In [1]:
```
import pandas as pd
```

In [2]:
```
import numpy as np
```

In [3]:
```
import matplotlib.pyplot as plt
```

In [4]:
```
from sklearn.cluster import KMeans
```

In [5]:
```
from sklearn.preprocessing import StandardScaler
```

In [6]:
```
ds=pd.read_csv("Mall_Customers.csv")
```

In [7]:
```
ds
```

Out[7]:

	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
...	...	...	...	...	...
195	196	Female	35	120	79
196	197	Female	45	126	28
197	198	Male	32	126	74
198	199	Male	32	137	18
199	200	Male	30	137	83

200 rows × 5 columns

In [9]:
```
features = ["Annual Income (k$)", "Spending Score (1-100)"]
X = ds[features]
```

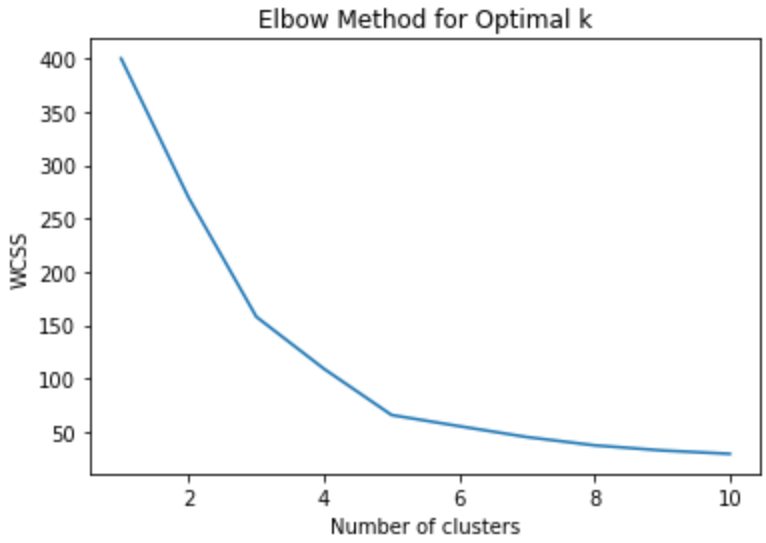
In [10]:
```
scaler = StandardScaler()
```

In [11]:
```
X_scaled = scaler.fit_transform(X)
```

In [12]:
```
# Determine the optimal number of clusters using the elbow method
wcss = []
for i in range(1, 11):
    kmeans = KMeans(n_clusters=i, init='k-means++', max_iter=300, n_init=10, r
    kmeans.fit(X_scaled)
    wcss.append(kmeans.inertia_)
```

```
G:\ML\lib\site-packages\sklearn\cluster\_kmeans.py:1440: UserWarning: KMeans
is known to have a memory leak on Windows with MKL, when there are less chunk
s than available threads. You can avoid it by setting the environment variabl
e OMP_NUM_THREADS=1.
  warnings.warn(
G:\ML\lib\site-packages\sklearn\cluster\_kmeans.py:1440: UserWarning: KMeans
is known to have a memory leak on Windows with MKL, when there are less chunk
s than available threads. You can avoid it by setting the environment variabl
e OMP_NUM_THREADS=1.
  warnings.warn(
G:\ML\lib\site-packages\sklearn\cluster\_kmeans.py:1440: UserWarning: KMeans
is known to have a memory leak on Windows with MKL, when there are less chunk
s than available threads. You can avoid it by setting the environment variabl
e OMP_NUM_THREADS=1.
  warnings.warn(
G:\ML\lib\site-packages\sklearn\cluster\_kmeans.py:1440: UserWarning: KMeans
is known to have a memory leak on Windows with MKL, when there are less chunk
s than available threads. You can avoid it by setting the environment variabl
e OMP_NUM_THREADS=1.
  warnings.warn(
G:\ML\lib\site-packages\sklearn\cluster\_kmeans.py:1440: UserWarning: KMeans
is known to have a memory leak on Windows with MKL, when there are less chunk
s than available threads. You can avoid it by setting the environment variabl
e OMP_NUM_THREADS=1.
  warnings.warn(
G:\ML\lib\site-packages\sklearn\cluster\_kmeans.py:1440: UserWarning: KMeans
is known to have a memory leak on Windows with MKL, when there are less chunk
s than available threads. You can avoid it by setting the environment variabl
e OMP_NUM_THREADS=1.
  warnings.warn(
G:\ML\lib\site-packages\sklearn\cluster\_kmeans.py:1440: UserWarning: KMeans
is known to have a memory leak on Windows with MKL, when there are less chunk
s than available threads. You can avoid it by setting the environment variabl
e OMP_NUM_THREADS=1.
  warnings.warn(
G:\ML\lib\site-packages\sklearn\cluster\_kmeans.py:1440: UserWarning: KMeans
is known to have a memory leak on Windows with MKL, when there are less chunk
s than available threads. You can avoid it by setting the environment variabl
e OMP_NUM_THREADS=1.
  warnings.warn(
G:\ML\lib\site-packages\sklearn\cluster\_kmeans.py:1440: UserWarning: KMeans
is known to have a memory leak on Windows with MKL, when there are less chunk
s than available threads. You can avoid it by setting the environment variabl
e OMP_NUM_THREADS=1.
  warnings.warn(
G:\ML\lib\site-packages\sklearn\cluster\_kmeans.py:1440: UserWarning: KMeans
is known to have a memory leak on Windows with MKL, when there are less chunk
s than available threads. You can avoid it by setting the environment variabl
e OMP_NUM_THREADS=1.
  warnings.warn(
```

In [13]:
```
# Plot the elbow method graph
plt.plot(range(1, 11), wcss)
plt.title('Elbow Method for Optimal k')
plt.xlabel('Number of clusters')
plt.ylabel('WCSS')  # Within-Cluster Sum of Squares
plt.show()
```



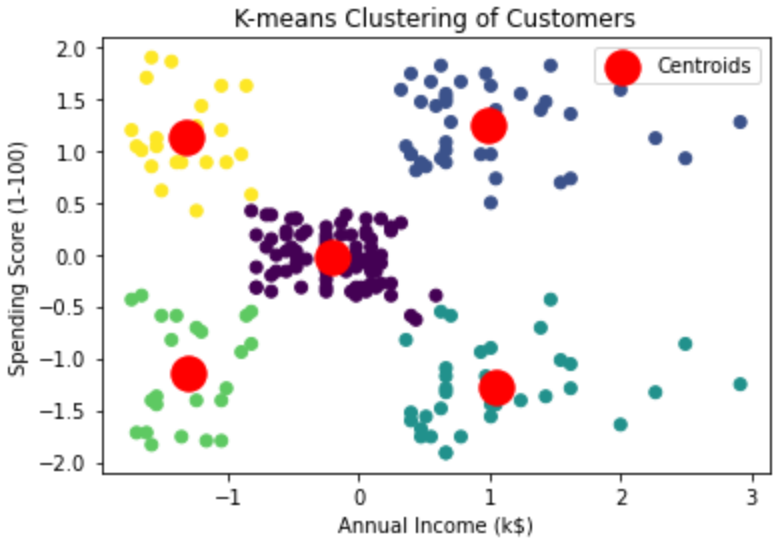
In [14]:
```
# Based on the elbow method, choose the optimal number of clusters
optimal_k = 5
```

In [15]:
```
# Apply K-means clustering with the chosen number of clusters
kmeans = KMeans(n_clusters=optimal_k, init='k-means++', max_iter=300, n_init=1
clusters = kmeans.fit_predict(X_scaled)
```

```
G:\ML\lib\site-packages\sklearn\cluster\_kmeans.py:1440: UserWarning: KMeans
is known to have a memory leak on Windows with MKL, when there are less chunk
s than available threads. You can avoid it by setting the environment variabl
e OMP_NUM_THREADS=1.
  warnings.warn(
```

In [17]:
```
# Add the cluster labels to the dataset
ds['Cluster'] = clusters
```

In [18]:
```
# Visualize the clusters
plt.scatter(X_scaled[:, 0], X_scaled[:, 1], c=clusters, cmap='viridis')
plt.scatter(kmeans.cluster_centers_[:, 0], kmeans.cluster_centers_[:, 1], s=30
plt.title('K-means Clustering of Customers')
plt.xlabel('Annual Income (k$)')
plt.ylabel('Spending Score (1-100)')
plt.legend()
plt.show()
```



In [ ]: