Walmart Case Study

>> Problem Statement

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? (Assume 50 million customers are male and 50 million are female).

In [1]:
```
!gdown 'https://d2beiqkhq929f0.cloudfront.net/public_assets/assets/000/001/293/original/walmart_data.csv?1641285(
```

```
Downloading...
From: https://d2beiqkhq929f0.cloudfront.net/public_assets/assets/000/001/293/original/walmart_data.csv?164128509
4
To: /Users/girl_intransition/walmart_data.csv?1641285094
100%|████████████████████████████████████████| 23.0M/23.0M [00:23<00:00, 992kB/s]
```

In [2]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from scipy.stats import binom,norm
import random
```

In [3]:
```
import warnings
warnings.filterwarnings('ignore')
```

In [4]:
```
df = pd.read_csv('/Users/girl_intransition/walmart_data.csv?1641285094')
```

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

Out[5]:

	User_ID	Product_ID	Gender	Age	Occupation	City_Category	Stay_In_Current_City_Years	Marital_Status	Product_Category	Purcha
0	1000001	P00069042	F	0-17	10	A	2	0	3	83
1	1000001	P00248942	F	0-17	10	A	2	0	1	152
2	1000001	P00087842	F	0-17	10	A	2	0	12	14
3	1000001	P00085442	F	0-17	10	A	2	0	12	10
4	1000002	P00285442	M	55+	16	C	4+	0	8	79

In [7]:
```
df.shape
```

Out[7]:
```
(550068, 10)
```

In [8]:
```
df.isnull().sum()
```

Out[8]:
```
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 [9]:
```
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
```

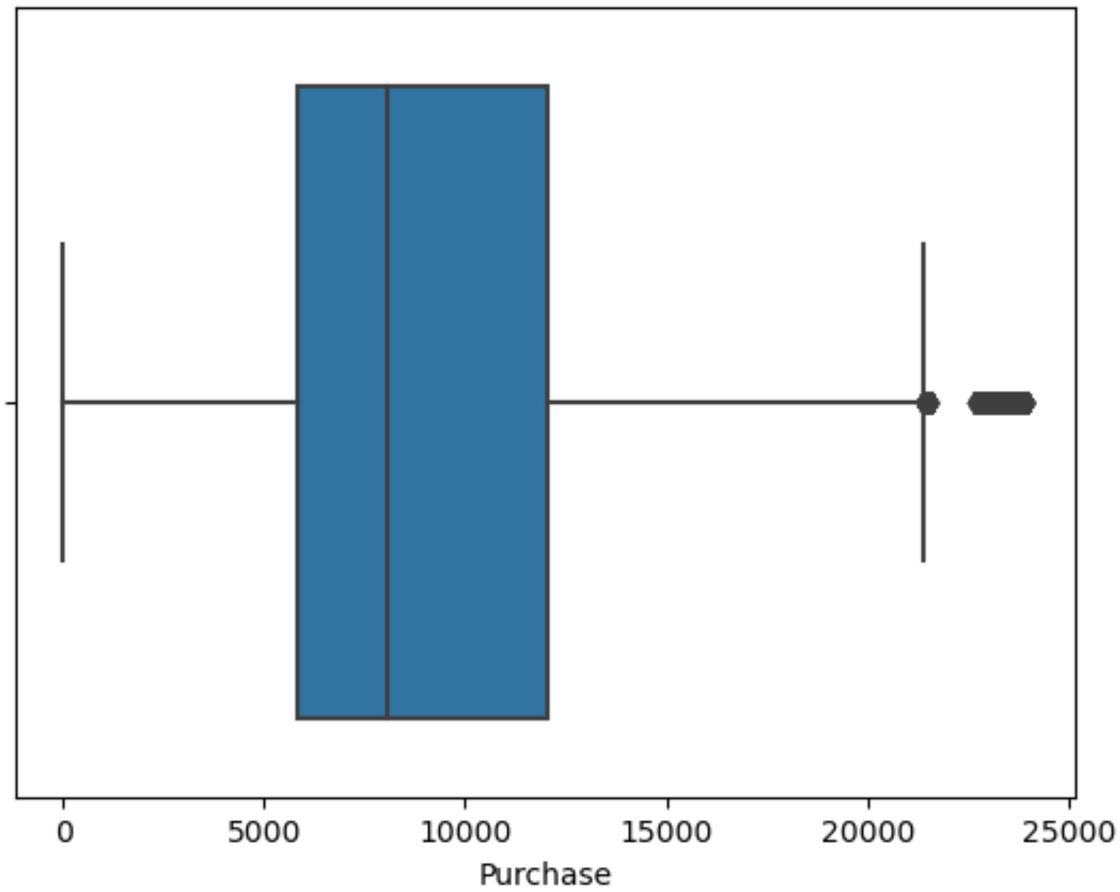
1. Age is of object data type and looking at the data, its divided into categories.
2. City is divided into categories A,B,C
3. There are no null values in the data set and the datatype conversion is also not required.

In [10]:
```
df.describe(include='all')
```

Out[10]:

	User_ID	Product_ID	Gender	Age	Occupation	City_Category	Stay_In_Current_City_Years	Marital_Status	Produc
count	5.500680e+05	550068	550068	550068	550068.000000	550068	550068	550068.000000	5500
unique	NaN	3631	2	7	NaN	3	5	NaN	
top	NaN	P00265242	M	26-35	NaN	B	1	NaN	
freq	NaN	1880	414259	219587	NaN	231173	193821	NaN	
mean	1.003029e+06	NaN	NaN	NaN	8.076707	NaN	NaN	0.409653	
std	1.727592e+03	NaN	NaN	NaN	6.522660	NaN	NaN	0.491770	
min	1.000001e+06	NaN	NaN	NaN	0.000000	NaN	NaN	0.000000	
25%	1.001516e+06	NaN	NaN	NaN	2.000000	NaN	NaN	0.000000	
50%	1.003077e+06	NaN	NaN	NaN	7.000000	NaN	NaN	0.000000	
75%	1.004478e+06	NaN	NaN	NaN	14.000000	NaN	NaN	1.000000	
max	1.006040e+06	NaN	NaN	NaN	20.000000	NaN	NaN	1.000000	

In [11]:
```
# purchase amount distribution

sns.boxplot(data=df,x='Purchase')
plt.show()
```



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

Out[12]:
```
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
```

In [13]:
```
# 1. there is an approximate gap of $1200 between mean and median of purchase amount.
# 2. this implies that we have a lot of outliers.
# 3. IQR = 6231 and upper and lower limit are 18285$ and 0$

print(df.loc[df['Purchase'] > 18285].shape[0])

# There are 44008 transactions whose purchase values are ouliers of this data.
```

```
44008
```

>> Non-Graphic analysis

In [14]:
```
print('No of unique users: ',df['User_ID'].nunique())
```

```
No of unique users:  5891
```

In [15]:
```
df['Age'].unique()
```

Out[15]:
```
array(['0-17', '55+', '26-35', '46-50', '51-55', '36-45', '18-25'],
      dtype=object)
```

In [16]:
```
df.groupby('Age')['User_ID'].nunique()
```

Out[16]:
```
Age
0-17      218
18-25    1069
26-35    2053
36-45    1167
46-50     531
51-55     481
55+       372
Name: User_ID, dtype: int64
```

In [17]:
```
df['Occupation'].unique()
```

Out[17]:
```
array([10, 16, 15,  7, 20,  9,  1, 12, 17,  0,  3,  4, 11,  8, 19,  2, 18,
        5, 14, 13,  6])
```

In [18]:
```
df.groupby('Gender')['User_ID'].nunique()
```

Out[18]:
```
Gender
F    1666
M    4225
Name: User_ID, dtype: int64
```

In [19]:
```
print('1 - Married, 0 - Single')

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

Out[19]:
```
1 - Married, 0 - Single
Marital_Status
0    3417
1    2474
Name: User_ID, dtype: int64
```

In [20]:
```
df['Product_Category'].unique()
```

Out[20]:
```
array([ 3,  1, 12,  8,  5,  4,  2,  6, 14, 11, 13, 15,  7, 16, 18, 10, 17,
        9, 20, 19])
```

--> The product category and city values have been masked for data security reason, so we have masked values instead.

In [21]:
```
print('No of unique product IDs : ',df['Product_ID'].nunique())
```

```
No of unique product IDs :  3631
```

In [22]:
```
df.groupby('City_Category')['User_ID'].nunique()
```

Out[22]:
```
City_Category
A    1045
B    1707
C    3139
Name: User_ID, dtype: int64
```

In [23]:
```
df.groupby('Stay_In_Current_City_Years')['User_ID'].nunique()
```

Out[23]:
```
Stay_In_Current_City_Years
0      772
1     2086
2     1145
3      979
4+     909
Name: User_ID, dtype: int64
```

In [24]:
```
# converting columns to categorical datatype

for col in ['Age','Gender','City_Category','Marital_Status']:
    df[col] = df[col].astype('category')
```

In [25]:
```
df.dtypes
```

Out[25]:
```
User_ID                          int64
Product_ID                      object
Gender                        category
Age                           category
Occupation                       int64
City_Category                 category
Stay_In_Current_City_Years      object
Marital_Status                category
Product_Category                 int64
Purchase                         int64
dtype: object
```

Insights from non-graphic analysis:

1. 55% customers(3231 out of 5891) from our sample dataset have been living in their respective cities for 1/2 years.
2. 53.2% customers live in the masked city C (sounds mysterious).
3. There are 58% unmarried customer in our sample dataset.
4. There are 7 age categories and 72.8% lie in the age range of 18 to 45 and 34.85% customers are from 26 to 35 age group.

>> Graphical analysis

In [26]:
```
# top 5 customers

df.groupby('User_ID')["Purchase"].sum().reset_index().sort_values('Purchase',ascending=False).head()
```

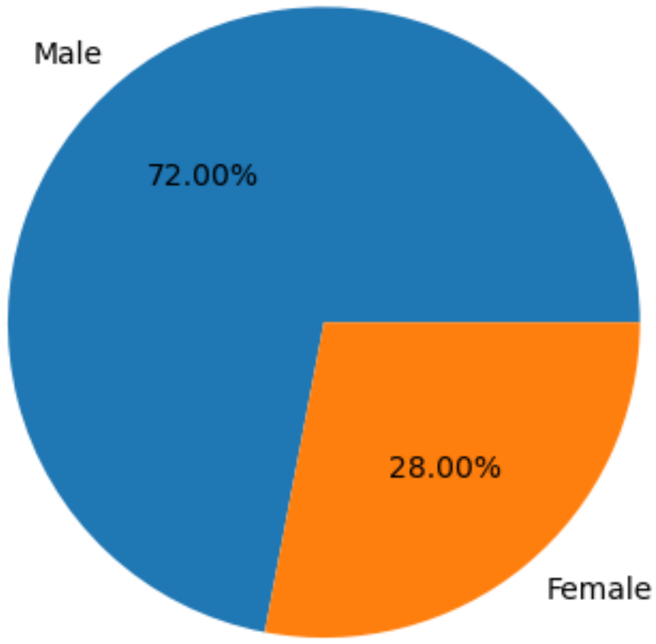
Out[26]:

	User_ID	Purchase
4166	1004277	10536909
1634	1001680	8699596
2831	1002909	7577756
1885	1001941	6817493
416	1000424	6573609

In [ ]:

In [27]:
```
gender_count = df.groupby('Gender')['User_ID'].nunique()
m_percent = (gender_count['M']/(gender_count['M']+gender_count['F'])).round(2)
f_percent = (gender_count['F']/(gender_count['M']+gender_count['F'])).round(2)
labels = ['Male','Female']
```

In [28]:
```
plt.pie([m_percent,f_percent],labels=labels,autopct = '%1.2f%%')
plt.show()
```



In [29]:
```
# which gender spent most

gender_purchase_data = df.groupby('Gender')['Purchase'].sum()
gender_purchase_data
```

Out[29]:
```
Gender
F    1186232642
M    3909580100
Name: Purchase, dtype: int64
```

In [30]:
```
# percentage of purchase amount by a female and male

total_amount = gender_purchase_data.loc['F'] + gender_purchase_data.loc['M']

print("Percentage of female purchase amount",(gender_purchase_data.loc['F']/total_amount).round(2))

print("Percentage of male purchase amount",(gender_purchase_data.loc['M']/total_amount).round(2))
```

```
Percentage of female purchase amount 0.23
Percentage of male purchase amount 0.77
```

In [31]:
```
pd.crosstab(df['Gender'],df['Marital_Status'],normalize='index')
```

Out[31]:

Marital_Status	0	1
Gender		
F	0.580381	0.419619
M	0.593614	0.406386

In [32]:
```
# average money spent by each gender per transaction

df.groupby(['Gender'])['Purchase'].mean().reset_index()
```

Out[32]:

	Gender	Purchase
0	F	8734.565765
1	M	9437.526040

Insights:

1. The percentage of female and male unmarried customers is almost same (58% and 59.3% respectively).
2. Out of the total purchase amount, the female customer contribution is 23% and that of male customer is 77%.
3. there are 28% female customers and 72% male customers.
4. 1004277, 1001680, 1002909, 1001941, 1000424 are the top 5 customers with respect to purchase amount.
5. From the data we have, the average money spent by a male customer is 9437.52 USD and a female customer is 8734.56 USD. This data is only valid for this sample but we cannot infer about male/female purchase aggregates of the population dataset from which this sample has been taken. --> Hence we apply Central Limit theorem in order to identify a range within which the aggreagte purchase values of the male/female customers may lie.

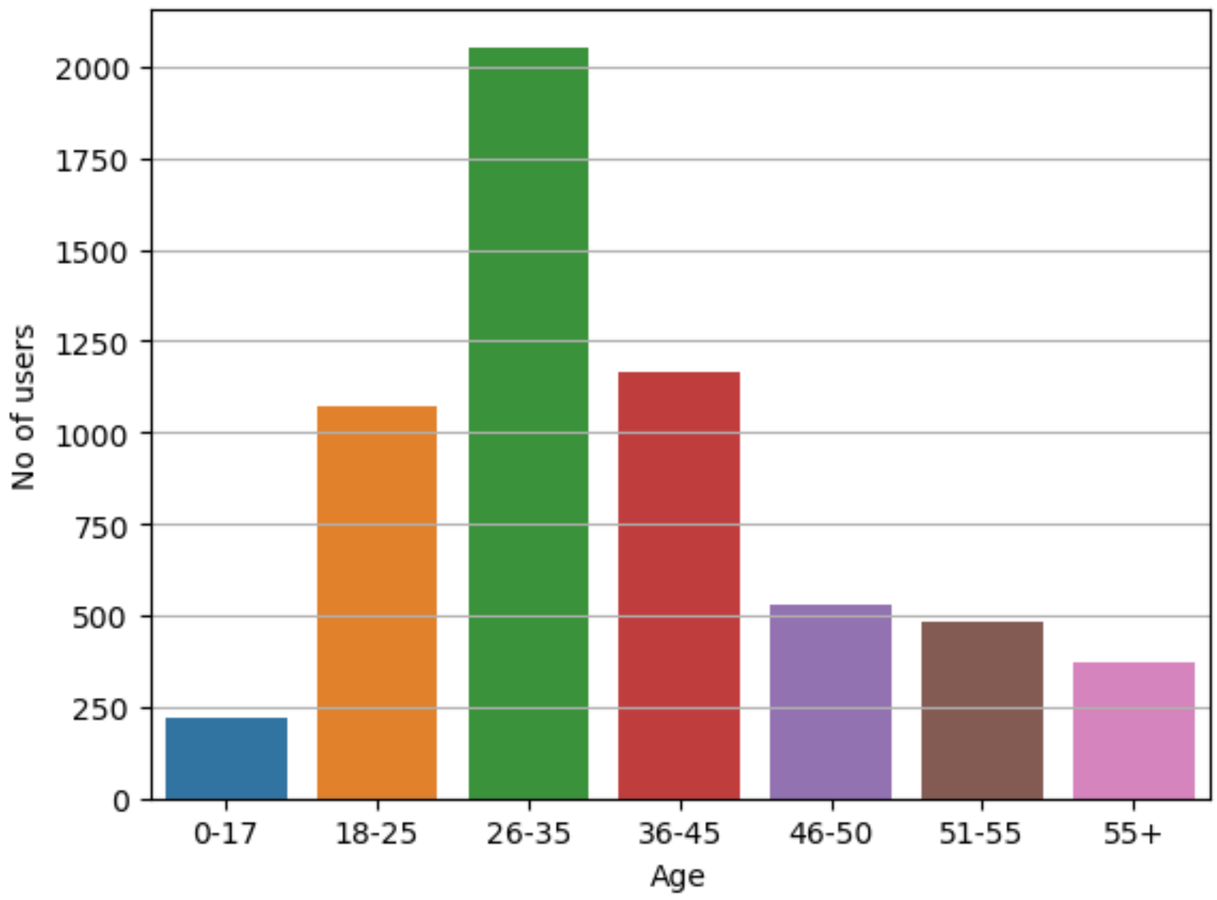
In [34]:
```
df.head()
```

Out[34]:

	User_ID	Product_ID	Gender	Age	Occupation	City_Category	Stay_In_Current_City_Years	Marital_Status	Product_Category	Purcha
0	1000001	P00069042	F	0-17	10	A	2	0	3	83
1	1000001	P00248942	F	0-17	10	A	2	0	1	152
2	1000001	P00087842	F	0-17	10	A	2	0	12	14
3	1000001	P00085442	F	0-17	10	A	2	0	12	10
4	1000002	P00285442	M	55+	16	C	4+	0	8	79

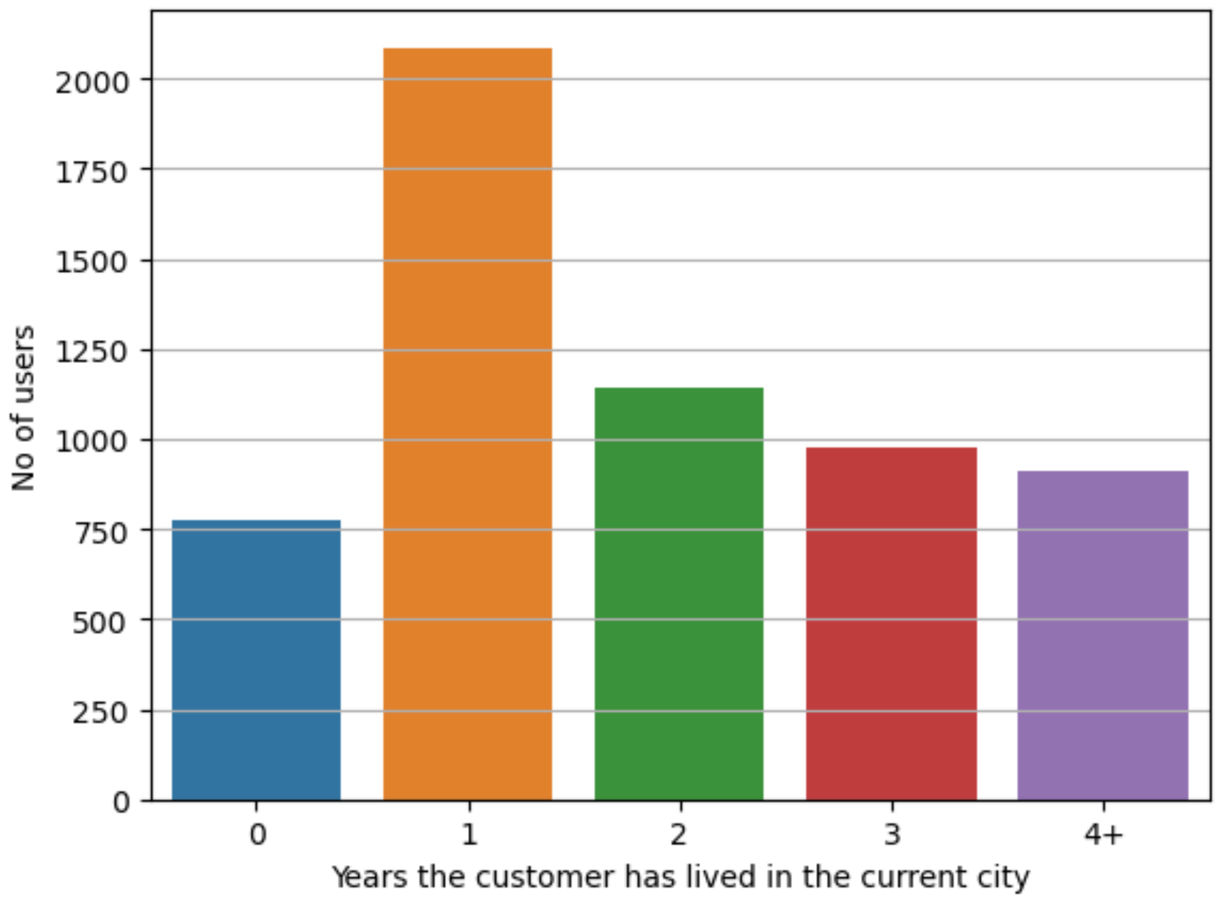
In [35]:

```
age_dist = df.groupby('Age')['User_ID'].nunique().reset_index()

plt.grid()
sns.barplot(data=age_dist,x='Age',y='User_ID')
plt.ylabel('No of users')
plt.show()
```

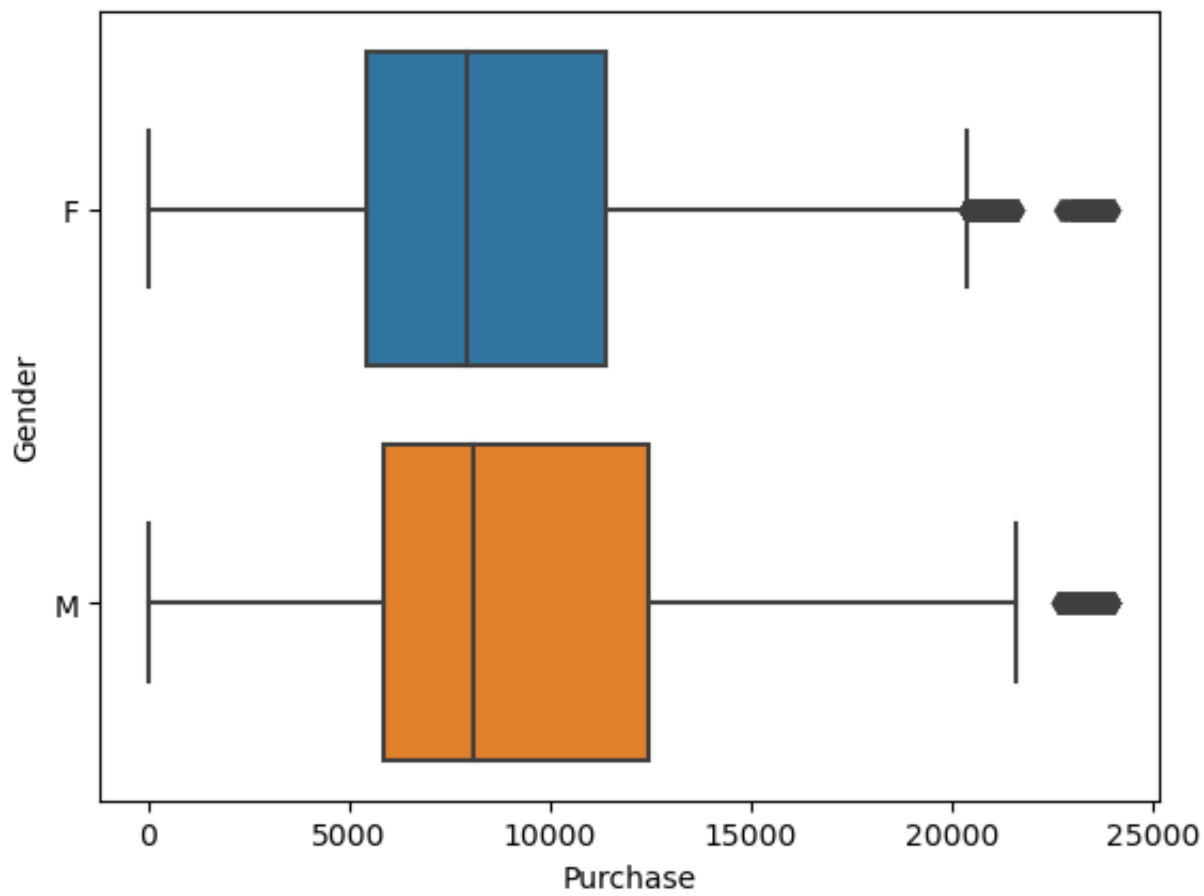


In [36]:

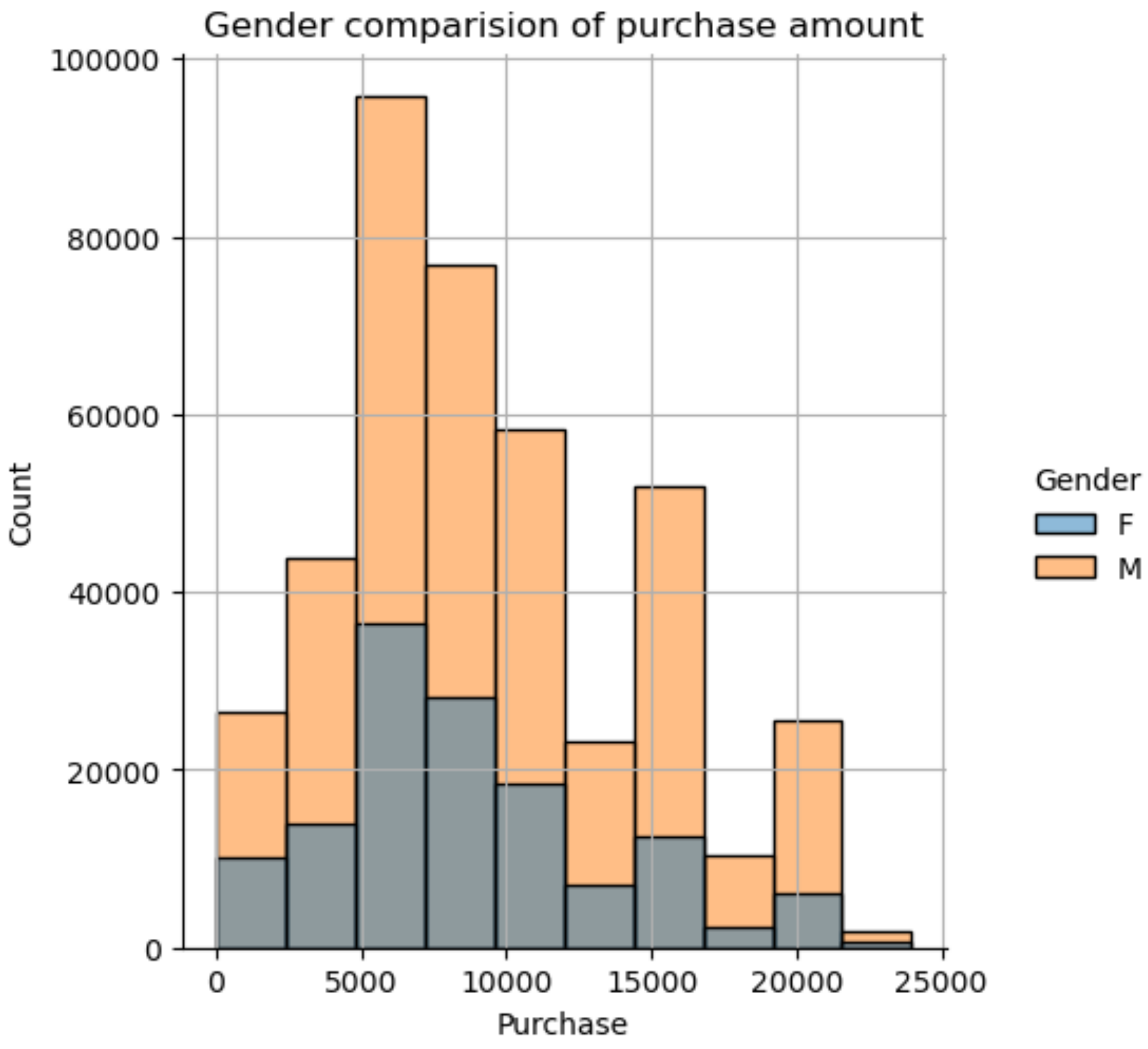
```
city_dur = df.groupby('Stay_In_Current_City_Years')['User_ID'].nunique().reset_index()

plt.grid()
sns.barplot(data=city_dur,x='Stay_In_Current_City_Years',y='User_ID')
plt.xlabel('Years the customer has lived in the current city')
plt.ylabel('No of users')
plt.show()
```



In [37]:
```
# Gender Vs Purchase

# plt.grid()
sns.boxplot(data=df,x='Purchase',y='Gender')
plt.show()
```



In [38]:
```
# Gender Vs Purchase

sns.displot(data = df, x = 'Purchase', bins = 10, hue = 'Gender')
plt.title("Gender comparision of purchase amount")
plt.grid()
plt.show()
```



Insights:

1. Age distribution: We can see that the bar graph for age distribution tells the same story as the non-graphical analysis, where the age group 26-35 constitute the most transactions where as the age groups 18-25, 26-35 and 36-45 make up most of the users on this sample data.

1. Years staying in current city: most customers have been staying in their city for 1 year.

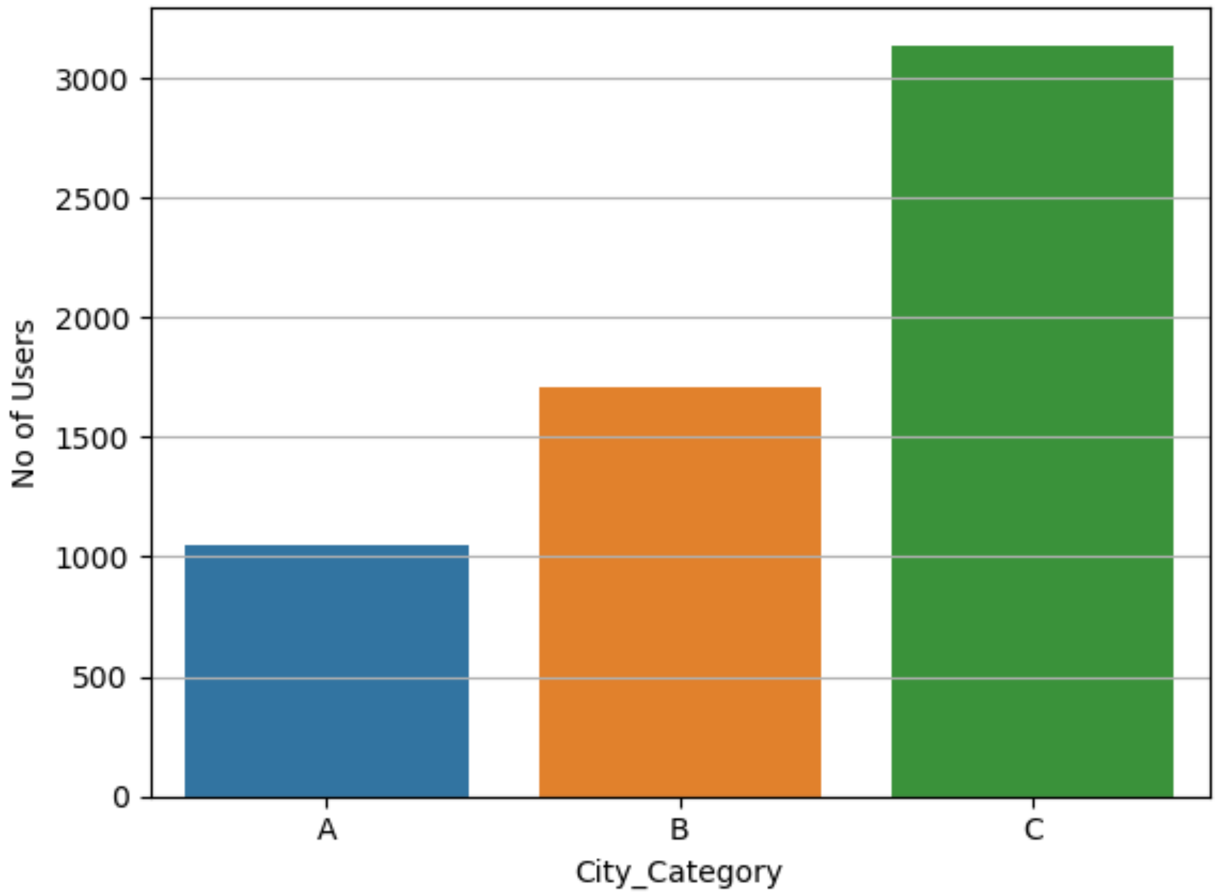
1. We can observe from the graph that the male median is slightly higher than median of purchase amount for female customers.

1. The number of females in each purchase bin is lower than the no of makes in each purchase amount range.

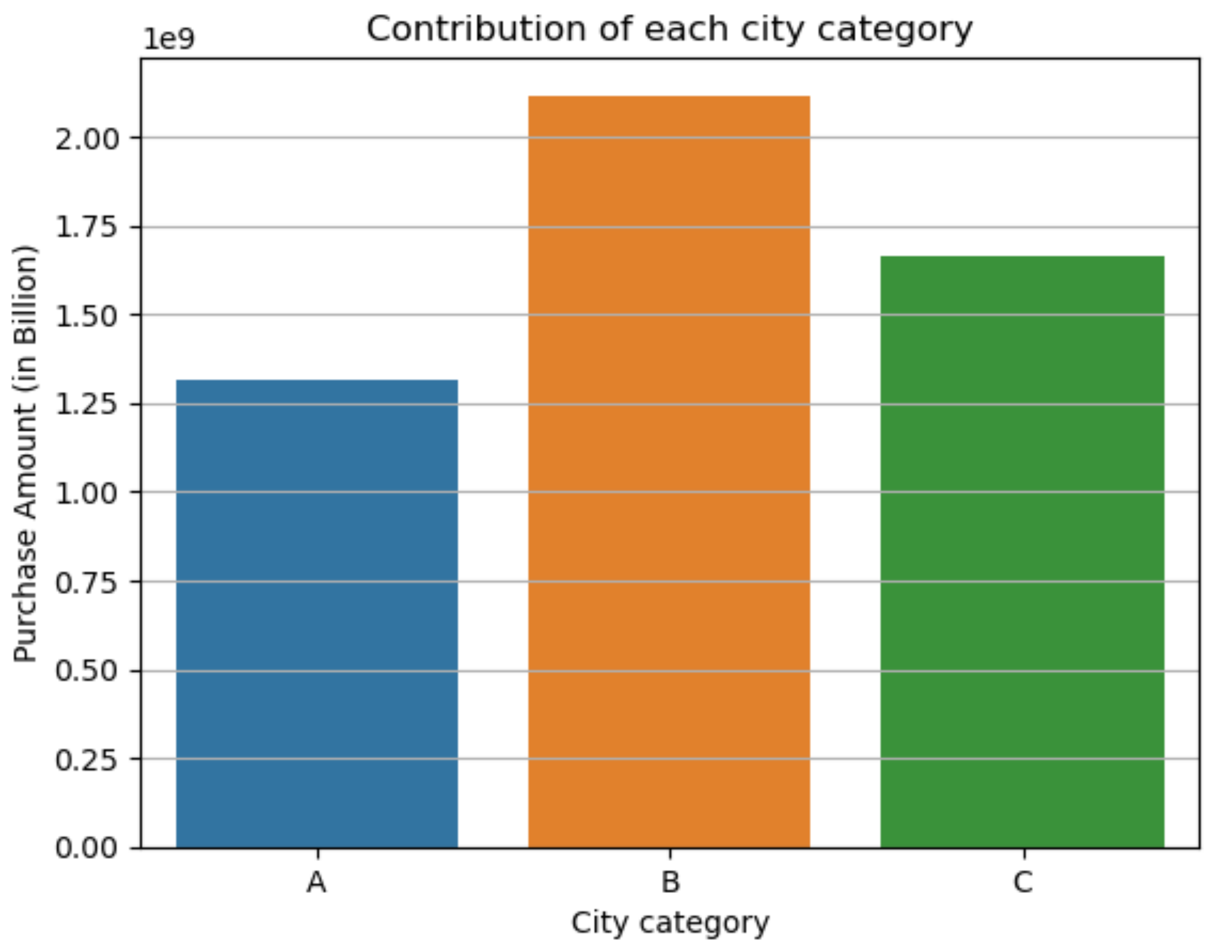
   This could be inferred in two ways:

   a. although they are equal in number in the population data, the purchase amount is lower because they might not be interested in the products walmart has to offer
   b. the change in male and female ratio in the population and sample dataset might attribute to some error.

In [39]:
```
# city category

city = df.groupby('City_Category')['User_ID'].nunique().reset_index()
city

plt.grid()
sns.barplot(data=city,x='City_Category',y='User_ID')
plt.ylabel('No of Users')
plt.show()
```



In [40]:
```
city_purchase = df.groupby('City_Category')['Purchase'].sum().reset_index()

plt.grid()
sns.barplot(data = city_purchase,x = 'City_Category',y='Purchase')
plt.ylabel('Purchase Amount (in Billion)')
plt.xlabel("City category")
plt.title('Contribution of each city category')
plt.show()
```



Insights:

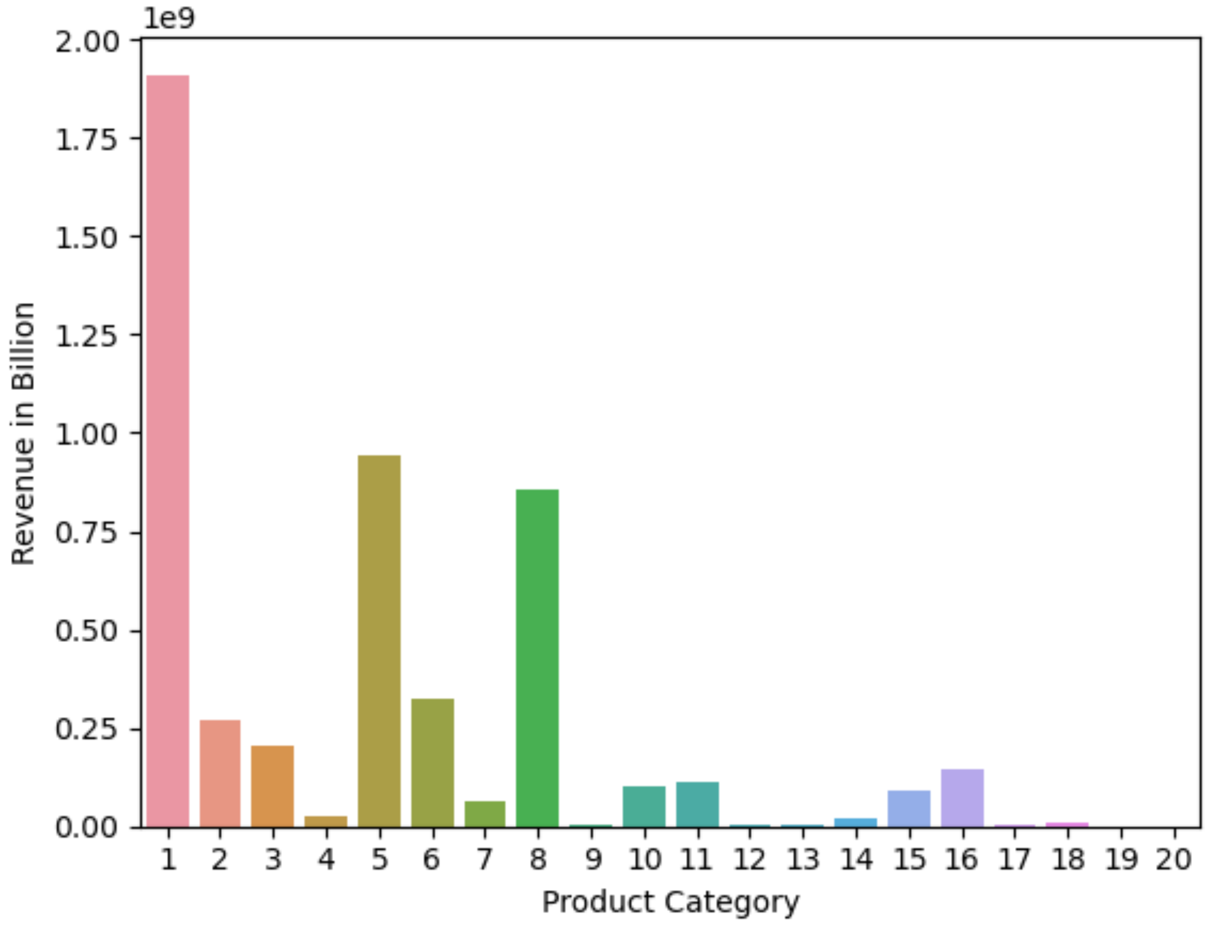
1. From the first graph, No of Users Vs City_Category, We can observe that the city category C has the most no of users followed by B and then A.
2. From the above graph that represents purchase amount vs city category, we can draw that City category B contributes more with respect to purchase amount even though there are more number of users in city category C.
3. City category C follows B and then A.

In [41]:
```
# which product category makes most revenue

prod_revenue = df.groupby(['Product_Category'])['Purchase'].sum().reset_index().sort_values(by = ['Purchase'],asc
prod_revenue.head()
```

Out[41]:

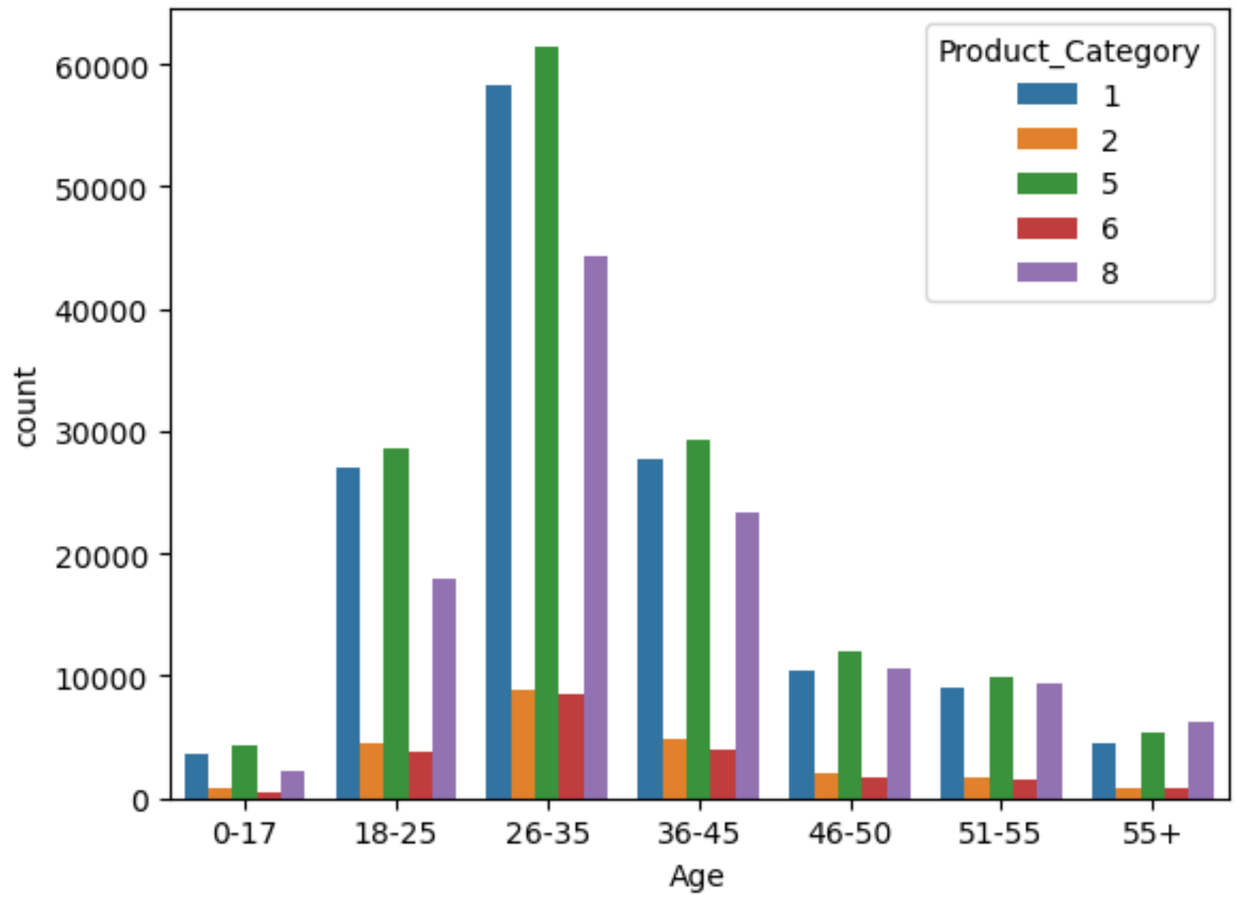
	Product_Category	Purchase
0	1	1910013754
4	5	941835229
7	8	854318799
5	6	324150302
1	2	268516186

In [42]:
```
# Product Category vs Purchase Amount

sns.barplot(data = prod_revenue,x = 'Product_Category', y= 'Purchase')
plt.ylabel("Revenue in Billion")
plt.xlabel('Product Category')
plt.show()
```



In [43]:
```
top_5_prod = df.loc[df['Product_Category'].isin(prod_revenue['Product_Category'][:5])]
```

In [44]:
```
sns.countplot(data=top_5_prod,x='Age',hue='Product_Category')
plt.show()
```

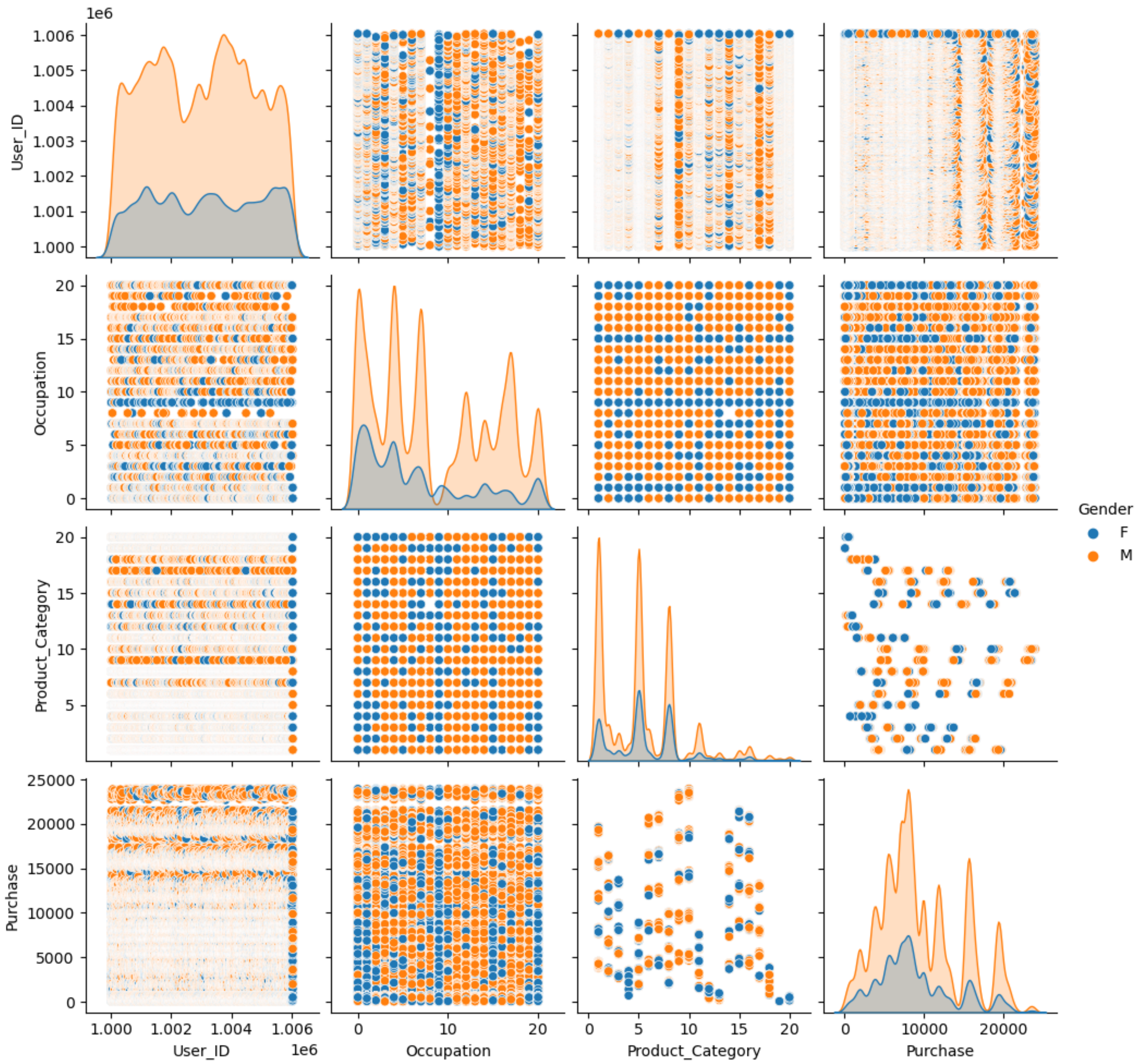


Insights:

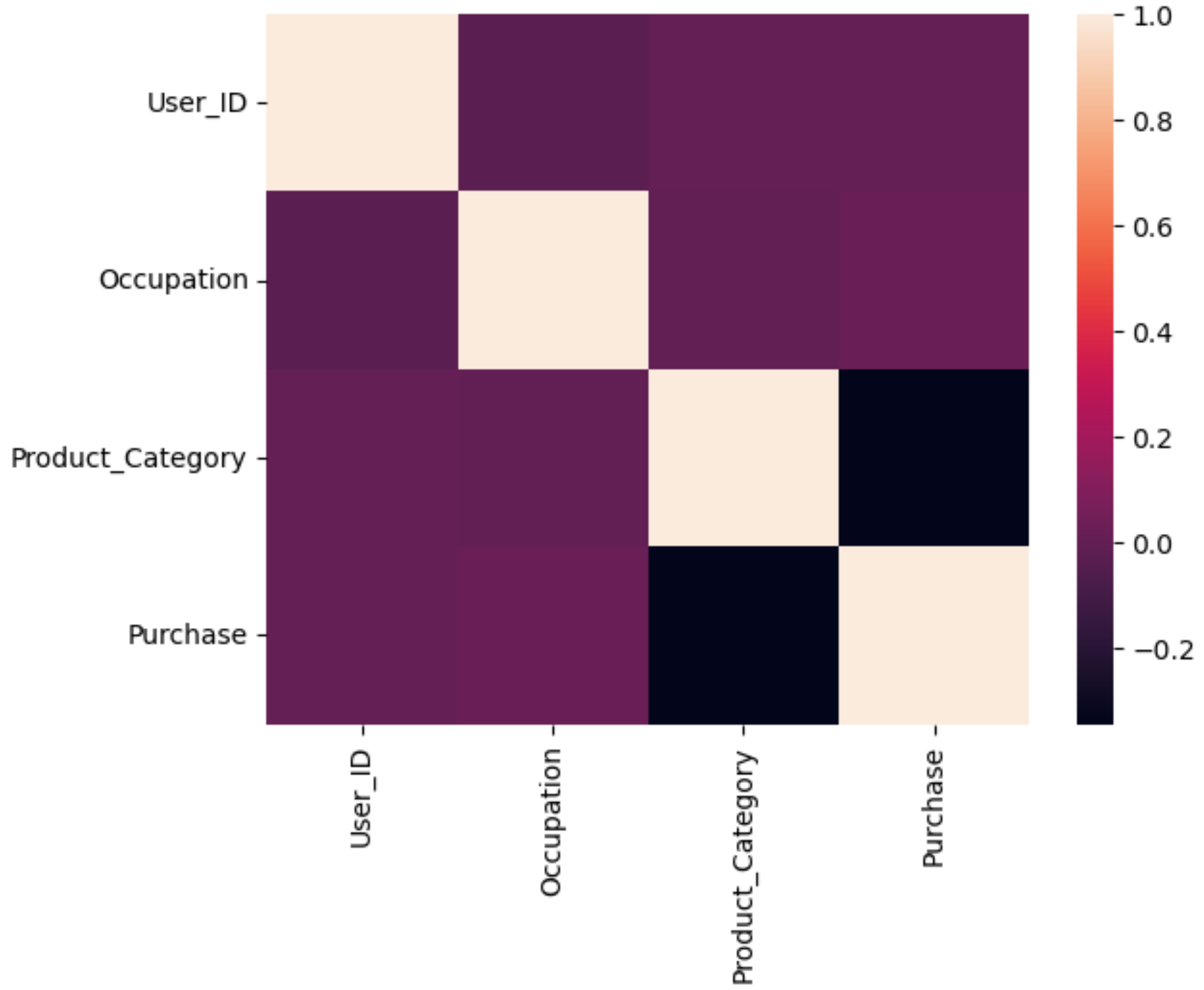
1. Ages 26-35 are contributing most to the revenue generated by the top product categories.
2. Age ranges 18-25 and 36-45 follow with conttribution around 30k USD each.
3. We can also observe that product categories 1 and 5 are the popular among all the age groups followed by 8, no exceptions.

In [45]:
```
sns.pairplot(df,hue='Gender')
```

Out[45]:
```
<seaborn.axisgrid.PairGrid at 0x7fb69c9accd0>
```



In [46]:
```
sns.heatmap(df.corr())
plt.show()
```



In [ ]:

>> Using Central limit theorem

1. In this business case we have the sample data that 500k records/transactions.
2. We cannot conclude based on the Exploratory Data Analysis alone that we have done.
3. Central Limit Theorem states that when we take a sufficiently large sample size, the mean of means will be close to the population mean. We will take the sample sizes 500, 3000 and 10000 and see the difference in results.
4. We will also check the change in range and overlap between two categories by taking different Confidence Intervals (90% , 95%, 99%).

In [47]:
```
df.head()
```

Out[47]:

	User_ID	Product_ID	Gender	Age	Occupation	City_Category	Stay_In_Current_City_Years	Marital_Status	Product_Category	Purcha
0	1000001	P00069042	F	0-17	10	A	2	0	3	83
1	1000001	P00248942	F	0-17	10	A	2	0	1	152
2	1000001	P00087842	F	0-17	10	A	2	0	12	14
3	1000001	P00085442	F	0-17	10	A	2	0	12	10
4	1000002	P00285442	M	55+	16	C	4+	0	8	79

In [48]:
```
print(f"Mean of female customers (from the given dataset) is {df.loc[df['Gender'] == 'F']['Purchase'].mean()}")
print(f"Mean of male customers (from the given dataset) is {df.loc[df['Gender'] == 'M']['Purchase'].mean()}")
```

```
Mean of female customers (from the given dataset) is 8734.565765155476
Mean of male customers (from the given dataset) is 9437.526040472265
```

>> Isolating male and female transactions and calculating respective population standard deviations

In [49]:
```
df_female = df.loc[df['Gender'] == 'F']
df_male = df.loc[df['Gender'] == 'M']

female_popu_std = df_female['Purchase'].std()
male_popu_std = df_male['Purchase'].std()

print(female_popu_std,male_popu_std)
```

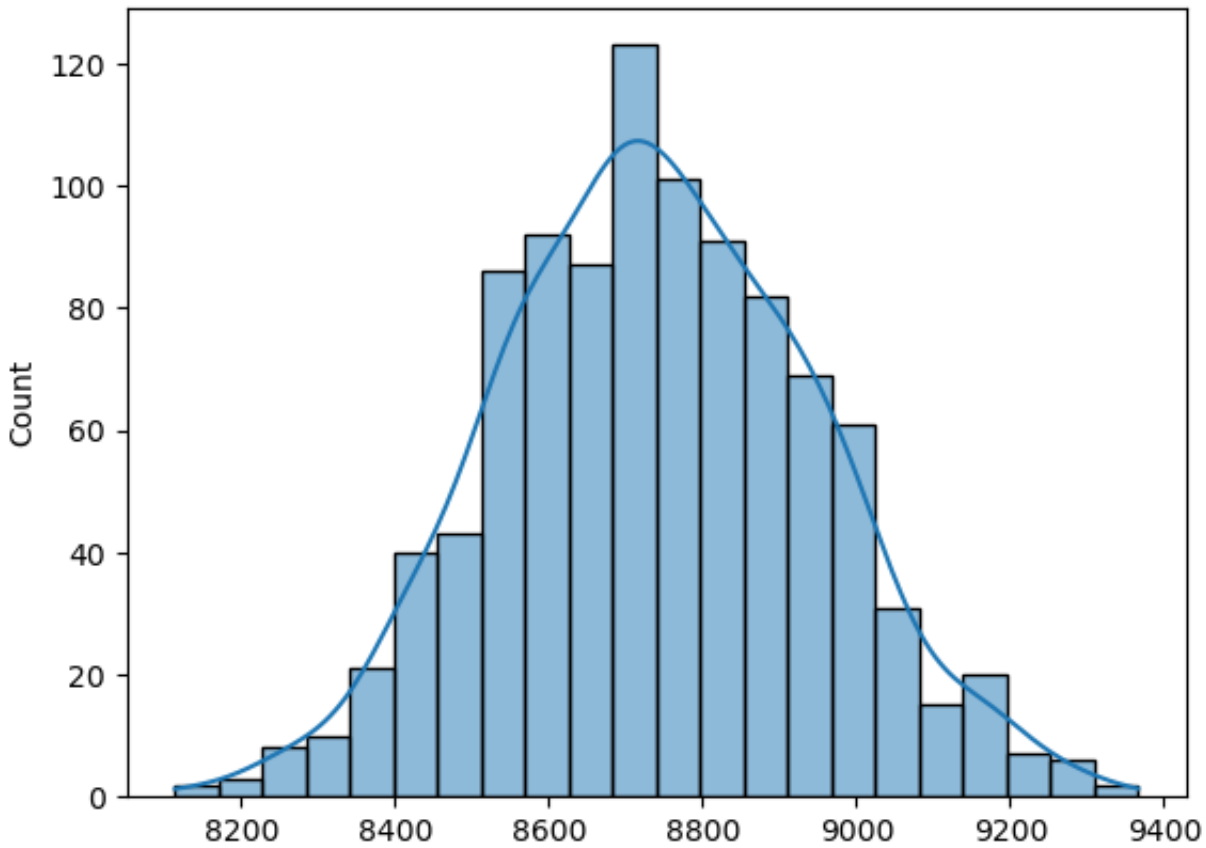
```
4767.233289291444 5092.186209777949
```

Note:

1. We will use central limit theorem for three sample sizes (300,3000 and 10000) to see how the same size impacts the outcomes of central limit theorem
2. If the range overlaps at 95% confidence intervals, we have to find range for 90% CI.

>> For 500 samples

In [50]:
```
sample_size = 500
n = 1000
sample_means_f = np.zeros(n)

for i in range(n):
    mean = np.array(random.sample(list(df_female['Purchase']),sample_size)).mean()
    sample_means_f[i] = mean

sns.histplot(x = sample_means_f,kde=True)
plt.show()
```

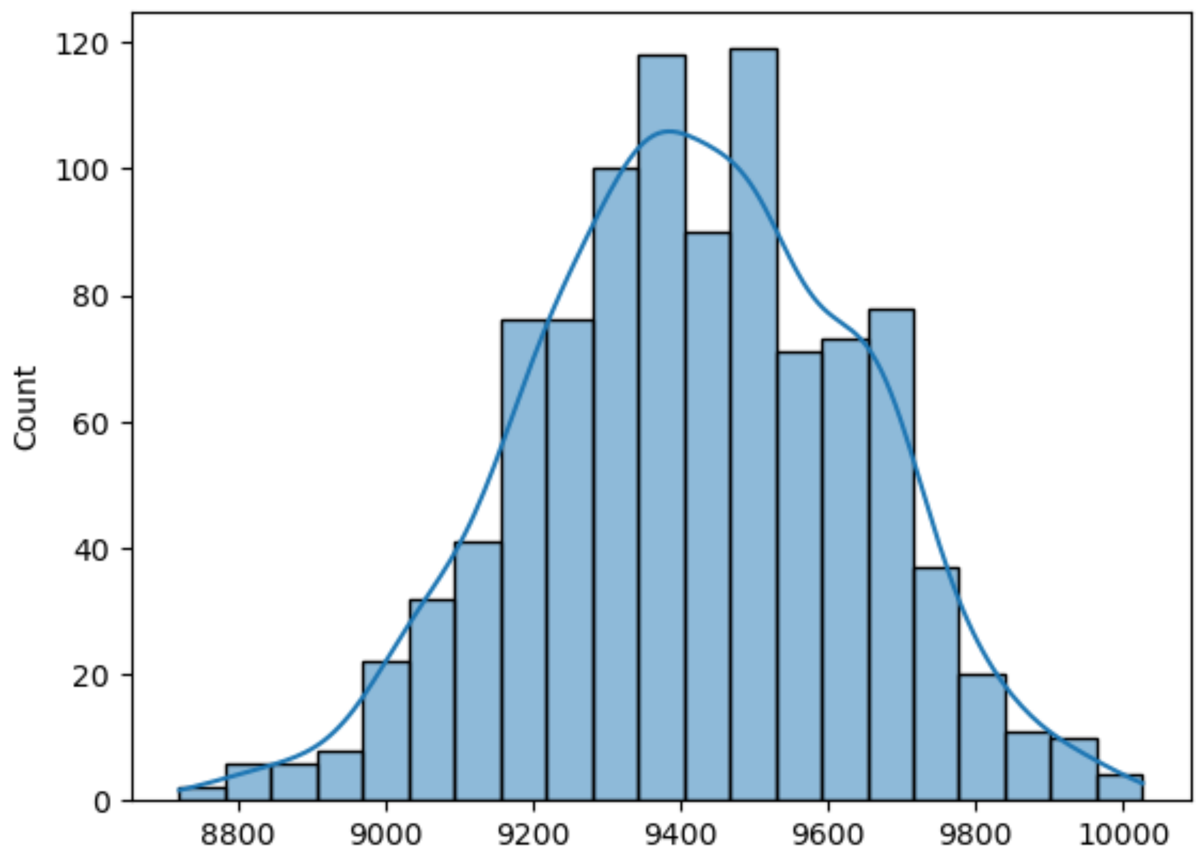


In [51]:
```
# finding confidence interval (95%) for the female population

female_sample_means_mean = sample_means_f.mean()

z = norm.ppf(0.975)

f_ci1 = round(female_sample_means_mean - z*(female_popu_std/(sample_size)**0.5),2)
f_ci2 = round(female_sample_means_mean + z*(female_popu_std/(sample_size)**0.5),2)

print(f"The confidence interval ranges from {f_ci1} to {f_ci2}")
```

```
The confidence interval ranges from 8321.85 to 9157.57
```

In [ ]:
```
# for male customers
```

In [52]:
```
sample_means_m = np.zeros(n)

for i in range(n):
    mean = np.array(random.sample(list(df_male['Purchase']),sample_size)).mean()
    sample_means_m[i] = mean

sns.histplot(x = sample_means_m,kde=True)
plt.show()
```



In [53]:

```
# finding confidence interval (95%) for the male population

male_sample_means_mean = sample_means_m.mean()
z = norm.ppf(0.975)
male_ci1 = round(male_sample_means_mean - z*(male_popu_std/(sample_size)**0.5),2)
male_ci2 = round(male_sample_means_mean + z*(male_popu_std/(sample_size)**0.5),2)

print(f"The confidence interval ranges from {male_ci1} to {male_ci2}")
```

```
The confidence interval ranges from 8970.01 to 9862.69
```

Insights:

At 95% confidence interval:

1. female population mean --> 8326.09 to 9161.81
2. male population mean --> 8995.68 to 9888.36

This implies that we are 95% confident that the mean amount spent by female customers lies in between 8326.09 to 9161.81 and the mean amount spent by male customers lies in between 8995.68 to 9888.36. These values are the mean range for the population.

> It can be noted that the confidence intervals are overlapping, so we have to find range of mean at 90% CI.

In [54]:

```
# finding confidence interval (90%) for the female population

female_sample_means_mean = sample_means_f.mean()

z = norm.ppf(0.95)

f_ci1 = round(female_sample_means_mean - z*(female_popu_std/(sample_size)**0.5),2)
f_ci2 = round(female_sample_means_mean + z*(female_popu_std/(sample_size)**0.5),2)

print(f"The confidence interval ranges from {f_ci1} to {f_ci2}")
```

```
The confidence interval ranges from 8389.03 to 9090.39
```

In [55]:

```
# finding confidence interval (90%) for the male population

male_sample_means_mean = sample_means_m.mean()
z = norm.ppf(0.95)

male_ci1 = round(male_sample_means_mean - z*(male_popu_std/(sample_size)**0.5),2)
male_ci2 = round(male_sample_means_mean + z*(male_popu_std/(sample_size)**0.5),2)

print(f"The confidence interval ranges from {male_ci1} to {male_ci2}")
```

```
The confidence interval ranges from 9041.77 to 9790.93
```

1. We can note that at 90% CI for female customers, the range is 8393.27 to 9094.63
2. We can note that at 90% CI for male customers, the range is 9067.44 to 9816.6
3. There is still an overlap in the CI range for male and female customers, we caanot conclude if one is greater than the other or not.

In [56]:
```
# finding confidence interval (85%) for the female population

female_sample_means_mean = sample_means_f.mean()
z = norm.ppf(0.925)

f_ci1 = round(female_sample_means_mean - z*(female_popu_std/(sample_size)**0.5),2)
f_ci2 = round(female_sample_means_mean + z*(female_popu_std/(sample_size)**0.5),2)

print(f"The confidence interval ranges from {f_ci1} to {f_ci2}")
```

```
The confidence interval ranges from 8432.81 to 9046.62
```

In [57]:
```
# finding confidence interval (85%) for the male population

male_sample_means_mean = sample_means_m.mean()
z = norm.ppf(0.925)

male_ci1 = round(male_sample_means_mean - z*(male_popu_std/(sample_size)**0.5),2)
male_ci2 = round(male_sample_means_mean + z*(male_popu_std/(sample_size)**0.5),2)

print(f"The confidence interval ranges from {male_ci1} to {male_ci2}")
```

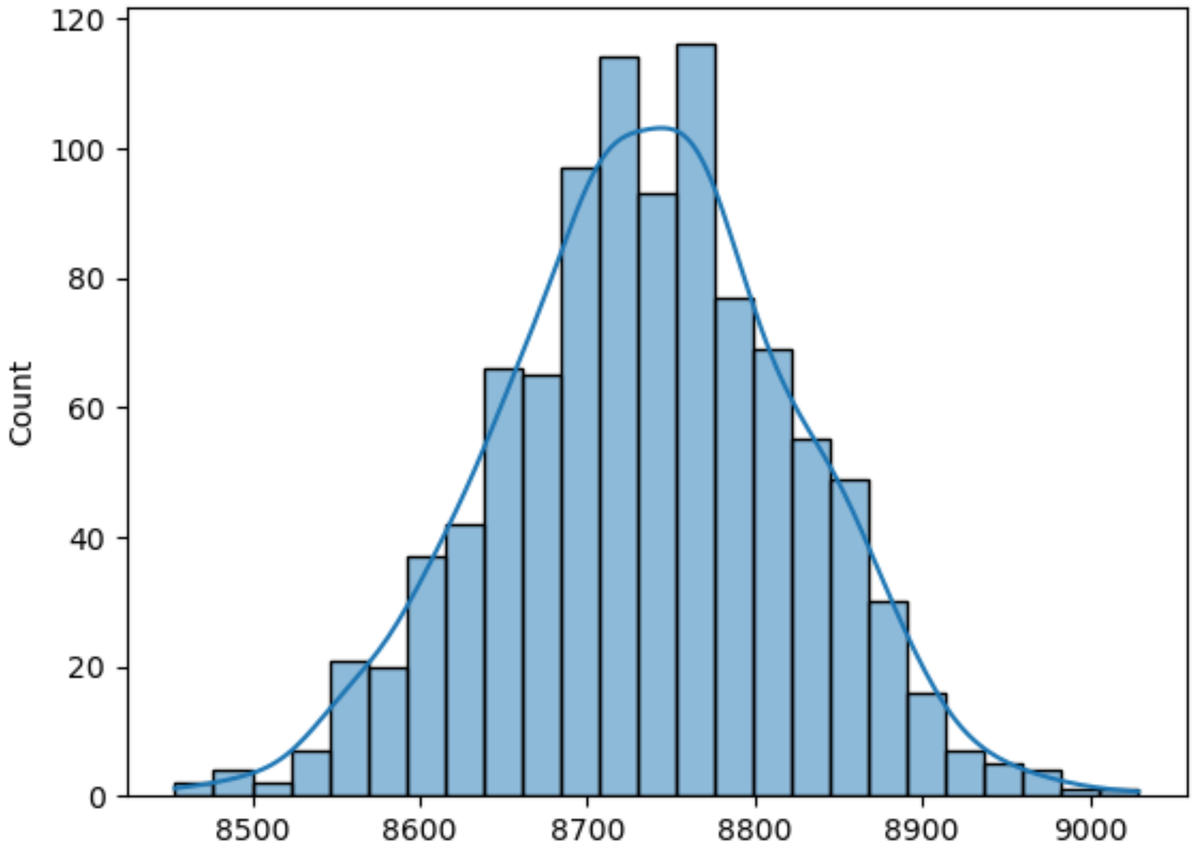
```
The confidence interval ranges from 9088.53 to 9744.17
```

Insights:

1. At 85% CI, the range for mean for female customers is 8437.04 to 9050.85
2. At 85% CI, the range for mean for male customers is 9114.19 to 9769.84
3. We can see that at 85% CI, the ranges dont overlap and we can say with certainity that the female average purchase amount is lower than that of males at 85% CI.

>> For 3000 samples

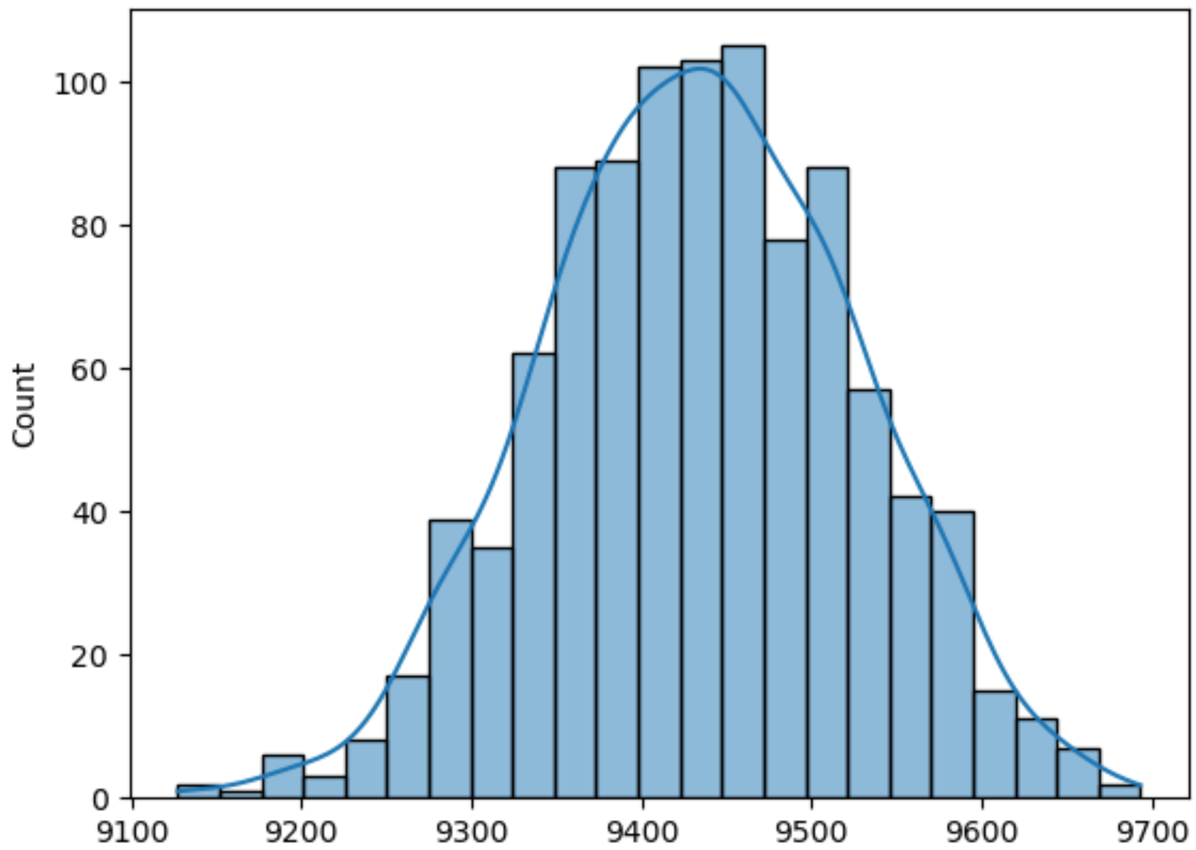
In [58]:
```
sample_size = 3000
n = 1000
sample_means_f = np.zeros(n)

for i in range(n):
    mean = np.array(random.sample(list(df_female['Purchase']),sample_size)).mean()
    sample_means_f[i] = mean

sns.histplot(x = sample_means_f,kde=True)
plt.show()
```



In [59]:
```
# finding confidence interval (99%) for the female population

female_sample_means_mean = sample_means_f.mean()

z = norm.ppf(0.995)

f_ci1 = round(female_sample_means_mean - z*(female_popu_std/(sample_size)**0.5),2)
f_ci2 = round(female_sample_means_mean + z*(female_popu_std/(sample_size)**0.5),2)

print(f"The confidence interval ranges from {f_ci1} to {f_ci2}")
```

```
The confidence interval ranges from 8511.3 to 8959.69
```

In [60]:
```
sample_means_m = np.zeros(n)

for i in range(n):
    mean = np.array(random.sample(list(df_male['Purchase']),sample_size)).mean()
    sample_means_m[i] = mean

sns.histplot(x = sample_means_m,kde=True)
plt.show()
```



In [61]:
```
# finding confidence interval (99%) for the male population

male_sample_means_mean = sample_means_m.mean()
z = norm.ppf(0.995)

male_ci1 = round(male_sample_means_mean - z*(male_popu_std/(sample_size)**0.5),2)
male_ci2 = round(male_sample_means_mean + z*(male_popu_std/(sample_size)**0.5),2)

print(f"The confidence interval ranges from {male_ci1} to {male_ci2}")
```

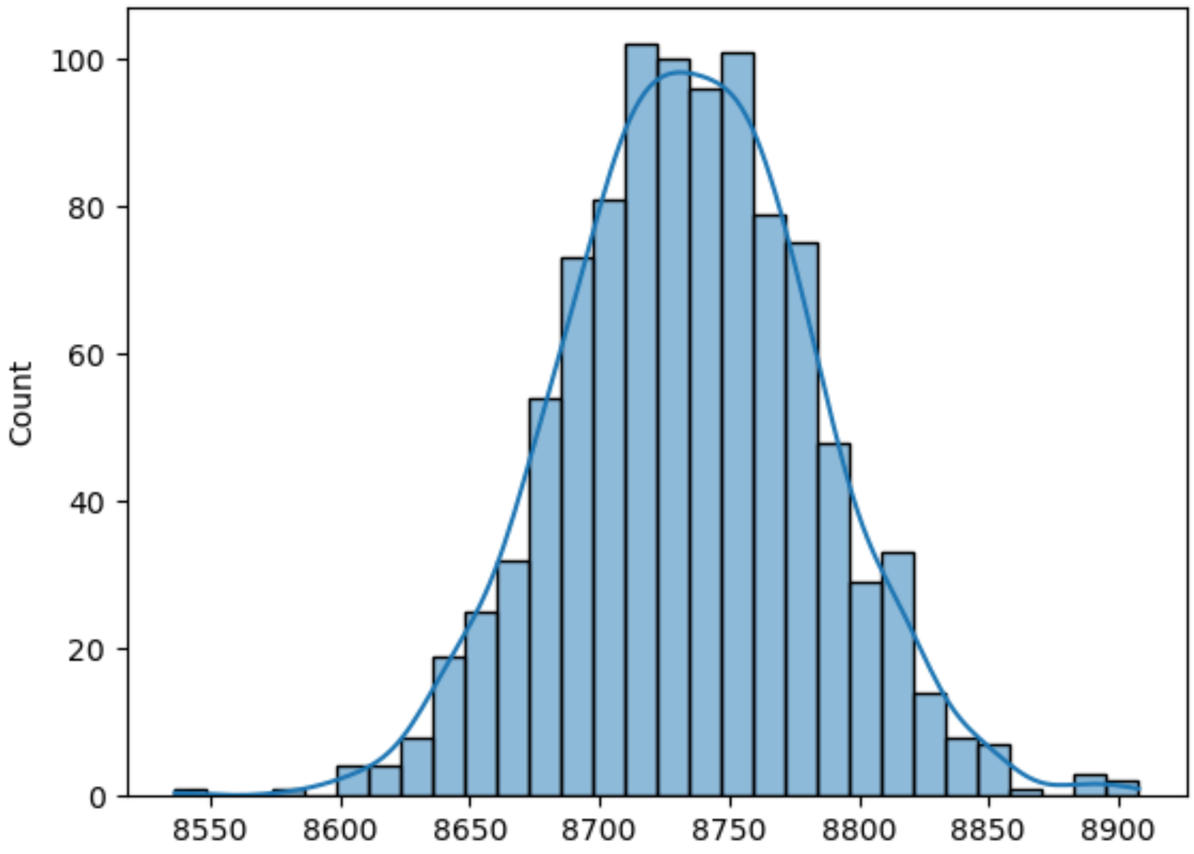
```
The confidence interval ranges from 9194.4 to 9673.35
```

Insights:

1. At 99% CI, the range for female customers is 8512.75 to 8961.14
2. At 99% CI, the range for female customers is 9191.5 to 9670.45
3. For 3000 samples, at 99% CI, we can see that there is no overlap in the ranges and we can conclude with 99% confidence that the average expediture per transaction for male customers is higher than female customers.

>> For 10,000 samples

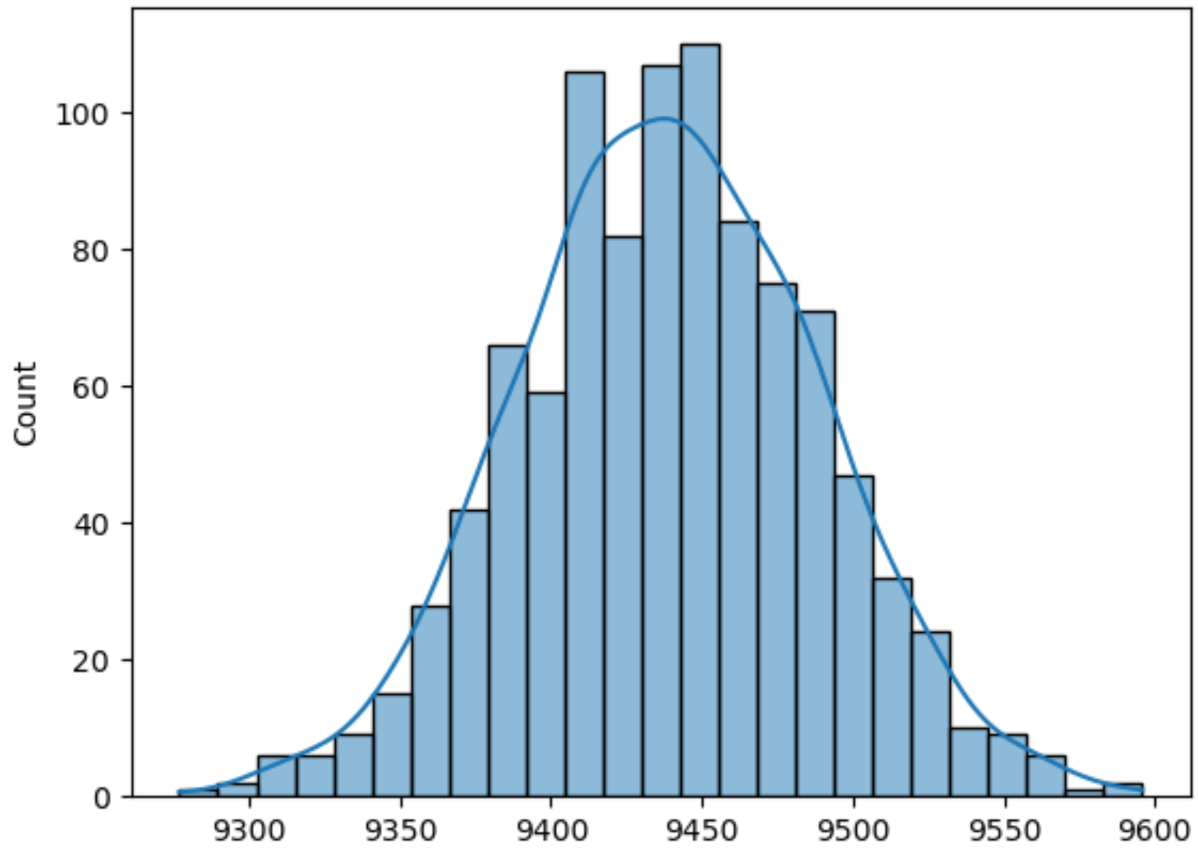
In [62]:
```
sample_size = 10000
n = 1000
sample_means_f = np.zeros(n)

for i in range(n):
    mean = np.array(random.sample(list(df_female['Purchase']),sample_size)).mean()
    sample_means_f[i] = mean

sns.histplot(x = sample_means_f, kde=True)
plt.show()
```



In [63]:
```
# finding confidence interval (99%) for the female population

female_sample_means_mean = sample_means_f.mean()

z = norm.ppf(0.995)

f_ci1 = round(female_sample_means_mean - z*(female_popu_std/(sample_size)**0.5),2)
f_ci2 = round(female_sample_means_mean + z*(female_popu_std/(sample_size)**0.5),2)

print(f"The confidence interval ranges from {f_ci1} to {f_ci2}")
```

```
The confidence interval ranges from 8611.81 to 8857.4
```

In [64]:
```
sample_means_m = np.zeros(n)

for i in range(n):
    mean = np.array(random.sample(list(df_male['Purchase']),sample_size)).mean()
    sample_means_m[i] = mean

sns.histplot(x = sample_means_m,kde=True)
plt.show()
```



In [65]:
```
# finding confidence interval (99%) for the male population

male_sample_means_mean = sample_means_m.mean()
z = norm.ppf(0.995)

male_ci1 = round(male_sample_means_mean - z*(male_popu_std/(sample_size)**0.5),2)
male_ci2 = round(male_sample_means_mean + z*(male_popu_std/(sample_size)**0.5),2)

print(f"The confidence interval ranges from {male_ci1} to {male_ci2}")
```

```
The confidence interval ranges from 9307.47 to 9569.8
```

Insights:

1. At 99% CI, the range for female customers is 8610.3 to 8855.89
2. At 99% CI, the range for female customers is 9305.85 to 9568.18
3. For 10000 samples, at 99% CI, we can see that there is no overlap in the ranges and we can conclude with 99% confidence that the avegrage expediture per transaction for male customers is higher than female customers.

This also indicates that 500 samples were not enough, we needed to take atleast 3000 sample transactions to compare male and female averages.

>> Purchase amount vs Marital status

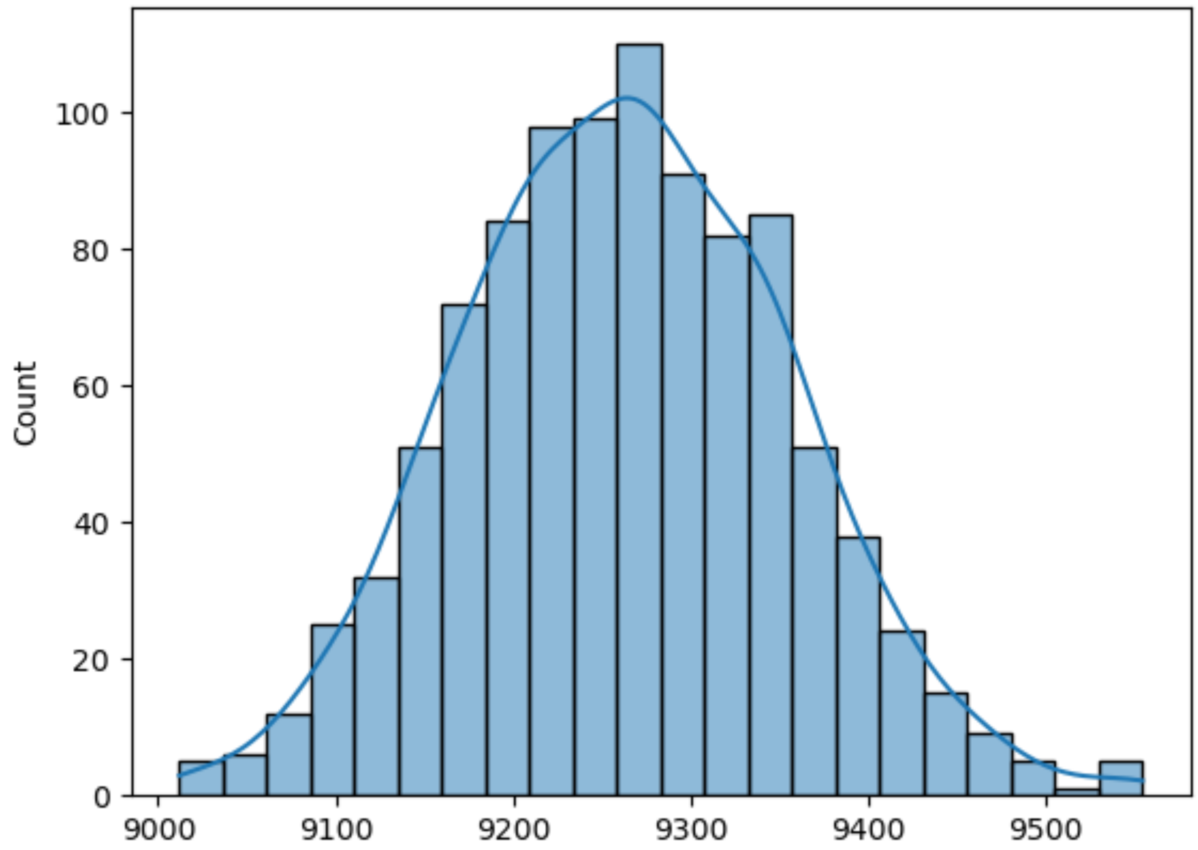
0 - Unmarried (um) 1 - Married (m)

In [66]:
```
unmarried_df = df.loc[df['Marital_Status'] == 0]
married_df = df.loc[df['Marital_Status'] == 1]

marr_pop_std = married_df['Purchase'].std()
unmarr_pop_std = unmarried_df['Purchase'].std()
print(marr_pop_std,unmarr_pop_std)
```

```
5016.89737779313 5027.347858674457
```

We are taking a sample size of 3000 because it is observed that 3000 sample size is giving better results.

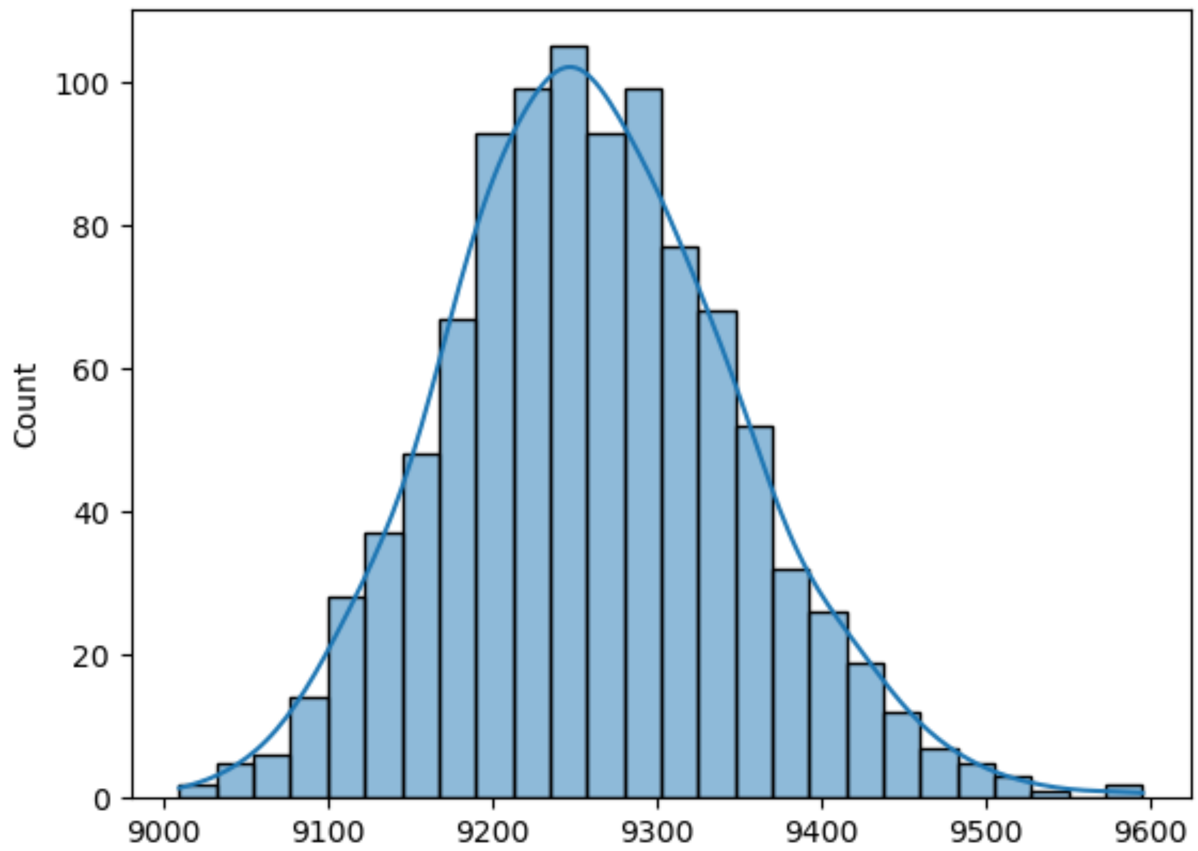
In [67]:
```
sample_size = 3000
n = 1000
um_sample_means = np.zeros(n)

for i in range(n):
    mean = np.array(random.sample(list(unmarried_df['Purchase']),sample_size)).mean()
    um_sample_means[i] = mean

sns.histplot(x = um_sample_means,kde=True)
plt.show()
```



In [68]:
```
# finding confidence interval (95%) for the unmarried population

um_sample_means_mean = um_sample_means.mean()
z = norm.ppf(0.95)

um_ci1 = round(um_sample_means_mean - z*(unmarr_pop_std/(sample_size)**0.5),2)
um_ci2 = round(um_sample_means_mean + z*(unmarr_pop_std/(sample_size)**0.5),2)

print(f"The confidence interval ranges from {um_ci1} to {um_ci2}")
```

```
The confidence interval ranges from 9112.43 to 9414.38
```

In [69]:
```
m_sample_means = np.zeros(n)

for i in range(n):
    mean = np.array(random.sample(list(married_df['Purchase']),sample_size)).mean()
    m_sample_means[i] = mean

sns.histplot(x = m_sample_means,kde=True)
plt.show()
```



In [70]:
```
# finding confidence interval (95%) for the unmarried population

m_sample_means_mean = m_sample_means.mean()
z = norm.ppf(0.975)

m_ci1 = round(m_sample_means_mean - z*(marr_pop_std/(sample_size)**0.5),2)
m_ci2 = round(m_sample_means_mean + z*(marr_pop_std/(sample_size)**0.5),2)

print(f"The confidence interval ranges from {m_ci1} to {m_ci2}")
```

```
The confidence interval ranges from 9081.72 to 9440.77
```

Insights:

1. at 95% CI, the ranges of mean purchase value per transaction for unmarried customers is 9166.72 to 9363.79
2. at 95% CI, the ranges of mean purchase value per transaction for married customers is 9155.57 to 9352.23
3. We can see that the interval almost completely overlaps, which means that both married and unmarried customers' average purchase value is very close/ almost same.

>>> Purchase amount vs Age

In [71]:
```
df['Age'].value_counts()
```

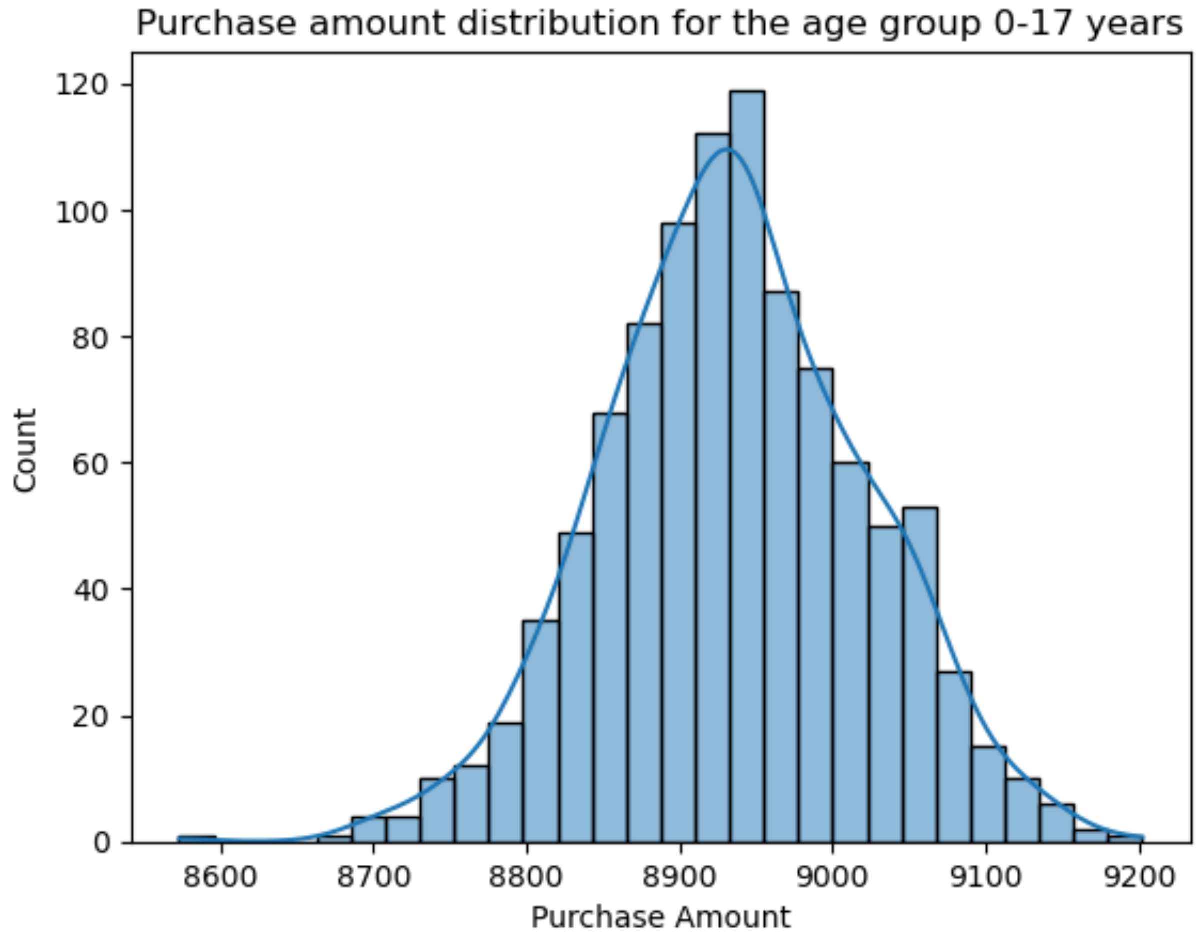
Out[71]:
```
26-35    219587
36-45    110013
18-25     99660
46-50     45701
51-55     38501
55+       21504
0-17      15102
Name: Age, dtype: int64
```

In [72]:
```
sample_size,n = 3000,1000
```

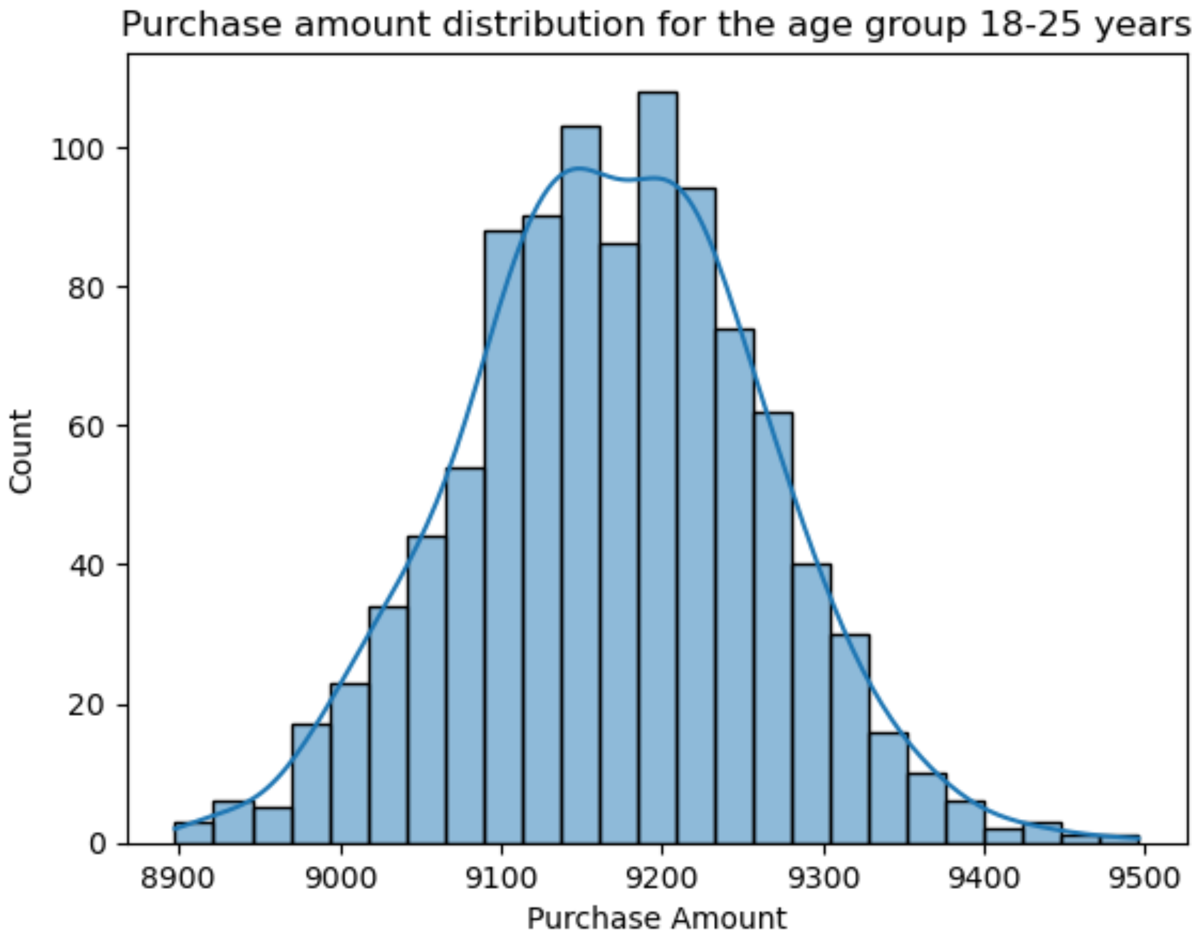
In [73]:
```
sample_means_below17 = [np.array(random.sample(list(df.loc[df['Age'] == '0-17']['Purchase']),sample_size)).mean()
sample_means_18to25 = [np.array(random.sample(list(df.loc[df['Age'] == '18-25']['Purchase']),sample_size)).mean()
sample_means_26to35 = [np.array(random.sample(list(df.loc[df['Age'] == '26-35']['Purchase']),sample_size)).mean()
sample_means_36to45 = [np.array(random.sample(list(df.loc[df['Age'] == '36-45']['Purchase']),sample_size)).mean()
sample_means_46to50 = [np.array(random.sample(list(df.loc[df['Age'] == '46-50']['Purchase']),sample_size)).mean()
sample_means_51to55 = [np.array(random.sample(list(df.loc[df['Age'] == '51-55']['Purchase']),sample_size)).mean()
sample_means_above55 = [np.array(random.sample(list(df.loc[df['Age'] == '55+']['Purchase']),sample_size)).mean()
```

In [74]:
```
# standard deviation values for all age categories

std_pop_below17  = df.loc[df['Age'] == '0-17']['Purchase'].std()
std_pop_18to25 = df.loc[df['Age'] == '18-25']['Purchase'].std()
std_pop_26to35 = df.loc[df['Age'] == '26-35']['Purchase'].std()
std_pop_36to45 = df.loc[df['Age'] == '36-45']['Purchase'].std()
std_pop_46to50 = df.loc[df['Age'] == '46-50']['Purchase'].std()
std_pop_51to55 = df.loc[df['Age'] == '51-55']['Purchase'].std()
std_pop_above55 = df.loc[df['Age'] == '55+']['Purchase'].std()
```

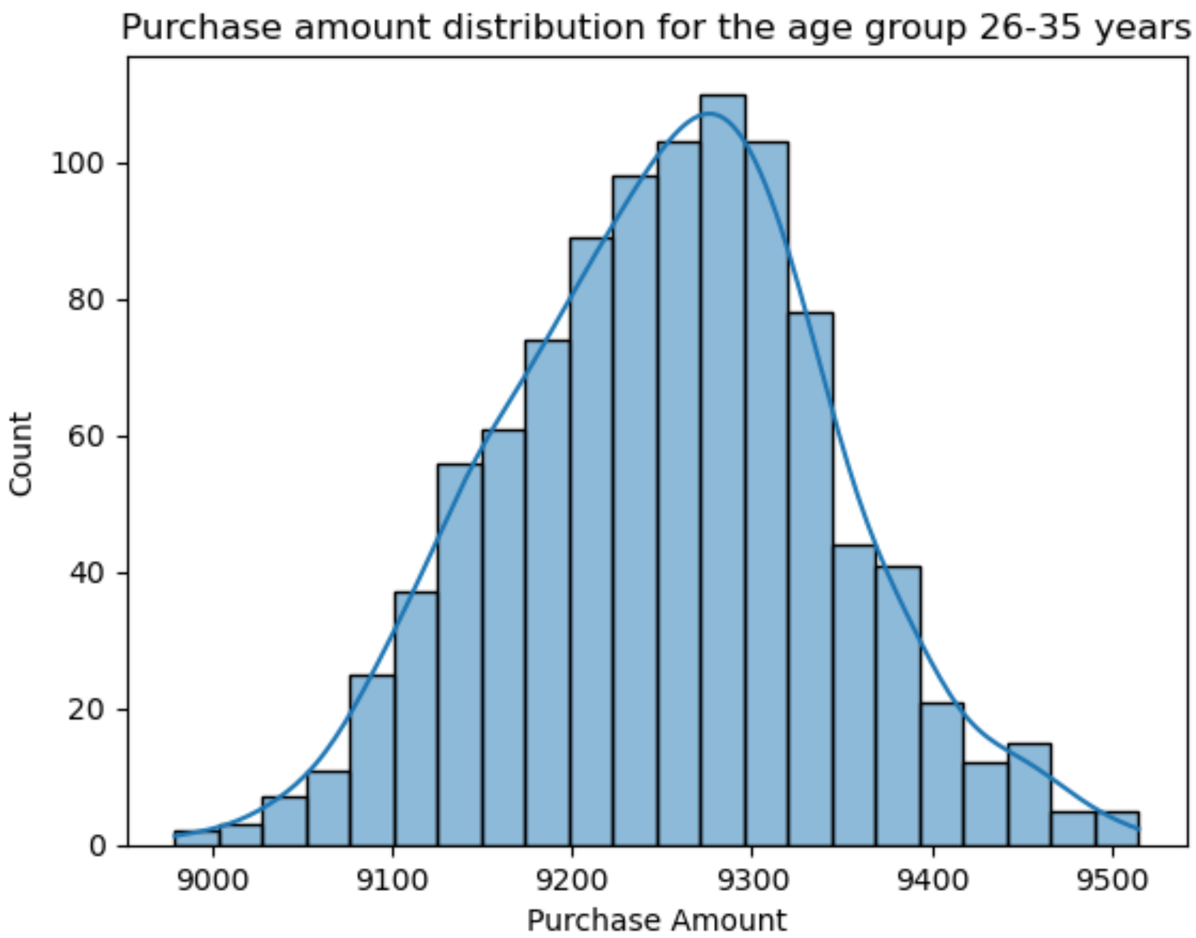
In [75]:
```
sns.histplot(x=sample_means_below17,kde=True)
plt.xlabel('Purchase Amount')
plt.title('Purchase amount distribution for the age group 0-17 years')
plt.show()
```



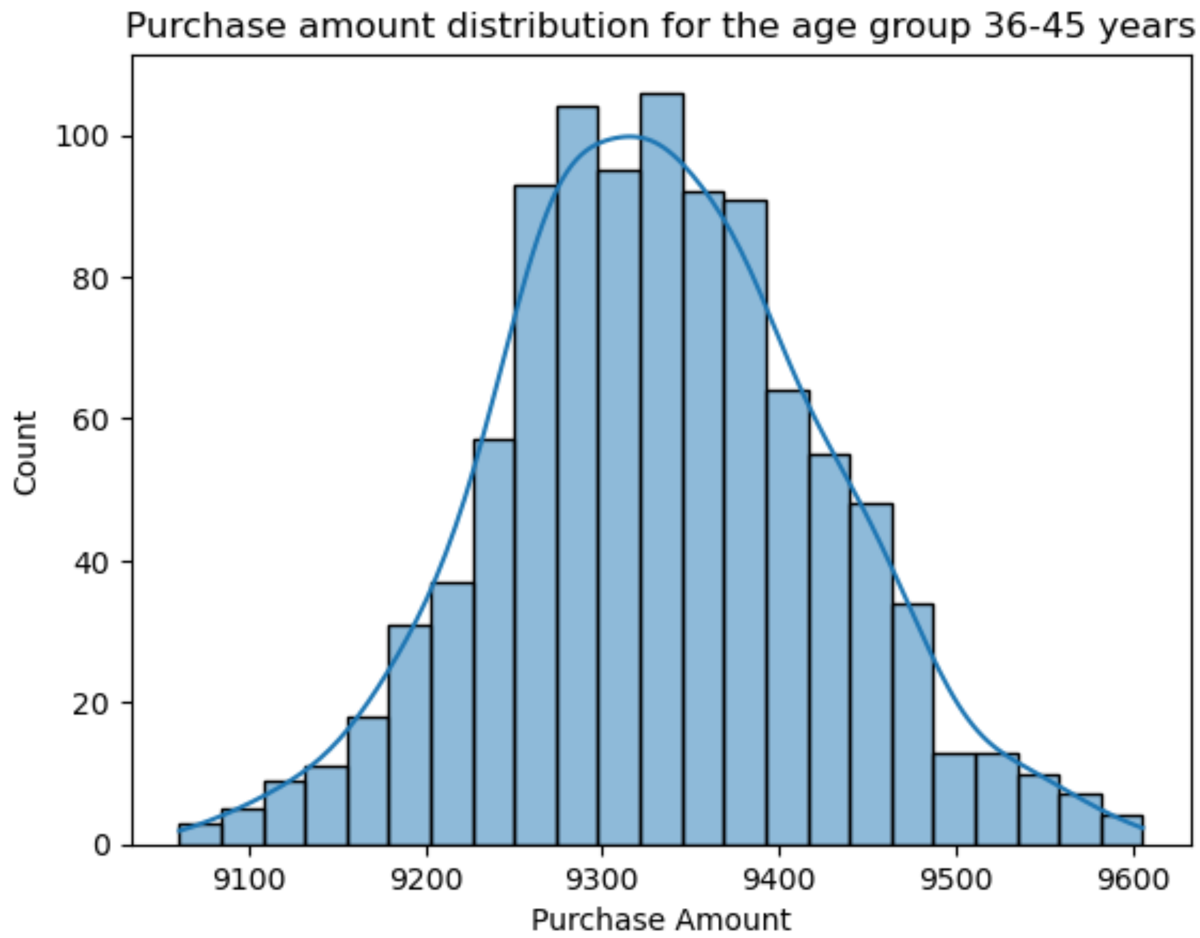
In [76]:
```
sns.histplot(x=sample_means_18to25,kde=True)
plt.xlabel('Purchase Amount')
plt.title('Purchase amount distribution for the age group 18-25 years')
plt.show()
```



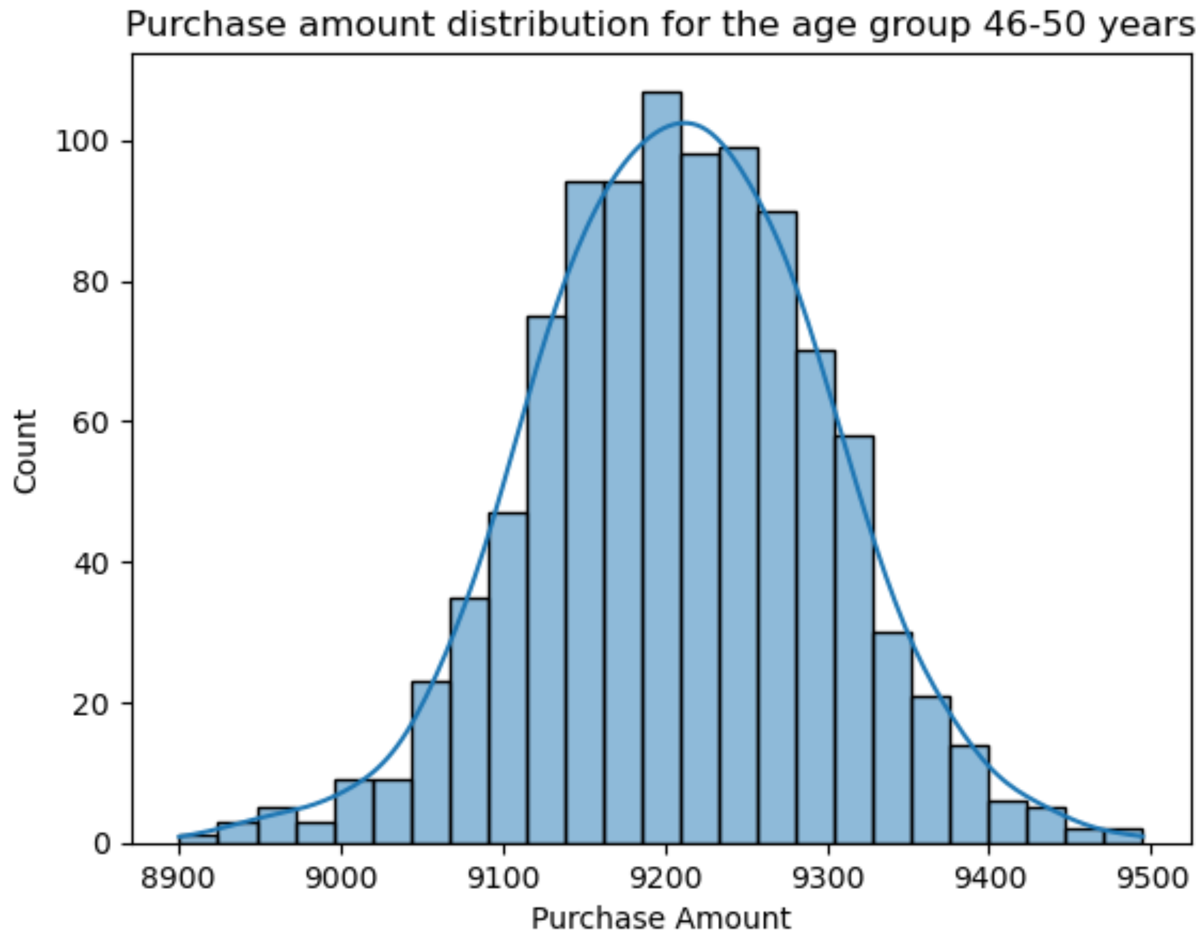
In [77]:
```
sns.histplot(x=sample_means_26to35,kde=True)
plt.xlabel('Purchase Amount')
plt.title('Purchase amount distribution for the age group 26-35 years')
plt.show()
```



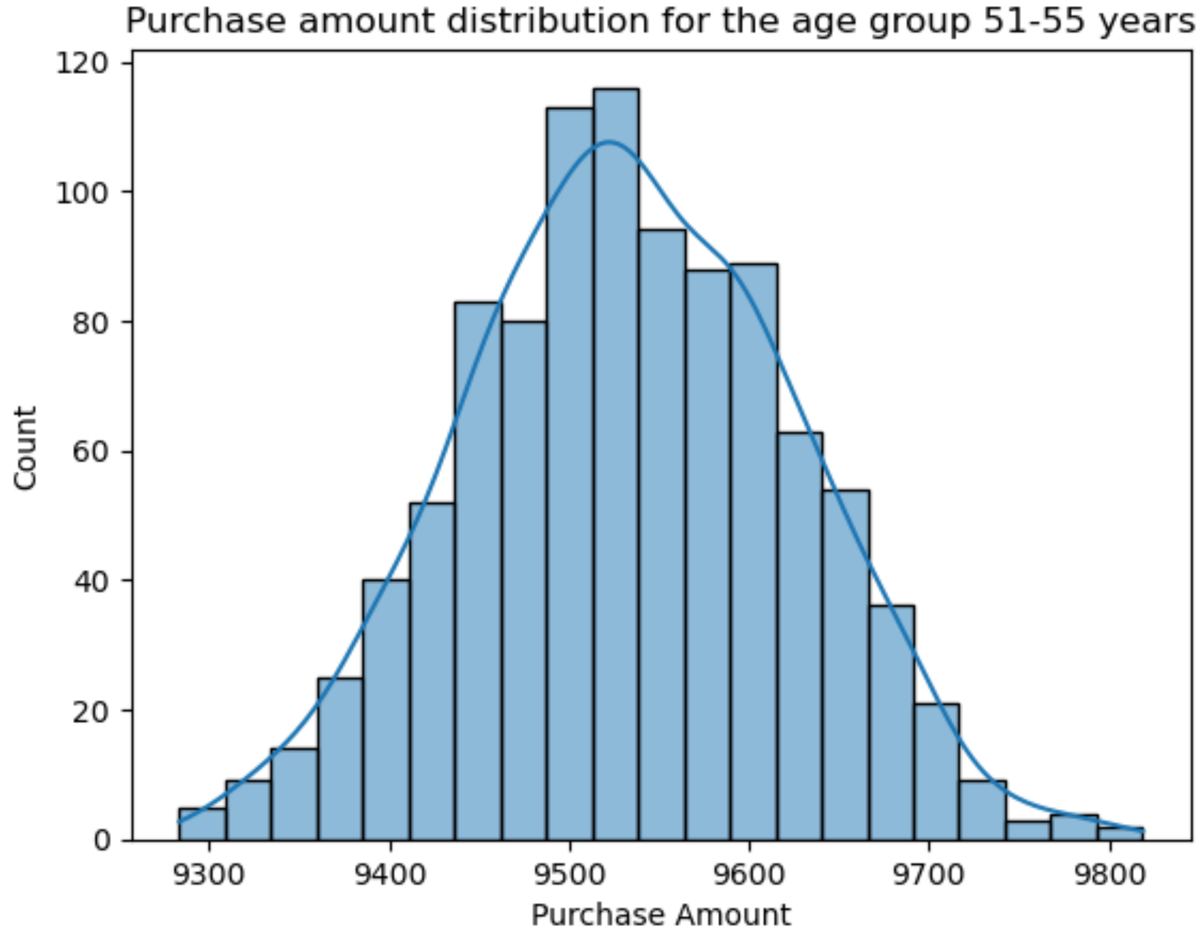
In [78]:
```
sns.histplot(x=sample_means_36to45,kde=True)
plt.xlabel('Purchase Amount')
plt.title('Purchase amount distribution for the age group 36-45 years')
plt.show()
```



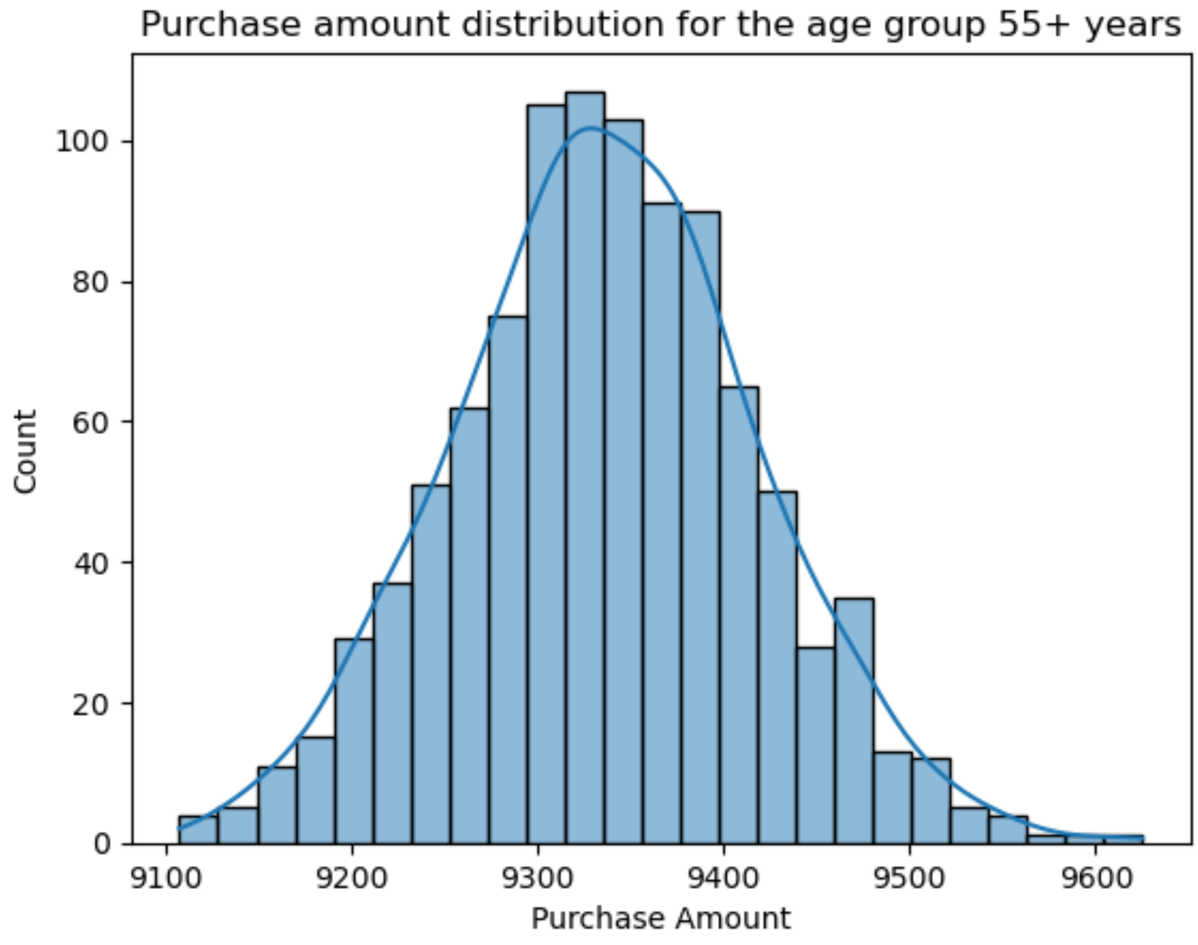
In [79]:
```
sns.histplot(x=sample_means_46to50,kde=True)
plt.xlabel('Purchase Amount')
plt.title('Purchase amount distribution for the age group 46-50 years')
plt.show()
```



In [80]:
```
sns.histplot(x=sample_means_51to55,kde=True)
plt.xlabel('Purchase Amount')
plt.title('Purchase amount distribution for the age group 51-55 years')
plt.show()
```



In [81]:
```
sns.histplot(x=sample_means_above55,kde=True)
plt.xlabel('Purchase Amount')
plt.title('Purchase amount distribution for the age group 55+ years')
plt.show()
```



95% Confidence Interval - Age < 17

In [82]:
```
# finding confidence interval (95%) for population with age below 17

sample_means_mean_below17 = np.mean(sample_means_below17)
z = norm.ppf(0.975)

ci1_b17 = round(sample_means_mean_below17 - z*(std_pop_below17/(sample_size)**0.5),2)
ci2_b17 = round(sample_means_mean_below17 + z*(std_pop_below17/(sample_size)**0.5),2)

print(f"Below 17: {ci1_b17} - {ci2_b17}")
```

```
Below 17: 8751.76 - 9117.55
```

95% Confidence Interval - Age: 18-25

In [83]:
```
# finding confidence interval (95%) for population with age 18-25

sample_means_mean_18to25 = np.mean(sample_means_18to25)
z = norm.ppf(0.975)

ci1_18to25 = round(sample_means_mean_18to25 - z*(std_pop_18to25/(sample_size)**0.5),2)
ci2_18to25 = round(sample_means_mean_18to25 + z*(std_pop_18to25/(sample_size)**0.5),2)

print(f"18 to 25: {ci1_18to25} - {ci2_18to25}")
```

```
18 to 25: 8989.09 - 9349.39
```

95% Confidence Interval - Age: 26-35

In [84]:
```
# finding confidence interval (95%) for population with age 26 to 35

sample_means_mean_26to35 = np.mean(sample_means_26to35)
z = norm.ppf(0.975)

ci1_26to35 = round(sample_means_mean_26to35 - z*(std_pop_26to35/(sample_size)**0.5),2)
ci2_26to35 = round(sample_means_mean_26to35 + z*(std_pop_26to35/(sample_size)**0.5),2)

print(f"26 to 35: {ci1_26to35} - {ci2_26to35}")
```

```
26 to 35: 9073.61 - 9432.2
```

95% Confidence Interval - Age: 36-45

In [85]:
```
# finding confidence interval (95%) for population with age 36-45

sample_means_mean_36to45 = np.mean(sample_means_36to45)
z = norm.ppf(0.975)

ci1_36to45 = round(sample_means_mean_36to45 - z*(std_pop_36to45/(sample_size)**0.5),2)
ci2_36to45 = round(sample_means_mean_36to45 + z*(std_pop_36to45/(sample_size)**0.5),2)

print(f"36 to 45: {ci1_36to45} - {ci2_36to45}")
```

```
36 to 45: 9152.76 - 9512.24
```

95% Confidence Interval - Age: 46-50

In [86]:
```
# finding confidence interval (95%) for population with age 46-50

sample_means_mean_46to50 = np.mean(sample_means_46to50)
z = norm.ppf(0.975)

ci1_46to50 = round(sample_means_mean_46to50 - z*(std_pop_46to50/(sample_size)**0.5),2)
ci2_46to50 = round(sample_means_mean_46to50 + z*(std_pop_46to50/(sample_size)**0.5),2)

print(f" 46 to 50: {ci1_46to50} - {ci2_46to50}")
```

```
 46 to 50: 9031.21 - 9386.7
```

95% Confidence Interval - Age: 51-55

In [87]:
```
# finding confidence interval (95%) for population with age 51-55

sample_means_mean_51to55 = np.mean(sample_means_51to55)
z = norm.ppf(0.975)

ci1_51to55 = round(sample_means_mean_51to55 - z*(std_pop_51to55/(sample_size)**0.5),2)
ci2_51to55 = round(sample_means_mean_51to55 + z*(std_pop_51to55/(sample_size)**0.5),2)

print(f"51 to 55: {ci1_51to55} - {ci2_51to55}")
```

```
51 to 55: 9351.18 - 9715.27
```

95% Confidence Interval - Age: 55+

In [88]:
```
# finding confidence interval (95%) for population with age 55+

sample_means_mean_above55 = np.mean(sample_means_above55)
z = norm.ppf(0.975)

ci1_above55 = round(sample_means_mean_above55 - z*(std_pop_above55/(sample_size)**0.5),2)
ci2_above55 = round(sample_means_mean_above55 + z*(std_pop_above55/(sample_size)**0.5),2)

print(f"55+ : {ci1_above55} - {ci2_above55}")
```

```
55+ : 9157.22 - 9515.88
```

-- for easy comparision -- Age CIs

Below 17: 8748.05 - 9113.84 18 to 25: 8987.67 - 9347.96 26 to 35: 9067.34 - 9425.93 36 to 45: 9151.48 - 9510.96 46 to 50: 9027.22 - 9382.71 51 to 55: 9353.95 - 9718.04 55+ : 9154.19 - 9512.86

Insights:

At 95% Confidence Interval:

1. The groups 18-25, 26-35 and 46-50 has a lower purchase amount range compared to other age groups in their prime earning age.
2. 51-55 age group has the highest average purchase amount despite no of users being less than 500.

Business Recommendations

1. The average purchase amount of male customers is higher than female customers at 99% confidence interval. It is possible that there are not many products that interest them, more research need to be done to identify the reasons for lower mean in female customers.

1. The company could take initiatives to improve female product range if there is scope for it,or roll out discounts or increase payback points on female products.

1. At 95% Confidence Interval, the average purchase amount for married and unmarried customers almost completely overlap which makes it equal to one another.

1. The age groups 18-25, 26-35 and 46-50 have have lower range at 95% Confidence Interval and have scope for improvement. The age group 51-55 have higher average purchase and yet do not contribute to a significant percent of revenue. It can be inferred that their spending potential is high, so Walmart could take measures to draw in more users of that age group. We could add equipment and products that are more relevant to their age, keep quick and user friendly shopping experience etc.

1. Walmart could also concentrate on City category A to improve it revenue potential.

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]: