In [1]:

```
!pip install geopandas
```

```
Requirement already satisfied: geopandas in /usr/local/lib/python3.10/dist-packages
(0.13.2)
Requirement already satisfied: fiona>=1.8.19 in /usr/local/lib/python3.10/dist-packag
es (from geopandas) (1.9.5)
Requirement already satisfied: packaging in /usr/local/lib/python3.10/dist-packages
(from geopandas) (23.2)
Requirement already satisfied: pandas>=1.1.0 in /usr/local/lib/python3.10/dist-packag
es (from geopandas) (1.5.3)
Requirement already satisfied: pyproj>=3.0.1 in /usr/local/lib/python3.10/dist-packag
es (from geopandas) (3.6.1)
Requirement already satisfied: shapely>=1.7.1 in /usr/local/lib/python3.10/dist-packa
ges (from geopandas) (2.0.2)
Requirement already satisfied: attrs>=19.2.0 in /usr/local/lib/python3.10/dist-packag
es (from fiona>=1.8.19->geopandas) (23.1.0)
Requirement already satisfied: certifi in /usr/local/lib/python3.10/dist-packages (fr
om fiona>=1.8.19->geopandas) (2023.11.17)
Requirement already satisfied: click~=8.0 in /usr/local/lib/python3.10/dist-packages
(from fiona>=1.8.19->geopandas) (8.1.7)
Requirement already satisfied: click-plugins>=1.0 in /usr/local/lib/python3.10/dist-p
ackages (from fiona>=1.8.19->geopandas) (1.1.1)
Requirement already satisfied: cligj>=0.5 in /usr/local/lib/python3.10/dist-packages
(from fiona>=1.8.19->geopandas) (0.7.2)
Requirement already satisfied: six in /usr/local/lib/python3.10/dist-packages (from f
iona>=1.8.19->geopandas) (1.16.0)
Requirement already satisfied: setuptools in /usr/local/lib/python3.10/dist-packages
(from fiona>=1.8.19->geopandas) (67.7.2)
Requirement already satisfied: python-dateutil>=2.8.1 in /usr/local/lib/python3.10/di
st-packages (from pandas>=1.1.0->geopandas) (2.8.2)
Requirement already satisfied: pytz>=2020.1 in /usr/local/lib/python3.10/dist-package
s (from pandas>=1.1.0->geopandas) (2023.3.post1)
Requirement already satisfied: numpy>=1.21.0 in /usr/local/lib/python3.10/dist-packag
es (from pandas>=1.1.0->geopandas) (1.23.5)
```

In [2]:

```
import pandas as pd
import geopandas as gpd
from matplotlib import pyplot as plt
import plotly.express as px
```

In [3]:

```
df_euro_data = pd.read_csv("https://raw.githubusercontent.com/SDuncan5/Eurostat-Data/m
df_euro_data = df_euro_data.drop(columns=["Unnamed: 0"])
df_euro_data
```

Out[3]:

	geo	TIME_PERIOD	CPI	Immigrants	Population	Housing Index	GDP	emigration	unempl
0	Austria	2011	93.35	82230.0	8391643.0	81.60	310128.7	51197.0	
1	Austria	2012	95.75	91557.0	8429991.0	87.57	318653.0	51812.0	
2	Austria	2013	97.77	101866.0	8479823.0	92.10	323910.2	54071.0	
3	Austria	2014	99.20	116262.0	8546356.0	95.33	333146.1	53491.0	
4	Austria	2015	100.00	166323.0	8642699.0	100.00	344269.2	56689.0	
...	...	...	...	...	...	...	...	...	
280	Slovakia	2017	100.90	7188.0	5439232.0	112.99	84669.9	3466.0	
281	Slovakia	2018	103.46	7253.0	5446771.0	121.32	89874.7	3298.0	
282	Slovakia	2019	106.33	7016.0	5454147.0	132.39	94429.7	3384.0	
283	Slovakia	2020	108.47	6775.0	5458827.0	145.06	93444.1	2428.0	
284	Slovakia	2021	111.53	5733.0	5447247.0	154.33	100255.7	3395.0	

285 rows × 12 columns

In [4]:

```
df_euro_data["geo"].value_counts().index
```

Out[4]:

```
Index(['Austria', 'Belgium', 'Slovenia', 'Sweden', 'Romania', 'Portugal',
       'Poland', 'Netherlands', 'Malta', 'Latvia', 'Luxembourg', 'Lithuania',
       'Italy', 'Ireland', 'Hungary', 'Croatia', 'France', 'Finland', 'Spain',
       'Estonia', 'Denmark', 'Germany', 'Czechia', 'Cyprus', 'Slovakia',
       'Bulgaria'],
      dtype='object')
```

Map Visualization

In [5]:

```
gpd.datasets.get_path("naturalearth_lowres")
!ls /usr/local/lib/python3.9/dist-packages/geopandas/datasets/naturalearth_lowres/
```

```
ls: cannot access '/usr/local/lib/python3.9/dist-packages/geopandas/datasets/naturale
arth_lowres/': No such file or directory
```

```
<ipython-input-5-b427af84020d>:1: FutureWarning: The geopandas.dataset module is depr
ecated and will be removed in GeoPandas 1.0. You can get the original 'naturalearth_l
owres' data from https://www.naturalearthdata.com/downloads/110m-cultural-vectors/.
  gpd.datasets.get_path("naturalearth_lowres")
```

In [6]:

```
world = gpd.read_file(gpd.datasets.get_path('naturalearth_lowres'))
world;
```

```
<ipython-input-6-4e58690a1ffb>:1: FutureWarning: The geopandas.dataset module is depr
ecated and will be removed in GeoPandas 1.0. You can get the original 'naturalearth_l
owres' data from https://www.naturalearthdata.com/downloads/110m-cultural-vectors/.
  world = gpd.read_file(gpd.datasets.get_path('naturalearth_lowres'))
```

In [7]:

```
europe = world[world["continent"] == "Europe"]
europe
```

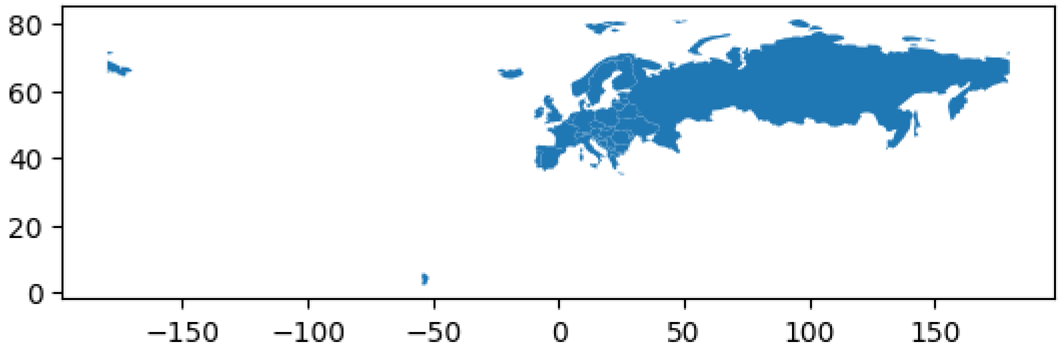
Out[7]:

	pop_est	continent	name	iso_a3	gdp_md_est	geometry
18	144373535.0	Europe	Russia	RUS	1699876	MULTIPOLYGON (((180.00000 71.51571, 180.00000 ...
21	5347896.0	Europe	Norway	NOR	403336	MULTIPOLYGON (((15.14282 79.67431, 15.52255 80...
43	67059887.0	Europe	France	FRA	2715518	MULTIPOLYGON (((-51.65780 4.15623, -52.24934 3...
110	10285453.0	Europe	Sweden	SWE	530883	POLYGON ((11.02737 58.85615, 11.46827 59.43239...
111	9466856.0	Europe	Belarus	BLR	63080	POLYGON ((28.17671 56.16913, 29.22951 55.91834...
112	44385155.0	Europe	Ukraine	UKR	153781	POLYGON ((32.15944 52.06125, 32.41206 52.28869...
113	37970874.0	Europe	Poland	POL	595858	POLYGON ((23.48413 53.91250, 23.52754 53.47012...
114	8877067.0	Europe	Austria	AUT	445075	POLYGON ((16.97967 48.12350, 16.90375 47.71487...
115	9769949.0	Europe	Hungary	HUN	163469	POLYGON ((22.08561 48.42226, 22.64082 48.15024...
116	2657637.0	Europe	Moldova	MDA	11968	POLYGON ((26.61934 48.22073, 26.85782 48.36821...
117	19356544.0	Europe	Romania	ROU	250077	POLYGON ((28.23355 45.48828, 28.67978 45.30403...
118	2786844.0	Europe	Lithuania	LTU	54627	POLYGON ((26.49433 55.61511, 26.58828 55.16718...
119	1912789.0	Europe	Latvia	LVA	34102	POLYGON ((27.28818 57.47453, 27.77002 57.24426...
120	1326590.0	Europe	Estonia	EST	31471	POLYGON ((27.98113 59.47537, 27.98112 59.47537...
121	83132799.0	Europe	Germany	DEU	3861123	POLYGON ((14.11969 53.75703, 14.35332 53.24817...
122	6975761.0	Europe	Bulgaria	BGR	68558	POLYGON ((22.65715 44.23492, 22.94483 43.82379...
123	10716322.0	Europe	Greece	GRC	209852	MULTIPOLYGON (((26.29000 35.29999, 26.16500 35...
125	2854191.0	Europe	Albania	ALB	15279	POLYGON ((21.02004 40.84273, 20.99999 40.58000...
126	4067500.0	Europe	Croatia	HRV	60752	POLYGON ((16.56481 46.50375, 16.88252 46.38063...
127	8574832.0	Europe	Switzerland	CHE	703082	POLYGON ((9.59423 47.52506, 9.63293 47.34760, ...
128	619896.0	Europe	Luxembourg	LUX	71104	POLYGON ((6.04307 50.12805, 6.24275 49.90223, ...

	pop_est	continent	name	iso_a3	gdp_md_est	geometry
129	11484055.0	Europe	Belgium	BEL	533097	POLYGON ((6.15666 50.80372, 6.04307 50.12805, ...
130	17332850.0	Europe	Netherlands	NLD	907050	POLYGON ((6.90514 53.48216, 7.09205 53.14404, ...
131	10269417.0	Europe	Portugal	PRT	238785	POLYGON ((-9.03482 41.88057, -8.67195 42.13469...
132	47076781.0	Europe	Spain	ESP	1393490	POLYGON ((-7.45373 37.09779, -7.53711 37.42890...
133	4941444.0	Europe	Ireland	IRL	388698	POLYGON ((-6.19788 53.86757, -6.03299 53.15316...
141	60297396.0	Europe	Italy	ITA	2003576	MULTIPOLYGON (((10.44270 46.89355, 11.04856 46...
142	5818553.0	Europe	Denmark	DNK	350104	MULTIPOLYGON (((9.92191 54.98310, 9.28205 54.8...
143	66834405.0	Europe	United Kingdom	GBR	2829108	MULTIPOLYGON (((-6.19788 53.86757, -6.95373 54...
144	361313.0	Europe	Iceland	ISL	24188	POLYGON ((-14.50870 66.45589, -14.73964 65.808...
150	2087946.0	Europe	Slovenia	SVN	54174	POLYGON ((13.80648 46.50931, 14.63247 46.43182...
151	5520314.0	Europe	Finland	FIN	269296	POLYGON ((28.59193 69.06478, 28.44594 68.36461...
152	5454073.0	Europe	Slovakia	SVK	105079	POLYGON ((22.55814 49.08574, 22.28084 48.82539...
153	10669709.0	Europe	Czechia	CZE	250680	POLYGON ((15.01700 51.10667, 15.49097 50.78473...
170	3301000.0	Europe	Bosnia and Herz.	BIH	20164	POLYGON ((18.56000 42.65000, 17.67492 43.02856...
171	2083459.0	Europe	North Macedonia	MKD	12547	POLYGON ((22.38053 42.32026, 22.88137 41.99930...
172	6944975.0	Europe	Serbia	SRB	51475	POLYGON ((18.82982 45.90887, 18.82984 45.90888...
173	622137.0	Europe	Montenegro	MNE	5542	POLYGON ((20.07070 42.58863, 19.80161 42.50009...
174	1794248.0	Europe	Kosovo	-99	7926	POLYGON ((20.59025 41.85541, 20.52295 42.21787...

In [8]:

```
europe.plot();
```



In [9]:

```
df_euro_2021 = europe.merge(df_euro_data[df_euro_data["TIME_PERIOD"] == 2021],
                           how="inner",
                           left_on="name",
                           right_on="geo")
df_euro_2021
```

Out[9]:

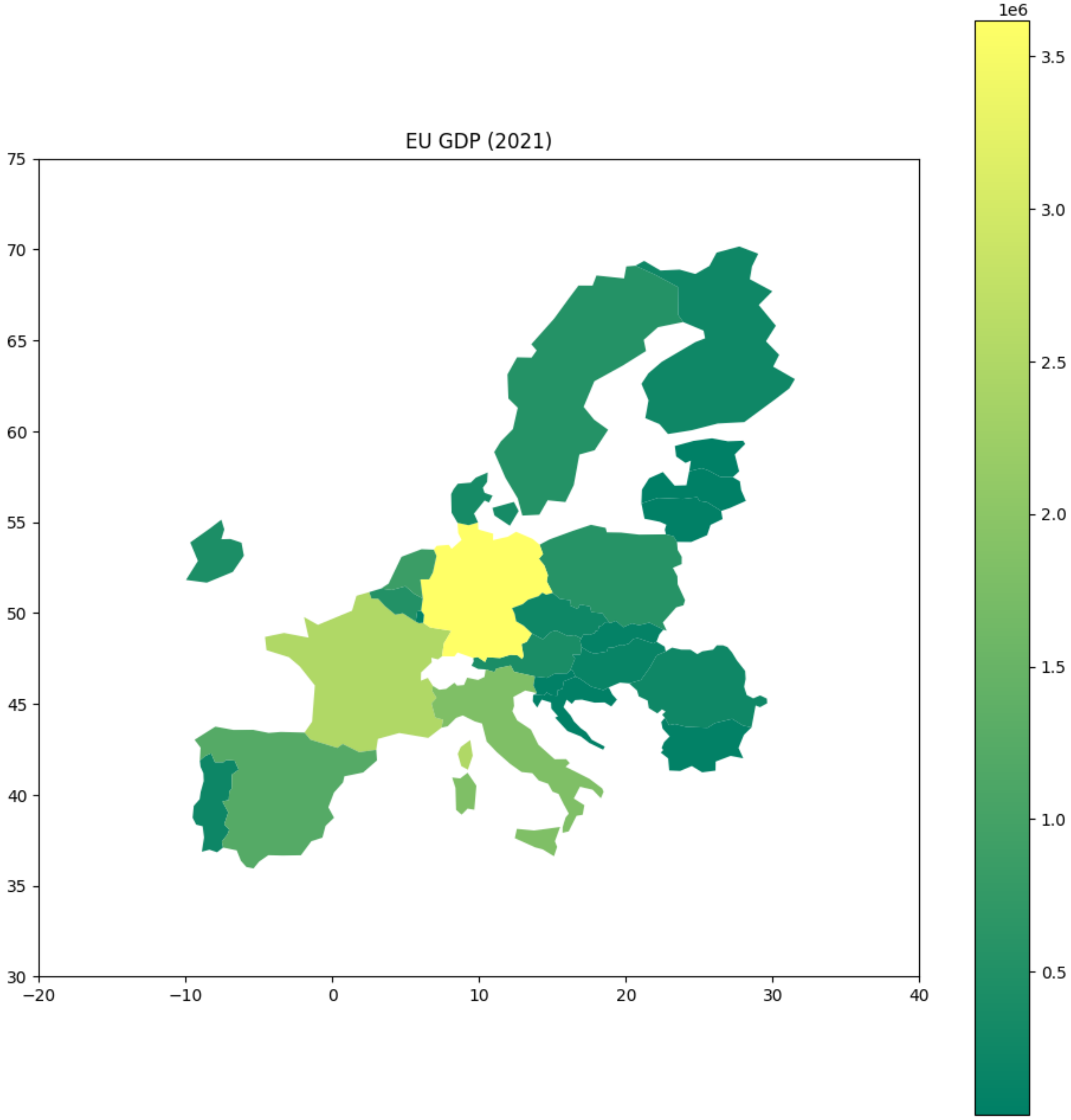
	pop_est	continent	name	iso_a3	gdp_md_est	geometry	geo	TIME_PERIO
0	67059887.0	Europe	France	FRA	2715518	MULTIPOLYGON (((-51.65780 4.15623, -52.24934 3...	France	202
1	10285453.0	Europe	Sweden	SWE	530883	POLYGON ((11.02737 58.85615, 11.46827 59.43239...	Sweden	202
2	37970874.0	Europe	Poland	POL	595858	POLYGON ((23.48413 53.91250, 23.52754 53.47012...	Poland	202
3	8877067.0	Europe	Austria	AUT	445075	POLYGON ((16.97967 48.12350, 16.90375 47.71487...	Austria	202
4	9769949.0	Europe	Hungary	HUN	163469	POLYGON ((22.08561 48.42226, 22.64082 48.15024...	Hungary	202
5	19356544.0	Europe	Romania	ROU	250077	POLYGON ((28.23355 45.48828, 28.67978 45.30403...	Romania	202
6	2786844.0	Europe	Lithuania	LTU	54627	POLYGON ((26.49433 55.61511, 26.58828 55.16718...	Lithuania	202
7	1912789.0	Europe	Latvia	LVA	34102	POLYGON ((27.28818 57.47453, 27.77002 57.24426...	Latvia	202
8	1326590.0	Europe	Estonia	EST	31471	POLYGON ((27.98113 59.47537, 27.98112 59.47537...	Estonia	202
9	83132799.0	Europe	Germany	DEU	3861123	POLYGON ((14.11969 53.75703, 14.35332 53.24817...	Germany	202

	pop_est	continent	name	iso_a3	gdp_md_est	geometry	geo	TIME_PERIO
10	6975761.0	Europe	Bulgaria	BGR	68558	POLYGON ((22.65715 44.23492, 22.94483 43.82379...	Bulgaria	202
11	4067500.0	Europe	Croatia	HRV	60752	POLYGON ((16.56481 46.50375, 16.88252 46.38063...	Croatia	202
12	619896.0	Europe	Luxembourg	LUX	71104	POLYGON ((6.04307 50.12805, 6.24275 49.90223, ...	Luxembourg	202
13	11484055.0	Europe	Belgium	BEL	533097	POLYGON ((6.15666 50.80372, 6.04307 50.12805, ...	Belgium	202
14	17332850.0	Europe	Netherlands	NLD	907050	POLYGON ((6.90514 53.48216, 7.09205 53.14404, ...	Netherlands	202
15	10269417.0	Europe	Portugal	PRT	238785	POLYGON ((-9.03482 41.88057, -8.67195 42.13469...	Portugal	202
16	47076781.0	Europe	Spain	ESP	1393490	POLYGON ((-7.45373 37.09779, -7.53711 37.42890...	Spain	202
17	4941444.0	Europe	Ireland	IRL	388698	POLYGON ((-6.19788 53.86757, -6.03299 53.15316...	Ireland	202
18	60297396.0	Europe	Italy	ITA	2003576	MULTIPOLYGON (((10.44270 46.89355, 11.04856 46...	Italy	202
19	5818553.0	Europe	Denmark	DNK	350104	MULTIPOLYGON (((9.92191 54.98310, 9.28205 54.8...	Denmark	202
20	2087946.0	Europe	Slovenia	SVN	54174	POLYGON ((13.80648	Slovenia	202

	pop_est	continent	name	iso_a3	gdp_md_est	geometry	geo	TIME_PERIO
						46.50931, 14.63247 46.43182...		
21	5520314.0	Europe	Finland	FIN	269296	POLYGON ((28.59193 69.06478, 28.44594 68.36461...	Finland	202
22	5454073.0	Europe	Slovakia	SVK	105079	POLYGON ((22.55814 49.08574, 22.28084 48.82539...	Slovakia	202
23	10669709.0	Europe	Czechia	CZE	250680	POLYGON ((15.01700 51.10667, 15.49097 50.78473...	Czechia	202

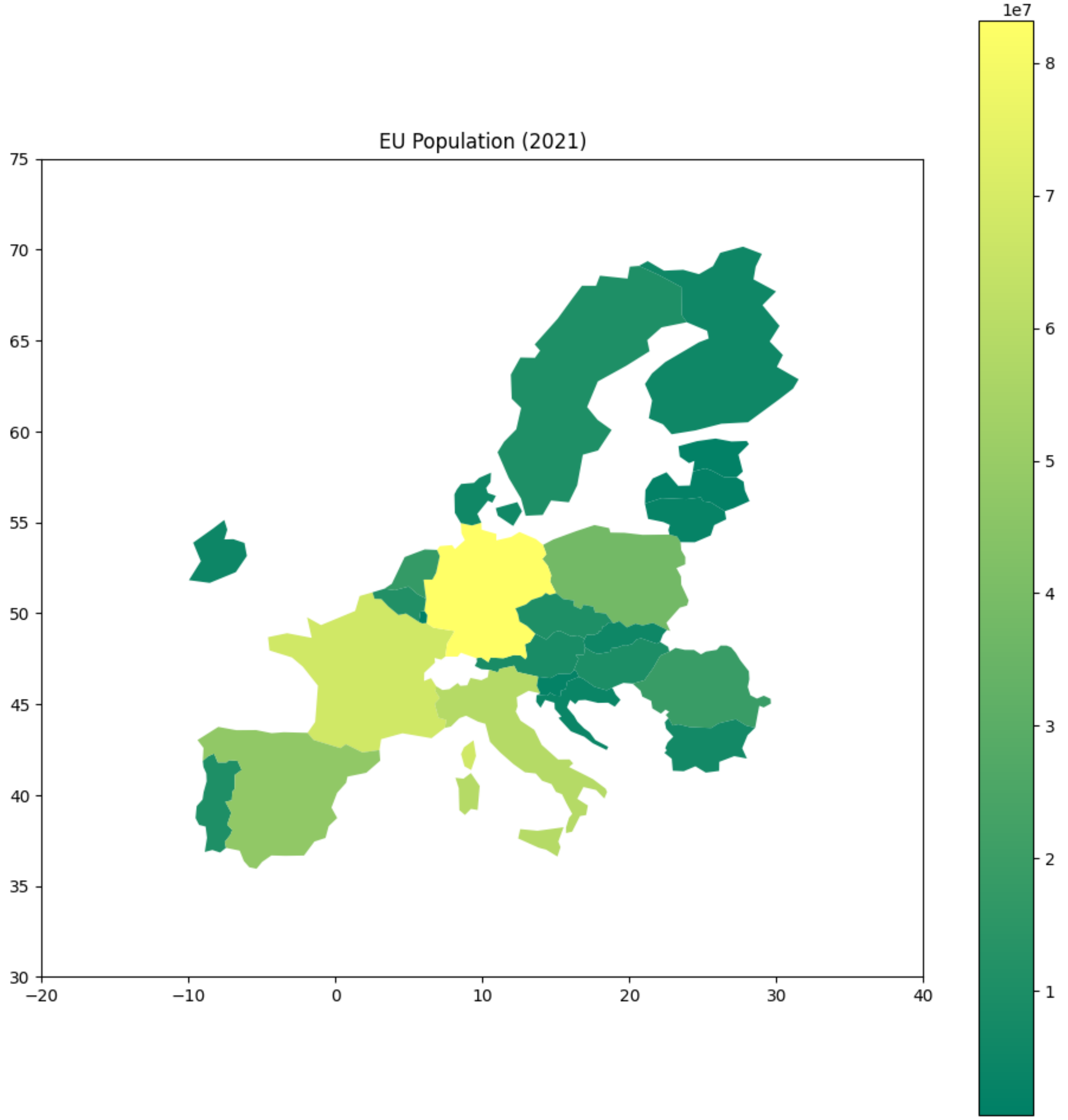
In [10]:

```
df_euro_2021.plot(column = "GDP", cmap = "summer", legend=True, figsize=(12, 12));
plt.xlim(-20, 40);
plt.ylim(30, 75);
plt.title("EU GDP (2021)");
```



In [11]:

```
df_euro_2021.plot(column = "Population", cmap = "summer", legend=True, figsize=(12, 12
plt.xlim(-20, 40);
plt.ylim(30, 75);
plt.title("EU Population (2021)");
```

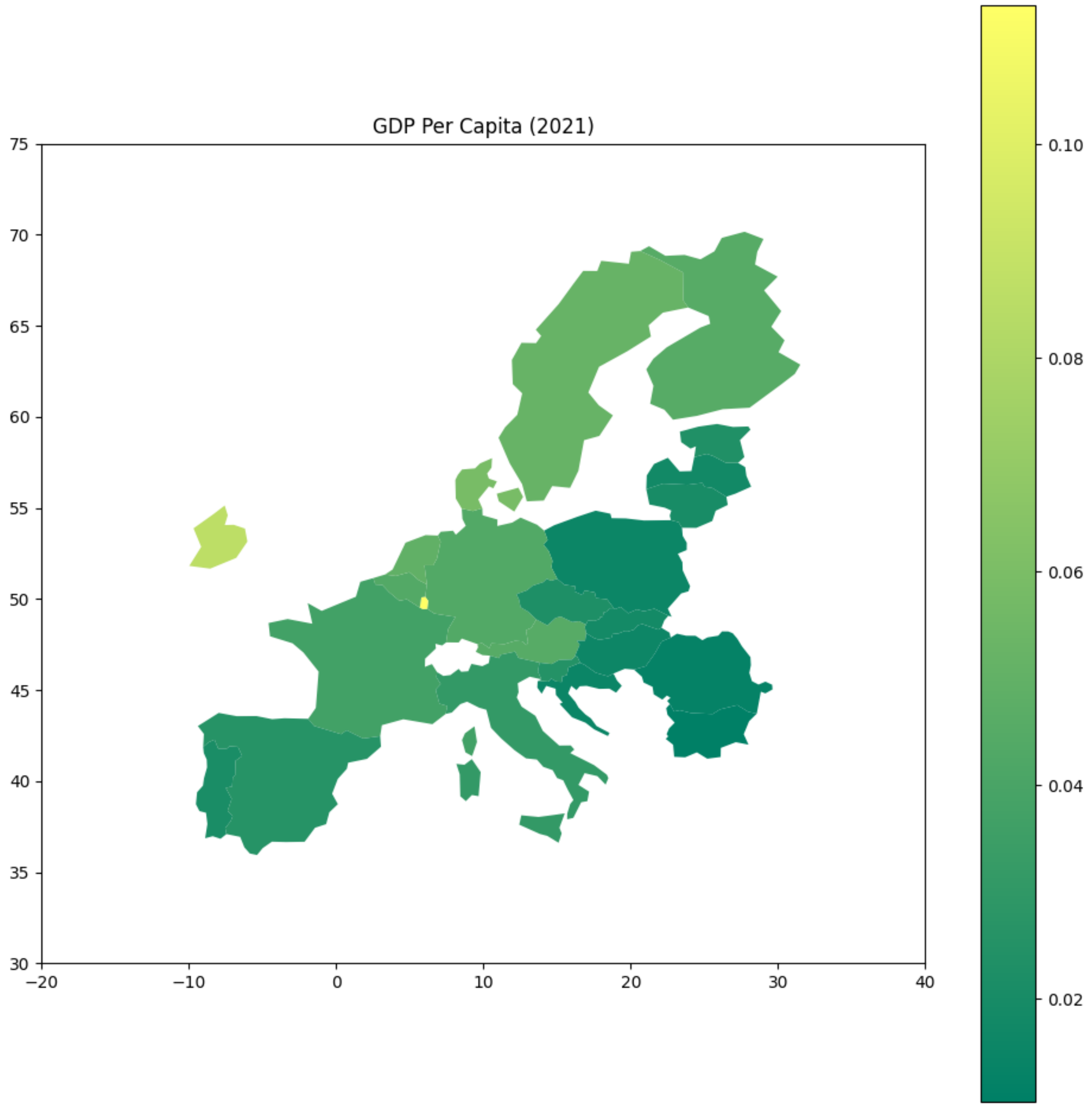


In [12]:

```
# GDP per capita
df_euro_2021["GDP Per Capita"] = df_euro_2021["GDP"] / df_euro_2021["Population"]
```

In [13]:

```
df_euro_2021.plot(column = "GDP Per Capita", cmap = "summer", legend=True, figsize=(12
plt.xlim(-20, 40);
plt.ylim(30, 75);
plt.title("GDP Per Capita (2021)");
```



Features Over Time (Line Plots)

Below are visualizations for each feature over time.

Because each country's population varies, some values will be higher or lower just because their population is higher/lower. To counteract that, I plotted line plots for each feature as well as values per capita (for features directly influenced by population)

```
In [14]: df_euro_data.rename(columns={'TIME_PERIOD': "Year"}, inplace=True)
         df_euro_data
```

Out[14]:

	geo	Year	CPI	Immigrants	Population	Housing Index	GDP	emigration	unemployment
0	Austria	2011	93.35	82230.0	8391643.0	81.60	310128.7	51197.0	3.3
1	Austria	2012	95.75	91557.0	8429991.0	87.57	318653.0	51812.0	3.5
2	Austria	2013	97.77	101866.0	8479823.0	92.10	323910.2	54071.0	3.8
3	Austria	2014	99.20	116262.0	8546356.0	95.33	333146.1	53491.0	4.0
4	Austria	2015	100.00	166323.0	8642699.0	100.00	344269.2	56689.0	4.1
...	...	...	...	...	...	...	...	...	...
280	Slovakia	2017	100.90	7188.0	5439232.0	112.99	84669.9	3466.0	5.4
281	Slovakia	2018	103.46	7253.0	5446771.0	121.32	89874.7	3298.0	4.3
282	Slovakia	2019	106.33	7016.0	5454147.0	132.39	94429.7	3384.0	3.8
283	Slovakia	2020	108.47	6775.0	5458827.0	145.06	93444.1	2428.0	4.4
284	Slovakia	2021	111.53	5733.0	5447247.0	154.33	100255.7	3395.0	4.5

285 rows × 12 columns

In [15]:

```python
# Plotting per capita data
df_euro_data_pcap = df_euro_data.copy()
df_euro_data_pcap["Immigrants_pcap"] = df_euro_data_pcap["Immigrants"] / df_euro_data_
df_euro_data_pcap["GDP_pcap"] = df_euro_data_pcap["GDP"] / df_euro_data_pcap["Populati
df_euro_data_pcap["Emigrants_pcap"] = df_euro_data_pcap["emigration"] / df_euro_data_p
df_euro_data_pcap["Deaths_pcap"] = df_euro_data_pcap["total_deaths"] / df_euro_data_pc
df_euro_data_pcap["Exports_pcap"] = df_euro_data_pcap["Exports"] / df_euro_data_pcap['
df_euro_data_pcap["Imports_pcap"] = df_euro_data_pcap["Imports"] / df_euro_data_pcap['
# Beware that all data is tiny b/c GDP is measured in million euro
# Multiplied GDP by 1000000 so we can measure in Euros (instead of million euros)

df_euro_data_pcap
```

Out[15]:

	geo	Year	CPI	Immigrants	Population	Housing Index	GDP	emigration	unemployment
0	Austria	2011	93.35	82230.0	8391643.0	81.60	310128.7	51197.0	3.3
1	Austria	2012	95.75	91557.0	8429991.0	87.57	318653.0	51812.0	3.5
2	Austria	2013	97.77	101866.0	8479823.0	92.10	323910.2	54071.0	3.8
3	Austria	2014	99.20	116262.0	8546356.0	95.33	333146.1	53491.0	4.0
4	Austria	2015	100.00	166323.0	8642699.0	100.00	344269.2	56689.0	4.1
...	...	...	...	...	...	...	...	...	...
280	Slovakia	2017	100.90	7188.0	5439232.0	112.99	84669.9	3466.0	5.4
281	Slovakia	2018	103.46	7253.0	5446771.0	121.32	89874.7	3298.0	4.3
282	Slovakia	2019	106.33	7016.0	5454147.0	132.39	94429.7	3384.0	3.8
283	Slovakia	2020	108.47	6775.0	5458827.0	145.06	93444.1	2428.0	4.4
284	Slovakia	2021	111.53	5733.0	5447247.0	154.33	100255.7	3395.0	4.5

285 rows × 18 columns

In [16]:

```
df_euro_data_pcap.to_csv('eurostat_pcap_no_nans.csv')
```

CPI

In [17]:

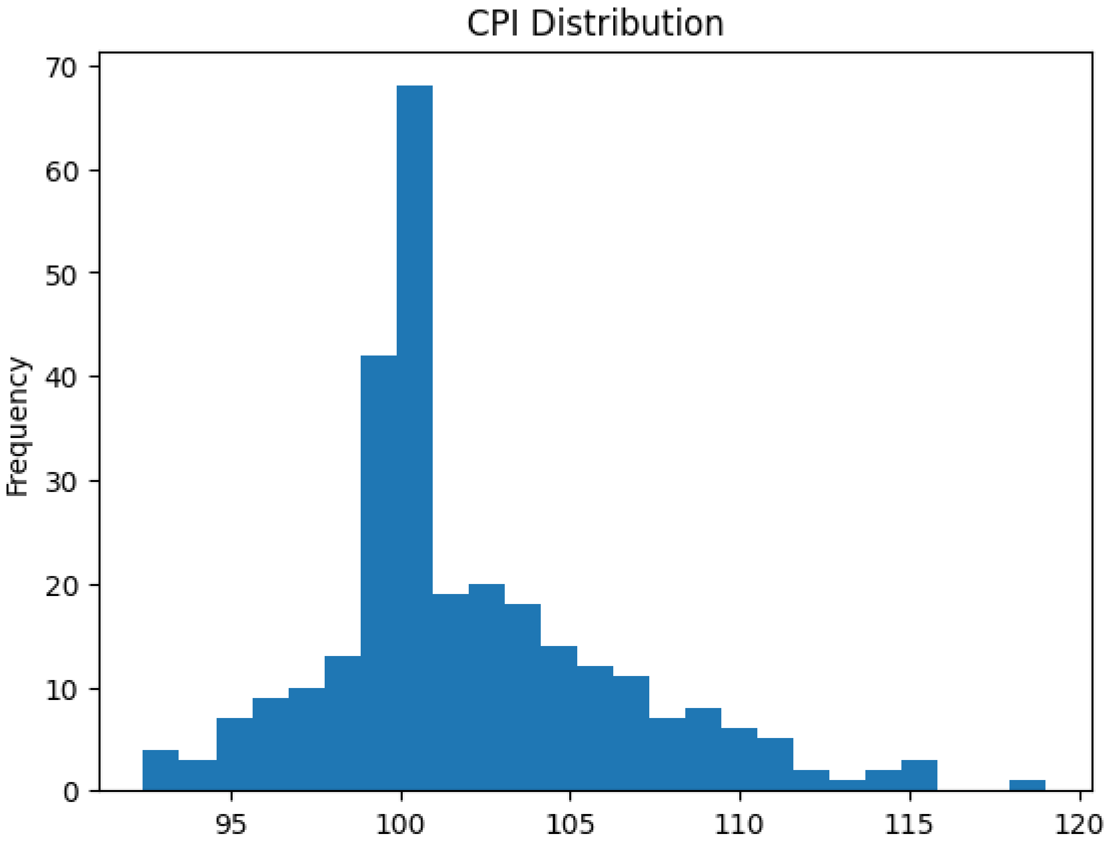
```
px.line(df_euro_data, x="Year", y="CPI", color="geo", markers=True, title="CPI Over Ti
```

In [36]:

```
df_euro_data_pcap['CPI'].plot.hist(xlabel="CPI", title="CPI Distribution", bins=25)
```

Out[36]:

```
<Axes: title={'center': 'CPI Distribution'}, ylabel='Frequency'>
```



Immigration

```
In [18]: px.line(df_euro_data, x="Year", y="Immigrants", color="geo", markers=True, title="Immi
                 labels={"Immigrants": "Number of Immigrants"})
```

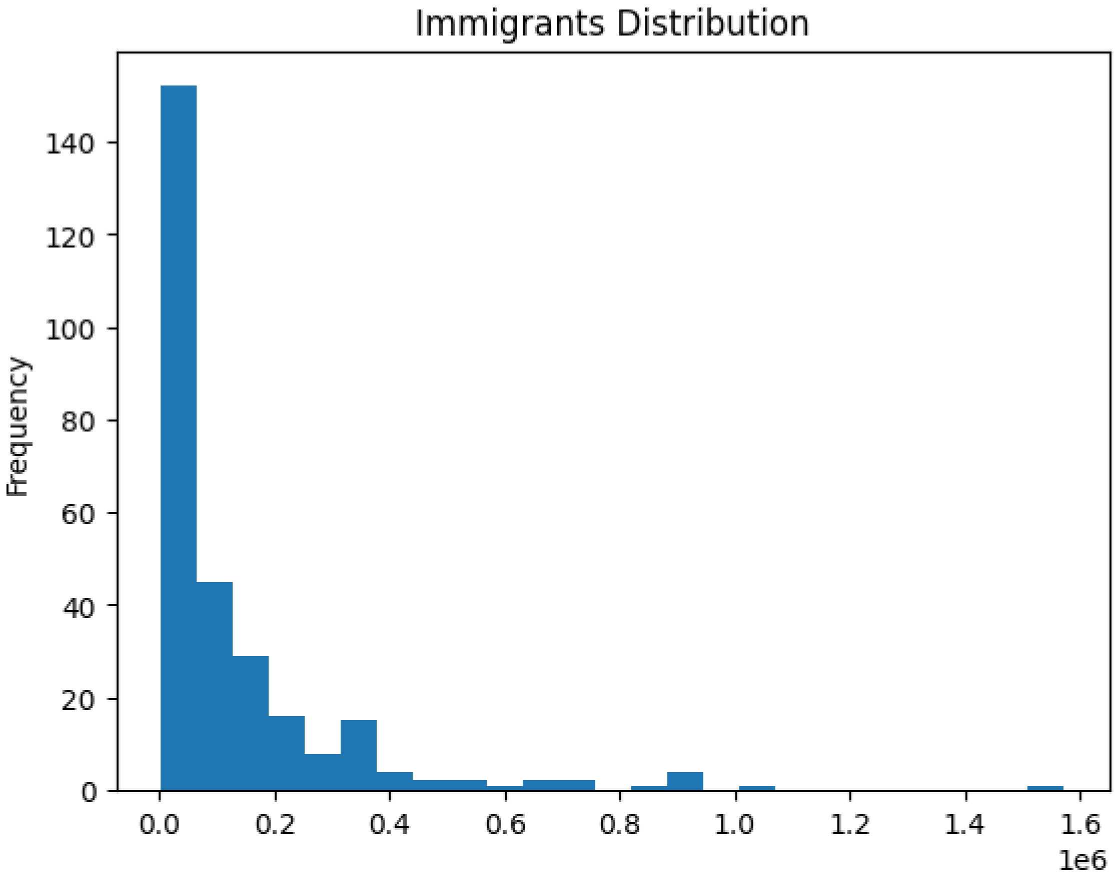
Germany spike in 2015 - Syrian immigration

In [37]:

```
df_euro_data_pcap['Immigrants'].plot.hist(xlabel="Number of Immigrants", title="Immigr
```

Out[37]:

```
<Axes: title={'center': 'Immigrants Distribution'}, ylabel='Frequency'>
```



In [19]:

```
px.line(df_euro_data_pcap, x="Year", y="Immigrants_pcap", color="geo", markers=True, t
        labels={"Immigrants_pcap": "Number of Immigrants Per Capita"})
```

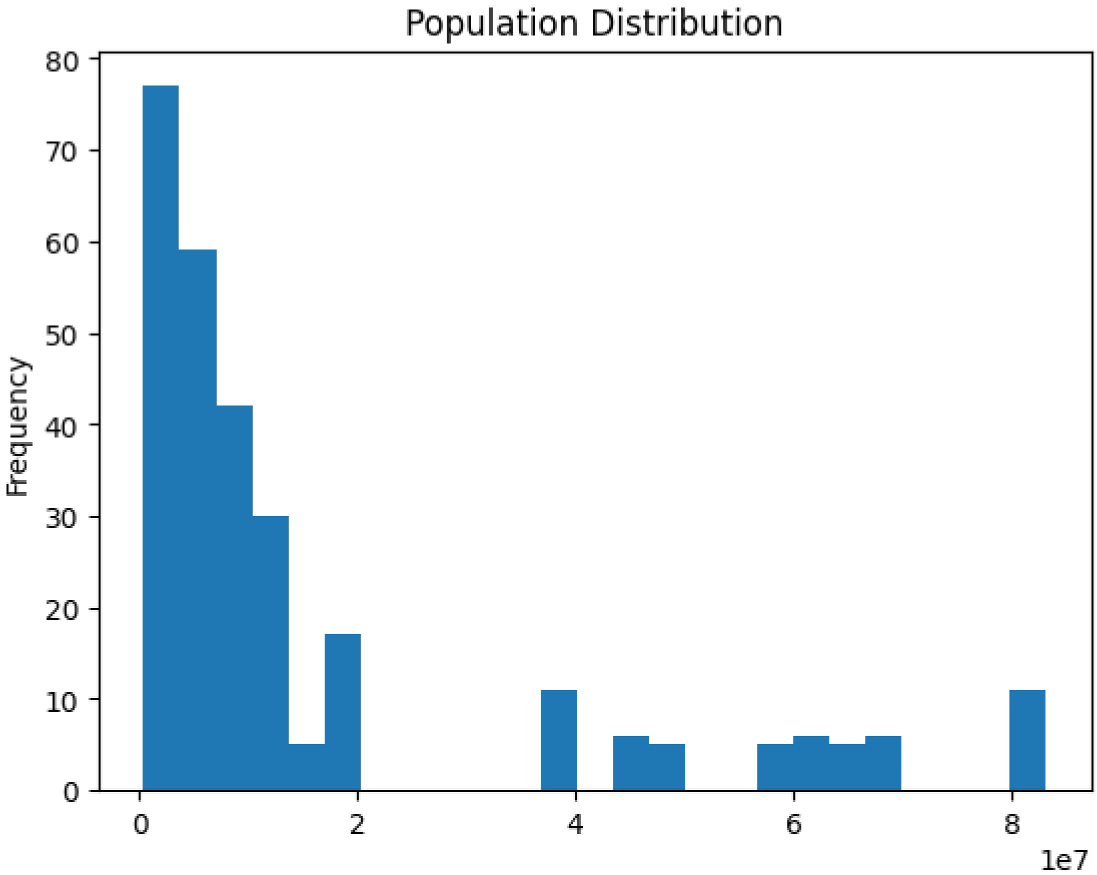
Population

```
In [20]: px.line(df_euro_data, x="Year", y="Population", color="geo", markers=True, title="Popu
                 labels={"Population": "Population (Number of People)"})
```

Population stays relatively steady

```
In [38]: df_euro_data_pcap['Population'].plot.hist(xlabel="Population (Number of People)", titl
```

```
Out[38]: <Axes: title={'center': 'Population Distribution'}, ylabel='Frequency'>
```



Large differences in population due to country size.

Housing Index

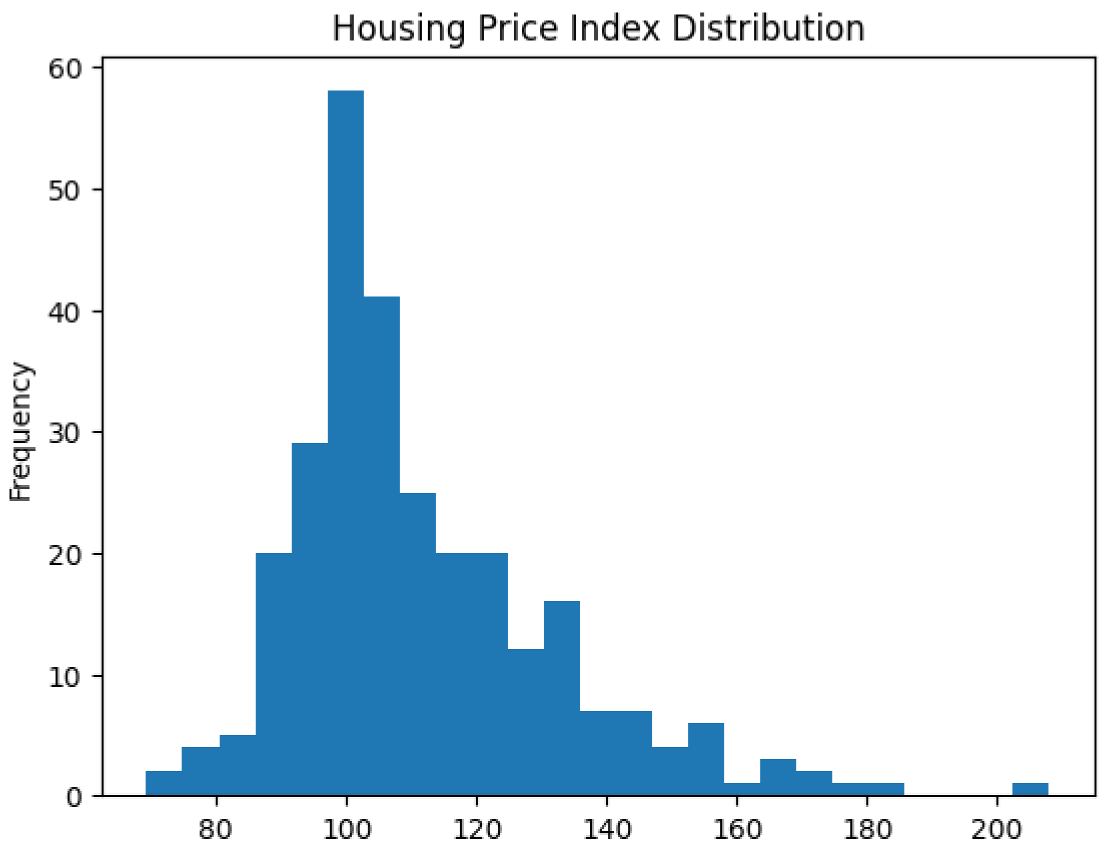
In [21]:

```
px.line(df_euro_data, x="Year", y="Housing Index", color="geo", markers=True, title="H
        labels={"Housing Index": "Housing Price Index (2015 = 100)"})
```

Hungary's doubled in 10 years!

```
In [41]: df_euro_data_pcap['Housing Index'].plot.hist(xlabel="Housing Index", title="Housing Pr
```

```
Out[41]: <Axes: title={'center': 'Housing Price Index Distribution'}, ylabel='Frequency'>
```



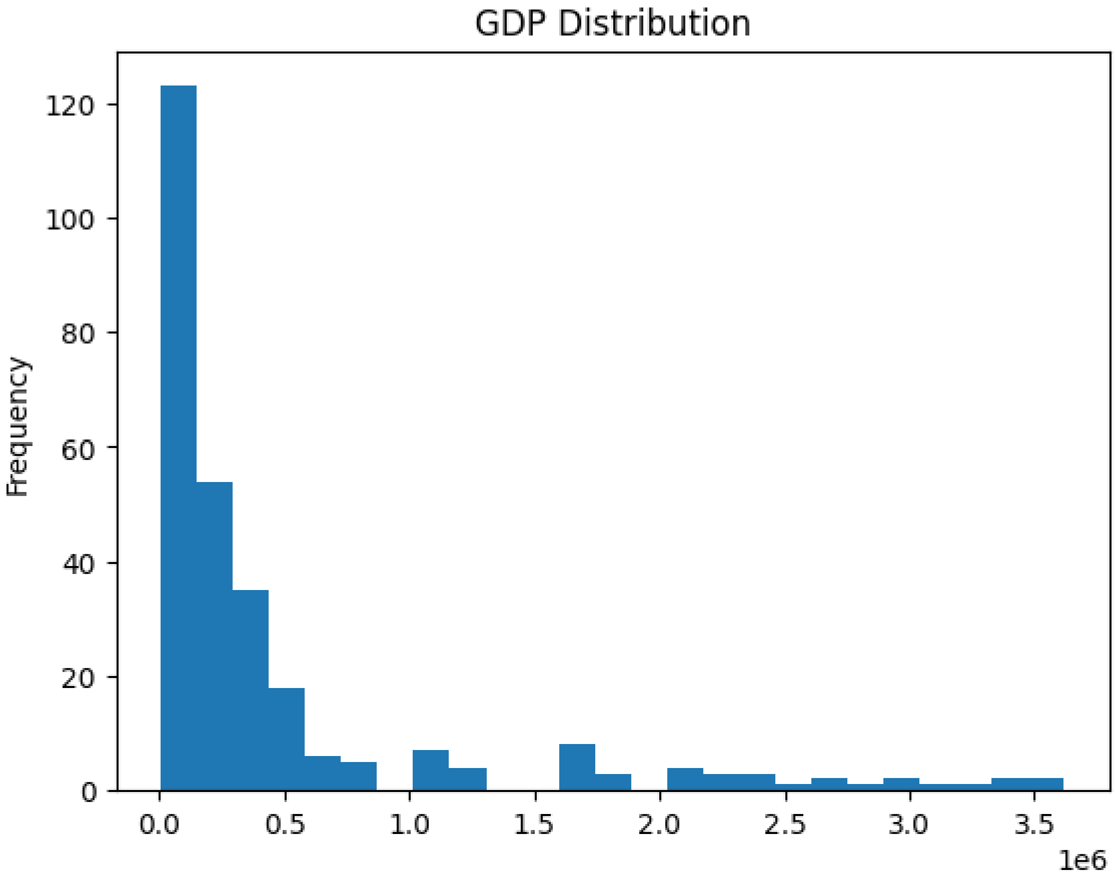
GDP

In [22]:

```
px.line(df_euro_data, x="Year", y="GDP", color="geo", markers=True, title="GDP Over Ti
        labels={"GDP": "GDP (million Euros)"})
```

```
In [42]: df_euro_data_pcap['GDP'].plot.hist(xlabel="GDP", title="GDP Distribution", bins=25)
```

```
Out[42]: <Axes: title={'center': 'GDP Distribution'}, ylabel='Frequency'>
```

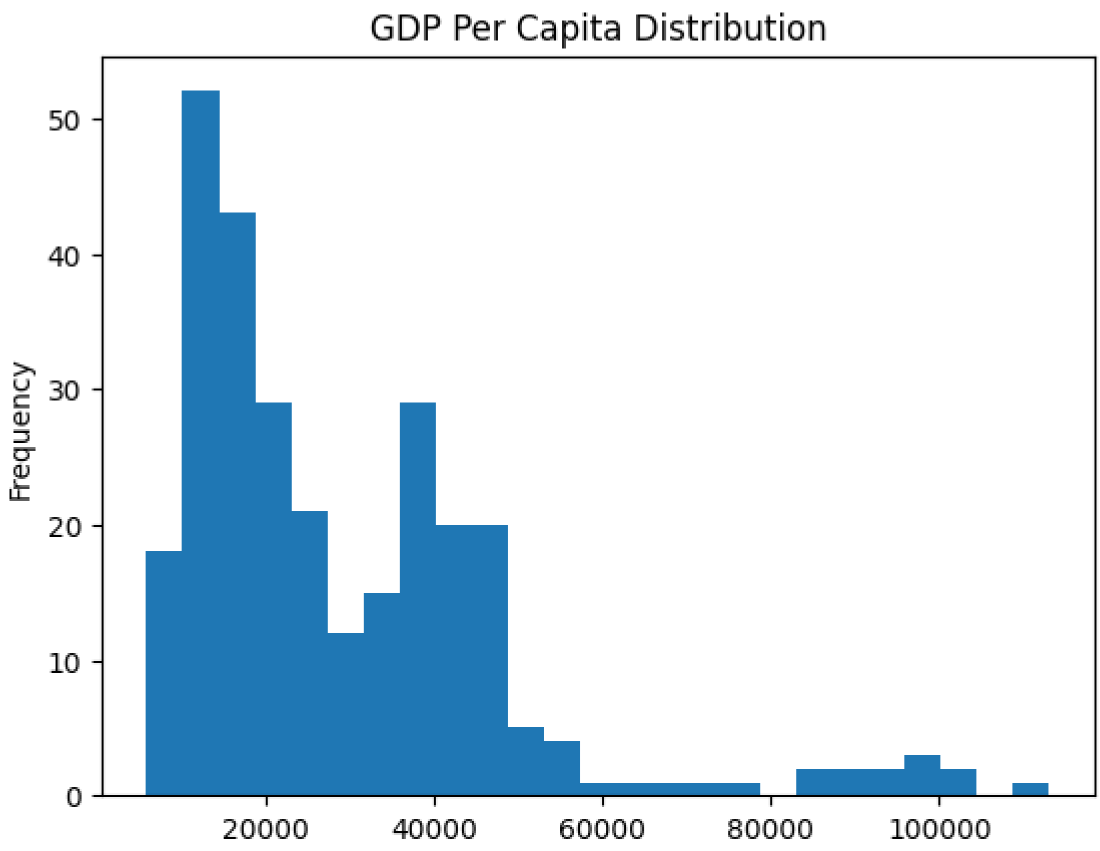


Notice that in the per capita graph, the measurement is in Euros, not million Euros

```
In [23]: px.line(df_euro_data_pcap, x="Year", y="GDP_pcap", color="geo", markers=True, title="
                 labels={"GDP_pcap": "GDP per Capita (Euros)"})
```

```
In [47]: df_euro_data_pcap['GDP_pcap'].plot.hist(xlabel="GDP per Capita", title="GDP Per Capita
```

```
Out[47]: <Axes: title={'center': 'GDP Per Capita Distribution'}, ylabel='Frequency'>
```



Emigration

```
In [24]: px.line(df_euro_data, x="Year", y="emigration", color="geo", markers=True, title="Emig
                 labels={"emigration": "Number of Emigrants"})
```

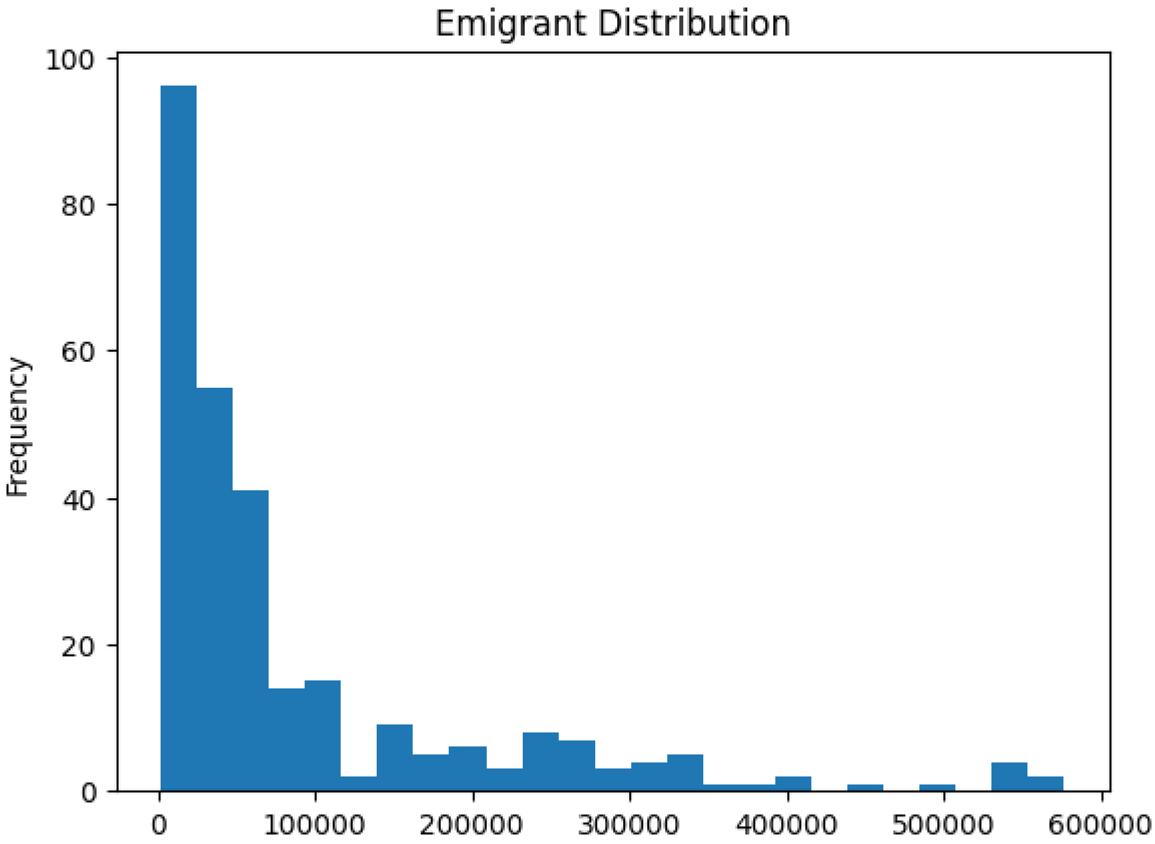
Some countries have a lot of fluctuation while others are consistent, interesting...

In [43]:

```
df_euro_data_pcap['emigration'].plot.hist(xlabel="Emigrants", title="Emigrant Distribu
```

Out[43]:

```
<Axes: title={'center': 'Emigrant Distribution'}, ylabel='Frequency'>
```



In [25]:

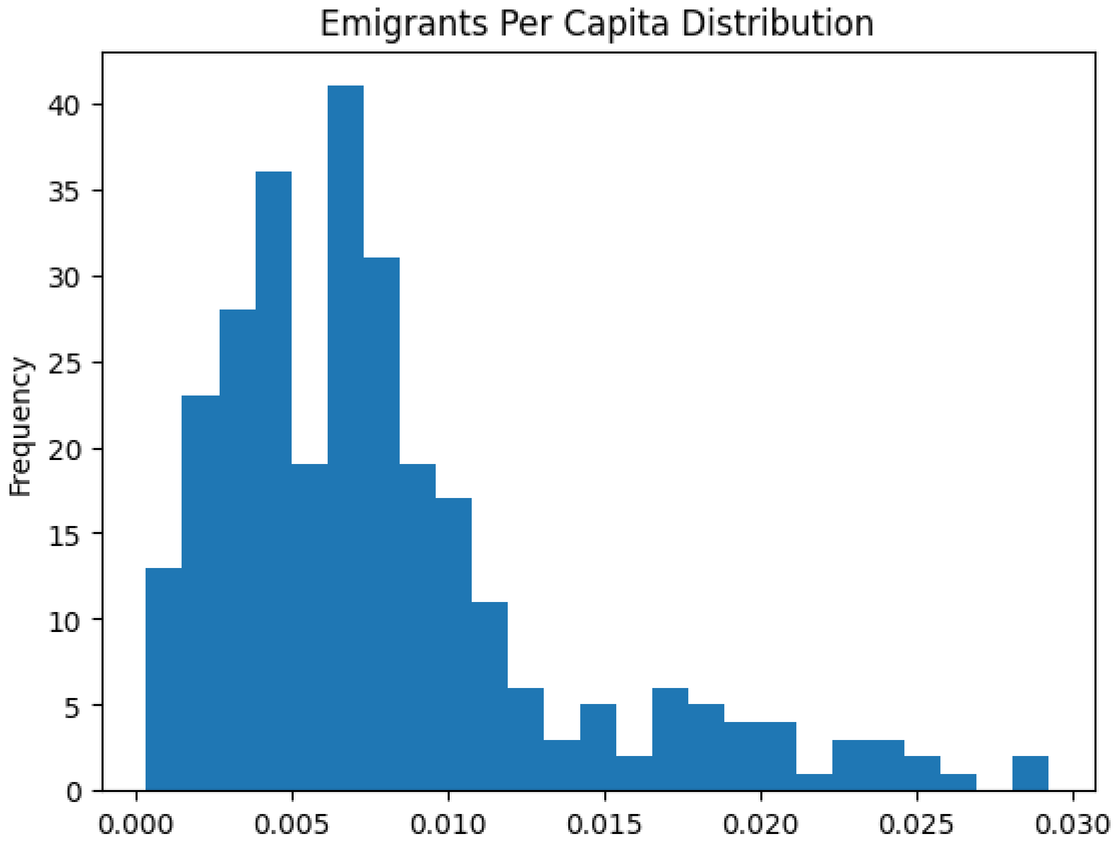
```
px.line(df_euro_data_pcap, x="Year", y="Emigrants_pcap", color="geo", markers=True, ti
        labels={"Emigrants_pcap": "Emigrants per Capita"})
```

In [49]:

```
df_euro_data_pcap['Emigrants_pcap'].plot.hist(xlabel="Emigrants per Capita", title="Em
```

Out[49]:

```
<Axes: title={'center': 'Emigrants Per Capita Distribution'}, ylabel='Frequency'>
```



Unemployment

In [26]:

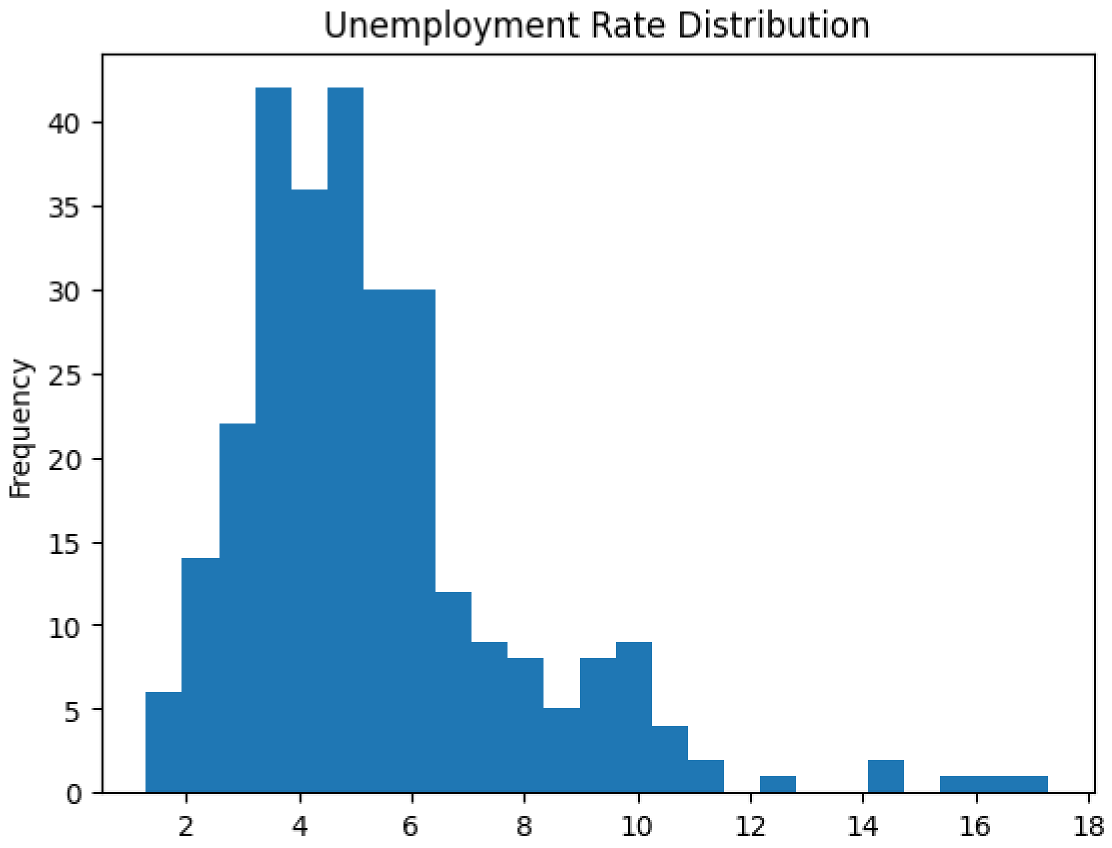
```
px.line(df_euro_data, x="Year", y="unemployment", color="geo", markers=True, title="Ur
        labels={"unemployment": "Unemployment Rate"})
```

In [44]:

```python
df_euro_data_pcap['unemployment'].plot.hist(xlabel="Unemployment Rate", title="Unemplo
```

Out[44]:

```
<Axes: title={'center': 'Unemployment Rate Distribution'}, ylabel='Frequency'>
```



Deaths

```
In [27]: px.line(df_euro_data, x="Year", y="total_deaths", color="geo", markers=True, title="De
                 labels={"total_deaths": "Number of Deaths"})
```

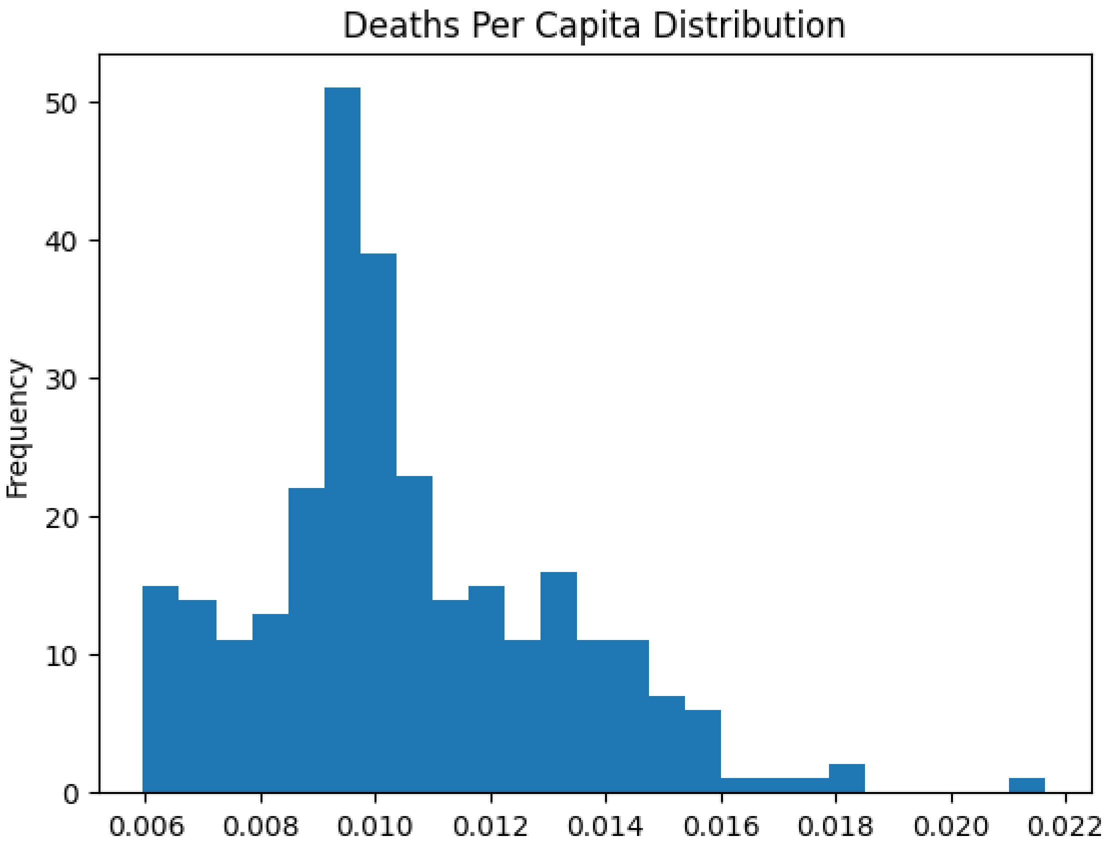
Included this feature because it was also in the population data set.

In [28]:

```
px.line(df_euro_data_pcap, x="Year", y="Deaths_pcap", color="geo", markers=True, title
        labels={"Deaths_pcap": "Deaths per Capita"})
```

```
In [50]: df_euro_data_pcap['Deaths_pcap'].plot.hist(xlabel="Deaths per Capita", title="Deaths P
```

```
Out[50]: <Axes: title={'center': 'Deaths Per Capita Distribution'}, ylabel='Frequency'>
```

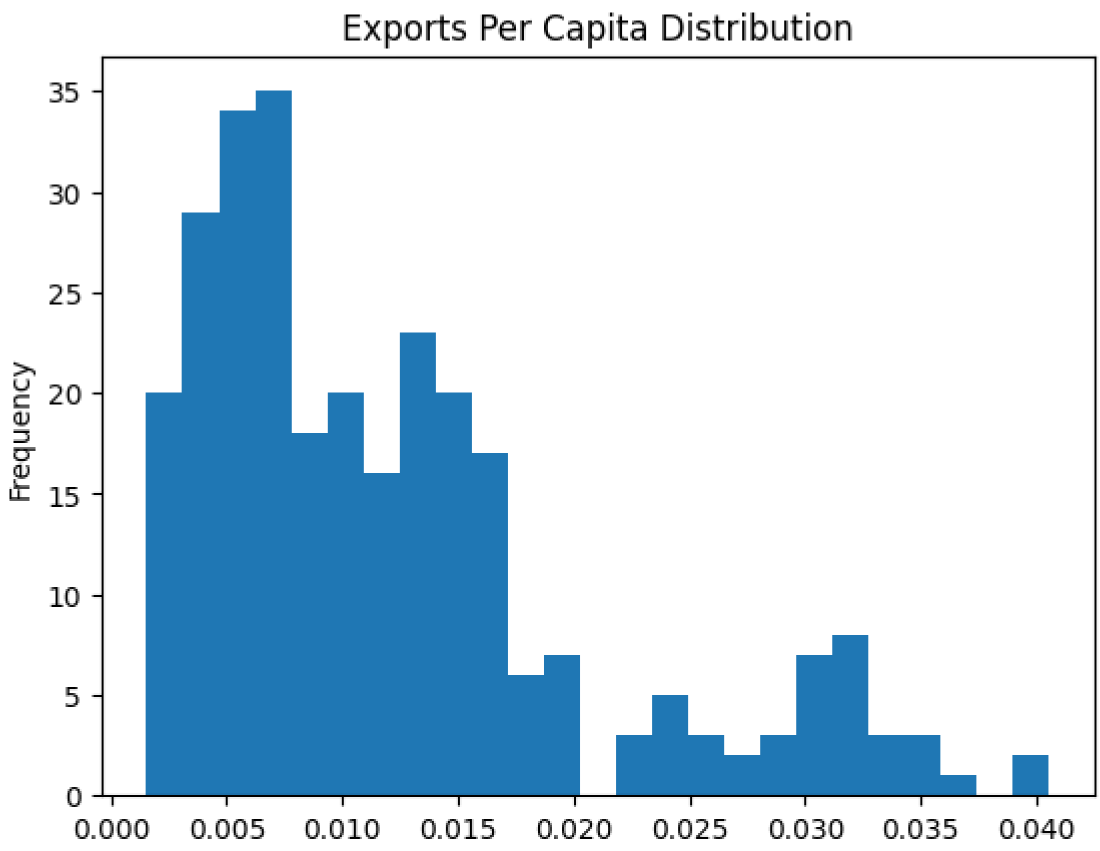


Exports

```
In [29]: px.line(df_euro_data, x="Year", y="Exports", color="geo", markers=True, title="Exports
                 labels={"Exports": "Exports (in million Euros)"})
```

```
In [51]: df_euro_data_pcap['Exports_pcap'].plot.hist(xlabel="Exports_pcap", title="Exports Per
```

```
Out[51]: <Axes: title={'center': 'Exports Per Capita Distribution'}, ylabel='Frequency'>
```



In [30]:
```
px.line(df_euro_data_pcap, x="Year", y="Exports_pcap", color="geo", markers=True, titl
        labels={"Exports_pcap": "Exports (in million Euros) per capita"})
```

Imports

In [31]:

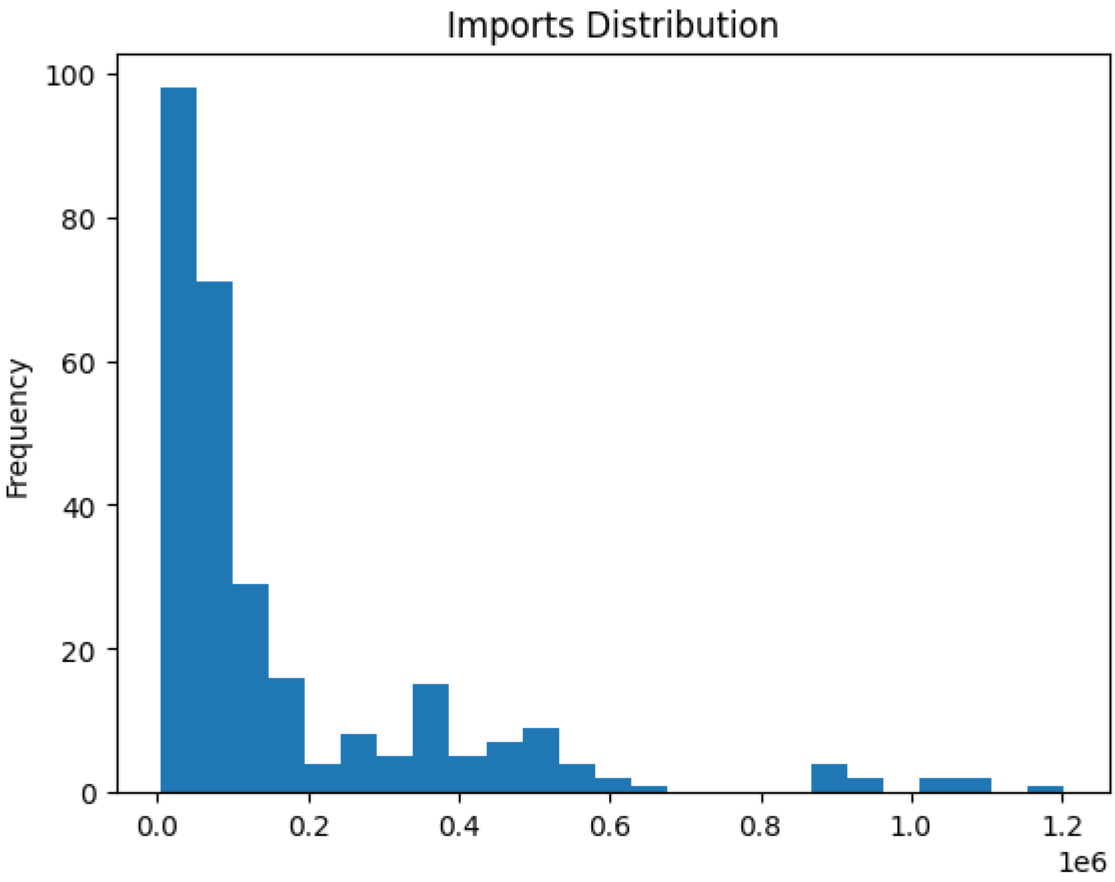
```
px.line(df_euro_data, x="Year", y="Imports", color="geo", markers=True, title="Imports
        labels={"Imports": "Imports (in million Euros)"})
```

In [46]:

```
df_euro_data_pcap['Imports'].plot.hist(xlabel="Imports", title="Imports Distribution",
```

Out[46]:

```
<Axes: title={'center': 'Imports Distribution'}, ylabel='Frequency'>
```



```
In [32]: px.line(df_euro_data_pcap, x="Year", y="Imports_pcap", color="geo", markers=True, titl
                 labels={"Imports_pcap": "Imports (in million Euros) per capita"})
```

In [ ]:

```
```