```
#Jamaree Moyer Database Design & Implementation
# Lab Assignment # 2 Database Design And Implementation


# if no module found, install using this command: !pip install networkx
import networkx as nx


#if no module found, install using this command: !pip install matplotlib
import matplotlib.pyplot as plt


#creat graph to represent the social network of students and their connections
G = nx.Graph()


#student list
students = ["Alice", "Bob", "Charlie", "David", "Eve", "Frank", "Grace"]

#add students as nodes on the graph
G.add_nodes_from(students)


print(students)
```

```
['Alice', 'Bob', 'Charlie', 'David', 'Eve', 'Frank', 'Grace']
```

```
#lish of connections between students, represents a connection between two students
connections = [
    ("Alice", "Bob"),
    ("ALice", "Charlie"),
    ("Bob", "Charlie"),
    ("Bob", "David"),
    ("Charlie", "Eve"),
    ("David", "Eve"),
    ("Eve", "Frank"),
    ("Frank", "Grace"),
    ("Grace", "Eve")
]

#add connections as edges to the graph
G.add_edges_from(connections)


print(connections)
```

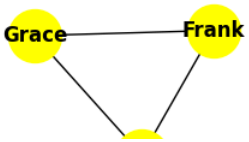
```
[('Alice', 'Bob'), ('ALice', 'Charlie'), ('Bob', 'Charlie'), ('Bob', 'David'), ('Charlie', 'Eve'), ('David', 'Eve'), ('Eve', 'Frank'), ('Frank', 'Grace'
```

```
#print basic information about the graph
print("Nodes of the graph:", G.nodes())
print("Edges of the graph:", G.edges())
print("Number of the nodes:", G.number_of_nodes())
print("Number of edges:", G.number_of_edges())
```

```
Nodes of the graph: ['Alice', 'Bob', 'Charlie', 'David', 'Eve', 'Frank', 'Grace', 'ALice']
Edges of the graph: [('Alice', 'Bob'), ('Bob', 'Charlie'), ('Bob', 'David'), ('Charlie', 'ALice'), ('Charlie', 'Eve'), ('David', 'Eve'), ('Eve', 'Frank'
Number of the nodes: 8
Number of edges: 9
```

```
#visualize network
nx.draw(G, with_labels=True, font_weight='bold', node_color='yellow', node_size=1000, edge_color='black')
plt.title("Social Network Graph Model")
plt.show()
```

Social Network Graph Model




```
#cnetrality means a network is directly connected to many others (degree centrality)
degree_centrality = nx.degree_centrality(G)
print("\nDegree Centrality:")
for student, centrality in degree_centrality.items():
  print(f"{student}: {centrality:.2f}")
```

```
Degree Centrality:
Alice: 0.14
Bob: 0.43
Charlie: 0.43
David: 0.29
Eve: 0.57
Frank: 0.29
Grace: 0.29
ALice: 0.14
```

```
# serve as a key broker between many other nodes (betweenness centrality)
betweenness_centrality = nx.betweenness_centrality(G)
print("\nBetweenness Centrality:")
for student, centrality in betweenness_centrality.items():
  print(f"{students}: {centrality:.2f}")
```

```
Betweenness Centrality:
['Alice', 'Bob', 'Charlie', 'David', 'Eve', 'Frank', 'Grace']: 0.00
['Alice', 'Bob', 'Charlie', 'David', 'Eve', 'Frank', 'Grace']: 0.33
['Alice', 'Bob', 'Charlie', 'David', 'Eve', 'Frank', 'Grace']: 0.43
['Alice', 'Bob', 'Charlie', 'David', 'Eve', 'Frank', 'Grace']: 0.14
['Alice', 'Bob', 'Charlie', 'David', 'Eve', 'Frank', 'Grace']: 0.52
['Alice', 'Bob', 'Charlie', 'David', 'Eve', 'Frank', 'Grace']: 0.00
['Alice', 'Bob', 'Charlie', 'David', 'Eve', 'Frank', 'Grace']: 0.00
['Alice', 'Bob', 'Charlie', 'David', 'Eve', 'Frank', 'Grace']: 0.00
```

```
#close to many others indirectly (closeness centrality)
closeness_centrality = nx.closeness_centrality(G)
print("\nCloseness Centrality:")
for student, centrality in closeness_centrality.items():
  print(f"{student}: {centrality:.2f}")
```

```
Closeness Centrality:
Alice: 0.37
Bob: 0.54
Charlie: 0.64
David: 0.54
Eve: 0.64
Frank: 0.44
Grace: 0.44
ALice: 0.41
```