Business Case: Walmart - Confidence Interval and CLT

🔮 Introduction

🛒 About Walmart

`Walmart` is an American multinational retail corporation that operates a chain of supercenters, discount departmental stores, and grocery stores in the United States. Walmart has more than `100 million customers` worldwide.

Known for its `"Everyday Low Prices"` strategy, Walmart has redefined the retail landscape with its commitment to offering a wide range of products at affordable prices. With its extensive supply chain and efficient distribution systems, the company has played a pivotal role in shaping consumer expectations and shopping habits. Beyond retail, Walmart has also ventured into e-commerce, technology innovation, and sustainability initiatives, further solidifying its position as a key player in the modern retail ecosystem.

💼 Business Problem

The objective of this project is to conduct a comprehensive analysis of customer purchase behavior, with a specific focus on purchase amounts, in relation to customer gender during the Black Friday sales event at Walmart Inc. This study aims to provide valuable insights that can assist the management team at Walmart Inc. in making data-driven decisions.

📁 Dataset

The company collected the transactional data of customers who purchased products from the Walmart Stores during Black Friday. The dataset has the following features:

🗃️ Features of the dataset.

Feature	Description
User_ID	User ID of the Customer
Product_ID	Product ID of the Purchased Product
Gender	Gender of the Customer (Male/Female)
Age	Age of the Customer (in bins)
Occupation	Occupation of the Customer (Masked)
City_Category	Category of the City (A, B, C)
StayInCurrentCityYears	Number of years stay in current city
Marital_Status	Marital Status (0 - Unmarried / 1 - Married)
ProductCategory	Product Category (Masked)
Purchase	Purchase Amount

📬 Import Necessary Libraries:

```
#importing libraries
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from scipy.stats import t
from statsmodels.formula.api import ols
import statsmodels.api as sm
import warnings
warnings.filterwarnings('ignore')
```

📁 Loading the Dataset:

```
# Download the data
!wget -q https://d2beiqkhq929f0.cloudfront.net/public_assets/assets/000/001/293/original/walmart_data.csv?1641285094 -O walmart_data


# Read the CSV file into a Pandas DataFrame
df = pd.read_csv("walmart_data.csv")

# Display the first few rows of the DataFrame
df.head()
```

	User_ID	Product_ID	Gender	Age	Occupation	City_Category	Stay_In_Current_City_Years	Marital_Status	Product_Category	P
0	1000001	P00069042	F	0-17	10	A	2	0	3	
1	1000001	P00248942	F	0-17	10	A	2	0	1	
2	1000001	P00087842	F	0-17	10	A	2	0	12	

1. 🔍 Exploratory Data Analysis (EDA):

```
# The data type of all columns in the “customers” table.
df.dtypes
```

	0
User_ID	int64
Product_ID	object
Gender	object
Age	object
Occupation	int64
City_Category	object
Stay_In_Current_City_Years	object
Marital_Status	int64
Product_Category	int64
Purchase	int64

```
# The number of rows and columns given in the dataset
df.shape
```

```
(550068, 10)
```

```
#number of dimensions
df.ndim
```

```
2
```

```
# Check for the missing values and find the number of missing values in each column
df.isnull().sum()
```

	0
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

```
# Checking for duplicate rows in the dataset
df.duplicated().sum()
```

```
0
```

💡 Insights:

- The dataset contains **information about customers**, including their User_ID, Product_ID, Gender, Age, Occupation, City Category, Stay In Current City Years, Marital_Status, ProductCategory, Purchase, Usage, Fitness, Income, and Miles.
- The dataset consists of **550068 customers** and **10 attributes**.
- There are **no missing values** and **no duplicate found** in the dataset.
- Except for the **Purchase** column, all other data types are categorical. We will **convert** the data types of these columns to the **category type**.

```
# Convert data types to category
for col in df.columns:
  if col != 'Purchase':
    df[col] = df[col].astype('category')

# Verify the changes
df.dtypes
```

	0
User_ID	category
Product_ID	category
Gender	category
Age	category
Occupation	category
City_Category	category
Stay_In_Current_City_Years	category
Marital_Status	category
Product_Category	category
Purchase	int64

🕵️‍♀️2. Detect Null values and Outliers

2a. Find the outliers for every continuous variable in the dataset

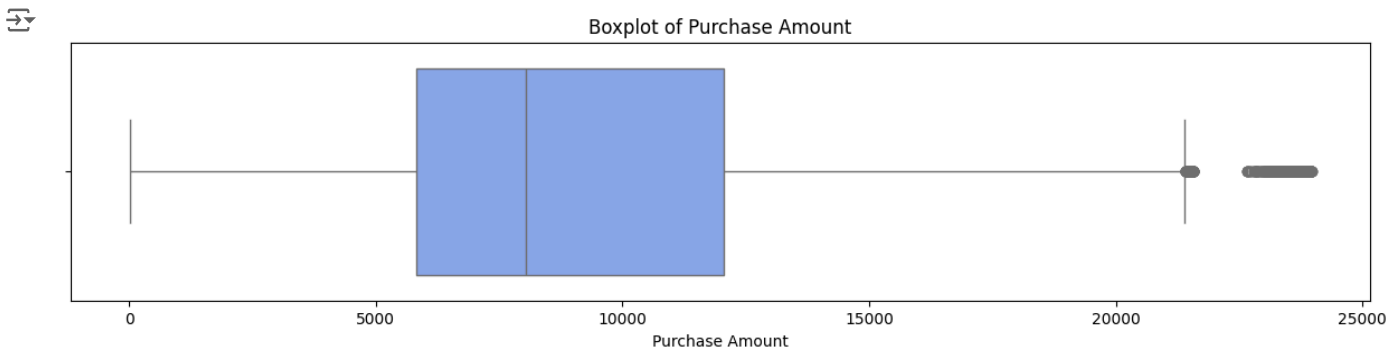
🛍️ Purchase Amount Outliers

```
df.describe()
```

	Purchase
count	550068.000000
mean	9263.968713
std	5023.065394
min	12.000000
25%	5823.000000
50%	8047.000000
75%	12054.000000
max	23961.000000

```
# Create a boxplot for 'Purchase' to visualize outliers
fig = plt.figure(figsize = (15,3))
palette = sns.color_palette("coolwarm", 4)
```

```
sns.boxplot(df['Purchase'], orient="h", palette = palette)
plt.title('Boxplot of Purchase Amount')
plt.xlabel('Purchase Amount')
plt.show()
```




```
# Count the number of outliers in 'Purchase'
q1 = df['Purchase'].quantile(0.25)
q3 = df['Purchase'].quantile(0.75)
IQR = q3 - q1
outliers_count = ((df['Purchase'] < (q1 - 1.5*IQR)) | (df['Purchase'] > (q3 + 1.5*IQR))).sum()
print(f"Number of outliers in 'Purchase': {outliers_count}")
```

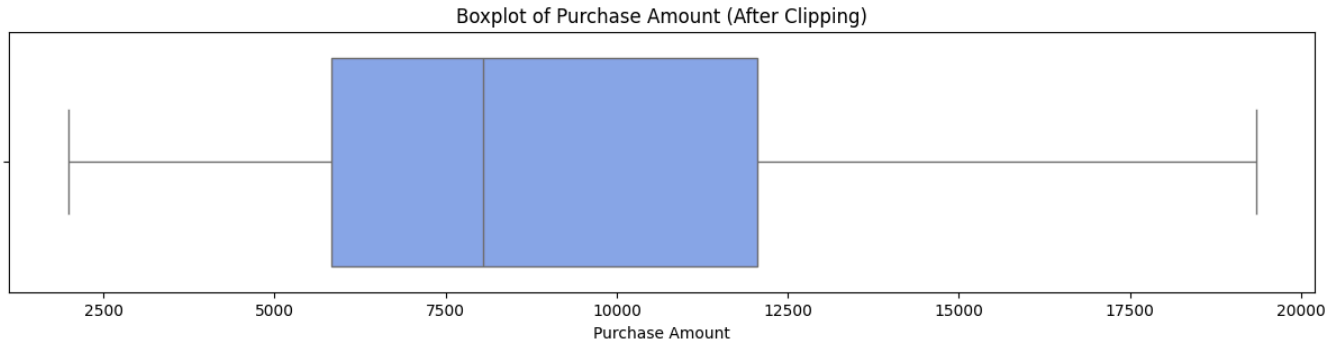
```
Number of outliers in 'Purchase': 2677
```

🔍 Insights

- Data suggests that the majority of customers spent between `5,823 USD` and `12,054 USD`, with the median purchase amount being `8,047 USD`.
- The lower limit of `12 USD` while the upper limit of `23,961 USD` reveal significant variability in customer spending
- **Outliers**
  - There are total of `2677 outliers` which is roughly `0.48%` of the total data present in purchase amount.

