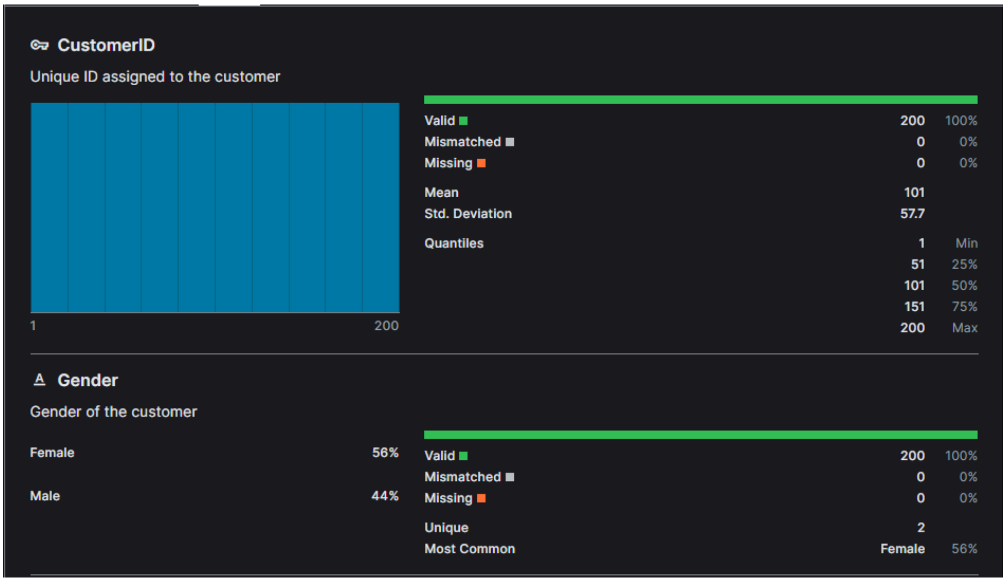
CUSTOMER SEGMENTATION USING CLUSTERING

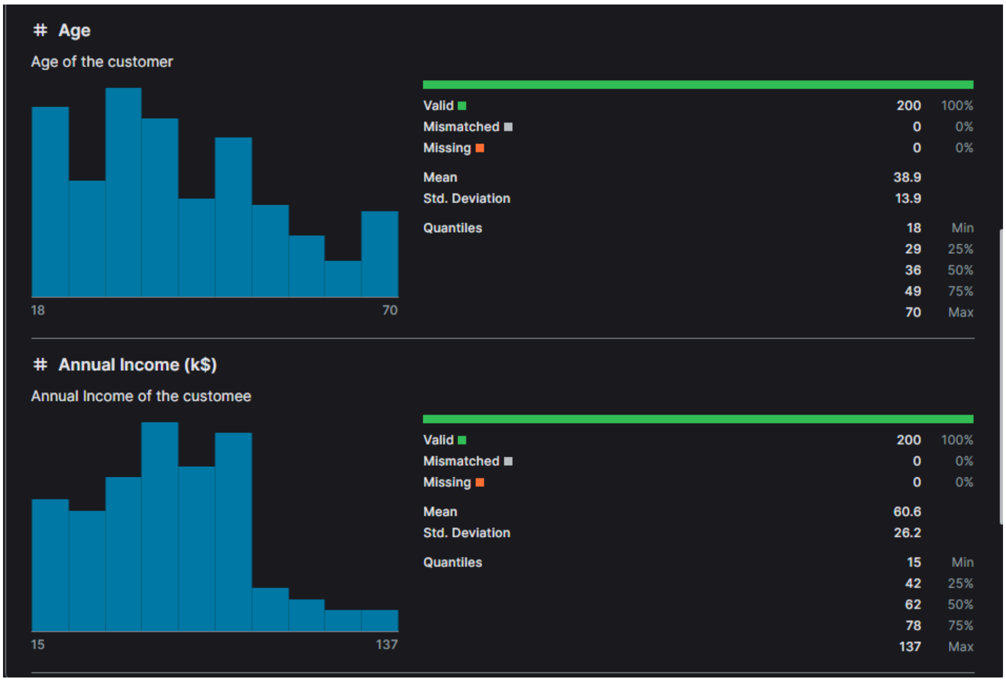
1. Objectives:
    - To understand and segment mall customers based on their spending behaviour.
    - To identify high-value target customers for personalised marketing strategies.
    - To provide actionable insights for the marketing team to improve conversion rates.

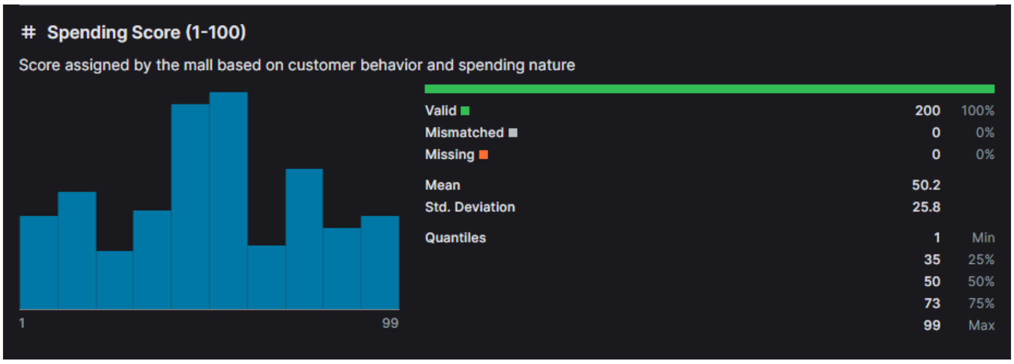
2. Dataset: Mall Customer Segmentation Data from Kaggle

    - Content: Contains 200 entries with columns: CustomerID, Gender, Age, Annual Income, and Spending Score.

      Spending Score is a value assigned to the customer based on defined parameters like customer behaviour and purchasing data.







3. Deliverables:

- Data preprocessing and exploratory data analysis (EDA).
- Clustering of customers using K-Means.
- Visualisation of customer segments.
- Identification of high-value target customer clusters.
- Marketing strategy recommendations based on cluster profiles.

4. Data Preprocessing and Exploratory Data Analysis(EDA)

4.1 Data Loading and Overview

- The dataset was loaded into a pandas DataFrame.
- Initial inspection showed 5 columns: CustomerID, Gender, Age, Annual Income, and Spending Score.

```
[3]: import pandas as pd
     import numpy as np
     import matplotlib.pyplot as plt
     import seaborn as sns
```

```
[4]: df = pd.read_csv("customer-segmentation-tutorial-in-python/Mall_Customers.csv")
```

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

[5]:

	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

```
[6]: df.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   Gender                  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
```

4.2 Data Cleaning

- Gender was encoded numerically: Male as 1 and Female as 0.
- No missing values were found in the dataset.

```
[7]: df.isnull().sum()

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

```
[8]: #no missing values
```

```
[9]: df.duplicated()

[9]: 0      False
     1      False
     2      False
     3      False
     4      False
            ...
     195    False
     196    False
     197    False
     198    False
     199    False
     Length: 200, dtype: bool
```

```
[10]: #no duplicates
```

```
[28]: from sklearn.preprocessing import LabelEncoder
```

```
[29]: df['Gender'] = LabelEncoder().fit_transform(df['Gender'])
```

4.3 Feature Selection and Scaling

- Selected features for clustering: Age, Annual Income, and Spending Score.
- StandardScaler was used to normalize the features for better clustering performance.

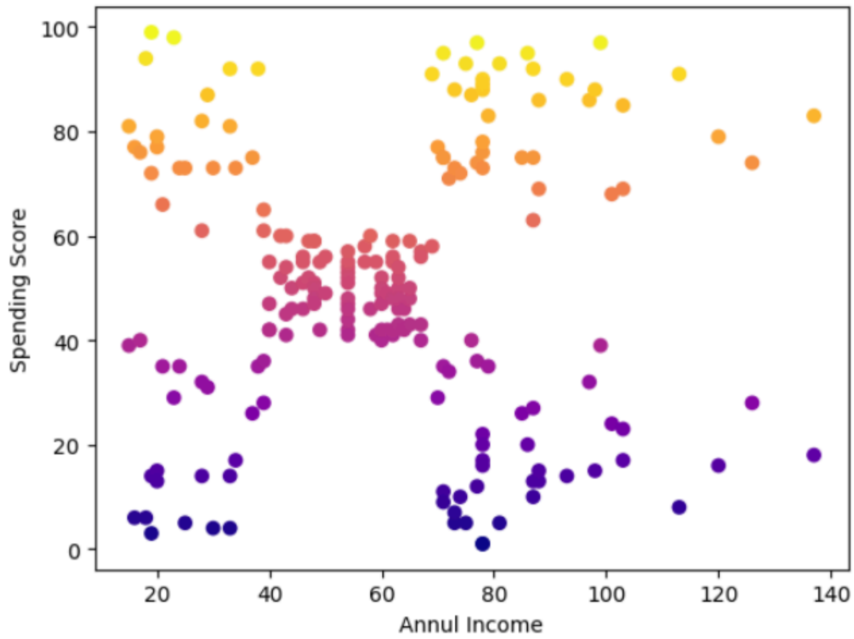
```
[32]: from sklearn.preprocessing import StandardScaler

      # Select features for clustering
      features = df[['Annual_Income', 'Spending_Score']]

      # Standardize the features
      scaler = StandardScaler()
      features_scaled = scaler.fit_transform(features)

      # Convert scaled features back to a DataFrame
      df_scaled = pd.DataFrame(features_scaled, columns=['Annual_Income', 'Spending_Score'])
```

4.4 EDA

- Performed EDA using various plots offered by seaborn and matplotlib to visualise data and understand relationships among different features

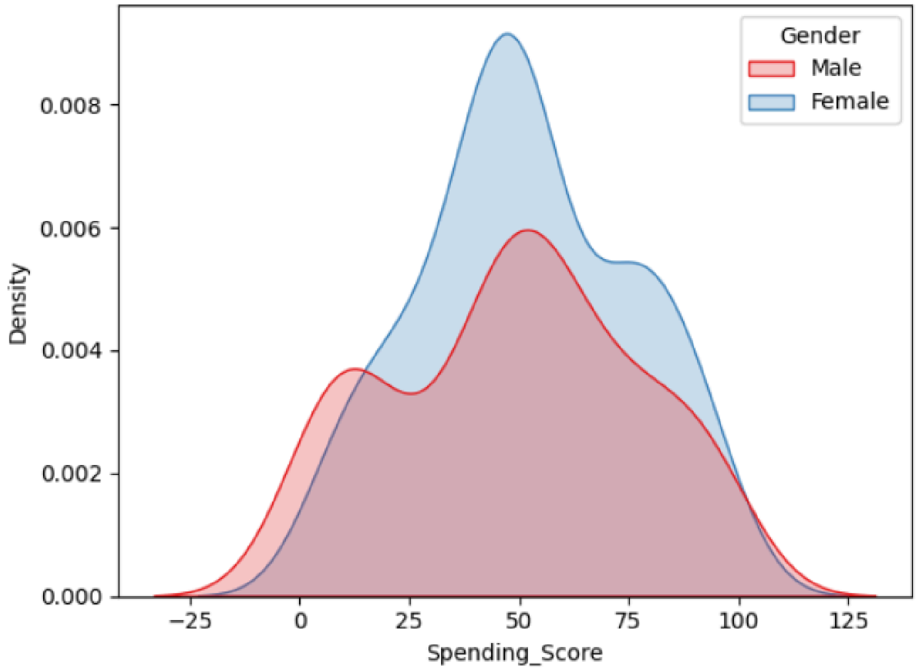
```
[23]: plt.scatter(df['Annual_Income'],df['Spending_Score'], c=df['Spending_Score'], cmap = plt.cm.plasma)
      plt.xlabel('Annul Income')
      plt.ylabel('Spending Score')
```

```
[23]: Text(0, 0.5, 'Spending Score')
```



[26]:
```
sns.kdeplot(data=df, x='Spending_Score', fill=True, hue='Gender',palette='Set1')
```

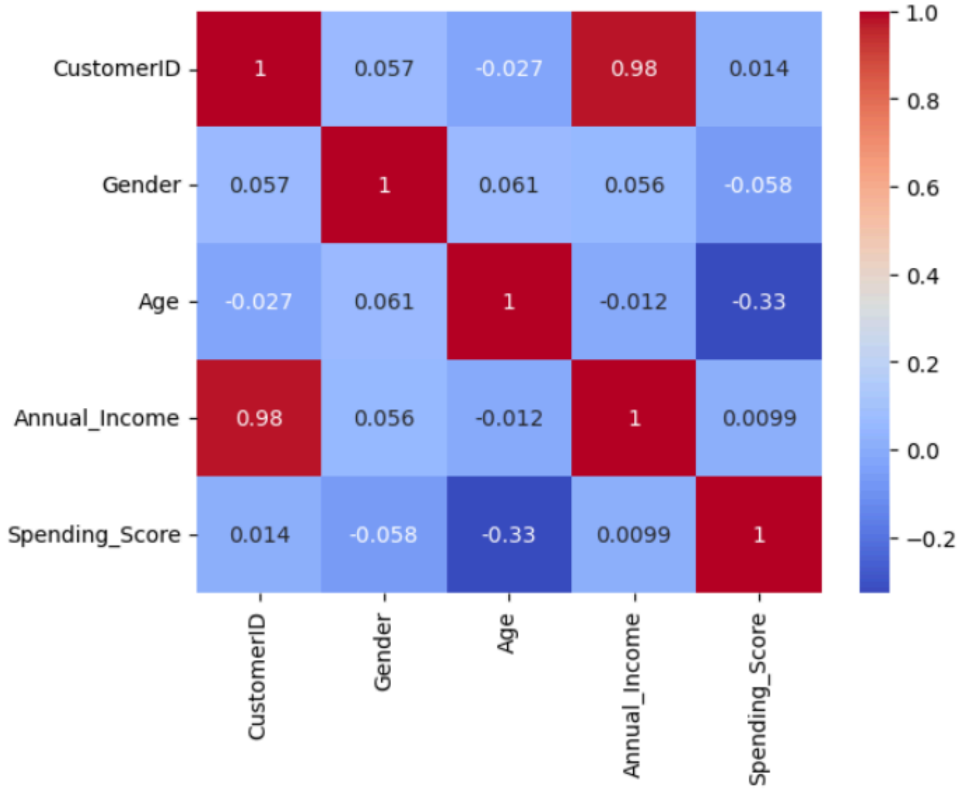
[26]: <Axes: xlabel='Spending_Score', ylabel='Density'>



[30]:
```
corr = df.corr()
```

[31]:
```
sns.heatmap(corr,annot=True, cmap='coolwarm')
```

[31]: <Axes: >

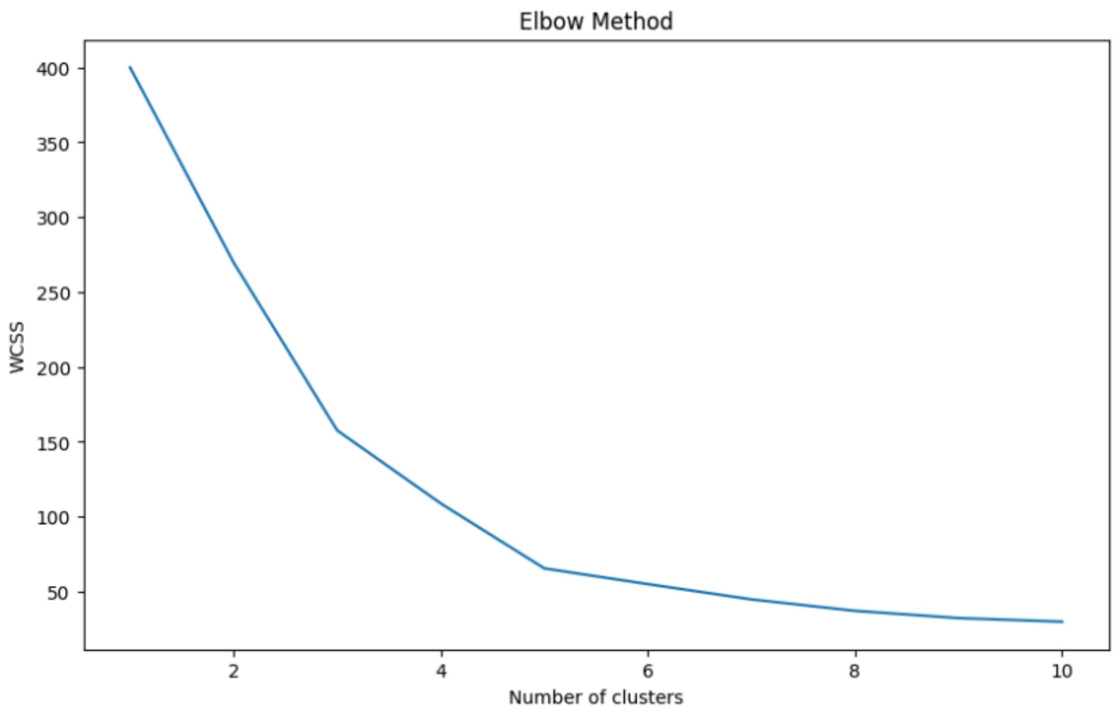


5. Clustering and Model Selection

5.1 K-Means Clustering

- K-Means algorithm was applied to the scaled features.
- A range of clusters (2 to 10) was tested to determine the optimal number of clusters using the silhouette score.

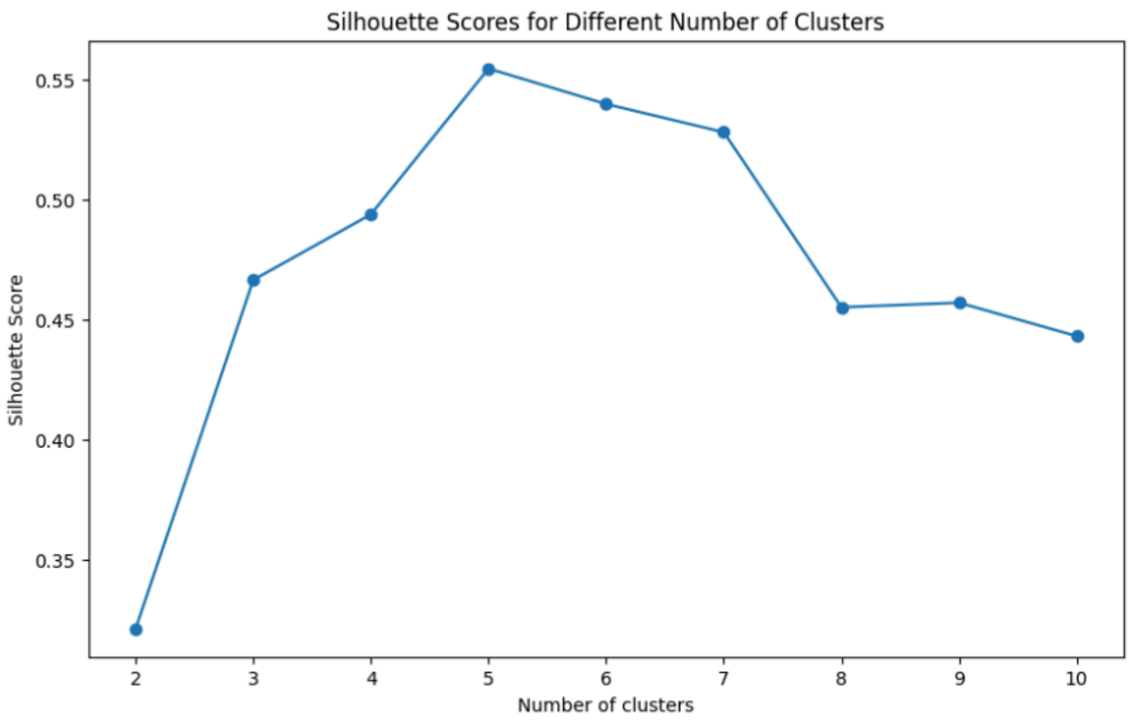
```
[33]: from sklearn.cluster import KMeans

# 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, random_state=42)
    kmeans.fit(df_scaled)
    wcss.append(kmeans.inertia_)

# Plot the Elbow Method graph
plt.figure(figsize=(10,6))
plt.plot(range(1, 11), wcss)
plt.title('Elbow Method')
plt.xlabel('Number of clusters')
plt.ylabel('WCSS')
plt.show()
```



5.2 Silhouette Analysis

- The silhouette score was calculated for each cluster count.
- The highest silhouette score (0.5551) was achieved with 5 clusters, indicating the optimal number of clusters.

```
[34]: #elbow at 3, so probably optimal number of clusters
#we will check with silhouette scores too
#higher score indicates better clusters

from sklearn.metrics import silhouette_score

silhouette_scores = []
for i in range(2, 11):
    kmeans = KMeans(n_clusters=i, init='k-means++', max_iter=300, n_init=10, random_state=42)
    kmeans.fit(df_scaled)
    y_kmeans = kmeans.predict(df_scaled)
    score = silhouette_score(df_scaled, y_kmeans)
    silhouette_scores.append(score)

plt.figure(figsize=(10,6))
plt.plot(range(2, 11), silhouette_scores, marker='o')
plt.title('Silhouette Scores for Different Number of Clusters')
plt.xlabel('Number of clusters')
plt.ylabel('Silhouette Score')
plt.show()
```

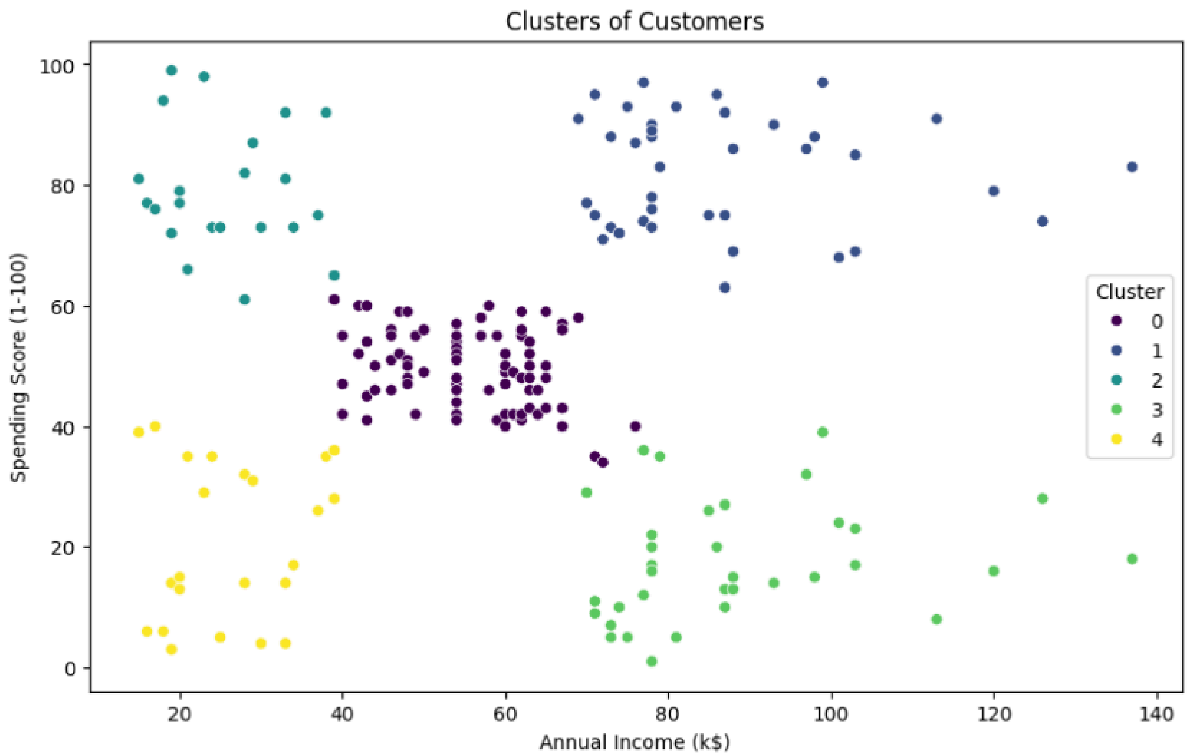


6. Visualisation

6.1 Cluster Visualization

- The clusters were visualized using a scatter plot of Annual Income vs. Spending Score.
- Each cluster was color-coded for better differentiation.

[37]:
```
kmeans = KMeans(n_clusters=5, init='k-means++', max_iter=300, n_init=10, random_state=42)
y_kmeans = kmeans.fit_predict(df_scaled)
df['Cluster'] = y_kmeans
plt.figure(figsize=(10,6))
sns.scatterplot(data=df, x='Annual_Income', y='Spending_Score', hue='Cluster', palette='viridis')
plt.title('Clusters of Customers')
plt.xlabel('Annual Income (k$)')
plt.ylabel('Spending Score (1-100)')
plt.legend(title='Cluster')
plt.show()
centroids = kmeans.cluster_centers_
print("Cluster Centroids:\n", centroids)
score = silhouette_score(df_scaled, y_kmeans)
print(f'Silhouette Score: {score}')
```



6.2 Cluster Profiles

- Cluster 0: Moderate income and spending.
- Cluster 1: High income, high spending.
- Cluster 2: Low income, high spending.
- Cluster 3: High income, low spending.
- Cluster 4: Low income, low spending.

```
[42]: #cluster profiling
```

```
[43]: cluster_profiles = df.groupby('Cluster').mean()
print("Cluster Profiles:\n", cluster_profiles)


for col in ['Age', 'Annual_Income', 'Spending_Score']:
    plt.figure(figsize=(10,6))
    sns.boxplot(x='Cluster', y=col, data=df)
    plt.title(f'Cluster Profiles: {col}')
    plt.xlabel('Cluster')
    plt.ylabel(col)
    plt.show()
```

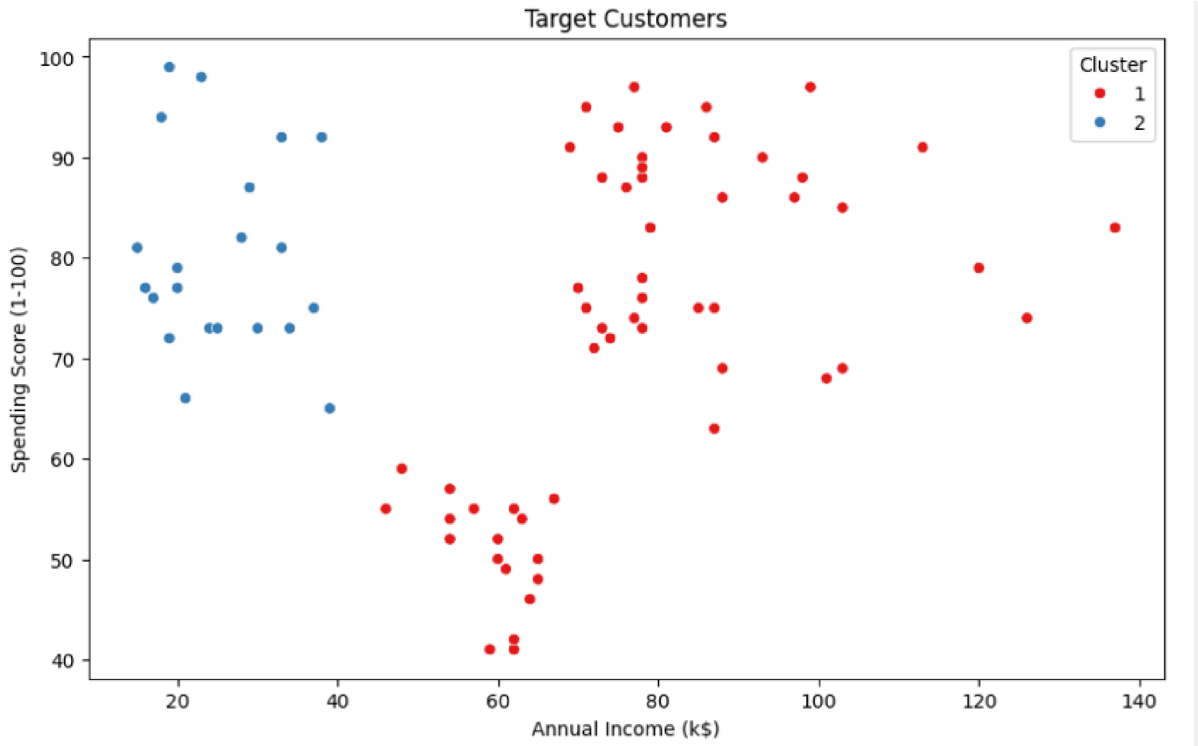
```
Cluster Profiles:
          CustomerID    Gender        Age  Annual_Income  Spending_Score
Cluster
0         86.320988  0.407407  42.716049      55.296296       49.518519
1        162.000000  0.461538  32.692308      86.538462       82.128205
2         23.090909  0.409091  25.272727      25.727273       79.363636
3        164.371429  0.542857  41.114286      88.200000       17.114286
4         23.000000  0.391304  45.217391      26.304348       20.913043
```

6.3 Target Customer Identification

- Target clusters were identified based on high spending scores: Clusters 1 and 2.

```
[58]: # Define criteria for target customers
target_clusters = cluster_profiles[cluster_profiles['Spending_Score'] > 60].index

print(f"Target Clusters: {target_clusters}")

# Filter data to get target customers
target_customers = df[df['Cluster'].isin(target_clusters)]
print("Target Customers:")
print(target_customers.head())
```

```
Target Clusters: Index([1, 2], dtype='int32', name='Cluster')
Target Customers:
   CustomerID  Gender  Age  Annual_Income  Spending_Score  Cluster  \
1           2       1   21             15              81        2
3           4       0   23             16              77        2
5           6       0   22             17              76        2
7           8       0   23             18              94        2
9          10       0   30             19              72        2

   Income_per_Age  Spending_per_Age  Income_to_Spending  Agg_Cluster
1        0.714286          3.857143            0.185185            4
3        0.695652          3.347826            0.207792            4
5        0.772727          3.454545            0.223684            4
7        0.782609          4.086957            0.191489            4
9        0.633333          2.400000            0.263889            4
```



7. Insights and Marketing Strategy Recommendations
   7.1 Cluster 1: High Income, High Spending

- **Customer Profile**: Young adults with high income and very high spending scores.
- **Marketing Strategy**:
  - Focus on luxury and premium products.
  - Leverage social media and influencer marketing.
  - Offer exclusive deals and early access to new products.

7.2 Cluster 2: Low Income, High Spending

- **Customer Profile**: Very young adults with low income but high spending scores.
- **Marketing Strategy**:
  - Emphasize value-for-money products.
  - Utilize online advertising, targeted promotions, and discounts.
  - Provide installment payment options.

8. Conclusion
   - Successfully segmented customers into distinct clusters.
   - Identified high-value target clusters for tailored marketing strategies.
   - Provided actionable insights to improve marketing effectiveness and customer engagement.