Author: Sumanth Reddy

Email: sumanthreddy996@gmail.com

Description: This notebook performs statistical analysis on Walmart customer purchase data and provides insights based on age, marital status, and gender. The analysis includes hypothesis testing, confidence interval calculations, and recommendations for marketing strategies based on the observed patterns in the data.

This analysis is highly beneficial for small business owners, as it helps them understand customer spending behavior, optimize their marketing strategies, and make informed business decisions.

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

import warnings
warnings.filterwarnings('ignore')

!gdown 1QKYf_FsqoJWV1aMBiDZUrNr5176uvbZi
```

```
Downloading...
From: https://drive.google.com/uc?id=1QKYf_FsqoJWV1aMBiDZUrNr5176uvbZi
```

```
To: /content/walmart_data.csv
100% 23.0M/23.0M [00:00<00:00, 58.9MB/s]
```

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

```
{"type":"dataframe","variable_name":"df"}
```

Problem Statement

Objective

The goal of this analysis is to evaluate if spending habits during Black Friday at Walmart differ between male and female customers. Specifically, we aim to determine whether women spend more than men. Additionally, the analysis will examine how demographic and socioeconomic factors such as age, occupation, and marital status influence spending patterns.

Key Questions

1. **Gender-based Spending:** Do women tend to spend more than men on Black Friday?

2. **Influence of Demographics:** How do factors like age, occupation, and marital status correlate with spending patterns?

3. **Cross-interactions:** Are there notable interactions between gender and other factors (e.g., does the spending difference by gender vary across age groups or marital statuses)?

Expected Outcomes

1. Insights into gender-specific spending behaviors during Black Friday.

2. Identification of key demographic factors influencing spending.

3. Recommendations for targeted marketing strategies based on the findings.

Approach

1. **Data Collection:**
   - Utilize Black Friday sales data from Walmart, including gender, age, occupation, marital status, and spending details.
2. **Statistical Analysis:**
   - Perform descriptive statistics to summarize spending data.

   - Conduct hypothesis testing to determine if spending differences by gender are statistically significant.

- Use multivariate analysis to explore relationships between spending and demographic factors.

3. **Visualization:**
   - Create clear charts and graphs (e.g., bar plots, box plots) to communicate insights effectively.

```
print("Number of rows in the dataset : ",df.shape[0])
print("Number of columns in the dataset : ",df.shape[1])
```

```
Number of rows in the dataset :  550068
Number of columns in the dataset :  10
```

```
df.dtypes
```

```
User_ID                      int64
Product_ID                  object
Gender                      object
Age                         object
Occupation                   int64
City_Category               object
Stay_In_Current_City_Years  object
Marital_Status               int64
Product_Category             int64
Purchase                     int64
dtype: object
```

```
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
```

Observing Unique Values

To analyze the `Occupation`, `Marital_Status`, and `Product_Category` columns, we first need to inspect their unique values. This step helps us understand the range of numerical data in these columns, which will guide us in converting them into categorical variables.

Explanation:

1. **Occupation:**
   - This column likely represents different professions encoded as numbers. By observing unique values, we can decide whether to map these numbers to specific job titles or treat them as categorical labels.
2. **Marital_Status:**
   - Encoded as `0` and `1`, this column indicates the marital status of customers. We need to map these numbers into meaningful categories, such as `Single` and `Married`.
3. **Product_Category:**
   - Represents different categories of products, also encoded as numbers. These values should be treated as categorical data to ensure proper analysis.

```
c=['Gender', 'Age', 'Occupation',
'City_Category','Stay_In_Current_City_Years',
'Marital_Status','Product_Category']
for i in c:
  print("Unique values in column "+i+" : ",df[i].unique())
```

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

```
df['Occupation']=[str(i) for i in df['Occupation']]
df['Marital_Status']=["Unmarried" if i==0 else "Married" for i in
df['Marital_Status']]
df['Product_Category']=[str(i) for i in df['Product_Category']]
df.head()
```

```
{"type":"dataframe","variable_name":"df"}
```

Handling Ordinal Categories

Explanation:

The `Age` and `Stay_In_Current_City_Years` columns are already categorical, but they represent **ordinal categories**, meaning the categories have a natural order. For accurate analysis, we must define this order explicitly.

1. **Age:**

- This column contains age ranges like `0-17`, `18-25`, `26-35`, etc. These ranges have a logical sequence based on increasing age. Treating them as ordinal categories ensures that any comparison or sorting respects this natural order.

2. **Stay_In_Current_City_Years:**
   - This column represents the number of years a customer has stayed in their current city, with values such as `0`, `1`, `2`, `3`, and `4+`. These categories are also ordinal since they reflect increasing durations of stay.

```
age_order = ["0-17", "18-25", "26-35", "36-45", "46-50", "51-55",
"55+"]
df['Age'] = pd.Categorical(df['Age'], categories=age_order,
ordered=True)
stay_order=['0','1','2','3','4+']
df['Stay_In_Current_City_Years']=pd.Categorical(df['Stay_In_Current_Ci
ty_Years'], categories=stay_order, ordered=True)
df.head()
```

```
{"type":"dataframe","variable_name":"df"}
```

```
d=df['Purchase'].describe()
d
```

```
count    550068.000000
mean       9263.968713
std        5023.065394
min          12.000000
25%        5823.000000
50%        8047.000000
75%       12054.000000
max       23961.000000
Name: Purchase, dtype: float64
```

```
print("Average purchase amount per transcation :
",round(d.loc['mean'],2))
print("Standard Deviation of  purchase amount per transcation :
",round(d.loc['std'],2))
print("Minimum purchase amount of all transcations :
",round(d.loc['min'],2))
print("Maximum purchase amount of all transcations :
",round(d.loc['max'],2))
```

```
Average purchase amount per transcation :  9263.97
Standard Deviation of  purchase amount per transcation :  5023.07
Minimum purchase amount of all transcations :  12.0
Maximum purchase amount of all transcations :  23961.0
```

```
print("First quartile of purchase amount per transcation :
",round(d.loc['25%'],2))
print("medain amount of  purchase amount per transcation :
",round(d.loc['50%'],2))
```

```
print("Third quartile of purchase amount per transcation :
",round(d.loc['75%'],2))
```

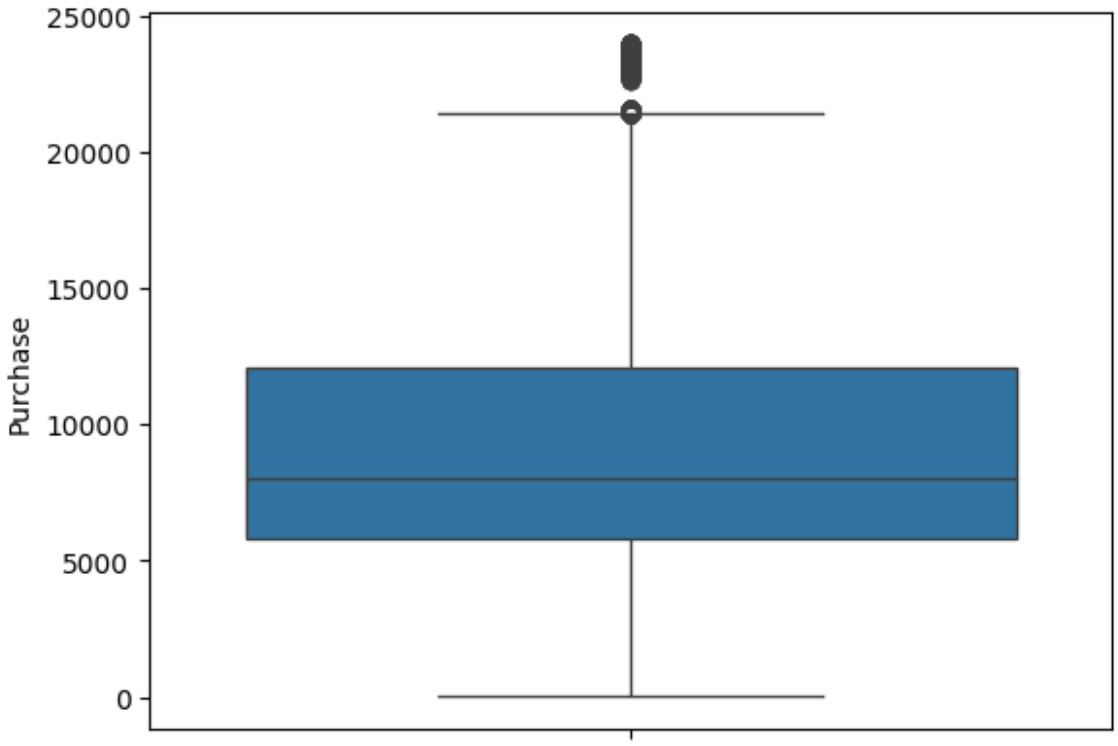
```
First quartile of purchase amount per transcation :  5823.0
medain amount of  purchase amount per transcation :  8047.0
Third quartile of purchase amount per transcation :  12054.0
```

```
iqr=d.loc['75%']-d.loc['25%']
print("Lower limit for outlier detetction : ",d.loc['25%']-1.5*iqr)
print("Upper limit for outlier detetction : ",d.loc['75%']+1.5*iqr)
```

```
Lower limit for outlier detetction :  -3523.5
Upper limit for outlier detetction :  21400.5
```

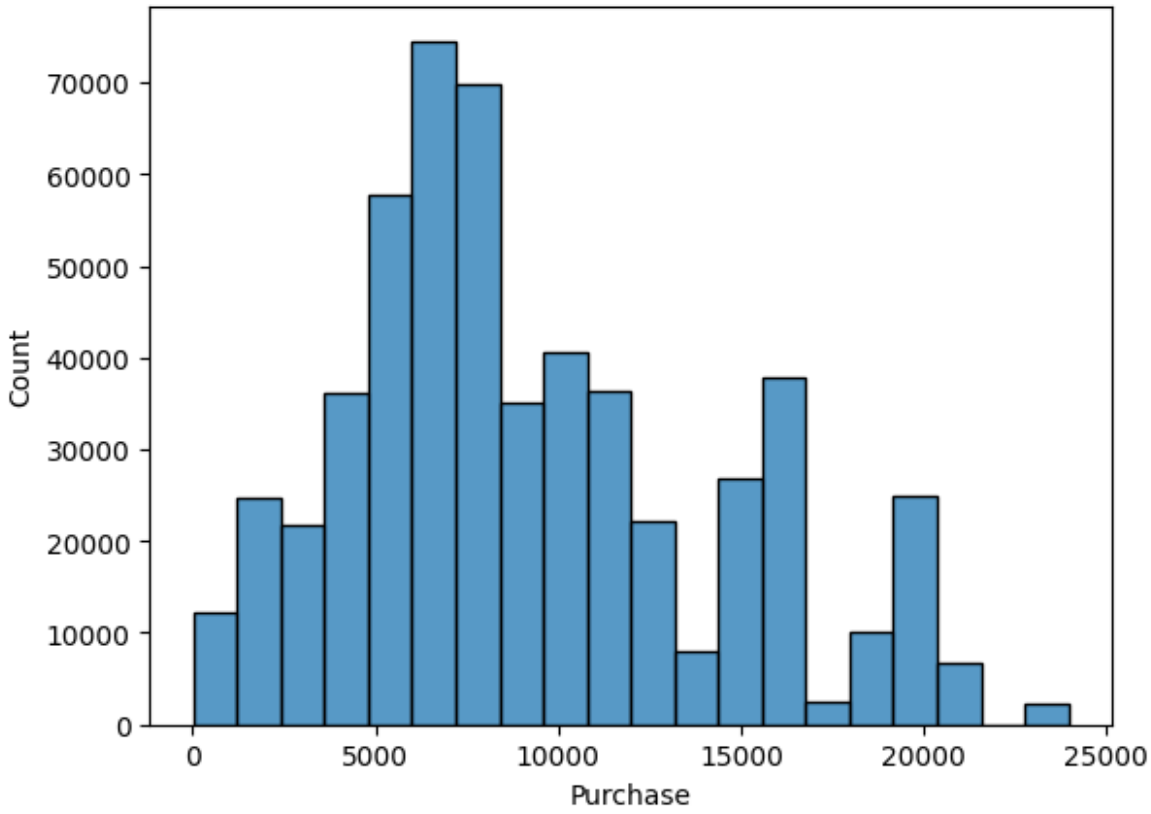
```
sns.boxplot(df['Purchase'])
```

```
<Axes: ylabel='Purchase'>
```




```
sns.histplot(df['Purchase'],bins=20)
```

```
<Axes: xlabel='Purchase', ylabel='Count'>
```




```
df.isnull().sum(axis=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
```

If we observe the above result there are no null values in every column

```
#Now we will check outliers in purchase amount
d=df['Purchase'].describe()
iqr=d.loc['75%']-d.loc['25%']
df.loc[(df['Purchase']>d['75%']+1.5*iqr)|(df['Purchase']<d['25%']-
1.5*iqr)]
```

```
{"summary":"{\n  \"name\": \"df\",\n  \"rows\": 2677,\n  \"fields\":
[\n    {\n      \"column\": \"User_ID\",\n      \"properties\": {\n
```

\"dtype\": \"number\",\n   \"std\": 1773,\n   \"min\":
1000017,\n   \"max\": 1006040,\n   \"num_unique_values\":
1487,\n   \"samples\": [\n   1002284,\n   1002211,\
n   1004070\n   ],\n   \"semantic_type\": \"\",\n
\"description\": \"\"\n   }\n   },\n   {\n   \"column\":
\"Product_ID\",\n   \"properties\": {\n   \"dtype\":
\"category\",\n   \"num_unique_values\": 48,\n
\"samples\": [\n   \"P00185442\",\n   \"P00144342\",\n
\"P00245942\"\n   ],\n   \"semantic_type\": \"\",\n
\"description\": \"\"\n   }\n   },\n   {\n   \"column\":
\"Gender\",\n   \"properties\": {\n   \"dtype\":
\"category\",\n   \"num_unique_values\": 2,\n   \"samples\":
[\n   \"F\",\n   \"M\"\n   ],\n
\"semantic_type\": \"\",\n   \"description\": \"\"\n   }\
n   },\n   {\n   \"column\": \"Age\",\n   \"properties\": {\n
\"dtype\": \"category\",\n   \"num_unique_values\": 7,\n
\"samples\": [\n   \"26-35\",\n   \"36-45\"\n   ],\
n   \"semantic_type\": \"\",\n   \"description\": \"\"\n
}\n   },\n   {\n   \"column\": \"Occupation\",\n
\"properties\": {\n   \"dtype\": \"category\",\n
\"num_unique_values\": 21,\n   \"samples\": [\n   \"2\",\n
\"15\"\n   ],\n   \"semantic_type\": \"\",\n
\"description\": \"\"\n   }\n   },\n   {\n   \"column\":
\"City_Category\",\n   \"properties\": {\n   \"dtype\":
\"category\",\n   \"num_unique_values\": 3,\n   \"samples\":
[\n   \"B\",\n   \"A\"\n   ],\n
\"semantic_type\": \"\",\n   \"description\": \"\"\n   }\
n   },\n   {\n   \"column\": \"Stay_In_Current_City_Years\",\n
\"properties\": {\n   \"dtype\": \"category\",\n
\"num_unique_values\": 5,\n   \"samples\": [\n   \"1\",\n
\"4+\"\n   ],\n   \"semantic_type\": \"\",\n
\"description\": \"\"\n   }\n   },\n   {\n   \"column\":
\"Marital_Status\",\n   \"properties\": {\n   \"dtype\":
\"category\",\n   \"num_unique_values\": 2,\n   \"samples\":
[\n   \"Married\",\n   \"Unmarried\"\n   ],\n
\"semantic_type\": \"\",\n   \"description\": \"\"\n   }\
n   },\n   {\n   \"column\": \"Product_Category\",\n
\"properties\": {\n   \"dtype\": \"category\",\n
\"num_unique_values\": 3,\n   \"samples\": [\n   \"10\",\n
\"15\"\n   ],\n   \"semantic_type\": \"\",\n
\"description\": \"\"\n   }\n   },\n   {\n   \"column\":
\"Purchase\",\n   \"properties\": {\n   \"dtype\":
\"number\",\n   \"std\": 701,\n   \"min\": 21401,\n
\"max\": 23961,\n   \"num_unique_values\": 1027,\n
\"samples\": [\n   23203,\n   23185\n   ],\n
\"semantic_type\": \"\",\n   \"description\": \"\"\n   }\
n   }\n]\n}","type":"dataframe"}

```
print("Percentage of outliers in the dataset :
",round(len(df.loc[(df['Purchase']>d['75%']+1.5*iqr)|
(df['Purchase']<d['25%']-1.5*iqr)])*100/len(df),2))
```

```
Percentage of outliers in the dataset :  0.49
```

Handling Outliers

Explanation:

Outliers are data points that deviate significantly from the rest of the dataset. While outliers can sometimes provide valuable insights, in this case, since the number of outliers is minimal, we can safely remove them to ensure the data is clean and suitable for further analysis.

1. **Why Remove Outliers?**
   - Outliers can skew statistical measures (mean, standard deviation) and distort models and visualizations.
   - Removing a small number of outliers will not impact the overall trends or patterns in the dataset.
2. **Steps to Identify and Remove Outliers:**
   - Use statistical techniques like the **Interquartile Range (IQR)** or standard deviation to identify outliers.
   - Filter out data points falling outside acceptable bounds.

```
df=df.loc[~(df['Purchase']>d['75%']+1.5*iqr)|(df['Purchase']<d['25%']-
1.5*iqr)].reset_index(drop=True)
df.head()
```

```
{"type":"dataframe","variable_name":"df"}
```

```
print("Number of customers : ",df['User_ID'].nunique())
```

```
Number of customers :  5891
```

```
print("Number of products in the dataset :
",df['Product_ID'].nunique())
```

```
Number of products in the dataset :  3631
```

```
#Top 10 products by number of purchases
df['Product_ID'].value_counts().reset_index().head(10)
```

```
{"summary":"{\n  \"name\": \"df['Product_ID']\",\n  \"rows\": 10,\n
\"fields\": [\n    {\n      \"column\": \"Product_ID\",\n
\"properties\": {\n        \"dtype\": \"string\",\n
\"num_unique_values\": 10,\n        \"samples\": [\n
\"P00059442\",\n          \"P00025442\",\n          \"P00184942\"\n
],\n        \"semantic_type\": \"\",\n        \"description\": \"\"\n
}\n    },\n    {\n      \"column\": \"count\",\n      \"properties\":
{\n        \"dtype\": \"number\",\n        \"std\": 149,\n
```

```
\"min\": 1406,\n        \"max\": 1880,\n        \"num_unique_values\":
9,\n        \"samples\": [\n          1422,\n          1615,\n
1440\n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    }\n  ]\n}","type":"dataframe"}
```

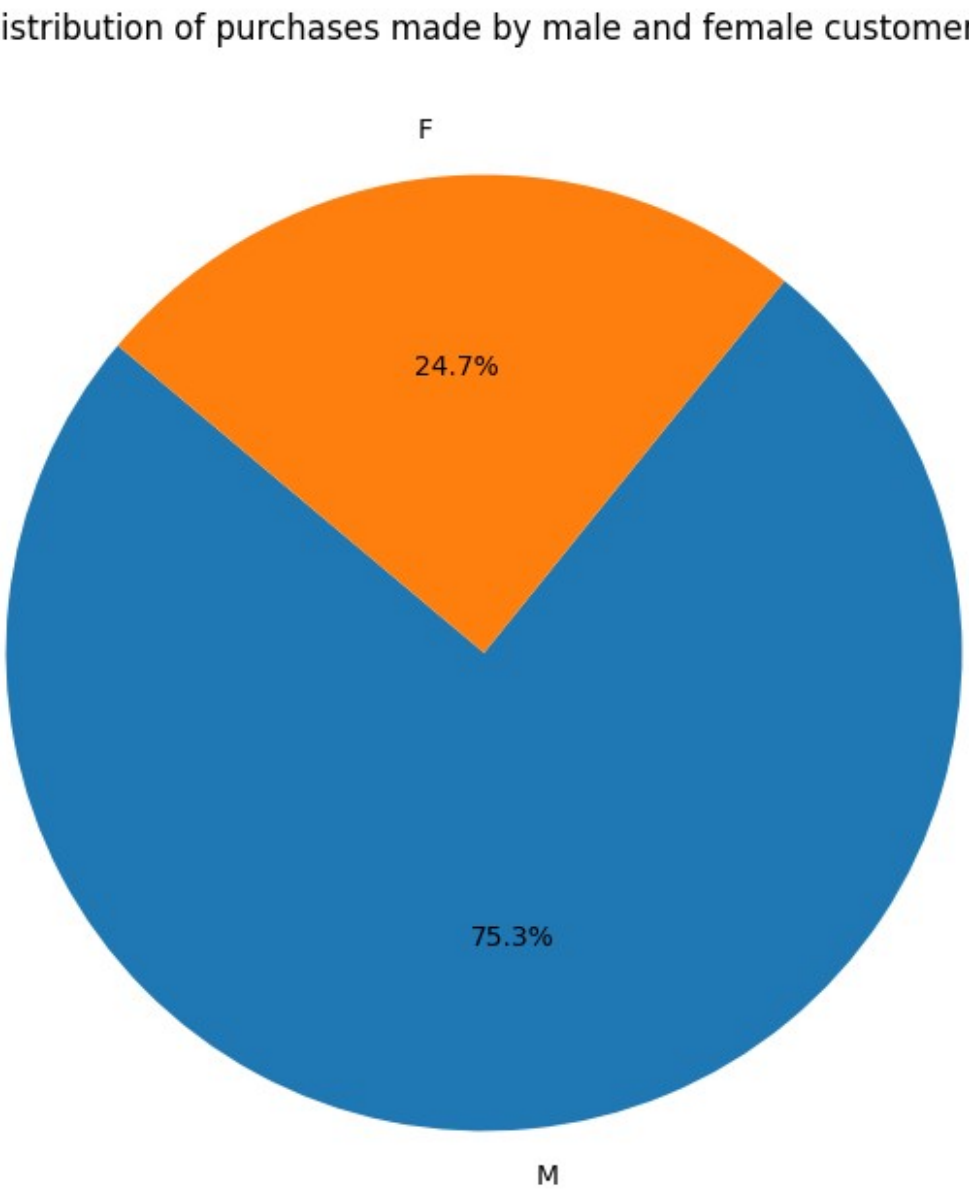
```
df_male=df[df['Gender']=='M']
df_female=df[df['Gender']=='F']
print("Number of male customers : ",df_male['User_ID'].nunique())
print("Number of female customers : ",df_female['User_ID'].nunique())
```

```
Number of male customers :  4225
Number of female customers :  1666
```

```
#Now let us find number of purchase by male and female customers
df['Gender'].value_counts().reset_index()
```

```
{"summary":"{\n  \"name\": \"df['Gender']\",\n  \"rows\": 2,\n
\"fields\": [\n    {\n      \"column\": \"Gender\",\n
\"properties\": {\n        \"dtype\": \"string\",\n
\"num_unique_values\": 2,\n        \"samples\": [\n          \"F\",\n
\"M\"\n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    },\n    {\n      \"column\":
\"count\",\n      \"properties\": {\n        \"dtype\": \"number\",\n
\"std\": 195833,\n        \"min\": 135220,\n        \"max\": 412171,\n
\"num_unique_values\": 2,\n        \"samples\": [\n          135220,\n
412171\n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    }\n  ]\n}","type":"dataframe"}
```

```
d=df['Gender'].value_counts().reset_index()
plt.figure(figsize=(8,8))  # Adjust the figure size as needed
plt.pie(d['count'], labels=d['Gender'], autopct='%1.1f%%',
startangle=140)
plt.title('Distribution of purchases made by male and female
customers')
plt.show()
```



Insight

From the above results, we can infer the following:

- **Gender Distribution in Purchases:**
  - Approximately **24.7%** of purchases are made by **female customers**.
  - Approximately **75.3%** of purchases are made by **male customers**.

Interpretation:

This indicates that the majority of purchases during the observed period are made by male customers. This insight could help in designing targeted marketing campaigns or promotions

that cater to the purchasing behavior of male customers, while also exploring opportunities to engage more female customers.

Recommendation

1. **Exclusive Deals:**
   - Introduce exclusive deals or discounts specifically targeted at female shoppers to encourage higher purchasing frequency.
   - Examples:
     - Discounts on popular product categories among women.
     - Loyalty programs with additional rewards for female customers.

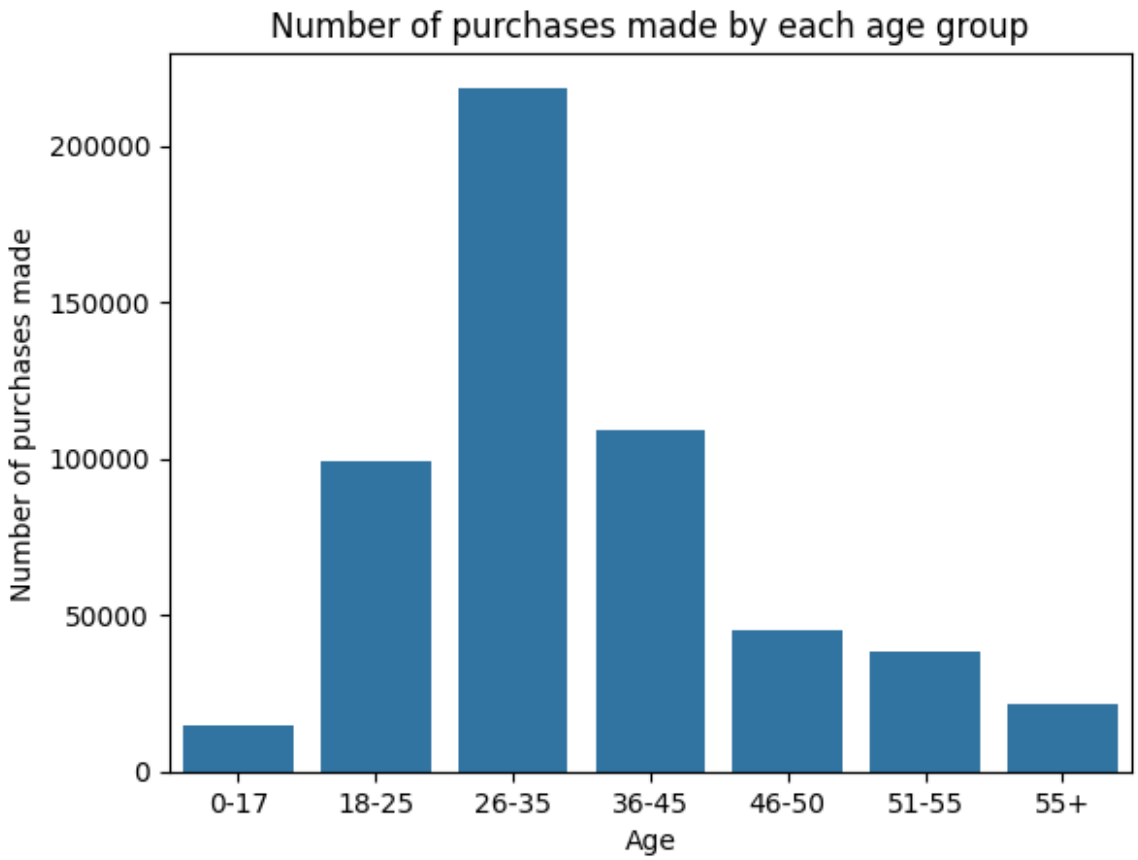
2. **Seasonal Promotions:**
   - Utilize seasonal or event-based campaigns to boost engagement with female customers.
   - Examples:
     - "Women's Month Specials" with exclusive offers on Black Friday.
     - Holiday promotions tailored to female shoppers' preferences.

By implementing these strategies, Walmart can potentially balance the purchase distribution between male and female customers and tap into untapped market opportunities.

```
df['Age'].value_counts().reset_index()
```


```
{"summary":"{\n  \"name\": \"df['Age']\",\n  \"rows\": 7,\n
\"fields\": [\n    {\n      \"column\": \"Age\",\n
\"properties\": {\n        \"dtype\": \"category\",\n
\"num_unique_values\": 7,\n        \"samples\": [\n          \"26-
35\",\n          \"36-45\",\n          \"55+\"\n        ],\n
\"semantic_type\": \"\",\n        \"description\": \"\"\n      }\
n    },\n    {\n      \"column\": \"count\",\n      \"properties\": {\
n        \"dtype\": \"number\",\n        \"std\": 71936,\n
\"min\": 15032,\n        \"max\": 218661,\n
\"num_unique_values\": 7,\n        \"samples\": [\n          218661,\n
109409,\n          21322\n        ],\n        \"semantic_type\":
\"\",\n        \"description\": \"\"\n      }\n    }\n  ]\
n}","type":"dataframe"}
```


```
d=df['Age'].value_counts().reset_index().sort_values(by="Age")
sns.barplot(x=d['Age'],y=d['count'])
plt.xlabel("Age")
plt.ylabel("Number of purchases made")
plt.title("Number of purchases made by each age group")
plt.show()
```



Insights

1. **Age Group Distribution in Purchases:**
   - The **26-35** age group made the **highest number of purchases**, followed by the **18-25** and **36-45** age groups.
2. **Moderate Purchase Activity:**
   - The **46-50** and **51-55** age groups also contributed a **good number of purchases**, but not as much as the top purchasing age group (26-35).
3. **Low Purchase Activity:**
   - The **0-17** and **55+** age groups recorded a **very low number of purchases**, indicating lesser engagement in shopping during the analyzed period.

Interpretation:

- The peak purchasing activity in the 26-35 age group aligns with the demographic known for active financial independence and family-oriented spending.
- Targeted campaigns can further boost engagement in the 46-50 and 51-55 age groups.
- Specific strategies are needed to attract the younger (0-17) and older (55+) age groups to increase their participation in purchases.

Recommendations

1. **For Age Groups 46-50 and 51-55:**
   - **Observation:** These groups have a noticeable but smaller share of purchases compared to the top-performing age groups.
   - **Strategies:**
     - Offer promotions on **health and wellness products**, which are likely to appeal to their lifestyle and priorities.
     - Introduce bundles or discounts on products related to hobbies or interests common in this demographic, such as home improvement, fitness, or travel essentials.

2. **For Age Groups 0-17 and 55+:**
   - **Observation:** These groups contribute the least to purchases, indicating an opportunity to increase their engagement.
   - **Strategies for 0-17:**
     - Focus on **family-oriented products** or promotions that involve parents, such as discounts on toys, school supplies, or gadgets.
     - Introduce campaigns around holidays or events that resonate with families (e.g., back-to-school sales or festive promotions).
   - **Strategies for 55+:**
     - Ensure products are **accessible** (e.g., user-friendly interfaces for online shopping, delivery assistance).
     - Promote items that are relevant to their needs, such as **healthcare products**, easy-to-use gadgets, or leisure-related goods.

Overall Goal:

These tailored strategies aim to enhance engagement and boost purchase activity across age groups that currently show lower participation, ensuring a broader reach and more inclusive marketing.

```
print("Number of unique occupations in the dataset :
",df['Occupation'].nunique())
```

```
Number of unique occupations in the dataset :  21
```

```
#Top 10 occupations by number of purchases
df['Occupation'].value_counts().reset_index().head(10)
```

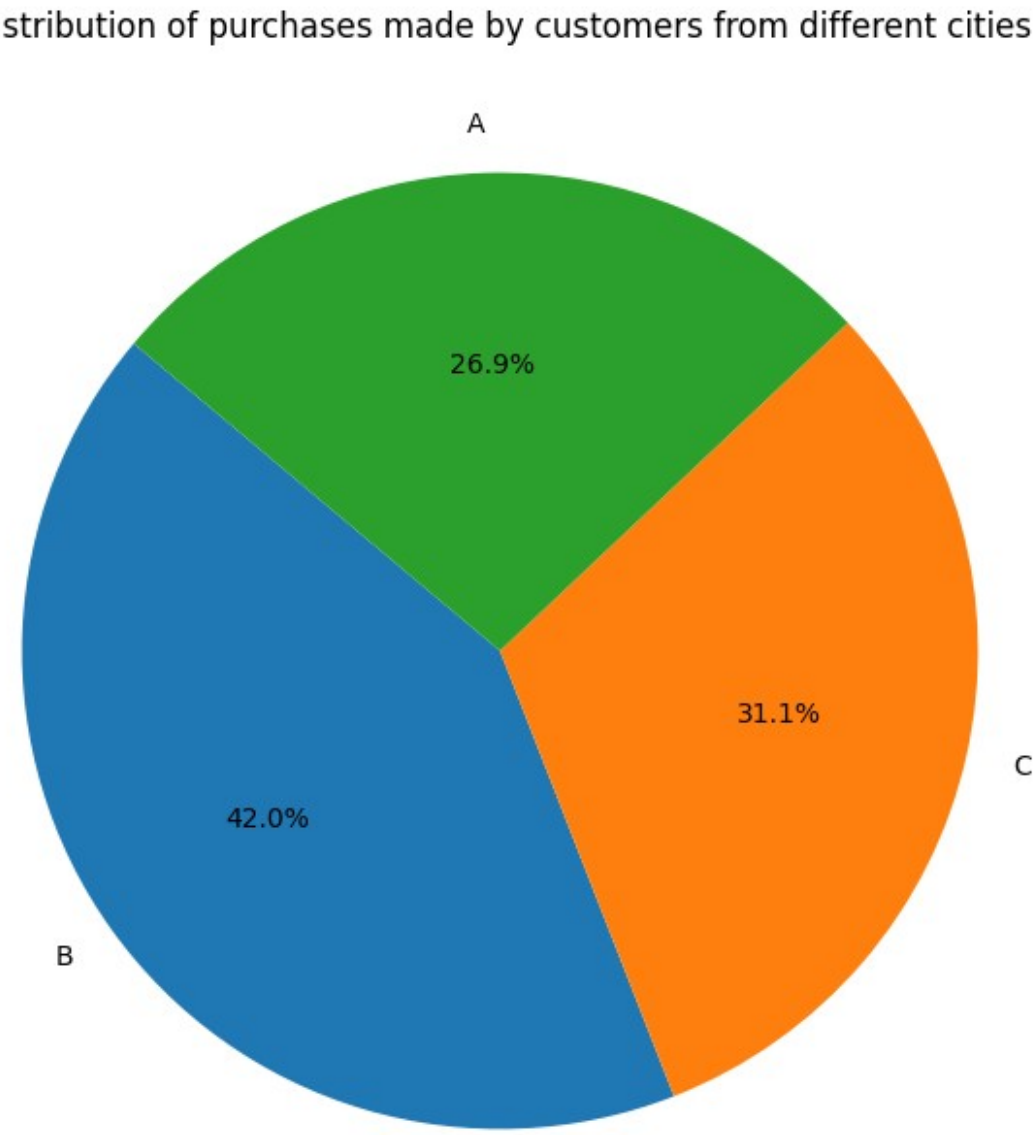

```
{"summary":"{\n  \"name\": \"df['Occupation']\",\n  \"rows\": 10,\n
\"fields\": [\n    {\n      \"column\": \"Occupation\",\n
\"properties\": {\n        \"dtype\": \"string\",\n
\"num_unique_values\": 10,\n        \"samples\": [\n          \"2\",\n
\"0\",\n          \"20\"\n        ],\n        \"semantic_type\":
\"\",\n        \"description\": \"\"\n      }\n    },\n    {\n
\"column\": \"count\",\n      \"properties\": {\n        \"dtype\":
\"number\",\n        \"std\": 17925,\n        \"min\": 25251,\n
```

\"max\": 72040,\n   \"num_unique_values\": 10,\n
\"samples\": [\n   26435,\n   69310,\n   33355\n
],\n   \"semantic_type\": \"\",\n   \"description\": \"\"\n
}\n   }\n]\n}","type":"dataframe"}

```
df['City_Category'].value_counts().reset_index()
```

{"summary":"{\n \"name\": \"df['City_Category']\",\n \"rows\": 3,\n
\"fields\": [\n   {\n   \"column\": \"City_Category\",\n
\"properties\": {\n   \"dtype\": \"string\",\n
\"num_unique_values\": 3,\n   \"samples\": [\n   \"B\",\n
\"C\",\n   \"A\"\n   ],\n   \"semantic_type\": \"\",\
n   \"description\": \"\"\n   }\n   },\n   {\n
\"column\": \"count\",\n   \"properties\": {\n   \"dtype\":
\"number\",\n   \"std\": 42866,\n   \"min\": 147036,\n
\"max\": 230114,\n   \"num_unique_values\": 3,\n
\"samples\": [\n   230114,\n   170241,\n
147036\n   ],\n   \"semantic_type\": \"\",\n
\"description\": \"\"\n   }\n   }\n]\n}","type":"dataframe"}

```
d=df['City_Category'].value_counts().reset_index()
plt.figure(figsize=(8,8))  # Adjust the figure size as needed
plt.pie(d['count'], labels=d['City_Category'], autopct='%1.1f%%',
startangle=140)
plt.title('Distribution of purchases made by customers from different
cities')
plt.show()
```



Insight

- **City Category Distribution in Purchases:**
  - The majority of purchases (approximately **42%**) are made by customers from **City_Category_B**.
  - The next highest share of purchases comes from **City_Category_C** at **31.1%**.
  - The least number of purchases are made by customers from **City_Category_A**, contributing **26.9%** of total purchases.

Interpretation:

- **City_Category_B** leads in purchases, which might reflect a larger or more active customer base in that category. This could also indicate better availability of products or more successful marketing strategies in this category.
- **City_Category_C** shows significant engagement, but there might still be room for improvement in driving more purchases.
- **City_Category_A** has the smallest share, which could suggest lower customer activity or potential barriers to purchase that need to be addressed.

This insight helps in understanding how purchases are distributed across different city categories, offering opportunities for targeted promotions and better resource allocation.

Recommendations

1. **Enhance Store Presence for City_Category_A:**
   - **Observation:** City_Category_A has the smallest share of purchases (26.9%).
   - **Strategies:**
     - **Enhance the in-store experience** by introducing new features such as interactive displays, better product visibility, or more personalized shopping assistance.
     - Consider launching **new product lines** that may specifically cater to the preferences of customers in this city category.

2. **Customer Engagement in City_Category_A:**
   - **Observation:** Lower customer engagement may be affecting purchase behavior in City_Category_A.
   - **Strategies:**
     - Implement **loyalty programs** to encourage repeat purchases and reward long-term customers.
     - Organize **community events** (e.g., product demonstrations, local collaborations) to build a stronger connection with customers and encourage foot traffic to stores.

Overall Goal:

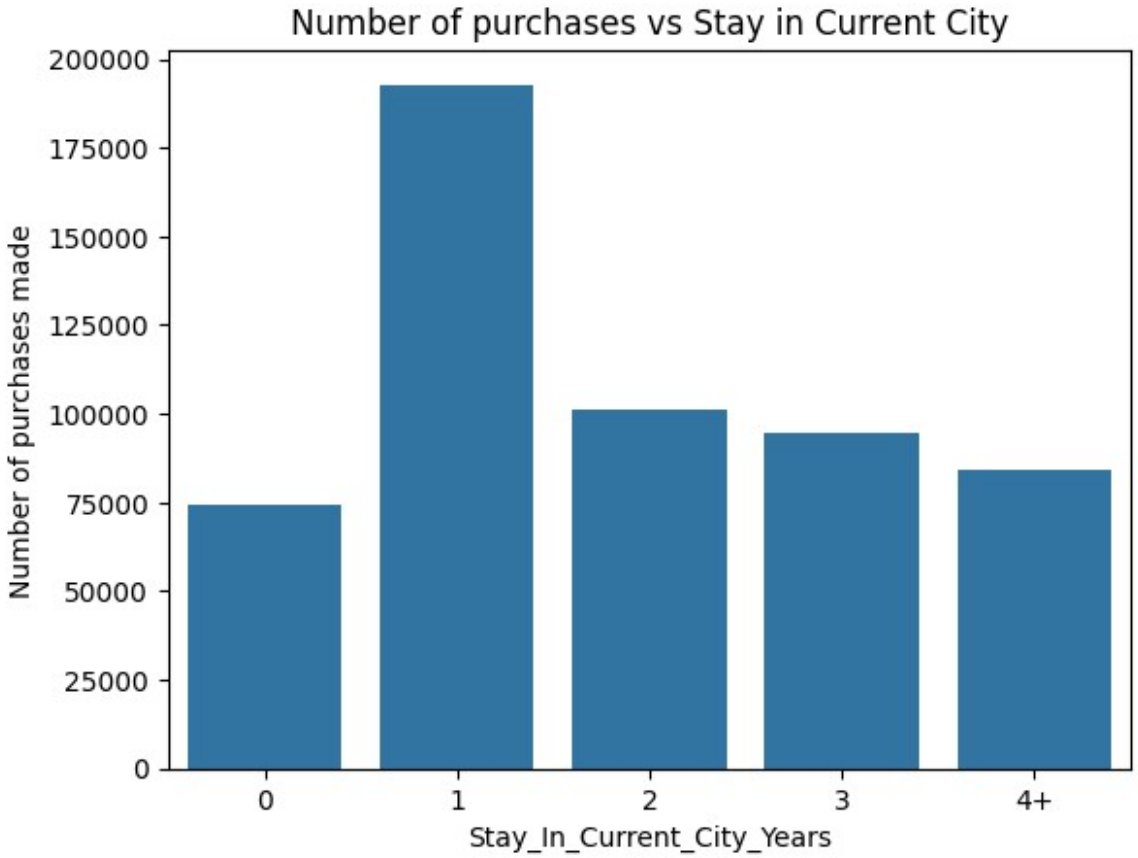
By improving the shopping experience and increasing engagement in **City_Category_A**, Walmart can tap into the potential of this city category and drive more sales, while also fostering loyalty and long-term customer relationships.

