Name: Tushar Nankani
Batch: C23
Roll No: 1902112
Question Number: 10

DWM PRACTICAL EXAM

AIM

We have several objects (5 types of medicines) and each object have two attributes or features Weight Index and as pH shown in table below.
Cluster the objects using hierarchical clustering.

Medicine	WeightIndex	pH
A	1	3
B	2	5
C	5	4
D	1	2
E	5	6

Solution

Data Cleaning:

Since an entire row is missing, we will have to ignore the tuple completely in the data cleaning process.

Cleaned and updated dataset:

Medicine	WeightIndex	pH
A	1	3
B	2	5
C	5	4
D	1	2
E	5	6

For hierarchical clustering:

- Importing Necessary Libraries

```
import math
import pandas as pd
import numpy as np
import scipy.cluster.hierarchy as shc
import matplotlib.pyplot as plt
```

- Loading the data in the notebook:

```
data = {
    'Medicine' : ['A', 'B', 'C', 'D', 'E'],
    'WeightIndex' : [1, 2, 5, 1, 5],
    'pH' : [3, 5, 4, 2, 6]
}
data = pd.DataFrame.from_dict(data)
```

```
In [3]: data = pd.DataFrame.from_dict(data)

In [4]: data

Out[4]:
```

	Medicine	WeightIndex	pH
0	A	1	3
1	B	2	5
2	C	5	4
3	D	1	2
4	E	5	6

- Final dataset for clustering

```
data = data[['WeightIndex', 'pH']]
```

```
In [8]: data = data[['WeightIndex', 'pH']]

In [9]: data

Out[9]:
```

	WeightIndex	pH
0	1	3
1	2	5
2	5	4
3	1	2
4	5	6

- calculate_dist function between 2 clusters

```
def calculate_dist(cluster1, cluster2):

    maximum_dist = -1
    for point1 in cluster1:
        for point2 in cluster2:

            dist = 0
            for i in range(len(point1)-1):
                dist += math.pow((point1[i] - point2[i]), 2)

            dist = math.sqrt(dist)

            if dist > maximum_dist:
                maximum_dist = dist

    return maximum_dist
```

- renaming_clusters function to rename the clusters

```
def renaming_clusters(dataset, num):

    clusters = list(pd.unique(dataset['Cluster']))
    new_clusters = list(range(num))

    mapping = {}
    for key, val in zip(clusters, new_clusters):
        mapping[key] = val

    dataset["Cluster"] = dataset["Cluster"].map(lambda x:
mapping[x])
    return dataset
```

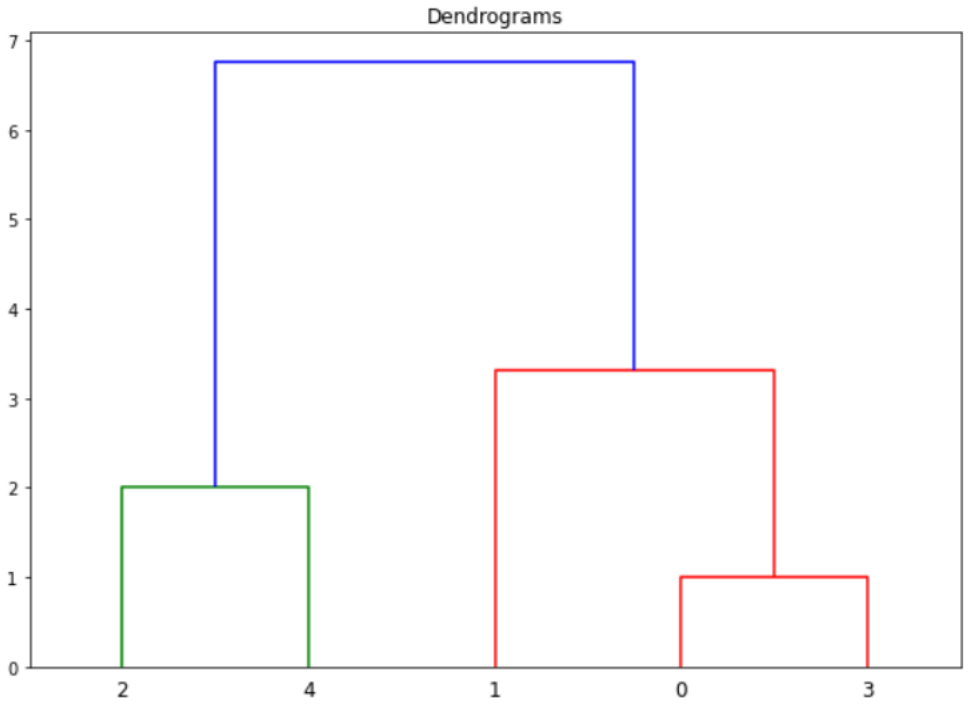
- hierarchical_clustering function between 2 clusters

```
def hierarchical_clustering(dataset, num):

    dataset['Cluster'] = dataset.index

    while (True):
        clusters = []

        for i in range(max(pd.unique(dataset['Cluster'])) 
+ 1):
            cl = 
dataset[dataset['Cluster']==i].values.tolist()
            if len(cl) > 0:
                clusters.append(cl)

        if len(clusters) == num:
            dataset = renaming_clusters(dataset,num)
            return dataset

        minimum_dist = math.inf
        c1 = -1
        c2 = -1

        for i in range(len(clusters) - 1):
            for j in range(i + 1, len(clusters)):

                cluster1 = clusters[i]
                cluster2 = clusters[j]
                dist = calculate_dist(cluster1, cluster2)

                if dist < minimum_dist:
                    minimum_dist = dist
                    c1 = cluster1[0][-1]
                    c2 = cluster2[0][-1]

        dataset.loc[dataset.Cluster == max(c1, c2), 
"Cluster"] = min(c1, c2)
```

- Creating a Dendogram for predicting the approximate number of clusters.

```
import scipy.cluster.hierarchy as shc
import matplotlib.pyplot as plt

plt.figure(figsize=(10, 7))
plt.title("Dendrograms")
dend = shc.dendrogram(shc.linkage(data, method='ward'))
```

```
In [37]: plt.figure(figsize=(10, 7))
         plt.title("Dendrograms")
         dend = shc.dendrogram(shc.linkage(data, method='ward'))
```



- The above dendogram shows that the ideal number of clusters.

```
num = int(input('Enter number of clusters : '))
result = hierarchical_clustering(data, num)
result
```

In [40]:
```
#The above dendogram shows that the ideal number of clusters should be 2
num = int(input('Enter number of clusters : '))
result = hierarchical_clustering(data, num)
result
```

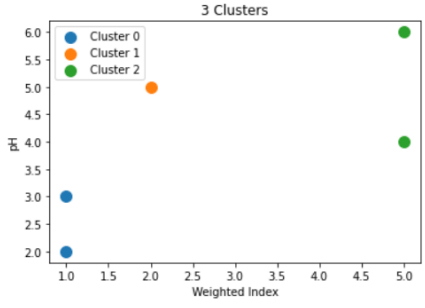
```
Enter number of clusters : 3
```

Out[40]:

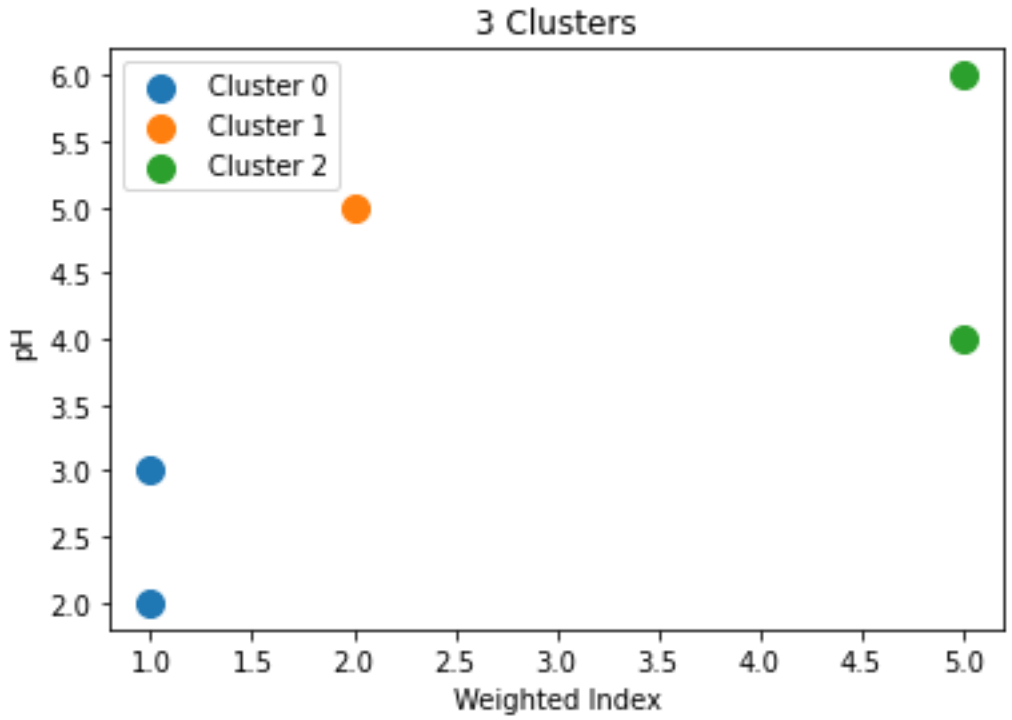
	WeightIndex	pH	Cluster
0	1	3	0
1	2	5	1
2	5	4	2
3	1	2	0
4	5	6	2

```
for i in range(num):
    plt.scatter(result[result['Cluster'] ==
i][result.columns[0]], result[result['Cluster'] ==
i][result.columns[1]], s = 100, label = 'Cluster
{}'.format(i))

plt.title('{} Clusters'.format(num))
plt.xlabel('Weighted Index')
plt.ylabel('pH')
plt.legend()
plt.show()
```

In [41]:
```
for i in range(num):
    plt.scatter(result[result['Cluster'] == i][result.columns[0]], result[result['Cluster'] == i][result.columns[1]], s = 100, la

plt.title('{} Clusters'.format(num))
plt.xlabel('Weighted Index')
plt.ylabel('pH')
plt.legend()
plt.show()
```



FINAL OUTPUT:



Name: Tushar Nankani

Roll No: 1902112

Batch: C23

In [1]:
```
import pandas as pd
import numpy as np
import math
```

In [2]:
```
data = {
    'Medicine' : ['A', 'B', 'C', 'D', 'E'],
    'WeightIndex' : [1, 2, 5, 1, 5],
    'pH' : [3, 5, 4, 2, 6]
}
```

In [3]:
```
data = pd.DataFrame.from_dict(data)
```

In [4]:
```
data
```

Out[4]:

	Medicine	WeightIndex	pH
0	A	1	3
1	B	2	5
2	C	5	4
3	D	1	2
4	E	5	6

In [8]:
```
data = data[['WeightIndex', 'pH']]
```

In [9]:
```
data
```

Out[9]:

	WeightIndex	pH
0	1	3
1	2	5
2	5	4
3	1	2
4	5	6

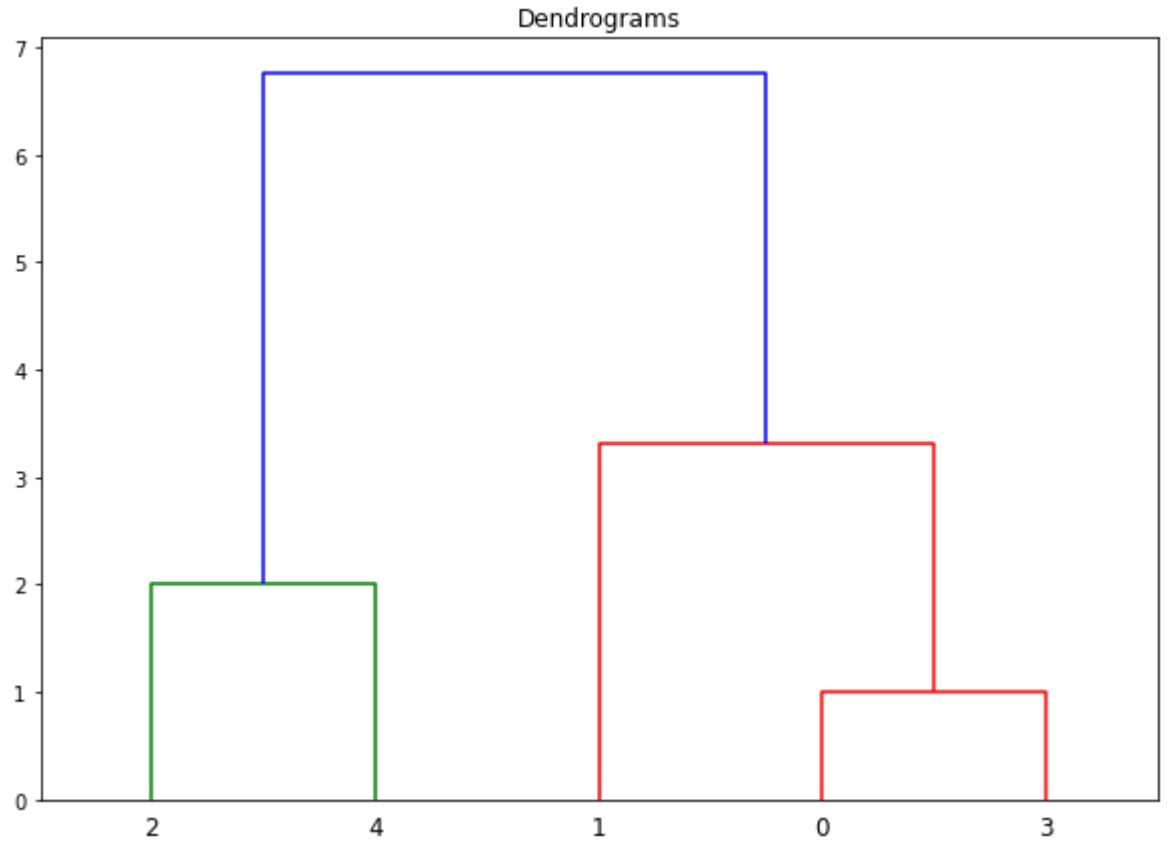
In [12]:
```
def calculate_dist(cluster1, cluster2):

    maximum_dist = -1
    for point1 in cluster1:
        for point2 in cluster2:

            dist = 0
            for i in range(len(point1)-1):
                dist += math.pow((point1[i] - point2[i]), 2)

            dist = math.sqrt(dist)

            if dist > maximum_dist:
                maximum_dist = dist

    return maximum_dist
```

In [13]:
```
def renaming_clusters(dataset, num):

    clusters = list(pd.unique(dataset['Cluster']))
    new_clusters = list(range(num))

    mapping = {}
    for key, val in zip(clusters, new_clusters):
        mapping[key] = val

    dataset["Cluster"] = dataset["Cluster"].map(lambda x: mapping[x])
    return dataset
```

In [35]:
```
def hierarchical_clustering(dataset, num):

    dataset['Cluster'] = dataset.index

    while (True):
        clusters = []

        for i in range(max(pd.unique(dataset['Cluster'])) + 1):
            cl = dataset[dataset['Cluster']==i].values.tolist()
            if len(cl) > 0:
                clusters.append(cl)

        if len(clusters) == num:
            dataset = renaming_clusters(dataset,num)
            return dataset

        minimum_dist = math.inf
        c1 = -1
        c2 = -1

        for i in range(len(clusters) - 1):
            for j in range(i + 1, len(clusters)):

                cluster1 = clusters[i]
                cluster2 = clusters[j]
                dist = calculate_dist(cluster1, cluster2)

                if dist < minimum_dist:
                    minimum_dist = dist
                    c1 = cluster1[0][-1]
                    c2 = cluster2[0][-1]

        dataset.loc[dataset.Cluster == max(c1, c2), "Cluster"] = min(c1, c2)
```

In [36]:
```
import scipy.cluster.hierarchy as shc
import matplotlib.pyplot as plt
```

In [37]:
```
plt.figure(figsize=(10, 7))
plt.title("Dendrograms")
dend = shc.dendrogram(shc.linkage(data, method='ward'))
```



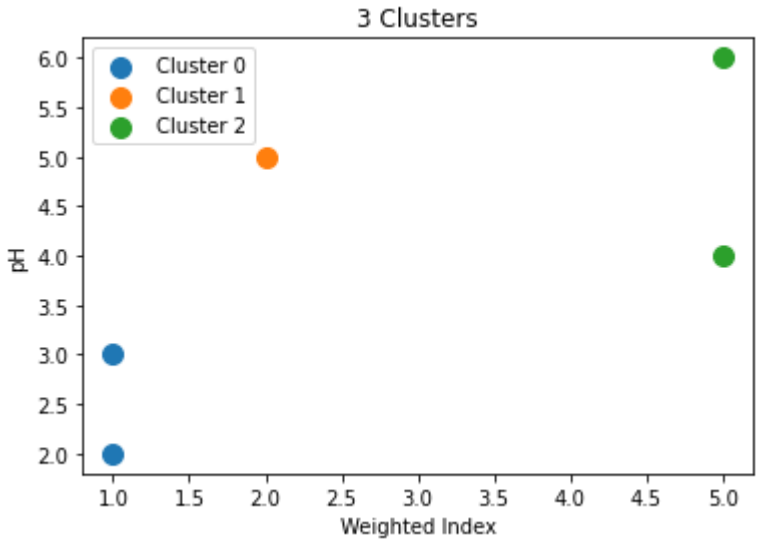
In [40]:
```
#The above dendogram shows that the ideal number of clusters should be 2
num = int(input('Enter number of clusters : '))
result = hierarchical_clustering(data, num)
result
```

```
Enter number of clusters : 3
```

Out[40]:

	WeightIndex	pH	Cluster
0	1	3	0
1	2	5	1
2	5	4	2
3	1	2	0
4	5	6	2

In [41]:
```
for i in range(num):
    plt.scatter(result[result['Cluster'] == i][result.columns[0]], result[result[

plt.title('{} Clusters'.format(num))
plt.xlabel('Weighted Index')
plt.ylabel('pH')
plt.legend()
plt.show()
```



In [ ]: