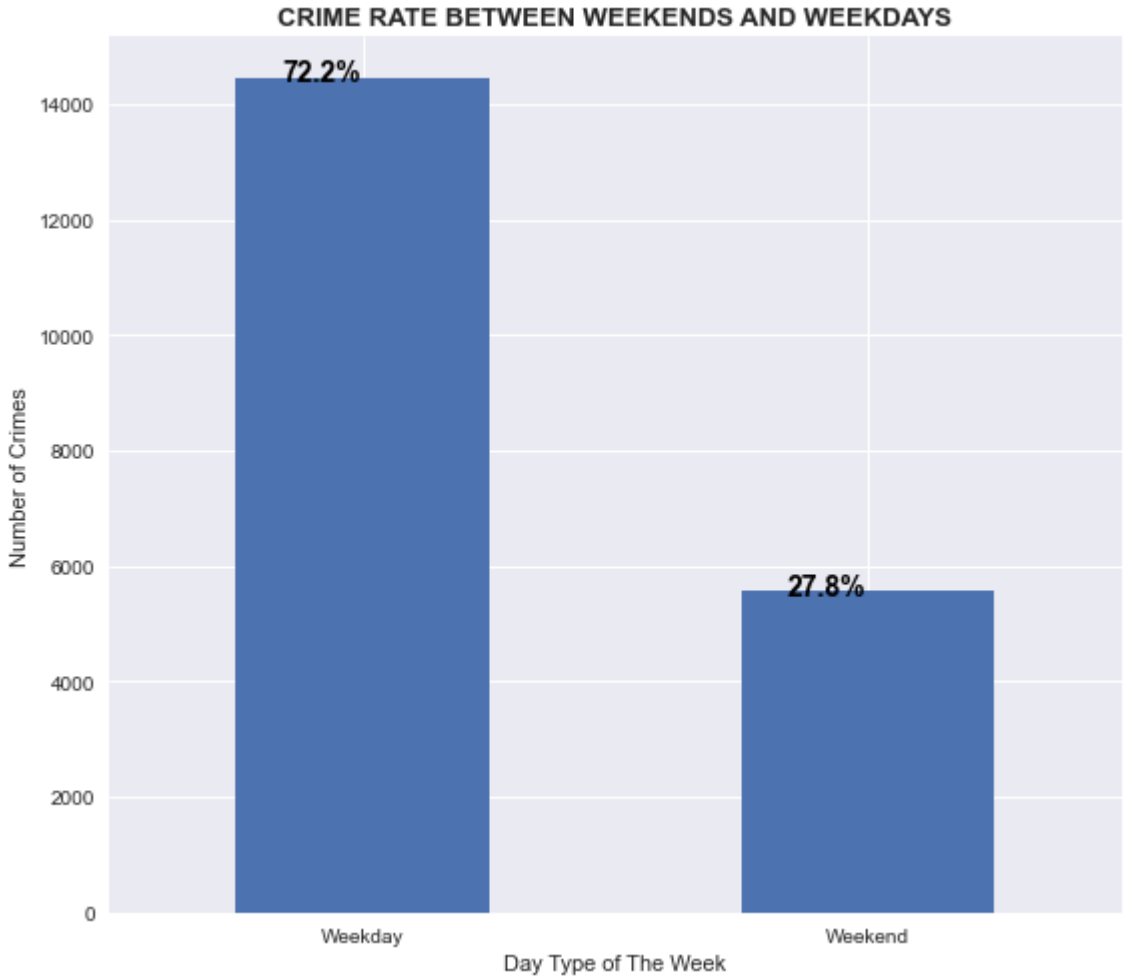
Out[102]:
```
Weekday    14441
Weekend     5559
Name: DayType, dtype: int64
```



In [103…
```
print(dfFinal.DayType.value_counts())

# dfFinal['DayType'].value_counts().plot(kind='bar').set_title('CRIMES ACROSS THE DAY TYPE') #Simple plot
fig, ax = plt.subplots(figsize=(8,7))
name = ["Weekday", "Weekend"]
ax = dfFinal.DayType.value_counts().plot(kind='bar')
ax.set_title("CRIME RATE BETWEEN WEEKENDS AND WEEKDAYS", fontsize = 13, weight = 'bold')
ax.set_xticklabels (name, rotation = 0)
plt.ylabel('Number of Crimes')
plt.xlabel('Day Type of The Week')

# To calculate the percentage
totals = []
for i in ax.patches:
 totals.append(i.get_height())
total = sum(totals)
```

```
for i in ax.patches:
 ax.text(i.get_x()+.09, i.get_height()-50, \
 str(round((i.get_height()/total)*100, 2))+'%', fontsize=14,
 color='black', weight = 'bold')

plt.tight_layout()
```

```
Weekday    14441
Weekend     5559
Name: DayType, dtype: int64
```

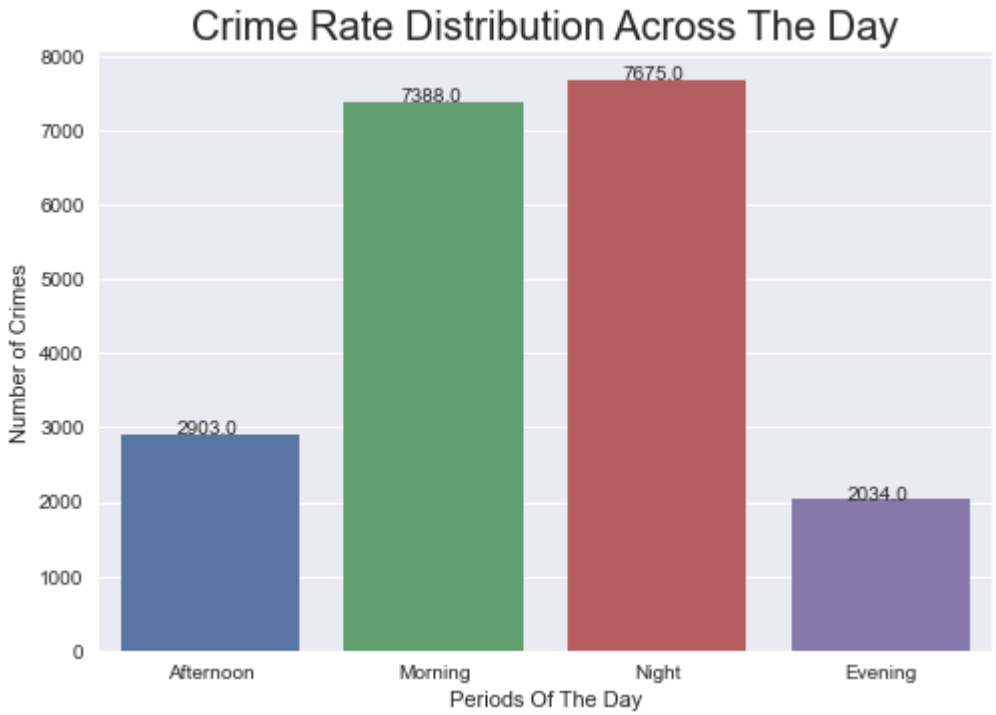


In [104…

```
#                    NOTE
# The above depicts that there are more crimes committed on weekdays than the weekends.
# Could this be that people tend to stay at home more on weekends and mostly go to work more on weekdays?
```

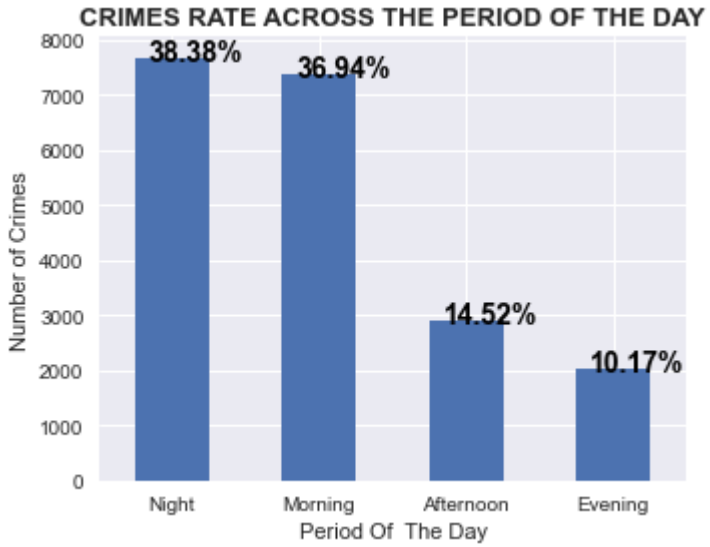
In [105…
```
# ANALYSING CRIME RATE ACROSS THE PERIOD OF THE DAY

# This tells dwellers/travellers/citizens and policemen alike when to be more careful and at alert throughout the day.

plt.title('CRIME RATE ACROSS THE PERIOD OF THE DAY')
plt.xlabel('Number of Axles')
plt.ylabel('Frequency [%]')
ax = sns.countplot(x="Period", data=dfFinal)
ax.set_title('Crime Rate Distribution Across The Day', fontsize=20)
'''for p in ax.patches:
 ax.annotate('%{:.1f}'.format(p.get_height()), (p.get_x()+0.1, p.get_height()+50))'''
for p in ax.patches:
 ax.annotate('{:.1f}'.format(p.get_height()), (p.get_x()+0.25, p.get_height()+0.01))
plt.ylabel('Number of Crimes')
plt.xlabel('Periods Of The Day')

dfFinal["Period"].value_counts()
```

Out[105]:
```
Night        7675
Morning      7388
Afternoon    2903
Evening      2034
Name: Period, dtype: int64
```



In [106…

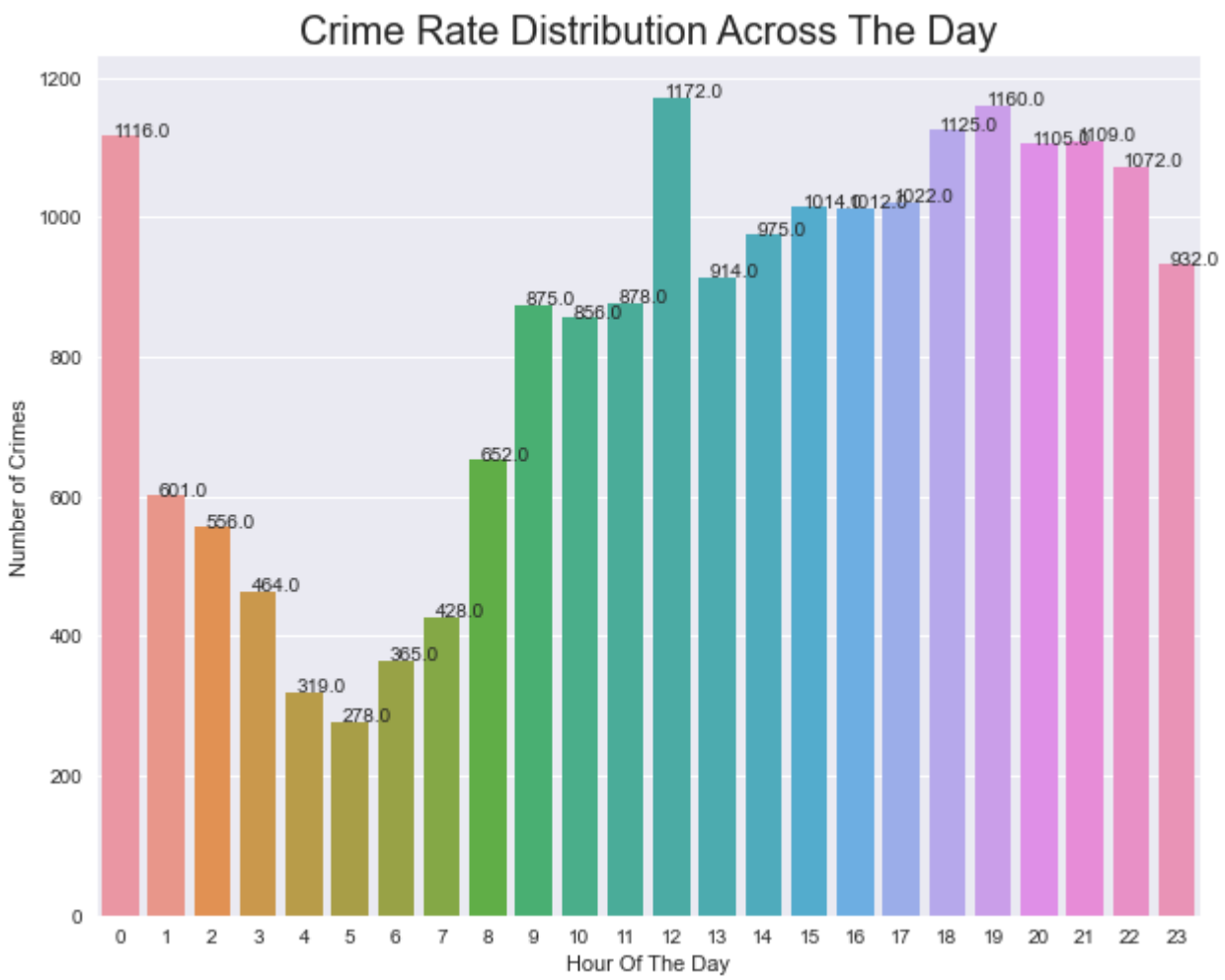
```
# This result tells dwellers/citizens and policemen alike when to be more careful throughout the day.

# This shows that crimes occur the most during the Night and Morning period.
# So, Police may be increasing the number of patrolling officers at that period of the day would be helpful.



# dfFinal['Period'].value_counts().plot(kind='bar').set_title('CRIMES ACROSS THE PERIOD') #Simple plot
fig, ax = plt.subplots(figsize=(5,4))
name = ["Night", "Morning", "Afternoon", "Evening"]
ax = dfFinal.Period.value_counts().plot(kind='bar')
ax.set_title("CRIMES RATE ACROSS THE PERIOD OF THE DAY", fontsize = 13, weight = 'bold')
ax.set_xticklabels (name, rotation = 0)
plt.ylabel('Number of Crimes')
plt.xlabel('Period Of  The Day')

# To calculate the percentage
totals = []
for i in ax.patches:
 totals.append(i.get_height())
total = sum(totals)
for i in ax.patches:
 ax.text(i.get_x()+.09, i.get_height()-50, \
 str(round((i.get_height()/total)*100, 2))+'%', fontsize=14,
 color='black', weight = 'bold')

plt.tight_layout()
```



In [107…

```
# Analysing crime distribution by the hours of the day

plt.figure(figsize = (10, 8))
plt.title('CRIME RATE ACROSS THE HOUR OF THE DAY')
plt.xlabel('Number of Axles')
plt.ylabel('Frequency [%]')
ax = sns.countplot(x="Hour", data=dfFinal)
ax.set_title('Crime Rate Distribution Across The Day', fontsize=20)
'''for p in ax.patches:
 ax.annotate('%{:.1f}'.format(p.get_height()), (p.get_x()+0.1, p.get_height()+50))'''
for p in ax.patches:
 ax.annotate('{:.1f}'.format(p.get_height()), (p.get_x()+0.25, p.get_height()+0.01))
plt.ylabel('Number of Crimes')
plt.xlabel('Hour Of The Day')

dfFinal["Hour"].value_counts()

# This tells dwellers/citizens and policemen alike when to be more careful throughout the day.

# This graph shows that the crime activities are more common at the peak hours 12 mid-day while least at 5:00am
```

Out[107]:
```
12    1172
19    1160
18    1125
0     1116
21    1109
20    1105
22    1072
17    1022
15    1014
16    1012
14     975
23     932
13     914
11     878
9      875
10     856
8      652
1      601
2      556
3      464
7      428
6      365
4      319
5      278
Name: Hour, dtype: int64
```



In [108…
```
# To analyse the distribution of the Arrest feature (column), the count of Arrest versus Non-Arrest made.
# using the following:

dfFinal["Arrest"].value_counts()

### Shows that the rate of Non-Arrest cases has the highest rate and chances of crime rate to increase is high.
```

Out[108]:
```
False    14610
True      5390
Name: Arrest, dtype: int64
```

In [109…
```
# TO KNOW THE PERCENTAGE OF ARREST AND NON-ARREST MADE FOR THE CRIMES ACROSS LOCATION AND DISTRICT
```

```
# df['Arrest'].head()
arrestRate = dfFinal["Arrest"].value_counts()
false = arrestRate[0]
true = arrestRate[1]

arrest = pd.DataFrame({'Status':['Not Arrested','Arrested'],'Value':list(arrestRate)})
print("Percentage of no arrests of all reported crimes :",false/(false+true)*100,'!')
```

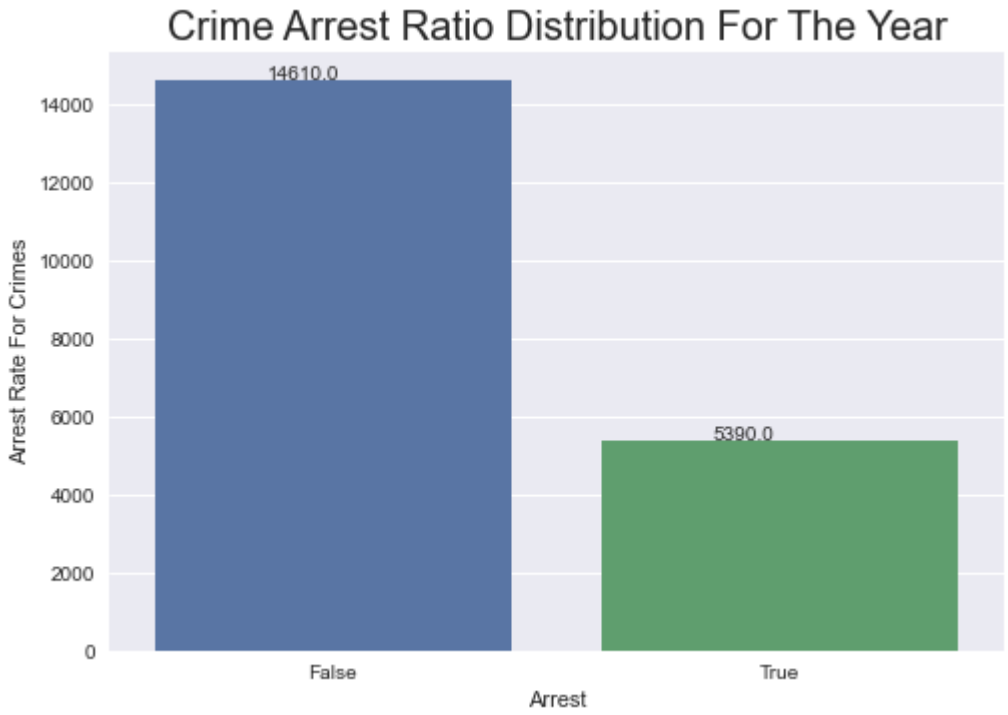
```
Percentage of no arrests of all reported crimes : 73.05 !
```

In [110…

```
# To analyse the distribution of the arrest feature, the count of NO ARREST versus ARREST for crimes in Chicago.

### Result shows that most of crime offenders get away with their crime as "No arrest rate" is about 73.05%
# while rate of arrest for crime is about 26.95% as shown above.

# Therefore, there is high(est) chances of crimes incresing as police department fails to apprend offenders.


plt.title('CRIME ARREST RATE')
plt.xlabel('Number of Axles')
#plt.ylabel('Frequency [%]')

ax = sns.countplot(x="Arrest", data=dfFinal)
ax.set_title('Crime Arrest Ratio Distribution For The Year', fontsize=20)
plt.ylabel('Arrest Rate For Crimes')

'''for p in ax.patches:
 ax.annotate('%{:.1f}'.format(p.get_height()), (p.get_x()+0.1, p.get_height()+50))'''
for p in ax.patches:
 ax.annotate('{:.1f}'.format(p.get_height()), (p.get_x()+0.25, p.get_height()+0.01))

dfFinal["Arrest"].value_counts()
```

Out[110]:

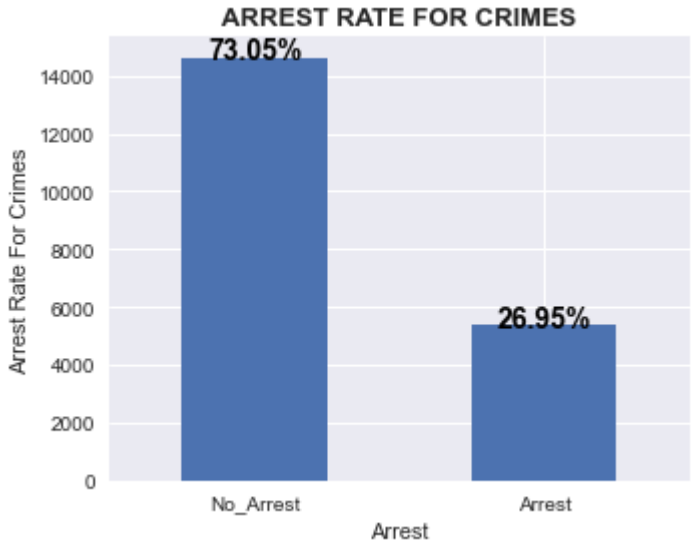
```
False    14610
True      5390
Name: Arrest, dtype: int64
```



In [111…

```
# % OF ARREST TO NON-ARREST RATE ACROSS CRIME TYPES, DISTRICTS AND LOCATIONS

# print(dfFinal.Arrest.value_counts())
# df['Arrest'].value_counts().plot(kind='bar').set_title('CRIME ARREST RATE') #Simple plot
fig, ax = plt.subplots(figsize=(5,4))
name = ["No_Arrest", "Arrest"]
ax = dfFinal.Arrest.value_counts().plot(kind='bar')
ax.set_title("ARREST RATE FOR CRIMES", fontsize = 13, weight = 'bold')
ax.set_xticklabels (name, rotation = 0)
plt.ylabel('Arrest Rate For Crimes')
plt.xlabel('Arrest')

# To calculate the percentage
totals = []
for i in ax.patches:
 totals.append(i.get_height())
total = sum(totals)
for i in ax.patches:
 ax.text(i.get_x()+.09, i.get_height()-50, \
 str(round((i.get_height()/total)*100, 2))+'%', fontsize=14,
 color='black', weight = 'bold')
```

```
plt.tight_layout()
```



In [112…
```
#         INFERRENCE

### Result shows that most of crime offenders get away with their crime as "No arrest rate" is about 73.05%
# while rate of arrest for crime is about 26.95% as shown above.

# Therefore, there is high(est) chances of crimes incresing as police department fails to apprend offenders.
```

BIVARIATE ANALYSIS

In [113…
```
# dfFinal.groupby(by=['District', 'Arrest']).size().sort_values(ascending=False)
```

In [114…
```
# Creating table for District and Arrest

pivot = dfFinal.pivot_table(index =['District'],
                        values =['Arrest'],
                        aggfunc ='sum')
print(pivot)
```

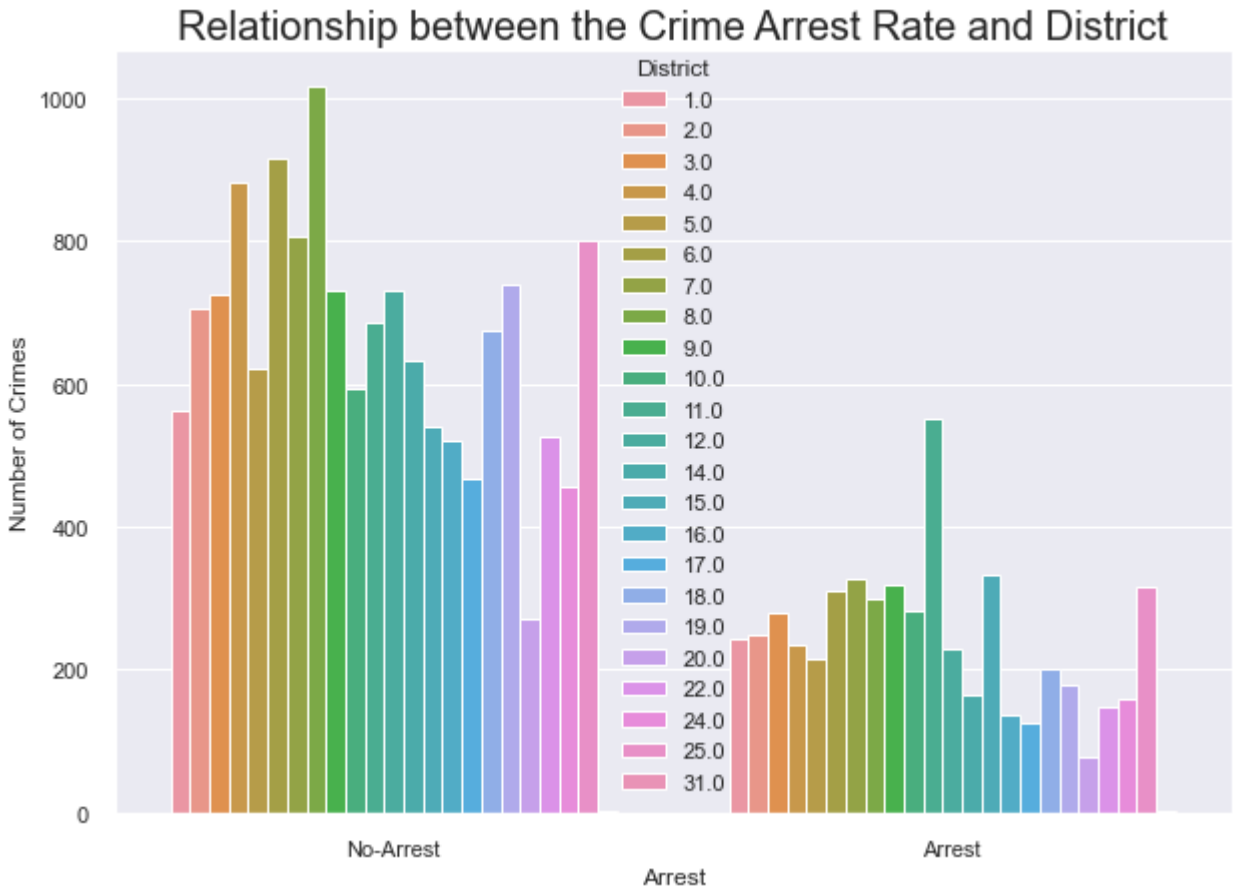
```
          Arrest
District
1.0          244
2.0          250
3.0          279
4.0          235
5.0          215
6.0          312
7.0          327
8.0          300
9.0          320
10.0         283
11.0         552
12.0         230
14.0         164
15.0         334
16.0         136
17.0         125
18.0         201
19.0         179
20.0          78
22.0         149
24.0         159
25.0         315
31.0           3
```

In [115…]
```
# Filtering out the Top 5 criminal districts
top_5_District = dfFinal['District'].value_counts().sort_values(ascending=False).head()
top_5_District

#plt.show()
```

Out[115]:
```
8.0     1316
11.0    1239
6.0     1227
7.0     1135
4.0     1118
Name: District, dtype: int64
```

In [116…]
```
# Looking into the proportion of arrest for each District.
# This time, I group the data by various District and arrest,
# Then retrieve the ratio between true and false arrest in each District.
```

```
sns.set(rc={'figure.figsize':(10,7)})
Crime = sns.countplot(x='Arrest', hue='District', data=dfFinal)
Crime.set_xticklabels(['No-Arrest','Arrest'])
Crime.set_title('Relationship between the Crime Arrest Rate and District', fontsize=20)
plt.ylabel('Number of Crimes')

plt.show()
```



In [117…

```
# Creating table for Primary_Type and Arrest

pivot = dfFinal.pivot_table(index =['Primary_Type'],
                            values =['District'],
                            aggfunc ='sum')
print(pivot)
```

Primary_Type	District
ARSON	360.0
ASSAULT	13068.0
BATTERY	40601.0
BURGLARY	11892.0
CONCEALED CARRY LICENSE VIOLATION	68.0
CRIM SEXUAL ASSAULT	769.0
CRIMINAL DAMAGE	26152.0
CRIMINAL SEXUAL ASSAULT	169.0
CRIMINAL TRESPASS	6157.0
DECEPTIVE PRACTICE	9901.0
GAMBLING	375.0
HOMICIDE	256.0
INTERFERENCE WITH PUBLIC OFFICER	370.0
INTIMIDATION	276.0
KIDNAPPING	142.0
LIQUOR LAW VIOLATION	566.0
MOTOR VEHICLE THEFT	10506.0
NARCOTICS	22296.0
NON - CRIMINAL	18.0
OBSCENITY	30.0
OFFENSE INVOLVING CHILDREN	1448.0
OTHER NARCOTIC VIOLATION	2.0
OTHER OFFENSE	14040.0
PROSTITUTION	2246.0
PUBLIC INDECENCY	18.0
PUBLIC PEACE VIOLATION	1400.0
RITUALISM	14.0
ROBBERY	8452.0
SEX OFFENSE	1186.0
STALKING	128.0
THEFT	50866.0
WEAPONS VIOLATION	2483.0

In [118…

```
# Pivot table justapoxing crime type and District


dfFinal.groupby(by=['Primary_Type', 'District']).size().sort_values(ascending=False)
```

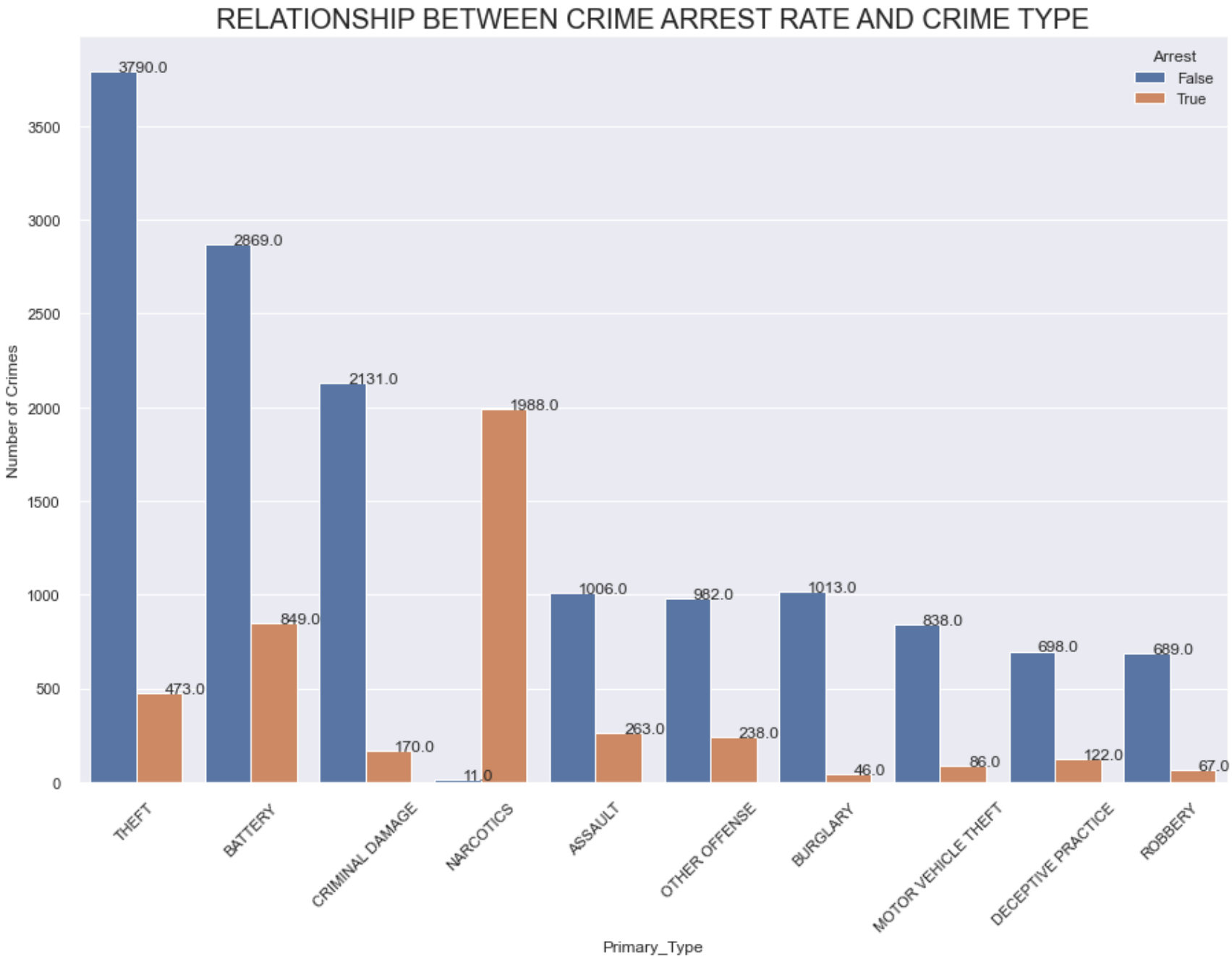
Out[118]:
```
Primary_Type       District
THEFT              18.0        387
                   1.0         376
NARCOTICS          11.0        343
THEFT              19.0        326
BATTERY            7.0         275
                               ...
NON - CRIMINAL     15.0          0
                   14.0          0
                   12.0          0
                   11.0          0
WEAPONS VIOLATION  31.0          0
Length: 736, dtype: int64
```

In [119…
```
# Pivot table  justapoxing crime type and arrest

dfFinal.groupby(by=['Primary_Type', 'Arrest']).size().sort_values(ascending=False)
```

Out[119]:
```
Primary_Type              Arrest
THEFT                     False     3790
BATTERY                   False     2869
CRIMINAL DAMAGE           False     2131
NARCOTICS                 True      1988
BURGLARY                  False     1013
                                    ...
LIQUOR LAW VIOLATION      False        0
PUBLIC INDECENCY          False        0
OBSCENITY                 False        0
NON - CRIMINAL            True         0
OTHER NARCOTIC VIOLATION  False        0
Length: 64, dtype: int64
```

In [120…
```
# CRIME TYPE AND ARREST RATE

# Arrest for Narcotics is on the side followed by Battery which could mean that
# the Police Ddepartment focuses more on these crimes (Narcotics and Battery) than others
# It also shows that PD has a hold on narcotics crimes and they seem
# not to apprehend the Theft crimes despite being the highest rate
# Therefore, theft crimes increases since they seem to get away as shown in by arrest rate.


sns.set(rc={'figure.figsize':(15,10)})
ax = sns.countplot(x='Primary_Type', hue='Arrest', data=dfFinal,
                   order = dfFinal['Primary_Type'].value_counts().iloc[:10].index)
plt.ylabel('Number of Crimes')
```

```
ax.set_title('RELATIONSHIP BETWEEN CRIME ARREST RATE AND CRIME TYPE', fontsize=20)
'''for p in ax.patches:
 ax.annotate('%{:.1f}'.format(p.get_height()), (p.get_x()+0.1, p.get_height()+50))'''

for p in ax.patches:
 ax.annotate('{:.1f}'.format(p.get_height()), (p.get_x()+0.25, p.get_height()+0.01))
plt.xticks(rotation=45)

plt.show()


#Crime.set_xticklabels(["Primary_Type"])
#Crime.set_title('Relationship between the Crime Arrest Rate and Primary Type', fontsize=20)
#plt.ylabel('Number of Crimes')
#plt.xticks(rotation=90)

#plt.show()
```



INFERENCE AND OBSERVATION ON CRIME TYPE AND ARREST RATE

Arrest for Narcotics is on the side followed by Battery which could mean that the Police Ddepartment focuses more on these crimes (Narcotics and Battery) than others. It also shows that PD has a hold on narcotics crimes and they seem not to apprehend the Theft crimes despite being the highest rate.

Therefore, theft crimes increases since they seem to get away as shown in by arrest rate.

In [121…
```
# EXAMINING THE RELATIONASIP BETWEEN CRIME TYPE VS LOCATION DESCRIPTION BY DISTRICT VS HOUR VS ARREST VS CRIME RATE

dfFinal.groupby(by=['Primary_Type', 'Location_Description', 'District', 'Hour', 'Arrest']).size().sort_values(ascending
```

Out[121]:
```
Primary_Type       Location_Description                 District  Hour  Arrest
THEFT              RESTAURANT                           1.0       12    False     18
NARCOTICS          SIDEWALK                             11.0      11    True      16
                                                                 18    True      13
THEFT              STREET                               18.0      0     False     13
NARCOTICS          SIDEWALK                             11.0      20    True      13
                                                                                 ..
GAMBLING           POLICE FACILITY / VEHICLE PARKING LOT  11.0    23    False      0
                                                                       True       0
                                                        12.0      0     False      0
                                                                       True       0
WEAPONS VIOLATION  YARD                                 31.0      23    True       0
Length: 4451328, dtype: int64
```

In [122…
```
# This show all crime type with their arrest rate across all location.

pivot = dfFinal.pivot_table(index =['Primary_Type'],
                        values =['Arrest'],
                        aggfunc ='sum')
print(pivot)
```

Primary_Type	Arrest
ARSON	5
ASSAULT	263
BATTERY	849
BURGLARY	46
CONCEALED CARRY LICENSE VIOLATION	6
CRIM SEXUAL ASSAULT	14
CRIMINAL DAMAGE	170
CRIMINAL SEXUAL ASSAULT	1
CRIMINAL TRESPASS	432
DECEPTIVE PRACTICE	122
GAMBLING	33
HOMICIDE	6
INTERFERENCE WITH PUBLIC OFFICER	33
INTIMIDATION	2
KIDNAPPING	0
LIQUOR LAW VIOLATION	40
MOTOR VEHICLE THEFT	86
NARCOTICS	1988
NON - CRIMINAL	0
OBSCENITY	3
OFFENSE INVOLVING CHILDREN	31
OTHER NARCOTIC VIOLATION	1
OTHER OFFENSE	238
PROSTITUTION	179
PUBLIC INDECENCY	1
PUBLIC PEACE VIOLATION	85
RITUALISM	0
ROBBERY	67
SEX OFFENSE	28
STALKING	2
THEFT	473
WEAPONS VIOLATION	186

In [123…

```
# Crime type description with their number of occurrence

dfFinal.groupby(by=['Primary_Type', 'Description']).size().sort_values(ascending=False)
```

Out[123]:
```
Primary_Type       Description
THEFT              $500 AND UNDER                    1649
BATTERY            DOMESTIC BATTERY SIMPLE           1602
                   SIMPLE                            1450
CRIMINAL DAMAGE    TO PROPERTY                       1078
                   TO VEHICLE                        1071
                                                     ...
GAMBLING           TO VEHICLE                           0
                   TRUCK, BUS, MOTOR HOME               0
                   UNAUTHORIZED VIDEOTAPING             0
                   UNIDENTIFIABLE RECORDING SOUND       0
WEAPONS VIOLATION  VIOLATION OF SUMMARY CLOSURE         0
Length: 9600, dtype: int64
```

In [124…
```
# Crime type, description with their number of occurrence versus arrest

pivot = dfFinal.pivot_table(index =['Primary_Type', 'Description'],
                        values =['Arrest'], aggfunc ='sum')
print (pivot)
```

```
                                                      Arrest
Primary_Type      Description
ARSON             $500 AND UNDER                           0
                  AGG CRIM SEX ABUSE FAM MEMBER            0
                  AGG CRIMINAL SEXUAL ABUSE                0
                  AGG PO HANDS ETC SERIOUS INJ             0
                  AGG PO HANDS NO/MIN INJURY               0
...                                                      ...
WEAPONS VIOLATION VEHICULAR HIJACKING                      0
                  VIOLATE ORDER OF PROTECTION              0
                  VIOLATION OF BAIL BOND - DOMESTIC VIOLENCE  0
                  VIOLATION OF CIVIL NO CONTACT ORDER      0
                  VIOLATION OF SUMMARY CLOSURE             0

[9600 rows x 1 columns]
```

In [125…
```
# Crime type description with their number of Arrest

pivot = dfFinal.pivot_table(index =['Description'],
                        values =['Arrest'],
                        aggfunc ='sum')
print(pivot)
```

```
                                                Arrest
Description
$500 AND UNDER                                     109
AGG CRIM SEX ABUSE FAM MEMBER                        2
AGG CRIMINAL SEXUAL ABUSE                            6
AGG PO HANDS ETC SERIOUS INJ                         1
AGG PO HANDS NO/MIN INJURY                          56
...                                                ...
VEHICULAR HIJACKING                                  4
VIOLATE ORDER OF PROTECTION                         37
VIOLATION OF BAIL BOND - DOMESTIC VIOLENCE           1
VIOLATION OF CIVIL NO CONTACT ORDER                  0
VIOLATION OF SUMMARY CLOSURE                         1

[300 rows x 1 columns]
```

EXAMINING CRIME TYPE BY SCENE LOCATION BY DISTRICT AND ARREST

1. This confirms Narcotics crime rate in District 11.0 and 15.0 are high as shown in the pivot table beow

2. It also affirms that arrest rate for Narcotics crimes are high in these locations while Theft arrest are low there.

3. Moreso, Narcotics crimes happen mostly by Sidewalk and Street.

In [126…

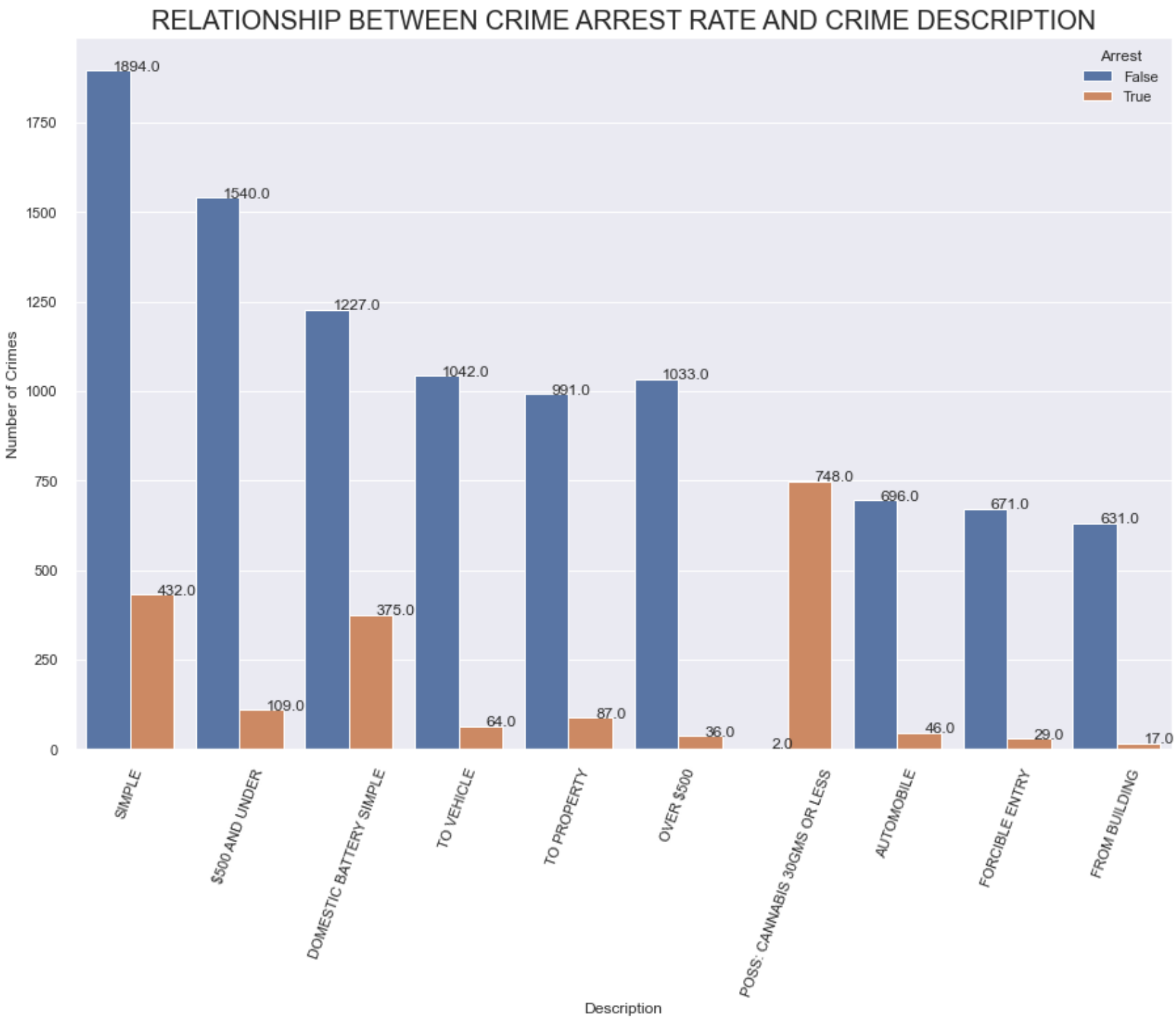
```
# EXAMINING CRIME TYPE BY SCENE LOCATION BY DISTRICT AND ARREST

dfFinal.groupby(by=['Primary_Type', 'Location_Description', 'District', 'Arrest']).size().sort_values(ascending=False)
```

Out[126]:

```
Primary_Type       Location_Description  District  Arrest
NARCOTICS          SIDEWALK              11.0      True      145
THEFT              STREET                12.0      False     102
NARCOTICS          SIDEWALK              15.0      True      100
                   STREET                11.0      True       94
THEFT              STREET                14.0      False      89
                                                            ...
GAMBLING           WAREHOUSE             24.0      False       0
                                                     True        0
                                         25.0      False       0
                                                     True        0
WEAPONS VIOLATION  YARD                  31.0      True        0
Length: 185472, dtype: int64
```

In [127…

```
# Crime type description with their number of Arrest

sns.set(rc={'figure.figsize':(15,10)})
ax = sns.countplot(x='Description', hue='Arrest', data=dfFinal,
                   order = dfFinal['Description'].value_counts().iloc[:10].index)
plt.ylabel('Number of Crimes')
ax.set_title('RELATIONSHIP BETWEEN CRIME ARREST RATE AND CRIME DESCRIPTION', fontsize=20)
'''for p in ax.patches:
 ax.annotate('%{:.1f}'.format(p.get_height()), (p.get_x()+0.1, p.get_height()+50))'''

for p in ax.patches:
 ax.annotate('{:.1f}'.format(p.get_height()), (p.get_x()+0.25, p.get_height()+0.01))
plt.xticks(rotation=70)

plt.show()
```


RELATIONSHIP BETWEEN CRIME ARREST RATE AND CRIME DESCRIPTION
Arrest
False
True
Number of Crimes
0
250
500
750
1000
1250
1500
1750
1894.0
432.0
1540.0
109.0
1227.0
375.0
1042.0
64.0
991.0
87.0
1033.0
36.0
2.0
748.0
696.0
46.0
671.0
29.0
631.0
17.0
SIMPLE
$500 AND UNDER
DOMESTIC BATTERY SIMPLE
TO VEHICLE
TO PROPERTY
OVER $500
POSS: CANNABIS 30GMS OR LESS
AUTOMOBILE
FORCIBLE ENTRY
FROM BUILDING
Description

In [128…
```
# This reveals most frequent (street, residence, apartment, sidewalk) and the least crime location too.

dfFinal.groupby(by=['Location_Description', 'Arrest']).size().sort_values(ascending=False)
```

Out[128]:
```
Location_Description            Arrest
STREET                          False     3701
RESIDENCE                       False     2821
APARTMENT                       False     1841
STREET                          True      1448
SIDEWALK                        True       988
                                          ...
SCHOOL - PUBLIC GROUNDS         True         0
AUTO                            True         0
ATM (AUTOMATIC TELLER MACHINE)  True         0
ATHLETIC CLUB                   True         0
YARD                            True         0
Length: 252, dtype: int64
```

In [129…
```
# Crime type Location_Description with their number of Arrest

pivot = dfFinal.pivot_table(index =['Location_Description'],
                        values =['Arrest'],
                        aggfunc ='sum')
print(pivot)
```

```
                                                Arrest
Location_Description
ABANDONED BUILDING                                  18
AIRCRAFT                                             0
AIRPORT BUILDING NON-TERMINAL - NON-SECURE AREA      0
AIRPORT EXTERIOR - NON-SECURE AREA                   0
AIRPORT PARKING LOT                                  0
...                                                ...
VEHICLE - OTHER RIDE SHARE SERVICE (E.G., UBER,...   1
VEHICLE NON-COMMERCIAL                             101
VEHICLE-COMMERCIAL                                   1
WAREHOUSE                                            8
YARD                                                 0

[126 rows x 1 columns]
```

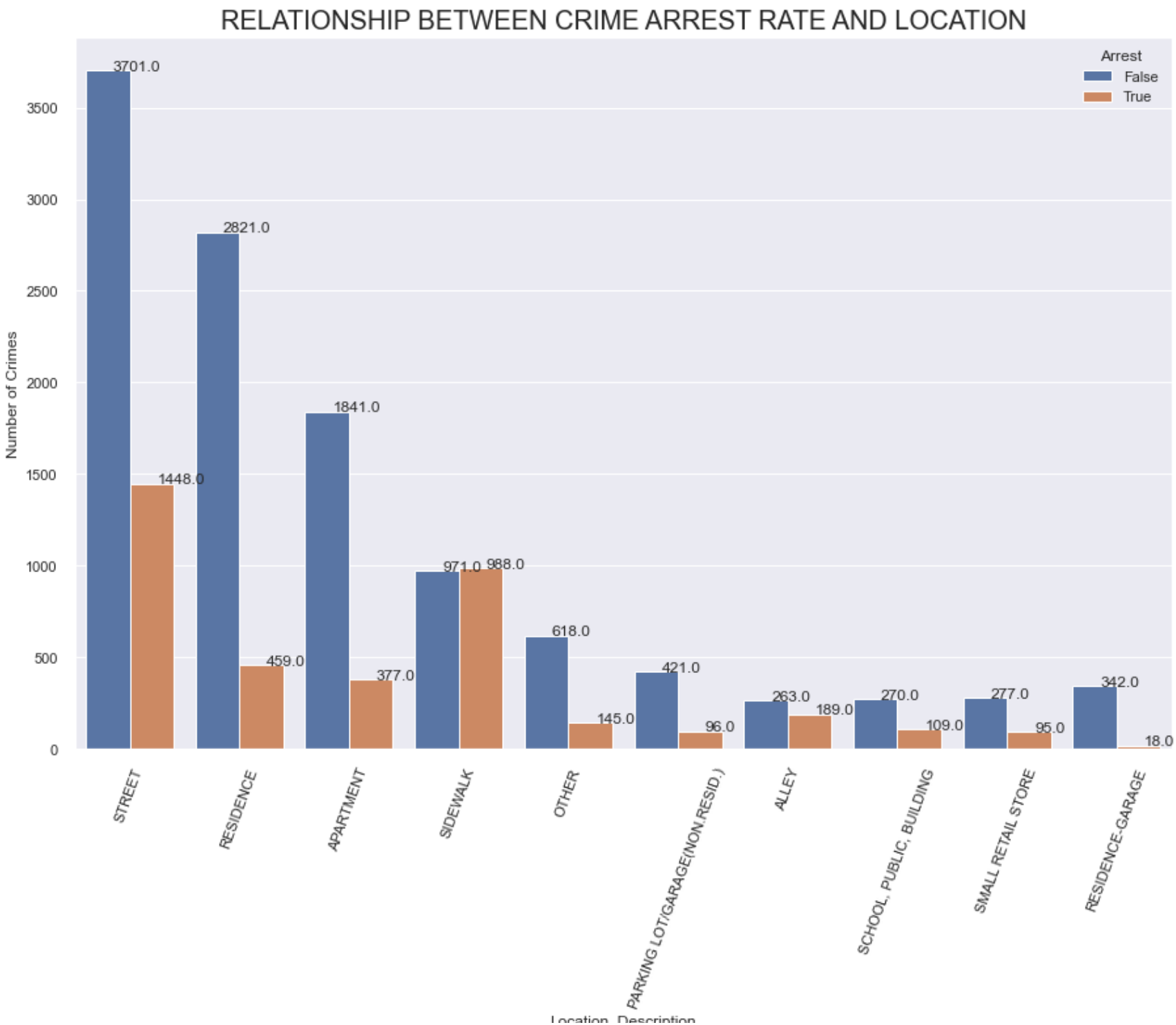
In [130…
```
# Crime type versus Location_Description with their number of Arrest

pivot = dfFinal.pivot_table(index =['Primary_Type', 'Location_Description'],
```

```
                       values =['Arrest'], aggfunc ='sum')
print (pivot)
```

```
                                                                Arrest
Primary_Type      Location_Description
ARSON             ABANDONED BUILDING                                0
                  AIRCRAFT                                          0
                  AIRPORT BUILDING NON-TERMINAL - NON-SECURE AREA    0
                  AIRPORT EXTERIOR - NON-SECURE AREA                0
                  AIRPORT PARKING LOT                               0
...                                                               ...
WEAPONS VIOLATION VEHICLE - OTHER RIDE SHARE SERVICE (E.G., UBER,...    0
                  VEHICLE NON-COMMERCIAL                           11
                  VEHICLE-COMMERCIAL                                0
                  WAREHOUSE                                         0
                  YARD                                              0

[4032 rows x 1 columns]
```

In [131…

```
# Relationship between the Crime Arrest Rate and Location

sns.set(rc={'figure.figsize':(15,10)})
ax = sns.countplot(x='Location_Description', hue='Arrest', data=dfFinal,
                       order = dfFinal['Location_Description'].value_counts().iloc[:10].index)
plt.ylabel('Number of Crimes')
ax.set_title('RELATIONSHIP BETWEEN CRIME ARREST RATE AND LOCATION', fontsize=20)
'''for p in ax.patches:
 ax.annotate('%{:.1f}'.format(p.get_height()), (p.get_x()+0.1, p.get_height()+50))'''

for p in ax.patches:
 ax.annotate('{:.1f}'.format(p.get_height()), (p.get_x()+0.25, p.get_height()+0.01))
plt.xticks(rotation=70)

plt.show()
```


RELATIONSHIP BETWEEN CRIME ARREST RATE AND LOCATION
Arrest
False
True
Number of Crimes
0
500
1000
1500
2000
2500
3000
3500
3701.0
1448.0
2821.0
459.0
1841.0
377.0
971.0
988.0
618.0
145.0
421.0
96.0
263.0
189.0
270.0
109.0
277.0
95.0
342.0
18.0
STREET
RESIDENCE
APARTMENT
SIDEWALK
OTHER
PARKING LOT/GARAGE(NON.RESID.)
ALLEY
SCHOOL, PUBLIC, BUILDING
SMALL RETAIL STORE
RESIDENCE-GARAGE
Location_Description

Location_Description

Interpretation and Observation on Relationship between the Crime Arrest Rate and Location

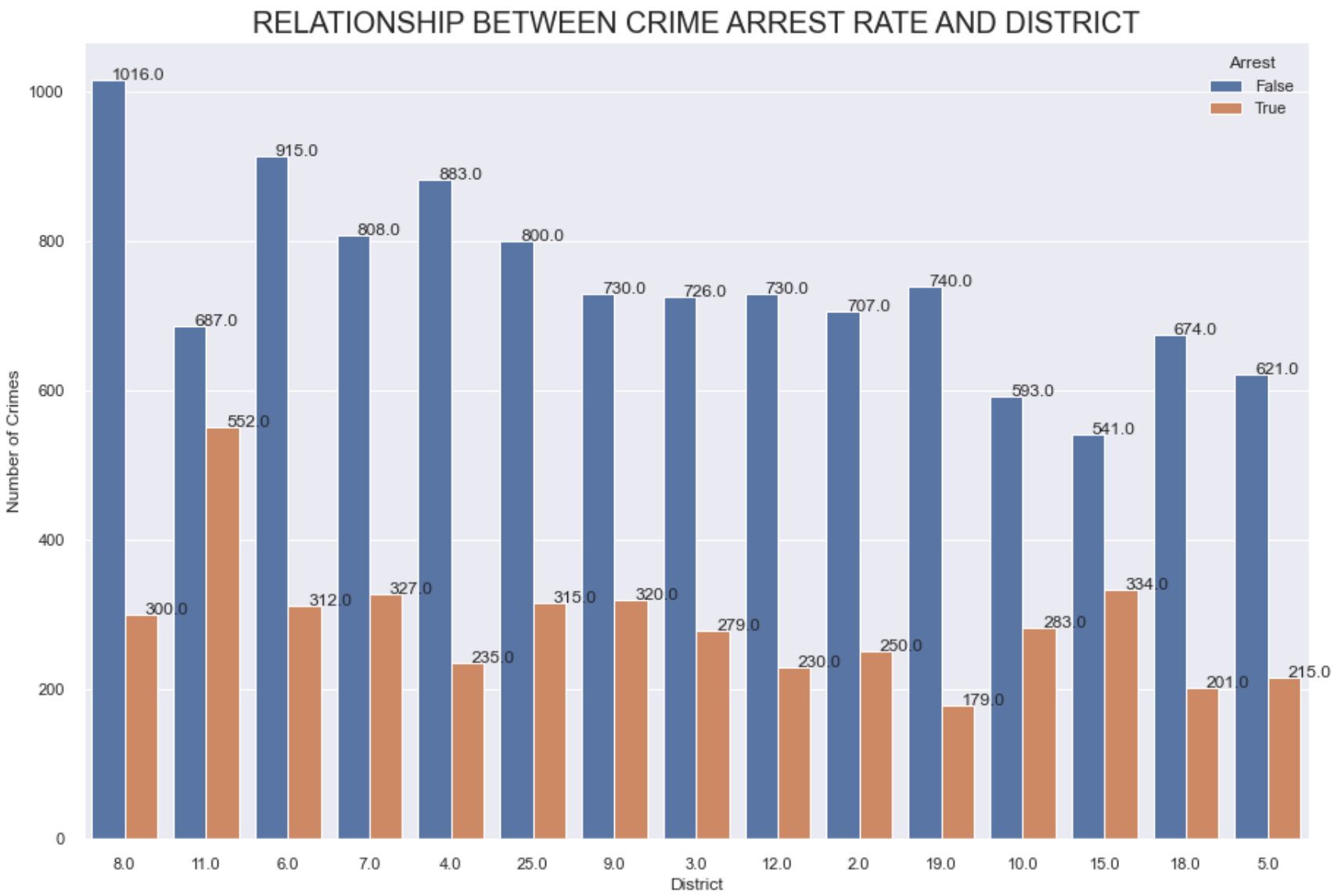
The sidewalk arrest rate equate sidewalk crimes, meaning the PD are doing a good job on sidewalk crimes as shown above.

Therefore, the arrest rate for sidewalk crime is at per with sidewalk crimes.

Knowing that Narcotics crimes are mostly on sidewalks which then justifies its high arrest rate.

In [132…

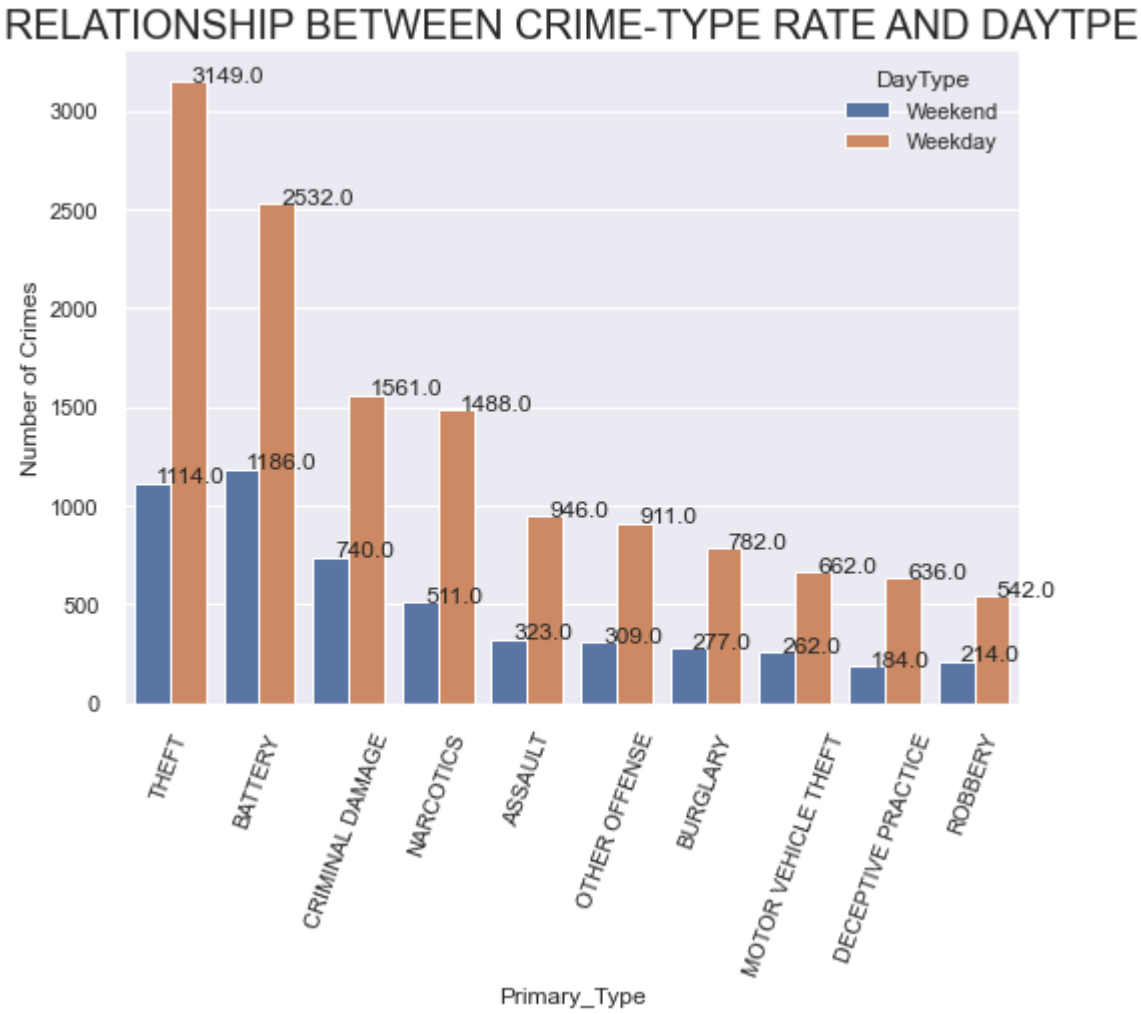
```
# DISTRICT VS ARREST

# District 11.0 has the highest arrest rate probably due to its predominant crime type
# which is Narcotics and mostly occur on sidewalks
# Hence, I can conclude that Narcotics are predominant crimes in district 11.0, also
# Narcotics are sidewalk crimes and the PD seem to have a hold on them.


sns.set(rc={'figure.figsize':(15,10)})
ax = sns.countplot(x='District', hue='Arrest', data=dfFinal,
                   order = dfFinal['District'].value_counts().iloc[:15].index)
plt.ylabel('Number of Crimes')
ax.set_title('RELATIONSHIP BETWEEN CRIME ARREST RATE AND DISTRICT', fontsize=20)
'''for p in ax.patches:
 ax.annotate('%{:.1f}'.format(p.get_height()), (p.get_x()+0.1, p.get_height()+50))'''

for p in ax.patches:
 ax.annotate('{:.1f}'.format(p.get_height()), (p.get_x()+0.25, p.get_height()+0.01))
#plt.xticks(rotation=70)

plt.show()
```



In [133…

```
# PRIMARY_TYPE VS DAYTYPE

# There seem to be high Battery crime cases on weekends than any other crimetype. While
# Theft is the predominant crime type for weekdays

sns.set(rc={'figure.figsize':(8,6)})
ax = sns.countplot(x='Primary_Type', hue='DayType', data=dfFinal,
                   order = dfFinal['Primary_Type'].value_counts().iloc[:10].index)
```

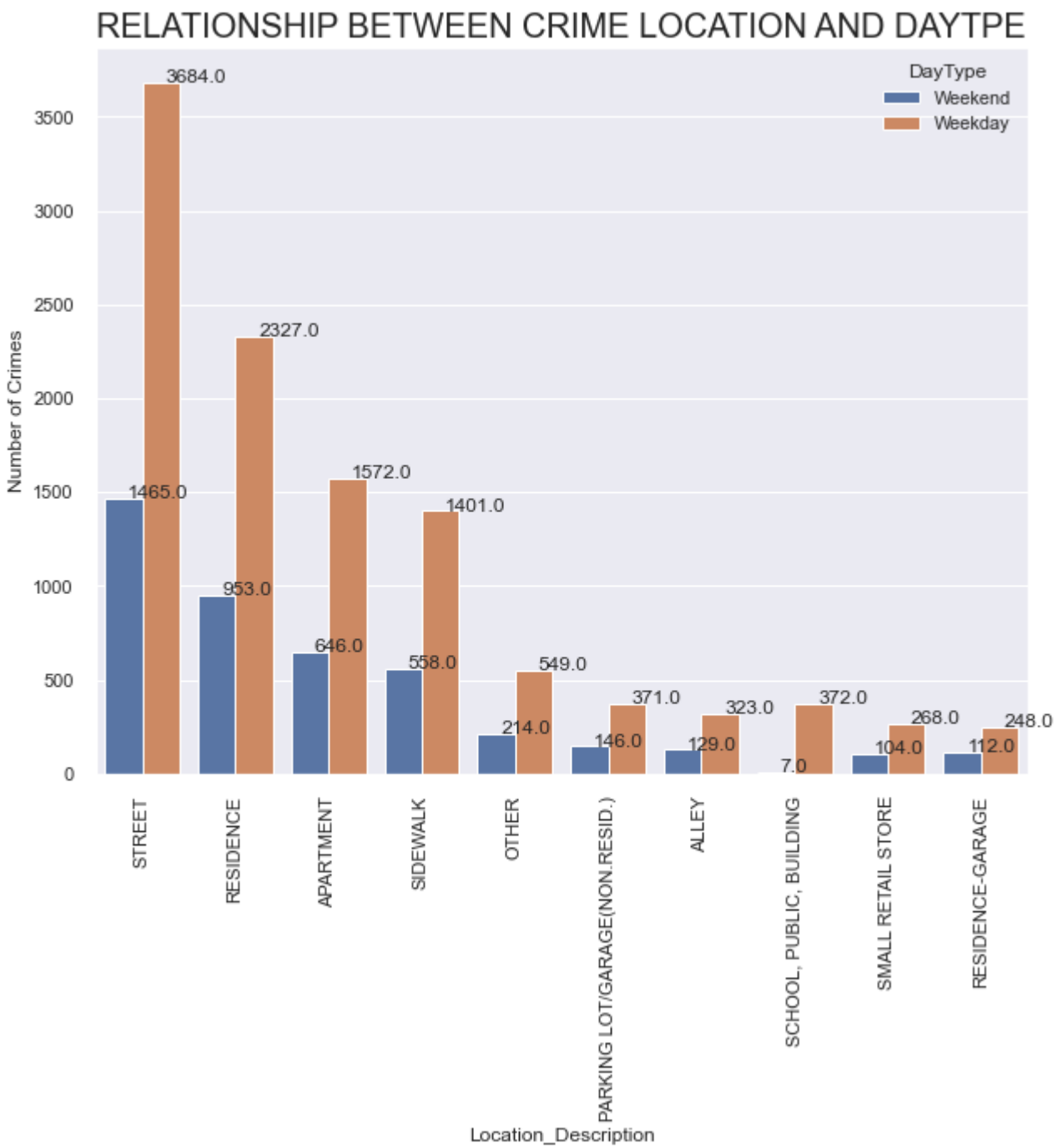
```
plt.ylabel('Number of Crimes')
ax.set_title('RELATIONSHIP BETWEEN CRIME-TYPE RATE AND DAYTPE', fontsize=20)
'''for p in ax.patches:
 ax.annotate('%{:.1f}'.format(p.get_height()), (p.get_x()+0.1, p.get_height()+50))'''

for p in ax.patches:
 ax.annotate('{:.1f}'.format(p.get_height()), (p.get_x()+0.25, p.get_height()+0.01))
plt.xticks(rotation=70)

plt.show()
```



In [134…

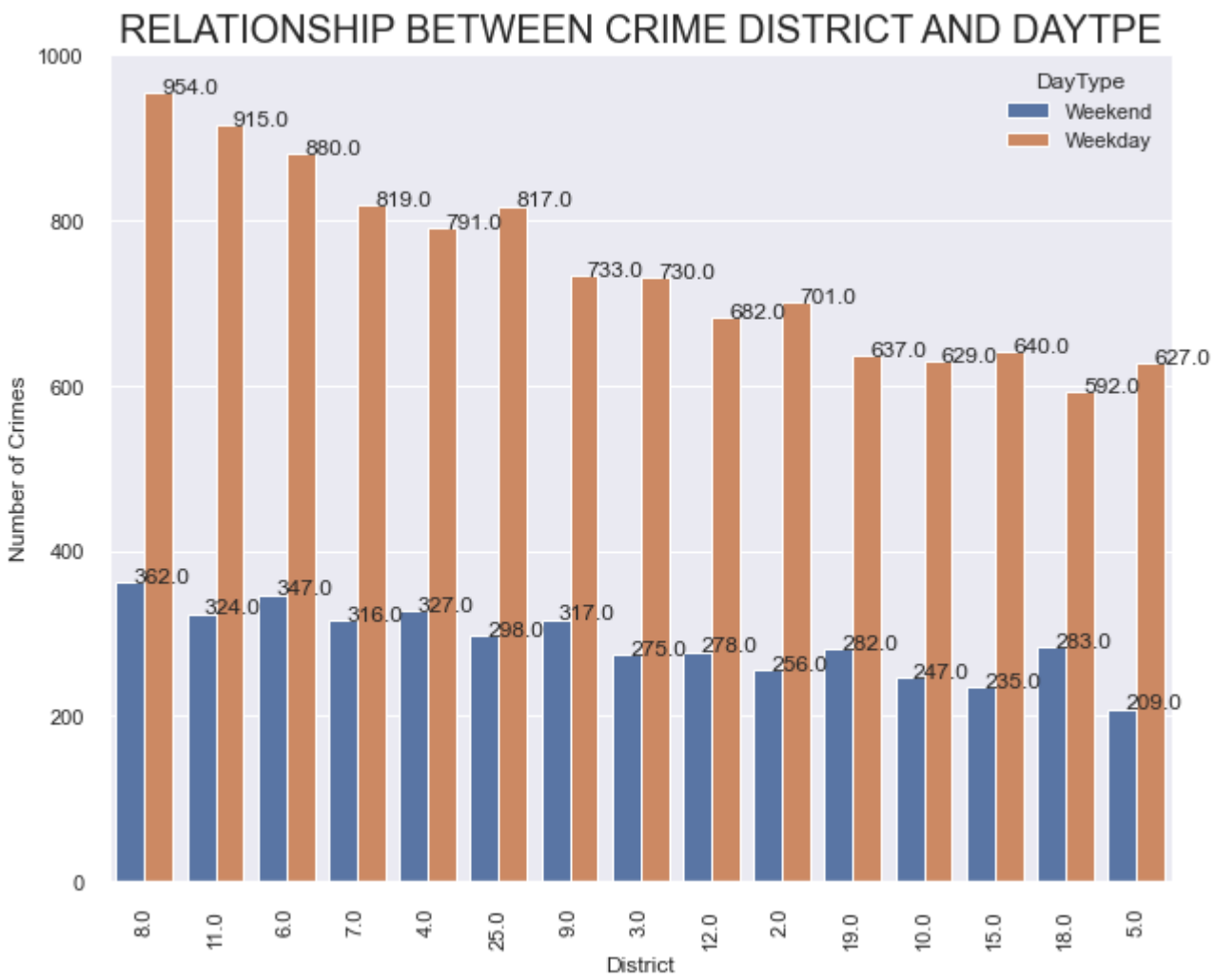
```
# CRIME LOCATION VS DAYTYPE
# Crimes occurs mostly during the weekdays more than weekends in all crime location description
```

```
sns.set(rc={'figure.figsize':(10,8)})
ax = sns.countplot(x='Location_Description', hue='DayType', data=dfFinal,
                   order = dfFinal['Location_Description'].value_counts().iloc[:10].index)
plt.ylabel('Number of Crimes')
ax.set_title('RELATIONSHIP BETWEEN CRIME LOCATION AND DAYTPE', fontsize=20)
'''for p in ax.patches:
 ax.annotate('%{:.1f}'.format(p.get_height()), (p.get_x()+0.1, p.get_height()+50))'''

for p in ax.patches:
 ax.annotate('{:.1f}'.format(p.get_height()), (p.get_x()+0.25, p.get_height()+0.01))
plt.xticks(rotation=90)

plt.show()
```



In [135…
```
# DISTRICT VS DAYTYPE
```

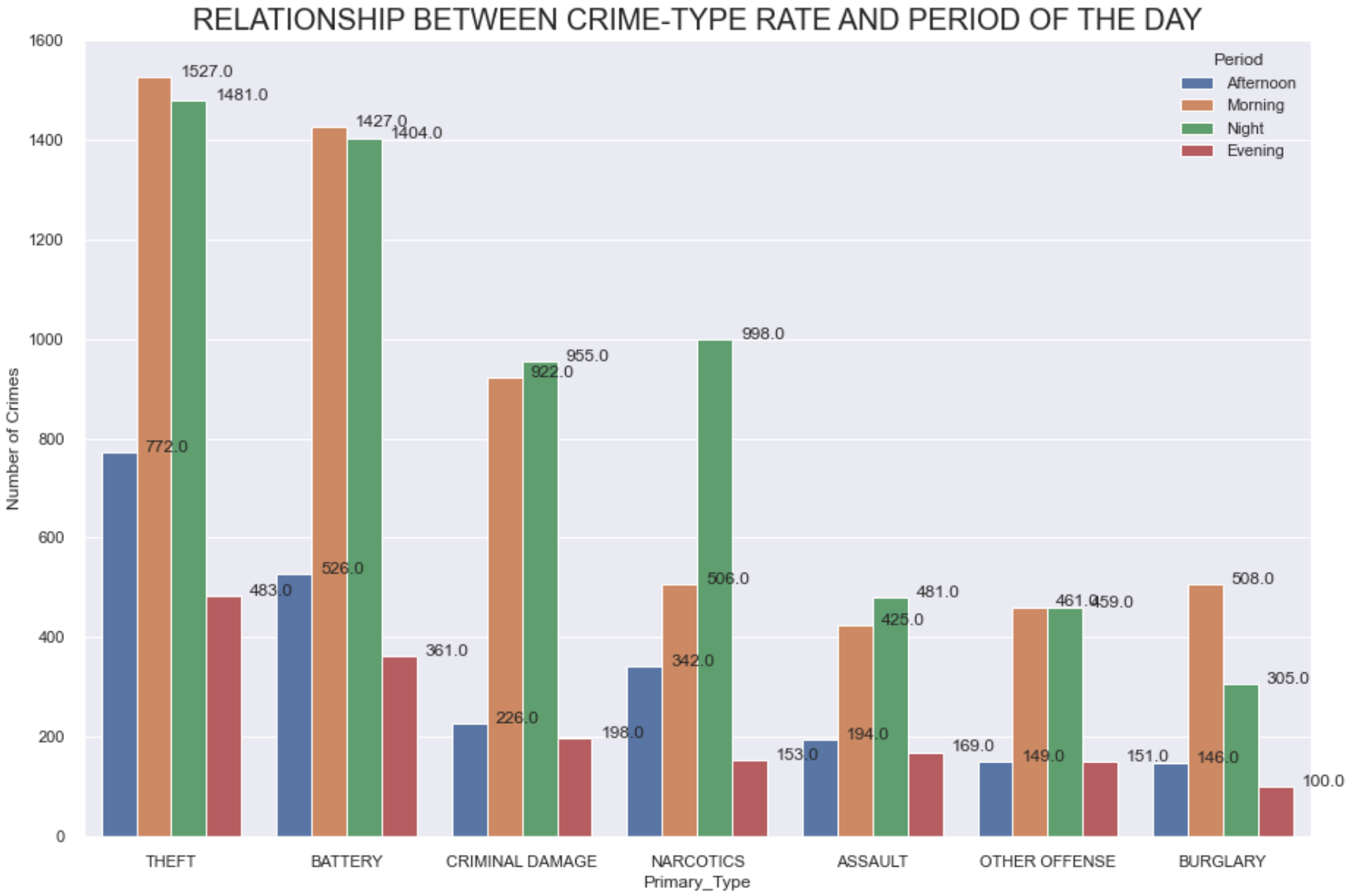
```
# District 8.0 has the most crime occurrence on weekends but generaly all crimes take place more on weekdays.

sns.set(rc={'figure.figsize':(10,8)})
ax = sns.countplot(x='District', hue='DayType', data=dfFinal,
                   order = dfFinal['District'].value_counts().iloc[:15].index)
plt.ylabel('Number of Crimes')
ax.set_title('RELATIONSHIP BETWEEN CRIME DISTRICT AND DAYTPE', fontsize=20)
'''for p in ax.patches:
 ax.annotate('%{:.1f}'.format(p.get_height()), (p.get_x()+0.1, p.get_height()+50))'''

for p in ax.patches:
 ax.annotate('{:.1f}'.format(p.get_height()), (p.get_x()+0.25, p.get_height()+0.01))
plt.xticks(rotation=90)

plt.show()
```



In [136…

```
# RELATIONSHIP BETWEEN PRIMARY-TYPE AND PERIOD OF CRIME OCCURRENCE

sns.set(rc={'figure.figsize':(15,10)})
ax = sns.countplot(x='Primary_Type', hue='Period', data=dfFinal,
                   order = dfFinal['Primary_Type'].value_counts().iloc[:7].index)
plt.ylabel('Number of Crimes')
ax.set_title('RELATIONSHIP BETWEEN CRIME-TYPE RATE AND PERIOD OF THE DAY', fontsize=20)
'''for p in ax.patches:
 ax.annotate('%{:.1f}'.format(p.get_height()), (p.get_x()+0.1, p.get_height()+50))'''

for p in ax.patches:
 ax.annotate('{:.1f}'.format(p.get_height()), (p.get_x()+0.25, p.get_height()+0.01))
```

```
#plt.xticks(rotation=45)

plt.show()

# There are more theft crimes in morning than night period, likewise battery crime rate. While
# Narcotics crimes occurs mostly at night than morning period which is also
# similar to Criminal damages, as well as Assaults occurs more at night than morning period
# Burglary happen more in the morning than nights
# Summarily, this confirms that crimes predominantly occur at nights then mornings.
```



OBSERVATION AND INFERENCE ON # PRIMARY_TYPE VS PERIOD FOR DIAGRAM ABOVE

There are more theft crimes in morning than night period, likewise battery crime rate. While

Narcotics crimes occurs mostly at night than morning period which is also

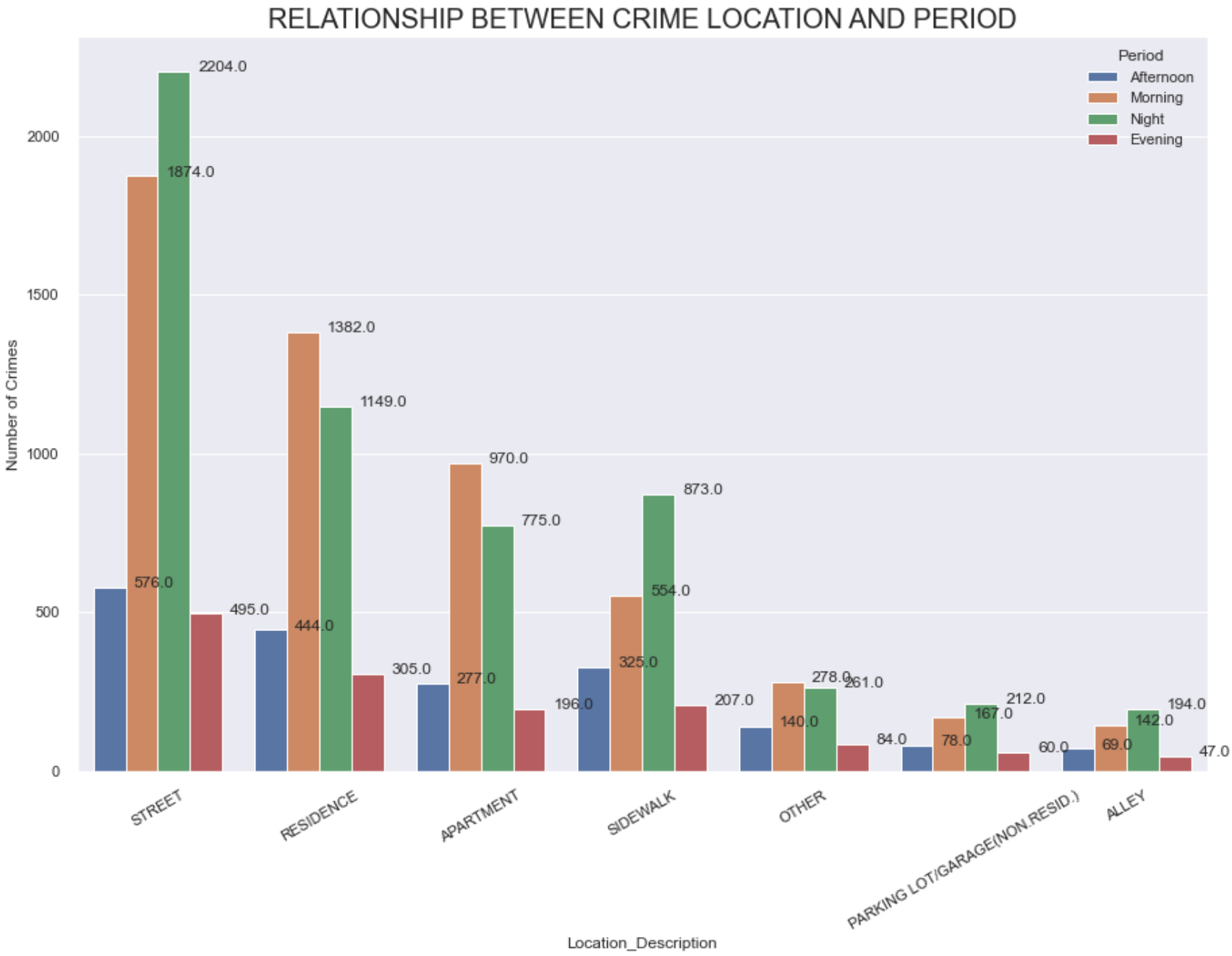
similar to Criminal damages, as well as Assaults occurs more at night than morning period

Burglary happen more in the morning than nights

Summarily, this confirms that crimes predominantly occur at nights then mornings.

In [137…

```
# CRIME LOCATION VS PERIOD

sns.set(rc={'figure.figsize':(15,10)})
ax = sns.countplot(x='Location_Description', hue='Period', data=dfFinal,
                   order = dfFinal['Location_Description'].value_counts().iloc[:7].index)
plt.ylabel('Number of Crimes')
ax.set_title('RELATIONSHIP BETWEEN CRIME LOCATION AND PERIOD', fontsize=20)
'''for p in ax.patches:
 ax.annotate('%{:.1f}'.format(p.get_height()), (p.get_x()+0.1, p.get_height()+50))'''

for p in ax.patches:
 ax.annotate('{:.1f}'.format(p.get_height()), (p.get_x()+0.25, p.get_height()+0.01))
plt.xticks(rotation=30)

plt.show()

# Most Street crimes happens more at night than the mornings likewise;
# Sidewalks,parking lots and Alleys crimes follow same trend. While
# Residence and Apartment crimes occurs more in the mornings than night period.
# Summarily, it wont be out context to deduce that Domestic crimes occur in the morning while non-domestics at night.
```



OBSERVATION AND INFERENCE FOR CRIME LOCATION VS PERIOD

Most Street crimes happens more at night than the mornings likewise; Sidewalks, parking lots and Alleys crimes follow same trend

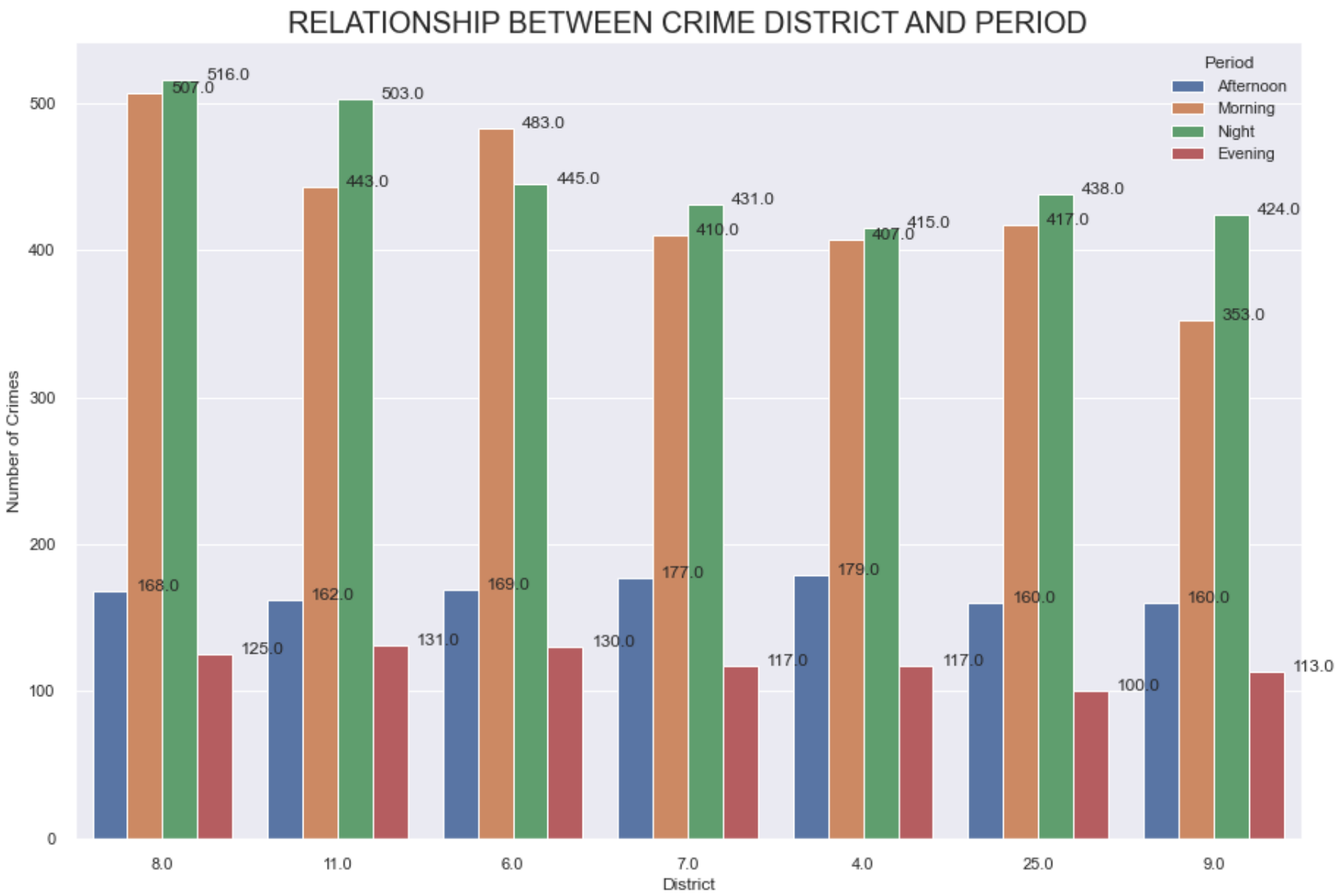
While

Residence and Apartment crimes occurs more in the mornings than night period.

Summarily, it wont be out context to deduce that Domestic crimes occur in the morning while non-domestics at night.

In [138…

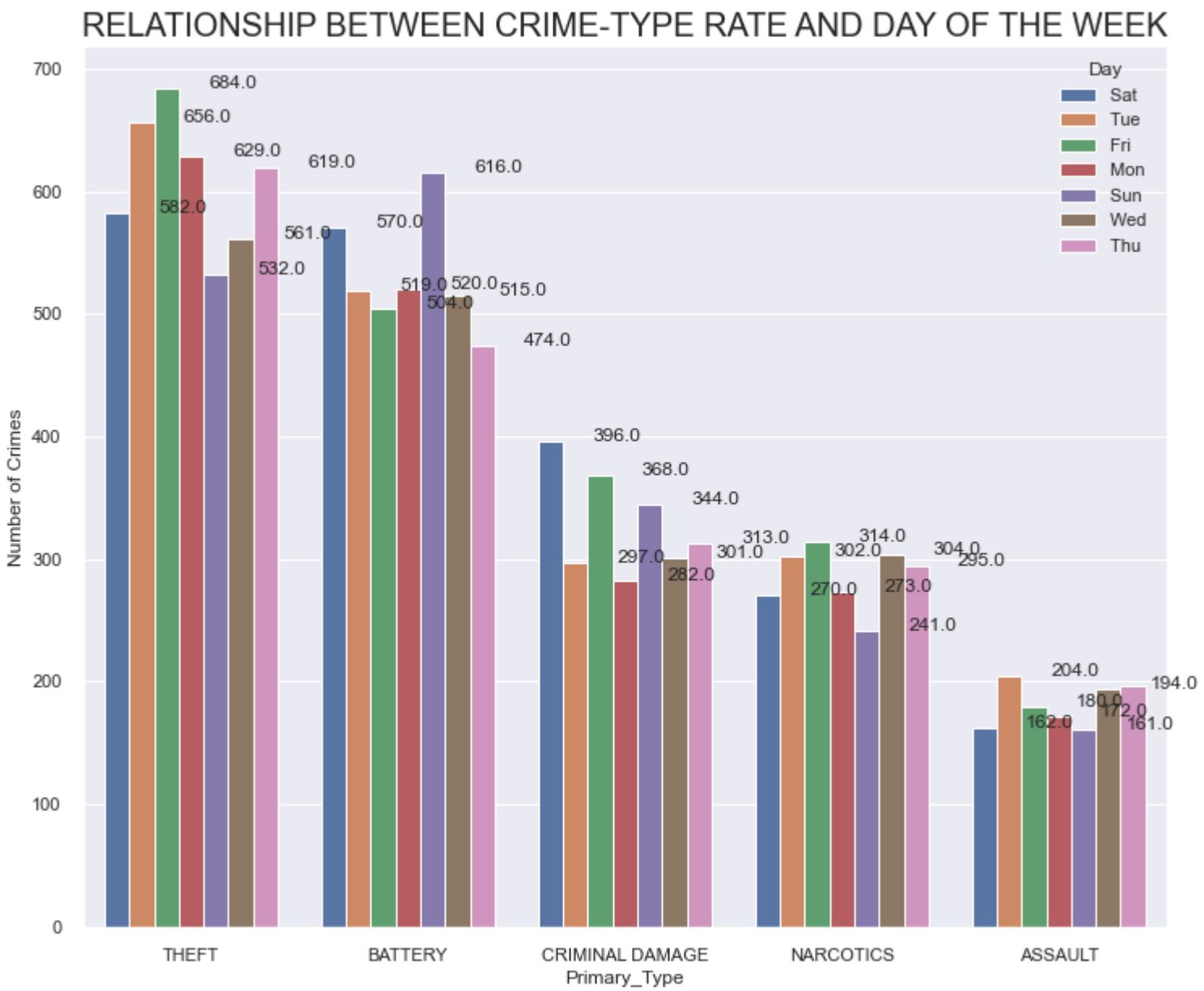
```
# DISTRICT VS PERIOD

sns.set(rc={'figure.figsize':(15,10)})
ax = sns.countplot(x='District', hue='Period', data=dfFinal,
                   order = dfFinal['District'].value_counts().iloc[:7].index)
plt.ylabel('Number of Crimes')
ax.set_title('RELATIONSHIP BETWEEN CRIME DISTRICT AND PERIOD', fontsize=20)
'''for p in ax.patches:
 ax.annotate('%{:.1f}'.format(p.get_height()), (p.get_x()+0.1, p.get_height()+50))'''

for p in ax.patches:
 ax.annotate('{:.1f}'.format(p.get_height()), (p.get_x()+0.25, p.get_height()+0.01))
#plt.xticks(rotation=30)

plt.show()

# These districts seem to follow a pattern except for District 6.0 with more crime rate in the morning than night.
```



In [139…

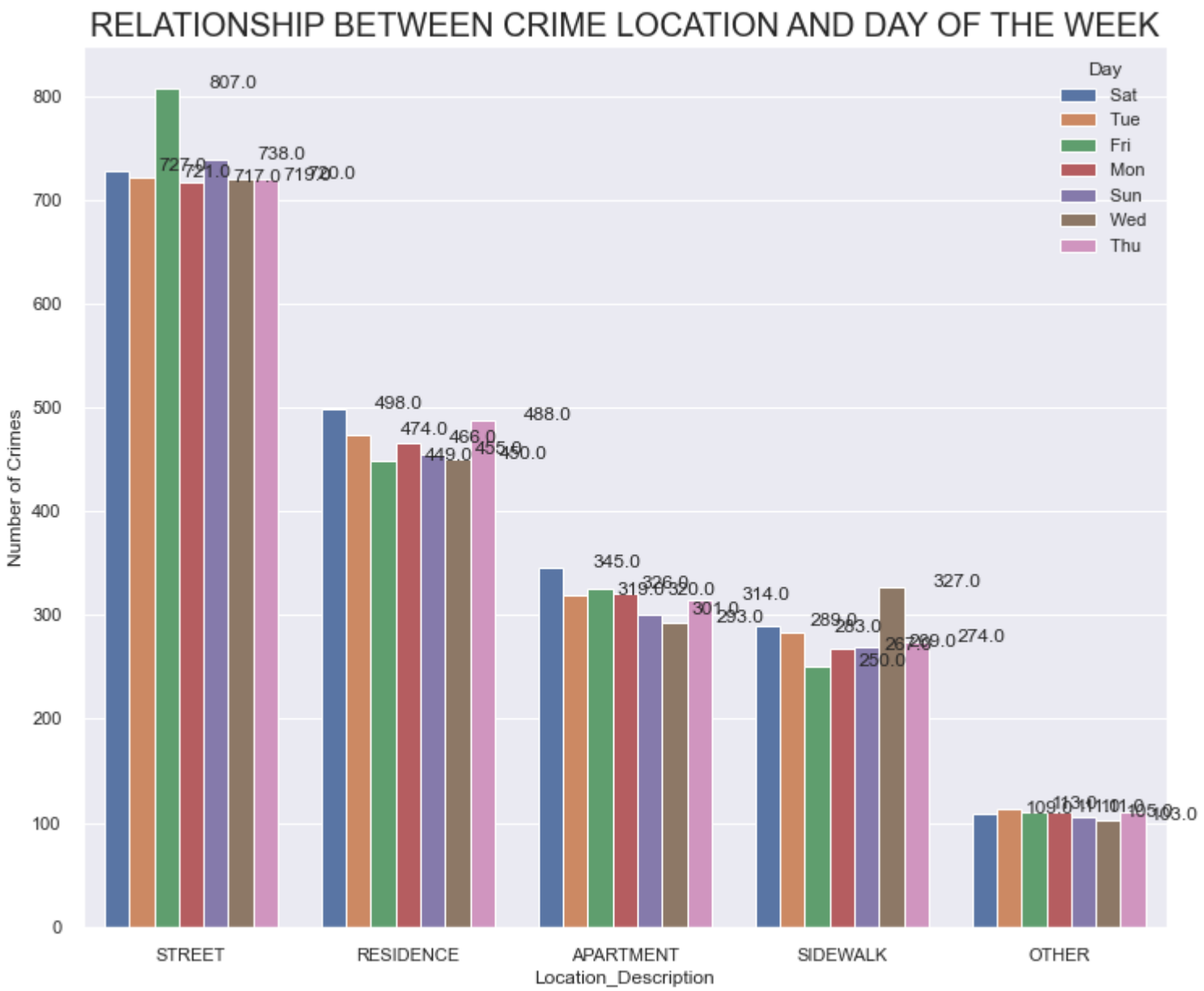
```
# PRIMARY_TYPE VS DAY

sns.set(rc={'figure.figsize':(12,10)})
ax = sns.countplot(x='Primary_Type', hue='Day', data=dfFinal,
                   order = dfFinal['Primary_Type'].value_counts().iloc[:5].index)
plt.ylabel('Number of Crimes')
ax.set_title('RELATIONSHIP BETWEEN CRIME-TYPE RATE AND DAY OF THE WEEK', fontsize=20)
'''for p in ax.patches:
```

```
 ax.annotate('%{:.1f}'.format(p.get_height()), (p.get_x()+0.1, p.get_height()+50))'''

for p in ax.patches:
 ax.annotate('{:.1f}'.format(p.get_height()), (p.get_x()+0.25, p.get_height()+0.01))
#plt.xticks(rotation=45)

plt.show()



# There are more theft crimes on Fridays likewise Narcotics crimes too
# QUESTION: Could it be that those involve in Narcotics crimes also embark on theft crimes???

# Battery crime rate seems to occur mostly on Sundays then Saturday.This confirms the high battery crimes on weekends.
# Criminal damages happens more on saturdays then fridays.
# There are more Assaults crimes on increase on Tuesdays,Thursdays and Wednesday.
```



In [140…
```
# LOCATION DESCRIPTION VS DAY

sns.set(rc={'figure.figsize':(12,10)})
ax = sns.countplot(x='Location_Description', hue='Day', data=dfFinal,
                   order = dfFinal['Location_Description'].value_counts().iloc[:5].index)
plt.ylabel('Number of Crimes')
```

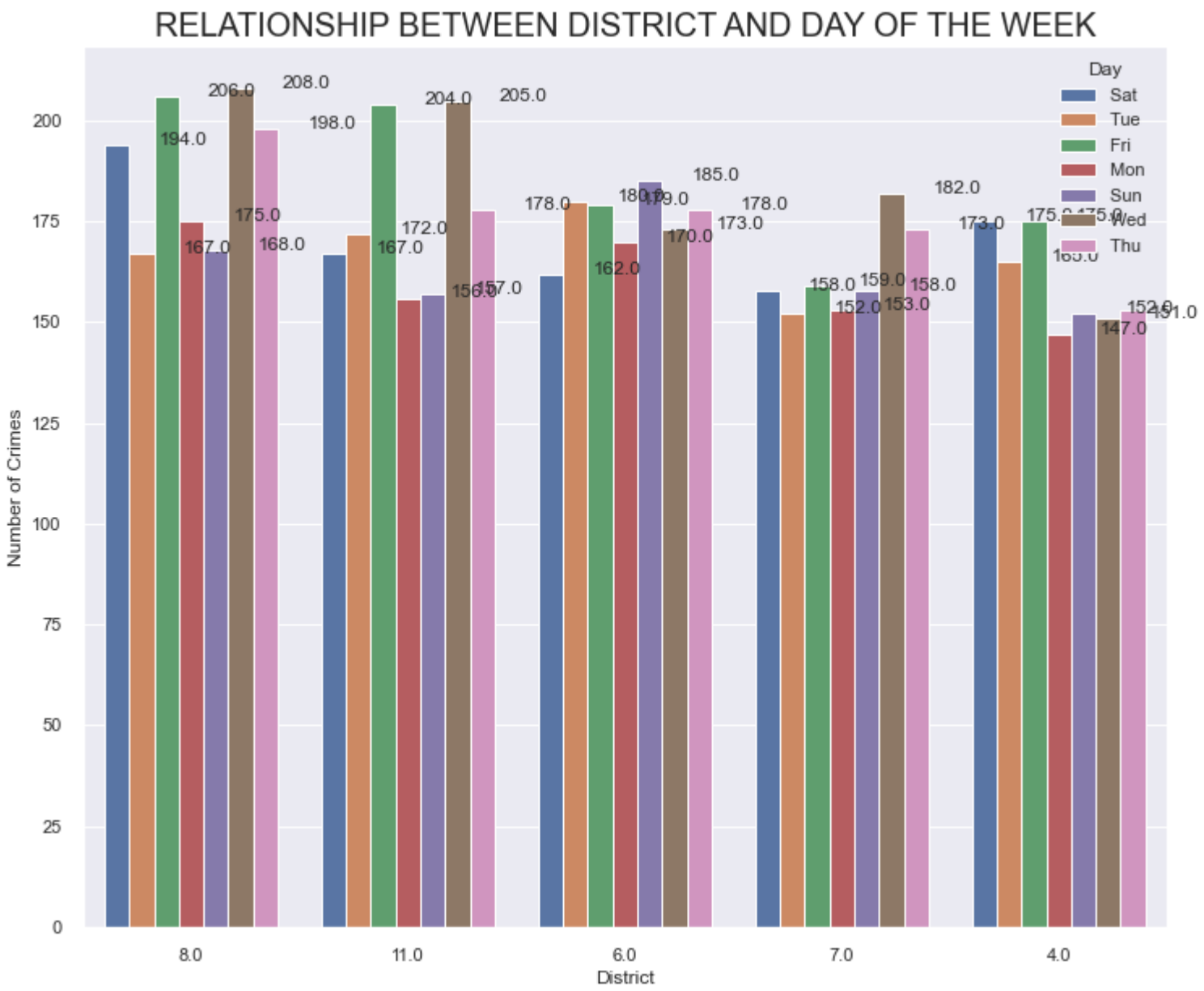
```
ax.set_title('RELATIONSHIP BETWEEN CRIME LOCATION AND DAY OF THE WEEK', fontsize=20)
'''for p in ax.patches:
 ax.annotate('%{:.1f}'.format(p.get_height()), (p.get_x()+0.1, p.get_height()+50))'''

for p in ax.patches:
 ax.annotate('{:.1f}'.format(p.get_height()), (p.get_x()+0.25, p.get_height()+0.01))
#plt.xticks(rotation=45)

plt.show()

# Crimes on the street seems to increase on Fridays (weekday) than any other days and crime Location.
# Residence crimes occurred most on Saturdays likewise that of Apartment thereby affirms domestic happen more on weeken
# Sidewalk crimes occurs more on Wednesdays (weekday) which also confirms non-domestic crimes be on increase on weekday
```



In [141…

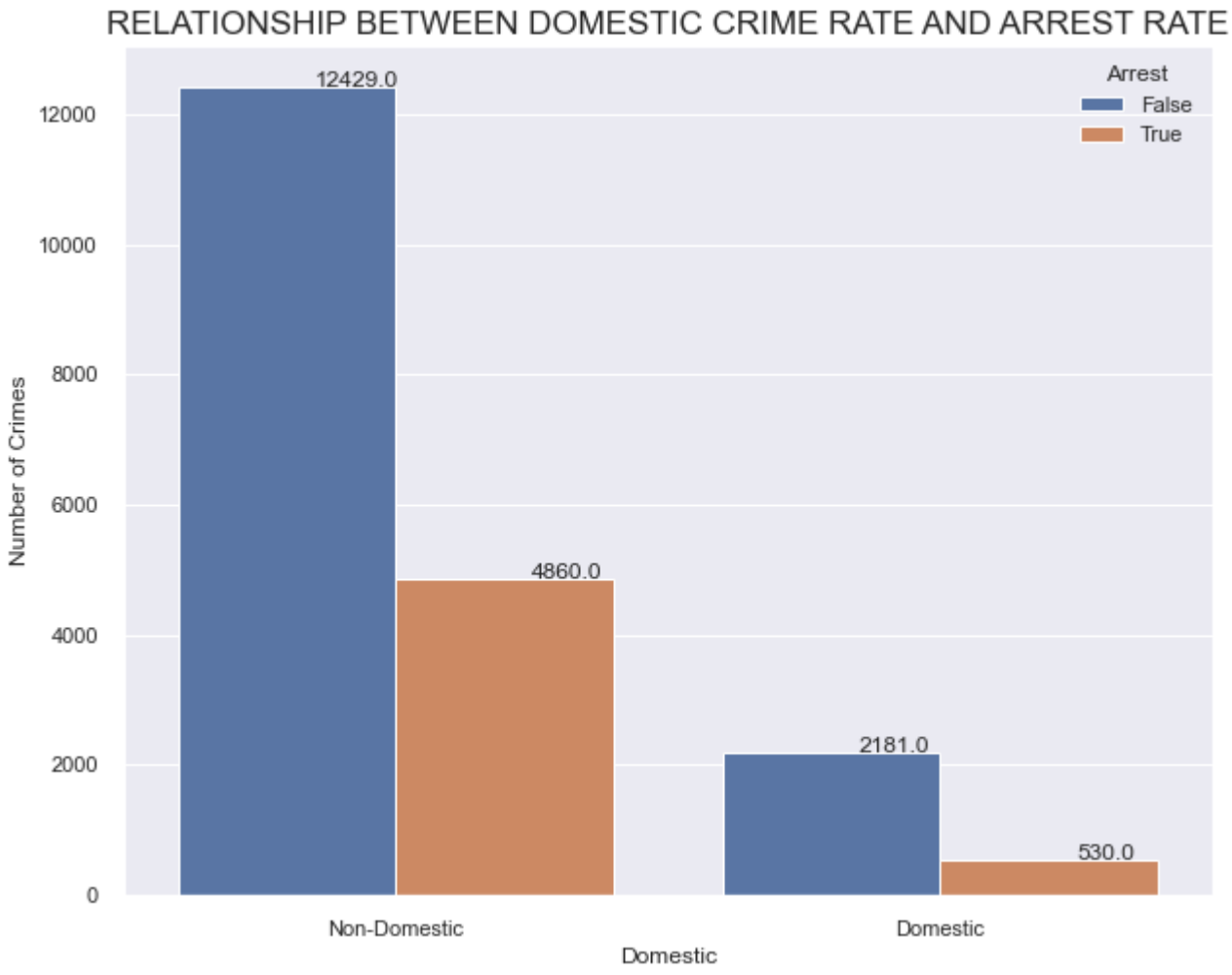
```
# DISTRICT VS DAY

sns.set(rc={'figure.figsize':(12,10)})
ax = sns.countplot(x='District', hue='Day', data=dfFinal,
                   order = dfFinal['District'].value_counts().iloc[:5].index)
plt.ylabel('Number of Crimes')
```

```
ax.set_title('RELATIONSHIP BETWEEN DISTRICT AND DAY OF THE WEEK', fontsize=20)
'''for p in ax.patches:
 ax.annotate('%{:.1f}'.format(p.get_height()), (p.get_x()+0.1, p.get_height()+50))'''

for p in ax.patches:
 ax.annotate('{:.1f}'.format(p.get_height()), (p.get_x()+0.25, p.get_height()+0.01))
#plt.xticks(rotation=45)

plt.show()


# District 8.0 has its crime peaks on Wednesdays, Fridays and Thurdays likewise District 11.0. and 7.0.
# District 6.0 has its crime spikes on Sundays and its drop on Saturday. While
#District 4.0 has its crime peaks on Fridays then Saturdays with crime drop on Mondays.
```



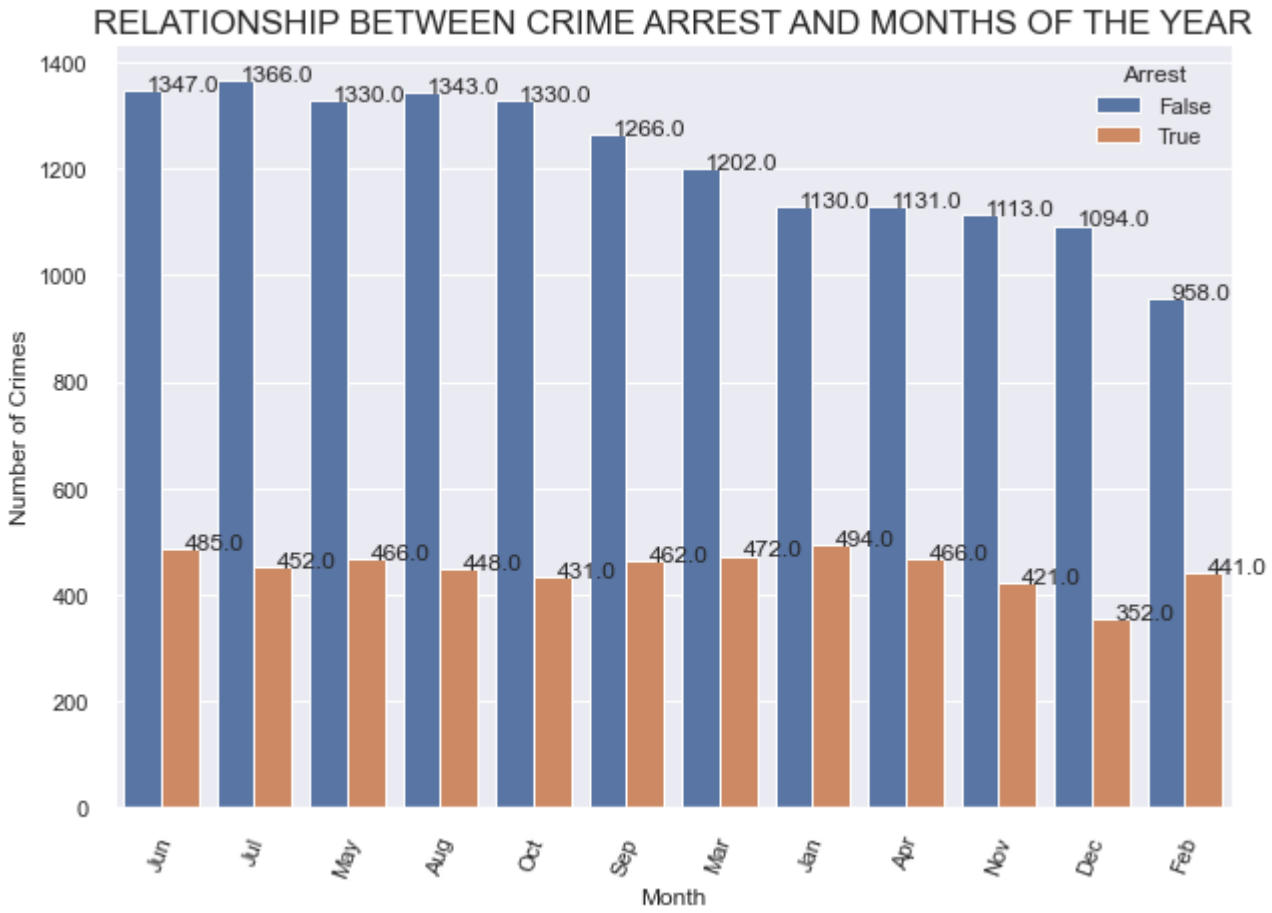
In [142…
```
# RELATIONSHIP BETWEEN DOMESTIC CRIME RATE VERSUS ARREST RATE
# Arrest rate seem to be in direct proportion to crime rate for the domestic and non-domestic.
# Therefore, the arrest rate non-domestic crimes exceeds that of domestic since it directly proportional crime rate.

sns.set(rc={'figure.figsize':(10,8)})
ax = sns.countplot(x='Domestic', hue='Arrest', data=dfFinal,
```

```
                      order = dfFinal['Domestic'].value_counts().iloc[:].index)
ax.set_xticklabels(['Non-Domestic','Domestic'])
plt.ylabel('Number of Crimes')
ax.set_title('RELATIONSHIP BETWEEN DOMESTIC CRIME RATE AND ARREST RATE', fontsize=17)
'''for p in ax.patches:
 ax.annotate('%{:.1f}'.format(p.get_height()), (p.get_x()+0.1, p.get_height()+50))'''

for p in ax.patches:
 ax.annotate('{:.1f}'.format(p.get_height()), (p.get_x()+0.25, p.get_height()+0.01))
#plt.xticks(rotation=70)

plt.show()
```



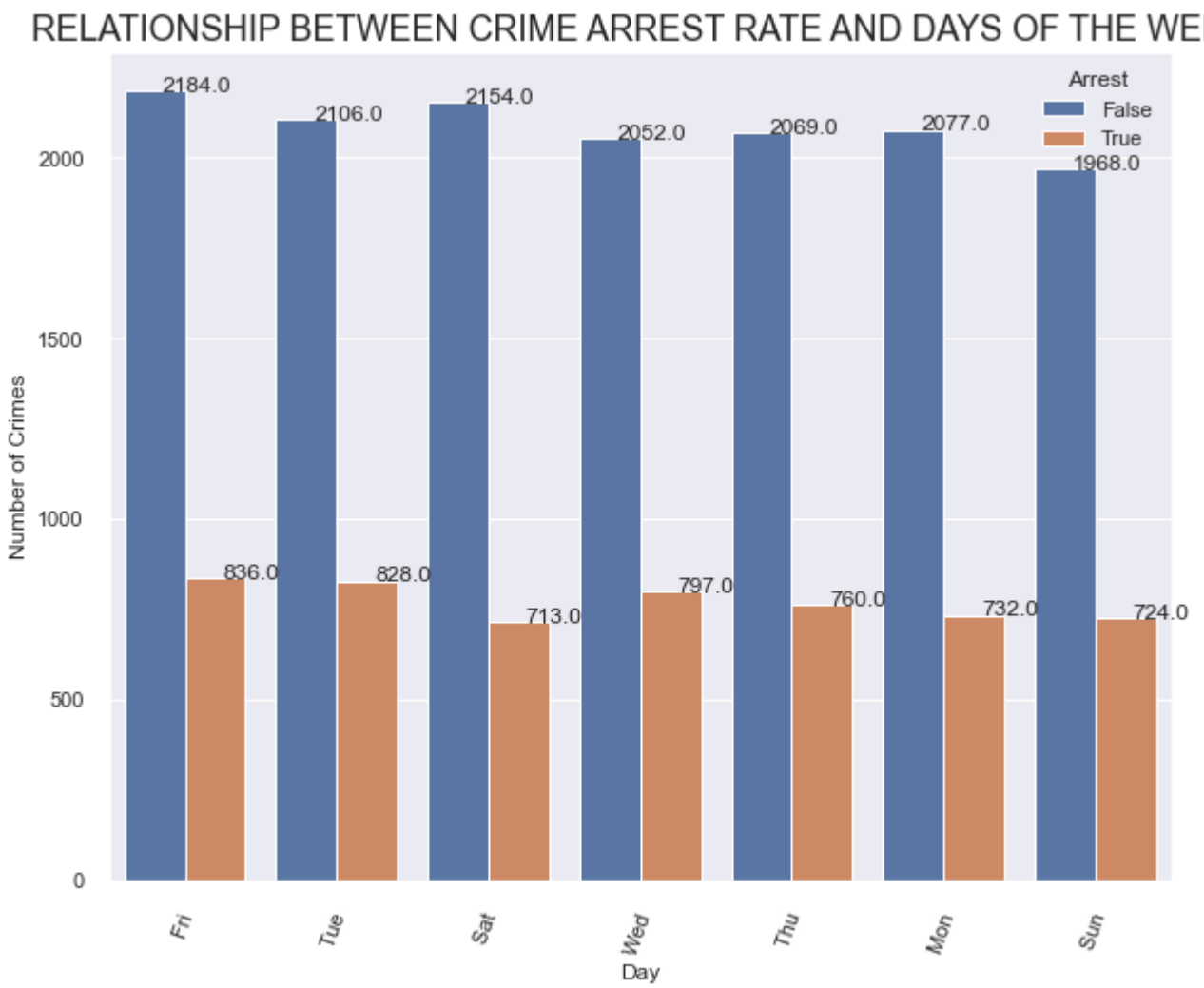
In [143…

```
# RELATIONSHIP BETWEEN CRIME ARREST AND MONTHS OF THE YEAR (descending order)
# Arrest rate seem to be in direct proportion to crime rate for each month.

sns.set(rc={'figure.figsize':(10,7)})
ax = sns.countplot(x='Month', hue='Arrest', data=dfFinal,
                   order = dfFinal['Month'].value_counts().iloc[:12].index)
plt.ylabel('Number of Crimes')
ax.set_title('RELATIONSHIP BETWEEN CRIME ARREST AND MONTHS OF THE YEAR', fontsize=17)
'''for p in ax.patches:
 ax.annotate('%{:.1f}'.format(p.get_height()), (p.get_x()+0.1, p.get_height()+50))'''

for p in ax.patches:
 ax.annotate('{:.1f}'.format(p.get_height()), (p.get_x()+0.25, p.get_height()+0.01))
plt.xticks(rotation=70)

plt.show()
```



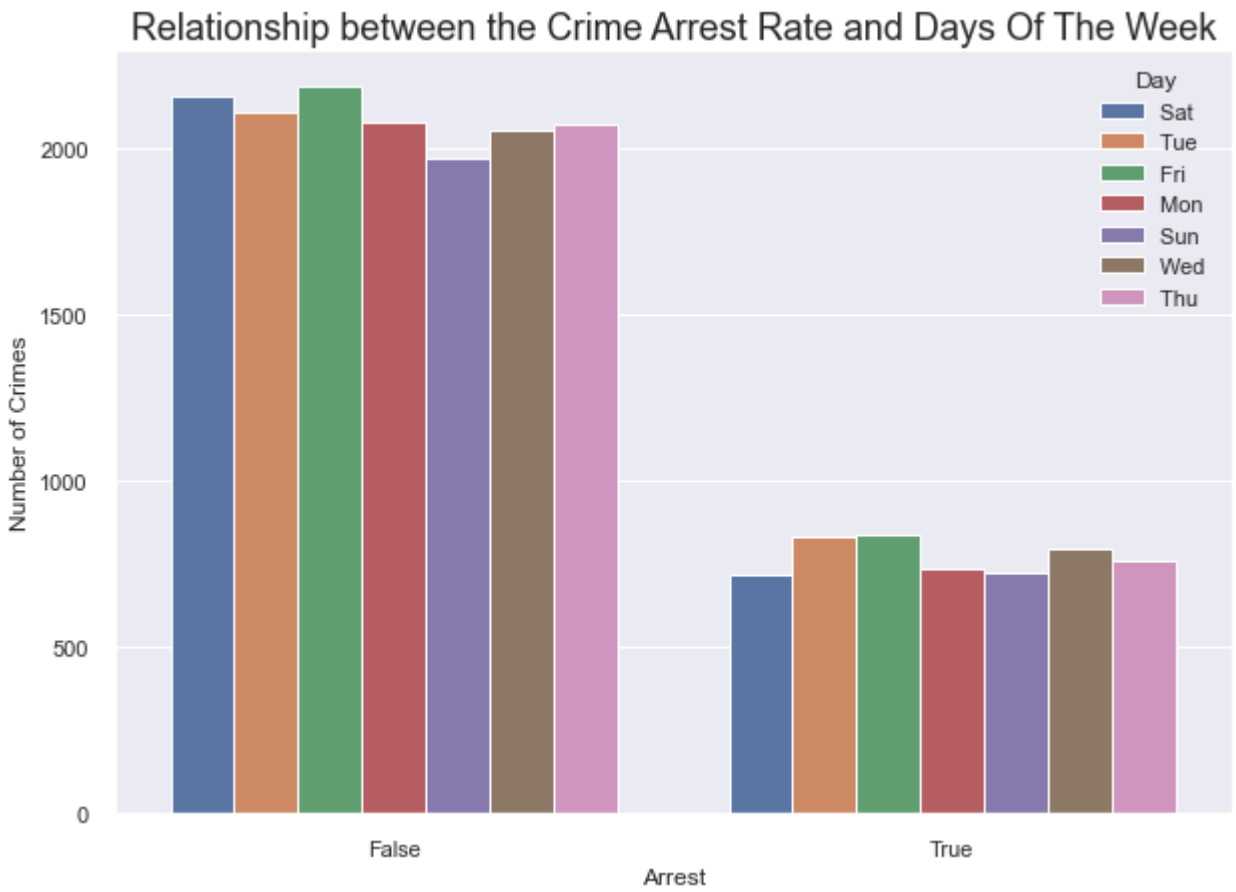
In [144…

```
# RELATIONSHIP BETWEEN CRIME ARREST RATE AND DAYS OF THE WEEK

# The arrest rate for Friday seems higher than any day of the week could this be because of the high crime rate on Frid

sns.set(rc={'figure.figsize':(10,8)})
ax = sns.countplot(x='Day', hue='Arrest', data=dfFinal,
                   order = dfFinal['Day'].value_counts().iloc[:].index)
plt.ylabel('Number of Crimes')
ax.set_title('RELATIONSHIP BETWEEN CRIME ARREST RATE AND DAYS OF THE WEEK', fontsize=18)
'''for p in ax.patches:
 ax.annotate('%{:.1f}'.format(p.get_height()), (p.get_x()+0.1, p.get_height()+50))'''

for p in ax.patches:
 ax.annotate('{:.1f}'.format(p.get_height()), (p.get_x()+0.25, p.get_height()+0.01))
plt.xticks(rotation=70)

plt.show()
```



In [145…

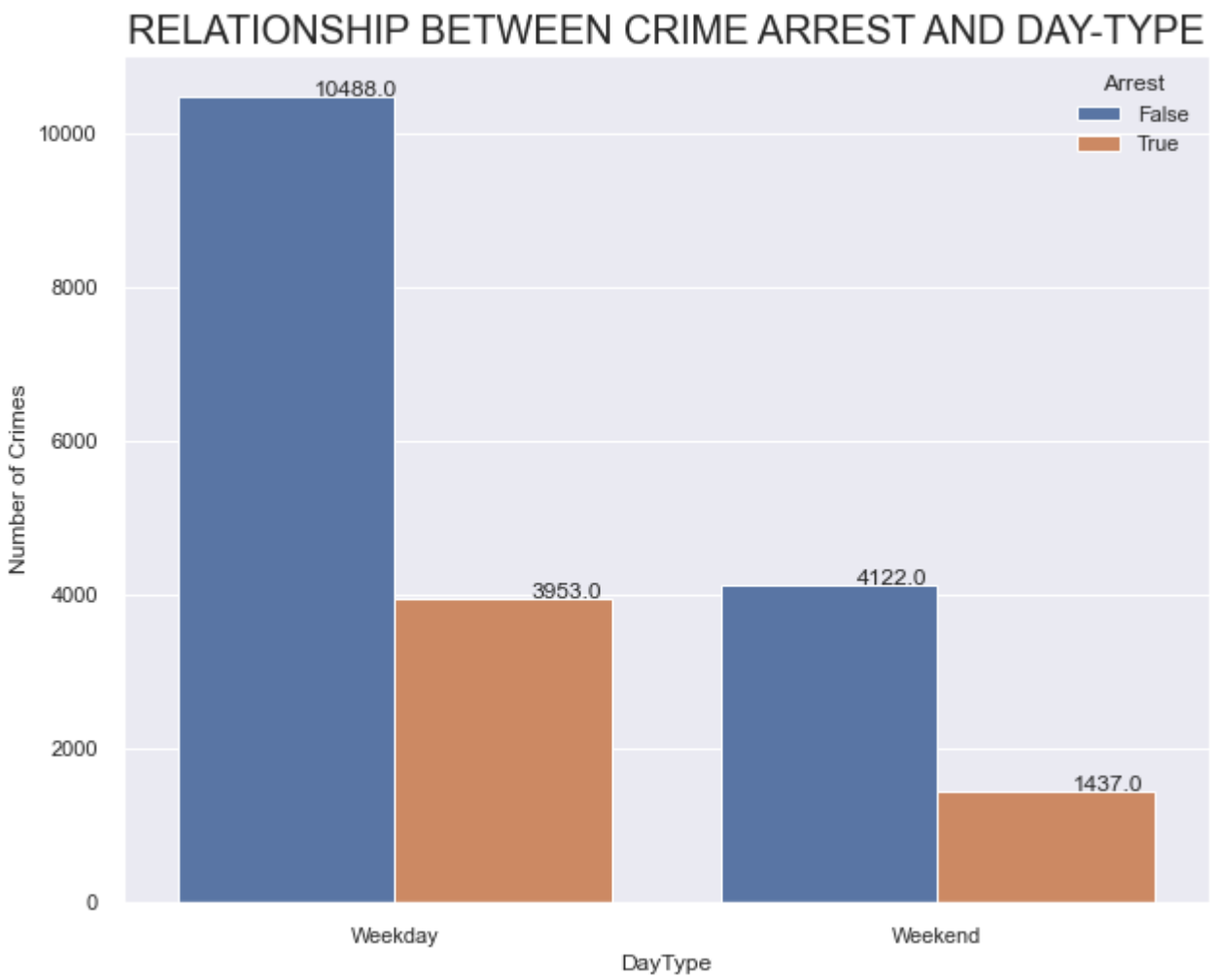
```
# The arrest rate for friday seems higher than any day of the week.

sns.set(rc={'figure.figsize':(10,7)})
Crime = sns.countplot(x='Arrest', hue='Day', data=dfFinal)
Crime.set_xticklabels(["False", "True"])
Crime.set_title('Relationship between the Crime Arrest Rate and Days Of The Week', fontsize=18)
plt.ylabel('Number of Crimes')

plt.show()
```



In [146…

```
# RELATIONSHIP BETWEEN CRIME ARREST AND DAY-TYPE

# Looking into the proportion of arrest for each dayType.
# Therefore, retrieving the ratio between true and false arrest.

sns.set(rc={'figure.figsize':(10,8)})
ax = sns.countplot(x='DayType', hue='Arrest', data=dfFinal,
                   order = dfFinal['DayType'].value_counts().iloc[:12].index)
plt.ylabel('Number of Crimes')
ax.set_title('RELATIONSHIP BETWEEN CRIME ARREST AND DAY-TYPE', fontsize=20)
'''for p in ax.patches:
 ax.annotate('%{:.1f}'.format(p.get_height()), (p.get_x()+0.1, p.get_height()+50))'''

for p in ax.patches:
 ax.annotate('{:.1f}'.format(p.get_height()), (p.get_x()+0.25, p.get_height()+0.01))
#plt.xticks(rotation=70)
```

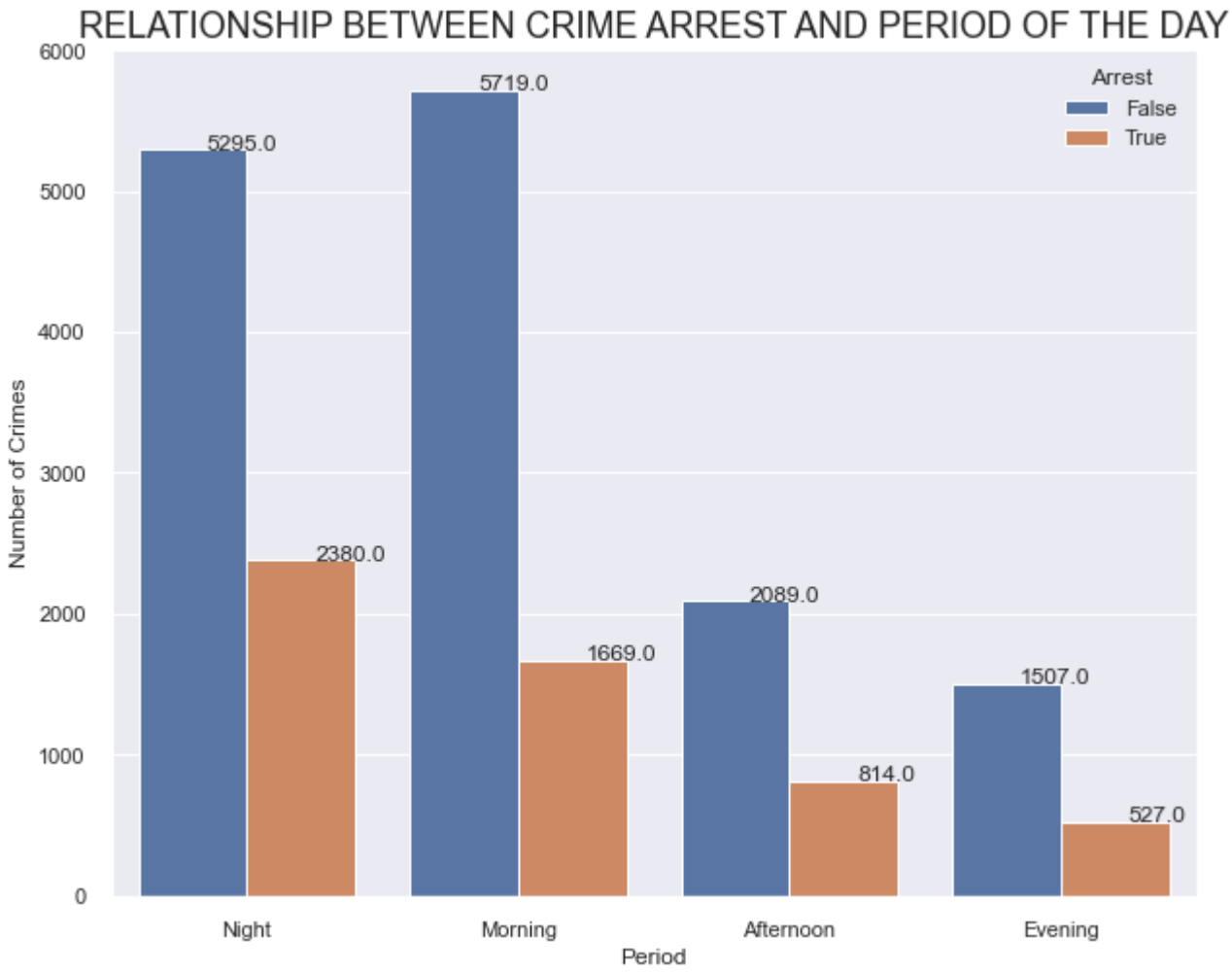
```
plt.show()
```



In [147…

```
# RELATIONSHIP BETWEEN CRIME ARREST AND PERIOD OF THE DAY

# There are more arrest at night than other period of the day and this could because of the high crime rate at night.
# Crime offenders tend to get away unarrested more in the morning than others (period of the day).

sns.set(rc={'figure.figsize':(10,8)})
ax = sns.countplot(x='Period', hue='Arrest', data=dfFinal,
                   order = dfFinal['Period'].value_counts().iloc[:10].index)
plt.ylabel('Number of Crimes')
ax.set_title('RELATIONSHIP BETWEEN CRIME ARREST AND PERIOD OF THE DAY', fontsize=18)
```

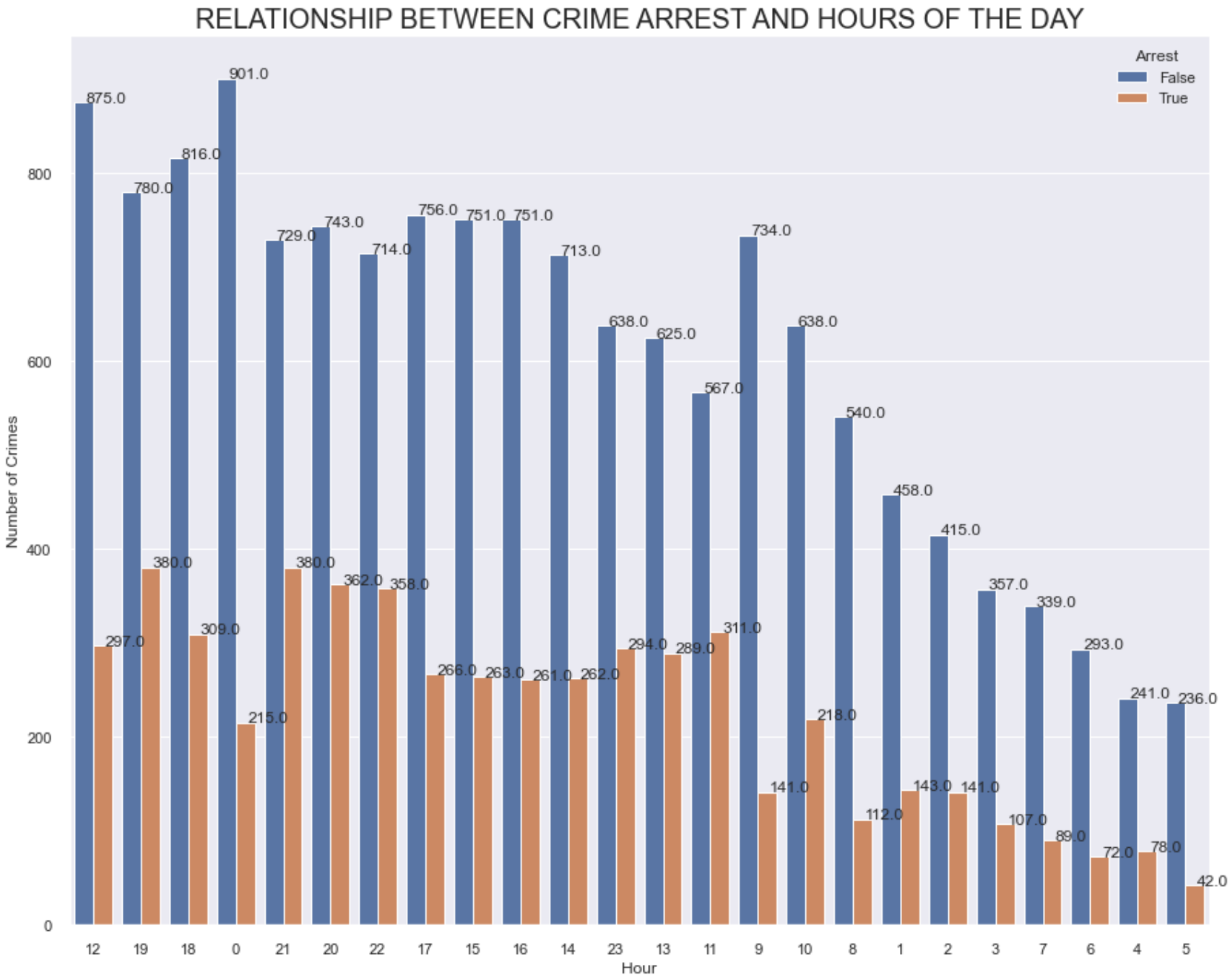
```
'''for p in ax.patches:
 ax.annotate('%{:.1f}'.format(p.get_height()), (p.get_x()+0.1, p.get_height()+50))'''

for p in ax.patches:
 ax.annotate('{:.1f}'.format(p.get_height()), (p.get_x()+0.25, p.get_height()+0.01))
#plt.xticks(rotation=70)

plt.show()
```



In [148…

```
# RELATIONSHIP BETWEEN CRIME ARREST AND HOURS OF THE DAY

# Crime offenders tend to get away unarrested more at Mid-night than other hours till about 8am.
# Arrest rate are high at 7pm and 9pm but generally from 6pm to 10pm at night.
```

```
# Which supports high crime rate at night period (which is a direct relationship).

# QUESTION ???:
# Could this be because there seem to be less activeness from PD at night till early hours in the morning.
# probably
# There seem to be a pattern for crimes in the Afternoon period from 2pm - 5pm.

sns.set(rc={'figure.figsize':(15,12)})
ax = sns.countplot(x='Hour', hue='Arrest', data=dfFinal,
                       order = dfFinal['Hour'].value_counts().iloc[:24].index)
plt.ylabel('Number of Crimes')
ax.set_title('RELATIONSHIP BETWEEN CRIME ARREST AND HOURS OF THE DAY', fontsize=20)
'''for p in ax.patches:
 ax.annotate('%{:.1f}'.format(p.get_height()), (p.get_x()+0.1, p.get_height()+50))'''

for p in ax.patches:
 ax.annotate('{:.1f}'.format(p.get_height()), (p.get_x()+0.25, p.get_height()+0.01))
#plt.xticks(rotation=70)

plt.show()
```



RELATIONSHIP BETWEEN CRIME ARREST AND HOURS OF THE DAY

Crime offenders tend to get away unarrested more at Mid-night than other hours till about 8am.

Arrest rate are high at 7pm and 9pm but generally from 6pm to 10pm at night.

Which supports high crime rate at night period (which is a direct relationship).

QUESTION ???:

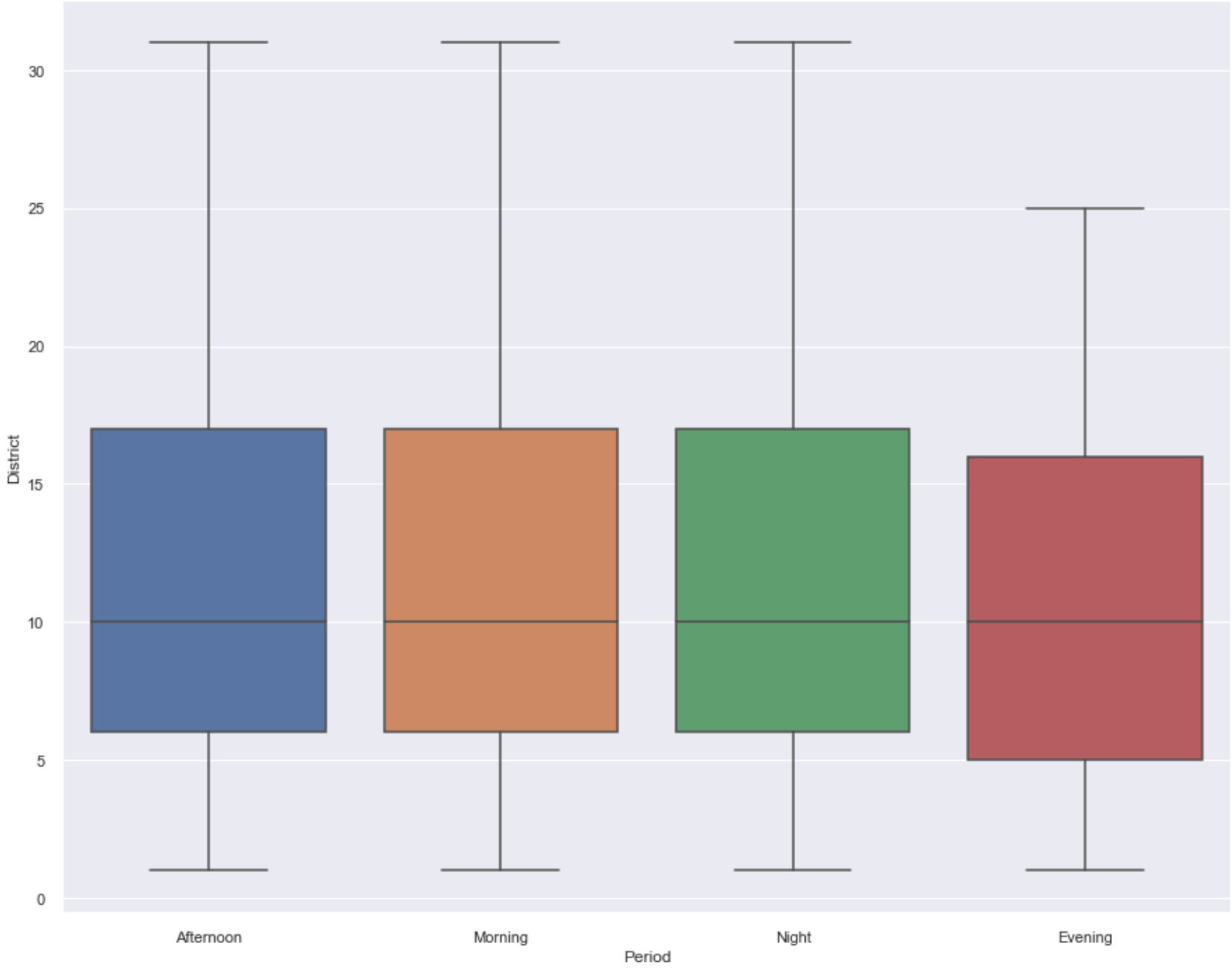
Could this be because there seem to be less activeness from PD at night till early hours in the morning.

probably

There seem to be a pattern for crimes in the Afternoon period from 2pm - 5pm.

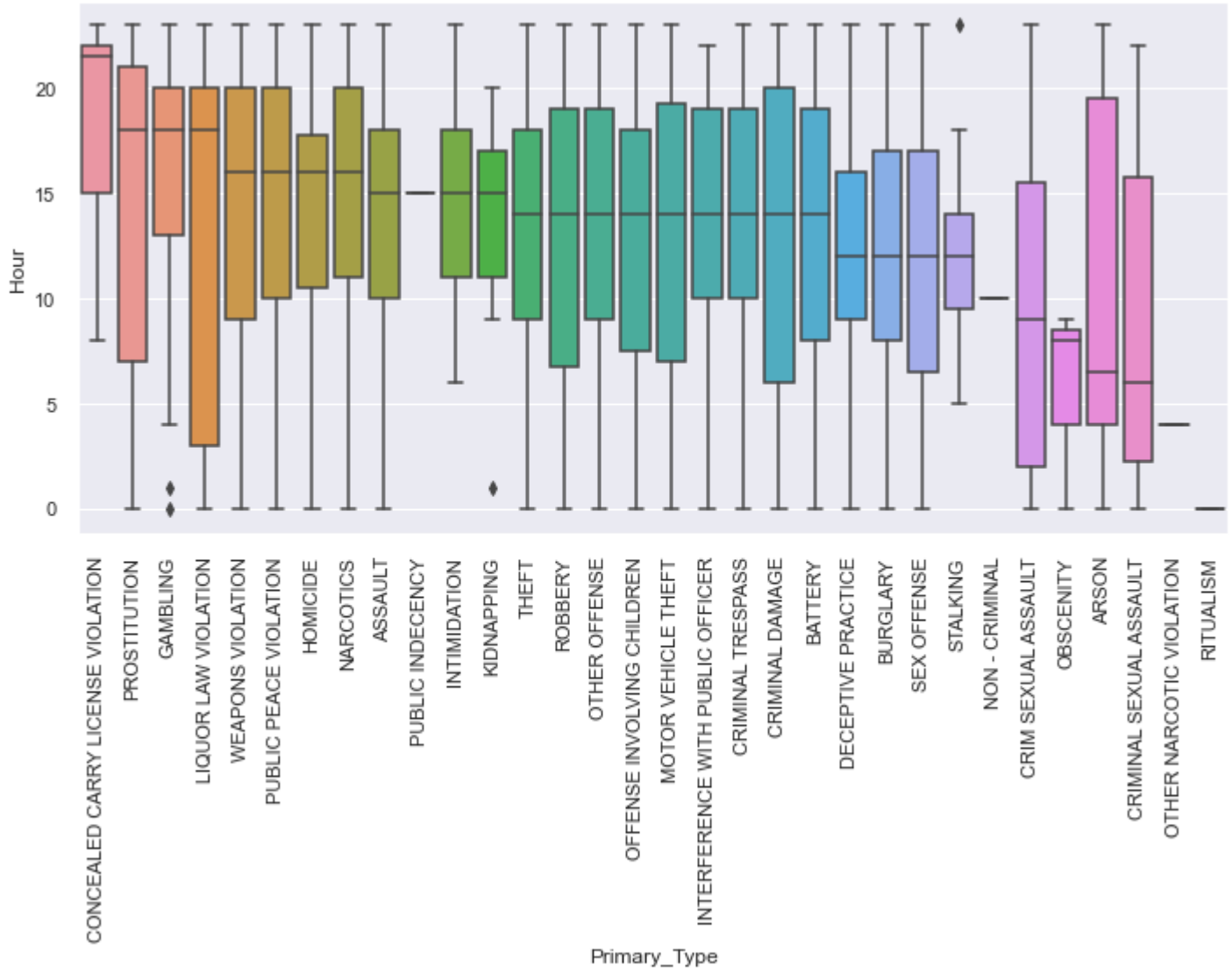
In [149…

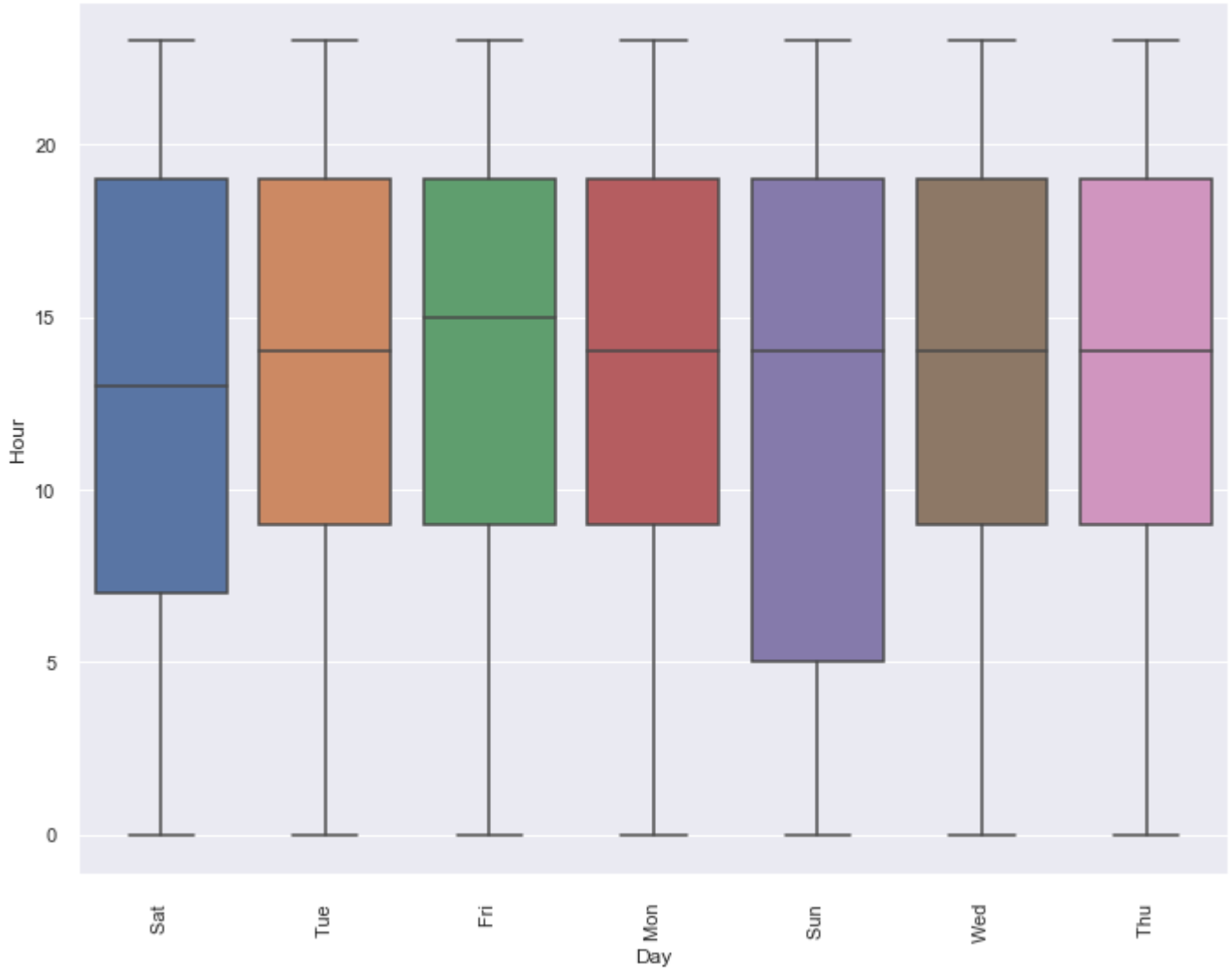
```
# DISTRICT AND PERIOD BOXPLOT ANALYSIS

sns.boxplot(x='Period', y='District', data=dfFinal)
plt.show()
```

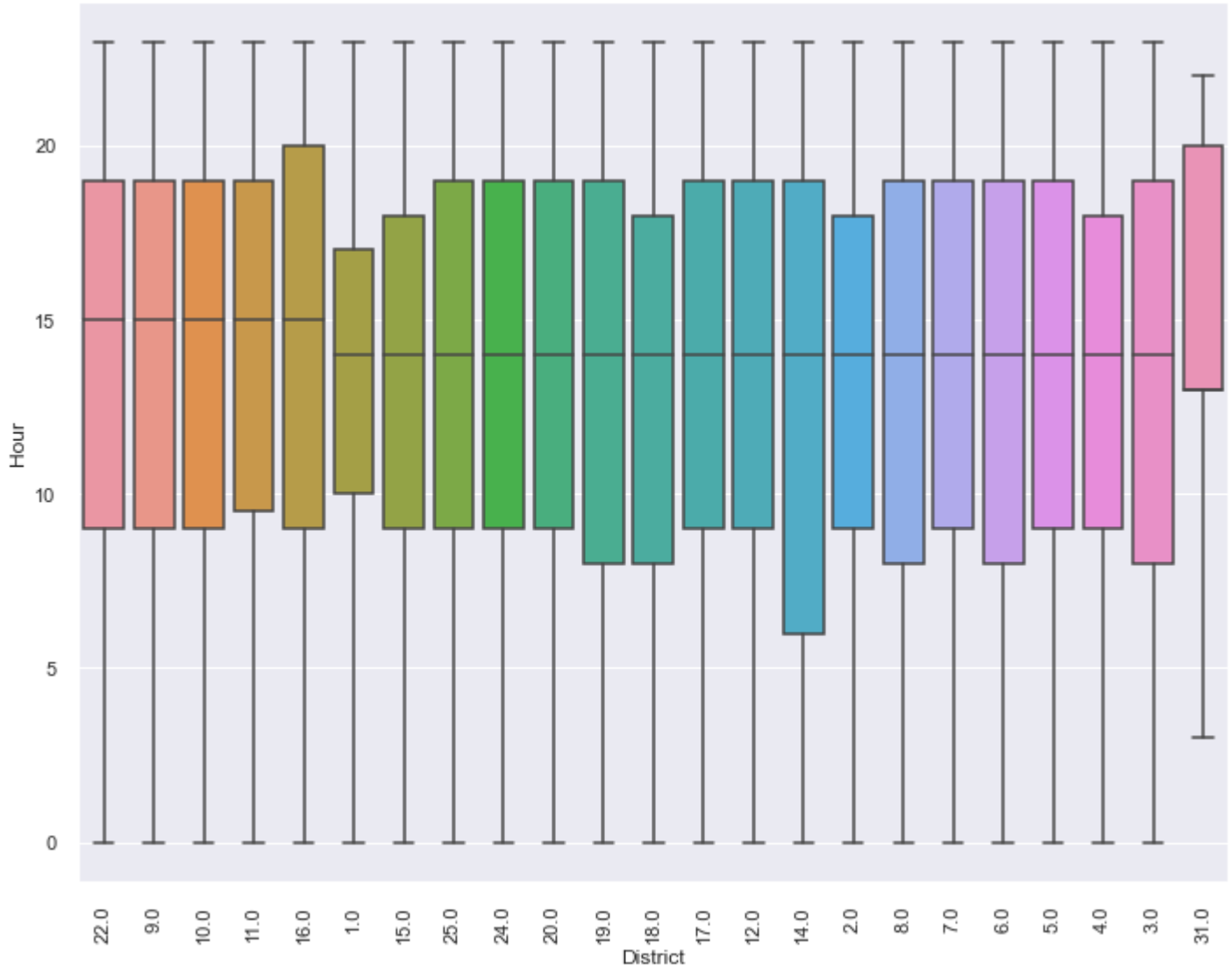


In [150…
```
# Primary type, Day and District
```

```
plt.subplots(figsize = (10,8))
group = dfFinal.groupby(by='Primary_Type').agg({'Hour': 'median'}).sort_values(by='Hour', ascending=False)
order = group.index
sns.boxplot(data=dfFinal, x='Primary_Type', y='Hour', order=order)
plt.xticks(rotation=90)
plt.tight_layout()

plt.subplots(figsize = (10,8))
order = ['Monday', 'Tuesday', 'Wednesday', 'Thursday', 'Friday', 'Saturday', 'Sunday']
sns.boxplot(data=dfFinal, x='Day', y='Hour')
plt.xticks(rotation=90)
plt.tight_layout()

plt.subplots(figsize = (10,8))
group = dfFinal.groupby(by='District').agg({'Hour': 'median'}).sort_values(by='Hour', ascending=False)
order = group.index
sns.boxplot(data=dfFinal, x='District', y='Hour', order=order)
plt.xticks(rotation=90)
plt.tight_layout()
```


Hour
0
5
10
15
20
CONCEALED CARRY LICENSE VIOLATION
PROSTITUTION
GAMBLING
LIQUOR LAW VIOLATION
WEAPONS VIOLATION
PUBLIC PEACE VIOLATION
HOMICIDE
NARCOTICS
ASSAULT
PUBLIC INDECENCY
INTIMIDATION
KIDNAPPING
THEFT
ROBBERY
OTHER OFFENSE
OFFENSE INVOLVING CHILDREN
MOTOR VEHICLE THEFT
INTERFERENCE WITH PUBLIC OFFICER
CRIMINAL TRESPASS
CRIMINAL DAMAGE
BATTERY
DECEPTIVE PRACTICE
BURGLARY
SEX OFFENSE
STALKING
NON - CRIMINAL
CRIM SEXUAL ASSAULT
OBSCENITY
ARSON
CRIMINAL SEXUAL ASSAULT
OTHER NARCOTIC VIOLATION
RITUALISM
Primary_Type

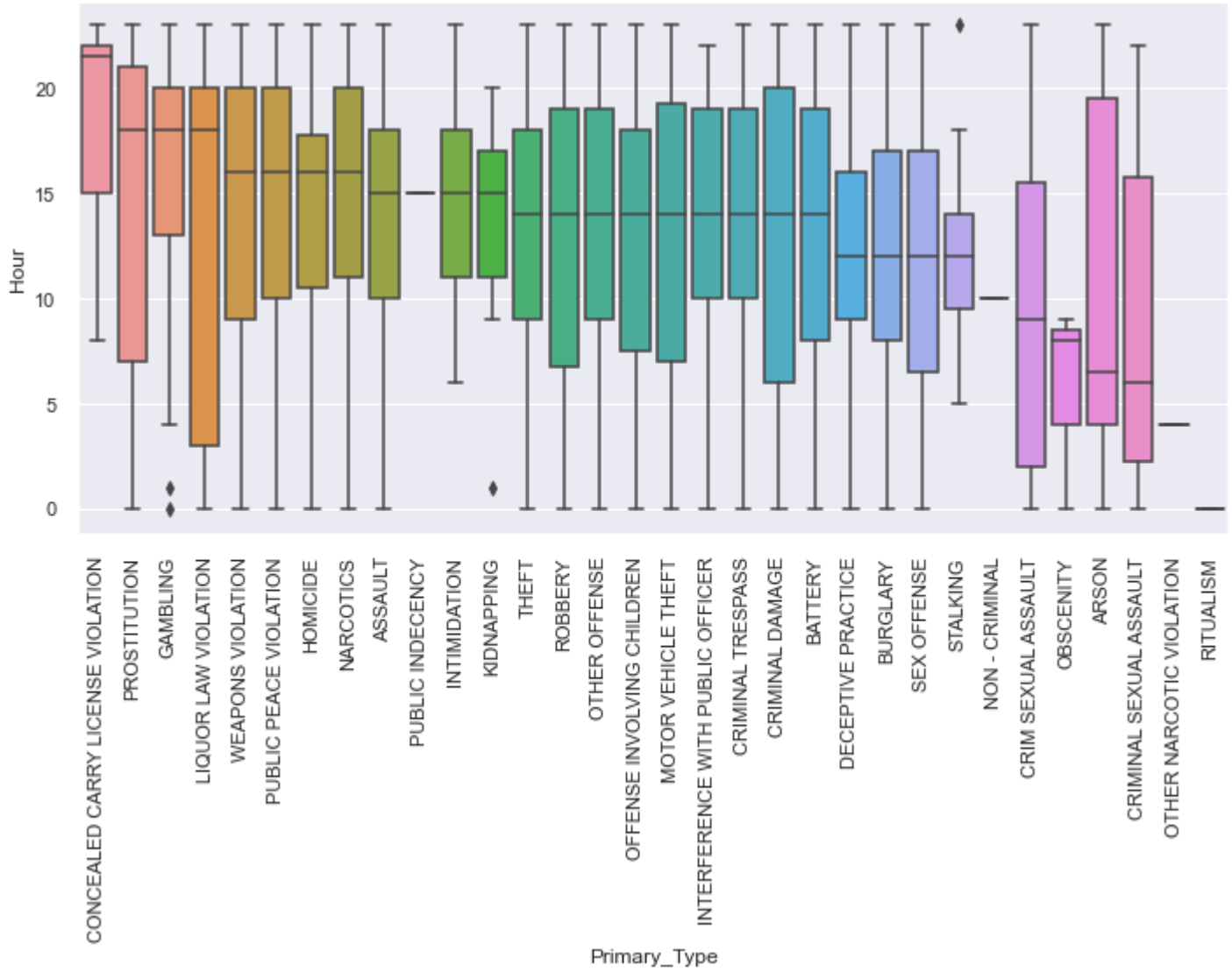

20
15
10
5
0
Hour
Sat
Tue
Fri
Mon
Sun
Wed
Thu
Day

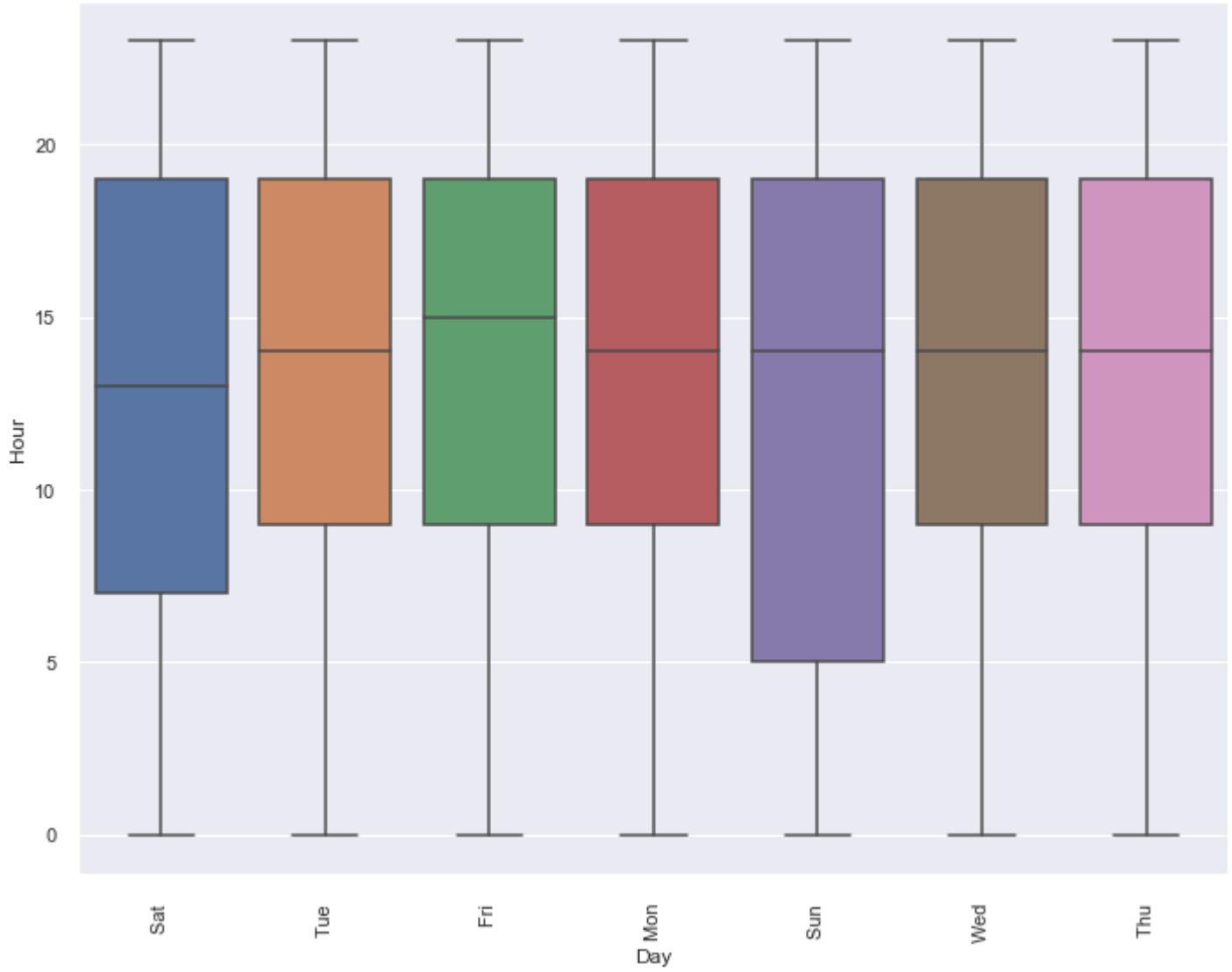


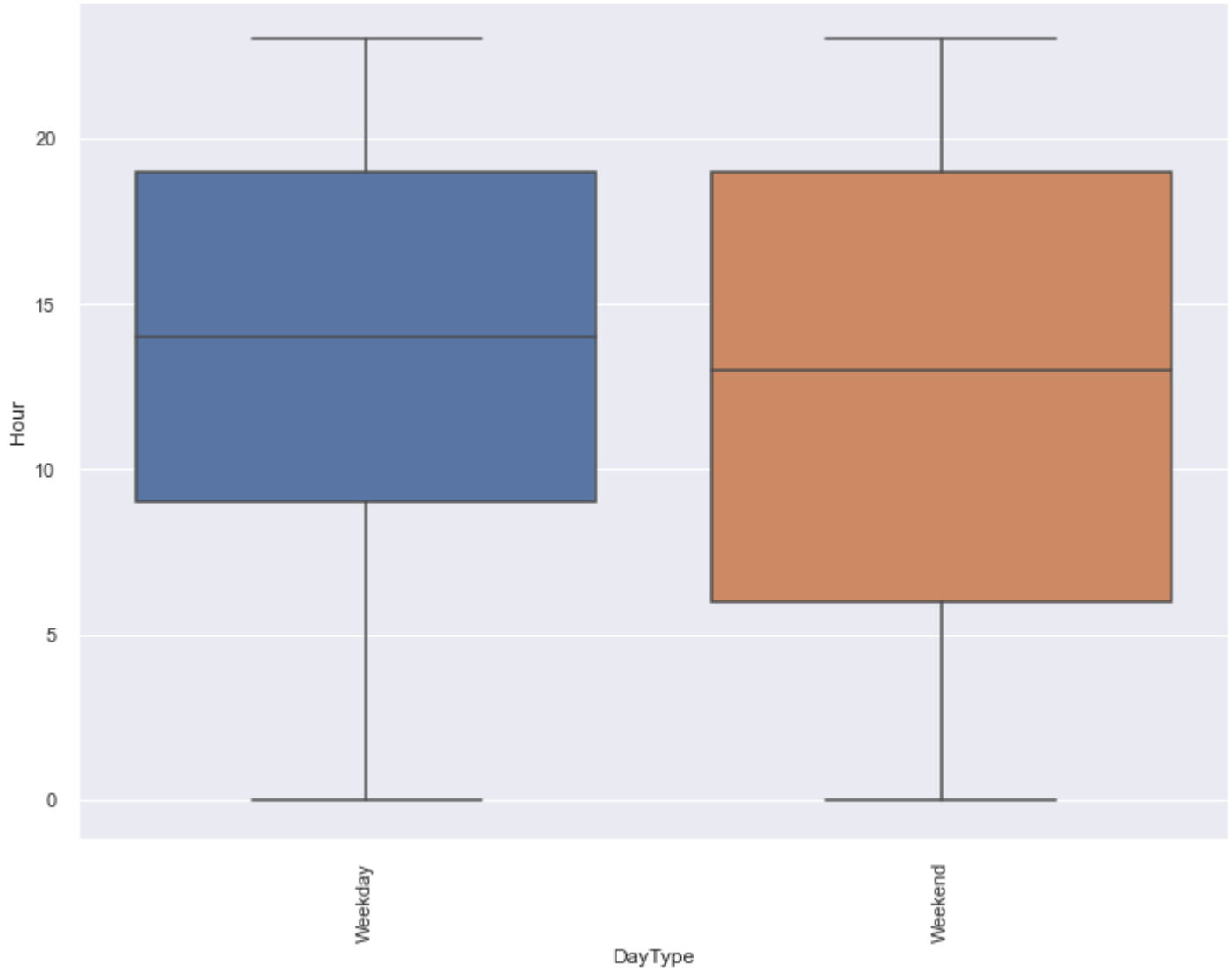
In [151…

```
# Primary type, Day and DayType

plt.subplots(figsize = (10,8))
group = dfFinal.groupby(by='Primary_Type').agg({'Hour': 'median'}).sort_values(by='Hour', ascending=False)
order = group.index
sns.boxplot(data=dfFinal, x='Primary_Type', y='Hour', order=order)
plt.xticks(rotation=90)
plt.tight_layout()
```

```
plt.subplots(figsize = (10,8))
order = ['Monday', 'Tuesday', 'Wednesday', 'Thursday', 'Friday', 'Saturday', 'Sunday']
sns.boxplot(data=dfFinal, x='Day', y='Hour')
plt.xticks(rotation=90)
plt.tight_layout()

plt.subplots(figsize = (10,8))
group = dfFinal.groupby(by='DayType').agg({'Hour': 'median'}).sort_values(by='Hour', ascending=False)
order = group.index
sns.boxplot(data=dfFinal, x='DayType', y='Hour', order=order)
plt.xticks(rotation=90)
plt.tight_layout()


# This shows that crimes occurs/happen/take place early hours on weekends than weekdays.
```


Hour
0
5
10
15
20
CONCEALED CARRY LICENSE VIOLATION
PROSTITUTION
GAMBLING
LIQUOR LAW VIOLATION
WEAPONS VIOLATION
PUBLIC PEACE VIOLATION
HOMICIDE
NARCOTICS
ASSAULT
PUBLIC INDECENCY
INTIMIDATION
KIDNAPPING
THEFT
ROBBERY
OTHER OFFENSE
OFFENSE INVOLVING CHILDREN
MOTOR VEHICLE THEFT
INTERFERENCE WITH PUBLIC OFFICER
CRIMINAL TRESPASS
CRIMINAL DAMAGE
BATTERY
DECEPTIVE PRACTICE
BURGLARY
SEX OFFENSE
STALKING
NON - CRIMINAL
CRIM SEXUAL ASSAULT
OBSCENITY
ARSON
CRIMINAL SEXUAL ASSAULT
OTHER NARCOTIC VIOLATION
RITUALISM
Primary_Type

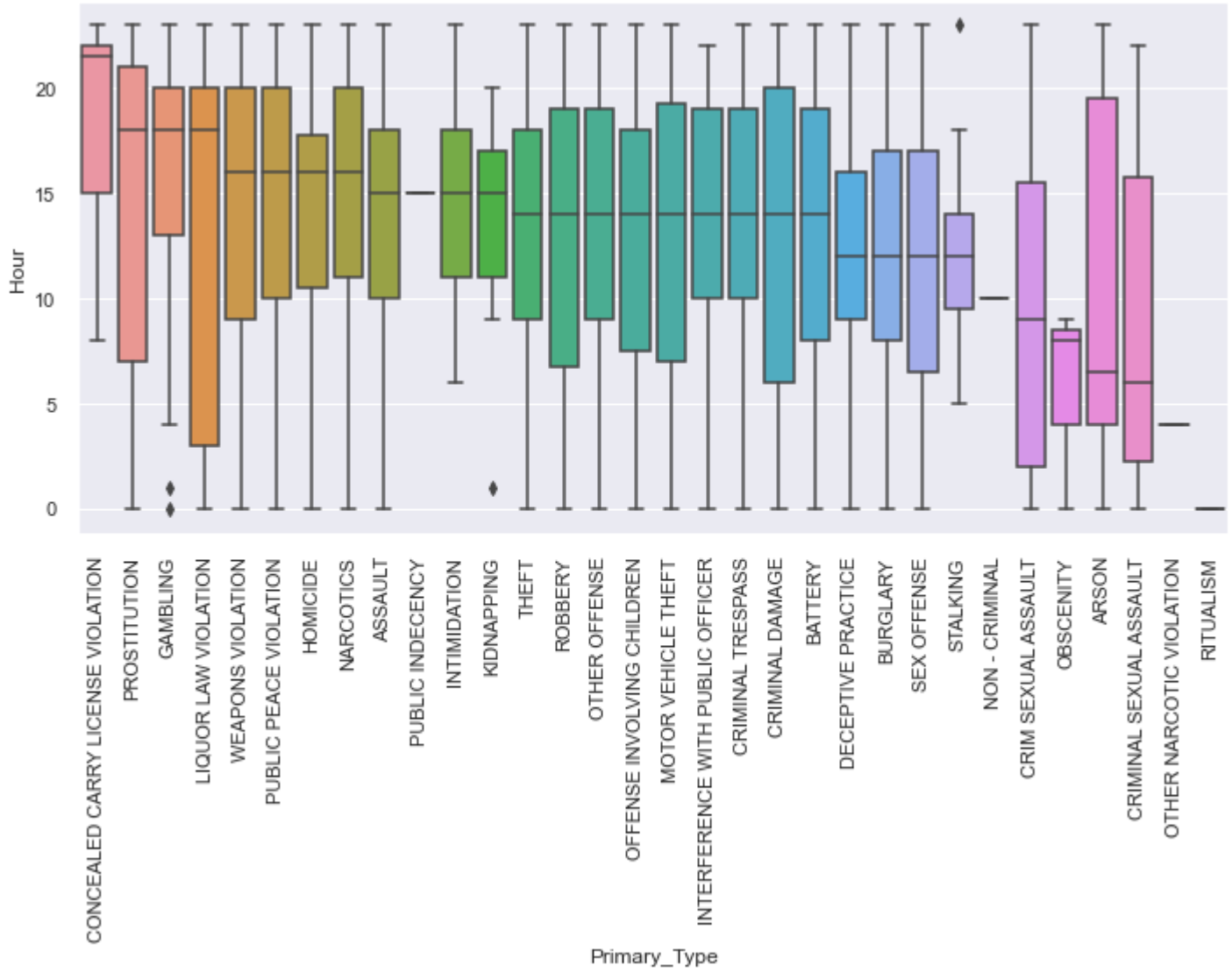

20
15
10
5
0
Hour
Sat
Tue
Fri
Mon
Sun
Wed
Thu
Day

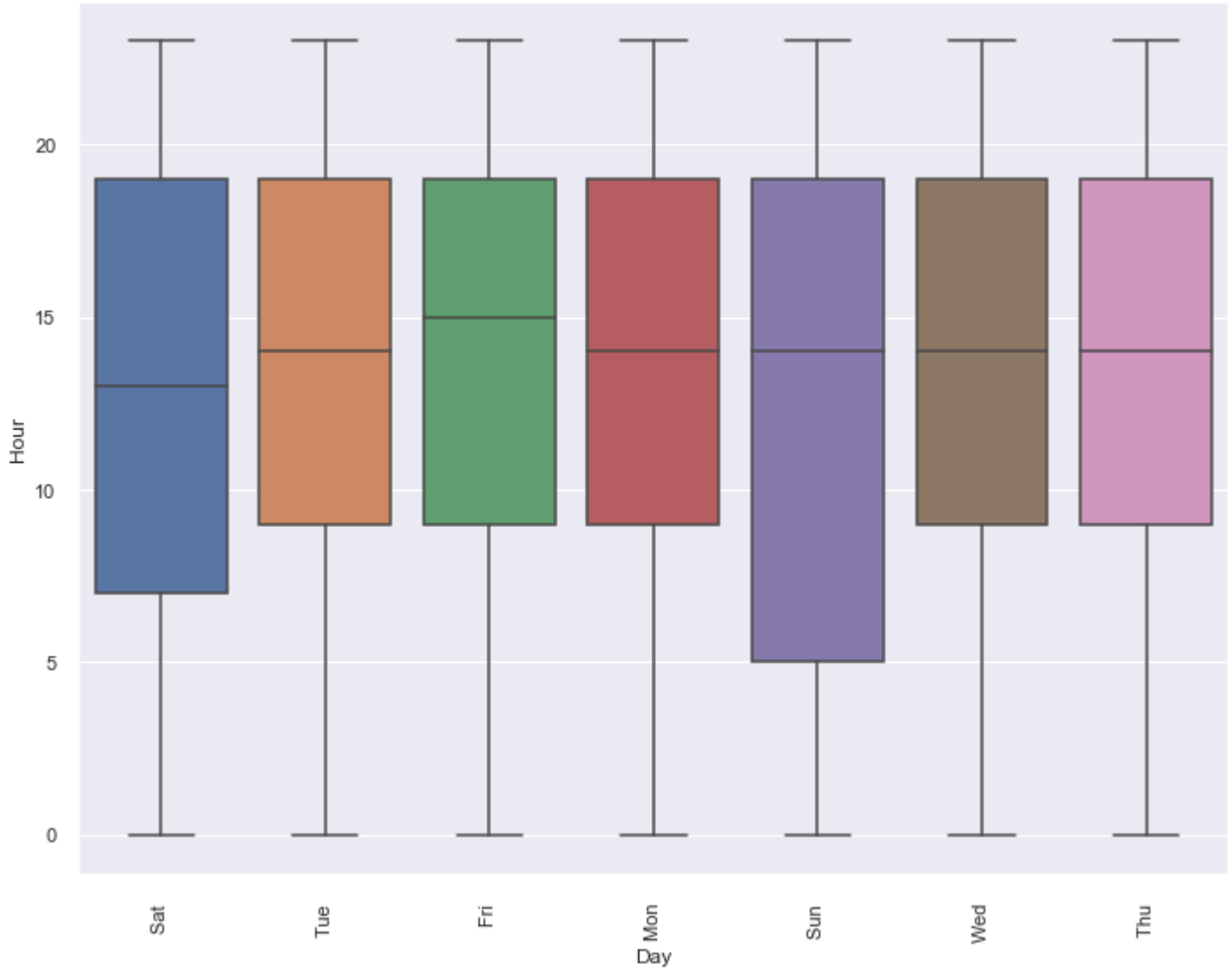


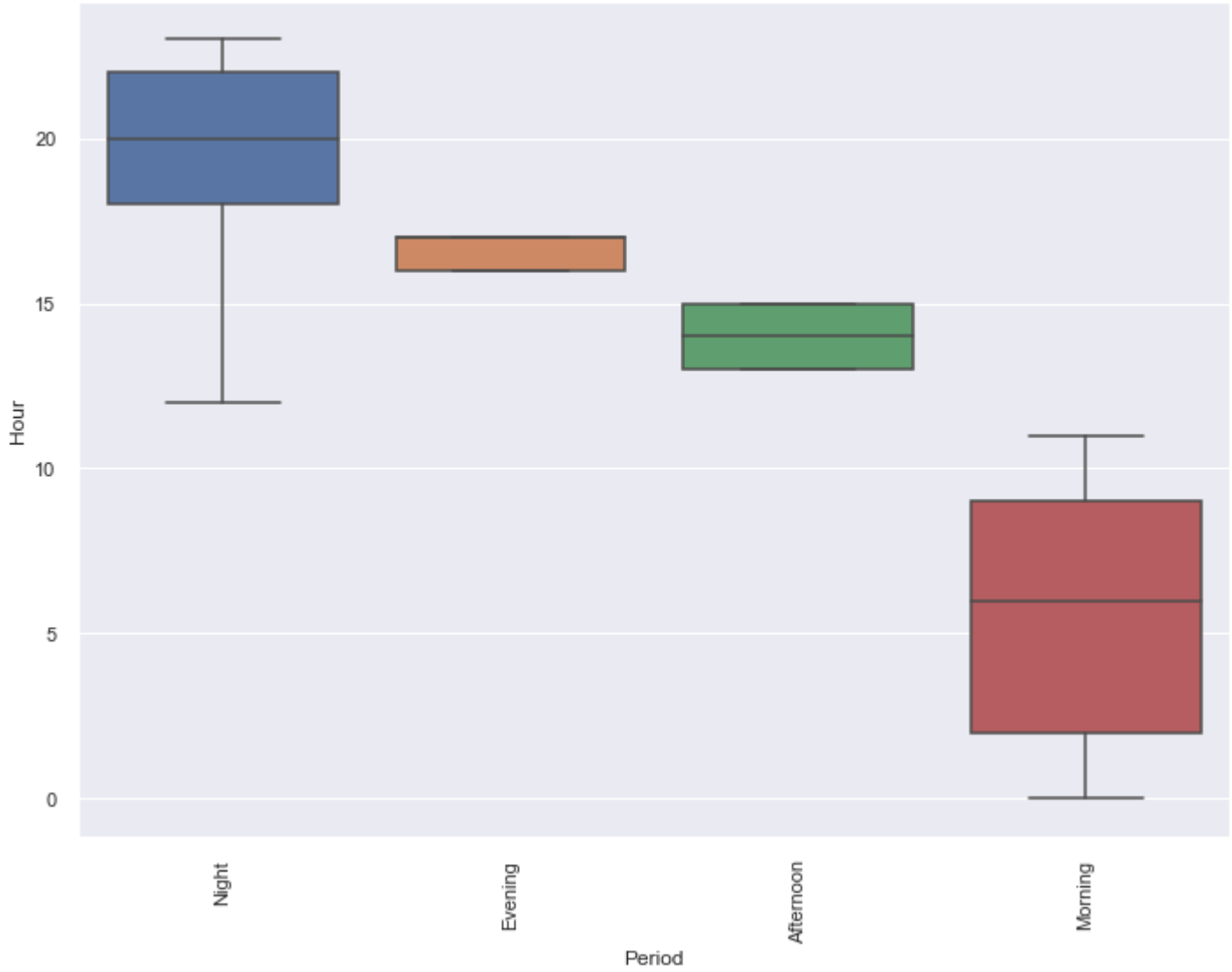
In [152…

```
# Primary type, Day and Period


plt.subplots(figsize = (10,8))
group = dfFinal.groupby(by='Primary_Type').agg({'Hour': 'median'}).sort_values(by='Hour', ascending=False)
order = group.index
sns.boxplot(data=dfFinal, x='Primary_Type', y='Hour', order=order)
plt.xticks(rotation=90)
plt.tight_layout()
```

```
plt.subplots(figsize = (10,8))
order = ['Monday', 'Tuesday', 'Wednesday', 'Thursday', 'Friday', 'Saturday', 'Sunday']
sns.boxplot(data=dfFinal, x='Day', y='Hour')
plt.xticks(rotation=90)
plt.tight_layout()

plt.subplots(figsize = (10,8))
group = dfFinal.groupby(by='Period').agg({'Hour': 'median'}).sort_values(by='Hour', ascending=False)
order = group.index
sns.boxplot(data=dfFinal, x='Period', y='Hour', order=order)
plt.xticks(rotation=90)
plt.tight_layout()



# Observations
# - Prostitution is concentrated in the evening
# - Public indecency is quite narrow around noon; BUT: Also small sample size
# - Most categories cover the complete data range with their 25%-75% quantile.
# - On weekends crime starts earlier



# This confirms that crimes tends to happen at the early hours on weekends than weekday
# considering Saturday and Sunday in the Second chart.
```


Hour
0
5
10
15
20
CONCEALED CARRY LICENSE VIOLATION
PROSTITUTION
GAMBLING
LIQUOR LAW VIOLATION
WEAPONS VIOLATION
PUBLIC PEACE VIOLATION
HOMICIDE
NARCOTICS
ASSAULT
PUBLIC INDECENCY
INTIMIDATION
KIDNAPPING
THEFT
ROBBERY
OTHER OFFENSE
OFFENSE INVOLVING CHILDREN
MOTOR VEHICLE THEFT
INTERFERENCE WITH PUBLIC OFFICER
CRIMINAL TRESPASS
CRIMINAL DAMAGE
BATTERY
DECEPTIVE PRACTICE
BURGLARY
SEX OFFENSE
STALKING
NON - CRIMINAL
CRIM SEXUAL ASSAULT
OBSCENITY
ARSON
CRIMINAL SEXUAL ASSAULT
OTHER NARCOTIC VIOLATION
RITUALISM
Primary_Type

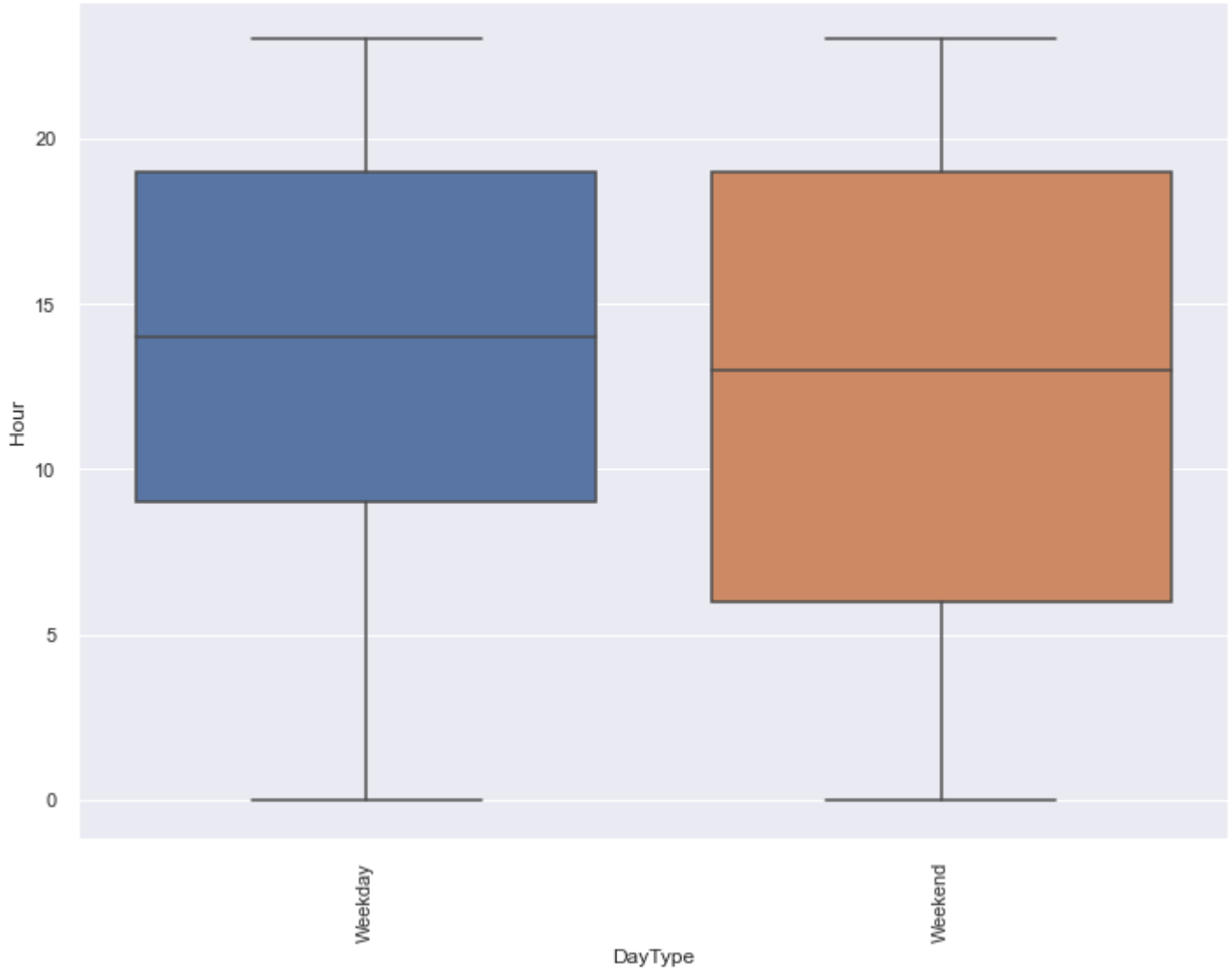

20
15
10
5
0
Hour
Sat
Tue
Fri
Mon
Sun
Wed
Thu
Day

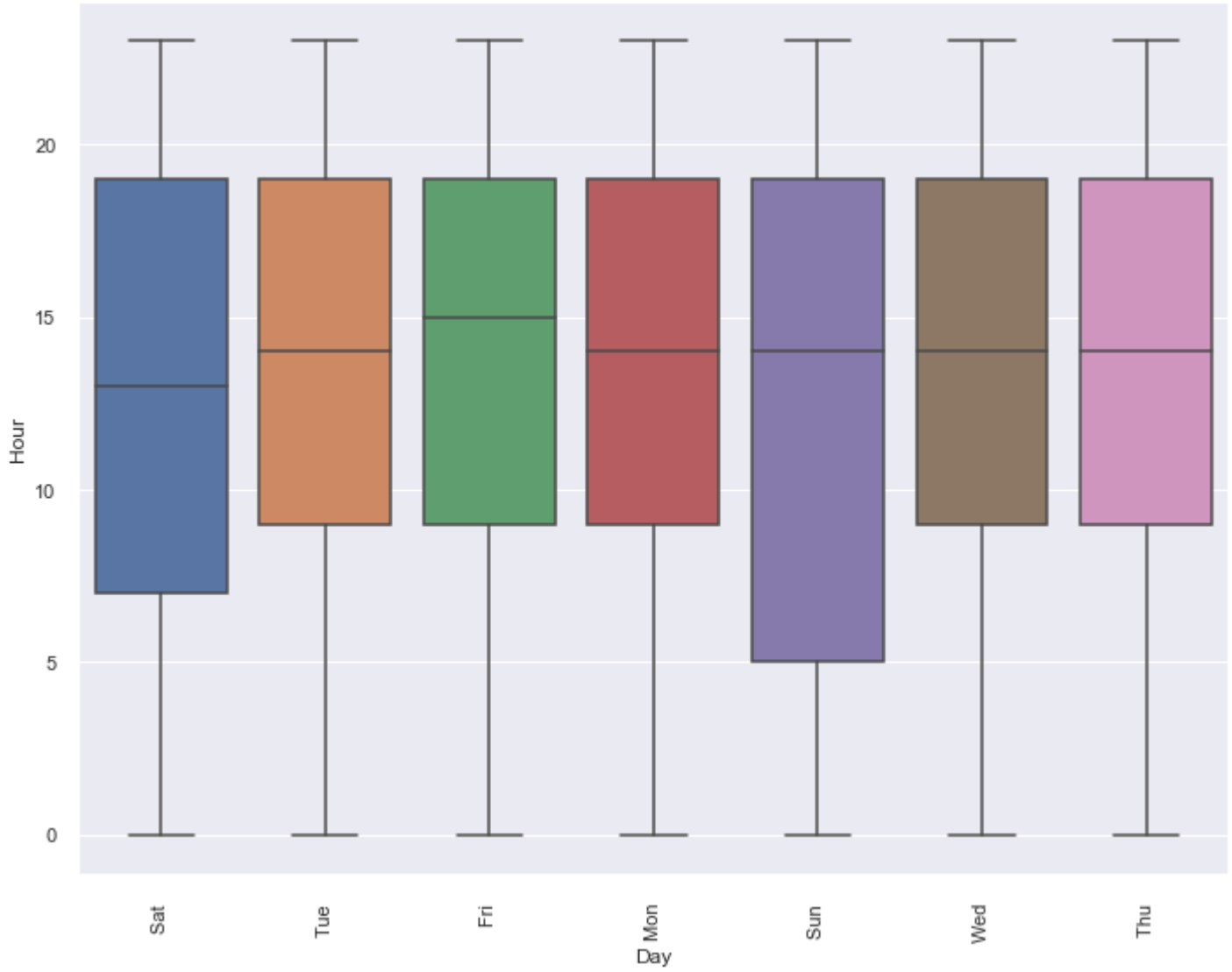


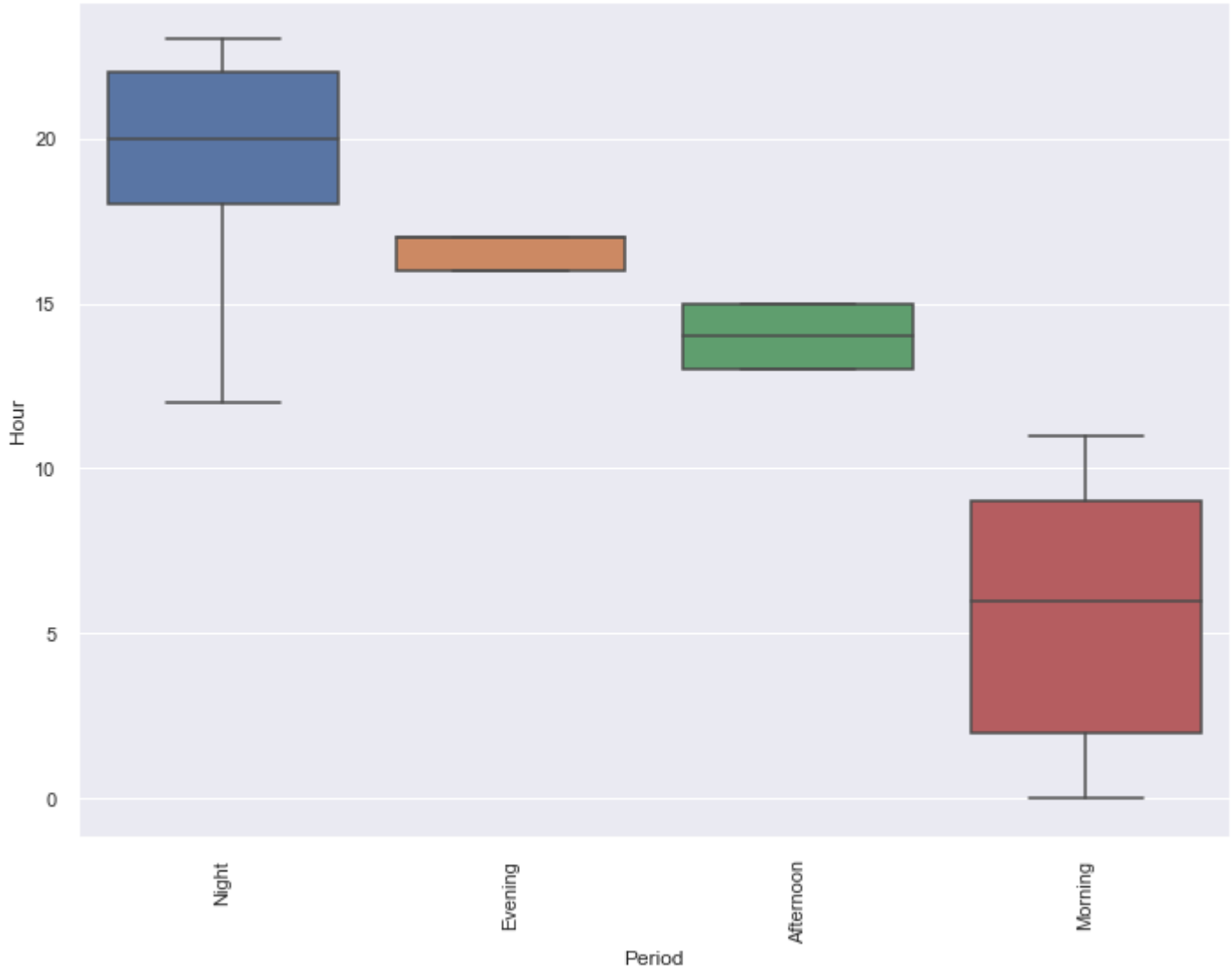
In [153…

```
# Daytype, Day and Period



plt.subplots(figsize = (10,8))
group = dfFinal.groupby(by='DayType').agg({'Hour': 'median'}).sort_values(by='Hour', ascending=False)
order = group.index
sns.boxplot(data=dfFinal, x='DayType', y='Hour', order=order)
plt.xticks(rotation=90)
```

```
plt.tight_layout()

plt.subplots(figsize = (10,8))
order = ['Monday', 'Tuesday', 'Wednesday', 'Thursday', 'Friday', 'Saturday', 'Sunday']
sns.boxplot(data=dfFinal, x='Day', y='Hour')
plt.xticks(rotation=90)
plt.tight_layout()

plt.subplots(figsize = (10,8))
group = dfFinal.groupby(by='Period').agg({'Hour': 'median'}).sort_values(by='Hour', ascending=False)
order = group.index
sns.boxplot(data=dfFinal, x='Period', y='Hour', order=order)
plt.xticks(rotation=90)
plt.tight_layout()
```


20
15
10
5
0
Hour
Weekday
Weekend
DayType

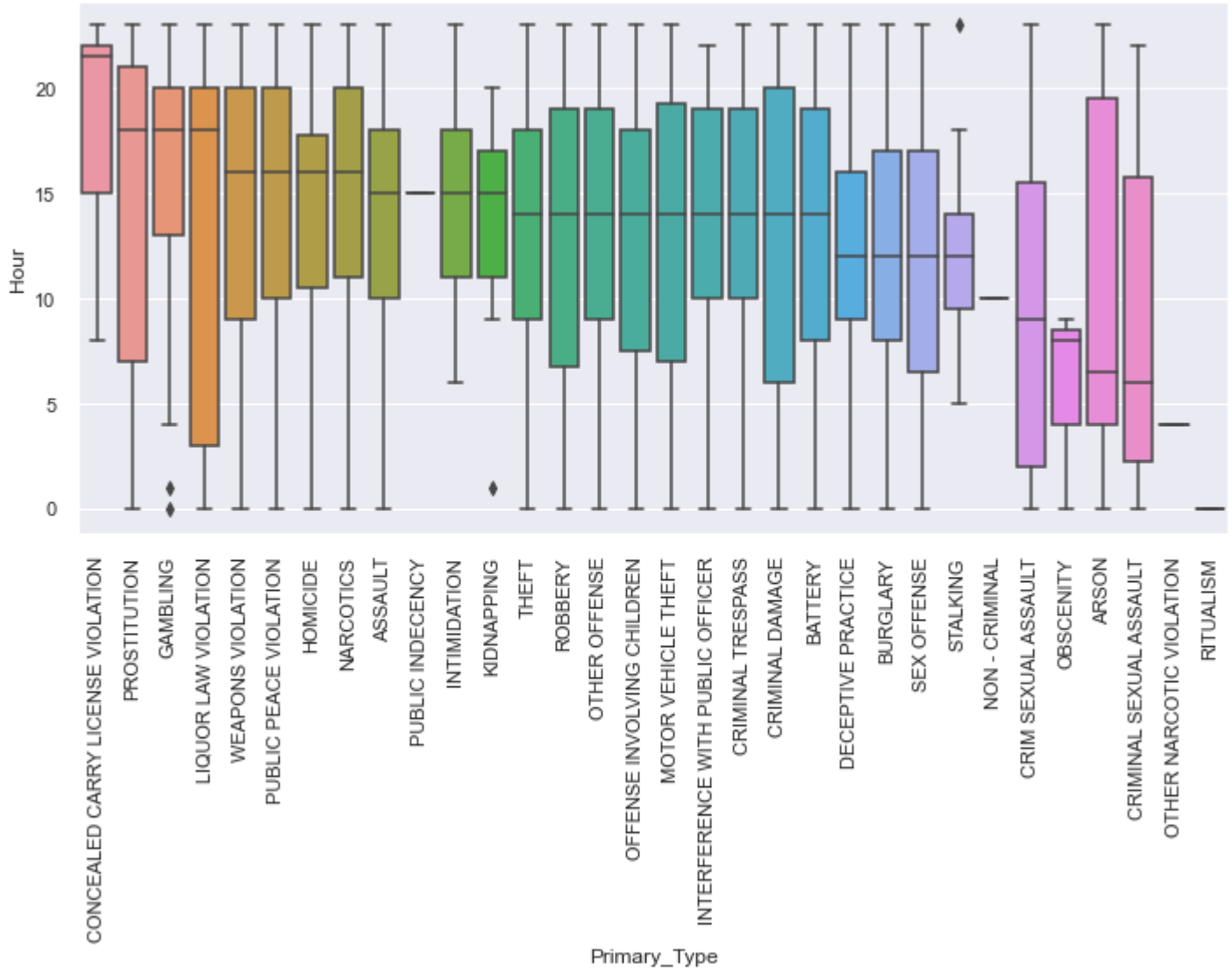

20
15
10
5
0
Hour
Sat
Tue
Fri
Mon
Sun
Wed
Thu
Day

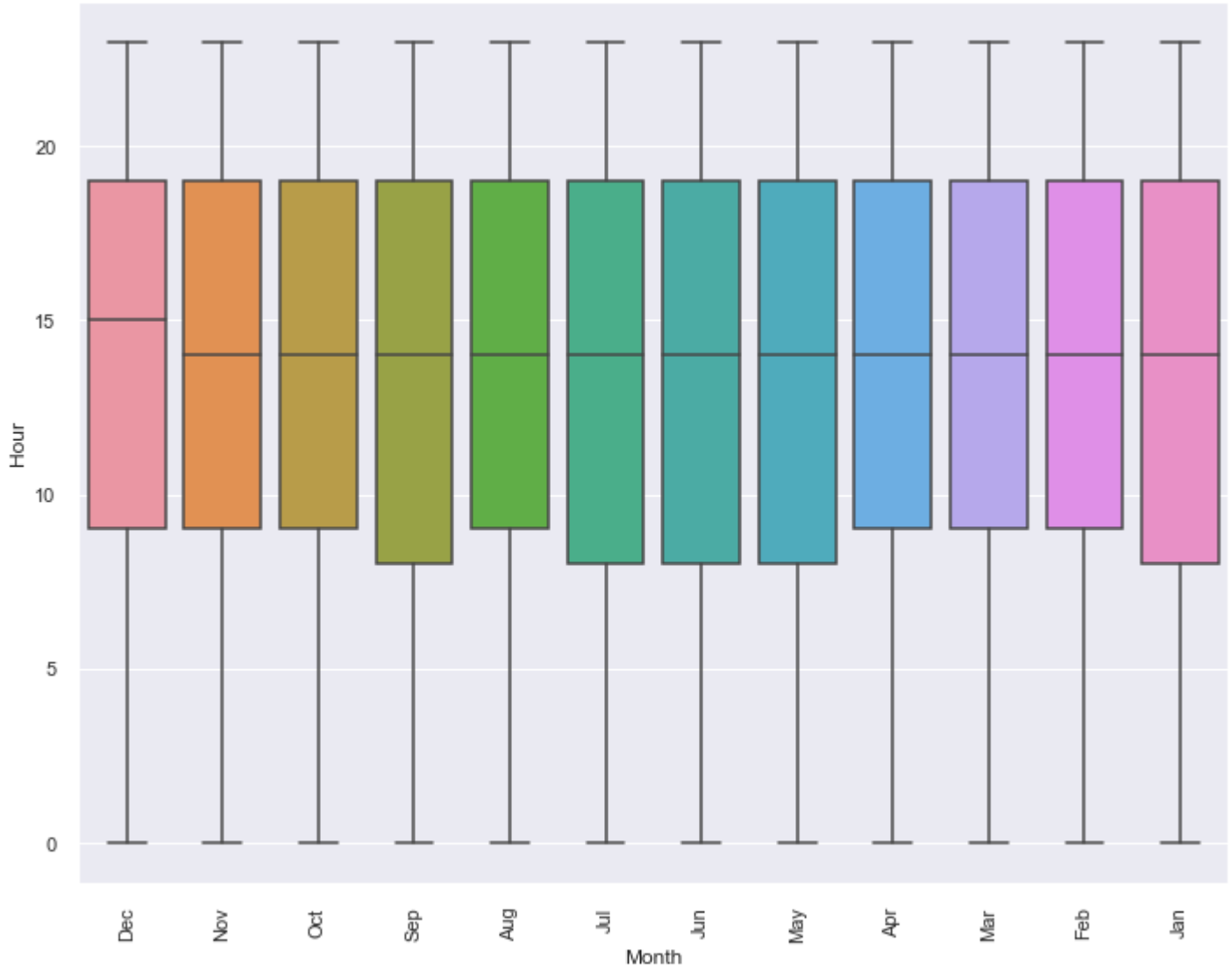


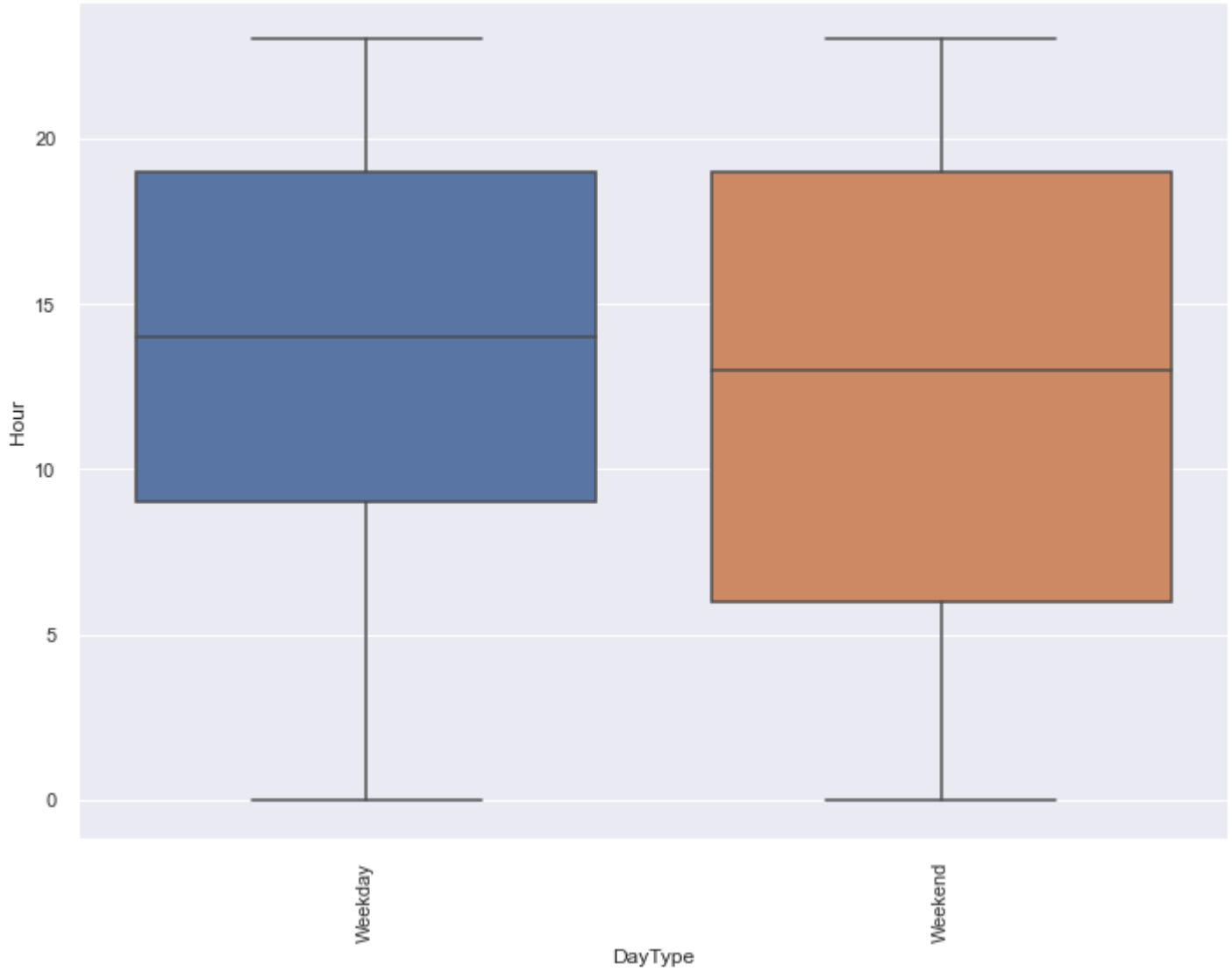
In [154…

```
# Primary type, Month and DayType



plt.subplots(figsize = (10,8))
group = dfFinal.groupby(by='Primary_Type').agg({'Hour': 'median'}).sort_values(by='Hour', ascending=False)
order = group.index
sns.boxplot(data=dfFinal, x='Primary_Type', y='Hour', order=order)
plt.xticks(rotation=90)
```

```
plt.tight_layout()

plt.subplots(figsize = (10,8))
order = ['Dec', 'Nov', 'Oct', 'Sep', 'Aug', 'Jul', 'Jun', 'May', 'Apr', 'Mar', 'Feb', 'Jan']
sns.boxplot(data=dfFinal, x='Month', y='Hour', order=order)
plt.xticks(rotation=90)
plt.tight_layout()

plt.subplots(figsize = (10,8))
group = dfFinal.groupby(by='DayType').agg({'Hour': 'median'}).sort_values(by='Hour', ascending=False)
order = group.index
sns.boxplot(data=dfFinal, x='DayType', y='Hour', order=order)
plt.xticks(rotation=90)
plt.tight_layout()



# This confirms that crimes tends to happen at the early hours on weekends than weekday
```


Hour
0
5
10
15
20
CONCEALED CARRY LICENSE VIOLATION
PROSTITUTION
GAMBLING
LIQUOR LAW VIOLATION
WEAPONS VIOLATION
PUBLIC PEACE VIOLATION
HOMICIDE
NARCOTICS
ASSAULT
PUBLIC INDECENCY
INTIMIDATION
KIDNAPPING
THEFT
ROBBERY
OTHER OFFENSE
OFFENSE INVOLVING CHILDREN
MOTOR VEHICLE THEFT
INTERFERENCE WITH PUBLIC OFFICER
CRIMINAL TRESPASS
CRIMINAL DAMAGE
BATTERY
DECEPTIVE PRACTICE
BURGLARY
SEX OFFENSE
STALKING
NON - CRIMINAL
CRIM SEXUAL ASSAULT
OBSCENITY
ARSON
CRIMINAL SEXUAL ASSAULT
OTHER NARCOTIC VIOLATION
RITUALISM
Primary_Type

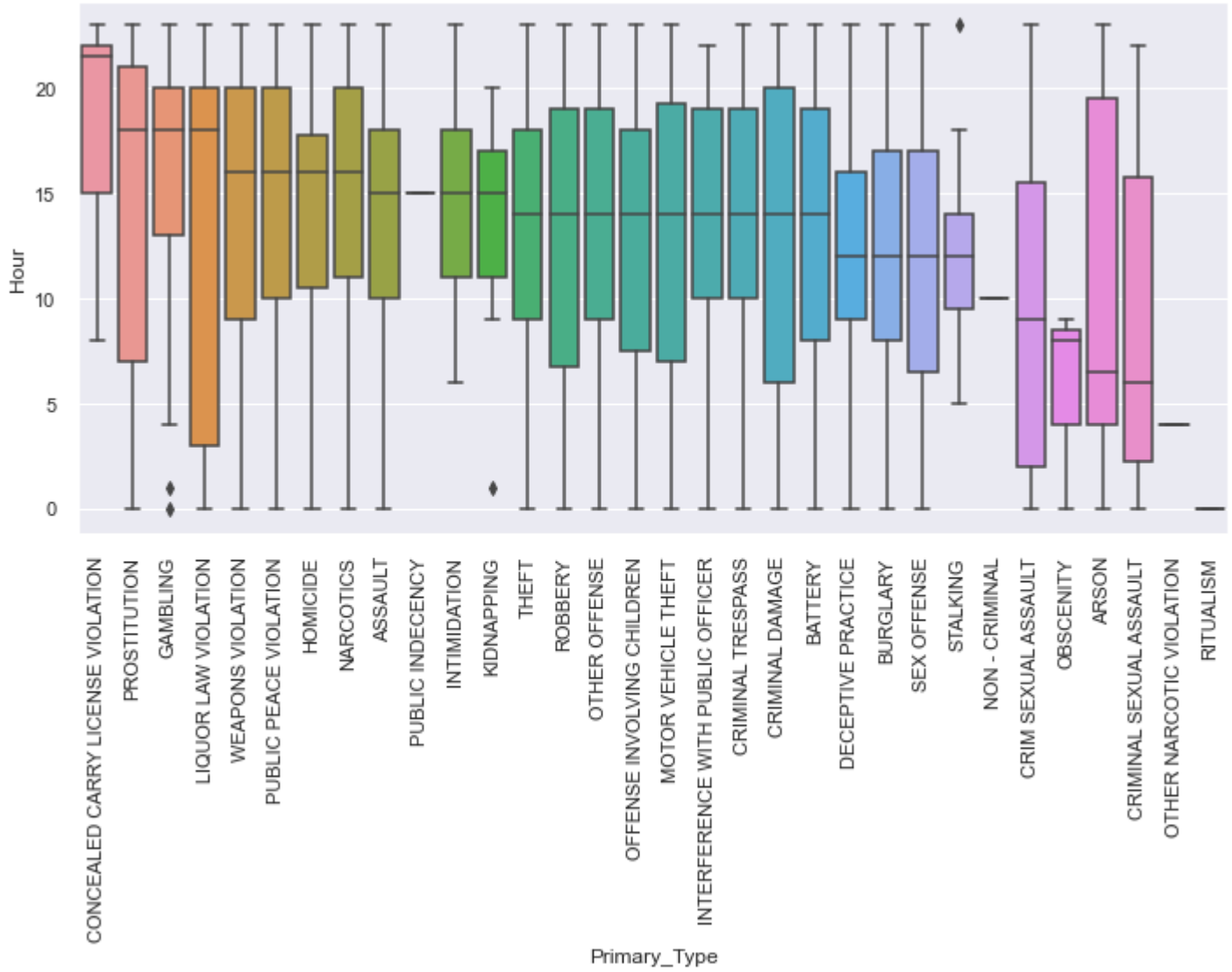

Hour
20
15
10
5
0
Dec
Nov
Oct
Sep
Aug
Jul
Jun
May
Apr
Mar
Feb
Jan
Month

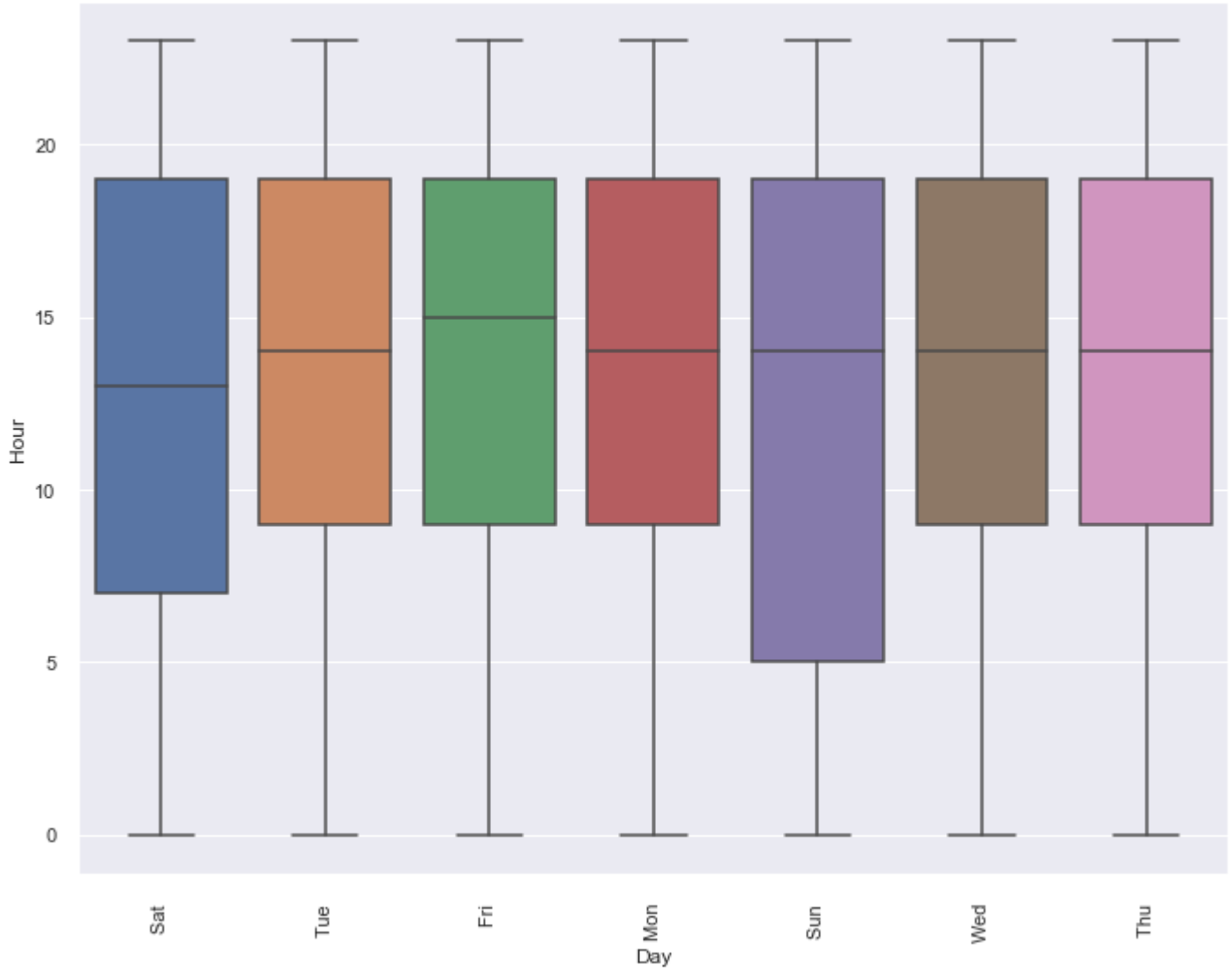


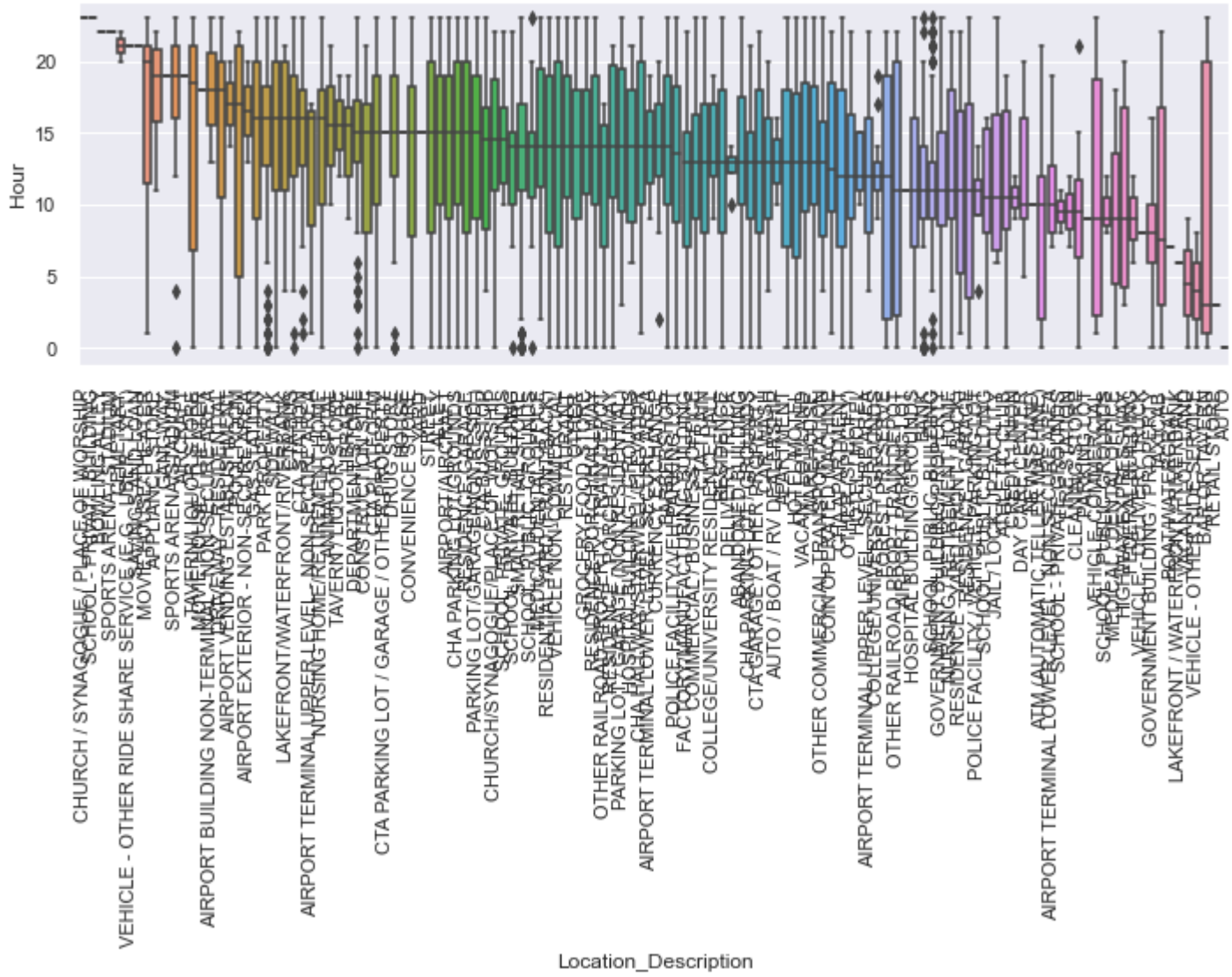
In [155…

```
# Primary type, Day and Location_Description

plt.subplots(figsize = (10,8))
group = dfFinal.groupby(by='Primary_Type').agg({'Hour': 'median'}).sort_values(by='Hour', ascending=False)
order = group.index
sns.boxplot(data=dfFinal, x='Primary_Type', y='Hour', order=order)
plt.xticks(rotation=90)
plt.tight_layout()
```

```
plt.subplots(figsize = (10,8))
order = ['Monday', 'Tuesday', 'Wednesday', 'Thursday', 'Friday', 'Saturday', 'Sunday']
sns.boxplot(data=dfFinal, x='Day', y='Hour')
plt.xticks(rotation=90)
plt.tight_layout()

plt.subplots(figsize = (10,8))
group = dfFinal.groupby(by='Location_Description').agg({'Hour': 'median'}).sort_values(by='Hour', ascending=False)
order = group.index
sns.boxplot(data=dfFinal, x='Location_Description', y='Hour', order=order)
plt.xticks(rotation=90)
plt.tight_layout()



# Observations
# - Prostitution is concentrated in the evening
# - Public indecency is quite narrow around noon; BUT: Also small sample size
# - Most categories cover the complete data range with their 25%-75% quantile.
# - On weekends crime starts earlier
```


Hour
0
5
10
15
20
CONCEALED CARRY LICENSE VIOLATION
PROSTITUTION
GAMBLING
LIQUOR LAW VIOLATION
WEAPONS VIOLATION
PUBLIC PEACE VIOLATION
HOMICIDE
NARCOTICS
ASSAULT
PUBLIC INDECENCY
INTIMIDATION
KIDNAPPING
THEFT
ROBBERY
OTHER OFFENSE
OFFENSE INVOLVING CHILDREN
MOTOR VEHICLE THEFT
INTERFERENCE WITH PUBLIC OFFICER
CRIMINAL TRESPASS
CRIMINAL DAMAGE
BATTERY
DECEPTIVE PRACTICE
BURGLARY
SEX OFFENSE
STALKING
NON - CRIMINAL
CRIM SEXUAL ASSAULT
OBSCENITY
ARSON
CRIMINAL SEXUAL ASSAULT
OTHER NARCOTIC VIOLATION
RITUALISM
Primary_Type

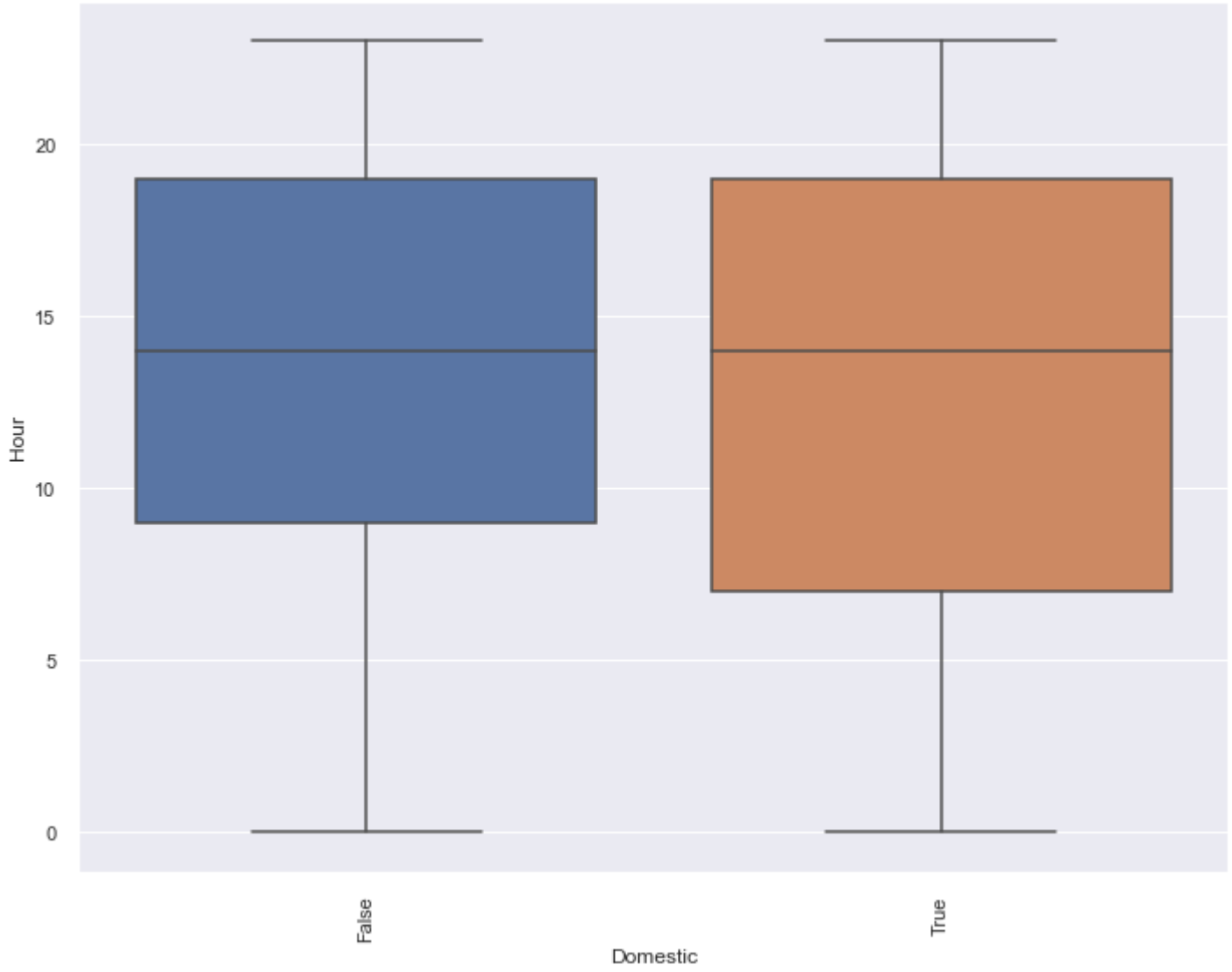

20
15
10
5
0
Hour
Sat
Tue
Fri
Mon
Sun
Wed
Thu
Day

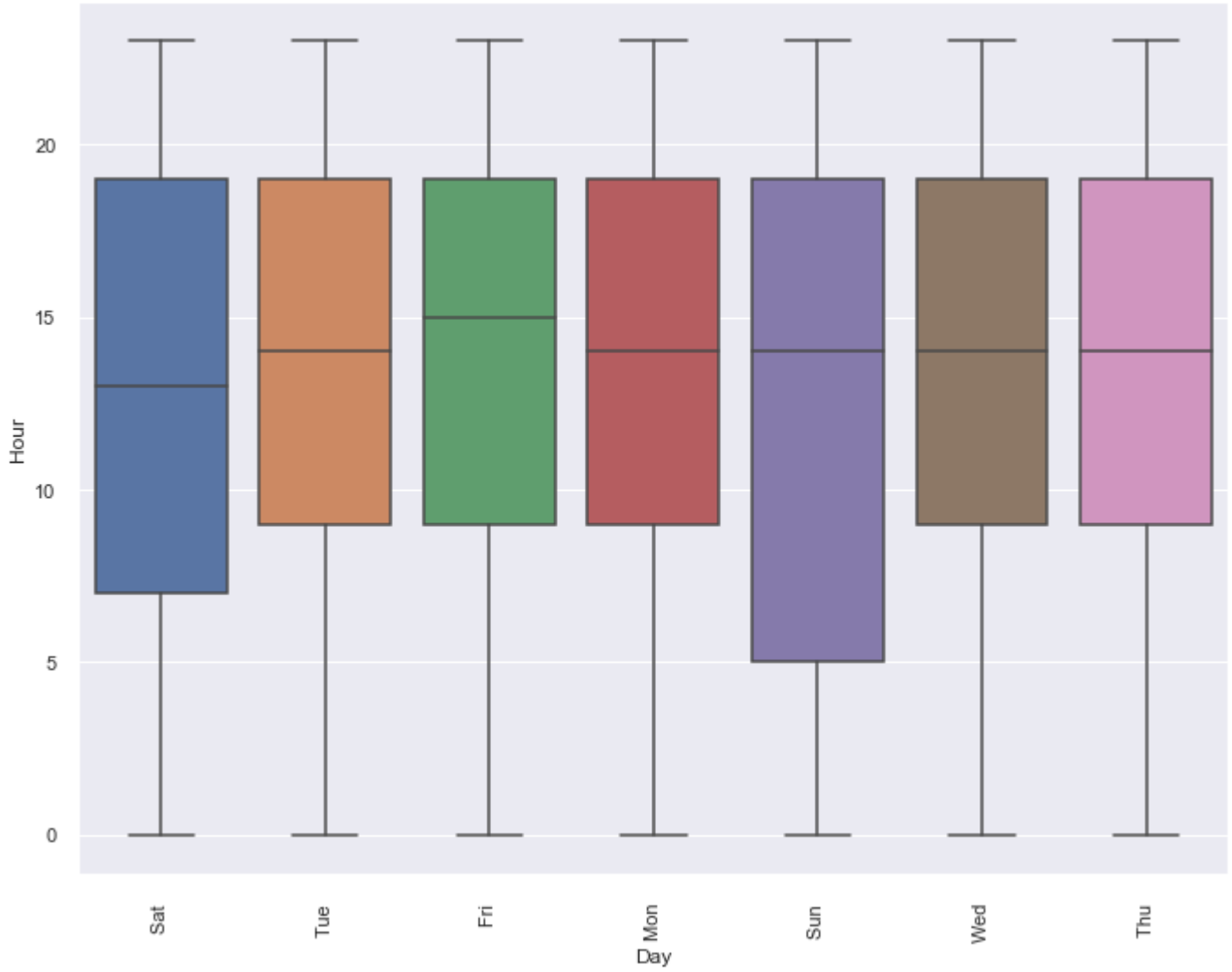


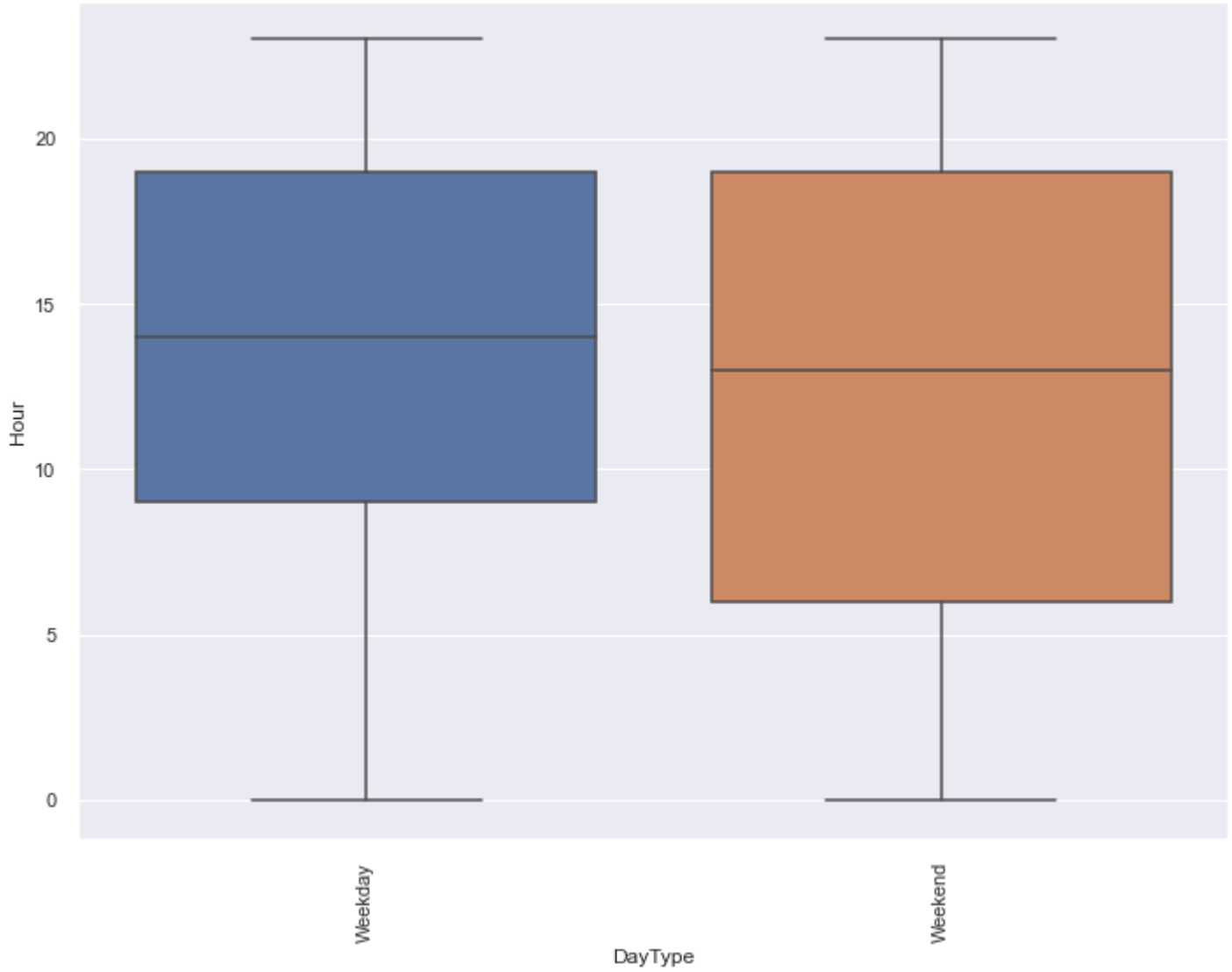
In [156…

```
# Domestic, type, Day and Period

plt.subplots(figsize = (10,8))
group = dfFinal.groupby(by='Domestic').agg({'Hour': 'median'}).sort_values(by='Hour', ascending=False)
order = group.index
sns.boxplot(data=dfFinal, x='Domestic', y='Hour', order=order)
plt.xticks(rotation=90)
plt.tight_layout()
```

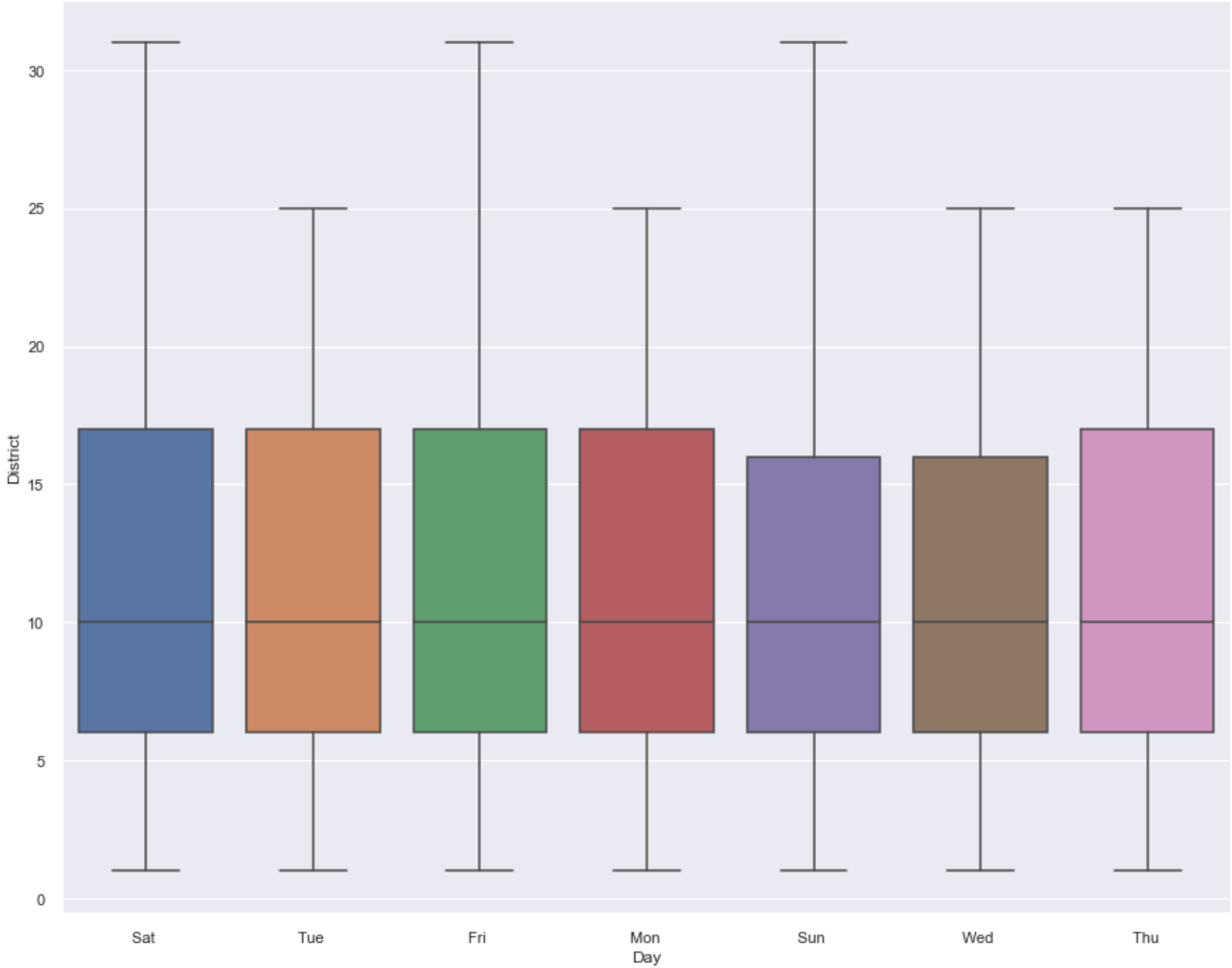
```
plt.subplots(figsize = (10,8))
order = ['Monday', 'Tuesday', 'Wednesday', 'Thursday', 'Friday', 'Saturday', 'Sunday']
sns.boxplot(data=dfFinal, x='Day', y='Hour')
plt.xticks(rotation=90)
plt.tight_layout()

plt.subplots(figsize = (10,8))
group = dfFinal.groupby(by='DayType').agg({'Hour': 'median'}).sort_values(by='Hour', ascending=False)
order = group.index
sns.boxplot(data=dfFinal, x='DayType', y='Hour', order=order)
plt.xticks(rotation=90)
plt.tight_layout()
```


20
15
10
5
0
Hour
False
True
Domestic


20
15
10
5
0
Hour
Sat
Tue
Fri
Mon
Sun
Wed
Thu
Day

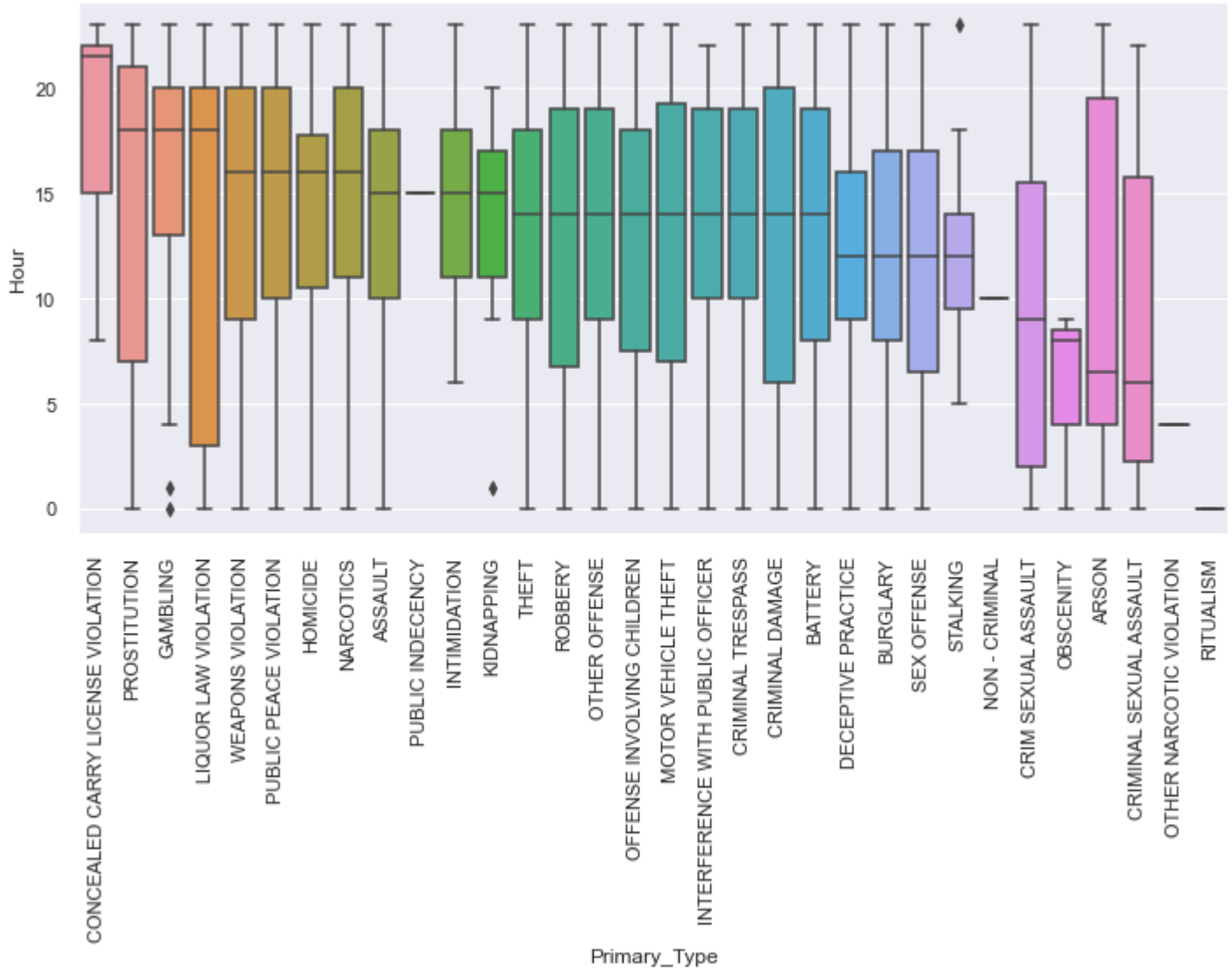


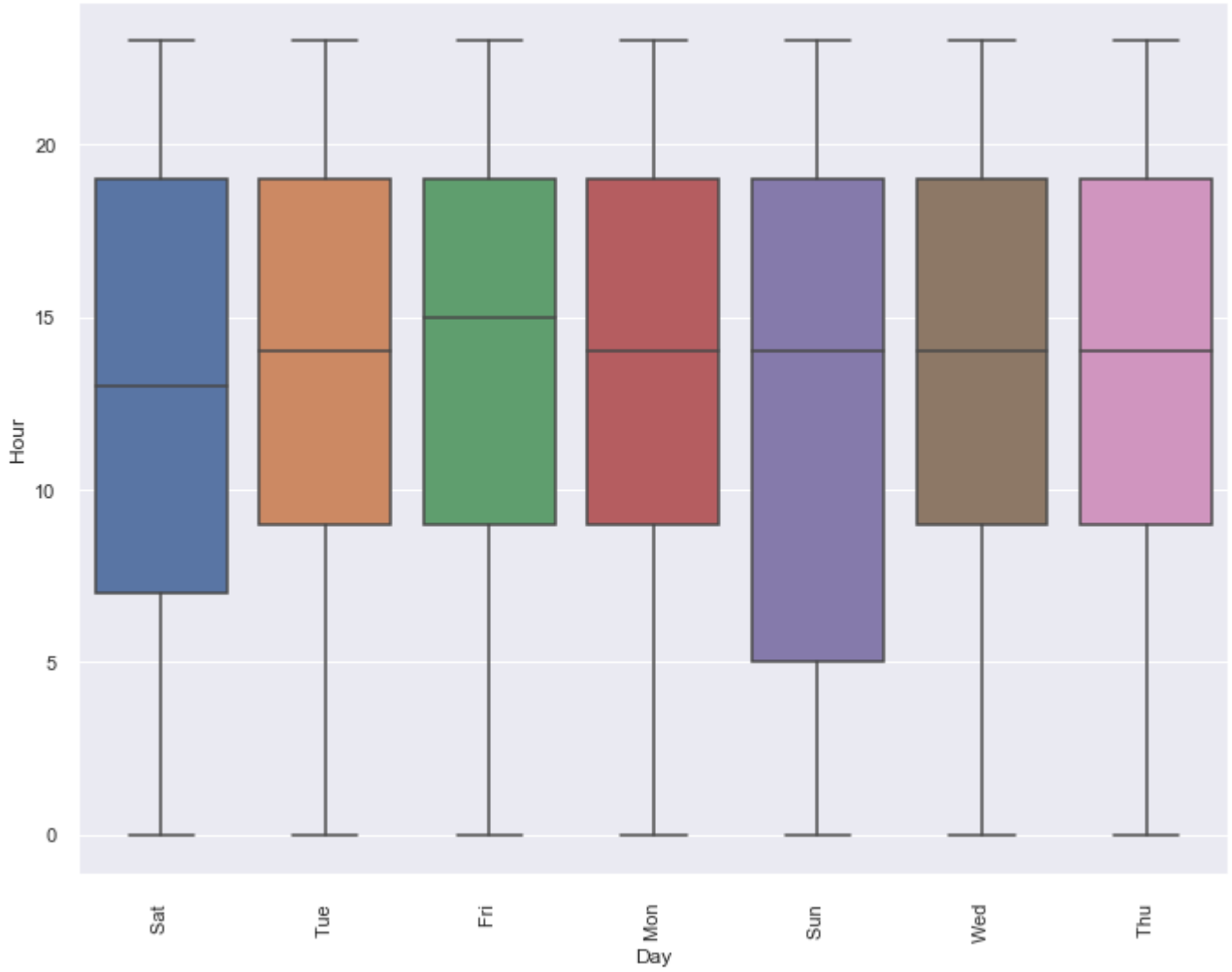
In [157…
```
#****
sns.boxplot(x='Day', y='District', data=dfFinal)
plt.show()
```

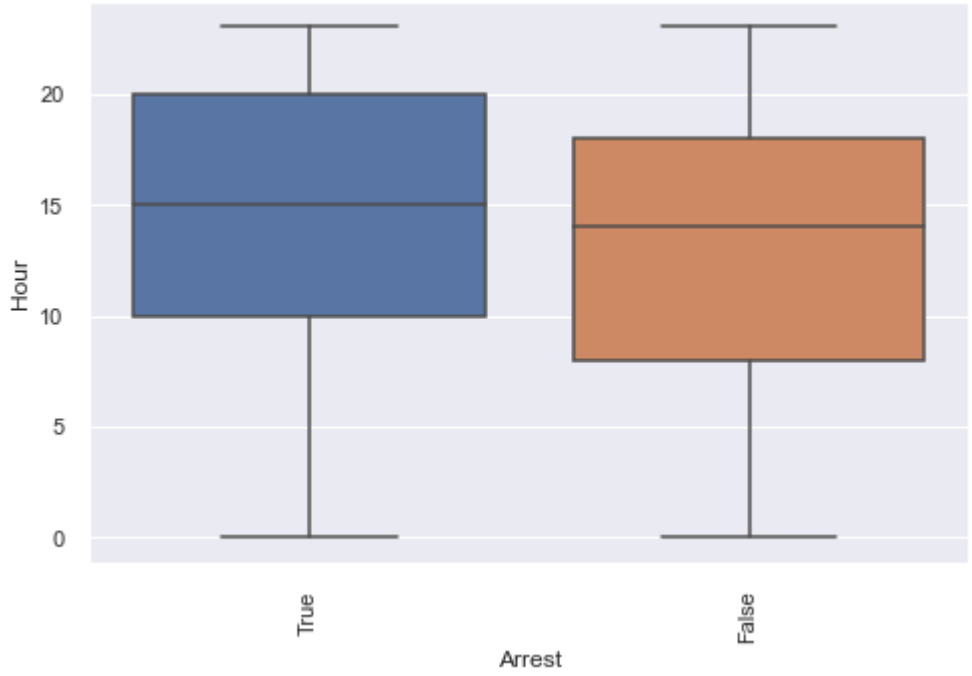


In [158…
```
# Arrest, type, Day and Arrest
```

```
plt.subplots(figsize = (10,8))
group = dfFinal.groupby(by='Primary_Type').agg({'Hour': 'median'}).sort_values(by='Hour', ascending=False)
order = group.index
sns.boxplot(data=dfFinal, x='Primary_Type', y='Hour', order=order)
plt.xticks(rotation=90)
plt.tight_layout()

plt.subplots(figsize = (10,8))
order = ['Monday', 'Tuesday', 'Wednesday', 'Thursday', 'Friday', 'Saturday', 'Sunday']
sns.boxplot(data=dfFinal, x='Day', y='Hour')
plt.xticks(rotation=90)
plt.tight_layout()

plt.subplots(figsize = (7,5))
group = dfFinal.groupby(by='Arrest').agg({'Hour': 'median'}).sort_values(by='Hour', ascending=False)
order = group.index
sns.boxplot(data=dfFinal, x='Arrest', y='Hour', order=order)
plt.xticks(rotation=90)
plt.tight_layout()
```

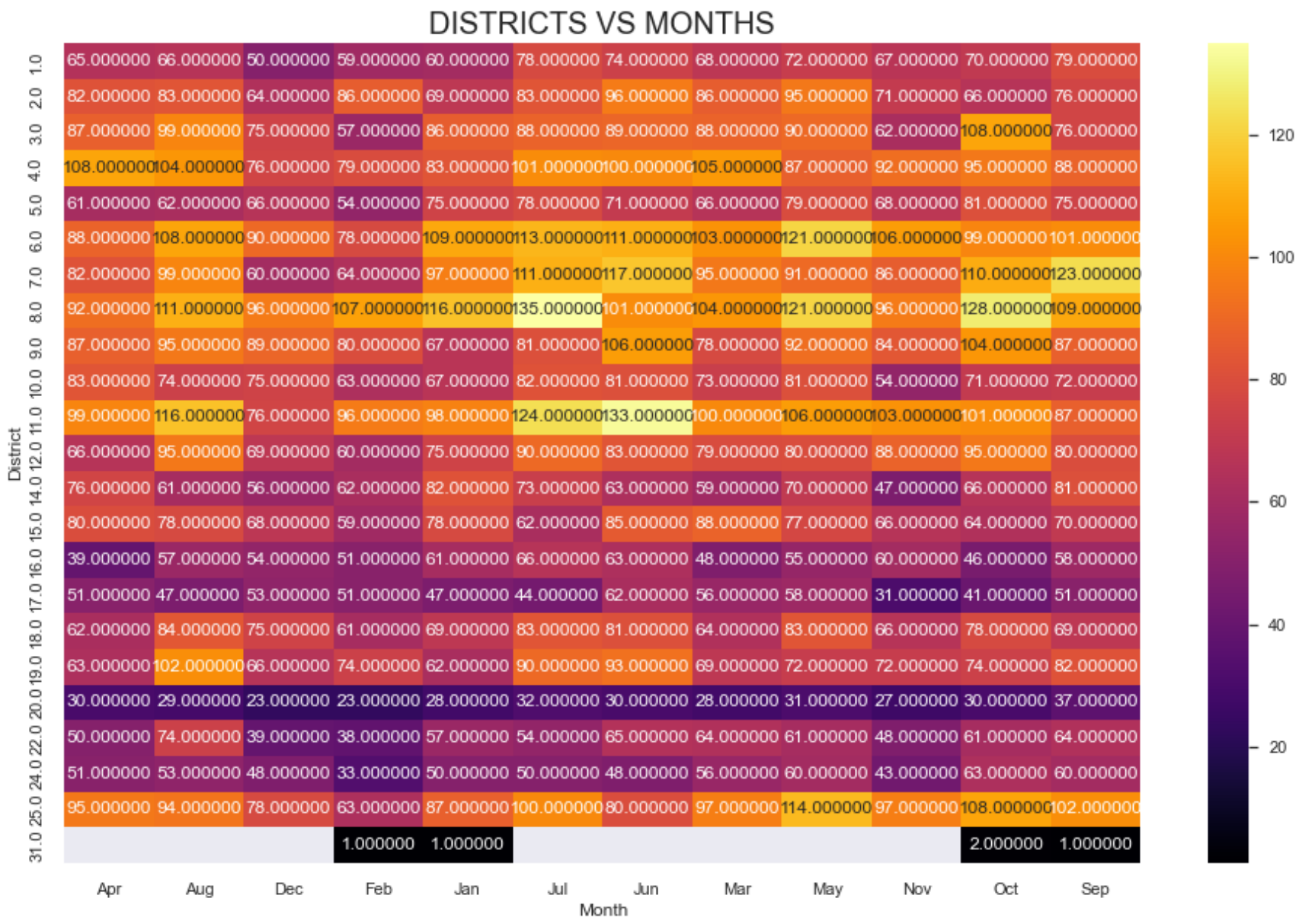

Hour
0
5
10
15
20
CONCEALED CARRY LICENSE VIOLATION
PROSTITUTION
GAMBLING
LIQUOR LAW VIOLATION
WEAPONS VIOLATION
PUBLIC PEACE VIOLATION
HOMICIDE
NARCOTICS
ASSAULT
PUBLIC INDECENCY
INTIMIDATION
KIDNAPPING
THEFT
ROBBERY
OTHER OFFENSE
OFFENSE INVOLVING CHILDREN
MOTOR VEHICLE THEFT
INTERFERENCE WITH PUBLIC OFFICER
CRIMINAL TRESPASS
CRIMINAL DAMAGE
BATTERY
DECEPTIVE PRACTICE
BURGLARY
SEX OFFENSE
STALKING
NON - CRIMINAL
CRIM SEXUAL ASSAULT
OBSCENITY
ARSON
CRIMINAL SEXUAL ASSAULT
OTHER NARCOTIC VIOLATION
RITUALISM
Primary_Type


20
15
10
5
0
Hour
Sat
Tue
Fri
Mon
Sun
Wed
Thu
Day



In [159…

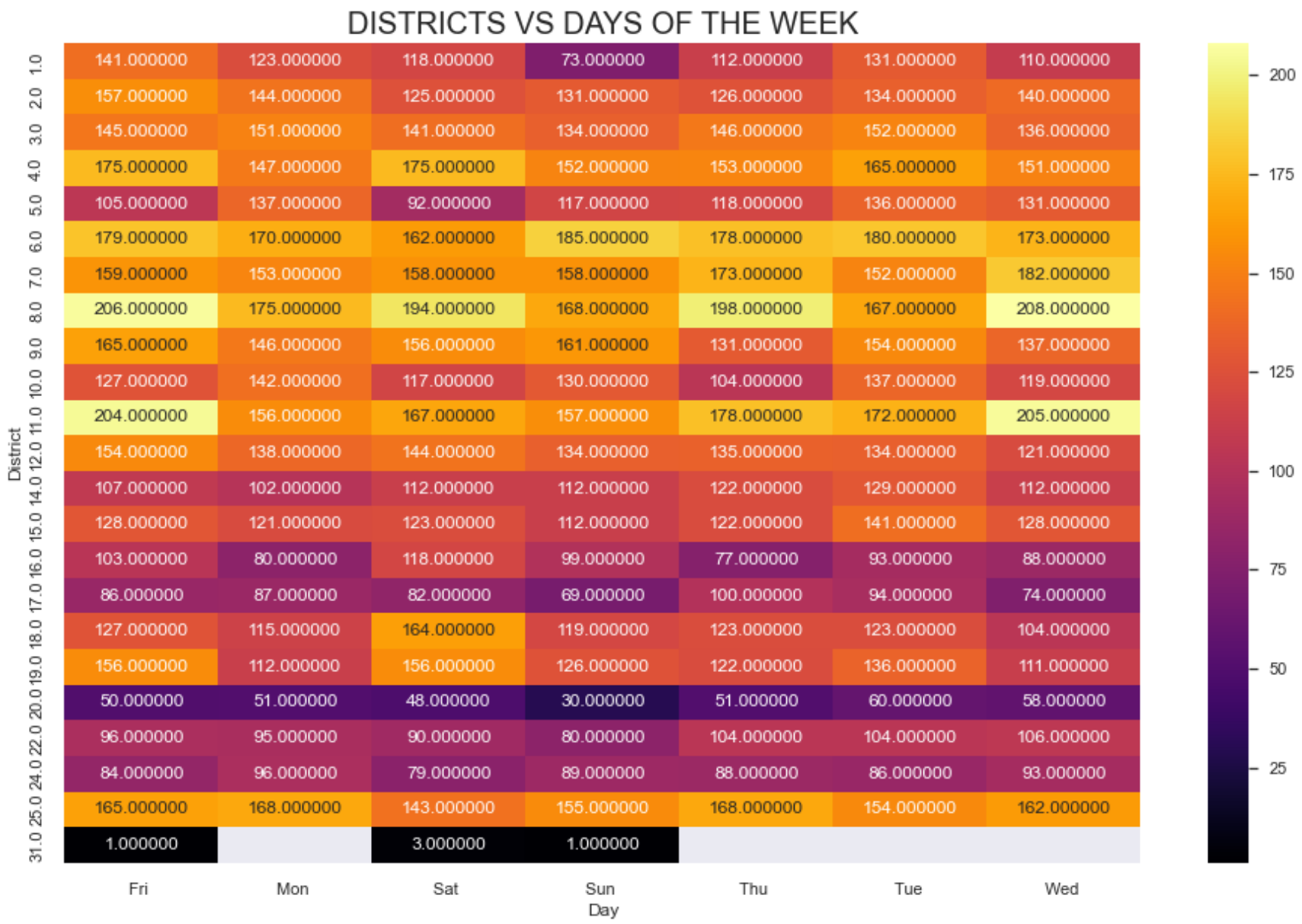
```
# Plotting for District vs Month in crimes
cri3 = dfFinal.groupby(['District','Month'],as_index=False).agg({'Primary_Type':"count"})
cri3.columns

cri3 = cri3.pivot("District", "Month", "Primary_Type")


plt.figure(figsize = (16,10))
plt.title("DISTRICTS VS MONTHS", fontsize=20)
with sns.axes_style("white"):
    sns.heatmap(cri3, mask=cri3.isnull(), cmap="inferno", annot=True, fmt="f")


# District 8.0 has its monthly crime rate PEAK in: July, October and May respectively.
# District 11.0 has its monthly crime rate PEAK in: June, July and August respectively
```



In [160…

```
# Plotting for District vs Day in crimes
cri3 = dfFinal.groupby(['District','Day'],as_index=False).agg({'Primary_Type':"count"})
cri3.columns

cri3 = cri3.pivot("District", "Day", "Primary_Type")
```

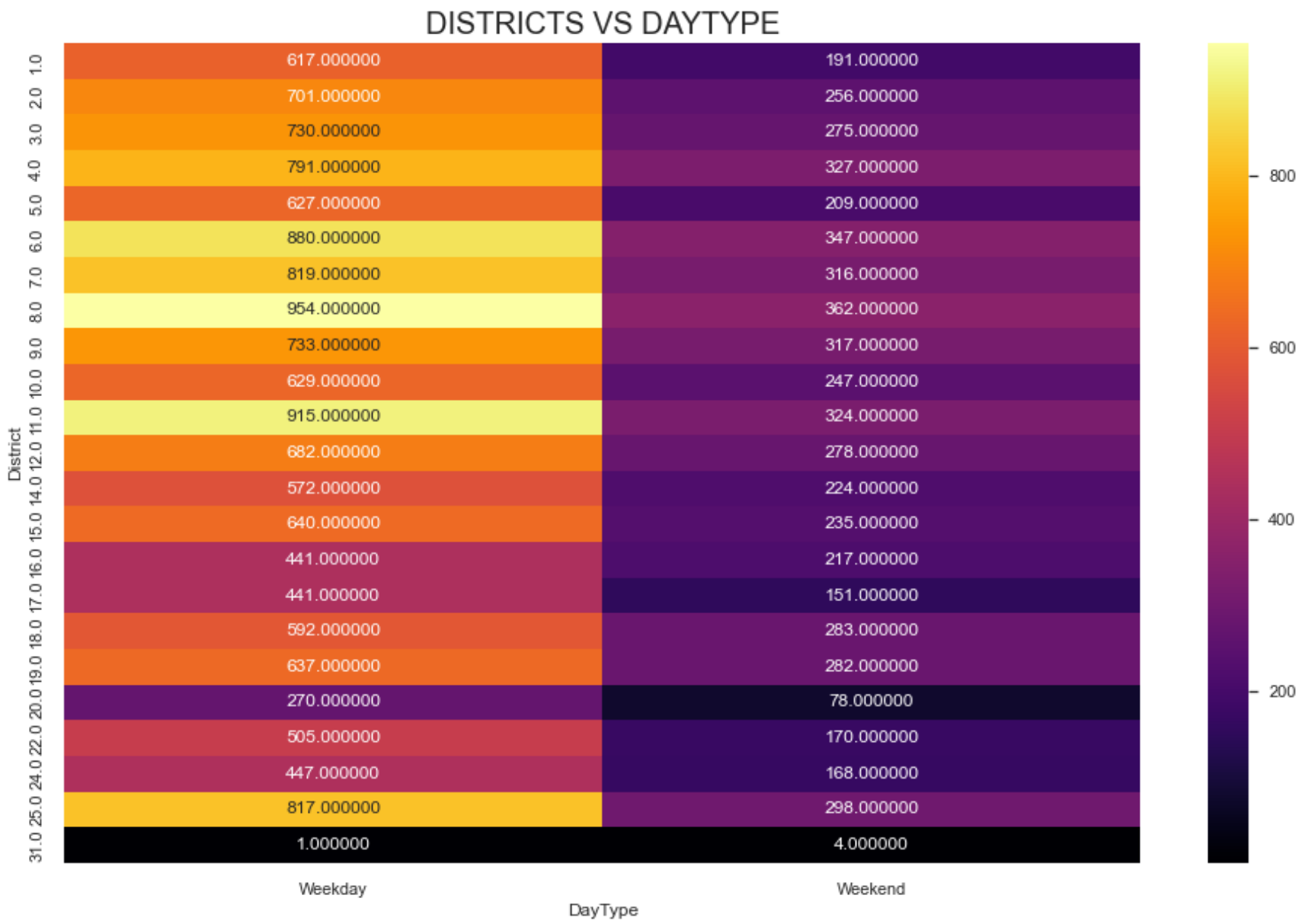
```
plt.figure(figsize = (16,10))
plt.title("DISTRICTS VS DAYS OF THE WEEK", fontsize=20)
with sns.axes_style("white"):
    sns.heatmap(cri3, mask=cri3.isnull(), cmap="inferno", annot=True, fmt="f")


# District 8.0 has its weekly crime rate PEAK on: Wednesdays, Fridays and Thursdays respectively. While,
# District 11.0 has its weekly crime rate PEAK in: Wednesdays and Fridays respectively.
```



In [161…

```
# Plotting for District vs DayTYPE in crimes
cri3 = dfFinal.groupby(['District','DayType'],as_index=False).agg({'Primary_Type':"count"})
cri3.columns

cri3 = cri3.pivot("District", "DayType", "Primary_Type")
```

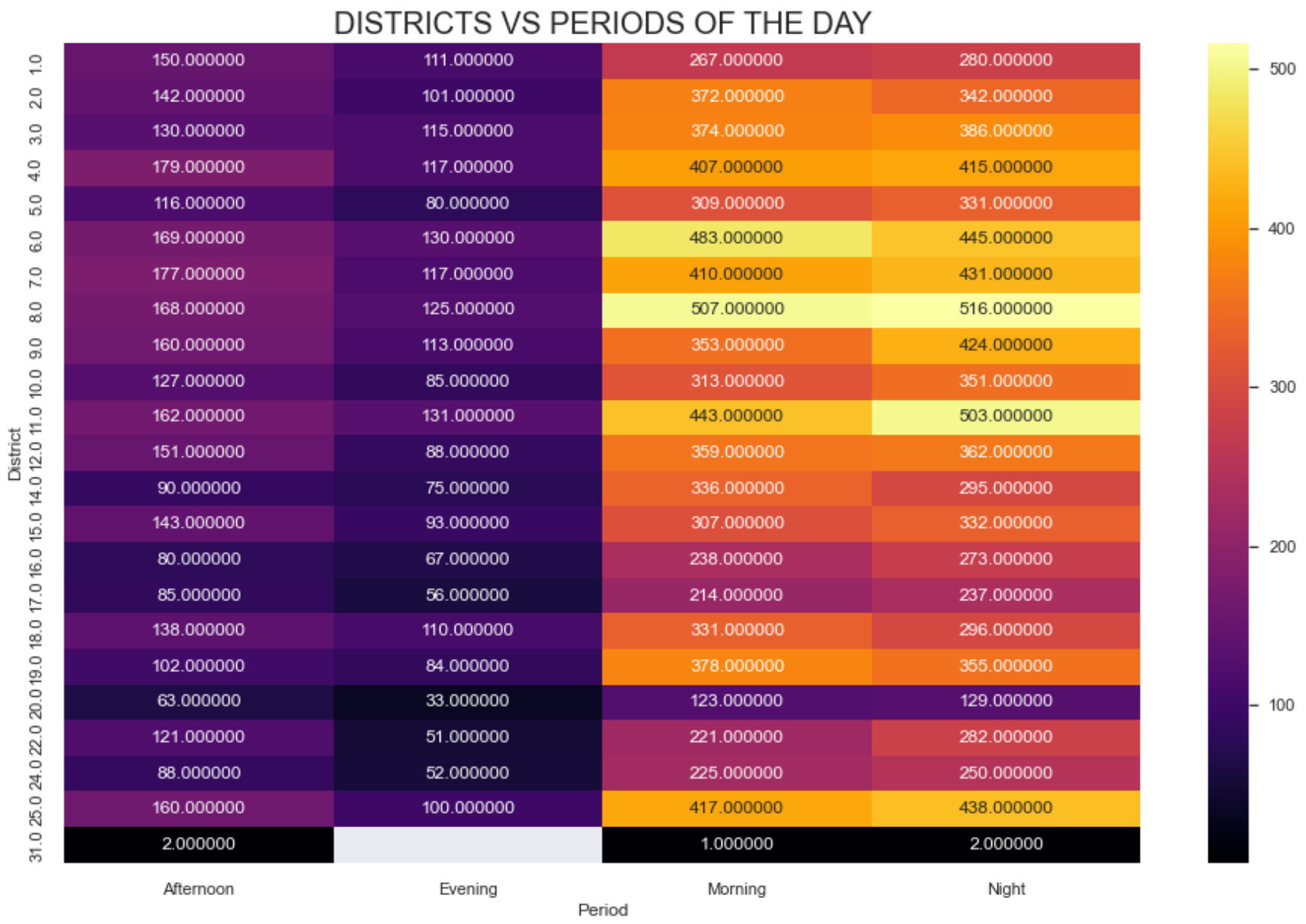
```
plt.figure(figsize = (16,10))
plt.title("DISTRICTS VS DAYTYPE", fontsize=20)
with sns.axes_style("white"):
    sns.heatmap(cri3, mask=cri3.isnull(), cmap="inferno", annot=True, fmt="f")


# Most district has their peak on weekdays except for district 31.0 having its most crimes on weekends and
# it is the most safest district in Chicago with the least crime rate.
```



In [162…

```
# Plotting for District vs Period in crimes
cri3 = dfFinal.groupby(['District','Period'],as_index=False).agg({'Primary_Type':"count"})
cri3.columns

cri3 = cri3.pivot("District", "Period", "Primary_Type")
```

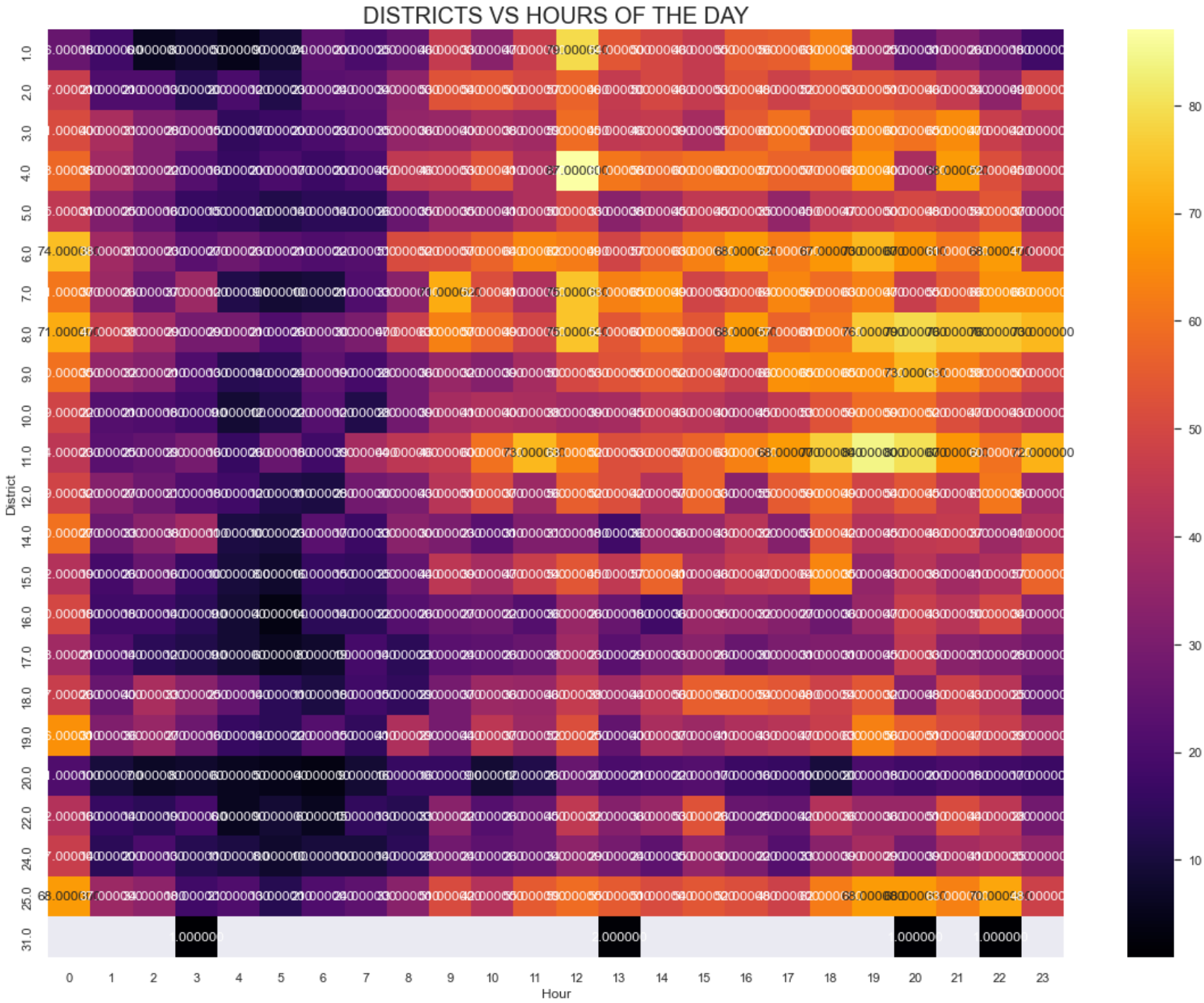
```
plt.figure(figsize = (16,10))
plt.title("DISTRICTS VS PERIODS OF THE DAY", fontsize=20)
with sns.axes_style("white"):
    sns.heatmap(cri3, mask=cri3.isnull(), cmap="inferno", annot=True, fmt="f")


# Most district have their peak period for at Night except for district 6.0, 14.0, 19.0 and 2.0 have their
# most crimes in the Morning and 31.0 being the most safest district in Chicago with the least crime rate.
```



In [163…

```
# Plotting for District vs Hour in crimes

cri3 = dfFinal.groupby(['District','Hour'],as_index=False).agg({'Primary_Type':"count"})
cri3.columns

cri3 = cri3.pivot("District", "Hour", "Primary_Type")
```

```
plt.figure(figsize = (20,15))
plt.title("DISTRICTS VS HOURS OF THE DAY", fontsize=20)
with sns.axes_style("white"):
    sns.heatmap(cri3, mask=cri3.isnull(), cmap="inferno", annot=True, fmt="f")
```


DISTRICTS VS HOURS OF THE DAY
District
Hour
1.0
2.0
3.0
4.0
5.0
6.0
7.0
8.0
9.0
10.0
11.0
12.0
14.0
15.0
16.0
17.0
18.0
19.0
20.0
22.0
24.0
25.0
31.0
0
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
80
70
60
50
40
30
20
10

DISCOVERIES:

- Primary type crime: Predominant crimes are theft, battery and criminal damage.

- Location description: Predominant crime location are street, residence and sidewalks

- Month: Bimodal distribution; crime peaks (in summer) at June and July (maybe due vacation time or day-light saving or tourism)

- Day: Friday has the most crime rate and Sunday has the least crime rate of the days of the week.

- DayType: There are more crimes on weekdays than weekends while

- Period: the periods crimes occur the most are the Night and Morning respectively

- Hour: The crime activities are more common at the peak hours 12 mid-day while least at 5:00am.

- District: District 31 has the least crime rate while district 8, 11, and 6 has the most crime chronologically

- There are some districts, and location description with more crime than other

- Domestic: Most crimes are non-domestic than domestic.

- Arrest: Arrest generally increase with the number of crime and about 27% of crimes were arrested in all locations.

- Also, most crimes do not lead to an arrest.

- Thereby, creating a huge gap between the rate of crime and the arrest status resulting in imbalanced data.

There has been a reoccurring crime trend over the months, weeks, days and period.

The trend appears to follow the same pattern throughout the years in which the timeframe.

This information is highly useful, and it pinpoints that despite the efforts made by the Chicago PD in terms of resource allocation, alarmingly, the crime frequency pattern with regards to timeframes per hour has not really changed.

In [ ]:

CONCLUSION

This Project investigated crime analysis using the Chicago dataset to understand why arrest status is low at specific times and in various Districts in the city of Chicago for crime types, such as: theft, battery and assault.

To achieve the given task, an in-depth analytics study (both univariate and bivariate) and visualization of crime type and

location against the arrest rate, months, week day, period and hours has been conducted to understand the correlations between timeframes, locations, and crime type. also, as well as relationship between crime types as Theft and Narcotics has similar trend and pattern of occurence.

This project's analytics result is to advise the Chicago PD to re-examine and structure/model a service-oriented system linked with effective crime prevention and management. The PD needs to allocate resources more efficiently to increase the possibility of crime prevention and thus reduce crimes in specific timeframes in recognised districts, and crime location description as indicated earlier.

According to the analysis result and visualization:

I can view the most frequently occurring crimes and the frequent occurring locations where crimes happened.

From these reports, the most occurred crimes were theft, battery, criminal damage and narcotics which is about 65% of all the crimes reported Chicago across all location.

The most common locations to occur the crimes are at street, sidewalks, residence, an apartment which are where people are probably mostly at. I specifically looked into certain crime types to view how they have changed over the months, day and period of the day such as theft, battery, and Narcostics crimes.

Even though there were a lot of reported crimes in Chicago each month, the arrest rate was not even as high as 30% for all crimes and locations.

This therefore, revealing that the Chicago’s Police arrest or investigation methods were not effective enough.

The analysis also reveals SAFEST and UNSAFE (that is peak and un-peak) periods of crime occurrence per crime type and

location (districts and crime location description) for months of the year, days of the week, period of the day and

hour of day.

I believe that the information of this data analytics about the security status of the Chicago city, provides much

more valuable informations which can be used as a powerful source to help Chicago Police department to make informed

decisions and for actions that increases the security status of Chicago.

RECOMMENDATION

Having gained insights on these three attributes(district, to find the 'hot' locations per crime type and time),Chicago

PD will be able to optimize the resources allocated, such as:

Community service (resident volunteering) among others such as:

- crime notification reward system to encourage the dwellers to report crimes in their location,

- Introduction of whistle blowing and rewards

New stations and more stations in the hot districts,

Patrols (frequent patrol on red zone streets): These three districts (11.0, 8.0 and 6.0) are classified as dangerous

neighbourhoods in Chicago. This pattern of crimes indicates an alarming situation for the PD requiring increased patrols

in these specific districts especially at peak times.

Placing more police community support officers in these communities provides a reassuring presence and

is a promising crime prevention strategy.

Education scholarships and job creation to reduce crime should be looked at by the government.

Equipping tPD with the type of resources needed per specific crime and training them.

Lighting the Streets at night to expose street crimes.

In [ ]:

In [ ]:

PLOTLY

In [164…

```
!pip install cufflinks
```

```
Requirement already satisfied: cufflinks in c:\users\oluwatriumph\anaconda3\lib\site-packages (0.17.3)
Requirement already satisfied: colorlover>=0.2.1 in c:\users\oluwatriumph\anaconda3\lib\site-packages (from cufflinks)
(0.3.0)
Requirement already satisfied: six>=1.9.0 in c:\users\oluwatriumph\anaconda3\lib\site-packages (from cufflinks) (1.16.
0)
Requirement already satisfied: setuptools>=34.4.1 in c:\users\oluwatriumph\anaconda3\lib\site-packages (from cufflinks)
(61.2.0)
Requirement already satisfied: ipython>=5.3.0 in c:\users\oluwatriumph\anaconda3\lib\site-packages (from cufflinks) (8.
2.0)
Requirement already satisfied: ipywidgets>=7.0.0 in c:\users\oluwatriumph\anaconda3\lib\site-packages (from cufflinks)
(7.6.5)
Requirement already satisfied: numpy>=1.9.2 in c:\users\oluwatriumph\anaconda3\lib\site-packages (from cufflinks) (1.2
1.5)
Requirement already satisfied: pandas>=0.19.2 in c:\users\oluwatriumph\anaconda3\lib\site-packages (from cufflinks) (1.
4.2)
Requirement already satisfied: plotly>=4.1.1 in c:\users\oluwatriumph\anaconda3\lib\site-packages (from cufflinks) (5.
6.0)
Requirement already satisfied: pickleshare in c:\users\oluwatriumph\anaconda3\lib\site-packages (from ipython>=5.3.0->c
ufflinks) (0.7.5)
Requirement already satisfied: pygments>=2.4.0 in c:\users\oluwatriumph\anaconda3\lib\site-packages (from ipython>=5.3.
0->cufflinks) (2.11.2)
Requirement already satisfied: stack-data in c:\users\oluwatriumph\anaconda3\lib\site-packages (from ipython>=5.3.0->cu
fflinks) (0.2.0)
Requirement already satisfied: decorator in c:\users\oluwatriumph\anaconda3\lib\site-packages (from ipython>=5.3.0->cuf
flinks) (5.1.1)
Requirement already satisfied: colorama in c:\users\oluwatriumph\anaconda3\lib\site-packages (from ipython>=5.3.0->cuff
links) (0.4.4)
Requirement already satisfied: matplotlib-inline in c:\users\oluwatriumph\anaconda3\lib\site-packages (from ipython>=5.
3.0->cufflinks) (0.1.2)
Requirement already satisfied: backcall in c:\users\oluwatriumph\anaconda3\lib\site-packages (from ipython>=5.3.0->cuff
links) (0.2.0)
Requirement already satisfied: traitlets>=5 in c:\users\oluwatriumph\anaconda3\lib\site-packages (from ipython>=5.3.0->
cufflinks) (5.1.1)
Requirement already satisfied: jedi>=0.16 in c:\users\oluwatriumph\anaconda3\lib\site-packages (from ipython>=5.3.0->cu
fflinks) (0.18.1)
Requirement already satisfied: prompt-toolkit!=3.0.0,!=3.0.1,<3.1.0,>=2.0.0 in c:\users\oluwatriumph\anaconda3\lib\site
-packages (from ipython>=5.3.0->cufflinks) (3.0.20)
Requirement already satisfied: widgetsnbextension~=3.5.0 in c:\users\oluwatriumph\anaconda3\lib\site-packages (from ipy
widgets>=7.0.0->cufflinks) (3.5.2)
Requirement already satisfied: ipykernel>=4.5.1 in c:\users\oluwatriumph\anaconda3\lib\site-packages (from ipywidgets>=
7.0.0->cufflinks) (6.9.1)
Requirement already satisfied: nbformat>=4.2.0 in c:\users\oluwatriumph\anaconda3\lib\site-packages (from ipywidgets>=
7.0.0->cufflinks) (5.3.0)
Requirement already satisfied: ipython-genutils~=0.2.0 in c:\users\oluwatriumph\anaconda3\lib\site-packages (from ipywi
dgets>=7.0.0->cufflinks) (0.2.0)
```

```
Requirement already satisfied: jupyterlab-widgets>=1.0.0 in c:\users\oluwatriumph\anaconda3\lib\site-packages (from ipy
widgets>=7.0.0->cufflinks) (1.0.0)
Requirement already satisfied: tornado<7.0,>=4.2 in c:\users\oluwatriumph\anaconda3\lib\site-packages (from ipykernel>=
4.5.1->ipywidgets>=7.0.0->cufflinks) (6.1)
Requirement already satisfied: nest-asyncio in c:\users\oluwatriumph\anaconda3\lib\site-packages (from ipykernel>=4.5.1
->ipywidgets>=7.0.0->cufflinks) (1.5.5)
Requirement already satisfied: jupyter-client<8.0 in c:\users\oluwatriumph\anaconda3\lib\site-packages (from ipykernel>
=4.5.1->ipywidgets>=7.0.0->cufflinks) (6.1.12)
Requirement already satisfied: debugpy<2.0,>=1.0.0 in c:\users\oluwatriumph\anaconda3\lib\site-packages (from ipykernel
>=4.5.1->ipywidgets>=7.0.0->cufflinks) (1.5.1)
Requirement already satisfied: parso<0.9.0,>=0.8.0 in c:\users\oluwatriumph\anaconda3\lib\site-packages (from jedi>=0.1
6->ipython>=5.3.0->cufflinks) (0.8.3)
Requirement already satisfied: pyzmq>=13 in c:\users\oluwatriumph\anaconda3\lib\site-packages (from jupyter-client<8.0-
>ipykernel>=4.5.1->ipywidgets>=7.0.0->cufflinks) (22.3.0)
Requirement already satisfied: python-dateutil>=2.1 in c:\users\oluwatriumph\anaconda3\lib\site-packages (from jupyter-
client<8.0->ipykernel>=4.5.1->ipywidgets>=7.0.0->cufflinks) (2.8.2)
Requirement already satisfied: jupyter-core>=4.6.0 in c:\users\oluwatriumph\anaconda3\lib\site-packages (from jupyter-c
lient<8.0->ipykernel>=4.5.1->ipywidgets>=7.0.0->cufflinks) (4.9.2)
Requirement already satisfied: pywin32>=1.0 in c:\users\oluwatriumph\anaconda3\lib\site-packages (from jupyter-core>=4.
6.0->jupyter-client<8.0->ipykernel>=4.5.1->ipywidgets>=7.0.0->cufflinks) (302)
Requirement already satisfied: fastjsonschema in c:\users\oluwatriumph\anaconda3\lib\site-packages (from nbformat>=4.2.
0->ipywidgets>=7.0.0->cufflinks) (2.15.1)
Requirement already satisfied: jsonschema>=2.6 in c:\users\oluwatriumph\anaconda3\lib\site-packages (from nbformat>=4.
2.0->ipywidgets>=7.0.0->cufflinks) (4.4.0)
Requirement already satisfied: attrs>=17.4.0 in c:\users\oluwatriumph\anaconda3\lib\site-packages (from jsonschema>=2.6
->nbformat>=4.2.0->ipywidgets>=7.0.0->cufflinks) (21.4.0)
Requirement already satisfied: pyrsistent!=0.17.0,!=0.17.1,!=0.17.2,>=0.14.0 in c:\users\oluwatriumph\anaconda3\lib\sit
e-packages (from jsonschema>=2.6->nbformat>=4.2.0->ipywidgets>=7.0.0->cufflinks) (0.18.0)
Requirement already satisfied: pytz>=2020.1 in c:\users\oluwatriumph\anaconda3\lib\site-packages (from pandas>=0.19.2->
cufflinks) (2021.3)
Requirement already satisfied: tenacity>=6.2.0 in c:\users\oluwatriumph\anaconda3\lib\site-packages (from plotly>=4.1.1
->cufflinks) (8.0.1)
Requirement already satisfied: wcwidth in c:\users\oluwatriumph\anaconda3\lib\site-packages (from prompt-toolkit!=3.0.
0,!=3.0.1,<3.1.0,>=2.0.0->ipython>=5.3.0->cufflinks) (0.2.5)
Requirement already satisfied: notebook>=4.4.1 in c:\users\oluwatriumph\anaconda3\lib\site-packages (from widgetsnbexte
nsion~=3.5.0->ipywidgets>=7.0.0->cufflinks) (6.4.8)
Requirement already satisfied: jinja2 in c:\users\oluwatriumph\anaconda3\lib\site-packages (from notebook>=4.4.1->widge
tsnbextension~=3.5.0->ipywidgets>=7.0.0->cufflinks) (2.11.3)
Requirement already satisfied: nbconvert in c:\users\oluwatriumph\anaconda3\lib\site-packages (from notebook>=4.4.1->wi
dgetsnbextension~=3.5.0->ipywidgets>=7.0.0->cufflinks) (6.4.4)
Requirement already satisfied: prometheus-client in c:\users\oluwatriumph\anaconda3\lib\site-packages (from notebook>=
4.4.1->widgetsnbextension~=3.5.0->ipywidgets>=7.0.0->cufflinks) (0.13.1)
Requirement already satisfied: Send2Trash>=1.8.0 in c:\users\oluwatriumph\anaconda3\lib\site-packages (from notebook>=
4.4.1->widgetsnbextension~=3.5.0->ipywidgets>=7.0.0->cufflinks) (1.8.0)
Requirement already satisfied: terminado>=0.8.3 in c:\users\oluwatriumph\anaconda3\lib\site-packages (from notebook>=4.
```

```
4.1->widgetsnbextension~=3.5.0->ipywidgets>=7.0.0->cufflinks) (0.13.1)
Requirement already satisfied: argon2-cffi in c:\users\oluwatriumph\anaconda3\lib\site-packages (from notebook>=4.4.1->
widgetsnbextension~=3.5.0->ipywidgets>=7.0.0->cufflinks) (21.3.0)
Requirement already satisfied: pywinpty>=1.1.0 in c:\users\oluwatriumph\anaconda3\lib\site-packages (from terminado>=0.
8.3->notebook>=4.4.1->widgetsnbextension~=3.5.0->ipywidgets>=7.0.0->cufflinks) (2.0.2)
Requirement already satisfied: argon2-cffi-bindings in c:\users\oluwatriumph\anaconda3\lib\site-packages (from argon2-c
ffi->notebook>=4.4.1->widgetsnbextension~=3.5.0->ipywidgets>=7.0.0->cufflinks) (21.2.0)
Requirement already satisfied: cffi>=1.0.1 in c:\users\oluwatriumph\anaconda3\lib\site-packages (from argon2-cffi-bindi
ngs->argon2-cffi->notebook>=4.4.1->widgetsnbextension~=3.5.0->ipywidgets>=7.0.0->cufflinks) (1.15.0)
Requirement already satisfied: pycparser in c:\users\oluwatriumph\anaconda3\lib\site-packages (from cffi>=1.0.1->argon2
-cffi-bindings->argon2-cffi->notebook>=4.4.1->widgetsnbextension~=3.5.0->ipywidgets>=7.0.0->cufflinks) (2.21)
Requirement already satisfied: MarkupSafe>=0.23 in c:\users\oluwatriumph\anaconda3\lib\site-packages (from jinja2->note
book>=4.4.1->widgetsnbextension~=3.5.0->ipywidgets>=7.0.0->cufflinks) (2.0.1)
Requirement already satisfied: bleach in c:\users\oluwatriumph\anaconda3\lib\site-packages (from nbconvert->notebook>=
4.4.1->widgetsnbextension~=3.5.0->ipywidgets>=7.0.0->cufflinks) (4.1.0)
Requirement already satisfied: defusedxml in c:\users\oluwatriumph\anaconda3\lib\site-packages (from nbconvert->noteboo
k>=4.4.1->widgetsnbextension~=3.5.0->ipywidgets>=7.0.0->cufflinks) (0.7.1)
Requirement already satisfied: jupyterlab-pygments in c:\users\oluwatriumph\anaconda3\lib\site-packages (from nbconvert
->notebook>=4.4.1->widgetsnbextension~=3.5.0->ipywidgets>=7.0.0->cufflinks) (0.1.2)
Requirement already satisfied: testpath in c:\users\oluwatriumph\anaconda3\lib\site-packages (from nbconvert->notebook>
=4.4.1->widgetsnbextension~=3.5.0->ipywidgets>=7.0.0->cufflinks) (0.5.0)
Requirement already satisfied: pandocfilters>=1.4.1 in c:\users\oluwatriumph\anaconda3\lib\site-packages (from nbconver
t->notebook>=4.4.1->widgetsnbextension~=3.5.0->ipywidgets>=7.0.0->cufflinks) (1.5.0)
Requirement already satisfied: entrypoints>=0.2.2 in c:\users\oluwatriumph\anaconda3\lib\site-packages (from nbconvert-
>notebook>=4.4.1->widgetsnbextension~=3.5.0->ipywidgets>=7.0.0->cufflinks) (0.4)
Requirement already satisfied: beautifulsoup4 in c:\users\oluwatriumph\anaconda3\lib\site-packages (from nbconvert->not
ebook>=4.4.1->widgetsnbextension~=3.5.0->ipywidgets>=7.0.0->cufflinks) (4.11.1)
Requirement already satisfied: mistune<2,>=0.8.1 in c:\users\oluwatriumph\anaconda3\lib\site-packages (from nbconvert->
notebook>=4.4.1->widgetsnbextension~=3.5.0->ipywidgets>=7.0.0->cufflinks) (0.8.4)
Requirement already satisfied: nbclient<0.6.0,>=0.5.0 in c:\users\oluwatriumph\anaconda3\lib\site-packages (from nbconv
ert->notebook>=4.4.1->widgetsnbextension~=3.5.0->ipywidgets>=7.0.0->cufflinks) (0.5.13)
Requirement already satisfied: soupsieve>1.2 in c:\users\oluwatriumph\anaconda3\lib\site-packages (from beautifulsoup4-
>nbconvert->notebook>=4.4.1->widgetsnbextension~=3.5.0->ipywidgets>=7.0.0->cufflinks) (2.3.1)
Requirement already satisfied: packaging in c:\users\oluwatriumph\anaconda3\lib\site-packages (from bleach->nbconvert->
notebook>=4.4.1->widgetsnbextension~=3.5.0->ipywidgets>=7.0.0->cufflinks) (21.3)
Requirement already satisfied: webencodings in c:\users\oluwatriumph\anaconda3\lib\site-packages (from bleach->nbconver
t->notebook>=4.4.1->widgetsnbextension~=3.5.0->ipywidgets>=7.0.0->cufflinks) (0.5.1)
Requirement already satisfied: pyparsing!=3.0.5,>=2.0.2 in c:\users\oluwatriumph\anaconda3\lib\site-packages (from pack
aging->bleach->nbconvert->notebook>=4.4.1->widgetsnbextension~=3.5.0->ipywidgets>=7.0.0->cufflinks) (3.0.4)
Requirement already satisfied: pure-eval in c:\users\oluwatriumph\anaconda3\lib\site-packages (from stack-data->ipython
>=5.3.0->cufflinks) (0.2.2)
Requirement already satisfied: executing in c:\users\oluwatriumph\anaconda3\lib\site-packages (from stack-data->ipython
>=5.3.0->cufflinks) (0.8.3)
Requirement already satisfied: asttokens in c:\users\oluwatriumph\anaconda3\lib\site-packages (from stack-data->ipython
>=5.3.0->cufflinks) (2.0.5)
```

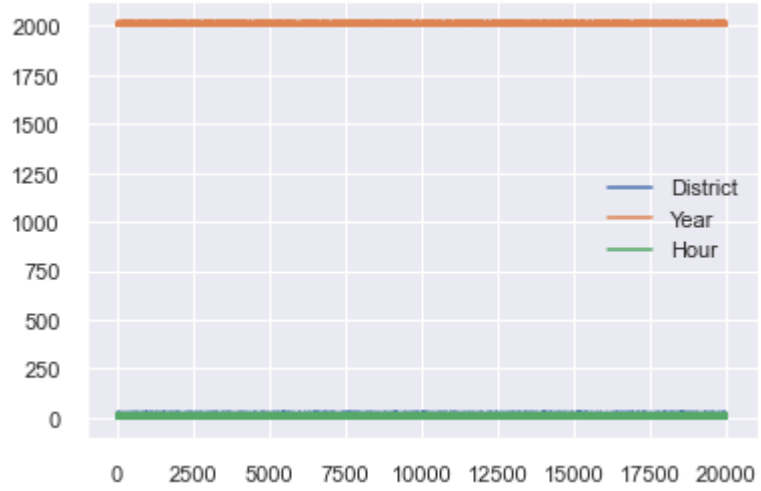
In [165…
```
!pip install chart_studio
```

```
Requirement already satisfied: chart_studio in c:\users\oluwatriumph\anaconda3\lib\site-packages (1.1.0)
Requirement already satisfied: retrying>=1.3.3 in c:\users\oluwatriumph\anaconda3\lib\site-packages (from chart_studio)
(1.3.3)
Requirement already satisfied: plotly in c:\users\oluwatriumph\anaconda3\lib\site-packages (from chart_studio) (5.6.0)
Requirement already satisfied: requests in c:\users\oluwatriumph\anaconda3\lib\site-packages (from chart_studio) (2.27.
1)
Requirement already satisfied: six in c:\users\oluwatriumph\anaconda3\lib\site-packages (from chart_studio) (1.16.0)
Requirement already satisfied: tenacity>=6.2.0 in c:\users\oluwatriumph\anaconda3\lib\site-packages (from plotly->chart
_studio) (8.0.1)
Requirement already satisfied: idna<4,>=2.5 in c:\users\oluwatriumph\anaconda3\lib\site-packages (from requests->chart_
studio) (3.3)
Requirement already satisfied: urllib3<1.27,>=1.21.1 in c:\users\oluwatriumph\anaconda3\lib\site-packages (from request
s->chart_studio) (1.26.9)
Requirement already satisfied: charset-normalizer~=2.0.0 in c:\users\oluwatriumph\anaconda3\lib\site-packages (from req
uests->chart_studio) (2.0.4)
Requirement already satisfied: certifi>=2017.4.17 in c:\users\oluwatriumph\anaconda3\lib\site-packages (from requests->
chart_studio) (2021.10.8)
```

In [166…
```
import cufflinks as cf
import seaborn as sns
import plotly.express as px
%matplotlib inline

# Make Plotly work in your Jupyter Notebook
from plotly.offline import download_plotlyjs, init_notebook_mode, plot, iplot
init_notebook_mode(connected=True)
# Use Plotly locally
cf.go_offline()
```

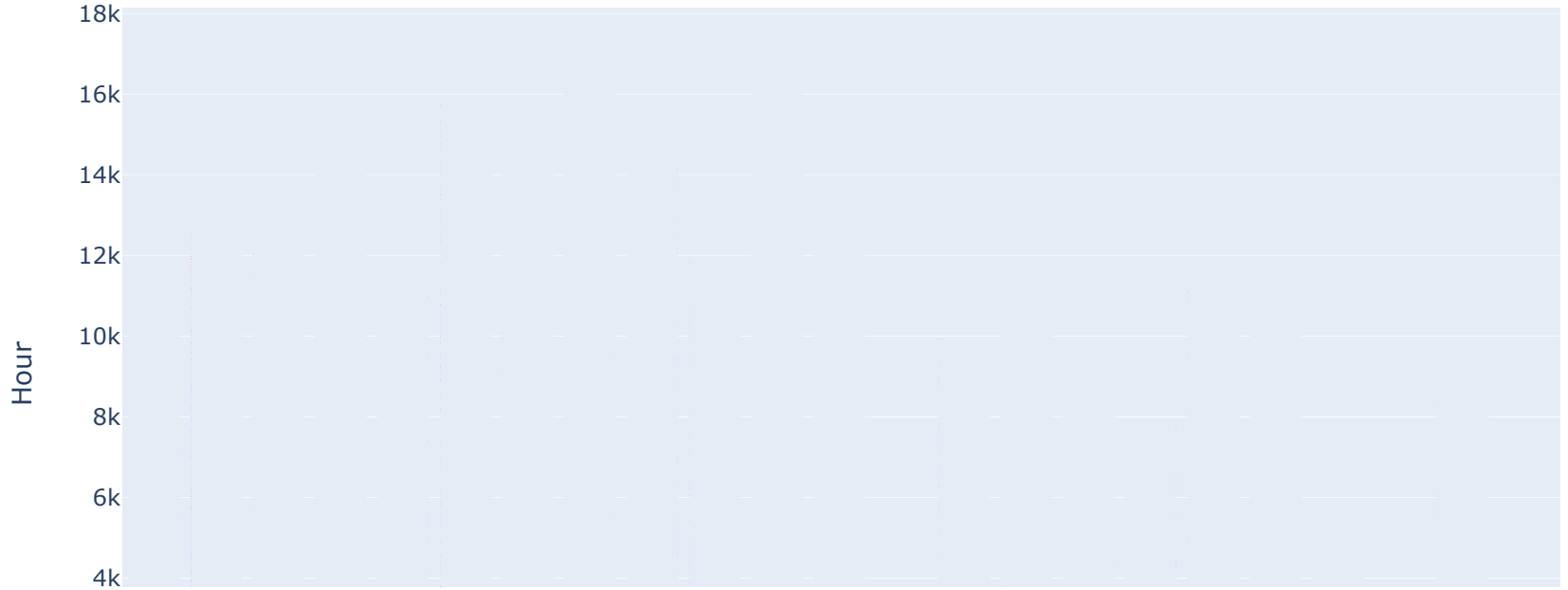
In [167…
```
dfFinal.plot()
```

Out[167]:
```
<AxesSubplot:>
```



In [168…

```
# ANALYSIS OF DISTRICT VS HOUR VS ARREST

# Bar chart
fig = px.bar(dfFinal, y="Hour", x="District", color="Arrest")
fig.show()
```

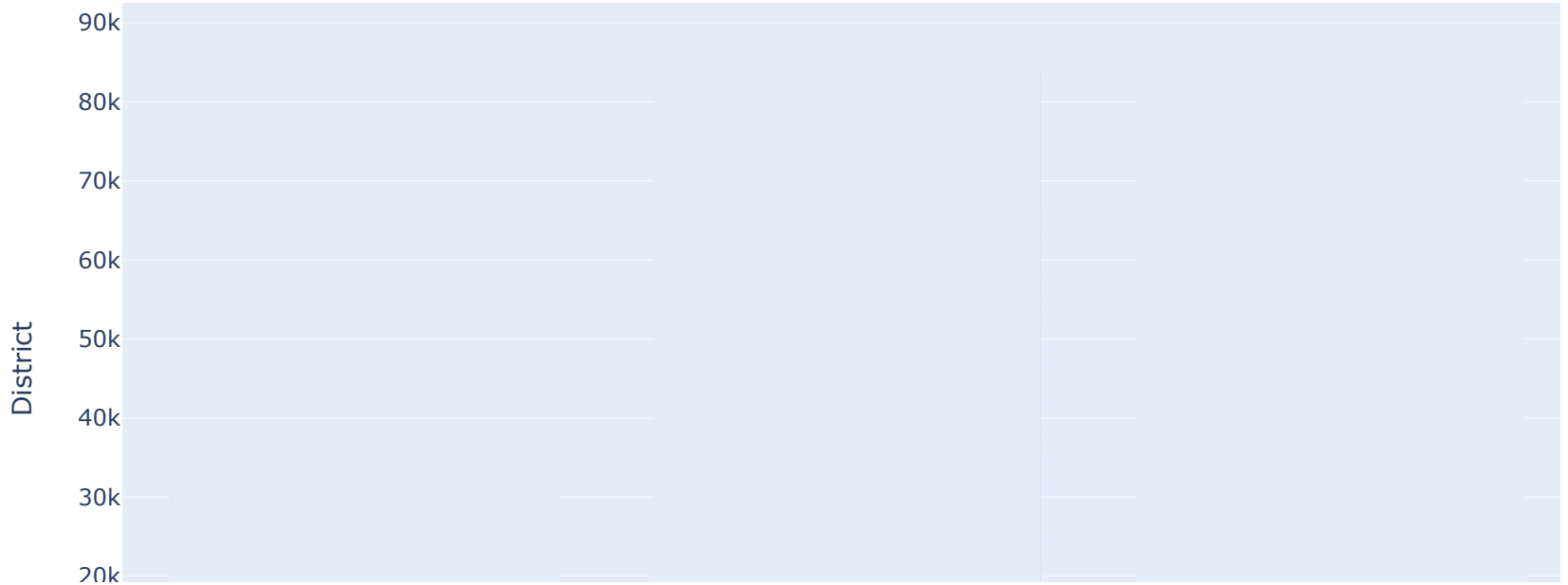


In [169…

```
# ANALYSIS OF DISTRICT VS DAY VS ARREST

fig = px.bar(dfFinal, y="District", x="Day", color="Arrest")
fig.show()
```

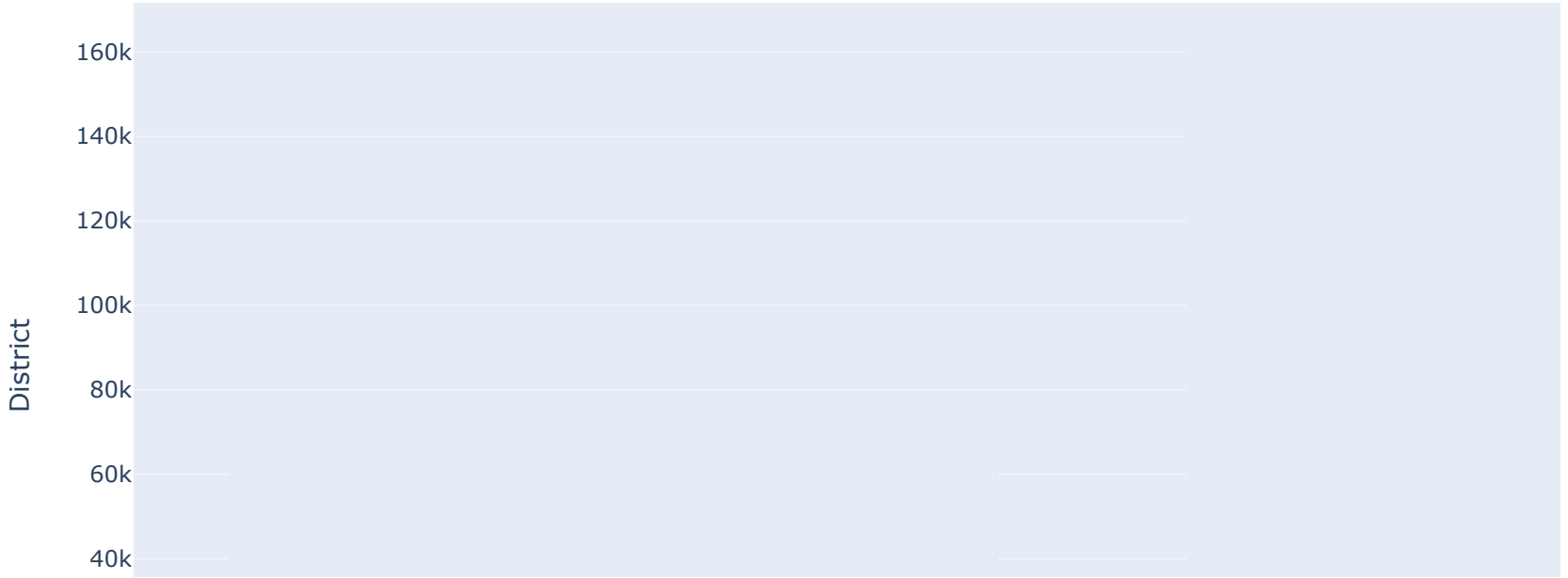
In [170…

```
# ANALYSIS OF DISTRICT VS PERIOD VS ARREST

fig = px.bar(dfFinal, y="District", x="Period", color="Arrest")
fig.show()
```



In [171…

```
# ANALYSIS OF DISTRICT VS DAYTYPE VS ARREST

fig = px.bar(dfFinal, y="District", x="DayType", color="Arrest")
fig.show()
```

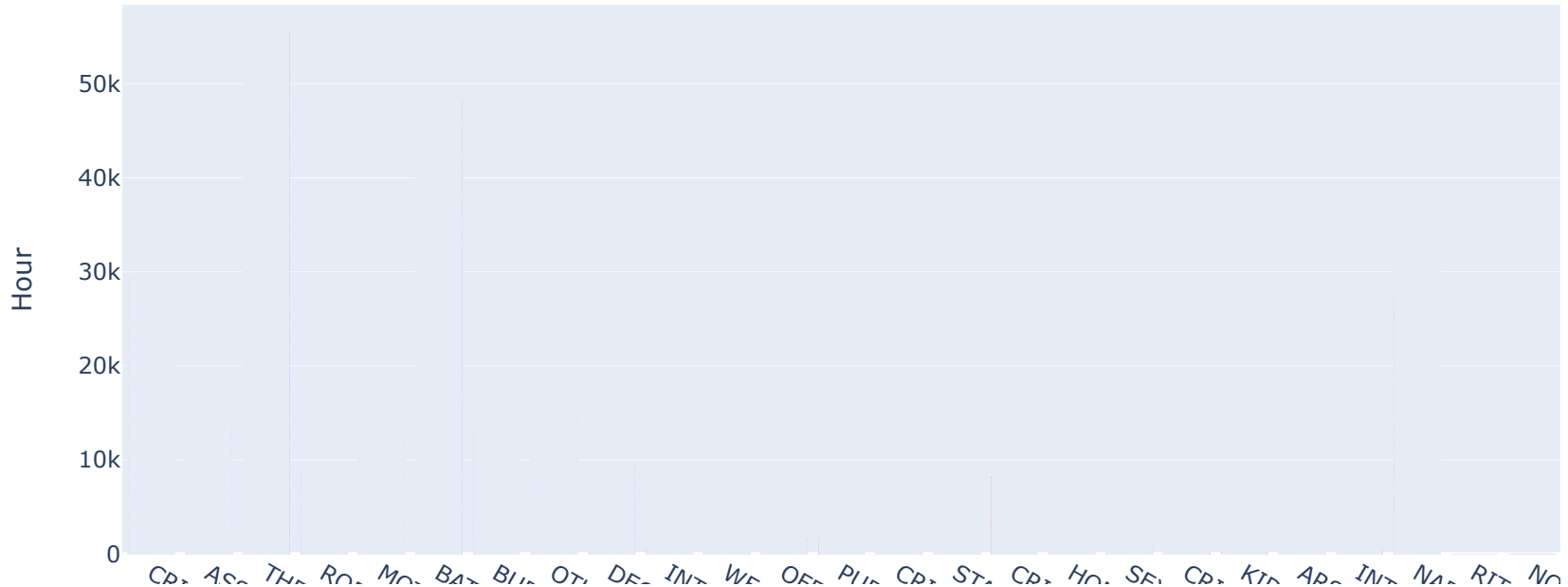


In [172…

```
# ANALYSIS OF PRIMARY-TYPE VS DAY VS ARREST

fig = px.bar(dfFinal, y="Primary_Type", x="Day", color="Arrest")
fig.show()
```

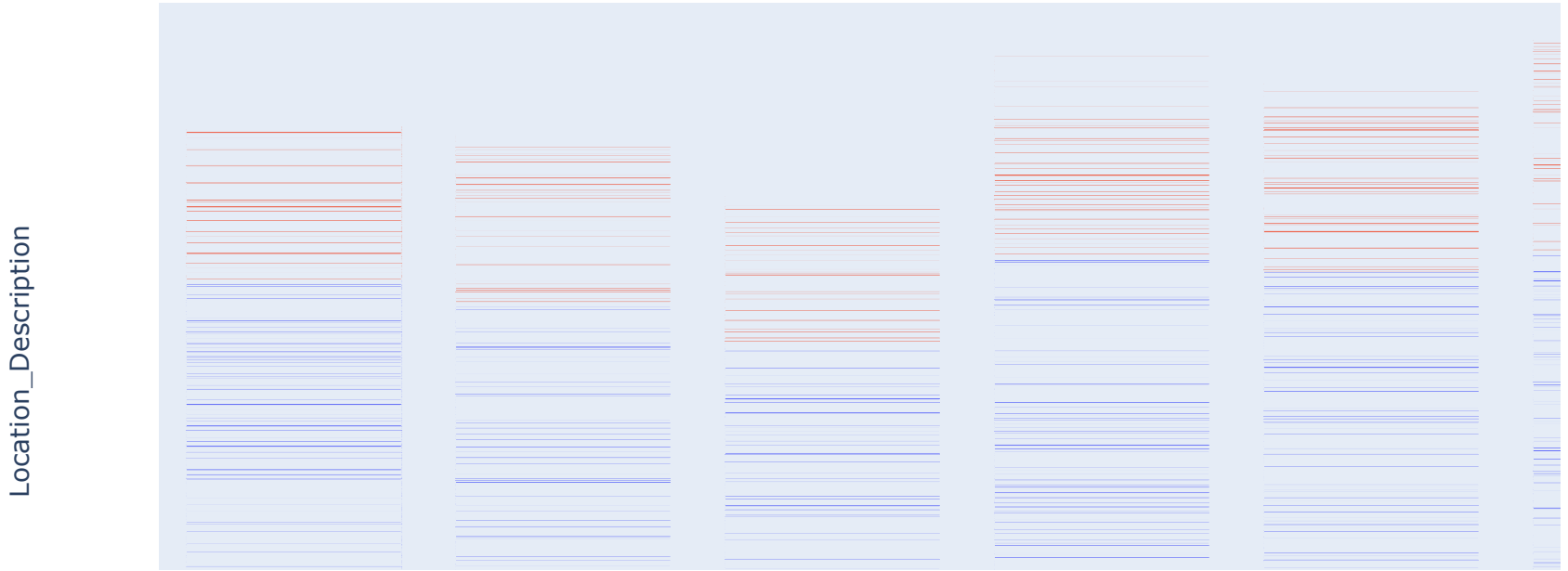
In [173…

```
# ANALYSIS OF PRIMARY-TYPE VS HOUR VS ARREST

fig = px.bar(dfFinal, y="Hour", x="Primary_Type", color="Arrest")
fig.show()
```



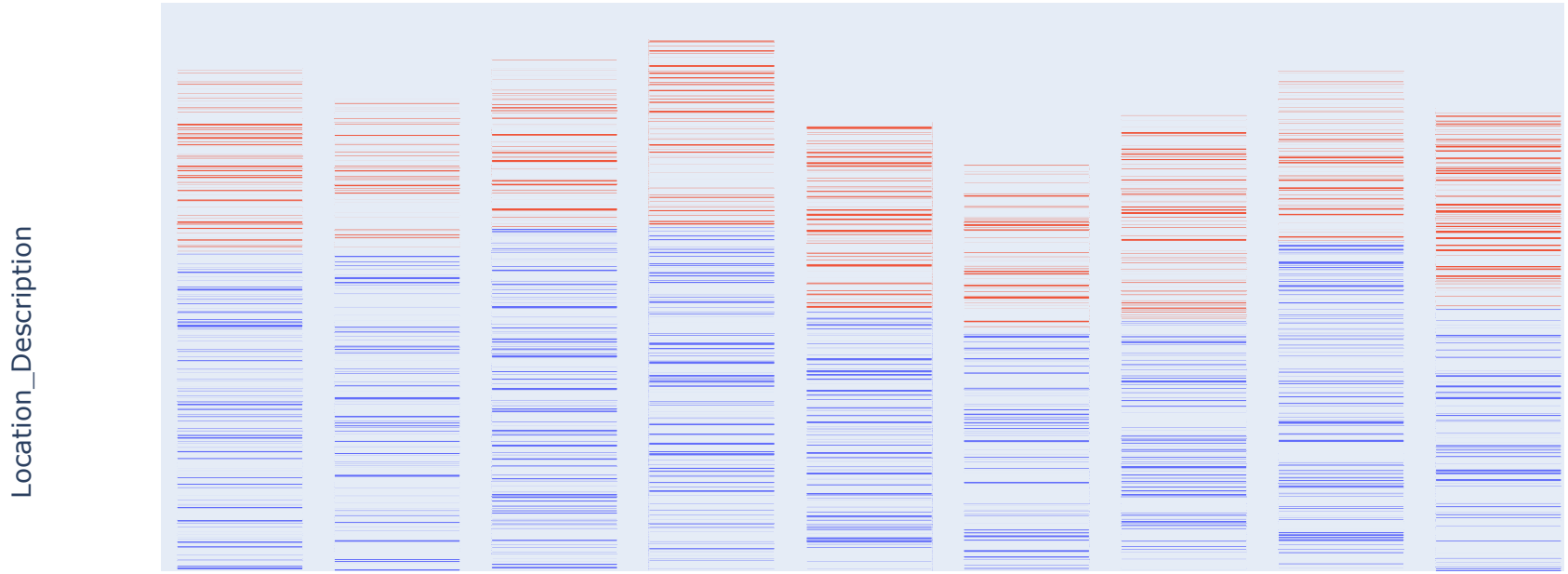
In [174…

```
# ANALYSIS OF LOCATION-DECRIPTION VS DAY VS ARREST

fig = px.bar(dfFinal, y="Location_Description", x="Day", color="Arrest")
fig.show()
```



In [175…

```
# ANALYSIS OF LOCATION-DESCRIPTION VS MONTH VS ARREST

fig = px.bar(dfFinal, y="Location_Description", x="Month", color="Arrest")
fig.show()
```

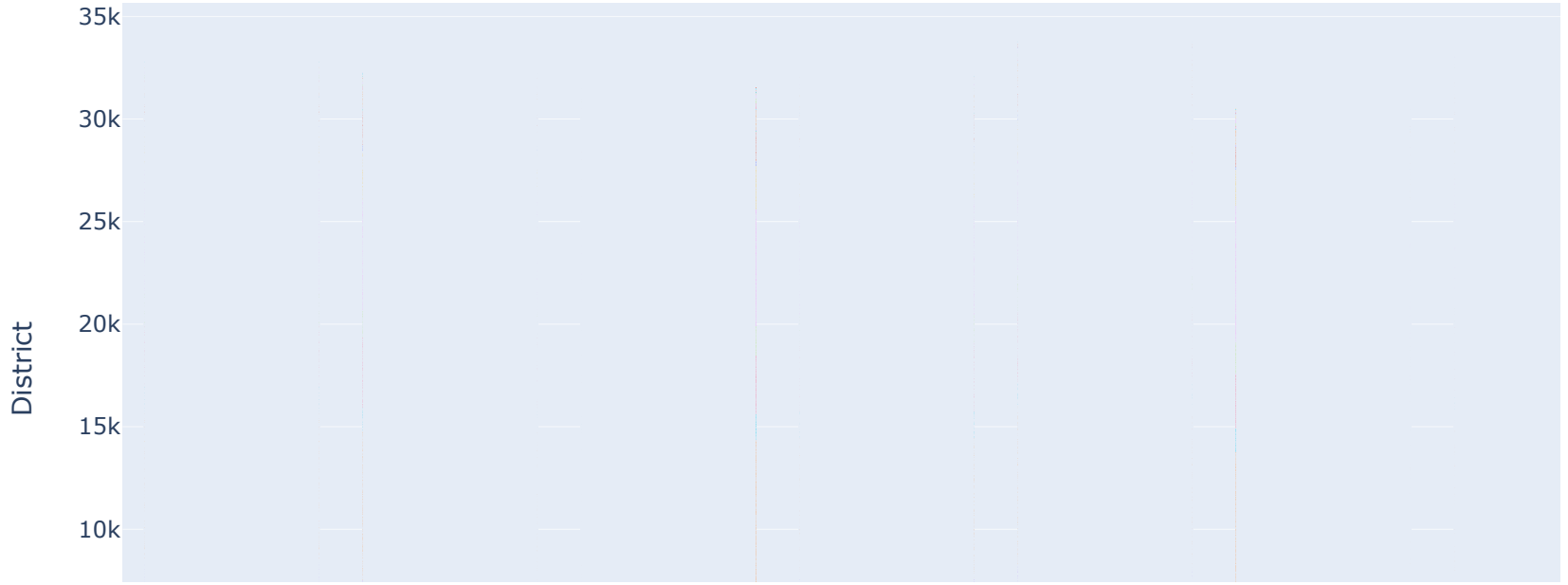


In [176…

```
# ANALYSIS OF DISTRICT VS DAY VS ARREST

fig = px.bar(dfFinal, y="District", x="Day", color="Arrest")
fig.show()
```

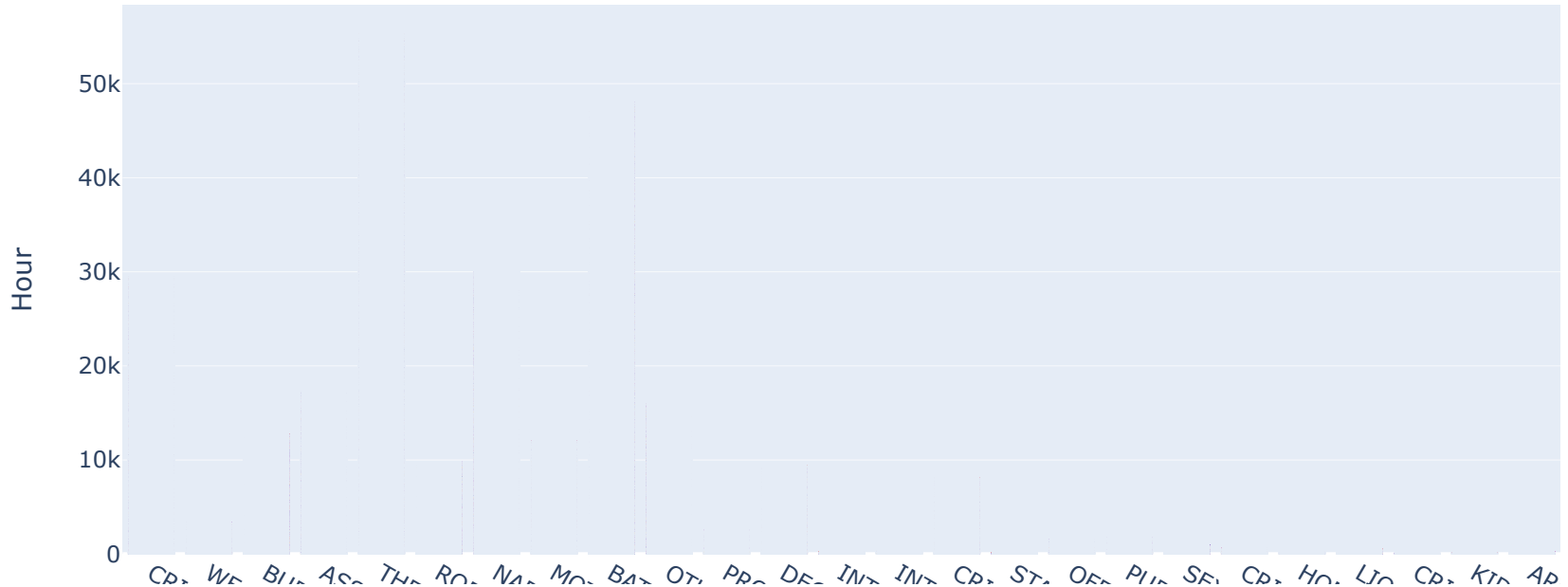
In [177…

```
# ANALYSIS OF DISTRICT VS DAY VS PRIMARY-TYPE

fig = px.bar(dfFinal, y="District", x="Day", color="Primary_Type")
fig.show()
```



In [178…

```
# ANALYSIS OF DISTRICT VS HOUR VS PRIMARY-TYPE

fig = px.bar(dfFinal, y="Hour", x="Primary_Type", color="District")
fig.show()
```

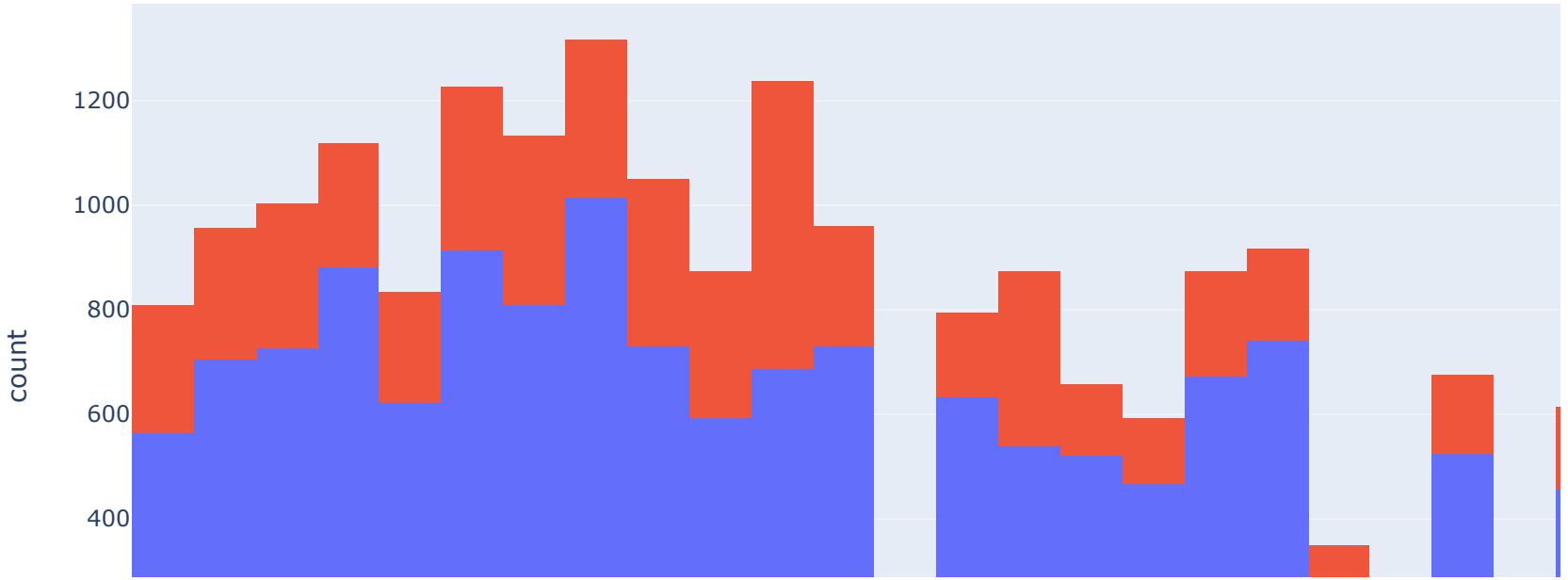


In [179…
```
#dfFinal = px.data.gapminder().query("dfFinal == 'THEFT' and year == 2007 and District > 2.e6")
#fig = px.bar(dfFinal_THEFT, y='District', x='Primary_Type', text='District', color='Primary_Type')
#fig.show()
```

In [180…
```
#fig = go.Figure() # Canvass
#fig = px.bar(df_europe, y='pop', x='country', text='pop', color='country')
#fig.update_traces(texttemplate='%{text:.2s}', textposition='outside')
#fig.update_layout(xaxis_tickangle=-45)
#fig.show()
```

In [181…
```
#dfFinal = px.data.tips()
```

In [182…
```
px.histogram(dfFinal, x="District", color="Arrest")
```

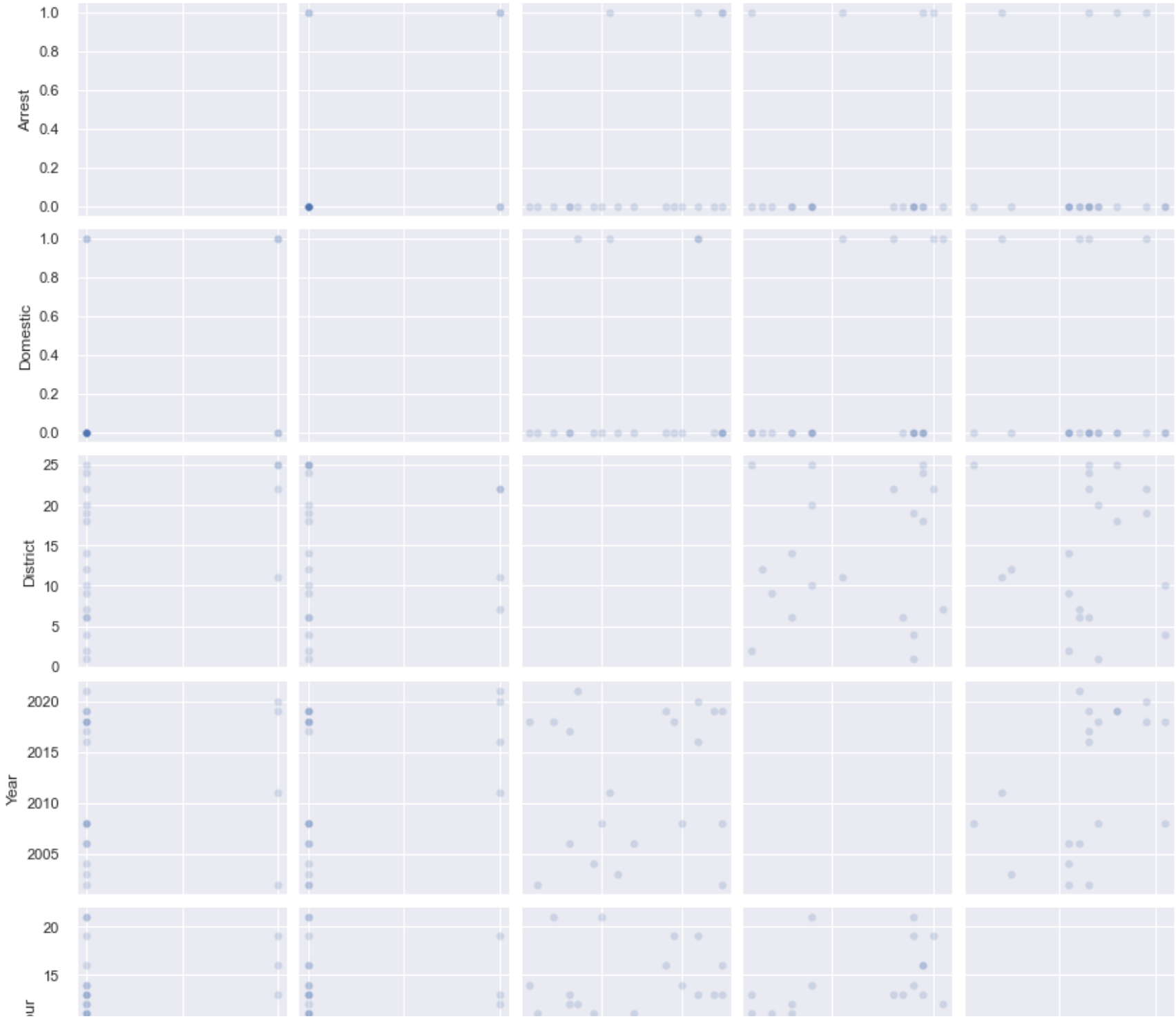


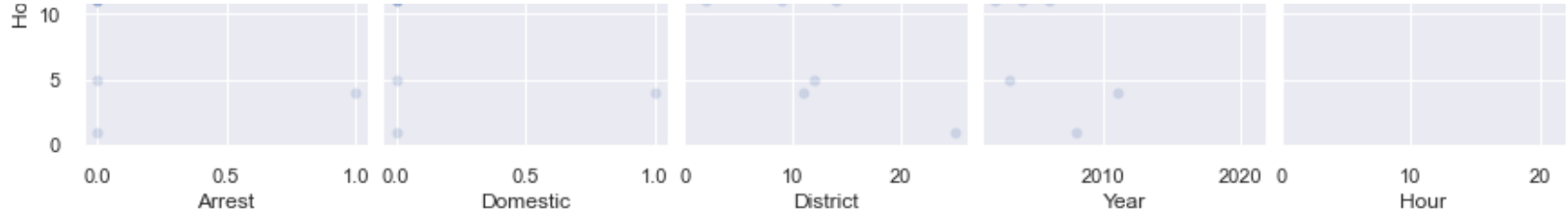
APPENDIX

In [ ]:
```
```

In [183…
```
####

sns.pairplot(data=dfFinal.sample(frac=.001), kind='scatter', diag_kind='scatter', plot_kws={'alpha': .2} )
```

Out[183]: <seaborn.axisgrid.PairGrid at 0x17f45292790>

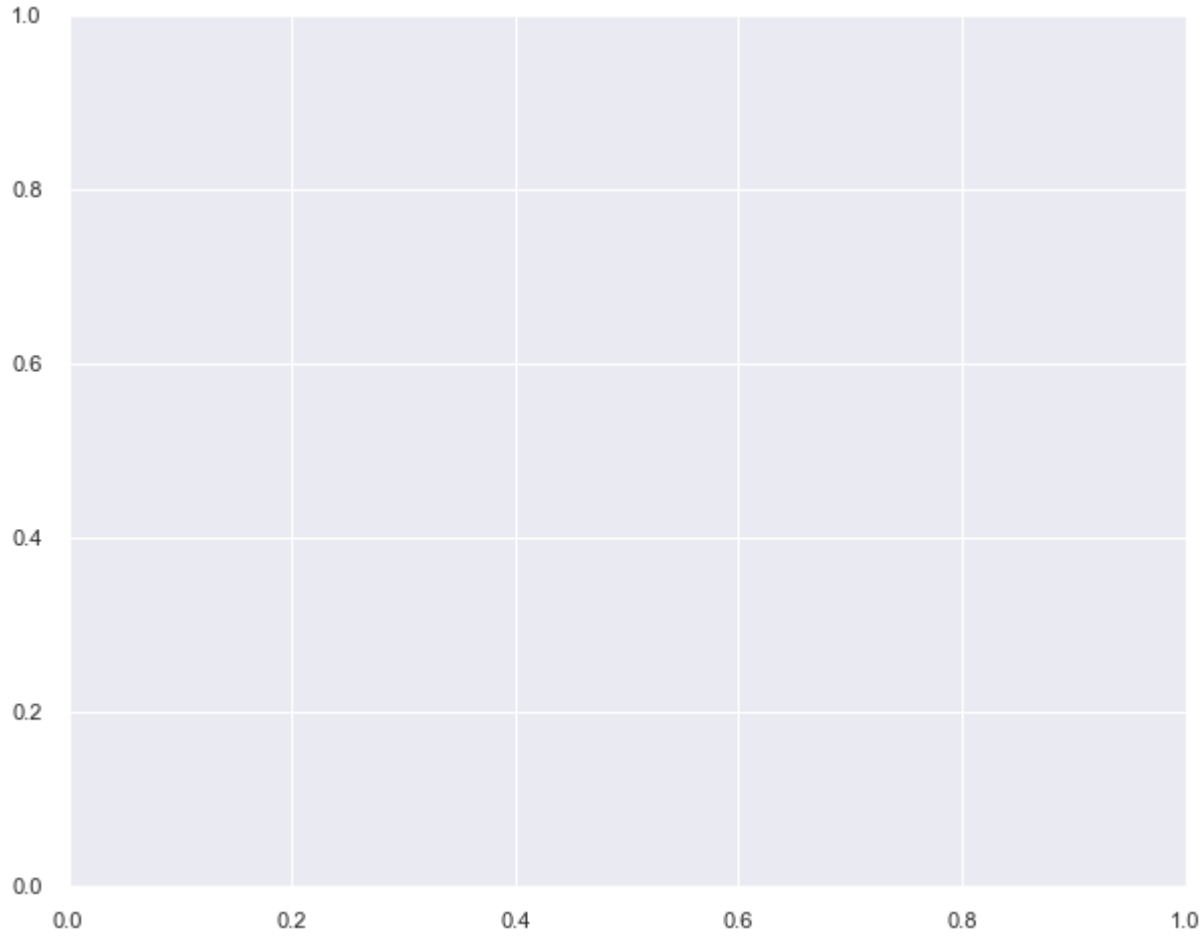

Arrest
1.0
0.8
0.6
0.4
0.2
0.0
Domestic
1.0
0.8
0.6
0.4
0.2
0.0
District
25
20
15
10
5
0
Year
2020
2015
2010
2005
20
15

