"Renewable Energy Revolution: Analyzing Global Progress Through Data Visualization"

loading dataset

```
[ ]  # Method 1: Using files.upload() - for smaller files
     from google.colab import files
     uploaded = files.upload()

     # After uploading, you can read it based on file type, for example:
     # For CSV:
     import pandas as pd
     df = pd.read_csv(next(iter(uploaded)))
```

Browse... global-data-on-sustainable-energy (1).csv

global-data-on-sustainable-energy (1).csv(application/vnd.ms-excel) - 513817 bytes, last modified: n/a - 100% done

```
Saving global-data-on-sustainable-energy (1).csv to global-data-on-sustainable-energy (1).csv
```

```
[ ]  # Get the filename of the uploaded file
     filename = list(uploaded.keys())[0]
```

```
[ ]  # Read the CSV file into a DataFrame
     df = pd.read_csv(filename)
```

```
[ ] df.head(300)
```

	Entity	Year	Access to electricity (% of population)	Access to clean fuels for cooking	Renewable-electricity-generating-capacity-per-capita	Financial flows to developing countries (US $)	Renewable energy share in the total final energy consumption (%)	Electricity from fossil fuels (TWh)	Electricity from nuclear (TWh)	Electricity from renewables (TWh)	...	Primary energy consumption per capita (kWh/person)	Energy intensity level of primary energy (MJ/$2017 PPP GDP)	Value_co2_emissions_kt_by_country	Renewables (% equivalent primary energy)	gdp_growth	gdp_per_capita	De
0	Afghanistan	2000	1.613591	6.2	9.22	20000.0	44.99	0.16	0.0	0.31	...	302.59482	1.64	760.000000	NaN	NaN	NaN	
1	Afghanistan	2001	4.074574	7.2	8.86	130000.0	45.60	0.09	0.0	0.50	...	236.89185	1.74	730.000000	NaN	NaN	NaN	
2	Afghanistan	2002	9.409158	8.2	8.47	3950000.0	37.83	0.13	0.0	0.56	...	210.86215	1.40	1029.999971	NaN	NaN	179.426579	
3	Afghanistan	2003	14.738506	9.5	8.09	25970000.0	36.66	0.31	0.0	0.63	...	229.96822	1.40	1220.000029	NaN	8.832278	190.683814	
4	Afghanistan	2004	20.064968	10.9	7.75	NaN	44.24	0.33	0.0	0.56	...	204.23125	1.20	1029.999971	NaN	1.414118	211.382074	
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	
295	Barbados	2001	100.000000	100.0	0.37	NaN	12.56	0.80	0.0	0.00	...	26262.20900	4.01	1310.000000	NaN	-2.365464	11209.421120	
296	Barbados	2002	100.000000	100.0	0.37	NaN	11.30	0.83	0.0	0.00	...	26331.50000	3.91	1370.000005	NaN	0.787402	11361.516770	
297	Barbados	2003	100.000000	100.0	0.37	NaN	9.95	0.87	0.0	0.00	...	26508.63000	3.92	1379.999995	NaN	2.175481	11699.370470	
298	Barbados	2004	99.996740	100.0	0.36	NaN	9.93	0.90	0.0	0.00	...	26606.23600	3.97	1360.000014	NaN	1.411599	12512.577970	
299	Barbados	2005	99.980450	100.0	0.72	NaN	9.91	0.95	0.0	0.00	...	26715.06000	3.97	1419.999957	NaN	3.955458	13822.741750	

300 rows × 21 columns

Preprocessing data - analyzing missing values...

```
[ ] df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 3649 entries, 0 to 3648
Data columns (total 21 columns):
 #   Column                                                  Non-Null Count  Dtype  
---  ------                                                  --------------  -----  
 0   Entity                                                  3649 non-null   object 
 1   Year                                                    3649 non-null   int64  
 2   Access to electricity (% of population)                 3639 non-null   float64
 3   Access to clean fuels for cooking                       3480 non-null   float64
 4   Renewable-electricity-generating-capacity-per-capita    2718 non-null   float64
 5   Financial flows to developing countries (US $)          1560 non-null   float64
 6   Renewable energy share in the total final energy consumption (%)  3455 non-null   float64
 7   Electricity from fossil fuels (TWh)                     3628 non-null   float64
 8   Electricity from nuclear (TWh)                          3523 non-null   float64
 9   Electricity from renewables (TWh)                       3628 non-null   float64
 10  Low-carbon electricity (% electricity)                  3607 non-null   float64
 11  Primary energy consumption per capita (kWh/person)      3649 non-null   float64
 12  Energy intensity level of primary energy (MJ/$2017 PPP GDP)  3442 non-null   float64
 13  Value_co2_emissions_kt_by_country                       3221 non-null   float64
 14  Renewables (% equivalent primary energy)                1512 non-null   float64
 15  gdp_growth                                              3332 non-null   float64
 16  gdp_per_capita                                          3367 non-null   float64
 17  Density\n(P/Km2)                                        3648 non-null   object 
 18  Land Area(Km2)                                          3648 non-null   float64
 19  Latitude                                                3648 non-null   float64
 20  Longitude                                               3648 non-null   float64
dtypes: float64(18), int64(1), object(2)
memory usage: 598.8+ KB
```

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

	Year	Access to electricity (% of population)	Access to clean fuels for cooking	Renewable-electricity-generating-capacity-per-capita	Financial flows to developing countries (US $)	Renewable energy share in the total final energy consumption (%)	Electricity from fossil fuels (TWh)	Electricity from nuclear (TWh)	Electricity from renewables (TWh)	Low-carbon electricity (% electricity)	Primary energy consumption per capita (kWh/person)	Energy intensity level of primary energy (MJ/$2017 PPP GDP)	Value_co2_emissions_kt_by_country	Renewables (% equivalent primary energy)	gdp_growth	gdp
count	3649.000000	3639.000000	3480.000000	2718.000000	1.560000e+03	3455.000000	3628.000000	3523.000000	3628.000000	3607.000000	3649.000000	3442.000000	3.221000e+03	1512.000000	3332.000000	
mean	2010.038367	78.933702	63.255287	113.137498	9.422400e+07	32.638165	70.365003	13.450190	23.968010	36.801182	25743.981745	5.307345	1.598665e+05	11.986707	3.441610	1
std	6.054228	30.275541	39.043658	244.167256	2.981544e+08	29.894901	348.051866	73.006623	104.431085	34.314884	34773.221366	3.532020	7.736611e+05	14.994644	5.686720	1
min	2000.000000	1.252269	0.000000	0.000000	0.000000e+00	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.110000	1.000000e+01	0.000000	-62.075920	
25%	2005.000000	59.800890	23.175000	3.540000	2.600000e+05	6.515000	0.290000	0.000000	0.040000	2.877847	3116.737300	3.170000	2.020000e+03	2.137095	1.383302	
50%	2010.000000	98.361570	83.150000	32.910000	5.665000e+06	23.300000	2.970000	0.000000	1.470000	27.865068	13120.570000	4.300000	1.050000e+04	6.290766	3.559855	
75%	2015.000000	100.000000	100.000000	112.210000	5.534750e+07	55.245000	26.837500	0.000000	9.600000	64.403792	33892.780000	6.027500	6.058000e+04	16.841638	5.830099	1
max	2020.000000	100.000000	100.000000	3060.190000	5.202310e+09	96.040000	5184.130000	809.410000	2184.940000	100.000010	262585.700000	32.570000	1.070722e+07	86.836586	123.139555	12

```python
# Display the number of missing values in each column
print(df.isnull().sum())

# Get the percentage of missing values in each column
print(df.isnull().sum() / len(df) * 100)
```

```
Entity                                                      0
Year                                                        0
Access to electricity (% of population)                    10
Access to clean fuels for cooking                         169
Renewable-electricity-generating-capacity-per-capita      931
Financial flows to developing countries (US $)           2089
Renewable energy share in the total final energy consumption (%)    194
Electricity from fossil fuels (TWh)                        21
Electricity from nuclear (TWh)                            126
Electricity from renewables (TWh)                          21
Low-carbon electricity (% electricity)                     42
Primary energy consumption per capita (kWh/person)          0
Energy intensity level of primary energy (MJ/$2017 PPP GDP)       207
Value_co2_emissions_kt_by_country                         428
Renewables (% equivalent primary energy)                 2137
gdp_growth                                                317
gdp_per_capita                                            282
Density\n(P/Km2)                                            1
Land Area(Km2)                                              1
Latitude                                                    1
Longitude                                                   1
dtype: int64
Entity                                                   0.000000
Year                                                     0.000000
Access to electricity (% of population)                  0.274048
Access to clean fuels for cooking                        4.631406
Renewable-electricity-generating-capacity-per-capita    25.513839
Financial flows to developing countries (US $)          57.248561
Renewable energy share in the total final energy consumption (%)    5.316525
Electricity from fossil fuels (TWh)                      0.575500
Electricity from nuclear (TWh)                           3.453001
Electricity from renewables (TWh)                        0.575500
Low-carbon electricity (% electricity)                   1.151000
Primary energy consumption per capita (kWh/person)       0.000000
Energy intensity level of primary energy (MJ/$2017 PPP GDP)       5.672787
Value_co2_emissions_kt_by_country                       11.729241
Renewables (% equivalent primary energy)                58.563990
gdp_growth                                               8.687312
gdp_per_capita                                           7.728145
Density\n(P/Km2)                                         0.027405
Land Area(Km2)                                           0.027405
Latitude                                                 0.027405
Longitude                                                0.027405
dtype: float64
```

```
# Count the number of duplicate rows
duplicate_count = df.duplicated().sum()
print(f"Number of duplicate rows: {duplicate_count}")

# Display duplicate rows (if any)
print(df[df.duplicated(keep=False)])
```

```
Number of duplicate rows: 0
Empty DataFrame
Columns: [Entity, Year, Access to electricity (% of population), Access to clean fuels for cooking, Renewable-electricity-generating-capacity-per-capita, Financial flows to developing countries (US $), Renewable energy share in the total final energy consumpt:
Index: []

[0 rows x 21 columns]
```

```
df.dropna()
```

	Entity	Year	Access to electricity (% of population)	Access to clean fuels for cooking	Renewable-electricity-generating-capacity-per-capita	Financial flows to developing countries (US $)	Renewable energy share in the total final energy consumption (%)	Electricity from fossil fuels (TWh)	Electricity from nuclear (TWh)	Electricity from renewables (TWh)	...	Primary energy consumption per capita (kWh/person)	Energy intensity level of primary energy (MJ/$2017 PPP GDP)	Value_co2_emissions_kt_by_country	Renewables (% equivalent primary energy)	gdp_growth	gdp_per_capita	Density\n(P/Km2)	Land Area(Km2)	Latitude	Long
43	Algeria	2001	98.96687	97.30	8.79	810000.0	0.43	24.96	0.0	0.07	...	9961.640	4.07	78650.00000	0.065218	3.000000	1740.606654	18	2381741.0	28.033886	1.6
44	Algeria	2002	98.95306	97.80	8.68	310000.0	0.51	25.94	0.0	0.06	...	10180.350	4.12	82400.00153	0.051677	5.600000	1781.828908	18	2381741.0	28.033886	1.6
45	Algeria	2003	98.93401	98.00	8.57	90000.0	0.47	27.54	0.0	0.26	...	10510.461	4.08	88190.00244	0.228104	7.200000	2103.381291	18	2381741.0	28.033886	1.6
46	Algeria	2004	98.91208	98.20	8.46	140000.0	0.44	29.14	0.0	0.25	...	10759.022	3.96	89489.99786	0.206787	4.300000	2610.185422	18	2381741.0	28.033886	1.6
47	Algeria	2005	98.88961	98.50	8.34	160000.0	0.58	31.36	0.0	0.55	...	11113.723	3.90	94190.00244	0.434119	5.900000	3113.094883	18	2381741.0	28.033886	1.6
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	
3559	Uzbekistan	2015	100.00000	85.35	60.83	270000.0	1.71	47.55	0.0	7.00	...	17386.195	7.86	99169.99817	3.012993	7.218774	2753.971072	79	447400.0	41.377491	64.5
3560	Uzbekistan	2016	100.00000	85.20	59.88	690000.0	1.61	48.75	0.0	7.25	...	16374.342	7.82	105230.00340	3.567936	5.932151	2704.677188	79	447400.0	41.377491	64.5
3561	Uzbekistan	2017	100.00000	84.90	58.24	60130000.0	1.75	49.71	0.0	8.35	...	16642.676	7.88	109529.99880	3.972285	4.395275	1916.764642	79	447400.0	41.377491	64.5
3562	Uzbekistan	2018	100.00000	84.30	59.09	84130000.0	1.49	53.58	0.0	5.85	...	16445.740	9.05	112470.00120	2.927033	5.354997	1597.068337	79	447400.0	41.377491	64.5
3563	Uzbekistan	2019	100.00000	84.60	57.96	65940000.0	1.57	53.64	0.0	6.47	...	16212.221	8.37	116709.99910	3.197033	5.709632	1784.009816	79	447400.0	41.377491	64.5

343 rows × 21 columns

```
df.dtypes
```

	0
Entity	object
Year	int64
Access to electricity (% of population)	float64
Access to clean fuels for cooking	float64
Renewable-electricity-generating-capacity-per-capita	float64
Financial flows to developing countries (US $)	float64
Renewable energy share in the total final energy consumption (%)	float64
Electricity from fossil fuels (TWh)	float64
Electricity from nuclear (TWh)	float64
Electricity from renewables (TWh)	float64
Low-carbon electricity (% electricity)	float64
Primary energy consumption per capita (kWh/person)	float64
Energy intensity level of primary energy (MJ/$2017 PPP GDP)	float64
Value_co2_emissions_kt_by_country	float64
Renewables (% equivalent primary energy)	float64
gdp_growth	float64
gdp_per_capita	float64
Density\n(P/Km2)	object
Land Area(Km2)	float64
Latitude	float64
Longitude	float64

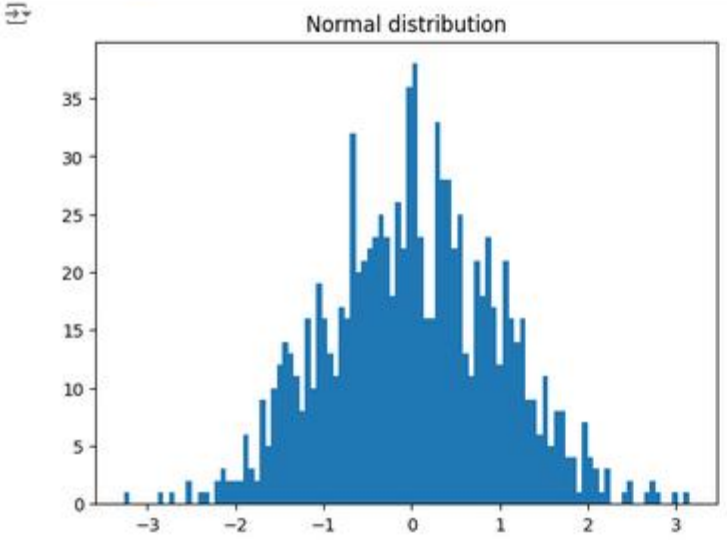
dtype: object

Working with data

```
[ ] #import libraries
    import pandas as pd
    import numpy as np
    !pip install matplotlib numpy
    import matplotlib.pyplot as plt
```

```
Requirement already satisfied: matplotlib in /usr/local/lib/python3.10/dist-packages (3.8.0)
Requirement already satisfied: numpy in /usr/local/lib/python3.10/dist-packages (1.26.4)
Requirement already satisfied: contourpy>=1.0.1 in /usr/local/lib/python3.10/dist-packages (from matplotlib) (1.3.1)
Requirement already satisfied: cycler>=0.10 in /usr/local/lib/python3.10/dist-packages (from matplotlib) (0.12.1)
Requirement already satisfied: fonttools>=4.22.0 in /usr/local/lib/python3.10/dist-packages (from matplotlib) (4.55.3)
Requirement already satisfied: kiwisolver>=1.0.1 in /usr/local/lib/python3.10/dist-packages (from matplotlib) (1.4.7)
Requirement already satisfied: packaging>=20.0 in /usr/local/lib/python3.10/dist-packages (from matplotlib) (24.2)
Requirement already satisfied: pillow>=6.2.0 in /usr/local/lib/python3.10/dist-packages (from matplotlib) (11.0.0)
Requirement already satisfied: pyparsing>=2.3.1 in /usr/local/lib/python3.10/dist-packages (from matplotlib) (3.2.0)
Requirement already satisfied: python-dateutil>=2.7 in /usr/local/lib/python3.10/dist-packages (from matplotlib) (2.8.2)
Requirement already satisfied: six>=1.5 in /usr/local/lib/python3.10/dist-packages (from python-dateutil>=2.7->matplotlib) (1.17.0)
```

```
[ ] # Generate some data (e.g., normally distributed data)
    x = np.random.normal(0, 1, 1000)

    # Create the histogram
    plt.hist(x, 100)

    # Set the title with a raw string to avoid issues with backslashes
    plt.title(r'Normal distribution')

    # Save the figure to a file
    plt.savefig('matplotlib_histogram.png')

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

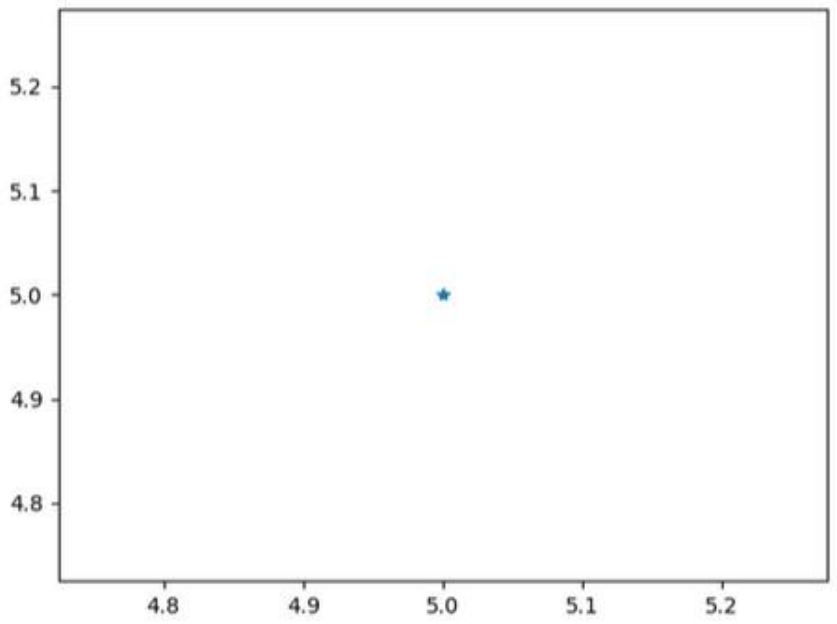


plottting artist layer to use

You're creating plots in a web application or server environment

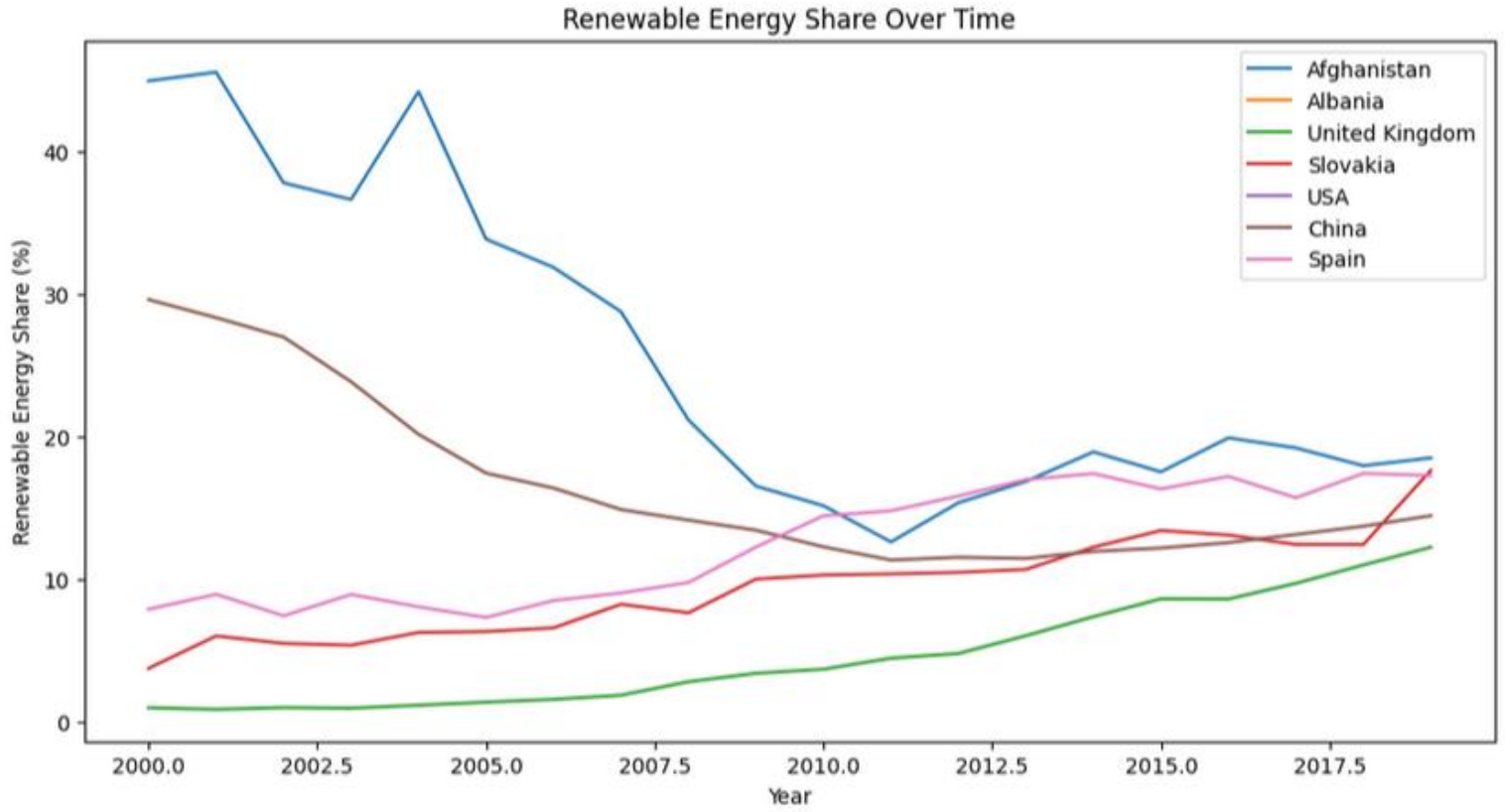
```
[ ] from matplotlib.backends.backend_agg import FigureCanvasAgg
    from matplotlib.figure import Figure

    # Create a new figure
    fig = Figure()

    # Create a canvas for the figure
    canvas = FigureCanvasAgg(fig)

    # Add a subplot to the figure
    ax = fig.add_subplot(111)

    # Plot a point at (5, 5)
    ax.plot(5, 5, '*')

    # Save the figure to a file
    canvas.print_figure('matplotlib_plot.png')
```

```
from IPython.display import display, Image
import matplotlib.pyplot as plt

# Create a new figure
fig = Figure()

# Create a canvas for the figure
canvas = FigureCanvasAgg(fig)

# Add a subplot to the figure
ax = fig.add_subplot(111)

# Plot a point at (5, 5)
ax.plot(5, 5, '*')

# Save the figure to a file
canvas.print_figure('matplotlib_plot.png')

# Display the saved image
display(Image(filename='matplotlib_plot.png'))
```

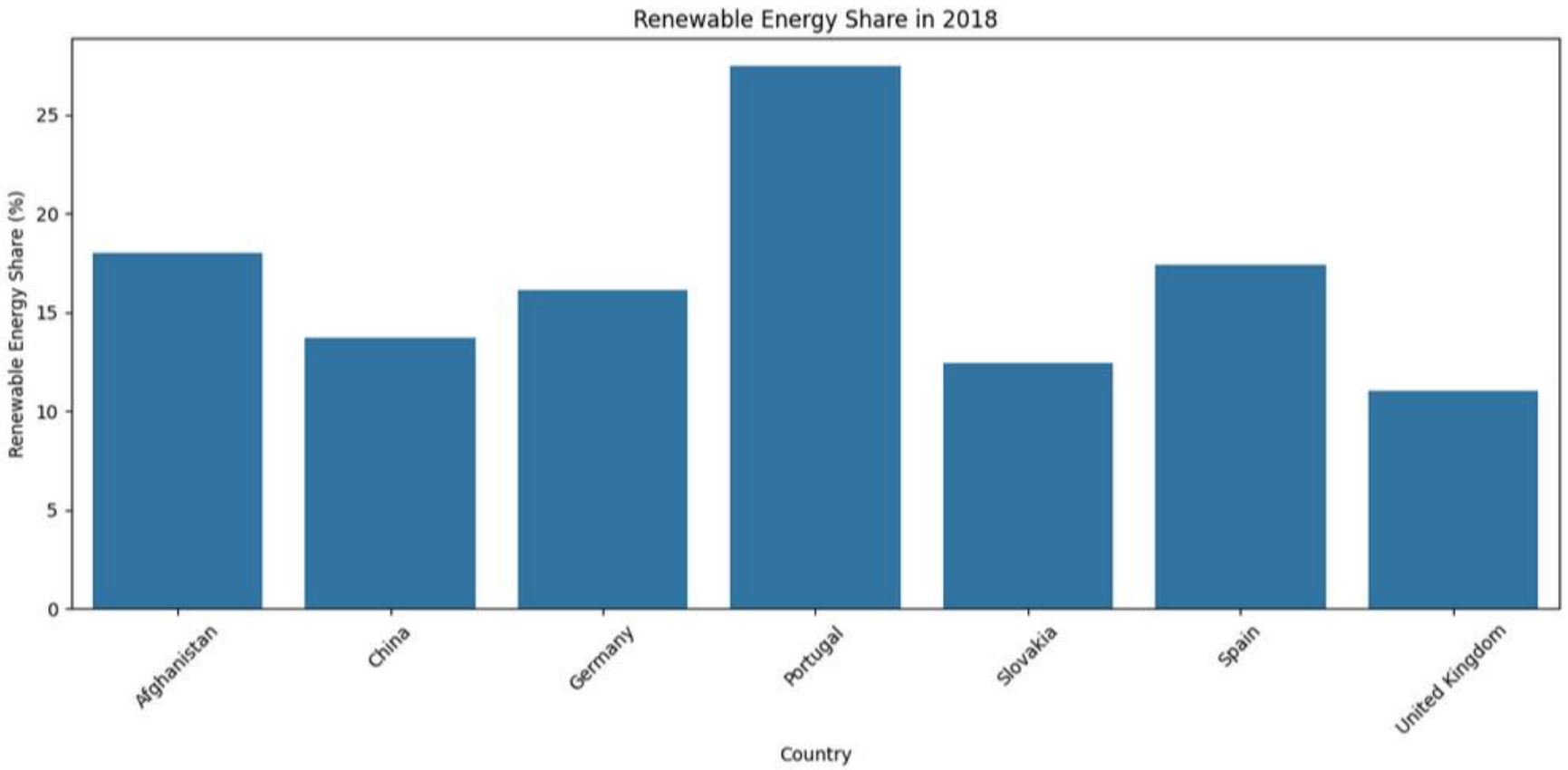


Visualizations

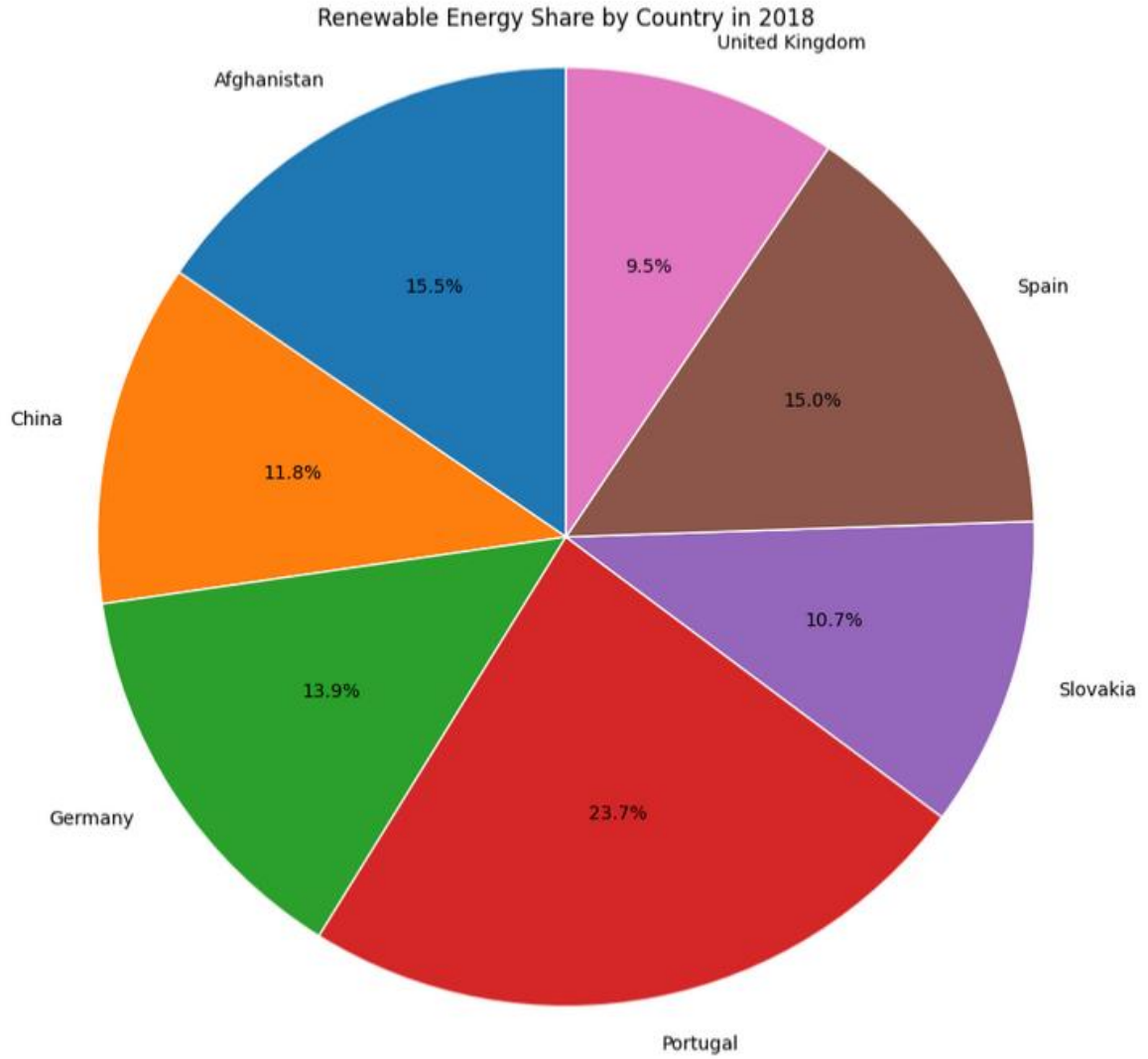
```
#import library
import seaborn as sns

# Plot renewable energy share for selected countries
countries = ['Afghanistan', 'Albania', 'United Kingdom','Slovakia','USA','China','Spain']
plt.figure(figsize=(12, 6))
for country in countries:
    country_data = df[df['Entity'] == country]
    plt.plot(country_data['Year'], country_data['Renewable energy share in the total final energy consumption (%)'], label=country)

plt.title('Renewable Energy Share Over Time')
plt.xlabel('Year')
plt.ylabel('Renewable Energy Share (%)')
plt.legend()
plt.show()
```



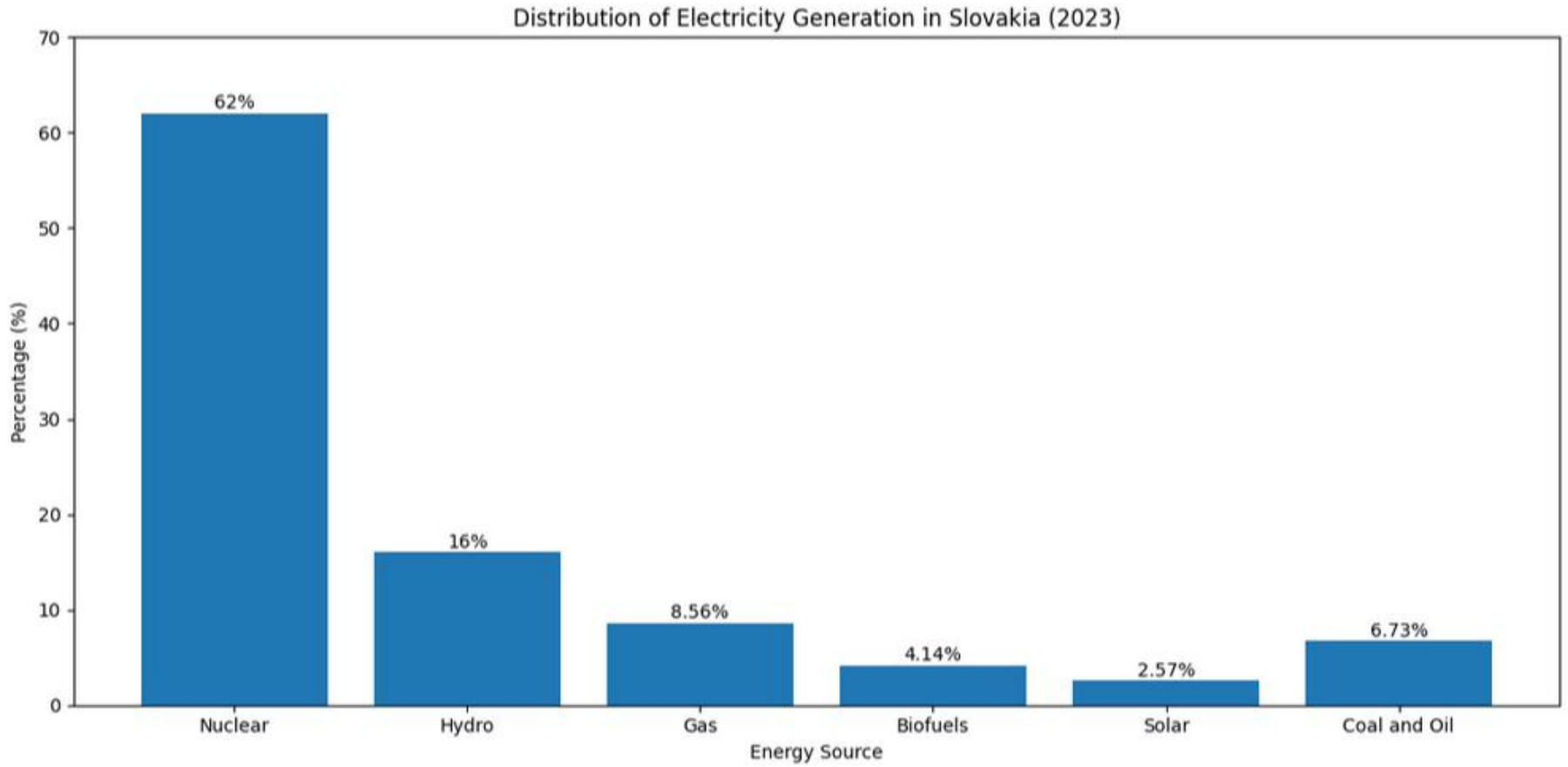
```python
# Select a specific year for the bar chart
year = 2018

# Filter data for the selected year and countries
countries = ['Afghanistan', 'United Kingdom', 'Slovakia','Germany','Franece','Spain','Portugal','China']
data_for_chart = df[(df['Entity'].isin(countries)) & (df['Year'] == year)]

# Create the bar chart
plt.figure(figsize=(12, 6))
sns.barplot(x='Entity', y='Renewable energy share in the total final energy consumption (%)', data=data_for_chart)

plt.title(f'Renewable Energy Share in {year}')
plt.xlabel('Country')
plt.ylabel('Renewable Energy Share (%)')
plt.xticks(rotation=45)  # Rotate x-axis labels for better readability
plt.tight_layout()  # Adjust layout to prevent clipping of labels
plt.show()
```



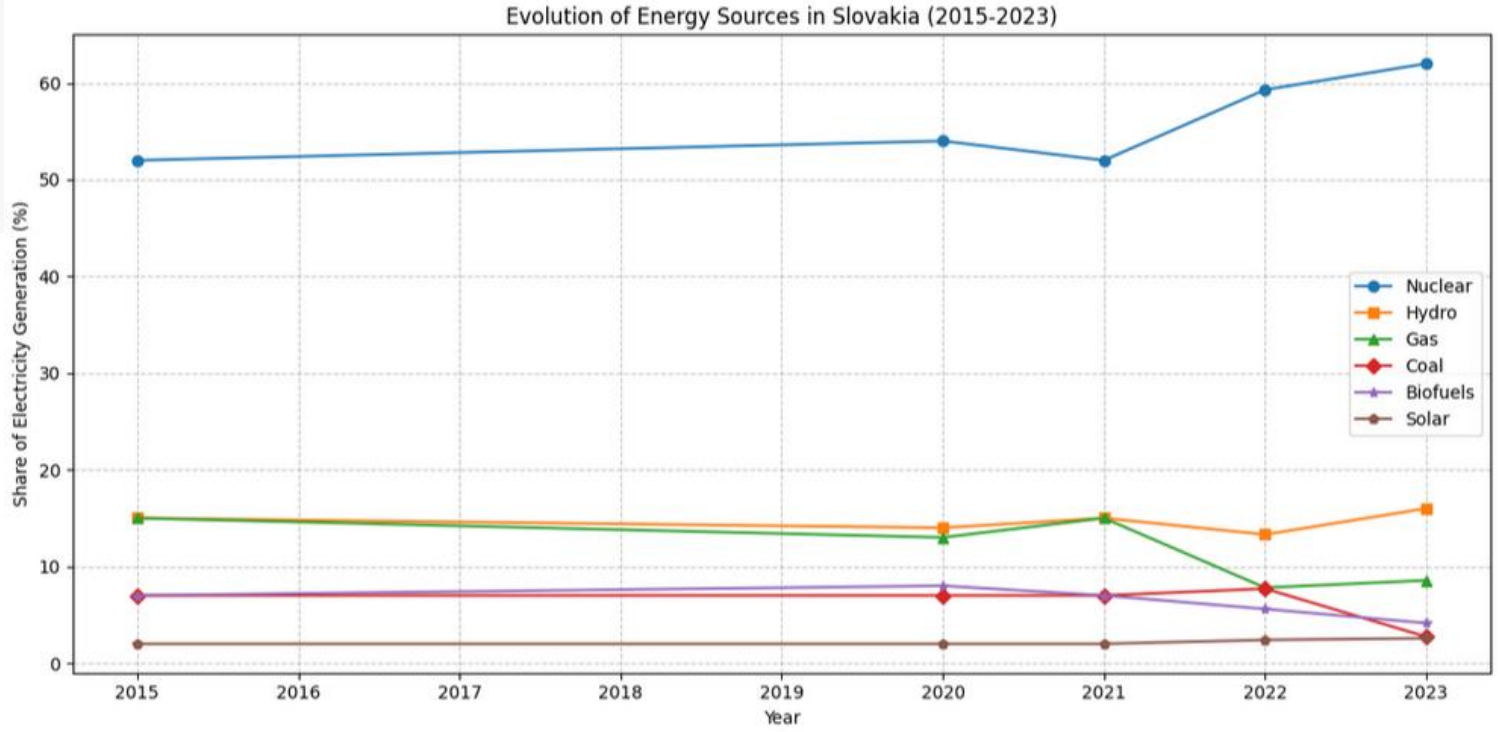
```python
# Create the pie chart
# setting up the value of the year
year = 2018
plt.figure(figsize=(10, 8))
plt.pie(data_for_chart['Renewable energy share in the total final energy consumption (%)'],
        labels=data_for_chart['Entity'],
        autopct='%1.1f%%',
        startangle=90,
        wedgeprops={'edgecolor': 'white'})

plt.title(f'Renewable Energy Share by Country in {year}')
plt.axis('equal')  # Equal aspect ratio ensures that pie is drawn as a circle
plt.tight_layout()
plt.show()
```



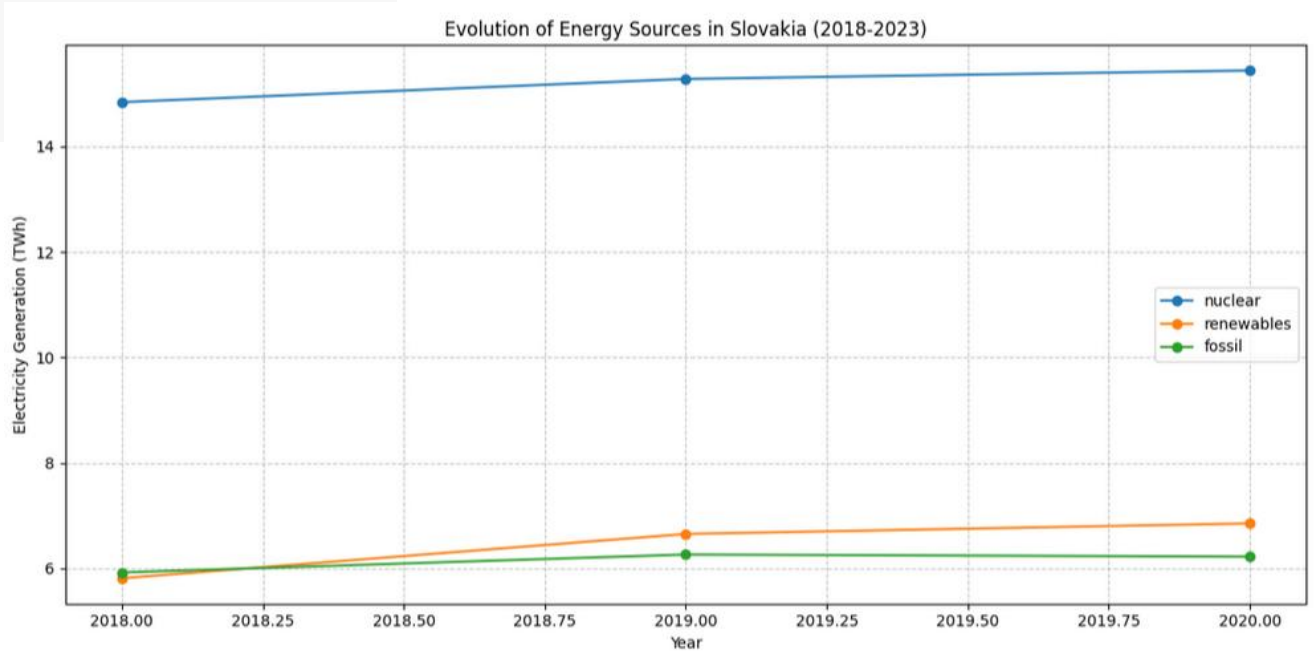
```python
#Distribution of Electricity Generation in Slovakia
sources = ['Nuclear', 'Hydro', 'Gas', 'Biofuels', 'Solar', 'Coal and Oil']
percentages = [62, 16, 8.56, 4.14, 2.57, 6.73]

plt.figure(figsize=(12, 6))
plt.bar(sources, percentages)
plt.title("Distribution of Electricity Generation in Slovakia (2023)")
plt.xlabel("Energy Source")
plt.ylabel("Percentage (%)")
plt.ylim(0, 70)

for i, v in enumerate(percentages):
    plt.text(i, v + 0.5, f'{v}%', ha='center')

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



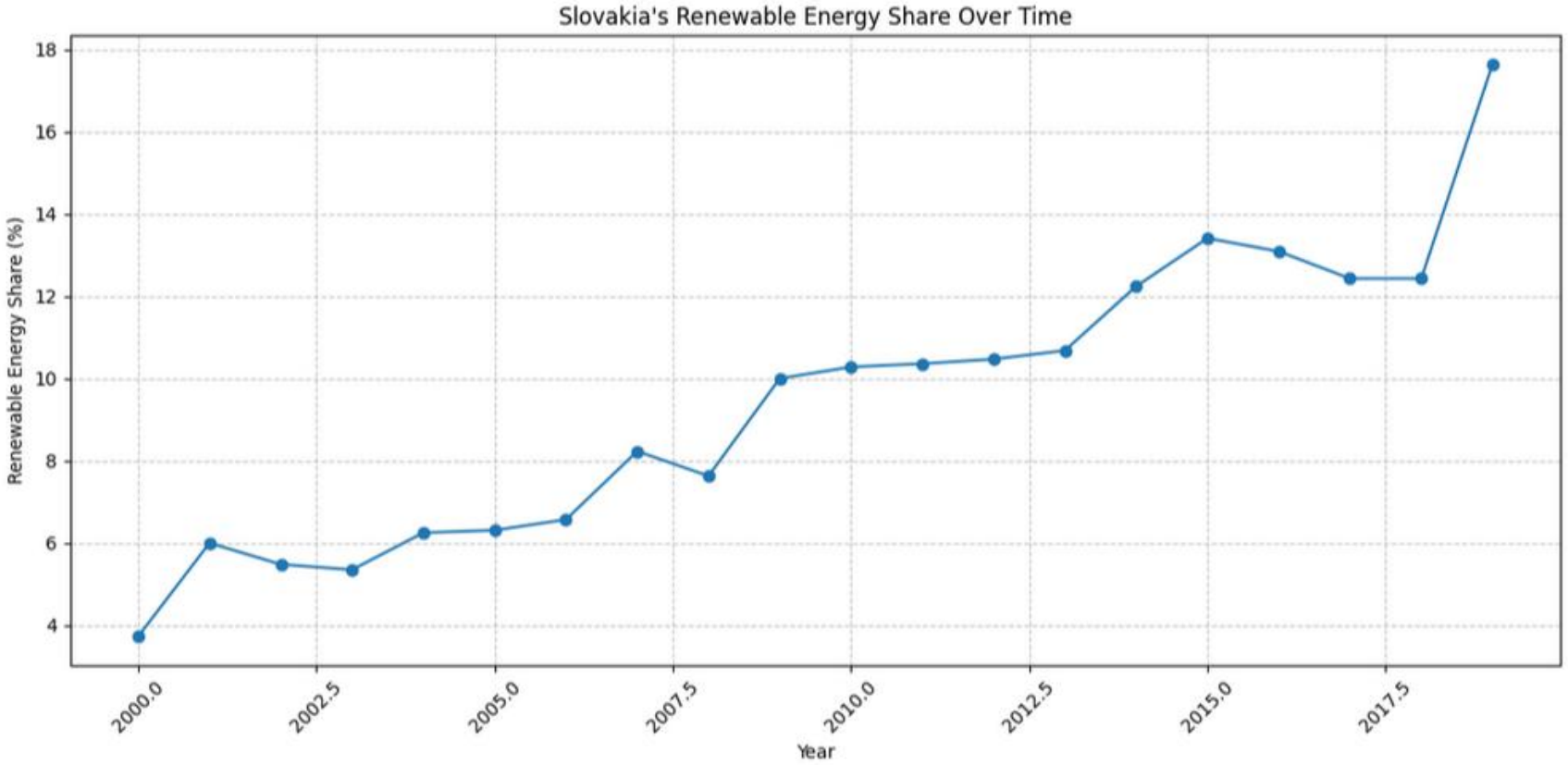
```python
#Distribution of the energy production in Slovakia over time
years = [2015, 2020, 2021, 2022, 2023]
nuclear = [52, 54, 52, 59.3, 62]
hydro = [15, 14, 15, 13.3, 16]
gas = [15, 13, 15, 7.8, 8.56]
coal = [7, 7, 7, 7.7, 2.73]
biofuels = [7, 8, 7, 5.6, 4.14]
solar = [2, 2, 2, 2.4, 2.57]

plt.figure(figsize=(12, 6))
plt.plot(years, nuclear, marker='o', label='Nuclear')
plt.plot(years, hydro, marker='s', label='Hydro')
plt.plot(years, gas, marker='^', label='Gas')
plt.plot(years, coal, marker='D', label='Coal')
plt.plot(years, biofuels, marker='*', label='Biofuels')
plt.plot(years, solar, marker='p', label='Solar')

plt.title("Evolution of Energy Sources in Slovakia (2015-2023)")
plt.xlabel("Year")
plt.ylabel("Share of Electricity Generation (%)")
plt.legend()
plt.grid(True, linestyle='--', alpha=0.7)
plt.tight_layout()
plt.show()
```



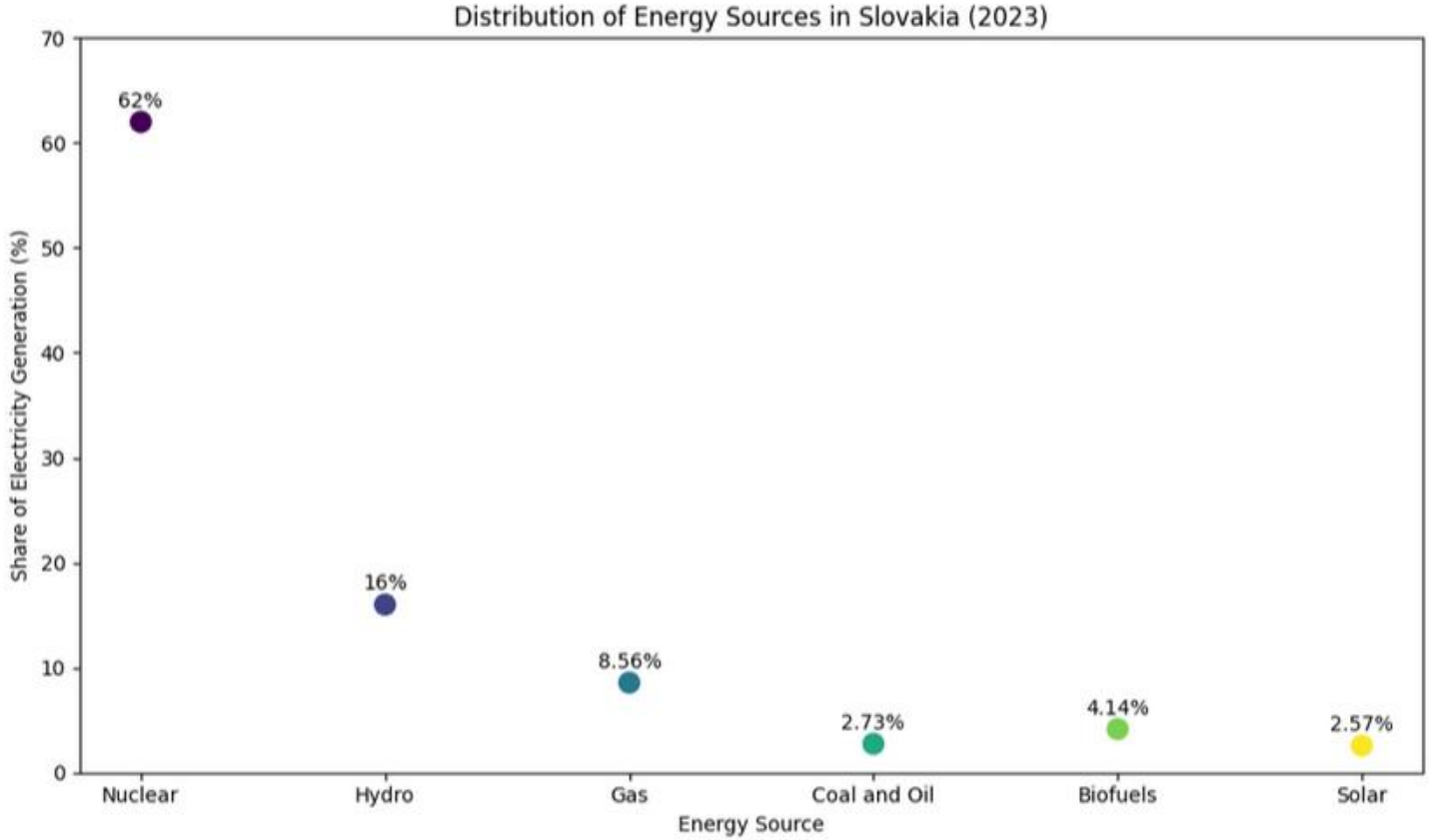
```python
import matplotlib.pyplot as plt
import pandas as pd

# Filter the data for Slovakia and the years 2018-2023
slovakia_data = df[(df['Entity'] == 'Slovakia') & (df['Year'].between(2018, 2023))]

# Create the line plot
plt.figure(figsize=(12, 6))

# List of energy sources to plot
energy_sources = [
    'Electricity from nuclear (TWh)',
    'Electricity from renewables (TWh)',  # This includes hydro, solar, biofuels, etc.
    'Electricity from fossil fuels (TWh)',  # This includes gas, coal, and oil
    'Electricity from gas (TWh)',
    'Electricity from coal (TWh)',
    'Electricity from biofuels (TWh)',
    'Electricity from oil (TWh)'
]

# Plot each energy source
for source in energy_sources:
    if source in slovakia_data.columns:
        plt.plot(slovakia_data['Year'], slovakia_data[source], marker='o', label=source.split(' ')[2] if len(source.split(' ')) > 2 else source)

plt.title("Evolution of Energy Sources in Slovakia (2018-2023)")
plt.xlabel("Year")
plt.ylabel("Electricity Generation (TWh)")
plt.legend()
plt.grid(True, linestyle='--', alpha=0.7)
plt.tight_layout()
plt.show()

# Print the data for verification
print(slovakia_data[['Year'] + [source for source in energy_sources if source in slovakia_data.columns]])
```



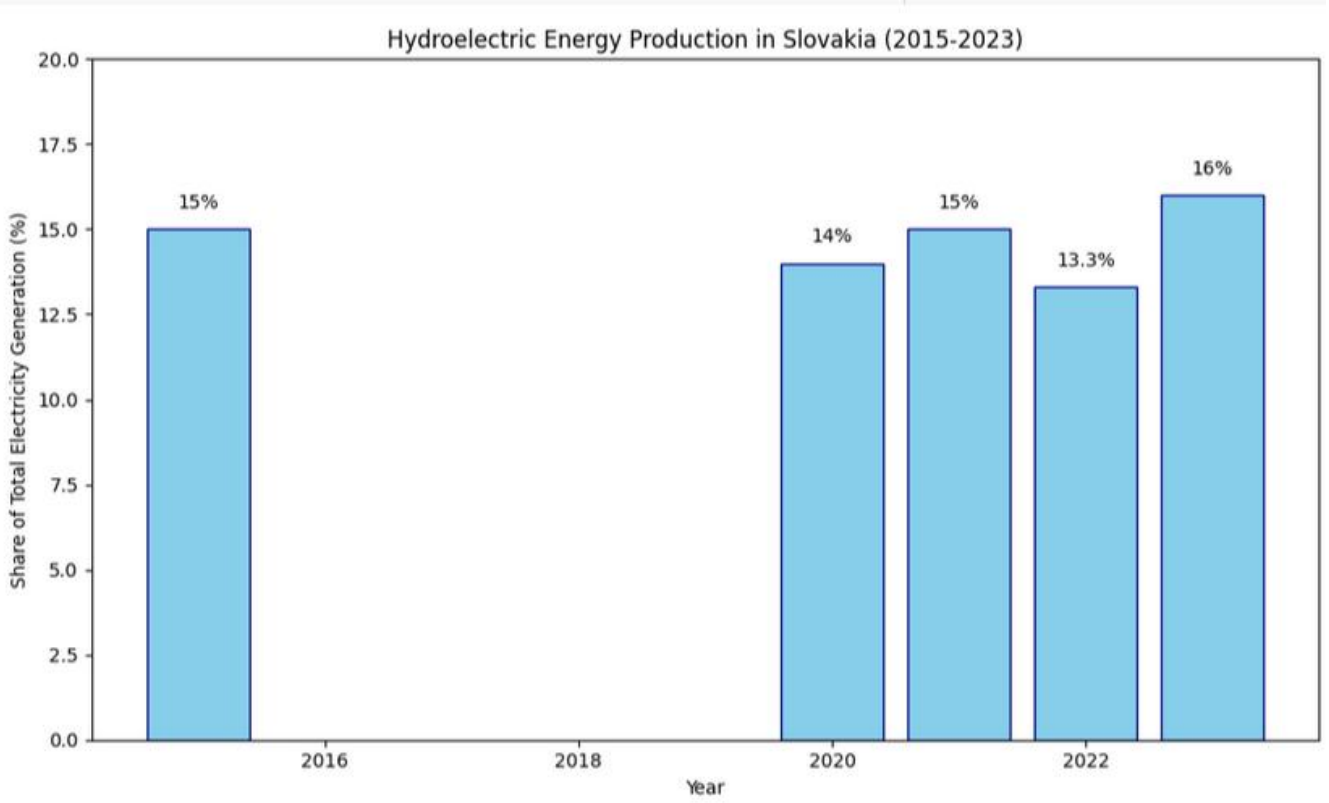
```python
#Slovakia Renewable Energy Share over time
# Filter data for Slovakia
slovakia_data = df[df['Entity'] == 'Slovakia']

# Sort data by year
slovakia_data = slovakia_data.sort_values('Year')

# Create the line chart
plt.figure(figsize=(12, 6))
plt.plot(slovakia_data['Year'], slovakia_data['Renewable energy share in the total final energy consumption (%)'], marker='o')

plt.title("Slovakia's Renewable Energy Share Over Time")
plt.xlabel('Year')
plt.ylabel('Renewable Energy Share (%)')
plt.grid(True, linestyle='--', alpha=0.7)

# Improve x-axis readability
plt.xticks(rotation=45)
plt.tight_layout()
plt.show()
```



```python
# Data for Slovakia's energy sources in 2023
sources = ['Nuclear', 'Hydro', 'Gas', 'Coal and Oil', 'Biofuels', 'Solar']
percentages = [62, 16, 8.56, 2.73, 4.14, 2.57]

plt.figure(figsize=(10, 6))
plt.scatter(sources, percentages, s=100, c=range(len(sources)), cmap='viridis')

plt.title("Distribution of Energy Sources in Slovakia (2023)")
plt.xlabel("Energy Source")
plt.ylabel("Share of Electricity Generation (%)")
plt.ylim(0, 70)

for i, txt in enumerate(percentages):
    plt.annotate(f'{txt}%', (sources[i], percentages[i]), xytext=(0, 5),
                 textcoords='offset points', ha='center', va='bottom')

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



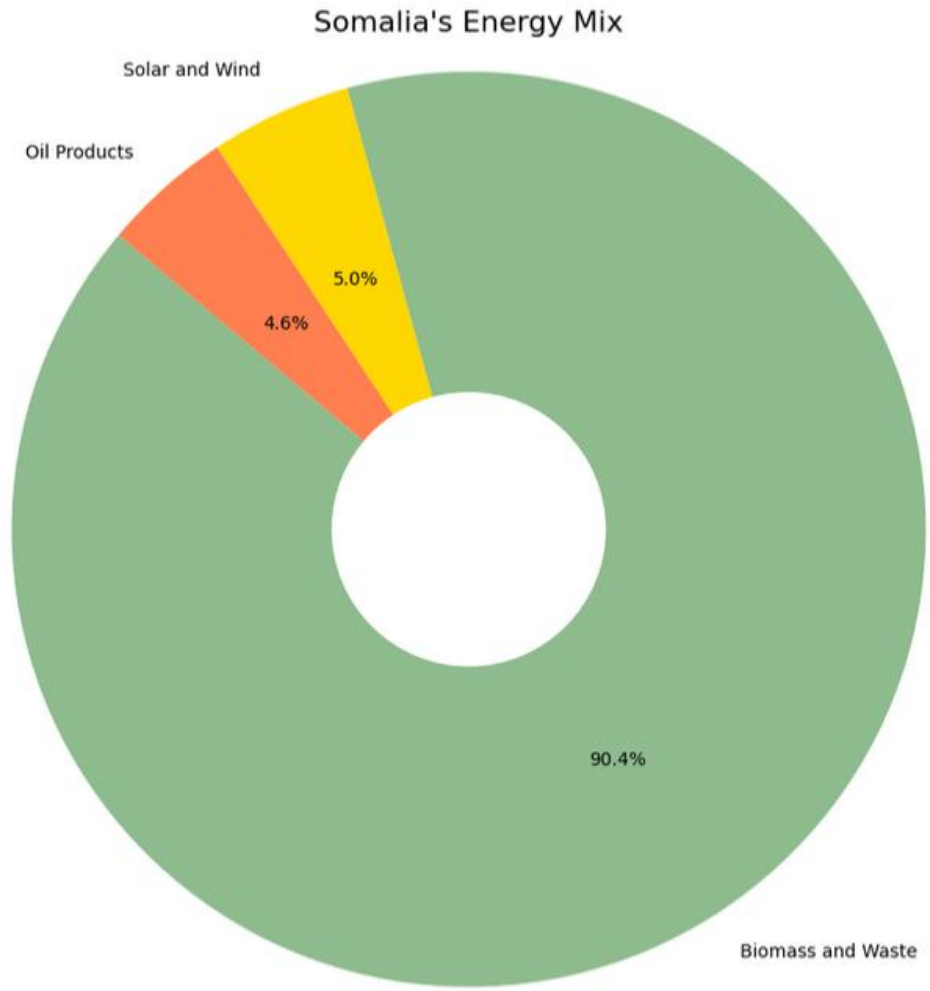
```python
# Hydroelectric Energy Production in Slovakia over time
years = [2015, 2020, 2021, 2022, 2023]
hydro_production = [15, 14, 15, 13.3, 16]

plt.figure(figsize=(10, 6))
plt.bar(years, hydro_production, width=0.8, color='skyblue', edgecolor='navy')

plt.title("Hydroelectric Energy Production in Slovakia (2015-2023)")
plt.xlabel("Year")
plt.ylabel("Share of Total Electricity Generation (%)")
plt.ylim(0, 20)

for i, v in enumerate(hydro_production):
    plt.text(years[i], v + 0.5, f'{v}%', ha='center', va='bottom')

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



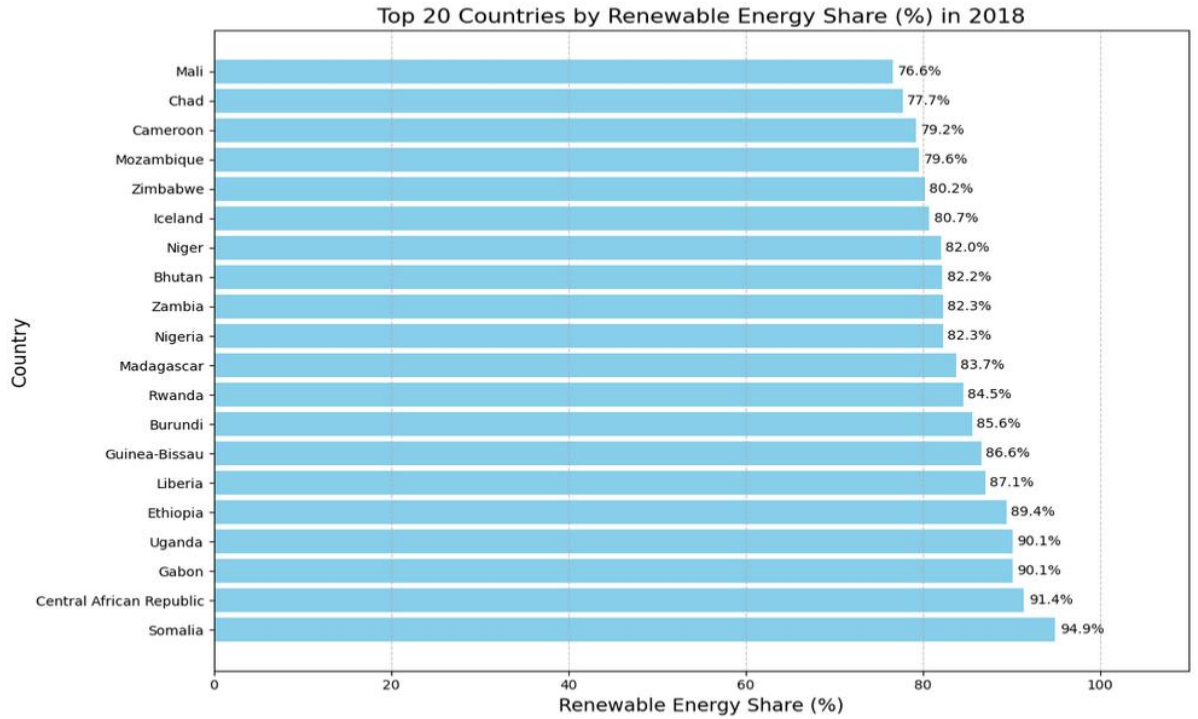
```python
import pandas as pd

def top_5_renewable_contributors(df):
    # Group by 'Entity' and get the maximum renewable energy share for each country
    max_renewable = df.groupby('Entity')['Renewable energy share in the total final energy consumption (%)'].max()

    # Sort in descending order and select top 5
    top_5 = max_renewable.sort_values(ascending=False).head(5)

    return top_5

# Call the function and display results
top_5_results = top_5_renewable_contributors(df)
print(top_5_results)
```

```
Entity
Burundi                     96.04
Ethiopia                    95.55
Uganda                      95.35
Central African Republic    95.08
Somalia                     95.03
Name: Renewable energy share in the total final energy consumption (%), dtype: float64
```

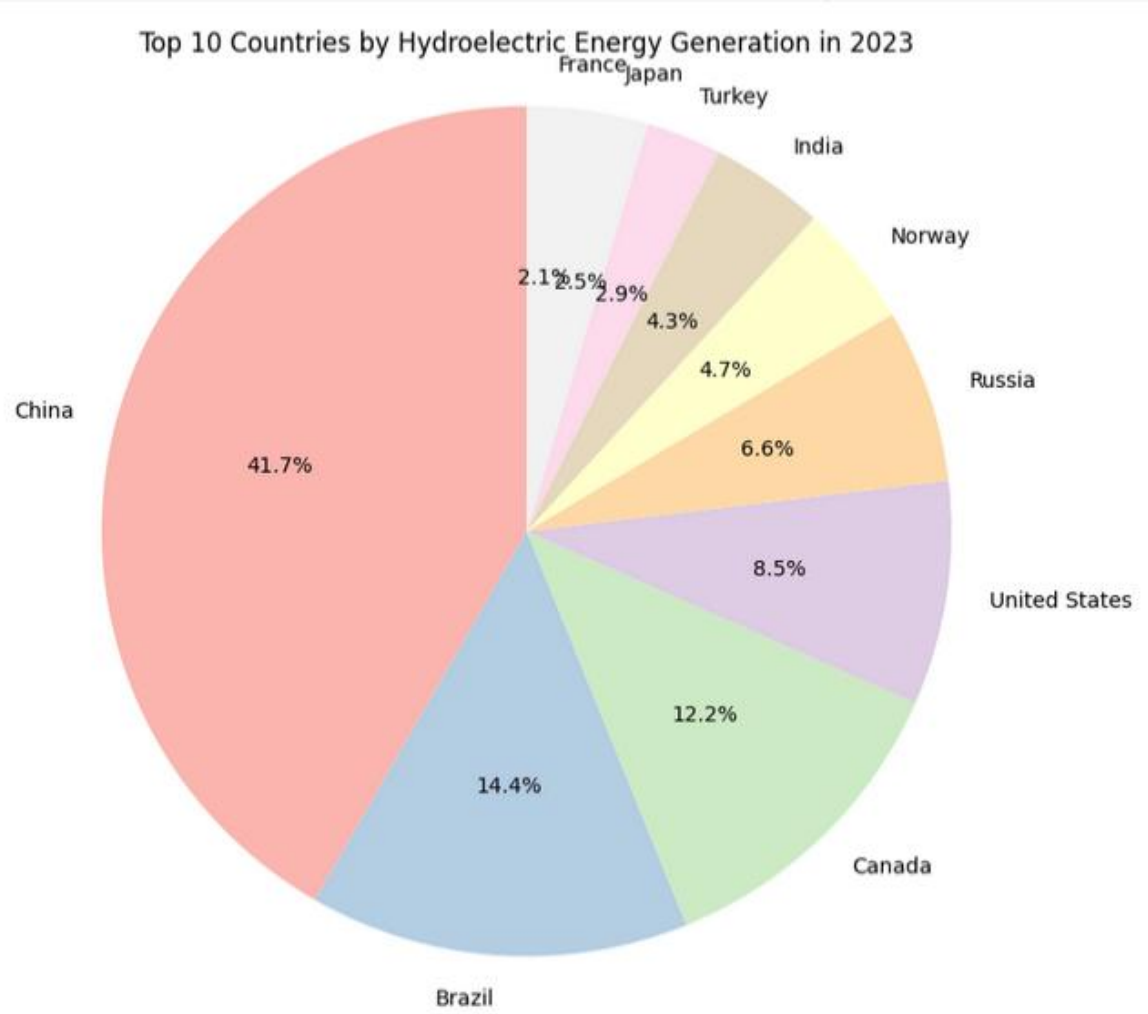
```python
#Pie chart visualization of the energy production in Somalia
# Data for Somalia's energy mix
energy_types = ['Biomass and Waste', 'Solar and Wind', 'Oil Products']
percentages = [90.4, 5, 4.6]  # Estimated breakdown

# Colors for the pie chart
colors = ['#8fbc8f', '#ffd700', '#ff7f50']

# Create the pie chart
plt.figure(figsize=(10, 8))
plt.pie(percentages, labels=energy_types, colors=colors, autopct='%1.1f%%', startangle=140)
plt.title("Somalia's Energy Mix", fontsize=16)

# Add a circle at the center to make it a donut chart (optional)
centre_circle = plt.Circle((0,0), 0.30, fc='white')
fig = plt.gcf()
fig.gca().add_artist(centre_circle)

# Equal aspect ratio ensures that pie is drawn as a circle
plt.axis('equal')
plt.tight_layout()
plt.show()
```



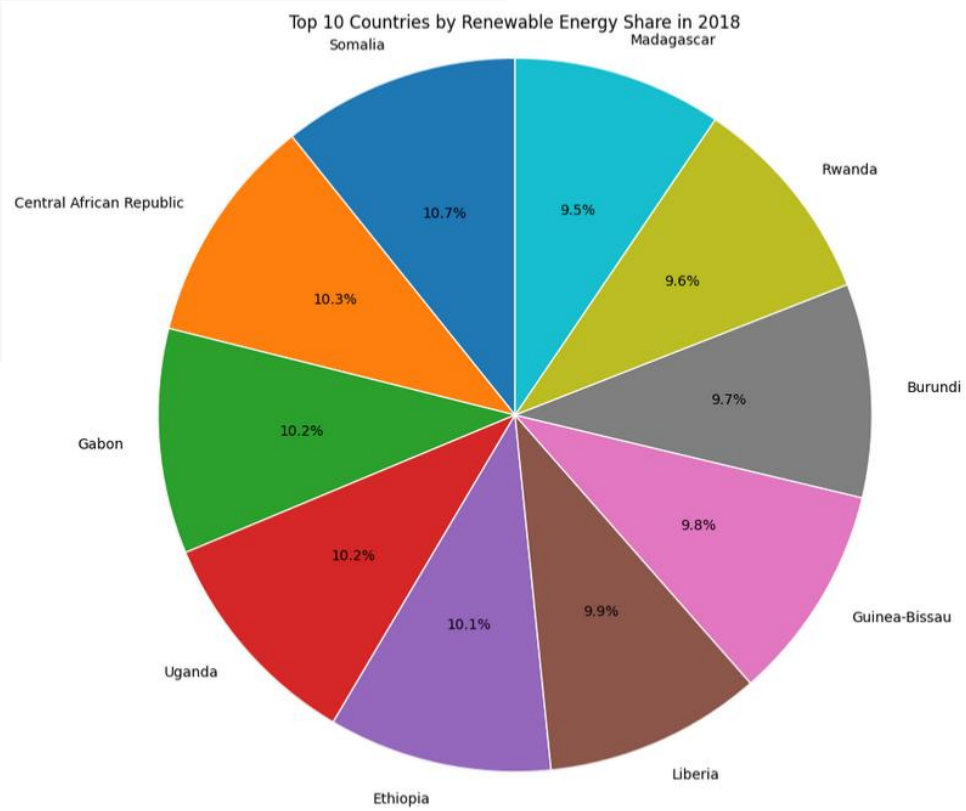
```python
# Filter the data for the year 2018
df_2018 = df[df['Year'] == 2018]

# Sort by Renewable Energy Share in descending order and get top 20
top_20 = df_2018.sort_values('Renewable energy share in the total final energy consumption (%)', ascending=False).head(20)

# Create a bar chart
plt.figure(figsize=(12, 8))
plt.barh(top_20['Entity'], top_20['Renewable energy share in the total final energy consumption (%)'], color='skyblue')
plt.title('Top 20 Countries by Renewable Energy Share (%) in 2018', fontsize=16)
plt.xlabel('Renewable Energy Share (%)', fontsize=14)
plt.ylabel('Country', fontsize=14)

# Add value labels to each bar
for index, value in enumerate(top_20['Renewable energy share in the total final energy consumption (%)']):
    plt.text(value + 0.5, index, f'{value:.1f}%', va='center')

plt.xlim(0, 110)  # Set x-axis limit to accommodate labels
plt.grid(axis='x', linestyle='--', alpha=0.7)
plt.tight_layout()
plt.show()

# Print the data for verification
print(top_20[['Entity', 'Renewable energy share in the total final energy consumption (%)']])
```



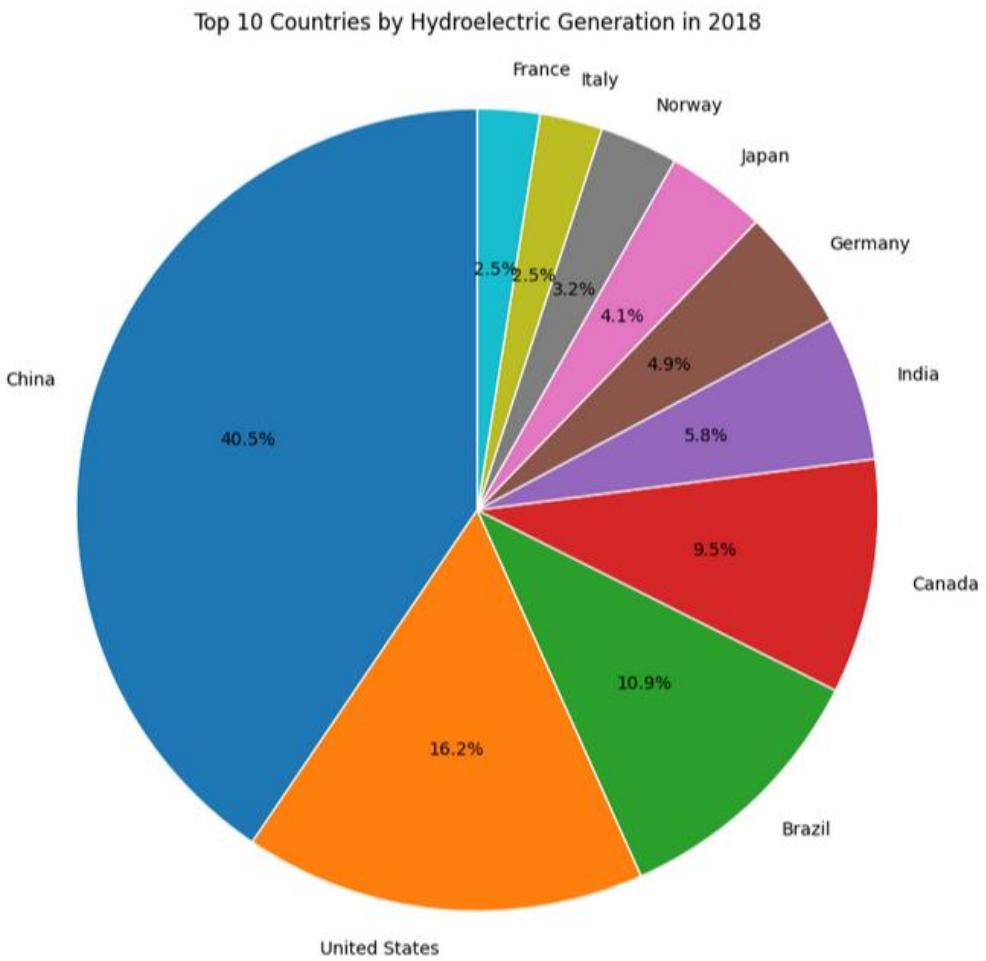
```python
# Data for top 10 countries using hydroelectric energy in 2023
countries = ['China', 'Brazil', 'Canada', 'United States', 'Russia', 'Norway', 'India', 'Turkey', 'Japan', 'France']
hydro_generation = [1245.17, 431.28, 365.39, 254.79, 197.41, 139.23, 129.75, 85.22, 74.38, 63.58]

# Create a pie chart
plt.figure(figsize=(12, 8))
plt.pie(hydro_generation, labels=countries, autopct='%1.1f%%', startangle=90, colors=plt.cm.Pastel1(np.arange(len(countries))))
plt.title('Top 10 Countries by Hydroelectric Energy Generation in 2023')
plt.axis('equal')
plt.show()
```



```python
import pandas as pd
import matplotlib.pyplot as plt

# Setting up the value of the year
year = 2018

# Filter the data for the specified year
data_for_year = df[df['Year'] == year]

# Sort and get top 10 countries by renewable energy share
data_for_chart = data_for_year.sort_values('Renewable energy share in the total final energy consumption (%)', ascending=False).head(10)

# Create the pie chart
plt.figure(figsize=(12, 8))
plt.pie(data_for_chart['Renewable energy share in the total final energy consumption (%)'],
        labels=data_for_chart['Entity'],
        autopct='%1.1f%%',
        startangle=90,
        wedgeprops={'edgecolor': 'white'})

plt.title(f'Top 10 Countries by Renewable Energy Share in {year}')
plt.axis('equal')  # Equal aspect ratio ensures that pie is drawn as a circle
plt.tight_layout()
plt.show()

# Print the data for verification
print(data_for_chart[['Entity', 'Renewable energy share in the total final energy consumption (%)']])
```



```python
import pandas as pd
import matplotlib.pyplot as plt

# Setting up the value of the year
year = 2018

# Filter the data for the specified year
data_for_year = df[df['Year'] == year]

# Sort and get top 10 countries by hydroelectric generation
data_for_chart = data_for_year.sort_values('Electricity from renewables (TWh)', ascending=False).head(10)

# Create the pie chart
plt.figure(figsize=(8, 8))
plt.pie(data_for_chart['Electricity from renewables (TWh)'],
        labels=data_for_chart['Entity'],
        autopct='%1.1f%%',
        startangle=90,
        wedgeprops={'edgecolor': 'white'})

plt.title(f'Top 10 Countries by Hydroelectric Generation in {year}')
plt.axis('equal')  # Equal aspect ratio ensures that pie is drawn as a circle
plt.tight_layout()
plt.show()

# Print the data for verification
print(data_for_chart[['Entity', 'Electricity from renewables (TWh)']])
```



Working with a map

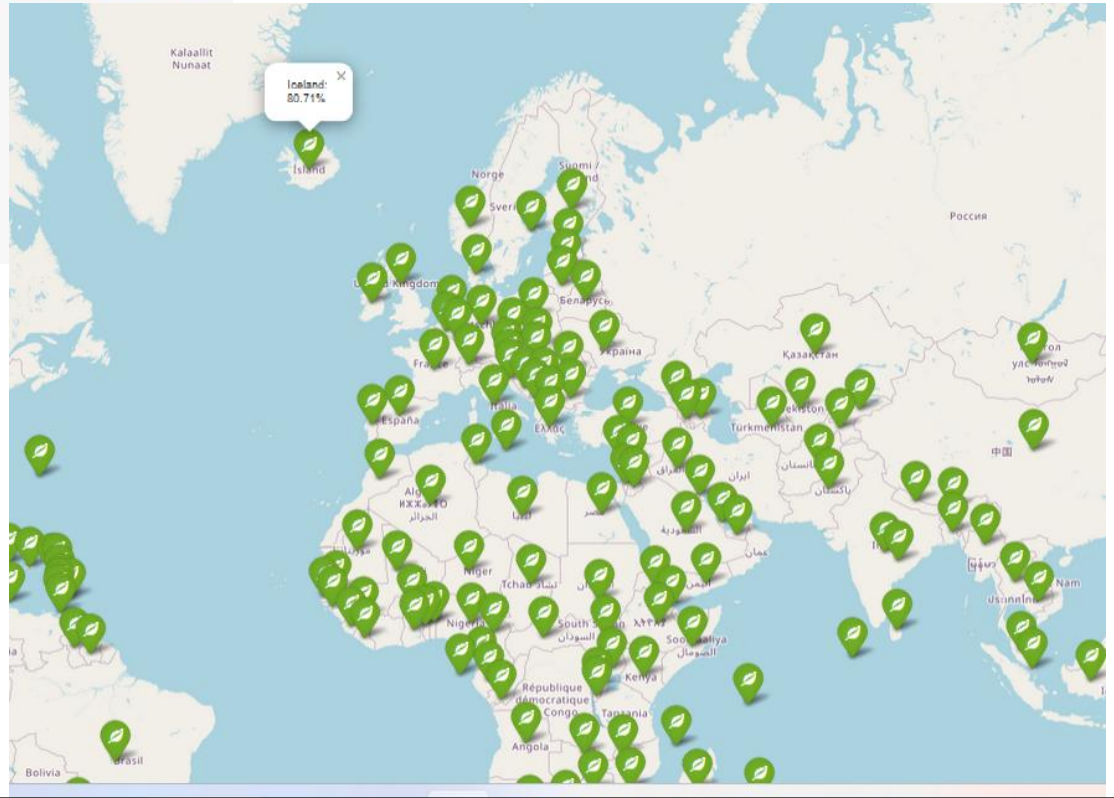
```
[ ]  pip install folium
```

```
Requirement already satisfied: folium in /usr/local/lib/python3.10/dist-packages (0.19.2)
Requirement already satisfied: branca>=0.6.0 in /usr/local/lib/python3.10/dist-packages (from folium) (0.8.1)
Requirement already satisfied: jinja2>=2.9 in /usr/local/lib/python3.10/dist-packages (from folium) (3.1.4)
Requirement already satisfied: numpy in /usr/local/lib/python3.10/dist-packages (from folium) (1.26.4)
Requirement already satisfied: requests in /usr/local/lib/python3.10/dist-packages (from folium) (2.32.3)
Requirement already satisfied: xyzservices in /usr/local/lib/python3.10/dist-packages (from folium) (2024.9.0)
Requirement already satisfied: MarkupSafe>=2.0 in /usr/local/lib/python3.10/dist-packages (from jinja2>=2.9->folium) (3.0.2)
Requirement already satisfied: charset-normalizer<4,>=2 in /usr/local/lib/python3.10/dist-packages (from requests->folium) (3.4.0)
Requirement already satisfied: idna<4,>=2.5 in /usr/local/lib/python3.10/dist-packages (from requests->folium) (3.10)
Requirement already satisfied: urllib3<3,>=1.21.1 in /usr/local/lib/python3.10/dist-packages (from requests->folium) (2.2.3)
Requirement already satisfied: certifi>=2017.4.17 in /usr/local/lib/python3.10/dist-packages (from requests->folium) (2024.12.14)
```

```python
import folium
import pandas as pd

# Assuming 'df' contains the relevant data with columns 'Entity' for country names and 'Electricity from nuclear (TWh)' for nuclear power generation.
# Filter the DataFrame for countries with nuclear power production
nuclear_data = df[df['Electricity from nuclear (TWh)'] > 0]

# Create a base map centered around a global view
world_map = folium.Map(location=[20, 0], zoom_start=2)

# Add markers for each country with nuclear power production
for index, row in nuclear_data.iterrows():
    folium.Marker(
        location=[row['Latitude'], row['Longitude']],
        popup=f"{row['Entity']}: {row['Electricity from nuclear (TWh)']} TWh",
        icon=folium.Icon(color='blue', icon='bolt')
    ).add_to(world_map)

# Save the map to an HTML file
world_map.save("nuclear_power_map.html")

# Display the map in a Jupyter Notebook (if applicable)
world_map
```

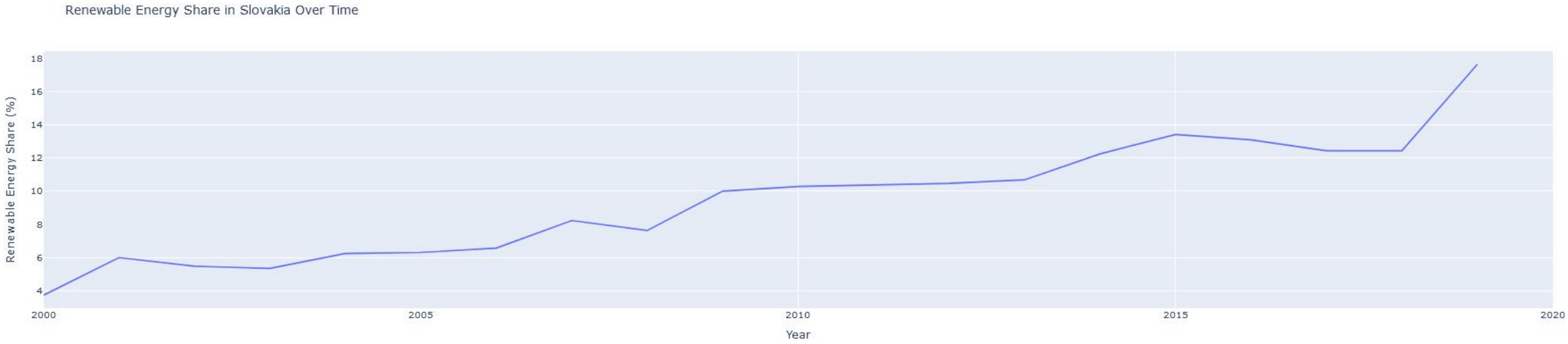
```python
# Assuming 'df' contains the relevant data with columns 'Entity' for country names,
# 'Electricity from renewables (TWh)' for total renewable power generation,
# and 'Latitude' and 'Longitude' for geographical coordinates.
# Filter the DataFrame for countries with hydroelectric generation
# Note: Replace 'Electricity from renewables (TWh)' with the actual column name for hydro generation if different.
hydro_data = df[df['Electricity from renewables (TWh)'] > 0]

# Create a base map centered around a global view
world_map = folium.Map(location=[20, 0], zoom_start=2)

# Add markers for each country with hydroelectric generation
for index, row in hydro_data.iterrows():
    # Assuming you have a specific column for hydro generation, replace if necessary
    hydro_generation = row['Electricity from renewables (TWh)']  # Adjust this if you have a specific column for hydro
    folium.Marker(
        location=[row['Latitude'], row['Longitude']],
        popup=f"{row['Entity']}: {hydro_generation} TWh",
        icon=folium.Icon(color='blue', icon='leaf')
    ).add_to(world_map)

# Save the map to an HTML file
world_map.save("hydro_power_map.html")

# Display the map in a Jupyter Notebook (if applicable)
world_map
```

```python
#Renewable energy share in the total final energy consumption (%) by country for year 2018
import folium
import pandas as pd

# Assuming 'df' contains the relevant data with columns 'Entity' for country names,
# 'Renewable energy share in the total final energy consumption (%)' for investment percentage,
# and 'Latitude' and 'Longitude' for geographical coordinates.
# Filter the DataFrame for countries with renewable energy investment data for 2018
renewable_data_2018 = df[df['Year'] == 2018]

# Create a base map centered around a global view
world_map = folium.Map(location=[20, 0], zoom_start=2)

# Add markers for each country with renewable energy investment
for index, row in renewable_data_2018.iterrows():
    if row['Renewable energy share in the total final energy consumption (%)'] > 0:
        folium.Marker(
            location=[row['Latitude'], row['Longitude']],
            popup=f"{row['Entity']}: {row['Renewable energy share in the total final energy consumption (%)']}%",
            icon=folium.Icon(color='green', icon='leaf')
        ).add_to(world_map)

# Save the map to an HTML file
world_map.save("renewable_energy_investment_map_2018.html")

# Display the map in a Jupyter Notebook (if applicable)
world_map
```



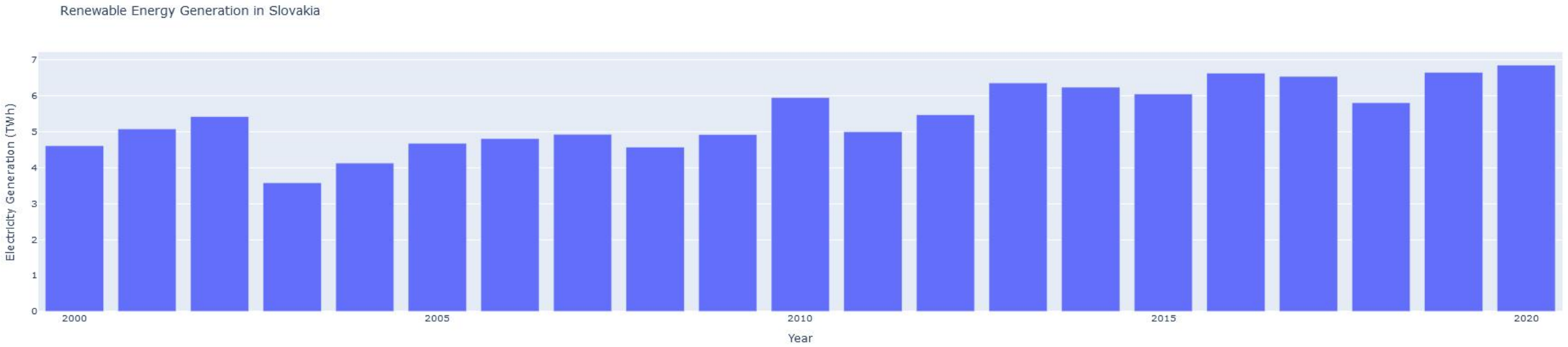
Working with plotly - more interactive visualisations

```
[ ] #line chart with data over time
    import plotly.express as px

    # Filter data for Slovakia
    slovakia_data = df[df['Entity'] == 'Slovakia']

    # Create a line plot of renewable energy share over time
    fig = px.line(slovakia_data,
                  x='Year',
                  y='Renewable energy share in the total final energy consumption (%)',
                  title='Renewable Energy Share in Slovakia Over Time')

    # Customize the layout
    fig.update_layout(
        xaxis_title='Year',
        yaxis_title='Renewable Energy Share (%)',
        hovermode='x unified'
    )

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

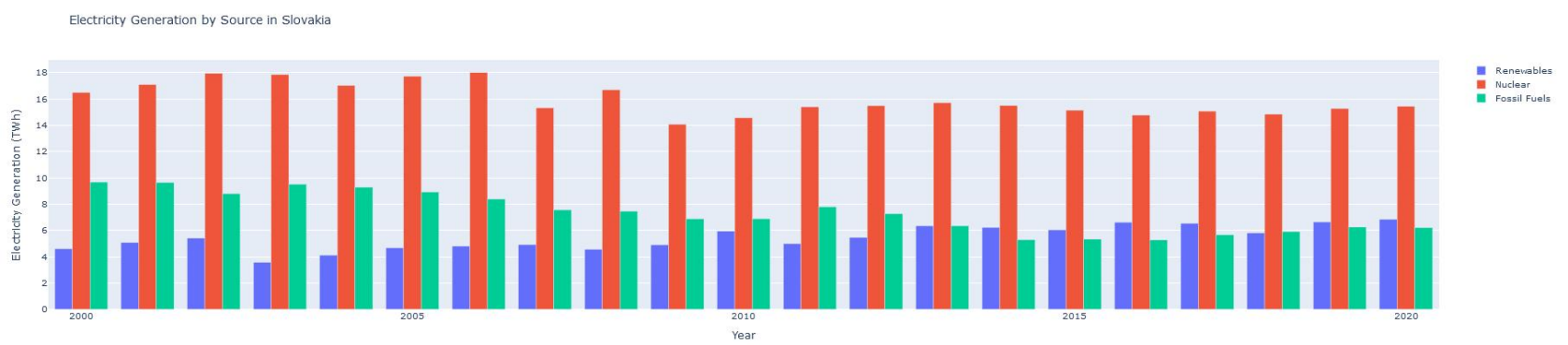


```python
#line chart
import plotly.graph_objects as go

# Filter data for Slovakia
slovakia_data = df[df['Entity'] == 'Slovakia']

# Create a figure with multiple traces
fig = go.Figure()

# Add trace for renewable energy
fig.add_trace(go.Scatter(
    x=slovakia_data['Year'],
    y=slovakia_data['Electricity from renewables (TWh)'],
    mode='lines+markers',
    name='Renewables'
))

# Add trace for nuclear energy
fig.add_trace(go.Scatter(
    x=slovakia_data['Year'],
    y=slovakia_data['Electricity from nuclear (TWh)'],
    mode='lines+markers',
    name='Nuclear'
))

# Add trace for fossil fuels
fig.add_trace(go.Scatter(
    x=slovakia_data['Year'],
    y=slovakia_data['Electricity from fossil fuels (TWh)'],
    mode='lines+markers',
    name='Fossil Fuels'
))

# Update layout
fig.update_layout(
    title='Electricity Generation by Source in Slovakia',
    xaxis_title='Year',
    yaxis_title='Electricity Generation (TWh)',
    hovermode='x unified'
)

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

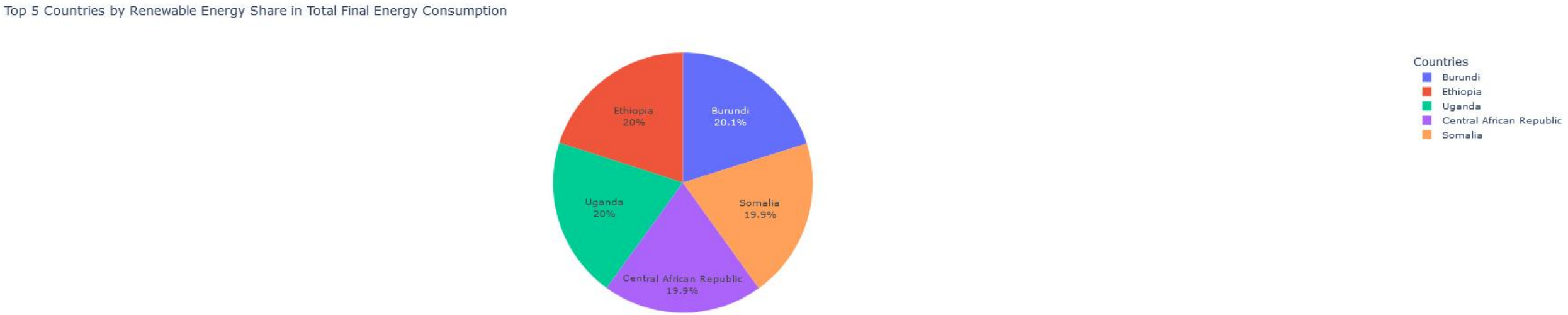
```
] #bar chart
  import plotly.express as px

  # Assuming 'df' is your DataFrame containing Slovakia data
  slovakia_data = df[df['Entity'] == 'Slovakia']

  # Create a bar chart
  fig = px.bar(slovakia_data,
               x='Year',
               y='Electricity from renewables (TWh)',
               title='Renewable Energy Generation in Slovakia')

  # Customize the layout
  fig.update_layout(
      xaxis_title='Year',
      yaxis_title='Electricity Generation (TWh)',
      bargap=0.2
  )

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



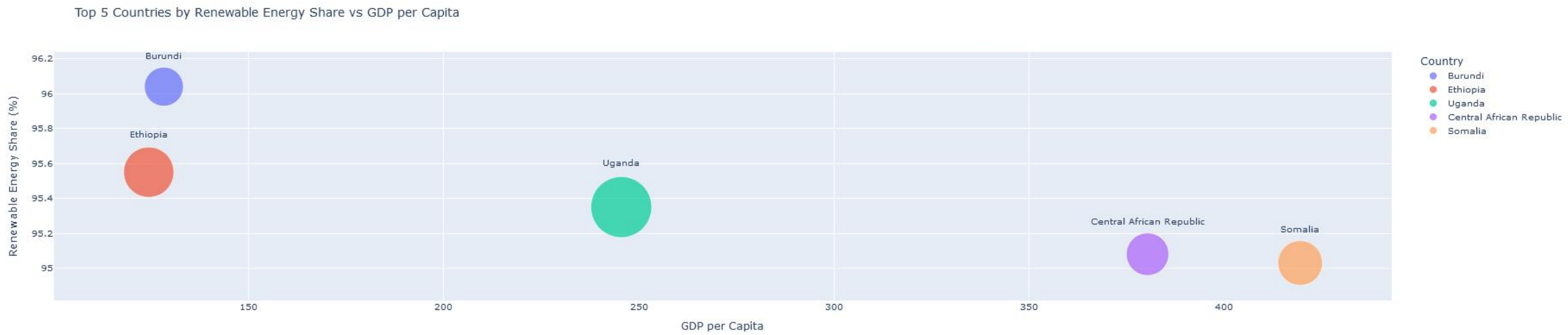
```python
#creating bar chart with multiple bars representing difference source of energy for the exact year
import plotly.graph_objects as go

# Filter data for Slovakia
slovakia_data = df[df['Entity'] == 'Slovakia']

# Create a figure with multiple traces
fig = go.Figure()

# Add trace for renewable energy
fig.add_trace(go.Bar(
    x=slovakia_data['Year'],
    y=slovakia_data['Electricity from renewables (TWh)'],
    name='Renewables'
))

# Add trace for nuclear energy
fig.add_trace(go.Bar(
    x=slovakia_data['Year'],
    y=slovakia_data['Electricity from nuclear (TWh)'],
    name='Nuclear'
))

# Add trace for fossil fuels
fig.add_trace(go.Bar(
    x=slovakia_data['Year'],
    y=slovakia_data['Electricity from fossil fuels (TWh)'],
    name='Fossil Fuels'
))

# Update layout
fig.update_layout(
    title='Electricity Generation by Source in Slovakia',
    xaxis_title='Year',
    yaxis_title='Electricity Generation (TWh)',
    barmode='group'
)

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



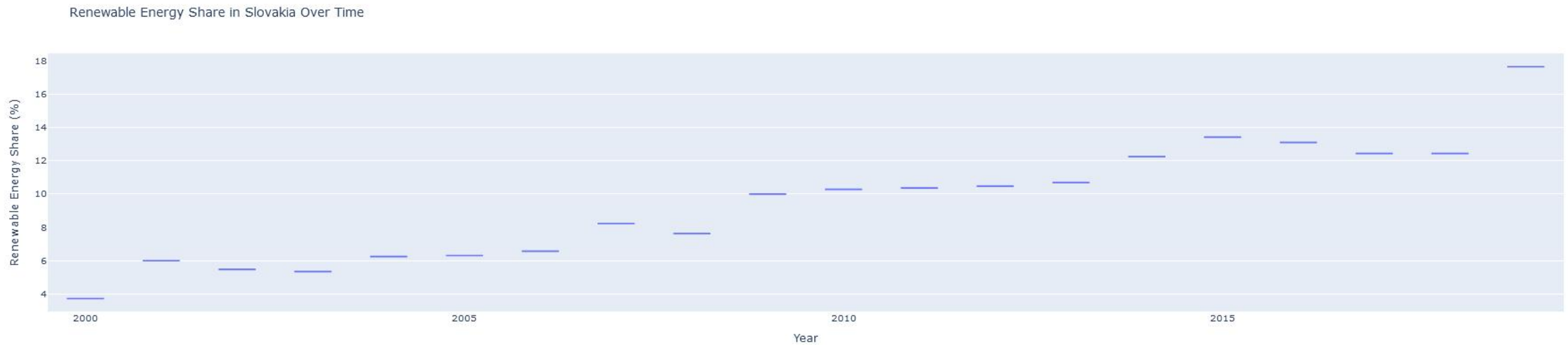
```python
import plotly.express as px

# Sort the DataFrame by renewable energy share and select top 5 countries
top_5_countries = df.sort_values('Renewable energy share in the total final energy consumption (%)', ascending=False).drop_duplicates('Entity').head(5)

# Create the pie chart
fig = px.pie(top_5_countries,
             values='Renewable energy share in the total final energy consumption (%)',
             names='Entity',
             title='Top 5 Countries by Renewable Energy Share in Total Final Energy Consumption')

# Customize the layout
fig.update_traces(textposition='inside', textinfo='percent+label')
fig.update_layout(
    legend_title_text='Countries',
    uniformtext_minsize=12,
    uniformtext_mode='hide'
)

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



```python
#Creating ScatterPlot
# Sort the DataFrame by renewable energy share and select top 5 countries
top_5_countries = df.sort_values('Renewable energy share in the total final energy consumption (%)', ascending=False).drop_duplicates('Entity').head(5)

# Create the scatter plot
fig = px.scatter(top_5_countries,
                 x='gdp_per_capita',
                 y='Renewable energy share in the total final energy consumption (%)',
                 size='Primary energy consumption per capita (kWh/person)',
                 color='Entity',
                 hover_name='Entity',
                 text='Entity',
                 size_max=60,
                 title='Top 5 Countries by Renewable Energy Share vs GDP per Capita')

# Customize the layout
fig.update_traces(textposition='top center')
fig.update_layout(
    xaxis_title='GDP per Capita',
    yaxis_title='Renewable Energy Share (%)',
    legend_title='Country'
)

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

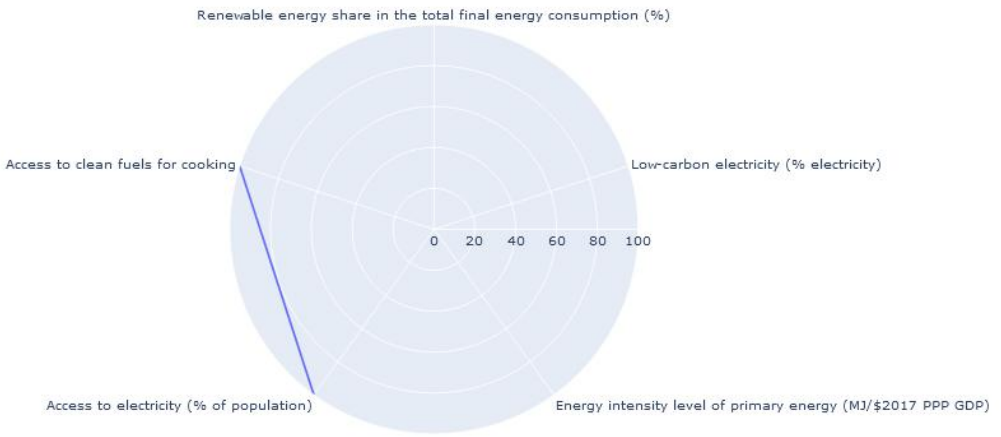


```python
#boxplot
# Filter data for Slovakia
slovakia_data = df[df['Entity'] == 'Slovakia']

# Create the boxplot
fig = px.box(slovakia_data,
             x='Year',
             y='Renewable energy share in the total final energy consumption (%)',
             title='Renewable Energy Share in Slovakia Over Time')

# Customize the layout
fig.update_layout(
    xaxis_title='Year',
    yaxis_title='Renewable Energy Share (%)',
    showlegend=False
)

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

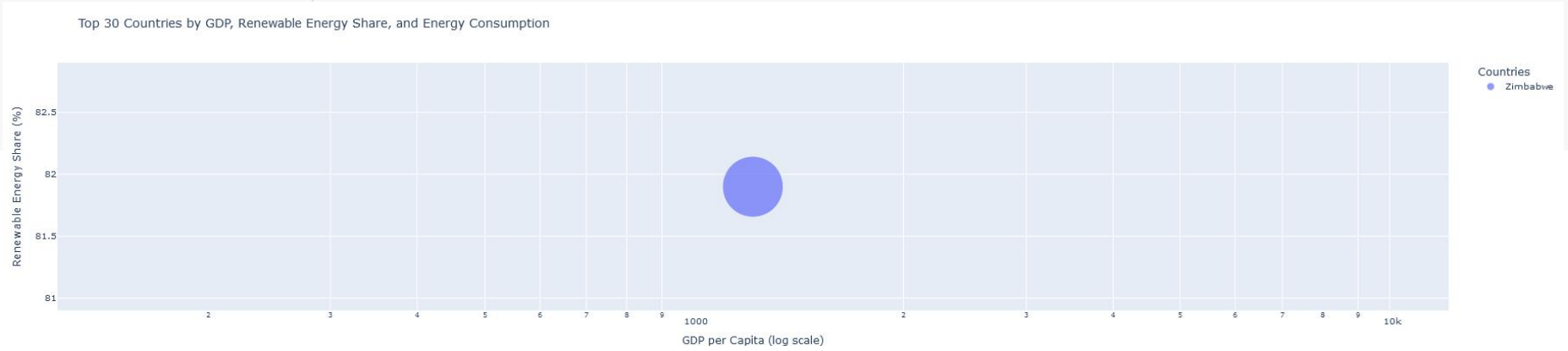


```python
#Radar chart
import plotly.express as px

# Filter data for Slovakia
slovakia_data = df[df['Entity'] == 'Slovakia'].iloc[-1]  # Get the most recent year

# Select relevant features for the radar chart
features = ['Renewable energy share in the total final energy consumption (%)',
            'Low-carbon electricity (% electricity)',
            'Energy intensity level of primary energy (MJ/$2017 PPP GDP)',
            'Access to electricity (% of population)',
            'Access to clean fuels for cooking']

# Create a DataFrame for the radar chart
radar_data = pd.DataFrame(dict(
    r=slovakia_data[features].values,
    theta=features
))

# Create the radar chart
fig = px.line_polar(radar_data, r='r', theta='theta', line_close=True)

# Customize the layout
fig.update_traces(fill='toself')
fig.update_layout(
    title='Energy Profile of Slovakia',
    polar=dict(
        radialaxis=dict(visible=True, range=[0, 100])
    )
)

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

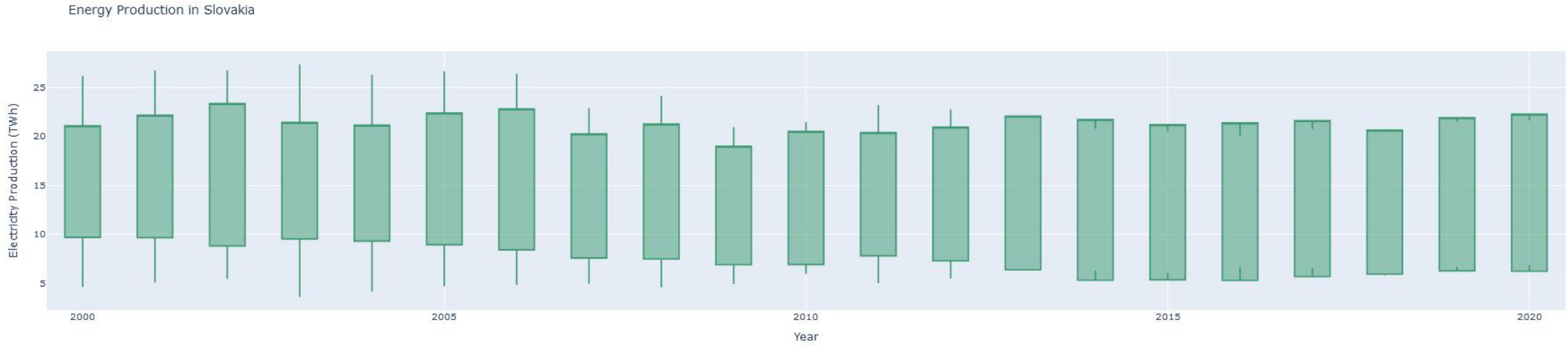
Energy Profile of Slovakia



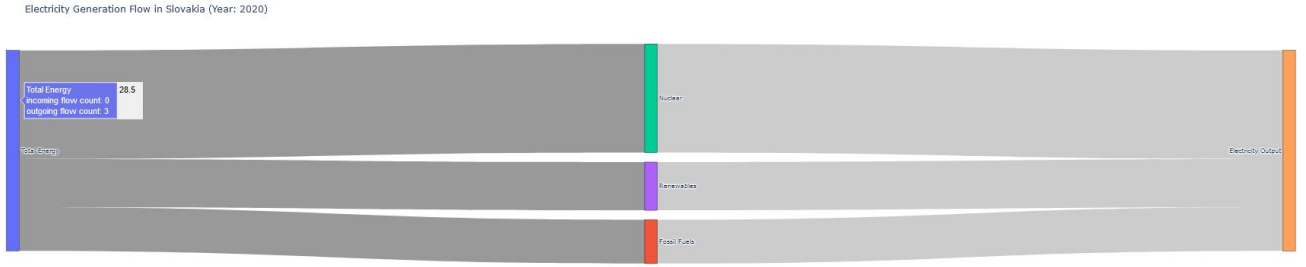
```python
import plotly.express as px

# Filter the data for the most recent year and remove any rows with missing values
recent_data = df[df['Year'] == df['Year'].max()].dropna(subset=['gdp_per_capita', 'Renewable energy share in the total final energy consumption (%)', 'Primary energy consumption per capita (kWh/person)'])

# Select top 30 countries by GDP per capita to reduce clutter
top_30_countries = recent_data.nlargest(30, 'gdp_per_capita')

# Create the bubble chart
fig = px.scatter(top_30_countries,
                 x='gdp_per_capita',
                 y='Renewable energy share in the total final energy consumption (%)',
                 size='Primary energy consumption per capita (kWh/person)',
                 color='Entity',
                 hover_name='Entity',
                 log_x=True,  # Use log scale for GDP per capita
                 size_max=60,  # Maximum size of bubbles
                 title='Top 30 Countries by GDP, Renewable Energy Share, and Energy Consumption')

# Highlight Slovakia if it's in the top 30
slovakia_data = top_30_countries[top_30_countries['Entity'] == 'Slovakia']
if not slovakia_data.empty:
    fig.add_trace(px.scatter(slovakia_data,
                             x='gdp_per_capita',
                             y='Renewable energy share in the total final energy consumption (%)',
                             size='Primary energy consumption per capita (kWh/person)',
                             color_discrete_sequence=['red'],
                             hover_name='Entity').data[0])

# Customize the layout
fig.update_layout(
    xaxis_title='GDP per Capita (log scale)',
    yaxis_title='Renewable Energy Share (%)',
    showlegend=True,
    legend_title_text='Countries'
)

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



```python
#Candle sticks chart
# Filter data for Slovakia
slovakia_data = df[df['Entity'] == 'Slovakia'].sort_values('Year')

# Create the candlestick chart
fig = go.Figure(data=[go.Candlestick(
    x=slovakia_data['Year'],
    open=slovakia_data['Electricity from fossil fuels (TWh)'],
    high=slovakia_data['Electricity from fossil fuels (TWh)'] + slovakia_data['Electricity from nuclear (TWh)'],
    low=slovakia_data['Electricity from renewables (TWh)'],
    close=slovakia_data['Electricity from renewables (TWh)'] + slovakia_data['Electricity from nuclear (TWh)']
)])

# Customize the layout
fig.update_layout(
    title='Energy Production in Slovakia',
    xaxis_title='Year',
    yaxis_title='Electricity Production (TWh)',
    xaxis_rangeslider_visible=False
)

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



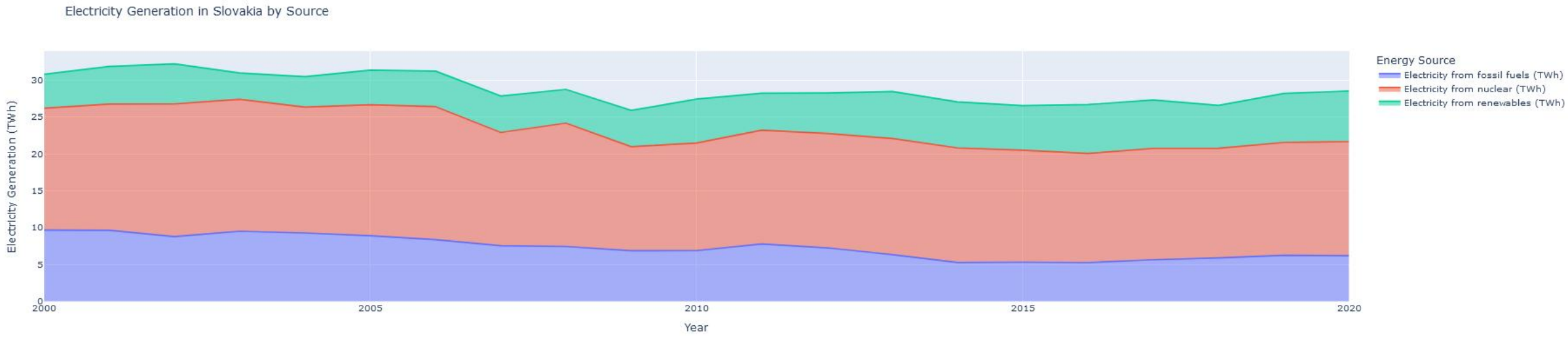
```python
#Using Sankey diagram
import plotly.graph_objects as go
import pandas as pd

# Filter data for Slovakia and get the most recent year
slovakia_data = df[df['Entity'] == 'Slovakia'].sort_values('Year', ascending=False).iloc[0]

# Prepare data for Sankey diagram
label = ["Total Energy", "Fossil Fuels", "Nuclear", "Renewables", "Electricity Output"]
source = [0, 0, 0, 1, 2, 3]
target = [1, 2, 3, 4, 4, 4]
value = [
    slovakia_data['Electricity from fossil fuels (TWh)'],
    slovakia_data['Electricity from nuclear (TWh)'],
    slovakia_data['Electricity from renewables (TWh)'],
    slovakia_data['Electricity from fossil fuels (TWh)'],
    slovakia_data['Electricity from nuclear (TWh)'],
    slovakia_data['Electricity from renewables (TWh)']
]

# Create color scale
color_scale = ['#636EFA', '#EF553B', '#00CC96', '#AB63FA', '#FFA15A']

# Create the Sankey diagram
fig = go.Figure(data=[go.Sankey(
    node = dict(
      pad = 15,
      thickness = 20,
      line = dict(color = "black", width = 0.5),
      label = label,
      color = color_scale
    ),
    link = dict(
      source = source,
      target = target,
      value = value
  ))])

# Update the layout
fig.update_layout(
    title_text=f"Electricity Generation Flow in Slovakia (Year: {slovakia_data['Year']})",
    font_size=10
)

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



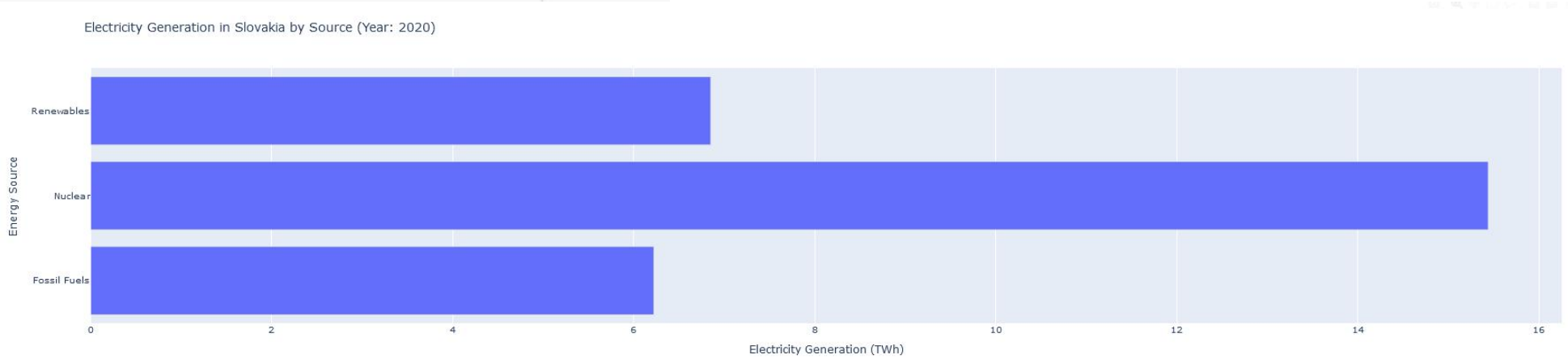
```python
#Using Treemap to visualize data
# Filter data for Slovakia and get the most recent year
slovakia_data = df[df['Entity'] == 'Slovakia'].sort_values('Year', ascending=False).iloc[0]

# Prepare data for Treemap
treemap_data = pd.DataFrame({
    'Energy Source': ['Total', 'Fossil Fuels', 'Nuclear', 'Renewables'],
    'Parent': ['', 'Total', 'Total', 'Total'],
    'Value': [
        slovakia_data['Electricity from fossil fuels (TWh)'] +
        slovakia_data['Electricity from nuclear (TWh)'] +
        slovakia_data['Electricity from renewables (TWh)'],
        slovakia_data['Electricity from fossil fuels (TWh)'],
        slovakia_data['Electricity from nuclear (TWh)'],
        slovakia_data['Electricity from renewables (TWh)']
    ]
})

# Create the Treemap
fig = px.treemap(treemap_data,
                 path=['Parent', 'Energy Source'],
                 values='Value',
                 title=f"Electricity Generation in Slovakia (Year: {slovakia_data['Year']})")

fig.update_traces(root_color="lightgrey")
fig.update_layout(margin=dict(t=50, l=25, r=25, b=25))

fig.show()
```

```python
#Dot plot
# Extract relevant data for Slovakia
years = [2018, 2019, 2020, 2021, 2022]
electricity_generation = [27.15, 28.61, 29.02, 31.11, 28.3]

# Create the dot plot
fig = go.Figure()

fig.add_trace(go.Scatter(
    x=years,
    y=electricity_generation,
    mode='markers',
    marker=dict(
        size=10,
        color=electricity_generation,
        colorscale='Viridis',
        showscale=True
    ),
    text=[f'{y:.2f} TWh' for y in electricity_generation],
    hoverinfo='text+x'
))

# Customize the layout
fig.update_layout(
    title='Electricity Generation in Slovakia (2018-2022)',
    xaxis_title='Year',
    yaxis_title='Electricity Generation (TWh)',
    xaxis=dict(tickmode='linear'),
    yaxis=dict(range=[25, 32])
)

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

```python
#Area Plot
# Filter data for Slovakia
slovakia_data = df[df['Entity'] == 'Slovakia'].sort_values('Year')

# Create the filled area plot
fig = px.area(slovakia_data,
              x='Year',
              y=['Electricity from fossil fuels (TWh)',
                 'Electricity from nuclear (TWh)',
                 'Electricity from renewables (TWh)'],
              title='Electricity Generation in Slovakia by Source',
              labels={'value': 'Electricity Generation (TWh)', 'variable': 'Source'})

# Customize the layout
fig.update_layout(
    xaxis_title='Year',
    yaxis_title='Electricity Generation (TWh)',
    legend_title='Energy Source'
)

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



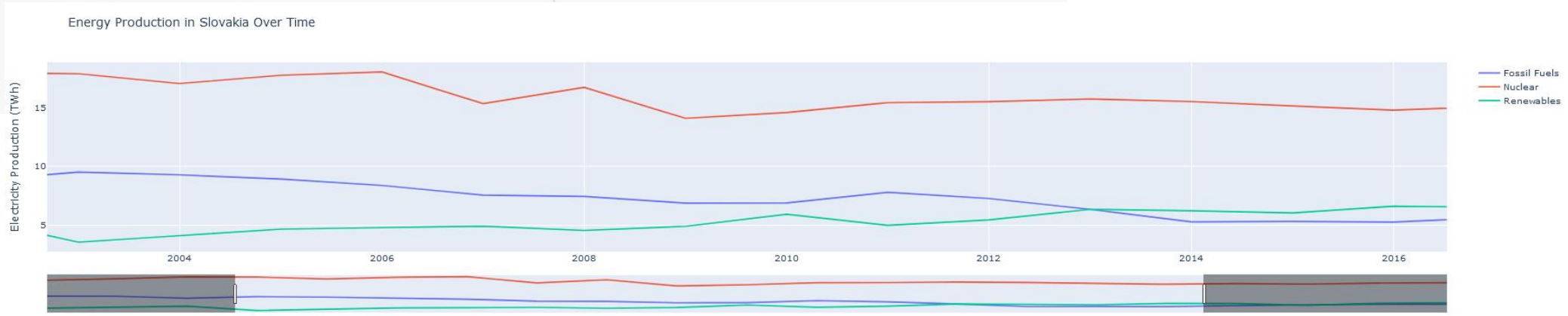
```python
#Horizontal bar chart
# Filter data for Slovakia and get the most recent year
slovakia_data = df[df['Entity'] == 'Slovakia'].sort_values('Year', ascending=False).iloc[0]

# Prepare data for horizontal bar chart
energy_sources = ['Fossil Fuels', 'Nuclear', 'Renewables']
values = [
    slovakia_data['Electricity from fossil fuels (TWh)'],
    slovakia_data['Electricity from nuclear (TWh)'],
    slovakia_data['Electricity from renewables (TWh)']
]

# Create the horizontal bar chart
fig = px.bar(
    x=values,
    y=energy_sources,
    orientation='h',
    title=f"Electricity Generation in Slovakia by Source (Year: {slovakia_data['Year']})"
)

# Customize the layout
fig.update_layout(
    xaxis_title='Electricity Generation (TWh)',
    yaxis_title='Energy Source'
)

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



```
] pip install --upgrade plotly
```

```
Requirement already satisfied: plotly in /usr/local/lib/python3.10/dist-packages (5.24.1)
Requirement already satisfied: tenacity>=6.2.0 in /usr/local/lib/python3.10/dist-packages (from plotly) (9.0.0)
Requirement already satisfied: packaging in /usr/local/lib/python3.10/dist-packages (from plotly) (24.2)
```

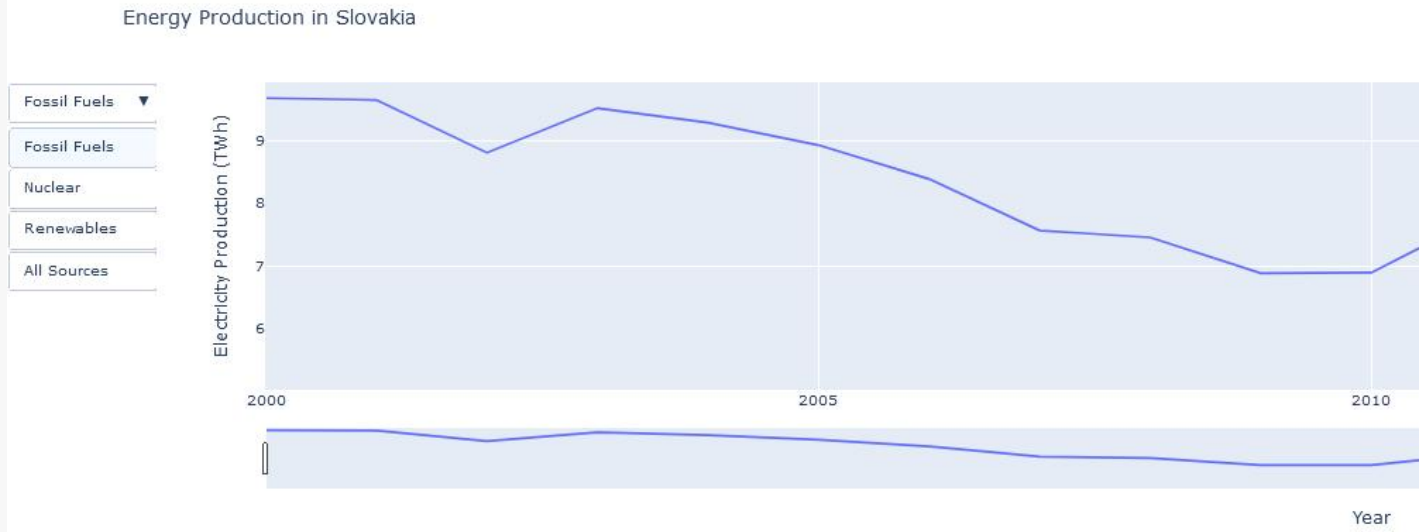
```python
#Creating a chart with sliders
import plotly.graph_objects as go

# Assuming df is your DataFrame with Slovakia data
slovakia_data = df[df['Entity'] == 'Slovakia']

# Create the figure
fig = go.Figure()

# Add traces for different energy sources
fig.add_trace(go.Scatter(x=slovakia_data['Year'], y=slovakia_data['Electricity from fossil fuels (TWh)'], name='Fossil Fuels'))
fig.add_trace(go.Scatter(x=slovakia_data['Year'], y=slovakia_data['Electricity from nuclear (TWh)'], name='Nuclear'))
fig.add_trace(go.Scatter(x=slovakia_data['Year'], y=slovakia_data['Electricity from renewables (TWh)'], name='Renewables'))

# Update layout with slider
fig.update_layout(
    title='Energy Production in Slovakia Over Time',
    xaxis=dict(
        rangeslider=dict(visible=True),
        type='date'
    ),
    yaxis=dict(title='Electricity Production (TWh)'),
    hovermode='x unified'
)

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



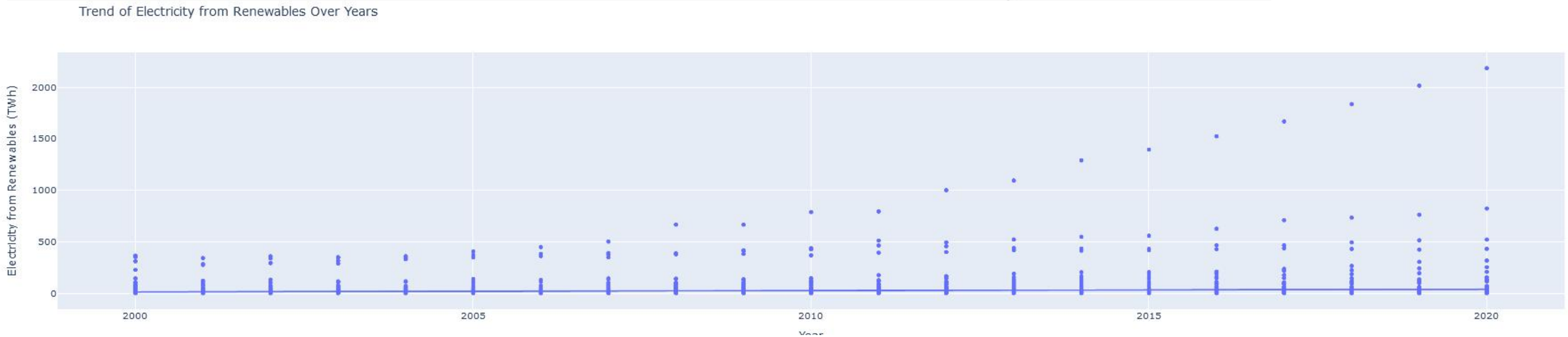
```python
import pandas as pd
import plotly.graph_objects as go

# Filter data for Slovakia
slovakia_data = df[df['Entity'] == 'Slovakia']

# Create the figure
fig = go.Figure()

# Add traces for each energy source
fig.add_trace(go.Scatter(
    x=slovakia_data['Year'],
    y=slovakia_data['Electricity from fossil fuels (TWh)'],
    name='Fossil Fuels',
    visible=True  # Initially visible
))

fig.add_trace(go.Scatter(
    x=slovakia_data['Year'],
    y=slovakia_data['Electricity from nuclear (TWh)'],
    name='Nuclear',
    visible=False  # Initially hidden
))

fig.add_trace(go.Scatter(
    x=slovakia_data['Year'],
    y=slovakia_data['Electricity from renewables (TWh)'],
    name='Renewables',
    visible=False  # Initially hidden
))

# Update layout with dropdown menu
fig.update_layout(
    title='Energy Production in Slovakia',
    xaxis_title='Year',
    yaxis_title='Electricity Production (TWh)',
    updatemenus=[
        {
            'buttons': [
                {
                    'label': 'Fossil Fuels',
                    'method': 'update',
                    'args': [{'visible': [True, False, False]}, {'title': 'Fossil Fuels Production'}]
                },
                {
                    'label': 'Nuclear',
                    'method': 'update',
                    'args': [{'visible': [False, True, False]}, {'title': 'Nuclear Production'}
```

```python
                {
                    'buttons': [
                        {
                            'label': 'Fossil Fuels',
                            'method': 'update',
                            'args': [{'visible': [True, False, False]}, {'title': 'Fossil Fuels Production'}]
                        },
                        {
                            'label': 'Nuclear',
                            'method': 'update',
                            'args': [{'visible': [False, True, False]}, {'title': 'Nuclear Production'}]
                        },
                        {
                            'label': 'Renewables',
                            'method': 'update',
                            'args': [{'visible': [False, False, True]}, {'title': 'Renewable Production'}]
                        },
                        {
                            'label': 'All Sources',
                            'method': 'update',
                            'args': [{'visible': [True, True, True]}, {'title': 'Energy Production in Slovakia'}]
                        }
                    ],
                    'direction': 'down',
                    'showactive': True,
                }
            ]
        )

        # Show the plot
```

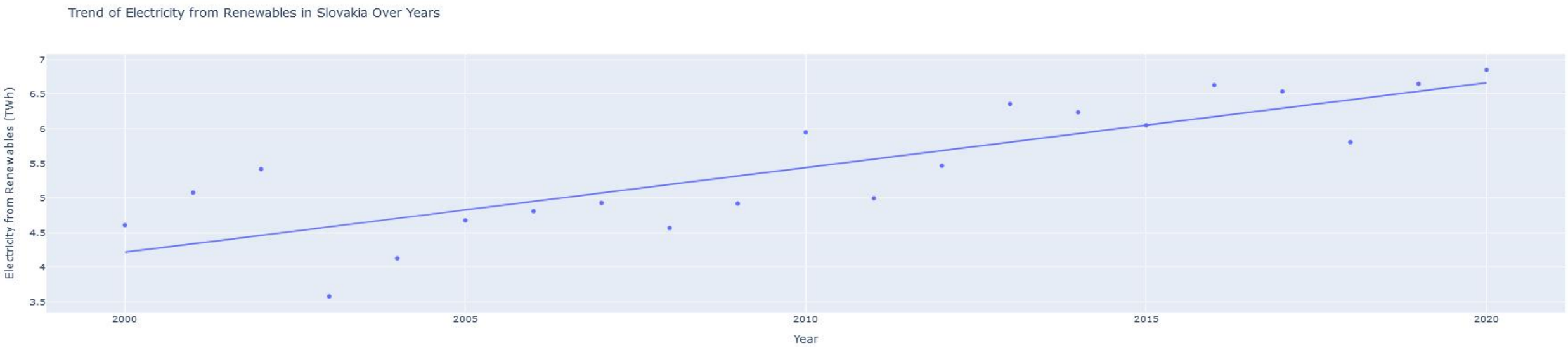
```python
#Dropdown menu combined with a sidebar from previous code
# Filter data for Slovakia
slovakia_data = df[df['Entity'] == 'Slovakia']

# Create the figure
fig = go.Figure()

# Add traces for each energy source
fig.add_trace(go.Scatter(
    x=slovakia_data['Year'],
    y=slovakia_data['Electricity from fossil fuels (TWh)'],
    name='Fossil Fuels',
    visible=True  # Initially visible
))

fig.add_trace(go.Scatter(
    x=slovakia_data['Year'],
    y=slovakia_data['Electricity from nuclear (TWh)'],
    name='Nuclear',
    visible=False  # Initially hidden
))

fig.add_trace(go.Scatter(
    x=slovakia_data['Year'],
    y=slovakia_data['Electricity from renewables (TWh)'],
    name='Renewables',
    visible=False  # Initially hidden
))

# Update layout with dropdown menu and slider
fig.update_layout(
    title='Energy Production in Slovakia',
    xaxis_title='Year',
    yaxis_title='Electricity Production (TWh)',
    xaxis=dict(
        rangeslider=dict(visible=True),
        type='linear'  # Ensure it's a linear scale for years
    ),
    updatemenus=[
        {
            'buttons': [
                {
                    'label': 'Fossil Fuels',
                    'method': 'update',
                    'args': [{'visible': [True, False, False]}, {'title': 'Fossil Fuels Production'}]
                },
                {
                    'label': 'Nuclear',
                    'method': 'update',
                    'args': [{'visible': [False, True, False]}, {'title': 'Nuclear Production'}]
                },
                {
                    'label': 'Renewables',
                    'method': 'update',
                    'args': [{'visible': [False, False, True]}, {'title': 'Renewable Production'}]
                },
                {
                    'label': 'All Sources',
                    'method': 'update',
                    'args': [{'visible': [True, True, True]}, {'title': 'Energy Production in Slovakia'}]
                }
            ],
            'direction': 'down',
            'showactive': True,
        }
    ]
)
```



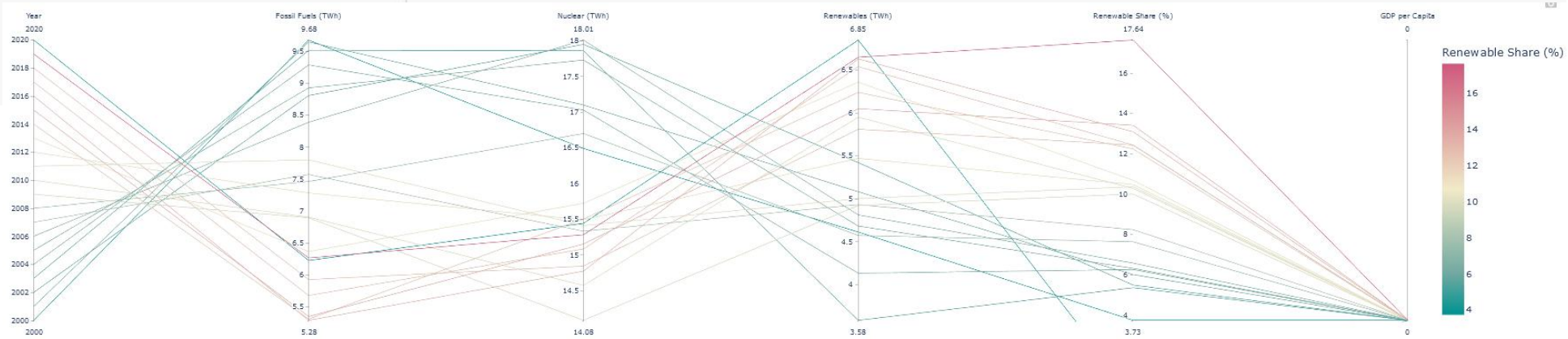
```python
# Create a scatter plot with a trend line
fig = px.scatter(
    df,
    x='Year',
    y='Electricity from renewables (TWh)',
    trendline='ols',  # Ordinary Least Squares regression for trend line
    title='Trend of Electricity from Renewables Over Years',
    labels={'Electricity from renewables (TWh)': 'Electricity from Renewables (TWh)', 'Year': 'Year'}
)

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



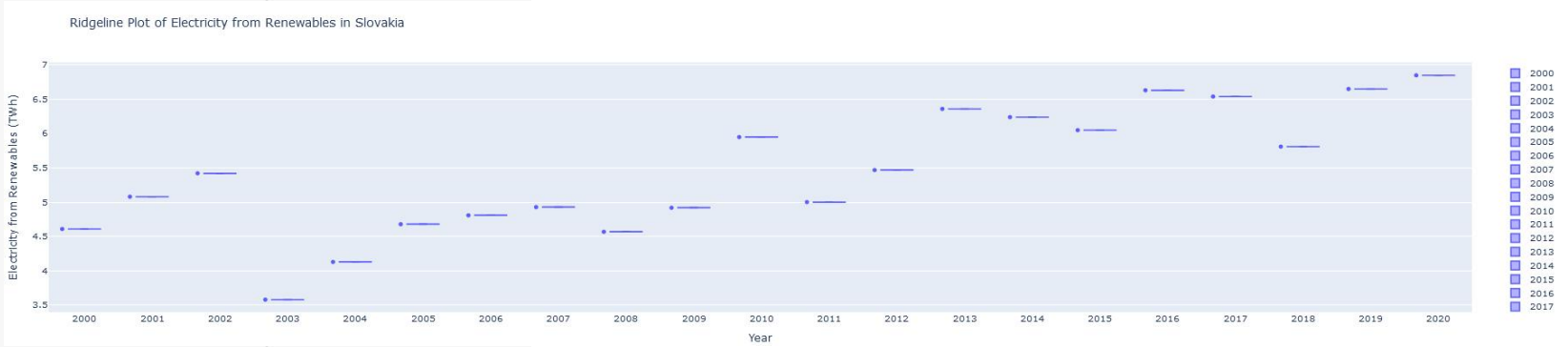
```python
#chart with a trendline

# Filter the DataFrame for Slovakia
slovakia_data = df[df['Entity'] == 'Slovakia']

# Create a scatter plot with trend lines for different energy types
fig = px.scatter(
    slovakia_data,
    x='Year',
    y='Electricity from renewables (TWh)',  # Change this to other columns as needed
    trendline='ols',  # Ordinary Least Squares regression for trend line
    title='Trend of Electricity from Renewables in Slovakia Over Years',
    labels={'Electricity from renewables (TWh)': 'Electricity from Renewables (TWh)', 'Year': 'Year'}
)

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



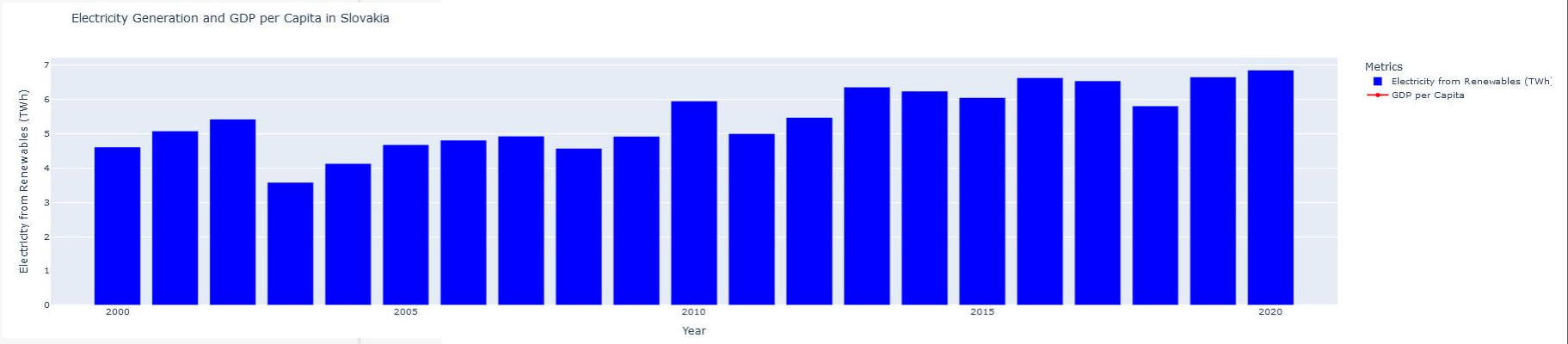
```python
# Parallel Coordinates Plot:
#The px.parallel_coordinates() function creates the plot.
#The color parameter is set to Renewable energy share in the total final energy consumption (%), allowing you to visualize how this variable changes across different years and energy types.
#The color_continuous_scale parameter sets the color gradient used for the lines.
# Filter the DataFrame for Slovakia
slovakia_data = df[df['Entity'] == 'Slovakia']

# Select relevant columns for the parallel coordinates plot
# Adjust the columns based on your interest and data availability
dimensions = [
    'Year',
    'Electricity from fossil fuels (TWh)',
    'Electricity from nuclear (TWh)',
    'Electricity from renewables (TWh)',
    'Renewable energy share in the total final energy consumption (%)',
    'gdp_per_capita'
]

# Create parallel coordinates plot
fig = px.parallel_coordinates(
    slovakia_data,
    dimensions=dimensions,
    color='Renewable energy share in the total final energy consumption (%)',  # Color by renewable share
    color_continuous_scale=px.colors.diverging.Tealrose,
    labels={
        'Electricity from fossil fuels (TWh)': 'Fossil Fuels (TWh)',
        'Electricity from nuclear (TWh)': 'Nuclear (TWh)',
        'Electricity from renewables (TWh)': 'Renewables (TWh)',
        'Renewable energy share in the total final energy consumption (%)': 'Renewable Share (%)',
        'gdp_per_capita': 'GDP per Capita'
    }
)

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



```python
#Working with Ridgeline charts
# Filter the DataFrame for Slovakia
slovakia_data = df[df['Entity'] == 'Slovakia']

# Select relevant columns for the ridgeline plot
# Here we will use 'Year' and 'Electricity from renewables (TWh)' as an example
years = slovakia_data['Year'].unique()
data_dict = {}

# Prepare data for ridgeline plot
for year in years:
    # Filter data for each year
    yearly_data = slovakia_data[slovakia_data['Year'] == year]['Electricity from renewables (TWh)']
    data_dict[year] = yearly_data

# Create a figure
fig = go.Figure()

# Add traces for each year's data
for year, values in data_dict.items():
    # Create a density trace for each year
    fig.add_trace(go.Violin(
        y=values,
        name=str(year),
        box_visible=True,
        line_color='blue',
        opacity=0.6,
        points='all',  # Show all points
        showlegend=True,
        scalegroup=str(year),  # Group by year to overlay them correctly
        spanmode='hard'
    ))

# Update layout to make it look like a ridgeline plot
fig.update_layout(
    title='Ridgeline Plot of Electricity from Renewables in Slovakia',
    yaxis_title='Electricity from Renewables (TWh)',
    xaxis_title='Year',
    yaxis=dict(zeroline=True),
    showlegend=True,
)

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



```python
#Plotly that combines multiple chart types
# Filter the DataFrame for Slovakia
slovakia_data = df[df['Entity'] == 'Slovakia']

# Create a bar chart for electricity generation from renewables
bar_chart = go.Bar(
    x=slovakia_data['Year'],
    y=slovakia_data['Electricity from renewables (TWh)'],
    name='Electricity from Renewables (TWh)',
    marker_color='blue'
)

# Create a line chart for GDP per capita
line_chart = go.Scatter(
    x=slovakia_data['Year'],
    y=slovakia_data['gdp_per_capita'],
    name='GDP per Capita',
    mode='lines+markers',
    marker=dict(color='red')
)

# Create a figure and add both charts
fig = go.Figure()

# Add bar and line charts to the figure
fig.add_trace(bar_chart)
fig.add_trace(line_chart)

# Update layout for better visualization
fig.update_layout(
    title='Electricity Generation and GDP per Capita in Slovakia',
    xaxis_title='Year',
    yaxis_title='Electricity (TWh) / GDP per Capita',
    legend_title='Metrics',
    yaxis=dict(title='Electricity from Renewables (TWh)', side='left', showgrid=True),
)

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



```python
import pandas as pd
import plotly.express as px

# Assuming df is your DataFrame and it contains relevant columns
# Filter for Slovakia if necessary
slovakia_data = df[df['Entity'] == 'Slovakia']

# Example: Create an animated scatter plot
fig = px.scatter(
    slovakia_data,
    x='Year',  # Replace with the appropriate x-axis variable
    y='Electricity from renewables (TWh)',  # Replace with the appropriate y-axis variable
    animation_frame='Year',  # Column for animation frames
    animation_group='Entity',  # Grouping variable, if applicable
    size='Electricity from nuclear (TWh)',  # Size of markers based on another variable, if desired
    color='Entity',  # Color by entity (if you have multiple entities)
    hover_name='Entity',  # Hover information
    title='Animated Energy Production in Slovakia Over Time',
    labels={'Electricity from renewables (TWh)': 'Renewable Energy (TWh)', 'Year': 'Year'}
)

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