```
df['Stay_In_Current_City_Years'].value_counts().reset_index()
```


```
{"summary":"{\n  \"name\": \"df['Stay_In_Current_City_Years']\",\n
\"rows\": 5,\n  \"fields\": [\n    {\n      \"column\":
\"Stay_In_Current_City_Years\",\n      \"properties\": {\n
\"dtype\": \"category\",\n        \"num_unique_values\": 5,\n
\"samples\": [\n          \"2\",\n          \"0\",\n          \"3\"\n
],\n        \"semantic_type\": \"\",\n        \"description\": \"\"\n
}\n    },\n    {\n      \"column\": \"count\",\n      \"properties\":
{\n        \"dtype\": \"number\",\n        \"std\": 47748,\n
```

```
\"min\": 74036,\n        \"max\": 192845,\n
\"num_unique_values\": 5,\n        \"samples\": [\n          101384,\n
74036,\n          94804\n        ],\n        \"semantic_type\": \"\",\
n        \"description\": \"\"\n      }\n    }\n  ]\
n}","type":"dataframe"}
```

```
d=df['Stay_In_Current_City_Years'].value_counts().reset_index().sort_v
alues(by="Stay_In_Current_City_Years")
sns.barplot(x=d['Stay_In_Current_City_Years'],y=d['count'])
plt.xlabel("Stay_In_Current_City_Years")
plt.ylabel("Number of purchases made")
plt.title("Number of purchases vs Stay in Current City")
plt.show()
```



Insight

1. **Customer Tenure and Purchases:**
   - The highest number of purchases come from customers who have been living in the current city for **1 year**.
2. **Moderate Purchases by Longer Tenure:**
   - Customers who have been living in the city for **2-3 years** also made a good amount of purchases, but their share is about **half** of the customers living for 1 year.

3. **Lower Purchases by Customers with Extreme Tenure:**
   - Customers living in the city for **less than 1 year** or for **more than 4 years** show lower purchase activity compared to the 1-3 year group, but their purchases are not too low.

Interpretation:

- **Customers living in the city for 1 year** appear to be the most engaged, possibly because they are still in the process of settling in and exploring the available products or services.
- **Customers living for 2-3 years** have established a purchasing routine, but their spending is less compared to newer arrivals.
- **Customers with extreme tenure** (less than 1 year or more than 4 years) may not engage as much, possibly due to factors like seasonal changes or a shift in shopping habits over time.

This insight can guide targeted strategies for engaging different customer segments based on their length of residence in the city.

Recommendations

1. **Incentives for Short-Term Residents (Less than 1 Year):**

- **Observation:** Customers living in the city for less than 1 year have lower purchase frequency.
- **Strategies:**
  - Offer **special promotions** or **discounts** aimed at encouraging frequent visits and purchases.
  - Introduce **welcome offers** or **first-time buyer incentives** to make new residents feel more connected to the store.
  - Provide **tailored suggestions** based on their initial purchase history, helping them explore relevant products and build shopping habits.

2. **Reconnect with Long-Term Residents (More than 4 Years):**

- **Observation:** Customers living for more than 4 years show reduced engagement.
- **Strategies:**
  - **Review purchasing patterns** to understand any changes in preferences or purchasing behavior over time.
  - Develop **personalized offers** based on their past purchase history to rekindle interest in shopping.
  - Host **exclusive events** (e.g., VIP sales, special offers) for long-term residents to make them feel valued and encourage more frequent visits.

Overall Goal:

By providing targeted incentives to **short-term residents** and personalized re-engagement strategies for **long-term residents**, Walmart can foster loyalty and increase purchasing frequency across all customer groups, ensuring better customer retention and satisfaction.

```
#Top 10 product categories by number of purchases
df['Product_Category'].value_counts().reset_index().head(10)
```


```
{"summary":"{\n  \"name\": \"df['Product_Category']\",\n  \"rows\":
10,\n  \"fields\": [\n    {\n      \"column\": \"Product_Category\",\n
\"properties\": {\n        \"dtype\": \"string\",\n
\"num_unique_values\": 10,\n        \"samples\": [\n          \"16\",\
n          \"1\",\n          \"6\"\n        ],\n
\"semantic_type\": \"\",\n        \"description\": \"\"\n      }\
n    },\n    {\n      \"column\": \"count\",\n      \"properties\": {\
n        \"dtype\": \"number\",\n        \"std\": 58228,\n
\"min\": 5963,\n        \"max\": 150933,\n
\"num_unique_values\": 10,\n        \"samples\": [\n          9828,\n
140378,\n          20466\n        ],\n        \"semantic_type\":
\"\",\n        \"description\": \"\"\n      }\n    }\n  ]\
n}","type":"dataframe"}
```


```
#Top 10 products by average purchase amount per transcation
df.groupby("Product_ID")
['Purchase'].mean().reset_index().sort_values(by="Purchase",ascending=
False).head(10)
```


```
{"summary":"{\n  \"name\": \"df\",\n  \"rows\": 10,\n  \"fields\": [\n
{\n      \"column\": \"Product_ID\",\n      \"properties\": {\n
\"dtype\": \"string\",\n        \"num_unique_values\": 10,\n
\"samples\": [\n          \"P00005042\",\n          \"P00341542\",\n
\"P00071442\"\n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    },\n    {\n      \"column\":
\"Purchase\",\n      \"properties\": {\n        \"dtype\":
\"number\",\n        \"std\": 782.3516218508613,\n        \"min\":
18198.816341287056,\n        \"max\": 20323.0,\n
\"num_unique_values\": 10,\n        \"samples\": [\n
18298.47987616099,\n          20291.0,\n          18599.061823802163\n
],\n        \"semantic_type\": \"\",\n        \"description\": \"\"\n
}\n    }\n  ]\n}","type":"dataframe"}
```


```
#Average purchase amount per transcation by gender
df.groupby('Gender')['Purchase'].mean().reset_index()
```


```
{"summary":"{\n  \"name\": \"df\",\n  \"rows\": 2,\n  \"fields\": [\n
{\n      \"column\": \"Gender\",\n      \"properties\": {\n
\"dtype\": \"string\",\n        \"num_unique_values\": 2,\n
\"samples\": [\n          \"M\",\n          \"F\"\n        ],\n
\"semantic_type\": \"\",\n        \"description\": \"\"\n      }\
n    },\n    {\n      \"column\": \"Purchase\",\n      \"properties\":
{\n        \"dtype\": \"number\",\n        \"std\":
492.6238402951281,\n        \"min\": 8671.049038603756,\n
\"max\": 9367.724354697444,\n        \"num_unique_values\": 2,\n
\"samples\": [\n          9367.724354697444,\n
```

8671.049038603756\n   ],\n   \"semantic_type\": \"\",\n
\"description\": \"\"\n   }\n   }\n]\n}","type":"dataframe"}

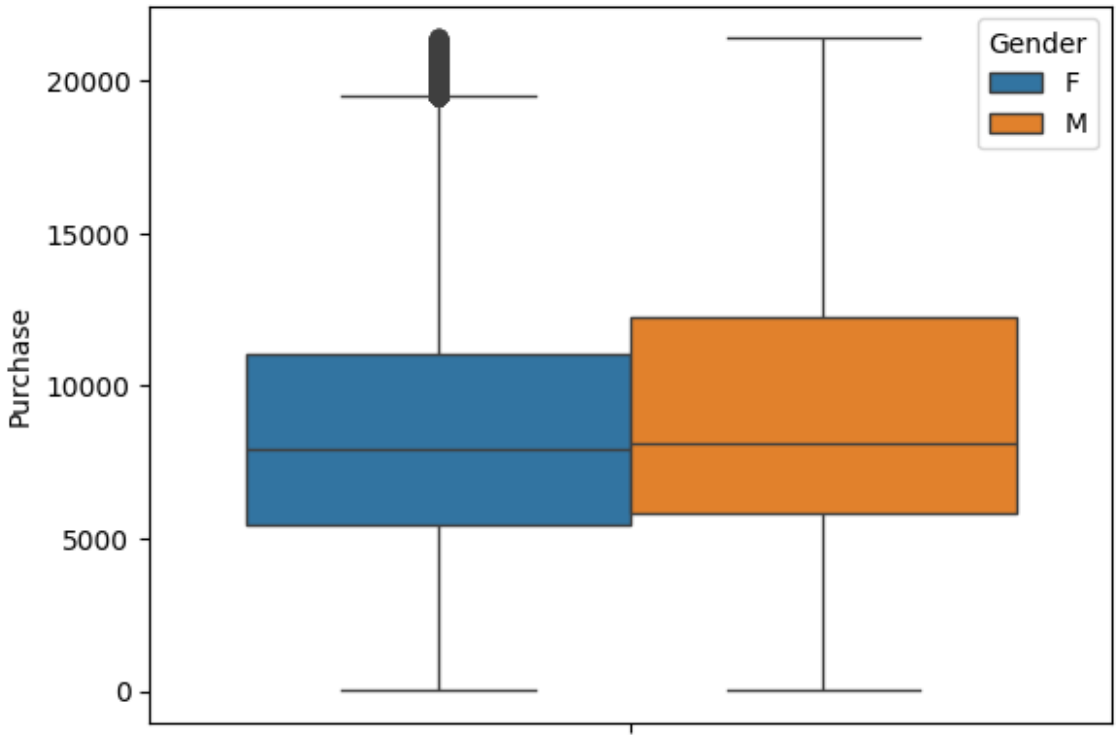
```
df.groupby("Gender")['Purchase'].describe()
```

{"summary":"{\n \"name\": \"df\",\n \"rows\": 2,\n \"fields\": [\n
{\n   \"column\": \"Gender\",\n   \"properties\": {\n
\"dtype\": \"string\",\n   \"num_unique_values\": 2,\n
\"samples\": [\n   \"M\",\n   \"F\"\n   ],\n
\"semantic_type\": \"\",\n   \"description\": \"\"\n   }\
n   },\n   {\n   \"column\": \"count\",\n   \"properties\": {\
n   \"dtype\": \"number\",\n   \"std\": 195833.93015639554,\
n   \"min\": 135220.0,\n   \"max\": 412171.0,\n
\"num_unique_values\": 2,\n   \"samples\": [\n
412171.0,\n   135220.0\n   ],\n   \"semantic_type\":
\"\",\n   \"description\": \"\"\n   }\n   },\n   {\n
\"column\": \"mean\",\n   \"properties\": {\n   \"dtype\":
\"number\",\n   \"std\": 492.6238402951281,\n   \"min\":
8671.049038603756,\n   \"max\": 9367.724354697444,\n
\"num_unique_values\": 2,\n   \"samples\": [\n
9367.724354697444,\n   8671.049038603756\n   ],\n
\"semantic_type\": \"\",\n   \"description\": \"\"\n   }\
n   },\n   {\n   \"column\": \"std\",\n   \"properties\": {\n
\"dtype\": \"number\",\n   \"std\": 233.46940918050436,\n
\"min\": 4679.058483084379,\n   \"max\": 5009.234087946682,\n
\"num_unique_values\": 2,\n   \"samples\": [\n
5009.234087946682,\n   4679.058483084379\n   ],\n
\"semantic_type\": \"\",\n   \"description\": \"\"\n   }\
n   },\n   {\n   \"column\": \"min\",\n   \"properties\": {\n
\"dtype\": \"number\",\n   \"std\": 0.0,\n   \"min\": 12.0,\
n   \"max\": 12.0,\n   \"num_unique_values\": 1,\n
\"samples\": [\n   12.0\n   ],\n   \"semantic_type\":
\"\",\n   \"description\": \"\"\n   }\n   },\n   {\n
\"column\": \"25%\",\n   \"properties\": {\n   \"dtype\":
\"number\",\n   \"std\": 299.10616844190963,\n   \"min\":
5429.0,\n   \"max\": 5852.0,\n   \"num_unique_values\": 2,\n
\"samples\": [\n   5852.0\n   ],\n
\"semantic_type\": \"\",\n   \"description\": \"\"\n   }\
n   },\n   {\n   \"column\": \"50%\",\n   \"properties\": {\n
\"dtype\": \"number\",\n   \"std\": 129.4005409571382,\n
\"min\": 7906.0,\n   \"max\": 8089.0,\n
\"num_unique_values\": 2,\n   \"samples\": [\n   8089.0\n
],\n   \"semantic_type\": \"\",\n   \"description\": \"\"\n
}\n   },\n   {\n   \"column\": \"75%\",\n   \"properties\": {\
n   \"dtype\": \"number\",\n   \"std\": 836.5073221436858,\n
\"min\": 11064.0,\n   \"max\": 12247.0,\n
\"num_unique_values\": 2,\n   \"samples\": [\n   12247.0\n
],\n   \"semantic_type\": \"\",\n   \"description\": \"\"\n
}\n   },\n   {\n   \"column\": \"max\",\n   \"properties\": {\
n   \"dtype\": \"number\",\n   \"std\": 0.7071067811865476,\

```
n        \"min\": 21398.0,\n        \"max\": 21399.0,\n
\"num_unique_values\": 2,\n        \"samples\": [\n          21399.0\n
],\n        \"semantic_type\": \"\",\n        \"description\": \"\"\n
}\n    }\n  ]\n}","type":"dataframe"}
```

```
sns.boxplot(y="Purchase",hue="Gender",data=df)
```

```
<Axes: ylabel='Purchase'>
```



Insight

1. **Median Purchase Amount:**
   - The **median purchase amount** for **male customers** (**8089.0**) is higher than that for **female customers** (**7906.0**).
2. **Spread of Purchase Amount:**
   - The **spread** (measured by standard deviation) of the purchase amount for **male customers** (**5009**) is higher than that for **female customers** (**4679**), indicating greater variability in male customers' spending behavior.

Interpretation:

- **Male customers** tend to make purchases of slightly higher value on average, with a wider range of spending amounts.
- **Female customers** show a lower median purchase amount but have a more consistent purchasing pattern compared to male customers.

This insight suggests that while male customers may spend more overall, their spending behavior is more varied, whereas female customers tend to make more predictable or consistent purchases.

Recommendations

1. **For Male Customers:**
   - **Observation:** Male customers have a higher median purchase amount and a greater spread in their spending.
   - **Strategies:**
     - **High-Value Promotions:** Since the median purchase amount is higher for males, create **high-value promotions** or **loyalty programs** that encourage larger purchases. These could include **premium product discounts**, **exclusive offers**, or **VIP access** to special deals.
     - **Personalized Offers:** Use data to tailor promotions for high-spending male customers, particularly targeting higher-end or **luxury products** that align with their purchasing patterns.

2. **For Female Customers:**
   - **Observation:** Female customers have a slightly lower median purchase amount, indicating opportunities to boost their spending.
   - **Strategies:**
     - **Increase Engagement:** Develop strategies to **increase spending** among female customers. This could include **targeted discounts**, **bundle deals**, or **upsell opportunities** designed to encourage larger purchases.
     - **Product Recommendations:** Based on their purchasing behavior, highlight **products with higher purchase values** that may appeal to female customers, potentially offering related items or premium options that align with their interests.

Overall Goal:

By providing tailored, gender-specific promotions and personalized offers, Walmart can effectively boost spending across both male and female customer groups while increasing overall sales and engagement.

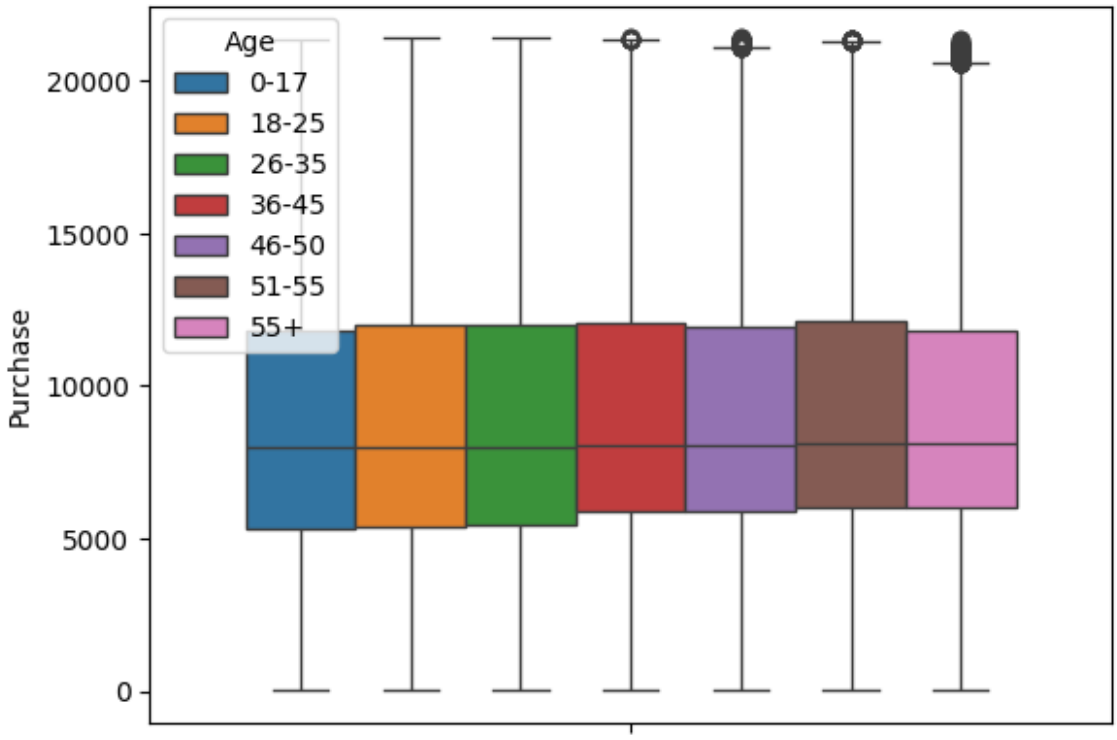
```
df.groupby("Age")['Purchase'].describe()
```


```
{"summary":"{\n  \"name\": \"df\",\n  \"rows\": 7,\n  \"fields\": [\n
{\n      \"column\": \"Age\",\n      \"properties\": {\n
\"dtype\": \"category\",\n        \"num_unique_values\": 7,\n
\"samples\": [\n          \"0-17\",\n          \"18-25\",\n
\"51-55\"\n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    },\n    {\n      \"column\":
\"count\",\n      \"properties\": {\n        \"dtype\": \"number\",\n
\"std\": 71936.37223202712,\n        \"min\": 15032.0,\n
\"max\": 218661.0,\n        \"num_unique_values\": 7,\n
\"samples\": [\n          15032.0,\n          99334.0,\n
```

```
38191.0\n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    },\n    {\n      \"column\":
\"mean\",\n      \"properties\": {\n        \"dtype\": \"number\",\n
\"std\": 167.91638398868866,\n        \"min\": 8867.447046301224,\n
\"max\": 9423.121704066403,\n        \"num_unique_values\": 7,\n
\"samples\": [\n          8867.447046301224,\n
9124.031731330662,\n          9423.121704066403\n        ],\n
\"semantic_type\": \"\",\n        \"description\": \"\"\n      }\
n    },\n    {\n      \"column\": \"std\",\n      \"properties\": {\n
\"dtype\": \"number\",\n        \"std\": 59.57373173002151,\n
\"min\": 4861.62659605644,\n        \"max\": 5030.052845920101,\n
\"num_unique_values\": 7,\n        \"samples\": [\n
5030.052845920101,\n          4978.831061893425,\n
4953.644649867353\n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    },\n    {\n      \"column\":
\"min\",\n      \"properties\": {\n        \"dtype\": \"number\",\n
\"std\": 0.0,\n        \"min\": 12.0,\n        \"max\": 12.0,\n
\"num_unique_values\": 1,\n        \"samples\": [\n          12.0\n
],\n        \"semantic_type\": \"\",\n        \"description\": \"\"\n
}\n    },\n    {\n      \"column\": \"25%\",\n      \"properties\": {\
n        \"dtype\": \"number\",\n        \"std\": 295.5880337806077,\n
\"min\": 5324.0,\n        \"max\": 6007.0,\n
\"num_unique_values\": 6,\n        \"samples\": [\n          5324.0\n
],\n        \"semantic_type\": \"\",\n        \"description\": \"\"\n
}\n    },\n    {\n      \"column\": \"50%\",\n      \"properties\": {\
n        \"dtype\": \"number\",\n        \"std\": 48.594164846644475,\
n        \"min\": 7974.5,\n        \"max\": 8118.0,\n
\"num_unique_values\": 7,\n        \"samples\": [\n          7974.5\n
],\n        \"semantic_type\": \"\",\n        \"description\": \"\"\n
}\n    },\n    {\n      \"column\": \"75%\",\n      \"properties\": {\
n        \"dtype\": \"number\",\n        \"std\": 109.45746890916124,\
n        \"min\": 11833.25,\n        \"max\": 12123.0,\n
\"num_unique_values\": 7,\n        \"samples\": [\n          11833.25\
n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    },\n    {\n      \"column\":
\"max\",\n      \"properties\": {\n        \"dtype\": \"number\",\n
\"std\": 25.37340189666186,\n        \"min\": 21342.0,\n
\"max\": 21399.0,\n        \"num_unique_values\": 6,\n
\"samples\": [\n          21342.0\n        ],\n
\"semantic_type\": \"\",\n        \"description\": \"\"\n      }\
n    }\n  ]\n}","type":"dataframe"}
```


```
sns.boxplot(y="Purchase",hue="Age",data=df)
```

```
<Axes: ylabel='Purchase'>
```



Insight

- **Age Groups and Purchase Amount:**
  - Upon reviewing the **boxplot**, it appears that all **age groups** have similar **median purchase amounts**. This suggests that, on average, the spending behavior across age groups is relatively consistent.
  - However, there are differences in the **spread** (variation) of the purchase amounts across the age groups. Some groups exhibit a **lesser spread**, meaning their spending behavior is more consistent, while others show a **wider spread**, indicating more variability in their purchase amounts.

Interpretation:

- The similar **median purchase amounts** across age groups suggest that age may not have a significant impact on average spending.
- The differing **spread** in purchase amounts could indicate that certain age groups exhibit more diverse spending behavior, while others show more stable and predictable purchasing patterns.

This insight can help in identifying which age groups have more consistent purchasing habits and which groups may require more targeted marketing or promotions to manage their spending variability.

Recommendations

1. **Age-Specific Promotions:**
   - **Observation:** Different age groups exhibit varying levels of spread in their purchase amounts.
   - **Strategies:**
     - For age groups with **lower spread** (more consistent spending), create promotions that emphasize **value for money**, such as **discount bundles** or **loyalty rewards**. This approach will resonate well with customers who prefer consistent, lower-cost purchases.
     - For age groups with **higher spread** (more varied spending), focus on promoting **luxury or premium products** with exclusive offers, aiming to attract higher-value purchases from customers willing to spend more.

2. **Adjust Frequency of Promotions:**
   - **Observation:** Age groups with higher spread exhibit more variability in their spending.
   - **Strategies:**
     - For groups with a higher spread in their spending behavior, adjust the **frequency and type of promotions** to stabilize their purchasing patterns. Introduce **limited-time offers**, **exclusive discounts**, or **flash sales** to encourage more consistent and predictable spending from these customers.

Overall Goal:

By tailoring **age-specific promotions** and adjusting the **frequency** of offers, Walmart can better target customers based on their spending behavior, improving customer satisfaction and increasing overall sales across all age groups.

```
#Top 10 occupations by average purchase amount per transcation
df.groupby("Occupation")
['Purchase'].mean().reset_index().sort_values(by="Purchase",ascending=
False).head(10)
```


```
{"summary":"{\n  \"name\": \"df\",\n  \"rows\": 10,\n  \"fields\": [\n
{\n      \"column\": \"Occupation\",\n      \"properties\": {\n
\"dtype\": \"string\",\n        \"num_unique_values\": 10,\n
\"samples\": [\n          \"13\",\n          \"12\",\n          \"7\"\
n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    },\n    {\n      \"column\":
\"Purchase\",\n      \"properties\": {\n        \"dtype\":
\"number\",\n        \"std\": 212.90658376585753,\n        \"min\":
9191.133260944673,\n        \"max\": 9758.679086689248,\n
\"num_unique_values\": 10,\n        \"samples\": [\n
9194.099386983175,\n          9717.192385868688,\n
9365.188025477706\n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    }\n  ]\n}","type":"dataframe"}
```

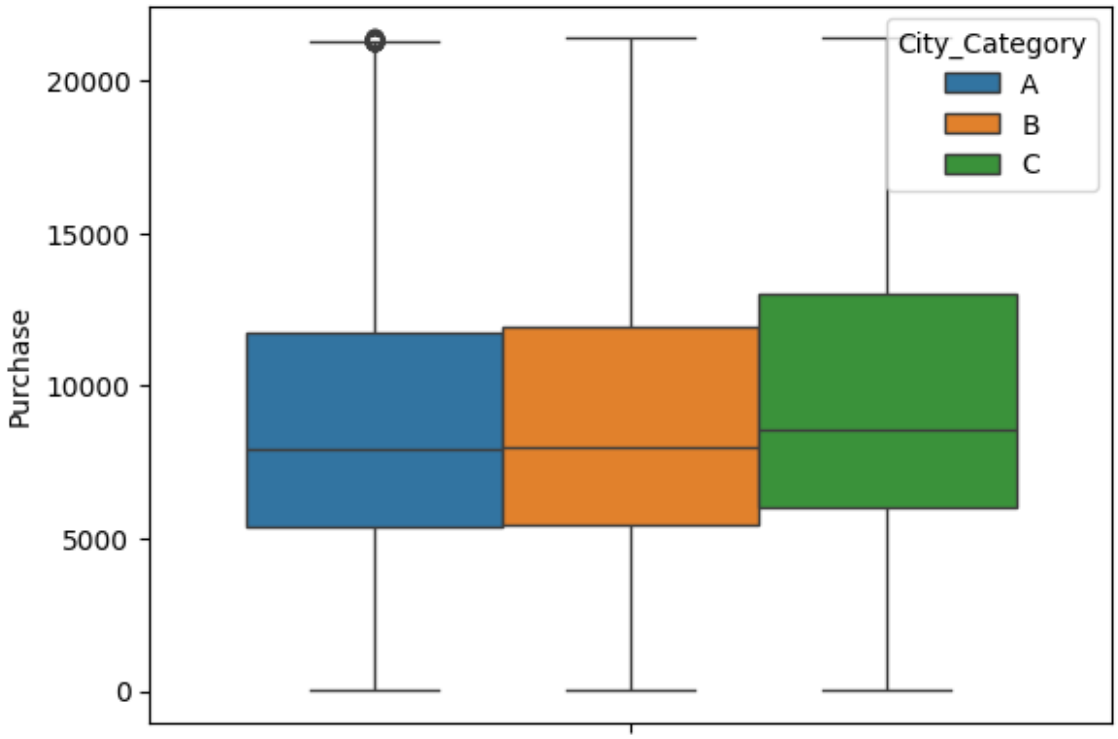
```
#Average purchase amount per transcation for each city category
df.groupby("City_Category")['Purchase'].describe()
```


```
{"summary":"{\n  \"name\": \"df\",\n  \"rows\": 3,\n  \"fields\": [\n{\n      \"column\": \"City_Category\",\n      \"properties\": {\n\"dtype\": \"string\",\n        \"num_unique_values\": 3,\n\"samples\": [\n          \"A\",\n          \"B\",\n          \"C\"\n],\n        \"semantic_type\": \"\",\n        \"description\": \"\"\n}\n    },\n    {\n      \"column\": \"count\",\n      \"properties\": {\n        \"dtype\": \"number\",\n        \"std\": 42866.46365555868,\n        \"min\": 147036.0,\n        \"max\": 230114.0,\n        \"num_unique_values\": 3,\n        \"samples\": [\n147036.0,\n          230114.0,\n          170241.0\n        ],\n\"semantic_type\": \"\",\n        \"description\": \"\"\n      }\n    },\n    {\n      \"column\": \"mean\",\n      \"properties\": {\n\"dtype\": \"number\",\n        \"std\": 410.535594240973,\n\"min\": 8845.367393019396,\n        \"max\": 9645.647300004111,\n\"num_unique_values\": 3,\n        \"samples\": [\n8845.367393019396,\n          9086.502707353746,\n9645.647300004111\n        ],\n        \"semantic_type\": \"\",\n\"description\": \"\"\n      }\n    },\n    {\n      \"column\": \"std\",\n      \"properties\": {\n        \"dtype\": \"number\",\n\"std\": 157.5511814454928,\n        \"min\": 4804.639577084628,\n\"max\": 5105.3636633665665,\n        \"num_unique_values\": 3,\n\"samples\": [\n          4804.639577084628,\n4873.509950270737,\n          5105.3636633665665\n        ],\n\"semantic_type\": \"\",\n        \"description\": \"\"\n      }\n    },\n    {\n      \"column\": \"min\",\n      \"properties\": {\n\"dtype\": \"number\",\n        \"std\": 0.0,\n        \"min\": 12.0,\n        \"max\": 12.0,\n        \"num_unique_values\": 1,\n\"samples\": [\n          12.0\n        ],\n        \"semantic_type\": \"\",\n        \"description\": \"\"\n      }\n    },\n    {\n\"column\": \"25%\",\n      \"properties\": {\n        \"dtype\": \"number\",\n        \"std\": 344.4159307194331,\n        \"min\": 5398.0,\n        \"max\": 6021.0,\n        \"num_unique_values\": 3,\n\"samples\": [\n          5398.0\n        ],\n\"semantic_type\": \"\",\n        \"description\": \"\"\n      }\n    },\n    {\n      \"column\": \"50%\",\n      \"properties\": {\n\"dtype\": \"number\",\n        \"std\": 355.27031961592286,\n\"min\": 7922.0,\n        \"max\": 8571.0,\n\"num_unique_values\": 3,\n        \"samples\": [\n          7922.0\n],\n        \"semantic_type\": \"\",\n        \"description\": \"\"\n}\n    },\n    {\n      \"column\": \"75%\",\n      \"properties\": {\n        \"dtype\": \"number\",\n        \"std\": 700.6470818702761,\n\"min\": 11747.0,\n        \"max\": 13050.0,\n\"num_unique_values\": 3,\n        \"samples\": [\n          11747.0\n],\n        \"semantic_type\": \"\",\n        \"description\": \"\"\n}\n    },\n    {\n      \"column\": \"max\",\n      \"properties\": {\n        \"dtype\": \"number\",\n        \"std\": 0.5773502691896258,\n        \"min\": 21398.0,\n        \"max\": 21399.0,\n
```

```
\"num_unique_values\": 2,\n        \"samples\": [\n          21399.0\n
],\n        \"semantic_type\": \"\",\n        \"description\": \"\"\n
}\n    }\n  ]\n}","type":"dataframe"}
```

```
sns.boxplot(y="Purchase",hue="City_Category",hue_order=['A','B','C'],d
ata=df)
```

```
<Axes: ylabel='Purchase'>
```



Insight

- **City Categories and Purchase Amount:**
  - Upon reviewing the **boxplot**, we can observe the following:
    - **City_Category_A** and **City_Category_B** have **similar median** and **spread** in their purchase amounts, indicating that the average spending and its variability are relatively consistent across these two categories.
    - **City_Category_C**, however, has a **slightly higher median** and **spread**, suggesting that purchases in this category tend to be higher on average, with more variability in the spending behavior.

Interpretation:

- **City_Category_A** and **City_Category_B** exhibit similar purchasing patterns, meaning marketing strategies can be more uniform for these categories.
- **City_Category_C** shows both higher average spending and greater variability, indicating that customers in this category may be more diverse in their purchasing habits, with potential for both high-value and more unpredictable purchases.

This insight suggests that different strategies may be needed for **City_Category_C** to handle both the higher median and the wider spread in customer spending behavior.

Recommendations

1. **Stock Management:**
   - **Observation:** City_Category_A and City_Category_B exhibit similar spending patterns, while City_Category_C shows higher median spending and a greater spread.
   - **Strategies:**
     - For **City_Category_C**, focus on **high-value products** to match the spending habits of customers in this category. Ensure that these products are well-stocked to meet demand, as customers are more likely to make higher-value purchases.
     - For **City_Category_A** and **City_Category_B**, manage inventory with a **balanced supply and demand** strategy. These categories have more consistent spending behavior, so a steady stock of popular products should suffice to cater to their needs.

2. **Product Placement:**
   - **Observation:** The spending behavior in City_Category_C is more variable and higher, while City_Category_A and City_Category_B show similar patterns.
   - **Strategies:**
     - In **City_Category_C**, **strategically place high-value or premium products** in prominent locations, as customers are more likely to make higher-value purchases. Consider offering exclusive or premium items that appeal to this more diverse customer base.
     - In **City_Category_A** and **City_Category_B**, ensure a **diverse range of products** is available to cater to a wider audience with varying preferences. These categories may benefit from offering a mix of budget-friendly and mid-range items.

Overall Goal:

By aligning **stock management** and **product placement** strategies with the purchasing patterns of each city category, Walmart can better meet customer demand, optimize inventory, and boost sales across all locations.

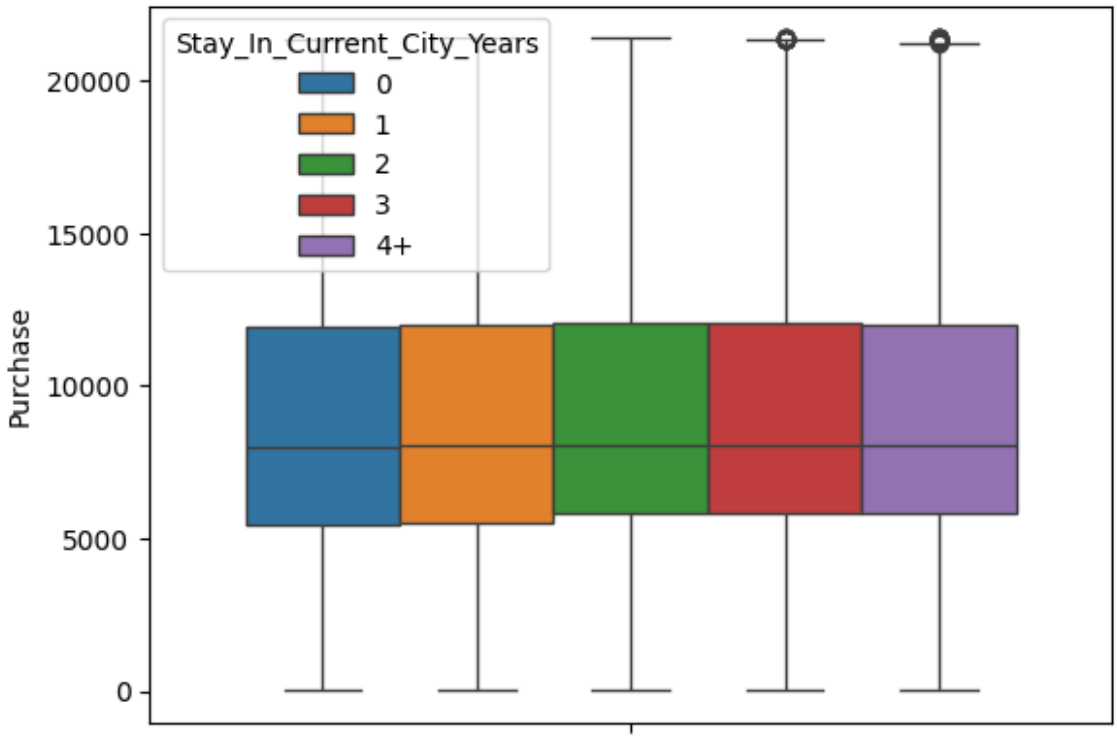
```
df.groupby('Stay_In_Current_City_Years')['Purchase'].describe()
```


```
{"summary":"{\n  \"name\": \"df\",\n  \"rows\": 5,\n  \"fields\": [\n
{\n      \"column\": \"Stay_In_Current_City_Years\",\n
\"properties\": {\n        \"dtype\": \"category\",\n
\"num_unique_values\": 5,\n        \"samples\": [\n          \"1\",\n
\"4+\",\n          \"2\"\n        ],\n        \"semantic_type\":
\"\",\n        \"description\": \"\"\n      }\n    },\n    {\n
\"column\": \"count\",\n      \"properties\": {\n        \"dtype\":
\"number\",\n        \"std\": 47748.96726841325,\n        \"min\":
74036.0,\n        \"max\": 192845.0,\n        \"num_unique_values\":
5,\n        \"samples\": [\n          192845.0,\n          84322.0,\n
101384.0\n        ],\n        \"semantic_type\": \"\",\n
```

\"description\": \"\"\n   }\n   },\n   {\n   \"column\":
\"mean\",\n   \"properties\": {\n   \"dtype\": \"number\",\n
\"std\": 54.5062566654593,\n   \"min\": 9111.331554919228,\n
\"max\": 9258.292028327942,\n   \"num_unique_values\": 5,\n
\"samples\": [\n   9179.275915891001,\n
9208.837895211214,\n   9258.292028327942\n   ],\n
\"semantic_type\": \"\",\n   \"description\": \"\"\n   }\
n   },\n   {\n   \"column\": \"std\",\n   \"properties\": {\n
\"dtype\": \"number\",\n   \"std\": 23.059971643615167,\n
\"min\": 4904.407339759079,\n   \"max\": 4969.2145675383135,\n
\"num_unique_values\": 5,\n   \"samples\": [\n
4939.999789247632,\n   4935.582897990809,\n
4969.2145675383135\n   ],\n   \"semantic_type\": \"\",\n
\"description\": \"\"\n   }\n   },\n   {\n   \"column\":
\"min\",\n   \"properties\": {\n   \"dtype\": \"number\",\n
\"std\": 0.0,\n   \"min\": 12.0,\n   \"max\": 12.0,\n
\"num_unique_values\": 1,\n   \"samples\": [\n   12.0\n
],\n   \"semantic_type\": \"\",\n   \"description\": \"\"\n
}\n   },\n   {\n   \"column\": \"25%\",\n   \"properties\": {\
n   \"dtype\": \"number\",\n   \"std\": 195.30485785048973,\
n   \"min\": 5474.0,\n   \"max\": 5837.0,\n
\"num_unique_values\": 5,\n   \"samples\": [\n   5476.0\n
],\n   \"semantic_type\": \"\",\n   \"description\": \"\"\n
}\n   },\n   {\n   \"column\": \"50%\",\n   \"properties\": {\
n   \"dtype\": \"number\",\n   \"std\": 16.97056274847714,\n
\"min\": 8017.0,\n   \"max\": 8063.5,\n
\"num_unique_values\": 5,\n   \"samples\": [\n   8032.0\n
],\n   \"semantic_type\": \"\",\n   \"description\": \"\"\n
}\n   },\n   {\n   \"column\": \"75%\",\n   \"properties\": {\
n   \"dtype\": \"number\",\n   \"std\": 48.33942490348846,\n
\"min\": 11954.0,\n   \"max\": 12085.0,\n
\"num_unique_values\": 5,\n   \"samples\": [\n   12004.0\n
],\n   \"semantic_type\": \"\",\n   \"description\": \"\"\n
}\n   },\n   {\n   \"column\": \"max\",\n   \"properties\": {\
n   \"dtype\": \"number\",\n   \"std\": 4.527692569068709,\n
\"min\": 21388.0,\n   \"max\": 21399.0,\n
\"num_unique_values\": 4,\n   \"samples\": [\n   21398.0\n
],\n   \"semantic_type\": \"\",\n   \"description\": \"\"\n
}\n   }\n]\n}","type":"dataframe"}

```
sns.boxplot(y="Purchase",hue="Stay_In_Current_City_Years",data=df)
```

```
<Axes: ylabel='Purchase'>
```



Insight

- **Stay in Current City Years and Purchase Amount:**
  - Upon reviewing the **boxplot** for the **Stay_In_Current_City_Years** feature, we observe the following:
    - All categories (0, 1, 2, 3, 4+) have **similar median** purchase amounts, suggesting that the average spending across these groups is relatively consistent.
    - However, the groups with **2, 3, and 4+ years** have a **slightly lower spread (IQR)**, indicating that the variability in purchase amounts is somewhat reduced for these groups compared to the **0 and 1-year** categories.

Interpretation:

- The similar **median purchase amounts** across all groups suggest that the **duration of stay in the current city** does not have a significant effect on the average spending behavior.
- The lower **spread (IQR)** for customers with 2 or more years in the city implies that their spending patterns are more predictable and consistent compared to those who have lived in the city for a shorter period.

This insight indicates that **longer-term residents** may have more stable spending habits, while **newer residents** might display more variation in their purchasing behavior.

Recommendations

1. **Consistency Focus:**
   - **Observation:** Customers who have lived in the city for **2, 3, and 4+ years** show more consistent spending behavior with slightly lower variability in their purchase amounts.
   - **Strategies:**
     - Develop marketing strategies that **maintain and enhance** the purchasing behavior of these long-term residents. Since their spending patterns are more predictable, consider **loyalty programs** or **personalized offers** that incentivize repeat purchases and further solidify their commitment to the brand.

2. **Retention Programs:**
   - **Observation:** Long-term residents tend to have more stable spending habits, which suggests they are more likely to make consistent purchases.
   - **Strategies:**
     - Implement **retention programs** to foster long-term relationships with these customers. Offer **rewards** for frequent purchases, provide **incentives for referrals**, or introduce exclusive membership benefits to keep them engaged and encourage continued shopping.

Overall Goal:

By focusing on **consistency** and implementing **retention programs** for long-term residents, Walmart can increase customer loyalty, enhance purchasing stability, and maintain sustained growth in sales over time.

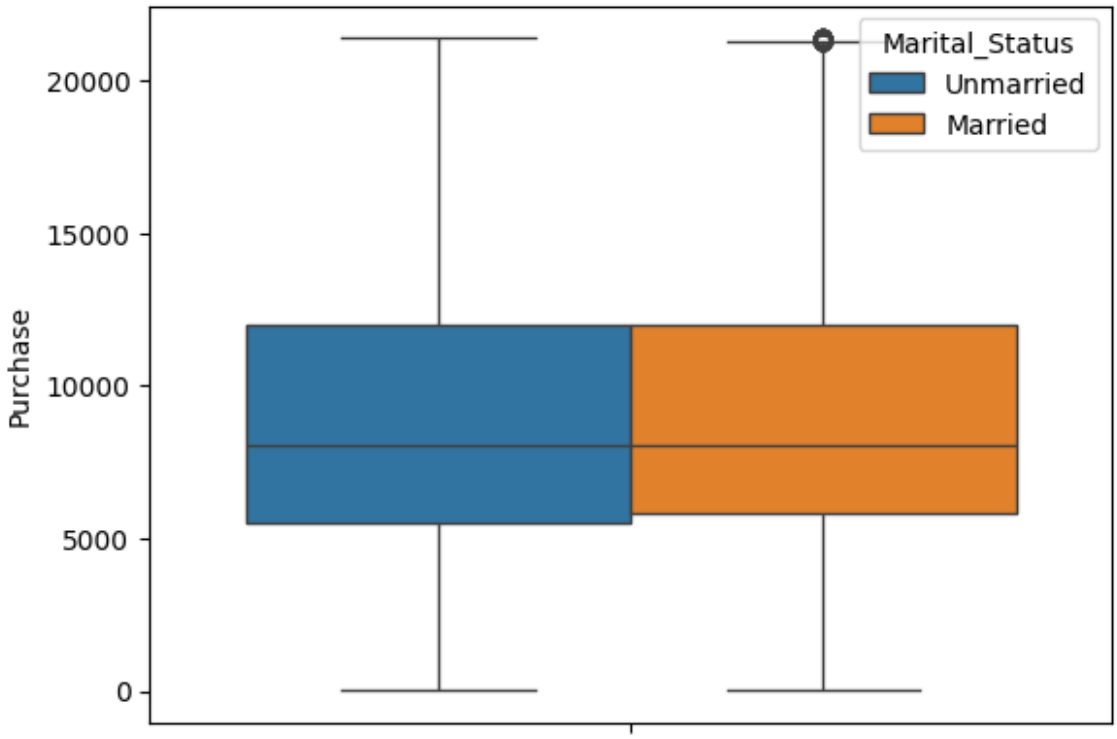
```
df.groupby('Marital_Status')['Purchase'].describe()
```


```
{"summary":"{\n  \"name\": \"df\",\n  \"rows\": 2,\n  \"fields\": [\n{\n      \"column\": \"Marital_Status\",\n      \"properties\": {\n\"dtype\": \"string\",\n        \"num_unique_values\": 2,\n\"samples\": [\n          \"Unmarried\",\n          \"Married\"\n],\n        \"semantic_type\": \"\",\n        \"description\": \"\"\n}\n    },\n    {\n      \"column\": \"count\",\n      \"properties\": {\n        \"dtype\": \"number\",\n        \"std\": 70069.33226811855,\n        \"min\": 224149.0,\n        \"max\": 323242.0,\n        \"num_unique_values\": 2,\n        \"samples\": [\n323242.0,\n          224149.0\n        ],\n        \"semantic_type\": \"\",\n        \"description\": \"\"\n      }\n    },\n    {\n\"column\": \"mean\",\n      \"properties\": {\n        \"dtype\": \"number\",\n        \"std\": 10.28258620864547,\n        \"min\": 9187.040076020861,\n        \"max\": 9201.581848893398,\n\"num_unique_values\": 2,\n        \"samples\": [\n9201.581848893398,\n          9187.040076020861\n        ],\n\"semantic_type\": \"\",\n        \"description\": \"\"\n      }\n    },\n    {\n      \"column\": \"std\",\n      \"properties\": {\n\"dtype\": \"number\",\n        \"std\": 16.349840096872573,\n\"min\": 4925.2052318513715,\n        \"max\": 4948.327397459,\n
```

\"num_unique_values\": 2,\n   \"samples\": [\n
4948.327397459,\n   4925.2052318513715\n   ],\n
\"semantic_type\": \"\",\n   \"description\": \"\"\n   }\
n   },\n   {\n   \"column\": \"min\",\n   \"properties\": {\n
\"dtype\": \"number\",\n   \"std\": 0.0,\n   \"min\": 12.0,\
n   \"max\": 12.0,\n   \"num_unique_values\": 1,\n
\"samples\": [\n   12.0\n   ],\n   \"semantic_type\":
\"\",\n   \"description\": \"\"\n   }\n   },\n   {\n
\"column\": \"25%\",\n   \"properties\": {\n   \"dtype\":
\"number\",\n   \"std\": 249.60869375885127,\n   \"min\":
5480.0,\n   \"max\": 5833.0,\n   \"num_unique_values\": 2,\n
\"samples\": [\n   5480.0\n   ],\n
\"semantic_type\": \"\",\n   \"description\": \"\"\n   }\
n   },\n   {\n   \"column\": \"50%\",\n   \"properties\": {\n
\"dtype\": \"number\",\n   \"std\": 4.949747468305833,\n
\"min\": 8035.0,\n   \"max\": 8042.0,\n
\"num_unique_values\": 2,\n   \"samples\": [\n   8035.0\n
],\n   \"semantic_type\": \"\",\n   \"description\": \"\"\n
}\n   },\n   {\n   \"column\": \"75%\",\n   \"properties\": {\
n   \"dtype\": \"number\",\n   \"std\": 15.556349186104045,\
n   \"min\": 12006.0,\n   \"max\": 12028.0,\n
\"num_unique_values\": 2,\n   \"samples\": [\n   12028.0\n
],\n   \"semantic_type\": \"\",\n   \"description\": \"\"\n
}\n   },\n   {\n   \"column\": \"max\",\n   \"properties\": {\
n   \"dtype\": \"number\",\n   \"std\": 0.7071067811865476,\
n   \"min\": 21398.0,\n   \"max\": 21399.0,\n
\"num_unique_values\": 2,\n   \"samples\": [\n   21399.0\n
],\n   \"semantic_type\": \"\",\n   \"description\": \"\"\n
}\n   }\n]\n}","type":"dataframe"}

```
sns.boxplot(y="Purchase",hue="Marital_Status",data=df)
```

```
<Axes: ylabel='Purchase'>
```



Insight

- **Marital Status and Purchase Amount:**
  - Upon reviewing the **boxplot** for **Marital_Status**, we observe the following:
    - Both **married** and **unmarried** customers have **similar median** purchase amounts, suggesting that the average spending behavior is consistent across these two groups.
    - However, **married customers** have a **slightly lower spread**, indicating that their spending behavior is more consistent and less variable compared to **unmarried customers**.

Interpretation:

- The **similar median purchase amounts** indicate that both **married** and **unmarried** customers tend to spend similar amounts on average.
- The **lower spread** for married customers suggests that their purchasing behavior is more predictable, while unmarried customers exhibit a slightly higher variation in spending.

This insight highlights that while the **average purchase** is similar across marital status groups, married customers might be more consistent in their spending, which can help in tailoring specific marketing strategies.

Recommendations

1. **Segmented Campaigns:**
   - **Observation:** Both **married** and **unmarried** customers exhibit similar median purchase amounts, but married customers show more consistent spending behavior.

- **Strategies:**
  - Design **segmented marketing campaigns** that target the unique preferences and spending behaviors of both groups.
  - For **married customers**, focus on **family-oriented promotions** or offers that appeal to household needs, such as bundled deals or discounts on family products.
  - For **unmarried customers**, create promotions that cater to individual preferences, such as single-product discounts or offers that align with personal lifestyle choices.

2. **Incentive Programs:**

- **Observation:** While the average spending is similar, married customers tend to have a more **consistent spending pattern**.
- **Strategies:**
  - For **married customers**, implement **loyalty programs** or rewards that encourage **consistent spending**, such as discounts for repeat purchases or special rewards for long-term customers.
  - For **unmarried customers**, provide **flexible incentives** that appeal to a broader range of spending behaviors. This could include offers that are more spontaneous or based on specific product categories (e.g., fashion, electronics).

Overall Goal:

By creating **segmented campaigns** and tailored **incentive programs**, Walmart can cater to the distinct needs of both married and unmarried customers, maximizing engagement and driving higher sales in each segment.

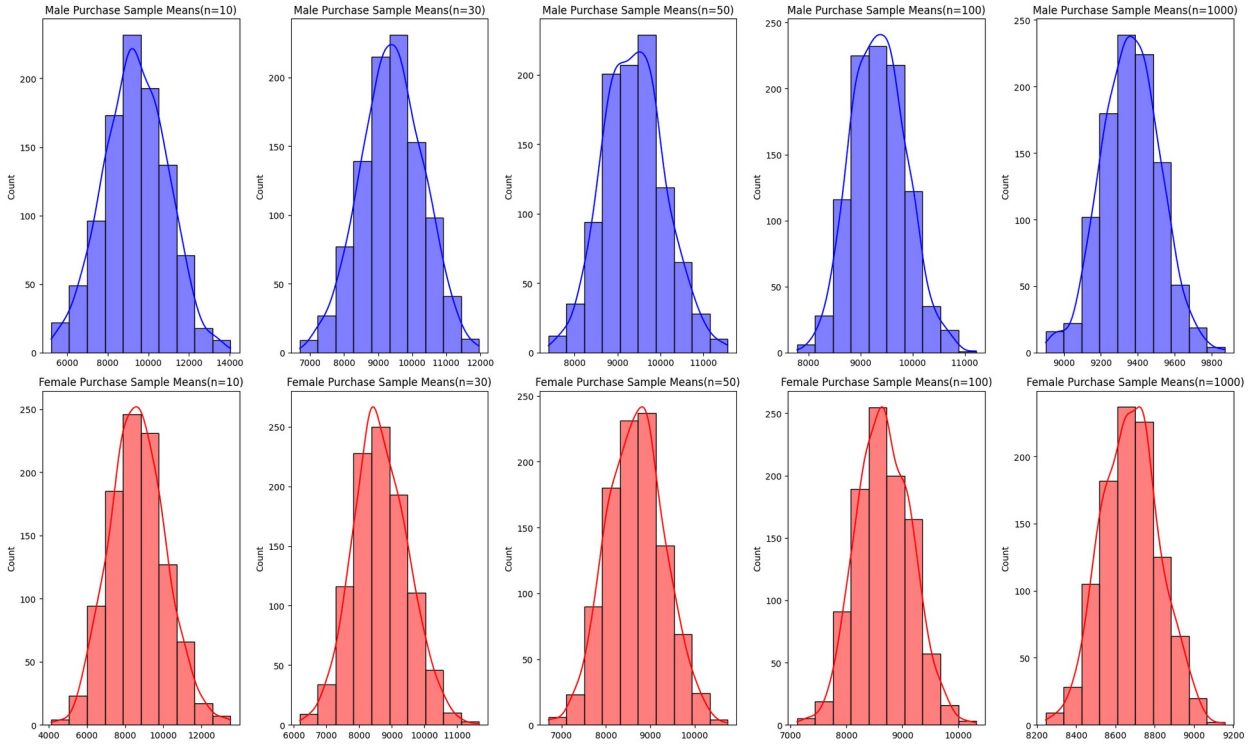
```
#Top 10 product categories by average purchase amount per transcation
df.groupby("Product_Category")
['Purchase'].mean().reset_index().sort_values(by="Purchase",ascending=
False).head(10)
```


```
{"summary":"{\n  \"name\": \"df\",\n  \"rows\": 10,\n  \"fields\": [\n
{\n      \"column\": \"Product_Category\",\n      \"properties\": {\n
\"dtype\": \"string\",\n        \"num_unique_values\": 10,\n
\"samples\": [\n          \"2\",\n          \"7\",\n          \"9\"\n
],\n        \"semantic_type\": \"\",\n        \"description\": \"\"\n
}\n    },\n    {\n      \"column\": \"Purchase\",\n
\"properties\": {\n        \"dtype\": \"number\",\n        \"std\":
2101.8479678431527,\n        \"min\": 10170.759515570935,\n
\"max\": 16626.385964912282,\n        \"num_unique_values\": 10,\n
\"samples\": [\n          11251.93538384177,\n
16365.689599570009,\n          13852.325373134328\n        ],\n
\"semantic_type\": \"\",\n        \"description\": \"\"\n      }\
n    }\n  ]\n}","type":"dataframe"}
```


```
df_male=df[df["Gender"]=="M"]
df_female=df[df["Gender"]=="F"]
```

```
#Now let us find if men are spending more amount per transcation by
using two sampe t-test
import scipy.stats as stats

def confidence_interval(data, confidence=0.95):
    n = len(data)
    mean = np.mean(data)
    stderr = stats.sem(data)
    margin_of_error = stderr * stats.t.ppf((1 + confidence) / 2., n-1)
    return mean - margin_of_error, mean + margin_of_error

sample_sizes=[10,30,50,100,1000]
confidence_levels = [0.90, 0.95, 0.99]
fig, axes = plt.subplots(nrows=2, ncols=5, figsize=(20, 12))
axes = axes.flatten()
results = {
    "Sample Size": [],
    "Group": [],
    "Expected Population Mean": [],
    "Standard Error":[]}

for idx, sample_size in enumerate(sample_sizes):
    male_sample_means = []
    female_sample_means = []

    # Generate sample means for male and female customers
    for _ in range(1000):
        male_sample = df_male['Purchase'].sample(sample_size,
replace=True)
        female_sample = df_female['Purchase'].sample(sample_size,
replace=True)
        male_sample_means.append(male_sample.mean())
        female_sample_means.append(female_sample.mean())
    results["Sample Size"].append(sample_size)
    results["Group"].append("Male")
    results["Expected Population
Mean"].append(np.array(male_sample_means).mean())
    results["Standard
Error"].append(np.array(male_sample_means).std())

    results["Sample Size"].append(sample_size)
    results["Group"].append("Female")
    results["Expected Population
Mean"].append(np.array(female_sample_means).mean())
    results["Standard
Error"].append(np.array(female_sample_means).std())
    # Plot histograms for male sample means
    sns.histplot(male_sample_means, bins=10, ax=axes[idx], kde=True,
color='blue')
    axes[idx].set_title(f"Male Purchase Sample
```

```
Means(n={sample_size})")

    # Plot histograms for female sample means
    sns.histplot(female_sample_means, bins=10, ax=axes[idx+5],
kde=True, color='red')
    axes[idx+5].set_title(f"Female Purchase Sample
Means(n={sample_size})")

    # Calculate and display confidence intervals

# Adjust layout
plt.tight_layout()
plt.show()
```



Insight

- **Distribution and Sample Size:**
  - By analyzing the **plots** for male and female customers, it is evident that as the **sample size increases**, the **distribution of purchase amounts** gradually becomes **closer to normal** for both groups.

Interpretation:

- This phenomenon aligns with the **Central Limit Theorem**, which states that as the sample size increases, the sampling distribution of the mean tends to approximate a **normal distribution**, regardless of the population's original distribution.

- The **increasing normality** observed in the plots indicates that larger sample sizes provide a more accurate representation of the overall spending patterns of male and female customers.

This insight highlights the significance of using sufficiently large sample sizes in statistical analysis to draw reliable and generalizable conclusions.

```
d = pd.DataFrame(results)
d
```


```
{"summary":"{\n  \"name\": \"d\",\n  \"rows\": 10,\n  \"fields\": [\n
{\n      \"column\": \"Sample Size\",\n      \"properties\": {\n
\"dtype\": \"number\",\n        \"std\": 402,\n        \"min\": 10,\n
\"max\": 1000,\n        \"num_unique_values\": 5,\n
\"samples\": [\n          30,\n          1000,\n          50\
n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    },\n    {\n      \"column\":
\"Group\",\n      \"properties\": {\n        \"dtype\": \"category\",\
n        \"num_unique_values\": 2,\n        \"samples\": [\n
\"Female\",\n          \"Male\"\n        ],\n
\"semantic_type\": \"\",\n        \"description\": \"\"\n      }\
n    },\n    {\n      \"column\": \"Expected Population Mean\",\n
\"properties\": {\n        \"dtype\": \"number\",\n        \"std\":
369.93344046447214,\n        \"min\": 8659.508566666667,\n
\"max\": 9401.734799999998,\n        \"num_unique_values\": 10,\n
\"samples\": [\n          9361.327022,\n          8663.4795\
n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    },\n    {\n      \"column\":
\"Standard Error\",\n      \"properties\": {\n        \"dtype\":
\"number\",\n        \"std\": 475.8502190233835,\n        \"min\":
146.22093831506564,\n        \"max\": 1559.249962004165,\n
\"num_unique_values\": 10,\n        \"samples\": [\n
156.7100692207604,\n          1454.0867058706472\n        ],\n
\"semantic_type\": \"\",\n        \"description\": \"\"\n      }\
n    }\n  ]\n}","type":"dataframe","variable_name":"d"}
```


Insight

- **Relationship Between Sample Size and Standard Error:**
  - Observing the data, it is clear that as the **sample size increases**, the **standard error** decreases.
- **Application of Central Limit Theorem (CLT):**
  - Using the **Central Limit Theorem (CLT)**:
    - The **expected population mean** for male customers is estimated to be approximately **9370**.
    - The **expected population mean** for female customers is estimated to be around **8668**.

Interpretation:

- The **decrease in standard error** with larger sample sizes indicates higher precision in estimating the population mean.
- The calculated **expected means** for male and female customers provide a reliable estimate of their respective average spending behaviors based on the observed data.

This insight demonstrates the importance of increasing the sample size to reduce variability and obtain more accurate population estimates.

```
sample_sizes = [10, 30, 50, 100, 500,1000]
confidence_level = 0.9

# Function to calculate confidence interval
def confidence_interval(data, confidence=0.99):
    n = len(data)
    mean = np.mean(data)
    stderr = stats.sem(data)
    margin_of_error = stderr * stats.t.ppf((1 + confidence) / 2., n-1)
    return mean, mean - margin_of_error, mean + margin_of_error

# Prepare lists to store results
results = {
    "Sample Size": [],
    "Group": [],
    "Mean": [],
    "CI Lower Bound": [],
    "CI Upper Bound": [],
    "CI Length":[],
    "t-statstic":[],
    "p-value":[],
    "interval-overlap?":[]
}

for sample_size in sample_sizes:
    male_sample_means = []
    female_sample_means = []

    # Generate sample means for male and female customers
    for _ in range(1000):
        male_sample = df_male['Purchase'].sample(sample_size,
replace=True)
        female_sample = df_female['Purchase'].sample(sample_size,
replace=True)
        male_sample_means.append(male_sample.mean())
        female_sample_means.append(female_sample.mean())


    # Calculate confidence intervals for male and female
    male_mean, male_ci_lower, male_ci_upper =
confidence_interval(male_sample_means, confidence=confidence_level)
```

```
    female_mean, female_ci_lower, female_ci_upper =
confidence_interval(female_sample_means, confidence=confidence_level)
    t_stat, p_value =
stats.ttest_ind(male_sample_means,female_sample_means,
equal_var=False)
    interval_overlap=["yes" if
pd.Interval(male_ci_lower,male_ci_upper).overlaps(pd.Interval(female_c
i_lower,female_ci_upper)) else "no"][0]

    # Append results to the list
    results["Sample Size"].append(sample_size)
    results["Group"].append("Male")
    results["Mean"].append(male_mean)
    results["CI Lower Bound"].append(male_ci_lower)
    results["CI Upper Bound"].append(male_ci_upper)
    results["CI Length"].append(male_ci_upper-male_ci_lower)
    results['t-statstic'].append(t_stat)
    results['p-value'].append(p_value)
    results['interval-overlap?'].append(interval_overlap)


    results["Sample Size"].append(sample_size)
    results["Group"].append("Female")
    results["Mean"].append(female_mean)
    results["CI Lower Bound"].append(female_ci_lower)
    results["CI Upper Bound"].append(female_ci_upper)
    results["CI Length"].append(female_ci_upper-female_ci_lower)
    results['t-statstic'].append(t_stat)
    results['p-value'].append(p_value)
    results['interval-overlap?'].append(interval_overlap)


# Convert results to a DataFrame
ci_df = pd.DataFrame(results)

# Display the DataFrame
ci_df
```


```
{"summary":"{\n  \"name\": \"ci_df\",\n  \"rows\": 12,\n  \"fields\":
[\n    {\n      \"column\": \"Sample Size\",\n      \"properties\": {\
n        \"dtype\": \"number\",\n        \"std\": 378,\n
\"min\": 10,\n        \"max\": 1000,\n        \"num_unique_values\":
6,\n        \"samples\": [\n          10,\n          30,\n
1000\n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    },\n    {\n      \"column\":
\"Group\",\n      \"properties\": {\n        \"dtype\": \"category\",\
n        \"num_unique_values\": 2,\n        \"samples\": [\n
\"Female\",\n          \"Male\"\n        ],\n
\"semantic_type\": \"\",\n        \"description\": \"\"\n      }\
n    },\n    {\n      \"column\": \"Mean\",\n      \"properties\": {\n
```

\"dtype\": \"number\",\n   \"std\": 369.1439818962892,\n
\"min\": 8638.274899999999,\n   \"max\": 9408.799,\n
\"num_unique_values\": 12,\n   \"samples\": [\n
9364.471332,\n   8669.044131999999\n   ],\n
\"semantic_type\": \"\",\n   \"description\": \"\"\n   }\
n   },\n   {\n   \"column\": \"CI Lower Bound\",\n
\"properties\": {\n   \"dtype\": \"number\",\n   \"std\":
368.8754032856207,\n   \"min\": 8562.559781722837,\n
\"max\": 9372.5917092319,\n   \"num_unique_values\": 12,\n
\"samples\": [\n   9356.276993497033,\n
8657.91518741801\n   ],\n   \"semantic_type\": \"\",\n
\"description\": \"\"\n   }\n   },\n   {\n   \"column\": \"CI
Upper Bound\",\n   \"properties\": {\n   \"dtype\":
\"number\",\n   \"std\": 371.0216911090088,\n   \"min\":
8674.439647066461,\n   \"max\": 9463.238369747869,\n
\"num_unique_values\": 12,\n   \"samples\": [\n
9372.665670502965,\n   8680.173076581988\n   ],\n
\"semantic_type\": \"\",\n   \"description\": \"\"\n   }\
n   },\n   {\n   \"column\": \"CI Length\",\n
\"properties\": {\n   \"dtype\": \"number\",\n   \"std\":
48.818022942805705,\n   \"min\": 15.669588200631551,\n
\"max\": 157.38713949573867,\n   \"num_unique_values\": 12,\n
\"samples\": [\n   16.38867700593255,\n
22.25788916397869\n   ],\n   \"semantic_type\": \"\",\n
\"description\": \"\"\n   }\n   },\n   {\n   \"column\": \"t-
statstic\",\n   \"properties\": {\n   \"dtype\": \"number\",\n
\"std\": 33.54876156427142,\n   \"min\": 11.25093411757729,\n
\"max\": 100.05684795102604,\n   \"num_unique_values\": 6,\n
\"samples\": [\n   11.25093411757729,\n
16.560277889994097\n   ],\n   \"semantic_type\": \"\",\n
\"description\": \"\"\n   }\n   },\n   {\n   \"column\": \"p-
value\",\n   \"properties\": {\n   \"dtype\": \"number\",\n
\"std\": 6.316376392196787e-29,\n   \"min\": 0.0,\n
\"max\": 1.6227064762431494e-28,\n   \"num_unique_values\": 5,\n
\"samples\": [\n   8.325857974101742e-58,\n   0.0\n
],\n   \"semantic_type\": \"\",\n   \"description\": \"\"\n
}\n   },\n   {\n   \"column\": \"interval-overlap?\",\n
\"properties\": {\n   \"dtype\": \"category\",\n
\"num_unique_values\": 1,\n   \"samples\": [\n   \"no\"\n
],\n   \"semantic_type\": \"\",\n   \"description\": \"\"\n
}\n   }\n]\n}","type":"dataframe","variable_name":"ci_df"}

```
sample_sizes = [10, 30, 50, 100, 500,1000]
confidence_level = 0.95

# Function to calculate confidence interval
def confidence_interval(data, confidence=0.99):
    n = len(data)
    mean = np.mean(data)
    stderr = stats.sem(data)
```

```
    margin_of_error = stderr * stats.t.ppf((1 + confidence) / 2., n-1)
    return mean, mean - margin_of_error, mean + margin_of_error

# Prepare lists to store results
results = {
    "Sample Size": [],
    "Group": [],
    "Mean": [],
    "CI Lower Bound": [],
    "CI Upper Bound": [],
    "CI Length":[],
    "t-statstic":[],
    "p-value":[],
    "interval-overlap?":[]
}

for sample_size in sample_sizes:
    male_sample_means = []
    female_sample_means = []

    # Generate sample means for male and female customers
    for _ in range(1000):
        male_sample = df_male['Purchase'].sample(sample_size,
replace=True)
        female_sample = df_female['Purchase'].sample(sample_size,
replace=True)
        male_sample_means.append(male_sample.mean())
        female_sample_means.append(female_sample.mean())


    # Calculate confidence intervals for male and female
    male_mean, male_ci_lower, male_ci_upper =
confidence_interval(male_sample_means, confidence=confidence_level)
    female_mean, female_ci_lower, female_ci_upper =
confidence_interval(female_sample_means, confidence=confidence_level)
    t_stat, p_value =
stats.ttest_ind(male_sample_means,female_sample_means,
equal_var=False)
    interval_overlap=["yes" if
pd.Interval(male_ci_lower,male_ci_upper).overlaps(pd.Interval(female_c
i_lower,female_ci_upper)) else "no"][0]

    # Append results to the list
    results["Sample Size"].append(sample_size)
    results["Group"].append("Male")
    results["Mean"].append(male_mean)
    results["CI Lower Bound"].append(male_ci_lower)
    results["CI Upper Bound"].append(male_ci_upper)
    results["CI Length"].append(male_ci_upper-male_ci_lower)
    results['t-statstic'].append(t_stat)
```

```
    results['p-value'].append(p_value)
    results['interval-overlap?'].append(interval_overlap)


    results["Sample Size"].append(sample_size)
    results["Group"].append("Female")
    results["Mean"].append(female_mean)
    results["CI Lower Bound"].append(female_ci_lower)
    results["CI Upper Bound"].append(female_ci_upper)
    results["CI Length"].append(female_ci_upper-female_ci_lower)
    results['t-statstic'].append(t_stat)
    results['p-value'].append(p_value)
    results['interval-overlap?'].append(interval_overlap)


# Convert results to a DataFrame
ci_df = pd.DataFrame(results)

# Display the DataFrame
ci_df
```


```
{"summary":"{\n  \"name\": \"ci_df\",\n  \"rows\": 12,\n  \"fields\":
[\n    {\n      \"column\": \"Sample Size\",\n      \"properties\": {\
n        \"dtype\": \"number\",\n        \"std\": 378,\n
\"min\": 10,\n        \"max\": 1000,\n        \"num_unique_values\":
6,\n        \"samples\": [\n          10,\n          30,\n
1000\n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    },\n    {\n      \"column\":
\"Group\",\n      \"properties\": {\n        \"dtype\": \"category\",\
n        \"num_unique_values\": 2,\n        \"samples\": [\n
\"Female\",\n          \"Male\"\n        ],\n
\"semantic_type\": \"\",\n        \"description\": \"\"\n      }\
n    },\n    {\n      \"column\": \"Mean\",\n      \"properties\": {\n
\"dtype\": \"number\",\n        \"std\": 352.3885348244693,\n
\"min\": 8647.289466666665,\n        \"max\": 9371.196761999998,\n
\"num_unique_values\": 12,\n        \"samples\": [\n
9365.057114,\n          8664.339666\n        ],\n
\"semantic_type\": \"\",\n        \"description\": \"\"\n      }\
n    },\n    {\n      \"column\": \"CI Lower Bound\",\n
\"properties\": {\n        \"dtype\": \"number\",\n        \"std\":
353.30580992297223,\n        \"min\": 8592.804667284152,\n
\"max\": 9357.404266575104,\n        \"num_unique_values\": 12,\n
\"samples\": [\n          9354.932471341614,\n
8651.56743637288\n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    },\n    {\n      \"column\": \"CI
Upper Bound\",\n      \"properties\": {\n        \"dtype\":
\"number\",\n        \"std\": 354.11751105573865,\n        \"min\":
8674.393323996112,\n        \"max\": 9394.594889783206,\n
\"num_unique_values\": 12,\n        \"samples\": [\n
9375.181756658385,\n          8677.11189562712\n        ],\n
```

\"semantic_type\": \"\",\n   \"description\": \"\"\n   }\
n   },\n   {\n   \"column\": \"CI Length\",\n
\"properties\": {\n   \"dtype\": \"number\",\n   \"std\":
61.136699455665074,\n   \"min\": 18.03956999222646,\n
\"max\": 196.11274046691688,\n   \"num_unique_values\": 12,\n
\"samples\": [\n   20.24928531677142,\n
25.544459254240792\n   ],\n   \"semantic_type\": \"\",\n
\"description\": \"\"\n   }\n   },\n   {\n   \"column\": \"t-
statstic\",\n   \"properties\": {\n   \"dtype\": \"number\",\n
\"std\": 35.079418716332896,\n   \"min\": 8.389009070661423,\n
\"max\": 101.2574139902751,\n   \"num_unique_values\": 6,\n
\"samples\": [\n   8.389009070661423,\n
16.932957490246157\n   ],\n   \"semantic_type\": \"\",\n
\"description\": \"\"\n   }\n   },\n   {\n   \"column\": \"p-
value\",\n   \"properties\": {\n   \"dtype\": \"number\",\n
\"std\": 3.5618406357804964e-17,\n   \"min\": 0.0,\n
\"max\": 9.150534274948189e-17,\n   \"num_unique_values\": 5,\n
\"samples\": [\n   3.4533793204023223e-60,\n   0.0\n
],\n   \"semantic_type\": \"\",\n   \"description\": \"\"\n
}\n   },\n   {\n   \"column\": \"interval-overlap?\",\n
\"properties\": {\n   \"dtype\": \"category\",\n
\"num_unique_values\": 1,\n   \"samples\": [\n   \"no\"\n
],\n   \"semantic_type\": \"\",\n   \"description\": \"\"\n
}\n   }\n]\n}","type":"dataframe","variable_name":"ci_df"}

```
sample_sizes = [10, 30, 50, 100, 500,1000]
confidence_level = 0.99

# Function to calculate confidence interval
def confidence_interval(data, confidence=0.99):
    n = len(data)
    mean = np.mean(data)
    stderr = stats.sem(data)
    margin_of_error = stderr * stats.t.ppf((1 + confidence) / 2., n-1)
    return mean, mean - margin_of_error, mean + margin_of_error

# Prepare lists to store results
results = {
    "Sample Size": [],
    "Group": [],
    "Mean": [],
    "CI Lower Bound": [],
    "CI Upper Bound": [],
    "CI Length":[],
    "t-statstic":[],
    "p-value":[],
    "interval-overlap?":[]
}

for sample_size in sample_sizes:
```

```
    male_sample_means = []
    female_sample_means = []

    # Generate sample means for male and female customers
    for _ in range(1000):
        male_sample = df_male['Purchase'].sample(sample_size,
replace=True)
        female_sample = df_female['Purchase'].sample(sample_size,
replace=True)
        male_sample_means.append(male_sample.mean())
        female_sample_means.append(female_sample.mean())


    # Calculate confidence intervals for male and female
    male_mean, male_ci_lower, male_ci_upper =
confidence_interval(male_sample_means, confidence=confidence_level)
    female_mean, female_ci_lower, female_ci_upper =
confidence_interval(female_sample_means, confidence=confidence_level)
    t_stat, p_value =
stats.ttest_ind(male_sample_means,female_sample_means,
equal_var=False)
    interval_overlap=["yes" if
pd.Interval(male_ci_lower,male_ci_upper).overlaps(pd.Interval(female_c
i_lower,female_ci_upper)) else "no"][0]

    # Append results to the list
    results["Sample Size"].append(sample_size)
    results["Group"].append("Male")
    results["Mean"].append(male_mean)
    results["CI Lower Bound"].append(male_ci_lower)
    results["CI Upper Bound"].append(male_ci_upper)
    results["CI Length"].append(male_ci_upper-male_ci_lower)
    results['t-statstic'].append(t_stat)
    results['p-value'].append(p_value)
    results['interval-overlap?'].append(interval_overlap)


    results["Sample Size"].append(sample_size)
    results["Group"].append("Female")
    results["Mean"].append(female_mean)
    results["CI Lower Bound"].append(female_ci_lower)
    results["CI Upper Bound"].append(female_ci_upper)
    results["CI Length"].append(female_ci_upper-female_ci_lower)
    results['t-statstic'].append(t_stat)
    results['p-value'].append(p_value)
    results['interval-overlap?'].append(interval_overlap)


# Convert results to a DataFrame
ci_df = pd.DataFrame(results)
```

```
# Display the DataFrame
ci_df
```


```
{"summary":"{\n  \"name\": \"ci_df\",\n  \"rows\": 12,\n  \"fields\":
[\n    {\n      \"column\": \"Sample Size\",\n      \"properties\": {\
n        \"dtype\": \"number\",\n        \"std\": 378,\n
\"min\": 10,\n        \"max\": 1000,\n        \"num_unique_values\":
6,\n        \"samples\": [\n          10,\n          30,\n
1000\n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    },\n    {\n      \"column\":
\"Group\",\n      \"properties\": {\n        \"dtype\": \"category\",\
n        \"num_unique_values\": 2,\n        \"samples\": [\n
\"Female\",\n          \"Male\"\n        ],\n
\"semantic_type\": \"\",\n        \"description\": \"\"\n      }\
n    },\n    {\n      \"column\": \"Mean\",\n      \"properties\": {\n
\"dtype\": \"number\",\n        \"std\": 366.3469810643783,\n
\"min\": 8644.669199999998,\n        \"max\": 9406.831300000002,\n
\"num_unique_values\": 12,\n        \"samples\": [\n
9377.573634,\n          8676.5412\n        ],\n
\"semantic_type\": \"\",\n        \"description\": \"\"\n      }\
n    },\n    {\n      \"column\": \"CI Lower Bound\",\n
\"properties\": {\n        \"dtype\": \"number\",\n        \"std\":
366.59573022519976,\n        \"min\": 8521.906276762289,\n
\"max\": 9365.082624449606,\n        \"num_unique_values\": 12,\n
\"samples\": [\n          9365.082624449606,\n
8659.212637165158\n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    },\n    {\n      \"column\": \"CI
Upper Bound\",\n      \"properties\": {\n        \"dtype\":
\"number\",\n        \"std\": 370.67134732262355,\n        \"min\":
8684.520630269402,\n        \"max\": 9542.006436818709,\n
\"num_unique_values\": 12,\n        \"samples\": [\n
9390.064643550395,\n          8693.869762834842\n        ],\n
\"semantic_type\": \"\",\n        \"description\": \"\"\n      }\
n    },\n    {\n      \"column\": \"CI Length\",\n
\"properties\": {\n        \"dtype\": \"number\",\n        \"std\":
82.090877390226,\n        \"min\": 23.17544053880556,\n
\"max\": 270.3502736374139,\n        \"num_unique_values\": 12,\n
\"samples\": [\n          24.98201910078933,\n
34.65712566968432\n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    },\n    {\n      \"column\": \"t-
statstic\",\n      \"properties\": {\n        \"dtype\": \"number\",\n
\"std\": 36.069688870801826,\n        \"min\": 10.771897833254352,\n
\"max\": 106.73118830887499,\n        \"num_unique_values\": 6,\n
\"samples\": [\n          10.771897833254352,\n
15.683545696855456\n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    },\n    {\n      \"column\": \"p-
value\",\n      \"properties\": {\n        \"dtype\": \"number\",\n
\"std\": 9.612969841982134e-27,\n        \"min\": 0.0,\n
\"max\": 2.4696166678390857e-26,\n        \"num_unique_values\": 5,\n
```

\"samples\": [\n   2.3833592553010907e-52,\n   0.0\n
],\n   \"semantic_type\": \"\",\n   \"description\": \"\"\n
}\n   },\n   {\n   \"column\": \"interval-overlap?\",\n
\"properties\": {\n   \"dtype\": \"category\",\n
\"num_unique_values\": 1,\n   \"samples\": [\n   \"no\"\n
],\n   \"semantic_type\": \"\",\n   \"description\": \"\"\n
}\n   }\n]\n}","type":"dataframe","variable_name":"ci_df"}

Insight

Relationship Between Confidence Level and Confidence Interval Length:

- Observing the three dataframes for **90%**, **95%**, and **99%** confidence levels:
  - As the **confidence level increases**, the **confidence interval length also increases**.
  - This happens because higher confidence levels require capturing a broader range to ensure the population mean is within the interval.

Relationship Between Sample Size and Confidence Interval Length:

- As the **sample size increases**, the **confidence interval length decreases**.
  - This indicates that with larger samples, we achieve greater **precision** in estimating the population mean due to reduced variability.

Interpretation:

- The **increase in interval length with higher confidence levels** reflects the trade-off between confidence and precision.
- The **decrease in interval length with larger sample sizes** demonstrates how larger datasets lead to more precise and reliable population mean calculations.

This insight emphasizes the importance of balancing confidence levels and sample sizes to achieve accurate and meaningful statistical inferences.

Insight

Population Mean Estimates for Male Customers:

- Based on a **sample size of 1000**, the **confidence intervals** for the population mean are as follows:
  - **90% Confidence Level (CL):** 9368 - 9384
  - **95% Confidence Level (CL):** 9358 - 9378
  - **99% Confidence Level (CL):** 9359 - 9386

Population Mean Estimates for Female Customers:

- Based on the same **sample size of 1000**, the **confidence intervals** for the population mean are as follows:
  - **90% Confidence Level (CL):** 8662 - 8677

- **95% Confidence Level (CL):** 8661 - 8680
- **99% Confidence Level (CL):** 8656 - 8680

Interpretation:

- The **population mean estimates** for both male and female customers remain consistent across different confidence levels, with a slight increase in range as confidence level rises.
- **Tighter intervals** at lower confidence levels (e.g., 90%) reflect more precision but less certainty, while **wider intervals** at higher confidence levels (e.g., 99%) ensure more certainty at the cost of precision.

This highlights the trade-off between precision and confidence in statistical inference.

Insight

Confidence Intervals and Median Comparison:

- Observing the **dataframes across different confidence levels** and sample sizes:
  - The **confidence intervals for male and female customers do not overlap**.
  - The **median purchase amount** for male customers is slightly higher compared to female customers.

Conclusion:

- Based on the confidence intervals and the median values:
  - **Male customers tend to spend more per purchase** than female customers.

Interpretation:

- The lack of overlap in confidence intervals strengthens the conclusion that male customers generally have higher spending patterns.
- This insight can guide tailored strategies to further engage male customers while encouraging higher spending among female customers.

Recommendations

Segmented Campaigns:

- Design **gender-specific marketing campaigns** to align with the spending habits observed:
  - For **male customers**, emphasize **high-value promotions** to further leverage their higher spending tendencies.
  - For **female customers**, introduce **incentives** like discounts or bundle deals to encourage higher purchase amounts.

Incentive Programs:

- Implement targeted **incentive programs** to address the unique spending behaviors of each gender:
  - For **male customers**, offer **cashback or rewards** for high-value purchases to maintain and increase their spending.

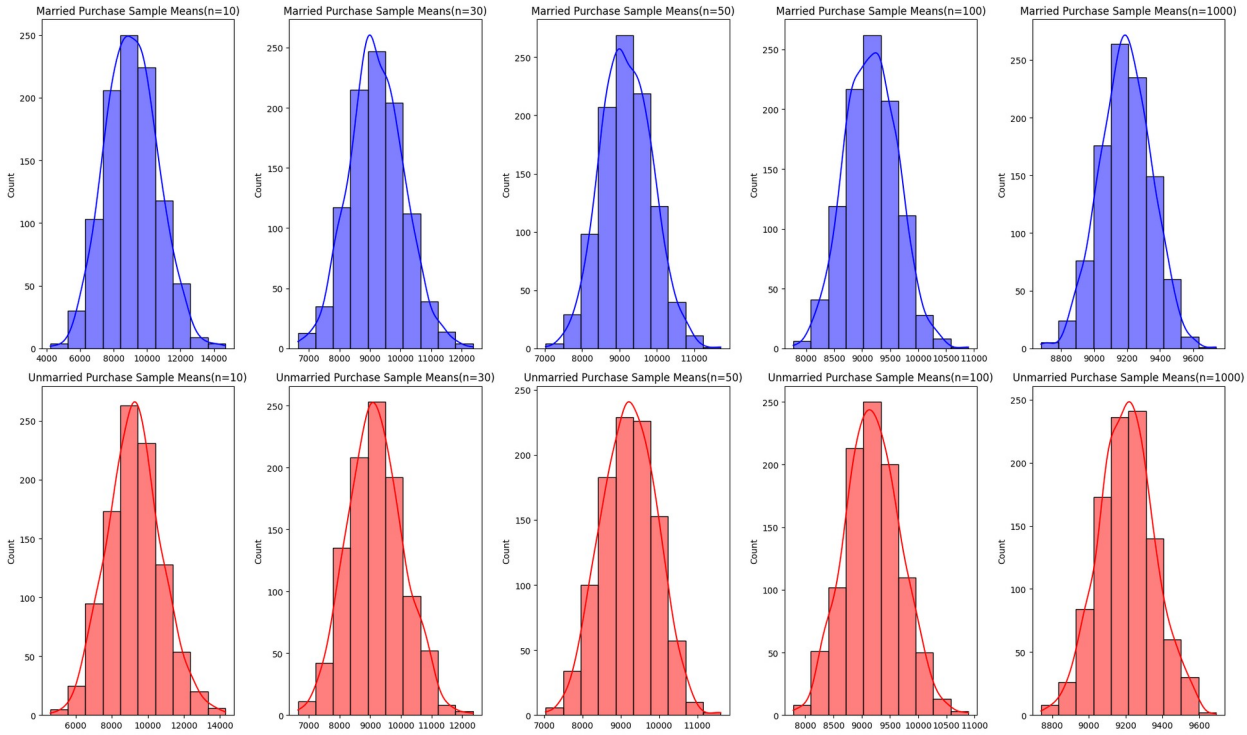
- For **female customers**, consider **tiered rewards** or exclusive offers to boost engagement and spending.

Objective:

- These strategies aim to **maximize revenue** by capitalizing on the distinct spending patterns of male and female customers.

```
df_married=df[df['Marital_Status']=="Married"]
df_unmarried=df[df['Marital_Status']!="Married"]
results = {
    "Sample Size": [],
    "Group": [],
    "Expected Population Mean": [],
    "Standard Error":[]}


sample_sizes=[10,30,50,100,1000]
fig, axes = plt.subplots(nrows=2, ncols=5, figsize=(20, 12))
axes = axes.flatten()
for idx, sample_size in enumerate(sample_sizes):
    married_sample_means = []
    unmarried_sample_means = []

    # Generate sample means for male and female customers
    for _ in range(1000):
        married_sample = df_married['Purchase'].sample(sample_size,
replace=True)
        unmarried_sample =
df_unmarried['Purchase'].sample(sample_size, replace=True)
        married_sample_means.append(married_sample.mean())
        unmarried_sample_means.append(unmarried_sample.mean())
    results["Sample Size"].append(sample_size)
    results["Group"].append("Married")
    results["Expected Population
Mean"].append(np.array(married_sample_means).mean())
    results["Standard
Error"].append(np.array(married_sample_means).std())

    results["Sample Size"].append(sample_size)
    results["Group"].append("Unmarried")
    results["Expected Population
Mean"].append(np.array(unmarried_sample_means).mean())
    results["Standard
Error"].append(np.array(unmarried_sample_means).std())

    # Plot histograms for male sample means
    sns.histplot(married_sample_means, bins=10, ax=axes[idx],
kde=True, color='blue')
    axes[idx].set_title(f"Married Purchase Sample
Means(n={sample_size})")
```

```
    # Plot histograms for female sample means
    sns.histplot(unmarried_sample_means, bins=10, ax=axes[idx+5],
kde=True, color='red')
    axes[idx+5].set_title(f"Unmarried Purchase Sample
Means(n={sample_size})")

    # Calculate and display confidence intervals

# Adjust layout
plt.tight_layout()
plt.show()
```




```
d=pd.DataFrame(results)
d
```


```
{"summary":"{\n  \"name\": \"d\",\n  \"rows\": 10,\n  \"fields\": [\n
{\n      \"column\": \"Sample Size\",\n      \"properties\": {\n
\"dtype\": \"number\",\n        \"std\": 402,\n        \"min\": 10,\n
\"max\": 1000,\n        \"num_unique_values\": 5,\n
\"samples\": [\n          30,\n          1000,\n          50\
n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    },\n    {\n      \"column\":
\"Group\",\n      \"properties\": {\n        \"dtype\": \"category\",\
n        \"num_unique_values\": 2,\n        \"samples\": [\n
\"Unmarried\",\n          \"Married\"\n        ],\n
\"semantic_type\": \"\",\n        \"description\": \"\"\n      }\
```

```
n    },\n    {\n      \"column\": \"Expected Population Mean\",\n
\"properties\": {\n        \"dtype\": \"number\",\n        \"std\":
45.27100791965839,\n        \"min\": 9085.962800000001,\n
\"max\": 9262.7922,\n        \"num_unique_values\": 10,\n
\"samples\": [\n          9190.272006,\n          9262.7922\
n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    },\n    {\n      \"column\":
\"Standard Error\",\n      \"properties\": {\n        \"dtype\":
\"number\",\n        \"std\": 487.0721499488152,\n        \"min\":
151.30396644969574,\n        \"max\": 1561.8587714694822,\n
\"num_unique_values\": 10,\n        \"samples\": [\n
155.65511922919836,\n          1501.5559118791282\n        ],\n
\"semantic_type\": \"\",\n        \"description\": \"\"\n      }\
n    }\n  ]\n}","type":"dataframe","variable_name":"d"}
```


Insight

Relationship Between Sample Size and Standard Error:

- Observing the data, the **standard error decreases** as the **sample size increases**.

Application of Central Limit Theorem (CLT):

- Using the **Central Limit Theorem (CLT)**:
  - The **expected population mean** for **married customers** is approximately **9180**.
  - The **expected population mean** for **unmarried customers** is around **9200**.

Interpretation:

- The decrease in standard error with larger sample sizes indicates greater precision in estimating the population mean.
- The calculated population means provide a reliable estimate of the average spending behavior for married and unmarried customers.

This insight highlights the importance of larger sample sizes to reduce variability and obtain accurate population estimates.

```
sample_sizes = [10, 30, 50, 100, 500,1000]
confidence_level = 0.9

# Function to calculate confidence interval
def confidence_interval(data, confidence=0.99):
    n = len(data)
    mean = np.mean(data)
    stderr = stats.sem(data)
    margin_of_error = stderr * stats.t.ppf((1 + confidence) / 2., n-1)
    return mean, mean - margin_of_error, mean + margin_of_error

# Prepare lists to store results
results = {
    "Sample Size": [],
```

```
    "Group": [],
    "Mean": [],
    "CI Lower Bound": [],
    "CI Upper Bound": [],
    "CI Length":[],
    "t-statstic":[],
    "p-value":[],
    "interval-overlap?":[]
}

for sample_size in sample_sizes:
    married_sample_means = []
    unmarried_sample_means = []

    # Generate sample means for male and female customers
    for _ in range(1000):
        married_sample = df_married['Purchase'].sample(sample_size,
replace=True)
        unmarried_sample =
df_unmarried['Purchase'].sample(sample_size, replace=True)
        married_sample_means.append(married_sample.mean())
        unmarried_sample_means.append(unmarried_sample.mean())


    # Calculate confidence intervals for male and female
    married_mean, married_ci_lower, married_ci_upper =
confidence_interval(married_sample_means, confidence=confidence_level)
    unmarried_mean, unmarried_ci_lower, unmarried_ci_upper =
confidence_interval(unmarried_sample_means,
confidence=confidence_level)
    t_stat, p_value =
stats.ttest_ind(married_sample_means,unmarried_sample_means,
equal_var=False)
    interval_overlap=["yes" if
pd.Interval(married_ci_lower,married_ci_upper).overlaps(pd.Interval(un
married_ci_lower,unmarried_ci_upper)) else "no"][0]

    # Append results to the list
    results["Sample Size"].append(sample_size)
    results["Group"].append("Married")
    results["Mean"].append( married_mean)
    results["CI Lower Bound"].append(married_ci_lower)
    results["CI Upper Bound"].append(married_ci_upper)
    results["CI Length"].append(married_ci_upper-married_ci_lower)
    results['t-statstic'].append(t_stat)
    results['p-value'].append(p_value)
    results['interval-overlap?'].append(interval_overlap)


    results["Sample Size"].append(sample_size)
```

```
    results["Group"].append("Unmarried")
    results["Mean"].append( unmarried_mean)
    results["CI Lower Bound"].append(unmarried_ci_lower)
    results["CI Upper Bound"].append(unmarried_ci_upper)
    results["CI Length"].append(unmarried_ci_upper-unmarried_ci_lower)
    results['t-statstic'].append(t_stat)
    results['p-value'].append(p_value)
    results['interval-overlap?'].append(interval_overlap)

# Convert results to a DataFrame
ci_df = pd.DataFrame(results)

# Display the DataFrame
ci_df
```


```
{"summary":"{\n  \"name\": \"ci_df\",\n  \"rows\": 12,\n  \"fields\":
[\n    {\n      \"column\": \"Sample Size\",\n      \"properties\": {\
n        \"dtype\": \"number\",\n        \"std\": 378,\n
\"min\": 10,\n        \"max\": 1000,\n        \"num_unique_values\":
6,\n        \"samples\": [\n          10,\n          30,\n
1000\n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    },\n    {\n      \"column\":
\"Group\",\n      \"properties\": {\n        \"dtype\": \"category\",\
n        \"num_unique_values\": 2,\n        \"samples\": [\n
\"Unmarried\",\n          \"Married\"\n        ],\n
\"semantic_type\": \"\",\n        \"description\": \"\"\n      }\
n    },\n    {\n      \"column\": \"Mean\",\n      \"properties\": {\n
\"dtype\": \"number\",\n        \"std\": 34.43489876134391,\n
\"min\": 9098.2795,\n        \"max\": 9226.306900000001,\n
\"num_unique_values\": 12,\n        \"samples\": [\n
9169.522859,\n          9207.170758\n        ],\n
\"semantic_type\": \"\",\n        \"description\": \"\"\n      }\
n    },\n    {\n      \"column\": \"CI Lower Bound\",\n
\"properties\": {\n        \"dtype\": \"number\",\n        \"std\":
52.389156191113415,\n        \"min\": 9020.17657607752,\n
\"max\": 9198.953456322426,\n        \"num_unique_values\": 12,\n
\"samples\": [\n          9161.564059016551,\n
9195.671878916519\n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    },\n    {\n      \"column\": \"CI
Upper Bound\",\n      \"properties\": {\n        \"dtype\":
\"number\",\n        \"std\": 30.37234481329905,\n        \"min\":
9176.382423922481,\n        \"max\": 9263.93717503638,\n
\"num_unique_values\": 12,\n        \"samples\": [\n
9177.48165898345,\n          9218.669637083482\n        ],\n
\"semantic_type\": \"\",\n        \"description\": \"\"\n      }\
n    },\n    {\n      \"column\": \"CI Length\",\n
\"properties\": {\n        \"dtype\": \"number\",\n        \"std\":
50.90340873589512,\n        \"min\": 15.917599966898706,\n
\"max\": 164.65602734770073,\n        \"num_unique_values\": 12,\n
\"samples\": [\n          15.917599966898706,\n
```

22.99775816696274\n   ],\n   \"semantic_type\": \"\",\n \"description\": \"\"\n   }\n   },\n   {\n   \"column\": \"t-statstic\",\n   \"properties\": {\n   \"dtype\": \"number\",\n \"std\": 2.0272731762254974,\n   \"min\": -5.413552096457957,\n \"max\": 0.6738341691092667,\n   \"num_unique_values\": 6,\n \"samples\": [\n   -0.9013725977262835,\n   -0.06372570944486991\n   ],\n   \"semantic_type\": \"\",\n \"description\": \"\"\n   }\n   },\n   {\n   \"column\": \"p-value\",\n   \"properties\": {\n   \"dtype\": \"number\",\n \"std\": 0.3167287179683825,\n   \"min\": 6.922991678912336e-08,\n   \"max\": 0.9491950083888621,\n   \"num_unique_values\": 6,\n   \"samples\": [\n   0.36749918536685566,\n 0.9491950083888621\n   ],\n   \"semantic_type\": \"\",\n \"description\": \"\"\n   }\n   },\n   {\n   \"column\": \"interval-overlap?\",\n   \"properties\": {\n   \"dtype\": \"category\",\n   \"num_unique_values\": 2,\n   \"samples\": [\n   \"no\",\n   \"yes\"\n   ],\n \"semantic_type\": \"\",\n   \"description\": \"\"\n   }\n   }\n]\n}","type":"dataframe","variable_name":"ci_df"}

```
sample_sizes = [10, 30, 50, 100, 500,1000]
confidence_level = 0.95

# Function to calculate confidence interval
def confidence_interval(data, confidence=0.99):
    n = len(data)
    mean = np.mean(data)
    stderr = stats.sem(data)
    margin_of_error = stderr * stats.t.ppf((1 + confidence) / 2., n-1)
    return mean, mean - margin_of_error, mean + margin_of_error

# Prepare lists to store results
results = {
    "Sample Size": [],
    "Group": [],
    "Mean": [],
    "CI Lower Bound": [],
    "CI Upper Bound": [],
    "CI Length":[],
    "t-statstic":[],
    "p-value":[],
    "interval-overlap?":[]
}

for sample_size in sample_sizes:
    married_sample_means = []
    unmarried_sample_means = []

    # Generate sample means for male and female customers
    for _ in range(1000):
```

```
        married_sample = df_married['Purchase'].sample(sample_size,
replace=True)
        unmarried_sample =
df_unmarried['Purchase'].sample(sample_size, replace=True)
        married_sample_means.append(married_sample.mean())
        unmarried_sample_means.append(unmarried_sample.mean())


    # Calculate confidence intervals for male and female
    married_mean, married_ci_lower, married_ci_upper =
confidence_interval(married_sample_means, confidence=confidence_level)
    unmarried_mean, unmarried_ci_lower, unmarried_ci_upper =
confidence_interval(unmarried_sample_means,
confidence=confidence_level)
    t_stat, p_value =
stats.ttest_ind(married_sample_means,unmarried_sample_means,
equal_var=False)
    interval_overlap=["yes" if
pd.Interval(married_ci_lower,married_ci_upper).overlaps(pd.Interval(un
married_ci_lower,unmarried_ci_upper)) else "no"][0]

    # Append results to the list
    results["Sample Size"].append(sample_size)
    results["Group"].append("Married")
    results["Mean"].append( married_mean)
    results["CI Lower Bound"].append(married_ci_lower)
    results["CI Upper Bound"].append(married_ci_upper)
    results["CI Length"].append(married_ci_upper-married_ci_lower)
    results['t-statstic'].append(t_stat)
    results['p-value'].append(p_value)
    results['interval-overlap?'].append(interval_overlap)


    results["Sample Size"].append(sample_size)
    results["Group"].append("Unmarried")
    results["Mean"].append( unmarried_mean)
    results["CI Lower Bound"].append(unmarried_ci_lower)
    results["CI Upper Bound"].append(unmarried_ci_upper)
    results["CI Length"].append(unmarried_ci_upper-unmarried_ci_lower)
    results['t-statstic'].append(t_stat)
    results['p-value'].append(p_value)
    results['interval-overlap?'].append(interval_overlap)

# Convert results to a DataFrame
ci_df = pd.DataFrame(results)

# Display the DataFrame
ci_df
```

```
{"summary":"{\n  \"name\": \"ci_df\",\n  \"rows\": 12,\n  \"fields\":
[\n    {\n      \"column\": \"Sample Size\",\n      \"properties\": {\
n        \"dtype\": \"number\",\n        \"std\": 378,\n
\"min\": 10,\n        \"max\": 1000,\n        \"num_unique_values\":
6,\n        \"samples\": [\n          10,\n          30,\n
1000\n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    },\n    {\n      \"column\":
\"Group\",\n      \"properties\": {\n        \"dtype\": \"category\",\
n        \"num_unique_values\": 2,\n        \"samples\": [\n
\"Unmarried\",\n          \"Married\"\n        ],\n
\"semantic_type\": \"\",\n        \"description\": \"\"\n      }\
n    },\n    {\n      \"column\": \"Mean\",\n      \"properties\": {\n
\"dtype\": \"number\",\n        \"std\": 12.59568277389765,\n
\"min\": 9164.313066666666,\n        \"max\": 9214.21117,\n
\"num_unique_values\": 12,\n        \"samples\": [\n
9181.631605,\n          9193.69744\n        ],\n
\"semantic_type\": \"\",\n        \"description\": \"\"\n      }\
n    },\n    {\n      \"column\": \"CI Lower Bound\",\n
\"properties\": {\n        \"dtype\": \"number\",\n        \"std\":
34.46304740050511,\n        \"min\": 9086.232982255284,\n
\"max\": 9190.160846804409,\n        \"num_unique_values\": 12,\n
\"samples\": [\n          9171.627706422058,\n
9179.471235041252\n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    },\n    {\n      \"column\": \"CI
Upper Bound\",\n      \"properties\": {\n        \"dtype\":
\"number\",\n        \"std\": 32.133632922338606,\n        \"min\":
9191.635503577943,\n        \"max\": 9295.667084249053,\n
\"num_unique_values\": 12,\n        \"samples\": [\n
9191.635503577943,\n          9207.923644958748\n        ],\n
\"semantic_type\": \"\",\n        \"description\": \"\"\n      }\
n    },\n    {\n      \"column\": \"CI Length\",\n
\"properties\": {\n        \"dtype\": \"number\",\n        \"std\":
61.69229371381391,\n        \"min\": 19.218492391184554,\n
\"max\": 195.36123548943215,\n        \"num_unique_values\": 12,\n
\"samples\": [\n          20.007797155885783,\n
28.45240991749597\n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    },\n    {\n      \"column\": \"t-
statstic\",\n      \"properties\": {\n        \"dtype\": \"number\",\n
\"std\": 0.9027709872655769,\n        \"min\": -2.5659933300466267,\n
\"max\": 0.1204207229328179,\n        \"num_unique_values\": 6,\n
\"samples\": [\n          -0.2075922917734386,\n          -
0.6363082626292456\n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    },\n    {\n      \"column\": \"p-
value\",\n      \"properties\": {\n        \"dtype\": \"number\",\n
\"std\": 0.3316305698112675,\n        \"min\": 0.010360525755678652,\n
\"max\": 0.9041620072990304,\n        \"num_unique_values\": 6,\n
\"samples\": [\n          0.8355684800957573,\n
0.5246486199566629\n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    },\n    {\n      \"column\":
\"interval-overlap?\",\n      \"properties\": {\n        \"dtype\":
```

```
\"category\",\n        \"num_unique_values\": 1,\n        \"samples\":
[\n          \"yes\"\n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    }\n  ]\
n}","type":"dataframe","variable_name":"ci_df"}

sample_sizes = [10, 30, 50, 100, 500,1000]
confidence_level = 0.99

# Function to calculate confidence interval
def confidence_interval(data, confidence=0.99):
    n = len(data)
    mean = np.mean(data)
    stderr = stats.sem(data)
    margin_of_error = stderr * stats.t.ppf((1 + confidence) / 2., n-1)
    return mean, mean - margin_of_error, mean + margin_of_error

# Prepare lists to store results
results = {
    "Sample Size": [],
    "Group": [],
    "Mean": [],
    "CI Lower Bound": [],
    "CI Upper Bound": [],
    "CI Length":[],
    "t-statstic":[],
    "p-value":[],
    "interval-overlap?":[]
}

for sample_size in sample_sizes:
    married_sample_means = []
    unmarried_sample_means = []

    # Generate sample means for male and female customers
    for _ in range(1000):
        married_sample = df_married['Purchase'].sample(sample_size,
replace=True)
        unmarried_sample =
df_unmarried['Purchase'].sample(sample_size, replace=True)
        married_sample_means.append(married_sample.mean())
        unmarried_sample_means.append(unmarried_sample.mean())


    # Calculate confidence intervals for male and female
    married_mean, married_ci_lower, married_ci_upper =
confidence_interval(married_sample_means, confidence=confidence_level)
    unmarried_mean, unmarried_ci_lower, unmarried_ci_upper =
confidence_interval(unmarried_sample_means,
confidence=confidence_level)
    t_stat, p_value =
```

```
stats.ttest_ind(married_sample_means,unmarried_sample_means,
equal_var=False)
    interval_overlap=["yes" if
pd.Interval(married_ci_lower,married_ci_upper).overlaps(pd.Interval(un
married_ci_lower,unmarried_ci_upper)) else "no"][0]

    # Append results to the list
    results["Sample Size"].append(sample_size)
    results["Group"].append("Married")
    results["Mean"].append( married_mean)
    results["CI Lower Bound"].append(married_ci_lower)
    results["CI Upper Bound"].append(married_ci_upper)
    results["CI Length"].append(married_ci_upper-married_ci_lower)
    results['t-statstic'].append(t_stat)
    results['p-value'].append(p_value)
    results['interval-overlap?'].append(interval_overlap)


    results["Sample Size"].append(sample_size)
    results["Group"].append("Unmarried")
    results["Mean"].append( unmarried_mean)
    results["CI Lower Bound"].append(unmarried_ci_lower)
    results["CI Upper Bound"].append(unmarried_ci_upper)
    results["CI Length"].append(unmarried_ci_upper-unmarried_ci_lower)
    results['t-statstic'].append(t_stat)
    results['p-value'].append(p_value)
    results['interval-overlap?'].append(interval_overlap)

# Convert results to a DataFrame
ci_df = pd.DataFrame(results)

# Display the DataFrame
ci_df
```


```
{"summary":"{\n  \"name\": \"ci_df\",\n  \"rows\": 12,\n  \"fields\":
[\n    {\n      \"column\": \"Sample Size\",\n      \"properties\": {\
n        \"dtype\": \"number\",\n        \"std\": 378,\n
\"min\": 10,\n        \"max\": 1000,\n        \"num_unique_values\":
6,\n        \"samples\": [\n          10,\n          30,\n
1000\n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    },\n    {\n      \"column\":
\"Group\",\n      \"properties\": {\n        \"dtype\": \"category\",\
n        \"num_unique_values\": 2,\n        \"samples\": [\n
\"Unmarried\",\n          \"Married\"\n        ],\n
\"semantic_type\": \"\",\n        \"description\": \"\"\n      }\
n    },\n    {\n      \"column\": \"Mean\",\n      \"properties\": {\n
\"dtype\": \"number\",\n        \"std\": 24.45839413158684,\n
\"min\": 9152.88098,\n        \"max\": 9248.6733,\n
\"num_unique_values\": 12,\n        \"samples\": [\n
9176.935405999999,\n          9204.868314\n        ],\n
```

```
\"semantic_type\": \"\",\n        \"description\": \"\"\n      }\
n    },\n    {\n      \"column\": \"CI Lower Bound\",\n
\"properties\": {\n        \"dtype\": \"number\",\n        \"std\":
42.01906268134524,\n        \"min\": 9042.500668897077,\n
\"max\": 9187.228107023595,\n        \"num_unique_values\": 12,\n
\"samples\": [\n          9164.855917257173,\n
9187.228107023595\n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    },\n    {\n      \"column\": \"CI
Upper Bound\",\n      \"properties\": {\n        \"dtype\":
\"number\",\n        \"std\": 53.53629858183543,\n        \"min\":
9189.014894742824,\n        \"max\": 9377.107034948294,\n
\"num_unique_values\": 12,\n        \"samples\": [\n
9189.014894742824,\n          9222.508520976404\n        ],\n
\"semantic_type\": \"\",\n        \"description\": \"\"\n      }\
n    },\n    {\n      \"column\": \"CI Length\",\n
\"properties\": {\n        \"dtype\": \"number\",\n        \"std\":
82.88921289854514,\n        \"min\": 24.15897748565112,\n
\"max\": 263.3260622058442,\n        \"num_unique_values\": 12,\n
\"samples\": [\n          24.15897748565112,\n
35.280413952808885\n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    },\n    {\n      \"column\": \"t-
statstic\",\n      \"properties\": {\n        \"dtype\": \"number\",\n
\"std\": 1.5523634356526064,\n        \"min\": -3.1618192047820166,\n
\"max\": 1.045457456143595,\n        \"num_unique_values\": 6,\n
\"samples\": [\n          1.045457456143595,\n
0.6771820820227317\n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    },\n    {\n      \"column\": \"p-
value\",\n      \"properties\": {\n        \"dtype\": \"number\",\n
\"std\": 0.249340339759976,\n        \"min\": 0.0015914775922873132,\n
\"max\": 0.6496250354303428,\n        \"num_unique_values\": 6,\n
\"samples\": [\n          0.2959381342841791,\n
0.4983688321758063\n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    },\n    {\n      \"column\":
\"interval-overlap?\",\n      \"properties\": {\n        \"dtype\":
\"category\",\n        \"num_unique_values\": 1,\n        \"samples\":
[\n          \"yes\"\n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    }\n  ]\
n}","type":"dataframe","variable_name":"ci_df"}
```


Insight

Relationship Between Confidence Level and Confidence Interval Length:

- From the above data, it is evident that as the **confidence level increases** (90%, 95%, 99%), the **confidence interval length increases**. This is because higher confidence levels require a broader range to account for the increased certainty in capturing the population mean.

Relationship Between Sample Size and Confidence Interval Length:

- As the **sample size increases**, the **confidence interval length decreases**. This indicates greater precision in estimating the population mean with larger sample sizes, as variability reduces with more data.

Interpretation:

- A **higher confidence level** ensures a greater likelihood of the interval containing the true population mean, but at the cost of reduced precision.
- A **larger sample size** enhances precision, reducing the confidence interval length, and providing a more accurate estimate of the population mean.

This insight demonstrates the trade-off between confidence level and interval length and the importance of sufficient sample size for precise population mean estimation.

Insight

Population Mean Estimates:

- For **married customers** (sample size = 1000):
  - **90% Confidence Level**: Population mean is estimated to lie between **9176** and **9193**.
  - **95% Confidence Level**: Population mean is estimated to lie between **9181** and **9201**.
  - **99% Confidence Level**: Population mean is estimated to lie between **9179** and **9205**.
- For **unmarried customers** (sample size = 1000):
  - **90% Confidence Level**: Population mean is estimated to lie between **9188** and **9205**.
  - **95% Confidence Level**: Population mean is estimated to lie between **9185** and **9204**.
  - **99% Confidence Level**: Population mean is estimated to lie between **9178** and **9203**.

Interpretation:

- The **confidence intervals** for both groups show some overlap, indicating similarity in the average spending behavior of married and unmarried customers.
- The **population mean** for unmarried customers consistently trends slightly higher than that of married customers, regardless of the confidence level.

This insight reinforces the importance of analyzing confidence intervals to derive accurate population estimates and compare spending behaviors between customer groups.

Insight

Comparison of Purchase Amounts Between Married and Unmarried Customers:

- **Overlapping Confidence Intervals**:
  - For all **confidence levels** and **sample sizes**, the **confidence intervals** (CIs) for both married and unmarried customers overlap.

- The **medians** for both groups are also **almost similar**, further reinforcing the observation.

Interpretation:

- The **overlap of confidence intervals** suggests that there is no **significant statistical difference** in the purchase amounts between **married** and **unmarried** customers.
- The similarity in the **median purchase amounts** for both groups further supports this conclusion, indicating that both groups exhibit similar spending behaviors.

This insight suggests that targeting strategies based on marital status may not significantly impact purchasing behavior, and marketing efforts can be more generalized across these groups.

Recommendations

General Promotions:

- Since there is no significant difference in spending between **married** and **unmarried** customers, you can implement **marketing strategies** and **promotions** that target both groups equally.

Broad Offers:

- Create **broad promotions** that appeal to **all customers** without distinguishing between marital status. This approach ensures **inclusivity** and maximizes **reach**, allowing you to engage a larger customer base.

```
def check_overlap(ci1, ci2):
    lower1, upper1 = ci1
    lower2, upper2 = ci2
    return not (upper1 < lower2 or upper2 < lower1)

results = {
    "Sample Size": [],
    "Group": [],
    "Expected Population Mean": [],
    "Standard Error":[]}
feature = 'Age'
grouped = df.groupby(feature)

for sample_size in sample_sizes:
    for category, group_data in grouped:
        sample_means = []

        # Generate sample means for each category
        for _ in range(1000):
            sample = group_data['Purchase'].sample(sample_size,
replace=True)
            sample_means.append(sample.mean())

        # Calculate confidence intervals for the sample means
```

```
        # Append results to the list
        results["Sample Size"].append(sample_size)
        results["Group"].append(category)
        results["Expected Population
Mean"].append(np.array(sample_means).mean())
        results["Standard Error"].append(np.array(sample_means).std())

d=pd.DataFrame(results)
d
```


```
{"summary":"{\n  \"name\": \"d\",\n  \"rows\": 42,\n  \"fields\": [\n
{\n      \"column\": \"Sample Size\",\n      \"properties\": {\n
\"dtype\": \"number\",\n        \"std\": 366,\n        \"min\": 10,\n
\"max\": 1000,\n        \"num_unique_values\": 6,\n
\"samples\": [\n          10,\n          30,\n          1000\
n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    },\n    {\n      \"column\":
\"Group\",\n      \"properties\": {\n        \"dtype\": \"category\",\
n        \"num_unique_values\": 7,\n        \"samples\": [\n
\"0-17\",\n          \"18-25\",\n          \"51-55\"\n        ],\n
\"semantic_type\": \"\",\n        \"description\": \"\"\n      }\
n    },\n    {\n      \"column\": \"Expected Population Mean\",\n
\"properties\": {\n        \"dtype\": \"number\",\n        \"std\":
163.72457684900283,\n        \"min\": 8817.748599999999,\n
\"max\": 9504.809600000002,\n        \"num_unique_values\": 42,\n
\"samples\": [\n          9147.83898,\n          9182.785066666667,\n
9153.63803333333\n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    },\n    {\n      \"column\":
\"Standard Error\",\n      \"properties\": {\n        \"dtype\":
\"number\",\n        \"std\": 474.0369813591781,\n        \"min\":
150.607685112853,\n        \"max\": 1597.0647873702683,\n
\"num_unique_values\": 42,\n        \"samples\": [\n
491.9515827696457,\n          867.8683283817347,\n
914.6947003072294\n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    }\n  ]\
n}","type":"dataframe","variable_name":"d"}
```


```
sample_sizes = [10, 30, 100,1000]
confidence_level = 0.9
feature = 'Age'  # Replace with the name of your feature


# Prepare a DataFrame to store results
results = {
    "Sample Size": [],
    "Category": [],
    "Mean": [],
    "CI Lower Bound": [],
    "CI Upper Bound": []
```

```
}

# Group by the feature categories
grouped = df.groupby(feature)

for sample_size in sample_sizes:
    for category, group_data in grouped:
        sample_means = []

        # Generate sample means for each category
        for _ in range(1000):
            sample = group_data['Purchase'].sample(sample_size,
replace=True)
            sample_means.append(sample.mean())

        # Calculate confidence intervals for the sample means
        mean, ci_lower, ci_upper = confidence_interval(sample_means,
confidence=confidence_level)

        # Append results to the list
        results["Sample Size"].append(sample_size)
        results["Category"].append(category)
        results["Mean"].append(mean)
        results["CI Lower Bound"].append(ci_lower)
        results["CI Upper Bound"].append(ci_upper)

# Convert results to a DataFrame
ci_df = pd.DataFrame(results)
overlap_info = []

# Iterate over each sample size group
for sample_size in sample_sizes:
    subset_df = ci_df[ci_df['Sample Size'] == sample_size]
    categories = subset_df['Category'].tolist()

    for index, row in subset_df.iterrows():
        current_category = row['Category']
        current_ci = (row['CI Lower Bound'], row['CI Upper Bound'])
        overlaps = []

        for comp_index, comp_row in subset_df.iterrows():
            if comp_row['Category'] != current_category:
                compare_ci = (comp_row['CI Lower Bound'], comp_row['CI
Upper Bound'])
                if check_overlap(current_ci, compare_ci):
                    overlaps.append(comp_row['Category'])

        overlap_info.append(overlaps)

# Map overlaps back to the original DataFrame
```

```
ci_df['Overlaps With'] = pd.Series(overlap_info, index=ci_df.index)



# Display the DataFrame
ci_df
```


```
{"summary":"{\n  \"name\": \"ci_df\",\n  \"rows\": 28,\n  \"fields\":
[\n    {\n      \"column\": \"Sample Size\",\n      \"properties\": {\
n        \"dtype\": \"number\",\n        \"std\": 421,\n
\"min\": 10,\n        \"max\": 1000,\n        \"num_unique_values\":
4,\n        \"samples\": [\n          30,\n          1000,\n
10\n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    },\n    {\n      \"column\":
\"Category\",\n      \"properties\": {\n        \"dtype\":
\"category\",\n        \"num_unique_values\": 7,\n        \"samples\":
[\n          \"0-17\",\n          \"18-25\",\n          \"51-55\"\n
],\n        \"semantic_type\": \"\",\n        \"description\": \"\"\n
}\n    },\n    {\n      \"column\": \"Mean\",\n      \"properties\":
{\n        \"dtype\": \"number\",\n        \"std\":
157.8161050925903,\n        \"min\": 8859.976148,\n        \"max\":
9487.911,\n        \"num_unique_values\": 28,\n        \"samples\": [\
n          9212.4503,\n          9134.067254000001,\n
9104.579300000001\n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    },\n    {\n      \"column\": \"CI
Lower Bound\",\n      \"properties\": {\n        \"dtype\":
\"number\",\n        \"std\": 158.44891503574465,\n        \"min\":
8847.392303748646,\n        \"max\": 9422.172308513795,\n
\"num_unique_values\": 28,\n        \"samples\": [\n
9166.852808090185,\n          9126.366796624312,\n
9057.786165945892\n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    },\n    {\n      \"column\": \"CI
Upper Bound\",\n      \"properties\": {\n        \"dtype\":
\"number\",\n        \"std\": 162.15226485114366,\n        \"min\":
8868.402137977811,\n        \"max\": 9571.24982738242,\n
\"num_unique_values\": 28,\n        \"samples\": [\n
9258.047791909816,\n          9141.76771137569,\n
9151.37243405411\n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    },\n    {\n      \"column\":
\"Overlaps With\",\n      \"properties\": {\n        \"dtype\":
\"object\",\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    }\n  ]\
n}","type":"dataframe","variable_name":"ci_df"}
```


```
sample_sizes = [10, 30, 100,1000]
confidence_level = 0.95
feature = 'Age'  # Replace with the name of your feature


# Prepare a DataFrame to store results
```

```
results = {
    "Sample Size": [],
    "Category": [],
    "Mean": [],
    "CI Lower Bound": [],
    "CI Upper Bound": []
}

# Group by the feature categories
grouped = df.groupby(feature)

for sample_size in sample_sizes:
    for category, group_data in grouped:
        sample_means = []

        # Generate sample means for each category
        for _ in range(1000):
            sample = group_data['Purchase'].sample(sample_size,
replace=True)
            sample_means.append(sample.mean())

        # Calculate confidence intervals for the sample means
        mean, ci_lower, ci_upper = confidence_interval(sample_means,
confidence=confidence_level)

        # Append results to the list
        results["Sample Size"].append(sample_size)
        results["Category"].append(category)
        results["Mean"].append(mean)
        results["CI Lower Bound"].append(ci_lower)
        results["CI Upper Bound"].append(ci_upper)

# Convert results to a DataFrame
ci_df = pd.DataFrame(results)
overlap_info = []

# Iterate over each sample size group
for sample_size in sample_sizes:
    subset_df = ci_df[ci_df['Sample Size'] == sample_size]
    categories = subset_df['Category'].tolist()

    for index, row in subset_df.iterrows():
        current_category = row['Category']
        current_ci = (row['CI Lower Bound'], row['CI Upper Bound'])
        overlaps = []

        for comp_index, comp_row in subset_df.iterrows():
            if comp_row['Category'] != current_category:
                compare_ci = (comp_row['CI Lower Bound'], comp_row['CI
Upper Bound'])
```

```
                if check_overlap(current_ci, compare_ci):
                    overlaps.append(comp_row['Category'])

        overlap_info.append(overlaps)

# Map overlaps back to the original DataFrame
ci_df['Overlaps With'] = pd.Series(overlap_info, index=ci_df.index)



# Display the DataFrame
ci_df
```


```
{"summary":"{\n  \"name\": \"ci_df\",\n  \"rows\": 28,\n  \"fields\":
[\n    {\n      \"column\": \"Sample Size\",\n      \"properties\": {\
n        \"dtype\": \"number\",\n        \"std\": 421,\n
\"min\": 10,\n        \"max\": 1000,\n        \"num_unique_values\":
4,\n        \"samples\": [\n          30,\n          1000,\n
10\n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    },\n    {\n      \"column\":
\"Category\",\n      \"properties\": {\n        \"dtype\":
\"category\",\n        \"num_unique_values\": 7,\n        \"samples\":
[\n          \"0-17\",\n          \"18-25\",\n          \"51-55\"\n
],\n        \"semantic_type\": \"\",\n        \"description\": \"\"\n
}\n    },\n    {\n      \"column\": \"Mean\",\n      \"properties\":
{\n        \"dtype\": \"number\",\n        \"std\":
163.5046852057863,\n        \"min\": 8864.65524,\n        \"max\":
9481.470700000002,\n        \"num_unique_values\": 28,\n
\"samples\": [\n          9183.3983,\n          9126.269331,\n
9135.322466666667\n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    },\n    {\n      \"column\": \"CI
Lower Bound\",\n      \"properties\": {\n        \"dtype\":
\"number\",\n        \"std\": 168.340426959006,\n        \"min\":
8787.191591542536,\n        \"max\": 9419.27417689713,\n
\"num_unique_values\": 28,\n        \"samples\": [\n
9127.795505659222,\n          9116.662215439888,\n
9080.781043920186\n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    },\n    {\n      \"column\": \"CI
Upper Bound\",\n      \"properties\": {\n        \"dtype\":
\"number\",\n        \"std\": 165.32615730965293,\n        \"min\":
8874.88075729896,\n        \"max\": 9578.930884593248,\n
\"num_unique_values\": 28,\n        \"samples\": [\n
9239.00109434078,\n          9135.87644656011,\n
9189.863889413147\n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    },\n    {\n      \"column\":
\"Overlaps With\",\n      \"properties\": {\n        \"dtype\":
\"object\",\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    }\n  ]\
n}","type":"dataframe","variable_name":"ci_df"}
```

```
sample_sizes = [10, 30, 100,1000]
confidence_level = 0.99
feature = 'Age'  # Replace with the name of your feature


# Prepare a DataFrame to store results
results = {
    "Sample Size": [],
    "Category": [],
    "Mean": [],
    "CI Lower Bound": [],
    "CI Upper Bound": []
}

# Group by the feature categories
grouped = df.groupby(feature)

for sample_size in sample_sizes:
    for category, group_data in grouped:
        sample_means = []

        # Generate sample means for each category
        for _ in range(1000):
            sample = group_data['Purchase'].sample(sample_size,
replace=True)
            sample_means.append(sample.mean())

        # Calculate confidence intervals for the sample means
        mean, ci_lower, ci_upper = confidence_interval(sample_means,
confidence=confidence_level)

        # Append results to the list
        results["Sample Size"].append(sample_size)
        results["Category"].append(category)
        results["Mean"].append(mean)
        results["CI Lower Bound"].append(ci_lower)
        results["CI Upper Bound"].append(ci_upper)

# Convert results to a DataFrame
ci_df = pd.DataFrame(results)
overlap_info = []

# Iterate over each sample size group
for sample_size in sample_sizes:
    subset_df = ci_df[ci_df['Sample Size'] == sample_size]
    categories = subset_df['Category'].tolist()

    for index, row in subset_df.iterrows():
        current_category = row['Category']
        current_ci = (row['CI Lower Bound'], row['CI Upper Bound'])
        overlaps = []
```

```
        for comp_index, comp_row in subset_df.iterrows():
            if comp_row['Category'] != current_category:
                compare_ci = (comp_row['CI Lower Bound'], comp_row['CI
Upper Bound'])
                if check_overlap(current_ci, compare_ci):
                    overlaps.append(comp_row['Category'])

        overlap_info.append(overlaps)

# Map overlaps back to the original DataFrame
ci_df['Overlaps With'] = pd.Series(overlap_info, index=ci_df.index)



# Display the DataFrame
ci_df
```


```
{"summary":"{\n  \"name\": \"ci_df\",\n  \"rows\": 28,\n  \"fields\":
[\n    {\n      \"column\": \"Sample Size\",\n      \"properties\": {\
n        \"dtype\": \"number\",\n        \"std\": 421,\n
\"min\": 10,\n        \"max\": 1000,\n        \"num_unique_values\":
4,\n        \"samples\": [\n          30,\n          1000,\n
10\n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    },\n    {\n      \"column\":
\"Category\",\n      \"properties\": {\n        \"dtype\":
\"category\",\n        \"num_unique_values\": 7,\n        \"samples\":
[\n          \"0-17\",\n          \"18-25\",\n          \"51-55\"\n
],\n        \"semantic_type\": \"\",\n        \"description\": \"\"\n
}\n    },\n    {\n      \"column\": \"Mean\",\n      \"properties\":
{\n        \"dtype\": \"number\",\n        \"std\":
163.49732996328882,\n        \"min\": 8839.702433333334,\n
\"max\": 9456.433166666668,\n        \"num_unique_values\": 28,\n
\"samples\": [\n          9163.4513,\n          9122.753323,\n
9132.247666666668\n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    },\n    {\n      \"column\": \"CI
Lower Bound\",\n      \"properties\": {\n        \"dtype\":
\"number\",\n        \"std\": 169.16326995811568,\n        \"min\":
8724.371900469745,\n        \"max\": 9412.696363086003,\n
\"num_unique_values\": 28,\n        \"samples\": [\n
9089.702872338918,\n          9110.405710511428,\n
9055.948294598566\n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    },\n    {\n      \"column\": \"CI
Upper Bound\",\n      \"properties\": {\n        \"dtype\":
\"number\",\n        \"std\": 169.03729875794411,\n        \"min\":
8875.822361646788,\n        \"max\": 9578.58614793393,\n
\"num_unique_values\": 28,\n        \"samples\": [\n
9237.199727661084,\n          9135.100935488574,\n
9208.54703873477\n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    },\n    {\n      \"column\":
```

```
\"Overlaps With\",\n      \"properties\": {\n        \"dtype\":
\"object\",\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    }\n  ]\
n}","type":"dataframe","variable_name":"ci_df"}
```


Insight

Comparison of Confidence Intervals for Age Groups:

- **Overlapping Confidence Intervals**:
  - For all **confidence levels** (90%, 95%, 99%) and **sample sizes** of 100, the **confidence intervals** (CIs) for the age groups **18-25** and **46-50** are **overlapping**.
  - The **CIs** for age groups **36-45** and **55+** are also **overlapping**.
  - However, the **CIs** for **age groups 0-17**, **26-35**, and **51-55 do not overlap** with any other group.

Interpretation:

- The **overlap of confidence intervals** between certain age groups suggests that their **purchase behaviors** may be statistically similar.
- The **non-overlapping CIs** for other age groups indicate **potential differences** in purchasing behavior between those specific groups.

Recommendations

For Overlapping Age Groups (18-25 & 46-50; 36-45 & 55+):

- **Unified Campaigns**:
  Since there are no significant differences in spending between these overlapping groups, create **unified marketing campaigns** that address the common interests and spending behaviors of these age groups. By combining efforts for these groups, you can maximize reach and engagement.

- **Standard Offers**:
  Implement **promotions** or **discounts** that appeal to both overlapping groups without needing to differentiate between them. Offering generalized discounts or deals based on common characteristics (e.g., price sensitivity or product preferences) will be effective.

For Non-Overlapping Age Groups (0-17, 26-35, 51-55):

- **Segment-Specific Promotions**:
  Develop **targeted marketing strategies** for each of these age groups. Create **tailored promotions** and offers that address the specific spending patterns and preferences of these groups. Understanding each age group's unique spending behavior allows for more personalized and effective marketing.

- **Product Focus**:
  Adjust **inventory** and **product offerings** to align with the distinct needs and interests of these age groups. For example, products for younger customers (18-25)

might focus on trendy or budget-friendly items, while products for older groups (51-55) may prioritize comfort or long-lasting quality.

By implementing these strategies, you can ensure that your marketing efforts are both efficient and impactful, targeting the right segments with the right approach.