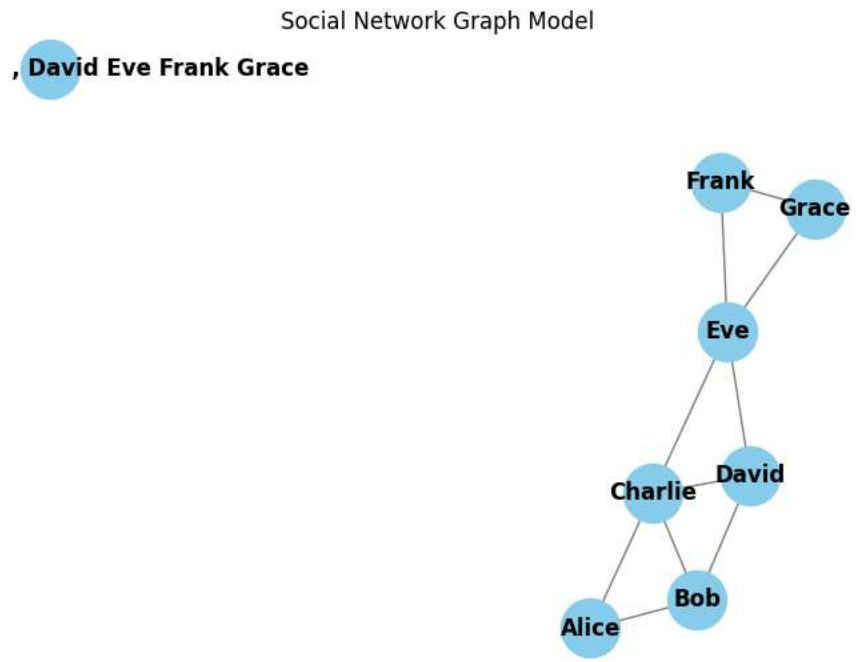
```
# Lab Assignment #2 Database Design and Implentation
# if no module found, install using this command: !pip install networkz
import networkx as nx
# if no module found, install using this command: !pip install matplotlib
import matplotlib.pyplot as plt
# create graph to represent the social network of students and their connectons
G = nx.Graph()
# student list
students = ["Alice, Bob Charlie, David Eve Frank Grace"]

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

# list of connections between students, represents a connceiton between two
students
connections = [
    ("Alice", "Bob"),
    ("Alice", "Charlie") ,
    ("Bob", "Charlie") ,
    ("Bob", "David"),
    ("Charlie", "David"),
    ("Charlie", "Eve"),
    ("David", "Eve"),
    ("Eve", "Frank"),
    ("Frank", "Grace"),
    ("Grace", "Eve")
]

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

print(connections)
#print basic informaiton about the graph
print("Nodes of the graph:", G.nodes())
print("edges of the graph:", G.edges())
print("Number of nodes:", G.number_of_nodes())
print("Number of edges:",  G.number_of_edges())
# visualize network
nx.draw(G, with_labels=True, font_weight='bold', node_color='skyblue' ,node_size=1000, edge_color='gray')
plt.title("Social Network Graph Model")
plt.show()
# centrality 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}")
    #serve as a key broker between many other nodes (betweenness centrality)
betweenness_centrality = nx.betweenness_centrality(G)
print("\nBetweeness Centtality:")
for student, centrality in betweenness_centrality.items():
    print(f"{student}: {centrality: .2f}")
# close to many other indirectly (closeness cintrality)
closeness_centrality = nx.closeness_centrality(G)
print("\nClosenes Centrality:")
for student, centrality in closeness_centrality. items():
    print(f"{student}: {centrality:.2f}")
```

```
['Alice, Bob Charlie, David Eve Frank Grace']
[('Alice', 'Bob'), ('Alice', 'Charlie'), ('Bob', 'Charlie'), ('Bob', 'David'), ('Charl
Nodes of the graph: ['Alice, Bob Charlie, David Eve Frank Grace', 'Alice', 'Bob', 'Cha
edges of the graph: [('Alice', 'Bob'), ('Alice', 'Charlie'), ('Bob', 'Charlie'), ('Bob
Number of nodes: 8
Number of edges: 10
```




```
Degree Centrality:
Alice, Bob Charlie, David Eve Frank Grace : 0.00
Alice : 0.29
Bob : 0.43
Charlie : 0.57
David : 0.43
Eve : 0.57
Frank : 0.29
Grace : 0.29

Betweeness Centtality:
Alice, Bob Charlie, David Eve Frank Grace:  0.00
Alice:  0.00
Bob:  0.02
Charlie:  0.24
David:  0.07
Eve:  0.38
Frank:  0.00
Grace:  0.00
```

```
# Lab Assigntment #2 Database Design and Implentation


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


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


# create graph to represent the social network of students and their connectons
G = nx.Graph()
```

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

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


print(students)
```

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

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

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


print(connections)
```

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

```
#print basic informaiton about the graph
print("Nodes of the graph:", G.nodes())
print("edges of the graph:", G.edges())
print("Number of 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', 'Bob', 'Charlie', 'David', 'Eve', 'Frank', 'Grace']
edges of the graph: [('Alice', 'Bob'), ('Alice', 'Charlie'), ('Bob', 'Charlie'), ('Bob', 'David'), ('Charlie', 'David'), ('Charlie', 'Ev
Number of nodes: 8
Number of edges: 10
```

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


```
# centrality 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, Bob Charlie, David Eve Frank Grace : 0.00
Alice : 0.29
Bob : 0.43
Charlie : 0.57
David : 0.43
Eve : 0.57
Frank : 0.29
Grace : 0.29
```

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

```
Betweeness Centtality:
Alice, Bob Charlie, David Eve Frank Grace:  0.00
Alice:  0.00
Bob:  0.02
Charlie:  0.24
David:  0.07
Eve:  0.38
Frank:  0.00
Grace:  0.00
```

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

```
Closenes Centrality:
Alice, Bob Charlie, David Eve Frank Grace: 0.00
Alice: 0.43
Bob: 0.47
Charlie: 0.64
David: 0.57
Eve: 0.64
Frank: 0.43
Grace: 0.43
```