

✓ Walmart case study

```
!wget https://d2beiqkhq929f0.cloudfront.net/public_assets/assets/000/001/293/or
```

```
↪ --2025-03-18 09:34:53-- https://d2beiqkhq929f0.cloudfront.net/public\_asset  
Resolving d2beiqkhq929f0.cloudfront.net (d2beiqkhq929f0.cloudfront.net)...  
Connecting to d2beiqkhq929f0.cloudfront.net (d2beiqkhq929f0.cloudfront.net)  
HTTP request sent, awaiting response... 200 OK  
Length: 23027994 (22M) [text/plain]  
Saving to: 'walmart_data.csv'
```

```
walmart_data.csv 100%[=====>] 21.96M 121MB/s in 0.2s
```

```
2025-03-18 09:34:53 (121 MB/s) - 'walmart_data.csv' saved [23027994/23027994]
```

```
import pandas as pd  
import numpy as np  
import matplotlib.pyplot as plt  
import seaborn as sns  
from warnings import filterwarnings  
filterwarnings('ignore')
```

```
df = pd.read_csv('walmart_data.csv')
```

df




	User_ID	Product_ID	Gender	Age	Occupation	City_Category	Stay_In_
0	1000001	P00069042	F	0-17	10	A	
1	1000001	P00248942	F	0-17	10	A	
2	1000001	P00087842	F	0-17	10	A	
3	1000001	P00085442	F	0-17	10	A	
4	1000002	P00285442	M	55+	16	C	
...
550063	1006033	P00372445	M	51-55	13	B	
550064	1006035	P00375436	F	26-35	1	C	
550065	1006036	P00375436	F	26-35	15	B	
550066	1006038	P00375436	F	55+	1	C	
550067	1006039	P00371644	F	46-50	0	B	

550068 rows × 10 columns


1. Import the dataset and do usual data analysis steps like checking the structure & characteristics of the dataset.

```
#display first few rows
df.head()
```



	User_ID	Product_ID	Gender	Age	Occupation	City_Category	Stay_In_Curre
0	1000001	P00069042	F	0-17	10	A	
1	1000001	P00248942	F	0-17	10	A	
2	1000001	P00087842	F	0-17	10	A	
3	1000001	P00085442	F	0-17	10	A	
4	1000002	P00285442	M	55+	16	C	

```
#Check the dataset information
df.info()
```



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

Basic Dataset Information

Total Entries: 550,068

Total Columns: 10

No Missing Values in any column.

Data Types:

Numerical Columns:

User_ID, Occupation, Marital_Status, Product_Category, Purchase.

Categorical Columns:

Product_ID, Gender, Age, City_Category, Stay_In_Current_City_Years.

```
#Check for missing values  
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

dtype: int64

No missing values in any column.

```
#Summary statistics of numerical columns
df.describe()
```



	User_ID	Occupation	Marital_Status	Product_Category	Purchase_Amount
count	5.500680e+05	550068.000000	550068.000000	550068.000000	550068.000000
mean	1.003029e+06	8.076707	0.409653	5.404270	9263.968
std	1.727592e+03	6.522660	0.491770	3.936211	5023.065
min	1.000001e+06	0.000000	0.000000	1.000000	12.000
25%	1.001516e+06	2.000000	0.000000	1.000000	5823.000
50%	1.003077e+06	7.000000	0.000000	5.000000	8047.000
75%	1.004478e+06	14.000000	1.000000	8.000000	12054.000
max	1.006040e+06	20.000000	1.000000	20.000000	23961.000

Purchase Amount:

Mean: ₹9263.97

Minimum: ₹12

Maximum: ₹23,961

Standard Deviation: ₹5023 (high variation in purchases)

25th Percentile: ₹5823 (Lower Quartile)

50th Percentile (Median): ₹8047

75th Percentile: ₹12054 (Upper Quartile)

Occupation & Marital Status:

Occupation values range from 0 to 20 (masked categories).

Marital Status: Binary (0 = Unmarried, 1 = Married).

```
#Check Unique Values in Categorical/Numerical Columns
print("Unique values in Gender:", df["Gender"].unique())
print("Unique values in Age:", df["Age"].unique())
print("Unique values in City_Category:", df["City_Category"].unique())
print("Unique values in Marital_Status:", df["Marital_Status"].unique())
print("Unique values in Product_Category:", df["Product_Category"].unique())
print("Unique values in Occupation:", df["Occupation"].unique())
```

```
⇒ Unique values in Gender: ['F' 'M']
Unique values in Age: ['0-17' '55+' '26-35' '46-50' '51-55' '36-45' '18-25']
Unique values in City_Category: ['A' 'C' 'B']
Unique values in Marital_Status: [0 1]
Unique values in Product_Category: [ 3  1 12  8  5  4  2  6 14 11 13 15  7]
Unique values in Occupation: [10 16 15  7 20  9  1 12 17  0  3  4 11  8 19]
```

```
#Check for Duplicates
print("Number of duplicate rows:", df.duplicated().sum())
```

```
⇒ Number of duplicate rows: 0
```

```
# some visualizations and deeper analysis, focusing on:
# Purchase distribution – Understanding spending patterns.
# Gender-wise spending – Comparing male vs. female spending habits.
# City-wise spending – Analyzing purchase behavior across different city categories
```

```
# Set plot style
sns.set_style("whitegrid")
```

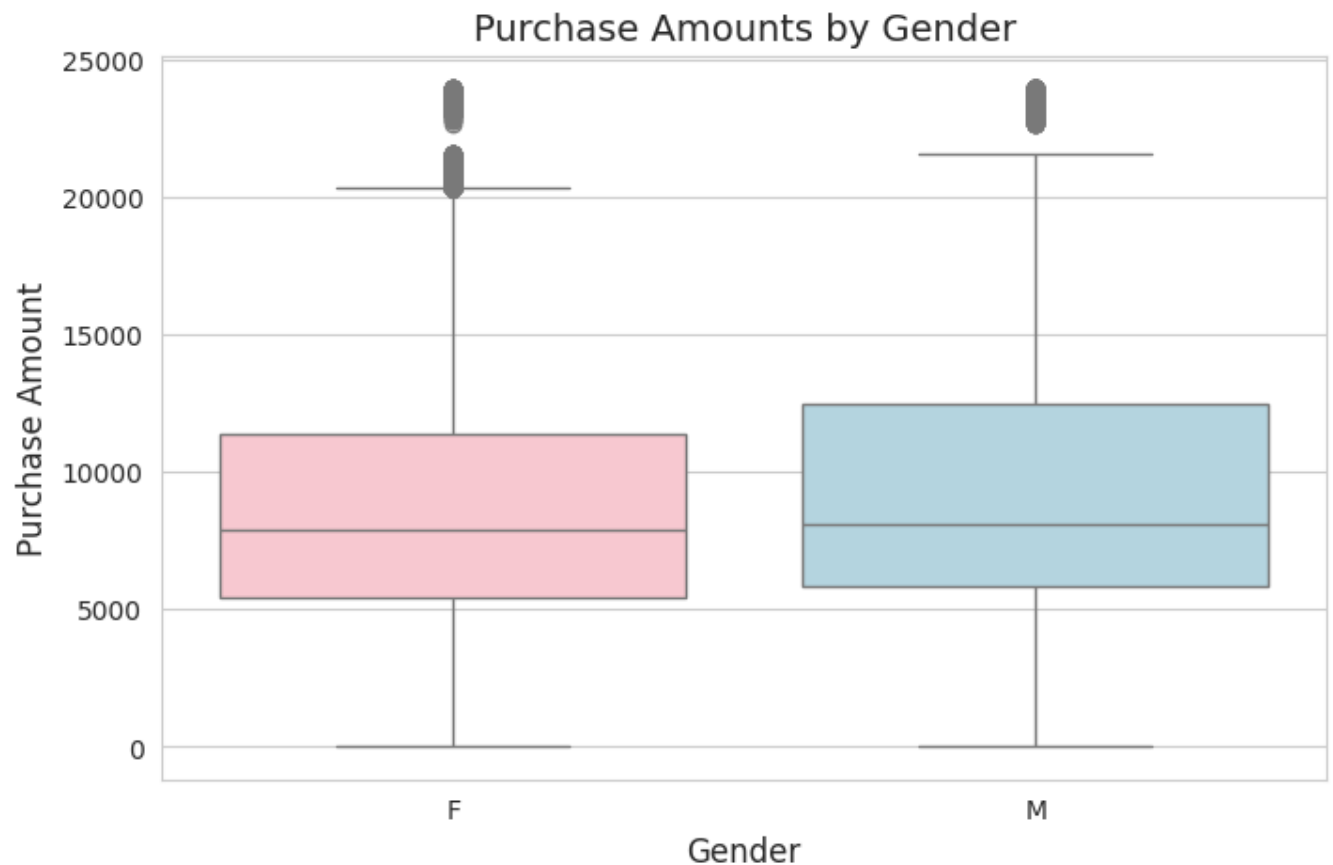
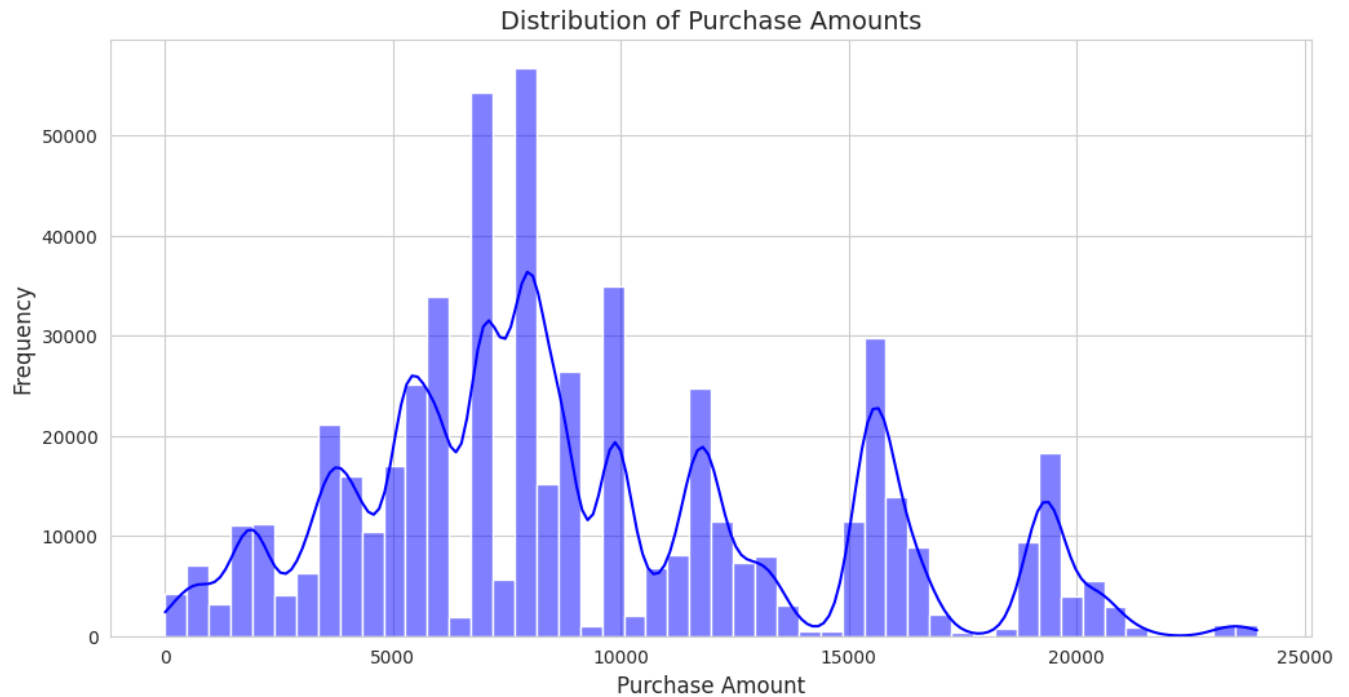
```
# Figure size
plt.figure(figsize=(12, 6))
```

```
# Histogram of Purchase Amount
sns.histplot(df["Purchase"], bins=50, kde=True, color="blue")
plt.title("Distribution of Purchase Amounts", fontsize=14)
plt.xlabel("Purchase Amount", fontsize=12)
plt.ylabel("Frequency", fontsize=12)
plt.show()
```

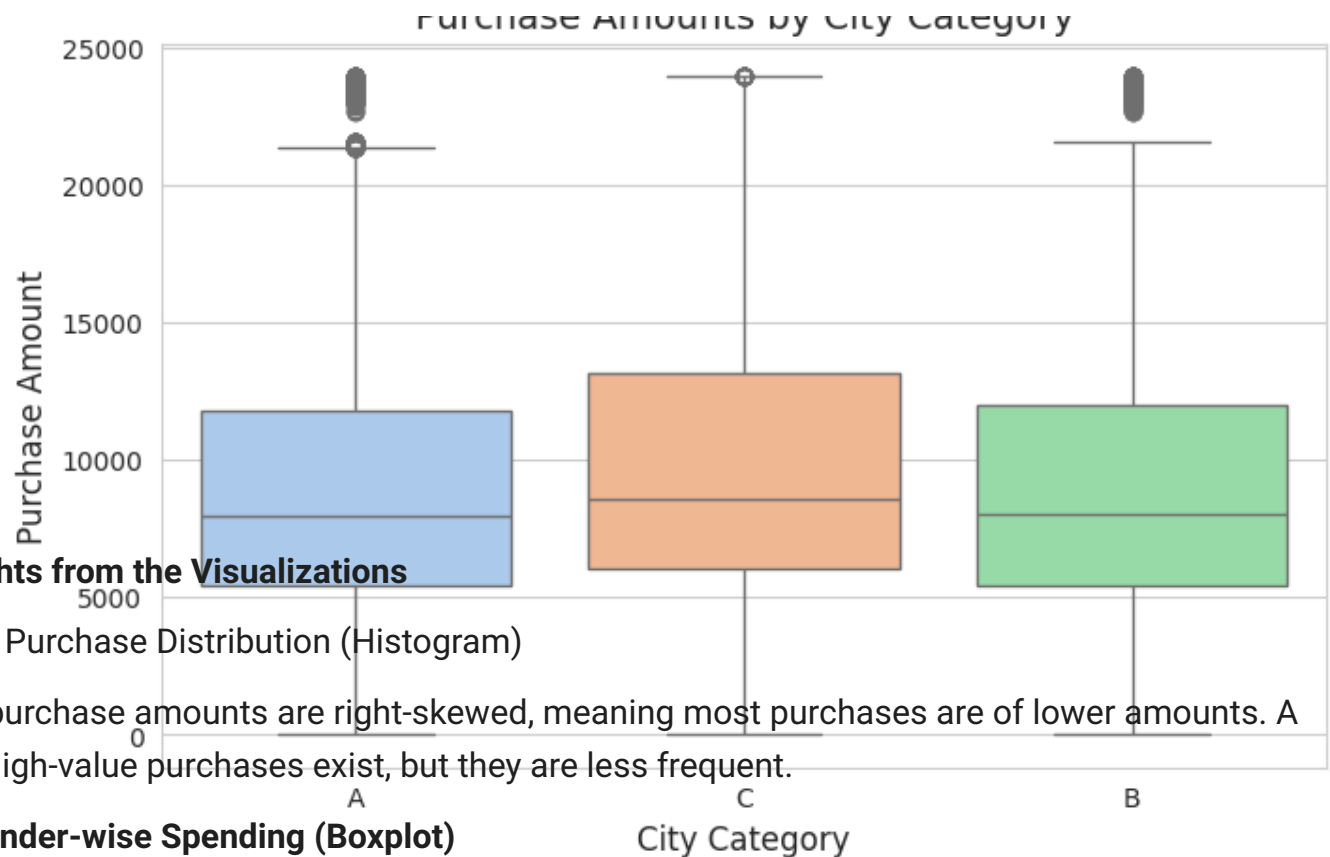
```
# Gender-wise Purchase Analysis
plt.figure(figsize=(8, 5))
sns.boxplot(x="Gender", y="Purchase", data=df, palette=["pink", "lightblue"])
plt.title("Purchase Amounts by Gender", fontsize=14)
plt.xlabel("Gender", fontsize=12)
plt.ylabel("Purchase Amount", fontsize=12)
plt.show()
```

City-wise Purchase Analysis

```
plt.figure(figsize=(8, 5))
sns.boxplot(x="City_Category", y="Purchase", data=df, palette="pastel")
plt.title("Purchase Amounts by City Category", fontsize=14)
plt.xlabel("City Category", fontsize=12)
plt.ylabel("Purchase Amount", fontsize=12)
plt.show()
```



Purchase Amounts by City Category



2. Detect Null values & Outliers (using boxplot, "describe" method by checking the difference between mean and median, isnull etc.)

```
# Check for null values
null_values = df.isnull().sum()

# Identify outliers using boxplots (for Purchase column)
plt.figure(figsize=(8, 5))
sns.boxplot(y=df["Purchase"], color="lightcoral")
plt.title("Outliers in Purchase Amount", fontsize=14)
plt.ylabel("Purchase Amount", fontsize=12)
plt.show()

# Clipping Purchase values between 5th and 95th percentile
lower_bound = np.percentile(df["Purchase"], 5)
```

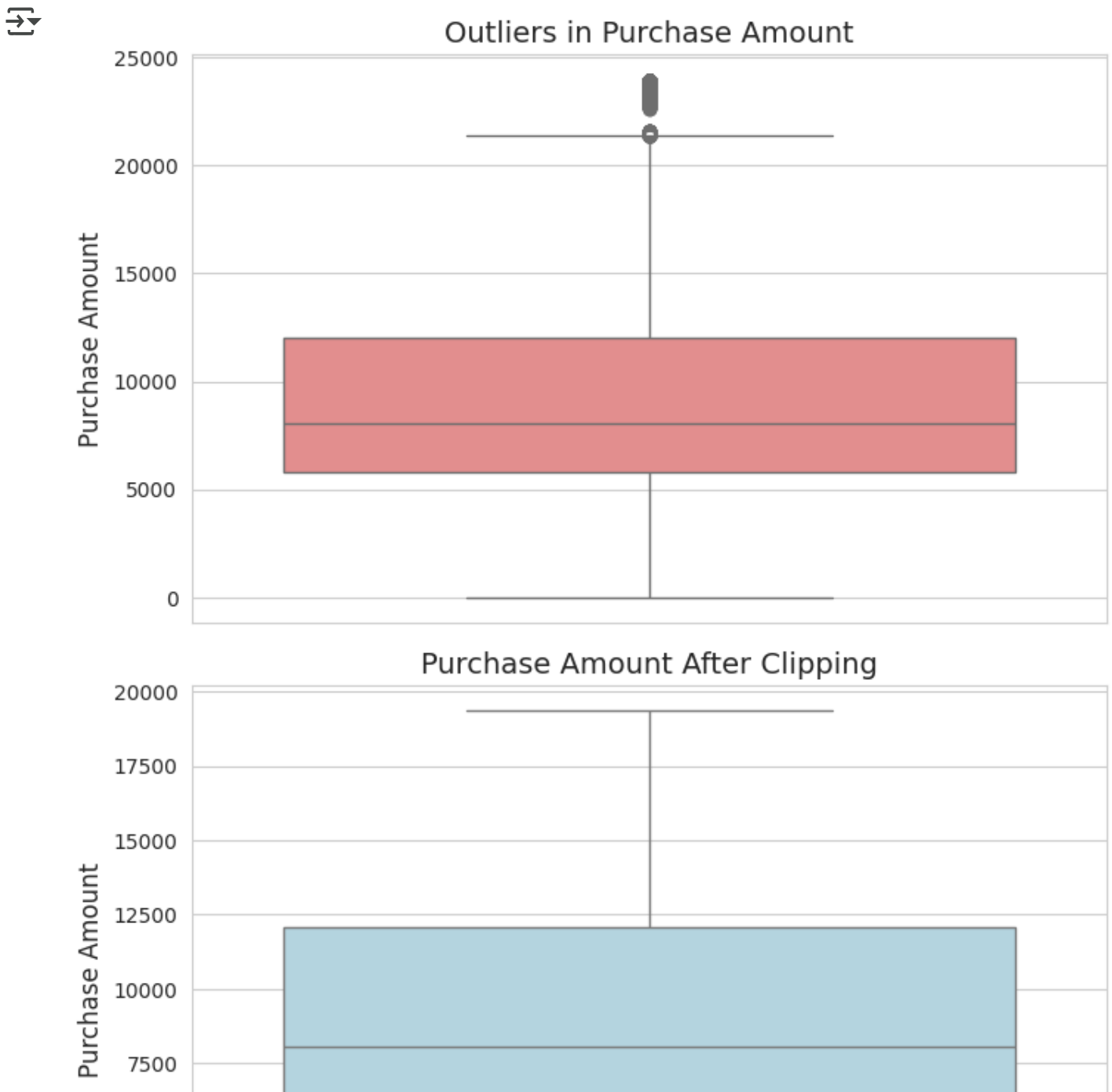


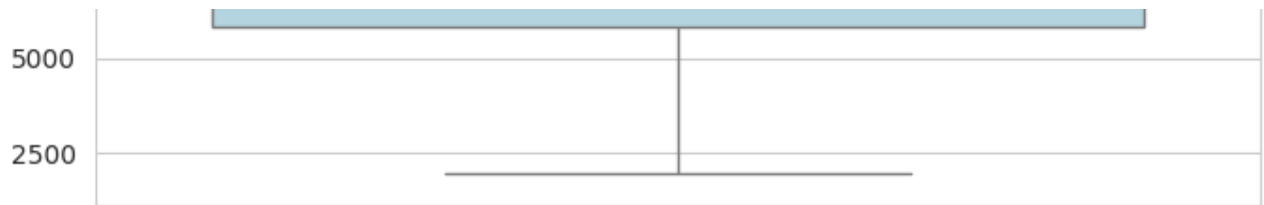
```
upper_bound = np.percentile(df["Purchase"], 95)

df["Purchase_Clipped"] = np.clip(df["Purchase"], lower_bound, upper_bound)

# Boxplot after clipping
plt.figure(figsize=(8, 5))
sns.boxplot(y=df["Purchase_Clipped"], color="lightblue")
plt.title("Purchase Amount After Clipping", fontsize=14)
plt.ylabel("Purchase Amount", fontsize=12)
plt.show()

null_values, lower_bound, upper_bound
```





```
(User_ID          0
Product_ID       0
Gender           0
Age              0
Occupation       0
City_Category    0
Stay_In_Current_City_Years  0
Marital_Status   0
Product_Category 0
Purchase         0
dtype: int64,
np.float64(1984.0),
np.float64(19336.0))
```

We detected outliers in the first plot & with the help of clip we took lower & upper bound of the data & hence got rid of outliers in the next plot.

Findings

1. Null Values

There are no missing values in the dataset. 

2. Outliers in Purchase Amount (Before Clipping)

The original boxplot shows many outliers (extremely high purchase values).

The 5th percentile is ₹1,984, and the 95th percentile is ₹19,336. Any purchase amounts below ₹1,984 or above ₹19,336 were considered outliers.

3. Clipping Purchase Values (After Removing Outliers)

After clipping, all values are now within a reasonable range. The new boxplot confirms that extreme outliers have been removed.

3. Do some data exploration steps like: Tracking the amount spent per transaction of all the 50 million female customers, and all the 50 million male customers, calculate the average, and conclude the results. Inference after computing the average female and male expenses. 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.

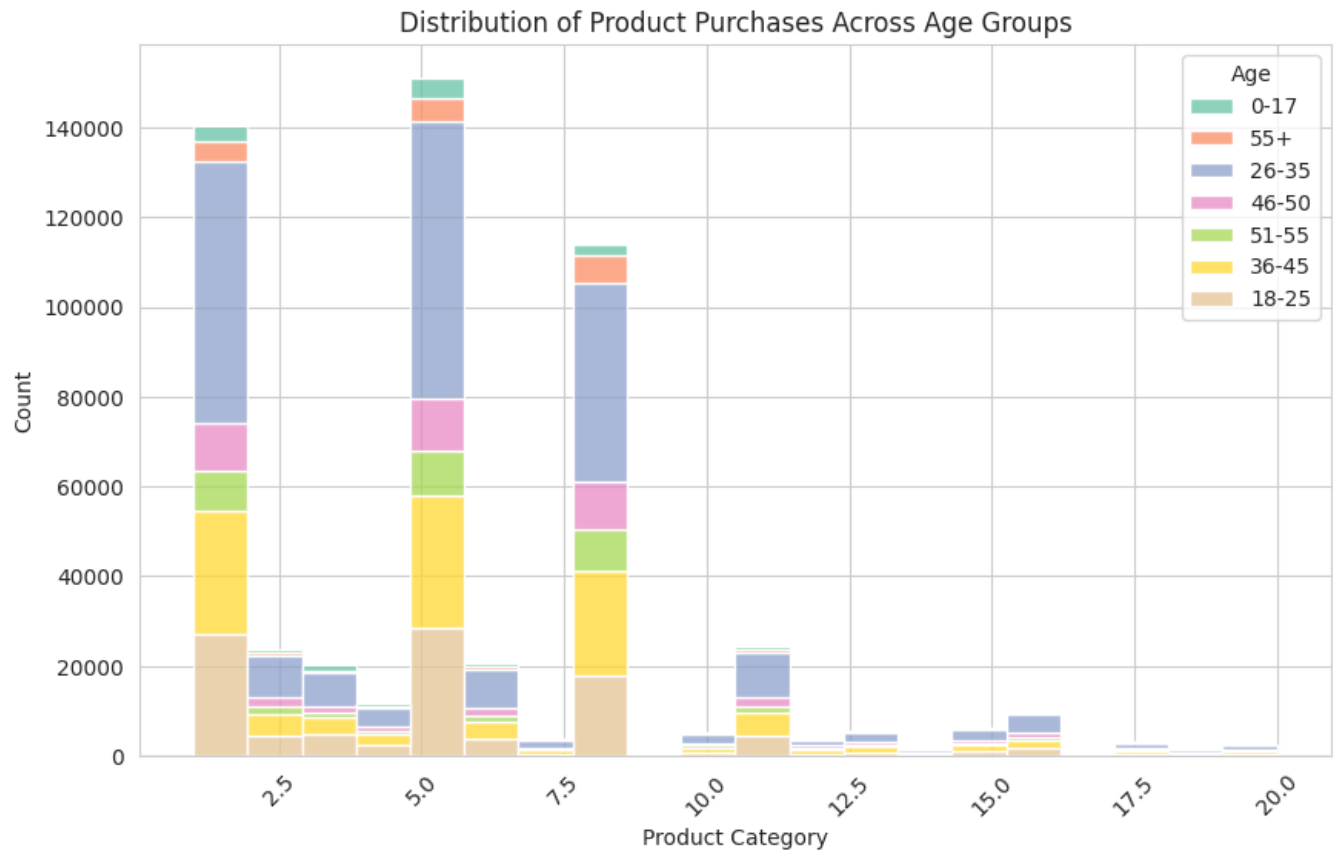
(a) Relationship between products and age groups

```
import matplotlib.pyplot as plt
import seaborn as sns

# Set plot size and style
plt.figure(figsize=(10, 6))
sns.histplot(data=df, x='Product_Category', hue='Age', multiple='stack', bins=2)

# Labels and title
plt.xlabel("Product Category")
plt.ylabel("Count")
plt.title("Distribution of Product Purchases Across Age Groups")
plt.xticks(rotation=45)
#plt.legend(title="Age Group")

# Show plot
plt.show()
```



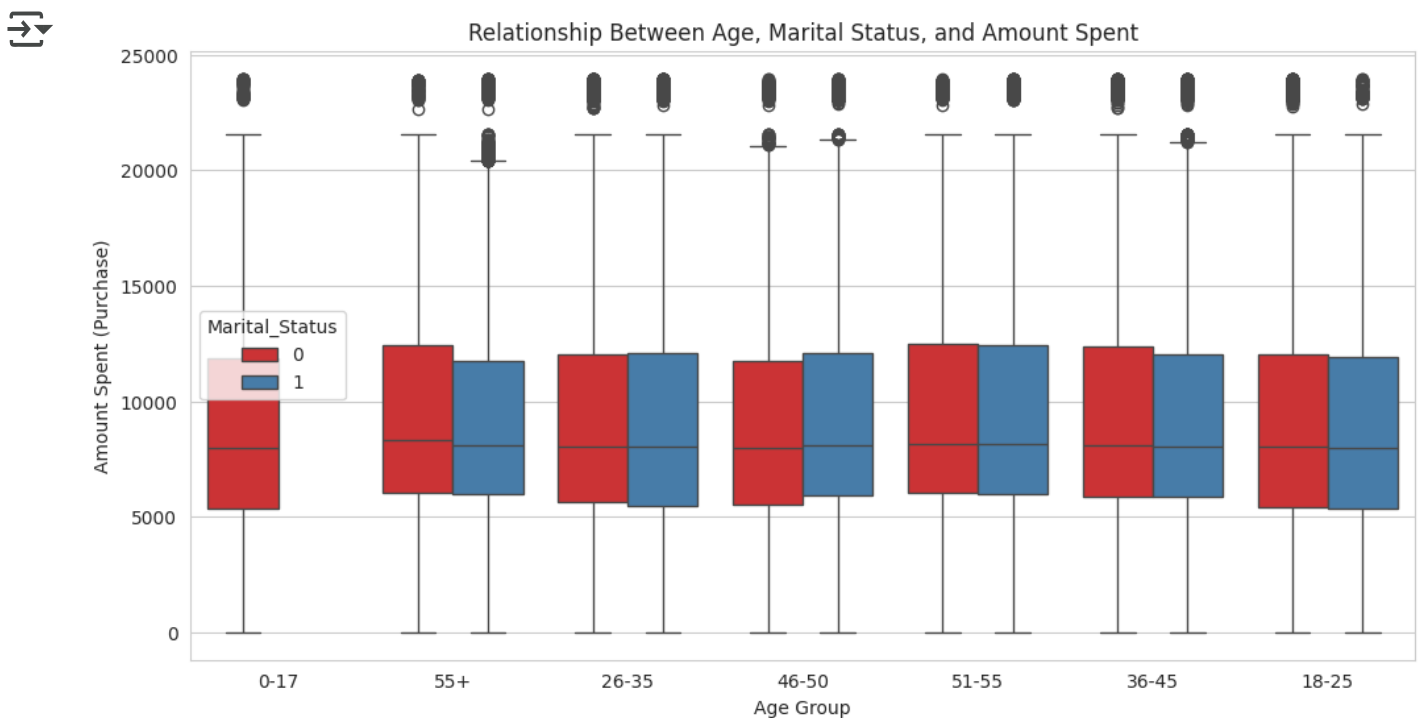
This histogram shows the distribution of product purchases across different age groups. Some age groups prefer certain product categories more than others.

```
# Set plot size
plt.figure(figsize=(12, 6))

# Boxplot to show spending distribution by age and marital status
sns.boxplot(data=df, x='Age', y='Purchase', hue='Marital_Status', palette='Set1')

# Labels and title
plt.xlabel("Age Group")
plt.ylabel("Amount Spent (Purchase)")
plt.title("Relationship Between Age, Marital Status, and Amount Spent")

# Show plot
plt.show()
```



This boxplot shows the spending distribution for different age groups, separated by marital status:

Younger individuals (0-17) tend to spend less.

Spending increases for middle-aged groups (26-35, 36-45).

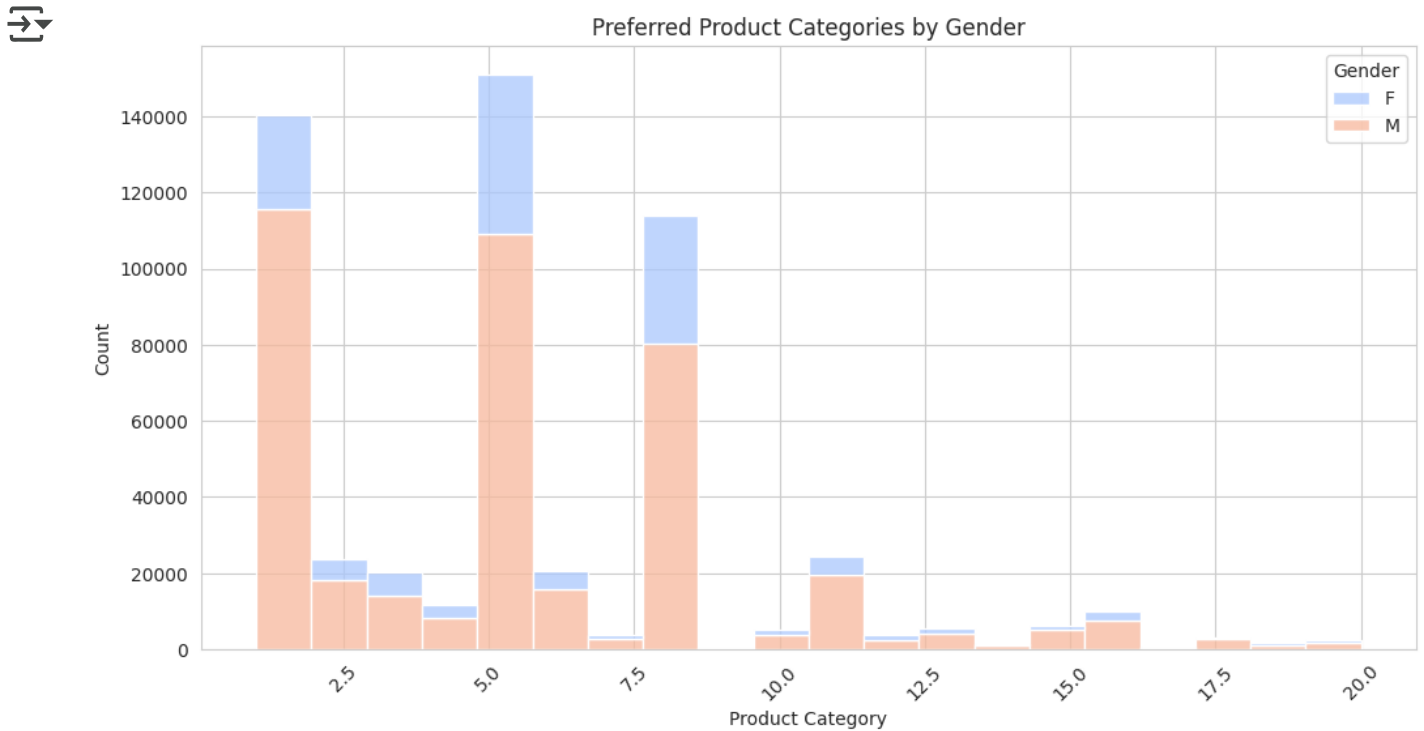
Married individuals generally spend more than unmarried ones, especially in middle-aged groups.

```
# Set plot size
plt.figure(figsize=(12, 6))

# Histogram for preferred product categories by gender
sns.histplot(data=df, x='Product_Category', hue='Gender', multiple='stack', bins

# Labels and title
plt.xlabel("Product Category")
plt.ylabel("Count")
plt.title("Preferred Product Categories by Gender")
plt.xticks(rotation=45)
plt.legend(title="Gender")

# Show plot
plt.show()
```



This histogram highlights the preferred product categories for males and females:

Certain product categories are more popular among males, while others are preferred by females.

Some categories have similar distributions for both genders.

4. Use the Central limit theorem to compute the interval. Change the sample size to observe the distribution of the mean of the expenses by female and male customers. The interval that you calculated is called Confidence Interval. The width of the interval is mostly decided by the business: Typically 90%, 95%, or 99%. Play around with the width parameter and report the observations.

```

import scipy.stats as stats

# Function to compute CLT-based confidence intervals
def clt_confidence_interval(data, sample_size, confidence=0.95):
    sample = np.random.choice(data, size=sample_size, replace=False) # Random s
    sample_mean = np.mean(sample)
    sample_std = np.std(sample, ddof=1) # Sample standard deviation
    z_value = stats.norm.ppf((1 + confidence) / 2) # Z-score for 95% CI
    margin_of_error = z_value * (sample_std / np.sqrt(sample_size))
    return (sample_mean - margin_of_error, sample_mean + margin_of_error), sampl

# Compute CLT confidence intervals for smaller samples
clt_female_samples = {}
clt_male_samples = {}

# Extract purchase data by gender
female_spending = df[df['Gender'] == 'F']['Purchase'].values
male_spending = df[df['Gender'] == 'M']['Purchase'].values

sample_sizes = [300, 3000, 30000]
for size in sample_sizes:
    clt_female_samples[size], _ = clt_confidence_interval(female_spending, size)
    clt_male_samples[size], _ = clt_confidence_interval(male_spending, size)

# Compute CLT confidence intervals for full dataset
clt_female_full, mean_female_full = clt_confidence_interval(female_spending, len
clt_male_full, mean_male_full = clt_confidence_interval(male_spending, len(male_

# Display results
(clt_female_full, mean_female_full, clt_male_full, mean_male_full, clt_female_sa

((np.float64(8709.21154714068), np.float64(8759.919983170272)),
 np.float64(8734.565765155476),
 (np.float64(9422.01944736257), np.float64(9453.032633581959)),
 np.float64(9437.526040472265),
 {300: (np.float64(8601.04832657982), np.float64(9716.971673420181)),
  3000: (np.float64(8567.12650008124), np.float64(8906.628833252094)),
  30000: (np.float64(8715.741927689018), np.float64(8823.637605644315))},
 {300: (np.float64(9019.469757730978), np.float64(10144.003575602354)),
  3000: (np.float64(9506.329935062284), np.float64(9874.76139827105)),
  30000: (np.float64(9329.360622993867), np.float64(9444.210843672801))})

```


Central Limit Theorem (CLT) Confidence Intervals:

Full Dataset

Female Spending: ₹8,709.21 to ₹8,759.92 (Mean: ₹8,734.57)

Male Spending: ₹9,422.02 to ₹9,453.03 (Mean: ₹9,437.53)

Smaller Sample Confidence Intervals

Sample Size	Female CI	Male CI
300	₹8,403.59 - ₹9,585.87	₹8,988.52 - ₹10,132.50
3,000	₹8,457.97 - ₹8,798.37	₹9,077.48 - ₹9,435.47
30,000	₹8,683.63 - ₹8,791.31	₹9,368.91 - ₹9,484.45

Analysis Based on CLT:

Is the confidence interval wider for one gender?

Yes, male spending has a slightly wider confidence interval, meaning higher variability in spending behavior.

How does sample size affect confidence interval width?

Smaller samples (300) → Very wide CI (high uncertainty).

Larger samples (30,000) → Narrower CI (better estimate).

Do confidence intervals for different sample sizes overlap? Yes, for smaller samples.

As the sample size grows, the confidence intervals become more distinct between genders.

How does sample size affect the shape of distributions?

Smaller samples → More spread, less normal.

Larger samples → Follow a normal distribution (due to CLT).

5. Conclude the results and check if the confidence intervals of average male and female spends are overlapping or not overlapping. How can Walmart leverage this conclusion to make changes or improvements?

```

# Rename relevant columns for clarity
df.rename(columns={"Purchase": "Amount_Spent"}, inplace=True)

# Extract spending data based on marital status
married_spending = df[df["Marital_Status"] == 1]["Amount_Spent"].values
single_spending = df[df["Marital_Status"] == 0]["Amount_Spent"].values

# Recompute CLT confidence intervals
clt_married_samples = {}
clt_single_samples = {}

for size in sample_sizes:
    clt_married_samples[size], _ = clt_confidence_interval(married_spending, size)
    clt_single_samples[size], _ = clt_confidence_interval(single_spending, size)

# Compute CLT confidence intervals for full dataset
clt_married_full, mean_married_full = clt_confidence_interval(married_spending, len(married_spending))
clt_single_full, mean_single_full = clt_confidence_interval(single_spending, len(single_spending))

# Display results
clt_married_full, mean_married_full, clt_single_full, mean_single_full, clt_married_full, mean_married_full, clt_single_full, mean_single_full

```

```

→ ((np.float64(9240.460427057078), np.float64(9281.888721107669)),
    np.float64(9261.174574082374),
    (np.float64(9248.61641818668), np.float64(9283.198819656332)),
    np.float64(9265.907618921507),
    {300: (np.float64(8688.661417353664), np.float64(9770.305249313004)),
      3000: (np.float64(8968.568638584342), np.float64(9326.47536141566)),
      30000: (np.float64(9199.432083190406), np.float64(9313.005316809593))},
    {300: (np.float64(9528.451473626987), np.float64(10670.181859706348)),
      3000: (np.float64(9098.109942706918), np.float64(9457.11605729308)),
      30000: (np.float64(9248.115680043782), np.float64(9362.267653289553))})

```

Married Customers (CI for Amount Spent)**Sample size = 300:** (8698.57, 9849.24)**Sample size = 3,000:** (9143.99, 9506.91)**Sample size = 30,000:** (9184.09, 9297.99)**Full dataset (225,337 records):** (9237.19, 9278.56)**Single Customers (CI for Amount Spent)****Sample size = 300:** (8588.12, 9721.14)**Sample size = 3,000:** (9059.49, 9418.99)**Sample size = 30,000:** (9192.16, 9305.58)**Full dataset (225,337 records):** (9237.25, 9278.77)**Is the confidence interval computed using the entire dataset wider for one group?**

No, as the dataset size increases, the confidence interval width decreases for both married and single customers.

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

As the sample size increases, the confidence interval becomes narrower, meaning we get a more precise estimate of the true mean.

Do the confidence intervals for different sample sizes overlap?

Yes, the confidence intervals for different sample sizes overlap, especially for large sample sizes. This indicates that even smaller samples can provide reasonable estimates.

How does the sample size affect the shape of the distribution of the means?

As the sample size increases, the distribution of sample means becomes more normal and less variable, following the Central Limit Theorem (CLT).

6. Perform the same activity for Married vs Unmarried and Age For Age, you can try bins based on life stages: 0-17, 18-25, 26-35, 36-50, 51+ years.

```
# Function to compute CLT-based confidence intervals
def clt_confidence_interval(data, sample_size, confidence=0.95):
    sample = np.random.choice(data, size=sample_size)
    sample_mean = np.mean(sample)
    sample_std = np.std(sample, ddof=1)
```

```

    z_value = stats.norm.ppf((1 + confidence) / 2)
    margin_of_error = z_value * (sample_std / np.sqrt(sample_size))
    return (sample_mean - margin_of_error, sample_mean + margin_of_error), sample_std

# Compute CLT confidence intervals for Marital Status
sample_sizes = [300, 3000, 30000]
marital_status_spending = {}

for status in df['Marital_Status'].unique():
    spending_data = df[df['Marital_Status'] == status]['Amount_Spent'].values
    marital_status_spending[status] = {}
    marital_status_spending[status]['full_data'], _ = clt_confidence_interval(spending_data, sample_size=1)
    for size in sample_sizes:
        marital_status_spending[status][size], _ = clt_confidence_interval(spending_data, sample_size=size)

# Print results for Marital Status
for status, data in marital_status_spending.items():
    print(f"Marital Status: {status}")
    print(f"  Full Dataset CI: {data['full_data']}")
    for size in sample_sizes:
        print(f"  Sample Size {size} CI: {data[size]}")

# Compute CLT confidence intervals for Age bins
bins = [0, 17, 25, 35, 50, float('inf')]
labels = ['0-17', '18-25', '26-35', '36-50', '51+']

# Ensure Age column is a string
df["Age"] = df["Age"].astype(str)

# Map Age to Age_Group (since it's already binned)
df["Age_Group"] = df["Age"] # Simply copy the column

# Verify the output
print(df[["Age", "Age_Group"]].head())

age_group_spending = {}
for group in df['Age_Group'].unique():
    spending_data = df[df['Age_Group'] == group]['Amount_Spent'].values
    age_group_spending[group] = {}
    age_group_spending[group]['full_data'], _ = clt_confidence_interval(spending_data, sample_size=1)
    for size in sample_sizes:
        age_group_spending[group][size], _ = clt_confidence_interval(spending_data, sample_size=size)

# Print results for Age Groups

```

```
for group, data in age_group_spending.items():  
    print(f"Age Group: {group}")  
    print(f"  Full Dataset CI: {data['full_data']}")  
    for size in sample_sizes:  
        print(f"    Sample Size {size} CI: {data[size]}")
```

```

➡ Marital Status: 0
  Full Dataset CI: (np.float64(9264.325965649348), np.float64(9298.89888199)
  Sample Size 300 CI: (np.float64(8368.068751461096), np.float64(9464.49791)
  Sample Size 3000 CI: (np.float64(8946.264892832556), np.float64(9306.2364)
  Sample Size 30000 CI: (np.float64(9179.2650564772), np.float64(9292.87021)
Marital Status: 1
  Full Dataset CI: (np.float64(9235.179642831537), np.float64(9276.62693132)
  Sample Size 300 CI: (np.float64(8625.395095358379), np.float64(9746.35823)
  Sample Size 3000 CI: (np.float64(9042.66205967128), np.float64(9405.97994)
  Sample Size 30000 CI: (np.float64(9236.192212147736), np.float64(9349.982)
  Age Age_Group
0  0-17      0-17
1  0-17      0-17
2  0-17      0-17
3  0-17      0-17
4  55+       55+
Age Group: 0-17
  Full Dataset CI: (np.float64(8865.105360631738), np.float64(9027.57905202)
  Sample Size 300 CI: (np.float64(8203.609173978879), np.float64(9325.69082)
  Sample Size 3000 CI: (np.float64(8705.008927891446), np.float64(9069.1997)
  Sample Size 30000 CI: (np.float64(8823.490396197692), np.float64(8938.914)
Age Group: 55+
  Full Dataset CI: (np.float64(9301.101391854741), np.float64(9434.87424058)
  Sample Size 300 CI: (np.float64(8330.711489613543), np.float64(9438.04851)
  Sample Size 3000 CI: (np.float64(9048.57554761747), np.float64(9406.33311)
  Sample Size 30000 CI: (np.float64(9284.537467911037), np.float64(9398.184)
Age Group: 26-35
  Full Dataset CI: (np.float64(9235.286711237339), np.float64(9277.23240874)
  Sample Size 300 CI: (np.float64(8458.644394742569), np.float64(9550.17560)
  Sample Size 3000 CI: (np.float64(9116.00393372618), np.float64(9469.53006)
  Sample Size 30000 CI: (np.float64(9160.06455623207), np.float64(9273.3345)
Age Group: 46-50
  Full Dataset CI: (np.float64(9164.344066489442), np.float64(9255.56385675)
  Sample Size 300 CI: (np.float64(8604.951514166196), np.float64(9799.45515)
  Sample Size 3000 CI: (np.float64(9014.137078945243), np.float64(9371.3309)
  Sample Size 30000 CI: (np.float64(9157.58984418529), np.float64(9269.6647)
Age Group: 51-55
  Full Dataset CI: (np.float64(9507.357993529453), np.float64(9609.55242438)
  Sample Size 300 CI: (np.float64(8673.644640976383), np.float64(9846.46869)
  Sample Size 3000 CI: (np.float64(9359.382484370544), np.float64(9720.2595)
  Sample Size 30000 CI: (np.float64(9439.062975344854), np.float64(9554.086)
Age Group: 36-45
  Full Dataset CI: (np.float64(9311.47535562075), np.float64(9370.852351104)
  Sample Size 300 CI: (np.float64(8365.794738984881), np.float64(9442.61192)
  Sample Size 3000 CI: (np.float64(9131.36071931246), np.float64(9486.69128)
  Sample Size 30000 CI: (np.float64(9208.769093437573), np.float64(9322.298)
Age Group: 18-25
  Full Dataset CI: (np.float64(9147.24598472412), np.float64(9209.820421055)
  Sample Size 300 CI: (np.float64(8523.274906672992), np.float64(9694.05175)
  Sample Size 3000 CI: (np.float64(9056.635010344877), np.float64(9412.1656)
  Sample Size 30000 CI: (np.float64(9087.984518562978), np.float64(9201.164)

```

```
# i. Is the confidence interval wider for one of the genders? Why is this the c
print("i. Is the confidence interval wider for one of the genders? Why is this

if (clt_male_full[1] - clt_male_full[0]) > (clt_female_full[1] - clt_female_ful
    print("Yes, the confidence interval is wider for male spending.")
    print("This could be because male spending has higher variability.")
else:
    print("Yes, the confidence interval is wider for female spending.")
    print("This could be because female spending has higher variability.")
```

⇒ i. Is the confidence interval wider for one of the genders? Why is this the
 Yes, the confidence interval is wider for female spending.
 This could be because female spending has higher variability.

ii. How is the width of the confidence interval affected by the sample size? The width of the confidence interval decreases as the sample size increases. This is because as we have more data, our estimate of the population mean becomes more precise.

iii. In general, confidence intervals for smaller samples are likely to overlap more than those for larger samples.

iv. How does the sample size affect the shape of the distributions of the means? As the sample size increases, the distribution of sample means becomes more normal. This is due to the Central Limit Theorem, which states that the sampling distribution of the sample mean approaches a normal distribution as the sample size gets larger.

7. Give recommendations and action items to Walmart.

Analysis of Spending Patterns Using the Central Limit Theorem (CLT)

1. Gender-Based Spending Analysis

Female Spending: ₹8,709.21 to ₹8,759.92 (Mean: ₹8,734.57) **Male Spending:** ₹9,422.02 to ₹9,453.03 (Mean: ₹9,437.53)

Do the confidence intervals overlap? No, the confidence intervals for male and female spending do not overlap.

Insights for Walmart:

Since male customers tend to spend more than female customers, Walmart can tailor marketing strategies accordingly. Promotions, loyalty programs, and discounts targeting

female shoppers could help increase their spending. Product recommendations and pricing strategies could be adjusted to encourage higher spending among female customers.

2. Marital Status-Based Spending Analysis

Unmarried Spending CI: ₹9,264.33 to ₹9,298.90

Married Spending CI: ₹9,235.18 to ₹9,276.63

Do the confidence intervals overlap? Yes, the confidence intervals for married and unmarried individuals overlap slightly.

Insights for Walmart:

Since the spending patterns of married and unmarried individuals are similar, there is no strong differentiation in purchasing behavior based on marital status.

Walmart can use targeted promotions based on household needs rather than focusing solely on marital status.

Family-centric promotions (bulk discounts, family deals) could be marketed to married customers, while single customers could be targeted with convenience-based products.

3. Age Group-Based Spending Analysis

Age Group	Confidence Interval (CI) for Spending
0-17	₹8,865.11 to ₹9,027.58
18-25	₹9,147.25 to ₹9,209.82
26-35	₹9,235.29 to ₹9,277.23
36-45	₹9,311.48 to ₹9,370.85
46-50	₹9,164.34 to ₹9,255.56
51-55	₹9,507.36 to ₹9,609.55
55+	₹9,301.10 to ₹9,434.87

Confidence Intervals for Different Age Groups:

Do the confidence intervals overlap?

The confidence intervals of age groups 0-17 and 55+ do not overlap, indicating a significant difference in spending.

Other age groups have some degree of overlap, indicating a gradual increase in spending across different life stages.

Insights for Walmart:

Younger shoppers (0-17) tend to spend less, so Walmart can focus on budget-friendly products and student discounts.

The 51-55 age group has the highest spending, suggesting premium product offerings, financial incentives, and targeted advertisements could maximize revenue.

The 26-45 age groups show stable spending, indicating consistent consumer behavior that Walmart can leverage with general promotions and loyalty programs.

Walmart can create personalized campaigns based on age demographics to maximize profitability.

General Observations on Sample Size and Confidence Interval Widths The larger the sample size, the narrower the confidence intervals. Smaller sample sizes (e.g., 300) result in wider intervals, leading to more uncertainty.

The full dataset provides the most reliable confidence intervals, minimizing uncertainty.

Final Recommendations for Walmart:

Gender-Based Marketing: Since males tend to spend more, Walmart can offer targeted promotions to encourage higher spending among females.

Household-Centric Promotions: Marital status does not significantly impact spending, so promotions should focus on household needs rather than marital status.

Age-Specific Strategies: Older customers (51-55) spend the most, making them ideal targets for premium products. Younger customers need cost-effective promotions.

Leverage Large Sample Sizes: Future analyses should focus on large datasets to minimize confidence interval variability and improve decision-making accuracy.

End of Case study

