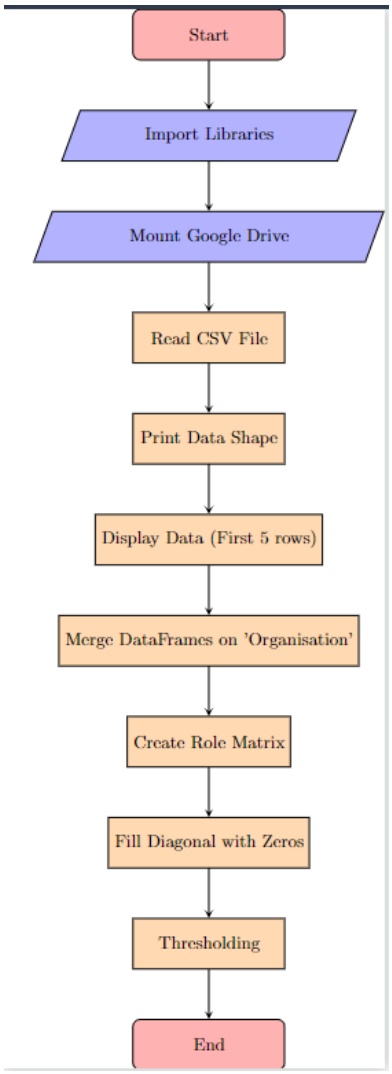
Experiment 1

Aim: Write a program to construct a social network object using Sociomatrix.

Flowchart:



Code:

```
#Sociomatrix- It is a square matrix where rows and columns
represent the nodes (individuals or entities) in the network
#and the entries in the matrix represent the presence or
absence of relationships between nodes.

#social network object
#a "social network object" refers to a data structure that
represents a social network or a network of relationships
between individuals or entities
```

```
import pandas as pd #Pandas is a Python library used for
working with data sets. #It has functions for analyzing,
cleaning, exploring, and manipulating data.
import numpy as np #. NumPy is used for working with arrays.
import networkx as nx #for working with graphs and networks
import matplotlib.pyplot as plt #for plotting graphs
%matplotlib inline
from operator import itemgetter #If multiple items are
specified, returns a tuple of lookup values. eg: itemgetter(1,
3, 5)('ABCDEFG') ->output:('B', 'D', 'F')

from google.colab import drive
drive.mount('/content/drive/')
df = pd.read_csv ("/content/drive/MyDrive/organization.csv")
print(df.shape)
df.head(5)
df_merge = df.merge(df,on="Organisation")
role_mat = pd.crosstab(df_merge.member_x,df_merge.member_y)
#This function allows you to compute a frequency table of two
or more variables # get counts of categories
np.fill_diagonal(role_mat.values, 0) # Return the filled value
in the diagonal of an array.
role_mat[role_mat >= 1] = 1
role_mat
```

Output:

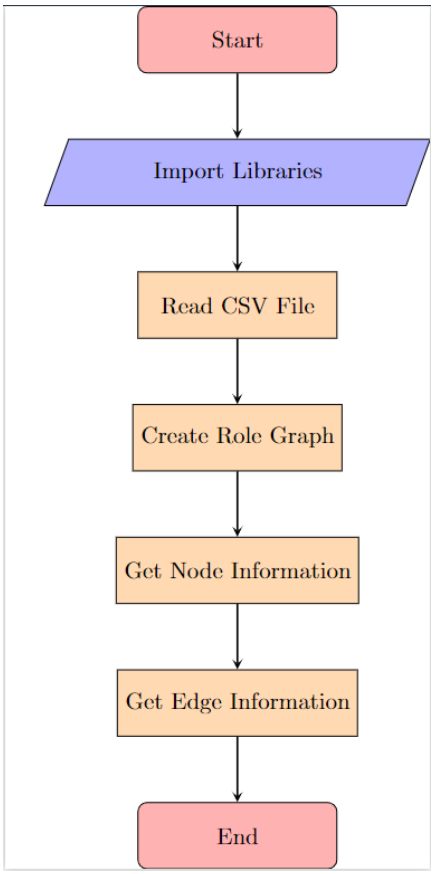
1 to 25 of 8879 entries Filter

member_x	Jc²	7	@profGbenga	A. Anil Sinaci	A. Cody Schuffelen	A. Jesse Jiryu Davis	A. Soroka	A.J. Roberts	AL Bot	APAR SINGHAL	Aanand Ramachandran	Aapo	Aaron Abramov	Aaron Ang	Aaron Bocko
Jc²	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
7	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
@profGbenga	0	0	0	0	1	0	0	0	0	0	0	0	0	0	
A. Anil Sinaci	0	0	0	0	0	0	1	0	0	0	0	0	0	0	
A. Cody Schuffelen	0	0	1	0	0	0	0	0	0	0	0	0	0	0	
A. Jesse Jiryu Davis	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
A.J. Roberts	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
AL Bot	0	0	0	0	0	0	0	0	0	1	0	0	0	0	
APAR SINGHAL	0	0	0	0	0	0	0	0	1	0	0	0	0	0	
Aanand Ramachandran	0	0	0	0	0	0	0	0	0	0	0	1	0	0	
Aapo	0	0	0	0	0	0	0	0	0	0	1	0	0	0	
Aaron Abramov	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
Aaron Ang	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
Aaron Bockover	0	0	0	0	0	0	0	0	1	1	0	0	0	0	
Aaron Bosley	0	0	0	0	0	0	0	0	0	0	1	1	0	0	
Aaron Chan	0	0	0	0	0	0	0	0	1	1	0	0	0	0	
Aaron Clauson	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
Aaron Crickenberger	0	0	1	0	1	0	0	0	0	0	0	0	0	0	
Aaron Cronin	0	0	0	0	0	0	0	0	1	1	0	0	0	0	
Aaron Fortner	0	0	0	0	0	0	0	0	1	1	0	0	0	0	
Aaron Goulet	1	0	0	0	0	0	0	0	0	0	0	0	0	0	
Aaron Gustafson	0	0	0	0	0	0	0	0	1	1	0	0	0	0	
Aaron Hallberg	0	0	0	0	0	0	0	0	1	1	0	0	0	0	
Aaron Marten	0	0	0	0	0	0	0	0	1	1	0	0	0	0	
Aaron Maxwell	0	0	0	0	0	0	0	0	0	0	1	1	0	0	

Experiment 2

Aim: Write a program to create node and edge lists network objects.

Flowchart:



Code:

```
import pandas as pd
import numpy as np
import networkx as nx #NetworkX provides classes for graphs
which allow multiple edges between any pair of nodes.
#for the creation, manipulation, and study of the structure,
dynamics, and functions of complex networks.
import matplotlib.pyplot as plt
%matplotlib inline
from operator import itemgetter #If multiple items are
specified, returns a tuple of lookup values. eg: itemgetter(1,
3, 5)('ABCDEFG') ->output:('B', 'D', 'F')
df = pd.read_csv ("/content/drive/MyDrive/organization.csv")

#Returns a graph from Pandas DataFrame containing an edge list
role_graph=
nx.from_pandas_edgelist(df,source="member",target="Organisatio
n")
```

```
type(role_graph)
```

```
role_graph.nodes()
```


```
len(role_graph.nodes())
```

8920

```
role_graph.edges()
```

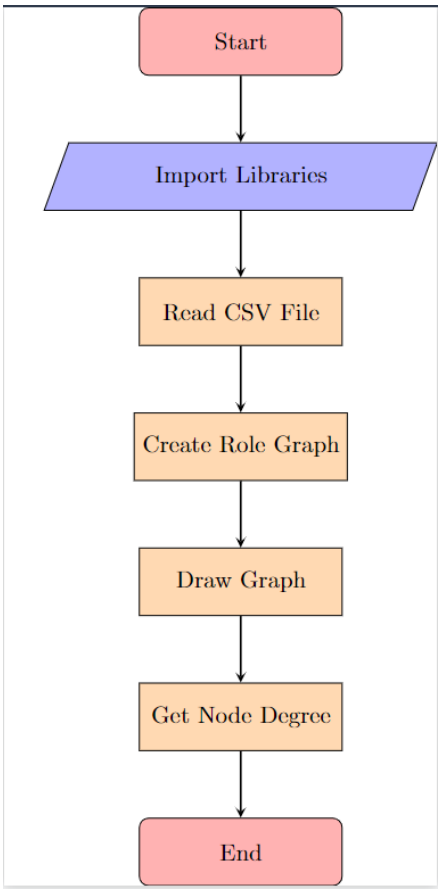

```
len(role_graph.edges())
```

9656

Experiment 3

Aim: Write a program to visualize a social network with matplotlib library.

Flowchart:



Code:

```
df = pd.read_csv("/content/drive/MyDrive/organization.csv")
nx.draw(role_graph)
#Draw the graph as a simple representation with no node labels
or edge labels and using the full Matplotlib figure area and
no axis labels by default



nx.degree(role_graph) #The node degree is the number of edges
adjacent to the node
```

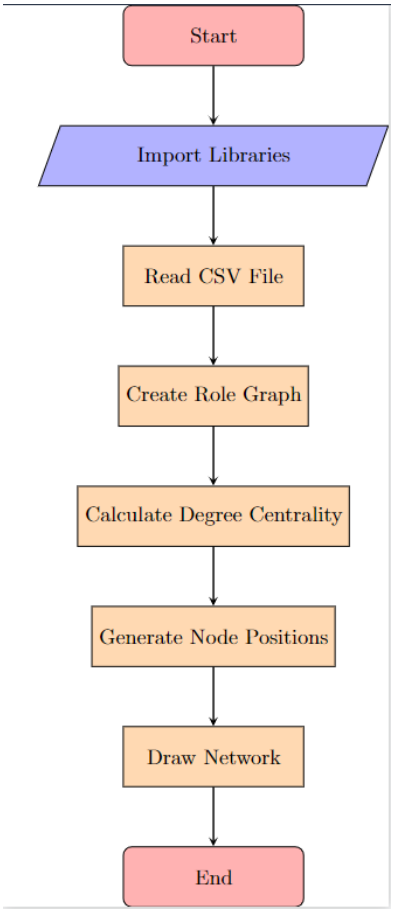
Output:

DegreeView({'Oleh Aldekein': 1, 'ethereum': 75, 'None': 27, 'Frank Szendzielarz': 1, 'Kolby Moroz Liebl': 1, 'Marius van der Wijden': 1, 'Mihai Alisie': 1, 'Anton Nashatyrev': 1, 'Justin Martin': 1, 'RJ Catalano': 1, 'Yoshitomo Nakanishi': 1, 'Alexander Arlt': 1, 'Alessandro Coglio': 1, 'acud': 1, 'Patricio Palladino': 1, 'Angela Lu': 1, 'Alex Beregszaszi': 1, 'becca': 1, 'Kamil Śliwak': 1, 'Cesar Brazon': 1, 'Christian Parpart': 1, 'Corwin Smith': 1, 'Marian OANCEA': 1, 'Darko Macesic': 1, 'Daniel Kirchner': 1, 'Ev': 1, 'franzihei': 1, 'Ferenc Szabo': 1, 'Grant Wuerker': 1, 'Gabriel Rocheleau': 1, 'Guillaume Ballet': 1, 'Andrei Maiboroda': 1, 'sacha': 1, 'Harikrishnan Mulackal': 1, 'Hugo': 1, 'David Disu': 1, 'Iuri Matias': 1, 'busyforking' 1, 'Jochem Brouwer': 1, 'Jamie Pitts': 1, 'Jutta Steiner': 1, 'Péter Szilágyi': 1, 'Karl Floersch': 1, 'kumavis': 1, 'Jakub Vysoký': 1, 'Leo': 2, 'ligi': 1, 'Maran': 1, 'Marc Garreau': 1, 'Mario Vega': 1, 'Matthew Di Ferrante': 1, 'Kevin Mai-Husan Chia': 1, 'Nicolás Quiroz': 1, 'Nicolas Fierro': 1, 'Samuel Furter': 1, 'Anton Evangelatov': 1, 'Jeffrey Wilcke': 1, 'Ognyan Genev': 1, 'Johnson Ogwuru': 1, 'Pablo Pettinari': 1, 'Piper Merriam': 1, 'Yoichi Hirai': 1, 'Alex Stokes': 1, 'Edward Ruchevits': 1, 'Sam Calder-Mason': 1, 'Rafael Matias': 1, 'Mário Havel': 1, 'tintin': 1, 'Jack Peterson': 1, 'Ulrich Petri': 1, 'Virgil Griffith': 1, 'Paul Wackerow': 1, 'Wesley': 1, 'Matthew Doty': 1, 'Andrew Ashikhmin': 1, 'Howard Huang': 2, 'pytorch': 48, 'Nicolas Hug': 1, 'Patrick Kan': 1, 'Aaron Shi': 1, 'Anthony Liu': 1, 'Asjad Syed': 2, 'Lucas Hosseini': 1, 'Jon Janzen': 2, 'Caroline Chen': 1, 'Sam Gross': 1, 'Tristan Rice': 1, 'Vasilis Vryniotis': 1, 'David Esiobu': 1, 'Dmytro Dzhulgakov': 1, 'Elizabeth Santorella': 1, 'Francisco Massa': 1, 'G. Jerry Chen': 1, 'Hanton Yang': 2, 'hl475': 2, 'Lu Fang': 1, 'Lin': 1, 'JIN SUN': 1, 'Jessica': 1, 'Jianyu Huang': 1, 'Jing Shan': 1, 'Jamie King': 1, 'Joel Pobar': 1, 'Sergii Dymchenko': 1, 'Karl Ostmo': 1, 'nateanl': 1, 'Naveed Golafshani': 1, 'Neeraj Pradhan': 1, 'Omkar Salpekar': 1, 'Pavithran Ramachandran': 1, 'Peng Wu': 1, 'Radhakrishnan Venkataramani': 1, 'Rohan Varma': 1, 'Peter Romov': 1, 'Shishir Bhat': 1, 'Eli Uriegas': 1, 'Stephen Macke': 1, 'Soumith Chintala': 2, 'Michael Suo': 1, 'Wei Wen': 1, 'Zafar': 1, 'Rui Zhu': 1, 'Zheng Yan': 1, 'Jacob Walker': 1, 'udacity': 5, 'Patrick': 1, 'Richard Kalehoff': 1,

Experiment 4

Aim: Write a program to calculate the node-level degree centrality measures for social network and plot the graph.

Flowchart:



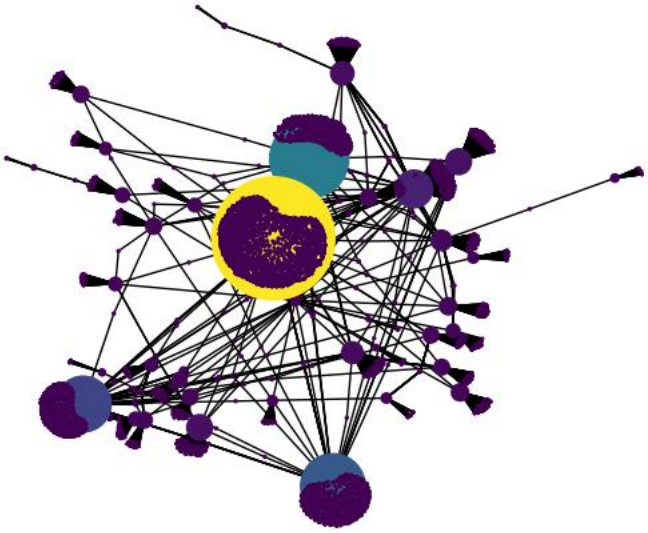
Code:

```
import pandas as pd
import numpy as np
import networkx as nx
import matplotlib.pyplot as plt
%matplotlib inline
from operator import itemgetter #suite of standard Python
operators
df=pd.read_csv("/content/drive/MyDrive/organization.csv")
nx.degree_centrality(role_graph) #The degree centrality for a
node v is the fraction of nodes it is connected to.
#G = nx.Graph([(0, 1), (0, 2), (0, 3), (1, 2), (1, 3)])
#nx.degree_centrality(G)
#{0: 1.0, 1: 1.0, 2: 0.6666666666666666, 3:
0.6666666666666666} 0 is connected with all 3 nodes so its
```

```
degree centrality is 1 while 2 is connected with 2 out of 3 so
66%

pos = nx.spring_layout(role_graph) #Position nodes using
Fruchterman-Reingold force-directed algorithm.

#The algorithm simulates a force-directed representation of
the network treating edges as springs holding nodes close
#while treating nodes as repelling objects, sometimes called
an anti-gravity force. Simulation continues until the
positions are close to an equilibrium.
degCent = nx.degree_centrality(role_graph)
node_color = [20000.0 * role_graph.degree(v) for v in
role_graph]
node_size =  [v * 10000 for v in degCent.values()]
plt.figure(figsize=(15,15))
nx.draw_networkx(role_graph, with_labels=False,
                 node_color=node_color,
                 node_size=node_size )
plt.axis('off')
sorted(degCent, key=degCent.get, reverse=True)[:5]
```

Output:

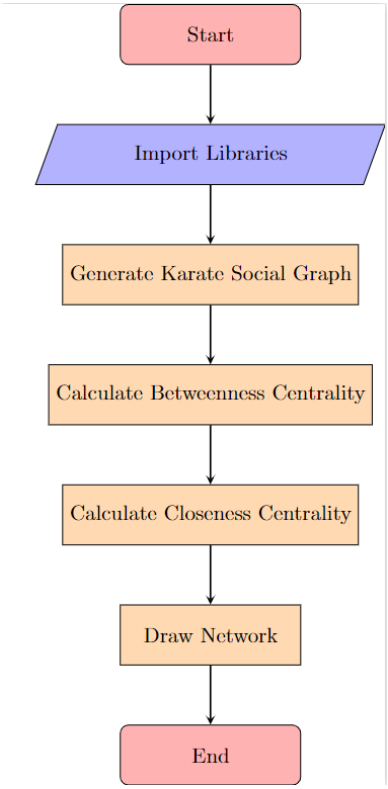
```
{'Oleh Aldekein': 0.00011208249271463797,
 'ethereum': 0.008406186953597848,
 'None': 0.003026227303295225,
 'Frank Szendzielarz': 0.00011208249271463797,
 'Kolby Moroz Liebl': 0.00011208249271463797,
 'Marius van der Wijden': 0.00011208249271463797,
 'Mihai Alisie': 0.00011208249271463797,
 'Anton Nashatyrev': 0.00011208249271463797,
 'Justin Martin': 0.00011208249271463797,
 'RJ Catalano': 0.00011208249271463797,
 'Yoshitomo Nakanishi': 0.00011208249271463797,
 'Alexander Arlt': 0.00011208249271463797,
 'Alessandro Coglio': 0.00011208249271463797,
 'acud': 0.00011208249271463797,
 'Patricio Palladino': 0.00011208249271463797,
 'Angela Lu': 0.00011208249271463797,
 'Alex Beregszaszi': 0.00011208249271463797,
 'becca': 0.00011208249271463797,
 'Kamil Śliwak': 0.00011208249271463797,
 'Cesar Brazon': 0.00011208249271463797,
 'Christian Parpart': 0.00011208249271463797,
 'Corwin Smith': 0.00011208249271463797,
 'Marian OANCΞA': 0.00011208249271463797,
 'Darko Macesic': 0.00011208249271463797,
 'Daniel Kirchner': 0.00011208249271463797,
 'Ev': 0.00011208249271463797,
 'franzihei': 0.00011208249271463797,
 'Ferenc Szabo': 0.00011208249271463797,
 'Grant Wuerker': 0.00011208249271463797,
 'Gabriel Rocheleau': 0.00011208249271463797,
 'Guillaume Ballet': 0.00011208249271463797,
 'Andrei Maiboroda': 0.00011208249271463797,
 'sacha': 0.00011208249271463797,
```



Experiment 5

Aim: Write a program to calculate the node-level betweenness centrality and closeness centrality measures for social network and plot the graph.

Flowchart:

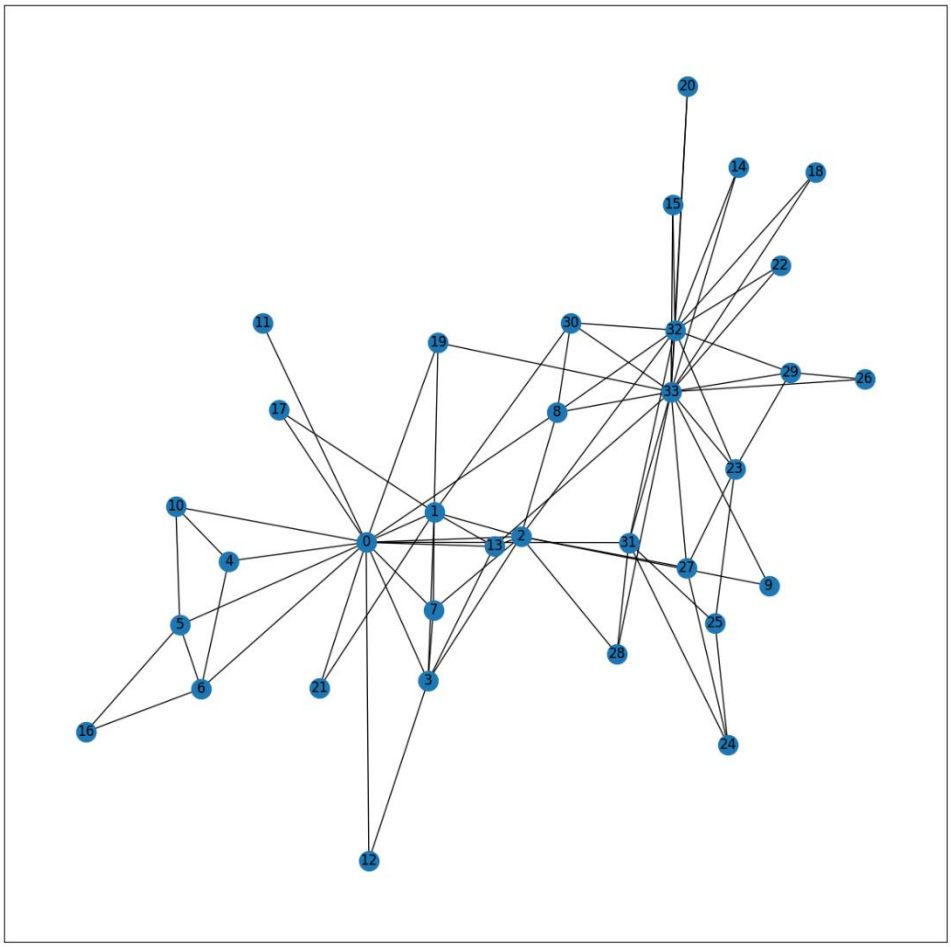


Code:

```
import random
import networkx as nx

# G is the Karate Social Graph
G = nx.karate_club_graph()

# betweenness_centrality
b=nx.betweenness_centrality(G)
print(b)

# closeness_centrality
c = nx.closeness_centrality(G)
print(c)

plt.figure(figsize =(15, 15))
nx.draw_networkx(G, with_labels = True)
```

Output:

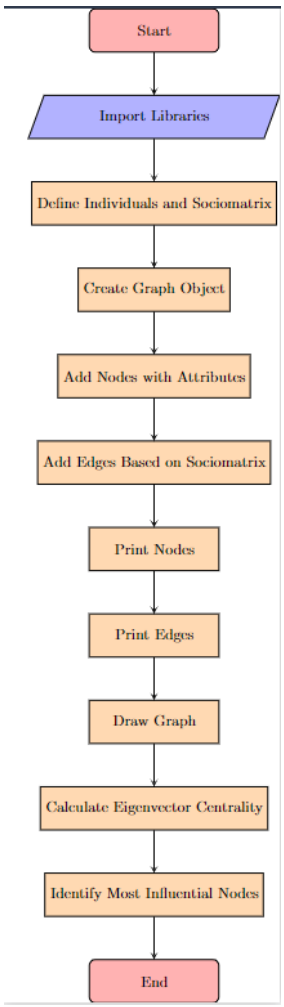

{0: 0.43763528138528146, 1: 0.05393668831168831, 2: 0.14365680615680618, 3: 0.011909271284271283, 4: 0.0006313131313131313, 5: 0.029987373737373737, 6: 0.029987373737373736, 7: 0.0, 8: 0.055926827801827801, 9: 0.0008477633477633478, 10: 0.000631313
{0: 0.5689655172413793, 1: 0.4852941176470588, 2: 0.5593220338983051, 3: 0.4647887323943662, 4: 0.3793103448275862, 5: 0.3837209302325581, 6: 0.3837209302325581, 7: 0.44, 8: 0.515625, 9: 0.4342105263157895, 10: 0.3793103448275862, 11: 0.3666666666


20
14
18
15
22
11
30
32
19
29
26
33
17
8
23
10
1
0
2
31
13
4
27
9
7
5
25
28
3
21
6
16
24
12

Experiment 6

Aim: Write a program to create network-level measures of centrality for social network.

Flowchart:



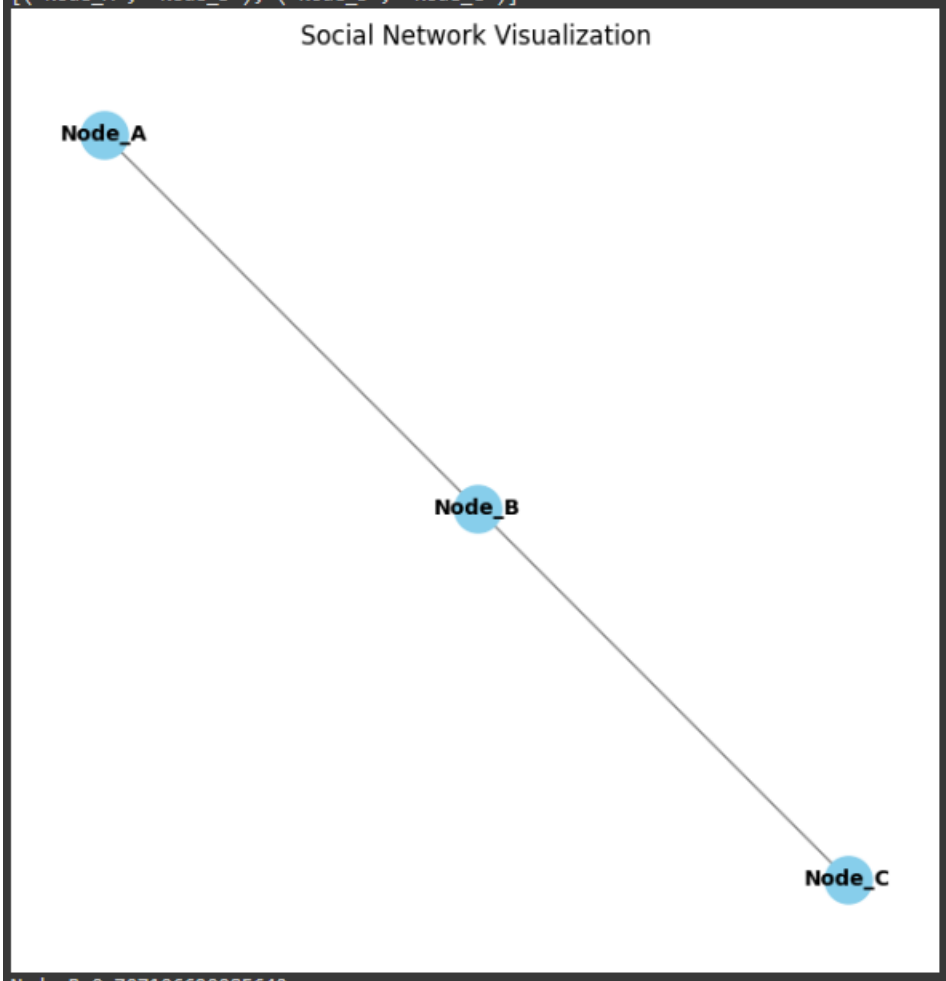
Code:

```
import numpy as np
import networkx as nx
import matplotlib.pyplot as plt

# Define the names of individuals
names = ['Node_A', 'Node_B', 'Node_C']

# Define the sociomatrix
sociomatrix = np.array([
#     A , B , C
    [0, 1, 0 ], # Node_A
    [1, 0, 1 ], # Node_B
```

```
    [0, 1, 0 ]  # Node_C
])

# Creating a Graph Object
G = nx.Graph()

# Adding Nodes with Attributes
for i, name in enumerate(names):
    G.add_node(name)

# Adding Edges Based on Sociomatrix
for i in range(len(names)):
    for j in range(i + 1, len(names)):
        if sociomatrix[i][j] == 1:
            G.add_edge(names[i], names[j])


# Print the nodes of the graph
print("Nodes of the social network:")
print(names)


# Print the edges of the graph
print("Edges of the social network:")
print(list(G.edges()))

# Drawing the Graph
pos = nx.spring_layout(G)  # Positions for all nodes
plt.figure(figsize=(6,6))
nx.draw(G, pos, with_labels=True, node_size=500,
node_color='skyblue',
        font_size=10, font_weight='bold', edge_color='gray')

# Displaying the Graph
plt.title('Social Network Visualization')
plt.show()

eigenvector_centrality = nx.eigenvector_centrality(G)
#Sort for identifying most inflential nodes using eigenvector
centrality
for node in sorted(eigenvector_centrality,
key=eigenvector_centrality.get, reverse=True):
  print(node, eigenvector_centrality[node])
```

Output:

```
Nodes of the social network:
['Node_A', 'Node_B', 'Node_C']
Edges of the social network:
[('Node_A', 'Node_B'), ('Node_B', 'Node_C')]
```

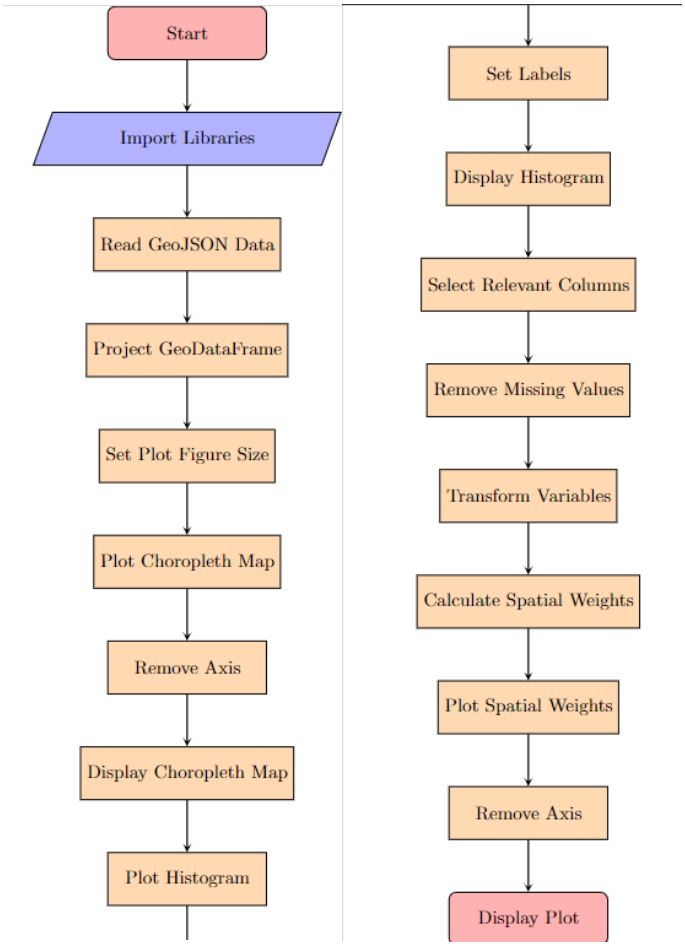



```
Node_B 0.707106690085642
Node_A 0.5000000644180599
Node_C 0.5000000644180599
```

Experiment 7

Aim: Write a program to incorporate and plot network connectivity in a spatial regression model.

Flowchart:



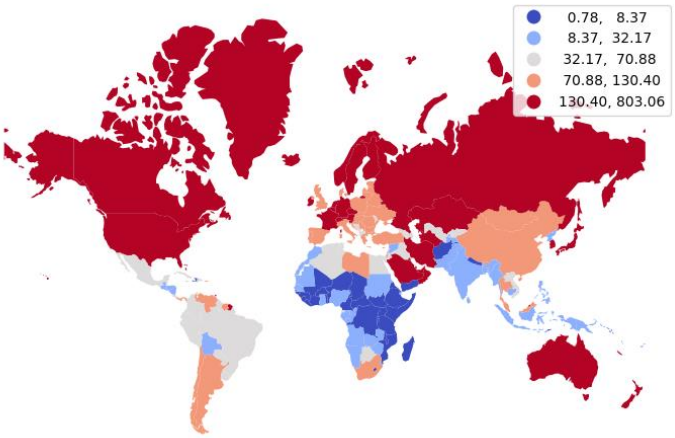
Code:

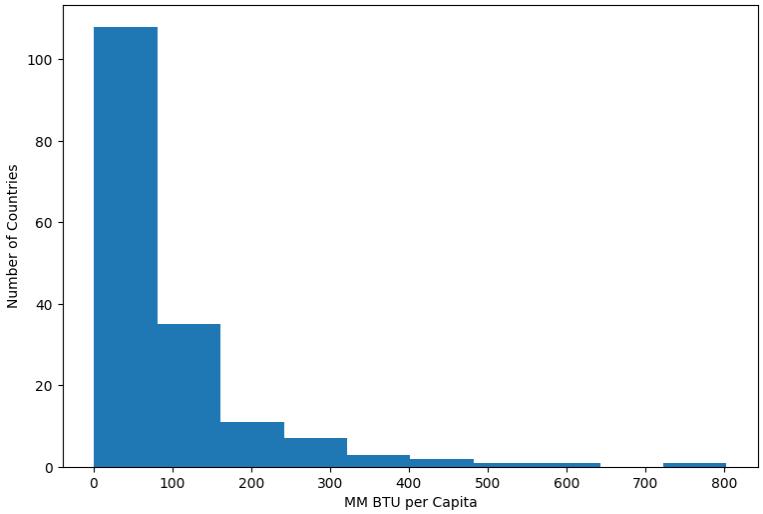
```
# Importing necessary libraries
import geopandas
import matplotlib.pyplot as plt
import numpy
import pysal.lib

# Reading GeoJSON data containing world energy indicators into
a GeoDataFrame
countries =
geopandas.read_file("https://michaelminn.net/tutorials/data/20
19-world-energy-indicators.geojson")

# Projecting the GeoDataFrame to Web Mercator (EPSG:3857)
projection
```

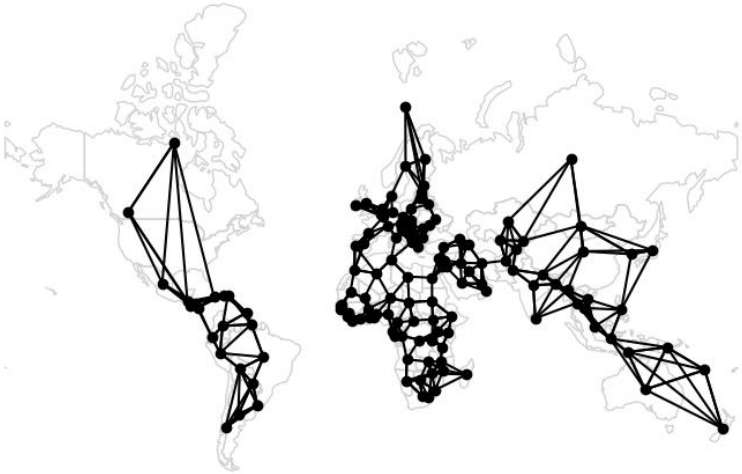
```
countries = countries.to_crs("EPSG:3857")

# Setting the size of the plot figure
plt.rcParams['figure.figsize'] = [9, 6]

# Plotting the choropleth map of MM_BTU_per_Capita using
GeoDataFrame plot function
axis = countries.plot("MM_BTU_per_Capita", cmap="coolwarm",
legend=True, scheme="quantiles")

# Removing axis
axis.set_axis_off()

# Displaying the plot
plt.show()

# Plotting a histogram of MM_BTU_per_Capita
axis = plt.hist(countries["MM_BTU_per_Capita"])

# Setting labels for x-axis and y-axis
plt.xlabel("MM BTU per Capita")
plt.ylabel("Number of Countries")

# Displaying the histogram plot
plt.show()

# Defining variables for modeling
dependent_name = ["Democracy_Index"]
independent_names = ["GDP_per_Capita_PPP_Dollars",
"Military_Percent_GDP", "Resource_Rent_Percent_GDP",
"Industry_Percent_GDP"]

# Selecting relevant columns from GeoDataFrame and removing
rows with missing values
model_data = countries[dependent_name + independent_names +
["geometry", "Latitude", "Longitude"]]
model_data = model_data.dropna()

# Transforming certain variables by taking their natural
logarithm plus one
transform_vars = ["GDP_per_Capita_PPP_Dollars",
"Military_Percent_GDP", "Resource_Rent_Percent_GDP"]
model_data[transform_vars] =
numpy.log(model_data[transform_vars] + 1)

# Calculating spatial weights using K-nearest neighbors (KNN)
method
weights = pysal.lib.weights.KNN.from_dataframe(model_data,
k=4)

# Plotting spatial weights on a map
```

```
axis = model_data.plot(edgecolor="lightgray",
facecolor="none")
model_data["index"] = model_data.index
weights.plot(gdf=model_data, indexed_on="index", ax=axis)

# Removing axis
axis.set_axis_off()

# Displaying the plot
plt.show()
```

Output:


0.78, 8.37
8.37, 32.17
32.17, 70.88
70.88, 130.40
130.40, 803.06

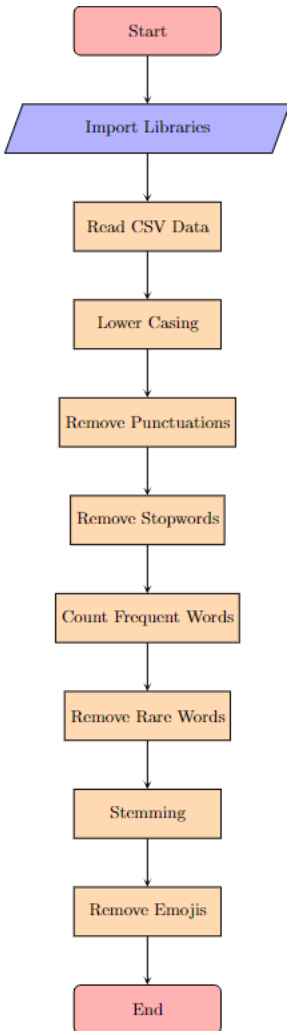

Number of Countries
MM BTU per Capita
0
20
40
60
80
100
0
100
200
300
400
500
600
700
800



Experiment 8

Aim: Write a program for preprocessing of social media text and detect sentiment from it.

Flowchart:



Code:

```
#TEXT PREPROCESSING
import numpy as np
import pandas as pd
import re
import nltk
import spacy
import string
pd.options.mode.chained_assignment = None

full_df = pd.read_csv("/content/drive/MyDrive/sample.csv")
df = full_df[["text"]]
df["text"] = df["text"].astype(str)
```

```
full_df.head()
```

	tweet_id	author_id	inbound	created_at	text	response_tweet_id	in_response_to_tweet_id
0	119237	105834	True	Wed Oct 11 06:55:44 +0000 2017	@AppleSupport causing the reply to be disregar...	119236	NaN
1	119238	ChaseSupport	False	Wed Oct 11 13:25:49 +0000 2017	@105835 Your business means a lot to us. Pleas...	NaN	119239.0
2	119239	105835	True	Wed Oct 11 13:00:09 +0000 2017	@76328 I really hope you all change but I'm su...	119238	NaN
3	119240	VirginTrains	False	Tue Oct 10 15:16:08 +0000 2017	@105836 LiveChat is online at the moment - htt...	119241	119242.0
4	119241	105836	True	Tue Oct 10 15:17:21 +0000 2017	@VirginTrains see attached error message. I've...	119243	119240.0

```
#Lower Casing
df["text_lower"] = df["text"].str.lower()
df.head()
```

	text	text_lower
0	@AppleSupport causing the reply to be disregar...	@applesupport causing the reply to be disregar...
1	@105835 Your business means a lot to us. Pleas...	@105835 your business means a lot to us. pleas...
2	@76328 I really hope you all change but I'm su...	@76328 i really hope you all change but i'm su...
3	@105836 LiveChat is online at the moment - htt...	@105836 livechat is online at the moment - htt...
4	@VirginTrains see attached error message. I've...	@virgintrains see attached error message. i've...

```
#Removal of Punctuations
#drop the new column created in last cell
df.drop(["text_lower"], axis=1, inplace=True)

PUNCT_TO_REMOVE = string.punctuation
def remove_punctuation(text):
    """custom function to remove the punctuation"""
    return text.translate(str.maketrans('', '',
PUNCT_TO_REMOVE))

df["text_wo_punct"] = df["text"].apply(lambda text:
remove_punctuation(text))
df.head()
```

	text	text_wo_punct
0	@AppleSupport causing the reply to be disregar...	AppleSupport causing the reply to be disregard...
1	@105835 Your business means a lot to us. Pleas...	105835 Your business means a lot to us Please ...
2	@76328 I really hope you all change but I'm su...	76328 I really hope you all change but Im sure...
3	@105836 LiveChat is online at the moment - htt...	105836 LiveChat is online at the moment https...
4	@VirginTrains see attached error message. I've...	VirginTrains see attached error message Ive tr...

```
#Removal of stopwords
#Stopwords are commonly occuring words in a language like
'the', 'a' and so on. They can be removed from the text most
of the times.
#import nltk
#nltk.download('stopwords')
from nltk.corpus import stopwords
", ".join(stopwords.words('english'))
```

```
'i, me, my, myself, we, our, ours, ourselves, you, you're, you've, you'll, you'd, your, yours, yourself, yourselves, he, him, his
what, which, who, whom, this, that, that'll, these, those, am, is, are, was, were, be, been, being, have, has, had, having, do, d
against, between, into, through, during, before, after, above, below, to, from, up, down, in, out, on, off, over, under, again, f
me, such, no, nor, not, only, own, same, so, than, too, very, s, t, can, will, just, don, don't, should, should've, now, d, ll, m
asn't, haven, haven't, isn, isn't, ma, mightn, mightn't, mustn, mustn't, needn, needn't, shan, shan't, shouldn, shouldn't, wasn,
```

```
STOPWORDS = set(stopwords.words('english'))
def remove_stopwords(text):
    """custom function to remove the stopwords"""
    return " ".join([word for word in str(text).split() if
word not in STOPWORDS])

df["text_wo_stop"] = df["text_wo_punct"].apply(lambda text:
remove_stopwords(text))
df.head()
```

	text	text_wo_punct	text_wo_stop
0	@AppleSupport causing the reply to be disregar...	AppleSupport causing the reply to be disregard...	AppleSupport causing reply disregarded tapped ...
1	@105835 Your business means a lot to us. Pleas...	105835 Your business means a lot to us Please ...	105835 Your business means lot us Please DM na...
2	@76328 I really hope you all change but I'm su...	76328 I really hope you all change but Im sure...	76328 I really hope change Im sure wont Becaus...
3	@105836 LiveChat is online at the moment - htt...	105836 LiveChat is online at the moment https...	105836 LiveChat online moment httpstcoSY94VtU8...
4	@VirginTrains see attached error message. I've...	VirginTrains see attached error message Ive tr...	VirginTrains see attached error message Ive tr...

```
#Removal of Frequent words
from collections import Counter
cnt = Counter()
for text in df["text_wo_stop"].values:
    for word in text.split():
        cnt[word] += 1

cnt.most_common(10)
```

```
[('I', 34),
 ('us', 25),
 ('DM', 19),
 ('help', 17),
 ('httpstcoGDrqU22YpT', 12),
 ('AppleSupport', 11),
 ('Thanks', 11),
 ('phone', 9),
 ('Hi', 8),
 ('get', 8)]
```

```
#Removal of Rare words
# Drop the two columns which are no more needed
df.drop(["text_wo_punct", "text_wo_stop"], axis=1,
inplace=True)

n_rare_words = 10
```

```
RAREWORDS = set([w for (w, wc) in cnt.most_common()[:-
n_rare_words-1:-1]])
def remove_rarewords(text):
    """custom function to remove the rare words"""
    return " ".join([word for word in str(text).split() if
word not in RAREWORDS])

df["text_wo_stopfreqrare"] =
df["text_wo_stopfreq"].apply(lambda text:
remove_rarewords(text))
df.head()
```

	text	text_wo_stopfreq	text_wo_stopfreqrare
0	@AppleSupport causing the reply to be disregar...	causing reply disregarded tapped notification ...	causing reply disregarded tapped notification ...
1	@105835 Your business means a lot to us. Pleas...	105835 Your business means lot Please name zip...	105835 Your business means lot Please name zip...
2	@76328 I really hope you all change but I'm su...	76328 really hope change Im sure wont Because ...	76328 really hope change Im sure wont Because ...
3	@105836 LiveChat is online at the moment - htt...	105836 LiveChat online moment httpstcoSY94VtU8...	105836 LiveChat online moment httpstcoSY94VtU8...
4	@VirginTrains see attached error message. I've...	VirginTrains see attached error message Ive tr...	VirginTrains see attached error message Ive tr...

```
#Stemming
#Stemming is the process of reducing inflected (or sometimes
derived) words to their word stem, base or root form
#for example, if there are two words in the corpus walks and
walking, then stemming will stem the suffix to make them walk
from nltk.stem.porter import PorterStemmer

# Drop the two columns
df.drop(["text_wo_stopfreq", "text_wo_stopfreqrare"], axis=1,
inplace=True)

stemmer = PorterStemmer()
def stem_words(text):
    return " ".join([stemmer.stem(word) for word in
text.split()])

df["text_stemmed"] = df["text"].apply(lambda text:
stem_words(text))
df.head()
```

	text	text_stemmed
0	@AppleSupport causing the reply to be disregar...	@applesupport caus the repli to be disregard a...
1	@105835 Your business means a lot to us. Pleas...	@105835 your busi mean a lot to us. pleas dm y...
2	@76328 I really hope you all change but I'm su...	@76328 i realli hope you all chang but i'm sur...
3	@105836 LiveChat is online at the moment - htt...	@105836 livechat is onlin at the moment - http...
4	@VirginTrains see attached error message. I've...	@virgintrain see attach error message. i'v tri...

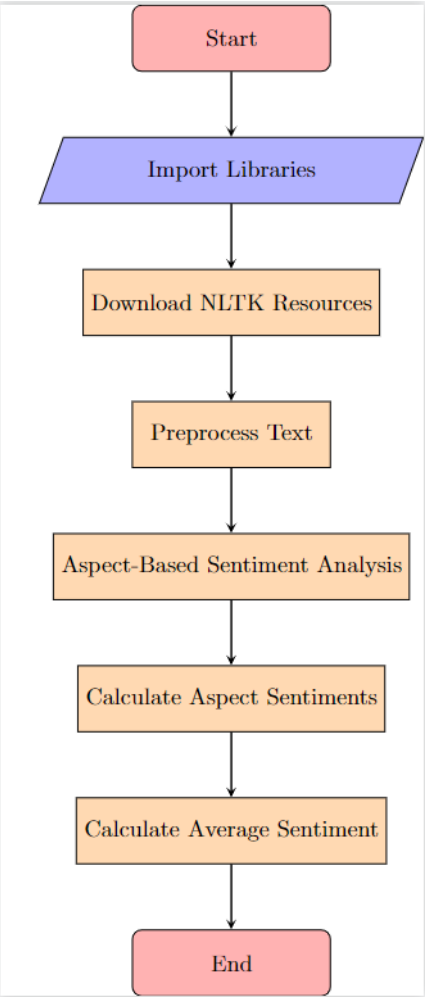
```
#Removal of Emojis
def remove_emoji(string):
    emoji_pattern = re.compile("["
                               u"\U0001F600-\U0001F64F"  #
emoticons
                               u"\U0001F300-\U0001F5FF"  # symbols
& pictographs
                               u"\U0001F680-\U0001F6FF"  #
transport & map symbols
                               u"\U0001F1E0-\U0001F1FF"  # flags
(iOS)
                               u"\U00002702-\U000027B0"
                               u"\U000024C2-\U0001F251"
                               "]+", flags=re.UNICODE)
    return emoji_pattern.sub(r'', string)

remove_emoji("game is on 🔥🔥")
```

```
'game is on '
```

Experiment 9

Aim: Write a program for Aspect-based sentiment analysis of social media text.

Flowchart:



Code:

```
import nltk
from textblob import TextBlob
from nltk.tokenize import word_tokenize, sent_tokenize
from nltk.corpus import stopwords
from collections import defaultdict

# Download necessary NLTK resources
nltk.download('punkt')
nltk.download('stopwords')

def preprocess_text(text):
    # Tokenize text into sentences
```

```
    sentences = sent_tokenize(text)

    # Tokenize each sentence into words, remove stopwords and
punctuation
    stop_words = set(stopwords.words('english'))
    preprocessed_sentences = []
    for sentence in sentences:
        words = word_tokenize(sentence)
        words = [word.lower() for word in words if
word.isalnum() and word.lower() not in stop_words]
        preprocessed_sentences.append(words)

    return preprocessed_sentences

def aspect_based_sentiment_analysis(text):
    preprocessed_text = preprocess_text(text)
    print(preprocessed_text)
    aspect_sentiments = defaultdict(list)

    for sentence in preprocessed_text:
        # Convert list of words back to sentence
        sentence = " ".join(sentence)
        blob = TextBlob(sentence)

        # Extract noun phrases as aspects
        noun_phrases = blob.noun_phrases

        # Analyze sentiment for each aspect
        for aspect in noun_phrases:

aspect_sentiments[aspect].append(blob.sentiment.polarity)

    # Calculate average sentiment for each aspect
    aspect_average_sentiments = {}
    for aspect, sentiments in aspect_sentiments.items():
        aspect_average_sentiments[aspect] = sum(sentiments) /
len(sentiments)

    return aspect_average_sentiments

# Example usage
social_media_text = """
The camera quality of this phone is amazing. But battery life
is poor.
I love the design of the new laptop. The performance is also
great!
"""
aspect_sentiments =
aspect_based_sentiment_analysis(social_media_text)
for aspect, sentiment in aspect_sentiments.items():
    print(f"Aspect: {aspect}, Sentiment: {sentiment}")
```

Output:

```
[['camera', 'quality', 'phone', 'amazing'], ['battery', 'life', 'poor'], ['love', 'design', 'new', 'laptop'], ['performance', 'also', 'great']]
Aspect: camera quality phone, Sentiment: 0.6000000000000001
Aspect: battery life, Sentiment: -0.4
Aspect: new laptop, Sentiment: 0.3181818181818182
```