In [1]:

```
#firstly we imported all the necessary libraries.
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error
from sklearn.preprocessing import StandardScaler
from sklearn.cluster import KMeans
from sklearn.preprocessing import LabelEncoder, OneHotEncoder
```

In [2]:

```
# Loading  the dataset that we got from kaggle
data = pd.read_csv('supermarket_sales - Sheet1.csv')
```

In [3]:

```
data
```

Out[3]:

	Invoice ID	Branch	City	Customer type	Gender	Product line	Unit price	Quantity	Tax 5%	
0	750-67-8428	A	Yangon	Member	Female	Health and beauty	74.69	7	26.1415	
1	226-31-3081	C	Naypyitaw	Normal	Female	Electronic accessories	15.28	5	3.8200	
2	631-41-3108	A	Yangon	Normal	Male	Home and lifestyle	46.33	7	16.2155	
3	123-19-1176	A	Yangon	Member	Male	Health and beauty	58.22	8	23.2880	
4	373-73-7910	A	Yangon	Normal	Male	Sports and travel	86.31	7	30.2085	
...	...	...	...	...	...	...	...	...	...	
995	233-67-5758	C	Naypyitaw	Normal	Male	Health and beauty	40.35	1	2.0175	
996	303-96-2227	B	Mandalay	Normal	Female	Home and lifestyle	97.38	10	48.6900	1
997	727-02-1313	A	Yangon	Member	Male	Food and beverages	31.84	1	1.5920	
998	347-56-2442	A	Yangon	Normal	Male	Home and lifestyle	65.82	1	3.2910	
999	849-09-3807	A	Yangon	Member	Female	Fashion accessories	88.34	7	30.9190	

1000 rows × 17 columns

In [4]:

```
# Exploring  the dataset and describing it
print(data.info())
print(data.describe())
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1000 entries, 0 to 999
Data columns (total 17 columns):
 #   Column                   Non-Null Count  Dtype  
---  ------                   --------------  -----  
 0   Invoice ID               1000 non-null   object 
 1   Branch                   1000 non-null   object 
 2   City                     1000 non-null   object 
 3   Customer type            1000 non-null   object 
 4   Gender                   1000 non-null   object 
 5   Product line             1000 non-null   object 
 6   Unit price               1000 non-null   float64
 7   Quantity                 1000 non-null   int64  
 8   Tax 5%                   1000 non-null   float64
 9   Total                    1000 non-null   float64
 10  Date                     1000 non-null   object 
 11  Time                     1000 non-null   object 
 12  Payment                  1000 non-null   object 
 13  cogs                     1000 non-null   float64
 14  gross margin percentage  1000 non-null   float64
 15  gross income             1000 non-null   float64
 16  Rating                   1000 non-null   float64
dtypes: float64(7), int64(1), object(9)
memory usage: 132.9+ KB
None
        Unit price     Quantity       Tax 5%        Total        cogs  \
count  1000.000000  1000.000000  1000.000000  1000.000000  1000.00000   
mean     55.672130     5.510000    15.379369   322.966749   307.58738   
std      26.494628     2.923431    11.708825   245.885335   234.17651   
min      10.080000     1.000000     0.508500    10.678500    10.17000   
25%      32.875000     3.000000     5.924875   124.422375   118.49750   
50%      55.230000     5.000000    12.088000   253.848000   241.76000   
75%      77.935000     8.000000    22.445250   471.350250   448.90500   
max      99.960000    10.000000    49.650000  1042.650000   993.00000   

       gross margin percentage  gross income      Rating  
count             1.000000e+03   1000.000000  1000.00000  
mean              4.761905e+00     15.379369     6.97270  
std               6.131498e-14     11.708825     1.71858  
min               4.761905e+00      0.508500     4.00000  
25%               4.761905e+00      5.924875     5.50000  
50%               4.761905e+00     12.088000     7.00000  
75%               4.761905e+00     22.445250     8.50000  
max               4.761905e+00     49.650000    10.00000  
```

In [5]:

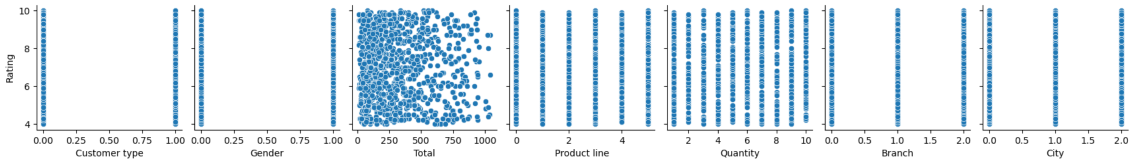
```
#  Preprocessing the dataset
label_encoder = LabelEncoder()
for column in data.columns:
    if data[column].dtype == 'object':
        data[column] = label_encoder.fit_transform(data[column])
```

In [6]:

```
# Defining the  features and target variable
features = ['Customer type', 'Gender', 'Total','Product line', 'Quantity','Branch','City
X = data[features]
y = data['Rating']
```

In [7]:

```
#  Doing the Exploratory Data Analysis (EDA)
sns.pairplot(data, x_vars=features, y_vars='Rating', kind='scatter')
plt.show()
```



In [8]:

```
# Splitting 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=42
```

In [9]:

```
# Building the  Linear Regression model
model = LinearRegression()
model.fit(X_train, y_train)
```

Out[9]:

```
LinearRegression()
```

In a Jupyter environment, please rerun this cell to show the HTML representation or trust the notebook.
On GitHub, the HTML representation is unable to render, please try loading this page with nbviewer.org.

In [10]:

```
# Making  predictions
y_pred = model.predict(X_test)
```

In [11]:

```
# Evaluating the  model performance
mse = mean_squared_error(y_test, y_pred)
print("Mean Squared Error:", mse)
```

```
Mean Squared Error: 3.1058213596610846
```

In [12]:

```
print("The Linear Regression model that we used suggests that the selected features (Bra
```

```
The Linear Regression model that we used suggests that the selected featu
res (Branch, City, Product line, Customer type, Gender, Total, Quantity)
have an impact on customer ratings. By analyzing these features,  we beli
ve the company can gain insights into how different branches, city locati
ons, and product lines influence customer satisfaction. This analysis can
guide decision-making to enhance the customer experience and optimize off
erings. Similar approaches can be replicated in other industries to under
stand the effects of various factors on customer ratings and preferences.
```

In [122]:

```
# Selecting the  features for clustering
features = ['Customer type', 'Gender', 'Total', 'Quantity', 'Rating']
X = data[features]
```

In [123]:

```
# Standardize features
scaler = StandardScaler()
X_scaled = scaler.fit_transform(X)
```

In [124]:

```
# Perform K-Means clustering
n_clusters = 3
kmeans = KMeans(n_clusters=n_clusters, random_state=42)
data['Cluster'] = kmeans.fit_predict(X_scaled)
```

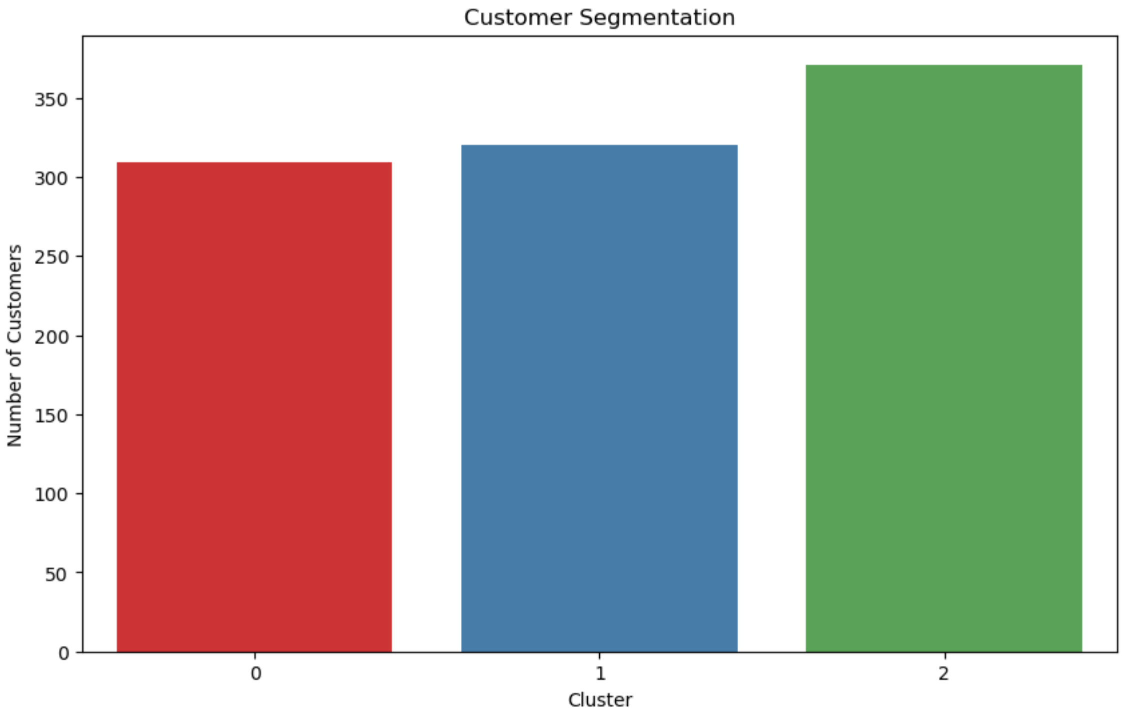
In [125]:

```
# Visualizing  customer segmentation
plt.figure(figsize=(10, 6))
sns.scatterplot(x='Total', y='Quantity', hue='Cluster', data=data, palette='Set1')
plt.title('Customer Segmentation based on Total Purchase and Quantity')
plt.xlabel('Total Purchase')
plt.ylabel('Quantity')
plt.show()

plt.figure(figsize=(10, 6))
sns.scatterplot(x='Total', y='Rating', hue='Cluster', data=data, palette='Set1')
plt.title('Customer Segmentation based on Total Purchase and Rating')
plt.xlabel('Total Purchase')
plt.ylabel('Rating')
plt.show()

# Analyzing clusters
cluster_means = data.groupby('Cluster').mean()
print(cluster_means)
```


```
         Invoice ID    Branch      City  Customer type    Gender  \
Cluster
0        496.844660  1.019417  0.990291       0.414239  0.430421
1        472.028125  1.021875  1.006250       0.000000  0.500000
2        525.407008  0.932615  1.024259       1.000000  0.555256

         Product line  Unit price  Quantity     Tax 5%       Total
Date  \
Cluster
0            2.498382   69.197864  8.611650  29.210325  613.416830  41.75
7282
1            2.453125   50.357250  3.865625   8.671409  182.099597  43.71
8750
2            2.412399   48.991024  4.345013   9.645627  202.558160  44.76
0108

               Time   Payment        cogs  gross margin percentage  \
Cluster
0        245.339806  1.022654  584.206505                 4.761905
1        251.400000  0.968750  173.428187                 4.761905
2        255.843666  1.010782  192.912534                 4.761905

         gross income    Rating
Cluster
0           29.210325  6.773139
1            8.671409  7.044375
2            9.645627  7.077089
```

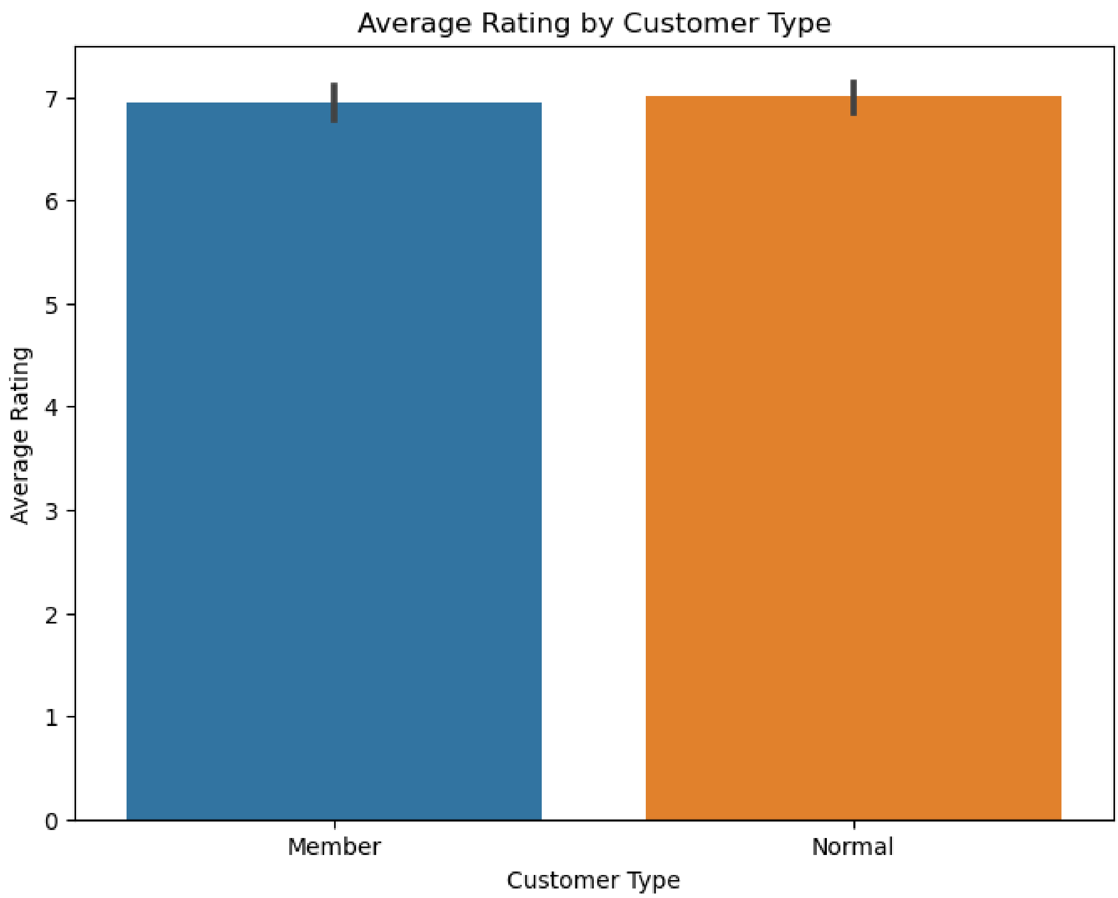
In [126]:

```
# Visualize customer segmentation with bar charts
plt.figure(figsize=(10, 6))
sns.countplot(x='Cluster', data=data, palette='Set1')
plt.title('Customer Segmentation')
plt.xlabel('Cluster')
plt.ylabel('Number of Customers')
plt.show()
```



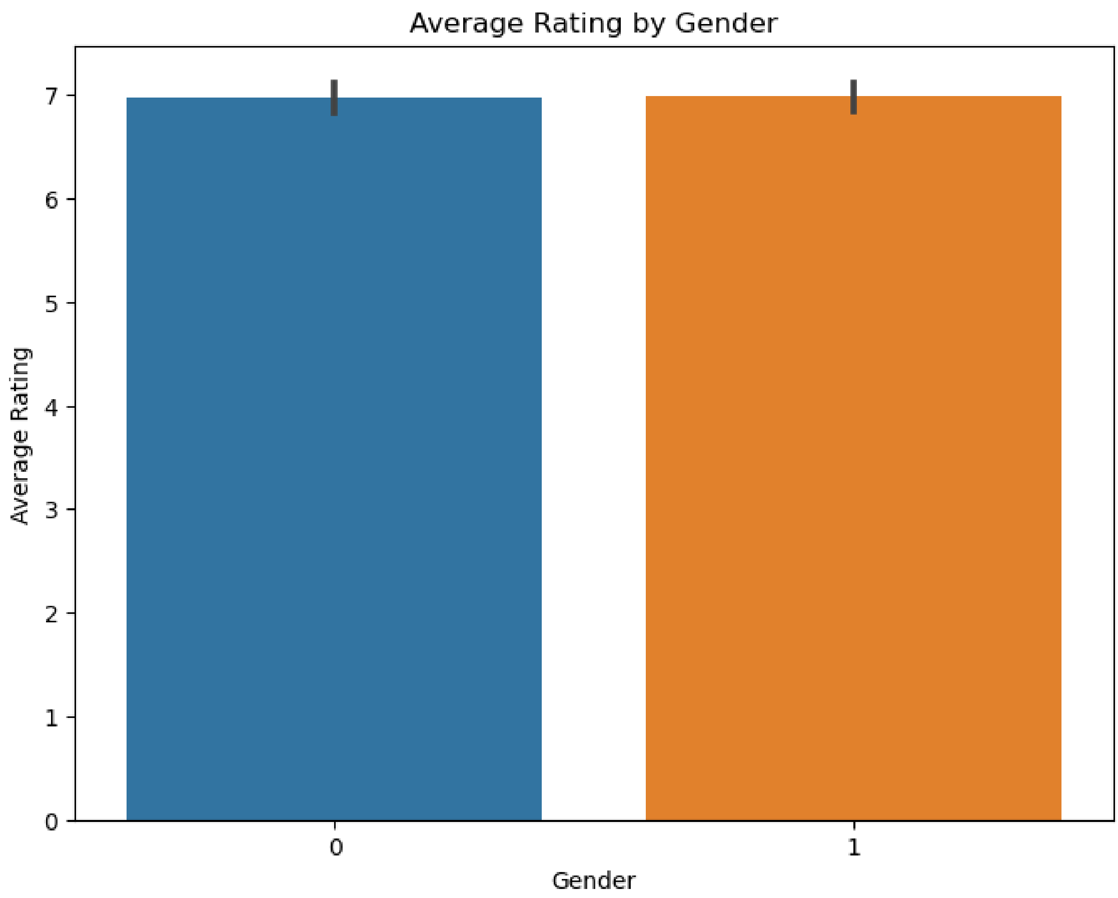
In [128]:

```
# Visualization 2: Average Rating by Customer Type
plt.figure(figsize=(8, 6))
sns.barplot(x='Customer type', y='Rating', data=data)
plt.title('Average Rating by Customer Type')
plt.xlabel('Customer Type')
plt.ylabel('Average Rating')
plt.xticks(ticks=[0, 1], labels=['Member', 'Normal'], rotation=0)
plt.show()
```



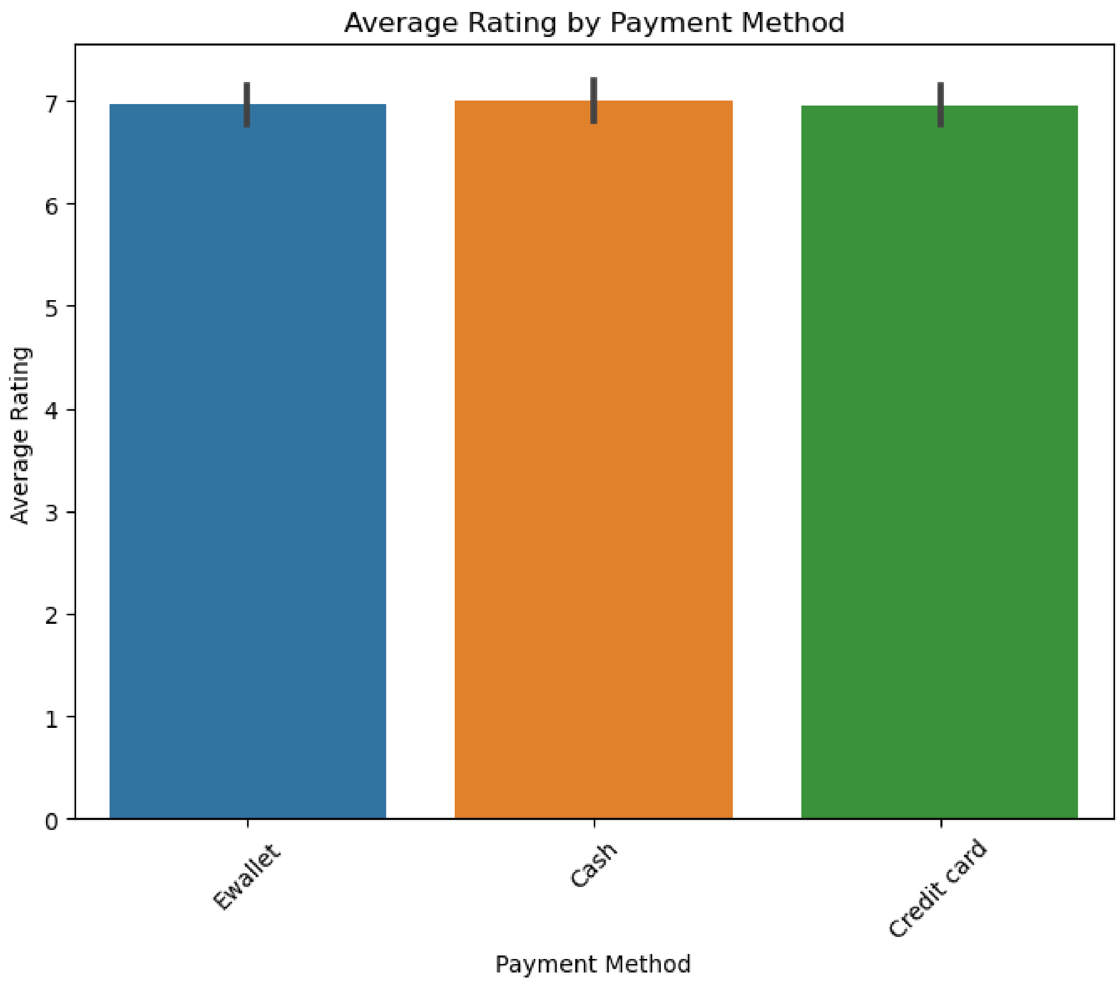
In [134]:

```
# Visualization 3: Average Rating by Gender
plt.figure(figsize=(8, 6))
sns.barplot(x='Gender', y='Rating', data=data)
plt.title('Average Rating by Gender')
plt.xlabel('Gender')
plt.ylabel('Average Rating')
plt.xticks(rotation=0)
plt.show()
```



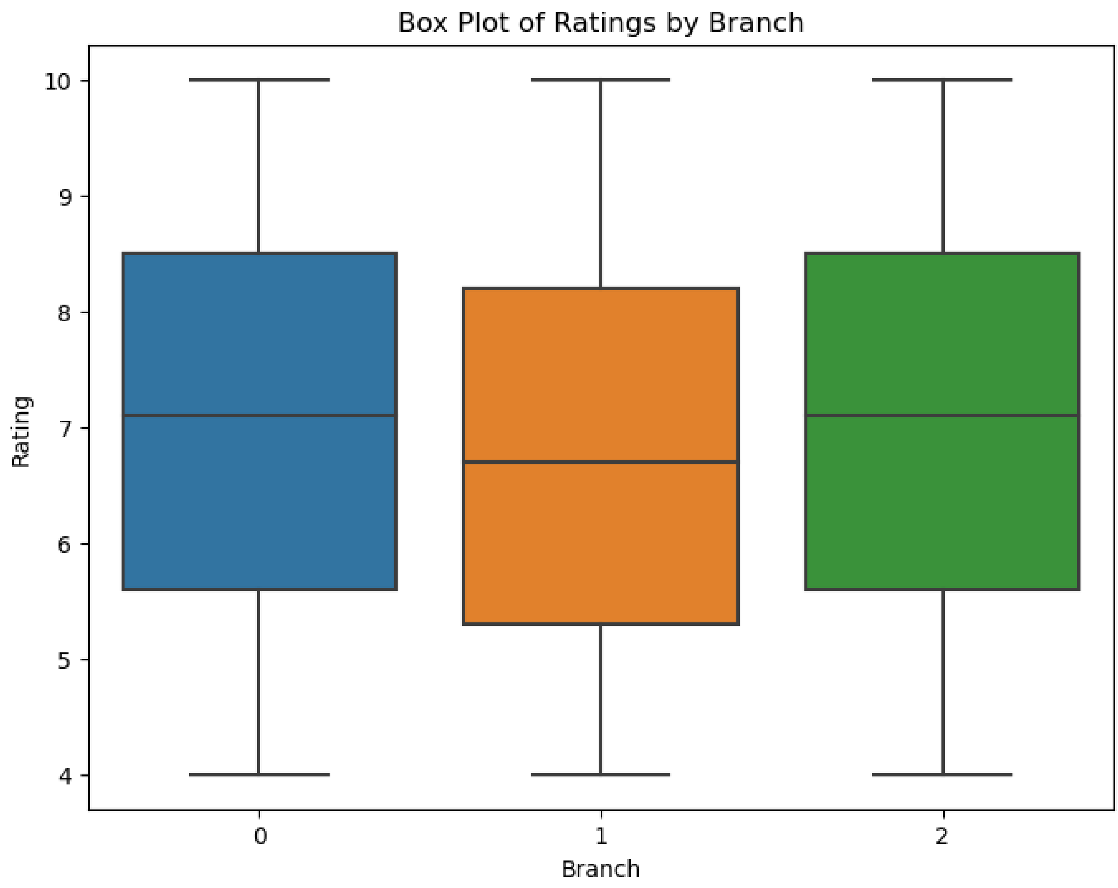
In [137]:

```
# Visualization 8: Rating by Payment Method
plt.figure(figsize=(8, 6))
sns.barplot(x='Payment', y='Rating', data=data)
plt.title('Average Rating by Payment Method')
plt.xlabel('Payment Method')
plt.ylabel('Average Rating')
plt.xticks(ticks=[0, 1, 2], labels=['Ewallet', 'Cash', 'Credit card'], rotation=45)
plt.show()
```



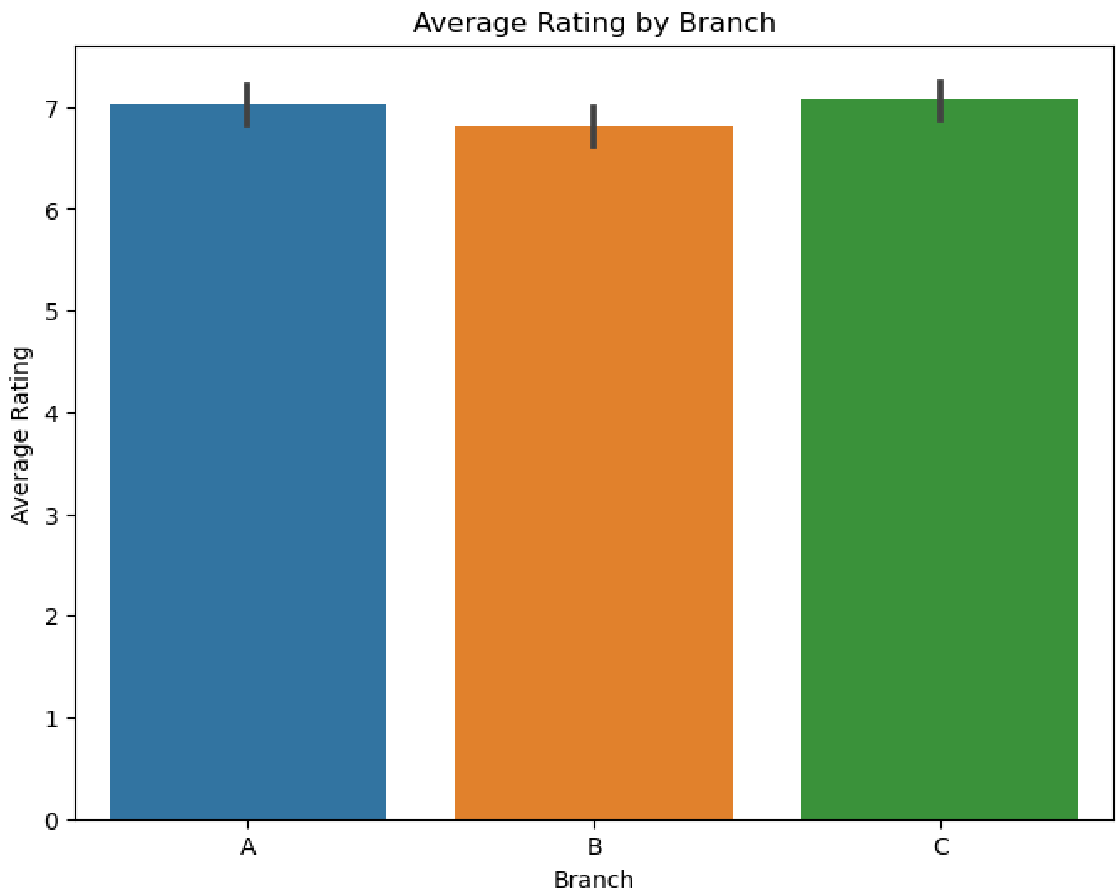
In [138]:

```
# Visualization 9: Box Plot of Ratings by Branch
plt.figure(figsize=(8, 6))
sns.boxplot(x='Branch', y='Rating', data=data)
plt.title('Box Plot of Ratings by Branch')
plt.xlabel('Branch')
plt.ylabel('Rating')
plt.show()
```



In [140]:

```
# Visualization 8: Rating by Branch
plt.figure(figsize=(8, 6))
sns.barplot(x='Branch', y='Rating', data=data)
plt.title('Average Rating by Branch')
plt.xlabel('Branch')
plt.ylabel('Average Rating')
plt.xticks(ticks=[0, 1, 2], labels=['A', 'B', 'C'], rotation=0)
plt.show()
```



In [ ]: