Group9_FDA

October 16, 2024

IE6400 Foundations Data Analytics Engineering Fall Semester 2024

Group Projects

Topic: Cleaning and Analyzing Crime Data

Objective: In this project, you'll work with real-world data dataset containing crime data 2020. Your goal is to clean and prepare the dataset for analysis, perform exploratory data analysis (EDA), and answer specific questions related to crime trends, patterns, and factors influencing crime rates.

Dataset: You will use the crime dataset available at Crime Data from 2020 to present.

Tasks: - 1. Data Acquisition: Download the dataset from the provided link and load it into your preferred data analysis tool.

[1]:
```
import pandas as pd
cp = pd.read_csv(r"c:\ms\IE6400\Crimedate_till_sep29.csv")
print(cp)
```

```
            DR_NO               Date Rptd                DATE OCC  TIME OCC  \
0       190326475        03-01-2020 00:00        03-01-2020 00:00      2130
1       200106753        02-09-2020 00:00        02-08-2020 00:00      1800
2       200320258        11-11-2020 00:00        11-04-2020 00:00      1700
3       200907217        05-10-2023 00:00        03-10-2020 00:00      2037
4       220614831  08/18/2022 12:00:00 AM  08/17/2020 12:00:00 AM      1200
…             …                       …                       …       …
978623  240710284  07/24/2024 12:00:00 AM  07/23/2024 12:00:00 AM      1400
978624  240104953  01/15/2024 12:00:00 AM  01/15/2024 12:00:00 AM       100
978625  241711348  07/19/2024 12:00:00 AM  07/19/2024 12:00:00 AM       757
978626  240309674  04/24/2024 12:00:00 AM  04/24/2024 12:00:00 AM      1500
978627  240910892  08/13/2024 12:00:00 AM        08-12-2024 00:00      2300

        AREA   AREA NAME  Rpt Dist No  Part 1-2  Crm Cd  \
0          7    Wilshire          784         1     510
1          1     Central          182         1     330
2          3   Southwest          356         1     480
3          9    Van Nuys          964         1     343
4          6   Hollywood          666         2     354
…        …          …          …        …
978623     7    Wilshire          788         1     510
978624     1     Central          101         2     745
```

```
978625    17  Devonshire         1751         2     888
978626     3   Southwest          358         1     230
978627     9    Van Nuys          914         1     510

                                           Crm Cd Desc  … Status  \
0                                     VEHICLE - STOLEN  …     AA
1                                BURGLARY FROM VEHICLE  …     IC
2                                       BIKE - STOLEN  …     IC
3             SHOPLIFTING-GRAND THEFT ($950.01 & OVER)  …     IC
4                                    THEFT OF IDENTITY  …     IC
…                                                  …  …    …
978623                                VEHICLE - STOLEN  …     IC
978624        VANDALISM - MISDEAMEANOR ($399 OR UNDER)  …     IC
978625                                     TRESPASSING  …     IC
978626  ASSAULT WITH DEADLY WEAPON, AGGRAVATED ASSAULT  …     IC
978627                                VEHICLE - STOLEN  …     IC

         Status Desc Crm Cd 1 Crm Cd 2  Crm Cd 3 Crm Cd 4  \
0       Adult Arrest    510.0    998.0       NaN      NaN
1        Invest Cont    330.0    998.0       NaN      NaN
2        Invest Cont    480.0      NaN       NaN      NaN
3        Invest Cont    343.0      NaN       NaN      NaN
4        Invest Cont    354.0      NaN       NaN      NaN
…                …       …       …         …        …
978623   Invest Cont    510.0      NaN       NaN      NaN
978624   Invest Cont    745.0      NaN       NaN      NaN
978625   Invest Cont    888.0      NaN       NaN      NaN
978626   Invest Cont    230.0      NaN       NaN      NaN
978627   Invest Cont    510.0      NaN       NaN      NaN

                                     LOCATION  \
0        1900 S  LONGWOOD                  AV
1        1000 S  FLOWER                    ST
2        1400 W  37TH                      ST
3       14000    RIVERSIDE                 DR
4                              1900    TRANSIENT
…                                           …
978623   4000 W  23RD                      ST
978624   1300 W  SUNSET                    BL
978625  10000    OLD DEPOT PLAZA           RD
978626           FLOWER                    ST
978627   6900    VESPER                    AV

                          Cross Street      LAT       LON
0                                  NaN  34.0375 -118.3506
1                                  NaN  34.0444 -118.2628
2                                  NaN  34.0210 -118.3002
3                                  NaN  34.1576 -118.4387
```

```
4                                    NaN  34.0944 -118.3277
…                                  …        …        …
978623                               NaN  34.0362 -118.3284
978624                               NaN  34.0685 -118.2460
978625                               NaN  34.2500 -118.5990
978626   JEFFERSON                    BL  34.0215 -118.2868
978627                               NaN  34.1961 -118.4510

[978628 rows x 28 columns]
```

2. Data Inspection: - Display the first few rows of the dataset?

[2]:
```python
#Display the first 10 rows of the dataframe
cp.head(10)
```

[2]:
```
       DR_NO              Date Rptd                DATE OCC  TIME OCC  AREA  \
0  190326475        03-01-2020 00:00        03-01-2020 00:00      2130     7
1  200106753        02-09-2020 00:00        02-08-2020 00:00      1800     1
2  200320258        11-11-2020 00:00        11-04-2020 00:00      1700     3
3  200907217        05-10-2023 00:00        03-10-2020 00:00      2037     9
4  220614831  08/18/2022 12:00:00 AM  08/17/2020 12:00:00 AM      1200     6
5  231808869        04-04-2023 00:00        12-01-2020 00:00      2300    18
6  230110144        04-04-2023 00:00        07-03-2020 00:00       900     1
7  220314085  07/22/2022 12:00:00 AM        05-12-2020 00:00      1110     3
8  231309864  04/28/2023 12:00:00 AM        12-09-2020 00:00      1400    13
9  211904005  12/31/2020 12:00:00 AM  12/31/2020 12:00:00 AM      1220    19

   AREA NAME  Rpt Dist No  Part 1-2  Crm Cd  \
0   Wilshire          784         1     510
1    Central          182         1     330
2  Southwest          356         1     480
3   Van Nuys          964         1     343
4  Hollywood          666         2     354
5  Southeast         1826         2     354
6    Central          182         2     354
7  Southwest          303         2     354
8     Newton         1375         2     354
9    Mission         1974         2     624

                                Crm Cd Desc  … Status   Status Desc  \
0                          VEHICLE - STOLEN  …     AA  Adult Arrest
1                     BURGLARY FROM VEHICLE  …     IC   Invest Cont
2                             BIKE - STOLEN  …     IC   Invest Cont
3  SHOPLIFTING-GRAND THEFT ($950.01 & OVER)  …     IC   Invest Cont
4                         THEFT OF IDENTITY  …     IC   Invest Cont
5                         THEFT OF IDENTITY  …     IC   Invest Cont
6                         THEFT OF IDENTITY  …     IC   Invest Cont
7                         THEFT OF IDENTITY  …     IC   Invest Cont
```

```
8                        THEFT OF IDENTITY  …     IC   Invest Cont
9                 BATTERY - SIMPLE ASSAULT  …     IC   Invest Cont

  Crm Cd 1 Crm Cd 2  Crm Cd 3 Crm Cd 4  \
0    510.0    998.0       NaN      NaN
1    330.0    998.0       NaN      NaN
2    480.0      NaN       NaN      NaN
3    343.0      NaN       NaN      NaN
4    354.0      NaN       NaN      NaN
5    354.0      NaN       NaN      NaN
6    354.0      NaN       NaN      NaN
7    354.0      NaN       NaN      NaN
8    354.0      NaN       NaN      NaN
9    624.0      NaN       NaN      NaN

                                 LOCATION Cross Street      LAT       LON
0   1900 S  LONGWOOD                   AV          NaN  34.0375 -118.3506
1   1000 S  FLOWER                     ST          NaN  34.0444 -118.2628
2   1400 W  37TH                       ST          NaN  34.0210 -118.3002
3  14000    RIVERSIDE                  DR          NaN  34.1576 -118.4387
4                         1900    TRANSIENT          NaN  34.0944 -118.3277
5   9900    COMPTON                    AV          NaN  33.9467 -118.2463
6   1100 S  GRAND                      AV          NaN  34.0415 -118.2620
7   2500 S  SYCAMORE                   AV          NaN  34.0335 -118.3537
8   1300 E  57TH                       ST          NaN  33.9911 -118.2521
9   9000    CEDROS                     AV          NaN  34.2336 -118.4535

[10 rows x 28 columns]
```

- Check the data types of each column?

[3]:
```python
#Display the datatypes of each column in the dataframe
cp.dtypes
```

[3]:
```
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
```

- Review column names and descriptions

[4]:
```
#Display summary of the dataframe including column names, non-null counts, and␣
↪data types
cp.describe()
```

[4]:
```
              DR_NO       TIME OCC           AREA    Rpt Dist No  \
count  9.786280e+05  978628.000000  978628.000000  978628.000000
mean   2.196564e+08    1338.802627      10.702561    1116.686084
std    1.290395e+07     651.622947       6.107280     610.836054
min    8.170000e+02       1.000000       1.000000     101.000000
25%    2.106073e+08     900.000000       5.000000     589.000000
50%    2.208116e+08    1420.000000      11.000000    1141.000000
75%    2.309110e+08    1900.000000      16.000000    1617.000000
max    2.499253e+08    2359.000000      21.000000    2199.000000

            Part 1-2         Crm Cd       Vict Age      Premis Cd  \
count  978628.000000  978628.000000  978628.000000  978613.000000
mean        1.404785     500.810635      29.122904     306.181502
std         0.490851     206.309796      21.961531     218.908131
min         1.000000     110.000000      -4.000000     101.000000
25%         1.000000     331.000000       0.000000     101.000000
50%         1.000000     442.000000      30.000000     203.000000
75%         2.000000     626.000000      44.000000     501.000000
max         2.000000     956.000000     120.000000     976.000000

       Weapon Used Cd       Crm Cd 1      Crm Cd 2     Crm Cd 3   Crm Cd 4  \
count   325959.000000  978617.000000  68816.000000  2309.000000   64.00000
mean       363.815372     500.564847    958.156344   984.192724  991.21875
std        123.673988     206.107451    110.251477    51.506344   27.06985
min        101.000000     110.000000    210.000000   310.000000  821.00000
25%        311.000000     331.000000    998.000000   998.000000  998.00000
```

```
50%        400.000000     442.000000     998.000000    998.000000   998.00000
75%        400.000000     626.000000     998.000000    998.000000   998.00000
max        516.000000     956.000000     999.000000    999.000000   999.00000

                 LAT            LON
count  978628.000000  978628.000000
mean       33.995399    -118.081108
std         1.640056       5.684520
min         0.000000    -118.667600
25%        34.014600    -118.430500
50%        34.058900    -118.322500
75%        34.164900    -118.273900
max        34.334300       0.000000
```

3.Data Cleaning: - Identify and handle missing data appropriately

[5]:
```
#Check for null values in the dataframe
cp.isnull()
```

[5]:
```
        DR_NO  Date Rptd  DATE OCC  TIME OCC   AREA  AREA NAME  Rpt Dist No  \
0       False      False     False     False  False      False        False
1       False      False     False     False  False      False        False
2       False      False     False     False  False      False        False
3       False      False     False     False  False      False        False
4       False      False     False     False  False      False        False
…         …         …        …         …      …        …          …
978623  False      False     False     False  False      False        False
978624  False      False     False     False  False      False        False
978625  False      False     False     False  False      False        False
978626  False      False     False     False  False      False        False
978627  False      False     False     False  False      False        False

        Part 1-2  Crm Cd  Crm Cd Desc  …  Status  Status Desc  Crm Cd 1  \
0          False   False        False  …   False        False     False
1          False   False        False  …   False        False     False
2          False   False        False  …   False        False     False
3          False   False        False  …   False        False     False
4          False   False        False  …   False        False     False
…            …       …            …  …     …            …         …
978623     False   False        False  …   False        False     False
978624     False   False        False  …   False        False     False
978625     False   False        False  …   False        False     False
978626     False   False        False  …   False        False     False
978627     False   False        False  …   False        False     False

        Crm Cd 2  Crm Cd 3  Crm Cd 4  LOCATION  Cross Street    LAT    LON
0          False      True      True     False          True  False  False
1          False      True      True     False          True  False  False
```

```
2          True    True    True    False        True  False  False
3          True    True    True    False        True  False  False
4          True    True    True    False        True  False  False
…          …       …       …       …        …    …    …
978623     True    True    True    False        True  False  False
978624     True    True    True    False        True  False  False
978625     True    True    True    False        True  False  False
978626     True    True    True    False       False  False  False
978627     True    True    True    False        True  False  False

[978628 rows x 28 columns]
```

[6]:

```
#Drop rows with null values from the dataframe (handling missing data)
cp.dropna()
```

[6]:

```
            DR_NO               Date Rptd                DATE OCC  TIME OCC  \
66026   201904032        01-02-2020 00:00        01-01-2020 00:00      2135
86496   200613424        08-02-2020 00:00        08-02-2020 00:00      2030
363643  210617136        10-08-2021 00:00        10-07-2021 00:00      1950
372408  210209196        05-08-2021 00:00        05-08-2021 00:00       230
489920  220600626  04/27/2022 12:00:00 AM  04/23/2022 12:00:00 AM      2300
537636  221718232  12/25/2022 12:00:00 AM  12/25/2022 12:00:00 AM      1150
585780  221401314        11-10-2022 00:00        11-10-2022 00:00      2117
728192  231717599  11/15/2023 12:00:00 AM  11/15/2023 12:00:00 AM       400
809005  231915572  10/21/2023 12:00:00 AM  10/21/2023 12:00:00 AM         1
922659  241905348        02-04-2024 00:00        02-03-2024 00:00      1100

        AREA   AREA NAME  Rpt Dist No  Part 1-2  Crm Cd  \
66026     19     Mission         1924         1     761
86496      6   Hollywood          657         1     761
363643     6   Hollywood          659         1     121
372408     2     Rampart          279         1     210
489920     6   Hollywood          646         1     821
537636    17  Devonshire         1797         1     122
585780    14     Pacific         1452         2     910
728192    17  Devonshire         1738         1     210
809005    19     Mission         1902         1     210
922659    19     Mission         1983         1     820

                                             Crm Cd Desc  … Status  \
66026                                     BRANDISH WEAPON  …     AA
86496                                     BRANDISH WEAPON  …     AO
363643                                     RAPE, FORCIBLE  …     IC
372408                                            ROBBERY  …     AO
489920  SODOMY/SEXUAL CONTACT B/W PENIS OF ONE PERS TO…  …     IC
537636                                     RAPE, ATTEMPTED  …     AA
585780                                         KIDNAPPING  …     IC
```

```
728192                                             ROBBERY  …     IC
809005                                             ROBBERY  …     AA
922659                                     ORAL COPULATION  …     AO

          Status Desc Crm Cd 1 Crm Cd 2  Crm Cd 3 Crm Cd 4  \
66026   Adult Arrest    761.0    930.0     997.0    998.0
86496    Adult Other    761.0    920.0     930.0    998.0
363643   Invest Cont    121.0    210.0     910.0    998.0
372408   Adult Other    210.0    510.0     910.0    998.0
489920   Invest Cont    230.0    821.0     910.0    998.0
537636  Adult Arrest    122.0    230.0     910.0    998.0
585780   Invest Cont    812.0    860.0     910.0    998.0
728192   Invest Cont    210.0    230.0     761.0    998.0
809005  Adult Arrest    210.0    250.0     761.0    998.0
922659   Adult Other    761.0    820.0     910.0    998.0

                                LOCATION                     Cross Street  \
66026   ASTORIA                       ST  SAN FERNANDO                 RD
86496                            WESTERN                          ROMAINE
363643                         NORMANDIE                       DE LONGPRE
372408                      JAMES M WOOD                            GREEN
489920                             SELMA                       LAS PALMAS
537636  PARTHENIA                     ST                       HAYVENHURST
585780                        WASHINGTON                         SPEEDWAY
728192  HASKELL                       AV  SAN FERNANDO                 BL
809005                              POLK                           BORDEN
922659                            BURNET                        PARTHENIA

            LAT       LON
66026   34.2949 -118.4571
86496   34.0885 -118.3092
363643  34.0966 -118.3005
372408  34.0503 -118.2720
489920  34.0997 -118.3363
537636  34.2285 -118.4939
585780  33.9792 -118.4666
728192  34.2692 -118.4789
809005  34.3103 -118.4467
922659  34.2282 -118.4633

[10 rows x 28 columns]
```

- Check for and remove dupicate rows

[12]:
```
cp.drop_duplicates().head()
```

```
[12]:        DR_NO               Date Rptd                DATE OCC  TIME OCC  AREA  \
0  190326475        03-01-2020 00:00        03-01-2020 00:00    2130.0     7
1  200106753        02-09-2020 00:00        02-08-2020 00:00    1800.0     1
2  200320258        11-11-2020 00:00        11-04-2020 00:00    1700.0     3
3  200907217        05-10-2023 00:00        03-10-2020 00:00    2037.0     9
4  220614831  08/18/2022 12:00:00 AM  08/17/2020 12:00:00 AM    1200.0     6

   AREA NAME  Rpt Dist No  Part 1-2  Crm Cd  \
0   Wilshire        784.0         1   510.0
1    Central        182.0         1   330.0
2  Southwest        356.0         1   480.0
3   Van Nuys        964.0         1   343.0
4  Hollywood        666.0         2   354.0

                              Crm Cd Desc  … Status   Status Desc  \
0                        VEHICLE - STOLEN  …     AA  Adult Arrest
1                   BURGLARY FROM VEHICLE  …     IC   Invest Cont
2                           BIKE - STOLEN  …     IC   Invest Cont
3  SHOPLIFTING-GRAND THEFT ($950.01 & OVER)  …     IC   Invest Cont
4                       THEFT OF IDENTITY  …     IC   Invest Cont

  Crm Cd 1 Crm Cd 2  Crm Cd 3 Crm Cd 4  \
0    510.0    998.0       NaN      NaN
1    330.0    998.0       NaN      NaN
2    480.0      NaN       NaN      NaN
3    343.0      NaN       NaN      NaN
4    354.0      NaN       NaN      NaN

                                  LOCATION Cross Street      LAT       LON
0   1900 S  LONGWOOD                    AV          NaN  34.0375 -118.3506
1   1000 S  FLOWER                      ST          NaN  34.0444 -118.2628
2   1400 W  37TH                        ST          NaN  34.0210 -118.3002
3  14000    RIVERSIDE                   DR          NaN  34.1576 -118.4387
4                       1900    TRANSIENT          NaN  34.0944 -118.3277

[5 rows x 28 columns]
```

- Convert data types if needed (e.g., dates to date format, numerical values to appropriate numeric types).

```
[13]: import pandas as pd
from datetime import datetime
# Sample data creation for demonstration (replace this with your actual data␣
↪loading)
cp = pd.read_csv(r"c:\ms\IE6400\Crimedate_till_sep29.csv")
# Convert the 'Date Rptd' column to datetime format, coercing invalid entries␣
↪to NaT
```

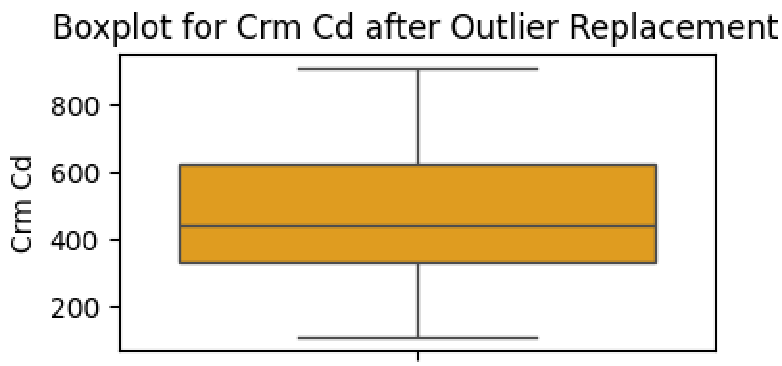
```
cp['Date Rptd'] = pd.to_datetime(cp['Date Rptd'], errors='coerce')
# Convert the 'DATE OCC' column to datetime format, coercing invalid entries to␣
↪NaT
cp['DATE OCC'] = pd.to_datetime(cp['DATE OCC'], errors='coerce')
# Function to process and convert 'TIME OCC' into HH:MM format
def process_time(x):
    try:
        # Ensure the value is treated as a string, fill with leading zeros if␣
↪necessary
        time_str = str(int(x)).zfill(4)  # Convert to string and pad
        # Convert the string to time format
        return (datetime.strptime(time_str, "%H%M").time()).strftime("%H:%M")
    except (ValueError, TypeError):
        # Return a default value if conversion fails
        return '00:00'
# Apply the function to 'TIME OCC'
cp['TIME OCC'] = cp['TIME OCC'].apply(process_time)
# Display the updated columns to verify conversion
print("Converted Dates and Times:")
print(cp[['Date Rptd', 'DATE OCC', 'TIME OCC']].head())# Display the first 5␣
↪rows
print(cp[['Date Rptd', 'DATE OCC', 'TIME OCC']].tail())# Display the last 5 rows
```

```
Converted Dates and Times:
   Date Rptd   DATE OCC TIME OCC
0 2020-03-01 2020-03-01    21:30
1 2020-02-09 2020-02-08    18:00
2 2020-11-11 2020-11-04    17:00
3 2023-05-10 2020-03-10    20:37
4        NaT        NaT    12:00
       Date Rptd   DATE OCC TIME OCC
978623       NaT        NaT    14:00
978624       NaT        NaT    01:00
978625       NaT        NaT    07:57
978626       NaT        NaT    15:00
978627       NaT 2024-08-12    23:00
```

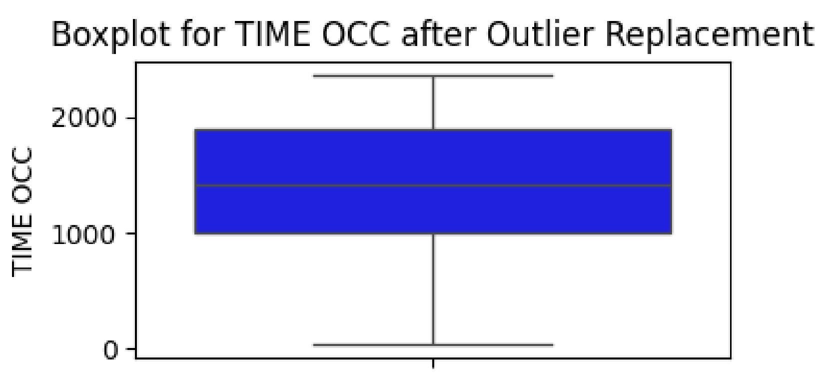
- Deal with outliers if relevant to your analysis.

[9]:
```
import numpy as np
from scipy.stats import zscore
import seaborn as sns
import matplotlib.pyplot as plt
import pandas as pd
cp = pd.read_csv(r"c:\ms\IE6400\Crimedate_till_sep29.csv")

# Z-score outlier detection and replacement for 'Crm Cd'
z_scores_crm_cd = zscore(cp['Crm Cd'])
```

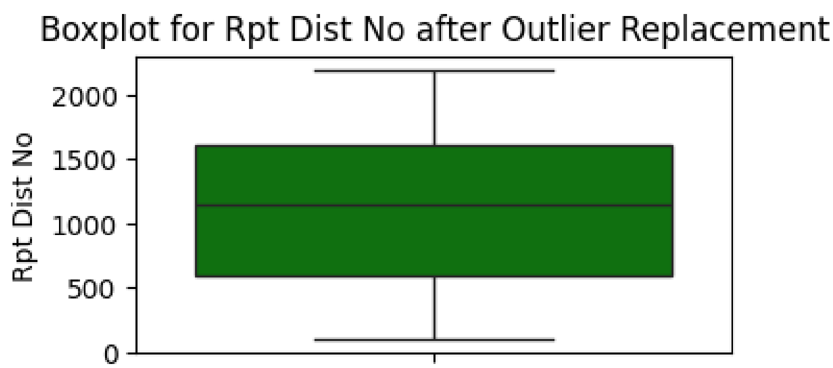
```
outliers_crm_cd = (np.abs(z_scores_crm_cd) > 2)
cp['Crm Cd'] = np.where(outliers_crm_cd, cp['Crm Cd'].median(), cp['Crm Cd'])

# Plot boxplot for 'Crm Cd'
plt.figure(figsize=(4, 2))
sns.boxplot(data=cp['Crm Cd'], color='orange')
plt.title('Boxplot for Crm Cd after Outlier Replacement')
plt.show()
```



[10]:
```
# Z-score outlier detection and replacement for 'TIME OCC'
z_scores_time_occ = zscore(cp['TIME OCC'])
outliers_time_occ = (np.abs(z_scores_time_occ) > 2)
cp['TIME OCC'] = np.where(outliers_time_occ, cp['TIME OCC'].median(), cp['TIME
 ↪OCC'])

# Plot boxplot for 'TIME OCC'
plt.figure(figsize=(4, 2))
sns.boxplot(data=cp['TIME OCC'], color='blue')
plt.title('Boxplot for TIME OCC after Outlier Replacement')
plt.show()
```



[11]:
```
# Z-score outlier detection and replacement for 'Rpt Dist No'
z_scores_rpt_dist_no = zscore(cp['Rpt Dist No'])
outliers_rpt_dist_no = (np.abs(z_scores_rpt_dist_no) > 2)
cp['Rpt Dist No'] = np.where(outliers_rpt_dist_no, cp['Rpt Dist No'].median(),␣
↪cp['Rpt Dist No'])

# Plot boxplot for 'Rpt Dist No'
plt.figure(figsize=(4, 2))
sns.boxplot(data=cp['Rpt Dist No'], color='green')
plt.title('Boxplot for Rpt Dist No after Outlier Replacement')
plt.show()
```



- Standardize or normalize numerical data as necessary.

[10]:
```
import pandas as pd
from sklearn.preprocessing import StandardScaler

# Load the dataset
cp = pd.read_csv(r"c:\ms\IE6400\Crimedate_till_sep29.csv")

# Select only the numerical columns for standardization
numerical_columns = cp.select_dtypes(include=['float64', 'int64']).columns

# Clean the dataset to remove NaN values before standardization
cp_cleaned = cp.dropna(subset=numerical_columns)

# Standardization: Scale the numerical data to have a mean of 0 and std of 1
scaler = StandardScaler()
cp_standardized = cp_cleaned.copy()  # Make a copy to preserve the original data
cp_standardized[numerical_columns] = scaler.
↪fit_transform(cp_cleaned[numerical_columns])
```

```
print("Standardized Data:")
print(cp_standardized[numerical_columns].head())
```

```
Standardized Data:
          DR_NO  TIME OCC      AREA  Rpt Dist No  Part 1-2    Crm Cd  \
2198  -1.282787 -1.623306 -0.983972    -0.954631 -0.388514  0.836245
4127  -1.166433  1.274569  1.356730     1.277183 -0.388514  1.040095
36672 -1.283900  1.303119 -0.983972    -0.937938 -0.388514  0.836245
37652 -1.284359 -2.037288 -0.983972    -0.996363 -0.388514  0.836245
39344 -1.283625 -1.723233 -0.983972    -0.969654 -0.388514  0.836245

       Vict Age  Premis Cd  Weapon Used Cd  Crm Cd 1  Crm Cd 2  Crm Cd 3  \
2198   0.924495   0.536197       -1.238294  1.223923  0.462166  0.646686
4127  -0.409332  -0.326574        0.869554 -0.883803  0.714340  0.403706
36672 -0.462685   0.536197       -1.238294  1.223923  0.462166  0.646686
37652 -0.409332   0.536197        1.654691  0.730956  0.462166  0.646686
39344  0.070846   0.531905       -0.538036  0.730956  0.462166  0.646686

       Crm Cd 4       LAT       LON
2198   0.233138 -0.370817  0.662651
4127   0.233138 -0.510100  0.786173
36672  0.233138 -0.494117  0.772449
37652  0.233138 -0.325150  0.173553
39344  0.233138 -0.360542  0.793659
```

[11]:

```
import pandas as pd
import numpy as np
from sklearn.preprocessing import MinMaxScaler

# Load the dataset
cp = pd.read_csv(r"c:\ms\IE6400\Crimedate_till_sep29.csv")

# Step to identify numerical columns for normalization
numerical_columns = cp.select_dtypes(include=[np.number]).columns.tolist()

# Clean the dataset to remove NaN values before normalization
cp_cleaned = cp.dropna(subset=numerical_columns)

# Normalization: Scale the numerical data to [0, 1]
normalizer = MinMaxScaler()
cp_normalized = cp_cleaned.copy()  # Make a copy to preserve the original data
cp_normalized[numerical_columns] = normalizer.
 ↪fit_transform(cp_cleaned[numerical_columns])

print("Normalized Data:")
print(cp_normalized[numerical_columns].head())
```

```
Normalized Data:
```

```
          DR_NO  TIME OCC      AREA  Rpt Dist No  Part 1-2   Crm Cd  Vict Age  \
2198   0.000455  0.127288  0.210526     0.208691       0.0  0.81375  0.645570
4127   0.034124  0.991486  0.947368     0.925966       0.0  0.88750  0.329114
36672  0.000133  1.000000  0.210526     0.214056       0.0  0.81375  0.316456
37652  0.000000  0.003831  0.210526     0.195279       0.0  0.81375  0.329114
39344  0.000212  0.097488  0.210526     0.203863       0.0  0.81375  0.443038

       Premis Cd  Weapon Used Cd  Crm Cd 1  Crm Cd 2  Crm Cd 3  Crm Cd 4  \
2198    0.496287        0.000000  0.915954  0.748641  0.817204  0.994382
4127    0.247525        0.726829  0.142450  0.828804  0.763441  0.994382
36672   0.496287        0.000000  0.915954  0.748641  0.817204  0.994382
37652   0.496287        0.997561  0.735043  0.748641  0.817204  0.994382
39344   0.495050        0.241463  0.735043  0.748641  0.817204  0.994382

            LAT       LON
2198   0.405982  0.743699
4127   0.372196  0.767237
36672  0.376073  0.764622
37652  0.417059  0.650499
39344  0.408474  0.768664
```

- Encode categorical data if present

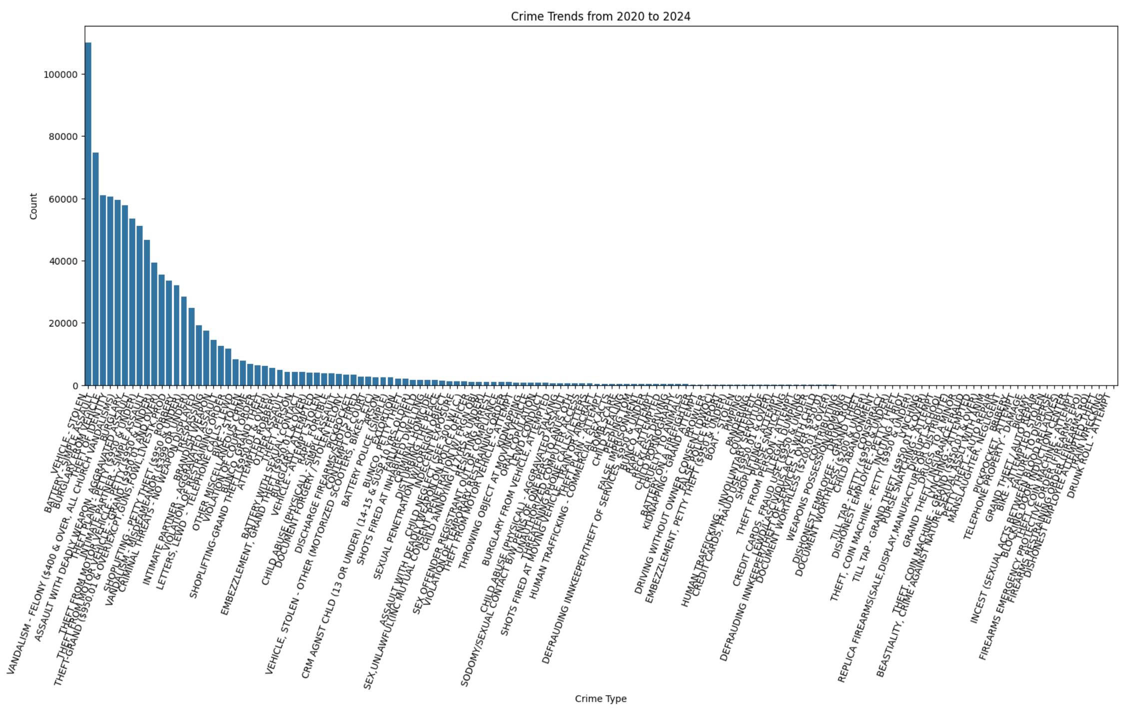
[12]:
```
#Import labelencoder from sklearn's preprocessing module
from sklearn.preprocessing import LabelEncoder
#Create an instance of labelEncoder
label_encoder = LabelEncoder()
#Transform the 'AREA NAME' column into numerical format
#Each unique category in 'AREA NAME' will be assigned a unique integer
cp['AREA NAME'] = label_encoder.fit_transform(cp['AREA NAME'])
#Display the first 10 rows of the dataframe to see the change in 'AREA NAME'␣
↪column
cp.head(10)
```

[12]:
```
       DR_NO              Date Rptd                DATE OCC  TIME OCC  AREA  \
0  190326475       03-01-2020 00:00        03-01-2020 00:00      2130     7
1  200106753       02-09-2020 00:00        02-08-2020 00:00      1800     1
2  200320258       11-11-2020 00:00        11-04-2020 00:00      1700     3
3  200907217       05-10-2023 00:00        03-10-2020 00:00      2037     9
4  220614831  08/18/2022 12:00:00 AM  08/17/2020 12:00:00 AM      1200     6
5  231808869       04-04-2023 00:00        12-01-2020 00:00      2300    18
6  230110144       04-04-2023 00:00        07-03-2020 00:00       900     1
7  220314085  07/22/2022 12:00:00 AM        05-12-2020 00:00      1110     3
8  231309864  04/28/2023 12:00:00 AM        12-09-2020 00:00      1400    13
9  211904005  12/31/2020 12:00:00 AM  12/31/2020 12:00:00 AM      1220    19

   AREA NAME  Rpt Dist No  Part 1-2  Crm Cd  \
0         20          784         1     510
```

```
1         1         182        1    330
2        15         356        1    480
3        17         964        1    343
4         6         666        2    354
5        14        1826        2    354
6         1         182        2    354
7        15         303        2    354
8         9        1375        2    354
9         7        1974        2    624

                              Crm Cd Desc  … Status   Status Desc  \
0                        VEHICLE - STOLEN  …     AA  Adult Arrest
1                   BURGLARY FROM VEHICLE  …     IC   Invest Cont
2                          BIKE - STOLEN  …     IC   Invest Cont
3  SHOPLIFTING-GRAND THEFT ($950.01 & OVER)  …     IC   Invest Cont
4                       THEFT OF IDENTITY  …     IC   Invest Cont
5                       THEFT OF IDENTITY  …     IC   Invest Cont
6                       THEFT OF IDENTITY  …     IC   Invest Cont
7                       THEFT OF IDENTITY  …     IC   Invest Cont
8                       THEFT OF IDENTITY  …     IC   Invest Cont
9                  BATTERY - SIMPLE ASSAULT  …     IC   Invest Cont

  Crm Cd 1 Crm Cd 2  Crm Cd 3 Crm Cd 4  \
0    510.0    998.0       NaN      NaN
1    330.0    998.0       NaN      NaN
2    480.0      NaN       NaN      NaN
3    343.0      NaN       NaN      NaN
4    354.0      NaN       NaN      NaN
5    354.0      NaN       NaN      NaN
6    354.0      NaN       NaN      NaN
7    354.0      NaN       NaN      NaN
8    354.0      NaN       NaN      NaN
9    624.0      NaN       NaN      NaN

                                   LOCATION Cross Street      LAT       LON
0    1900 S  LONGWOOD                    AV          NaN  34.0375 -118.3506
1    1000 S  FLOWER                      ST          NaN  34.0444 -118.2628
2    1400 W  37TH                        ST          NaN  34.0210 -118.3002
3   14000    RIVERSIDE                   DR          NaN  34.1576 -118.4387
4                         1900    TRANSIENT          NaN  34.0944 -118.3277
5    9900    COMPTON                     AV          NaN  33.9467 -118.2463
6    1100 S  GRAND                       AV          NaN  34.0415 -118.2620
7    2500 S  SYCAMORE                    AV          NaN  34.0335 -118.3537
8    1300 E  57TH                        ST          NaN  33.9911 -118.2521
9    9000    CEDROS                      AV          NaN  34.2336 -118.4535

[10 rows x 28 columns]
```

4. Exploratory Data Analysis (EDA): - Visualize overall crime trends from 2020 to present year.

[13]:
```
#Import seaborn and matplotlib for data visualization
import seaborn as sns
import matplotlib.pyplot as plt
#Set the figure size for plot
plt.figure(figsize=(20,7))
#Create a count plot for the 'Crm Cd Desc' column, ordering the bars by count
sns.countplot(data=cp, x='Crm Cd Desc', order=cp['Crm Cd Desc'].value_counts().
 ↪index)
plt.title('Crime Trends from 2020 to 2024') #Set the title for the plot
plt.xlabel('Crime Type') #Set the x-label for the plot
plt.ylabel('Count') #Set the y-label for the plot
plt.xticks(rotation=70, ha='right') #Rotate the x-axis labels for readability
plt.show() #Display the plot
```

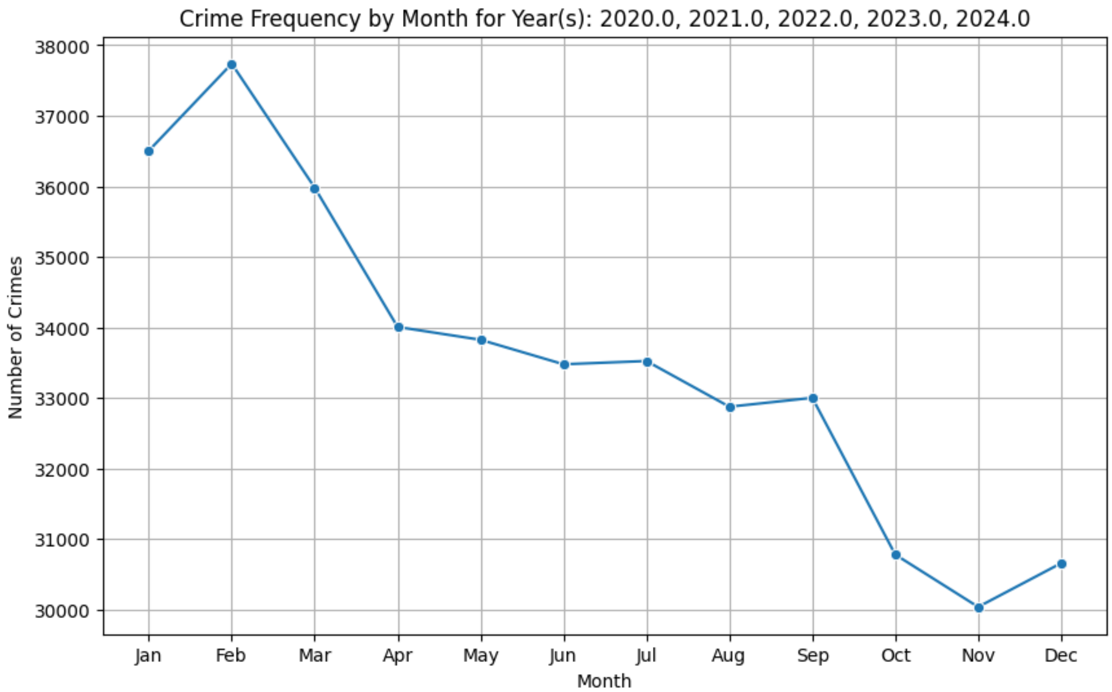


- Analyze and visualize seasonal patterns in crime data

[26]:
```
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
cp = pd.read_csv(r"c:\ms\IE6400\Crimedate_till_sep29.csv")
# Convert the 'DATE OCC' column to datetime format, coercing invalid entries to␣
 ↪NaT
```

```python
cp['DATE OCC'] = pd.to_datetime(cp['DATE OCC'], errors='coerce')

# Extract the month and year from 'DATE OCC' and create new columns
cp['Month'] = cp['DATE OCC'].dt.month
cp['Year'] = cp['DATE OCC'].dt.year

# Drop rows where 'DATE OCC' is NaT
cp = cp.dropna(subset=['DATE OCC'])

# Group the dataframe by 'Month' and count the number of crimes per month
crimes_per_month = cp.groupby('Month').size()

# Print the year(s) of data being plotted
unique_years = cp['Year'].unique()
print(f"The data contains crimes from the following year(s): {unique_years}")

# Plot the number of crimes per month
plt.figure(figsize=(10, 6))
sns.lineplot(x=crimes_per_month.index, y=crimes_per_month.values, marker='o')

# Set the title and labels for the plot
plt.title(f'Crime Frequency by Month for Year(s): {", ".join(map(str,␣
↪unique_years))}')
plt.xlabel('Month')
plt.ylabel('Number of Crimes')

# Set the X-ticks to display month names
plt.xticks(ticks=range(1, 13), labels=['Jan', 'Feb', 'Mar', 'Apr', 'May',␣
↪'Jun', 'Jul', 'Aug', 'Sep', 'Oct', 'Nov', 'Dec'])

plt.grid(True) # Enable grid lines for better readability
plt.show() # Display the plot
```

```
The data contains crimes from the following year(s): [2020. 2021. 2022. 2023.
2024.]
```

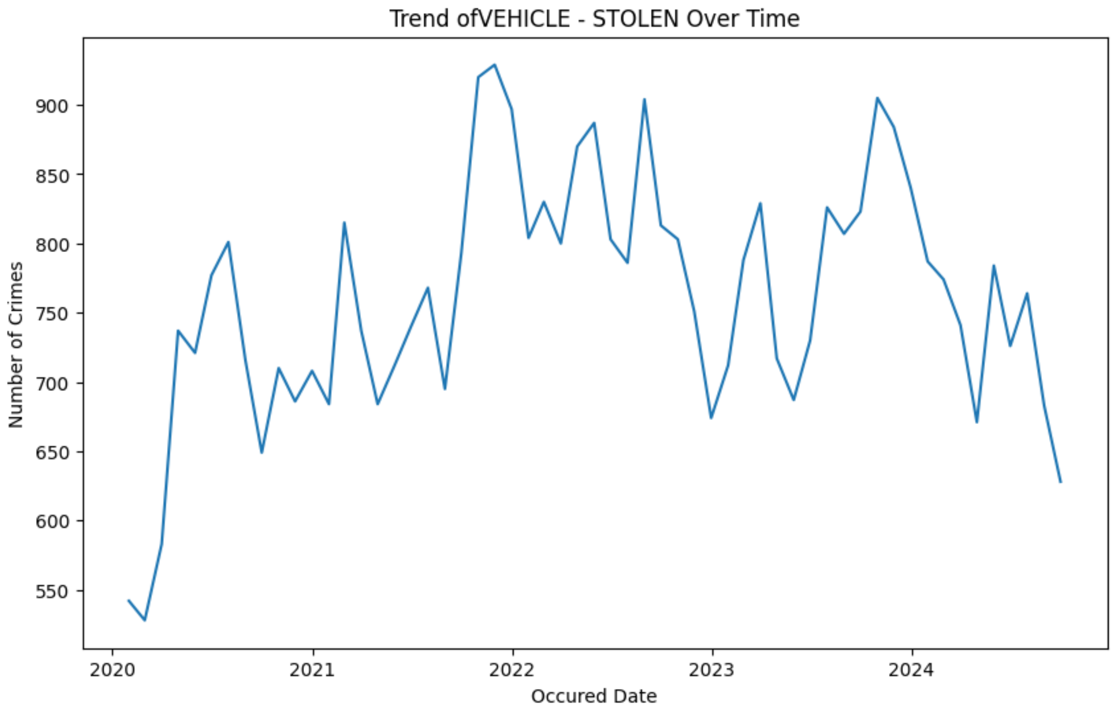


- Identify the most common type of crime and its trends over time.

[15]:
```
import warnings # Import the warnings module to handle warning messages
# Suppress FutureWarning messages
warnings.simplefilter(action='ignore', category=FutureWarning)
# Identify the most common crime type from 'Crm Cd Desc' column
common_crime = cp['Crm Cd Desc'].value_counts().idxmax() # Display the most␣
↪common crime
print(f"The most common crime is:{common_crime}")
# Filter the Dataframe to get data related to the most common crime
common_crime_data = cp[cp['Crm Cd Desc'] == common_crime]
# Convert the 'DATE OCC' column to datetime format, coercing invalid entries to␣
↪NaT
common_crime_data.loc[:, 'DATE OCC'] = pd.to_datetime(common_crime_data['DATE␣
↪OCC'], errors='coerce')
# Resample the data to get the count of crimes per month, ME stands month-end␣
↪frequency
crime_trends = common_crime_data.resample('ME', on='DATE OCC').size()
plt.figure(figsize=(10,6)) # set the figure size for plot
# Create a line plot for the trend of the most common crime over time
sns.lineplot(x=crime_trends.index, y=crime_trends.values)
#set the title and lables for the plot
plt.title(f"Trend of{  common_crime} Over Time")
plt.xlabel('Occured Date')
```

```
plt.ylabel('Number of Crimes')
plt.show() # Display the plot
```

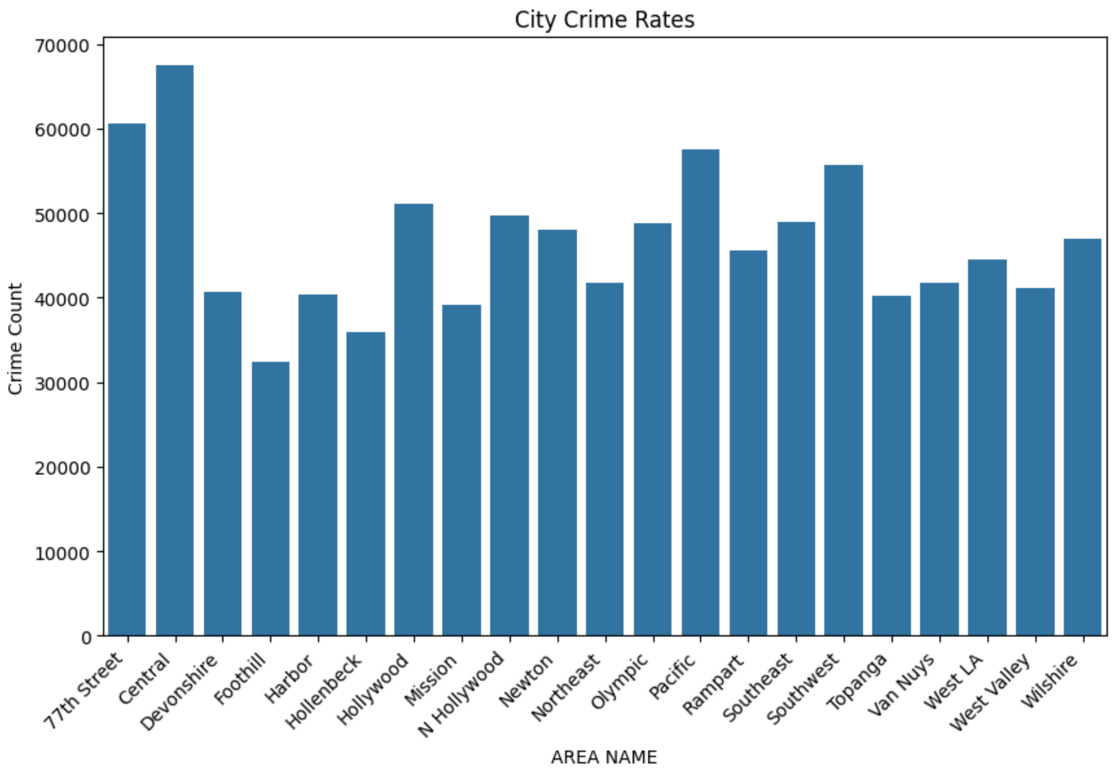
```
The most common crime is:VEHICLE - STOLEN
```



- Investigate if there are any notable differences in crime rates between regions or cities.

[4]:
```
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
cp = pd.read_csv(r"c:\ms\IE6400\Crimedate_till_sep29.csv")
# Group the data by 'AREA NAME' and count the occurences of'Crm Cd Desc' (Crime␣
↪types)
crime_by_streetname = cp.groupby('AREA NAME')['Crm Cd Desc'].count().
↪reset_index()
crime_by_streetname.columns = ['AREA NAME', 'Crime Count'] # Rename the columns␣
↪for better understanding
plt.figure(figsize=(10,6)) # Set the figure size for the plot
# Create a bar plot for crime counts by area name
sns.barplot(x='AREA NAME', y='Crime Count', data=crime_by_streetname)
# Set the title and labels for the plot
plt.title('City Crime Rates')
plt.xlabel('AREA NAME')
plt.ylabel('Crime Count')
```

```
plt.xticks(rotation=45, ha='right') # rotate the x-axix for 45 degrees
plt.show() # display the plot
```



- Explore correlations between economic factors(if available) and crime rates.

[17]:
```
import pandas as pd
# Read the crime data from specified csv file into a datframe
cp = pd.read_csv(r"c:\ms\IE6400\Crimedate_till_sep29.csv")
# Read the unemployment data from the specified csv file into a dataframe
uer = pd.read_csv(r"c:\ms\IE6400\unemployment_data.csv")
# Convert the 'DATE OCC' column in the crime dataframe to datetime format,␣
↪coercing invalid entries to NaT
cp['DATE OCC'] = pd.to_datetime(cp['DATE OCC'], errors='coerce')
# Convert the 'DATE' column in the unemployment rate dataframe to datetime␣
↪format, coercing invalid entries to NaT
uer['DATE'] = pd.to_datetime(uer['DATE'], errors='coerce')
# Create a new column 'month' in the crime dataframe representing the month of␣
↪each crime occurence
cp['month'] = cp['DATE OCC'].dt.to_period('M')
# Create a new column 'month' in the unemployment rate datframe representing␣
↪the month of each unemployment record
uer['month'] = uer['DATE'].dt.to_period('M')
```

```
# Merge the two dataframes on the 'month' column using inner join
merged_df = pd.merge(cp, uer, on='month', how='inner')
# save the merged dataframe to a new csv file named'␣
↪crime_and_unemployment_rate.csv'
merged_df.to_csv('crime_and_unemploymentrate.csv', index=False)
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
# read the merged csv file containing crime and unemployment rate data into a␣
↪dataframe
cer = pd.read_csv(r"C:\ms\jupyter\crime_and_unemploymentrate.csv")
cer.head() # Display first few rows of the dataframe
```

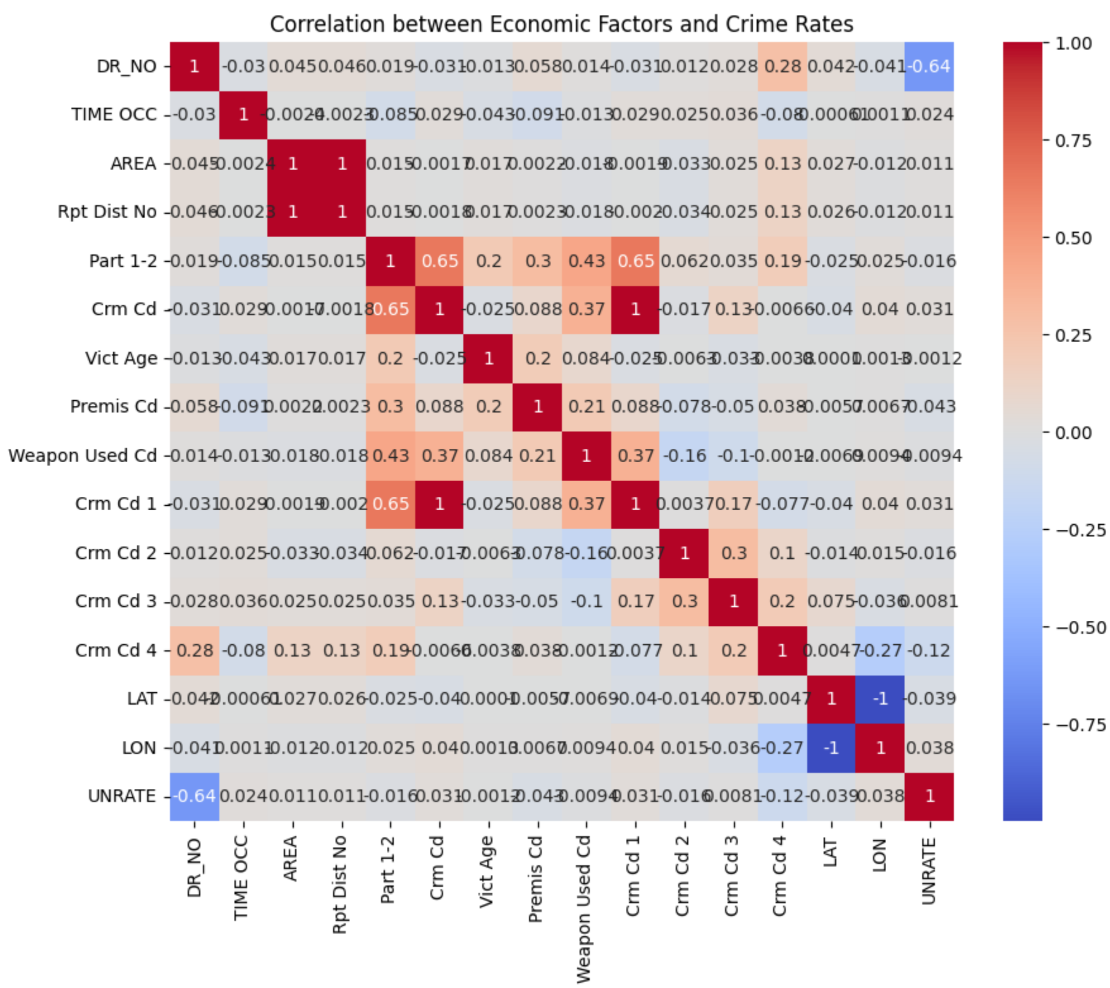
[17]:

```
       DR_NO          Date Rptd    DATE OCC  TIME OCC  AREA  AREA NAME  \
0  190326475  03-01-2020 00:00  2020-03-01      2130     7   Wilshire
1  200106753  02-09-2020 00:00  2020-02-08      1800     1    Central
2  200320258  11-11-2020 00:00  2020-11-04      1700     3  Southwest
3  200907217  05-10-2023 00:00  2020-03-10      2037     9   Van Nuys
4  231808869  04-04-2023 00:00  2020-12-01      2300    18  Southeast

   Rpt Dist No  Part 1-2  Crm Cd                               Crm Cd Desc  \
0          784         1     510                          VEHICLE - STOLEN
1          182         1     330                     BURGLARY FROM VEHICLE
2          356         1     480                              BIKE - STOLEN
3          964         1     343  SHOPLIFTING-GRAND THEFT ($950.01 & OVER)
4         1826         2     354                         THEFT OF IDENTITY

   … Crm Cd 2  Crm Cd 3 Crm Cd 4                                  LOCATION  \
0  …    998.0       NaN      NaN   1900 S  LONGWOOD                      AV
1  …    998.0       NaN      NaN   1000 S  FLOWER                        ST
2  …      NaN       NaN      NaN   1400 W  37TH                          ST
3  …      NaN       NaN      NaN  14000    RIVERSIDE                     DR
4  …      NaN       NaN      NaN   9900    COMPTON                       AV

   Cross Street      LAT       LON    month        DATE UNRATE
0           NaN  34.0375 -118.3506  2020-03  2020-03-01    4.4
1           NaN  34.0444 -118.2628  2020-02  2020-02-01    3.5
2           NaN  34.0210 -118.3002  2020-11  2020-11-01    6.7
3           NaN  34.1576 -118.4387  2020-03  2020-03-01    4.4
4           NaN  33.9467 -118.2463  2020-12  2020-12-01    6.7

[5 rows x 31 columns]
```

[18]:

```
# Read the merged csv file containing crime and unemployment rate data
cer = pd.read_csv(r"C:\ms\jupyter\crime_and_unemploymentrate.csv")
# Identify the non numeric columns in dataframe
non_numeric_columns = cer.select_dtypes(exclude=['float64', 'int64']).columns
```

```
# Convert the 'DATE OCC' column to datetime format, coercing invalid entries to NaT
cer['DATE OCC'] = pd.to_datetime(cer['DATE OCC'], errors='coerce')
# Convert the 'DATE' column to datetime format, coercing invalid entries to NaT
cer['DATE'] = pd.to_datetime(cer['DATE'], errors='coerce')
# Identify numeric columns in the dataframe
numeric_columns = cer.select_dtypes(include=['float64', 'int64']).columns
# Calculate the correlation matrix for the numeric columns
correlation_matrix = cer[numeric_columns].corr()
plt.figure(figsize=(10, 8)) # set the figure size for the plot
# Create the heatmap to visualize the correlation matrix with annotations
sns.heatmap(correlation_matrix, annot=True, cmap='coolwarm')
plt.title('Correlation between Economic Factors and Crime Rates') # set the title for heatmap
plt.show() # display the heatmap
```



- Analyze the relationship between the day of the week and the frequency of certain types of crimes.

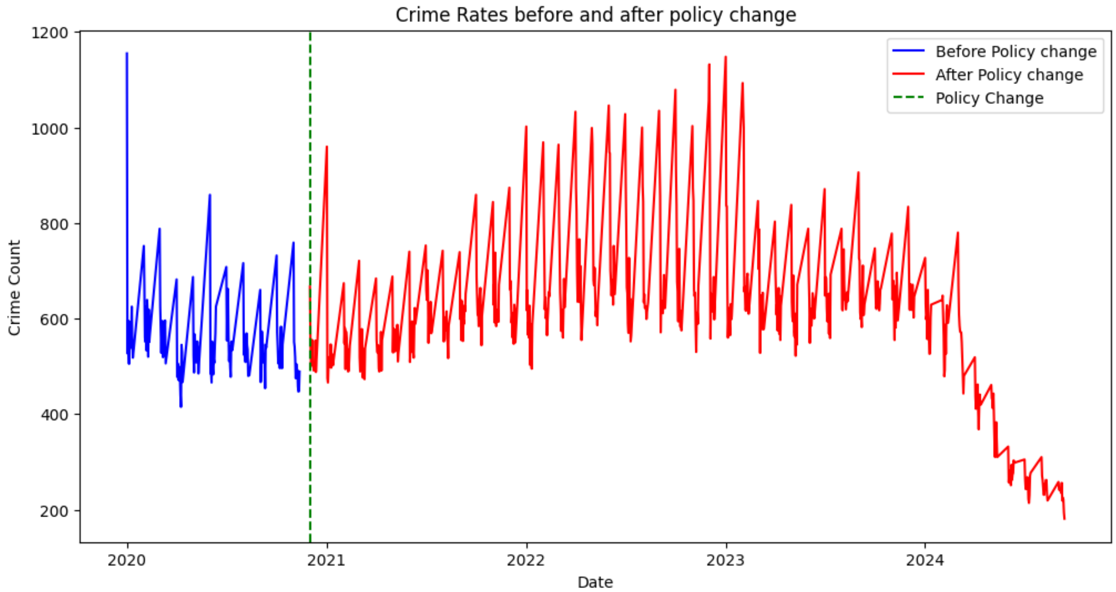
[19]:
```
# Convert the 'DATE OCC' column to datetime format, coercing invalid entries to␣
↪NaT
cp['DATE OCC'] = pd.to_datetime(cp['DATE OCC'], errors='coerce')
# Extract the day of week from 'DATE OCC' column and create a new column␣
↪'Day_of_Week
cp['Day_of_Week'] = cp['DATE OCC'].dt.day_name()
# Group the data by 'Day_of_week' and 'Crm Cd Desc', to get crime count
crime_by_day = cp.groupby(['Day_of_Week', 'Crm Cd Desc']).size().
↪reset_index(name='Crime Count')
plt.figure(figsize=(10,6)) # Set the size of figure for the plot
# specify the order of the days of the week for the x-axis
day_order = ['Monday', 'Tuesday', 'Wednesday', 'Thursday', 'Friday',␣
↪'Saturday', 'Sunday']
# Create a bar plot showing crime counts by day of the week, with hue for␣
↪different crime types
sns.barplot(x='Day_of_Week', y='Crime Count', hue='Crm Cd Desc',␣
↪data=crime_by_day, order=day_order)
# set the title and labels for the plot
plt.title('Crime Frequency by Day of the week and Crime Type')
plt.xlabel('Day of the Week')
plt.ylabel('Crime Count')
plt.xticks(rotation=45, ha='right') # rotate x-axis for 45 degrees
# Add a legend with a title for crime types, positioned outside the plot
plt.legend(title='Crime Type', bbox_to_anchor=(1,1), loc='upper left')
plt.show() # display the plot
```


Crime Frequency by Day of the week and Crime Type
Crime Count
0
1000
2000
3000
4000
5000
6000
7000
Monday
Tuesday
Wednesday
Thursday
Friday
Saturday
Sunday
Day of the Week
Crime Type
ARSON
ASSAULT WITH DEADLY WEAPON ON POLICE OFFICER
ASSAULT WITH DEADLY WEAPON, AGGRAVATED ASSAULT
ATTEMPTED ROBBERY
BATTERY - SIMPLE ASSAULT
BATTERY ON A FIREFIGHTER
BATTERY POLICE (SIMPLE)
BATTERY WITH SEXUAL CONTACT
BEASTIALITY, CRIME AGAINST NATURE SEXUAL ASSLT WITH ANIM
BIKE - ATTEMPTED STOLEN
BIKE - STOLEN
BOAT - STOLEN
BOMB SCARE
BRANDISH WEAPON
BUNCO, ATTEMPT
BUNCO, GRAND THEFT
BUNCO, PETTY THEFT
BURGLARY
BURGLARY FROM VEHICLE
BURGLARY FROM VEHICLE, ATTEMPTED
BURGLARY, ATTEMPTED
CHILD ABANDONMENT
CHILD ABUSE (PHYSICAL) - AGGRAVATED ASSAULT
CHILD ABUSE (PHYSICAL) - SIMPLE ASSAULT
CHILD ANNOYING (17YRS & UNDER)
CHILD NEGLECT (SEE 300 W.I.C.)
CHILD PORNOGRAPHY
CHILD STEALING
CONTEMPT OF COURT
CONTRIBUTING
COUNTERFEIT
CREDIT CARDS, FRAUD USE ($950 & UNDER
CREDIT CARDS, FRAUD USE ($950.01 & OVER)
CRIMINAL HOMICIDE
CRIMINAL THREATS - NO WEAPON DISPLAYED
CRM AGNST CHLD (13 OR UNDER) (14-15 & SUSP 10 YRS OLDER)
CRUELTY TO ANIMALS
DEFRAUDING INNKEEPER/THEFT OF SERVICES, $950 & UNDER
DEFRAUDING INNKEEPER/THEFT OF SERVICES, OVER $950.01
DISCHARGE FIREARMS/SHOTS FIRED
DISHONEST EMPLOYEE - GRAND THEFT
DISHONEST EMPLOYEE - PETTY THEFT
DISRUPT SCHOOL
DISTURBING THE PEACE
DOCUMENT FORGERY / STOLEN FELONY
DOCUMENT WORTHLESS ($200 & UNDER)
DOCUMENT WORTHLESS ($200.01 & OVER)
DRIVING WITHOUT OWNER CONSENT (DWOC)
DRUNK ROLL
EMBEZZLEMENT, GRAND THEFT ($950.01 & OVER)
EMBEZZLEMENT, PETTY THEFT ($950 & UNDER)
EXTORTION
FAILURE TO YIELD
FALSE IMPRISONMENT
FALSE POLICE REPORT
HUMAN TRAFFICKING - COMMERCIAL SEX ACTS
HUMAN TRAFFICKING - INVOLUNTARY SERVITUDE
ILLEGAL DUMPING
INCEST (SEXUAL ACTS BETWEEN BLOOD RELATIVES)
INCITING A RIOT
INDECENT EXPOSURE
INTIMATE PARTNER - AGGRAVATED ASSAULT
INTIMATE PARTNER - SIMPLE ASSAULT
KIDNAPPING
KIDNAPPING - GRAND ATTEMPT
LETTERS, LEWD - TELEPHONE CALLS, LEWD
LEWD CONDUCT
LEWD/LASCIVIOUS ACTS WITH CHILD
LYNCHING - ATTEMPTED
ORAL COPULATION
OTHER ASSAULT
OTHER MISCELLANEOUS CRIME
PANDERING
PEEPING TOM
PICKPOCKET
PIMPING
PROWLER
PURSE SNATCHING
RAPE, ATTEMPTED
RAPE, FORCIBLE
RECKLESS DRIVING
RESISTING ARREST
ROBBERY
SEX OFFENDER REGISTRANT OUT OF COMPLIANCE
SEX,UNLAWFUL(INC MUTUAL CONSENT, PENETRATION W/ FRGN OBJ
SEXUAL PENETRATION W/FOREIGN OBJECT
SHOPLIFTING - ATTEMPT
SHOPLIFTING - PETTY THEFT ($950 & UNDER)
SHOPLIFTING-GRAND THEFT ($950.01 & OVER)
SHOTS FIRED AT INHABITED DWELLING
SHOTS FIRED AT MOVING VEHICLE, TRAIN OR AIRCRAFT
SODOMY/SEXUAL CONTACT B/W PENIS OF ONE PERS TO ANUS OTH
STALKING
TELEPHONE PROPERTY - DAMAGE
THEFT FROM MOTOR VEHICLE - ATTEMPT
THEFT FROM MOTOR VEHICLE - GRAND ($950.01 AND OVER)
THEFT FROM MOTOR VEHICLE - PETTY ($950 & UNDER)
THEFT FROM PERSON - ATTEMPT
THEFT OF IDENTITY
THEFT PLAIN - ATTEMPT
THEFT PLAIN - PETTY ($950 & UNDER)
THEFT, COIN MACHINE - ATTEMPT
THEFT, COIN MACHINE - PETTY ($950 & UNDER)
THEFT, PERSON
THEFT-GRAND ($950.01 & OVER)EXCPT,GUNS,FOWL,LIVESTK,PROD
THREATENING PHONE CALLS/LETTERS
THROWING OBJECT AT MOVING VEHICLE
TILL TAP - PETTY ($950 & UNDER)
TRESPASSING
UNAUTHORIZED COMPUTER ACCESS
VANDALISM - FELONY ($400 & OVER, ALL CHURCH VANDALISMS)
VANDALISM - MISDEAMEANOR ($399 OR UNDER)
VEHICLE - ATTEMPT STOLEN
VEHICLE - STOLEN
VEHICLE, STOLEN - OTHER (MOTORIZED SCOOTERS, BIKES, ETC)
VIOLATION OF COURT ORDER
VIOLATION OF RESTRAINING ORDER
VIOLATION OF TEMPORARY RESTRAINING ORDER
WEAPONS POSSESSION/BOMBING
BIGAMY
CONSPIRACY
DRUGS, TO A MINOR
GRAND THEFT / AUTO REPAIR
GRAND THEFT / INSURANCE FRAUD
PICKPOCKET, ATTEMPT
REPLICA FIREARMS(SALE,DISPLAY,MANUFACTURE OR DISTRIBUTE)
THEFT, COIN MACHINE - GRAND ($950.01 & OVER)
TILL TAP - GRAND THEFT ($950.01 & OVER)
FAILURE TO DISPERSE
LYNCHING
MANSLAUGHTER, NEGLIGENT
PETTY THEFT - AUTO REPAIR
PURSE SNATCHING - ATTEMPT
BLOCKING DOOR INDUCTION CENTER
DISHONEST EMPLOYEE ATTEMPTED THEFT
FIREARMS EMERGENCY PROTECTIVE ORDER (FIREARMS EPO)
BRIBERY

- Investigate any impact of significant events or policy changes on crime rates.

[20]:
```python
# Convert the 'DATE OCC' column to datetime format, coercing invalid entries to␣
↪NaT
cp['DATE OCC'] = pd.to_datetime(cp['DATE OCC'], errors='coerce')
# Group the data by date and count the occurencies to get crime counts
crime_by_date = cp.groupby(cp['DATE OCC'].dt.date).size().
↪reset_index(name='Crime Count')
# Convert the "DATE OCC' column back to datetime for plotting
crime_by_date['DATE OCC'] = pd.to_datetime(crime_by_date['DATE OCC'],␣
↪errors='coerce')
# define the date for policy change
# let's assume the policy had changed from 1st December 2020
event_date = pd.to_datetime('2020-12-01').date()
print(f"The crime rate policy has been changed from 01 December 2020")
# Filter the data to seperate crime counts before and after the policy change
before_event = crime_by_date[crime_by_date['DATE OCC'].dt.date < event_date]
after_event = crime_by_date[crime_by_date['DATE OCC'].dt.date>= event_date]
plt.figure(figsize=(12,6)) #  set the figure size for the plot
# plot the crime counts before the policy change
plt.plot(before_event['DATE OCC'], before_event['Crime Count'], label='Before␣
↪Policy change', color='blue')
# plot the crime counts after the policy change
plt.plot(after_event['DATE OCC'], after_event['Crime Count'], label='After␣
↪Policy change', color='red')
# add a vertical line to indicate the date of policy change
plt.axvline(event_date, color='green', linestyle='--', label='Policy Change')
# set the title and labels
plt.title('Crime Rates before and after policy change')
plt.xlabel('Date')
plt.ylabel('Crime Count')
plt.legend() # Add a legend to plot
plt.show() # display the plot
```

```
The crime rate policy has been changed from 01 December 2020
```

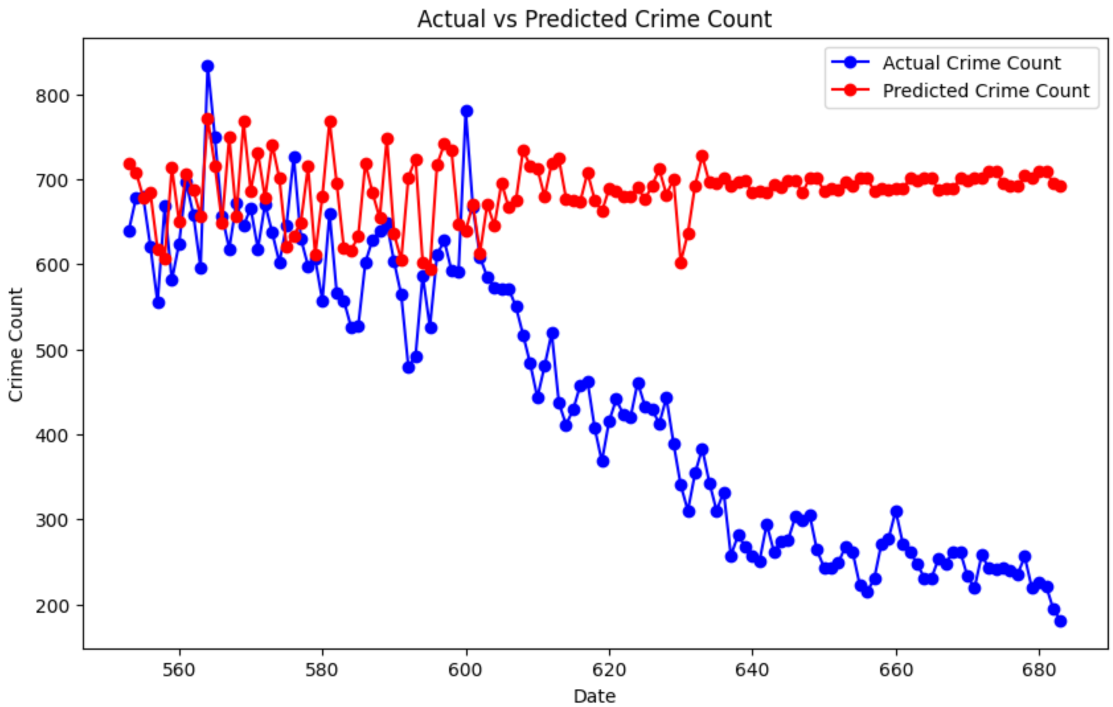


5. Advanced Analysis: - Use predictive modelling techniques (e.g., time series forecasting) to predict future crime trends.

[21]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from sklearn.ensemble import RandomForestRegressor
from sklearn.model_selection import train_test_split
from sklearn.metrics import mean_squared_error

# Load the dataset
cp = pd.read_csv(r"c:\ms\IE6400\Crimedate_till_sep29.csv")
cp['DATE OCC'] = pd.to_datetime(cp['DATE OCC'], errors='coerce')

# Group the data by date to get daily crime counts
crime_by_date = cp.groupby(cp['DATE OCC'].dt.date).size().
 ↪reset_index(name='Crime Count')

# Feature Engineering: Day of the week, month, year
crime_by_date['Day of Week'] = pd.to_datetime(crime_by_date['DATE OCC']).dt.
 ↪dayofweek
crime_by_date['Month'] = pd.to_datetime(crime_by_date['DATE OCC']).dt.month
crime_by_date['Year'] = pd.to_datetime(crime_by_date['DATE OCC']).dt.year

# Create lag features (lag 1 day, lag 7 days, etc.)
crime_by_date['Lag 1'] = crime_by_date['Crime Count'].shift(1)
crime_by_date['Lag 7'] = crime_by_date['Crime Count'].shift(7)
```

```
crime_by_date['Lag 30'] = crime_by_date['Crime Count'].shift(30)

# Drop rows with NaN values caused by lagging
crime_by_date.dropna(inplace=True)

# Create X and y datasets (X = features, y = target)
X = crime_by_date[['Day of Week', 'Month', 'Year', 'Lag 1', 'Lag 7', 'Lag 30']]
y = crime_by_date['Crime Count']
# Split the data into training and test sets (80% training, 20% testing)
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,␣
↪shuffle=False)
# Initialize Random Forest model
rf_model = RandomForestRegressor(n_estimators=100, random_state=42)

# Train the model
rf_model.fit(X_train, y_train)
# Make predictions on the test set
y_pred = rf_model.predict(X_test)

# Evaluate the model using RMSE and R-squared
rmse = mean_squared_error(y_test, y_pred, squared=False)
r_squared = rf_model.score(X_test, y_test)

print(f'RMSE: {rmse}')
from sklearn.metrics import mean_absolute_error

# Calculate MAE
mae = mean_absolute_error(y_test, y_pred)

print(f'MAE: {mae}')

# Plot the actual vs predicted crime counts
plt.figure(figsize=(10, 6))
plt.plot(y_test.index, y_test.values, label='Actual Crime Count', color='blue',␣
↪marker='o')
plt.plot(y_test.index, y_pred, label='Predicted Crime Count', color='red',␣
↪marker='o')
plt.title('Actual vs Predicted Crime Count')
plt.xlabel('Date')
plt.ylabel('Crime Count')
plt.legend()
plt.show()
```

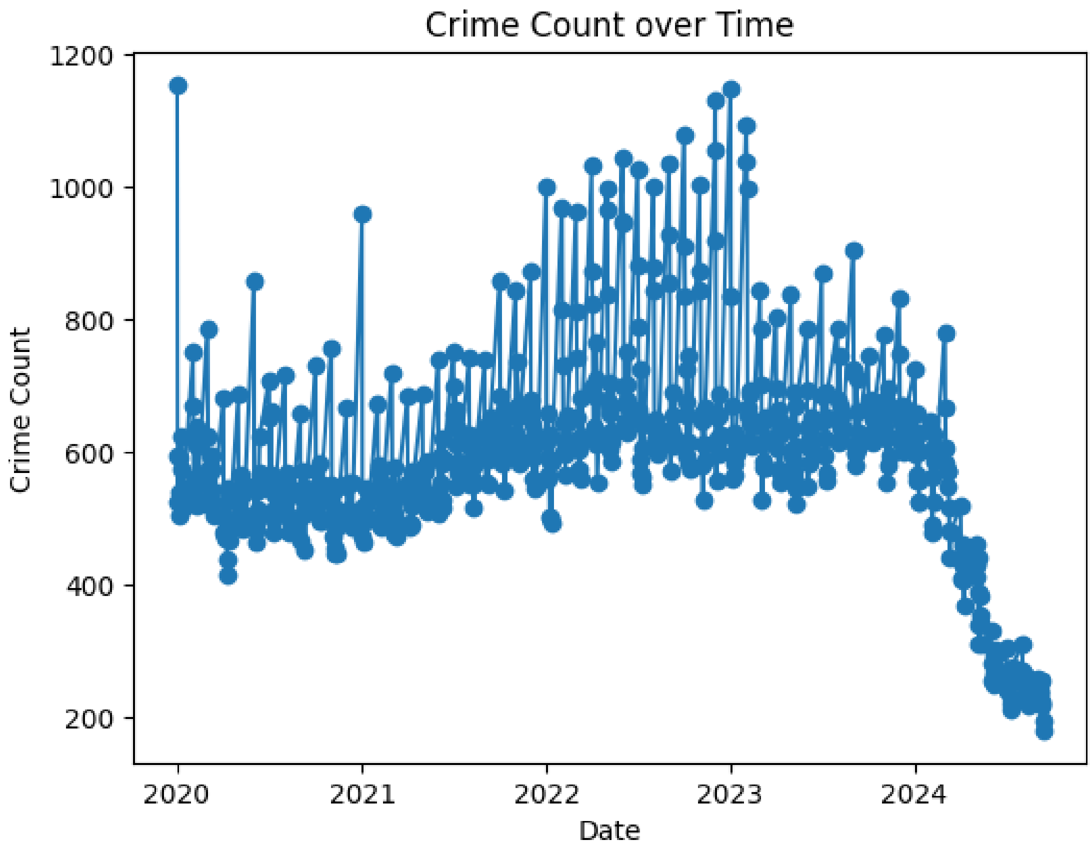
```
RMSE: 302.29649104937255
MAE: 250.9690076335878
```



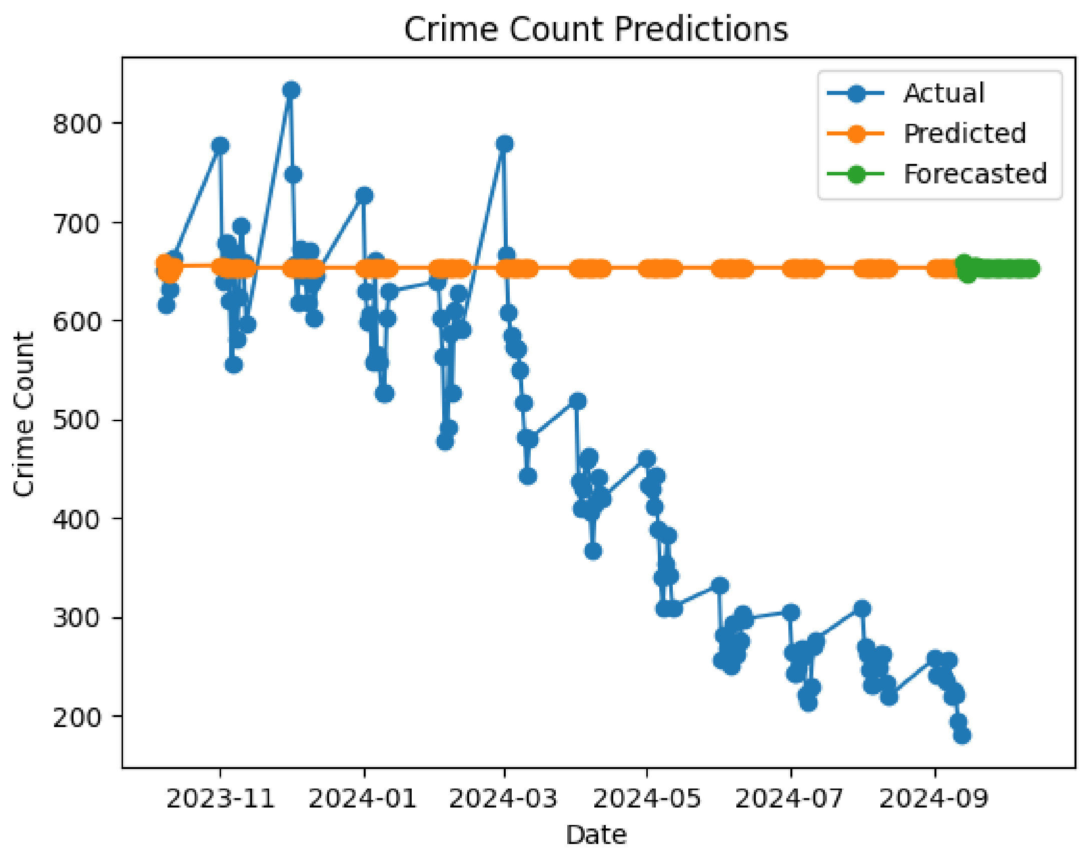
[22]:

```
# Import ARIMA model for time series forecasting
from statsmodels.tsa.arima.model import ARIMA
from sklearn.metrics import mean_squared_error, r2_score  # Import r2_score for␣
 ↪R-squared

# Convert the 'DATE OCC' column to datetime format, coercing invalid entries to␣
 ↪NaT
cp['DATE OCC'] = pd.to_datetime(cp['DATE OCC'], errors='coerce')

# Group the data by the date and count the occurrences to get daily crime counts
crime_by_date = cp.groupby(cp['DATE OCC'].dt.date).size().
 ↪reset_index(name='Crime Count')

# Plot the daily crime counts
plt.plot(crime_by_date['DATE OCC'], crime_by_date['Crime Count'], marker='o')
plt.title('Crime Count over Time')
plt.xlabel('Date')
plt.ylabel('Crime Count')
plt.show()

# Split the data into training and testing sets (80% training, 20% testing)
train_size = int(len(crime_by_date) * 0.8)
train, test = crime_by_date[:train_size], crime_by_date[train_size:]
```

```
# Fit an ARIMA model to the training data
model = ARIMA(train['Crime Count'], order=(5, 1, 0))  # Order is (p, d, q)
model_fit = model.fit()  # Fit the model to training data

# Forecast the test data period
predictions = model_fit.forecast(steps=len(test))

# Calculate the root mean squared error (RMSE) of the predictions
rmse = mean_squared_error(test['Crime Count'], predictions, squared=False)
print(f'RMSE: {rmse}')

# Forecast future values for the next 30 days
future_predictions = model_fit.forecast(steps=30)

# Plot actual vs predicted values for the test set
plt.plot(test['DATE OCC'].values, test['Crime Count'], label='Actual',␣
↪marker='o')
plt.plot(test['DATE OCC'].values, predictions, label='Predicted', marker='o')

# Generate future dates for the forecasted values
future_dates = pd.date_range(start=test['DATE OCC'].values[-1] + pd.
↪Timedelta(days=1), periods=30)

# Plot future predictions
plt.plot(future_dates, future_predictions, label='Forecasted', marker='o')

plt.legend()  # Adding a legend
plt.title('Crime Count Predictions')
plt.xlabel('Date')
plt.ylabel('Crime Count')
plt.show()  # Display the plot
```



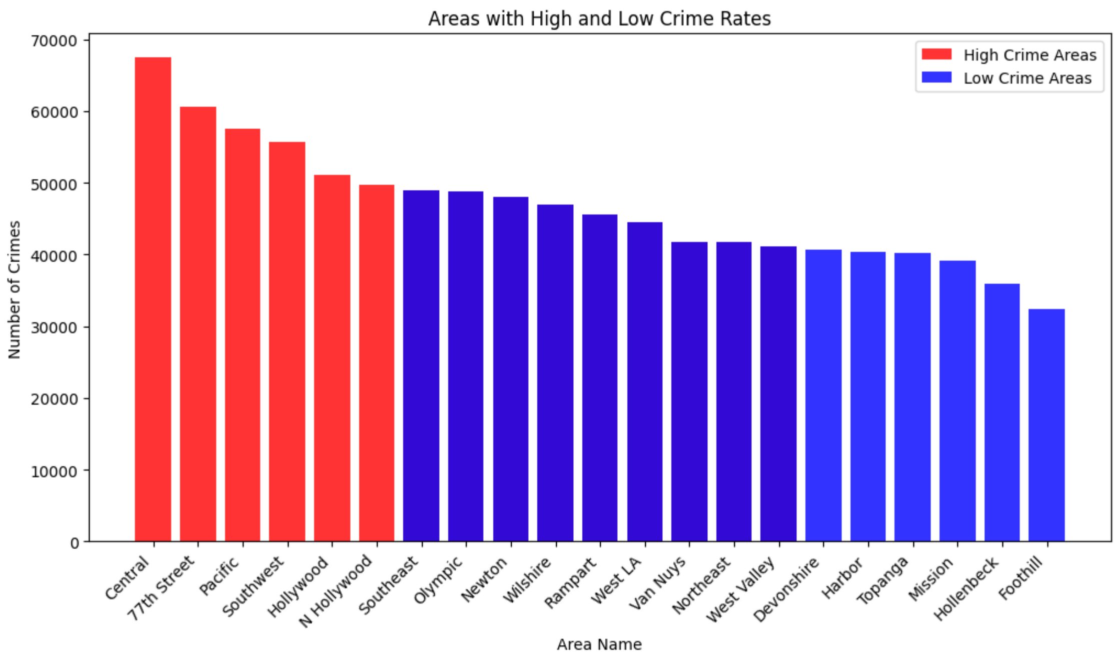

```
RMSE: 265.2728192163623
```



- Explore additional questions or hypothesis related to the dataset.
- Identify and compare areas with high and low crime rates based on the available dataset?

[23]:
```
# count the crimes in each area
crime_by_area = cp['AREA NAME'].value_counts()
# sort the areas by number of crimes in descending order
crime_by_area = crime_by_area.sort_values(ascending=False)
top_15_areas = crime_by_area.head(15) # get top 15 areas with highest crime␣
↪counts
bottom_15_areas = crime_by_area.tail(15) # get bottom 15 areas with lowest␣
↪crime counts
plt.figure(figsize=(12,6)) # set the figure size for plot
# plot the top 15 areas with high crime rates
plt.bar(top_15_areas.index, top_15_areas.values, color='red', alpha=0.8,␣
↪label='High Crime Areas')
# plot the bottom 15 areas with low crime rates
plt.bar(bottom_15_areas.index, bottom_15_areas.values, color='blue', alpha=0.8,␣
↪label='Low Crime Areas')
# set the title and labels for the plot
```

```
plt.title('Areas with High and Low Crime Rates')
plt.xlabel('Area Name')
plt.ylabel('Number of Crimes')
plt.xticks(rotation=45, ha='right') # rotate the x-axis to 45 degrees for␣
↪better readability
plt.legend() # add legend to differentiate between high and low crime areas
plt.show() # display the plot
```



Visualize geographic crime data using latitude and longitude coordinates.

[27]:
```
import geopandas as gpd # import geopandas for geospatial data handling
import folium # import folium for creating interactive maps
from folium.plugins import FastMarkerCluster # import fastmarkercluster for␣
↪clusteringmarkers on the map
import pandas as pd # import pandas for data manipulation
# convert latitude and longitude columns to numeric type for proper formatting
cp['LAT'] = pd.to_numeric(cp['LAT'])
cp['LON'] = pd.to_numeric(cp['LON'])
# create a geodataframe using the latitude and longitude for geometry
gdf = gpd.GeoDataFrame(cp, geometry=gpd.points_from_xy(cp.LON, cp.LAT))
# create a base map centered on los angeles with a specified zoom level and␣
↪tile style
m = folium.Map(location=[42.3601, -71.0589], zoom_start=10,␣
↪tiles='openstreetmap')
# add clustered markers for crime locations on the map
```

```python
FastMarkerCluster(data=list(zip(gdf['LAT'], gdf['LON']))).add_to(m)
#save map to html file for viewing in a web browser
m.save('crime_hotspots_map_boston.html')
m # display the map in a jupyter notebook
```

[27]: <folium.folium.Map at 0x187600b1820>

[ ]:

```python

```