walmart-case-study-5-1

April 16, 2025

[1]:
```
import numpy as np
import pandas as pd
import scipy.stats as stats
import statsmodels.api as sm
import matplotlib
import matplotlib.pyplot as plt
import seaborn as sns
```

[4]:
```
data=pd.read_csv('/content/walmart_data.txt')
```

[5]:
```
data
```

[5]:

	User_ID	Product_ID	Gender	Age	Occupation	City_Category	Stay_In_Current_City_Years	Marital_Status	Product_Category	Purchase
0	1000001	P00069042	F	0-17	10.0	A	2	0.0	3.0	8370.0
1	1000001	P00248942	F	0-17	10.0	A	2	0.0	1.0	15200.0
2	1000001	P00087842	F	0-17	10.0	A	2	0.0	12.0	1422.0
3	1000001	P00085442	F	0-17	10.0	A	2	0.0	12.0	1057.0
4	1000002	P00285442	M	55+	16.0	C	4+	0.0	8.0	7969.0
…	…	…	…	…	…	…	…	…	…	…
125209	1001306	P00198742	M	26-35	3.0	B	3	0.0	1.0	19446.0
125210	1001306	P00332142	M	26-35	3.0	B	3	0.0	8.0	8132.0
125211	1001306	P00168442	M	26-35	3.0	B	3	0.0	5.0	3500.0
125212	1001306	P00002042	M	26-35	3.0	B	3	0.0	1.0	15270.0
125213	10	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN

```
[125214 rows x 10 columns]
```

[4]:
```
data.isnull().sum()
```

[4]:
```
User_ID                       0
Product_ID                    0
Gender                        1
Age                           1
Occupation                    1
City_Category                 1
Stay_In_Current_City_Years    1
Marital_Status                1
Product_Category              1
Purchase                      1
dtype: int64
```

[6]:
```
data.nunique()
```

[6]:
```
User_ID                       5692
Product_ID                    3229
Gender                           2
Age                              7
Occupation                      21
City_Category                    3
Stay_In_Current_City_Years       5
Marital_Status                   2
Product_Category                18
Purchase                     13038
dtype: int64
```

[8]:
```
data.shape
```

[8]:
```
(75133, 10)
```

[9]:
```
data.describe()
```

[9]:

	User_ID	Occupation	Marital_Status	Product_Category	\
count	7.513300e+04	75132.000000	75132.000000	75132.000000	
mean	1.002905e+06	8.102500	0.406937	5.302228	
std	1.671908e+03	6.514722	0.491266	3.728501	
min	1.000001e+06	0.000000	0.000000	1.000000	
25%	1.001447e+06	2.000000	0.000000	1.000000	
50%	1.002946e+06	7.000000	0.000000	5.000000	
75%	1.004310e+06	14.000000	1.000000	8.000000	
max	1.006040e+06	20.000000	1.000000	18.000000	

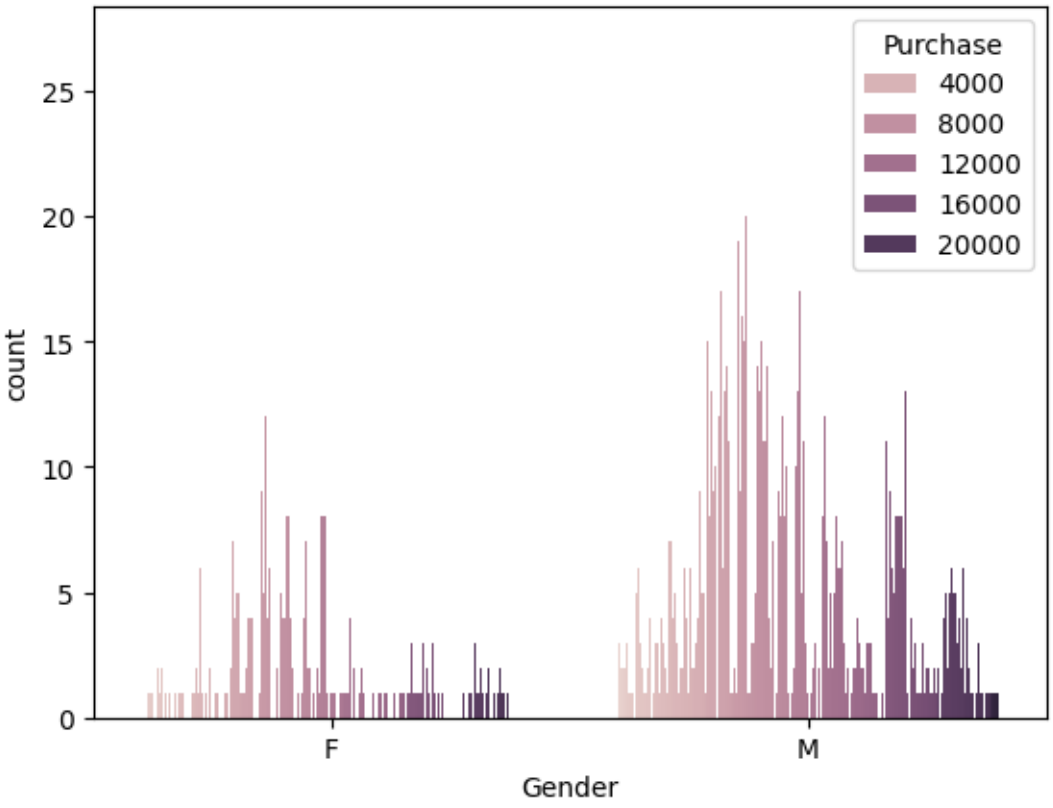
```
           Purchase
count  75132.000000
mean    9298.807619
std     4965.135617
min      185.000000
25%     5863.000000
50%     8051.000000
75%    12043.000000
max    23958.000000
```

[10]:
```
data.head(5)
```

[10]:
```
   User_ID Product_ID Gender   Age  Occupation City_Category  \
0  1000001  P00069042      F  0-17        10.0             A
1  1000001  P00248942      F  0-17        10.0             A
2  1000001  P00087842      F  0-17        10.0             A
3  1000001  P00085442      F  0-17        10.0             A
4  1000002  P00285442      M   55+        16.0             C

  Stay_In_Current_City_Years  Marital_Status  Product_Category  Purchase
0                          2             0.0               3.0    8370.0
1                          2             0.0               1.0   15200.0
2                          2             0.0              12.0    1422.0
3                          2             0.0              12.0    1057.0
4                         4+             0.0               8.0    7969.0
```

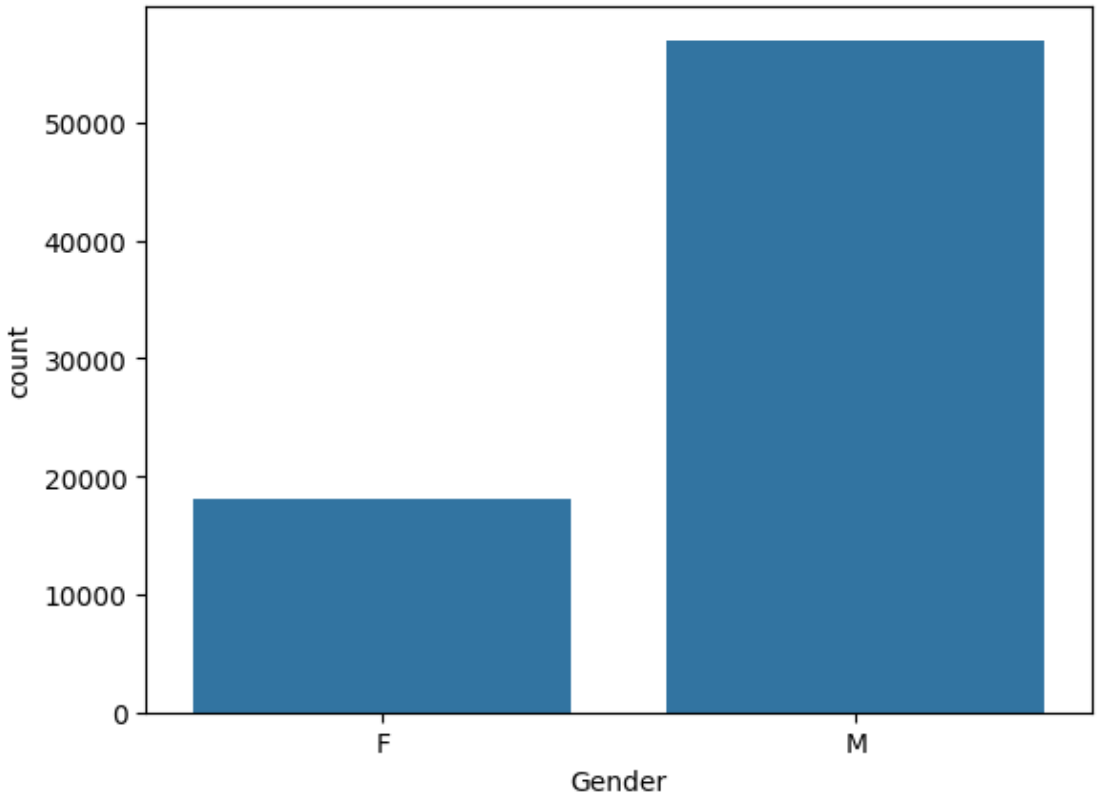
[14]:
```
sns.countplot(x=data['Gender'],data=data,hue=data['Purchase'])
plt.show()
```

```
/usr/local/lib/python3.11/dist-packages/IPython/core/pylabtools.py:151:
UserWarning: Creating legend with loc="best" can be slow with large amounts of
data.
  fig.canvas.print_figure(bytes_io, **kw)
```



