Avesh Raza Nagauri

9082425683 [aveshnagauri5@gmail.com (mailto:aveshnagauri5@gmail.com)](mailto:aveshnagauri5@gmail.com)

Sales Analytics Project: Inference and Confidence Analysis

In [1]:
```python
# Importing Libraries

import numpy as np,pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
from scipy import stats
```

In [2]:
```python
df = pd.read_csv("walmart_data.csv") #Importing the data
```

In [3]:
```python
df.head()
```

Out[3]:

	User_ID	Product_ID	Gender	Age	Occupation	City_Category	Stay_In_Current_City_Years
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·

In [79]:
```python
df.isna().sum() # Checking for null values
```

Out[79]:
```
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
```

Insight:

There are **no null values in this data**

In [7]:
```
df.describe()
```

Out[7]:

	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 [34]:
```
df.nunique() # Count of unique values in each columns
```

Out[34]:
```
User_ID                        5891
Gender                            2
Occupation                       21
City_Category                     3
Stay_In_Current_City_Years        5
Marital_Status                    2
Product_Category                 20
Purchase                      18105
dtype: int64
```

In [35]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 550068 entries, 0 to 550067
Data columns (total 8 columns):
 #   Column                      Non-Null Count   Dtype 
---  ------                      --------------   ----- 
 0   User_ID                     550068 non-null  int64 
 1   Gender                      550068 non-null  int64 
 2   Occupation                  550068 non-null  int64 
 3   City_Category               550068 non-null  int64 
 4   Stay_In_Current_City_Years  550068 non-null  object
 5   Marital_Status              550068 non-null  int64 
 6   Product_Category            550068 non-null  int64 
 7   Purchase                    550068 non-null  int64 
dtypes: int64(7), object(1)
memory usage: 33.6+ MB
```

In [11]:
```
df.shape
```

Out[11]:
```
(550068, 10)
```

There are 550068 rows and 10 columns

In [24]:
```
# Changing City_Category into Continous variable

df['City_Category'].replace({"A": 1, "B": 2, "C": 3}, inplace=True)
df['Gender'].replace({"M":1, "F":2},inplace = True)
df['Stay_In_Current_City_Years'].replace({"4+":5}, inplace = True)
```

In [20]:
```
df.drop(columns=['Product_ID', 'Age'], inplace=True)
```

In [25]:
```
df
```

Out[25]:

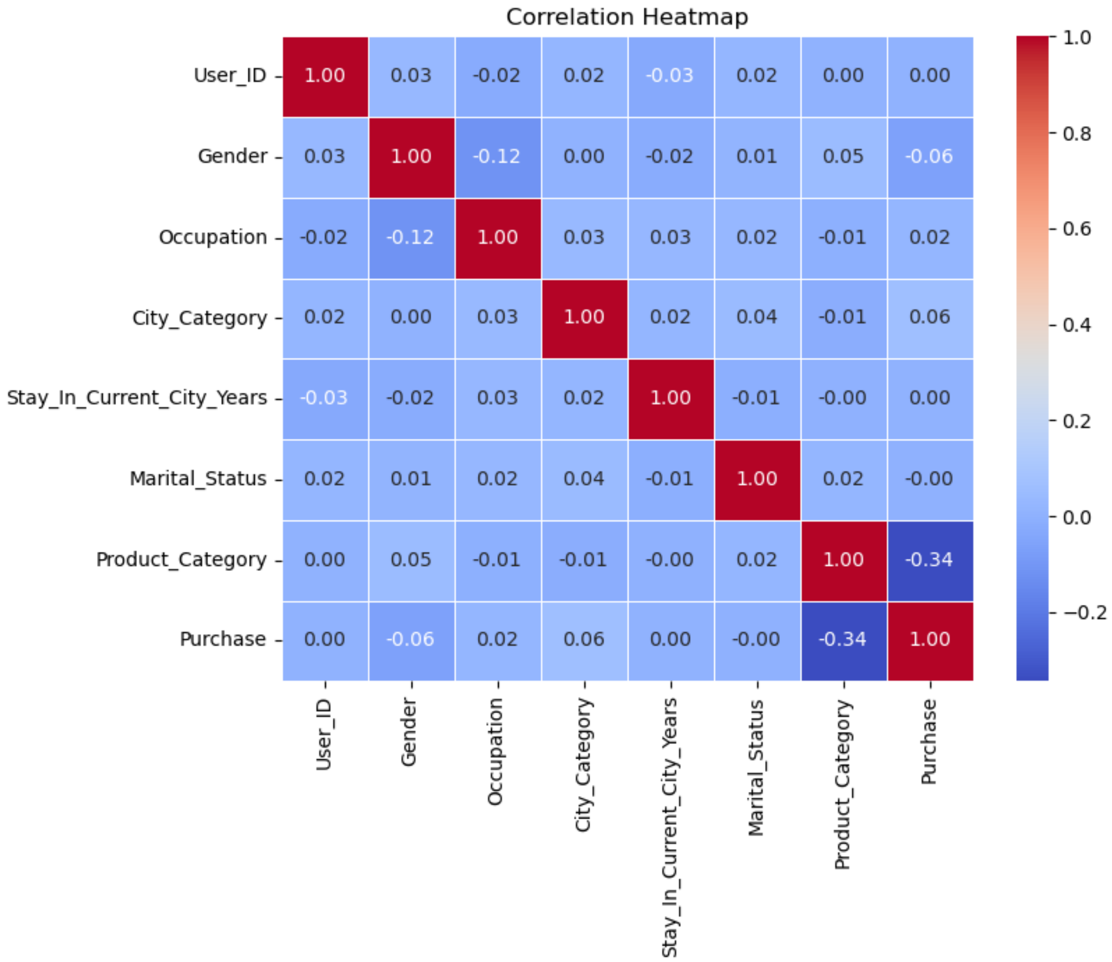
	User_ID	Gender	Occupation	City_Category	Stay_In_Current_City_Years	Marital_St
0	1000001	2	10	1	2	
1	1000001	2	10	1	2	
2	1000001	2	10	1	2	
3	1000001	2	10	1	2	
4	1000002	1	16	3	5	
...	...	...	...	...	...	
550063	1006033	1	13	2	1	
550064	1006035	2	1	3	3	
550065	1006036	2	15	2	5	
550066	1006038	2	1	3	2	
550067	1006039	2	0	2	5	

550068 rows × 8 columns

Heatmap to find co-relations

In [32]:

```
correlation_matrix = df.corr()

plt.figure(figsize=(8, 6))
sns.heatmap(correlation_matrix, annot=True, cmap='coolwarm', fmt=".2f",
plt.title("Correlation Heatmap")
plt.show()
```



NoteWorthypoints

Insights:

1. UserId shows correlations with gender and MaritalStatus.
2. Gender (0.05) exhibits a relatively high correlation with Product_Category.
3. Occupation correlates with UserId, CityCategory, StayinCityYears, MaritalStatus, and Purchase.
4. CityCategory demonstrates positive correlations with MaritalStatus (0.04) and Occupation (0.03). Additionally, it is positively related to Occupation and StayinCity.
5. StayinCity shows positive correlations with occupation and CityCategory.

More Observation and Possiblities

1. A robust positive correlation is evident between CityCategory and Occupation, supporting the hypothesis that tier 1 cities likely have a more educated population compared to other areas.
2. Similarly, a clear connection surfaces between gender and Product_Category, indicating that certain products are predominantly bought by specific genders.

Note:- Above 2 point may or mayn't be true as Correlation doesn't imply Causation

In [7]:
```
df.head()
```

Out[7]:

	User_ID	Product_ID	Gender	Age	Occupation	City_Category	Stay_In_Current_City_Years
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+

In [48]:
```
df['Gender'].value_counts().reset_index()
```

Out[48]:

	Gender	count
0	M	414259
1	F	135809

In [47]:
```
df['Age'].value_counts().reset_index()
```

Out[47]:

	Age	count
0	26-35	219587
1	36-45	110013
2	18-25	99660
3	46-50	45701
4	51-55	38501
5	55+	21504
6	0-17	15102

In [46]:
```
df['Occupation'].value_counts().reset_index()
```

Out[46]:

	Occupation	count
0	4	72308
1	0	69638
2	7	59133
3	1	47426
4	17	40043
5	20	33562
6	12	31179
7	14	27309
8	2	26588
9	16	25371
10	6	20355
11	3	17650
12	10	12930
13	5	12177
14	15	12165
15	11	11586
16	19	8461
17	13	7728
18	18	6622
19	9	6291
20	8	1546

In [92]:
```
df['City_Category'].value_counts().reset_index()
```

Out[92]:

	City_Category	count
0	B	231173
1	C	171175
2	A	147720

In [93]:
```
df['Stay_In_Current_City_Years'].value_counts().reset_index()
```

Out[93]:

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

In [94]:
```
df['Marital_Status'].value_counts().reset_index()
```

Out[94]:

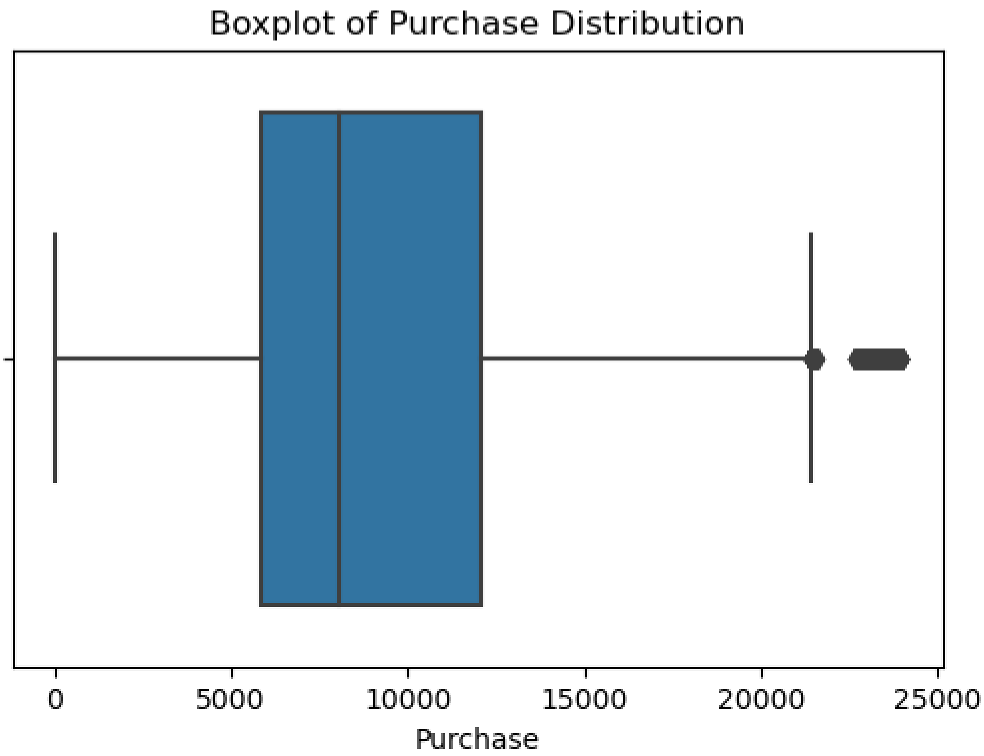
	Marital_Status	count
0	0	324731
1	1	225337

In [91]:
```
df['Product_Category'].value_counts().reset_index().head()
```

Out[91]:

	Product_Category	count
0	5	150933
1	1	140378
2	8	113925
3	11	24287
4	2	23864

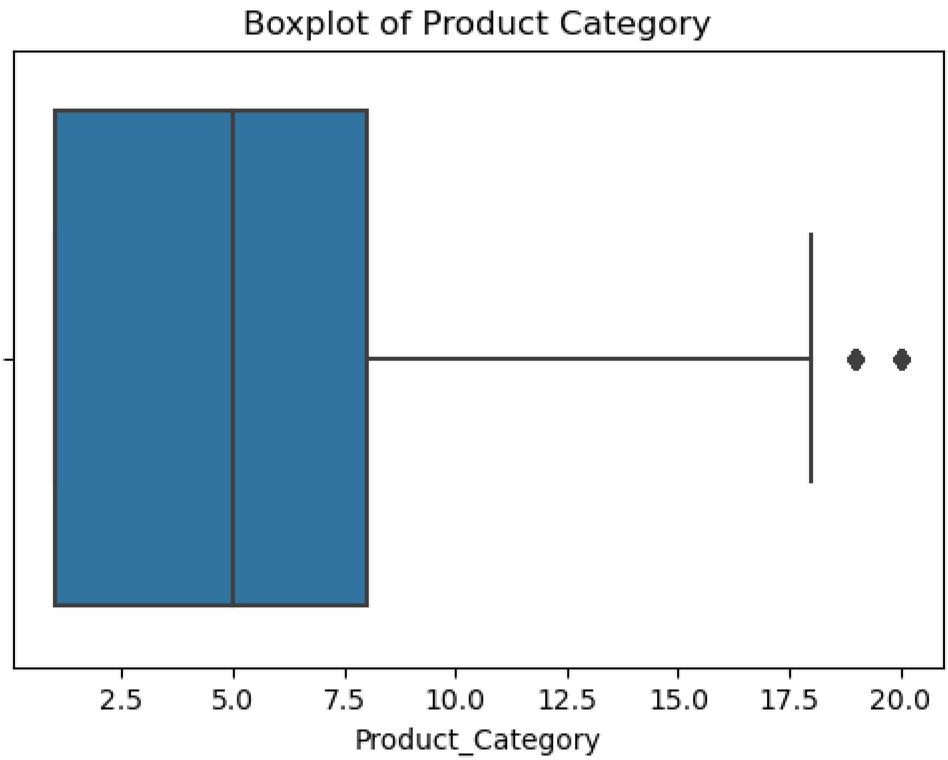
In [90]:
```
#Boxplot for Product Category

plt.figure(figsize=(6, 4))
sns.boxplot(x='Purchase', data=df)
plt.title('Boxplot of Purchase Distribution')
plt.show()
```



Insight:

The Purchase column exhibits a **few outliers,** suggesting the presence of values that significantly differ from the overall distribution within the dataset.

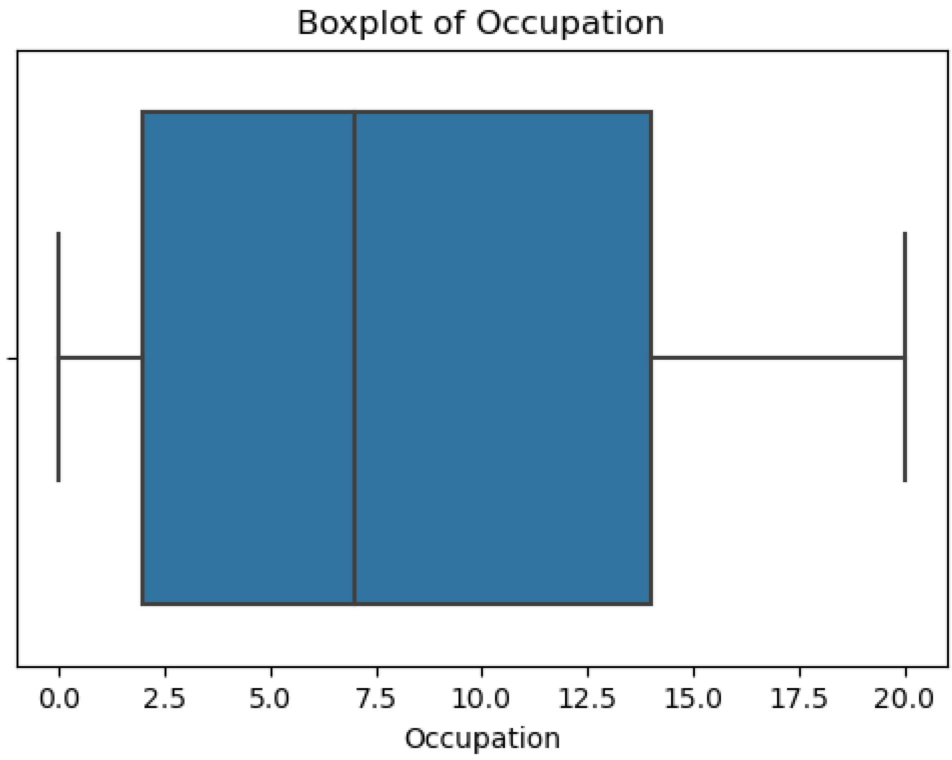
In [27]:
```
# Boxplot for Product Category

plt.figure(figsize=(6, 4))
sns.boxplot(x='Product_Category', data=df)
plt.title('Boxplot of Product Category')
plt.show()
```



Insight:

The Product Category column reveals the **presence of two outliers,** indicating unusual or extreme values that deviate from the general pattern within the dataset.

In [88]:
```
# Boxplot for Occupation

plt.figure(figsize=(6, 4))
sns.boxplot(x='Occupation', data=df)
plt.title('Boxplot of Occupation')
plt.show()
```

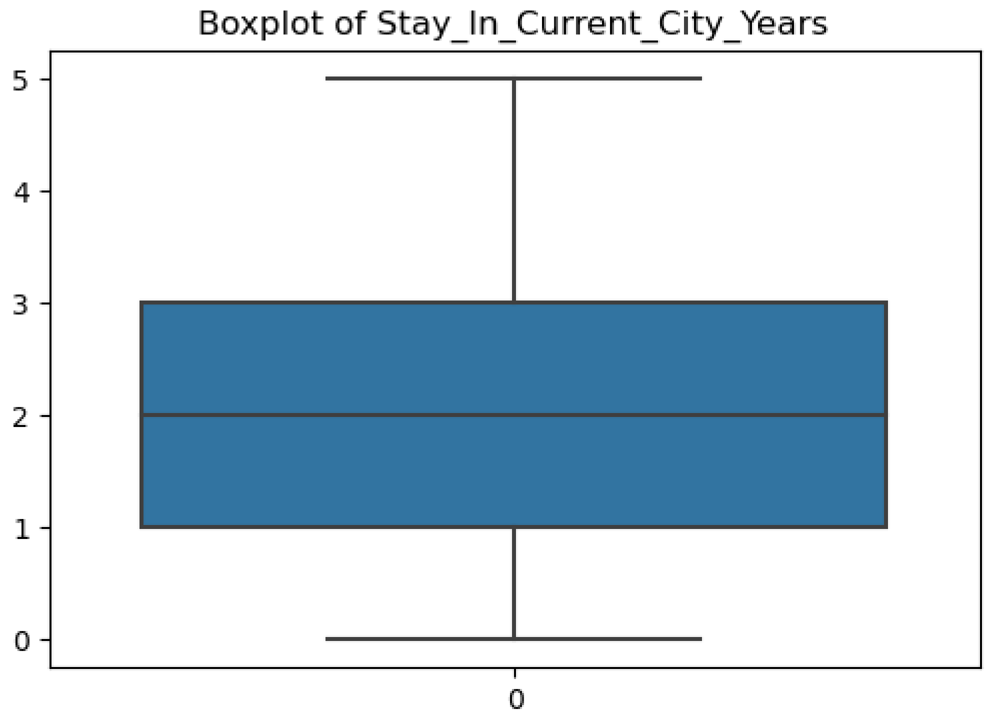


Insight:

The Occupation column exhibits a **lack of outliers**, suggesting a uniform distribution without significant deviations or extreme values that could potentially skew the dataset.

In [87]:
```
# Boxplot for Stay_In_Current_City_Years

arr = df.copy()
arr['Stay_In_Current_City_Years'].replace({"4+":5}, inplace = True)
```

In [86]:
```
plt.figure(figsize=(6, 4))
sns.boxplot(arr['Stay_In_Current_City_Years'])
plt.title('Boxplot of Stay_In_Current_City_Years')
plt.show()
```



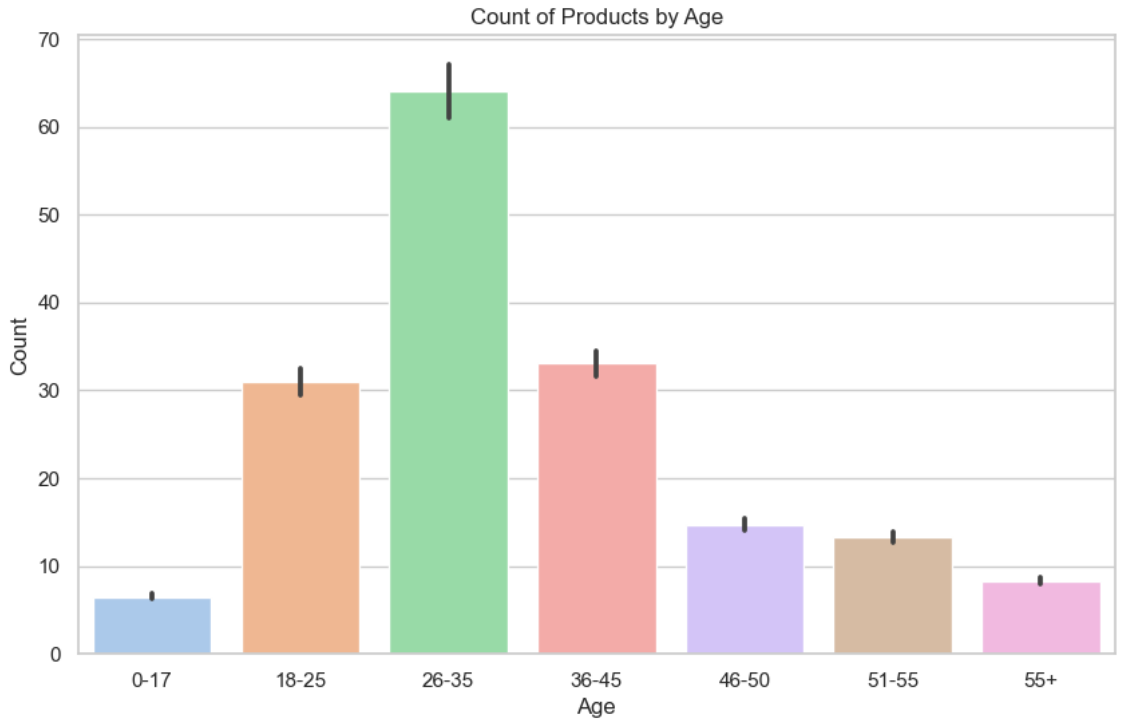
Insight:

The analysis reveals a **absence of outliers within the Stay_In_Current_City_Years,** indicating a consistent distribution without extreme values that could significantly impact the dataset.

In [16]:
```
# Count of ProductID with every age bin

count_by_age_product = df.groupby('Age')[['Product_ID']].value_counts()
filter_df = count_by_age_product.reset_index()
```

In [17]:

```
plt.figure(figsize=(10, 6))
sns.barplot(data=filter_df, x='Age', y='count', palette='pastel')
plt.title('Count of Products by Age')
plt.xlabel('Age')
plt.ylabel('Count')
plt.show()
```

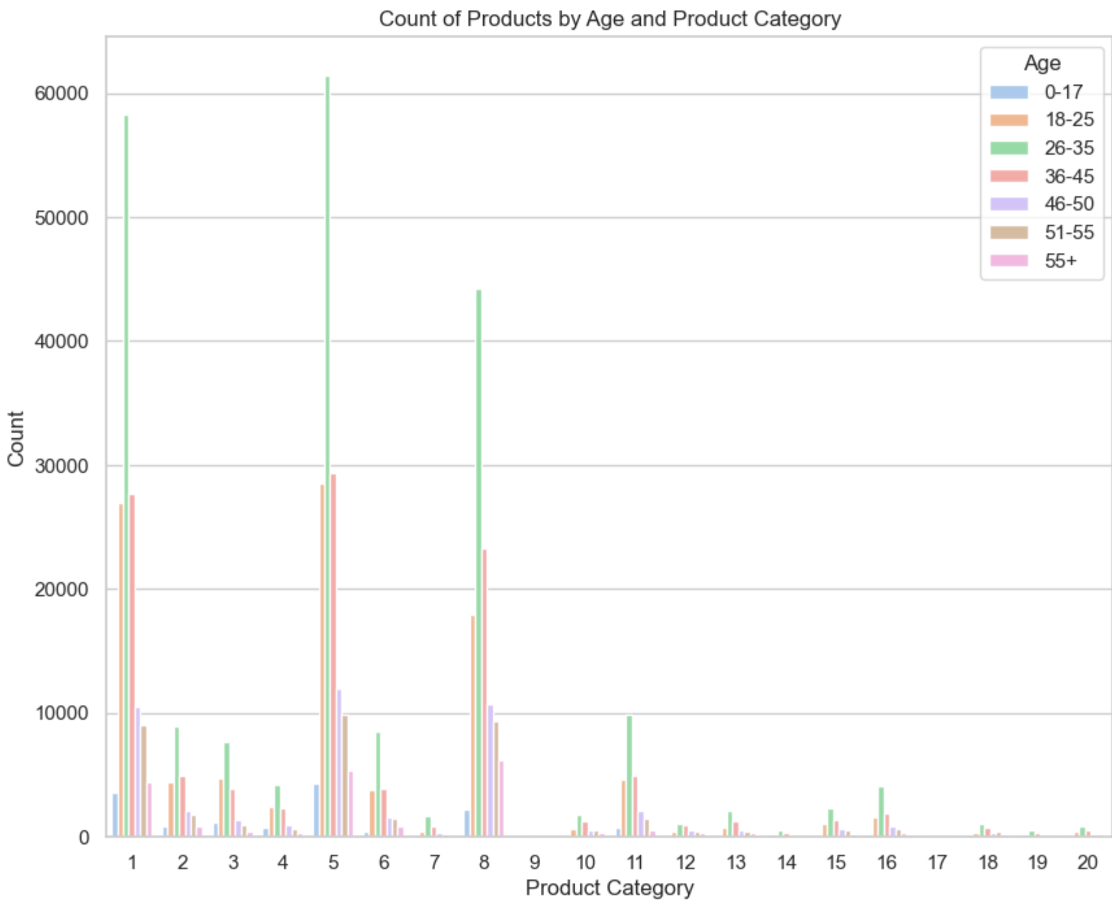


Insight:

The age group between **26-35 emerges as the most active consumer segment, demonstrating the highest purchasing behavior.** Age groups **36-45 and 18-25 closely follow,** maintaining comparable levels of engagement, while **other age groups exhibit comparatively lower contribution to overall purchases.**

What products are different age groups buying?

In [14]:
```
filter_category = df.groupby('Age')['Product_Category'].value_counts().r

sns.set(style="whitegrid")
colors = sns.color_palette("pastel")

plt.figure(figsize=(10, 8))
sns.barplot(data=filter_category, x='Product_Category', y='count', hue=

plt.title('Count of Products by Age and Product Category')
plt.xlabel('Product Category')
plt.ylabel('Count')

plt.legend(title='Age', title_fontsize='12')

plt.show()
```



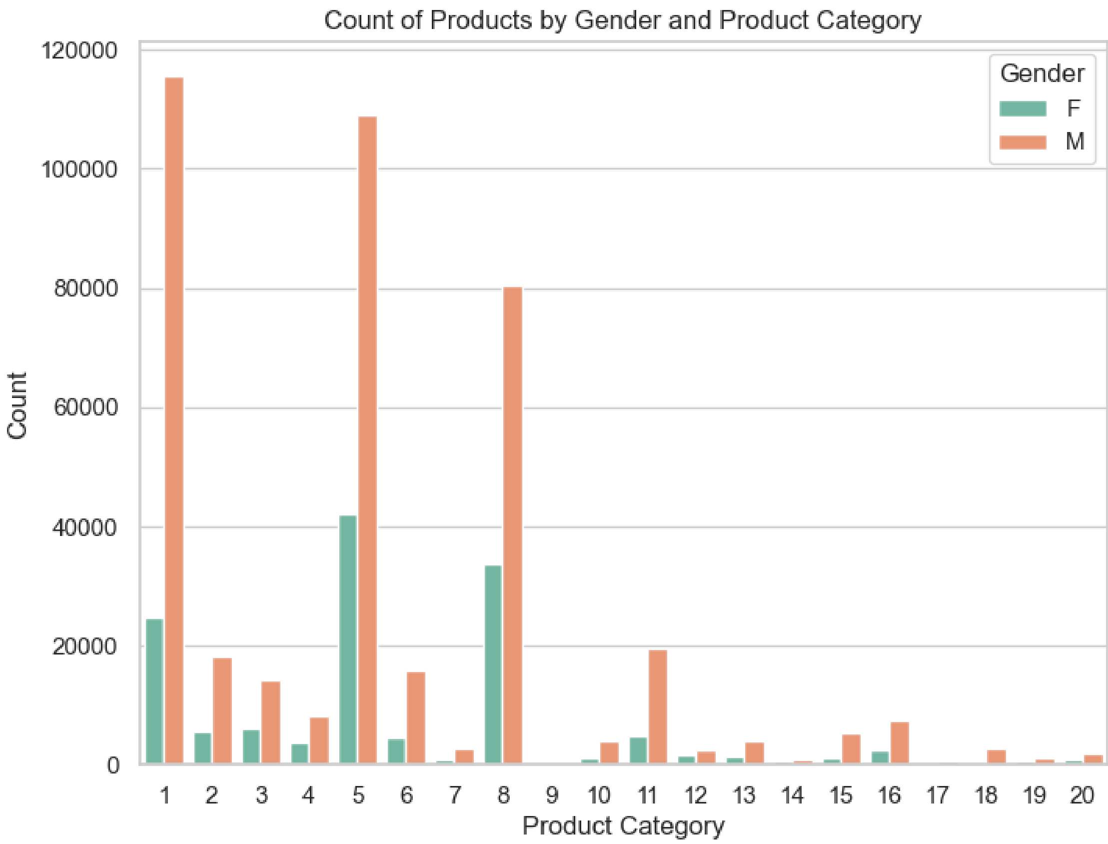
Insight:

1. The **significant consumer activity is observed in product categories 1, 5, and 8, while categories 2, 3, 4, 6, 11,and 16 witness decent footfall.** Others either have negligible or no substantial contribution to the business.
2. Notably, the **age group 26-35 stands out with the highest consumer presence in every product category.** This highlights that our primary consumer base belongs to this age bracket, **followed by the 36-45 and 18-25 age groups.**

Are there preferred product categories for different genders?

In [12]:

```
sns.set(style="whitegrid")
colors = sns.color_palette("Set2")

plt.figure(figsize=(8, 6))
sns.countplot(data=df, x='Product_Category', hue='Gender', palette=colo


plt.title('Count of Products by Gender and Product Category')
plt.xlabel('Product Category')
plt.ylabel('Count')

plt.show()
```



Insight:

1. Product Categories **1, 5, and 8 consistently attract the highest number of consumers across both genders,** indicating their popularity. Other categories face challenges in garnering similar attention.
2. **Males dominate every product category,** underscoring that the majority of our customers are males, suggesting a **gender-based preference** in our customer base.

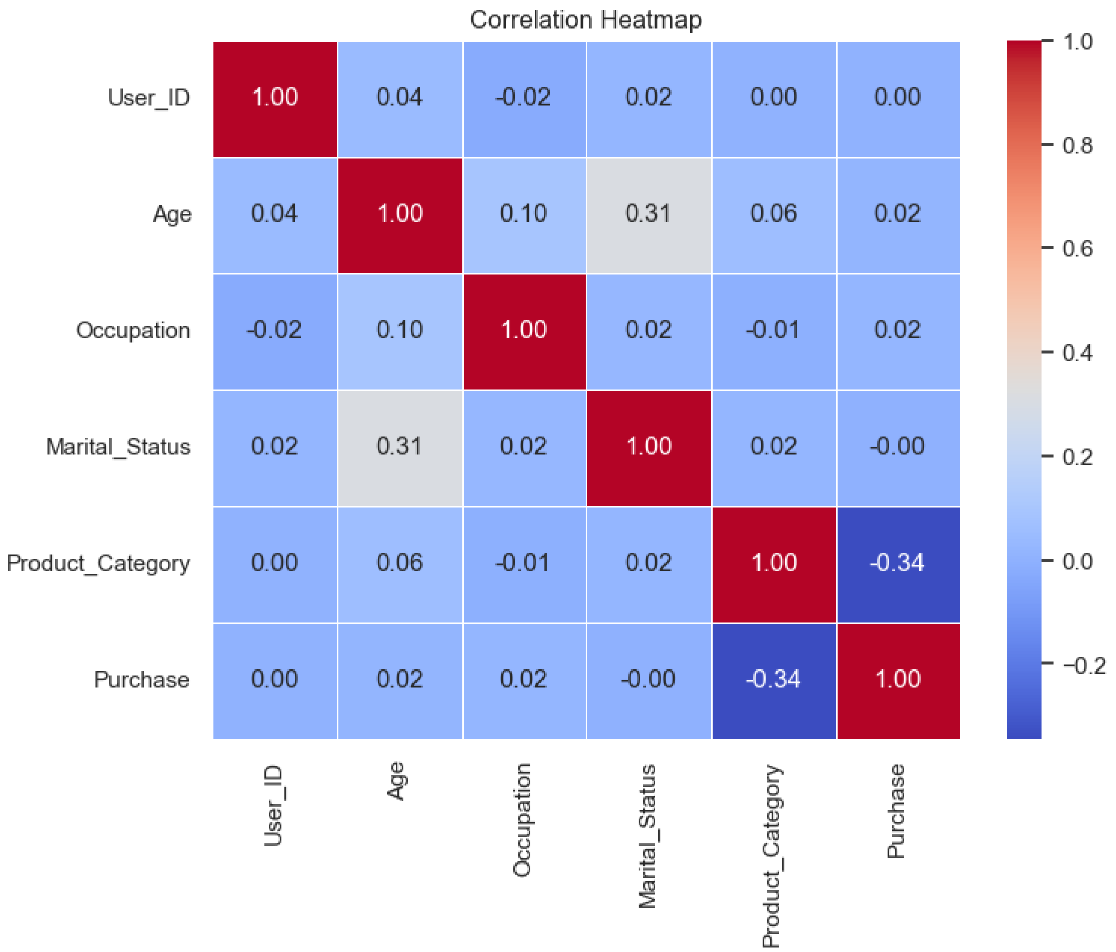
Is there a relationship between age, marital status, and the amount spent?

In [34]:

```
arr = df.copy()  # Creating a copy of data set

arr['Age'].replace({"0-17":17,"18-25":25,"26-35":35,"36-45":45,"46-50":5

arr = arr.drop(columns=['Product_ID', 'Gender', 'City_Category', 'Stay_I
```

In [35]:

```
correlation_matrix = arr.corr()

plt.figure(figsize=(8, 6))
sns.heatmap(correlation_matrix, annot=True, cmap='coolwarm', fmt=".2f",
plt.title("Correlation Heatmap")
plt.show()
```



Insight:

1. **Age and Purchase exhibit a positive correlation,** suggesting that there may be a connection between the age of consumers and their spending behavior.
2. Conversely, there is a **negative correlation between MartialStatus and Purchase,** indicating a potential association between marital status and the amount spent.

3. In conclusion, **it is challenging to definitively assert that consumer spending depends solely on age or marital status, as the relationships observed are**

How does gender affect the amount spent?

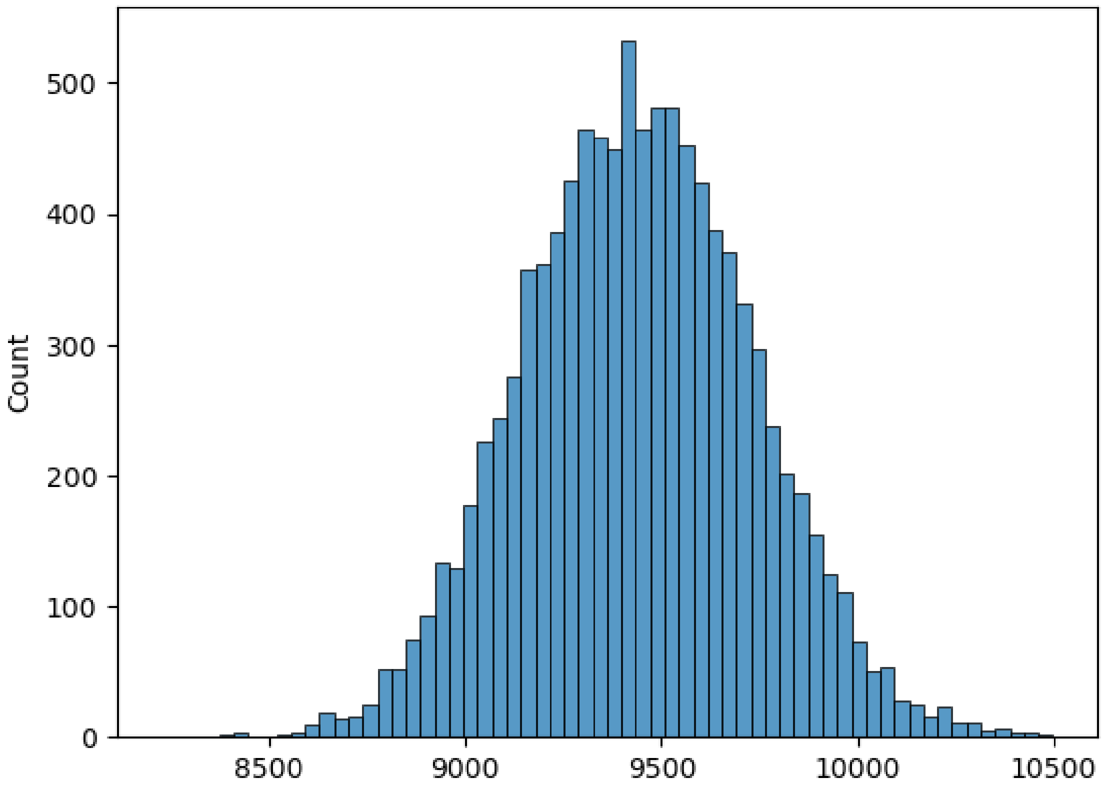
In [38]:
```
df_male = df[df['Gender']=='M']
df_female = df[df['Gender']=='F']
```

Sample for 300 Males

In [119]:
```
sample_male_300 = [np.mean(df_male['Purchase'].sample(300)) for i in ran
```

In [101]:
```
sns.histplot(sample_male_300)
```

Out[101]: <Axes: ylabel='Count'>



In [129]:
```
mu_sample_300 = np.mean(sample_male_300)    # Mean
mu_sample_300
```

Out[129]: 9438.251175

In [130]:
```
sigma_300 = np.std(sample_male_300) # std Dev
```

In [131]:
```
sigma = sigma_300/300**0.5   # Updated Std Dev
sigma
```

Out[131]:
```
16.993324627046984
```

In [132]:
```
# Confidence interval using code

stats.norm.interval(.95,loc = mu_sample_300, scale = sigma) # Confidence
```

Out[132]:
```
(9404.94487075339, 9471.557479246609)
```

In [133]:
```
CI_plus = mu_sample_300 + (1.96 * sigma)
CI_minus = mu_sample_300 - (1.96 * sigma)
```

In [121]:
```
CI_minus,CI_plus # Confidence Interval using Formula
```

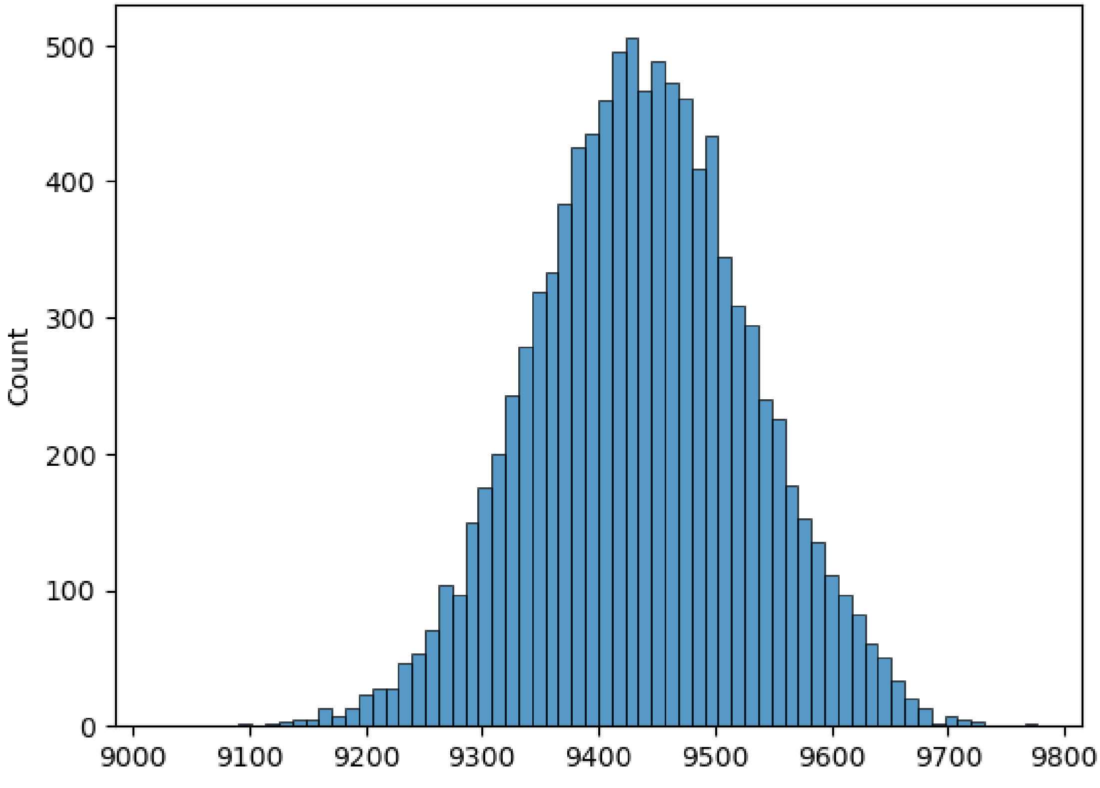
Out[121]:
```
(9404.307769914474, 9470.744311030056)
```

Sample for 3000 Males

In [148]:
```
sample_male_3000 = [np.mean(df_male['Purchase'].sample(3000)) for i in r
```

In [149]:
```
sns.histplot(sample_male_3000)
```

Out[149]: <Axes: ylabel='Count'>



In [150]:
```
sigma_3000 = np.std(sample_male_3000) #std dev
```

In [151]:
```
sigma = sigma_3000/3000**0.5 # updated std dev
sigma
```

Out[151]: 1.699017981621608

In [152]:
```
mu_sample_3000 = np.mean(sample_male_3000) #mean
mu_sample_3000
```

Out[152]: 9436.734933766666

In [153]:
```
stats.norm.interval(.95,loc = mu_sample_300, scale = sigma) #Confidence
```

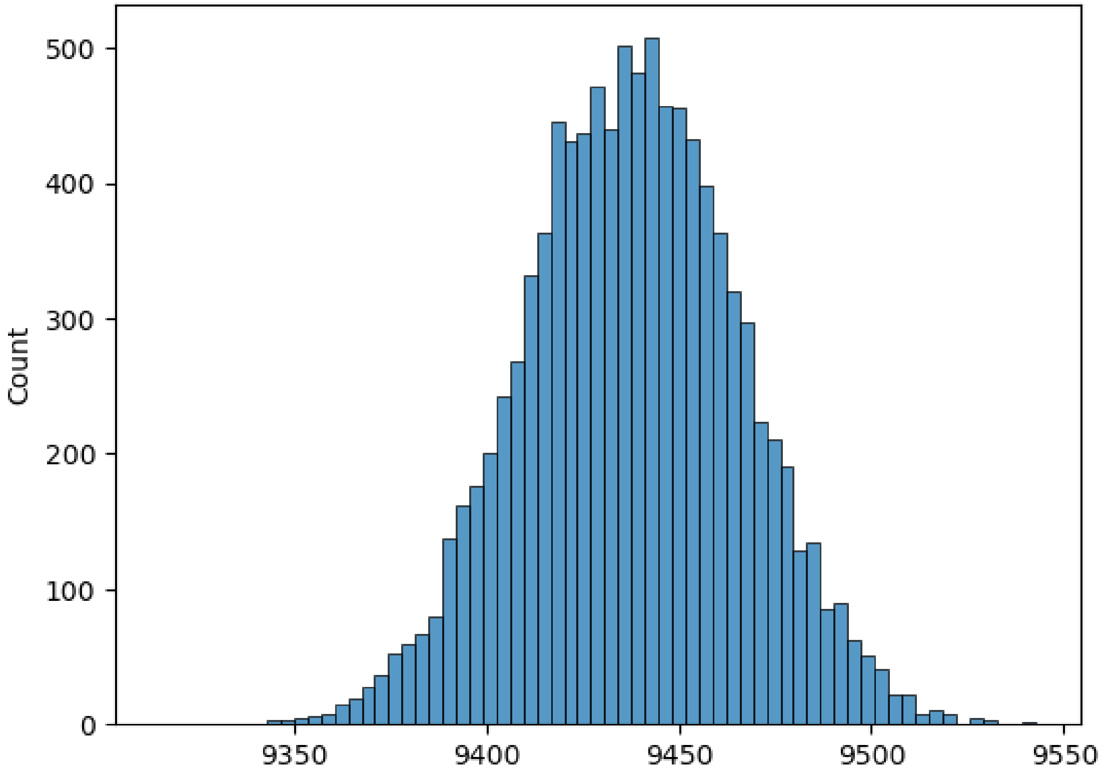
Out[153]: (9434.921160946935, 9441.581189053064)

Sample for 30000 Males

In [140]:
```
sample_male_30000 = [np.mean(df_male['Purchase'].sample(30000)) for i in
```

In [141]:
```
sns.histplot(sample_male_30000)
```

Out[141]: <Axes: ylabel='Count'>



In [144]:
```
sigma_30000 = np.std(sample_male_30000) #std dev
sigma_30000
```

Out[144]: 28.161534770279903

In [145]:
```
sigma = sigma_30000/30000**0.5 # updated std dev
sigma
```

Out[145]: 0.1625906968041411

In [146]:
```
mu_sample_30000 = np.mean(sample_male_30000) #mean
mu_sample_30000
```

Out[146]: 9437.75691272

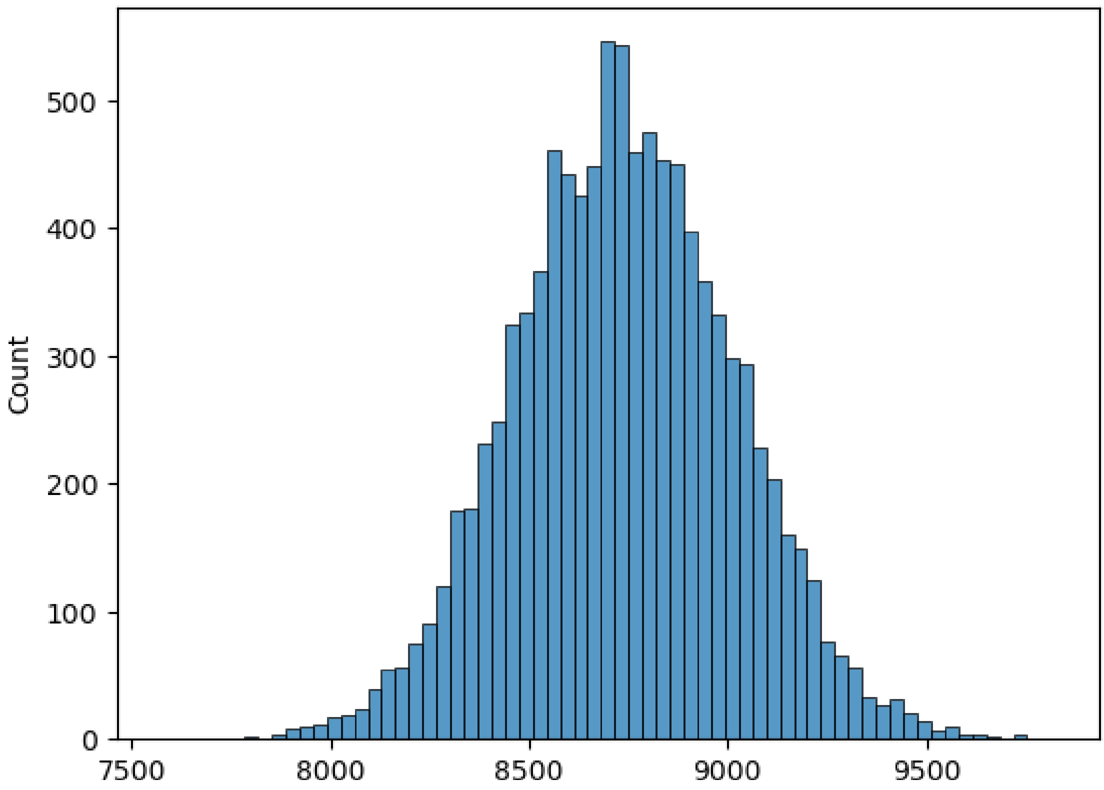
In [147]:
```
stats.norm.interval(.95,loc = mu_sample_30000, scale = sigma)
```

Out[147]: (9437.438240810043, 9438.075584629958)

Sample for 300 females

In [167]:
```
sample_female_300 = [np.mean(df_female['Purchase'].sample(300)) for i in
sns.histplot(sample_female_300)
```

Out[167]: <Axes: ylabel='Count'>



In [171]:
```
sigma_300 = np.std(sample_female_300) #std dev
sigma_300
```

Out[171]: 276.7342883674166

In [172]:
```
sigma = sigma_300/300 ** 0.5 # Updated Std Dev
sigma
```

Out[172]: 15.977261588292746

In [173]:
```
mu_sample_300 = np.mean(sample_female_300) #mean
mu_sample_300
```

Out[173]: 8738.214141333332

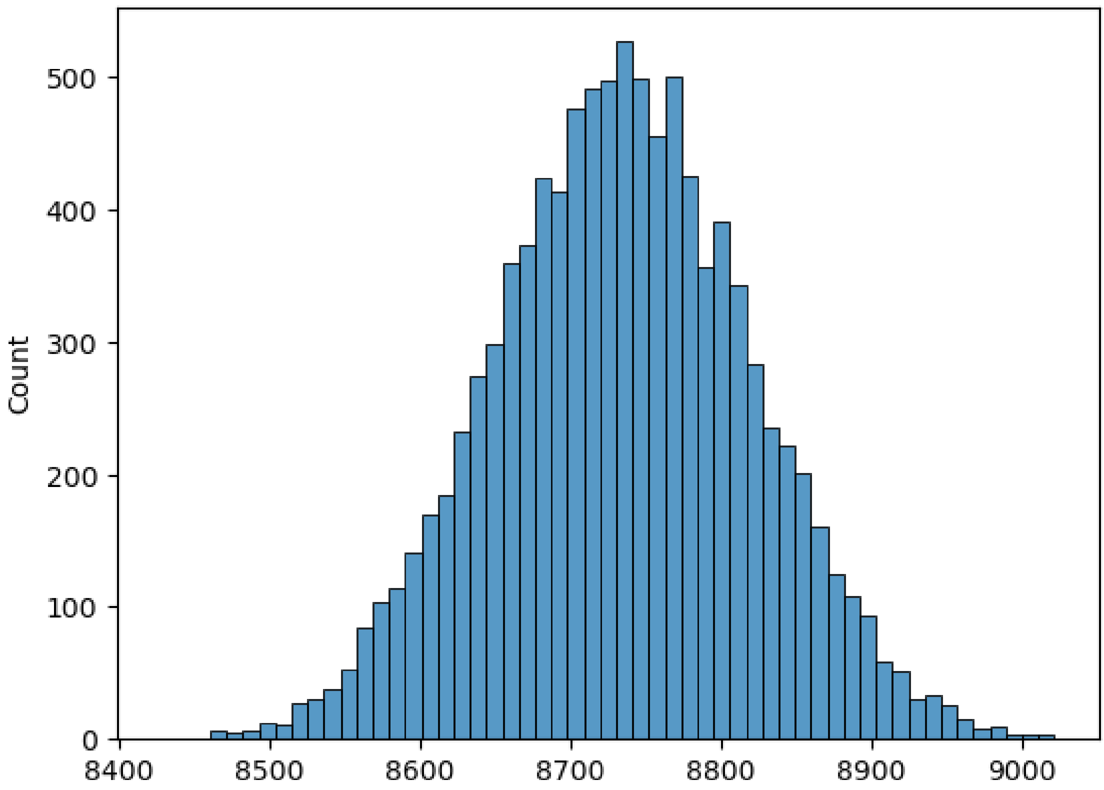
In [174]:
```
stats.norm.interval(.95,loc = mu_sample_300, scale = sigma) # Confidence
```

Out[174]: (8706.899284048703, 8769.528998617961)

Sample for 3000 Females

In [162]:
```
sample_female_3000 = [np.mean(df_female['Purchase'].sample(3000)) for i
sns.histplot(sample_female_3000)
```

Out[162]: <Axes: ylabel='Count'>



In [163]:
```
sigma_3000 = np.std(sample_female_3000) #std dev
sigma_3000
```

Out[163]: 86.09773458510762

In [164]:
```
sigma = sigma_3000/3000 ** 0.5 # Updated Std Dev
sigma
```

Out[164]: 1.571922379411871

In [165]:
```
mu_sample_3000 = np.mean(sample_female_3000) #mean
mu_sample_3000
```

Out[165]: 8733.947207866666

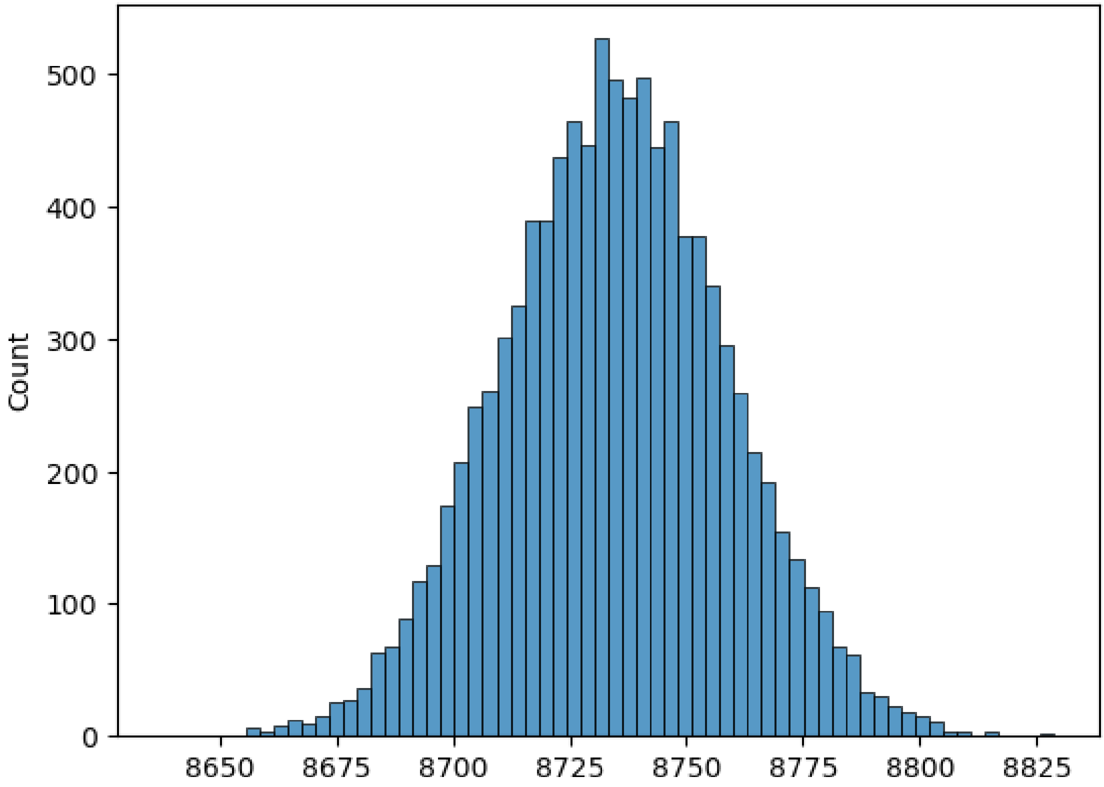
In [166]:
```
stats.norm.interval(.95,loc = mu_sample_3000, scale = sigma) # Confidenc
```

Out[166]: (8730.866296616527, 8737.028119116805)

Sample for 30000 Females

In [175]:
```
sample_female_30000 = [np.mean(df_female['Purchase'].sample(30000)) for
sns.histplot(sample_female_30000)
```

Out[175]: <Axes: ylabel='Count'>



In [176]:
```
sigma_30000 = np.std(sample_female_30000) #std dev
sigma_30000
```

Out[176]: 24.214617964716805

In [177]:
```
sigma = sigma_30000/30000 ** 0.5 # Updated Std Dev
sigma
```

Out[177]: 0.13980316200253196

In [178]:
```
mu_sample_30000 = np.mean(sample_female_30000) #mean
mu_sample_30000
```

Out[178]: 8734.496603833335

In [179]:
```
stats.norm.interval(.95,loc = mu_sample_30000, scale = sigma) # Confider
```

Out[179]: (8734.222594670886, 8734.770612995784)

Insight:

1. The **wider Confidence Interval (CI) for males (9438) compared to females (8734) suggests a potential variance in spending patterns, with males possibly exhibiting higher expenditures**.

2. Notably, the trend observed across various examples indicates that **as the sample size increases, the width of the CI narrows, emphasizing the impact of larger datasets on reducing uncertainty**.
3. Across different sample sizes, **overlapping Confidence Intervals (CI) highlight the consistent observation that with an increase in sample size.**
4. The transformation of the **distribution shape with increasing sample size signifies a progression towards a more normal distribution curve.** This shift indicates improved statistical reliability, providing a clearer representation of the

Does CI of Males and Females overlap?

The confidence intervals for the average amount spent by males and females are as follows:

Confidence interval for males: (9437.438240810043, 9438.075584629958) Confidence interval for females: (8734.222594670886, 8734.770612995784) Since the confidence intervals for males and females do not overlap, it suggests that there might be a significant difference in the average amount spent between the two groups.

How Walmart can leverage this conclusion:

1.Targeted Marketing: Walmart can customize marketing campaigns to better suit the preferences and spending behaviors of males and females. This may involve creating gender-specific promotions, advertisements, or product recommendations.

2.Product Assortment: Tailor the product assortment based on the shopping preferences of males and females. Ensure that popular products for each gender are adequately stocked to meet demand.

3.Store Layout and Merchandising: Optimize the layout of the store and product placement to cater to the preferences of male and female shoppers. This can enhance the overall shopping experience and encourage increased spending.

4.Personalized Loyalty Programs: Implement personalized loyalty programs that offer rewards and incentives based on the spending patterns of males and females. This can help increase customer loyalty and satisfaction.

5.Customer Engagement Strategies: Engage with customers through channels that resonate with their gender-specific preferences. For example, use social media platforms to share content that appeals to the target demographic.

6.Promotional Events: Plan promotional events or sales that cater specifically to the interests and needs of males and females. This could include gender-specific discount days or exclusive offers.

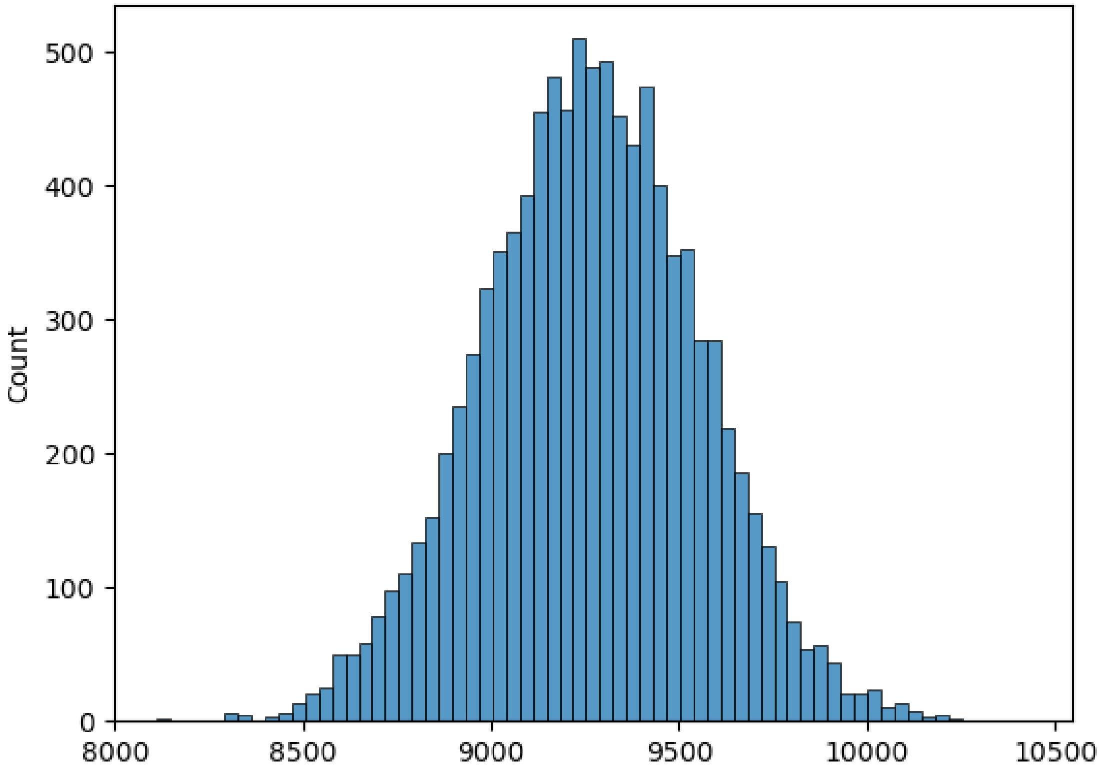
How does Marital_Status affect the amount spent?

In [206]:
```
df_married = df[df['Marital_Status']==1]
df_unmarried = df[df['Marital_Status']==0]
```

Samples for 300 Married

In [184]:
```
sample_married_300 = [np.mean(df_married['Purchase'].sample(300)) for i
sns.histplot(sample_married_300)
```

Out[184]: <Axes: ylabel='Count'>



In [185]:
```
mu_sample_300 = np.mean(sample_married_300)     # Mean
mu_sample_300
```

Out[185]: 9264.266868

In [186]:
```
sigma_300 = np.std(sample_married_300)# std Dev
sigma_300
```

Out[186]: 289.7415432904557

In [189]:
```
sigma = sigma_300 /(300**0.5) # Updated Std Dev
sigma
```

Out[189]: 16.72823580141622

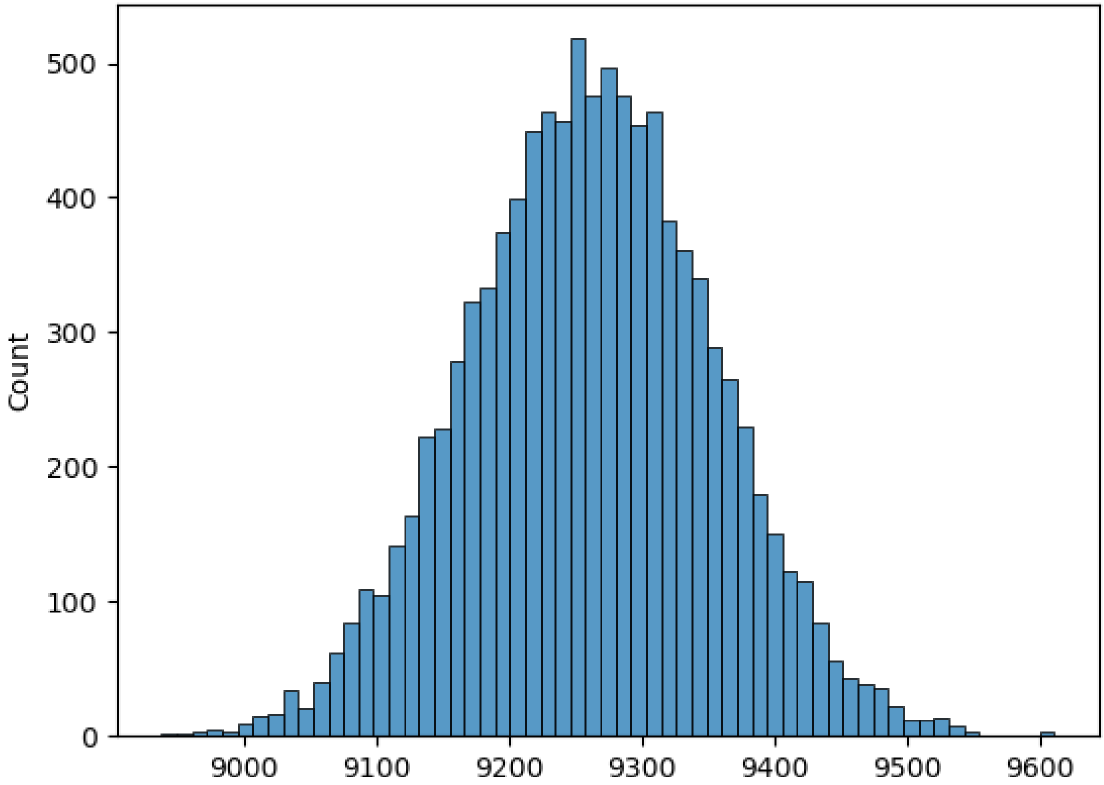
In [188]:
```
stats.norm.interval(.95,loc = mu_sample_300, scale = sigma) # Confidence
```

Out[188]: (9231.48012830433, 9297.05360769567)

Sample for 3000 Married

In [190]:
```
sample_married_3000 = [np.mean(df_married['Purchase'].sample(3000)) for
sns.histplot(sample_married_3000)
```

Out[190]: <Axes: ylabel='Count'>



In [191]:
```
mu_sample_3000 = np.mean(sample_married_3000)    # Mean
mu_sample_3000
```

Out[191]: 9260.971529866667

In [192]:
```
sigma_3000 = np.std(sample_married_3000)# std Dev
sigma_3000
```

Out[192]: 92.28991202178261

In [193]:
```
sigma = sigma_3000 /(3000**0.5) # Updated Std Dev
sigma
```

Out[193]: 1.6849755548165848

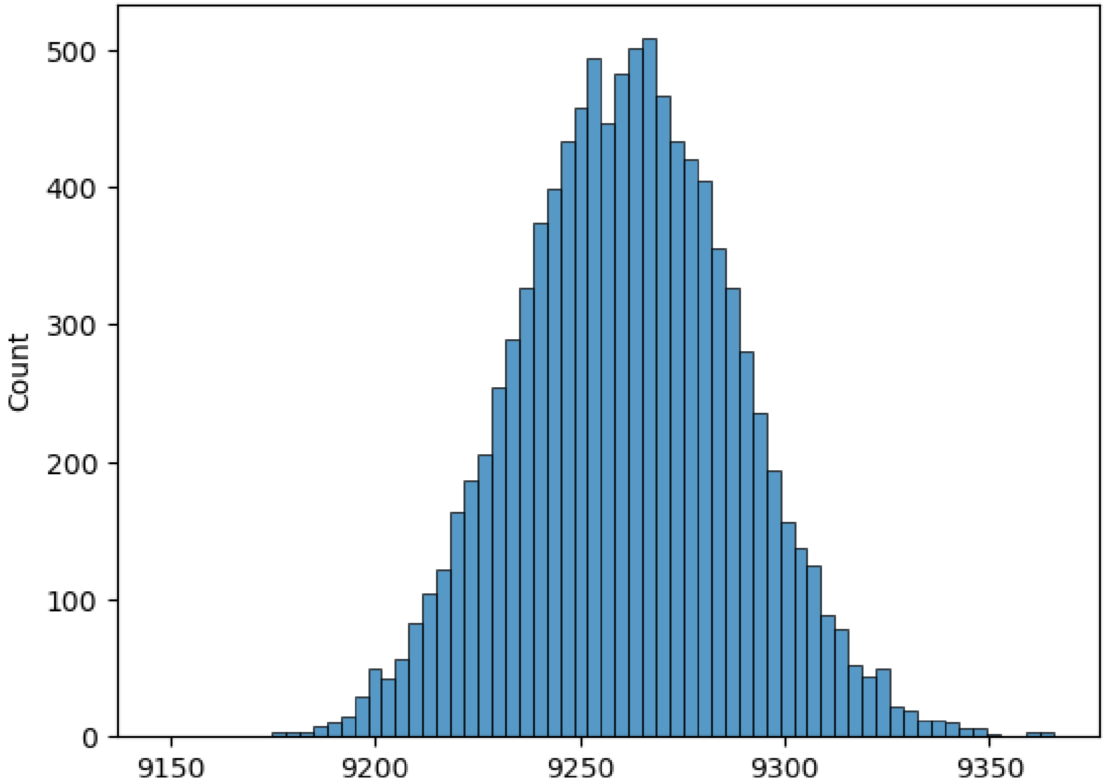
In [194]:
```
stats.norm.interval(.95,loc = mu_sample_3000, scale = sigma) # Confidenc
```

Out[194]: (9257.669038464395, 9264.274021268939)

Sample for 30000 Married

In [195]:
```
sample_married_30000 = [np.mean(df_married['Purchase'].sample(30000)) fo
sns.histplot(sample_married_30000)
```

Out[195]: <Axes: ylabel='Count'>



In [196]:
```
mu_sample_30000 = np.mean(sample_married_30000)     # Mean
mu_sample_30000
```

Out[196]: 9261.621688973333

In [197]:
```
sigma_30000 = np.std(sample_married_30000)# std Dev
sigma_30000
```

Out[197]: 27.00703753532681

In [198]:
```
sigma = sigma_30000 /(30000**0.5) # Updated Std Dev
sigma
```

Out[198]: 0.1559252039103526

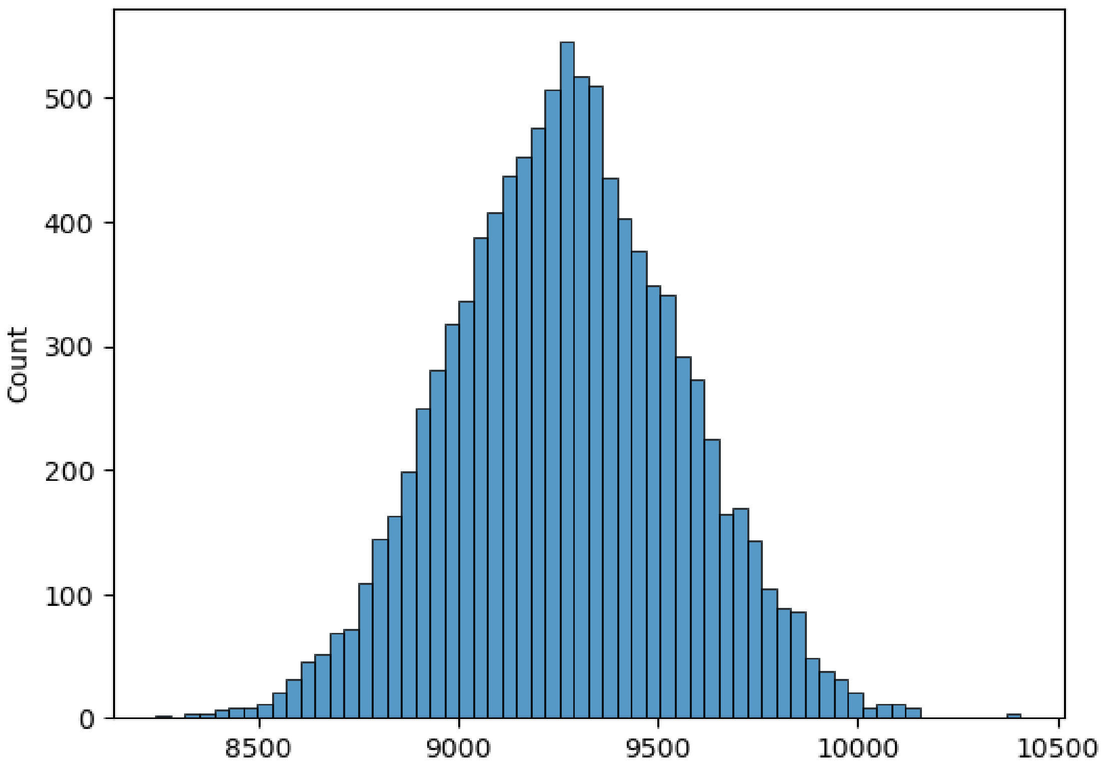
In [199]:
```
stats.norm.interval(.95,loc = mu_sample_30000, scale = sigma) # Confider
```

Out[199]: (9261.316081189387, 9261.92729675728)

Sample for 300 Unmarried

In [208]:
```
sample_unmarried_300 = [np.mean(df_unmarried['Purchase'].sample(300)) fo
sns.histplot(sample_unmarried_300)
```

Out[208]: <Axes: ylabel='Count'>



In [209]:
```
mu_sample_300 = np.mean(sample_unmarried_300)     # Mean
mu_sample_300
```

Out[209]: 9268.094742666664

In [211]:
```
sigma_300 = np.std(sample_unmarried_300)# std Dev
sigma_300
```

Out[211]: 287.8479267830583

In [212]:
```
sigma = sigma_300 /(300**0.5) # Updated Std Dev
sigma
```

Out[212]: 16.61890780138744

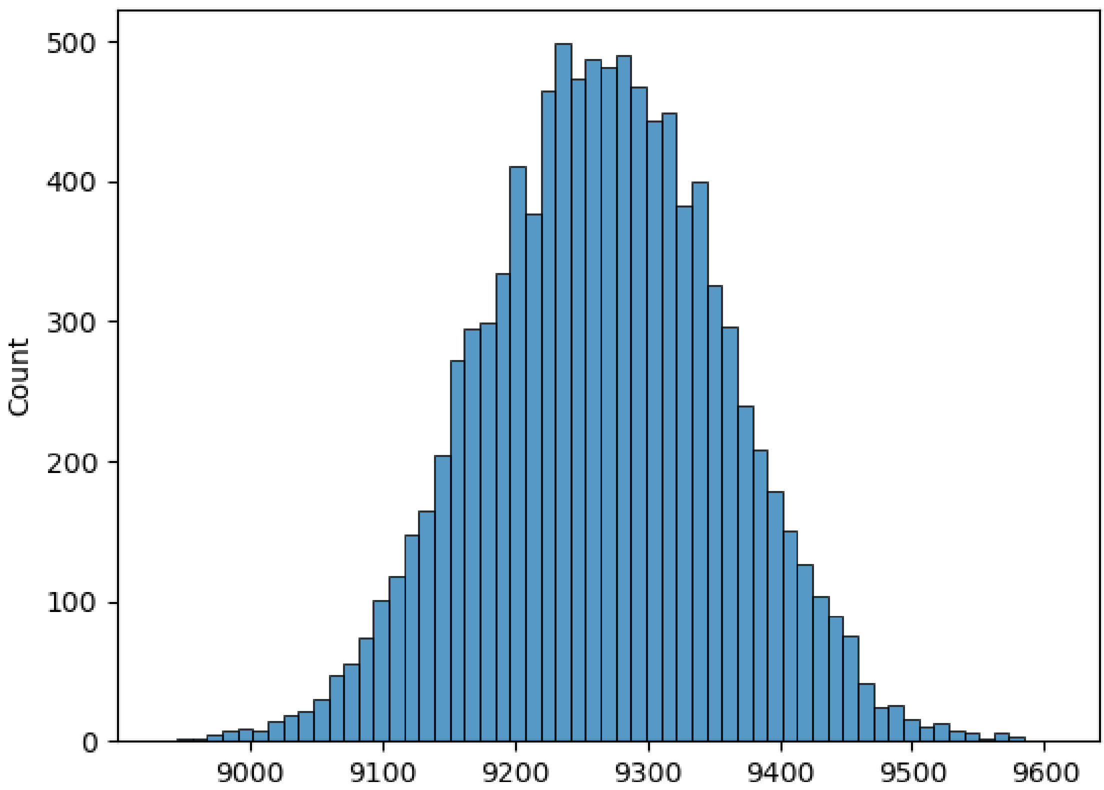
In [213]:
```
stats.norm.interval(.95,loc = mu_sample_300, scale = sigma) # Confidence
```

Out[213]: (9235.522281913552, 9300.667203419776)

Sample for 3000 umarried

In [224]:
```
sample_unmarried_3000 = [np.mean(df_unmarried['Purchase'].sample(3000))
sns.histplot(sample_unmarried_3000)
```

Out[224]: <Axes: ylabel='Count'>



In [215]:
```
mu_sample_3000 = np.mean(sample_unmarried_3000)     # Mean
mu_sample_3000
```

Out[215]: 9266.972015366668

In [216]:
```
sigma_3000 = np.std(sample_unmarried_3000)# std Dev
sigma_3000
```

Out[216]: 91.79094284432749

In [217]:
```
sigma = sigma_3000 /(3000**0.5) # Updated Std Dev
sigma
```

Out[217]: 5.299552556034202

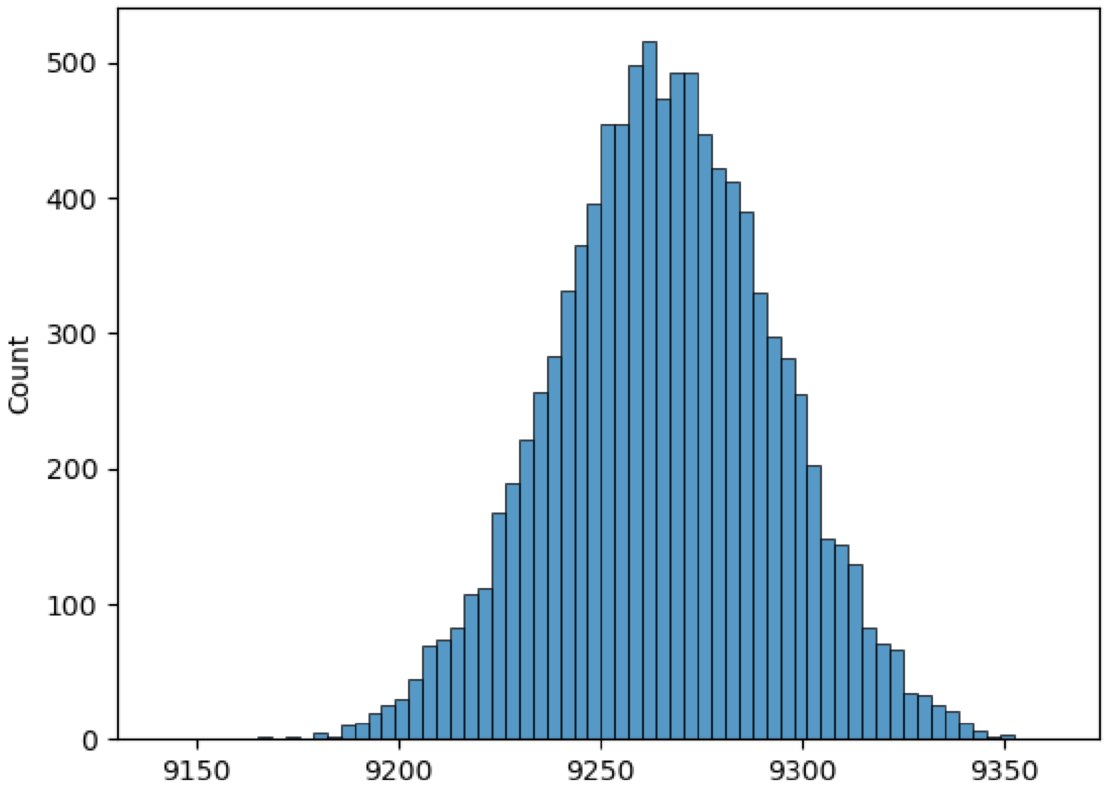
In [218]:
```
stats.norm.interval(.95,loc = mu_sample_3000, scale = sigma) # Confidenc
```

Out[218]: (9256.585083222664, 9277.358947510671)

Samples for 30000 Unmarried

In [219]:
```
sample_unmarried_30000 = [np.mean(df_unmarried['Purchase'].sample(30000)
sns.histplot(sample_unmarried_30000)
```

Out[219]: <Axes: ylabel='Count'>



In [220]:
```
mu_sample_30000 = np.mean(sample_unmarried_30000)     # Mean
mu_sample_30000
```

Out[220]: 9265.85116514

In [221]:
```
sigma_30000 = np.std(sample_unmarried_30000)# std Dev
sigma_30000
```

Out[221]: 27.468582681586156

In [222]:
```
sigma = sigma_30000 /(30000**0.5) # Updated Std Dev
sigma
```

Out[222]: 0.1585899360547126

In [223]:
```
stats.norm.interval(.95,loc = mu_sample_30000, scale = sigma) # Confider
```

Out[223]: (9265.540334577023, 9266.161995702978)

Insight:

1. The **Confidence Intervals(CI) with 30000 sample size for both married (9261) and unmarried (9266) individuals closely align, suggesting a similarity in average amount spent between these marital status categories**.
2. Demonstrated by the examples provided, an intriguing trend emerges — **as the sample size increases, the width of the CI interval decreases.** This implies a more precise estimate of the average amount spent with larger sample sizes.
3. Notably, the **CI intervals for sample sizes of 3000 and 30000 overlap**. This suggests a degree of consistency in estimating the average amount spent, even with substantial variations in sample size.
4. The **shape of the distribution undergoes changes as the sample size grows larger, resulting in a more normal distribution curve**. This transformation is indicative of improved statistical reliability and a clearer representation of the underlying data pattern.

Does CI of Married and Unmarried customers overlap?

The confidence intervals for the average amount spent by unmarried and married customers are as follows:

Unmarried Customers: (9265.54, 9266.16) Married Customers: (9261.32, 9261.93) Since the confidence intervals do not overlap, it suggests that there is a potential difference in the average amount spent between married and unmarried customers.

Implications for Walmart:

1.Targeted Marketing: Walmart can tailor its marketing strategies based on the spending patterns of married and unmarried customers. For example, specific promotions or loyalty programs could be designed to attract or retain each segment.

2.Product Placement: Understanding the spending habits of different customer segments can inform product placement within stores. Walmart can strategically place products that are more appealing to either married or unmarried customers in areas that cater to their preferences.

3.Inventory Management: The data indicates that there might be a significant difference in spending between the two groups. Walmart can optimize inventory levels for products popular among each segment, ensuring that high-demand items are well-stocked.

4.Personalized Shopping Experience: Leveraging insights into spending patterns can enable Walmart to create a more personalized shopping experience. This may involve tailoring recommendations, promotions, or discounts based on the customer's marital status.

5.Promotions and Discounts: Walmart can design promotions or discounts specifically targeted at either married or unmarried customers, encouraging increased spending within each group.

6.Customer Loyalty Programs: Walmart could explore the implementation of loyalty programs that are attractive to specific demographics. For example, loyalty points or discounts that align with the spending behaviors of customers.

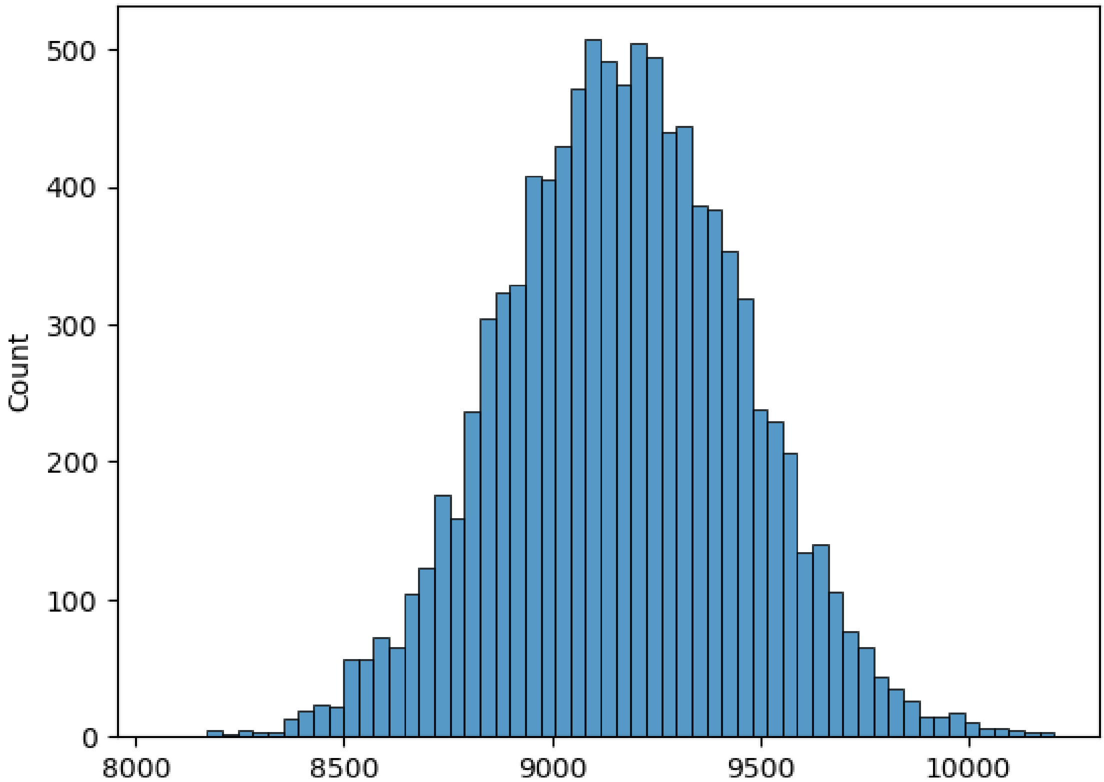
How does Age affect the amount spent?

Sample for 300 Age 18-25

In [24]:
```
df_age_25 = df[df['Age']=='18-25']
```

In [25]:
```
sample_age_300 = [np.mean(df_age_25['Purchase'].sample(300)) for i in ra
sns.histplot(sample_age_300)
```

Out[25]: <Axes: ylabel='Count'>



In [27]:
```
mu_sample_300 = np.mean(sample_age_300)     # Mean
mu_sample_300
```

Out[27]: 9168.318157666667

In [28]:
```
sigma_300 = np.std(sample_age_300)# std Dev
sigma_300
```

Out[28]: 290.3048617342087

In [29]:
```
sigma = sigma_300 /(300**0.5) # Updated Std Dev
sigma
```

Out[29]: 16.760759006930247

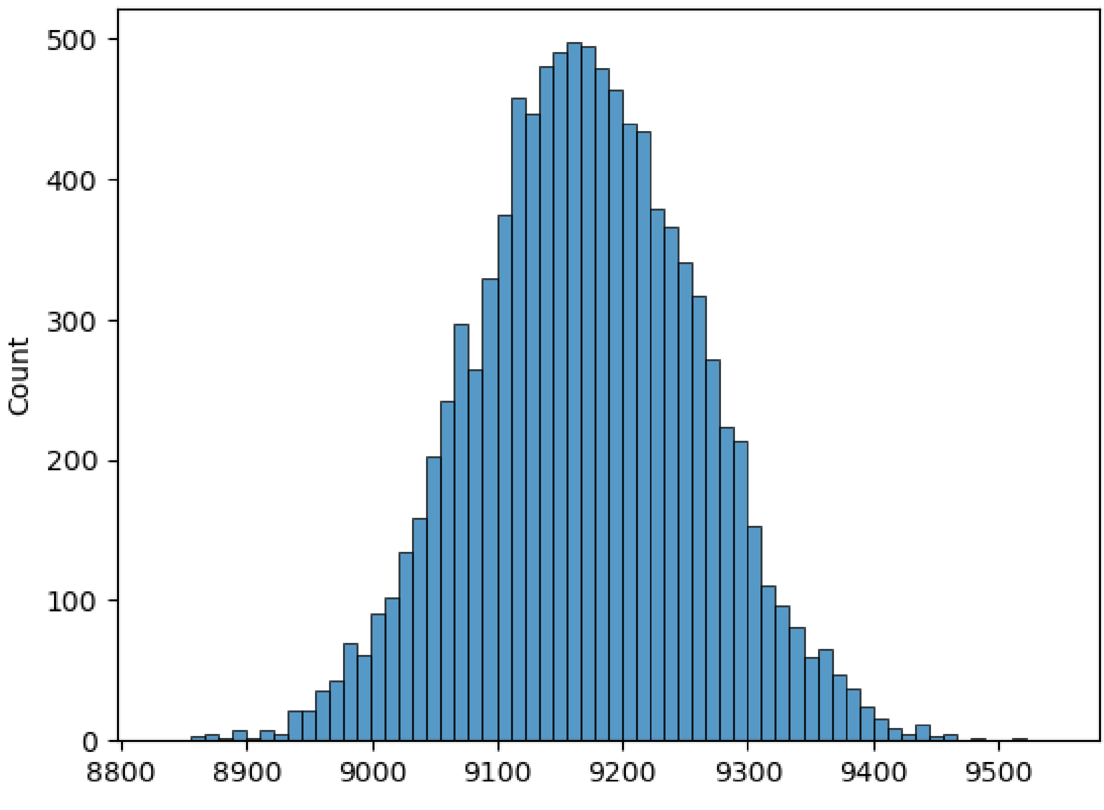
In [30]:
```
stats.norm.interval(.95,loc = mu_sample_300, scale = sigma) # Confidence
```

Out[30]: (9135.467673659528, 9201.168641673805)

Sample for 3000 Age 18-25

In [48]:
```
sample_age_3000 = [np.mean(df_age_25['Purchase'].sample(3000)) for i in
sns.histplot(sample_age_3000)
```

Out[48]: <Axes: ylabel='Count'>



In [49]:
```
mu_sample_3000 = np.mean(sample_age_3000)    # Mean
mu_sample_3000
```

Out[49]: 9170.791359733334

In [50]:
```
sigma_3000 = np.std(sample_age_3000)# std Dev
sigma_3000
```

Out[50]: 90.03866756591655

In [51]:
```
sigma = sigma_3000 /(3000**0.5) # Updated Std Dev
sigma
```

Out[51]: 1.6438736424520421

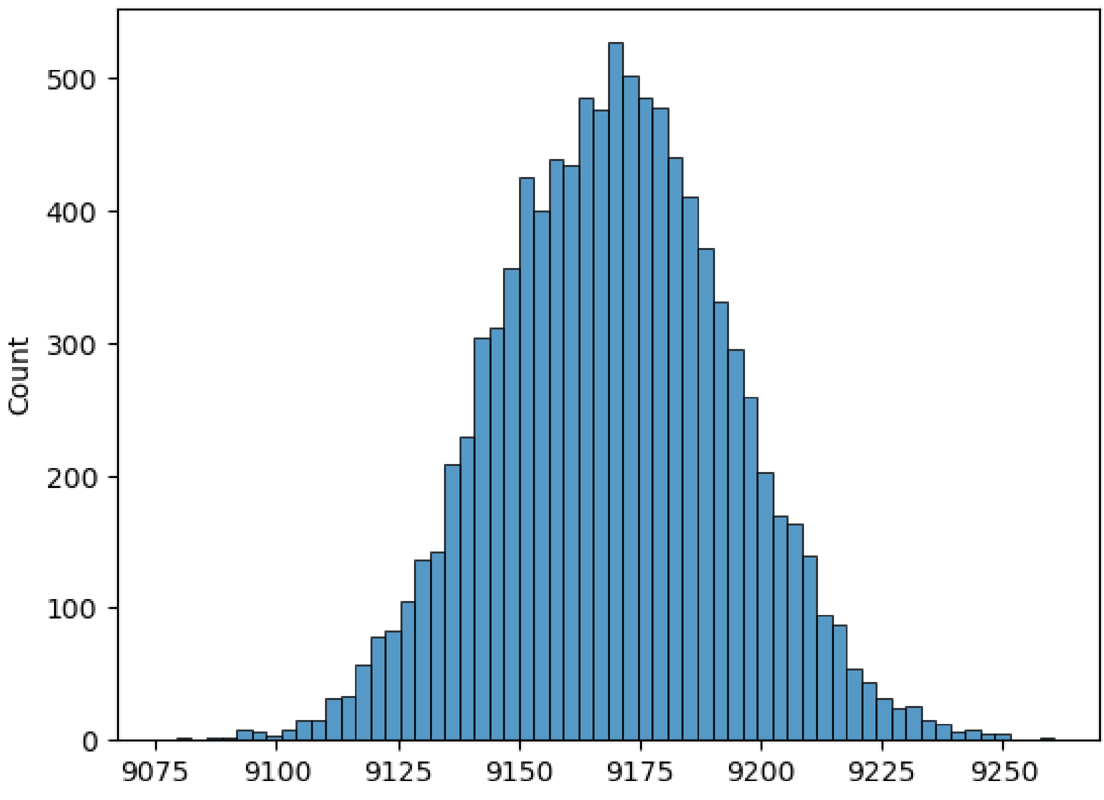
In [52]:
```
stats.norm.interval(.95,loc = mu_sample_3000, scale = sigma) # Confiden
```

Out[52]: (9167.569426598993, 9174.013292867674)

Sample for 30000 Age 18-25

In [53]:
```
sample_age_30000 = [np.mean(df_age_25['Purchase'].sample(30000)) for i i
sns.histplot(sample_age_30000)
```

Out[53]: <Axes: ylabel='Count'>



In [54]:
```
mu_sample_30000 = np.mean(sample_age_30000)     # Mean
mu_sample_30000
```

Out[54]: 9169.454389543334

In [55]:
```
sigma_30000 = np.std(sample_age_30000)# std Dev
sigma_30000
```

Out[55]: 24.367038706569982

In [56]:
```
sigma = sigma_30000 /(30000**0.5) # Updated Std Dev
sigma
```

Out[56]: 0.1406831635659221

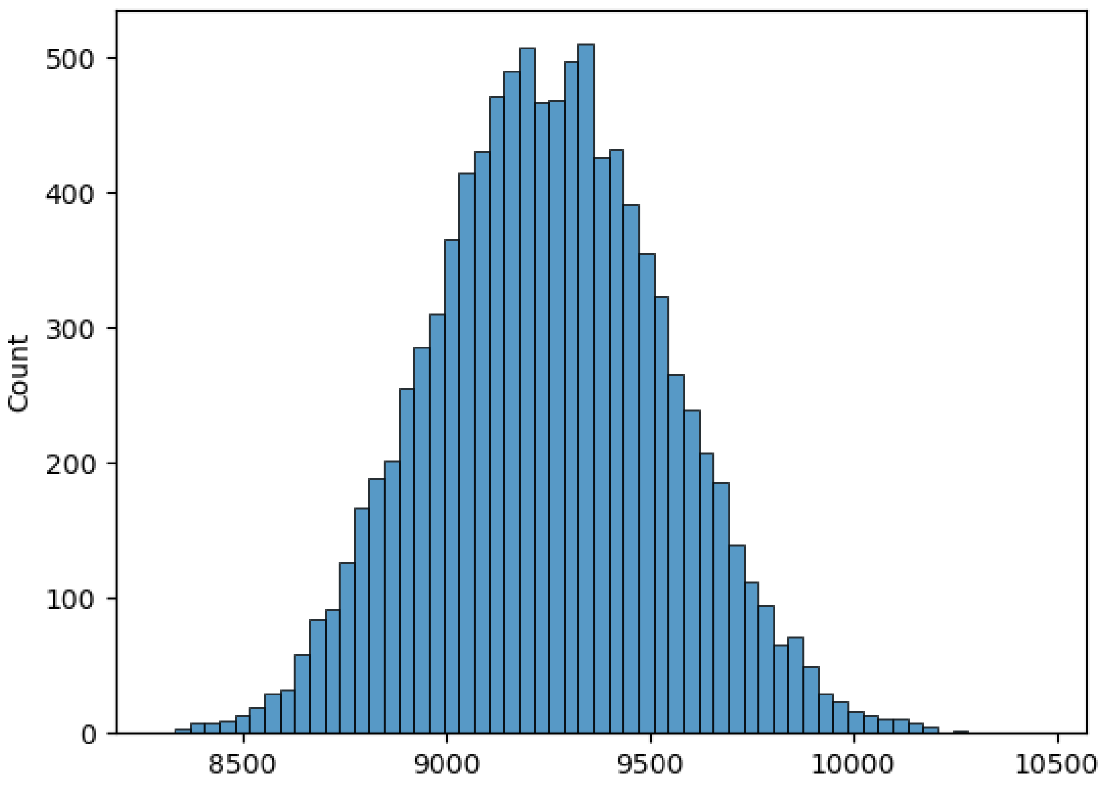
In [57]:
```
stats.norm.interval(.95,loc = mu_sample_30000, scale = sigma) # Confider
```

Out[57]: (9169.178655609514, 9169.730123477155)

Sample for 300 Age 26-35

In [58]:
```
sample_age_300 = [np.mean(df_age_35['Purchase'].sample(300)) for i in ra
sns.histplot(sample_age_300)
```

Out[58]: <Axes: ylabel='Count'>



In [63]:
```
mu_sample_300 = np.mean(sample_age_300)     # Mean
mu_sample_300
```

Out[63]: 9252.296861

In [64]:
```
sigma_300 = np.std(sample_age_300)# std Dev
sigma_300
```

Out[64]: 291.0302928120122

In [65]:
```
sigma = sigma_300 /(300**0.5) # Updated Std Dev
sigma
```

Out[65]: 16.802641789735084

In [66]:
```
stats.norm.interval(.95,loc = mu_sample_300, scale = sigma) # Confidence
```

Out[66]: (9219.364288246992, 9285.22943375301)

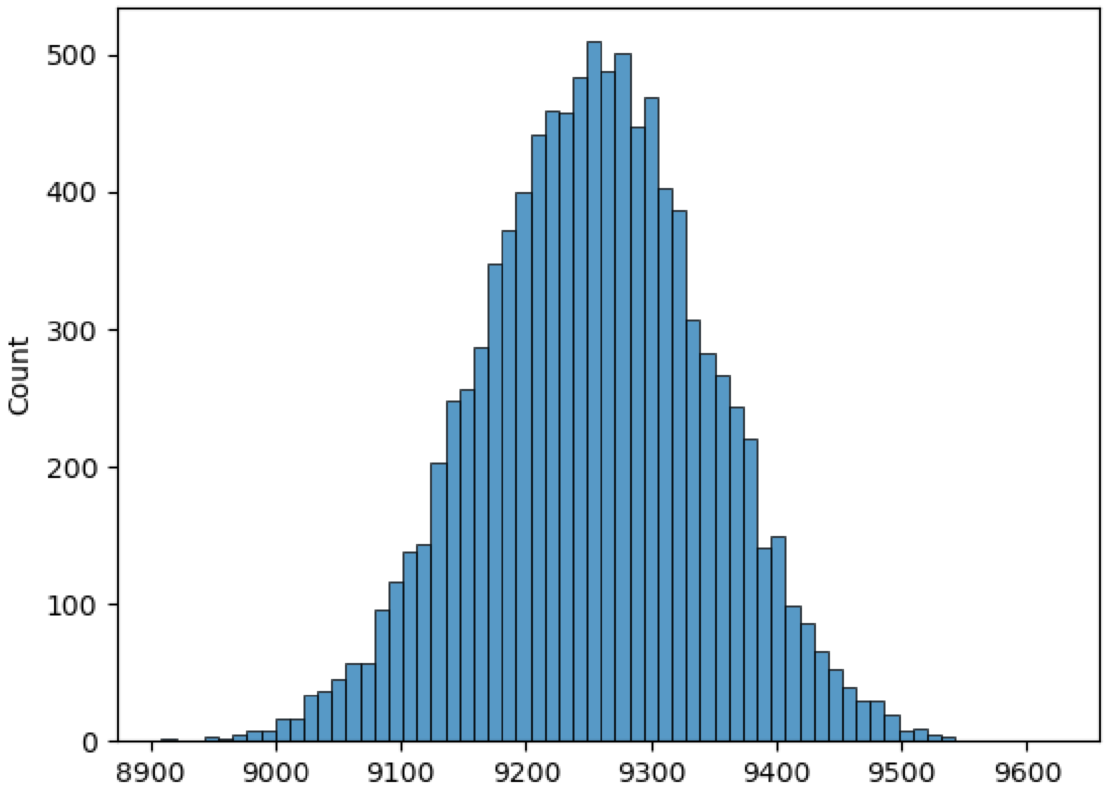
In [ ]:

Sample for 3000 Age 26-35

In [26]:
```
df_age_35 = df[df['Age']=='26-35']
```

In [37]:
```
sample_age_3000 = [np.mean(df_age_35['Purchase'].sample(3000)) for i in
sns.histplot(sample_age_3000)
```

Out[37]: <Axes: ylabel='Count'>



In [38]:
```
mu_sample_3000 = np.mean(sample_age_3000)    # Mean
mu_sample_3000
```

Out[38]: 9253.5212343

In [39]:
```
sigma_3000 = np.std(sample_age_3000)# std Dev
sigma_3000
```

Out[39]: 91.54200636155636

In [40]:
```
sigma = sigma_3000 /(3000**0.5) # Updated Std Dev
sigma
```

Out[40]: 1.6713207281168612

In [41]:
```
stats.norm.interval(.95,loc = mu_sample_3000, scale = sigma) # Confidenc
```

Out[41]: (9250.245505866276, 9256.796962733724)

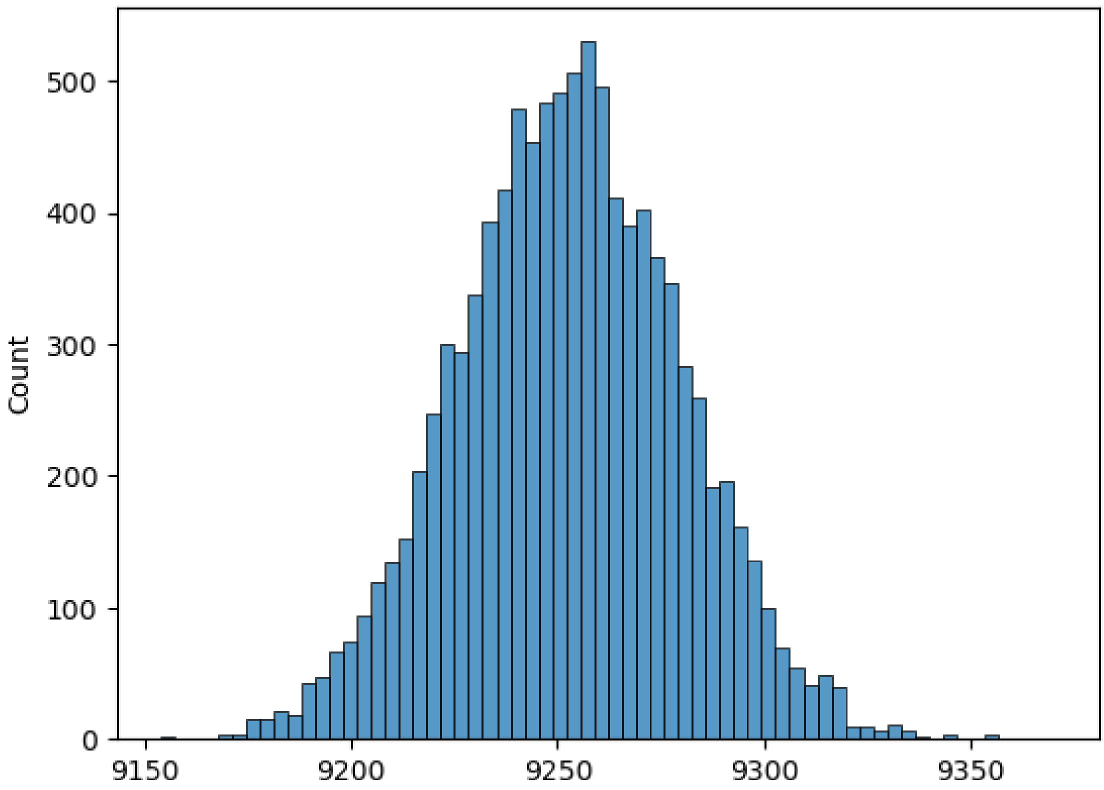
Insight:

A speculative observation suggests that individuals between the ages of 30 to 40 might exhibit higher spending patterns, potentially due to increased income during this life stage. Conversely, as people age, a presumed inclination towards saving more, perhaps for family-related expenses, may contribute to a decline in purchases. It's important to note that this conclusion is based on assumptions, as income data for customers is not available.

Sample for 30000 Age 26-35

In [67]:
```
sample_age_30000 = [np.mean(df_age_35['Purchase'].sample(30000)) for i i
sns.histplot(sample_age_30000)
```

Out[67]: <Axes: ylabel='Count'>



In [68]:
```
mu_sample_30000 = np.mean(sample_age_30000)     # Mean
mu_sample_30000
```

Out[68]: 9252.574935233331

In [69]:
```
sigma_30000 = np.std(sample_age_30000)# std Dev
sigma_30000
```

Out[69]: 27.16907989464619

In [70]:
```
sigma = sigma_30000 /(30000**0.5) # Updated Std Dev
sigma
```

Out[70]: 0.1568607559080843

In [72]:
```
stats.norm.interval(.95,loc = mu_sample_30000, scale = sigma) # Confider
```

Out[72]: (9252.267493801164, 9252.882376665499)

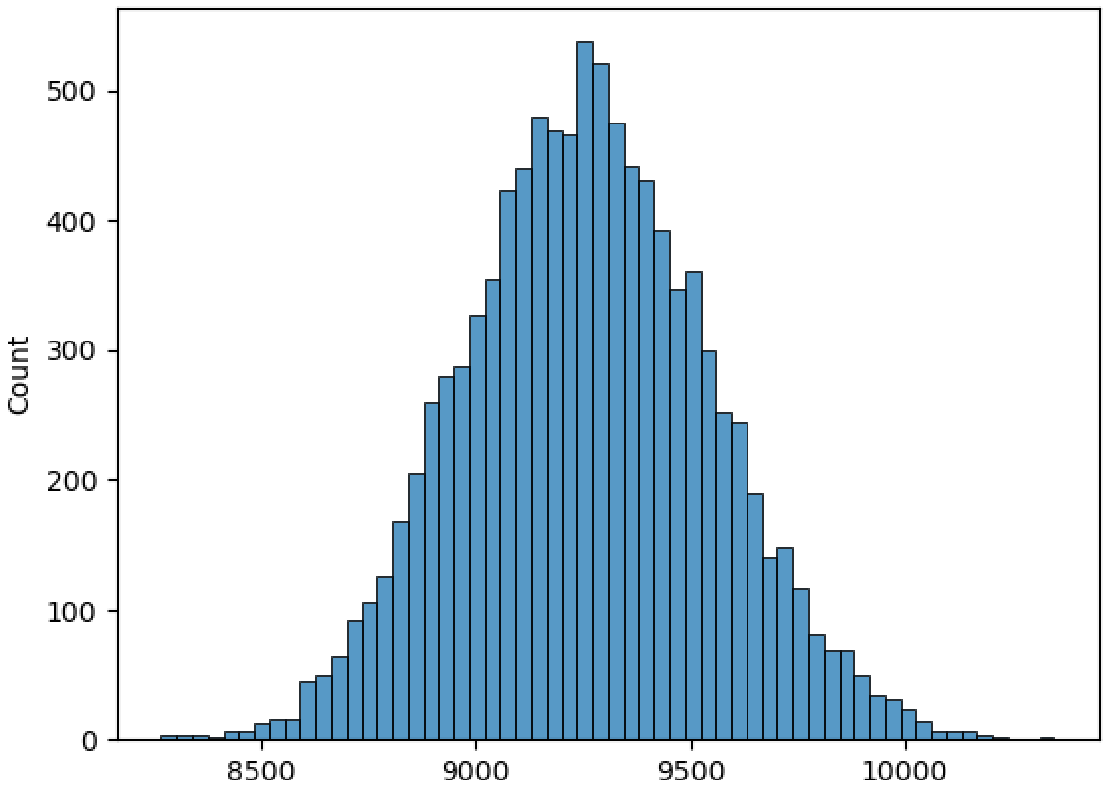
Insight:

The observed trend reveals that as the **sample size increases, there is a noticeable reduction in the range of Confidence Intervals.**

Sample for 300 Age 35-45

In [74]:
```
sample_age_300 = [np.mean(df_age_35['Purchase'].sample(300)) for i in ra
sns.histplot(sample_age_300)
```

Out[74]: <Axes: ylabel='Count'>



In [75]:
```
mu_sample_300 = np.mean(sample_age_300)    # Mean
mu_sample_300
```

Out[75]: 9258.518775

In [76]:
```
sigma_300 = np.std(sample_age_300)# std Dev
sigma_300
```

Out[76]: 288.2554596954553

In [77]:
```
sigma = sigma_300 /(300**0.5) # Updated Std Dev
sigma
```

Out[77]: 16.642436725055042

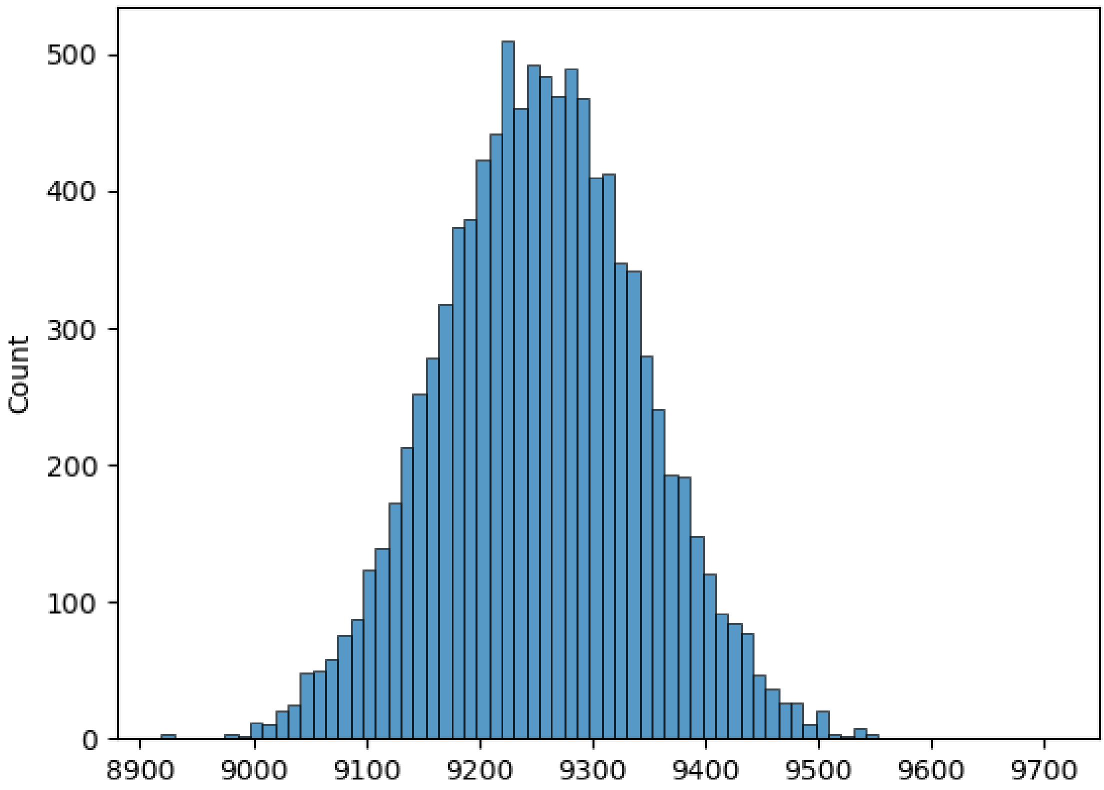
In [78]:
```
stats.norm.interval(.95,loc = mu_sample_300, scale = sigma) # Confidence
```

Out[78]: (9225.900198403906, 9291.137351596095)

Sample for 3000 Age 35-45

In [79]:
```
sample_age_3000 = [np.mean(df_age_35['Purchase'].sample(3000)) for i in
sns.histplot(sample_age_3000)
```

Out[79]: <Axes: ylabel='Count'>



In [80]:
```
mu_sample_3000 = np.mean(sample_age_3000)     # Mean
mu_sample_3000
```

Out[80]: 9253.7073407

In [81]:
```
sigma_3000 = np.std(sample_age_3000)# std Dev
sigma_3000
```

Out[81]: 89.7942779685411

In [82]:
```
sigma = sigma_3000 /(3000**0.5) # Updated Std Dev
sigma
```

Out[82]: 1.639411719275304

In [83]:
```
stats.norm.interval(.95,loc = mu_sample_3000, scale = sigma) # Confidenc
```

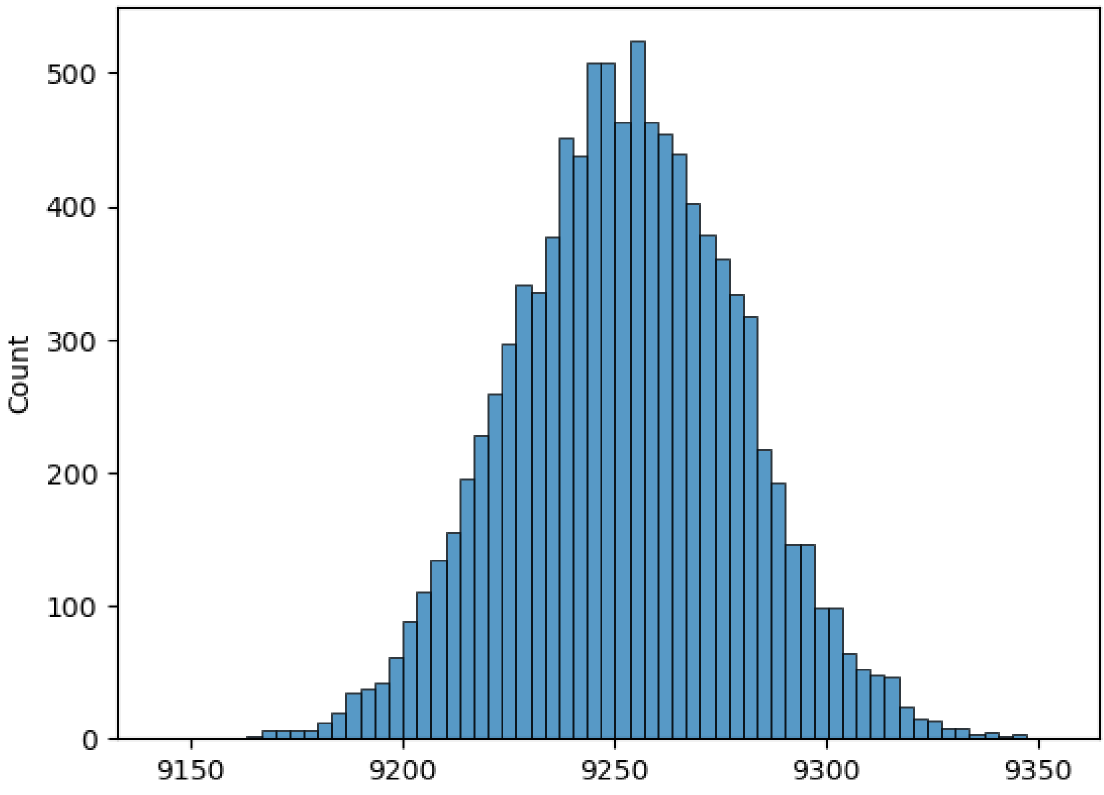
Out[83]: (9250.494152774389, 9256.920528625613)

Sample for 30000 Age 36-45

In [73]:
```
df_age_45 = df[df['Age']=='36-45']
```

In [43]:
```
sample_age_30000 = [np.mean(df_age_35['Purchase'].sample(30000)) for i :
sns.histplot(sample_age_30000)
```

Out[43]: <Axes: ylabel='Count'>



In [44]:
```
mu_sample_30000 = np.mean(sample_age_30000)    # Mean
mu_sample_30000
```

Out[44]: 9252.84466069

In [45]:
```
sigma_30000 = np.std(sample_age_30000)# std Dev
sigma_30000
```

Out[45]: 26.92999355553742

In [46]:
```
sigma = sigma_30000 /(30000**0.5) # Updated Std Dev
sigma
```

Out[46]: 0.15548039028564417

In [47]:
```
stats.norm.interval(.95,loc = mu_sample_30000, scale = sigma) # Confiden
```

Out[47]:
```
(9252.539924724737, 9253.149396655263)
```

Insight:

1.The confidence intervals for the average amount spent by different age groups, **specifically for ages between 35-45 and ages between 18-25, do not overlap. This suggests a statistically significant difference in the average amount spent between these two age groups.**

2.Walmart can leverage this conclusion to **tailor marketing strategies, promotions, or product offerings based on the spending behaviors of distinct age groups**. For example, they could design targeted advertising campaigns or introduce products that appeal specifically to the spending preferences of customers in the 35-45 age range and separately for those in the 18-25 age range.

3.This personalized approach **can enhance customer engagement and potentially increase sales** by catering to the unique needs and preferences of each age group.

Recommendations

1. Targeted Marketing: Given that certain product categories attract more consumers, Walmart can tailor its marketing strategies to focus on these popular categories. This includes designing promotions, advertisements, and in-store displays to highlight products from categories 1, 5, and 8, which are more likely to attract attention.
2. Demographic Considerations: The data suggests that the age group between 26-35 contributes the most to purchases across different categories. Walmart could consider tailoring promotions and offerings to cater specifically to this age bracket. Additionally, considering the significant presence of males in every category, marketing efforts can be crafted to specifically target this demographic.
3. Optimizing Stock: The insight about outliers in the Product Category and Purchase columns indicates potential irregularities or unique patterns. Walmart can investigate these outliers to understand if there are specific products or purchases that deviate significantly from the norm. This can help in optimizing stock, ensuring popular products are well-stocked, and addressing potential issues with less popular items.
4. Customer Experience Improvements: By leveraging insights into the relationship between variables like Age, MaritalStatus, and Purchase, Walmart can enhance the overall customer experience. For instance, tailoring promotions or loyalty programs based on age groups or marital status can create a more personalized shopping experience.
5. Data-Driven Decision-Making: Encourage a data-driven culture within Walmart, where decisions are informed by insights derived from data analysis. Regularly updating and analyzing customer data can provide ongoing guidance for marketing, sales, and inventory management strategies.

In [ ]: