walmart-my-casestudy-1

May 12, 2024

[75]:
```
import matplotlib.pyplot as plt
import seaborn as sns

# Import libraries
import numpy as np
import pandas as pd
from scipy.stats import norm
```

[4]:
```
# Loading the dataset
df = pd.read_csv('walmart_data.csv')
df.head()
```

[4]:

	User_ID	Product_ID	Gender	Age	Occupation	City_Category	Stay_In_Current_City_Years	Marital_Status	Product_Category	Purchase
0	1000001	P00069042	F	0-17	10	A	2	0	3	8370
1	1000001	P00248942	F	0-17	10	A	2	0	1	15200
2	1000001	P00087842	F	0-17	10	A	2	0	12	1422
3	1000001	P00085442	F	0-17	10	A	2	0	12	1057
4	1000002	P00285442	M	55+	16	C	4+	0	8	7969

[ ]:
```
# getting the counts of rows and columns in the dataset
df.shape
```

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

no of rows : 550068
no of columns : 10

[ ]:
```
# getting the information of df dataset
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
```

[ ]:
```
# no of unique values in each column
for i in df.columns:
  print(i,':',df[i].nunique())
```

```
User_ID : 5891
Product_ID : 3631
Gender : 2
Age : 7
Occupation : 21
City_Category : 3
Stay_In_Current_City_Years : 5
Marital_Status : 2
Product_Category : 20
Purchase : 18105
```

according to above result, it seems Age column is binned by 7 bins.

[ ]:
```
# all the 7 bins of Age
df['Age'].unique()
```

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

[ ]:
```
# Get the unique values of the 'City_Category' column
unique_cities = df['City_Category'].unique()
unique_cities
```

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

[ ]:
```
# Get the unique values of the 'Stay_In_Current_City_Years' column
unique_stay_years = df['Stay_In_Current_City_Years'].unique()
```

```
unique_stay_years
```

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

```
# Get the unique values of the 'Marital_Status' column
df['Marital_Status'].unique()
```

```
array([0, 1])
```

```
# Get the unique values of the 'Product_Category' column
df['Product_Category'].unique()
```

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

```
# checking for null values
df.isnull().sum()
```

```
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
```

From above analysis. we see that there is no null values.

```
df.head()
```

	User_ID	Product_ID	Gender	Age	Occupation	City_Category	Stay_In_Current_City_Years	Marital_Status	Product_Category	Purchase
0	1000001	P00069042	F	0-17	10	A	2	0	3	8370
1	1000001	P00248942	F	0-17	10	A	2	0	1	15200
2	1000001	P00087842	F	0-17	10	A	2	0	12	1422
3	1000001	P00085442	F	0-17	10	A	2	0	12	1057
4	1000002	P00285442	M	55+	16	C	4+	0	8	7969

[ ]:
```
# checking the outliers in Purchase column
# as it is the most significant integer column
df['Purchase'].describe()
```

[ ]:
```
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
```

[ ]:
```
df['Purchase'].mode()
```

[ ]:
```
0    7011
Name: Purchase, dtype: int64
```

[ ]:
```
df['Purchase'].median()
```

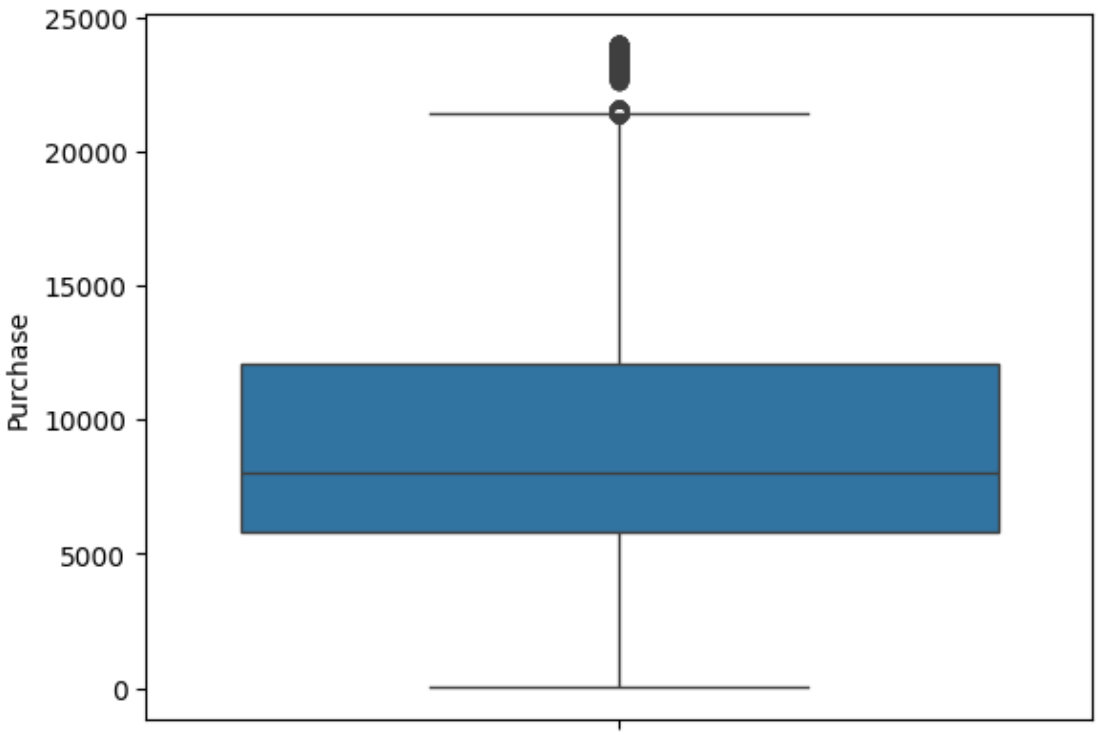
[ ]:
```
8047.0
```

Based on the above summary we can observe that on Purchase:
mean : 9263.968713
median : 8047.0
mode : 7011

As mean is greater than median, median is greater than mode.
So. Purchase column follows the ***right skewed distribution***.

As mean != median and from the bellow plot we know that Purchase column has ***outliers***.

[ ]:
```
# checking outliers by using boxplot on Purchase column
sns.boxplot(data = df['Purchase'])
plt.show()
```



[20]:
```
df_age = df['Age'].describe()
df_age
```

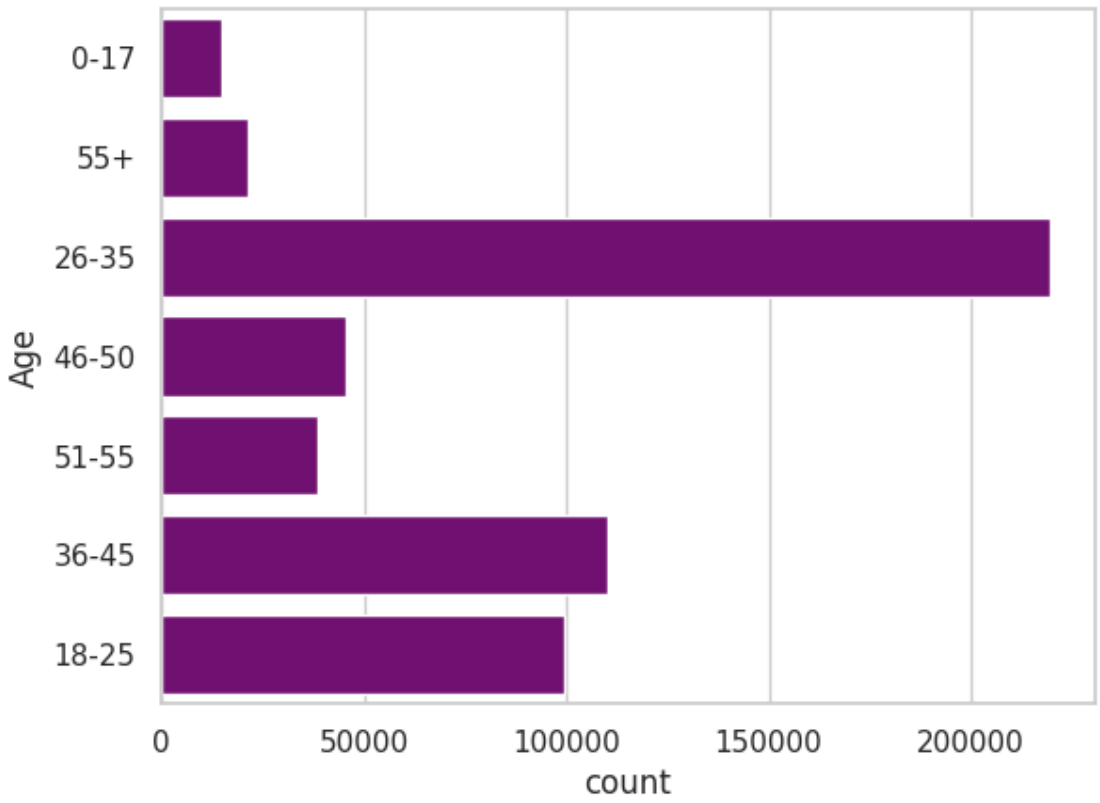
[20]:
```
count     550068
unique         7
top        26-35
freq      219587
Name: Age, dtype: object
```

[25]:
```
df_age['freq']*100/df_age['count']
```

[25]:
```
39.91997353054531
```

From above analysis:
Almost 40 percent people, who purchase any product there age is in the range of 26-35 years

[38]:
```
sns.countplot(data = df['Age'], color = "purple")
plt.show()
```



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

[ ]:

	User_ID	Product_ID	Gender	Age	Occupation	City_Category	Stay_In_Current_City_Years	Marital_Status	Product_Category	Purchase
0	1000001	P00069042	F	0-17	10	A	2	0	3	8370
1	1000001	P00248942	F	0-17	10	A	2	0	1	15200
2	1000001	P00087842	F	0-17	10	A	2	0	12	1422
3	1000001	P00085442	F	0-17	10	A	2	0	12	1057
4	1000002	P00285442	M	55+	16	C	4+	0	8	7969

[6]:
```
Total_transaction_amount = df['Purchase'].sum()
Total_transaction_amount
```

[6]: 5095812742

[13]:
```
# Total amount transaction by Male amd Female
data = df.groupby('Gender').sum('Purchase')
d1 = data['Purchase']
d1
```

[13]:
```
Gender
F    1186232642
M    3909580100
Name: Purchase, dtype: int64
```

[10]:
```
# Total no of transaction
df['Product_ID'].count()
```

[10]:
```
550068
```

[16]:
```
# avarage Male transaction
d1['M']/df['Product_ID'].count()
```

[16]:
```
7107.448715431547
```

[17]:
```
# avarage Female transaction
d1['F']/df['Product_ID'].count()
```

[17]:
```
2156.519997527578
```

From above analysis: 1. the avarage purchase amount for every transaction made by Male is Far grater then transaction made by Female. 2. Male expanses is grater than Female Expanses.

[18]:
```
# Mean of Purchase amount
mu_population = df['Purchase'].sum()/df['Product_ID'].count()
mu_population
```

[18]:
```
9263.968712959126
```

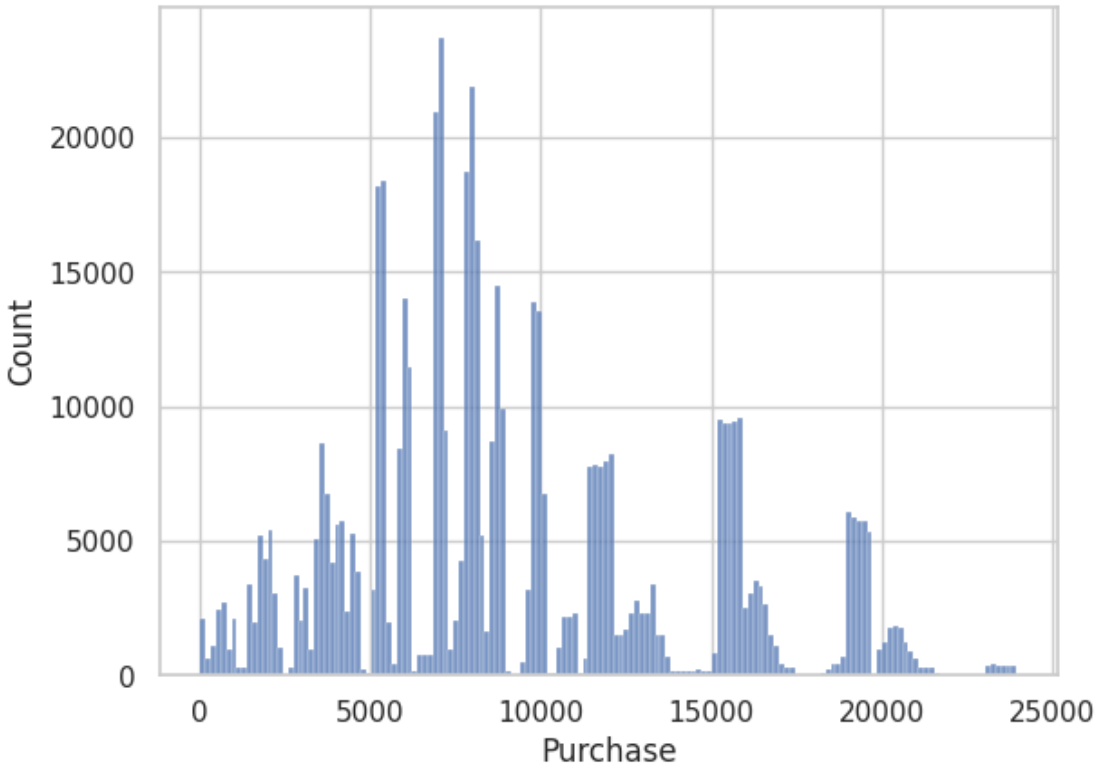
0.0.1 Central Limit Theoream

[40]:
```
# only getting Gender & Purchase columns
df_clt = df[['Gender','Purchase']]
df_clt.head()
```

[40]:

	Gender	Purchase
0	F	8370
1	F	15200
2	F	1422
3	F	1057
4	M	7969

[42]:
```
# checking the distribution of purchase column
sns.histplot(df_clt['Purchase'])
plt.show()
```



From the above chart, we can clearly understand that the distributaion is **RIGHT SKEWED**

[44]:
```
# mean of the entire population
mu = df_clt['Purchase'].mean()
mu
```

[44]: 9263.968712959126

[46]:
```
# getting the standard deviation of the population
sigma = df_clt['Purchase'].std()
sigma
```

[46]: 5023.065393820582

We will now randomly select 20 samples and determine the average of these samples, taking **sample size(n) = 20**

[60]:
```
# taking the 20 samples randomly
df_clt['Purchase'].sample(20)
```

[60]:
```
515726     6934
232605     5854
453095    15296
71366     11461
262276     9917
517140     8877
468526     4161
166709    11662
395797    10655
518066     6878
231970     4231
134342    19346
333717     7018
154079     7066
255454     1462
250845     2310
274600     6160
397307     7074
95426      6920
508786    15860
Name: Purchase, dtype: int64
```

[61]:
```
# geting the mean of 20 samples
np.mean(df_clt['Purchase'].sample(20))
```

[61]:
```
10806.45
```

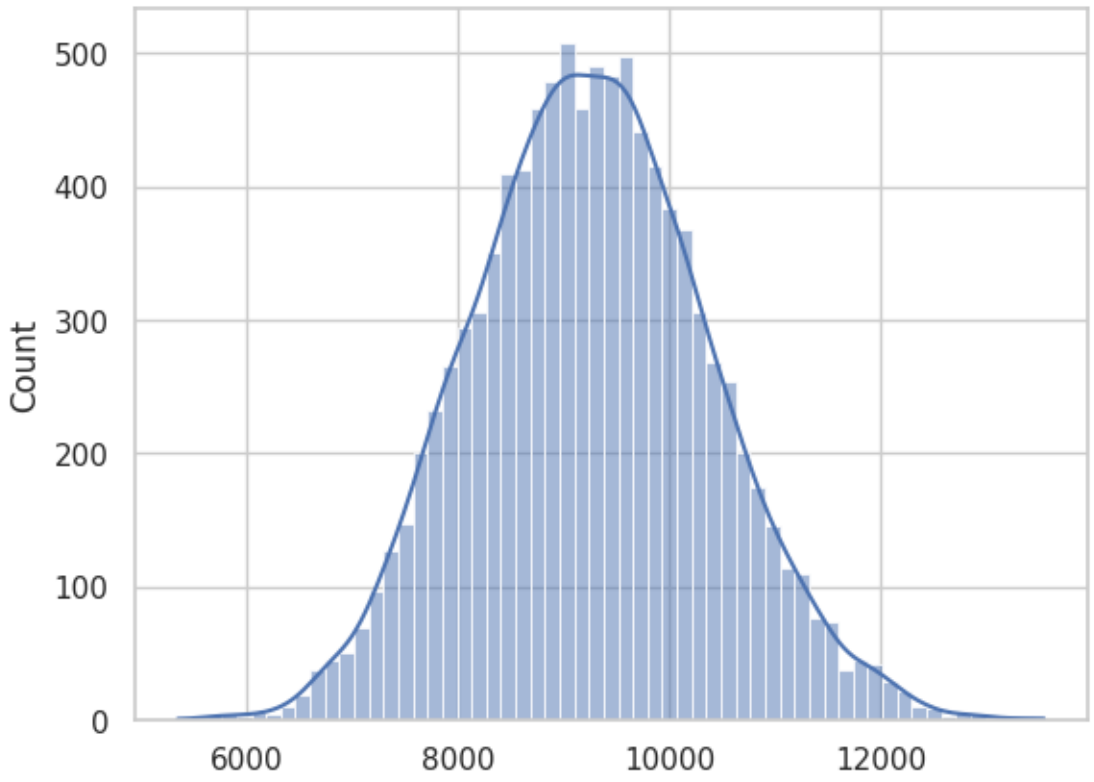
Let's repeat this process 10,000 times so we will get sample means of 10,000 unique ***samples (size = 20)***.

We will plot the distributions of these 10,000 sample means to see if they follow the normal distribution.

[62]:
```
sample_20 = [np.mean(df_clt['Purchase'].sample(20)) for i in range(10000) ]
```

[63]:
```
# creating the distribution of 10000 sample means
sns.histplot(sample_20, kde=True)
```

[63]:
```
<Axes: ylabel='Count'>
```



[67]:
```
#getting mean of 20 samples taken 10000 times
np.mean(sample_20)
```

[67]: 9252.857875

[67]: 9252.857875

[66]:
```
#getting Standard deviation of 20 samples taken 10000 times
np.std(sample_20)
```

[66]: 1115.5243114272473

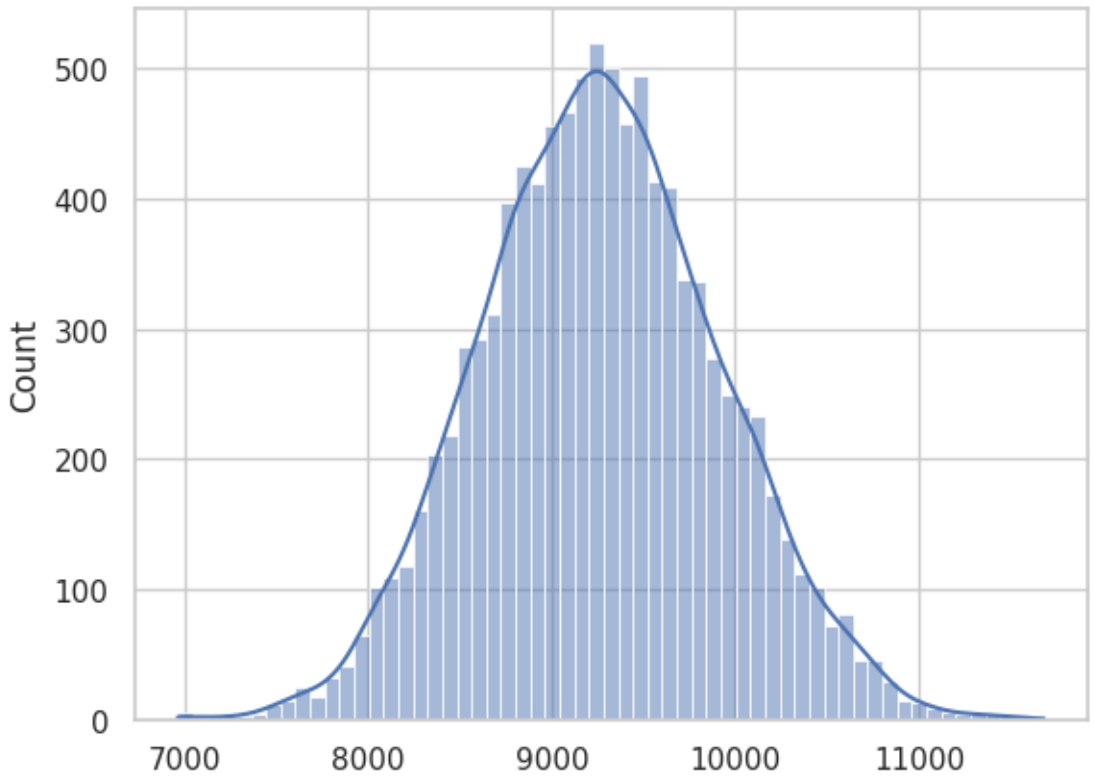
Let's repeat this process 10,000 times so we will get sample means of 10,000 unique samples **(size = 60)**.

We will plot the distributions of these 10,000 sample means to see if they follow the normal distribution.

[68]:
```
sample_60 = [np.mean(df_clt['Purchase'].sample(60)) for i in range(10000) ]
```

[69]:
```
sns.histplot(sample_60, kde=True)
```

[69]: <Axes: ylabel='Count'>



[70]:
```
#getting mean of 60 samples taken 10000 times
np.mean(sample_60)
```

[70]: 9262.935776666667

[71]:
```
#getting standard deviation of 60 samples taken 10000 times
np.std(sample_60)
```

[71]: 648.6580028163079

We can clearly see that as we increase the number of samples from 20 to 60, the sample means come closer to the actual mean and the standard deviation becomes less

0.0.2 Let's compare the statistics of population data and sample data to observe some patterns

[74]:
```
# population mean
mu = df_clt['Purchase'].mean()

# population SD
sigma = df_clt['Purchase'].std()
```

```
# mean of sample distributions having sample size = 20
mu_20 = np.mean(sample_20)

# SD of sample distributions having sample size = 20
sigma_20 = np.std(sample_20)

# mean of sample distributions having sample size = 60
mu_60 = np.mean(sample_60)

# SD of sample distributions having sample size = 20
sigma_60 = np.std(sample_60)
print(mu, mu_20, mu_60)
print(sigma, sigma_20, sigma_60)
```

```
9263.968712959126 9252.857875 9262.935776666667
5023.065393820582 1115.5243114272473 648.6580028163079
```

Observations:
As we increase the sample size, the SD of sample means decreases.
Mean of the sample_60 distribution is nearly equal to the mean of the population

0.0.3 Confidence Interval

[76]:
```
# Sample mean and standard deviation
mu_60 = 9262.935776666667 # Example sample mean
sigma_60 = 648.6580028163079    # Example sample standard deviation
sample_size = 60 # Example sample size

# Confidence levels
confidence_levels = [0.90, 0.95, 0.99]

for confidence_level in confidence_levels:
    # Calculate z-score based on confidence level
    z_score = norm.ppf(1 - (1 - confidence_level) / 2)

    # Calculate standard error of the mean
    standard_error = sigma_60 / np.sqrt(sample_size)

    # Calculate margin of error
    margin_of_error = z_score * standard_error

    # Calculate confidence interval
    lower_bound = mu_60 - margin_of_error
    upper_bound = mu_60 + margin_of_error

    # Print results
    print(f"Confidence Level: {confidence_level * 100}%")
    print(f"Confidence Interval: [{lower_bound:.2f}, {upper_bound:.2f}]")
```

```
    print(f"Width of Interval: {2 * margin_of_error:.2f}\n")
```

```
Confidence Level: 90.0%
Confidence Interval: [9125.19, 9400.68]
Width of Interval: 275.48

Confidence Level: 95.0%
Confidence Interval: [9098.81, 9427.07]
Width of Interval: 328.26

Confidence Level: 99.0%
Confidence Interval: [9047.23, 9478.64]
Width of Interval: 431.41
```

0.0.4 confidence intervals of average male

[77]:
```
df.head()
```

[77]:

	User_ID	Product_ID	Gender	Age	Occupation	City_Category	Stay_In_Current_City_Years	Marital_Status	Product_Category	Purchase
0	1000001	P00069042	F	0-17	10	A	2	0	3	8370
1	1000001	P00248942	F	0-17	10	A	2	0	1	15200
2	1000001	P00087842	F	0-17	10	A	2	0	12	1422
3	1000001	P00085442	F	0-17	10	A	2	0	12	1057
4	1000002	P00285442	M	55+	16	C	4+	0	8	7969

[83]:
```
df_m = df[df['Gender']=='M']['Purchase']
df_m
```

[83]:
```
4          7969
5         15227
6         19215
7         15854
8         15686
          ...
550057       61
550058      121
550060      494
550062      473
550063      368
```

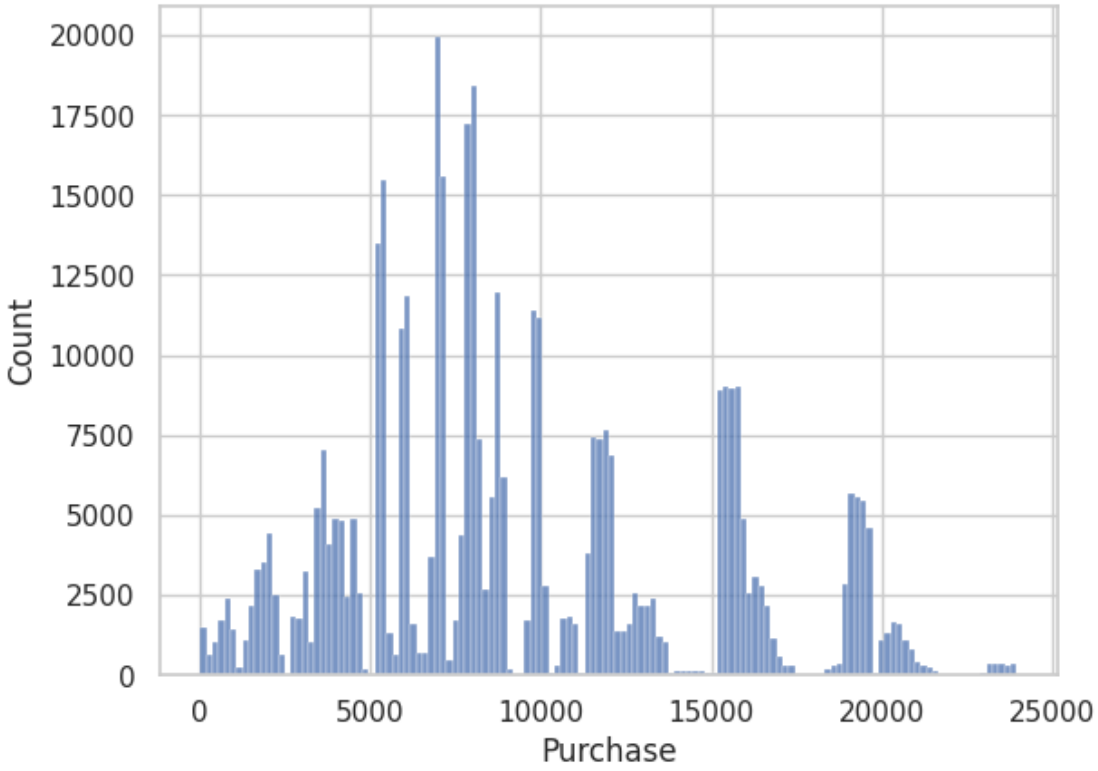
```
Name: Purchase, Length: 414259, dtype: int64
```

[94]:
```
# population mean, standard deviation for male
print(df_m.mean(),df_m.std())
```

```
9437.526040472265 5092.18620977797
```

[84]:
```
# creating the distribution of Purchased amount by Male
sns.histplot(df_m)
```

[84]:
```
<Axes: xlabel='Purchase', ylabel='Count'>
```



above chart shows that the above chart is right skewed

[85]:
```
# taking the 40 samples randomly
df_m.sample(40)
```

[85]:
```
4694       7404
505796    18362
489765     5893
86921      3472
321580     5971
```

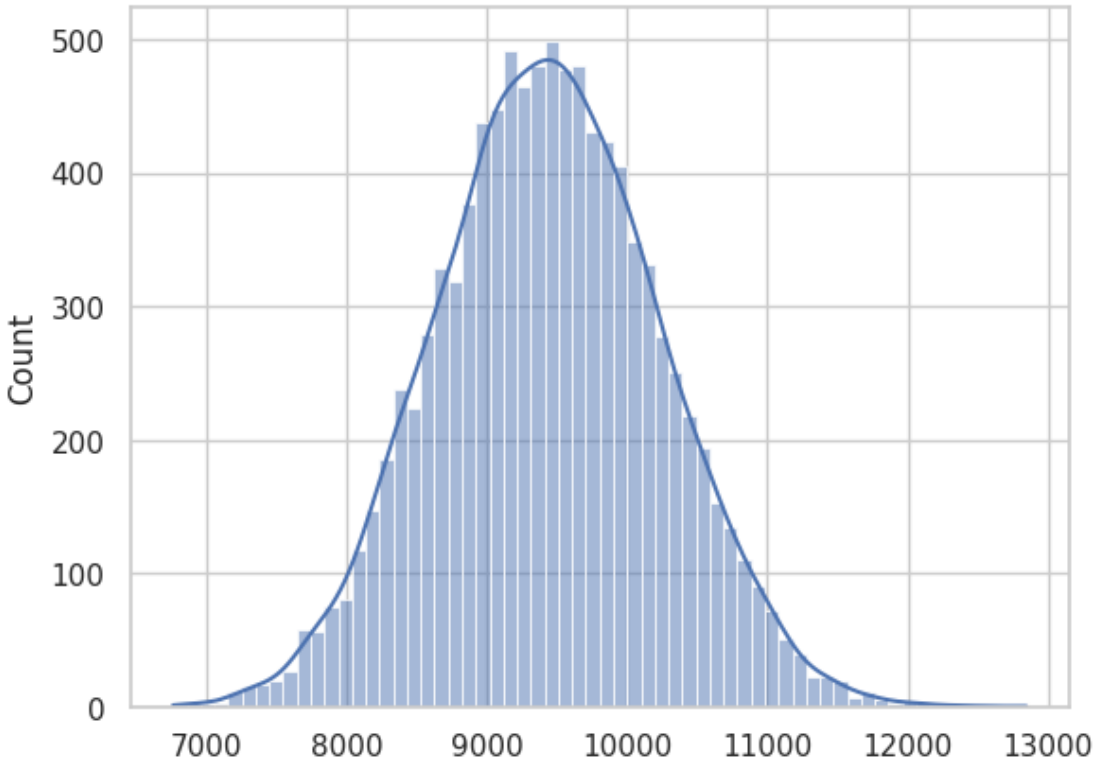
```
316283    12887
298757     1680
209992     8613
205272     6897
88430      3677
302457    11064
325440     1504
234028    11417
408834     6954
423503     6994
66191      6084
517638     9959
294018     6965
393736     9970
524751     6430
350856    19277
219055     3736
402211    16330
176809    15191
361026     4649
271726    12941
534369    12228
207330    15391
296467    11822
3299       5433
492197     7083
491601     9913
151249     1791
456593    12022
196809    10000
349903    20542
143166     8095
468024    15281
148958     4230
230732     8769
Name: Purchase, dtype: int64
```

process 10,000 times, will get sample means of 10,000 unique samples (size = 40).

[88]:
```
# geting the mean of 40 samples
s_40 = [np.mean(df_m.sample(40)) for i in range(10000) ]
```

[90]:
```
sns.histplot(s_40, kde = True)
```

[90]:
```
<Axes: ylabel='Count'>
```



[95]:
```
#getting mean of 40 samples taken 10000 times
m_40 = np.mean(s_40)
m_40
```

[95]: 9434.966155

[92]:
```
#getting Standard eviation of 40 samples taken 10000 times
np.std(s_40)
```

[92]: 794.6407996481121

[98]:
```
# Confidence interval purchased from Male
sample_size = 40 # Example sample size

# Confidence levels
confidence_levels = [0.90, 0.95, 0.99]

for confidence_level in confidence_levels:
    # Calculate z-score based on confidence level
    z_score = norm.ppf(1 - (1 - confidence_level) / 2)

    # Calculate standard error of the mean
```

```
    standard_error = s_40 / np.sqrt(sample_size)

    # Calculate margin of error
    margin_of_error = z_score * standard_error

    # Calculate confidence interval
    lower_bound = m_40 - margin_of_error
    upper_bound = m_40 + margin_of_error

    # Print results
    lower_bound_list = lower_bound.tolist()
    upper_bound_list = upper_bound.tolist()

    # Print results
    print(f"Confidence Level: {confidence_level * 100}%")
    print(f"Confidence Interval: [{lower_bound_list[0]:.2f},␣
↪{upper_bound_list[0]:.2f}]")
    print(f"Width of Interval: {2 * margin_of_error[0]:.2f}\n")
```

```
Confidence Level: 90.0%
Confidence Interval: [7044.83, 11825.10]
Width of Interval: 4780.27

Confidence Level: 95.0%
Confidence Interval: [6586.95, 12282.99]
Width of Interval: 5696.04

Confidence Level: 99.0%
Confidence Interval: [5692.03, 13177.90]
Width of Interval: 7485.87
```

0.1 Recomendation:

1 . purchase rate of 26-35 years people is very high , should recomend more product to them. 2. Pruchasing rate for male is higher than Female, should recomend more product to men. 2. as the female prchacer is less , we shold focus on female purchaser more