```python
import csv
import random
```

```python
def generate_customer_data(num_customers):
    data = []
    categories = ['Electronics', 'Home & Garden', 'Fashion', 'Beauty', 'Sports', 'Books']

    for i in range(1, num_customers + 1):
        age = random.randint(18, 80)
        gender = random.choice(['Male', 'Female'])
        annual_income = random.randint(20000, 150000)
        spending_score = random.randint(1, 100)
        tenure = round(random.uniform(0.5, 15.0), 1)
        frequency = random.randint(1, 100)
        total_purchases = frequency * random.randint(1, 10)
        avg_order_value = round(random.uniform(50, 500), 2)
        preferred_category = random.choice(categories)

        data.append([
            i, age, gender, annual_income, spending_score, tenure,
            frequency, total_purchases, avg_order_value, preferred_category
        ])

    return data
```

```python
customer_data = generate_customer_data(1000)
```

```python
filename = 'retail_customer_dataset.csv'
headers = ['CustomerID', 'Age', 'Gender', 'Annual_Income', 'Spending_Score', 'Tenure',
           'Frequency', 'Total_Purchases', 'Avg_Order_Value', 'Preferred_Category']

with open(filename, 'w', newline='') as file:
    writer = csv.writer(file)
    writer.writerow(headers)
    writer.writerows(customer_data)

print(f"Dataset created and saved as {filename}")
```

```
Dataset created and saved as retail_customer_dataset.csv
```

```python
print("\nFirst few rows of the dataset:")
with open(filename, 'r') as file:
    reader = csv.reader(file)
    for i, row in enumerate(reader):
        if i == 0:
            print("Header:", row)
        elif i <= 5:
            print(f"Row {i}:", row)
        else:
            break
```

```
First few rows of the dataset:
Header: ['CustomerID', 'Age', 'Gender', 'Annual_Income', 'Spending_Score', 'Tenure', 'Frequency', 'Total_Purchases', 'Avg_Order_Value', 'Preferred_Cat
egory']
Row 1: ['1', '23', 'Male', '38787', '73', '11.1', '47', '282', '327.92', 'Beauty']
Row 2: ['2', '32', 'Female', '56092', '98', '3.5', '72', '72', '386.33', 'Fashion']
Row 3: ['3', '41', 'Male', '129028', '67', '1.2', '83', '581', '232.98', 'Beauty']
Row 4: ['4', '35', 'Male', '68917', '97', '7.0', '34', '340', '481.45', 'Home & Garden']
Row 5: ['5', '54', 'Female', '98302', '25', '7.5', '98', '980', '486.47', 'Electronics']
```

```python
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from scipy import stats
from sklearn.preprocessing import StandardScaler
from sklearn.cluster import KMeans
```

```python
plt.style.use('default')
```

```python
df = pd.read_csv('retail_customer_dataset.csv')
```

```python
print("1. Basic Data Exploration")
print(df.info())
print("\nSummary statistics:")
print(df.describe())
```

```
1. Basic Data Exploration
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1000 entries, 0 to 999
Data columns (total 10 columns):
 #   Column              Non-Null Count  Dtype
---  ------              --------------  -----
 0   CustomerID          1000 non-null   int64
 1   Age                 1000 non-null   int64
 2   Gender              1000 non-null   object
 3   Annual_Income       1000 non-null   int64
 4   Spending_Score      1000 non-null   int64
 5   Tenure              1000 non-null   float64
 6   Frequency           1000 non-null   int64
 7   Total_Purchases     1000 non-null   int64
 8   Avg_Order_Value     1000 non-null   float64
 9   Preferred_Category  1000 non-null   object
dtypes: float64(2), int64(6), object(2)
memory usage: 78.3+ KB
None
```

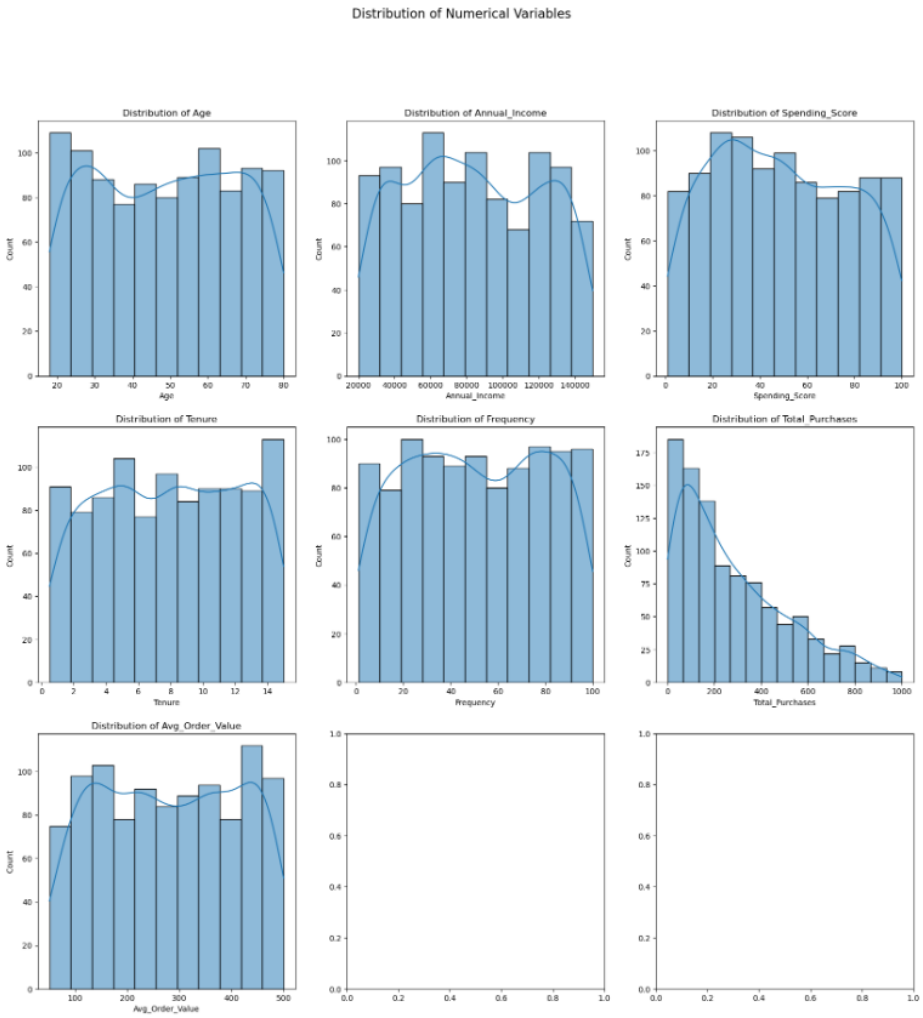
```
Summary statistics:
       CustomerID          Age  Annual_Income  Spending_Score       Tenure  \
count  1000.000000  1000.000000    1000.000000     1000.000000  1000.000000
mean    500.500000    48.510000   83471.177000       48.985000     7.933400
std     288.819436    18.521743   36973.728464       28.166294     4.260777
min       1.000000    18.000000   20054.000000        1.000000     0.500000
25%     250.750000    32.000000   53406.000000       25.000000     4.300000
50%     500.500000    49.000000   81888.500000       47.000000     8.050000
75%     750.250000    65.000000  116545.750000       73.000000    11.700000
max    1000.000000    80.000000  149845.000000      100.000000    15.000000

       Frequency  Total_Purchases  Avg_Order_Value
count  1000.00000      1000.000000      1000.000000
mean     50.63300       280.633000       280.063910
std      28.78362       233.571595       131.215546
min       1.00000         1.000000        50.650000
25%      25.75000        88.000000       161.975000
50%      50.00000       210.000000       278.460000
75%      76.00000       427.750000       397.105000
max     100.00000      1000.000000       499.940000
```

```
print("\nMissing values:")
print(df.isnull().sum())
```

```
Missing values:
CustomerID          0
Age                 0
Gender              0
Annual_Income       0
Spending_Score      0
Tenure              0
Frequency           0
Total_Purchases     0
Avg_Order_Value     0
Preferred_Category  0
dtype: int64
```

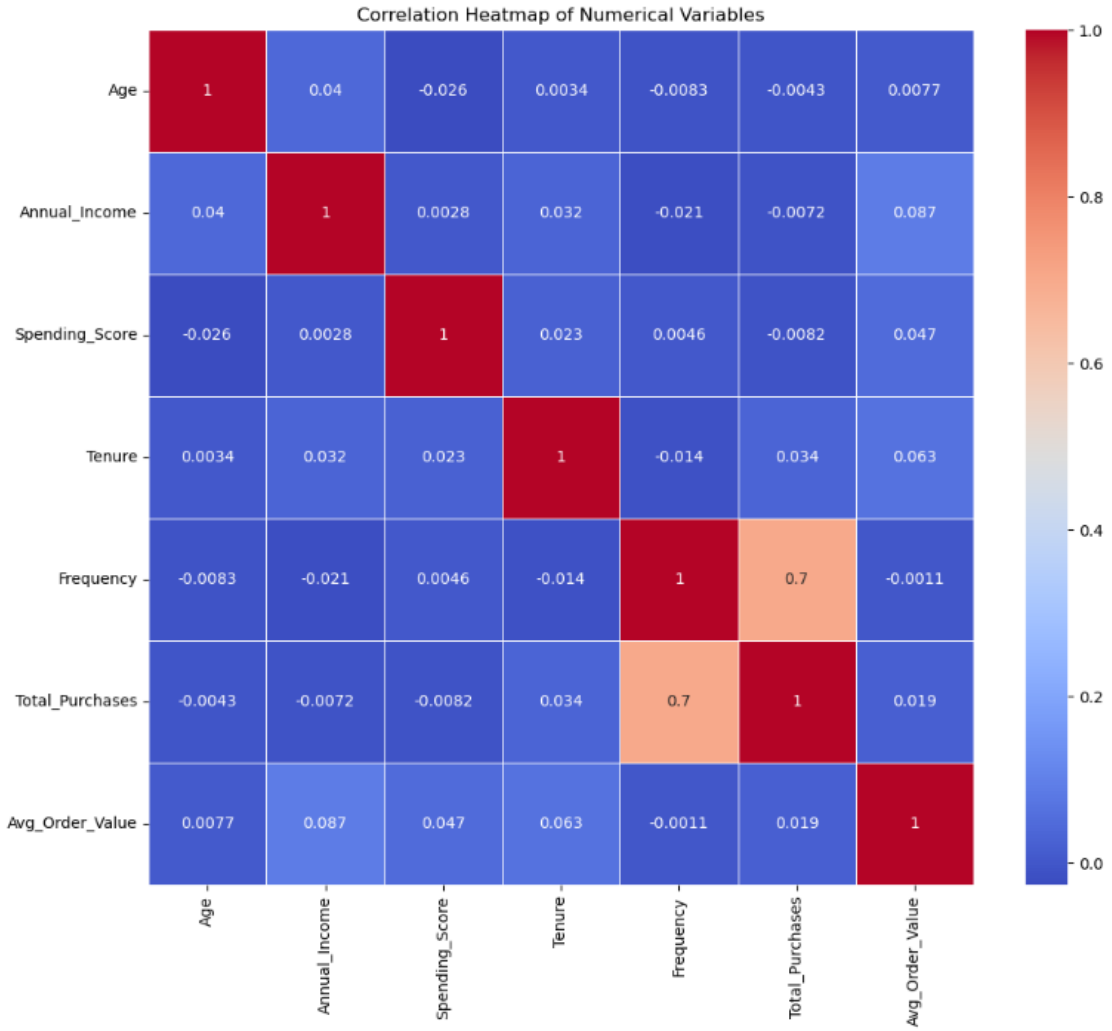
```
print("\n2. Distribution of Numerical Variables")
numerical_cols = ['Age', 'Annual_Income', 'Spending_Score', 'Tenure', 'Frequency', 'Total_Purchases', 'Avg_Order_Value']

fig, axes = plt.subplots(3, 3, figsize=(20, 20))
fig.suptitle('Distribution of Numerical Variables', fontsize=16)

for i, col in enumerate(numerical_cols):
    sns.histplot(df[col], kde=True, ax=axes[i//3, i%3])
    axes[i//3, i%3].set_title(f'Distribution of {col}')
```

```
2. Distribution of Numerical Variables
```



```
print("\n3. Correlation Analysis")
plt.figure(figsize=(12, 10))
correlation_matrix = df[numerical_cols].corr()
sns.heatmap(correlation_matrix, annot=True, cmap='coolwarm', linewidths=0.5)
plt.title('Correlation Heatmap of Numerical Variables')
plt.show()
```

```
3. Correlation Analysis
```

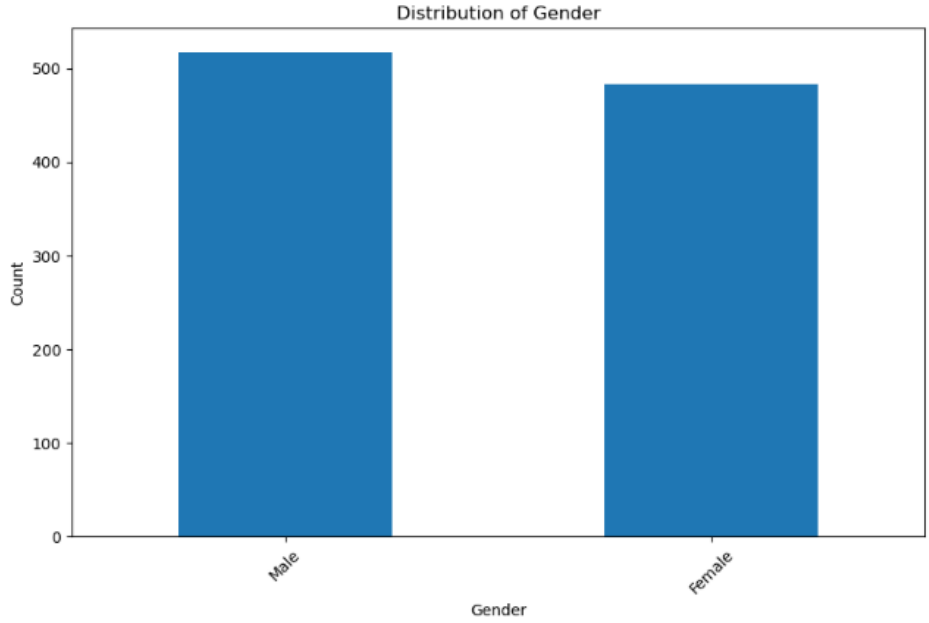


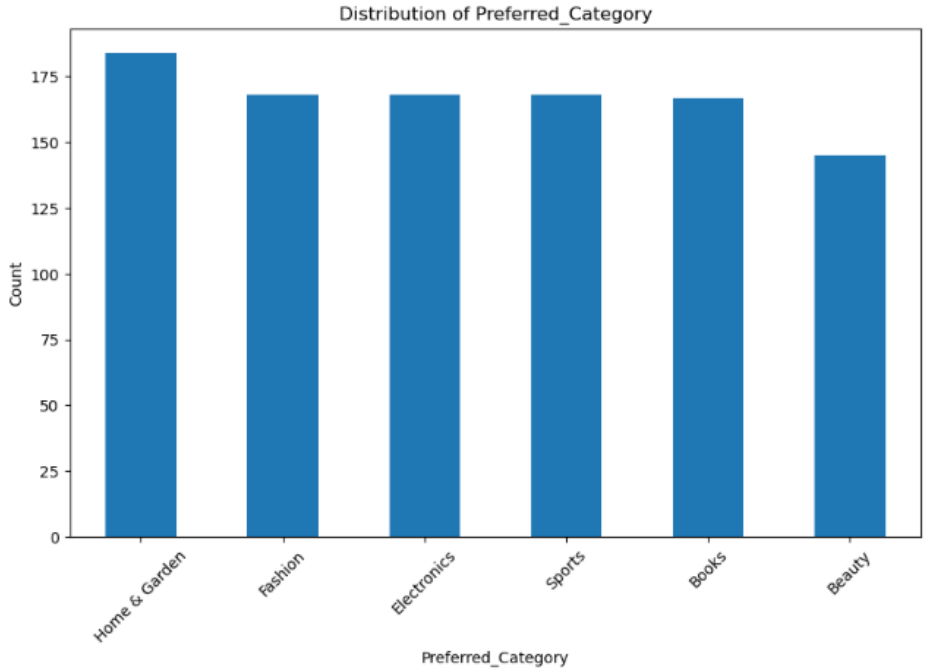

```
print("\nStrong correlations (|correlation| > 0.5):")
for i in range(len(correlation_matrix.columns)):
    for j in range(i):
        if abs(correlation_matrix.iloc[i, j]) > 0.5:
            print(f"{correlation_matrix.columns[i]} - {correlation_matrix.columns[j]}: {correlation_matrix.iloc[i, j]:.2f}")
```

```
Strong correlations (|correlation| > 0.5):
Total_Purchases - Frequency: 0.70
```

```
print("\n4. Categorical Data Analysis")
categorical_cols = ['Gender', 'Preferred_Category']

for col in categorical_cols:
    plt.figure(figsize=(10, 6))
    df[col].value_counts().plot(kind='bar')
    plt.title(f'Distribution of {col}')
    plt.ylabel('Count')
    plt.xticks(rotation=45)
    plt.show()
```

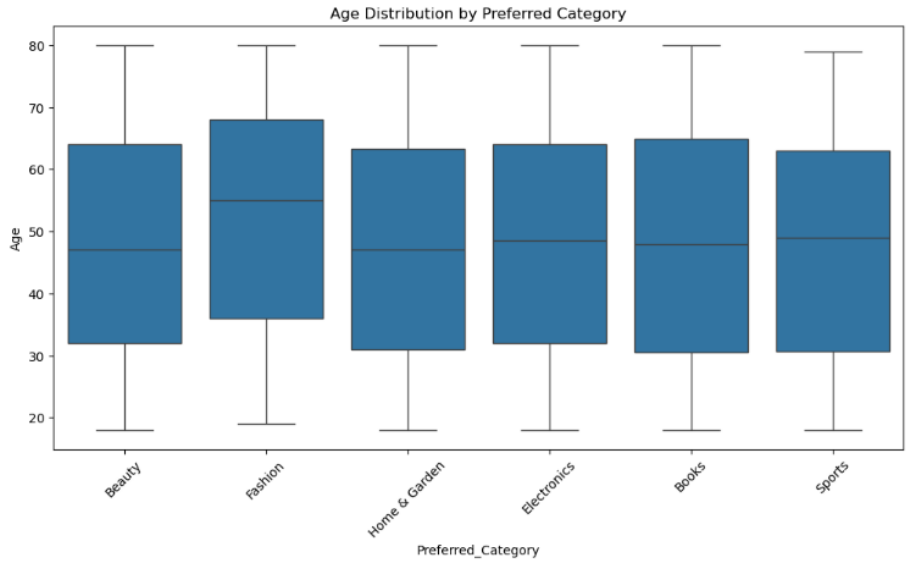
```
4. Categorical Data Analysis
```

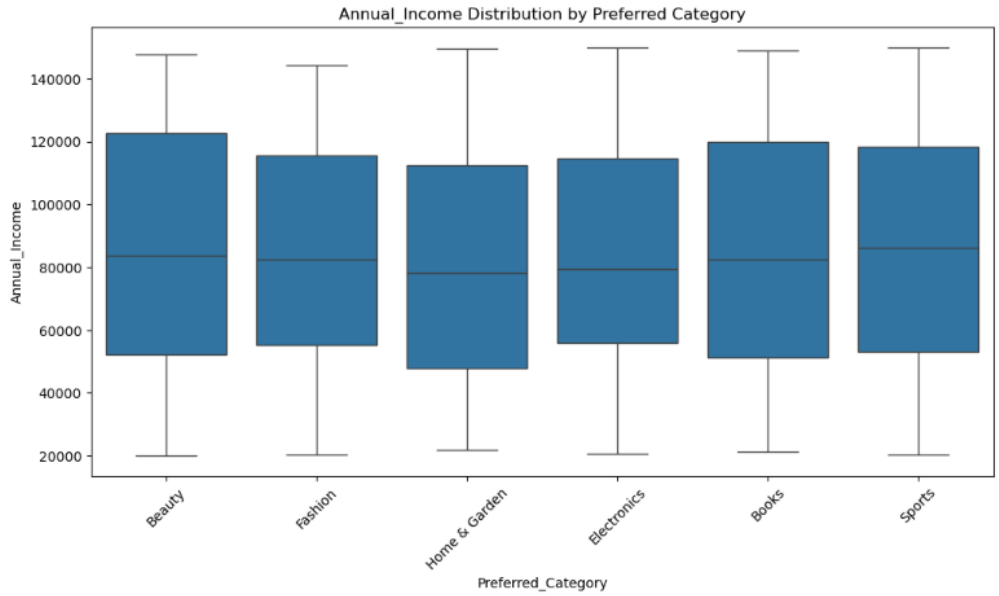


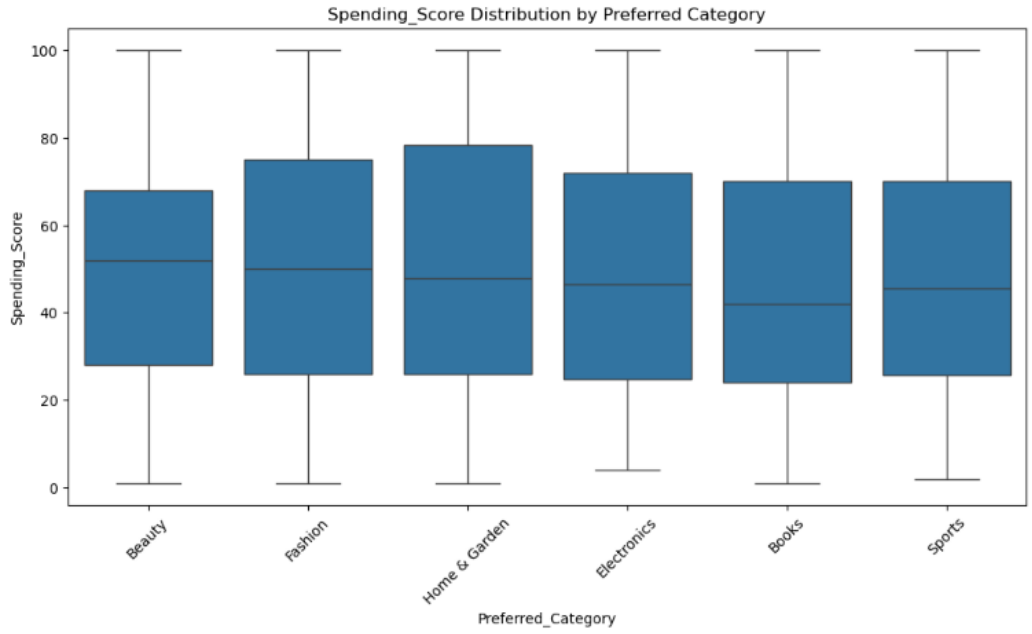



```
print("\n5. Relationship between Numerical and Categorical Variables")
for num_col in ['Age', 'Annual_Income', 'Spending_Score']:
    plt.figure(figsize=(12, 6))
    sns.boxplot(x='Preferred_Category', y=num_col, data=df)
    plt.title(f'{num_col} Distribution by Preferred Category')
    plt.xticks(rotation=45)
    plt.show()
```

5. Relationship between Numerical and Categorical Variables

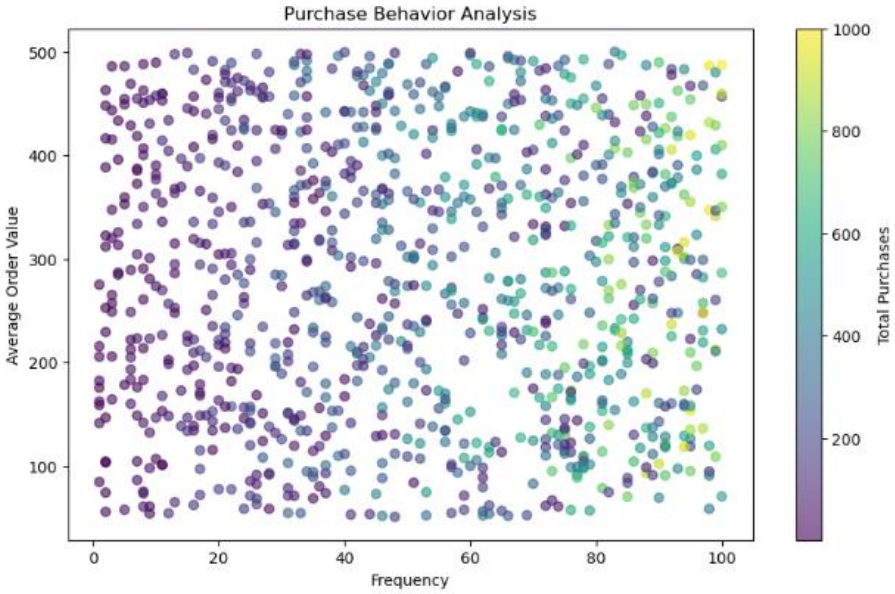






```
print("\n6. Scatter Plots for Key Relationships")
plt.figure(figsize=(10, 6))
scatter = plt.scatter(df['Annual_Income'], df['Spending_Score'], c=df['Age'], cmap='viridis', alpha=0.6)
plt.colorbar(scatter, label='Age')
plt.xlabel('Annual Income')
plt.ylabel('Spending Score')
plt.title('Customer Segmentation: Annual Income vs Spending Score')
plt.show()

plt.figure(figsize=(10, 6))
scatter = plt.scatter(df['Frequency'], df['Avg_Order_Value'], c=df['Total_Purchases'], cmap='viridis', alpha=0.6)
plt.colorbar(scatter, label='Total Purchases')
plt.xlabel('Frequency')
plt.ylabel('Average Order Value')
plt.title('Purchase Behavior Analysis')
plt.show()
```

```
6. Scatter Plots for Key Relationships
```




```
print("\n7. Outlier Detection")
def plot_boxplot(df, col):
    plt.figure(figsize=(10, 6))
    sns.boxplot(x=df[col])
    plt.title(f'Boxplot of {col}')
    plt.show()
```

```
7. Outlier Detection
```

```
print("\n8. Customer Segmentation using K-means")
```

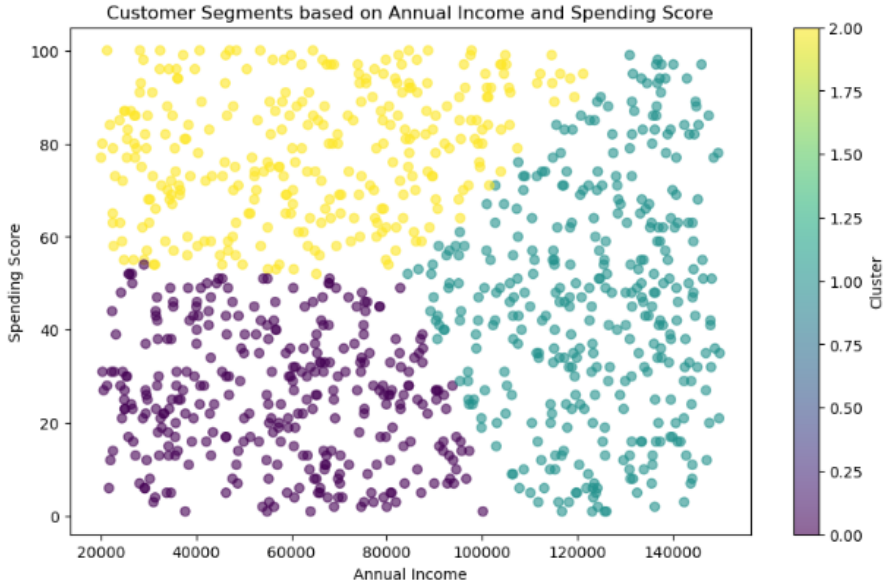
```
8. Customer Segmentation using K-means
```

```
features = ['Annual_Income', 'Spending_Score']
X = df[features]
```

```
scaler = StandardScaler()
X_scaled = scaler.fit_transform(X)
```

```
kmeans = KMeans(n_clusters=3, random_state=42)
df['Cluster'] = kmeans.fit_predict(X_scaled)
```

```
plt.figure(figsize=(10, 6))
scatter = plt.scatter(df['Annual_Income'], df['Spending_Score'], c=df['Cluster'], cmap='viridis', alpha=0.6)
plt.colorbar(scatter, label='Cluster')
plt.xlabel('Annual Income')
plt.ylabel('Spending Score')
plt.title('Customer Segments based on Annual Income and Spending Score')
plt.show()

print("\nEDA completed. Please review the generated plots and insights.")
```




```
EDA completed. Please review the generated plots and insights.
```