A Safer World: Understanding and Reducing Crime

In [1]:
```
# Importing required libraries
import pandas as pd # required to work with data
import numpy as np # required to round the data in the correlation matrix
import matplotlib.pyplot as plt # required for data visualization
from ydata_profiling import ProfileReport # Summarize and profile the data
import seaborn as sns # required for data visualization
import os # Data dirctoy
os.chdir("C://Users//HP//Desktop//crime_pop") # Data directory
pd.set_option("display.max_columns",None) # Displaying all columns
```

In [2]:
```
# Load data
Population_Age_Sex = pd.read_csv("PopulationByAgeSex.csv")
violent_and_sexual_crime = pd.read_excel("data_cts_violent_and_sexual_crime.xlsx")
corruption_and_economic_crime = pd.read_excel("data_cts_corruption_and_economic_cri
total_government_expenditure_on_education_gdp = pd.read_csv("total-government-expen
countries_of_the_world = pd.read_csv("Countries of the World.csv")
```

Population by Age and Sex Dataset

Data Profiling

In [3]:
```
# Data information
Population_Age_Sex.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 66893 entries, 0 to 66892
Data columns (total 71 columns):
 #   Column            Non-Null Count  Dtype  
---  ------            --------------  -----  
 0   Id                66893 non-null  int64  
 1   LocID             66893 non-null  int64  
 2   Location          66893 non-null  object 
 3   Time              66893 non-null  int64  
 4   PopMale_0_4       66893 non-null  float64
 5   PopFemale_0_4     66893 non-null  float64
 6   PopTotal_0_4      66893 non-null  float64
 7   PopMale_5_9       66893 non-null  float64
 8   PopFemale_5_9     66893 non-null  float64
 9   PopTotal_5_9      66893 non-null  float64
 10  PopMale_10_14     66893 non-null  float64
 11  PopFemale_10_14   66893 non-null  float64
 12  PopTotal_10_14    66893 non-null  float64
 13  PopMale_15_19     66893 non-null  float64
 14  PopFemale_15_19   66893 non-null  float64
 15  PopTotal_15_19    66893 non-null  float64
 16  PopMale_20_24     66893 non-null  float64
 17  PopFemale_20_24   66893 non-null  float64
 18  PopTotal_20_24    66893 non-null  float64
 19  PopMale_25_29     66893 non-null  float64
 20  PopFemale_25_29   66893 non-null  float64
 21  PopTotal_25_29    66893 non-null  float64
 22  PopMale_30_34     66893 non-null  float64
 23  PopFemale_30_34   66893 non-null  float64
 24  PopTotal_30_34    66893 non-null  float64
 25  PopMale_35_39     66893 non-null  float64
 26  PopFemale_35_39   66893 non-null  float64
 27  PopTotal_35_39    66893 non-null  float64
 28  PopMale_40_44     66893 non-null  float64
 29  PopFemale_40_44   66893 non-null  float64
 30  PopTotal_40_44    66893 non-null  float64
 31  PopMale_45_49     66893 non-null  float64
 32  PopFemale_45_49   66893 non-null  float64
 33  PopTotal_45_49    66893 non-null  float64
 34  PopMale_50_54     66893 non-null  float64
 35  PopFemale_50_54   66893 non-null  float64
 36  PopTotal_50_54    66893 non-null  float64
 37  PopMale_55_59     66893 non-null  float64
 38  PopFemale_55_59   66893 non-null  float64
 39  PopTotal_55_59    66893 non-null  float64
 40  PopMale_60_64     66893 non-null  float64
 41  PopFemale_60_64   66893 non-null  float64
 42  PopTotal_60_64    66893 non-null  float64
 43  PopMale_65_69     66893 non-null  float64
 44  PopFemale_65_69   66893 non-null  float64
 45  PopTotal_65_69    66893 non-null  float64
 46  PopMale_70_74     66893 non-null  float64
 47  PopFemale_70_74   66893 non-null  float64
 48  PopTotal_70_74    66893 non-null  float64
 49  PopMale_75_79     66893 non-null  float64
 50  PopFemale_75_79   66893 non-null  float64
```

```
 51  PopTotal_75_79     66893 non-null  float64
 52  PopMale_80_84      66893 non-null  float64
 53  PopFemale_80_84    66893 non-null  float64
 54  PopTotal_80_84     66893 non-null  float64
 55  PopMale_85_89      66893 non-null  float64
 56  PopFemale_85_89    66893 non-null  float64
 57  PopTotal_85_89     66893 non-null  float64
 58  PopMale_90_94      66893 non-null  float64
 59  PopFemale_90_94    66893 non-null  float64
 60  PopTotal_90_94     66893 non-null  float64
 61  PopMale_95_99      66893 non-null  float64
 62  PopFemale_95_99    66893 non-null  float64
 63  PopTotal_95_99     66893 non-null  float64
 64  PopMale_100Plus    66893 non-null  float64
 65  PopFemale_100Plus  66893 non-null  float64
 66  PopTotal_100Plus   66893 non-null  float64
 67  PopMale            66893 non-null  float64
 68  PopFemale          66893 non-null  float64
 69  PopTotal           66893 non-null  float64
 70  YearDataCompleted  66893 non-null  int64
dtypes: float64(66), int64(4), object(1)
memory usage: 36.2+ MB
```

In [4]:
```
# Show the first 5 rows
Population_Age_Sex.head()
```

Out[4]:

	Id	LocID	Location	Time	PopMale_0_4	PopFemale_0_4	PopTotal_0_4	PopMale_5_9
0	1	4	Afghanistan	1950	630.044	661.578	1291.622	516.206
1	2	4	Afghanistan	1951	641.199	673.293	1314.492	525.302
2	3	4	Afghanistan	1952	650.825	669.274	1320.099	533.097
3	4	4	Afghanistan	1953	659.896	663.606	1323.502	538.351
4	5	4	Afghanistan	1954	670.694	663.295	1333.989	540.820

In [5]:
```
# Show the last 5 rows
Population_Age_Sex.tail()
```

Out[5]:

	Id	LocID	Location	Time	PopMale_0_4	PopFemale_0_4	PopTotal_0_4	PopMa
66888	66889	716	Zimbabwe	2096	953.505	939.918	1893.423	9
66889	66890	716	Zimbabwe	2097	950.059	936.463	1886.522	9
66890	66891	716	Zimbabwe	2098	946.047	932.455	1878.502	9
66891	66892	716	Zimbabwe	2099	941.631	928.001	1869.632	9
66892	66893	716	Zimbabwe	2100	937.118	923.351	1860.469	9

In [6]:
```
#loop through the columns and check the missing values
for col in Population_Age_Sex.columns:
    pct_missing = Population_Age_Sex[col].isnull().mean()
    print(f"{col} - {pct_missing :.1%}")
```

```
Id - 0.0%
LocID - 0.0%
Location - 0.0%
Time - 0.0%
PopMale_0_4 - 0.0%
PopFemale_0_4 - 0.0%
PopTotal_0_4 - 0.0%
PopMale_5_9 - 0.0%
PopFemale_5_9 - 0.0%
PopTotal_5_9 - 0.0%
PopMale_10_14 - 0.0%
PopFemale_10_14 - 0.0%
PopTotal_10_14 - 0.0%
PopMale_15_19 - 0.0%
PopFemale_15_19 - 0.0%
PopTotal_15_19 - 0.0%
PopMale_20_24 - 0.0%
PopFemale_20_24 - 0.0%
PopTotal_20_24 - 0.0%
PopMale_25_29 - 0.0%
PopFemale_25_29 - 0.0%
PopTotal_25_29 - 0.0%
PopMale_30_34 - 0.0%
PopFemale_30_34 - 0.0%
PopTotal_30_34 - 0.0%
PopMale_35_39 - 0.0%
PopFemale_35_39 - 0.0%
PopTotal_35_39 - 0.0%
PopMale_40_44 - 0.0%
PopFemale_40_44 - 0.0%
PopTotal_40_44 - 0.0%
PopMale_45_49 - 0.0%
PopFemale_45_49 - 0.0%
PopTotal_45_49 - 0.0%
PopMale_50_54 - 0.0%
PopFemale_50_54 - 0.0%
PopTotal_50_54 - 0.0%
PopMale_55_59 - 0.0%
PopFemale_55_59 - 0.0%
PopTotal_55_59 - 0.0%
PopMale_60_64 - 0.0%
PopFemale_60_64 - 0.0%
PopTotal_60_64 - 0.0%
PopMale_65_69 - 0.0%
PopFemale_65_69 - 0.0%
PopTotal_65_69 - 0.0%
PopMale_70_74 - 0.0%
PopFemale_70_74 - 0.0%
PopTotal_70_74 - 0.0%
PopMale_75_79 - 0.0%
PopFemale_75_79 - 0.0%
PopTotal_75_79 - 0.0%
PopMale_80_84 - 0.0%
PopFemale_80_84 - 0.0%
PopTotal_80_84 - 0.0%
PopMale_85_89 - 0.0%
```

```
PopFemale_85_89 - 0.0%
PopTotal_85_89 - 0.0%
PopMale_90_94 - 0.0%
PopFemale_90_94 - 0.0%
PopTotal_90_94 - 0.0%
PopMale_95_99 - 0.0%
PopFemale_95_99 - 0.0%
PopTotal_95_99 - 0.0%
PopMale_100Plus - 0.0%
PopFemale_100Plus - 0.0%
PopTotal_100Plus - 0.0%
PopMale - 0.0%
PopFemale - 0.0%
PopTotal - 0.0%
YearDataCompleted - 0.0%
```

In [7]:
```
# Data statistics
Population_Age_Sex.describe().T
```

Out[7]:

	count	mean	std	min	25%	50%
Id	66893.0	33447.000000	1.931049e+04	1.000	16724.000	33447.000
LocID	66893.0	1077.688488	7.290784e+02	4.000	462.000	922.000
Time	66893.0	2025.000000	4.358932e+01	1950.000	1987.000	2025.000
PopMale_0_4	66893.0	28467.893812	5.917742e+04	0.545	289.002	2913.602
PopFemale_0_4	66893.0	27006.915349	5.604457e+04	0.512	278.516	2810.628
...	...	...	...	...	...	...
PopTotal_100Plus	66893.0	330.551003	1.319731e+03	0.000	0.097	2.655
PopMale	66893.0	339577.045142	7.439528e+05	6.812	3286.896	34439.569
PopFemale	66893.0	335464.382981	7.319416e+05	6.889	3269.616	34608.330
PopTotal	66893.0	675041.428124	1.475830e+06	13.763	6585.116	69631.853
YearDataCompleted	66893.0	1.000000	0.000000e+00	1.000	1.000	1.000

70 rows × 8 columns

In [8]:
```
# Data statistics
Population_Age_Sex.describe(include="object").T
```

Out[8]:

	count	unique	top	freq
Location	66893	440	Latin America and the Caribbean	302

In [9]:
```
# Check duplicates row
duplicates = Population_Age_Sex.duplicated().sum()
print(f"Number of duplicate rows: {duplicates}")
```

```
Number of duplicate rows: 0
```

In [10]:
```
# Check duplicates based on Location and Time
duplicates_location_time = Population_Age_Sex.duplicated(subset=["Location","Time"]
print(f"Number of duplicate rows: {duplicates_location_time}")
```

```
Number of duplicate rows: 453
```

In [11]:
```
# Duplicated values
Population_Age_Sex[Population_Age_Sex.duplicated(subset=["Location","Time"])]["Loca
```

Out[11]:
```
Location
Europe                             151
Latin America and the Caribbean    151
Northern America                   151
Name: count, dtype: int64
```

In [12]:
```
# Locations that have more than 1 unique LocID
duplicates = Population_Age_Sex.groupby('Location')['LocID'].nunique()
duplicates = duplicates[duplicates > 1]
print(duplicates)
```

```
Location
Europe                             2
Latin America and the Caribbean    2
Northern America                   2
Name: LocID, dtype: int64
```

In [13]:
```
# Data profiling
"""
profile = ProfileReport(Population_Age_Sex,title="Population_Age_Sex",minimal=True)

profile.to_file("Population_Age_Sex.html")   # Save report to HTML file
"""
```

Out[13]:
```
'\nprofile = ProfileReport(Population_Age_Sex,title="Population_Age_Sex",minimal=T
rue) # Summarize and profile the data\n\nprofile.to_file("Population_Age_Sex.htm
l")   # Save report to HTML file \n'
```

Data Wrangling

In [14]:
```
Population_Age_Sex.head(1)
```

Out[14]:

	Id	LocID	Location	Time	PopMale_0_4	PopFemale_0_4	PopTotal_0_4	PopMale_5_9
0	1	4	Afghanistan	1950	630.044	661.578	1291.622	516.206

In [15]:
```
# Drop unnecessary columns
Population_Age_Sex_1 = Population_Age_Sex.drop(columns=["Id","YearDataCompleted"])
```

In [16]:
```
# Rename columns
Population_Age_Sex_2 = Population_Age_Sex_1.rename(columns={"Location":"Country","T
```

In [17]:
```
# Divide age into groups
age_groups = {
    "Children_0_14": ['PopMale_0_4', 'PopFemale_0_4', 'PopMale_5_9', 'PopFemale_5_9
                    'PopMale_10_14', 'PopFemale_10_14'],
    "YoungAdults_15_24": ['PopMale_15_19', 'PopFemale_15_19',
                           'PopMale_20_24', 'PopFemale_20_24'],
    "WorkingAdults_25_44": ['PopMale_25_29', 'PopFemale_25_29',
                             'PopMale_30_34', 'PopFemale_30_34',
                             'PopMale_35_39', 'PopFemale_35_39',
                             'PopMale_40_44', 'PopFemale_40_44'],
    "MatureAdults_45_64": ['PopMale_45_49', 'PopFemale_45_49',
                            'PopMale_50_54', 'PopFemale_50_54',
                            'PopMale_55_59', 'PopFemale_55_59',
                            'PopMale_60_64', 'PopFemale_60_64'],
    "Elderly_65Plus": ['PopMale_65_69', 'PopFemale_65_69',
                       'PopMale_70_74', 'PopFemale_70_74',
                       'PopMale_75_79', 'PopFemale_75_79',
                       'PopMale_80_84', 'PopFemale_80_84',
                       'PopMale_85_89', 'PopFemale_85_89',
                       'PopMale_90_94', 'PopFemale_90_94',
                       'PopMale_95_99', 'PopFemale_95_99',
                       'PopMale_100Plus', 'PopFemale_100Plus']
            }

# Add new columns for each age group
for group_name, columns in age_groups.items():
    Population_Age_Sex_2[f"{group_name}_Male"] = Population_Age_Sex_2[[col for col
    Population_Age_Sex_2[f"{group_name}_Female"] = Population_Age_Sex_2[[col for co
    Population_Age_Sex_2[f"{group_name}_Total"] = Population_Age_Sex_2[f"{group_nam
```

In [18]:
```
# Check if age groups were divided correctly
new =  Population_Age_Sex_2["Children_0_14_Male"].sum()
old = (Population_Age_Sex_2[["PopMale_0_4","PopMale_5_9","PopMale_10_14"]].sum()).s
print(f"New:{new}, Old: {old}")
```

```
New:5541837016.446, Old: 5541837016.446
```

In [19]:
```
# Check if age groups were divided correctly
new = Population_Age_Sex_2["WorkingAdults_25_44_Female"].sum()
old = (Population_Age_Sex_2[["PopFemale_25_29","PopFemale_30_34","PopFemale_35_39",
print(f"New:{new}, Old: {old}")
```

```
New:5844074542.214, Old: 5844074542.214001
```

In [20]:
```
# Drop old ages columns
Population_Age_Sex_3 = Population_Age_Sex_2.drop(columns=['PopMale_0_4', 'PopFemale
       'PopTotal_0_4', 'PopMale_5_9', 'PopFemale_5_9', 'PopTotal_5_9',
       'PopMale_10_14', 'PopFemale_10_14', 'PopTotal_10_14', 'PopMale_15_19',
       'PopFemale_15_19', 'PopTotal_15_19', 'PopMale_20_24', 'PopFemale_20_24',
       'PopTotal_20_24', 'PopMale_25_29', 'PopFemale_25_29', 'PopTotal_25_29',
       'PopMale_30_34', 'PopFemale_30_34', 'PopTotal_30_34', 'PopMale_35_39',
       'PopFemale_35_39', 'PopTotal_35_39', 'PopMale_40_44', 'PopFemale_40_44',
       'PopTotal_40_44', 'PopMale_45_49', 'PopFemale_45_49', 'PopTotal_45_49',
       'PopMale_50_54', 'PopFemale_50_54', 'PopTotal_50_54', 'PopMale_55_59',
       'PopFemale_55_59', 'PopTotal_55_59', 'PopMale_60_64', 'PopFemale_60_64',
       'PopTotal_60_64', 'PopMale_65_69', 'PopFemale_65_69', 'PopTotal_65_69',
```

```
       'PopMale_70_74', 'PopFemale_70_74', 'PopTotal_70_74', 'PopMale_75_79',
       'PopFemale_75_79', 'PopTotal_75_79', 'PopMale_80_84', 'PopFemale_80_84',
       'PopTotal_80_84', 'PopMale_85_89', 'PopFemale_85_89', 'PopTotal_85_89',
       'PopMale_90_94', 'PopFemale_90_94', 'PopTotal_90_94', 'PopMale_95_99',
       'PopFemale_95_99', 'PopTotal_95_99', 'PopMale_100Plus',
       'PopFemale_100Plus', 'PopTotal_100Plus'])
```

In [21]:

```
# Ensure country consistency and format
Population_Age_Sex_3["Country"] = Population_Age_Sex_3["Country"].str.title().str.s
```

In [22]:

```
# Calcualte male to female ratio for each group
population_columns = [
    ('PopMale', 'PopFemale', 'Male_Female_Ratio'),
    ('Children_0_14_Male', 'Children_0_14_Female', 'Children_0_14_MF_Ratio'),
    ('YoungAdults_15_24_Male', 'YoungAdults_15_24_Female', 'YoungAdults_15_24_MF_Ra
    ('WorkingAdults_25_44_Male', 'WorkingAdults_25_44_Female', 'WorkingAdults_25_44
    ('MatureAdults_45_64_Male', 'MatureAdults_45_64_Female', 'MatureAdults_45_64_MF
    ('Elderly_65Plus_Male', 'Elderly_65Plus_Female', 'Elderly_65Plus_MF_Ratio')
                    ]

# Calculate male-to-female ratios for all specified groups
for male_col, female_col, ratio_col in population_columns:
    Population_Age_Sex_3[ratio_col] = Population_Age_Sex_3[male_col] / Population_A
```

In [23]:

```
# Show the final dataset
Population_Age_Sex_final = Population_Age_Sex_3.copy()
Population_Age_Sex_final.head(2)
```

Out[23]:

	LocID	Country	Year	PopMale	PopFemale	PopTotal	Children_0_14_Male	Children_
0	4	Afghanistan	1950	4099.243	3652.874	7752.117	1607.628	
1	4	Afghanistan	1951	4134.756	3705.395	7840.151	1632.112	

Violent and Sexual Crime (2003 – 2022) Dataset

Data Profiling

In [24]:

```
# Data information
violent_and_sexual_crime.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 26114 entries, 0 to 26113
Data columns (total 13 columns):
 #   Column               Non-Null Count  Dtype  
---  ------               --------------  -----  
 0   Iso3_code            26114 non-null  object 
 1   Country              26114 non-null  object 
 2   Region               26114 non-null  object 
 3   Subregion            26114 non-null  object 
 4   Indicator            26114 non-null  object 
 5   Dimension            26114 non-null  object 
 6   Category             26114 non-null  object 
 7   Sex                  26114 non-null  object 
 8   Age                  26114 non-null  object 
 9   Year                 26114 non-null  int64  
 10  Unit of measurement  26114 non-null  object 
 11  VALUE                26114 non-null  float64
 12  Source               26114 non-null  object 
dtypes: float64(1), int64(1), object(11)
memory usage: 2.6+ MB
```

In [25]:

```
# Show the first 5 rows
violent_and_sexual_crime.head()
```

Out[25]:

	Iso3_code	Country	Region	Subregion	Indicator	Dimension	Category	Sex	Age
0	AZE	Azerbaijan	Asia	Western Asia	Violent offences	by type of offence	Serious assault	Total	Total
1	BEL	Belgium	Europe	Western Europe	Violent offences	by type of offence	Serious assault	Total	Total
2	BGR	Bulgaria	Europe	Eastern Europe	Violent offences	by type of offence	Serious assault	Total	Total
3	BHR	Bahrain	Asia	Western Asia	Violent offences	by type of offence	Serious assault	Total	Total
4	BLR	Belarus	Europe	Eastern Europe	Violent offences	by type of offence	Serious assault	Total	Total

In [26]:

```
# Show the last 5 rows
violent_and_sexual_crime.tail()
```

Out[26]:

	Iso3_code	Country	Region	Subregion	Indicator	Dimension	Category	Sex
26109	MNE	Montenegro	Europe	Southern Europe	Violent offences	by type of offence	Acts intended to induce fear or emotional dist...	Tota
26110	MUS	Mauritius	Africa	Sub-Saharan Africa	Violent offences	by type of offence	Acts intended to induce fear or emotional dist...	Tota
26111	SLV	El Salvador	Americas	Latin America and the Caribbean	Violent offences	by type of offence	Acts intended to induce fear or emotional dist...	Tota
26112	SRB	Serbia	Europe	Southern Europe	Violent offences	by type of offence	Acts intended to induce fear or emotional dist...	Tota
26113	SWZ	Eswatini	Africa	Sub-Saharan Africa	Violent offences	by type of offence	Acts intended to induce fear or emotional dist...	Tota

In [27]:

```
#loop through the columns and check the missing values
for col in violent_and_sexual_crime.columns:
    pct_missing = violent_and_sexual_crime[col].isnull().mean()
    print(f"{col} - {pct_missing :.1%}")
```

```
Iso3_code - 0.0%
Country - 0.0%
Region - 0.0%
Subregion - 0.0%
Indicator - 0.0%
Dimension - 0.0%
Category - 0.0%
Sex - 0.0%
Age - 0.0%
Year - 0.0%
Unit of measurement - 0.0%
VALUE - 0.0%
Source - 0.0%
```

In [28]:
```
# Data statistics
violent_and_sexual_crime.describe().T
```

Out[28]:

	count	mean	std	min	25%	50%	75%	ı
Year	26114.0	2015.013250	5.169799	2003.0	2011.000000	2016.000000	2019.0	20
VALUE	26114.0	4422.568304	35003.600857	0.0	4.052739	35.615407	393.0	9491

In [29]:
```
# Data statistics
violent_and_sexual_crime.describe(include="object").T
```

Out[29]:

	count	unique	top	freq
Iso3_code	26114	157	AUT	558
Country	26114	157	Austria	558
Region	26114	5	Europe	12002
Subregion	26114	15	Latin America and the Caribbean	6578
Indicator	26114	3	Violent offences	19848
Dimension	26114	2	by type of offence	19848
Category	26114	13	Sexual violence: Rape	4006
Sex	26114	3	Total	19848
Age	26114	1	Total	26114
Unit of measurement	26114	2	Counts	13073
Source	26114	18	CTS	24578

In [30]:
```
# Check duplicates row
duplicates = violent_and_sexual_crime.duplicated().sum()
print(f"Number of duplicate rows: {duplicates}")
```

```
Number of duplicate rows: 0
```

In [31]:
```
# Data profiling
"""
profile = ProfileReport(violent_and_sexual_crime,title="violent_and_sexual_crime",m

profile.to_file("violent_and_sexual_crime.html")   # Save report to HTML file
"""
```

Out[31]:
```
'\nprofile = ProfileReport(violent_and_sexual_crime,title="violent_and_sexual_crim
e",minimal=True) # Summarize and profile the data\n\nprofile.to_file("violent_and_
sexual_crime.html")   # Save report to HTML file \n'
```

Data Wrangling

In [32]:
```
# Ensure object columns format and consistency
col_without_numeric = list(violent_and_sexual_crime.select_dtypes(exclude=("float",
for col in col_without_numeric:
    print(f"Before '{col}': {len(set(violent_and_sexual_crime[col]))} / After '{col
```

```
Before 'Iso3_code': 157 / After 'Iso3_code': 157
Before 'Country': 157 / After 'Country': 157
Before 'Region': 5 / After 'Region': 5
Before 'Subregion': 15 / After 'Subregion': 15
Before 'Indicator': 3 / After 'Indicator': 3
Before 'Dimension': 2 / After 'Dimension': 2
Before 'Category': 13 / After 'Category': 13
Before 'Sex': 3 / After 'Sex': 3
Before 'Age': 1 / After 'Age': 1
Before 'Unit of measurement': 2 / After 'Unit of measurement': 2
Before 'Source': 18 / After 'Source': 18
```

In [33]:
```
# Filter column Indicator on Violent offences
violent_and_sexual_crime_1 = violent_and_sexual_crime[violent_and_sexual_crime["Ind
```

In [34]:
```
# Drop unnecessary columns
violent_and_sexual_crime_2 = violent_and_sexual_crime_1.drop(columns=["Dimension","
```

In [35]:
```
# Filter Unit of measurement to keep only Rate per 100,000 population
violent_and_sexual_crime_3 = violent_and_sexual_crime_2[violent_and_sexual_crime_2[
```

In [36]:
```
# Drop column Unit of measurement
violent_and_sexual_crime_4 = violent_and_sexual_crime_3.drop(columns=["Unit of meas
```

In [37]:
```
# Change column name
violent_and_sexual_crime_5 = violent_and_sexual_crime_4.rename(columns={"VALUE":"Cr
```

In [38]:
```
# Check the countries with 0 crime Rate_per_10000_pop
violent_and_sexual_crime_5[violent_and_sexual_crime_5["Crime_rate_per_100000_popula
```

Out[38]:
```
Country
Holy See                                          43
Grenada                                           38
Dominica                                          20
Liechtenstein                                     19
Lithuania                                         17
Andorra                                           16
Malta                                             15
Guyana                                            14
Sao Tome and Principe                             13
China, Macao Special Administrative Region        13
Name: count, dtype: int64
```

In [39]:
```
# Remove rows with 0 values in Rate_per_10000_pop column
violent_and_sexual_crime_6 = violent_and_sexual_crime_5[violent_and_sexual_crime_5[
```

In [40]:
```
# Chane the value Violent offences in Indicator to violent and sexual to merge the
violent_and_sexual_crime_6["Indicator"] = violent_and_sexual_crime_6["Indicator"].s
```

In [41]:
```
# Show the final dataset
violent_and_sexual_crime_final = violent_and_sexual_crime_6.copy()
violent_and_sexual_crime_final.head(2)
```

Out[41]:

	Iso3_code	Country	Region	Subregion	Indicator	Crime_category	Year	Crime_r
13073	AZE	Azerbaijan	Asia	Western Asia	Violent and Sexual	Serious assault	2003	
13074	BEL	Belgium	Europe	Western Europe	Violent and Sexual	Serious assault	2003	

Corruption and Economic Crime (2003 – 2022) Dataset

Data Profiling

In [42]:
```
# Data information
corruption_and_economic_crime.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 22754 entries, 0 to 22753
Data columns (total 13 columns):
 #   Column               Non-Null Count  Dtype  
---  ------               --------------  -----  
 0   Iso3_code            22754 non-null  object 
 1   Country              22754 non-null  object 
 2   Region               22754 non-null  object 
 3   Subregion            22754 non-null  object 
 4   Indicator            22754 non-null  object 
 5   Dimension            22754 non-null  object 
 6   Category             22754 non-null  object 
 7   Sex                  22754 non-null  object 
 8   Age                  22754 non-null  object 
 9   Year                 22754 non-null  int64  
 10  Unit of measurement  22754 non-null  object 
 11  VALUE                22754 non-null  float64
 12  Source               22754 non-null  object 
dtypes: float64(1), int64(1), object(11)
memory usage: 2.3+ MB
```

In [43]:
```
# Show the first 5 rows
corruption_and_economic_crime.head()
```

Out[43]:

	Iso3_code	Country	Region	Subregion	Indicator	Dimension	Category	Sex	Ag
0	ARM	Armenia	Asia	Western Asia	Offences	by type of offence	Corruption	Total	Tot
1	AUT	Austria	Europe	Western Europe	Offences	by type of offence	Corruption	Total	Tot
2	CHE	Switzerland	Europe	Western Europe	Offences	by type of offence	Corruption	Total	Tot
3	CHL	Chile	Americas	Latin America and the Caribbean	Offences	by type of offence	Corruption	Total	Tot
4	COL	Colombia	Americas	Latin America and the Caribbean	Offences	by type of offence	Corruption	Total	Tot

In [44]:
```
# Show the last 5 rows
corruption_and_economic_crime.tail()
```

Out[44]:

	Iso3_code	Country	Region	Subregion	Indicator	Dimension	Category	Sex
22749	SRB	Serbia	Europe	Southern Europe	Acts against the environment	by type of offence	Acts that result in the depletion of degradati...	Total
22750	SVK	Slovakia	Europe	Eastern Europe	Acts against the environment	by type of offence	Acts that result in the depletion of degradati...	Total
22751	SVN	Slovenia	Europe	Southern Europe	Acts against the environment	by type of offence	Acts that result in the depletion of degradati...	Total
22752	SWE	Sweden	Europe	Northern Europe	Acts against the environment	by type of offence	Acts that result in the depletion of degradati...	Total
22753	SWZ	Eswatini	Africa	Sub-Saharan Africa	Acts against the environment	by type of offence	Acts that result in the depletion of degradati...	Total

In [45]:

```
#loop through the columns and check the missing values
for col in corruption_and_economic_crime.columns:
    pct_missing = corruption_and_economic_crime[col].isnull().mean()
    print(f"{col} - {pct_missing :.1%}")
```

```
Iso3_code - 0.0%
Country - 0.0%
Region - 0.0%
Subregion - 0.0%
Indicator - 0.0%
Dimension - 0.0%
Category - 0.0%
Sex - 0.0%
Age - 0.0%
Year - 0.0%
Unit of measurement - 0.0%
VALUE - 0.0%
Source - 0.0%
```

In [46]:
```
# Data statistics
corruption_and_economic_crime.describe().T
```

Out[46]:

	count	mean	std	min	25%	50%	75%	
Year	22754.0	2015.348159	5.059154	2003.0	2013.0	2017.000000	2019.00000	
VALUE	22754.0	26752.747120	213446.469070	0.0	6.0	127.296302	1484.81667	702

In [47]:
```
# Data statistics
corruption_and_economic_crime.describe(include="object").T
```

Out[47]:

	count	unique	top	freq
Iso3_code	22754	157	SVK	372
Country	22754	157	Slovakia	372
Region	22754	5	Europe	11338
Subregion	22754	15	Latin America and the Caribbean	5282
Indicator	22754	2	Offences	20414
Dimension	22754	1	by type of offence	22754
Category	22754	17	Theft	3808
Sex	22754	1	Total	22754
Age	22754	1	Total	22754
Unit of measurement	22754	2	Counts	11377
Source	22754	15	CTS	21610

In [48]:
```
# Check duplicates row
duplicates = corruption_and_economic_crime.duplicated().sum()
print(f"Number of duplicate rows: {duplicates}")
```

```
Number of duplicate rows: 0
```

In [49]:
```
# Data profiling
"""
profile = ProfileReport(corruption_and_economic_crime,title="corruption_and_economi

profile.to_file("corruption_and_economic_crime.html")   # Save report to HTML file
"""
```

Out[49]:
```
'\nprofile = ProfileReport(corruption_and_economic_crime,title="corruption_and_eco
nomic_crime",minimal=True) # Summarize and profile the data\n\nprofile.to_file("co
rruption_and_economic_crime.html")   # Save report to HTML file \n'
```

Data Wrangling

In [50]:
```
# Ensure object columns format and consistency
col_without_numeric = list(corruption_and_economic_crime.select_dtypes(exclude=("fl
for col in col_without_numeric:
    print(f"Before '{col}': {len(set(corruption_and_economic_crime[col]))} / After
```

```
Before 'Iso3_code': 157 / After 'Iso3_code': 157
Before 'Country': 157 / After 'Country': 157
Before 'Region': 5 / After 'Region': 5
Before 'Subregion': 15 / After 'Subregion': 15
Before 'Indicator': 2 / After 'Indicator': 2
Before 'Dimension': 1 / After 'Dimension': 1
Before 'Category': 17 / After 'Category': 17
Before 'Sex': 1 / After 'Sex': 1
Before 'Age': 1 / After 'Age': 1
Before 'Unit of measurement': 2 / After 'Unit of measurement': 2
Before 'Source': 15 / After 'Source': 15
```

In [51]:
```
# Filter column Indicator on Violent offences
corruption_and_economic_crime_1 = corruption_and_economic_crime[corruption_and_econ
```

In [52]:
```
# Drop unnecessary columns
corruption_and_economic_crime_2 = corruption_and_economic_crime_1.drop(columns=["Di
```

In [53]:
```
# Filter Unit of measurement to keep only Rate per 100,000 population
corruption_and_economic_crime_3 = corruption_and_economic_crime_2[corruption_and_ec
```

In [54]:
```
# Drop column Unit of measurement
corruption_and_economic_crime_4 = corruption_and_economic_crime_3.drop(columns=["Un
```

In [55]:
```
# Change column name
corruption_and_economic_crime_5 = corruption_and_economic_crime_4.rename(columns={"
```

In [56]:
```
# Check the countries with 0 crime Rate_per_10000_pop
corruption_and_economic_crime_5[corruption_and_economic_crime_5["Crime_rate_per_100
```

Out[56]:
```
Country
Grenada                              60
Holy See                             41
Barbados                             37
Dominica                             35
Guyana                               22
Saint Kitts and Nevis                19
Antigua and Barbuda                  17
Bahamas                              15
Belize                               15
United Kingdom (Northern Ireland)    13
Name: count, dtype: int64
```

In [57]:
```
# Remove rows with 0 values in Rate_per_10000_pop column
corruption_and_economic_crime_6 = corruption_and_economic_crime_5[corruption_and_ec
```

In [58]:
```
# Chane the value Offences in Indicator to corruption and economic to merge the dat
corruption_and_economic_crime_6["Indicator"] = corruption_and_economic_crime_6["Ind
```

In [59]:
```
# Show the final dataset
corruption_and_economic_crime_final = corruption_and_economic_crime_6.copy()
corruption_and_economic_crime_final.head(2)
```

Out[59]:

	Iso3_code	Country	Region	Subregion	Indicator	Crime_category	Year	Crime_ra
11377	ARM	Armenia	Asia	Western Asia	Corruption and Economic	Corruption	2013	
11378	AUT	Austria	Europe	Western Europe	Corruption and Economic	Corruption	2013	

Total government Expenditure on Education GDP Dataset

Data Profiling

In [60]:
```
# Data information
total_government_expenditure_on_education_gdp.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 5676 entries, 0 to 5675
Data columns (total 4 columns):
 #   Column                                   Non-Null Count  Dtype  
---  ------                                   --------------  -----  
 0   Entity                                   5676 non-null   object 
 1   Code                                     5248 non-null   object 
 2   Year                                     5676 non-null   int64  
 3   Public spending on education as a share of GDP  5676 non-null   float64
dtypes: float64(1), int64(1), object(2)
memory usage: 177.5+ KB
```

In [61]:
```
# Show the first 5 rows
total_government_expenditure_on_education_gdp.head()
```

Out[61]:

	Entity	Code	Year	Public spending on education as a share of GDP
0	Afghanistan	AFG	2006	4.684761
1	Afghanistan	AFG	2007	4.174895
2	Afghanistan	AFG	2008	4.383672
3	Afghanistan	AFG	2009	4.810640
4	Afghanistan	AFG	2010	3.479450

In [62]:
```
# Show the last 5 rows
total_government_expenditure_on_education_gdp.tail()
```

Out[62]:

	Entity	Code	Year	Public spending on education as a share of GDP
5671	Zimbabwe	ZWE	2012	6.07021
5672	Zimbabwe	ZWE	2013	5.99598
5673	Zimbabwe	ZWE	2014	6.13835
5674	Zimbabwe	ZWE	2017	5.81878
5675	Zimbabwe	ZWE	2018	2.05049

In [63]:
```
#Loop through the columns and check the missing values
for col in total_government_expenditure_on_education_gdp.columns:
    pct_missing = total_government_expenditure_on_education_gdp[col].isnull().mean(
    print(f"{col} - {pct_missing :.1%}")
```

```
Entity - 0.0%
Code - 7.5%
Year - 0.0%
Public spending on education as a share of GDP - 0.0%
```

In [64]:
```
# Data statistics
total_government_expenditure_on_education_gdp.describe().T
```

Out[64]:

	count	mean	std	min	25%	50%	75%
Year	5676.0	2002.662967	15.650096	1870.0	1993.000000	2006.000000	2015.00000 2
Public spending on education as a share of GDP	5676.0	4.324355	2.076605	0.0	3.063535	4.195975	5.31437

In [65]:
```
# Data statistics
total_government_expenditure_on_education_gdp.describe(include="object").T
```

Out[65]:

	count	unique	top	freq
Entity	5676	220	Norway	54
Code	5248	204	FRA	54

In [66]:
```
# Check duplicates row
duplicates = total_government_expenditure_on_education_gdp.duplicated().sum()
print(f"Number of duplicate rows: {duplicates}")
```

```
Number of duplicate rows: 0
```

In [67]:
```
# Data profiling
"""
profile = ProfileReport(total_government_expenditure_on_education_gdp,title="total_

profile.to_file("total_government_expenditure_on_education_gdp.html")   # Save repo
"""
```

Out[67]:
```
'\nprofile = ProfileReport(total_government_expenditure_on_education_gdp,title="to
tal_government_expenditure_on_education_gdp",minimal=True) # Summarize and profile
the data\n\nprofile.to_file("total_government_expenditure_on_education_gdp.html")
# Save report to HTML file \n'
```

Data Wrangling

In [68]:
```
# Ensure object columns format and consistency
col_without_numeric = list(total_government_expenditure_on_education_gdp.select_dty
for col in col_without_numeric:
    print(f"Before '{col}': {len(set(total_government_expenditure_on_education_gdp[
```

```
Before 'Entity': 220 / After 'Entity': 220
Before 'Code': 205 / After 'Code': 205
```

In [69]:
```
# Change column names
total_government_expenditure_on_education_gdp_1 = total_government_expenditure_on_e
```

In [70]:
```
# Drop unnecessary columns
total_government_expenditure_on_education_gdp_2 = total_government_expenditure_on_e
```

In [71]:
```
# Show the final dataset
total_government_expenditure_on_education_gdp_final = total_government_expenditure_
total_government_expenditure_on_education_gdp_final.head(2)
```

Out[71]:

	Country	Year	Puplic_education_spending%_of_gdp
0	Afghanistan	2006	4.684761
1	Afghanistan	2007	4.174895

Countries of the World Dataset

Data Profiling

In [72]:
```
# Data information
countries_of_the_world.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 250 entries, 0 to 249
Data columns (total 4 columns):
 #   Column       Non-Null Count  Dtype
---  ------       --------------  -----
 0   Countries    250 non-null    object
 1   Capital      242 non-null    object
 2   Population   250 non-null    int64
 3   Area(Sq.Km)  250 non-null    float64
dtypes: float64(1), int64(1), object(2)
memory usage: 7.9+ KB
```

In [73]:
```
# Show the first 5 rows
countries_of_the_world.head()
```

Out[73]:

	Countries	Capital	Population	Area(Sq.Km)
0	Andorra	Andorra la Vella	84000	468.0
1	United Arab Emirates	Abu Dhabi	4975593	82880.0
2	Afghanistan	Kabul	29121286	647500.0
3	Antigua and Barbuda	St. John's	86754	443.0
4	Anguilla	The Valley	13254	102.0

In [74]:
```
# Show the last 5 rows
countries_of_the_world.tail()
```

Out[74]:

	Countries	Capital	Population	Area(Sq.Km)
245	Yemen	Sanaa	23495361	527970.0
246	Mayotte	Mamoudzou	159042	374.0
247	South Africa	Pretoria	49000000	1219912.0
248	Zambia	Lusaka	13460305	752614.0
249	Zimbabwe	Harare	11651858	390580.0

In [75]:
```
#loop through the columns and check the missing values
for col in countries_of_the_world.columns:
    pct_missing = countries_of_the_world[col].isnull().mean()
    print(f"{col} - {pct_missing :.1%}")
```

```
Countries - 0.0%
Capital - 3.2%
Population - 0.0%
Area(Sq.Km) - 0.0%
```

In [76]:
```python
# Data statistics
countries_of_the_world.describe().T
```

Out[76]:

	count	mean	std	min	25%	50%	75%
Population	250.0	2.744568e+07	1.168626e+08	0.0	179856.25	4288138.5	15420625.0
Area(Sq.Km)	250.0	5.996369e+05	1.911821e+06	0.0	1174.75	64894.5	372631.5

In [77]:
```python
# Data statistics
countries_of_the_world.describe(include="object").T
```

Out[77]:

	count	unique	top	freq
Countries	250	250	Andorra	1
Capital	242	241	Kingston	2

In [78]:
```python
# Check duplicates row
duplicates = countries_of_the_world.duplicated().sum()
print(f"Number of duplicate rows: {duplicates}")
```

```
Number of duplicate rows: 0
```

In [79]:
```python
# Data profiling
"""
profile = ProfileReport(countries_of_the_world,title="countries_of_the_world",minim

profile.to_file("countries_of_the_world.html")   # Save report to HTML file
"""
```

Out[79]:
```
'\nprofile = ProfileReport(countries_of_the_world,title="countries_of_the_world",m
inimal=True) # Summarize and profile the data\n\nprofile.to_file("countries_of_the
_world.html")   # Save report to HTML file \n'
```

Data Wrangling

In [80]:
```python
# Ensure object columns format and consistency
col_without_numeric = list(countries_of_the_world.select_dtypes(exclude=("float","i
for col in col_without_numeric:
    print(f"Before '{col}': {len(set(countries_of_the_world[col]))} / After '{col}'
```

```
Before 'Countries': 250 / After 'Countries': 250
Before 'Capital': 242 / After 'Capital': 242
```

In [81]:
```python
# Change column names
countries_of_the_world_1 = countries_of_the_world.rename(columns={"Countries":"Coun
```

In [82]:
```
# Drop unnecessary columns
countries_of_the_world_2 = countries_of_the_world_1.drop(columns = ["Capital","Popu
```

In [83]:
```
# Check the countries with 0 crime Rate_per_10000_pop
countries_of_the_world_2[countries_of_the_world_2["Area(Sq.Km)"] == 0]["Country"].v
```

Out[83]:
```
Country
U.S. Minor Outlying Islands    1
Name: count, dtype: int64
```

In [84]:
```
# Remove rows with 0 values in Rate_per_10000_pop column
countries_of_the_world_3 = countries_of_the_world_2[countries_of_the_world_2["Area(
```

In [85]:
```
# Show the final dataset
countries_of_the_world_final = countries_of_the_world_3.copy()
countries_of_the_world_final.head(2)
```

Out[85]:

	Country	Area(Sq.Km)
0	Andorra	468.0
1	United Arab Emirates	82880.0

Final dataset (Merged data)

Merging data

In [86]:
```
# Append violent_and_sexual_crime_final corruption_and_economic_crime_final
crime_final = pd.concat([violent_and_sexual_crime_final,corruption_and_economic_cri
crime_final.head(2)
```

Out[86]:

	Iso3_code	Country	Region	Subregion	Indicator	Crime_category	Year	Crime_rate_p
0	AZE	Azerbaijan	Asia	Western Asia	Violent and Sexual	Serious assault	2003	
1	BEL	Belgium	Europe	Western Europe	Violent and Sexual	Serious assault	2003	

In [87]:
```
# Check data statistics
crime_final.describe().T
```

Out[87]:

	count	mean	std	min	2!
Year	19292.0	2014.437280	5.335623	2003.000000	2010.000(
Crime_rate_per_100000_population	19292.0	158.122869	416.038718	0.000823	2.517

In [88]:

```
# Check data statistics
crime_final.describe(include="object").T
```

Out[88]:

	count	unique	top	freq
Iso3_code	19292	159	AUT	264
Country	19292	159	Austria	264
Region	19292	5	Europe	8902
Subregion	19292	15	Latin America and the Caribbean	4409
Indicator	19292	2	Corruption and Economic	9766
Crime_category	19292	22	Sexual violence: Rape	1983

In [89]:

```
# Merge crime and Population_Age_Sex_final
df_1 = pd.merge(Population_Age_Sex_final,crime_final,on=["Country","Year"],how="inn
df_1.head(2)
```

Out[89]:

	LocID	Country	Year	PopMale	PopFemale	PopTotal	Children_0_14_Male	Children_0_1
0	8	Albania	2005	1551.975	1534.835	3086.81	426.158	
1	8	Albania	2005	1551.975	1534.835	3086.81	426.158	

In [90]:

```
# Check data statistics
df_1.describe()
```

Out[90]:

	LocID	Year	PopMale	PopFemale	PopTotal	Children_0
count	16024.000000	16024.000000	16024.000000	16024.000000	1.602400e+04	160
mean	390.155142	2014.466987	14537.706601	14677.666313	2.921537e+04	37
std	240.224539	5.347049	45642.489141	42968.841790	8.857870e+04	141
min	8.000000	2003.000000	52.493000	52.165000	1.046580e+02	
25%	191.000000	2010.000000	1563.881000	1606.333000	3.170214e+03	3
50%	380.000000	2016.000000	4311.692000	4402.555000	8.762662e+03	7
75%	604.000000	2019.000000	13771.465000	13922.113000	2.768459e+04	30
max	887.000000	2022.000000	726052.413000	687996.940000	1.414049e+06	2008

In [91]:

```
# Check data statistics
df_1.describe(include="object").T
```

Out[91]:

	count	unique	top	freq
Country	16024	130	Austria	264
Iso3_code	16024	130	AUT	264
Region	16024	5	Europe	7309
Subregion	16024	15	Latin America and the Caribbean	3690
Indicator	16024	2	Corruption and Economic	8177
Crime_category	16024	22	Sexual violence: Rape	1634

In [92]:

```
# Merge data with total_government_expenditure_on_education_gdp_final
df_2 = pd.merge(total_government_expenditure_on_education_gdp_final,df_1,on=["Count
df_2.head(2)
```

Out[92]:

	Country	Year	Puplic_education_spending%_of_gdp	LocID	PopMale	PopFemale	PopTo
0	Albania	2005	3.28155	8	1551.975	1534.835	3086
1	Albania	2005	3.28155	8	1551.975	1534.835	3086

In [93]:

```
# Check data statistics
df_2.describe()
```

Out[93]:

	Year	Puplic_education_spending%_of_gdp	LocID	PopMale	
count	13509.000000	13509.000000	13509.000000	13509.000000	13
mean	2014.646976	4.631238	377.356947	14075.253394	14
std	5.132573	1.402082	238.207149	44868.199185	42
min	2003.000000	0.496686	8.000000	53.667000	
25%	2011.000000	3.647770	191.000000	1690.481000	1
50%	2016.000000	4.609030	372.000000	4311.692000	4
75%	2019.000000	5.446427	591.000000	11443.331000	11
max	2022.000000	12.328740	887.000000	726052.413000	687

In [94]:

```
# Check data statistics
df_2.describe(include="object").T
```

Out[94]:

	count	unique	top	freq
Country	13509	119	Austria	264
Iso3_code	13509	119	AUT	264
Region	13509	5	Europe	6417
Subregion	13509	15	Latin America and the Caribbean	2974
Indicator	13509	2	Corruption and Economic	6881
Crime_category	13509	22	Sexual violence: Rape	1350

In [95]:

```
# Merge data with countries_of_the_world_final
df_3 = pd.merge(countries_of_the_world_final,df_2,on=["Country"],how="inner")
df_3.head(2)
```

Out[95]:

	Country	Area(Sq.Km)	Year	Puplic_education_spending%_of_gdp	LocID	PopMale	Popl
0	United Arab Emirates	82880.0	2019	3.86021	784	6766.806	3
1	United Arab Emirates	82880.0	2019	3.86021	784	6766.806	3

In [97]:

```
# Rearrange columns order
new_order = [
    'LocID','Iso3_code','Country', 'Region', 'Subregion',  # Identification and Loc
    'Year', 'Area(Sq.Km)',                                  # Time and Area
```

```
    'PopMale', 'PopFemale', 'PopTotal',           # Total Population Data
    'Male_Female_Ratio',                          # General Ratios
    'Children_0_14_Male', 'Children_0_14_Female', 'Children_0_14_Total', 'Children_
    'YoungAdults_15_24_Male', 'YoungAdults_15_24_Female', 'YoungAdults_15_24_Total'
    'WorkingAdults_25_44_Male', 'WorkingAdults_25_44_Female', 'WorkingAdults_25_44_
    'MatureAdults_45_64_Male', 'MatureAdults_45_64_Female', 'MatureAdults_45_64_Tot
    'Elderly_65Plus_Male', 'Elderly_65Plus_Female', 'Elderly_65Plus_Total', 'Elderl
    'Puplic_education_spending%_of_gdp',          # Education and Economy
    'Indicator', 'Crime_category', 'Crime_rate_per_100000_population'  # Crime and
]
df_final = df_3[new_order]
df_final.head(2)
```

Out[97]:

	LocID	Iso3_code	Country	Region	Subregion	Year	Area(Sq.Km)	PopMale	PopFemal
0	784	ARE	United Arab Emirates	Asia	Western Asia	2019	82880.0	6766.806	3003.7
1	784	ARE	United Arab Emirates	Asia	Western Asia	2019	82880.0	6766.806	3003.7

In [98]:

```
# Remove countries that do not appear in at least 15 different years
country_year_counts = df_final.groupby('Country')['Year'].nunique()
valid_countries = country_year_counts[country_year_counts >= 15].index
df_final = df_final[df_final['Country'].isin(valid_countries)]
```

Data Profiling

In [99]:

```
# Data information
df_final.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Index: 9462 entries, 36 to 13072
Data columns (total 35 columns):
 #   Column                            Non-Null Count  Dtype  
---  ------                            --------------  -----  
 0   LocID                             9462 non-null   int64  
 1   Iso3_code                         9462 non-null   object 
 2   Country                           9462 non-null   object 
 3   Region                            9462 non-null   object 
 4   Subregion                         9462 non-null   object 
 5   Year                              9462 non-null   int64  
 6   Area(Sq.Km)                       9462 non-null   float64
 7   PopMale                           9462 non-null   float64
 8   PopFemale                         9462 non-null   float64
 9   PopTotal                          9462 non-null   float64
 10  Male_Female_Ratio                 9462 non-null   float64
 11  Children_0_14_Male                9462 non-null   float64
 12  Children_0_14_Female              9462 non-null   float64
 13  Children_0_14_Total               9462 non-null   float64
 14  Children_0_14_MF_Ratio            9462 non-null   float64
 15  YoungAdults_15_24_Male            9462 non-null   float64
 16  YoungAdults_15_24_Female          9462 non-null   float64
 17  YoungAdults_15_24_Total           9462 non-null   float64
 18  YoungAdults_15_24_MF_Ratio        9462 non-null   float64
 19  WorkingAdults_25_44_Male          9462 non-null   float64
 20  WorkingAdults_25_44_Female        9462 non-null   float64
 21  WorkingAdults_25_44_Total         9462 non-null   float64
 22  WorkingAdults_25_44_MF_Ratio      9462 non-null   float64
 23  MatureAdults_45_64_Male           9462 non-null   float64
 24  MatureAdults_45_64_Female         9462 non-null   float64
 25  MatureAdults_45_64_Total          9462 non-null   float64
 26  MatureAdults_45_64_MF_Ratio       9462 non-null   float64
 27  Elderly_65Plus_Male               9462 non-null   float64
 28  Elderly_65Plus_Female             9462 non-null   float64
 29  Elderly_65Plus_Total              9462 non-null   float64
 30  Elderly_65Plus_MF_Ratio           9462 non-null   float64
 31  Puplic_education_spending%_of_gdp 9462 non-null   float64
 32  Indicator                         9462 non-null   object 
 33  Crime_category                    9462 non-null   object 
 34  Crime_rate_per_100000_population  9462 non-null   float64
dtypes: float64(27), int64(2), object(6)
memory usage: 2.6+ MB
```

In [100…

```
# Check data statistics
df_final.describe()
```

Out[100…

	LocID	Year	Area(Sq.Km)	PopMale	PopFemale	PopTot
count	9462.000000	9462.000000	9.462000e+03	9462.000000	9462.000000	9462.00000
mean	366.446417	2014.536039	6.302700e+05	10461.735801	10881.576218	21343.31202
std	237.240180	5.174895	1.805733e+06	16305.837923	16943.210763	33247.85765
min	8.000000	2003.000000	3.160000e+02	132.170000	143.113000	275.28300
25%	188.000000	2011.000000	4.309400e+04	1652.003000	1784.026000	3424.13900
50%	352.000000	2016.000000	9.303000e+04	4028.900000	4190.586000	8216.81000
75%	604.000000	2019.000000	3.570210e+05	10232.553000	10654.676750	20906.39200
max	858.000000	2022.000000	9.984670e+06	104435.783000	108123.626000	212559.40900

In [101…

```
# Check data statistics
df_final.describe(include="object").T
```

Out[101…

	count	unique	top	freq
Iso3_code	9462	51	AUT	264
Country	9462	51	Austria	264
Region	9462	5	Europe	5916
Subregion	9462	12	Northern Europe	1773
Indicator	9462	2	Corruption and Economic	5081
Crime_category	9462	22	Theft	892

In [102…

```
# Show the first 5 rows
df_final.head()
```

Out[102…

	LocID	Iso3_code	Country	Region	Subregion	Year	Area(Sq.Km)	PopMale	PopFem
36	8	ALB	Albania	Europe	Southern Europe	2005	28748.0	1551.975	1534.8
37	8	ALB	Albania	Europe	Southern Europe	2005	28748.0	1551.975	1534.8
38	8	ALB	Albania	Europe	Southern Europe	2005	28748.0	1551.975	1534.8
39	8	ALB	Albania	Europe	Southern Europe	2005	28748.0	1551.975	1534.8
40	8	ALB	Albania	Europe	Southern Europe	2005	28748.0	1551.975	1534.8

In [103…

```
# Show the last 5 rows
df_final.tail()
```

Out[103…

	LocID	Iso3_code	Country	Region	Subregion	Year	Area(Sq.Km)	PopMale	Po
13068	858	URY	Uruguay	Americas	Latin America and the Caribbean	2022	176220.0	1690.481	1
13069	858	URY	Uruguay	Americas	Latin America and the Caribbean	2022	176220.0	1690.481	1
13070	858	URY	Uruguay	Americas	Latin America and the Caribbean	2022	176220.0	1690.481	1
13071	858	URY	Uruguay	Americas	Latin America and the Caribbean	2022	176220.0	1690.481	1
13072	858	URY	Uruguay	Americas	Latin America and the Caribbean	2022	176220.0	1690.481	1

In [104…

```
#loop through the columns and check the missing values
for col in df_final.columns:
    pct_missing = df_final[col].isnull().mean()
    print(f"{col} - {pct_missing :.1%}")
```

```
LocID - 0.0%
Iso3_code - 0.0%
Country - 0.0%
Region - 0.0%
Subregion - 0.0%
Year - 0.0%
Area(Sq.Km) - 0.0%
PopMale - 0.0%
PopFemale - 0.0%
PopTotal - 0.0%
Male_Female_Ratio - 0.0%
Children_0_14_Male - 0.0%
Children_0_14_Female - 0.0%
Children_0_14_Total - 0.0%
Children_0_14_MF_Ratio - 0.0%
YoungAdults_15_24_Male - 0.0%
YoungAdults_15_24_Female - 0.0%
YoungAdults_15_24_Total - 0.0%
YoungAdults_15_24_MF_Ratio - 0.0%
WorkingAdults_25_44_Male - 0.0%
WorkingAdults_25_44_Female - 0.0%
WorkingAdults_25_44_Total - 0.0%
WorkingAdults_25_44_MF_Ratio - 0.0%
MatureAdults_45_64_Male - 0.0%
MatureAdults_45_64_Female - 0.0%
MatureAdults_45_64_Total - 0.0%
MatureAdults_45_64_MF_Ratio - 0.0%
Elderly_65Plus_Male - 0.0%
Elderly_65Plus_Female - 0.0%
Elderly_65Plus_Total - 0.0%
Elderly_65Plus_MF_Ratio - 0.0%
Puplic_education_spending%_of_gdp - 0.0%
Indicator - 0.0%
Crime_category - 0.0%
Crime_rate_per_100000_population - 0.0%
```

In [105…

```
# Check duplicates row
duplicates = df_final.duplicated().sum()
print(f"Number of duplicate rows: {duplicates}")
```

```
Number of duplicate rows: 0
```

In [106…

```
# Data profiling
"""
profile = ProfileReport(df_final,title="df_final",minimal=True) # Summarize and pro

profile.to_file("df_final.html")   # Save report to HTML file
"""
```

Out[106…

```
'\nprofile = ProfileReport(df_final,title="df_final",minimal=True) # Summarize and
profile the data\n\nprofile.to_file("df_final.html")   # Save report to HTML file
\n'
```

In [ ]: