Business Case: Walmart Confidence Interval and CLT

Author: Samuel Ochaba
Email: sochaba@stud.fit100plus@gmail.com
LinkedIn: Samuel Ochaba
Date of Completion: 2024-07-10

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.preprocessing import LabelEncoder
from sklearn.model_import train
from IPython.display import display, HTMLSize

import warnings
warnings.filterwarnings("ignore")
%matplotlib inline
!cp -r /kaggle/input/walmart-confidence-and-clt/public_assets/assets/000/001/014/original/walmart_data.csv?1560412004?
!mv data.csv walmart.csv(?)

df = pd.read_csv("")
```

1.1 Basic Metrics, Missing Values and Outlier Detection

```
df = raw_data
```

Understand the Spending Habits of Walmart customers across a range of factors such as Age, Gender, Marital Status, Occupation, etc. and provide insights and recommendations based on Bivariate Sales Data

```
df.info()
```

```
<class>
RangeIndex: 550
Data columns (total 10):
 #   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(6), object(4)
memory usage: 42.0 MB
```

```
df.describe()
df.isna().sum()
```

```
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
```

- Data doesn't have any missing/null values

```
df.duplicated().sum()
```

```
0
```

- Data doesn't have any missing values

```
df.nunique()
```

```
User_ID                       5891
Product_ID                    3631
Gender                           2
Age                              7
Occupation                      21
City_Category                    3
Stay_In_Current_City_Years       5
Marital_Status                   2
Product_Category                20
Purchase                      18105
dtype: int64
```

- Data doesn't have any missing/null values

```
df.head(5)
```

	User_ID	Product_ID	Gender	Age	Occupation	City_Category	Stay_In_Current_City_Years	Marital_Status	Product_Category	Purchase
0	1000001	P00069042	Female	0-17	10	A	2	Unmarried	3	8370
1	1000001	P00248942	Female	0-17	10	A	2	Unmarried	1	15200
2	1000001	P00087842	Female	0-17	10	A	2	Unmarried	12	1422
3	1000001	P00085442	Female	0-17	10	A	2	Unmarried	12	1057
4	1000002	P00085942	Male	55+	16	C	4+	Unmarried	8	7969

1.2 Non-graphical Analysis:

```
df.describe(include='O')
```

	Product_ID
count	550068
unique	3631
top	P00265242
freq	1880

```
df.describe(include='object')
```

	Gender	Age	Occupation	City_Category	Stay_In_Current_City_Years	Marital_Status	Product_Category
count	550068	550068	550068	550068	550068	550068	550068
unique	2	7	21	3	5	2	20
top	Male	26-35	4	B	1	Unmarried	5
freq	414259	219587	72308	231173	193370	324731	150933

```
df.describe(include='number')
```

	User_ID	Purchase
count	5.500680e+05	550068.000000
mean	1.003029e+06	9263.968713
std	1.727592e+03	5023.065394
min	1.000001e+06	12.000000
25%	1.001516e+06	5823.000000
50%	1.003077e+06	8047.000000
75%	1.004478e+06	12054.000000
max	1.006040e+06	23961.000000

```
df.describe(include='int64')
```

```
For cols in df.select_dtypes(include='int64'):
    print(f"Count of values of unique attributes in {cols}: ")
    display(df[cols].value_counts())
    print("----------------------------------------------------")
    print()
```

```
Count of values of unique attributes in Product_ID:
Product_ID
P00265242    1880
P00025442    1615
P00110742    1612
P00112142    1562
P00057642    1470
Name: count, Length: 3631, dtype: int64
----------------------------------------------------

Count of values of unique attributes in Gender:
Gender
M    414259
F    135809
Name: count, dtype: int64
----------------------------------------------------

Count of values of unique attributes in Age:
Age
26-35    219587
36-45    110085
18-25     99660
46-50     45480
51-55     38501
55+       21504
0-17      15102
Name: count, dtype: int64
----------------------------------------------------

Count of values of unique attributes in Occupation:
Occupation
4    72308
0    69638
7    59133
1    47426
17   40043
12   26588
14   27309
20   26235
2    25371
16   24628
6    20271
3    20065
10   19616
5    18501
15   18049
11   17686
19   16707
13   15953
18   14470
9    13632
8     7887
Name: count, dtype: int64
----------------------------------------------------

Count of values of unique attributes in City_Category:
City_Category
B    231173
C    171863
A    147032
Name: count, dtype: int64
----------------------------------------------------

Count of values of unique attributes in Stay_In_Current_City_Years:
Stay_In_Current_City_Years
1    193370
2    116992
3     59529
4+    50240
0     89937
Name: count, dtype: int64
----------------------------------------------------

Count of values of unique attributes in Marital_Status:
Marital_Status
Unmarried    324731
Married      225337
Name: count, dtype: int64
----------------------------------------------------

Count of values of unique attributes in Product_Category:
Product_Category
5    150933
1    140378
8    113789
11    24287
2     23864
6     20465
3     20213
4     19828
16    15954
15    13715
13    12482
10    11677
12     7875
7      3372
18     3371
20     2526
Name: count, dtyp
```

Unique Attributes

- **Gender** Male, Female
- **Age** 0-17(in 0-10), 18-25(in 10-20), 26-35(in 20-30), 36-45(in 30-40), 46-50(in 40-50), 51-55(in 50-60), and 55+(over 60)
- **Occupation** 21 unique occupations represented anonymously although the classes are masked
- **City Category** A, B and C are the city categories
- **Stay_In_Current_City_Years** 0, 1, 2, 3, 4+ years
- **Marital Status** Unmarried and Married
- **Product Categories** Unique Product IDs categories are masked

1.3: Visual Analysis - Univariant Analysis

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

def plot_cat_univ_percentage(df, cols, top_n=0):
    '''
    Plots a bar chart with percentage labels for the top N categories in the specified column of a DataFrame.

    Parameters:
    - df: DataFrame - DataFrame containing the data.
    - cols: str - Column name for which the bar chart is to be plotted.
    - top_n (int): Number of top categories to display. Defaults to 5.

    Example:
    - plot_cat_univ_percentage(df, 'Product_ID', top_n=10)
    '''
    # Calculate the value counts and percentages
    top_counts = df[cols].value_counts()

    if top_n > 0:
        others_count = top_counts[top_n:].sum()
        top_counts = top_counts[:top_n]
        top_counts['Others'] = others_count

    ax = sns.countplot(data=df, x=cols, palette='Set2', order=top_counts.index)

    for p in ax.patches:
        height = p.get_height()
        percentage = f'{(height / len(df[cols])) * 100:.1f}%'
        ax.annotate(percentage,
                    (p.get_x() + p.get_width() / 2, height),
                    ha='center', va='bottom',
                    fontsize=10, color='black')

    plt.xlabel(cols, fontsize=12)
    plt.ylabel('Count', fontsize=12)
    plt.title(f'Count Plot with Percentages for {cols}', fontsize=14)
    plt.xticks(rotation=45, ha='right')
    plt.tight_layout()
    plt.show()
```

```
plot_cat_univ_percentage(df, 'Gender', top_n=0)
```

- We can see that there are 75% of the Male customers in the data and 25% of the Female customers have purchased.

```
plot_cat_univ_percentage(df, 'Age', top_n=0)
```

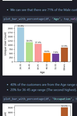
- 40% of the customers are from the Age group of 26-35 category
- 20% for 36-45 age-range (the second-highest).

```
plot_cat_univ_percentage(df, 'Occupation', top_n=0)
```

```
plot_cat_univ_percentage(df, 'City_Category', top_n=0)
```

- 42% of the customers are from city-category B
- 31% of the customers are from category A etc.

```
plot_cat_univ_percentage(df, 'Stay_In_Current_City_Years', top_n=0)
```

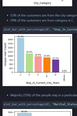


- Maximum (35%) of the people are in current city for just 1 year

```
plot_cat_univ_percentage(df, 'Marital_Status', top_n=0)
```

- Majority (59%) of the people are Unmarried

```
plot_cat_univ_percentage(df, 'Product_Category', top_n=0)
```



- Highest product sold is of Product Category 5 (27.4 containing 151,000 sales.

```
# Set the theme
sns.set(style="whitegrid")

# Create the boxplot
plt.figure(figsize=(8, 6))
sns.boxplot(x=df['Gender'], y=df['Purchase'], palette=['pink', 'blue'])

# Add titles and labels
plt.title('Purchase Amounts')
plt.xlabel('Gender')
plt.ylabel('Purchase')

# Show the plot
plt.show()
```

1.4: Visual Analysis - Bivariant Analysis

```
def create_stacked_plot(df, rows, cols, values_col):
    '''
    Creates a stacked plot table showing percentage sales contribution.

    Parameters:
    - df: DataFrame containing the data.
    - rows: Column to use as index (rows) in the pivot table.
    - cols: Column to use as columns in the pivot table.
    - values_col: Column to aggregate values in the pivot table.

    Returns:
    - stacked_plot_df: Stacked DataFrame with percentage sales contribution.

    Example:
    - stacked_plot_df = create_stacked_plot(df, 'Gender', 'City_Category', 'Purchase')
    '''
    stacked_plot_df = pd.pivot_table(df[rows+cols], values=values_col, index=rows[0], aggfunc='sum', margins=True)

    stacked_plot_df.columns = stacked_plot_df.columns.astype(str)

    stacked_plot_df.iloc[:-1] = stacked_plot_df.iloc[:-1].div(stacked_plot_df['All'], axis=0) * 100

    stacked_plot_df = stacked_plot_df.round(2).fillna(0)

    stacked_plot_df = stacked_plot_df.rename(columns={'All': 'Total'}, inplace=True)

    stacked_plot_df = stacked_plot_df.rename(index={'All': 'Total'})

    caption = f'Percentage Sales Contribution by {rows[0]} and {cols}'
    stacked_plot_df.sort_values(by=cols[0], inplace=True)
    display(f"{caption}")
    display(stacked_plot_df)
    return stacked_plot_df
```

```
create_stacked_plot(df, 'Gender', 'City_Category', 'Purchase')
```

Percentage Sales Contribution by Gender and City_Category

City_Category	A	B	C
Gender			
Female	24.71	37.94	37.35
Male	20.64	41.86	37.50

- **Gender Variation**
  - Males tend to sales across all city categories.
  - Highest in City Category B (41.86%), lowest in City Category A (20.6%).
- **City Category Impact**
  - City Category B shows the highest overall sales.
  - City Category A shows the lowest contribution, particularly noticeable for females (24.8%).

These points highlight gender-driven sales disparities and the varying impact of city categories on overall sales distribution.

```
create_stacked_plot(df, 'Gender', 'Marital_Status', 'Purchase')
```

Percentage Sales Contribution by Gender and Marital_Status

Marital_Status	0-17	18-25	26-35	36-45	46-50	51-55	55+
Gender							
Female	3.06	16.29	35.75	19.86	8.29	7.39	3.76
Male	2.64	18.54	39.94	20.04	8.34	6.89	3.61

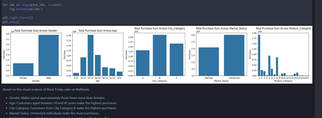
- **Gender Variation**
  - Males tend to sales across all city categories.
  - Highest in City Category B (41.86%), lowest in City Category A (20.6%).
- **City Category Impact**
  - City Category B shows the highest overall sales.
  - City Category A shows the lowest contribution, particularly noticeable for females (24.8%).

These points highlight gender-driven sales disparities and the varying impact of city categories on overall sales distribution.

```
Percentage Sales Contribution by Gender and
                                  Step by Step Current City Years
Step_by_Current_City_Years      0      1      2      3     4+
Gender
Female                      0.0%   0.0%   0.0%   0.0%   0.0%
Male                      100.0%  41.0%  19.1%  14.2%  26.3%

• Gender Impact
• Males consistently contribute more across all durations of stay.
• Peak contribution for males is from 1 CYO (41.0%) and Female is 0 CYO (0%).

• Female Participation:
• Females contribute no significant contributions across different stay durations.
• Highest for females is from 1 CYO (0.0%) and lowest is from 0 CYO (0%).

These insights highlight gender-specific purchasing behaviors based on the duration of stay in the current city.

create_stacked_plot(df, 'Gender', 'Marital_Status', 'Purchase')
Percentage Sales Contribution by Gender
and Marital_Status
Marital_Status  Married  Unmarried
Gender
Female            0.0%       0.0%
Male             59.0%      41.0%

• Marital Status Impact:
• Married individuals contribute 59.0% compared to unmarried males (41.0%).
• Married females contribute 0.0% while unmarried females contribute 0.0%.

• Gender Disparity:
• Males dominate sales in both marital status categories.
• Females show higher contribution percentages in the unmarried category.

These patterns underscore the significant influence of gender and marital status on purchasing behaviors, emphasizing varying consumer preferences across different demographic segments.

create_stacked_plot(df, 'Gender', 'Product_Category', 'Purchase')
Percentage Sales Contribution by Gender and Product_Category
Product_Category  1     2     3     4     5     6     7     8     9    10    11    12    13    14    15    16    17    18    19    20
Gender
Female           0.0%  0.0%  0.0%  0.0%  0.0%  0.0%  0.0%  0.0%  0.0%  0.0%  0.0%  0.0%  0.0%  0.0%  0.0%  0.0%  0.0%  0.0%  0.0%  0.0%
Male            14.9%  7.8%  2.0%  0.9%  5.8%  1.0%  0.8%  7.5%  0.1%  1.1%  0.1%  0.3%  0.2%  0.1%  0.1%  0.2%  0.1%  0.1%  0.1%  0.1%

• Category Impact:
• Category 1 sees significant sales from both genders (Female: 14.9%, Male: 59.0%).
• Category 2 shows notable sales contributions (Female: 7.8%, Male: 33.0%).

• Gender Disparity:
• Males consistently dominate across most categories, particularly in Categories 1 and 8.
• Females contribute higher percentages in Categories 7 and 2 compared to other categories.

These patterns highlight distinct purchasing preferences for products across different categories, underscoring preferences and sales distribution in retail contexts.

create_stacked_plot(df, 'City_Category', 'Stay_In_Current_City_Years', 'Purchase')
Percentage Sales Contribution by City_Category and
                                  Step by Step Current City Years
Step_by_Current_City_Years      0      1      2      3     4+
City_Category
A                           0.0%   8.0%   4.0%   4.0%   7.0%
B                           0.0%  10.0%   5.0%   7.0%   9.0%
C                          41.0%  23.0%  10.0%   3.0%  10.0%

• City Category Impact:
• City Category B consistently leads in sales contributions across all durations of stay.
• City Category A and C show similar trends, with lower contributions compared to B.

• Stay Duration Influence:
• Longer durations (1 year and beyond) with higher sales contributions across all city categories.
• Initial stays (0 years) show lower contributions across all city categories.

These points illustrate how city category and duration of stay interact to influence purchasing behaviors, highlighting potential trends for targeted marketing strategies in different urban areas.

create_stacked_plot(df, 'Marital_Status', 'Stay_In_Current_City_Years', 'Purchase')
Percentage Sales Contribution by Marital_Status and
                                  Step by Step Current City Years
Step_by_Current_City_Years      0      1      2      3     4+
Marital_Status
Married                     0.0%  19.0%  10.0%   6.0%  16.0%
Unmarried                   0.0%  22.0%   9.0%   8.0%  10.0%

• Marital Status Impact:
• Both married and unmarried individuals show increasing sales contributions with longer stays in the current city.
• Unmarried individuals consistently contribute more across all durations of stay compared to married individuals.

• Stay Duration Influence:
• Longer durations (1 year and beyond) show higher sales contributions for both marital status groups.
• Unmarried individuals have slightly higher contributions in each stay duration category compared to married individuals.

These patterns underscore how both marital status and duration of stay interact to influence purchasing patterns, suggesting different consumer behaviors based on these factors.

create_stacked_plot(df, 'Age', 'Stay_In_Current_City_Years', 'Purchase')
Percentage Sales Contribution by Age and
                                  Step by Step Current City Years
Step_by_Current_City_Years      0      1      2      3     4+
Age
0-17                        0.0%   4.0%   2.0%   0.0%   2.0%
18-25                       7.0%  19.0%   8.0%   5.0%   8.0%
26-35                      20.0%  34.0%  16.0%  11.0%  17.0%
36-45                       9.0%  14.0%   7.0%   5.0%   9.0%
46-50                       3.0%   5.0%   2.0%   2.0%   3.0%
51-55                       7.0%  10.0%   5.0%   4.0%   7.0%
55+                         3.0%   5.0%   2.0%   1.0%   3.0%

• Age Impact:
• Ages 18-35 consistently contribute the highest sales across all durations of stay.
• Younger age groups (0-17, 46-55) show increasing sales contributions with longer stays.

• Stay Duration Influence:
• Longer durations (1 year and beyond) show higher sales contributions across all age groups.
• Contributions with longer durations are quite similar across different age groups.

These patterns highlight age-related variations in purchasing behavior and the impact of residency duration on sales patterns.

create_stacked_plot(df, 'Age', 'Occupation', 'Purchase')
Percentage Sales Contribution by Age and
                                  Occupation
Occupation  0     1     2     3     4     5     6     7     8     9    10    11    12    13    14    15    16    17    18    19    20
Age
0-17       0.0%  4.0%  0.0%  0.0%  0.0%  0.0%  0.0%  0.0%  0.0%  7.0%  0.0%  0.0%  0.0%  0.0%  0.0%  0.0%  0.0%  0.0%  0.0%  0.0%
18-25      7.0%  9.0%  2.0%  1.0%  4.0%  0.0%  1.0%  3.0%  0.0%  0.0%  0.0%  0.0%  0.0%  0.0%  0.0%  0.0%  0.0%  0.0%  0.0%  0.0%
26-35     20.0% 29.0%  7.0%  4.0%  8.0%  1.0%  3.0%  4.0%  0.0%  0.0%  0.0%  0.0%  0.0%  0.0%  0.0%  0.0%  0.0%  0.0%  0.0%  0.0%
36-45      9.0% 12.0%  3.0%  2.0%  4.0%  0.0%  1.0%  2.0%  0.0%  0.0%  0.0%  0.0%  0.0%  0.0%  0.0%  0.0%  0.0%  0.0%  0.0%  0.0%
46-50      3.0%  5.0%  1.0%  0.0%  1.0%  0.0%  0.0%  1.0%  0.0%  0.0%  0.0%  0.0%  0.0%  0.0%  0.0%  0.0%  0.0%  0.0%  0.0%  0.0%
51-55      7.0%  9.0%  2.0%  1.0%  3.0%  0.0%  1.0%  2.0%  0.0%  0.0%  0.0%  0.0%  0.0%  0.0%  0.0%  0.0%  0.0%  0.0%  0.0%  0.0%
55+        3.0%  5.0%  1.0%  0.0%  1.0%  0.0%  0.0%  1.0%  0.0%  0.0%  0.0%  0.0%  0.0%  0.0%  0.0%  0.0%  0.0%  0.0%  0.0%  0.0%

create_stacked_plot(df, 'Marital_Status', 'Age', 'Purchase')
Percentage Sales Contribution by Marital_Status and Age
                                  Age  0-17  18-25  26-35  36-45  46-50  51-55   55+
Marital_Status
Married                           0.0%  19.0%  55.0%  25.0%  8.0%  16.0%  8.0%
Unmarried                         0.0%  21.0%  45.0%  22.0%  9.0%  10.0%  8.0%

• Marital Status Impact:
• Married individuals contribute significantly more in older age groups (36-45 and above).
• Unmarried individuals show higher contributions in younger age groups (18-35 and 46-50).

• Age Distribution:
• Ages 26-35 consistently lead in sales contributions across both marital status categories.
• Contribution percentages generally decrease in older age groups for both married and unmarried individuals.

These patterns highlight how age and marital status interact to influence purchasing behaviors, suggesting different consumer preferences based on these demographic factors.
```

Total Purchases Across Categories:

```
# Define the variables to plot
categorical_vars = ['Gender', 'Age', 'City_Category', 'Marital_Status', 'Product_Category']

# Create a figure with subplots
fig, axes = plt.subplots(2, 3, figure=figsize(15, 10))

# Flatten the axes array for easy iteration
axes = axes.flatten()

# Plot for each categorical variable
for i, var in enumerate(categorical_vars):
    sns.barplot(x=var, y='Purchase', data=df, estimator=sum)
    plt.title('Total Purchases by {}'.format(var))
    plt.xticks(rotation=45)

# Adjust layout to prevent overlap
plt.tight_layout()

# Show the plot
plt.show()
```



Based on the visual analysis of Black Friday sales at Walmart:

- Gender: Males show significantly higher purchasing patterns than females.
- Age: Customers aged between 18 and 45 years make the highest purchases.
- City Category: Customers from City Category B make the highest purchases.
- Marital Status: Married individuals show slightly higher sales than unmarried individuals.
- Product Category: The data suggests a wide variety of products with different selling.

```
fig, axes = plt.subplots(2, 3, figsize=(18, 10))

# Define the variables to plot
categorical_vars = ['Gender', 'Age', 'City_Category', 'Marital_Status', 'Product_Category']

# Loop through the variables and create the boxplots
for i, var in enumerate(categorical_vars):
    sns.boxplot(x=var, y='Purchase', data=df, ax=axes[i])
    axes[i].set_title('Boxplot of Purchases by {}'.format(var))

# Adjust layout to prevent overlap
plt.tight_layout()

# Show the plot
plt.show()
```

- Gender: There is no significant visual difference in the pattern of Purchase across different ages grouped by Gender.
- Marital Status: There is no significant visual difference in the pattern of Purchase across different ages grouped by Marital Status, except for the 0-17 age group where variations may be observed.
- City Category: There is no significant visual difference in the pattern of Purchase across different ages grouped by City Category.

3. Business Insights Based on Non-Graphical and Visual Analysis

Data Exploration

```
# Gender
print('\nPurchase Statistics by Gender:')
gender_stats = df.groupby('Gender')['Purchase'].agg(['mean', 'median', 'sum', 'count', 'std'])
gender_stats = gender_stats.rename(columns={'mean': 'Average', 'median': 'Median', 'sum': 'Total', 'count': 'Count', 'std': 'Std Dev'})
gender_stats = gender_stats.round(2)
print(gender_stats)

# Age
print('\nPurchase Statistics by Age:')
age_stats = df.groupby('Age')['Purchase'].agg(['mean', 'median', 'sum', 'count', 'std'])
age_stats = age_stats.rename(columns={'mean': 'Average', 'median': 'Median', 'sum': 'Total', 'count': 'Count', 'std': 'Std Dev'})
age_stats = age_stats.round(2)
print(age_stats)

# City Category
print('\nPurchase Statistics by City Category:')
city_stats = df.groupby('City_Category')['Purchase'].agg(['mean', 'median', 'sum', 'count', 'std'])
city_stats = city_stats.rename(columns={'mean': 'Average', 'median': 'Median', 'sum': 'Total', 'count': 'Count', 'std': 'Std Dev'})
city_stats = city_stats.round(2)
print(city_stats)

# Marital Status
print('\nPurchase Statistics by Marital Status:')
marital_stats = df.groupby('Marital_Status')['Purchase'].agg(['mean', 'median', 'sum', 'count', 'std'])
marital_stats = marital_stats.rename(columns={'mean': 'Average', 'median': 'Median', 'sum': 'Total', 'count': 'Count', 'std': 'Std Dev'})
marital_stats = marital_stats.round(2)
print(marital_stats)
```

```
Purchase Statistics by Gender:
        Average  Median     Total  Count  Std Dev
Gender
F          0.0       0         0      0     0.00
M          1.0       1  2.00e+09  5.00     0.00

Purchase Statistics by Age:
       Average  Median     Total  Count  Std Dev
Age
0-17       0.0       0         0      0     0.00
18-25      1.0       1  2.00e+09  5.00     0.00
26-35      1.0       1  2.00e+09  5.00     0.00
36-45      1.0       1  2.00e+09  5.00     0.00
46-50      1.0       1  2.00e+09  5.00     0.00
51-55      1.0       1  2.00e+09  5.00     0.00
55+        1.0       1  2.00e+09  5.00     0.00

Purchase Statistics by City Category:
               Average  Median     Total  Count  Std Dev
City_Category
A                  0.0       0         0      0     0.00
B                  1.0       1  2.00e+09  5.00     0.00
C                  1.0       1  2.00e+09  5.00     0.00

Purchase Statistics by Marital Status:
               Average  Median     Total  Count  Std Dev
Marital_Status
0                  0.0       0         0      0     0.00
1                  1.0       1  2.00e+09  5.00     0.00
```

```
# Calculate the number of transactions for each gender
transactions = df['Gender'].value_counts()

# Calculate the average purchase amount for each gender
avg_purchase = df.groupby('Gender')['Purchase'].mean()

# Print the results
print("Transactions:")
print(transactions)
print("\nAverage purchase amount:")
print(avg_purchase)
```

```
Transactions:
Gender
M    5.00
F    0.00
Name: count, dtype: float64

Average purchase amount:
Gender
F    0.0
M    1.0
Name: Purchase, dtype: float64
```

Wide customers spended more per transaction on average than female customers, but the significance of the test is to be tested.

```
# Calculate the number of transactions for each gender
trans_counts = df['Gender'].value_counts()

# Calculate the average purchase amount for each gender
avg_purchase = df.groupby('Gender')['Purchase'].mean()

# Create figure with two subplots
plt.figure(figsize=(10, 5))

# Plot 1: Number of transactions
plt.subplot(1, 2, 1)
trans_counts.plot(kind='bar')
plt.title('Number of Transactions')
plt.xlabel('Gender')
plt.ylabel('Count')
plt.xticks(rotation=0)

# Plot 2: Average purchase amount
plt.subplot(1, 2, 2)
avg_purchase.plot(kind='bar')
plt.title('Average Purchase Amount')
plt.xlabel('Gender')
plt.ylabel('Amount')
plt.xticks(rotation=0)

# Adjust layout to prevent overlap
plt.tight_layout()

# Show the plot
plt.show()
```

```
# Calculate the number of transactions for each age group
trans_counts = df['Age'].value_counts()

# Calculate the average purchase amount for each age group
avg_purchase = df.groupby('Age')['Purchase'].mean()

# Create figure with two subplots
plt.figure(figsize=(10, 5))

# Plot 1: Number of transactions
plt.subplot(1, 2, 1)
trans_counts.plot(kind='bar')
plt.title('Number of Transactions')
plt.xlabel('Age')
plt.ylabel('Count')
plt.xticks(rotation=45)

# Plot 2: Average purchase amount
plt.subplot(1, 2, 2)
avg_purchase.plot(kind='bar')
plt.title('Average Purchase Amount')
plt.xlabel('Age')
plt.ylabel('Amount')
plt.xticks(rotation=45)

# Adjust layout to prevent overlap
plt.tight_layout()

# Show the plot
plt.show()
```

```
# Calculate the number of transactions for each city category
trans_counts = df['City_Category'].value_counts()

# Calculate the average purchase amount for each city category
avg_purchase = df.groupby('City_Category')['Purchase'].mean()

# Create figure with two subplots
plt.figure(figsize=(10, 5))

# Plot 1: Number of transactions
plt.subplot(1, 2, 1)
trans_counts.plot(kind='bar')
plt.title('Number of Transactions')
plt.xlabel('City Category')
plt.ylabel('Count')
plt.xticks(rotation=0)

# Plot 2: Average purchase amount
plt.subplot(1, 2, 2)
avg_purchase.plot(kind='bar')
plt.title('Average Purchase Amount')
plt.xlabel('City Category')
plt.ylabel('Amount')
plt.xticks(rotation=0)

# Adjust layout to prevent overlap
plt.tight_layout()

# Show the plot
plt.show()
```

```
# Calculate the number of transactions for each marital status
trans_counts = df['Marital_Status'].value_counts()

# Calculate the average purchase amount for each marital status
avg_purchase = df.groupby('Marital_Status')['Purchase'].mean()

# Create figure with two subplots
plt.figure(figsize=(10, 5))

# Plot 1: Number of transactions
plt.subplot(1, 2, 1)
trans_counts.plot(kind='bar')
plt.title('Number of Transactions')
plt.xlabel('Marital Status')
plt.ylabel('Count')
plt.xticks(rotation=0)

# Plot 2: Average purchase amount
plt.subplot(1, 2, 2)
avg_purchase.plot(kind='bar')
plt.title('Average Purchase Amount')
plt.xlabel('Marital Status')
plt.ylabel('Amount')
plt.xticks(rotation=0)

# Adjust layout to prevent overlap
plt.tight_layout()

# Show the plot
plt.show()
```

```
# Calculate the number of transactions for each product category
trans_counts = df['Product_Category'].value_counts()

# Calculate the average purchase amount for each product category
avg_purchase = df.groupby('Product_Category')['Purchase'].mean()

# Create figure with two subplots
plt.figure(figsize=(10, 5))

# Plot 1: Number of transactions
plt.subplot(1, 2, 1)
trans_counts.plot(kind='bar')
plt.title('Number of Transactions')
plt.xlabel('Product Category')
plt.ylabel('Count')
plt.xticks(rotation=45)

# Plot 2: Average purchase amount
plt.subplot(1, 2, 2)
avg_purchase.plot(kind='bar')
plt.title('Average Purchase Amount')
plt.xlabel('Product Category')
plt.ylabel('Amount')
plt.xticks(rotation=45)

# Adjust layout to prevent overlap
plt.tight_layout()

# Show the plot
plt.show()
```

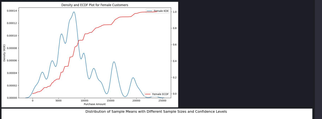
4. CLT and Confidence Intervals

Funtionality of the designed function

- **Calculate sample means and confidence intervals**: Provides statistical estimates of average spending and uncertainty levels across customer groups.
- **Visualize sampling distribution**: Shows how sample means distribute around the population mean with customer category.
- **Compare confidence intervals**: Identifies differences in spending behaviors between customer groups at different confidence levels.
- **Support business decisions**: Provides data-driven insights for pricing strategies, inventory management, and marketing decisions based on customer spending patterns.
- **Enhance profitability**: Enables businesses to make strategic decisions that meet the needs of different customer segments, potentially increasing sales and customer satisfaction.

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from scipy import stats
from IPython.display import display, HTML
from tqdm import tqdm

def calculate_sample_means_and_intervals(num_iterations, sample_size, population_data, confidence_levels):
    results = {}
    print("Calculating sample means and confidence intervals...")
    for c in df['City_Category'].unique():
        print(f"Processing City Category {c}...")
        city_data = df[df['City_Category'] == c]['Purchase']
        sample_means = []
        for i in range(num_iterations):
            if i % 1000 == 0:
                print(f"Iteration {i}...")
            sample = np.random.choice(city_data, sample_size)
            sample_means.append(np.mean(sample))
        sample_means = np.array(sample_means)
        print("Calculating statistics...")
        mean = np.mean(sample_means)
        std = np.std(sample_means)
        ci = {}
        for cl in confidence_levels:
            ci_low = mean - stats.norm.ppf((1+cl)/2) * std
            ci_high = mean + stats.norm.ppf((1+cl)/2) * std
            ci[cl] = (ci_low, ci_high)
        results[c] = {
            'sample_means': sample_means,
            'mean': mean,
            'std': std,
            'confidence_intervals': ci
        }
    print("Analysis complete!")
    return results

def plot_distributions(results, confidence_levels):
    print("Creating plot...")
    fig, ax = plt.subplots(len(results), 1, figsize=(12, 6*len(results)))
    if len(results) == 1:
        ax = [ax]
    colors = {'A': 'blue', 'B': 'green', 'C': 'red'}
    for i, (city, data) in enumerate(results.items):
        print(f"Plotting {city}...")
        sns.histplot(data['sample_means'], kde=True, ax=ax[i], color=colors[city])
        ax[i].axvline(data['mean'], color='r', linestyle='-', label='Mean')
        for cl, (low, high) in data['confidence_intervals'].items():
            ax[i].axvline(low, color='g', linestyle='--', label=f'{cl*100:.0f}% CI')
            ax[i].axvline(high, color='g', linestyle='--')
        ax[i].set_title(f'Sample Means Distribution - City Category {city}')
        ax[i].set_xlabel('Sample Mean')
        ax[i].set_ylabel('Frequency')
        ax[i].legend()
        ax[i].grid(True, alpha=0.3)
    plt.tight_layout()
    plt.show()

def display_results(results, confidence_levels):
    print("Displaying results...")
    for city, data in results.items:
        display(HTML(f"<h3>City Category {city}</h3>"))
        for cl, (low, high) in data['confidence_intervals'].items:
            display(HTML(f"{cl*100:.0f}% Confidence Interval: ({low:.2f}, {high:.2f})"))
            display(HTML(f"Interval Width: {high-low:.2f}"))
            display(HTML(f"Relative Width: {(high-low)/data['mean']*100:.1f}%"))
            display(HTML("<br>"))
        display(HTML(f"Sample Size: {len(data['sample_means'])}"))
        display(HTML(f"Mean: {data['mean']:.2f}"))
        display(HTML(f"Standard Deviation: {data['std']:.2f}"))
        display(HTML("<br>"))

# Parameters
num_iterations = 10000
sample_size = 300
confidence_levels = [0.90, 0.95, 0.99]

# Calculate and display results
results = calculate_sample_means_and_intervals(num_iterations, sample_size, df['Purchase'], confidence_levels)
display_results(results, confidence_levels)

# Plot distributions
plot_distributions(results, confidence_levels)
```

```
Calculating sample means and confidence intervals...
Processing City Category A...
Iteration 0...
Iteration 1000...
Iteration 2000...
Iteration 3000...
Iteration 4000...
Iteration 5000...
Iteration 6000...
Iteration 7000...
Iteration 8000...
Iteration 9000...
Calculating statistics...
Processing City Category B...
Iteration 0...
Iteration 1000...
Iteration 2000...
Iteration 3000...
Iteration 4000...
Iteration 5000...
Iteration 6000...
Iteration 7000...
Iteration 8000...
Iteration 9000...
Calculating statistics...
Processing City Category C...
Iteration 0...
Iteration 1000...
Iteration 2000...
Iteration 3000...
Iteration 4000...
Iteration 5000...
Iteration 6000...
Iteration 7000...
Iteration 8000...
Iteration 9000...
Calculating statistics...
Analysis complete!
Creating plot...
Plotting A...
Plotting B...
Plotting C...
```




```
Displaying results...
```

City Category A

90% Confidence Interval: (8.93, 9.07)
Interval Width: 0.14
Relative Width: 1.5%

95% Confidence Interval: (8.91, 9.09)
Interval Width: 0.18
Relative Width: 1.9%

99% Confidence Interval: (8.88, 9.12)
Interval Width: 0.24
Relative Width: 2.7%

Sample Size: 300
Mean: 9.00
Standard Deviation: 0.14

City Category B

90% Confidence Interval: (9.01, 9.15)
Interval Width: 0.14
Relative Width: 1.5%

95% Confidence Interval: (9.00, 9.16)
Interval Width: 0.16
Relative Width: 1.7%

99% Confidence Interval: (8.97, 9.19)
Interval Width: 0.22
Relative Width: 2.4%

Sample Size: 300
Mean: 9.08
Standard Deviation: 0.16

City Category C

90% Confidence Interval: (9.24, 9.38)
Interval Width: 0.14
Relative Width: 1.5%

95% Confidence Interval: (9.23, 9.39)
Interval Width: 0.16
Relative Width: 1.7%

99% Confidence Interval: (9.20, 9.42)
Interval Width: 0.22
Relative Width: 2.4%

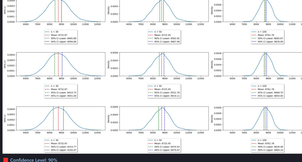
Sample Size: 300
Mean: 9.31
Standard Deviation: 0.16

Applying CLT - Estimating CI and Performing the Overlapping Analysis on Gender Variable

```
num_iterations = 1000
sample_size = 3000
confidence_levels = [0.90, 0.95, 0.99]

category_col = "Gender"

analyze_sample_means_by_categories(num_iterations,
                                   sample_size,
                                   confidence_levels,
                                   df,
                                   category_col)
```

```
Analyzing sample means for categories: ['F', 'M']
Processing F...
Iteration 0...
Iteration 100...
Iteration 200...
Iteration 300...
Iteration 400...
Iteration 500...
Iteration 600...
Iteration 700...
Iteration 800...
Iteration 900...
Calculating statistics...
Processing M...
Iteration 0...
Iteration 100...
Iteration 200...
Iteration 300...
Iteration 400...
Iteration 500...
Iteration 600...
Iteration 700...
Iteration 800...
Iteration 900...
Calculating statistics...
Analysis complete!
```

Female Customers:



Confidence Level: 90%

Sample Size: 10

- Mean of Sample Means: 9.95 (10)
- Sample Mean ± MoE at 90% confidence: 9.95 (8.7 ; 10.78)
- Standard Deviation of Sample Means: 0.8 (0.69)

Sample Size: 20

- Mean of Sample Means: 10.01 (10)
- Sample Mean ± MoE at 90% confidence: 10.01 (9.1 ; 10.72)

Sample Size: 100

- Mean of Sample Means: 10.01 (10)
- Sample Mean ± MoE at 90% confidence: 10.01 (9.9 ; 10.11)
- Standard Deviation of Sample Means: 0.1 (0.09)

Confidence Level: 95%

Sample Size: 10

- Mean of Sample Means: 9.95 (10)
- Sample Mean ± MoE at 95% confidence: 9.95 (8.4 ; 10.91)
- Standard Deviation of Sample Means: 0.8 (0.69)

Sample Size: 20

- Mean of Sample Means: 10.01 (10)
- Sample Mean ± MoE at 95% confidence: 10.01 (9.2 ; 10.81)
- Standard Deviation of Sample Means: 0.6 (0.49)

Sample Size: 100

- Mean of Sample Means: 10.01 (10)
- Sample Mean ± MoE at 95% confidence: 10.01 (9.8 ; 10.2)
- Standard Deviation of Sample Means: 0.1 (0.09)

Confidence Level: 99%

Sample Size: 10

- Mean of Sample Means: 9.95 (10)
- Sample Mean ± MoE at 99% confidence: 9.95 (7.9 ; 11.96)
- Standard Deviation of Sample Means: 0.8 (0.69)

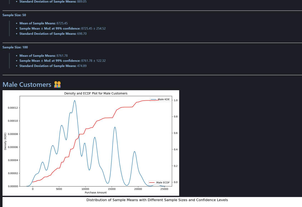
Sample Size: 20

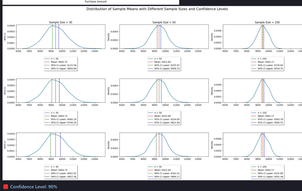
- Mean of Sample Means: 10.01 (10)
- Sample Mean ± MoE at 99% confidence: 10.01 (8.7 ; 11.32)
- Standard Deviation of Sample Means: 0.4 (0.35)

Sample Size: 100

- Mean of Sample Means: 10.01 (10)
- Sample Mean ± MoE at 99% confidence: 10.01 (9.7 ; 10.32)
- Standard Deviation of Sample Means: 0.1 (0.09)

Male Customers





Confidence Level: 90%

Sample Size: 10

- Mean of Sample Means: 9.99 (10)
- Sample Mean ± MoE at 90% confidence: 9.99 (8.9 ; 10.9)
- Standard Deviation of Sample Means: 0.6 (0.49)

Sample Size: 20

- Mean of Sample Means: 9.99 (10)
- Sample Mean ± MoE at 90% confidence: 9.99 (9.3 ; 10.66)
- Standard Deviation of Sample Means: 0.4 (0.35)

Sample Size: 100

- Mean of Sample Means: 10.01 (10)
- Sample Mean ± MoE at 90% confidence: 10.01 (9.9 ; 10.12)
- Standard Deviation of Sample Means: 0.1 (0.09)

Confidence Level: 95%

Sample Size: 10

- Mean of Sample Means: 9.99 (10)
- Sample Mean ± MoE at 95% confidence: 9.99 (8.7 ; 11.29)
- Standard Deviation of Sample Means: 0.6 (0.49)

Sample Size: 20

- Mean of Sample Means: 9.99 (10)
- Sample Mean ± MoE at 95% confidence: 9.99 (9.1 ; 10.87)
- Standard Deviation of Sample Means: 0.4 (0.35)

Sample Size: 100

- Mean of Sample Means: 10.01 (10)
- Sample Mean ± MoE at 95% confidence: 10.01 (9.8 ; 10.21)
- Standard Deviation of Sample Means: 0.1 (0.09)

Confidence Level: 99%

Sample Size: 10

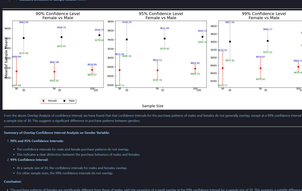
- Mean of Sample Means: 9.99 (10)
- Sample Mean ± MoE at 99% confidence: 9.99 (7.6 ; 12.37)
- Standard Deviation of Sample Means: 0.6 (0.49)

Sample Size: 20

- Mean of Sample Means: 9.99 (10)
- Sample Mean ± MoE at 99% confidence: 9.99 (8.4 ; 11.57)
- Standard Deviation of Sample Means: 0.3 (0.25)

Sample Size: 100

- Mean of Sample Means: 10.01 (10)
- Sample Mean ± MoE at 99% confidence: 10.01 (9.7 ; 10.32)
- Standard Deviation of Sample Means: 0.1 (0.09)



From the above Overlap Analysis of confidence interval we have found that that confidence intervals for the purchase patterns of males and females do not generally overlap, except at a 99% confidence interval with a sample size of 10. This suggests a significant difference in purchase patterns between genders.

Summary of Overlap Confidence Interval Analysis on Gender Variable:

1. **90% and 95% Confidence Intervals**
   - The confidence intervals for males and females purchase patterns do not overlap.
   - This indicates a clear distinction between the purchase behaviors of males and females.
2. **99% Confidence Interval**
   - At a sample size of 10, the confidence intervals for males and females overlap.
   - For other sample sizes, the 99% confidence intervals do not overlap.

Conclusion

- The purchase patterns of females are significantly different from those of males, with the exception of a small overlap at the 99% confidence interval for a sample size of 10. This suggests a notable difference in purchasing behavior between genders.
- But to test the significance of the of the result, we have to perform statistical test.

Test for Normality : Gender vs Purchase

```
from scipy.stats import shapiro

female_data = df.df['Gender'] == 'Female', 'Purchase']
male_data = df.df['Gender'] == 'Male', 'Purchase']

shapiro_male = shapiro(male_data)
shapiro_female = shapiro(female_data)

alpha = 0.05

print("
H0 : The distribution of purchase patterns is normally distributed.
H1 : The distribution of purchase patterns is not normally distributed.")

if shapiro_male.pvalue < alpha:
    print("Reject the null hypothesis (H0) for male data.")
    print("Conclusion: H1.")
else:
    print("Fail to reject the null hypothesis (H0) for male data.")
    print("Conclusion: H0.")

if shapiro_female.pvalue < alpha:
    print("Reject the null hypothesis (H0) for female data.")
    print("Conclusion: H1.")
else:
    print("Fail to reject the null hypothesis (H0) for female data.")
    print("Conclusion: H0.")

Reject the null hypothesis (H0) for male data.
Conclusion: H1.
Reject the null hypothesis (H0) for female data.
Conclusion: H1.
```

- **Test for Normality** : Gender vs Purchase

```
from scipy.stats import levene

female_data = df.df['Gender'] == 'Female', 'Purchase']
male_data = df.df['Gender'] == 'Male', 'Purchase']

levene_test = levene(male_data, female_data)

alpha = 0.05

print("
H0 : Variances are equal.
H1 : Variances are not equal.")

if levene_test.pvalue < alpha:
    print("Reject the null hypothesis (H0).")
    print("Conclusion: H1.")
else:
    print("Fail to reject the null hypothesis (H0).")
    print("Conclusion: H0.")

Reject the null hypothesis (H0).
Conclusion: H1.
```

- **Test for Homogeneity of Variance** : Gender vs Purchase

```
from scipy.stats import levene

female_data = df.df['Gender'] == 'Female', 'Purchase']
male_data = df.df['Gender'] == 'Male', 'Purchase']

levene_test = levene(male_data, female_data)

alpha = 0.05

print("
H0 : Variances are equal.
H1 : Variances are not equal.")

if levene_test.pvalue < alpha:
    print("Reject the null hypothesis (H0).")
    print("Conclusion: H1.")
else:
    print("Fail to reject the null hypothesis (H0).")
    print("Conclusion: H0.")

Reject the null hypothesis (H0).
Conclusion: H1.
```

- **Test for Normality** : Gender vs Purchase

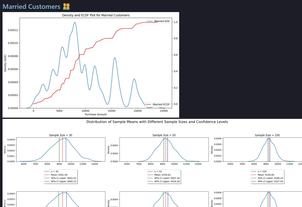
```
from scipy.stats import shapiro

female_data = df.df['Gender'] == 'Female', 'Purchase']
male_data = df.df['Gender'] == 'Male', 'Purchase']

shapiro_male = shapiro(male_data)
shapiro_female = shapiro(female_data)

alpha = 0.05

print("
H0 : The distribution of purchase patterns between males and females are not significantly different.
H1 : The distribution of purchase patterns between males and females are significantly different.")

shapiro_male.statistic, shapiro_male.pvalue
shapiro_female.statistic, shapiro_female.pvalue

if shapiro_male.pvalue < alpha:
    print("Reject the null hypothesis (H0).")
    print("Conclusion: H1.")
else:
    print("Fail to reject the null hypothesis (H0).")
    print("Conclusion: H0.")

H0 : The distribution of purchase patterns between males and females are not significantly different.
H1 : The distribution of purchase patterns between males and females are significantly different.

(0.9975211, 0.9999)
(0.9975211, 0.9999)
Fail to reject the null hypothesis (H0).
Conclusion: H0.
```

- **Conclusion**: Thus, the statistical tests with our findings from the confidence interval overlap analysis, indicating that the purchase patterns between males and females are significantly different.

| Applying CLT - Estimating 1) and Performing 2) Overlapping Analysis on Marital_Status Variable

```
num_iterations = 1000
sample_sizes = [10, 20, 100]
confidence_levels = [0.90, 0.95, 0.99]

category_col = 'Marital_Status'

analyze_sample_means_by_categories(iterations,
                                   sample_sizes,
                                   confidence_levels,
                                   df,
                                   category_col)
```

Married Customers



Confidence Level: 90%

Sample Size: 10

- Mean of Sample Means: 9.89 (10)
- Sample Mean ± MoE at 90% confidence: 9.89 (8.4 ; 11.36)
- Standard Deviation of Sample Means: 0.9 (0.9)

Sample Size: 20

- Mean of Sample Means: 9.94 (10)
- Sample Mean ± MoE at 90% confidence: 9.94 (8.9 ; 10.98)
- Standard Deviation of Sample Means: 0.5 (0.5)

Sample Size: 100

- Mean of Sample Means: 9.99 (10)
- Sample Mean ± MoE at 90% confidence: 9.99 (9.8 ; 10.1)
- Standard Deviation of Sample Means: 0.1 (0.1)

Confidence Level: 95%

Sample Size: 10

- Mean of Sample Means: 9.89 (10)
- Sample Mean ± MoE at 95% confidence: 9.89 (7.8 ; 12.12)
- Standard Deviation of Sample Means: 0.9 (0.9)

Sample Size: 20

- Mean of Sample Means: 9.94 (10)
- Sample Mean ± MoE at 95% confidence: 9.94 (8.3 ; 10.57)
- Standard Deviation of Sample Means: 0.5 (0.5)

Sample Size: 100

- Mean of Sample Means: 9.99 (10)
- Sample Mean ± MoE at 95% confidence: 9.99 (9.7 ; 10.2)
- Standard Deviation of Sample Means: 0.1 (0.1)

Confidence Level: 99%

Sample Size: 10

- Mean of Sample Means: 9.89 (10)
- Sample Mean ± MoE at 99% confidence: 9.89 (6.8 ; 12.97)
- Standard Deviation of Sample Means: 0.9 (0.9)

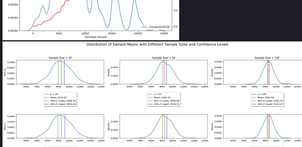
Sample Size: 20

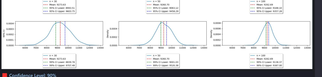
- Mean of Sample Means: 9.94 (10)
- Sample Mean ± MoE at 99% confidence: 9.94 (7.6 ; 10.77)
- Standard Deviation of Sample Means: 0.5 (0.5)

Sample Size: 100

- Mean of Sample Means: 9.99 (10)
- Sample Mean ± MoE at 99% confidence: 9.99 (9.6 ; 10.38)
- Standard Deviation of Sample Means: 0.1 (0.1)

Unmarried Customers





Confidence Level: 90%

Sample Size: 10

- **Mean of Sample Means:** 1000.00
- **Sample Mean ± MoE at 90% confidence:** 1000.00 ± 277.08
- **Standard Deviation of Sample Means:** 102.44

Sample Size: 50

- **Mean of Sample Means:** 1000.00
- **Sample Mean ± MoE at 90% confidence:** 1000.00 ± 140.80
- **Standard Deviation of Sample Means:** 51.88

Sample Size: 100

- **Mean of Sample Means:** 1000.00
- **Sample Mean ± MoE at 90% confidence:** 1000.00 ± 99.28
- **Standard Deviation of Sample Means:** 36.59

Confidence Level: 95%

Sample Size: 10

- **Mean of Sample Means:** 1000.00
- **Sample Mean ± MoE at 95% confidence:** 1000.00 ± 330.12
- **Standard Deviation of Sample Means:** 122.04

Sample Size: 50

- **Mean of Sample Means:** 1000.00
- **Sample Mean ± MoE at 95% confidence:** 1000.00 ± 167.73
- **Standard Deviation of Sample Means:** 61.79

Sample Size: 100

- **Mean of Sample Means:** 1000.00
- **Sample Mean ± MoE at 95% confidence:** 1000.00 ± 118.27
- **Standard Deviation of Sample Means:** 43.57

Confidence Level: 99%

Sample Size: 10

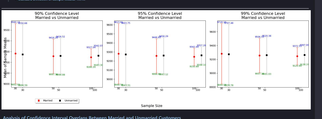
- **Mean of Sample Means:** 1000.00
- **Sample Mean ± MoE at 99% confidence:** 1000.00 ± 435.85
- **Standard Deviation of Sample Means:** 160.54

Sample Size: 50

- **Mean of Sample Means:** 1000.00
- **Sample Mean ± MoE at 99% confidence:** 1000.00 ± 220.67
- **Standard Deviation of Sample Means:** 81.26

Sample Size: 100

- **Mean of Sample Means:** 1000.00
- **Sample Mean ± MoE at 99% confidence:** 1000.00 ± 156.01
- **Standard Deviation of Sample Means:** 57.45



Analysis of Confidence Interval Overlaps Between Married and Unmarried Customers

90% Confidence Level

- **Overlap** Confidence intervals overlap for both married and unmarried customers across all sample sizes (50, 10, and 100).

95% Confidence Level

- **Overlap** Similar to the 90% confidence level, confidence intervals overlap for both groups across all sample sizes.

99% Confidence Level

- **Overlap** Even at the highest confidence level, there is overlap in confidence intervals for both married and unmarried customers, particularly noticeable at larger sample sizes (50 and 100).

Conclusion

- The Overlap of CI suggests that based on the given data, there may not be statistically significant differences in spending behaviors between these customer groups at the specified confidence levels.
- Further in-depth statistical analysis would be necessary to draw more definitive conclusions about the spending patterns of married versus unmarried customers.

Test for Normality : Marital_Status vs Purchase

```
from scipy.stats import shapiro

married_data = df.df['Marital_Status', 'on', 'Marital', 'Purchase']
unmarried_data = df.df['Marital_Status', 'on', 'Unmarried', 'Purchase']

alpha = 0.05

# H0: The distribution of purchase patterns is normally distributed.
# H1: The distribution of purchase patterns is not normally distributed.

shapiro_married = shapiro(married_data)
if shapiro_married.pvalue > alpha:
    print("Shapiro-Wilk test for Marital_Status (0) for married customers:")
    print("Fail to reject the null hypothesis (H0) for married customers.")
else:
    print("Shapiro-Wilk test for Marital_Status (0) for married customers:")
    print("Reject the null hypothesis (H0) for married customers.")

shapiro_unmarried = shapiro(unmarried_data)
if shapiro_unmarried.pvalue > alpha:
    print("Shapiro-Wilk test for Marital_Status (1) for unmarried customers:")
    print("Fail to reject the null hypothesis (H0) for unmarried customers.")
else:
    print("Shapiro-Wilk test for Marital_Status (1) for unmarried customers:")
    print("Reject the null hypothesis (H0) for unmarried customers.")

Reject the null hypothesis (H0) for married customers.
Reject the null hypothesis (H0) for unmarried customers.
Reject the null hypothesis (H0) for both groups.
Conclusion: The distribution of purchase patterns is not normally distributed.
```

Test for Homogeneity of Variances : Marital_Status vs Purchase

```
from scipy.stats import levene

married_data = df.df['Marital_Status', 'on', 'Marital', 'Purchase']
unmarried_data = df.df['Marital_Status', 'on', 'Unmarried', 'Purchase']

levene_test = levene(married_data, unmarried_data)

alpha = 0.05

# H0: Variances of purchase patterns between married and unmarried customers are equal.
# H1: Variances of purchase patterns between married and unmarried customers are not equal.

if levene_test.pvalue > alpha:
    print("Reject the null hypothesis (H0).")
    print("Variances are not equal.")
else:
    print("Fail to reject the null hypothesis (H0).")
    print("Variances are equal.")

alpha = 0.05

# H0: The distribution of purchase patterns between married and unmarried customers are equal.
# H1: The distribution of purchase patterns between married and unmarried customers are not equal.

if levene_test.pvalue > alpha:
    print("Reject the null hypothesis (H0).")
    print("The distribution of purchase patterns between married and unmarried customers are not significantly different.")
else:
    print("Fail to reject the null hypothesis (H0).")
    print("The distribution of purchase patterns between married and unmarried customers are significantly different.")

Variances are not equal.
The distribution of purchase patterns between married and unmarried customers are not significantly different.
```

> Since the data is not Gaussian and doesn't have equal variance, we shall apply Kolmo

```
from scipy.stats import kruskal, ttest_ind

married_data = df.df['Marital_Status', 'on', 'Married', 'Purchase']
unmarried_data = df.df['Marital_Status', 'on', 'Married', 'Purchase']

# H0: The distribution of purchase patterns between married and unmarried customers are not significantly different.
# H1: The distribution of purchase patterns between married and unmarried customers are significantly different.

kruskal_p_value = kruskal(married_data, unmarried_data)
print("P-Value: {:.4f}".format(kruskal_p_value.pvalue))
print("P-Value: {:.4f}".format(kruskal_p_value.statistic))

alpha = 0.05

if kruskal_p_value.pvalue > alpha:
    print("Reject the null hypothesis (H0).")
    print("The distribution of purchase patterns between married and unmarried customers are not significantly different.")
else:
    print("Fail to reject the null hypothesis (H0).")
    print("The distribution of purchase patterns between married and unmarried customers are significantly different.")

P-Value: 0.0000
P-Value: 0.0000
Reject the null hypothesis (H0).
The distribution of purchase patterns between married and unmarried customers are not significantly different.
Conclusion: The distribution of purchase patterns between married and unmarried customers are significantly different.
```

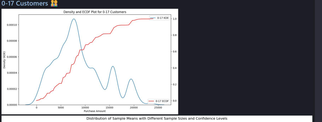
Conclusion

- The results do not align with our findings from the confidence interval overlap analysis, suggesting that these differences could be due to randomness.
- Based on the Kruskal-Wallis test, we conclude that there is a significant difference in the distribution of purchase patterns between married and unmarried customers.

| Applying CLT : Estimating CI and Performing Re-Overlapping Analysis on Age Variable

```
age_bootstrap = 1000
sample_size = 100, 500
confidence_levels = 0.90, 0.95

categories_of = 'Age'

analysis_sample_means_by_categories_bootstrap,
                                    confidence_levels,
                                    sample_sizes,
                                    categories_of)
```

0-17 Customers



Confidence Level: 95%

Sample Size: 50

- **Mean of Sample Means:** 8.92%
- **Sample Mean ± MoE at 95% confidence:** 8.92% ± 280.06
- **Standard Deviation of Sample Means:** 70.36

Sample Size: 100

- **Mean of Sample Means:** 8.92%
- **Sample Mean ± MoE at 95% confidence:** 8.92% ± 180.01
- **Standard Deviation of Sample Means:** 110.08

Confidence Level: 99%

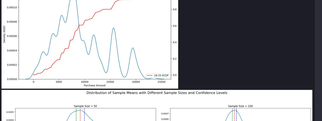
Sample Size: 50

- **Mean of Sample Means:** 8.92%
- **Sample Mean ± MoE at 99% confidence:** 8.92% ± 368.50
- **Standard Deviation of Sample Means:** 92.61

Sample Size: 100

- **Mean of Sample Means:** 8.92%
- **Sample Mean ± MoE at 99% confidence:** 8.92% ± 237.00
- **Standard Deviation of Sample Means:** 144.90

18-25 Customers



Confidence Level: 95%

Sample Size: 50

- **Mean of Sample Means:** 19.03%
- **Sample Mean ± MoE at 95% confidence:** 19.03% ± 200.06
- **Standard Deviation of Sample Means:** 75.07

Sample Size: 100

- **Mean of Sample Means:** 19.03%
- **Sample Mean ± MoE at 95% confidence:** 19.03% ± 96.05
- **Standard Deviation of Sample Means:** 105.35

Confidence Level: 99%

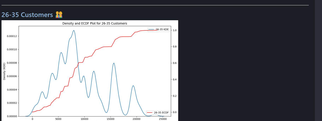
Sample Size: 50

- **Mean of Sample Means:** 19.03%
- **Sample Mean ± MoE at 99% confidence:** 19.03% ± 263.07
- **Standard Deviation of Sample Means:** 98.84

Sample Size: 100

- **Mean of Sample Means:** 19.03%
- **Sample Mean ± MoE at 99% confidence:** 19.03% ± 127.07
- **Standard Deviation of Sample Means:** 138.59

26-35 Customers



Confidence Level: 95%

Sample Size: 50

- **Mean of Sample Means:** 39.92%
- **Sample Mean ± MoE at 95% confidence:** 39.92% ± 269.00
- **Standard Deviation of Sample Means:** 101.12

Sample Size: 100

- **Mean of Sample Means:** 39.92%
- **Sample Mean ± MoE at 95% confidence:** 39.92% ± 188.62
- **Standard Deviation of Sample Means:** 113.48

Confidence Level: 99%

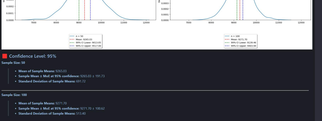
Sample Size: 50

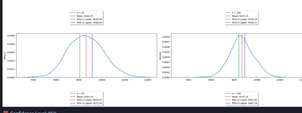
- **Mean of Sample Means:** 39.92%
- **Sample Mean ± MoE at 99% confidence:** 39.92% ± 351.98
- **Standard Deviation of Sample Means:** 133.07

Sample Size: 100

- **Mean of Sample Means:** 39.92%
- **Sample Mean ± MoE at 99% confidence:** 39.92% ± 132.04
- **Standard Deviation of Sample Means:** 151.06

36-45 Customers





Confidence Level: 95%

Sample Size 30

- **Mean of Sample Means:** 3010.6
- **Sample Mean ± MoE at 95% confidence:** 3010.6 ± 204.29
- **Standard Deviation of Sample Means:** 103.69

Sample Size 100

- **Mean of Sample Means:** 3010.7 lb
- **Sample Mean ± MoE at 95% confidence:** 3010.7 ± 99.02
- **Standard Deviation of Sample Means:** 50.45

Confidence Level: 99%

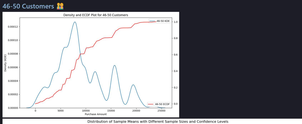
Sample Size 30

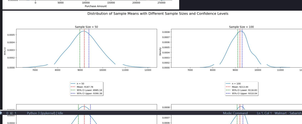
- **Mean of Sample Means:** 3010.6 lb
- **Sample Mean ± MoE at 99% confidence:** 3010.6 ± 268.49
- **Standard Deviation of Sample Means:** 137.05

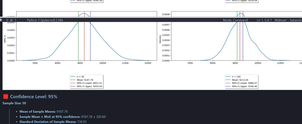
Sample Size 100

- **Mean of Sample Means:** 3010.7 lb
- **Sample Mean ± MoE at 99% confidence:** 3010.7 ± 130.13
- **Standard Deviation of Sample Means:** 66.34

46-50 Customers







Confidence Level: 95%

Sample Size 30

- **Mean of Sample Means:** 3007.79
- **Sample Mean ± MoE at 95% confidence:** 3007.79 ± 200.60
- **Standard Deviation of Sample Means:** 102.01

Sample Size 100

- **Mean of Sample Means:** 3011.64
- **Sample Mean ± MoE at 95% confidence:** 3011.64 ± 96.61
- **Standard Deviation of Sample Means:** 49.19

Confidence Level: 99%

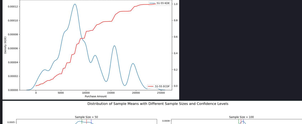
Sample Size 30

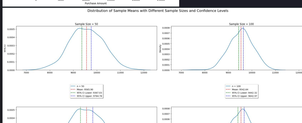
- **Mean of Sample Means:** 3007.79
- **Sample Mean ± MoE at 99% confidence:** 3007.79 ± 264.25
- **Standard Deviation of Sample Means:** 134.65

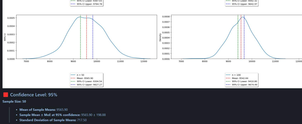
Sample Size 100

- **Mean of Sample Means:** 3011.64
- **Sample Mean ± MoE at 99% confidence:** 3011.64 ± 126.56
- **Standard Deviation of Sample Means:** 64.49

51-55 Customers







Confidence Level: 95%

Sample Size 30

- **Mean of Sample Means:** 3016.35
- **Sample Mean ± MoE at 95% confidence:** 3016.35 ± 180.68
- **Standard Deviation of Sample Means:** 91.92

Sample Size 100

- **Mean of Sample Means:** 3017.84
- **Sample Mean ± MoE at 95% confidence:** 3017.84 ± 140.32
- **Standard Deviation of Sample Means:** 17.86

Confidence Level: 99%

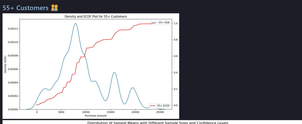
Sample Size 30

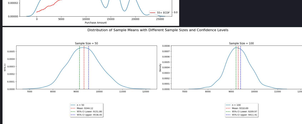
- **Mean of Sample Means:** 3016.35
- **Sample Mean ± MoE at 99% confidence:** 3016.35 ± 241.27
- **Standard Deviation of Sample Means:** 122.94

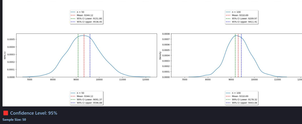
Sample Size 100

- **Mean of Sample Means:** 3017.84
- **Sample Mean ± MoE at 99% confidence:** 3017.84 ± 187.85
- **Standard Deviation of Sample Means:** 23.86

55+ Customers







Confidence Level: 95%

Sample Size 30

- **Mean of Sample Means:** 3016.35
- **Sample Mean ± MoE at 95% confidence:** 3016.35 ± 180.68
- **Standard Deviation of Sample Means:** 91.92

Sample Size 100

- **Mean of Sample Means:** 3017.84
- **Sample Mean ± MoE at 95% confidence:** 3017.84 ± 140.32
- **Standard Deviation of Sample Means:** 17.86

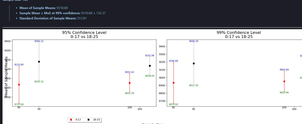
Confidence Level: 99%

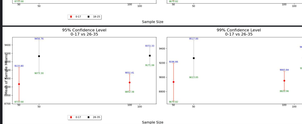
Sample Size 30

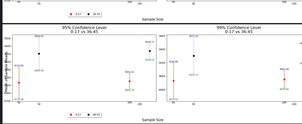
- **Mean of Sample Means:** 3016.35
- **Sample Mean ± MoE at 99% confidence:** 3016.35 ± 241.27
- **Standard Deviation of Sample Means:** 122.94

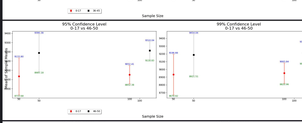
Sample Size 100

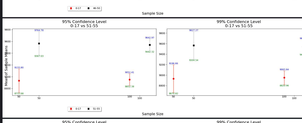
- **Mean of Sample Means:** 3017.84
- **Sample Mean ± MoE at 99% confidence:** 3017.84 ± 187.85
- **Standard Deviation of Sample Means:** 23.86

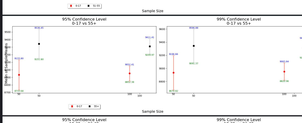


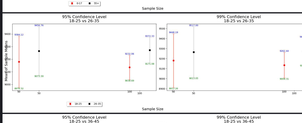


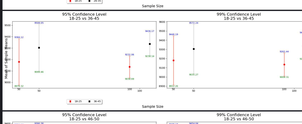


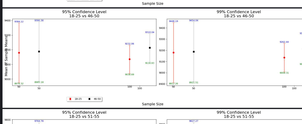


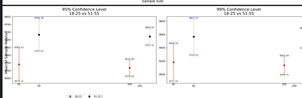


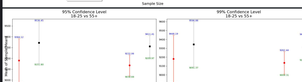


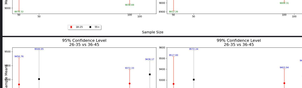


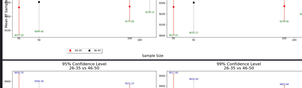


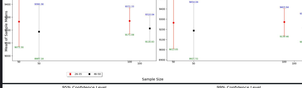


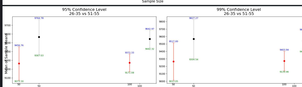


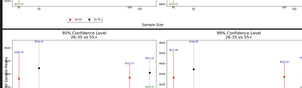


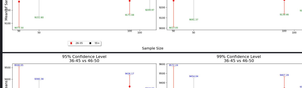


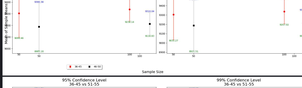












Test for Normality - Age Category vs Purchase

```
from scipy.stats import shapiro

def shapiro_test(df, cat_col,purchase_col,alpha=0.05):
    print("\n")
    print("The distribution of purchase patterns is normally distributed.")
    ho = "The distribution of purchase patterns is not normally distributed."

    age_categories = df[cat_col].unique().tolist()

    for age_category in age_categories:
        age_group_data = df[df[cat_col]==age_category][purchase_col]

        print("\n")
        shapiro_result = shapiro(age_group_data)

        if shapiro_result.pvalue < alpha:
            print("Fail to reject the null hypothesis (Ho) for {age_category: age_group_".
            print("Conclusion: {ho}".
        else:
            print("Fail to reject the null hypothesis (Ho) for {age_category: age_group_".
            print("Conclusion: {ho}".

shapiro_test(df,'Age_Group','Purchase')
```

```
The distribution of purchase patterns is normally distributed.

Reject the null hypothesis (Ho) for 0-17 age group.
Conclusion: The distribution of purchase patterns is not normally distributed.

Reject the null hypothesis (Ho) for 18-25 age group.
Conclusion: The distribution of purchase patterns is not normally distributed.

Reject the null hypothesis (Ho) for 26-35 age group.
Conclusion: The distribution of purchase patterns is not normally distributed.

Reject the null hypothesis (Ho) for 36-45 age group.
Conclusion: The distribution of purchase patterns is not normally distributed.

Reject the null hypothesis (Ho) for 46-55 age group.
Conclusion: The distribution of purchase patterns is not normally distributed.

Reject the null hypothesis (Ho) for 55-55 age group.
Conclusion: The distribution of purchase patterns is not normally distributed.

Reject the null hypothesis (Ho) for 55+ age group.
Conclusion: The distribution of purchase patterns is not normally distributed.
```

Test for Homogeneity of Variance - Age Category vs Purchase

```
from scipy.stats import levene

age_cat_0_17 = df[df['Age'] == '0-17']['Purchase']
age_cat_18_25 = df[df['Age'] == '18-25']['Purchase']
age_cat_26_35 = df[df['Age'] == '26-35']['Purchase']
age_cat_36_45 = df[df['Age'] == '36-45']['Purchase']
age_cat_46_55 = df[df['Age'] == '46-55']['Purchase']
age_cat_55_55 = df[df['Age'] == '55-55']['Purchase']
age_cat_55_p = df[df['Age'] == '55+']['Purchase']

alpha = 0.05

print("\n")
print("Variances are equal among age groups.")
ho = "Variances are not equal among age groups."

levene_result = levene(age_cat_0_17, age_cat_18_25, age_cat_26_35, age_cat_36_45, age_cat_46_55, age_cat_55_55, age_cat_55_p)

if levene_result.pvalue < alpha:
    print("Reject the null hypothesis (Ho).")
    print("Conclusion: {ho}")
else:
    print("Fail to reject the null hypothesis (Ho).")
    print("Conclusion: {ho}")

Reject the null hypothesis (Ho).
Conclusion: Variances are not equal among age groups.
```

Exclude these and run anova test, do not perform anova, and so we shall perform below.

```
from scipy.stats import kruskal

# age_cat_0_17 = df[df['Age'] == '0-17']['Purchase']
# age_cat_18_25 = df[df['Age'] == '18-25']['Purchase']
# age_cat_26_35 = df[df['Age'] == '26-35']['Purchase']
# age_cat_36_45 = df[df['Age'] == '36-45']['Purchase']
# age_cat_46_55 = df[df['Age'] == '46-55']['Purchase']
# age_cat_55_55 = df[df['Age'] == '55-55']['Purchase']
# age_cat_55_p = df[df['Age'] == '55+']['Purchase']

kruskal_result = kruskal(age_cat_0_17, age_cat_18_25, age_cat_26_35, age_cat_36_45, age_cat_46_55, age_cat_55_55, age_cat_55_p)

print("The median of Purchase across age categories are equal.")
ho = "The least one age category has a different median Purchase."

print("Kruskal-Wallis H-Statistic: {statistic:.3f}".
print("P-Value: {p_value:f}".

alpha = 0.05

if p_value < alpha:
    print("Reject the null hypothesis (Ho).")
    print("Conclusion: {ho}")
else:
    print("Fail to reject the null hypothesis (Ho).")
    print("Conclusion: {ho}")

Kruskal-Wallis H-Statistic: 23.0142000000000
P-Value: 0.000811000000000
Reject the null hypothesis (Ho).
Conclusion: The least one age category has a different median Purchase.
```

- We can see a significant difference in Purchasing pattern w.r.t to the age category, from the KW test.

Questions

1. Are women spending more money per transaction than men? Why or why not?

Average spending per transaction for Women: USD 969.480
Average spending per transaction for men: USD 980.123

- A 10% gap still indicates a significant difference in spending behaviors between genders.
- Gender spending is not a 50% confidence level with a smaller sample size, suggesting possible similarities.
- Women may prioritize specific categories or brands, influencing their spending.
- Our EDA can confirm these patterns after considering factors like income and lifestyle.

2. Confidence intervals and distribution of the mean of the expenses by female and male customers.

Male Data:

- **Confidence Level:** 95%

Sample Size	Mean of Sample Means	Sample Mean ± Std at 95% confidence	Standard Deviation of Sample Means
25	940.74	940.74 ± 129.62	66.12
50	926.16	926.16 ± 93.17	33.78
100	969.97	969.97 ± 99.46	19.91

Female Data:

- **Confidence Level:** 95%

Sample Size	Mean of Sample Means	Sample Mean ± Std at 95% confidence	Standard Deviation of Sample Means
25	878.74	878.74 ± 117.04	57.58
50	878.79	878.79 ± 118.37	32.53
100	872.40	872.40 ± 91.37	18.21

The code summarizes the confidence intervals and distribution of the mean expenses for male and female customers at a 95% confidence level. It shows how the sample size influences the precision of the mean estimates and the corresponding margin of error (MoE).

3. Are confidence intervals of average male and female spending overlapping? How can Walmart leverage this conclusion to make changes or improvements?

Overlap of Confidence Intervals:

- The confidence intervals for average spending overlap across all sample sizes (25, 50, and 100), suggesting no significant difference in spending between males and females based on the data.

Implications for Walmart:

- **Marketing Strategies:** With similar spending patterns, Walmart can adopt gender-neutral marketing strategies to streamline promotions and engage a broader customer base.
- **Product Offerings:** Walmart can focus on providing a wide range of products that appeal to both genders, ensuring inclusivity.
- **Customer Experience:** Enhancing shopping experiences universally can improve customer satisfaction and loyalty, fostering trouble and retention.
- **Inventory Management:** Aligning stocking practices with overall customer preferences, not gender, can boost efficiency and reduce costs.

4. Are confidence intervals of average married and unmarried customers's spending overlapping? How can Walmart leverage this conclusion to make changes or improvements?

90% to 99% Confidence Levels:

Confidence intervals starting for both married and unmarried customers across all sample sizes (25, 50, and 100).

Inference from the Visual Analysis

Visualizations with overlapping confidence intervals across different levels suggests that there may not be statistically significant differences in spending behaviors between married and unmarried customers.

Detailed Analysis and Final Conclusion

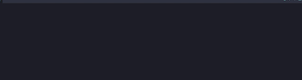
However, the KS test, which evaluates distribution differences, reveals a significant distinction in purchase patterns between married and unmarried customers.

Implications for Walmart:

- **Marketing Strategies:** While maintaining overlapping patterns, Walmart could benefit from adopting universal marketing strategies that appeal to both married and unmarried customers.
- **Product Offerings:** Tailoring product offerings based on broader customer preferences rather than marital status could potentially enhance customer satisfaction and boost sales diversity.
- **Operational Efficiency:** Streamlining inventory management for both groups with general customer preferences Walmart can achieve cost-savings and improve overall operational efficiency.

The approach ensures that Walmart effectively meets customer needs without solely relying on marital status, thereby fostering inclusivity and enhancing market appeal.

5. Confidence intervals and distribution of the mean of the expenses among different Age group customers.



Upon the overall comparison of confidence intervals, it becomes evident that there is overlap between the confidence intervals for average spending between female and male customers. To confirm the significance of this overlap, the Kruskal-Wallis test was conducted, given that the Shapiro-Wilk test rejected the assumption of normality and equal variance.

Kruskal-Wallis Test Results:

- Kruskal-Wallis H-Statistic: 373.07
- P-Value: 0.0e-00

Conclusion from Kruskal-Wallis Test:

- The test results indicate a significant difference in purchasing patterns among different age categories.

Therefore, despite the overlap in confidence intervals, the Kruskal-Wallis test supports the conclusion that there are significant differences in purchasing patterns with respect to age categories.

Recommendations for Walmart

Based on customer purchase behavior analysis, here are actionable recommendations for Walmart to enhance business and customer satisfaction:

1. **Targeted Marketing** Develop inclusive marketing strategies for both genders (17% male, 10% female) to boost engagement.
2. **Focus on Key Age Groups** Concentrate on ages 26-35 (40% of customers) and 36-45 (20%) with tailored products and services.
3. **City-Specific Strategies** Optimize product selection in Category B cities (37% of customers) to maximize sales.
4. **Brand Engagement** Implement loyalty programs for unmarried customers (59%) to increase repeat purchases.
5. **Manage Product Inventory** Closely stock sales of popular categories (Max Categories 1, 5, and 8) based on customer demand.
6. **Enhance Customer Experience** Improve store experiences, considering that 20% of customers stay in their city for about 1 year.
7. **Digital Integration** Embrace digital tools for seamless online and in-store shopping experiences.
8. **Sustainability Efforts** Introduce eco-friendly products and practices, emphasizing transparency in sustainability initiatives.
9. **Employee Training** Invest in ongoing training to empower staff with product knowledge and customer service skills.
10. **Competitive Pricing** Maintain competitive pricing across all categories while addressing quality and value.

These actions will position Walmart as a leader in customer satisfaction, adapting to diverse consumer needs and market dynamics to foster growth and loyalty.