```
import matplotlib.pyplot as plt
import seaborn as sns
import pandas as pd
import numpy as np


url = "https://d2beiqkhq929f0.cloudfront.net/public_assets/assets/000/001/293/original/walmart_data.csv?1641285094"
data = pd.read_csv(url)
data.head(10)
```

	User_ID	Product_ID	Gender	Age	Occupation	City_Category	Stay_In_Current_City_Years	Marital_Status	Product_Category	Purcha
0	1000001	P00069042	F	0-17	10	A	2	0	3	83
1	1000001	P00248942	F	0-17	10	A	2	0	1	152
2	1000001	P00087842	F	0-17	10	A	2	0	12	14
3	1000001	P00085442	F	0-17	10	A	2	0	12	10
4	1000002	P00285442	M	55+	16	C	4+	0	8	79
5	1000003	P00193542	M	26-35	15	A	3	0	1	152
6	1000004	P00184942	M	46-50	7	B	2	1	1	192

```
print("Dataset Shape (Rows, Columns):", data.shape)
```

```
Dataset Shape (Rows, Columns): (550068, 10)
```

```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 550068 entries, 0 to 550067
Data columns (total 10 columns):
 #   Column                      Non-Null Count   Dtype
---  ------                      --------------   -----
 0   User_ID                     550068 non-null  int64
 1   Product_ID                  550068 non-null  object
 2   Gender                      550068 non-null  object
 3   Age                         550068 non-null  object
 4   Occupation                  550068 non-null  int64
 5   City_Category               550068 non-null  object
 6   Stay_In_Current_City_Years  550068 non-null  object
 7   Marital_Status              550068 non-null  int64
 8   Product_Category            550068 non-null  int64
 9   Purchase                    550068 non-null  int64
dtypes: int64(5), object(5)
memory usage: 42.0+ MB
```

```
data.describe()
```

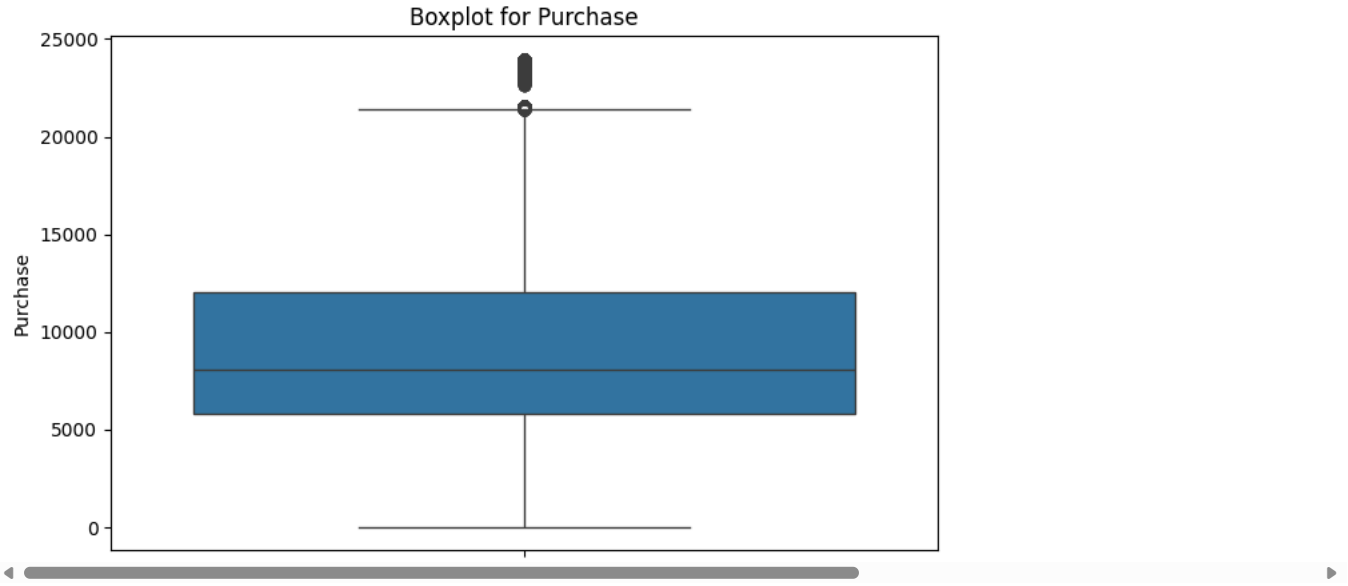
	User_ID	Occupation	Marital_Status	Product_Category	Purchase
count	5.500680e+05	550068.000000	550068.000000	550068.000000	550068.000000
mean	1.003029e+06	8.076707	0.409653	5.404270	9263.968713
std	1.727592e+03	6.522660	0.491770	3.936211	5023.065394
min	1.000001e+06	0.000000	0.000000	1.000000	12.000000
25%	1.001516e+06	2.000000	0.000000	1.000000	5823.000000
50%	1.003077e+06	7.000000	0.000000	5.000000	8047.000000
75%	1.004478e+06	14.000000	1.000000	8.000000	12054.000000
max	1.006040e+06	20.000000	1.000000	20.000000	23961.000000

```
print("Missing Values in Each Column:")
print (data.isnull().sum())
```

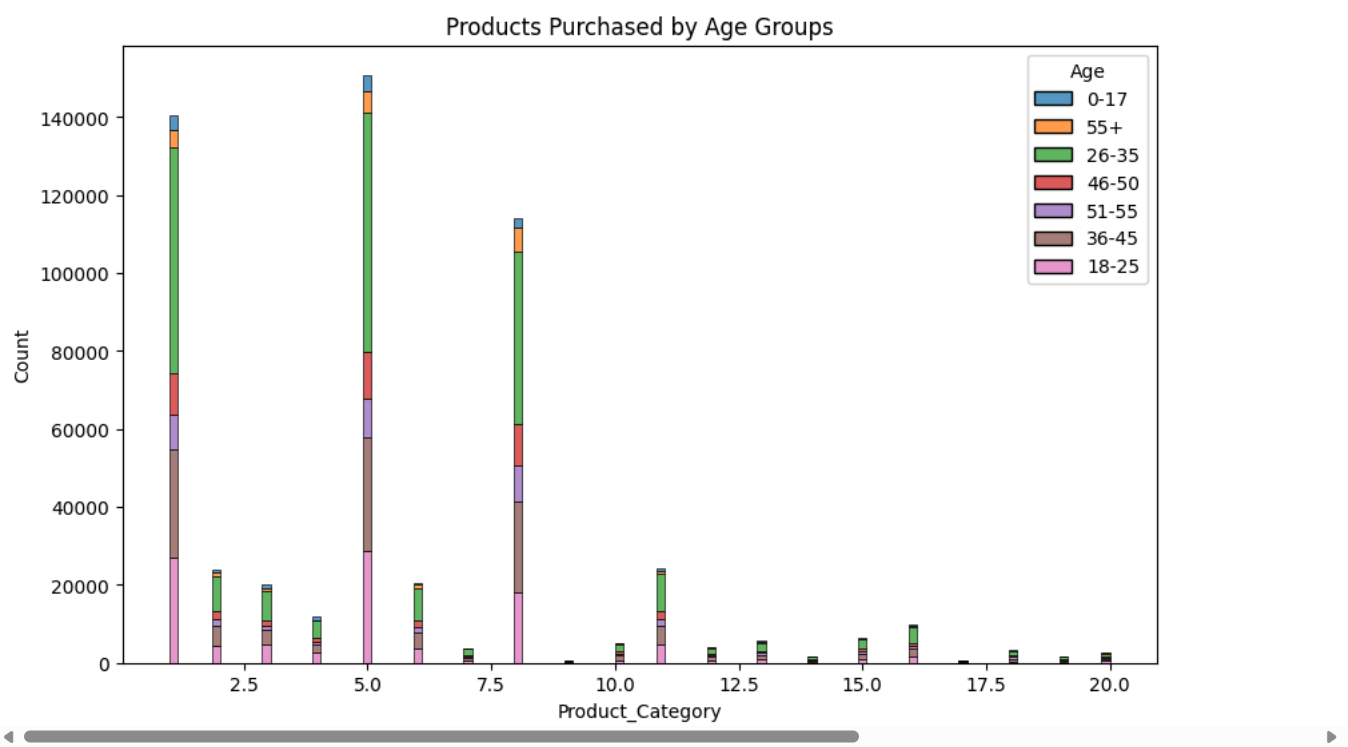
```
Missing Values in Each Column:
User_ID                       0
Product_ID                    0
Gender                        0
Age                           0
Occupation                    0
City_Category                 0
```

```
Stay_In_Current_City_Years    0
Marital_Status                0
Product_Category              0
Purchase                      0
dtype: int64
```

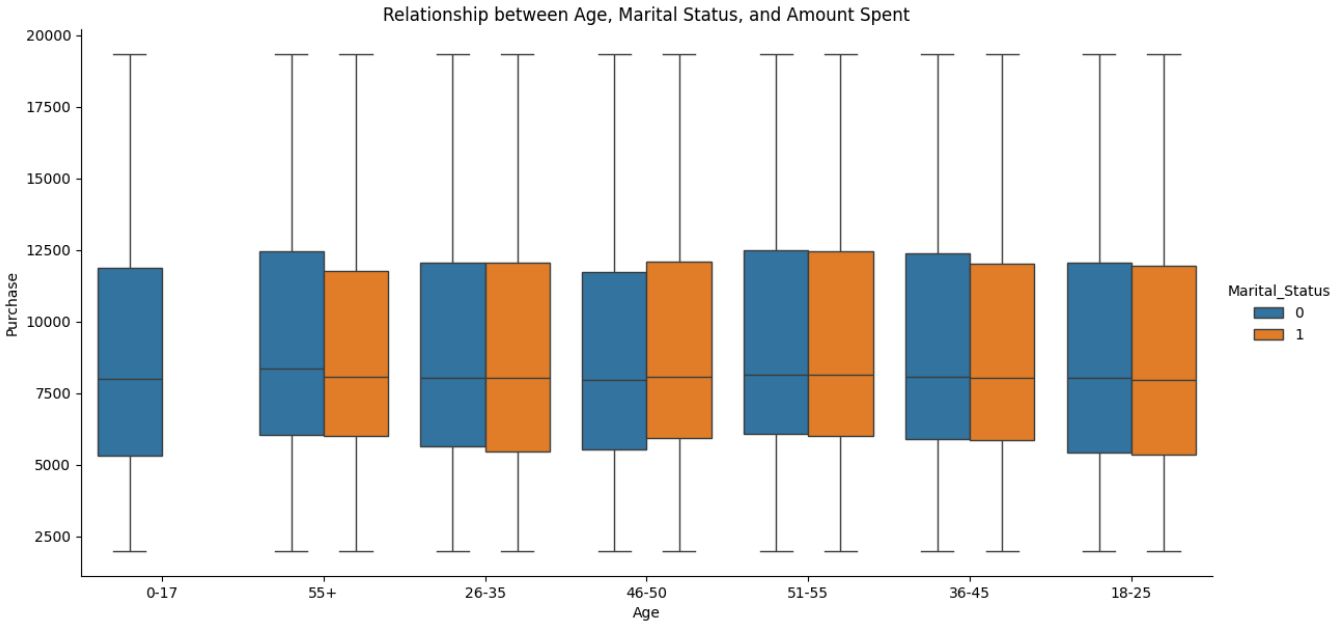
```
# (a) Outliers detection using boxplots for continuous variables
continuous_columns = ['Purchase']
for col in continuous_columns:
    plt.figure(figsize=(8, 5))
    sns.boxplot(data[col])
    plt.title(f"Boxplot for {col}")
    plt.show()

# (b) Clipping data between 5th and 95th percentile
for col in continuous_columns:
    lower_limit = np.percentile(data[col], 5)
    upper_limit = np.percentile(data[col], 95)
    data[col] = np.clip(data[col], lower_limit, upper_limit)
```



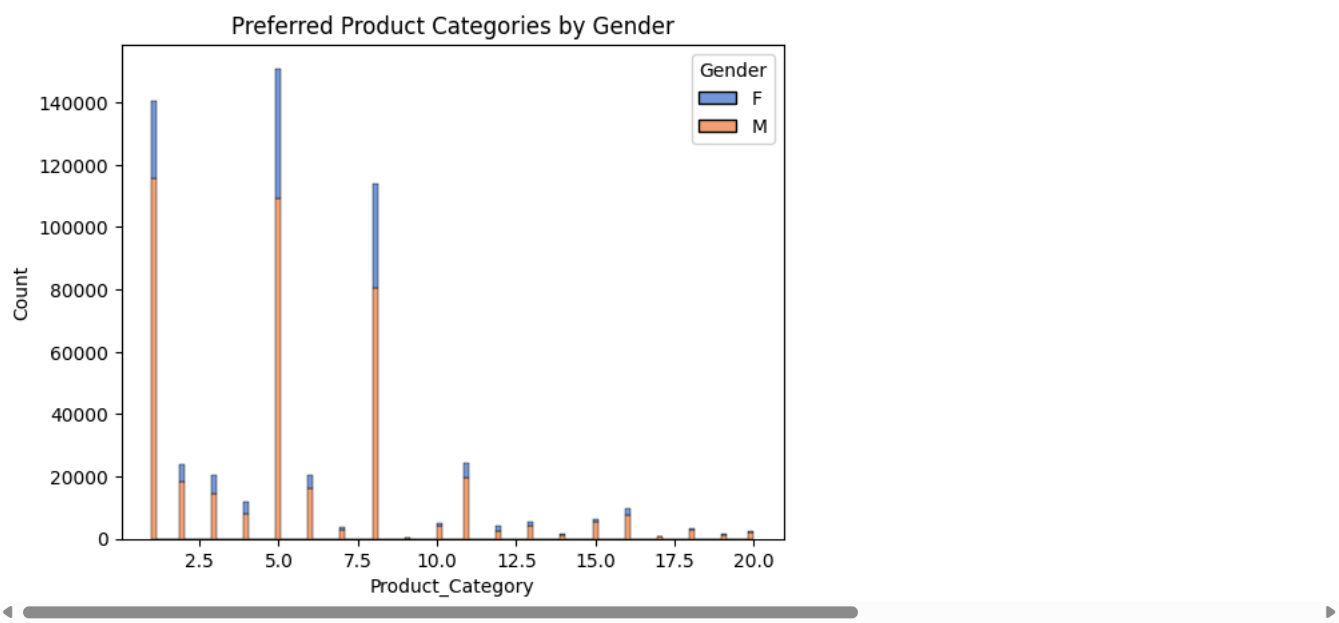

```
# (a) Products purchased by different age groups
plt.figure(figsize=(10, 6))
sns.histplot(data=data, x='Product_Category', hue='Age', multiple='stack')
plt.title("Products Purchased by Age Groups")
plt.show()
```



```
# (b) Relationship between age, marital status, and amount spent
sns.catplot(data=data, x='Age', y='Purchase', hue='Marital_Status', kind='box', height=6, aspect=2)
plt.title("Relationship between Age, Marital Status, and Amount Spent")
plt.show()
```




```
# (c) Preferred product categories for different genders
sns.histplot(data=data, x='Product_Category', hue='Gender', multiple='stack', palette='muted')
plt.title("Preferred Product Categories by Gender")
plt.show()
```




```
def bootstrap_confidence_interval(data, n_samples, sample_size, column):
    means = []
    for _ in range(n_samples):
        sample = data[column].sample(n=sample_size, replace=True)
        means.append(sample.mean())
    ci_lower = np.percentile(means, 2.5)
    ci_upper = np.percentile(means, 97.5)
    return ci_lower, ci_upper, means

# Confidence intervals for genders
genders = data['Gender'].unique()
for gender in genders:
    subset = data[data['Gender'] == gender]
    print(f"\nConfidence Interval for {gender}:")
```

```
    for size in [len(subset), 300, 3000, 30000]:
        ci = bootstrap_confidence_interval(subset, 1000, size, 'Purchase')
        print(f"Sample Size {size}: {ci[:2]}")
```

```
Confidence Interval for F:
Sample Size 135809: (8711.528347348114, 8759.376410068553)
Sample Size 300: (8259.034916666667, 9255.859999999999)
Sample Size 3000: (8559.511391666667, 8897.125233333334)
Sample Size 30000: (8684.918426666665, 8788.374099166667)

Confidence Interval for M:
Sample Size 414259: (9412.060575750918, 9443.164062579208)
Sample Size 300: (8871.585416666667, 9982.472083333332)
Sample Size 3000: (9251.932858333334, 9601.1675)
Sample Size 30000: (9374.023946666666, 9483.333472499999)
```

```
# Confidence intervals for marital status
statuses = data['Marital_Status'].unique()
for status in statuses:
    subset = data[data['Marital_Status'] == status]
    print(f"\nConfidence Interval for Marital Status {status}:")
    for size in [len(subset), 300, 3000, 30000]:
        ci = bootstrap_confidence_interval(subset, 1000, size, 'Purchase')
        print(f"Sample Size {size}: {ci[:2]}")
```

```
Confidence Interval for Marital Status 0:
Sample Size 324731: (9241.230408476555, 9275.313175212714)
Sample Size 300: (8724.838416666666, 9807.18125)
Sample Size 3000: (9081.822875, 9450.506858333334)
Sample Size 30000: (9200.293735833333, 9309.948346666668)

Confidence Interval for Marital Status 1:
Sample Size 225337: (9234.073591886818, 9274.115447196864)
Sample Size 300: (8705.054416666666, 9786.2755)
Sample Size 3000: (9088.085008333333, 9428.65265)
Sample Size 30000: (9199.342725833332, 9308.085151666666)
```

```
# Confidence intervals for different age groups
ages = data['Age'].unique()
for age in ages:
    subset = data[data['Age'] == age]
    print(f"\nConfidence Interval for Age Group {age}:")
    for size in [len(subset), 300, 3000, 30000]:
        ci = bootstrap_confidence_interval(subset, 1000, size, 'Purchase')
        print(f"Sample Size {size}: {ci[:2]}")
```

```
Confidence Interval for Age Group 0-17:
Sample Size 15102: (8865.503456495828, 9015.961597801615)
Sample Size 300: (8404.127666666665, 9516.780416666666)
Sample Size 3000: (8756.688675, 9122.665366666668)
Sample Size 30000: (8887.610594166666, 8992.908834166667)

Confidence Interval for Age Group 55+:
Sample Size 21504: (9265.84271530878, 9390.011852446056)
Sample Size 300: (8800.433583333333, 9851.695583333332)
Sample Size 3000: (9156.430841666666, 9491.554016666667)
Sample Size 30000: (9270.782683333335, 9381.112595)

Confidence Interval for Age Group 26-35:
Sample Size 219587: (9225.061488498863, 9263.651989871896)
Sample Size 300: (8709.063166666667, 9836.642833333333)
Sample Size 3000: (9072.387716666666, 9423.542591666666)
Sample Size 30000: (9186.0919125, 9296.554071666667)

Confidence Interval for Age Group 46-50:
Sample Size 45701: (9160.470384674296, 9248.591805977987)
Sample Size 300: (8622.108833333334, 9717.619666666666)
Sample Size 3000: (9035.347600000001, 9377.877841666666)
Sample Size 30000: (9146.942451666668, 9255.579011666667)

Confidence Interval for Age Group 51-55:
Sample Size 38501: (9466.464231448534, 9561.608067322926)
Sample Size 300: (8983.836916666667, 10025.896583333333)
Sample Size 3000: (9334.387858333333, 9687.123975)
Sample Size 30000: (9459.419595833333, 9568.48215)

Confidence Interval for Age Group 36-45:
Sample Size 110013: (9294.426501640715, 9350.84199662767)
Sample Size 300: (8790.171250000001, 9906.687083333334)
Sample Size 3000: (9160.85715, 9504.244316666667)
Sample Size 30000: (9268.3776475, 9379.639568333334)

Confidence Interval for Age Group 18-25:
```

```
Sample Size 99660: (9139.871410295003, 9198.158622566727)
Sample Size 300: (8648.196583333332, 9737.171083333333)
Sample Size 3000: (8986.419691666668, 9349.034158333332)
Sample Size 30000: (9114.123111666668, 9226.678635833334)
```

Start coding or generate with AI.

Start coding or generate with AI.

Start coding or generate with AI.

Start coding or generate with AI.

+ Code + Text

Start coding or generate with AI.

Start coding or generate with AI.

Start coding or generate with AI.

Start coding or generate with AI.

Start coding or generate with AI.

Start coding or generate with AI.

Start coding or generate with AI.

Start coding or generate with AI.

Start coding or generate with AI.

Start coding or generate with AI.

Start coding or generate with AI.

Start coding or generate with AI.

Start coding or generate with AI.

Start coding or generate with AI.

Start coding or generate with AI.

Start coding or generate with AI.

Start coding or generate with AI.

Start coding or generate with AI.

Start coding or generate with AI.

Start coding or generate with AI.