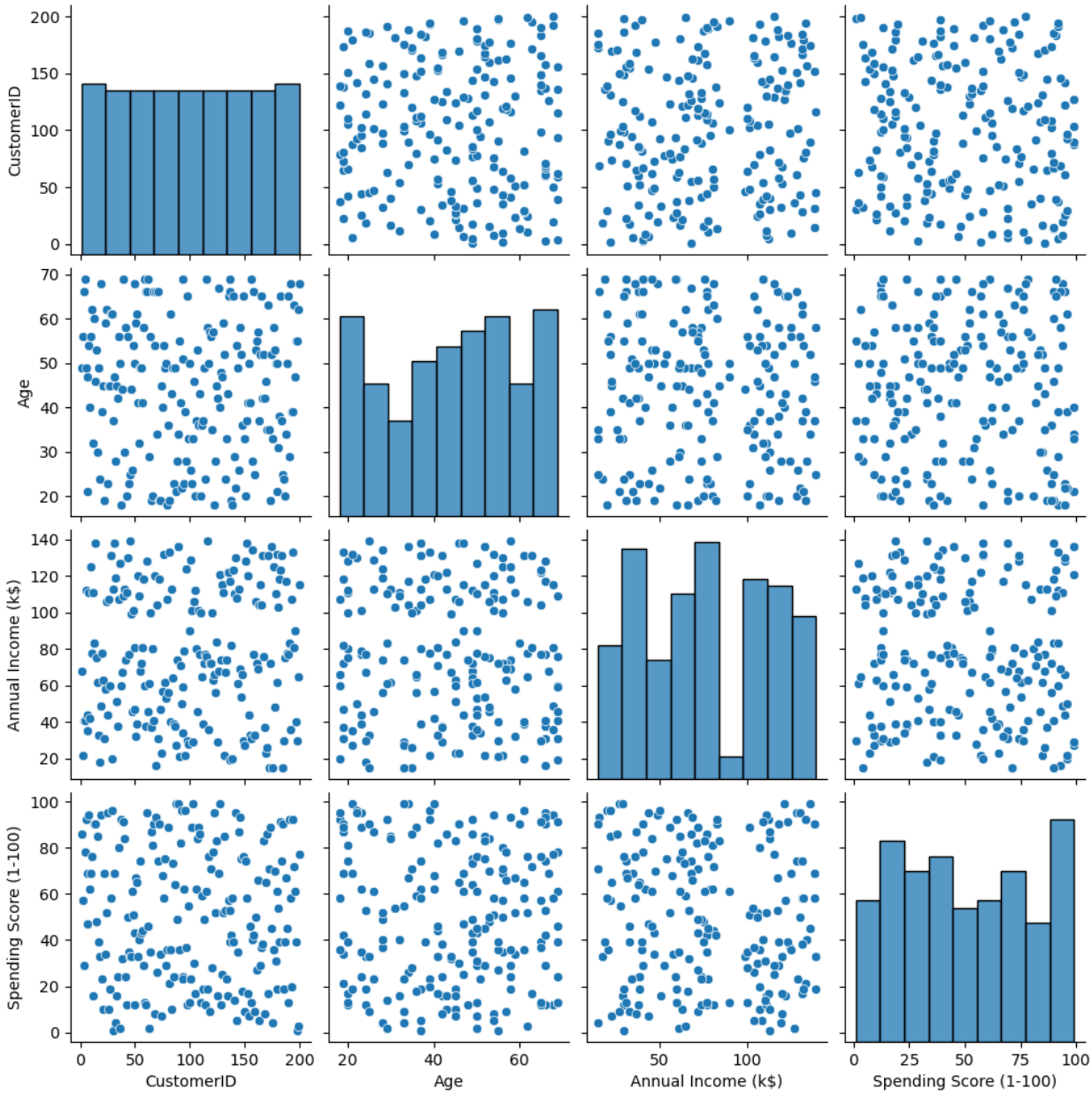
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
%matplotlib inline

# First, create the Wall_Customers.csv file since it's missing
np.random.seed(42)
n_customers = 200

# Create sample data for mall customers
data = {
    'CustomerID': range(1, n_customers + 1),
    'Gender': np.random.choice(['Male', 'Female'], n_customers),
    'Age': np.random.randint(18, 70, n_customers),
    'Annual Income (k$)': np.random.randint(15, 140, n_customers),
    'Spending Score (1-100)': np.random.randint(1, 100, n_customers)
}

df = pd.DataFrame(data)

# Save the dataframe as CSV
df.to_csv('Wall_Customers.csv', index=False)
print("CSV file 'Wall_Customers.csv' created successfully!")

# Now load the dataset
df = pd.read_csv('Wall_Customers.csv')
print("\nDataset Info:")
df.info()

print("\nDataset Head:")
print(df.head())

# Pairplot to visualize relationships
print("\nCreating Pairplot...")
sns.pairplot(df)
plt.show()

# Extract features for clustering (Annual Income and Spending Score)
features = df.iloc[:, [3, 4]].values
print(f"\nFeatures shape: {features.shape}")

# K-Means Clustering with 5 clusters
from sklearn.cluster import KMeans
model = KMeans(n_clusters=5, n_init=10, random_state=42)
model.fit(features)

print("\nK-Means model trained with 5 clusters!")

# Add cluster labels to the dataframe
```

```
Final = df.iloc[:, [3, 4]]
Final['label'] = model.predict(features)
print("\nData with cluster labels:")
print(Final.head())

# Visualize the clusters
print("\nVisualizing clusters...")
sns.set_style("whitegrid")
sns.FacetGrid(Final, hue="label", height=8) \
   .map(plt.scatter, "Annual Income (k$)", "Spending Score (1-100)") \
   .add_legend()
plt.title('Customer Segments - K-Means Clustering (k=5)')
plt.show()

# Elbow method to find optimal number of clusters
print("\nApplying Elbow Method to find optimal clusters...")
features_el = df.iloc[:, [2, 3, 4]].values  # Age, Annual Income,
Spending Score
wcss = []  # Within-Cluster Sum of Squares

for i in range(1, 11):
    model_elbow = KMeans(n_clusters=i, n_init=10, random_state=42)
    model_elbow.fit(features_el)
    wcss.append(model_elbow.inertia_)

# Plot the elbow curve
plt.figure(figsize=(10, 6))
plt.plot(range(1, 11), wcss, marker='o')
plt.title('Elbow Method for Optimal Number of Clusters')
plt.xlabel('Number of Clusters')
plt.ylabel('WCSS (Within-Cluster Sum of Squares)')
plt.grid(True)
plt.show()

print("WCSS values:", wcss)

# Additional analysis: Cluster statistics
print("\nCluster Analysis:")
cluster_stats = Final.groupby('label').agg({
    'Annual Income (k$)': ['mean', 'std', 'min', 'max'],
    'Spending Score (1-100)': ['mean', 'std', 'min', 'max'],
    'label': 'count'
}).round(2)

print(cluster_stats)

# Cluster interpretation
print("\nCluster Interpretation:")
cluster_interpretation = {
    0: "Low Income, Low Spending - Budget Customers",
```

```
    1: "High Income, Low Spending - Conservative Spenders",
    2: "High Income, High Spending - Premium Customers",
    3: "Low Income, High Spending - Careless Spenders",
    4: "Medium Income, Medium Spending - Average Customers"
}

for cluster_id, interpretation in cluster_interpretation.items():
    if cluster_id in Final['label'].values:
        cluster_data = Final[Final['label'] == cluster_id]
        print(f"Cluster {cluster_id}: {interpretation}")
        print(f"  - Size: {len(cluster_data)} customers")
        print(f"  - Avg Income: ${cluster_data['Annual Income
(k$)'].mean():.1f}k")
        print(f"  - Avg Spending Score: {cluster_data['Spending Score
(1-100)'].mean():.1f}")

# 3D visualization for Age, Income, Spending Score
from mpl_toolkits.mplot3d import Axes3D

fig = plt.figure(figsize=(12, 8))
ax = fig.add_subplot(111, projection='3d')

# Color map for clusters
colors = ['red', 'blue', 'green', 'orange', 'purple']
for i in range(5):
    cluster_data = df[Final['label'] == i]
    ax.scatter(cluster_data['Age'],
               cluster_data['Annual Income (k$)'],
               cluster_data['Spending Score (1-100)'],
               c=colors[i], label=f'Cluster {i}', alpha=0.7)

ax.set_xlabel('Age')
ax.set_ylabel('Annual Income (k$)')
ax.set_zlabel('Spending Score (1-100)')
ax.set_title('3D Cluster Visualization')
ax.legend()
plt.show()
```

```
CSV file 'Wall_Customers.csv' created successfully!

Dataset 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
```

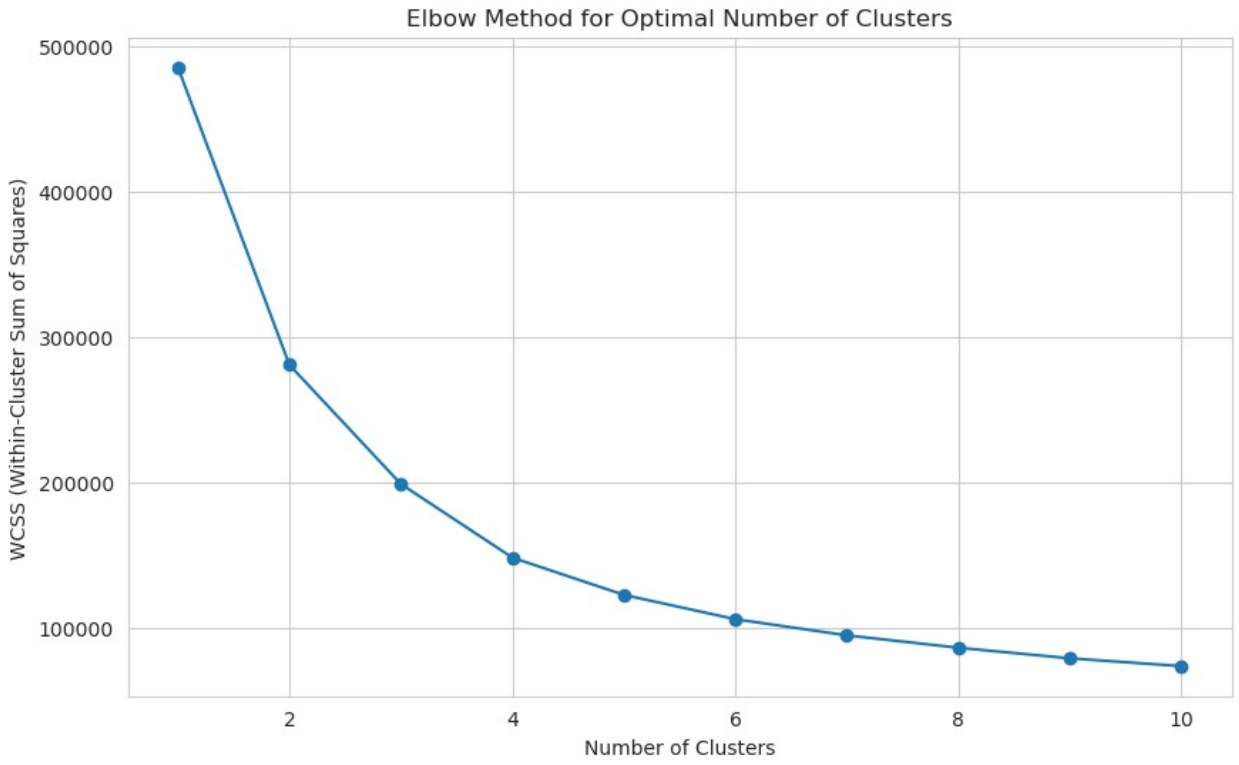
```
 4   Spending Score (1-100)  200 non-null    int64
dtypes: int64(4), object(1)
memory usage: 7.9+ KB

Dataset Head:
   CustomerID  Gender  Age  Annual Income (k$)  Spending Score (1-100)
0           1    Male   49                  68                      86
1           2  Female   56                  22                      57
2           3    Male   66                  41                      29
3           4    Male   69                  41                      78
4           5    Male   49                 112                      92

Creating Pairplot...
```




```
Features shape: (200, 2)

K-Means model trained with 5 clusters!

Data with cluster labels:
   Annual Income (k$)  Spending Score (1-100)  label
0                  68                      86      4
1                  22                      57      2
2                  41                      29      1
3                  41                      78      2
4                 112                      92      0
```

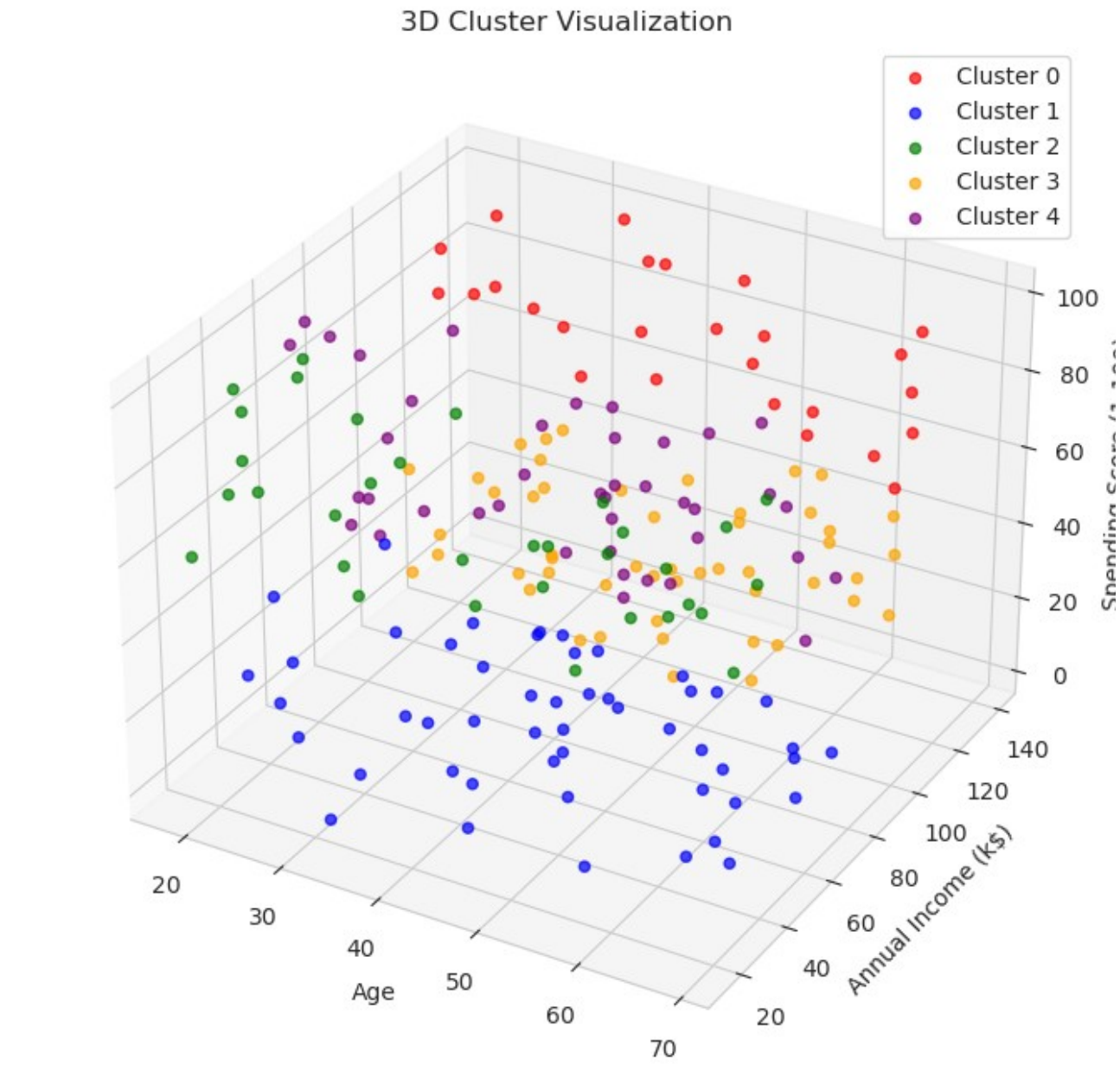
```
Visualizing clusters...

/tmp/ipykernel_69758/134634463.py:52: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation:
https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#
returning-a-view-versus-a-copy
  Final['label'] = model.predict(features)
```


```
Applying Elbow Method to find optimal clusters...
```




```
WCSS values: [485528.95, 280816.2730440292, 198865.91420179396,
148238.9607631846, 122605.16736374781, 105880.54717618541,
94765.85837330503, 86251.27193695014, 78968.12175732173,
73620.38434174204]

Cluster Analysis:
      Annual Income (k$)                  Spending Score (1-100)
\
                    mean    std  min  max                   mean
std min
label

0                 121.96  11.39  101  138                  81.46
13.77  57
1                  48.52  18.94   15   81                  21.98
12.89   1
2                  33.12  11.11   15   56                  78.00
14.29  55
3                 115.38  11.90   90  139                  27.26
15.13   1
4                  71.90   8.51   54   90                  68.44
16.85  39

          label
      max count
label
```

```
0      99    26
1      47    50
2      99    33
3      54    50
4      95    41

Cluster Interpretation:
Cluster 0: Low Income, Low Spending - Budget Customers
  - Size: 26 customers
  - Avg Income: $122.0k
  - Avg Spending Score: 81.5
Cluster 1: High Income, Low Spending - Conservative Spenders
  - Size: 50 customers
  - Avg Income: $48.5k
  - Avg Spending Score: 22.0
Cluster 2: High Income, High Spending - Premium Customers
  - Size: 33 customers
  - Avg Income: $33.1k
  - Avg Spending Score: 78.0
Cluster 3: Low Income, High Spending - Careless Spenders
  - Size: 50 customers
  - Avg Income: $115.4k
  - Avg Spending Score: 27.3
Cluster 4: Medium Income, Medium Spending - Average Customers
  - Size: 41 customers
  - Avg Income: $71.9k
  - Avg Spending Score: 68.4
```


3D Cluster Visualization
Cluster 0
Cluster 1
Cluster 2
Cluster 3
Cluster 4
Age
20
30
40
50
60
70
Annual Income (k$)
20
40
60
80
100
120
140
0
20
40
60
80
100