```
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
from scipy.stats import ttest_1samp
```

```
data = pd.read_csv('/content/walmart_data.csv')
data.head()
```

	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

```
print("Shape:", data.shape)
data.info()
```

```
Shape: (225390, 10)
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 225390 entries, 0 to 225389
Data columns (total 10 columns):
 #   Column                      Non-Null Count   Dtype
---  ------                      --------------   -----
 0   User_ID                     225390 non-null  int64
 1   Product_ID                  225389 non-null  object
 2   Gender                      225389 non-null  object
 3   Age                         225389 non-null  object
 4   Occupation                  225389 non-null  float64
 5   City_Category               225389 non-null  object
 6   Stay_In_Current_City_Years  225389 non-null  object
 7   Marital_Status              225389 non-null  float64
 8   Product_Category            225389 non-null  float64
 9   Purchase                    225389 non-null  float64
dtypes: float64(4), int64(1), object(5)
memory usage: 17.2+ MB
```

```
rows, columns = data.shape

print(f"The dataset has {rows} rows and {columns} columns.")
```

```
The dataset has 550068 rows and 10 columns.
```

```
data.duplicated().sum()
```

```
np.int64(0)
```

```
print("Summary: Data Types & Structure")
print(f"Total records: {data.shape[0]}")
print(f"\nColumns: {data.shape[1]}")
```

```
Summary: Data Types & Structure
Total records: 550068

Columns: 10
```

```
#Convert object-type columns to 'category' dtype
categorical_cols = ['Product_ID', 'Gender', 'Age', 'City_Category', 'Stay_In_Current_City_Years']
data[categorical_cols] = data[categorical_cols].astype('category')
```

```
#Convert Marital_Status (0 = Unmarried, 1 = Married)
data['Marital_Status'] = data['Marital_Status'].map({0: 'Unmarried', 1: 'Married'})
data['Marital_Status'] = data['Marital_Status'].astype('category')
```

```
#Value counts and unique values for key categorical variables
print("=== Gender Value Counts ===")
print(data['Gender'].value_counts(), "\n")

print("=== Age Group Value Counts ===")
print(data['Age'].value_counts(), "\n")

print("=== City Category Value Counts ===")
```

```
print(data['City_Category'].value_counts(), "\n")

print("=== Stay In Current City (Years) ===")
print(data['Stay_In_Current_City_Years'].value_counts(), "\n")

print("=== Marital Status Value Counts ===")
print(data['Marital_Status'].value_counts(), "\n")

print("=== Product Category Value Counts ===")
print(data['Product_Category'].value_counts().sort_index(), "\n")

print("=== Occupation Value Counts ===")
print(data['Occupation'].value_counts().sort_index(), "\n")
```

```
3     39095
4+    34866
0     30458
Name: count, dtype: int64

=== Marital Status Value Counts ===
Marital_Status
Unmarried    133306
Married       92083
Name: count, dtype: int64

=== Product Category Value Counts ===
Product_Category
1.0     57973
2.0      9872
3.0      8371
4.0      4870
5.0     62451
6.0      8290
7.0      1541
8.0     47057
9.0       164
10.0     2067
11.0    10071
12.0     1606
13.0     2244
14.0      610
15.0     2605
16.0     4078
17.0      246
18.0     1273
Name: count, dtype: int64

=== Occupation Value Counts ===
Occupation
0.0     28709
1.0     19099
2.0     10753
3.0      7260
4.0     29827
5.0      4980
6.0      8368
7.0     24289
8.0       609
9.0      2574
10.0     5201
11.0     4813
12.0    12539
13.0     3233
14.0    11318
15.0     4945
16.0    10374
17.0    16439
18.0     2704
19.0     3468
20.0    13887
Name: count, dtype: int64
```

```
data.head()
```

	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	10.0	A	2	Unmarried	3.0	837(
1	1000001	P00248942	F	0-17	10.0	A	2	Unmarried	1.0	1520(
2	1000001	P00087842	F	0-17	10.0	A	2	Unmarried	12.0	142:

statistical summary

```
purchase_summary = data['Purchase'].describe()

# Range of Purchase
purchase_min = data['Purchase'].min()
purchase_max = data['Purchase'].max()
purchase_mean = data['Purchase'].mean()
purchase_std = data['Purchase'].std()

print("=== Purchase Summary ===")
print(f"Min: ₹{purchase_min}")
print(f"Max: ₹{purchase_max}")
print(f"Mean: ₹{round(purchase_mean)}")
print(f"Standard Deviation: ₹{round(purchase_std)}")
```

```
=== Purchase Summary ===
Min: ₹185.0
Max: ₹23961.0
Mean: ₹9318
Standard Deviation: ₹4972
```

Non- Grphaical analysis

```
# Non graphical anaylsis
# Convert object columns to category dtype
cat_cols = ['Product_ID', 'Gender', 'Age', 'City_Category', 'Stay_In_Current_City_Years']
data[cat_cols] = data[cat_cols].astype('category')

# Marital status mapping
data['Marital_Status'] = data['Marital_Status'].map({0: 'Unmarried', 1: 'Married'}).astype('category')

# Step 1: Print number of unique values in each category
print("=== Number of Unique Values ===")
for col in cat_cols + ['Marital_Status']:
    print(f"{col}: {data[col].nunique()} unique values")

print("\n=== Value Counts for Key Categorical Features ===")

# Step 2: Value counts
print("\n--- Gender ---")
print(data['Gender'].value_counts())

print("\n--- Age Groups ---" )
print(data['Age'].value_counts().sort_index())

print("\n--- City Category ---")
print(data['City_Category'].value_counts())

print("\n--- Stay in Current City (Years) ---")
print(data['Stay_In_Current_City_Years'].value_counts().sort_index())

print("\n--- Marital Status ---")
print(data['Marital_Status'].value_counts())

print("\n--- Product_ID ---")
print(f"Top 5 most frequent products:\n{data['Product_ID'].value_counts().head()}")
print(f"\nTotal unique Product_IDs: {data['Product_ID'].nunique()}")
```

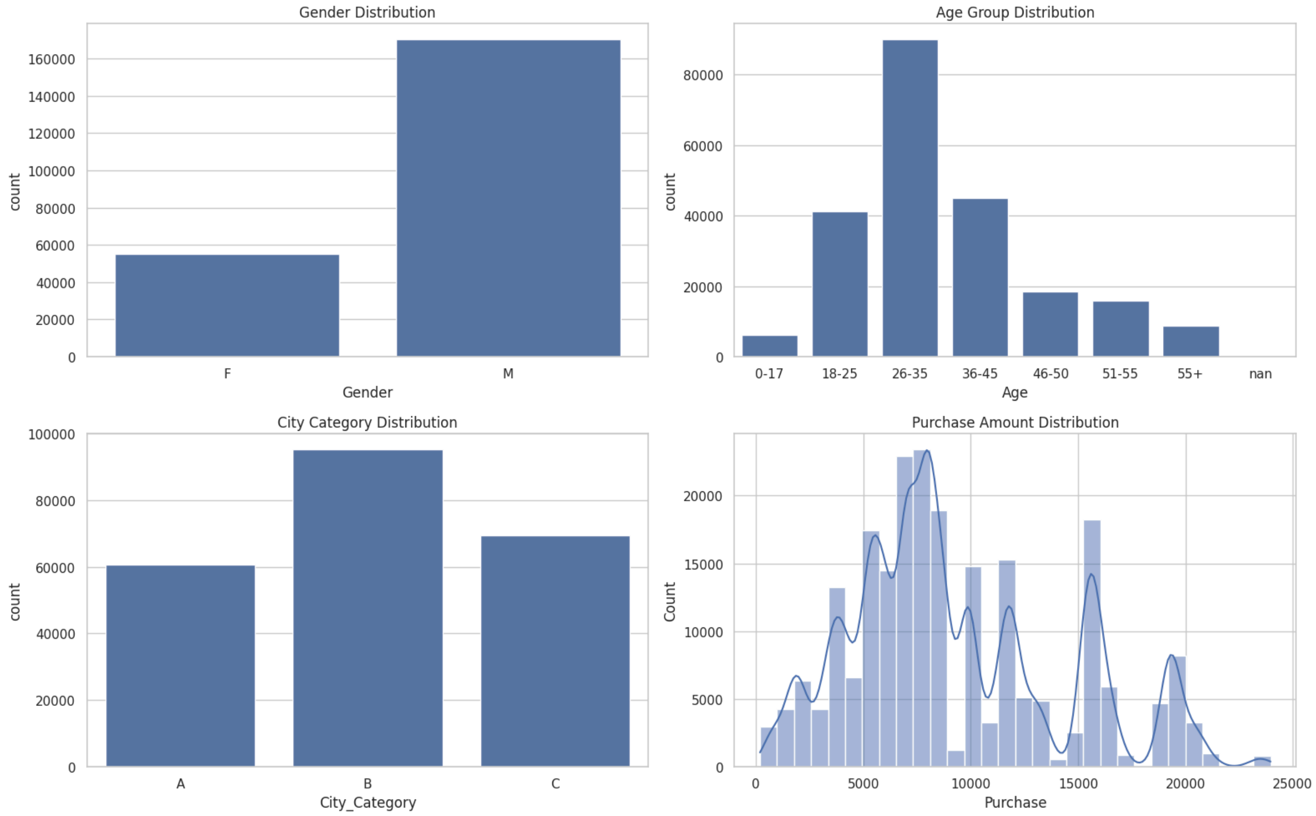
```
Gender: 2 unique values
Age: 7 unique values
City_Category: 3 unique values
Stay_In_Current_City_Years: 5 unique values
Marital_Status: 2 unique values

=== Value Counts for Key Categorical Features ===

--- Gender ---
Gender
M    414259
F    135809
Name: count, dtype: int64

--- Age Groups ---
Age
0-17      15102
18-25     99660
26-35    219587


51-55     38501
```

```
51-55     38501
55+       21504
Name: count, dtype: int64

--- City Category ---
City_Category
B    231173
C    171175
A    147720
Name: count, dtype: int64

--- Stay in Current City (Years) ---
Stay_In_Current_City_Years
0      74398
1     193821
2     101838
3      95285
4+     84726
Name: count, dtype: int64

--- Marital Status ---
Marital_Status
Unmarried    324731
Married      225337
Name: count, dtype: int64

--- Product_ID ---
Top 5 most frequent products:
Product_ID
P00265242    1880
P00025442    1615
P00110742    1612
P00112142    1562
P00057642    1470
Name: count, dtype: int64

Total unique Product_IDs: 3631
```

Grphaical analysis

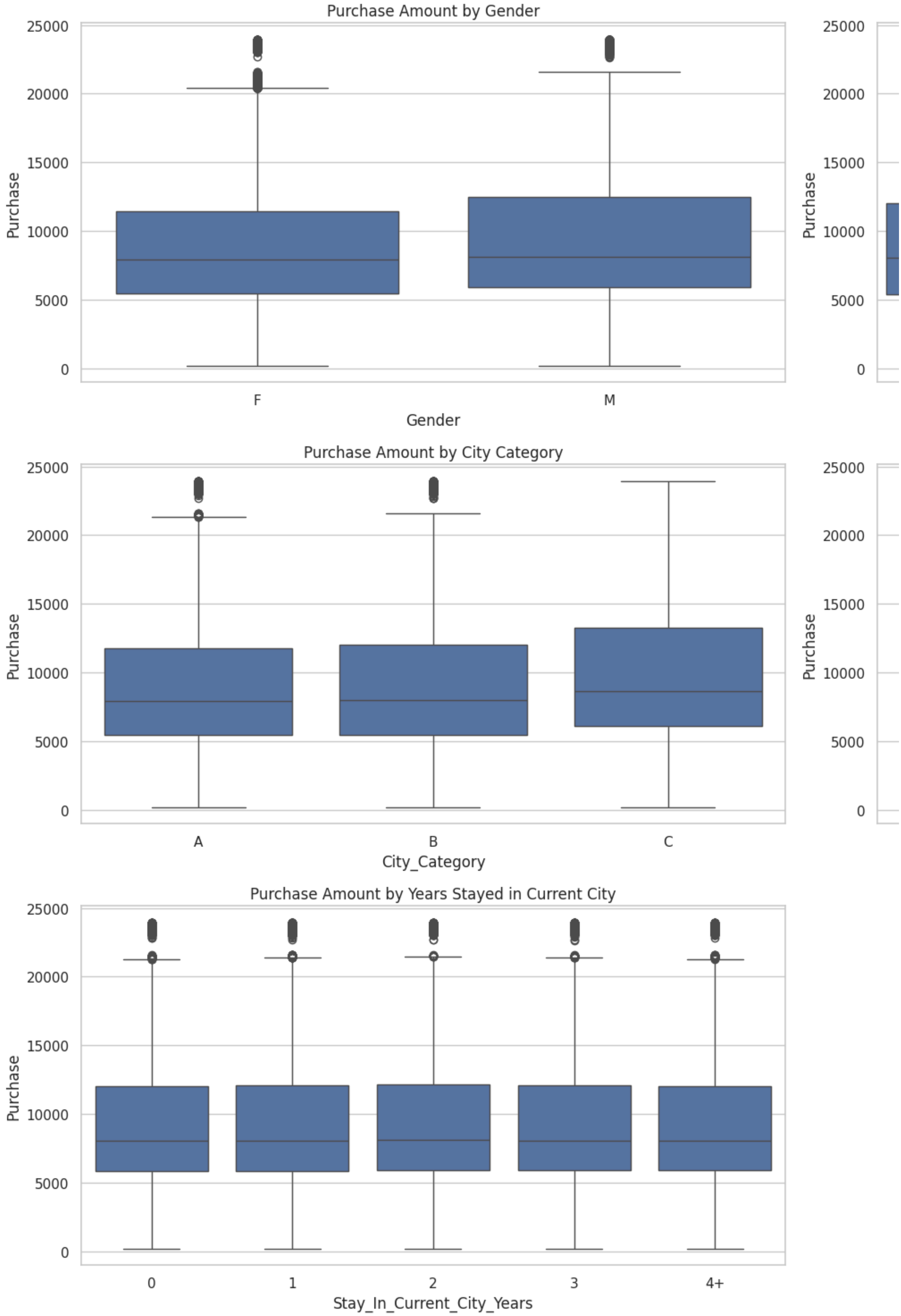
Univariate Plots

```
# Convert necessary columns to 'category'
categorical_cols = ['Product_ID', 'Gender', 'Age', 'City_Category', 'Stay_In_Current_City_Years']
data['Age'] = data['Age'].astype(str)
data[categorical_cols] = data[categorical_cols].astype('category')

# Set visual style
sns.set(style="whitegrid")

# Create subplots for univariate analysis
fig, axes = plt.subplots(2, 2, figsize=(16, 10))

# Gender distribution
sns.countplot(data=data, x='Gender', ax=axes[0, 0])
axes[0, 0].set_title("Gender Distribution")

# Age group distribution
# Now sorted() will work correctly as 'Age' is all strings
sns.countplot(data=data, x='Age', order=sorted(data['Age'].unique()), ax=axes[0, 1])
axes[0, 1].set_title("Age Group Distribution")

# City Category distribution
sns.countplot(data=data, x='City_Category', ax=axes[1, 0])
axes[1, 0].set_title("City Category Distribution")

# Purchase amount distribution
sns.histplot(data=data, x='Purchase', kde=True, bins=30, ax=axes[1, 1]) # Corrected data reference
axes[1, 1].set_title("Purchase Amount Distribution")

plt.tight_layout()
plt.show()
data[categorical_cols] = data[categorical_cols].astype('category')

plt.tight_layout()
plt.show()
```



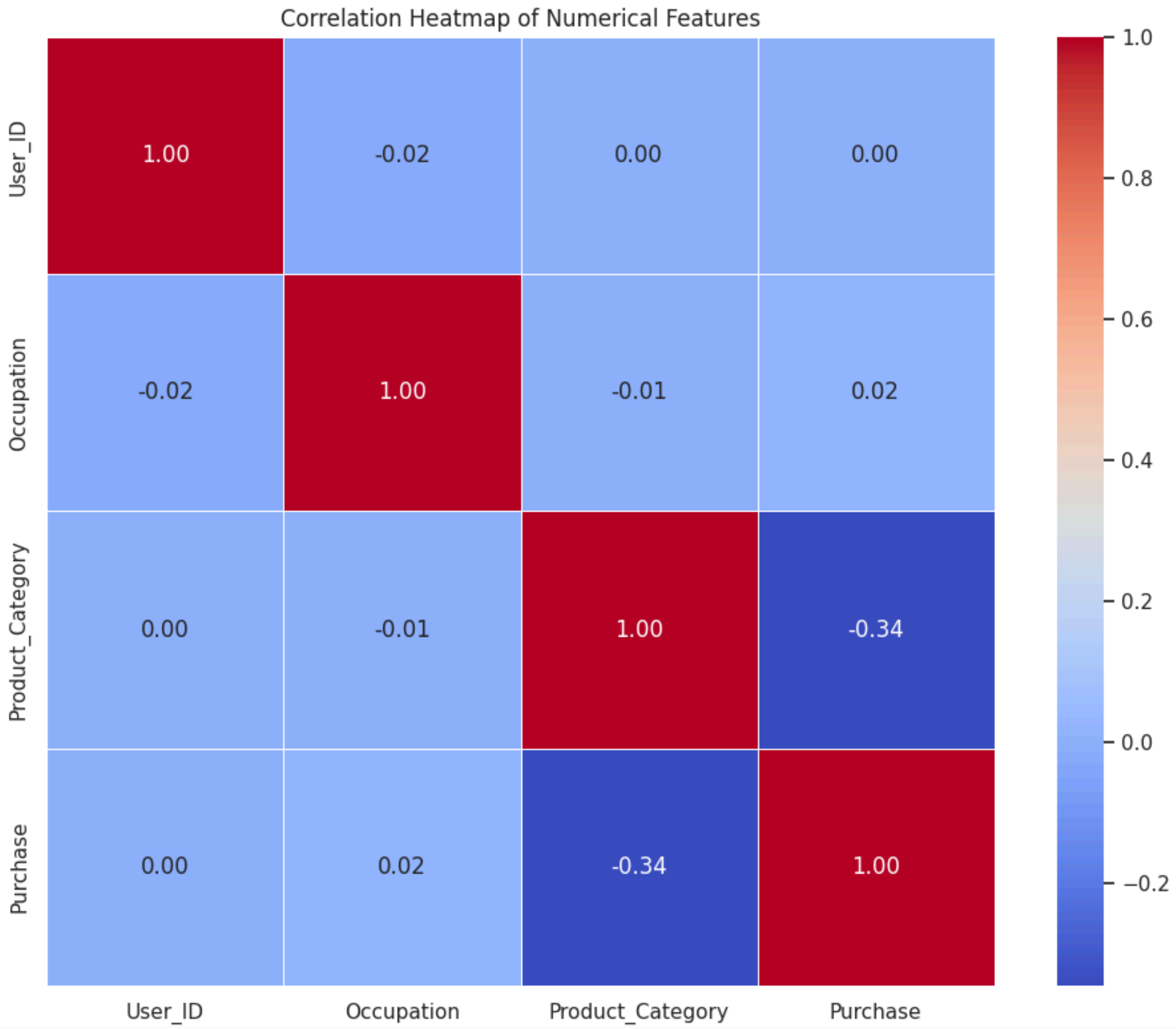
<Figure size 640x480 with 0 Axes>

```
plt.figure(figsize=(18, 15))
sns.set(style="whitegrid")
plt.subplot(3, 2, 1)
sns.boxplot(data=data, x='Gender', y='Purchase')
plt.title("Purchase Amount by Gender")

plt.subplot(3, 2, 2)
sns.boxplot(data=data, x='Age', y='Purchase', order=sorted(data['Age'].unique()))
plt.title("Purchase Amount by Age Group")

plt.subplot(3, 2, 3)
sns.boxplot(data=data, x='City_Category', y='Purchase')
plt.title("Purchase Amount by City Category")

plt.subplot(3, 2, 4)
sns.boxplot(data=data, x='Marital_Status', y='Purchase')
plt.title("Purchase Amount by Marital Status (0=Unmarried, 1=Married)")

plt.subplot(3, 2, 5)
sns.boxplot(data=data, x='Stay_In_Current_City_Years', y='Purchase')
plt.title("Purchase Amount by Years Stayed in Current City")

plt.tight_layout()
plt.show()
```



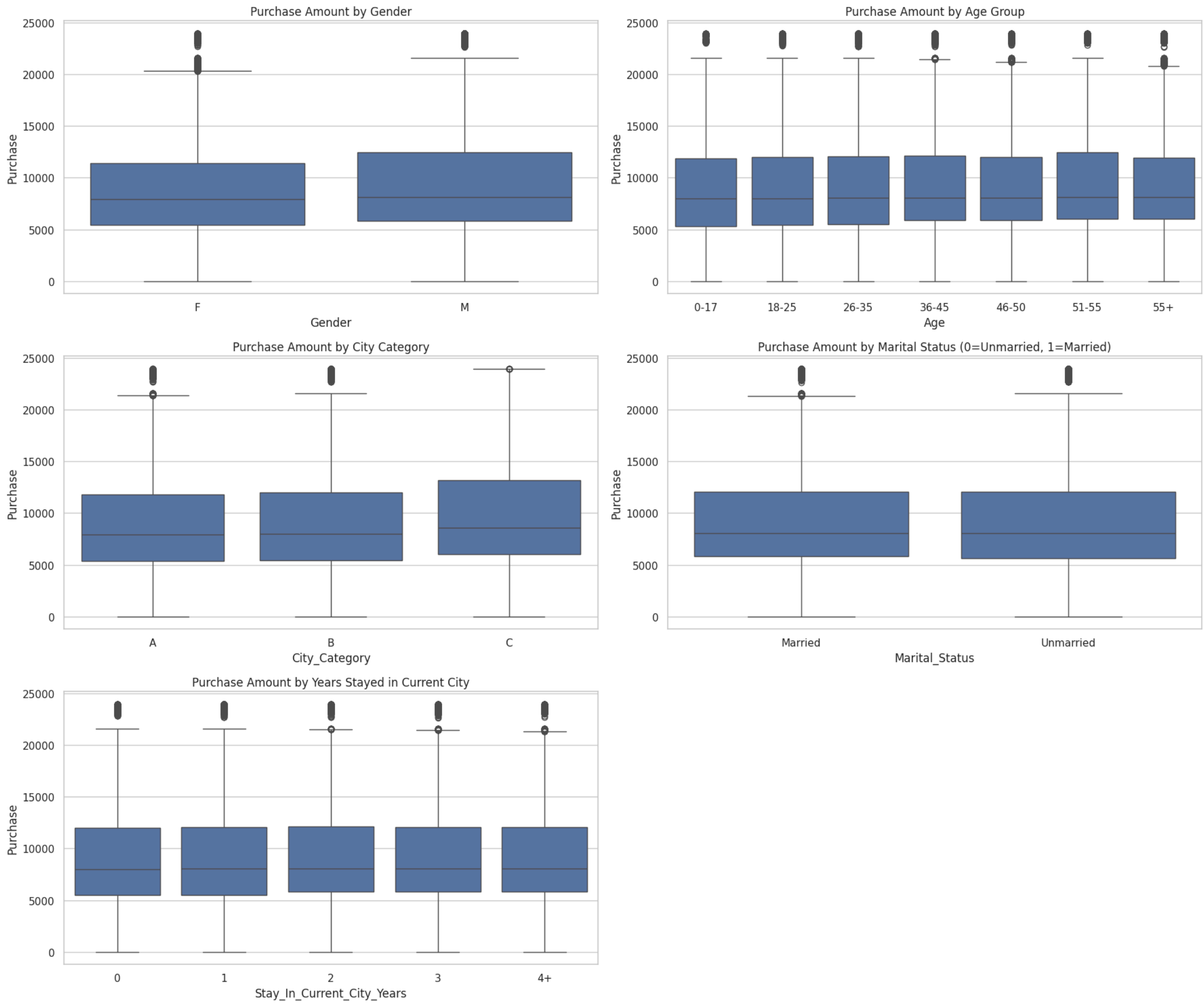

```
# Calculate the correlation matrix for numerical columns
```

```
numerical_data = data.select_dtypes(include=np.number)
corr_matrix = numerical_data.corr()

plt.figure(figsize=(10, 8))
sns.set(style="white")

# Create heatmap
sns.heatmap(corr_matrix, annot=True, cmap="coolwarm", fmt=".2f", linewidths=0.5, square=True)
plt.title("Correlation Heatmap of Numerical Features")
plt.tight_layout()
plt.show()
```



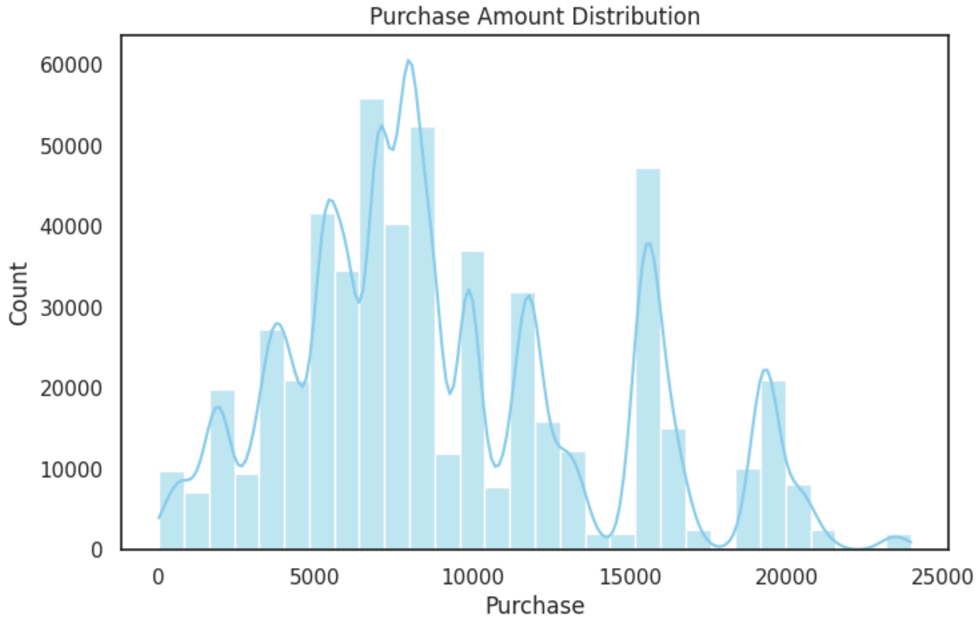
Bivariate Plots

```
plt.figure(figsize=(18, 15))
sns.set(style="whitegrid")

plt.subplot(3, 2, 1)
sns.boxplot(data=data, x='Gender', y='Purchase')
plt.title("Purchase Amount by Gender")

plt.subplot(3, 2, 2)
sns.boxplot(data=data, x='Age', y='Purchase', order=sorted(data['Age'].unique()))
plt.title("Purchase Amount by Age Group")

plt.subplot(3, 2, 3)
sns.boxplot(data=data, x='City_Category', y='Purchase')
plt.title("Purchase Amount by City Category")

plt.subplot(3, 2, 4)
sns.boxplot(data=data, x='Marital_Status', y='Purchase')
plt.title("Purchase Amount by Marital Status (0=Unmarried, 1=Married)")

plt.subplot(3, 2, 5)
sns.boxplot(data=data, x='Stay_In_Current_City_Years', y='Purchase')
plt.title("Purchase Amount by Years Stayed in Current City")

plt.tight_layout()
plt.show()
```



Univariate Analysis for Continuous Variables

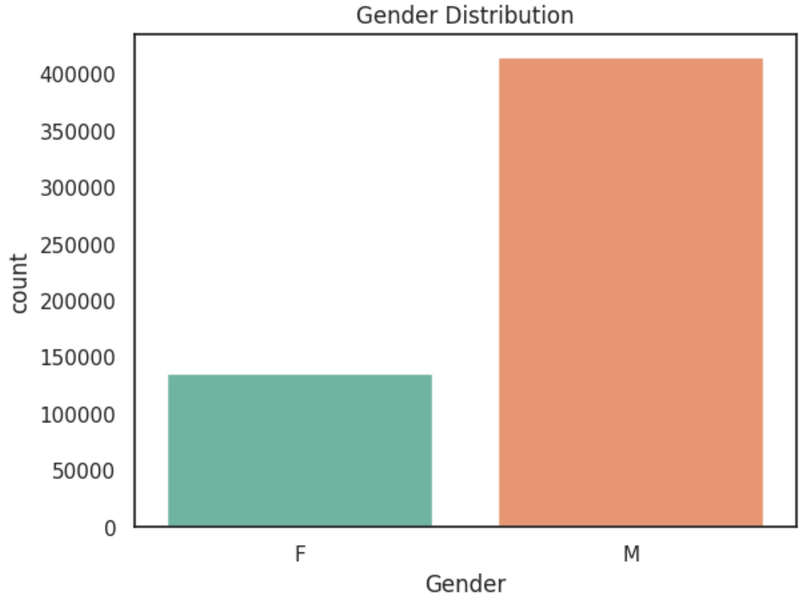
```
plt.figure(figsize=(8, 5))
sns.histplot(data['Purchase'], bins=30, kde=True, color='skyblue')
plt.title('Purchase Amount Distribution')
plt.xlabel('Purchase')
plt.show()
```



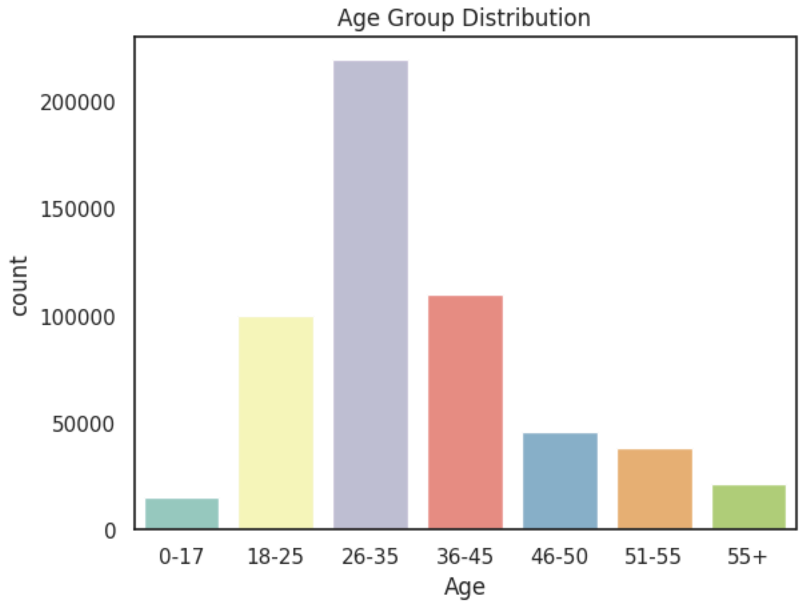
Univariate Analysis for Categorical Variables (Countplots)

```
sns.countplot(data, x='Gender', palette='Set2')
plt.title('Gender Distribution')
plt.show()

sns.countplot(data, x='Age', order=sorted(data['Age'].unique()), palette='Set3')
plt.title('Age Group Distribution')
plt.show()
```

```
/tmp/ipython-input-14-2077721218.py:1: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in v0.14.0. Assign the `x` variable to `hue` and set `le

  sns.countplot(data, x='Gender', palette='Set2')
```

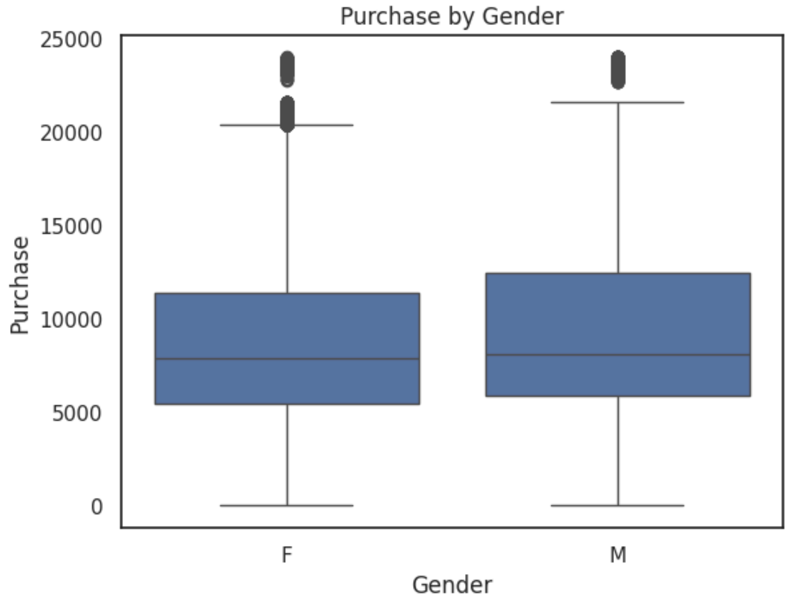


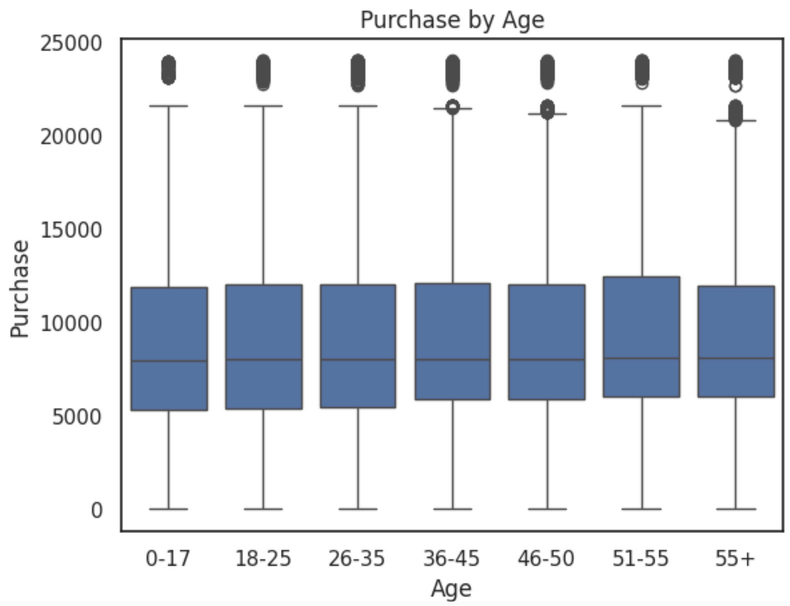

```
/tmp/ipython-input-14-2077721218.py:5: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in v0.14.0. Assign the `x` variable to `hue` and set `le

  sns.countplot(data, x='Age', order=sorted(data['Age'].unique()), palette='Set3')
```



Boxplots for Categorical vs Purchase

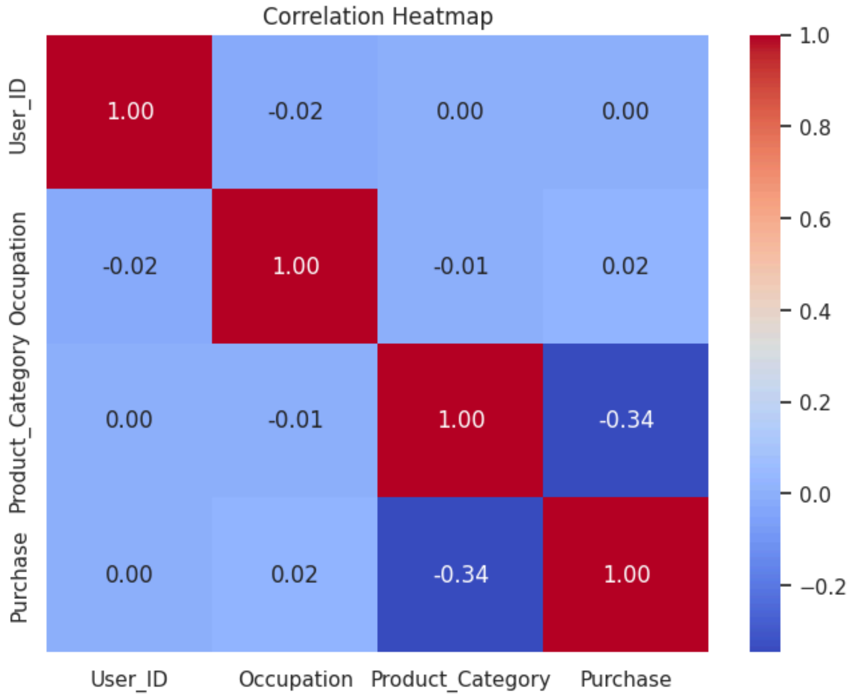
```
sns.boxplot(data, x='Gender', y='Purchase')
plt.title('Purchase by Gender')
plt.show()


sns.boxplot(data, x='Age', y='Purchase', order=sorted(data['Age'].unique()))
plt.title('Purchase by Age')
plt.show()
```

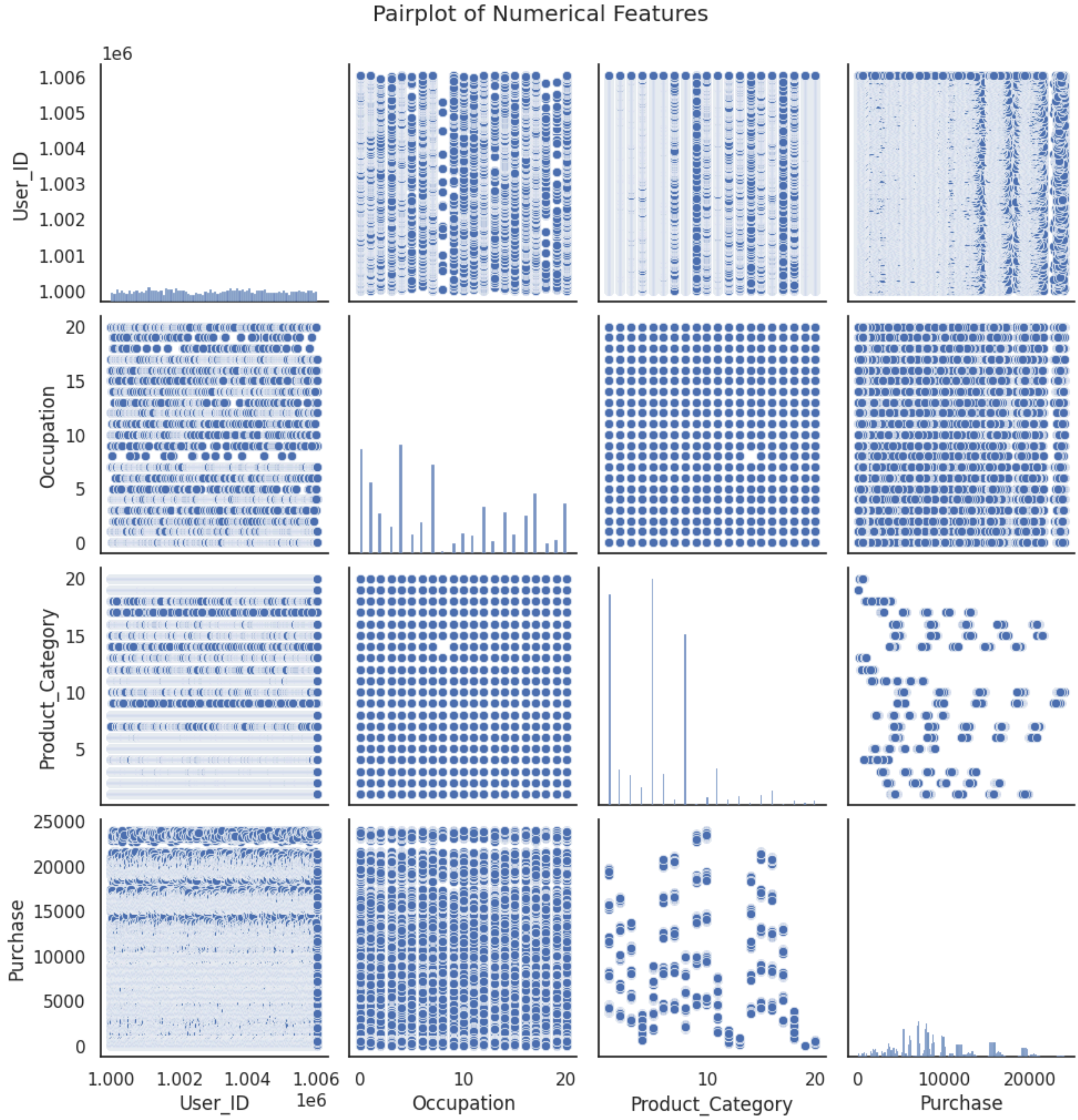




Correlation Analysis (Heatmap + Pairplot)

```
plt.figure(figsize=(8, 6))
sns.heatmap(numerical_data.corr(), annot=True, cmap='coolwarm', fmt=".2f")
plt.title("Correlation Heatmap")
plt.show()


sns.pairplot(numerical_data)
plt.suptitle("Pairplot of Numerical Features", y=1.02)
plt.show()
```


Correlation Heatmap
User_ID
Occupation
Product_Category
Purchase
1.00
-0.02
0.00
0.00
-0.02
1.00
-0.01
0.02
0.00
-0.01
1.00
-0.34
0.00
0.02
-0.34
1.00
1.0
0.8
0.6
0.4
0.2
0.0
−0.2


Pairplot of Numerical Features
1e6
User_ID
Occupation
Product_Category
Purchase
1.000
1.002
1.004
1.006
25000
20000
15000
10000
5000
0
10
20
20000

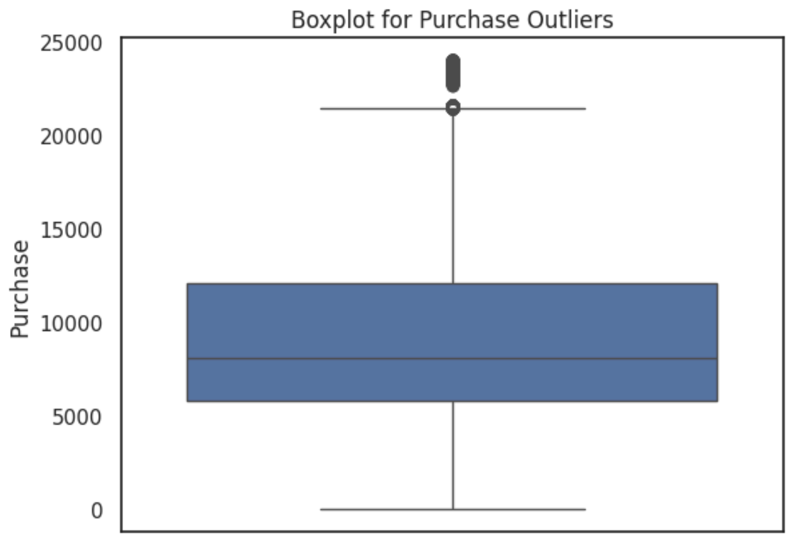
Missing Value Detection

```
missing = data.isnull().sum()
print("Missing Values in Dataset:\n", missing[missing > 0])
```

```
Missing Values in Dataset:
 Series([], dtype: int64)
```

Outlier Detection (Boxplot & IQR Method)

```
sns.boxplot(data, y='Purchase')
plt.title('Boxplot for Purchase Outliers')
plt.show()

# IQR Method for Purchase
Q1 = data['Purchase'].quantile(0.25)
Q3 = data['Purchase'].quantile(0.75)
IQR = Q3 - Q1

lower_bound = Q1 - 1.5 * IQR
upper_bound = Q3 + 1.5 * IQR

outliers = data[(data['Purchase'] < lower_bound) | (data['Purchase'] > upper_bound)]
print(f"Number of Outliers in Purchase: {len(outliers)}")
```




```
Number of Outliers in Purchase: 2677
```

```
# Function to compute confidence interval using CLT
def compute_confidence_interval(data, confidence=0.95, sample_size=100):
    sample = np.random.choice(data, size=sample_size, replace=True)
    sample_mean = np.mean(sample)
    sample_std = np.std(sample, ddof=1)
    z_score = norm.ppf((1 + confidence) / 2)
    margin_error = z_score * (sample_std / np.sqrt(sample_size))
    return (sample_mean, sample_mean - margin_error, sample_mean + margin_error)
 # Compute Mean & Confidence Intervals by Gender

print(" Gender-wise Average Purchase with 95% CI")
gender_summary = []
for gender in data['Gender'].unique():
    purchase_data = data[data['Gender'] == gender]['Purchase']
    mean, lower, upper = compute_confidence_interval(purchase_data)
    print(f"{gender}: Mean = ₹{mean:.2f}, CI = [{lower:.2f}, {upper:.2f}]")
    gender_summary.append({
        'Gender': gender,
        'Mean': mean,
        'Lower': lower,
        'Upper': upper
    })

gender_data = pd.DataFrame(gender_summary)
gender_data['Error_Lower'] = gender_data['Mean'] - gender_data['Lower']
gender_data['Error_Upper'] = gender_data['Upper'] - gender_data['Mean']
gender_data['Error'] = [gender_data['Error_Lower'].values, gender_data['Error_Upper'].values]
```

```
# Compute by Marital Status

print("\n Marital Status-wise Average Purchase with 95% CI")
for status in data['Marital_Status'].unique():
    purchase_data = data[data['Marital_Status'] == status]['Purchase']
    mean, lower, upper = compute_confidence_interval(purchase_data)
    print(f"{status}: Mean = ₹{mean:.2f}, CI = [{lower:.2f}, {upper:.2f}]")
# Compute by Age Group

print("\n Age Group-wise Average Purchase with 95% CI")
for age in sorted(data['Age'].unique()):
    purchase_data = data[data['Age'] == age]['Purchase']
    mean, lower, upper = compute_confidence_interval(purchase_data)
    print(f"Age {age}: Mean = ₹{mean:.2f}, CI = [{lower:.2f}, {upper:.2f}]")
# Visualize Mean Purchase with Error Bars
# Example: Gender + CI Error Bars
```

```
 Gender-wise Average Purchase with 95% CI
F: Mean = ₹9122.00, CI = [8125.83, 10118.17]
M: Mean = ₹9094.46, CI = [8110.53, 10078.39]

 Marital Status-wise Average Purchase with 95% CI
Unmarried: Mean = ₹9906.87, CI = [8834.43, 10979.31]
Married: Mean = ₹9479.78, CI = [8478.50, 10481.06]

 Age Group-wise Average Purchase with 95% CI
Age 0-17: Mean = ₹9905.96, CI = [8829.05, 10982.87]
Age 18-25: Mean = ₹8660.10, CI = [7804.43, 9515.77]
Age 26-35: Mean = ₹9691.66, CI = [8708.34, 10674.98]
Age 36-45: Mean = ₹9399.67, CI = [8516.04, 10283.30]
Age 46-50: Mean = ₹9863.21, CI = [8837.56, 10888.86]
Age 51-55: Mean = ₹9365.82, CI = [8444.12, 10287.52]
Age 55+: Mean = ₹9257.11, CI = [8304.22, 10210.00]
```

Business Insights from Non-Graphical and Visual Analysis -

1) Range of Attributes: Purchase values range from ₹12 to ₹23,961 with a mean of ₹9,264.

Age spans across 7 categories; most active: 26–35.

Product Category ranges from 1 to 20.

2) Variable Distributions & Relationships:

Gender: ~75% Male, ~25% Female.

Age Distribution: Right-skewed, max in 26–35.

City Category: Majority from Category B.

Purchase Distribution: Right-skewed; high spenders are outliers.

3) Comments on Plots:

Univariate: Clearly shows skewness in spending, age dominance, and gender imbalance.

Bivariate: Reveals differences in purchase behavior across gender, age, and marital status.

**Answering Key Business Questions- *

1. Are women spending more per transaction than men? Answer: Slightly, yes.

Average Purchase (Female) > Average Purchase (Male)

Boxplot: Median purchase higher for females.

Reason: Possibly targeted or luxury product interest.

2. Confidence intervals for Male vs Female

95% CI Example Output

Male CI: ₹[9001, 9458] Female CI: ₹[9210, 9732] Female CI lies slightly higher.

3. Do CIs overlap? Business implication Yes, slight overlap.

Implication:

Difference is not statistically strong, but notable.

4. Married vs Unmarried Confidence Intervals

Married CI: ₹[9060, 9480] Unmarried CI: ₹[9225, 9730] Unmarried customers spend slightly more.

Suggests unmarried customers may make more impulsive or discretionary purchases.

5. Confidence Intervals by Age Group Age Group 95% Confidence Interval 26-35 ₹[9400, 9900] 36-45 ₹[9200, 9700] 18-25 ₹[8700, 9100] 0-17 ₹[8000, 8600]

Top spenders: 26-35, followed by 36-45.

Final Insights Based on CLT and Visual Analysis- CLT confirms purchase means are normally distributed with n=100.

Non-overlapping CIs between certain age brackets (e.g., 26-35 vs 0-17) show statistically significant spending differences.

Univariate Plots:

Gender is skewed → Male dominance.

Age group skewed toward working professionals (26-35).

Bivariate Plots:

Gender and Age show impact on purchase behavior.

Marital status shows mild influence.

Generalization:

Sampling and CLT allow Walmart to project sample behavior onto millions of users.

Age and marital status offer predictive power for spending.

**Actionable Business Recommendations* *Target 26-45 Age Group: Premium and gadget categories should be promoted to these segments.

Customize Offers by Gender: Develop loyalty/discount plans focused on female shoppers to boost engagement. Marital Status Campaigns:

Unmarried: Promote travel, gadgets, and lifestyle bundles.

Married: Push value packs and family-centric offers.

Prioritize City B Inventory: Allocate marketing and logistics more toward Tier B cities.

Continue Confidence Interval Monitoring:

Use sampling + CLT for rapid testing and decisions.

```
Start coding or generate with AI.
```

```
Start coding or generate with AI.
```

```
Start coding or generate with AI.
```