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 (3.4.2)
Requirement already satisfied: matplotlib in c:\users\asus\anaconda3\new folder\lib\site-packages (3.9.2)
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

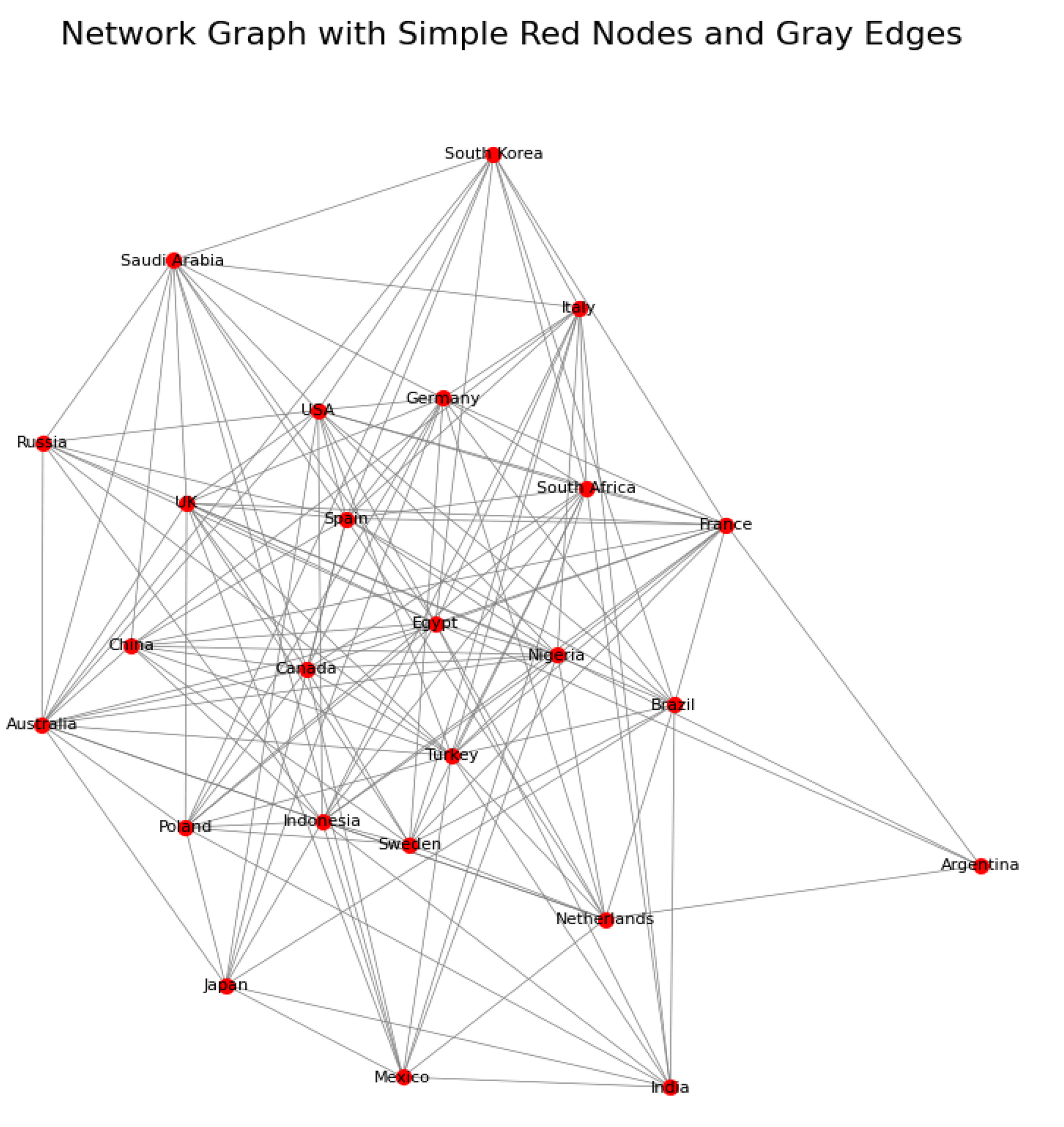
In [3]:
```python
# Import necessary libraries
import pandas as pd
import networkx as nx
from pyvis.network import Network
# Load the CSV file
relationships_df = pd.read_csv('country_relationships.csv')
# Display the first few rows to confirm the data
relationships_df.head()
```

Out[3]:

	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

In [10]:
```python
import networkx as nx
import matplotlib.pyplot as plt

# Sample data loading (assuming 'relationships_df' is your DataFrame)
relationships_df = pd.read_csv('country_relationships.csv')

# Create the graph object and add edges
G = nx.Graph()
for index, row in relationships_df.iterrows():
    G.add_edge(row['Country1'], row['Country2'], weight=row['Strength'])

# Improved static graph visualization using spring layout
fig, ax = plt.subplots(figsize=(10, 10))
pos = nx.spring_layout(G, seed=42)

# Draw nodes as red dots and edges as thin gray lines
nx.draw_networkx_nodes(G, pos, node_color='red', node_size=50, ax=ax)
nx.draw_networkx_edges(G, pos, edge_color='gray', width=0.5, ax=ax)

# Hide the labels for a cleaner look
nx.draw_networkx_labels(G, pos, font_size=8, font_color='black', ax=ax)

ax.set_title("Network Graph with Simple Red Nodes and Gray Edges", fontsize=16)
plt.axis('off')  # Turn off the axis for a cleaner look
plt.show()
```



Step 4: Visualize the Static Graph

Ensure that our visualization is clear, as per the mentor's suggestion for better use of colors. we can experiment with different color schemes or node sizes based on the centrality values.

Step 5: Create an Interactive Network Graph

In [11]:
```python
import pandas as pd  # Import pandas to load the CSV file
import networkx as nx
from pyvis.network import Network

# Sample data loading (assuming 'relationships_df' is your DataFrame)
relationships_df = pd.read_csv('country_relationships.csv')

# Create the graph object and add edges
G = nx.Graph()
for index, row in relationships_df.iterrows():
    G.add_edge(row['Country1'], row['Country2'], weight=row['Strength'])

# Interactive PyVis network
net = Network(notebook=True, height="750px", width="100%", bgcolor="#222222", font_color="white")

# Convert the NetworkX graph to a PyVis network
net.from_nx(G)

# Save the interactive graph to an HTML file
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[11]:

Step 6: Apply Leiden Algorithm for Community Detection

In [13]:
```python
import igraph as ig
import leidenalg

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

# Apply Leiden algorithm for community detection
partition = leidenalg.find_partition(igraph_g, leidenalg.ModularityVertexPartition)

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

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

Step 7: Visualize Communities with Distinct Colors

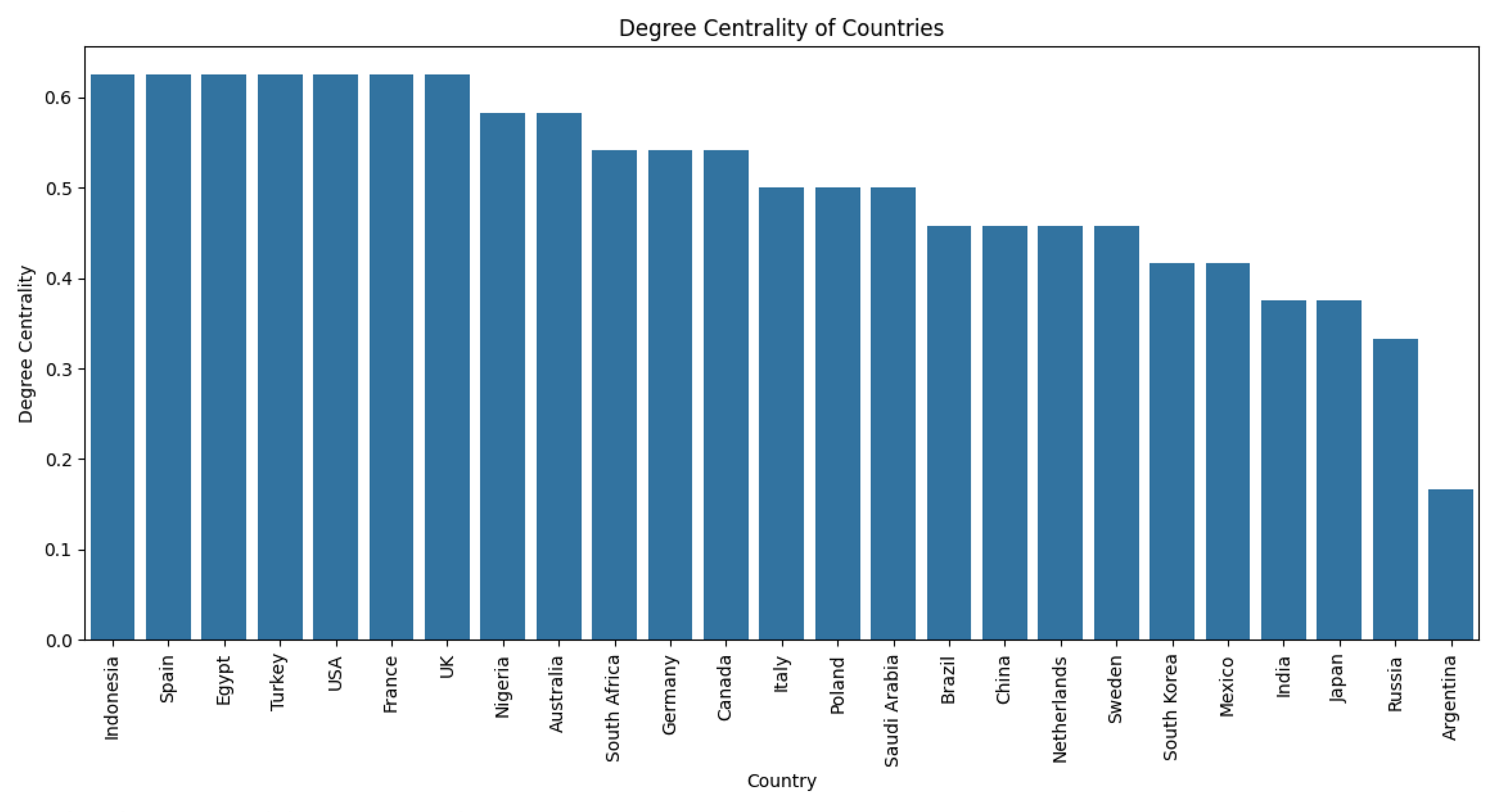
In [15]:
```python
# Create a PyVis network
community_net = Network(notebook=True, height="750px", width="100%", bgcolor="#222222", font_color="white")
colors = ['red', 'green', 'blue', 'yellow', 'purple', 'orange']

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

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[15]:

Step 8: Calculate Centrality Measures

In [8]:
```python
import seaborn as sns

# Visualize Degree Centrality
degree_centrality = nx.degree_centrality(G)
closeness_centrality = nx.closeness_centrality(G)
betweenness_centrality = nx.betweenness_centrality(G)

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())
})

# Degree centrality bar plot
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()

# Example of countries with low Degree Centrality
print("Examples of countries with low Degree Centrality:")
print(centrality_df.nsmallest(5, 'Degree Centrality'))
```




```
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

Based on your analysis, conclude by summarizing the strategic importance of centrality. For example:

Community Detection Insights: The Leiden algorithm identifies groups based on historical and cultural relationships. Countries like Indonesia, USA, Spain, and others show higher centrality, indicating strong interconnectedness.

Centrality Insights: Countries with high centrality (e.g., USA, Russia) may have significant influence in global affairs, while low-centrality countries (e.g., Argentina, Russia) may be more isolated.

Strategic Implications: This analysis can help policymakers understand the geopolitical importance of certain countries in historical and modern contexts.