Aim:

To perform customer segmentation using the K-Means clustering algorithm on the Mall Customers dataset based on income and spending score.

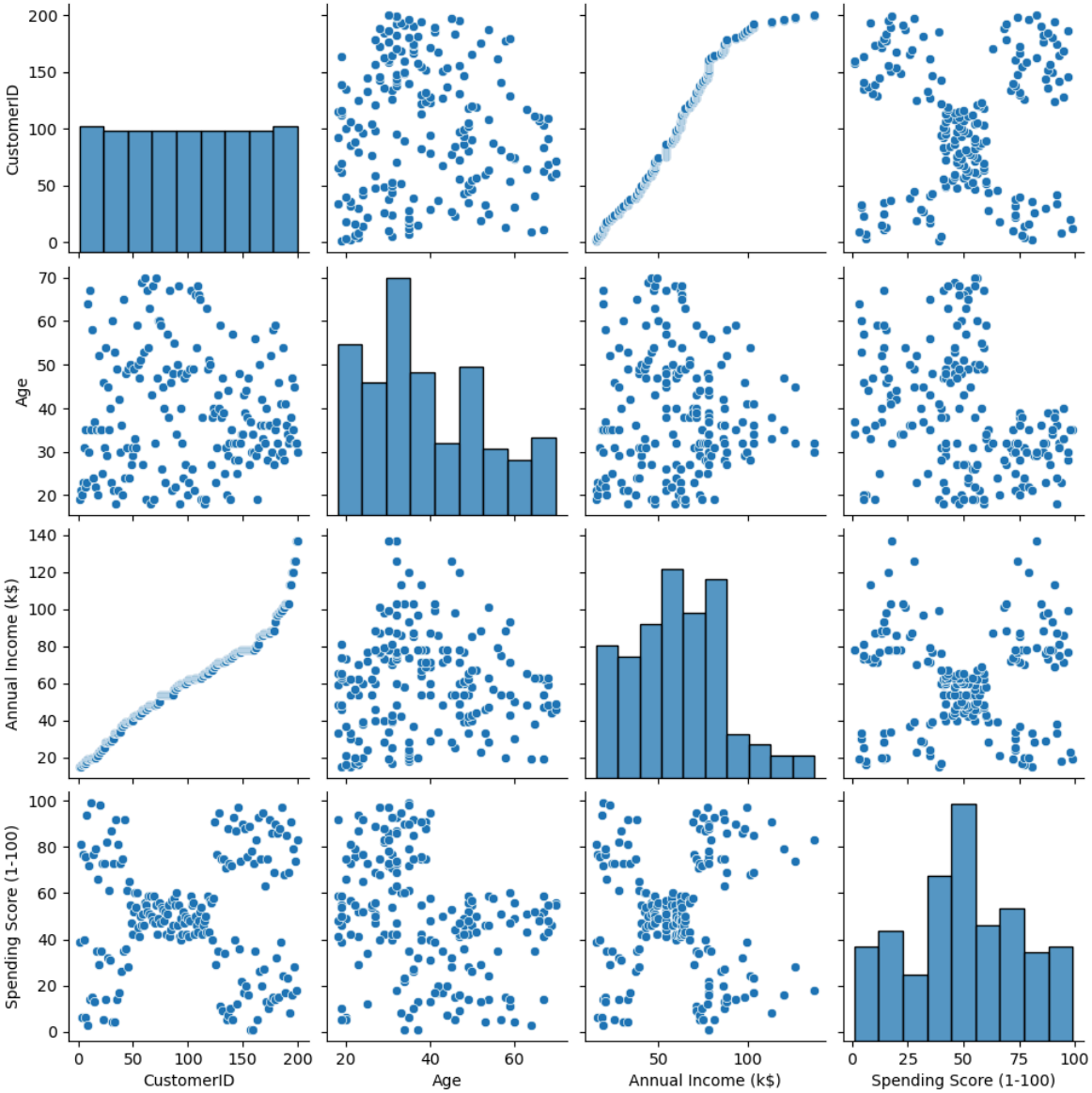
Procedure:

1. Import libraries — Import numpy, pandas, matplotlib.pyplot, seaborn, and sklearn.cluster.KMeans.
2. Load dataset — Read Mall_Customers.csv using pd.read_csv().
3. Explore data — Use df.info() and df.head() to inspect dataset structure.
4. Visualize data — Use sns.pairplot() to understand feature relationships.
5. Select features — Use columns Annual Income (k$) and Spending Score (1-100) for clustering.
6. Train K-Means model — Fit KMeans(n_clusters=5) to identify 5 customer segments.
7. Add labels — Assign predicted cluster labels to the dataset.
8. Visualize clusters — Plot clusters using FacetGrid and color them by cluster labels.
9. Determine optimal k — Apply the Elbow Method by plotting WCSS for different k values (1–9).
10. Interpret results — Choose optimal number of clusters and analyze segment behavior.

In [11]:
```
# Experiment No.: 9
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.cluster import KMeans

df = pd.read_csv('Mall_Customers.csv')

df.info()
print(df.head())

sns.pairplot(df)
plt.show()

features = df.iloc[:, [3, 4]].values

model = KMeans(n_clusters=5, n_init=10)
model.fit(features)

Final = df.iloc[:, [3, 4]].copy()
Final['label'] = model.predict(features)
print(Final.head())

sns.set_style("whitegrid")
sns.FacetGrid(Final, hue="label", height=8) \
    .map(plt.scatter, "Annual Income (k$)", "Spending Score (1-100)") \
    .add_legend()
plt.show()

features_el = df.iloc[:, [2, 3, 4]].values
wcss = []

for i in range(1, 10):
    model = KMeans(n_clusters=i, n_init=10)
    model.fit(features_el)
    wcss.append(model.inertia_)

plt.plot(range(1, 10), wcss)
plt.title("Elbow Method")
plt.xlabel("Number of Clusters (k)")
plt.ylabel("WCSS")
plt.show()
```

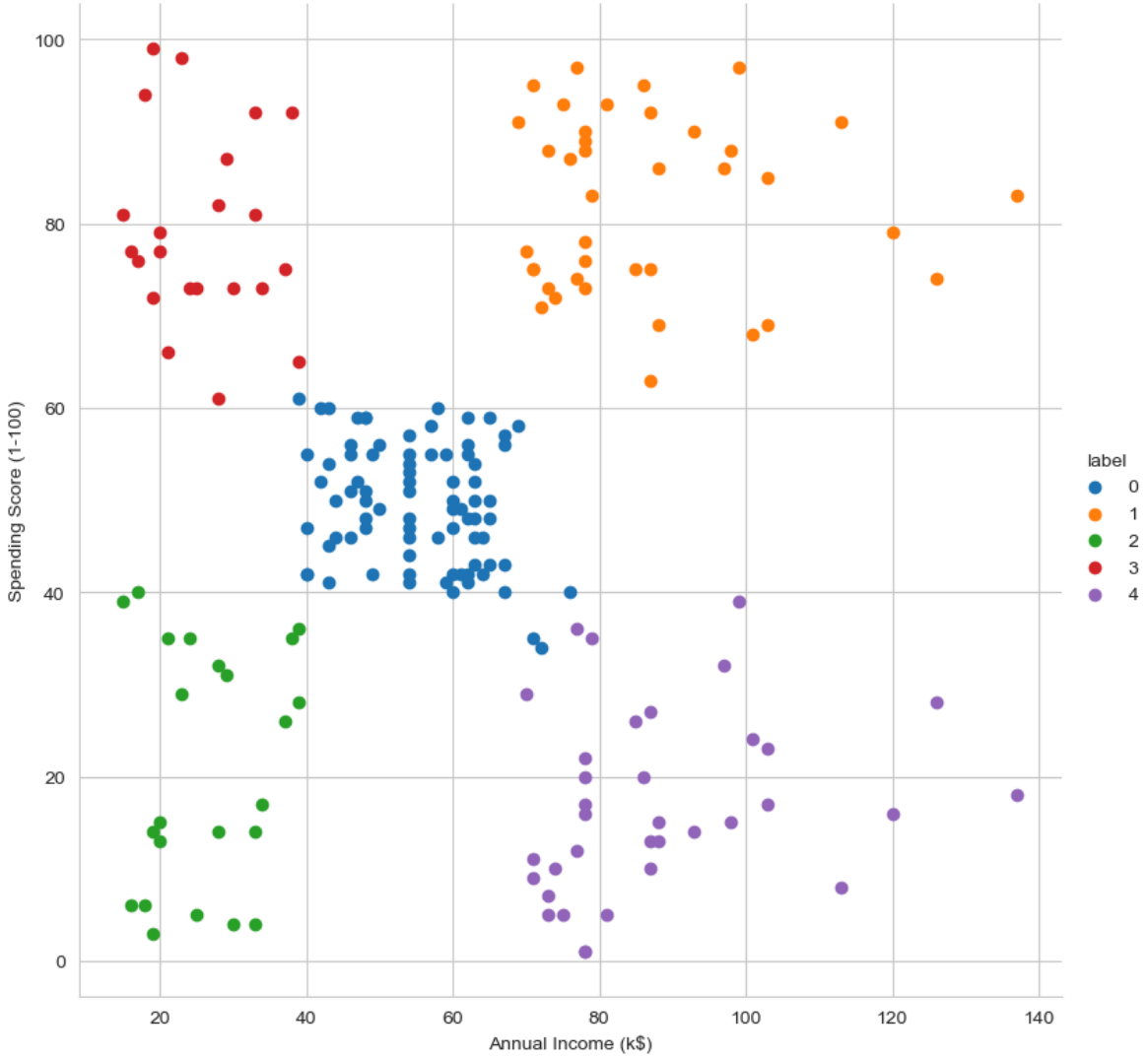
```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 200 entries, 0 to 199
Data columns (total 5 columns):
 #   Column                  Non-Null Count  Dtype
---  ------                  --------------  -----
 0   CustomerID              200 non-null    int64
 1   Gender                  200 non-null    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
```

	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

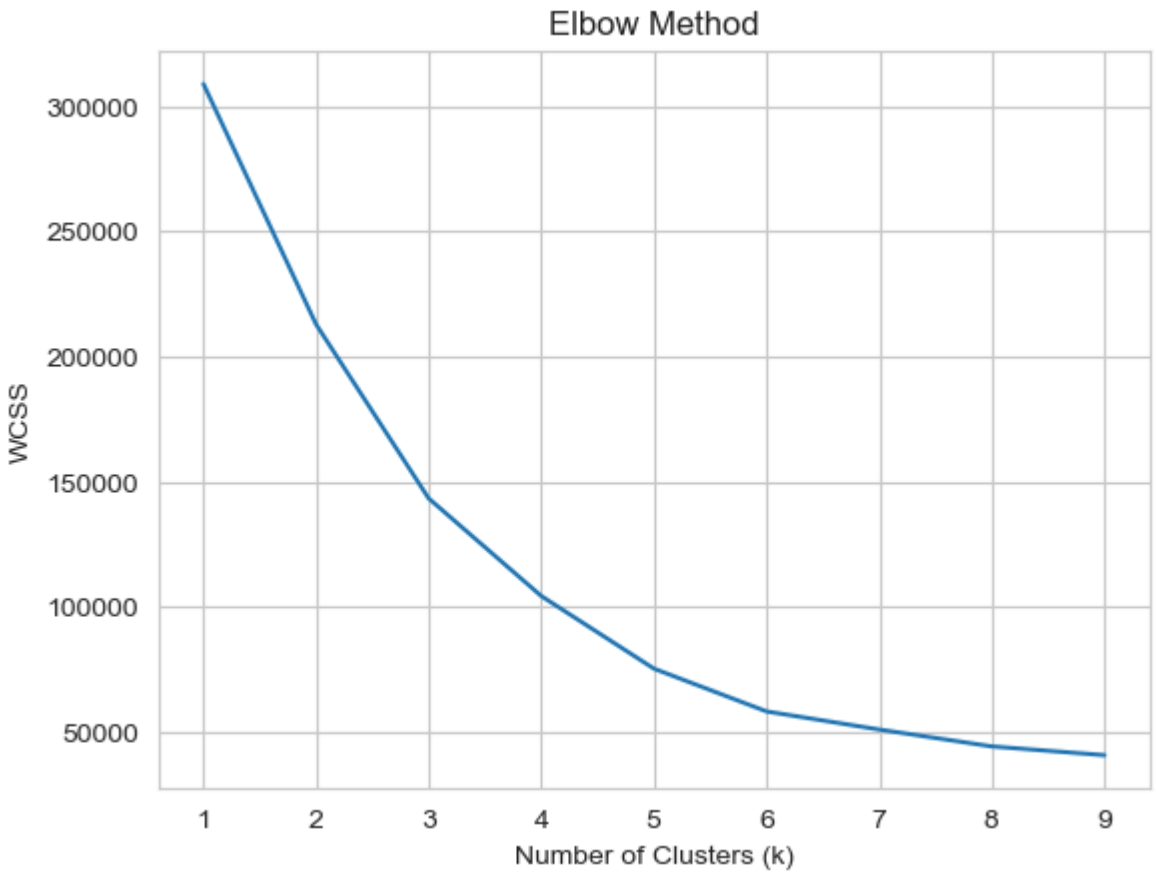


```
C:\Users\Maha Lakshmi\Downloads\New folder\Lib\site-packages\joblib\externals\lok
y\backend\context.py:136: UserWarning: Could not find the number of physical core
s for the following reason:
[WinError 2] The system cannot find the file specified
Returning the number of logical cores instead. You can silence this warning by se
tting LOKY_MAX_CPU_COUNT to the number of cores you want to use.
  warnings.warn(
  File "C:\Users\Maha Lakshmi\Downloads\New folder\Lib\site-packages\joblib\exter
nals\loky\backend\context.py", line 257, in _count_physical_cores
    cpu_info = subprocess.run(
        "wmic CPU Get NumberOfCores /Format:csv".split(),
        capture_output=True,
        text=True,
    )
  File "C:\Users\Maha Lakshmi\Downloads\New folder\Lib\subprocess.py", line 554,
in run
    with Popen(*popenargs, **kwargs) as process:
         ~~~~~^^^^^^^^^^^^^^^^^^^^^^
  File "C:\Users\Maha Lakshmi\Downloads\New folder\Lib\subprocess.py", line 1039,
in __init__
    self._execute_child(args, executable, preexec_fn, close_fds,
    ~~~~~~~~~~~~~~~~~~~^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
                        pass_fds, cwd, env,
                        ^^^^^^^^^^^^^^^^^^^
    ...<5 lines>...
                        gid, gids, uid, umask,
                        ^^^^^^^^^^^^^^^^^^^^^^
                        start_new_session, process_group)
                        ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
  File "C:\Users\Maha Lakshmi\Downloads\New folder\Lib\subprocess.py", line 1554,
in _execute_child
    hp, ht, pid, tid = _winapi.CreateProcess(executable, args,
                       ~~~~~~~~~~~~~~~~~~~~~^^^^^^^^^^^^^^^^^^
                             # no special security
                             ^^^^^^^^^^^^^^^^^^^^^^
    ...<4 lines>...
                             cwd,
                             ^^^^
                             startupinfo)
                             ^^^^^^^^^^^^
C:\Users\Maha Lakshmi\Downloads\New folder\Lib\site-packages\sklearn\cluster\_kme
ans.py:1419: UserWarning: KMeans is known to have a memory leak on Windows with M
KL, when there are less chunks than available threads. You can avoid it by settin
g the environment variable OMP_NUM_THREADS=1.
  warnings.warn(
```

	Annual Income (k$)	Spending Score (1-100)	label
0	15	39	2
1	15	81	3
2	16	6	2
3	16	77	3
4	17	40	2


Spending Score (1-100)
Annual Income (k$)
label
0
1
2
3
4

```
C:\Users\Maha Lakshmi\Downloads\New folder\Lib\site-packages\sklearn\cluster\_kme
ans.py:1419: UserWarning: KMeans is known to have a memory leak on Windows with M
KL, when there are less chunks than available threads. You can avoid it by settin
g the environment variable OMP_NUM_THREADS=1.
  warnings.warn(
C:\Users\Maha Lakshmi\Downloads\New folder\Lib\site-packages\sklearn\cluster\_kme
ans.py:1419: UserWarning: KMeans is known to have a memory leak on Windows with M
KL, when there are less chunks than available threads. You can avoid it by settin
g the environment variable OMP_NUM_THREADS=1.
  warnings.warn(
C:\Users\Maha Lakshmi\Downloads\New folder\Lib\site-packages\sklearn\cluster\_kme
ans.py:1419: UserWarning: KMeans is known to have a memory leak on Windows with M
KL, when there are less chunks than available threads. You can avoid it by settin
g the environment variable OMP_NUM_THREADS=1.
  warnings.warn(
C:\Users\Maha Lakshmi\Downloads\New folder\Lib\site-packages\sklearn\cluster\_kme
ans.py:1419: UserWarning: KMeans is known to have a memory leak on Windows with M
KL, when there are less chunks than available threads. You can avoid it by settin
g the environment variable OMP_NUM_THREADS=1.
  warnings.warn(
C:\Users\Maha Lakshmi\Downloads\New folder\Lib\site-packages\sklearn\cluster\_kme
ans.py:1419: UserWarning: KMeans is known to have a memory leak on Windows with M
KL, when there are less chunks than available threads. You can avoid it by settin
g the environment variable OMP_NUM_THREADS=1.
  warnings.warn(
C:\Users\Maha Lakshmi\Downloads\New folder\Lib\site-packages\sklearn\cluster\_kme
ans.py:1419: UserWarning: KMeans is known to have a memory leak on Windows with M
KL, when there are less chunks than available threads. You can avoid it by settin
g the environment variable OMP_NUM_THREADS=1.
  warnings.warn(
C:\Users\Maha Lakshmi\Downloads\New folder\Lib\site-packages\sklearn\cluster\_kme
ans.py:1419: UserWarning: KMeans is known to have a memory leak on Windows with M
KL, when there are less chunks than available threads. You can avoid it by settin
g the environment variable OMP_NUM_THREADS=1.
  warnings.warn(
C:\Users\Maha Lakshmi\Downloads\New folder\Lib\site-packages\sklearn\cluster\_kme
ans.py:1419: UserWarning: KMeans is known to have a memory leak on Windows with M
KL, when there are less chunks than available threads. You can avoid it by settin
g the environment variable OMP_NUM_THREADS=1.
  warnings.warn(
C:\Users\Maha Lakshmi\Downloads\New folder\Lib\site-packages\sklearn\cluster\_kme
ans.py:1419: UserWarning: KMeans is known to have a memory leak on Windows with M
KL, when there are less chunks than available threads. You can avoid it by settin
g the environment variable OMP_NUM_THREADS=1.
  warnings.warn(
```


Elbow Method
WCSS
300000
250000
200000
150000
100000
50000
1
2
3
4
5
6
7
8
9
Number of Clusters (k)

In [ ]:

Result:

The K-Means algorithm effectively grouped customers into five segments based on income and spending behavior.

The Elbow Method confirmed that k = 5 is the optimal number of clusters.

The resulting customer groups help in identifying spending patterns and income-based segmentation for targeted marketing.