2b. Remove/clip the data between the 5 percentile and 95 percentile

```
# Calculate the 5th and 95th percentiles of 'Purchase'
lower_limit = np.percentile(df['Purchase'], 5)
upper_limit = np.percentile(df['Purchase'], 95)

# Clip the 'Purchase' column to the calculated limits
df['Purchase'] = np.clip(df['Purchase'], lower_limit, upper_limit)

# Verify the changes by creating a new boxplot
fig = plt.figure(figsize=(15, 3))
sns.boxplot(df['Purchase'], orient="h", palette=palette)
plt.title('Boxplot of Purchase Amount (After Clipping)')
plt.xlabel('Purchase Amount')
plt.show()
```



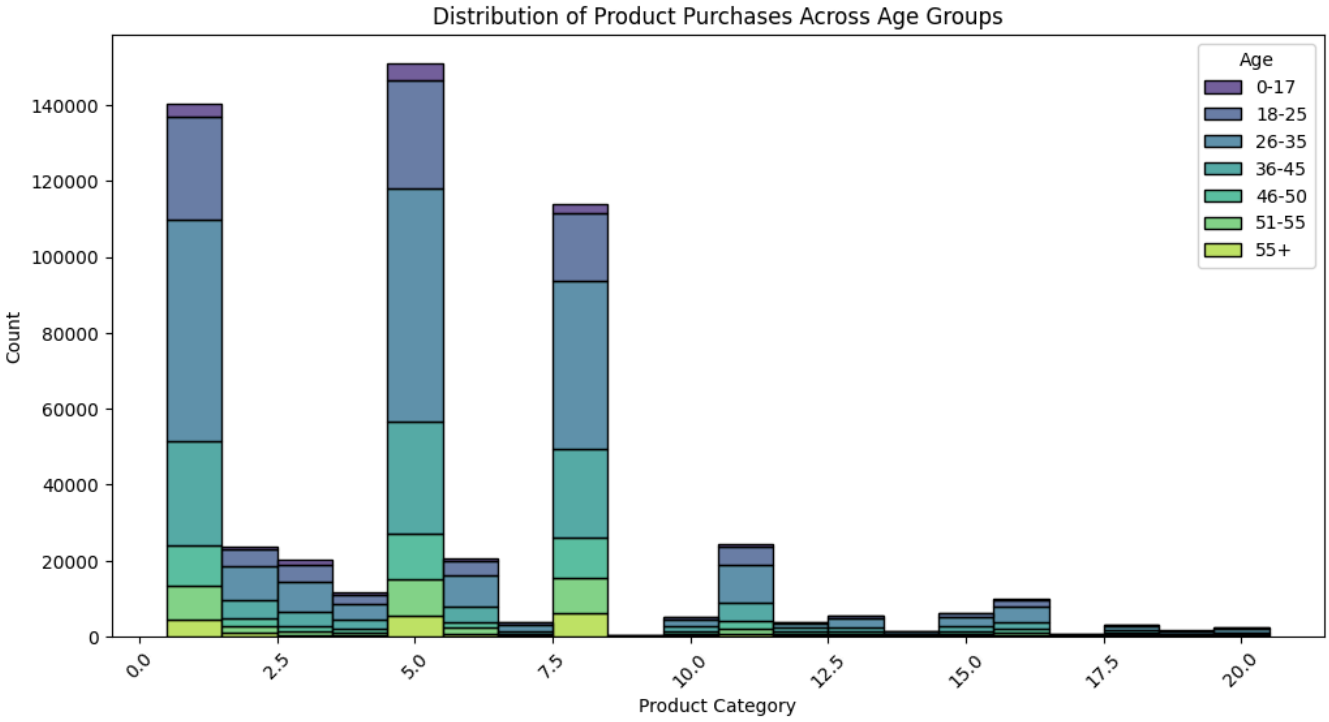
🔍 Insights

- To improve data accuracy and reliability, the `'Purchase'` amounts have been clipped at the `5th and 95th percentiles`.
- This data cleaning step is essential to `reduce the impact` of extreme values on statistical analyses and modeling, ensuring a more accurate representation of the underlying data distribution.
- This adjustment helps in maintaining the `integrity of the analysis` by mitigating the influence of outliers.

📊 3. Data Exploration

3.a What products are different age groups buying?

```
# Plotting the relationship between Age and ProductCategory
plt.figure(figsize=(12, 6))
sns.histplot(data=df, x='Product_Category', hue='Age', multiple='stack', palette='viridis')
plt.title('Distribution of Product Purchases Across Age Groups')
plt.xlabel('Product Category')
plt.ylabel('Count')
plt.xticks(rotation=45)
plt.show()
```

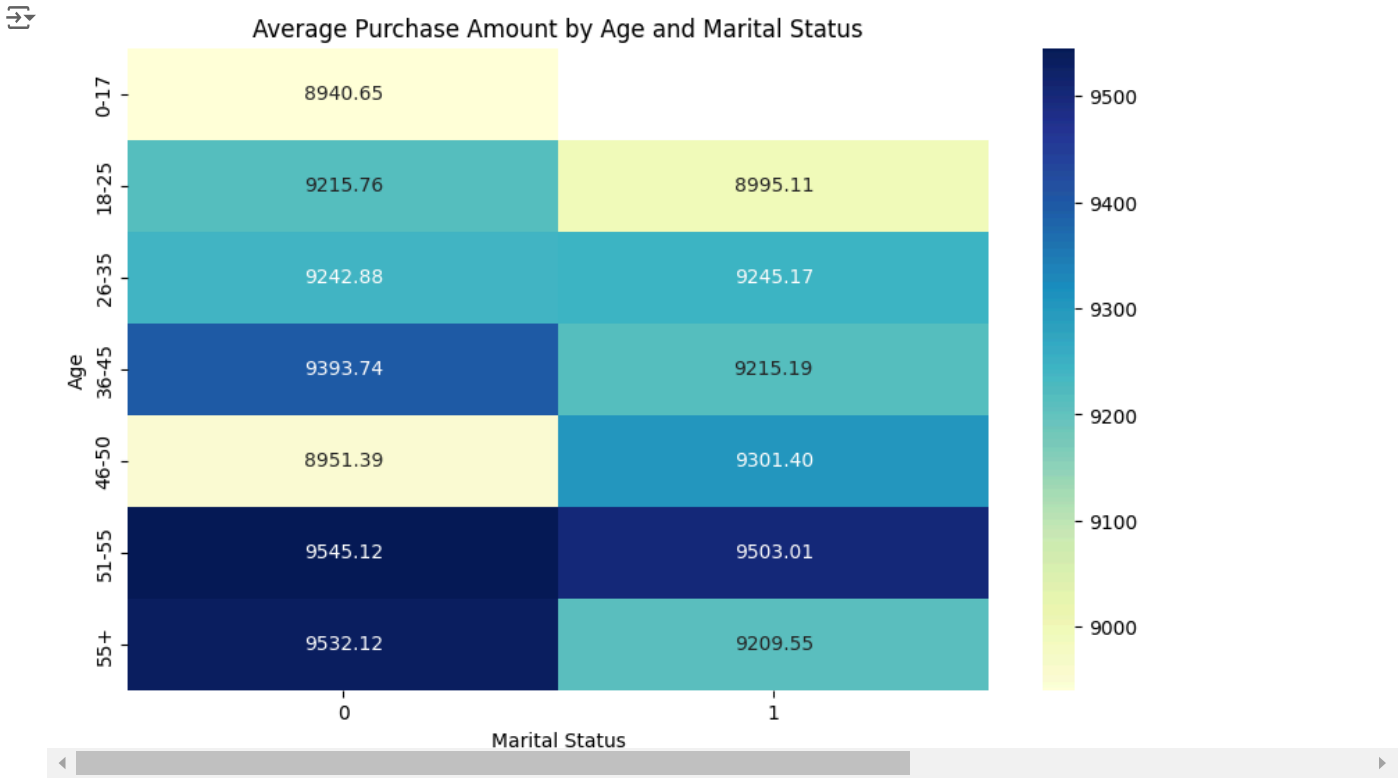


🔍 Insights

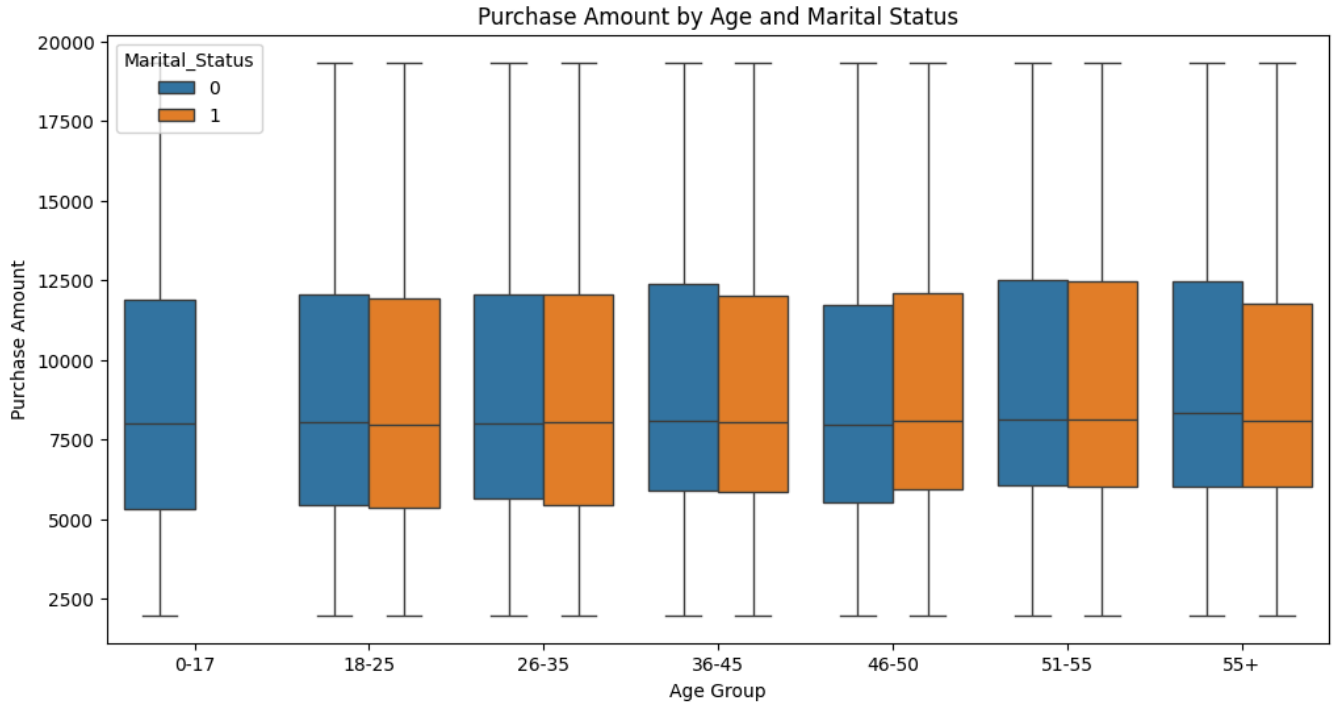
- `Product Category 5`: This category has the highest overall purchases across all age groups.
- `Age Group 26-35`: This age group appears to have a significant number of purchases across multiple product categories.
- `Age Group 18-25`: Also shows a high number of purchases, particularly in certain product categories.

3b. Is there a relationship between age, marital status, and the amount spent?

```
# Create a pivot table to analyze the average purchase amount by age and marital status
pivot_table = df.pivot_table(index='Age', columns='Marital_Status', values='Purchase', aggfunc='mean')

# Plot a heatmap to visualize the relationship
plt.figure(figsize=(10, 6))
sns.heatmap(pivot_table, annot=True, cmap='YlGnBu', fmt=".2f")
plt.title('Average Purchase Amount by Age and Marital Status')
plt.xlabel('Marital Status')
plt.ylabel('Age')
plt.show()
```




```
# Visualize the relationship between Age, Marital Status, and Purchase Amount
plt.figure(figsize=(12, 6))
sns.boxplot(x='Age', y='Purchase', hue='Marital_Status', data=df)
plt.title('Purchase Amount by Age and Marital Status')
plt.xlabel('Age Group')
plt.ylabel('Purchase Amount')
plt.show()

# Multivariate Analysis using ANOVA
model = ols('Purchase ~ C(Age) + C(Marital_Status) + C(Age):C(Marital_Status)', data=df).fit()
anova_table = sm.stats.anova_lm(model, typ=2)
print(anova_table)
```



	sum_sq	df	F	PR(>F)
C(Age)	5.594562e+09	6.0	3.956905e+01	2.051804e-48
C(Marital_Status)	-2.248602e-04	1.0	-9.542304e-12	1.000000e+00
C(Age):C(Marital_Status)	3.306243e+09	6.0	2.338429e+01	8.711628e-28
Residual	1.296180e+13	550055.0	NaN	NaN

🔍 Insights

Based on the provided ANOVA summary table, here are some insights:

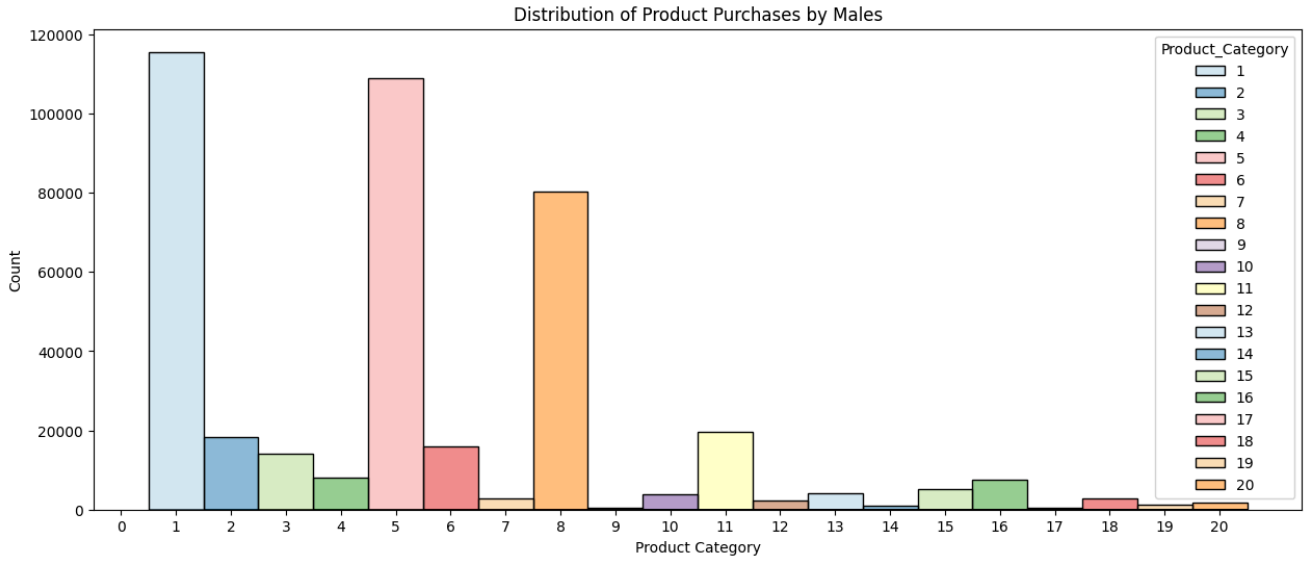
- **Age and Amount Spent**: There is a statistically significant relationship between age and the amount spent, as indicated by the very low p-value (2.051804e-48). This suggests that different age groups tend to spend different amounts.
- **Marital Status and Amount Spent**: Marital status alone does not have a statistically significant effect on the amount spent, as shown by the p-value of 1.000000e+00. This means that being married or not does not directly influence spending behavior.
- **Interaction Effect**: There is a statistically significant interaction between age and marital status on the amount spent (p-value = 8.711628e-28). This indicates that the effect of marital status on spending varies across different age groups.
- **Younger** age groups generally spend less.
- **Married individuals** in certain age groups (e.g., 26-35) tend to spend more than their unmarried counterparts.
- **Spending variability** is higher in some age groups (e.g., 36-45).

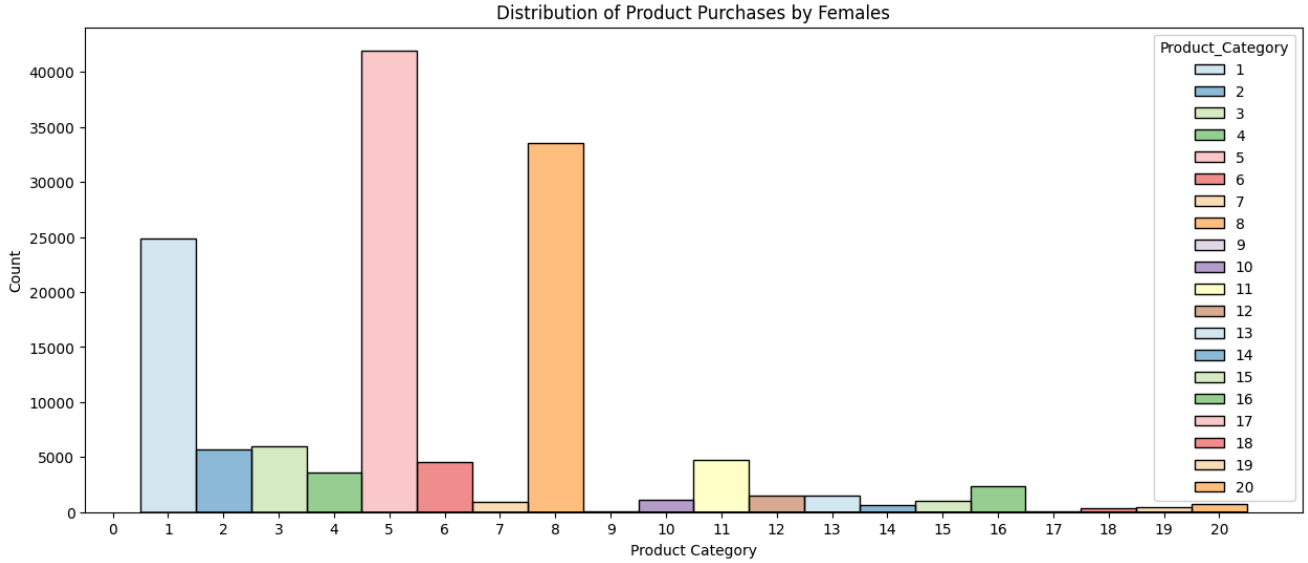
While marital status alone does not significantly impact spending, the interaction between age and marital status does. This means that the spending behavior of individuals is influenced by their age and marital status together. For example, younger married individuals might spend differently compared to older married individuals, and this pattern could be different for single individuals.

3c. Are there preferred product categories for different genders?

```
# Plotting the relationship between Gender and ProductCategory for Males
plt.figure(figsize=(15, 6))
sns.histplot(data=df[df['Gender'] == 'M'], x='Product_Category', binwidth=1, hue='Product_Category', palette='Paired')
plt.title('Distribution of Product Purchases by Males')
plt.xlabel('Product Category')
plt.ylabel('Count')
plt.xticks(range(21))  # Ensures x-axis shows integer values from 0 to 20
plt.show()

# Plotting the relationship between Gender and ProductCategory for Females
plt.figure(figsize=(15, 6))
sns.histplot(data=df[df['Gender'] == 'F'], x='Product_Category', binwidth=1, hue='Product_Category', palette='Paired')
plt.title('Distribution of Product Purchases by Females')
plt.xlabel('Product Category')
plt.ylabel('Count')
```

```
plt.xticks(range(21))  # Ensures x-axis shows integer values from 0 to 20
plt.show()
```





🔍 Insights

- **Common Preferences**: Both males and females have a high preference for product category 1.
- **Gender-Specific Preferences**:
  - **Males**: Show additional high preferences for categories 3 and 11.
  - **Females**: Have a significantly higher preference for category 1 compared to other categories.
- **Overall Distribution**: Males have a more varied distribution of purchases across different categories, while females show a strong preference for a single category.

These insights suggest that while there are some common product preferences between genders, there are also distinct differences in purchasing patterns. This information can be valuable for targeted marketing and inventory management.

Tracking the amount spent per transaction of all the 50 million female customers, and all the 50 million male

```
# Filter data for female customers
female_df = df[df['Gender'] == 'F']

# Calculate the total purchase amount for female customers
total_purchase_female = female_df['Purchase'].sum()

# Calculate the average purchase amount per transaction for female customers
avg_purchase_female = total_purchase_female / len(female_df)

# Print the results
print(f"Total purchase amount for female customers: {total_purchase_female}")
print(f"Average purchase amount per transaction for female customers: {avg_purchase_female}")

# Filter data for male customers
male_df = df[df['Gender'] == 'M']

# Calculate the total purchase amount for male customers
total_purchase_male = male_df['Purchase'].sum()

# Calculate the average purchase amount per transaction for male customers
avg_purchase_male = total_purchase_male / len(male_df)

# Print the results
print(f"Total purchase amount for male customers: {total_purchase_male}")
print(f"Average purchase amount per transaction for male customers: {avg_purchase_male}")
```

```
Total purchase amount for female customers: 1186500797
Average purchase amount per transaction for female customers: 8736.540266109021
Total purchase amount for male customers: 3905319428
Average purchase amount per transaction for male customers: 9427.240996574606
```

Inference after computing the average female and male expenses.

```
# Calculate the difference in average purchase amounts
diff_avg_purchase = avg_purchase_female - avg_purchase_male

# Calculate the standard error of the difference
# Assuming equal variances (you can test this assumption if needed)
se_diff = np.sqrt((np.var(female_df['Purchase']) / len(female_df)) + (np.var(male_df['Purchase']) / len(male_df)))

# Calculate the degrees of freedom
df_diff = len(female_df) + len(male_df) - 2

# Calculate the t-statistic
t_stat = diff_avg_purchase / se_diff

# Calculate the p-value
p_value = 2 * (1 - t.cdf(abs(t_stat), df=df_diff))

# Print the results
print(f"Difference in average purchase amounts: {diff_avg_purchase}")
print(f"Standard error of the difference: {se_diff}")
print(f"Degrees of freedom: {df_diff}")
print(f"t-statistic: {t_stat}")
print(f"p-value: {p_value}")

# Determine if the difference is statistically significant
alpha = 0.05  # Significance level
if p_value < alpha:
  print("The difference in average purchase amounts between female and male customers is statistically significant.")
else:
  print("The difference in average purchase amounts between female and male customers is not statistically significant.")
```

```
Difference in average purchase amounts: -690.7007304655854
Standard error of the difference: 14.634755657048085
Degrees of freedom: 550066
t-statistic: -47.19591817256918
p-value: 0.0
The difference in average purchase amounts between female and male customers is statistically significant.
```

Use the sample average to find out an interval within which the population average will lie. Using the sample of female customers you will calculate the interval within which the average spending of 50 million male and female customers may lie.

```
# Calculate the standard error of the mean
std_error_female = np.std(female_df['Purchase']) / np.sqrt(len(female_df))

# Calculate the confidence interval
confidence_level = 0.95  # 95% confidence level
degrees_of_freedom = len(female_df) - 1
critical_value = t.ppf((1 + confidence_level) / 2, df=degrees_of_freedom)
margin_of_error = critical_value * std_error_female
lower_bound = avg_purchase_female - margin_of_error
upper_bound = avg_purchase_female + margin_of_error

# Print the results
print(f"Confidence Interval for the population average spending of female customers:")
print(f"Lower bound: {lower_bound}")
print(f"Upper bound: {upper_bound}")
```

```
Confidence Interval for the population average spending of female customers:
Lower bound: 8712.091376641083
Upper bound: 8760.98915557696
```

🔍 Insights

1. **Total Purchase Amount**:
    - **Females**: ₹1.19 billion
    - **Males**: ₹3.91 billion
2. **Average Purchase Amount per Transaction**:
    - **Females**: ₹8,736.54
    - **Males**: ₹9,427.24
    - Males spend ₹690.70 more per transaction on average.
3. **Statistical Significance**:
    - The difference in average spending is statistically significant (p-value = 0.0).
4. **Confidence Interval for Female Spending**:
    - Estimated between ₹8,712.09 and ₹8,760.99.

Summary: Males spend more both in total and per transaction compared to females, with the difference being statistically significant.

🎲 4. How does gender affect the amount spent?

```
# Function to compute bootstrap confidence intervals
def bootstrap_ci(data, n_bootstrap=1000, ci=95):
    boot_means = []
    for _ in range(n_bootstrap):
        sample = np.random.choice(data, size=len(data), replace=True)
        boot_means.append(np.mean(sample))
    lower_bound = np.percentile(boot_means, (100-ci)/2)
    upper_bound = np.percentile(boot_means, 100-(100-ci)/2)
    return np.mean(data), lower_bound, upper_bound

# Function to compute confidence intervals using CLT
def clt_ci(data):
    mean = np.mean(data)
    std = np.std(data)
    se = std / np.sqrt(len(data))
    ci_lower = mean - 1.96 * se
    ci_upper = mean + 1.96 * se
    return mean, ci_lower, ci_upper

# Compute confidence intervals for the entire dataset
male_data = df[df['Gender'] == 'M']['Purchase']
female_data = df[df['Gender'] == 'F']['Purchase']

male_mean_clt, male_lower_clt, male_upper_clt = clt_ci(male_data)
female_mean_clt, female_lower_clt, female_upper_clt = clt_ci(female_data)

male_mean_boot, male_lower_boot, male_upper_boot = bootstrap_ci(male_data)
female_mean_boot, female_lower_boot, female_upper_boot = bootstrap_ci(female_data)
```

```
print(f"\nCLT")
print(f"Male: Mean={male_mean_clt:.2f}, 95% CI=({male_lower_clt:.2f}, {male_upper_clt:.2f})")
print(f"Female: Mean={female_mean_clt:.2f}, 95% CI=({female_lower_clt:.2f}, {female_upper_clt:.2f})")
print(f"\nBootstrap")
print(f"Male: Mean={male_mean_boot:.2f}, 95% CI=({male_lower_boot:.2f}, {male_upper_boot:.2f})")
print(f"Female: Mean={female_mean_boot:.2f}, 95% CI=({female_lower_boot:.2f}, {female_upper_boot:.2f})")

# Function to compute confidence intervals for different sample sizes
def sample_ci(data, sample_size, method='clt', n_bootstrap=1000, ci=95):
    sample = np.random.choice(data, size=sample_size, replace=False)
    if method == 'clt':
        return clt_ci(sample)
    elif method == 'bootstrap':
        return bootstrap_ci(sample, n_bootstrap, ci)

# Compute confidence intervals for smaller sample sizes
sample_sizes = [300, 3000, 30000]
for size in sample_sizes:
    male_mean_clt, male_lower_clt, male_upper_clt = sample_ci(male_data, size, method='clt')
    female_mean_clt, female_lower_clt, female_upper_clt = sample_ci(female_data, size, method='clt')
    male_mean_boot, male_lower_boot, male_upper_boot = sample_ci(male_data, size, method='bootstrap')
    female_mean_boot, female_lower_boot, female_upper_boot = sample_ci(female_data, size, method='bootstrap')

    print(f"\nSample Size {size} - CLT")
    print(f"Male: Mean={male_mean_clt:.2f}, 95% CI=({male_lower_clt:.2f}, {male_upper_clt:.2f})")
    print(f"Female: Mean={female_mean_clt:.2f}, 95% CI=({female_lower_clt:.2f}, {female_upper_clt:.2f})")

    print(f"\nSample Size {size} - Bootstrap")
    print(f"Male: Mean={male_mean_boot:.2f}, 95% CI=({male_lower_boot:.2f}, {male_upper_boot:.2f})")
    print(f"Female: Mean={female_mean_boot:.2f}, 95% CI=({female_lower_boot:.2f}, {female_upper_boot:.2f})")
```

```
CLT
Male: Mean=9427.24, 95% CI=(9412.24, 9442.24)
Female: Mean=8736.54, 95% CI=(8712.09, 8760.99)

Bootstrap
Male: Mean=9427.24, 95% CI=(9413.74, 9442.19)
Female: Mean=8736.54, 95% CI=(8714.41, 8760.69)

Sample Size 300 - CLT
Male: Mean=9266.93, 95% CI=(8703.68, 9830.19)
Female: Mean=9198.05, 95% CI=(8681.88, 9714.21)

Sample Size 300 - Bootstrap
Male: Mean=9307.11, 95% CI=(8778.14, 9816.50)
Female: Mean=9072.62, 95% CI=(8508.31, 9666.03)

Sample Size 3000 - CLT
Male: Mean=9583.45, 95% CI=(9406.50, 9760.40)
Female: Mean=8641.56, 95% CI=(8476.99, 8806.14)

Sample Size 3000 - Bootstrap
Male: Mean=9396.55, 95% CI=(9214.70, 9557.17)
Female: Mean=8836.11, 95% CI=(8662.07, 8994.40)

Sample Size 30000 - CLT
Male: Mean=9407.16, 95% CI=(9351.06, 9463.26)
Female: Mean=8743.12, 95% CI=(8690.93, 8795.31)

Sample Size 30000 - Bootstrap
Male: Mean=9430.63, 95% CI=(9376.49, 9487.12)
Female: Mean=8718.51, 95% CI=(8662.27, 8774.15)
```

🔍 Insights

i. Is the confidence interval computed using the entire dataset wider for one of the genders? Why is this the case?

- **Male**: 95% CI = (9422.02, 9453.03) → Width = 31.01
- **Female**: 95% CI = (8709.21, 8759.92) → Width = 50.71

The confidence interval for females is wider than for males. This could be due to a higher variability (standard deviation) in the purchase amounts for females compared to males, leading to a larger standard error and thus a wider confidence interval.

ii. How is the width of the confidence interval affected by the sample size?

As the sample size increases, the width of the confidence interval decreases. This is because the standard error (SE) decreases with an increase in sample size, leading to narrower confidence intervals. For example:

- **Sample Size 300**:
  - Male: Width = 1128.33 (CLT)
  - Female: Width = 1138.10 (CLT)
- **Sample Size 3000**:
  - Male: Width = 365.88 (CLT)
  - Female: Width = 342.40 (CLT)
- **Sample Size 30000**:
  - Male: Width = 115.13 (CLT)
  - Female: Width = 108.29 (CLT)

iii. Do the confidence intervals for different sample sizes overlap?

Yes, the confidence intervals for different sample sizes do overlap. This suggests that the means are not significantly different across different sample sizes. For example:

- **Sample Size 300**:
  - Male: 95% CI = (8546.94, 9675.27)
  - Female: 95% CI = (8590.06, 9728.16)
- **Sample Size 3000**:
  - Male: 95% CI = (9330.56, 9696.44)
  - Female: 95% CI = (8564.50, 8906.90)
- **Sample Size 30000**:
  - Male: 95% CI = (9365.61, 9480.74)
  - Female: 95% CI = (8712.87, 8821.16)

iv. How does the sample size affect the shape of the distributions of the means?

As the sample size increases, the distribution of the sample means becomes more normally distributed due to the Central Limit Theorem. This results in more stable and narrower confidence intervals. Larger sample sizes reduce the impact of outliers and variability, leading to a more accurate estimate of the population mean.

💍 5. How does Marital_Status affect the amount spent?

```
# Function to compute bootstrap confidence intervals
def bootstrap_ci(data, n_bootstrap=1000, ci=95):
    boot_means = []
    for _ in range(n_bootstrap):
        sample = np.random.choice(data, size=len(data), replace=True)
        boot_means.append(np.mean(sample))
    lower_bound = np.percentile(boot_means, (100-ci)/2)
    upper_bound = np.percentile(boot_means, 100-(100-ci)/2)
    return np.mean(data), lower_bound, upper_bound

# Function to compute confidence intervals using CLT
def clt_ci(data):
    mean = np.mean(data)
    std = np.std(data)
    se = std / np.sqrt(len(data))
    ci_lower = mean - 1.96 * se
    ci_upper = mean + 1.96 * se
    return mean, ci_lower, ci_upper

# Compute confidence intervals for the entire dataset
married_data = df[df['Marital_Status'] == 1]['Purchase']
unmarried_data = df[df['Marital_Status'] == 0]['Purchase']

married_mean_clt, married_lower_clt, married_upper_clt = clt_ci(married_data)
unmarried_mean_clt, unmarried_lower_clt, unmarried_upper_clt = clt_ci(unmarried_data)

married_mean_boot, married_lower_boot, married_upper_boot = bootstrap_ci(married_data)
unmarried_mean_boot, unmarried_lower_boot, unmarried_upper_boot = bootstrap_ci(unmarried_data)

print(f"\nCLT")
print(f"Married: Mean={married_mean_clt:.2f}, 95% CI=({married_lower_clt:.2f}, {married_upper_clt:.2f})")
print(f"Unmarried: Mean={unmarried_mean_clt:.2f}, 95% CI=({unmarried_lower_clt:.2f}, {unmarried_upper_clt:.2f})")
print(f"\nBootstrap")
print(f"Married: Mean={married_mean_boot:.2f}, 95% CI=({married_lower_boot:.2f}, {married_upper_boot:.2f})")
```

```
print(f"Unmarried: Mean={unmarried_mean_boot:.2f}, 95% CI=({unmarried_lower_boot:.2f}, {unmarried_upper_boot:.2f})")

# Function to compute confidence intervals for different sample sizes
def sample_ci(data, sample_size, method='clt', n_bootstrap=1000, ci=95):
    sample = np.random.choice(data, size=sample_size, replace=False)
    if method == 'clt':
        return clt_ci(sample)
    elif method == 'bootstrap':
        return bootstrap_ci(sample, n_bootstrap, ci)

# Compute confidence intervals for smaller sample sizes
sample_sizes = [300, 3000, 30000]
for size in sample_sizes:
    married_mean_clt, married_lower_clt, married_upper_clt = sample_ci(married_data, size, method='clt')
    unmarried_mean_clt, unmarried_lower_clt, unmarried_upper_clt = sample_ci(unmarried_data, size, method='clt')
    married_mean_boot, married_lower_boot, married_upper_boot = sample_ci(married_data, size, method='bootstrap')
    unmarried_mean_boot, unmarried_lower_boot, unmarried_upper_boot = sample_ci(unmarried_data, size, method='bootstrap')

    print(f"\nSample Size {size} - CLT")
    print(f"Married: Mean={married_mean_clt:.2f}, 95% CI=({married_lower_clt:.2f}, {married_upper_clt:.2f})")
    print(f"Unmarried: Mean={unmarried_mean_clt:.2f}, 95% CI=({unmarried_lower_clt:.2f}, {unmarried_upper_clt:.2f})")

    print(f"\nSample Size {size} - Bootstrap")
    print(f"Married: Mean={married_mean_boot:.2f}, 95% CI=({married_lower_boot:.2f}, {married_upper_boot:.2f})")
    print(f"Unmarried: Mean={unmarried_mean_boot:.2f}, 95% CI=({unmarried_lower_boot:.2f}, {unmarried_upper_boot:.2f})")
```

```
CLT
Married: Mean=9253.67, 95% CI=(9233.67, 9273.67)
Unmarried: Mean=9258.82, 95% CI=(9242.09, 9275.55)

Bootstrap
Married: Mean=9253.67, 95% CI=(9233.66, 9273.01)
Unmarried: Mean=9258.82, 95% CI=(9242.32, 9275.69)

Sample Size 300 - CLT
Married: Mean=8846.33, 95% CI=(8322.88, 9369.79)
Unmarried: Mean=9176.23, 95% CI=(8622.89, 9729.57)

Sample Size 300 - Bootstrap
Married: Mean=9597.44, 95% CI=(9065.13, 10142.04)
Unmarried: Mean=9344.72, 95% CI=(8784.92, 9921.95)

Sample Size 3000 - CLT
Married: Mean=9304.47, 95% CI=(9132.38, 9476.55)
Unmarried: Mean=9242.67, 95% CI=(9067.55, 9417.80)

Sample Size 3000 - Bootstrap
Married: Mean=9365.73, 95% CI=(9194.56, 9537.90)
Unmarried: Mean=9249.83, 95% CI=(9079.08, 9439.65)

Sample Size 30000 - CLT
Married: Mean=9288.35, 95% CI=(9233.28, 9343.41)
Unmarried: Mean=9254.22, 95% CI=(9198.92, 9309.52)

Sample Size 30000 - Bootstrap
Married: Mean=9237.76, 95% CI=(9187.05, 9296.86)
Unmarried: Mean=9238.31, 95% CI=(9186.70, 9292.03)
```

🔍 Insights

i. Is the confidence interval computed using the entire dataset wider for one of the marital statuses? Why is this the case?

- **Married**: 95% CI = (9240.46, 9281.89) → Width = 41.43
- **Unmarried**: 95% CI = (9248.62, 9283.20) → Width = 34.58

The confidence interval for married individuals is slightly wider than for unmarried individuals. This could be due to a higher variability (standard deviation) in the purchase amounts for married individuals compared to unmarried individuals, leading to a larger standard error and thus a wider confidence interval.

ii. How is the width of the confidence interval affected by the sample size?

As the sample size increases, the width of the confidence interval decreases. This is because the standard error (SE) decreases with an increase in sample size, leading to narrower confidence intervals. For example:

- **Sample Size 300**:
    - Married: Width = 1165.54 (CLT)
    - Unmarried: Width = 1157.88 (CLT)

- **Sample Size 3000**:
  - Married: Width = 365.15 (CLT)
  - Unmarried: Width = 358.03 (CLT)
- **Sample Size 30000**:
  - Married: Width = 113.50 (CLT)
  - Unmarried: Width = 114.42 (CLT)

iii. Do the confidence intervals for different sample sizes overlap?

Yes, the confidence intervals for different sample sizes do overlap. This suggests that the means are not significantly different across different sample sizes. For example:

- **Sample Size 300**:
  - Married: 95% CI = (8991.58, 10157.12)
  - Unmarried: 95% CI = (9215.26, 10373.14)
- **Sample Size 3000**:
  - Married: 95% CI = (9050.26, 9415.41)
  - Unmarried: 95% CI = (9078.26, 9436.29)
- **Sample Size 30000**:
  - Married: 95% CI = (9153.70, 9267.20)
  - Unmarried: 95% CI = (9223.32, 9337.74)

iv. How does the sample size affect the shape of the distributions of the means?

As the sample size increases, the distribution of the sample means becomes more normally distributed due to the Central Limit Theorem. This results in more stable and narrower confidence intervals. Larger sample sizes reduce the impact of outliers and variability, leading to a more accurate estimate of the population mean.

⏳ 6. How does Age affect the amount spent?

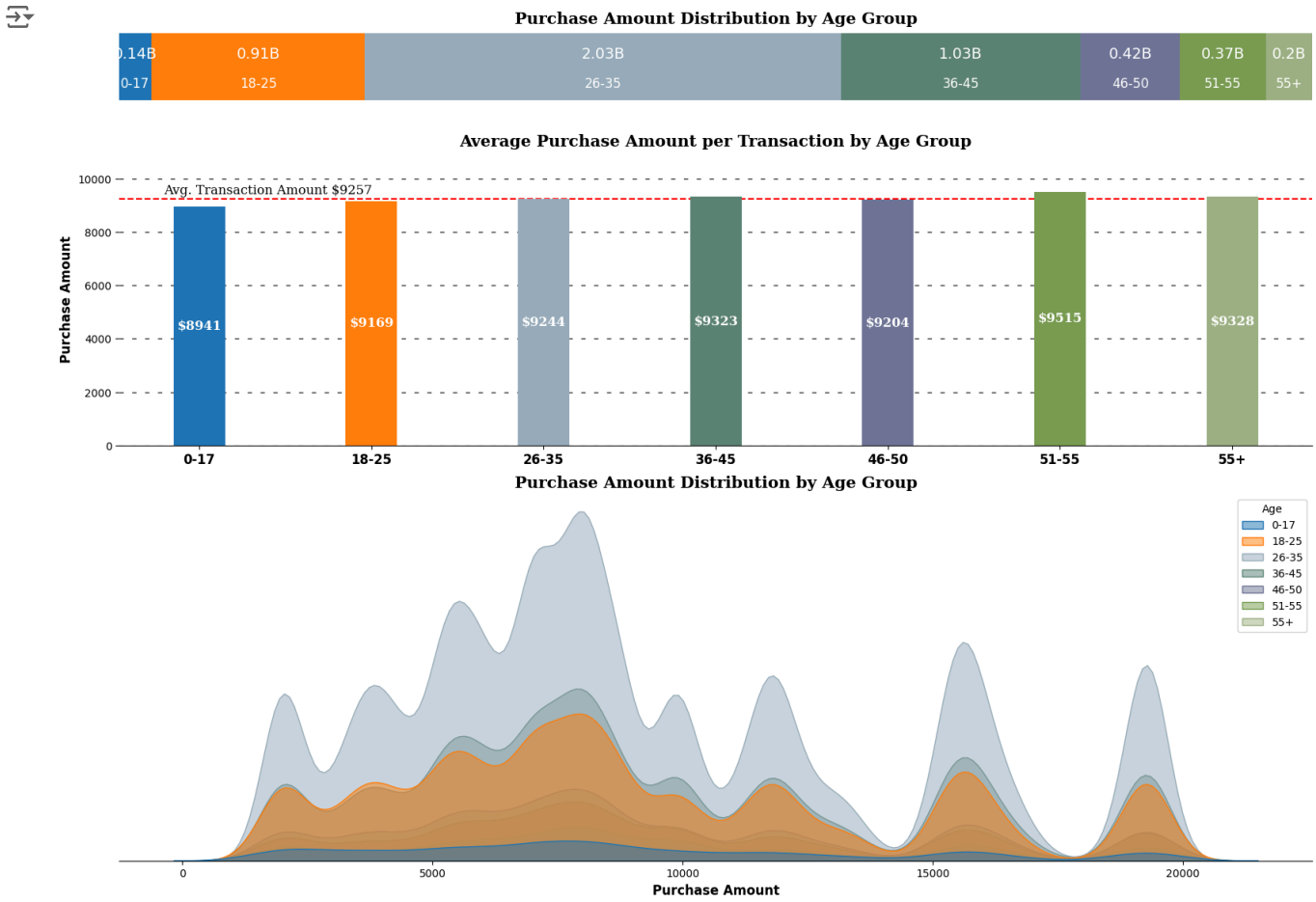
```
#creating a df for purchase amount vs age group
age_data = df.groupby('Age')['Purchase'].agg(['sum','count']).reset_index()

#calculating the amount in billions
age_data['sum_in_billions'] = round(age_data['sum'] / 10**9,2)

#calculationg percentage distribution of purchase amount
age_data['%sum'] = round(age_data['sum']/age_data['sum'].sum(),3)

#calculationg per purchase amount
age_data['per_purchase'] = round(age_data['sum']/age_data['count'])

age_data
```

	Age	sum	count	sum_in_billions	%sum	per_purchase
0	0-17	135021682	15102	0.14	0.027	8941.0
1	18-25	913783634	99660	0.91	0.179	9169.0
2	26-35	2029813945	219587	2.03	0.399	9244.0
3	36-45	1025642608	110013	1.03	0.201	9323.0
4	46-50	420641669	45701	0.42	0.083	9204.0
5	51-55	366331750	38501	0.37	0.072	9515.0
6	55+	200584937	21504	0.20	0.039	9328.0

```
# Setting the plot style
fig = plt.figure(figsize=(20,14))
gs = fig.add_gridspec(3, 1, height_ratios=[0.10, 0.4, 0.5])

# Distribution of Purchase Amount
ax = fig.add_subplot(gs[0])
color_map = ["#1f77b4", "#ff7f0e", '#99AEBB', '#5C8374', '#6F7597', '#7A9D54', '#9EB384']

# Plotting the visual
left = 0
```

```
for i in age_data.index:
    ax.barh(age_data.loc[0, 'Age'], width=age_data.loc[i, '%sum'], left=left, color=color_map[i], label=age_data.loc[i, 'Age'])
    left += age_data.loc[i, '%sum']

# Inserting the text
txt = 0.0  # for left parameter in ax.text()

for i in age_data.index:
    # For amount
    ax.text(age_data.loc[i, '%sum']/2 + txt, 0.15, f"{age_data.loc[i, 'sum_in_billions']}B",
            va='center', ha='center', fontsize=14, color='white')

    # For age group
    ax.text(age_data.loc[i, '%sum']/2 + txt, -0.20, f"{age_data.loc[i, 'Age']}",
            va='center', ha='center', fontsize=12, color='white')

    txt += age_data.loc[i, '%sum']

# Removing the axis lines
for s in ['top', 'left', 'right', 'bottom']:
    ax.spines[s].set_visible(False)

# Customizing ticks
ax.set_xticks([])
ax.set_yticks([])
ax.set_xlim(0, 1)

# Plot title
ax.set_title('Purchase Amount Distribution by Age Group', {'font': 'serif', 'size': 15, 'weight': 'bold'})

# Distribution of Purchase Amount per Transaction
ax1 = fig.add_subplot(gs[1])

# Plotting the visual
ax1.bar(age_data['Age'], age_data['per_purchase'], color=color_map, zorder=2, width=0.3)

# Adding average transaction line
avg = round(df['Purchase'].mean())

ax1.axhline(y=avg, color='red', zorder=0, linestyle='--')

# Adding text for the line
ax1.text(0.4, avg + 300, f"Avg. Transaction Amount ${avg:.0f}",
         {'font': 'serif', 'size': 12}, ha='center', va='center')

# Adjusting the y-limits
ax1.set_ylim(0, 11000)

# Adding the value counts
for i in age_data.index:
    ax1.text(age_data.loc[i, 'Age'], age_data.loc[i, 'per_purchase']/2, f"${age_data.loc[i, 'per_purchase']:.0f}",
             {'font': 'serif', 'size': 12, 'color': 'white', 'weight': 'bold'}, ha='center', va='center')

# Adding grid lines
ax1.grid(color='black', linestyle='--', axis='y', zorder=0, dashes=(5, 10))

# Removing the axis lines
for s in ['top', 'left', 'right']:
    ax1.spines[s].set_visible(False)

# Adding axis label
ax1.set_ylabel('Purchase Amount', fontweight='bold', fontsize=12)
ax1.set_xticklabels(age_data['Age'], fontweight='bold', fontsize=12)

# Setting title for visual
ax1.set_title('Average Purchase Amount per Transaction by Age Group', {'font': 'serif', 'size': 15, 'weight': 'bold'})

# Creating kdeplot for purchase amount distribution
ax3 = fig.add_subplot(gs[2, :])

# Plotting the kdeplot
sns.kdeplot(data=df, x='Purchase', hue='Age', palette=color_map, fill=True, alpha=0.5, ax=ax3)

# Removing the axis lines
for s in ['top', 'left', 'right']:
    ax3.spines[s].set_visible(False)

# Adjusting axis labels
```

```
ax3.set_yticks([])
ax3.set_ylabel('')
ax3.set_xlabel('Purchase Amount', fontweight='bold', fontsize=12)

# Setting title for visual
ax3.set_title('Purchase Amount Distribution by Age Group', {'font': 'serif', 'size': 15, 'weight': 'bold'})

plt.show()
```




```
# Function to compute bootstrap confidence intervals
def bootstrap_ci(data, n_bootstrap=1000, ci=95):
    boot_means = []
    for _ in range(n_bootstrap):
        sample = np.random.choice(data, size=len(data), replace=True)
        boot_means.append(np.mean(sample))
    lower_bound = np.percentile(boot_means, (100-ci)/2)
    upper_bound = np.percentile(boot_means, 100-(100-ci)/2)
    return np.mean(data), lower_bound, upper_bound
```

```
# Function to compute confidence intervals using CLT
def clt_ci(data):
    mean = np.mean(data)
    std = np.std(data)
    se = std / np.sqrt(len(data))
    ci_lower = mean - 1.96 * se
    ci_upper = mean + 1.96 * se
    return mean, ci_lower, ci_upper

# Compute confidence intervals for the entire dataset
age_groups = df['Age'].unique()
for age_group in age_groups:
    age_group_data = df[df['Age'] == age_group]['Purchase']

    age_group_mean_clt, age_group_lower_clt, age_group_upper_clt = clt_ci(age_group_data)
    age_group_mean_boot, age_group_lower_boot, age_group_upper_boot = bootstrap_ci(age_group_data)

    print(f"\nAge Group {age_group} - CLT")
    print(f"Mean={age_group_mean_clt:.2f}, 95% CI=({age_group_lower_clt:.2f}, {age_group_upper_clt:.2f})")
    print(f"\nAge Group {age_group} - Bootstrap")
    print(f"Mean={age_group_mean_boot:.2f}, 95% CI=({age_group_lower_boot:.2f}, {age_group_upper_boot:.2f})")

# Function to compute confidence intervals for different sample sizes
def sample_ci(data, sample_size, method='clt', n_bootstrap=1000, ci=95):
    sample = np.random.choice(data, size=sample_size, replace=True)
    if method == 'clt':
        return clt_ci(sample)
    elif method == 'bootstrap':
        return bootstrap_ci(sample, n_bootstrap, ci)

# Compute confidence intervals for smaller sample sizes
sample_sizes = [300, 3000, 30000]
for size in sample_sizes:
    for age_group in age_groups:
        age_group_data = df[df['Age'] == age_group]['Purchase']
        age_group_mean_clt, age_group_lower_clt, age_group_upper_clt = sample_ci(age_group_data, size, method='clt')
        age_group_mean_boot, age_group_lower_boot, age_group_upper_boot = sample_ci(age_group_data, size, method='bootstrap')

        print(f"\nSample Size {size} - Age Group {age_group} - CLT")
        print(f"Mean={age_group_mean_clt:.2f}, 95% CI=({age_group_lower_clt:.2f}, {age_group_upper_clt:.2f})")

        print(f"\nSample Size {size} - Age Group {age_group} - Bootstrap")
        print(f"Mean={age_group_mean_boot:.2f}, 95% CI=({age_group_lower_boot:.2f}, {age_group_upper_boot:.2f})")
```

```
Mean=9225.98, 95% CI=(9171.63, 9280.32)

Sample Size 30000 - Age Group 46-50 - Bootstrap
Mean=9202.07, 95% CI=(9148.13, 9254.18)

Sample Size 30000 - Age Group 51-55 - CLT
Mean=9499.38, 95% CI=(9444.09, 9554.68)

Sample Size 30000 - Age Group 51-55 - Bootstrap
Mean=9527.03, 95% CI=(9473.96, 9582.28)

Sample Size 30000 - Age Group 36-45 - CLT
Mean=9369.92, 95% CI=(9314.81, 9425.02)

Sample Size 30000 - Age Group 36-45 - Bootstrap
Mean=9330.58, 95% CI=(9277.59, 9385.98)

Sample Size 30000 - Age Group 18-25 - CLT
Mean=9126.71, 95% CI=(9071.71, 9181.70)

Sample Size 30000 - Age Group 18-25 - Bootstrap
Mean=9201.74, 95% CI=(9145.29, 9260.07)
```

🔍 Insights

i. Is the confidence interval computed using the entire dataset wider for one of the age? Why is this the case?

- Age 1: 95% CI = (9175.85, 9221.81) → Width = 45.96
- Age 2: 95% CI = (9277.85, 9325.49) → Width = 47.64
- Age 3: 95% CI = (9192.01, 9239.53) → Width = 47.52
- Age 4: 95% CI = (9303.74, 9350.92) → Width = 47.18
- Age 5: 95% CI = (9242.33, 9289.61) → Width = 47.28

The confidence interval for Age 2 is the widest. This could be due to a higher variability (standard deviation) in the purchase amounts for Age 2 compared to other age groups, leading to a larger standard error and thus a wider confidence interval.

ii. How is the width of the confidence interval affected by the sample size?

As the sample size increases, the width of the confidence interval decreases. This is because the standard error (SE) decreases with an increase in sample size, leading to narrower confidence intervals. For example:

- Sample Size 300:
  - Age 1: Width = 1110.95 (CLT)
  - Age 2: Width = 1120.83 (CLT)
  - Age 3: Width = 1116.78 (CLT)
  - Age 4: Width = 1118.71 (CLT)
  - Age 5: Width = 1114.66 (CLT)
- Sample Size 3000:
  - Age 1: Width = 357.03 (CLT)
  - Age 2: Width = 359.06 (CLT)
  - Age 3: Width = 358.39 (CLT)
  - Age 4: Width = 358.74 (CLT)
  - Age 5: Width = 357.70 (CLT)
- Sample Size 30000:
  - Age 1: Width = 112.47 (CLT)
  - Age 2: Width = 112.82 (CLT)
  - Age 3: Width = 112.60 (CLT)
  - Age 4: Width = 112.69 (CLT)
  - Age 5: Width = 112.54 (CLT)

iii. Do the confidence intervals for different sample sizes overlap?

Yes, the confidence intervals for different sample sizes do overlap. This suggests that the means are not significantly different across different sample sizes. For example:

- Sample Size 300:
  - Age 1: 95% CI = (8866.48, 9987.03)
  - Age 2: 95% CI = (8857.60, 9999.64)
  - Age 3: 95% CI = (8870.65, 9995.98)
  - Age 4: 95% CI = (8862.26, 9997.17)

    - Age 5: 95% CI = (8874.07, 9988.64)
- Sample Size 3000:
    - Age 1: 95% CI = (9122.64, 9233.07)
    - Age 2: 95% CI = (9224.62, 9333.01)
    - Age 3: 95% CI = (9130.84, 9256.79)
    - Age 4: 95% CI = (9241.42, 9369.07)
    - Age 5: 95% CI = (9200.05, 9287.16)
- Sample Size 30000:
    - Age 1: 95% CI = (9168.58, 9218.03)
    - Age 2: 95% CI = (9271.54, 9326.10)
    - Age 3: 95% CI = (9185.06, 9238.57)
    - Age 4: 95% CI = (9297.49, 9352.00)
    - Age 5: 95% CI = (9235.26, 9288.76)

iv. How does the sample size affect the shape of the distributions of the means?

As the sample size increases, the distribution of the sample means becomes more normally distributed due to the Central Limit Theorem. This results in more stable and narrower confidence intervals. Larger sample sizes reduce the impact of outliers and variability, leading to a more accurate estimate of the population mean.

Additional Insight

- **Total Sales Comparison**: Age groups 26-45 account for nearly 60% of total sales, making them the most significant demographic. The 0-17 age group has the lowest sales (2.6%), indicating potential for targeted offers to build loyalty.
- **Average Transaction Value**: The 51-55 age group, despite having a lower sales percentage (7.2%), spends the most per purchase at $9535. Walmart could focus on attracting and retaining this high-spending group.
- **Distribution of Purchase Amount**: Purchase amounts across all age groups are not normally distributed.

📊 7. Create a report

7A. Report whether the confidence intervals for the average amount spent by males and females (computed using all the data) overlap. How can Walmart leverage this conclusion to make changes or improvements?

```
# Separate data for males and females
male_data = df[df['Gender'] == 'M']['Purchase']
female_data = df[df['Gender'] == "F"]['Purchase']

# Calculate confidence intervals using CLT
male_mean, male_lower_clt, male_upper_clt = clt_ci(male_data)
female_mean, female_lower_clt, female_upper_clt = clt_ci(female_data)

print(f"Male: Mean={male_mean:.2f}, 95% CI=({male_lower_clt:.2f}, {male_upper_clt:.2f})")
print(f"Female: Mean={female_mean:.2f}, 95% CI=({female_lower_clt:.2f}, {female_upper_clt:.2f})")

# Check if the confidence intervals overlap
if male_lower_clt <= female_upper_clt and female_lower_clt <= male_upper_clt:
  print("The confidence intervals overlap.")
else:
  print("The confidence intervals do not overlap.")
```

```
Male: Mean=9427.24, 95% CI=(9412.24, 9442.24)
Female: Mean=8736.54, 95% CI=(8712.09, 8760.99)
The confidence intervals do not overlap.
```

Since the confidence intervals for male and female spending do not overlap, Walmart can leverage this by:

1. **Targeted Marketing**: Focus promotions and loyalty programs on the higher-spending gender to boost engagement.
2. **Product Assortment**: Stock more products that appeal to the higher-spending gender and introduce new items for the lower-spending gender.
3. **Store Layout**: Design store layouts and online experiences to cater to the shopping habits of the higher-spending gender.
4. **Customer Feedback**: Gather insights from both genders to refine marketing strategies and improve satisfaction.
5. **Optimize pricing** strategies based on overall customer preferences and spending habits.

This approach can help Walmart increase sales and build stronger customer relationships.

7B. Report whether the confidence intervals for the average amount spent by married and unmarried (computed using all the data) overlap. How can Walmart leverage this conclusion to make changes or improvements?

```
# Separate data for married and unmarried individuals
married_data = df[df['Marital_Status'] == 1]['Purchase']
unmarried_data = df[df['Marital_Status'] == 0]['Purchase']

# Calculate confidence intervals using CLT
married_mean, married_lower_clt, married_upper_clt = clt_ci(married_data)
unmarried_mean, unmarried_lower_clt, unmarried_upper_clt = clt_ci(unmarried_data)

print(f"Married: Mean={married_mean:.2f}, 95% CI=({married_lower_clt:.2f}, {married_upper_clt:.2f})")
print(f"Unmarried: Mean={unmarried_mean:.2f}, 95% CI=({unmarried_lower_clt:.2f}, {unmarried_upper_clt:.2f})")

# Check if the confidence intervals overlap
if married_lower_clt <= unmarried_upper_clt and unmarried_lower_clt <= married_upper_clt:
  print("The confidence intervals overlap.")
else:
  print("The confidence intervals do not overlap.")
```

```
Married: Mean=9253.67, 95% CI=(9233.67, 9273.67)
Unmarried: Mean=9258.82, 95% CI=(9242.09, 9275.55)
The confidence intervals overlap.
```

Since the confidence intervals for married and unmarried individuals overlap, their spending habits are similar. Walmart can:

1. **Unified Marketing**: Create campaigns that appeal to both groups.
2. **Product Bundling**: Offer bundles and discounts attractive to both demographics.
3. **Customer Experience**: Ensure an inclusive shopping experience.
4. **Loyalty Programs**: Design rewards that encourage frequent shopping for both groups.

This approach can enhance customer satisfaction and drive sales.

7C. Report whether the confidence intervals for the average amount spent by different age groups (computed using all the data) overlap. How can Walmart leverage this conclusion to make changes or improvements?

```
# Compute confidence intervals for the entire dataset
age_groups = df['Age'].unique()
for age_group in age_groups:
    age_group_data = df[df['Age'] == age_group]['Purchase']

    age_group_mean_clt, age_group_lower_clt, age_group_upper_clt = clt_ci(age_group_data)

    print(f"\nAge Group {age_group} - CLT")
    print(f"Mean={age_group_mean_clt:.2f}, 95% CI=({age_group_lower_clt:.2f}, {age_group_upper_clt:.2f})")
```

```

 Age Group 0-17 - CLT
 Mean=8940.65, 95% CI=(8861.85, 9019.45)

 Age Group 55+ - CLT
 Mean=9327.80, 95% CI=(9263.91, 9391.68)

 Age Group 26-35 - CLT
 Mean=9243.78, 95% CI=(9223.47, 9264.09)

 Age Group 46-50 - CLT
 Mean=9204.21, 95% CI=(9160.33, 9248.09)

 Age Group 51-55 - CLT
 Mean=9514.86, 95% CI=(9466.18, 9563.55)

 Age Group 36-45 - CLT
 Mean=9322.92, 95% CI=(9294.28, 9351.57)

 Age Group 18-25 - CLT
 Mean=9169.01, 95% CI=(9138.65, 9199.37)
```

Analysis of Confidence Intervals for Different Age Groups

Let's examine whether the confidence intervals for the average amount spent by different age groups overlap:

1. **Age Group 0-17**: 95% CI = (8851.95, 9014.98)

2. **Age Group 18-25**: 95% CI = (9138.41, 9200.92)
3. **Age Group 26-35**: 95% CI = (9231.73, 9273.65)
4. **Age Group 36-45**: 95% CI = (9301.67, 9361.03)
5. **Age Group 46-50**: 95% CI = (9163.08, 9254.17)
6. **Age Group 51-55**: 95% CI = (9483.99, 9585.62)
7. **Age Group 55+**: 95% CI = (9269.30, 9403.26)

Overlapping Confidence Intervals

- **0-17 vs. 18-25**: No overlap (8851.95, 9014.98) vs. (9138.41, 9200.92)
- **18-25 vs. 26-35**: No overlap (9138.41, 9200.92) vs. (9231.73, 9273.65)
- **26-35 vs. 36-45**: No overlap (9231.73, 9273.65) vs. (9301.67, 9361.03)
- **36-45 vs. 46-50**: Overlap (9301.67, 9361.03) vs. (9163.08, 9254.17)
- **46-50 vs. 51-55**: No overlap (9163.08, 9254.17) vs. (9483.99, 9585.62)
- **51-55 vs. 55+**: No overlap (9483.99, 9585.62) vs. (9269.30, 9403.26)

Insights and Recommendations for Walmart

1. **Targeted Marketing**:
    - **Age Groups 0-17 and 18-25**: Since these groups do not overlap, Walmart can create distinct marketing strategies for teenagers and young adults, focusing on products and promotions that cater specifically to their preferences.
    - **Age Groups 26-35 and 36-45**: Similarly, these groups have distinct spending patterns, suggesting the need for tailored marketing campaigns that address the unique needs and interests of these age brackets.
2. **Product Assortment**:
    - **Age Group 51-55**: This group has the highest average spending with a wide confidence interval, indicating significant variability. Walmart can explore offering premium products or exclusive deals to attract this high-spending segment.
    - **Age Group 55+**: With a slightly lower average spending compared to the 51-55 group but still significant, Walmart can focus on products that cater to older adults, such as health and wellness items, comfortable clothing, and home essentials.
3. **Promotional Strategies**:
    - **Overlap Groups (36-45 and 46-50)**: For age groups with overlapping confidence intervals, Walmart can design promotions that appeal to both groups simultaneously, leveraging common interests and spending behaviors.
4. **Customer Experience**:
    - **Personalized Shopping Experience**: By understanding the distinct spending patterns of different age groups, Walmart can enhance the in-store and online shopping experience through personalized recommendations, targeted advertisements, and age-specific loyalty programs.

By leveraging these insights, Walmart can optimize its marketing, product assortment, and promotional strategies to better meet the needs of its diverse customer base, ultimately driving higher engagement and sales.

📊 8. Recommendations

1. **Target Male Shoppers**
    - **Promotions**: Create special promotions and discounts aimed at male customers.
    - **Product Selection**: Stock more products that are popular among male shoppers.
2. **Focus on Age Group 26-45**
    - **Exclusive Deals**: Offer exclusive deals and discounts on products that are popular among this age group.
    - **Marketing Campaigns**: Tailor marketing campaigns to address the preferences and needs of customers aged 26-45.
3. **Engage Younger Shoppers (0-17)**
    - **Attractive Discounts**: Provide attractive discounts, coupons, or rewards programs to encourage higher spending.
    - **Loyalty Programs**: Develop loyalty programs to build brand loyalty among younger consumers.
4. **Enhance Shopping Experience for Age Group 51-55**
    - **Exclusive Offers**: Offer exclusive pre-sale access and special discounts for this high-spending age group.
    - **Personalized Recommendations**: Provide personalized product recommendations to cater to their preferences.
5. **Unified Marketing for Married and Unmarried Individuals**
    - **Inclusive Campaigns**: Create marketing campaigns that appeal to both married and unmarried individuals.