K - means clustering is a popular unsupervised machine learning algorithm used for partitioning data into groups based on similarity. It is widely used in various fields , including:

Customer segementation:- K-means clustering can helps business identify different groups of customers based on their purchasing behaviour, demographics, or preferences. This information can be used to create targeted marketing campaigns or personalize product recommedations.

Image segementation:- K-means clustering can be applied to segment images to distinct regions based on color or texture similarit. This can be useful in computer vision tasks sich as object recongnition or image compression.

Anomaly detection:- K-means clustering can be used to detect outliers or anomalies in data. By clustering the data and identifying instances that do not belong to any cluster or belong to small clusters, unsual patterns or outliers can be detected.

Document clustering:- K-means clustering can be applied to group similar documents together based on their content or textual features. This can be helpful in tasks such as document organization, topic modeling, or information retrieval.

However, there are certain scenarios where K-means clustering may not be suitable:

Non-linear data K-means clustering assumes that clusters are spherical and have similar sizes. If the data has complex shapes or contains non-linear relationships, K-means may not perform well. In such cases, other clustering algorithms like DBSCAN or Gaussian Mixture Models(GMM) may be more appropriate.

Categorical data:- K-means clustering is primarily designed for continous numerical data. If you have categorical feature, you need to preprocess or transform the data toa suitable numerical representation before applying K-means clustering.

Example - 1

In [1]:
```
from sklearn.cluster import KMeans
import numpy as np

# Generate sample data
X = np.random.rand(100 , 2)        # 100 data points with 2 features

# Create a K-Means clustering model with 3 clusters
kmeans = KMeans(n_clusters = 3)

# Fit the model to data
kmeans.fit(X)
```

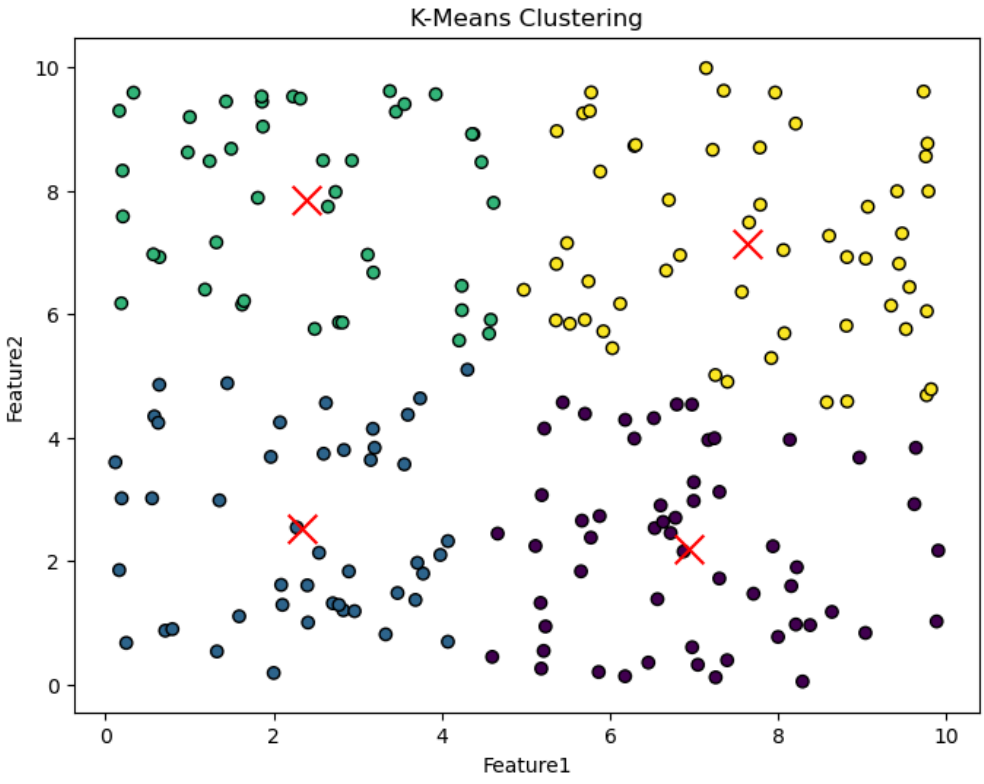
```
C:\ProgramData\anaconda3\Lib\site-packages\sklearn\cluster\_kmeans.py:1412: FutureWarning: The default value of `n_
init` will change from 10 to 'auto' in 1.4. Set the value of `n_init` explicitly to suppress the warning
  super()._check_params_vs_input(X, default_n_init=10)
C:\ProgramData\anaconda3\Lib\site-packages\sklearn\cluster\_kmeans.py:1436: UserWarning: KMeans is known to have a
memory leak on Windows with MKL, when there are less chunks than available threads. You can avoid it by setting the
environment variable OMP_NUM_THREADS=1.
  warnings.warn(
```

Out[1]:
```
KMeans
KMeans(n_clusters=3)
```

Example - 2

In [2]:
```
import numpy as np
import matplotlib.pyplot as plt
from sklearn.cluster import KMeans

# Generate synthetic data
np.random.seed(0)
n_samples = 200
n_clusters = 4
X = np.random.rand(n_samples , 2) * 10

# Create K-Means clustering model
kmeans = KMeans(n_clusters = n_clusters)
kmeans.fit(X)

# Get cluster lables and centers
labels = kmeans.labels_
centers = kmeans.cluster_centers_

# Visiualize the clusters
plt.figure(figsize = (8,6))
plt.scatter(X[:,0] , X[:,1] , c = labels,
           cmap = 'viridis', edgecolor='k')
plt.scatter(centers[:,0], centers[:,1],
           c='red' , marker='x',s=200)
plt.title('K-Means Clustering')
plt.xlabel('Feature1')
plt.ylabel('Feature2')
plt.show()
```

```
C:\ProgramData\anaconda3\Lib\site-packages\sklearn\cluster\_kmeans.py:1412: FutureWarning: The default value of `n_
init` will change from 10 to 'auto' in 1.4. Set the value of `n_init` explicitly to suppress the warning
  super()._check_params_vs_input(X, default_n_init=10)
C:\ProgramData\anaconda3\Lib\site-packages\sklearn\cluster\_kmeans.py:1436: UserWarning: KMeans is known to have a
memory leak on Windows with MKL, when there are less chunks than available threads. You can avoid it by setting the
environment variable OMP_NUM_THREADS=1.
  warnings.warn(
```



In [3]:
```
import pandas as pd
import numpy as np
```

In [4]:
```
df = pd.read_csv("D:\Summer Training Video\ML\mall.csv")
```

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

Out[5]:

	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

In [6]:
```
df = df.drop(columns = ['CustomerID' , 'Genre'])
```

In [7]:
```
df.head(10)
```

Out[7]:

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

In [8]:
```
x = df.iloc[: ,[0,1]].values
```

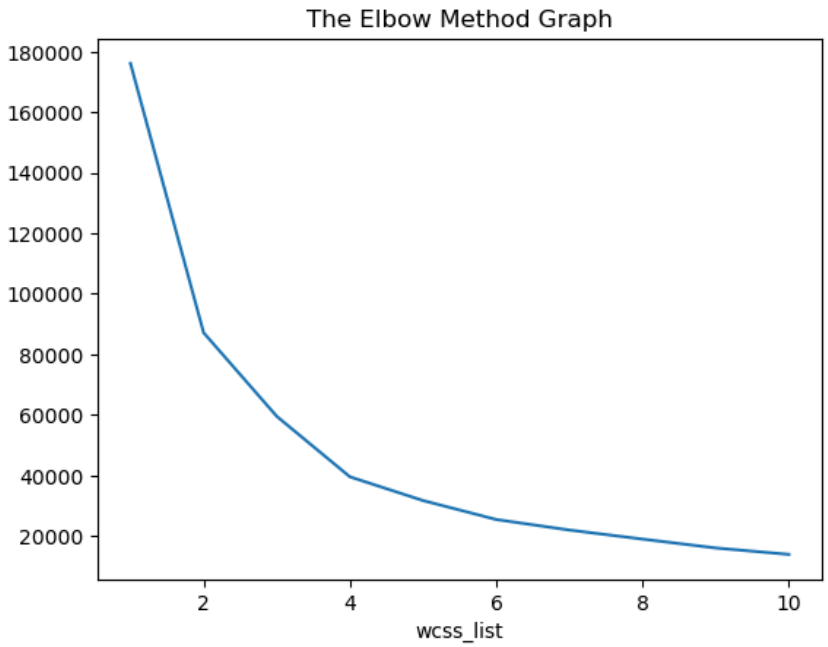
In [9]:
```
from sklearn.cluster import KMeans
```

In [10]:
```
import matplotlib.pyplot as plt
```

In [12]:

```
a = []

for i in range(1,11):
    b = KMeans(n_clusters = i , init = 'k-means++' , random_state = 42)
    b.fit(x)
    a.append(b.inertia_)

plt.plot(range(1,11) , a)

plt.title('The Elbow Method Graph')
plt.xlabel('Number of clusters(k)')
plt.xlabel('wcss_list')
plt.show()
```

```
C:\ProgramData\anaconda3\Lib\site-packages\sklearn\cluster\_kmeans.py:1412: FutureWarning: The default value of `n_init` will change from 10 to 'auto' in 1.4. Set the value of `n_init` explicitly to suppress the warning
  super()._check_params_vs_input(X, default_n_init=10)
C:\ProgramData\anaconda3\Lib\site-packages\sklearn\cluster\_kmeans.py:1436: UserWarning: KMeans is known to have a memory leak on Windows with MKL, when there are less chunks than available threads. You can avoid it by setting the environment variable OMP_NUM_THREADS=1.
  warnings.warn(
C:\ProgramData\anaconda3\Lib\site-packages\sklearn\cluster\_kmeans.py:1412: FutureWarning: The default value of `n_init` will change from 10 to 'auto' in 1.4. Set the value of `n_init` explicitly to suppress the warning
  super()._check_params_vs_input(X, default_n_init=10)
C:\ProgramData\anaconda3\Lib\site-packages\sklearn\cluster\_kmeans.py:1436: UserWarning: KMeans is known to have a memory leak on Windows with MKL, when there are less chunks than available threads. You can avoid it by setting the environment variable OMP_NUM_THREADS=1.
  warnings.warn(
C:\ProgramData\anaconda3\Lib\site-packages\sklearn\cluster\_kmeans.py:1412: FutureWarning: The default value of `n_init` will change from 10 to 'auto' in 1.4. Set the value of `n_init` explicitly to suppress the warning
  super()._check_params_vs_input(X, default_n_init=10)
C:\ProgramData\anaconda3\Lib\site-packages\sklearn\cluster\_kmeans.py:1436: UserWarning: KMeans is known to have a memory leak on Windows with MKL, when there are less chunks than available threads. You can avoid it by setting the environment variable OMP_NUM_THREADS=1.
  warnings.warn(
C:\ProgramData\anaconda3\Lib\site-packages\sklearn\cluster\_kmeans.py:1412: FutureWarning: The default value of `n_init` will change from 10 to 'auto' in 1.4. Set the value of `n_init` explicitly to suppress the warning
  super()._check_params_vs_input(X, default_n_init=10)
C:\ProgramData\anaconda3\Lib\site-packages\sklearn\cluster\_kmeans.py:1436: UserWarning: KMeans is known to have a memory leak on Windows with MKL, when there are less chunks than available threads. You can avoid it by setting the environment variable OMP_NUM_THREADS=1.
  warnings.warn(
C:\ProgramData\anaconda3\Lib\site-packages\sklearn\cluster\_kmeans.py:1412: FutureWarning: The default value of `n_init` will change from 10 to 'auto' in 1.4. Set the value of `n_init` explicitly to suppress the warning
  super()._check_params_vs_input(X, default_n_init=10)
C:\ProgramData\anaconda3\Lib\site-packages\sklearn\cluster\_kmeans.py:1436: UserWarning: KMeans is known to have a memory leak on Windows with MKL, when there are less chunks than available threads. You can avoid it by setting the environment variable OMP_NUM_THREADS=1.
  warnings.warn(
C:\ProgramData\anaconda3\Lib\site-packages\sklearn\cluster\_kmeans.py:1412: FutureWarning: The default value of `n_init` will change from 10 to 'auto' in 1.4. Set the value of `n_init` explicitly to suppress the warning
  super()._check_params_vs_input(X, default_n_init=10)
C:\ProgramData\anaconda3\Lib\site-packages\sklearn\cluster\_kmeans.py:1436: UserWarning: KMeans is known to have a memory leak on Windows with MKL, when there are less chunks than available threads. You can avoid it by setting the environment variable OMP_NUM_THREADS=1.
  warnings.warn(
C:\ProgramData\anaconda3\Lib\site-packages\sklearn\cluster\_kmeans.py:1412: FutureWarning: The default value of `n_init` will change from 10 to 'auto' in 1.4. Set the value of `n_init` explicitly to suppress the warning
  super()._check_params_vs_input(X, default_n_init=10)
C:\ProgramData\anaconda3\Lib\site-packages\sklearn\cluster\_kmeans.py:1436: UserWarning: KMeans is known to have a memory leak on Windows with MKL, when there are less chunks than available threads. You can avoid it by setting the environment variable OMP_NUM_THREADS=1.
  warnings.warn(
C:\ProgramData\anaconda3\Lib\site-packages\sklearn\cluster\_kmeans.py:1412: FutureWarning: The default value of `n_init` will change from 10 to 'auto' in 1.4. Set the value of `n_init` explicitly to suppress the warning
  super()._check_params_vs_input(X, default_n_init=10)
C:\ProgramData\anaconda3\Lib\site-packages\sklearn\cluster\_kmeans.py:1436: UserWarning: KMeans is known to have a memory leak on Windows with MKL, when there are less chunks than available threads. You can avoid it by setting the environment variable OMP_NUM_THREADS=1.
  warnings.warn(
C:\ProgramData\anaconda3\Lib\site-packages\sklearn\cluster\_kmeans.py:1412: FutureWarning: The default value of `n_init` will change from 10 to 'auto' in 1.4. Set the value of `n_init` explicitly to suppress the warning
  super()._check_params_vs_input(X, default_n_init=10)
C:\ProgramData\anaconda3\Lib\site-packages\sklearn\cluster\_kmeans.py:1436: UserWarning: KMeans is known to have a memory leak on Windows with MKL, when there are less chunks than available threads. You can avoid it by setting the environment variable OMP_NUM_THREADS=1.
  warnings.warn(
C:\ProgramData\anaconda3\Lib\site-packages\sklearn\cluster\_kmeans.py:1412: FutureWarning: The default value of `n_init` will change from 10 to 'auto' in 1.4. Set the value of `n_init` explicitly to suppress the warning
  super()._check_params_vs_input(X, default_n_init=10)
C:\ProgramData\anaconda3\Lib\site-packages\sklearn\cluster\_kmeans.py:1436: UserWarning: KMeans is known to have a memory leak on Windows with MKL, when there are less chunks than available threads. You can avoid it by setting the environment variable OMP_NUM_THREADS=1.
  warnings.warn(
```

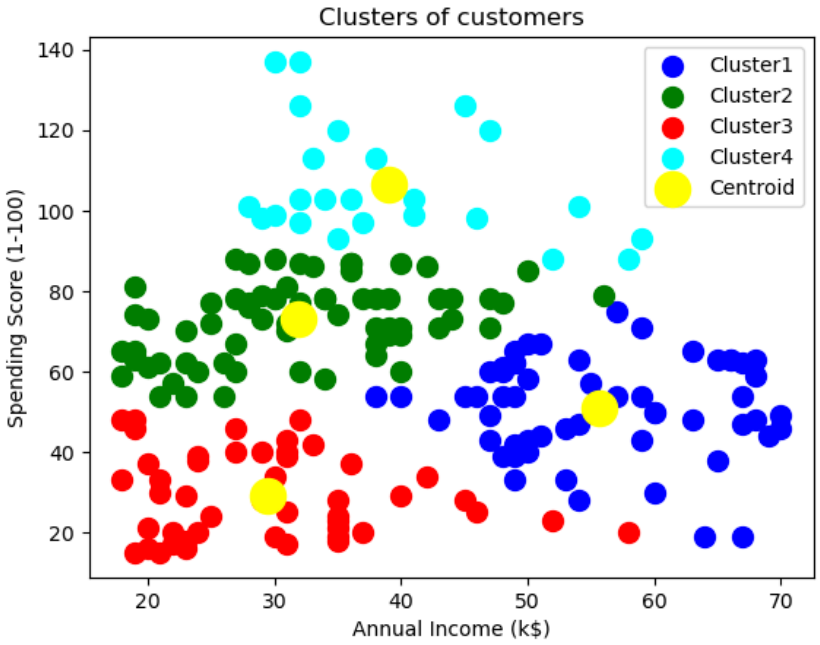


From the above plot, we can see the elbow point is at 4. So the number of clusters here will be 4.

In [13]:
```
b = KMeans(n_clusters = 4 , init = 'k-means++', random_state = 42)
y_predict = b.fit_predict(x)
```

```
C:\ProgramData\anaconda3\Lib\site-packages\sklearn\cluster\_kmeans.py:1412: FutureWarning: The default value of `n_
init` will change from 10 to 'auto' in 1.4. Set the value of `n_init` explicitly to suppress the warning
  super()._check_params_vs_input(X, default_n_init=10)
C:\ProgramData\anaconda3\Lib\site-packages\sklearn\cluster\_kmeans.py:1436: UserWarning: KMeans is known to have a
memory leak on Windows with MKL, when there are less chunks than available threads. You can avoid it by setting the
environment variable OMP_NUM_THREADS=1.
  warnings.warn(
```

In [14]:
```
# visualazing the clusters
plt.scatter(x[y_predict == 0, 0], x[y_predict == 0, 1], s = 100, c = 'blue', label='Cluster1')    #for first cluster
plt.scatter(x[y_predict == 1, 0], x[y_predict == 1, 1], s = 100, c = 'green', label='Cluster2')   #for second cluste
plt.scatter(x[y_predict == 2, 0], x[y_predict == 2, 1], s = 100, c = 'red', label='Cluster3')     #for third cluster
plt.scatter(x[y_predict == 3, 0], x[y_predict == 3, 1], s = 100, c = 'cyan', label='Cluster4')    #for fourth cluste

plt.scatter(b.cluster_centers_[:,0], b.cluster_centers_[:,1], s = 300 , c = 'yellow',
           label = 'Centroid')
plt.title('Clusters of customers')
plt.xlabel('Annual Income (k$)')
plt.ylabel('Spending Score (1-100)')
plt.legend()
plt.show()
```



In [ ]: