sma-lab1

February 25, 2023

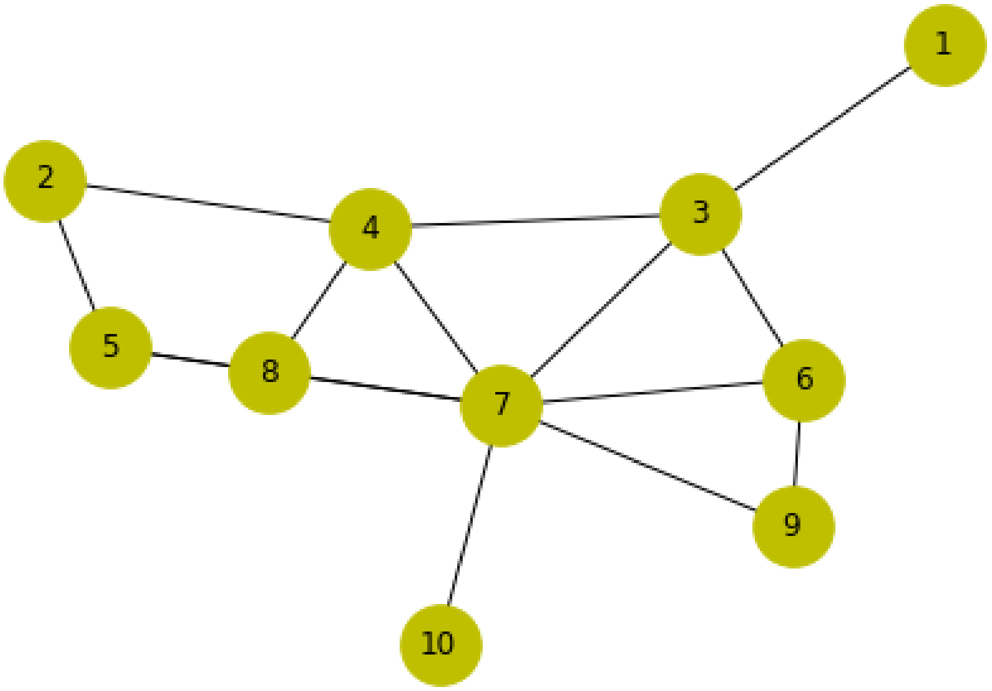
1 LAB 1 PART- A

1. Visualizing graph

[1]:
```
import networkx as nx
import pandas as pd
import matplotlib.pyplot as plt
```

[2]:
```
G = nx.Graph()
G.add_nodes_from([1, 2, 3, 4, 5, 6, 7, 8, 9, 10])
G.add_edges_from([(1,3), (2,4), (2,5), (3,4), (3,6), (3,7), (4,7), (4,8),␣
 ↪(5,7), (5,8), (6,7), (6,9), (7,8), (7,9), (7,10)])
```

[ ]:
```
nx.draw(G, with_labels = True, node_color = 'y', node_size = 1000)
plt.show()
```



2. Degree centrality of all nodes

[ ]:
```
DC = nx.degree_centrality(G)
df = pd.DataFrame.from_dict(DC, orient='index', columns=['  Degree Centrality'])
print(df)
```

```
       Degree Centrality
1               0.111111
2               0.222222
3               0.444444
4               0.444444
5               0.333333
6               0.333333
7               0.777778
8               0.333333
9               0.222222
10              0.111111
```

3. Neighbors of 2

[ ]:
```
twoNeighbors = list(G.neighbors(2))
print(twoNeighbors)
```

```
[4, 5]
```

4. Average degree of graph

[ ]:
```
avg_deg = sum(dict(G.degree()).values())/len(G)
print(avg_deg)
```

```
3.0
```

5. Density of graph

[ ]:
```
density = nx.density(G)
print(density)
```

```
0.3333333333333333
```

6. Possible Paths from node 10 to node 4

[ ]:
```
path_possible = list(nx.all_simple_paths(G, source = 10, target = 4))
for path in path_possible:
  print(path)
```

```
[10, 7, 3, 4]
[10, 7, 4]
[10, 7, 5, 2, 4]
[10, 7, 5, 8, 4]
```

```
[10, 7, 6, 3, 4]
[10, 7, 8, 4]
[10, 7, 8, 5, 2, 4]
[10, 7, 9, 6, 3, 4]
```

7. Longest shortest path between 2 and 9

[ ]:
```python
lg_path = max([len(path) for path in nx.all_shortest_paths(G, source=2,␣
 ↪target=9)])
print(lg_path)
```

```
4
```

8. Shortest path between node 1 and 10

[ ]:
```python
shortest_path = nx.shortest_path(G, source=1, target=10)
print(shortest_path)
```

```
[1, 3, 7, 10]
```

9. Diameter, radius and eccentricity of graph

[ ]:
```python
print("Graph diameter is:", nx.diameter(G))
print("Graph radius is:", nx.radius(G))
print("Graph eccentricity is:", nx.eccentricity(G))
```

```
Graph diameter is: 3
Graph radius is: 2
Graph eccentricity is: {1: 3, 2: 3, 3: 2, 4: 2, 5: 3, 6: 3, 7: 2, 8: 3, 9: 3,
10: 3}
```

10. Trail, walk and circuit for nodes

[6]:
```python
paths = list(nx.all_simple_paths(G, source = 1, target = 10, cutoff = None))
trails = [p for p in paths if len(set(p)) == len(p)]
walks = [p for p in paths if len(set(p)) < len(p)]
walks = nx.cycle_basis(G)
circuits = [p for p in paths if len(p) > p[0] == p[-1]]
circuit = nx.cycle_basis(G)
print(f"Trails between nodes 1 and 10: {trails}")
print(f"Walks between nodes 1 and 10: {walks}")
print(f"Circuits between nodes 1 and 10: {circuits}")
```

```
Trails between nodes 1 and 10: [[1, 3, 4, 2, 5, 7, 10], [1, 3, 4, 2, 5, 8, 7,
10], [1, 3, 4, 7, 10], [1, 3, 4, 8, 5, 7, 10], [1, 3, 4, 8, 7, 10], [1, 3, 6, 7,
10], [1, 3, 6, 9, 7, 10], [1, 3, 7, 10]]
Walks between nodes 1 and 10: [[4, 7, 3], [6, 7, 3], [6, 9, 7], [4, 8, 7], [5,
8, 7], [4, 2, 5, 7]]
Circuits between nodes 1 and 10: []
```

11. Top five nodes by degree

[ ]:

```
degrees = dict(G.degree())
top_five_nodes = sorted(degrees, key=degrees.get, reverse = True)[:5]
print(f"Top 5 nodes by degree: {top_five_nodes}")
```

```
Top 5 nodes by degree: [7, 3, 4, 5, 6]
```

#LAB 1 PART-B

2 New Section

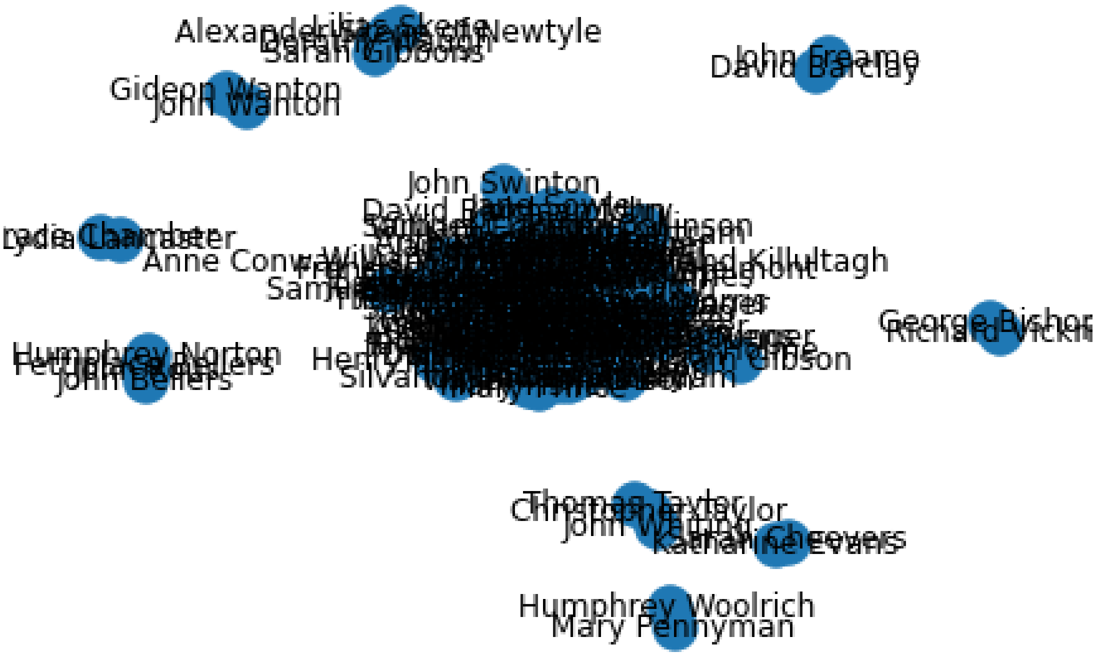
[ ]:

```
nodelist = pd.read_csv("quakers_nodelist.csv")
edgelist = pd.read_csv("quakers_edgelist.csv")

G = nx.from_pandas_edgelist(edgelist, 'Source', 'Target')

for i, r in nodelist.iterrows():
    G.nodes[r['Name']].update(r[1:].to_dict())
```

1. Visualizing graph

[ ]:

```
nx.draw(G, with_labels = True)
plt.show()
```



2. Number of nodes

[ ]:

```python
print("No. of nodes are: ", G.number_of_nodes())
```

```
No. of nodes are:  119
```

3. Number of edges

[ ]:

```python
print("No. of edges are: ", G.number_of_edges())
```

```
No. of edges are:  174
```

4. Identifying node attributes

[ ]:

```python
print("Node attributes are: ", nodelist.columns[0:].tolist())
```

```
Node attributes are:  ['Name', 'Historical Significance', 'Gender', 'Birthdate',
'Deathdate', 'ID']
```

5. Average degree of network

[ ]:

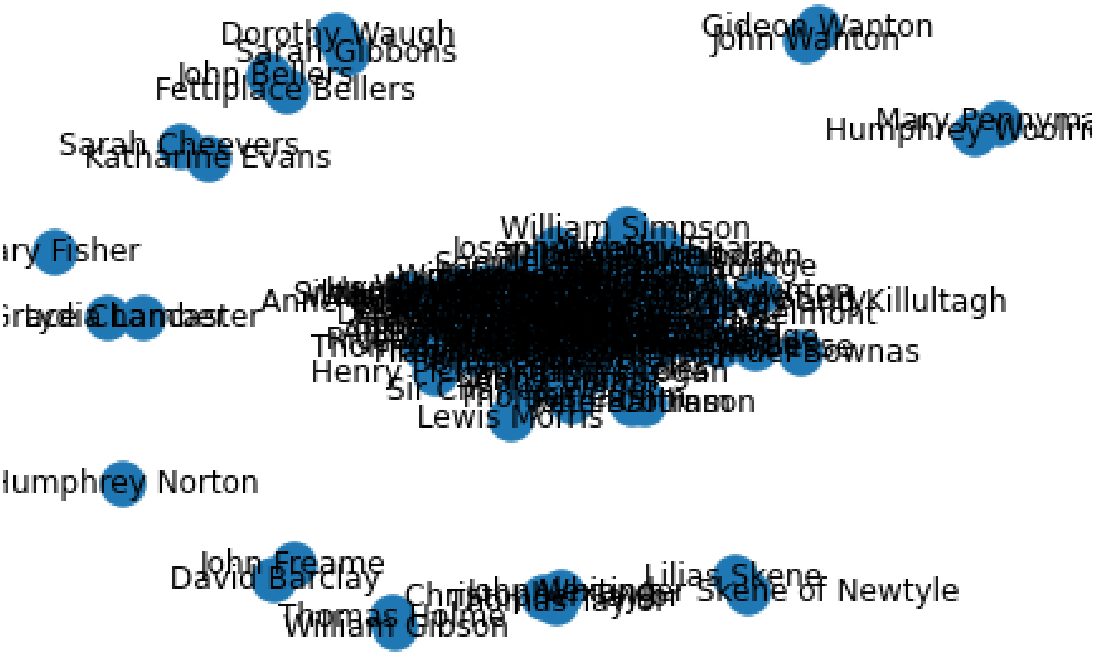
```python
avg_deg = sum(dict(G.degree()).values())/len(G)
print(avg_deg)
```

```
2.9243697478991595
```

6. Nodes with date of birth between 1600 to 1700

[ ]:

```python
required_nodes = nodelist[(nodelist['Birthdate'] >= 1600) & (nodelist['Birthdate'] < 1700)]['Name'].tolist()
birth_graph = G.subgraph(required_nodes)

nx.draw(birth_graph, with_labels = True)
plt.show()
```



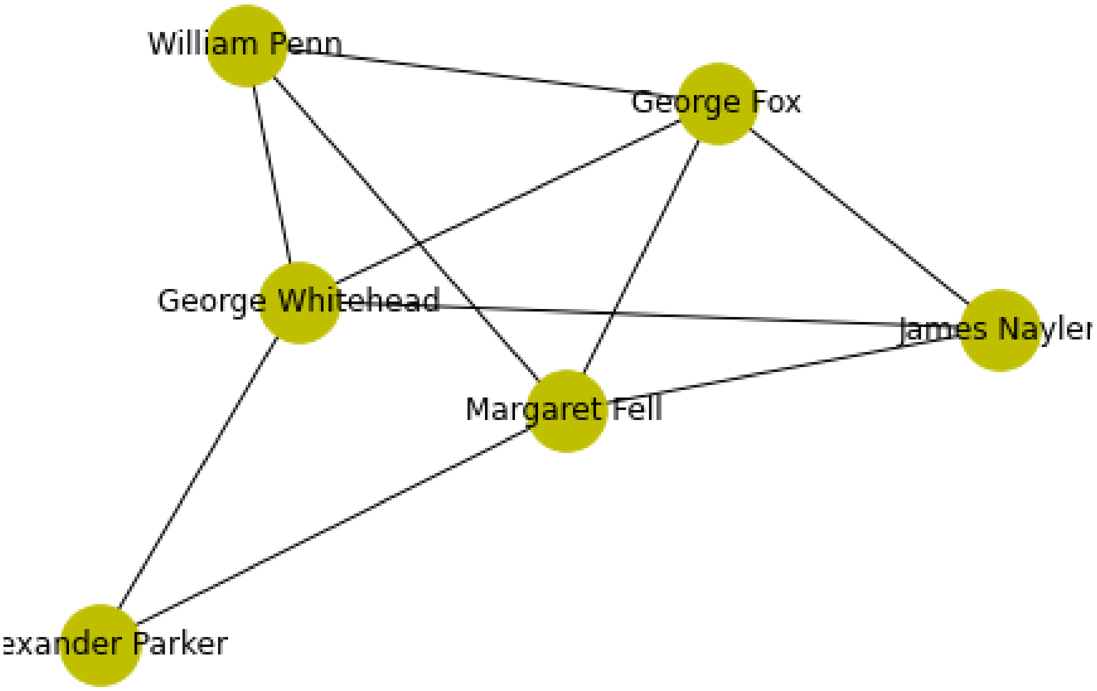
7. Network Density

```
[ ]: density = nx.density(G)
     print(density)
```

```
0.024782794473772169
```

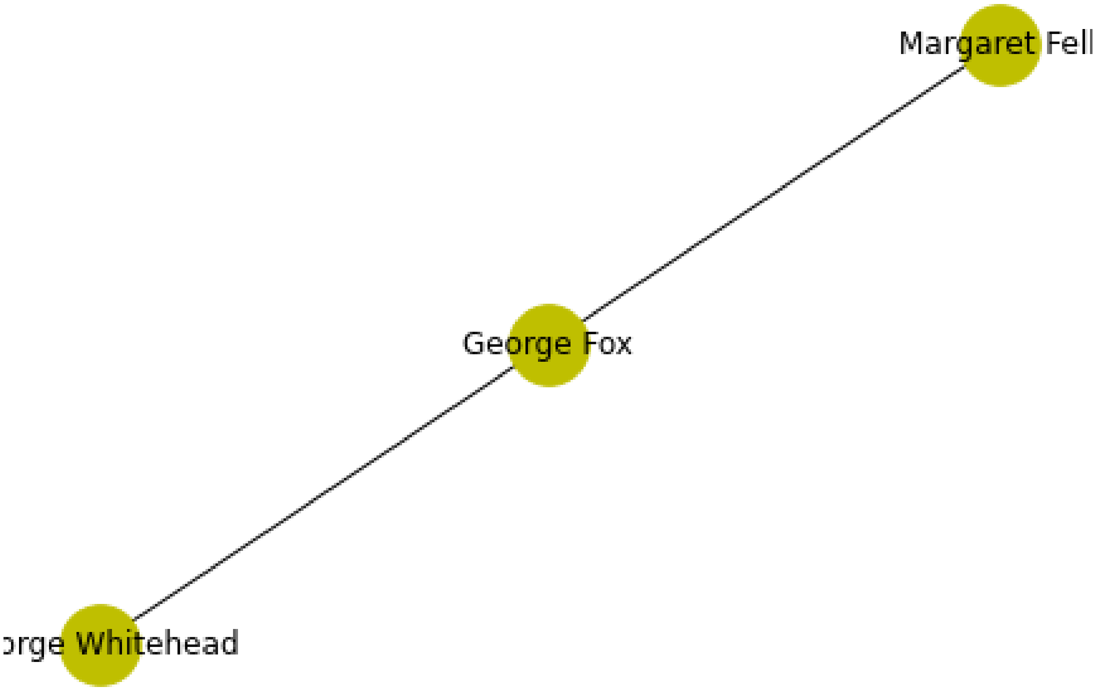
8. Paths between source= “Margaret Fell”, target = “George Whitehead”

```
[ ]: all_paths = list(nx.all_shortest_paths(G, source = "Margaret Fell", target = "George Whitehead"))

     required_nodes = [node for path in all_paths for node in path]
     subgraph = G.subgraph(required_nodes)

     nx.draw(subgraph, with_labels = True, node_color = 'y', node_size = 1000)
     plt.show()
```



9. Length of shortest path between source = “Margaret Fell”, target = “George Whitehead”

[ ]:
```
shortest_path = nx.shortest_path(G, source = "Margaret Fell", target = "George Whitehead")

subgraph = G.subgraph(shortest_path)

nx.draw(subgraph, with_labels = True, node_color = 'y', node_size = 1000)
plt.show()

print("Length of shortest path: ", len(shortest_path)-1)
```



```
Length of shortest path:  2
```

10. Top 25 nodes by degree

```
[ ]: topNodes = sorted(dict(G.degree()), key = lambda x: x[1], reverse = True)[:25]

     for node in topNodes:
         print(f"{node}")
```

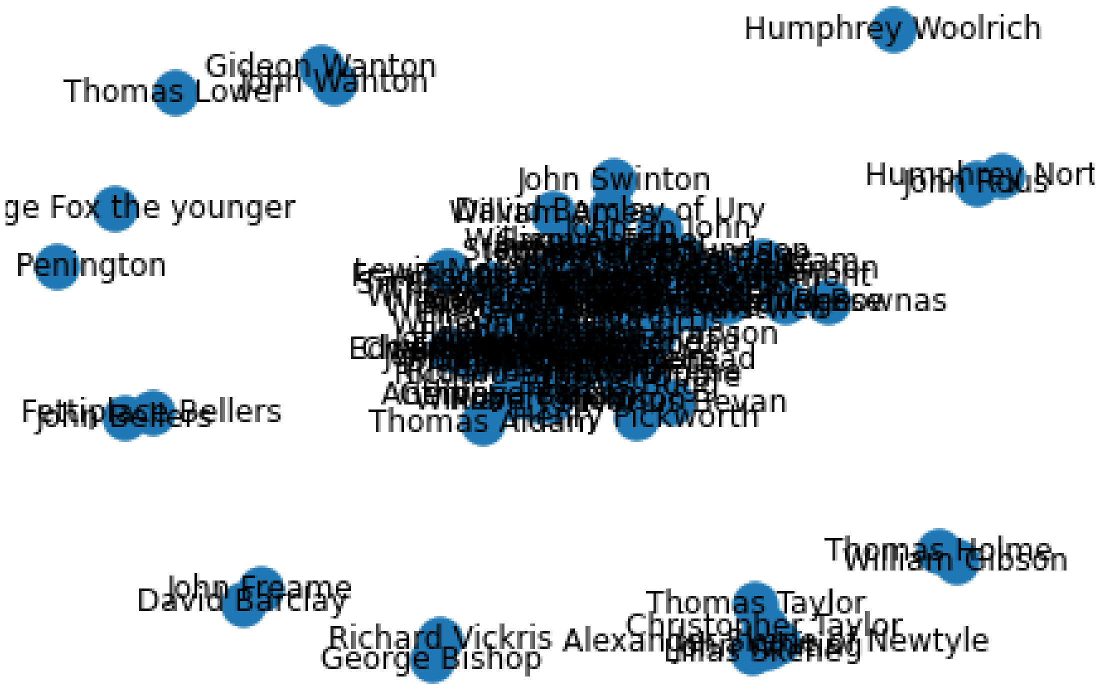
```
Lydia Lancaster
Humphrey Norton
Humphrey Woolrich
Stephen Crisp
Isabel Yeamans
Isaac Norris
Franciscus Mercurius van Helmont
Francis Howgill
Grace Chamber
Francis Bugg
Robert Barclay
John Bartram
Joseph Wyeth
Dorcas Erbery
John Crook
John Audland
John Camm
John Bellers
John Wilkinson
John Stubbs
John Perrot
John Burnyeat
John Story
Solomon Eccles
John Swinton
```

11. Number of nodes where gender is “Male”

[ ]:

```python
male_cnt = len(nodelist[nodelist['Gender'] == 'male'])

required_node = nodelist[nodelist['Gender'] == 'male']['Name'].tolist()
subgraph = G.subgraph(required_node)
print("Nodes with gender 'male':", male_cnt)

nx.draw(subgraph, with_labels = True)
plt.show()
```

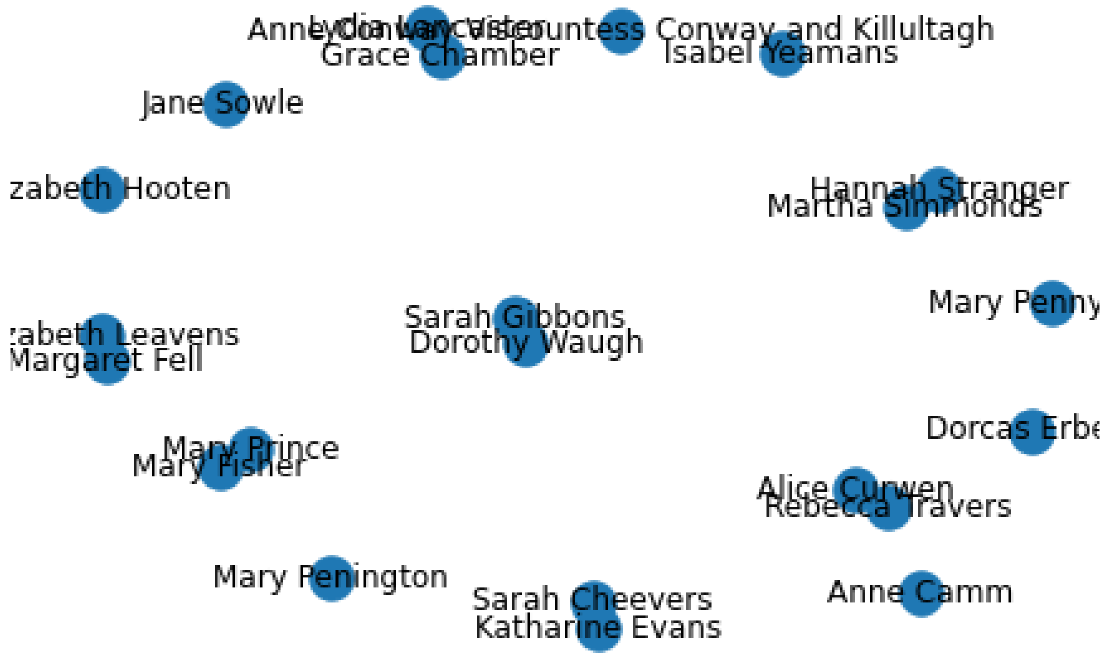
```
Nodes with gender 'male': 97
```



12. Number of nodes where gender is “Female”

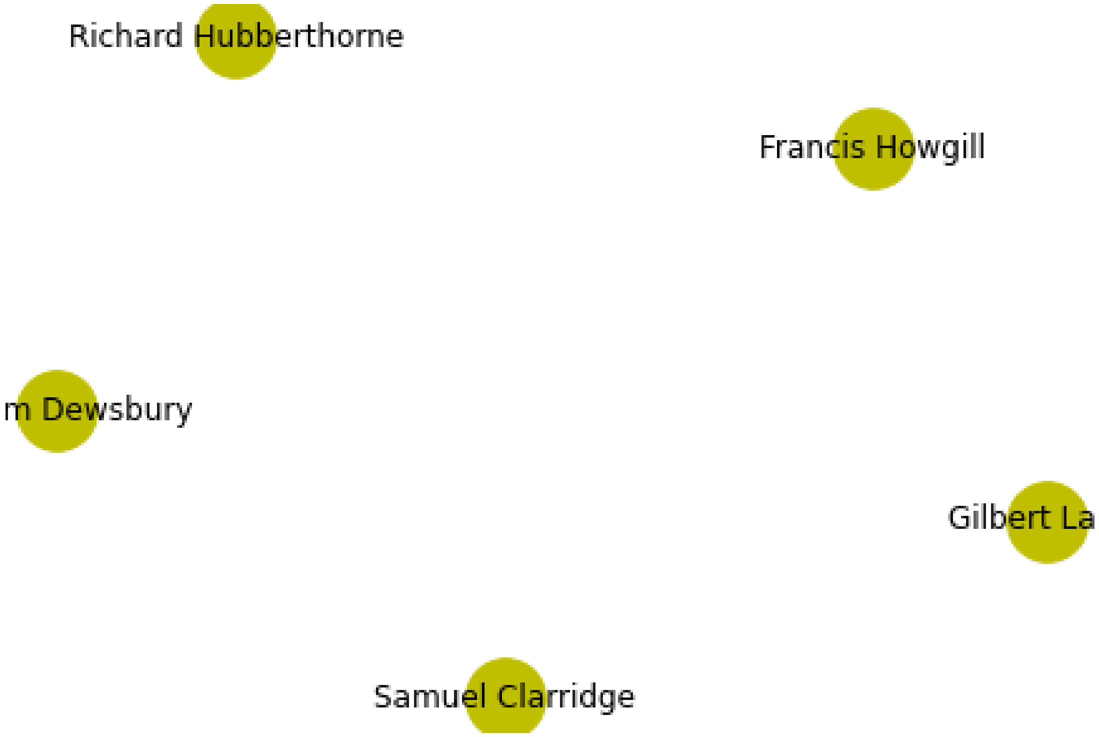
```
[ ]: female_cnt = len(nodelist[nodelist['Gender'] == 'female'])

     required_node = nodelist[nodelist['Gender'] == 'female']['Name'].tolist()
     subgraph = G.subgraph(required_node)
     print("Nodes with gender 'female':", female_cnt)

     nx.draw(subgraph, with_labels = True)
     plt.show()
```

```
Nodes with gender 'female': 22
```



13. “Quaker activist” who are “Male”

```
[ ]: required_node = nodelist[(nodelist['Gender'] == 'male') & (nodelist['Historical␣
      ↪Significance'] == 'Quaker activist')]['Name'].tolist()
     subgraph = G.subgraph(required_node)

     nx.draw(subgraph, with_labels = True, node_color = 'y', node_size = 1000)
     plt.show()

     print("Details of 'Quaker activist' who are 'male':")
print(nodelist[(nodelist['Gender'] == 'male') & (nodelist['Historical␣
  ↪Significance'] == 'Quaker activist')][['Name', 'Gender', 'Historical␣
  ↪Significance']])
```



```
Details of 'Quaker activist' who are 'male':
                    Name Gender Historical Significance
7       William Dewsbury   male         Quaker activist
22      Samuel Clarridge   male         Quaker activist
33         Gilbert Latey   male         Quaker activist
71       Francis Howgill   male         Quaker activist
72   Richard Hubberthorne   male         Quaker activist
```