Project 3_b Target Customer Compass

1. Import Libraries

In [12]:
```
# Data Manipulation
import pandas as pd
import numpy as np

# Visualization
import matplotlib.pyplot as plt
import seaborn as sns

# Clustering
from sklearn.cluster import KMeans
from sklearn.preprocessing import StandardScaler
from sklearn.metrics import silhouette_score
```

2. Load the Dataset

In [15]:
```
# Load the dataset
data = r"C:\Users\ssing\OneDrive\Desktop\Mall_Customers.csv"
data = pd.read_csv('Mall_Customers.csv')
data.head()
```

Out[15]:

	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

3. Explore Data

In [20]:
```
 # Display basic information about the dataset
data.info()

# Check statistical summary
data.describe()

# Check for missing values
data.isnull().sum()
```

```
<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
```

Out[20]:
```
CustomerID                0
Genre                     0
Age                       0
Annual Income (k$)        0
Spending Score (1-100)    0
dtype: int64
```

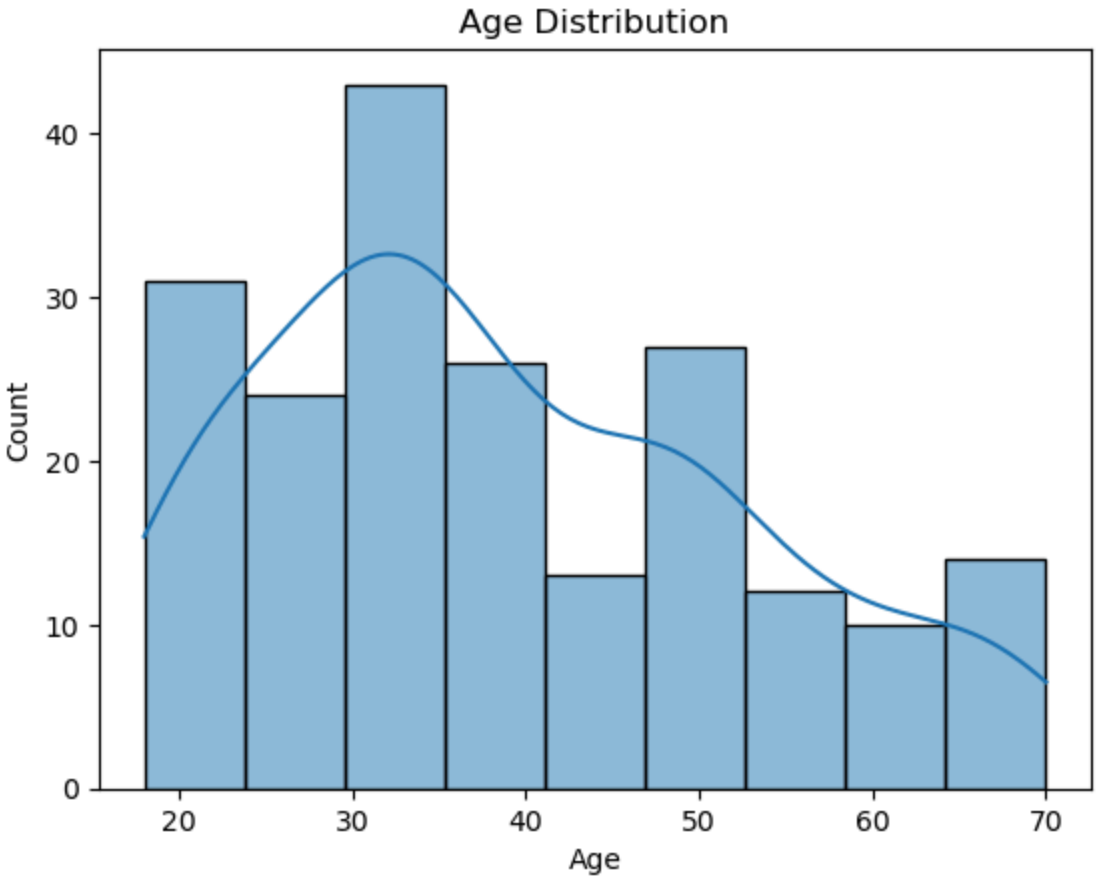
4. Data Cleaning

In [23]:
```
# Handle missing values (if any)
data = data.dropna()

# Check for duplicates
data = data.drop_duplicates()
```

5. Data Visualization

In [26]:
```
# Distribution of age
sns.histplot(data['Age'], kde=True)
```

```
plt.title('Age Distribution')
plt.show()

# Spending Score vs Annual Income
sns.scatterplot(x='Annual Income (k$)', y='Spending Score (1-100)', data=data)
plt.title('Spending Score vs Annual Income')
plt.show()
```



6. Feature Selection

In [29]:
```
# Select features for clustering
features = data[['Annual Income (k$)', 'Spending Score (1-100)']]
```

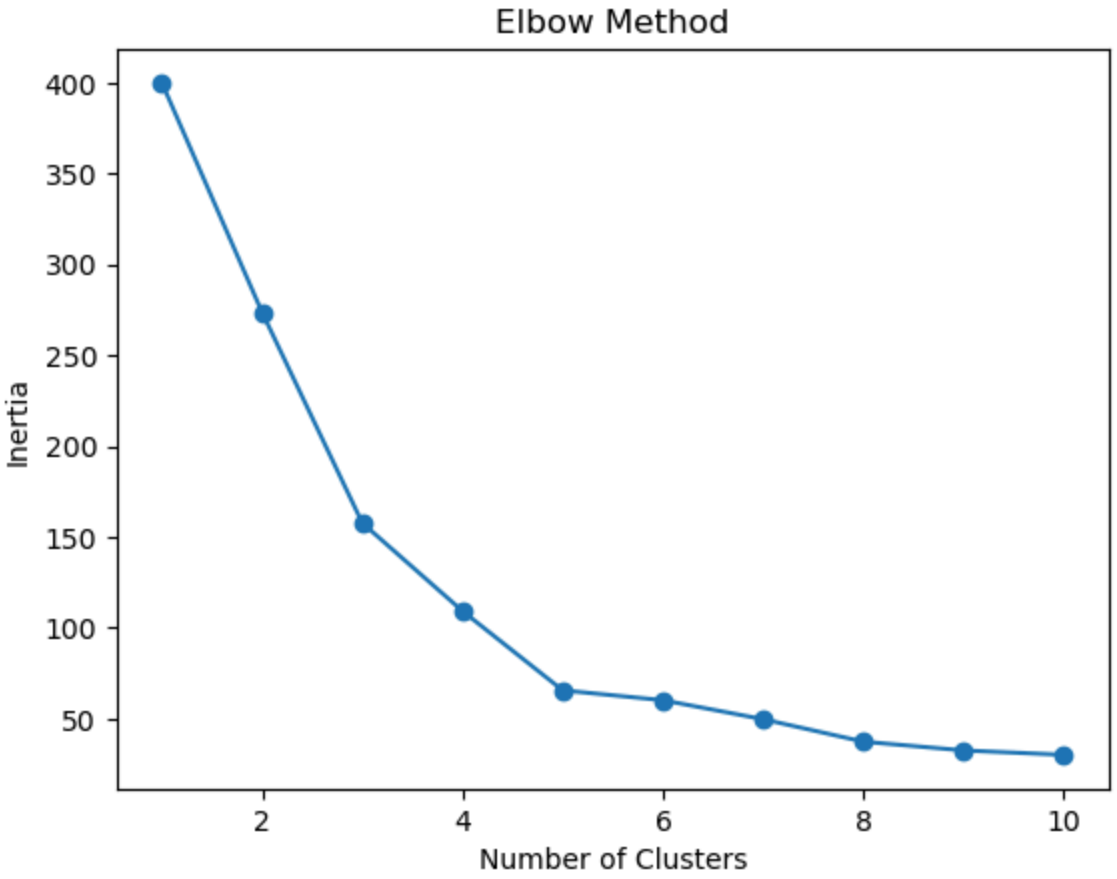
7. Standardize Data

In [38]:
```
# Standardize features
scaler = StandardScaler()
scaled_features = scaler.fit_transform(features)
```

8. Determine Optimal Number of Clusters

In [40]:
```
# Elbow Method
inertia = []
for k in range(1, 11):
    kmeans = KMeans(n_clusters=k, random_state=42)
    kmeans.fit(scaled_features)
    inertia.append(kmeans.inertia_)

# Plot Elbow Curve
plt.plot(range(1, 11), inertia, marker='o')
plt.title('Elbow Method')
plt.xlabel('Number of Clusters')
plt.ylabel('Inertia')
plt.show()

# Silhouette Score
for k in range(2, 11):
    kmeans = KMeans(n_clusters=k, random_state=42)
    labels = kmeans.fit_predict(scaled_features)
    score = silhouette_score(scaled_features, labels)
    print(f'Silhouette Score for {k} clusters: {score}')
```

```
C:\Users\ssing\anaconda3\Lib\site-packages\sklearn\cluster\_kmeans.py:1429: UserWarning: KMeans is known to have a me
mory leak on Windows with MKL, when there are less chunks than available threads. You can avoid it by setting the env
ironment variable OMP_NUM_THREADS=1.
  warnings.warn(
C:\Users\ssing\anaconda3\Lib\site-packages\sklearn\cluster\_kmeans.py:1429: UserWarning: KMeans is known to have a me
mory leak on Windows with MKL, when there are less chunks than available threads. You can avoid it by setting the env
ironment variable OMP_NUM_THREADS=1.
  warnings.warn(
C:\Users\ssing\anaconda3\Lib\site-packages\sklearn\cluster\_kmeans.py:1429: UserWarning: KMeans is known to have a me
mory leak on Windows with MKL, when there are less chunks than available threads. You can avoid it by setting the env
ironment variable OMP_NUM_THREADS=1.
  warnings.warn(
C:\Users\ssing\anaconda3\Lib\site-packages\sklearn\cluster\_kmeans.py:1429: UserWarning: KMeans is known to have a me
mory leak on Windows with MKL, when there are less chunks than available threads. You can avoid it by setting the env
ironment variable OMP_NUM_THREADS=1.
  warnings.warn(
C:\Users\ssing\anaconda3\Lib\site-packages\sklearn\cluster\_kmeans.py:1429: UserWarning: KMeans is known to have a me
mory leak on Windows with MKL, when there are less chunks than available threads. You can avoid it by setting the env
ironment variable OMP_NUM_THREADS=1.
  warnings.warn(
C:\Users\ssing\anaconda3\Lib\site-packages\sklearn\cluster\_kmeans.py:1429: UserWarning: KMeans is known to have a me
mory leak on Windows with MKL, when there are less chunks than available threads. You can avoid it by setting the env
ironment variable OMP_NUM_THREADS=1.
  warnings.warn(
C:\Users\ssing\anaconda3\Lib\site-packages\sklearn\cluster\_kmeans.py:1429: UserWarning: KMeans is known to have a me
mory leak on Windows with MKL, when there are less chunks than available threads. You can avoid it by setting the env
ironment variable OMP_NUM_THREADS=1.
  warnings.warn(
C:\Users\ssing\anaconda3\Lib\site-packages\sklearn\cluster\_kmeans.py:1429: UserWarning: KMeans is known to have a me
mory leak on Windows with MKL, when there are less chunks than available threads. You can avoid it by setting the env
ironment variable OMP_NUM_THREADS=1.
  warnings.warn(
C:\Users\ssing\anaconda3\Lib\site-packages\sklearn\cluster\_kmeans.py:1429: UserWarning: KMeans is known to have a me
mory leak on Windows with MKL, when there are less chunks than available threads. You can avoid it by setting the env
ironment variable OMP_NUM_THREADS=1.
  warnings.warn(
C:\Users\ssing\anaconda3\Lib\site-packages\sklearn\cluster\_kmeans.py:1429: UserWarning: KMeans is known to have a me
mory leak on Windows with MKL, when there are less chunks than available threads. You can avoid it by setting the env
ironment variable OMP_NUM_THREADS=1.
  warnings.warn(
```




```
Silhouette Score for 2 clusters: 0.3973270007887498
Silhouette Score for 3 clusters: 0.46658474419000145
Silhouette Score for 4 clusters: 0.49434988482196784
Silhouette Score for 5 clusters: 0.5546571631111091
Silhouette Score for 6 clusters: 0.5138257534676561
Silhouette Score for 7 clusters: 0.50200146805547
Silhouette Score for 8 clusters: 0.4550112502601921
Silhouette Score for 9 clusters: 0.4566624374485964
Silhouette Score for 10 clusters: 0.44475993501732874
```

```
C:\Users\ssing\anaconda3\Lib\site-packages\sklearn\cluster\_kmeans.py:1429: UserWarning: KMeans is known to have a me
mory leak on Windows with MKL, when there are less chunks than available threads. You can avoid it by setting the env
ironment variable OMP_NUM_THREADS=1.
  warnings.warn(
C:\Users\ssing\anaconda3\Lib\site-packages\sklearn\cluster\_kmeans.py:1429: UserWarning: KMeans is known to have a me
mory leak on Windows with MKL, when there are less chunks than available threads. You can avoid it by setting the env
ironment variable OMP_NUM_THREADS=1.
  warnings.warn(
C:\Users\ssing\anaconda3\Lib\site-packages\sklearn\cluster\_kmeans.py:1429: UserWarning: KMeans is known to have a me
mory leak on Windows with MKL, when there are less chunks than available threads. You can avoid it by setting the env
ironment variable OMP_NUM_THREADS=1.
  warnings.warn(
C:\Users\ssing\anaconda3\Lib\site-packages\sklearn\cluster\_kmeans.py:1429: UserWarning: KMeans is known to have a me
mory leak on Windows with MKL, when there are less chunks than available threads. You can avoid it by setting the env
ironment variable OMP_NUM_THREADS=1.
  warnings.warn(
C:\Users\ssing\anaconda3\Lib\site-packages\sklearn\cluster\_kmeans.py:1429: UserWarning: KMeans is known to have a me
mory leak on Windows with MKL, when there are less chunks than available threads. You can avoid it by setting the env
ironment variable OMP_NUM_THREADS=1.
  warnings.warn(
C:\Users\ssing\anaconda3\Lib\site-packages\sklearn\cluster\_kmeans.py:1429: UserWarning: KMeans is known to have a me
mory leak on Windows with MKL, when there are less chunks than available threads. You can avoid it by setting the env
ironment variable OMP_NUM_THREADS=1.
  warnings.warn(
C:\Users\ssing\anaconda3\Lib\site-packages\sklearn\cluster\_kmeans.py:1429: UserWarning: KMeans is known to have a me
mory leak on Windows with MKL, when there are less chunks than available threads. You can avoid it by setting the env
ironment variable OMP_NUM_THREADS=1.
  warnings.warn(
C:\Users\ssing\anaconda3\Lib\site-packages\sklearn\cluster\_kmeans.py:1429: UserWarning: KMeans is known to have a me
mory leak on Windows with MKL, when there are less chunks than available threads. You can avoid it by setting the env
ironment variable OMP_NUM_THREADS=1.
  warnings.warn(
C:\Users\ssing\anaconda3\Lib\site-packages\sklearn\cluster\_kmeans.py:1429: UserWarning: KMeans is known to have a me
mory leak on Windows with MKL, when there are less chunks than available threads. You can avoid it by setting the env
ironment variable OMP_NUM_THREADS=1.
  warnings.warn(
```

9. Apply K-means Clustering

In [43]:
```
# Apply K-means
optimal_k = 5  # Replace with the chosen number of clusters
```

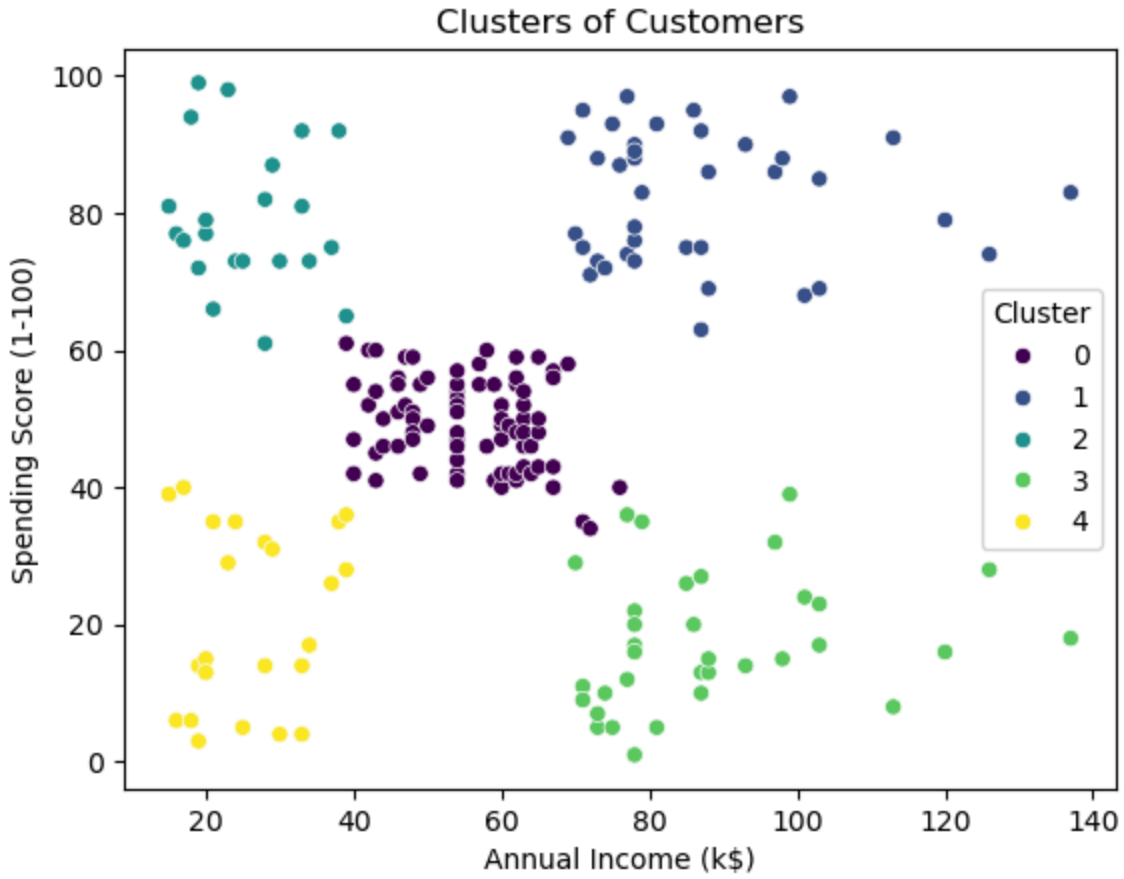
```
kmeans = KMeans(n_clusters=optimal_k, random_state=42)
data['Cluster'] = kmeans.fit_predict(scaled_features)
```

```
C:\Users\ssing\anaconda3\Lib\site-packages\sklearn\cluster\_kmeans.py:1429: UserWarning: KMeans is known to have a me
mory leak on Windows with MKL, when there are less chunks than available threads. You can avoid it by setting the env
ironment variable OMP_NUM_THREADS=1.
  warnings.warn(
```

10. Analyze and Visualize Clusters

In [46]:
```
# Cluster centers
centers = scaler.inverse_transform(kmeans.cluster_centers_)
print("Cluster Centers:", centers)

# Visualize clusters
sns.scatterplot(x=features['Annual Income (k$)'], y=features['Spending Score (1-100)'], hue=data['Cluster'], palette=
plt.title('Clusters of Customers')
plt.show()
```

```
Cluster Centers: [[55.2962963  49.51851852]
 [86.53846154 82.12820513]
 [25.72727273 79.36363636]
 [88.2        17.11428571]
 [26.30434783 20.91304348]]
```



11. Generate Insights

In [49]:
```
for i in range(optimal_k):
    print(f"Cluster {i}:")
    print(data[data['Cluster'] == i].describe())
    print("\n")
```

```
Cluster 0:
       CustomerID        Age  Annual Income (k$)  Spending Score (1-100)  \
count   81.000000  81.000000           81.000000               81.000000
mean    86.320988  42.716049           55.296296               49.518519
std     24.240889  16.447822            8.988109                6.530909
min     44.000000  18.000000           39.000000               34.000000
25%     66.000000  27.000000           48.000000               44.000000
50%     86.000000  46.000000           54.000000               50.000000
75%    106.000000  54.000000           62.000000               55.000000
max    143.000000  70.000000           76.000000               61.000000

       Cluster
count     81.0
mean       0.0
std        0.0
min        0.0
25%        0.0
50%        0.0
75%        0.0
max        0.0


Cluster 1:
       CustomerID        Age  Annual Income (k$)  Spending Score (1-100)  \
count   39.000000  39.000000           39.000000               39.000000
mean   162.000000  32.692308           86.538462               82.128205
std     22.803509   3.728650           16.312485                9.364489
min    124.000000  27.000000           69.000000               63.000000
25%    143.000000  30.000000           75.500000               74.500000
50%    162.000000  32.000000           79.000000               83.000000
75%    181.000000  35.500000           95.000000               90.000000
max    200.000000  40.000000          137.000000               97.000000

       Cluster
count     39.0
mean       1.0
std        0.0
min        1.0
25%        1.0
50%        1.0
75%        1.0
max        1.0
```

```
Cluster 2:
       CustomerID        Age  Annual Income (k$)  Spending Score (1-100)  \
count   22.000000  22.000000           22.000000               22.000000
mean    23.090909  25.272727           25.727273               79.363636
std     13.147185   5.257030            7.566731               10.504174
min      2.000000  18.000000           15.000000               61.000000
25%     12.500000  21.250000           19.250000               73.000000
50%     23.000000  23.500000           24.500000               77.000000
75%     33.500000  29.750000           32.250000               85.750000
max     46.000000  35.000000           39.000000               99.000000

       Cluster
count     22.0
mean       2.0
std        0.0
min        2.0
25%        2.0
50%        2.0
75%        2.0
max        2.0


Cluster 3:
       CustomerID        Age  Annual Income (k$)  Spending Score (1-100)  \
count   35.000000  35.000000           35.000000               35.000000
mean   164.371429  41.114286           88.200000               17.114286
std     21.457325  11.341676           16.399067                9.952154
min    125.000000  19.000000           70.000000                1.000000
25%    148.000000  34.000000           77.500000               10.000000
50%    165.000000  42.000000           85.000000               16.000000
75%    182.000000  47.500000           97.500000               23.500000
max    199.000000  59.000000          137.000000               39.000000

       Cluster
count     35.0
mean       3.0
std        0.0
min        3.0
25%        3.0
50%        3.0
```

```
75%        3.0
max        3.0


Cluster 4:
       CustomerID        Age  Annual Income (k$)  Spending Score (1-100)  \
count    23.00000  23.000000           23.000000               23.000000
mean     23.00000  45.217391           26.304348               20.913043
std      13.56466  13.228607            7.893811               13.017167
min       1.00000  19.000000           15.000000                3.000000
25%      12.00000  35.500000           19.500000                9.500000
50%      23.00000  46.000000           25.000000               17.000000
75%      34.00000  53.500000           33.000000               33.500000
max      45.00000  67.000000           39.000000               40.000000

       Cluster
count     23.0
mean       4.0
std        0.0
min        4.0
25%        4.0
50%        4.0
75%        4.0
max        4.0
```

In [ ]: