In [1]:
```
#import packages and clean data before running the principal component analysis
import numpy as np
import pandas as pd
from sklearn import linear_model
from sklearn.preprocessing import StandardScaler
import matplotlib.pyplot as plt
import seaborn as sns
%matplotlib inline
pd.set_option('display.max_columns', None)
import pylab
from pylab import rcParams
import statsmodels.api as sm
import statistics
from scipy import stats
from scipy.cluster.hierarchy import linkage, fcluster
from scipy.cluster.hierarchy import dendrogram
from sklearn.metrics import silhouette_score
import sklearn
from sklearn import preprocessing
from sklearn.model_selection import train_test_split
from sklearn import metrics
from sklearn.metrics import classification_report
from scipy.stats import chisquare
from scipy.stats import chi2_contingency
from sklearn.decomposition import PCA
from sklearn.model_selection import train_test_split
from sklearn.tree import DecisionTreeClassifier
from sklearn.metrics import confusion_matrix
from sklearn.metrics import roc_auc_score
from sklearn.metrics import roc_curve
from sklearn.metrics import accuracy_score

df = pd.read_csv (r'C:\Users\fahim\Documents\0_WGUDocuments\d208\1medical_clean.csv')
df.head()
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 10000 entries, 0 to 9999
Data columns (total 50 columns):
 #   Column              Non-Null Count  Dtype
---  ------              --------------  -----
 0   CaseOrder           10000 non-null  int64
 1   Customer_id         10000 non-null  object
 2   Interaction         10000 non-null  object
 3   UID                 10000 non-null  object
 4   City                10000 non-null  object
 5   State               10000 non-null  object
 6   County              10000 non-null  object
 7   Zip                 10000 non-null  int64
 8   Lat                 10000 non-null  float64
 9   Lng                 10000 non-null  float64
 10  Population          10000 non-null  int64
 11  Area                10000 non-null  object
 12  TimeZone            10000 non-null  object
 13  Job                 10000 non-null  object
 14  Children            10000 non-null  int64
 15  Age                 10000 non-null  int64
 16  Income              10000 non-null  float64
 17  Marital             10000 non-null  object
 18  Gender              10000 non-null  object
 19  ReAdmis             10000 non-null  object
 20  VitD_levels         10000 non-null  float64
 21  Doc_visits          10000 non-null  int64
 22  Full_meals_eaten    10000 non-null  int64
 23  vitD_supp           10000 non-null  int64
 24  Soft_drink          10000 non-null  object
 25  Initial_admin       10000 non-null  object
 26  HighBlood           10000 non-null  object
 27  Stroke              10000 non-null  object
 28  Complication_risk   10000 non-null  object
 29  Overweight          10000 non-null  object
 30  Arthritis           10000 non-null  object
 31  Diabetes            10000 non-null  object
 32  Hyperlipidemia      10000 non-null  object
 33  BackPain            10000 non-null  object
 34  Anxiety             10000 non-null  object
 35  Allergic_rhinitis   10000 non-null  object
 36  Reflux_esophagitis  10000 non-null  object
 37  Asthma              10000 non-null  object
 38  Services            10000 non-null  object
 39  Initial_days        10000 non-null  float64
```

```
 40  TotalCharge         10000 non-null  float64
 41  Additional_charges  10000 non-null  float64
 42  Item1               10000 non-null  int64
 43  Item2               10000 non-null  int64
 44  Item3               10000 non-null  int64
 45  Item4               10000 non-null  int64
 46  Item5               10000 non-null  int64
 47  Item6               10000 non-null  int64
 48  Item7               10000 non-null  int64
 49  Item8               10000 non-null  int64
dtypes: float64(7), int64(16), object(27)
memory usage: 3.8+ MB
```

In [2]:
```
#check if there are any missing data entries - if there are none then the output should be False
df.isna().any()
```

Out[2]:

CaseOrder	False
Customer_id	False
Interaction	False
UID	False
City	False
State	False
County	False
Zip	False
Lat	False
Lng	False
Population	False
Area	False
TimeZone	False
Job	False
Children	False
Age	False
Income	False
Marital	False
Gender	False
ReAdmis	False
VitD_levels	False
Doc_visits	False
Full_meals_eaten	False
vitD_supp	False
Soft_drink	False
Initial_admin	False
HighBlood	False
Stroke	False
Complication_risk	False
Overweight	False
Arthritis	False
Diabetes	False
Hyperlipidemia	False
BackPain	False
Anxiety	False
Allergic_rhinitis	False
Reflux_esophagitis	False
Asthma	False
Services	False
Initial_days	False
TotalCharge	False
Additional_charges	False
Item1	False
Item2	False
Item3	False

```
Item4                False
Item5                False
Item6                False
Item7                False
Item8                False
dtype: bool
```

In [3]:
```
# check if there are any duplicated columns in the data set - if there are none then the output should be False
df.columns.duplicated().any()
```

Out[3]:
```
False
```

In [4]:
```
# check if there are any duplicated rows in the data set - if there are none then the output should be False
df.duplicated().any()
```

Out[4]:
```
False
```

In [5]:
```
# rename the item columns accordingly
df.rename(columns={'Item1':'Timely_admis','Item2':'Timely_treat',
 'Item3':'Timely_visits','Item4':'Reliability',
 'Item5':'Options','Item6':'Hrs_treat',
 'Item7':'Courteous','Item8':'Active_listen'},inplace=True)
df.head()
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 10000 entries, 0 to 9999
Data columns (total 50 columns):
 #   Column              Non-Null Count  Dtype  
---  ------              --------------  -----  
 0   CaseOrder           10000 non-null  int64  
 1   Customer_id         10000 non-null  object 
 2   Interaction         10000 non-null  object 
 3   UID                 10000 non-null  object 
 4   City                10000 non-null  object 
 5   State               10000 non-null  object 
 6   County              10000 non-null  object 
 7   Zip                 10000 non-null  int64  
 8   Lat                 10000 non-null  float64
 9   Lng                 10000 non-null  float64
 10  Population          10000 non-null  int64  
 11  Area                10000 non-null  object 
 12  TimeZone            10000 non-null  object 
 13  Job                 10000 non-null  object 
 14  Children            10000 non-null  int64  
 15  Age                 10000 non-null  int64  
 16  Income              10000 non-null  float64
 17  Marital             10000 non-null  object 
 18  Gender              10000 non-null  object 
 19  ReAdmis             10000 non-null  object 
 20  VitD_levels         10000 non-null  float64
 21  Doc_visits          10000 non-null  int64  
 22  Full_meals_eaten    10000 non-null  int64  
 23  vitD_supp           10000 non-null  int64  
 24  Soft_drink          10000 non-null  object 
 25  Initial_admin       10000 non-null  object 
 26  HighBlood           10000 non-null  object 
 27  Stroke              10000 non-null  object 
 28  Complication_risk   10000 non-null  object 
 29  Overweight          10000 non-null  object 
 30  Arthritis           10000 non-null  object 
 31  Diabetes            10000 non-null  object 
 32  Hyperlipidemia      10000 non-null  object 
 33  BackPain            10000 non-null  object 
 34  Anxiety             10000 non-null  object 
 35  Allergic_rhinitis   10000 non-null  object 
 36  Reflux_esophagitis  10000 non-null  object 
 37  Asthma              10000 non-null  object 
 38  Services            10000 non-null  object 
 39  Initial_days        10000 non-null  float64
```

```
 40  TotalCharge         10000 non-null  float64
 41  Additional_charges  10000 non-null  float64
 42  Timely_admis        10000 non-null  int64
 43  Timely_treat        10000 non-null  int64
 44  Timely_visits       10000 non-null  int64
 45  Reliability         10000 non-null  int64
 46  Options             10000 non-null  int64
 47  Hrs_treat           10000 non-null  int64
 48  Courteous           10000 non-null  int64
 49  Active_listen       10000 non-null  int64
dtypes: float64(7), int64(16), object(27)
memory usage: 3.8+ MB
```

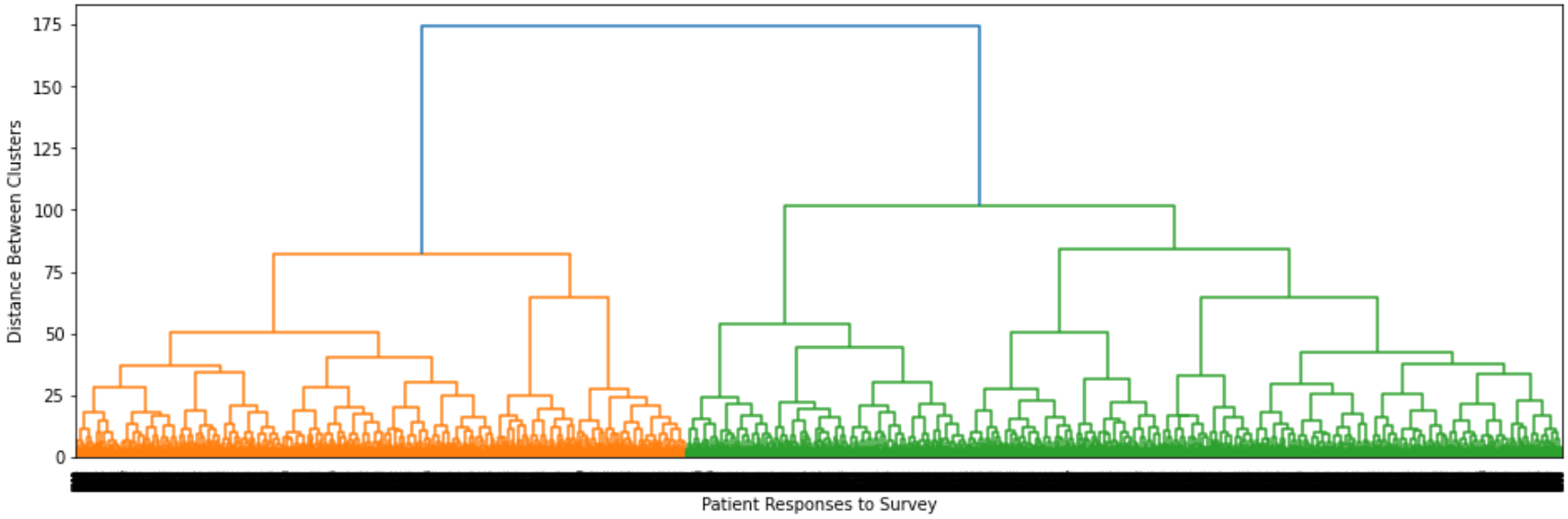
In [6]:
```
#print the mean overall scores and and standard deviations of the target variables
print(f"The mean overall score for Timely_admis is {round(df.Timely_admis.mean(), 3)}, with a standard deviation of {ro
print(f"The mean overall score for Timely_treat is {round(df.Timely_treat.mean(), 3)}, with a standard deviation of {ro
print(f"The mean overall score for Timely_visits is {round(df.Timely_visits.mean(), 3)}, with a standard deviation of {
print(f"The mean overall score for Reliability is {round(df.Reliability.mean(), 3)}, with a standard deviation of {roun
print(f"The mean overall score for Options is {round(df.Options.mean(), 3)}, with a standard deviation of {round(df.Opt
print(f"The mean overall score for Hrs_treat is {round(df.Hrs_treat.mean(), 3)}, with a standard deviation of {round(df
print(f"The mean overall score for Courteous is  {round(df.Courteous.mean(), 3)}, with a standard deviation of {round(d
print(f"The mean overall score for Active_listen is  {round(df.Active_listen.mean(), 3)}, with a standard deviation of
```

```
The mean overall score for Timely_admis is 3.519, with a standard deviation of 1.032.
The mean overall score for Timely_treat is 3.507, with a standard deviation of 1.035.
The mean overall score for Timely_visits is 3.511, with a standard deviation of 1.033.
The mean overall score for Reliability is 3.515, with a standard deviation of 1.036.
The mean overall score for Options is 3.497, with a standard deviation of 1.03.
The mean overall score for Hrs_treat is 3.522, with a standard deviation of 1.032.
The mean overall score for Courteous is  3.494, with a standard deviation of 1.021.
The mean overall score for Active_listen is  3.51, with a standard deviation of 1.042.
```

In [7]:
```
#now that we have our final dataframe, save and export this dataframe as a CSV file
df.to_csv(r'C:\Users\fahim\Documents\0_WGUDocuments\d212\1medical_clean-PREPAREDTASK1d212.csv', index=False)
```

In [8]:
```
#use the ward method linkage() function to implement hierarchical clustering
distance_matrix_ward = linkage(df[["Timely_admis", "Timely_treat", "Timely_visits",
                                   "Reliability", "Options", "Hrs_treat",
                                   "Courteous", "Active_listen"]], method = 'ward', metric = 'euclidean')
# now that our linkage has been established, generate a dendrogram
plt.figure(figsize = [16,5])
dendrogram_ward = dendrogram(distance_matrix_ward)
plt.xlabel("Patient Responses to Survey")
plt.ylabel("Distance Between Clusters");
plt.show()
```

```
#assign cluster labels to the dendrogram
df['ward_cluster_labels'] = fcluster(distance_matrix_ward, 2, criterion='maxclust')
print(df['ward_cluster_labels'].value_counts().sort_index())
```



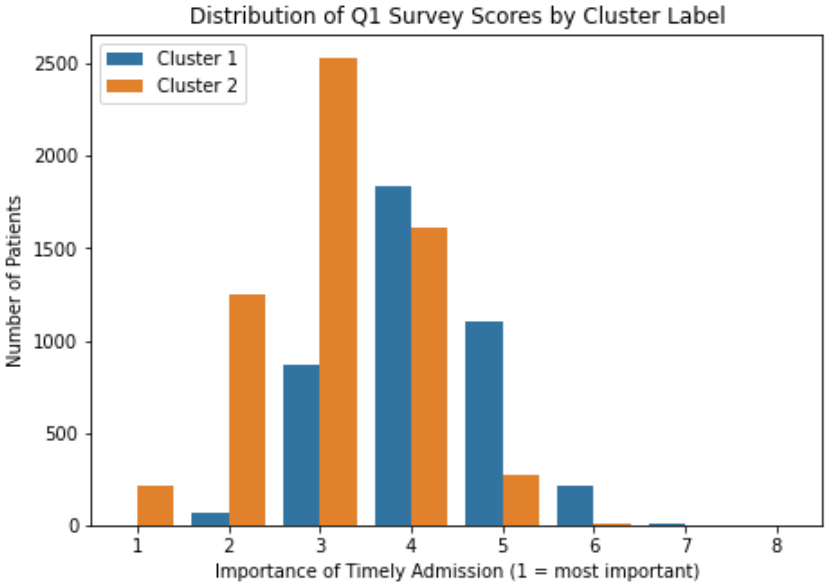

```
1    4105
2    5895
Name: ward_cluster_labels, dtype: int64
```

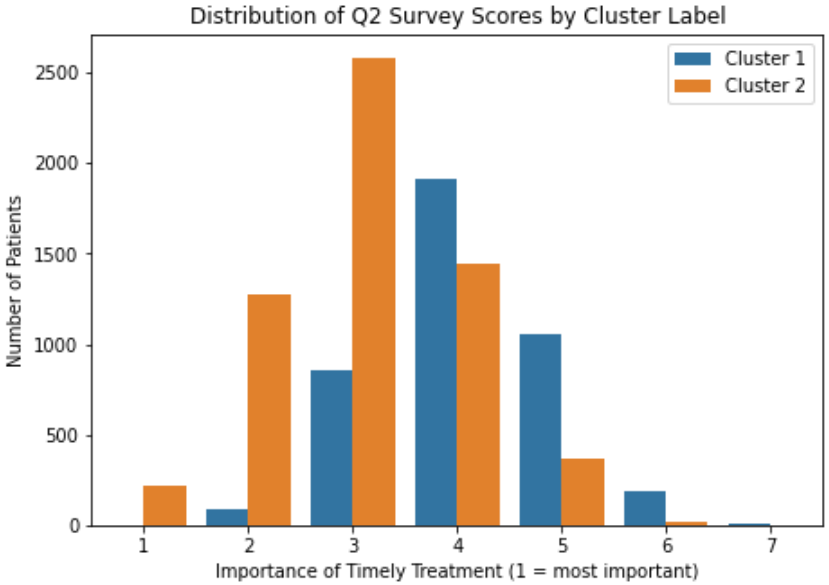
In [9]:

```
#plot the distribution scores for survey questions 1 and 2
plt.figure(figsize = [16,5])

# LEFT plot: Distribution of scores for survey question 1, by cluster label
plt.subplot(1, 2, 1)
plt.title('Distribution of Q1 Survey Scores by Cluster Label')
sns.countplot(data = df, x="Timely_admis", hue="ward_cluster_labels")
plt.legend(["Cluster 1", "Cluster 2"])
plt.xlabel("Importance of Timely Admission (1 = most important)")
plt.ylabel("Number of Patients");

# RIGHT plot: Distribution of scores for survey question 2, by cluster label
plt.subplot(1, 2, 2)
plt.title("Distribution of Q2 Survey Scores by Cluster Label")
sns.countplot(data = df, x="Timely_treat", hue="ward_cluster_labels")
plt.legend(["Cluster 1", "Cluster 2"])
plt.xlabel("Importance of Timely Treatment (1 = most important)")
plt.ylabel("Number of Patients");
```

```
q1_c1_mean = df.loc[df['ward_cluster_labels'] == 1, 'Timely_admis'].mean()
q1_c2_mean = df.loc[df['ward_cluster_labels'] == 2, 'Timely_admis'].mean()
print(f"\nFor Importance of Timely Admission respondents from Cluster 1 scored this at {round(q1_c1_mean, 3)}, on avera
print(f"Respondents from Cluster 2 scored this at {round(q1_c2_mean, 3)}, on average.")
q2_c1_mean = df.loc[df['ward_cluster_labels'] == 1, 'Timely_treat'].mean()
q2_c2_mean = df.loc[df['ward_cluster_labels'] == 2, 'Timely_treat'].mean()
print(f"\nFor Importance of Timely Treatment, respondents from Cluster 1 scored this at {round(q2_c1_mean, 3)}, on aver
print(f"Respondents from Cluster 2 scored this at {round(q2_c2_mean, 3)}, on average.")
```

```
For Importance of Timely Admission respondents from Cluster 1 scored this at 4.135, on average.
Respondents from Cluster 2 scored this at 3.09, on average.

For Importance of Timely Treatment, respondents from Cluster 1 scored this at 4.105, on average.
Respondents from Cluster 2 scored this at 3.09, on average.
```



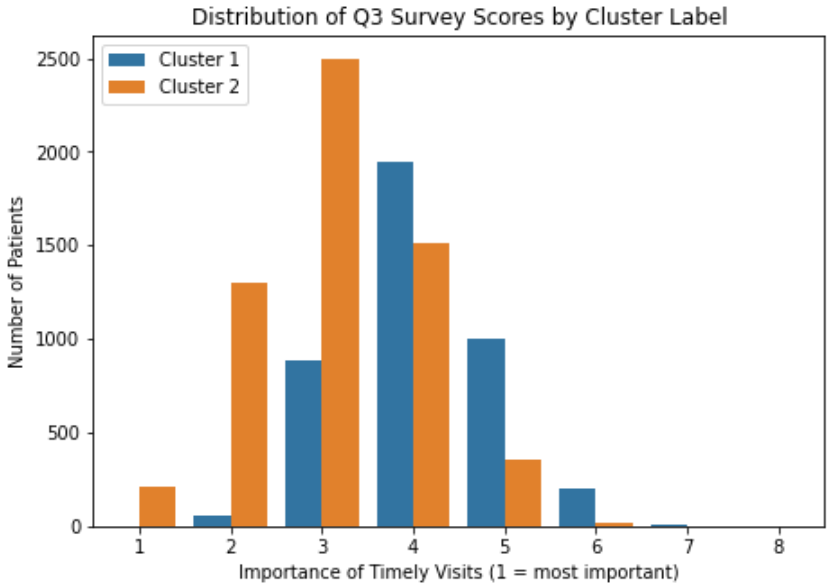


In [10]:

```
#plot the distribution scores for survey questions 3 and 4
plt.figure(figsize = [16,5])

# LEFT plot: Distribution of scores for survey question 3, by cluster label
plt.subplot(1, 2, 1)
plt.title('Distribution of Q3 Survey Scores by Cluster Label')
sns.countplot(data = df, x="Timely_visits", hue="ward_cluster_labels")
plt.legend(["Cluster 1", "Cluster 2"])
plt.xlabel("Importance of Timely Visits (1 = most important)")
plt.ylabel("Number of Patients");

# RIGHT plot: Distribution of scores for survey question 4, by cluster label
```

```
plt.subplot(1, 2, 2)
plt.title("Distribution of Q4 Survey Scores by Cluster Label")
sns.countplot(data = df, x="Reliability", hue="ward_cluster_labels")
plt.legend(["Cluster 1", "Cluster 2"])
plt.xlabel("Importance of Reliability (8 = most important)")
plt.ylabel("Number of Patients");

q3_c1_mean = df.loc[df['ward_cluster_labels'] == 1, 'Timely_visits'].mean()
q3_c2_mean = df.loc[df['ward_cluster_labels'] == 2, 'Timely_visits'].mean()
print(f"\nFor Importance of Timely Visits, respondents from Cluster 1 scored this at {round(q3_c1_mean, 3)}, on average
print(f"Respondents from Cluster 2 scored this at {round(q3_c2_mean, 3)}, on average.")
q4_c1_mean = df.loc[df['ward_cluster_labels'] == 1, 'Reliability'].mean()
q4_c2_mean = df.loc[df['ward_cluster_labels'] == 2, 'Reliability'].mean()
print(f"\nFor Importance of Reliability, respondents from Cluster 1 scored this at {round(q4_c1_mean, 3)}, on average."
print(f"Respondents from Cluster 2 scored this at {round(q4_c2_mean, 3)}, on average.")
```

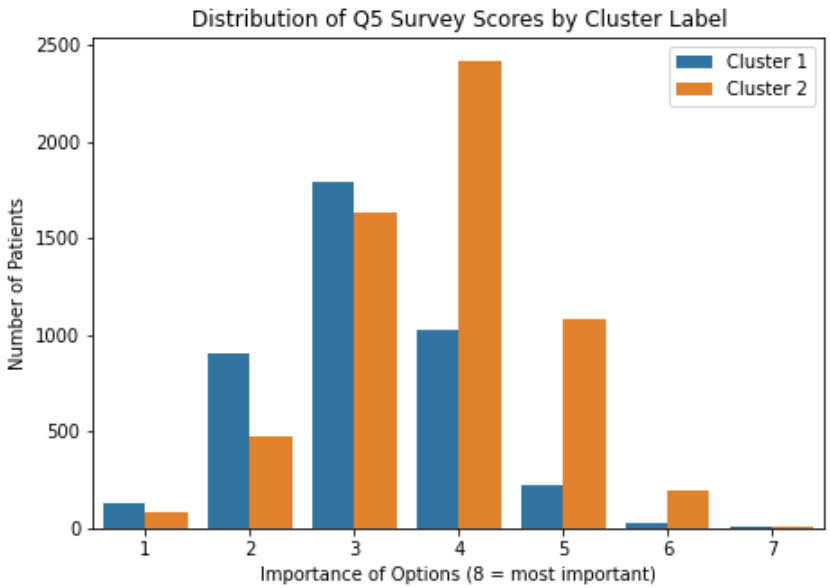
```
For Importance of Timely Visits, respondents from Cluster 1 scored this at 4.11, on average.
Respondents from Cluster 2 scored this at 3.094, on average.

For Importance of Reliability, respondents from Cluster 1 scored this at 3.663, on average.
Respondents from Cluster 2 scored this at 3.412, on average.
```

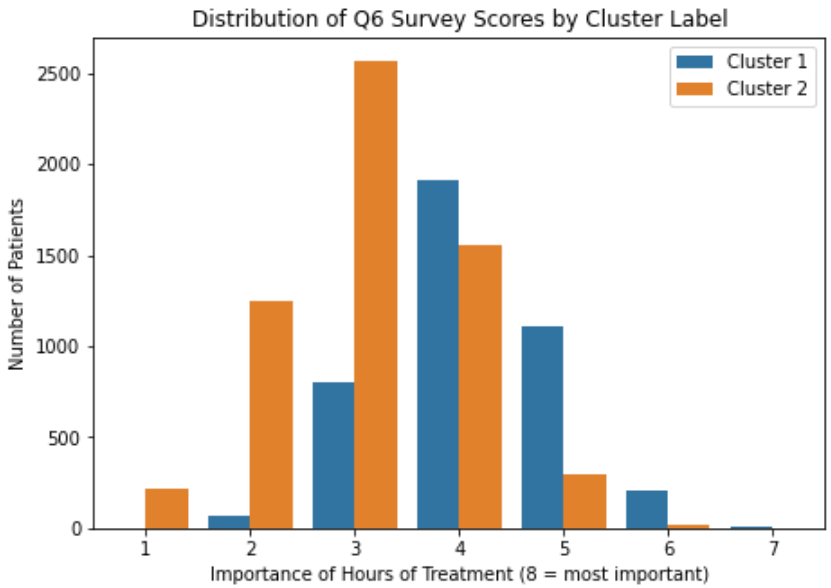


In [11]:
```
#plot the distribution scores for survey questions 5 and 6
plt.figure(figsize = [16,5])

# LEFT plot: Distribution of scores for survey question 5, by cluster label
plt.subplot(1, 2, 1)
```

```
plt.title('Distribution of Q5 Survey Scores by Cluster Label')
sns.countplot(data = df, x="Options", hue="ward_cluster_labels")
plt.legend(["Cluster 1", "Cluster 2"])
plt.xlabel("Importance of Options (8 = most important)")
plt.ylabel("Number of Patients");

# RIGHT plot: Distribution of scores for survey question 6, by cluster label
plt.subplot(1, 2, 2)
plt.title("Distribution of Q6 Survey Scores by Cluster Label")
sns.countplot(data = df, x="Hrs_treat", hue="ward_cluster_labels")
plt.legend(["Cluster 1", "Cluster 2"])
plt.xlabel("Importance of Hours of Treatment (8 = most important)")
plt.ylabel("Number of Patients");

q5_c1_mean = df.loc[df['ward_cluster_labels'] == 1, 'Options'].mean()
q5_c2_mean = df.loc[df['ward_cluster_labels'] == 2, 'Options'].mean()
print(f"\nFor Importance of Options, respondents from Cluster 1 scored this at {round(q5_c1_mean, 3)}, on average.")
print(f"Respondents from Cluster 2 scored this at {round(q5_c2_mean, 3)}, on average.")
q6_c1_mean = df.loc[df['ward_cluster_labels'] == 1, 'Hrs_treat'].mean()
q6_c2_mean = df.loc[df['ward_cluster_labels'] == 2, 'Hrs_treat'].mean()
print(f"\nFor Importance of Hours of Treatment, respondents from Cluster 1 scored this at {round(q6_c1_mean, 3)}, on av
print(f"Respondents from Cluster 2 scored this at {round(q6_c2_mean, 3)}, on average.")
```

```
For Importance of Options, respondents from Cluster 1 scored this at 3.095, on average.
Respondents from Cluster 2 scored this at 3.776, on average.

For Importance of Hours of Treatment, respondents from Cluster 1 scored this at 4.146, on average.
Respondents from Cluster 2 scored this at 3.088, on average.
```

