

# Visualizing world's population from 1960

August 8, 2024

## 1 Importing libraries

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

## 2 Loading the dataset

```
[ ]: df = pd.read_csv('/Users/benfathe1/Documents/Work/PRODIGY_DS_01/PRODIGY_DS_01/
      ↪Data/API_SP.POP.TOTL_DS2_en_csv_v2_2778915.csv', skiprows=4)
      df
```

```
[ ]:          Country Name Country Code    Indicator Name \
0                  Aruba        ABW  Population, total
1  Africa Eastern and Southern        AFE  Population, total
2          Afghanistan        AFG  Population, total
3  Africa Western and Central        AFW  Population, total
4          Angola        AGO  Population, total
..                   ...
261                 Kosovo        XKX  Population, total
262           Yemen, Rep.        YEM  Population, total
263          South Africa        ZAF  Population, total
264            Zambia        ZMB  Population, total
265         Zimbabwe        ZWE  Population, total
```

	Indicator Code	1960	1961	1962	1963	\
0	SP.POP.TOTL	54608.0	55811.0	56682.0	57475.0	
1	SP.POP.TOTL	130692579.0	134169237.0	137835590.0	141630546.0	
2	SP.POP.TOTL	8622466.0	8790140.0	8969047.0	9157465.0	
3	SP.POP.TOTL	97256290.0	99314028.0	101445032.0	103667517.0	
4	SP.POP.TOTL	5357195.0	5441333.0	5521400.0	5599827.0	
..	..	..	..	..	..	
261	SP.POP.TOTL	990150.0	1014211.0	1038618.0	1063175.0	
262	SP.POP.TOTL	5542459.0	5646668.0	5753386.0	5860197.0	
263	SP.POP.TOTL	16520441.0	16989464.0	17503133.0	18042215.0	

264	SP.POP.TOTL	3119430.0	3219451.0	3323427.0	3431381.0	
265	SP.POP.TOTL	3806310.0	3925952.0	4049778.0	4177931.0	
	1964	1965	...	2015	2016	2017 \
0	58178.0	58782.0	...	104257.0	104874.0	105439.0
1	145605995.0	149742351.0	...	600008424.0	616377605.0	632746570.0
2	9355514.0	9565147.0	...	33753499.0	34636207.0	35643418.0
3	105959979.0	108336203.0	...	408690375.0	419778384.0	431138704.0
4	5673199.0	5736582.0	...	28127721.0	29154746.0	30208628.0
..	...	...	...	...	...	...
261	1087700.0	1111812.0	...	1788196.0	1777557.0	1791003.0
262	5973803.0	6097298.0	...	28516545.0	29274002.0	30034389.0
263	18603097.0	19187194.0	...	55876504.0	56422274.0	56641209.0
264	3542764.0	3658024.0	...	16248230.0	16767761.0	17298054.0
265	4310332.0	4447149.0	...	14154937.0	14452704.0	14751101.0
	2018	2019	2020	2021	2022	\
0	105962.0	106442.0	106585.0	106537.0	106445.0	
1	649757148.0	667242986.0	685112979.0	702977106.0	720859132.0	
2	36686784.0	37769499.0	38972230.0	40099462.0	41128771.0	
3	442646825.0	454306063.0	466189102.0	478185907.0	490330870.0	
4	31273533.0	32353588.0	33428486.0	34503774.0	35588987.0	
..	...	...	...	...	...	...
261	1797085.0	1788878.0	1790133.0	1786038.0	1768086.0	
262	30790513.0	31546691.0	32284046.0	32981641.0	33696614.0	
263	57339635.0	58087055.0	58801927.0	59392255.0	59893885.0	
264	17835893.0	18380477.0	18927715.0	19473125.0	20017675.0	
265	15052184.0	15354608.0	15669666.0	15993524.0	16320537.0	
	2023	Unnamed: 68				
0	106277.0	NaN				
1	739108306.0	NaN				
2	42239854.0	NaN				
3	502789511.0	NaN				
4	36684202.0	NaN				
..	...	...				
261	1756374.0	NaN				
262	34449825.0	NaN				
263	60414495.0	NaN				
264	20569737.0	NaN				
265	16665409.0	NaN				

[266 rows x 69 columns]

## 2.1 Data Cleaning

```
[ ]: df.shape
[ ]: (266, 69)
[ ]: df.columns
[ ]: Index(['Country Name', 'Country Code', 'Indicator Name', 'Indicator Code',
       '1960', '1961', '1962', '1963', '1964', '1965', '1966', '1967', '1968',
       '1969', '1970', '1971', '1972', '1973', '1974', '1975', '1976', '1977',
       '1978', '1979', '1980', '1981', '1982', '1983', '1984', '1985', '1986',
       '1987', '1988', '1989', '1990', '1991', '1992', '1993', '1994', '1995',
       '1996', '1997', '1998', '1999', '2000', '2001', '2002', '2003', '2004',
       '2005', '2006', '2007', '2008', '2009', '2010', '2011', '2012', '2013',
       '2014', '2015', '2016', '2017', '2018', '2019', '2020', '2021', '2022',
       '2023', 'Unnamed: 68'],
      dtype='object')

[ ]: # Delete the last column
df = df.drop(columns=['Unnamed: 68'])

# Verify the change
print(df.shape)
print(df.columns)

(266, 68)
Index(['Country Name', 'Country Code', 'Indicator Name', 'Indicator Code',
       '1960', '1961', '1962', '1963', '1964', '1965', '1966', '1967', '1968',
       '1969', '1970', '1971', '1972', '1973', '1974', '1975', '1976', '1977',
       '1978', '1979', '1980', '1981', '1982', '1983', '1984', '1985', '1986',
       '1987', '1988', '1989', '1990', '1991', '1992', '1993', '1994', '1995',
       '1996', '1997', '1998', '1999', '2000', '2001', '2002', '2003', '2004',
       '2005', '2006', '2007', '2008', '2009', '2010', '2011', '2012', '2013',
       '2014', '2015', '2016', '2017', '2018', '2019', '2020', '2021', '2022',
       '2023'],
      dtype='object')

[ ]: df.info()

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 266 entries, 0 to 265
Data columns (total 68 columns):
 #   Column           Non-Null Count  Dtype  
 ---  -- 
 0   Country Name    266 non-null    object 
 1   Country Code    266 non-null    object 
 2   Indicator Name  266 non-null    object 
 3   Indicator Code  266 non-null    object
```

4	1960	264	non-null	float64
5	1961	264	non-null	float64
6	1962	264	non-null	float64
7	1963	264	non-null	float64
8	1964	264	non-null	float64
9	1965	264	non-null	float64
10	1966	264	non-null	float64
11	1967	264	non-null	float64
12	1968	264	non-null	float64
13	1969	264	non-null	float64
14	1970	264	non-null	float64
15	1971	264	non-null	float64
16	1972	264	non-null	float64
17	1973	264	non-null	float64
18	1974	264	non-null	float64
19	1975	264	non-null	float64
20	1976	264	non-null	float64
21	1977	264	non-null	float64
22	1978	264	non-null	float64
23	1979	264	non-null	float64
24	1980	264	non-null	float64
25	1981	264	non-null	float64
26	1982	264	non-null	float64
27	1983	264	non-null	float64
28	1984	264	non-null	float64
29	1985	264	non-null	float64
30	1986	264	non-null	float64
31	1987	264	non-null	float64
32	1988	264	non-null	float64
33	1989	264	non-null	float64
34	1990	265	non-null	float64
35	1991	265	non-null	float64
36	1992	265	non-null	float64
37	1993	265	non-null	float64
38	1994	265	non-null	float64
39	1995	265	non-null	float64
40	1996	265	non-null	float64
41	1997	265	non-null	float64
42	1998	265	non-null	float64
43	1999	265	non-null	float64
44	2000	265	non-null	float64
45	2001	265	non-null	float64
46	2002	265	non-null	float64
47	2003	265	non-null	float64
48	2004	265	non-null	float64
49	2005	265	non-null	float64
50	2006	265	non-null	float64
51	2007	265	non-null	float64

```
52 2008          265 non-null    float64
53 2009          265 non-null    float64
54 2010          265 non-null    float64
55 2011          265 non-null    float64
56 2012          265 non-null    float64
57 2013          265 non-null    float64
58 2014          265 non-null    float64
59 2015          265 non-null    float64
60 2016          265 non-null    float64
61 2017          265 non-null    float64
62 2018          265 non-null    float64
63 2019          265 non-null    float64
64 2020          265 non-null    float64
65 2021          265 non-null    float64
66 2022          265 non-null    float64
67 2023          265 non-null    float64
```

dtypes: float64(64), object(4)

memory usage: 141.4+ KB

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

	1960	1961	1962	1963	1964	\
count	2.640000e+02	2.640000e+02	2.640000e+02	2.640000e+02	2.640000e+02	
mean	1.157939e+08	1.173869e+08	1.195401e+08	1.222050e+08	1.248922e+08	
std	3.639920e+08	3.684672e+08	3.751049e+08	3.837174e+08	3.923714e+08	
min	2.646000e+03	2.888000e+03	3.171000e+03	3.481000e+03	3.811000e+03	
25%	5.132212e+05	5.231345e+05	5.337595e+05	5.449288e+05	5.566630e+05	
50%	3.708088e+06	3.816540e+06	3.931214e+06	4.033994e+06	4.112910e+06	
75%	2.670606e+07	2.748694e+07	2.830289e+07	2.914708e+07	3.001684e+07	
max	3.031517e+09	3.072470e+09	3.126894e+09	3.193470e+09	3.260480e+09	

	1965	1966	1967	1968	1969	\
count	2.640000e+02	2.640000e+02	2.640000e+02	2.640000e+02	2.640000e+02	
mean	1.276182e+08	1.304676e+08	1.333152e+08	1.362430e+08	1.392759e+08	
std	4.011556e+08	4.104328e+08	4.196670e+08	4.291879e+08	4.390998e+08	
min	4.161000e+03	4.531000e+03	4.930000e+03	5.354000e+03	5.646000e+03	
25%	5.651150e+05	5.691470e+05	5.773872e+05	5.832700e+05	5.875942e+05	
50%	4.194930e+06	4.257383e+06	4.317222e+06	4.410692e+06	4.515734e+06	
75%	3.084892e+07	3.163010e+07	3.209247e+07	3.249927e+07	3.277149e+07	
max	3.328243e+09	3.398510e+09	3.468395e+09	3.540186e+09	3.614593e+09	

	2014	2015	2016	2017	\
count	... 2.650000e+02	2.650000e+02	2.650000e+02	2.650000e+02	
mean	... 2.948007e+08	2.986442e+08	3.024871e+08	3.063370e+08	
std	... 9.224214e+08	9.336474e+08	9.448081e+08	9.559803e+08	
min	... 1.089900e+04	1.087700e+04	1.085200e+04	1.082800e+04	
25%	... 1.743309e+06	1.788196e+06	1.777557e+06	1.791003e+06	

```

50% ... 1.028212e+07 1.035808e+07 1.032545e+07 1.030030e+07
75% ... 6.078914e+07 6.073058e+07 6.062750e+07 6.053671e+07
max ... 7.317305e+09 7.404251e+09 7.490956e+09 7.577110e+09

          2018      2019      2020      2021      2022 \
count  2.650000e+02 2.650000e+02 2.650000e+02 2.650000e+02 2.650000e+02
mean   3.101259e+08 3.138348e+08 3.174293e+08 3.206783e+08 3.236218e+08
std    9.668651e+08 9.774204e+08 9.875137e+08 9.965683e+08 1.004474e+09
min    1.086500e+04 1.095600e+04 1.106900e+04 1.120400e+04 1.131200e+04
25%   1.797085e+06 1.788878e+06 1.790133e+06 1.786038e+06 1.768086e+06
50%   1.039533e+07 1.044767e+07 1.060623e+07 1.050577e+07 1.048694e+07
75%   6.042176e+07 5.987258e+07 6.170452e+07 6.358833e+07 6.549775e+07
max   7.661178e+09 7.742725e+09 7.821272e+09 7.888964e+09 7.951595e+09

          2023
count  2.650000e+02
mean   3.269710e+08
std    1.013469e+09
min    1.139600e+04
25%   1.756374e+06
50%   1.059380e+07
75%   6.743811e+07
max   8.024997e+09

[8 rows x 64 columns]

```

```
[ ]: df.duplicated().sum()
```

```
[ ]: np.int64(0)
```

```
[ ]: df.isna().sum().any()
```

```
[ ]: np.True_
```

```
[ ]: df=df.fillna(method='ffill')
df
```

```
/var/folders/9s/09g1kdgn5p3fdcwfbo_r_hrzh0000gn/T/ipykernel_23211/2469475080.py:1
: FutureWarning: DataFrame.fillna with 'method' is deprecated and will raise in
a future version. Use obj.ffill() or obj.bfill() instead.
df=df.fillna(method='ffill')
```

```
[ ]:          Country Name Country Code      Indicator Name \
0                      Aruba        ABW  Population, total
1  Africa Eastern and Southern           AFE  Population, total
2                  Afghanistan         AFG  Population, total
3  Africa Western and Central          AFW  Population, total
4                      Angola        AGO  Population, total
```

..		...	...	...	...	\\
261		Kosovo	XKK	Population, total		
262		Yemen, Rep.	YEM	Population, total		
263		South Africa	ZAF	Population, total		
264		Zambia	ZMB	Population, total		
265		Zimbabwe	ZWE	Population, total		
	Indicator Code	1960	1961	1962	1963	\\
0	SP.POP.TOTL	54608.0	55811.0	56682.0	57475.0	
1	SP.POP.TOTL	130692579.0	134169237.0	137835590.0	141630546.0	
2	SP.POP.TOTL	8622466.0	8790140.0	8969047.0	9157465.0	
3	SP.POP.TOTL	97256290.0	99314028.0	101445032.0	103667517.0	
4	SP.POP.TOTL	5357195.0	5441333.0	5521400.0	5599827.0	
..	..	..	..	..	..	
261	SP.POP.TOTL	990150.0	1014211.0	1038618.0	1063175.0	
262	SP.POP.TOTL	5542459.0	5646668.0	5753386.0	5860197.0	
263	SP.POP.TOTL	16520441.0	16989464.0	17503133.0	18042215.0	
264	SP.POP.TOTL	3119430.0	3219451.0	3323427.0	3431381.0	
265	SP.POP.TOTL	3806310.0	3925952.0	4049778.0	4177931.0	
	1964	1965	...	2014	2015	2016 \\
0	58178.0	58782.0	...	103594.0	104257.0	104874.0
1	145605995.0	149742351.0	...	583651101.0	600008424.0	616377605.0
2	9355514.0	9565147.0	...	32716210.0	33753499.0	34636207.0
3	105959979.0	108336203.0	...	397855507.0	408690375.0	419778384.0
4	5673199.0	5736582.0	...	27128337.0	28127721.0	29154746.0
..	..	..	..	..	..	
261	1087700.0	1111812.0	...	1812771.0	1788196.0	1777557.0
262	5973803.0	6097298.0	...	27753304.0	28516545.0	29274002.0
263	18603097.0	19187194.0	...	54729551.0	55876504.0	56422274.0
264	3542764.0	3658024.0	...	15737793.0	16248230.0	16767761.0
265	4310332.0	4447149.0	...	13855753.0	14154937.0	14452704.0
	2017	2018	2019	2020	2021	\\
0	105439.0	105962.0	106442.0	106585.0	106537.0	
1	632746570.0	649757148.0	667242986.0	685112979.0	702977106.0	
2	35643418.0	36686784.0	37769499.0	38972230.0	40099462.0	
3	431138704.0	442646825.0	454306063.0	466189102.0	478185907.0	
4	30208628.0	31273533.0	32353588.0	33428486.0	34503774.0	
..	..	..	..	..	..	
261	1791003.0	1797085.0	1788878.0	1790133.0	1786038.0	
262	30034389.0	30790513.0	31546691.0	32284046.0	32981641.0	
263	56641209.0	57339635.0	58087055.0	58801927.0	59392255.0	
264	17298054.0	17835893.0	18380477.0	18927715.0	19473125.0	
265	14751101.0	15052184.0	15354608.0	15669666.0	15993524.0	
	2022	2023				

```
0      106445.0    106277.0
1    720859132.0   739108306.0
2    41128771.0    42239854.0
3   490330870.0   502789511.0
4   35588987.0    36684202.0
..
261     ...       ...
262  1768086.0    1756374.0
263  33696614.0   34449825.0
263  59893885.0   60414495.0
264  20017675.0   20569737.0
265  16320537.0   16665409.0
```

```
[266 rows x 68 columns]
```

```
[ ]: df.isna().sum().any()
```

```
[ ]: np.False_
```

```
[ ]: df.drop(['Indicator Name','Indicator Code','Country Code'],axis=1,inplace=True)
```

### 3 Preparing the data for visualization

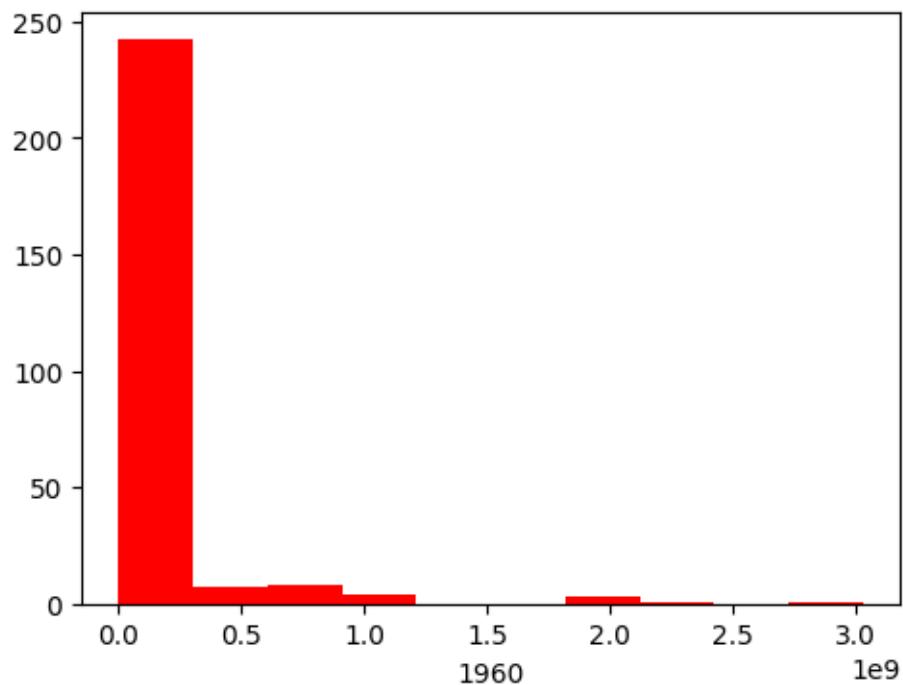
```
[ ]: df.columns
```

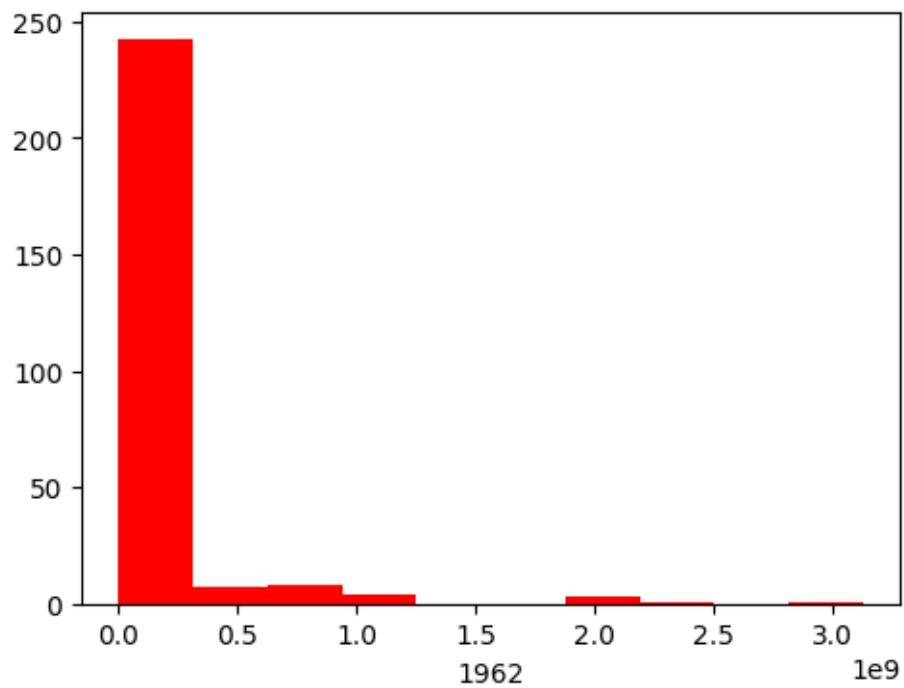
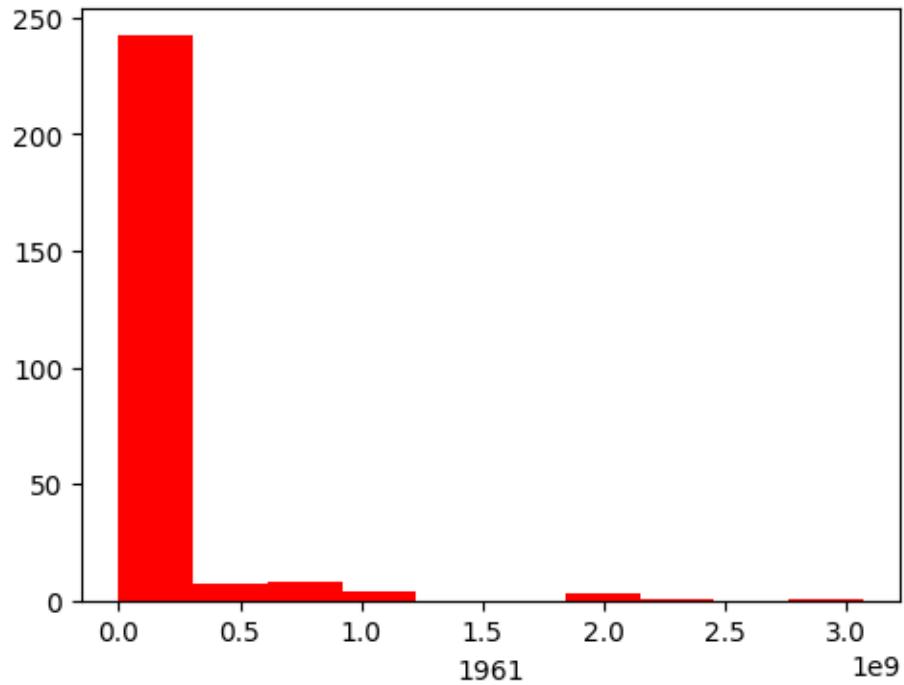
```
[ ]: Index(['Country Name', '1960', '1961', '1962', '1963', '1964', '1965', '1966',
       '1967', '1968', '1969', '1970', '1971', '1972', '1973', '1974', '1975',
       '1976', '1977', '1978', '1979', '1980', '1981', '1982', '1983', '1984',
       '1985', '1986', '1987', '1988', '1989', '1990', '1991', '1992', '1993',
       '1994', '1995', '1996', '1997', '1998', '1999', '2000', '2001', '2002',
       '2003', '2004', '2005', '2006', '2007', '2008', '2009', '2010', '2011',
       '2012', '2013', '2014', '2015', '2016', '2017', '2018', '2019', '2020',
       '2021', '2022', '2023'],
      dtype='object')
```

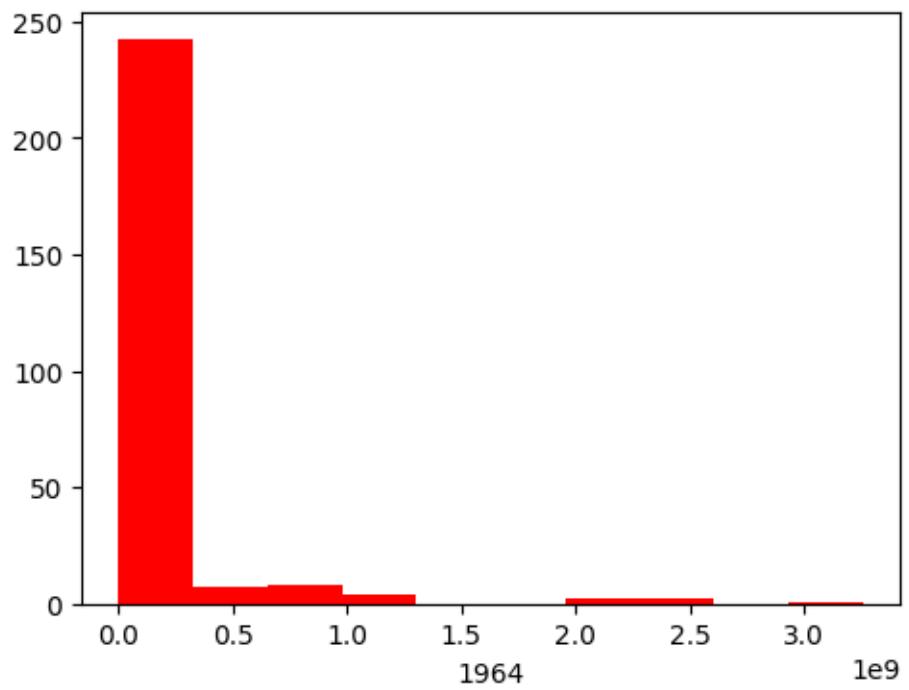
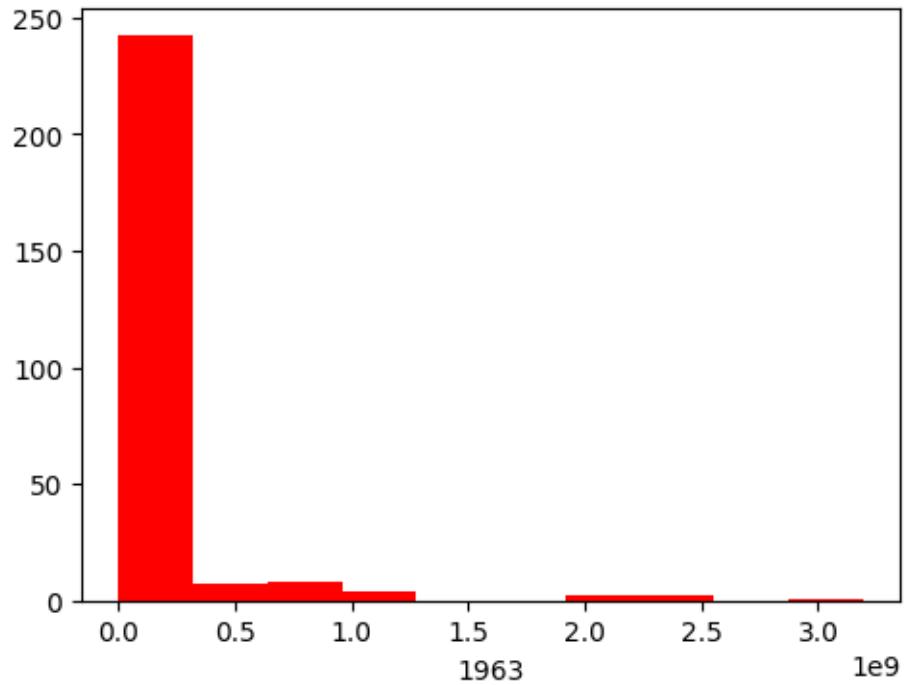
```
[ ]: cols=['1960', '1961', '1962', '1963', '1964', '1965', '1966',
       '1967', '1968', '1969', '1970', '1971', '1972', '1973', '1974', '1975',
       '1976', '1977', '1978', '1979', '1980', '1981', '1982', '1983', '1984',
       '1985', '1986', '1987', '1988', '1989', '1990', '1991', '1992', '1993',
       '1994', '1995', '1996', '1997', '1998', '1999', '2000', '2001', '2002',
       '2003', '2004', '2005', '2006', '2007', '2008', '2009', '2010', '2011',
       '2012', '2013', '2014', '2015', '2016', '2017', '2018', '2019', '2020',
       '2021', '2022', '2023']
```

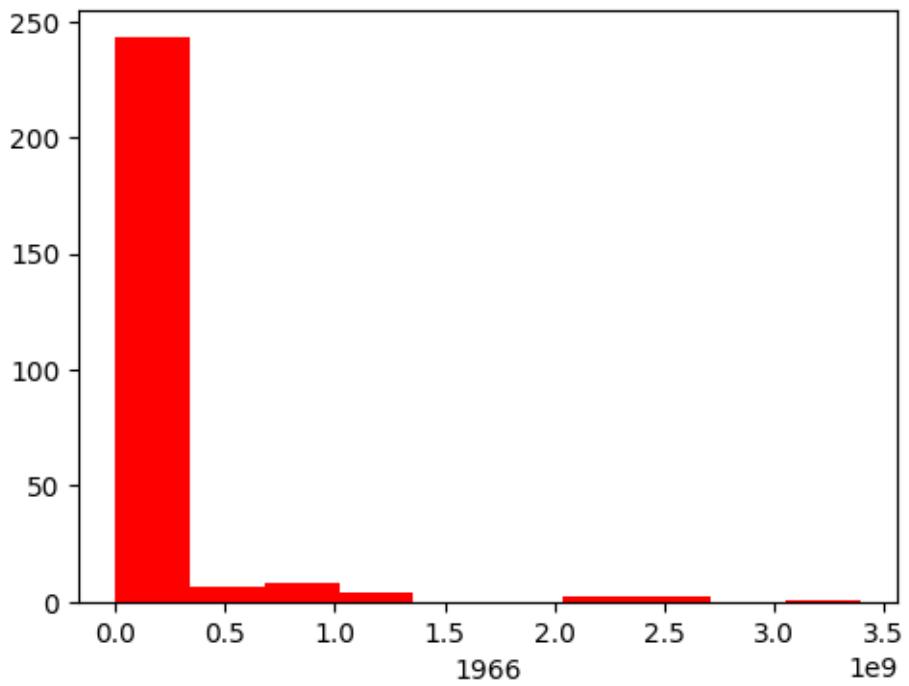
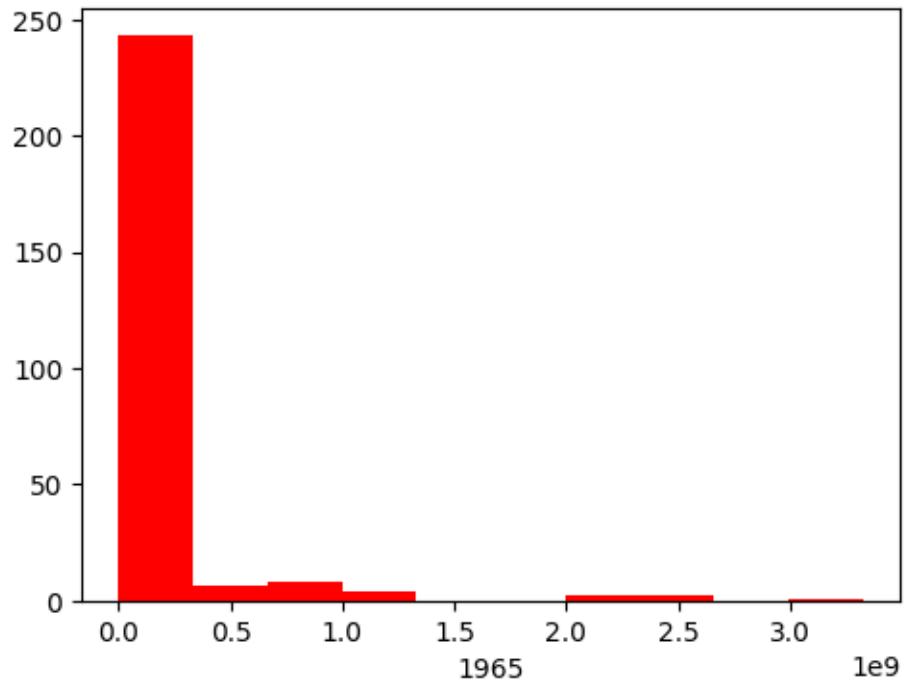
## 4 Visualizing the data

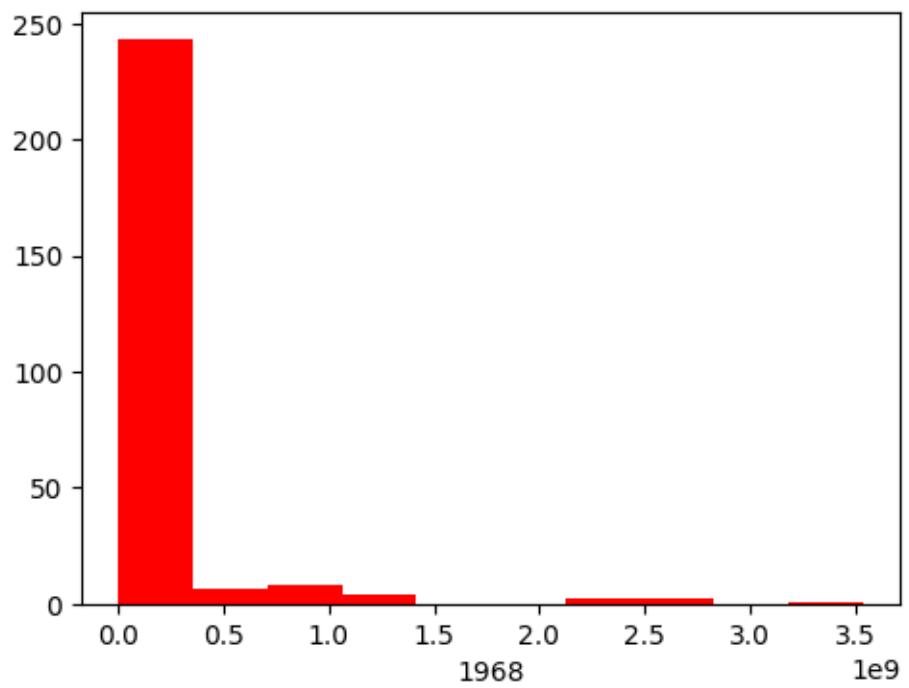
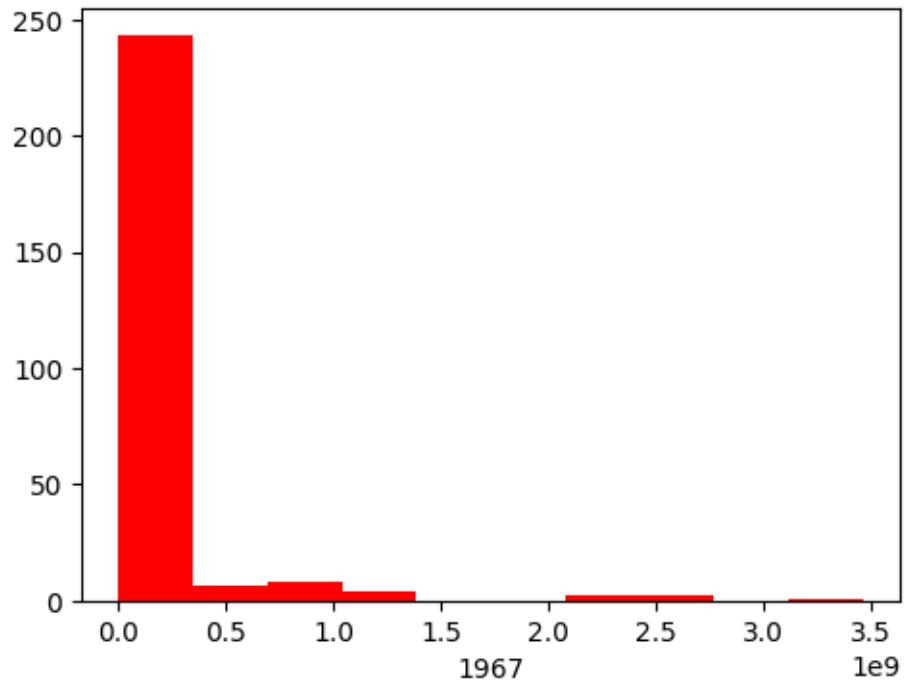
```
[ ]: for col in cols:  
    fig = plt.figure(figsize=(5.5, 4))  
    plt.hist(df[col], color='red', bins=10)  
    plt.xlabel(col)  
    plt.show()
```

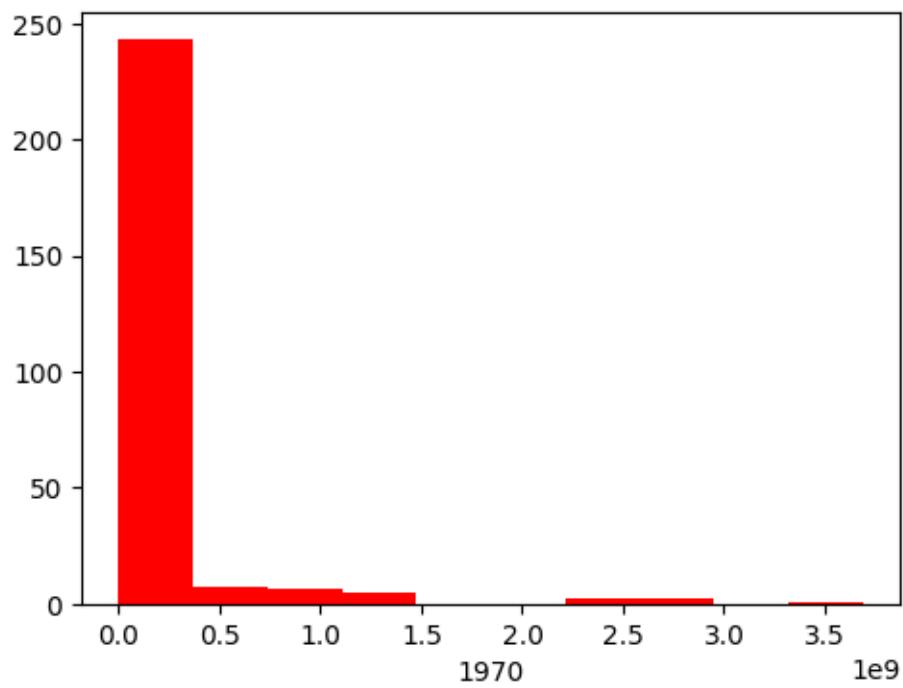
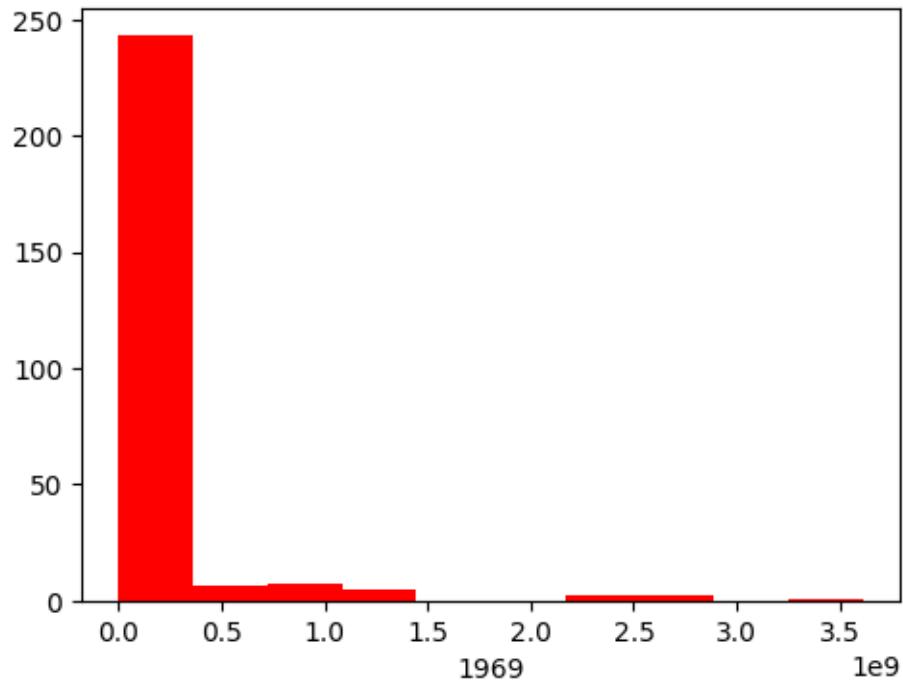


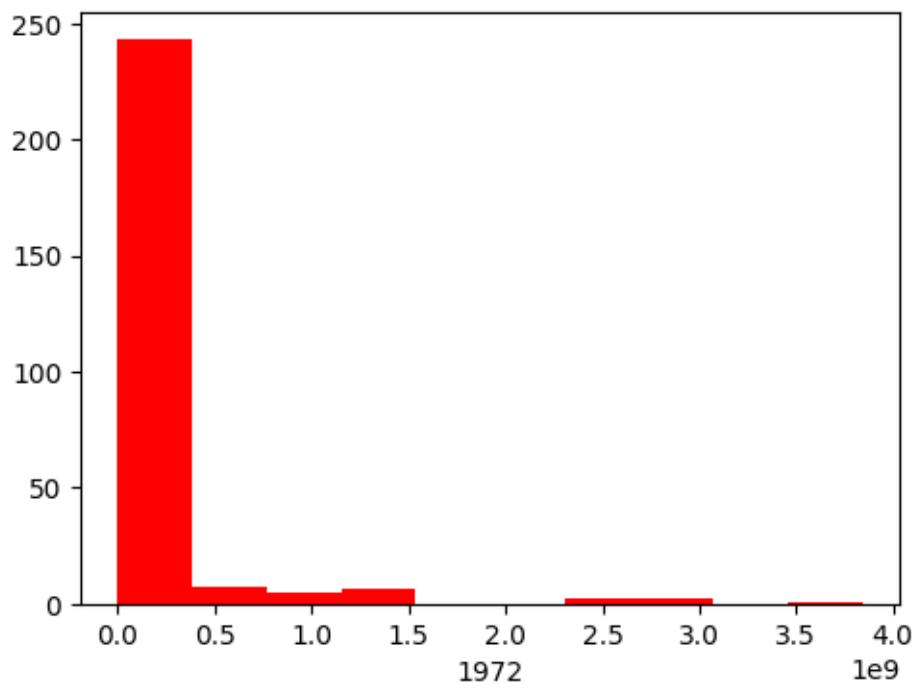
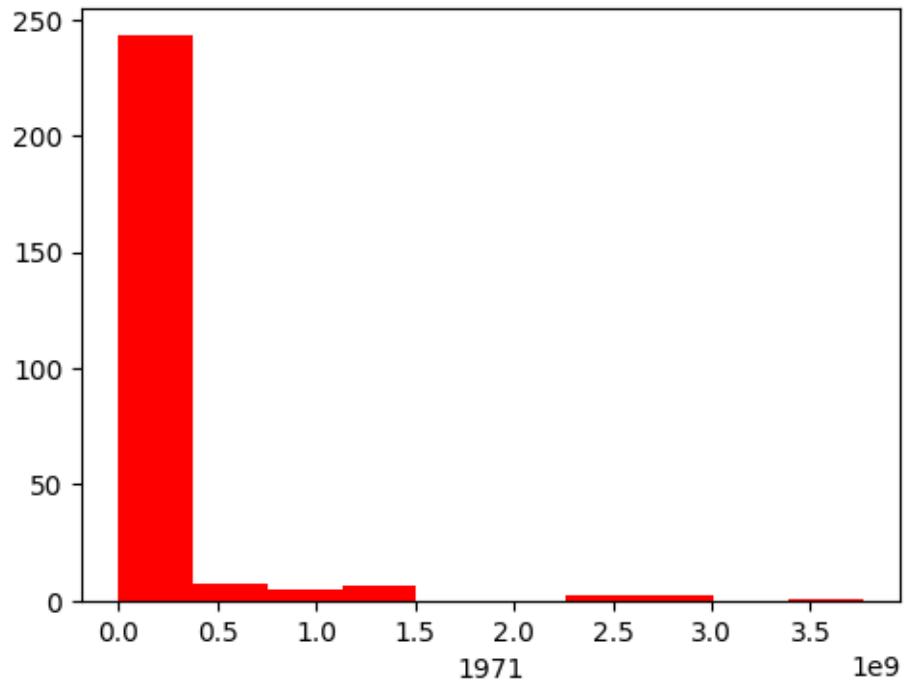


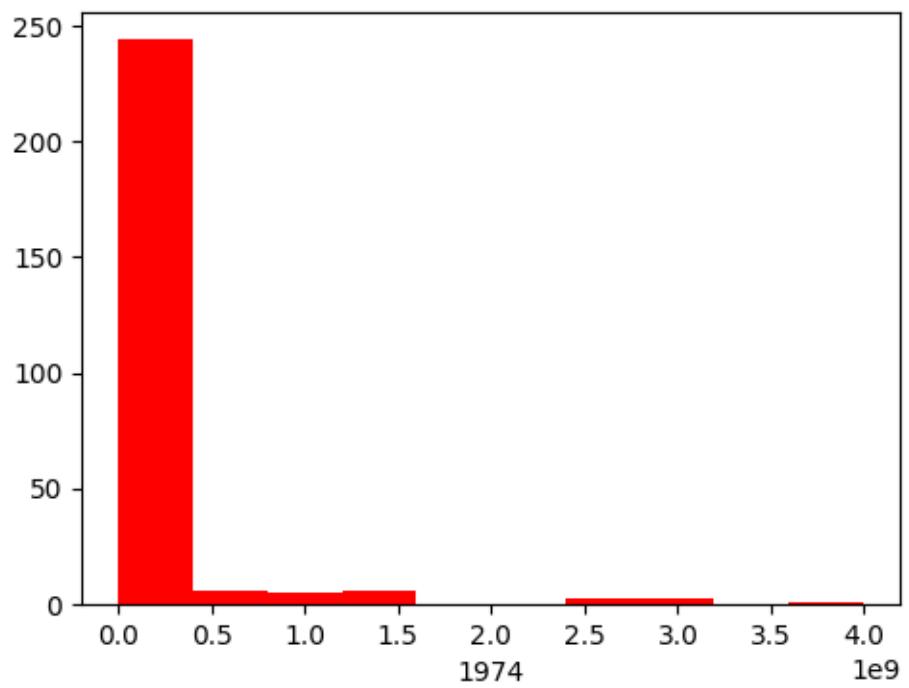
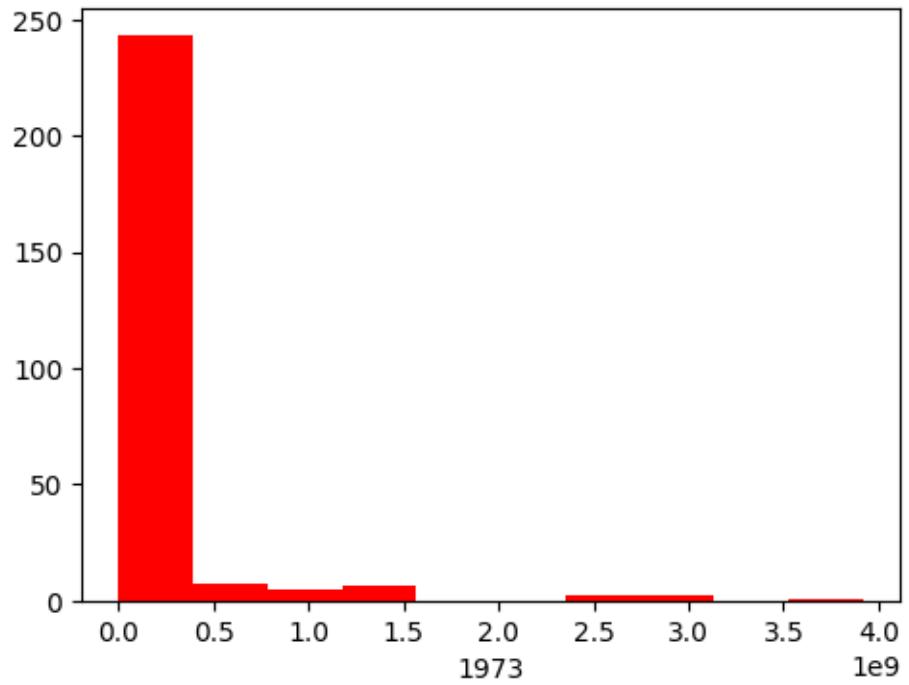


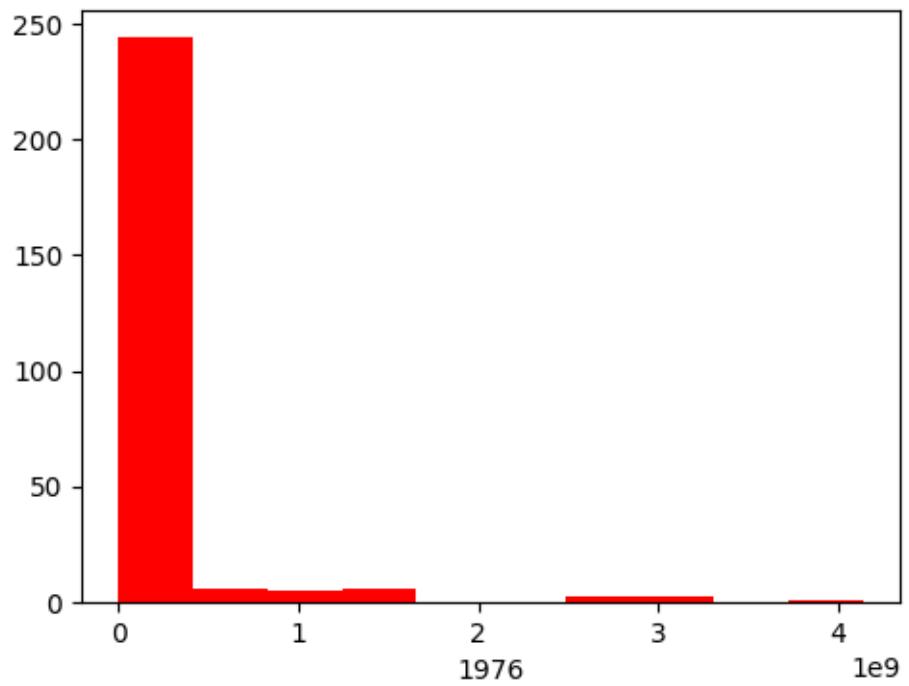
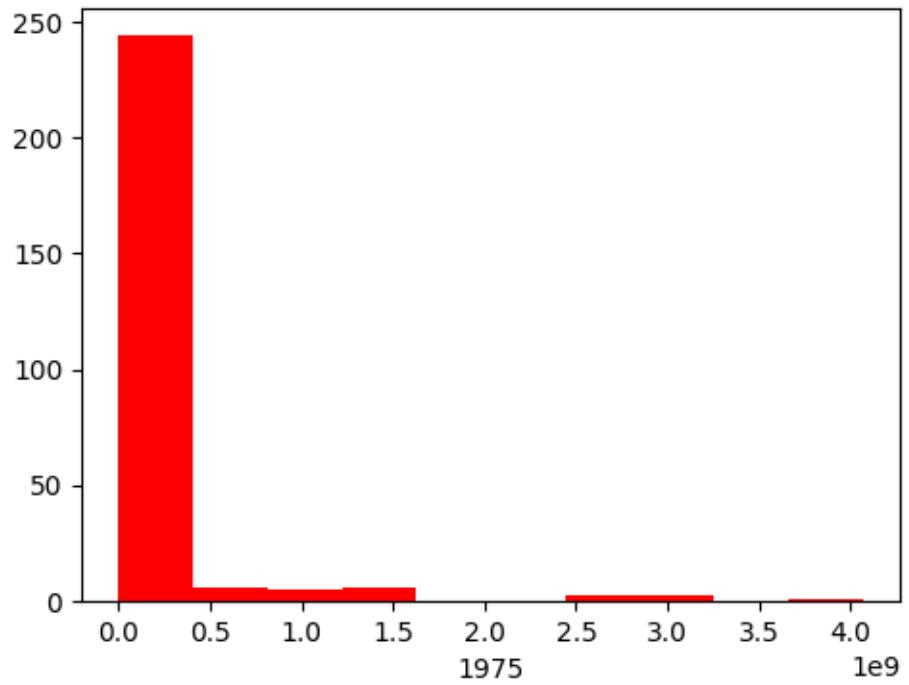


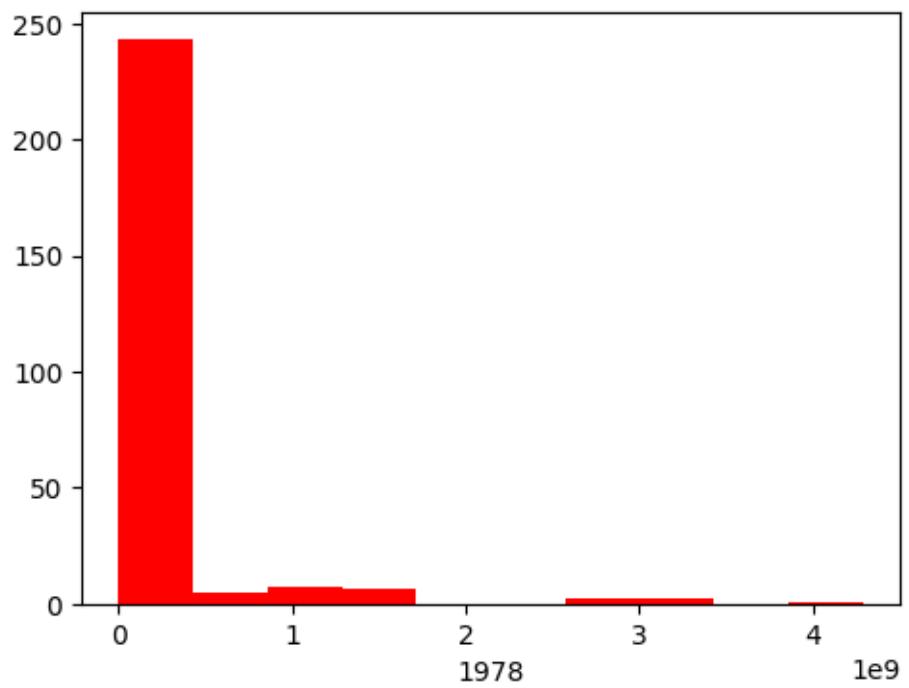
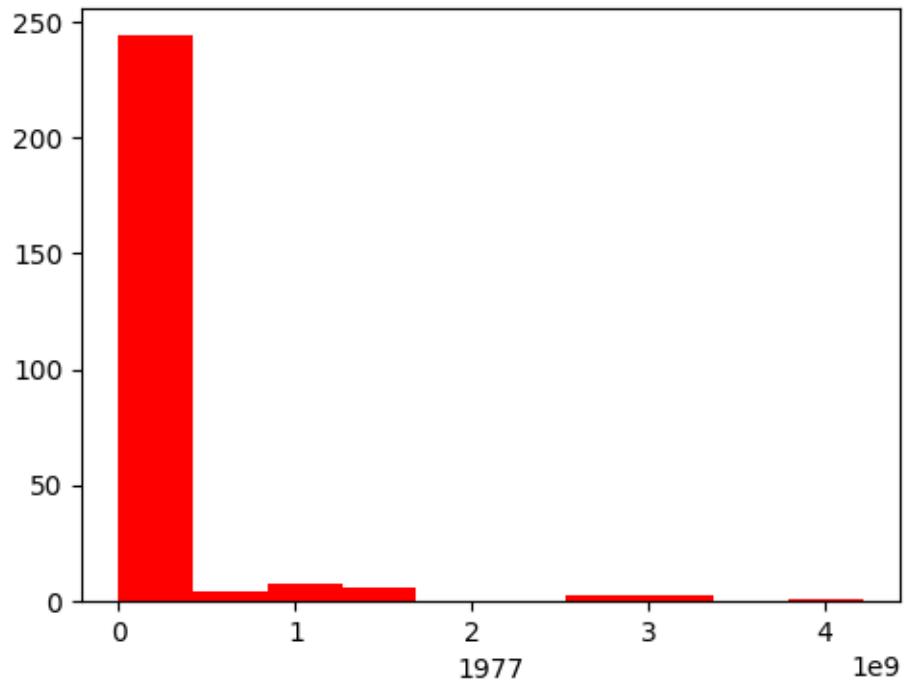


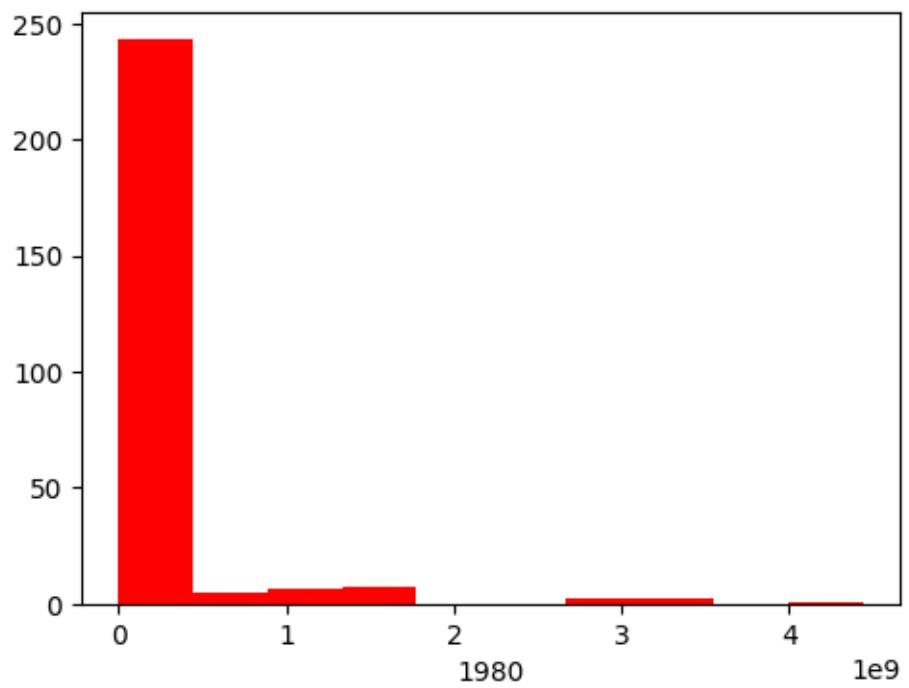
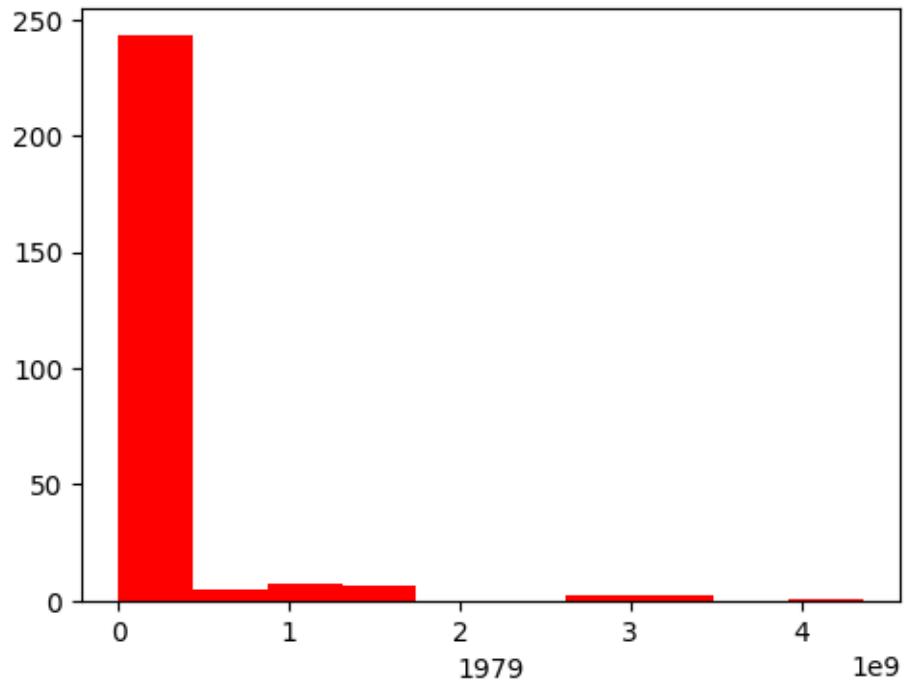


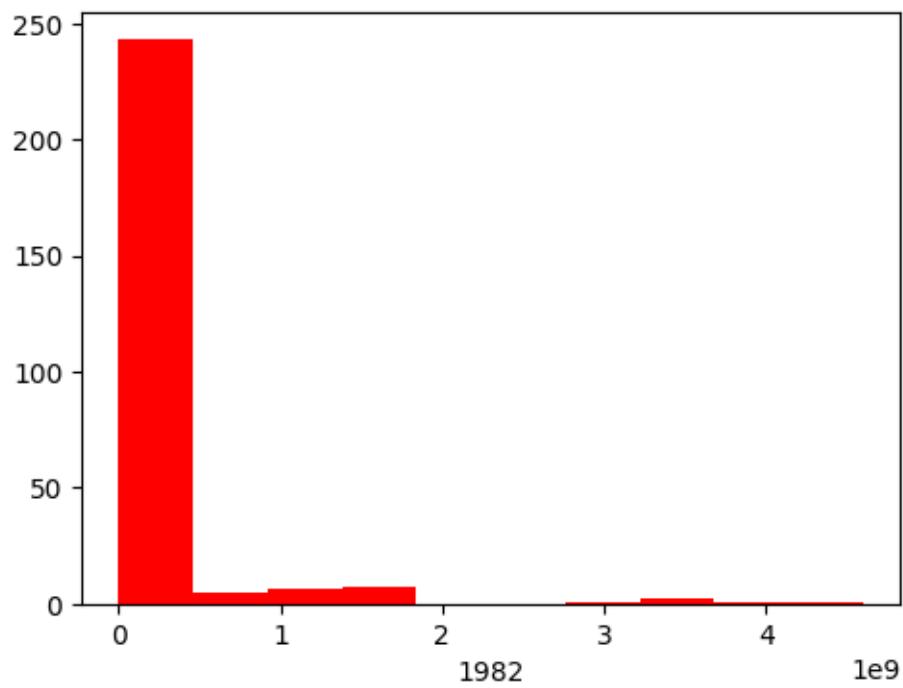
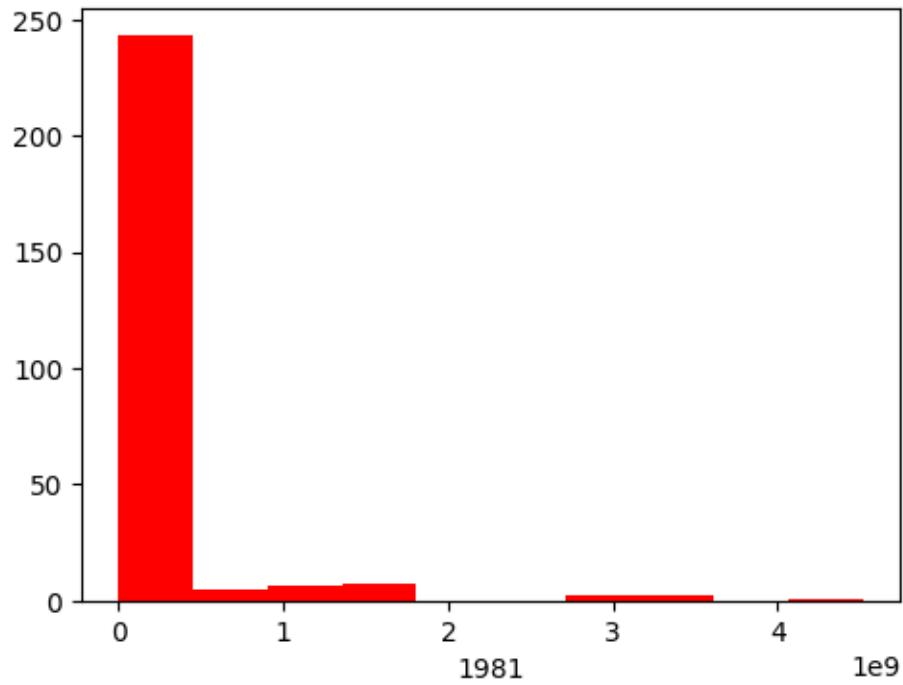


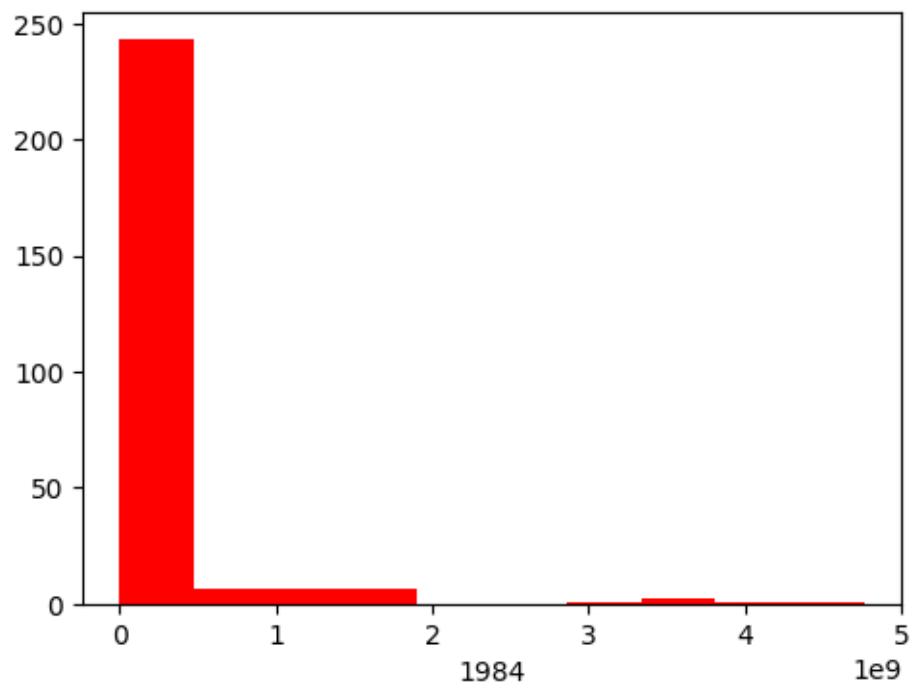
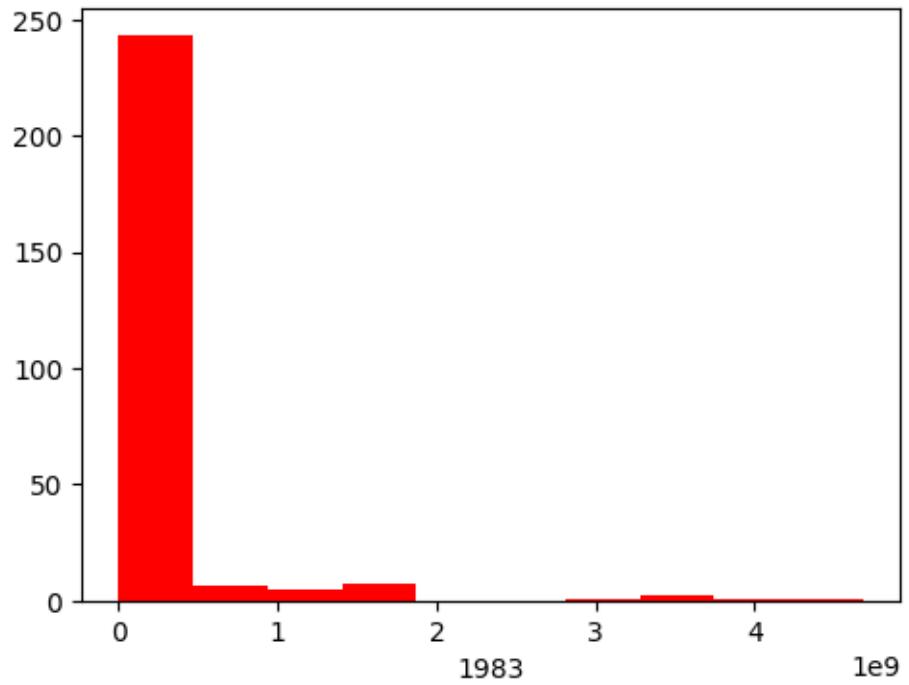


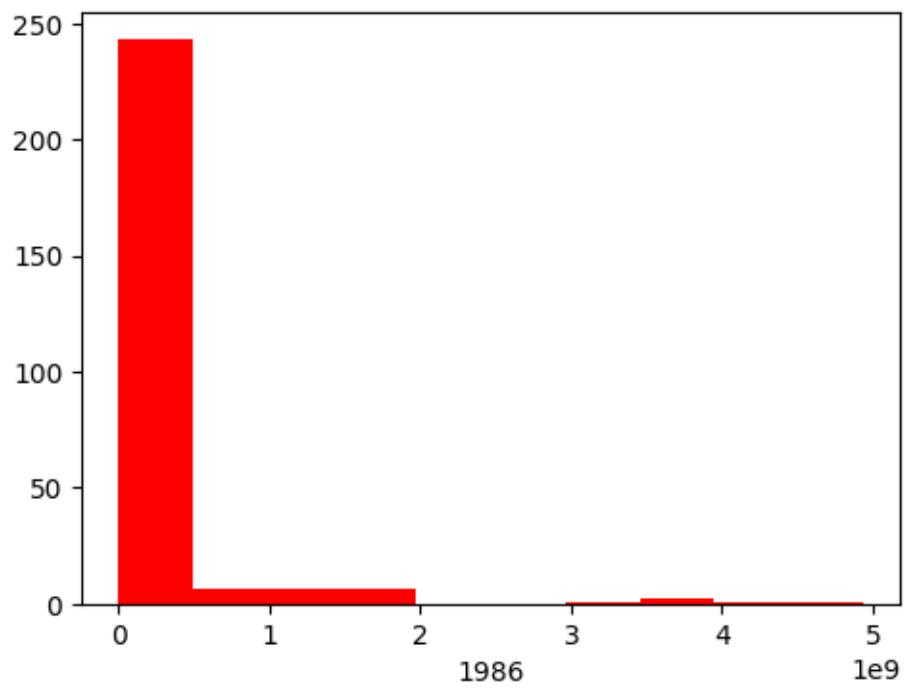
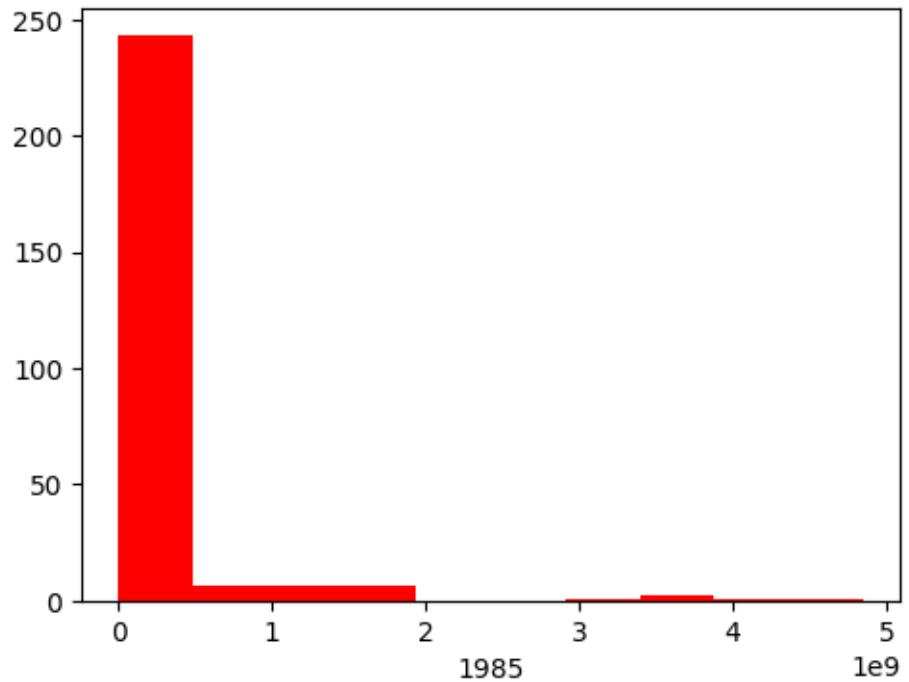


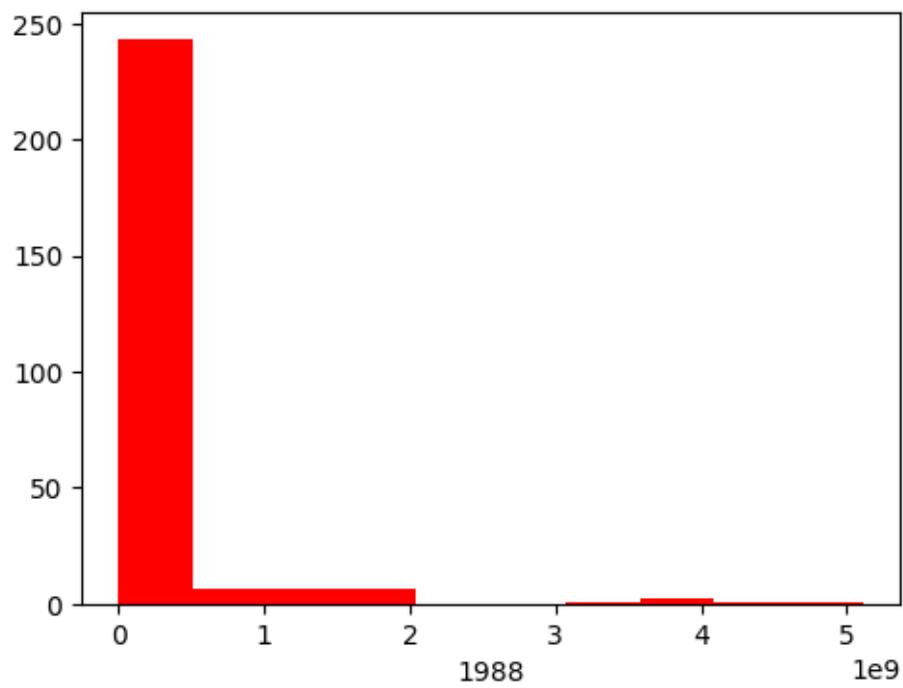
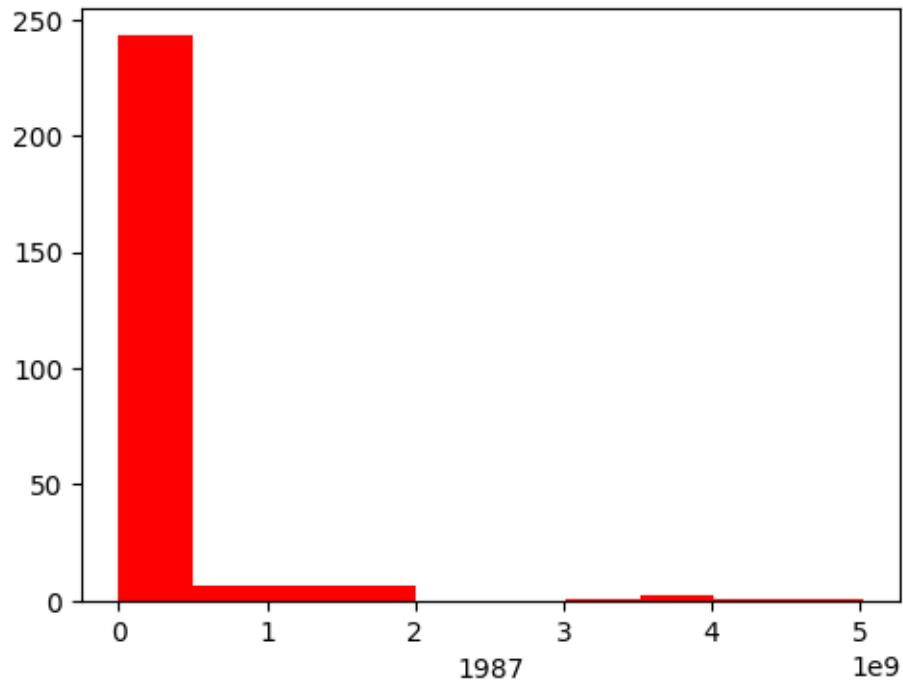


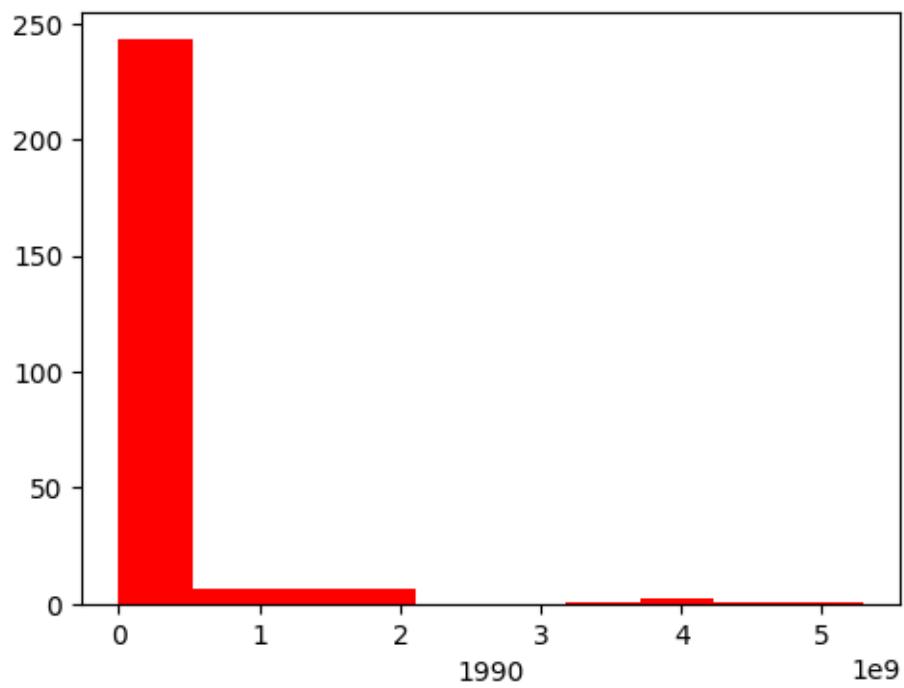
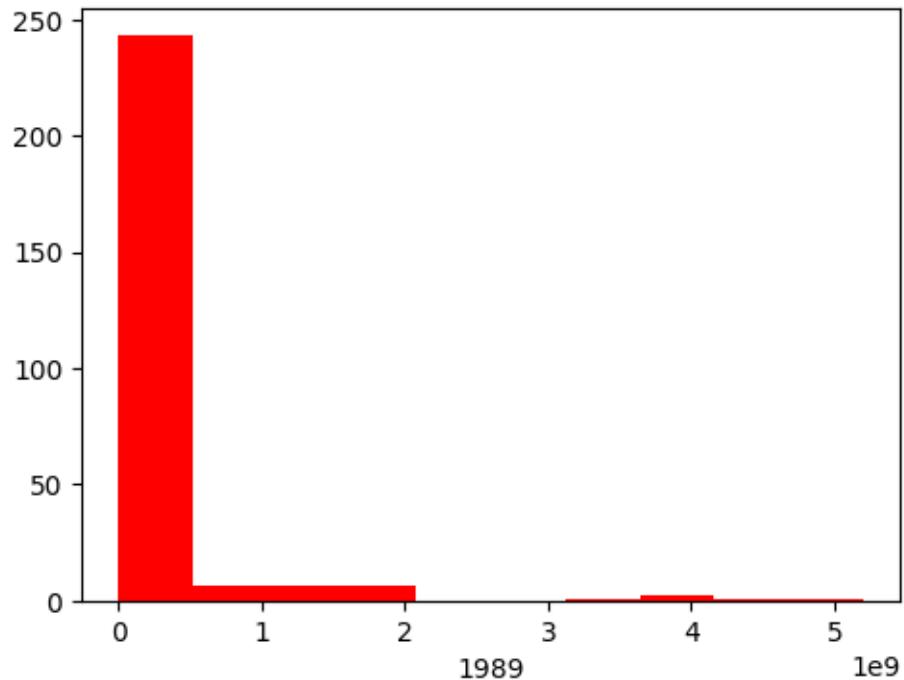


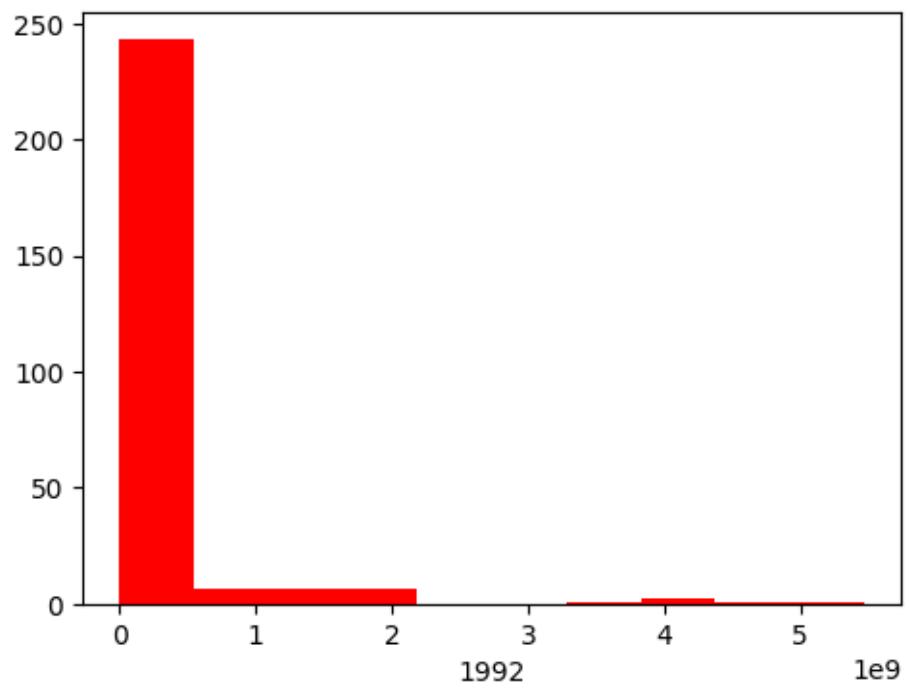
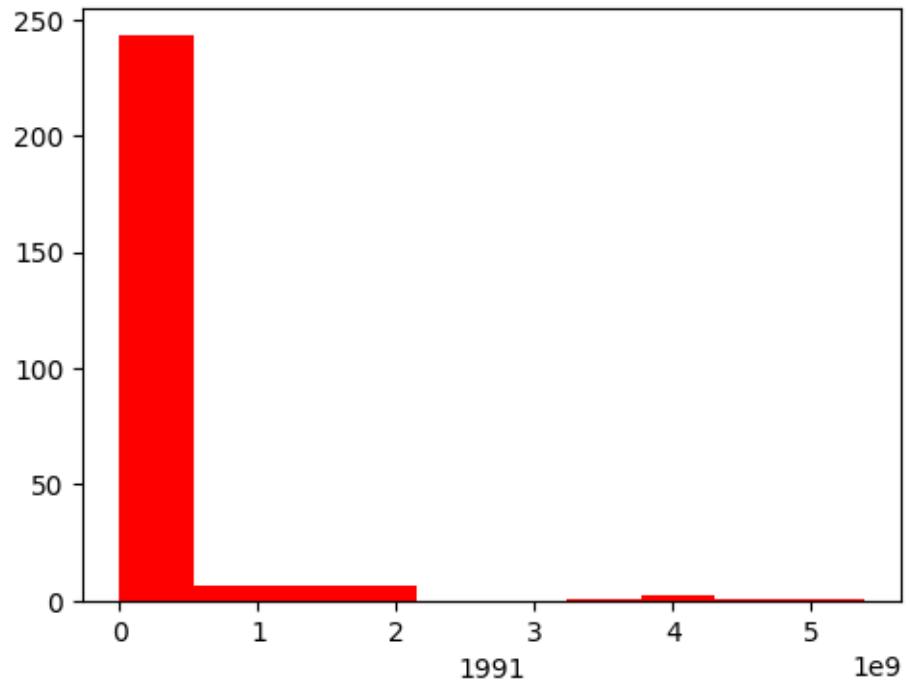


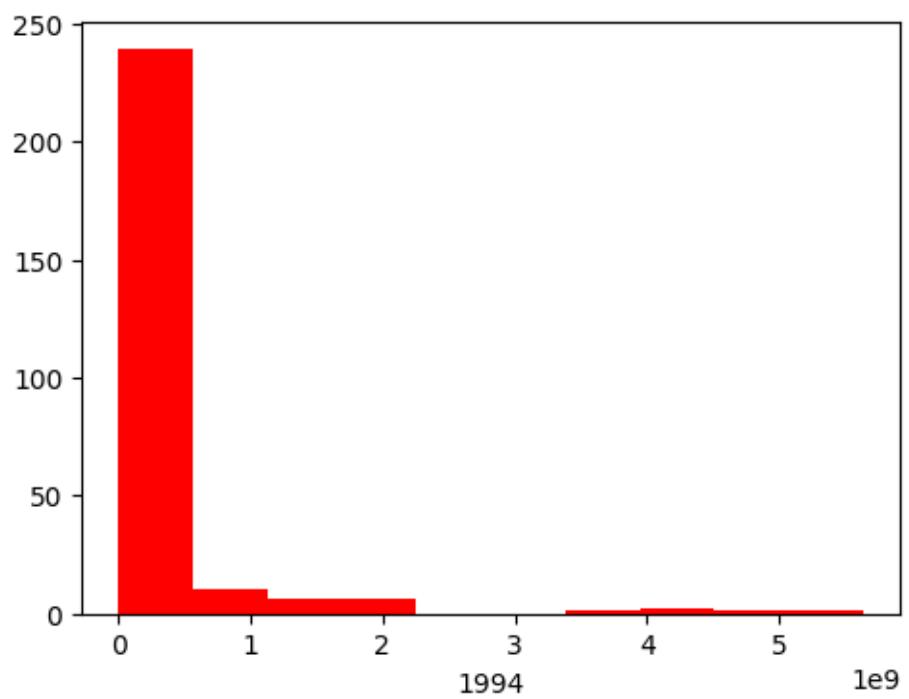
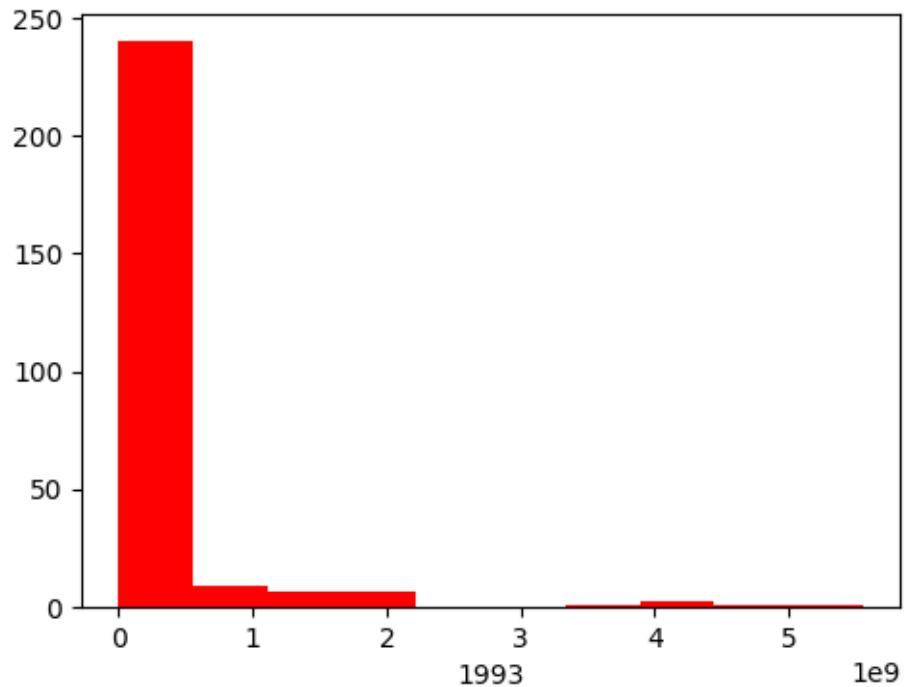


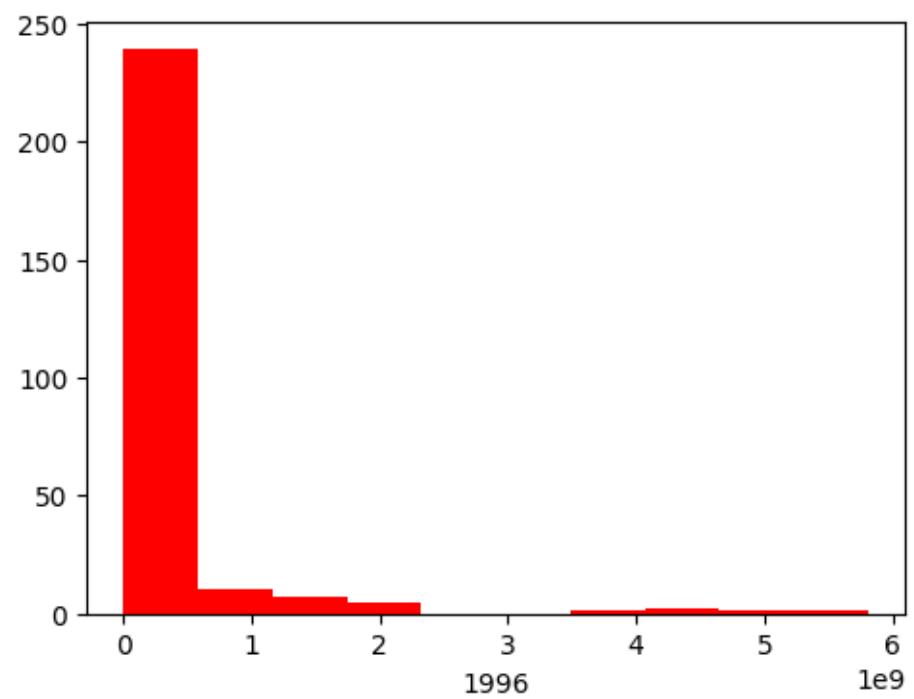
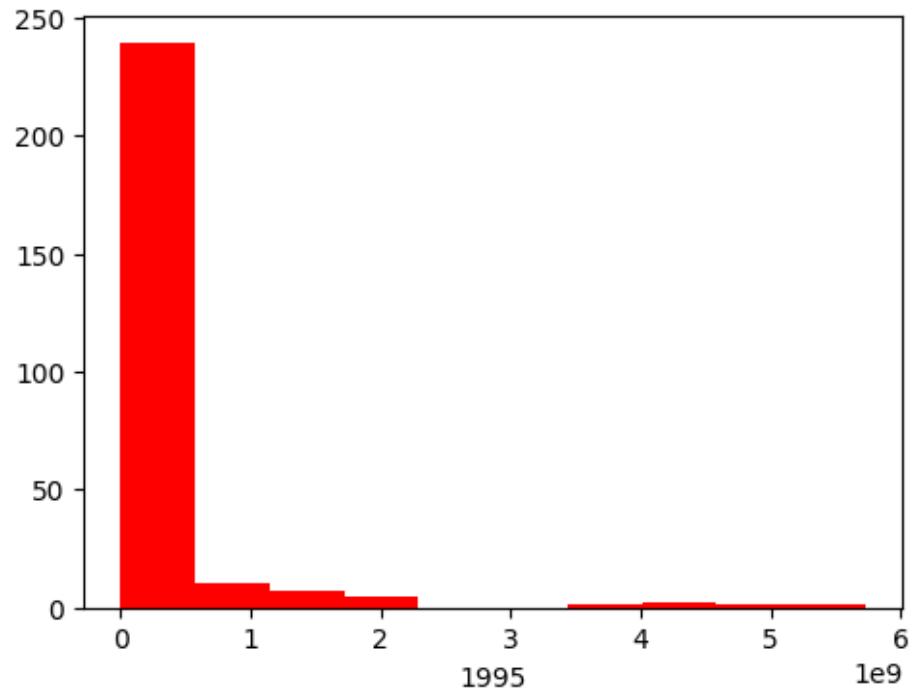


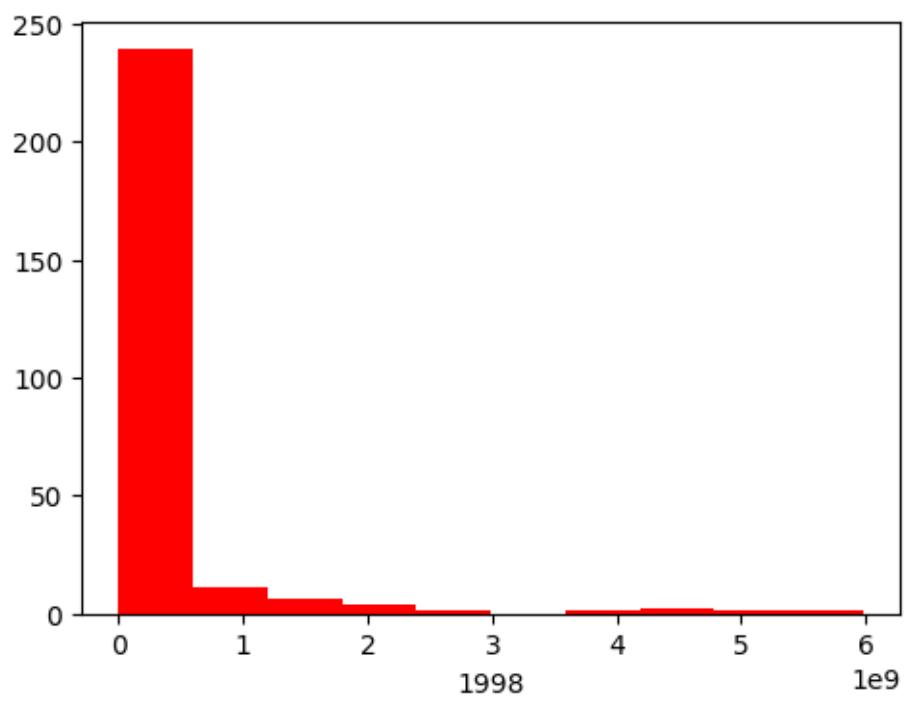
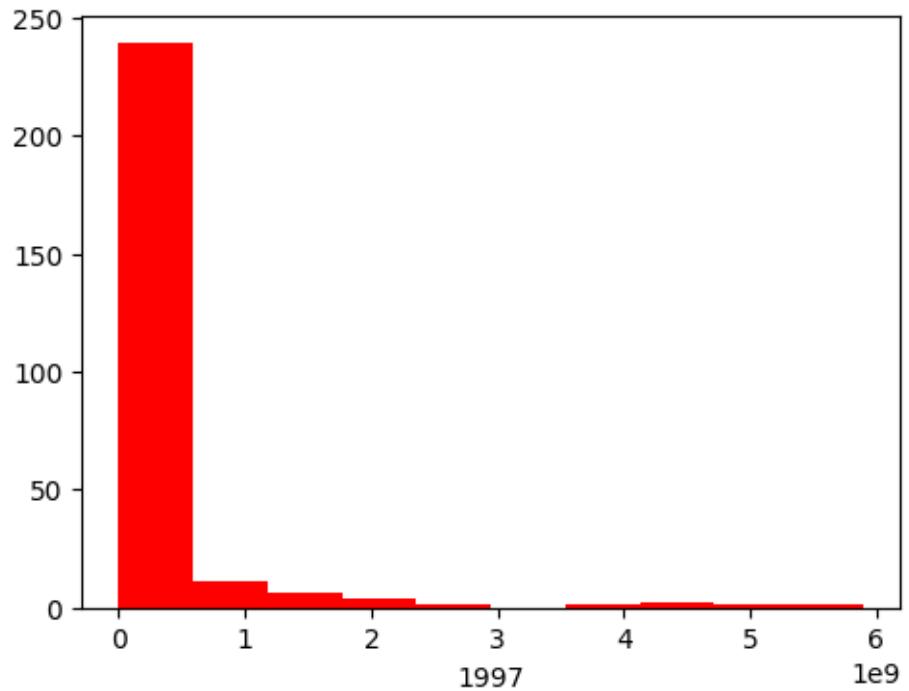


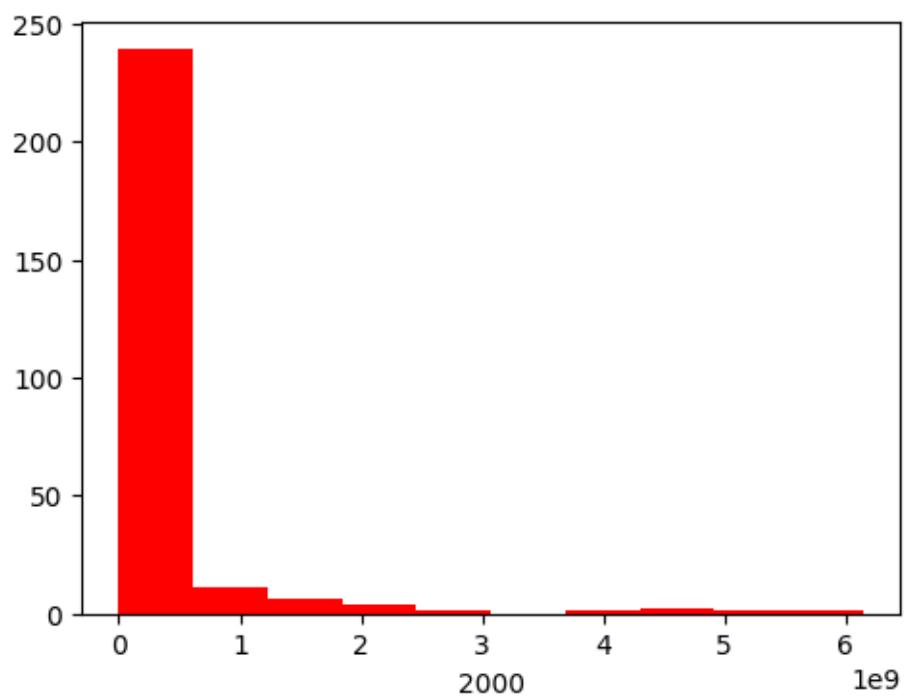
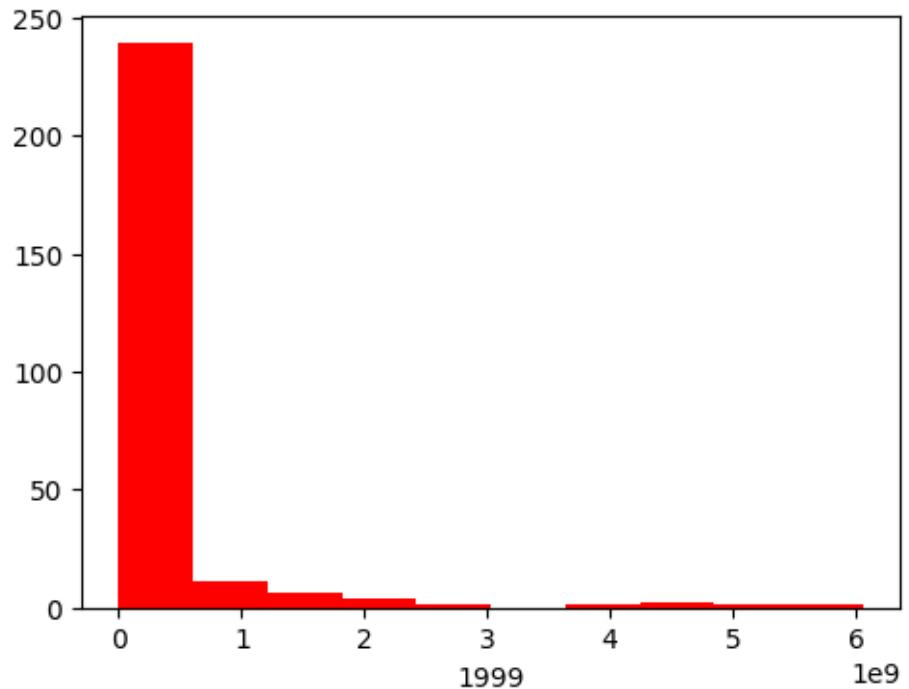


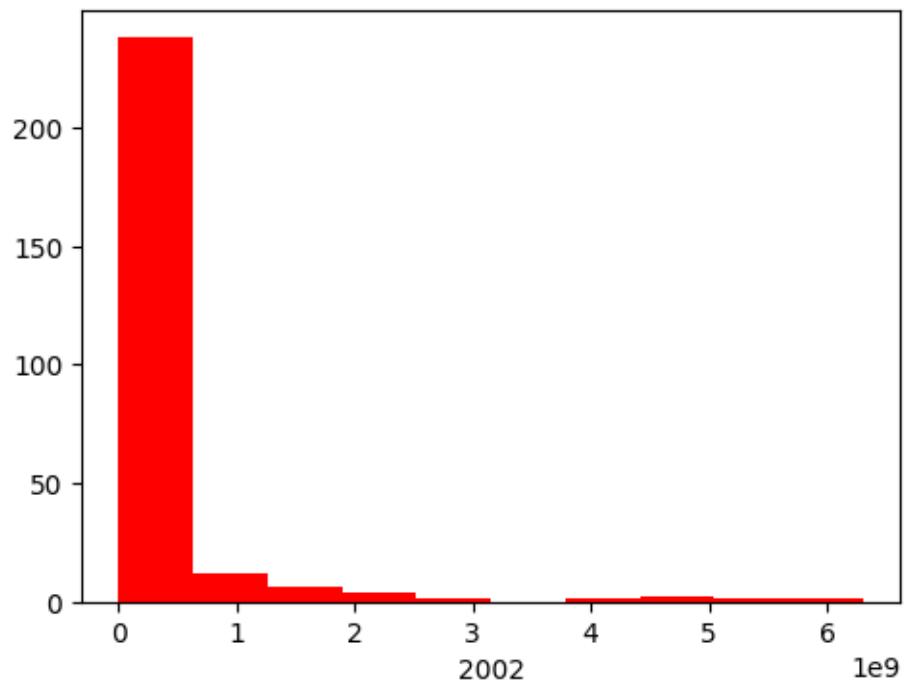
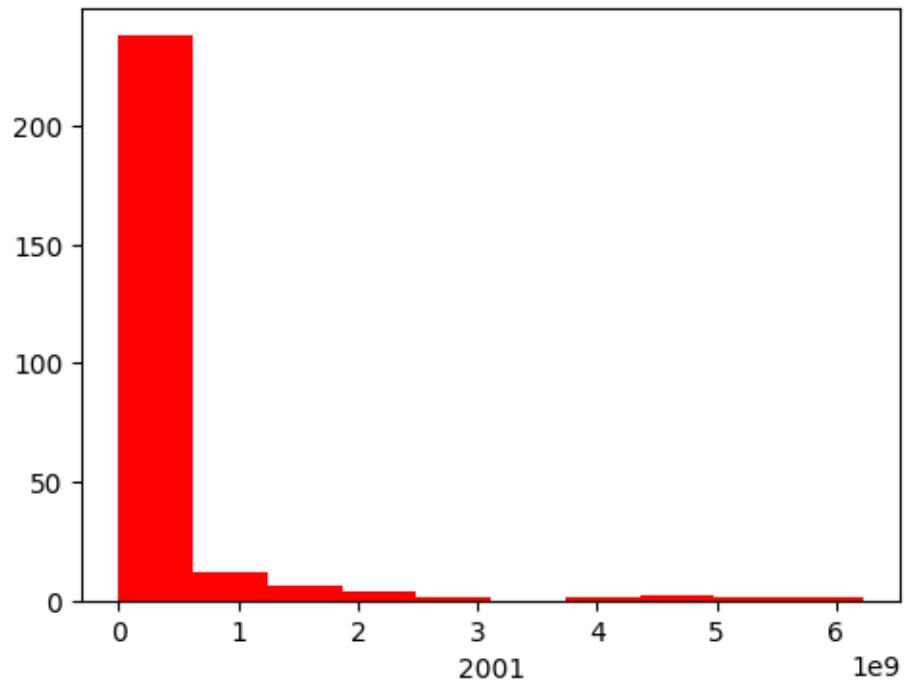


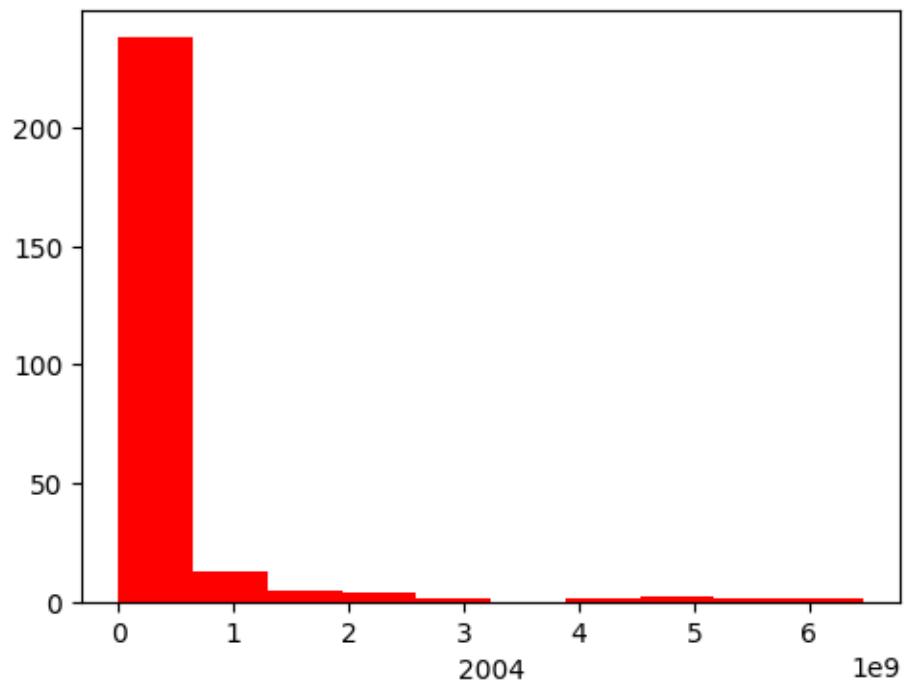
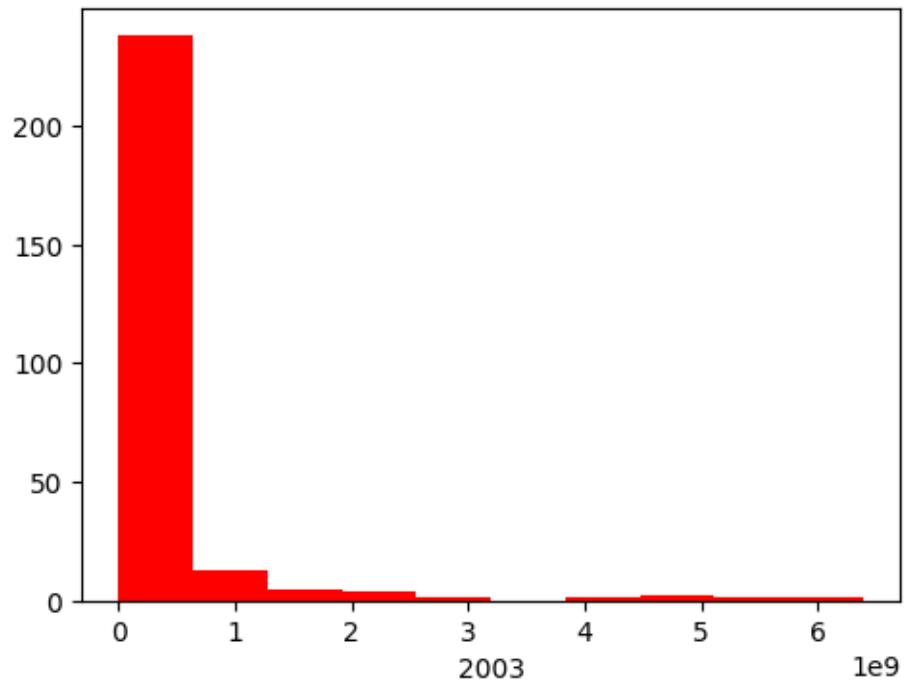


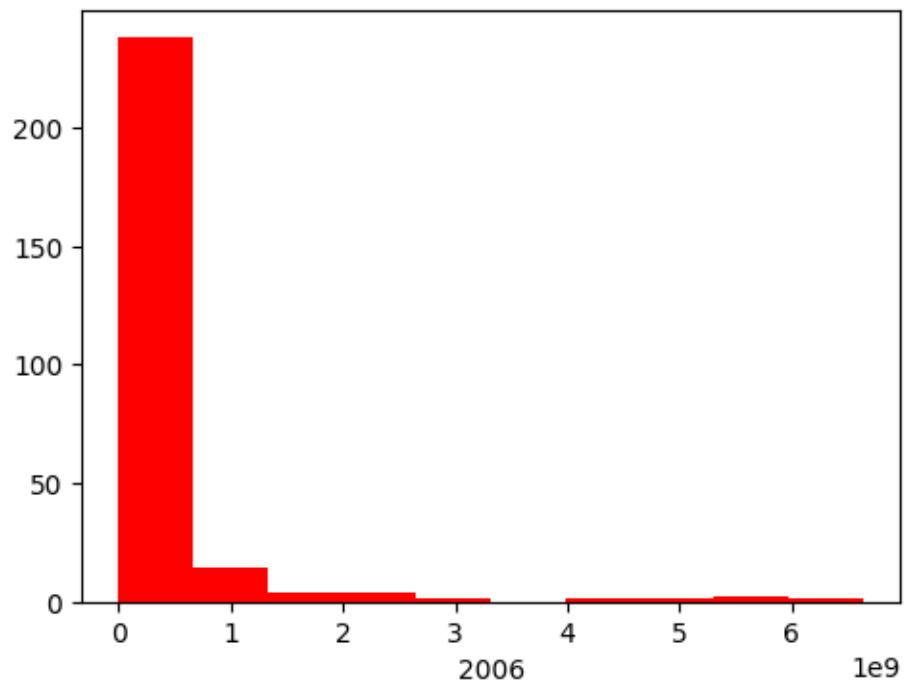
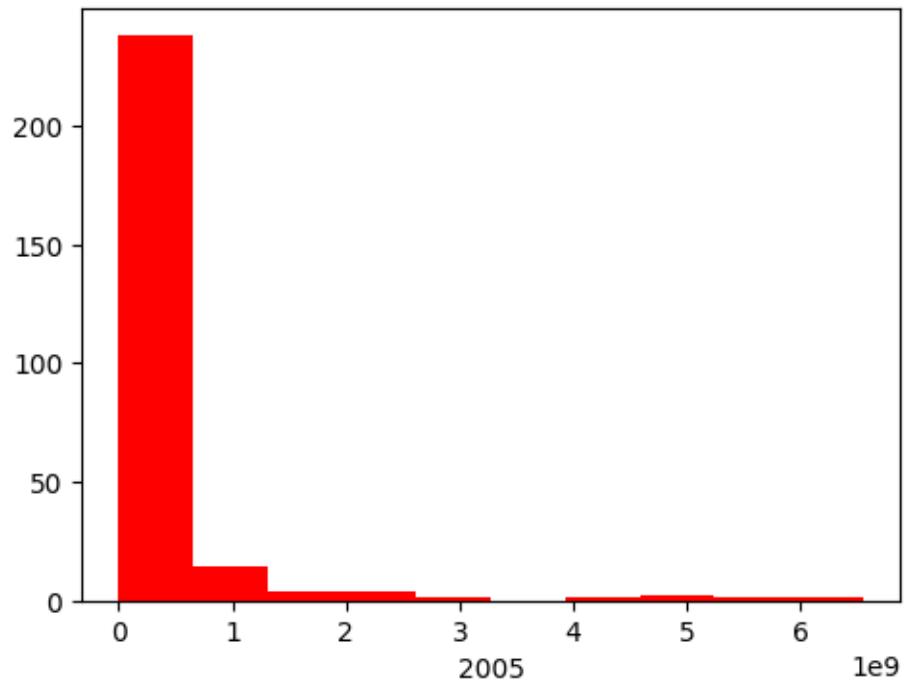


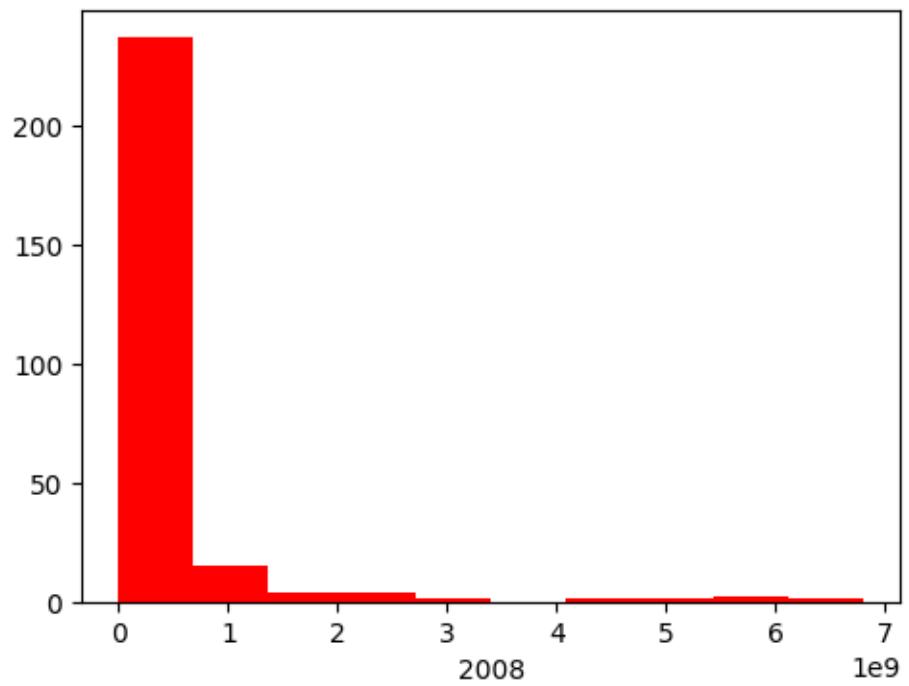
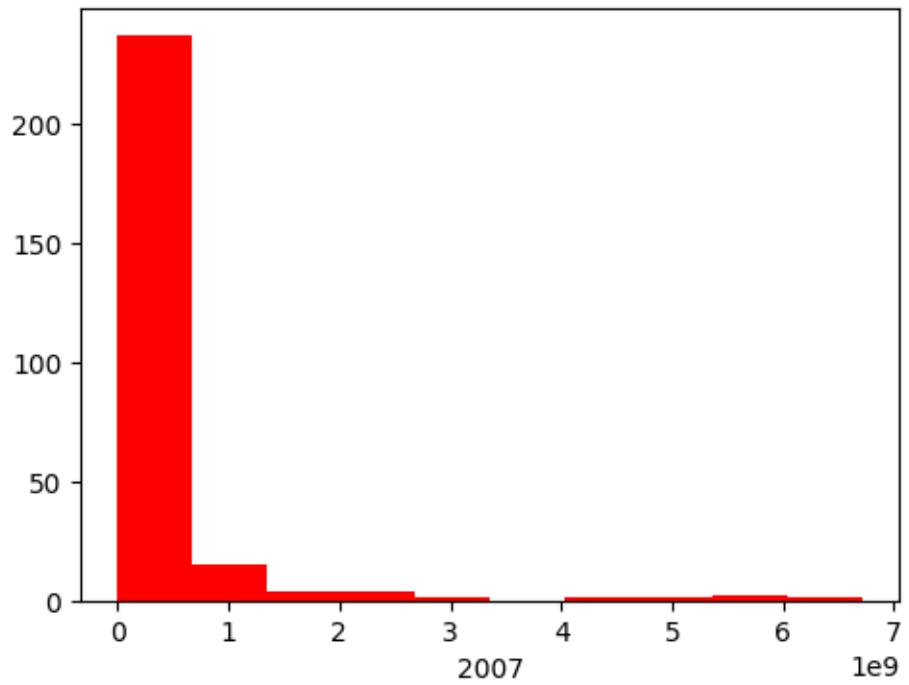


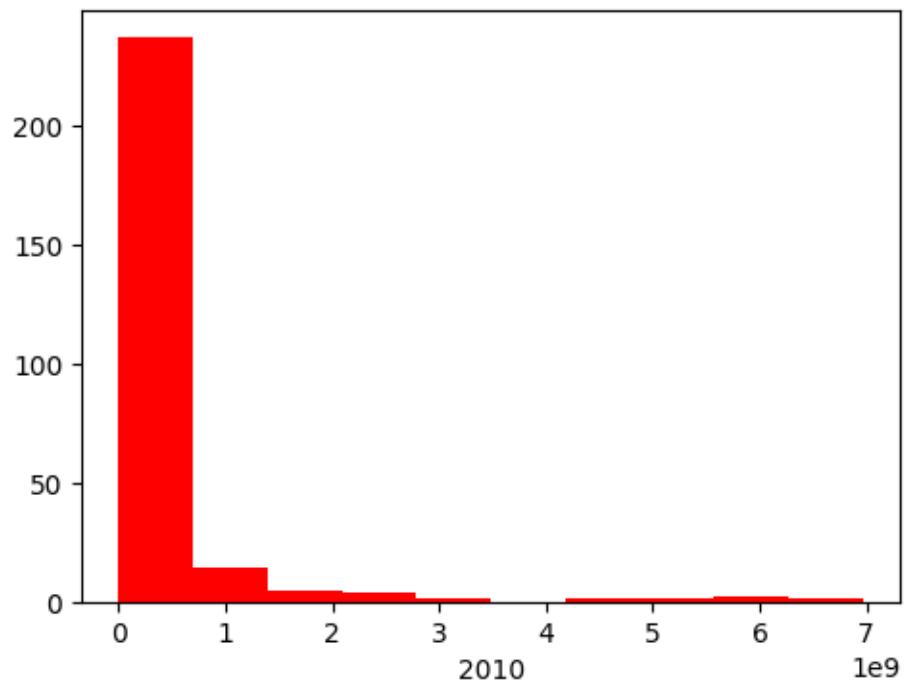
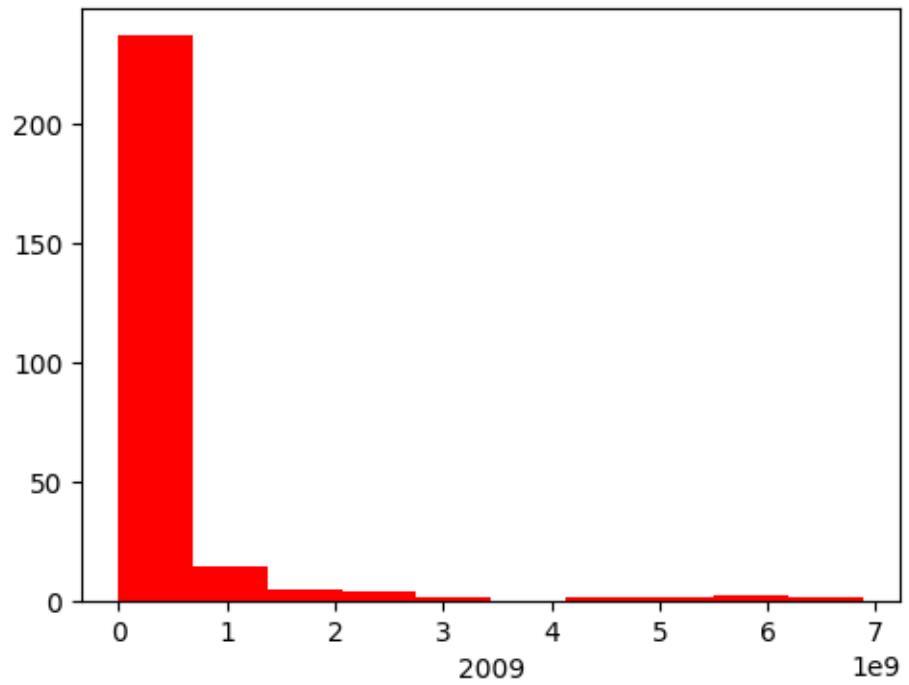


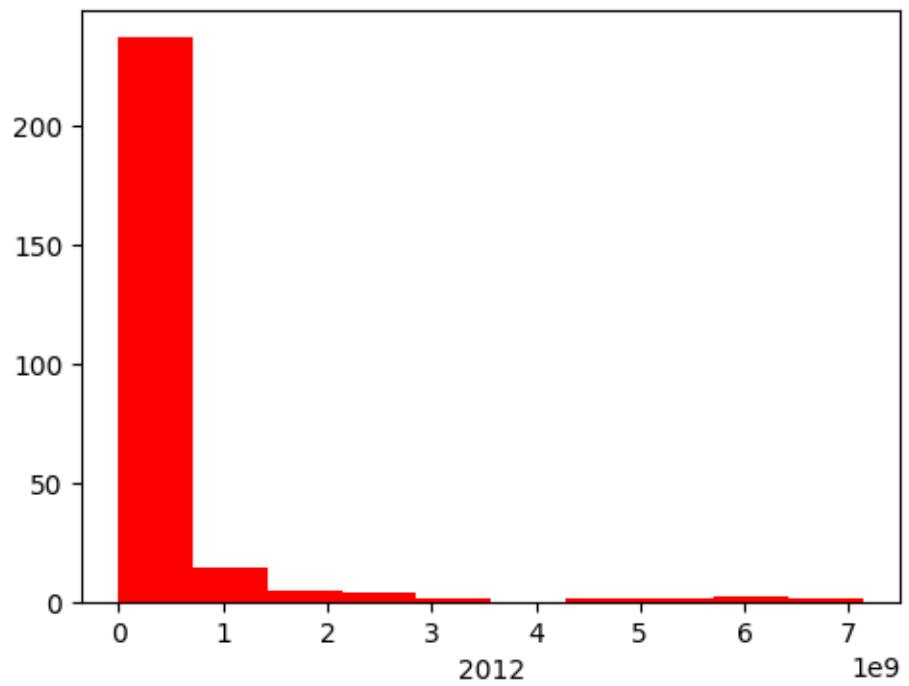
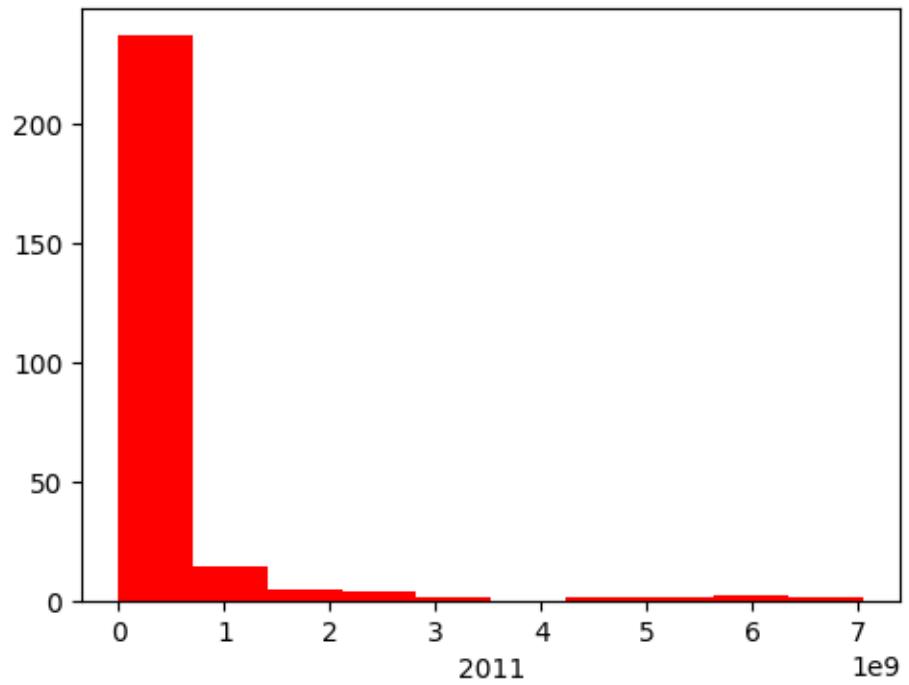


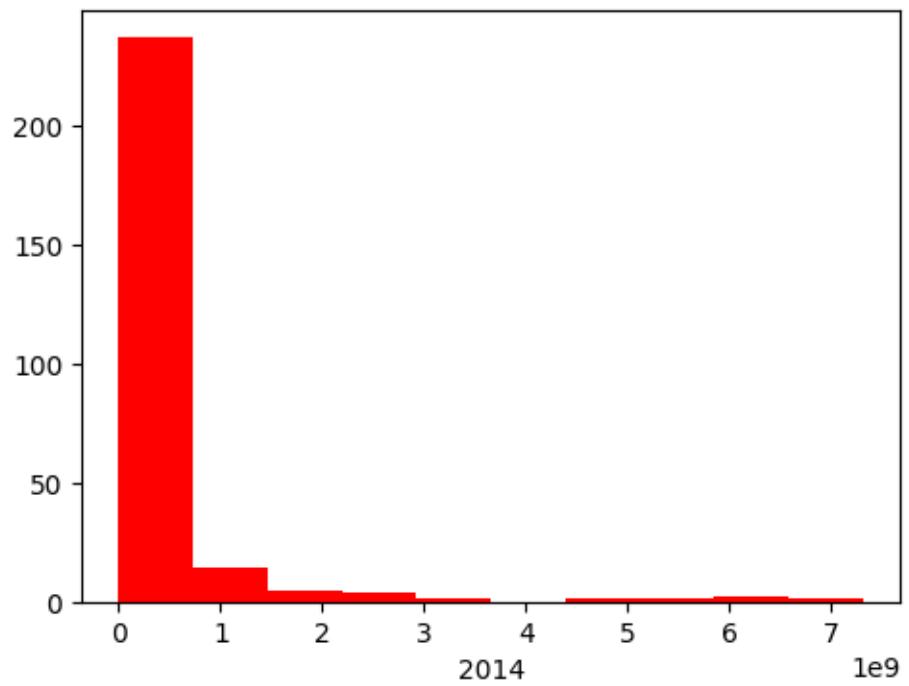
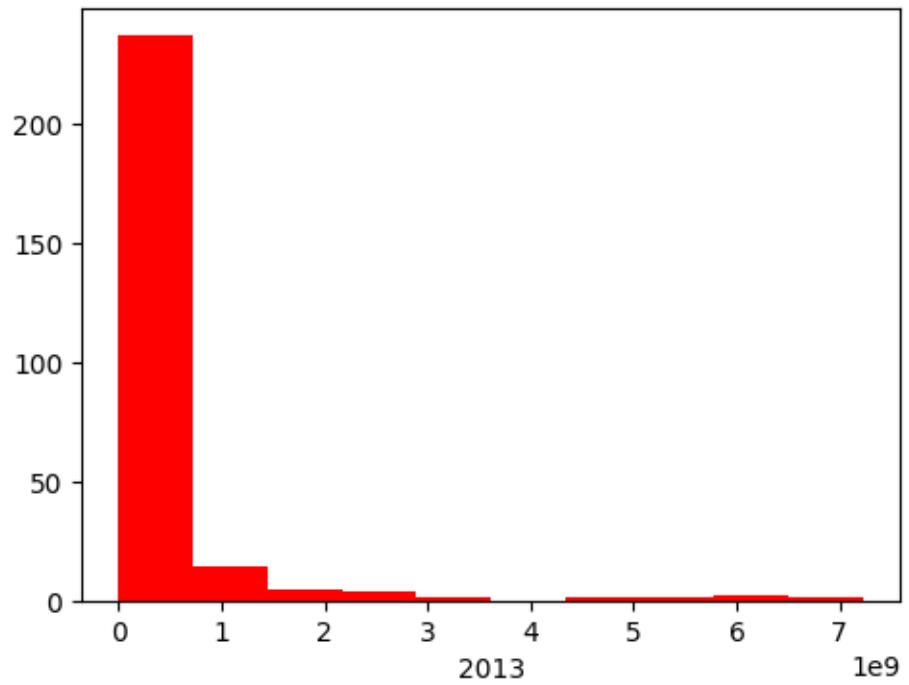


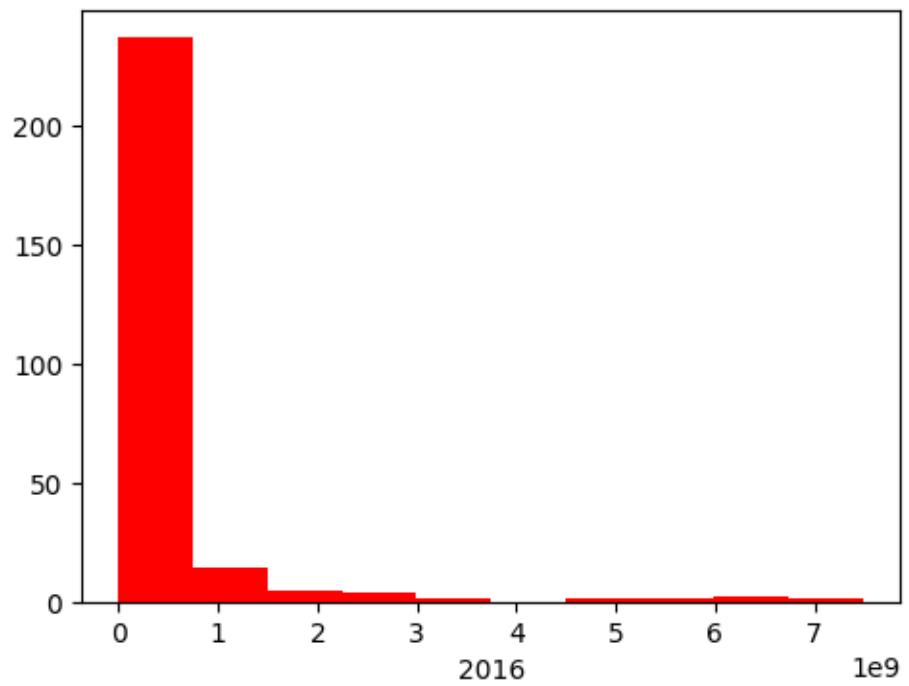
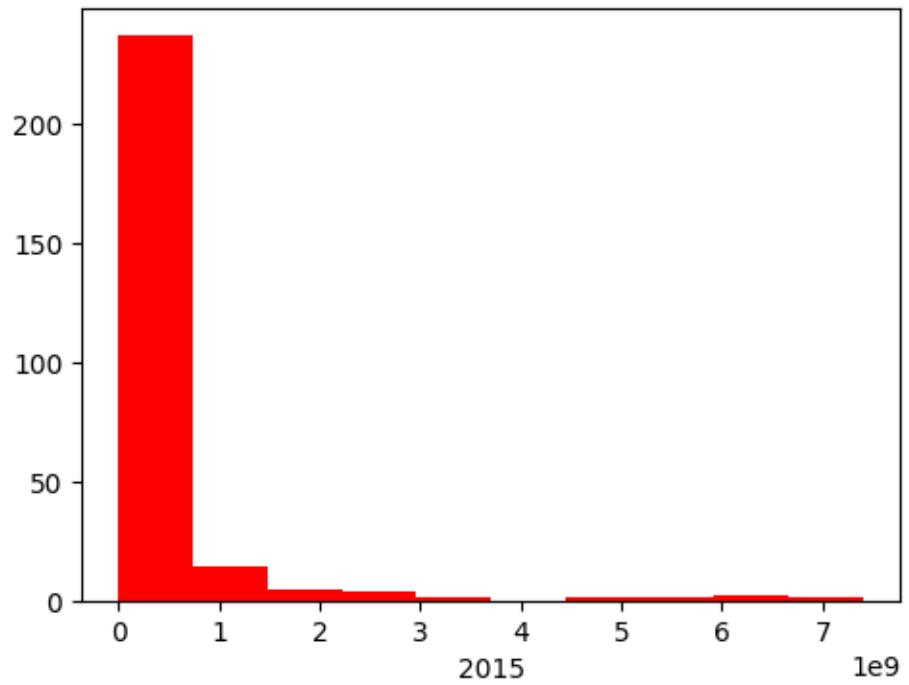


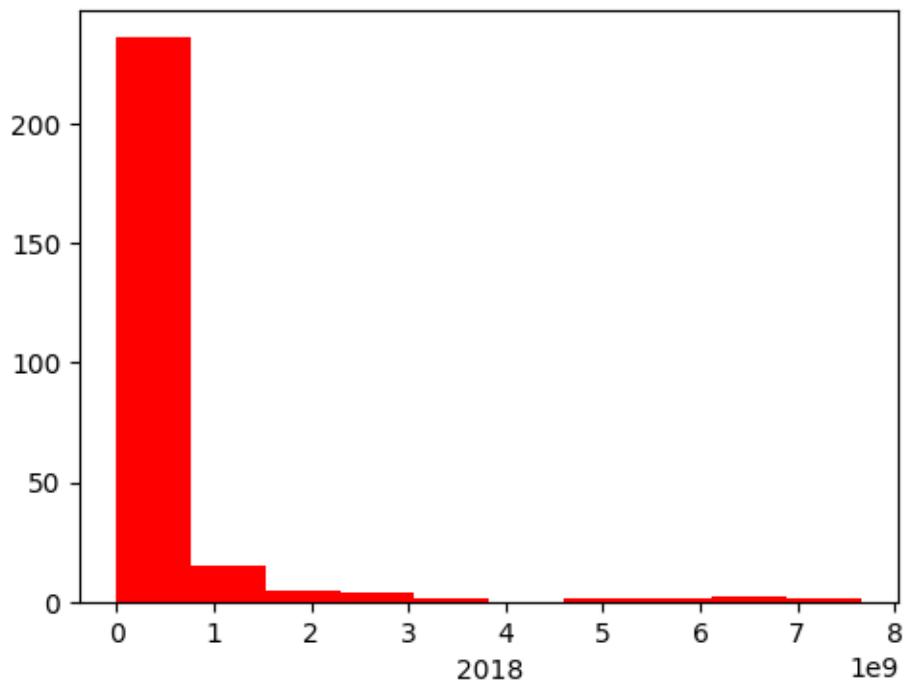
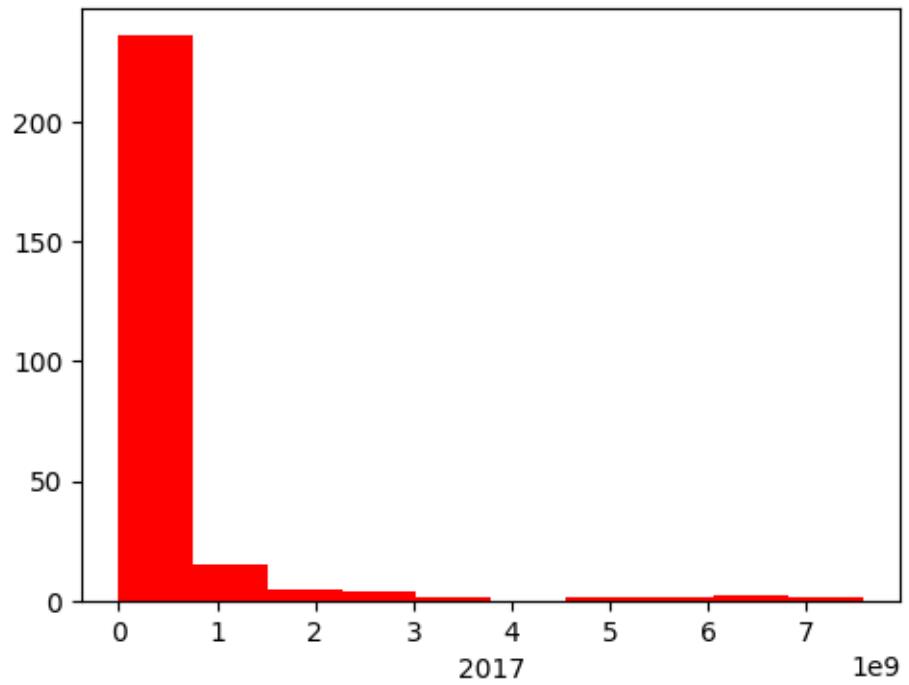


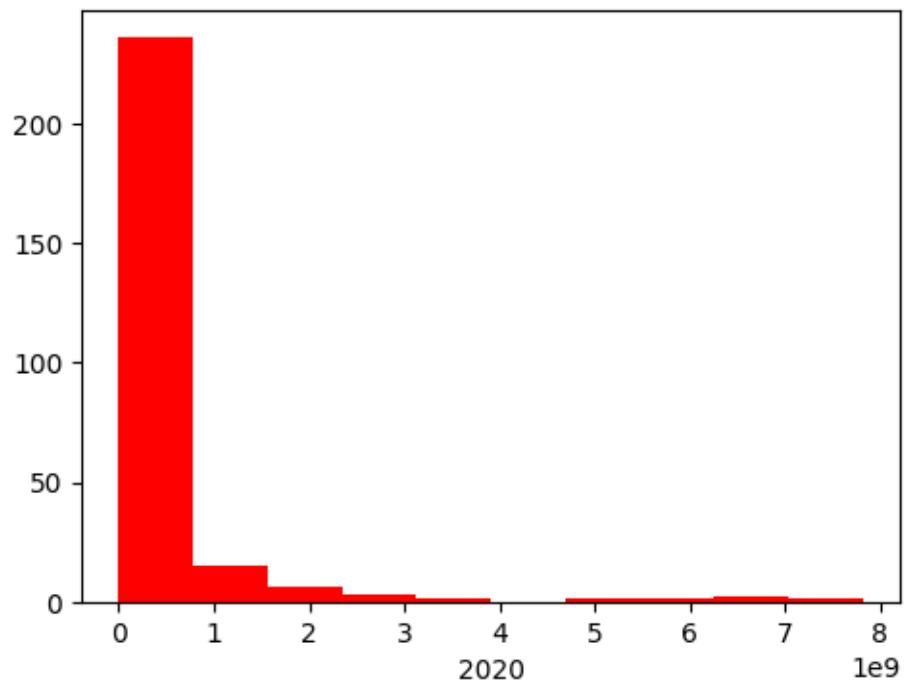
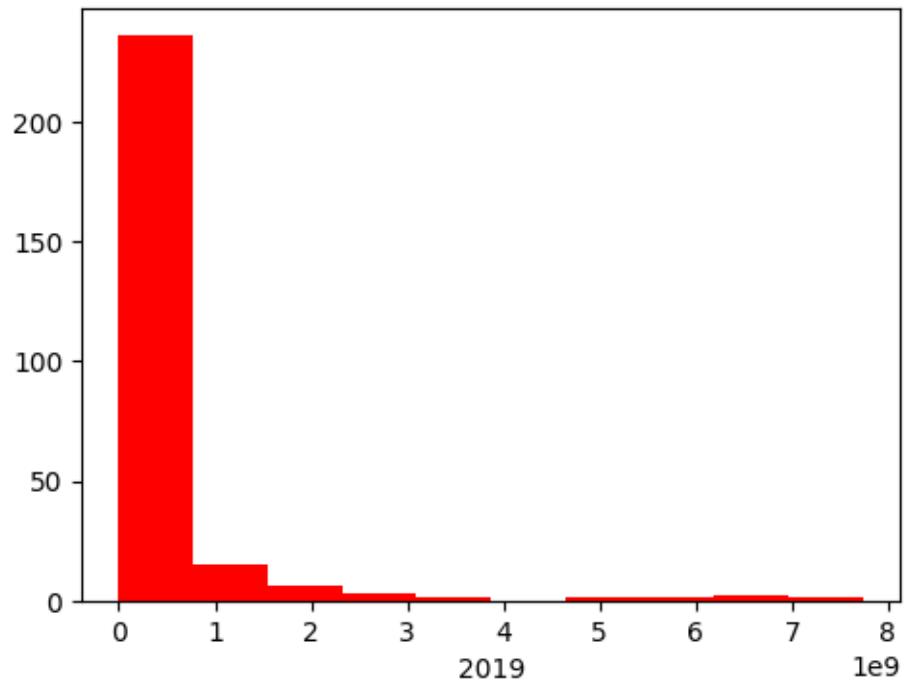


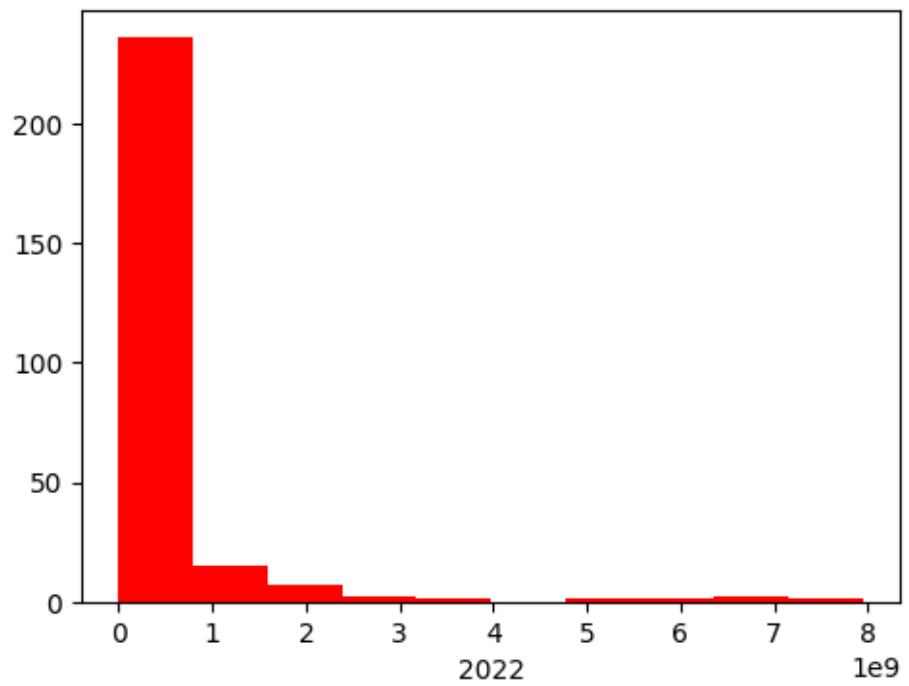
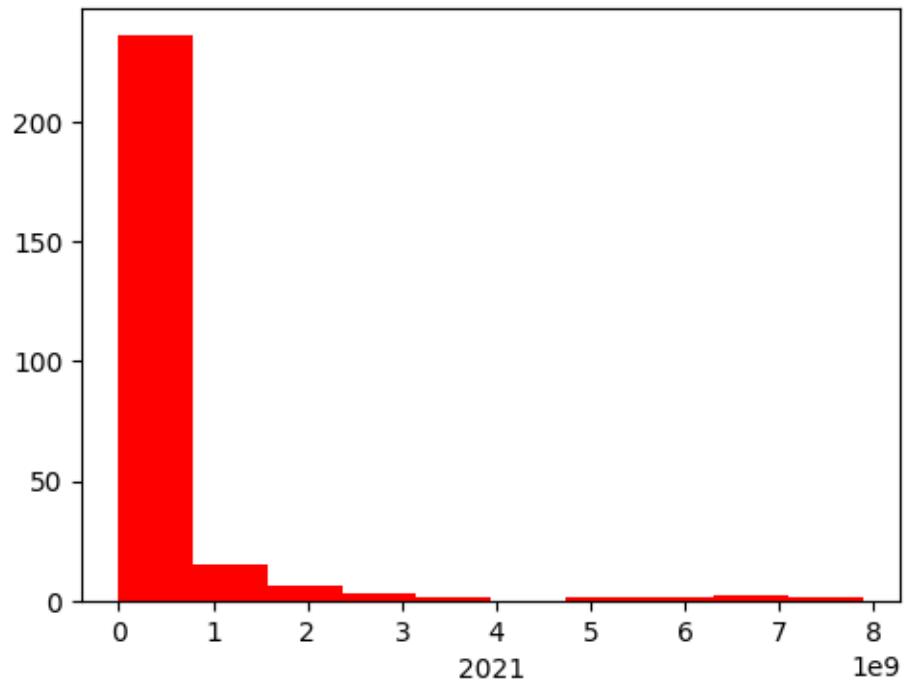


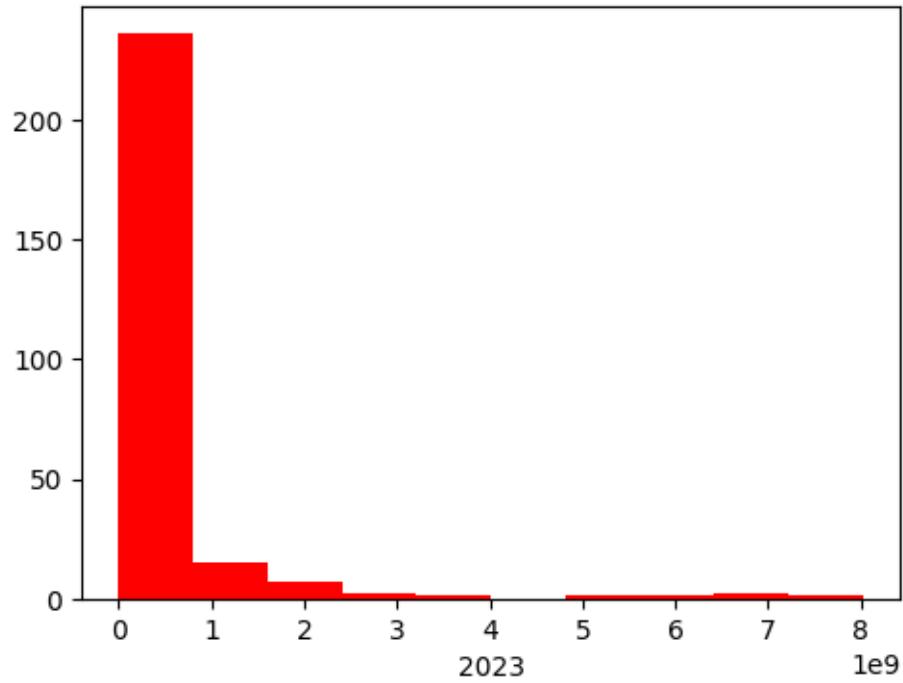








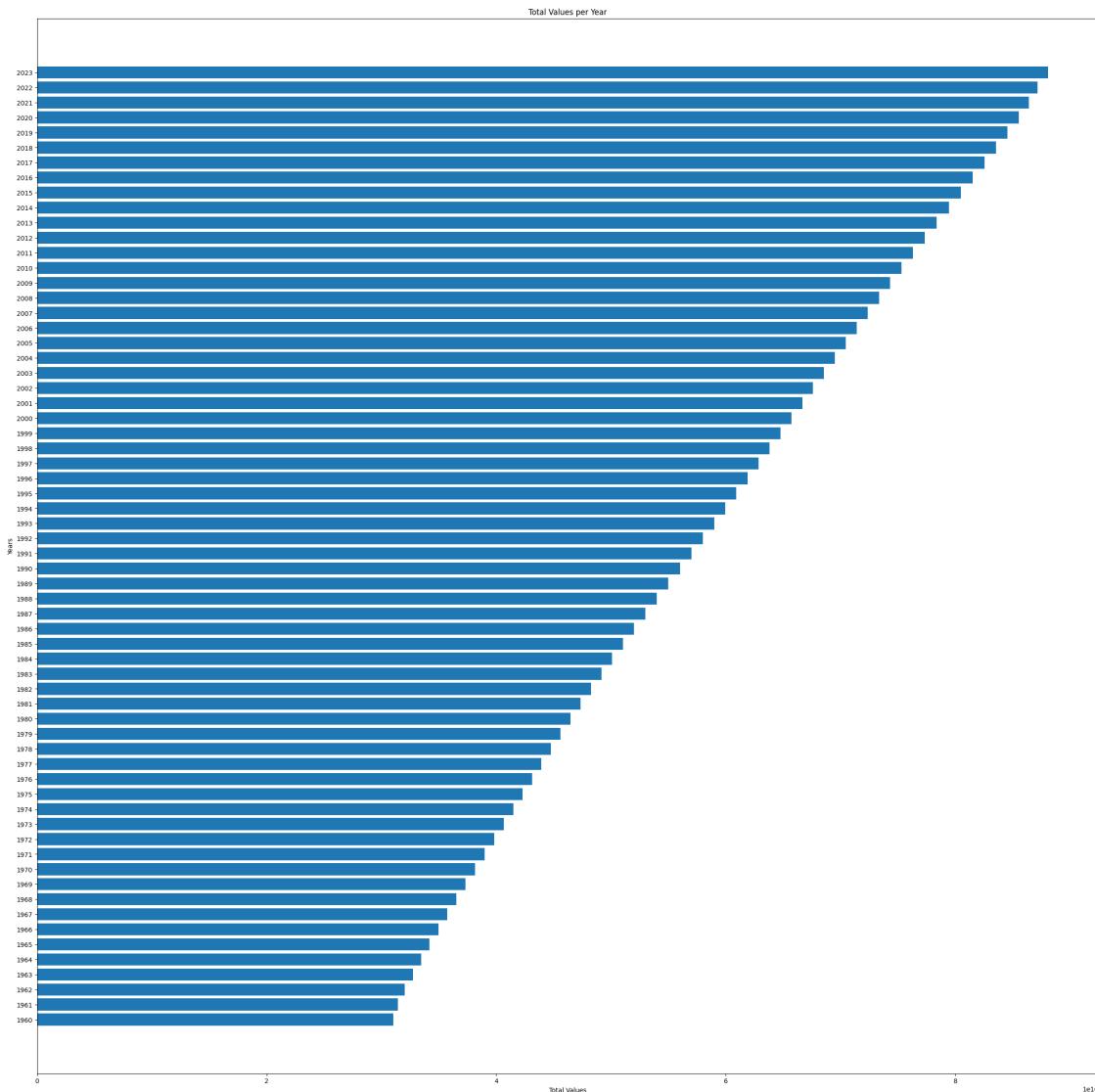




```
[ ]: years=df.columns[1:]

total_values=df[years].sum()

plt.figure(figsize=(30, 30))
plt.barh(years, total_values)
plt.xlabel('Total Values')
plt.ylabel('Years')
plt.title('Total Values per Year')
plt.show()
```



```
[ ]: country_by_1960=df.sort_values(by='1960').head(20)
country_by_1960
```

	Country Name	1960	1961	1962	1963	1964	\
225	Sint Maarten (Dutch part)	2646.0	2888.0	3171.0	3481.0	3811.0	
147	St. Martin (French part)	4135.0	4258.0	4388.0	4524.0	4666.0	
179	Nauru	4582.0	4753.0	4950.0	5198.0	5484.0	
245	Tuvalu	5404.0	5436.0	5471.0	5503.0	5525.0	
228	Turks and Caicos Islands	5604.0	5625.0	5633.0	5634.0	5642.0	
255	British Virgin Islands	7850.0	7885.0	7902.0	7919.0	7949.0	
52	Cayman Islands	8473.0	8626.0	8799.0	8985.0	9172.0	
164	Northern Mariana Islands	8702.0	8965.0	9252.0	9561.0	9890.0	
6	Andorra	9443.0	10216.0	11014.0	11839.0	12690.0	

188		Palau	9446.0	9639.0	9851.0	10076.0	10318.0
155		Marshall Islands	15374.0	15867.0	16387.0	16947.0	17537.0
212		San Marino	15556.0	15895.0	16242.0	16583.0	16926.0
137		Liechtenstein	16472.0	16834.0	17221.0	17625.0	18058.0
11		American Samoa	20085.0	20626.0	21272.0	21949.0	22656.0
149		Monaco	21797.0	21907.0	22106.0	22442.0	22766.0
84		Gibraltar	21822.0	21907.0	22249.0	22796.0	23347.0
91		Greenland	32500.0	33700.0	35000.0	36400.0	37600.0
256		Virgin Islands (U.S.)	32500.0	34300.0	35000.0	39800.0	40800.0
78		Faroe Islands	34154.0	34572.0	34963.0	35385.0	35841.0
200		Qatar	36385.0	40111.0	45123.0	50950.0	57531.0

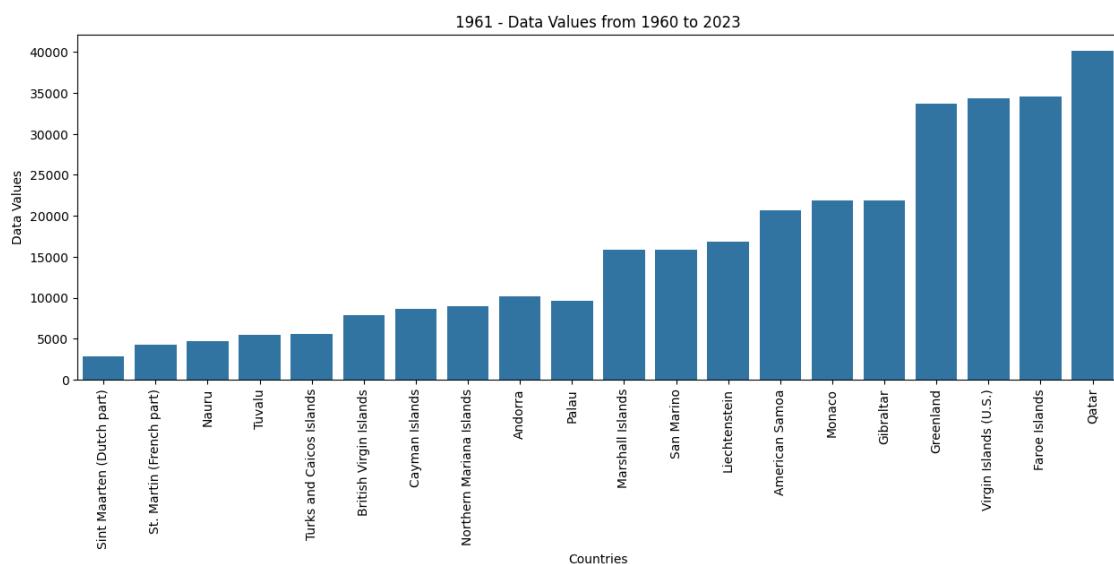
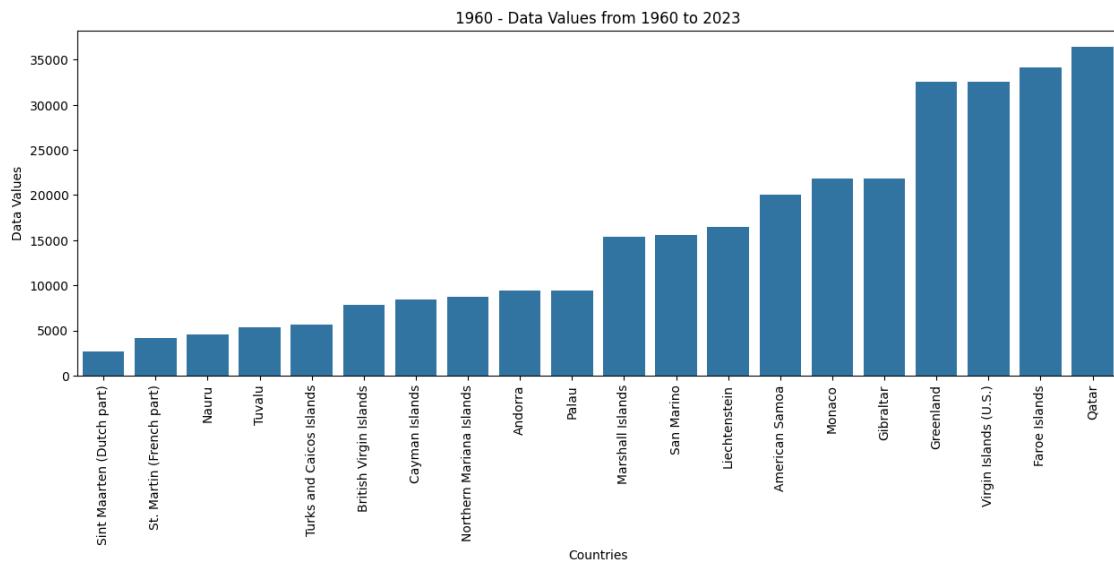
	1965	1966	1967	1968	...	2014	2015	2016	\
225	4161.0	4531.0	4930.0	5354.0	...	37685.0	38825.0	38992.0	
147	4832.0	5044.0	5294.0	5497.0	...	35261.0	35020.0	34811.0	
179	5804.0	6021.0	6114.0	6288.0	...	10940.0	11185.0	11437.0	
245	5548.0	5591.0	5657.0	5729.0	...	10899.0	10877.0	10852.0	
228	5650.0	5652.0	5662.0	5668.0	...	34985.0	36538.0	38246.0	
255	8018.0	8139.0	8337.0	8649.0	...	28971.0	29366.0	29739.0	
52	9366.0	9566.0	9771.0	9981.0	...	59559.0	60911.0	62255.0	
164	10229.0	10577.0	10720.0	10440.0	...	51856.0	51514.0	51133.0	
6	13563.0	14546.0	15745.0	17079.0	...	71621.0	71746.0	72540.0	
188	10563.0	10813.0	10992.0	11079.0	...	17796.0	17794.0	17816.0	
155	18154.0	18794.0	19665.0	21001.0	...	50419.0	49410.0	48329.0	
212	17273.0	17588.0	17907.0	18291.0	...	33389.0	33570.0	33834.0	
137	18500.0	18957.0	19467.0	20011.0	...	37096.0	37355.0	37609.0	
11	23391.0	24122.0	24848.0	25608.0	...	52217.0	51368.0	50448.0	
149	23022.0	23198.0	23281.0	23481.0	...	36110.0	36760.0	37071.0	
84	23910.0	24477.0	25047.0	25610.0	...	32452.0	32520.0	32565.0	
91	39200.0	40500.0	41900.0	43400.0	...	56295.0	56114.0	56186.0	
256	43500.0	46200.0	49100.0	55700.0	...	107882.0	107712.0	107516.0	
78	36346.0	36825.0	37234.0	37630.0	...	48465.0	48816.0	49500.0	
200	64843.0	73102.0	82517.0	93022.0	...	2214465.0	2414573.0	2595166.0	

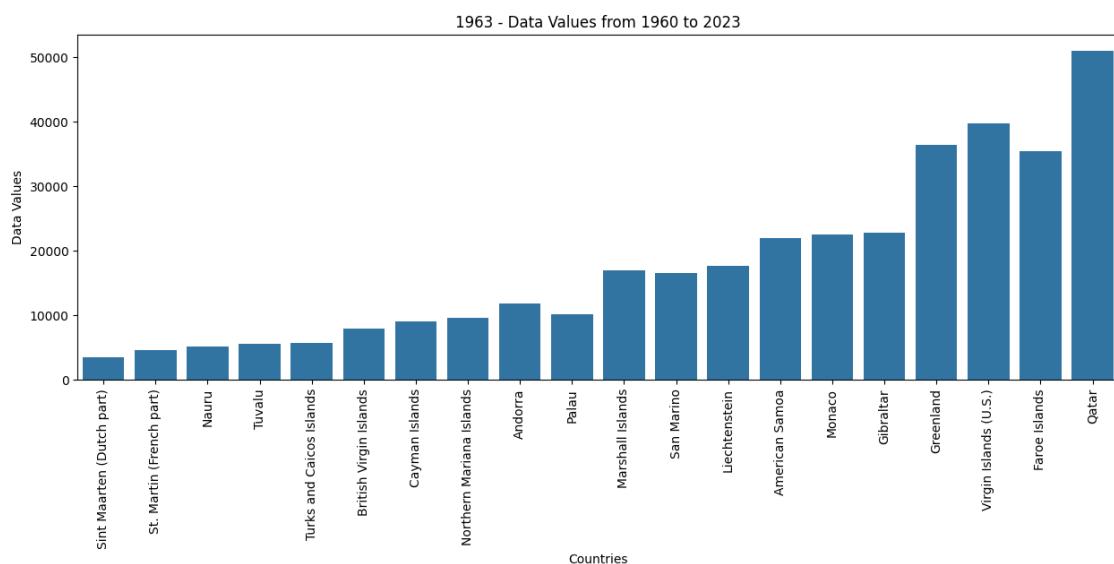
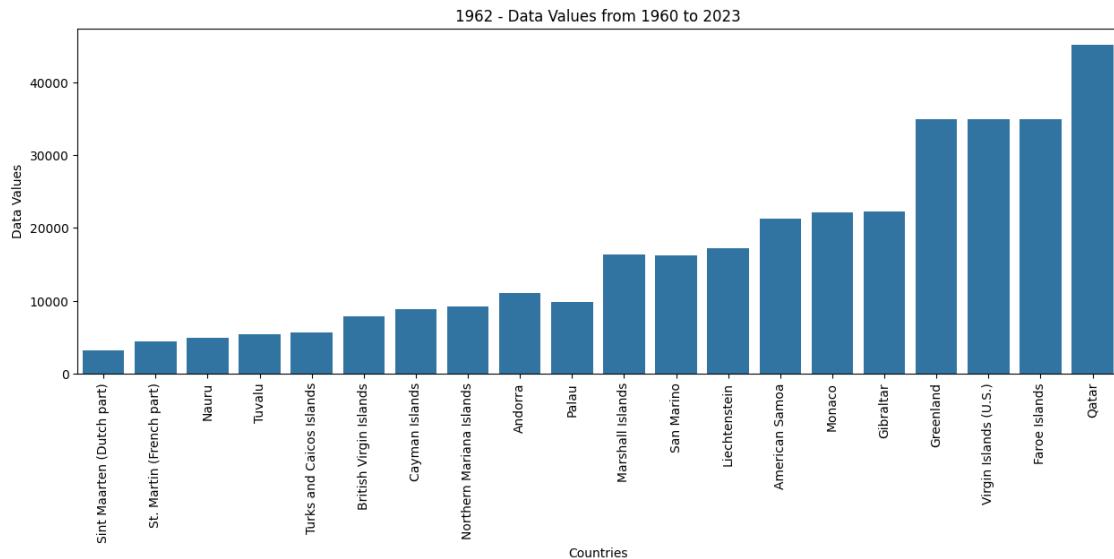
	2017	2018	2019	2020	2021	2022	\
225	38615.0	38934.0	39648.0	40350.0	40708.0	40888.0	
147	34496.0	33852.0	33121.0	32553.0	31948.0	31791.0	
179	11682.0	11924.0	12132.0	12315.0	12511.0	12668.0	
245	10828.0	10865.0	10956.0	11069.0	11204.0	11312.0	
228	39844.0	41487.0	43080.0	44276.0	45114.0	45703.0	
255	30060.0	30335.0	30610.0	30910.0	31122.0	31305.0	
52	63581.0	64884.0	66134.0	67311.0	68136.0	68706.0	
164	50729.0	50304.0	49858.0	49587.0	49481.0	49551.0	
6	73837.0	75013.0	76343.0	77700.0	79034.0	79824.0	
188	17837.0	17864.0	17916.0	17972.0	18024.0	18055.0	
155	47187.0	45989.0	44728.0	43413.0	42050.0	41569.0	
212	34056.0	34156.0	34178.0	34007.0	33745.0	33660.0	

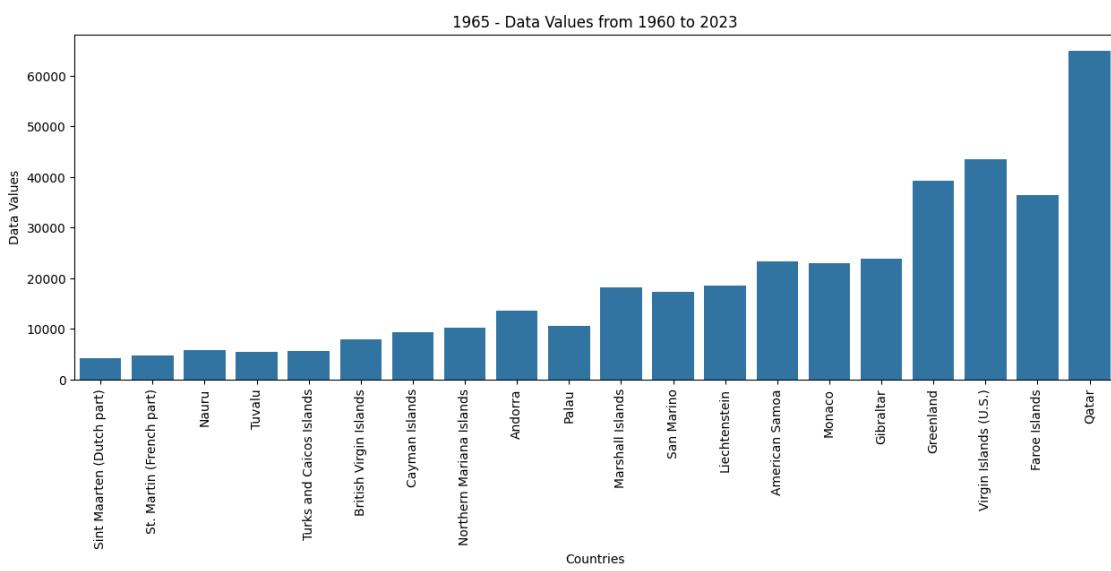
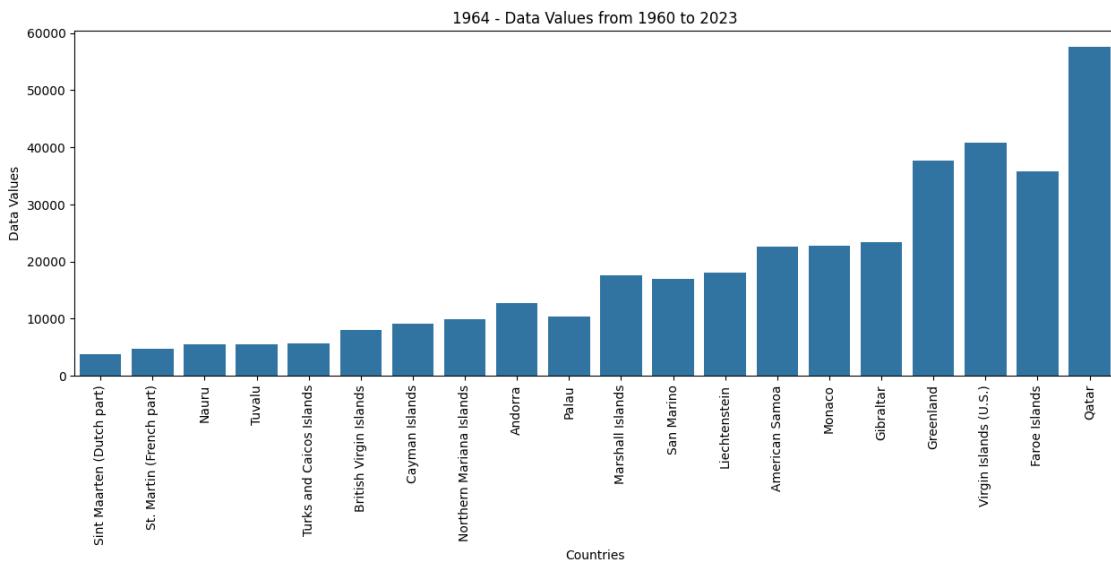
137	37889.0	38181.0	38482.0	38756.0	39039.0	39327.0
11	49463.0	48424.0	47321.0	46189.0	45035.0	44273.0
149	37044.0	37029.0	37034.0	36922.0	36686.0	36469.0
84	32602.0	32648.0	32685.0	32709.0	32669.0	32649.0
91	56172.0	56023.0	56225.0	56367.0	56653.0	56661.0
256	107281.0	107001.0	106669.0	106290.0	105870.0	105413.0
78	50230.0	50955.0	51681.0	52415.0	52889.0	53090.0
200	2711755.0	2766732.0	2807235.0	2760385.0	2688235.0	2695122.0
						2023
225	41163.0					
147	32077.0					
179	12780.0					
245	11396.0					
228	46062.0					
255	31538.0					
52	69310.0					
164	49796.0					
6	80088.0					
188	18058.0					
155	41996.0					
212	33642.0					
137	39584.0					
11	43914.0					
149	36297.0					
84	32688.0					
91	56865.0					
256	104917.0					
78	53270.0					
200	2716391.0					

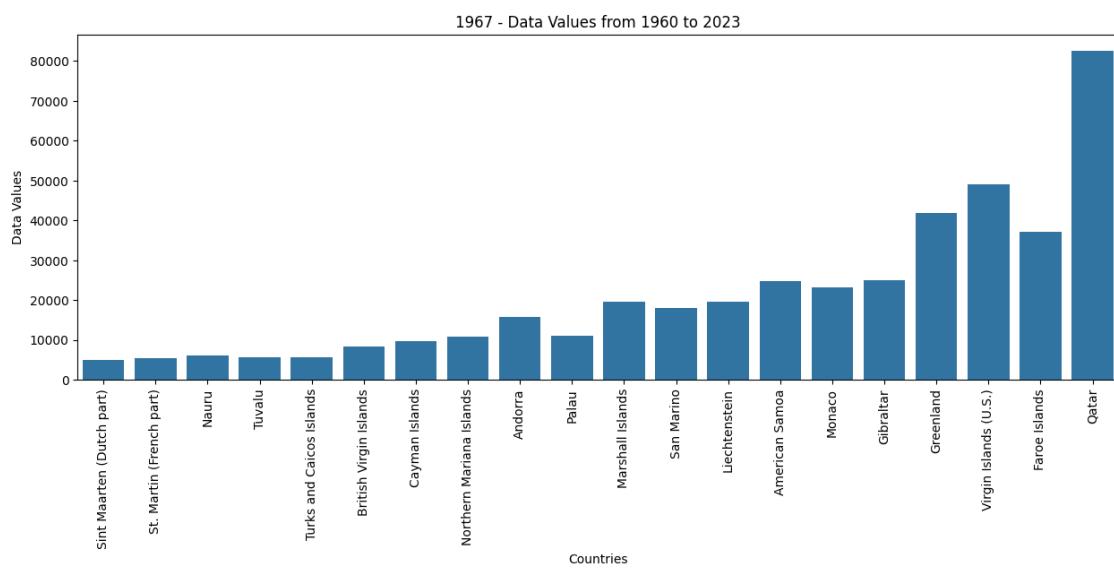
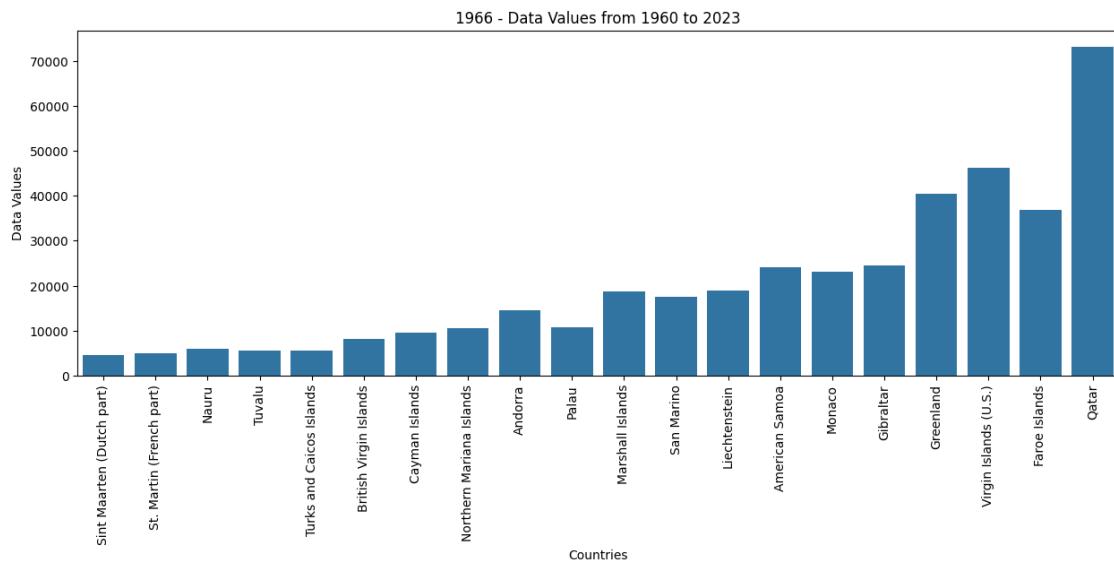
[20 rows x 65 columns]

```
[ ]: country_by_1960_t=country_by_1960.set_index('Country Name').T
for country_name,data_values in country_by_1960_t.iterrows():
    fig=plt.figure(figsize=(15,5))
    sns.barplot(x=data_values.index,y=data_values.values)
    plt.xlabel('Countries')
    plt.ylabel('Data Values')
    plt.title(f'{country_name} - Data Values from 1960 to 2023')
    plt.xticks(rotation=90)
    plt.show()
```

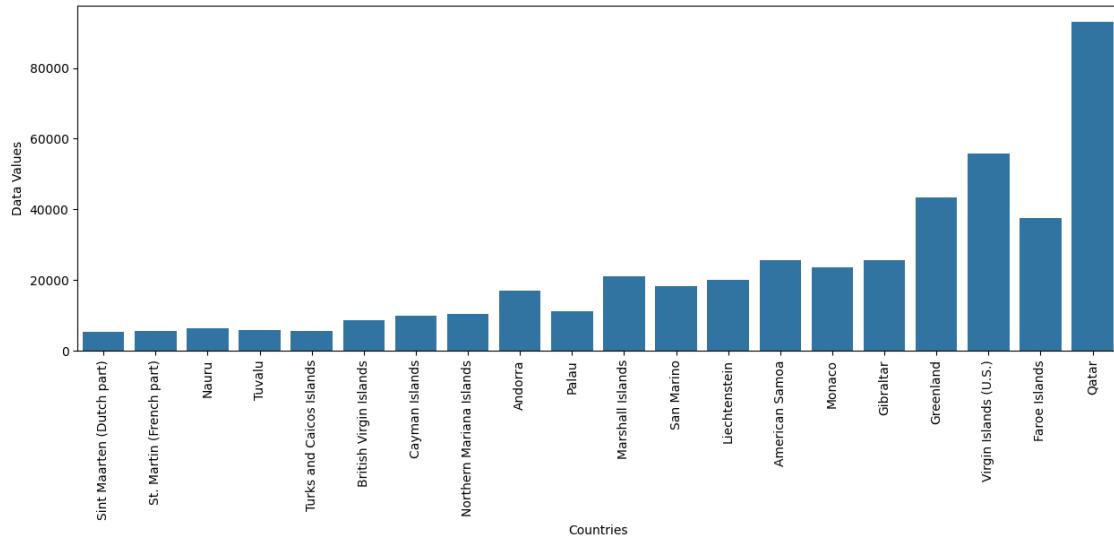




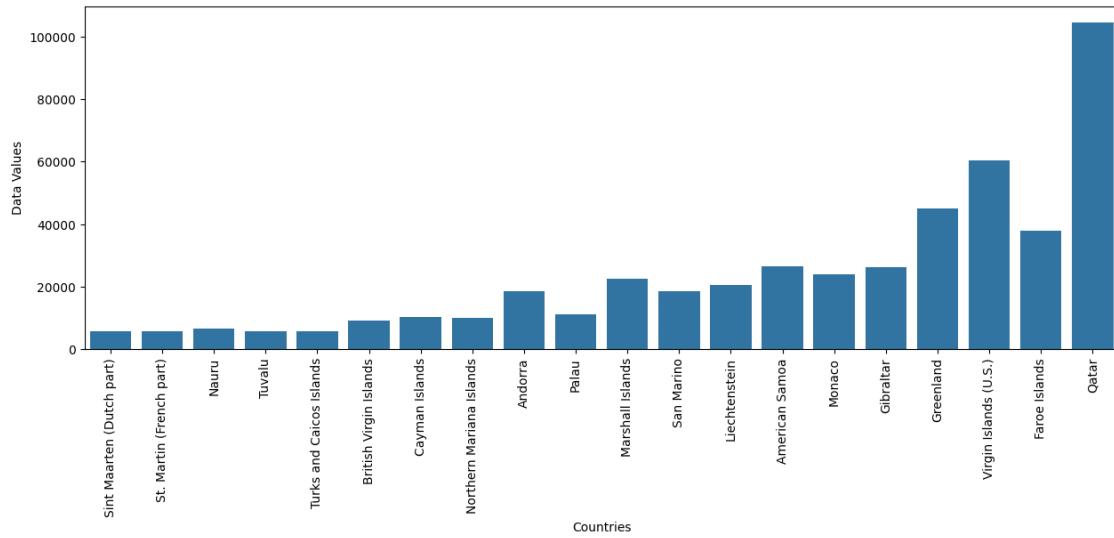




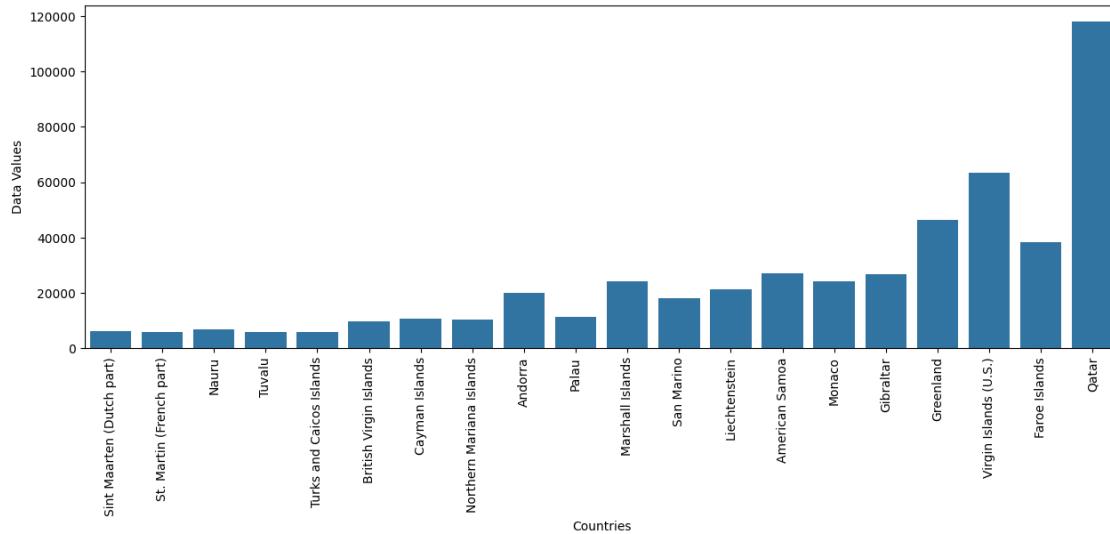
1968 - Data Values from 1960 to 2023



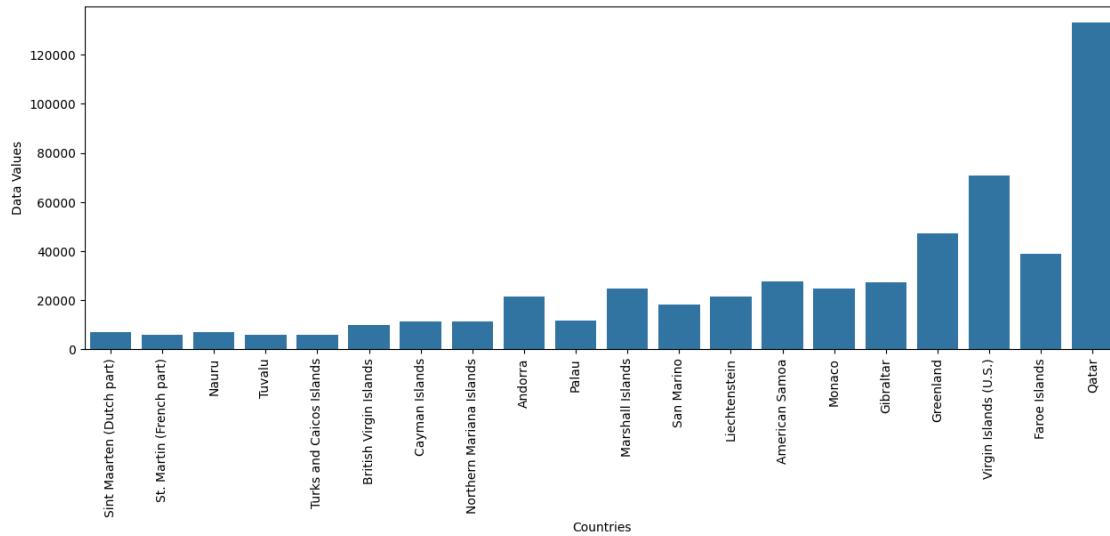
1969 - Data Values from 1960 to 2023



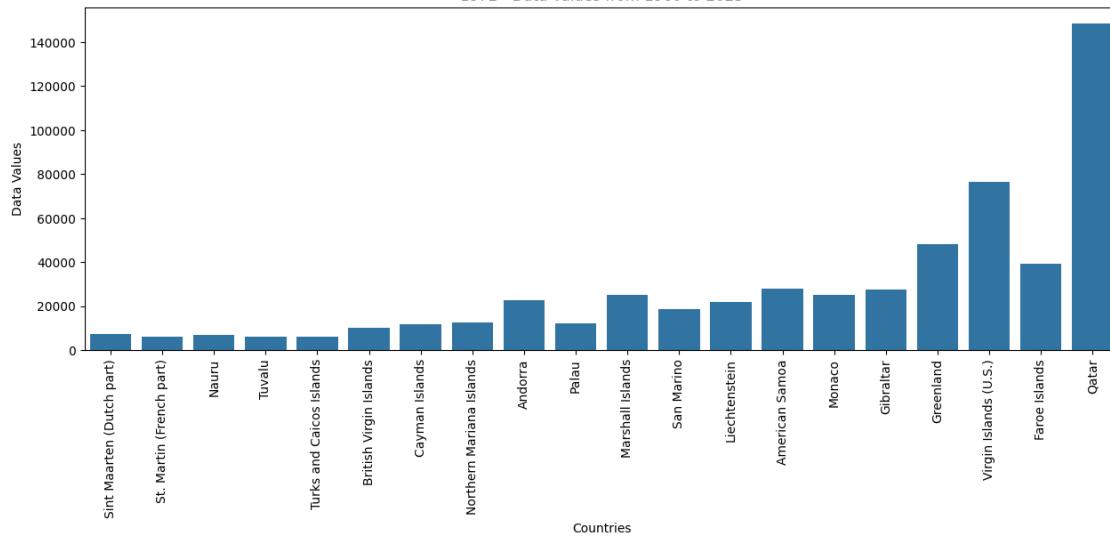
1970 - Data Values from 1960 to 2023



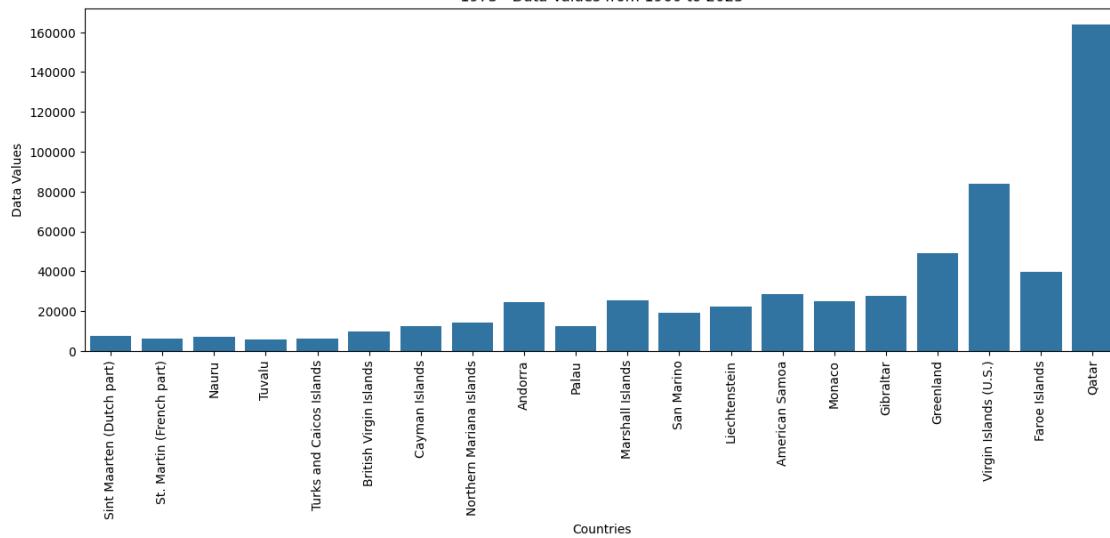
1971 - Data Values from 1960 to 2023



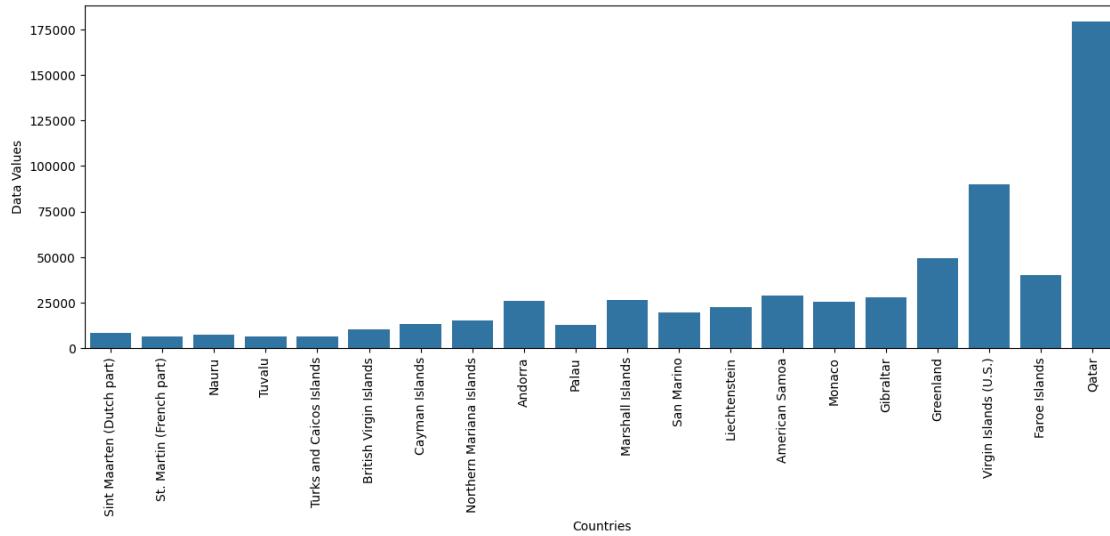
1972 - Data Values from 1960 to 2023



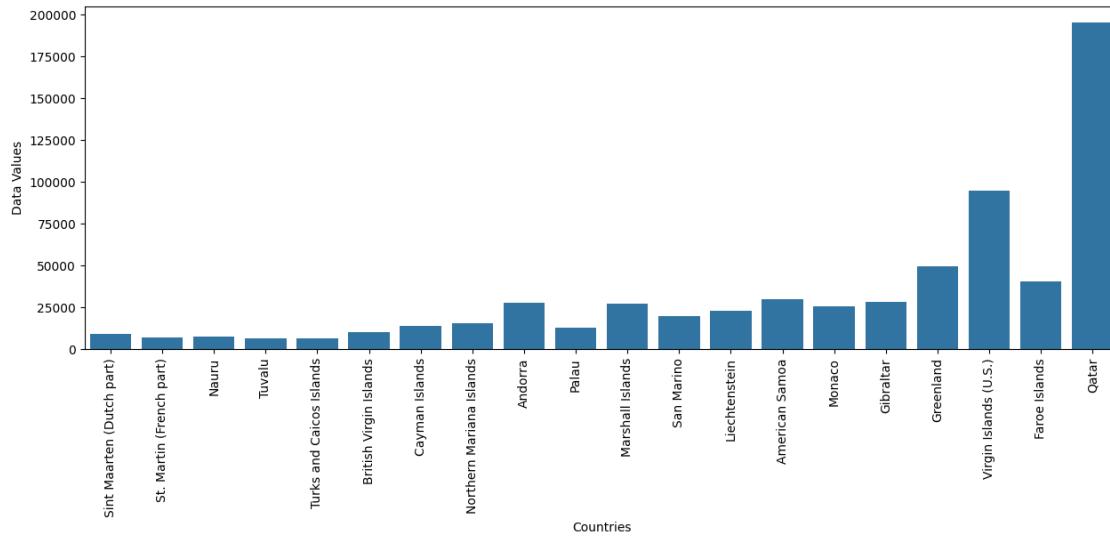
1973 - Data Values from 1960 to 2023



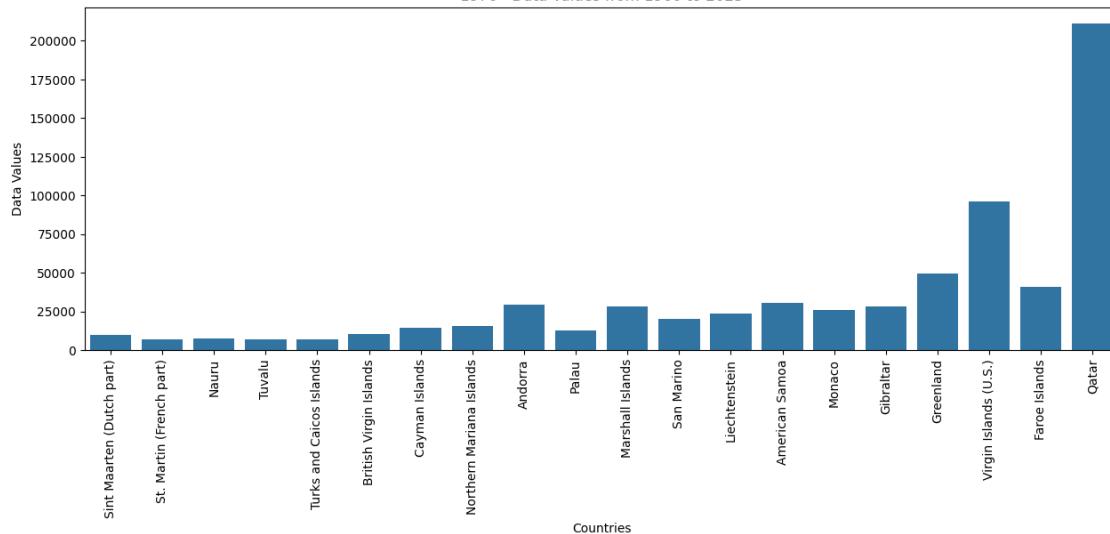
1974 - Data Values from 1960 to 2023



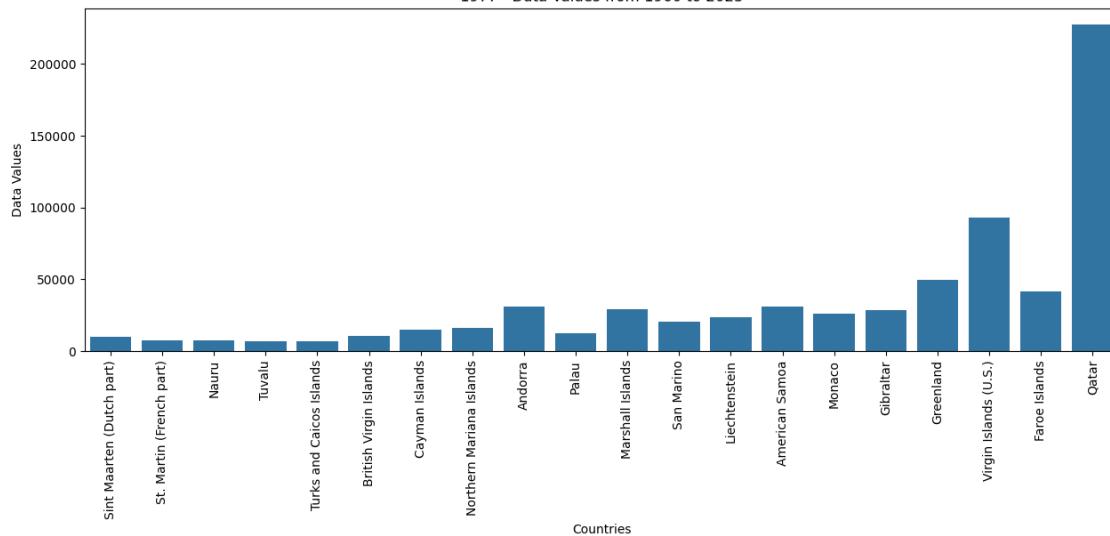
1975 - Data Values from 1960 to 2023

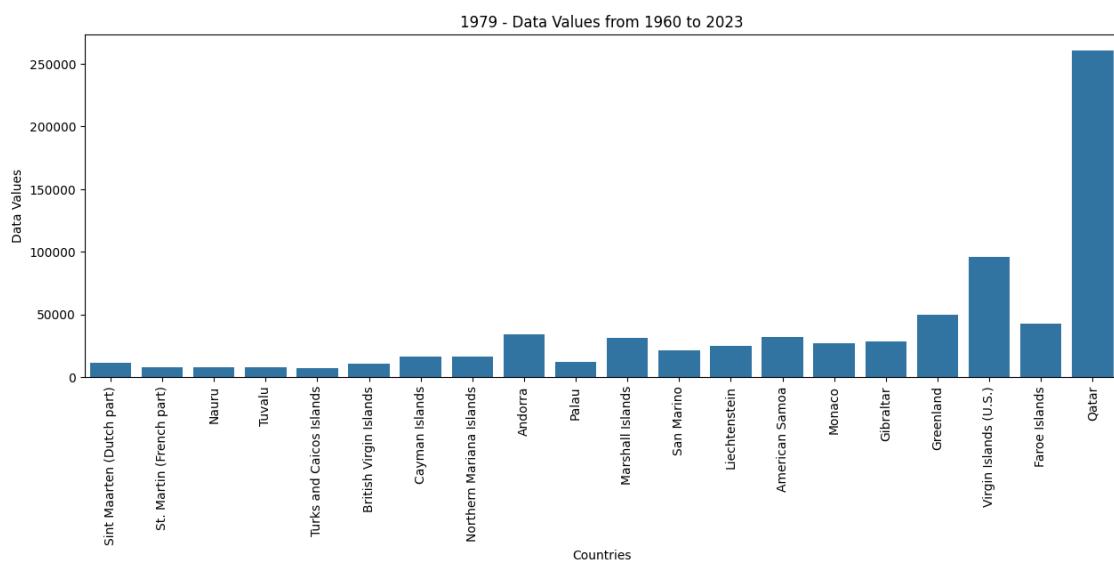
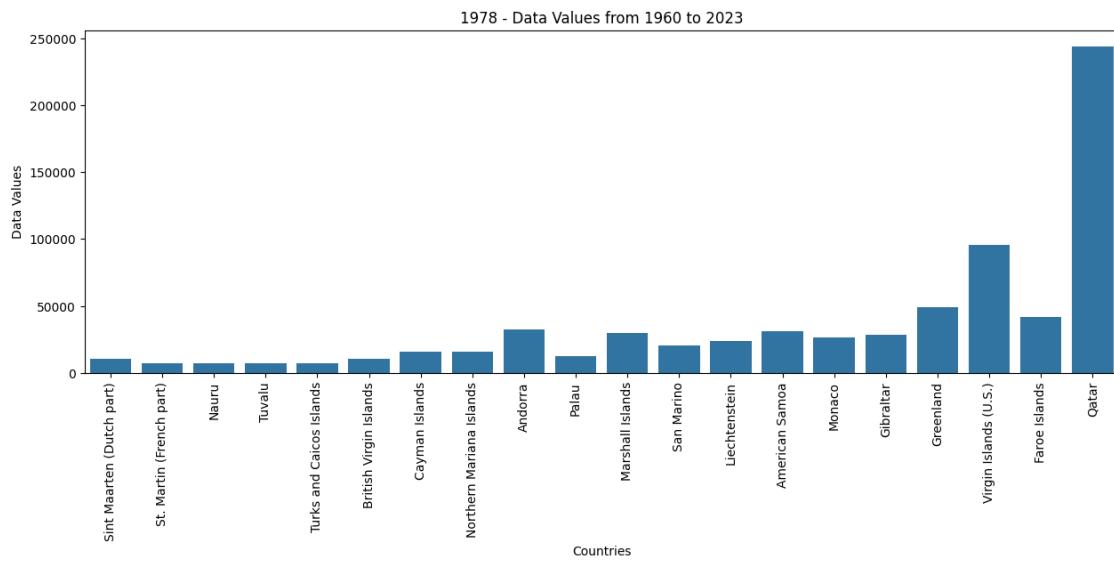


1976 - Data Values from 1960 to 2023

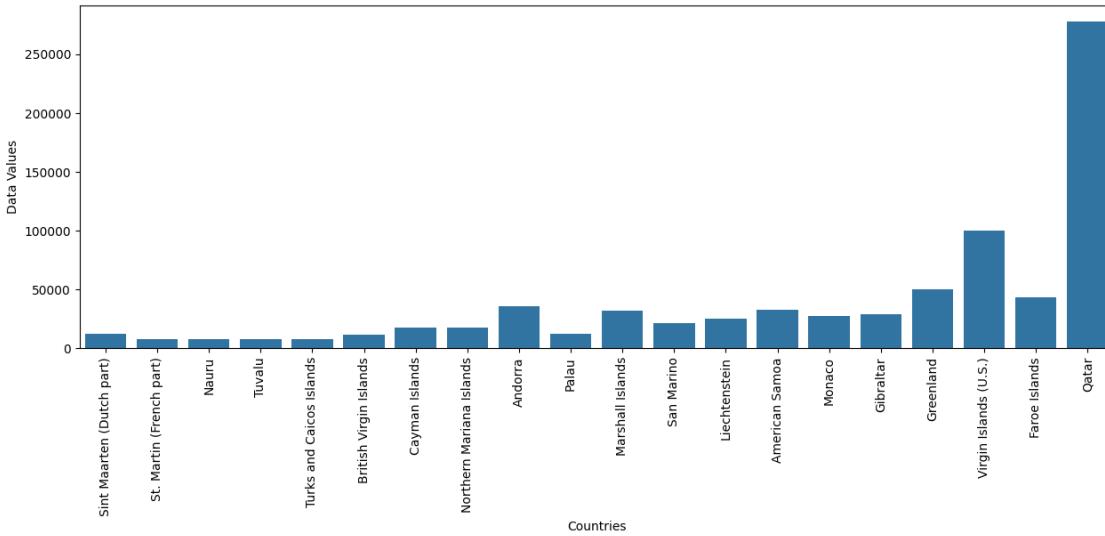


1977 - Data Values from 1960 to 2023

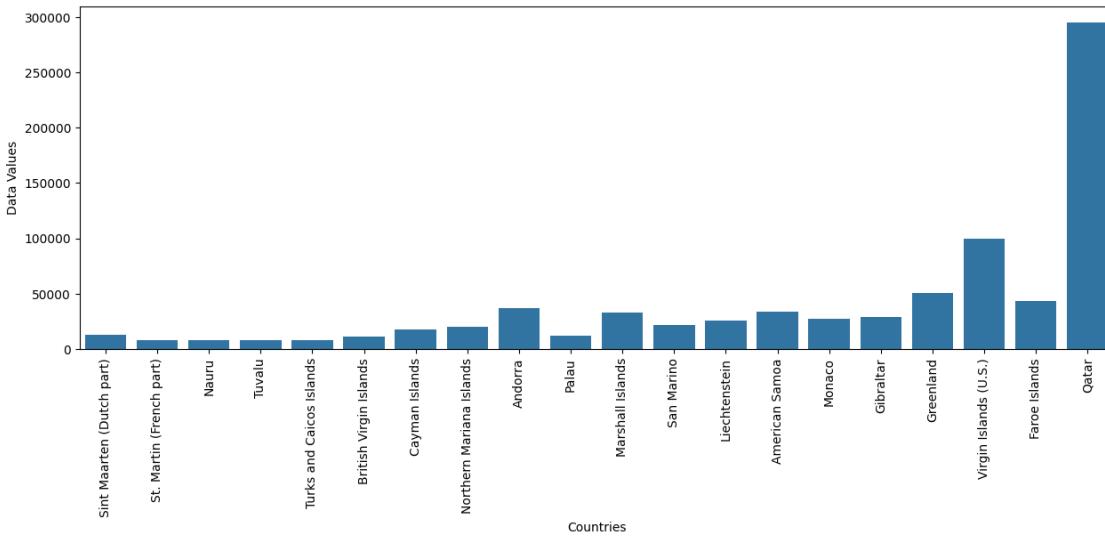




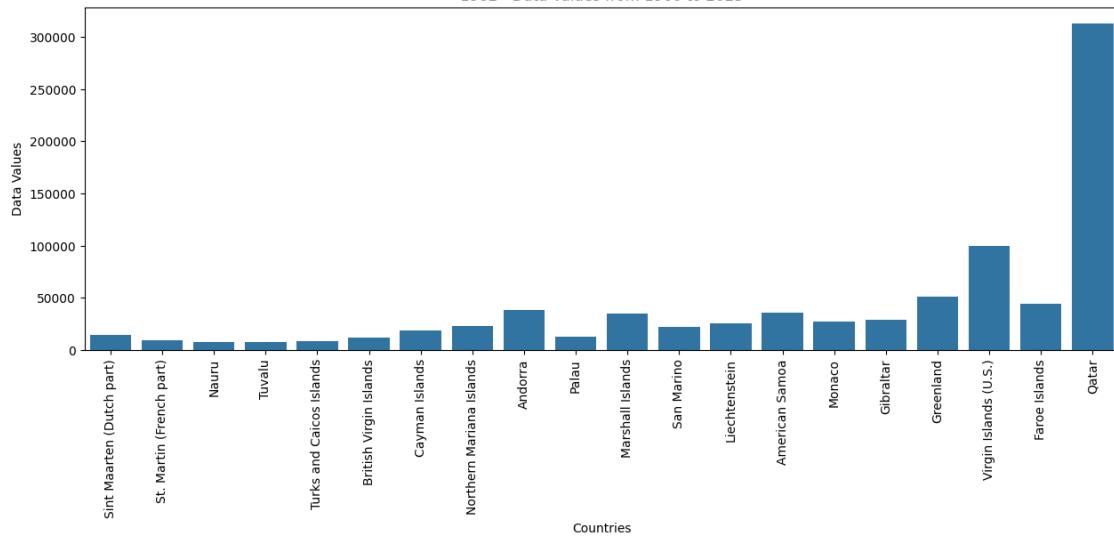
1980 - Data Values from 1960 to 2023



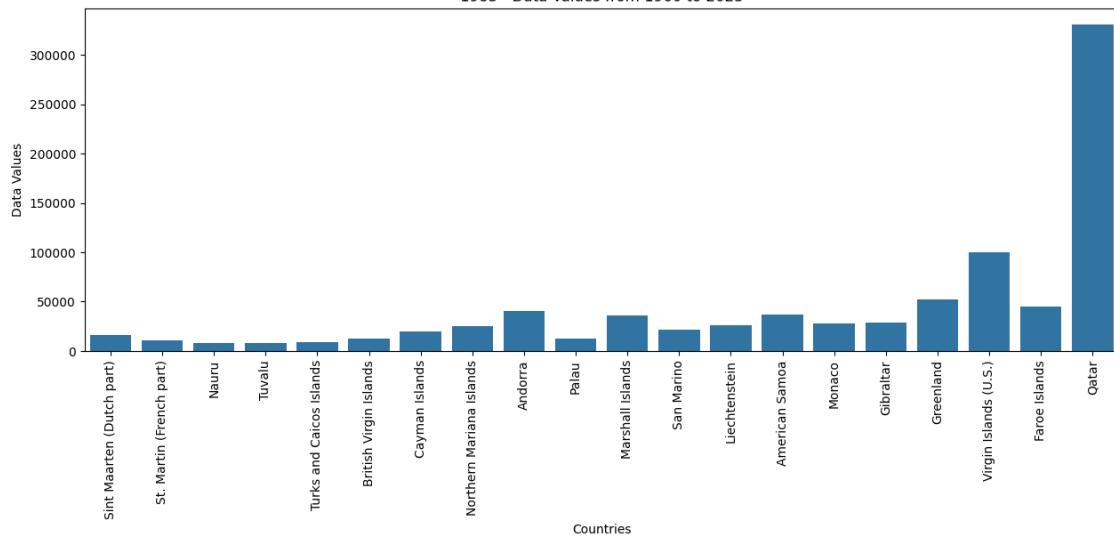
1981 - Data Values from 1960 to 2023



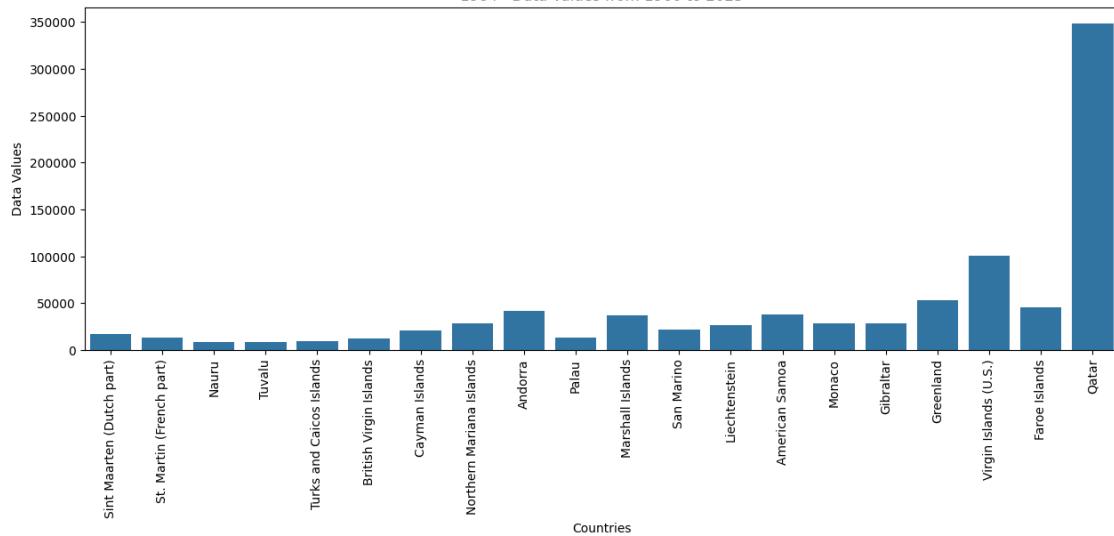
1982 - Data Values from 1960 to 2023



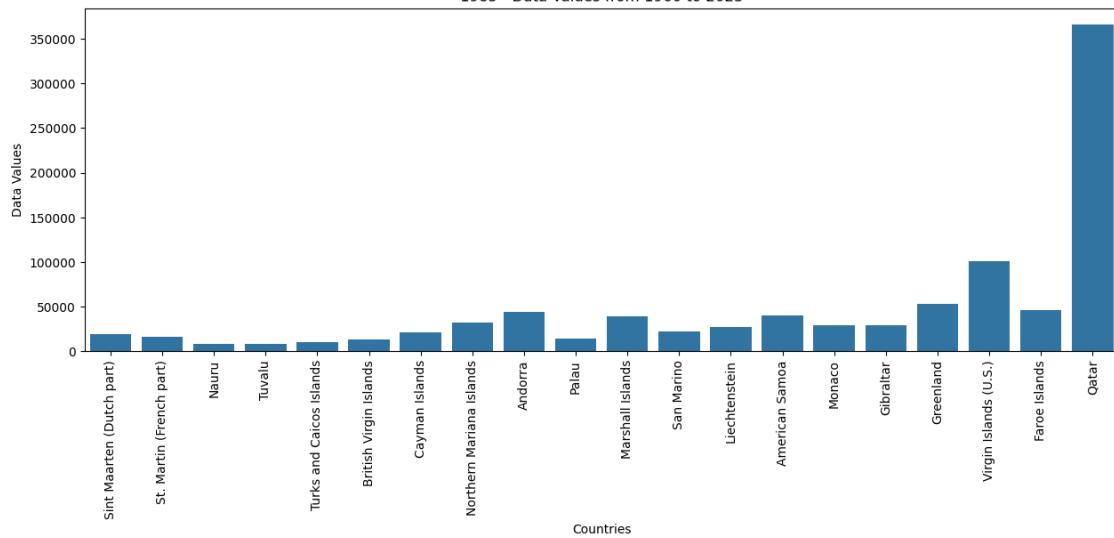
1983 - Data Values from 1960 to 2023

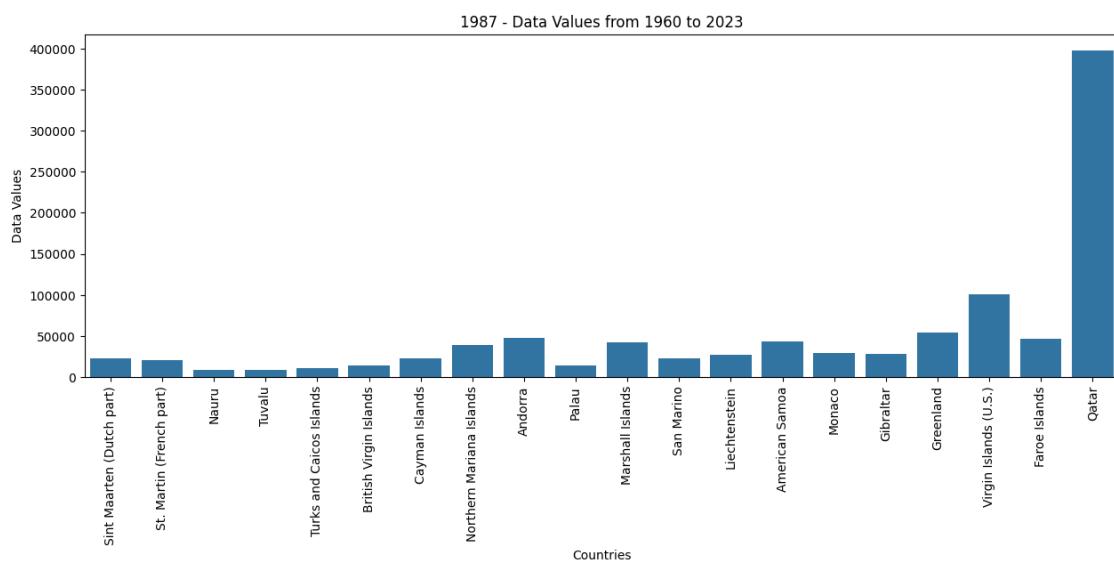
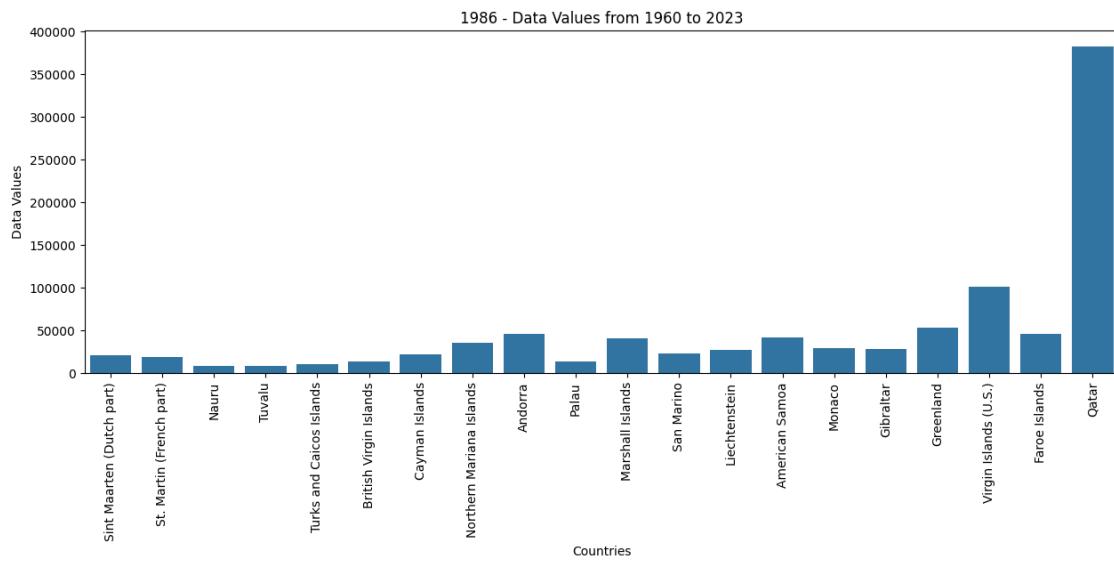


1984 - Data Values from 1960 to 2023

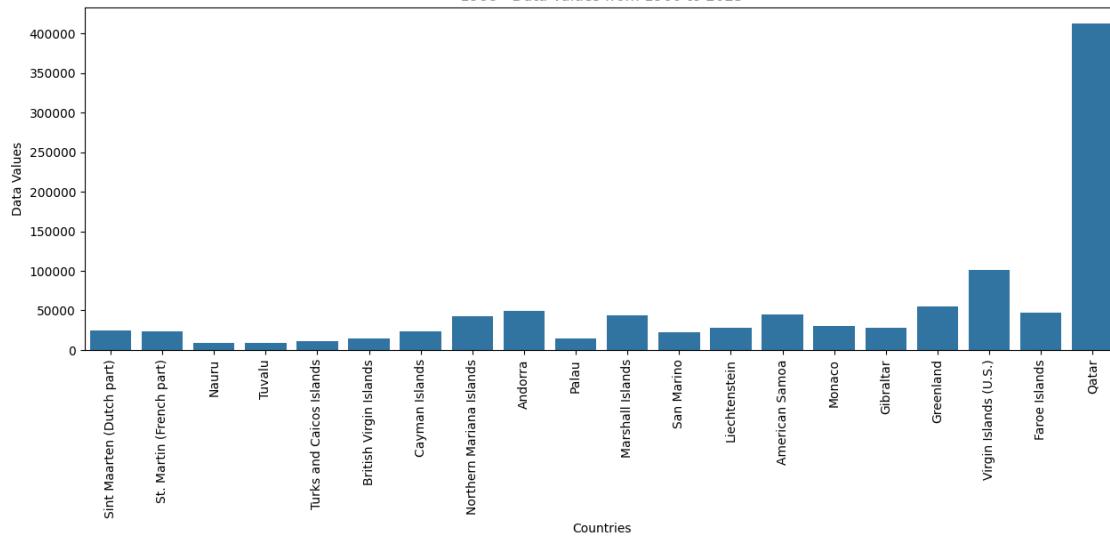


1985 - Data Values from 1960 to 2023

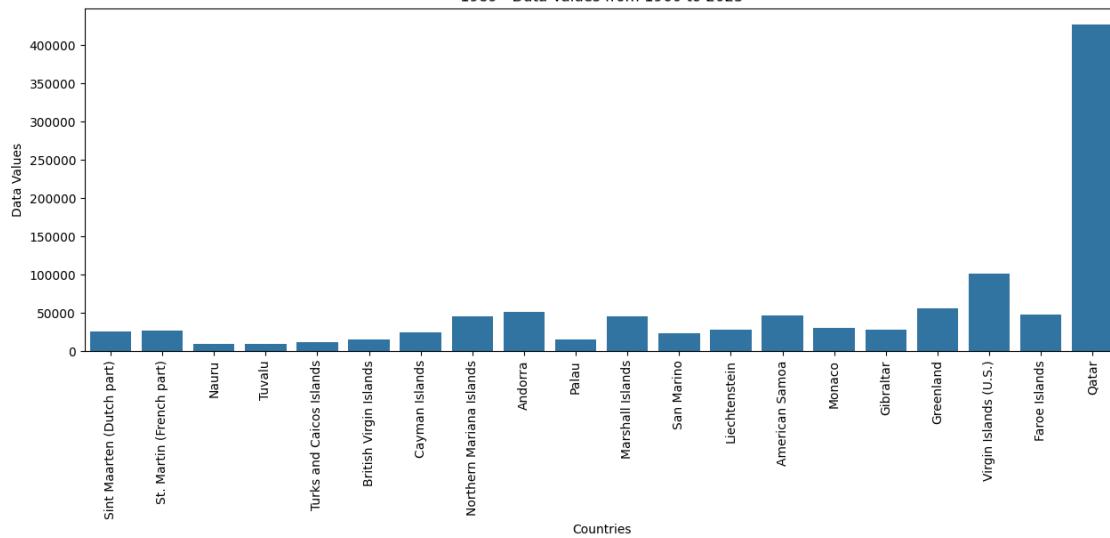




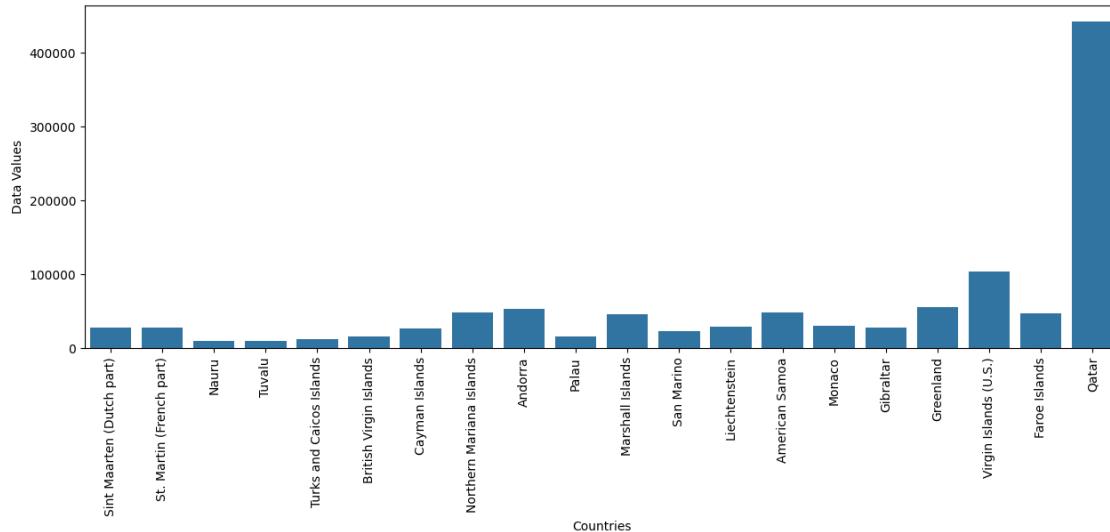
1988 - Data Values from 1960 to 2023



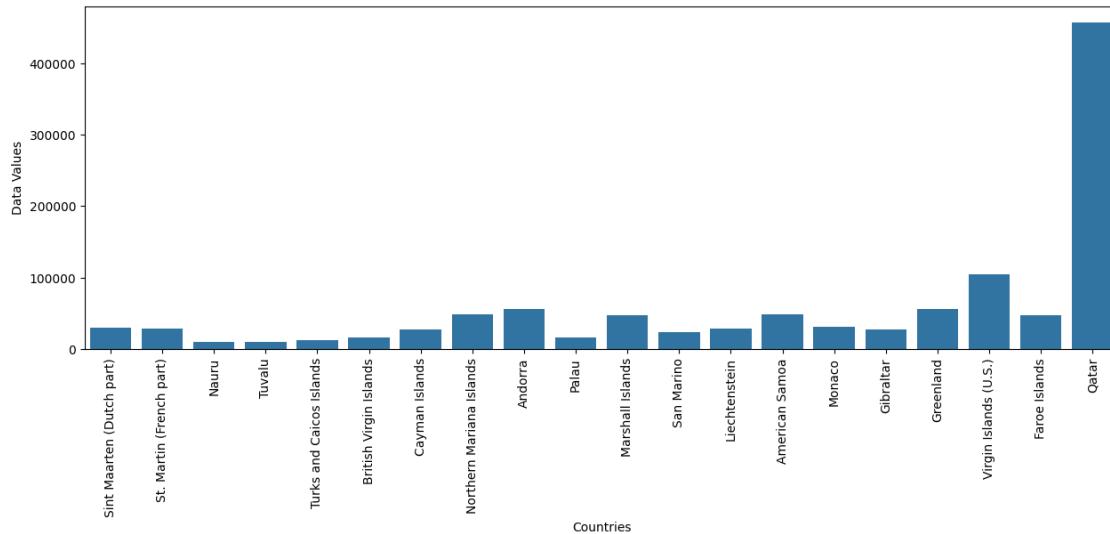
1989 - Data Values from 1960 to 2023



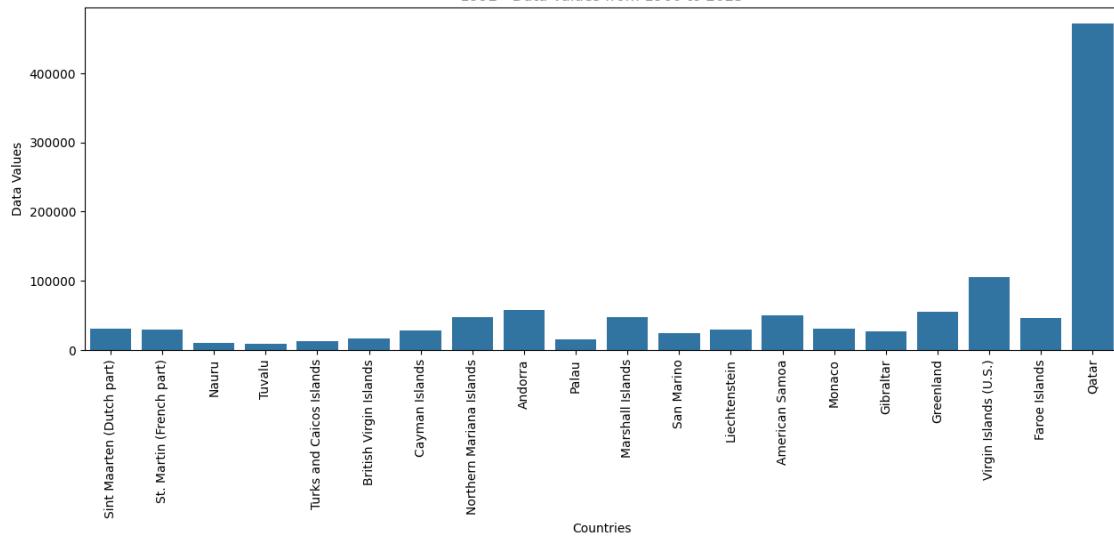
1990 - Data Values from 1960 to 2023



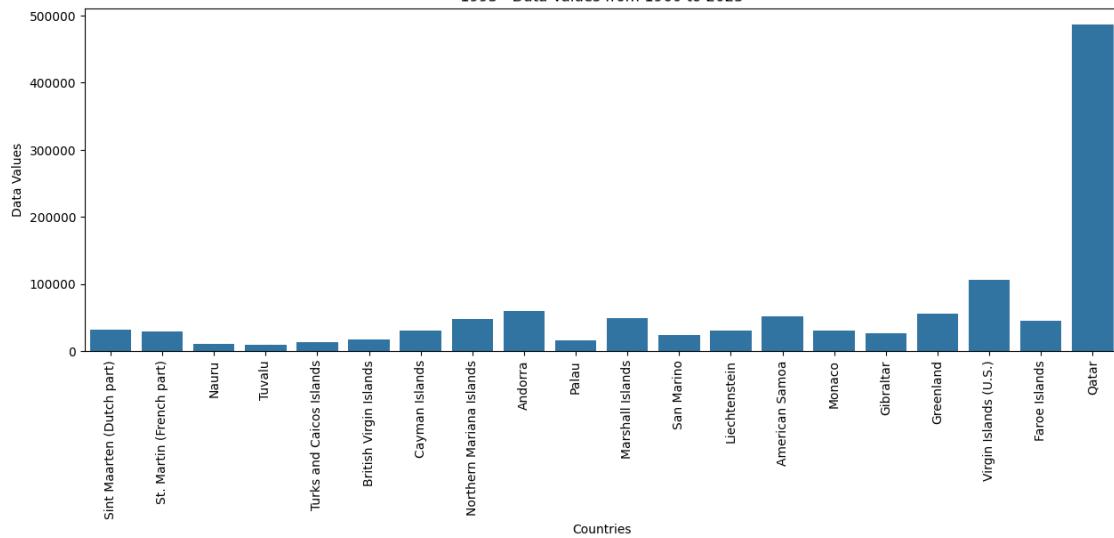
1991 - Data Values from 1960 to 2023



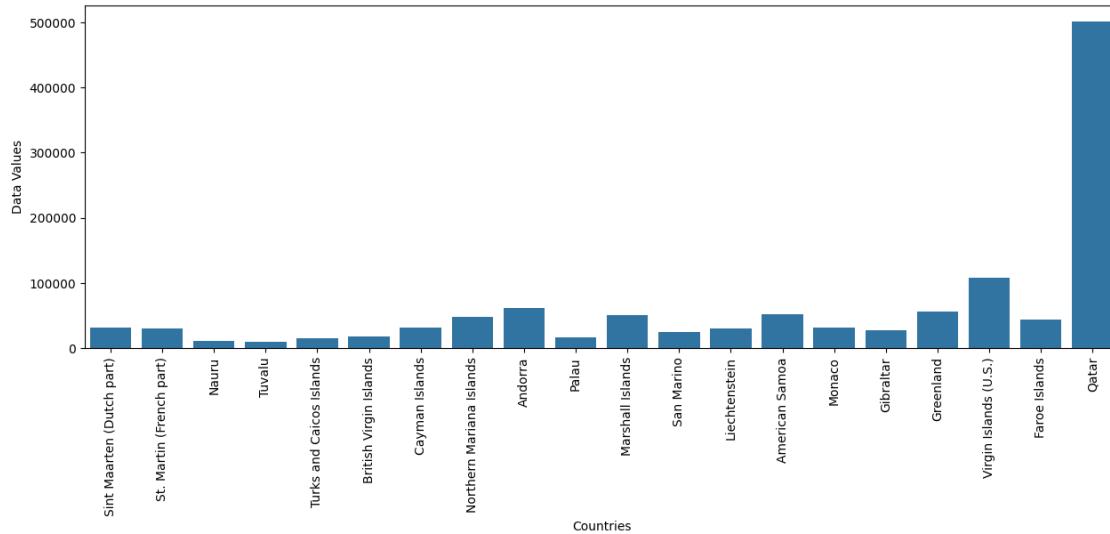
1992 - Data Values from 1960 to 2023



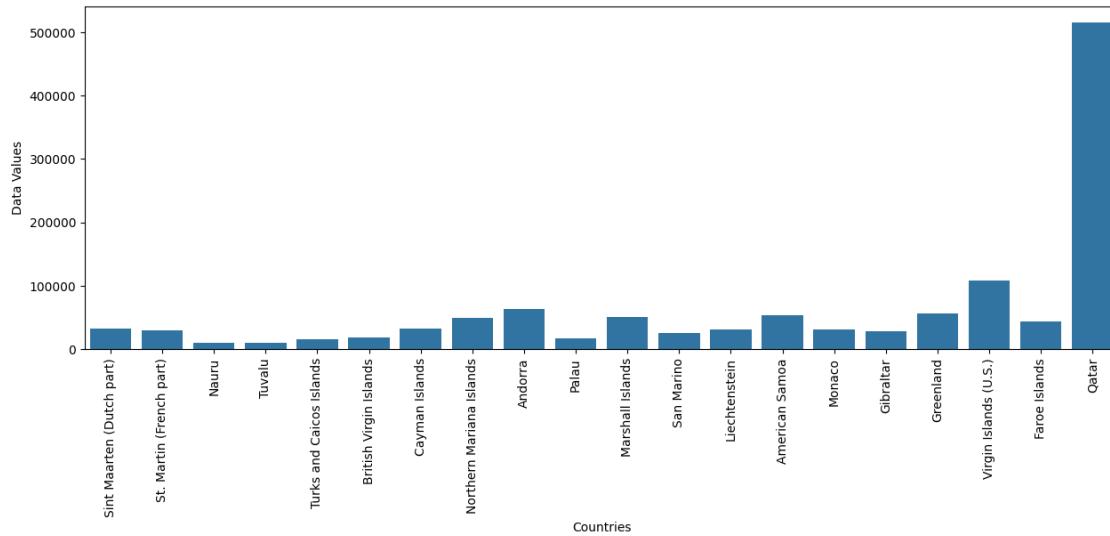
1993 - Data Values from 1960 to 2023



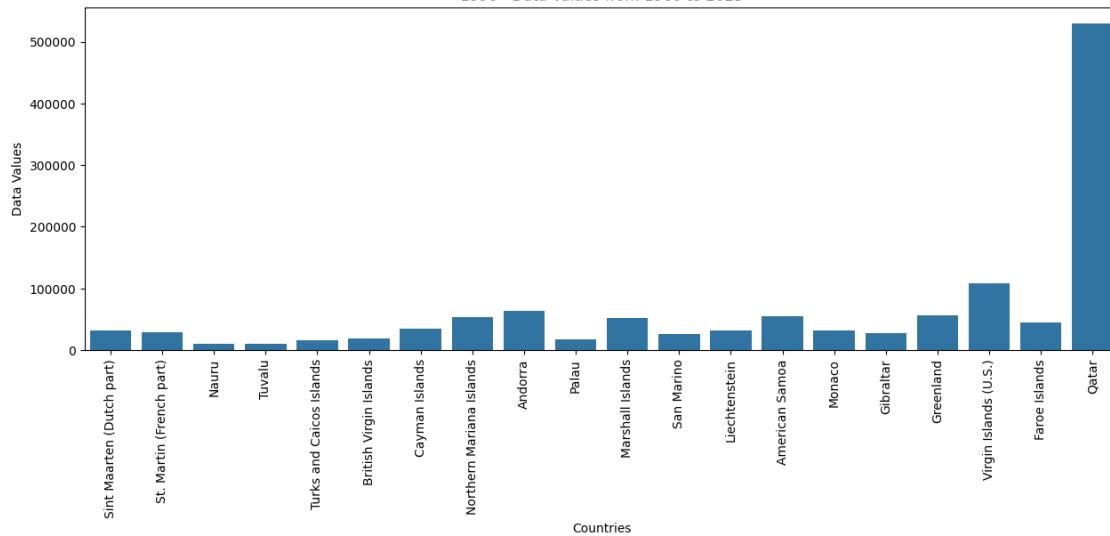
1994 - Data Values from 1960 to 2023



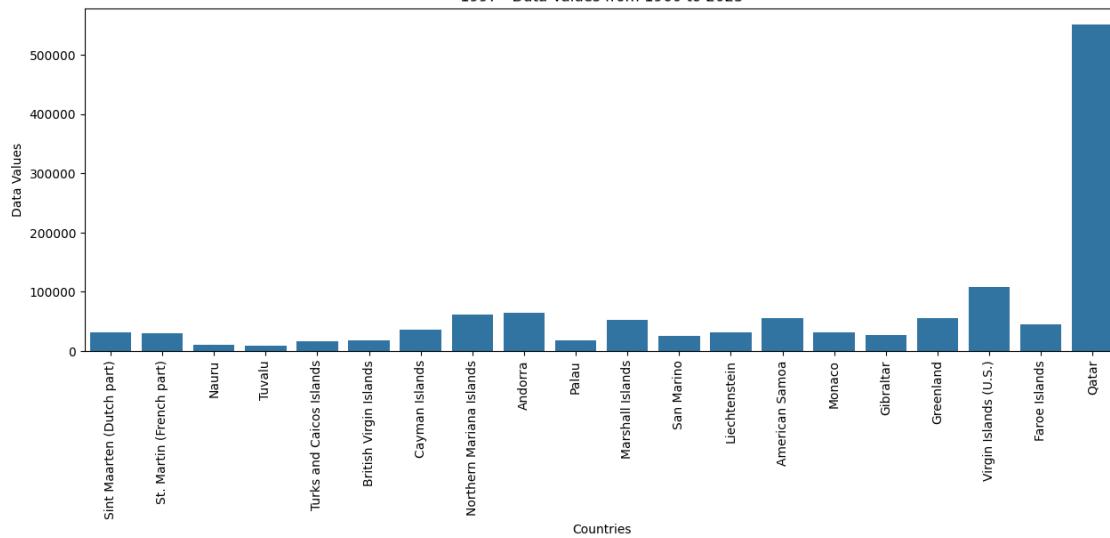
1995 - Data Values from 1960 to 2023

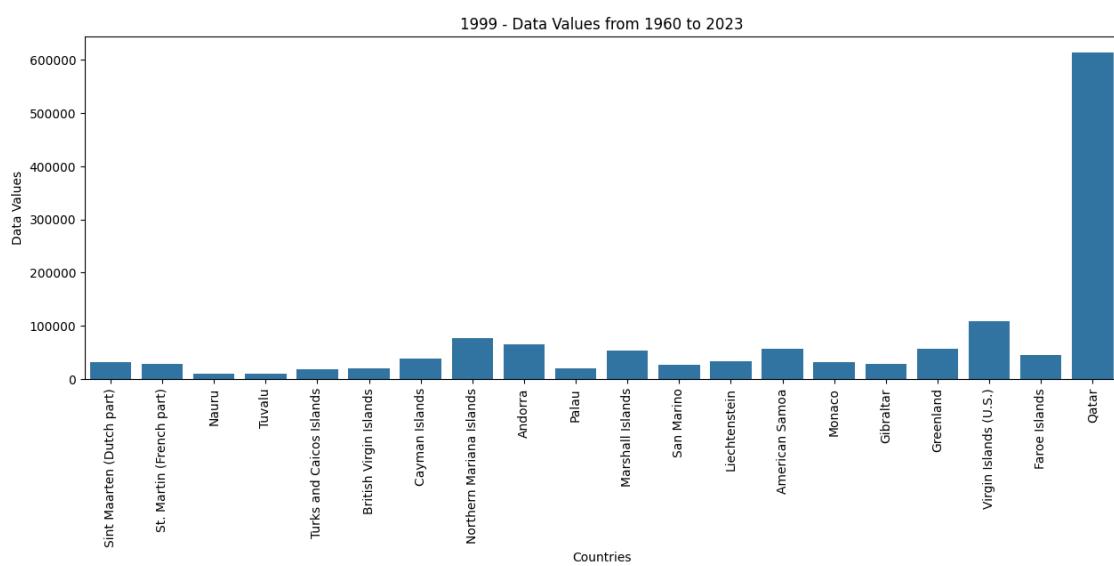
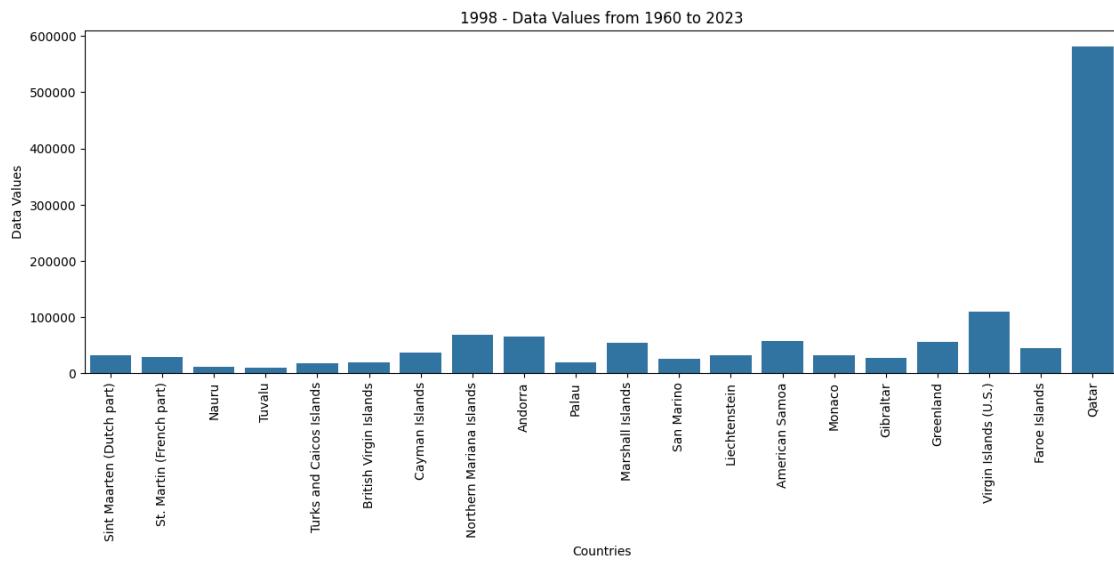


1996 - Data Values from 1960 to 2023

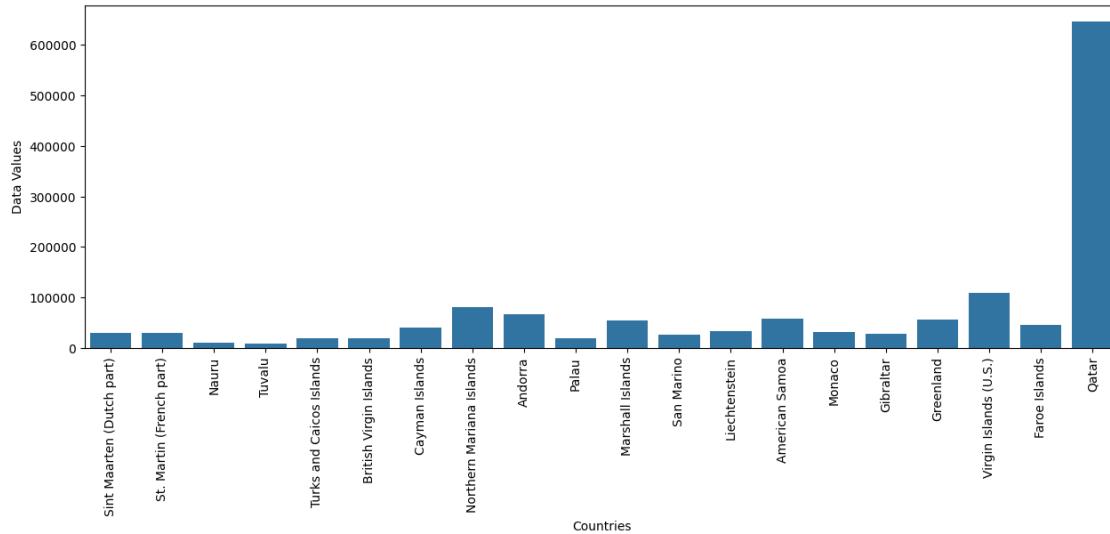


1997 - Data Values from 1960 to 2023

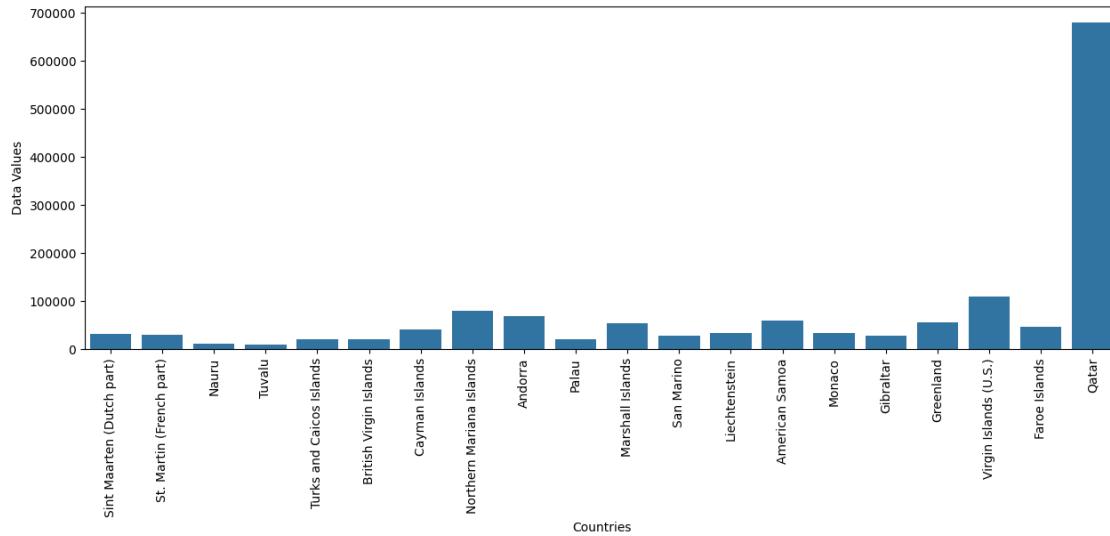




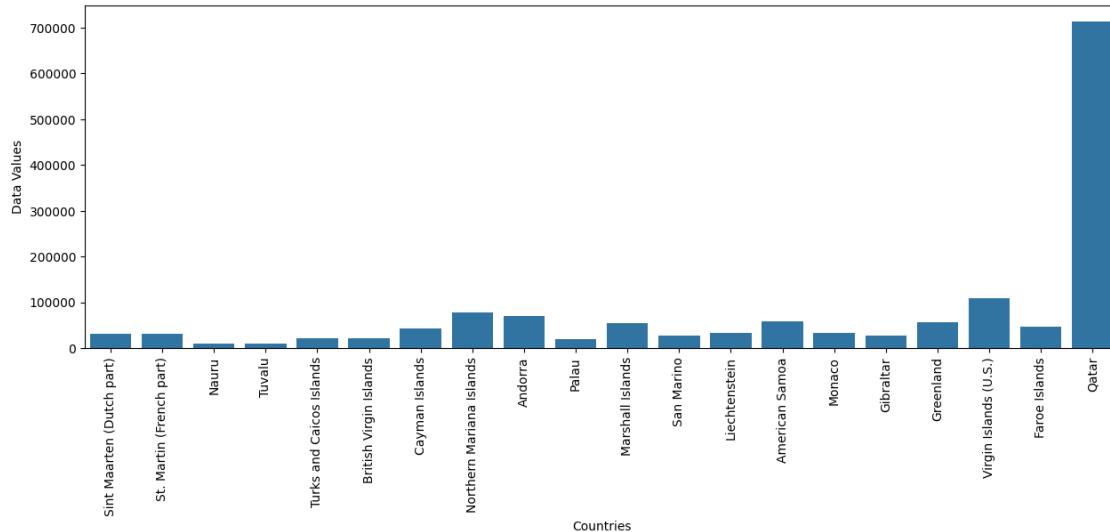
2000 - Data Values from 1960 to 2023



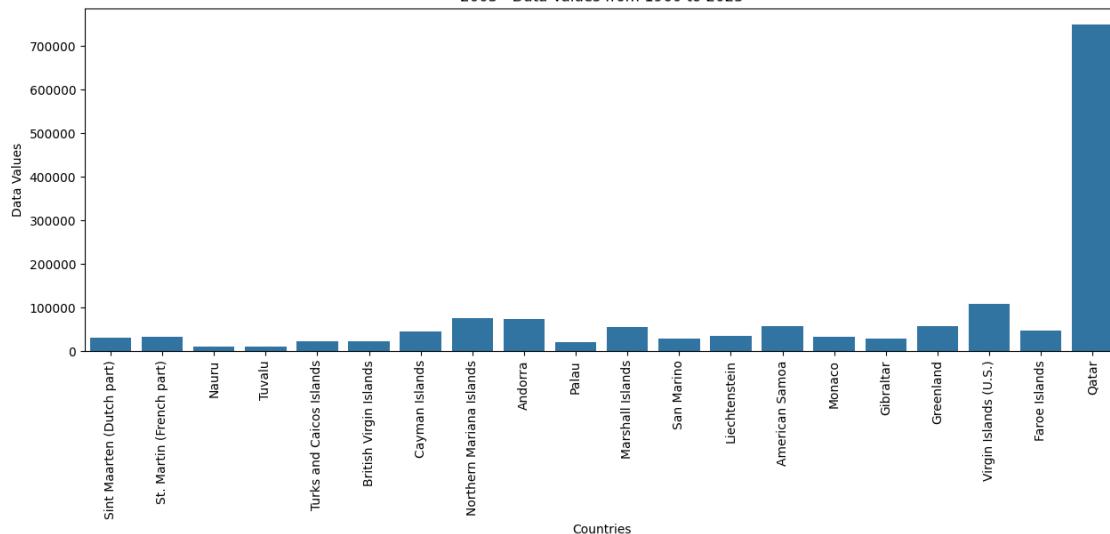
2001 - Data Values from 1960 to 2023



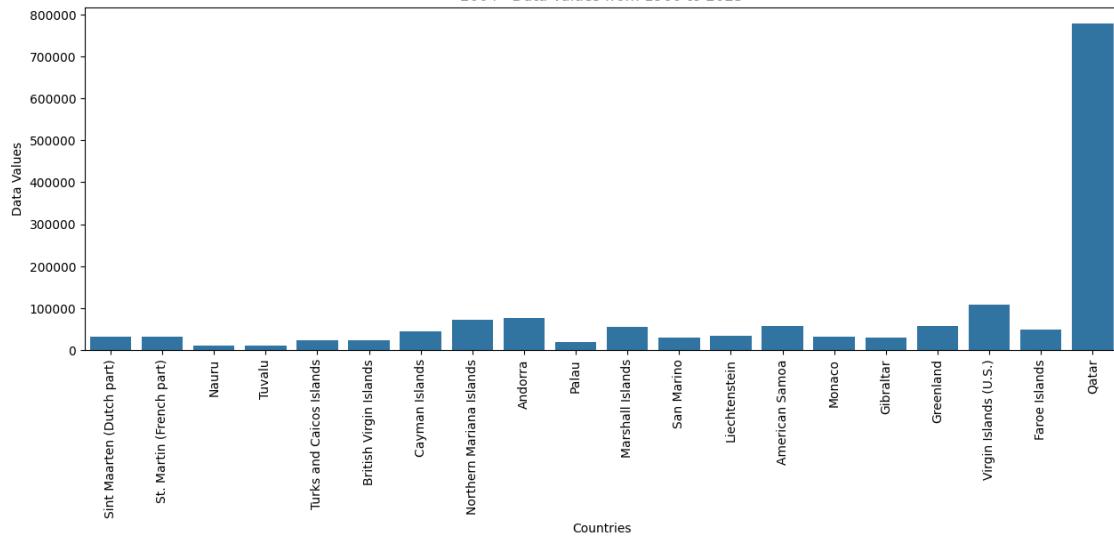
2002 - Data Values from 1960 to 2023



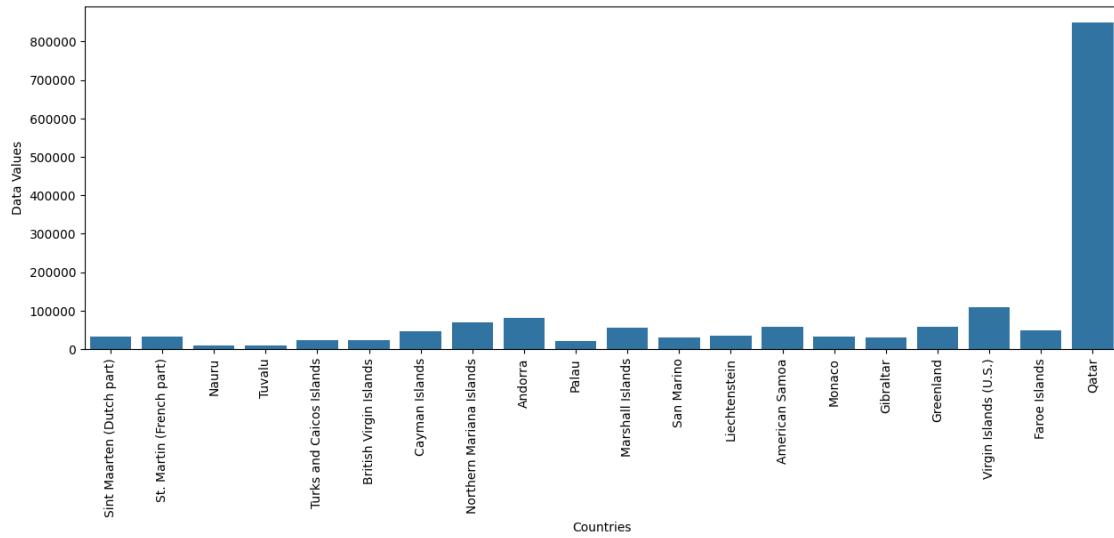
2003 - Data Values from 1960 to 2023

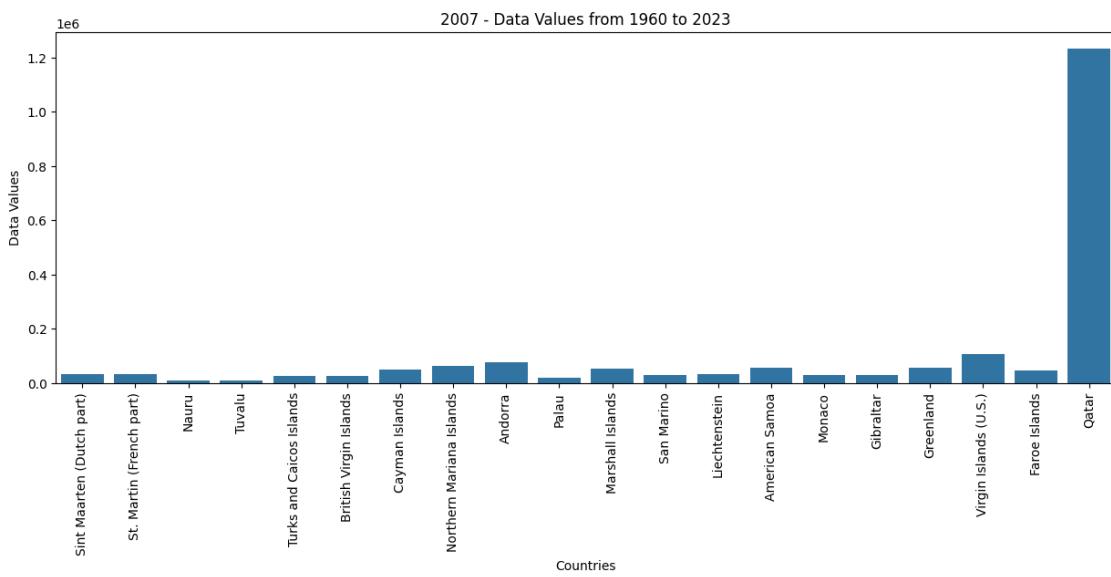
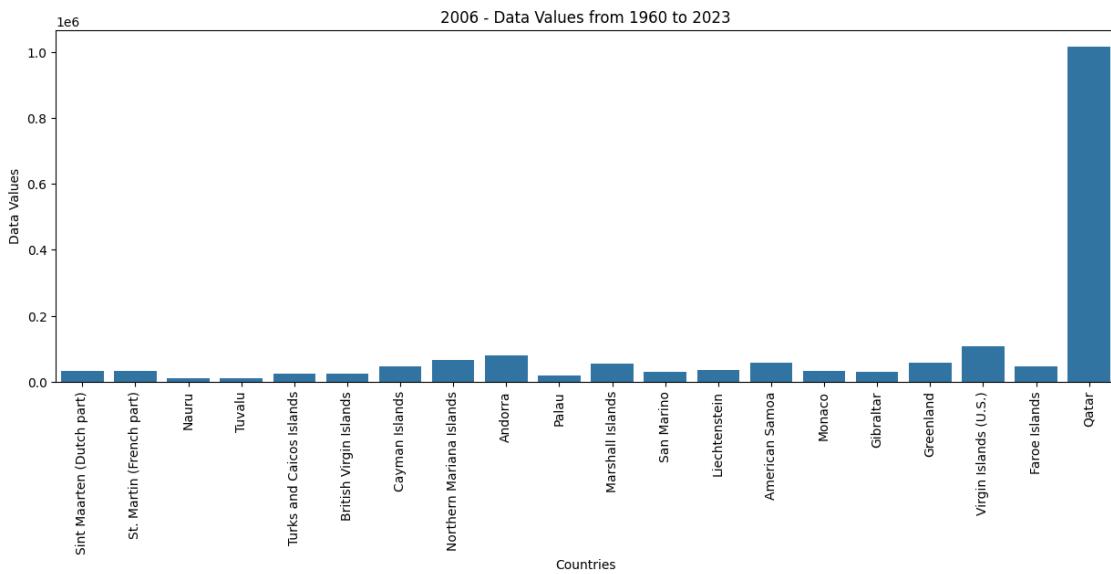


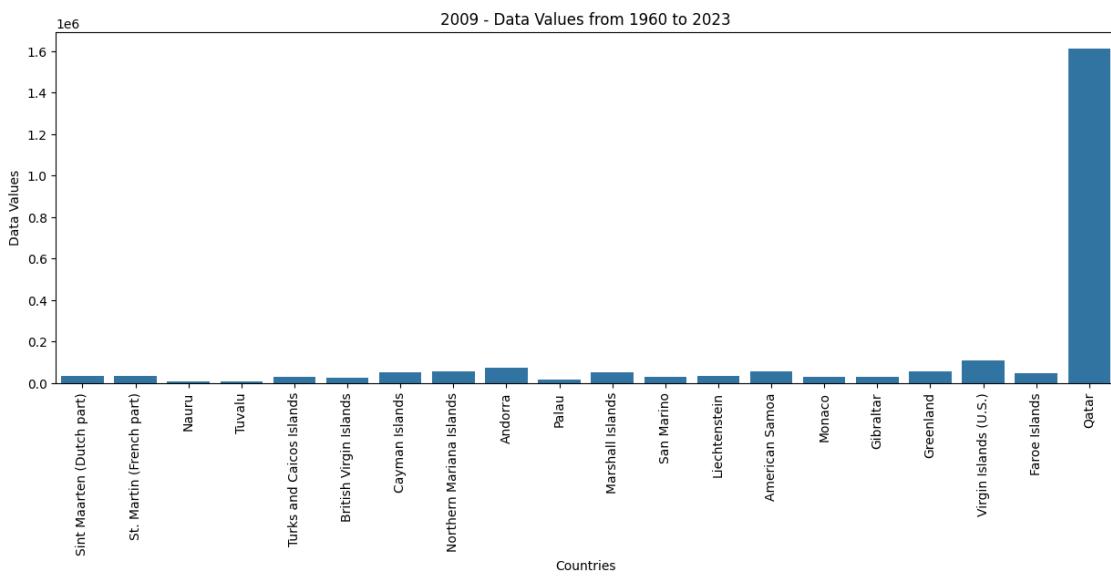
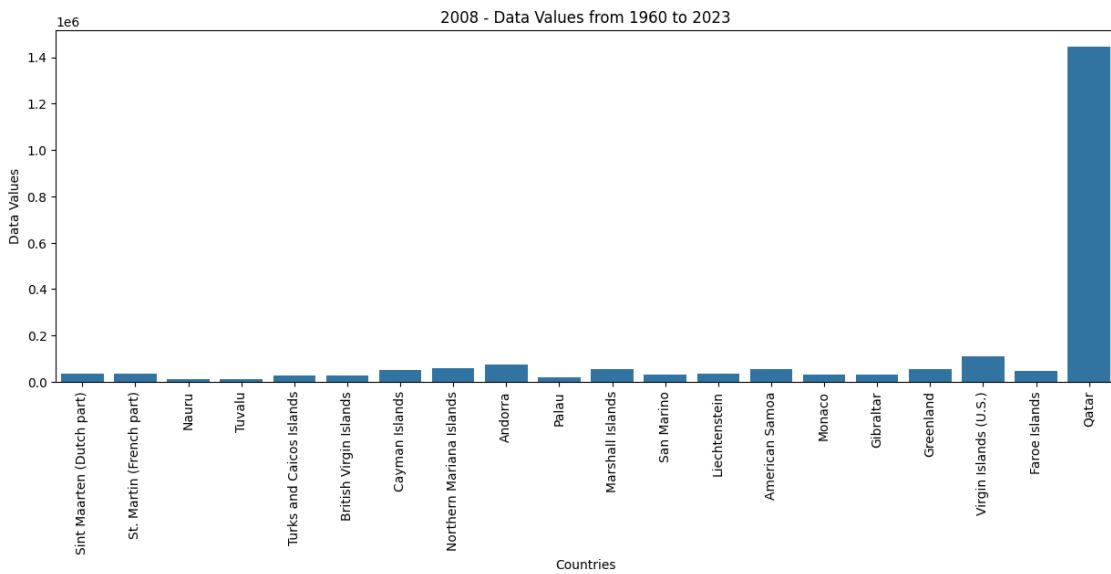
2004 - Data Values from 1960 to 2023

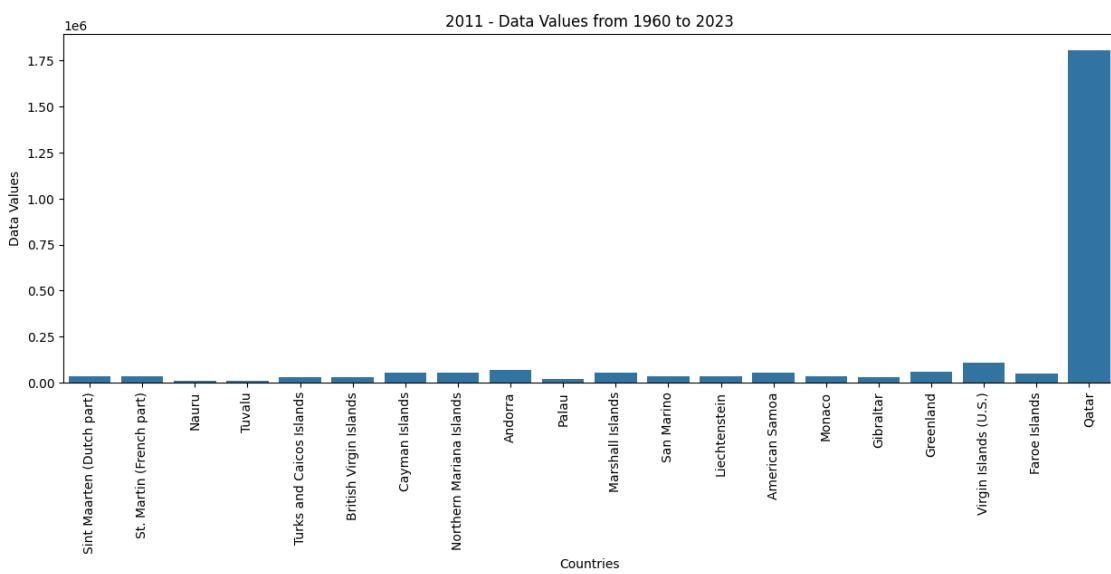
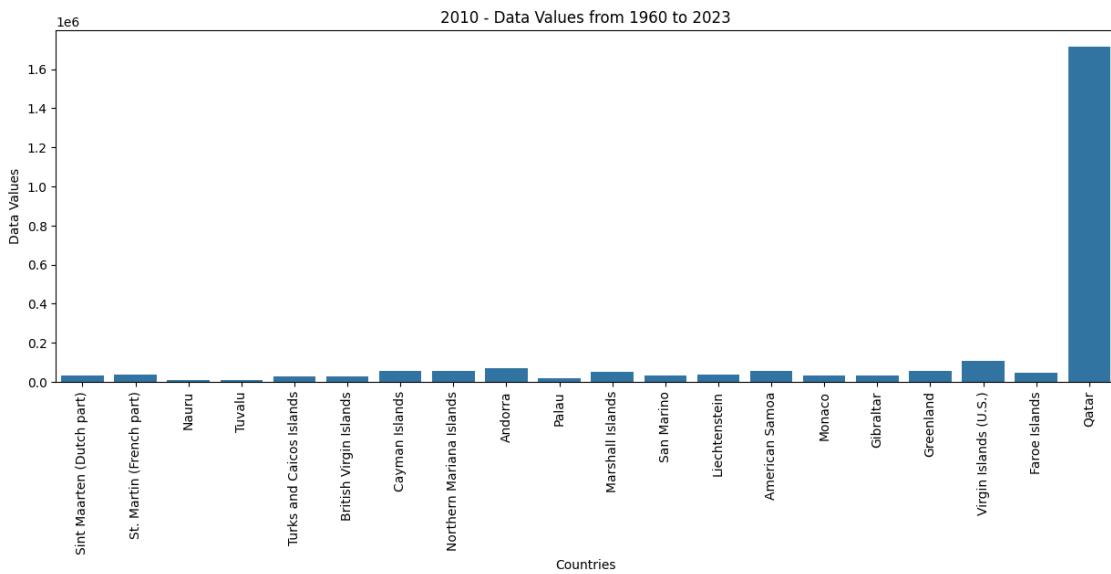


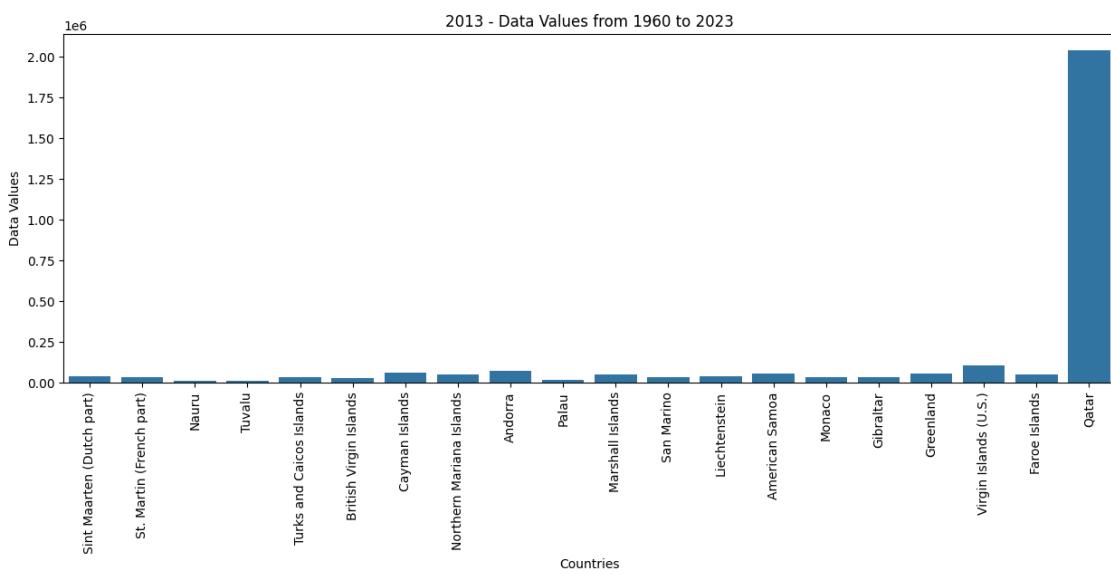
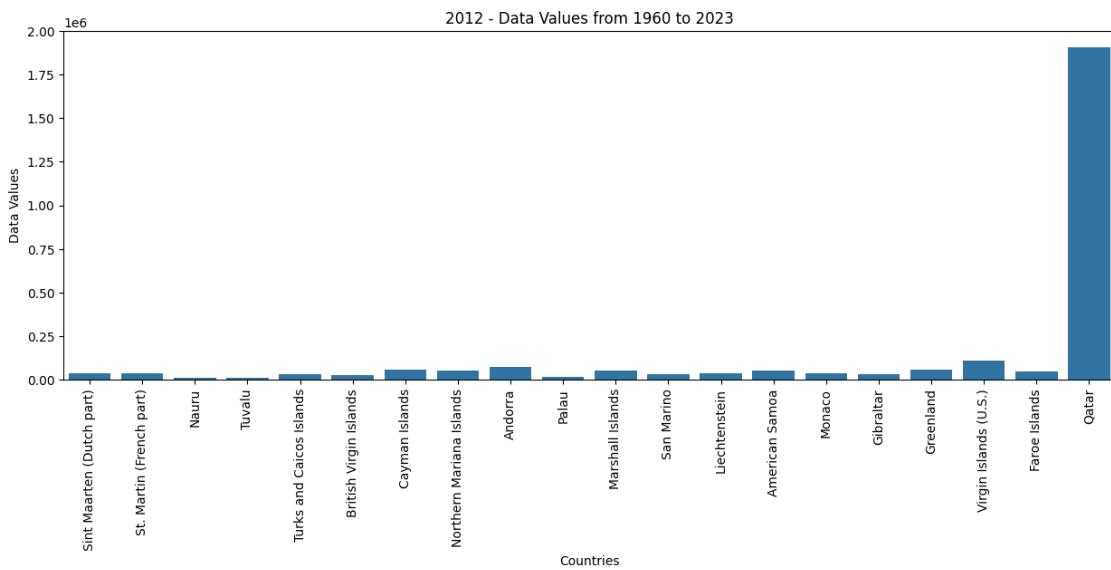
2005 - Data Values from 1960 to 2023

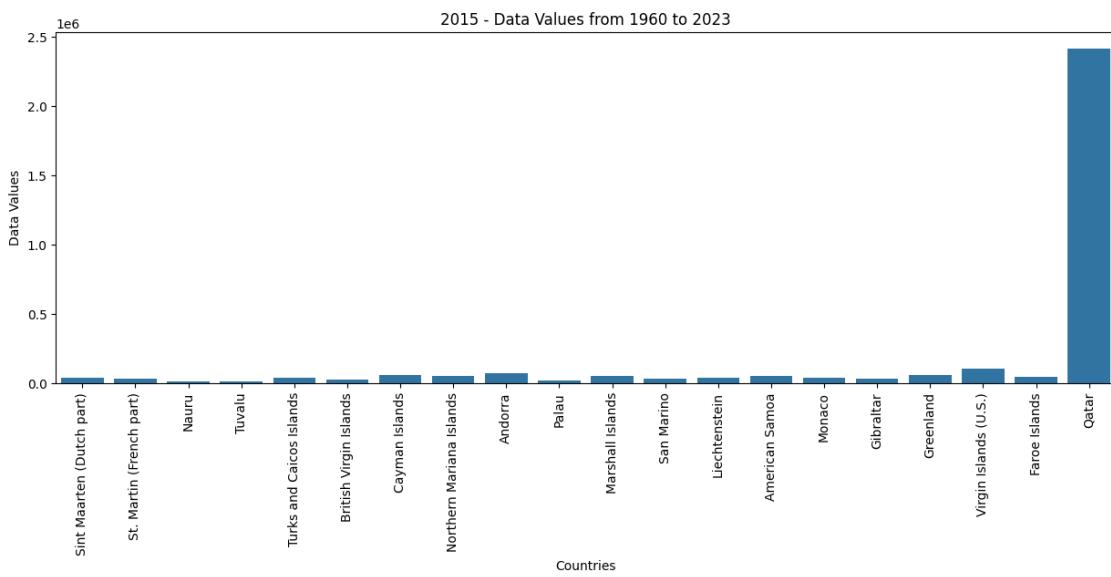
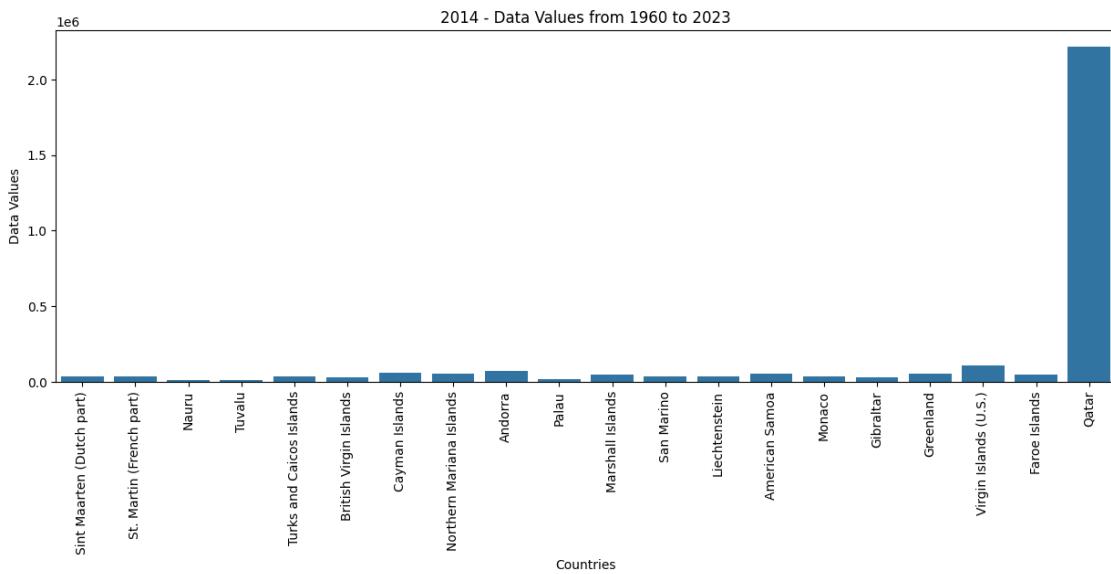


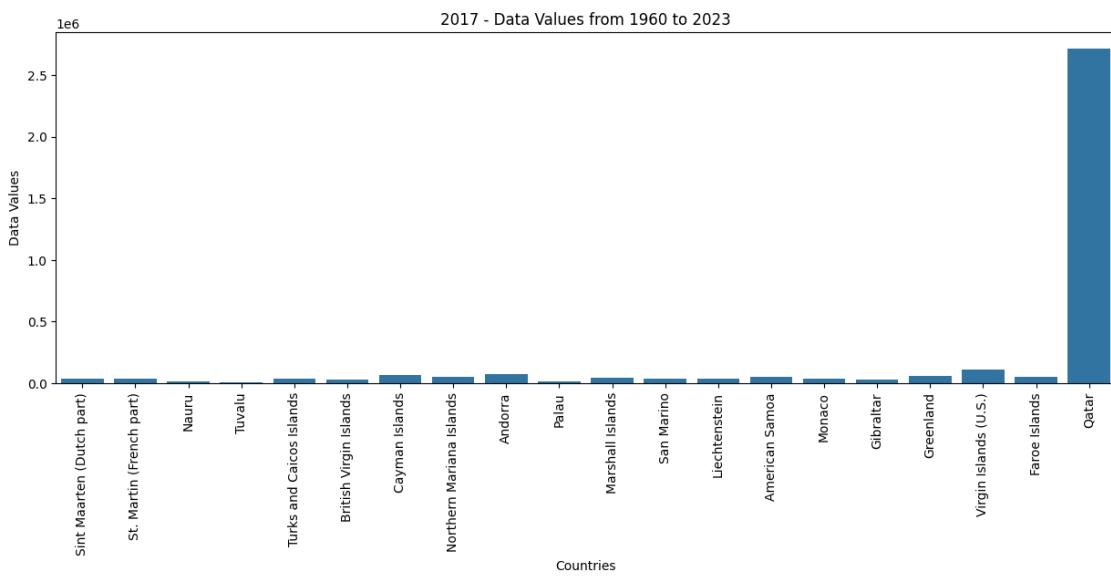
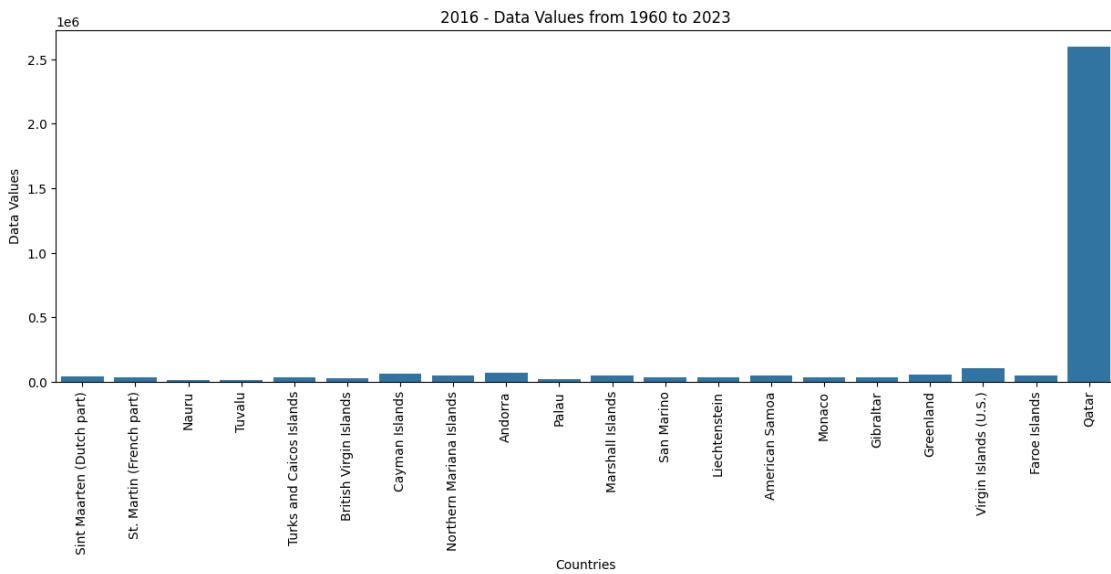


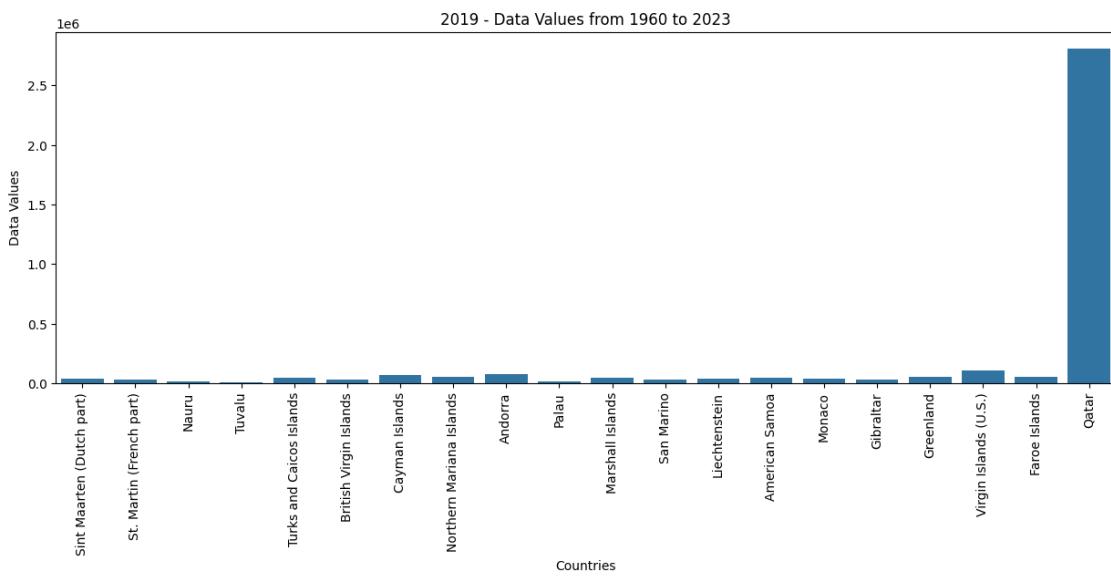
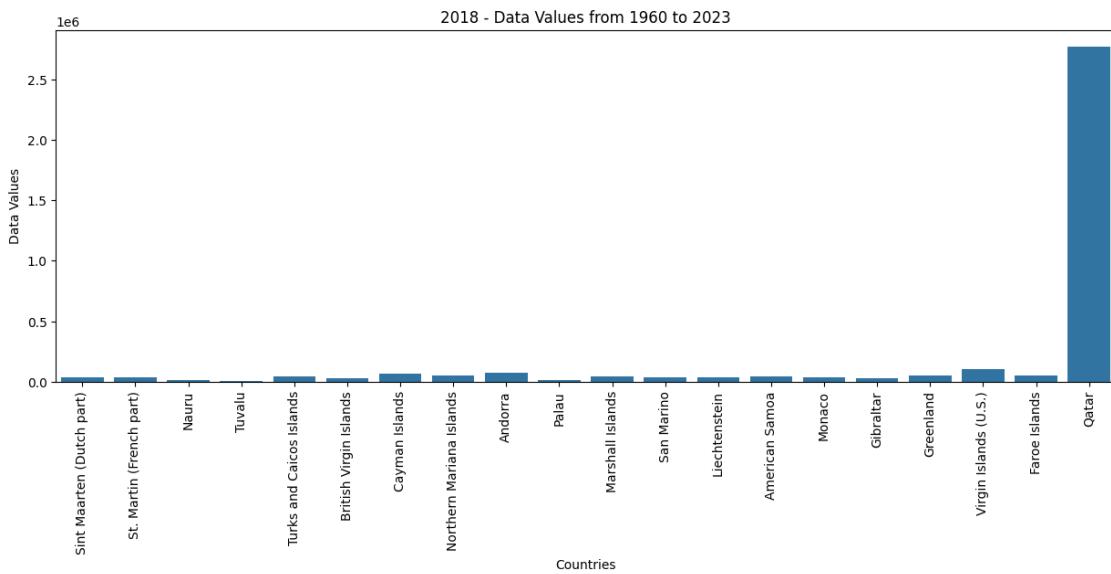


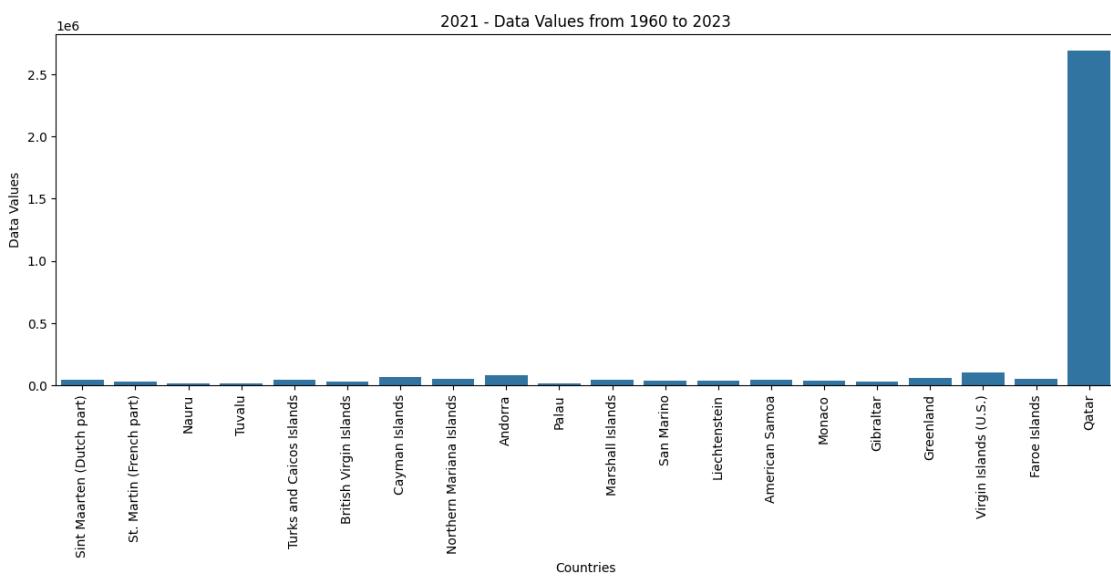
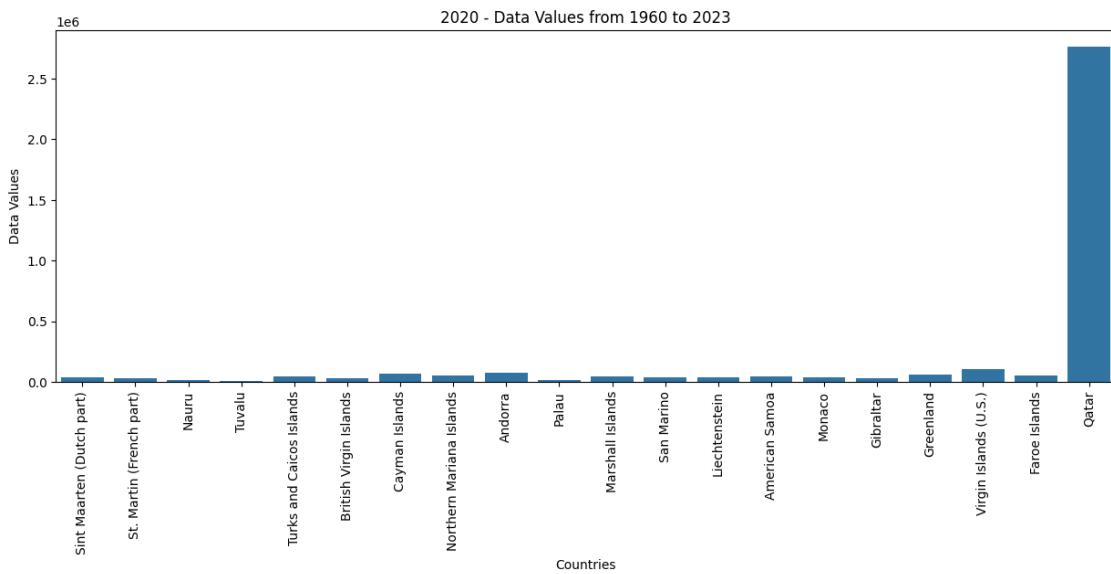


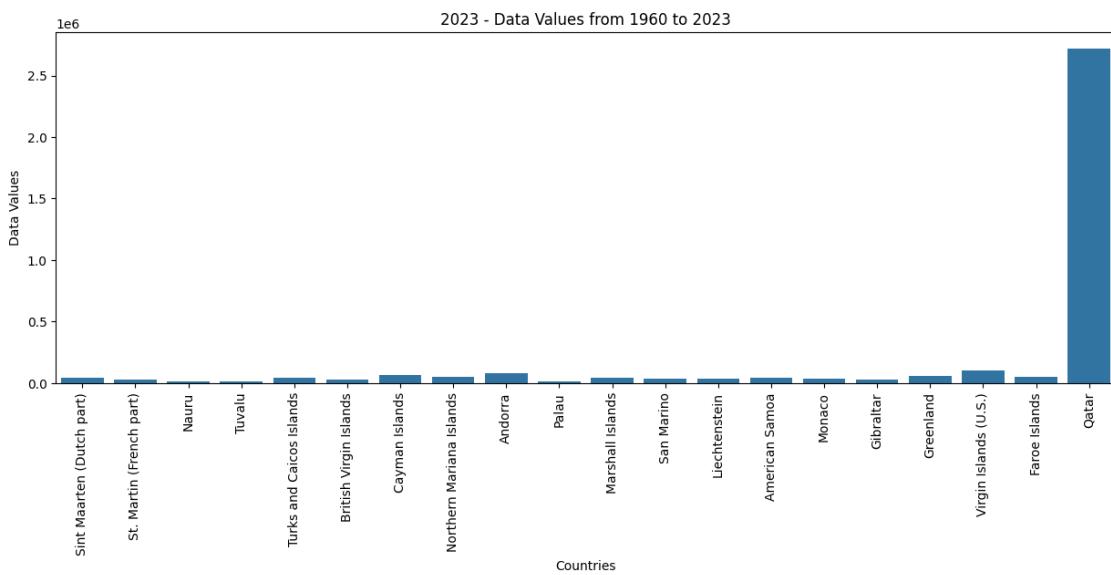
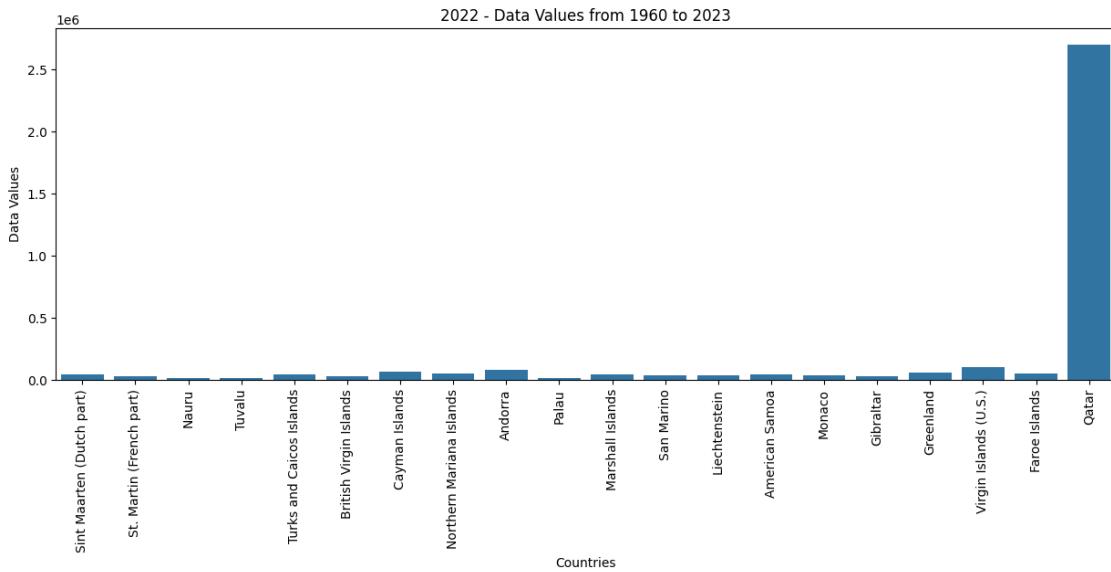






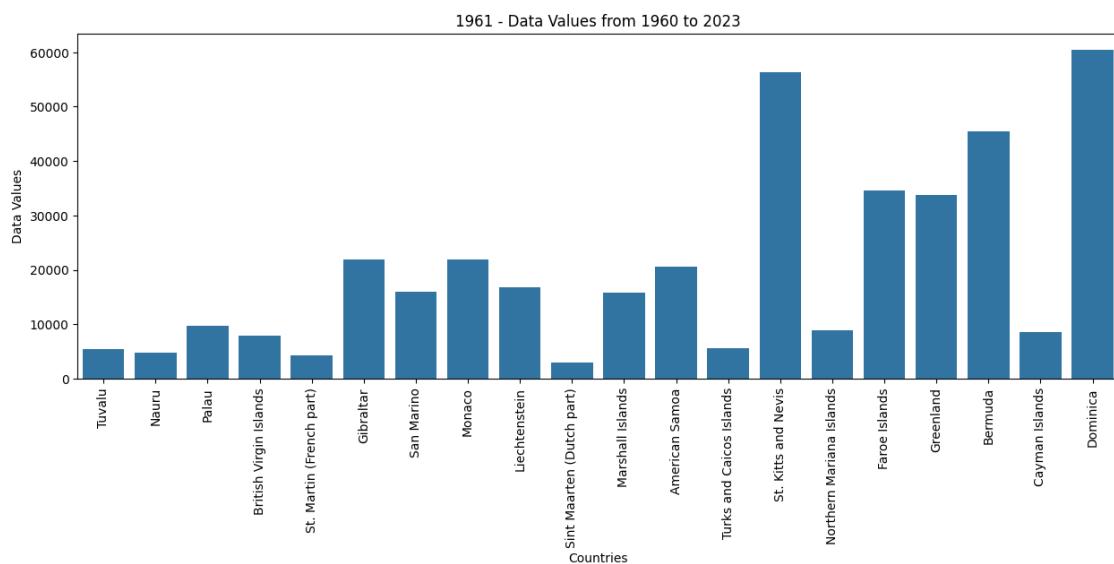
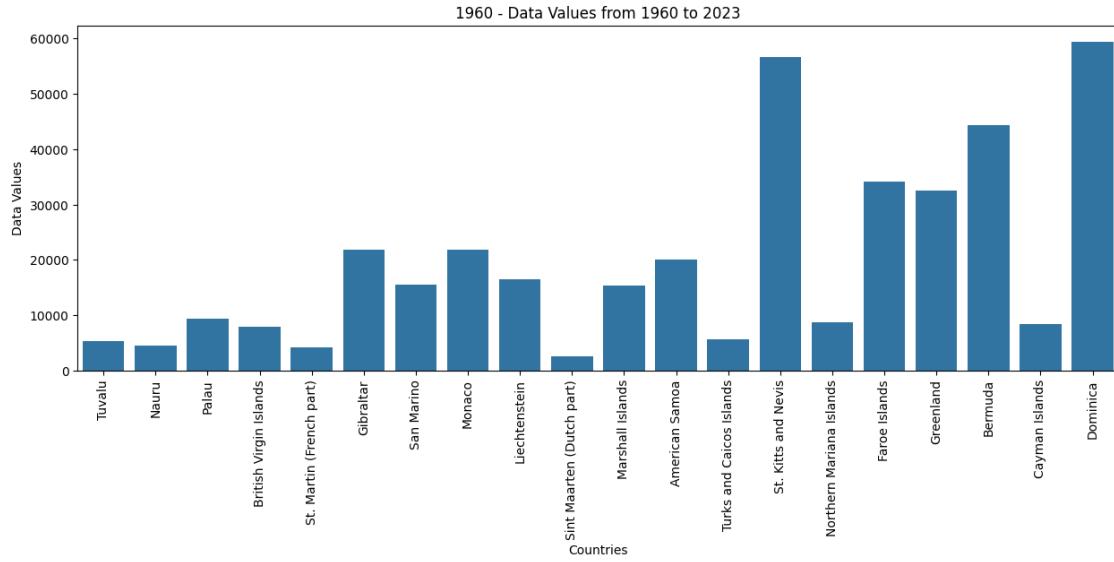


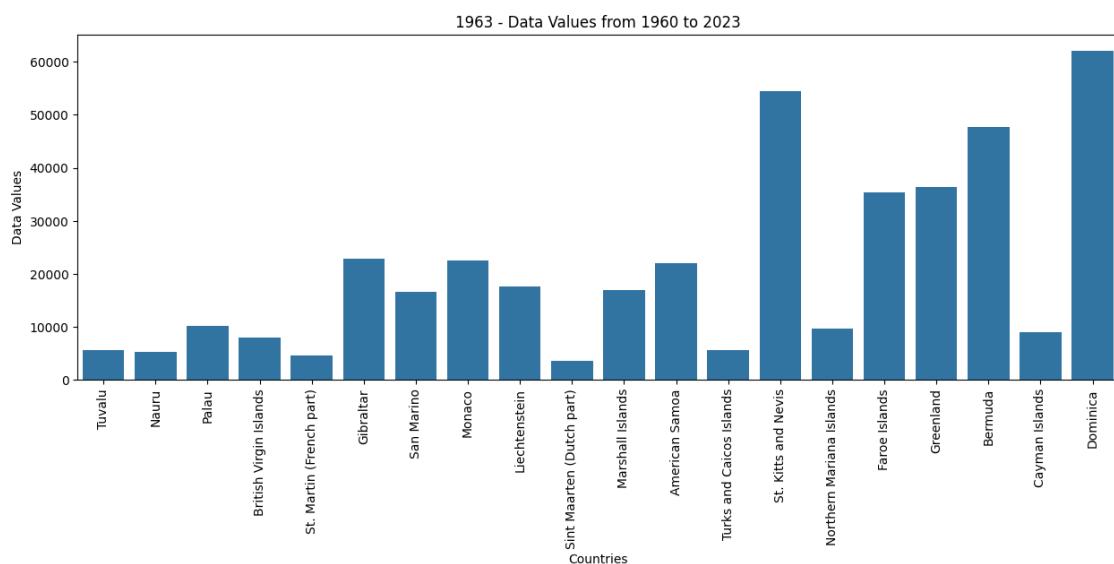
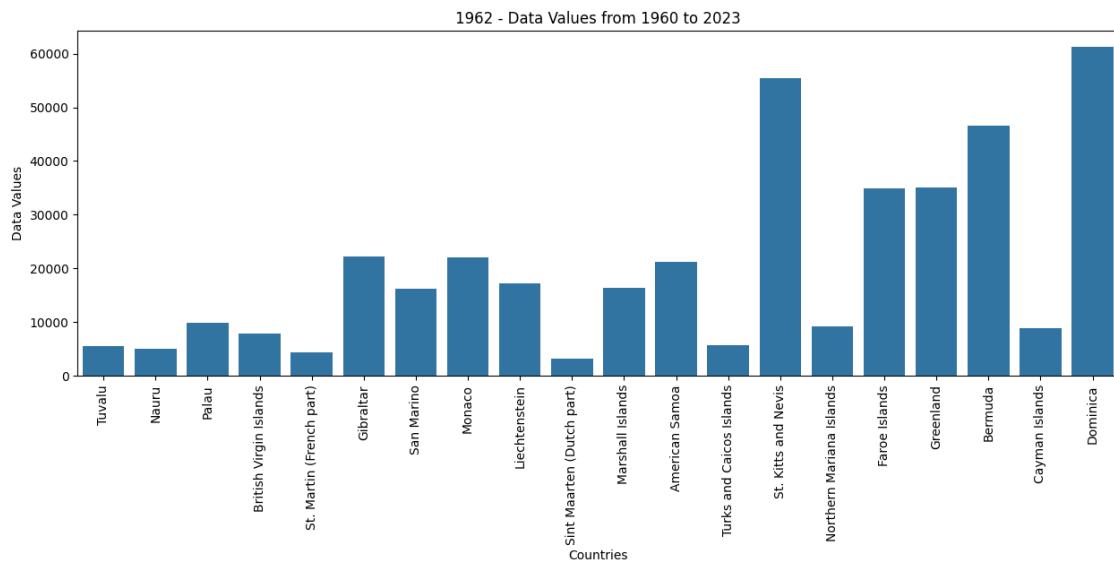




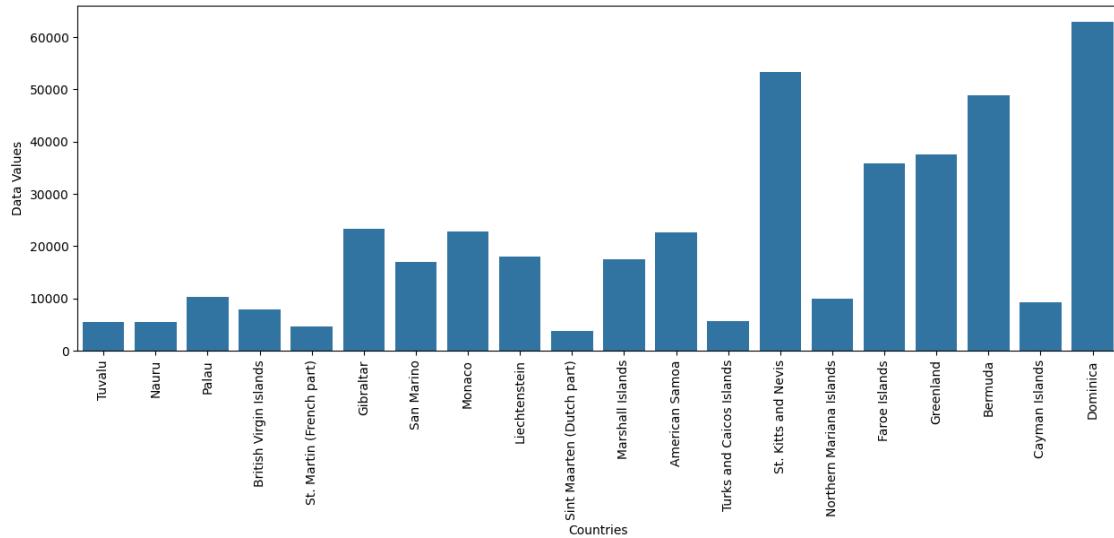
```
[ ]: country_by_2023=df.sort_values(by='2023').head(20)
country_by_2023_t=country_by_2023.set_index('Country Name').T
for country_name,data_values in country_by_2023_t.iterrows():
    fig=plt.figure(figsize=(15,5))
    sns.barplot(x=data_values.index,y=data_values.values)
    plt.xlabel('Countries')
    plt.ylabel('Data Values')
    plt.title(f'{country_name} - Data Values from 1960 to 2023')
    plt.xticks(rotation=90)
```

```
plt.show()
```

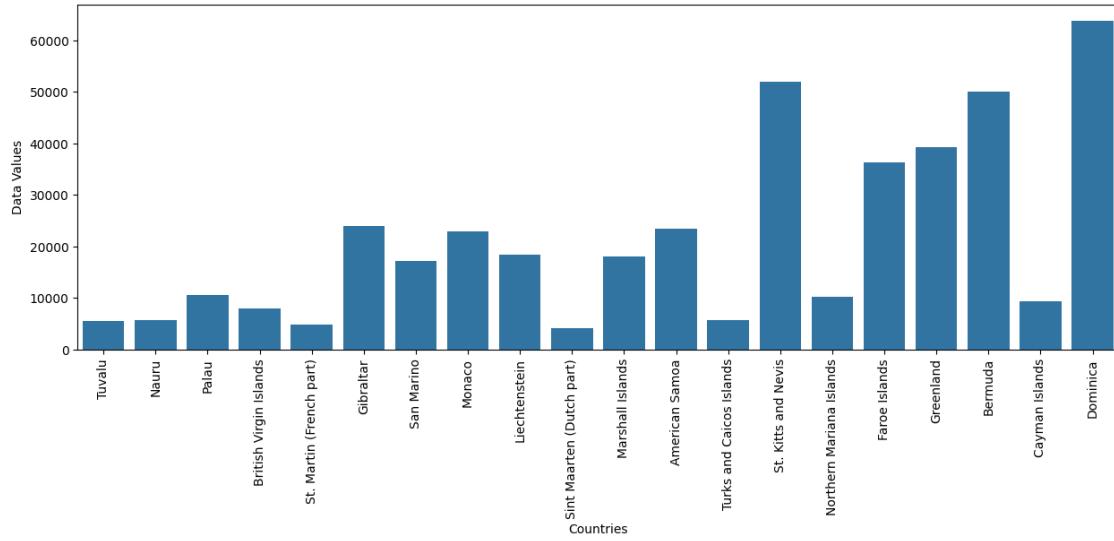




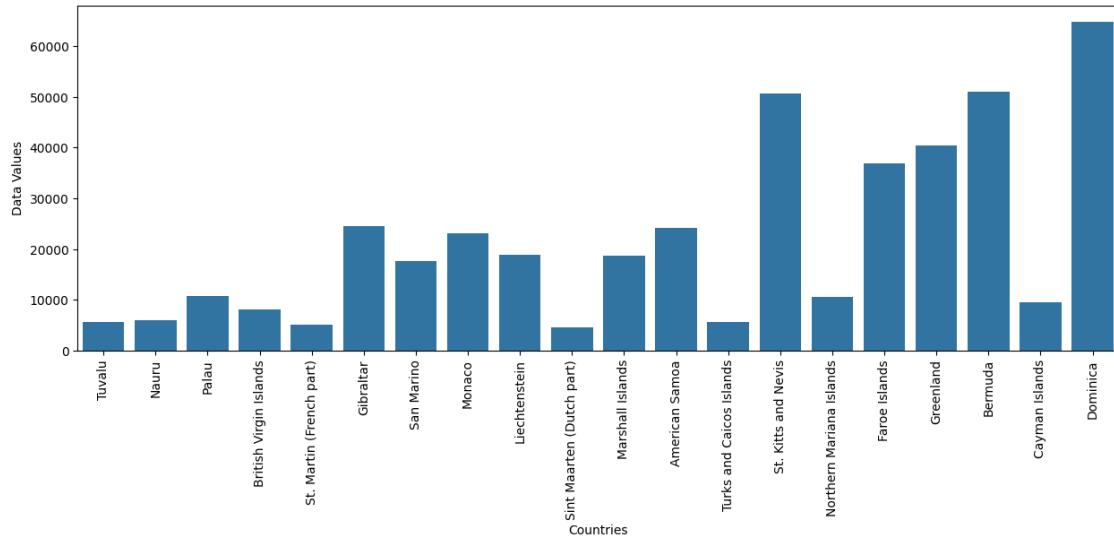
1964 - Data Values from 1960 to 2023



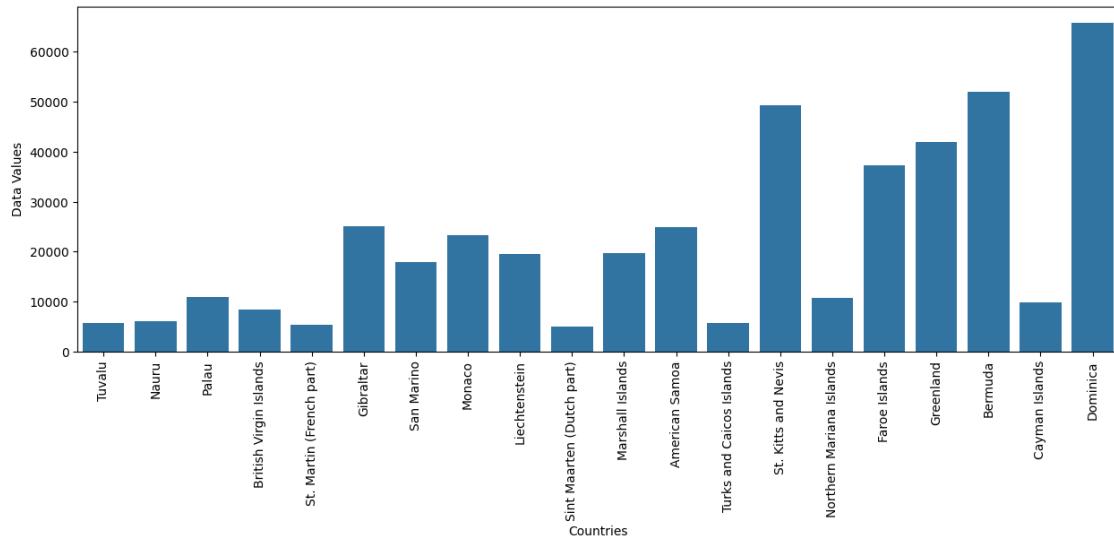
1965 - Data Values from 1960 to 2023

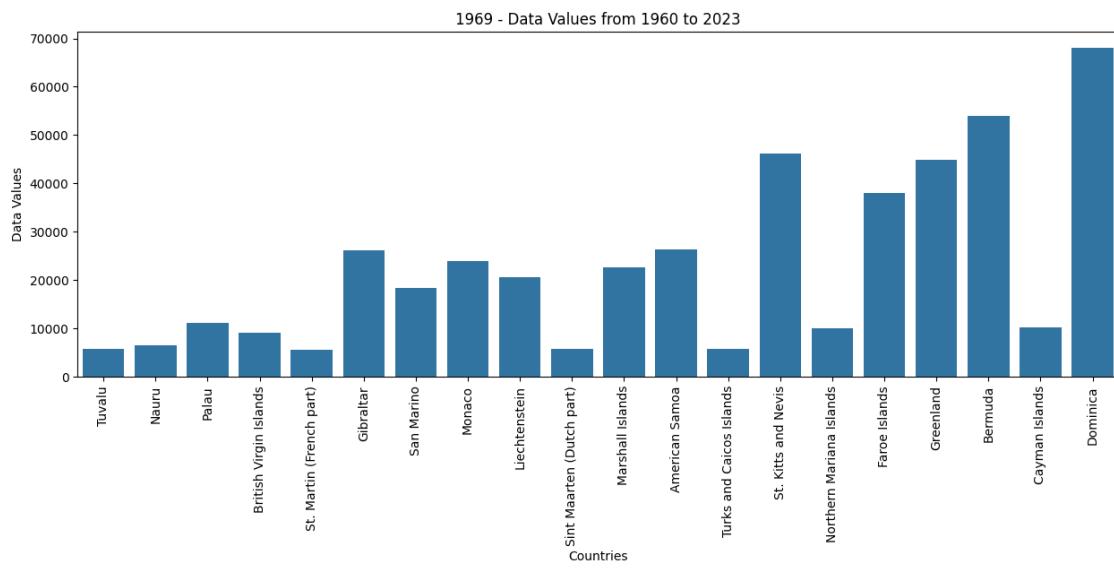
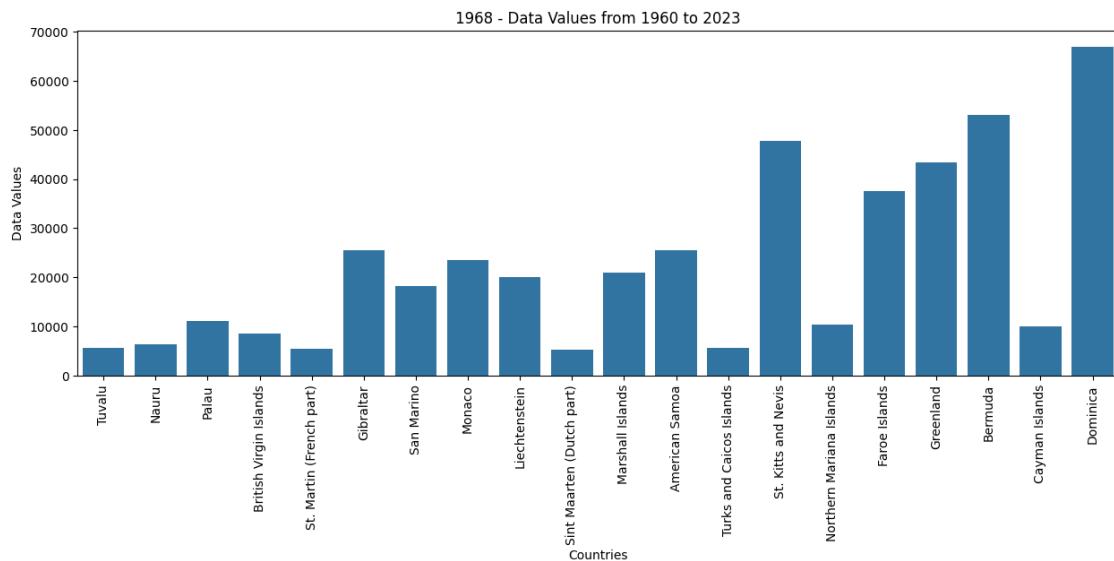


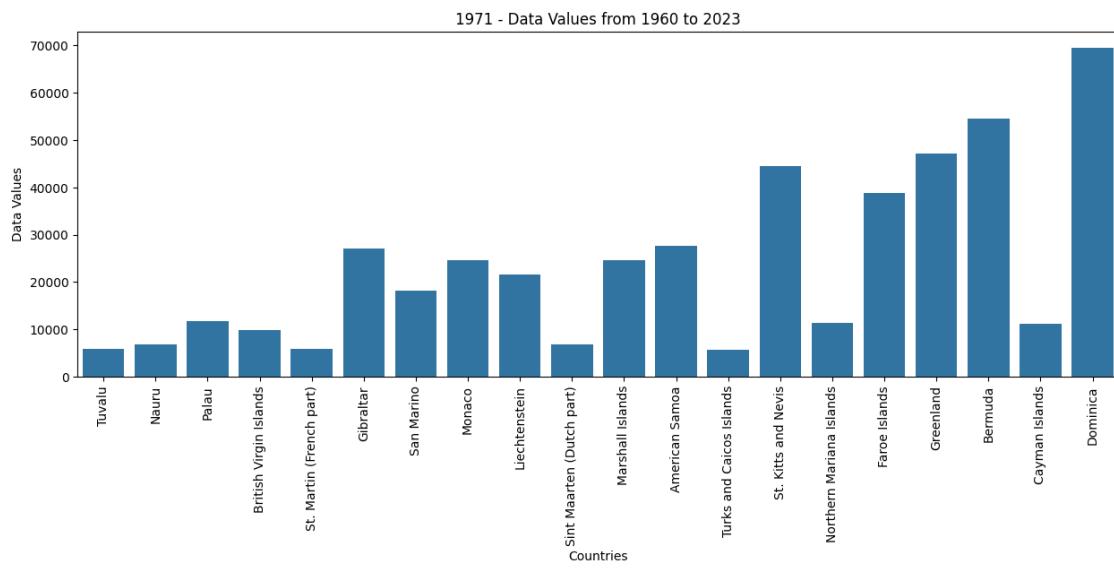
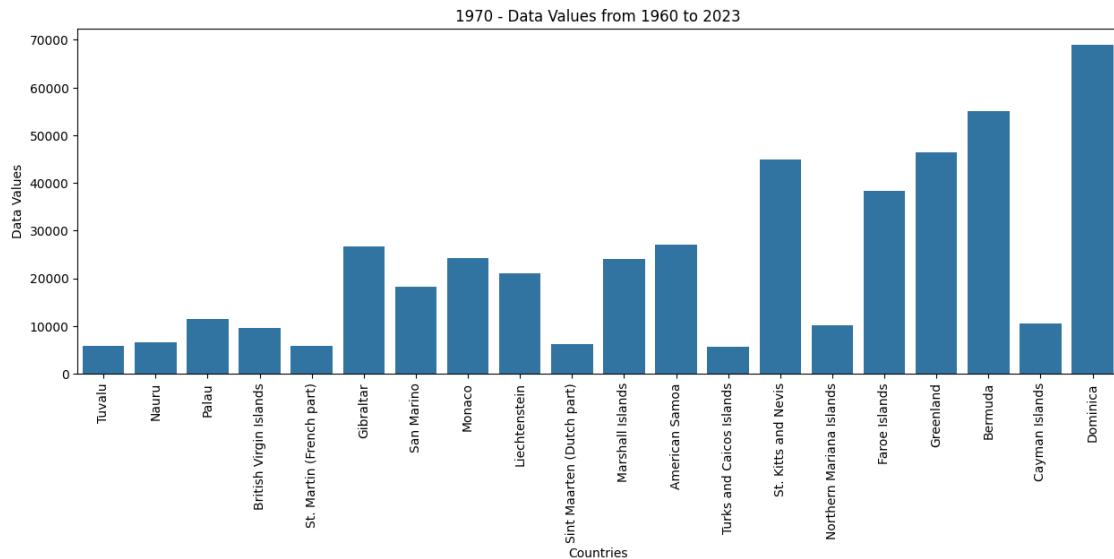
1966 - Data Values from 1960 to 2023



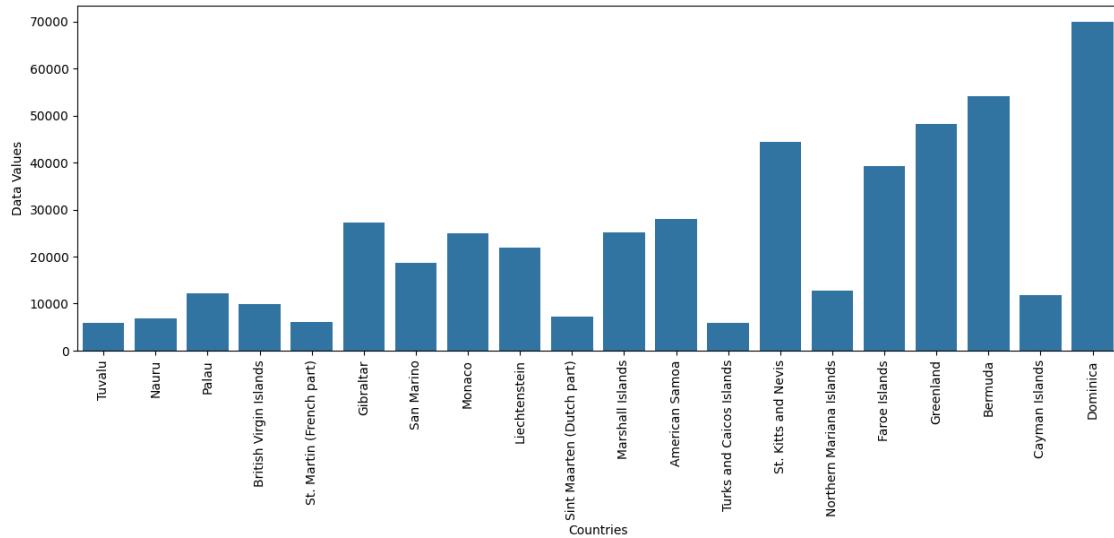
1967 - Data Values from 1960 to 2023



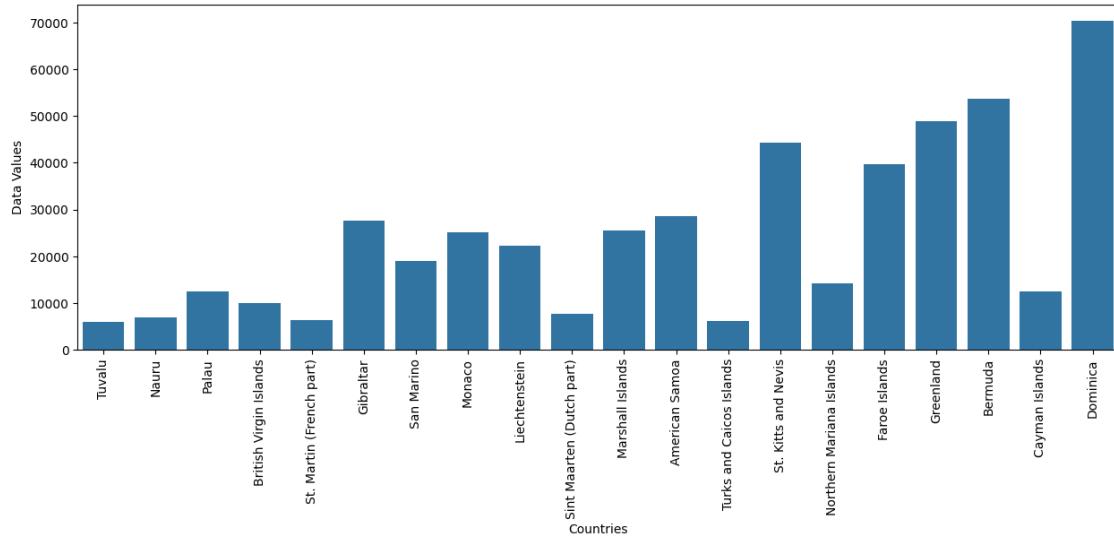




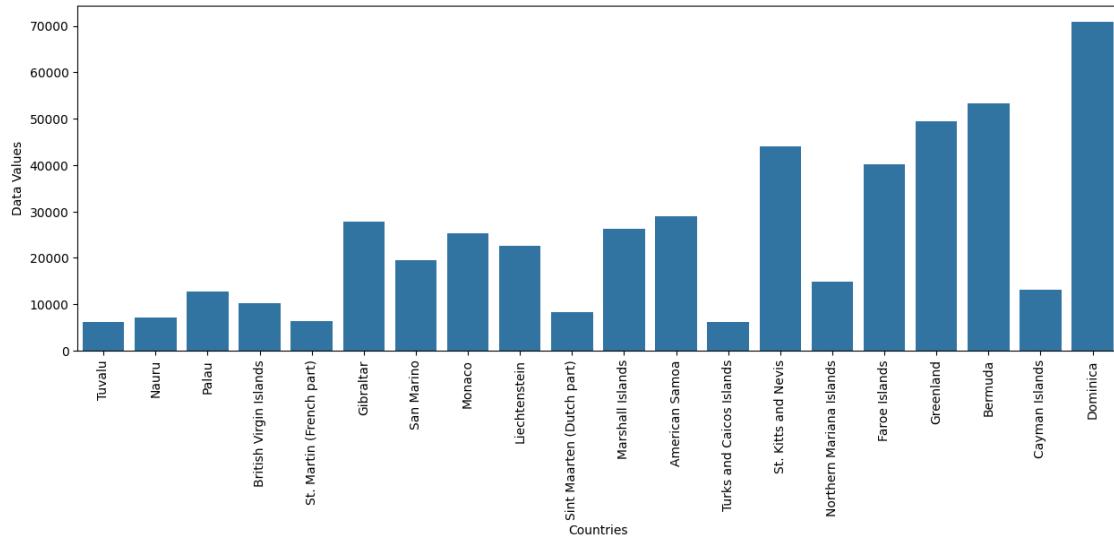
1972 - Data Values from 1960 to 2023



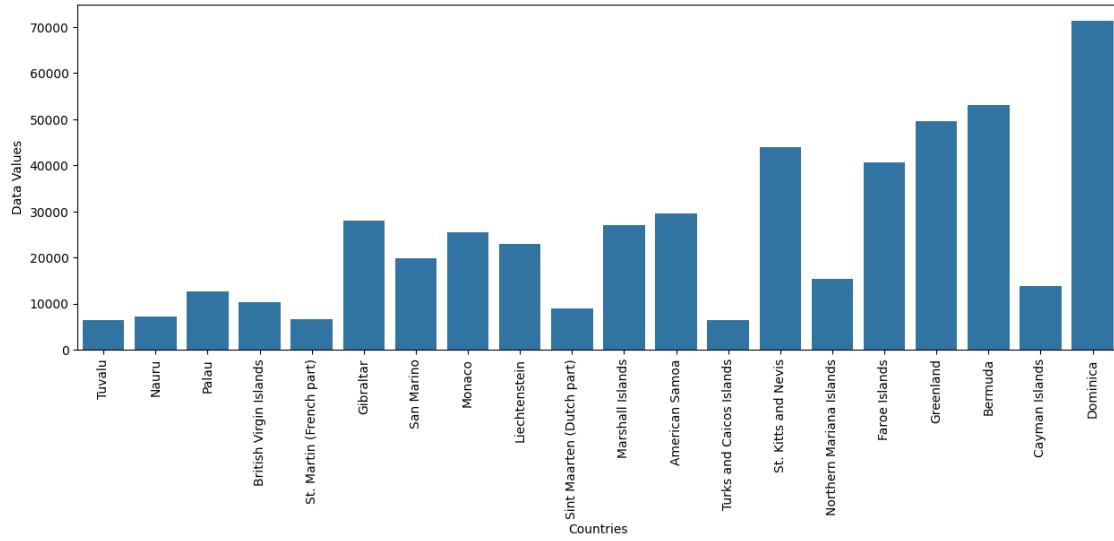
1973 - Data Values from 1960 to 2023



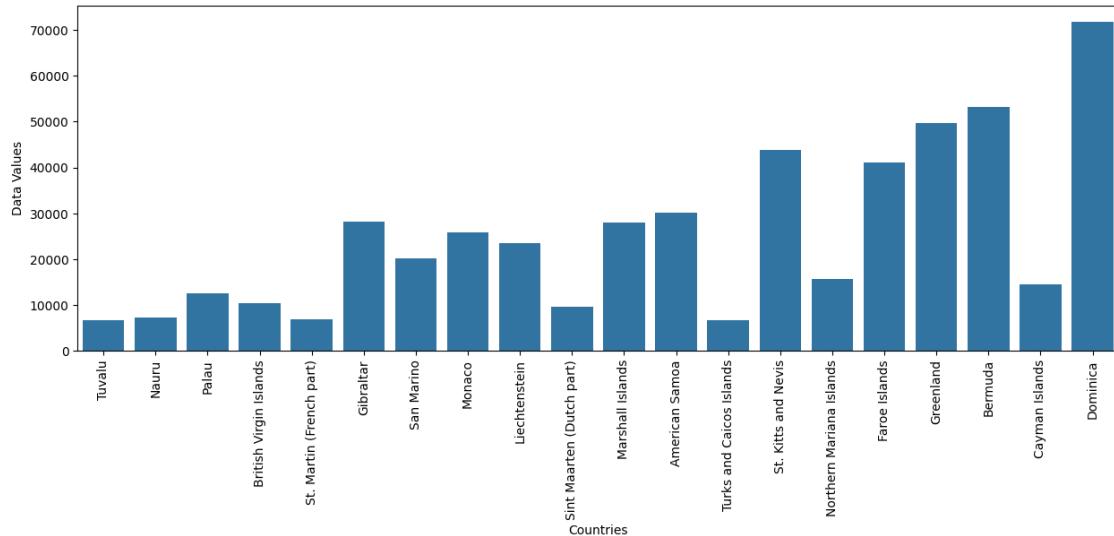
1974 - Data Values from 1960 to 2023



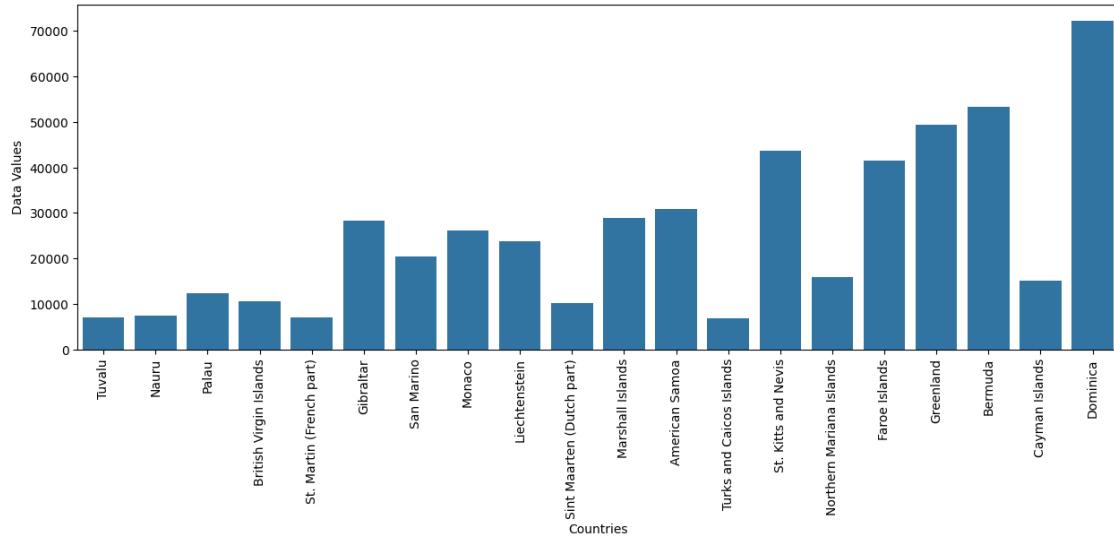
1975 - Data Values from 1960 to 2023

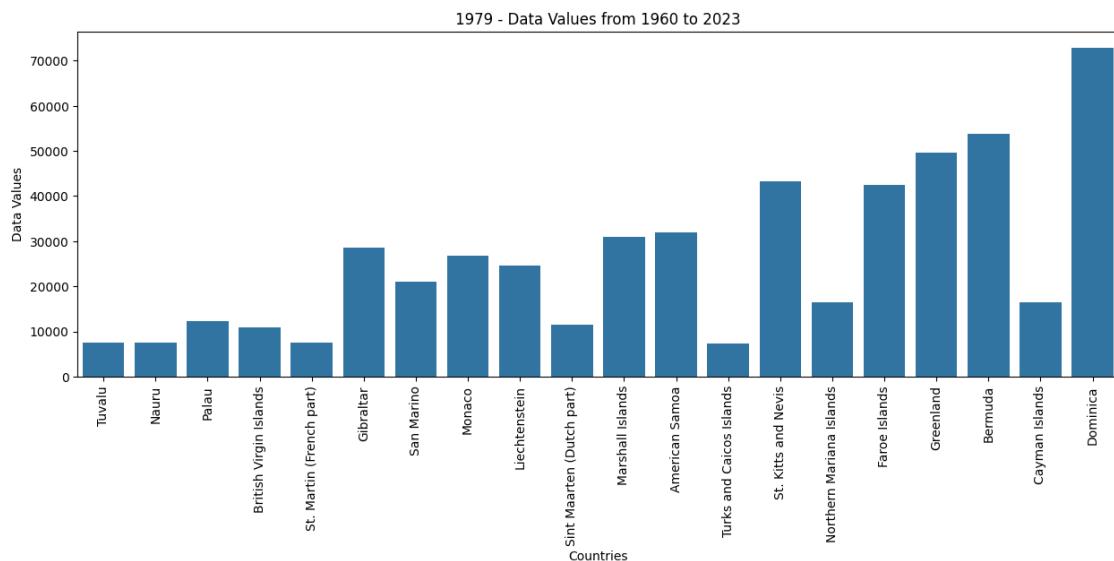
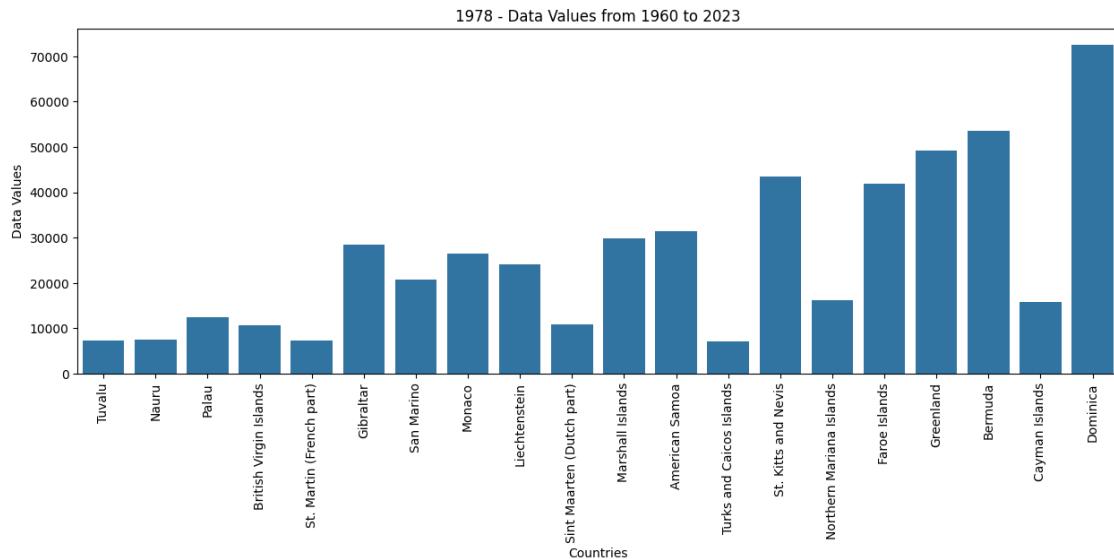


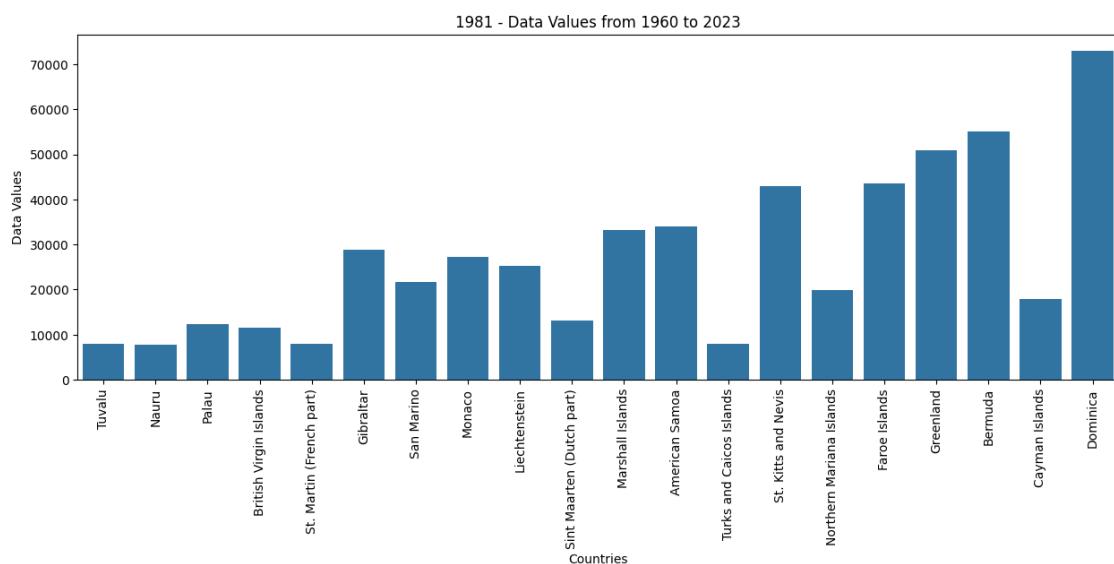
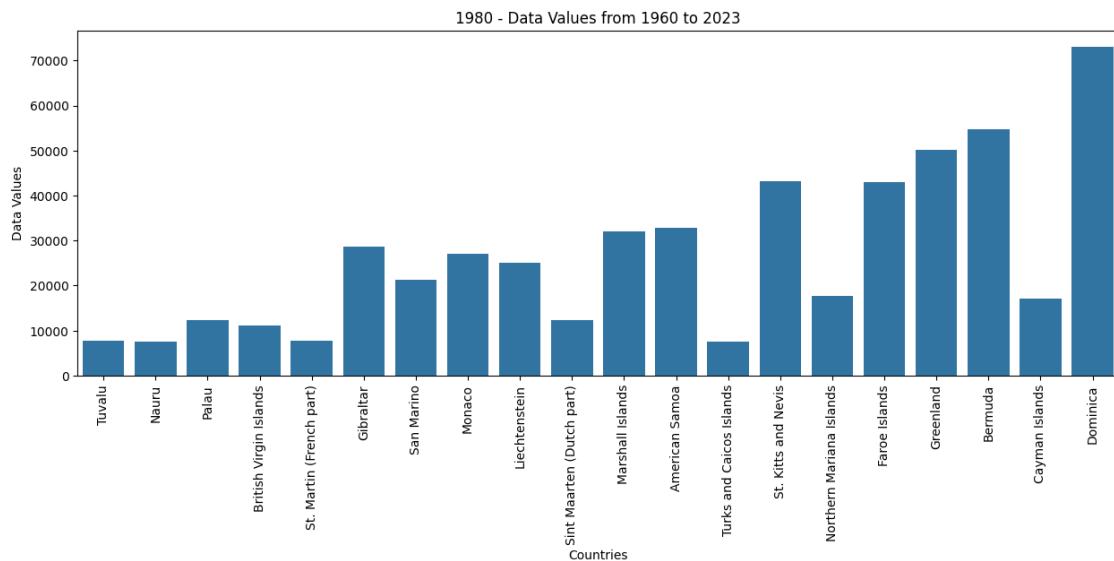
1976 - Data Values from 1960 to 2023

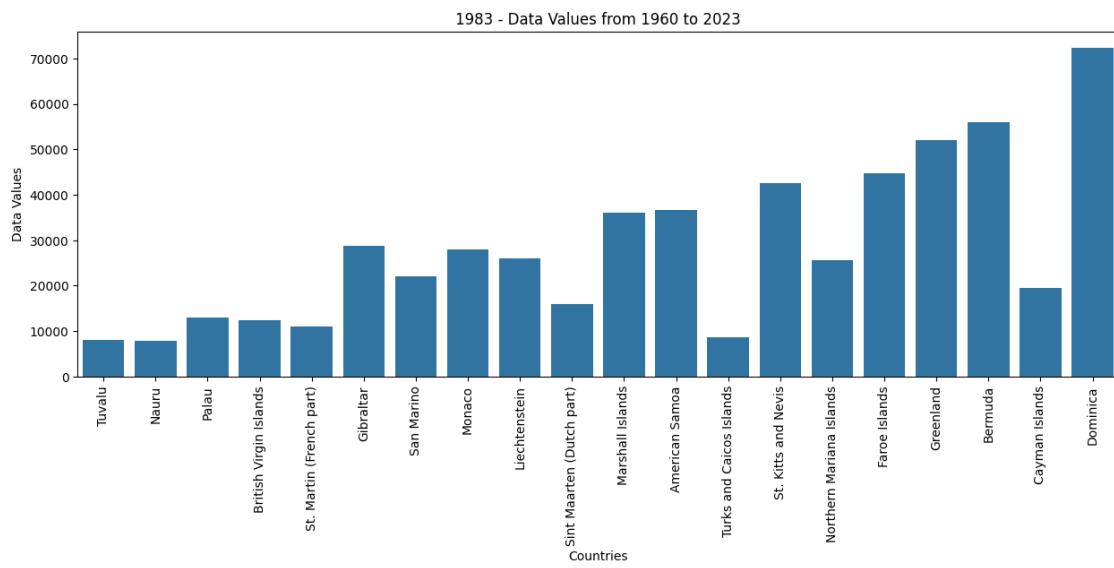
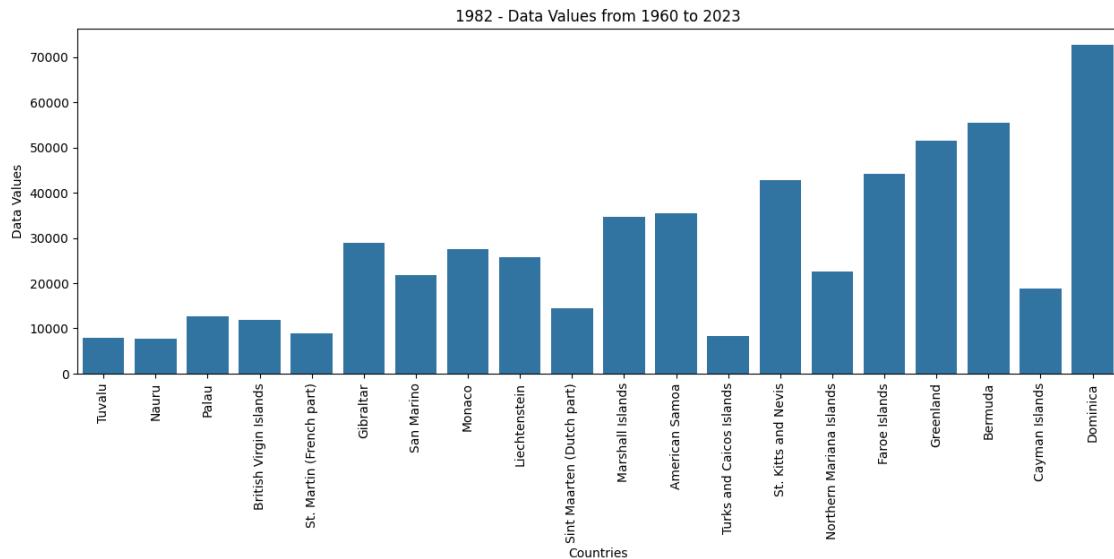


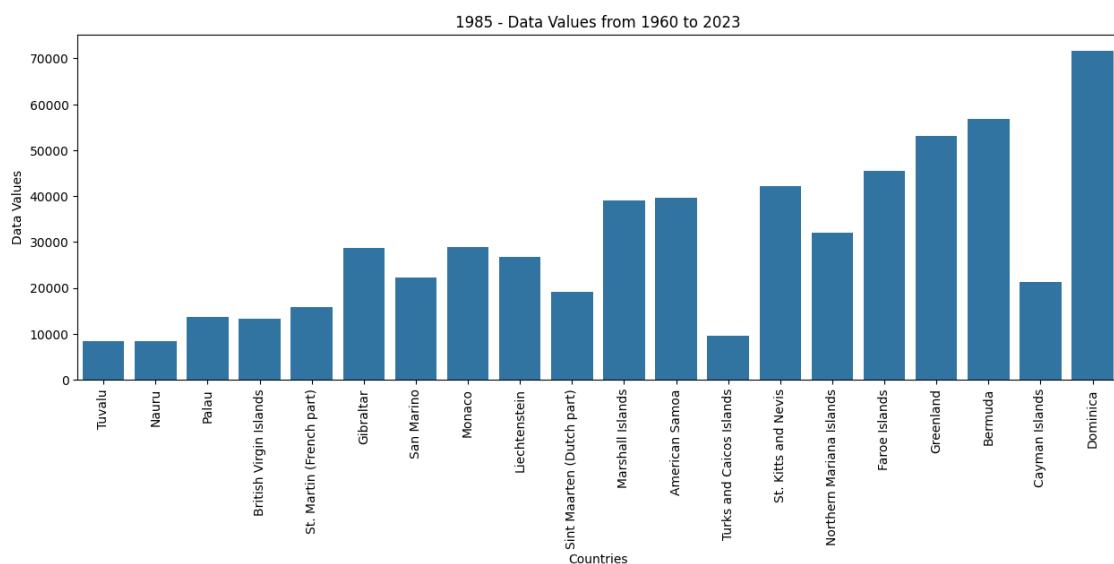
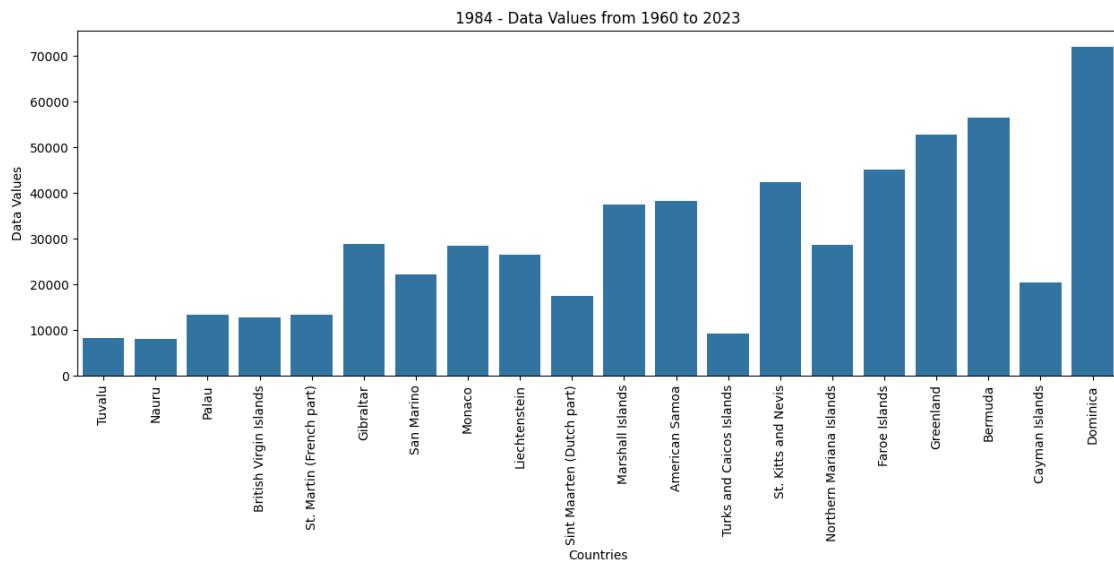
1977 - Data Values from 1960 to 2023



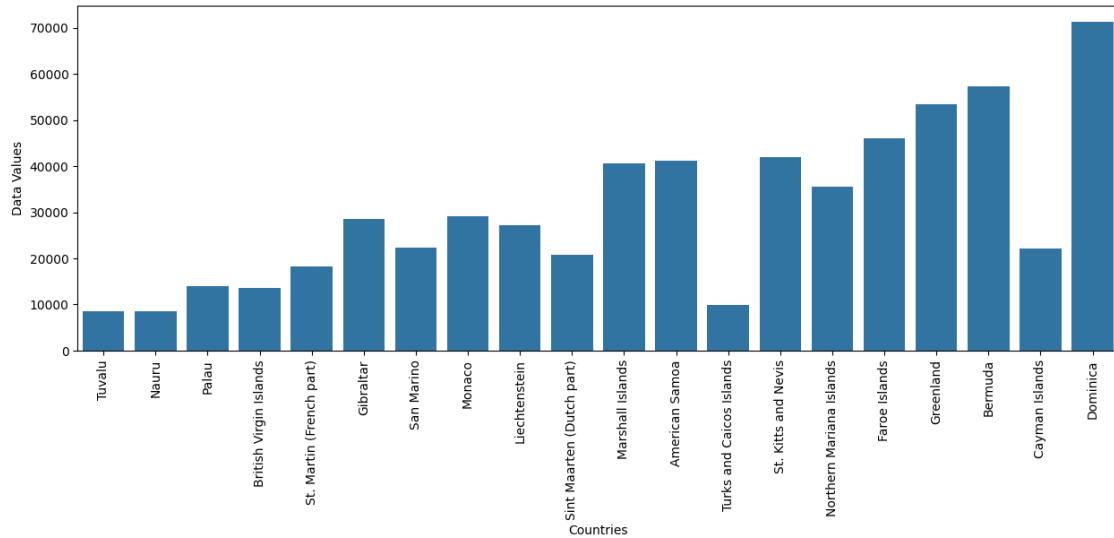




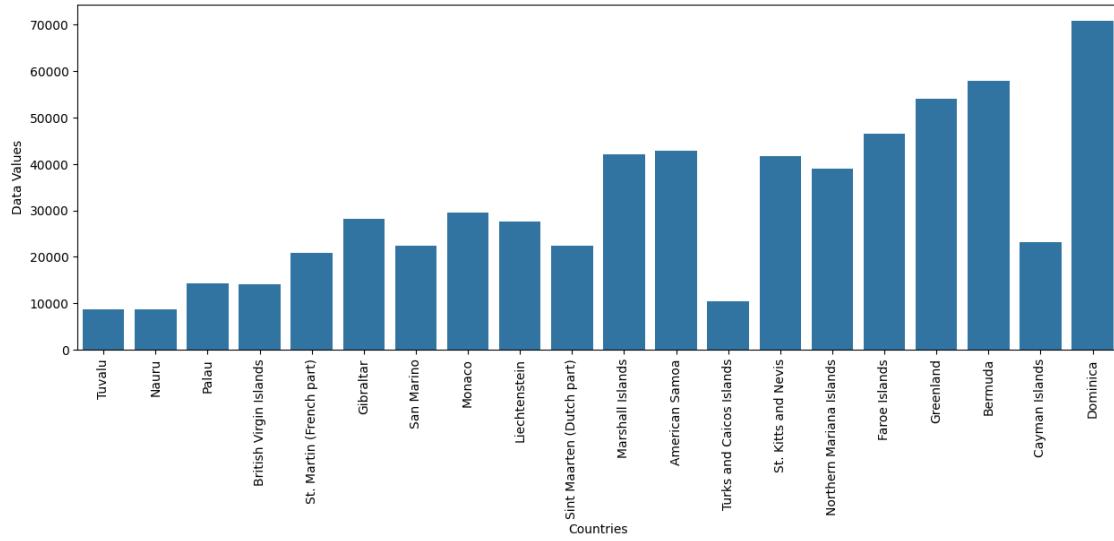




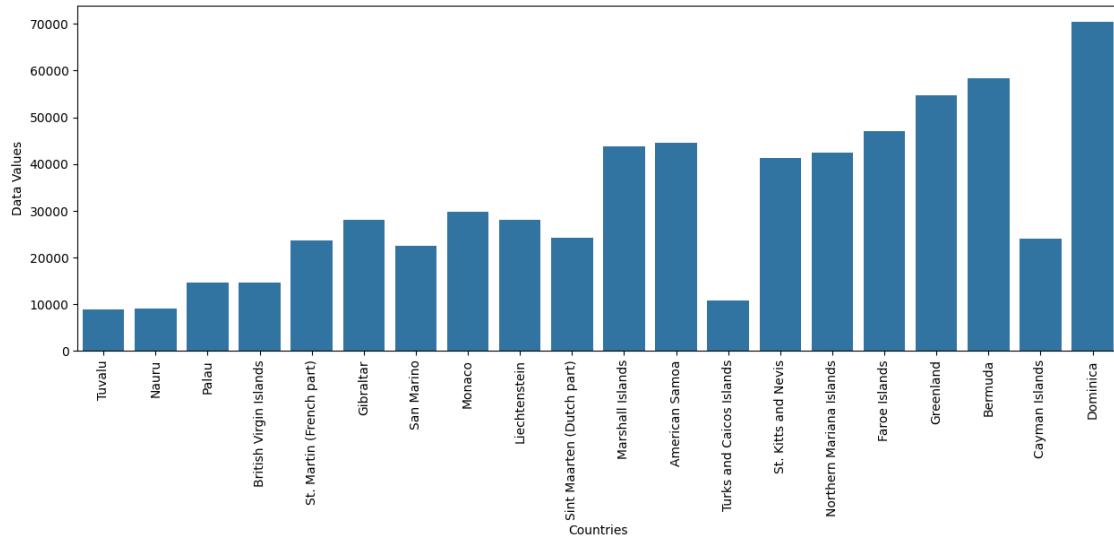
1986 - Data Values from 1960 to 2023



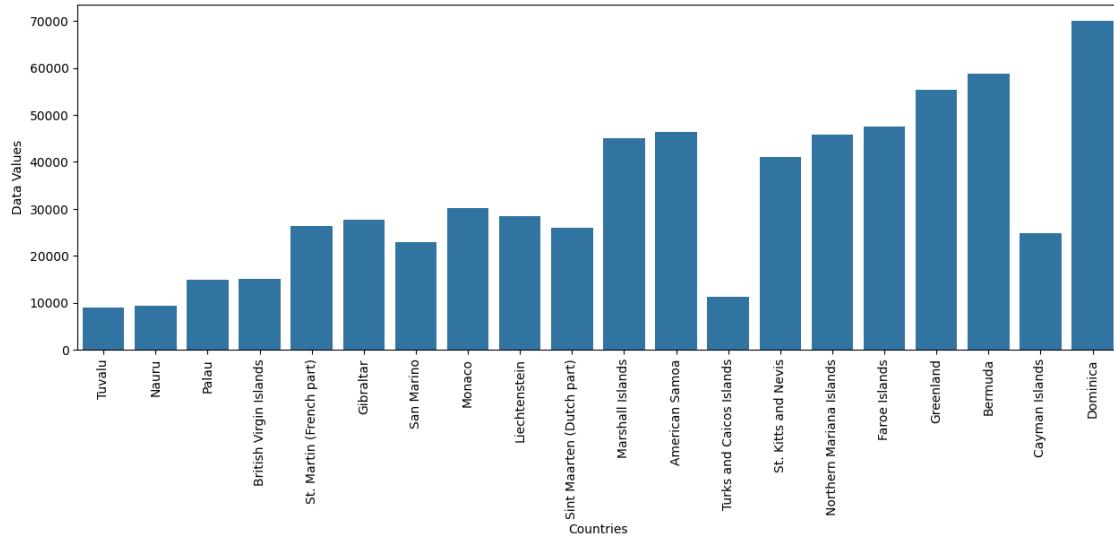
1987 - Data Values from 1960 to 2023

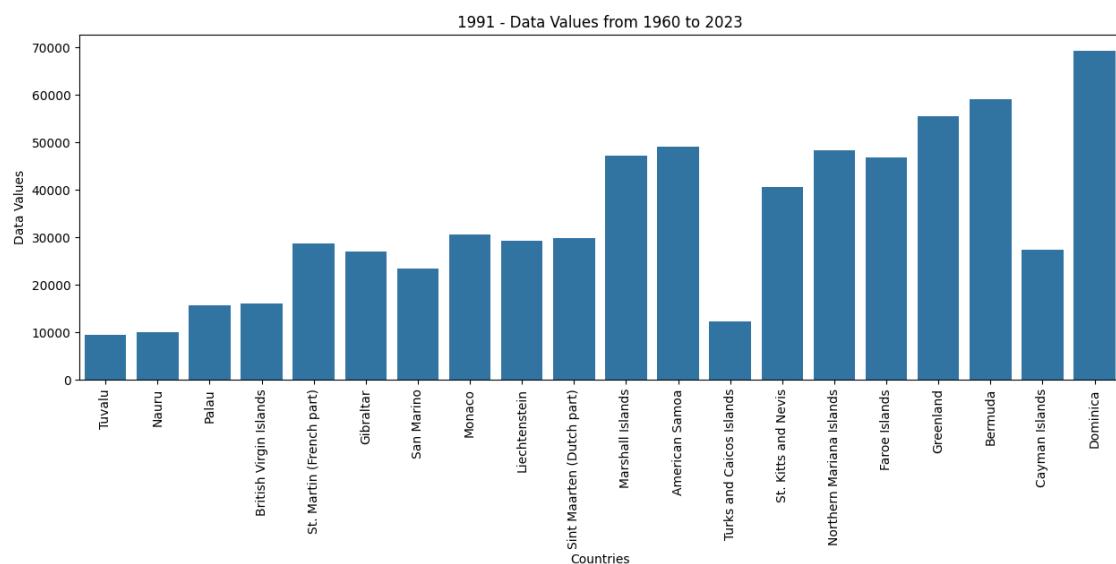
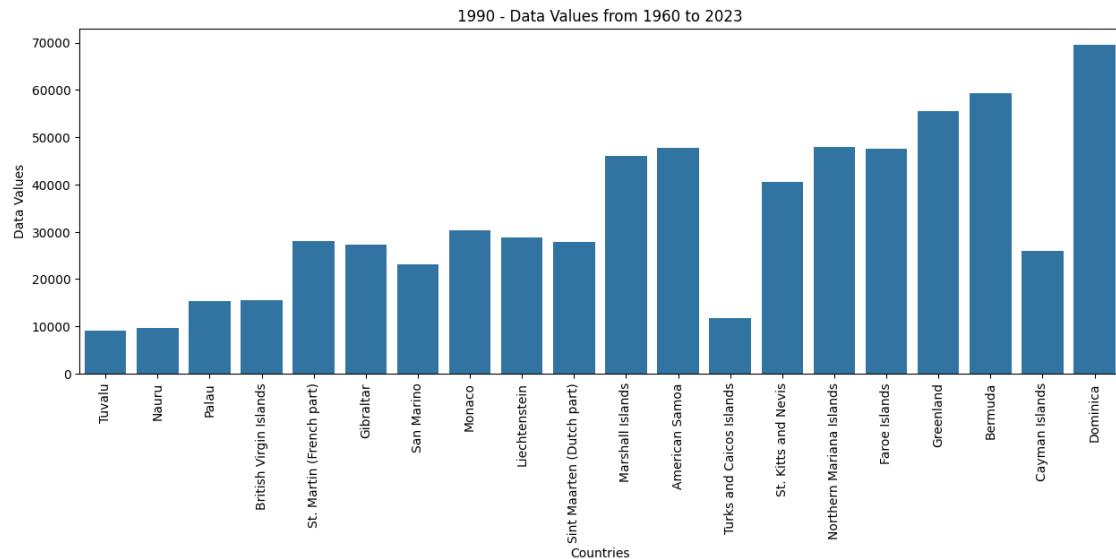


1988 - Data Values from 1960 to 2023

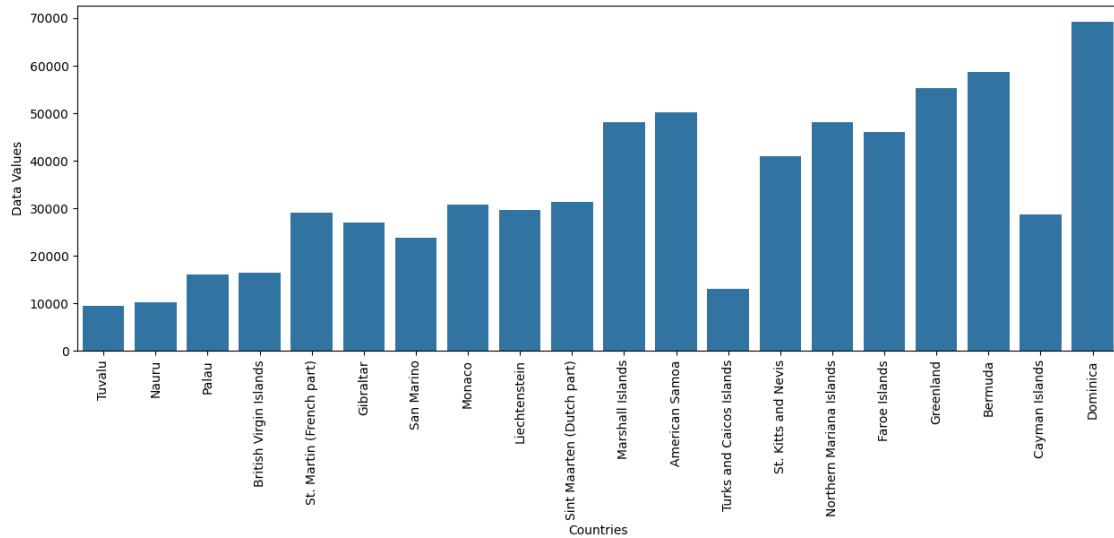


1989 - Data Values from 1960 to 2023

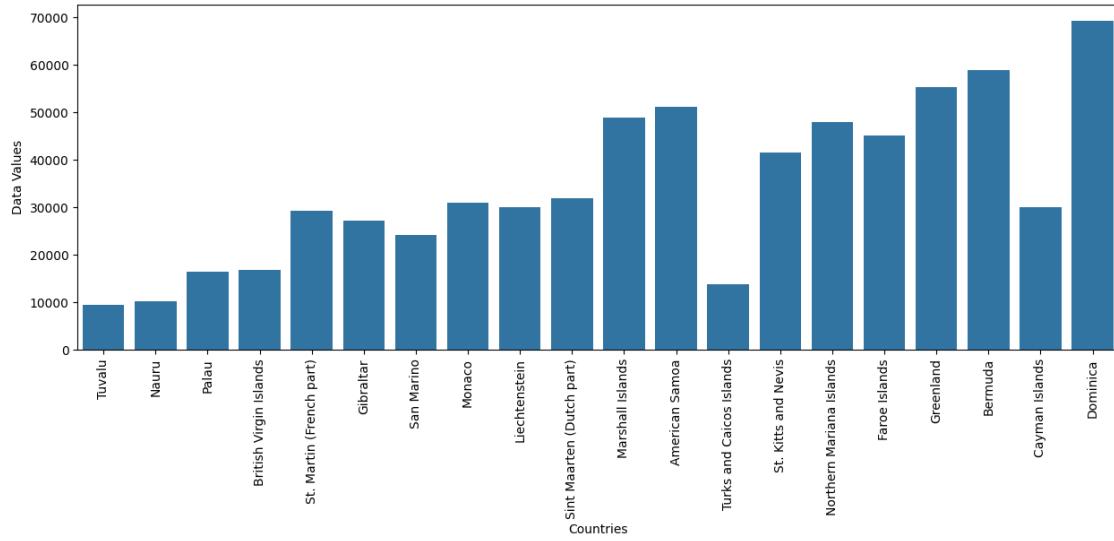




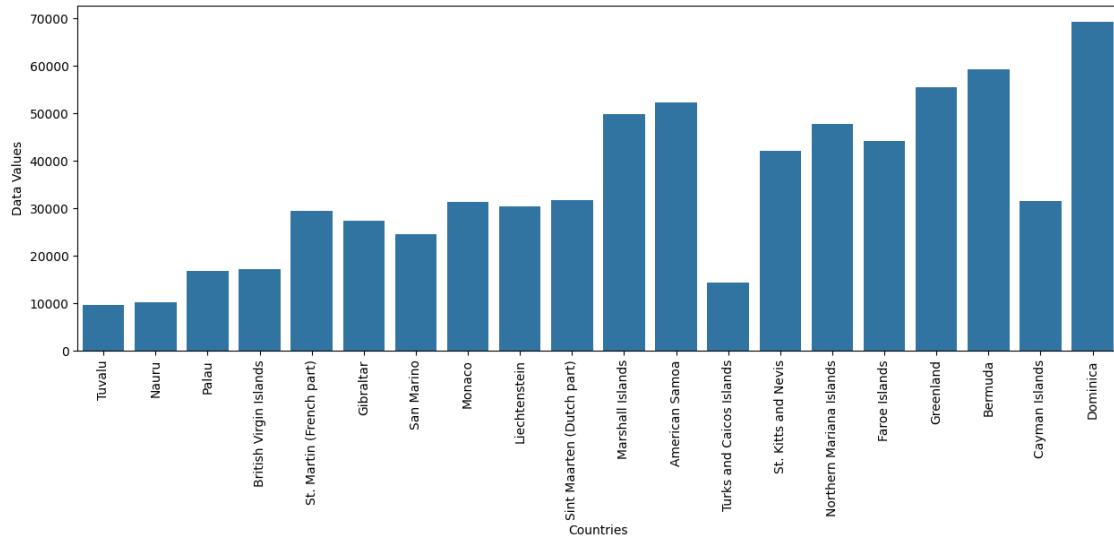
1992 - Data Values from 1960 to 2023



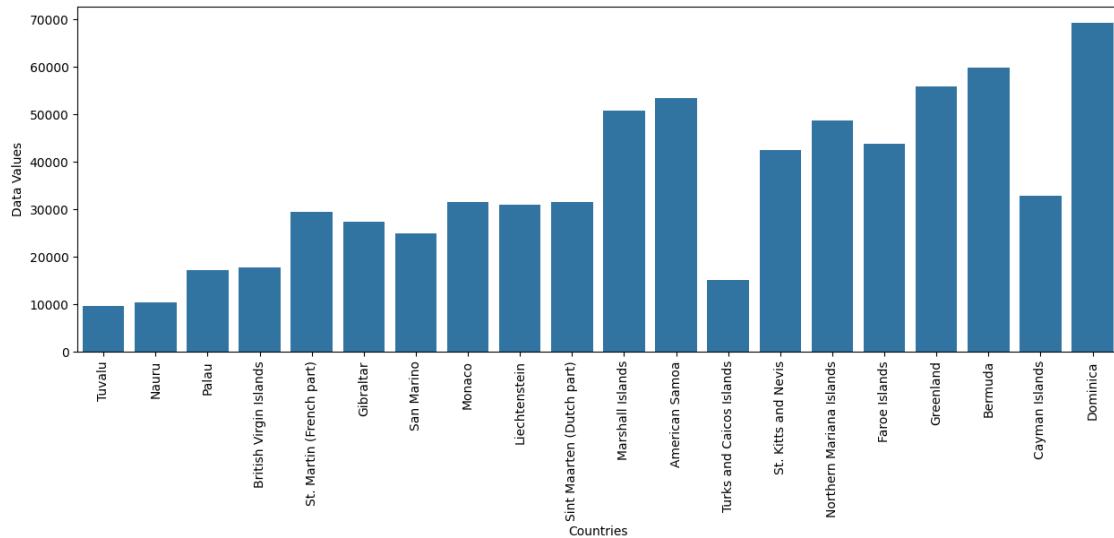
1993 - Data Values from 1960 to 2023



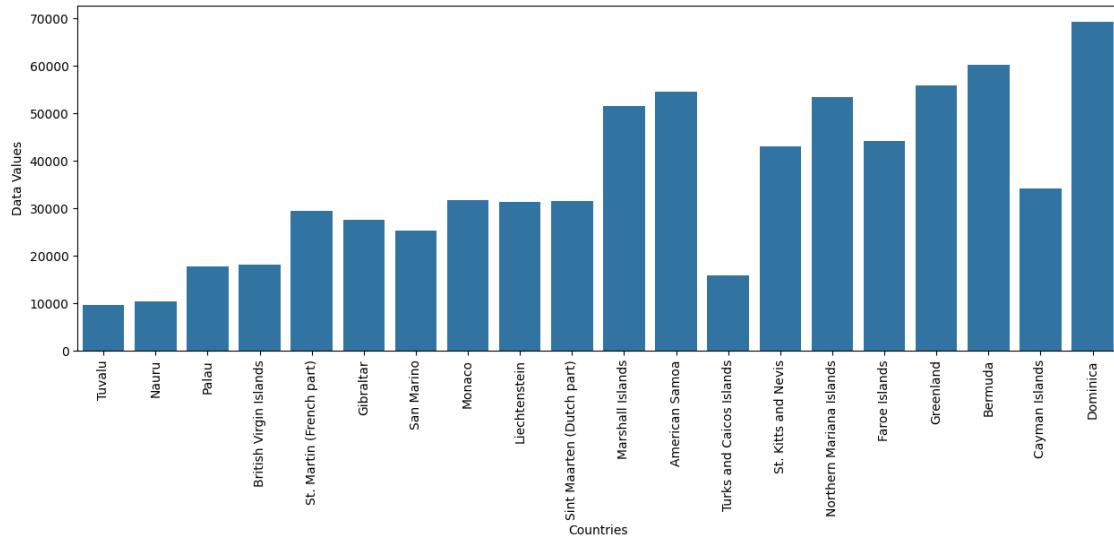
1994 - Data Values from 1960 to 2023



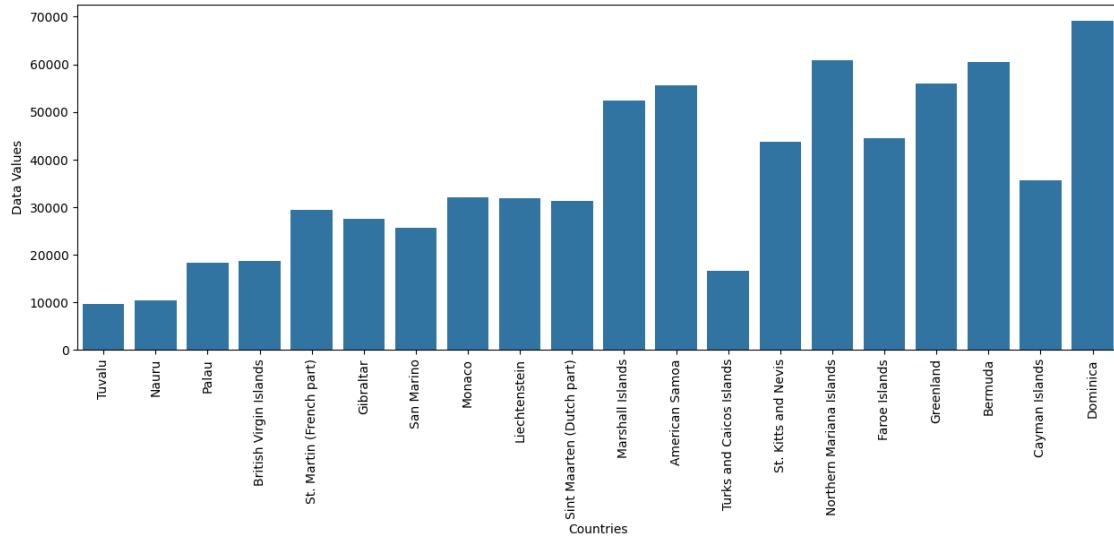
1995 - Data Values from 1960 to 2023

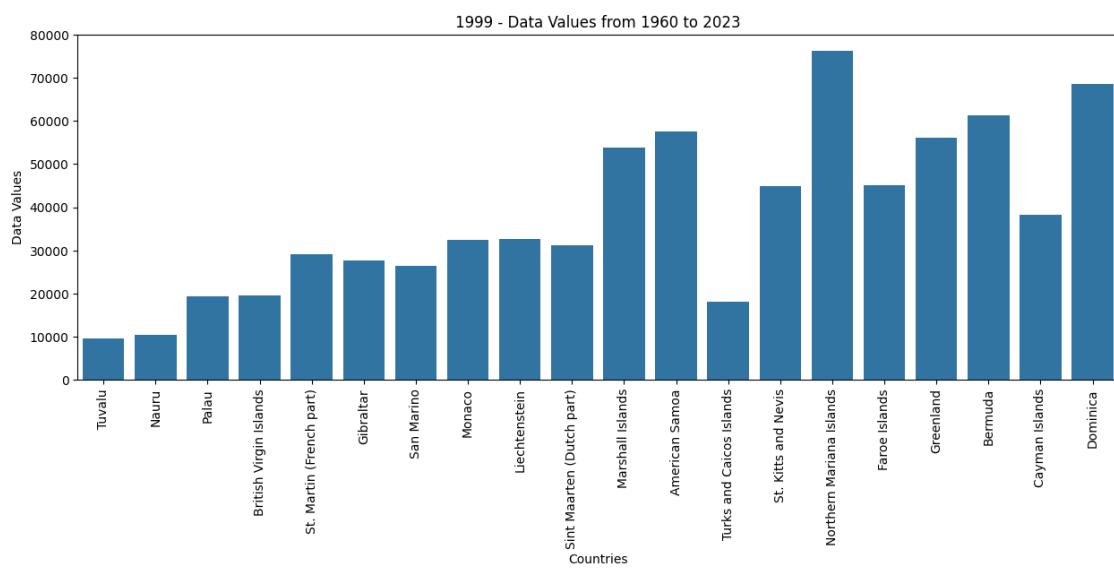
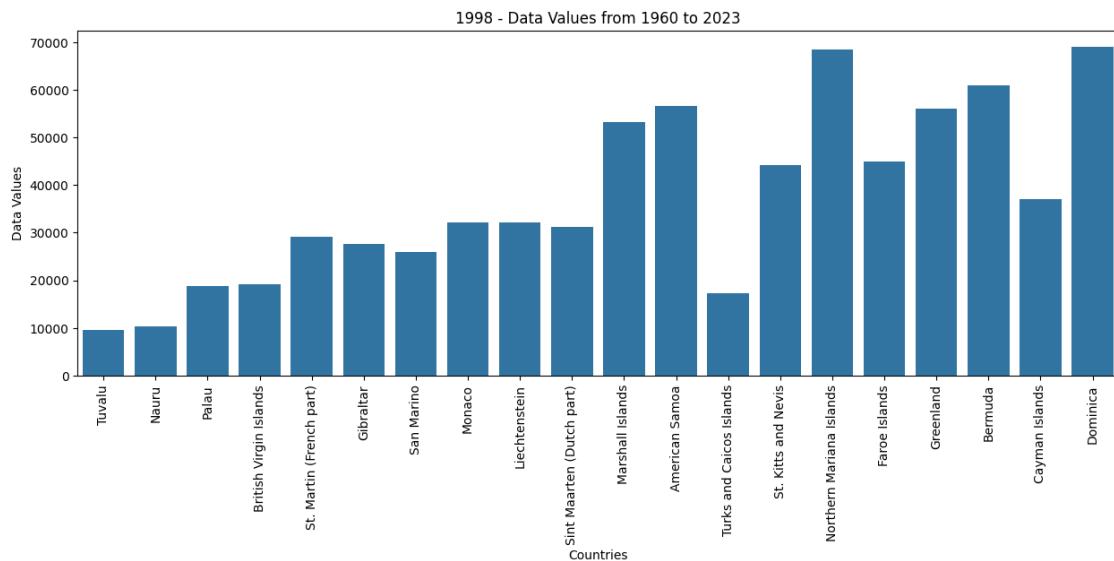


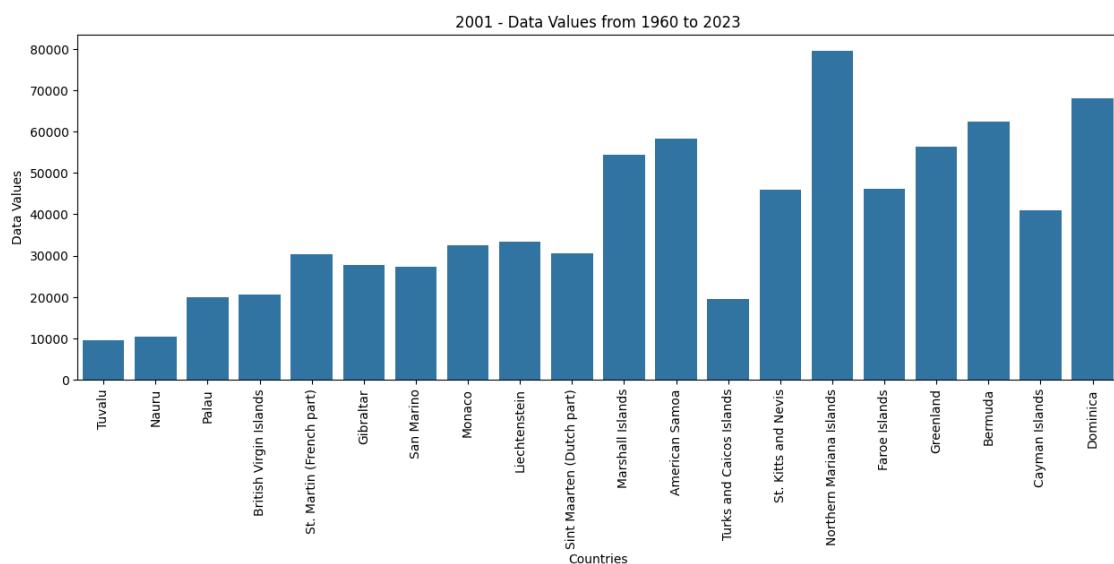
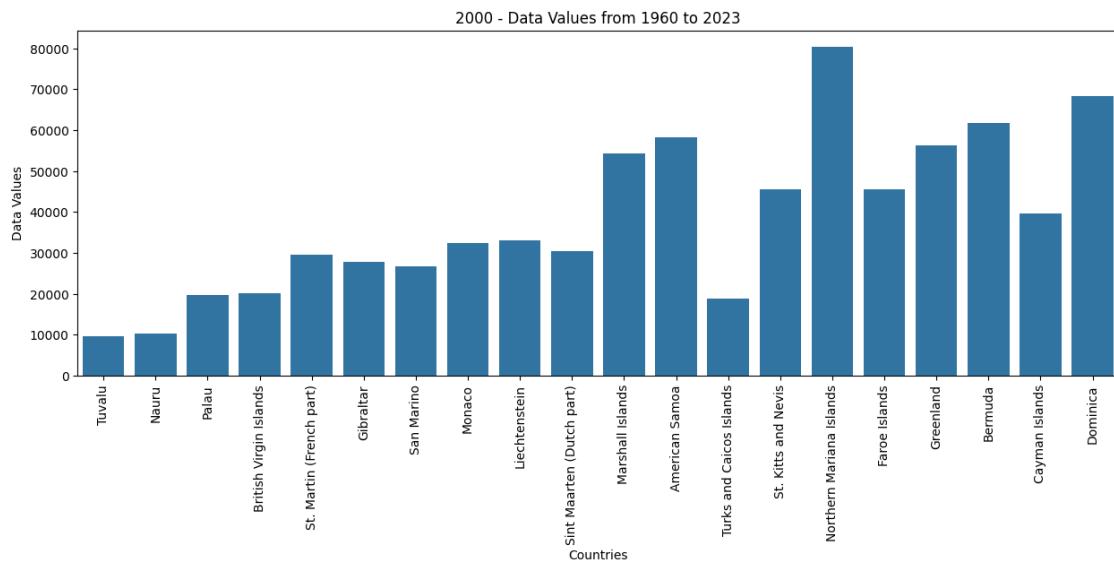
1996 - Data Values from 1960 to 2023

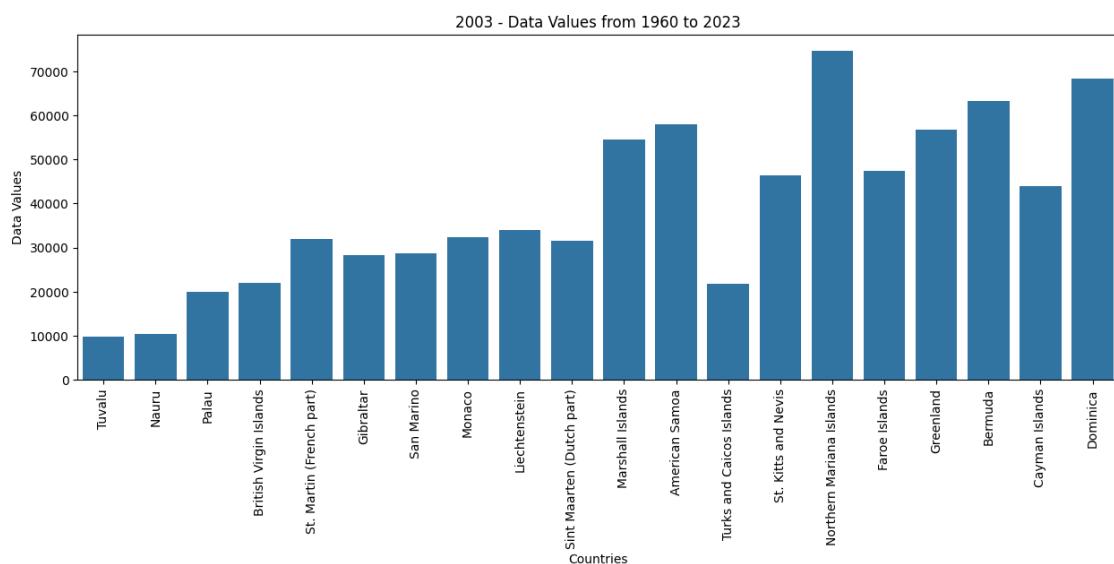
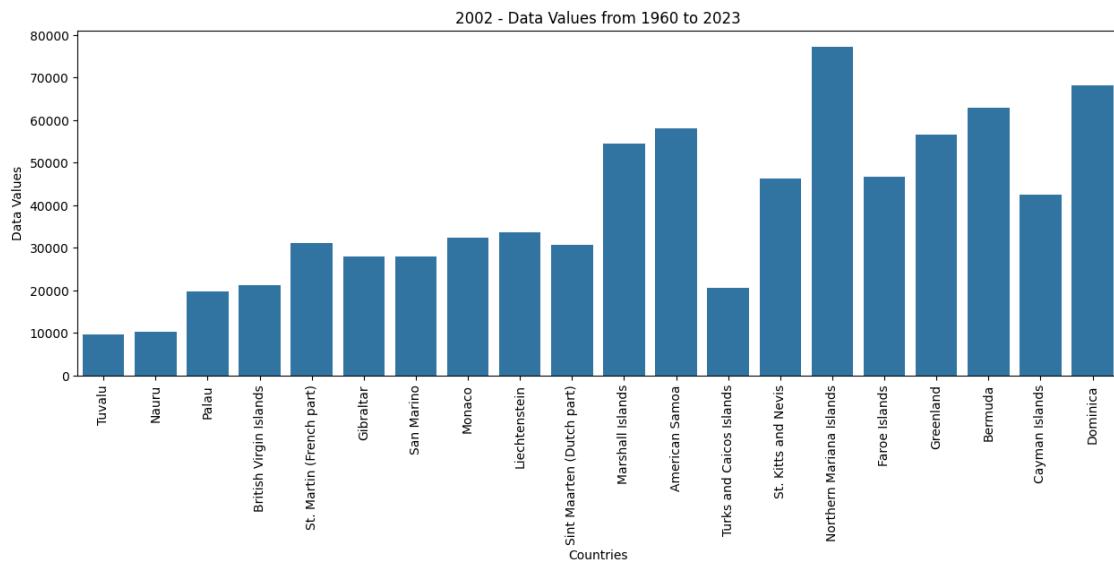


1997 - Data Values from 1960 to 2023

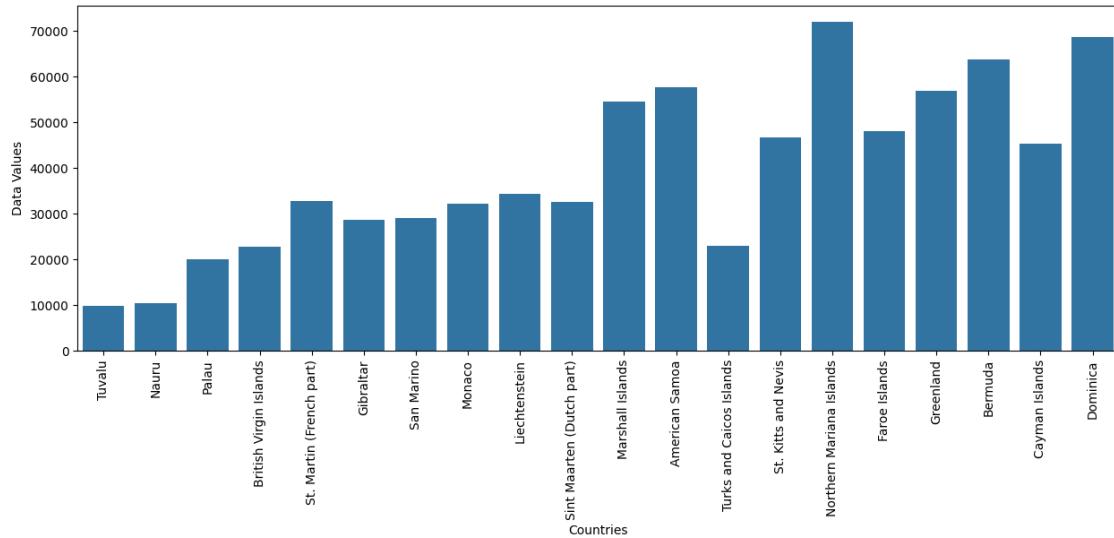




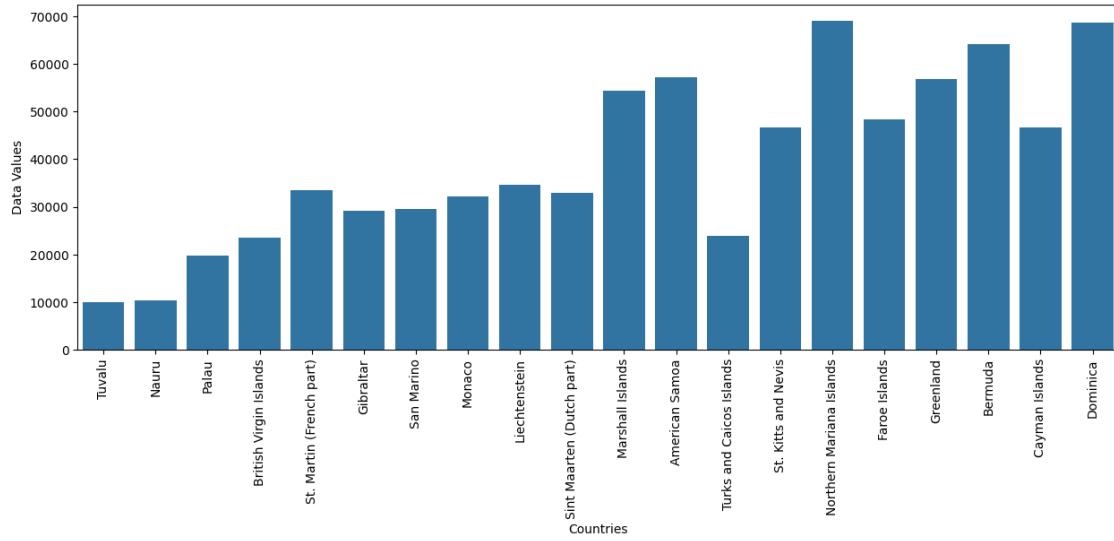


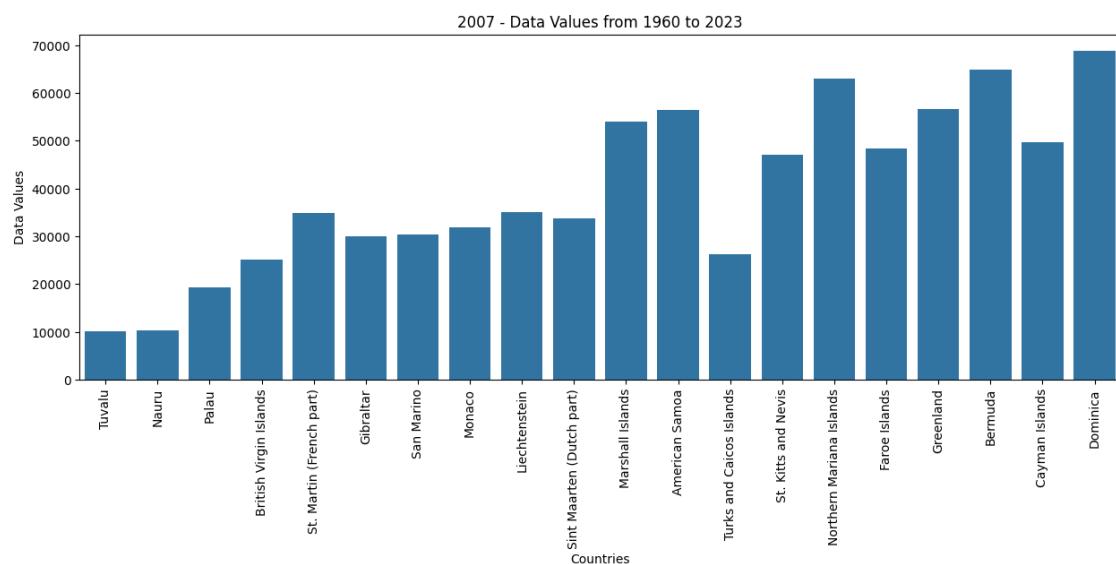
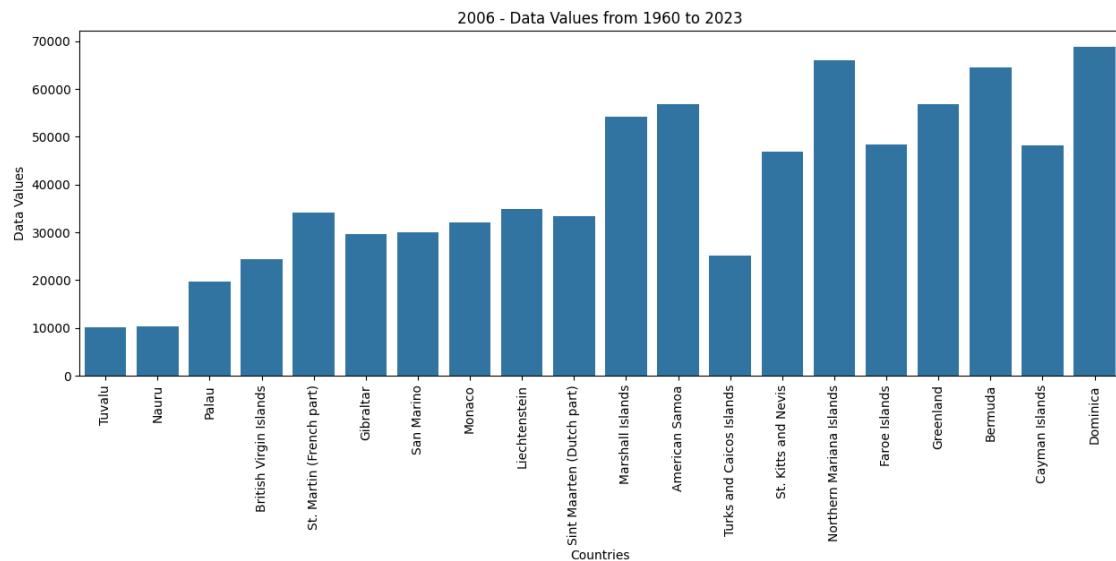


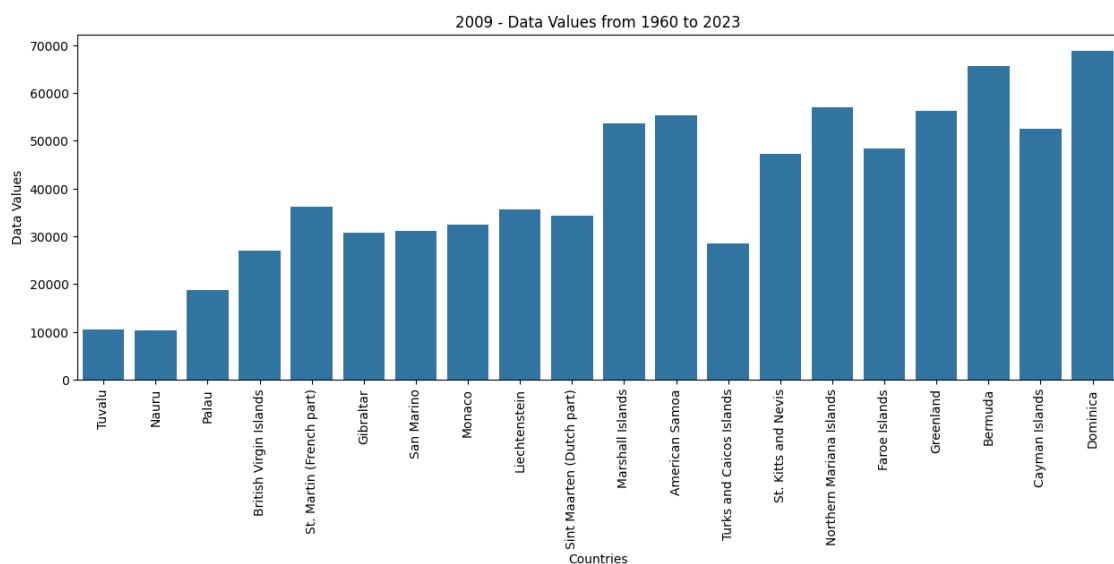
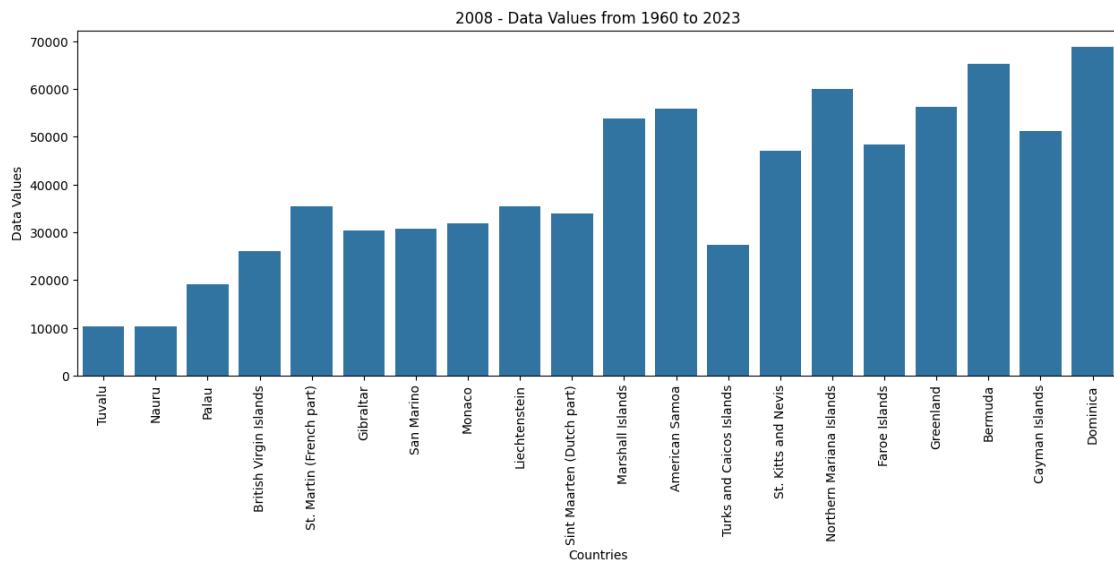
2004 - Data Values from 1960 to 2023

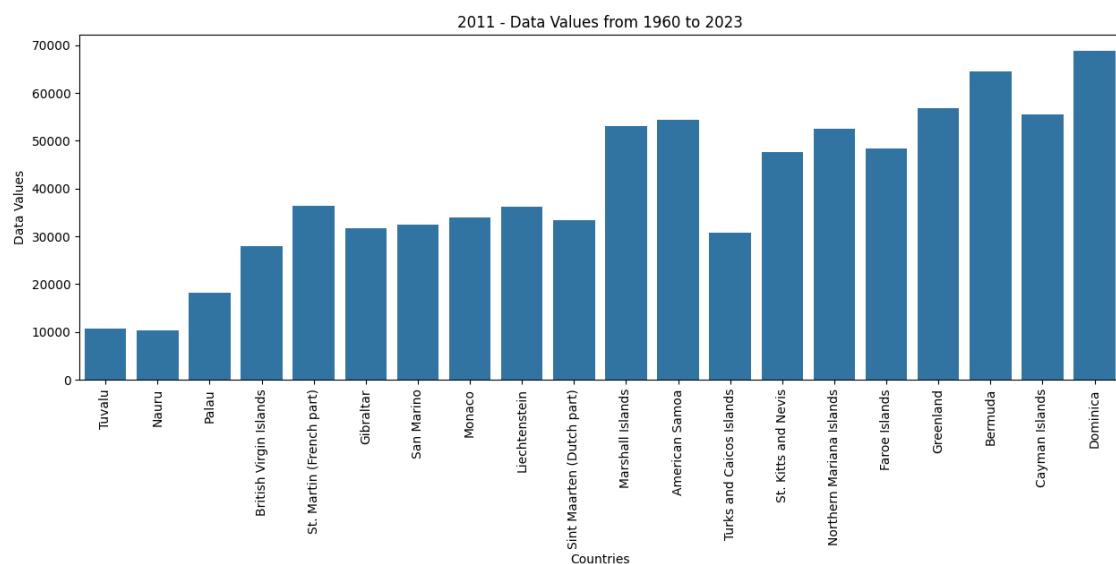
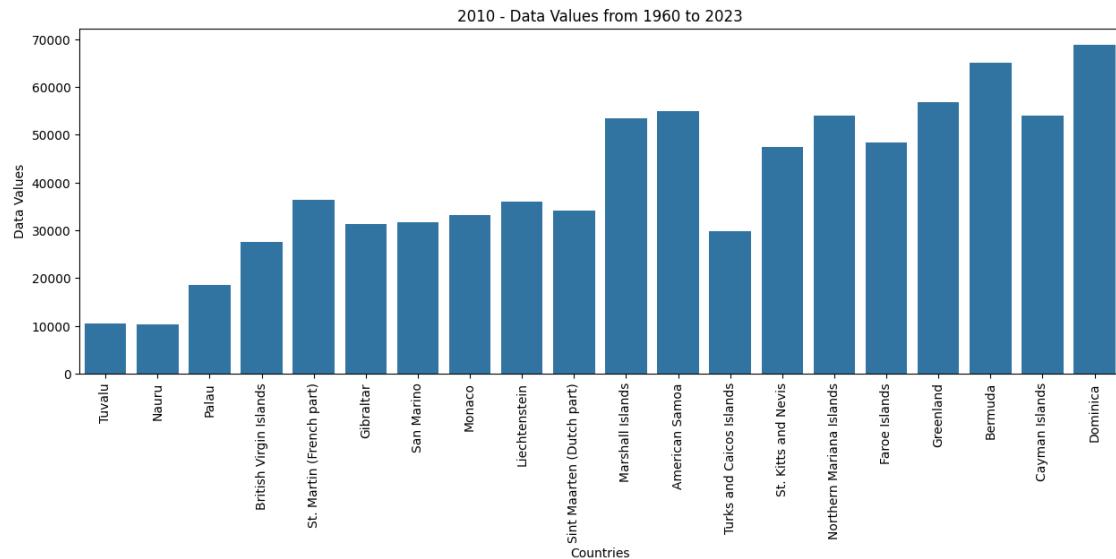


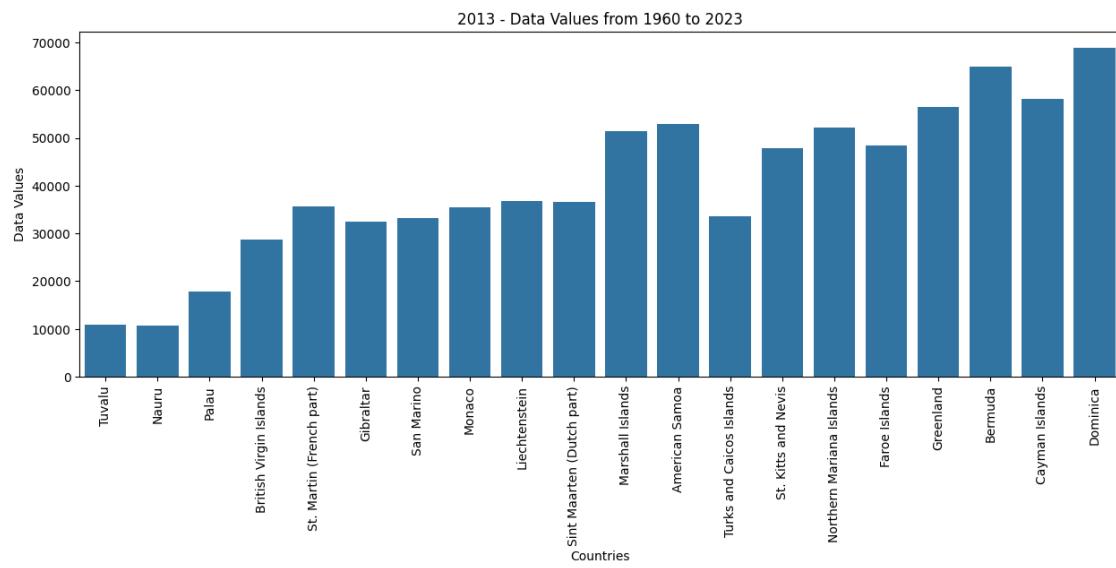
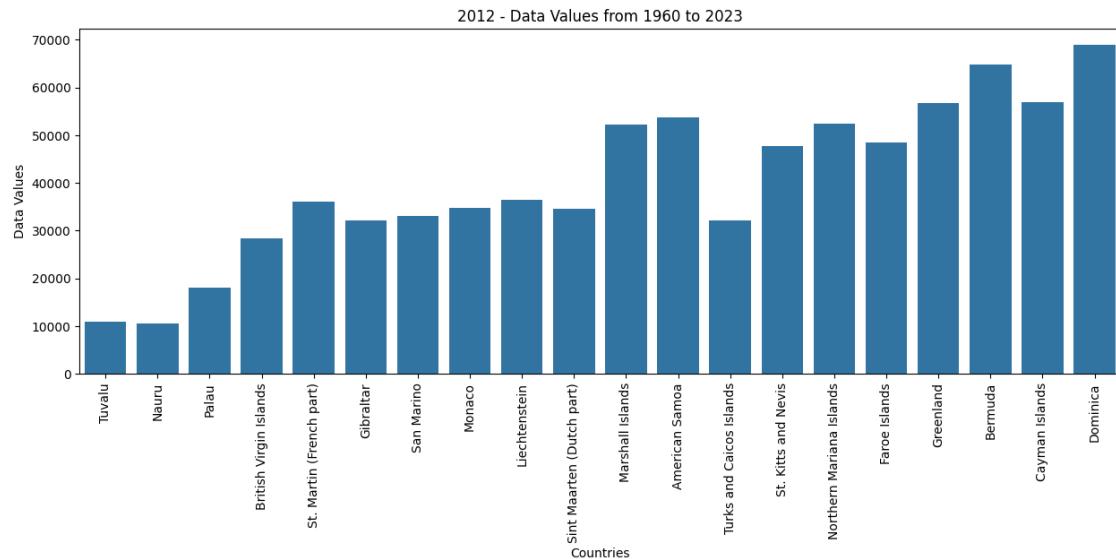
2005 - Data Values from 1960 to 2023

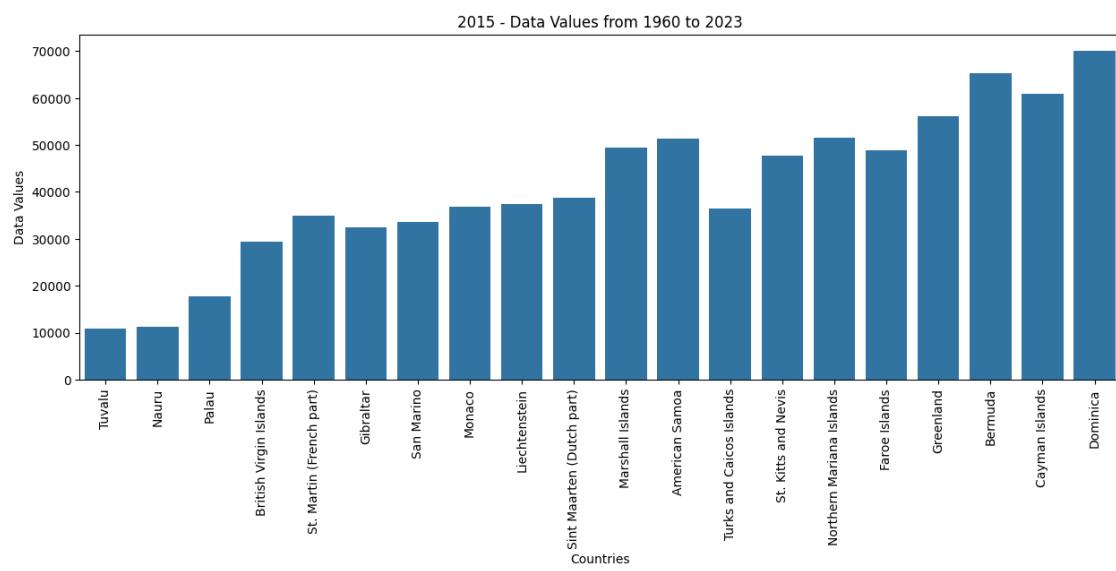
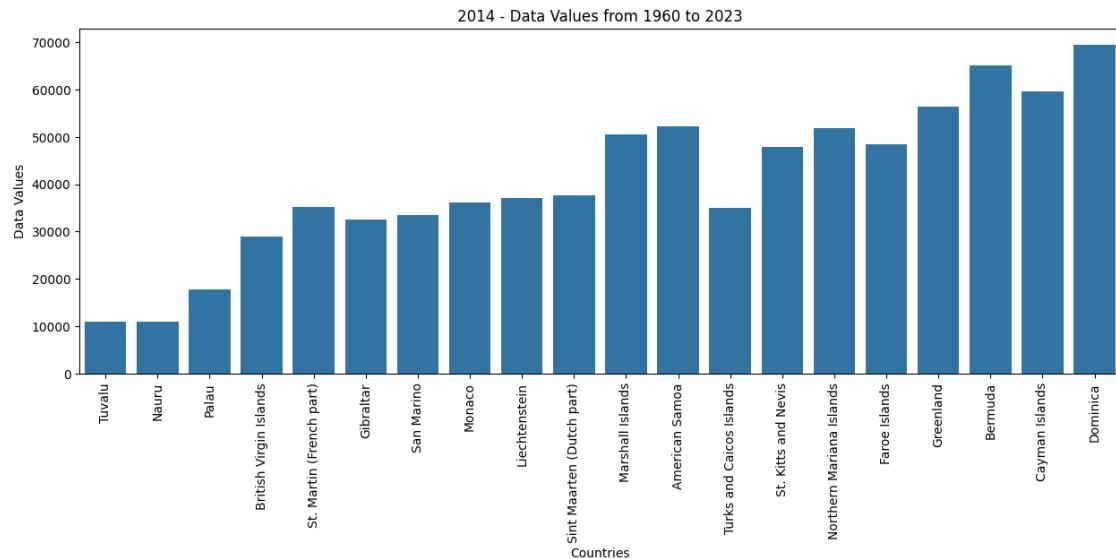




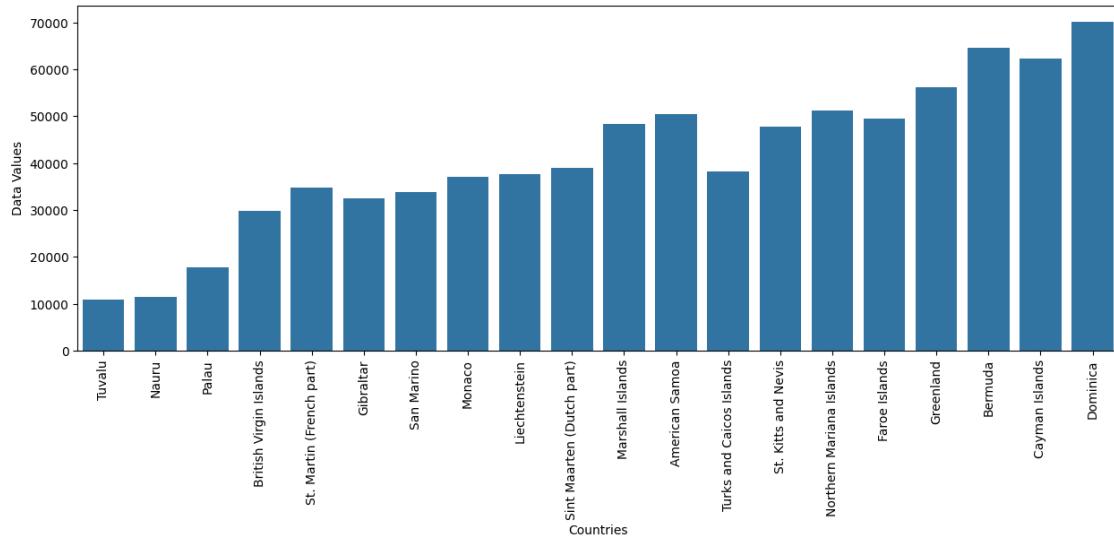




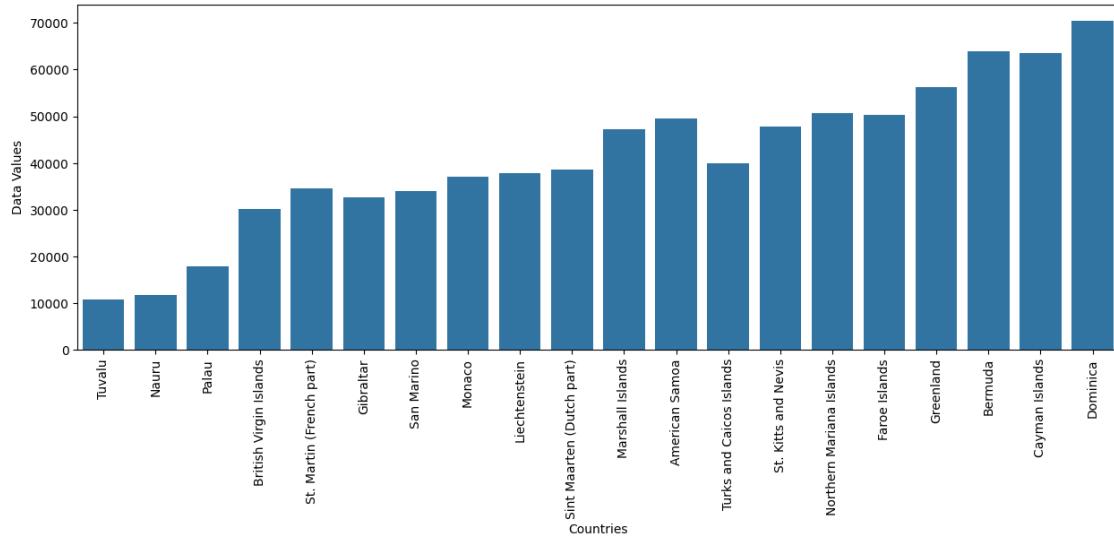




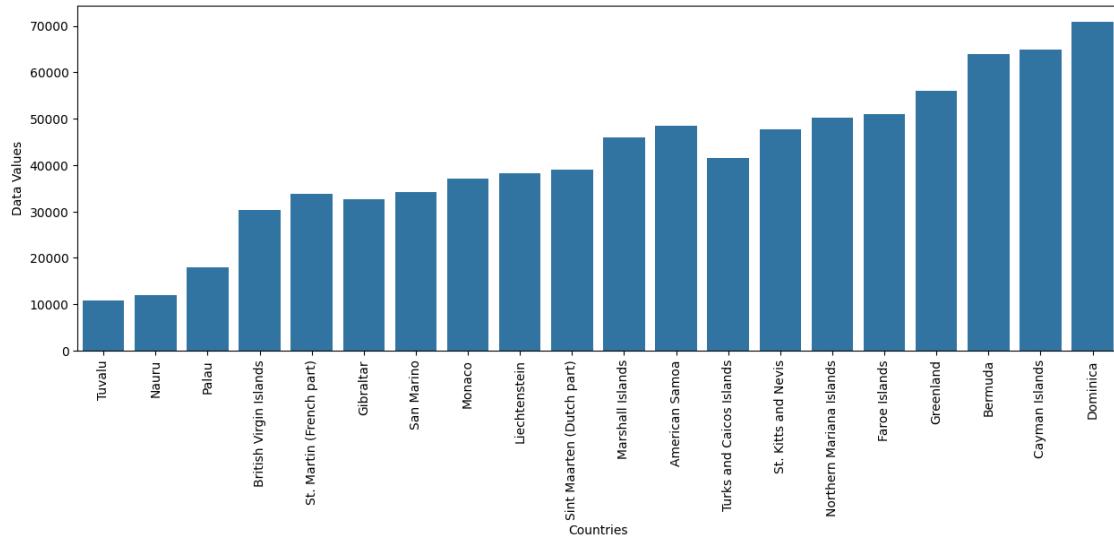
2016 - Data Values from 1960 to 2023



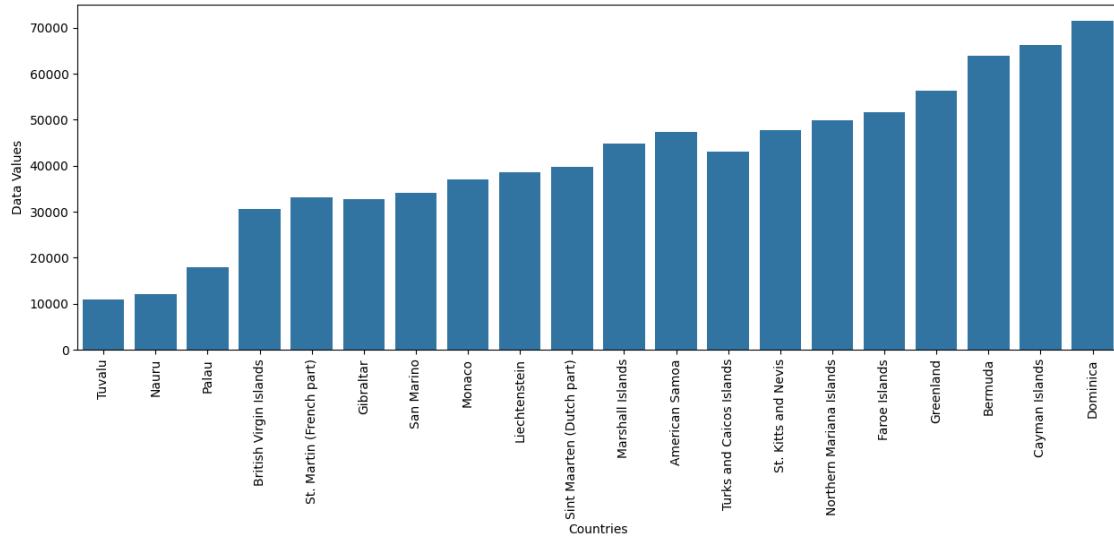
2017 - Data Values from 1960 to 2023

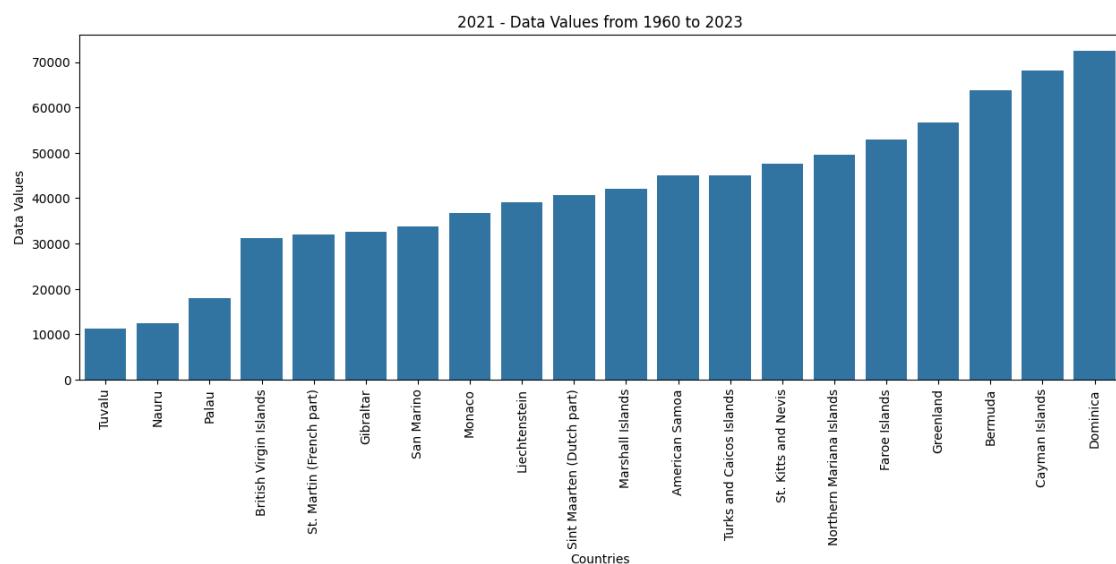
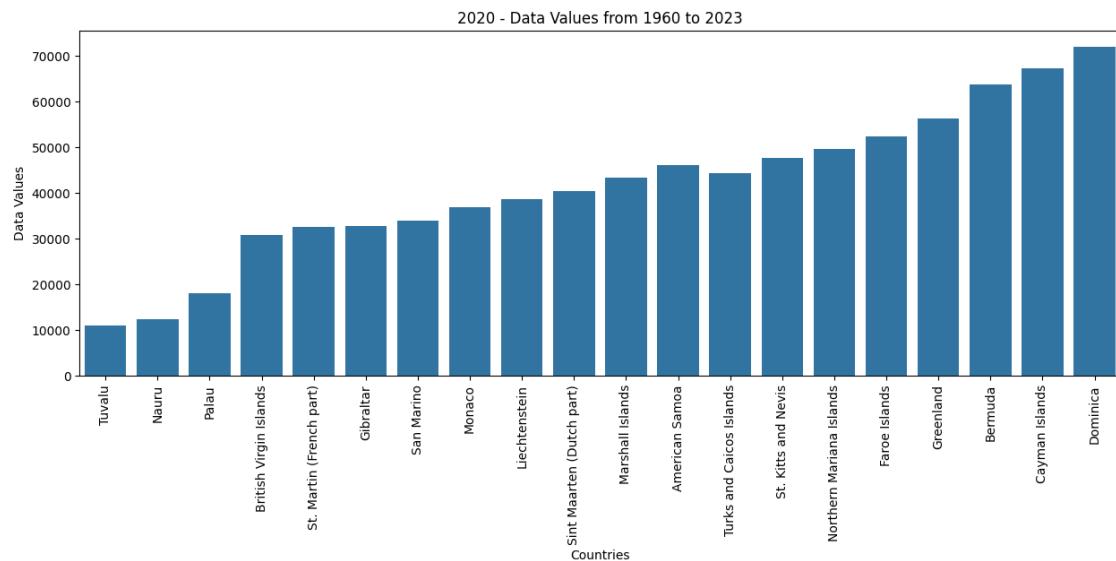


2018 - Data Values from 1960 to 2023

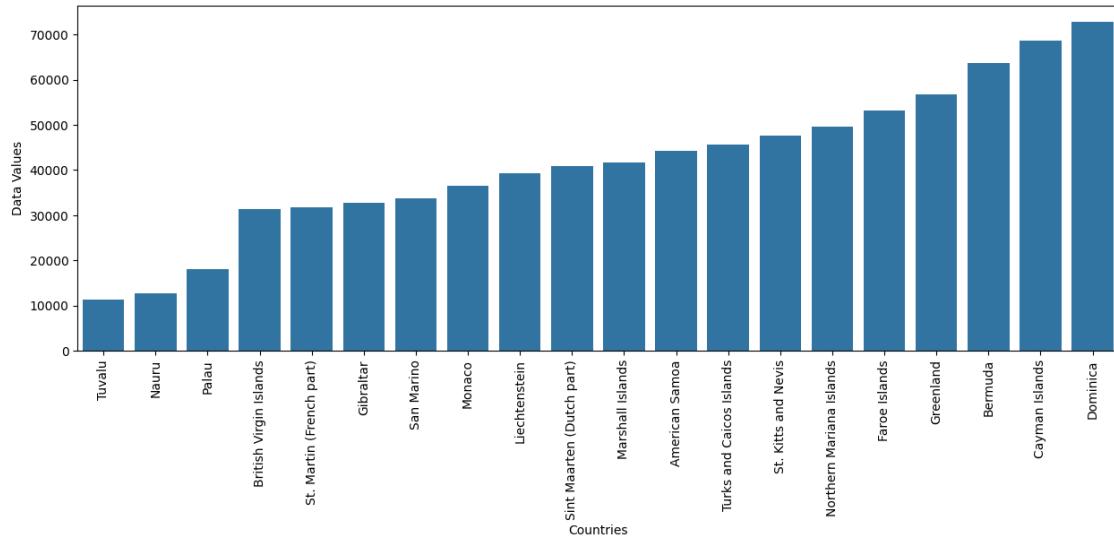


2019 - Data Values from 1960 to 2023





2022 - Data Values from 1960 to 2023



2023 - Data Values from 1960 to 2023

