Thesis: Predicting E-Commerce Marketing Analytics Using Machine Learning and Big Data Analytics Code Paper

Didem B. Aykurt

Colorado State University Global

MIS581: Business Intelligence and Data Analytics

Dr. Steve Chung

January 14,2024

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

In [2]:
```
# Load the datasets
online_sales = pd.read_csv('Online_Sales.csv')
customers_data = pd.read_excel('CustomersData.xlsx', sheet_name='Customers')
discount_coupon = pd.read_csv('Discount_Coupon.csv')
marketing_spend = pd.read_csv('Marketing_Spend.csv')
tax_amount = pd.read_excel('Tax_amount.xlsx', sheet_name='GSTDetails')
```

In [3]:
```
# Display basic information about the datasets
print(online_sales.info())
print(customers_data.info())
print(discount_coupon.info())
print(marketing_spend.info())
print(tax_amount.info())
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 52924 entries, 0 to 52923
Data columns (total 10 columns):
 #   Column               Non-Null Count  Dtype  
---  ------               --------------  -----  
 0   CustomerID           52924 non-null  int64  
 1   Transaction_ID       52924 non-null  int64  
 2   Transaction_Date     52924 non-null  object 
 3   Product_SKU          52924 non-null  object 
 4   Product_Description  52924 non-null  object 
 5   Product_Category     52924 non-null  object 
 6   Quantity             52924 non-null  int64  
 7   Avg_Price            52924 non-null  float64
 8   Delivery_Charges     52924 non-null  float64
 9   Coupon_Status        52924 non-null  object 
dtypes: float64(2), int64(3), object(5)
memory usage: 4.0+ MB
None
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1468 entries, 0 to 1467
Data columns (total 4 columns):
 #   Column         Non-Null Count  Dtype 
---  ------         --------------  ----- 
 0   CustomerID     1468 non-null   int64 
 1   Gender         1468 non-null   object
 2   Location       1468 non-null   object
 3   Tenure_Months  1468 non-null   int64 
dtypes: int64(2), object(2)
memory usage: 46.0+ KB
None
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 204 entries, 0 to 203
Data columns (total 4 columns):
 #   Column            Non-Null Count  Dtype 
---  ------            --------------  ----- 
 0   Month             204 non-null    object
 1   Product_Category  204 non-null    object
 2   Coupon_Code       204 non-null    object
 3   Discount_pct      204 non-null    int64 
dtypes: int64(1), object(3)
memory usage: 6.5+ KB
None
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 365 entries, 0 to 364
Data columns (total 3 columns):
 #   Column         Non-Null Count  Dtype  
---  ------         --------------  -----  
 0   Date           365 non-null    object 
 1   Offline_Spend  365 non-null    int64  
 2   Online_Spend   365 non-null    float64
dtypes: float64(1), int64(1), object(1)
memory usage: 8.7+ KB
None
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 20 entries, 0 to 19
Data columns (total 2 columns):
 #   Column            Non-Null Count  Dtype  
---  ------            --------------  -----  
 0   Product_Category  20 non-null     object 
 1   GST               20 non-null     float64
```

```
dtypes: float64(1), object(1)
memory usage: 452.0+ bytes
None
```

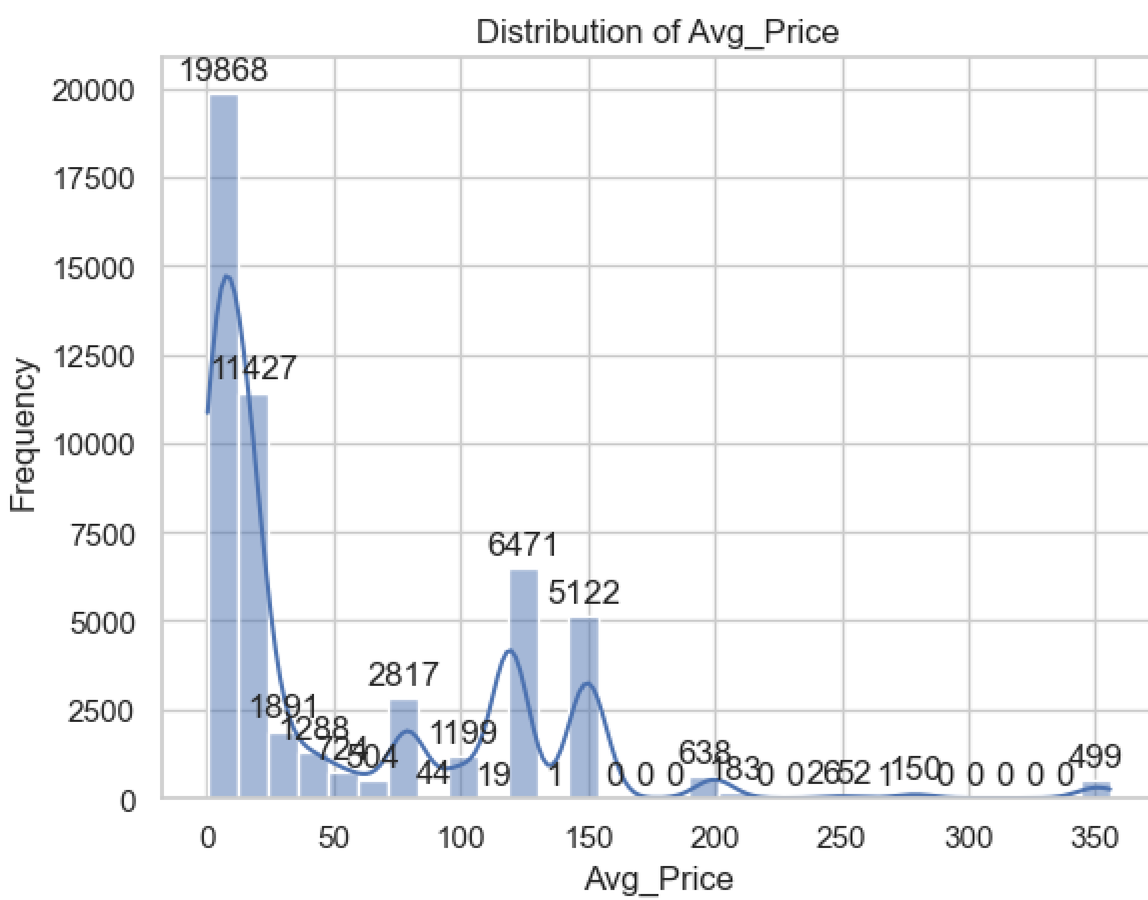
In [4]:

```python
import seaborn as sns
import matplotlib.pyplot as plt
# Assuming 'Avg_Price' is the numeric variable in the 'online_sales' dataset
numeric_variable = 'Avg_Price'
plt.figure(figsize=(50, 5))
sns.set(style="whitegrid")
```

```
<Figure size 5000x500 with 0 Axes>
```

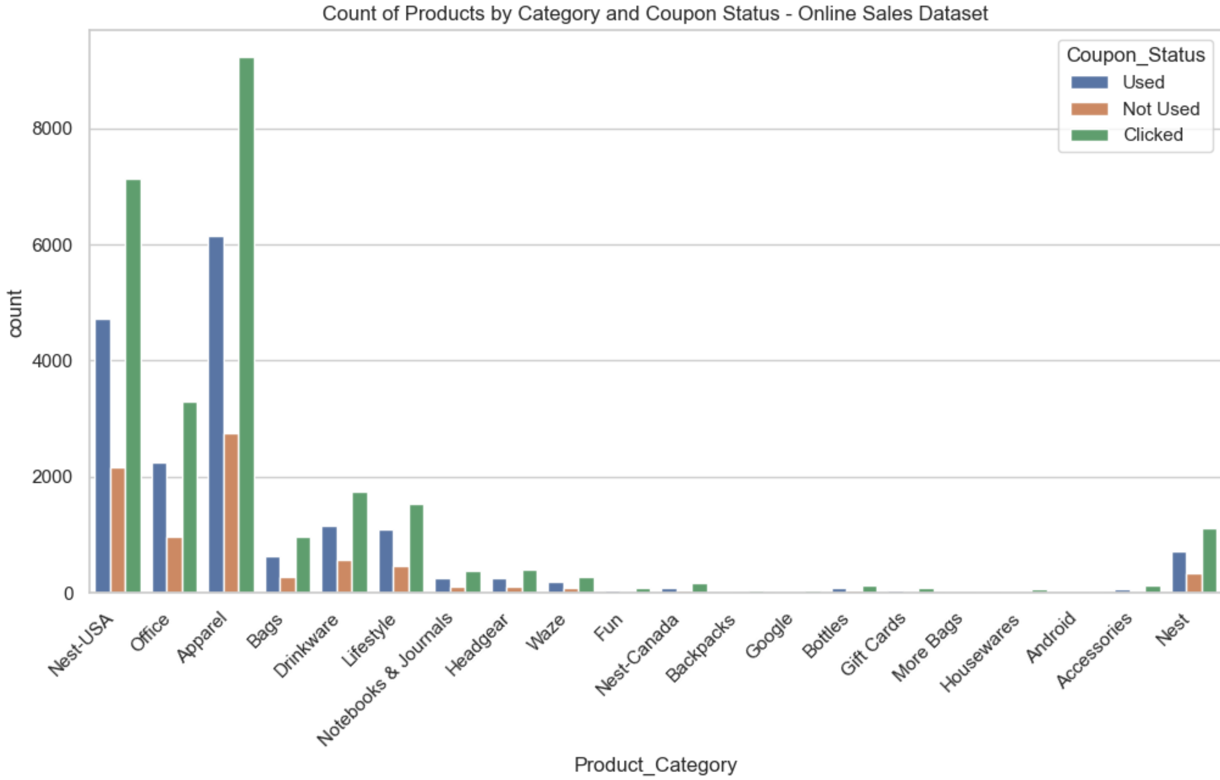
In [5]:

```python
# Plot the histogram with count annotations",
ax = sns.histplot(online_sales[numeric_variable], kde=True, bins=30)
for rect in ax.patches:
    height = rect.get_height()
    ax.annotate(f'{int(height)}', xy=(rect.get_x() + rect.get_width() / 2, height),
                xytext=(0, 3), textcoords='offset points', ha='center', va='bottom')

plt.title(f'Distribution of {numeric_variable}')
plt.xlabel(numeric_variable)
plt.ylabel('Frequency')
plt.show()
```



In [6]:

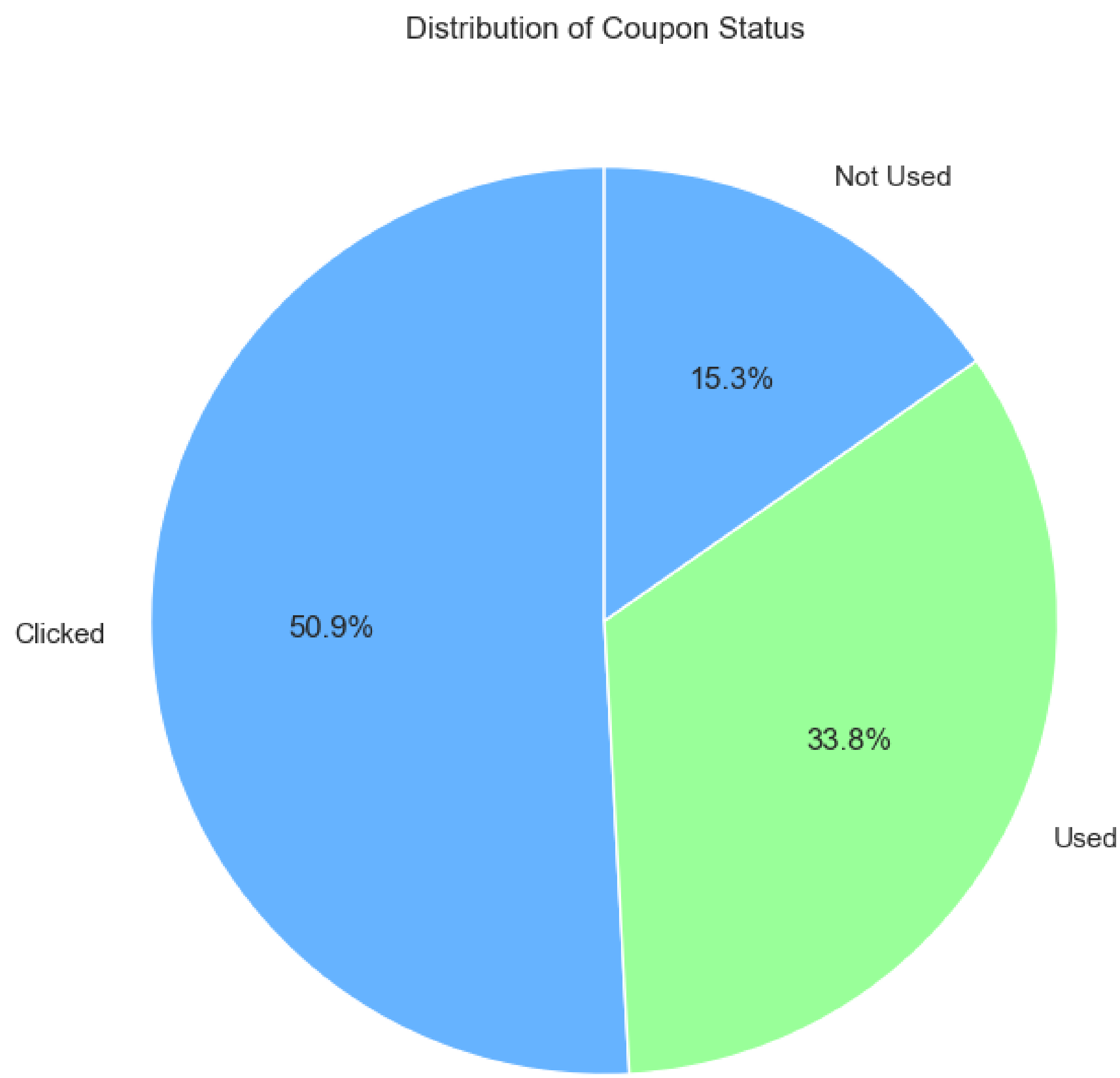
```python
# Visualize categorical variables in Online Sales dataset
plt.figure(figsize=(12, 6))
sns.countplot(x='Product_Category', data=online_sales, hue='Coupon_Status')
plt.title('Count of Products by Category and Coupon Status - Online Sales Dataset')
plt.xticks(rotation=45, ha='right')  # Rotate x-axis labels for better visibility
plt.show()
```



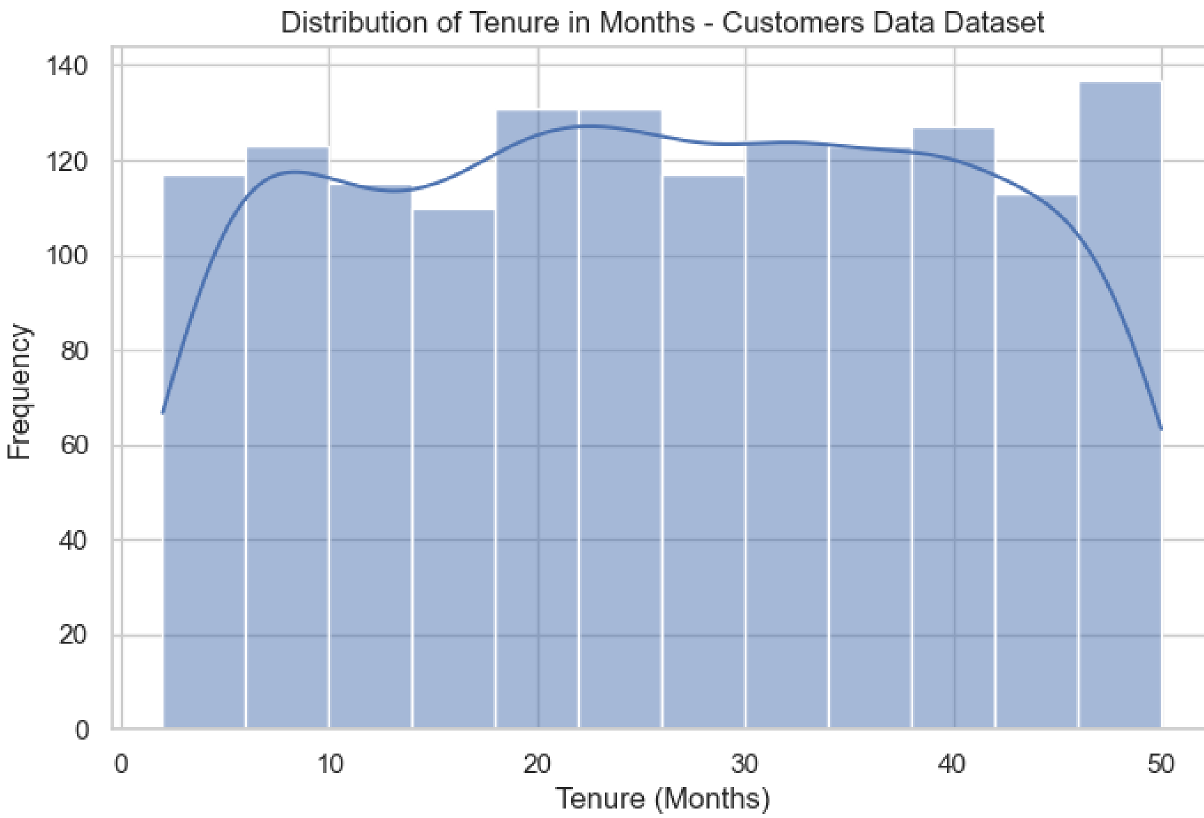
In [7]:

```
# Assuming 'Coupon_Status' is a categorical variable in the 'Online_Sales' dataset\n",
online_sales = pd.read_csv('Online_Sales.csv')
# Count the occurrences of each coupon status
coupon_counts = online_sales['Coupon_Status'].value_counts()

print(coupon_counts)
#Create a pie chart
plt.figure(figsize=(8, 8))
plt.pie(coupon_counts, labels=coupon_counts.index, autopct='%1.1f%%', startangle=90, c
plt.title('Distribution of Coupon Status')
plt.show()
```

```
Coupon_Status
Clicked     26926
Used        17904
Not Used     8094
Name: count, dtype: int64
```



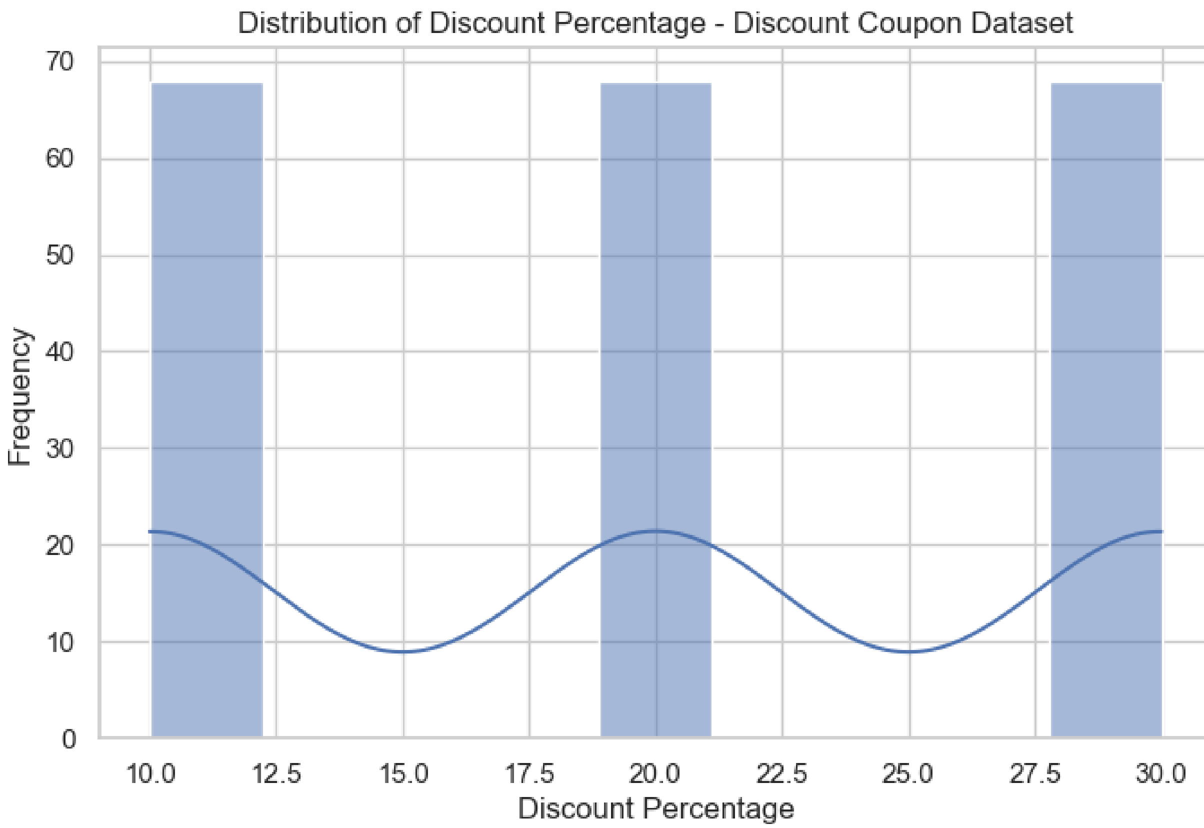
In [8]:
```
# Assuming 'Tenure_Months' is the variable in the 'customers_data' dataset
plt.figure(figsize=(8, 5))
sns.histplot(customers_data['Tenure_Months'], kde=True)
plt.title('Distribution of Tenure in Months - Customers Data Dataset')
plt.xlabel('Tenure (Months)')
plt.ylabel('Frequency')
plt.show()
# Print summary statistics of the 'Tenure_Months' variable
tenure_stats = customers_data['Tenure_Months'].describe()
print('Summary Statistics of Tenure_Months')
print(tenure_stats)
```




```
Summary Statistics of Tenure_Months
count    1468.000000
mean       25.912125
std        13.959667
min         2.000000
25%        14.000000
50%        26.000000
75%        38.000000
max        50.000000
Name: Tenure_Months, dtype: float64
```

In [9]:

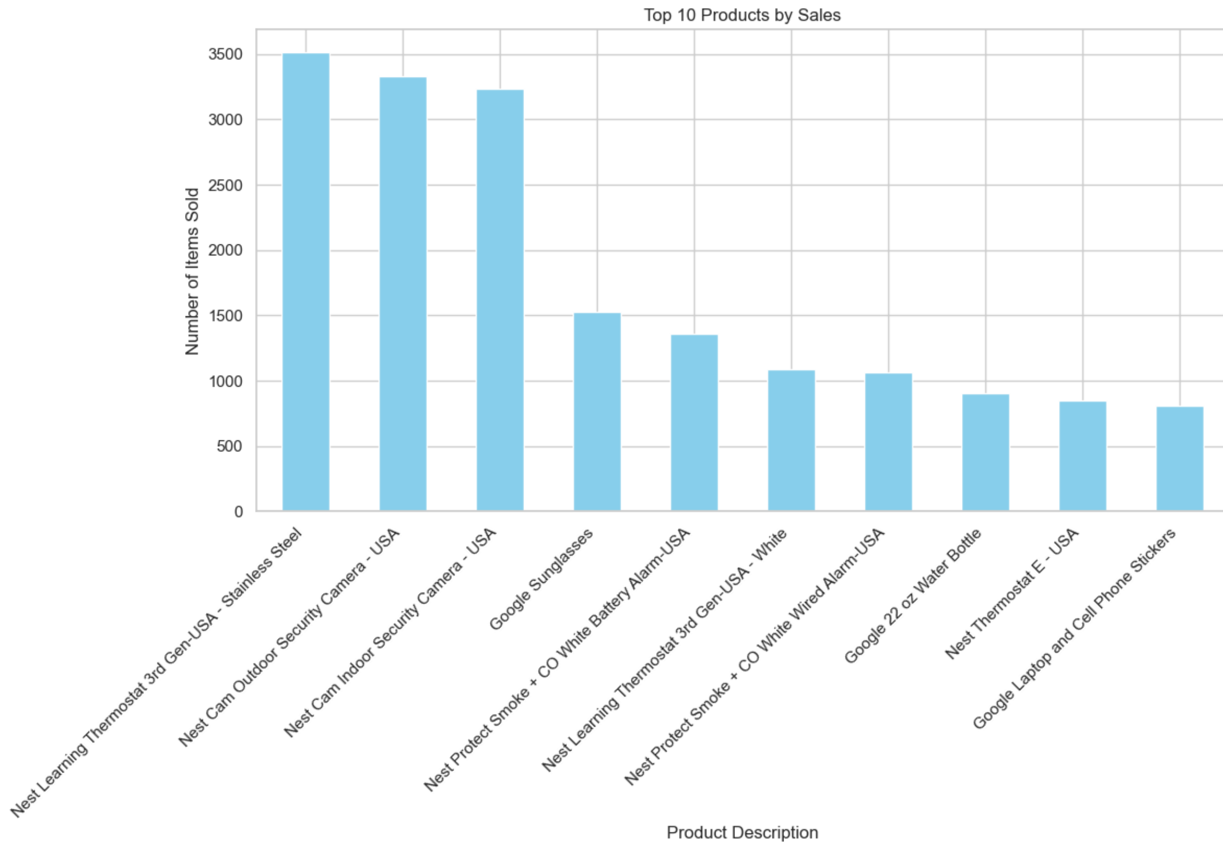
```python
# Assuming 'Discount_pct' is the variable in the 'discount_coupon' dataset
plt.figure(figsize=(8, 5))
sns.histplot(discount_coupon['Discount_pct'], kde=True)
plt.title('Distribution of Discount Percentage - Discount Coupon Dataset')
plt.xlabel('Discount Percentage')
plt.ylabel('Frequency')
plt.show()
```



In [10]:
```
# Group by discount percentage and count the number of coupons for each percentage
coupon_counts_by_percentage = discount_coupon['Discount_pct'].value_counts().reset_inc
coupon_counts_by_percentage.columns = ['Discount Percentage', 'Coupon Count']
print('Coupon Counts by Discount Percentage')
print(coupon_counts_by_percentage)
```

```
Coupon Counts by Discount Percentage
   Discount Percentage  Coupon Count
0                   10            68
1                   20            68
2                   30            68
```

In [11]:
```
# Assuming 'Product_Description' is the variable representing product names in the 'On
online_sales = pd.read_csv('Online_Sales.csv')
# Calculate the top products by counting occurrences
top_products = online_sales['Product_Description'].value_counts().nlargest(10)
# Create a bar chart for the top products
plt.figure(figsize=(12, 6))
top_products.plot(kind='bar', color='skyblue')
plt.title('Top 10 Products by Sales')
plt.xlabel('Product Description')
plt.ylabel('Number of Items Sold')
plt.xticks(rotation=45, ha='right')  # Rotate x-axis labels for better visibility
plt.show()
print(top_products)
```



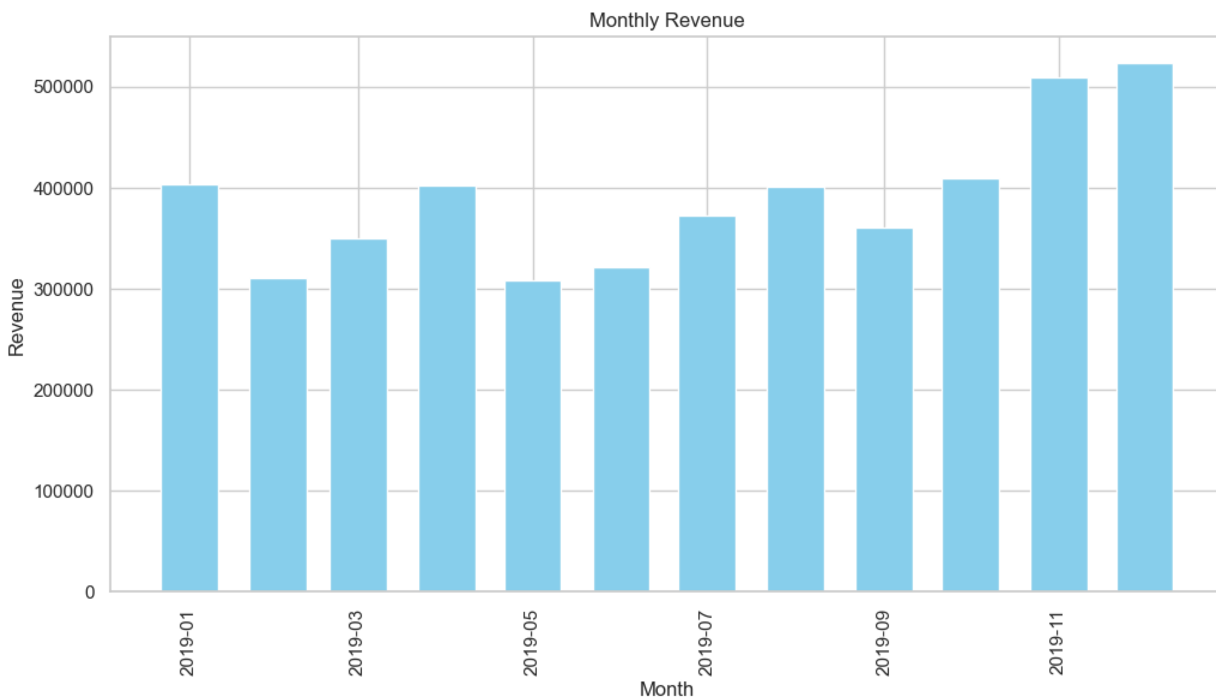

```
Product_Description
Nest Learning Thermostat 3rd Gen-USA - Stainless Steel    3511
Nest Cam Outdoor Security Camera - USA                    3328
Nest Cam Indoor Security Camera - USA                     3230
Google Sunglasses                                         1523
Nest Protect Smoke + CO White Battery Alarm-USA           1361
Nest Learning Thermostat 3rd Gen-USA - White              1089
Nest Protect Smoke + CO White Wired Alarm-USA             1065
Google 22 oz Water Bottle                                  902
Nest Thermostat E - USA                                    844
Google Laptop and Cell Phone Stickers                      806
Name: count, dtype: int64
```

In [26]:

```
online_sales['Transaction_Date'] = pd.to_datetime(online_sales['Transaction_Date'])

# Create a new column for 'Revenue' by multiplying 'Quantity' and 'Avg_Price'
online_sales['Revenue'] = online_sales['Quantity'] * online_sales['Avg_Price']

# Group by month and sum the revenue for each month
monthly_revenue = online_sales.groupby(online_sales['Transaction_Date'].dt.to_period('

# Convert the 'Transaction_Date' back to datetime for plotting
monthly_revenue['Transaction_Date'] = monthly_revenue['Transaction_Date'].dt.to_timest

# Print the monthly revenue
print("Monthly Revenue:")
print(monthly_revenue)

# Visualize the monthly revenue
plt.figure(figsize=(12, 6))
plt.bar(monthly_revenue['Transaction_Date'], monthly_revenue['Revenue'], color='skyblu
plt.title('Monthly Revenue')
plt.xlabel('Month')
```

```
plt.ylabel('Revenue')
plt.xticks(rotation=90)  # Rotate x-axis labels for better readability
plt.show()
```

```
Monthly Revenue:
   Transaction_Date    Revenue
0        2019-01-01  403624.58
1        2019-02-01  310819.80
2        2019-03-01  349608.09
3        2019-04-01  401618.42
4        2019-05-01  307763.42
5        2019-06-01  321081.38
6        2019-07-01  372638.07
7        2019-08-01  401210.37
8        2019-09-01  360548.40
9        2019-10-01  409681.28
10       2019-11-01  508942.62
11       2019-12-01  523258.19
```



In [13]:

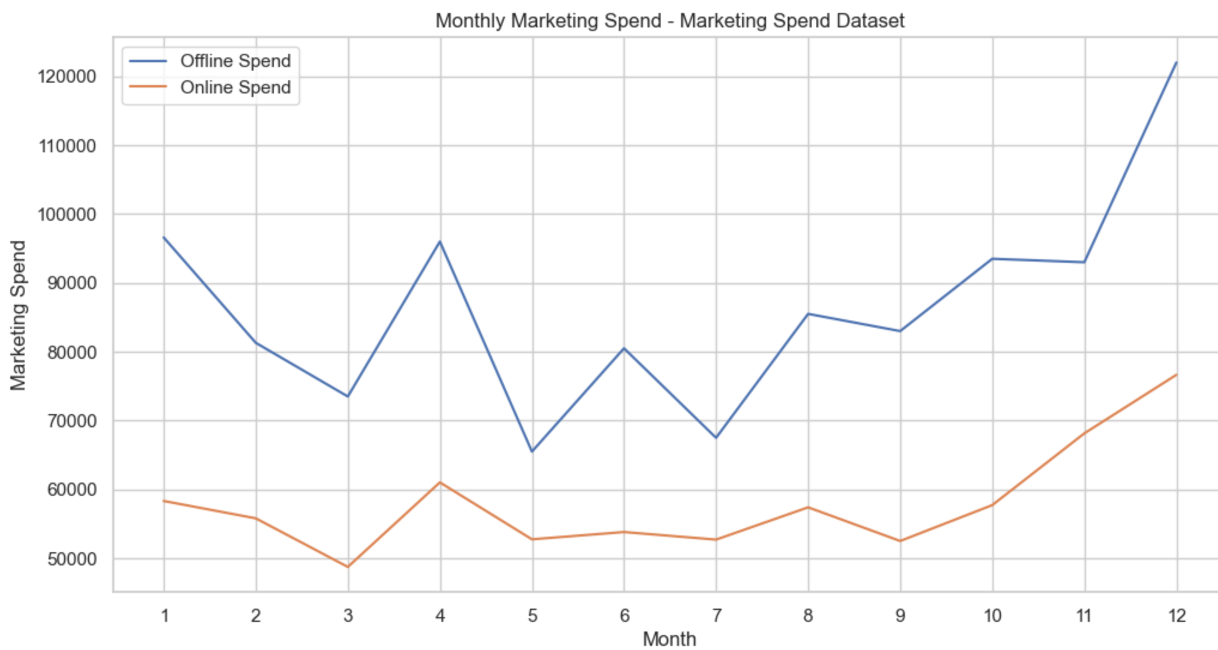
```
print(online_sales['Transaction_Date'])
```

```
0       2019-01-01
1       2019-01-01
2       2019-01-01
3       2019-01-01
4       2019-01-01
           ...
52919   2019-12-31
52920   2019-12-31
52921   2019-12-31
52922   2019-12-31
52923   2019-12-31
Name: Transaction_Date, Length: 52924, dtype: datetime64[ns]
```

In [14]:

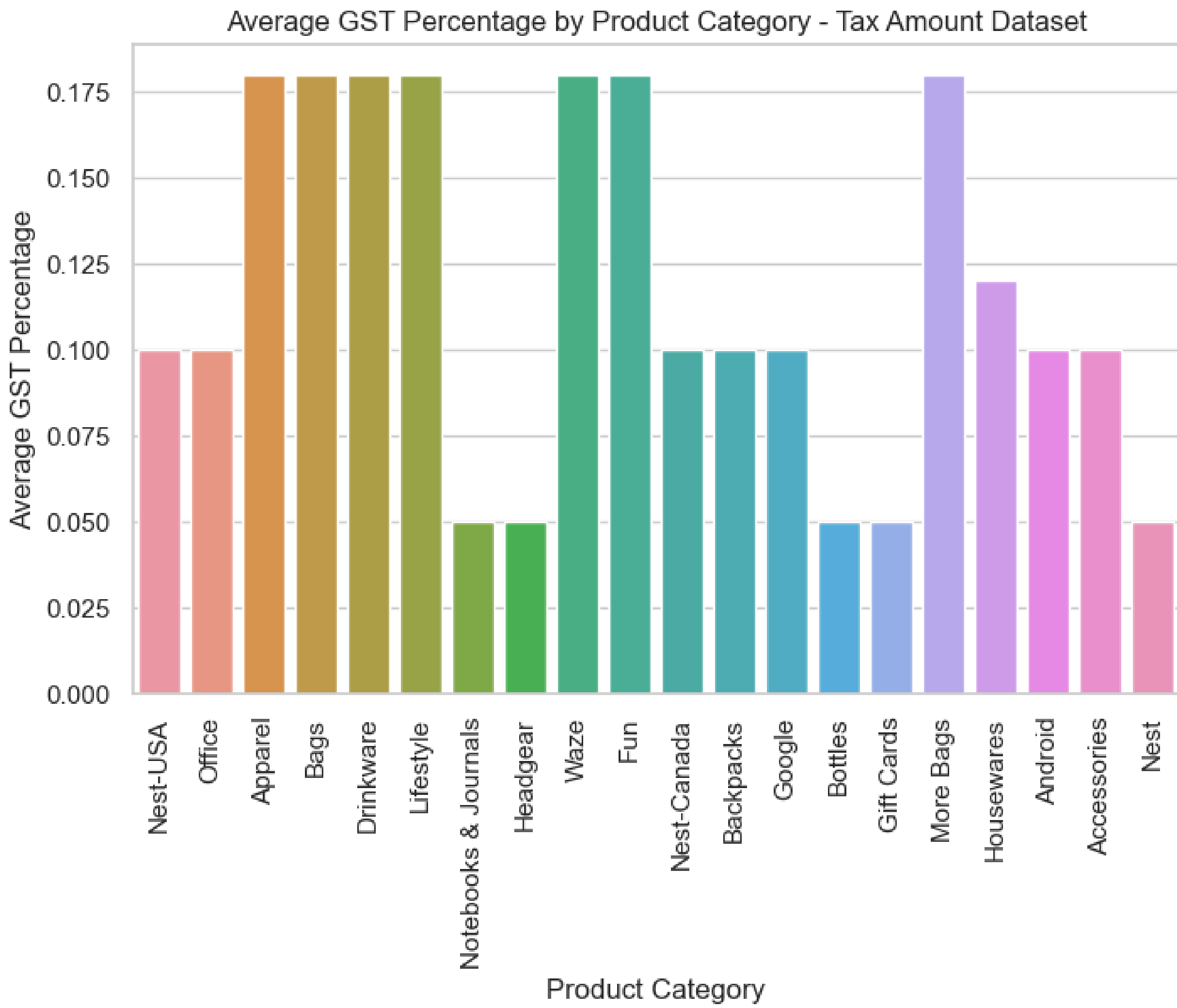
```
# Make sure the 'Date' column is in datetime format
marketing_spend['Date'] = pd.to_datetime(marketing_spend['Date'])
# Create a new column for the month
marketing_spend['Month'] = marketing_spend.Date.dt.month
```

```python
# Select only the numeric columns
numeric_cols = marketing_spend.select_dtypes(['int', 'float']).columns
# Group by month and sum the values
monthly_spend = marketing_spend.groupby(marketing_spend.Date.dt.month)[numeric_cols].s
# Plot the data
plt.figure(figsize=(12, 6))
sns.lineplot(x=monthly_spend.index.astype(str), y=monthly_spend['Offline_Spend'], labe
sns.lineplot(x=monthly_spend.index.astype(str), y=monthly_spend['Online_Spend'], label
plt.title('Monthly Marketing Spend - Marketing Spend Dataset')
plt.xlabel('Month')
plt.ylabel('Marketing Spend')
plt.legend()
plt.show()
```



In [15]:

```python
# Visualize GST Percentage distribution in Tax Amount dataset
plt.figure(figsize=(8, 5))
sns.barplot(x='Product_Category', y='GST', data=tax_amount)
plt.title('Average GST Percentage by Product Category - Tax Amount Dataset')
plt.xticks(rotation=90)
plt.xlabel('Product Category')
plt.ylabel('Average GST Percentage')
plt.show()
```



In [16]:

```
import matplotlib.pyplot as plt
from sklearn.datasets import make_blobs
from matplotlib import pyplot as plt
%matplotlib inline
# Select relevant features for clustering
features_for_clustering = customers_data[['Tenure_Months']]
# Standardize the features
scaler = StandardScaler()
features_for_clustering_scaled = scaler.fit_transform(features_for_clustering)
# Determine optimal number of clusters using silhouette score
best_num_clusters = 2  # Set the initial number of clusters
best_silhouette_score = -1
#initialize kmeans parameters
k_rng=range(2,6)
for num_clusters in k_rng:  # Adjust the range as needed
    kmeans = KMeans(n_clusters=num_clusters, random_state=42)
    cluster_labels = kmeans.fit_predict(features_for_clustering_scaled)
    silhouette_avg = silhouette_score(features_for_clustering_scaled, cluster_labels)
    if silhouette_avg > best_silhouette_score:
        best_silhouette_score = silhouette_avg
        best_num_clusters = num_clusters
# Apply K-Means clustering with the optimal number of clusters
kmeans = KMeans(n_clusters=best_num_clusters, random_state=42)
customers_data['Cluster_Label'] = kmeans.fit_predict(features_for_clustering_scaled)
```

```
C:\ProgramData\anaconda3\Lib\site-packages\sklearn\cluster\_kmeans.py:1412: FutureWar
ning: The default value of `n_init` will change from 10 to 'auto' in 1.4. Set the val
ue of `n_init` explicitly to suppress the warning
  super()._check_params_vs_input(X, default_n_init=10)
C:\ProgramData\anaconda3\Lib\site-packages\sklearn\cluster\_kmeans.py:1412: FutureWar
ning: The default value of `n_init` will change from 10 to 'auto' in 1.4. Set the val
ue of `n_init` explicitly to suppress the warning
  super()._check_params_vs_input(X, default_n_init=10)
C:\ProgramData\anaconda3\Lib\site-packages\sklearn\cluster\_kmeans.py:1412: FutureWar
ning: The default value of `n_init` will change from 10 to 'auto' in 1.4. Set the val
ue of `n_init` explicitly to suppress the warning
  super()._check_params_vs_input(X, default_n_init=10)
C:\ProgramData\anaconda3\Lib\site-packages\sklearn\cluster\_kmeans.py:1412: FutureWar
ning: The default value of `n_init` will change from 10 to 'auto' in 1.4. Set the val
ue of `n_init` explicitly to suppress the warning
  super()._check_params_vs_input(X, default_n_init=10)
C:\ProgramData\anaconda3\Lib\site-packages\sklearn\cluster\_kmeans.py:1412: FutureWar
ning: The default value of `n_init` will change from 10 to 'auto' in 1.4. Set the val
ue of `n_init` explicitly to suppress the warning
  super()._check_params_vs_input(X, default_n_init=10)
```

In [17]:

```python
import matplotlib.pyplot as plt
import seaborn as sns
# Print the count of customers in each cluster
cluster_counts = customers_data['Cluster_Label'].value_counts()
print("Count of Customers in Each Cluster:")
print(cluster_counts)
# Plotting the distribution of customers across clusters
plt.figure(figsize=(5, 3))
sns.countplot(x='Cluster_Label', data=customers_data, palette='viridis')
plt.title('Customer Distribution Across Clusters')
plt.xlabel('Cluster Label')
plt.ylabel('Number of Customers')
plt.show()
```

```
Count of Customers in Each Cluster:
Cluster_Label
1    758
0    710
Name: count, dtype: int64
```

In [18]:
```
print(customers_data.Cluster_Label)
```

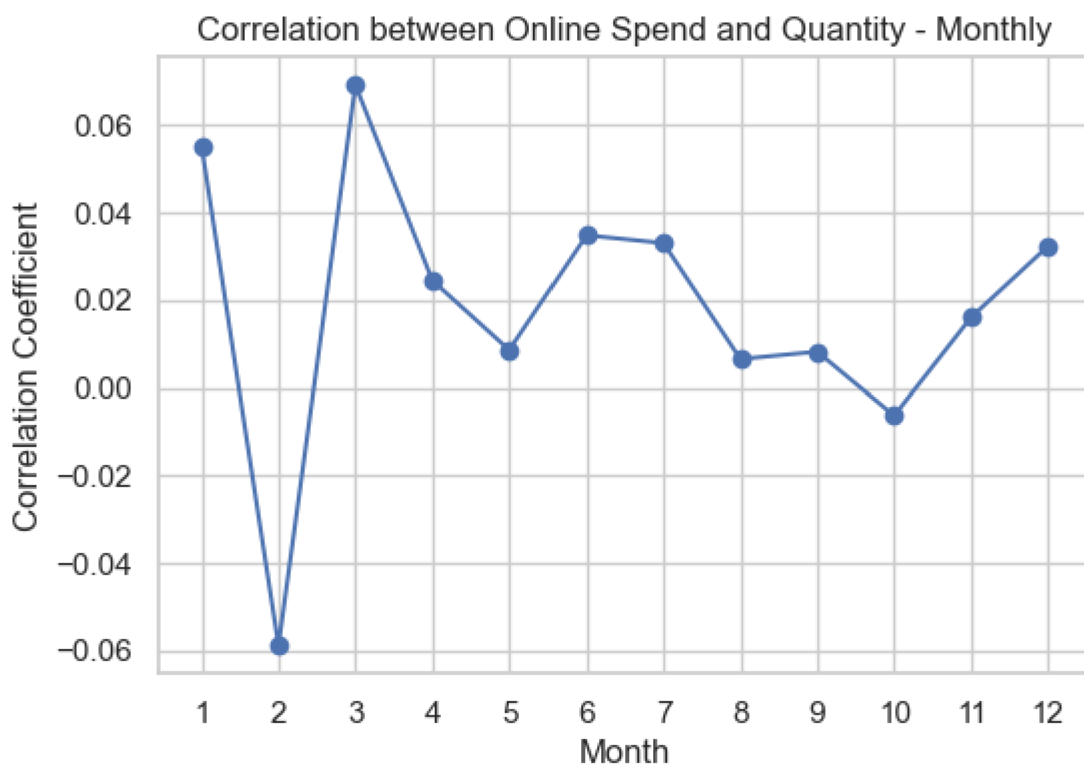
```
0       1
1       0
2       0
3       0
4       0
       ..
1463    0
1464    0
1465    1
1466    0
1467    1
Name: Cluster_Label, Length: 1468, dtype: int32
```

In [19]:
```
import pandas as pd
from scipy.stats import f_oneway, pearsonr
import statsmodels.api as sm

# Merge datasets as needed
# For example, if comparing purchasing behavior between customer segments
merged_data = pd.merge(online_sales, customers_data, on='CustomerID', how='inner')
# Extract relevant columns for hypothesis testing
grouped_data = merged_data.groupby('Cluster_Label')['Quantity'].apply(list)
# Perform ANOVA for Customer Segmentation
anova_result = f_oneway(*grouped_data)
# Print ANOVA result
print("ANOVA Result for Customer Segmentation:")
print(anova_result)
```

```
ANOVA Result for Customer Segmentation:
F_onewayResult(statistic=7.785165506101219, pvalue=0.005269580727491623)
```

In [20]:
```
marketing_spend['Date'] = pd.to_datetime(marketing_spend['Date'])
online_sales['Transaction_Date'] = pd.to_datetime(online_sales['Transaction_Date'])
# Merge datasets on the 'Date' column
merged_data_monthly = pd.merge(online_sales, marketing_spend, left_on='Transaction_Dat
# Group by month and calculate correlation for each month
correlation_per_month = []
months = merged_data_monthly['Date'].dt.month.unique()
for month in months:
        monthly_data = merged_data_monthly[merged_data_monthly['Date'].dt.month == mor
        correlation, p_value = pearsonr(monthly_data['Online_Spend'], monthly_data['Qu
        correlation_per_month.append((month, correlation, p_value))

# Create a DataFrame from the correlation results
correlation_df = pd.DataFrame(correlation_per_month, columns=['Month', 'Correlation',

# Print the correlation results
print("Correlation Results for Each Month:")
print(correlation_df)

# Visualize the correlation results
plt.figure(figsize=(6, 4))
plt.plot(correlation_df['Month'], correlation_df['Correlation'], marker='o', linestyle
plt.title('Correlation between Online Spend and Quantity - Monthly')
plt.xlabel('Month')
plt.ylabel('Correlation Coefficient')
plt.xticks(months)  # Ensure all months are displayed on the x-axis
plt.show()
```

```
Correlation Results for Each Month:
    Month  Correlation   P-value
0       1     0.055157  0.000436
1       2    -0.058863  0.000738
2       3     0.069173  0.000005
3       4     0.024510  0.114397
4       5     0.008602  0.560930
5       6     0.034807  0.024205
6       7     0.033028  0.016694
7       8     0.006526  0.608867
8       9     0.008214  0.590751
9      10    -0.006354  0.681897
10     11     0.016054  0.312443
11     12     0.032312  0.030161
```



In [21]:
```
# Extract relevant columns for ANOVA
grouped_data_monthly = [merged_data_monthly[merged_data_monthly['Date'].dt.month == mo
# Perform ANOVA
anova_result_monthly = f_oneway(*grouped_data_monthly)
# Print ANOVA result
print("ANOVA Result for Marketing Effectiveness Hypotheses (H2):")
print(anova_result_monthly)
```

```
ANOVA Result for Marketing Effectiveness Hypotheses (H2):
F_onewayResult(statistic=6.849259460432781, pvalue=1.1899761489346113e-11)
```

In [22]:
```
#Anova test for GST by product category
# For example, if comparing GST between product category
merged_data_tax = pd.merge(merged_data, tax_amount, on='Product_Category', how='inner'
# Extract relevant columns for hypothesis testing
grouped_data = merged_data_tax.groupby('Cluster_Label')['GST'].apply(list)
# Perform ANOVA for Customer Segmentation
anova_result = f_oneway(*grouped_data)
# Print ANOVA result
```

```
print("ANOVA Result for  GST Impact on Prodcut Category Hypotheses (H3):")
print(anova_result)
```

```
ANOVA Result for  GST Impact on Prodcut Category Hypotheses (H3):
F_onewayResult(statistic=2.479078982005758, pvalue=0.11537577143026681)
```

In [24]:

```
import pandas as pd
import statsmodels.api as sm
import statsmodels.formula.api as smf
import scipy.stats as stats

merged_data_tax_next = pd.merge(online_sales, tax_amount, on='Product_Category', how='
#clean null variable
df_clean = merged_data_tax_next.dropna(axis=0, how="any")
# Create a two-way ANOVA model
model =smf.ols('Quantity ~ C(GST) + C(Product_Category) + C(GST):C(Product_Category)',

# Generate an ANOVA table
anova_table = sm.stats.anova_lm(model, typ=3)

# Print the ANOVA table
print("ANOVA Result for GST Impact on Product Categories Hypotheses (H3):")
print(anova_table)
```

```
ANOVA Result for GST Impact on Product Categories Hypotheses (H3):
                                  sum_sq       df         F    PR(>F)
Intercept                   1.723600e-02      1.0  0.000045  0.994661
C(GST)                      8.655525e-01      3.0  0.000750  0.999972
C(Product_Category)         3.490048e+00     19.0  0.000477  1.000000
C(GST):C(Product_Category)  1.308968e+01     57.0  0.000597  1.000000
Residual                    2.035981e+07  52901.0       NaN       NaN
```

```
C:\ProgramData\anaconda3\Lib\site-packages\statsmodels\base\model.py:1888: ValueWarni
ng: covariance of constraints does not have full rank. The number of constraints is 1
9, but rank is 15
  warnings.warn('covariance of constraints does not have full '
C:\ProgramData\anaconda3\Lib\site-packages\statsmodels\base\model.py:1888: ValueWarni
ng: covariance of constraints does not have full rank. The number of constraints is 5
7, but rank is 12
  warnings.warn('covariance of constraints does not have full '
```

In [25]:

```
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score, classification_report, confusion_matrix
from sklearn.preprocessing import StandardScaler
from sklearn.pipeline import make_pipeline
# Assuming 'Date' is present in the 'online_sales' dataset
online_sales['Transaction_Date'] = pd.to_datetime(online_sales['Transaction_Date'])
# Create a new column for the month of the first purchase (cohort)
online_sales['Cohort_Month'] = online_sales.groupby('CustomerID')['Transaction_Date'].
# Label customers as retained (1) or not retained (0)
online_sales['Retained'] = online_sales.groupby('CustomerID')['Transaction_Date'].tran
# Feature engineering
features = pd.get_dummies(online_sales['Product_Category'], prefix='Product_Category')
features['Total_Spend'] = online_sales.groupby('CustomerID')['Avg_Price'].transform('s
features['Total_Quantity'] = online_sales.groupby('CustomerID')['Quantity'].transform(
# Select features and target variable
X = features
y = online_sales['Retained']
# Split the data into training and testing sets
```

```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=
# Build a predictive model (Logistic Regression)
model = make_pipeline(StandardScaler(), LogisticRegression())
model.fit(X_train, y_train)
# Make predictions on the test set
y_pred = model.predict(X_test)
# Evaluate the model
accuracy = accuracy_score(y_test, y_pred)
conf_matrix = confusion_matrix(y_test, y_pred)
class_report = classification_report(y_test, y_pred)
print("Accuracy: {:.2f}".format(accuracy))
print("Confusion Matrix:")
print(conf_matrix)
print("Classification Report:")
print(class_report)
```

```
Accuracy: 0.67
Confusion Matrix:
[[2229 2017]
 [1464 4875]]
Classification Report:
              precision    recall  f1-score   support

           0       0.60      0.52      0.56      4246
           1       0.71      0.77      0.74      6339

    accuracy                           0.67     10585
   macro avg       0.66      0.65      0.65     10585
weighted avg       0.67      0.67      0.67     10585
```

In [ ]: