fda-group-project-1

November 4, 2023

Importing Necessary Libraries

[107]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
```

Data Acquisition

[114]:
```
from google.colab import files
uploaded = files.upload()
```

```
<IPython.core.display.HTML object>
```

```
Saving Crime_Data_from_2020_to_Present.csv to Crime_Data_from_2020_to_Present
(3).csv
```

[115]:
```
import io
df = pd.read_csv(io.BytesIO(uploaded['Crime_Data_from_2020_to_Present (3).
 ↪csv']))
```

Data Inspection

[ ]:
```
#Visually analyzing the first few rows
df.head()
```

[ ]:

	DR_NO	Date Rptd	DATE OCC	TIME OCC	AREA	\
0	10304468	01/08/2020 12:00:00 AM	01/08/2020 12:00:00 AM	2230	3	
1	190101086	01/02/2020 12:00:00 AM	01/01/2020 12:00:00 AM	330	1	
2	200110444	04/14/2020 12:00:00 AM	02/13/2020 12:00:00 AM	1200	1	
3	191501505	01/01/2020 12:00:00 AM	01/01/2020 12:00:00 AM	1730	15	
4	191921269	01/01/2020 12:00:00 AM	01/01/2020 12:00:00 AM	415	19	

	AREA NAME	Rpt Dist No	Part 1-2	Crm Cd	\
0	Southwest	377	2	624	
1	Central	163	2	624	
2	Central	155	2	845	
3	N Hollywood	1543	2	745	
4	Mission	1998	2	740	

```
                                      Crm Cd Desc  … Status  \
0                          BATTERY - SIMPLE ASSAULT  …     AO
1                          BATTERY - SIMPLE ASSAULT  …     IC
2          SEX OFFENDER REGISTRANT OUT OF COMPLIANCE  …     AA
3           VANDALISM - MISDEAMEANOR ($399 OR UNDER)  …     IC
4  VANDALISM - FELONY ($400 & OVER, ALL CHURCH VA…  …     IC

    Status Desc Crm Cd 1 Crm Cd 2  Crm Cd 3 Crm Cd 4  \
0   Adult Other    624.0      NaN       NaN      NaN
1   Invest Cont    624.0      NaN       NaN      NaN
2  Adult Arrest    845.0      NaN       NaN      NaN
3   Invest Cont    745.0    998.0       NaN      NaN
4   Invest Cont    740.0      NaN       NaN      NaN

                                LOCATION Cross Street      LAT       LON
0   1100 W  39TH                      PL          NaN  34.0141 -118.2978
1    700 S  HILL                      ST          NaN  34.0459 -118.2545
2    200 E  6TH                       ST          NaN  34.0448 -118.2474
3   5400    CORTEEN                   PL          NaN  34.1685 -118.4019
4  14400    TITUS                     ST          NaN  34.2198 -118.4468

[5 rows x 28 columns]
```

[ ]:
```
#Getting datatypes
df.dtypes
```

[ ]:
```
DR_NO              int64
Date Rptd         object
DATE OCC          object
TIME OCC           int64
AREA               int64
AREA NAME         object
Rpt Dist No        int64
Part 1-2           int64
Crm Cd             int64
Crm Cd Desc       object
Mocodes           object
Vict Age           int64
Vict Sex          object
Vict Descent      object
Premis Cd        float64
Premis Desc       object
Weapon Used Cd   float64
Weapon Desc       object
Status            object
Status Desc       object
```

```
Crm Cd 1          float64
Crm Cd 2          float64
Crm Cd 3          float64
Crm Cd 4          float64
LOCATION           object
Cross Street       object
LAT               float64
LON               float64
dtype: object
```

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

[ ]:
```
              DR_NO       TIME OCC           AREA    Rpt Dist No  \
count  7.051390e+05  705139.000000  705139.000000  705139.000000
mean   2.159633e+08    1320.408799      10.652555    1111.602846
std    1.076728e+07     653.593830       6.111664     611.074999
min    1.030447e+07       1.000000       1.000000     101.000000
25%    2.101194e+08     900.000000       6.000000     621.000000
50%    2.201120e+08    1400.000000      11.000000    1132.000000
75%    2.219102e+08    1840.000000      16.000000    1612.000000
max    2.399165e+08    2359.000000      21.000000    2199.000000

            Part 1-2         Crm Cd       Vict Age      Premis Cd  \
count  705139.000000  705139.000000  705139.000000  705139.000000
mean        1.473945     501.203753      34.332811     332.914092
std         0.499321     222.351337      19.769297     215.620712
min         1.000000     110.000000      -3.000000     101.000000
25%         1.000000     330.000000      23.000000     108.000000
50%         1.000000     440.000000      34.000000     401.000000
75%         2.000000     661.000000      48.000000     502.000000
max         2.000000     956.000000      99.000000     976.000000

       Weapon Used Cd       Crm Cd 1      Crm Cd 2     Crm Cd 3    Crm Cd 4  \
count   282657.000000  705130.000000  59662.000000  1999.000000   60.000000
mean       362.910881     500.902202    957.685880   983.546773  990.766667
std        123.736374     222.124960    111.062021    52.929465   27.912919
min        101.000000     110.000000    210.000000   310.000000  821.000000
25%        310.000000     330.000000    998.000000   998.000000  998.000000
50%        400.000000     440.000000    998.000000   998.000000  998.000000
75%        400.000000     649.000000    998.000000   998.000000  998.000000
max        516.000000     956.000000    999.000000   999.000000  999.000000

                LAT            LON
count  705139.000000  705139.000000
mean       33.969270    -117.990158
std         1.892708       6.563936
min         0.000000    -118.667600
```

```
25%        34.015300    -118.431200
50%        34.059200    -118.323600
75%        34.163000    -118.274000
max        34.334300       0.000000
```

[ ]:
```
#Looking at first row
df.loc[0]
```

[ ]:
```
DR_NO                                                   10304468
Date Rptd                                  01/08/2020 12:00:00 AM
DATE OCC                                   01/08/2020 12:00:00 AM
TIME OCC                                                     2230
AREA                                                            3
AREA NAME                                               Southwest
Rpt Dist No                                                   377
Part 1-2                                                        2
Crm Cd                                                        624
Crm Cd Desc                                BATTERY - SIMPLE ASSAULT
Mocodes                                                0444 0913
Vict Age                                                       36
Vict Sex                                                        F
Vict Descent                                                    B
Premis Cd                                                   501.0
Premis Desc                                SINGLE FAMILY DWELLING
Weapon Used Cd                                              400.0
Weapon Desc       STRONG-ARM (HANDS, FIST, FEET OR BODILY FORCE)
Status                                                         AO
Status Desc                                            Adult Other
Crm Cd 1                                                    624.0
Crm Cd 2                                                      NaN
Crm Cd 3                                                      NaN
Crm Cd 4                                                      NaN
LOCATION                  1100 W  39TH                         PL
Cross Street                                                  NaN
LAT                                                       34.0141
LON                                                      -118.2978
Name: 0, dtype: object
```

Data Cleaning

[ ]:
```
#Handelling missing data
#Getting total number of missing values in each column
df.isnull().sum()
```

[ ]:
```
DR_NO                 0
Date Rptd             0
DATE OCC              0
```

```
TIME OCC              0
AREA                  0
AREA NAME             0
Rpt Dist No           0
Part 1-2              0
Crm Cd                0
Crm Cd Desc           0
Mocodes          112024
Vict Age              0
Vict Sex         106524
Vict Descent     106532
Premis Cd             9
Premis Desc         479
Weapon Used Cd   528880
Weapon Desc      528880
Status                0
Status Desc           0
Crm Cd 1             10
Crm Cd 2         751848
Crm Cd 3         809663
Crm Cd 4         811603
LOCATION              0
Cross Street     681791
LAT                   0
LON                   0
dtype: int64
```

[ ]:
```
#Getting percentage of missing values from each column
df.isnull().mean()
```

[ ]:
```
DR_NO            0.000000
Date Rptd        0.000000
DATE OCC         0.000000
TIME OCC         0.000000
AREA             0.000000
AREA NAME        0.000000
Rpt Dist No      0.000000
Part 1-2         0.000000
Crm Cd           0.000000
Crm Cd Desc      0.000000
Mocodes          0.138018
Vict Age         0.000000
Vict Sex         0.131242
Vict Descent     0.131252
Premis Cd        0.000011
Premis Desc      0.000590
Weapon Used Cd   0.651600
```

```
Weapon Desc       0.651600
Status            0.000000
Status Desc       0.000000
Crm Cd 1          0.000012
Crm Cd 2          0.926306
Crm Cd 3          0.997536
Crm Cd 4          0.999926
LOCATION          0.000000
Cross Street      0.839993
LAT               0.000000
LON               0.000000
dtype: float64
```

[ ]:
```
df.shape
```

[ ]:
```
(811663, 28)
```

[116]:
```
#Drooping columns where victim sex is unknown due to less percentage of missing␣
 ↪values
#also because victim sex is important for data analysis
df.dropna(subset=['Vict Sex'],inplace=True)
```

[6]:
```
df.shape
```

[6]:
```
(705139, 28)
```

[ ]:
```
#Rechecking the percent wise distribution of null values
df.isnull().mean()
```

[ ]:
```
DR_NO             0.000000
Date Rptd         0.000000
DATE OCC          0.000000
TIME OCC          0.000000
AREA              0.000000
AREA NAME         0.000000
Rpt Dist No       0.000000
Part 1-2          0.000000
Crm Cd            0.000000
Crm Cd Desc       0.000000
Mocodes           0.008279
Vict Age          0.000000
Vict Sex          0.000000
Vict Descent      0.000017
Premis Cd         0.000000
Premis Desc       0.000665
Weapon Used Cd    0.599147
Weapon Desc       0.599147
```

```
Status            0.000000
Status Desc       0.000000
Crm Cd 1          0.000013
Crm Cd 2          0.915390
Crm Cd 3          0.997165
Crm Cd 4          0.999915
LOCATION          0.000000
Cross Street      0.844296
LAT               0.000000
LON               0.000000
dtype: float64
```

[117]:
```
#Dropping values where vict descent is null, since the cases where this is true␣
 ↪are low
#and its an important metric for analysis
df.dropna(subset=['Vict Descent'],inplace=True)
```

[ ]:
```
df.isnull().mean()
```

[ ]:
```
DR_NO             0.000000
Date Rptd         0.000000
DATE OCC          0.000000
TIME OCC          0.000000
AREA              0.000000
AREA NAME         0.000000
Rpt Dist No       0.000000
Part 1-2          0.000000
Crm Cd            0.000000
Crm Cd Desc       0.000000
Mocodes           0.008272
Vict Age          0.000000
Vict Sex          0.000000
Vict Descent      0.000000
Premis Cd         0.000000
Premis Desc       0.000665
Weapon Used Cd    0.599147
Weapon Desc       0.599147
Status            0.000000
Status Desc       0.000000
Crm Cd 1          0.000013
Crm Cd 2          0.915390
Crm Cd 3          0.997165
Crm Cd 4          0.999915
LOCATION          0.000000
Cross Street      0.844296
LAT               0.000000
LON               0.000000
```

```
dtype: float64
```

[118]:
```
#Dropping values where Mocodes is null, since the cases where this is true are low
#and its an important metric for analysis
df.dropna(subset=['Mocodes'],inplace=True)
```

[9]:
```
df.shape
```

[9]:
```
(699294, 28)
```

[119]:
```
#keeping only original values in dr_no
df.drop_duplicates(subset=['DR_NO'],inplace=True)
```

[120]:
```
#converting reported date to datetime
df['Date Rptd']=pd.to_datetime(df['Date Rptd'])
```

[121]:
```
df['DATE OCC']=pd.to_datetime(df['DATE OCC'])
```

[122]:
```
#extracting years and months from our dates
df['Rept Year']=df['Date Rptd'].dt.year
df['Rept Month']=df['Date Rptd'].dt.month
df['Occurance Year']=df['DATE OCC'].dt.year
df['Occurance Month']=df['DATE OCC'].dt.month
```

[ ]:
```
df.head()
```

[ ]:

	DR_NO	Date Rptd	DATE OCC	TIME OCC	AREA	AREA NAME	Rpt Dist No	\
0	10304468	2020-01-08	2020-01-08	2230	3	Southwest	377	
1	190101086	2020-01-02	2020-01-01	330	1	Central	163	
2	200110444	2020-04-14	2020-02-13	1200	1	Central	155	
3	191501505	2020-01-01	2020-01-01	1730	15	N Hollywood	1543	
4	191921269	2020-01-01	2020-01-01	415	19	Mission	1998	

	Part 1-2	Crm Cd	Crm Cd Desc	…	\
0	2	624	BATTERY - SIMPLE ASSAULT	…	
1	2	624	BATTERY - SIMPLE ASSAULT	…	
2	2	845	SEX OFFENDER REGISTRANT OUT OF COMPLIANCE	…	
3	2	745	VANDALISM - MISDEAMEANOR ($399 OR UNDER)	…	
4	2	740	VANDALISM - FELONY ($400 & OVER, ALL CHURCH VA…	…	

	Crm Cd 3	Crm Cd 4			LOCATION	Cross Street	\
0	NaN	NaN	1100 W	39TH	PL	NaN	
1	NaN	NaN	700 S	HILL	ST	NaN	
2	NaN	NaN	200 E	6TH	ST	NaN	
3	NaN	NaN	5400	CORTEEN	PL	NaN	
4	NaN	NaN	14400	TITUS	ST	NaN	

```
       LAT       LON  Rept Year Rept Month Occurance Year Occurance Month
0  34.0141 -118.2978       2020          1            2020                1
1  34.0459 -118.2545       2020          1            2020                1
2  34.0448 -118.2474       2020          4            2020                2
3  34.1685 -118.4019       2020          1            2020                1
4  34.2198 -118.4468       2020          1            2020                1

[5 rows x 32 columns]
```

[123]:
```
#Writing a function that standardizes time format
temp=[]
def standardize_time(time):
  new_time=str(time)
  if len(new_time)==1:
    new_time=new_time+':'+'00'
    #temp.append('l')
  elif len(new_time)==2:
    #temp.append('m')
    if int(new_time)<=24:
      new_time=new_time+':'+'00'
      #temp.append('n')
    else:
      new_time='0'+new_time[0]+':'+'0'+new_time[1]
      #temp.append('o')
  elif len(new_time)==3:
    new_time='0'+new_time[0]+':'+new_time[1]+new_time[2]
    #temp.append('p')
  else:
    new_time=new_time[0]+new_time[1]+':'+new_time[2]+new_time[3]
    #temp.append('q')
  return new_time
```

[124]:
```
df['TIME OCC'] = df['TIME OCC'].apply(standardize_time)
```

[ ]:
```
df.head()
```

[ ]:
```
       DR_NO  Date Rptd   DATE OCC TIME OCC  AREA    AREA NAME  Rpt Dist No  \
0   10304468 2020-01-08 2020-01-08    22:30     3    Southwest          377
1  190101086 2020-01-02 2020-01-01    03:30     1      Central          163
2  200110444 2020-04-14 2020-02-13    12:00     1      Central          155
3  191501505 2020-01-01 2020-01-01    17:30    15  N Hollywood         1543
4  191921269 2020-01-01 2020-01-01    04:15    19      Mission         1998

   Part 1-2  Crm Cd                                   Crm Cd Desc  …  \
0         2     624                     BATTERY - SIMPLE ASSAULT  …
1         2     624                     BATTERY - SIMPLE ASSAULT  …
```

```
2         2     845          SEX OFFENDER REGISTRANT OUT OF COMPLIANCE  …
3         2     745           VANDALISM - MISDEAMEANOR ($399 OR UNDER)  …
4         2     740  VANDALISM - FELONY ($400 & OVER, ALL CHURCH VA…  …

  Crm Cd 3  Crm Cd 4                                LOCATION Cross Street  \
0      NaN       NaN   1100 W  39TH                      PL           NaN
1      NaN       NaN    700 S  HILL                      ST           NaN
2      NaN       NaN    200 E  6TH                       ST           NaN
3      NaN       NaN   5400    CORTEEN                   PL           NaN
4      NaN       NaN  14400    TITUS                     ST           NaN

       LAT       LON  Rept Year Rept Month Occurance Year Occurance Month
0  34.0141 -118.2978       2020          1            2020                1
1  34.0459 -118.2545       2020          1            2020                1
2  34.0448 -118.2474       2020          4            2020                2
3  34.1685 -118.4019       2020          1            2020                1
4  34.2198 -118.4468       2020          1            2020                1

[5 rows x 32 columns]
```

[125]:
```
#defining a function that capitalizes and removes trailing white
#spaces from text based columns
temp=[]
def capitalize_and_remove_trailing_spaces(input_string):
    # Capitalize the string
    capitalized_string = str(input_string).upper()
    #print(capitalized_string)

    # Remove trailing white spaces
    stripped_string = capitalized_string.rstrip()

    return(stripped_string)
    #print(stripped_string)
```

[126]:
```
df['AREA NAME']=df['AREA NAME'].apply(capitalize_and_remove_trailing_spaces)
```

[127]:
```
df['Crm Cd Desc']=df['Crm Cd Desc'].apply(capitalize_and_remove_trailing_spaces)
```

[128]:
```
df['Premis Desc']=df['Premis Desc'].apply(capitalize_and_remove_trailing_spaces)
```

[20]:
```
df[['Crm Cd Desc','Premis Desc','AREA NAME','Weapon Desc']].head()
```

[20]:
```
                                    Crm Cd Desc  \
0                        BATTERY - SIMPLE ASSAULT
1                        BATTERY - SIMPLE ASSAULT
2          SEX OFFENDER REGISTRANT OUT OF COMPLIANCE
3           VANDALISM - MISDEAMEANOR ($399 OR UNDER)
```

```
4  VANDALISM - FELONY ($400 & OVER, ALL CHURCH VA…

                                    Premis Desc    AREA NAME  \
0                         SINGLE FAMILY DWELLING    SOUTHWEST
1                                       SIDEWALK      CENTRAL
2                                 POLICE FACILITY      CENTRAL
3  MULTI-UNIT DWELLING (APARTMENT, DUPLEX, ETC)  N HOLLYWOOD
4                             BEAUTY SUPPLY STORE      MISSION

                                       Weapon Desc
0  STRONG-ARM (HANDS, FIST, FEET OR BODILY FORCE)
1                      UNKNOWN WEAPON/OTHER WEAPON
2                                              NaN
3                                              NaN
4                                              NaN
```

[129]:
```
df['Weapon Desc']=df['Weapon Desc'].apply(capitalize_and_remove_trailing_spaces)
```

[130]:
```
#Defining a function that gives us the total number of mocodes
def number_of_monocodes(input_string):
  word_list=input_string.split()
  return(len(word_list))
```

[131]:
```
#counting the number of mocodes
df['Nos_of_mocodes']=df['Mocodes'].apply(number_of_monocodes)
```

[132]:
```
def string_to_comma_separated(input_string):
    # Split the input string into a list based on whitespace
    word_list = input_string.split()

    # Concatenate the list elements into a new string with a comma delimiter
    result_string = ', '.join(word_list)

    return result_string
```

[133]:
```
df['Mocodes']=df['Mocodes'].apply(string_to_comma_separated)
```

[134]:
```
#Dropping cases where victim age <= 0 as part of outlier detection and action
df = df.drop(df[df['Vict Age'] <= 0].index)
```

[ ]:
```
df.shape
```

[ ]:
```
(606933, 33)
```

[135]:
```
#Writing a function which encompasses genders other than
#male or female onto other, as part of outlier detection
def defining_gender(gender):
```

```
  if str(gender)=='M':
    return('M')
  elif str(gender)=='F':
    return('F')
  else:
    return('O')
```

[136]:
```
df['Vict Sex'].value_counts()
```

[136]:
```
M    306421
F    293097
X      7328
H        87
Name: Vict Sex, dtype: int64
```

[137]:
```
#standardizing gender column
df['Vict Sex_New']=df['Vict Sex'].apply(defining_gender)
```

[30]:
```
df['Vict Sex_New'].value_counts()
```

[30]:
```
M    306421
F    293097
O      7415
Name: Vict Sex_New, dtype: int64
```

[138]:
```
#Writing a function that leaves only one whitespace between
#complex string values
import re

def keep_single_space_between_words(input_string):
  input_string=str(input_string)
  if input_string.isnumeric()==False:
    cleaned_string = re.sub(r'\s+', ' ', input_string)

    return cleaned_string
```

[139]:
```
df['LOCATION']=df['LOCATION'].apply(keep_single_space_between_words)
```

[140]:
```
df['Cross Street']=df['Cross Street'].apply(keep_single_space_between_words)
```

[ ]:
```
#checking updated columns
df.head()
```

[ ]:
```
       DR_NO  Date Rptd   DATE OCC TIME OCC  AREA    AREA NAME  Rpt Dist No  \
0   10304468 2020-01-08 2020-01-08    22:30     3    SOUTHWEST          377
1  190101086 2020-01-02 2020-01-01    03:30     1      CENTRAL          163
3  191501505 2020-01-01 2020-01-01    17:30    15  N HOLLYWOOD         1543
```

```
4  191921269 2020-01-01 2020-01-01    04:15    19      MISSION         1998
5  200100501 2020-01-02 2020-01-01    03:00     1      CENTRAL          163

   Part 1-2  Crm Cd                                       Crm Cd Desc  …  \
0         2     624                           BATTERY - SIMPLE ASSAULT  …
1         2     624                           BATTERY - SIMPLE ASSAULT  …
3         2     745          VANDALISM - MISDEAMEANOR ($399 OR UNDER)  …
4         2     740  VANDALISM - FELONY ($400 & OVER, ALL CHURCH VA…  …
5         1     121                                    RAPE, FORCIBLE  …

          LOCATION  Cross Street      LAT       LON  Rept Year Rept Month  \
0   1100 W 39TH PL           nan  34.0141 -118.2978       2020          1
1    700 S HILL ST           nan  34.0459 -118.2545       2020          1
3  5400 CORTEEN PL           nan  34.1685 -118.4019       2020          1
4   14400 TITUS ST           nan  34.2198 -118.4468       2020          1
5   700 S BROADWAY           nan  34.0452 -118.2534       2020          1

   Occurance Year Occurance Month Nos_of_mocodes Vict Sex_New
0            2020               1              2           F
1            2020               1              3           M
3            2020               1              2           F
4            2020               1              1           O
5            2020               1              4           F

[5 rows x 34 columns]
```

[ ]:
```
df[['DR_NO','Date Rptd','DATE OCC','TIME OCC','AREA NAME','Rept Year','Rept␣
↪Month','Occurance Year','Occurance Month','Nos_of_mocodes','Crm Cd␣
↪Desc','Mocodes','Vict Sex','LOCATION','Cross Street']].head()
```

[ ]:
```
       DR_NO  Date Rptd   DATE OCC TIME OCC    AREA NAME  Rept Year  \
0   10304468 2020-01-08 2020-01-08    22:30    SOUTHWEST       2020
1  190101086 2020-01-02 2020-01-01    03:30      CENTRAL       2020
3  191501505 2020-01-01 2020-01-01    17:30  N HOLLYWOOD       2020
4  191921269 2020-01-01 2020-01-01    04:15      MISSION       2020
5  200100501 2020-01-02 2020-01-01    03:00      CENTRAL       2020

   Rept Month  Occurance Year  Occurance Month  Nos_of_mocodes  \
0           1            2020                1               2
1           1            2020                1               3
3           1            2020                1               2
4           1            2020                1               1
5           1            2020                1               4

                                    Crm Cd Desc                Mocodes  \
0                      BATTERY - SIMPLE ASSAULT              0444, 0913
1                      BATTERY - SIMPLE ASSAULT       0416, 1822, 1414
```

```
3           VANDALISM - MISDEAMEANOR ($399 OR UNDER)              0329, 1402
4  VANDALISM - FELONY ($400 & OVER, ALL CHURCH VA…                   0329
5                                    RAPE, FORCIBLE  0413, 1822, 1262, 1415

  Vict Sex         LOCATION Cross Street
0        F   1100 W 39TH PL          nan
1        M    700 S HILL ST          nan
3        F  5400 CORTEEN PL          nan
4        X   14400 TITUS ST          nan
5        F   700 S BROADWAY          nan
```

[141]:
```python
#Writing a function that gives a month name instead of
# month number
def number_to_month(month_number):
    # Define a dictionary mapping month numbers to month names
    month_dict = {
        1: 'January',
        2: 'February',
        3: 'March',
        4: 'April',
        5: 'May',
        6: 'June',
        7: 'July',
        8: 'August',
        9: 'September',
        10: 'October',
        11: 'November',
        12: 'December'
    }

    try:
        month_name = month_dict[month_number]
        return month_name
    except KeyError:
        return None  # Return None for invalid month numbers
```

[142]:
```python
df['Month Name']=df['Rept Month'].apply(number_to_month)
```

[36]:
```python
df['Month Name'].value_counts()
```

[36]:
```
July         56525
August       56434
June         54671
May          54549
September    53704
March        53201
January      53098
```

```
April        51988
February     51667
October      41909
December     39893
November     39294
Name: Month Name, dtype: int64
```

[143]:
```
#adding an additional column, would help in calculating the frequecy
#of catagorical variables.
df['to_summarize']=1
```

[ ]:
```
df['DATE OCC'].value_counts()
```

[ ]:
```
2020-01-01    965
2022-12-02    916
2022-10-01    863
2023-02-01    852
2023-01-01    848
              …
2020-05-31    303
2020-04-09    287
2023-09-30    279
2023-10-01    242
2023-10-02     29
Name: DATE OCC, Length: 1371, dtype: int64
```

[144]:
```
#getting a day of the week from date of crime occurance
from datetime import datetime
def day_of_the_week(date):
  date_str = date
  #date_obj = datetime.strptime(date_str, '%m-%d-%Y')
  day_of_week = date_str.strftime('%A')
  return day_of_week
```

[145]:
```
df['Day_of_week']=df['DATE OCC'].apply(day_of_the_week)
```

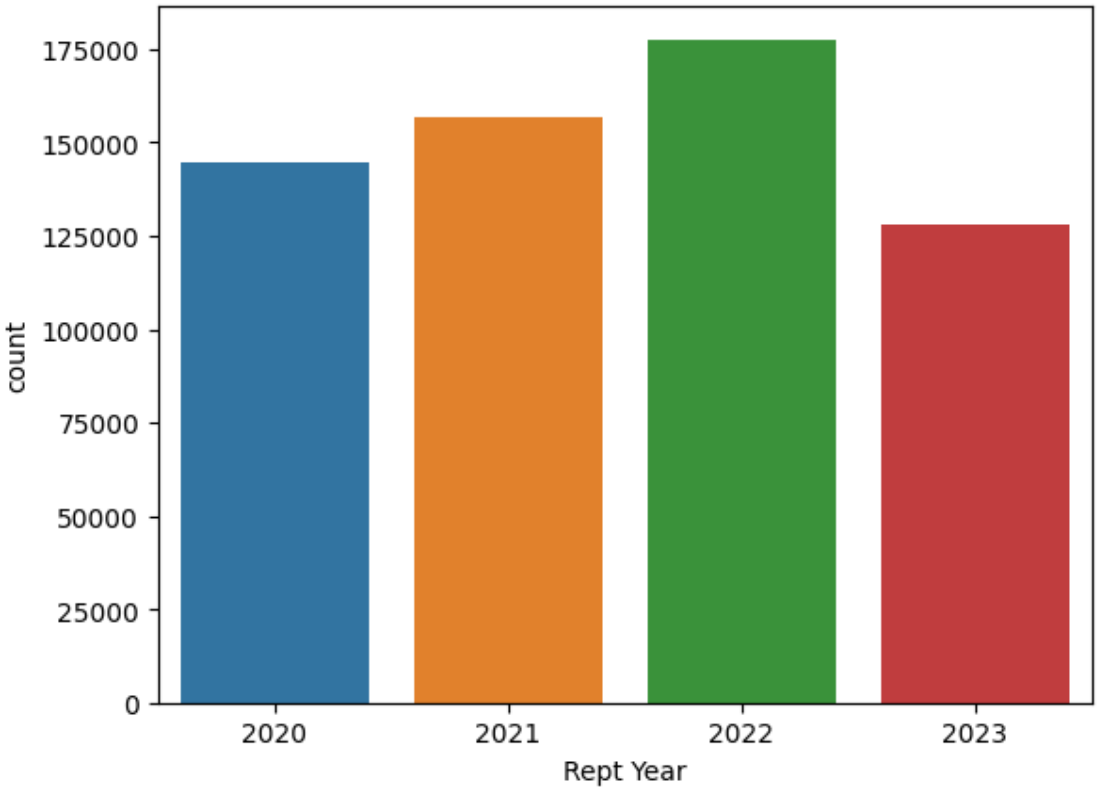
[40]:
```
df['Day_of_week'].value_counts()
```

[40]:
```
Friday       91407
Saturday     89573
Monday       86037
Sunday       85839
Wednesday    85779
Thursday     85339
Tuesday      82959
Name: Day_of_week, dtype: int64
```

Data Visualisation

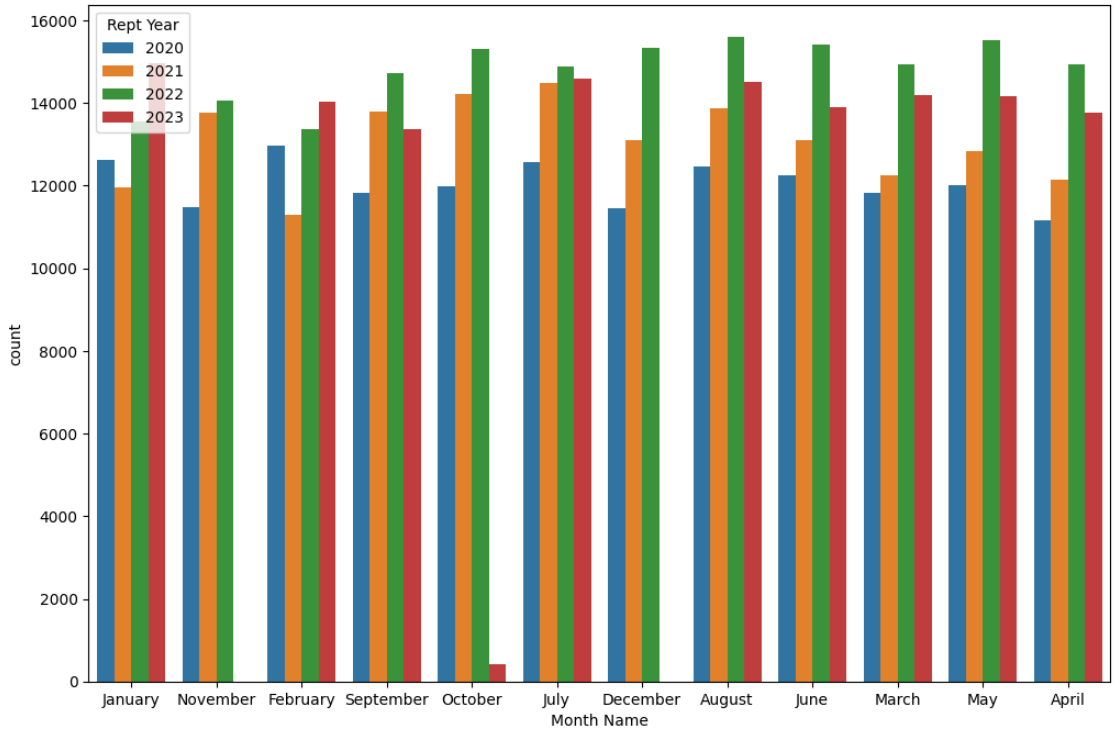
[ ]:
```
#Total Number of crimes reported from 2020 to 2023
sns.countplot(data=df,x='Rept Year')
#Observation: Shows an increasing trend in the amount of crime from 2020 to␣
 ↪2022, and a reduction in 2023
#reduction could also be possible due to the inavailibility of future data for␣
 ↪2023
```

[ ]:
```
<Axes: xlabel='Rept Year', ylabel='count'>
```



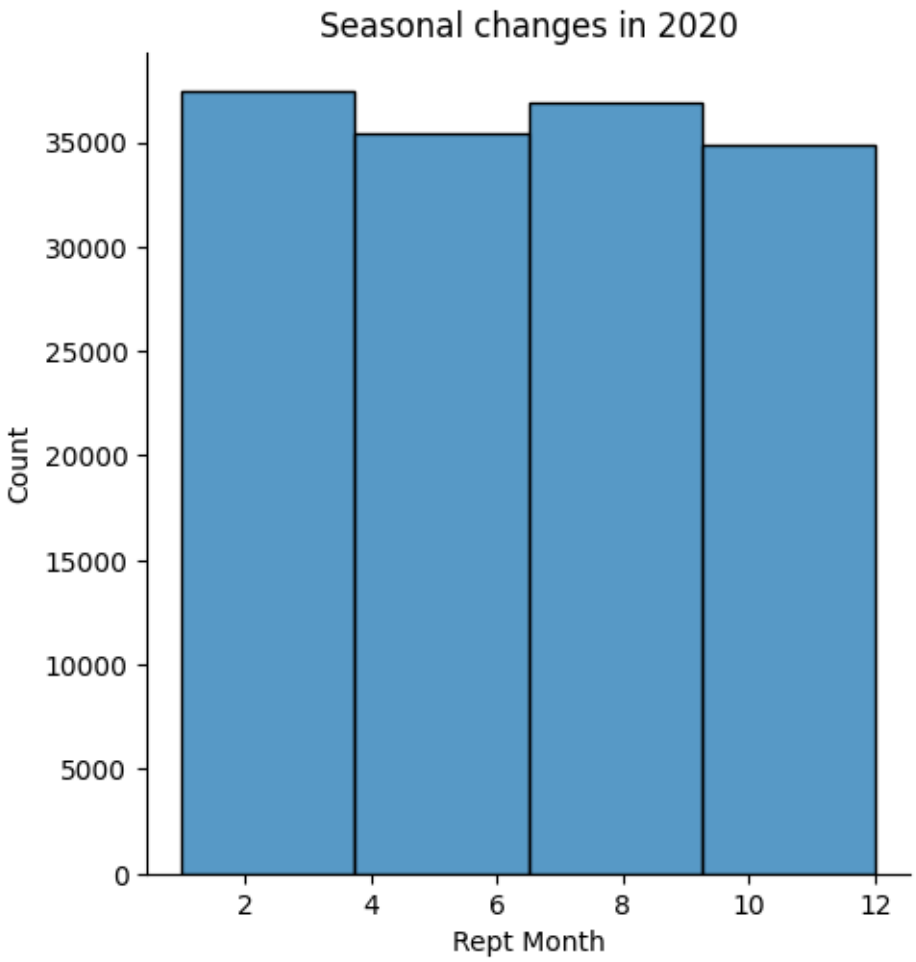
[ ]:
```
#Year wise data w.r.t month
plt.figure(figsize=(12,8))
sns.countplot(data=df,x='Month Name',hue='Rept Year')
#month wise analysis shows that for the number of months where the data for␣
 ↪2023 is available,
#there is a decreament of crime rate in all months for 2023 except jan 2023 and␣
 ↪feb 2023
#indicating that some sort of policy level change has been made in feb2023,␣
 ↪causing
#a massive reduction in the overall crime rate.
```

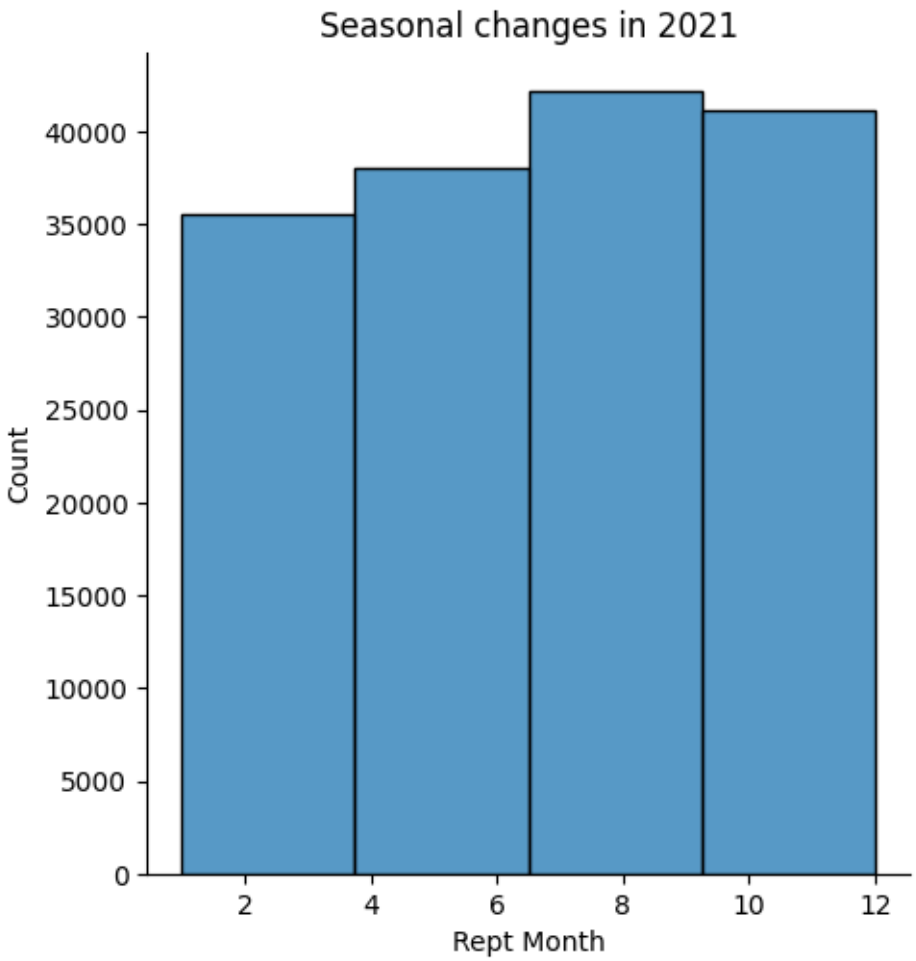
[ ]: <Axes: xlabel='Month Name', ylabel='count'>

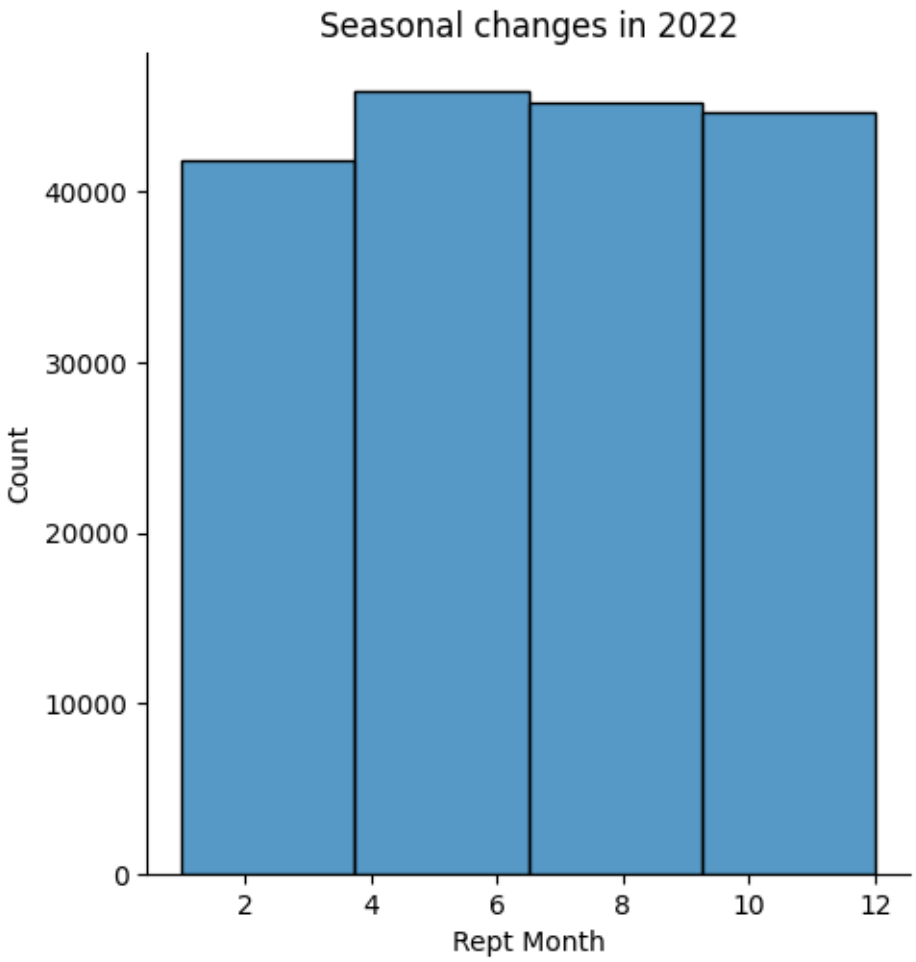


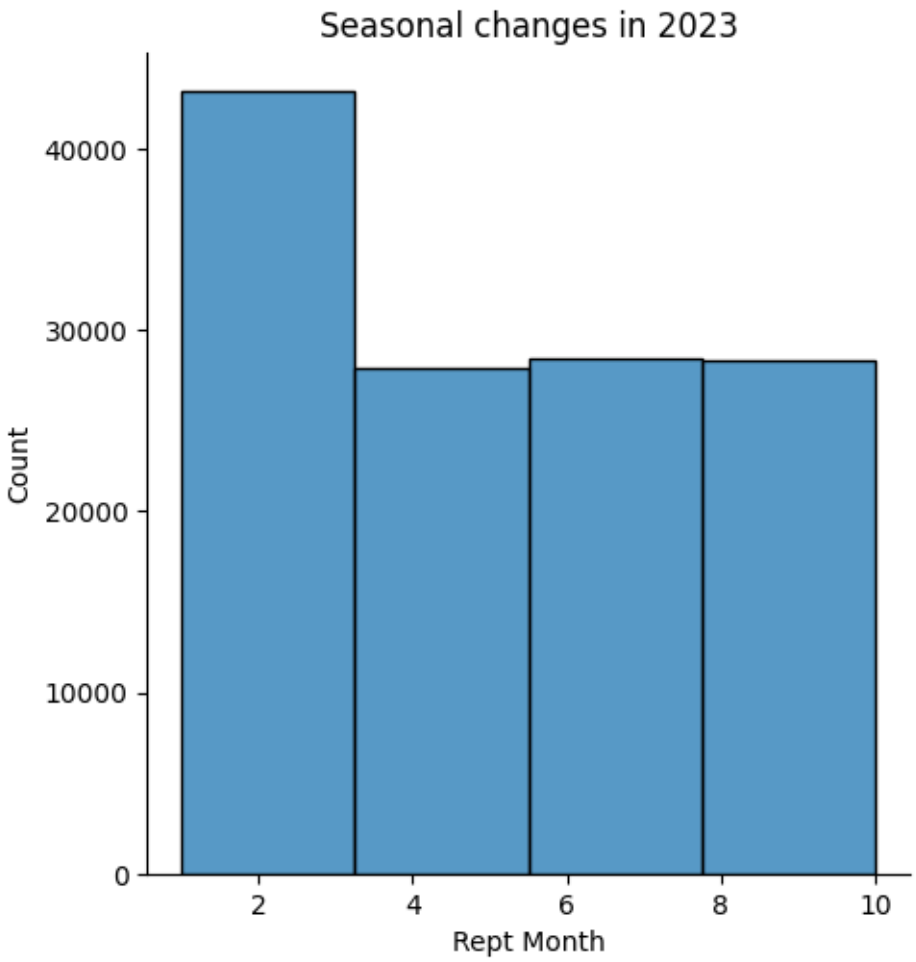
[ ]:
```
#seasonal changes year wise
sns.displot(data=df[df['Rept Year']==2020],x='Rept Month',bins=4,)
ax1 = plt.gca()
ax1.set_title('Seasonal changes in 2020')

sns.displot(data=df[df['Rept Year']==2021],x='Rept Month',bins=4,)
ax2 = plt.gca()
ax2.set_title('Seasonal changes in 2021')

sns.displot(data=df[df['Rept Year']==2022],x='Rept Month',bins=4,)
ax3 = plt.gca()
ax3.set_title('Seasonal changes in 2022')

sns.displot(data=df[df['Rept Year']==2023],x='Rept Month',bins=4,)
ax4 = plt.gca()
ax4.set_title('Seasonal changes in 2023')

#Majority of crime occurs between 6th and 9th month of the year, essantially in␣
 ↪the middle of the year.
```

[ ]: Text(0.5, 1.0, 'Seasonal changes in 2023')


Seasonal changes in 2020
35000
30000
25000
20000
15000
10000
5000
0
Count
2
4
6
8
10
12
Rept Month


Seasonal changes in 2021
40000
35000
30000
25000
20000
15000
10000
5000
0
Count
2
4
6
8
10
12
Rept Month


Seasonal changes in 2022
Count
40000
30000
20000
10000
0
2
4
6
8
10
12
Rept Month



[ ]:
```
#Saving the most common type of crime, top 5
temp_df=pd.DataFrame(df['Crm Cd Desc'].value_counts().head(5))
temp_df.reset_index(inplace=True)
temp_df.columns=['Crime','amount_of_occurance']
temp_list=list(temp_df['Crime'])
```

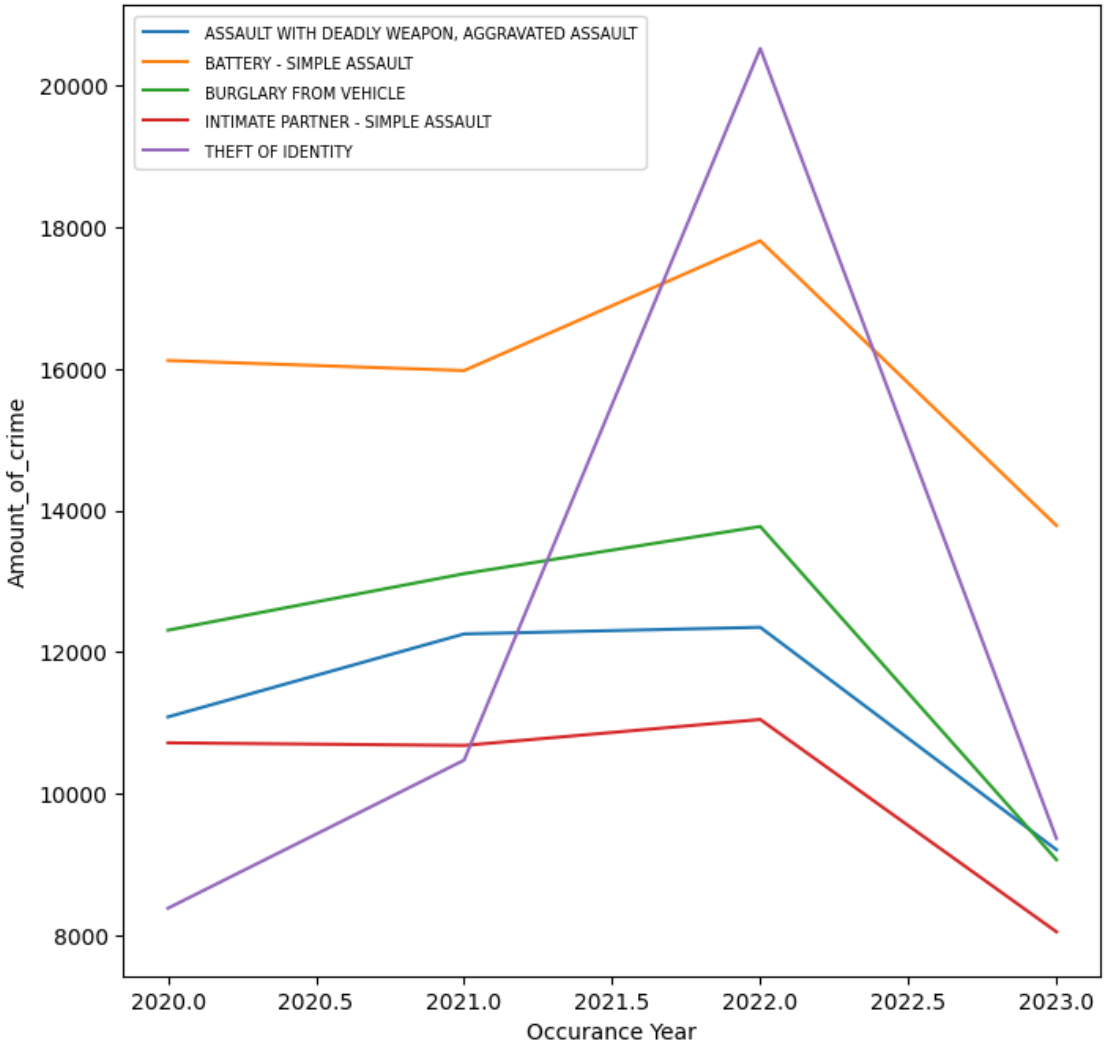
[ ]:
```
df_1=df[['Crm Cd Desc','to_summarize','Occurance Year']][df['Crm Cd Desc'].
  ↪isin(temp_list)]
df_2=pd.DataFrame(df_1.groupby(['Occurance Year','Crm Cd Desc']).sum())
df_2.reset_index(inplace=True)
```

[ ]:
```
df_2.columns=['Occurance Year','Crm_Cd_Desc','Amount_of_crime']
```

[ ]:
```
#distribution of most common type of crimes over the years
plt.figure(figsize=(8, 8))
sns.lineplot(data=df_2, x="Occurance Year",␣
  ↪y="Amount_of_crime",hue='Crm_Cd_Desc')
```

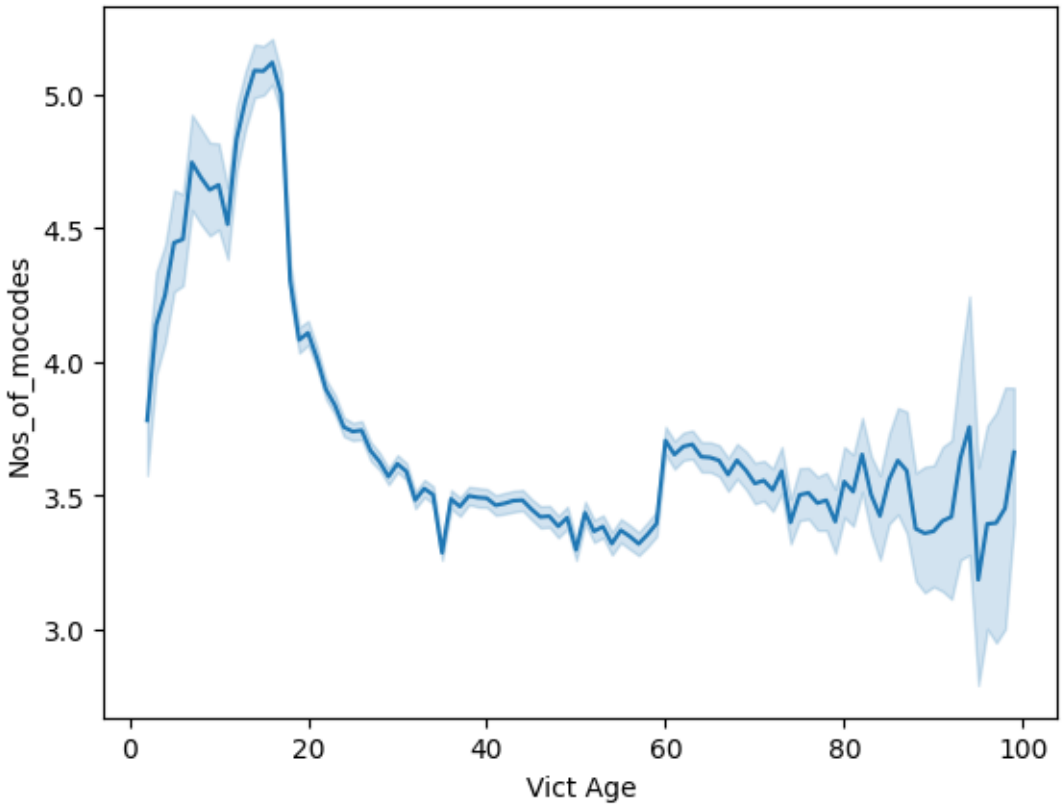
```
plt.legend(loc="upper left", title="Crm Cd Desc", fontsize="small")
plt.setp(plt.legend().get_title(), fontsize="small")
plt.setp(plt.legend().get_texts(), fontsize="x-small")
```

[ ]: [None, None, None, None, None, None, None, None, None, None]



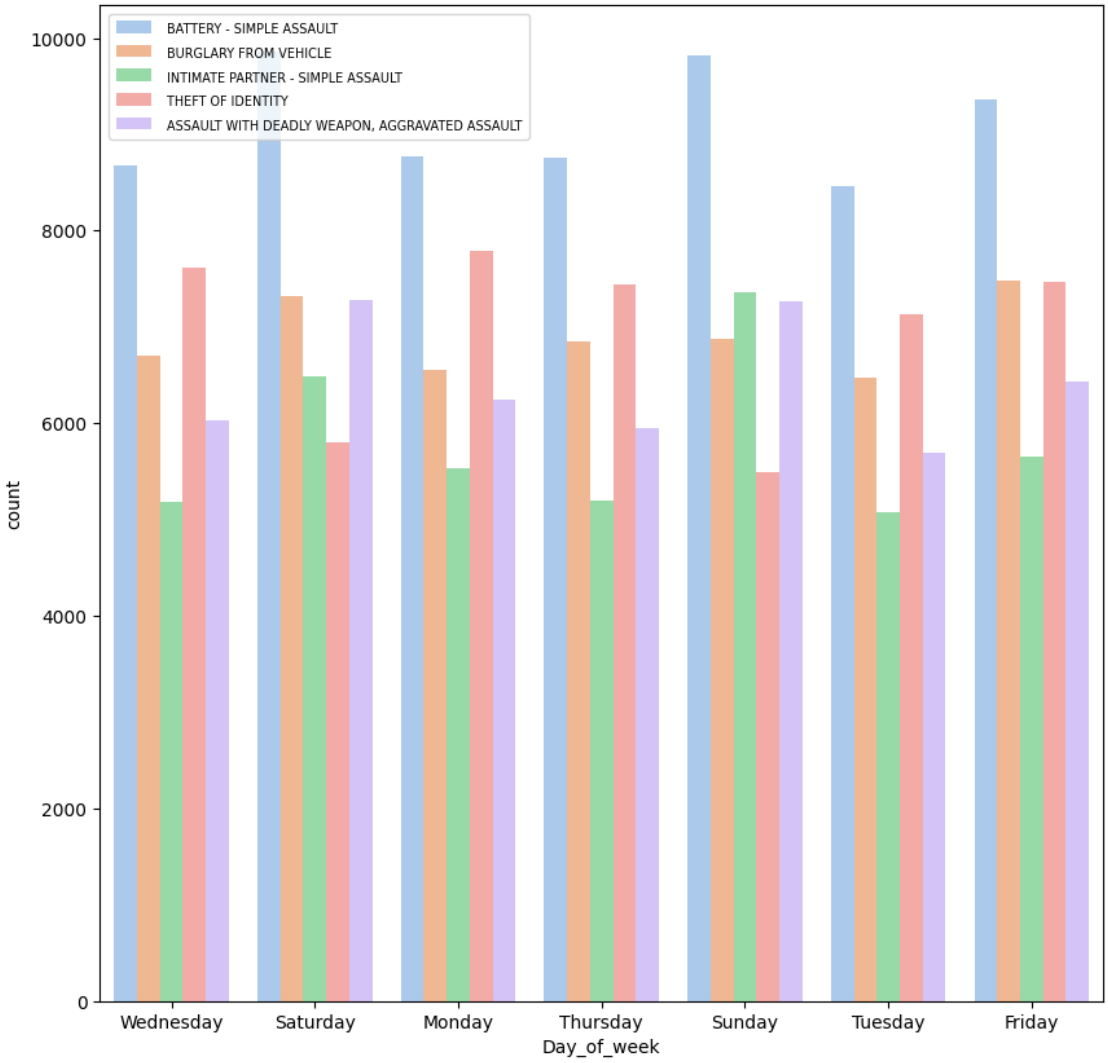
[ ]:
```
#Plotting relation between age of victims and amount of crime committed on them
↪via the number of mocodes
sns.lineplot(data=df, x="Vict Age", y="Nos_of_mocodes")
```

[ ]: <Axes: xlabel='Vict Age', ylabel='Nos_of_mocodes'>



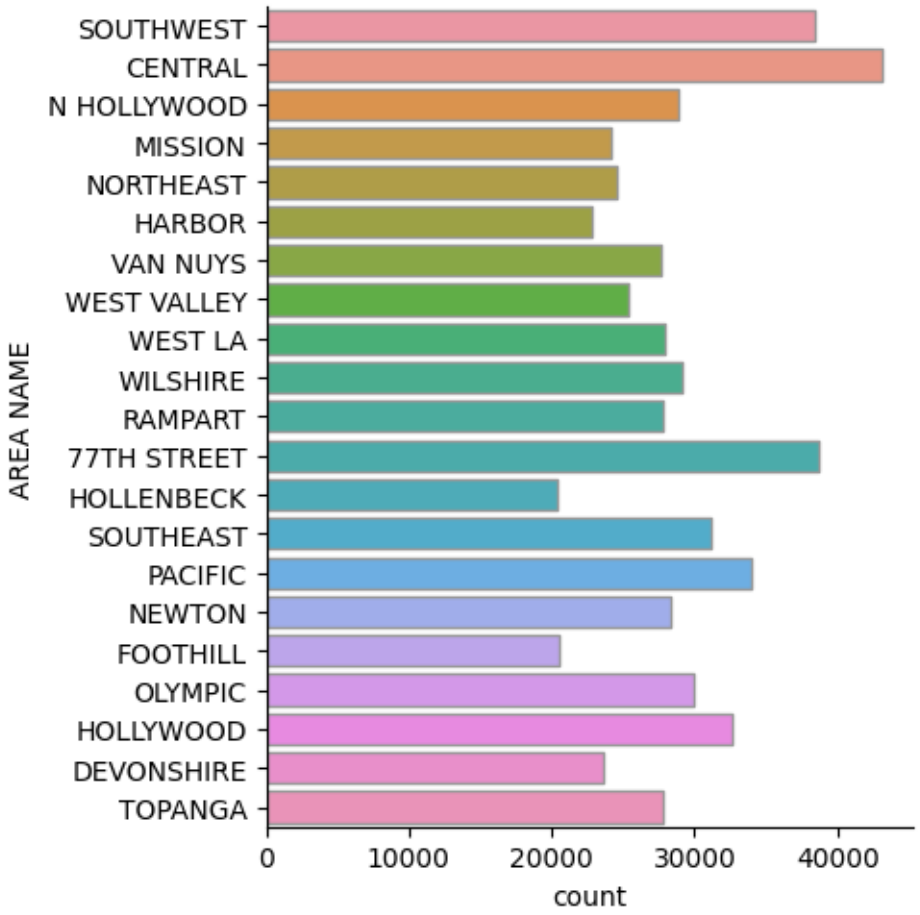
[ ]:
```
df_1=df[['Day_of_week','Crm Cd Desc']][df['Crm Cd Desc'].isin(temp_list)]
#df_2=pd.DataFrame(df_1.groupby(['Occurance Year','Crm Cd Desc']).sum())
#df_2.reset_index(inplace=True)
```

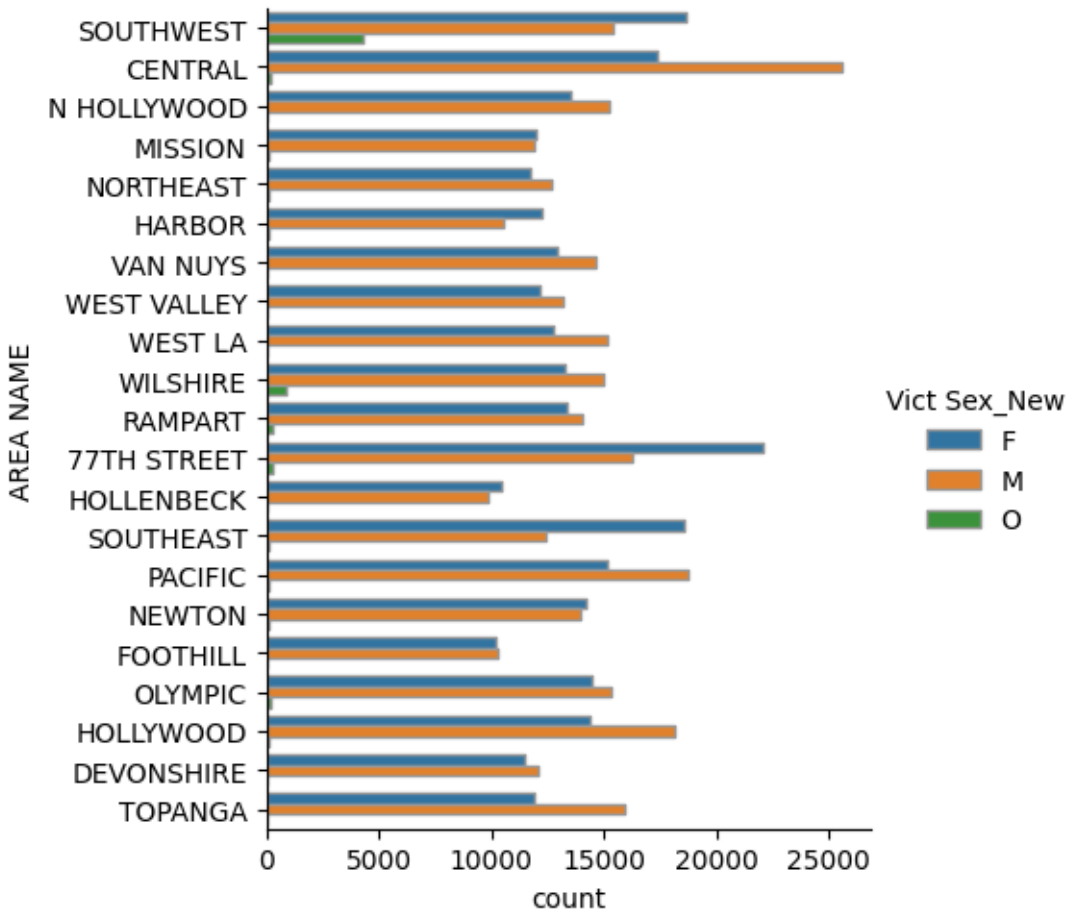
[ ]:
```
#distribution of most common types of crime wrt day of the week
plt.figure(figsize=(10, 10))
sns.countplot(data=df_1,x='Day_of_week',hue='Crm Cd Desc',palette="pastel")

plt.legend(loc="upper right", title="Crm Cd Desc", fontsize="small")
plt.setp(plt.legend().get_title(), fontsize="small")
plt.setp(plt.legend().get_texts(), fontsize="x-small")
plt.show()
```



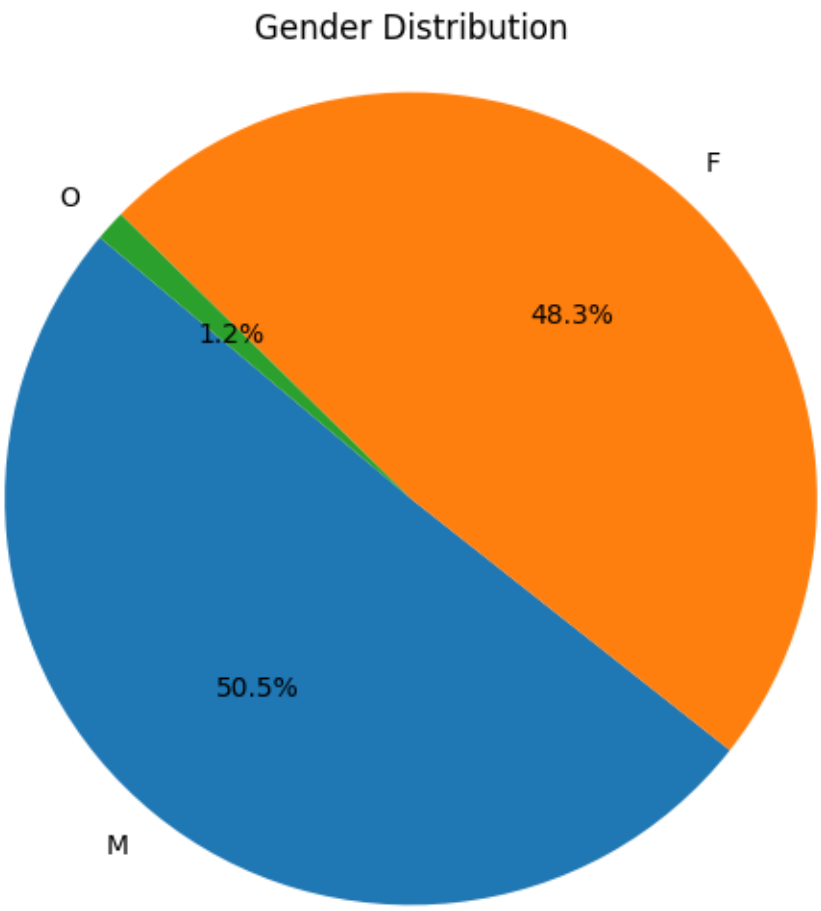
[ ]:
```
#distribution of crime wrt area and sex
sns.catplot(data=df, y="AREA NAME", kind="count", edgecolor=".6")
sns.catplot(data=df, y="AREA NAME", hue="Vict Sex_New", kind="count",␣
 ↪edgecolor=".6")
```

[ ]:
```
<seaborn.axisgrid.FacetGrid at 0x78c14cbc7850>
```


SOUTHWEST
CENTRAL
N HOLLYWOOD
MISSION
NORTHEAST
HARBOR
VAN NUYS
WEST VALLEY
WEST LA
WILSHIRE
RAMPART
77TH STREET
HOLLENBECK
SOUTHEAST
PACIFIC
NEWTON
FOOTHILL
OLYMPIC
HOLLYWOOD
DEVONSHIRE
TOPANGA
AREA NAME
0
10000
20000
30000
40000
count

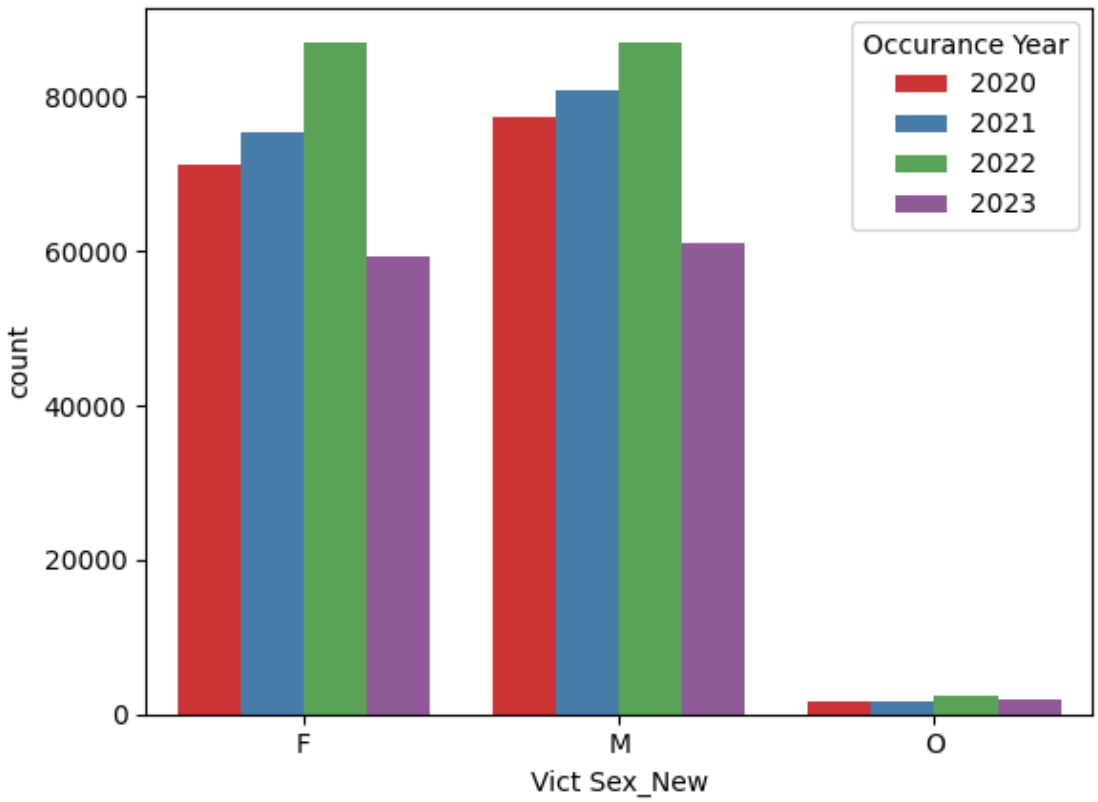


[ ]:
```
#total crime comitted on each sex
# Count the occurrences of each unique value in the 'gender' column
value_counts = df['Vict Sex_New'].value_counts()

# Create a pie chart
plt.figure(figsize=(6, 6))
plt.pie(value_counts, labels=value_counts.index, autopct='%1.1f%%',␣
 ↪startangle=140)
plt.title('Gender Distribution')
plt.axis('equal')  # Equal aspect ratio ensures that pie is drawn as a circle.
plt.show()
```



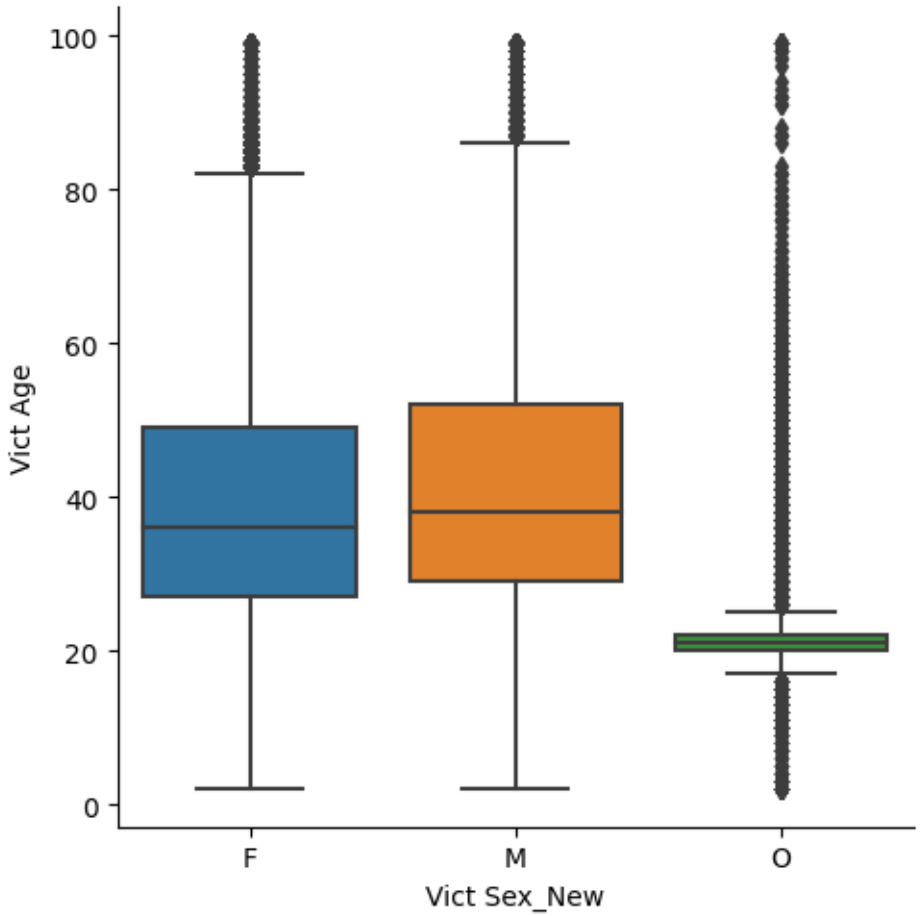
[ ]:
```
#crimes comitted on each sex wrt year
sns.countplot(data=df,x='Vict Sex_New',hue='Occurance Year',palette='Set1')
```

[ ]: <Axes: xlabel='Vict Sex_New', ylabel='count'>



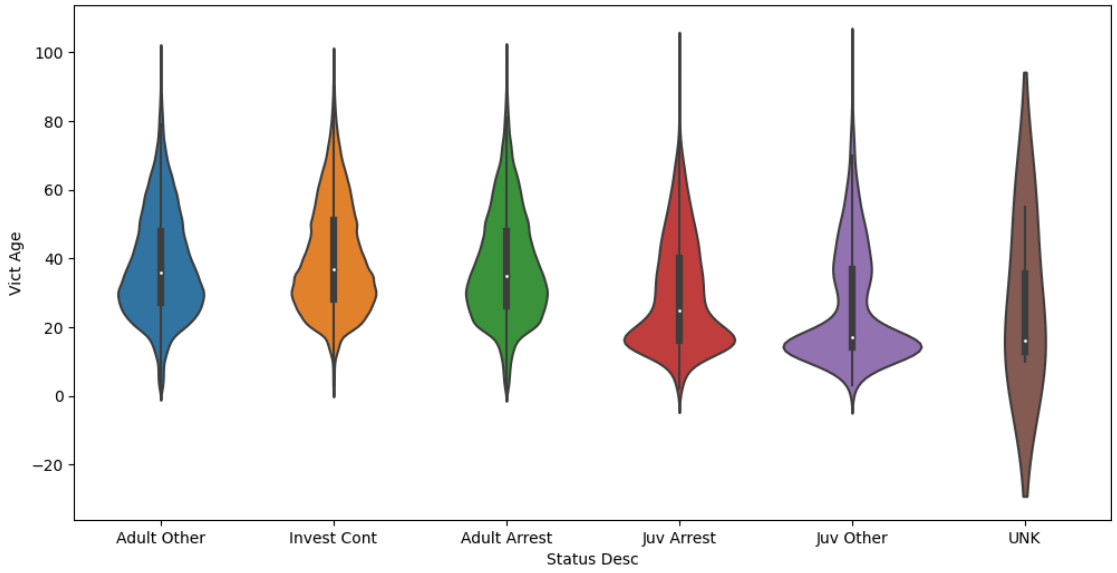
[ ]:
```
#mean age of crime comitted against men and women
sns.catplot(x='Vict Sex_New',y='Vict Age',data=df,kind='box')
```

[ ]:
```
<seaborn.axisgrid.FacetGrid at 0x78c105cf68f0>
```



[ ]:
```
plt.figure(figsize=(12,6))
sns.violinplot(x='Status Desc',y='Vict Age',data=df)
#action taken against reported crime wrt victim age
#obs: as the age increases, the chnaces of getting ban actual arrest decrease.
```

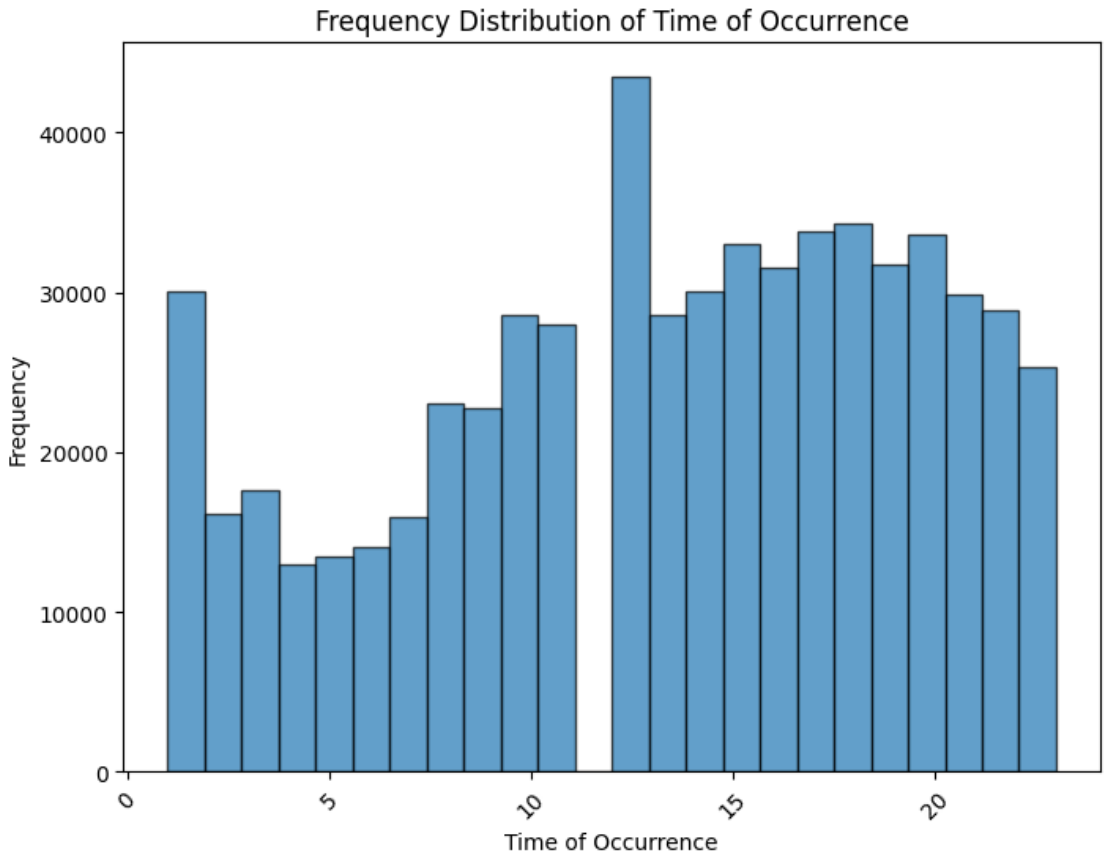
[ ]:
```
<Axes: xlabel='Status Desc', ylabel='Vict Age'>
```



[ ]:
```
df['TIME OCC']=np.where(df['TIME OCC']=='24:00','23:59',df['TIME OCC'])
```

[ ]:
```
df['TIME OCC_New'] = pd.to_datetime(df['TIME OCC'], format='%H:%M').dt.hour
```

[ ]:
```
#amount of crime vs time of the day (in terms of hours)
plt.figure(figsize=(8, 6))
plt.hist(df['TIME OCC_New'], bins=24, edgecolor='k', alpha=0.7)
plt.xlabel('Time of Occurrence')
plt.ylabel('Frequency')
plt.title('Frequency Distribution of Time of Occurrence')
plt.xticks(rotation=45)  # Rotate x-axis labels for better visibility
plt.show()
```



Advanced analysis- Time Series Analysis

[41]:
```
df['DATE OCC']=pd.to_datetime(df['DATE OCC'])
```

[42]:
```
df_date=df[['to_summarize','DATE OCC']]
df_date_data=df_date.groupby(['DATE OCC']).sum()
```

[43]:
```
df_date_data=pd.DataFrame(df_date_data)
```

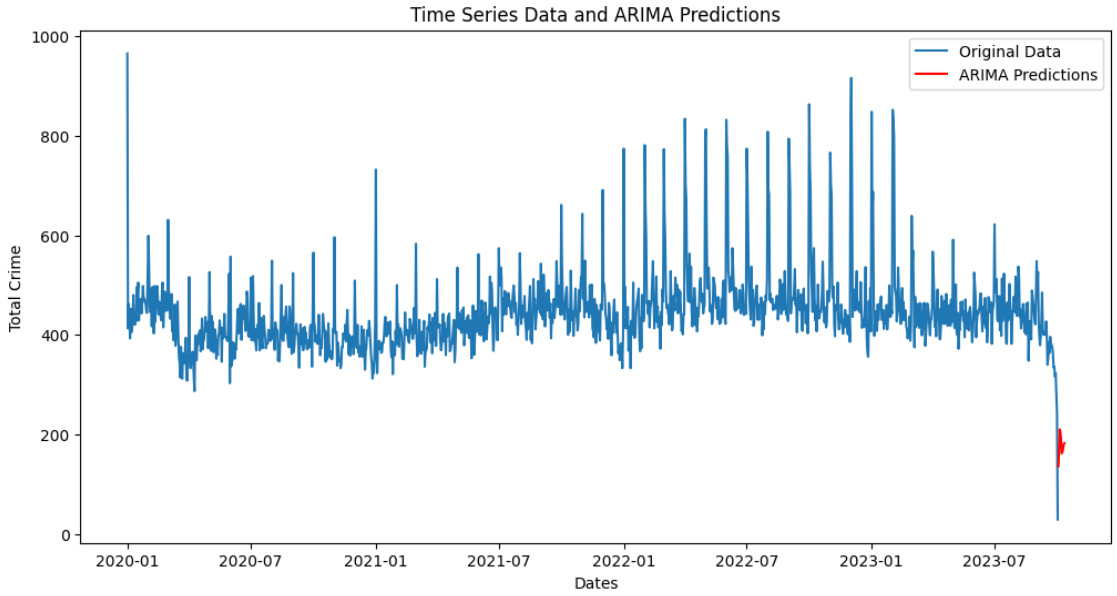
[44]:
```
df_date_data.columns
```

[44]:
```
Index(['to_summarize'], dtype='object')
```

[45]:
```
import pandas as pd
import matplotlib.pyplot as plt
from statsmodels.tsa.seasonal import seasonal_decompose
from statsmodels.tsa.stattools import adfuller
from statsmodels.tsa.arima.model import ARIMA
from statsmodels.tsa.statespace.sarimax import SARIMAX
```

```
# Fit an ARIMA model to the example time series data
model = ARIMA(df_date_data['to_summarize'], order=(5, 1, 0))
model_fit = model.fit()

# Make predictions
predictions = model_fit.forecast(steps=10)  # Predict the next 10 values

# Plot the original data and predictions
plt.figure(figsize=(12, 6))
plt.plot(df_date_data.index, df_date_data['to_summarize'], label='Original␣
 ↪Data')
plt.plot(pd.date_range(start='2023-10-03', periods=10, freq='D'), predictions,␣
 ↪color='red', label='ARIMA Predictions')
plt.xlabel('Dates')
plt.ylabel('Total Crime')
plt.title('Time Series Data and ARIMA Predictions')
plt.legend()
plt.show()
```

```
/usr/local/lib/python3.10/dist-packages/statsmodels/tsa/base/tsa_model.py:473:
ValueWarning: No frequency information was provided, so inferred frequency D
will be used.
  self._init_dates(dates, freq)
/usr/local/lib/python3.10/dist-packages/statsmodels/tsa/base/tsa_model.py:473:
ValueWarning: No frequency information was provided, so inferred frequency D
will be used.
  self._init_dates(dates, freq)
/usr/local/lib/python3.10/dist-packages/statsmodels/tsa/base/tsa_model.py:473:
ValueWarning: No frequency information was provided, so inferred frequency D
will be used.
  self._init_dates(dates, freq)
```



[46]:
```
print(predictions)
```

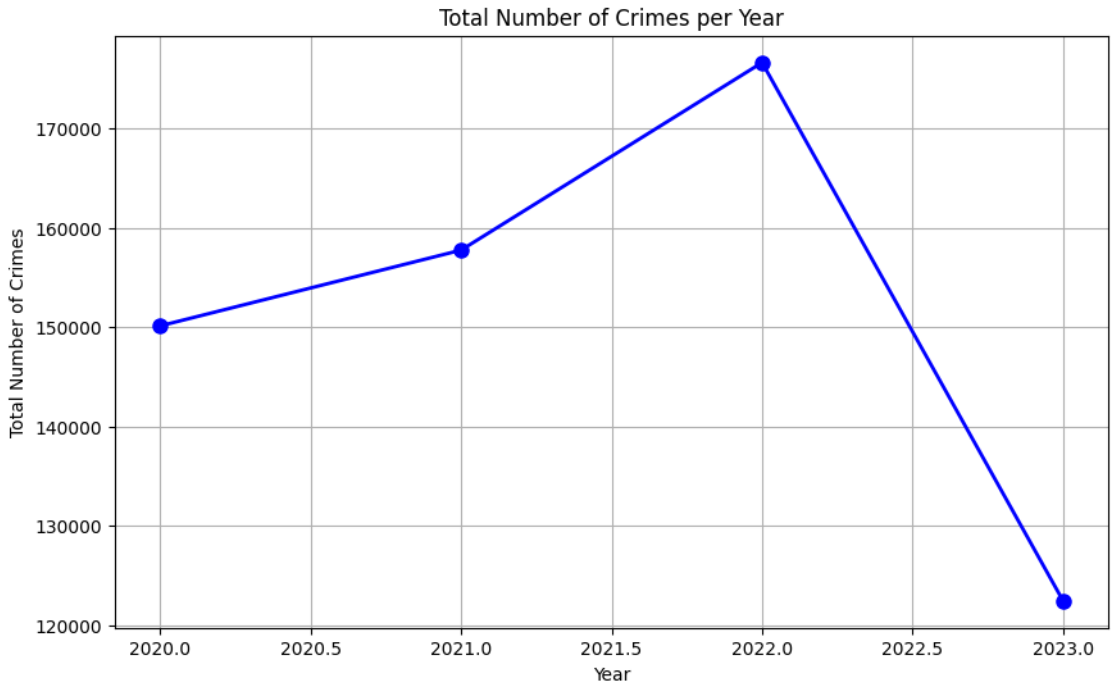
```
2023-10-03    136.004766
2023-10-04    169.415412
2023-10-05    209.929883
2023-10-06    204.090197
2023-10-07    186.491892
2023-10-08    161.921028
2023-10-09    164.771181
2023-10-10    172.029094
2023-10-11    180.885259
2023-10-12    182.284677
Freq: D, Name: predicted_mean, dtype: float64
```

Solutions to specific questions

[47]:
```
#Question-1 (Anagha)-Overall Crime trends
```

[54]:
```
crime_per_year = df.groupby('Occurance Year')['to_summarize'].sum().
 ↪reset_index()

plt.figure(figsize=(10, 6))
plt.plot(crime_per_year['Occurance Year'],crime_per_year['to_summarize'],␣
 ↪marker='o', color='b', linestyle='-', linewidth=2, markersize=8)
plt.title('Total Number of Crimes per Year')
plt.xlabel('Year')
plt.ylabel('Total Number of Crimes')
plt.grid(True)
```

```
plt.show()
```



[ ]:
```
#Question-2 (Jash)-Seasonal Patterns
```

[ ]:
```
#Already explained in exploratory data analysis, majority of crimes happen in␣
↪the middle of the year.
```

[ ]:
```
#Question-3 (Jash)-Most Common Crime Type
```

[96]:
```
#Saving the most common type of crime, top 10
temp_df=pd.DataFrame(df['Crm Cd Desc'].value_counts().head(10))
temp_df.reset_index(inplace=True)
temp_df.columns=['Crime','amount_of_occurance']
temp_list=list(temp_df['Crime'])
```

[98]:
```
df_1=df[['Crm Cd Desc','to_summarize']][df['Crm Cd Desc'].isin(temp_list)]
df_2=pd.DataFrame(df_1.groupby(['Crm Cd Desc']).sum())
df_2.reset_index(inplace=True)
```

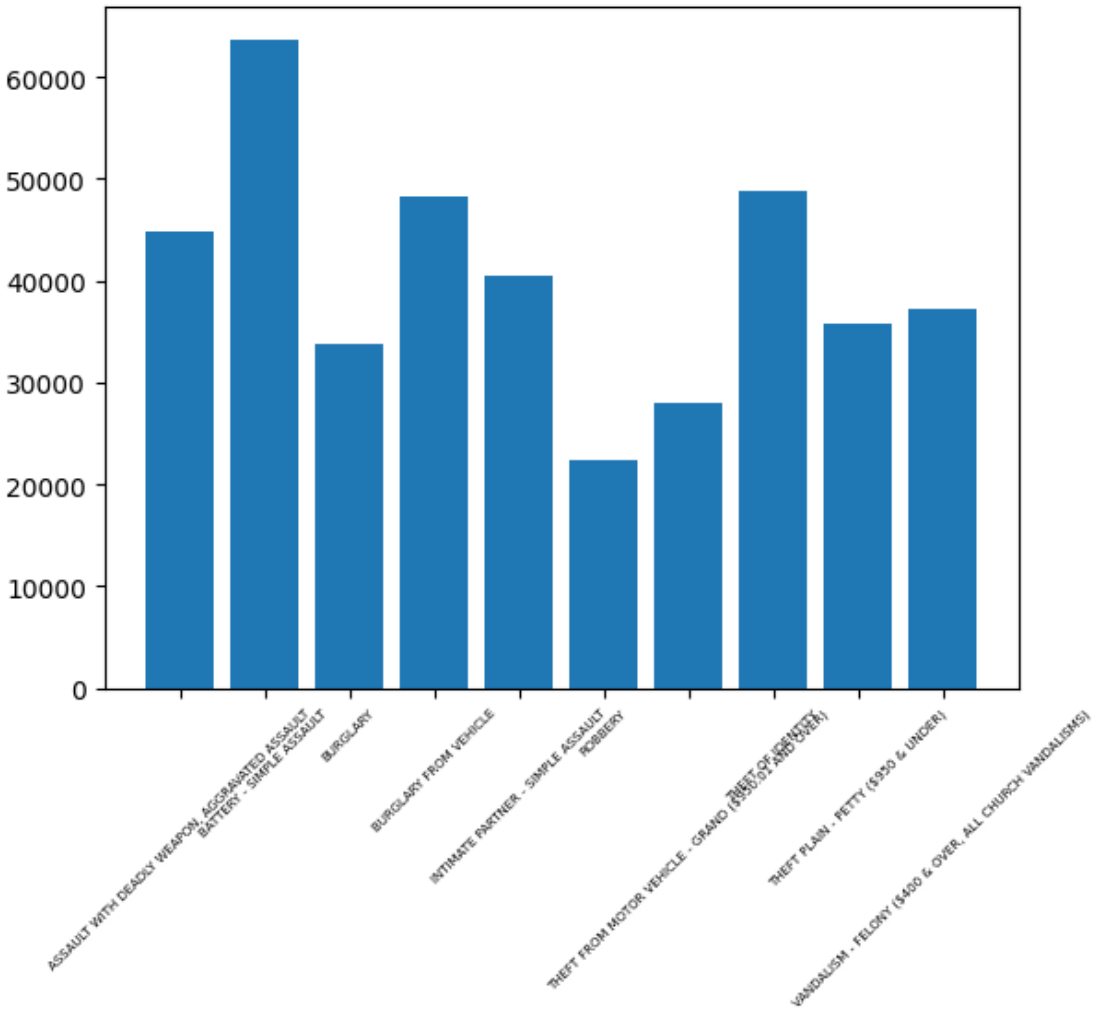
[103]:
```
df_2
```

[103]:

	Crm Cd Desc	to_summarize
0	ASSAULT WITH DEADLY WEAPON, AGGRAVATED ASSAULT	44891
1	BATTERY - SIMPLE ASSAULT	63697

```
2                                          BURGLARY         33790
3                              BURGLARY FROM VEHICLE         48253
4                   INTIMATE PARTNER - SIMPLE ASSAULT         40486
5                                           ROBBERY         22372
6  THEFT FROM MOTOR VEHICLE - GRAND ($950.01 AND …        27969
7                                 THEFT OF IDENTITY         48743
8                 THEFT PLAIN - PETTY ($950 & UNDER)         35787
9  VANDALISM - FELONY ($400 & OVER, ALL CHURCH VA…        37267
```

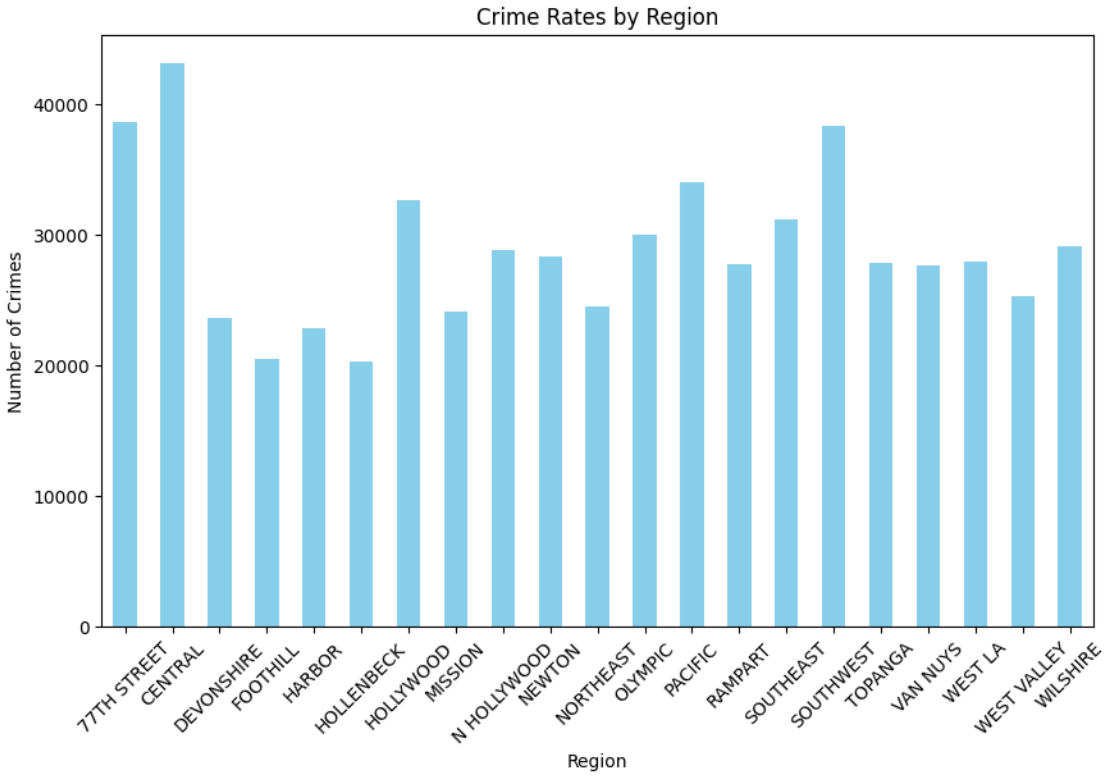
[106]:
```
plt.bar(df_2['Crm Cd Desc'],df_2['to_summarize'])
plt.xticks(rotation=45, fontsize=5)
plt.show()
```



[ ]:
```
#Question-4 (Rutuja)- Regional Differences
```

[146]:
```
# Group the data by region (in this case, 'AREA NAME') and calculate the total␣
 ↪number of crimes in each region
crime_by_region = df.groupby('AREA NAME')['Crm Cd Desc'].count()

# Calculate descriptive statistics for crime rates in different regions
crime_stats = crime_by_region.describe()
print(crime_stats)

# Example: Bar chart to compare crime rates between regions
plt.figure(figsize=(10, 6))
crime_by_region.plot(kind='bar', color='skyblue')
plt.title("Crime Rates by Region")
plt.xlabel("Region")
plt.ylabel("Number of Crimes")
plt.xticks(rotation=45)
plt.show()

#In the dataset, there are 21 regions or cities being analyzed.
#On average, each region experiences approximately 28,901 crimes. However,␣
 ↪there is a notable variation as
#indicated by the standard deviation of approximately 5,924 crimes above or␣
 ↪below the average. The region with the lowest
#crime rate sees around 20,332 crimes, while the highest crime rate in any␣
 ↪region is approximately 43,134 crimes. The middle region,
#or the median, experiences roughly 27,966 crimes. These statistics provide a␣
 ↪clear picture of how crime rates differ among the regions,
#with some having significantly higher or lower crime rates than others.
```

```
count       21.000000
mean     28901.571429
std       5923.526868
min      20332.000000
25%      24557.000000
50%      27966.000000
75%      31151.000000
max      43134.000000
Name: Crm Cd Desc, dtype: float64
```



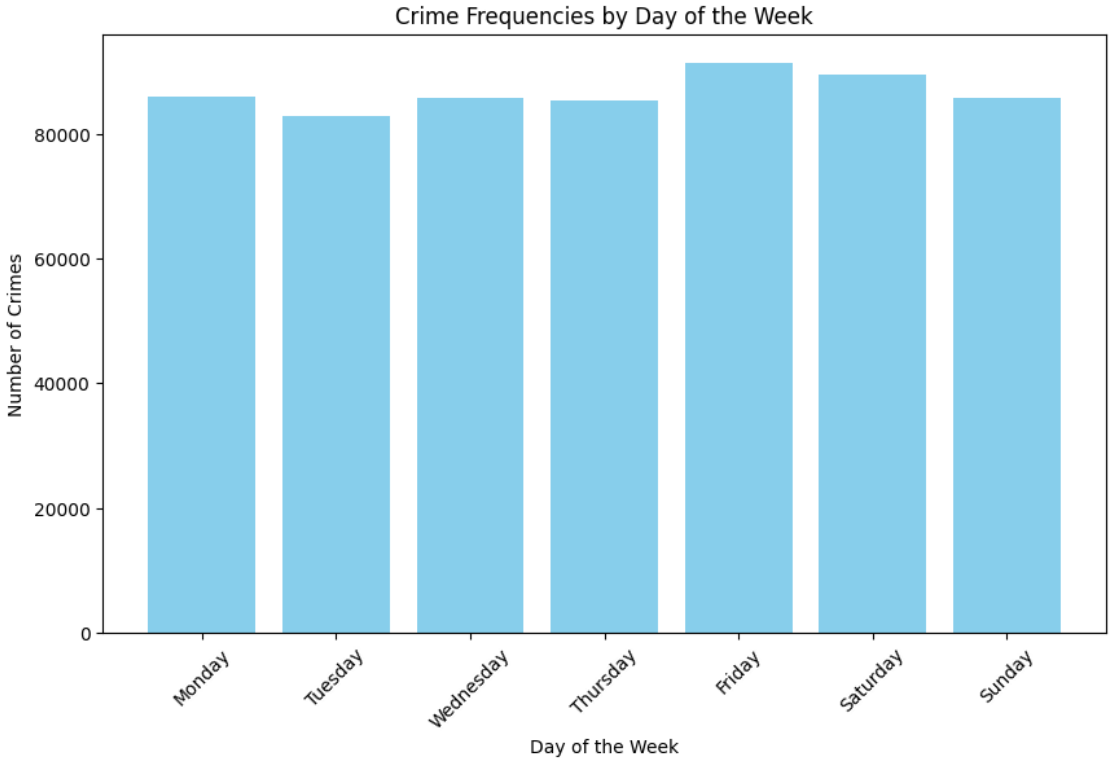
[147]:
```
#Question-5 (Saheel)- Correlation with economic factors
```

[148]:
```
#Economic factors are unavailable
```

[149]:
```
#Question-6 (Anagha)- Day of the week analysis
```

[55]:
```
df['DATE OCC'] = pd.to_datetime(df['DATE OCC'])
df['Day_of_Week'] = df['DATE OCC'].dt.day_name()
crime_by_day = df['Day_of_Week'].value_counts().reset_index()
crime_by_day.columns = ['Day_of_Week', 'Number_of_Crimes']
crime_by_day = crime_by_day.sort_values(by='Day_of_Week',
                                        key=lambda x: x.map({"Monday": 1,␣
 ↪"Tuesday": 2, "Wednesday": 3, "Thursday": 4, "Friday": 5, "Saturday": 6,␣
 ↪"Sunday": 7}))
import matplotlib.pyplot as plt

# Plotting the data
plt.figure(figsize=(10, 6))
plt.bar(crime_by_day['Day_of_Week'], crime_by_day['Number_of_Crimes'],␣
 ↪color='skyblue')
plt.title('Crime Frequencies by Day of the Week')
plt.xlabel('Day of the Week')
```

```
plt.ylabel('Number of Crimes')
plt.xticks(rotation=45)
plt.show()
```



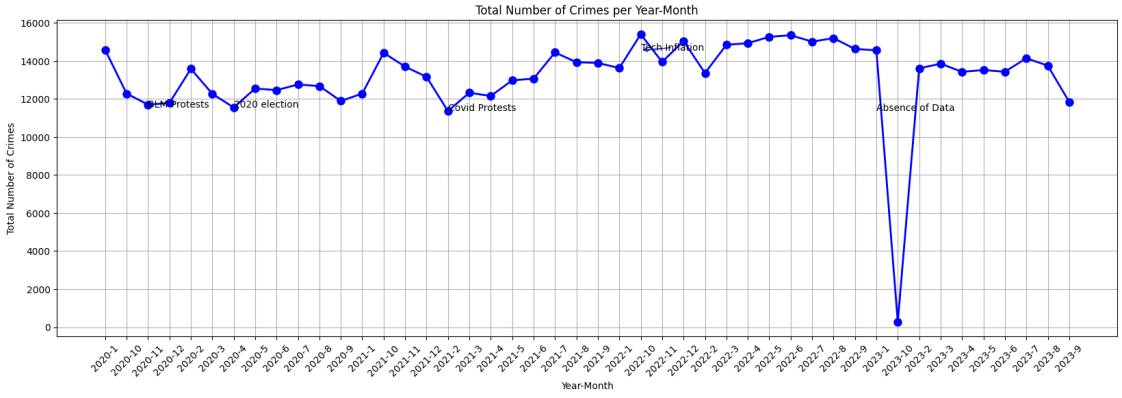
[ ]:
```
#Question-7 (Harsh)- Impact of major events
```

[66]:
```
def concatinate_string(a,b):
  c=str(a)+'-'+str(b)
  return(c)
```

[69]:
```
df['Year_Month'] = df.apply(lambda row: concatinate_string(row['Occurance␣
↪Year'], row['Occurance Month']), axis=1)
```

[82]:
```
crime_per_year_new = df.groupby('Year_Month')['to_summarize'].sum().
↪reset_index()
first_event=crime_per_year_new['to_summarize'][crime_per_year_new['Year_Month']=='2020-4']
second_event=crime_per_year_new['to_summarize'][crime_per_year_new['Year_Month']=='2021-2']
third_event=crime_per_year_new['to_summarize'][crime_per_year_new['Year_Month']=='2020-4']
fourth_event=crime_per_year_new['to_summarize'][crime_per_year_new['Year_Month']=='2023-1']
fifth_event=crime_per_year_new['to_summarize'][crime_per_year_new['Year_Month']=='2022-10']
```

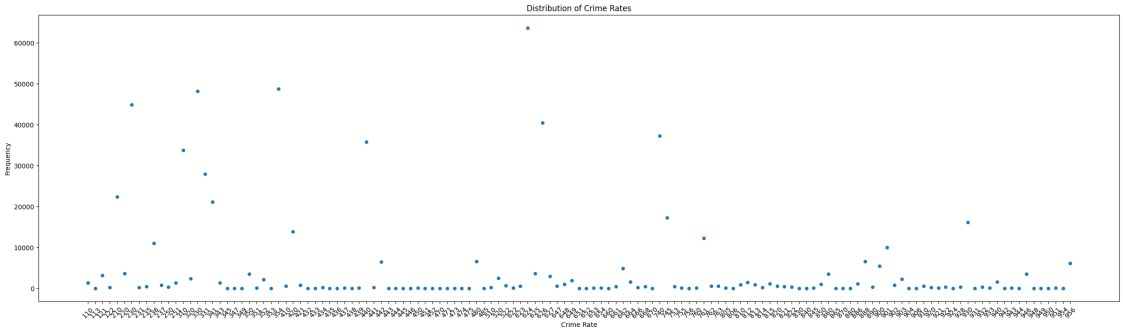
```
plt.figure(figsize=(20, 6))
plt.plot(crime_per_year_new['Year_Month'],crime_per_year_new['to_summarize'],␣
 ↪marker='o', color='b', linestyle='-', linewidth=2, markersize=8)
plt.title('Total Number of Crimes per Year-Month')
plt.xlabel('Year-Month')
plt.ylabel('Total Number of Crimes')
plt.xticks(rotation=45)
plt.grid(True)
annotations = [
    {'text': 'BLM Protests', 'xy': ('2020-11',first_event)},
    {'text': 'Covid Protests', 'xy': ('2021-2', second_event)},
    {'text': '2020 election', 'xy': ('2020-4', third_event)},
    {'text': 'Absence of Data', 'xy': ('2023-1', second_event)},
    {'text': 'Tech Inflation', 'xy': ('2022-10', fourth_event),
     'arrowprops': {'arrowstyle': '->', 'color': 'blue'}},
]

for annotation in annotations:
    plt.annotate(annotation['text'], xy=annotation['xy'],
                 arrowprops=annotation.get('arrowprops', None))
plt.show()
```



[ ]:
```
#Question-8 (Harsh)- Outliers and anamolies
```

[92]:
```
import seaborn as sns
crime_per_code = df.groupby('Crm Cd')['to_summarize'].sum().reset_index()
crime_per_code['Crm Cd']=crime_per_code['Crm Cd'].astype(str)
plt.figure(figsize=(30,8))
sns.scatterplot(data=crime_per_code,x=crime_per_code['Crm␣
 ↪Cd'],y=crime_per_code['to_summarize'])
plt.title('Distribution of Crime Rates')
plt.xlabel('Crime Rate')
```

```
plt.ylabel('Frequency')
plt.xticks(rotation=45)
plt.show()
```



[95]:
```
#Anomalies
crime_per_code['Crm Cd'][crime_per_code['to_summarize']>=35000].value_counts()
```

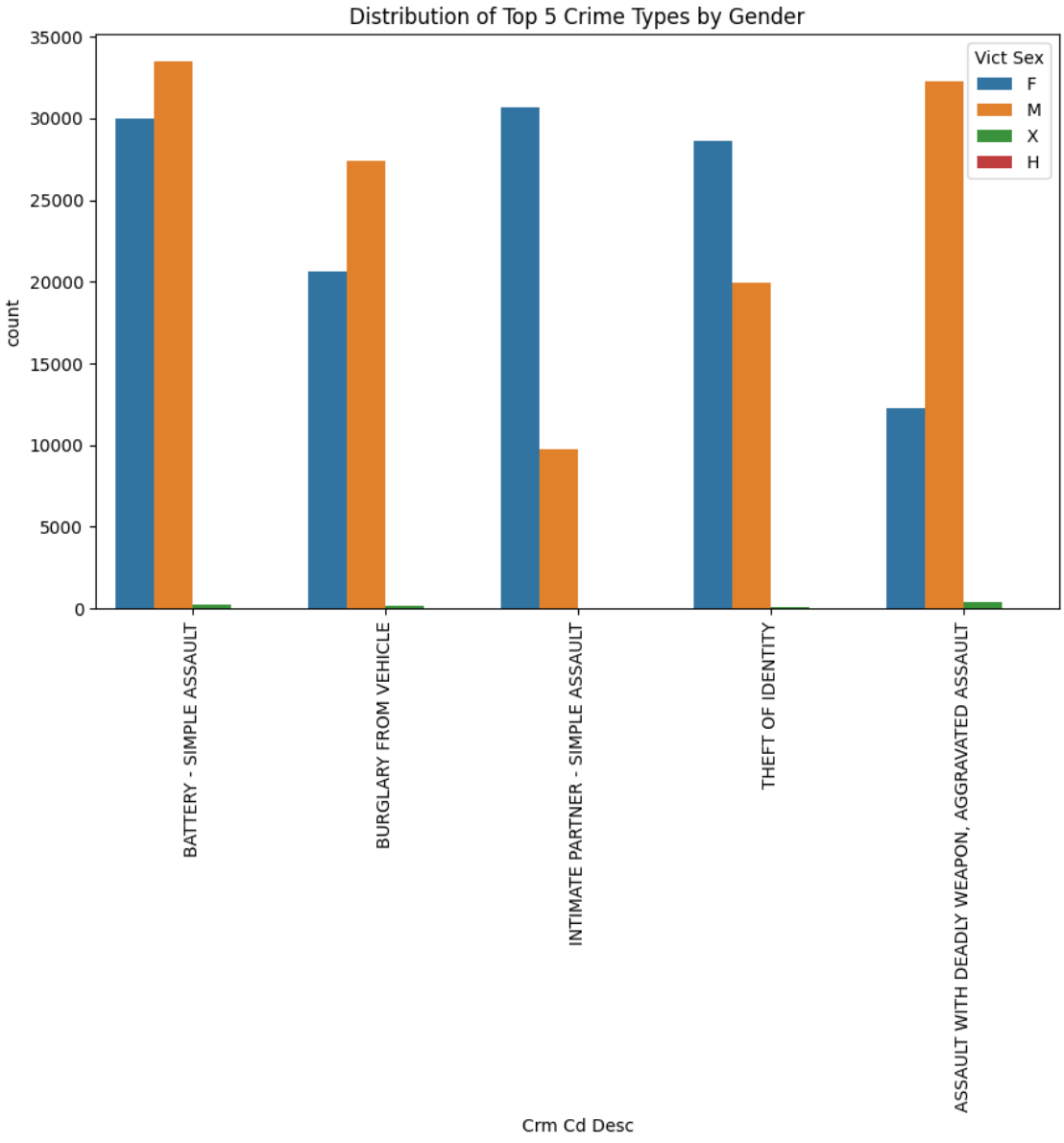
[95]:
```
230    1
330    1
354    1
440    1
624    1
626    1
740    1
Name: Crm Cd, dtype: int64
```

[ ]:
```
#Question-9 (Saheel)- Demographic Factors
```

[150]:
```
# Find the top 5 most frequent crime types
top_5_crimes = df['Crm Cd Desc'].value_counts().nlargest(5).index

# Filter the dataset to include only the top 5 crime types
df_top_5_crimes = df[df['Crm Cd Desc'].isin(top_5_crimes)]

# Calculate the correlation between age and crime code for the top 5 crimes
age_crime_correlation = df_top_5_crimes['Vict Age'].corr(df_top_5_crimes['Crm␣
 ↪Cd'])

# Perform a chi-squared test for gender and crime type (assuming 'Vict Sex' and␣
 ↪'Crm Cd' are categorical) for the top 5 crimes
from scipy.stats import chi2_contingency

contingency_table = pd.crosstab(df_top_5_crimes['Vict Sex'],␣
 ↪df_top_5_crimes['Crm Cd'])
```

```
chi2, p, _, _ = chi2_contingency(contingency_table)

# Print or analyze the correlations and test results for the top 5 crimes
print("Age-Crime Correlation for Top 5 Crimes:", age_crime_correlation)
print("Gender-Crime Chi-Squared Test p-value for Top 5 Crimes:", p)

# Visualize the distribution of the top 5 crime types by gender
plt.figure(figsize=(10, 6))
sns.countplot(data=df_top_5_crimes, x='Crm Cd Desc', hue='Vict Sex')
plt.title("Distribution of Top 5 Crime Types by Gender")
plt.xticks(rotation=90)
plt.show()

#Age and Crime Connection: The data shows a very weak link between a person's␣
 ↪age and the occurrence of
#the top 5 most frequent crimes. In simpler terms, as people get older, there␣
 ↪is only a slight tendency for these
#crimes to occur less often. The correlation between age and these crimes is␣
 ↪minimal.

#Gender and Crime Association: Gender has a significant impact on these␣
 ↪specific crimes. This means that these particular
#crimes are more likely to happen to people of one gender compared to the other.
 ↪ The data strongly supports a connection between gender and
#the occurrence of these top 5 crimes.
```

```
Age-Crime Correlation for Top 5 Crimes: -0.017505092699881233
Gender-Crime Chi-Squared Test p-value for Top 5 Crimes: 0.0
```

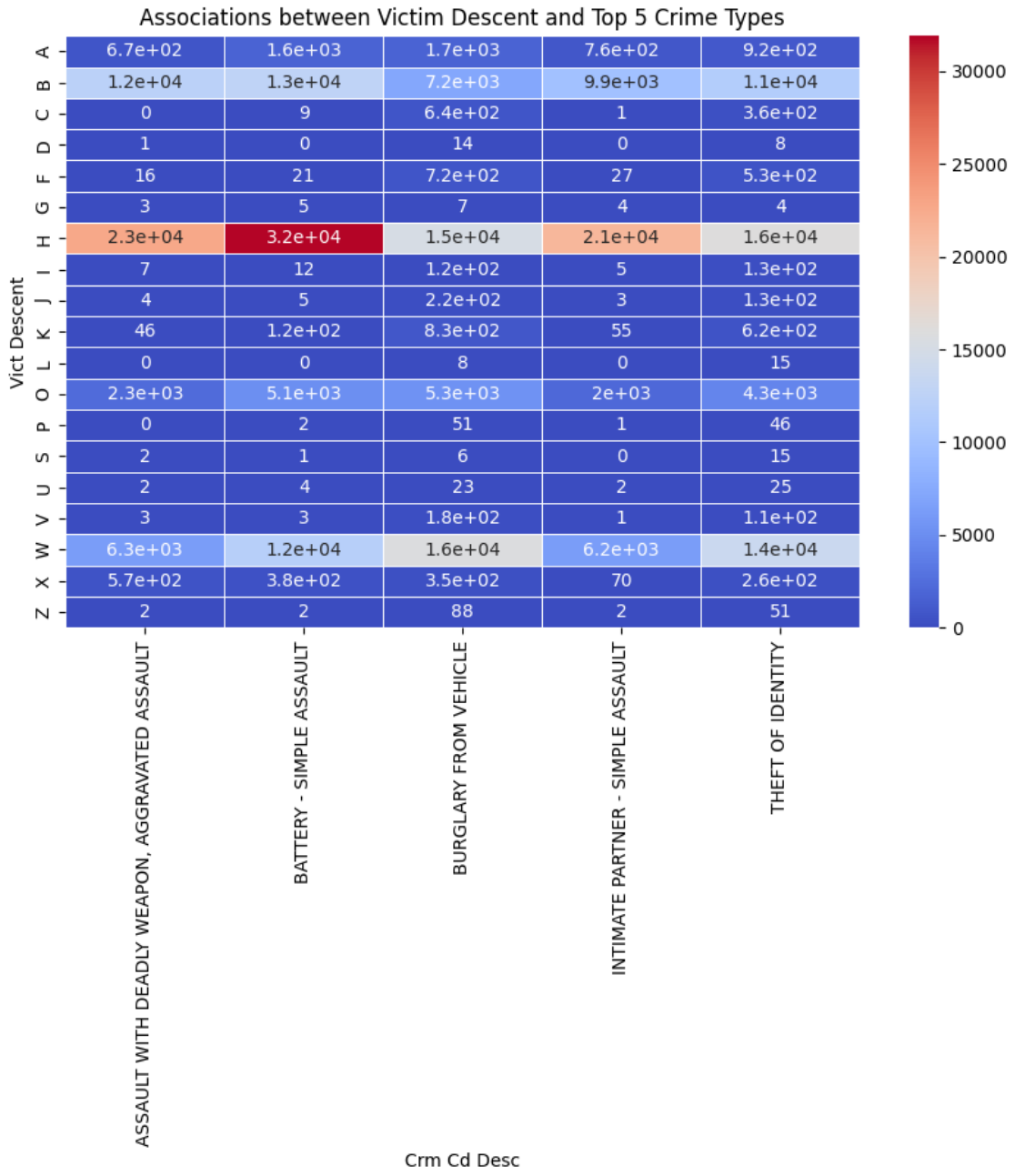


n top 5 crimes, 3 of them males have the higher majority of victims namely in Battery - Simple Asssault, Burglary from vehicle, Assault with deadly weapon - Aggrevated weapon. Whereas Females have more majority in Intimate Partner - Simple Assault, Theft of Identity. In all the top 5 occuring crimes Other sex categories have the least occurences.

[151]:
```
# Perform a chi-squared test for victim descent and crime type (assuming 'Vict␣
 ↪Descent' and 'Crm Cd Desc' are categorical) for the top 5 crimes
from scipy.stats import chi2_contingency

contingency_table = pd.crosstab(df_top_5_crimes['Vict Descent'],␣
 ↪df_top_5_crimes['Crm Cd Desc'])
```

```
chi2, p, _, _ = chi2_contingency(contingency_table)

# Print or analyze the chi-squared test results
print("Victim Descent-Crime Chi-Squared Test p-value for Top 5 Crimes:", p)

# Example: Heatmap to visualize associations between victim descent and the top␣
 ↪5 crime types
plt.figure(figsize=(10, 6))
sns.heatmap(contingency_table, annot=True, cmap='coolwarm', linewidths=0.5)
plt.title("Associations between Victim Descent and Top 5 Crime Types")
plt.show()

# Descent Code: 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
#As per the data it is visible that H Category people have a significant␣
 ↪contribution in the crimerate
```

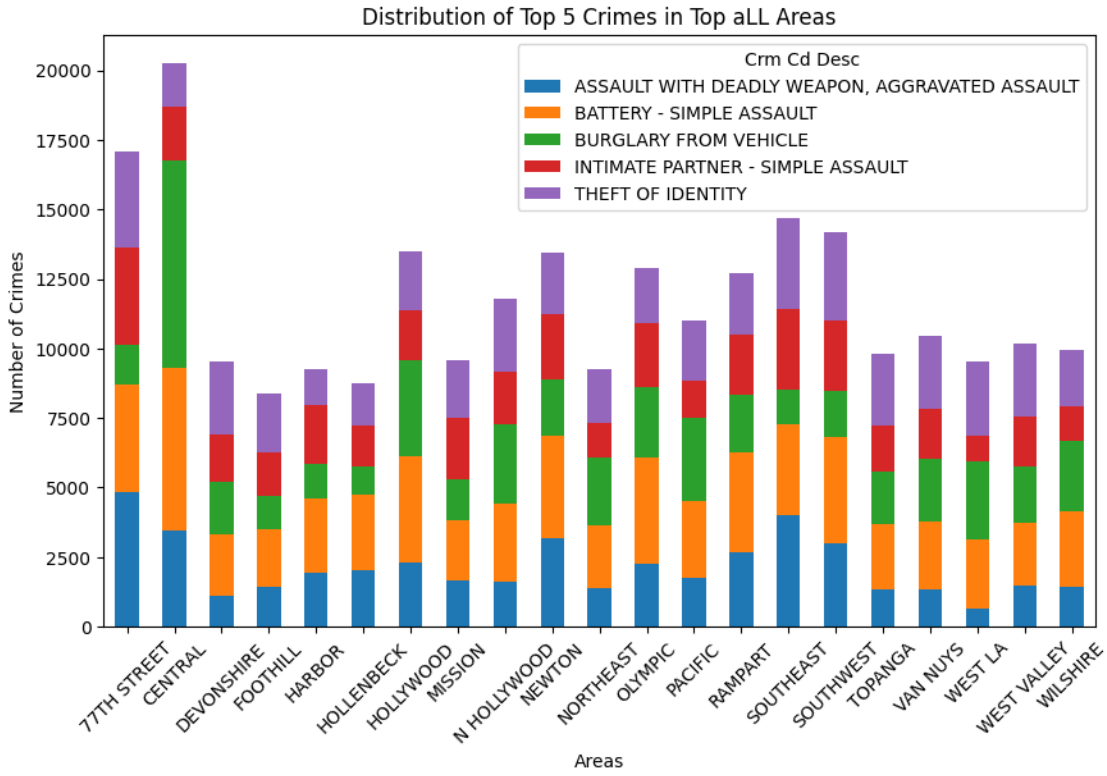
```
Victim Descent-Crime Chi-Squared Test p-value for Top 5 Crimes: 0.0
```



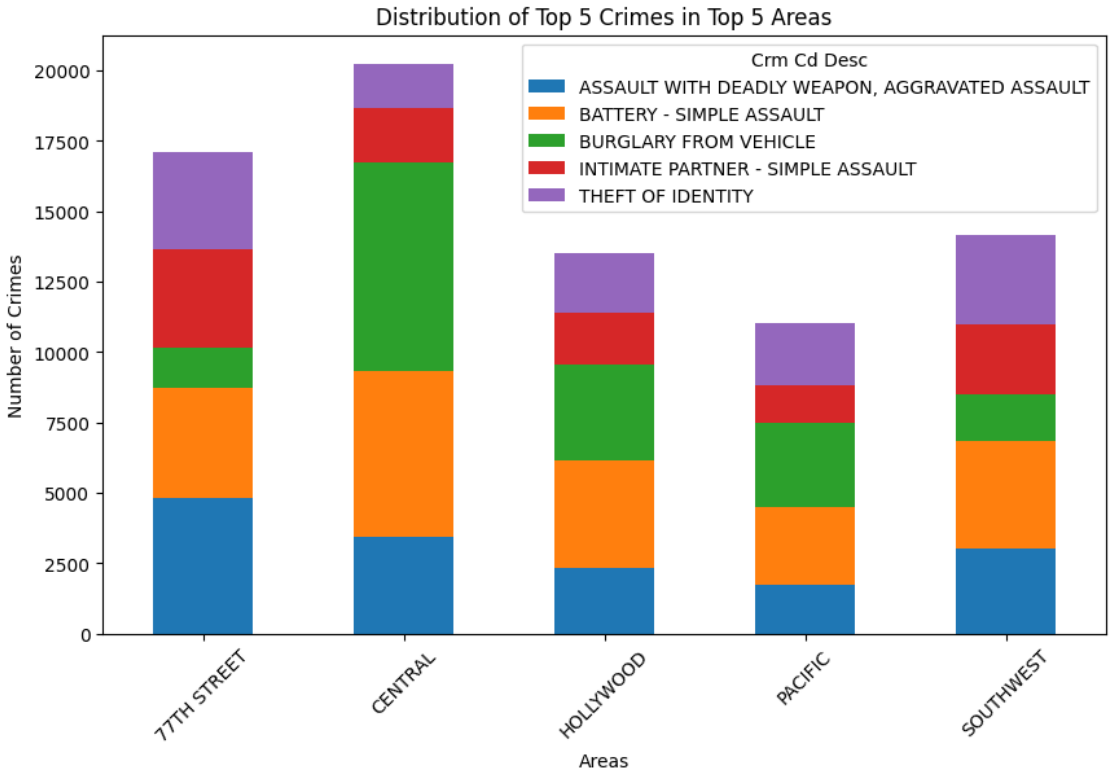
[152]:
```
# Find the top 5 areas with the most crimes
top_5_areas = df['AREA NAME'].value_counts().nlargest(5).index

# Filter the dataset to include only the top 5 crimes and the top 5 areas
df_top_5_crimes = df[df['Crm Cd Desc'].isin(top_5_crimes)]
df_top_5_areas = df[df['AREA NAME'].isin(top_5_areas)]

# Perform a chi-squared test for the association between crime types and areas
```

```
from scipy.stats import chi2_contingency

contingency_table = pd.crosstab(df_top_5_crimes['Crm Cd Desc'],␣
 ↪df_top_5_areas['AREA NAME'])
chi2, p, _, _ = chi2_contingency(contingency_table)

# Print or analyze the chi-squared test results
print("Crime-Area Chi-Squared Test p-value for Top 5 Crimes and Top 5 Areas:",␣
 ↪p)


# Create a pivot table to count the occurrences of each crime within each area
crime_area_pivot = pd.pivot_table(data=df_top_5_crimes, index='AREA NAME',␣
 ↪columns='Crm Cd Desc', aggfunc='size', fill_value=0)

# Create a stacked bar chart
fig, ax = plt.subplots(figsize=(10, 6))
crime_area_pivot.plot(kind='bar', stacked=True, ax=ax)

# Set labels and title
ax.set_xlabel("Areas")
ax.set_ylabel("Number of Crimes")
plt.title("Distribution of Top 5 Crimes in Top aLL Areas")

# Rotate x-axis labels for better readability
plt.xticks(rotation=45)

plt.show()
```

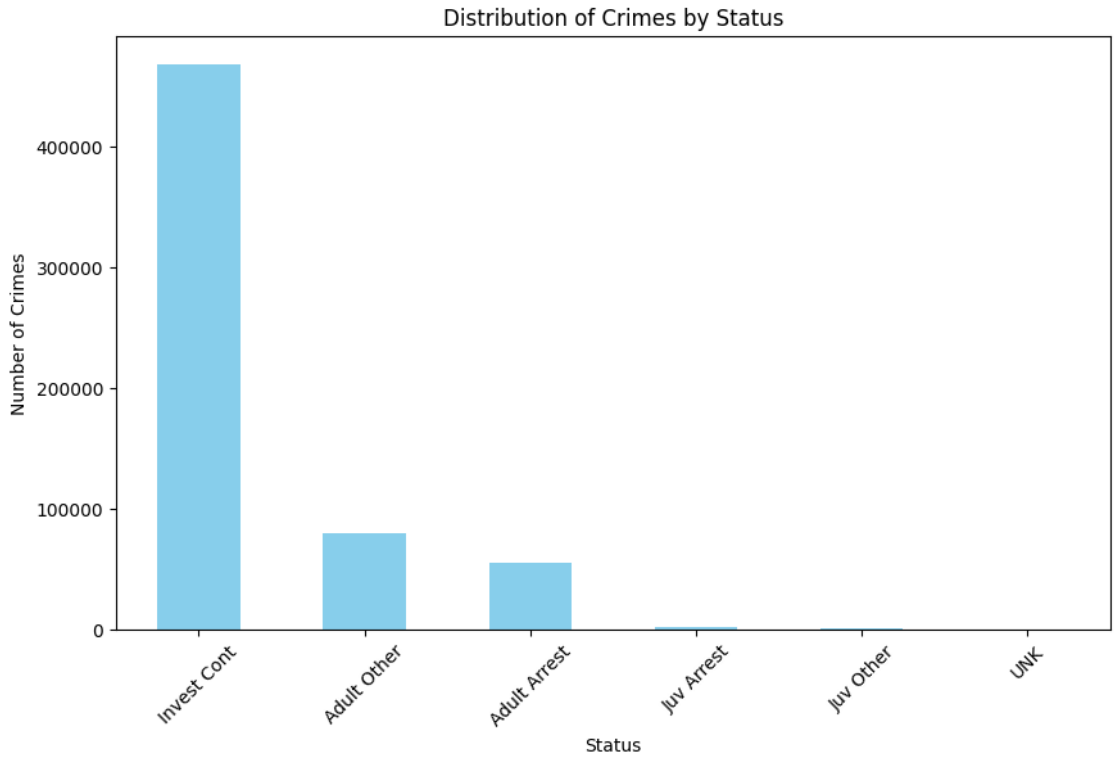
```
Crime-Area Chi-Squared Test p-value for Top 5 Crimes and Top 5 Areas: 0.0
```



[153]:
```
# Filter the dataset to include only the top 5 areas
top_5_areas = df_top_5_areas['AREA NAME'].value_counts().index[:5]
df_filtered = df_top_5_crimes[df_top_5_crimes['AREA NAME'].isin(top_5_areas)]

# Create a pivot table to count the occurrences of each crime within each area
crime_area_pivot = pd.pivot_table(data=df_filtered, index='AREA NAME',␣
 ↪columns='Crm Cd Desc', aggfunc='size', fill_value=0)

# Create a stacked bar chart
fig, ax = plt.subplots(figsize=(10, 6))
crime_area_pivot.plot(kind='bar', stacked=True, ax=ax)

# Set labels and title
ax.set_xlabel("Areas")
ax.set_ylabel("Number of Crimes")
plt.title("Distribution of Top 5 Crimes in Top 5 Areas")

# Rotate x-axis labels for better readability
plt.xticks(rotation=45)

plt.show()
```



[154]:
```
# Calculate the total number of crimes in each status category
crime_status_counts = df.groupby('Status Desc').size()

# Calculate the total number of crimes in each status category
crime_status_counts = df['Status Desc'].value_counts()

# Create a bar chart to visualize the distribution of crimes in different␣
 ↪status categories
plt.figure(figsize=(10, 6))
crime_status_counts.plot(kind='bar', color='skyblue')
plt.title("Distribution of Crimes by Status")
plt.xlabel("Status")
plt.ylabel("Number of Crimes")
plt.xticks(rotation=45)
plt.show()
```

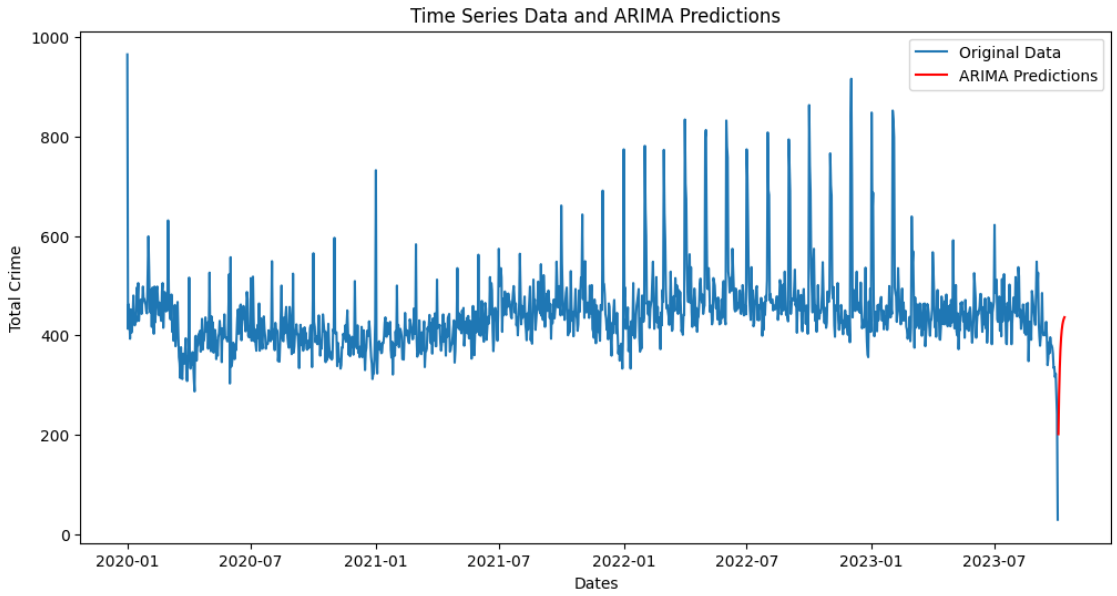


[155]:
```
#Question-10 (Rutuja)- Predicting future trends
```

[156]:
```
# Fit an ARIMA model to the example time series data
#changing hyper parameters
import pandas as pd
import matplotlib.pyplot as plt
from statsmodels.tsa.seasonal import seasonal_decompose
from statsmodels.tsa.stattools import adfuller
from statsmodels.tsa.arima.model import ARIMA
from statsmodels.tsa.statespace.sarimax import SARIMAX
model = ARIMA(df_date_data['to_summarize'], order=(1,0,1))
model_fit = model.fit()

# Make predictions
predictions = model_fit.forecast(steps=10)  # Predict the next 10 values

# Plot the original data and predictions
plt.figure(figsize=(12, 6))
plt.plot(df_date_data.index, df_date_data['to_summarize'], label='Original␣
 ↪Data')
plt.plot(pd.date_range(start='2023-10-03', periods=10, freq='D'), predictions,␣
 ↪color='red', label='ARIMA Predictions')
```

```
plt.xlabel('Dates')
plt.ylabel('Total Crime')
plt.title('Time Series Data and ARIMA Predictions')
plt.legend()
plt.show()
```

```
/usr/local/lib/python3.10/dist-packages/statsmodels/tsa/base/tsa_model.py:473:
ValueWarning: No frequency information was provided, so inferred frequency D
will be used.
  self._init_dates(dates, freq)
/usr/local/lib/python3.10/dist-packages/statsmodels/tsa/base/tsa_model.py:473:
ValueWarning: No frequency information was provided, so inferred frequency D
will be used.
  self._init_dates(dates, freq)
/usr/local/lib/python3.10/dist-packages/statsmodels/tsa/base/tsa_model.py:473:
ValueWarning: No frequency information was provided, so inferred frequency D
will be used.
  self._init_dates(dates, freq)
```



[157]:
```
predictions
```

[157]:
```
2023-10-03    201.149430
2023-10-04    281.387032
2023-10-05    334.971768
2023-10-06    370.757035
2023-10-07    394.655358
2023-10-08    410.615273
```

```
2023-10-09    421.273716
2023-10-10    428.391698
2023-10-11    433.145270
2023-10-12    436.319828
Freq: D, Name: predicted_mean, dtype: float64
```

[ ]: