**University of Central Florida** **Fall 2022**

**College of Engineering & Computer Science** **Course Project**

COP 5537: Network Optimization

# Analyzing World Trade Routes based on Centrality Measures & Community Detection Algorithm

Name: Sayali Sanjeevkumar Bhosale

ID: 5494025

# Abstract

Investigating the world trade routes through the use of networks and graph theory concepts. Main focus of this project is to implement Centrality analysis on the trade data of 227 countries and to find the groups of densely connected components by implementing community detection algorithm (Girvan-Newman algorithm) with and without removing top 10 GDP countries. Removal of top nodes in trade dataset changes the behaviour of graph so by creating communities we note the significant changes.

# Introduction

Every single item that we own, use, wear, eat has reached our door step and crossed several miles with the aid of trade routes. The entirety of the human race has been heavily influenced and improved by trade since the beginning of time; this was made possible by trade routes that connected different groups and spread over the expanse of land and sea. The trade that happens every day play a vital role in the rise or fall of the economies. Developing better trade routes and improving the transportation and access to foods can greatly improve the development of an economy.

In this project, we analyzed the trade that happens between 227 countries and the influence it has on these countries. For this implementation we used 'Graph Theory' Concepts which includes "Centrality analysis" to get the importance of each country (taken as node) in the whole network. It can be computed in various different terms as it tells importance of the country (node) according to different perspectives. We also implemented "Page Rank Algorithm" through which we can see the ranking of coutries which are likely to receive more links from other links. Also, applied the Community Detection Algorithm to the given data.

# Information about dataset

The dataset has been collected from the website: World Integrated Trade Solution (WITS) | Data on Export, Import, Tariff, NTM (worldbank.org). It is a software which gives access to the all international merchandise trade, tariff, and non-tariff measures (NTM) data. It shows country wise profiles in terms of exports, imports and statistics related to tariff with relevance of developments.

I've downloaded the zip file from '*Trade Stats – Data Download – Trade Stats – Country Summary'* which includes import and export details of 227 countries around the globe from the year 1988 to 2019. Product details of costings in import/export of the trade is provided in USD. Total size of the whole dataset is 24848 rows and 37 columns after compiling all the countries.

```
In [47]: all_countries_mapping_1
```

Out[47]:

| | Reporter | Partner | Product categories | Indicator Type | Indicator | 2019 | 2018 | 2017 | 2016 | 2015 | ... | 1995 | 1994 | 1993 | 1992 | 1991 | 1990 | 1989 | 198 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | Aruba | China | All Products | Import | Trade (US$ Mil)-Top 5 Import Partner | NaN | 28.66 | NaN | NaN | NaN | ... | NaN | NaN | NaN | NaN | NaN | NaN | NaN | Na |
| 1 | Aruba | China | All Products | Import | Partner share(%)-Top 5 Import Partner | NaN | 2.28 | NaN | NaN | NaN | ... | NaN | NaN | NaN | NaN | NaN | NaN | NaN | Na |
| 2 | Aruba | Colombia | All Products | Export | Trade (US$ Mil)-Top 5 Export Partner | NaN | 14.63 | 17.31 | 22.71 | 21.89 | ... | NaN | NaN | NaN | NaN | NaN | NaN | NaN | Na |
| | | | | | Trade | | | | | | | | | | | | | | |

# Problem Statement

There are several major players in world of trade, these countries tie up the entire world trade into one huge community. If we try splitting these into communities, it will not provide valid communities because it is strongly connected graph. But, if we remove top layers, it will form separate communities. The problem with this is, if we have communities, the trade between these countries might be affected because of weak links. The problem would be the reduction of goods and services to certain countries. To avoid this, we need our top GDP Countries as connectors. So we are creating communities to see the behaviour.

# Methods used in the implementations

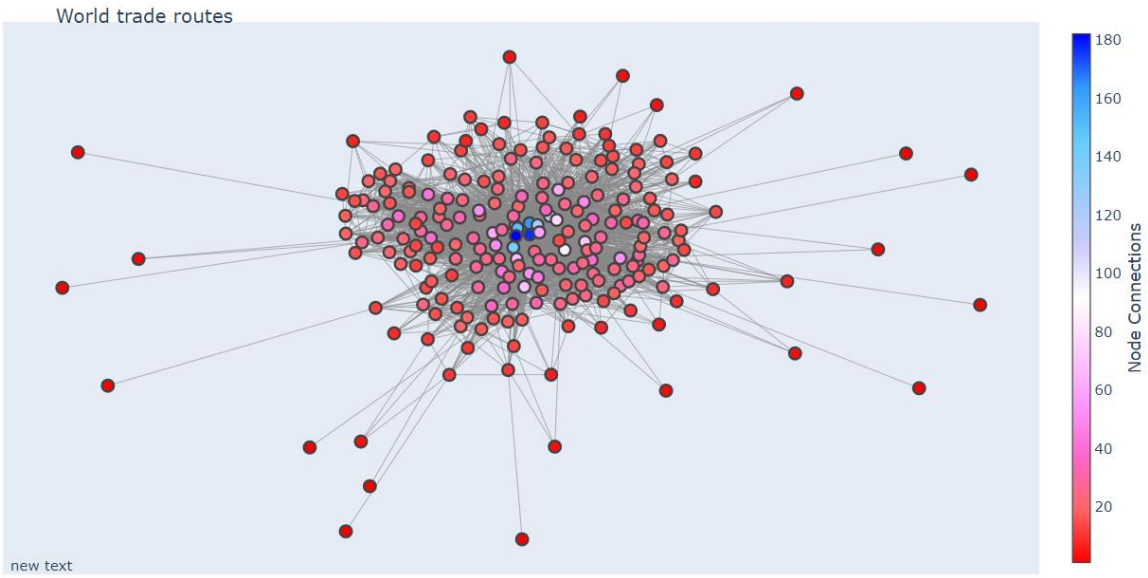
Creating Graph from the dataset of World Trade Route:

### Creating nodes and edges

```
In [8]: graph_nodes = []
        for i in range(len(all_countries_mapping_1)):
            val = (all_countries_mapping_1.Number[i],all_countries_mapping_1.ID[i])
            graph_nodes.append(val)
```

```
In [9]: G = nx.Graph() #Creating a graph
        G.add_edges_from(graph_nodes)
        pos_G = nx.spring_layout(G) #Giving the graph a layout
```

Visualization the entire trade route:




For entire world trade route:

```
In [11]: deg = []
         for i in range(len(list(G.degree()))):
             deg.append(list(G.degree())[i][1])
         avg_deg = sum(deg)/len(G.nodes)
         print("Average degree of this network is: ",avg_deg)

         Average degree of this network is:  25.559471365638768
```

Average degree of the network is 25.559% and The global clustering coefficient of graph is 0.282%.

**After removing the Top 10 Key-Players in trade:**

Removing Top 10 countries with highest GDP to see it's empact in the network

```
In [19]: rem_nodes = nx.Graph()
         rem_nodes.add_edges_from(graph_nodes)

In [20]: nodes_to_remove = [218,46,84,217,109,107,77,149,201,101]
         for i in nodes_to_remove:
             rem_nodes.remove_node(i)

In [21]: pos_G = nx.spring_layout(rem_nodes)
```

In [23]:
```
deg = []
for i in range(len(list(rem_nodes.degree()))):
    deg.append(list(rem_nodes.degree())[i][1])
avg_deg = sum(deg)/len(rem_nodes.nodes)
print("Average degree of this network is: ",avg_deg)
```

```
Average degree of this network is:  14.949308755760368
```

Average degree of the network is reduced to 14.949% and The global clustering coefficient of graph is 0.246%.

### To get the bridges (Cut-Edges)

In [12]:
```
bridges_list = list(nx.bridges(G))
bridge_nodes = []
for i in range(len(bridges_list)):
    bridge_nodes.append(bridges_list[i][0])
    bridge_nodes.append(bridges_list[i][1])
```

In [13]:
```
#for i in bridge_nodes:
bridge_countries = [countries_mapping[countries_mapping.ID == node].Partner.iloc[0] for node in bridge_nodes]
print(bridge_countries)
```

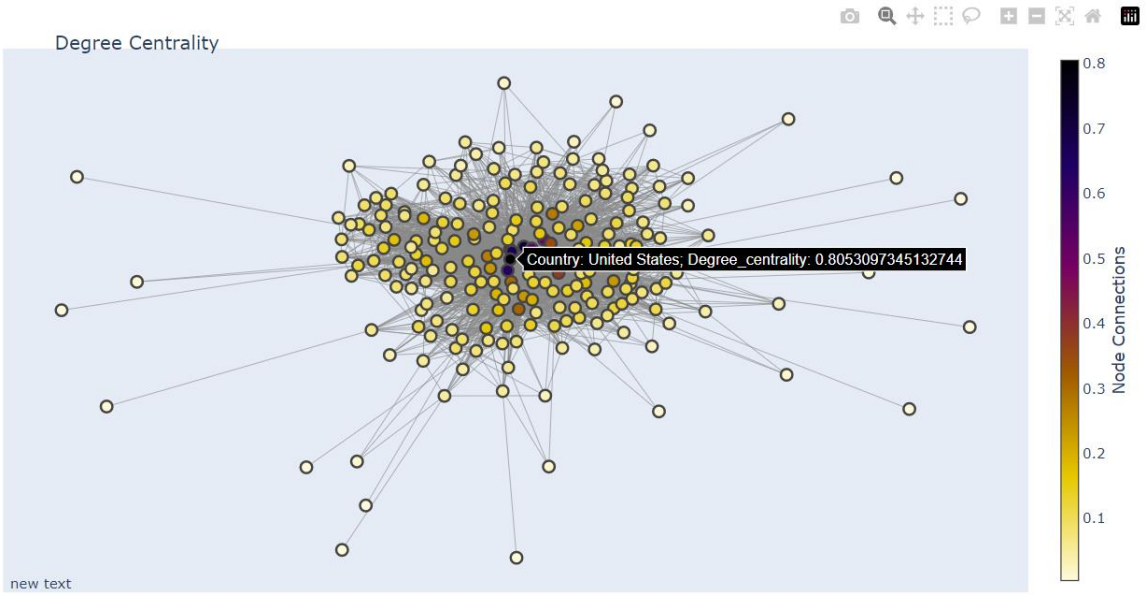
```
['Antigua and Barbuda', 'Saint Maarten (Dutch part)', 'Burundi', 'Saint Helena', 'Rwanda', 'Bouvet Island', 'Uganda', 'South Su
dan', 'Bangladesh', 'Antarctica', 'Turks and Caicos Isl.', 'San Marino', 'Congo, Rep.', 'Heard Island and McDonald Isla', 'Papu
a New Guinea', 'Pitcairn', 'Micronesia, Fed. Sts.', 'Northern Mariana Islands', 'Solomon Islands', 'Korea, Dem. Rep.', 'Samoa',
'Tokelau']
```

## **Centralities**:

For identification of important node in the graph, we use the centralities in which we compute how central particular node is in the whole graph. Centrality can be calculated in different ways as it tells importance of a node according to different perspectives. Here we have considered 4 types of centralities:

1) Degree centrality: Importance of a particular node in the network is being measured by its degree i.e. higher the degree of a node, the importance will be set as more in a graph.

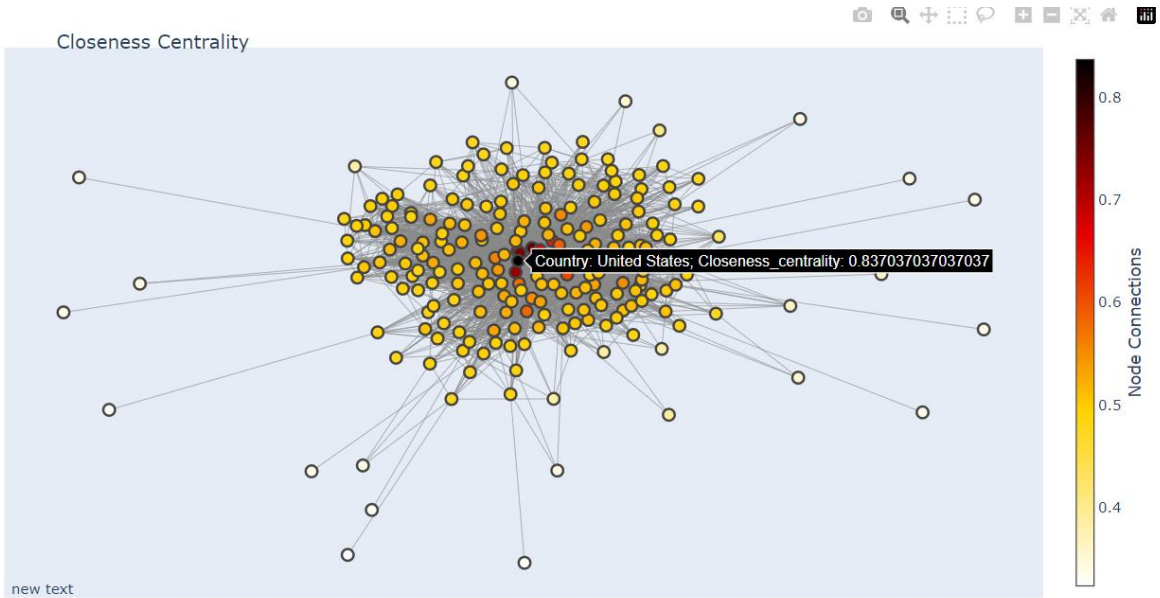
$\Sigma_j m(i,j)\ where\ m(i,j) = 1\ if\ there\ is\ linknode\ from\ node$ "i" $to\ node$ "$j$"




2) Closeness centrality: more the central node is, it will be closer to all other nodes. Calculated as the sum of the length of the shortest paths between the node and all other nodes in the graph.

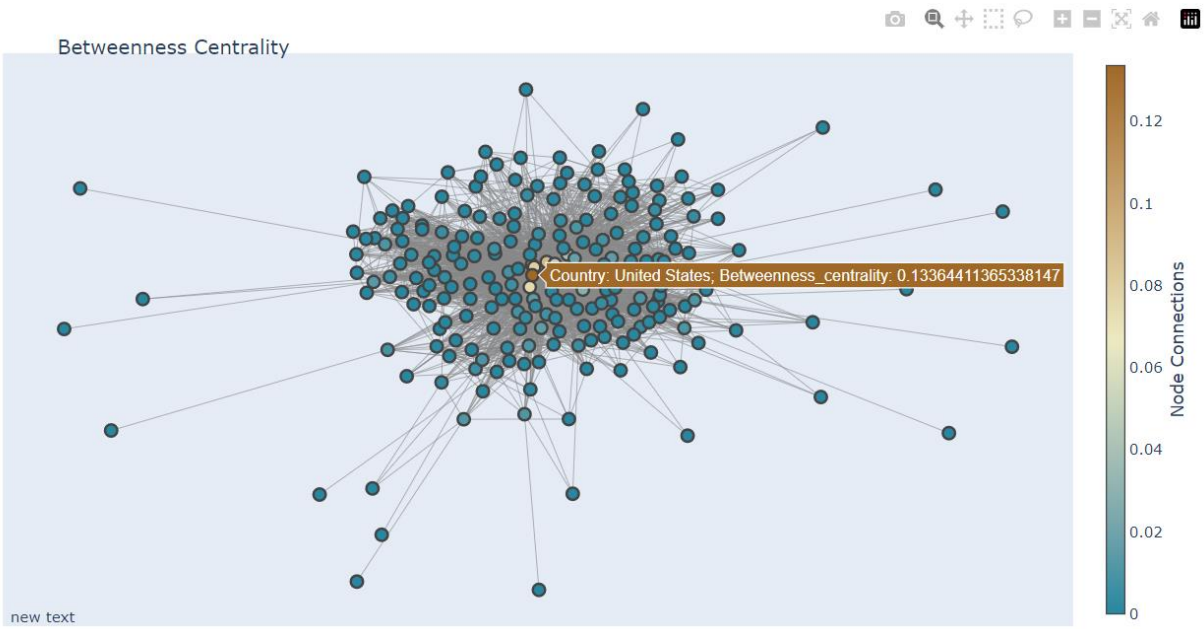
$$C(x) = \frac{1}{\sum d(x,y)}$$

where d(x, y) is the distance between vertices x and y.

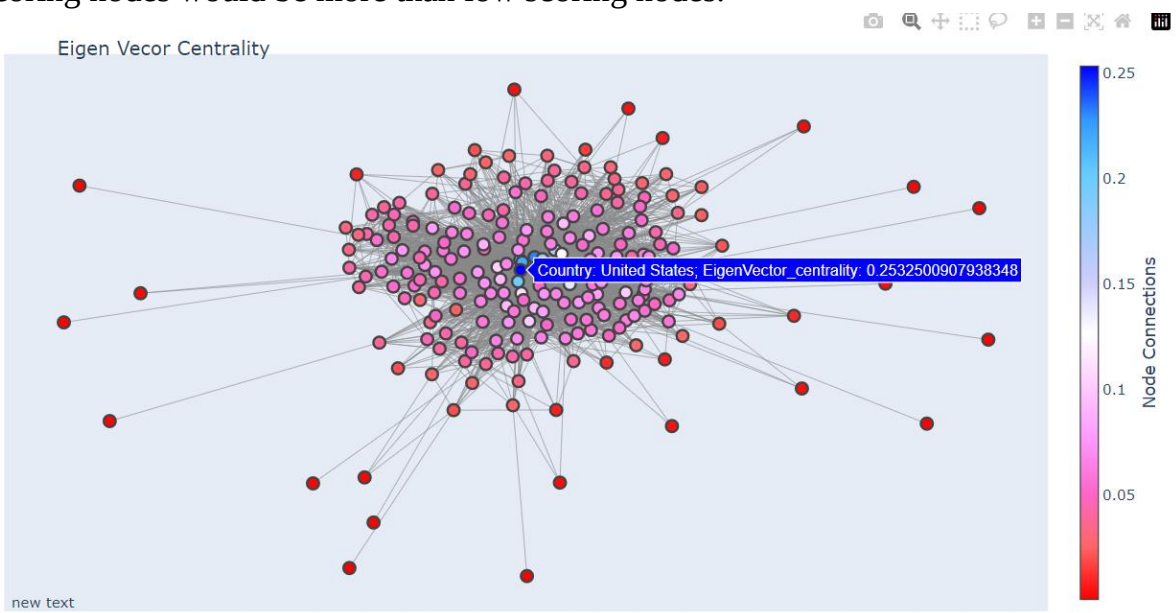


3) Betweenness centrality: Importance of a particular node in a network is being measured based upon how many times it is occurring in the shortest path between all pairs of nodes in a networks.

$$B(i,j) = \sum_{a,b} \frac{g_{aib}}{g_{ab}}$$




4) Eigen-vector Centrality: It is the measure of influence of a node in a network. Assigning relative scores to every node based upon contribution of high scoring nodes would be more than low scoring nodes.

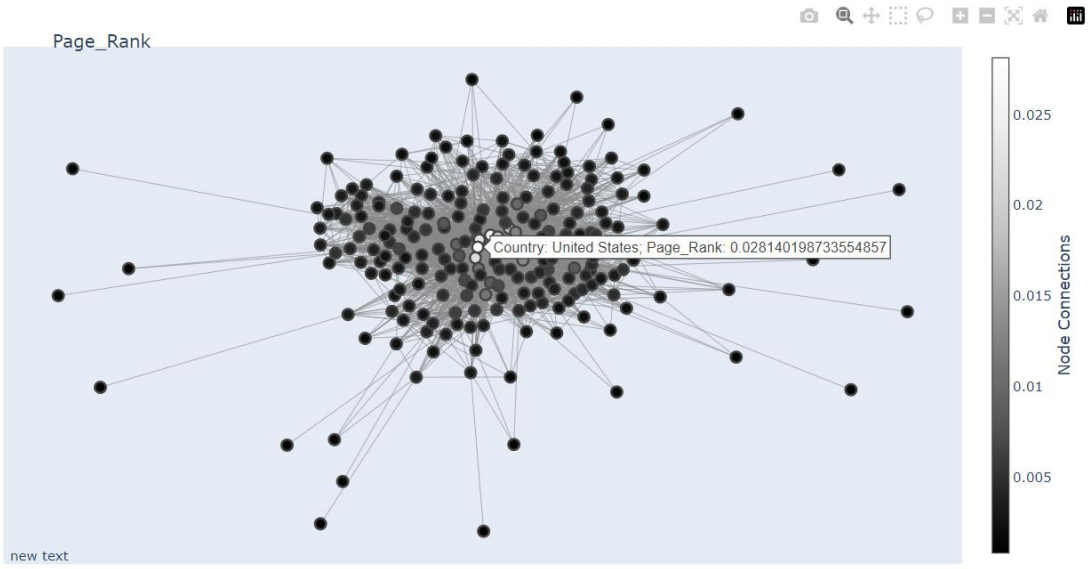


**Page Rank Algorithm:**

It is an algorithm used by Google search to rank the websites. It works by counting the number and quality of the links to node for estimation of how important the node is. If node is receiving more links from other nodes, then it will get higher rankings.

$$\sum_{v \in B} \frac{PR(v)}{L(v)}$$

PageRank value for a page u is dependent on the PageRank values for each page v contained in the set Bu i.e. the set containing all pages linking to page u, L(v) is links from page v.




**Entire trade network will look like this removal of top 10 GDP countries:**

**Community Detection Algorithm:**

Divided the nodes of the graph into several communities using the *Girwan Newman algorithm*. This algorithm removes the edges which has highest betweenness centrality until no edges are remained.

Algorithm:

1. Create a Graph G, with n nodes and its edges.
2. Calculate betweenness of all the edges.
3. Remove all the edges with highest betweenness.
4. Re-calculate the betweenness of all the edges which got affected by the removal.
5. Now, repeat the steps 3 & 4 until no edges remained.

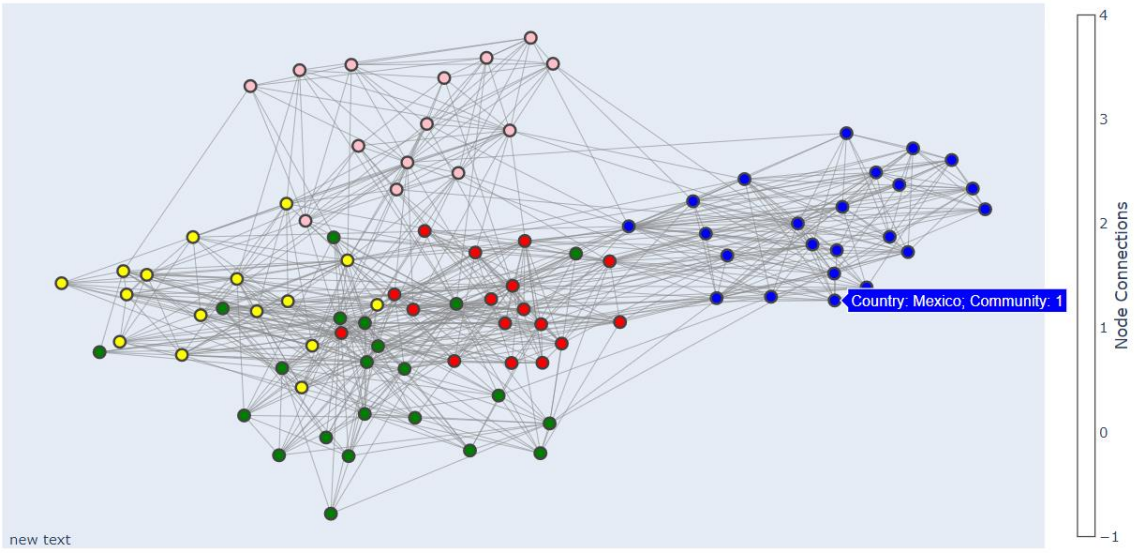
Applying Girvan-Newman Algorithm after removal of top 10 GDP Countries

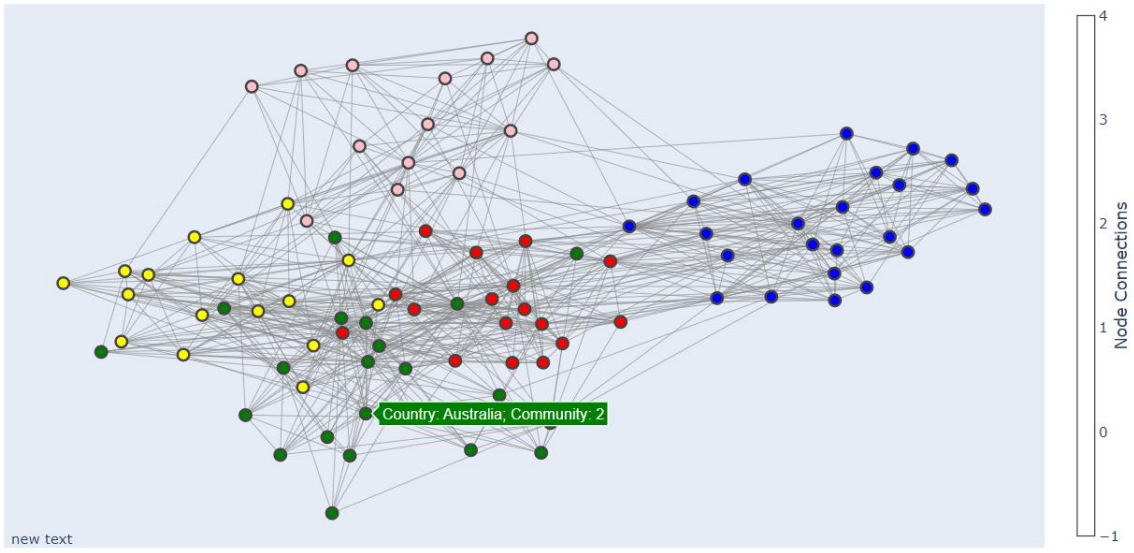
In [24]:
```
l = []
k = 100 #Creating 100 communities
comp = girvan_newman(rem_nodes)
#tuple(sorted(c) for c in next(comp))
for communities in itertools.islice(comp, k):
    l.append(tuple(sorted(c) for c in communities))
#print(l)
comm = l[99]
comm_size = {}
k = 0
for i in comm:
    comm_size[k] = len(i)
    k+=1
sortedDict = sorted(comm_size.items(), key=lambda x:x[1],reverse = True)
commm = []
for i in range(5):
    commm.append(sortedDict[i][0])
#print(commm)
len(comm)
```

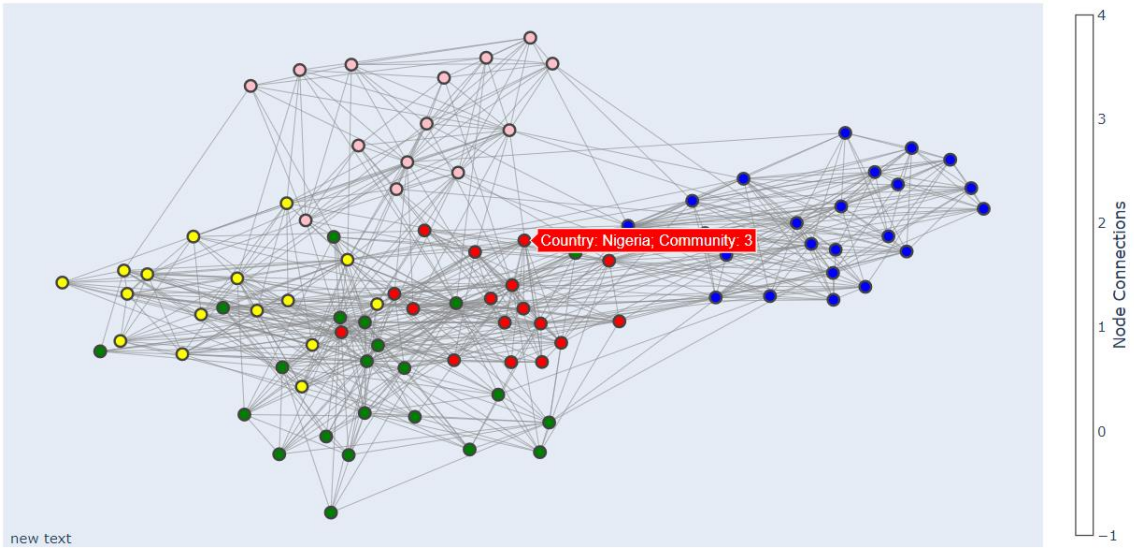
Out[24]: 101

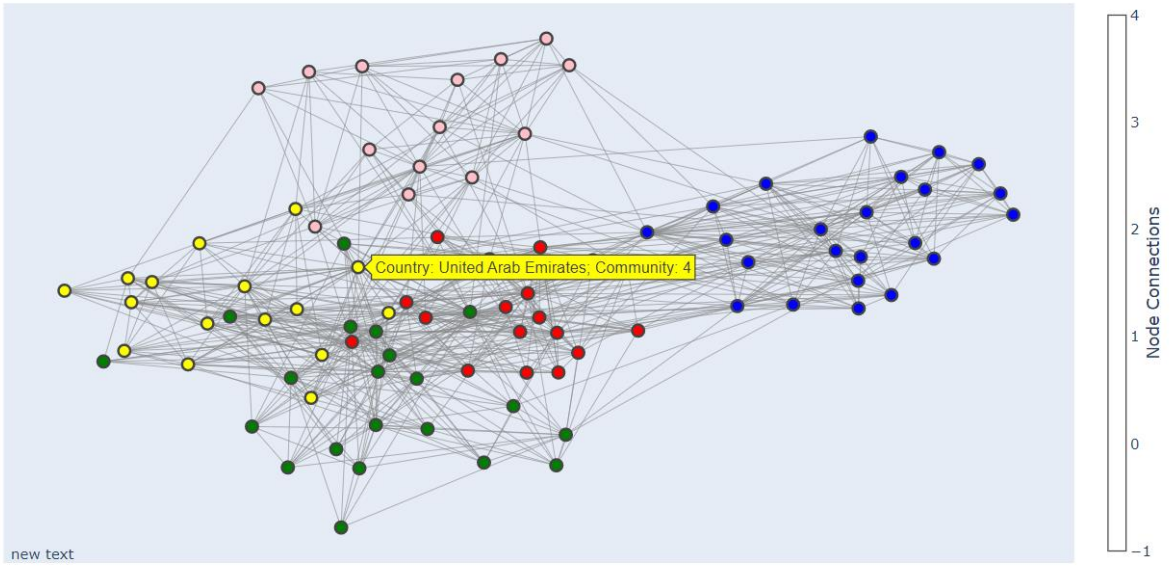
In [25]:
```
communities_dict = {}
community_number = 1
for i in commm:
    communities_dict[community_number] = comm[i]
    community_number+=1
```

In [26]:
```
community_map = {}
for com_, nodelist in communities_dict.items():
    for node in nodelist:
        community_map[node] = com_
```

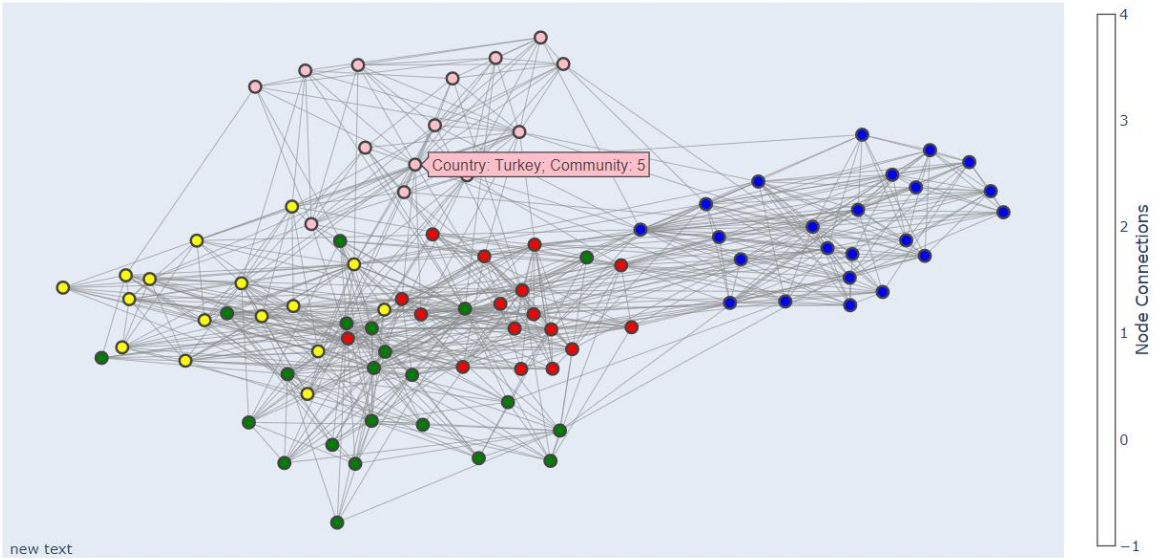

Country: Mexico; Community: 1
new text
Node Connections
4
3
2
1
0
−1


Country: Australia; Community: 2
new text
Node Connections
4
3
2
1
0
−1

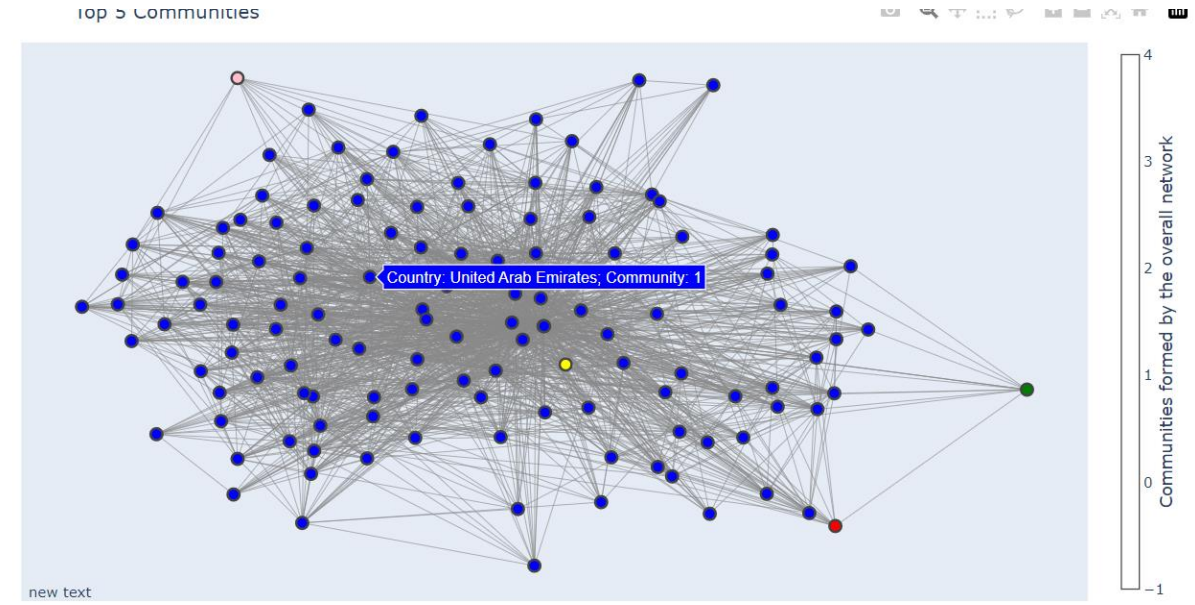

Country: Nigeria; Community: 3
new text
Node Connections
4
3
2
1
0
−1

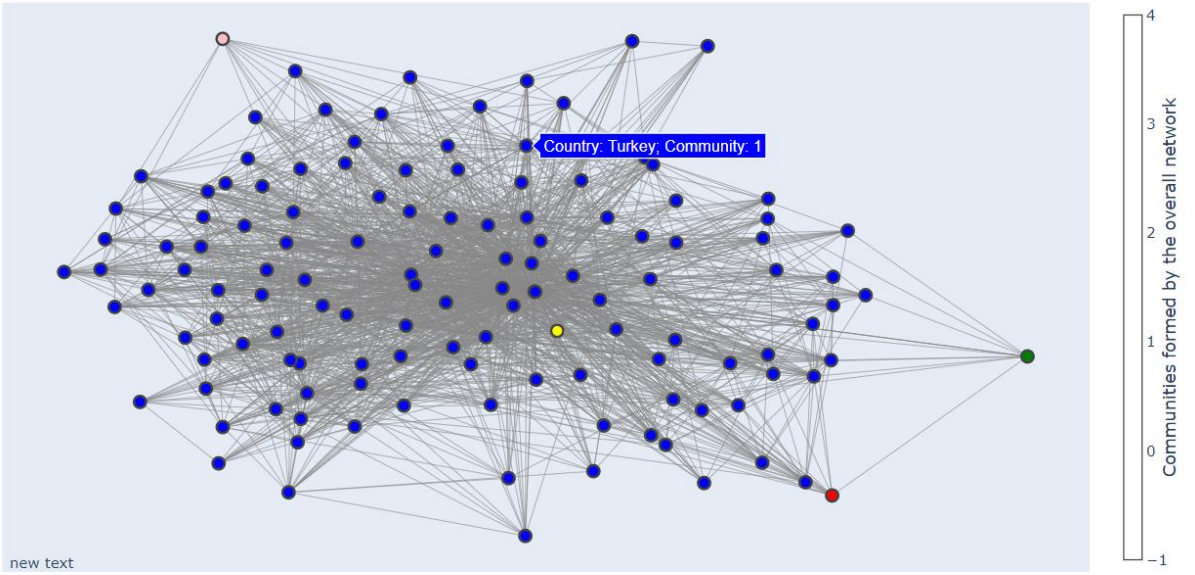



Top 5 Communities



**Implementing Girwan Newman algorithm on the whole Trade network including top players:**




It will form 5 communities among all the nodes, as top players are inclusive in this most of nodes are saturated in community 1.

# Results

From all the analysis I have performed, I observed that top countries are always remained the same through all the centralities and ranking measures. United States and China are always top players in the network. They are strongly connected to all the other countries in the network. By splitting the country nodes into communities, trade is affecting significantly because weak links are present in graph. As per the visualizations, presence of top players significantly overpowers the clustering of communities. That way we can continue regular flow of goods and services to connected countries.