Section 1 – Describe your dataset

In [1]:
```
# Section 1
# 1.1 Describe your dataset
# Dataset: Customer purchase behavior - Electronic Sales Data
# Description:
"""
The dataset contains sales transaction information for an electronics company during a one-year period from Sep
It contains detailed data about client demographics, product categories, and purchasing habits. Here, all custo
My dataset contains 20000 rows and 16 columns.
"""
# Link: https://www.kaggle.com/datasets/cameronseamons/electronic-sales-sep2023-sep2024

# 1.2. Explain the features:
# Customer ID: A unique identification for each customer.
# Age: The age of the consumer in numerical figures.
# Gender: Is the consumer male or female?
# Loyalty Members: This is a yes or no value, and the values fluctuate over time based on who cancelled and who
# Product Type: The type of electronic products sold, such as a smartphone, laptop, or tablet.
# SKU: This is a unique code for each product.
# Rating: Customers rate the product on a scale of 1 to 5 stars. It should also have no Null Ratings.
# Order Status: Status of the order is Completed or Cancelled.
# Payment Method: Payment methods include cash, credit card, and Paypal.
# Total Price: The total price of the transaction is a numerical amount.
# Unit Price: The price per unit of the product is a numerical figure.
# Quantity: The number of units purchased is a numerical figure.
# Purchase Date: Date of the product purchase (format: YYYY-MM-DD)
# Shipping Type: Type of shipping chosen are standard, overnight, express.
# Add-ons Purchased: List of any additional items purchased (Accessories, Extended Warranty)
# Add-on Total: Total price of add-ons purchased is numeric value

print()
```

Section 2 - Load your dataset

In [2]:
```
# 2.1 Load the dataset into a pandas DataFrame
# Libraries
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
from scipy import stats
from sklearn.cluster import KMeans
from sklearn.preprocessing import StandardScaler
from sklearn.discriminant_analysis import LinearDiscriminantAnalysis as LDA
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score, silhouette_score, davies_bouldin_score
import seaborn as sns

# Loading the dataset before cleaning
df = pd.read_csv('Electronic_sales_Sep2023-Sep2024.csv')
df.shape
```

Out[2]: (20000, 16)

In [3]:
```
# 2.2 Display some rows in the dataset
df.head()
```

Out[3]:

	Customer ID	Age	Gender	Loyalty Member	Product Type	SKU	Rating	Order Status	Payment Method	Total Price	Unit Price	Quantity	Purchase Date	Shippi Ty
0	1000	53	Male	No	Smartphone	SKU1004	2	Cancelled	Credit Card	5538.33	791.19	7	2024-03-20	Standa
1	1000	53	Male	No	Tablet	SKU1002	3	Completed	Paypal	741.09	247.03	3	2024-04-20	Overnig
2	1002	41	Male	No	Laptop	SKU1005	3	Completed	Credit Card	1855.84	463.96	4	2023-10-17	Expre
3	1002	41	Male	Yes	Smartphone	SKU1004	2	Completed	Cash	3164.76	791.19	4	2024-08-09	Overnig
4	1003	75	Male	Yes	Smartphone	SKU1001	5	Completed	Cash	41.50	20.75	2	2024-05-21	Expre

In [4]:
```
# 2.3. Show the loaded features
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 20000 entries, 0 to 19999
Data columns (total 16 columns):
 #   Column             Non-Null Count  Dtype
---  ------             --------------  -----
 0   Customer ID        20000 non-null  int64
 1   Age                20000 non-null  int64
 2   Gender             19999 non-null  object
 3   Loyalty Member     20000 non-null  object
 4   Product Type       20000 non-null  object
 5   SKU                20000 non-null  object
 6   Rating             20000 non-null  int64
 7   Order Status       20000 non-null  object
 8   Payment Method     20000 non-null  object
 9   Total Price        20000 non-null  float64
 10  Unit Price         20000 non-null  float64
 11  Quantity           20000 non-null  int64
 12  Purchase Date      20000 non-null  object
 13  Shipping Type      20000 non-null  object
 14  Add-ons Purchased  15132 non-null  object
 15  Add-on Total       20000 non-null  float64
dtypes: float64(3), int64(4), object(9)
memory usage: 2.4+ MB
```

In [5]:
```
# 2.4. Show some trend statistics
df.describe()
```

Out[5]:

	Customer ID	Age	Rating	Total Price	Unit Price	Quantity	Add-on Total
count	20000.000000	20000.000000	20000.000000	20000.000000	20000.000000	20000.000000	20000.000000
mean	10483.526550	48.994100	3.093950	3180.133418	578.631867	5.485550	62.244848
std	5631.732525	18.038745	1.223764	2544.978675	312.274076	2.870854	58.058431
min	1000.000000	18.000000	1.000000	20.750000	20.750000	1.000000	0.000000
25%	5478.000000	33.000000	2.000000	1139.680000	361.180000	3.000000	7.615000
50%	10499.500000	49.000000	3.000000	2534.490000	463.960000	5.000000	51.700000
75%	15504.000000	65.000000	4.000000	4639.600000	791.190000	8.000000	93.842500
max	19998.000000	80.000000	5.000000	11396.800000	1139.680000	10.000000	292.770000

Data Pre-processing

In [6]:
```
# Looking for missing data
df.isnull().sum()
```

Out[6]:
```
Customer ID             0
Age                     0
Gender                  1
Loyalty Member          0
Product Type            0
SKU                     0
Rating                  0
Order Status            0
Payment Method          0
Total Price             0
Unit Price              0
Quantity                0
Purchase Date           0
Shipping Type           0
Add-ons Purchased    4868
Add-on Total            0
dtype: int64
```

There is not need to drop null values.

In [7]:
```
# Data Preprocessing
df['Purchase Date'] = pd.to_datetime(df['Purchase Date'])

# Remove duplicates
df.drop_duplicates(inplace=True)
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 20000 entries, 0 to 19999
Data columns (total 16 columns):
 #   Column             Non-Null Count  Dtype
---  ------             --------------  -----
 0   Customer ID        20000 non-null  int64
 1   Age                20000 non-null  int64
 2   Gender             19999 non-null  object
 3   Loyalty Member     20000 non-null  object
 4   Product Type       20000 non-null  object
 5   SKU                20000 non-null  object
 6   Rating             20000 non-null  int64
 7   Order Status       20000 non-null  object
 8   Payment Method     20000 non-null  object
 9   Total Price        20000 non-null  float64
 10  Unit Price         20000 non-null  float64
 11  Quantity           20000 non-null  int64
 12  Purchase Date      20000 non-null  datetime64[ns]
 13  Shipping Type      20000 non-null  object
 14  Add-ons Purchased  15132 non-null  object
 15  Add-on Total       20000 non-null  float64
dtypes: datetime64[ns](1), float64(3), int64(4), object(8)
memory usage: 2.6+ MB
```

In [8]:
```
# Saving the cleaned data to use it for further use
df.to_csv('Electronic_sales_cleaned.csv', index=False)
```

In [9]:
```
# Loading the clean dataset
df = pd.read_csv('Electronic_sales_cleaned.csv')
df.shape
```

Out[9]: (20000, 16)

In [10]:
```
# Additional insights like median and variance
median = df.median()
variance = df.var()
mean = df.mean()
```

Section 3 – Correlation Matrix

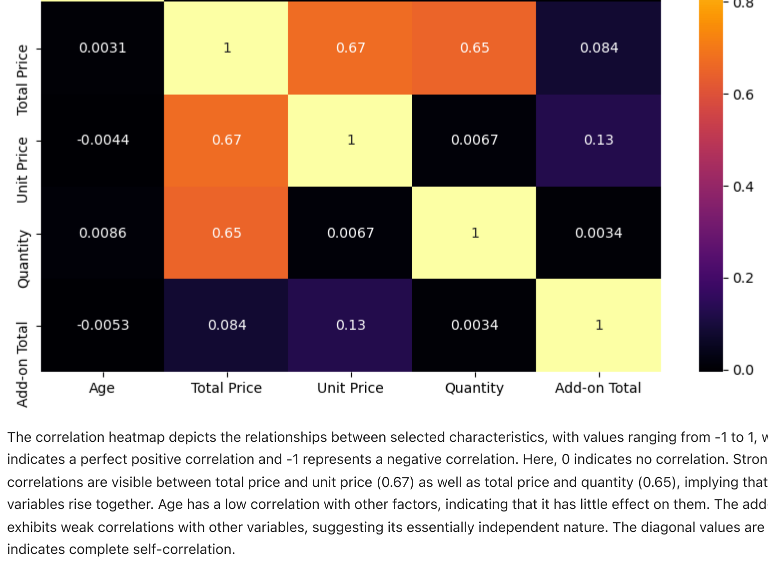
In [11]:
```
all_features = df.columns.tolist()
all_features
```

Out[11]:
```
['Customer ID',
 'Age',
 'Gender',
 'Loyalty Member',
 'Product Type',
 'SKU',
 'Rating',
 'Order Status',
 'Payment Method',
 'Total Price',
 'Unit Price',
 'Quantity',
 'Purchase Date',
 'Shipping Type',
 'Add-ons Purchased',
 'Add-on Total']
```

In [12]:
```
# Select relevant numerical features
numerical_features = df[['Age', 'Total Price', 'Unit Price', 'Quantity', 'Add-on Total']]

# Correlation matrix
correlation_matrix = numerical_features.corr()
print(correlation_matrix)
```

```
                   Age  Total Price  Unit Price  Quantity  Add-on Total
Age           1.000000     0.003096   -0.004402  0.008555     -0.005291
Total Price   0.003096     1.000000    0.673951  0.653872      0.083924
Unit Price   -0.004402     0.673951    1.000000  0.006715      0.125189
Quantity      0.008555     0.653872    0.006715  1.000000      0.003419
Add-on Total -0.005291     0.083924    0.125189  0.003419      1.000000
```

In [13]:
```
# Visualize the correlation matrix using a heatmap
plt.figure(figsize=(10, 6))
sns.heatmap(correlation_matrix, annot=True, cmap='inferno')
plt.title('Correlation Heatmap of Selected Features', fontsize=16)
plt.show()
```



The correlation heatmap depicts the relationships between selected characteristics, with values ranging from -1 to 1, where 1 indicates a perfect positive correlation and -1 represents a negative correlation. Here, 0 indicates no correlation. Strong positive correlations are visible between total price and unit price (0.67) as well as total price and quantity (0.65), implying that these variables rise together. Age has a low correlation with other factors, indicating that it has little effect on them. The add-on total exhibits weak correlations with other variables, suggesting its essentially independent nature. The diagonal values are all 1, which indicates complete self-correlation.

Section 4 – Clustering

In [14]:
```
scaler = StandardScaler()
scaled_features = scaler.fit_transform(numerical_features)
```

In [15]:
```
# Apply K-Means clustering
kmeans = KMeans(n_clusters=4, n_init=10, random_state=42)
df['Cluster'] = kmeans.fit_predict(scaled_features)
```

In [16]:
```
# Adjust pandas settings to display full content without truncation
pd.set_option('display.max_rows', None)
pd.set_option('display.max_colwidth', None)

# Group clusters and display the full list of 'Product Type' for each group
product_type_cluster = df.groupby('Cluster')['Product Type'].apply(list)

for cluster, product_types in product_type_cluster.items():
    print(f"Cluster {cluster}:")
    print(set(product_types))  # Show unique product types for each cluster
    print("\n")

pd.reset_option('display.max_rows')
pd.reset_option('display.max_colwidth')
```

```
Cluster 0:
{'Laptop', 'Headphones', 'Tablet', 'Smartwatch', 'Smartphone'}


Cluster 1:
{'Laptop', 'Headphones', 'Tablet', 'Smartwatch', 'Smartphone'}


Cluster 2:
{'Smartwatch', 'Laptop', 'Tablet', 'Smartphone'}


Cluster 3:
{'Laptop', 'Headphones', 'Tablet', 'Smartwatch', 'Smartphone'}
```

In [17]:
```
# Calculate and print silhouette and Davies-Bouldin scores
silhouette_avg = silhouette_score(scaled_features, df['Cluster'])
print(f'Silhouette score: {silhouette_avg}')

davies_bouldin_avg = davies_bouldin_score(scaled_features, df['Cluster'])
print(f'Davies-Bouldin score: {davies_bouldin_avg}')
```

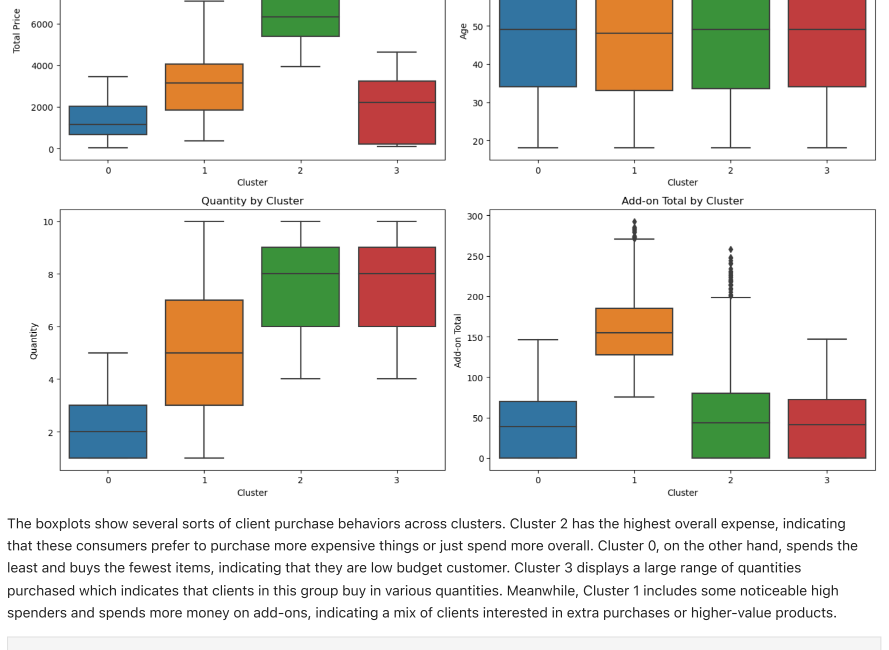
```
Silhouette score: 0.22287295287171419
Davies-Bouldin score: 1.4432039204777931
```

For k = 4, Cluster 0 includes a wide range of consumer electronics such as tablets, headphones, smartwatches, laptops, and smartphones which indicates overlap between product categories. Cluster 1 contains a similar mix of these products, but with small differences in grouping. This indicates that these items are commonly purchased together across various clusters. Cluster 2 captures focuses on tablets, smartwatches, smartphones, and laptops, which may indicate more tech savvy or high spending clients. Cluster 3 also overlaps with other clusters, which include tablets, headphones, smartwatches, and smartphones which is distinguishing clear customer segments based on the selected features.

The Davies-Bouldin score of 1.4432 and the silhouette score of 0.2228 indicate a strong overlap in clusters and a weak separation between them. This suggests that the product categories might not have clear clusters. To capture more unique purchase patterns we need further dimensions the clustering to demonstrates little significant distinction.

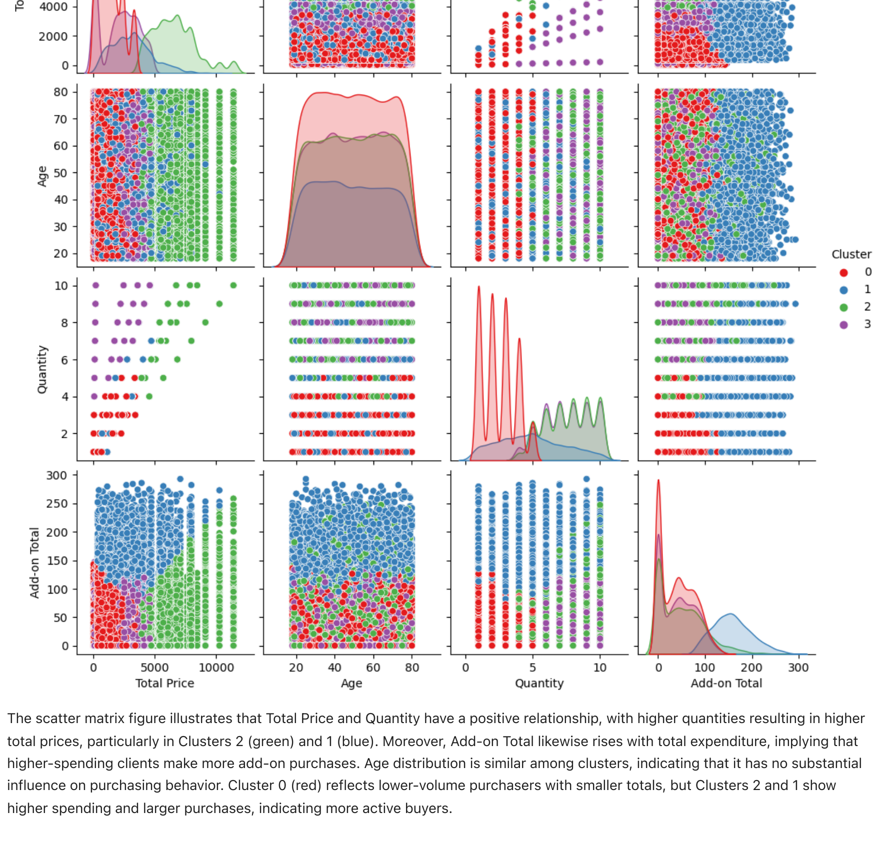
Section 5 – Plots

In [18]:
```
# 5.1. Use Box Plots
features3 = ['Total Price', 'Age', 'Quantity', 'Add-on Total']

# Create a 2x2 plot grid to accommodate the four selected features
fig, axes = plt.subplots(2, 2, figsize=(14, 10))  # Adjusted to 2x2 for four features

# Loop over the variables and create a box plot for each one in a subplot
for var, ax in zip(features3, axes.flatten()):
    sns.boxplot(x='Cluster', y=var, data=df, ax=ax)
    ax.set_title(f'{var} by Cluster')

# Adjust layout to prevent overlap
plt.tight_layout()

# Show the plots
plt.show()
```



The boxplots show several sorts of client purchase behaviors across clusters. Cluster 2 has the highest overall expense, indicating that these consumers prefer to purchase more expensive things or just spend more overall. Cluster 0, on the other hand, spends the least and buys the fewest items, indicating that they are low budget customer. Cluster 3 displays a large range of quantities purchased which indicates that clients in this group buy in various quantities. Meanwhile, Cluster 1 includes some noticeable high spenders and spends more money on add-ons, indicating a mix of clients interested in extra purchases or higher-value products.

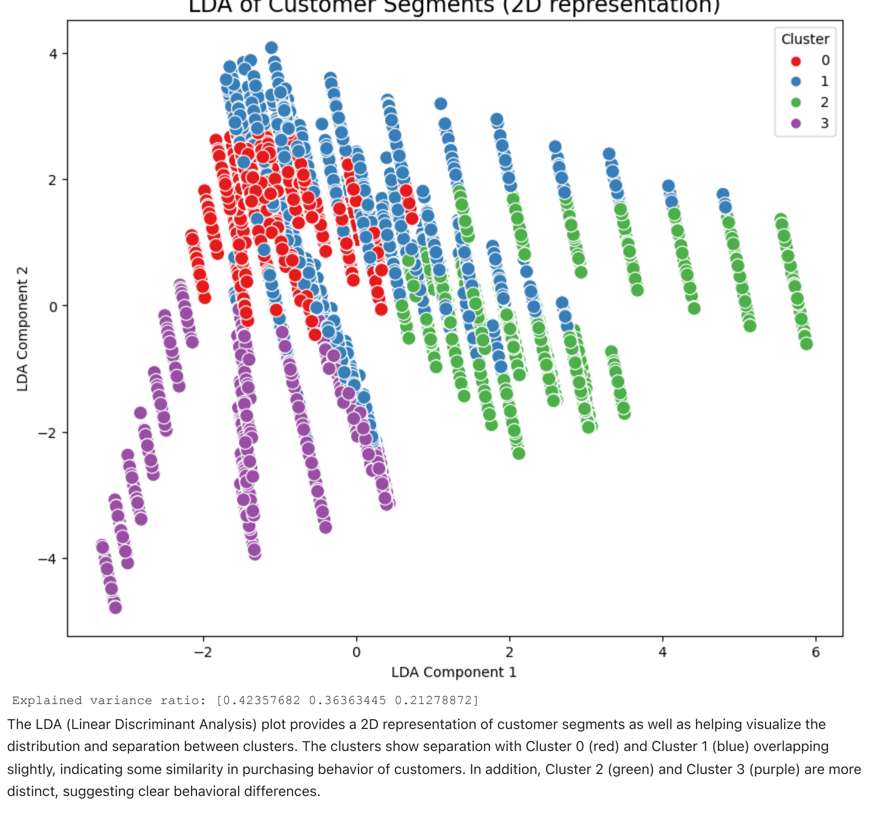
In [19]:
```
# 5.3. Scatter matrix plot with clusters

# Create a pair plot (scatter matrix) with the selected features and color by cluster
sns.pairplot(df[features3 + ['Cluster']], hue='Cluster', palette='Set1')  # MAKE MODIFICATIONS - features3 or c

# Show the plot
plt.show()
```



The scatter matrix figure illustrates that Total Price and Quantity have a positive relationship, with higher quantities resulting in higher total prices, particularly in Clusters 2 (green) and 1 (blue). Moreover, Add-on Total likewise rises with total expenditure, implying that higher-spending clients make more add-on purchases. Age distribution is similar among clusters, indicating that it has no substantial influence on purchasing behavior. Cluster 0 (red) reflects lower-volume purchasers with smaller totals, but Clusters 2 and 1 show higher spending and larger purchases, indicating more active buyers.

Section 6 – LDA

In [20]:
```
# SECTION 6

# 6.1. Select features and labels for LDA
features1 = ['Total Price', 'Age', 'Quantity', 'Add-on Total']
X = df[features1]  # Features from your dataset
y = df['Cluster']  # Cluster labels

# 6.2. Apply LDA to reduce dimensions
lda = LDA(n_components=min(len(features1), len(df['Cluster'].unique()) - 1))
X_lda = lda.fit_transform(X, y)

# Create a DataFrame with the LDA results and the corresponding cluster
lda_df = pd.DataFrame(X_lda, columns=[f'LDA{i+1}' for i in range(X_lda.shape[1])])
lda_df['Cluster'] = y

# Plot the LDA-transformed data points (just the first two components for 2D representation)
plt.figure(figsize=(10, 8))
sns.scatterplot(x='LDA1', y='LDA2', hue='Cluster', palette='Set1', data=lda_df, s=100)

# Add titles and labels
plt.title('LDA of Customer Segments (2D representation)', fontsize=16)  # Title of the plot
plt.xlabel('LDA Component 1')  # Label for x-axis
plt.ylabel('LDA Component 2')  # Label for y-axis

# Show the plot
plt.show()

# Print the explained variance ratio for all LDA components
print(f'Explained variance ratio: {lda.explained_variance_ratio_}')
```




```
Explained variance ratio: [0.42357682 0.36363445 0.21278872]
```

The LDA (Linear Discriminant Analysis) plot provides a 2D representation of customer segments as well as helping visualize the distribution and separation between clusters. The clusters show separation with Cluster 0 (red) and Cluster 1 (blue) overlapping slightly, indicating some similarity in purchasing behavior of customers. In addition, Cluster 2 (green) and Cluster 3 (purple) are more distinct, suggesting clear behavioral differences.

The explained variance ratio values (0.42357682 0.36363445 0.21278872) indicate that the first component captures the most information, followed by the second and third. These components effectively reduce dimensionality while retaining key information. Although the reduction to 2D helps in understanding the cluster structure visually where some overlaps suggest that additional dimensions might provide further clarity. Overall, LDA aids in interpreting the data but also highlights where clusters may not be entirely distinct.

Section 7 Bonus

The analysis of customer purchasing behavior using clustering and Linear Discriminant Analysis (LDA) revealed distinct segments of this dataset. By applying K-means clustering, it identified multiple customer clusters characteristics. The subsequent box plots showed differences in features such as Total Price and Quantity across these clusters while the scatter matrix highlighted the relationships between key variables.

Cluster 0 indicates lower-volume buyers with minimum add-on purchases, and Cluster 2 represents high-spending customers with bigger quantities. A wide range of behaviors, including high total prices with outliers are displayed by Cluster 1. Moderate consumers with consistent buying habits make up in Cluster 3.

Here, age has no impact on purchasing behavior and according to the scatter matrix and LDA plot, whereas Total Price and Quantity are strongly correlated. Some overlaps imply that client actions might not always easily fit into discrete categories even though the LDA plot offers helpful visualization about the cluster separation. These results might assist companies in customizing promotions and marketing plans to successfully target particular customer groups.