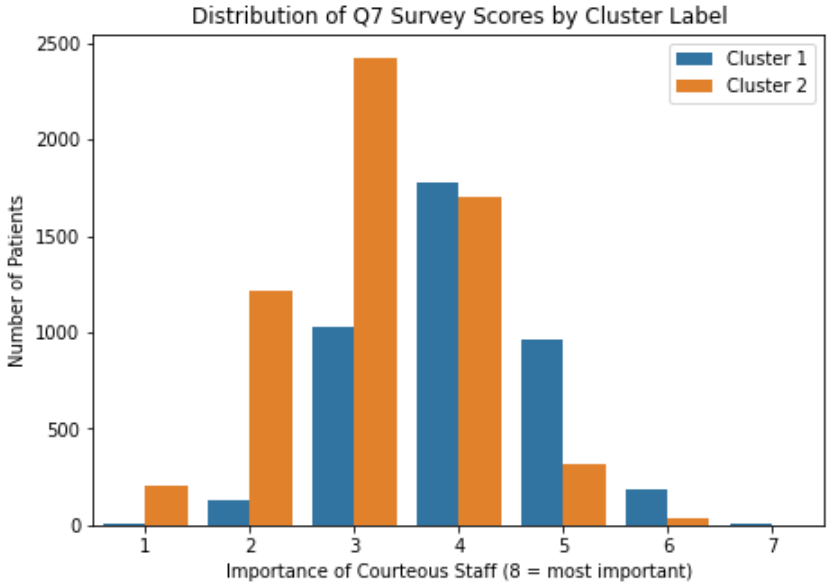


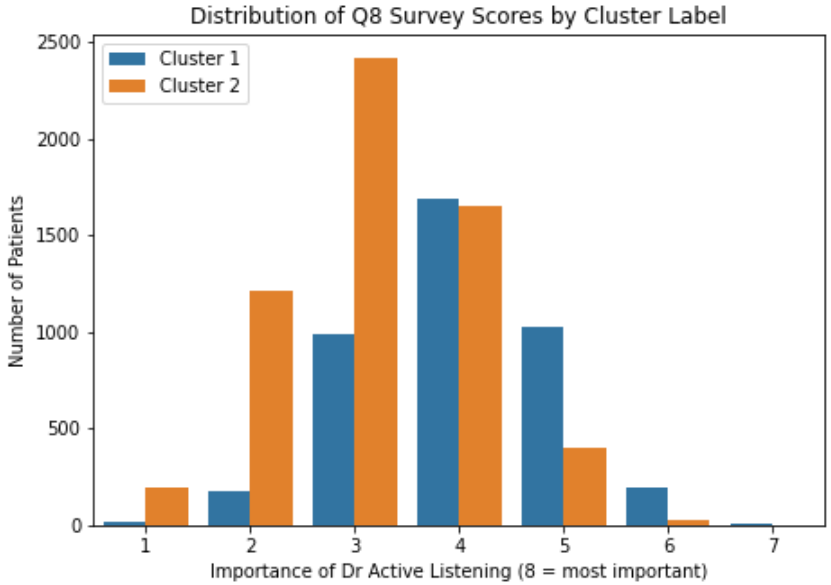


In [12]:
```
#plot the distribution scores for survey questions 7 and 8
plt.figure(figsize = [16,5])

# LEFT plot: Distribution of scores for survey question 7, by cluster label
plt.subplot(1, 2, 1)
plt.title('Distribution of Q7 Survey Scores by Cluster Label')
sns.countplot(data = df, x="Courteous", hue="ward_cluster_labels")
plt.legend(["Cluster 1", "Cluster 2"])
plt.xlabel("Importance of Courteous Staff (8 = most important)")
plt.ylabel("Number of Patients");

# RIGHT plot: Distribution of scores for survey question 8, by cluster label
plt.subplot(1, 2, 2)
plt.title("Distribution of Q8 Survey Scores by Cluster Label")
sns.countplot(data = df, x="Active_listen", hue="ward_cluster_labels")
plt.legend(["Cluster 1", "Cluster 2"])
plt.xlabel("Importance of Dr Active Listening (8 = most important)")
plt.ylabel("Number of Patients");

q7_c1_mean = df.loc[df['ward_cluster_labels'] == 1, 'Courteous'].mean()
q7_c2_mean = df.loc[df['ward_cluster_labels'] == 2, 'Courteous'].mean()
print(f"\nFor Importance of Courteous Staff, respondents from Cluster 1 scored this at {round(q7_c1_mean, 3)}, on avera
print(f"Respondents from Cluster 2 scored this at {round(q7_c2_mean, 3)}, on average.")
q8_c1_mean = df.loc[df['ward_cluster_labels'] == 1, 'Active_listen'].mean()
q8_c2_mean = df.loc[df['ward_cluster_labels'] == 2, 'Active_listen'].mean()
```

```
print(f"\nFor Importance of Dr Active Listening, respondents from Cluster 1 scored this at {round(q8_c1_mean, 3)}, on a
print(f"Respondents from Cluster 2 scored this at {round(q8_c2_mean, 3)}, on average.")
```

```
For Importance of Courteous Staff, respondents from Cluster 1 scored this at 4.009, on average.
Respondents from Cluster 2 scored this at 3.136, on average.

For Importance of Dr Active Listening, respondents from Cluster 1 scored this at 4.015, on average.
Respondents from Cluster 2 scored this at 3.158, on average.
```





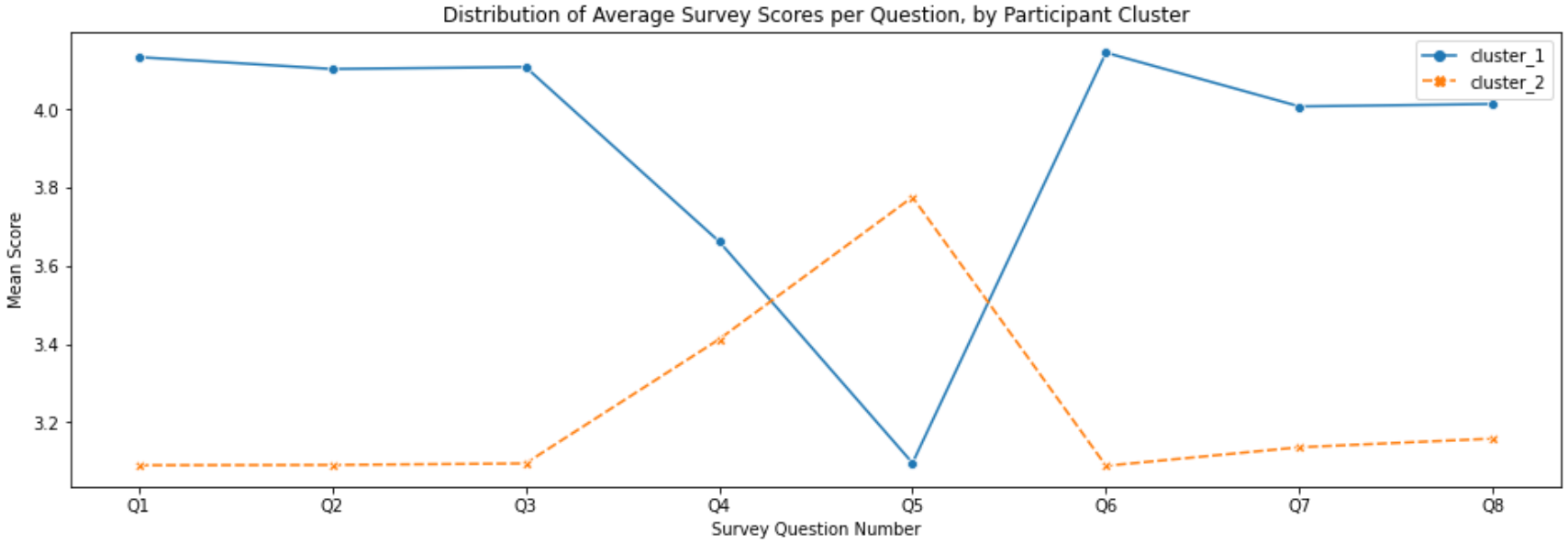
In [13]:
```
#print out the summary of the means for each question, by cluster
summary_dict = {'cluster_1' : [q1_c1_mean, q2_c1_mean, q3_c1_mean, q4_c1_mean, q5_c1_mean, q6_c1_mean, q7_c1_mean, q8_c
summary_df = pd.DataFrame(data = summary_dict, index=['Q1', 'Q2', 'Q3', 'Q4', 'Q5', 'Q6', 'Q7', 'Q8'])
print("A summary of the means for each question, by participant cluster, can be seen here:")
summary_df
```

```
A summary of the means for each question, by participant cluster, can be seen here:
```

Out[13]:

	cluster_1	cluster_2
Q1	4.134957	3.089737
Q2	4.104507	3.090416
Q3	4.109622	3.094317
Q4	3.663094	3.412044
Q5	3.095493	3.776421
Q6	4.146163	3.088210
Q7	4.008526	3.135708
Q8	4.014860	3.157930

In [14]:
```
#create a line graph plotting the distribution of average survey scores per question, by participant cluster
plt.figure(figsize = [16,5])
sns.lineplot(data = summary_df, markers=True)
plt.title("Distribution of Average Survey Scores per Question, by Participant Cluster")
plt.xlabel("Survey Question Number")
plt.ylabel("Mean Score");
```



In [15]:
```
# define and X (feature columns) and y (resulting cluster labels)
X = df[["Timely_admis", "Timely_treat", "Timely_visits",
                                          "Reliability", "Options", "Hrs_treat",
                                          "Courteous", "Active_listen"]]
y = df['ward_cluster_labels']
# now that all the visualizations have been made, generate a silhouette score to complete the analysis
model_score = silhouette_score(X, y, metric='euclidean')
print(f"The silhouette score of this hierarchical clustering is: {round(model_score, 3)}")
```

```
The silhouette score of this hierarchical clustering is: 0.148
```