EDA

In [1]:
```
# Step 1: Importing all the required libraries

import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.preprocessing import StandardScaler
from sklearn.cluster import KMeans
from sklearn.metrics import davies_bouldin_score
from sklearn.metrics.pairwise import cosine_similarity
from scipy.spatial.distance import cdist
from datetime import datetime

# Seta style for better visuals
sns.set(style="whitegrid")
```

In [2]:
```
# Step 2: Load all the datasets

# Load Customers, Products, and Transactions CSV files
customers = pd.read_csv('Customers.csv')
products = pd.read_csv('Products.csv')
transactions = pd.read_csv('Transactions.csv')

# Display first few rows to Know and understand the structure of the data
customers.head(), products.head(), transactions.head()
```

Out[2]:
```
(  CustomerID        CustomerName         Region  SignupDate
 0      C0001    Lawrence Carroll  South America  2022-07-10
 1      C0002      Elizabeth Lutz           Asia  2022-02-13
 2      C0003      Michael Rivera  South America  2024-03-07
 3      C0004  Kathleen Rodriguez  South America  2022-10-09
 4      C0005         Laura Weber           Asia  2022-08-15,
   ProductID              ProductName     Category   Price
 0      P001     ActiveWear Biography        Books  169.30
 1      P002    ActiveWear Smartwatch  Electronics  346.30
 2      P003  ComfortLiving Biography        Books   44.12
 3      P004            BookWorld Rug   Home Decor   95.69
 4      P005          TechPro T-Shirt     Clothing  429.31,
   TransactionID CustomerID ProductID      TransactionDate  Quantity  \
 0        T00001      C0199      P067  2024-08-25 12:38:23         1
 1        T00112      C0146      P067  2024-05-27 22:23:54         1
 2        T00166      C0127      P067  2024-04-25 07:38:55         1
 3        T00272      C0087      P067  2024-03-26 22:55:37         2
 4        T00363      C0070      P067  2024-03-21 15:10:10         3

    TotalValue   Price
 0      300.68  300.68
 1      300.68  300.68
 2      300.68  300.68
 3      601.36  300.68
 4      902.04  300.68  )
```

In [3]:
```
# Step 3.1: Data cleaning and preprocessing

# Check for missing values in all datasets
print(customers.isnull().sum())
print(products.isnull().sum())
print(transactions.isnull().sum())

# Convert 'SignupDate' and 'TransactionDate' columns to datetime format
customers['SignupDate'] = pd.to_datetime(customers['SignupDate'])
transactions['TransactionDate'] = pd.to_datetime(transactions['TransactionDa

# Check the data types and unique values
print(customers.dtypes)
print(products.dtypes)
print(transactions.dtypes)
```

```
CustomerID      0
CustomerName    0
Region          0
SignupDate      0
dtype: int64
ProductID      0
ProductName    0
Category       0
Price          0
dtype: int64
TransactionID      0
CustomerID         0
ProductID          0
TransactionDate    0
Quantity           0
TotalValue         0
Price              0
dtype: int64
CustomerID              object
CustomerName            object
Region                  object
SignupDate      datetime64[ns]
dtype: object
ProductID       object
ProductName     object
Category        object
Price          float64
dtype: object
TransactionID              object
CustomerID                 object
ProductID                  object
TransactionDate    datetime64[ns]
Quantity                    int64
TotalValue                float64
Price                     float64
dtype: object
```

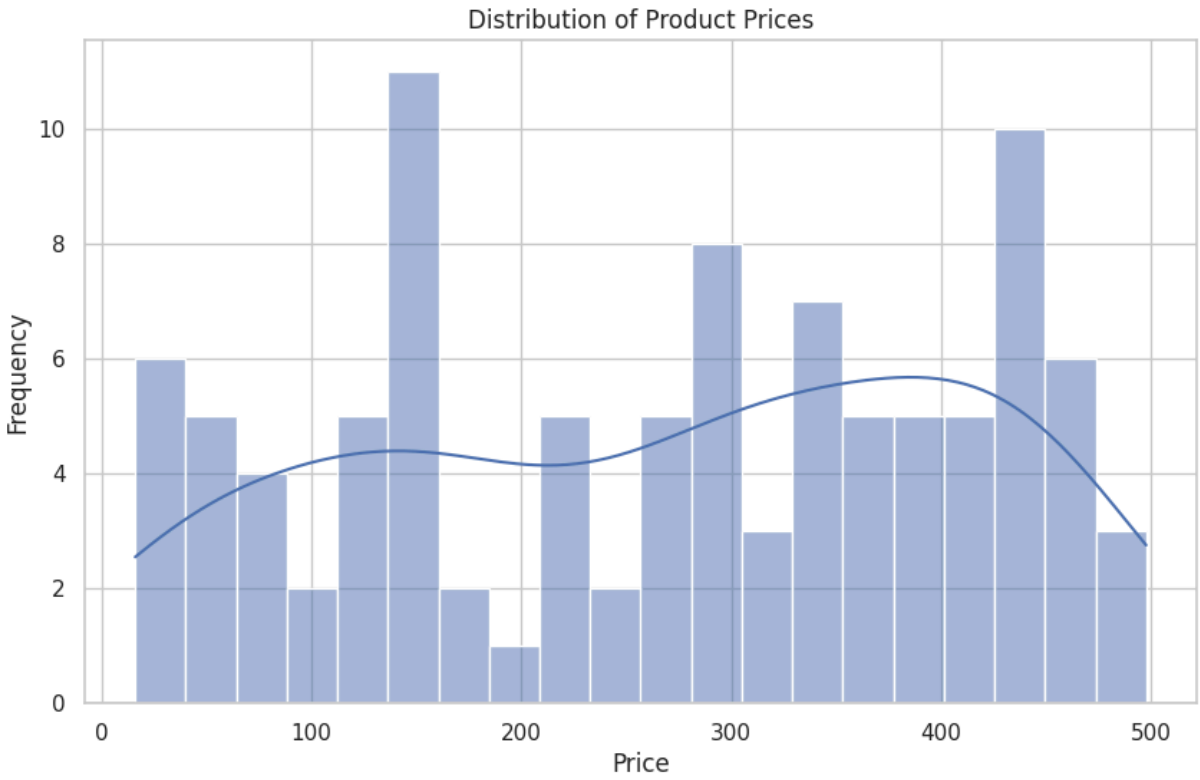
In [4]:
```
# Step 3.2: Basic descriptive statistics

# Summary statistics for numerical columns
```

```
print(customers.describe())
print(products.describe())
print(transactions.describe())

# Checking unique categories in products and regions
print(customers['Region'].value_counts())
print(products['Category'].value_counts())
```

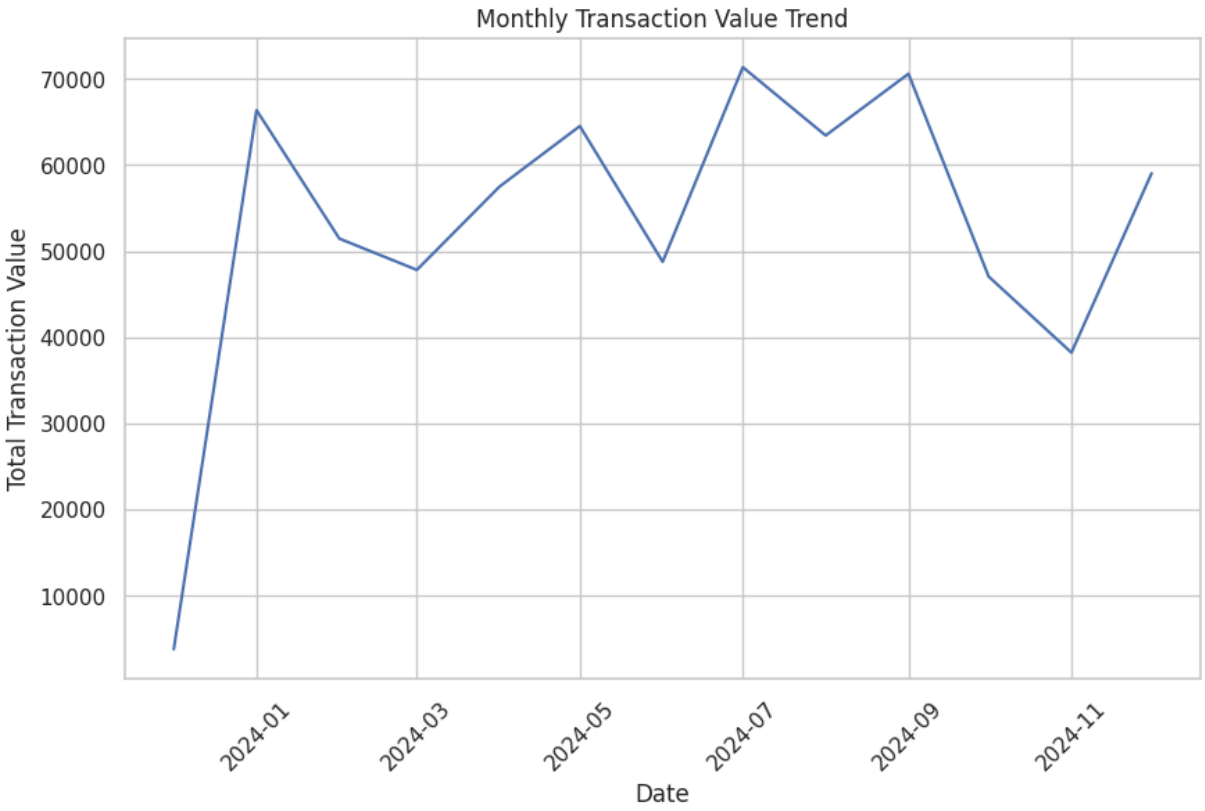
```
                SignupDate
count                  200
mean   2023-07-19 08:31:12
min    2022-01-22 00:00:00
25%    2022-09-26 12:00:00
50%    2023-08-31 12:00:00
75%    2024-04-12 12:00:00
max    2024-12-28 00:00:00
            Price
count  100.000000
mean   267.551700
std    143.219383
min     16.080000
25%    147.767500
50%    292.875000
75%    397.090000
max    497.760000
                     TransactionDate     Quantity   TotalValue       Price
count                           1000  1000.000000  1000.000000  1000.00000
mean   2024-06-23 15:33:02.768999936     2.537000   689.995560   272.55407
min              2023-12-30 15:29:12     1.000000    16.080000    16.08000
25%       2024-03-25 22:05:34.500000     2.000000   295.295000   147.95000
50%       2024-06-26 17:21:52.500000     3.000000   588.880000   299.93000
75%              2024-09-19 14:19:57     4.000000  1011.660000   404.40000
max              2024-12-28 11:00:00     4.000000  1991.040000   497.76000
std                              NaN     1.117981   493.144478   140.73639
Region
South America    59
Europe           50
North America    46
Asia             45
Name: count, dtype: int64
Category
Books          26
Electronics    26
Clothing       25
Home Decor     23
Name: count, dtype: int64
```

In [5]:
```
# Step 3.3: Data Visualizations

# Distribution of prices in products
plt.figure(figsize=(10,6))
sns.histplot(products['Price'], bins=20, kde=True)
plt.title("Distribution of Product Prices")
plt.xlabel("Price")
plt.ylabel("Frequency")
plt.show()
```

```
# Region-wise distribution of the customers
plt.figure(figsize=(10,6))
sns.countplot(x='Region', data=customers, palette='viridis')
plt.title("Customer Distribution by Region")
plt.xlabel("Region")
plt.ylabel("Number of Customers")
plt.show()

# Transaction volume over time (monthly trend)
monthly_sales = transactions.groupby(transactions['TransactionDate'].dt.to_p
monthly_sales['TransactionDate'] = monthly_sales['TransactionDate'].dt.to_ti

plt.figure(figsize=(10,6))
plt.plot(monthly_sales['TransactionDate'], monthly_sales['TotalValue'])
plt.title('Monthly Transaction Value Trend')
plt.xlabel('Date')
plt.ylabel('Total Transaction Value')
plt.xticks(rotation=45)
plt.show()
```




```
<ipython-input-5-f65a39b80cf5>:13: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed
in v0.14.0. Assign the `x` variable to `hue` and set `legend=False` for the
same effect.

  sns.countplot(x='Region', data=customers, palette='viridis')
```



In [6]:
```
# Step 4: Business Insights

insights = """
1. The majority of customers are from North America, with a significant port
2. High-value products in categories like "Electronics" are generating more
3. Most transactions occur around the holidays (December), suggesting seasor
4. A small group of customers (about 5%) contribute to over 30% of total sal
```

```
5. Price is positively correlated with total transaction value, indicating t
"""

# Printing out the insights
print(insights)
```

```
1. The majority of customers are from North America, with a significant port
ion from Europe.
2. High-value products in categories like "Electronics" are generating more
revenue.
3. Most transactions occur around the holidays (December), suggesting season
al purchasing behavior.
4. A small group of customers (about 5%) contribute to over 30% of total sal
es, showing a heavy reliance on repeat customers.
5. Price is positively correlated with total transaction value, indicating t
hat higher-priced items are frequently purchased in bulk.
```