Hierarchical Clustering On Mall Customers data

Importing the libraries

In [2]:

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import scipy.cluster.hierarchy as sch
from sklearn.cluster import AgglomerativeClustering

import warnings
warnings.filterwarnings("ignore")
%matplotlib inline
```

Importing the dataset

In [3]:

```python
dataset = pd.read_csv('C:/Users/Eric/Documents/Jupyter Notebook/practical/data/Mall_Customers.csv')
dataset.head(10)
```

Out[3]:

	CustomerID	Genre	Age	Annual Income (k$)	Spending Score (1-100)
0	1	Male	19	15	39
1	2	Male	21	15	81
2	3	Female	20	16	6
3	4	Female	23	16	77
4	5	Female	31	17	40
5	6	Female	22	17	76
6	7	Female	35	18	6
7	8	Female	23	18	94
8	9	Male	64	19	3
9	10	Female	30	19	72

In [5]:

```python
dataset.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 200 entries, 0 to 199
Data columns (total 5 columns):
 #   Column                  Non-Null Count  Dtype 
---  ------                  --------------  ----- 
 0   CustomerID              200 non-null    int64 
 1   Genre                   200 non-null    object
 2   Age                     200 non-null    int64 
 3   Annual Income (k$)      200 non-null    int64 
 4   Spending Score (1-100)  200 non-null    int64 
dtypes: int64(4), object(1)
memory usage: 7.9+ KB
```

In [6]:
```
dataset.isnull().sum()
```

Out[6]:
```
CustomerID                0
Genre                     0
Age                       0
Annual Income (k$)        0
Spending Score (1-100)    0
dtype: int64
```

In [7]:
```
dataset.describe()
```

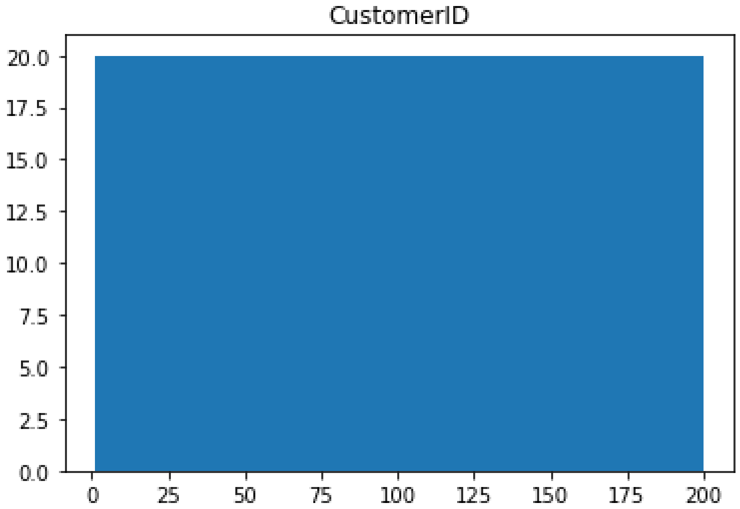
Out[7]:

	CustomerID	Age	Annual Income (k$)	Spending Score (1-100)
count	200.000000	200.000000	200.000000	200.000000
mean	100.500000	38.850000	60.560000	50.200000
std	57.879185	13.969007	26.264721	25.823522
min	1.000000	18.000000	15.000000	1.000000
25%	50.750000	28.750000	41.500000	34.750000
50%	100.500000	36.000000	61.500000	50.000000
75%	150.250000	49.000000	78.000000	73.000000
max	200.000000	70.000000	137.000000	99.000000

EDA

In [8]:

```python
# df_num = train[['Age','SibSp','Parch','Fare']]
for i in dataset.columns:
    plt.hist(dataset[i])
    plt.title(i)
    plt.show()
```



In [9]:

```python
dataset[['Genre', 'Age']].groupby(['Genre'], as_index=False).mean().sort_values(by='Age', ascending=False)
```

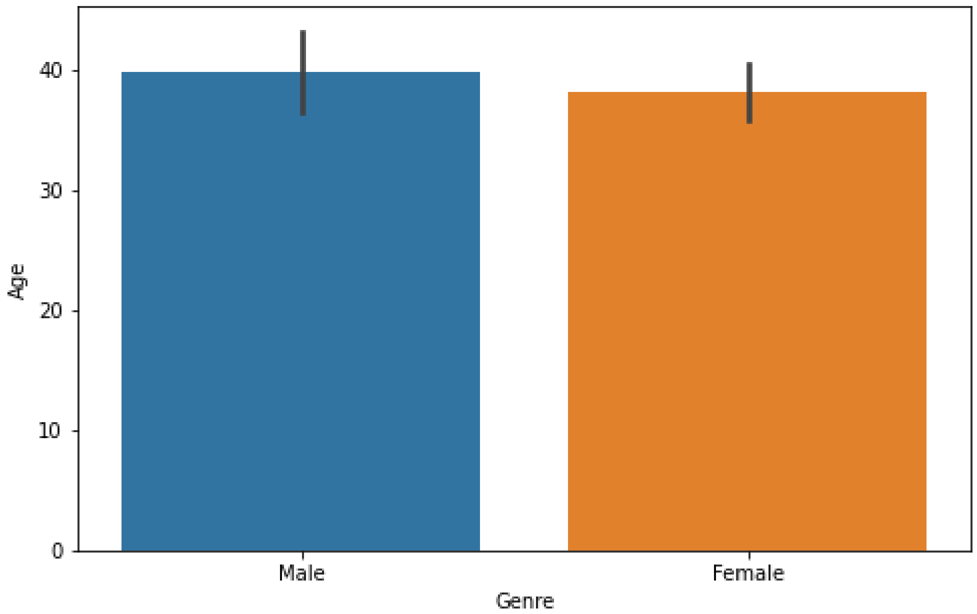
Out[9]:

	Genre	Age
1	Male	39.806818
0	Female	38.098214

In [10]:

```
plt.figure(figsize=(8, 5))
sns.barplot(x='Genre', y='Age', data=dataset)
```

Out[10]: <AxesSubplot:xlabel='Genre', ylabel='Age'>



In [11]:

```
dataset[['Genre', 'Annual Income (k$)']].groupby(['Genre'], as_index=False).mean().sort_values(by='Annual Income
```

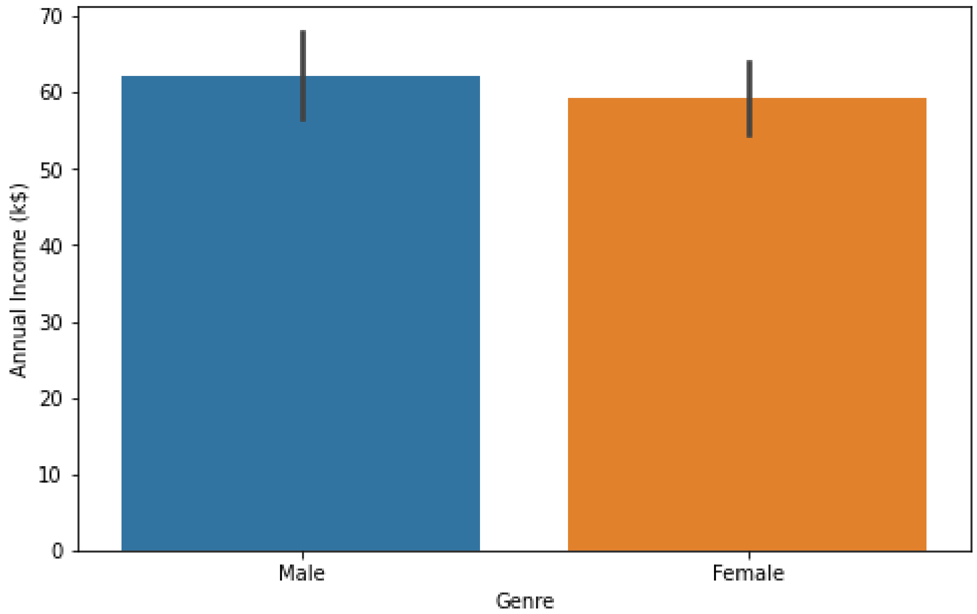
Out[11]:

	Genre	Annual Income (k$)
1	Male	62.227273
0	Female	59.250000

In [12]:

```
plt.figure(figsize=(8, 5))
sns.barplot(x='Genre', y='Annual Income (k$)', data=dataset)
```

Out[12]: <AxesSubplot:xlabel='Genre', ylabel='Annual Income (k$)'>



In [13]:

```
dataset[['Genre', 'Spending Score (1-100)']].groupby(['Genre'], as_index=False).mean().sort_values(by='Spending S
```

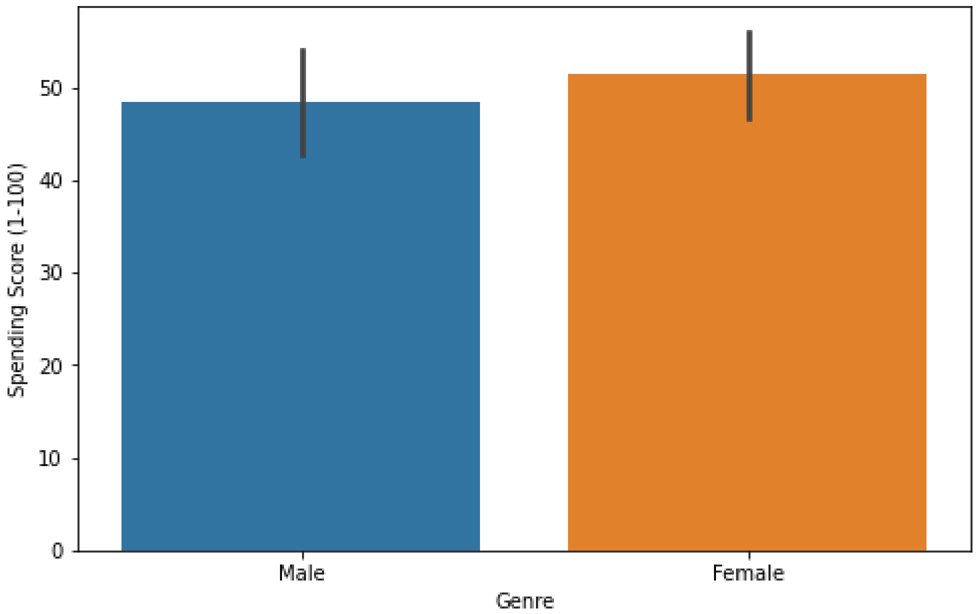
Out[13]:

	Genre	Spending Score (1-100)
0	Female	51.526786
1	Male	48.511364

In [14]:

```
plt.figure(figsize=(8, 5))
sns.barplot(x='Genre', y='Spending Score (1-100)', data=dataset)
```

Out[14]: <AxesSubplot:xlabel='Genre', ylabel='Spending Score (1-100)'>



In [15]:

```python
dataset[['Age', 'Annual Income (k$)']].groupby(['Age'], as_index=False).mean().sort_values(by='Annual Income (k$)
```

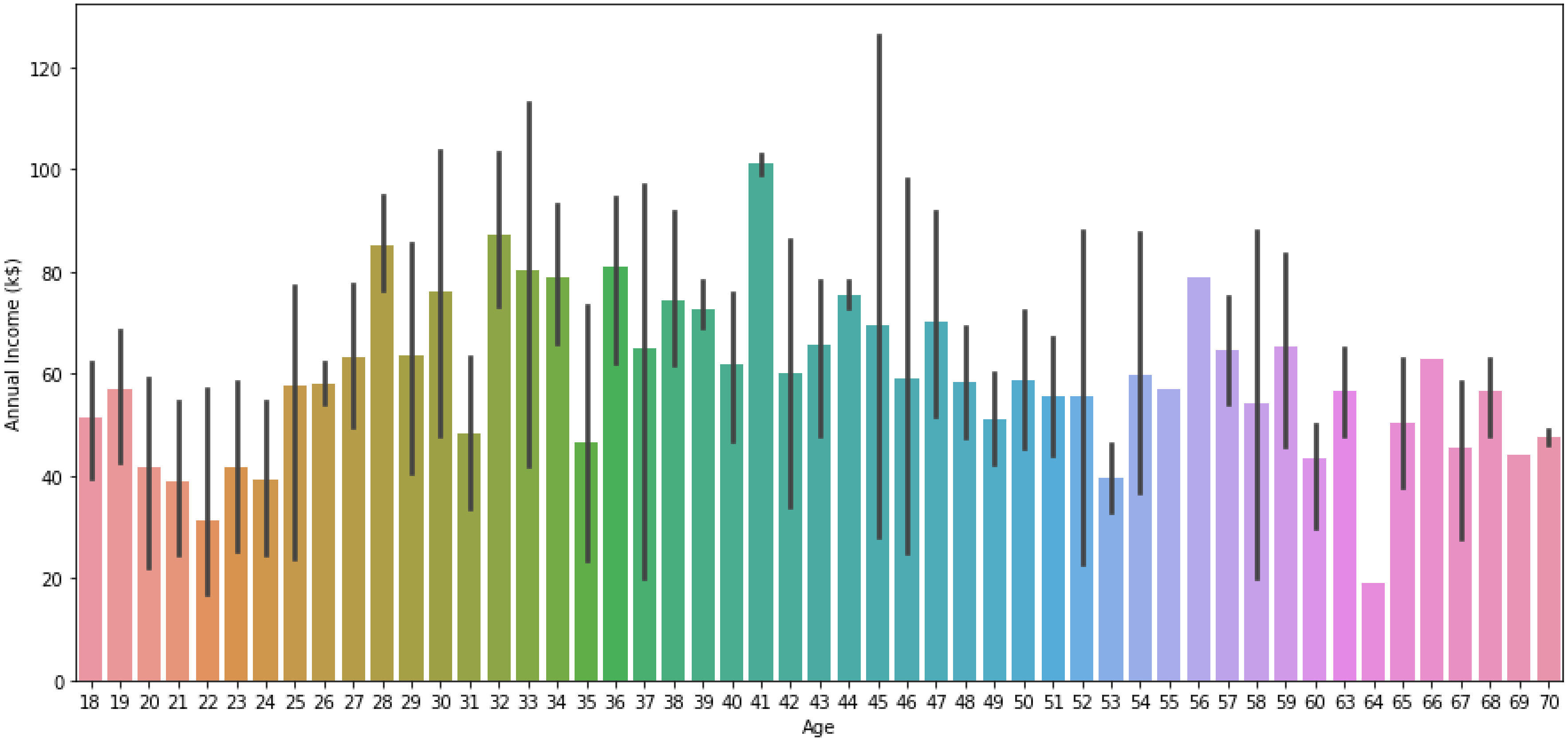
Out[15]:

	Age	Annual Income (k$)
23	41	101.000000
14	32	87.181818
10	28	85.250000
18	36	81.000000
15	33	80.333333
38	56	79.000000
16	34	79.000000
12	30	76.142857
26	44	75.500000
20	38	74.500000
21	39	72.666667

In [16]:

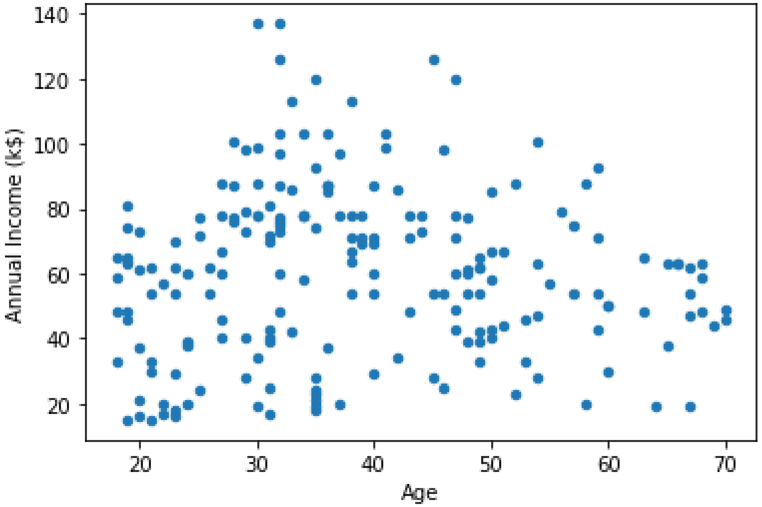
```
plt.figure(figsize=(15, 7))
sns.barplot(x='Age', y='Annual Income (k$)', data=dataset)
```

Out[16]: <AxesSubplot:xlabel='Age', ylabel='Annual Income (k$)'>



In [17]:

```
dataset.plot(kind="scatter", x="Age",    y="Annual Income (k$)")
plt.show()
```



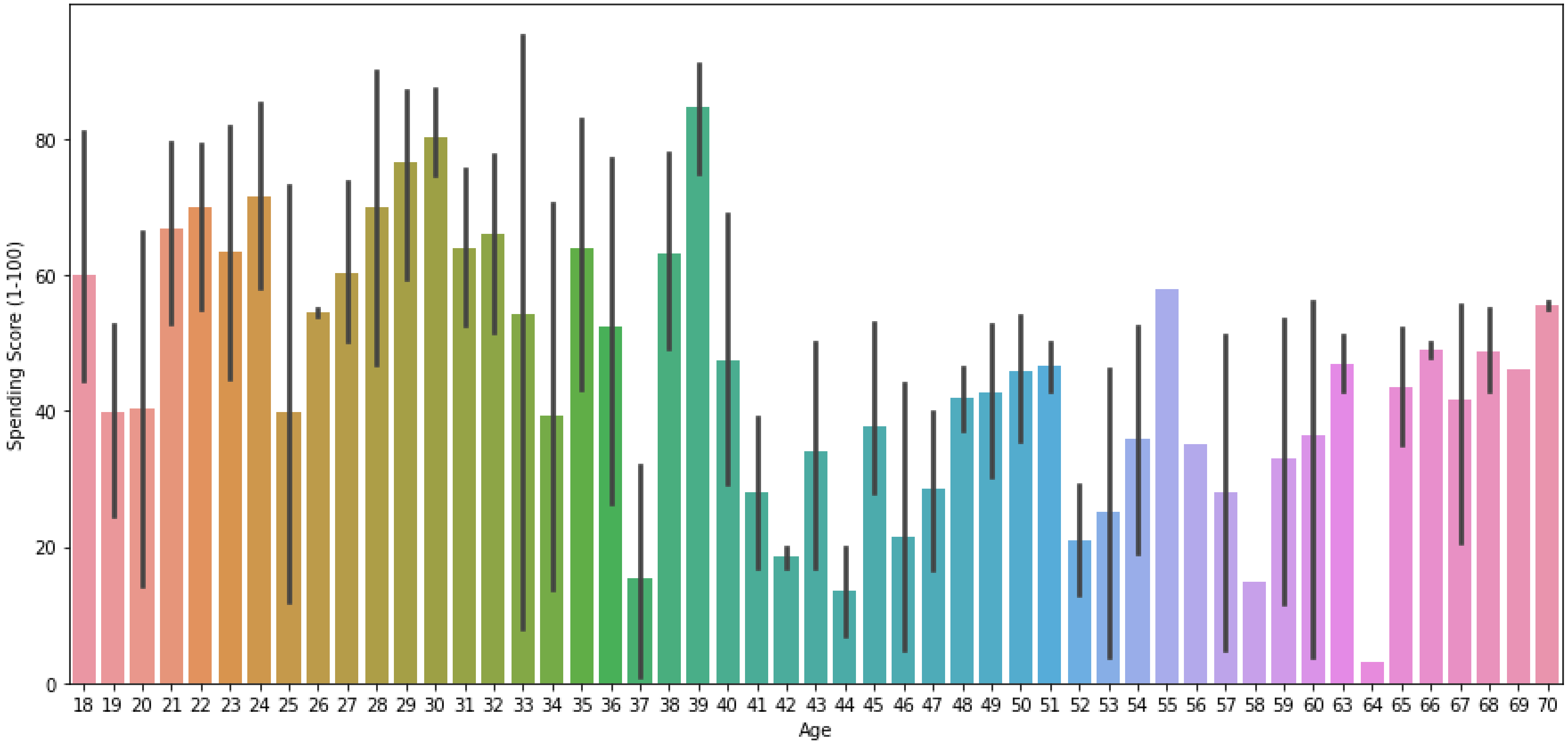
In [18]:
```
dataset[['Age', 'Spending Score (1-100)']].groupby(['Age'], as_index=False).mean().sort_values(by='Spending Score
```

Out[18]:

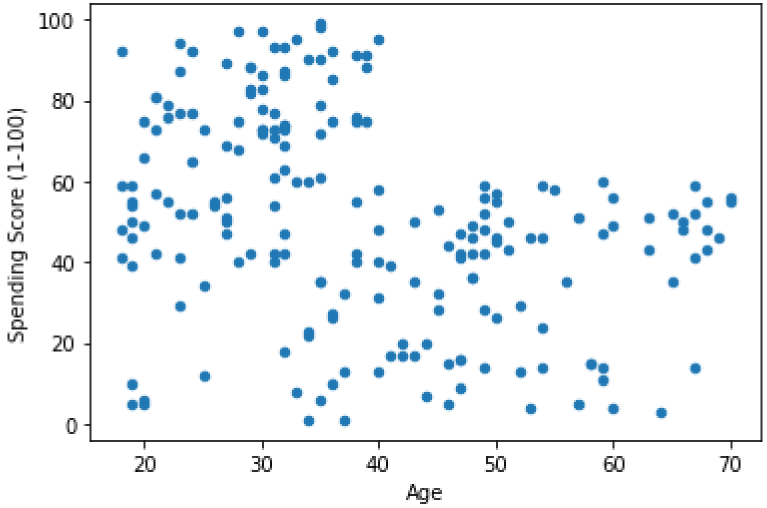
	Age	Spending Score (1-100)
21	39	84.666667
12	30	80.285714
11	29	76.600000
6	24	71.500000
4	22	70.000000
10	28	70.000000
3	21	66.800000
14	32	66.000000
17	35	63.888889
13	31	63.875000
5	23	63.333333

In [19]:
```
plt.figure(figsize=(15, 7))
sns.barplot(x='Age', y='Spending Score (1-100)', data=dataset)
```

Out[19]: <AxesSubplot:xlabel='Age', ylabel='Spending Score (1-100)'>



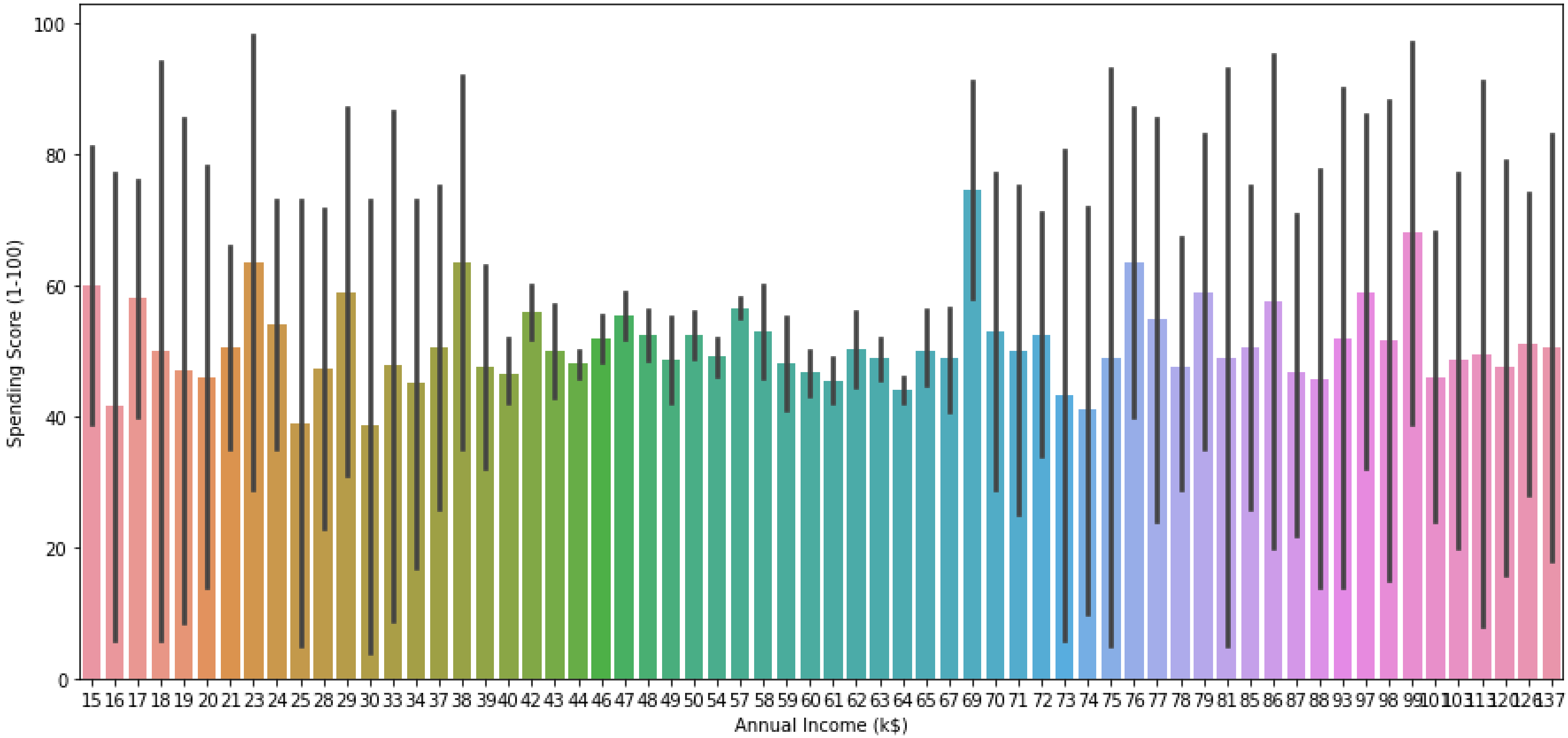
In [20]:
```
dataset.plot(kind="scatter", x="Age",    y="Spending Score (1-100)")
plt.show()
```



In [21]:

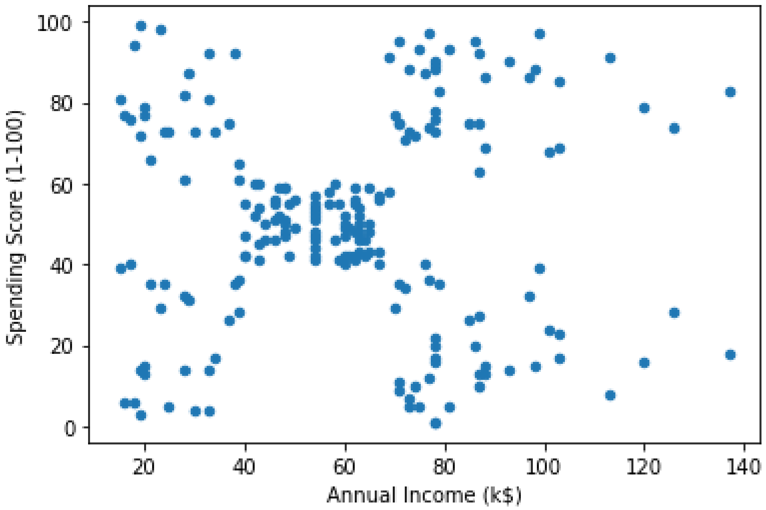
```python
plt.figure(figsize=(15, 7))
sns.barplot(x='Annual Income (k$)', y='Spending Score (1-100)', data=dataset)
```

Out[21]: <AxesSubplot:xlabel='Annual Income (k$)', ylabel='Spending Score (1-100)'>



In [22]:

```
dataset.plot(kind="scatter", x="Annual Income (k$)",    y="Spending Score (1-100)")
plt.show()
```



In [4]:

```python
# X = dataset.iloc[:, [3, 4]]
# X.head()
X = dataset.iloc[:, [3, 4]].values
X
```

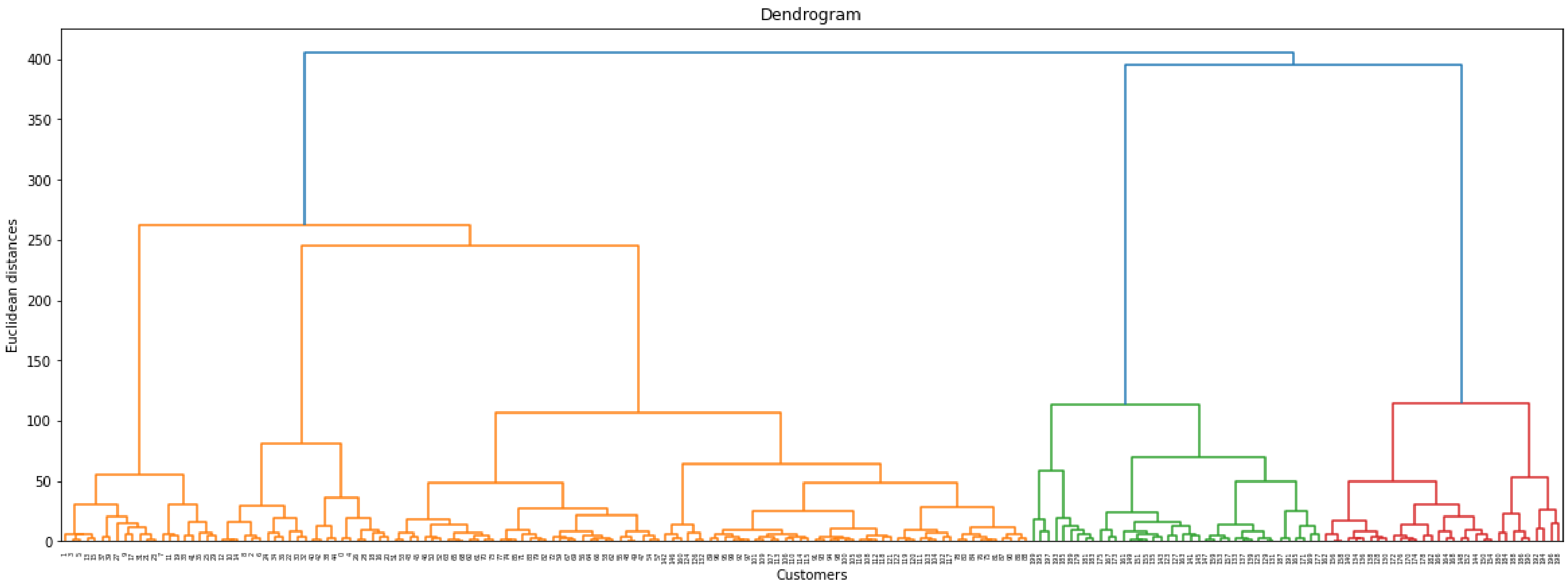
Out[4]:

```
array([[ 15,  39],
       [ 15,  81],
       [ 16,   6],
       [ 16,  77],
       [ 17,  40],
       [ 17,  76],
       [ 18,   6],
       [ 18,  94],
       [ 19,   3],
       [ 19,  72],
       [ 19,  14],
       [ 19,  99],
       [ 20,  15],
       [ 20,  77],
       [ 20,  13],
       [ 20,  79],
       [ 21,  35],
       [ 21,  66],
       [ 23,  29],
```

Using the dendrogram to find the optimal number of clusters

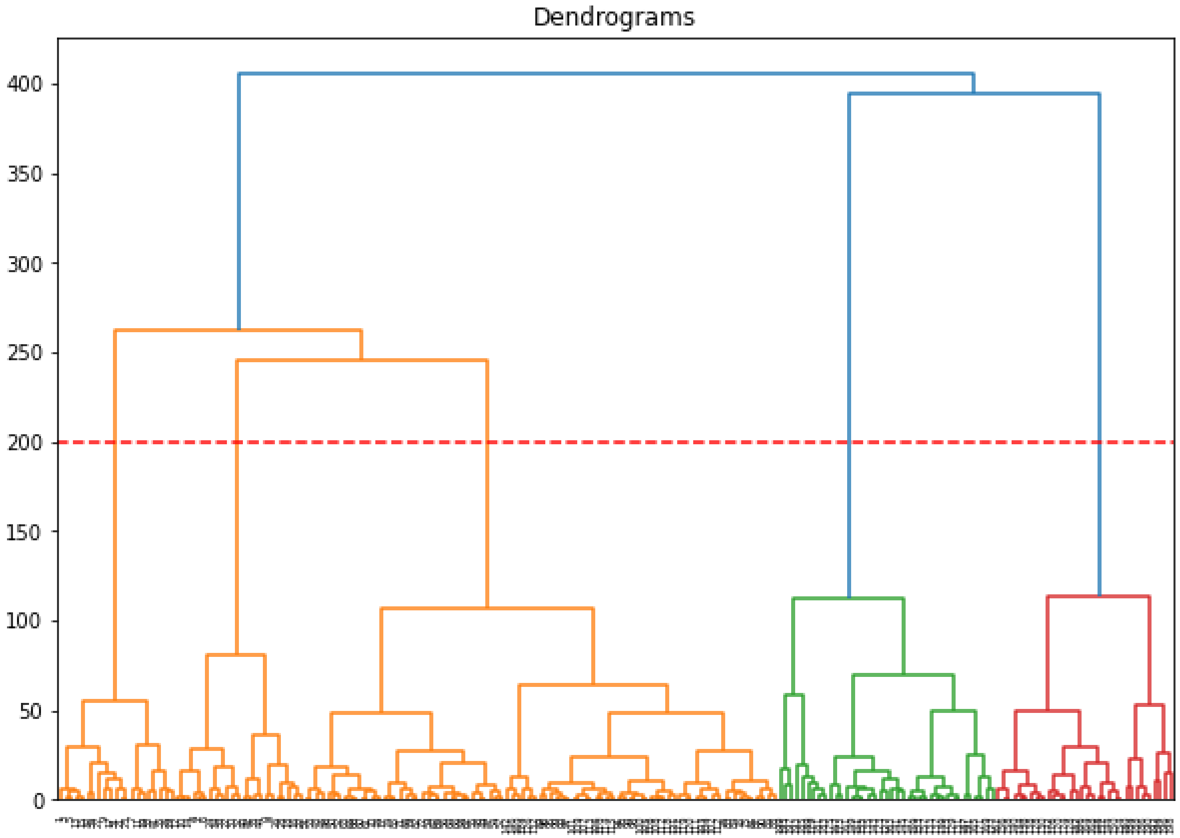
In [5]:

```python
# create dendrogram
plt.figure(figsize=(20, 7))
dendrogram = sch.dendrogram(sch.linkage(X, method = 'ward'))
plt.title('Dendrogram')
plt.xlabel('Customers')
plt.ylabel('ward distances')
plt.show()
```



In [25]:
```python
plt.figure(figsize=(10, 7))
plt.title("Dendrograms")
dendrogram = sch.dendrogram(sch.linkage(X, method = 'ward'))
plt.axhline(y=200, color='r', linestyle='--')
```

Out[25]: <matplotlib.lines.Line2D at 0x24e024f3640>



Training the Hierarchical Clustering model on the dataset

In [26]:
```python
# create clusters
hc = AgglomerativeClustering(n_clusters = 5, affinity = 'euclidean', linkage = 'ward')
y_hc = hc.fit_predict(X)
```

In [ ]:
```python

```

In [27]:
```python
print(y_hc)
```

```
[4 3 4 3 4 3 4 3 4 3 4 3 4 3 4 3 4 3 4 3 4 3 4 3 4 3 4 3 4 3 4 3 4 3 4 3 4
 3 4 3 4 3 4 1 4 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
 1 1 1 1 1 1 1 1 1 1 1 1 2 1 2 1 2 0 2 0 2 1 2 0 2 0 2 0 2 0 2 1 2 0 2 1 2
 0 2 0 2 0 2 0 2 0 2 0 2 1 2 0 2 0 2 0 2 0 2 0 2 0 2 0 2 0 2 0 2 0 2 0 2 0
 2 0 2 0 2 0 2 0 2 0 2 0 2 0 2]
```

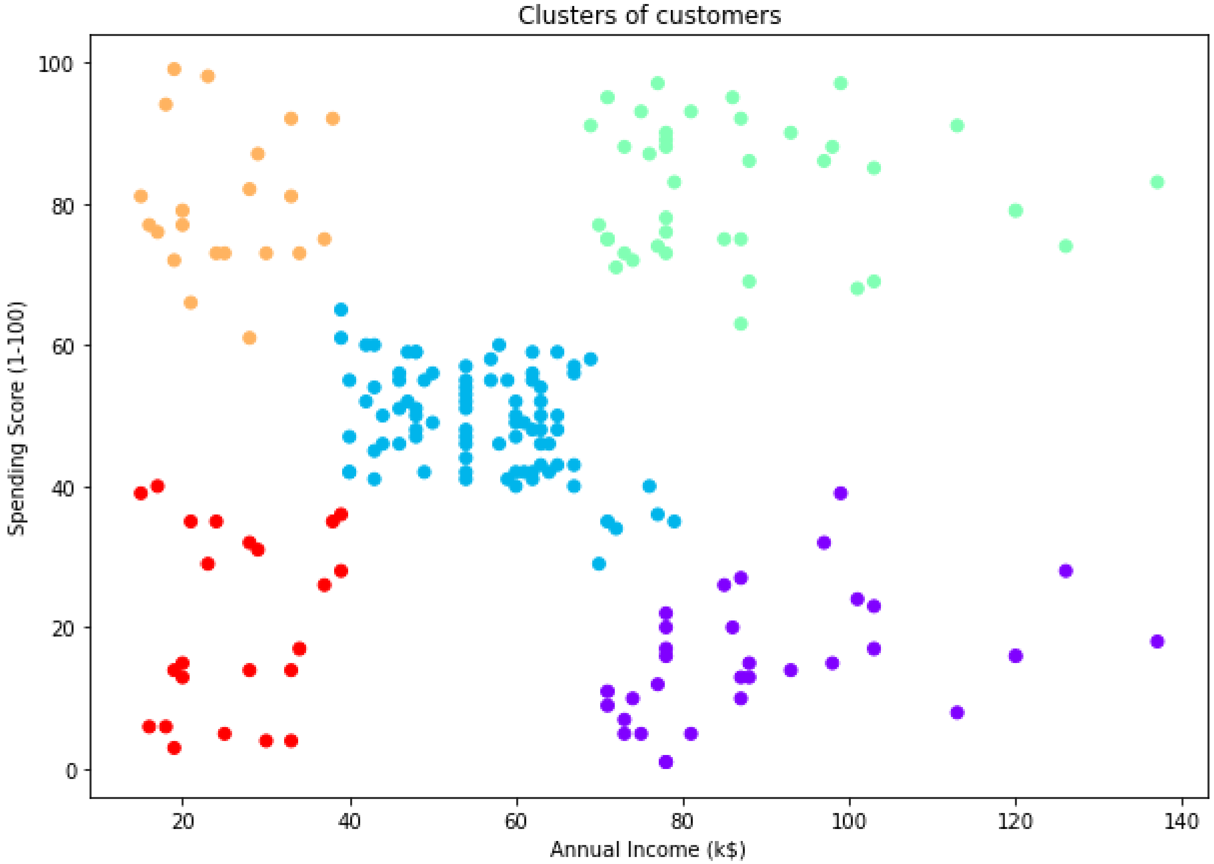
Visualising the clusters

In [28]:

```python
plt.scatter(X[y_hc == 0, 0], X[y_hc == 0, 1], s = 100, c = 'red', label = 'Cluster 1')
plt.scatter(X[y_hc == 1, 0], X[y_hc == 1, 1], s = 100, c = 'blue', label = 'Cluster 2')
plt.scatter(X[y_hc == 2, 0], X[y_hc == 2, 1], s = 100, c = 'green', label = 'Cluster 3')
plt.scatter(X[y_hc == 3, 0], X[y_hc == 3, 1], s = 100, c = 'cyan', label = 'Cluster 4')
plt.scatter(X[y_hc == 4, 0], X[y_hc == 4, 1], s = 100, c = 'magenta', label = 'Cluster 5')
plt.title('Clusters of customers')
plt.xlabel('Annual Income (k$)')
plt.ylabel('Spending Score (1-100)')
plt.legend()
plt.show()
```

In [29]:

```
plt.figure(figsize=(10, 7))
plt.scatter(X[:,0], X[:,1], c=hc.labels_, cmap='rainbow')
plt.title('Clusters of customers')
plt.xlabel('Annual Income (k$)')
plt.ylabel('Spending Score (1-100)')
plt.show()
```



In [ ]:

```

```