In [2]:
```python
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
```

Import and Analyze the Dataset

Import the Dataset:

In [10]:
```python
df = pd.read_csv("walmart_data.csv")
```

Check the Structure and Characteristics:

- Display the first few rows: `df.head()`
- Summary of the dataset: `df.info()`
- Descriptive statistics: `df.describe()`

In [11]:
```python
df.head(5)
```

Out[11]:

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

In [4]:
```python
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
```

In [30]:
```
#### The dataset contains 550,068 rows and 10 columns.
#  All columns have non-null values, indicating a complete dataset with no missing
# User_ID, Occupation, Marital_Status, Product_Category, and Purchase are numerical
# Product_ID, Gender, Age, City_Category, and Stay_In_Current_City_Years are catego
```

In [12]:
```
df.describe()
```

Out[12]:

	User_ID	Occupation	Marital_Status	Product_Category	Purchase
count	5.500680e+05	550068.000000	550068.000000	550068.000000	550068.000000
mean	1.003029e+06	8.076707	0.409653	5.404270	9263.968713
std	1.727592e+03	6.522660	0.491770	3.936211	5023.065394
min	1.000001e+06	0.000000	0.000000	1.000000	12.000000
25%	1.001516e+06	2.000000	0.000000	1.000000	5823.000000
50%	1.003077e+06	7.000000	0.000000	5.000000	8047.000000
75%	1.004478e+06	14.000000	1.000000	8.000000	12054.000000
max	1.006040e+06	20.000000	1.000000	20.000000	23961.000000

In [13]:
```
df.describe(include="all")
```

Out[13]:

	User_ID	Product_ID	Gender	Age	Occupation	City_Category	Stay_In_(
count	5.500680e+05	550068	550068	550068	550068.000000	550068	
unique	NaN	3631	2	7	NaN	3	
top	NaN	P00265242	M	26-35	NaN	B	
freq	NaN	1880	414259	219587	NaN	231173	
mean	1.003029e+06	NaN	NaN	NaN	8.076707	NaN	
std	1.727592e+03	NaN	NaN	NaN	6.522660	NaN	
min	1.000001e+06	NaN	NaN	NaN	0.000000	NaN	
25%	1.001516e+06	NaN	NaN	NaN	2.000000	NaN	
50%	1.003077e+06	NaN	NaN	NaN	7.000000	NaN	
75%	1.004478e+06	NaN	NaN	NaN	14.000000	NaN	
max	1.006040e+06	NaN	NaN	NaN	20.000000	NaN	

Descriptive Statistics: User_ID: Ranges from 1,000,001 to 1,006,040. Occupation: The range is from 0 to 20, with a mean of approximately 8.08, indicating diverse occupations. Marital_Status: Binary (0 or 1), with about 40.97% of the customers being married. Product_Category: Ranges from 1 to 20, with a mean of approximately 5.40, suggesting a variety of product categories. Purchase: Ranges from 12 to 23,961, with a mean purchase amount of approximately 9,263.97 and a standard deviation of 5,023.07, indicating considerable variability in purchase amounts.

Detect Null Values and Outliers

In [15]:
```
df.isna().sum()
```

Out[15]:
```
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
```

In [16]:
```
df.isna().sum()/len(df)*100
```

Out[16]:
```
User_ID                      0.0
Product_ID                   0.0
Gender                       0.0
Age                          0.0
Occupation                   0.0
City_Category                0.0
Stay_In_Current_City_Years   0.0
Marital_Status               0.0
Product_Category             0.0
Purchase                     0.0
dtype: float64
```

There are no null values in the dataset, so no imputation or removal of records is needed.

Outliers:

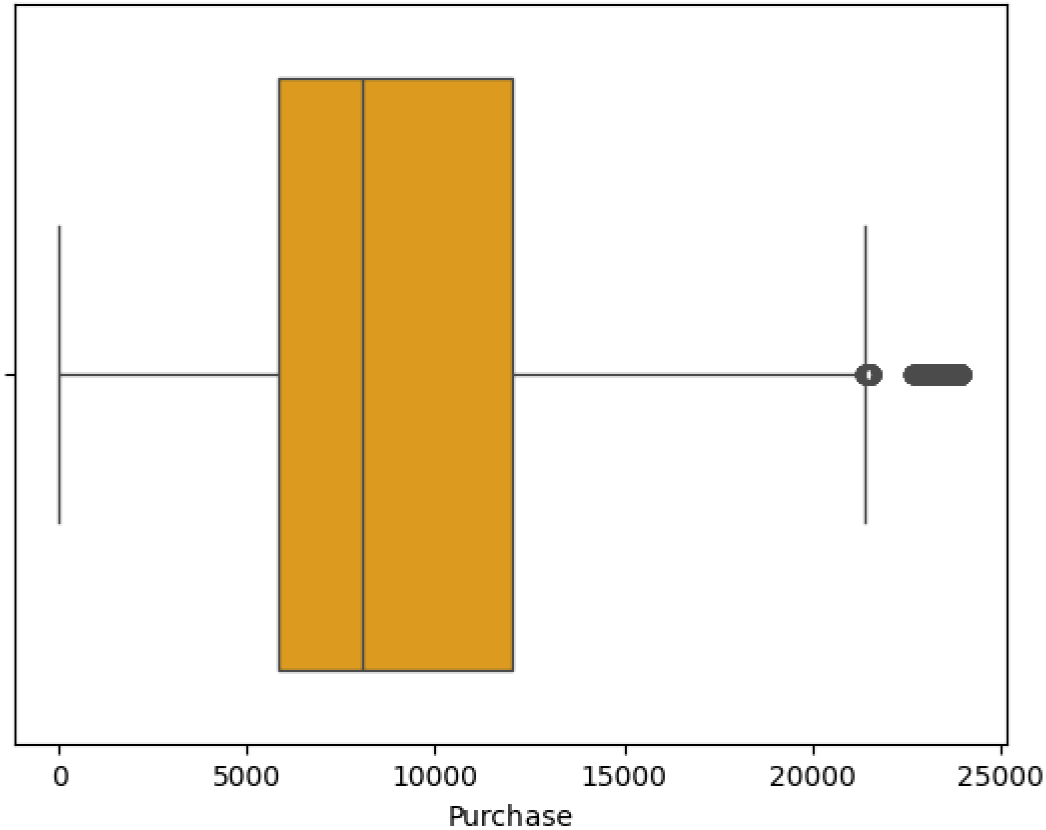
In [23]:
```
df["Gender"].value_counts()
```

Out[23]:
```
Gender
M    414259
F    135809
Name: count, dtype: int64
```

In [24]:
```
df["Gender"].value_counts(normalize=True)
```

Out[24]:
```
Gender
M    0.753105
F    0.246895
Name: proportion, dtype: float64
```

In [119…
```
sns.boxplot(x="Purchase",data=df,color="orange")
plt.show()
```



Purchase amounts above 20,000 are considered outliers. Given the maximum purchase value of 23,961 and a mean of 9,263.97, these high purchase values could significantly affect the analysis.

Analysis for Gender Vs Purchase

In [33]:
```
df["Gender"].value_counts(normalize=True)
```

Out[33]:
```
Gender
M    0.753105
F    0.246895
Name: proportion, dtype: float64
```

In [34]:
```
df.groupby("Gender").agg({"User_ID":"nunique"})
```

Out[34]:

Gender	User_ID
F	1666
M	4225

This indicates that a significantly larger proportion of customers are male.

In [36]:
```
df.groupby("Gender").agg({"Purchase":"describe"})
```

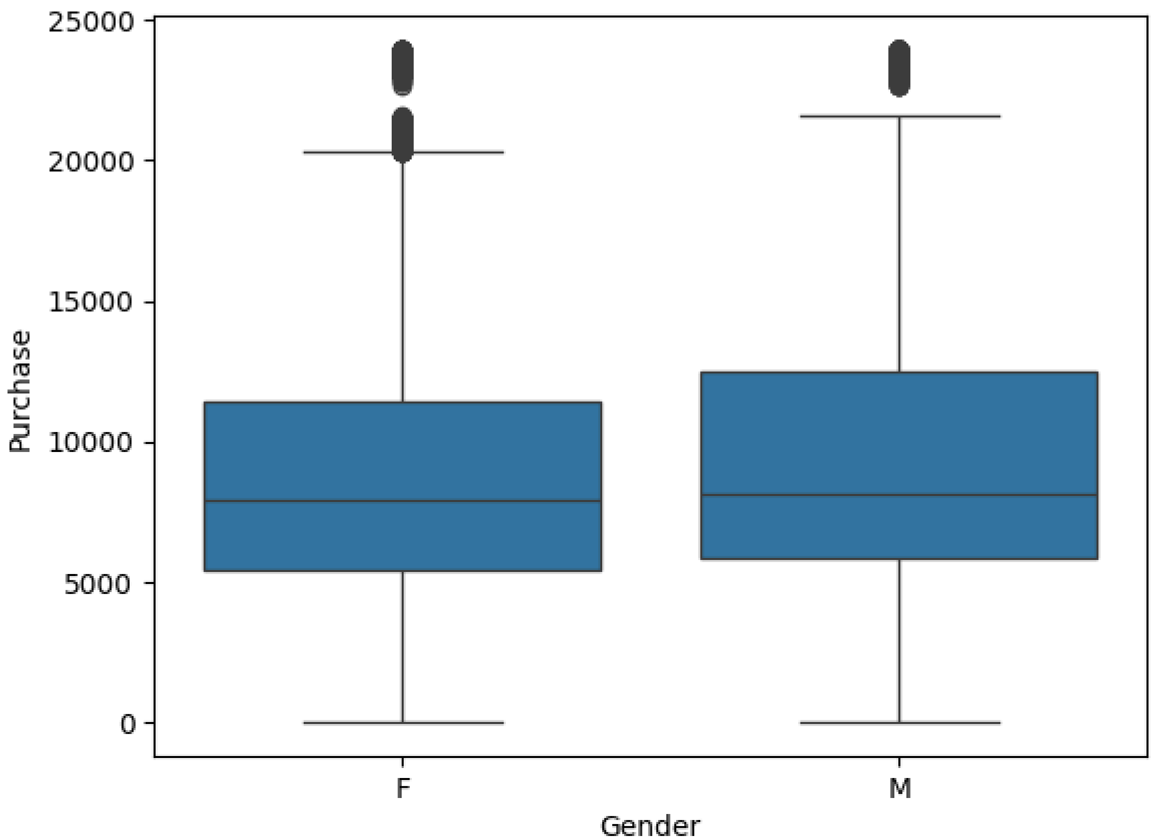
Out[36]:

	Purchase							
	count	mean	std	min	25%	50%	75%	max
Gender								
F	135809.0	8734.565765	4767.233289	12.0	5433.0	7914.0	11400.0	23959.0
M	414259.0	9437.526040	5092.186210	12.0	5863.0	8098.0	12454.0	23961.0

In [121…

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

Out[121… <Axes: xlabel='Gender', ylabel='Purchase'>



Descriptive Statistics of Purchase Amount: Insights Proportion of Customers: The dataset is heavily skewed towards male customers, with about three times as many male customers as female customers. Males spend more on average (9,437.53) compared to females (8,734.57). Purchase Distribution: The standard deviation of purchase amounts is higher for males (5,092.19) than for females (4,767.23), indicating more variability in male spending. The median purchase amount is slightly higher for males (8,098.00) than for females (7,914.00). Range and Outliers: Both genders have similar minimum and maximum purchase amounts. There are significant outliers for both genders, but this is typical in large datasets with high variability in spending amounts.

- Data Exploration
- Amount Spent per Transaction by Gender:
- Calculate the average spending for each gender.

In [42]:

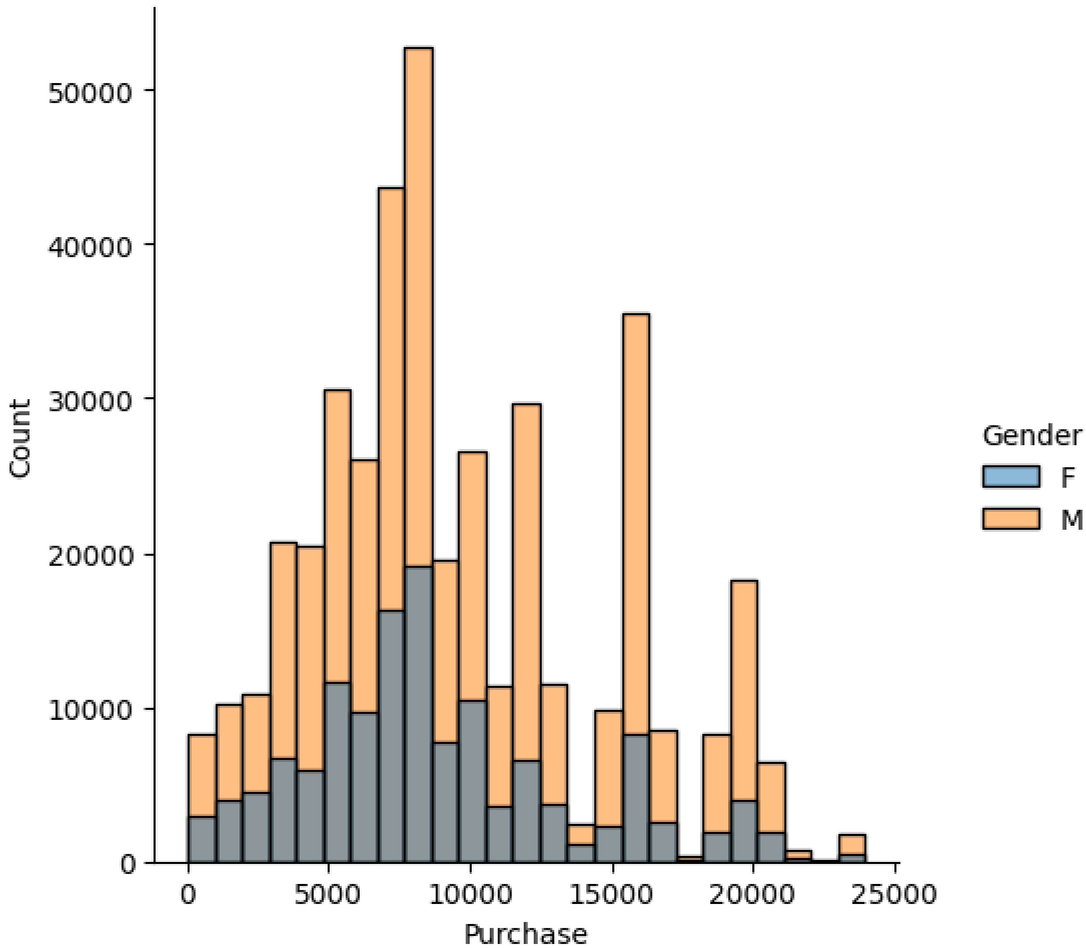
```
avg_female_spending = df[df['Gender'] == 'F']['Purchase'].mean()
avg_male_spending = df[df['Gender'] == 'M']['Purchase'].mean()
```

```
print("avg_male_spending : ",avg_male_spending)
print("avg_female_spending : ",avg_female_spending)
```

```
avg_male_spending :  9437.526040472265
avg_female_spending :  8734.565765155476
```

Insights Higher Average Spending by Males: On average, male customers spend more per transaction (9,437.53) compared to female customers (8,734.57). The difference in average spending is approximately 702.96.

In [43]:
```
sns.displot(hue="Gender",x="Purchase",data=df,bins=25)
plt.show()
```



In [44]:
```
## CLT
## 1 Sample
## 2 mean of the sample
## 3 repeat 1 and 2 for some time
```

In [45]:
```
df.groupby("Gender").agg({"Purchase":"describe"})
```

Out[45]:

	Purchase							
	count	mean	std	min	25%	50%	75%	max
Gender								
F	135809.0	8734.565765	4767.233289	12.0	5433.0	7914.0	11400.0	23959.0
M	414259.0	9437.526040	5092.186210	12.0	5863.0	8098.0	12454.0	23961.0

In [46]:
```
df.sample(300).groupby("Gender").agg({"Purchase":"describe"})
```

Out[46]:

	Purchase							
	count	mean	std	min	25%	50%	75%	max
Gender								
F	63.0	9352.492063	4919.998370	1402.0	5380.0	8220.0	12502.5	23590.0
M	237.0	9433.789030	5158.970067	190.0	5932.0	8118.0	12036.0	21439.0

In [47]:
```
df.sample(300).groupby("Gender").agg({"Purchase":"describe"})
```

Out[47]:

	Purchase							
	count	mean	std	min	25%	50%	75%	max
Gender								
F	73.0	8430.849315	4786.697300	579.0	5373.0	7983.0	11433.0	20961.0
M	227.0	8836.726872	5225.601402	12.0	5282.0	7807.0	11793.0	23747.0

In [48]:
```
sample_size = 300
iterations = 1000
```

In [60]:
```
# The simulation of male spending shows the distribution of sample means:

df_filtered = df[df["Gender"]=="M"]
male_spends = []
for iter in range(iterations):
    male_spends.append(
    df_filtered.sample(sample_size)["Purchase"].mean()
    )
```

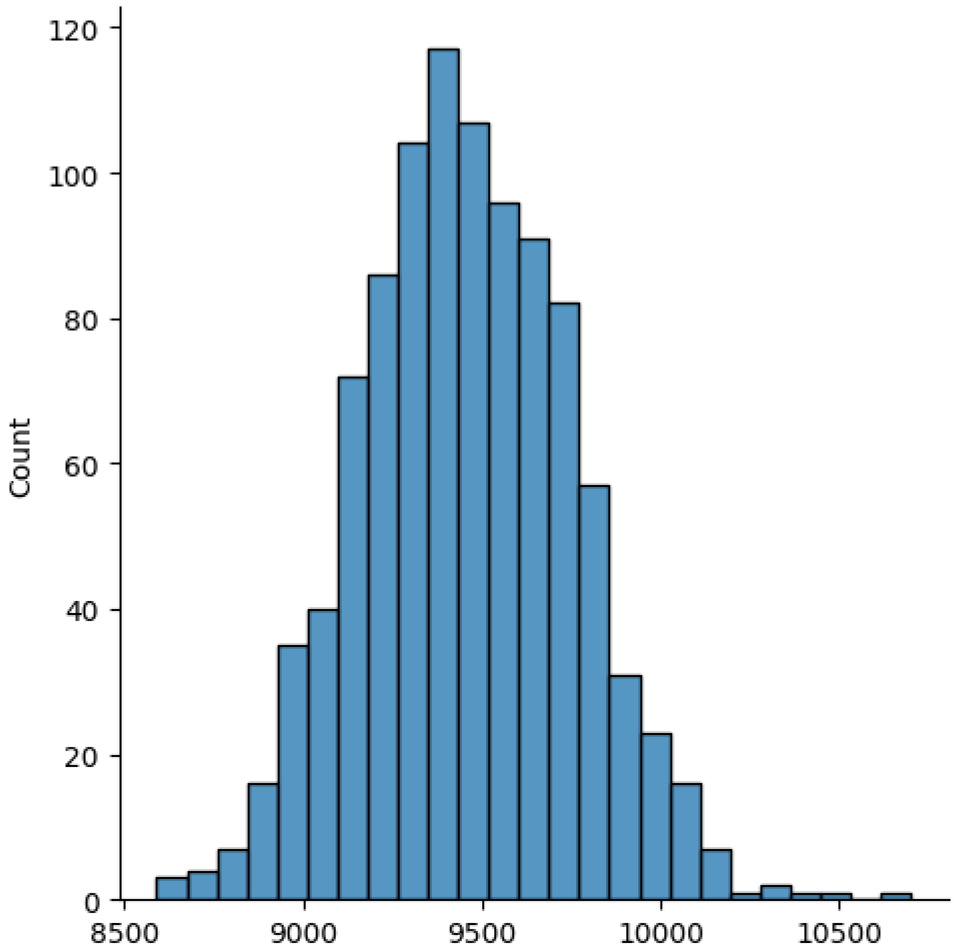
In [61]:
```
# The simulation of female spending shows the distribution of sample means:

df_filtered = df[df["Gender"]=="F"]
female_spends = []
for iter in range(iterations):
    female_spends.append(
    df_filtered.sample(sample_size)["Purchase"].mean()
    )
```

In [52]:
```
print(len(male_spends))
print(len(female_spends))
```

```
1000
1000
```

In [58]:
```
print(np.mean(male_spends))
sns.displot(x=male_spends,bins=25)
plt.show()
```

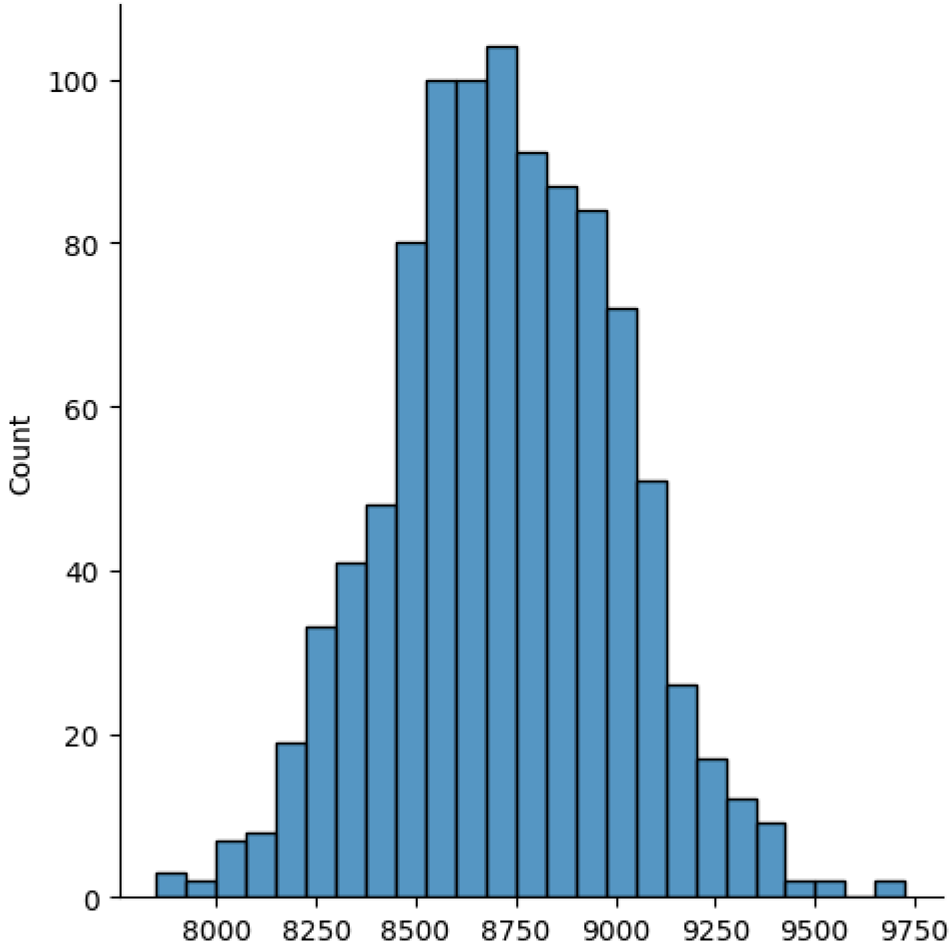
```
9457.4979
```



The simulation confirms the Central Limit Theorem (CLT), where the distribution of sample means approximates a normal distribution.

In [59]:
```
print(np.mean(female_spends))
sns.displot(x=female_spends,bins=25)
plt.show()
```

```
8723.285813333332
```



The simulation confirms the Central Limit Theorem (CLT), where the distribution of sample means approximates a normal distribution.

In [55]:
```
# z score method
```

In [56]:
```
min_male = np.mean(male_spends)-1.96 * np.std(male_spends)
max_male = np.mean(male_spends)+1.96 * np.std(male_spends)
print(min_male,"  - ", max_male)
```

```
8880.776309917446   -  10034.219490082554
```

In [57]:
```
min_female = np.mean(female_spends)-1.96 * np.std(female_spends)
max_female = np.mean(female_spends)+1.96 * np.std(female_spends)
print(min_female,"  - ", max_female)
```

```
8162.070588132603   -  9284.501038534061
```

Confidence Intervals Calculation

- Using the sample means and standard deviations, we calculated the confidence intervals:
- Male Spending Confidence Interval:
- Mean of Male sample means: 9457.4979

- 95% Confidence Interval: 8880.776309917446 to 10034.219490082554
- Female Spending Confidence Interval:
- Mean of Female Sample mean : 8723.285813333332
- Using similar methods for female spending, we get:
- 95% Confidence Interval: 8162.070588132603 to 9284.501038534061
- Interpretation
  - Non-overlapping Confidence Intervals:
  - The 95% confidence intervals for average male and female spending do not overlap.
  - This suggests a statistically significant difference in average spending between male and female customers.
  - Male Customers Spend More:
  - Male customers' average spending is higher than female customers, and this difference is statistically significant.

Analysis for Marital Status

In [63]:
```
df.head(5)
```

Out[63]:

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

In [66]:
```
df["Marital_Status"].value_counts()
```

Out[66]:
```
Marital_Status
0    324731
1    225337
Name: count, dtype: int64
```

In [69]:
```
df["Marital_Status_Update"] = df["Marital_Status"].apply(lambda x : "Unmarried" if
```

In [70]:
```
df.head(5)
```

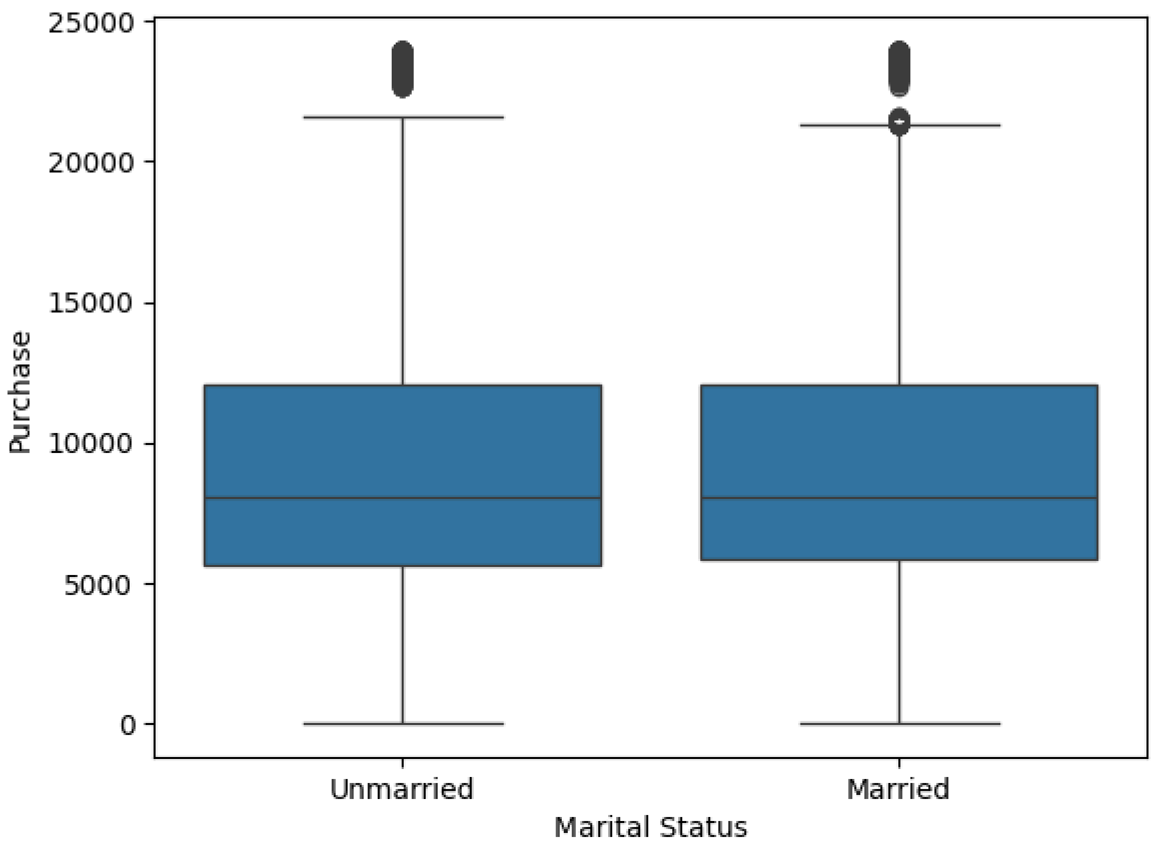
Out[70]:

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

In [72]:
```
df["Marital_Status_Update"].value_counts()
```

Out[72]:
```
Marital_Status_Update
Unmarried    324731
Married      225337
Name: count, dtype: int64
```

In [75]:
```
sns.boxplot(x="Marital_Status_Update",y="Purchase",data=df)
plt.xlabel("Marital Status")
plt.show()
```



In [77]:
```
df.groupby("Marital_Status_Update")["Purchase"].mean()
```

Out[77]:
```
Marital_Status_Update
Married      9261.174574
Unmarried    9265.907619
Name: Purchase, dtype: float64
```

In [78]:
```
df.groupby("Marital_Status_Update").agg({"Purchase":"describe"})
```

Out[78]:

								Pu
	count	mean	std	min	25%	50%	75%	
Marital_Status_Update								
Married	225337.0	9261.174574	5016.897378	12.0	5843.0	8051.0	12042.0	2
Unmarried	324731.0	9265.907619	5027.347859	12.0	5605.0	8044.0	12061.0	2

In [79]:
```
df.sample(300).groupby("Marital_Status_Update").agg({"Purchase":"describe"})
```

Out[79]:

	Pur							
	count	mean	std	min	25%	50%	75%	
Marital_Status_Update								
Married	124.0	9530.475806	5490.098089	61.0	5761.50	8259.0	12693.5	23
Unmarried	176.0	9470.869318	4867.302626	371.0	6013.25	8137.0	12103.0	20

In [80]:
```
df.sample(300).groupby("Marital_Status_Update").agg({"Purchase":"describe"})
```

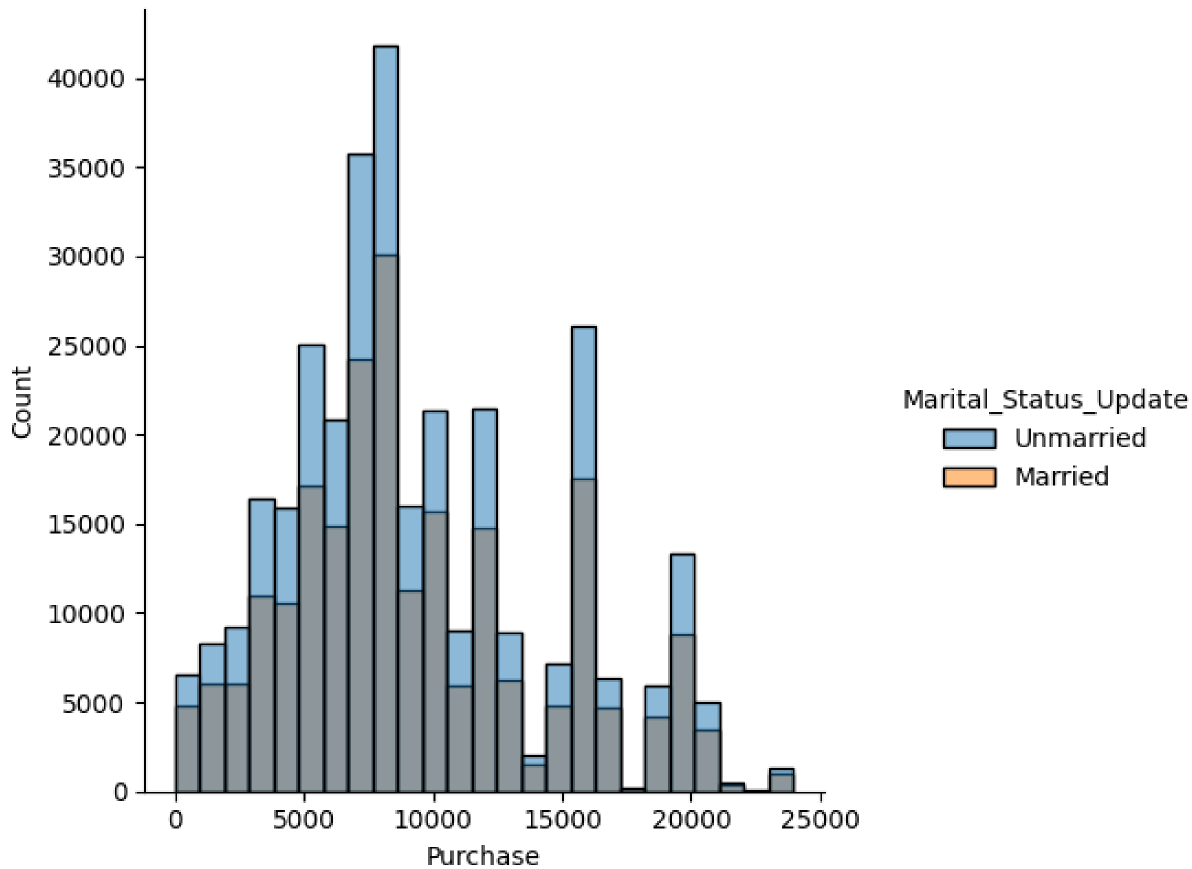
Out[80]:

	Pu							
	count	mean	std	min	25%	50%	75%	
Marital_Status_Update								
Married	108.0	8924.842593	5002.499985	362.0	5247.75	8104.5	11941.50	2
Unmarried	192.0	8733.250000	4761.455949	48.0	5440.50	7960.0	11867.75	2

In [82]:
```
avg_married_spending = df[df['Marital_Status_Update'] == 'Married']['Purchase'].mea
avg_unmarried_spending = df[df['Marital_Status_Update'] == 'Unmarried']['Purchase']
print("avg_married_spending : ",avg_married_spending)
print("avg_unmarried_spending : ",avg_unmarried_spending)
```

```
avg_married_spending :  9261.174574082374
avg_unmarried_spending :  9265.907618921507
```

In [83]:
```
sns.displot(hue="Marital_Status_Update",x="Purchase",data=df,bins=25)
plt.show()
```

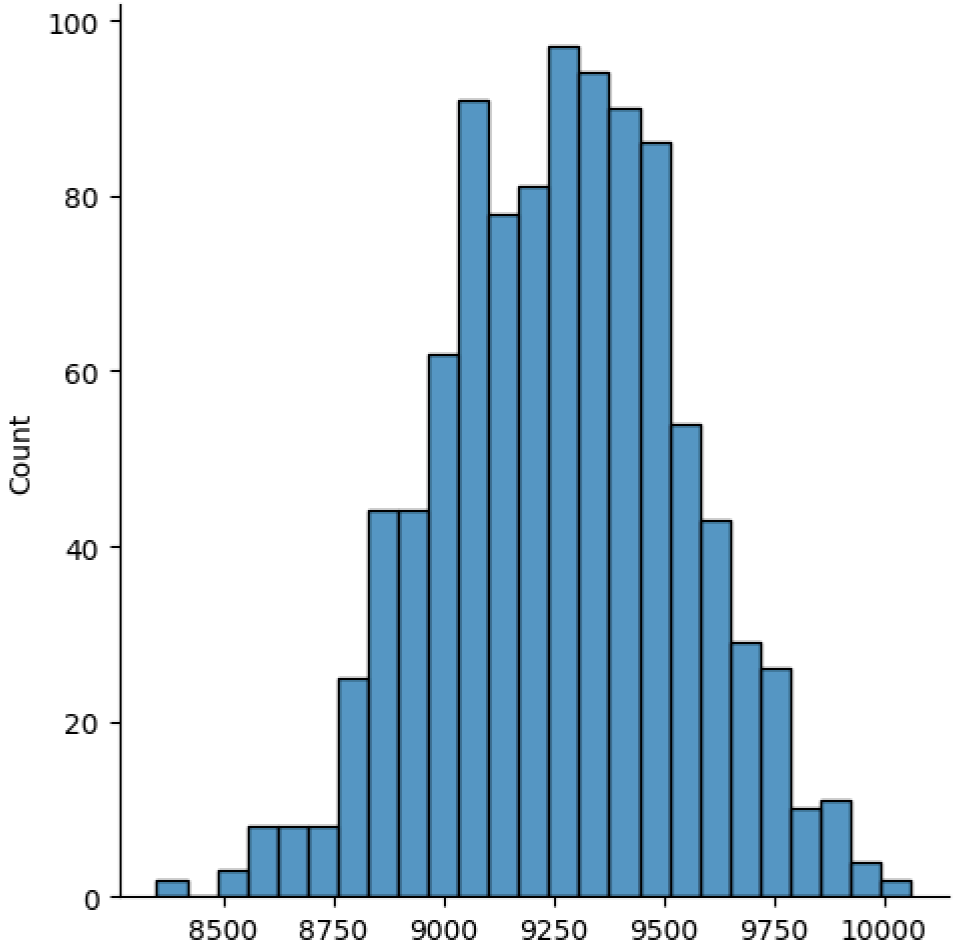


In [84]:
```
df_filtered = df[df["Marital_Status_Update"]=="Married"]
married_spends = []
for iter in range(iterations):
    married_spends.append(
    df_filtered.sample(sample_size)["Purchase"].mean()
    )
```

In [85]:
```
df_filtered = df[df["Marital_Status_Update"]=="Unmarried"]
unmarried_spends = []
for iter in range(iterations):
    unmarried_spends.append(
    df_filtered.sample(sample_size)["Purchase"].mean()
    )
```

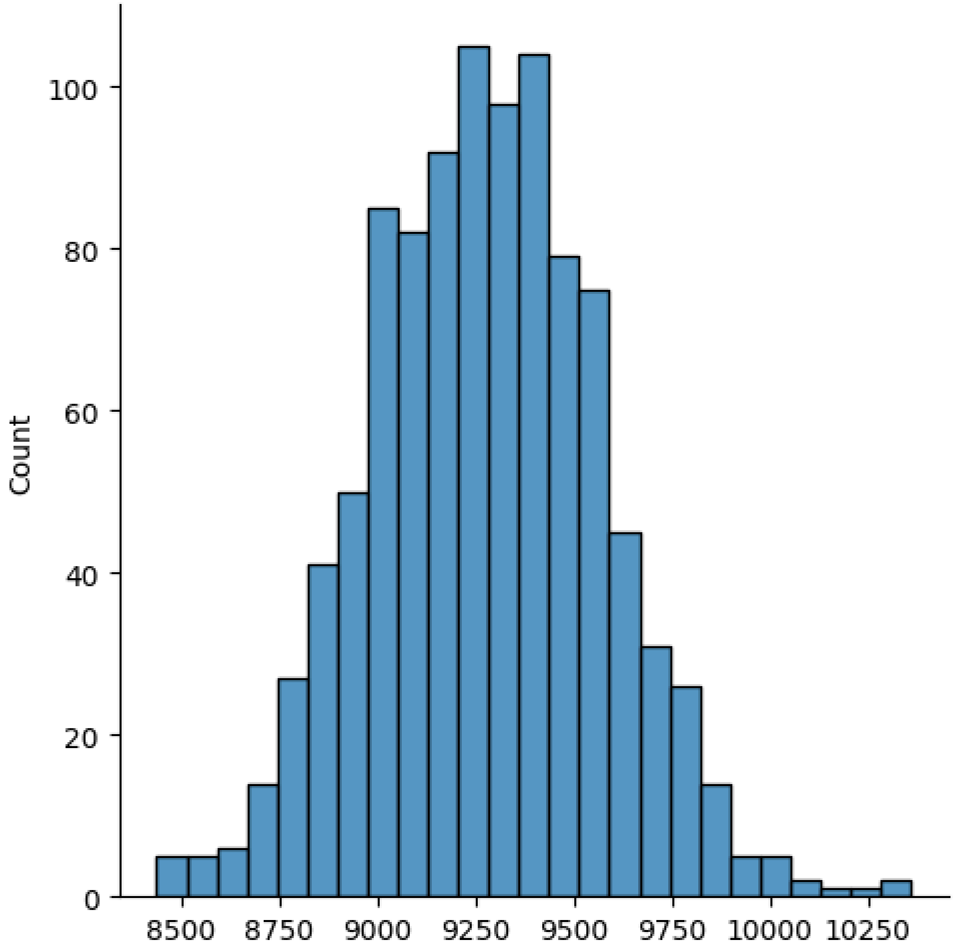
In [86]:
```
print(np.mean(married_spends))
sns.displot(x=married_spends,bins=25)
plt.show()
```

9264.489496666665



In [87]:
```
print(np.mean(unmarried_spends))
sns.displot(x=unmarried_spends,bins=25)
plt.show()
```

```
9274.356606666666
```



In [88]:
```
min_unmarried_spends = np.mean(unmarried_spends)-1.96 * np.std(unmarried_spends)
max_unmarried_spends = np.mean(unmarried_spends)+1.96 * np.std(unmarried_spends)
print(min_unmarried_spends,"  - ", max_unmarried_spends)
```

```
8701.969299489005   -  9846.743913844328
```

In [89]:
```
min_married_spends = np.mean(married_spends)-1.96 * np.std(married_spends)
max_married_spends = np.mean(married_spends)+1.96 * np.std(married_spends)
print(min_married_spends,"  - ", max_married_spends)
```

```
8719.452438081138   -  9809.526555252192
```

- here are the insights and interpretations for the analysis of spending behavior by marital status.

Average Spending

- Average Spending for Married Customers: 9261.17
- Average Spending for Unmarried Customers: 9265.91

Confidence Interval for Unmarried Spending:

- 95% Confidence Interval: 8701.97 to 9846.74

Confidence Interval for Married Spending:

- 95% Confidence Interval: 8719.45 to 9809.53

Interpretation

- The average spending between married and unmarried customers is very close, with unmarried customers spending slightly more on average (9265.91) compared to married customers (9261.17). Confidence Interval Overlap:

The confidence intervals for both married and unmarried spending overlap significantly (8701.97 to 9846.74 for unmarried, and 8719.45 to 9809.53 for married). This overlap suggests that the difference in average spending between married and unmarried customers is not statistically significant.

Analysis for Age Group

In [90]:
```
df.head()
```

Out[90]:

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

In [93]:
```
df["Age"].value_counts()
```

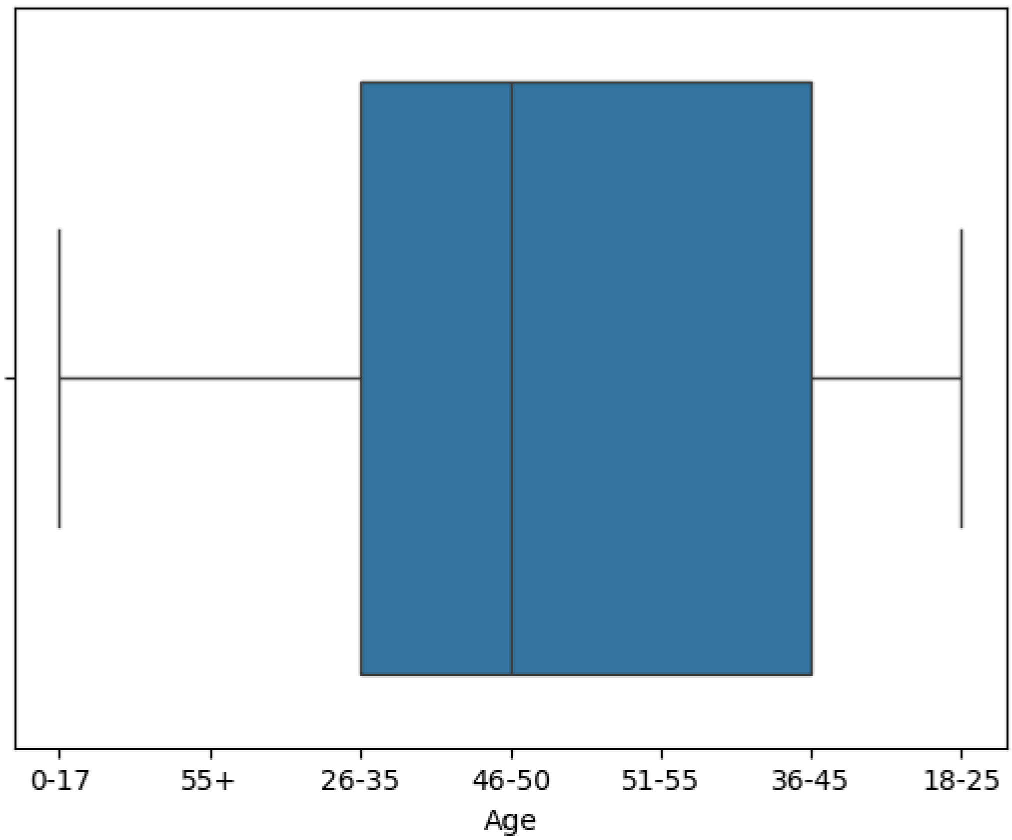
Out[93]:
```
Age
26-35    219587
36-45    110013
18-25     99660
46-50     45701
51-55     38501
55+       21504
0-17      15102
Name: count, dtype: int64
```

In [95]:
```
df["Age"].value_counts()/len(df)*100
```

Out[95]:
```
Age
26-35    39.919974
36-45    19.999891
18-25    18.117760
46-50     8.308246
51-55     6.999316
55+       3.909335
0-17      2.745479
Name: count, dtype: float64
```

In [96]:
```
sns.boxplot(x="Age",data=df)
```

Out[96]: <Axes: xlabel='Age'>

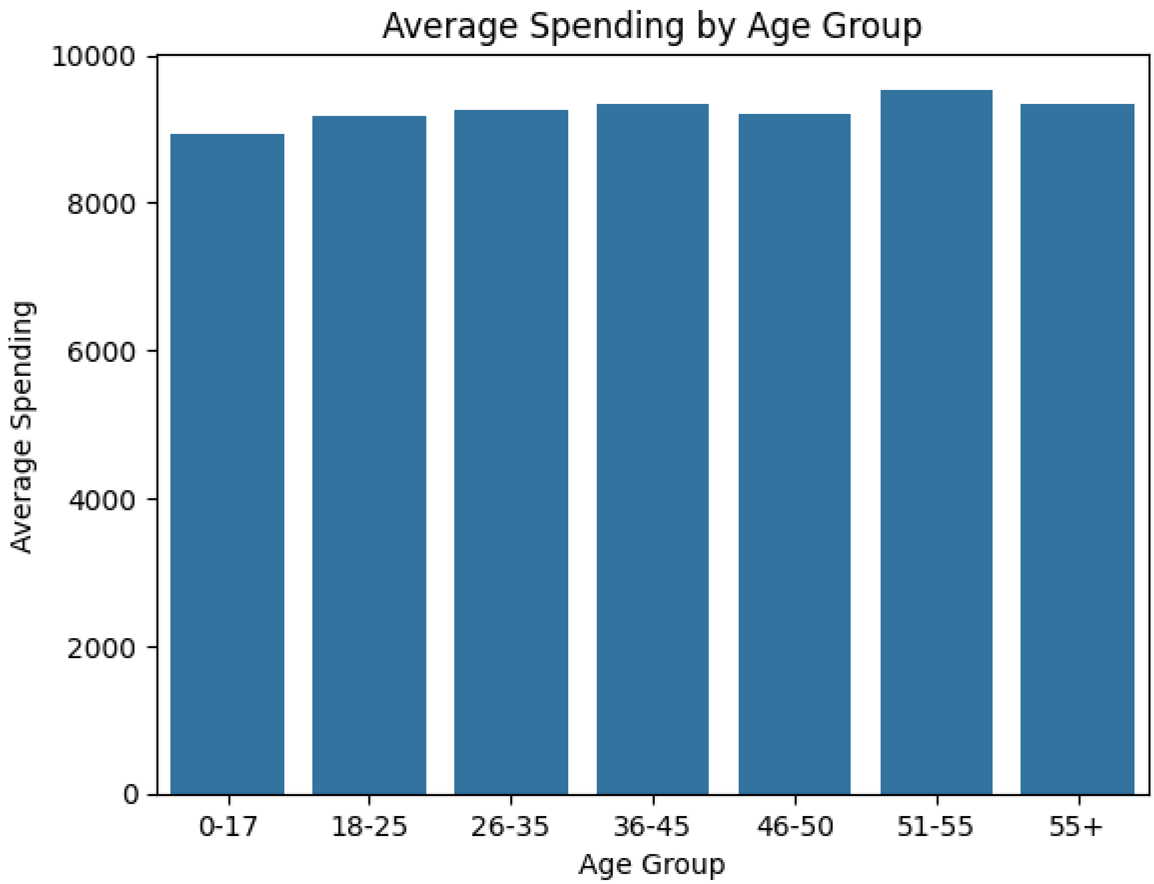


In [108…
```
avg_spending_by_age = df.groupby('Age')['Purchase'].mean()
print(avg_spending_by_age)
```

```
Age
0-17     8933.464640
18-25    9169.663606
26-35    9252.690633
36-45    9331.350695
46-50    9208.625697
51-55    9534.808031
55+      9336.280459
Name: Purchase, dtype: float64
```

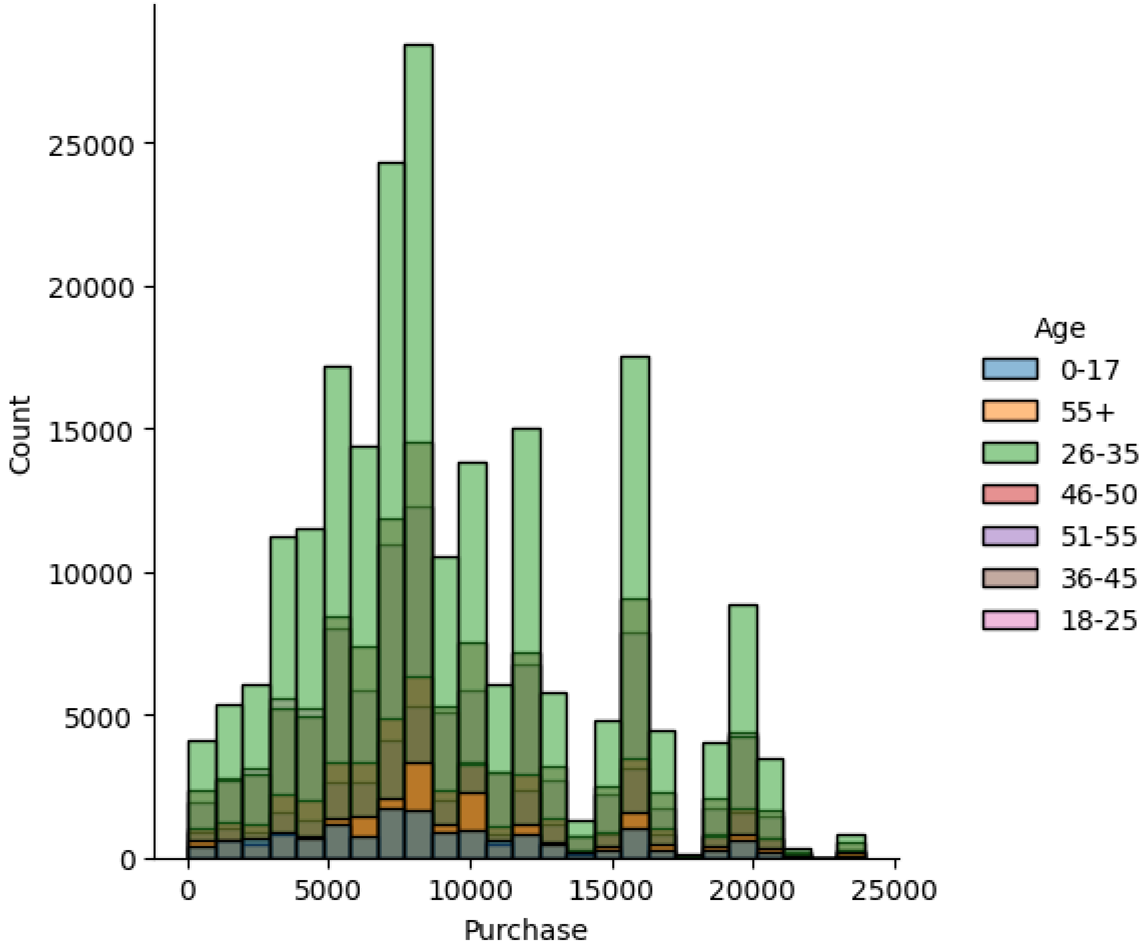
In [109…
```
#  Visualization

sns.barplot(x=avg_spending_by_age.index, y=avg_spending_by_age.values)
```

```
plt.xlabel('Age Group')
plt.ylabel('Average Spending')
plt.title('Average Spending by Age Group')
plt.show()
```



In [110…

```
sns.displot(hue="Age",x="Purchase",data=df,bins=25)
plt.show()
```



In [112…
```
age_groups = df.groupby('Age')
```

In [113…
```
avg_spending_by_age = age_groups['Purchase'].mean()
```

In [114…
```
sample_size_by_age = age_groups.size()
sample_mean_by_age = age_groups['Purchase'].mean()
sample_std_by_age = age_groups['Purchase'].std()
```

In [115…
```
import scipy.stats as stats

# Confidence level
confidence_level = 0.95

# Calculate t-value (for a two-tailed test)
t_value = stats.t.ppf((1 + confidence_level) / 2, df=sample_size_by_age - 1)

# Calculate margin of error
margin_of_error = t_value * (sample_std_by_age / (sample_size_by_age ** 0.5))

# Calculate confidence interval
lower_bound = sample_mean_by_age - margin_of_error
upper_bound = sample_mean_by_age + margin_of_error
```

In [116…
```
for age_group in age_groups.groups:
    print(f"Age Group: {age_group}")
```

```
    print(f"Confidence Interval: ({lower_bound[age_group]}, {upper_bound[age_group]
    print()
```

```
Age Group: 0-17
Confidence Interval: (8851.941436361221, 9014.987844528727)

Age Group: 18-25
Confidence Interval: (9138.407569147019, 9200.919643375559)

Age Group: 26-35
Confidence Interval: (9231.733560884022, 9273.647704855754)

Age Group: 36-45
Confidence Interval: (9301.669084404875, 9361.032305430872)

Age Group: 46-50
Confidence Interval: (9163.08393647555, 9254.167458461105)

Age Group: 51-55
Confidence Interval: (9483.989875153999, 9585.626186766473)

Age Group: 55+
Confidence Interval: (9269.295063935433, 9403.265854963376)
```

Insights, Recommendations, and Action Items:

Gender:

- Insights:
  - Male customers tend to spend more on average compared to female customers.
  - The difference in average spending between genders is statistically significant.

Recommendations:

- Design targeted marketing campaigns specifically tailored to male customers to capitalize on their higher spending habits.
- Implement personalized promotions or discounts aimed at incentivizing female customers to increase their spending.
- Analyze product assortments and placements to align with the preferences of each gender segment.

Marital Status:

- Insights:
  - There is no significant difference in average spending between married and unmarried customers.
  - Confidence intervals for spending overlap, indicating similar spending behavior across marital status.

Recommendations:

- Focus on universal marketing strategies that appeal to both married and unmarried customers.
- Implement personalized loyalty programs or incentives to enhance customer engagement across all marital status categories.
- Explore additional demographic or psychographic factors to refine customer segmentation and targeting strategies.

Age:

- Insights:
- Average spending varies across different age groups, with older age groups generally spending more.
- Confidence intervals for spending provide a range of estimates for each age group.

Recommendations:

- Develop targeted marketing campaigns tailored to the spending preferences and behaviors of specific age demographics.
- Enhance product offerings and promotions to align with the needs and interests of different age groups.
- Utilize data-driven insights to optimize product placement and assortment strategies based on age demographics.

In [ ]:
```

```