Clustering

In [7]:
```
# Step 5.1: Preparing the data for the Lookalike Model

# Merging customer and transaction data to get product purchase history per
customer_transactions = pd.merge(transactions, customers, on='CustomerID', h
customer_transactions = pd.merge(customer_transactions, products, on='Produc

# Aggregating purchase information by customer
customer_profile = customer_transactions.groupby(['CustomerID', 'ProductID']
    TotalAmountSpent=('TotalValue', 'sum'),
    TotalQuantity=('Quantity', 'sum')
).reset_index()

# Pivot table to get a matrix of customers vs products
customer_pivot = customer_profile.pivot_table(index='CustomerID', columns='F

# Display the customer-product matrix
customer_pivot.head()
```

Out[7]:

ProductID	P001	P002	P003	P004	P005	P006	P007	P008	P009	P0
CustomerID										
C0001	0.0	0.0	0.0	0.00	0.0	0.00	0.0	0.0	0.0	(
C0002	0.0	0.0	0.0	382.76	0.0	0.00	0.0	0.0	0.0	(
C0003	0.0	1385.2	0.0	0.00	0.0	363.96	0.0	0.0	0.0	(
C0004	0.0	0.0	0.0	0.00	0.0	0.00	0.0	293.7	0.0	(
C0005	0.0	0.0	0.0	0.00	0.0	0.00	0.0	0.0	0.0	(

5 rows × 100 columns

In [8]:
```
# Step 5.2: Compute cosine similarity between customers based on their produ

# Normalize the data to scale the features
scaler = StandardScaler()
customer_scaled = scaler.fit_transform(customer_pivot)

# Compute cosine similarity
cosine_sim = cosine_similarity(customer_scaled)

# Convert cosine similarity to DataFrame for easier handling
cosine_sim_df = pd.DataFrame(cosine_sim, index=customer_pivot.index, columns

# Display similarity matrix
cosine_sim_df.head()
```

Out[8]:

CustomerID	C0001	C0002	C0003	C0004	C0005	C0006	
CustomerID							
C0001	1.000000	-0.048829	-0.061476	-0.079060	-0.051689	-0.064034	0
C0002	-0.048829	1.000000	-0.035699	-0.051683	-0.023066	-0.033697	-0
C0003	-0.061476	-0.035699	1.000000	0.040222	0.244296	-0.046598	-0
C0004	-0.079060	-0.051683	0.040222	1.000000	0.079853	-0.065466	-0
C0005	-0.051689	-0.023066	0.244296	0.079853	1.000000	-0.032509	-0

5 rows × 199 columns

In [13]:
```
# Step 5.3: Generate recommendations for customers C0001 to C0020

lookalike_recommendations = {}

for customer_id in range(1, 21):
    customer_similarities = cosine_sim_df.loc[f'C{customer_id:04d}']
    top_3_similar_customers = customer_similarities.sort_values(ascending=Fa
    lookalike_recommendations[f'C{customer_id:04d}'] = top_3_similar_custome

# Convert to DataFrame
lookalike_df = pd.DataFrame.from_dict(lookalike_recommendations, orient='inc

# Save lookalike recommendations to a CSV file
lookalike_df.to_csv('Anusha_Khot_Lookalike.csv', index=False)

# Display the recommendations
lookalike_df.head()
```

Out[13]:

	C0194	C0104	C0020	C0030	C0091	C0071	C0181	C01
C0001	0.404928	0.374002	0.366609	NaN	NaN	NaN	NaN	N
C0020	NaN	0.472465	NaN	NaN	NaN	NaN	NaN	N
C0007	NaN	NaN	0.456615	NaN	NaN	NaN	NaN	N
C0002	NaN	NaN	NaN	0.404617	0.383778	0.320158	NaN	N
C0008	NaN	NaN	NaN	NaN	0.260560	NaN	NaN	N

5 rows × 48 columns

In [10]:
```
# Step 6.1: Preparing data for clustering

# Aggregating customer transactions by product category
category_sales = customer_transactions.groupby(['CustomerID', 'Category']).a
    TotalAmountSpent=('TotalValue', 'sum')
).reset_index()

# Pivoting to get customer vs category matrix
```

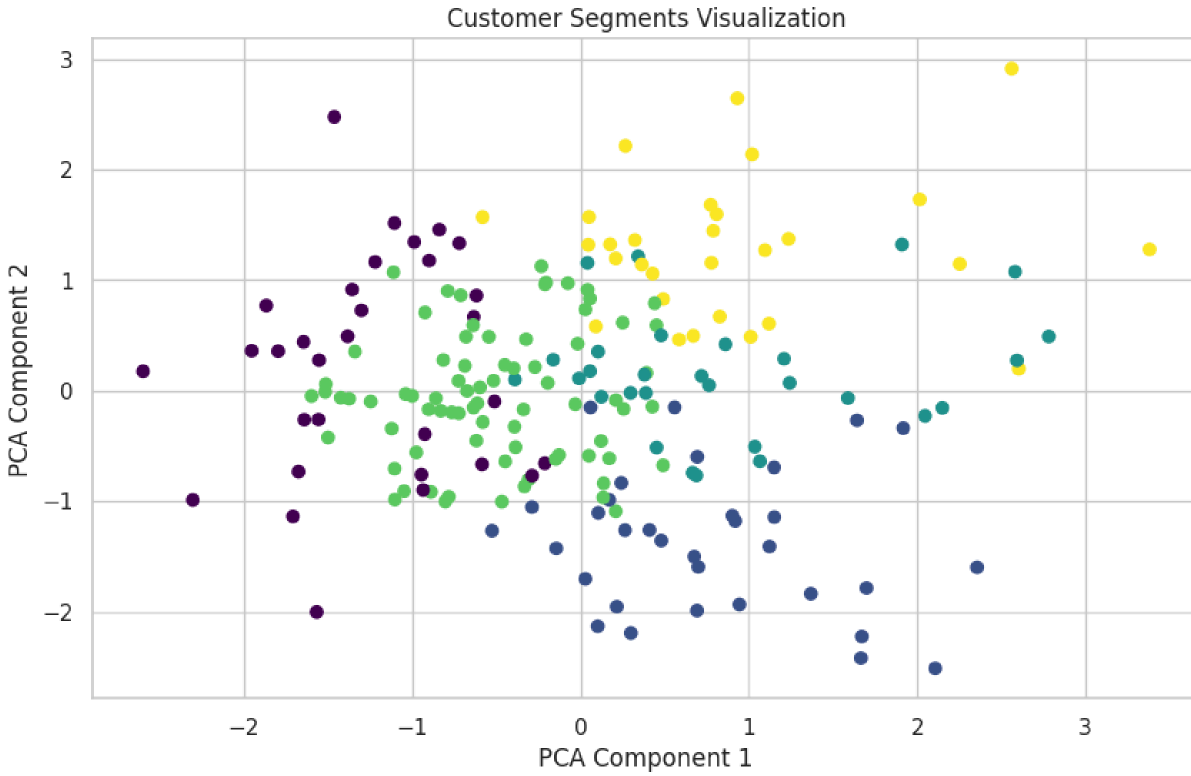
```
category_pivot = category_sales.pivot_table(index='CustomerID', columns='Cat

# Scaling data for clustering
category_scaled = scaler.fit_transform(category_pivot)

# Display the data
category_pivot.head()
```

Out[10]:

Category	Books	Clothing	Electronics	Home Decor
CustomerID				
C0001	114.60	0.00	2827.30	412.62
C0002	0.00	1025.46	0.00	837.28
C0003	0.00	122.36	1385.20	1217.82
C0004	1888.48	0.00	1355.74	2110.66
C0005	0.00	0.00	1180.38	853.86

In [14]:

```
# Step 6.2: Applying KMeans clustering

# Let's test with 5 clusters
kmeans = KMeans(n_clusters=5, random_state=42)
kmeans.fit(category_scaled)

# Add cluster labels to the original data
category_pivot['Cluster'] = kmeans.labels_

# Visualizing the clusters using a scatter plot (2D PCA)
from sklearn.decomposition import PCA

pca = PCA(n_components=2)
pca_components = pca.fit_transform(category_scaled)

plt.figure(figsize=(10,6))
plt.scatter(pca_components[:, 0], pca_components[:, 1], c=category_pivot['Cl
plt.title('Customer Segments Visualization')
plt.xlabel('PCA Component 1')
plt.ylabel('PCA Component 2')
plt.show()

# Calculate DB Index
db_index = davies_bouldin_score(category_scaled, kmeans.labels_)
print("DB Index: ", db_index)

# Save clustering results to a CSV file
category_pivot.to_csv('Anusha_Khot_Customer_Segmentation.csv', index=False)

# Display final results
category_pivot.head()
```




```
DB Index:  1.108179281522266
```

Out[14]:

Category	Books	Clothing	Electronics	Home Decor	Cluster
CustomerID					
C0001	114.60	0.00	2827.30	412.62	1
C0002	0.00	1025.46	0.00	837.28	3
C0003	0.00	122.36	1385.20	1217.82	3
C0004	1888.48	0.00	1355.74	2110.66	0
C0005	0.00	0.00	1180.38	853.86	3

In [12]:

```
# Step 7: Save all final outputs as required

# Save clustering results to a CSV file
category_pivot.to_csv('Customer_Segmentation.csv')

# Display final results
category_pivot.head()
```

Out[12]:

Category	Books	Clothing	Electronics	Home Decor	Cluster
CustomerID					
C0001	114.60	0.00	2827.30	412.62	1
C0002	0.00	1025.46	0.00	837.28	3
C0003	0.00	122.36	1385.20	1217.82	3
C0004	1888.48	0.00	1355.74	2110.66	0
C0005	0.00	0.00	1180.38	853.86	3