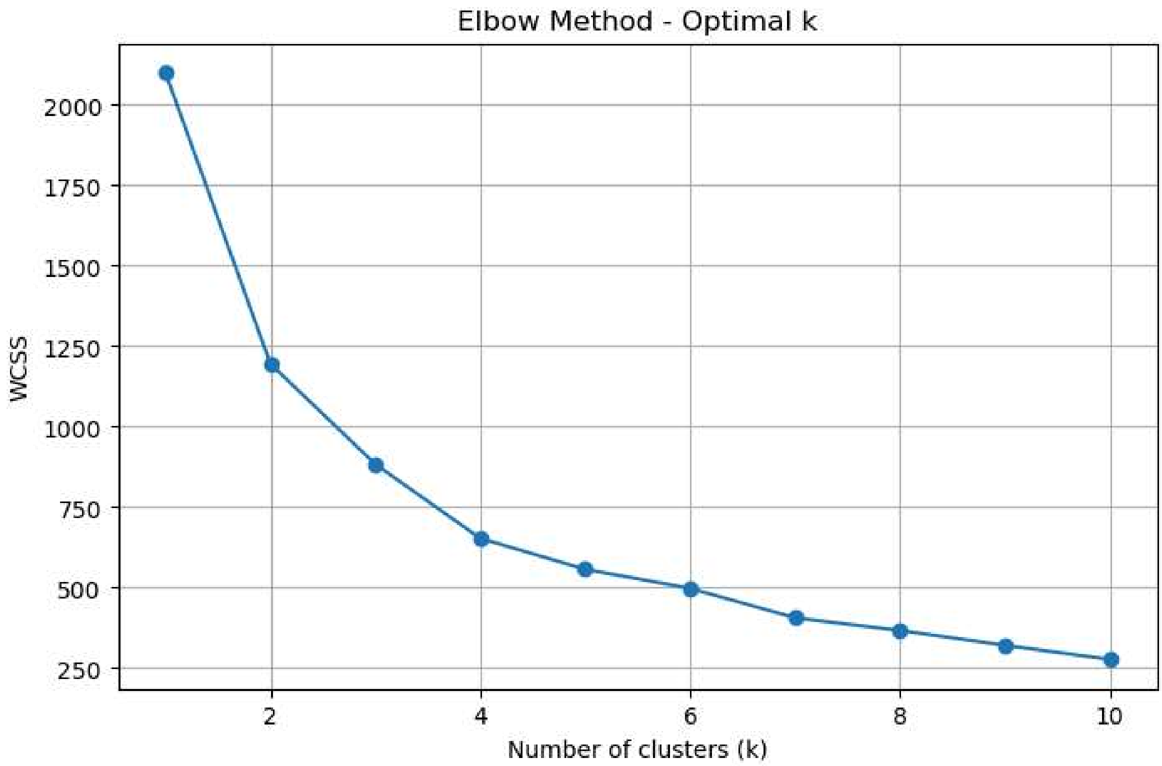
In [3]:
```
import pandas as pd
from sklearn.preprocessing import StandardScaler
from sklearn.cluster import KMeans

df = pd.read_csv("ecommerce.csv")

numeric_data = df.select_dtypes(include=['float64', 'int64'])

scaler = StandardScaler()
scaled_data = scaler.fit_transform(numeric_data)

kmeans = KMeans(n_clusters=3, random_state=42)
kmeans_labels = kmeans.fit_predict(scaled_data)

df['KMeans_Cluster'] = kmeans_labels
print(df.head())
```

```
   Customer ID  Gender  Age           City Membership Type  Total Spend  \
0          101  Female   29       New York            Gold      1120.20
1          102    Male   34    Los Angeles          Silver       780.50
2          103  Female   43        Chicago          Bronze       510.75
3          104    Male   30  San Francisco            Gold      1480.30
4          105    Male   27          Miami          Silver       720.40

   Items Purchased  Average Rating  Discount Applied  \
0               14             4.6              True
1               11             4.1             False
2                9             3.4              True
3               19             4.7             False
4               13             4.0              True

   Days Since Last Purchase Satisfaction Level  KMeans_Cluster
0                        25          Satisfied               1
1                        18            Neutral               0
2                        42        Unsatisfied               0
3                        12          Satisfied               1
4                        55        Unsatisfied               2
```

```
C:\ProgramData\anaconda3\Lib\site-packages\sklearn\cluster\_kmeans.py:1419: 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=2.
  warnings.warn(
```

In [4]:
```
import matplotlib.pyplot as plt
from sklearn.cluster import KMeans
wcss = []
for k in range(1, 11):
    kmeans = KMeans(n_clusters=k, random_state=42)
    kmeans.fit(scaled_data)
    wcss.append(kmeans.inertia_)

plt.figure(figsize=(8, 5))
plt.plot(range(1, 11), wcss, marker='o')
plt.title('Elbow Method - Optimal k')
plt.xlabel('Number of clusters (k)')
plt.ylabel('WCSS')
plt.grid(True)
plt.show()
```

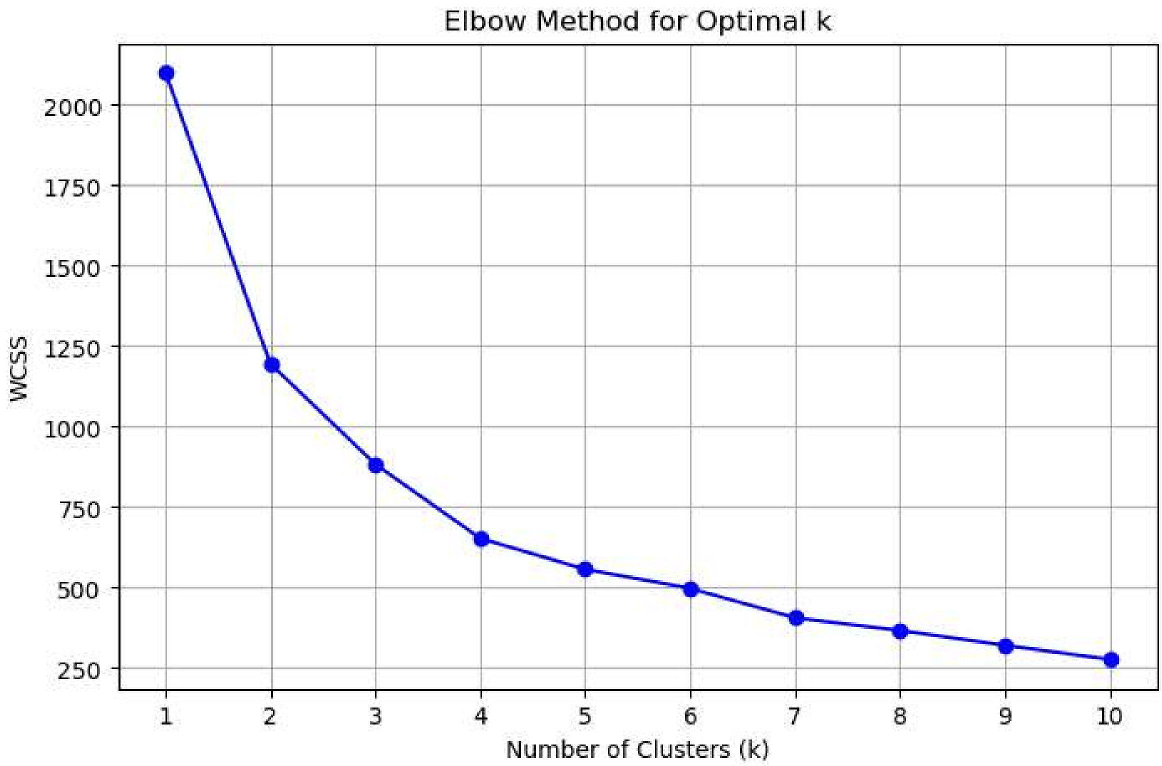
```
C:\ProgramData\anaconda3\Lib\site-packages\sklearn\cluster\_kmeans.py:1419: UserW
arning: KMeans is known to have a memory leak on Windows with MKL, when there are
less chunks than available threads. You can avoid it by setting the environment v
ariable OMP_NUM_THREADS=2.
  warnings.warn(
C:\ProgramData\anaconda3\Lib\site-packages\sklearn\cluster\_kmeans.py:1419: 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=2.
  warnings.warn(
C:\ProgramData\anaconda3\Lib\site-packages\sklearn\cluster\_kmeans.py:1419: 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=2.
  warnings.warn(
C:\ProgramData\anaconda3\Lib\site-packages\sklearn\cluster\_kmeans.py:1419: 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=2.
  warnings.warn(
C:\ProgramData\anaconda3\Lib\site-packages\sklearn\cluster\_kmeans.py:1419: 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=2.
  warnings.warn(
C:\ProgramData\anaconda3\Lib\site-packages\sklearn\cluster\_kmeans.py:1419: 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=2.
  warnings.warn(
C:\ProgramData\anaconda3\Lib\site-packages\sklearn\cluster\_kmeans.py:1419: 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=2.
  warnings.warn(
C:\ProgramData\anaconda3\Lib\site-packages\sklearn\cluster\_kmeans.py:1419: 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=2.
  warnings.warn(
C:\ProgramData\anaconda3\Lib\site-packages\sklearn\cluster\_kmeans.py:1419: 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=2.
  warnings.warn(
C:\ProgramData\anaconda3\Lib\site-packages\sklearn\cluster\_kmeans.py:1419: 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=2.
  warnings.warn(
```



In [5]:

```
import matplotlib.pyplot as plt
from sklearn.cluster import KMeans

wcss = []
K_range = range(1, 11)

for k in K_range:
    kmeans = KMeans(n_clusters=k, random_state=42)
    kmeans.fit(scaled_data)
    wcss.append(kmeans.inertia_)

plt.figure(figsize=(8, 5))
plt.plot(K_range, wcss, 'bo-')
plt.title('Elbow Method for Optimal k')
plt.xlabel('Number of Clusters (k)')
plt.ylabel('WCSS')
plt.xticks(K_range)
plt.grid(True)
plt.show()
```

```
C:\ProgramData\anaconda3\Lib\site-packages\sklearn\cluster\_kmeans.py:1419: 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=2.
  warnings.warn(
C:\ProgramData\anaconda3\Lib\site-packages\sklearn\cluster\_kmeans.py:1419: 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=2.
  warnings.warn(
C:\ProgramData\anaconda3\Lib\site-packages\sklearn\cluster\_kmeans.py:1419: 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=2.
  warnings.warn(
C:\ProgramData\anaconda3\Lib\site-packages\sklearn\cluster\_kmeans.py:1419: 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=2.
  warnings.warn(
C:\ProgramData\anaconda3\Lib\site-packages\sklearn\cluster\_kmeans.py:1419: 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=2.
  warnings.warn(
C:\ProgramData\anaconda3\Lib\site-packages\sklearn\cluster\_kmeans.py:1419: 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=2.
  warnings.warn(
C:\ProgramData\anaconda3\Lib\site-packages\sklearn\cluster\_kmeans.py:1419: 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=2.
  warnings.warn(
C:\ProgramData\anaconda3\Lib\site-packages\sklearn\cluster\_kmeans.py:1419: 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=2.
  warnings.warn(
C:\ProgramData\anaconda3\Lib\site-packages\sklearn\cluster\_kmeans.py:1419: 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=2.
  warnings.warn(
C:\ProgramData\anaconda3\Lib\site-packages\sklearn\cluster\_kmeans.py:1419: 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=2.
  warnings.warn(
```



In [6]:
```
from sklearn.cluster import AgglomerativeClustering

agglo = AgglomerativeClustering(n_clusters=3, linkage='ward')
agglo_labels = agglo.fit_predict(scaled_data)

df['Agglo_Cluster'] = agglo_labels

print(df.head())
```

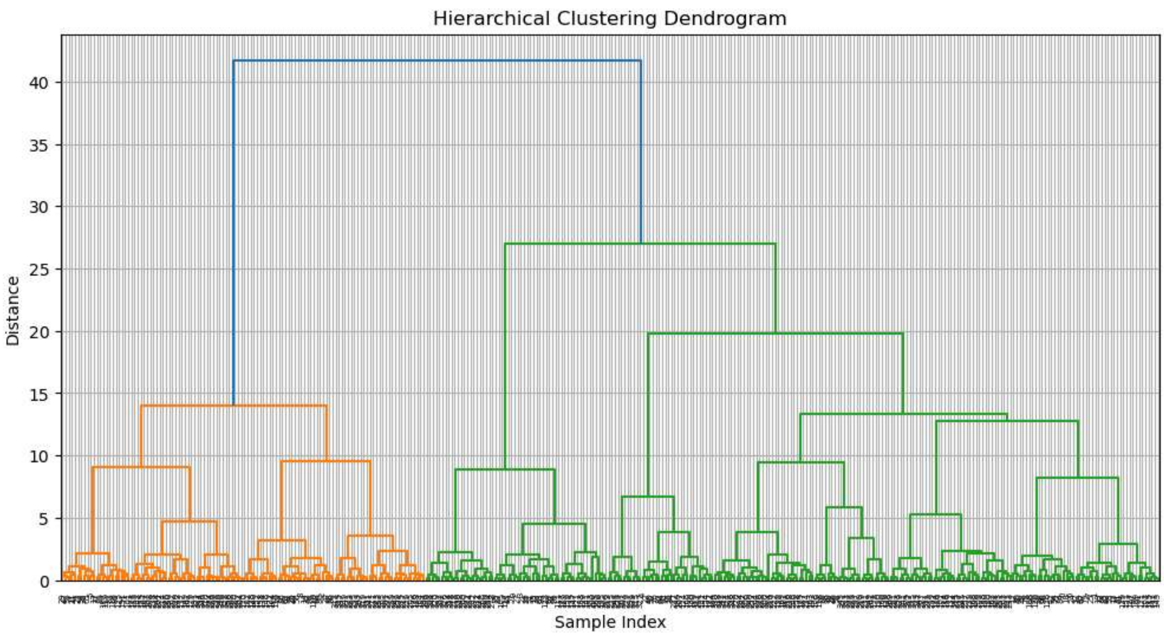
```
   Customer ID  Gender  Age           City Membership Type  Total Spend  \
0          101  Female   29       New York            Gold      1120.20
1          102    Male   34    Los Angeles          Silver       780.50
2          103  Female   43        Chicago          Bronze       510.75
3          104    Male   30  San Francisco            Gold      1480.30
4          105    Male   27          Miami          Silver       720.40

   Items Purchased  Average Rating  Discount Applied  \
0               14             4.6              True
1               11             4.1             False
2                9             3.4              True
3               19             4.7             False
4               13             4.0              True

   Days Since Last Purchase Satisfaction Level  KMeans_Cluster  Agglo_Cluster
0                        25          Satisfied               1              0
1                        18            Neutral               0              0
2                        42        Unsatisfied               0              1
3                        12          Satisfied               1              2
4                        55        Unsatisfied               2              0
```

In [10]:
```
from sklearn.cluster import AgglomerativeClustering

agglo = AgglomerativeClustering(n_clusters=3, metric='euclidean', linkage='ward'
agglo_labels = agglo.fit_predict(scaled_data)

df['Agglo_Cluster'] = agglo_labels
```

```
print(df.head())
```

```
   Customer ID  Gender  Age           City Membership Type  Total Spend  \
0          101  Female   29       New York            Gold      1120.20   
1          102    Male   34    Los Angeles          Silver       780.50   
2          103  Female   43        Chicago          Bronze       510.75   
3          104    Male   30  San Francisco            Gold      1480.30   
4          105    Male   27          Miami          Silver       720.40   

   Items Purchased  Average Rating  Discount Applied  \
0               14             4.6              True   
1               11             4.1             False   
2                9             3.4              True   
3               19             4.7             False   
4               13             4.0              True   

   Days Since Last Purchase Satisfaction Level  KMeans_Cluster  Agglo_Cluster  
0                        25          Satisfied               1              0  
1                        18            Neutral               0              0  
2                        42        Unsatisfied               0              1  
3                        12          Satisfied               1              2  
4                        55        Unsatisfied               2              0  
```

In [11]:
```
import matplotlib.pyplot as plt
import scipy.cluster.hierarchy as sch

linkage_matrix = sch.linkage(scaled_data, method='ward')

plt.figure(figsize=(12, 6))
sch.dendrogram(linkage_matrix)
plt.title("Hierarchical Clustering Dendrogram")
plt.xlabel("Sample Index")
plt.ylabel("Distance")
plt.grid(True)
plt.show()
```



In [12]:
```
print("K-Means Cluster Counts:")
print(df['KMeans_Cluster'].value_counts())
```

```
print("\nAgglomerative Cluster Counts:")
print(df['Agglo_Cluster'].value_counts())
```

```
K-Means Cluster Counts:
KMeans_Cluster
0    190
1    126
2     34
Name: count, dtype: int64

Agglomerative Cluster Counts:
Agglo_Cluster
0    176
1    116
2     58
Name: count, dtype: int64
```

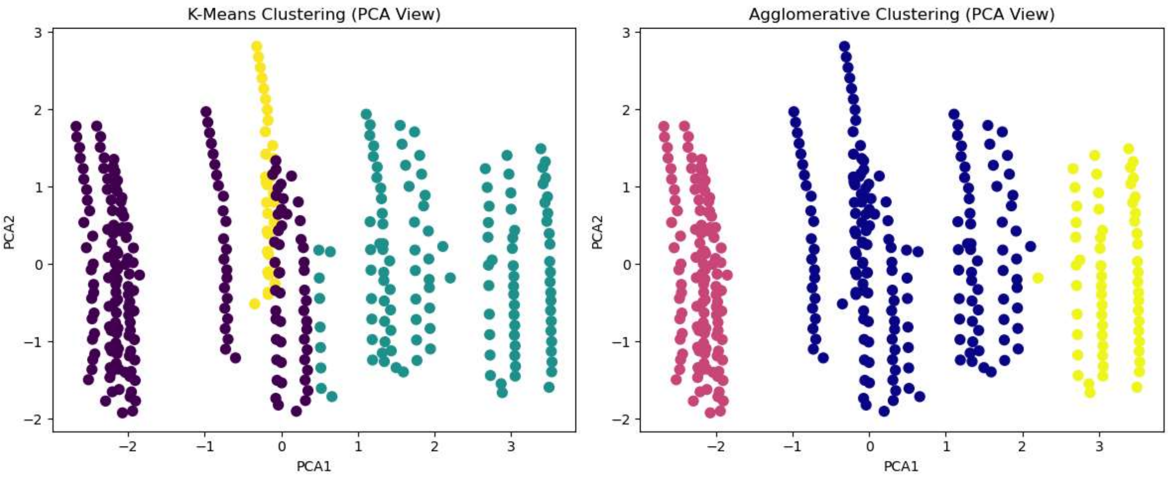
In [13]:

```
comparison = pd.crosstab(df['KMeans_Cluster'], df['Agglo_Cluster'])
print("\nCross-tabulation of KMeans vs Agglomerative Clusters:")
print(comparison)
```

```
Cross-tabulation of KMeans vs Agglomerative Clusters:
Agglo_Cluster    0    1   2
KMeans_Cluster
0               74  116   0
1               68    0  58
2               34    0   0
```

In [14]:

```
from sklearn.decomposition import PCA
import matplotlib.pyplot as plt

pca = PCA(n_components=2)
pca_data = pca.fit_transform(scaled_data)

df['PCA1'] = pca_data[:, 0]
df['PCA2'] = pca_data[:, 1]

plt.figure(figsize=(12, 5))

plt.subplot(1, 2, 1)
plt.scatter(df['PCA1'], df['PCA2'], c=df['KMeans_Cluster'], cmap='viridis', s=50
plt.title("K-Means Clustering (PCA View)")
plt.xlabel("PCA1")
plt.ylabel("PCA2")

plt.subplot(1, 2, 2)
plt.scatter(df['PCA1'], df['PCA2'], c=df['Agglo_Cluster'], cmap='plasma', s=50)
plt.title("Agglomerative Clustering (PCA View)")
plt.xlabel("PCA1")
plt.ylabel("PCA2")

plt.tight_layout()
plt.show()
```

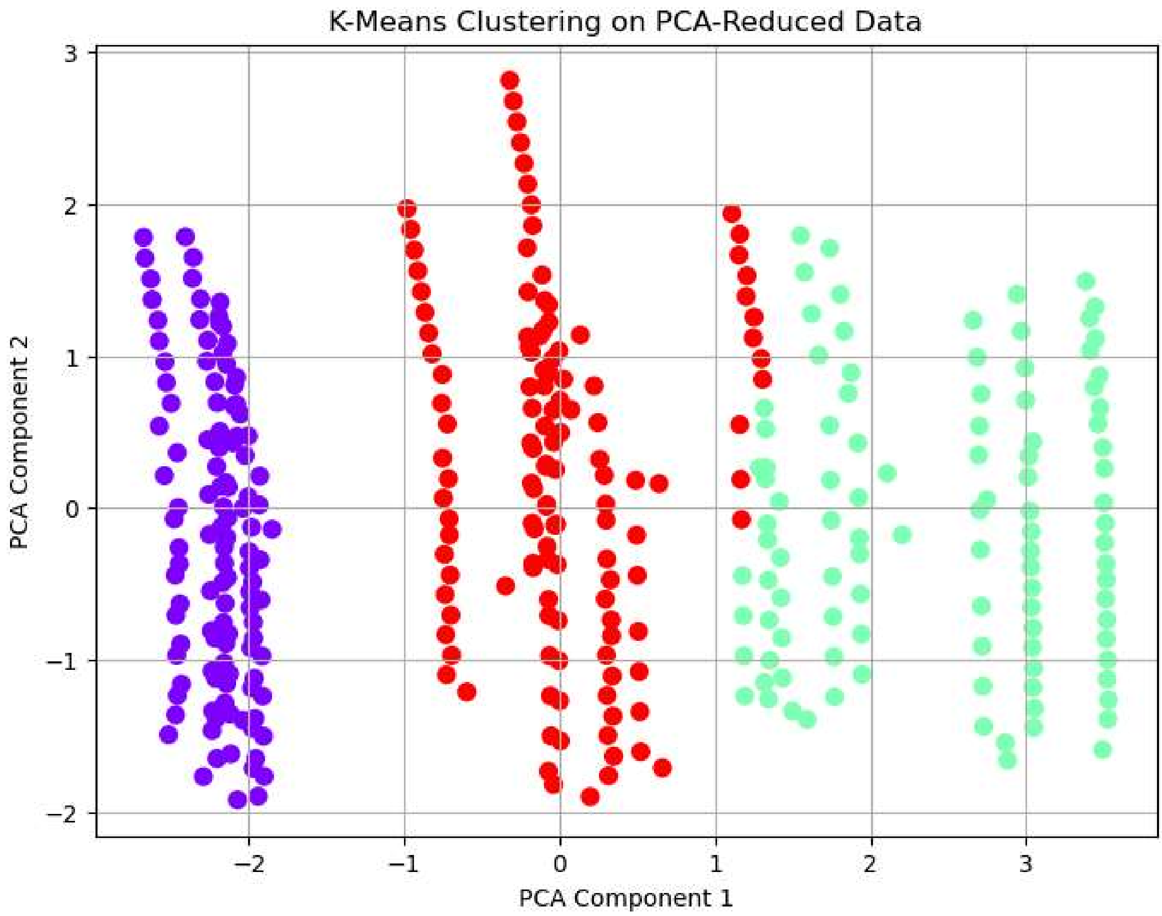


In [15]:
```
from sklearn.decomposition import PCA

pca = PCA(n_components=2)
pca_data = pca.fit_transform(scaled_data)

print("Explained variance ratio:", pca.explained_variance_ratio_)
```

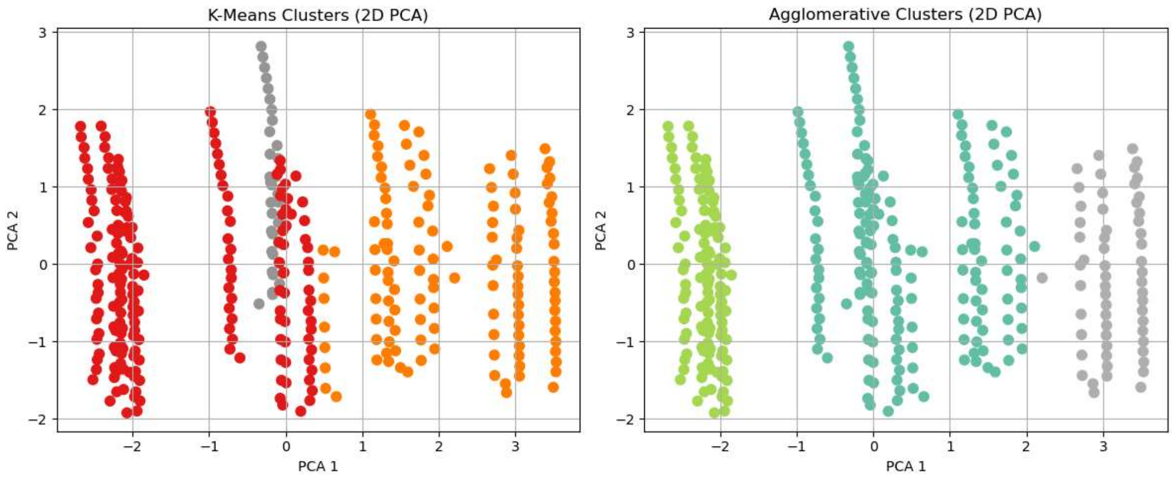
```
Explained variance ratio: [0.61865549 0.17541405]
```

In [17]:
```
kmeans_pca = KMeans(n_clusters=3, random_state=42)
kmeans_pca_labels = kmeans_pca.fit_predict(pca_data)

df['KMeans_PCA_Cluster'] = kmeans_pca_labels
import matplotlib.pyplot as plt

plt.figure(figsize=(8, 6))
plt.scatter(pca_data[:, 0], pca_data[:, 1], c=kmeans_pca_labels, cmap='rainbow',
plt.title("K-Means Clustering on PCA-Reduced Data")
plt.xlabel("PCA Component 1")
plt.ylabel("PCA Component 2")
plt.grid(True)
plt.show()
```

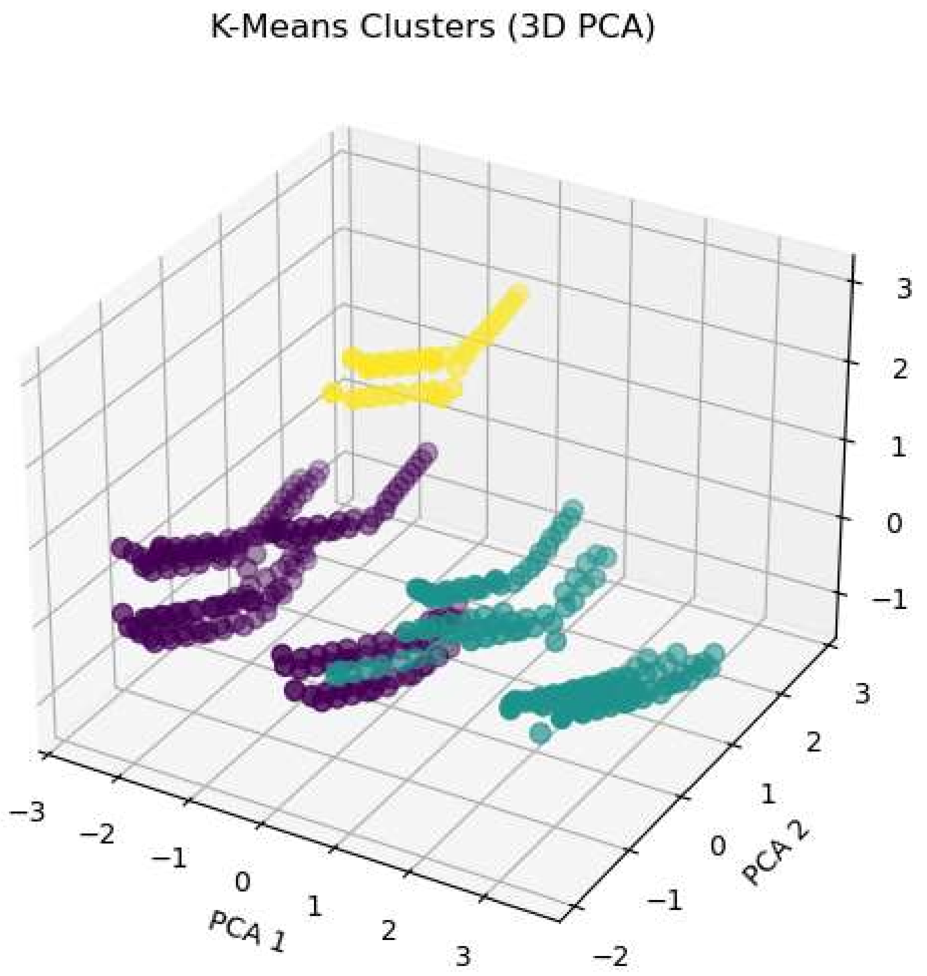
```
C:\ProgramData\anaconda3\Lib\site-packages\sklearn\cluster\_kmeans.py:1419: UserW
arning: KMeans is known to have a memory leak on Windows with MKL, when there are
less chunks than available threads. You can avoid it by setting the environment v
ariable OMP_NUM_THREADS=2.
  warnings.warn(
```



In [18]:
```
import matplotlib.pyplot as plt
from sklearn.decomposition import PCA

pca_2d = PCA(n_components=2)
pca_data_2d = pca_2d.fit_transform(scaled_data)

df['PCA1'] = pca_data_2d[:, 0]
df['PCA2'] = pca_data_2d[:, 1]

plt.figure(figsize=(12, 5))

plt.subplot(1, 2, 1)
plt.scatter(df['PCA1'], df['PCA2'], c=df['KMeans_Cluster'], cmap='Set1', s=50)
plt.title("K-Means Clusters (2D PCA)")
plt.xlabel("PCA 1")
plt.ylabel("PCA 2")
plt.grid(True)

plt.subplot(1, 2, 2)
plt.scatter(df['PCA1'], df['PCA2'], c=df['Agglo_Cluster'], cmap='Set2', s=50)
plt.title("Agglomerative Clusters (2D PCA)")
plt.xlabel("PCA 1")
plt.ylabel("PCA 2")
plt.grid(True)

plt.tight_layout()
plt.show()
```



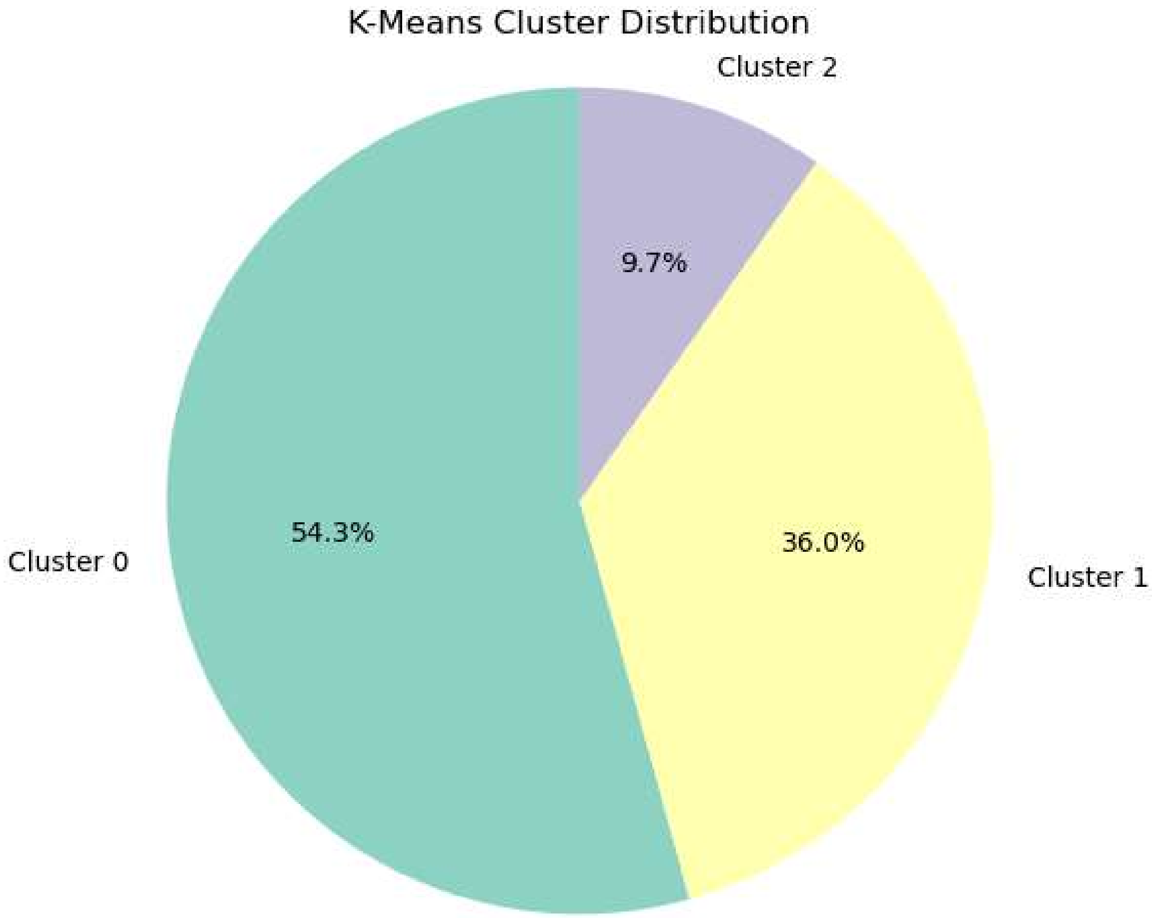
In [19]:
```
from mpl_toolkits.mplot3d import Axes3D

# Step 1: Apply PCA (3 components)
pca_3d = PCA(n_components=3)
pca_data_3d = pca_3d.fit_transform(scaled_data)

# Step 2: 3D Scatter Plot for K-Means
fig = plt.figure(figsize=(10, 6))
ax = fig.add_subplot(111, projection='3d')

scatter = ax.scatter(
    pca_data_3d[:, 0], pca_data_3d[:, 1], pca_data_3d[:, 2],
    c=df['KMeans_Cluster'], cmap='viridis', s=50
)

ax.set_title("K-Means Clusters (3D PCA)")
ax.set_xlabel("PCA 1")
ax.set_ylabel("PCA 2")
ax.set_zlabel("PCA 3")
plt.show()
```



In [20]:
```
from sklearn.metrics import silhouette_score

score_kmeans = silhouette_score(scaled_data, df['KMeans_Cluster'])
print(f"Silhouette Score (K-Means): {score_kmeans:.3f}")

score_agglo = silhouette_score(scaled_data, df['Agglo_Cluster'])
print(f"Silhouette Score (Agglomerative): {score_agglo:.3f}")
```

```
Silhouette Score (K-Means): 0.392
Silhouette Score (Agglomerative): 0.336
```

In [21]:
```
import matplotlib.pyplot as plt

cluster_counts = df['KMeans_Cluster'].value_counts().sort_index()
labels = [f'Cluster {i}' for i in cluster_counts.index]

plt.figure(figsize=(6, 6))
plt.pie(cluster_counts, labels=labels, autopct='%1.1f%%', startangle=90, colors=
plt.title("K-Means Cluster Distribution")
plt.axis('equal')
plt.show()
```



In [26]:
```
kmeans_profile = df.groupby('KMeans_Cluster').mean(numeric_only=True)

print("KMeans Cluster Profile:")
print(kmeans_profile)
```

```
KMeans Cluster Profile:
                Customer ID        Age  Total Spend  Items Purchased  \
KMeans_Cluster
0                279.926316  37.136842   584.933684         9.473684
1                270.992063  30.095238  1276.355556        17.269841
2                267.470588  26.794118   703.688235        12.764706

                Average Rating  Discount Applied  Days Since Last Purchase  \
KMeans_Cluster
0                     3.605789          0.431579                 27.736842
1                     4.642857          0.468254                 17.682540
2                     4.017647          1.000000                 53.176471

                Agglo_Cluster      PCA1      PCA2  KMeans_PCA_Cluster
KMeans_Cluster
0                    0.610526 -1.416555 -0.141727            0.778947
1                    0.920635  2.183864 -0.052756            1.166667
2                    0.000000 -0.177100  0.987512            2.000000
```

In [31]:
```
kmeans_profile = df.groupby('KMeans_Cluster').mean(numeric_only=True)
print(kmeans_profile.round(2))
```

```
                Customer ID    Age  Total Spend  Items Purchased  \
KMeans_Cluster
0                    279.93  37.14       584.93             9.47
1                    270.99  30.10      1276.36            17.27
2                    267.47  26.79       703.69            12.76

                Average Rating  Discount Applied  Days Since Last Purchase  \
KMeans_Cluster
0                         3.61              0.43                     27.74
1                         4.64              0.47                     17.68
2                         4.02              1.00                     53.18

                Agglo_Cluster  PCA1  PCA2  KMeans_PCA_Cluster
KMeans_Cluster
0                        0.61 -1.42 -0.14                0.78
1                        0.92  2.18 -0.05                1.17
2                        0.00 -0.18  0.99                2.00
```

In [32]:

```
cluster_labels = {
    0: 'High Spenders',
    1: 'Lost',
    2: 'Loyal'
}

df['Customer_Segment'] = df['KMeans_Cluster'].map(cluster_labels)

df[['KMeans_Cluster', 'Customer_Segment']].head()
```

Out[32]:

	KMeans_Cluster	Customer_Segment
0	1	Lost
1	0	High Spenders
2	0	High Spenders
3	1	Lost
4	2	Loyal

In [33]:

```
print(df['Customer_Segment'].value_counts())
```

```
Customer_Segment
High Spenders    190
Lost             126
Loyal             34
Name: count, dtype: int64
```

In [ ]: