

| **TITLE : Performing Graph Analytics** |
| --- |

**AIM:** To analyze the structural properties of a real-world social network by constructing a graph representation, identifying key players and influential individuals through centrality measures, and detecting communities within the network using appropriate algorithms.

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**Expected OUTCOME of Experiment:**

CO3: Perform the social data analytics

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**Books/ Journals/ Websites referred:**

Students have to list.

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**Pre Lab/ Prior Concepts:**

Students should have a basic understanding of:

Graph theory: Nodes, edges, directed and undirected graphs, weighted graphs.

Data structures: Lists, dictionaries.

Python programming: Basic syntax, data manipulation, libraries like NetworkX.

Statistical concepts: Mean, standard deviation, correlation.

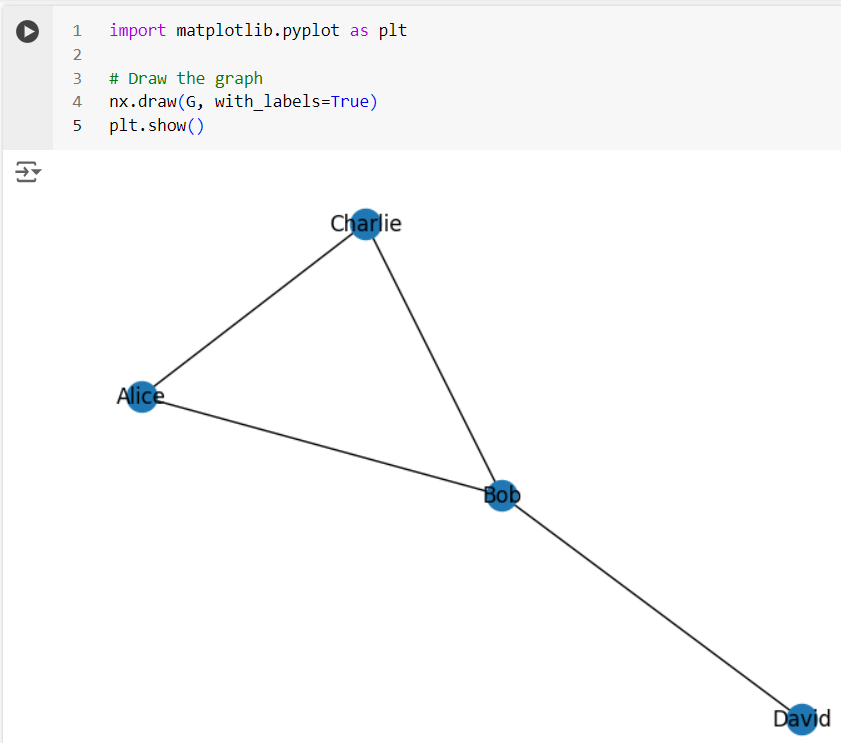
Visualization techniques: Basic plotting using libraries like Matplotlib.

**Procedure:**

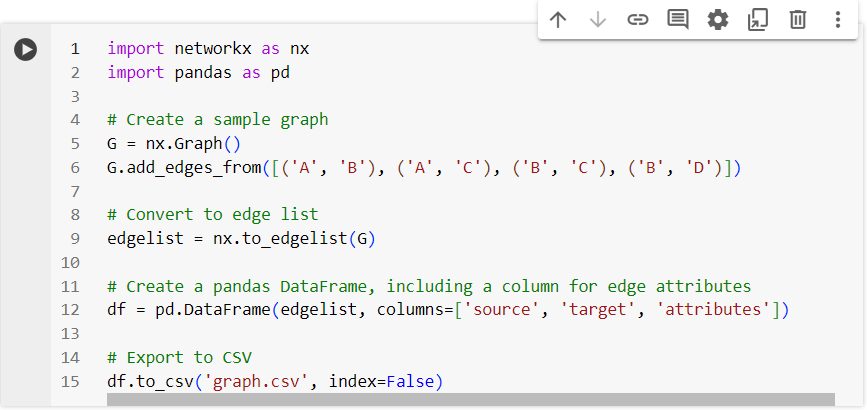
### Building a Social Network Graph with NetworkX



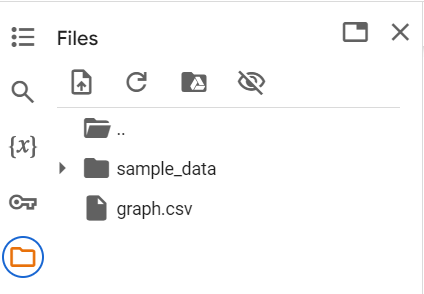
### Visualizing the Graph



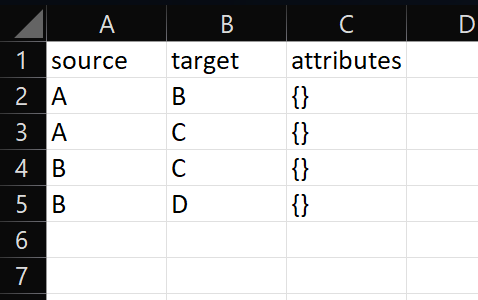
**Exporting a NetworkX Graph to CSV**

****

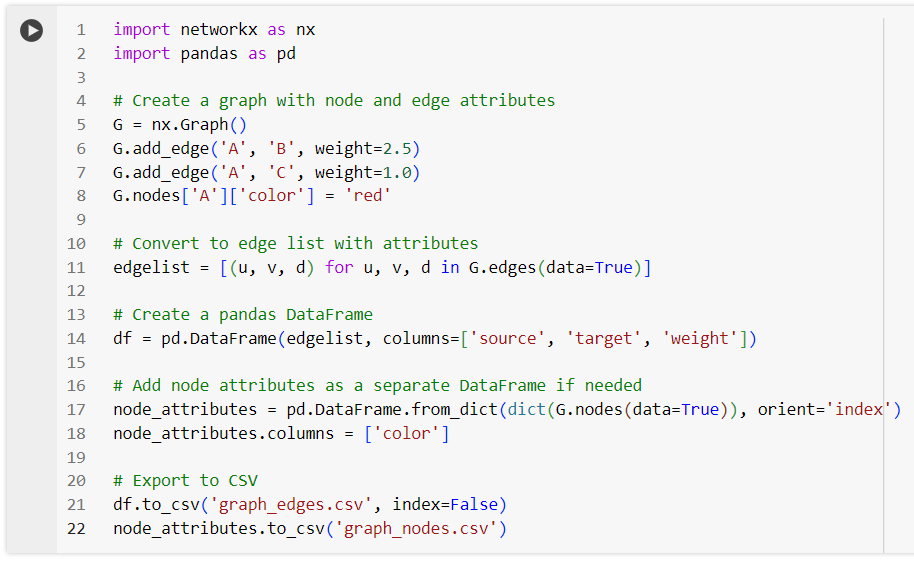
**The csv file gets created**



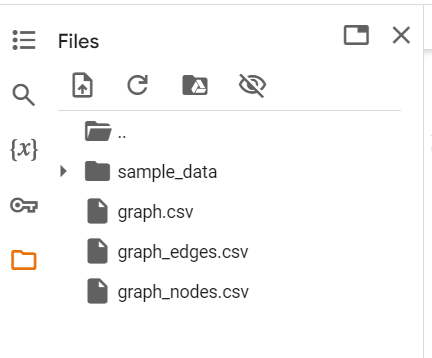
**Contents of the csv file**



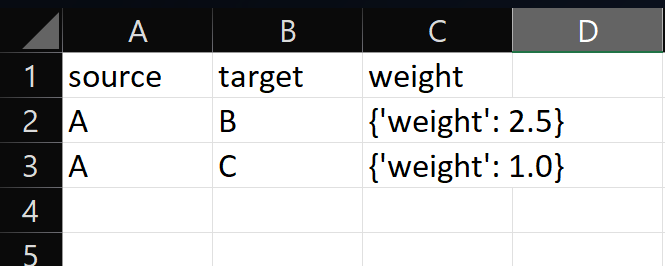
**Creating and exporting a NetworkX Graph with edge attributes and node attributes to a csv file**

****

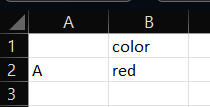
**csv files get created**

****

**Contents of graph\_edges.csv**

****

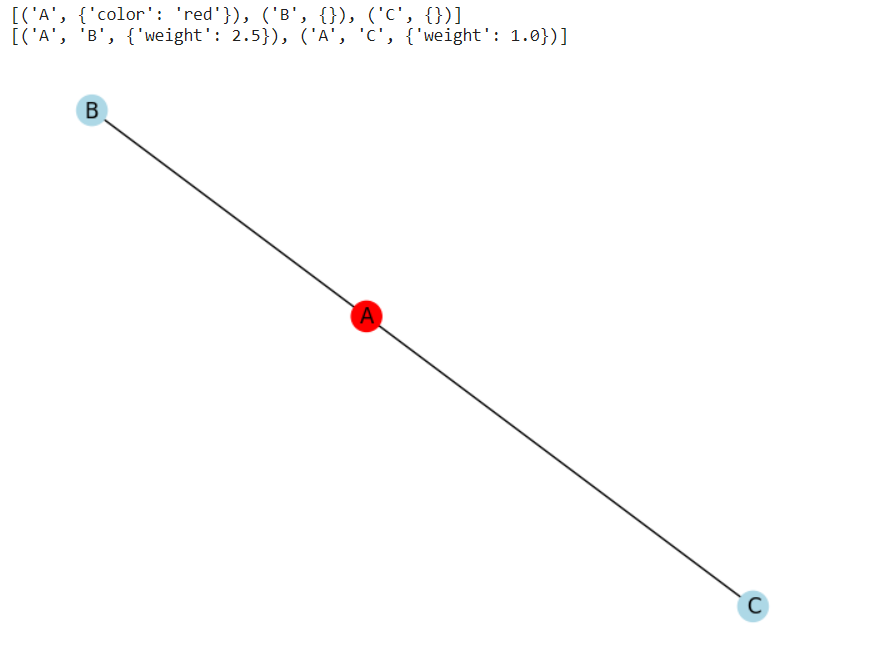
**Contents of graph\_nodes.csv**

****

**Importing a graph from a csv file**

****

**Output (List of nodes and edges, and visualizing the imported graph)**

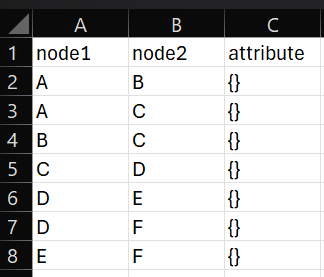
****

**Graph Analytics**

1. Degree centrality : The degree centrality for a node v is the fraction of nodes it is connected to. The degree centrality values are normalized by dividing by the maximum possible degree in a simple graph n-1 where n is the number of nodes in G.
2. Betweenness centrality : Betweenness centrality of a node v is the sum of the fraction of all-pairs shortest paths that pass through v. The betweenness centrality is normalized by dividing by the total number of shortest paths.
3. Edge betweenness centrality : Betweenness centrality of a node e is the sum of the fraction of all-pairs shortest paths that pass through e. The betweenness centrality is normalized by dividing by the maximum possible number of edges in a graph G.
4. Communities can be identified using the Girvan Newman algorithm, by successively deleting the edges with the highest betweenness centrality values.

**Importing a graph from csv file and performing graph analytics**

**The graph in csv file:**

****

**Importing the graph, printing its edge list and visualizing it:**

**import pandas as pd**

**import networkx as nx**

**import matplotlib.pyplot as plt**

**# Read edge list from CSV**

**df\_edges = pd.read\_csv('new\_graph\_edges.csv')**

**# Create a graph from the edge list**

**G = nx.from\_pandas\_edgelist(df\_edges,source='node1', target='node2')**

**# Print the graph**

**print(G.nodes(data=True))**

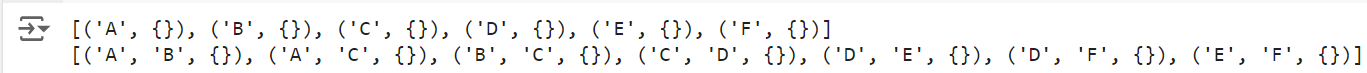
**print(G.edges(data=True))**

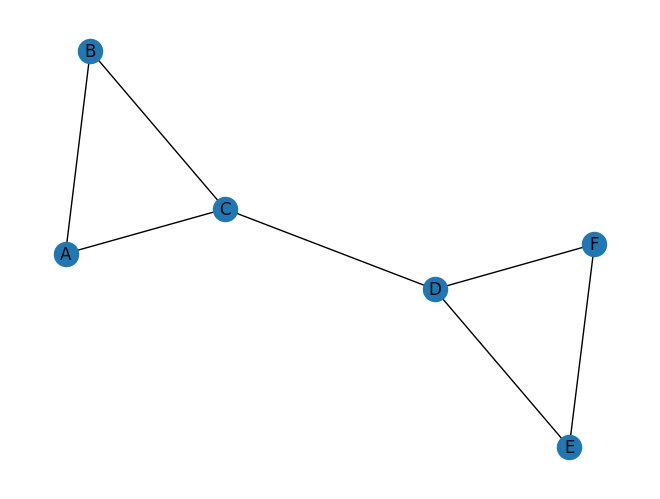
**# Draw the graph**

**nx.draw(G, with\_labels=True)**

**plt.show()**

**Output (graph details and visualization):**

****

****

**Performing analytics on this graph:**

**# Basic graph properties**

**print("Number of nodes:", G.number\_of\_nodes())**

**print("Number of edges:", G.number\_of\_edges())**

**# Degree centrality**

**degrees = dict(G.degree())**

**print("\nDegree Centrality:", degrees)**

**# Betweenness centrality**

**betweenness = nx.betweenness\_centrality(G, normalized=False)**

**print("\nBetweenness Centrality:", betweenness)**

**betweenness = nx.betweenness\_centrality(G)**

**print("Normalized Betweenness Centrality:", betweenness)**

**# Closeness centrality**

**e\_betwenness = nx.edge\_betweenness\_centrality(G,normalized=False)**

**print("\nEdge Betweenness Centrality:", e\_betwenness)**

**e\_betwenness = nx.edge\_betweenness\_centrality(G)**

**print("Normalized Edge Betweenness Centrality:", e\_betwenness)**

**# Community detection (Girvan-Newman)**

**communities = nx.algorithms.community.girvan\_newman(G)**

**try:**

**top\_level\_communities = next(communities)**

**print("\nCommunities after 1 step:", top\_level\_communities)**

**top\_level\_communities = next(communities)**

**print("\nCommunities after 2 steps:", top\_level\_communities)**

**top\_level\_communities = next(communities)**

**print("\nCommunities after 3 steps:", top\_level\_communities)**

**top\_level\_communities = next(communities)**

**print("\nCommunities after 4 steps:", top\_level\_communities)**

**top\_level\_communities = next(communities)**

**print("\nCommunities after 5 steps:", top\_level\_communities)**

**except StopIteration:**

**print("\nNo more splits are possible.")**

**Output:**

**Number of nodes: 6**

**Number of edges: 7**

**Degree Centrality: {'A': 2, 'B': 2, 'C': 3, 'D': 3, 'E': 2, 'F': 2}**

**Betweenness Centrality: {'A': 0.0, 'B': 0.0, 'C': 6.0, 'D': 6.0, 'E': 0.0, 'F': 0.0}**

**Normalized Betweenness Centrality: {'A': 0.0, 'B': 0.0, 'C': 0.6000000000000001, 'D': 0.6000000000000001, 'E': 0.0, 'F': 0.0}**

**Edge Betweenness Centrality: {('A', 'B'): 1.0, ('A', 'C'): 4.0, ('B', 'C'): 4.0, ('C', 'D'): 9.0, ('D', 'E'): 4.0, ('D', 'F'): 4.0, ('E', 'F'): 1.0}**

**Normalized Edge Betweenness Centrality: {('A', 'B'): 0.06666666666666667, ('A', 'C'): 0.26666666666666666, ('B', 'C'): 0.26666666666666666, ('C', 'D'): 0.6, ('D', 'E'): 0.26666666666666666, ('D', 'F'): 0.26666666666666666, ('E', 'F'): 0.06666666666666667}**

**Communities after 1 step: ({'A', 'C', 'B'}, {'E', 'F', 'D'})**

**Communities after 2 steps: ({'A'}, {'C', 'B'}, {'E', 'F', 'D'})**

**Communities after 3 steps: ({'A'}, {'B'}, {'C'}, {'E', 'F', 'D'})**

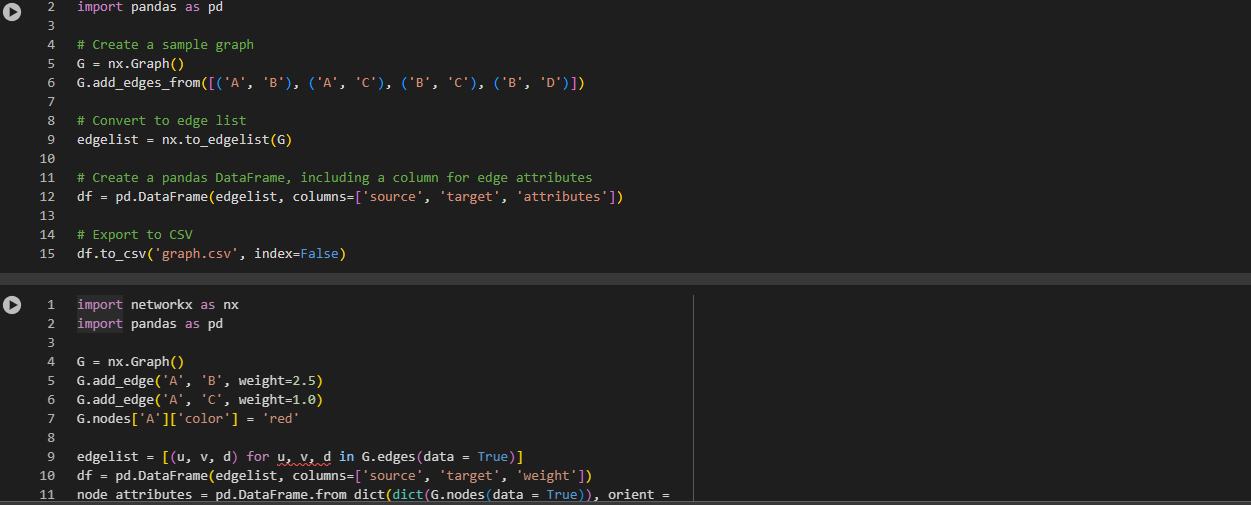
**Communities after 4 steps: ({'A'}, {'B'}, {'C'}, {'D'}, {'E', 'F'})**

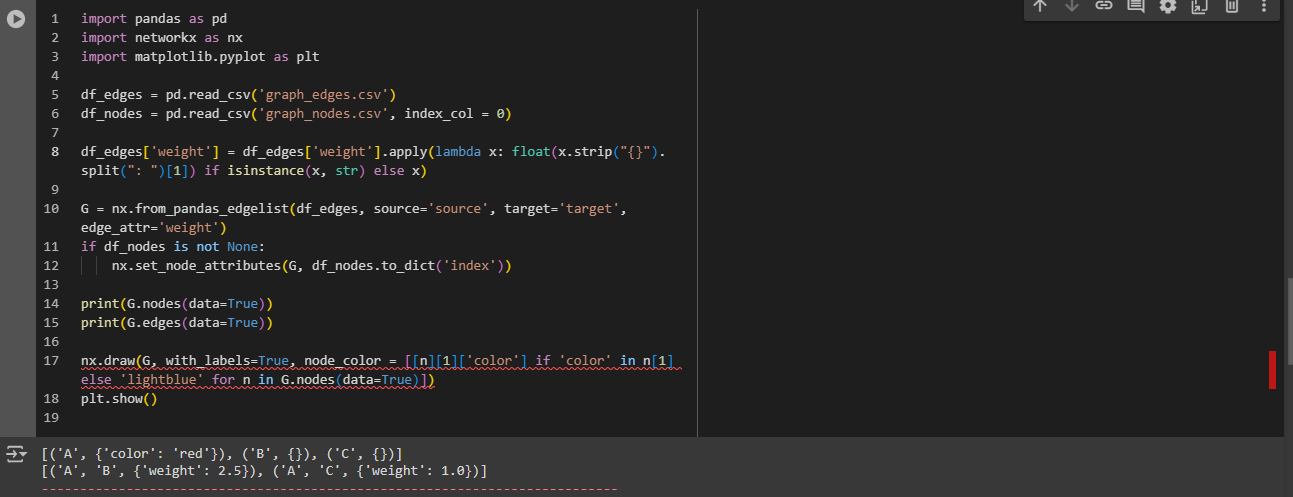
**Communities after 5 steps: ({'A'}, {'B'}, {'C'}, {'D'}, {'E'}, {'F'})**

**Students have to perform all the tasks illustrated above by creating a social network graph with nodes labelled with their own names and their friends’ names. The graph should have at least 10 nodes.**

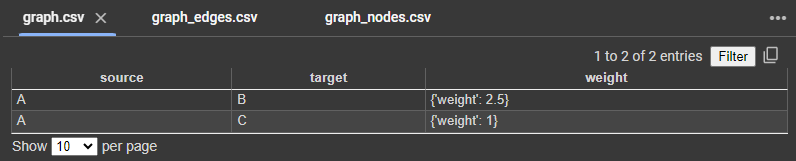
**Students have to paste their code and screenshots of output and csv file below.**

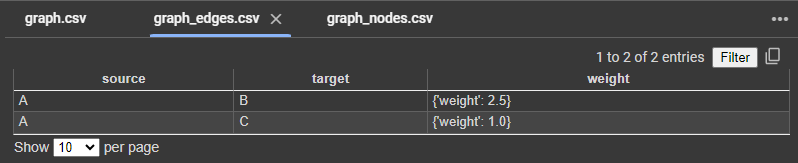
Implementation details:

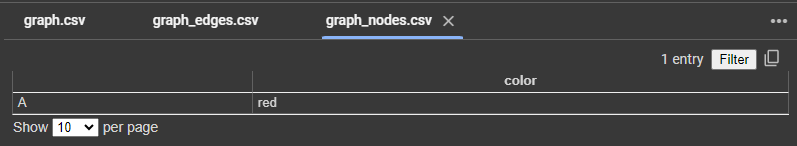




Output:

****

****

****

**Date: \_\_\_\_\_\_\_\_\_\_\_\_\_ Signature of faculty in-charge**

**Post Lab Descriptive Questions:**

1. Analyze the centrality measures you calculated. Which nodes were identified as the most influential? What does this mean in the context of the social network?
2. Describe the communities identified using the Girvan-Newman algorithm. What are the characteristics of these communities? How do they relate to the social network's structure?
3. Discuss the implications of identifying influential nodes in the network. How can this information be used?