Reading data from the diabetes.csv file.

In [ ]:
```
import pandas as pd
input_data = pd.read_csv(r'..\\Inputs\\diabetes.csv')
input_data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 768 entries, 0 to 767
Data columns (total 9 columns):
 #   Column                    Non-Null Count  Dtype  
---  ------                    --------------  -----  
 0   Pregnancies               768 non-null    int64  
 1   Glucose                   768 non-null    int64  
 2   BloodPressure             768 non-null    int64  
 3   SkinThickness             768 non-null    int64  
 4   Insulin                   768 non-null    int64  
 5   BMI                       768 non-null    float64
 6   DiabetesPedigreeFunction  768 non-null    float64
 7   Age                       768 non-null    int64  
 8   Outcome                   768 non-null    int64  
dtypes: float64(2), int64(7)
memory usage: 54.1 KB
```

1. Construct a scatterplot with x-axis to be the mass variable and y-axis to be the age variable. Moreover, determine the color of the points based on the class of the candidate (0 or 1).

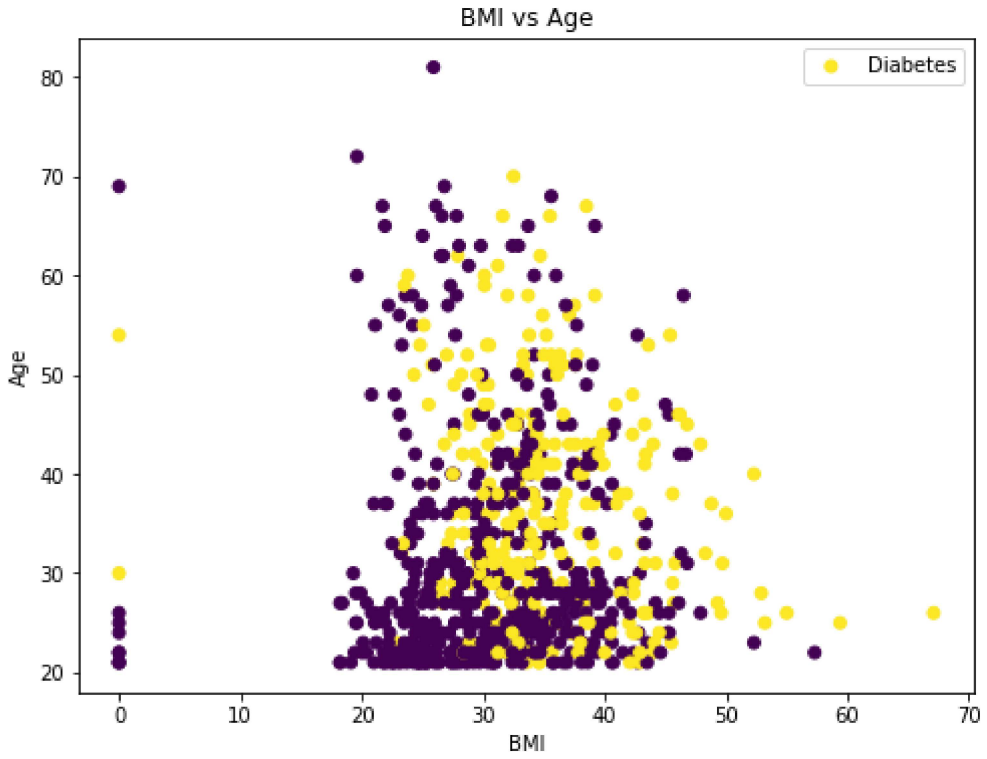
In [ ]:
```
data = input_data[['BMI','Age','Outcome']]
data.head()
```

Out[ ]:

	BMI	Age	Outcome
0	33.6	50	1
1	26.6	31	0
2	23.3	32	1
3	28.1	21	0
4	43.1	33	1

In [ ]:
```
import matplotlib.pyplot as plt
fig, ax = plt.subplots(figsize=(8,6))
ax.scatter(data['BMI'],data['Age'],c=data['Outcome'])
plt.xlabel('BMI')
plt.ylabel('Age')
plt.title('BMI vs Age')
ax.legend(['Diabetes','No Diabetes'],loc='upper right')
```

Out[ ]:
```
<matplotlib.legend.Legend at 0x2b564777580>
```



2. Create a distance matrix for the data.

In [ ]:
```
import pandas as pd
from scipy.spatial import distance_matrix
pd.DataFrame(distance_matrix(data[['BMI']], data[['Age']]), index=data.index, colu
```

Out[ ]:

	0	1	2	3	4	5	6	7	8	9	...	758	759	760	761	762	7
0	16.4	2.6	1.6	12.6	0.6	3.6	7.6	4.6	19.4	20.4	...	7.6	32.4	11.6	9.4	0.6	2
1	23.4	4.4	5.4	5.6	6.4	3.4	0.6	2.4	26.4	27.4	...	0.6	39.4	4.6	16.4	6.4	3
2	26.7	7.7	8.7	2.3	9.7	6.7	2.7	5.7	29.7	30.7	...	2.7	42.7	1.3	19.7	9.7	3
3	21.9	2.9	3.9	7.1	4.9	1.9	2.1	0.9	24.9	25.9	...	2.1	37.9	6.1	14.9	4.9	3
4	6.9	12.1	11.1	22.1	10.1	13.1	17.1	14.1	9.9	10.9	...	17.1	22.9	21.1	0.1	10.1	1
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	
763	17.1	1.9	0.9	11.9	0.1	2.9	6.9	3.9	20.1	21.1	...	6.9	33.1	10.9	10.1	0.1	3
764	13.2	5.8	4.8	15.8	3.8	6.8	10.8	7.8	16.2	17.2	...	10.8	29.2	14.8	6.2	3.8	2
765	23.8	4.8	5.8	5.2	6.8	3.8	0.2	2.8	26.8	27.8	...	0.2	39.8	4.2	16.8	6.8	3
766	19.9	0.9	1.9	9.1	2.9	0.1	4.1	1.1	22.9	23.9	...	4.1	35.9	8.1	12.9	2.9	3
767	19.6	0.6	1.6	9.4	2.6	0.4	4.4	1.4	22.6	23.6	...	4.4	35.6	8.4	12.6	2.6	3

768 rows × 768 columns

3. Make a hierarchical clustering analysis using the single linkage method. Then create an object that contains only two clusters.

In [ ]:
```
from sklearn.cluster import AgglomerativeClustering
import numpy as np
cluster_single = AgglomerativeClustering(n_clusters=2, affinity='euclidean', linka
cluster_single.fit_predict(np.array(data[['BMI','Age']]))
```

Out[ ]:
```
array([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, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 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, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0],
      dtype=int64)
```

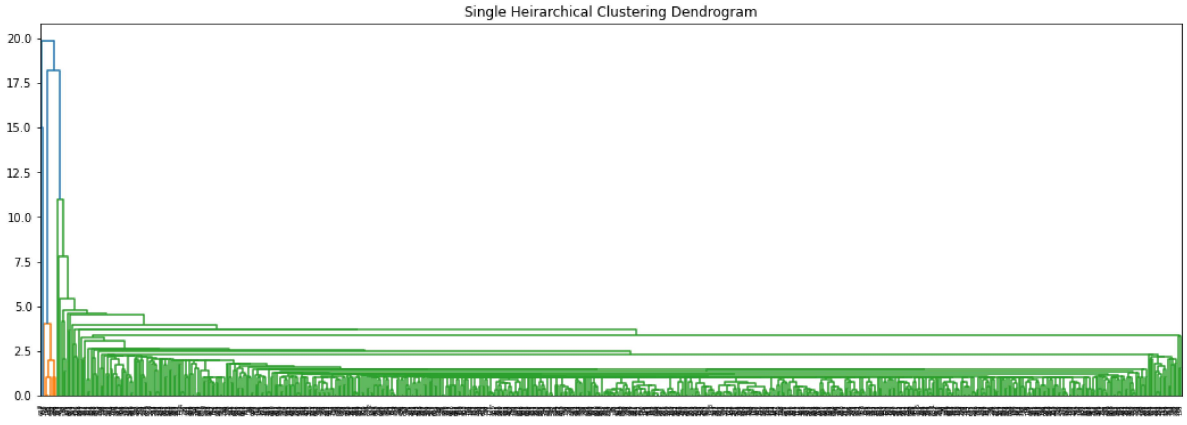
4. Make a hierarchical clustering analysis using the complete linkage method. Then create an object that contains only two clusters.

In [ ]:
```python
from sklearn.cluster import AgglomerativeClustering
import numpy as np
cluster_complete = AgglomerativeClustering(n_clusters=2, affinity='euclidean', link
cluster_complete.fit_predict(np.array(data[['BMI','Age']]))
```

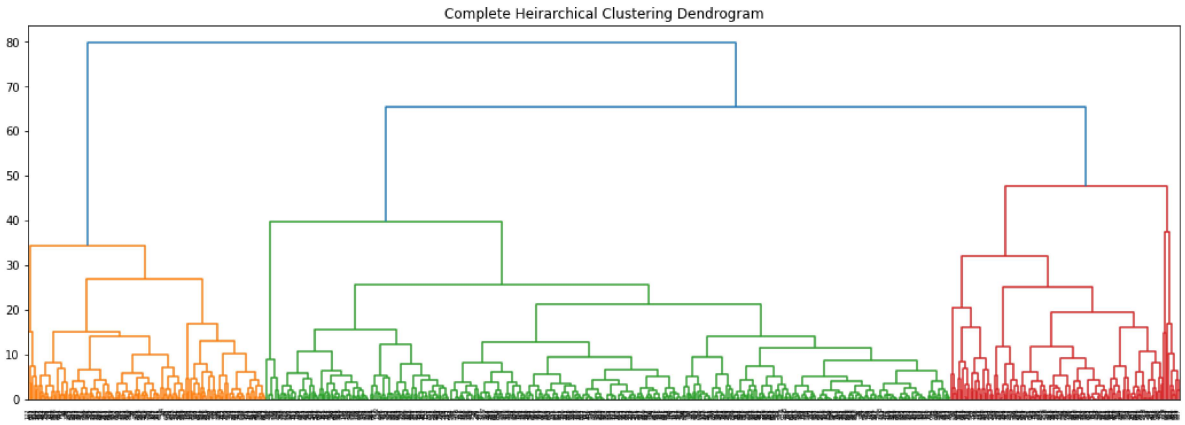
Out[ ]:
```
array([0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 1, 1, 0, 0, 0, 0, 1, 0, 1, 0, 1, 0,
       1, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 1, 0, 0,
       0, 1, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 1, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1, 1, 0, 0, 0, 0, 0, 1, 0, 0, 1,
       0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 1, 1, 0, 0, 0, 1, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1, 1, 0, 0, 0, 0, 0,
       0, 1, 0, 0, 0, 0, 0, 1, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1,
       1, 1, 0, 0, 0, 1, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0,
       0, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 1, 0, 0,
       0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 1, 0, 1, 0, 0, 0, 1,
       0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 1, 0, 0, 0, 1, 0, 1, 0, 0, 0, 0,
       0, 0, 1, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 1, 0, 0, 1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1, 1, 0, 0, 0, 0,
       1, 1, 0, 1, 0, 0, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 1, 0, 0,
       0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0,
       0, 0, 1, 0, 0, 1, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 0,
       0, 0, 1, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0,
       0, 0, 0, 0, 1, 1, 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, 0, 0, 1, 0, 0, 1, 0, 0, 0, 0, 1,
       0, 0, 1, 0, 1, 0, 1, 1, 0, 0, 1, 0, 0, 0, 0, 0, 0, 1, 1, 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, 1, 1, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 0, 1, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 1, 1, 0, 0, 0, 0, 0, 0, 1, 1, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 1, 0, 1, 0, 1, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 1, 0, 0, 1, 0, 0, 1, 0, 0, 0, 0, 0, 0, 1, 1, 0, 0, 0,
       0, 0, 1, 0, 1, 0, 0, 0, 0, 0, 0, 0, 1, 0, 1, 0, 0, 0, 1, 0, 0, 0, 0,
       0, 0, 0, 0, 1, 0, 1, 1, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       1, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 1, 0, 0, 0,
       0, 1, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 1,
       1, 0, 0, 0, 0, 0, 0, 1, 0, 1, 1, 1, 0, 0, 0, 0, 0, 1, 0, 0, 0, 1,
       0, 1, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 1,
       0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 1, 1, 0, 0, 0, 1, 0, 1, 1,
       0, 0, 0, 1, 0, 1, 0, 0, 0, 0, 1, 0, 0, 1, 0, 0, 1, 0, 0, 0],
      dtype=int64)
```

5. Construct the trees that are produced by Questions 2 and 3 and draw the two clusters (at the plots)

In [ ]:
```
import scipy.cluster.hierarchy as shc
plt.figure(figsize=(18, 6))
plt.title("Single Heirarchical Clustering Dendrogram")
dend_complete = shc.dendrogram(shc.linkage(data[['BMI','Age']], method='single'))
```



In [ ]:
```
plt.figure(figsize=(18, 6))
plt.title("Complete Heirarchical Clustering Dendrogram")
dend_complete = shc.dendrogram(shc.linkage(data[['BMI','Age']], method='complete'))
```

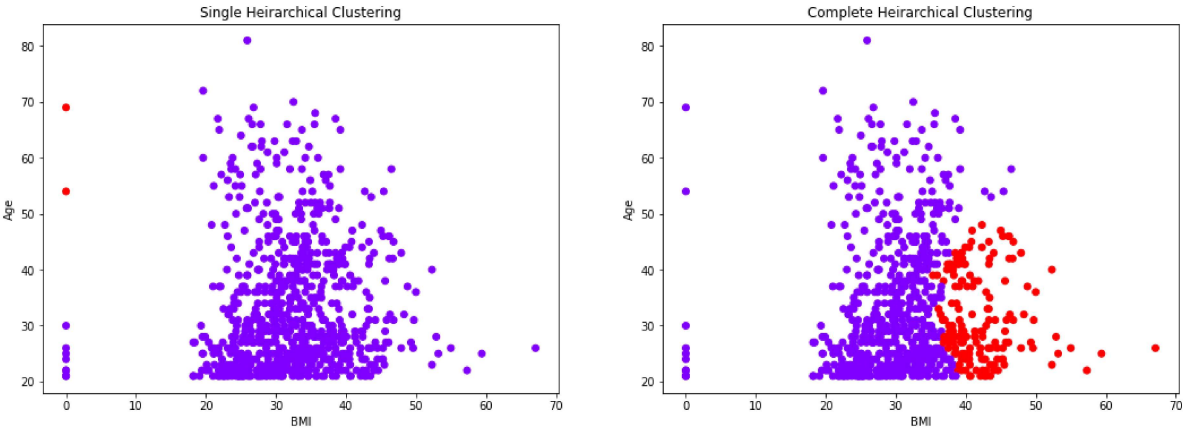


6. Construct two scatterplot with x-axis to be the mass variable and y-axis to be the age variable. Moreover, determine the color of the points based on the cluster that those points belong to. Each scatterplot is for different clustering method.

In [ ]:
```
fig, ax = plt.subplots(1,2,figsize=(18,6))
ax[0].set_xlabel('BMI')
ax[0].set_ylabel('Age')
ax[0].set_title('Single Heirarchical Clustering')
ax[0].scatter(data['BMI'],data['Age'], c=cluster_single.labels_, cmap='rainbow')

ax[1].set_xlabel('BMI')
ax[1].set_ylabel('Age')
ax[1].set_title('Complete Heirarchical Clustering')
ax[1].scatter(data['BMI'],data['Age'], c=cluster_complete.labels_, cmap='rainbow')
```

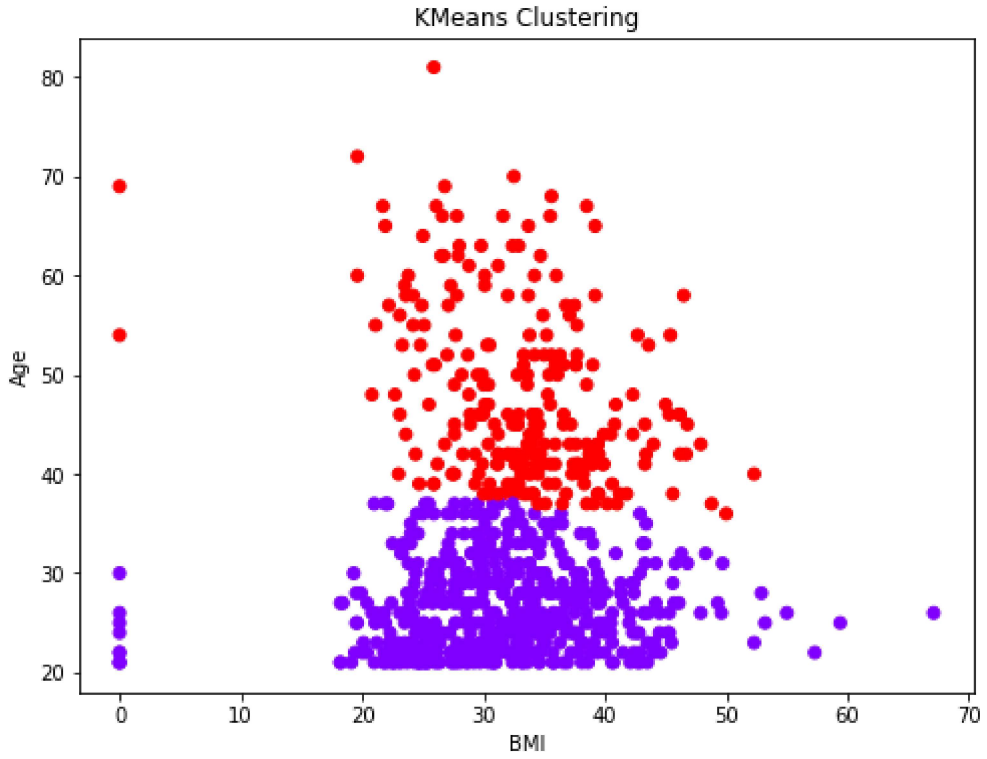
Out[ ]:
```
<matplotlib.collections.PathCollection at 0x2b56df74dc0>
```



7. Construct a scatterplot with x-axis to be the mass variable and y-axis to be the age variable. Moreover, determine the color of the points based on the cluster (retrieved from k-mean method) that those points belong to.

In [ ]:
```
from sklearn.cluster import KMeans
kmeans = KMeans(n_clusters=2, random_state=0).fit(data[['BMI','Age']])
kmeans.predict(data[['BMI','Age']])
fig, ax = plt.subplots(figsize=(8,6))
ax.set_xlabel('BMI')
ax.set_ylabel('Age')
ax.set_title('KMeans Clustering')
ax.scatter(data['BMI'],data['Age'], c=kmeans.labels_, cmap='rainbow')
```

Out[ ]:
```
<matplotlib.collections.PathCollection at 0x2b56e05c580>
```



8. Construct a scatterplot with x-axis to be the mass variable and y-axis to be the age variable. Moreover, determine the color of the points based on the cluster (retrieved from k-median method) that those points belong to.

In [ ]:
```
from sklearn_extra.cluster import KMedoids
kmediods = KMedoids(n_clusters=2, random_state=0).fit(data[['BMI','Age']])
kmediods.predict(data[['BMI','Age']])
fig, ax = plt.subplots(figsize=(8,6))
ax.set_xlabel('BMI')
ax.set_ylabel('Age')
ax.set_title('KMeans Clustering')
ax.scatter(data['BMI'],data['Age'], c=kmediods.labels_, cmap='rainbow')
```