1. Importing Packages

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

import datetime as dt
import seaborn as sns
# import gmplot

from pylab import rcParams
%matplotlib inline
rcParams["figure.figsize"] = 12, 8
```

2. Reading the Data

In [2]:
```
crime = pd.read_csv("Crime_Data_from_2020_to_Presen.csv")
```

3. Previewing the Data Variables

3.1. Shape

In [3]:
```
print("The shape is {}".format(crime.shape))
```

```
The shape is (467654, 28)
```

In [4]:
```
crime.tail(3)
```

Out[4]:

	DR_NO	Date Rptd	DATE OCC	TIME OCC	AREA	AREA NAME	Rpt Dist No	Part 1-2	Crm Cd	Crm Cd Desc
467651	221005507	02/10/2022 12:00:00 AM	02/09/2022 12:00:00 AM	1530	10	West Valley	1024	1	510	VEHICLE - STOLEN
467652	221105477	02/10/2022 12:00:00 AM	02/08/2022 12:00:00 AM	2000	11	Northeast	1171	1	510	VEHICLE - STOLEN
467653	221605448	02/15/2022 12:00:00 AM	02/14/2022 12:00:00 AM	1800	16	Foothill	1613	1	331	THEFT FROM MOTOR VEHICLE - GRAND ($950.01 AND ...

3 rows × 28 columns

3.2. Date of Crime Reported and Crime Occurred

In [5]:
```
crime['DATE OCC'] = pd.to_datetime(crime['DATE OCC'])
crime['Date Rptd'] = pd.to_datetime(crime['Date Rptd'])
try:
    date_reported = [dt.datetime.strptime(d, "%y/%m/%d").date() for d in crime["
except:
    print("Already converted Date Reported")

try:
    date_occurred = [dt.datetime.strptime(d, "%y/%m/%d").date() for d in crime["
except:
    print("Already converted Date Occurred")
```

```
Already converted Date Reported
Already converted Date Occurred
```

In [6]:
```
# Making lists of days, months, and years for reported from datetime objects
day_reported = [d.isoweekday() for d in crime["Date Rptd"]]
mon_reported = [d.month for d in crime["Date Rptd"]]
year_reported = [d.year for d in crime["Date Rptd"]]

day_occurred = [d.isoweekday() for d in crime["DATE OCC"]]
mon_occurred = [d.month for d in crime["DATE OCC"]]
year_occurred = [d.year for d in crime["DATE OCC"]]
# Making new columns for each
crime["Day Reported"] = np.array(day_reported)
crime["Month Reported"] = np.array(mon_reported)
crime["Year Reported"] = np.array(year_reported)

crime["Day Occurred"] = np.array(day_occurred)
crime["Month Occurred"] = np.array(mon_occurred)
crime["Year Occurred"] = np.array(year_occurred)
crime
```

Out[6]:

	DR_NO	Date Rptd	DATE OCC	TIME OCC	AREA	AREA NAME	Rpt Dist No	Part 1-2	Crm Cd	Crm
0	10304468	2020-01-08	2020-01-08	2230	3	Southwest	377	2	624	BATTERY - SIMPLE
1	190101086	2020-01-02	2020-01-01	330	1	Central	163	2	624	BATTERY - SIMPLE
2	191501505	2020-01-01	2020-01-01	1730	15	N Hollywood	1543	2	745	VAN MISDEAMEANOR
3	191921269	2020-01-01	2020-01-01	415	19	Mission	1998	2	740	VANDALISM - FELC & OVER, ALL CHU
4	200100501	2020-01-02	2020-01-01	30	1	Central	163	1	121	RAPE,
...	...	...	...	...	...	...	...	...	...	
467649	221907283	2022-03-21	2022-03-20	100	19	Mission	1901	1	341	THEFT-GRAND (OVER)EXCPT,GUNS,

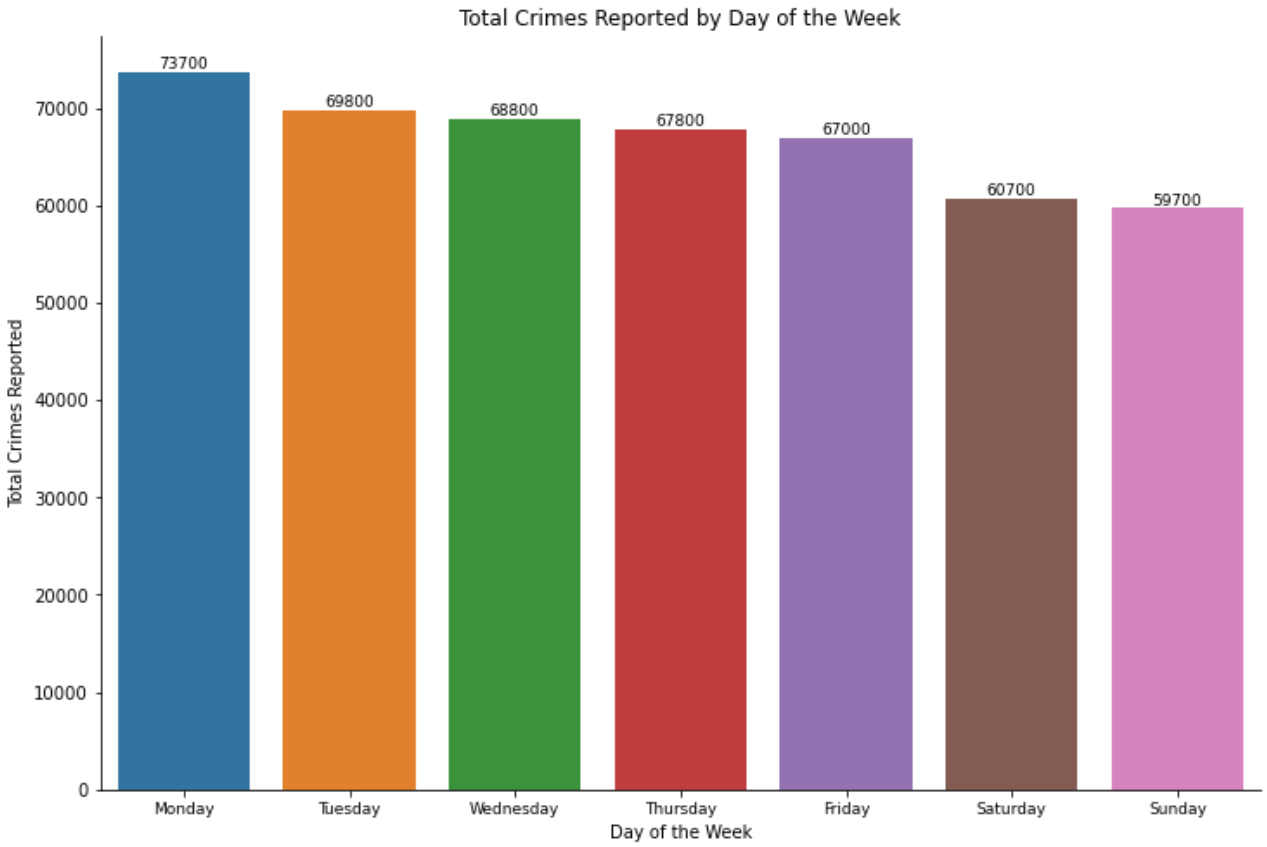
	DR_NO	Date Rptd	DATE OCC	TIME OCC	AREA	AREA NAME	Rpt Dist No	Part 1-2	Crm Cd	Crm
467650	221906145	2022-02-23	2022-02-23	1210	19	Mission	1985	1	421	THEFT FRO VEHICLE -
467651	221005507	2022-02-10	2022-02-09	1530	10	West Valley	1024	1	510	VEHICLE
467652	221105477	2022-02-10	2022-02-08	2000	11	Northeast	1171	1	510	VEHICLE
467653	221605448	2022-02-15	2022-02-14	1800	16	Foothill	1613	1	331	THEFT FRO VEHICLE - GRANE

467654 rows × 34 columns

3.2.1.1. Crime by Day of the Week

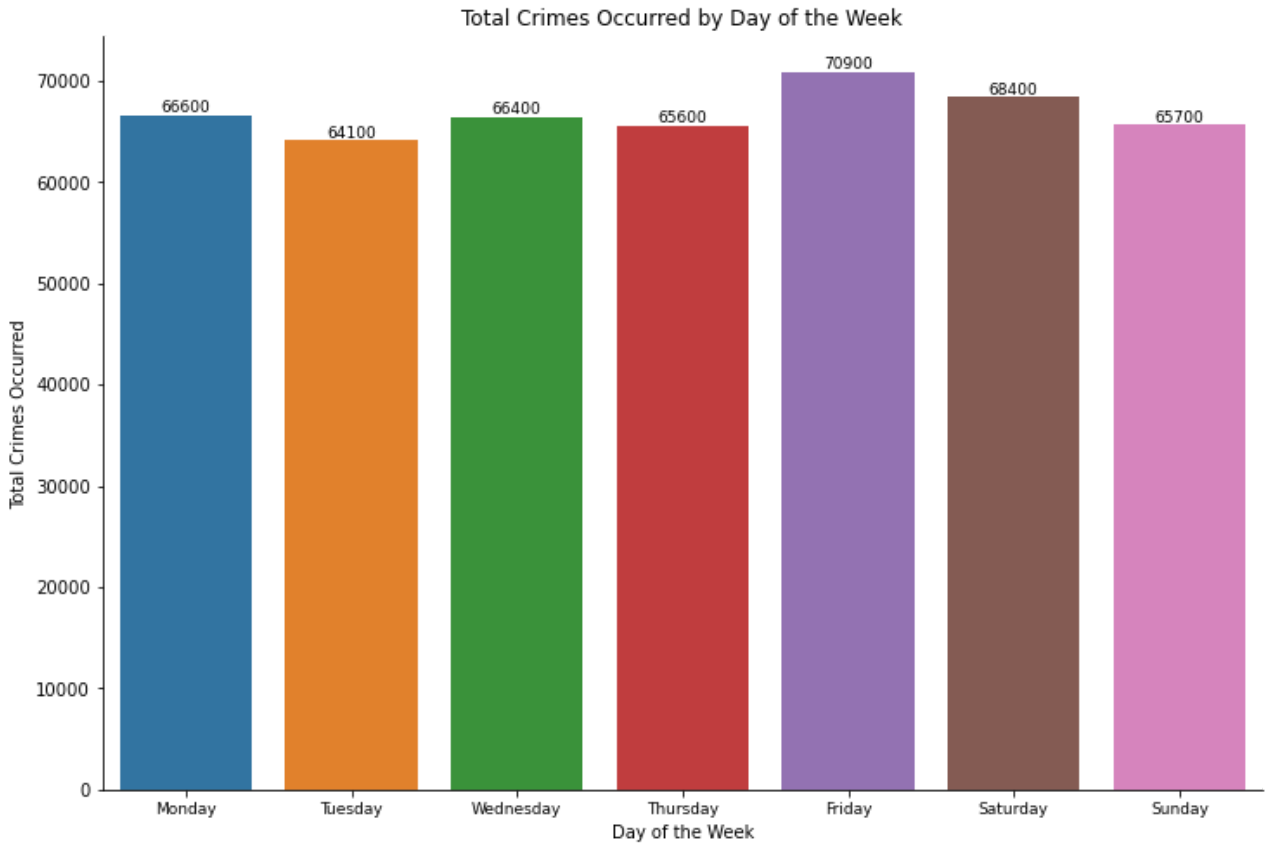
In [7]:

```
fig, ax = plt.subplots()
# Plotting crimes reported by day
sns.barplot(x=crime["Day Reported"].value_counts().index, y=crime["Day Reported"
# Axes
ax.set_title("Total Crimes Reported by Day of the Week")
ax.set_xticklabels(["Monday", "Tuesday", "Wednesday", "Thursday", "Friday", "Sat
ax.set_xlabel("Day of the Week")
ax.set_ylabel("Total Crimes Reported")
# Adding values
for p in ax.patches:
    ax.text(p.get_x() + p.get_width()/2., p.get_height(), '%d' % round(int(p.get
            fontsize=9, color='black', ha='center', va='bottom')
sns.despine()
```



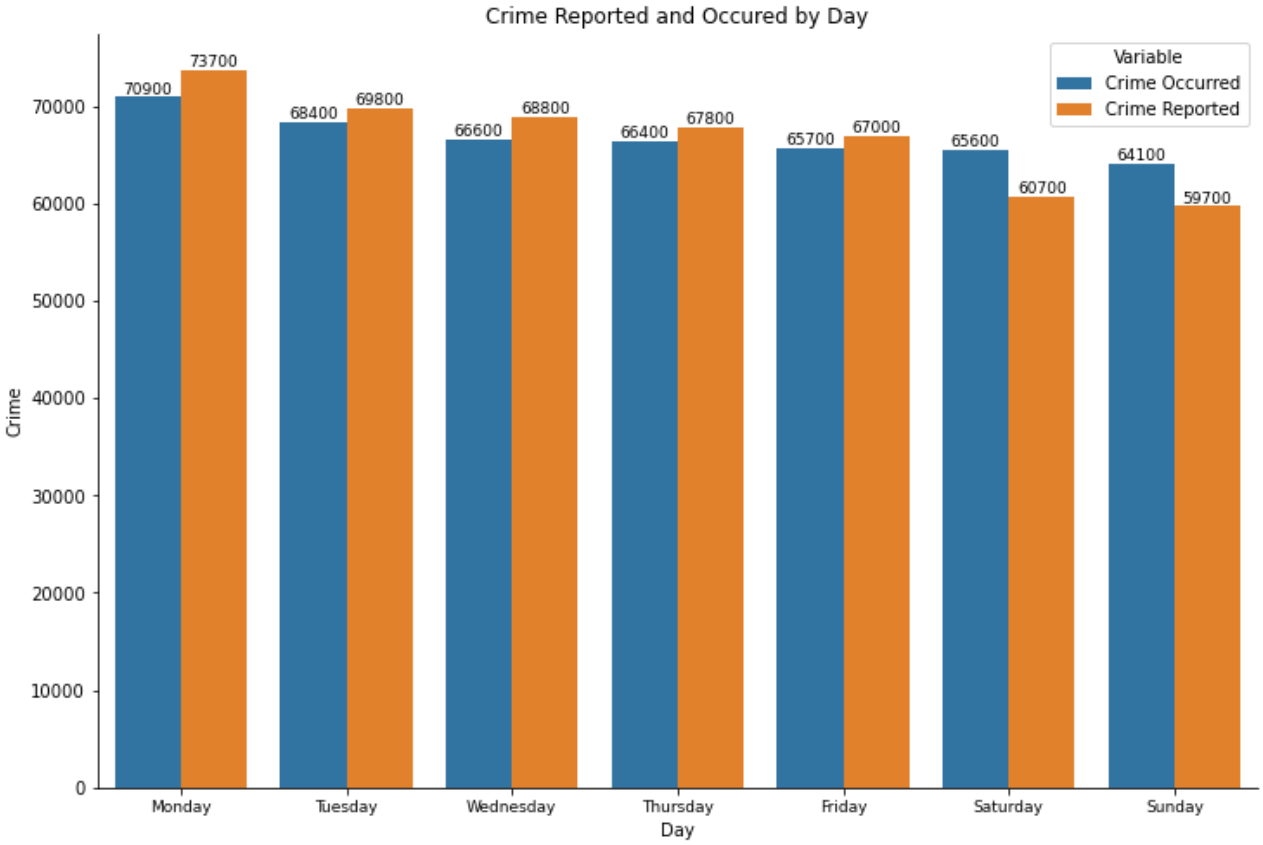
In [8]:

```
fig, ax = plt.subplots()
# Plotting crimes occurred by day
sns.barplot(x = crime["Day Occurred"].value_counts().index, y = crime["Day Occur
# Axes
ax.set_title("Total Crimes Occurred by Day of the Week")
ax.set_xticklabels(["Monday", "Tuesday", "Wednesday", "Thursday", "Friday", "Sat
ax.set_xlabel("Day of the Week")
ax.set_ylabel("Total Crimes Occurred")
# Adding values
for p in ax.patches:
    ax.text(p.get_x() + p.get_width()/2., p.get_height(), '%d' % round(int(p.get
            fontsize=9, color='black', ha='center', va='bottom')
sns.despine()
```



In [9]:
```
# Making a new dataframe
df1 = pd.DataFrame({
    "Day" : list(crime["Day Reported"].value_counts().index),
    "Crime Occurred" : list(crime["Day Occurred"].value_counts()),
    "Crime Reported" : list(crime["Day Reported"].value_counts())
})
dayrepocc = df1.set_index("Day").stack().reset_index().rename(columns={"level_1"
```

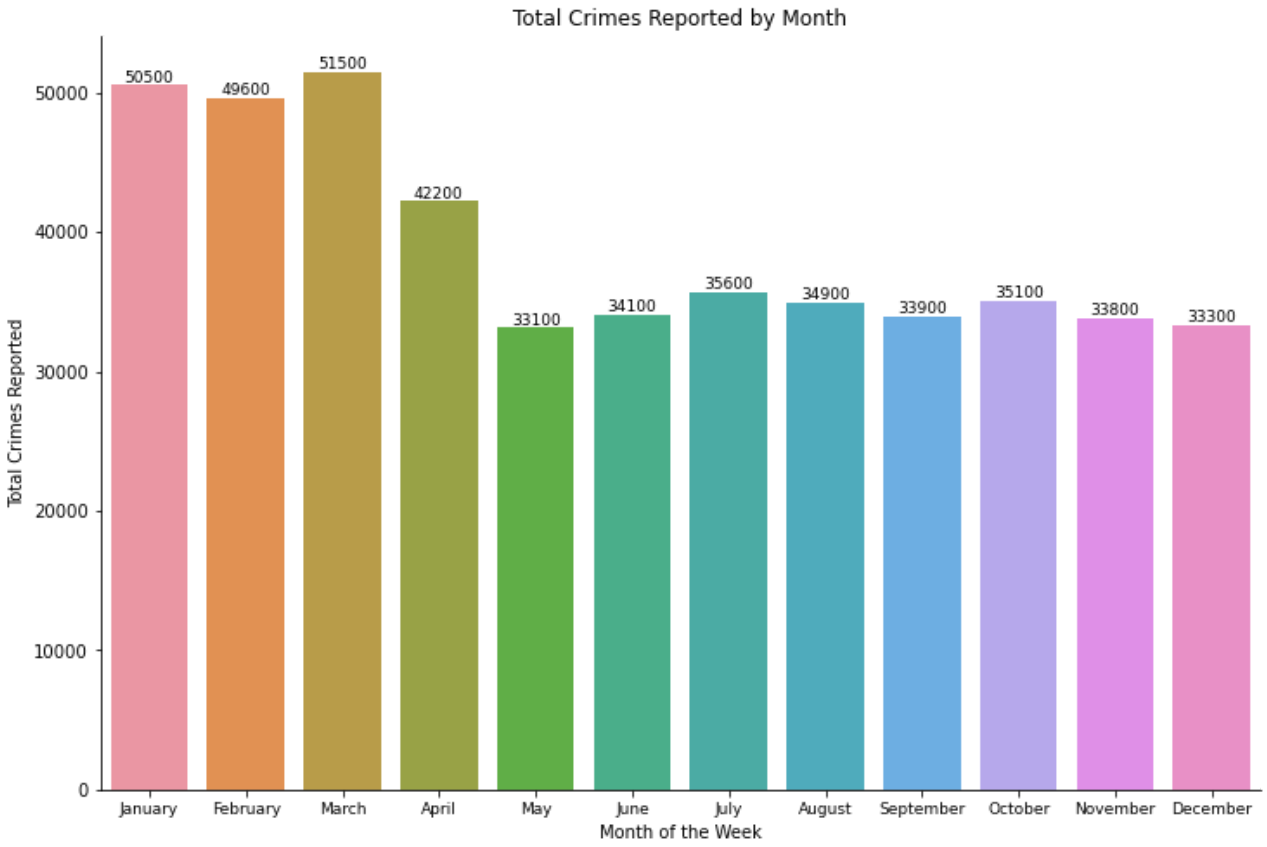
In [10]:
```
fig, ax = plt.subplots()
# Plotting side by side crime rep and occ by day
sns.barplot(x = "Day", y = "Crime", hue = "Variable", data=dayrepocc, ax=ax)
# Axes
ax.set_title("Crime Reported and Occured by Day")
ax.set_xticklabels(["Monday", "Tuesday", "Wednesday", "Thursday", "Friday", "Sat
ax.set_ylabel("Crime")
# Adding values
for p in ax.patches:
    ax.text(p.get_x() + p.get_width()/2., p.get_height(), '%d' % round(int(p.get
            fontsize=9, color='black', ha='center', va='bottom')
sns.despine(fig)
```



3.2.1.2.1. Crime Reported by Month

In [11]:

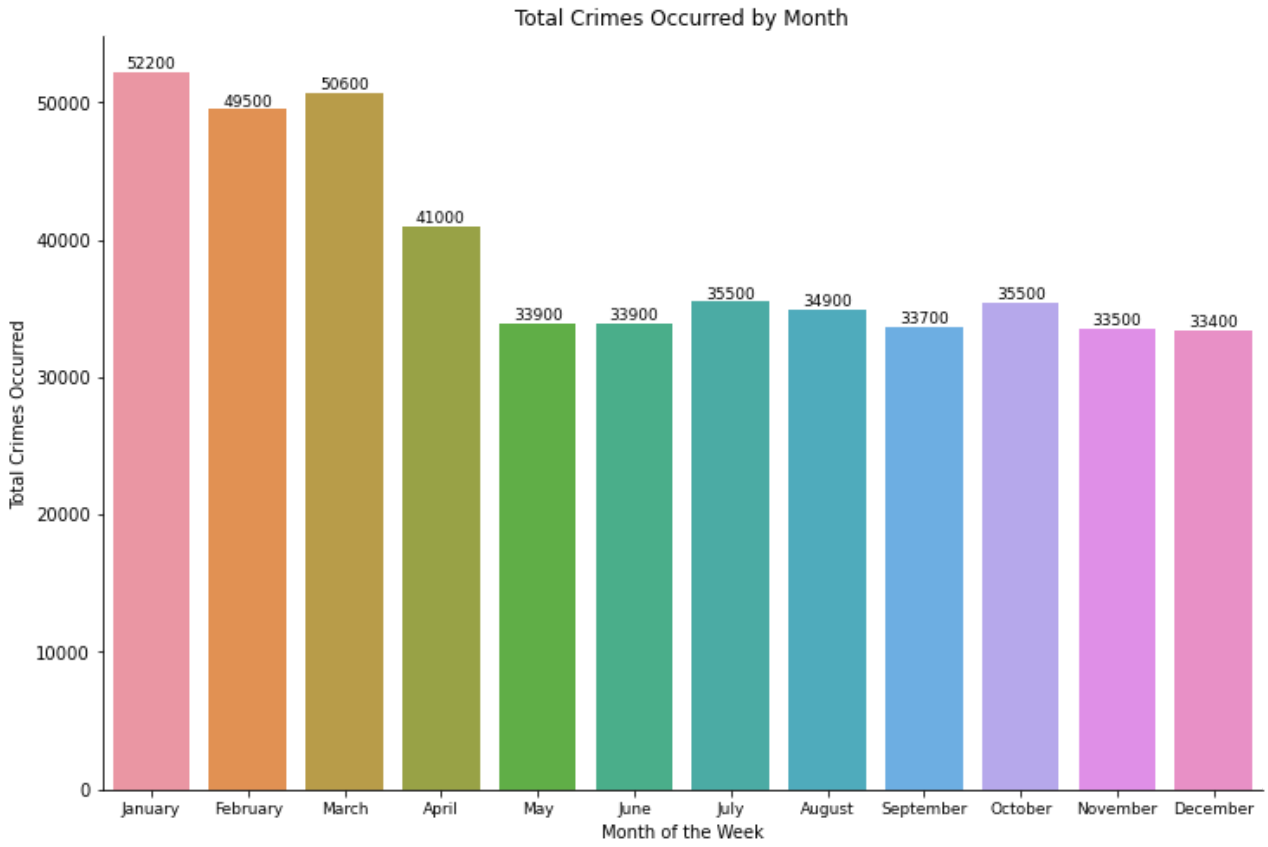
```
fig, ax = plt.subplots()
# Plotting crimes reported by month
sns.barplot(x = crime["Month Reported"].value_counts().index, y = crime["Month R
# Axes
ax.set_title("Total Crimes Reported by Month")
ax.set_xticklabels(["January", "February", "March", "April", "May", "June",
                    "July", "August", "September", "October", "November", "Decem
ax.set_xlabel("Month of the Week")
ax.set_ylabel("Total Crimes Reported")
# Adding Values
for p in ax.patches:
    ax.text(p.get_x() + p.get_width()/2., p.get_height(), '%d' % round(int(p.get
            fontsize=9, color='black', ha='center', va='bottom')
sns.despine()
```



Note that data is collected up until April 2022 which makes visualization difficult to ascertain

In [12]:

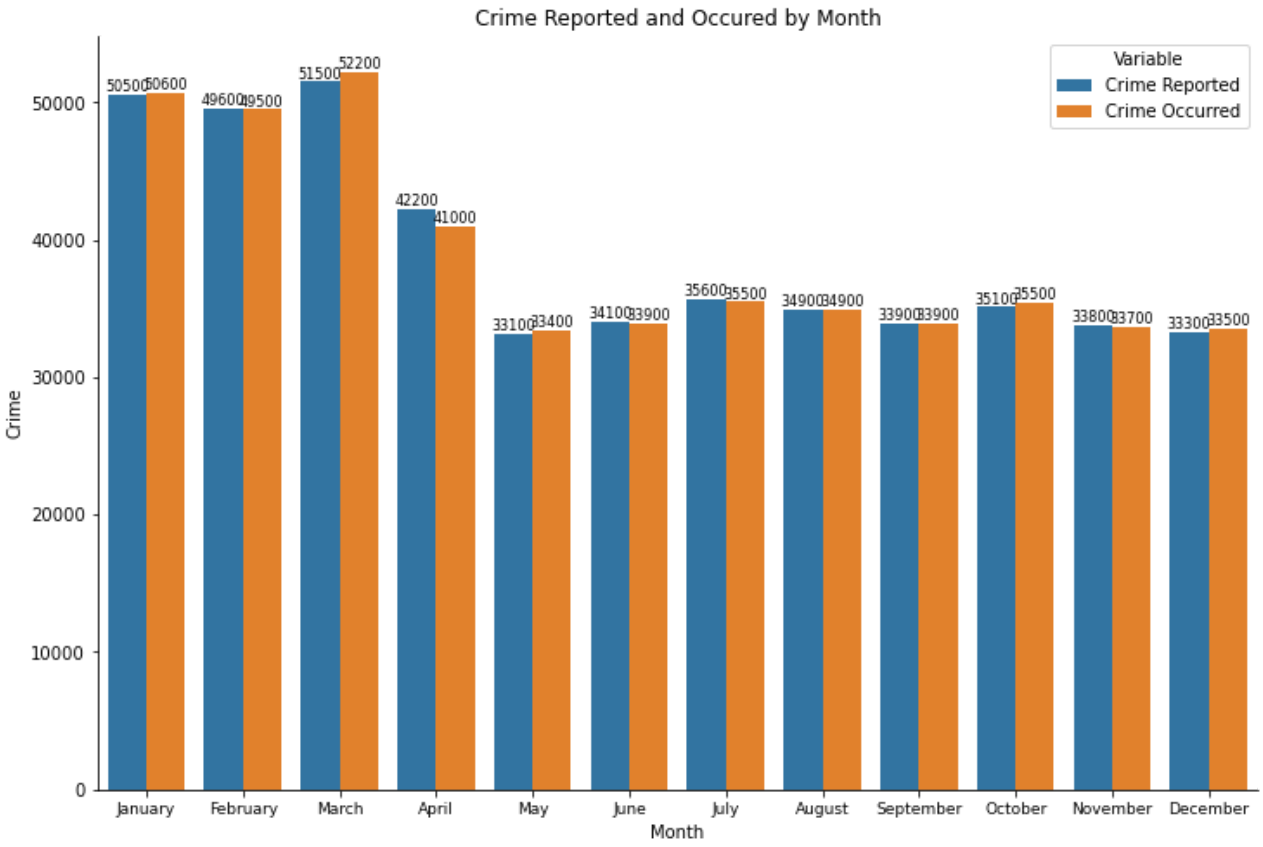
```
fig, ax = plt.subplots()
# Plotting crimes occurred by month
sns.barplot(x=crime["Month Occurred"].value_counts().index, y=crime["Month Occur
# Axes
ax.set_title("Total Crimes Occurred by Month")
ax.set_xticklabels(["January", "February", "March", "April", "May", "June",
                    "July", "August", "September", "October", "November", "Decem
ax.set_xlabel("Month of the Week")
ax.set_ylabel("Total Crimes Occurred")
# Adding values
for p in ax.patches:
    ax.text(p.get_x() + p.get_width()/2., p.get_height(), '%d' % round(int(p.get
            fontsize=9, color='black', ha='center', va='bottom')
sns.despine()
```



3.2.1.2.3. Comparing Crime Reported and Occurred by Month Side by Side¶

In [13]:
```
# Making a new dataframe
df2 = pd.DataFrame({
    "Month" : list(crime["Month Reported"].value_counts().index),
    "Crime Reported" : list(crime["Month Reported"].value_counts()),
    "Crime Occurred" : list(crime["Month Occurred"].value_counts())
})
monrepocc = df2.set_index("Month").stack().reset_index().rename(columns={"level_
```

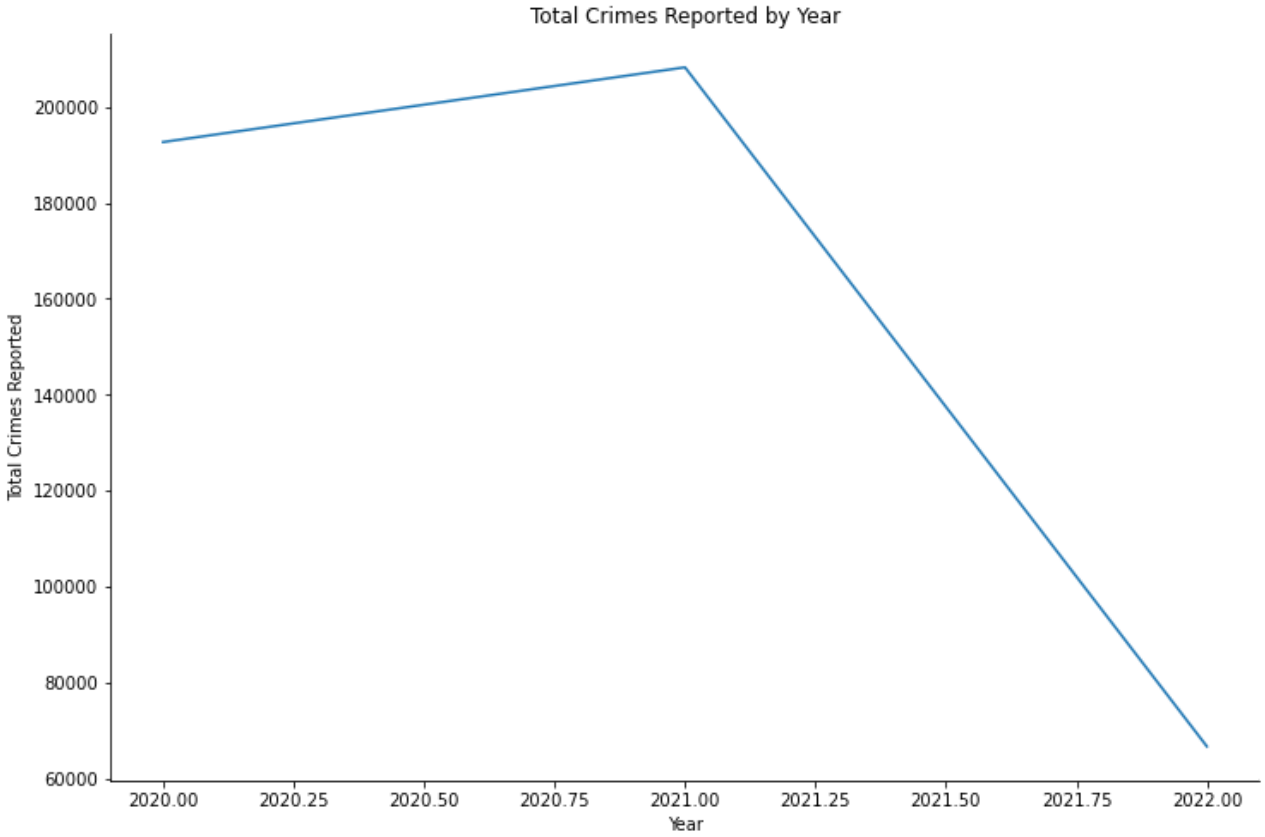
In [14]:
```
fig, ax = plt.subplots()
# Plotting side by side crime rep and occ by day
sns.barplot(x = "Month", y = "Crime", hue = "Variable", data=monrepocc, ax=ax)
# Axes
ax.set_title("Crime Reported and Occured by Month")
ax.set_xticklabels(["January", "February", "March", "April", "May", "June",
                    "July", "August", "September", "October", "November", "Decem
ax.set_ylabel("Crime")
# Adding values
for p in ax.patches:
    ax.text(p.get_x() + p.get_width()/2., p.get_height(), '%d' % round(int(p.get
            fontsize=8, color='black', ha='center', va='bottom')
sns.despine(fig)
```



3.2.1.3. Crime by Year

In [15]:

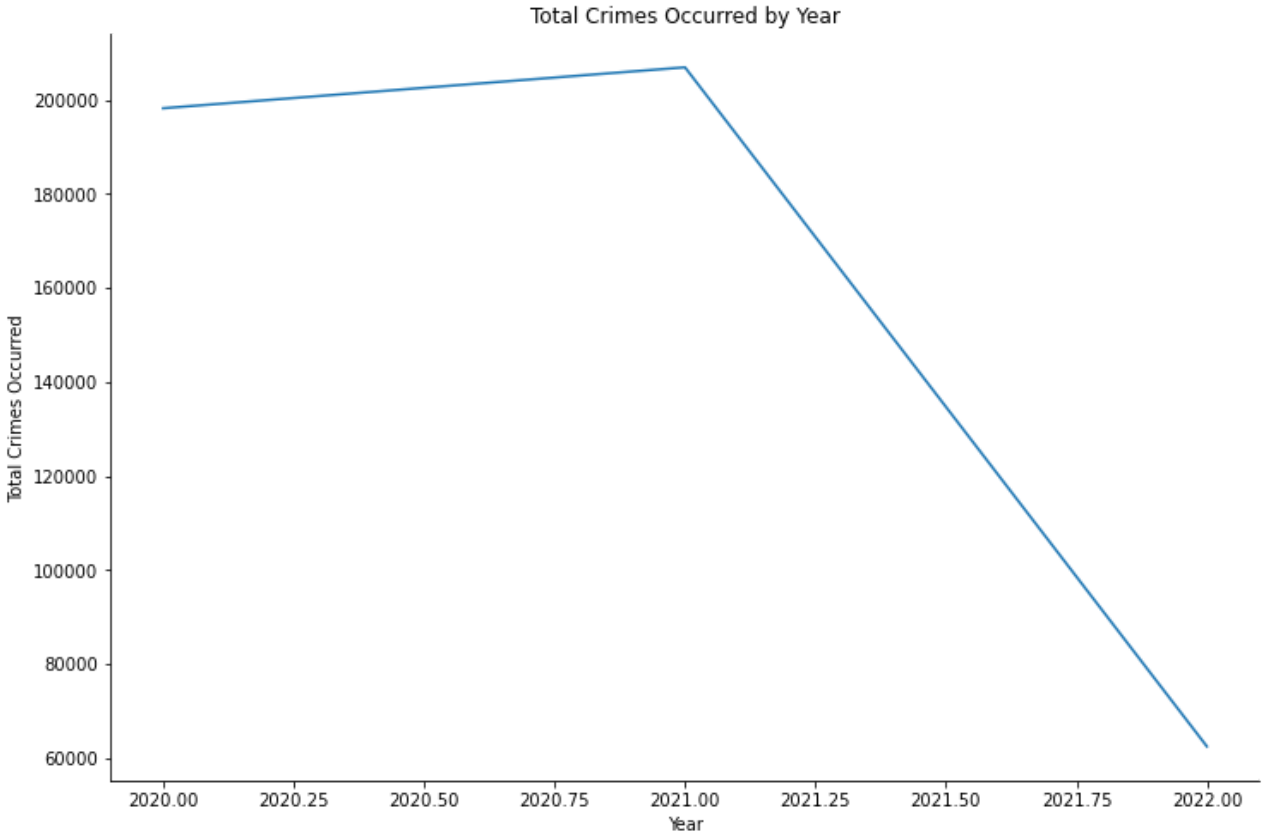
```
fig, ax = plt.subplots()
# Plotting crimes reported by year
plt.plot(crime["Year Reported"].value_counts().sort_index().index, crime["Year R
# Axes
ax.set_title("Total Crimes Reported by Year")
ax.set_xlabel("Year")
ax.set_ylabel("Total Crimes Reported")
sns.despine()
```



3.2.1.3.2. Crime Occurred by Year

In [16]:

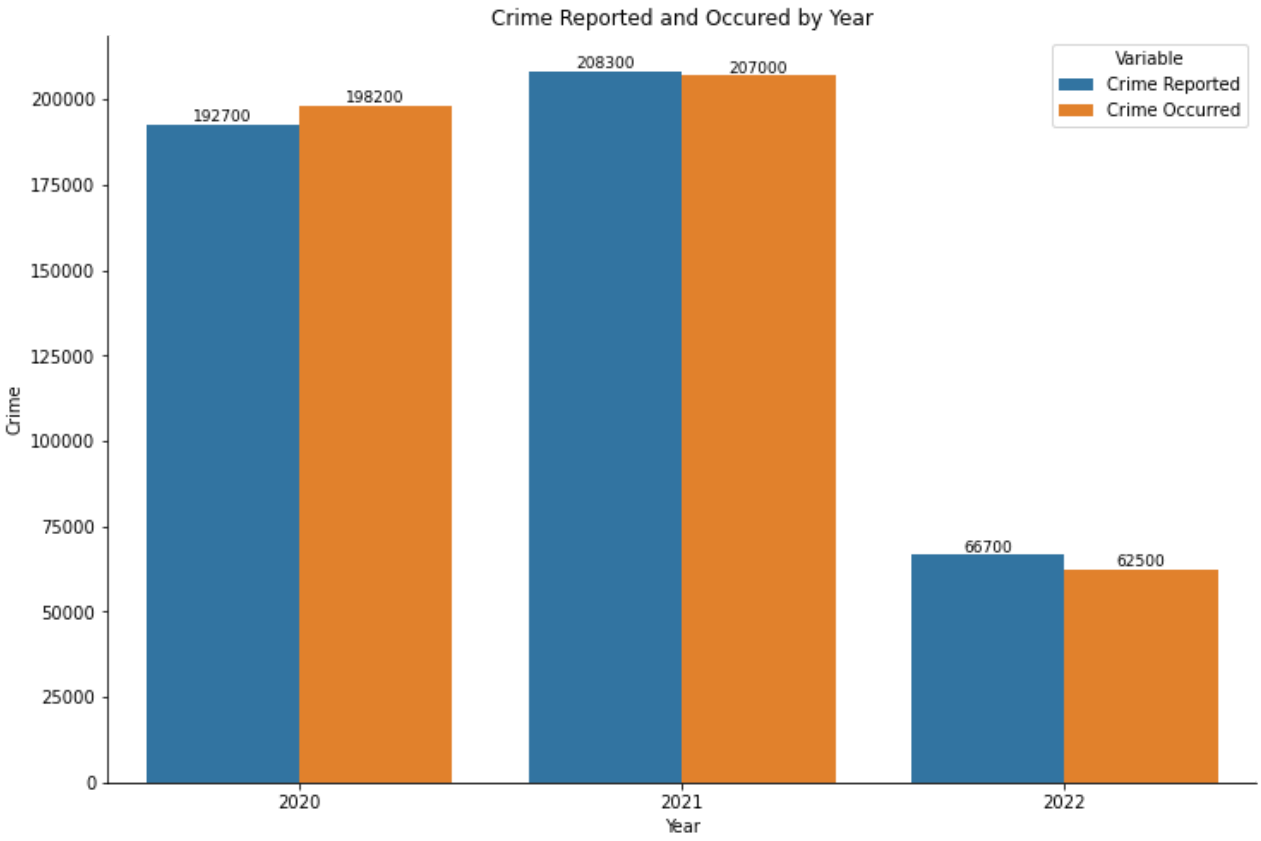
```
fig, ax = plt.subplots()
# Plotting crimes occured by year
plt.plot(crime["Year Occurred"].value_counts().sort_index().index, crime["Year O
# AXes
ax.set_title("Total Crimes Occurred by Year")
ax.set_xlabel("Year")
ax.set_ylabel("Total Crimes Occurred")
sns.despine()
```



3.2.1.3.3. Comparing Crime Reported and Occured by Year Side to Side

In [17]:
```
# Making a new dataframe
df3 = pd.DataFrame({
    "Year" : list(crime["Year Reported"].value_counts().index),
    "Crime Reported" : list(crime["Year Reported"].value_counts()),
    "Crime Occurred" : list(crime["Year Occurred"].value_counts())
})
yearrepocc = df3.set_index("Year").stack().reset_index().rename(columns={"level_
```

In [18]:
```
fig, ax = plt.subplots()
# Plotting side by side crime rep and occ by day
sns.barplot(x = "Year", y = "Crime", hue = "Variable", data=yearrepocc, ax=ax)
# Axes
ax.set_title("Crime Reported and Occured by Year")
ax.set_ylabel("Crime")
# Adding values
for p in ax.patches:
    ax.text(p.get_x() + p.get_width()/2., p.get_height(), '%d' % round(int(p.get
            fontsize=9, color='black', ha='center', va='bottom')
sns.despine(fig)
```

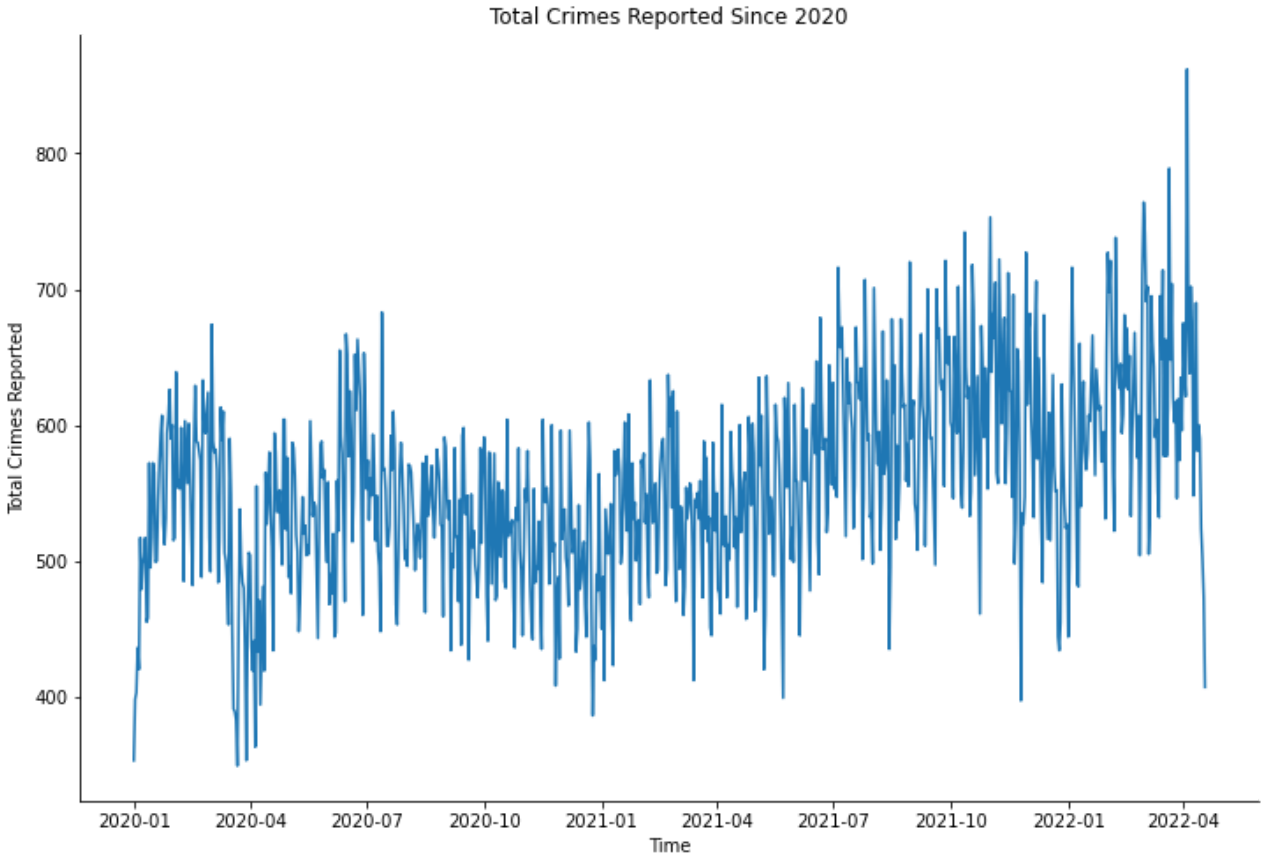


3.2.2. Crime Chronologically

3.2.2.1. Crimes Over Time

In [19]:

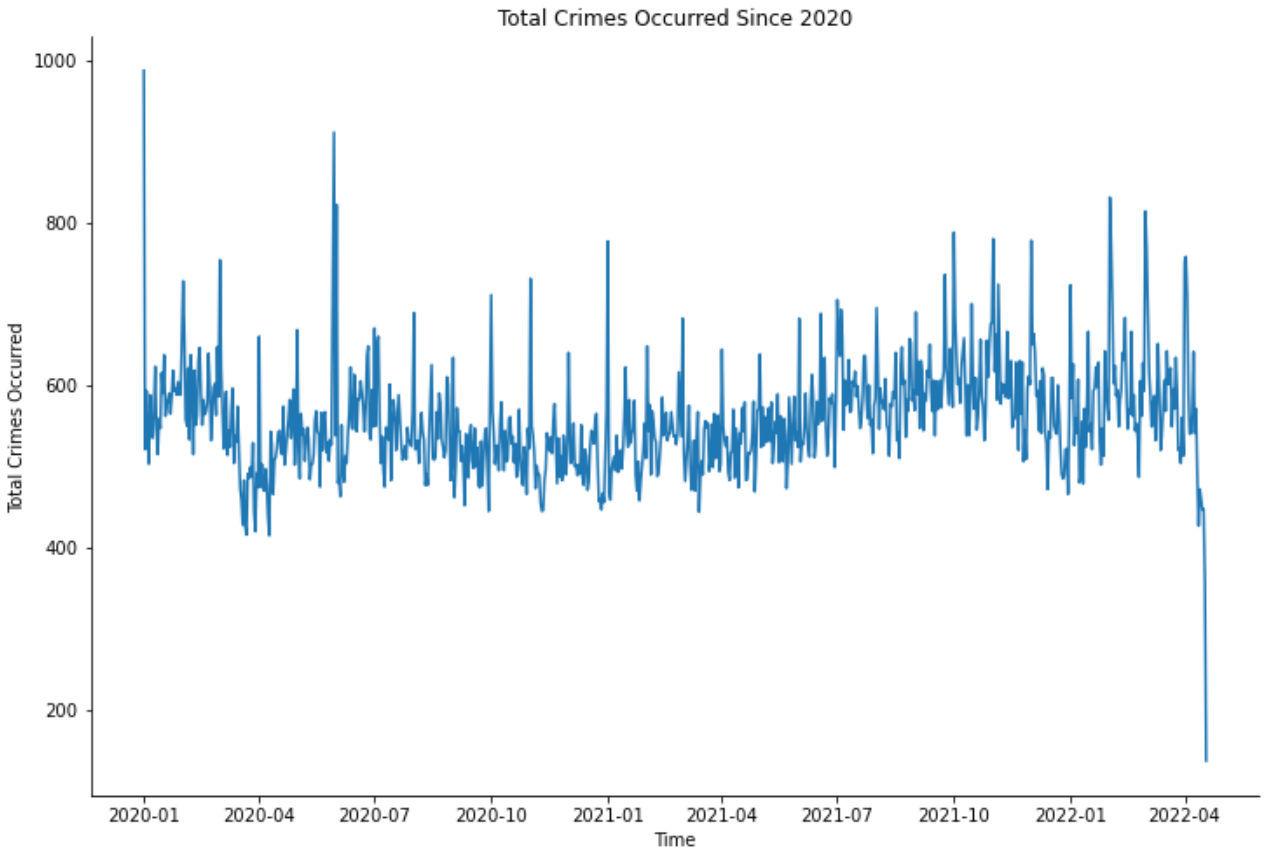
```
fig, ax = plt.subplots()
# Plot crimes reported over time
plt.plot(crime["Date Rptd"].value_counts().sort_index().index, crime["Date Rptd"
# Axes
ax.set_title("Total Crimes Reported Since 2020")
ax.set_xlabel("Time")
ax.set_ylabel("Total Crimes Reported")
sns.despine()
```



Crimes Occurred Over Time

In [20]:

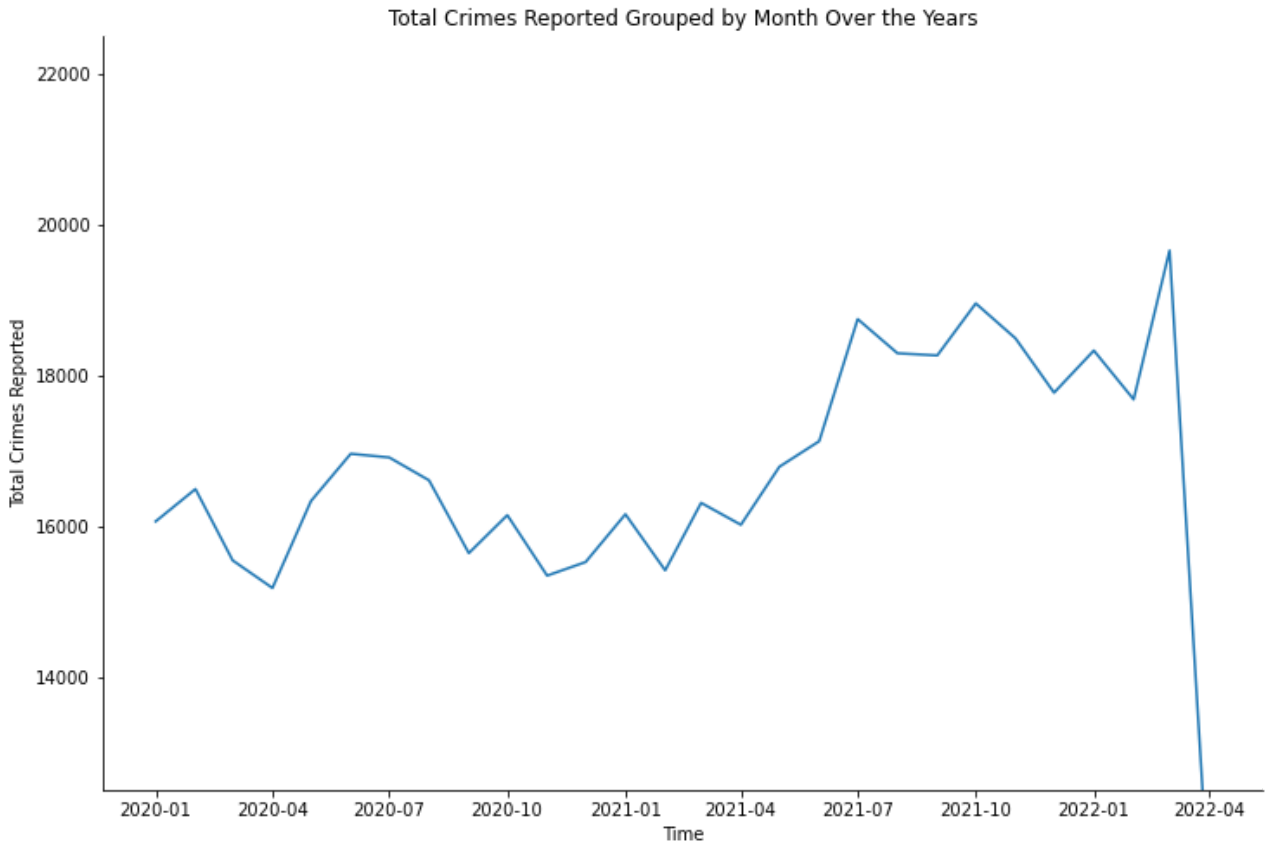
```
fig, ax = plt.subplots()
# Plot crimes occurred over time
plt.plot(crime["DATE OCC"].value_counts().sort_index().index, crime["DATE OCC"].
# Axes
ax.set_title("Total Crimes Occurred Since 2020")
ax.set_xlabel("Time")
ax.set_ylabel("Total Crimes Occurred")
sns.despine()
```



3.2.2.2. Crimes Over Time by Month and Year 3.2.2.2.1. Crimes Reported over Time by Month and Year

In [21]:
```
 # Strip the month and the year as string
month_year_rep = [str(m)+"/"+str(y) for m,y in zip(crime["Month Reported"], crim
# Make them date time objects as a list
month_year_rep_formatted = [dt.datetime.strptime(d, "%m/%Y") for d in month_year
# Turn the list of datetime month and year into a new column
crime["Month Year Rep"] = np.array(month_year_rep_formatted)
```

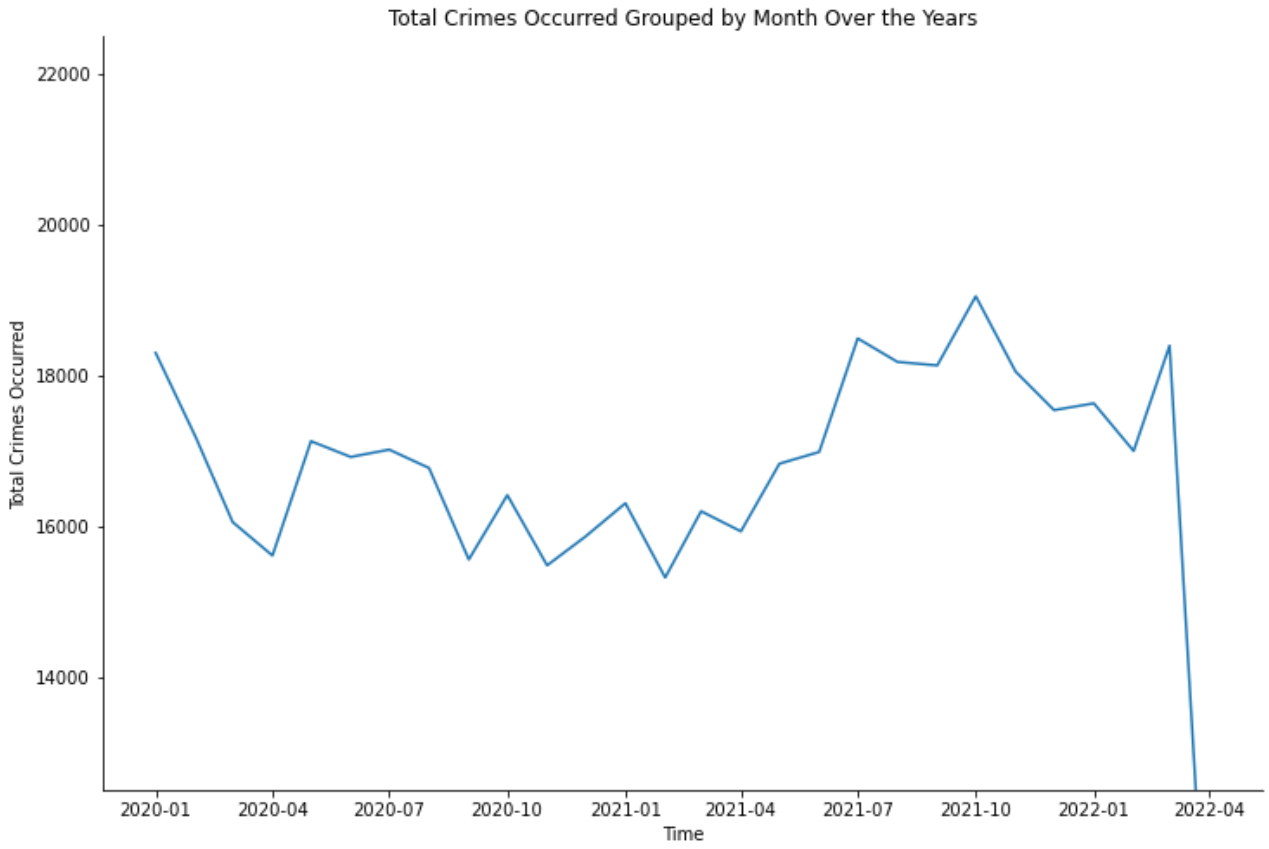
In [22]:
```
fig, ax = plt.subplots()
# Plot crimes reported over months and years
plt.plot(crime["Month Year Rep"].value_counts().sort_index().index, crime["Month
# Axes
ax.set_title("Total Crimes Reported Grouped by Month Over the Years")
ax.set_ylim(12500, 22500)
ax.set_xlabel("Time")
ax.set_ylabel("Total Crimes Reported")
sns.despine()
```



3.2.2.2.2. Crimes Occurred over Time by Month and Year

In [23]:
```
# Strip the month and the year as string
month_year_occ = [str(m)+"/"+str(y) for m,y in zip(crime["Month Occurred"], crim
# Make them date time objects as a list
month_year_occ_formatted = [dt.datetime.strptime(d, "%m/%Y") for d in month_year
# Turn the list of datetime month and year into a new column
crime["Month Year Occ"] = np.array(month_year_occ_formatted)
```

In [24]:
```
fig, ax = plt.subplots()
# Plot crimes occurred over months and years
plt.plot(crime["Month Year Occ"].value_counts().sort_index().index, crime["Month
# Axes
ax.set_title("Total Crimes Occurred Grouped by Month Over the Years")
ax.set_ylim(12500, 22500)
ax.set_xlabel("Time")
ax.set_ylabel("Total Crimes Occurred")
sns.despine()
```



3.3. Time Occurred

In [25]:
```
def makemil(time):
    ntime = ""
    if len(str(time)) == 1:
        ntime = "000" + str(time)
    if len(str(time)) == 2:
        ntime = "00" + str(time)
    if len(str(time)) == 3:
        ntime = "0" + str(time)
    if len(str(time)) == 4:
        ntime = str(time)
    return ntime

def returnhour(miltime):
    return miltime[:2]
```

In [26]:
```
# Formatting to 4 char string
crime["TIME OCC"] = crime["TIME OCC"].apply(makemil)
```

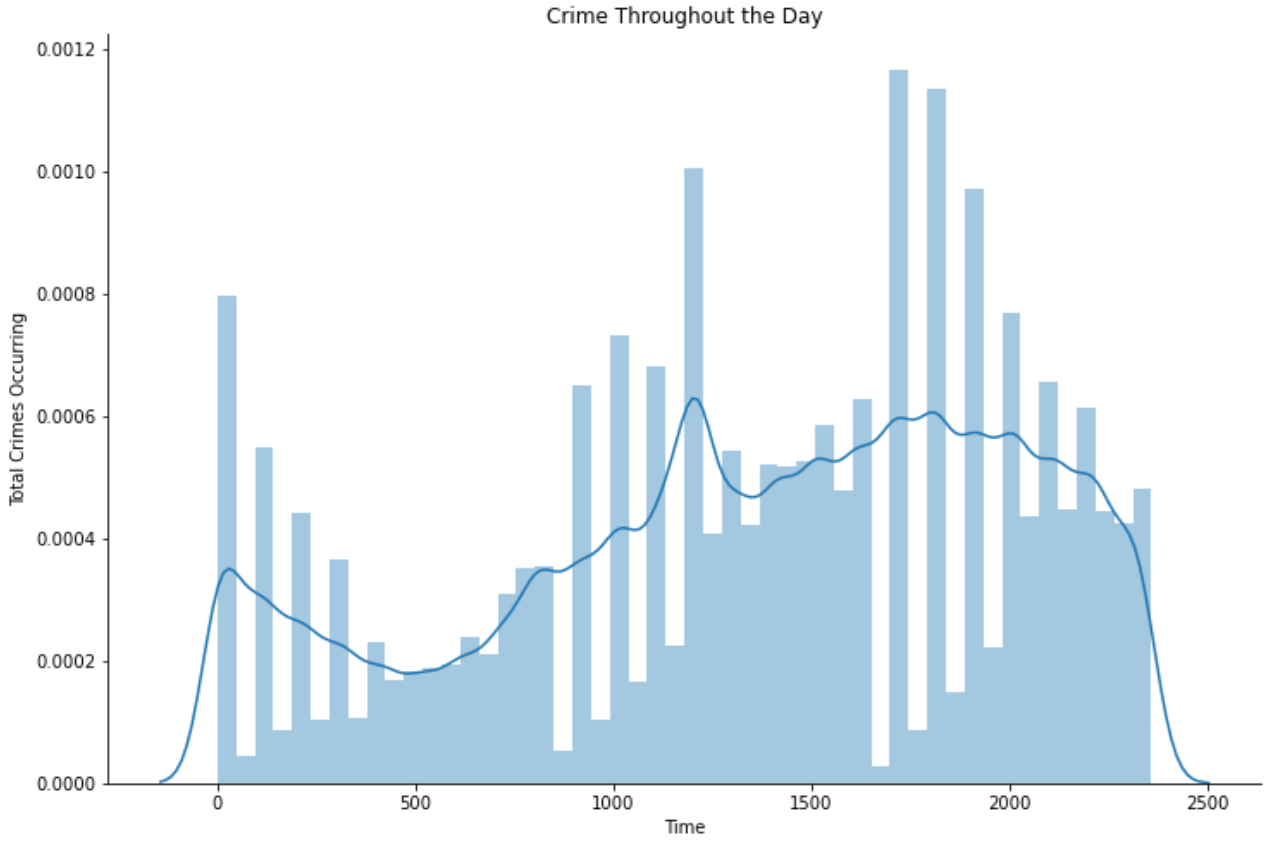
3.3.1. Crime Throughout the Day

In [27]:
```
# Formatting to int so it can be sorted
crime["TIME OCC"] = crime["TIME OCC"].apply(int)
```

In [28]:
```
fig, ax = plt.subplots()
```

```
# Plot crime throughout a single day hours
sns.distplot(crime["TIME OCC"])
# Axes
ax.set_title("Crime Throughout the Day")
ax.set_xlabel("Time")
ax.set_ylabel("Total Crimes Occurring")
sns.despine()
```

```
/Users/anthonyvega/opt/anaconda3/lib/python3.9/site-packages/seaborn/distributio
ns.py:2619: FutureWarning: `distplot` is a deprecated function and will be remov
ed in a future version. Please adapt your code to use either `displot` (a figure
-level function with similar flexibility) or `histplot` (an axes-level function
for histograms).
  warnings.warn(msg, FutureWarning)
```



3.3.1. Crime Throughout the Hour

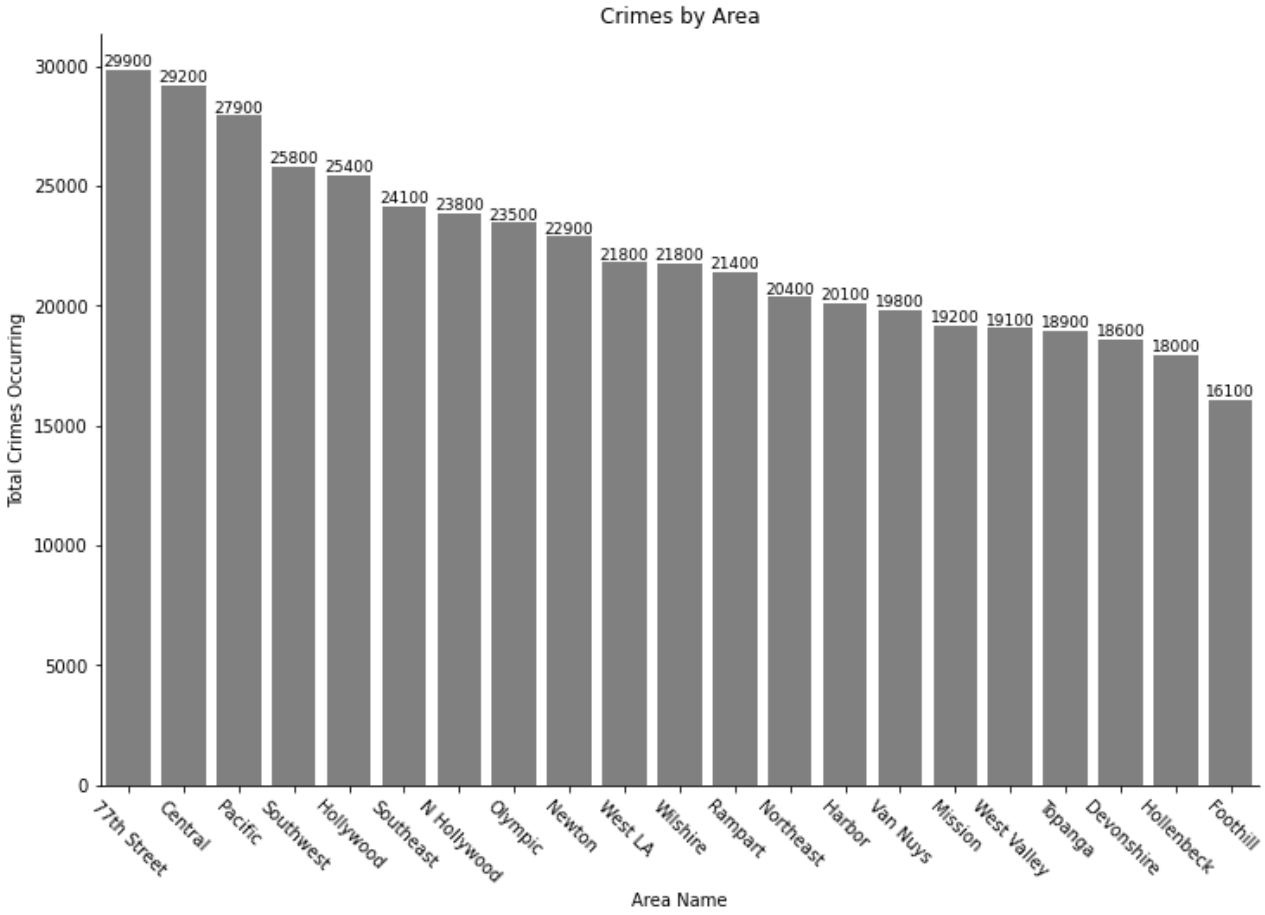
3.4 AREA NAME

In [29]:

```
fig, ax = plt.subplots()
# Plotting crimes by neighborhood area
sns.barplot(crime["AREA NAME"].value_counts().index, crime["AREA NAME"].value_co
# Axes
ax.set_title("Crimes by Area")
ax.set_xticklabels(ax.get_xticklabels(), rotation=-45)
ax.set_xlabel("Area Name")
ax.set_ylabel("Total Crimes Occurring")
# Adding Values
for p in ax.patches:
    ax.text(p.get_x() + p.get_width()/2., p.get_height(), '%d' % round(int(p.get
```

```
            fontsize=9, color='black', ha='center', va='bottom')
sns.despine()
```

```
/Users/anthonyvega/opt/anaconda3/lib/python3.9/site-packages/seaborn/_decorator
s.py:36: FutureWarning: Pass the following variables as keyword args: x, y. From
version 0.12, the only valid positional argument will be `data`, and passing oth
er arguments without an explicit keyword will result in an error or misinterpret
ation.
  warnings.warn(
```



3.5. Crime Code

In [30]:

```
# Tally total of top 20 crimes
crime["Crm Cd Desc"].value_counts().head(20)
```

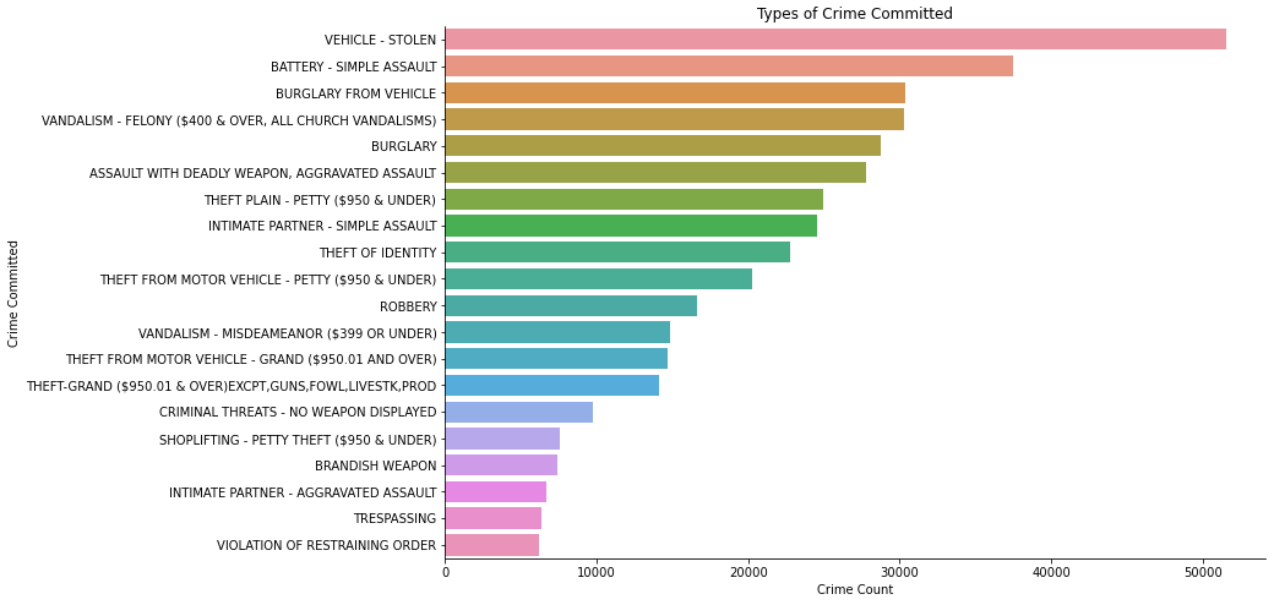
Out[30]:

```
VEHICLE - STOLEN                                          51562
BATTERY - SIMPLE ASSAULT                                  37470
BURGLARY FROM VEHICLE                                     30350
VANDALISM - FELONY ($400 & OVER, ALL CHURCH VANDALISMS)   30269
BURGLARY                                                  28779
ASSAULT WITH DEADLY WEAPON, AGGRAVATED ASSAULT            27786
THEFT PLAIN - PETTY ($950 & UNDER)                        24962
INTIMATE PARTNER - SIMPLE ASSAULT                         24537
THEFT OF IDENTITY                                         22790
THEFT FROM MOTOR VEHICLE - PETTY ($950 & UNDER)           20245
ROBBERY                                                   16648
VANDALISM - MISDEAMEANOR ($399 OR UNDER)                  14842
THEFT FROM MOTOR VEHICLE - GRAND ($950.01 AND OVER)       14650
```

```
THEFT-GRAND ($950.01 & OVER)EXCPT,GUNS,FOWL,LIVESTK,PROD    14107
CRIMINAL THREATS - NO WEAPON DISPLAYED                       9775
SHOPLIFTING - PETTY THEFT ($950 & UNDER)                     7533
BRANDISH WEAPON                                              7422
INTIMATE PARTNER - AGGRAVATED ASSAULT                        6713
TRESPASSING                                                  6391
VIOLATION OF RESTRAINING ORDER                               6171
Name: Crm Cd Desc, dtype: int64
```

In [31]:

```
# Since the number of unique crimes are more than 100, plot top 20
fig, ax = plt.subplots()
# Plotting crimes by type
sns.barplot(y=crime["Crm Cd Desc"].value_counts().index[0:20],
                  x=crime["Crm Cd Desc"].value_counts().head(20), ax=ax)
# Axes
ax.set_title("Types of Crime Committed")
ax.set_xlabel("Crime Count")
ax.set_ylabel("Crime Committed")
sns.despine()
```



3.6. MO Codes

In [32]:

```
# Modus Operandi - an individual or group's habitual way of operating, which rep
```

In [33]:

```
# Splitting the MO codes per whitespace
MO_list = []
for item in crime["Mocodes"].dropna():
    MO_list.append(str(item).split())
```

In [34]:

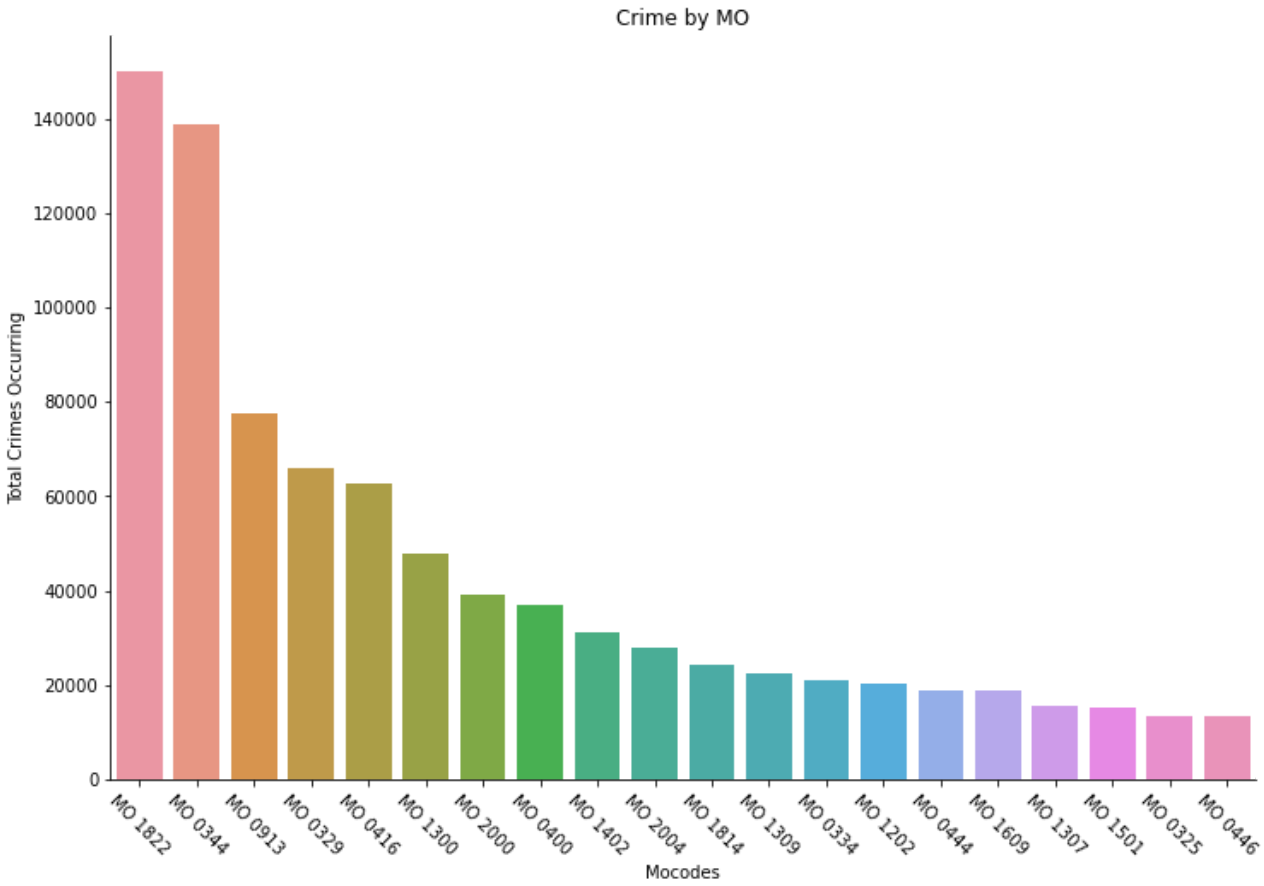
```
# Making a new DataFrame for MO Codes
tempo_MO_split = []
for i in MO_list:
    for j in i:
        tempo_MO_split.append("MO "+j)
```

```
tempo_MO_split = np.array(tempo_MO_split)

pre_MO_df = [["","Mocodes"]]
for i in range(len(tempo_MO_split)):
    pre_MO_df.append([i, tempo_MO_split[i]])

pre_MO_data = np.array(pre_MO_df)

post_MO_df = pd.DataFrame(data=pre_MO_data[1:,1:],
                  index=pre_MO_data[1:,0],
                  columns=pre_MO_data[0,1:])
```

In [35]:

```
fig, ax = plt.subplots()
# Looking into crime by MO
sns.barplot(post_MO_df["Mocodes"].value_counts().index[:20], post_MO_df["Mocodes
# Axes
ax.set_title("Crime by MO")
ax.set_xticklabels(ax.get_xticklabels(), rotation=-45)
ax.set_xlabel("Mocodes")
ax.set_ylabel("Total Crimes Occurring")
sns.despine()
```

```
/Users/anthonyvega/opt/anaconda3/lib/python3.9/site-packages/seaborn/_decorator
s.py:36: FutureWarning: Pass the following variables as keyword args: x, y. From
version 0.12, the only valid positional argument will be `data`, and passing oth
er arguments without an explicit keyword will result in an error or misinterpret
ation.
  warnings.warn(
```



-MO 0344: Removes vict property -MO 0416: Hit-hit w/ weapon -MO 0329: Vandalized -MO

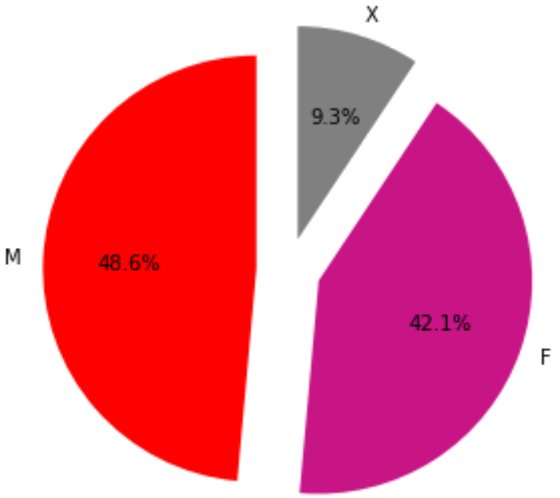
1822: Stranger -MO 2000: Domestic violence

3.7. Victim Sex

In [41]:
```
# Victim Sex Demographics
crime["Vict Sex"].value_counts()
```

Out[41]:
```
M    196751
F    170587
X     37714
H        52
Name: Vict Sex, dtype: int64
```

In [48]:
```
fig, ax = plt.subplots(figsize=(6,4))
# Plotting piechart of victim sex
ax.pie(crime["Vict Sex"].value_counts()[:3],labels=crime["Vict Sex"].value_count
       explode=(0.15,0.15,.15), autopct="%0.1f%%", colors=("red","mediumvioletre
fig.tight_layout()
```



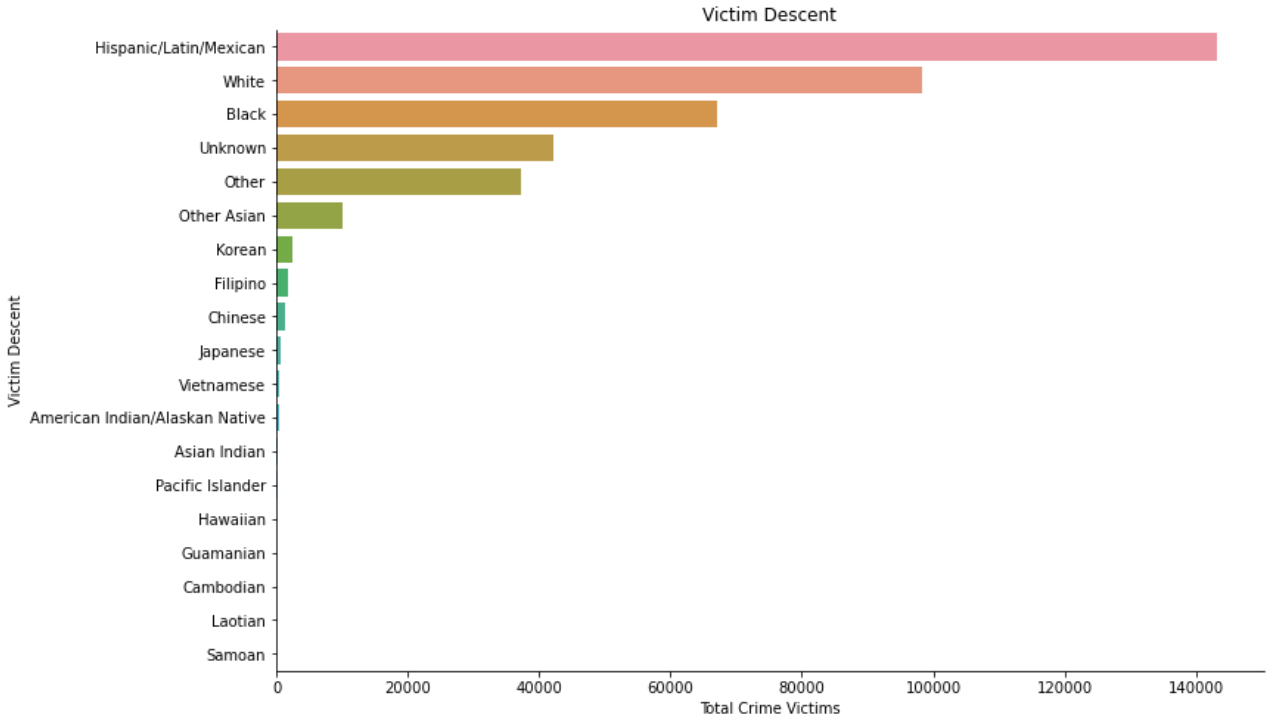
3.8. Victim Descent

In [43]:
```
# Changing the abbreviations to the whole description
Victims_bg = {
    "A": "Other Asian",
    "B": "Black",
    "C": "Chinese",
    "D": "Cambodian",
    "F": "Filipino",
    "G": "Guamanian",
    "H": "Hispanic/Latin/Mexican",
    "I": "American Indian/Alaskan Native",
    "J": "Japanese",
    "K": "Korean",
    "L": "Laotian",
    "O": "Other",
    "P": "Pacific Islander",
```

```
    "S": "Samoan",
    "U": "Hawaiian",
    "V": "Vietnamese",
    "W": "White",
    "X": "Unknown",
    "Z": "Asian Indian"
}
crime["Vict Descent"] = crime["Vict Descent"].map(Victims_bg)
```

In [45]:

```
fig, ax = plt.subplots()
# Plotting by victim gescent generally
sns.barplot(y=crime["Vict Descent"].value_counts().index, x=crime["Vict Descent"
# Axes
ax.set_title("Victim Descent")
ax.set_xlabel("Total Crime Victims")
ax.set_ylabel("Victim Descent")
sns.despine()
```



3.9. Premise Description

In [46]:

```
# Previewing the total tally
crime["Premis Desc"].value_counts()
```

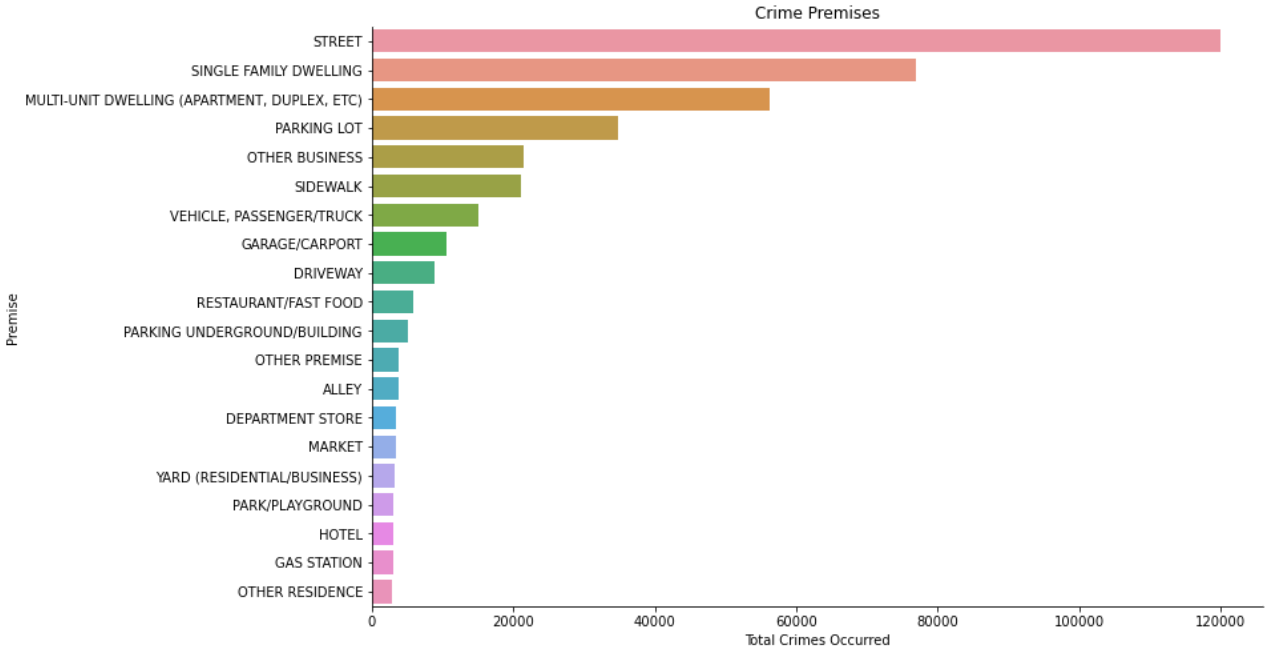
Out[46]:

```
STREET                                              119920
SINGLE FAMILY DWELLING                               76904
MULTI-UNIT DWELLING (APARTMENT, DUPLEX, ETC)         56202
PARKING LOT                                          34836
OTHER BUSINESS                                       21371
                                                       ...
MTA - GOLD LINE - LITTLE TOKYO/ARTS DISTRICT             1
TRAM/STREETCAR(BOXLIKE WAG ON RAILS)*                    1
MUSCLE BEACH                                             1
```

```
HANDBALL COURTS                                  1
MTA - SILVER LINE - MANCHESTER                   1
Name: Premis Desc, Length: 305, dtype: int64
```

In [47]:

```
# We will only be looking at the top 20 premises
fig, ax = plt.subplots()
# Plotting top 20 premises
sns.barplot(y=crime["Premis Desc"].value_counts().head(20).index,
                  x=crime["Premis Desc"].value_counts().head(20), ax=ax)
# Axes
ax.set_title("Crime Premises")
ax.set_xlabel("Total Crimes Occurred")
ax.set_ylabel("Premise")
sns.despine()
```



3.10. Weapon Description

In [49]:

```
# Number of Na values
missvals = crime["Weapon Desc"].isnull().sum()
print("There are {} missing values".format(missvals))
```

```
There are 300259 missing values
```

In [50]:

```
crime["Weapon Desc"].value_counts().head(10)
```

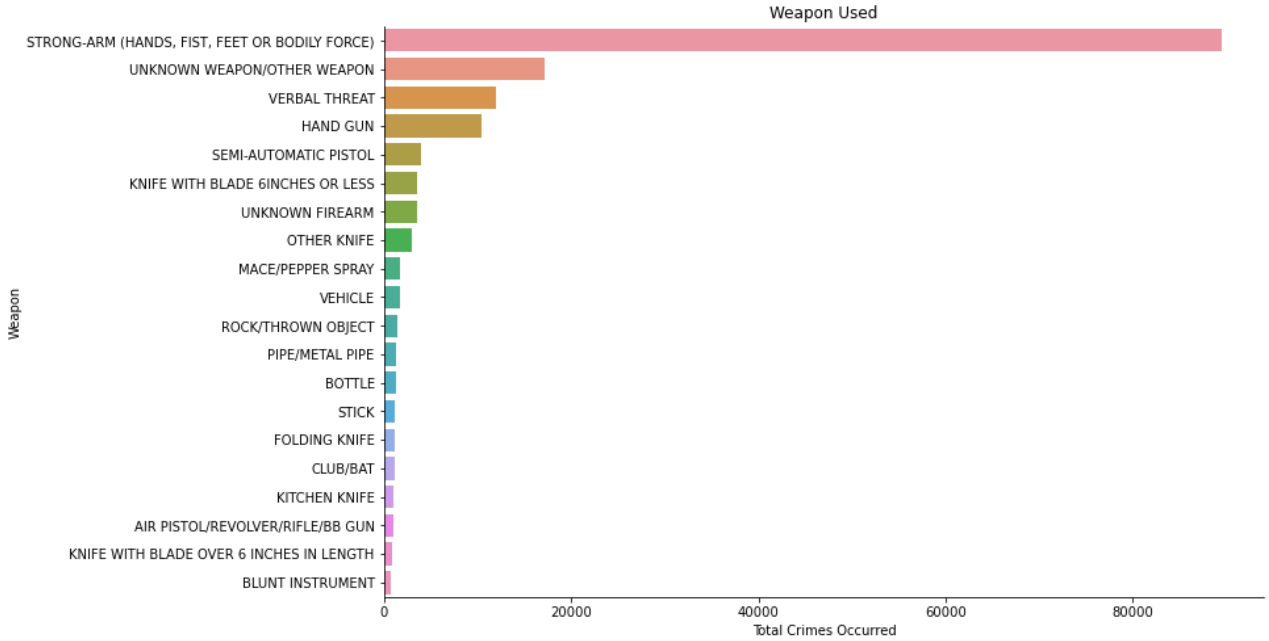
Out[50]:

```
STRONG-ARM (HANDS, FIST, FEET OR BODILY FORCE)    89548
UNKNOWN WEAPON/OTHER WEAPON                       17235
VERBAL THREAT                                     12035
HAND GUN                                          10462
SEMI-AUTOMATIC PISTOL                              3909
KNIFE WITH BLADE 6INCHES OR LESS                   3553
UNKNOWN FIREARM                                    3539
OTHER KNIFE                                        2944
MACE/PEPPER SPRAY                                  1767
```

```
VEHICLE                                                    1721
Name: Weapon Desc, dtype: int64
```

In [51]:

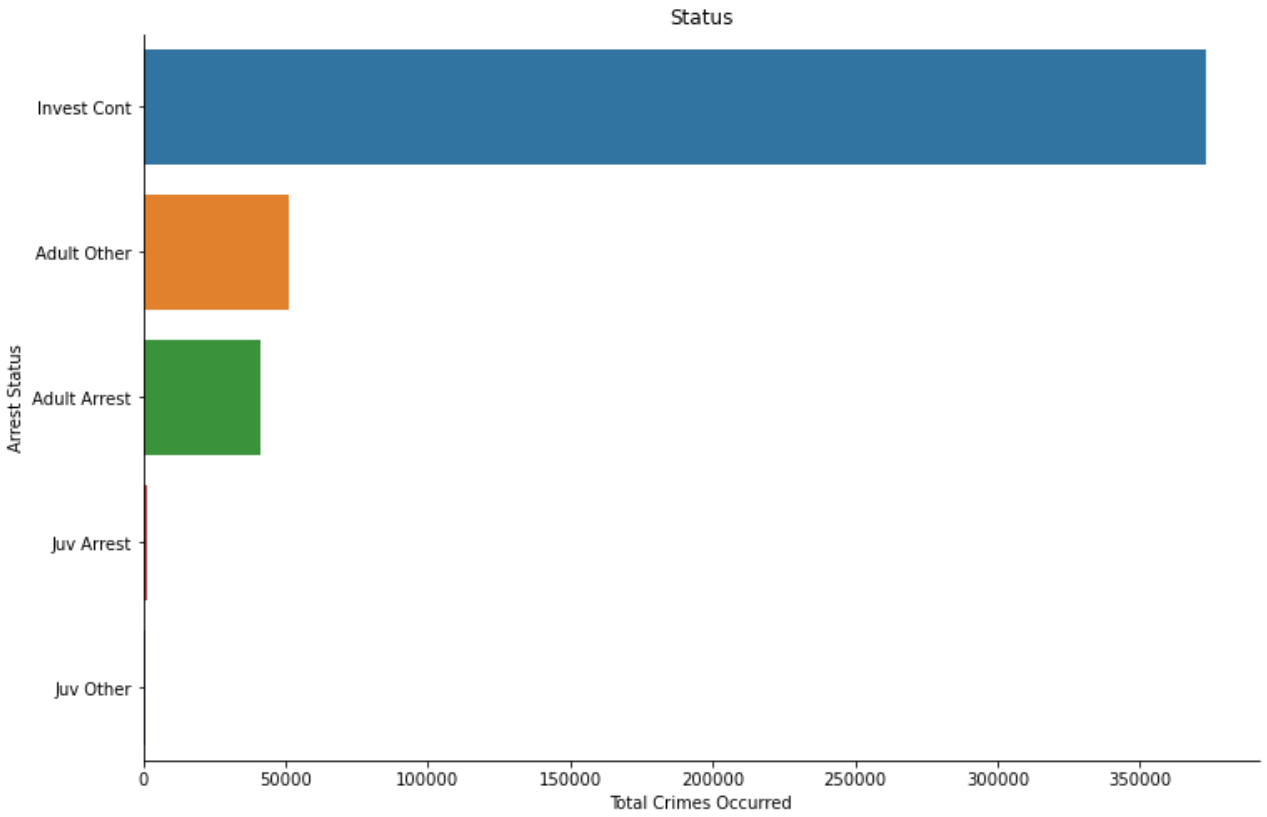
```
fig, ax = plt.subplots()
# Plotting weapons used
sns.barplot(y=crime["Weapon Desc"].value_counts().head(20).index,
                  x=crime["Weapon Desc"].value_counts().head(20), ax=ax)
# Axes
ax.set_title("Weapon Used")
ax.set_xlabel("Total Crimes Occurred")
ax.set_ylabel("Weapon")
sns.despine()
```



3.11. Status Description

In [37]:

```
fig, ax = plt.subplots()
# Plotting the arrest status
sns.barplot(y=crime["Status Desc"].value_counts().index,
                  x=crime["Status Desc"].value_counts(), ax=ax)
# Axes
ax.set_title("Status")
ax.set_xlabel("Total Crimes Occurred")
ax.set_ylabel("Arrest Status")
sns.despine()
```



In [ ]:

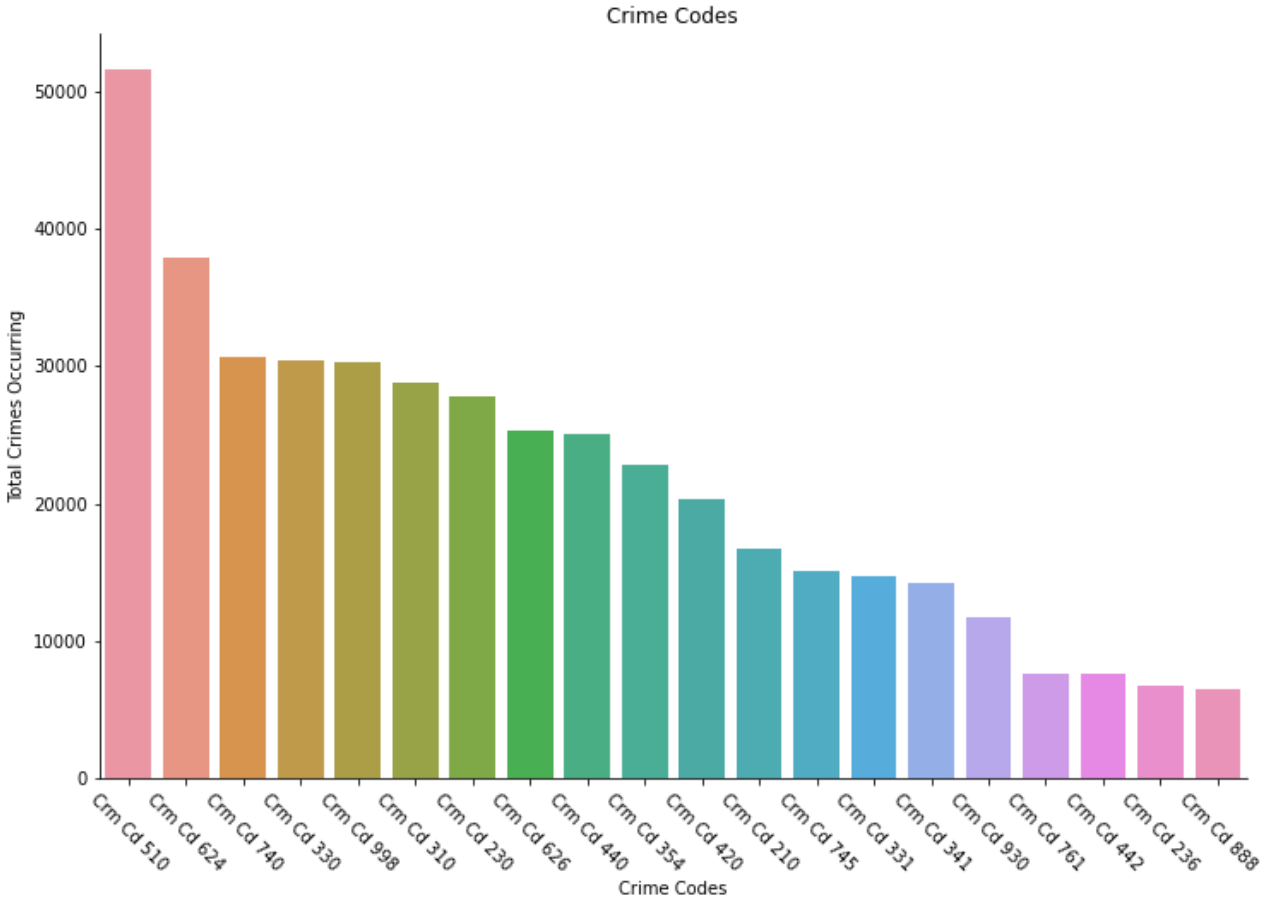
In [ ]:

3.13. Crime Code

In [38]:
```
# Making a new dataframe
CC_list = []
for i in range(1,5):
    for item in crime["Crm Cd "+str(i)].dropna():
        CC_list.append("Crm Cd " +str(int(item)))

tempo_CC = np.array(CC_list)

CC_df = pd.DataFrame(tempo_CC)
CC_df = CC_df.rename(columns = {0 : "Crime Codes"})
```

In [39]:
```
fig, ax = plt.subplots()
# Plotting crime codes
ax = sns.barplot(CC_df["Crime Codes"].value_counts().head(20).index, CC_df["Crim
# Axes
ax.set_title("Crime Codes")
ax.set_xticklabels(ax.get_xticklabels(), rotation=-45)
ax.set_xlabel("Crime Codes")
ax.set_ylabel("Total Crimes Occurring")
sns.despine()
```

```
/Users/anthonyvega/opt/anaconda3/lib/python3.9/site-packages/seaborn/_decorator
s.py:36: FutureWarning: Pass the following variables as keyword args: x, y. From
version 0.12, the only valid positional argument will be `data`, and passing oth
er arguments without an explicit keyword will result in an error or misinterpret
ation.
  warnings.warn(
```



4. Data Exploration

In [59]:
```
# Removing Entries for X and H and - (by elimination)
crime["Victim Gender"] = crime["Vict Sex"][crime["Vict Sex"] != "X"]
crime["Victim Gender"] = crime["Victim Gender"][crime["Victim Gender"] != "H"]
crime["Victim Gender"] = crime["Victim Gender"][crime["Victim Gender"] != "-"]
```

In [61]:
```
# Combining two columns into a dataframe
cc_vg = crime[["Crm Cd Desc", "Victim Gender"]]
# Dropping null values
cc_vg = cc_vg[pd.notnull(cc_vg["Victim Gender"])]
```

In [62]:
```
# Saving top 10 crimes
crimetop10 = cc_vg["Crm Cd Desc"].value_counts().head(10).index
# Choosing data that is included in the top 10 crimes (by selection)
crimecc = cc_vg.loc[cc_vg["Crm Cd Desc"].isin(crimetop10)]
```

In [63]:
```
# Group by Crime Code Description and Victim Gender
```

```
cc_gender = crimecc.groupby(["Crm Cd Desc", "Victim Gender"]).size().reset_index
cc_gender
```

Out[63]:

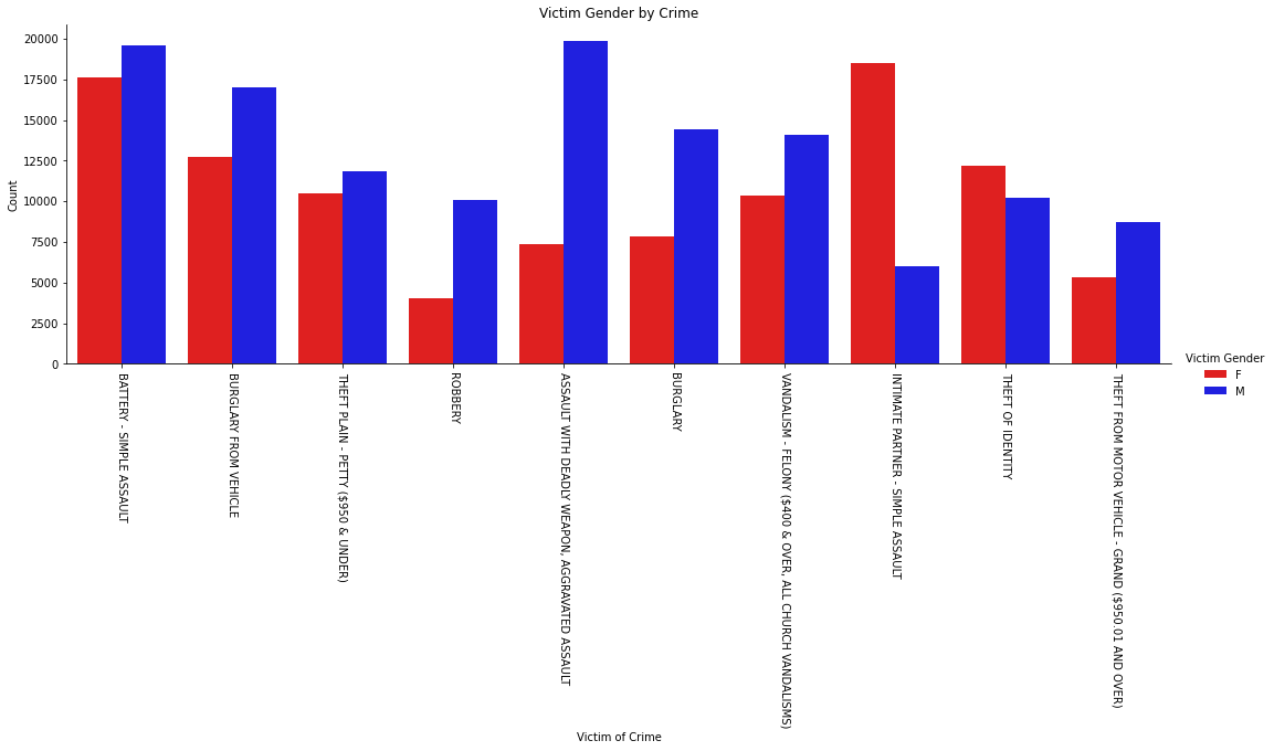
	Crm Cd Desc	Victim Gender	Count
0	ASSAULT WITH DEADLY WEAPON, AGGRAVATED ASSAULT	F	7366
1	ASSAULT WITH DEADLY WEAPON, AGGRAVATED ASSAULT	M	19894
2	BATTERY - SIMPLE ASSAULT	F	17633
3	BATTERY - SIMPLE ASSAULT	M	19612
4	BURGLARY	F	7863
5	BURGLARY	M	14429
6	BURGLARY FROM VEHICLE	F	12720
7	BURGLARY FROM VEHICLE	M	16998
8	INTIMATE PARTNER - SIMPLE ASSAULT	F	18496
9	INTIMATE PARTNER - SIMPLE ASSAULT	M	5989
10	ROBBERY	F	4054
11	ROBBERY	M	10053
12	THEFT FROM MOTOR VEHICLE - GRAND ($950.01 AND ...	F	5336
13	THEFT FROM MOTOR VEHICLE - GRAND ($950.01 AND ...	M	8715
14	THEFT OF IDENTITY	F	12195
15	THEFT OF IDENTITY	M	10254
16	THEFT PLAIN - PETTY ($950 & UNDER)	F	10487
17	THEFT PLAIN - PETTY ($950 & UNDER)	M	11869
18	VANDALISM - FELONY ($400 & OVER, ALL CHURCH VA...	F	10345
19	VANDALISM - FELONY ($400 & OVER, ALL CHURCH VA...	M	14116

In [64]:

```
# Factorplot Crime and Gender based on count
ax = sns.factorplot(x="Crm Cd Desc", hue="Victim Gender", kind="count", data=cri
                    palette=["red", "blue"])
# Axes
plt.title("Victim Gender by Crime")
ax.set_xticklabels(rotation=-90)
ax.set_xlabels("Victim of Crime")
ax.set_ylabels("Count")
sns.despine()
```

```
/Users/anthonyvega/opt/anaconda3/lib/python3.9/site-packages/seaborn/categorica
l.py:3717: UserWarning: The `factorplot` function has been renamed to `catplot`.
The original name will be removed in a future release. Please update your code.
Note that the default `kind` in `factorplot` (`'point'`) has changed `'strip'` i
n `catplot`.
  warnings.warn(msg)
/Users/anthonyvega/opt/anaconda3/lib/python3.9/site-packages/seaborn/categorica
l.py:3723: UserWarning: The `size` parameter has been renamed to `height`; pleas
```

```
e update your code.
  warnings.warn(msg, UserWarning)
```



Some things to note are:

The two crimes that women victims are more often are Intimate partner sexual assault & identity theft.