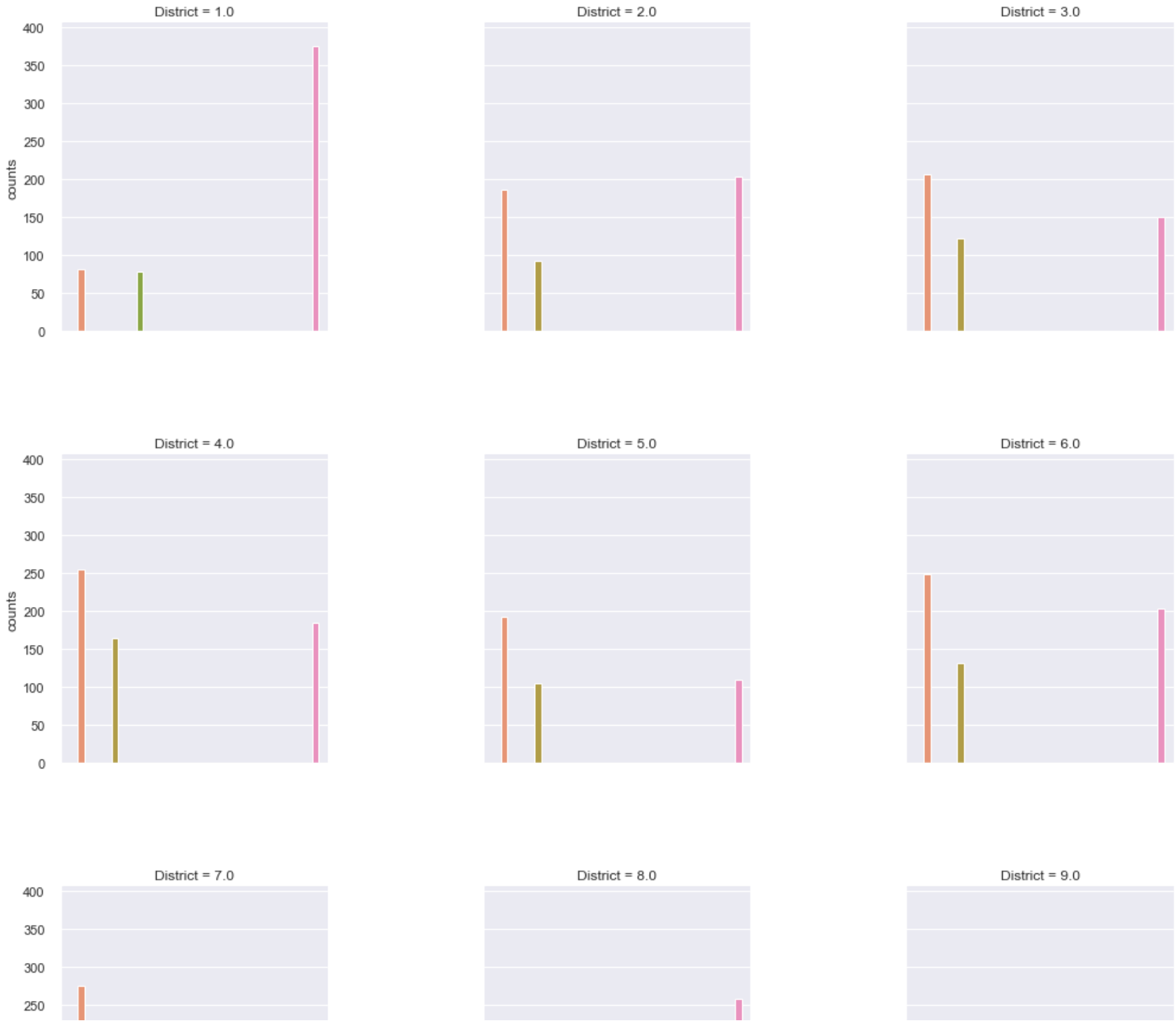


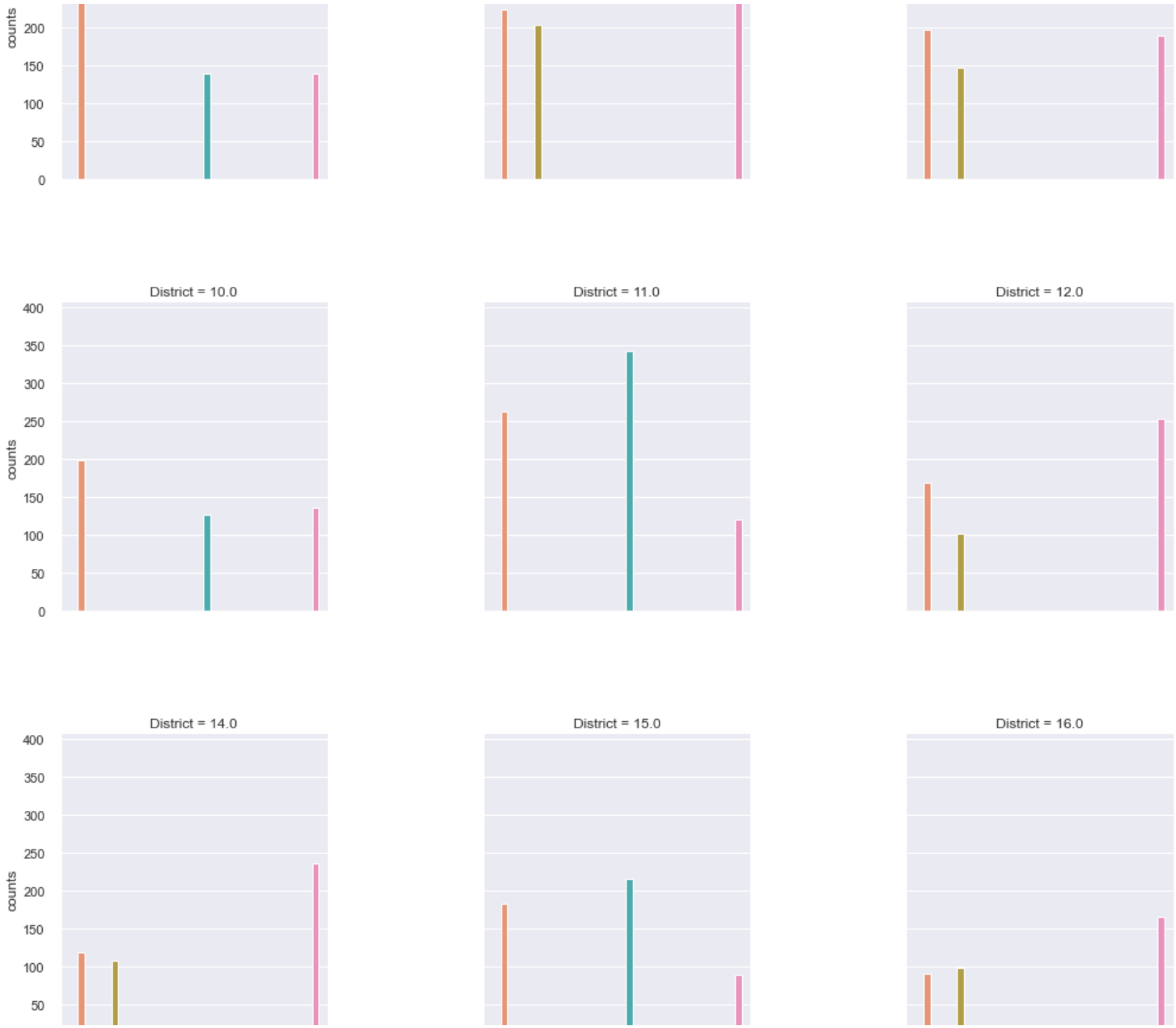
In [184…

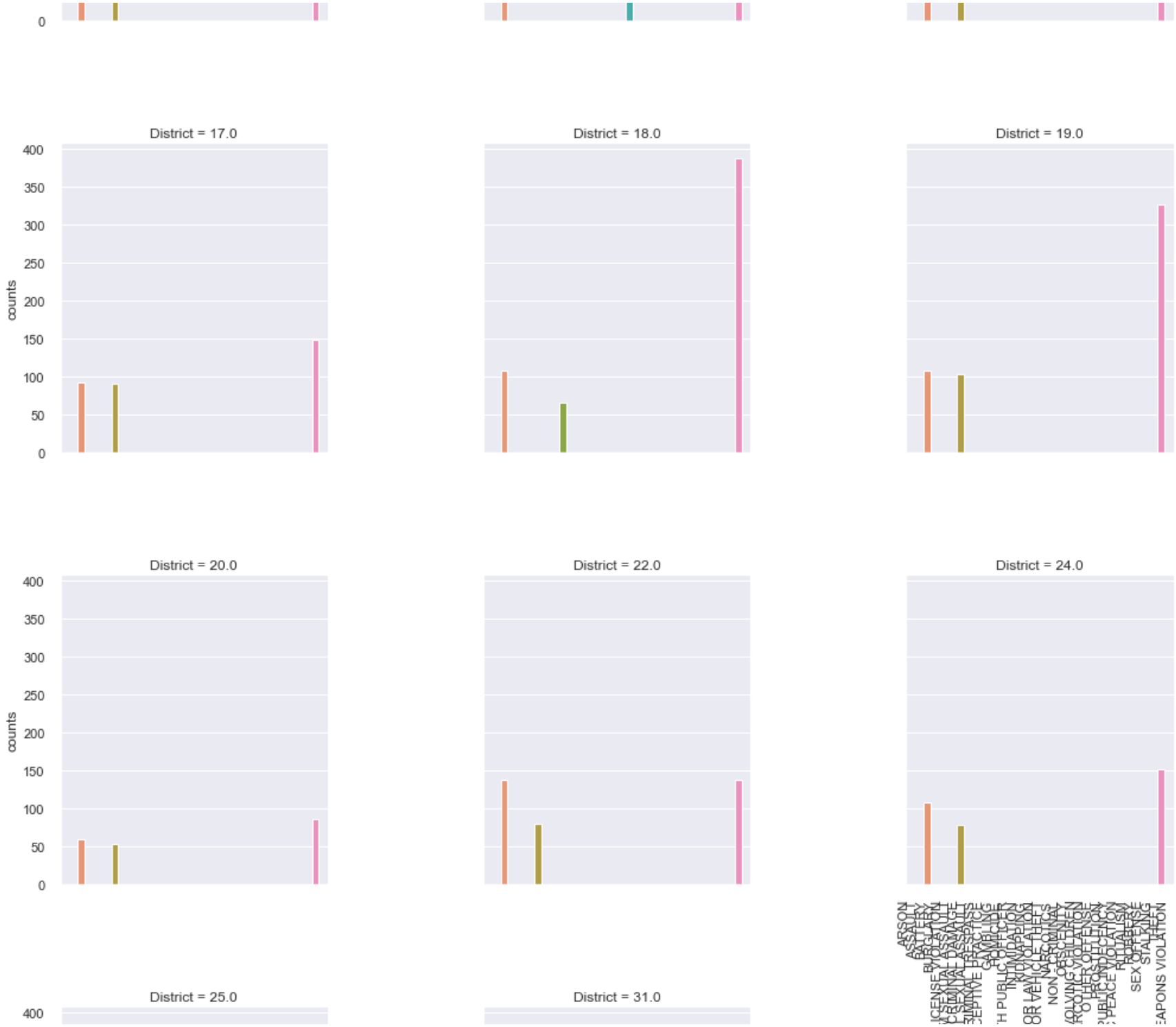
```
# location attributes = ['Location Description, 'District', ','Block']
%matplotlib inline

#plt.figure(figsize = (10, 8))
plt.subplots(figsize = (10,8))
#fig, ax = plt.subplots(figsize=(10, 5))
top = dfFinal.groupby(['District', 'Primary_Type']).size().reset_index(name='counts').groupby('District').apply(lambda
#print(topk)

# factor plot to make multiple plots
g =sns.catplot("Primary_Type", y='counts', col="District", col_wrap=3,
                    data=top, kind='bar')
for ax in g.axes:
    plt.setp(ax.get_xticklabels(), visible=True, rotation=90, ha='right')

plt.subplots_adjust(hspace=0.4)
```

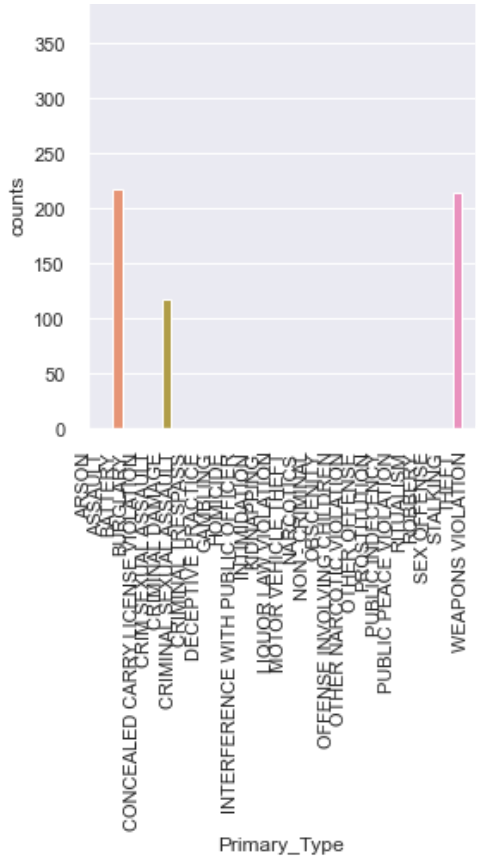
```
C:\Users\OLUWATRIUMPH\anaconda3\lib\site-packages\seaborn\_decorators.py:36: FutureWarning:

Pass the following variable as a keyword arg: x. From version 0.12, the only valid positional argument will be `data`,
and passing other arguments without an explicit keyword will result in an error or misinterpretation.
```

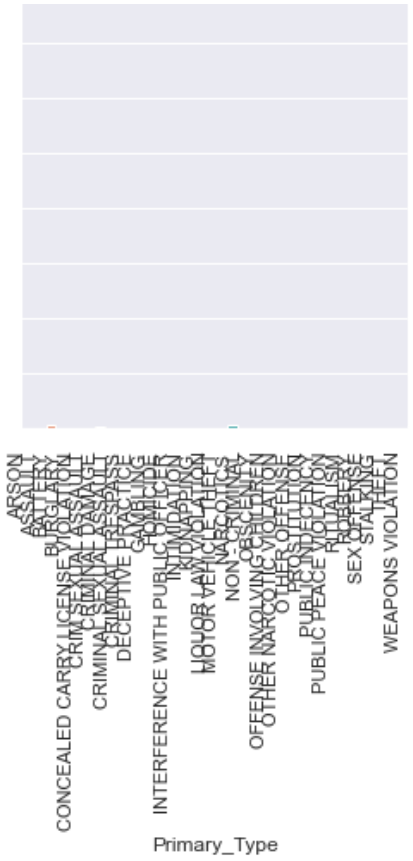

1.0
0.8
0.6
0.4
0.2
0.0
0.0
0.2
0.4
0.6
0.8
1.0


District = 1.0
District = 2.0
District = 3.0
District = 4.0
District = 5.0
District = 6.0
District = 7.0
District = 8.0
District = 9.0
counts
0
50
100
150
200
250
300
350
400


counts
District = 10.0
District = 11.0
District = 12.0
District = 14.0
District = 15.0
District = 16.0


District = 17.0
District = 18.0
District = 19.0
District = 20.0
District = 22.0
District = 24.0
District = 25.0
District = 31.0
counts
0
50
100
150
200
250
300
350
400



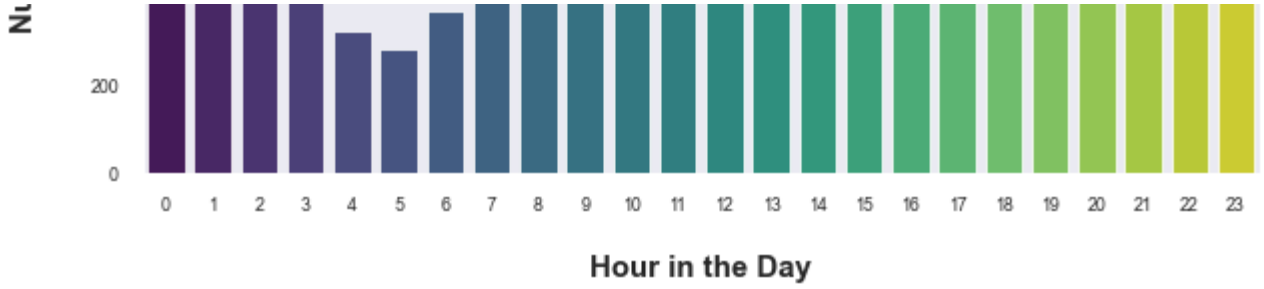


In [185…

```
# Set plot style
plt.style.use('seaborn-dark')
sns.set_context('paper')

# Write code to plot
fig, ax = plt.subplots(figsize=(10, 5))
sns.countplot(x='Hour', data=dfFinal, palette="viridis")

# Aesthetic appeal
plt.title("Unsafest Hours in Chicago", fontdict={'fontsize': 40, 'color': '#bb0e14','fontname':'Agency FB'}, weight="bo
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" }, weight='bold')

plt.show()
```

Lowest Crime Rate


Unsafest Hours in Chicago
1200
1000
800
600
400



In [186…
```
# ANALYSIS OF CRIME PRIMARY TYPE VS LOCATION DESCRIPTION VS DISTRICT VS HOUR


dfFinal.groupby(by=['Primary_Type', 'Location_Description', 'District', 'Hour']).size().sort_values(ascending=False)
```

Out[186]:
```
Primary_Type       Location_Description  District  Hour
THEFT              RESTAURANT            1.0       12      18
NARCOTICS          SIDEWALK              11.0      11      16
                                                   20      13
THEFT              STREET                18.0      0       13
NARCOTICS          SIDEWALK              11.0      18      13
                                                           ..
GAMBLING           RESIDENCE - GARAGE    8.0       5        0
                                                   4        0
                                                   3        0
                                                   2        0
WEAPONS VIOLATION  YARD                  31.0      23       0
Length: 2225664, dtype: int64
```

In [ ]:
```

```

In [187…
```
# ANALYSIS OF LOCATION DESCRIPTION AND DAYTYPE

dfFinal.groupby(by=['Location_Description', 'DayType']).size().sort_values(ascending=False)
```

Out[187]:
```
Location_Description                       DayType
STREET                                     Weekday    3684
RESIDENCE                                  Weekday    2327
APARTMENT                                  Weekday    1572
STREET                                     Weekend    1465
SIDEWALK                                   Weekday    1401
                                                      ...
HOUSE                                      Weekend       0
CTA PARKING LOT / GARAGE / OTHER PROPERTY  Weekend       0
ANIMAL HOSPITAL                            Weekend       0
GOVERNMENT BUILDING / PROPERTY             Weekend       0
SCHOOL - PUBLIC GROUNDS                    Weekend       0
Length: 252, dtype: int64
```

In [188…
```
# ANALYSIS OF CRIME PRIMARY TYPE AND DAY


dfFinal.groupby(by=['Primary_Type', 'Day']).size().sort_values(ascending=False)
```

Out[188]:
```
Primary_Type              Day
THEFT                     Fri     684
                          Tue     656
                          Mon     629
                          Thu     619
BATTERY                   Sun     616
                                  ...
CRIMINAL SEXUAL ASSAULT   Sat       0
OTHER NARCOTIC VIOLATION  Wed       0
                          Thu       0
                          Sun       0
                          Tue       0
Length: 224, dtype: int64
```

In [189…
```
# ANALYSIS OF CRIME PRIMARY TYPE AND DAYTYPE


dfFinal.groupby(by=['Primary_Type', 'DayType']).size().sort_values(ascending=False)
```

Out[189]:
```
Primary_Type              DayType
THEFT                     Weekday    3149
BATTERY                   Weekday    2532
CRIMINAL DAMAGE           Weekday    1561
NARCOTICS                 Weekday    1488
BATTERY                   Weekend    1186
                                     ...
CRIMINAL SEXUAL ASSAULT   Weekend       1
NON - CRIMINAL            Weekend       0
RITUALISM                 Weekend       0
PUBLIC INDECENCY          Weekend       0
OTHER NARCOTIC VIOLATION  Weekend       0
Length: 64, dtype: int64
```

In [190…
```
# ANALYSIS OF CRIME PRIMARY TYPE AND MONTH


dfFinal.groupby(by=['Primary_Type', 'Month']).size().sort_values(ascending=False)
```

Out[190]:
```
Primary_Type              Month
THEFT                     Jul      420
                          Aug      411
                          May      381
                          Oct      377
BATTERY                   May      371
                                   ...
OTHER NARCOTIC VIOLATION  Feb        0
                          Dec        0
                          Aug        0
                          Apr        0
ARSON                     Apr        0
Length: 384, dtype: int64
```

In [191…
```
# ANALYSIS OF CRIME PRIMARY TYPE AND PERIOD


dfFinal.groupby(by=['Primary_Type', 'Period']).size().sort_values(ascending=False)
```

Out[191]:
```
Primary_Type              Period
THEFT                     Morning      1527
                          Night        1481
BATTERY                   Morning      1427
                          Night        1404
NARCOTICS                 Night         998
                                        ...
RITUALISM                 Night           0
                          Evening         0
                          Afternoon       0
PUBLIC INDECENCY          Evening         0
OTHER NARCOTIC VIOLATION  Evening         0
Length: 128, dtype: int64
```

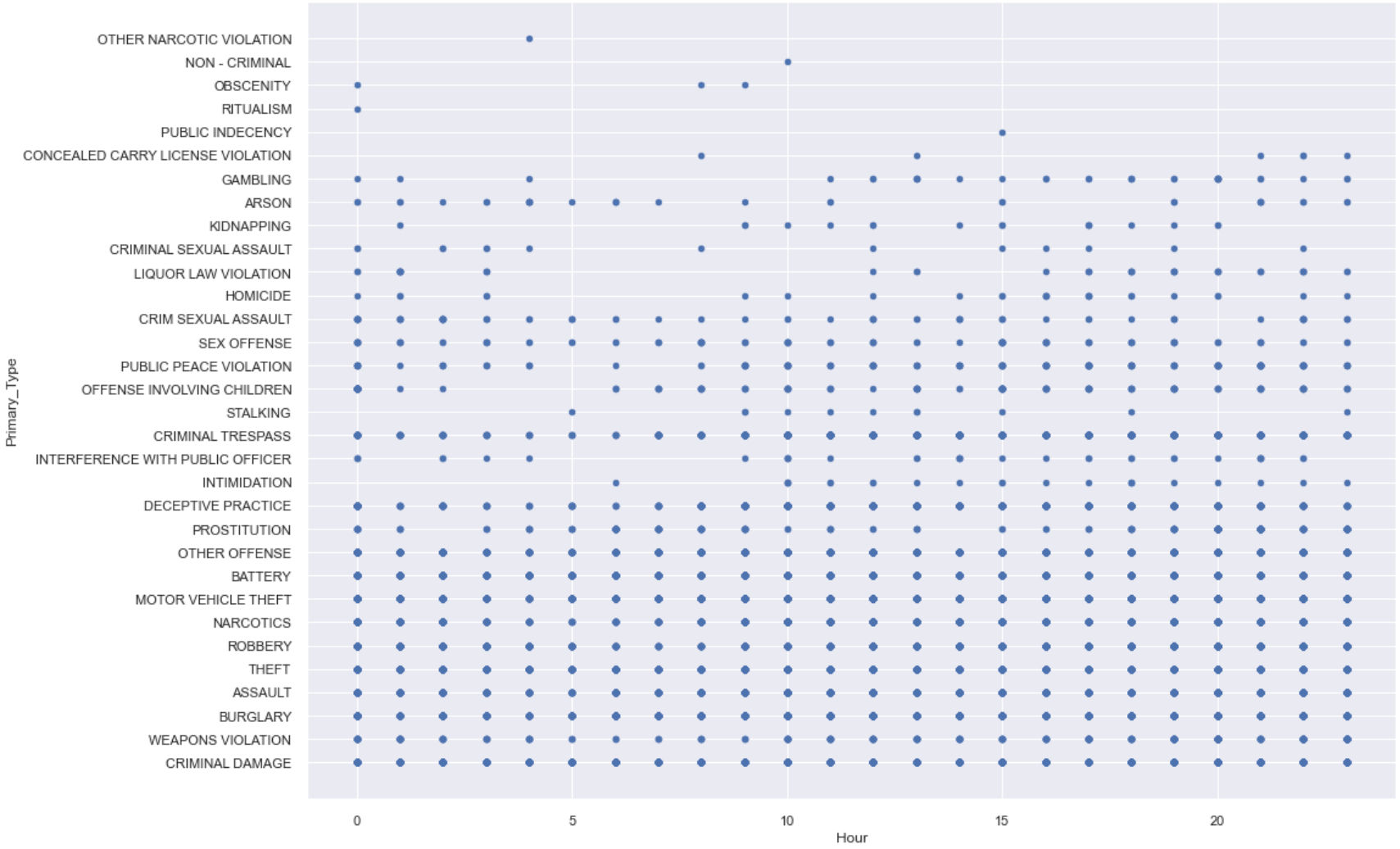
In [192…
```
# ANALYSIS OF CRIME PRIMARY TYPE AND HOUR OF THE DAY


dfFinal.groupby(by=['Primary_Type', 'Hour']).size().sort_values(ascending=False)
```

Out[192]:
```
Primary_Type                      Hour
THEFT                             12      328
                                  15      286
                                  14      260
                                  18      259
                                  0       254
                                         ...
INTERFERENCE WITH PUBLIC OFFICER  6         0
                                  5         0
HOMICIDE                          11        0
PROSTITUTION                      2         0
INTERFERENCE WITH PUBLIC OFFICER  8         0
Length: 768, dtype: int64
```

In [193…
```
# Scatter plot of relationship between location description and hour

sns.set(rc={'figure.figsize':(16,12)})
dfFinal[dfFinal["Hour"] < 90].plot.scatter(x="Hour", y="Primary_Type")
```

```
*c* argument looks like a single numeric RGB or RGBA sequence, which should be avoided as value-mapping will have prece
dence in case its length matches with *x* & *y*.  Please use the *color* keyword-argument or provide a 2D array with a
single row if you intend to specify the same RGB or RGBA value for all points.
```

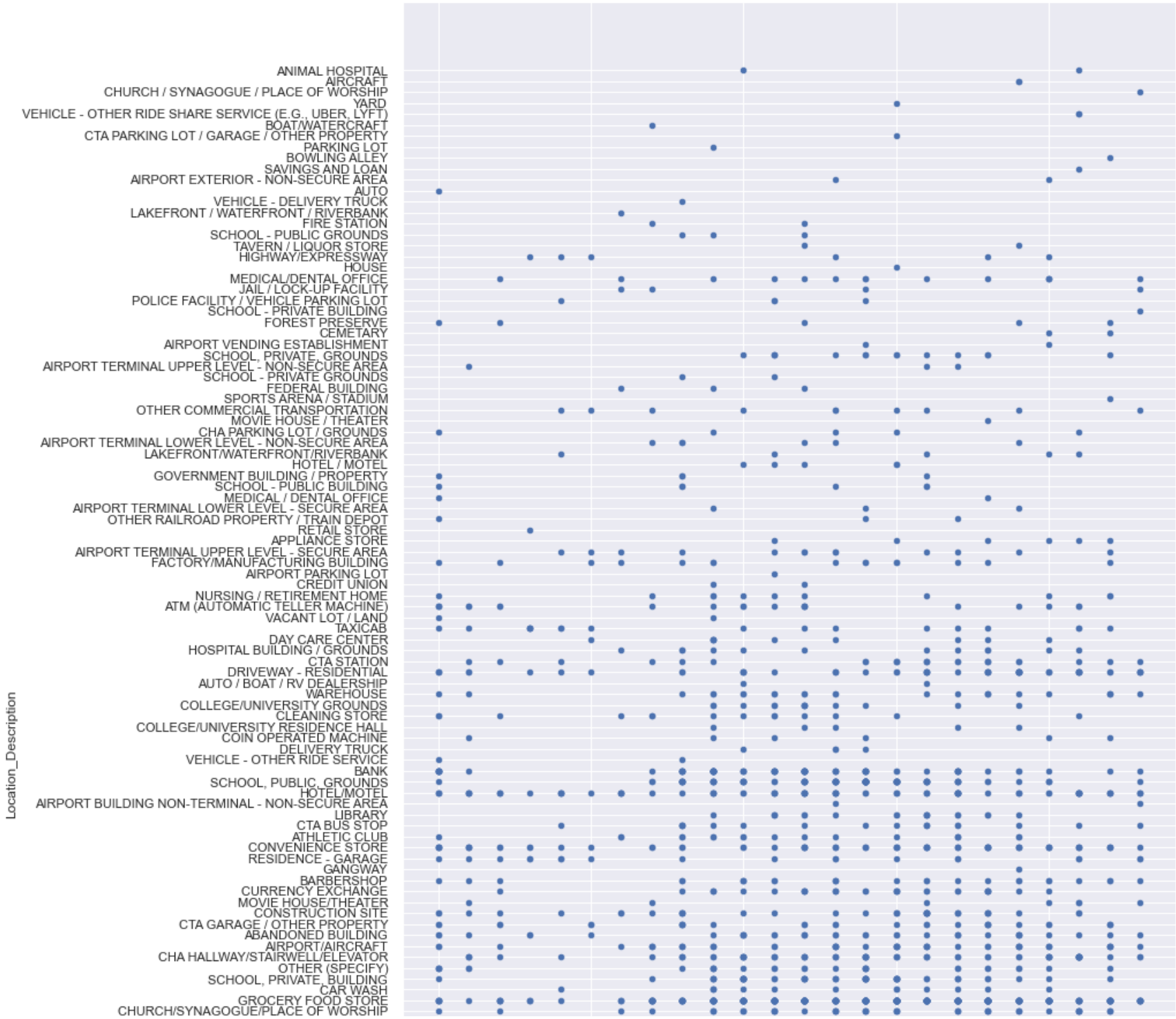
Out[193]:
```
<AxesSubplot:xlabel='Hour', ylabel='Primary_Type'>
```

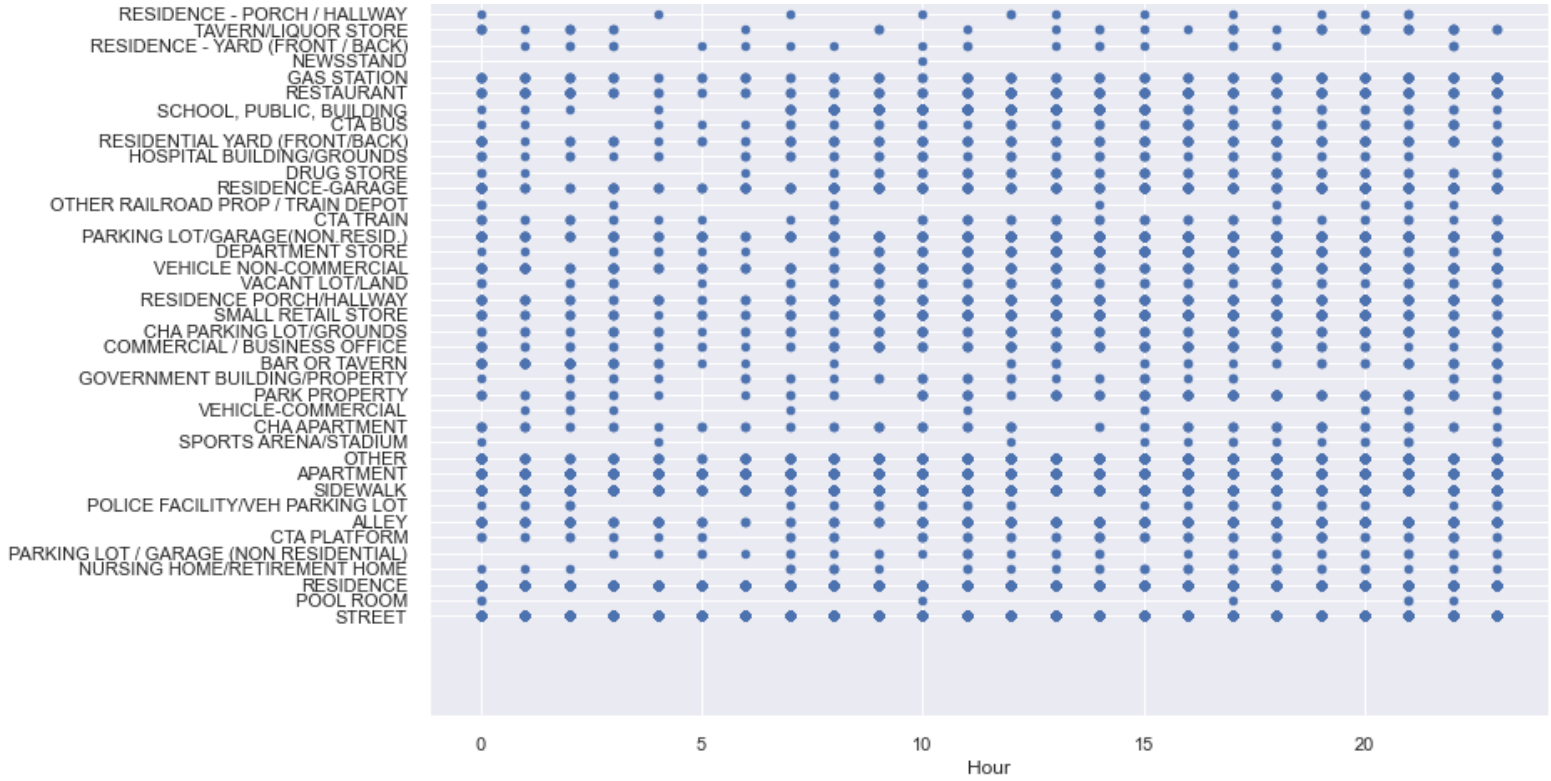


In [194…
```
# Scatter plot of relationship between location description and hour
sns.set(rc={'figure.figsize':(12,24)})
dfFinal[dfFinal["Hour"] < 24].plot.scatter(x="Hour", y="Location_Description")
```

```
*c* argument looks like a single numeric RGB or RGBA sequence, which should be avoided as value-mapping will have precedence in case its length matches with *x* & *y*.  Please use the *color* keyword-argument or provide a 2D array with a single row if you intend to specify the same RGB or RGBA value for all points.
```

Out[194]:
```
<AxesSubplot:xlabel='Hour', ylabel='Location_Description'>
```

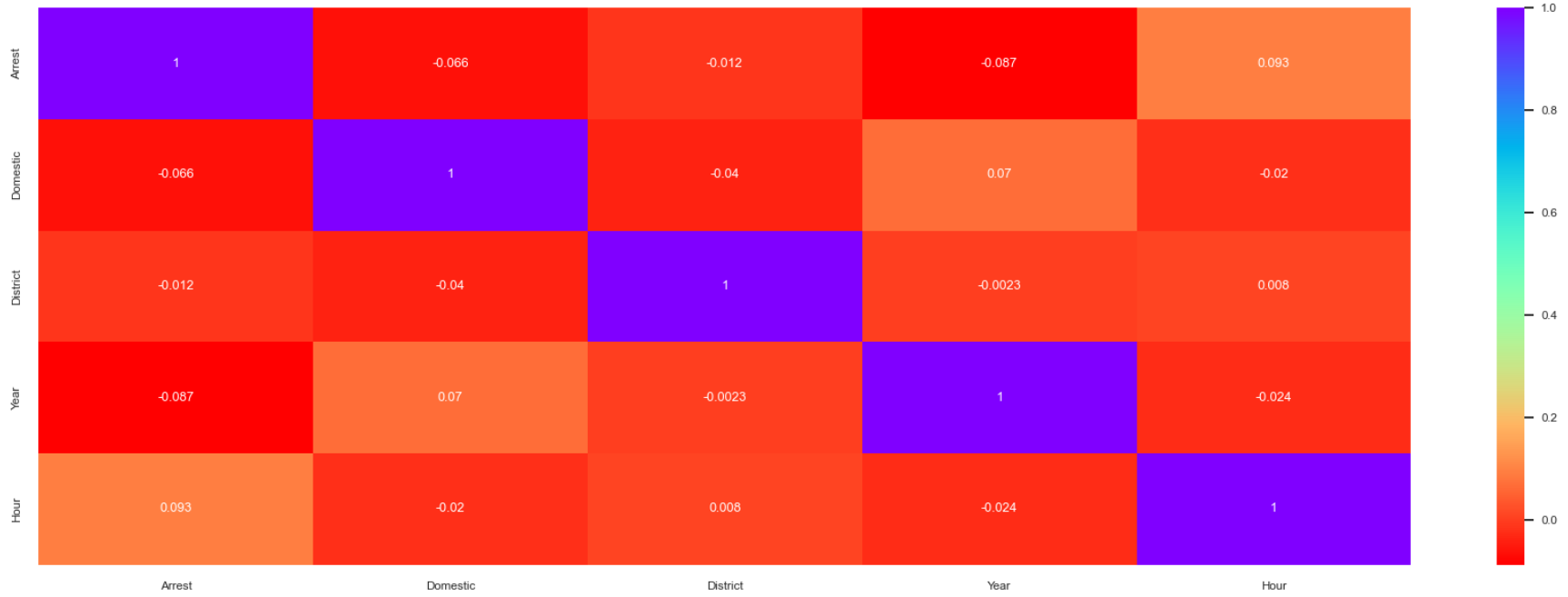

Location_Description
ANIMAL HOSPITAL
AIRCRAFT
CHURCH / SYNAGOGUE / PLACE OF WORSHIP
YARD
VEHICLE - OTHER RIDE SHARE SERVICE (E.G., UBER, LYFT)
BOAT/WATERCRAFT
CTA PARKING LOT / GARAGE / OTHER PROPERTY
PARKING LOT
BOWLING ALLEY
SAVINGS AND LOAN
AIRPORT EXTERIOR - NON-SECURE AREA
AUTO
VEHICLE - DELIVERY TRUCK
LAKEFRONT / WATERFRONT / RIVERBANK
FIRE STATION
SCHOOL - PUBLIC GROUNDS
TAVERN / LIQUOR STORE
HIGHWAY/EXPRESSWAY
HOUSE
MEDICAL/DENTAL OFFICE
JAIL / LOCK-UP FACILITY
POLICE FACILITY / VEHICLE PARKING LOT
SCHOOL - PRIVATE BUILDING
FOREST PRESERVE
CEMETARY
AIRPORT VENDING ESTABLISHMENT
SCHOOL, PRIVATE, GROUNDS
AIRPORT TERMINAL UPPER LEVEL - NON-SECURE AREA
SCHOOL - PRIVATE GROUNDS
FEDERAL BUILDING
SPORTS ARENA / STADIUM
OTHER COMMERCIAL TRANSPORTATION
MOVIE HOUSE / THEATER
CHA PARKING LOT / GROUNDS
AIRPORT TERMINAL LOWER LEVEL - NON-SECURE AREA
LAKEFRONT/WATERFRONT/RIVERBANK
HOTEL / MOTEL
GOVERNMENT BUILDING / PROPERTY
SCHOOL - PUBLIC BUILDING
MEDICAL / DENTAL OFFICE
AIRPORT TERMINAL LOWER LEVEL - SECURE AREA
OTHER RAILROAD PROPERTY / TRAIN DEPOT
RETAIL STORE
APPLIANCE STORE
AIRPORT TERMINAL UPPER LEVEL - SECURE AREA
FACTORY/MANUFACTURING BUILDING
AIRPORT PARKING LOT
CREDIT UNION
NURSING / RETIREMENT HOME
ATM (AUTOMATIC TELLER MACHINE)
VACANT LOT / LAND
TAXICAB
DAY CARE CENTER
HOSPITAL BUILDING / GROUNDS
CTA STATION
DRIVEWAY - RESIDENTIAL
AUTO / BOAT / RV DEALERSHIP
WAREHOUSE
COLLEGE/UNIVERSITY GROUNDS
CLEANING STORE
COLLEGE/UNIVERSITY RESIDENCE HALL
COIN OPERATED MACHINE
DELIVERY TRUCK
VEHICLE - OTHER RIDE SERVICE
BANK
SCHOOL, PUBLIC, GROUNDS
HOTEL/MOTEL
AIRPORT BUILDING NON-TERMINAL - NON-SECURE AREA
LIBRARY
CTA BUS STOP
ATHLETIC CLUB
CONVENIENCE STORE
RESIDENCE - GARAGE
GANGWAY
BARBERSHOP
CURRENCY EXCHANGE
MOVIE HOUSE/THEATER
CONSTRUCTION SITE
CTA GARAGE / OTHER PROPERTY
ABANDONED BUILDING
AIRPORT/AIRCRAFT
CHA HALLWAY/STAIRWELL/ELEVATOR
OTHER (SPECIFY)
SCHOOL, PRIVATE, BUILDING
CAR WASH
GROCERY FOOD STORE
CHURCH/SYNAGOGUE/PLACE OF WORSHIP



In [195…
```
# Evaluating the correlation between features (Columns) using a Heatman

# Plot the heatmap for all the features in the Dataframe by using sns.heatmap and
# keep the figure size as 30,10 for better visibility:

sns.set(rc={'figure.figsize':(30,10)})
sns.set_context("talk", font_scale=0.7)
```

In [196…
```
# Use Spearman as the method parameter to conpute Spearman's rank correlation coefficient:

sns.heatmap(dfFinal.loc[:,:].corr(method='spearman'),cmap='rainbow_r', annot=True)
```

Out[196]:
```
<AxesSubplot:>
```



In [197…
```
# Loading the dataset into jupyter notebook.

fileName = "crime_data_Proj1.CSV"

filePath ="C:/Users/OLUWATRIUMPH/Documents/Myworkspace/"
df = pd.read_csv(filePath + fileName)
```

In [198…
```
#dfFinal = pd.read_csv('https://plotly.github.io/datasets/country_indicators.csv')
available_indicators = dfFinal['District'].unique()
```

In [199…
```
# Dataframes for each crime (Will be used in the further parts of our analysis)
loc_to_change  = list(dfFinal['Location_Description'].value_counts()[15:].index)
desc_to_change = list(dfFinal['Description'].value_counts()[5:].index)
type_to_change = list(dfFinal['Primary_Type'].value_counts()[5:].index)

dfSub.loc[dfFinal['Location_Description'].isin(loc_to_change) , dfFinal.columns=='Location_Description'] = 'OTHER'
dfSub.loc[dfFinal['Description'].isin(desc_to_change) , dfFinal.columns=='Description'] = 'OTHER'
dfSub.loc[dfFinal['Primary_Type'].isin(type_to_change) , dfFinal.columns=='Primary_Type'] = 'OTHER'
```

In [200…
```
#### Converting those columns in 'Categorical' data type

dfFinal['Primary_Type']         = pd.Categorical(dfFinal['Primary_Type'])
```

```
dfFinal['Location_Description'] = pd.Categorical(dfFinal['Location_Description'])
dfFinal['Description']          = pd.Categorical(dfFinal['Description'])
```

In [201…
```
dfFinal.Month.unique()
```

Out[201]:
```
array(['Jul', 'Apr', 'Aug', 'Oct', 'Dec', 'Jun', 'Jan', 'Nov', 'Mar',
       'Sep', 'May', 'Feb'], dtype=object)
```

In [202…
```
dfFinal.pivot_table(dfFinal,index =['Location_Description', 'Description'],columns=['Month'],aggfunc=[np.sum])
```

Out[202]:

		Arrest	...																	
	Month	Apr	Aug	Dec	Feb	Jan	Jul	Jun	Mar	May	Nov	...	Dec	Feb	Jan	Jul	Jun	Mar	May	Nov
Location_Description	Description																			
ABANDONED BUILDING	$500 AND UNDER	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0
	AGG CRIM SEX ABUSE FAM MEMBER	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0
	AGG CRIMINAL SEXUAL ABUSE	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	2003	0	0	0
	AGG PO HANDS ETC SERIOUS INJ	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0
	AGG PO HANDS NO/MIN INJURY	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
YARD	VEHICULAR HIJACKING	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0
	VIOLATE ORDER OF PROTECTION	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0
	VIOLATION OF BAIL BOND - DOMESTIC VIOLENCE	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0

		Arrest	...																	
	Month	Apr	Aug	Dec	Feb	Jan	Jul	Jun	Mar	May	Nov	...	Dec	Feb	Jan	Jul	Jun	Mar	May	Nov
Location_Description	Description																			
	VIOLATION OF CIVIL NO CONTACT ORDER	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0
	VIOLATION OF SUMMARY CLOSURE	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0

37800 rows × 60 columns

In [203…

```
#             NOTE TO-DO

####      GRAPHS TO PLOT
# 1. (PREDOMINANT CRIME TYPE IN A LOCATION).
# 2. PRIMARY TYPRE OF CRIME TO DISTRICT (PREDOMINANT CRIME TO EACH DISTRICT...TOP 5)
# 3. DESCRIPTION TO LOCATION DESCRIPTION (THAT IS, LOCATION OF CRIME OCCURRENCE)
# 4. DISTRICT WITH THE HIGHEST ARREST OF CRIME (AND DISTRICT WITH LEAST ARREST)
# 5. PERIOD OF THE DAY WITH PREDOMINANT CRIMES TYPES... (WHICH CRIME HAPPENS AT CERTAIN PERIOD OF THE DAY)
# 6.




###N ARRANGEMENT

# arrange by location or places or district or domestic
# arrange by year, month, days of the week and daytype
# arrange by primary type  (i.e, types and kinds of crimes), descriition,
# arrange by crime rate by arrest
```

In [204…

```
# Let's order the above dictionaries for proper plotting
# months=['January','February','March','April','May','June','July','August','September','October','November','December'
```

```
# 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 [205…

```
##### I believe this data analytics project reveals the situational and scientific view about the security status and
#### crime rate of the Chicago city.
```

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]: