```
# Import necessary libraries
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import plotly.express as px
import plotly.graph_objects as go
from plotly.subplots import make_subplots
import ipywidgets as widgets  # Make sure this line is included
from IPython.display import display

sns.set(style="whitegrid")
%matplotlib inline
```

```
# Load the dataset
df = pd.read_csv('/content/World Energy Consumption.csv')
```

```
# print shape, size, keys
print(df.shape )
print(df.size)
print(df.keys())
```

```
(17432, 122)
2126704
Index(['iso_code', 'country', 'year', 'coal_prod_change_pct',
       'coal_prod_change_twh', 'gas_prod_change_pct', 'gas_prod_change_twh',
       'oil_prod_change_pct', 'oil_prod_change_twh', 'energy_cons_change_pct',
       ...
       'solar_elec_per_capita', 'solar_energy_per_capita', 'gdp',
       'wind_share_elec', 'wind_cons_change_pct', 'wind_share_energy',
       'wind_cons_change_twh', 'wind_consumption', 'wind_elec_per_capita',
       'wind_energy_per_capita'],
      dtype='object', length=122)
```

```
print("Information of the dataset: \n",df.info())
print("After removing duplicates: \n",df.drop_duplicates(inplace=True)) #cleaning the data by checking the dupl
print("Size of the dataset after removing duplicates: \n",df.size)
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 17432 entries, 0 to 17431
Columns: 122 entries, iso_code to wind_energy_per_capita
dtypes: float64(119), int64(1), object(2)
memory usage: 16.2+ MB
Information of the dataset:
 None
After removing duplicates:
 None
Size of the dataset after removing duplicates:
 2126704
```

```
df.describe()
```

	year	coal_prod_change_pct	coal_prod_change_twh	gas_prod_change_pct	gas_prod_change_twh	oil_prod_change
count	17432.000000	7445.000000	10394.000000	4.862000e+03	7893.000000	6521.00
mean	1973.094367	20.830774	8.798102	1.921623e+14	14.369018	18.24
std	34.333995	697.178744	135.503698	1.339910e+16	85.415649	335.49
min	1900.000000	-100.000000	-2326.870000	-1.000000e+02	-1054.320000	-100.00
25%	1946.000000	-1.532000	0.000000	0.000000e+00	0.000000	-1.42
50%	1983.000000	0.000000	0.000000	2.583500e+00	0.000000	0.27
75%	2002.000000	7.690000	0.334000	9.703500e+00	2.559000	9.09
max	2020.000000	44965.754000	3060.593000	9.342930e+17	2112.975000	25500.00

8 rows × 120 columns

```
print(df.shape )
print(df.size)
print(df.keys())
```

```
(17432, 122)
2126704
Index(['iso_code', 'country', 'year', 'coal_prod_change_pct',
       'coal_prod_change_twh', 'gas_prod_change_pct', 'gas_prod_change_twh',
       'oil_prod_change_pct', 'oil_prod_change_twh', 'energy_cons_change_pct',
       ...
       'solar_elec_per_capita', 'solar_energy_per_capita', 'gdp',
       'wind_share_elec', 'wind_cons_change_pct', 'wind_share_energy',
       'wind_cons_change_twh', 'wind_consumption', 'wind_elec_per_capita',
       'wind_energy_per_capita'],
      dtype='object', length=122)
```

```
# Data Preparation: Handle missing values and clean data
df.drop_duplicates(inplace=True)
numeric_cols = df.select_dtypes(include=np.number).columns
df[numeric_cols] = df[numeric_cols].fillna(df[numeric_cols].mean())
# Drop rows with any null or empty values
df.dropna(inplace=True)
```

```
# print shape, size, keys
print(df.shape )
print(df.size)
print(df.keys())
df.describe()
```

```
(15630, 122)
1906860
Index(['iso_code', 'country', 'year', 'coal_prod_change_pct',
       'coal_prod_change_twh', 'gas_prod_change_pct', 'gas_prod_change_twh',
       'oil_prod_change_pct', 'oil_prod_change_twh', 'energy_cons_change_pct',
       ...
       'solar_elec_per_capita', 'solar_energy_per_capita', 'gdp',
       'wind_share_elec', 'wind_cons_change_pct', 'wind_share_energy',
       'wind_cons_change_twh', 'wind_consumption', 'wind_elec_per_capita',
       'wind_energy_per_capita'],
      dtype='object', length=122)
/usr/local/lib/python3.10/dist-packages/numpy/lib/function_base.py:4655: RuntimeWarning: invalid value encountered in subt
  diff_b_a = subtract(b, a)
```

	year	coal_prod_change_pct	coal_prod_change_twh	gas_prod_change_pct	gas_prod_change_twh	oil_prod_change
count	15630.000000	15630.000000	15630.000000	1.563000e+04	15630.000000	15630.00
mean	1972.652271	21.900332	7.458285	2.018257e+14	12.154091	18.88
std	34.687517	481.035771	85.272762	7.472488e+15	51.444414	216.44
min	1900.000000	-100.000000	-2326.870000	-1.000000e+02	-944.242000	-100.00
25%	1944.000000	2.366000	0.000000	5.536425e+01	0.000000	6.59
50%	1983.000000	20.830774	1.303000	1.921623e+14	14.369018	18.24
75%	2002.000000	20.830774	8.798102	1.921623e+14	14.369018	18.24
max	2020.000000	44965.754000	3060.593000	9.342930e+17	2112.975000	25500.00

8 rows × 120 columns

```
df.head(10)
```

	iso_code	country	year	coal_prod_change_pct	coal_prod_change_twh	gas_prod_change_pct	gas_prod_change_twh	oil_p
0	AFG	Afghanistan	1900	20.830774	8.798102	1.921623e+14	14.369018	
1	AFG	Afghanistan	1901	20.830774	0.000000	1.921623e+14	14.369018	
2	AFG	Afghanistan	1902	20.830774	0.000000	1.921623e+14	14.369018	
3	AFG	Afghanistan	1903	20.830774	0.000000	1.921623e+14	14.369018	
4	AFG	Afghanistan	1904	20.830774	0.000000	1.921623e+14	14.369018	
5	AFG	Afghanistan	1905	20.830774	0.000000	1.921623e+14	14.369018	
6	AFG	Afghanistan	1906	20.830774	0.000000	1.921623e+14	14.369018	
7	AFG	Afghanistan	1907	20.830774	0.000000	1.921623e+14	14.369018	
8	AFG	Afghanistan	1908	20.830774	0.000000	1.921623e+14	14.369018	
9	AFG	Afghanistan	1909	20.830774	0.000000	1.921623e+14	14.369018	

10 rows × 122 columns

```
df.isnull().sum()
```

	0
iso_code	0
country	0
year	0
coal_prod_change_pct	0
coal_prod_change_twh	0
...	...
wind_share_energy	0
wind_cons_change_twh	0
wind_consumption	0
wind_elec_per_capita	0
wind_energy_per_capita	0

122 rows × 1 columns

```
numerical_cols = df.select_dtypes(include=['float64', 'int64']).columns
for col in numerical_cols:
    df[col].fillna(df[col].mean(), inplace=True)  # You can also use median with `data[col].median()
```

```
<ipython-input-12-0e98a4dc3074>:3: FutureWarning: A value is trying to be set on a copy of a DataFrame or Series through c
The behavior will change in pandas 3.0. This inplace method will never work because the intermediate object on which we ar

For example, when doing 'df[col].method(value, inplace=True)', try using 'df.method({col: value}, inplace=True)' or df[col


  df[col].fillna(df[col].mean(), inplace=True)  # You can also use median with `data[col].median()
```

```
df.drop_duplicates(inplace=True)
```

```
# print shape, size, keys
print(df.shape )
print(df.size)
print(df.keys())
df.describe()
```

```
(15630, 122)
1906860
Index(['iso_code', 'country', 'year', 'coal_prod_change_pct',
       'coal_prod_change_twh', 'gas_prod_change_pct', 'gas_prod_change_twh',
       'oil_prod_change_pct', 'oil_prod_change_twh', 'energy_cons_change_pct',
       ...
       'solar_elec_per_capita', 'solar_energy_per_capita', 'gdp',
       'wind_share_elec', 'wind_cons_change_pct', 'wind_share_energy',
       'wind_cons_change_twh', 'wind_consumption', 'wind_elec_per_capita',
       'wind_energy_per_capita'],
      dtype='object', length=122)
/usr/local/lib/python3.10/dist-packages/numpy/lib/function_base.py:4655: RuntimeWarning: invalid value encountered in subt
  diff_b_a = subtract(b, a)
```

	year	coal_prod_change_pct	coal_prod_change_twh	gas_prod_change_pct	gas_prod_change_twh	oil_prod_change
count	15630.000000	15630.000000	15630.000000	1.563000e+04	15630.000000	15630.00
mean	1972.652271	21.900332	7.458285	2.018257e+14	12.154091	18.88
std	34.687517	481.035771	85.272762	7.472488e+15	51.444414	216.44
min	1900.000000	-100.000000	-2326.870000	-1.000000e+02	-944.242000	-100.00
25%	1944.000000	2.366000	0.000000	5.536425e+01	0.000000	6.59
50%	1983.000000	20.830774	1.303000	1.921623e+14	14.369018	18.24
75%	2002.000000	20.830774	8.798102	1.921623e+14	14.369018	18.24
max	2020.000000	44965.754000	3060.593000	9.342930e+17	2112.975000	25500.00

8 rows × 120 columns

```
# Save the cleaned data into a new CSV file
cleaned_file_path = '/content/Cleaned_World_Energy_Consumption.csv'  # Define your desired file path
df.to_csv(cleaned_file_path, index=False)  # index=False to avoid saving the row indices
```

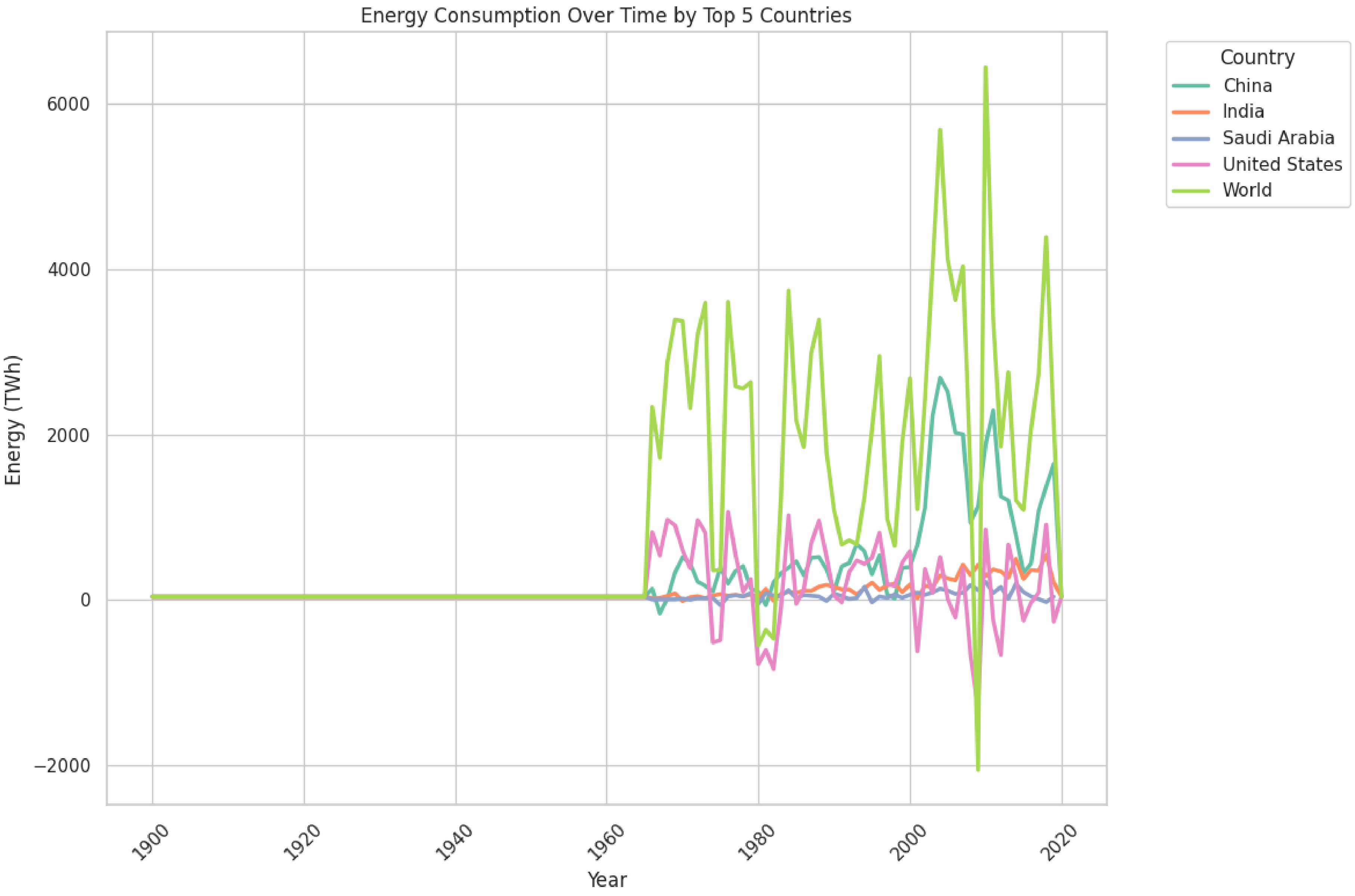
```
# Load the dataset
df = pd.read_csv('/content/Cleaned_World_Energy_Consumption.csv')
```

```
# Example for a column named 'Energy_Exajoules'
if 'Energy_Exajoules' in df.columns:
    df['Energy_TWh'] = df['Energy_Exajoules'] * 277.8
```

```
if 'Population' in df.columns and 'Energy_TWh' in df.columns:
    df['Energy_per_Capita'] = df['Energy_TWh'] / df['Population']
```

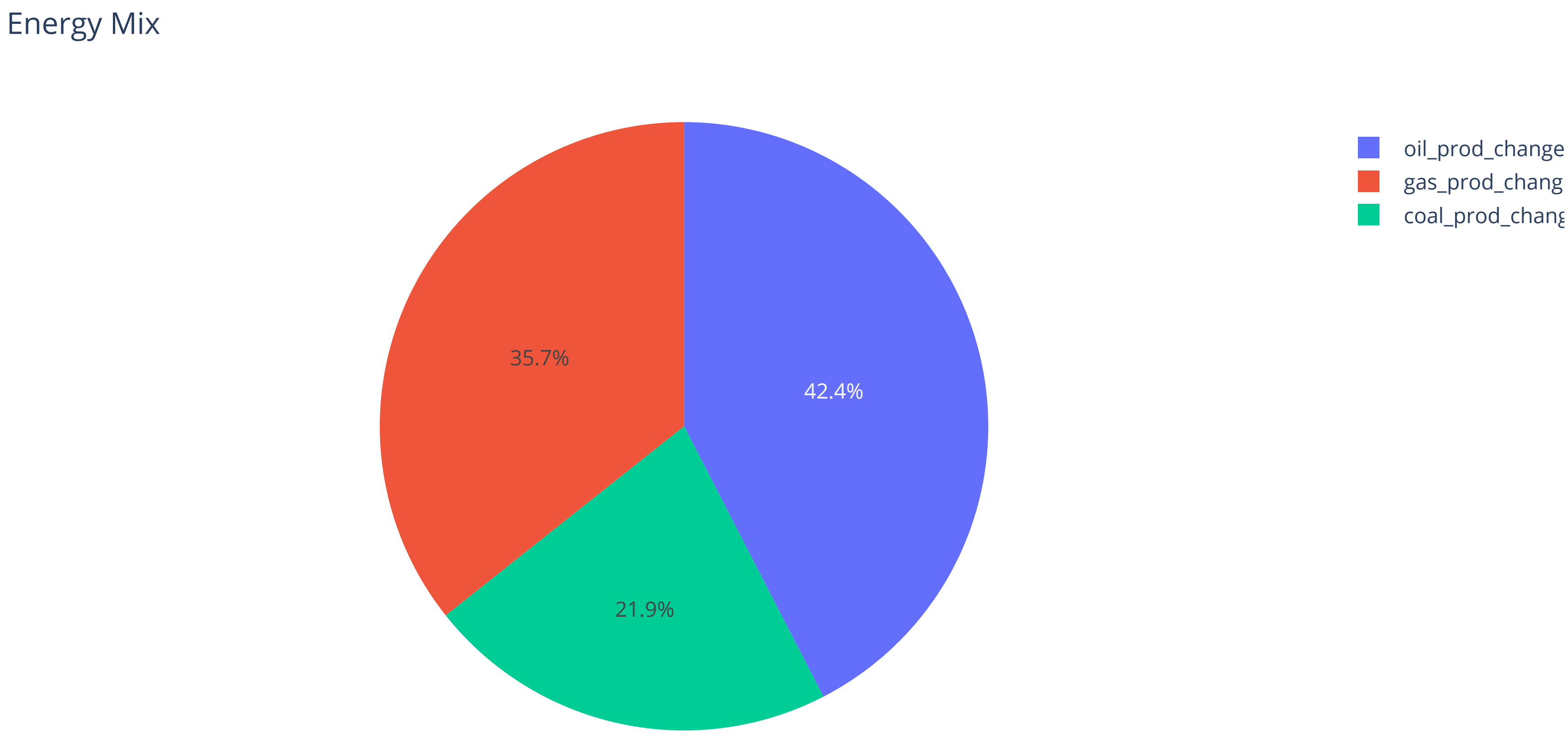
Visualize Overall Energy Consumption Trends

```
# Select a subset of countries for better readability (e.g., top 5 by average energy consumption)
top_countries = df.groupby('country')['energy_cons_change_twh'].mean().nlargest(5).index
subset_df = df[df['country'].isin(top_countries)]

plt.figure(figsize=(12, 8))
sns.set_style("whitegrid")
sns.set_palette("Set2")  # Use a color palette that is visually distinct

# Plot the data
sns.lineplot(data=subset_df, x='year', y='energy_cons_change_twh', hue='country', linewidth=2.5)
plt.title("Energy Consumption Over Time by Top 5 Countries")
plt.xlabel("Year")
plt.ylabel("Energy (TWh)")
plt.xticks(rotation=45)
plt.legend(title="Country", bbox_to_anchor=(1.05, 1), loc='upper left')
plt.tight_layout()
plt.show()
```


Energy Consumption Over Time by Top 5 Countries
Energy (TWh)
6000
4000
2000
0
−2000
Year
1900
1920
1940
1960
1980
2000
2020
Country
China
India
Saudi Arabia
United States
World

Electricity vs. Energy Mix

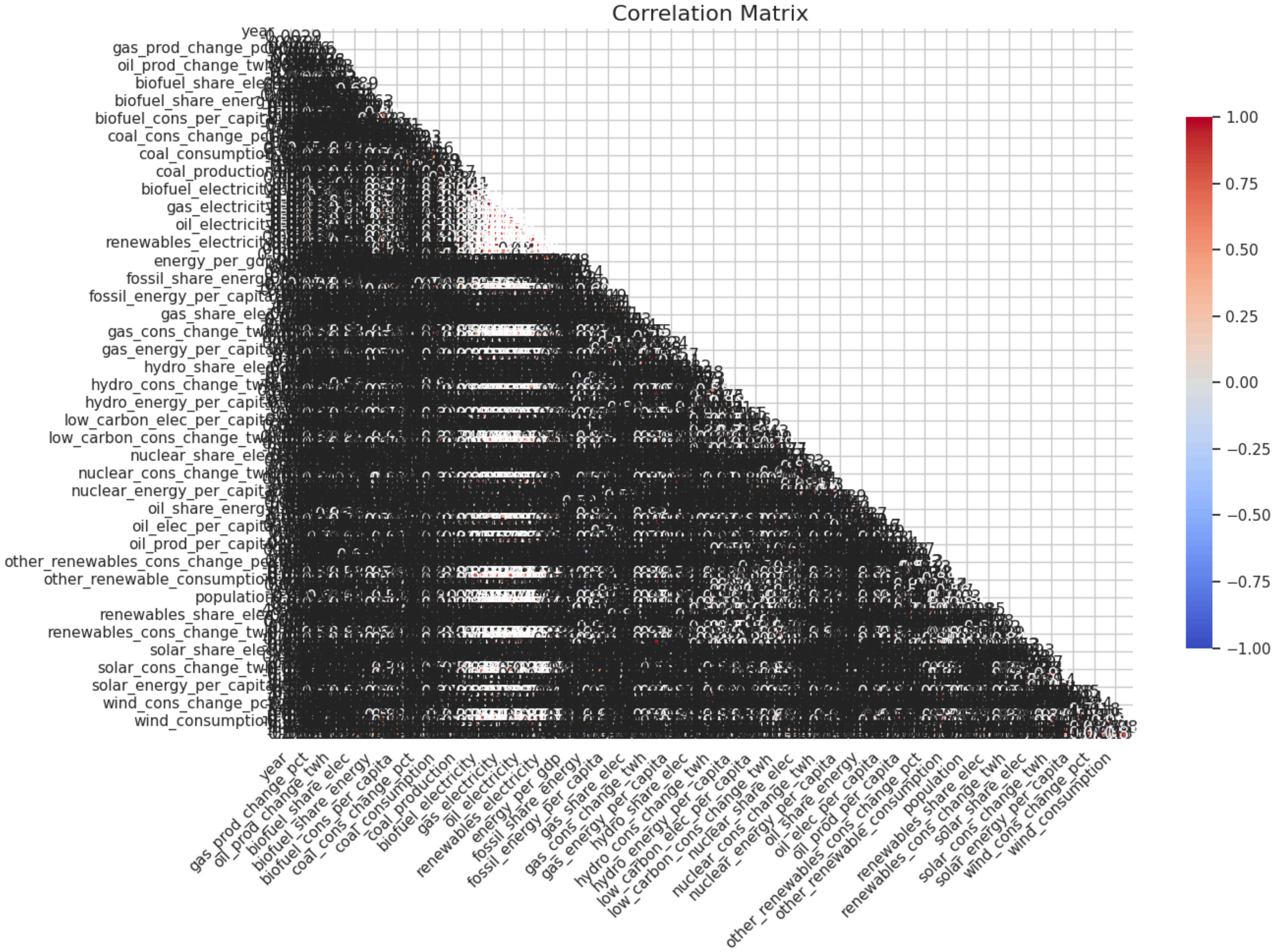
```
# Example of how to create a summary DataFrame
energy_types = ['coal_prod_change_twh', 'gas_prod_change_twh', 'oil_prod_change_twh']  # Add more energy types
summary_df = df[energy_types].sum().reset_index()
summary_df.columns = ['Energy_Type', 'Energy_Mix_Column']

# Create the pie chart
fig = px.pie(summary_df, values='Energy_Mix_Column', names='Energy_Type', title="Energy Mix")
# Display the chart
fig.show()
```



Correlation Analysis

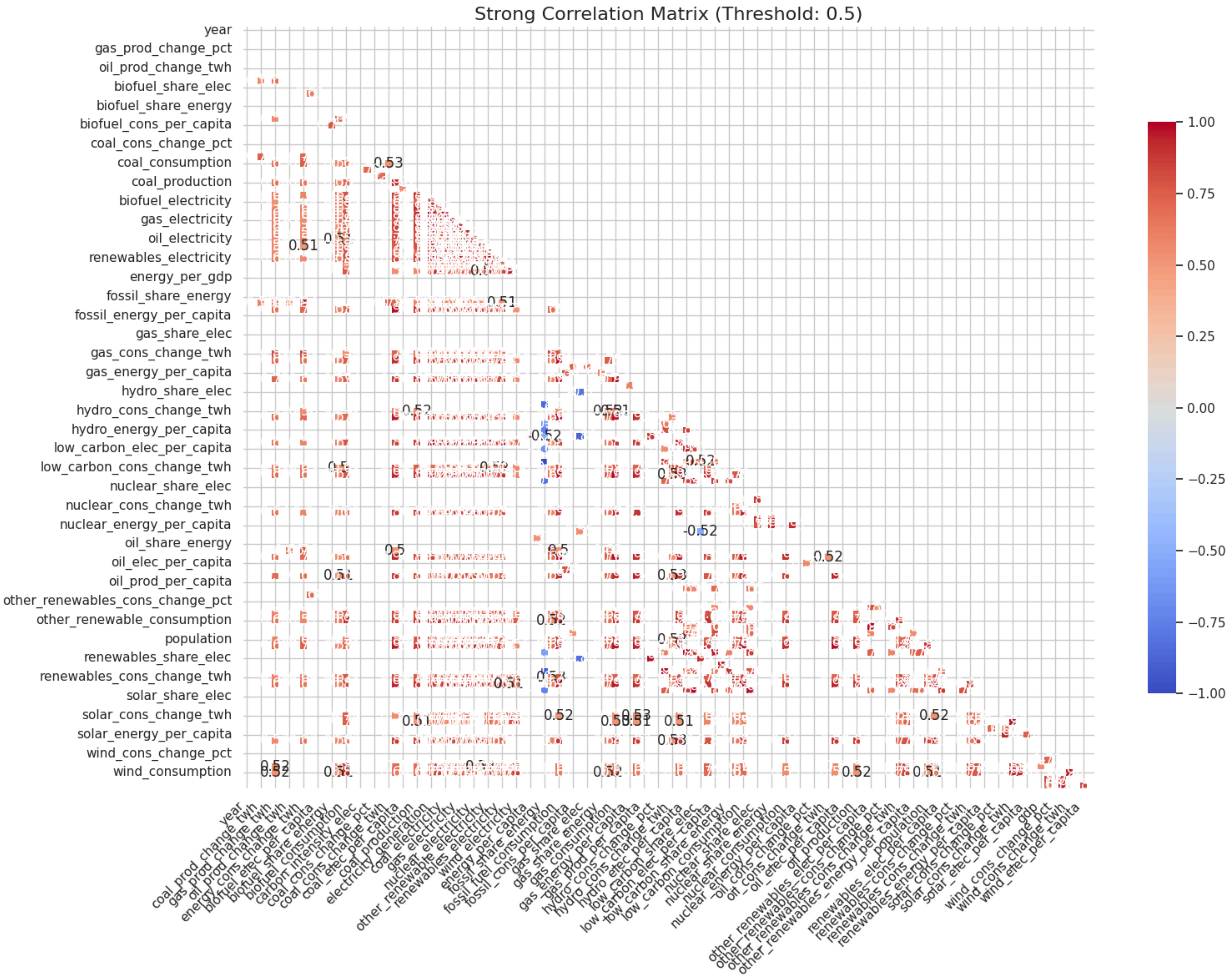
```
# Select only numeric columns
numeric_df = df.select_dtypes(include=['float64', 'int64'])

# Create the correlation matrix
corr_matrix = numeric_df.corr()
```

```
# Create a mask for the upper triangle (since correlation matrix is symmetric)
mask = np.triu(np.ones_like(corr_matrix, dtype=bool))

# Plot the heatmap
plt.figure(figsize=(14, 10))
sns.heatmap(corr_matrix, annot=True, cmap='coolwarm', mask=mask,
            vmin=-1, vmax=1, cbar_kws={'shrink': 0.75})
plt.title("Correlation Matrix", fontsize=16)
plt.xticks(rotation=45, ha='right')
plt.yticks(rotation=0)
plt.tight_layout()  # Adjust layout for better spacing
plt.show()
```



Correlation Matrix
year
gas_prod_change_pct
oil_prod_change_twh
biofuel_share_elec
biofuel_share_energy
biofuel_cons_per_capita
coal_cons_change_pct
coal_consumption
coal_production
biofuel_electricity
gas_electricity
oil_electricity
renewables_electricity
energy_per_gdp
fossil_share_energy
fossil_energy_per_capita
gas_share_elec
gas_cons_change_twh
gas_energy_per_capita
hydro_share_elec
hydro_cons_change_twh
hydro_energy_per_capita
low_carbon_elec_per_capita
low_carbon_cons_change_twh
nuclear_share_elec
nuclear_cons_change_twh
nuclear_energy_per_capita
oil_share_energy
oil_elec_per_capita
oil_prod_per_capita
other_renewables_cons_change_pct
other_renewable_consumption
population
renewables_share_elec
renewables_cons_change_twh
solar_share_elec
solar_cons_change_twh
solar_energy_per_capita
wind_cons_change_pct
wind_consumption
1.00
0.75
0.50
0.25
0.00
−0.25
−0.50
−0.75
−1.00

```
# Define a threshold for strong correlations
threshold = 0.5

# Select only correlations with an absolute value above the threshold
strong_corr = corr_matrix[(corr_matrix >= threshold) | (corr_matrix <= -threshold)]

# Plot the heatmap with strong correlations only
plt.figure(figsize=(16, 12))  # Increase figure size for clarity
mask = np.triu(np.ones_like(strong_corr, dtype=bool))
sns.heatmap(strong_corr, annot=True, cmap='coolwarm', mask=mask, vmin=-1, vmax=1, cbar_kws={'shrink': 0.75})
plt.title("Strong Correlation Matrix (Threshold: 0.5)", fontsize=16)
plt.xticks(rotation=45, ha='right')
plt.yticks(rotation=0)
plt.tight_layout()
plt.show()
```


Strong Correlation Matrix (Threshold: 0.5)
year
gas_prod_change_pct
oil_prod_change_twh
biofuel_share_elec
biofuel_share_energy
biofuel_cons_per_capita
coal_cons_change_pct
coal_consumption
coal_production
biofuel_electricity
gas_electricity
oil_electricity
renewables_electricity
energy_per_gdp
fossil_share_energy
fossil_energy_per_capita
gas_share_elec
gas_cons_change_twh
gas_energy_per_capita
hydro_share_elec
hydro_cons_change_twh
hydro_energy_per_capita
low_carbon_elec_per_capita
low_carbon_cons_change_twh
nuclear_share_elec
nuclear_cons_change_twh
nuclear_energy_per_capita
oil_share_energy
oil_elec_per_capita
oil_prod_per_capita
other_renewables_cons_change_pct
other_renewable_consumption
population
renewables_share_elec
renewables_cons_change_twh
solar_share_elec
solar_cons_change_twh
solar_energy_per_capita
wind_cons_change_pct
wind_consumption
year
coal_prod_change_twh
gas_prod_change_twh
oil_prod_change_twh
energy_cons_change_twh
biofuel_elec_per_capita
biofuel_share_energy
biofuel_consumption
carbon_intensity_elec
coal_cons_change_pct
coal_cons_change_twh
coal_elec_per_capita
coal_production
electricity_generation
coal_electricity
gas_electricity
nuclear_electricity
other_renewable_electricity
renewables_electricity
wind_electricity
energy_per_capita
fossil_share_energy
fossil_fuel_consumption
fossil_cons_per_capita
gas_share_elec
gas_share_energy
gas_consumption
gas_energy_per_capita
gas_prod_per_capita
hydro_cons_change_pct
hydro_cons_change_twh
hydro_elec_per_capita
low_carbon_share_elec
low_carbon_elec_per_capita
low_carbon_share_energy
low_carbon_consumption
nuclear_share_elec
nuclear_share_energy
nuclear_consumption
nuclear_energy_per_capita
oil_cons_change_pct
oil_cons_change_twh
oil_elec_per_capita
oil_production
other_renewables_elec_per_capita
other_renewables_cons_change_pct
other_renewables_cons_change_twh
other_renewables_energy_per_capita
population
renewables_elec_per_capita
renewables_cons_change_pct
renewables_cons_change_twh
renewables_energy_per_capita
solar_cons_change_pct
solar_cons_change_twh
solar_elec_per_capita
gdp
wind_cons_change_pct
wind_cons_change_twh
wind_elec_per_capita
1.00
0.75
0.50
0.25
0.00
−0.25
−0.50
−0.75
−1.00
0.53
0.51
-0.52
0.52
0.5

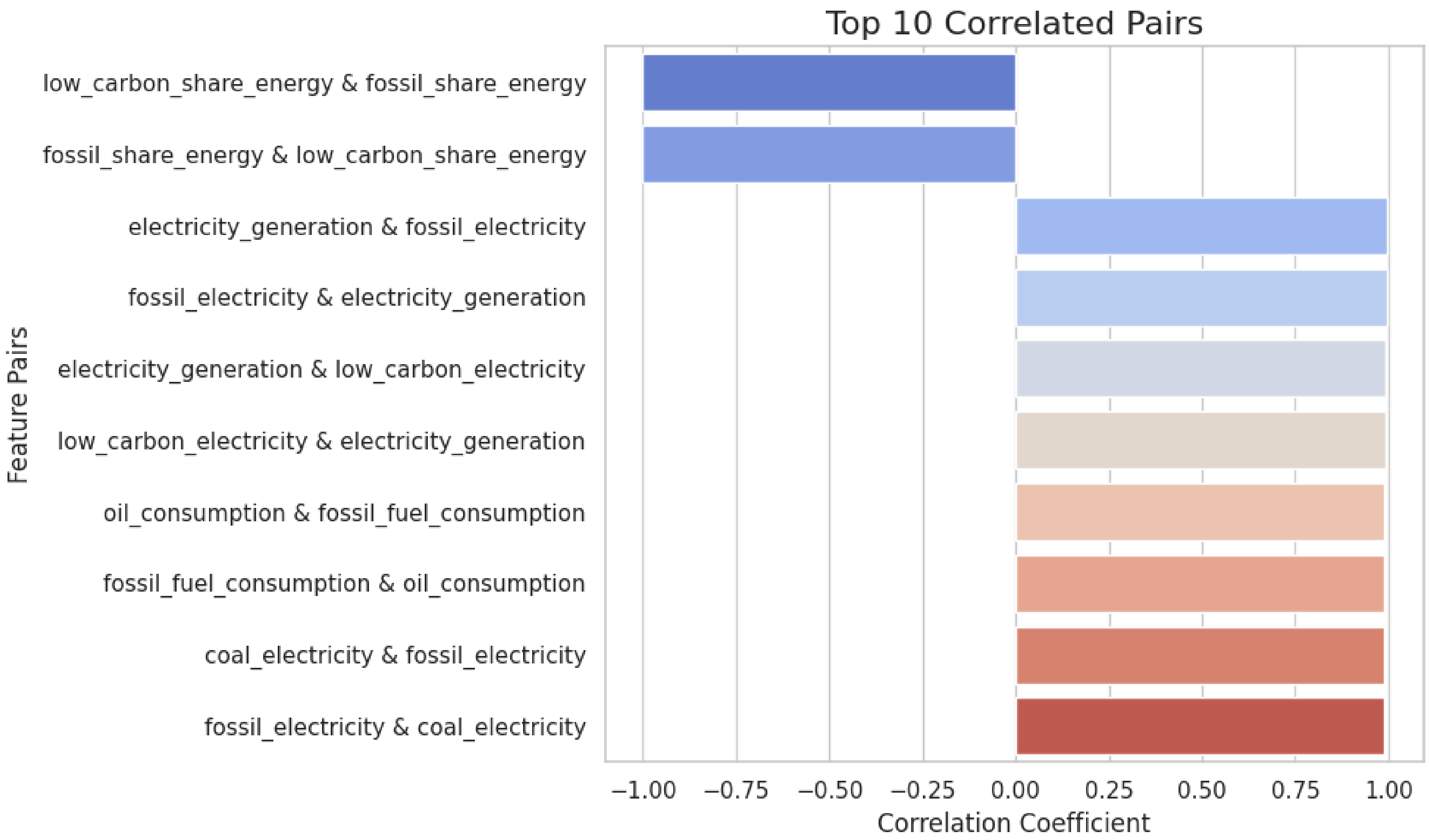
```
# Unstack the correlation matrix and sort by absolute correlation value
corr_unstacked = corr_matrix.unstack().sort_values(ascending=False, key=lambda x: abs(x))

# Filter the top correlations (e.g., the top 10 correlations)
top_corr = corr_unstacked[corr_unstacked != 1].head(10)
print("Top 10 Correlations:\n", top_corr)

# Visualize only the top correlated pairs
# Optional: You can manually select the pairs from this output and visualize them
```

```
Top 10 Correlations:
 low_carbon_share_energy  fossil_share_energy       -0.999847
fossil_share_energy      low_carbon_share_energy   -0.999847
electricity_generation   fossil_electricity         0.995292
fossil_electricity       electricity_generation     0.995292
electricity_generation   low_carbon_electricity     0.993445
low_carbon_electricity   electricity_generation     0.993445
oil_consumption          fossil_fuel_consumption    0.990400
fossil_fuel_consumption  oil_consumption            0.990400
coal_electricity         fossil_electricity         0.989202
fossil_electricity       coal_electricity           0.989202
dtype: float64
```

```
import matplotlib.pyplot as plt
import seaborn as sns

# Unstack the correlation matrix and sort by absolute correlation value
corr_unstacked = corr_matrix.unstack().sort_values(ascending=False, key=lambda x: abs(x))

# Filter the top correlations (excluding self-correlations)
top_corr = corr_unstacked[corr_unstacked != 1].head(10)

# Convert the index (which is a multi-index) to a string for better labeling
```

```
top_corr_pairs = [' & '.join(pair) for pair in top_corr.index]

# Create a bar plot for the top correlations
plt.figure(figsize=(10, 6))
sns.barplot(x=top_corr.values, y=top_corr_pairs, palette='coolwarm')
plt.title("Top 10 Correlated Pairs", fontsize=16)
plt.xlabel("Correlation Coefficient", fontsize=12)
plt.ylabel("Feature Pairs", fontsize=12)
plt.tight_layout()
plt.show()
```

```
<ipython-input-24-0753ba290884>:15: FutureWarning:


Passing `palette` without assigning `hue` is deprecated and will be removed in v0.14.0. Assign the `y` variable to `hue` a
```



Growth Rate Calculation

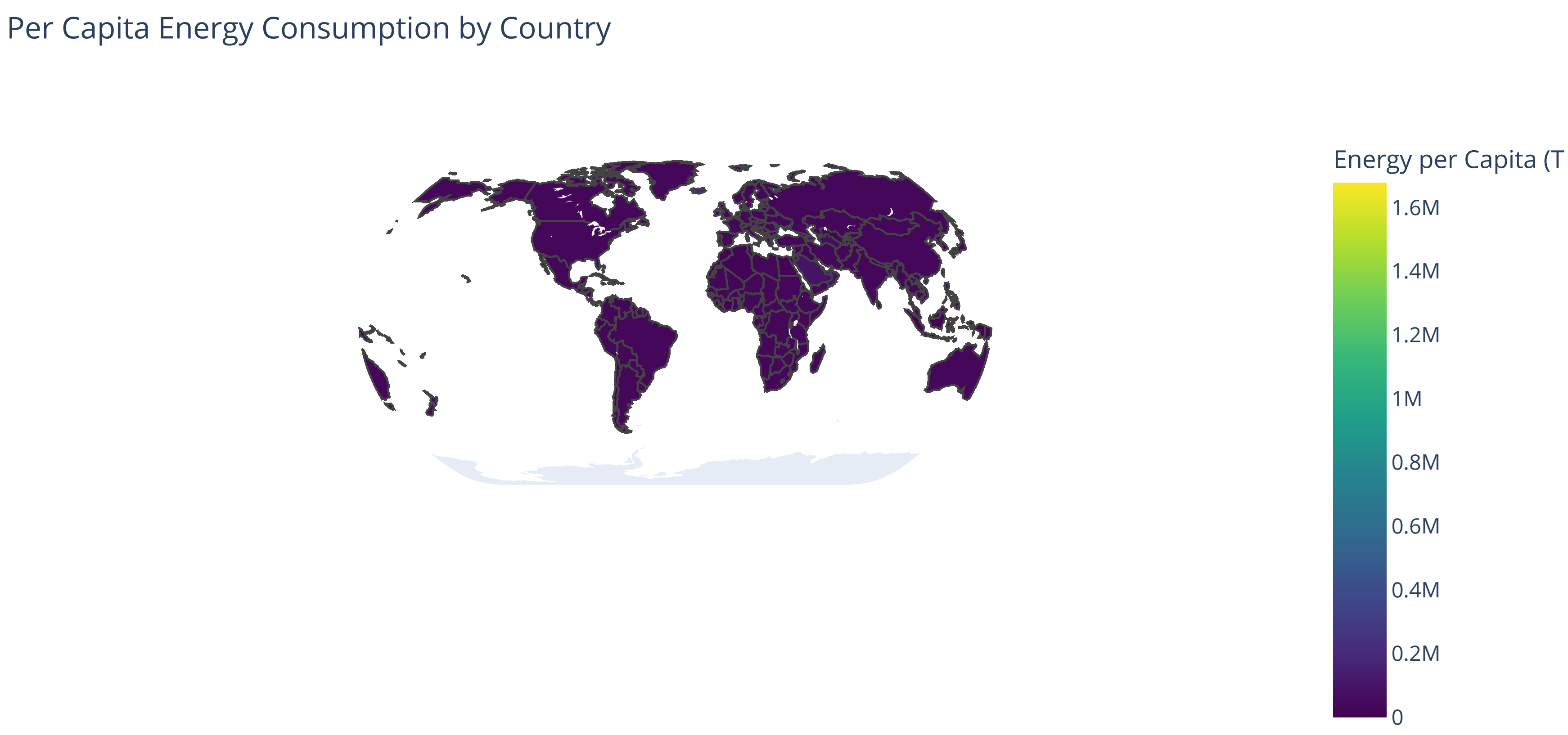
```
# Example: Calculate the growth rate of wind energy consumption in TWh

# Check if 'wind_consumption' exists in the dataset
if 'wind_consumption' in df.columns:
    # Fill missing values in 'wind_consumption' if necessary
    df['wind_consumption'].fillna(method='ffill', inplace=True)  # Forward fill missing values
    df['wind_consumption'].fillna(method='bfill', inplace=True)  # Backward fill remaining NaNs

    # Calculate the growth rate as percentage change in 'wind_consumption'
    df['Wind_Consumption_Growth'] = df['wind_consumption'].pct_change() * 100

    # Display the new growth rate column
    print(df[['wind_consumption', 'Wind_Consumption_Growth']].head())
else:
    print("Error: 'wind_consumption' column does not exist.")
```

```
   wind_consumption  Wind_Consumption_Growth
0         15.080935                      NaN
1         15.080935                      0.0
2         15.080935                      0.0
3         15.080935                      0.0
4         15.080935                      0.0
<ipython-input-26-d7b02d710497>:6: FutureWarning:

A value is trying to be set on a copy of a DataFrame or Series through chained assignment using an inplace method.
The behavior will change in pandas 3.0. This inplace method will never work because the intermediate object on which we ar

For example, when doing 'df[col].method(value, inplace=True)', try using 'df.method({col: value}, inplace=True)' or df[co



<ipython-input-26-d7b02d710497>:6: FutureWarning:
```

```
Series.fillna with 'method' is deprecated and will raise in a future version. Use obj.ffill() or obj.bfill() instead.

<ipython-input-26-d7b02d710497>:7: FutureWarning:

A value is trying to be set on a copy of a DataFrame or Series through chained assignment using an inplace method.
The behavior will change in pandas 3.0. This inplace method will never work because the intermediate object on which we ar

For example, when doing 'df[col].method(value, inplace=True)', try using 'df.method({col: value}, inplace=True)' or df[col



<ipython-input-26-d7b02d710497>:7: FutureWarning:

Series.fillna with 'method' is deprecated and will raise in a future version. Use obj.ffill() or obj.bfill() instead.
```

Interactive Visualizations with Plotly

```python
import plotly.express as px

# Make sure 'country' and 'energy_per_capita' columns exist in your dataset
if 'country' in df.columns and 'energy_per_capita' in df.columns:
    # Create the choropleth map
    fig = px.choropleth(df,
                        locations="country",  # Column with country names
                        locationmode="country names",  # Specify location mode
                        color="energy_per_capita",  # Column to color the map by
                        hover_name="country",  # Hover text will show country names
                        title="Per Capita Energy Consumption by Country",
                        color_continuous_scale="Viridis",  # Color scale for visualization
                        projection="natural earth"  # Use 'natural earth' map projection
                        )

    # Customize layout for better clarity
    fig.update_layout(
        geo=dict(showframe=False, showcoastlines=False),  # Remove borders and coastlines
        coloraxis_colorbar=dict(title="Energy per Capita (TWh)")  # Color bar title
    )

    # Show the interactive plot
    fig.show()

else:
    print("Error: 'country' or 'energy_per_capita' column does not exist.")
```



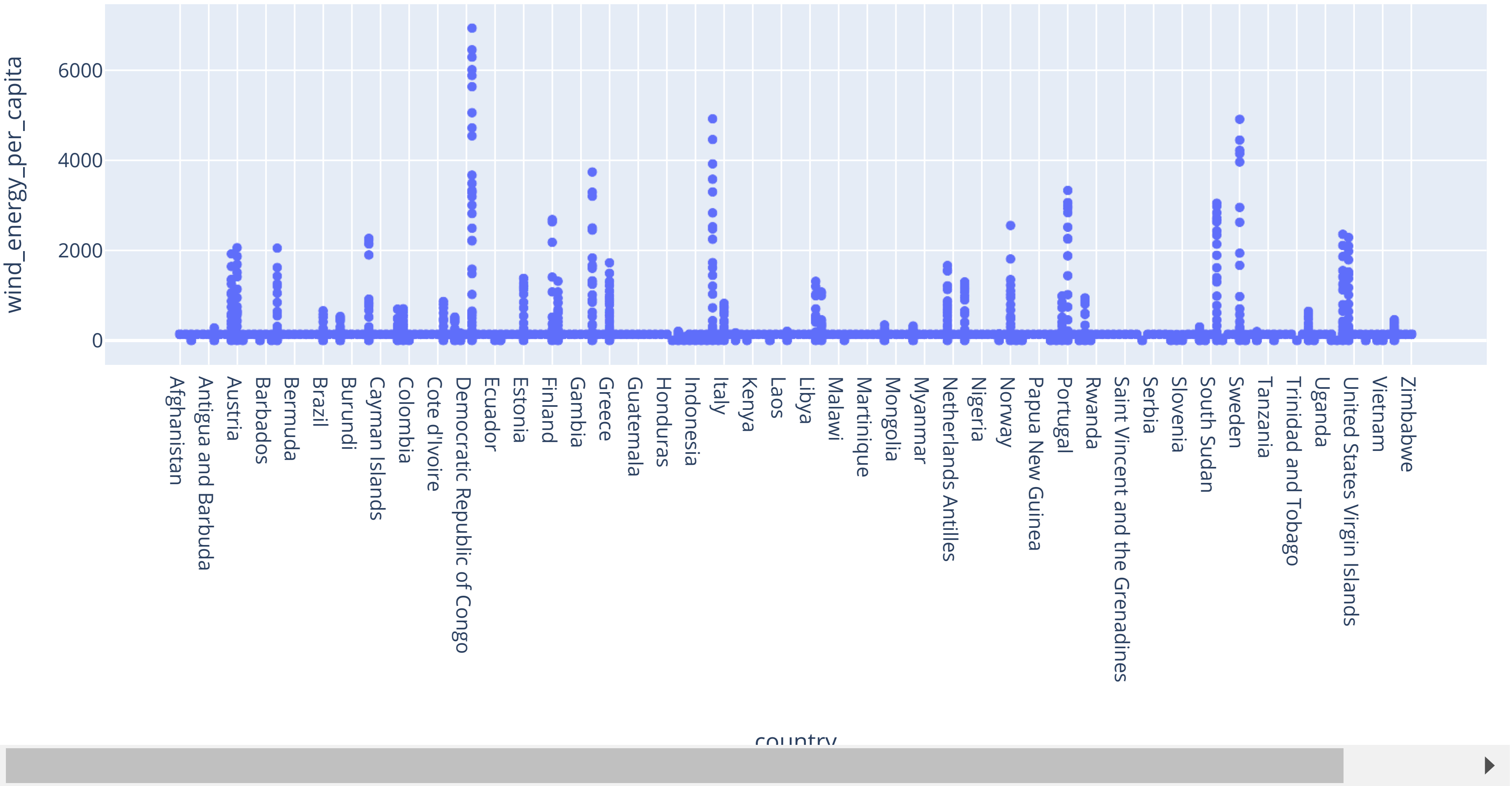
Start coding or generate with AI.

```
print(df.head())
```

```
  iso_code      country  year  coal_prod_change_pct  coal_prod_change_twh  \
0      AFG  Afghanistan  1900             20.830774              8.798102   
1      AFG  Afghanistan  1901             20.830774              0.000000   
2      AFG  Afghanistan  1902             20.830774              0.000000   
3      AFG  Afghanistan  1903             20.830774              0.000000   
```

```
4     AFG  Afghanistan  1904            20.830774             0.000000

   gas_prod_change_pct  gas_prod_change_twh  oil_prod_change_pct  \
0         1.921623e+14            14.369018             18.24219   
1         1.921623e+14            14.369018             18.24219   
2         1.921623e+14            14.369018             18.24219   
3         1.921623e+14            14.369018             18.24219   
4         1.921623e+14            14.369018             18.24219   

   oil_prod_change_twh  energy_cons_change_pct  ...  solar_energy_per_capita  \
0            18.033792                     inf  ...                29.375128   
1            18.033792                     inf  ...                29.375128   
2            18.033792                     inf  ...                29.375128   
3            18.033792                     inf  ...                29.375128   
4            18.033792                     inf  ...                29.375128   

            gdp  wind_share_elec  wind_cons_change_pct  wind_share_energy  \
0  5.417833e+11         1.006011            313.478014           0.345406   
1  5.417833e+11         1.006011            313.478014           0.345406   
2  5.417833e+11         1.006011            313.478014           0.345406   
3  5.417833e+11         1.006011            313.478014           0.345406   
4  5.417833e+11         1.006011            313.478014           0.345406   

   wind_cons_change_twh  wind_consumption  wind_elec_per_capita  \
0               2.16383         15.080935             53.625783   
1               2.16383         15.080935             53.625783   
2               2.16383         15.080935             53.625783   
3               2.16383         15.080935             53.625783   
4               2.16383         15.080935             53.625783   

   wind_energy_per_capita  Wind_Consumption_Growth  
0              134.003056                      NaN  
1              134.003056                      0.0  
2              134.003056                      0.0  
3              134.003056                      0.0  
4              134.003056                      0.0  

[5 rows x 123 columns]
```

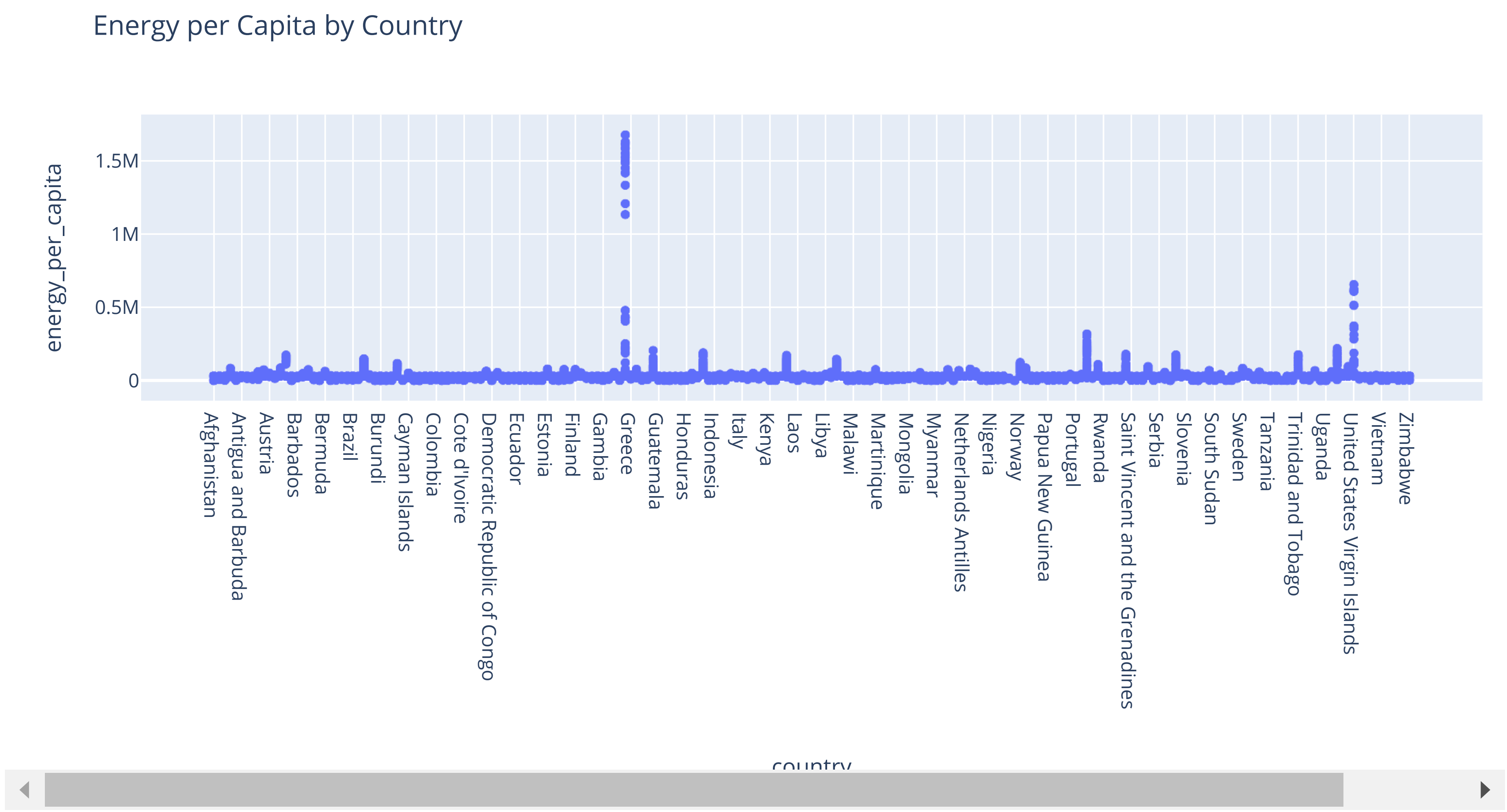
```
if 'country' in df.columns and 'wind_energy_per_capita' in df.columns:
    # Your plotting code here, e.g.:
    fig = px.scatter(df, x='country', y='wind_energy_per_capita')
    fig.show()
else:
    print("Error: 'country' or 'wind_energy_per_capita' column does not exist.")
```



```
print(df.columns.tolist())
```

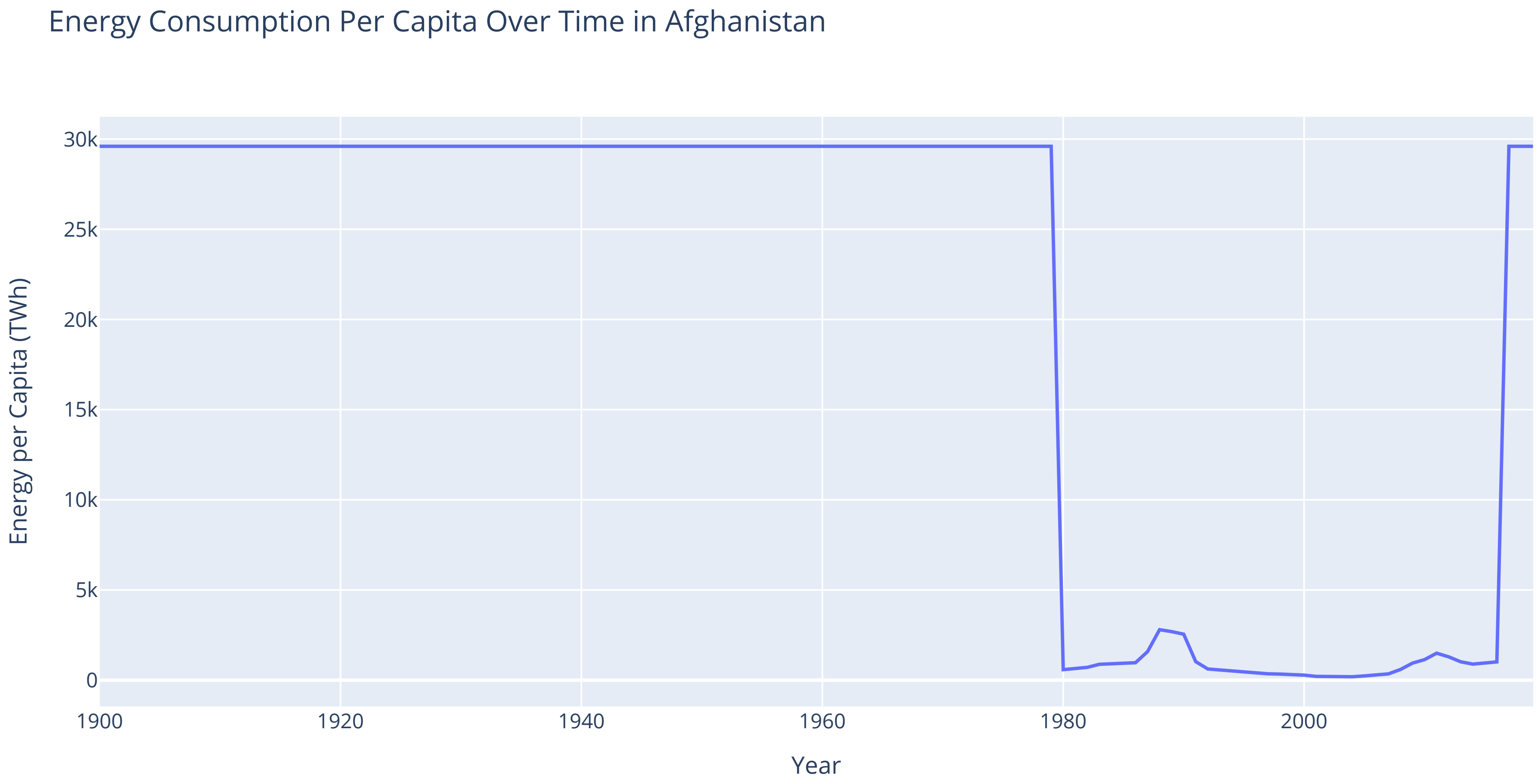
```
['iso_code', 'country', 'year', 'coal_prod_change_pct', 'coal_prod_change_twh', 'gas_prod_change_pct', 'gas_prod_change_tw
```

```
import plotly.express as px

# Check if the required columns exist
if 'country' in df.columns and 'energy_per_capita' in df.columns:
    # Create the interactive plot
    fig = px.scatter(df, x='country', y='energy_per_capita', title='Energy per Capita by Country')

    # Show the interactive plot
    fig.show()
else:
    print("Error: 'country' or 'energy_per_capita' column does not exist.")
```



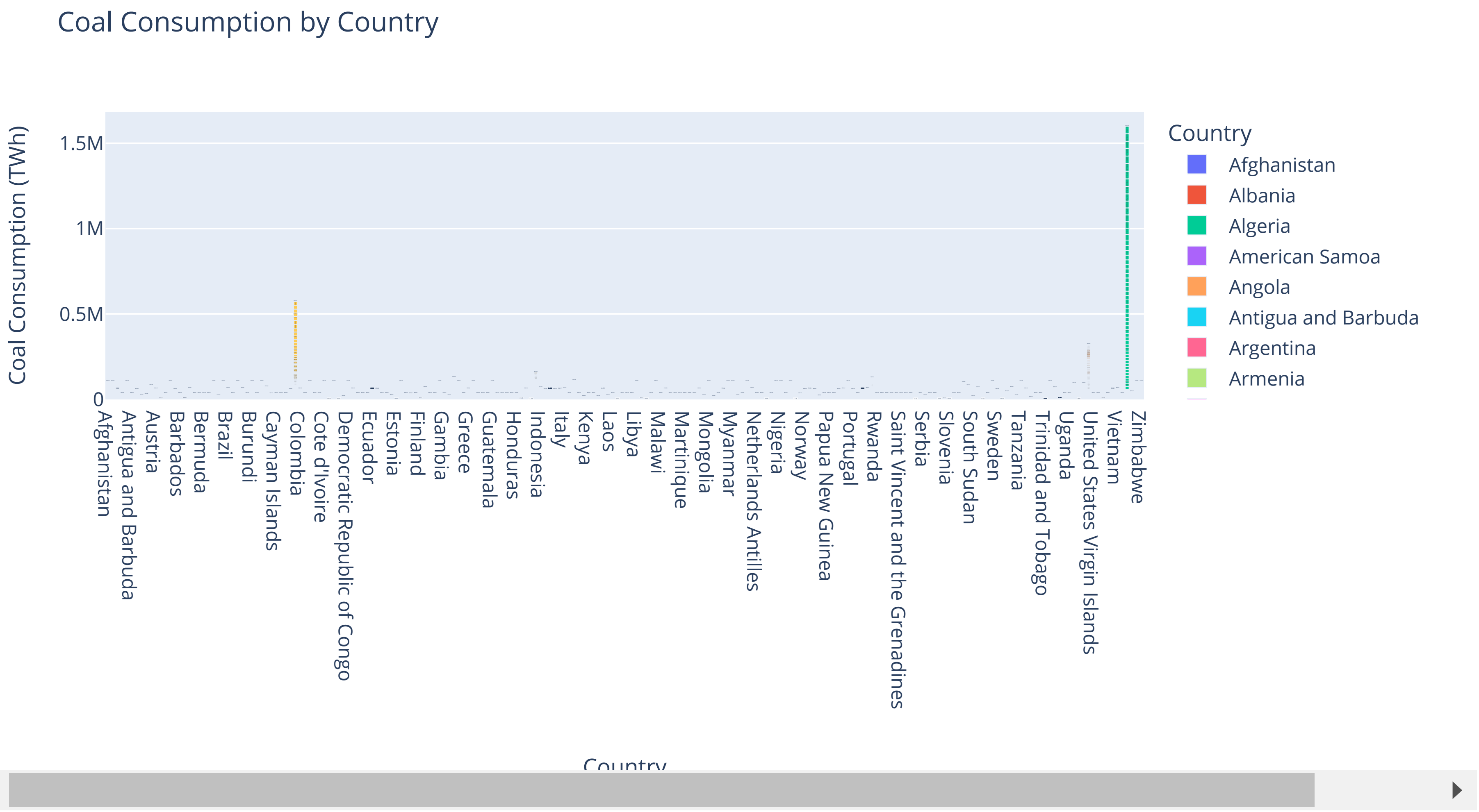
Time Series Plot

```
import plotly.express as px

# Filter for a specific country (e.g., 'Afghanistan')
country_data = df[df['country'] == 'Afghanistan']

fig = px.line(country_data, x='year', y='energy_per_capita'
fig = px.line(country_data, x='year', y='energy_per_capita',
              title='Energy Consumption Per Capita Over Time in Afghanistan',
              labels={'year': 'Year', 'energy_per_capita': 'Energy per Capita (TWh)'})
fig.show()
```



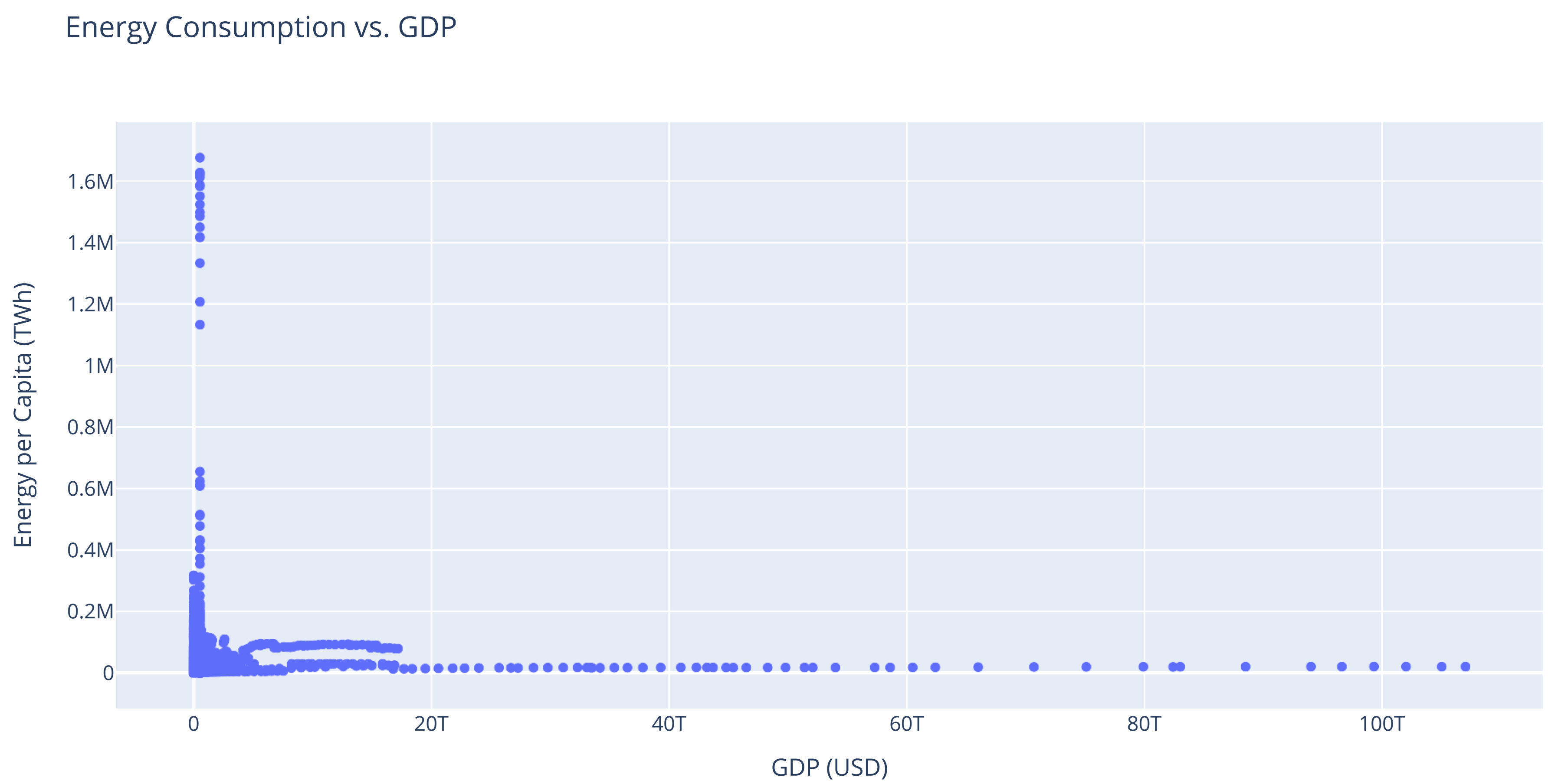
Bar Chart

```
fig = px.bar(df,
fig = px.bar(df,
             x='country',  # Use a specific subset or top N countries
             y='coal_consumption',  # Or any other energy-related metric
             title='Coal Consumption by Country',
             labels={'country': 'Country', 'coal_consumption': 'Coal Consumption (TWh)'},
             color='country',  # Color by country for differentiation
             text='coal_consumption')  # Show values on bars
fig.update_traces(texttemplate='%{text:.2f}', textposition='outside')
fig.show()
```



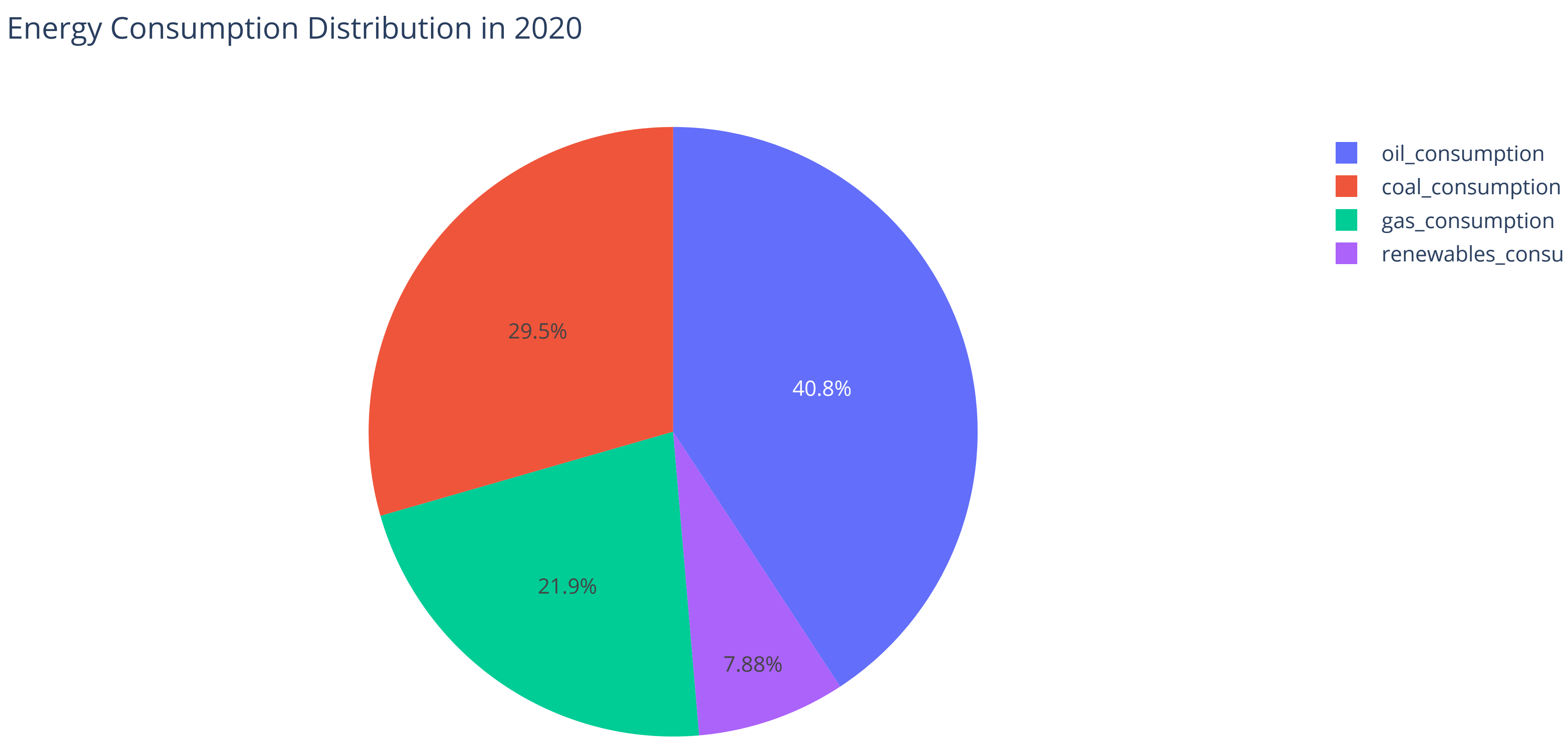
Scatter Plot

```
fig = px.scatter(df, x='gdp', y='energy_per_capita',
                 title='Energy Consumption vs. GDP',
                 labels={'gdp': 'GDP (USD)', 'energy_per_capita': 'Energy per Capita (TWh)'},
                 hover_name='country')
fig.show()
```



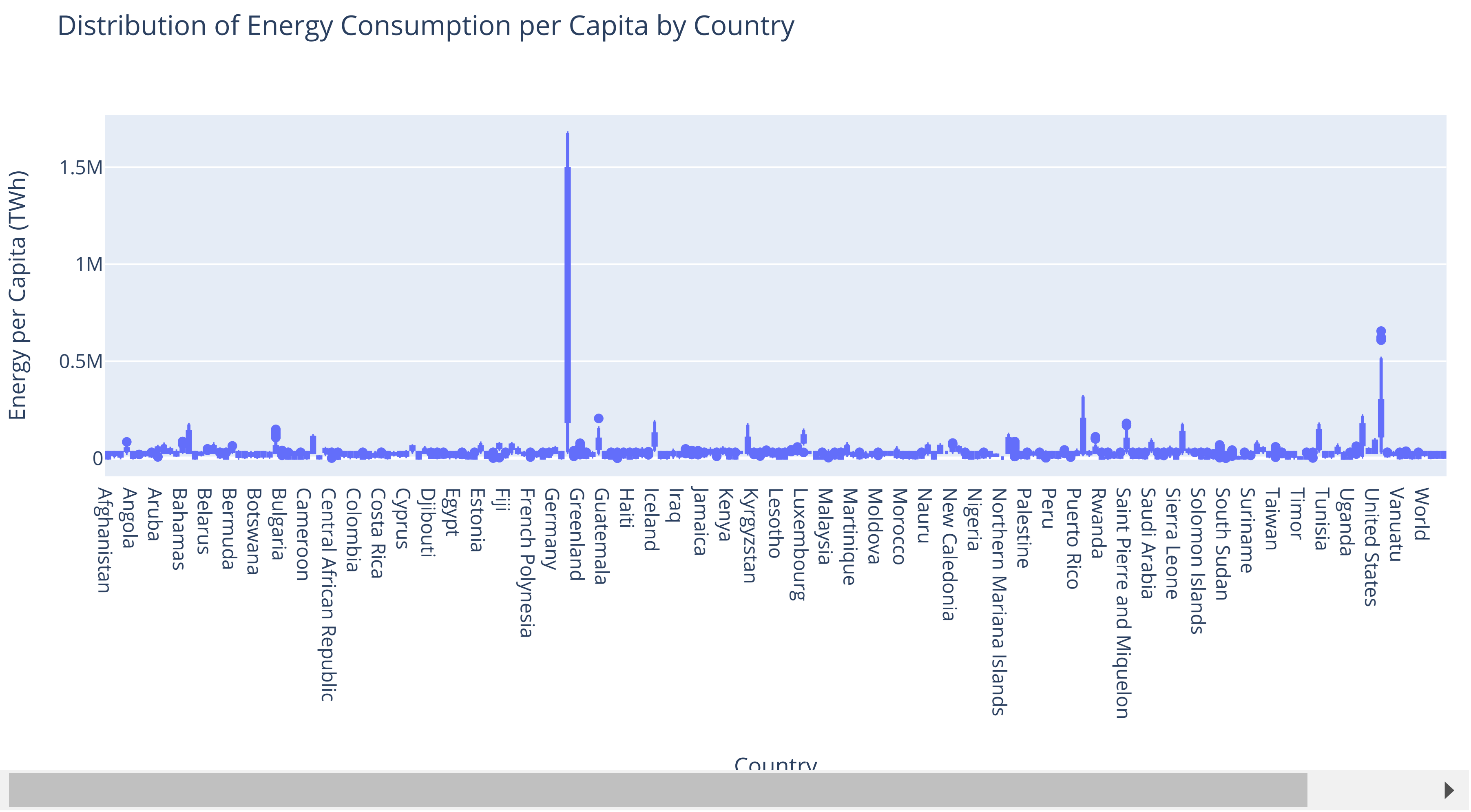
Pie Chart

```
# Aggregating data for a specific year, e.g., 2020
total_consumption = df[df['year'] == 2020].sum(numeric_only=True)
energy_types = ['coal_consumption', 'gas_consumption', 'oil_consumption', 'renewables_consumption']

fig = px.pie(values=total_consumption[energy_types],
             names=energy_types,
             names=energy_types,
             title='Energy Consumption Distribution in 2020')
fig.show()
```



Box Plot

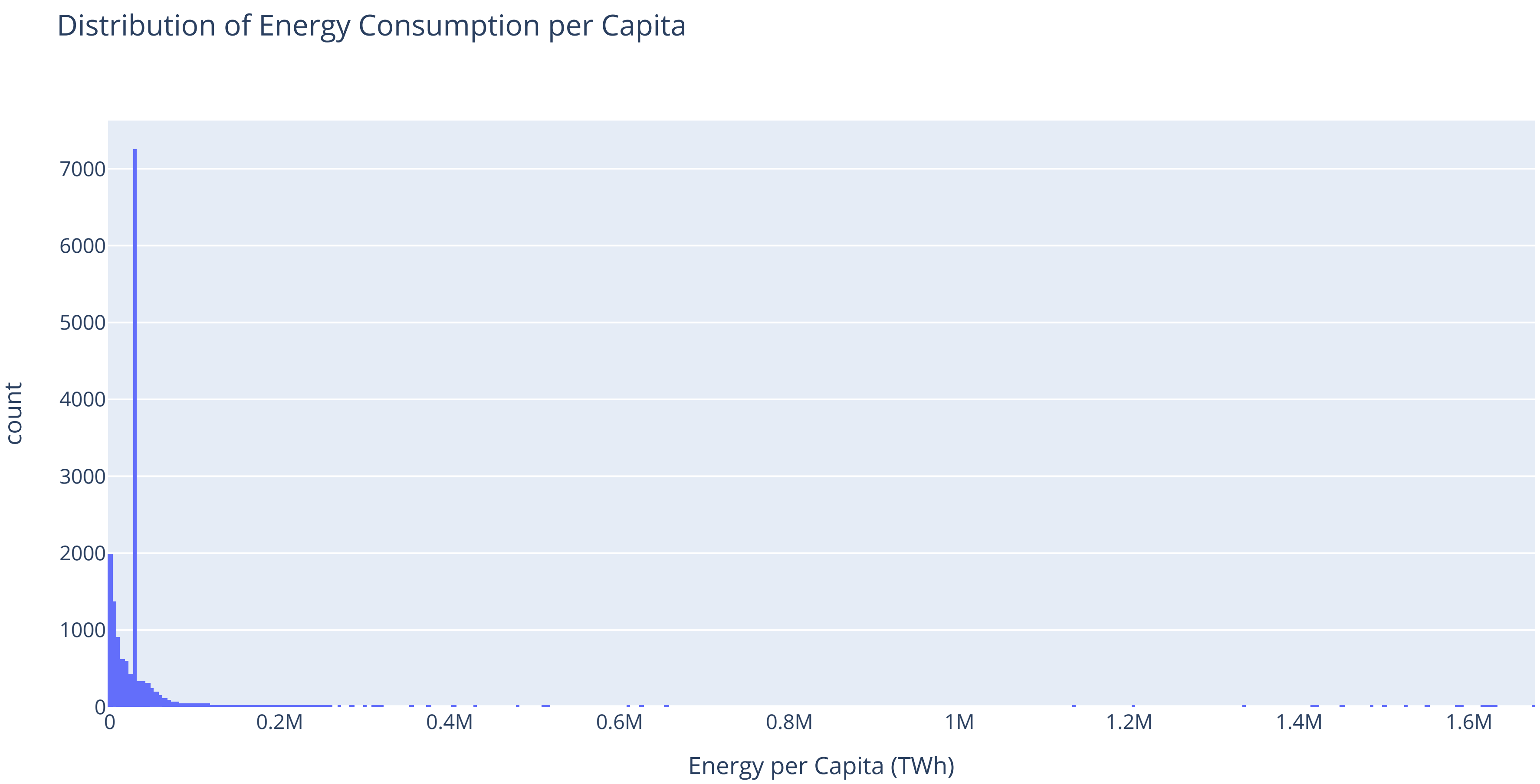
```
fig = px.box(df, x='country', y='energy_per_capita',
             title='Distribution of Energy Consumption per Capita by Country'
             title='Distribution of Energy Consumption per Capita by Country',
             labels={'country': 'Country', 'energy_per_capita': 'Energy per Capita (TWh)'})
fig.show()
```



Distribution of Energy Consumption per Capita by Country

Histogram

```
fig = px.histogram(df, x='energy_per_capita',
                   title='Distribution of Energy Consumption per Capita',
                   title='Distribution of Energy Consumption per Capita',
                   labels={'energy_per_capita': 'Energy per Capita (TWh)'})
fig.show()
```



```
def create_choropleth(df):
    fig = px.choropleth(df,
                        locations='country',
                        locationmode='country names',
                        color='energy_per_capita',
```