Assignment_4

July 16, 2025

1 Assignment 4

220107089 | Altynbek Adilkhan

Q1: Import the required libraries and Modules

[1]:
```python
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
```

Q2: Read the file 'Boston_crime.csv'. Use the argument encoding = 'latin-1' if you find problem related to encoding

#Read the file Boston_crime.csv. If you found an error use the argument encoding = 'latin-1'

[2]:
```python
try:
    crime_df = pd.read_csv('Desktop/SDU/DBMS2/Boston_crime.csv')
except UnicodeDecodeError:
    crime_df = pd.read_csv('Desktop/SDU/DBMS2/Boston_crime.csv',␣
 ↪encoding='latin-1')


crime_df.head(5)
```

[2]:
```
  INCIDENT_NUMBER  OFFENSE_CODE    OFFENSE_CODE_GROUP   OFFENSE_DESCRIPTION  \
0      I182070945           619               Larceny    LARCENY ALL OTHERS
1      I182070943          1402             Vandalism             VANDALISM
2      I182070941          3410                 Towed   TOWED MOTOR VEHICLE
3      I182070940          3114  Investigate Property  INVESTIGATE PROPERTY
4      I182070938          3114  Investigate Property  INVESTIGATE PROPERTY

  DISTRICT REPORTING_AREA SHOOTING  YEAR  MONTH DAY_OF_WEEK  HOUR    UCR_PART  \
0      D14            808      NaN  2020      4      Sunday    13    Part One
1      C11            347      NaN  2020      4     Tuesday     0    Part Two
2       D4            151      NaN  2020      4      Monday    19  Part Three
3       D4            272      NaN  2020      4      Monday    21  Part Three
4       B3            421      NaN  2020      4      Monday    21  Part Three
```

```
        STREET        Lat       Long                     Location
0   LINCOLN ST  42.357791 -71.139371  (42.35779134, -71.13937053)
1     HECLA ST  42.306821 -71.060300  (42.30682138, -71.06030035)
2  CAZENOVE ST  42.346589 -71.072429  (42.34658879, -71.07242943)
3   NEWCOMB ST  42.334182 -71.078664  (42.33418175, -71.07866441)
4     DELHI ST  42.275365 -71.090361  (42.27536542, -71.09036101)
```

Q3: Show your tabulated data and calculate the five number summary

```
#Show your tabulated data
#Five number summary
```

[3]:
```python
crime_df.describe(include='all')
```

[3]:
```
       INCIDENT_NUMBER   OFFENSE_CODE               OFFENSE_CODE_GROUP  \
count           319073  319073.000000                           319073
unique          282517            NaN                               67
top         I162030584            NaN  Motor Vehicle Accident Response
freq                13            NaN                            37132
mean               NaN    2317.546956                              NaN
std                NaN    1185.285543                              NaN
min                NaN     111.000000                              NaN
25%                NaN    1001.000000                              NaN
50%                NaN    2907.000000                              NaN
75%                NaN    3201.000000                              NaN
max                NaN    3831.000000                              NaN

                   OFFENSE_DESCRIPTION DISTRICT REPORTING_AREA SHOOTING  \
count                           319073   317308         319073     1019
unique                             244       12            879        1
top     SICK/INJURED/MEDICAL - PERSON       B2                       Y
freq                             18783    49945          20250     1019
mean                               NaN      NaN            NaN      NaN
std                                NaN      NaN            NaN      NaN
min                                NaN      NaN            NaN      NaN
25%                                NaN      NaN            NaN      NaN
50%                                NaN      NaN            NaN      NaN
75%                                NaN      NaN            NaN      NaN
max                                NaN      NaN            NaN      NaN

                 YEAR          MONTH DAY_OF_WEEK           HOUR    UCR_PART  \
count   319073.000000  319073.000000      319073  319073.000000      318983
unique            NaN            NaN           7            NaN           4
top               NaN            NaN      Friday            NaN  Part Three
freq              NaN            NaN       48495            NaN      158553
mean      2018.393264       5.158609         NaN      13.118205         NaN
std          1.286184       3.304448         NaN       6.294205         NaN
min       2016.000000       1.000000         NaN       0.000000         NaN
```

```
25%       2018.000000      4.000000        NaN     9.000000        NaN
50%       2019.000000      4.000000        NaN    14.000000        NaN
75%       2019.000000      4.000000        NaN    18.000000        NaN
max       2020.000000     12.000000        NaN    23.000000        NaN

                STREET            Lat           Long                  Location
count           308202  299074.000000  299074.000000                    319073
unique            4657            NaN            NaN                     18194
top     WASHINGTON ST             NaN            NaN  (0.00000000, 0.00000000)
freq             14194            NaN            NaN                     19999
mean               NaN      42.214381     -70.908272                       NaN
std                NaN       2.159766       3.493618                       NaN
min                NaN      -1.000000     -71.178674                       NaN
25%                NaN      42.297442     -71.097135                       NaN
50%                NaN      42.325538     -71.077524                       NaN
75%                NaN      42.348624     -71.062467                       NaN
max                NaN      42.395042      -1.000000                       NaN
```

[4]:
```
crime_df.describe(percentiles=[0.25, 0.5, 0.75]).loc[['min', '25%', '50%',
 '75%', 'max']]
```

[4]:
```
     OFFENSE_CODE    YEAR  MONTH  HOUR        Lat       Long
min         111.0  2016.0    1.0   0.0  -1.000000 -71.178674
25%        1001.0  2018.0    4.0   9.0  42.297442 -71.097135
50%        2907.0  2019.0    4.0  14.0  42.325538 -71.077524
75%        3201.0  2019.0    4.0  18.0  42.348624 -71.062467
max        3831.0  2020.0   12.0  23.0  42.395042  -1.000000
```

Q4: Determine the data types of the features in the dataset ?

```
# What are the dataset features types (dtypes)
```

[5]:
```
crime_df.dtypes
```

[5]:
```
INCIDENT_NUMBER         object
OFFENSE_CODE             int64
OFFENSE_CODE_GROUP      object
OFFENSE_DESCRIPTION     object
DISTRICT                object
REPORTING_AREA          object
SHOOTING                object
YEAR                     int64
MONTH                    int64
DAY_OF_WEEK             object
HOUR                     int64
UCR_PART                object
STREET                  object
Lat                    float64
```

```
Long                       float64
Location                    object
dtype: object
```

Q5: Convert the data type of 'DAY_OF_WEEK' and 'OFFENSE_CODE_GROUP' to category type and then check the data types.

Convert the type of data in 'DAY_OF_WEEK' and 'OFFENSE_CODE_GROUP' into catrgory type
Check again the dataset features types

[6]:
```
crime_df['DAY_OF_WEEK'] = crime_df['DAY_OF_WEEK'].astype('category')
crime_df['OFFENSE_CODE_GROUP'] = crime_df['OFFENSE_CODE_GROUP'].
 ↪astype('category')
```

[7]:
```
crime_df.dtypes
```

[7]:
```
INCIDENT_NUMBER           object
OFFENSE_CODE               int64
OFFENSE_CODE_GROUP      category
OFFENSE_DESCRIPTION       object
DISTRICT                  object
REPORTING_AREA            object
SHOOTING                  object
YEAR                       int64
MONTH                      int64
DAY_OF_WEEK             category
HOUR                       int64
UCR_PART                  object
STREET                    object
Lat                      float64
Long                     float64
Location                  object
dtype: object
```

Q6: Display the value counts for the 'DAY_OF_WEEK' and 'OFFENSE_CODE_GROUP' columns.

#Show value counts of DAY_OF_WEEK and OFFENSE_CODE_GROUP columns

[8]:
```
day_of_week_counts = crime_df['DAY_OF_WEEK'].value_counts()
offense_code_group_counts = crime_df['OFFENSE_CODE_GROUP'].value_counts()
```

[9]:
```
print("Value counts for 'DAY_OF_WEEK':")
print(day_of_week_counts)

print("\nValue counts for 'OFFENSE_CODE_GROUP':")
print(offense_code_group_counts)
```

```
Value counts for 'DAY_OF_WEEK':
DAY_OF_WEEK
Friday       48495
Wednesday    46729
Thursday     46656
Tuesday      46383
Monday       45679
Saturday     44818
Sunday       40313
Name: count, dtype: int64

Value counts for 'OFFENSE_CODE_GROUP':
OFFENSE_CODE_GROUP
Motor Vehicle Accident Response            37132
Larceny                                    25935
Medical Assistance                         23540
Investigate Person                         18750
Other                                      18075
                                           …
HUMAN TRAFFICKING                              7
INVESTIGATE PERSON                             4
Biological Threat                              2
Burglary - No Property Taken                   2
HUMAN TRAFFICKING - INVOLUNTARY SERVITUDE      2
Name: count, Length: 67, dtype: int64
```

Q7: Show the unique values in the UCR_PART column to see its contents

```
# Show the unique values in the UCR_PART column to understand the contents
```

[14]:
```
ucr_part_unique = crime_df['UCR_PART'].unique()
print("Unique values in 'UCR_PART':")
print(ucr_part_unique)
```

```
Unique values in 'UCR_PART':
['Part One' 'Part Two' 'Part Three' 'Other' nan]
```

Q8: Filter the dataset to include only the rows where the UCR_PART column is Part One, Part Two, or Part Three. Don't forget to check the size of the dataset.

```
#Filter dataset and make it based only on Part One, Part Two, and Part Three (in UCR_PART colu
# Show the size of the dataset
```

[15]:
```
filtered_df = crime_df[crime_df['UCR_PART'].isin(['Part One', 'Part Two', 'Part␣
↪Three'])]
print(f"\nFiltered dataset size (rows, columns): {filtered_df.shape}")
```

```

Filtered dataset size (rows, columns): (317751, 16)
```

Q9: Drope the columns INCIDENT_NUMBER, OFFENSE_CODE, OFFENSE_DESCRIPTION, REPORTING_AREA, SHOOTING, STREET, Lat, Long, Location from the dataset.

```
Don't forget to check the above step by showing the dataset.
# Drope the columns INCIDENT_NUMBER, OFFENSE_CODE, OFFENSE_DESCRIPTION, REPORTING_AREA, SHOOTI
#Check the above step by showing the dataset
```

[16]:
```python
columns_to_drop = ['INCIDENT_NUMBER', 'OFFENSE_CODE', 'OFFENSE_DESCRIPTION',␣
 ↪'REPORTING_AREA',
                    'SHOOTING', 'STREET', 'Lat', 'Long', 'Location']
cleaned_df = filtered_df.drop(columns=columns_to_drop)

print("\nDataset after dropping specified columns:")
print(cleaned_df.head())
```

```
Dataset after dropping specified columns:
     OFFENSE_CODE_GROUP DISTRICT  YEAR  MONTH DAY_OF_WEEK  HOUR    UCR_PART
0               Larceny      D14  2020      4      Sunday    13    Part One
1             Vandalism      C11  2020      4     Tuesday     0    Part Two
2                 Towed       D4  2020      4      Monday    19  Part Three
3  Investigate Property       D4  2020      4      Monday    21  Part Three
4  Investigate Property       B3  2020      4      Monday    21  Part Three
```

Q10: Show the number of the missing values in each column at once

```
#Show the number of missing values in each column at once
```

[18]:
```python
missing_values = crime_df.isnull().sum()
print(f"Missing values in each column: \n{missing_values}")
```

```
Missing values in each column:
INCIDENT_NUMBER            0
OFFENSE_CODE               0
OFFENSE_CODE_GROUP         0
OFFENSE_DESCRIPTION        0
DISTRICT                1765
REPORTING_AREA             0
SHOOTING              318054
YEAR                       0
MONTH                      0
DAY_OF_WEEK                0
HOUR                       0
UCR_PART                  90
STREET                 10871
Lat                    19999
Long                   19999
Location                   0
```

```
dtype: int64
```

Q11: Drop the missing values in the DISTRICT column, then check to see if there are any remaining missing values.

```
#Drope the missing value on column DISTRICT
#Again, show the missing values again
```

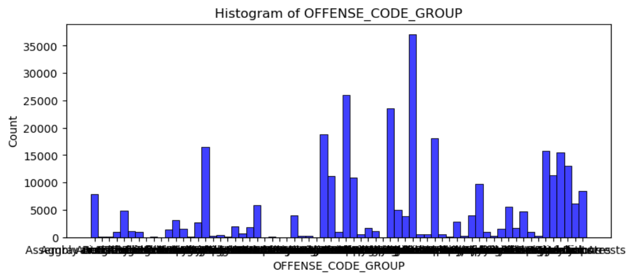
[19]:
```
crime_df_dropped_district = crime_df.dropna(subset=['DISTRICT'])
missing_values_after_dropping = crime_df_dropped_district.isnull().sum()
print(f"Missing values after dropping 'DISTRICT' column:
↪\n{missing_values_after_dropping}")
```

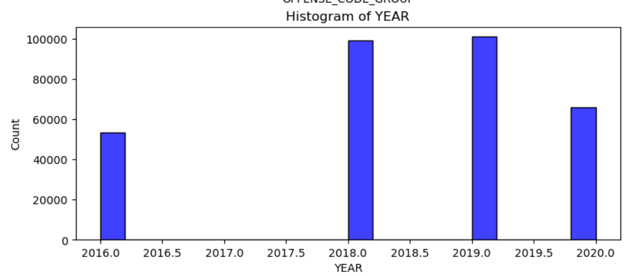
```
Missing values after dropping 'DISTRICT' column:
INCIDENT_NUMBER             0
OFFENSE_CODE                0
OFFENSE_CODE_GROUP          0
OFFENSE_DESCRIPTION         0
DISTRICT                    0
REPORTING_AREA              0
SHOOTING               316291
YEAR                        0
MONTH                       0
DAY_OF_WEEK                 0
HOUR                        0
UCR_PART                   90
STREET                   9824
Lat                     19715
Long                    19715
Location                    0
dtype: int64
```

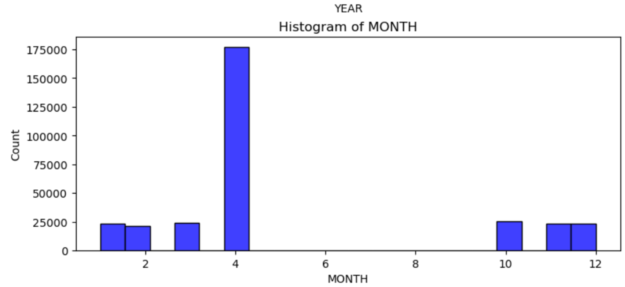
Q11: Use subplots to draw a histogram as described above.

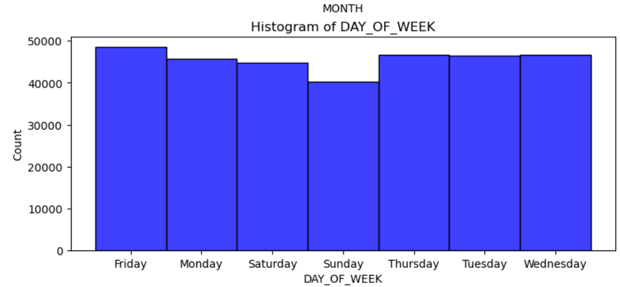
```
Don't forget to use Using constrained_layout argument to set the spacing between subplots
#Another way to graph all the features at once that your interested to study them
```

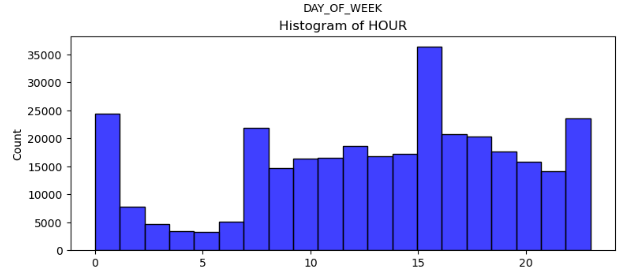
[20]:
```
columns_to_plot = ['OFFENSE_CODE_GROUP', 'YEAR', 'MONTH', 'DAY_OF_WEEK',
↪'HOUR', 'DISTRICT', 'UCR_PART']
fig, axes = plt.subplots(nrows=7, ncols=1, figsize=(8, 24),
↪constrained_layout=True)
for i, col in enumerate(columns_to_plot):
    sns.histplot(data=crime_df, x=col, ax=axes[i], kde=False, color='blue',
↪bins=20)
    axes[i].set_title(f'Histogram of {col}')
    axes[i].set_xlabel(col)
    axes[i].set_ylabel('Count')
plt.show()
```

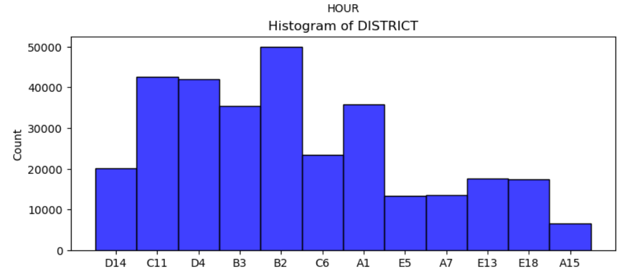

Histogram of OFFENSE_CODE_GROUP
Count
OFFENSE_CODE_GROUP
0
5000
10000
15000
20000
25000
30000
35000

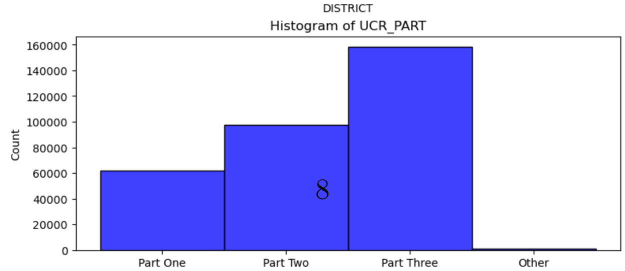

Histogram of YEAR
Count
YEAR
0
20000
40000
60000
80000
100000
2016.0
2016.5
2017.0
2017.5
2018.0
2018.5
2019.0
2019.5
2020.0


Histogram of MONTH
Count
MONTH
0
25000
50000
75000
100000
125000
150000
175000
2
4
6
8
10
12


Histogram of DAY_OF_WEEK
Count
DAY_OF_WEEK
0
10000
20000
30000
40000
50000
Friday
Monday
Saturday
Sunday
Thursday
Tuesday
Wednesday


Histogram of HOUR
Count
HOUR
0
5000
10000
15000
20000
25000
30000
35000
0
5
10
15
20


Histogram of DISTRICT
Count
DISTRICT
0
10000
20000
30000
40000
50000
D14
C11
D4
B3
B2
C6
A1
E5
A7
E13
E18
A15

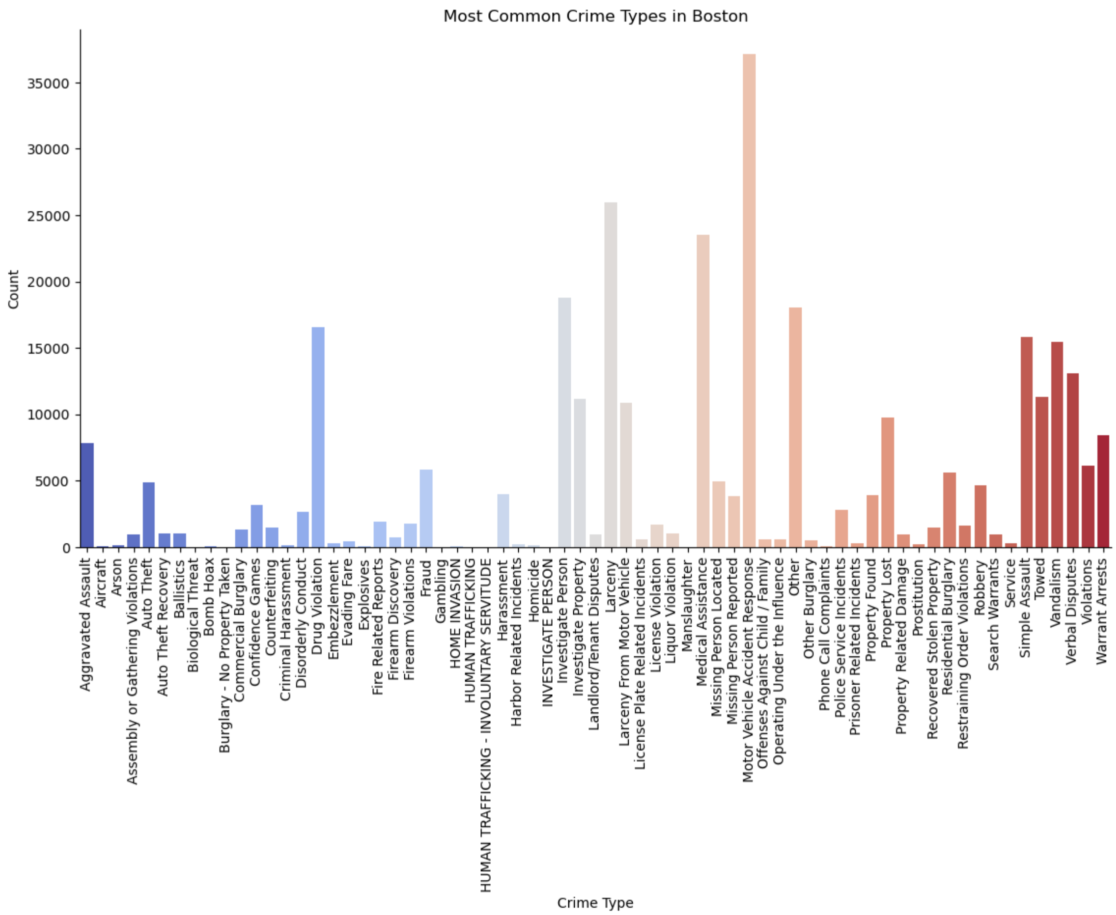

Histogram of UCR_PART
Count
UCR_PART
0
20000
40000
60000
80000
100000
120000
140000
160000
Part One
Part Two
Part Three
Other

Q12. What is the most common type of crime in Boston?

```
Answer this question both numerically, by providing counts, and graphically, by using a histog
#All the crimes type in Bostons are in the column OFFENSE_CODE_GROUP.
#Calculate the counts for all crimes
#Visualize all the crimes on Boston using catplot. Show the graph in the horizontal axis
```

[23]:
```
crime_counts = crime_df['OFFENSE_CODE_GROUP'].value_counts()
print("Crime type counts:")
print(crime_counts)
sns.catplot(data=crime_df, x='OFFENSE_CODE_GROUP', kind='count', height=6,␣
 ↪aspect=2, hue='OFFENSE_CODE_GROUP', palette='coolwarm', legend=False)
plt.xticks(rotation=90)
plt.title("Most Common Crime Types in Boston")
plt.xlabel('Crime Type')
plt.ylabel('Count')
plt.show()
```

```
Crime type counts:
OFFENSE_CODE_GROUP
Motor Vehicle Accident Response             37132
Larceny                                     25935
Medical Assistance                          23540
Investigate Person                          18750
Other                                       18075
                                            …
HUMAN TRAFFICKING                               7
INVESTIGATE PERSON                              4
Biological Threat                               2
Burglary - No Property Taken                    2
HUMAN TRAFFICKING - INVOLUNTARY SERVITUDE       2
Name: count, Length: 67, dtype: int64
```



Based on the results of Q12, give in order most common type of crime in Boston

```
Answer:
    1- Larceny Theft
    2- Motor Vehicle Theft
    3- Aggravated Assault
```

Q13. Which year has the highest rate of crimes?

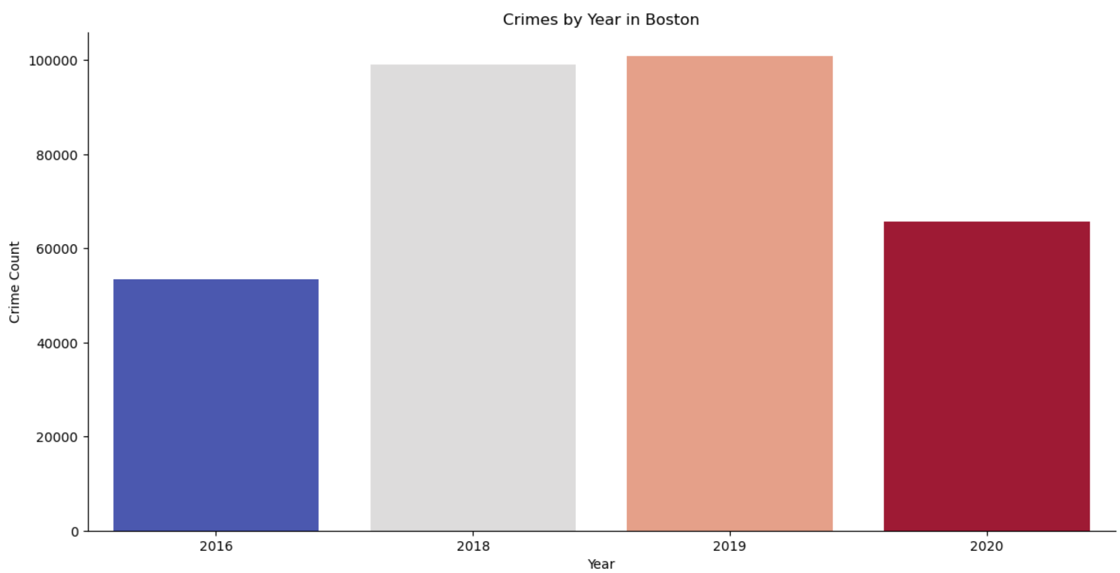
```
Answer this question both numerically, by providing counts, and graphically, by using a histog
# Calculate the counts of crimes based on year
#Use cat plot to answer Q2
```

[26]:
```python
yearly_crime_counts = crime_df['YEAR'].value_counts()
print("Crime counts by year:")
print(yearly_crime_counts)

sns.catplot(data=crime_df, x='YEAR', kind='count', height=6, aspect=2,␣
 ↪hue='YEAR', palette='coolwarm', legend=False)
```

```
plt.title("Crimes by Year in Boston")
plt.xlabel('Year')
plt.ylabel('Crime Count')
plt.show()
```

```
Crime counts by year:
YEAR
2019    100886
2018     99114
2020     65685
2016     53388
Name: count, dtype: int64
```



Based on the results of Q13, Which year has the highest rate of crimes in Boston
Answer: 2019

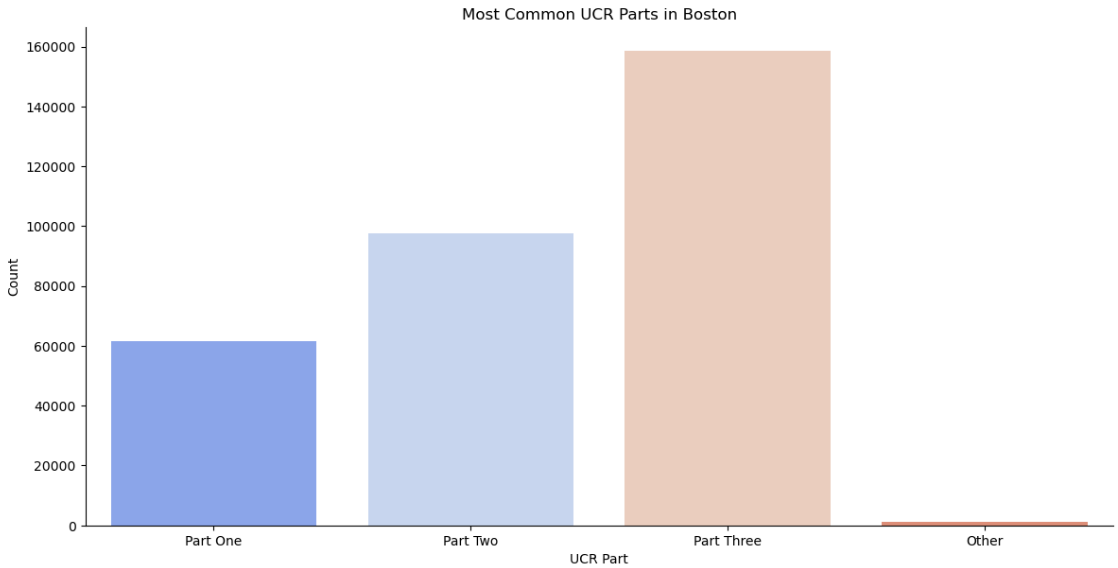
Q14. What is the most common type of Uniform Crime Reporting Offence (UCR Part) in Boston?

```
Answer this question both numerically, by providing counts, and graphically, by using a histog
#Calculate the counts of crimes that is based on UCR part
#Visualize Q14
```

[28]:
```
ucr_part_counts = crime_df['UCR_PART'].value_counts()
print("Crime counts by UCR Part:")
print(ucr_part_counts)
sns.catplot(data=crime_df, x='UCR_PART', kind='count', height=6, aspect=2,␣
 ↪hue='UCR_PART', palette='coolwarm', legend=False)
```

```
plt.title("Most Common UCR Parts in Boston")
plt.xlabel('UCR Part')
plt.ylabel('Count')
plt.show()
```

```
Crime counts by UCR Part:
UCR_PART
Part Three    158553
Part Two       97569
Part One       61629
Other           1232
Name: count, dtype: int64
```



Based on the result of Q 14, What is the most common type of Uniform Crime Reporting Offence (UCR Part) in Boston? Answer: Part Three

Q15. Which district in boston is the most dangerous district and which one is the safest? District Names To help you see the names of each district, we've replaced the district codes with their corresponding real names. This adjustment has already been made for you.

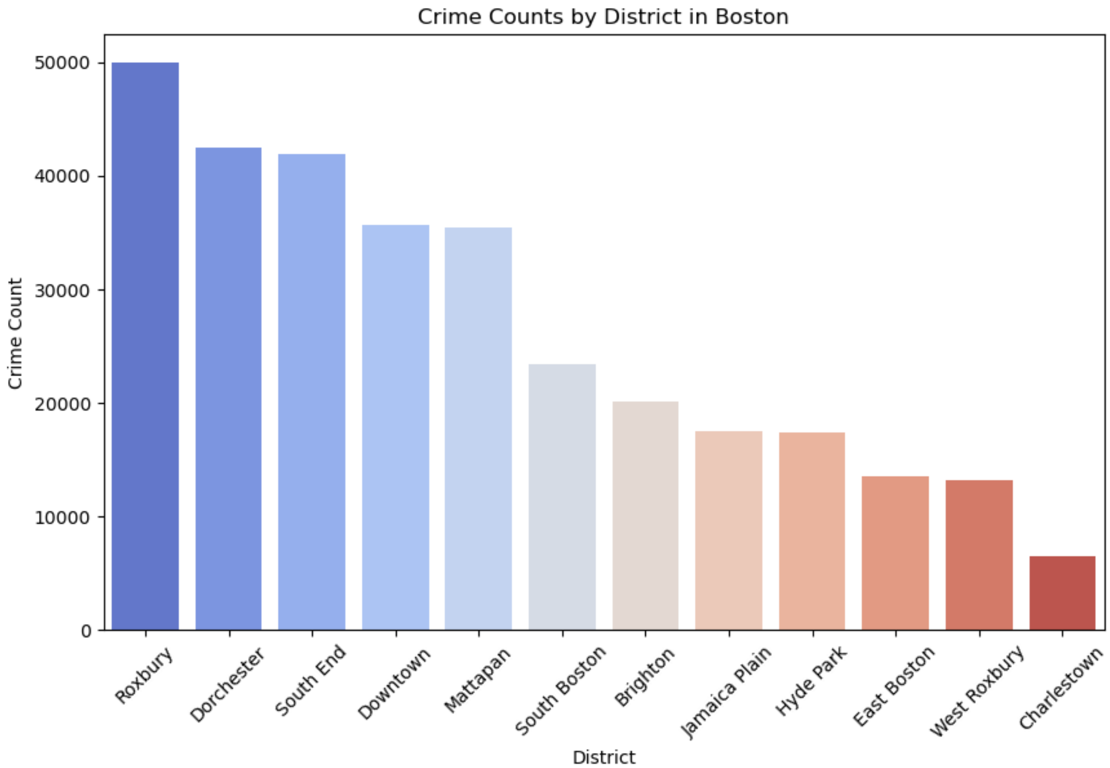
[30]:
```
# Replace district codes with real names without inplace=True
crime_df['DISTRICT'] = crime_df['DISTRICT'].replace({
    'A1': 'Downtown', 'A15': 'Charlestown', 'A7': 'East Boston', 'B2':␣
↪'Roxbury', 'B3': 'Mattapan',
    'C6': 'South Boston', 'C11': 'Dorchester', 'D4': 'South End', 'D14':␣
↪'Brighton', 'E5': 'West Roxbury',
    'E13': 'Jamaica Plain', 'E18': 'Hyde Park'
})
```

```python
# Calculate crime counts by district
district_crime_counts = crime_df['DISTRICT'].value_counts()

# Print the crime counts by district
print("Crime counts by district:")
print(district_crime_counts)

# Visualize with bar plot and avoid deprecation by adding 'hue'
plt.figure(figsize=(10, 6))
sns.barplot(x=district_crime_counts.index, y=district_crime_counts.values,␣
↪hue=district_crime_counts.index, palette='coolwarm', legend=False)
plt.title("Crime Counts by District in Boston")
plt.xlabel('District')
plt.ylabel('Crime Count')
plt.xticks(rotation=45)
plt.show()
```

```
Crime counts by district:
DISTRICT
Roxbury          49945
Dorchester       42530
South End        41915
Downtown         35717
Mattapan         35442
South Boston     23460
Brighton         20127
Jamaica Plain    17536
Hyde Park        17348
East Boston      13544
West Roxbury     13239
Charlestown       6505
Name: count, dtype: int64
```



Based on the result of Q 14, Which district in boston is the most dangerous district and which one is the safest? Answer: The most dangerous district is — Roxbury The safest district is — Charlestown