1:

(A). Collect any dataset without decision attribute. The dataset may be collected from UCI machine learning repository. If the dataset contains the decision attribute then remove it as you will perform clustering of objects in the dataset.

In [30]:

```
import plotly.express as px
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
%matplotlib inline
import seaborn as sns
import warnings
warnings.filterwarnings("ignore")
import os
os.chdir(r'D:\Datasets\Dataset-SC')
```

In [31]:

```
df=pd.read_csv("iris.data")
df0 = df
```

In [32]:

```
df.head()
```

Out[32]:

	5.1	3.5	1.4	0.2	Iris-setosa
0	4.9	3.0	1.4	0.2	Iris-setosa
1	4.7	3.2	1.3	0.2	Iris-setosa
2	4.6	3.1	1.5	0.2	Iris-setosa
3	5.0	3.6	1.4	0.2	Iris-setosa
4	5.4	3.9	1.7	0.4	Iris-setosa

In [33]:

```
df.shape
```

Out[33]:

```
(149, 5)
```

In [34]:

```
df.columns
```

Out[34]:

```
Index(['5.1', '3.5', '1.4', '0.2', 'Iris-setosa'], dtype='object')
```

In [35]:

```
df.rename(columns={'5.1':'sepal_length', '3.5':'sepal_width', '1.4':'petal_length', '0.2
':'petal_width', 'Iris-setosa':'species'}, inplace=True)
```

In [36]:

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 149 entries, 0 to 148
Data columns (total 5 columns):
 #   Column        Non-Null Count  Dtype  
---  ------        --------------  -----  
 0   sepal_length  149 non-null    float64
 1   sepal_width   149 non-null    float64
 2   petal_length  149 non-null    float64
 3   petal_width   149 non-null    float64
 4   species       149 non-null    object 
dtypes: float64(4), object(1)
memory usage: 5.9+ KB
```

In [37]:

```
df.drop(columns="species", axis=0 , inplace=True)
df.head()
```

Out[37]:

	sepal_length	sepal_width	petal_length	petal_width
0	4.9	3.0	1.4	0.2
1	4.7	3.2	1.3	0.2
2	4.6	3.1	1.5	0.2
3	5.0	3.6	1.4	0.2
4	5.4	3.9	1.7	0.4

(B). Let the dataset has m rows and n columns where, each row is an object and each column is an attribute or feature of the object in the dataset. So you can consider the dataset as an m×n matrix.

In [38]:

```
df.describe()
```

Out[38]:

	sepal_length	sepal_width	petal_length	petal_width
count	149.000000	149.000000	149.000000	149.000000
mean	5.848322	3.051007	3.774497	1.205369
std	0.828594	0.433499	1.759651	0.761292
min	4.300000	2.000000	1.000000	0.100000
25%	5.100000	2.800000	1.600000	0.300000
50%	5.800000	3.000000	4.400000	1.300000
75%	6.400000	3.300000	5.100000	1.800000
max	7.900000	4.400000	6.900000	2.500000

(C). Normalize the attribute values within the range [0,1] using any normalization technique to give all the attributes an equal importance.

In [39]:

```
from sklearn.preprocessing import MinMaxScaler
```

In [40]:

```
scaler = MinMaxScaler()
```

In [41]:

```
scaler.fit(df)
```

Out[41]:

```
▾ MinMaxScaler
MinMaxScaler()
```

In [42]:

```
df2 = scaler.transform(df)
```

In [43]:

```
df2
```

Out[43]:

```
array([[0.16666667, 0.41666667, 0.06779661, 0.04166667],
       [0.11111111, 0.5       , 0.05084746, 0.04166667],
       [0.08333333, 0.45833333, 0.08474576, 0.04166667],
       [0.19444444, 0.66666667, 0.06779661, 0.04166667],
       [0.30555556, 0.79166667, 0.11864407, 0.125     ],
       [0.08333333, 0.58333333, 0.06779661, 0.08333333],
       [0.19444444, 0.58333333, 0.08474576, 0.04166667],
       [0.02777778, 0.375     , 0.06779661, 0.04166667],
       [0.16666667, 0.45833333, 0.08474576, 0.        ],
       [0.30555556, 0.70833333, 0.08474576, 0.04166667],
       [0.13888889, 0.58333333, 0.10169492, 0.04166667],
       [0.13888889, 0.41666667, 0.06779661, 0.        ],
       [0.        , 0.41666667, 0.01694915, 0.        ],
       [0.41666667, 0.83333333, 0.03389831, 0.04166667],
       [0.38888889, 1.        , 0.08474576, 0.125     ],
       [0.30555556, 0.79166667, 0.05084746, 0.125     ],
       [0.22222222, 0.625     , 0.06779661, 0.08333333],
       [0.38888889, 0.75      , 0.11864407, 0.08333333],
       [0.22222222, 0.75      , 0.08474576, 0.08333333],
       [0.30555556, 0.58333333, 0.11864407, 0.04166667],
       [0.22222222, 0.70833333, 0.08474576, 0.125     ],
       [0.08333333, 0.66666667, 0.        , 0.04166667],
       [0.22222222, 0.54166667, 0.11864407, 0.16666667],
       [0.13888889, 0.58333333, 0.15254237, 0.04166667],
       [0.19444444, 0.41666667, 0.10169492, 0.04166667],
       [0.19444444, 0.58333333, 0.10169492, 0.125     ],
       [0.25      , 0.625     , 0.08474576, 0.04166667],
       [0.25      , 0.58333333, 0.06779661, 0.04166667],
       [0.11111111, 0.5       , 0.10169492, 0.04166667],
       [0.13888889, 0.45833333, 0.10169492, 0.04166667],
       [0.30555556, 0.58333333, 0.08474576, 0.125     ],
       [0.25      , 0.875     , 0.08474576, 0.        ],
       [0.33333333, 0.91666667, 0.06779661, 0.04166667],
       [0.16666667, 0.45833333, 0.08474576, 0.        ],
       [0.19444444, 0.5       , 0.03389831, 0.04166667],
       [0.33333333, 0.625     , 0.05084746, 0.04166667],
       [0.16666667, 0.45833333, 0.08474576, 0.        ],
       [0.02777778, 0.41666667, 0.05084746, 0.04166667],
       [0.22222222, 0.58333333, 0.08474576, 0.04166667],
       [0.19444444, 0.625     , 0.05084746, 0.08333333],
       [0.05555556, 0.125     , 0.05084746, 0.08333333],
       [0.02777778, 0.5       , 0.05084746, 0.04166667],
       [0.19444444, 0.625     , 0.10169492, 0.20833333],
       [0.22222222, 0.75      , 0.15254237, 0.125     ],
       [0.13888889, 0.41666667, 0.06779661, 0.08333333],
       [0.22222222, 0.75      , 0.10169492, 0.04166667],
       [0.08333333, 0.5       , 0.06779661, 0.04166667],
       [0.27777778, 0.70833333, 0.08474576, 0.04166667],
       [0.19444444, 0.54166667, 0.06779661, 0.04166667],
```

```
       [0.75      , 0.5       , 0.62711864, 0.54166667],
       [0.58333333, 0.5       , 0.59322034, 0.58333333],
       [0.72222222, 0.45833333, 0.66101695, 0.58333333],
       [0.33333333, 0.125     , 0.50847458, 0.5       ],
       [0.61111111, 0.33333333, 0.61016949, 0.58333333],
       [0.38888889, 0.33333333, 0.59322034, 0.5       ],
       [0.55555556, 0.54166667, 0.62711864, 0.625     ],
       [0.16666667, 0.16666667, 0.38983051, 0.375     ],
       [0.63888889, 0.375     , 0.61016949, 0.5       ],
       [0.25      , 0.29166667, 0.49152542, 0.54166667],
       [0.19444444, 0.        , 0.42372881, 0.375     ],
       [0.44444444, 0.41666667, 0.54237288, 0.58333333],
       [0.47222222, 0.08333333, 0.50847458, 0.375     ],
       [0.5       , 0.375     , 0.62711864, 0.54166667],
       [0.36111111, 0.375     , 0.44067797, 0.5       ],
       [0.66666667, 0.45833333, 0.57627119, 0.54166667],
       [0.36111111, 0.41666667, 0.59322034, 0.58333333],
       [0.41666667, 0.29166667, 0.52542373, 0.375     ],
       [0.52777778, 0.08333333, 0.59322034, 0.58333333],
       [0.36111111, 0.20833333, 0.49152542, 0.41666667],
       [0.44444444, 0.5       , 0.6440678 , 0.70833333],
       [0.5       , 0.33333333, 0.50847458, 0.5       ],
       [0.55555556, 0.20833333, 0.66101695, 0.58333333],
       [0.5       , 0.33333333, 0.62711864, 0.45833333],
       [0.58333333, 0.375     , 0.55932203, 0.5       ],
       [0.63888889, 0.41666667, 0.57627119, 0.54166667],
       [0.69444444, 0.33333333, 0.6440678 , 0.54166667],
       [0.66666667, 0.41666667, 0.6779661 , 0.66666667],
       [0.47222222, 0.375     , 0.59322034, 0.58333333],
       [0.38888889, 0.25      , 0.42372881, 0.375     ],
       [0.33333333, 0.16666667, 0.47457627, 0.41666667],
       [0.33333333, 0.16666667, 0.45762712, 0.375     ],
       [0.41666667, 0.29166667, 0.49152542, 0.45833333],
       [0.47222222, 0.29166667, 0.69491525, 0.625     ],
       [0.30555556, 0.41666667, 0.59322034, 0.58333333],
       [0.47222222, 0.58333333, 0.59322034, 0.625     ],
       [0.66666667, 0.45833333, 0.62711864, 0.58333333],
       [0.55555556, 0.125     , 0.57627119, 0.5       ],
       [0.36111111, 0.41666667, 0.52542373, 0.5       ],
       [0.33333333, 0.20833333, 0.50847458, 0.5       ],
       [0.33333333, 0.25      , 0.57627119, 0.45833333],
       [0.5       , 0.41666667, 0.61016949, 0.54166667],
       [0.41666667, 0.25      , 0.50847458, 0.45833333],
       [0.19444444, 0.125     , 0.38983051, 0.375     ],
       [0.36111111, 0.29166667, 0.54237288, 0.5       ],
       [0.38888889, 0.41666667, 0.54237288, 0.45833333],
       [0.38888889, 0.375     , 0.54237288, 0.5       ],
       [0.52777778, 0.375     , 0.55932203, 0.5       ],
       [0.22222222, 0.20833333, 0.33898305, 0.41666667],
       [0.38888889, 0.33333333, 0.52542373, 0.5       ],
       [0.55555556, 0.54166667, 0.84745763, 1.        ],
       [0.41666667, 0.29166667, 0.69491525, 0.75      ],
       [0.77777778, 0.41666667, 0.83050847, 0.83333333],
       [0.55555556, 0.375     , 0.77966102, 0.70833333],
       [0.61111111, 0.41666667, 0.81355932, 0.875     ],
       [0.91666667, 0.41666667, 0.94915254, 0.83333333],
       [0.16666667, 0.20833333, 0.59322034, 0.66666667],
       [0.83333333, 0.375     , 0.89830508, 0.70833333],
       [0.66666667, 0.20833333, 0.81355932, 0.70833333],
       [0.80555556, 0.66666667, 0.86440678, 1.        ],
       [0.61111111, 0.5       , 0.69491525, 0.79166667],
       [0.58333333, 0.29166667, 0.72881356, 0.75      ],
       [0.69444444, 0.41666667, 0.76271186, 0.83333333],
       [0.38888889, 0.20833333, 0.6779661 , 0.79166667],
       [0.41666667, 0.33333333, 0.69491525, 0.95833333],
       [0.58333333, 0.5       , 0.72881356, 0.91666667],
       [0.61111111, 0.41666667, 0.76271186, 0.70833333],
       [0.94444444, 0.75      , 0.96610169, 0.875     ],
       [0.94444444, 0.25      , 1.        , 0.91666667],
       [0.47222222, 0.08333333, 0.6779661 , 0.58333333],
       [0.72222222, 0.5       , 0.79661017, 0.91666667],
       [0.36111111, 0.33333333, 0.66101695, 0.79166667],
```

```
       [0.94444444, 0.33333333, 0.96610169, 0.79166667],
       [0.55555556, 0.29166667, 0.66101695, 0.70833333],
       [0.66666667, 0.54166667, 0.79661017, 0.83333333],
       [0.80555556, 0.5       , 0.84745763, 0.70833333],
       [0.52777778, 0.33333333, 0.6440678 , 0.70833333],
       [0.5       , 0.41666667, 0.66101695, 0.70833333],
       [0.58333333, 0.33333333, 0.77966102, 0.83333333],
       [0.80555556, 0.41666667, 0.81355932, 0.625     ],
       [0.86111111, 0.33333333, 0.86440678, 0.75      ],
       [1.        , 0.75      , 0.91525424, 0.79166667],
       [0.58333333, 0.33333333, 0.77966102, 0.875     ],
       [0.55555556, 0.33333333, 0.69491525, 0.58333333],
       [0.5       , 0.25      , 0.77966102, 0.54166667],
       [0.94444444, 0.41666667, 0.86440678, 0.91666667],
       [0.55555556, 0.58333333, 0.77966102, 0.95833333],
       [0.58333333, 0.45833333, 0.76271186, 0.70833333],
       [0.47222222, 0.41666667, 0.6440678 , 0.70833333],
       [0.72222222, 0.45833333, 0.74576271, 0.83333333],
       [0.66666667, 0.45833333, 0.77966102, 0.95833333],
       [0.72222222, 0.45833333, 0.69491525, 0.91666667],
       [0.41666667, 0.29166667, 0.69491525, 0.75      ],
       [0.69444444, 0.5       , 0.83050847, 0.91666667],
       [0.66666667, 0.54166667, 0.79661017, 1.        ],
       [0.66666667, 0.41666667, 0.71186441, 0.91666667],
       [0.55555556, 0.20833333, 0.6779661 , 0.75      ],
       [0.61111111, 0.41666667, 0.71186441, 0.79166667],
       [0.52777778, 0.58333333, 0.74576271, 0.91666667],
       [0.44444444, 0.41666667, 0.69491525, 0.70833333]])
```

In [44]:

```
d1=[]
d2=[]
d3=[]
d4=[]
for i in range(len(df2)):
    for j in range(len(df2[0])):
        if j==0:
            d1.append(df2[i][j])
        elif j==1:
            d2.append(df2[i][j])
        elif j==2:
            d3.append(df2[i][j])
        else:
            d4.append(df2[i][j])
```

In [45]:

```
z=1
for i in df.columns:
    if z==1:
        df[i]=d1
    elif z==2:
        df[i]=d2
    elif z==3:
        df[i]=d3
    else:
        df[i]=d4
    z+=1
```

In [46]:

```
df.head()
```

Out[46]:

	sepal_length	sepal_width	petal_length	petal_width
0	0.166667	0.416667	0.067797	0.041667
1	0.111111	0.500000	0.050847	0.041667

	sepal_length	sepal_width	petal_length	petal_width
2	0.083333	0.458333	0.084746	0.041667
3	0.194444	0.666667	0.067797	0.041667
4	0.305556	0.791667	0.118644	0.125000

In [47]:

```
df.describe()
```

Out[47]:

	sepal_length	sepal_width	petal_length	petal_width
count	149.000000	149.000000	149.000000	149.000000
mean	0.430089	0.437919	0.470254	0.460570
std	0.230165	0.180625	0.298246	0.317205
min	0.000000	0.000000	0.000000	0.000000
25%	0.222222	0.333333	0.101695	0.083333
50%	0.416667	0.416667	0.576271	0.500000
75%	0.583333	0.541667	0.694915	0.708333
max	1.000000	1.000000	1.000000	1.000000

2:

(A). Create a similarity matrix S of size m×m where, each (i, j)-th entry in the matrix gives the dissimilarity measurement between i-th and j-th objects. Use Eucledian distance to measure the dissimilarity

In [48]:

```
from scipy.spatial.distance import euclidean, pdist, squareform
def similarity_func(u, v):
    return 1/(1+euclidean(u,v))
```

In [49]:

```
dist = pdist(df,similarity_func)
```

In [50]:

```
dist
```

Out[50]:

```
array([0.90778845, 0.91349343, 0.79901659, ..., 0.81504375, 0.84238316,
       0.77875805])
```

In [51]:

```
df_elucid = pd.DataFrame(squareform(dist), columns=df.index, index=df.index)
```

In [52]:

```
df_elucid
```

Out[52]:

	0	1	2	3	4	5	6	7	8	9	...	139	14
0	0.000000	0.907788	0.913493	0.799017	0.708397	0.839672	0.854837	0.873359	0.942228	0.755577	...	0.441614	0.4520
1	0.907788	0.000000	0.942977	0.842383	0.731726	0.910176	0.890766	0.868671	0.919292	0.777010	...	0.435379	0.4449
2	0.913493	0.942977	0.000000	0.808591	0.708921	0.882732	0.856719	0.907788	0.914771	0.749351	...	0.436728	0.4457

	0	1	2	3	4	5	6	7	8	9	...	139	1
3	0.799017	0.842383	0.808591	0.000000	0.837766	0.873359	0.921625	0.748544	0.823088	0.892960	...	0.440489	0.4510
4	0.708397	0.731726	0.708921	0.837766	0.000000	0.762416	0.798295	0.662165	0.722734	0.890766	...	0.460467	0.4727
...	...	...	...	...	...	...	...	...	...	...	...	...	
144	0.455371	0.448002	0.449414	0.452333	0.472169	0.451375	0.457185	0.442263	0.450709	0.460334	...	0.917578	0.9332
145	0.491698	0.479575	0.483722	0.475726	0.487381	0.479283	0.485318	0.479283	0.484933	0.480796	...	0.736055	0.7440
146	0.479868	0.471501	0.473533	0.475837	0.496263	0.474869	0.481761	0.465414	0.475238	0.484263	...	0.838318	0.8523
147	0.459517	0.455489	0.456654	0.463174	0.486643	0.462409	0.465949	0.447960	0.455996	0.470396	...	0.837224	0.8086
148	0.511118	0.502867	0.506558	0.504749	0.522932	0.507457	0.512195	0.498377	0.506025	0.510270	...	0.742080	0.7408

149 rows × 149 columns

(B). The i-th row indicates similarity of i-th object with all other objects. Find the average dissimilarity of i-th object with other objects and form a cluster Ci with i-th object and objects having dissimilarity less than the average similarity. Repeat this process for all rows of the similarity matrix. Thus, you have now m clusters.

In [53]:

```
def form_clusters(similarity_matrix):
    m = len(similarity_matrix)
    clusters = []
    for i in range(m):
        average_dissimilarity = sum(similarity_matrix[i]) - similarity_matrix[i][i]
        average_dissimilarity /= m-1
        cluster_i = [i]
        for j in range(m):
            if j != i and similarity_matrix[i][j] < average_dissimilarity:
                cluster_i.append(j)
        clusters.append(cluster_i)
    return clusters
```

In [54]:

```
clusters = form_clusters(df_elucid)
```

3:

(A). Remove the clusters (if any) which are subset of some other clusters. As a result you have now say, p(<m) clusters.

In [55]:

```
def remove_subsets(clusters):
    final_clusters = []
    for cluster in clusters:
        is_subset = False
        for existing_cluster in final_clusters:
            if set(cluster).issubset(set(existing_cluster)):
                is_subset = True
                break
            if set(existing_cluster).issubset(set(cluster)):
                is_subset = True
                break
        if not is_subset:
            final_clusters.append(cluster)
    return final_clusters
```

In [56]:

```
clusters_new = remove_subsets(clusters)
```

In [57]:

```
len(clusters_new)
```

Out[57]:

```
149
```

Due to some reasons the clusters I have made atleast have one element unique, that's why the number of clusters remain the same even after remove subset operation.

(B). Create a similarity matrix C of size p×p where, each (i, j)-th entry in the matrix gives the similarity measurement between i-th cluster Ci and j-th cluster Cj using following similarity measure.

Cij = |Ci ∩ Cj| / |Ci ∪ Cj|

In [58]:

```
def compute_similarity(cluster_i, cluster_j):
    intersection = len(set(cluster_i) & set(cluster_j))
    union = len(set(cluster_i) | set(cluster_j))
    similarity = intersection / union
    return similarity
```

In [59]:

```
def create_similarity_matrix(clusters):
    p = len(clusters)
    similarity_matrix = [[0.0] * p for _ in range(p)]
    for i in range(p):
        for j in range(i, p):
            similarity = compute_similarity(clusters[i], clusters[j])
            similarity_matrix[i][j] = similarity
            similarity_matrix[j][i] = similarity
    return similarity_matrix
```

In [60]:

```
similarity_mat = create_similarity_matrix(clusters_new)
```

In [61]:

```
smmat = pd.DataFrame(similarity_mat)
smmat
```

Out[61]:

	0	1	2	3	4	5	6	7	8	9	...	139	14
0	1.000000	0.959596	0.969388	0.922330	0.903846	0.940594	0.931373	0.938776	0.959596	0.913462	...	0.134228	0.1140
1	0.959596	1.000000	0.969388	0.941176	0.922330	0.960000	0.950495	0.919192	0.979592	0.932039	...	0.134228	0.1140
2	0.969388	0.969388	1.000000	0.931373	0.912621	0.950000	0.940594	0.928571	0.969388	0.922330	...	0.127517	0.1073
3	0.922330	0.941176	0.931373	1.000000	0.961165	0.960784	0.970588	0.883495	0.941176	0.970874	...	0.161074	0.1409
4	0.903846	0.922330	0.912621	0.961165	1.000000	0.941748	0.951456	0.883495	0.922330	0.970874	...	0.161074	0.1409
...	...	...	...	...	...	...	...	...	...	...	...	...	
144	0.107383	0.107383	0.100671	0.134228	0.134228	0.120805	0.127517	0.080537	0.107383	0.140940	...	0.917808	0.9571
145	0.054422	0.047297	0.047619	0.067114	0.074324	0.053691	0.060403	0.048611	0.047297	0.073826	...	0.688312	0.7162
146	0.093960	0.093960	0.087248	0.120805	0.128378	0.107383	0.114094	0.074324	0.093960	0.127517	...	0.840000	0.8750

	0	1	2	3	4	5	6	7	8	9	...	139	1
147	0.114094	0.114094	0.107383	0.140940	0.148649	0.127517	0.134228	0.094595	0.114094	0.147651	...	0.880000	0.8904
148	0.081081	0.081081	0.074324	0.100671	0.108108	0.087248	0.093960	0.075862	0.081081	0.107383	...	0.687500	0.71428

149 rows × 149 columns

(C). Out of all p^2 entries in matrix C, find out the maximum value. If multiple maximum values occur, choose any one randomly. Let, Ckl is the maximum value selected, that implies clusters Ck and Cl are the most similar clusters among all p clusters. Merge these two clusters Ck and Cl to get a new cluster Ckl, i.e. Ckl = Ck ∪ Cl

In [62]:

```
def find_max_similarity(similarity_matrix):
    max_value = 0.0
    max_indices = (0, 0)
    p = len(similarity_matrix)
    for i in range(p):
        for j in range(i + 1, p):
            if similarity_matrix[i][j] > max_value:
                max_value = similarity_matrix[i][j]
                max_indices = (i, j)
    return max_value, max_indices

def merge_clusters(cluster_i, cluster_j):
    return list(set(cluster_i) | set(cluster_j))

def merge_most_similar_clusters(clusters, similarity_matrix):
    max_similarity, (idx_k, idx_l) = find_max_similarity(similarity_matrix)
    cluster_k = clusters[idx_k]
    cluster_l = clusters[idx_l]
    new_cluster_kl = merge_clusters(cluster_k, cluster_l)
    del clusters[max(idx_k, idx_l)]
    del clusters[min(idx_k, idx_l)]
    clusters.append(new_cluster_kl)
    return clusters
```

(D). Repeat steps 3. (A) to 3. (C) until desire number (say, at most K) of clusters are obtained.

In [63]:

```
def desired_clusters(similarity_matrix, K):
    clusters = form_clusters(similarity_matrix)
    while len(clusters) > K:
        similarity_matrix = create_similarity_matrix(clusters)
        clusters = remove_subsets(clusters)
        clusters = merge_most_similar_clusters(clusters, similarity_matrix)
    return clusters
```

In [64]:

```
len(clusters)
```

Out[64]:

```
149
```

In [65]:

```
K=4
final_clusters = desired_clusters(similarity_mat, K)
print(final_clusters)
```

```
[[0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24
```

```
[[0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 2
, 25, 26, 27, 28, 29, 30, 31, 32, 33, 34, 35, 36, 37, 38, 39, 40, 41, 42, 43, 44, 45, 46,
47, 48, 72, 83, 86, 96, 99, 101, 104, 105, 108, 111, 116, 117, 119, 121, 123, 129, 130, 1
34, 135, 139, 142, 143, 147], [0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 1
7, 18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31, 32, 33, 34, 35, 36, 37, 38, 39,
40, 41, 42, 43, 44, 45, 46, 47, 48, 52, 56, 58, 59, 61, 63, 68, 78, 79, 80, 88, 89, 91, 92
, 93, 97, 98, 99, 103, 106, 111, 114, 124, 129, 138, 140, 144, 147], [0, 1, 2, 3, 4, 5, 6
, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29,
30, 31, 32, 33, 34, 35, 36, 37, 38, 39, 40, 41, 42, 43, 44, 45, 46, 47, 48, 56, 57, 59, 60
, 62, 64, 65, 67, 71, 73, 74, 77, 90, 92, 97, 108, 116, 118, 130, 132, 133, 134, 136, 137
, 141, 145], [0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21
, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31, 32, 33, 34, 35, 36, 37, 38, 39, 40, 41, 42, 43,
44, 45, 46, 47, 48, 49, 50, 51, 53, 55, 56, 59, 69, 75, 76, 78, 80, 82, 84, 85, 92, 97, 10
0, 102, 107, 109, 110, 112, 113, 115, 120, 122, 125, 126, 127, 128, 131, 146, 148]]
```

In [66]:

```
len(final_clusters)
```

Out[66]:

```
4
```

Note: You will get the overlapping clusters. Next find the probability of an object to be in all theclusters. For this, you may compute the similarity of it to the mean of the cluster in which itlies. Set similarity of it to a cluster, in which it does not lie, as zero. Also put the object into a single cluster to which the similarity is maximum. Break the tie arbitrarily.

In [67]:

```
def compute_cluster_mean_similarity(cluster, similarity_matrix):
    p = len(similarity_matrix)
    cluster_sum = [0.0] * p
    for i in cluster:
        for j in range(p):
            cluster_sum[j] += similarity_matrix[i][j]
    cluster_mean = [val / len(cluster) for val in cluster_sum]
    return cluster_mean

def compute_object_similarity_to_cluster(obj_idx, cluster_mean):
    return cluster_mean[obj_idx]
```

In [68]:

```
def compute_probability_of_objects(clusters, similarity_matrix):
    p = len(similarity_matrix)
    object_to_cluster = [0] * p
    for i, cluster in enumerate(clusters):
        cluster_mean = compute_cluster_mean_similarity(cluster, similarity_matrix)
        for obj_idx in cluster:
            obj_similarity = compute_object_similarity_to_cluster(obj_idx, cluster_mean)
            object_to_cluster[obj_idx] = i
    total_objects = len(object_to_cluster)
    cluster_counts = [object_to_cluster.count(cluster_idx) for cluster_idx in set(object
_to_cluster)]
    object_probabilities = [count / total_objects for count in cluster_counts]
    return object_to_cluster, object_probabilities
```

In [69]:

```
object_to_cluster, object_probabilities = compute_probability_of_objects(final_clusters,
similarity_mat)
```

In [71]:

```
object_probabilities
```

Out[71]:

```
[0.1476510067114094,
 0.1476510067114094,
 0.1476510067114094,
 0.5570469798657718]
```

In [72]:

```
len(final_clusters)
```

Out[72]:

```
4
```

In [73]:

```
final_clusters
```

Out[73]:

```
[[0,
  1,
  2,
  3,
  4,
  5,
  6,
  7,
  8,
  9,
  10,
  11,
  12,
  13,
  14,
  15,
  16,
  17,
  18,
  19,
  20,
  21,
  22,
  23,
  24,
  25,
  26,
  27,
  28,
  29,
  30,
  31,
  32,
  33,
  34,
  35,
  36,
  37,
  38,
  39,
  40,
  41,
  42,
  43,
  44,
  45,
  46,
  47,
  48,
  72,
  83,
  86,
  96,
  99
```

```
 99,
 101,
 104,
 105,
 108,
 111,
 116,
 117,
 119,
 121,
 123,
 129,
 130,
 134,
 135,
 139,
 142,
 143,
 147],
[0,
 1,
 2,
 3,
 4,
 5,
 6,
 7,
 8,
 9,
 10,
 11,
 12,
 13,
 14,
 15,
 16,
 17,
 18,
 19,
 20,
 21,
 22,
 23,
 24,
 25,
 26,
 27,
 28,
 29,
 30,
 31,
 32,
 33,
 34,
 35,
 36,
 37,
 38,
 39,
 40,
 41,
 42,
 43,
 44,
 45,
 46,
 47,
 48,
 52,
 56,
 58,
 59,
 61,
```

61,
63,
68,
78,
79,
80,
88,
89,
91,
92,
93,
97,
98,
99,
103,
106,
111,
114,
124,
129,
138,
140,
144,
147],
[0,
1,
2,
3,
4,
5,
6,
7,
8,
9,
10,
11,
12,
13,
14,
15,
16,
17,
18,
19,
20,
21,
22,
23,
24,
25,
26,
27,
28,
29,
30,
31,
32,
33,
34,
35,
36,
37,
38,
39,
40,
41,
42,
43,
44,
45,
46,
47,
48

```
  48,
  56,
  57,
  59,
  60,
  62,
  64,
  65,
  67,
  71,
  73,
  74,
  77,
  90,
  92,
  97,
  108,
  116,
  118,
  130,
  132,
  133,
  134,
  136,
  137,
  141,
  145],
 [0,
  1,
  2,
  3,
  4,
  5,
  6,
  7,
  8,
  9,
  10,
  11,
  12,
  13,
  14,
  15,
  16,
  17,
  18,
  19,
  20,
  21,
  22,
  23,
  24,
  25,
  26,
  27,
  28,
  29,
  30,
  31,
  32,
  33,
  34,
  35,
  36,
  37,
  38,
  39,
  40,
  41,
  42,
  43,
  44,
  45
```

```
  45,
  46,
  47,
  48,
  49,
  50,
  51,
  53,
  55,
  56,
  59,
  69,
  75,
  76,
  78,
  80,
  82,
  84,
  85,
  92,
  97,
  100,
  102,
  107,
  109,
  110,
  112,
  113,
  115,
  120,
  122,
  125,
  126,
  127,
  128,
  131,
  146,
  148]]
```

In [74]:

```
sim_mat = create_similarity_matrix(final_clusters)
```

In [75]:

```
sm = pd.DataFrame(sim_mat)
```

In [76]:

```
sm
```

Out[76]:

	0	1	2	3
0	1.000000	0.552083	0.563830	0.462264
1	0.552083	1.000000	0.535354	0.523810
2	0.563830	0.535354	1.000000	0.504762
3	0.462264	0.523810	0.504762	1.000000

Membership value table

In [77]:

```
def create_membership_matrix(clusters, similarity_matrix):
    p = len(similarity_matrix)
    total_clusters = len(clusters)
    membership_matrix = [[0.0] * total_clusters for _ in range(p)]
```

```
    for obj_idx in range(p):
        object_belongs_to_cluster = [obj_idx in cluster for cluster in clusters]
        if any(object_belongs_to_cluster):
            total_similarity = sum([similarity_matrix[obj_idx][cluster_idx] for cluster_
idx, belongs in enumerate(object_belongs_to_cluster) if belongs])
            if total_similarity != 0:
                membership_matrix[obj_idx] = [similarity_matrix[obj_idx][cluster_idx] /
total_similarity if belongs else 0.0 for cluster_idx, belongs in enumerate(object_belong
s_to_cluster)]
    return membership_matrix
```

In [78]:

```
membership_matrix = create_membership_matrix(final_clusters, similarity_mat)
```

In [79]:

```
membership_matrix
```

Out[79]:

```
[[0.25965165366164317,
  0.24916067775612222,
  0.25170313365159286,
  0.23948453493064176],
 [0.24794734937437435,
  0.25838723776908484,
  0.2504774243679904,
  0.24318798848855044],
 [0.25047820917127644,
  0.25047820917127644,
  0.25838804735563253,
  0.2406555343018146],
 [0.24304597556999172,
  0.24801224008628256,
  0.24542877925205048,
  0.26351300509167525],
 [0.24428520966846648,
  0.24928092042186004,
  0.24665691073320886,
  0.25977695917646465],
 [0.24678579960381836,
  0.251877380311434,
  0.24925365759985654,
  0.25208316248489104],
 [0.24554713890879312,
  0.25058859658102156,
  0.24797829869996926,
  0.2558859658102 16],
 [0.2557947717333422,
  0.25045869283244904,
  0.2530143937797189,
  0.24073214165448983],
 [0.24926175870730996,
  0.2544558275783323,
  0.25180524604105803,
  0.2444771676732997],
 [0.2443257010162551,
  0.2492946140977303,
  0.24669779520087895,
  0.2596818896851357],
 [0.24678579960381836,
  0.251877380311434,
  0.24925365759985654,
  0.25208316248489104],
 [0.25570763850921846,
  0.2504880171002550 3,
  0.2530440172747474,
  0.24076032711577913],
 [0.2518198014762026,
  0.2518198014762026,
  0.25441588190379233,
```

```
  0.2419445151438025],
 [0.2443257010162551,
  0.2492946140977303,
  0.24669779520087895,
  0.2596818896851357],
 [0.24806201550387597,
  0.24806201550387597,
  0.2454780361757106,
  0.25839793281653745],
 [0.2443257010162551,
  0.2492946140977303,
  0.24669779520087895,
  0.2596818896851357],
 [0.24430829740564253,
  0.24930035549504576,
  0.24670347679197235,
  0.2596878703073393],
 [0.24428520966846648,
  0.24928092042186004,
  0.24665691073320886,
  0.25977695917646465],
 [0.2443257010162551,
  0.2492946140977303,
  0.24669779520087895,
  0.2596818896851357],
 [0.24678579960381836,
  0.251877380311434,
  0.24925365759985654,
  0.25208316248489104],
 [0.24430829740564253,
  0.24930035549504576,
  0.24670347679197235,
  0.2596878703073393],
 [0.24430829740564253,
  0.24930035549504576,
  0.24670347679197235,
  0.2596878703073393],
 [0.25181980147620264,
  0.25181980147620264,
  0.25441588190379233,
  0.24194451514380252],
 [0.24802407563970308,
  0.2531665205891706,
  0.2505293693330334,
  0.24828003443809285],
 [0.2557947717333422,
  0.25045869283244904,
  0.2530143937797189,
  0.24073214165448983],
 [0.2480036118400632,
  0.25317368498609505,
  0.2505086988283467,
  0.24831400434549508],
 [0.24430829740564253,
  0.24930035549504576,
  0.24670347679197235,
  0.2596878703073393],
 [0.24554713890879312,
  0.25058859658102156,
  0.24797829869996926,
  0.255885965810216],
 [0.24926175870730996,
  0.2544558275783323,
  0.25180524604105803,
  0.2444771676732997],
 [0.2518198014762026,
  0.2518198014762026,
  0.25441588190379233,
  0.2419445151438025],
 [0.24675865469693314,
  0.2518775842358589,
  0.24922624124390244,
```

```
  0.25213751982330557],
 [0.2443257010162551,
  0.2492946140977303,
  0.24669779520087895,
  0.2596818896851357],
 [0.2443257010162551,
  0.2492946140977303,
  0.24669779520087895,
  0.2596818896851357],
 [0.24926175870730996,
  0.2544558275783323,
  0.25180524604105803,
  0.2444771676732997],
 [0.24802407563970308,
  0.2531665205891706,
  0.2505293693330334,
  0.24828003443809285],
 [0.24554713890879312,
  0.25058859658102156,
  0.24797829869996926,
  0.255885965810216],
 [0.24926175870730996,
  0.2544558275783323,
  0.25180524604105803,
  0.2444771676732997],
 [0.25572872847478273,
  0.2504809193737053,
  0.2530368471224166,
  0.24075350502909543],
 [0.24554713890879312,
  0.25058859658102156,
  0.24797829869996926,
  0.255885965810216],
 [0.24430829740564253,
  0.24930035549504576,
  0.24670347679197235,
  0.2596878703073393],
 [0.2560666425208258,
  0.25032207189653355,
  0.25282529261549885,
  0.2407859929671418],
 [0.24926175870730996,
  0.2544558275783323,
  0.25180524604105803,
  0.2444771676732997],
 [0.24796136033156324,
  0.25318847728189925,
  0.25046602053693257,
  0.24838414184960508],
 [0.24428520966846648,
  0.24928092042186004,
  0.24665691073320886,
  0.25977695917646465],
 [0.25575027075400486,
  0.2504736694231679,
  0.2530295231927921,
  0.24074653663003517],
 [0.24430829740564253,
  0.24930035549504576,
  0.24670347679197235,
  0.2596878703073393],
 [0.24926175870730996,
  0.2544558275783323,
  0.25180524604105803,
  0.2444771676732997],
 [0.24430829740564253,
  0.24930035549504576,
  0.24670347679197235,
  0.2596878703073393],
 [0.24678579960381836,
  0.251877380311434,
  0.24925365759985654,
```

```
 0.25208316248489104],
[0.0, 0.0, 0.0, 1.0],
[0.0, 0.0, 0.0, 1.0],
[0.0, 0.0, 0.0, 1.0],
[0.0, 1.0, 0.0, 0.0],
[0.0, 0.0, 0.0, 1.0],
[0.0, 0.0, 0.0, 0.0],
[0.0, 0.0, 0.0, 1.0],
[0.0, 0.3298290083314288, 0.3324260871371881, 0.3377449045313831],
[0.0, 0.0, 1.0, 0.0],
[0.0, 1.0, 0.0, 0.0],
[0.0, 0.3356684381254745, 0.33811857271033197, 0.3262129891641935],
[0.0, 0.0, 1.0, 0.0],
[0.0, 1.0, 0.0, 0.0],
[0.0, 0.0, 1.0, 0.0],
[0.0, 1.0, 0.0, 0.0],
[0.0, 0.0, 1.0, 0.0],
[0.0, 0.0, 1.0, 0.0],
[0.0, 0.0, 0.0, 0.0],
[0.0, 0.0, 1.0, 0.0],
[0.0, 1.0, 0.0, 0.0],
[0.0, 0.0, 0.0, 1.0],
[0.0, 0.0, 0.0, 0.0],
[0.0, 0.0, 1.0, 0.0],
[1.0, 0.0, 0.0, 0.0],
[0.0, 0.0, 1.0, 0.0],
[0.0, 0.0, 1.0, 0.0],
[0.0, 0.0, 0.0, 1.0],
[0.0, 0.0, 0.0, 1.0],
[0.0, 0.0, 1.0, 0.0],
[0.0, 0.4989117696300017, 0.0, 0.5010882303699984],
[0.0, 1.0, 0.0, 0.0],
[0.0, 0.5068027210884355, 0.0, 0.4931972789115646],
[0.0, 0.0, 0.0, 0.0],
[0.0, 0.0, 0.0, 1.0],
[1.0, 0.0, 0.0, 0.0],
[0.0, 0.0, 0.0, 1.0],
[0.0, 0.0, 0.0, 1.0],
[1.0, 0.0, 0.0, 0.0],
[0.0, 0.0, 0.0, 0.0],
[0.0, 1.0, 0.0, 0.0],
[0.0, 1.0, 0.0, 0.0],
[0.0, 0.0, 1.0, 0.0],
[0.0, 1.0, 0.0, 0.0],
[0.0, 0.3358720487433359, 0.32901751713632904, 0.33511043412033514],
[0.0, 1.0, 0.0, 0.0],
[0.0, 0.0, 0.0, 0.0],
[0.0, 0.0, 0.0, 0.0],
[1.0, 0.0, 0.0, 0.0],
[0.0, 0.3359324236517219, 0.32878492527615333, 0.33528265107212474],
[0.0, 1.0, 0.0, 0.0],
[0.5, 0.5, 0.0, 0.0],
[0.0, 0.0, 0.0, 1.0],
[1.0, 0.0, 0.0, 0.0],
[0.0, 0.0, 0.0, 1.0],
[0.0, 1.0, 0.0, 0.0],
[1.0, 0.0, 0.0, 0.0],
[1.0, 0.0, 0.0, 0.0],
[0.0, 1.0, 0.0, 0.0],
[0.0, 0.0, 0.0, 1.0],
[0.5102040816326532, 0.0, 0.4897959183673469, 0.0],
[0.0, 0.0, 0.0, 1.0],
[0.0, 0.0, 0.0, 1.0],
[0.5, 0.5, 0.0, 0.0],
[0.0, 0.0, 0.0, 1.0],
[0.0, 0.0, 0.0, 1.0],
[0.0, 1.0, 0.0, 0.0],
[0.0, 0.0, 0.0, 1.0],
[0.5102040816326532, 0.0, 0.4897959183673469, 0.0],
[1.0, 0.0, 0.0, 0.0],
[0.0, 0.0, 1.0, 0.0],
[1.0, 0.0, 0.0, 0.0],
```

```
 [0.0, 0.0, 0.0, 1.0],
 [1.0, 0.0, 0.0, 0.0],
 [0.0, 0.0, 0.0, 1.0],
 [1.0, 0.0, 0.0, 0.0],
 [0.0, 1.0, 0.0, 0.0],
 [0.0, 0.0, 0.0, 1.0],
 [0.0, 0.0, 0.0, 1.0],
 [0.0, 0.0, 0.0, 1.0],
 [0.0, 0.0, 0.0, 1.0],
 [0.5, 0.5, 0.0, 0.0],
 [0.5106382978723404, 0.0, 0.4893617021276596, 0.0],
 [0.0, 0.0, 0.0, 1.0],
 [0.0, 0.0, 1.0, 0.0],
 [0.0, 0.0, 1.0, 0.0],
 [0.5111111111111111, 0.0, 0.48888888888888893, 0.0],
 [1.0, 0.0, 0.0, 0.0],
 [0.0, 0.0, 1.0, 0.0],
 [0.0, 0.0, 1.0, 0.0],
 [0.0, 1.0, 0.0, 0.0],
 [1.0, 0.0, 0.0, 0.0],
 [0.0, 1.0, 0.0, 0.0],
 [0.0, 0.0, 1.0, 0.0],
 [1.0, 0.0, 0.0, 0.0],
 [1.0, 0.0, 0.0, 0.0],
 [0.0, 1.0, 0.0, 0.0],
 [0.0, 0.0, 1.0, 0.0],
 [0.0, 0.0, 0.0, 1.0],
 [0.5, 0.5, 0.0, 0.0],
 [0.0, 0.0, 0.0, 1.0]]
```

In [80]:

```
fc = final_clusters
fcd = pd.DataFrame(fc)
fcd
```

Out[80]:

	0	1	2	3	4	5	6	7	8	9	...	73	74	75	76	77	78	79	80	81	82
0	0	1	2	3	4	5	6	7	8	9	...	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
1	0	1	2	3	4	5	6	7	8	9	...	138.0	140.0	144.0	147.0	NaN	NaN	NaN	NaN	NaN	NaN
2	0	1	2	3	4	5	6	7	8	9	...	141.0	145.0	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
3	0	1	2	3	4	5	6	7	8	9	...	115.0	120.0	122.0	125.0	126.0	127.0	128.0	131.0	146.0	148.0

4 rows × 83 columns

In [81]:

```
mem_mat = pd.DataFrame(membership_matrix)
mem_mat
```

Out[81]:

	0	1	2	3
0	0.259652	0.249161	0.251703	0.239485
1	0.247947	0.258387	0.250477	0.243188
2	0.250478	0.250478	0.258388	0.240656
3	0.243046	0.248012	0.245429	0.263513
4	0.244285	0.249281	0.246657	0.259777
...	...	...	...	...
144	0.000000	1.000000	0.000000	0.000000
145	0.000000	0.000000	1.000000	0.000000
146	0.000000	0.000000	0.000000	1.000000

	0	1	2	3
147	0.500000	0.500000	0.000000	0.000000
148	0.000000	0.000000	0.000000	1.000000

149 rows × 4 columns