Lab 9: Customer Segmentation with K-Means Clustering

In this lab, you will apply K-Means clustering to segment customers based on their **Annual Income** and **Spending Score**. You will:

1. Load and explore the Mall Customers dataset
2. Preprocess features using standardisation
3. Use the **Elbow Method** to find the optimal number of clusters
4. Train a K-Means model and assign cluster labels
5. Visualise and interpret the resulting customer segments
6. *(Bonus)* Evaluate clustering quality with the Silhouette Score

Dataset: Mall Customer Segmentation — download `Mall_Customers.csv` and upload it to your Google Drive.

Step 0: Imports

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

Step 1: Mount Google Drive & Load the Dataset

Upload `Mall_Customers.csv` to your Google Drive (e.g. into a folder called `datasets`), then adjust the path below to match.

In [2]:
```
# Running locally: place Mall_Customers.csv next to this notebook (no Google
```

In [3]:
```
DATA_PATH = "Mall_Customers.csv"
df = pd.read_csv(DATA_PATH)
```

Explore the Data

Display the first rows, check shape, data types, and missing values.

In [4]:
```
df.head()
```

Out[4]:

	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

In [5]:
```
print(df.shape)
print(df.dtypes)
print(df.isnull().sum())
```

```
(200, 5)
CustomerID                int64
Gender                      str
Age                       int64
Annual Income (k$)        int64
Spending Score (1-100)    int64
dtype: object
CustomerID                0
Gender                    0
Age                       0
Annual Income (k$)        0
Spending Score (1-100)    0
dtype: int64
```

In [6]:
```
df.describe()
```

Out[6]:

	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

Step 2: Data Preprocessing

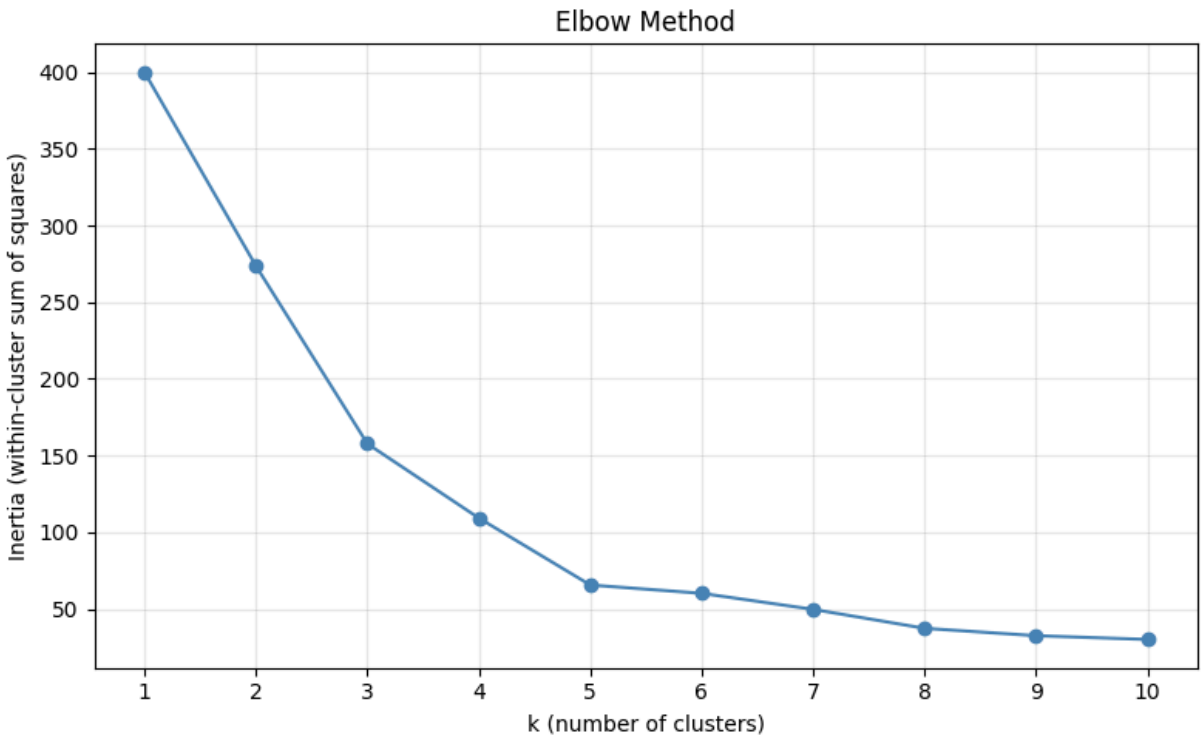
Select the features **Annual Income (k$)** and **Spending Score (1-100)**, then standardise them using `StandardScaler`.

In [7]:
```
X = df[["Annual Income (k$)", "Spending Score (1-100)"]]

scaler = StandardScaler()
X_scaled = scaler.fit_transform(X)
```

Step 3: Elbow Method — Finding the Optimal *k*

Fit K-Means for *k* = 1 … 10 and record the **inertia** (within-cluster sum of squares). Plot the elbow curve.

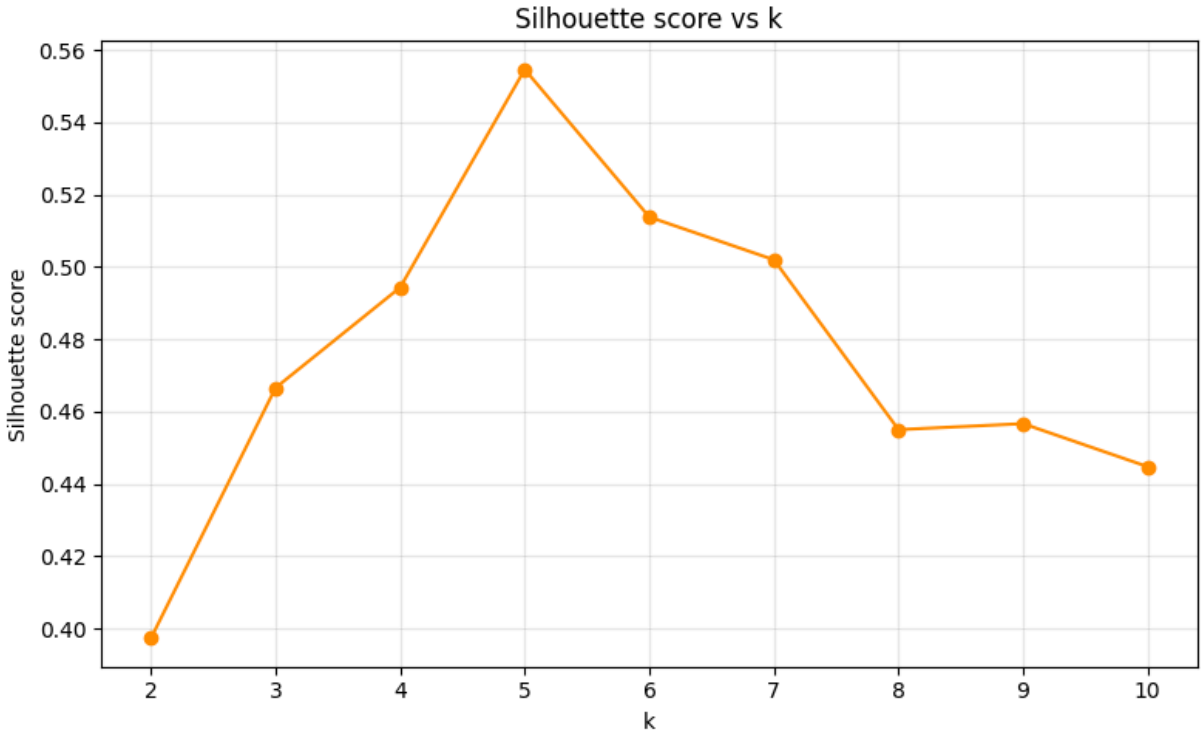
In [8]:
```
inertia = []
k_values = range(1, 11)

for k in k_values:
    model = KMeans(n_clusters=k, random_state=42)
    model.fit(X_scaled)
    inertia.append(model.inertia_)

plt.figure(figsize=(8, 5))
plt.plot(list(k_values), inertia, "o-", color="steelblue")
plt.xlabel("k (number of clusters)")
plt.ylabel("Inertia (within-cluster sum of squares)")
plt.title("Elbow Method")
plt.xticks(list(k_values))
plt.grid(alpha=0.3)
plt.tight_layout()
plt.show()
```



(Bonus) Silhouette Analysis

Compute the silhouette score for k = 2 … 10 and plot it. The peak indicates the best separation.

In [9]:
```
sil = []
for k in range(2, 11):
    km = KMeans(n_clusters=k, random_state=42)
    labels = km.fit_predict(X_scaled)
    sil.append(silhouette_score(X_scaled, labels))

plt.figure(figsize=(8, 5))
plt.plot(range(2, 11), sil, "o-", color="darkorange")
plt.xlabel("k")
plt.ylabel("Silhouette score")
plt.title("Silhouette score vs k")
plt.grid(alpha=0.3)
plt.tight_layout()
plt.show()
```



Step 4: Train the K-Means Model

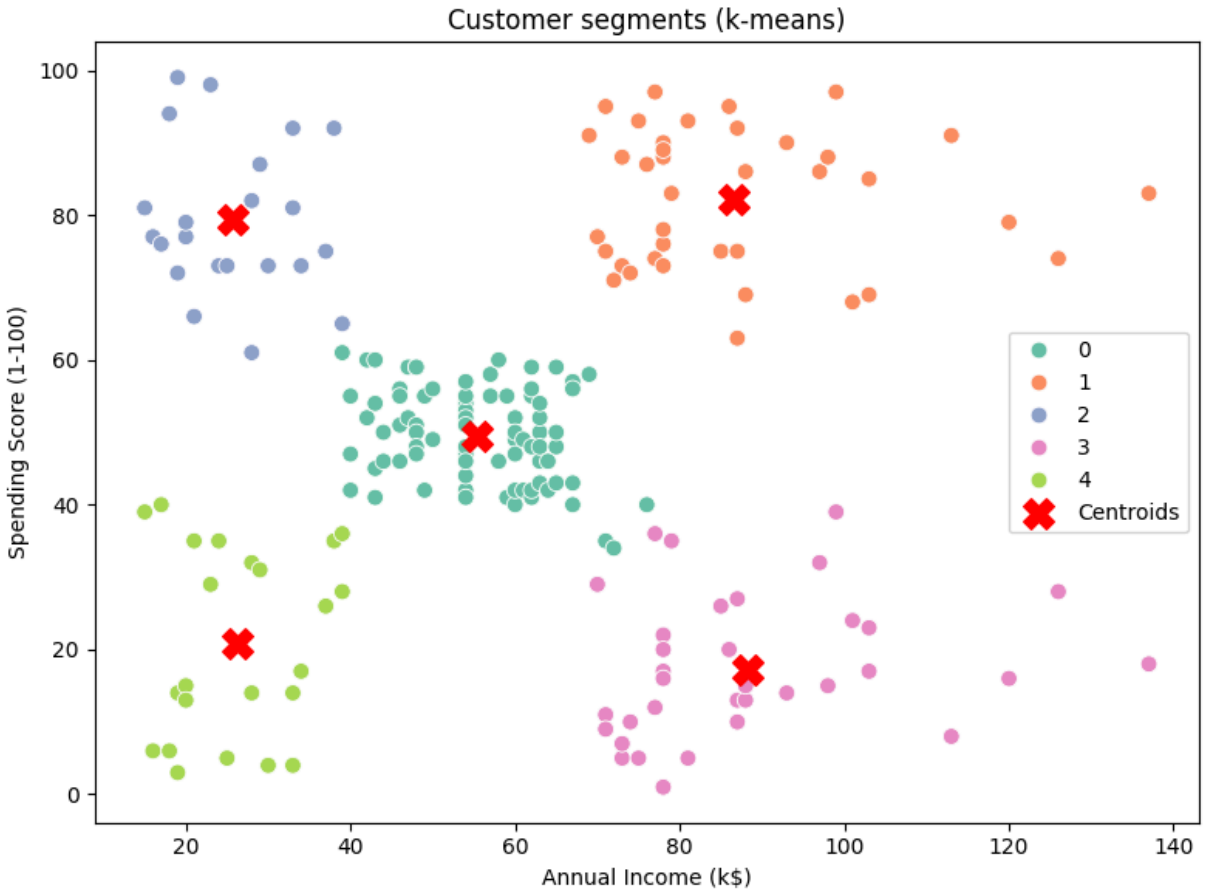
Based on the elbow curve (and optionally the silhouette plot), choose the optimal k and fit the model.

In [10]:
```
optimal_k = 5
kmeans = KMeans(n_clusters=optimal_k, random_state=42)
clusters = kmeans.fit_predict(X_scaled)

df['Cluster'] = clusters
```

Step 5: Visualise the Clusters

Create a scatter plot of **Annual Income** vs **Spending Score**, coloured by cluster. Also plot the cluster centroids.

In [11]:
```
centroids_orig = scaler.inverse_transform(kmeans.cluster_centers_)

plt.figure(figsize=(8, 6))
sns.scatterplot(
    data=df,
    x="Annual Income (k$)",
    y="Spending Score (1-100)",
    hue="Cluster",
    palette="Set2",
    s=60,
)
plt.scatter(
    centroids_orig[:, 0],
    centroids_orig[:, 1],
    c="red",
    s=200,
    marker="X",
    label="Centroids",
)
plt.title("Customer segments (k-means)")
plt.legend()
plt.tight_layout()
plt.show()
```



Step 6: Interpret the Clusters

Compute the mean of each original feature per cluster and describe the customer segments.

In [12]:
```
cluster_summary = df.groupby("Cluster")[["Age", "Annual Income (k$)", "Spenc
cluster_summary
```

Out[12]:

Cluster	Age	Annual Income (k$)	Spending Score (1-100)
0	42.716049	55.296296	49.518519
1	32.692308	86.538462	82.128205
2	25.272727	25.727273	79.363636
3	41.114286	88.200000	17.114286
4	45.217391	26.304348	20.913043

Your interpretation: *(double-click to edit)*

- Cluster 0: ...
- Cluster 1: ...

- Cluster 2: ...
- Cluster 3: ...
- Cluster 4: ...