In [6]:
```
import pandas as pd
```

In [7]:
```
import numpy as np
from sklearn.preprocessing import StandardScaler
```

In [8]:
```
df=pd.read_csv(r"C:\Users\vijay\OneDrive\Desktop\Jupyter projects\Customer_data.
df
```

Out[8]:

	Serie no.	Customer_ID	Age	Annual Income	Spending score
0	1	8	50	32725	11
1	2	9	28	71176	6
2	3	10	45	56493	9
3	4	10	37	37506	15
4	5	1	30	47339	13
5	6	3	50	34325	15
6	7	1	28	86041	8
7	8	4	42	46171	12
8	9	8	35	43339	14
9	10	8	32	34406	11

In [9]:
```
print(df.head())  # This should display the first few rows
```

```
   Serie no.  Customer_ID  Age  Annual Income  Spending score
0          1            8   50          32725              11
1          2            9   28          71176               6
2          3           10   45          56493               9
3          4           10   37          37506              15
4          5            1   30          47339              13
```

In [10]:
```
df.shape
```

Out[10]:
```
(10, 5)
```

In [11]:
```
df.isnull().sum()
```

Out[11]:
```
Serie no.          0
Customer_ID        0
Age                0
Annual Income      0
Spending score     0
dtype: int64
```

In [12]:
```
df.duplicated().sum()
```

Out[12]:
```
0
```

In [13]:
```
df.dtypes
```

Out[13]:
```
Serie no.          int64
Customer_ID        int64
Age                int64
Annual Income      int64
Spending score     int64
dtype: object
```

In [14]:
```
df.describe()
```

Out[14]:

	Serie no.	Customer_ID	Age	Annual Income	Spending score
count	10.00000	10.000000	10.000000	10.000000	10.000000
mean	5.50000	6.200000	37.700000	48952.100000	11.400000
std	3.02765	3.583915	8.577101	17600.725891	3.025815
min	1.00000	1.000000	28.000000	32725.000000	6.000000
25%	3.25000	3.250000	30.500000	35181.000000	9.500000
50%	5.50000	8.000000	36.000000	44755.000000	11.500000
75%	7.75000	8.750000	44.250000	54204.500000	13.750000
max	10.00000	10.000000	50.000000	86041.000000	15.000000

In [15]:
```
from sklearn.preprocessing import StandardScaler, MinMaxScaler
```

In [16]:
```
features=['Age', 'Annual Income','Spending score']
```

In [17]:
```
features=df.select_dtypes(include=['int64','float64']).columns
```

In [18]:
```
scaler = StandardScaler()
scaled_data = scaler.fit_transform(df[features])
```

In [19]:
```
scaler = MinMaxScaler()
scaled_data = scaler.fit_transform(df[features])
```

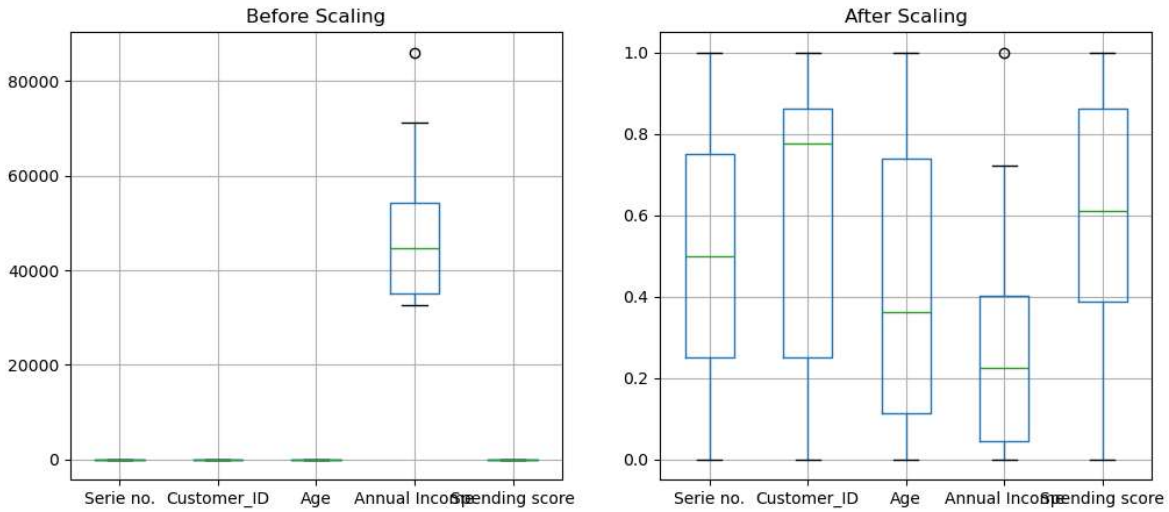
In [20]:
```
scaled_df = pd.DataFrame(scaled_data, columns=features)
```

In [21]:
```
scaled_df.describe()
```

Out[21]:

	Serie no.	Customer_ID	Age	Annual Income	Spending score
count	10.000000	10.000000	10.000000	10.000000	10.000000
mean	0.500000	0.577778	0.440909	0.304357	0.600000
std	0.336406	0.398213	0.389868	0.330121	0.336202
min	0.000000	0.000000	0.000000	0.000000	0.000000
25%	0.250000	0.250000	0.113636	0.046065	0.388889
50%	0.500000	0.777778	0.363636	0.225636	0.611111
75%	0.750000	0.861111	0.738636	0.402872	0.861111
max	1.000000	1.000000	1.000000	1.000000	1.000000

In [22]:
```
import matplotlib.pyplot as plt

plt.figure(figsize=(12,5))
plt.subplot(1, 2, 1)
plt.title("Before Scaling")
df[features].boxplot()

plt.subplot(1, 2, 2)
plt.title("After Scaling")
scaled_df.boxplot()
plt.show()
```



In [24]:
```
from sklearn.cluster import KMeans
from sklearn.metrics import silhouette_score
import matplotlib.pyplot as plt
import seaborn as sns
```

In [25]:
```
wcss = []
K = range(1, 11)

for k in K:
    kmeans = KMeans(n_clusters=k, random_state=42)
    kmeans.fit(scaled_df)
    wcss.append(kmeans.inertia_)  # WCSS
```

```
C:\ProgramData\anaconda3\Lib\site-packages\sklearn\cluster\_kmeans.py:1429: UserW
arning: KMeans is known to have a memory leak on Windows with MKL, when there are
less chunks than available threads. You can avoid it by setting the environment v
ariable OMP_NUM_THREADS=1.
  warnings.warn(
C:\ProgramData\anaconda3\Lib\site-packages\sklearn\cluster\_kmeans.py:1429: UserW
arning: KMeans is known to have a memory leak on Windows with MKL, when there are
less chunks than available threads. You can avoid it by setting the environment v
ariable OMP_NUM_THREADS=1.
  warnings.warn(
C:\ProgramData\anaconda3\Lib\site-packages\sklearn\cluster\_kmeans.py:1429: UserW
arning: KMeans is known to have a memory leak on Windows with MKL, when there are
less chunks than available threads. You can avoid it by setting the environment v
ariable OMP_NUM_THREADS=1.
  warnings.warn(
C:\ProgramData\anaconda3\Lib\site-packages\sklearn\cluster\_kmeans.py:1429: UserW
arning: KMeans is known to have a memory leak on Windows with MKL, when there are
less chunks than available threads. You can avoid it by setting the environment v
ariable OMP_NUM_THREADS=1.
  warnings.warn(
C:\ProgramData\anaconda3\Lib\site-packages\sklearn\cluster\_kmeans.py:1429: UserW
arning: KMeans is known to have a memory leak on Windows with MKL, when there are
less chunks than available threads. You can avoid it by setting the environment v
ariable OMP_NUM_THREADS=1.
  warnings.warn(
C:\ProgramData\anaconda3\Lib\site-packages\sklearn\cluster\_kmeans.py:1429: UserW
arning: KMeans is known to have a memory leak on Windows with MKL, when there are
less chunks than available threads. You can avoid it by setting the environment v
ariable OMP_NUM_THREADS=1.
  warnings.warn(
C:\ProgramData\anaconda3\Lib\site-packages\sklearn\cluster\_kmeans.py:1429: UserW
arning: KMeans is known to have a memory leak on Windows with MKL, when there are
less chunks than available threads. You can avoid it by setting the environment v
ariable OMP_NUM_THREADS=1.
  warnings.warn(
C:\ProgramData\anaconda3\Lib\site-packages\sklearn\cluster\_kmeans.py:1429: UserW
arning: KMeans is known to have a memory leak on Windows with MKL, when there are
less chunks than available threads. You can avoid it by setting the environment v
ariable OMP_NUM_THREADS=1.
  warnings.warn(
C:\ProgramData\anaconda3\Lib\site-packages\sklearn\cluster\_kmeans.py:1429: UserW
arning: KMeans is known to have a memory leak on Windows with MKL, when there are
less chunks than available threads. You can avoid it by setting the environment v
ariable OMP_NUM_THREADS=1.
  warnings.warn(
C:\ProgramData\anaconda3\Lib\site-packages\sklearn\cluster\_kmeans.py:1429: UserW
arning: KMeans is known to have a memory leak on Windows with MKL, when there are
less chunks than available threads. You can avoid it by setting the environment v
ariable OMP_NUM_THREADS=1.
  warnings.warn(
```

In [26]:

```
for k in range(2, 11):
    kmeans = KMeans(n_clusters=k, random_state=42)
    labels = kmeans.fit_predict(scaled_df)
    score = silhouette_score(scaled_df, labels)
    print(f'Silhouette Score for k={k}: {score:.3f}')
```

```
C:\ProgramData\anaconda3\Lib\site-packages\sklearn\cluster\_kmeans.py:1429: UserW
arning: KMeans is known to have a memory leak on Windows with MKL, when there are
less chunks than available threads. You can avoid it by setting the environment v
ariable OMP_NUM_THREADS=1.
  warnings.warn(
C:\ProgramData\anaconda3\Lib\site-packages\sklearn\cluster\_kmeans.py:1429: UserW
arning: KMeans is known to have a memory leak on Windows with MKL, when there are
less chunks than available threads. You can avoid it by setting the environment v
ariable OMP_NUM_THREADS=1.
  warnings.warn(
C:\ProgramData\anaconda3\Lib\site-packages\sklearn\cluster\_kmeans.py:1429: UserW
arning: KMeans is known to have a memory leak on Windows with MKL, when there are
less chunks than available threads. You can avoid it by setting the environment v
ariable OMP_NUM_THREADS=1.
  warnings.warn(
C:\ProgramData\anaconda3\Lib\site-packages\sklearn\cluster\_kmeans.py:1429: UserW
arning: KMeans is known to have a memory leak on Windows with MKL, when there are
less chunks than available threads. You can avoid it by setting the environment v
ariable OMP_NUM_THREADS=1.
  warnings.warn(
C:\ProgramData\anaconda3\Lib\site-packages\sklearn\cluster\_kmeans.py:1429: UserW
arning: KMeans is known to have a memory leak on Windows with MKL, when there are
less chunks than available threads. You can avoid it by setting the environment v
ariable OMP_NUM_THREADS=1.
  warnings.warn(
C:\ProgramData\anaconda3\Lib\site-packages\sklearn\cluster\_kmeans.py:1429: UserW
arning: KMeans is known to have a memory leak on Windows with MKL, when there are
less chunks than available threads. You can avoid it by setting the environment v
ariable OMP_NUM_THREADS=1.
  warnings.warn(
```

```
Silhouette Score for k=2: 0.242
Silhouette Score for k=3: 0.172
Silhouette Score for k=4: 0.149
Silhouette Score for k=5: 0.078
Silhouette Score for k=6: 0.078
Silhouette Score for k=7: 0.187
Silhouette Score for k=8: 0.130
Silhouette Score for k=9: 0.074
```

```
C:\ProgramData\anaconda3\Lib\site-packages\sklearn\cluster\_kmeans.py:1429: UserW
arning: KMeans is known to have a memory leak on Windows with MKL, when there are
less chunks than available threads. You can avoid it by setting the environment v
ariable OMP_NUM_THREADS=1.
  warnings.warn(
C:\ProgramData\anaconda3\Lib\site-packages\sklearn\cluster\_kmeans.py:1429: UserW
arning: KMeans is known to have a memory leak on Windows with MKL, when there are
less chunks than available threads. You can avoid it by setting the environment v
ariable OMP_NUM_THREADS=1.
  warnings.warn(
C:\ProgramData\anaconda3\Lib\site-packages\sklearn\cluster\_kmeans.py:1429: UserW
arning: KMeans is known to have a memory leak on Windows with MKL, when there are
less chunks than available threads. You can avoid it by setting the environment v
ariable OMP_NUM_THREADS=1.
  warnings.warn(
```

```
---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
Cell In[26], line 4
      2 kmeans = KMeans(n_clusters=k, random_state=42)
      3 labels = kmeans.fit_predict(scaled_df)
----> 4 score = silhouette_score(scaled_df, labels)
      5 print(f'Silhouette Score for k={k}: {score:.3f}')

File C:\ProgramData\anaconda3\Lib\site-packages\sklearn\utils\_param_validation.p
y:213, in validate_params.<locals>.decorator.<locals>.wrapper(*args, **kwargs)
    207 try:
    208     with config_context(
    209         skip_parameter_validation=(
    210             prefer_skip_nested_validation or global_skip_validation
    211         )
    212     ):
--> 213         return func(*args, **kwargs)
    214 except InvalidParameterError as e:
    215     # When the function is just a wrapper around an estimator, we allow
    216     # the function to delegate validation to the estimator, but we replac
e
    217     # the name of the estimator by the name of the function in the error
    218     # message to avoid confusion.
    219     msg = re.sub(
    220         r"parameter of \w+ must be",
    221         f"parameter of {func.__qualname__} must be",
    222         str(e),
    223     )

File C:\ProgramData\anaconda3\Lib\site-packages\sklearn\metrics\cluster\_unsuperv
ised.py:141, in silhouette_score(X, labels, metric, sample_size, random_state, **
kwds)
    139     else:
    140         X, labels = X[indices], labels[indices]
--> 141 return np.mean(silhouette_samples(X, labels, metric=metric, **kwds))

File C:\ProgramData\anaconda3\Lib\site-packages\sklearn\utils\_param_validation.p
y:186, in validate_params.<locals>.decorator.<locals>.wrapper(*args, **kwargs)
    184 global_skip_validation = get_config()["skip_parameter_validation"]
    185 if global_skip_validation:
--> 186     return func(*args, **kwargs)
    188 func_sig = signature(func)
    190 # Map *args/**kwargs to the function signature

File C:\ProgramData\anaconda3\Lib\site-packages\sklearn\metrics\cluster\_unsuperv
ised.py:299, in silhouette_samples(X, labels, metric, **kwds)
    297 n_samples = len(labels)
    298 label_freqs = np.bincount(labels)
--> 299 check_number_of_labels(len(le.classes_), n_samples)
    301 kwds["metric"] = metric
    302 reduce_func = functools.partial(
    303     _silhouette_reduce, labels=labels, label_freqs=label_freqs
    304 )

File C:\ProgramData\anaconda3\Lib\site-packages\sklearn\metrics\cluster\_unsuperv
ised.py:38, in check_number_of_labels(n_labels, n_samples)
     27 """Check that number of labels are valid.
     28
     29 Parameters
   (...)
```

```
     35     Number of samples.
     36 """
     37 if not 1 < n_labels < n_samples:
---> 38     raise ValueError(
     39         "Number of labels is %d. Valid values are 2 to n_samples - 1 (inc
lusive)"
     40         % n_labels
     41     )

ValueError: Number of labels is 10. Valid values are 2 to n_samples - 1 (inclusiv
e)
```

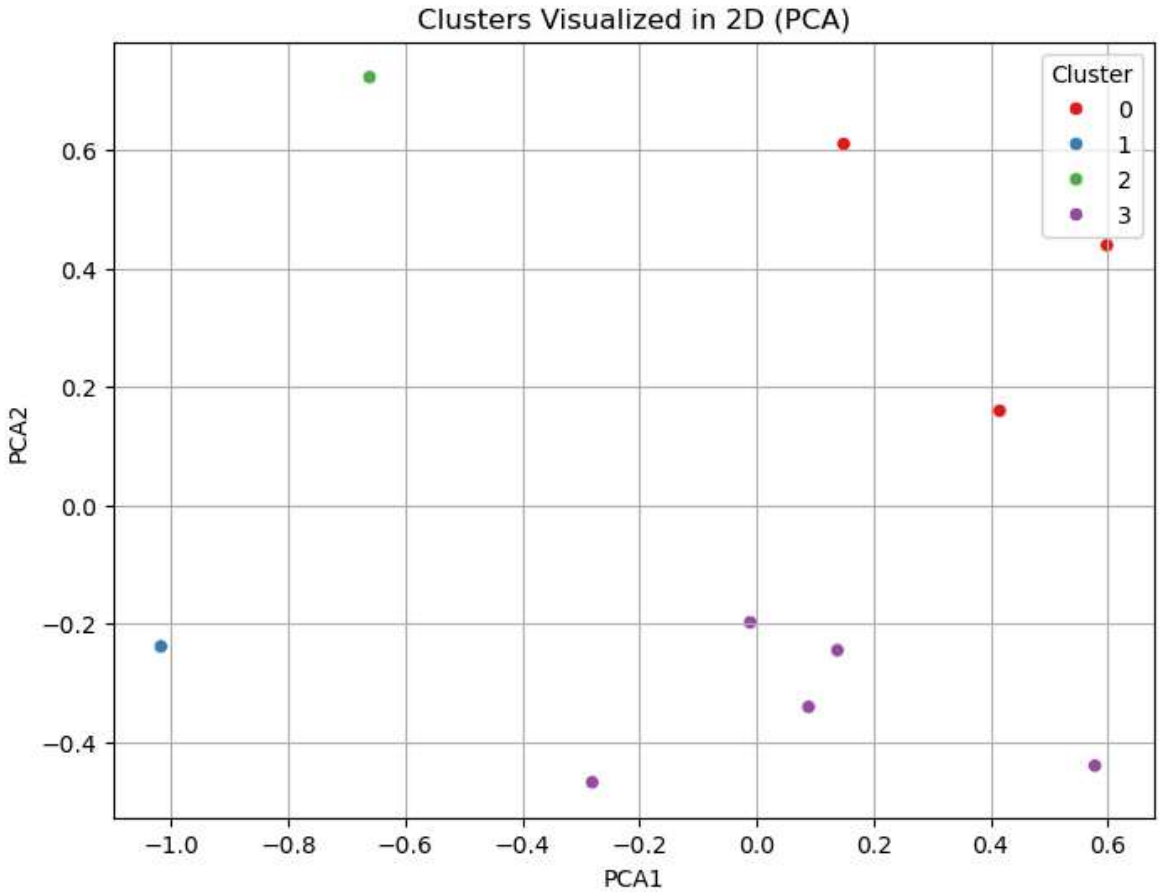
In [27]:
```
optimal_k = 4  # change if your elbow point is different
kmeans = KMeans(n_clusters=optimal_k, random_state=42)
df['Cluster'] = kmeans.fit_predict(scaled_df)
```

```
C:\ProgramData\anaconda3\Lib\site-packages\sklearn\cluster\_kmeans.py:1429: UserW
arning: KMeans is known to have a memory leak on Windows with MKL, when there are
less chunks than available threads. You can avoid it by setting the environment v
ariable OMP_NUM_THREADS=1.
  warnings.warn(
```

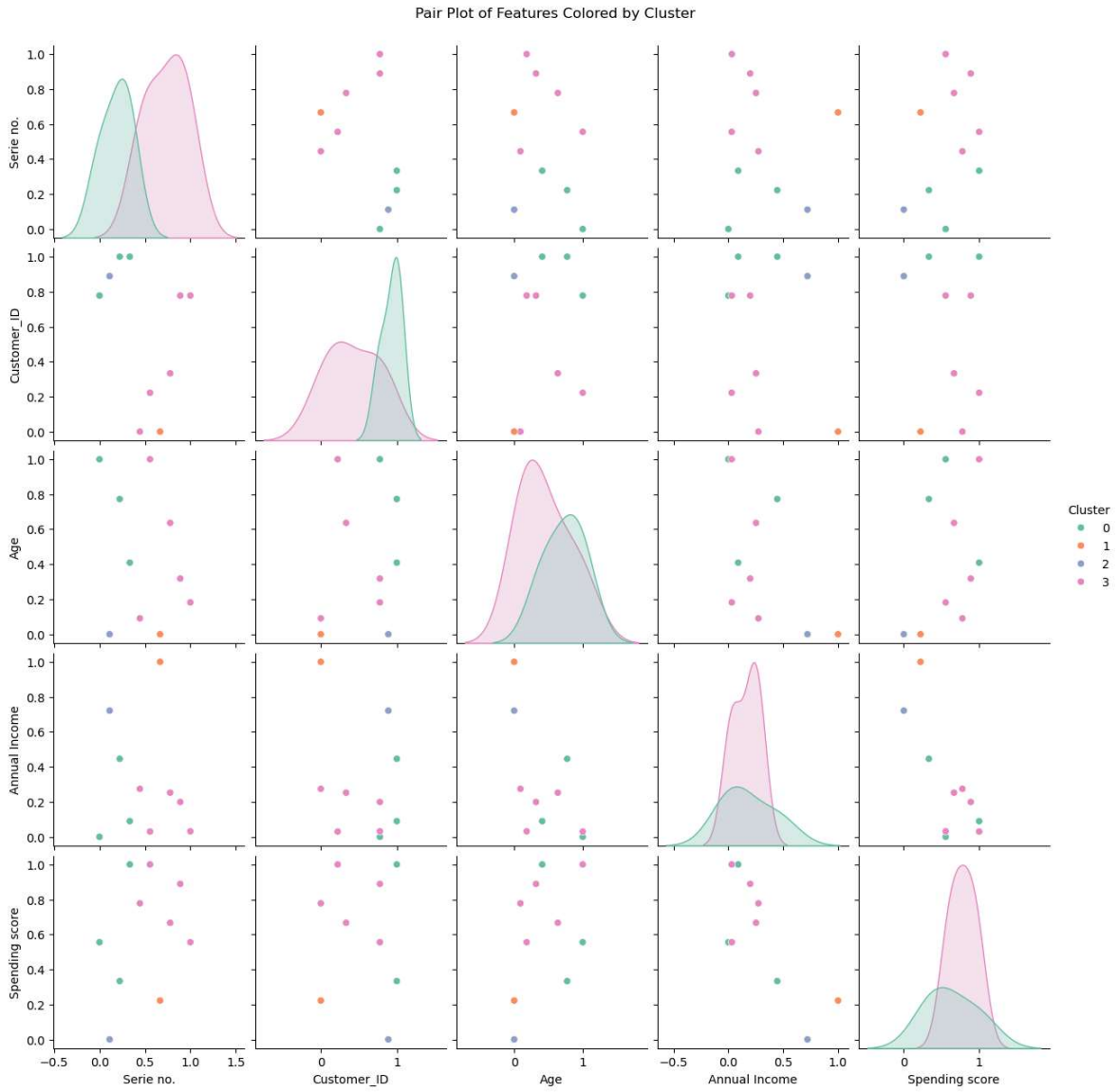
In [28]:
```
df.head()
df['Cluster'].value_counts()
```

Out[28]:
```
Cluster
3    5
0    3
2    1
1    1
Name: count, dtype: int64
```

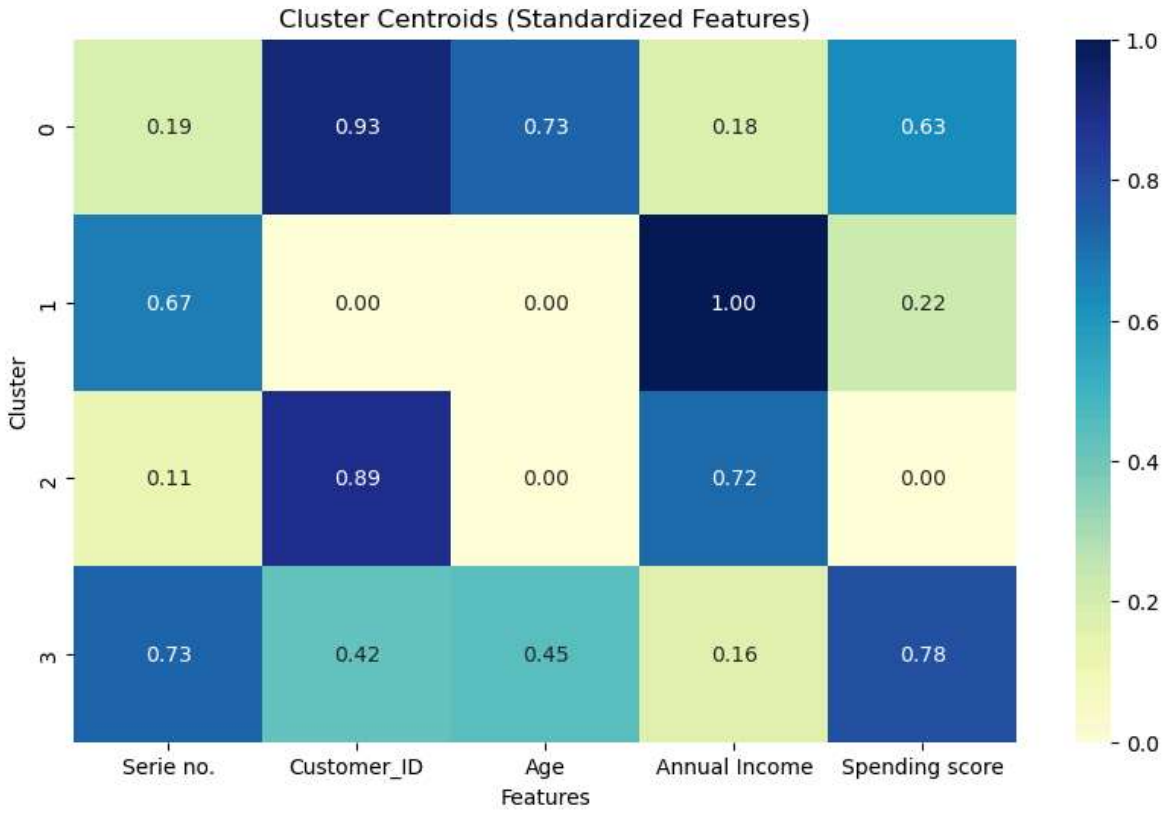
In [29]:
```
from sklearn.decomposition import PCA
import matplotlib.pyplot as plt

# Reduce to 2 principal components
pca = PCA(n_components=2)
pca_components = pca.fit_transform(scaled_df)

# Create a DataFrame with PCA results and cluster labels
pca_df = pd.DataFrame(data=pca_components, columns=['PCA1', 'PCA2'])
pca_df['Cluster'] = df['Cluster']

# Plot
plt.figure(figsize=(8, 6))
sns.scatterplot(x='PCA1', y='PCA2', hue='Cluster', data=pca_df, palette='Set1')
plt.title('Clusters Visualized in 2D (PCA)')
plt.grid(True)
plt.show()
```



In [30]:
```
# Add cluster label to scaled data for visualization
scaled_df_with_cluster = scaled_df.copy()
scaled_df_with_cluster['Cluster'] = df['Cluster']

# Use a subset if there are many features
sns.pairplot(scaled_df_with_cluster, hue='Cluster', palette='Set2')
plt.suptitle('Pair Plot of Features Colored by Cluster', y=1.02)
plt.show()
```



In [31]:
```
# Get centroids from KMeans
centroids = kmeans.cluster_centers_

# Create DataFrame for centroids
centroids_df = pd.DataFrame(centroids, columns=scaled_df.columns)

# Plot as heatmap
plt.figure(figsize=(10, 6))
sns.heatmap(centroids_df, annot=True, cmap='YlGnBu', fmt=".2f")
plt.title('Cluster Centroids (Standardized Features)')
plt.xlabel('Features')
plt.ylabel('Cluster')
plt.show()
```


Cluster Centroids (Standardized Features)
Cluster
0
1
2
3
0.19
0.93
0.73
0.18
0.63
0.67
0.00
0.00
1.00
0.22
0.11
0.89
0.00
0.72
0.00
0.73
0.42
0.45
0.16
0.78
Serie no.
Customer_ID
Age
Annual Income
Spending score
Features
1.0
0.8
0.6
0.4
0.2
0.0

In [ ]: