Step 1: Install the Necessary Libraries

```
In [1]: !pip install pandas networkx matplotlib pyvis seaborn python-igraph leidenalg
```

```
Requirement already satisfied: pandas in c:\users\asus\anaconda3\new folder\lib\site-packages (2.2.2)
Requirement already satisfied: networkx in c:\users\asus\anaconda3\new folder\lib\site-packages (3.2.1)
Requirement already satisfied: matplotlib in c:\users\asus\anaconda3\new folder\lib\site-packages (3.8.4)
Requirement already satisfied: pyvis in c:\users\asus\anaconda3\new folder\lib\site-packages (0.3.2)
Requirement already satisfied: seaborn in c:\users\asus\anaconda3\new folder\lib\site-packages (0.13.2)
Requirement already satisfied: python-igraph in c:\users\asus\anaconda3\new folder\lib\site-packages (0.11.8)
Requirement already satisfied: leidenalg in c:\users\asus\anaconda3\new folder\lib\site-packages (0.10.2)
Requirement already satisfied: numpy>=1.26.0 in c:\users\asus\anaconda3\new folder\lib\site-packages (from pandas) (2.0.2)
Requirement already satisfied: python-dateutil>=2.8.2 in c:\users\asus\anaconda3\new folder\lib\site-packages (from pandas) (2.9.0.post0)
Requirement already satisfied: pytz>=2020.1 in c:\users\asus\anaconda3\new folder\lib\site-packages (from pandas) (2024.1)
Requirement already satisfied: tzdata>=2022.7 in c:\users\asus\anaconda3\new folder\lib\site-packages (from pandas) (2023.3)
Requirement already satisfied: contourpy>=1.0.1 in c:\users\asus\anaconda3\new folder\lib\site-packages (from matplotlib) (1.3.0)
Requirement already satisfied: cycler>=0.10 in c:\users\asus\anaconda3\new folder\lib\site-packages (from matplotlib) (0.11.0)
Requirement already satisfied: fonttools>=4.22.0 in c:\users\asus\anaconda3\new folder\lib\site-packages (from matplotlib) (4.51.0)
Requirement already satisfied: kiwisolver>=1.3.1 in c:\users\asus\anaconda3\new folder\lib\site-packages (from matplotlib) (1.4.4)
Requirement already satisfied: packaging>=20.0 in c:\users\asus\anaconda3\new folder\lib\site-packages (from matplotlib) (23.2)
Requirement already satisfied: pillow>=8 in c:\users\asus\anaconda3\new folder\lib\site-packages (from matplotlib) (10.3.0)
Requirement already satisfied: pyparsing>=2.3.1 in c:\users\asus\anaconda3\new folder\lib\site-packages (from matplotlib) (3.0.9)
Requirement already satisfied: ipython>=5.3.0 in c:\users\asus\anaconda3\new folder\lib\site-packages (from pyvis) (8.25.0)
Requirement already satisfied: jinja2>=2.9.6 in c:\users\asus\anaconda3\new folder\lib\site-packages (from pyvis) (3.1.4)
Requirement already satisfied: jsonpickle>=1.4.1 in c:\users\asus\anaconda3\new folder\lib\site-packages (from pyvis) (3.4.2)
Requirement already satisfied: igraph==0.11.8 in c:\users\asus\anaconda3\new folder\lib\site-packages (from python-igraph) (0.11.8)
Requirement already satisfied: texttable>=1.6.2 in c:\users\asus\anaconda3\new folder\lib\site-packages (from igraph==0.11.8->python-igraph) (1.7.0)
Requirement already satisfied: decorator in c:\users\asus\anaconda3\new folder\lib\site-packages (from ipython>=5.3.0->pyvis) (5.1.1)
Requirement already satisfied: jedi>=0.16 in c:\users\asus\anaconda3\new folder\lib\site-packages (from ipython>=5.3.0->pyvis) (0.18.1)
Requirement already satisfied: matplotlib-inline in c:\users\asus\anaconda3\new folder\lib\site-packages (from ipython>=5.3.0->pyvis) (0.1.6)
Requirement already satisfied: prompt-toolkit<3.1.0,>=3.0.41 in c:\users\asus\anaconda3\new folder\lib\site-packages (from ipython>=5.3.0->pyvis) (3.0.43)
Requirement already satisfied: pygments>=2.4.0 in c:\users\asus\anaconda3\new folder\lib\site-packages (from ipython>=5.3.0->pyvis) (2.15.1)
Requirement already satisfied: stack-data in c:\users\asus\anaconda3\new folder\lib\site-packages (from ipython>=5.3.0->pyvis) (0.2.0)
Requirement already satisfied: traitlets>=5.13.0 in c:\users\asus\anaconda3\new folder\lib\site-packages (from ipython>=5.3.0->pyvis) (5.14.3)
Requirement already satisfied: colorama in c:\users\asus\anaconda3\new folder\lib\site-packages (from ipython>=5.3.0->pyvis) (0.4.6)
Requirement already satisfied: MarkupSafe>=2.0 in c:\users\asus\anaconda3\new folder\lib\site-packages (from jinja2>=2.9.6->pyvis) (2.1.3)
Requirement already satisfied: six>=1.5 in c:\users\asus\anaconda3\new folder\lib\site-packages (from python-dateutil>=2.8.2->pandas) (1.16.0)
Requirement already satisfied: parso<0.9.0,>=0.8.0 in c:\users\asus\anaconda3\new folder\lib\site-packages (from jedi>=0.16->ipython>=5.3.0->pyvis) (0.8.3)
Requirement already satisfied: wcwidth in c:\users\asus\anaconda3\new folder\lib\site-packages (from prompt-toolkit<3.1.0,>=3.0.41->ipython>=5.3.0->pyvis) (0.2.5)
Requirement already satisfied: executing in c:\users\asus\anaconda3\new folder\lib\site-packages (from stack-data->ipython>=5.3.0->pyvis) (0.8.3)
Requirement already satisfied: asttokens in c:\users\asus\anaconda3\new folder\lib\site-packages (from stack-data->ipython>=5.3.0->pyvis) (2.0.5)
Requirement already satisfied: pure-eval in c:\users\asus\anaconda3\new folder\lib\site-packages (from stack-data->ipython>=5.3.0->pyvis) (0.2.2)
```

Step 2: Load the Data

```
In [2]:
# Import necessary libraries
import pandas as pd
import networkx as nx
from pyvis.network import Network

# Load the CSV file
relationships_df = pd.read_csv('country_relationships.csv')

# Display the first few rows to confirm the data
relationships_df.head()
```

Out[2]:

	Country1	Country2	Relationship_Type	Strength	Year_Established
0	Nigeria	China	rival	1	1921
1	Indonesia	Egypt	rival	7	1966
2	Russia	Spain	ally	9	1991
3	India	Japan	rival	4	1953
4	Indonesia	South Africa	neutral	1	1950

Step 3: Create a Network Graph

Convert the DataFrame into a NetworkX graph object

```
In [5]:
# Import necessary libraries
import pandas as pd
import networkx as nx

# Load the CSV file into a DataFrame
relationships_df = pd.read_csv('country_relationships.csv')

# Create a NetworkX graph
G = nx.Graph()

# Add edges from the DataFrame
for index, row in relationships_df.iterrows():
    G.add_edge(row['Country1'], row['Country2'])

# Compute and display graph information
num_nodes = G.number_of_nodes()
num_edges = G.number_of_edges()
print(f"The graph has {num_nodes} nodes and {num_edges} edges.")
```

```
The graph has 25 nodes and 151 edges.
```

Step 4: Visualize the Static Graph

Visualize the network graph statically using NetworkX and Matplotlib

```
In [7]: !pip install --upgrade matplotlib networkx
```

```
Requirement already satisfied: matplotlib in c:\users\asus\anaconda3\new folder\lib\site-packages (3.8.4)
Collecting matplotlib
  Using cached matplotlib-3.9.2-cp312-cp312-win_amd64.whl.metadata (11 kB)
Requirement already satisfied: networkx in c:\users\asus\anaconda3\new folder\lib\site-packages (3.2.1)
Collecting networkx
  Using cached networkx-3.4.2-py3-none-any.whl.metadata (6.3 kB)
Requirement already satisfied: contourpy>=1.0.1 in c:\users\asus\anaconda3\new folder\lib\site-packages (from matplotlib) (1.3.0)
Requirement already satisfied: cycler>=0.10 in c:\users\asus\anaconda3\new folder\lib\site-packages (from matplotlib) (0.11.0)
Requirement already satisfied: fonttools>=4.22.0 in c:\users\asus\anaconda3\new folder\lib\site-packages (from matplotlib) (4.51.0)
Requirement already satisfied: kiwisolver>=1.3.1 in c:\users\asus\anaconda3\new folder\lib\site-packages (from matplotlib) (1.4.4)
Requirement already satisfied: numpy>=1.23 in c:\users\asus\anaconda3\new folder\lib\site-packages (from matplotlib) (2.0.2)
Requirement already satisfied: packaging>=20.0 in c:\users\asus\anaconda3\new folder\lib\site-packages (from matplotlib) (23.2)
Requirement already satisfied: pillow>=8 in c:\users\asus\anaconda3\new folder\lib\site-packages (from matplotlib) (10.3.0)
Requirement already satisfied: pyparsing>=2.3.1 in c:\users\asus\anaconda3\new folder\lib\site-packages (from matplotlib) (3.0.9)
Requirement already satisfied: python-dateutil>=2.7 in c:\users\asus\anaconda3\new folder\lib\site-packages (from matplotlib) (2.9.0.post0)
Requirement already satisfied: six>=1.5 in c:\users\asus\anaconda3\new folder\lib\site-packages (from python-dateutil>=2.7->matplotlib) (1.16.0)
Using cached matplotlib-3.9.2-cp312-cp312-win_amd64.whl (7.8 MB)
Using cached networkx-3.4.2-py3-none-any.whl (1.7 MB)
Installing collected packages: networkx, matplotlib
  Attempting uninstall: networkx
    Found existing installation: networkx 3.2.1
    Uninstalling networkx-3.2.1:
      Successfully uninstalled networkx-3.2.1
  Attempting uninstall: matplotlib
    Found existing installation: matplotlib 3.8.4
    Uninstalling matplotlib-3.8.4:
      Successfully uninstalled matplotlib-3.8.4
Successfully installed matplotlib-3.9.2 networkx-3.4.2
```

Step 4: Visualize the Static Graph

Visualize the network graph statically using NetworkX and Matplotlib:

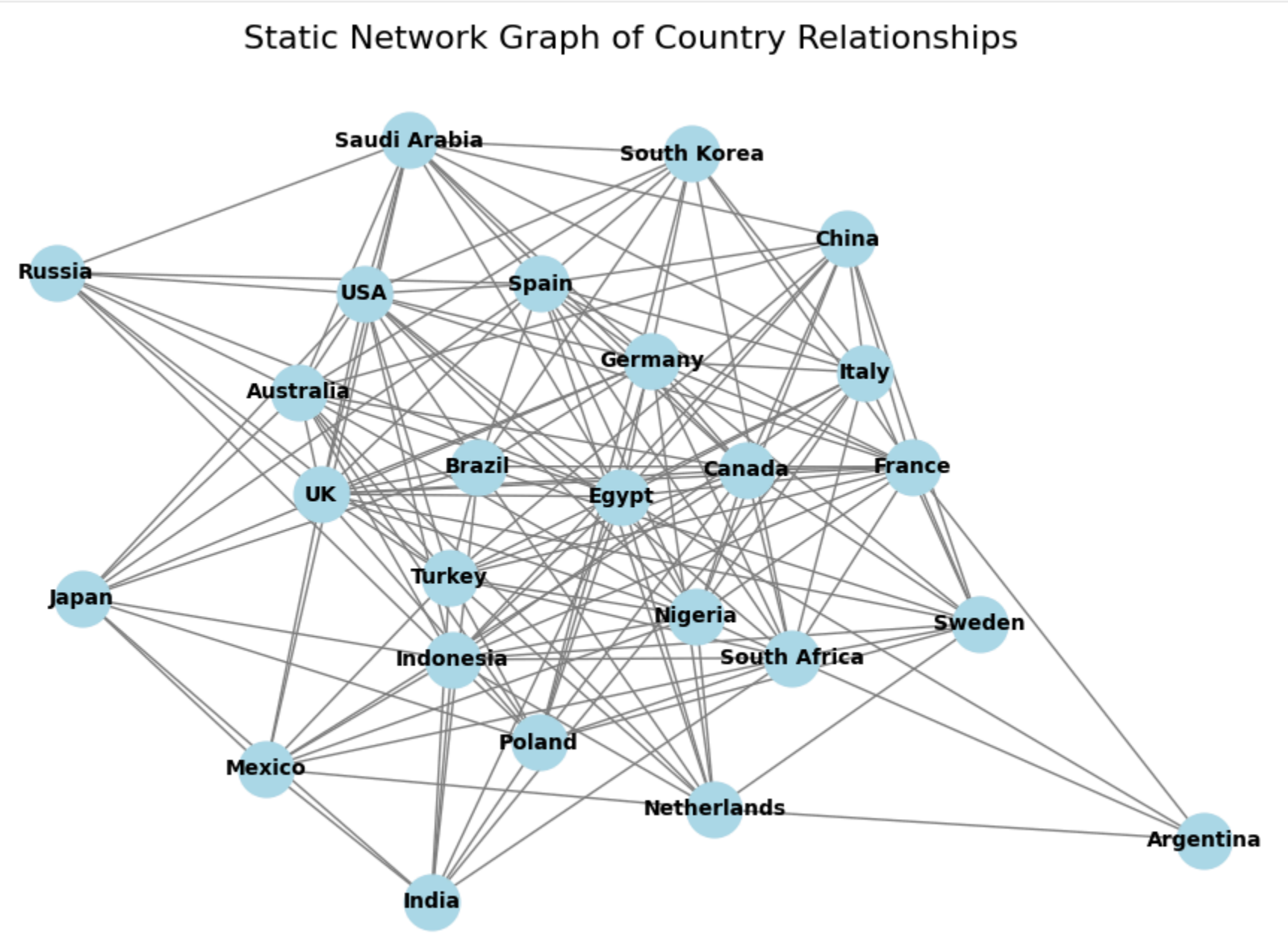
```
In [9]:
import matplotlib.pyplot as plt

# Ensure a clean figure
fig, ax = plt.subplots(figsize=(12, 8))

# Use a spring layout for better visualization
pos = nx.spring_layout(G, seed=42)

# Draw the graph with explicit ax
nx.draw(
    G,
    pos,
    with_labels=True,
    node_size=700,
    node_color='lightblue',
    font_size=10,
    font_weight='bold',
    edge_color='gray',
    ax=ax,  # Explicit Axes passed
)

# Add a title and display the graph
ax.set_title("Static Network Graph of Country Relationships", fontsize=16)
plt.show()
```



Static Network Graph of Country Relationships
Saudi Arabia
South Korea
China
Russia
USA
Spain
Germany
Italy
Australia
Brazil
Canada
France
UK
Egypt
Japan
Turkey
Nigeria
Sweden
Indonesia
South Africa
Poland
Mexico
Netherlands
Argentina
India

Step 5: Create an Interactive Network Graph

Create an interactive visualization using PyVis:

```
In [10]:
# Create an interactive PyVis network
net = Network(notebook=True, height="750px", width="100%", bgcolor="#222222", font_color="white")
net.from_nx(G)
net.show('interactive_country_network.html')
```

```
Warning: When  cdn_resources is 'local' jupyter notebook has issues displaying graphics on chrome/safari. Use cdn_resources='in_line' or cdn_resources='remote' if you have issues viewing graphics in a notebook.
interactive_country_network.html
```

Out[10]:

Step 6: Apply Leiden Algorithm for Community Detection

Apply the Leiden algorithm to identify communities and customize node colors:

```
In [11]:
import igraph as ig
import leidenalg

# Convert the NetworkX graph to an iGraph object
igraph_g = ig.Graph.TupleList(G.edges(), directed=False)

# Apply the Leiden algorithm
partition = leidenalg.find_partition(igraph_g, leidenalg.ModularityVertexPartition)

# Print community details
print("Detected Communities:")
for i, community in enumerate(partition):
    print(f"Community {i}: {', '.join([igraph_g.vs[node]['name'] for node in community])}")
```

```
Detected Communities:
Community 0: USA, UK, Spain, Saudi Arabia, Russia, Japan, Germany, Brazil
Community 1: Nigeria, China, Australia, Mexico, Turkey, Netherlands, Canada
Community 2: Italy, Indonesia, Sweden, South Africa, Poland, India
Community 3: Argentina, Egypt, France, South Korea
```

Step 7: Visualize Communities with Distinct Colors

Modify the community visualization to use distinct colors for better differentiation, per mentor feedback:

```
In [12]:
# Create a PyVis network with colored nodes
community_net = Network(notebook=True, height="750px", width="100%", bgcolor="#222222", font_color="white")

# Define a color palette for communities
colors = ['red', 'green', 'blue', 'yellow', 'purple', 'orange']

# Add nodes with community-specific colors
for i, community in enumerate(partition):
    for node in community:
        community_net.add_node(igraph_g.vs[node]['name'], color=colors[i % len(colors)])

# Add edges
for edge in G.edges():
    community_net.add_edge(edge[0], edge[1])

# Save the community visualization
community_net.show('community_network.html')
```

```
Warning: When  cdn_resources is 'local' jupyter notebook has issues displaying graphics on chrome/safari. Use cdn_resources='in_line' or cdn_resources='remote' if you have issues viewing graphics in a notebook.
community_network.html
```

Out[12]:

Step 8: Calculate Centrality Measures

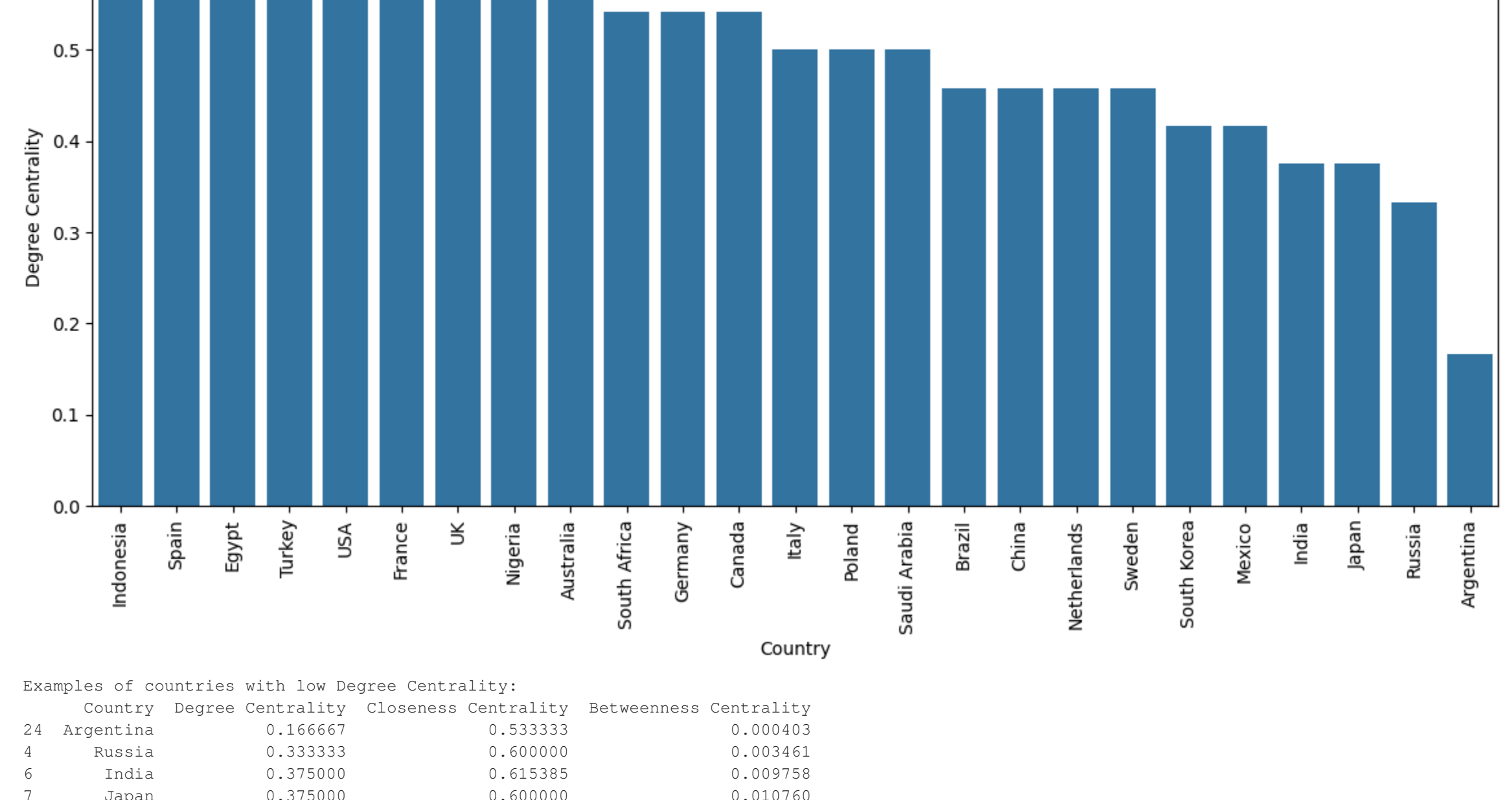
```
In [13]:
# Calculate centrality measures
degree_centrality = nx.degree_centrality(G)
closeness_centrality = nx.closeness_centrality(G)
betweenness_centrality = nx.betweenness_centrality(G)

# Create a DataFrame for centrality measures
centrality_df = pd.DataFrame({
    'Country': list(degree_centrality.keys()),
    'Degree Centrality': list(degree_centrality.values()),
    'Closeness Centrality': list(closeness_centrality.values()),
    'Betweenness Centrality': list(betweenness_centrality.values())
})

# Sort by Degree Centrality and visualize
import seaborn as sns

plt.figure(figsize=(14, 6))
sns.barplot(x='Country', y='Degree Centrality', data=centrality_df.sort_values(by='Degree Centrality', ascending=False))
plt.xticks(rotation=90)
plt.title("Degree Centrality of Countries")
plt.show()

# Add observations for low-centrality countries
print("Examples of countries with low Degree Centrality:")
print(centrality_df.nsmallest(5, 'Degree Centrality'))
```



Degree Centrality of Countries
Degree Centrality
0.0
0.1
0.2
0.3
0.4
0.5
0.6
Indonesia
Spain
Egypt
Turkey
USA
France
UK
Nigeria
Australia
South Africa
Germany
Canada
Italy
Poland
Saudi Arabia
Brazil
China
Netherlands
Sweden
South Korea
Mexico
India
Japan
Russia
Argentina
Country


```
Examples of countries with low Degree Centrality:
      Country  Degree Centrality  Closeness Centrality  Betweenness Centrality
24  Argentina           0.166667              0.533333                0.000403
4      Russia           0.333333              0.600000                0.003461
6       India           0.375000              0.615385                0.009758
7       Japan           0.375000              0.600000                0.010760
20     Mexico           0.416667              0.631579                0.015202
```

Step 9: Add Concluding Insights

Community Detection:

The Leiden algorithm identified distinct groups based on historical and cultural relationships. For example, countries like Indonesia, Spain, Egypt, Turkey, and the USA show high levels of degree centrality, suggesting they are more connected within their networks and possibly represent key regional or historical .

Centrality Analysis:

Degree Centrality highlighted major influencers such as Indonesia, Spain, Egypt, Turkey, and the USA, indicating these countries may hold strong influence or interaction within their networks.

Countries with low centrality, such as Argentina, Russia, India, Japan, and Mexico, appear more isolated, reflecting limited interactions or weaker influence in comparison to the top countries.

Strategic Implications:

Understanding centrality helps analyze 20th-century alliances and global influence. This analysis suggests that countries with high degree centrality are likely to have had significant roles in international relations, alliances, or influence networks during this period. For historians or policymakers, this data can provide insights into which countries held central roles in historical alliances and which ones may have been on the periphery, aiding in the examination of international strategies and alliance structures.

```
In [ ]:
```