Environment Setup

Importing all required libraries that going to be used in the script.

In [1]:

```
import pandas as pd
import os
import csv
import numpy as np
import matplotlib.pyplot as plt
import statsmodels.api as sm
import seaborn as sns
sns.set
```

Out[1]:

```
<function seaborn.rcmod.set(*args, **kwargs)>
```

Helper Functions

All the functions that perform helping tasks with the code

In [2]:

```
# This function removes .csv from file names

def remove_csv_filename(file_name):
    return file_name.split('.')[0]
```

In [3]:

```
# This function merges csv files in a given directory
def merge_files():
    directory_path = "DataSet/"
    list_of_years = os.listdir(directory_path)
    merged_data = pd.DataFrame()
    for year in list_of_years:
      # we have two directories as 2014 and 2018, we want to loop for each year
        data_directory = directory_path + year
        for file_name in os.listdir(data_directory):
          # since year and month info is in the name of the file we split the name and ge
            year = file_name.split("_")[4].lower()
            month = file_name.split("_")[3].lower()
            with open(os.path.join(data_directory, file_name), "r") as csv_file:
                # loading the contents of the csv file
                data = pd.read_csv(csv_file)
                # creating new columns year and month and populating them
                data["year"] = year.replace(".csv","")
                data["month"] = month
                # pushing all the contents in the merged data frame
                merged_data = pd.concat([merged_data, data], ignore_index=True)

    return merged_data
```

In [4]:

```
# this function removes given columns from the data.
def remove_columns(column_name_with_string, data):
    delete_columns = [col for col in data.columns if column_name_with_string in col]
    # get a list of columns to delete
    data.drop(columns=delete_columns, inplace=True)

    return data
```

In [5]:

```
# This function renames the columns to make it more programming friendly
# e.g Name of County
def rename_columns(data):
  data.columns = map(str.lower, data.columns)
  data.rename(columns={'unnamed: 0':'county','number of homicide convictions':'successful
       'number of homicide unsuccessful':'unsuccessful_homicide',
       'number of offences against the person convictions':'successful_against_person',
       'number of offences against the person unsuccessful':'unsuccessful_against_person'
       'number of sexual offences convictions':'successful_sexual_offences',
       'number of sexual offences unsuccessful':'unsuccessful_sexual_offences',
       'number of burglary convictions':'successful_burglary', 'number of burglary unsucc
       'number of robbery convictions':'successful_robbery', 'number of robbery unsuccess
       'number of theft and handling convictions':'successful_theft',
       'number of theft and handling unsuccessful':'unsuccessful_theft',
       'number of fraud and forgery convictions':'successful_fraud',
       'number of fraud and forgery unsuccessful':'unsuccessful_fraud',
       'number of criminal damage convictions':'successful_criminal_damage',
       'number of criminal damage unsuccessful':'unsuccessful_criminal_damage',
       'number of drugs offences convictions':'successful_drugs',
       'number of drugs offences unsuccessful':'unsuccessful_drugs',
       'number of public order offences convictions':'successful_public_order',
       'number of public order offences unsuccessful':'unsuccessful_public_order',
       'number of all other offences (excluding motoring) convictions':'successful_other'
       'number of all other offences (excluding motoring) unsuccessful':'unsuccessful_oth
       'number of motoring offences convictions':'successful_motoring',
       'number of motoring offences unsuccessful':'unsuccessful_motoring',
       'number of admin finalised unsuccessful':'unsuccessful_admin'}, inplace=True)
  return data
```

In [6]:

```
merged_data = merge_files()
```

In [7]:

```
print(merged_data)
```

```
            Unnamed: 0  Number of Homicide Convictions  \
0             National                              81
1     Avon and Somerset                              1
2          Bedfordshire                              0
3        Cambridgeshire                              0
4             Cheshire                               1
..                 ...                             ...
855        Warwickshire                              0
856         West Mercia                              6
857       West Midlands                             11
858      West Yorkshire                              5
859           Wiltshire                              0

    Percentage of Homicide Convictions  Number of Homicide Unsuccessful  \
0                                85.3%                               14
1                               100.0%                                0
2                                    -                                0
3                                    -                                0
4                                50.0%                                1
..                                 ...                              ...
855                                  -                                0
856                             100.0%                                0
857                              91.7%                                1
858                              71.4%                                2
859                                  -                                0

    Percentage of Homicide Unsuccessful  \
0                                 14.7%
1                                  0.0%
2                                     -
3                                     -
4                                 50.0%
..                                  ...
855                                   -
856                                0.0%
857                                8.3%
858                               28.6%
859                                   -

    Number of Offences Against The Person Convictions  \
0                                               7,805
1                                                 167
2                                                  69
3                                                  99
4                                                 140
..                                                ...
855                                                65
856                                               220
857                                               609
858                                               446
859                                                85

    Percentage of Offences Against The Person Convictions  \
0                                               74.1%
1                                               78.8%
2                                               75.0%
3                                               81.1%
4                                               74.9%
..                                                ...
855                                             80.2%
856                                             78.6%
```

```
857                                        78.0%
858                                        85.9%
859                                        86.7%

    Number of Offences Against The Person Unsuccessful  \
0                                          2,722
1                                             45
2                                             23
3                                             23
4                                             47
..                                           ...
855                                           16
856                                           60
857                                          172
858                                           73
859                                           13

    Percentage of Offences Against The Person Unsuccessful  \
0                                          25.9%
1                                          21.2%
2                                          25.0%
3                                          18.9%
4                                          25.1%
..                                           ...
855                                        19.8%
856                                        21.4%
857                                        22.0%
858                                        14.1%
859                                        13.3%

    Number of Sexual Offences Convictions  ...  \
0                                     698  ...
1                                      36  ...
2                                       5  ...
3                                       6  ...
4                                      17  ...
..                                    ...  ...
855                                     9  ...
856                                    20  ...
857                                    66  ...
858                                    71  ...
859                                    12  ...

    Number of All Other Offences (excluding Motoring) Unsuccessful  \
0                                            513
1                                             16
2                                              6
3                                              2
4                                              6
..                                           ...
855                                            0
856                                            2
857                                           15
858                                            8
859                                            1

     Percentage of All Other Offences (excluding Motoring) Unsuccessful  \
0                                          16.3%
1                                          19.5%
2                                          35.3%
3                                          25.0%
```

```
4                                            10.7%
..                                             ...
855                                           0.0%
856                                          15.4%
857                                          17.9%
858                                          25.0%
859                                          12.5%

    Number of Motoring Offences Convictions  \
0                                      8,283
1                                        188
2                                         40
3                                         79
4                                        209
..                                       ...
855                                       78
856                                      190
857                                      280
858                                      236
859                                       64

    Percentage of Motoring Offences Convictions  \
0                                          86.3%
1                                          83.6%
2                                          88.9%
3                                          92.9%
4                                          94.6%
..                                           ...
855                                        78.8%
856                                        87.6%
857                                        80.5%
858                                        91.8%
859                                        97.0%

    Number of Motoring Offences Unsuccessful  \
0                                       1,314
1                                          37
2                                           5
3                                           6
4                                          12
..                                        ...
855                                        21
856                                        27
857                                        68
858                                        21
859                                         2

     Percentage of Motoring Offences Unsuccessful  \
0                                           13.7%
1                                           16.4%
2                                           11.1%
3                                            7.1%
4                                            5.4%
..                                            ...
855                                         21.2%
856                                         12.4%
857                                         19.5%
858                                          8.2%
859                                          3.0%

    Number of Admin Finalised Unsuccessful  \
```

```
0                                      718
1                                       24
2                                       16
3                                        4
4                                        1
..                                     ...
855                                     11
856                                     12
857                                     92
858                                     51
859                                      4

     Percentage of L Motoring Offences Unsuccessful  year      month
0                                            100.0%  2014      april
1                                            100.0%  2014      april
2                                            100.0%  2014      april
3                                            100.0%  2014      april
4                                            100.0%  2014      april
..                                              ...   ...        ...
855                                          100.0%  2018  september
856                                          100.0%  2018  september
857                                          100.0%  2018  september
858                                          100.0%  2018  september
859                                          100.0%  2018  september

[860 rows x 53 columns]
```

In [8]:

```
merged_data.to_csv("merged_file.csv", index=False)
```

Exploring Merged Data

In [9]:

```
merged_data.head()
```

Out[9]:

	Unnamed: 0	Number of Homicide Convictions	Percentage of Homicide Convictions	Number of Homicide Unsuccessful	Percentage of Homicide Unsuccessful	Number of Offences Against The Person Convictions	Perce of Offe Agains P Convic
0	National	81	85.3%	14	14.7%	7,805	7
1	Avon and Somerset	1	100.0%	0	0.0%	167	7
2	Bedfordshire	0	-	0	-	69	7
3	Cambridgeshire	0	-	0	-	99	8
4	Cheshire	1	50.0%	1	50.0%	140	7

5 rows × 53 columns

In [10]:

```
merged_data.shape
```

Out[10]:

```
(860, 53)
```

In [11]:

```
merged_data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 860 entries, 0 to 859
Data columns (total 53 columns):
 #   Column                                                    N
on-Null Count  Dtype
---  ------                                                    -
-------------  -----
 0   Unnamed: 0                                                8
60 non-null    object
 1   Number of Homicide Convictions                            8
60 non-null    int64
 2   Percentage of Homicide Convictions                        8
60 non-null    object
 3   Number of Homicide Unsuccessful                           8
60 non-null    int64
 4   Percentage of Homicide Unsuccessful                       8
60 non-null    object
 5   Number of Offences Against The Person Convictions         8
60 non-null    object
 6   Percentage of Offences Against The Person Convictions     8
60 non-null    object
 7   Number of Offences Against The Person Unsuccessful        8
60 non-null    object
 8   Percentage of Offences Against The Person Unsuccessful    8
60 non-null    object
 9   Number of Sexual Offences Convictions                     8
60 non-null    object
 10  Percentage of Sexual Offences Convictions                 8
60 non-null    object
 11  Number of Sexual Offences Unsuccessful                    8
60 non-null    int64
 12  Percentage of Sexual Offences Unsuccessful                8
60 non-null    object
 13  Number of Burglary Convictions                            8
60 non-null    object
 14  Percentage of Burglary Convictions                        8
60 non-null    object
 15  Number of Burglary Unsuccessful                           8
60 non-null    int64
 16  Percentage of Burglary Unsuccessful                       8
60 non-null    object
 17  Number of Robbery Convictions                             8
60 non-null    int64
 18  Percentage of Robbery Convictions                         8
60 non-null    object
 19  Number of Robbery Unsuccessful                            8
60 non-null    int64
 20  Percentage of Robbery Unsuccessful                        8
60 non-null    object
 21  Number of Theft And Handling Convictions                  8
60 non-null    object
 22  Percentage of Theft And Handling Convictions              8
60 non-null    object
 23  Number of Theft And Handling Unsuccessful                 8
60 non-null    object
 24  Percentage of Theft And Handling Unsuccessful             8
60 non-null    object
 25  Number of Fraud And Forgery Convictions                   8
60 non-null    int64
 26  Percentage of Fraud And Forgery Convictions               8
60 non-null    object
```

```
 27  Number of Fraud And Forgery Unsuccessful                           8
60 non-null    int64
 28  Percentage of Fraud And Forgery Unsuccessful                       8
60 non-null    object
 29  Number of Criminal Damage Convictions                              8
60 non-null    object
 30  Percentage of Criminal Damage Convictions                          8
60 non-null    object
 31  Number of Criminal Damage Unsuccessful                             8
60 non-null    int64
 32  Percentage of Criminal Damage Unsuccessful                         8
60 non-null    object
 33  Number of Drugs Offences Convictions                               8
60 non-null    object
 34  Percentage of Drugs Offences Convictions                           8
60 non-null    object
 35  Number of Drugs Offences Unsuccessful                              8
60 non-null    int64
 36  Percentage of Drugs Offences Unsuccessful                          8
60 non-null    object
 37  Number of Public Order Offences Convictions                        8
60 non-null    object
 38  Percentage of Public Order Offences Convictions                    8
60 non-null    object
 39  Number of Public Order Offences Unsuccessful                       8
60 non-null    int64
 40  Percentage of Public Order Offences Unsuccessful                   8
60 non-null    object
 41  Number of All Other Offences (excluding Motoring) Convictions      8
60 non-null    object
 42  Percentage of All Other Offences (excluding Motoring) Convictions  8
60 non-null    object
 43  Number of All Other Offences (excluding Motoring) Unsuccessful     8
60 non-null    int64
 44  Percentage of All Other Offences (excluding Motoring) Unsuccessful 8
60 non-null    object
 45  Number of Motoring Offences Convictions                            8
60 non-null    object
 46  Percentage of Motoring Offences Convictions                        8
60 non-null    object
 47  Number of Motoring Offences Unsuccessful                           8
60 non-null    object
 48  Percentage of Motoring Offences Unsuccessful                       8
60 non-null    object
 49  Number of Admin Finalised Unsuccessful                             8
60 non-null    int64
 50  Percentage of L Motoring Offences Unsuccessful                     8
60 non-null    object
 51  year                                                               8
60 non-null    object
 52  month                                                              8
60 non-null    object
dtypes: int64(13), object(40)
memory usage: 356.2+ KB
```

In [12]:

```
merged_data.describe(include = 'all')
```

Out[12]:

	Unnamed: 0	Number of Homicide Convictions	Percentage of Homicide Convictions	Number of Homicide Unsuccessful	Percentage of Homicide Unsuccessful	Number of Offences Against The Person Convictions	Perc of Of Agair I Conv
count	860	860.000000	860	860.000000	860	860	
unique	43	NaN	60	NaN	60	365	
top	National	NaN	100.0%	NaN	0.0%	83	
freq	20	NaN	345	NaN	345	10	
mean	NaN	3.404651	NaN	0.818605	NaN	NaN	
std	NaN	11.657370	NaN	2.995088	NaN	NaN	
min	NaN	0.000000	NaN	0.000000	NaN	NaN	
25%	NaN	0.000000	NaN	0.000000	NaN	NaN	
50%	NaN	1.000000	NaN	0.000000	NaN	NaN	
75%	NaN	2.000000	NaN	0.000000	NaN	NaN	
max	NaN	125.000000	NaN	32.000000	NaN	NaN	

11 rows × 53 columns

Data Cleaning

After looking at the data we realized it needs cleaning before we proceed. Percentage columns are needed to be removed as they can be calculated any time with the help of existing columns. Then data needed to converted into correct format before proceeding

In [13]:

```
data = remove_columns('Percentage', merged_data)
```

In [14]:

```
data = rename_columns(data)
```

In [15]:

```
# The National rows are just totals of the columns per month and therefore we will drop t
data = data.drop(data[data['county'] == 'National'].index)
```

In [16]:

```
# Checking data types for all columns.
# Mixed data types object and int the best is to convert all to interger
print (data.dtypes)
```

```
county                          object
successful_homicide              int64
unsuccessful_homicide            int64
successful_against_person       object
unsuccessful_against_person     object
successful_sexual_offences      object
unsuccessful_sexual_offences     int64
successful_burglary             object
unsuccessful_burglary            int64
successful_robbery               int64
unsuccessful_robbery             int64
successful_theft                object
unsuccessful_theft              object
successful_fraud                 int64
unsuccessful_fraud               int64
successful_criminal_damage      object
unsuccessful_criminal_damage     int64
successful_drugs                object
unsuccessful_drugs               int64
successful_public_order         object
unsuccessful_public_order        int64
successful_other                object
unsuccessful_other               int64
successful_motoring             object
unsuccessful_motoring           object
unsuccessful_admin               int64
year                            object
month                           object
dtype: object
```

To change all data types with object to integer type

In [17]:

```
print(data.columns)
```

```
Index(['county', 'successful_homicide', 'unsuccessful_homicide',
       'successful_against_person', 'unsuccessful_against_person',
       'successful_sexual_offences', 'unsuccessful_sexual_offences',
       'successful_burglary', 'unsuccessful_burglary', 'successful_robber
y',
       'unsuccessful_robbery', 'successful_theft', 'unsuccessful_theft',
       'successful_fraud', 'unsuccessful_fraud', 'successful_criminal_dama
ge',
       'unsuccessful_criminal_damage', 'successful_drugs',
       'unsuccessful_drugs', 'successful_public_order',
       'unsuccessful_public_order', 'successful_other', 'unsuccessful_othe
r',
       'successful_motoring', 'unsuccessful_motoring', 'unsuccessful_admi
n',
       'year', 'month'],
      dtype='object')
```

In [18]:

```
# convert object columns to numeric type with NaN values for non-numeric values
data[['successful_homicide', 'unsuccessful_homicide', 'successful_against_person',
       'unsuccessful_against_person', 'successful_sexual_offences',
       'unsuccessful_sexual_offences', 'successful_burglary', 'unsuccessful_burglary',
       'successful_robbery', 'unsuccessful_robbery', 'successful_theft',
       'unsuccessful_theft', 'successful_fraud',
       'unsuccessful_fraud', 'successful_criminal_damage',
       'unsuccessful_criminal_damage', 'successful_drugs', 'unsuccessful_drugs',
       'successful_public_order', 'unsuccessful_public_order', 'successful_other',
       'unsuccessful_other', 'successful_motoring', 'unsuccessful_motoring',
       'unsuccessful_admin',]] = data[['successful_homicide', 'unsuccessful_homicide', 's
       'unsuccessful_against_person', 'successful_sexual_offences',
       'unsuccessful_sexual_offences', 'successful_burglary', 'unsuccessful_burglary',
       'successful_robbery', 'unsuccessful_robbery', 'successful_theft',
       'unsuccessful_theft', 'successful_fraud',
       'unsuccessful_fraud', 'successful_criminal_damage',
       'unsuccessful_criminal_damage', 'successful_drugs', 'unsuccessful_drugs',
       'successful_public_order', 'unsuccessful_public_order', 'successful_other',
       'unsuccessful_other', 'successful_motoring', 'unsuccessful_motoring',
       'unsuccessful_admin']].apply(pd.to_numeric, errors='coerce')
```

