Importing libraries

In [1]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.preprocessing import StandardScaler
from sklearn.metrics import silhouette_score
from sklearn.cluster import KMeans
import warnings
warnings.filterwarnings('ignore')
```

Importing data

In [2]:
```
df=pd.read_csv(r"C:\Users\USER\Downloads\archive\Mall_Customers.csv")
```

In [3]:
```
df.head()
```

Out[3]:

	CustomerID	Gender	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

EDA - Exploratory data analysis

In [4]:
```
df.isnull().sum()
```

Out[4]:
```
CustomerID                0
Gender                    0
Age                       0
Annual Income (k$)        0
Spending Score (1-100)    0
dtype: int64
```

In [5]:
```
df.describe()
```

Out[5]:

	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

Now, I think we can convert Gender to numerical:

In [6]:
```
from sklearn.preprocessing import LabelEncoder
```

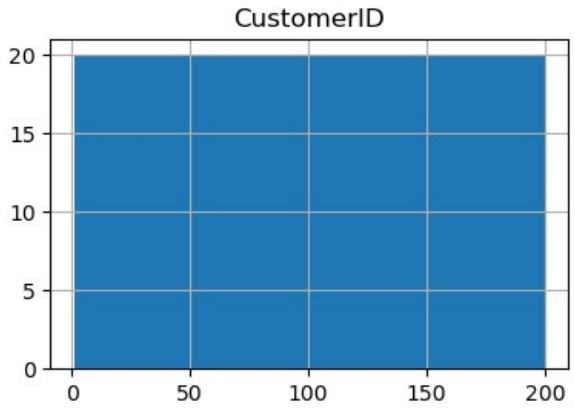
In [7]:
```
# Encode the 'Gender' column
label_encoder = LabelEncoder()
df['Gender'] = label_encoder.fit_transform(df['Gender'])
```

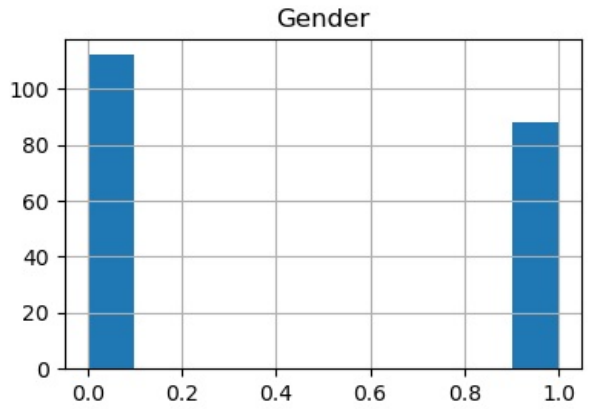
In [8]:
```
df.head()                # 1 is male, 0 is feamle
```

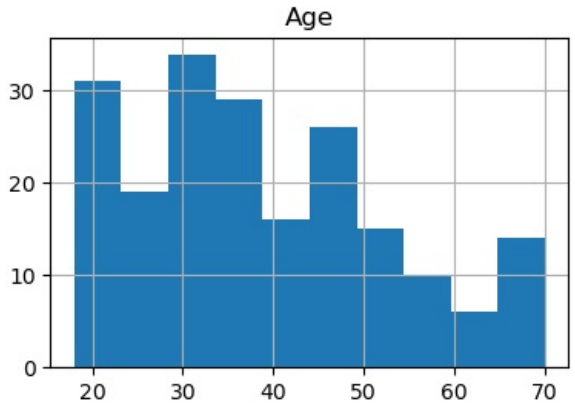
Out[8]:

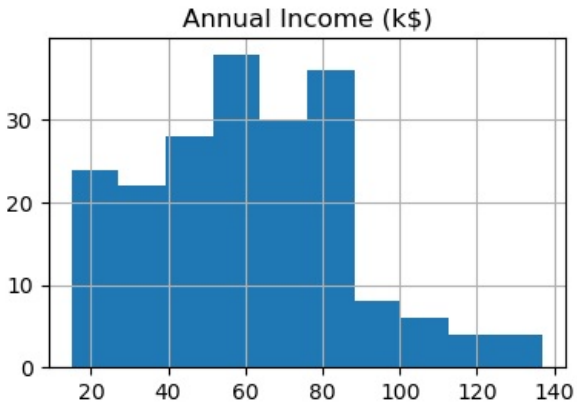
	CustomerID	Gender	Age	Annual Income (k$)	Spending Score (1-100)
0	1	1	19	15	39
1	2	1	21	15	81
2	3	0	20	16	6
3	4	0	23	16	77
4	5	0	31	17	40

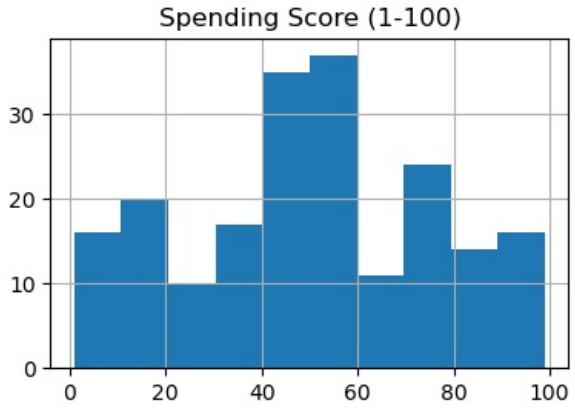
In [9]:
```
df.hist(figsize=(10,10))
plt.show()
```





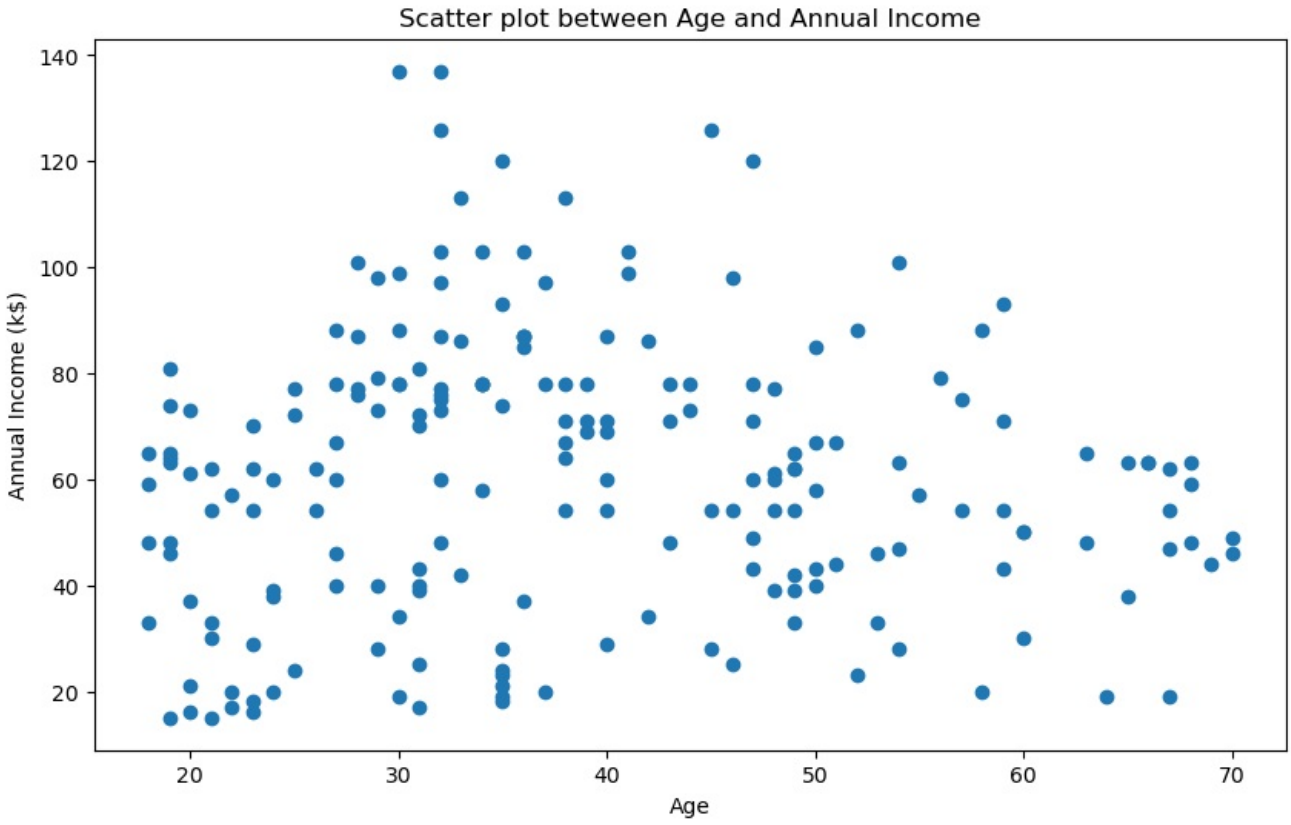






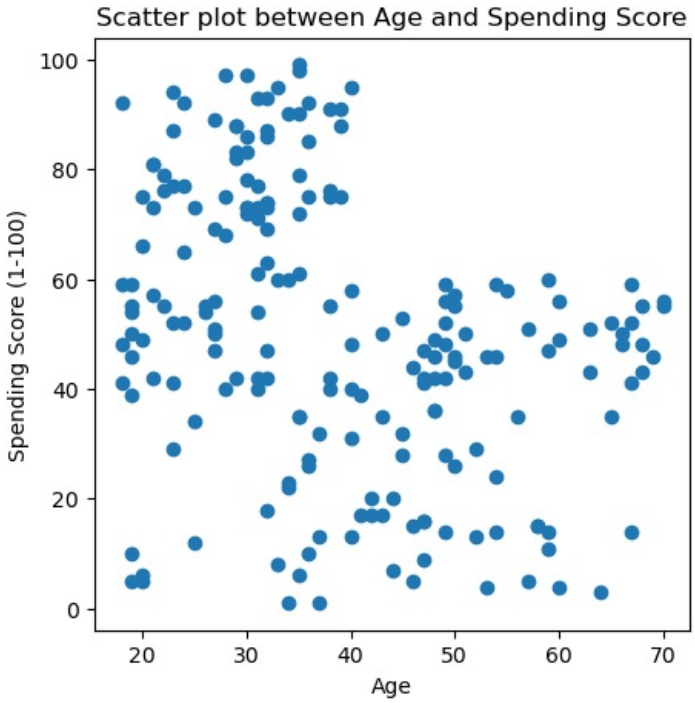
In [10]:
```
plt.figure(figsize=(10,6))
plt.scatter(df['Age'],df['Annual Income (k$)']);
plt.xlabel('Age')
plt.ylabel('Annual Income (k$)')
plt.title('Scatter plot between Age and Annual Income')
```

Out[10]:
```
Text(0.5, 1.0, 'Scatter plot between Age and Annual Income')
```



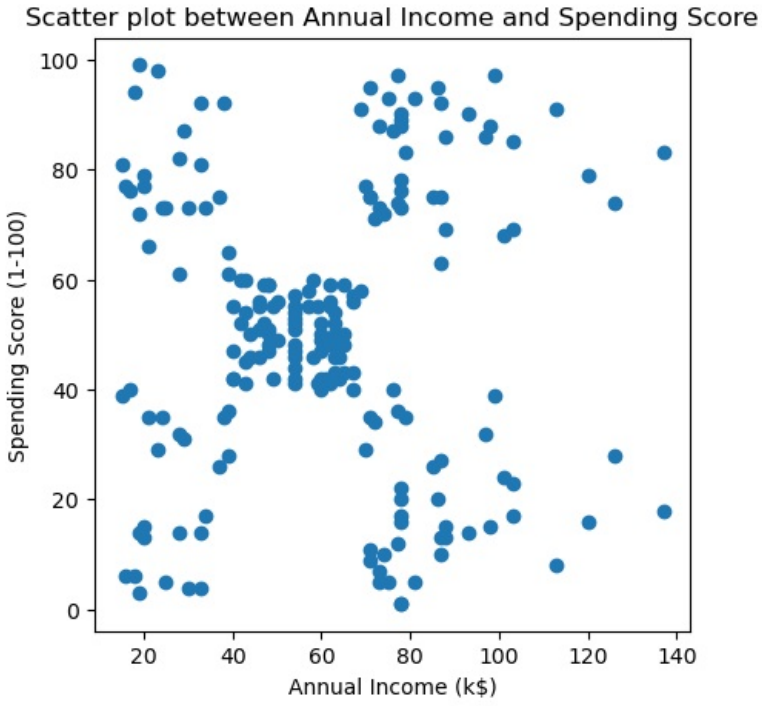
In [11]:
```
plt.figure(figsize=(5,5))
plt.scatter(df['Age'],df['Spending Score (1-100)']);
plt.xlabel('Age')
plt.ylabel('Spending Score (1-100)')
plt.title('Scatter plot between Age and Spending Score')
```

Out[11]:
```
Text(0.5, 1.0, 'Scatter plot between Age and Spending Score')
```



In [12]:
```
plt.figure(figsize=(5,5))
plt.scatter(df['Annual Income (k$)'],df['Spending Score (1-100)']);
plt.xlabel('Annual Income (k$)')
plt.ylabel('Spending Score (1-100)')
plt.title('Scatter plot between Annual Income and Spending Score')
```

Out[12]:
```
Text(0.5, 1.0, 'Scatter plot between Annual Income and Spending Score')
```

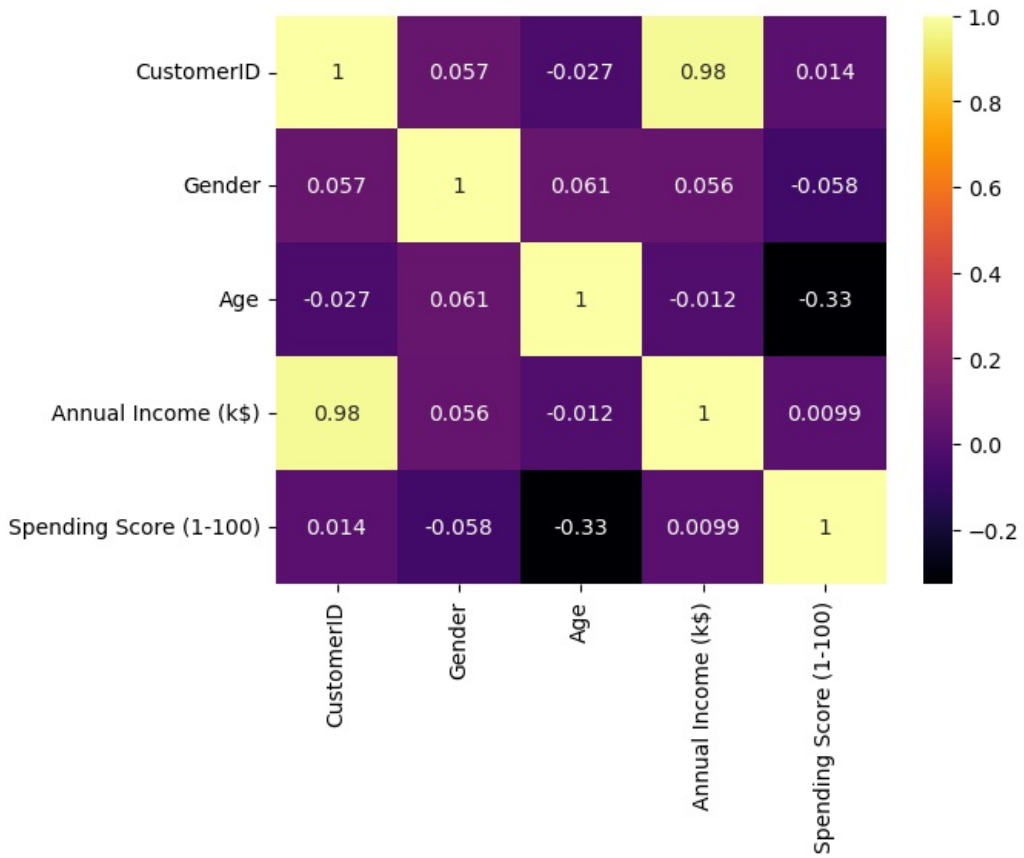


From above graphs - bivariate analysis - it can seen that there is group division or clustering, when we see grpah between spending score and Annual Income

Multivariate Analysis

In [13]:
```
sns.heatmap(df.corr(), annot=True, cmap='inferno')
```

Out[13]:
```
<Axes: >
```



Standardization

It works by putting various variables on the same scale and enables the data to be internally consistent.

In [14]:
```
from sklearn.preprocessing import StandardScaler
```

In [15]:
```
scaler = StandardScaler()
```

In [16]:
```
df[['Age', 'Annual Income (k$)', 'Spending Score (1-100)']] = scaler.fit_transform(df[['Age', 'Annual Income (k
```

In [17]:
```
df.head()
```

Out[17]:

	CustomerID	Gender	Age	Annual Income (k$)	Spending Score (1-100)
0	1	1	-1.424569	-1.738999	-0.434801
1	2	1	-1.281035	-1.738999	1.195704
2	3	0	-1.352802	-1.700830	-1.715913
3	4	0	-1.137502	-1.700830	1.040418
4	5	0	-0.563369	-1.662660	-0.395980

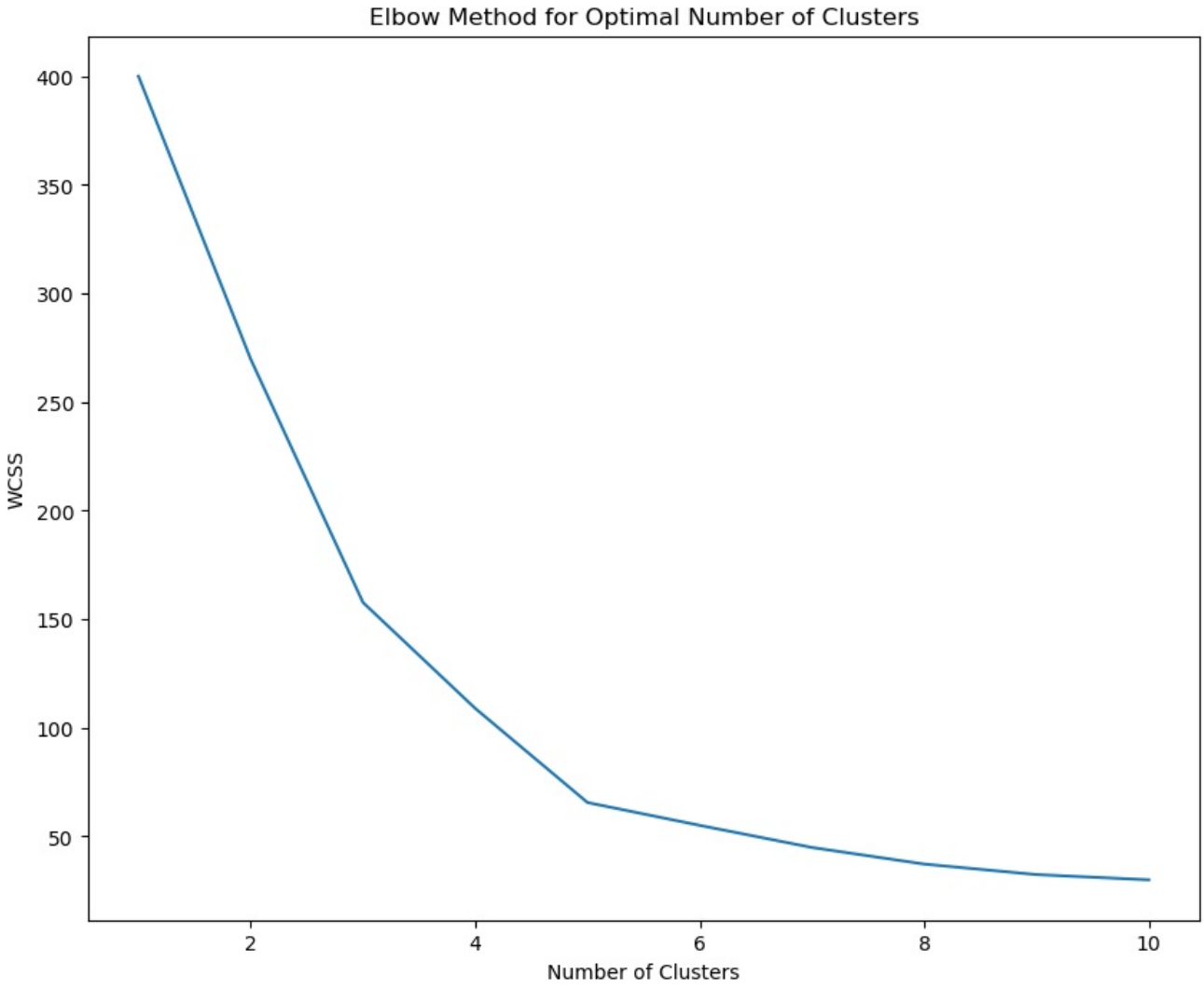
In [23]:
```
# Selecting features for clustering
X = df[['Annual Income (k$)', 'Spending Score (1-100)']].values
```

In [24]:
```
# Elbow method to find the optimal number of clusters
wcss = []  # Within-cluster sum of squares

for i in range(1, 11):
    kmeans = KMeans(n_clusters=i, init='k-means++', random_state=42)
    kmeans.fit(X)
    wcss.append(kmeans.inertia_)
```

In [25]:
```
plt.figure(figsize=(10, 8))
plt.plot(range(1, 11), wcss)
plt.title('Elbow Method for Optimal Number of Clusters')
plt.xlabel('Number of Clusters')
plt.ylabel('WCSS')
```

Out[25]:
```
Text(0, 0.5, 'WCSS')
```



Training the model with optimal cluster 5

In [26]:
```
kmeans = KMeans(n_clusters=5, init='k-means++', random_state=0)

# return a label for each data point based on their cluster
Y = kmeans.fit_predict(X)

print(Y)
```

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

Visulaization

In [28]:
```
# plotting all the clusters and their Centroids

plt.figure(figsize=(8,8))
plt.scatter(X[Y==0,0], X[Y==0,1], s=50, c='green', label='Cluster 1')
plt.scatter(X[Y==1,0], X[Y==1,1], s=50, c='red', label='Cluster 2')
plt.scatter(X[Y==2,0], X[Y==2,1], s=50, c='yellow', label='Cluster 3')
plt.scatter(X[Y==3,0], X[Y==3,1], s=50, c='black', label='Cluster 4')
plt.scatter(X[Y==4,0], X[Y==4,1], s=50, c='blue', label='Cluster 5')

# plot the centroids
plt.scatter(kmeans.cluster_centers_[:,0], kmeans.cluster_centers_[:,1], s=100, c='cyan', label='Centroids')

plt.title('Customer Groups')
plt.xlabel('Annual Income')
plt.ylabel('Spending Score')
plt.show()
```