[16]:
```
sns.countplot(x=data['Gender'],data=data)
```

[16]:
```
<Axes: xlabel='Gender', ylabel='count'>
```



[20]:
```
data['Occupation'].value_counts().sort_values(ascending=False)
```

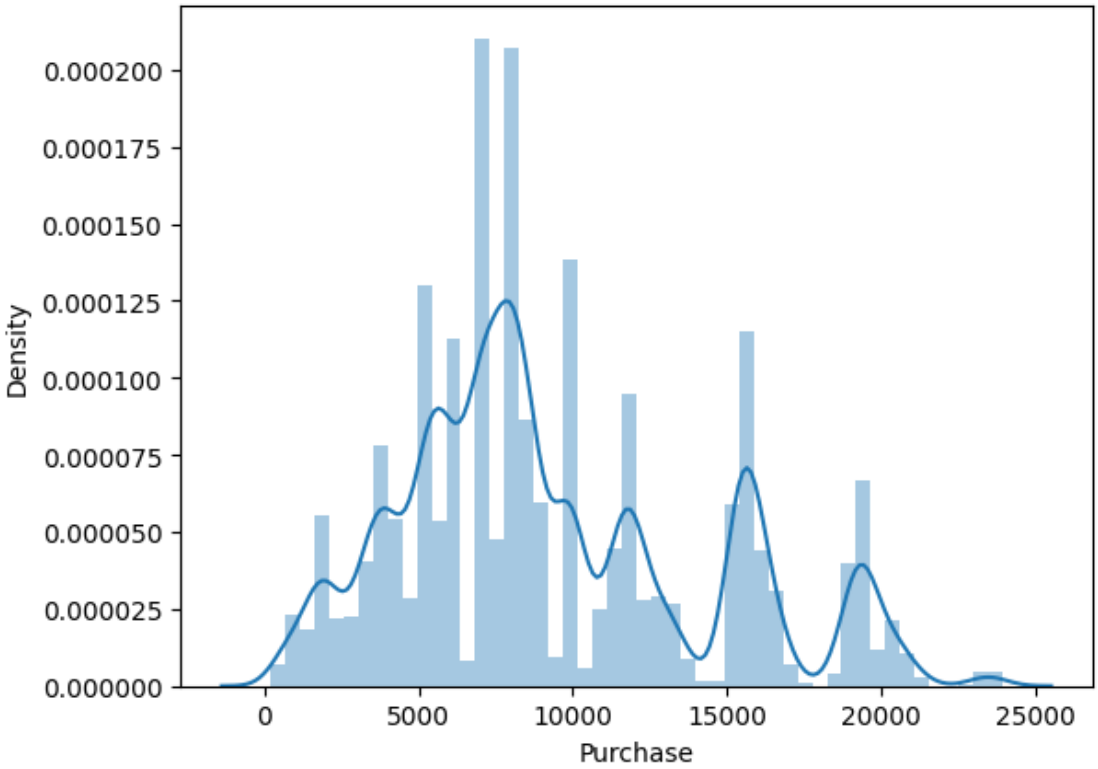
[20]:
```
Occupation
4.0     10087
0.0      9518
7.0      7979
1.0      6136
17.0     5518
20.0     4480
12.0     4248
14.0     3837
2.0      3635
16.0     3463
6.0      2810
3.0      2498
10.0     1741
15.0     1649
5.0      1646
11.0     1571
19.0     1200
13.0     1094
```

```
18.0      958
9.0       854
8.0       210
Name: count, dtype: int64
```

[21]:
```
sns.distplot(data['Purchase'])
```

```
<ipython-input-21-0a2b175eddea>:1: UserWarning:

`distplot` is a deprecated function and will be removed in seaborn v0.14.0.

Please adapt your code to use either `displot` (a figure-level function with
similar flexibility) or `histplot` (an axes-level function for histograms).

For a guide to updating your code to use the new functions, please see
https://gist.github.com/mwaskom/de44147ed2974457ad6372750bbe5751

  sns.distplot(data['Purchase'])
```

[21]: <Axes: xlabel='Purchase', ylabel='Density'>



[8]:
```
q1=data['Purchase'].quantile(0.25)
```

[7]:
```
q3=data['Purchase'].quantile(0.75)
```

[9]:
```
IQR=q3-q1
```

[12]:
```
lower_bound = q1 - 1.5 * IQR
upper_bound = q3 + 1.5 * IQR
```

[13]:
```
outliers=data[~((data["Purchase"] >= lower_bound) & (data["Purchase"] <=␣
 ↪upper_bound))]
```

[14]:
```
print(outliers)
```

```
        User_ID Product_ID Gender    Age  Occupation City_Category  \
343     1000058  P00117642      M  26-35         2.0             B
375     1000062  P00119342      F  36-45         3.0             A
652     1000126  P00087042      M  18-25         9.0             B
736     1000139  P00159542      F  26-35        20.0             C
1041    1000175  P00052842      F  26-35         2.0             B
…          …        …        …    …    …         …            …
124628  1001243  P00116142      M  36-45        15.0             B
124855  1001272  P00052842      M  18-25        20.0             B
124859  1001273  P00117642      M  36-45         2.0             C
125093  1001298  P00119342      M  36-45         6.0             B
125213       10         NaN    NaN    NaN         NaN           NaN

       Stay_In_Current_City_Years  Marital_Status  Product_Category  Purchase
343                             3             0.0              10.0   23603.0
375                             1             0.0              10.0   23792.0
652                             1             0.0              10.0   23233.0
736                             2             0.0              10.0   23595.0
1041                            1             0.0              10.0   23341.0
…                             …             …                …       …
124628                         4+             1.0              10.0   23690.0
124855                          0             0.0              10.0   23104.0
124859                          3             1.0              10.0   23550.0
125093                         4+             0.0              10.0   23237.0
125213                        NaN            NaN               NaN       NaN

[629 rows x 10 columns]
```

[31]:
```
data.shape
```

[31]:
```
(75133, 10)
```

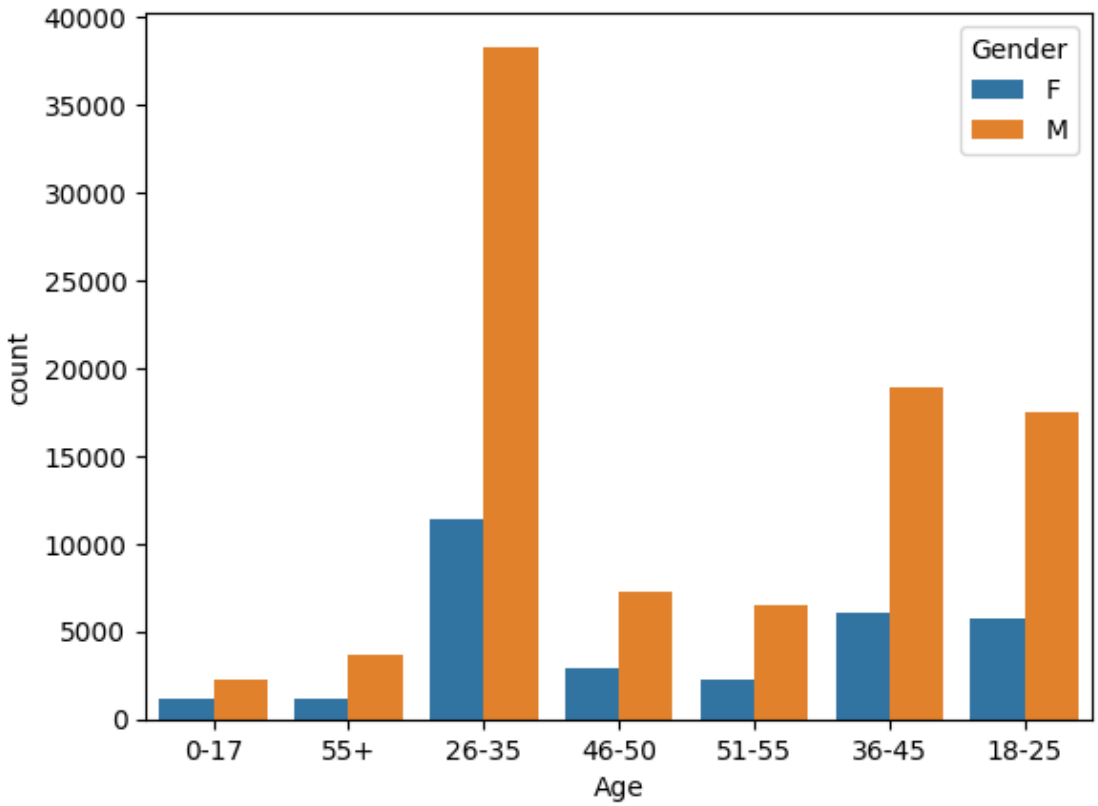
[15]:
```
Pur_median=data['Purchase'].median()
Pur_mean=data['Purchase'].mean()
```

[16]:
```
print(f"Purchase Median:{Pur_median}")
print(f"Purchase Mean:{Pur_mean}")
```

```
Purchase Median:8052.0
Purchase Mean:9305.25529298076
```

[17]:
```
sns.countplot(x=data['Age'],data=data,hue=data['Gender'])
```

[17]:
```
<Axes: xlabel='Age', ylabel='count'>
```



[18]:
```
# gender based mean value and median value diffence
data.groupby(data['Gender'])['Purchase'].agg(['mean','median'])
```

[18]:

Gender	mean	median
F	8776.171117	7924.0
M	9477.447919	8101.0

[19]:
```
# Age based mean value and median value diffence

data.groupby("Age")["Purchase"].agg(["mean","median"])
```

[19]:
```
              mean  median
Age
0-17   9087.653120  8059.0
18-25  9200.716442  8021.0
26-35  9294.094459  8038.0
36-45  9365.214807  8062.0
46-50  9235.462522  8033.0
51-55  9617.439707  8154.0
55+    9349.828090  8119.0
```

[20]:
```
gender_avg_purchase = data.groupby('Gender')['Purchase'].mean()
```

[21]:
```
df=data
```

[22]:
```
from scipy import stats
```

[44]:
```
#calculating sample mean and standard deviation
```

[25]:
```
mean_female = df[df['Gender']=='F']['Purchase'].mean()
```

[26]:
```
mean_male=df[df['Gender']=='M']['Purchase'].mean()
```

[27]:
```
female_sem=stats.sem(df[df['Gender']=='F']['Purchase'])
```

[28]:
```
male_sem=stats.sem(df[df['Gender']=='M']['Purchase'])
```

[30]:
```
df_female=len(df[df['Gender']=='F'])-1
```

[31]:
```
df_male=len(df[df['Gender']=='M'])
```

[32]:
```
alpha=0.95
```

[33]:
```
confidence_interval_female=stats.t.
 ↪interval(alpha,df_female,loc=mean_female,scale=female_sem)
```

[34]:
```
confidence_interval_male=stats.t.
 ↪interval(alpha,df_male,loc=mean_male,scale=male_sem)
```

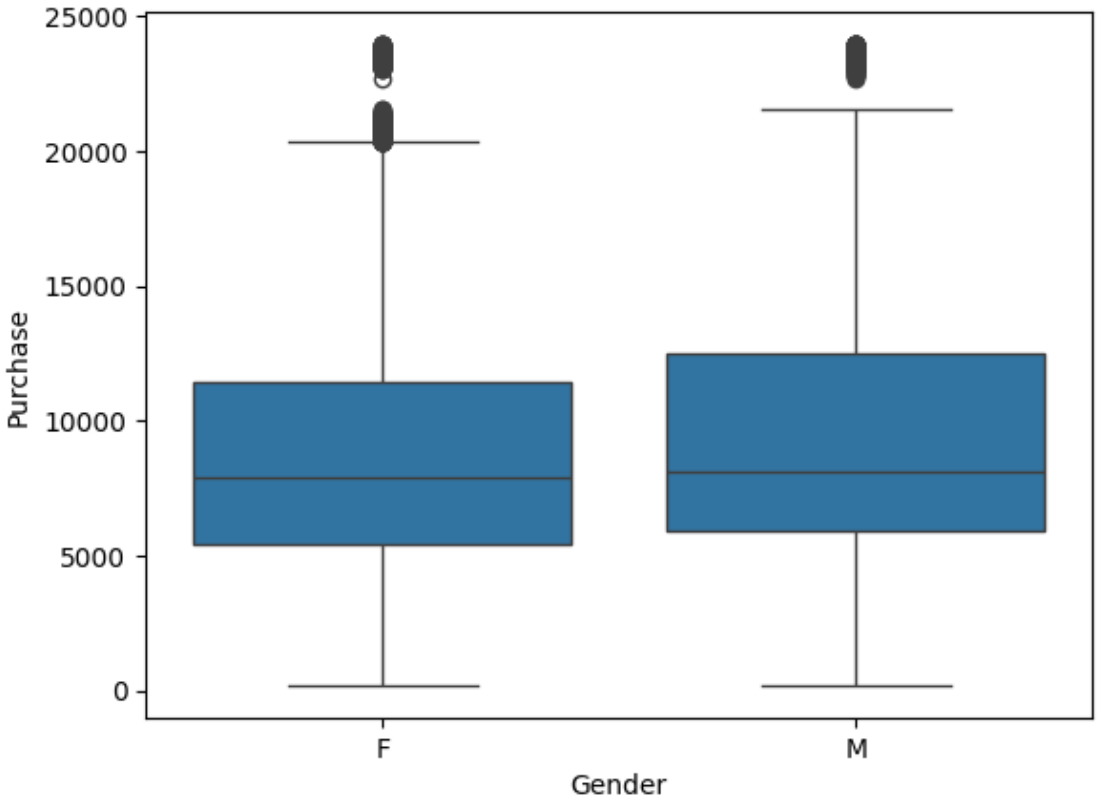
[35]:
```
print(f"Confidence Interval for Female: {confidence_interval_female}")
print(f"Confidence Interval for Male: {confidence_interval_male}")
```

```
Confidence Interval for Female: (np.float64(8723.716394711822),
np.float64(8828.625839797854))
Confidence Interval for Male: (np.float64(9445.203720230928),
np.float64(9509.692117513072))
```

