Business Problem

The Management team at Walmart Inc. wants to **analyze the customer purchase behavior** (specifically, purchase amount) against the customer's gender and the various other factors to help the business make better decisions.

They want to understand if the **spending habits differ between male and female customers**: Do women spend more on Black Friday than men?

Required Libraries

In [1]:
```
import pandas as pd
import numpy as np

# For visualizing the data -
import matplotlib.pyplot as plt
import seaborn as sns

# For statistical functions -
import scipy.stats as statsort
import warnings
warnings.filterwarnings('ignore')
```

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

load dataset

In [2]:
```
url = "https://d2beiqkhq929f0.cloudfront.net/public_assets/assets/000/001/293/original/walmar
df = pd.read_csv(url)
```

first 5 top rows of the dataset

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

Out[ ]:

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

last 5 rows of the dataset

In [ ]:
```
df.tail()
```

Out[ ]:

	User_ID	Product_ID	Gender	Age	Occupation	City_Category	Stay_In_Current_City_Years	Ma
550063	1006033	P00372445	M	51-55	13	B	1	
550064	1006035	P00375436	F	26-35	1	C	3	
550065	1006036	P00375436	F	26-35	15	B	4+	
550066	1006038	P00375436	F	55+	1	C	2	
550067	1006039	P00371644	F	46-50	0	B	4+	

Basic EDA

In [ ]:
```
print('Number of rows: ',df.shape[0])
print('Number of coulms: ',df.shape[1])
```

```
Number of rows:  550068
Number of coulms:  10
```

In [ ]:
```
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 [ ]:
```
#  descriptive statistics of the numerical columns
df.describe().T
```

Out[ ]:

	count	mean	std	min	25%	50%	75%	
User_ID	550068.0	1.003029e+06	1727.591586	1000001.0	1001516.0	1003077.0	1004478.0	1(
Occupation	550068.0	8.076707e+00	6.522660	0.0	2.0	7.0	14.0	
Marital_Status	550068.0	4.096530e-01	0.491770	0.0	0.0	0.0	1.0	
Product_Category	550068.0	5.404270e+00	3.936211	1.0	1.0	5.0	8.0	
Purchase	550068.0	9.263969e+03	5023.065394	12.0	5823.0	8047.0	12054.0	

Occupation:

Occupation values range from 0 to 20, representing 21 unique occupation categories.

Marital_Status

The feature is binary (0 = single, 1 = married).

The mean value is 0.41, meaning 41% of the customers are married and 59% are single.

The median is 0, confirming that singles form the majority of customers.

This insight could be valuable — for example:

Singles are a larger customer base.

Married customers (though fewer) might show different spending patterns worth analyzing.

Purchase

The purchase amount ranges from $12 to$23,961.

The mean purchase is $9,263, while the median is$8,047.

Since the mean is greater than the median, the distribution is right-skewed, i.e., a few very high purchases pull the average upward.

25% of purchases are below $5,823, 50$8,047, and 75% below $12,054 \rightarrow the bulk of purchases lie in the$ 5,000– $12,000 range.

The high maximum ($23,961) compared to the 75th percentile suggests the presence of outliers (very high-value purchases).

Business implication:

Majority of purchases are mid-range ($5k–$12k).

A small number of customers contribute to high-value purchases, making them potential premium/loyalty segment customers worth targeting.

In [ ]:
```
#  descriptive statistics of the categorical columns
df.describe(include=['O'])
```

Out[ ]:

	Product_ID	Gender	Age	City_Category	Stay_In_Current_City_Years
count	550068	550068	550068	550068	550068
unique	3631	2	7	3	5
top	P00265242	M	26-35	B	1
freq	1880	414259	219587	231173	193821

Product_ID

The most frequently purchased product is Product ID: P00265242, bought 1,880 times.

The fact that one product has such a high purchase frequency suggests product popularity concentration — a few products might account for a large share of sales.

Gender

Dataset contains 550,068 entries, with 2 gender categories (M/F).

Most frequent gender: Male (M) with 414,259 entries (approx. 75%), while females make up only about 135,809 (approx. 25%).

This shows a strong male dominance in customer base.

Age

There are 7 unique age groups in the dataset.

The most frequent group is 26–35 years, with 219,587 customers (~40%).

This shows that young adults (26–35) are the largest shopping segment, likely forming the backbone of sales.

City_Category

There are 3 unique city categories (A, B, C).

The most frequent category is B, with 231,173 customers (~42%).

This means nearly half of the customers come from Category B cities (tier-2 urban cities).

Stay_In_Current_City_Years

There are 5 unique values: 0, 1, 2, 3, 4+ (years of stay).

The most common is 1 year, with 193,821 customers (~35%).

This suggests that a large portion of the customers are relatively new residents in their cities.

checking duplicate values

In [ ]:
```
df.duplicated().sum()
```

Out[ ]:
```
np.int64(0)
```

no duplicate rows in dataset

checking unique values in age, gender, city_category, stay_in_current_year categorical columns

In [ ]:
```
df['Gender'].unique()
```

Out[ ]:
```
array(['F', 'M'], dtype=object)
```

In [ ]:
```
df['Age'].unique()
```

Out[ ]:
```
array(['0-17', '55+', '26-35', '46-50', '51-55', '36-45', '18-25'],
      dtype=object)
```

In [ ]:
```
df['City_Category'].unique()
```

Out[ ]:
```
array(['A', 'C', 'B'], dtype=object)
```

In [ ]:
```
df['Stay_In_Current_City_Years'].unique()
```

Out[ ]:
```
array(['2', '4+', '3', '1', '0'], dtype=object)
```

2. Detect Null values and outliers

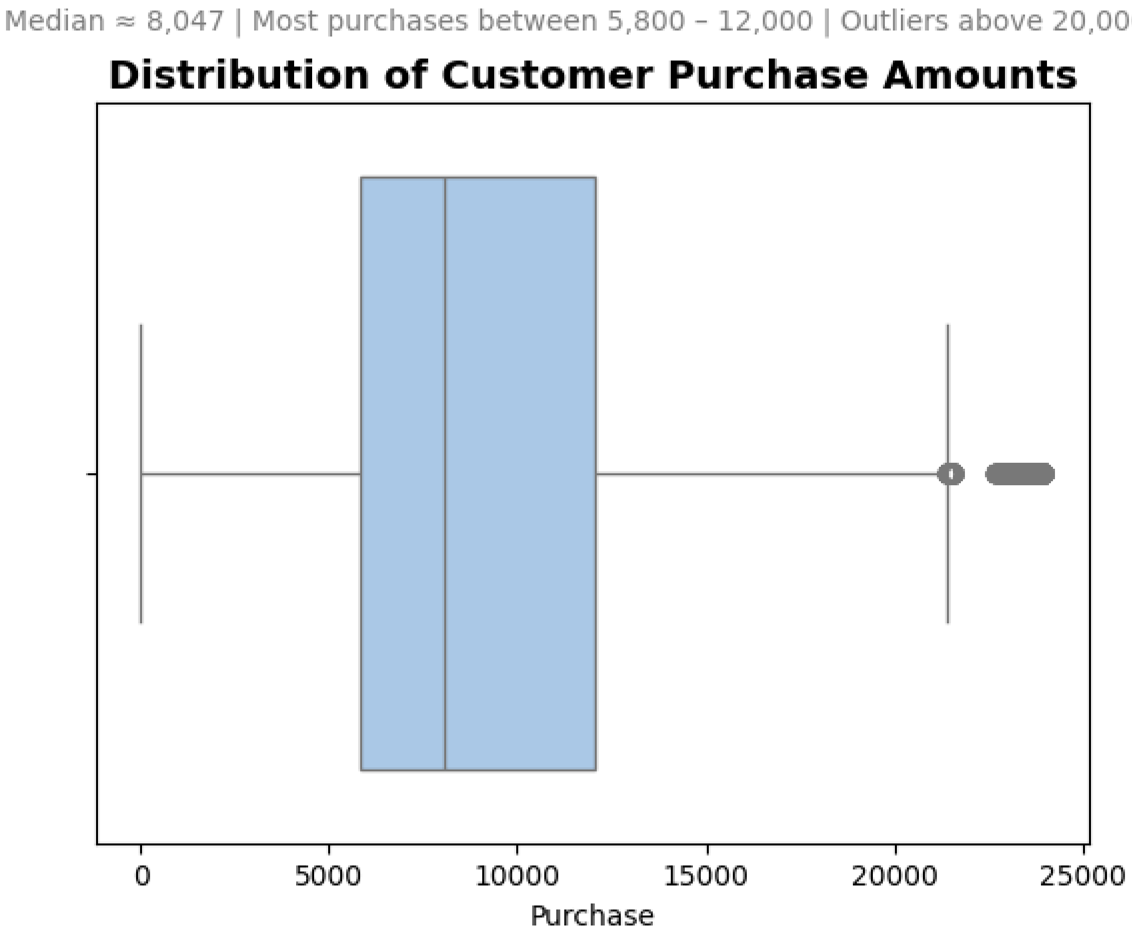
In [ ]:
```
#  null values
df.isna().sum().sum()
```

Out[ ]: `np.int64(0)`

no null values in dataset

checking for outliers

In [ ]:
```
sns.boxplot(x = df['Purchase'],  palette="pastel")
plt.title("Distribution of Customer Purchase Amounts", fontsize=14, fontweight="bold")
plt.suptitle("Median ≈ 8,047 | Most purchases between 5,800 – 12,000 | Outliers above 20,000",
             fontsize=10, color="gray")
plt.show()
```



Checking value counts for categorical columns

In [ ]:
```
df['Gender'].value_counts()
```

Out[ ]:

Gender	count
M	414259
F	135809

dtype: int64

In [ ]:
```
df['Age'].value_counts()
```

Out[ ]:

Age	count
26-35	219587
36-45	110013
18-25	99660
46-50	45701
51-55	38501
55+	21504
0-17	15102

dtype: int64

In [ ]:
```
df['Occupation'].value_counts()[:5]
```

Out[ ]:

Occupation	count
4	72308
0	69638
7	59133
1	47426
17	40043

dtype: int64

In [ ]:
```
df['City_Category'].value_counts()
```

Out[ ]:

City_Category	count
B	231173
C	171175
A	147720

dtype: int64

In [ ]:
```
df['Stay_In_Current_City_Years'].value_counts()
```

Out[ ]:

Stay_In_Current_City_Years	count
1	193821
2	101838
3	95285
4+	84726
0	74398

dtype: int64

In [ ]:
```
df['Marital_Status'].value_counts()
```

Out[ ]:

Marital_Status	count
0	324731
1	225337

dtype: int64

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

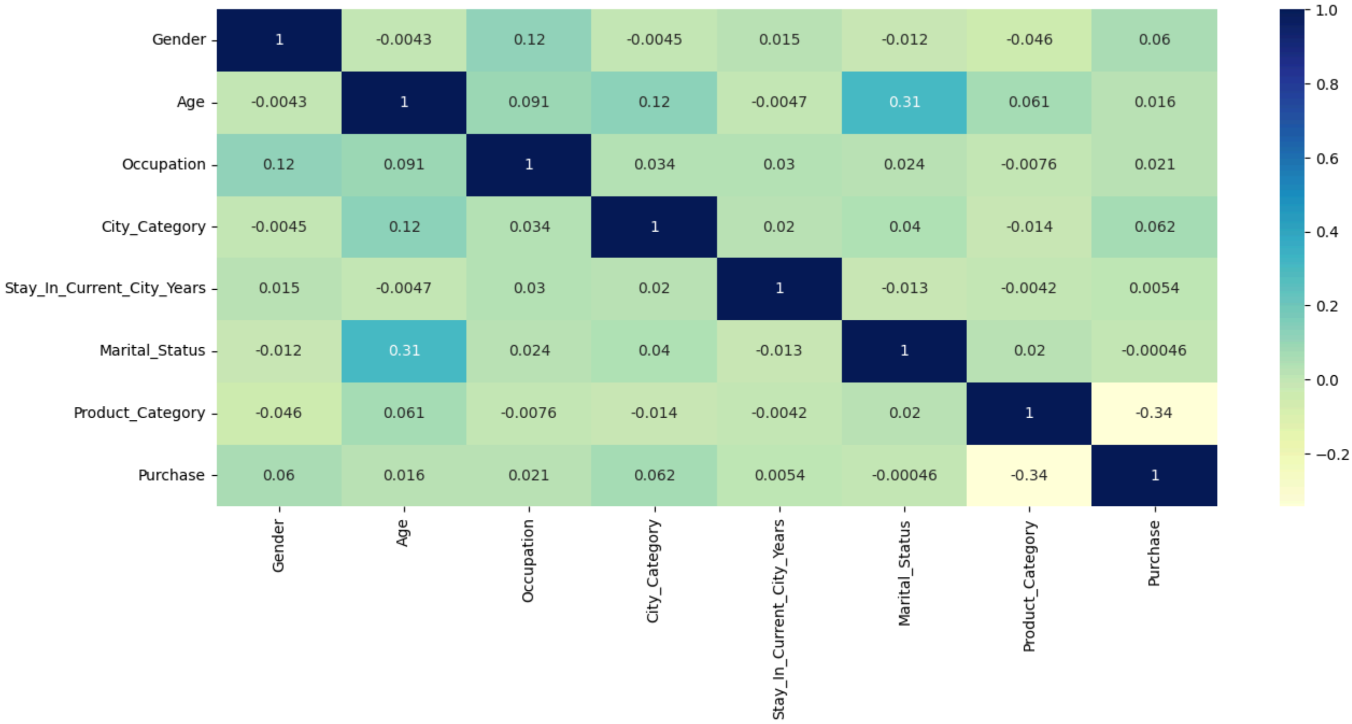
Out[ ]:

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

Correalation Analysis

In [ ]:
```
df_copied = df.copy()
df_copied['Gender'].replace(['F','M'], [0,1], inplace  = True)
df_copied['Age'].replace(['0-17', '18-25', '26-35', '36-45', '46-50', '51-55', '55+'],[0,1,2,
df_copied['City_Category'].replace(['A', 'B', 'C'], [0,1,2] , inplace = True)
df_copied['Stay_In_Current_City_Years'] = df_copied['Stay_In_Current_City_Years'].replace({'4


df_copied = df_copied[['Gender', 'Age', 'Occupation', 'City_Category', 'Stay_In_Current_City_

# Correlation Plot above as a Heatmap -

plt.figure(figsize=(15,6))
sns.heatmap(df_copied.corr(), cmap="YlGnBu", annot=True)
plt.show()
```

```python
df_copied.corr()
```



Out[ ]:

	Gender	Age	Occupation	City_Category	Stay_In_Current_City_Years
Gender	1.000000	-0.004262	0.117291	-0.004515	0.014660
Age	-0.004262	1.000000	0.091463	0.123079	-0.004712
Occupation	0.117291	0.091463	1.000000	0.034479	0.030005
City_Category	-0.004515	0.123079	0.034479	1.000000	0.019946
Stay_In_Current_City_Years	0.014660	-0.004712	0.030005	0.019946	1.000000
Marital_Status	-0.011603	0.311738	0.024280	0.039790	-0.012819
Product_Category	-0.045594	0.061197	-0.007618	-0.014364	-0.004213
Purchase	0.060346	0.015839	0.020833	0.061914	0.005422

Age: No correlation; purchase amount is fairly consistent across age groups. Differences in sales likely come from group size (26–35 being the largest) rather than spending behavior.

Gender: Slight positive correlation with purchases; males tend to spend marginally more, but since they form ~75% of the base, they dominate overall sales. Female customers, though smaller in number, are a potential growth segment. Further testing can help check whether the gender difference is statistically reliable.

Marital Status: No correlation; being single or married doesn't influence spending. Further analysis can confirm if this lack of relationship holds across different product categories or subgroups.

Further analysis using CLT and confidence intervals can validate whether the observed differences are statistically meaningful.

Is there a relationship between gender and the amount spent?

In [ ]:
```python
df.groupby("Gender")['User_ID'].nunique()
```

Out[ ]:

	User_ID
Gender	
F	1666
M	4225

dtype: int64

In [ ]:

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

Out[ ]:

	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 [22]:

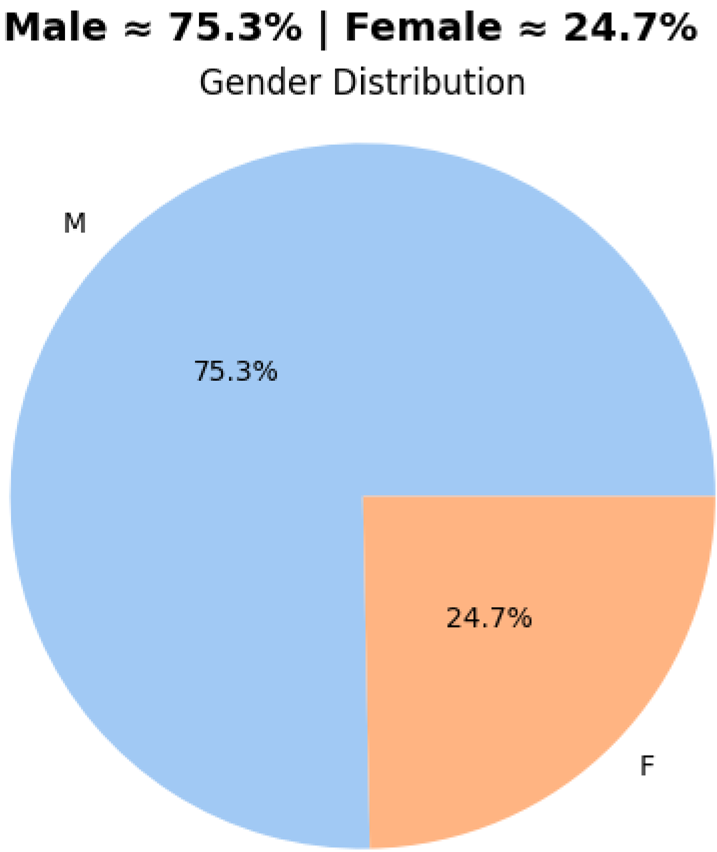
```
x = df['Gender'].value_counts().values
label = df['Gender'].value_counts().index
x, label
```

Out[22]:

```
(array([414259, 135809]), Index(['M', 'F'], dtype='object', name='Gender'))
```

In [26]:

```
plt.figure(figsize=(5, 5))
colors = sns.color_palette("pastel")[0:2]
plt.pie(x, center=(0, 0), radius=1.5, labels= label, autopct='%1.1f%%', pctdistance=0.5, colo

plt.suptitle("Male ≈ 75.3% | Female ≈ 24.7%",
             fontsize=14, fontweight='bold')
plt.title('Gender Distribution',  fontsize=12)

plt.axis('equal')
plt.show()
```



How Gender VS Purchase values are distributed

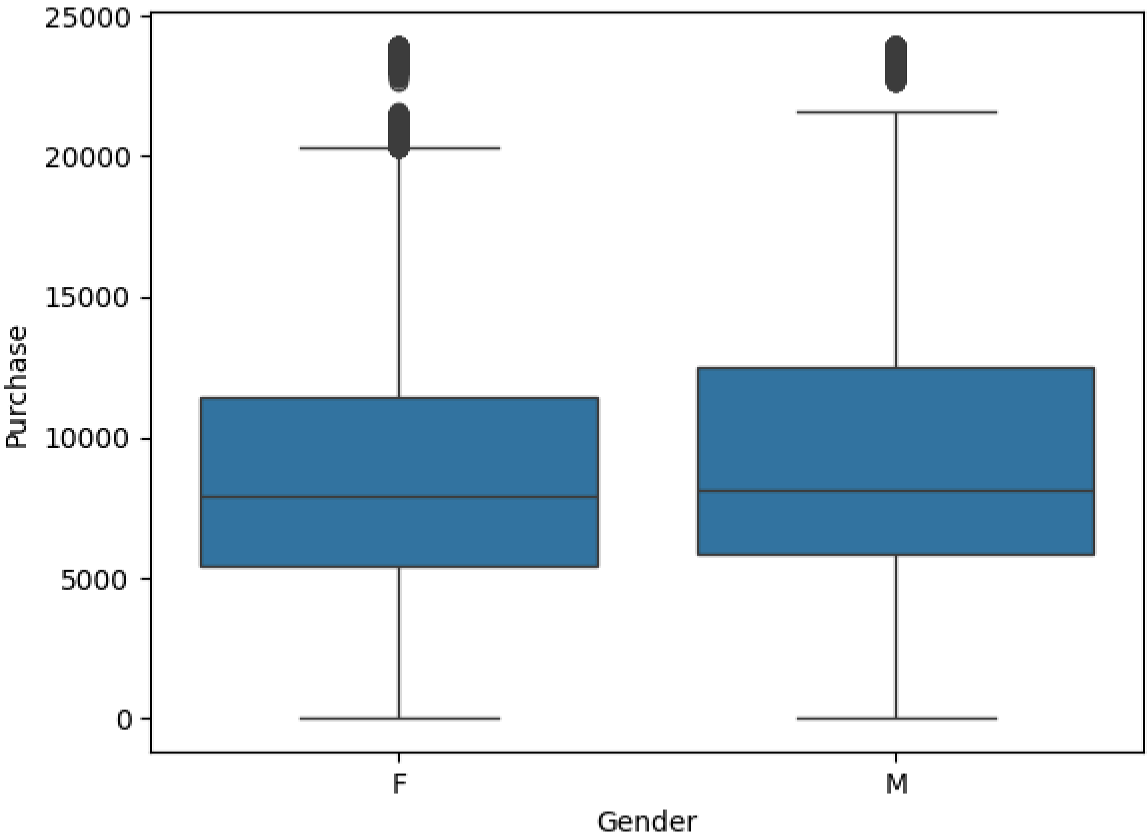
In [ ]:
```
round(df.groupby('Gender')['Purchase'].mean() ,2)
```

Out[ ]:

Gender	Purchase
F	8734.57
M	9437.53

dtype: float64

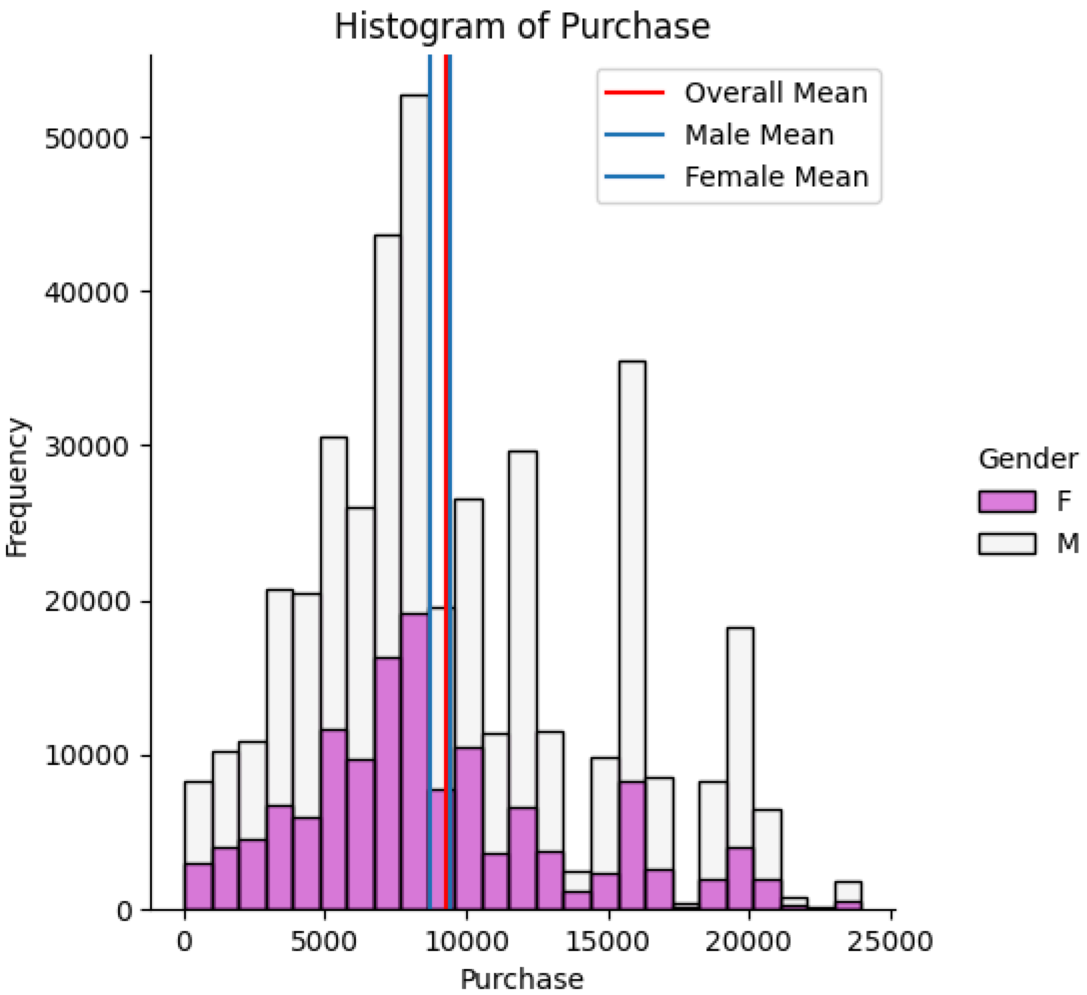
In [ ]:
```
sns.boxplot(x = 'Gender', y = 'Purchase',  data = df)
plt.show()
```



Histogram - Males vs Females purchase data

In [ ]:
```
plt.figure(figsize = (20,8))
sns.displot(x = 'Purchase', data = df, bins = 25, hue = 'Gender', palette="light:m_r")

# Plot vertical lines for mean values
plt.axvline(x=df['Purchase'].mean(), color='r', label='Overall Mean')
plt.axvline(x=df[df['Gender'] == 'M']['Purchase'].mean(), label='Male Mean')
plt.axvline(x=df[df['Gender'] == 'F']['Purchase'].mean(), label='Female Mean')

# Axis labels and title
plt.xlabel('Purchase')
plt.ylabel('Frequency')
plt.title('Histogram of Purchase')

# Show legend and plot
plt.legend()
plt.show()
```

```
<Figure size 2000x800 with 0 Axes>
```



Insights: Males (M) generally exhibit higher counts across most bins, implying either more male customers or higher purchase frequency.

In [ ]:
```
df.sample(300).groupby('Gender')['Purchase'].describe()
```

Out[ ]:

Gender	count	mean	std	min	25%	50%	75%	max
F	77.0	9026.207792	5211.856890	407.0	5400.0	7916.0	11925.0	23678.0
M	223.0	9470.520179	5598.807256	363.0	5383.0	8016.0	15196.0	23700.0

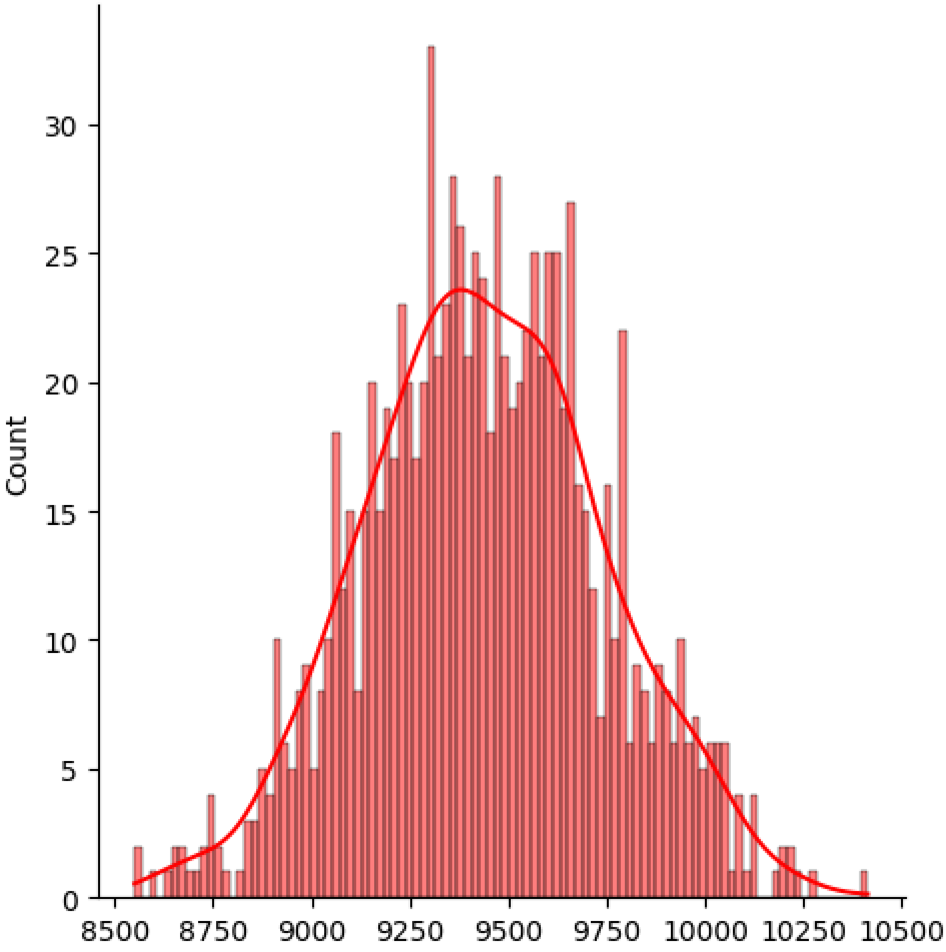
sample size 300

In [5]:
```
size = 300
iterations = 1000
```

In [6]:
```
male_sample_means = [df[df['Gender']=='M']['Purchase'].sample(size).mean() for i in range(ite
```

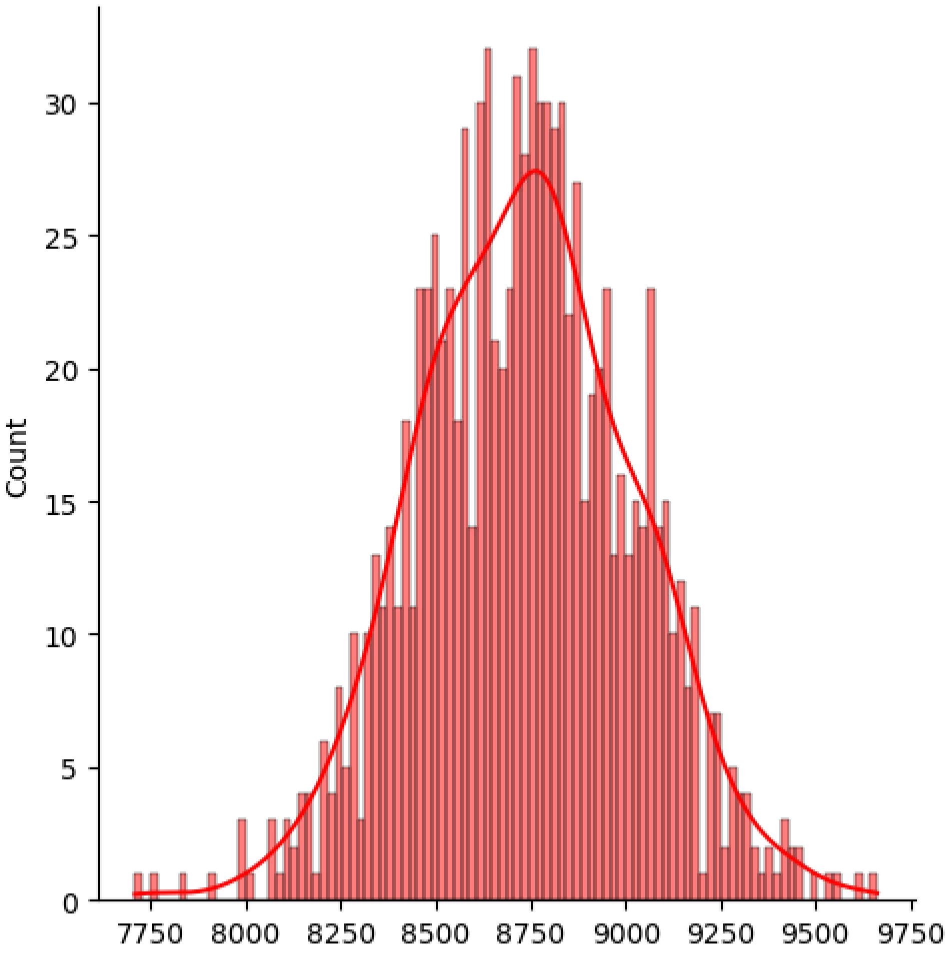
In [7]:
```
female_sample_means = [df[df['Gender']=='F']['Purchase'].sample(size).mean() for i in range(i
```

In [ ]:
```
sns.displot(male_sample_means, bins=100, kde=True, color='r')

plt.show()

print('Mean (for Males): ', round(pd.Series(male_sample_means).mean(),2))
```




```
Mean (for Males):  9439.54
```

In [ ]:
```
sns.displot(female_sample_means, bins=100, kde=True, color='r')
plt.show()
print('Mean (for Females): ', round(pd.Series(female_sample_means).mean(),2))
```




```
Mean (for Females):  8734.72
```

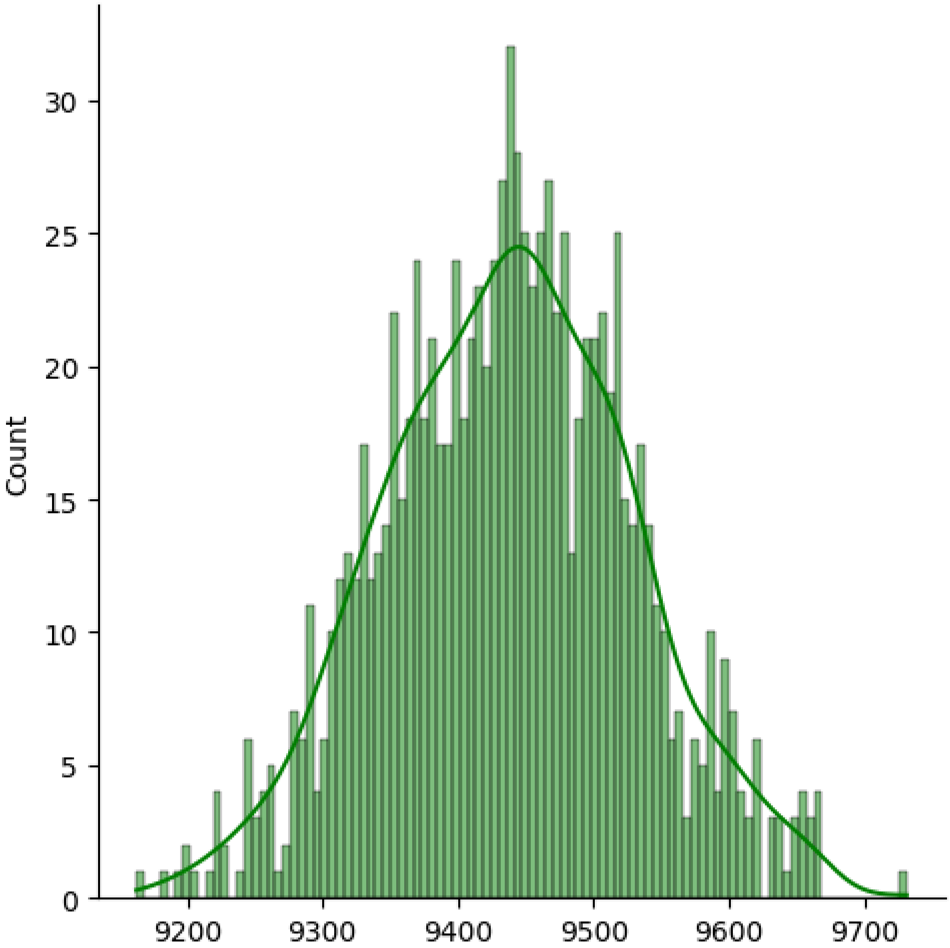
gender vs purchase distribution for sample size 3000

In [ ]:
```
size_3000 = 3000
iterations_3000 = 1000
```

In [ ]:
```
male_sample_means_3000 = [df[df['Gender']=='M']['Purchase'].sample(size_3000).mean() for i in
```

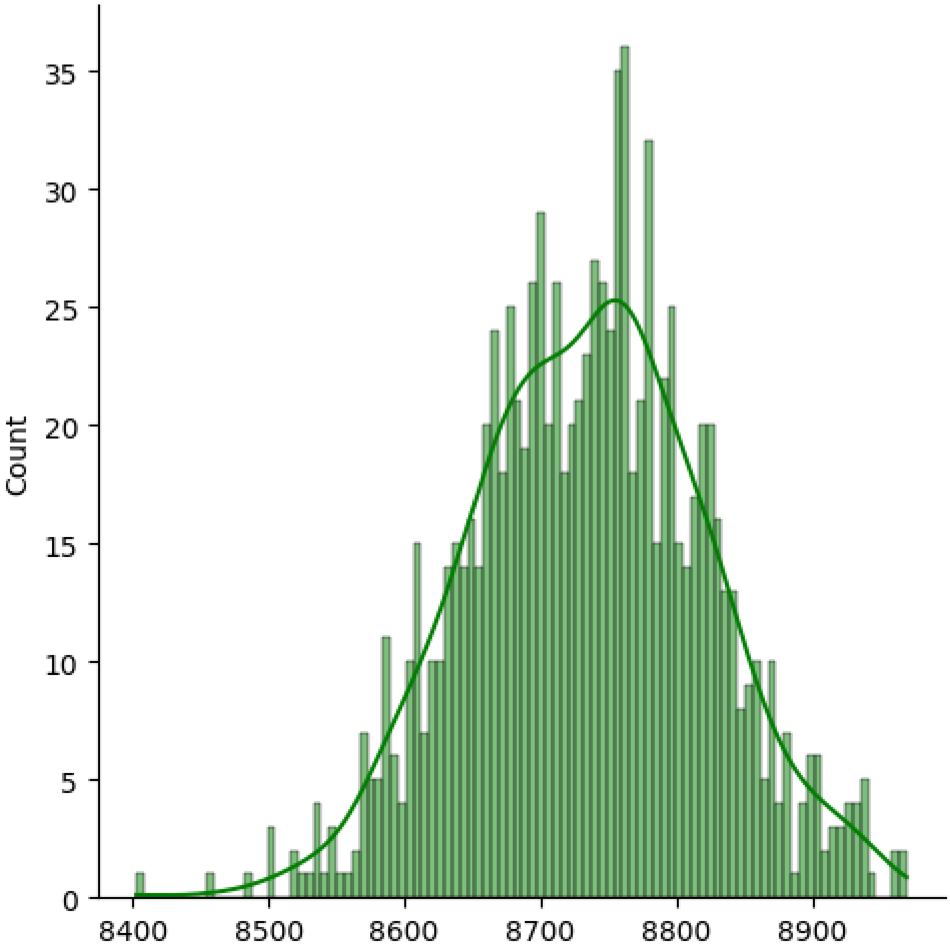
In [ ]:
```
female_sample_means_3000 = [df[df['Gender']=='F']['Purchase'].sample(size_3000).mean() for i .
```

In [ ]:
```
sns.displot(male_sample_means_3000, bins=100, kde=True, color='g')

plt.show()

print('Mean (for Males) for 3000 sample size: ', round(pd.Series(male_sample_means_3000).mean
```




```
Mean (for Males) for 3000 sample size:  9437.56
```

In [ ]:
```
sns.displot(female_sample_means_3000, bins=100, kde=True, color='g')

plt.show()

print('Mean (for Feales) for 3000 sample size: ', round(pd.Series(female_sample_means_3000).m
```



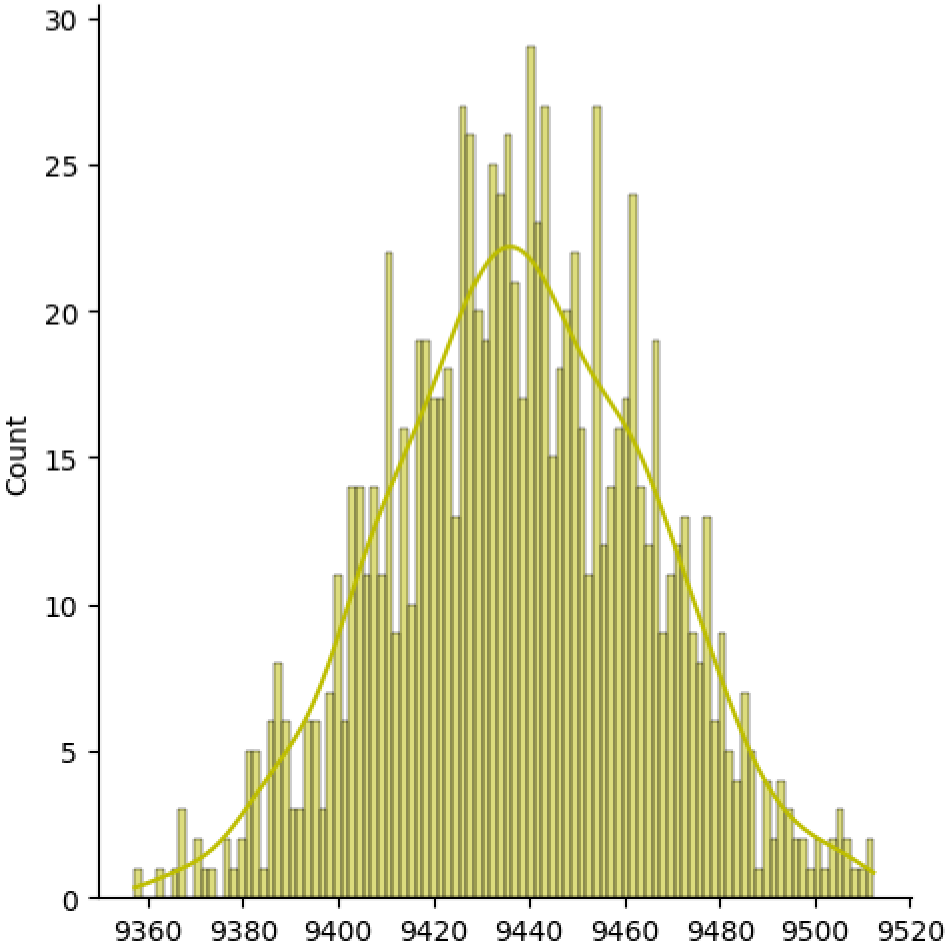

```
Mean (for Feales) for 3000 sample size:  8734.61
```

In [ ]:
```
size_30000 = 30000
iterations_30000 = 1000
```

In [ ]:
```
male_sample_means_30000 = [df[df['Gender']=='M']['Purchase'].sample(size_30000).mean() for i
```

In [ ]:
```
female_sample_means_30000 = [df[df['Gender']=='F']['Purchase'].sample(size_30000).mean() for
```

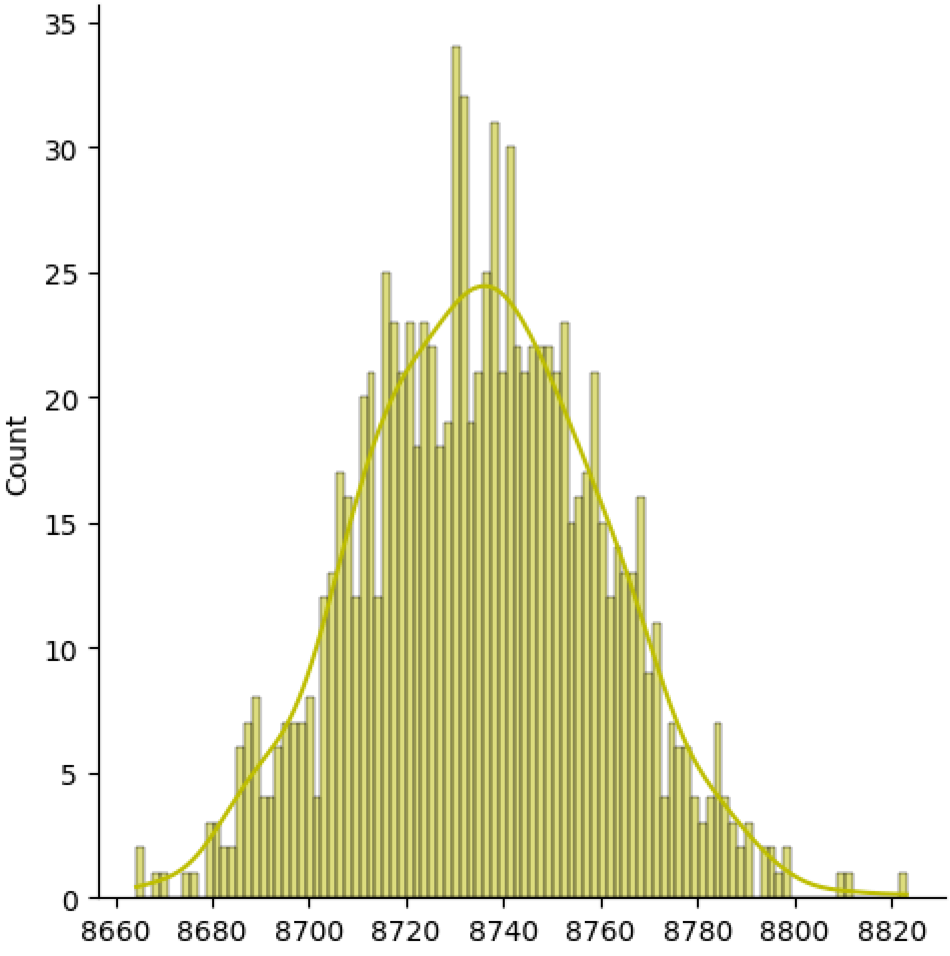
In [ ]:
```
sns.displot(male_sample_means_30000, bins=100, kde=True, color='y')

plt.show()

print('Mean (for males) for 30000 sample size: ', round(pd.Series(male_sample_means_30000).me
```




```
Mean (for males) for 30000 sample size:  9438.07
```

In [ ]:

```
sns.displot(female_sample_means_30000, bins=100, kde=True, color='y')

plt.show()

print('Mean (for Females) for 30000 sample size: ', round(pd.Series(female_sample_means_30000
```



```
Mean (for Females) for 30000 sample size:  8735.42
```

```
In [ ]:
```

Insights:

When we repeatedly draw random samples from a population and compute their means, the distribution of those sample means tends to be normal (bell-shaped).

As we increase the sample size, the average of the sample means moves closer to the population mean.

Confidence Interval for 90% of confidence

```
In [10]: lower_limit_m = pd.Series(male_sample_means).mean() - pd.Series(male_sample_means).std()/np.s
         upper_limit_m = pd.Series(male_sample_means).mean() + pd.Series(male_sample_means).std()/np.s

         print("Lower limit for Males is {:.2f} ".format(lower_limit_m))
         print("Upper limit for Males is {:.2f} ".format(upper_limit_m))
```

```
Lower limit for Males is 9411.52
Upper limit for Males is 9440.61
```

```
In [11]: lower_limit_f = pd.Series(female_sample_means).mean() - pd.Series(female_sample_means).std()/
         upper_limit_f = pd.Series(female_sample_means).mean() + pd.Series(female_sample_means).std()/

         print("Lower limit for Females is {:.2f} ".format(lower_limit_f))
         print("Upper limit for Females is {:.2f} ".format(upper_limit_f))
```

```
Lower limit for Females is 8725.78
Upper limit for Females is 8753.93
```

Confidence Interval for 95% of confidence

```
In [13]: lower_limit_m_95 = pd.Series(male_sample_means).mean() - pd.Series(male_sample_means).std()/n
         upper_limit_m_95 = pd.Series(male_sample_means).mean() + pd.Series(male_sample_means).std()/n

         print("Lower limit for Males is {:.2f} ".format(lower_limit_m_95))
         print("Upper limit for Males is {:.2f} ".format(upper_limit_m_95))
```

```
Lower limit for Males is 9408.73
Upper limit for Males is 9443.40
```

```
In [14]: lower_limit_f_95 = pd.Series(female_sample_means).mean() - pd.Series(female_sample_means).std
         upper_limit_f_95 = pd.Series(female_sample_means).mean() + pd.Series(female_sample_means).std
         print("Lower limit for Females is {:.2f} ".format(lower_limit_f_95))
         print("Upper limit for Females is {:.2f} ".format(upper_limit_f_95))
```

```
Lower limit for Females is 8723.09
Upper limit for Females is 8756.63
```

Non-overlapping Confidence Intervals

The male interval lies entirely above the female interval.

Since the intervals do not overlap, this indicates a statistically significant difference in purchase averages between males and females at the 90% and 95% confidence level.

Is there a relationship between marital status, and the amount spent?

```
In [ ]: df.groupby("Marital_Status")['User_ID'].nunique()
```

Out[ ]:

	User_ID
Marital_Status	
0	3417
1	2474

dtype: int64

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

Out[ ]:

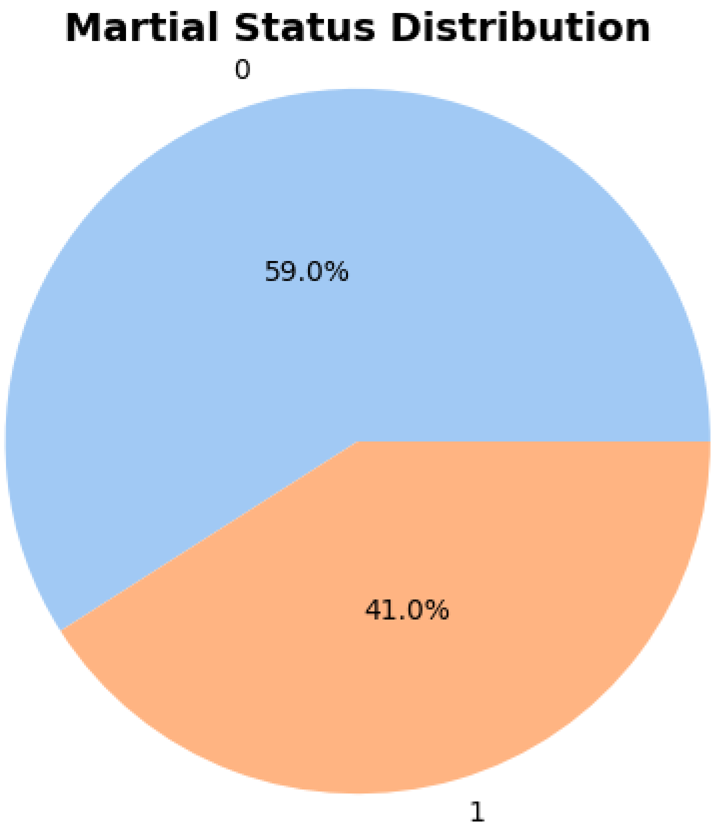
	count	mean	std	min	25%	50%	75%	max
Marital_Status								
0	324731.0	9265.907619	5027.347859	12.0	5605.0	8044.0	12061.0	23961.0
1	225337.0	9261.174574	5016.897378	12.0	5843.0	8051.0	12042.0	23961.0

In [ ]:
```
x = df['Marital_Status'].value_counts().values
label = df['Marital_Status'].value_counts().index
x, label
```

Out[ ]:
```
(array([324731, 225337]), Index([0, 1], dtype='int64', name='Marital_Status'))
```

In [ ]:
```
plt.figure(figsize=(5, 5))

colors = sns.color_palette("pastel")[0:2]
plt.pie(x, center=(0, 0), radius=1.5, labels= label, autopct='%1.1f%%', pctdistance=0.5, colo

plt.suptitle("Unmarried ≈ 59.0% | Female ≈ 41.0%",
             fontsize=10, color="gray")
plt.title('Martial Status Distribution', fontsize=14, fontweight="bold")

plt.axis('equal')
plt.show()
```

Unmarried ≈ 59.0% | Female ≈ 41.0%



How Martial Status vs Purchase values are distributed

In [ ]:
```
round(df.groupby('Marital_Status')['Purchase'].mean() ,2)
```

Out[ ]:

	Purchase
Marital_Status	
0	9265.91
1	9261.17

dtype: float64

Insights

Mean purchase for Unmarried (0): $9265.91

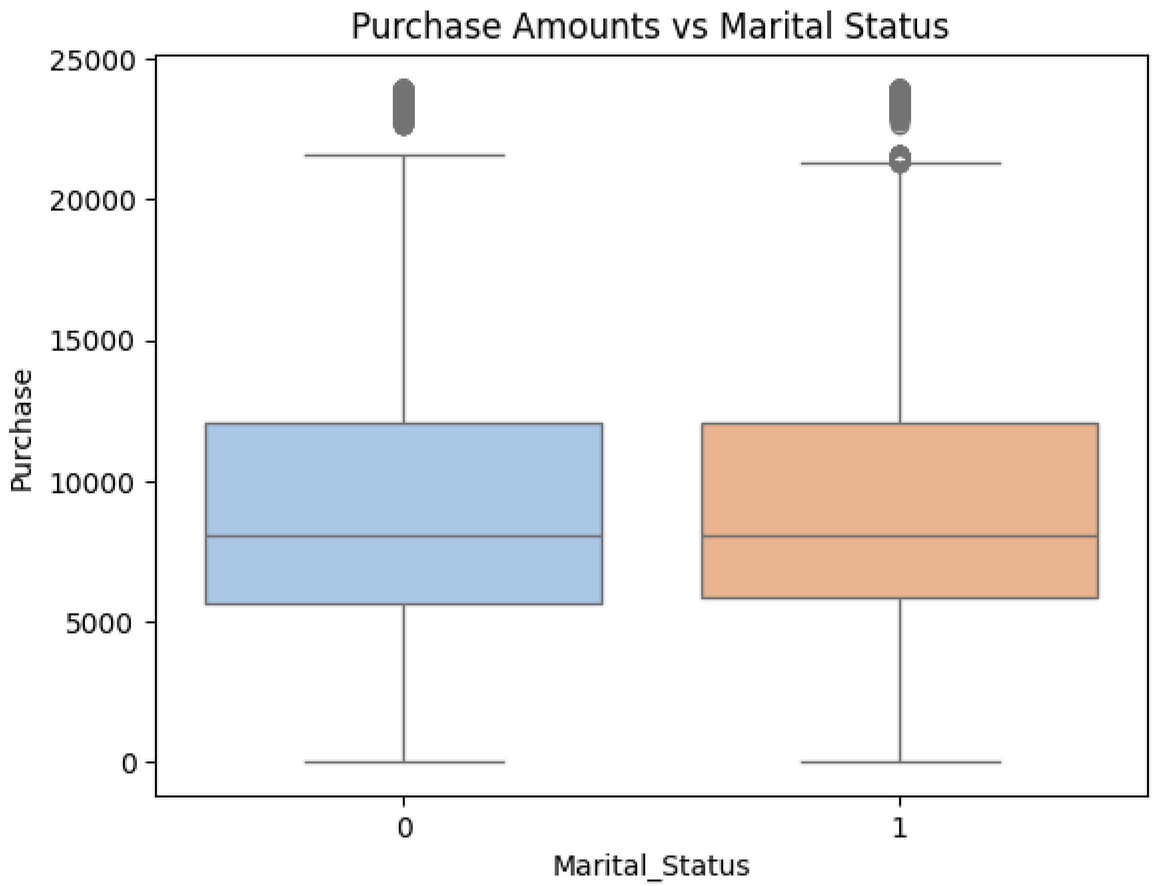
Mean purchase for Married (1): $9261.17

Difference is extremely small compared to overall purchase amounts (ranging 0–25,000).

So, they look the same. But we need to check:

Is this similarity statistically significant, or just by chance?

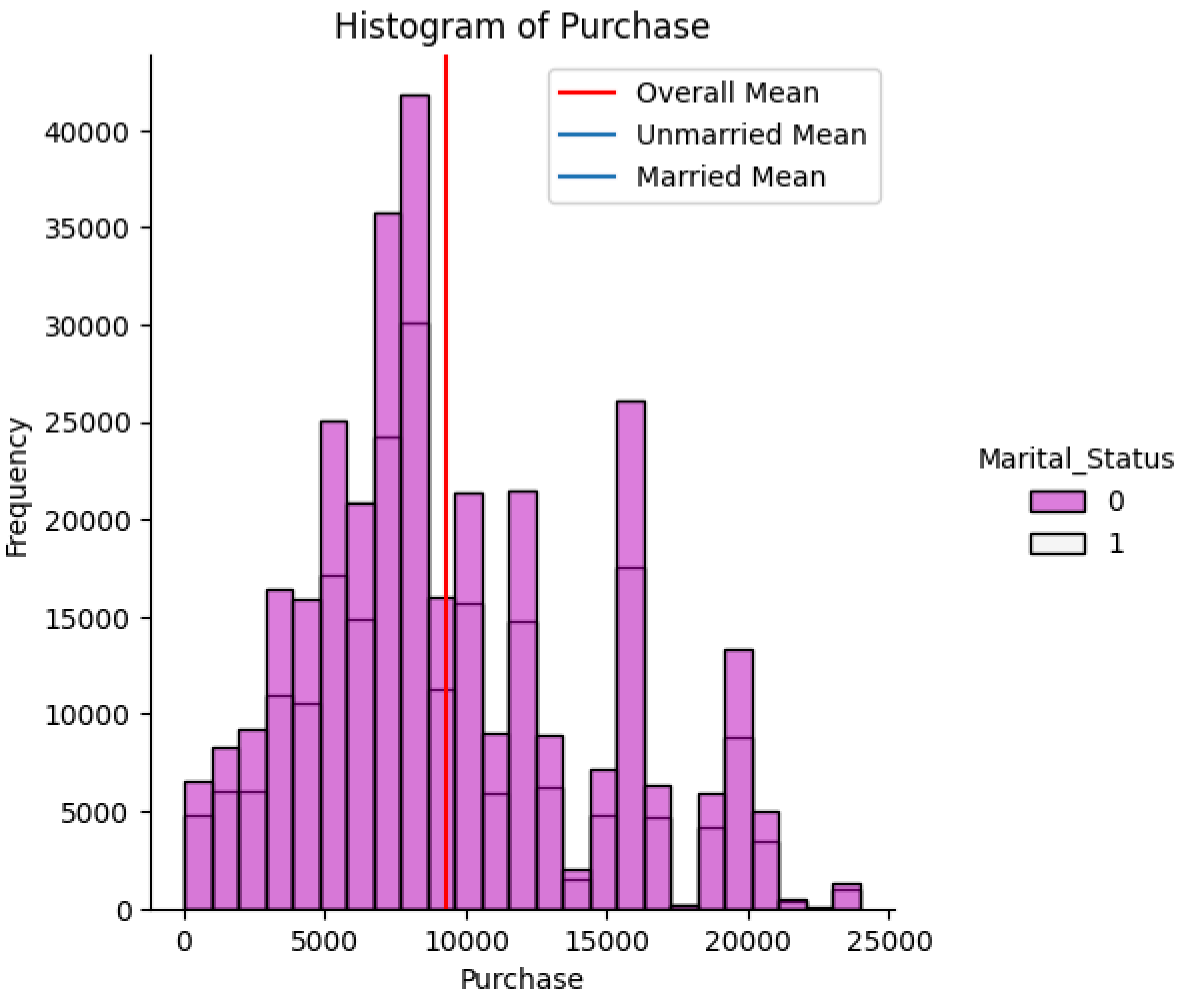
In [20]:
```
sns.boxplot(x= 'Marital_Status', y = 'Purchase', data = df, palette="pastel")

plt.title("Purchase Amounts vs Marital Status", fontsize=12)
plt.show()
```



Histogram - Unmarried and Married customers vs purchase data

In [4]:
```
plt.figure(figsize = (20,8))
sns.displot(x = 'Purchase', data = df, bins = 25, hue = 'Marital_Status', palette="light:m_r"
# Plot vertical lines for mean values
plt.axvline(x=df['Purchase'].mean(), color='r', label='Overall Mean')
plt.axvline(x=df[df['Marital_Status'] == '0']['Purchase'].mean(), label='Unmarried Mean')
plt.axvline(x=df[df['Gender'] == '1']['Purchase'].mean(), label='Married Mean')
# Axis labels and title
plt.xlabel('Purchase')
plt.ylabel('Frequency')
plt.title('Histogram of Purchase')
# Show legend and plot
plt.legend()
plt.show()
```

```
<Figure size 2000x800 with 0 Axes>
```



Insights:

Both married and unmarried customers spend nearly the same amount on average, is it significant or by chance

In [5]:
```
df.sample(300).groupby('Marital_Status')['Purchase'].describe()
```

Out[5]:

Marital_Status	count	mean	std	min	25%	50%	75%	max
0	174.0	9142.678161	5454.838429	352.0	5231.75	7933.0	12800.5	23471.0
1	126.0	8884.142857	5047.443351	50.0	5307.00	7903.0	12179.0	20626.0

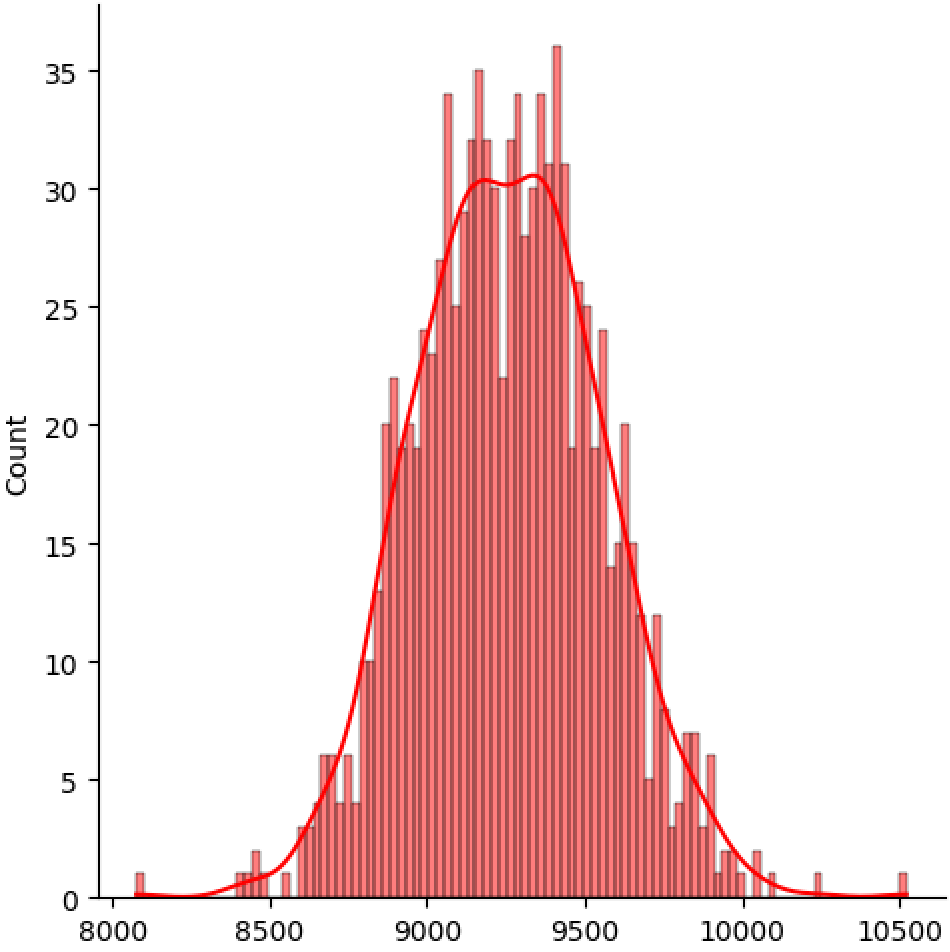
sample size 300

In [27]:
```
# sample size 300 and iterations 1000 for CLT
um_size_300 = 300
um_iterations = 1000
```

In [28]:
```
um_sample_means = [df[df['Marital_Status']==0]['Purchase'].sample(um_size_300).mean() for i in
```

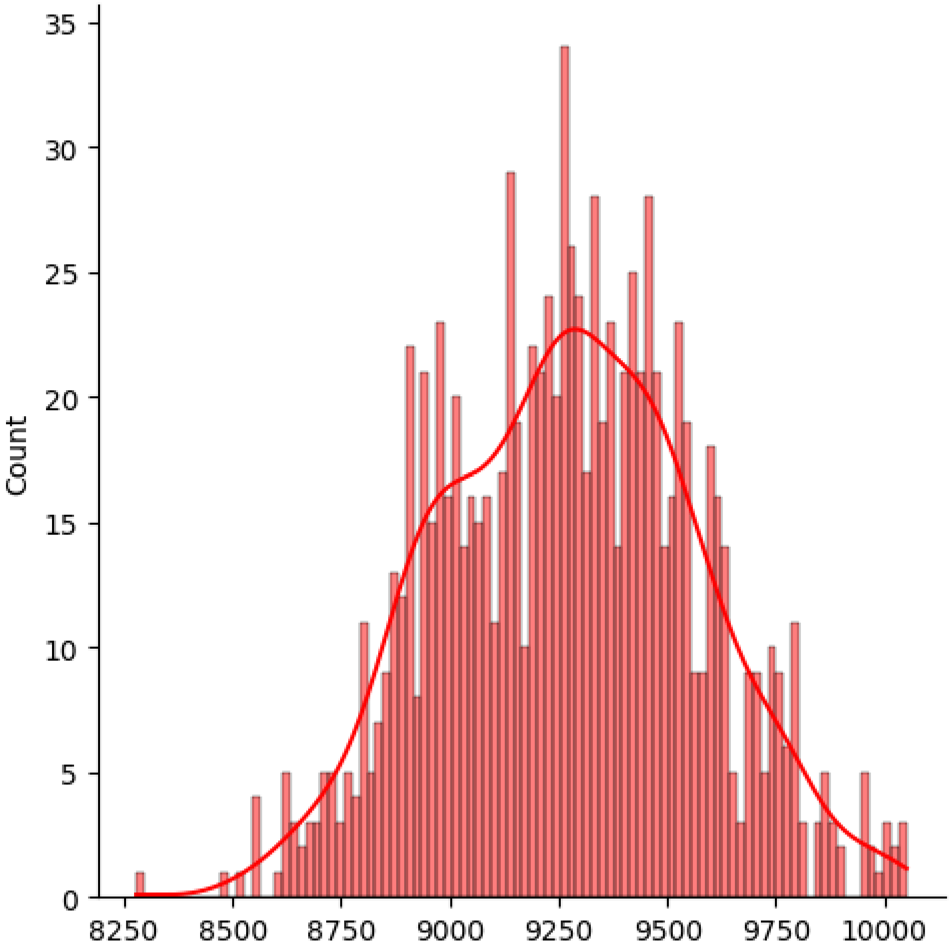
In [29]:
```
m_sample_means = [df[df['Marital_Status']==1]['Purchase'].sample(um_size_300).mean() for i in
```

In [13]:
```
sns.displot(um_sample_means, bins=100, kde=True, color='r')
plt.show()
print('Mean (for Unmarried Customers): ', round(pd.Series(um_sample_means).mean(),2))
```




```
Mean (for Unmarried Customers):  9257.97
```

In [14]:
```python
sns.displot(m_sample_means, bins=100, kde=True, color='r')
plt.show()
print('Mean (for Married Customers): ', round(pd.Series(m_sample_means).mean(),2))
```




```
Mean (for Married Customers):  9272.17
```

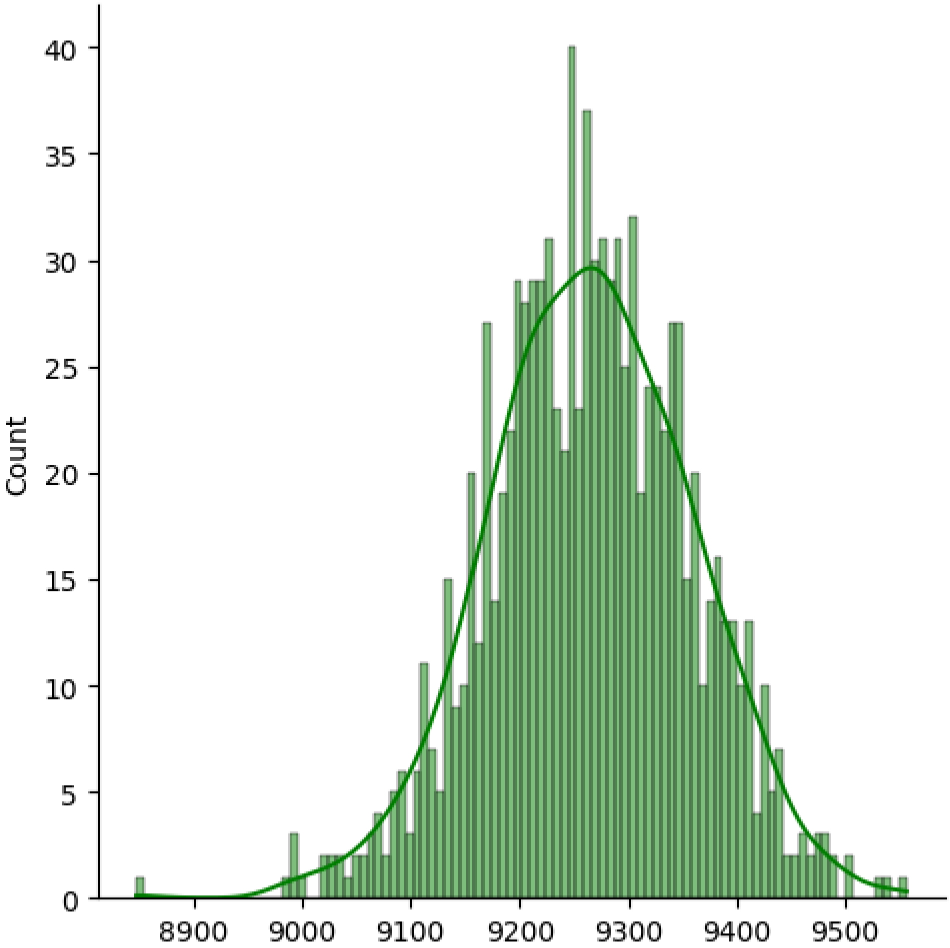
Marital_status vs purchase distribution for sample size 3000

In [15]:
```
um_size_3000 = 3000
um_iterations_3000 = 1000
```

In [16]:
```
um_sample_means_3000 = [df[df['Marital_Status']==0]['Purchase'].sample(um_size_3000).mean() fo
```

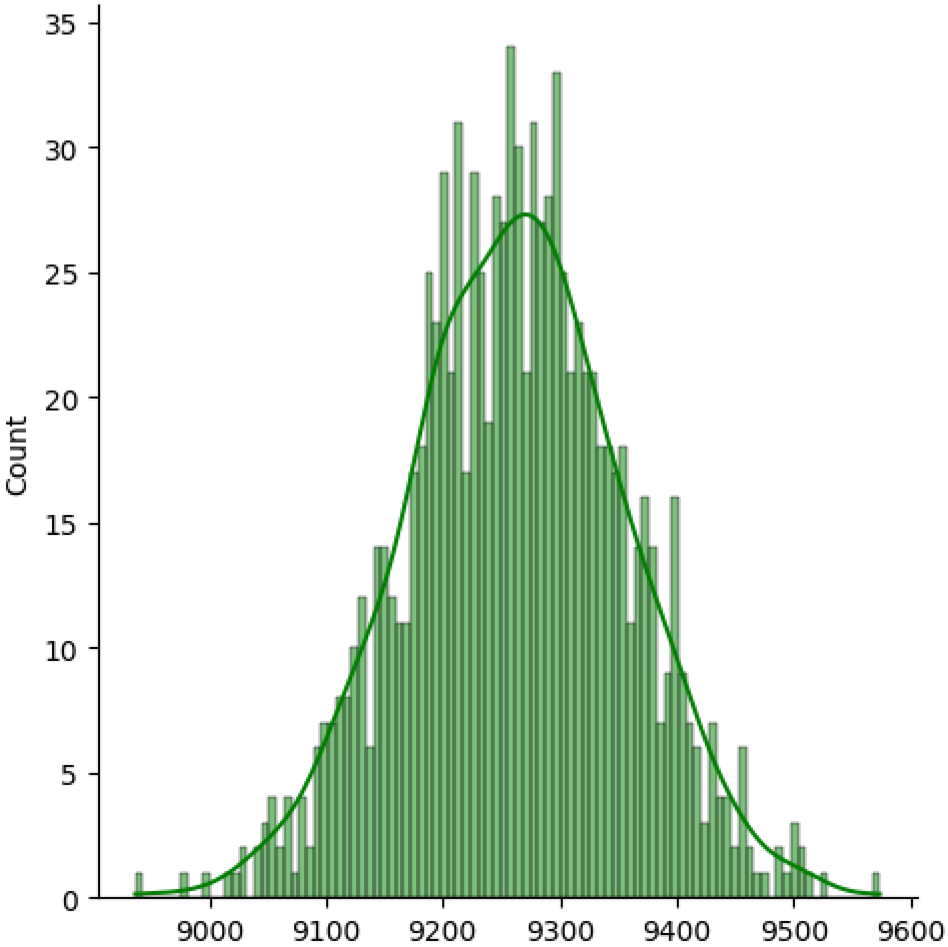
In [17]:
```
mm_sample_means_3000 = [df[df['Marital_Status']==1]['Purchase'].sample(um_size_3000).mean() fo
```

In [18]:
```
sns.displot(um_sample_means_3000, bins=100, kde=True, color='g')
plt.show()
print('Mean (for Unmarried Customers): ', round(pd.Series(um_sample_means_3000).mean(),2))
```




```
Mean (for Unmarried Customers):  9264.72
```

In [20]:
```
sns.displot(mm_sample_means_3000, bins=100, kde=True, color='g')
plt.show()
print('Mean (for Married Customers): ', round(pd.Series(mm_sample_means_3000).mean(),2))
```



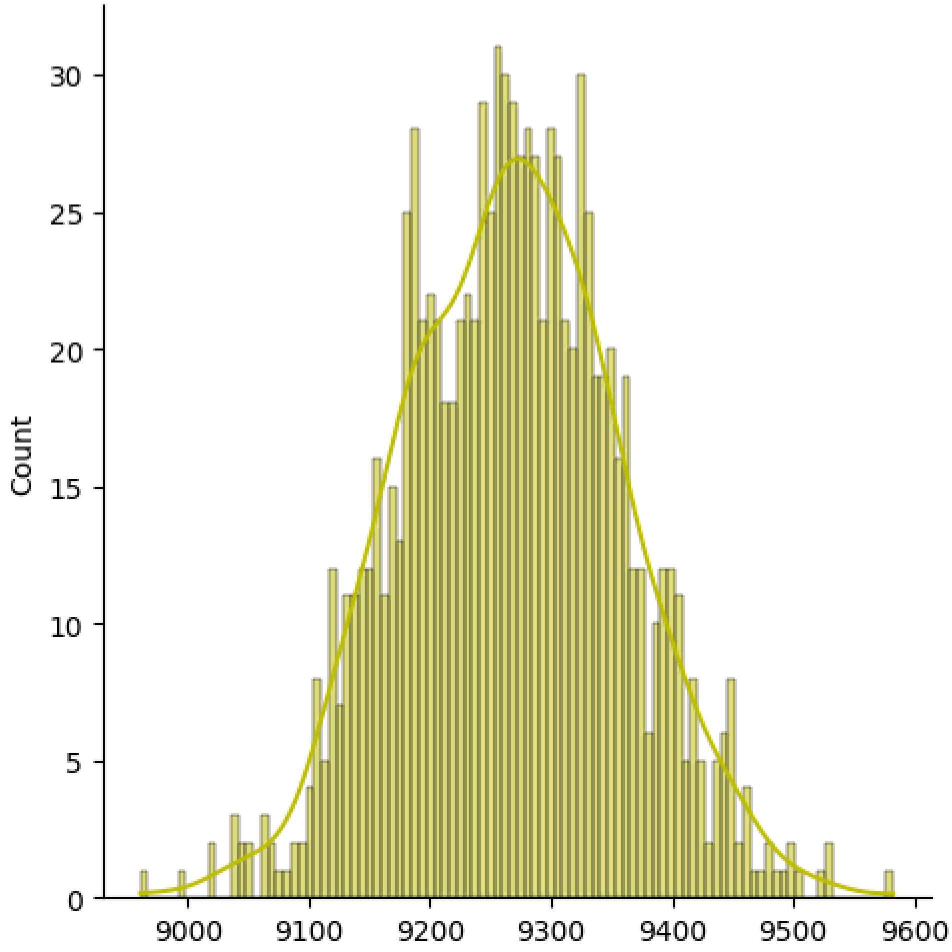

```
Mean (for Married Customers):  9262.24
```

In [21]:
```
um_size_30000 = 3000
um_iterations_30000 = 1000
```

In [22]:
```
um_sample_means_30000 = [df[df['Marital_Status']==0]['Purchase'].sample(um_size_30000).mean()
```

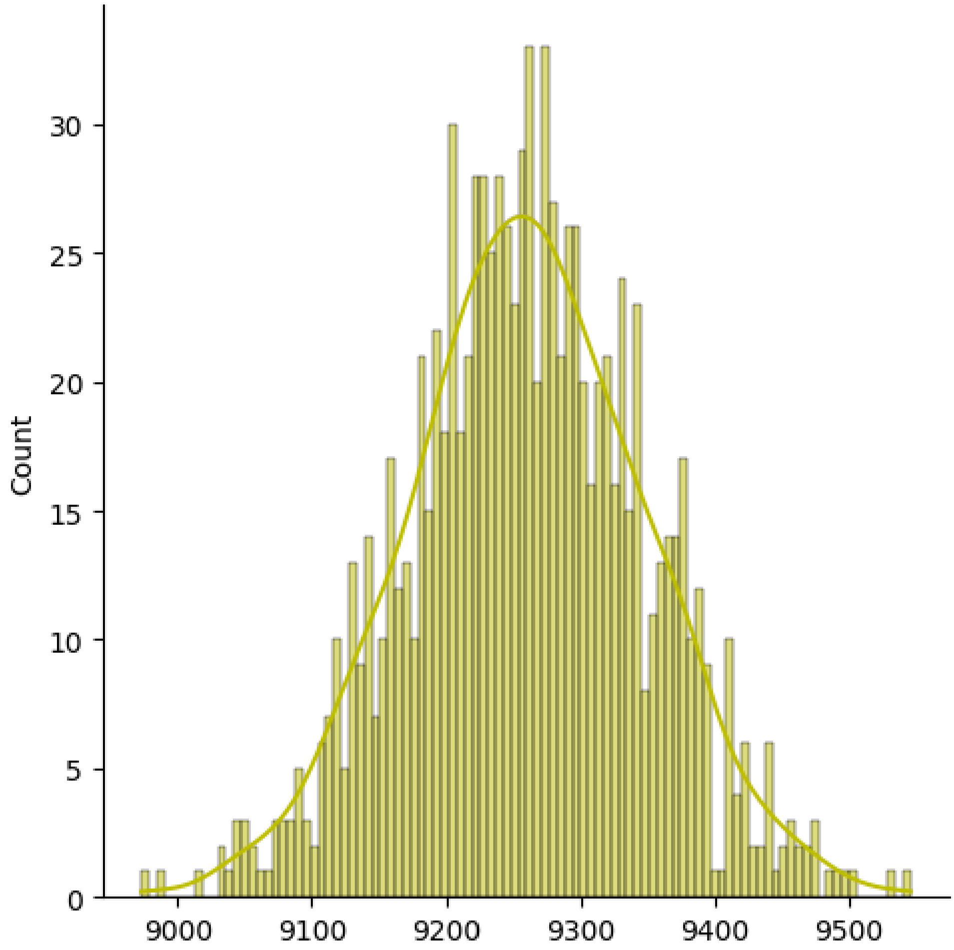
In [28]:
```
m_sample_means_30000 = [df[df['Marital_Status']==1]['Purchase'].sample(um_size_30000).mean()
```

In [24]:
```
sns.displot(um_sample_means_30000, bins=100, kde=True, color='y')
plt.show()
print('Mean (for Unmarried Customers): ', round(pd.Series(um_sample_means_30000).mean(),2))
```




```
Mean (for Unmarried Customers):  9267.61
```

In [29]:
```python
sns.displot(m_sample_means_30000, bins=100, kde=True, color='y')
plt.show()
print('Mean (for Married Customers): ', round(pd.Series(m_sample_means_30000).mean(),2))
```




```
Mean (for Married Customers):  9259.62
```

Insights: When we repeatedly draw random samples from a population and compute their means, the distribution of those sample means tends to be normal (bell-shaped). As we increase the sample size, the average of the sample means moves closer to the population mean.

Confidence Interval for 90% of confidence

In [35]:
```
# for unmarried customers
import scipy.stats as stats
alpha = 0.90
dof = len(um_sample_means) - 1
stats.t.interval(alpha, dof, loc = np.mean(um_sample_means),
                             scale  = stats.sem(um_sample_means))
```

Out[35]: (np.float64(9242.331285428976), np.float64(9271.617514571022))

In [36]:
```
# for married customers
import scipy.stats as stats
alpha = 0.90
dof = len(m_sample_means) - 1
stats.t.interval(alpha, dof, loc = np.mean(m_sample_means),
                             scale  = stats.sem(m_sample_means))
```

Out[36]: (np.float64(9239.897480346475), np.float64(9270.023666320189))

Confidence Interval for 95% of confidence

In [37]:
```
# for unmarried customers
import scipy.stats as stats
alpha = 0.95
dof = len(um_sample_means) - 1
stats.t.interval(alpha, dof, loc = np.mean(um_sample_means),
                             scale  = stats.sem(um_sample_means))
```

Out[37]: (np.float64(9239.521087390522), np.float64(9274.427712609477))

In [38]:
```
# for married customers
import scipy.stats as stats
alpha = 0.95
dof = len(m_sample_means) - 1
stats.t.interval(alpha, dof, loc = np.mean(m_sample_means),
                             scale  = stats.sem(m_sample_means))
```

Out[38]: (np.float64(9237.006683162626), np.float64(9272.914463504037))

Overlapping Confidence Intervals

Since the intervals of married and unmarried customers overlap, this indicates there is no significant differnce in averages purchases between unmarried and married at the 90% and 95% confidence level.

Is there a relationship between differenct age groups and the amount spent?

In [39]:
```
df.groupby("Age")['User_ID'].nunique()
```

Out[39]:

Age	User_ID
0-17	218
18-25	1069
26-35	2053
36-45	1167
46-50	531
51-55	481
55+	372

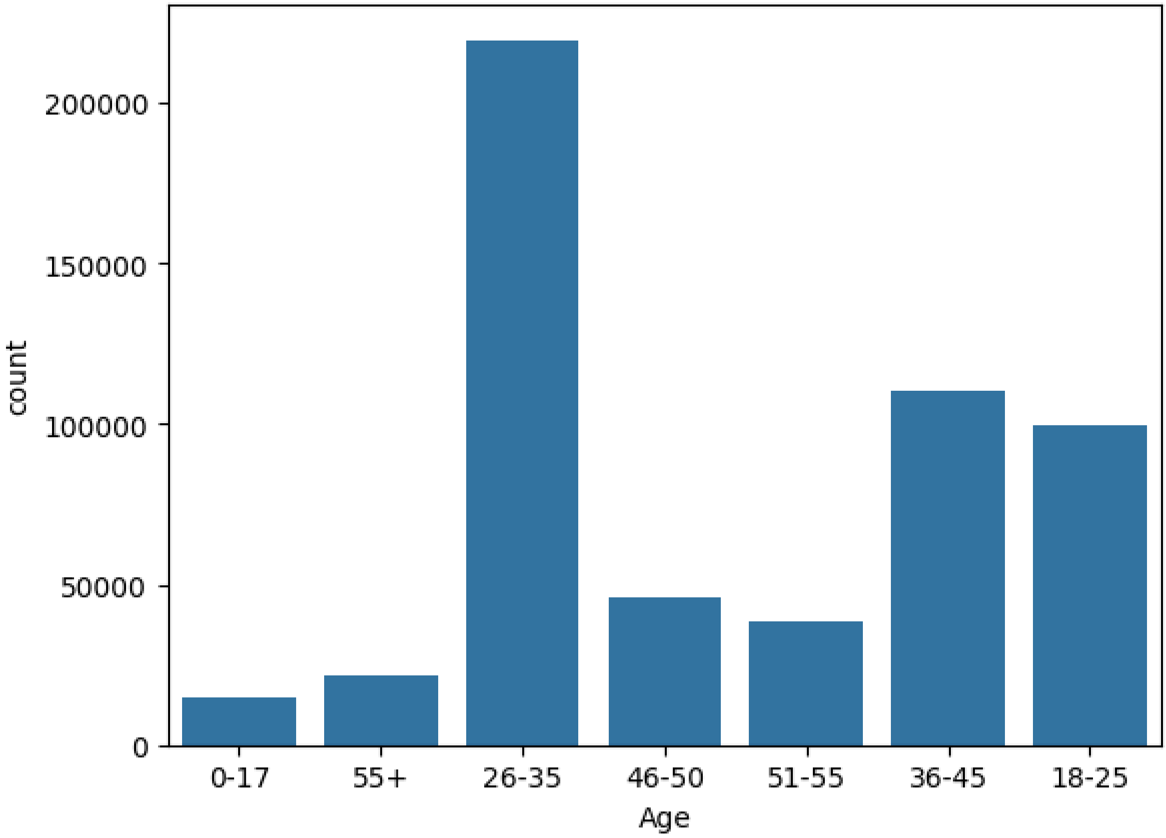
dtype: int64

In [42]:
```
df.groupby("Age")['Purchase'].describe().T
```

Out[42]:

Age	0-17	18-25	26-35	36-45	46-50	51-55	
count	15102.000000	99660.000000	219587.000000	110013.000000	45701.000000	38501.000000	21504.0
mean	8933.464640	9169.663606	9252.690633	9331.350695	9208.625697	9534.808031	9336.2
std	5111.114046	5034.321997	5010.527303	5022.923879	4967.216367	5087.368080	5011.4
min	12.000000	12.000000	12.000000	12.000000	12.000000	12.000000	12.0
25%	5328.000000	5415.000000	5475.000000	5876.000000	5888.000000	6017.000000	6018.0
50%	7986.000000	8027.000000	8030.000000	8061.000000	8036.000000	8130.000000	8105.5
75%	11874.000000	12028.000000	12047.000000	12107.000000	11997.000000	12462.000000	11932.0
max	23955.000000	23958.000000	23961.000000	23960.000000	23960.000000	23960.000000	23960.0

In [49]:
```
sns.countplot(data = df, x = 'Age')
plt.show()
```



How Gender VS Purchase values are distributed

In [51]:
```
round(df.groupby('Age')['Purchase'].mean() ,2).sort_values(ascending = False)
```

Out[51]:

Age	Purchase
51-55	9534.81
55+	9336.28
36-45	9331.35
26-35	9252.69
46-50	9208.63
18-25	9169.66
0-17	8933.46

dtype: float64

Highest spending comes from 51–55 years (~$9535).

Lowest spending comes from 0–17 years (~$8933).

Overall, older customers tend to spend more.

Applying the Central Limit Theorem (CLT)

Def CLT:

If we repeatedly draw random samples of size n from a population and compute their means, the distribution of sample means will be approximately normal (even if the population itself is skewed).

This allows us to construct confidence intervals for the true mean purchase in each age group.

sample size: 300

In [59]:
```
size_300 = 300
iterations = 1000
```

In [60]:
```
sample_means_0_17 = [df[df['Age']=='0-17']['Purchase'].sample(size_300).mean() for i in range
```

In [62]:
```
sample_means_18_25 = [df[df['Age']=='18-25']['Purchase'].sample(size_300).mean() for i in ran
```

In [64]:
```
sample_means_26_35 = [df[df['Age']=='26-35']['Purchase'].sample(size_300).mean() for i in ran
```

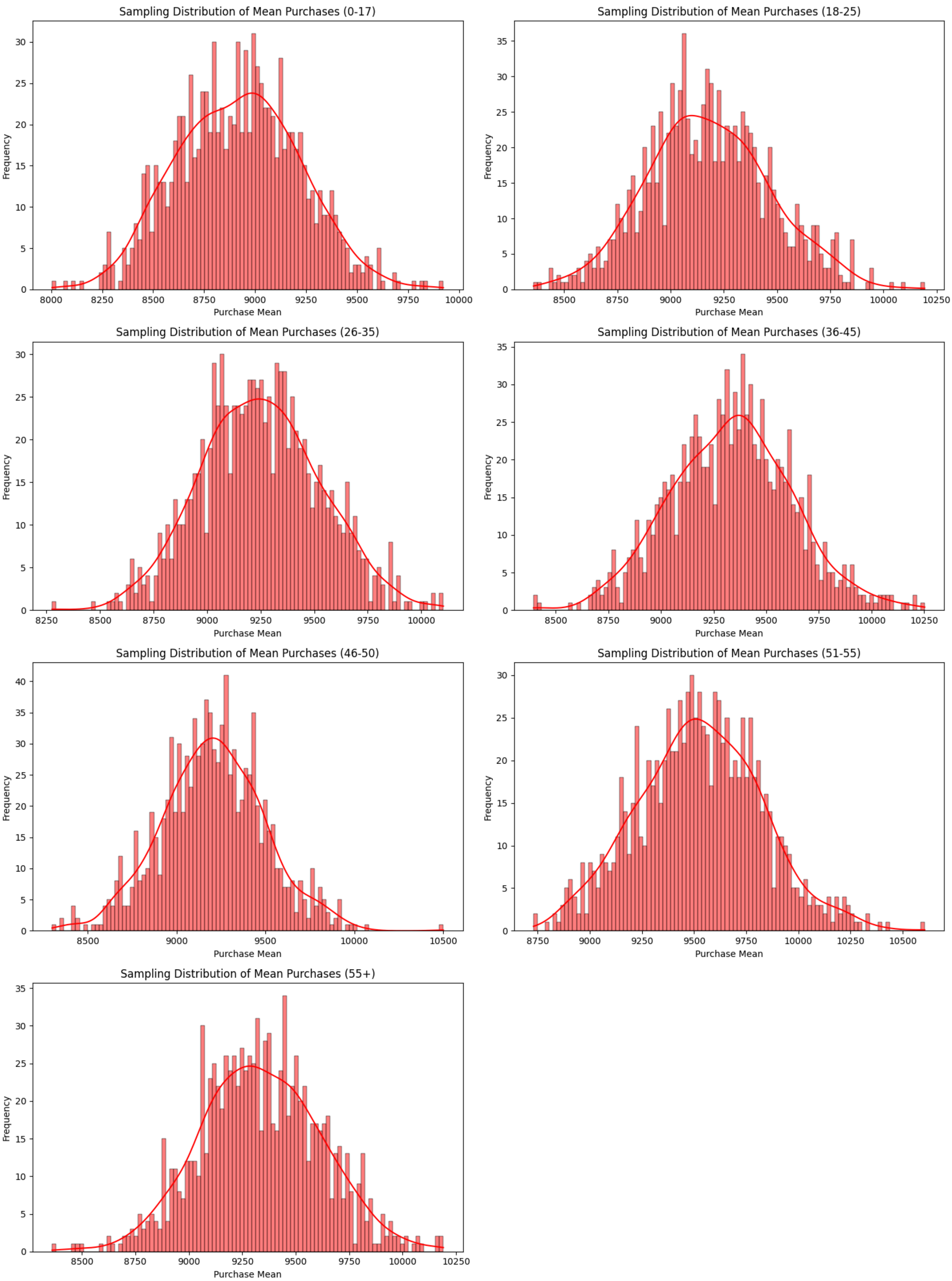
In [65]:
```
sample_means_36_45 = [df[df['Age']=='36-45']['Purchase'].sample(size_300).mean() for i in ran
```

In [67]:
```
sample_means_46_50 = [df[df['Age']=='46-50']['Purchase'].sample(size_300).mean() for i in ran
```

In [ ]:
```
sample_means_51_55 = [df[df['Age']=='51-55']['Purchase'].sample(size_300).mean() for i in ran
```

In [70]:
```
sample_means_55 = [df[df['Age']=='55+']['Purchase'].sample(size_300).mean() for i in range(it
```

In [77]:
```
fig, axes = plt.subplots(4, 2, figsize=(15, 20))  # 4 rows, 2 cols
axes = axes.flatten()

age_groups = {
    '0-17': sample_means_0_17,
    '18-25': sample_means_18_25,
    '26-35': sample_means_26_35,
    '36-45': sample_means_36_45,
    '46-50': sample_means_46_50,
    '51-55': sample_means_51_55,
    '55+': sample_means_55
}


for ax, (age, means) in zip(axes, age_groups.items()):
    sns.histplot(means, kde=True, ax=ax, bins=100, color='r')
    ax.set_title(f"Sampling Distribution of Mean Purchases ({age})", fontsize=12)
    ax.set_xlabel("Purchase Mean")
    ax.set_ylabel("Frequency")


for j in range(len(age_groups), len(axes)):
    fig.delaxes(axes[j])

plt.tight_layout()
plt.show()
```



In [78]:

```
size_3000 = 3000
iterations = 1000
```

In [88]:

```
sample_means_0_17_3000 = [df[df['Age']=='0-17']['Purchase'].sample(size_3000).mean() for i in
```

In [81]:

```
sample_means_18_25_3000 = [df[df['Age']=='18-25']['Purchase'].sample(size_3000).mean() for i
```

In [90]:

```
sample_means_26_35_3000 = [df[df['Age']=='26-35']['Purchase'].sample(size_3000).mean() for i
```

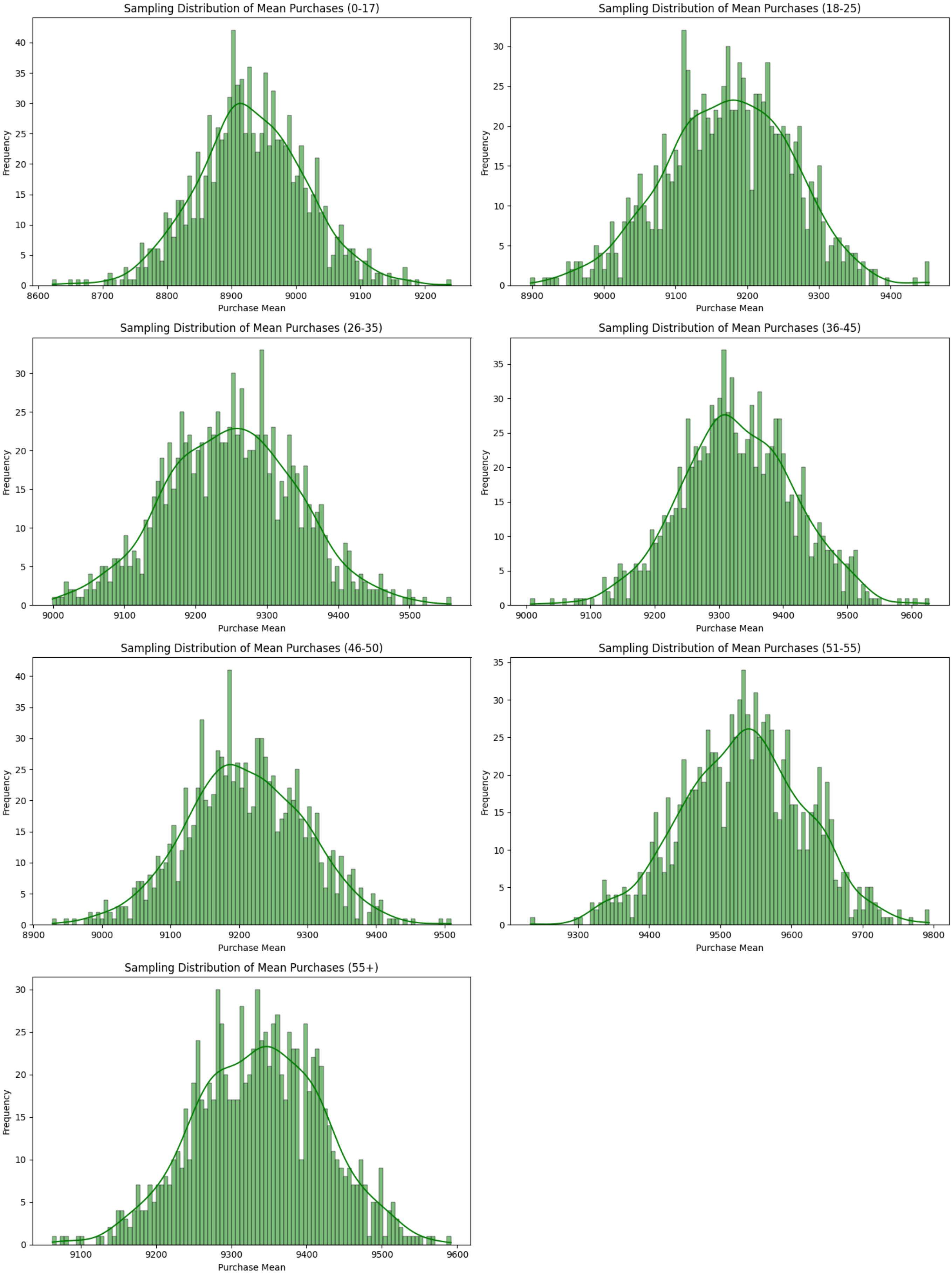
In [82]:
```
sample_means_36_45_3000 = [df[df['Age']=='36-45']['Purchase'].sample(size_3000).mean() for i .
```

In [83]:
```
sample_means_46_50_3000 = [df[df['Age']=='46-50']['Purchase'].sample(size_3000).mean() for i .
```

In [84]:
```
sample_means_51_55_3000 = [df[df['Age']=='51-55']['Purchase'].sample(size_3000).mean() for i .
```

In [85]:
```
sample_means_55_3000 = [df[df['Age']=='55+']['Purchase'].sample(size_3000).mean() for i in ra
```

In [91]:
```
fig, axes = plt.subplots(4, 2, figsize=(15, 20))  # 4 rows, 2 cols
axes = axes.flatten()

age_groups = {
    '0-17': sample_means_0_17_3000,
    '18-25': sample_means_18_25_3000,
    '26-35': sample_means_26_35_3000,
    '36-45': sample_means_36_45_3000,
    '46-50': sample_means_46_50_3000,
    '51-55': sample_means_51_55_3000,
    '55+': sample_means_55_3000
}


for ax, (age, means) in zip(axes, age_groups.items()):
    sns.histplot(means, kde=True, ax=ax, bins=100, color='g')
    ax.set_title(f"Sampling Distribution of Mean Purchases ({age})", fontsize=12)
    ax.set_xlabel("Purchase Mean")
    ax.set_ylabel("Frequency")

for j in range(len(age_groups), len(axes)):
    fig.delaxes(axes[j])

plt.tight_layout()
plt.show()
```



In [94]:

```
# for age group 0-17

alpha = 0.95
dof = len(sample_means_0_17) - 1
stats.t.interval(alpha, dof, loc = np.mean(sample_means_0_17),
                                           scale  = stats.sem(sample_means_0_17))
```

Out[94]:

```
(np.float64(8900.294837691019), np.float64(8937.334028975647))
```

In [95]:
```
# for age group 18-25

alpha = 0.95
dof = len(sample_means_18_25) - 1
stats.t.interval(alpha, dof, loc = np.mean(sample_means_18_25),
                                   scale  = stats.sem(sample_means_18_25))
```

Out[95]: (np.float64(9162.04153455992), np.float64(9198.107812106746))

In [96]:
```
# for age group 26-35

alpha = 0.95
dof = len(sample_means_26_35) - 1
stats.t.interval(alpha, dof, loc = np.mean(sample_means_26_35),
                                   scale  = stats.sem(sample_means_26_35))
```

Out[96]: (np.float64(9232.696689020138), np.float64(9267.271477646527))

In [97]:
```
# for age group 36-45

alpha = 0.95
dof = len(sample_means_36_45) - 1
stats.t.interval(alpha, dof, loc = np.mean(sample_means_36_45),
                                   scale  = stats.sem(sample_means_36_45))
```

Out[97]: (np.float64(9316.220182437633), np.float64(9352.150297562368))

In [98]:
```
# for age group 46-50

alpha = 0.95
dof = len(sample_means_46_50) - 1
stats.t.interval(alpha, dof, loc = np.mean(sample_means_46_50),
                                   scale  = stats.sem(sample_means_46_50))
```

Out[98]: (np.float64(9188.862211429689), np.float64(9224.92486190364))

In [99]:
```
# for age group 51-55

alpha = 0.95
dof = len(sample_means_51_55) - 1
stats.t.interval(alpha, dof, loc = np.mean(sample_means_51_55),
                                   scale  = stats.sem(sample_means_51_55))
```

Out[99]: (np.float64(9507.387225408627), np.float64(9544.206481258036))

In [100…
```
# for age group 55+

alpha = 0.95
dof = len(sample_means_55) - 1
stats.t.interval(alpha, dof, loc = np.mean(sample_means_55),
                                   scale  = stats.sem(sample_means_55))
```

Out[100…: (np.float64(9319.449802060215), np.float64(9354.45049793979))

In [ ]:
```
 sample_means_0_17,
    '18-25': sample_means_18_25,
    '26-35': sample_means_26_35,
    '36-45': sample_means_36_45,
    '46-50': sample_means_46_50,
    '51-55': sample_means_51_55,
    '55+': sample_means_55
```

Non-overlapping Confidence Intervals

Since the intervals do not overlap, this indicates a statistically significant difference in purchase averages between different age groups at the 90% and 95% confidence level.

Insights:

Customer with age groups 51-55 showed higher average purchase values compared other age groups.

Marital status has little to no effect on purchase behavior.

Purchases were almost the same for both married and unmarried customers.

Recommendations:

Age and Gender are the key demographic factors that influence purchases.

Marital status has little to no effect on purchase behavior.

Businesses should focus more on age- and gender-based targeting (e.g., tailoring offers for the 51-55 segment, and optimizing gender-specific promotions) rather than marital status.

In [ ]:

```

```