INDEX

BCA 507(C):- PRACTILE ON DATA MINING USING PYTHON

SR.NO	TITLE	REMARK	SIGN
1.	Calculate the mean and standard deviation.		
2.	Read the CSV file.		
3.	Perform data filtering and calculate aggregate statistics.		
4.	Calculate total sales by month.		
5.	Implement the Clustering using K-means.		
6.	Classification using Random Forcst.		
7.	Regression Analysis using Linear Regression.		
8.	Association Rule Mining using Apriori.		
9.	Visualize the result of the clustering and compare.		
10.	Visualize the correlation matrix using a pseudocolor plot.		
11.	Use of degrees distribtution of a network.		
12.	Graph visualization of a network using maximum , minimum , median , first quartile and third quartile.		

1.Calculate the mean and standard deviation.

```
import numpy as np

data=np.array([10,20,30,40,50])

mean_value=np.mean(data)

std_dev=np.std(data,ddof=1)

print(f"mean:{mean_value}")

print(f"Standard Deviation:{std_dev}")
```

Output:-

mean:30.0

Standard Deviation:15.811388300841896

2.Read the CSV file.

#create csv file:-

```
import pandas as pd
data = {
    'Name': ['Alice', 'Bob', 'Charlie'],
    'Age': [30, 25, 35],
    'City': ['New York', 'Los Angeles', 'Chicago']
}
df = pd.DataFrame(data)
df.to_csv('people.csv', index=False)
print("CSV file created successfully.")
```

Output:-

CSV file created successfully.

#Read csv file:-

```
import pandas as pd
file_path = 'people.csv'
df = pd.read_csv(file_path)
df.head()  # Shows the first 5 rows of the data
```

Output:-

	Name	Age	City
0	Alice	30	New York
1	Bob	25	Los Angeles
2	Charlie	35	Chicago

3.Perform data filtering and calculate aggregate statistics.

```
import pandas as pd
data={
    'name':[ 'alice', 'bob', 'charlie','david','eve'],
    'age':[20,22,32,21,19],
    'salary':[3000,4000,2000,5000,3500]
}
df=pd.DataFrame(data)
f_d=df[df['age']>20]
ave_sal=f_d['salary'].mean()
ave_sal1=f_d['salary'].sum()
print(f_d)
print('____________________________________________________________________________
______________________________________________________________________________
_____________')
print(f'average salary of employees older than 25:(ave_sal,ave_sal1)')
```

Output:-

```
      name  age  salary
1      bob   22    4000
2  charlie   32    2000
3    david   21    5000
__________________________________________________________________________________
average salary of employees older than 25:(ave_sal,ave_sal1)
```

4.Calculate total sales by month.

```
import pandas as pd
data = {
    'Date': ['2023-01-05', '2023-01-12', '2023-02-01','2023-02-14'],
    'Sales': [100, 150,200,300]
}
df = pd.DataFrame(data)
df['Date'] = pd.to_datetime(df['Date'])
df['Year-Month'] = df['Date'].dt.to_period('M')
total_sales_by_month = df.groupby('Year-Month')['Sales'].sum().reset_index()
print(total_sales_by_month)
```

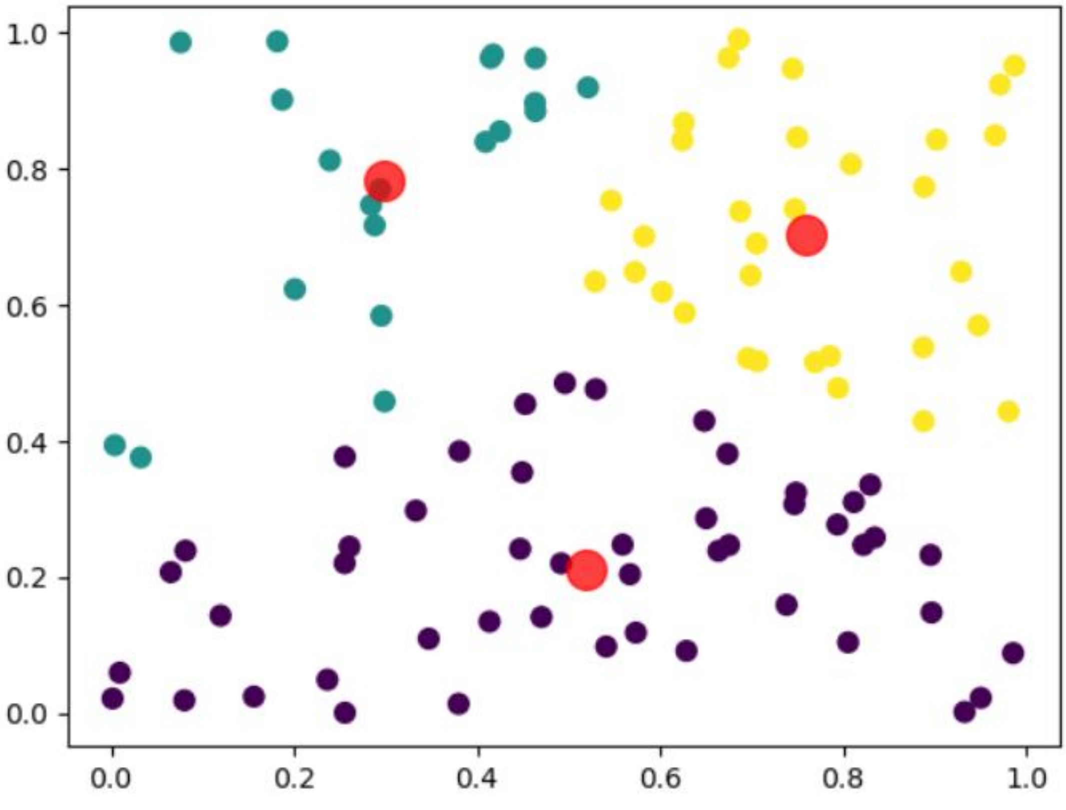
Output:-

```
  Year-Month  Sales
0    2023-01    250
1    2023-02    500
```

5.Implement the Clustering using K-means.

```
import numpy as np
import matplotlib.pyplot as plt
from sklearn.cluster import KMeans
np.random.rand(0)
x=np.random.rand(100,2)
kmeans=KMeans(n_clusters=3)
kmeans.fit(x)
centers=kmeans.cluster_centers_
labels=kmeans.labels_
plt.scatter(x[:,0],x [:,1],c=labels,s=50,cmap='viridis')
plt.scatter(centers[:,0],centers[:,1],c='red',s=200,alpha=0.75)
plt.show()
```

Output:-



6.Classification using Random Forcst.

```
import numpy as np

import pandas as pd

from sklearn.model_selection import train_test_split

from sklearn.ensemble import RandomForestClassifier

from sklearn.metrics import classification_report, accuracy_score

data = {

    'Age': [25, 45, 35, 33, 50, 23, 43, 36, 29, 48],

    'Income': [40000, 80000, 60000, 58000, 90000, 42000, 75000, 62000, 50000, 
85000],

    'Education': [1, 2, 1, 0, 2, 1, 2, 0, 1, 2],

    'Purchased': [0, 1, 0, 0, 1, 0, 1, 0, 0, 1]

}

df = pd.DataFrame(data)

X = df[['Age', 'Income', 'Education']]

y = df['Purchased']

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, 
random_state=42)

rf_model = RandomForestClassifier(n_estimators=100, random_state=42)

rf_model.fit(X_train, y_train)

y_pred = rf_model.predict(X_test)

print("Accuracy:", accuracy_score(y_test, y_pred))

print("\nClassification Report:")

print(classification_report(y_test, y_pred))

feature_importances = pd.DataFrame({

    'Feature': X.columns,

    'Importance': rf_model.feature_importances_

}).sort_values(by='Importance', ascending=False)

print("\nFeature Importances:")
```

print(feature_importances)

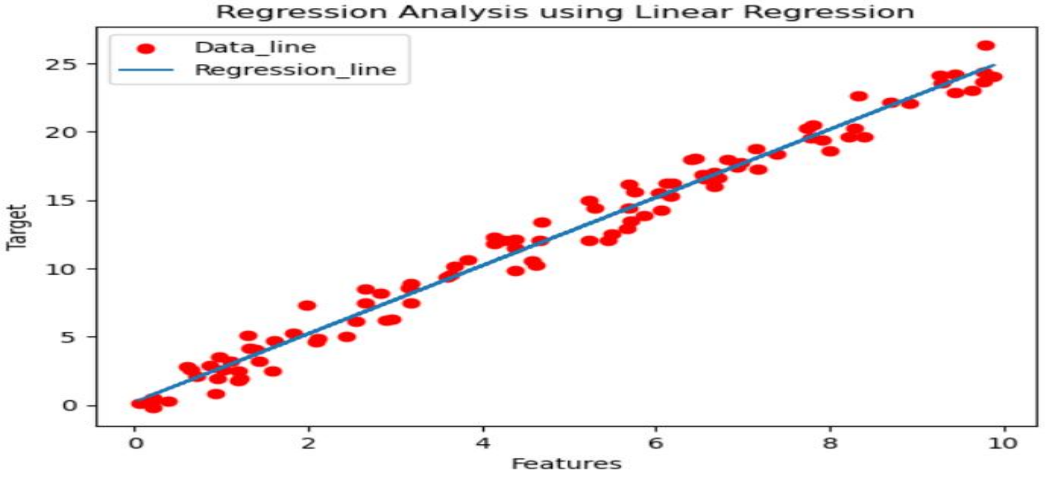
Output:-

```
Accuracy: 1.0

Classification Report:
              precision    recall  f1-score   support

           0       1.00      1.00      1.00         2
           1       1.00      1.00      1.00         1

    accuracy                           1.00         3
   macro avg       1.00      1.00      1.00         3
weighted avg       1.00      1.00      1.00         3


Feature Importances:
     Feature  Importance
1     Income    0.363636
0        Age    0.353535
2  Education    0.282828
```

7.Regression Analysis using Linear Regression.

```
import numpy as np

import pandas as pd

import matplotlib.pyplot as plt

from sklearn.linear_model import LinearRegression

np.random.seed(0)

x=np.random.rand(100,1)*10

y=2.5*x+np.random.randn(100,1)

data=pd.DataFrame(np.hstack((x,y)),

columns=['Feature','Target'])

model=LinearRegression()

model.fit(data[['Feature']],data[['Target']])

y_pred=model.predict(data[['Feature']])

plt.scatter(data['Feature'], data['Target'],color='red', label='Data_line')

plt.plot(data['Feature'],y_pred,label='Regression_line')

plt.xlabel('Features')

plt.ylabel('Target')

plt.title('Regression Analysis using Linear Regression')

plt.legend()

plt.show()
```

Output:-



8.Association Rule Mining using Apriori.

```
import pandas as pd
from mlxtend.preprocessing import TransactionEncoder
from mlxtend.frequent_patterns import apriori
from mlxtend.frequent_patterns import association_rules
d_S=[['Milk','Bread'],
    ['Bread','Beer','Eggs'],
    ['Milk','Beer','Cola'],
    ['Bread','Milk','cola']]
encoder=TransactionEncoder()
onehot=encoder.fit(d_S).transform(d_S)
df=pd.DataFrame(onehot,columns=encoder.columns_)
f_i=apriori(df,min_support=0.4,use_colnames=True)
print(f_i)
rules = association_rules(f_i, num_itemsets=len(f_i), metric="lift", min_threshold=1)
print("\nAssociation Rules:")
print(rules)
```

Output:-

```
   support       itemsets
0     0.50         (Beer)
1     0.75        (Bread)
2     0.75         (Milk)
3     0.50  (Bread, Milk)

Association Rules:
Empty DataFrame
Columns: [antecedents, consequents, antecedent support, consequent support, support, confidence, lift, representativity, leverage, conviction, zhangs_met
ric, jaccard, certainty, kulczynski]
Index: []
```

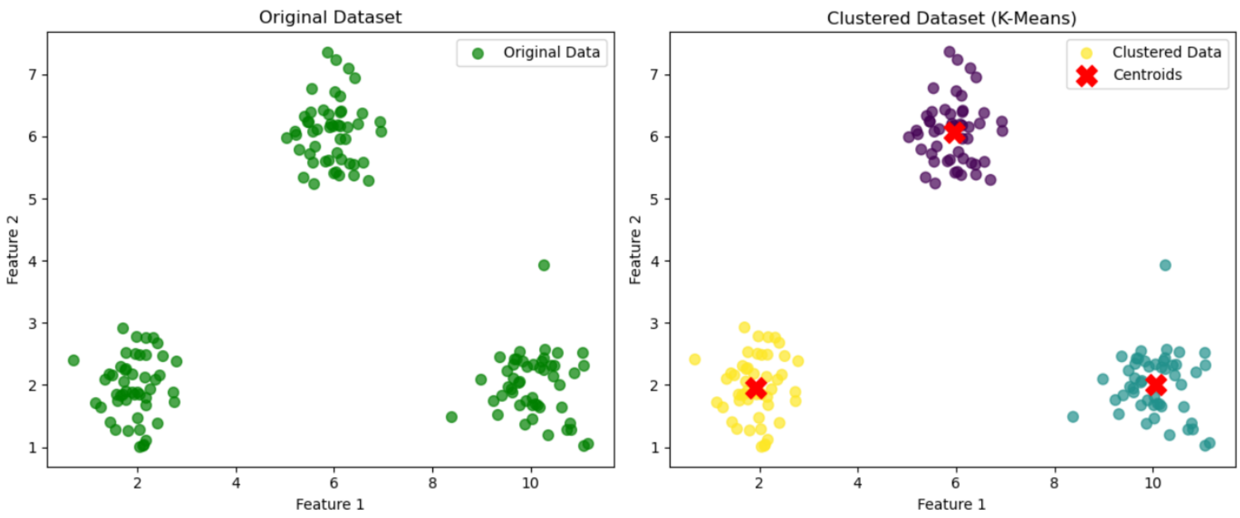
9.Visualize the result of the clustering and compare.

```
import numpy as np

import matplotlib.pyplot as plt

from sklearn.cluster import KMeans

np.random.seed(42)

X1 = np.random.normal(loc=[2, 2], scale=0.5, size=(50, 2))

X2 = np.random.normal(loc=[6, 6], scale=0.5, size=(50, 2))

X3 = np.random.normal(loc=[10, 2], scale=0.5, size=(50, 2))

X = np.vstack((X1, X2, X3))  # Combine into a single dataset

kmeans = KMeans(n_clusters=3, random_state=42)

kmeans.fit(X)

labels = kmeans.labels_

centroids = kmeans.cluster_centers_

plt.figure(figsize=(12, 5))

plt.subplot(1, 2, 1)

plt.scatter(X[:, 0], X[:, 1], c='green', s=50, alpha=0.7, label="Original Data")

plt.title("Original Dataset")

plt.xlabel("Feature 1")

plt.ylabel("Feature 2")

plt.legend()

plt.subplot(1, 2, 2)

plt.scatter(X[:, 0], X[:, 1], c=labels, cmap='viridis', s=50, alpha=0.7, label="Clustered
Data")

plt.scatter(centroids[:, 0], centroids[:, 1], c='red', s=200, marker='X',
label="Centroids")

plt.title("Clustered Dataset (K-Means)")

plt.xlabel("Feature 1")

plt.ylabel("Feature 2")

plt.legend()
```

plt.tight_layout()

plt.show()

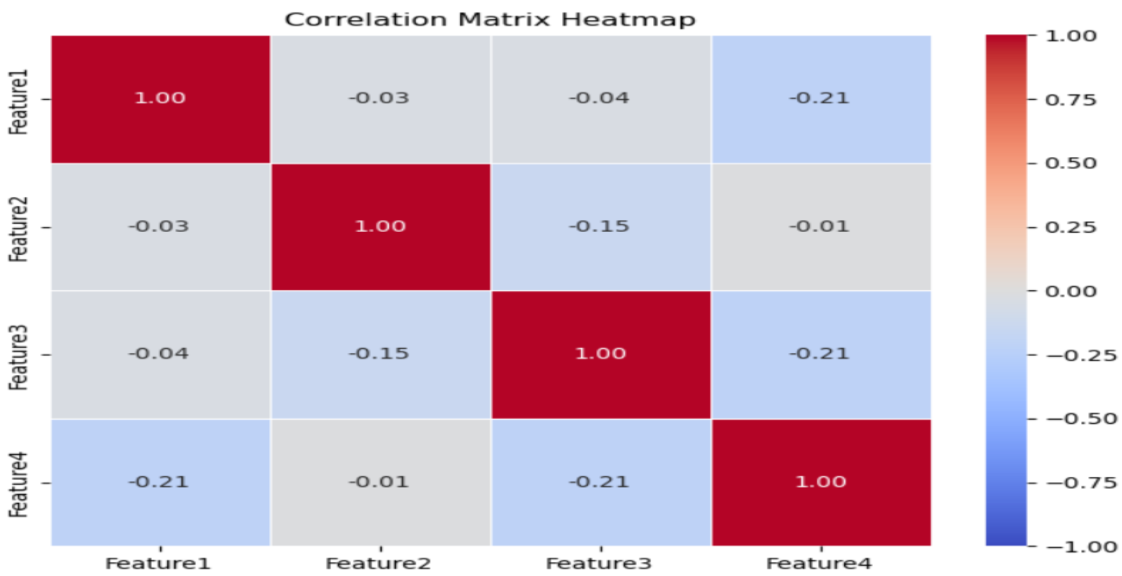
Output:-



10.Visualize the correlation matrix using a pseudocolor plot.

```
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
np.random.seed(42)
data = {
    'Feature1': np.random.rand(100),
    'Feature2': np.random.rand(100),
    'Feature3': np.random.rand(100),
    'Feature4': np.random.rand(100)
}
df = pd.DataFrame(data)
corr_matrix = df.corr()
plt.figure(figsize=(8, 6))
sns.heatmap(corr_matrix, annot=True, cmap='coolwarm', vmin=-1, vmax=1, fmt='.2f',
linewidths=0.5)
plt.title('Correlation Matrix Heatmap')
plt.show()
```

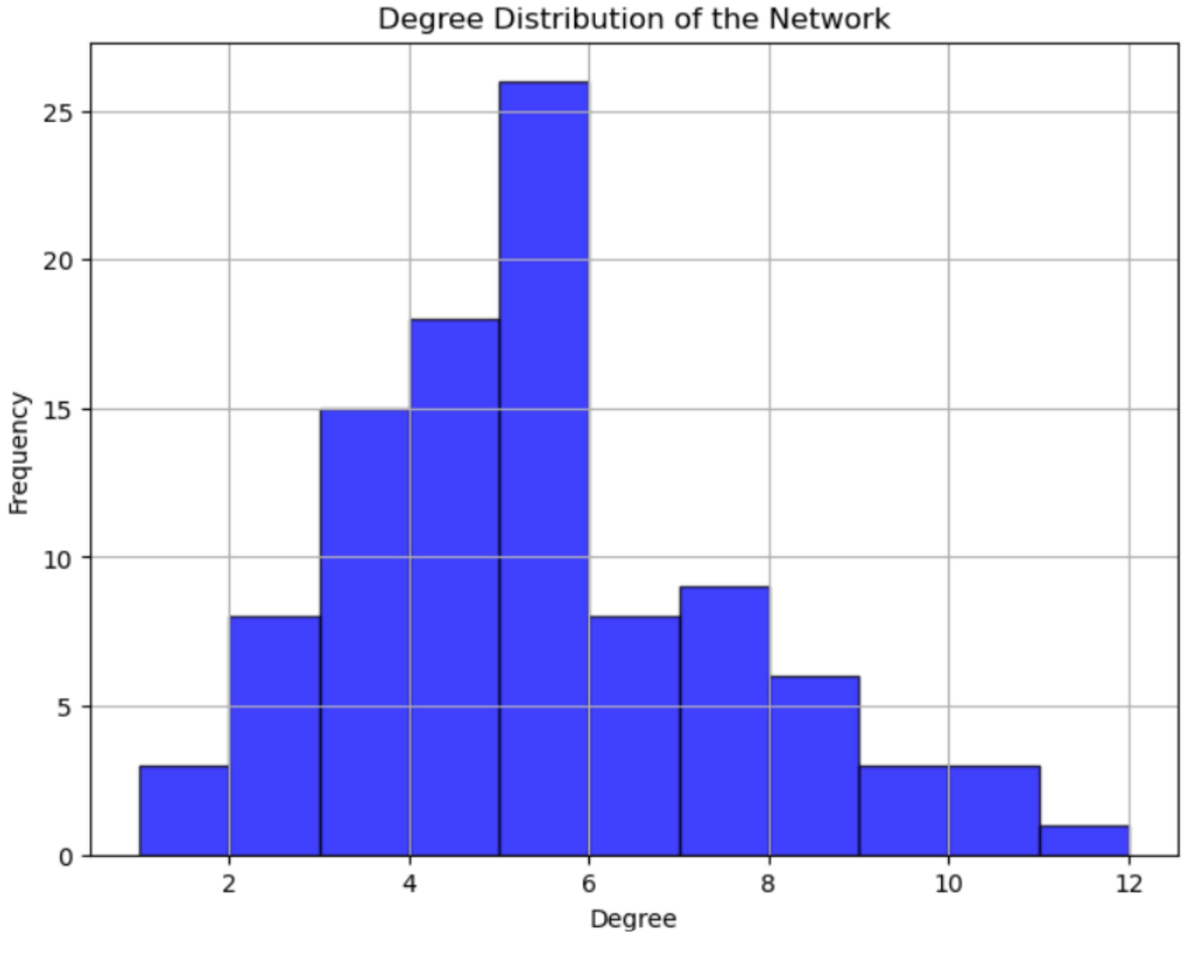
Output:-



11.Use of degrees distribtution of a network.

```
import networkx as nx

import matplotlib.pyplot as plt

G = nx.erdos_renyi_graph(n=100, p=0.05)

degree_sequence = [G.degree(n) for n in G.nodes()]

plt.figure(figsize=(8, 6))

plt.hist(degree_sequence, bins=range(min(degree_sequence), max(degree_sequence)
+ 1), alpha=0.75, color='blue', edgecolor='black')

plt.title("Degree Distribution of the Network")

plt.xlabel("Degree")

plt.ylabel("Frequency")

plt.grid(True)

plt.show()
```

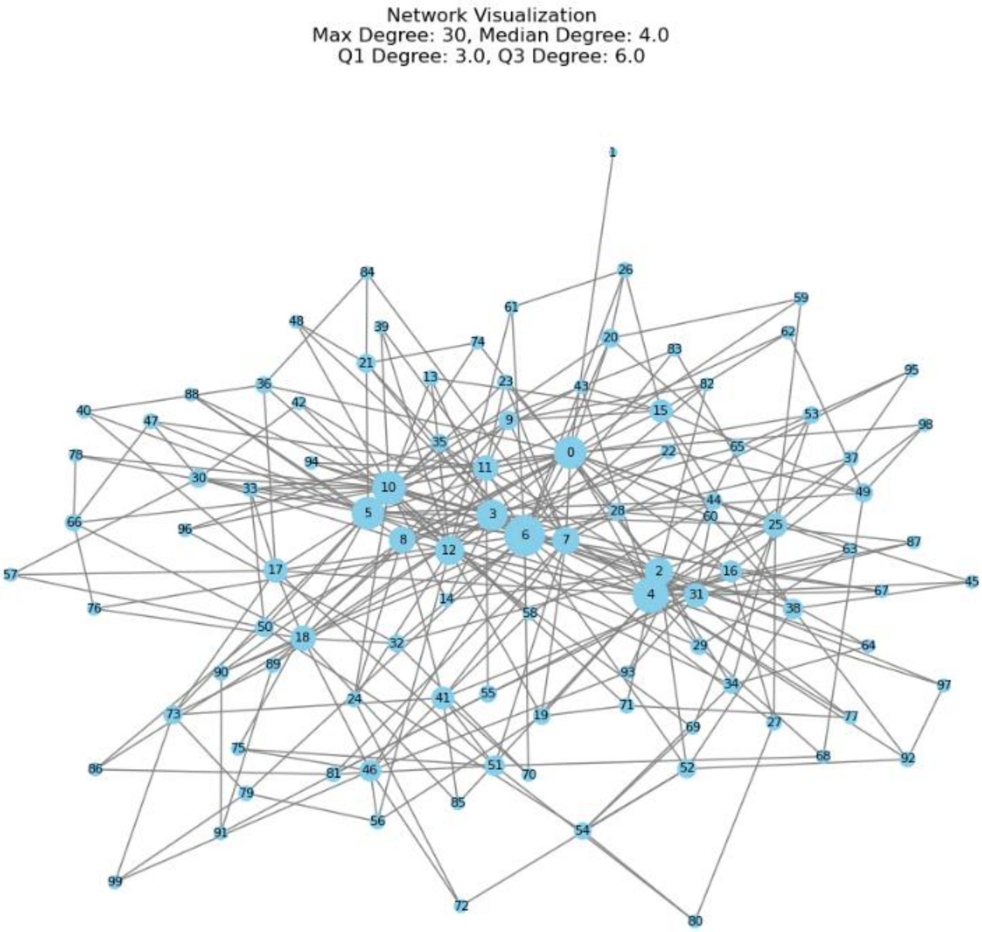
Output:-



12.Graph visualization of a network using maximum , minimum , median , first quartile and third quartile.

```
import networkx as nx

import matplotlib.pyplot as plt

import numpy as np

G = nx.barabasi_albert_graph(n=100, m=3)

degree_sequence = [G.degree(n) for n in G.nodes()]

max_degree = max(degree_sequence)

median_degree = np.median(degree_sequence)

q1_degree = np.percentile(degree_sequence, 25)

q3_degree = np.percentile(degree_sequence, 75)

node_size = [degree * 20 for degree in degree_sequence]

plt.figure(figsize=(10, 8))

pos = nx.spring_layout(G)

nx.draw(G, pos, node_size=node_size, with_labels=True, node_color='skyblue',
edge_color='gray', font_size=8)

plt.title(f"Network Visualization\nMax Degree: {max_degree}, Median Degree:
{median_degree}\n"

          f"Q1 Degree: {q1_degree}, Q3 Degree: {q3_degree}", fontsize=12)

plt.show()
```

Output:-


Network Visualization
Max Degree: 30, Median Degree: 4.0
Q1 Degree: 3.0, Q3 Degree: 6.0