**Batch: HDA2 Roll No.: 16010123012**

**Experiment No. 4**

# 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

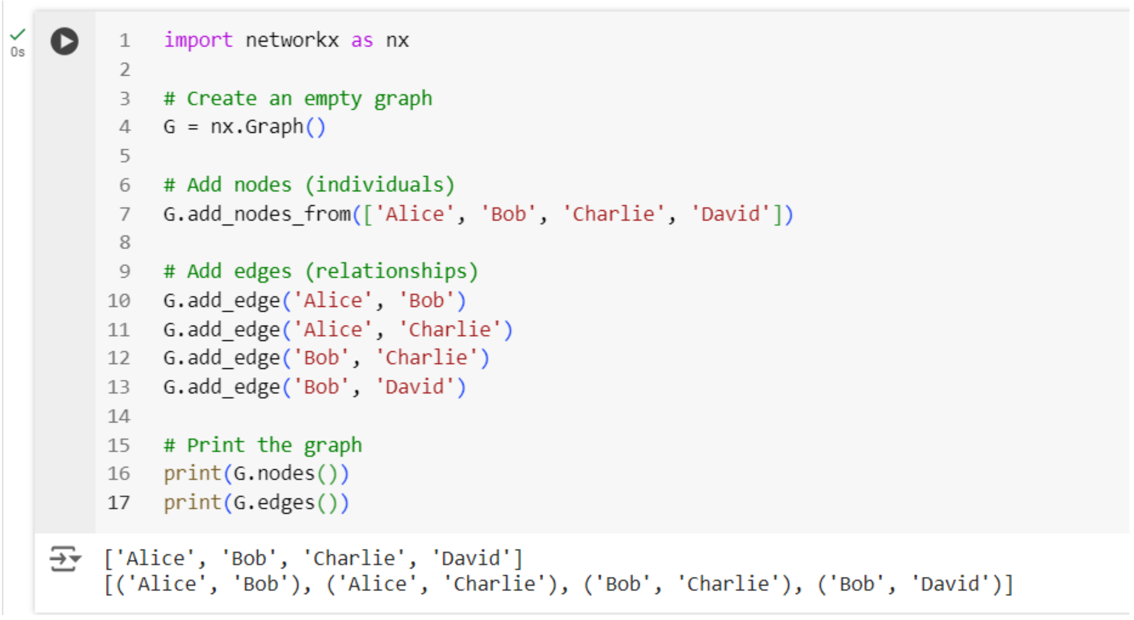
**Books/ Journals/ Websites referred:**

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:**

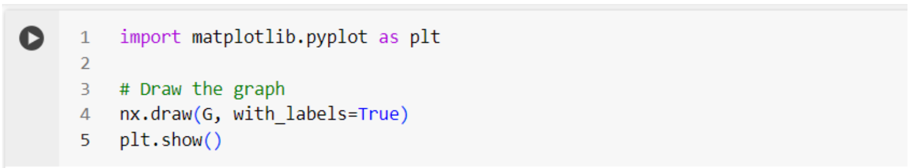
**Building a Social Network Graph with NetworkX**



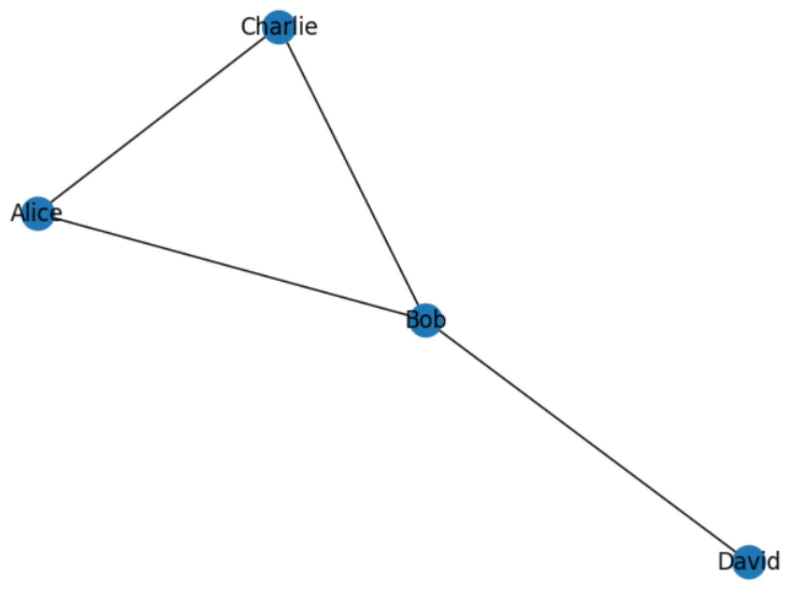
```python
import networkx as nx

# Create an empty graph
G = nx.Graph()

# Add nodes (individuals)
G.add_nodes_from(['Alice', 'Bob', 'Charlie', 'David'])

# Add edges (relationships)
G.add_edge('Alice', 'Bob')
G.add_edge('Alice', 'Charlie')
G.add_edge('Bob', 'Charlie')
G.add_edge('Bob', 'David')

# Print the graph
print(G.nodes())
print(G.edges())
```

```
['Alice', 'Bob', 'Charlie', 'David']
[('Alice', 'Bob'), ('Alice', 'Charlie'), ('Bob', 'Charlie'), ('Bob', 'David')]
```

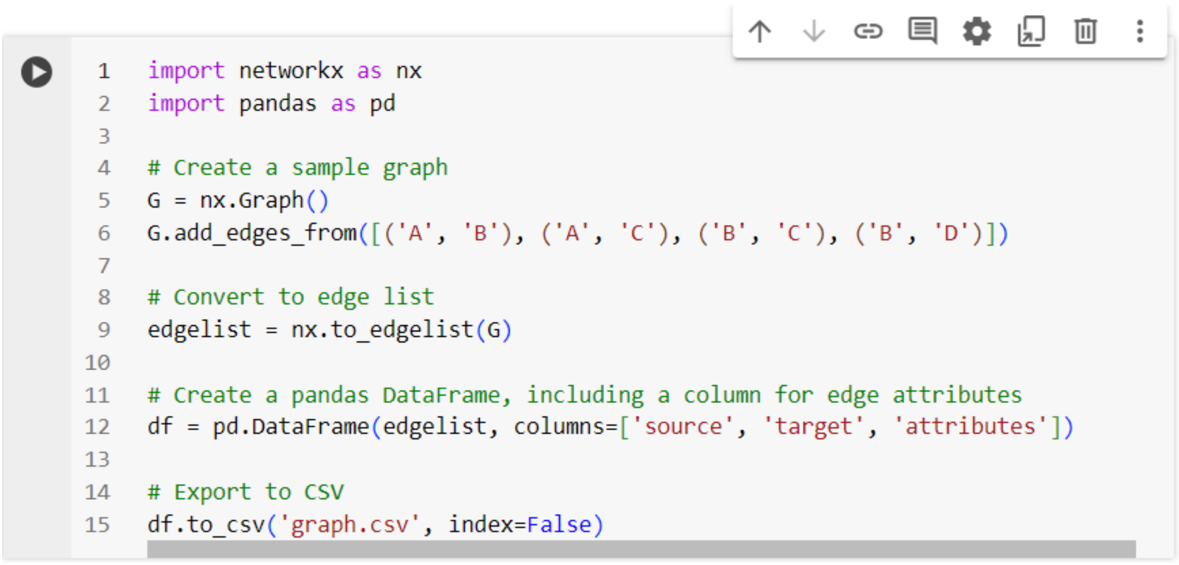
## Visualizing the Graph



```python
import matplotlib.pyplot as plt

# Draw the graph
nx.draw(G, with_labels=True)
plt.show()
```

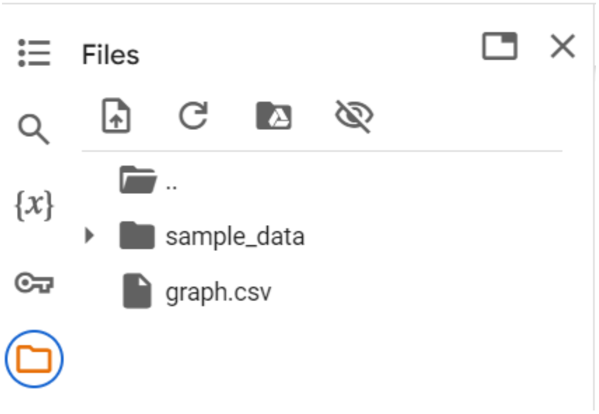


## Exporting a NetworkX Graph to CSV

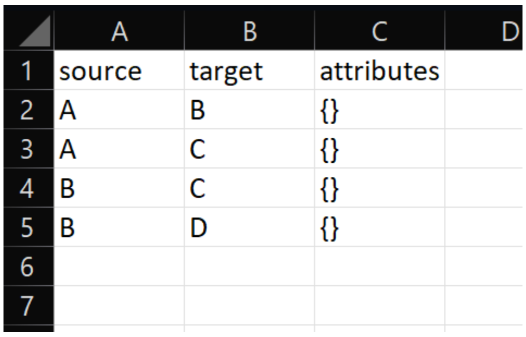


```python
import networkx as nx
import pandas as pd

# Create a sample graph
G = nx.Graph()
G.add_edges_from([('A', 'B'), ('A', 'C'), ('B', 'C'), ('B', 'D')])

# Convert to edge list
edgelist = nx.to_edgelist(G)

# Create a pandas DataFrame, including a column for edge attributes
df = pd.DataFrame(edgelist, columns=['source', 'target', 'attributes'])

# Export to CSV
df.to_csv('graph.csv', index=False)
```

**The csv file gets created**

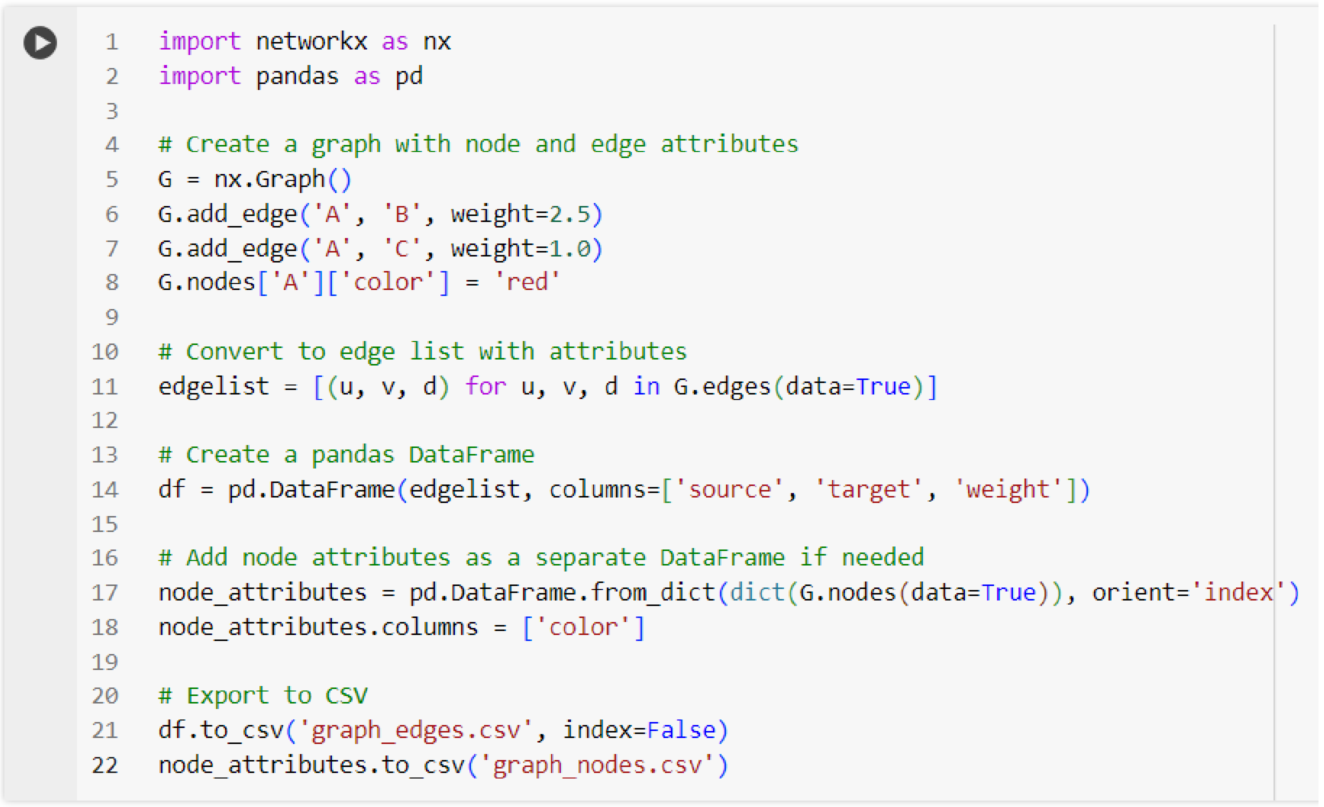


**Contents of the csv file**



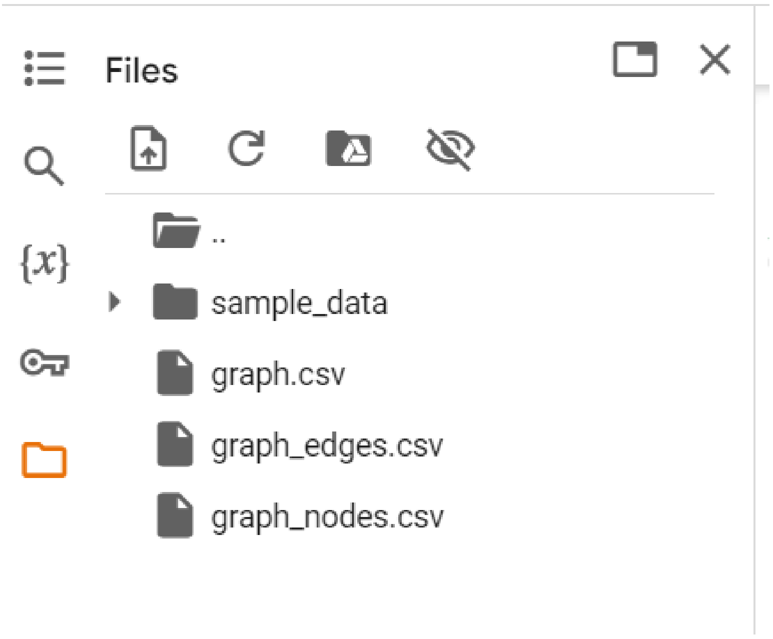
| | A | B | C | D |
|---|---|---|---|---|
| 1 | source | target | attributes | |
| 2 | A | B | {} | |
| 3 | A | C | {} | |
| 4 | B | C | {} | |
| 5 | B | D | {} | |
| 6 | | | | |
| 7 | | | | |

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

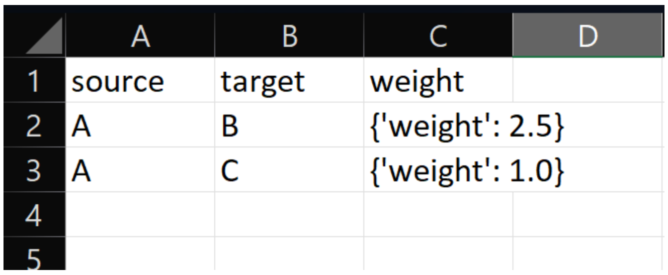


```python
import networkx as nx
import pandas as pd

# Create a graph with node and edge attributes
G = nx.Graph()
G.add_edge('A', 'B', weight=2.5)
G.add_edge('A', 'C', weight=1.0)
G.nodes['A']['color'] = 'red'

# Convert to edge list with attributes
edgelist = [(u, v, d) for u, v, d in G.edges(data=True)]

# Create a pandas DataFrame
df = pd.DataFrame(edgelist, columns=['source', 'target', 'weight'])

# Add node attributes as a separate DataFrame if needed
node_attributes = pd.DataFrame.from_dict(dict(G.nodes(data=True)), orient='index')
node_attributes.columns = ['color']

# Export to CSV
df.to_csv('graph_edges.csv', index=False)
node_attributes.to_csv('graph_nodes.csv')
```

**csv files get created**

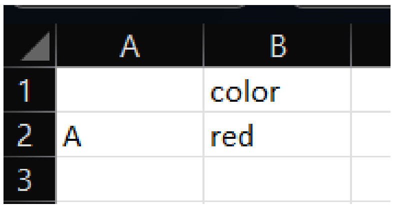


**Contents of graph_edges.csv**



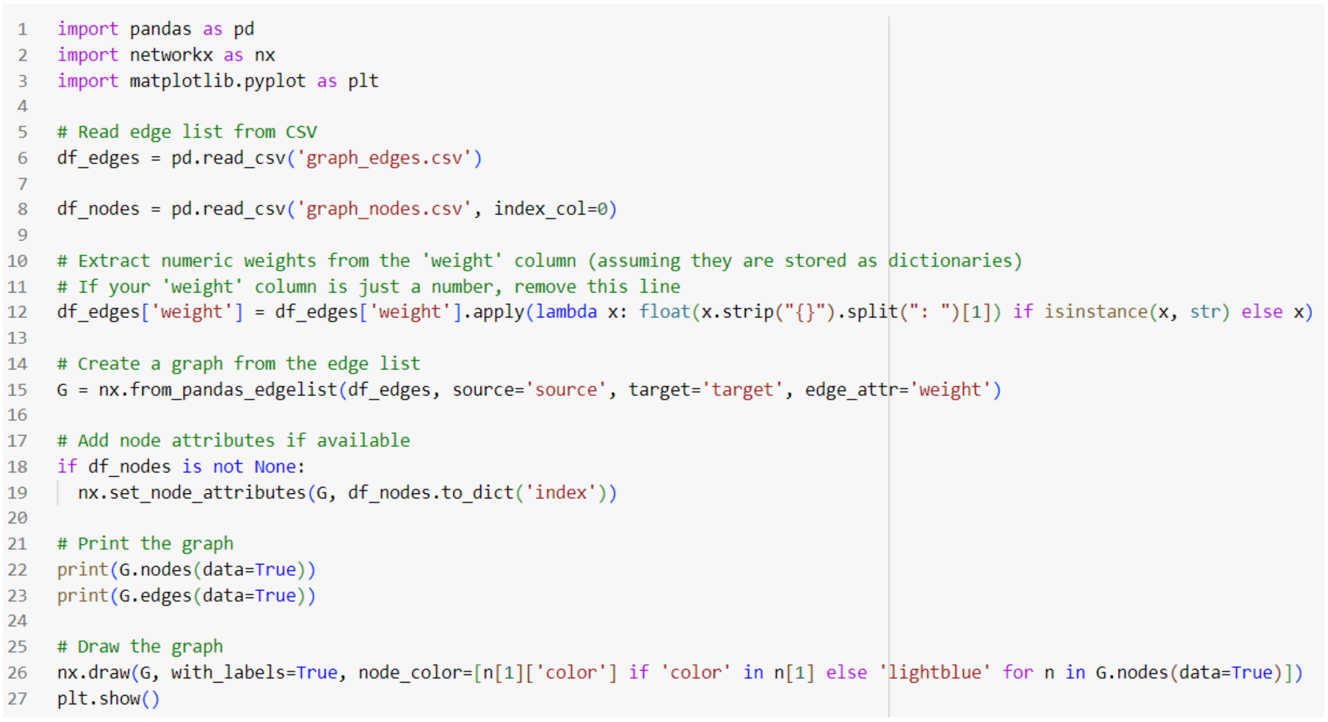
| | A | B | C | D |
|---|---|---|---|---|
| 1 | source | target | weight | |
| 2 | A | B | {'weight': 2.5} | |
| 3 | A | C | {'weight': 1.0} | |
| 4 | | | | |
| 5 | | | | |

**Contents of graph_nodes.csv**



| | A | B |
|---|---|---|
| 1 | | color |
| 2 | A | red |
| 3 | | |

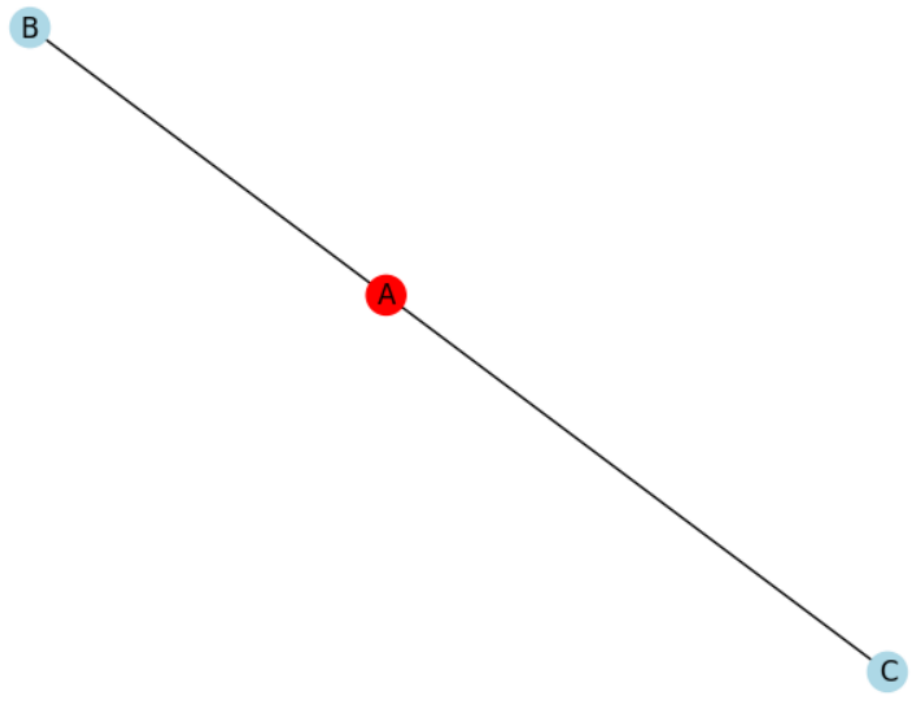
## Importing a graph from a csv file



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

# Read edge list from CSV
df_edges = pd.read_csv('graph_edges.csv')

df_nodes = pd.read_csv('graph_nodes.csv', index_col=0)

# Extract numeric weights from the 'weight' column (assuming they are stored as dictionaries)
# If your 'weight' column is just a number, remove this line
df_edges['weight'] = df_edges['weight'].apply(lambda x: float(x.strip("{}").split(": ")[1]) if isinstance(x, str) else x)

# Create a graph from the edge list
G = nx.from_pandas_edgelist(df_edges, source='source', target='target', edge_attr='weight')

# Add node attributes if available
if df_nodes is not None:
    nx.set_node_attributes(G, df_nodes.to_dict('index'))

# Print the graph
print(G.nodes(data=True))
print(G.edges(data=True))

# Draw the graph
nx.draw(G, with_labels=True, node_color=[n[1]['color'] if 'color' in n[1] else 'lightblue' for n in G.nodes(data=True)])
plt.show()
```

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

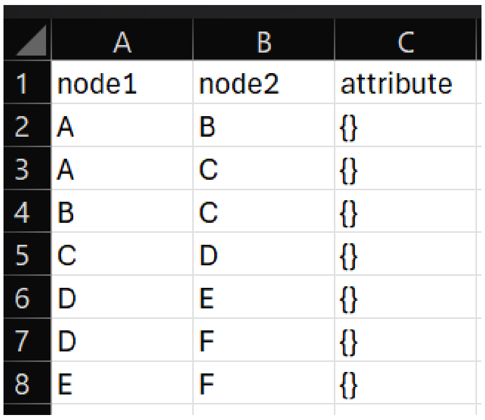
```
[('A', {'color': 'red'}), ('B', {}), ('C', {})]
[('A', 'B', {'weight': 2.5}), ('A', 'C', {'weight': 1.0})]
```



**<u>Graph Analytics</u>**

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:**

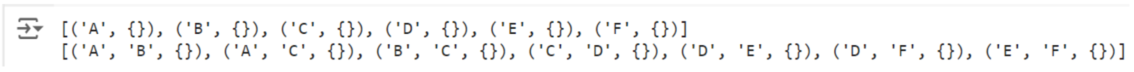


| | A | B | C |
|---|---|---|---|
| 1 | node1 | node2 | attribute |
| 2 | A | B | {} |
| 3 | A | C | {} |
| 4 | B | C | {} |
| 5 | C | D | {} |
| 6 | D | E | {} |
| 7 | D | F | {} |
| 8 | E | F | {} |

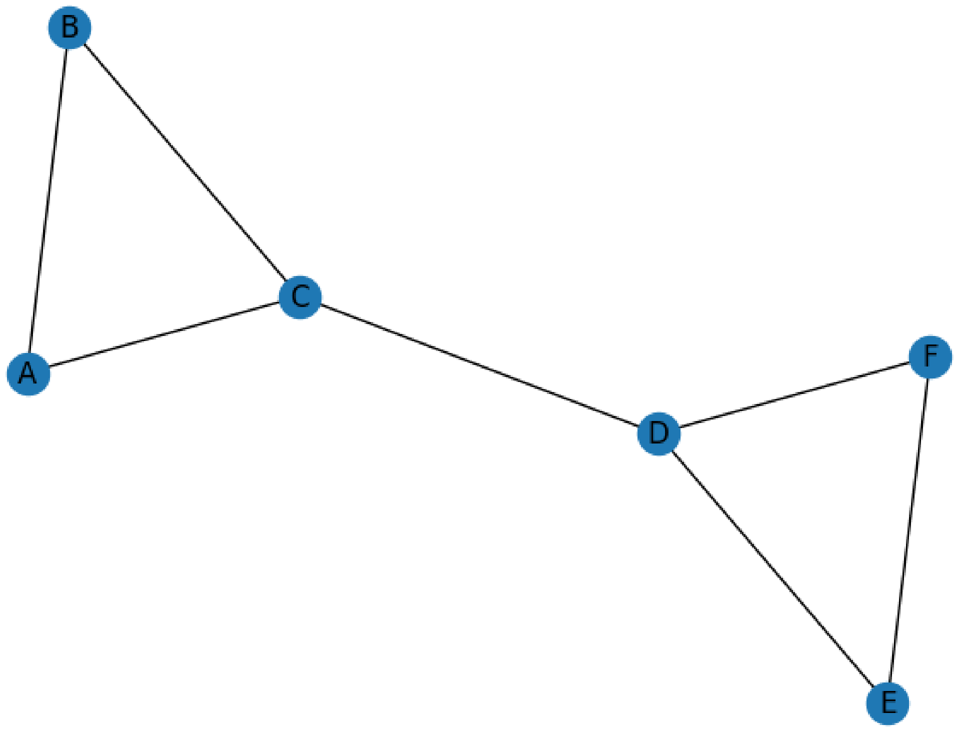
**Importing the graph, printing its edge list and visualizing it:**

```python
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):**



```
[('A', {}), ('B', {}), ('C', {}), ('D', {}), ('E', {}), ('F', {})]
[('A', 'B', {}), ('A', 'C', {}), ('B', 'C', {}), ('C', 'D', {}), ('D', 'E', {}), ('D', 'F', {}), ('E', 'F', {})]
```



**Performing analytics on this graph:**

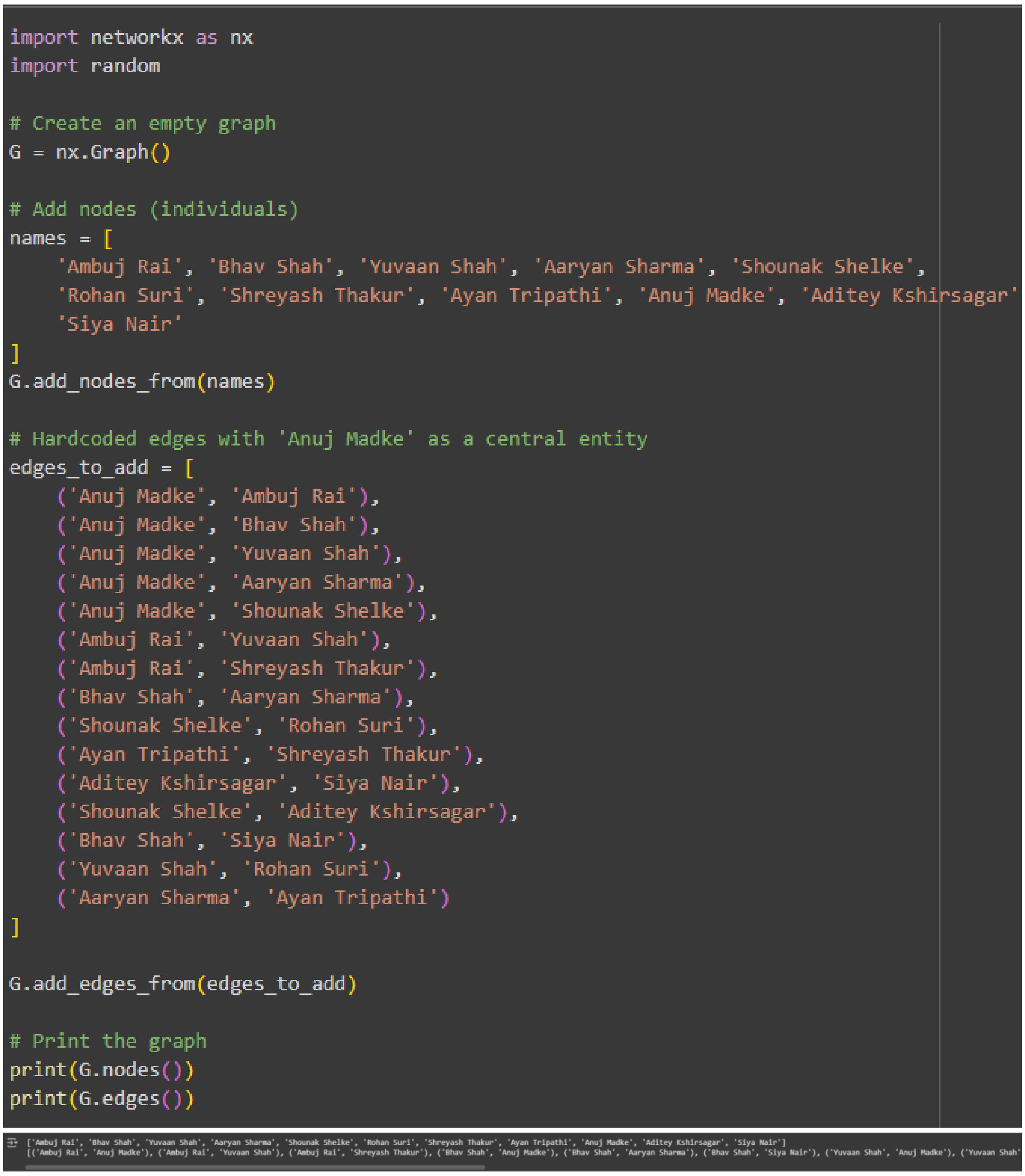
```
# Basic graph properties
print("Number of nodes:", G.number_of_nodes())
print("Number of edges:", G.number_of_edges())

# 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_betweenness = nx.edge_betweenness_centrality(G,normalized=False)
print("\nEdge Betweenness Centrality:", e_betwenness)
e_betweenness = 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'})
```

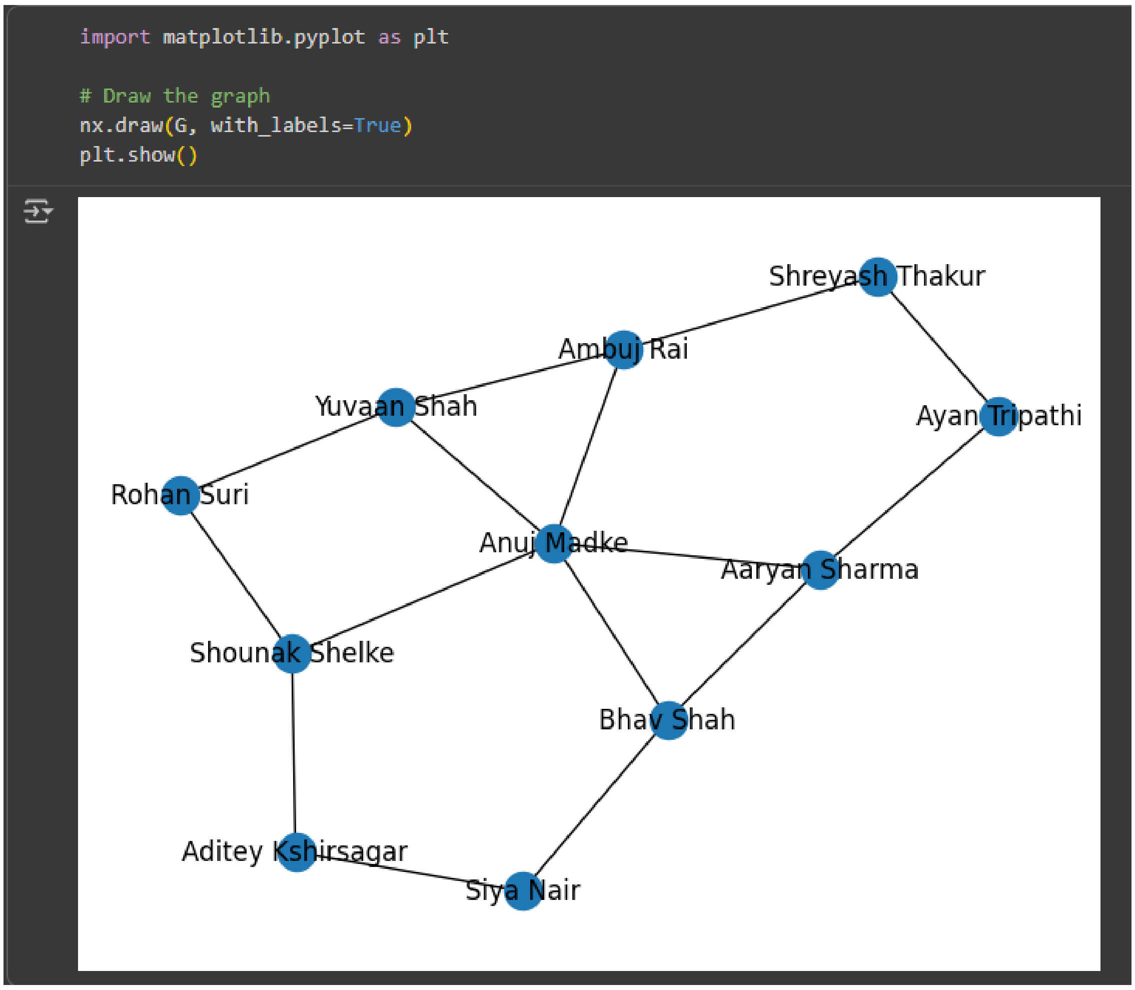
**Students have to perform all the tasks illustrated above by creating a social network graph with nodes labelled with their own names and their friends' names. The graph should have at least 10 nodes.**
**Students have to paste their code and screenshots of output and csv file below.**

Implementation details:

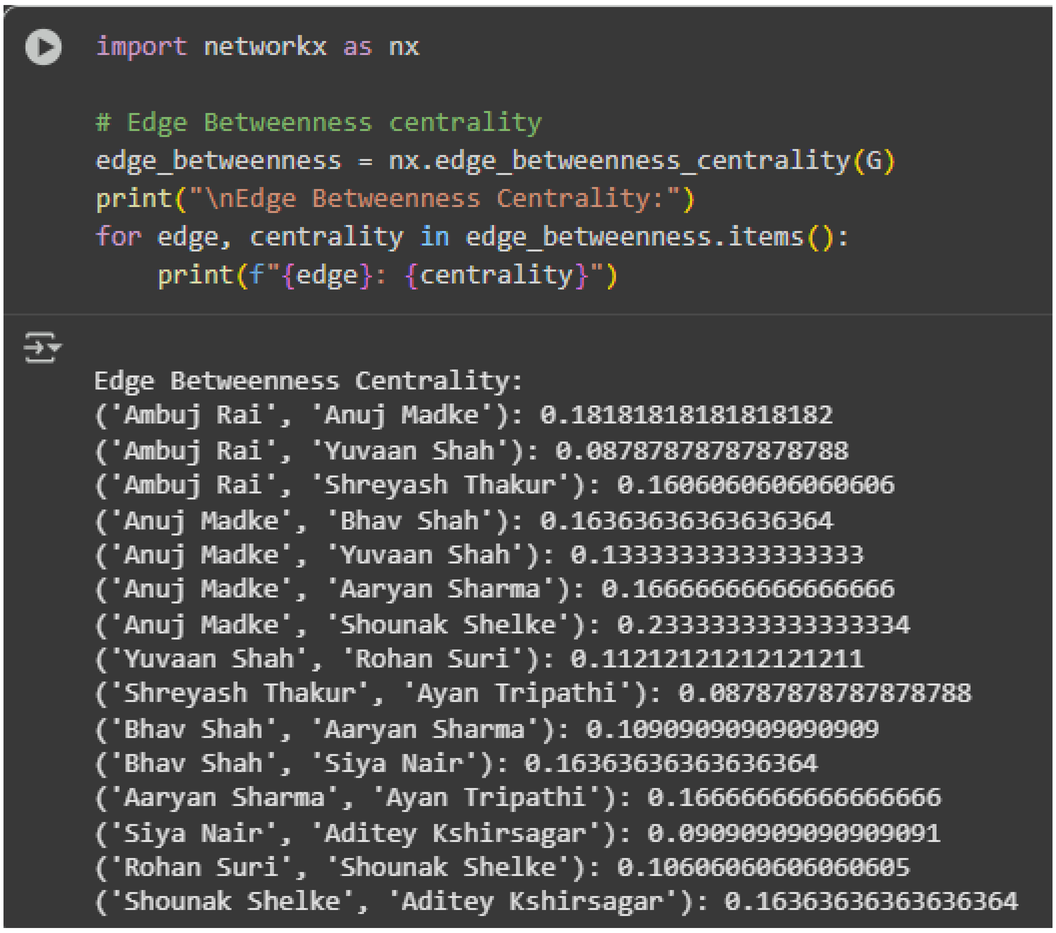


```
import networkx as nx
import random

# Create an empty graph
G = nx.Graph()

# Add nodes (individuals)
names = [
    'Ambuj Rai', 'Bhav Shah', 'Yuvaan Shah', 'Aaryan Sharma', 'Shounak Shelke',
    'Rohan Suri', 'Shreyash Thakur', 'Ayan Tripathi', 'Anuj Madke', 'Aditey Kshirsagar'
    'Siya Nair'
]
G.add_nodes_from(names)

# Hardcoded edges with 'Anuj Madke' as a central entity
edges_to_add = [
    ('Anuj Madke', 'Ambuj Rai'),
    ('Anuj Madke', 'Bhav Shah'),
    ('Anuj Madke', 'Yuvaan Shah'),
    ('Anuj Madke', 'Aaryan Sharma'),
    ('Anuj Madke', 'Shounak Shelke'),
    ('Ambuj Rai', 'Yuvaan Shah'),
    ('Ambuj Rai', 'Shreyash Thakur'),
    ('Bhav Shah', 'Aaryan Sharma'),
    ('Shounak Shelke', 'Rohan Suri'),
    ('Ayan Tripathi', 'Shreyash Thakur'),
    ('Aditey Kshirsagar', 'Siya Nair'),
    ('Shounak Shelke', 'Aditey Kshirsagar'),
    ('Bhav Shah', 'Siya Nair'),
    ('Yuvaan Shah', 'Rohan Suri'),
    ('Aaryan Sharma', 'Ayan Tripathi')
]

G.add_edges_from(edges_to_add)

# Print the graph
print(G.nodes())
print(G.edges())
```

```
['Ambuj Rai', 'Bhav Shah', 'Yuvaan Shah', 'Aaryan Sharma', 'Shounak Shelke', 'Rohan Suri', 'Shreyash Thakur', 'Ayan Tripathi', 'Anuj Madke', 'Aditey Kshirsagar', 'Siya Nair']
[('Ambuj Rai', 'Anuj Madke'), ('Ambuj Rai', 'Yuvaan Shah'), ('Ambuj Rai', 'Shreyash Thakur'), ('Bhav Shah', 'Anuj Madke'), ('Bhav Shah', 'Aaryan Sharma'), ('Bhav Shah', 'Siya Nair'), ('Yuvaan Shah', 'Anuj Madke'), ('Yuvaan Shah'
```



```
import matplotlib.pyplot as plt

# Draw the graph
nx.draw(G, with_labels=True)
plt.show()
```



```python
import networkx as nx

# Edge Betweenness centrality
edge_betweenness = nx.edge_betweenness_centrality(G)
print("\nEdge Betweenness Centrality:")
for edge, centrality in edge_betweenness.items():
    print(f"{edge}: {centrality}")
```

```
Edge Betweenness Centrality:
('Ambuj Rai', 'Anuj Madke'): 0.18181818181818182
('Ambuj Rai', 'Yuvaan Shah'): 0.08787878787878788
('Ambuj Rai', 'Shreyash Thakur'): 0.16060606060606060606
('Anuj Madke', 'Bhav Shah'): 0.16363636363636364
('Anuj Madke', 'Yuvaan Shah'): 0.13333333333333333
('Anuj Madke', 'Aaryan Sharma'): 0.16666666666666666
('Anuj Madke', 'Shounak Shelke'): 0.23333333333333334
('Yuvaan Shah', 'Rohan Suri'): 0.11212121212121211
('Shreyash Thakur', 'Ayan Tripathi'): 0.08787878787878788
('Bhav Shah', 'Aaryan Sharma'): 0.10909090909090909
('Bhav Shah', 'Siya Nair'): 0.16363636363636364
('Aaryan Sharma', 'Ayan Tripathi'): 0.16666666666666666
('Siya Nair', 'Aditey Kshirsagar'): 0.09090909090909091
('Rohan Suri', 'Shounak Shelke'): 0.10606060606060605
('Shounak Shelke', 'Aditey Kshirsagar'): 0.16363636363636364
```

**Date: 25/08/2025** **Signature of faculty in-charge**