In [19]:

```
# replace NaN values with a default value (e.g., 0)
data[['successful_homicide', 'unsuccessful_homicide', 'successful_against_person',
       'unsuccessful_against_person', 'successful_sexual_offences',
       'unsuccessful_sexual_offences', 'successful_burglary', 'unsuccessful_burglary',
       'successful_robbery', 'unsuccessful_robbery', 'successful_theft',
       'unsuccessful_theft', 'successful_fraud',
       'unsuccessful_fraud', 'successful_criminal_damage',
       'unsuccessful_criminal_damage', 'successful_drugs', 'unsuccessful_drugs',
       'successful_public_order', 'unsuccessful_public_order', 'successful_other',
       'unsuccessful_other', 'successful_motoring', 'unsuccessful_motoring',
       'unsuccessful_admin']] = data[['successful_homicide', 'unsuccessful_homicide', 'su
       'unsuccessful_against_person', 'successful_sexual_offences',
       'unsuccessful_sexual_offences', 'successful_burglary', 'unsuccessful_burglary',
       'successful_robbery', 'unsuccessful_robbery', 'successful_theft',
       'unsuccessful_theft', 'successful_fraud',
       'unsuccessful_fraud', 'successful_criminal_damage',
       'unsuccessful_criminal_damage', 'successful_drugs', 'unsuccessful_drugs',
       'successful_public_order', 'unsuccessful_public_order', 'successful_other',
       'unsuccessful_other', 'successful_motoring', 'unsuccessful_motoring',
       'unsuccessful_admin']].fillna(0).astype('int64')
```

In [20]:

```
# Check changes
print (data.dtypes)
```

```
county                           object
successful_homicide               int64
unsuccessful_homicide             int64
successful_against_person         int64
unsuccessful_against_person       int64
successful_sexual_offences        int64
unsuccessful_sexual_offences      int64
successful_burglary               int64
unsuccessful_burglary             int64
successful_robbery                int64
unsuccessful_robbery              int64
successful_theft                  int64
unsuccessful_theft                int64
successful_fraud                  int64
unsuccessful_fraud                int64
successful_criminal_damage        int64
unsuccessful_criminal_damage      int64
successful_drugs                  int64
unsuccessful_drugs                int64
successful_public_order           int64
unsuccessful_public_order         int64
successful_other                  int64
unsuccessful_other                int64
successful_motoring               int64
unsuccessful_motoring             int64
unsuccessful_admin                int64
year                             object
month                            object
dtype: object
```

In [21]:

```
# Checking for sum of missing values per column
print(data.isnull().sum())
```

```
county                           0
successful_homicide              0
unsuccessful_homicide            0
successful_against_person        0
unsuccessful_against_person      0
successful_sexual_offences       0
unsuccessful_sexual_offences     0
successful_burglary              0
unsuccessful_burglary            0
successful_robbery               0
unsuccessful_robbery             0
successful_theft                 0
unsuccessful_theft               0
successful_fraud                 0
unsuccessful_fraud               0
successful_criminal_damage       0
unsuccessful_criminal_damage     0
successful_drugs                 0
unsuccessful_drugs               0
successful_public_order          0
unsuccessful_public_order        0
successful_other                 0
unsuccessful_other               0
successful_motoring              0
unsuccessful_motoring            0
unsuccessful_admin               0
year                             0
month                            0
dtype: int64
```

In [22]:

```
# Obtaining a list of unique values in month column
data.month.unique()
```

Out[22]:

```
array(['april', 'august', 'december', 'february', 'january', 'july',
       'june', 'march', 'may', 'november', 'october', 'september'],
      dtype=object)
```

In [23]:

```
# Obtaining a list of unique values in year column
data.year.unique()
```

Out[23]:

```
array(['2014', '2018'], dtype=object)
```

In [24]:

```
# reorder columns to place year and month column to index 0 and 1 respectively
cols = list(data.columns)
cols.insert(0, cols.pop(cols.index('year')))
cols.insert(1, cols.pop(cols.index('month')))
data = data.loc[:, cols]
```

In [25]:

```
# print dataframe
print(data.head())
```

```
   year  month             county  successful_homicide  unsuccessful_homic
ide  \
1  2014  april  Avon and Somerset                    1                    
0  
2  2014  april        Bedfordshire                    0                    
0  
3  2014  april     Cambridgeshire                    0                    
0  
4  2014  april           Cheshire                    1                    
1  
5  2014  april          Cleveland                    0                    
0  

   successful_against_person  unsuccessful_against_person  \
1                        167                           45   
2                         69                           23   
3                         99                           23   
4                        140                           47   
5                         85                           41   

   successful_sexual_offences  unsuccessful_sexual_offences  \
1                          36                             8   
2                           5                             1   
3                           6                             3   
4                          17                             3   
5                          11                             4   

   successful_burglary  ...  unsuccessful_criminal_damage  successful_drug
s  \
1                   37  ...                             6               13
5   
2                   16  ...                             6                4
5   
3                    8  ...                             1                4
0   
4                   26  ...                             9                7
5   
5                   25  ...                             8                6
3   

   unsuccessful_drugs  successful_public_order  unsuccessful_public_order
\
1                   2                       68                         11
2                   2                       29                          6
3                   2                       45                          9
4                  10                       86                          7
5                   7                       74                         27

   successful_other  unsuccessful_other  successful_motoring  \
1                66                  16                  188   
2                11                   6                   40   
3                 6                   2                   79   
4                50                   6                  209   
5                28                   5                  124   

   unsuccessful_motoring  unsuccessful_admin  
1                     37                  24  
2                      5                  16  
3                      6                   4  
4                     12                   1  
5                     17                  10  
```

```
[5 rows x 28 columns]
```

Exploring Data

After cleaning we again need to look at the data and make sure everything is in correct order

In [26]:

```
data.describe(include ='all')
```

Out[26]:

	year	month	county	successful_homicide	unsuccessful_homicide	successful_agai
count	840	840	840	840.000000	840.000000	
unique	2	12	42	NaN	NaN	
top	2014	august	Avon and Somerset	NaN	NaN	
freq	504	84	20	NaN	NaN	
mean	NaN	NaN	NaN	1.742857	0.419048	
std	NaN	NaN	NaN	2.998700	1.197183	
min	NaN	NaN	NaN	0.000000	0.000000	
25%	NaN	NaN	NaN	0.000000	0.000000	
50%	NaN	NaN	NaN	1.000000	0.000000	
75%	NaN	NaN	NaN	2.000000	0.000000	
max	NaN	NaN	NaN	29.000000	11.000000	

11 rows × 28 columns

In [27]:

```
data.describe()
```

Out[27]:

	successful_homicide	unsuccessful_homicide	successful_against_person	unsuccessful_
count	840.000000	840.000000	840.000000	
mean	1.742857	0.419048	184.514286	
std	2.998700	1.197183	116.608510	
min	0.000000	0.000000	0.000000	
25%	0.000000	0.000000	100.750000	
50%	1.000000	0.000000	158.000000	
75%	2.000000	0.000000	242.000000	
max	29.000000	11.000000	688.000000	

8 rows × 25 columns

In [28]:

```
# Exclude the "year" column from consideration
columns_to_exclude = ["year"]
filtered_columns = [col for col in data.columns if col not in columns_to_exclude]
```

In [29]:

```
# Calculate the mean of each column excluding the "year" column
column_means = data[filtered_columns].mean()
```

```
C:\Users\s4215274\AppData\Local\Temp\ipykernel_35152\2990723375.py:2: Futu
reWarning: Dropping of nuisance columns in DataFrame reductions (with 'num
eric_only=None') is deprecated; in a future version this will raise TypeEr
ror.  Select only valid columns before calling the reduction.
  column_means = data[filtered_columns].mean()
```

In [30]:

```
# Find the column with the highest mean
column_with_highest_mean = column_means.idxmax()

print("Column with the highest mean (excluding 'year'): ", column_with_highest_mean)
```

```
Column with the highest mean (excluding 'year'):  successful_against_perso
n
```

Observations

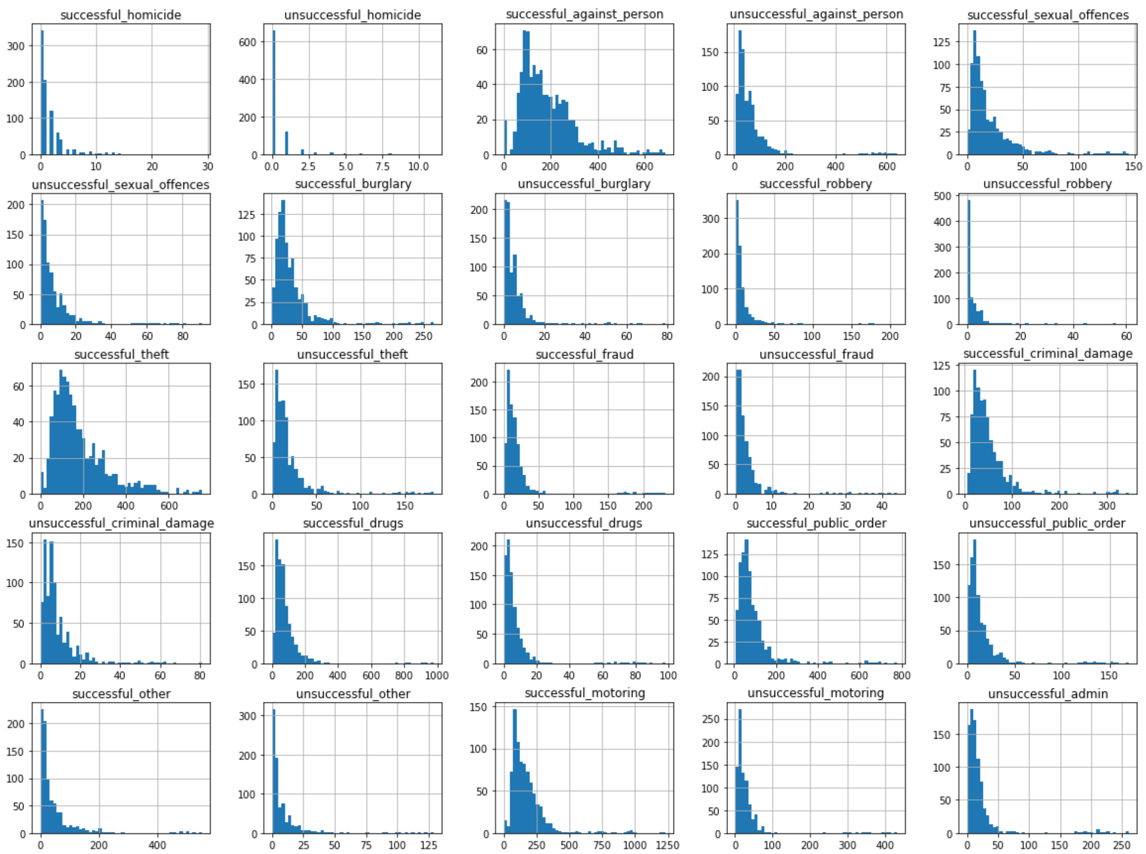
There is a very high percentage ratio of unsuccessful convictions to successful convictions for crimes against person of about 93.31% The unique value of counties shows that data has been taken in 42 counties. The successful convictions for crimes against person have the highest average of 184.51 and high standard deviation of 116.61 meaning the values in the column are more spread out or dispersed from the mean.

In [31]:

```
data.hist(bins =50, figsize = (20,15))
```

Out[31]:

```
array([[<AxesSubplot:title={'center':'successful_homicide'}>,
        <AxesSubplot:title={'center':'unsuccessful_homicide'}>,
        <AxesSubplot:title={'center':'successful_against_person'}>,
        <AxesSubplot:title={'center':'unsuccessful_against_person'}>,
        <AxesSubplot:title={'center':'successful_sexual_offences'}>],
       [<AxesSubplot:title={'center':'unsuccessful_sexual_offences'}>,
        <AxesSubplot:title={'center':'successful_burglary'}>,
        <AxesSubplot:title={'center':'unsuccessful_burglary'}>,
        <AxesSubplot:title={'center':'successful_robbery'}>,
        <AxesSubplot:title={'center':'unsuccessful_robbery'}>],
       [<AxesSubplot:title={'center':'successful_theft'}>,
        <AxesSubplot:title={'center':'unsuccessful_theft'}>,
        <AxesSubplot:title={'center':'successful_fraud'}>,
        <AxesSubplot:title={'center':'unsuccessful_fraud'}>,
        <AxesSubplot:title={'center':'successful_criminal_damage'}>],
       [<AxesSubplot:title={'center':'unsuccessful_criminal_damage'}>,
        <AxesSubplot:title={'center':'successful_drugs'}>,
        <AxesSubplot:title={'center':'unsuccessful_drugs'}>,
        <AxesSubplot:title={'center':'successful_public_order'}>,
        <AxesSubplot:title={'center':'unsuccessful_public_order'}>],
       [<AxesSubplot:title={'center':'successful_other'}>,
        <AxesSubplot:title={'center':'unsuccessful_other'}>,
        <AxesSubplot:title={'center':'successful_motoring'}>,
        <AxesSubplot:title={'center':'unsuccessful_motoring'}>,
        <AxesSubplot:title={'center':'unsuccessful_admin'}>]],
      dtype=object)
```



Attributes are scaled differently therefore we might need to consider normalisation Some of the variables have a right skewed distribution which means we might need to take into consideration some outliers eg sexual_offences, criminal_damage, against person and theft

In [32]:

```
# Create a new DataFrame with the totals of successful crimes per crime type
crime_totals = data.groupby([
    'successful_homicide', 'successful_against_person', 'successful_sexual_offences', 'su
    'successful_theft', 'successful_fraud', 'successful_criminal_damage', 'successful_dru
    'successful_public_order', 'successful_other', 'successful_motoring'
])[['unsuccessful_homicide', 'unsuccessful_against_person', 'unsuccessful_sexual_offences
    'unsuccessful_burglary', 'unsuccessful_robbery', 'unsuccessful_theft',
    'unsuccessful_fraud', 'unsuccessful_criminal_damage', 'unsuccessful_drugs',
    'unsuccessful_public_order', 'unsuccessful_other']].sum().reset_index()

# Create a correlation matrix
corr_matrix = crime_totals.corr()

print(corr_matrix)
```

```
                              successful_homicide  successful_against_pe
rson  \
successful_homicide                      1.000000                   0.14
4947
successful_against_person                0.144947                   1.00
0000
successful_sexual_offences               0.712174                   0.35
0218
successful_burglary                      0.678108                   0.24
7621
successful_robbery                       0.702485                   0.10
4757
successful_theft                         0.238092                   0.63
4617
successful_fraud                         0.731843                   0.01
3358
successful_criminal_damage               0.639470                   0.33
1396
successful_drugs                         0.469637                   0.27
```

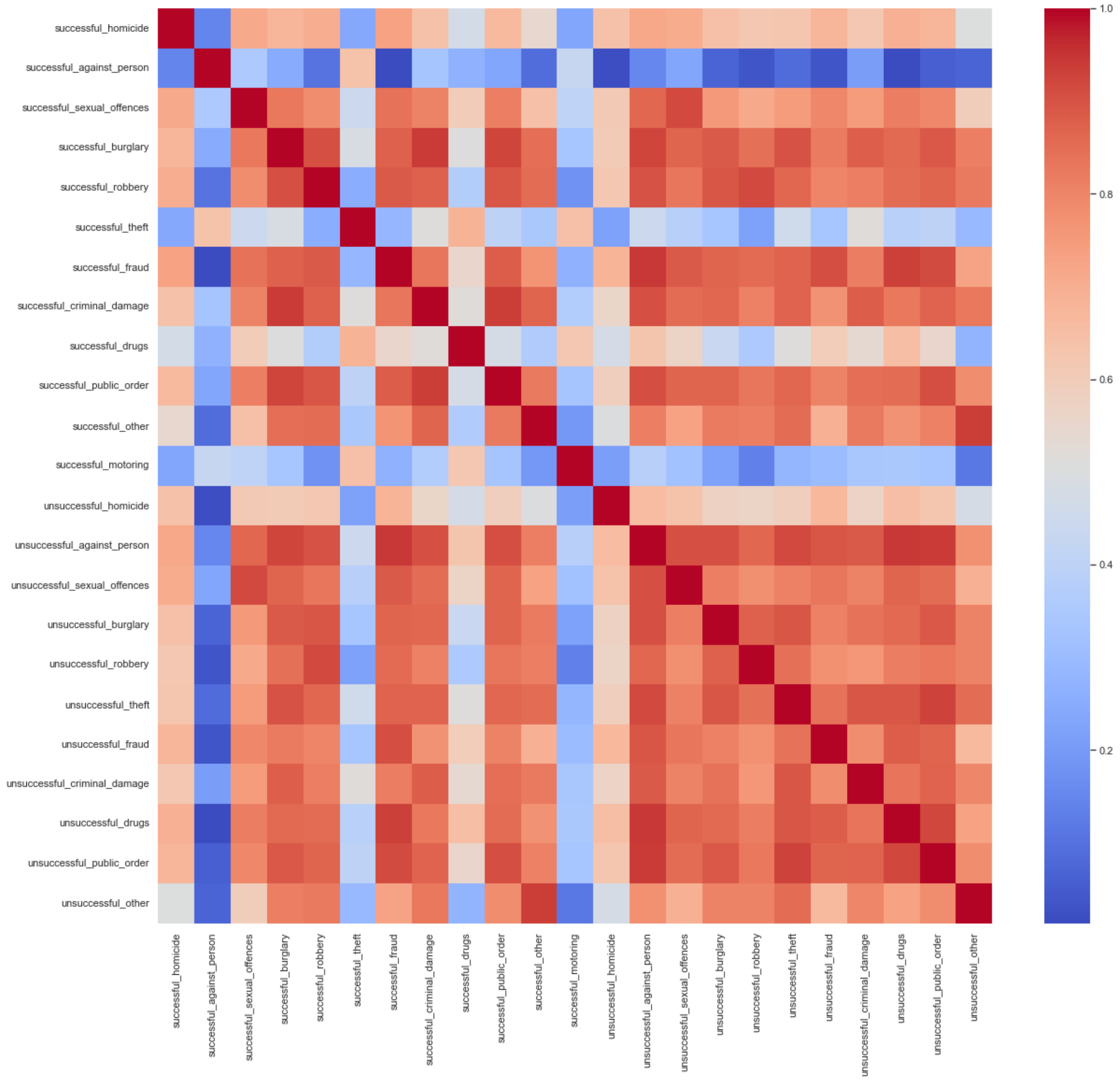
In [33]:

```
min_corr= min(crime_totals.corr())
print(min_corr)
```

```
successful_against_person
```

In [34]:

```
# Create the heatmap using Seaborn
sns.set (rc = {'figure.figsize':(20, 18)})
sns.heatmap(corr_matrix, cmap='coolwarm', annot=False)

plt.show()
```

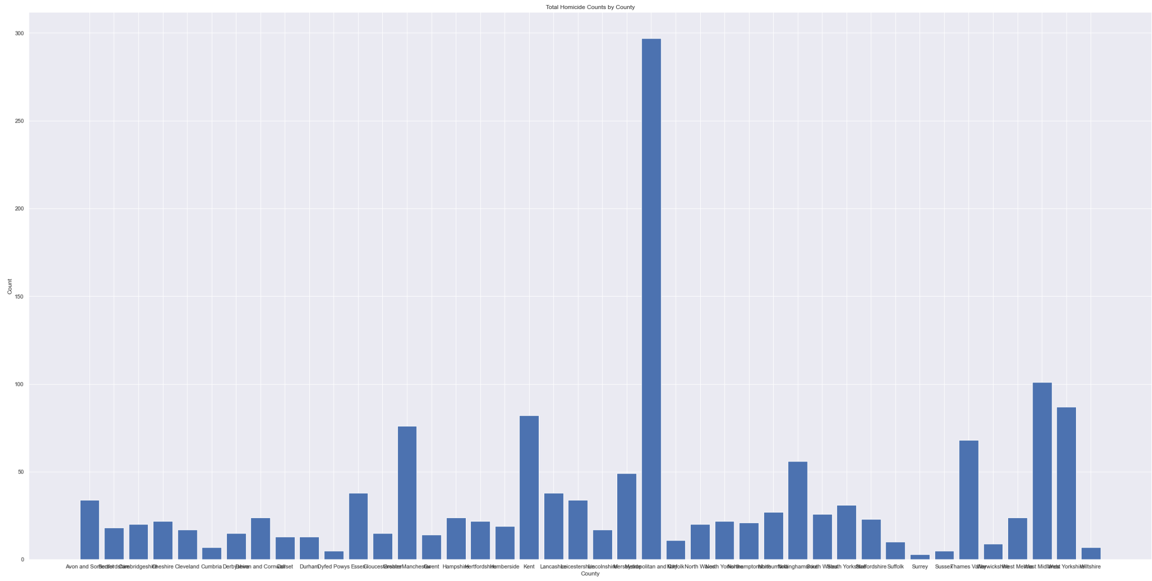


In [35]:

```
# Subset the data to only include the columns of interest
cols_of_interest = ['county', 'successful_homicide', 'unsuccessful_homicide',
       'successful_against_person', 'unsuccessful_against_person',
       'successful_sexual_offences', 'unsuccessful_sexual_offences',
       'successful_burglary', 'unsuccessful_burglary', 'successful_robbery',
       'unsuccessful_robbery', 'successful_theft', 'unsuccessful_theft',
       'successful_fraud', 'unsuccessful_fraud', 'successful_criminal_damage',
       'unsuccessful_criminal_damage', 'successful_drugs',
       'unsuccessful_drugs', 'successful_public_order',
       'unsuccessful_public_order', 'successful_other', 'unsuccessful_other',
       'successful_motoring', 'unsuccessful_motoring', 'unsuccessful_admin',
       'year', 'month']

df_subset = data[cols_of_interest]
```

In [36]:

```
county_counts = df_subset.groupby('county').sum().reset_index()
plt.figure(figsize=(40, 20))
plt.bar(county_counts['county'], county_counts['successful_homicide'])
plt.title('Total Homicide Counts by County')
plt.xlabel('County')
plt.ylabel('Count')
plt.show()
```

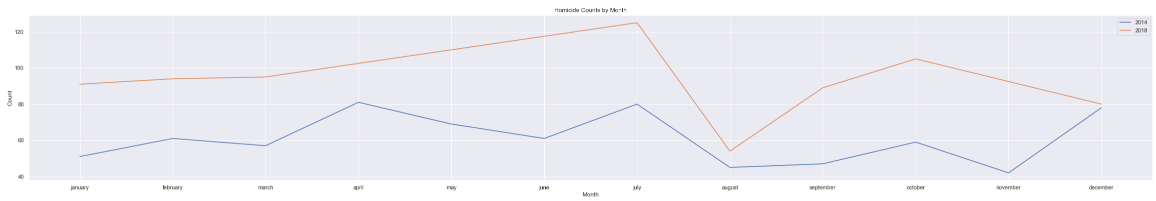


In [ ]:

```
```

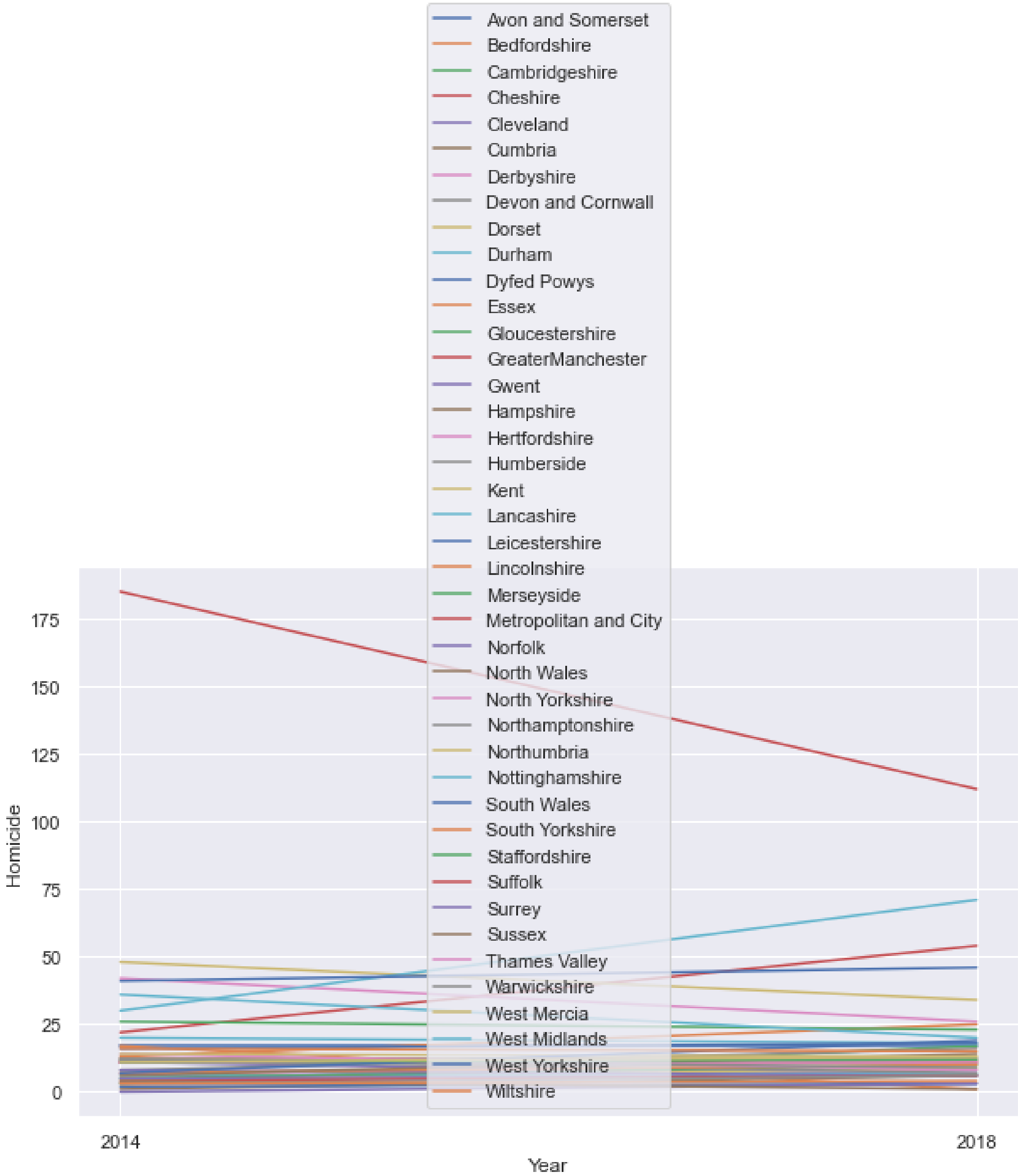
In [37]:

```
df_subset_2014 = df_subset[df_subset['year'] == '2014']
df_subset_2018 = df_subset[df_subset['year'] == '2018']
monthly_counts_2014 = df_subset_2014.groupby(['year', 'month']).sum().reset_index()
monthly_counts_2018 = df_subset_2018.groupby(['year', 'month']).sum().reset_index()

# Convert month values to datetime objects
monthly_counts_2014['date'] = pd.to_datetime(monthly_counts_2014['year'].astype(str) + '-
monthly_counts_2018['date'] = pd.to_datetime(monthly_counts_2018['year'].astype(str) + '-

# Sort the DataFrame based on the month values
monthly_counts_2014 = monthly_counts_2014.sort_values('date')
monthly_counts_2018 = monthly_counts_2018.sort_values('date')


fig, ax = plt.subplots(figsize=(40, 6))
# Plot data for 2014
ax.plot(monthly_counts_2014['month'].astype(str),
        monthly_counts_2014['successful_homicide'], label='2014')

# Plot data for 2018
ax.plot(monthly_counts_2018['month'].astype(str),
        monthly_counts_2018['successful_homicide'], label='2018')

# Set plot title, x-axis label, and y-axis label
ax.set_title('Homicide Counts by Month')
ax.set_xlabel('Month')
ax.set_ylabel('Count')

# Display legend
ax.legend()

# Show the plot
plt.show()
```



In [38]:

```
# Group the data by county and year
grouped_data = data.groupby(['county', 'year']).sum()

# Plot the data using a line chart
fig, ax = plt.subplots(figsize=(10, 6))
for county in grouped_data.index.levels[0]:
    ax.plot(grouped_data.loc[county]['successful_homicide'], label=county)
ax.set_xlabel('Year')
ax.set_ylabel('Homicide')
ax.legend()
plt.show()
```



1. From the correlation matrix and the correlation heat map we are able to draw some assmptions based on the observations and therefore state a new hypothesis:

The number of successful homicide cases in a county is positively correlated with the number of unsuccessful homicide cases in the same county.

2. From the bar chart and subplots above it seems the number of successful homicide convictions is more prevalent in some areas than most. For the purposes of this analysis we shall classify counties under two classes urban areas and rural areas.

We can therefore state a hypothesis as follows: The success rate of homicide cases is higher in urban areas compared to rural areas in the UK.

***There are distinct clusters of crime patterns across different counties in the UK, based on the success and type of crime reported to the Crown Prosecution Service from 2014 to 2018.

In [39]:

```
from sklearn.linear_model import LinearRegression
from sklearn.model_selection import train_test_split
from sklearn.metrics import mean_squared_error, r2_score
```

In [40]:

```
# Select relevant columns
df = data[['county', 'successful_homicide', 'unsuccessful_homicide']]
```

In [41]:

```
df
```

Out[41]:

	county	successful_homicide	unsuccessful_homicide
1	Avon and Somerset	1	0
2	Bedfordshire	0	0
3	Cambridgeshire	0	0
4	Cheshire	1	1
5	Cleveland	0	0
...	...	...	...
855	Warwickshire	0	0
856	West Mercia	6	0
857	West Midlands	11	1
858	West Yorkshire	5	2
859	Wiltshire	0	0

840 rows × 3 columns

In [42]:

```
# Group data by county and calculate the total number of successful and unsuccessful crim
df = df.groupby('county').agg({'successful_homicide': 'sum', 'unsuccessful_homicide': 'su
```

In [43]:

```
df
```

Out[43]:

	county	successful_homicide	unsuccessful_homicide
0	Avon and Somerset	34	9
1	Bedfordshire	18	6
2	Cambridgeshire	20	2
3	Cheshire	22	4
4	Cleveland	17	4
5	Cumbria	7	0
6	Derbyshire	15	1
7	Devon and Cornwall	24	8
8	Dorset	13	1
9	Durham	13	1
10	Dyfed Powys	5	1
11	Essex	38	4
12	Gloucestershire	15	2
13	GreaterManchester	76	15
14	Gwent	14	3
15	Hampshire	24	6
16	Hertfordshire	22	5
17	Humberside	19	2
18	Kent	82	20
19	Lancashire	38	6
20	Leicestershire	34	7
21	Lincolnshire	17	3
22	Merseyside	49	6
23	Metropolitan and City	297	110
24	Norfolk	11	3
25	North Wales	20	6
26	North Yorkshire	22	5
27	Northamptonshire	21	2
28	Northumbria	27	2
29	Nottinghamshire	56	14
30	South Wales	26	1
31	South Yorkshire	31	10
32	Staffordshire	23	2
33	Suffolk	10	1
34	Surrey	3	2
35	Sussex	5	2
36	Thames Valley	68	21

	county	successful_homicide	unsuccessful_homicide
37	Warwickshire	9	1
38	West Mercia	24	9
39	West Midlands	101	25
40	West Yorkshire	87	19
41	Wiltshire	7	1

In [44]:

```
# Create a new column for the total number of against person cases
df['total_homicide'] = df['successful_homicide'] + df['unsuccessful_homicide']
```

In [45]:

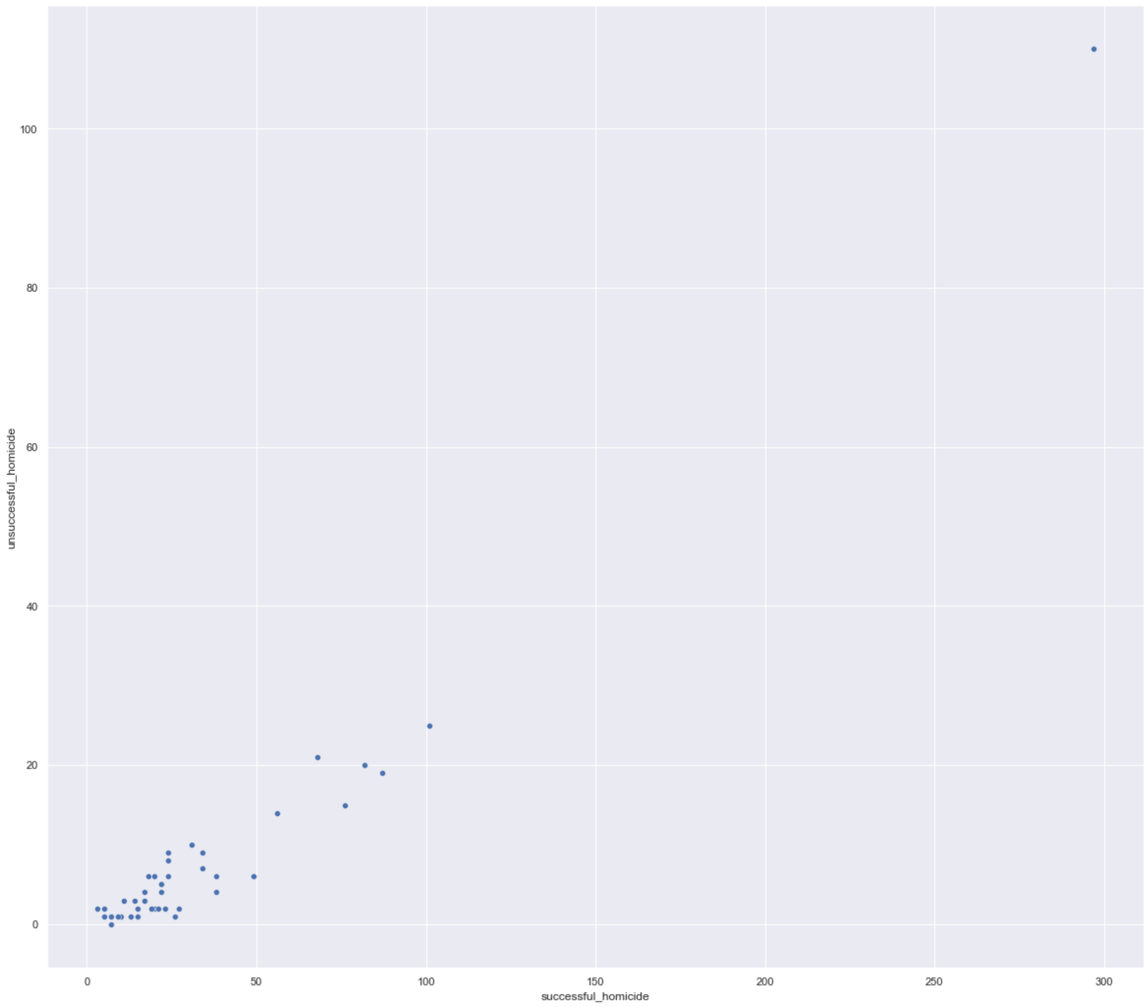
```
# Calculate the percentage of successful against person cases for each county
df['success_rate'] = df['successful_homicide'] / df['total_homicide']
```

In [46]:

```
# Plot a scatter plot to visualize the relationship between successful and unsuccessful c
sns.scatterplot(x='successful_homicide', y='unsuccessful_homicide', data=df)
```

Out[46]:

```
<AxesSubplot:xlabel='successful_homicide', ylabel='unsuccessful_homicide'>
```



In [47]:

```
# Split the data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(df['successful_homicide'], df['unsucc
```

In [48]:

```
# Fit a linear regression model to the training data
reg = LinearRegression().fit(X_train.values.reshape(-1, 1), y_train)
```

In [49]:

```
# Predict the number of unsuccessful criminal damage cases using the trained model and th
y_pred = reg.predict(X_test.values.reshape(-1, 1))
```

In [50]:

```
# Calculate the mean squared error and the coefficient of determination (R-squared) for t
mse = mean_squared_error(y_test, y_pred)
r2 = r2_score(y_test, y_pred)
```

In [51]:

```
print("Mean squared error: {:.2f}".format(mse))
print("Coefficient of determination (R-squared): {:.2f}".format(r2))
```

```
Mean squared error: 21.96
Coefficient of determination (R-squared): 0.61
```

We first select the columns relevant to the hypothesis (county, successful homicide convictions, and unsuccessful homicide convictions). We then groups the data by county and calculate the total number of successful and unsuccessful homicide convictions for each county.(Type of feature engineering). We thenalso creates a new column for the total number of homicide convictions and calculates the percentage of successful homicide convictions for each county.

Then we plot a scatter plot to visualize the relationship between successful and unsuccessful homicide convictions. We then split the data into training and testing sets and fit a linear regression model to the training data. We then use the trained model to predict the number of unsuccessful homicide convictions using the test data and calculate the mean squared error and the coefficient of determination (R-squared) for the predictions.

In this case, the MSE is reported as 21.96. A lower MSE indicates that the model's predictions are closer to the actual values, suggesting a better fit.

The coefficient of determination, commonly known as R-squared, is a statistical measure that represents the proportion of the variance in the dependent variable that is predictable from the independent variables in a regression model. R-squared values range from 0 to 1, with higher values indicating a better fit of the model to the data. In this case, the R-squared value is reported as 0.61, suggesting that approximately 61% of the variance in the dependent variable can be explained by the independent variables in the model.

Feature Engineering 1

Since our target feature is successful homicide convictions calculating the ratio of successful homicide convictions to total convictions (successful + unsuccessful) may make sense. This feature has been introduced because it can provide insights into the overall success rate for different types of crimes. Using filtering we can use Homicide convictions only.

In [52]:

```
import re
```

In [53]:

```
# Regular expression specifying a set of strings matching successful and unsuccessful col
data['total_convictions'] = data.filter(regex='^(successful|unsuccessful)').sum(axis=1)
data['success_ratio'] = data['successful_homicide'] / data['total_convictions']
```

In [54]:

```
data.head()
```

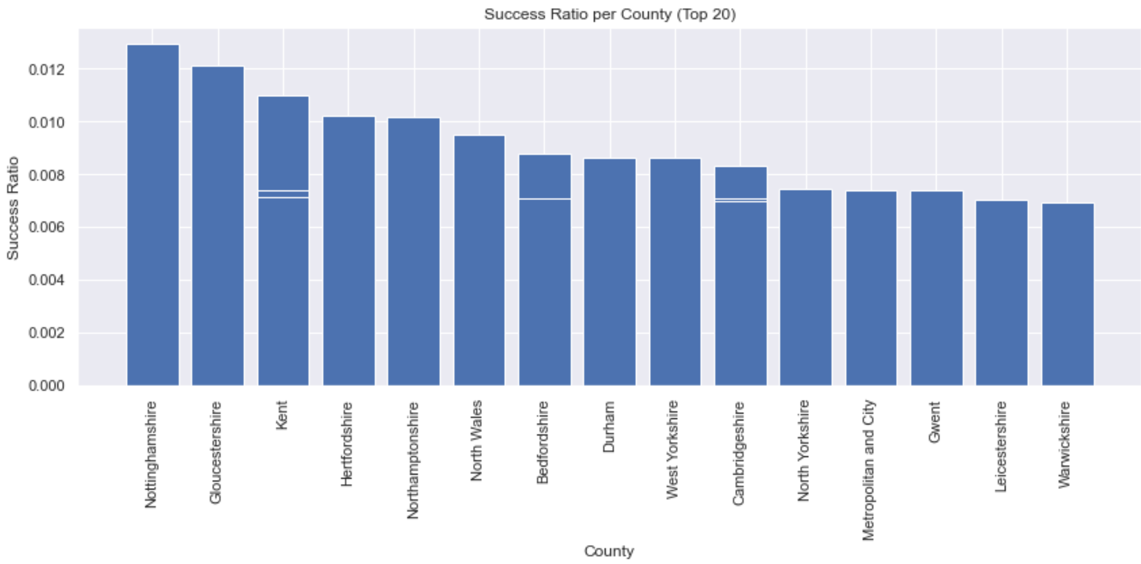
Out[54]:

	year	month	county	successful_homicide	unsuccessful_homicide	successful_aga
1	2014	april	Avon and Somerset	1	0	
2	2014	april	Bedfordshire	0	0	
3	2014	april	Cambridgeshire	0	0	
4	2014	april	Cheshire	1	1	
5	2014	april	Cleveland	0	0	

5 rows × 30 columns

In [55]:

```
# Sort the DataFrame by the 'success_ratio' column in descending order and select the top
top_20_counties = data.sort_values('success_ratio', ascending=False).head(20)

# Create a bar chart
plt.figure(figsize=(12, 6))
plt.bar(top_20_counties['county'], top_20_counties['success_ratio'])
plt.xlabel('County')
plt.ylabel('Success Ratio')
plt.title('Success Ratio per County (Top 20)')
plt.xticks(rotation=90)  # Rotate the x-axis labels for better visibility

plt.tight_layout()
plt.show()
```



Extracting additional time-related feature like the quarter feature from an engineered "date,"columns. This can help analyze crime trends over time.

Feature Engineering 2

In [56]:

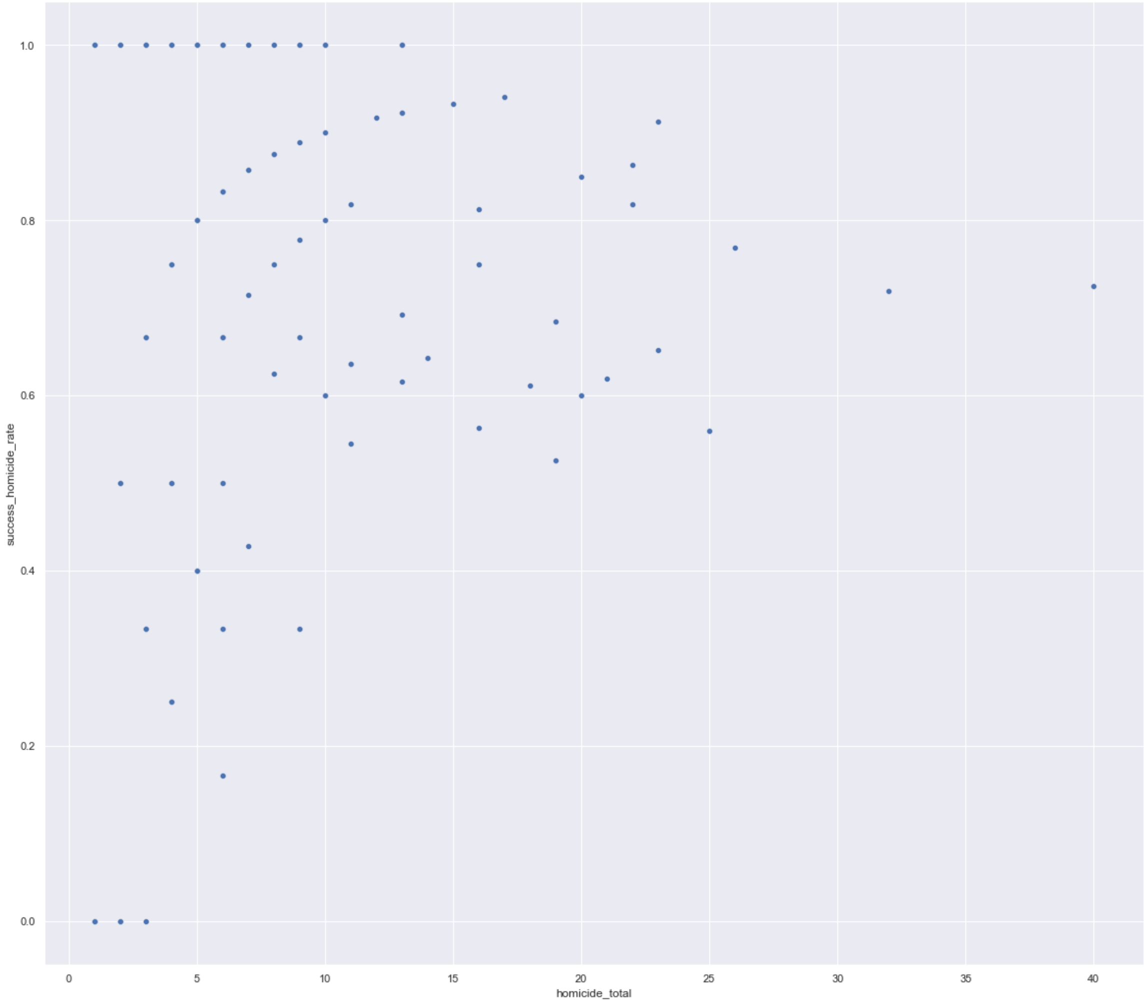
```
# Extracting a new feature successful homicide convictions rate.
data['homicide_total'] =data['successful_homicide'] + data['unsuccessful_homicide']
data['success_homicide_rate'] = data['successful_homicide']/data['homicide_total']
```

In [57]:

```
sns.scatterplot(x='homicide_total', y='success_homicide_rate', data=data)
```

Out[57]:

```
<AxesSubplot:xlabel='homicide_total', ylabel='success_homicide_rate'>
```



There seams to be an exponential relationship between the total homicide cases and the successful conviction rate therefore this feature was quite valuable in realising this observation. However we might want to consider removing the capping using algorithms as it may affect the machine learning models.

Additional Time Related Feature

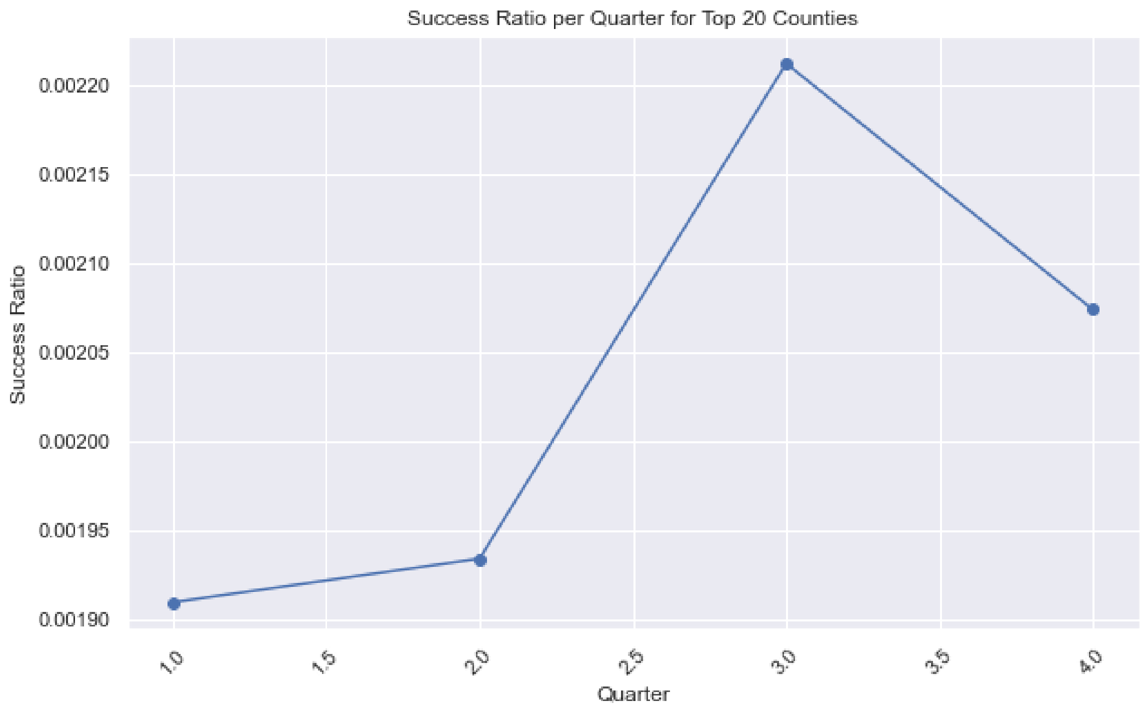
Extracting additional time-related feature, that is the quarter feature from an engineered "date"column. This can help analyze crime trends over time. The capping at one may not neccessarily be futal as it is a rate therefore we expect a capping.

In [58]:

```
# Create a date column by combining month and year
data['date'] = pd.to_datetime(data['month'] + ' ' + data['year'].astype(str))
data['quarter'] = data['date'].dt.quarter
```

In [59]:

```
# Select the top 20 counties based on success_ratio
top_counties = data.groupby('county').mean().nlargest(20, 'success_ratio').index

# Filter the dataset for the top counties
df_top_counties = data[data['county'].isin(top_counties)]

# Group the data by quarter and calculate the average success_ratio
df_grouped = df_top_counties.groupby('quarter').mean()

# Plotting the chart
quarters = df_grouped.index
success_ratio = df_grouped['success_ratio']

plt.figure(figsize=(10, 6))
plt.plot(quarters, success_ratio, marker='o')
plt.xlabel('Quarter')
plt.ylabel('Success Ratio')
plt.title('Success Ratio per Quarter for Top 20 Counties')
plt.xticks(rotation=45)
plt.grid(True)
plt.show()
```



Defining Transformer Classes for the Data before Machine Learning

Transformer classes enable transformation and data cleaning while preprocessing the data for machine learning.In our case we will use a custom transformer to remove outliers, since we have noted earlier on that there might be outliers within components of the train set. We also define a custom transformer class for selecting specific columns from the dataset since the success_ratio feature extracted earlier depends on columns selected.

In [60]:

```
# Importing the libraries
from sklearn.base import BaseEstimator, TransformerMixin
```

In [61]:

```
# Defining a custom transformer class for selecting specific columns from the dataset
class ColumnSelector(BaseEstimator, TransformerMixin):
    def __init__(self, columns):
        self.columns = columns

    def fit(self, data, y=None):
        return self

    def transform(self, data):
        return data[self.columns]
```

In [62]:

```
data.columns
```

Out[62]:

```
Index(['year', 'month', 'county', 'successful_homicide',
       'unsuccessful_homicide', 'successful_against_person',
       'unsuccessful_against_person', 'successful_sexual_offences',
       'unsuccessful_sexual_offences', 'successful_burglary',
       'unsuccessful_burglary', 'successful_robbery', 'unsuccessful_robber
y',
       'successful_theft', 'unsuccessful_theft', 'successful_fraud',
       'unsuccessful_fraud', 'successful_criminal_damage',
       'unsuccessful_criminal_damage', 'successful_drugs',
       'unsuccessful_drugs', 'successful_public_order',
       'unsuccessful_public_order', 'successful_other', 'unsuccessful_othe
r',
       'successful_motoring', 'unsuccessful_motoring', 'unsuccessful_admi
n',
       'total_convictions', 'success_ratio', 'homicide_total',
       'success_homicide_rate', 'date', 'quarter'],
      dtype='object')
```

In [63]:

```
# Creating a case for the transformer to select columns
columns_to_select = ['year', 'month', 'county', 'successful_homicide',
       'unsuccessful_homicide', 'successful_against_person','successful_sexual_offences',
       'successful_burglary', 'successful_robbery', 'successful_theft','successful_fraud'
       'successful_criminal_damage','successful_drugs','successful_public_order', 'succes
       'successful_motoring','total_convictions', 'success_ratio', 'homicide_total','succ
column_selector = ColumnSelector(columns=columns_to_select)
```

In [64]:

```
# Applying the transformer to the data
selected_data = column_selector.transform(data)
```

In [65]:

```
# We define a class named 'Outlier' that inherits from the BaseEstimator and TransformerM

class Outlier(BaseEstimator, TransformerMixin):
    def __init__(self, threshold=3):
        self.threshold = threshold

    def fit(self, data, y=None):
        return self

    def transform(self, data):
        # Copying the input data
        data_out = data.copy()

        # Iterating over each column and remove outliers
        for column in data.columns:
            # Calculate the z-score for each value in the column
            z_scores = np.abs((data[column] - data[column].mean()) / data[column].std())

            # Replace outliers with NaN values
            data_out[column][z_scores > self.threshold] = np.nan

        # Drop rows with NaN values
        data_out.dropna(inplace=True)

        return data_out
```