```
#Import necessary libraries
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from scipy import stats
```

```
# Load the dataset
try:
    df = pd.read_csv('walmart_data.csv')
    print("Dataset loaded successfully.")
except FileNotFoundError:
    print("Error: walmart_data.csv not found in the current directory.")
    exit()

print("\n--- Initial Data Analysis ---")
```

```
Dataset loaded successfully.

--- Initial Data Analysis ---
```

```
# Display basic information
print("\n1. Basic Information:")
print("Shape of the dataset:", df.shape)
print("\nData Types and Non-Null Counts:")
df.info()

# Display statistical summary
print("\n2. Statistical Summary:")
print(df.describe(include='all'))

# Check for null values
print("\n3. Null Value Counts:")
print(df.isnull().sum())
```

```
 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

2. Statistical Summary:
             User_ID Product_ID  Gender     Age     Occupation City_Category  \
count   5.500680e+05     550068  550068  550068  550068.000000        550068
unique           NaN       3631       2       7            NaN             3
top              NaN  P00265242       M   26-35            NaN             B
freq             NaN       1880  414259  219587            NaN        231173
mean    1.003029e+06        NaN     NaN     NaN       8.076707           NaN
std     1.727592e+03        NaN     NaN     NaN       6.522660           NaN
min     1.000001e+06        NaN     NaN     NaN       0.000000           NaN
25%     1.001516e+06        NaN     NaN     NaN       2.000000           NaN
50%     1.003077e+06        NaN     NaN     NaN       7.000000           NaN
75%     1.004478e+06        NaN     NaN     NaN      14.000000           NaN
max     1.006040e+06        NaN     NaN     NaN      20.000000           NaN

       Stay_In_Current_City_Years  Marital_Status  Product_Category  \
count                      550068   550068.000000     550068.000000
unique                          5             NaN               NaN
top                             1             NaN               NaN
freq                       193821             NaN               NaN
mean                          NaN        0.409653          5.404270
std                           NaN        0.491770          3.936211
min                           NaN        0.000000          1.000000
25%                           NaN        0.000000          1.000000
50%                           NaN        0.000000          5.000000
75%                           NaN        1.000000          8.000000
max                           NaN        1.000000         20.000000
```

```
Age                            0
Occupation                     0
City_Category                  0
Stay_In_Current_City_Years     0
Marital_Status                 0
Product_Category               0
Purchase                       0
dtype: int64
```

```python
# --- Data Cleaning and Preparation ---
print("\n--- Data Cleaning and Preparation ---")

# Handle missing values
# Product_Category_2 and Product_Category_3 have significant missing values.
# For this analysis, we might not need them directly for the core questions,
# but if we were building a predictive model, we'd need a strategy (e.g., imputation, treating as a separate category).
# Let's fill with 0 or a placeholder for now, assuming missing means the category doesn't apply or wasn't recorded.
# However, the prompt focuses on Purchase amount vs Gender, Age, Marital Status, so let's check if 'Purchase' has NaNs.
if df['Purchase'].isnull().any():
    print("\nWarning: 'Purchase' column contains missing values. Dropping rows with missing Purchase amount.")
    df.dropna(subset=['Purchase'], inplace=True)
else:
    print("\n'Purchase' column has no missing values.")

# Fill missing Product Categories if needed for specific analysis later, but focus on core task first.
# For now, we'll proceed without filling Product_Category_2 & 3 as they aren't central to the main questions.
print("\nNote: Product_Category_2 and Product_Category_3 have missing values, which are not being filled at this stage.")


# Convert relevant columns to appropriate types
df['Gender'] = df['Gender'].astype('category')
df['Age'] = df['Age'].astype('category')
df['City_Category'] = df['City_Category'].astype('category')
df['Stay_In_Current_City_Years'] = df['Stay_In_Current_City_Years'].astype('category')
df['Marital_Status'] = df['Marital_Status'].astype('category')
# User_ID and Product_ID could be treated as strings or objects if not used numerically
df['User_ID'] = df['User_ID'].astype(str)
df['Product_ID'] = df['Product_ID'].astype(str)


print("\nData types after conversion:")
df.info()


# Create Age Bins as requested
print("\nCreating Age Bins...")
# Define the mapping for age categories to sortable labels if needed,
# but for binning, the original categories work.
# Let's create a new column 'Age_Group' based on the specified bins.
def map_age_to_group(age_str):
    if age_str == '0-17':
        return '0-17'
    elif age_str == '18-25':
        return '18-25'
    elif age_str == '26-35':
        return '26-35'
    elif age_str == '36-45' or age_str == '46-50':
        return '36-50' # Grouping 36-45 and 46-50 into 36-50
    elif age_str == '51-55' or age_str == '55+':
        return '51+' # Grouping 51-55 and 55+ into 51+
    else:
        return 'Unknown'

df['Age_Group'] = df['Age'].apply(map_age_to_group).astype('category')
# Define the order for plotting if necessary
age_group_order = ['0-17', '18-25', '26-35', '36-50', '51+']
df['Age_Group'] = pd.Categorical(df['Age_Group'], categories=age_group_order, ordered=True)

print("Age groups created:")
print(df['Age_Group'].value_counts())
```

```
--- Data Cleaning and Preparation ---

'Purchase' column has no missing values.

Note: Product_Category_2 and Product_Category_3 have missing values, which are not being filled at this stage.

Data types after conversion:
<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  object
 1   Product_ID                  550068 non-null  object
 2   Gender                      550068 non-null  category
 3   Age                         550068 non-null  category
 4   Occupation                  550068 non-null  int64
 5   City_Category               550068 non-null  category
 6   Stay_In_Current_City_Years  550068 non-null  category
 7   Marital_Status              550068 non-null  category
 8   Product_Category            550068 non-null  int64
 9   Purchase                    550068 non-null  int64
dtypes: category(5), int64(3), object(2)
memory usage: 23.6+ MB

Creating Age Bins...
Age groups created:
Age_Group
26-35    219587
36-50    155714
18-25     99660
51+       60005
0-17      15102
Name: count, dtype: int64
```

```python
# --- Exploratory Data Analysis (EDA) ---
print("\n--- Exploratory Data Analysis (EDA) ---")

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

# 1. Univariate Analysis
print("\n1. Univariate Analysis...")

# Distribution of Purchase Amount
plt.figure(figsize=(10, 6))
sns.histplot(df['Purchase'], kde=True, bins=50)
plt.title('Distribution of Purchase Amount')
plt.xlabel('Purchase Amount')
plt.ylabel('Frequency')
# plt.show() # Displaying plots might not work directly in script execution, consider saving them.
plt.savefig('purchase_distribution.png')
plt.close()
print("Saved purchase_distribution.png")

# Count plots for categorical features
categorical_features = ['Gender', 'Age_Group', 'City_Category', 'Marital_Status', 'Stay_In_Current_City_Years']
for feature in categorical_features:
    plt.figure(figsize=(8, 5))
    sns.countplot(data=df, x=feature, order=df[feature].value_counts().index)
    plt.title(f'Count Plot for {feature}')
    plt.xlabel(feature)
    plt.ylabel('Count')
    plt.xticks(rotation=45 if len(df[feature].unique()) > 5 else 0)
    plt.tight_layout()
    plt.savefig(f'{feature}_countplot.png')
    plt.close()
    print(f"Saved {feature}_countplot.png")
```

```
--- Exploratory Data Analysis (EDA) ---

1. Univariate Analysis...
Saved purchase_distribution.png
Saved Gender_countplot.png
Saved Age_Group_countplot.png
Saved City_Category_countplot.png
Saved Marital_Status_countplot.png
Saved Stay_In_Current_City_Years_countplot.png
```

```python
# 2. Bivariate Analysis
print("\n2. Bivariate Analysis...")

# Purchase vs. Gender
plt.figure(figsize=(8, 6))
sns.boxplot(data=df, x='Gender', y='Purchase')
plt.title('Purchase Amount vs. Gender')
plt.xlabel('Gender')
plt.ylabel('Purchase Amount')
plt.savefig('purchase_vs_gender_boxplot.png')
plt.close()
print("Saved purchase_vs_gender_boxplot.png")

# Purchase vs. Marital Status
```

```
plt.figure(figsize=(8, 6))
sns.boxplot(data=df, x='Marital_Status', y='Purchase')
plt.title('Purchase Amount vs. Marital Status')
plt.xlabel('Marital Status (0=Single, 1=Married)')
plt.ylabel('Purchase Amount')
plt.savefig('purchase_vs_marital_status_boxplot.png')
plt.close()
print("Saved purchase_vs_marital_status_boxplot.png")

# Purchase vs. Age Group
plt.figure(figsize=(10, 6))
sns.boxplot(data=df, x='Age_Group', y='Purchase', order=age_group_order)
plt.title('Purchase Amount vs. Age Group')
plt.xlabel('Age Group')
plt.ylabel('Purchase Amount')
plt.savefig('purchase_vs_age_group_boxplot.png')
plt.close()
print("Saved purchase_vs_age_group_boxplot.png")

# Purchase vs. City Category
plt.figure(figsize=(8, 6))
sns.boxplot(data=df, x='City_Category', y='Purchase', order=['A', 'B', 'C'])
plt.title('Purchase Amount vs. City Category')
plt.xlabel('City Category')
plt.ylabel('Purchase Amount')
plt.savefig('purchase_vs_city_category_boxplot.png')
plt.close()
print("Saved purchase_vs_city_category_boxplot.png")

# Correlation Heatmap (for numerical columns if any were relevant - Purchase is the main one)
# Since most predictors are categorical, a heatmap isn't the primary tool here.
# We focus on comparing Purchase across categories.
```

```
2. Bivariate Analysis...
Saved purchase_vs_gender_boxplot.png
Saved purchase_vs_marital_status_boxplot.png
Saved purchase_vs_age_group_boxplot.png
Saved purchase_vs_city_category_boxplot.png
```

```
# --- Answering Business Questions ---
print("\n--- Answering Business Questions ---")

# Q1: Are women spending more money per transaction than men? Why or Why not?
print("\nQ1: Average Spending per Transaction by Gender")
# Add observed=False to silence FutureWarning and maintain current behavior
avg_spending_gender = df.groupby('Gender', observed=False)['Purchase'].mean()
print(avg_spending_gender)

male_avg = avg_spending_gender['M']
female_avg = avg_spending_gender['F']

if female_avg > male_avg:
    print(f"\nOn average, females spend slightly more per transaction (${female_avg:.2f}) than males (${male_avg:.2f}).")
else:
    print(f"\nOn average, males spend slightly more or equal per transaction (${male_avg:.2f}) compared to females (${female_avg:.2f}).")

# Potential Reasons (based on data exploration, requires domain knowledge for full explanation):
# - Differences in product categories purchased (requires analyzing Product_Category vs Gender).
# - Marketing targeting.
# - Cultural factors.
# The boxplot (purchase_vs_gender_boxplot.png) visually compares the distributions.


# Q2 & Q3: Confidence intervals and distribution for mean expenses (Gender)
print("\nQ2 & Q3: Confidence Intervals for Average Spending by Gender (using CLT)")

# Separate data for male and female customers
male_purchases = df[df['Gender'] == 'M']['Purchase']
female_purchases = df[df['Gender'] == 'F']['Purchase']

# Function to calculate confidence interval using CLT
def calculate_confidence_interval(data, confidence=0.95):
    """Calculates the confidence interval for the mean of a dataset."""
    n = len(data)
    if n < 30: # Basic check for CLT applicability, though often works for smaller samples if distribution isn't heavily skewed.
        print(f"Warning: Sample size ({n}) is small for CLT assumption.")
    mean = data.mean()
    std_err = stats.sem(data) # Standard Error of the Mean = std_dev / sqrt(n)
    if std_err == 0:
         print(f"Warning: Standard error is zero. Cannot calculate interval reliably. Data might be constant.")
         return (mean, mean) # Or handle as an error/special case
    interval = stats.norm.interval(confidence, loc=mean, scale=std_err)
```

```
    interval = stats.norm.interval(confidence, loc=mean, scale=std_err)
    return interval

# Calculate CIs for different confidence levels
confidence_levels = [0.90, 0.95, 0.99]
male_cis = {}
female_cis = {}

print("\nCalculating CIs with full sample data:")
for conf in confidence_levels:
    male_cis[conf] = calculate_confidence_interval(male_purchases, conf)
    female_cis[conf] = calculate_confidence_interval(female_purchases, conf)
    # Format CI output for better readability
    print(f"  {int(conf*100)}% CI for Males:   ({male_cis[conf][0]:.2f}, {male_cis[conf][1]:.2f})")
    print(f"  {int(conf*100)}% CI for Females: ({female_cis[conf][0]:.2f}, {female_cis[conf][1]:.2f})")

# Check for overlap
print("\nChecking for CI Overlap (Gender):")
overlapping = {}
for conf in confidence_levels:
    male_lower, male_upper = male_cis[conf]
    female_lower, female_upper = female_cis[conf]
    # Overlap exists if one interval's start is before the other's end, AND vice-versa
    overlap = (male_lower < female_upper) and (female_lower < male_upper)
    overlapping[conf] = overlap
    print(f"  {int(conf*100)}% CI Overlap: {overlap}")

# Interpretation of Overlap:
# If CIs overlap, we cannot conclude with that level of confidence that the true population means are different.
# If CIs do not overlap, we can conclude with that level of confidence that the true population means are different.

print("\nLeveraging Conclusion (Gender):")
if overlapping[0.95]: # Using 95% as standard
    print("  At 95% confidence, the intervals for average male and female spending overlap.")
    print("  This suggests that while there might be a small difference in the sample averages,")
    print("  we cannot be statistically confident that the true average spending for ALL male and female")
    print("  customers in the population is significantly different.")
    print("  Walmart might consider marketing strategies that appeal broadly rather than heavily gender-segmented based solely on average
else:
    print("  At 95% confidence, the intervals for average male and female spending DO NOT overlap.")
    print("  This provides statistical evidence that the true average spending differs between genders in the population.")
    # Determine who spends more based on non-overlapping intervals
    if male_cis[0.95][0] > female_cis[0.95][1]: # Male lower bound > Female upper bound
         print("  Males likely spend significantly more on average.")
         print("  Walmart could tailor promotions or product recommendations differently based on gender.")
    elif female_cis[0.95][0] > male_cis[0.95][1]: # Female lower bound > Male upper bound
         print("  Females likely spend significantly more on average.")
         print("  Walmart could tailor promotions or product recommendations differently based on gender.")


# Effect of Sample Size (Demonstration - requires resampling)
print("\nDemonstrating Effect of Sample Size on CI Width (using Male data):")
sample_sizes = [100, 1000, 10000, 50000]
for size in sample_sizes:
    if size <= len(male_purchases):
        sample = male_purchases.sample(n=size, random_state=42) # Use random_state for reproducibility
        ci = calculate_confidence_interval(sample, 0.95)
        width = ci[1] - ci[0]
        print(f"  Sample Size: {size}, 95% CI: {ci}, Width: {width:.2f}")
    else:
        print(f"  Sample Size: {size} exceeds available male data ({len(male_purchases)}). Skipping.")
print("Observation: As sample size increases, the confidence interval width decreases (becomes more precise).")
```

```
--- Answering Business Questions ---

Q1: Average Spending per Transaction by Gender
Gender
F    8734.565765
M    9437.526040
Name: Purchase, dtype: float64

On average, males spend slightly more or equal per transaction ($9437.53) compared to females ($8734.57).

Q2 & Q3: Confidence Intervals for Average Spending by Gender (using CLT)

Calculating CIs with full sample data:
  90% CI for Males:   (9424.51, 9450.54)
  90% CI for Females: (8713.29, 8755.84)
  95% CI for Males:   (9422.02, 9453.03)
  95% CI for Females: (8709.21, 8759.92)
  99% CI for Males:   (9417.15, 9457.91)
  99% CI for Females: (8701.24, 8767.89)
```

```
    Checking for CI Overlap (Gender):
      90% CI Overlap: False
      95% CI Overlap: False
      99% CI Overlap: False

    Leveraging Conclusion (Gender):
      At 95% confidence, the intervals for average male and female spending DO NOT overlap.
      This provides statistical evidence that the true average spending differs between genders in the population.
      Males likely spend significantly more on average.
      Walmart could tailor promotions or product recommendations differently based on gender.

    Demonstrating Effect of Sample Size on CI Width (using Male data):
      Sample Size: 100, 95% CI: (np.float64(8683.350394858206), np.float64(10755.729605141796)), Width: 2072.38
      Sample Size: 1000, 95% CI: (np.float64(9367.763916452222), np.float64(9998.610083547777)), Width: 630.85
      Sample Size: 10000, 95% CI: (np.float64(9409.419092422633), np.float64(9609.689907577367)), Width: 200.27
      Sample Size: 50000, 95% CI: (np.float64(9393.884281608176), np.float64(9483.131958391825)), Width: 89.25
    Observation: As sample size increases, the confidence interval width decreases (becomes more precise).
```

```python
#Q4: Results for Married vs Unmarried
print("\nQ4: Analysis for Marital Status")
# Add observed=False to silence FutureWarning and maintain current behavior
avg_spending_marital = df.groupby('Marital_Status', observed=False)['Purchase'].mean()
print("\nAverage Spending per Transaction by Marital Status (0=Single, 1=Married):")
print(avg_spending_marital)

single_purchases = df[df['Marital_Status'] == 0]['Purchase']
married_purchases = df[df['Marital_Status'] == 1]['Purchase']

single_ci_95 = calculate_confidence_interval(single_purchases, 0.95)
married_ci_95 = calculate_confidence_interval(married_purchases, 0.95)

# Format CI output
print(f"\n95% CI for Average Spending (Single):   ({single_ci_95[0]:.2f}, {single_ci_95[1]:.2f})")
print(f"95% CI for Average Spending (Married): ({married_ci_95[0]:.2f}, {married_ci_95[1]:.2f})")

# Check overlap for Marital Status
single_lower, single_upper = single_ci_95
married_lower, married_upper = married_ci_95
marital_overlap = (single_lower < married_upper) and (married_lower < single_upper)
print(f"\nOverlap in 95% CIs for Marital Status: {marital_overlap}")

if marital_overlap:
    print("  The confidence intervals for single and married customers overlap significantly.")
    print("  We cannot confidently conclude a difference in true average spending based on marital status alone.")
    print("  Marketing might not need strong differentiation based solely on marital status for average purchase value.")
else:
    print("  The confidence intervals DO NOT overlap, suggesting a statistically significant difference")
    print("  in average spending between single and married customers in the population.")
    # Determine who spends more
    if single_ci_95[0] > married_ci_95[1]:
        print("  Single customers likely spend significantly more on average.")
    elif married_ci_95[0] > single_ci_95[1]:
        print("  Married customers likely spend significantly more on average.")


# Q5: Results for Age Groups
print("\nQ5: Analysis for Age Groups")
# Add observed=False to silence FutureWarning and maintain current behavior
avg_spending_age = df.groupby('Age_Group', observed=False)['Purchase'].mean()
print("\nAverage Spending per Transaction by Age Group:")
print(avg_spending_age)

age_group_cis_95 = {}
print("\n95% Confidence Intervals for Average Spending by Age Group:")
for group in age_group_order:
    age_data = df[df['Age_Group'] == group]['Purchase']
    if len(age_data) > 0:
        age_group_cis_95[group] = calculate_confidence_interval(age_data, 0.95)
        # Format CI output
        print(f"  {group}: ({age_group_cis_95[group][0]:.2f}, {age_group_cis_95[group][1]:.2f})")
    else:
        print(f"  {group}: No data available.")

# Check for overlaps between adjacent or key groups (e.g., youngest vs oldest, peak vs others)
# This can get complex to report all pairs. Let's highlight key observations.
print("\nObservations on Age Group CIs:")
# Example comparison: 26-35 vs 51+
if '26-35' in age_group_cis_95 and '51+' in age_group_cis_95:
    ci1 = age_group_cis_95['26-35']
    ci2 = age_group_cis_95['51+']
    overlap = (ci1[0] < ci2[1]) and (ci2[0] < ci1[1])
    print(f"  Overlap between 26-35 and 51+ CIs: {overlap}")
    if not overlap:
        print("    Statistically significant difference in average spending between these groups.")
```

```
        print("    Statistically significant difference in average spending between these groups.")
else:
    print("  Could not compare '26-35' and '51+' due to missing data or intervals.")

print("  Visual inspection of the boxplot (purchase_vs_age_group_boxplot.png) and CIs suggests")
print("  that while average spending varies slightly across age groups, the distributions and CIs")
print("  show considerable overlap, indicating average spending might not differ dramatically")
print("  or statistically significantly between *all* adjacent age groups.")
print("  However, specific groups might show differences (e.g., potentially 18-50 groups vs. 0-17 or 51+).")
print("  Walmart could explore targeted promotions for age groups showing distinct higher spending patterns,")
print("  but broad strategies might be effective across the main adult age brackets (18-50).")
```

```
Q4: Analysis for Marital Status

Average Spending per Transaction by Marital Status (0=Single, 1=Married):
Marital_Status
0    9265.907619
1    9261.174574
Name: Purchase, dtype: float64

95% CI for Average Spending (Single):   (9248.62, 9283.20)
95% CI for Average Spending (Married): (9240.46, 9281.89)

Overlap in 95% CIs for Marital Status: True
  The confidence intervals for single and married customers overlap significantly.
  We cannot confidently conclude a difference in true average spending based on marital status alone.
  Marketing might not need strong differentiation based solely on marital status for average purchase value.

Q5: Analysis for Age Groups

Average Spending per Transaction by Age Group:
Age_Group
0-17     8933.464640
18-25    9169.663606
26-35    9252.690633
36-50    9295.331743
51+      9463.661678
Name: Purchase, dtype: float64

95% Confidence Intervals for Average Spending by Age Group:
  0-17: (8851.95, 9014.98)
  18-25: (9138.41, 9200.92)
  26-35: (9231.73, 9273.65)
  36-50: (9270.46, 9320.20)
  51+: (9423.17, 9504.16)

Observations on Age Group CIs:
  Overlap between 26-35 and 51+ CIs: False
    Statistically significant difference in average spending between these groups.
  Visual inspection of the boxplot (purchase_vs_age_group_boxplot.png) and CIs suggests
  that while average spending varies slightly across age groups, the distributions and CIs
  show considerable overlap, indicating average spending might not differ dramatically
  or statistically significantly between *all* adjacent age groups.
  However, specific groups might show differences (e.g., potentially 18-50 groups vs. 0-17 or 51+).
  Walmart could explore targeted promotions for age groups showing distinct higher spending patterns,
  but broad strategies might be effective across the main adult age brackets (18-50).
```

```
# --- Final Insights & Recommendations ---
print("\n--- Final Insights & Recommendations ---")

print("\nKey Insights:")
# Correction: Based on the CI calculation (non-overlap), males DO spend significantly more on average.
print("1.  **Gender:** Males show a statistically significantly higher average spending per transaction than females (at 95% confidence)
print("2.  **Marital Status:** The average spending per transaction between single and married customers shows overlapping confidence in
print("3.  **Age:** Average spending varies across age groups. While many adjacent adult groups (18-50) have overlapping confidence inte
print("4.  **City Category:** Visual analysis (boxplots) suggests potential differences in spending distributions based on City Category
print("5.  **Overall Purchase Distribution:** The purchase amount is right-skewed, with most purchases concentrated at lower values but
print("6.  **Sample Size & Confidence:** Larger sample sizes lead to narrower (more precise) confidence intervals. Higher confidence lev

print("\nRecommendations for Walmart:")
# Adjustment: Acknowledge the gender difference but still recommend broad appeal as primary, with potential for *some* targeting.
print("1.  **Primary Broad Appeal Marketing:** While males show statistically higher average spending, the difference might not warrant
print("2.  **Consider Gender-Specific Promotions (Secondary):** Given the statistically significant higher average spend by males, cons
print("3.  **Target High-Value Segments (Beyond Averages):** Explore if specific demographics (combinations of age, occupation, city ca
print("4.  **Focus on Core & High-Spending Age Groups:** The 18-50 age groups are key customer bases. Additionally, the 51+ group shows
print("5.  **Investigate City Category Differences:** Explore why customers in certain city categories might spend more (as suggested by
print("6.  **Personalization Beyond Demographics:** Leverage User_ID and Product_ID data for personalized recommendations based on past
print("7.  **Monitor Trends:** Continuously monitor these metrics over time and across different sales events to see if patterns change.
print("5.  **Personalization Beyond Demographics:** Leverage User_ID and Product_ID data for personalized recommendations based on past
print("6.  **Monitor Trends:** Continuously monitor these metrics over time and across different sales events to see if patterns change.

print("\nAnalysis Complete. Plots saved as PNG files in the current directory.")
```

```
--- Final Insights & Recommendations ---

Key Insights:
1.  **Gender:** Males show a statistically significantly higher average spending per transaction than females (at 95% confidence). T
2.  **Marital Status:** The average spending per transaction between single and married customers shows overlapping confidence inter
3.  **Age:** Average spending varies across age groups. While many adjacent adult groups (18-50) have overlapping confidence interva
4.  **City Category:** Visual analysis (boxplots) suggests potential differences in spending distributions based on City Category (e
5.  **Overall Purchase Distribution:** The purchase amount is right-skewed, with most purchases concentrated at lower values but a l
6.  **Sample Size & Confidence:** Larger sample sizes lead to narrower (more precise) confidence intervals. Higher confidence levels

Recommendations for Walmart:
1.  **Primary Broad Appeal Marketing:** While males show statistically higher average spending, the difference might not warrant cor
2.  **Consider Gender-Specific Promotions (Secondary):** Given the statistically significant higher average spend by males, consider
3.  **Target High-Value Segments (Beyond Averages):** Explore if specific demographics (combinations of age, occupation, city catego
4.  **Focus on Core & High-Spending Age Groups:** The 18-50 age groups are key customer bases. Additionally, the 51+ group shows sig
5.  **Investigate City Category Differences:** Explore why customers in certain city categories might spend more (as suggested by bo
6.  **Personalization Beyond Demographics:** Leverage User_ID and Product_ID data for personalized recommendations based on past pur
7.  **Monitor Trends:** Continuously monitor these metrics over time and across different sales events to see if patterns change.
5.  **Personalization Beyond Demographics:** Leverage User_ID and Product_ID data for personalized recommendations based on past pur
6.  **Monitor Trends:** Continuously monitor these metrics over time and across different sales events to see if patterns change.

Analysis Complete. Plots saved as PNG files in the current directory.
```