[36]:
```
#purchase by gender
```

[39]:
```
sns.boxplot(x=df['Gender'],y=df['Purchase'],data=df)
```

[39]:
```
<Axes: xlabel='Gender', ylabel='Purchase'>
```



[41]:
```
# Group by 'Marital_Status' and calculate mean purchase
```

[45]:
```
df['Marital_Status'].replace({0: 'Unmarried', 1: 'Married'}, inplace=True)
```

```
<ipython-input-45-6c1147bc9e60>:1: FutureWarning: A value is trying to be set on
a copy of a DataFrame or Series through chained assignment using an inplace
method.
The behavior will change in pandas 3.0. This inplace method will never work
because the intermediate object on which we are setting values always behaves as
a copy.

For example, when doing 'df[col].method(value, inplace=True)', try using
'df.method({col: value}, inplace=True)' or df[col] = df[col].method(value)
instead, to perform the operation inplace on the original object.


  df['Marital_Status'].replace({0: 'Unmarried', 1: 'Married'}, inplace=True)
```

[46]:
```
df.head(5)
```

[46]:

	User_ID	Product_ID	Gender	Age	Occupation	City_Category	Stay_In_Current_City_Years	Marital_Status	Product_Category	Purchase
0	1000001	P00069042	F	0-17	10.0	A	2	Unmarried	3.0	8370.0
1	1000001	P00248942	F	0-17	10.0	A	2	Unmarried	1.0	15200.0
2	1000001	P00087842	F	0-17	10.0	A	2	Unmarried	12.0	1422.0
3	1000001	P00085442	F	0-17	10.0	A	2	Unmarried	12.0	1057.0
4	1000002	P00285442	M	55+	16.0	C	4+	Unmarried	8.0	7969.0

[47]:
```
marital_status_spending = df.groupby('Marital_Status')['Purchase'].mean()
print(marital_status_spending)
```

```
Marital_Status
Married      9324.828181
Unmarried    9291.655739
Name: Purchase, dtype: float64
```

[49]:
```
married_income=df[df['Marital_Status']=='Married']['Purchase']
```

[51]:
```
unmarried_income=df[df['Marital_Status']=='Unmarried']['Purchase']
```

[54]:
```
married_sem = stats.sem(df[df['Marital_Status'] == 'Married']['Purchase'])
```

[55]:
```
unmarried_sem=stats.sem(df[df['Marital_Status']=='Unmarried']["Purchase"])
```

[56]:
```
df_married= len(df[df['Marital_Status']=='Married'])-1
```

[57]:
```
df_unmarried=len(df[df['Marital_Status']=='Unmarried'])-1
```

[58]:
```
alhpa=0.95
```

[59]:
```
confidence_interval_married=stats.t.
 ↪interval(alpha,df_married,loc=married_income.mean(),scale=married_sem)
```

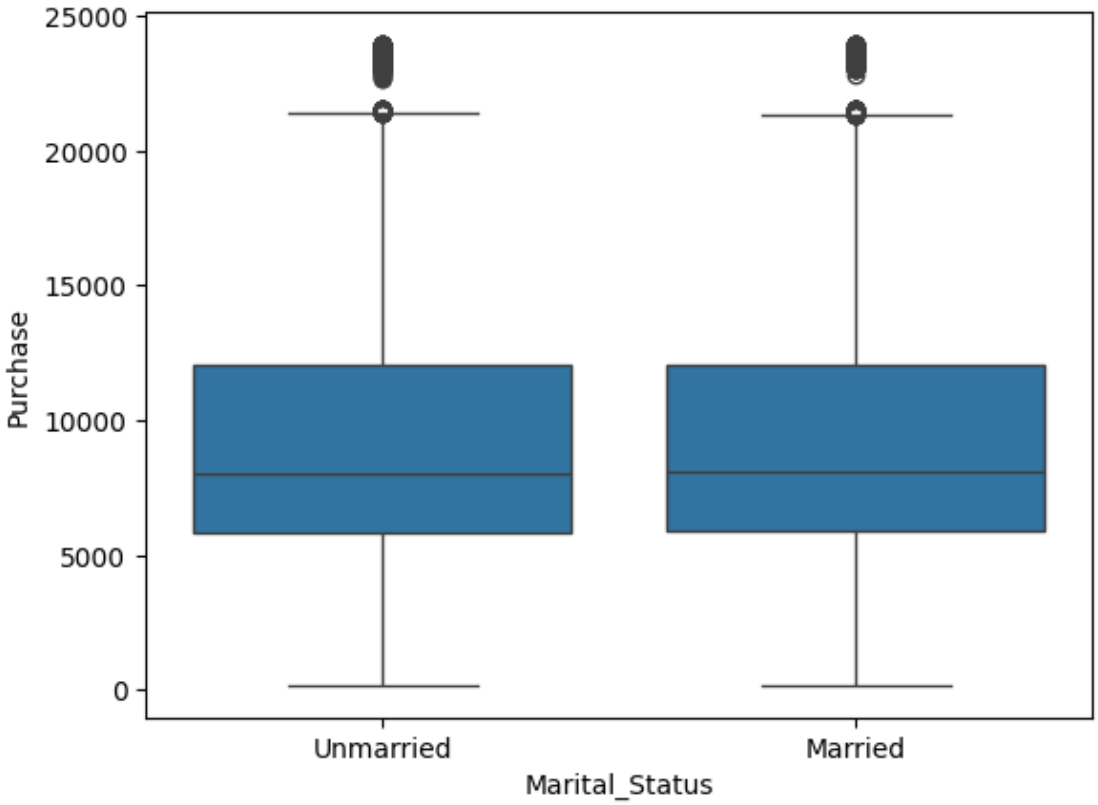
[60]:
```
confidence_interval_unmarried=stats.t.
 ↪interval(alpha,df_unmarried,loc=unmarried_income.mean(),scale=unmarried_sem)
```

[61]:
```
print(f'Confidence Interval for Married: {confidence_interval_married}')
print(f'Confidence Interval for Unmarried: {confidence_interval_unmarried}')
```

```
Confidence Interval for Married: (np.float64(9281.839654268762),
np.float64(9367.816707136182))
Confidence Interval for Unmarried: (np.float64(9255.708141892692),
np.float64(9327.60333617986))
```

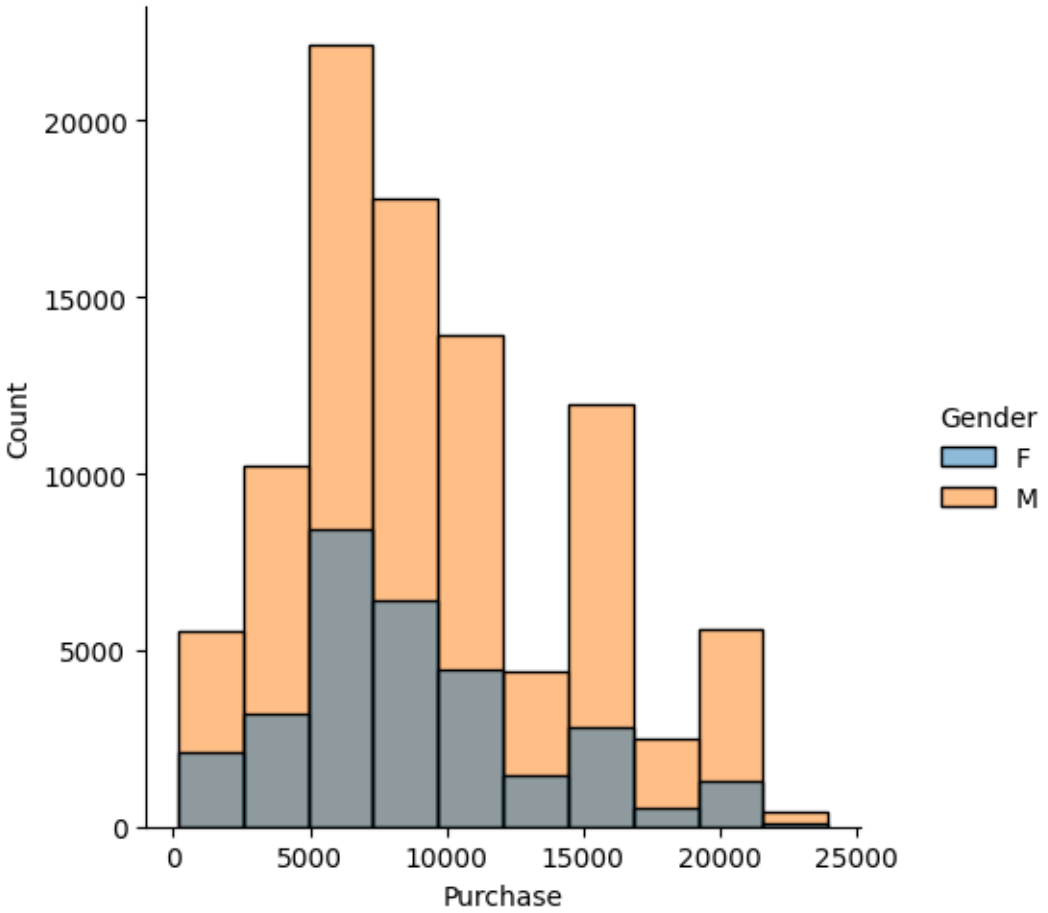
[62]:
```
sns.boxplot(x=df['Marital_Status'],y=df['Purchase'],data=df)
```

[62]:
```
<Axes: xlabel='Marital_Status', ylabel='Purchase'>
```



[70]:
```
sns.displot(data=data,hue="Gender",x="Purchase",bins=10)
```

[70]:
```
<seaborn.axisgrid.FacetGrid at 0x7d61b047cd90>
```



[71]:
```
data.groupby("Gender")["Purchase"].describe()
```

[71]:

Gender	count	mean	std	min	25%	50%	75%	max
F	30745.0	8776.171117	4692.525634	188.0	5450.00	7924.0	11403.00	23948.0
M	94468.0	9477.447919	5056.386848	185.0	5890.75	8101.0	12517.25	23961.0

Objective: The purpose of this analysis is to explore gender-based purchasing behavior among Walmart customers and to determine whether there are statistically significant differences in purchase amounts between male and female shoppers.

1. **Dataset Overview:** The dataset used in this study contains individual transaction records including customer demographic attributes such as Gender, Age, Marital Status, and Purchase Amount. Initial checks were conducted to confirm the quality and usability of the

dataset:

No missing values were found.

The dataset contains categorical and numerical variables suitable for both descriptive and inferential analysis.

2. **Gender-Based Purchase Summary:** We first segmented the data by the "Gender" attribute and calculated the following:

Mean Purchase (Female): ~8726.17

Mean Purchase (Male): ~9477.45

This indicates that male customers on average spent more than female customers.

3. **Statistical Precision: Standard Error of the Mean (SEM)**: The Standard Error of the Mean was calculated for both genders:

SEM (Female): ~26.66

SEM (Male): ~16.49

A smaller SEM indicates more precision around the mean purchase value. Males exhibited more consistency in their purchase behavior.

4. **Confidence Intervals (95%):** Using a t-distribution, 95% confidence intervals for the average purchase amount were computed:

Female: (8723.72, 8828.63)

Male: (9445.20, 9509.69)

Since the intervals do not overlap, we can conclude that there is a statistically significant difference in purchase amounts between male and female customers at the 95% confidence level.

5. **Implications for Business Strategy:**

Male customers are spending more per transaction and show higher consistency. Targeted promotions or loyalty programs could reinforce this group.

Female customers, while spending less, represent a sizable group. Campaigns such as discounts, bundled offers, or referral rewards could increase purchase size.

##Business Strategy Recommendations: **Male Customers**: Leverage their higher spending and consistency with targeted loyalty programs or premium product promotions to maintain engagement.

Female Customers: Increase their purchase amounts through tailored campaigns, such as discounts, bundled offers, or referral rewards, to tap into this sizable segment.

[ ]: