About Walmart

Walmart is an American multinational retail corporation that operates a chain of supercenters, discount departmental stores, and grocery stores from the United States. Walmart has more than 100 million customers worldwide.

Business Problem

The Management team at Walmart Inc. wants to analyze the customer purchase behavior (specifically, purchase amount) against the customer's gender and the various other factors to help the business make better decisions. They want to understand if the spending habits differ between male and female customers: Do women spend more on Black Friday than men? (Assume 50 million customers are male and 50 million are female).

```
# Import the libraries
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import plotly.express as px
import plotly.figure_factory as ff
from scipy.stats import norm
import warnings
warnings.filterwarnings('ignore')
```

```
# Read the file
df = pd.read_csv("walmart_data.txt")
df
```

	User_ID	Product_ID	Gender	Age	Occupation	City_Category	Stay_In_Current_City_Years	Marital_Status	Product_Category	P
0	1000001	P00069042	F	0-17	10	A	2	0	3	
1	1000001	P00248942	F	0-17	10	A	2	0	1	
2	1000001	P00087842	F	0-17	10	A	2	0	12	
3	1000001	P00085442	F	0-17	10	A	2	0	12	
4	1000002	P00285442	M	55+	16	C	4+	0	8	
...	...	...	...	...	...	...	...	...	...	
550063	1006033	P00372445	M	51-55	13	B	1	1	20	
550064	1006035	P00375436	F	26-35	1	C	3	0	20	

```
# Validate the 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
```

```
df.keys()
```

```
Index(['User_ID', 'Product_ID', 'Gender', 'Age', 'Occupation', 'City_Category',
       'Stay_In_Current_City_Years', 'Marital_Status', 'Product_Category',
       'Purchase'],
      dtype='object')
```

```
# validate the duplicate records
df.duplicated().sum(axis=0)
```

```
0
```

```
# identifying of misising / null values
df.isna().sum()
```

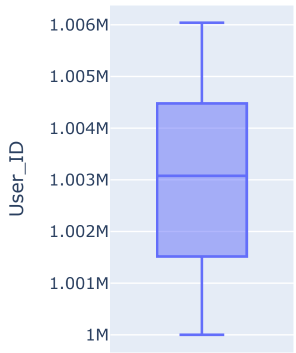
	0
User_ID	0
Product_ID	0
Gender	0
Age	0
Occupation	0
City_Category	0
Stay_In_Current_City_Years	0
Marital_Status	0
Product_Category	0
Purchase	0

dtype: int64

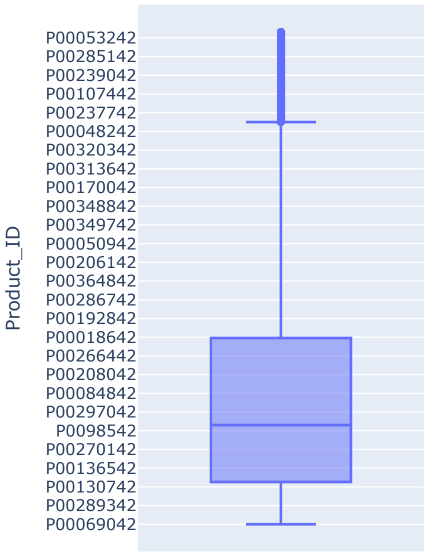
```
df.keys()
```

```
Index(['User_ID', 'Product_ID', 'Gender', 'Age', 'Occupation', 'City_Category',
       'Stay_In_Current_City_Years', 'Marital_Status', 'Product_Category',
       'Purchase'],
      dtype='object')
```

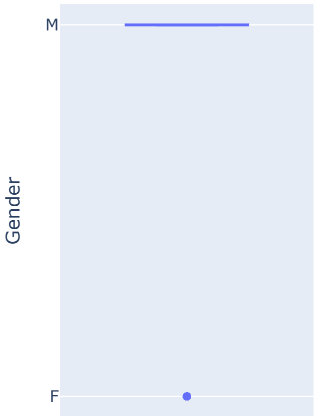
```
px.box(df, y="User_ID", width=300, height=400)
```



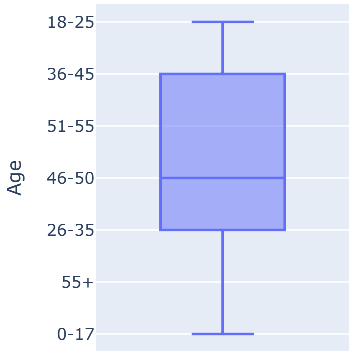

```
px.box(df,y="Product_ID",width=400,height=550)
```



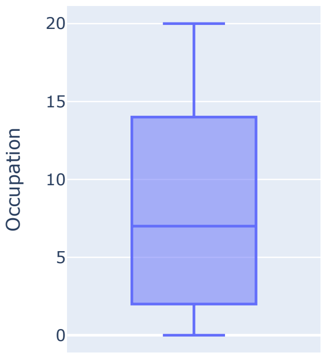

```
px.box(df,y="Gender",width=350,height=450)
```



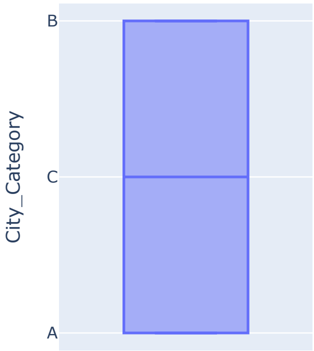

```
px.box(df,y="Age",width=350,height=400)
```



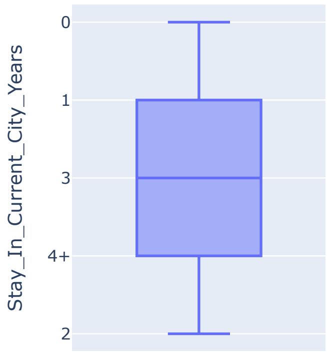

```
px.box(df,y="Occupation",width=350,height=400)
```



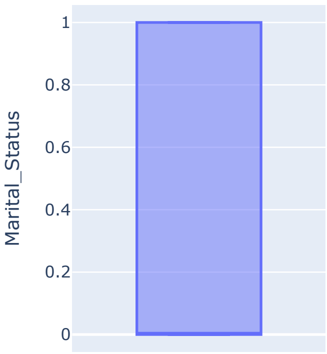

```
px.box(df,y="City_Category",width=350,height=400)
```



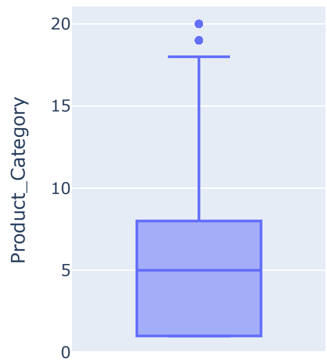

```
px.box(df,y="Stay_In_Current_City_Years",width=350,height=400)
```



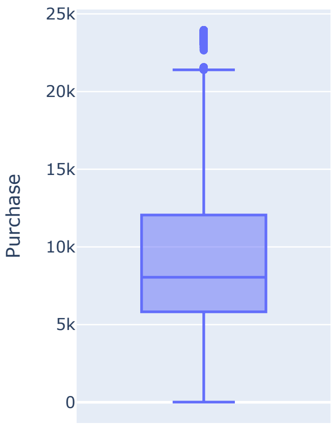

```
px.box(df,y="Marital_Status",width=350,height=400)
```




```
px.box(df,y="Product_Category",width=350,height=400)
```




```
px.box(df,y="Purchase",width=350,height=450)
```



Insights

From dataset to till here we conclude there are neither duplicate records nor a null value in the dataset

```
df.nunique()
```

	0
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

dtype: int64

```
# Glimpse of descriptive stats
df.describe()
```

	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

```
df.describe(include='object')
```

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

```
df.describe(include='all')
```

	User_ID	Product_ID	Gender	Age	Occupation	City_Category	Stay_In_Current_City_Years	Marital_Status	Product_(
count	5.500680e+05	550068	550068	550068	550068.000000	550068	550068	550068.000000	55006
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	2

```
df.groupby(["Marital_Status"])["Purchase"].describe()
```

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

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

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

```
df.groupby(["Age"])["Purchase"].describe()
```

Age	count	mean	std	min	25%	50%	75%	max
0-17	15102.0	8933.464640	5111.114046	12.0	5328.0	7986.0	11874.0	23955.0
18-25	99660.0	9169.663606	5034.321997	12.0	5415.0	8027.0	12028.0	23958.0
26-35	219587.0	9252.690633	5010.527303	12.0	5475.0	8030.0	12047.0	23961.0
36-45	110013.0	9331.350695	5022.923879	12.0	5876.0	8061.0	12107.0	23960.0
46-50	45701.0	9208.625697	4967.216367	12.0	5888.0	8036.0	11997.0	23960.0
51-55	38501.0	9534.808031	5087.368080	12.0	6017.0	8130.0	12462.0	23960.0
55+	21504.0	9336.280459	5011.493996	12.0	6018.0	8105.5	11932.0	23960.0

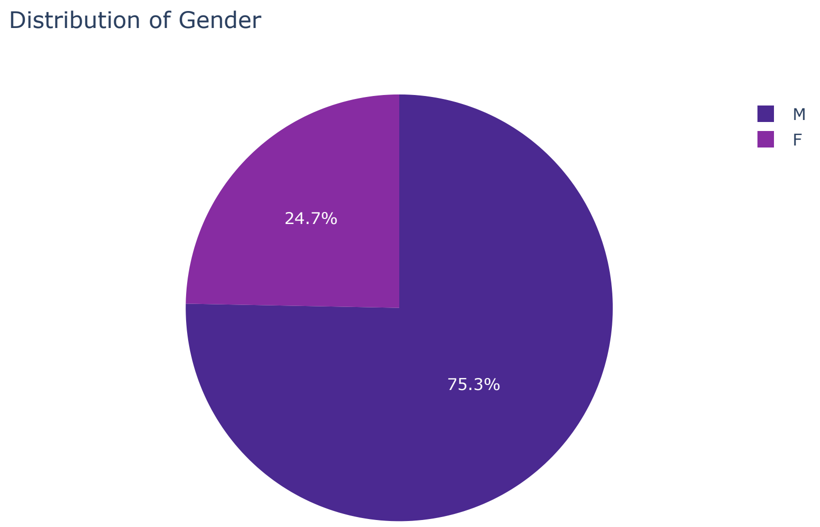
```
df.groupby(["City_Category"])["Purchase"].describe()
```

	count	mean	std	min	25%	50%	75%	max
City_Category								
A	147720.0	8911.939216	4892.115238	12.0	5403.0	7931.0	11786.0	23961.0
B	231173.0	9151.300563	4955.496566	12.0	5460.0	8005.0	11986.0	23960.0
C	171175.0	9719.920993	5189.465121	12.0	6031.5	8585.0	13197.0	23961.0

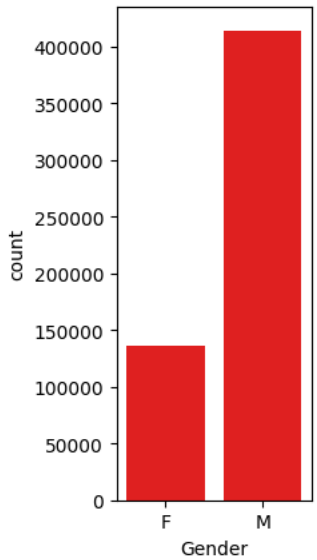
```
df.keys()
```

```
Index(['User_ID', 'Product_ID', 'Gender', 'Age', 'Occupation', 'City_Category',
       'Stay_In_Current_City_Years', 'Marital_Status', 'Product_Category',
       'Purchase'],
      dtype='object')
```

```
fig = px.pie(df, names='Gender', title='Distribution of Gender',color_discrete_sequence=px.colors.sequential.Agsunset)
fig.update_layout(width=650, height=500)
fig.show()
```




```
plt.subplot(1,3,2)
ax= sns.countplot(df, x= 'Gender',color="Red")
plt.show()
```



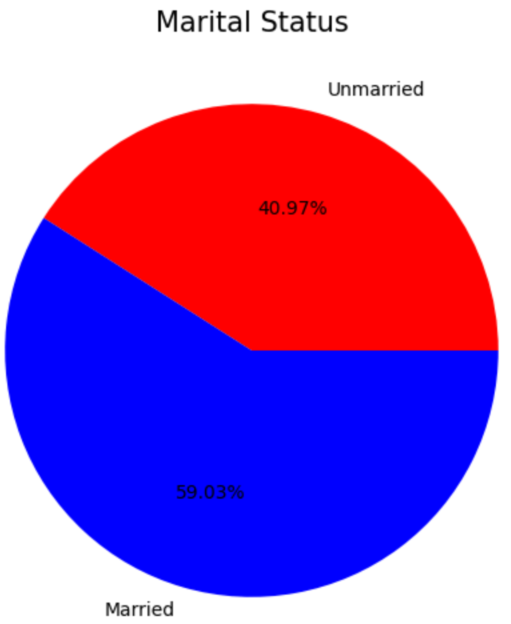

```
df['Marital_Status'].replace(to_replace= 0, value='Unmarried', inplace= True)
df['Marital_Status'].replace(to_replace= 1, value='Married', inplace= True)
```

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

Marital_Status	count
Unmarried	324731
Married	225337

dtype: int64

```
plt.figure(figsize=(14,6))
labels= ['Unmarried', 'Married']
plt.title('Marital Status', fontsize=15)
plt.pie(df.groupby(["Marital_Status"])['Marital_Status'].count(),labels=labels, autopct='%2.2f%%',colors=['Red','Blue'])
plt.show()
```



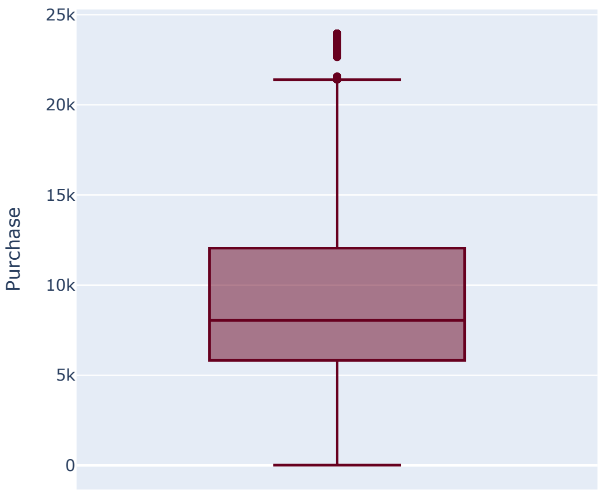

```
round(df["Purchase"].describe(),2)
```

	Purchase
count	550068.00
mean	9263.97
std	5023.07
min	12.00
25%	5823.00
50%	8047.00
75%	12054.00
max	23961.00

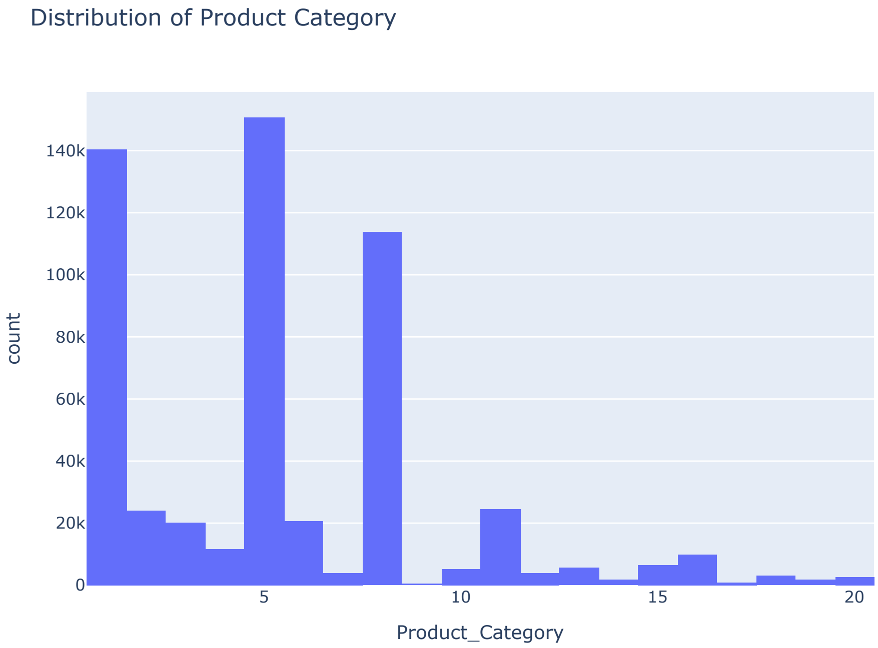
dtype: float64

```
fig = px.histogram(df, x="Purchase", nbins=25, title="Distribution of Purchases",color_discrete_sequence=px.colors.sequential.Agsunset)
fig.update_layout(width=750, height=600)
fig.show()
```


```
px.box(df,y="Purchase",labels=labels,width=550,height=500,color_discrete_sequence=px.colors.sequential.RdBu)
```




```
fig = px.histogram(df, x="Product_Category", nbins=25, title="Distribution of Product Category")
fig.update_layout(width=750, height=550)
fig.show()
```



Central Limit Theorem

```
male_df =df[df["Gender"]=="M"]
male_df
```

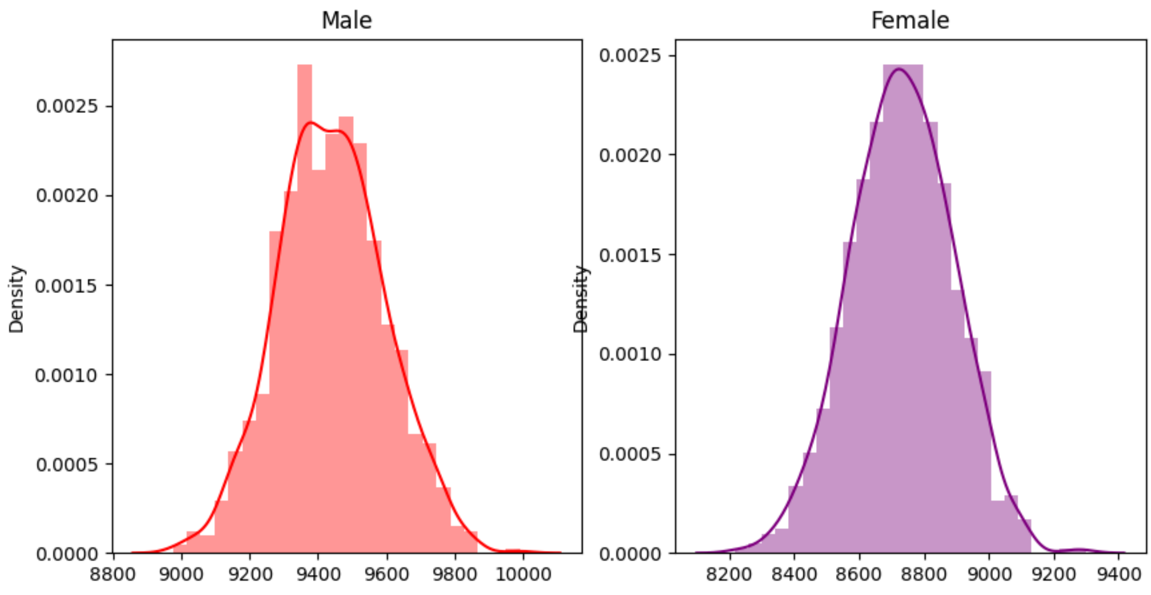
	User_ID	Product_ID	Gender	Age	Occupation	City_Category	Stay_In_Current_City_Years	Marital_Status	Product_Category	P
4	1000002	P00285442	M	55+	16	C	4+	Unmarried	8	
5	1000003	P00193542	M	26-35	15	A	3	Unmarried	1	
6	1000004	P00184942	M	46-50	7	B	2	Married	1	
7	1000004	P00346142	M	46-50	7	B	2	Married	1	
8	1000004	P0097242	M	46-50	7	B	2	Married	1	
...	...	...	...	...	...	...	...	...	...	
550057	1006023	P00370853	M	26-35	0	C	2	Married	19	
550058	1006024	P00372445	M	26-35	12	A	0	Married	20	

```
female_df =df[df["Gender"]=="F"]
female_df
```

	User_ID	Product_ID	Gender	Age	Occupation	City_Category	Stay_In_Current_City_Years	Marital_Status	Product_Category	P
0	1000001	P00069042	F	0-17	10	A	2	Unmarried	3	
1	1000001	P00248942	F	0-17	10	A	2	Unmarried	1	
2	1000001	P00087842	F	0-17	10	A	2	Unmarried	12	
3	1000001	P00085442	F	0-17	10	A	2	Unmarried	12	
14	1000006	P00231342	F	51-55	9	A	1	Unmarried	5	
...	...	...	...	...	...	...	...	...	...	
550061	1006029	P00372445	F	26-35	1	C	1	Married	20	
550064	1006035	P00375436	F	26-35	1	C	3	Unmarried	20	

```
male_means=[]
female_means=[]
for i in range(1000):
 male_mean=male_df['Purchase'].sample(1000).mean()
 female_mean=female_df['Purchase'].sample(1000).mean()
 male_means.append(male_mean)
 female_means.append(female_mean)
```

```
plt.figure(figsize=(10,5))
plt.subplot(1,2,1)
sns.distplot(male_means,color="red")
plt.title('Male')
plt.subplot(1,2,2)
sns.distplot(female_means,color ="Purple")
plt.title('Female')
plt.show()
```



Taking the values for z at 90%, 95% and 99% confidence interval as:

```
Z_value_90_perc=norm.ppf(0.90)
Z_value_90_perc
```

```
1.2815515655446004
```

```
unmarried_data=df[df['Marital_Status']==0]
unmarried_data
```

User_ID	Product_ID	Gender	Age	Occupation	City_Category	Stay_In_Current_City_Years	Marital_Status	Product_Category	Purchas

Calculating 90% Confidence Interval for 1000 Samples:

```
print('Avg spend of male population:',np.mean(male_df['Purchase']))
print('Avg spend of female population:',np.mean(female_df['Purchase']))
print('Standard deviation of male population:',np.std(male_df['Purchase']))
print('Standard deviation of female population:',np.std(female_df['Purchase']))
male_Standard_of_error=np.std(male_df['Purchase'])/np.sqrt(1000)
print('Standard of error:',male_Standard_of_error)
female_Standard_of_error=np.std(female_df['Purchase'])/np.sqrt(1000)
print('Standard of error:',female_Standard_of_error)
Upper_Limit_male=Z_value_90_perc*male_Standard_of_error+np.mean(male_means)
Lower_Limit_male=np.mean(male_means)-Z_value_90_perc*male_Standard_of_error
Upper_limit_female=Z_value_90_perc*female_Standard_of_error+np.mean(female_means)
Lower_limit_female=np.mean(female_means)-Z_value_90_perc*female_Standard_of_error
print('90% confidence interval for male population:',Lower_Limit_male,'to',Upper_Limit_male)
print('90% confidence interval for female population:',Lower_limit_female,'to',Upper_limit_female)
```

```
Avg spend of male population: 9437.526040472265
Avg spend of female population: 8734.565765155476
Standard deviation of male population: 5092.180063635943
Standard deviation of female population: 4767.215738016988
Standard of error: 161.0288725679074
Standard of error: 150.75259829534235
90% confidence interval for male population: 9233.216789262717 to 9645.950396737286
90% confidence interval for female population: 8539.10437664469 to 8925.498833355312
```

Calculating 95% Confidence Interval for Sample Size 1000:

```
Z_value_95_perc=norm.ppf(0.95)
Z_value_95_perc
```

```
1.6448536269514722
```

```
print('Avg spend of male population:',np.mean(male_df['Purchase']))
print('Avg spend of female population:',np.mean(female_df['Purchase']))
print('Standard deviation of male population:',np.std(male_df['Purchase']))
print('Standard deviation of female population:',np.std(female_df['Purchase']))
male_Standard_of_error=np.std(male_df['Purchase'])/np.sqrt(1000)
print('Standard of error:',male_Standard_of_error)
female_Standard_of_error=np.std(female_df['Purchase'])/np.sqrt(1000)
print('Standard of error:',female_Standard_of_error)
Upper_Limit_male=Z_value_95_perc*male_Standard_of_error+np.mean(male_means)
Lower_Limit_male=np.mean(male_means)-Z_value_95_perc*male_Standard_of_error
Upper_limit_female=Z_value_95_perc*female_Standard_of_error+np.mean(female_means)
Lower_limit_female=np.mean(female_means)-Z_value_95_perc*female_Standard_of_error
print('95% confidence interval for male population:',Lower_Limit_male,'to',Upper_Limit_male)
print('95% confidence interval for female population:',Lower_limit_female,'to',Upper_limit_female)
```

```
Avg spend of male population: 9437.526040472265
Avg spend of female population: 8734.565765155476
Standard deviation of male population: 5092.180063635943
Standard deviation of female population: 4767.215738016988
Standard of error: 161.0288725679074
Standard of error: 150.75259829534235
95% confidence interval for male population: 9174.714667912773 to 9704.45251808723
95% confidence interval for female population: 8484.335646921549 to 8980.267563078452
```

Calculating 99% Confidence Interval for Sample size 1000:

```
Z_value_99_perc=norm.ppf(0.99)
Z_value_99_perc
```

```
2.3263478740408408
```

```
print('Avg spend of male population:',np.mean(male_df['Purchase']))
print('Avg spend of female population:',np.mean(female_df['Purchase']))
print('Standard deviation of male population:',np.std(male_df['Purchase']))
print('Standard deviation of female population:',np.std(female_df['Purchase']))
male_Standard_of_error=np.std(male_df['Purchase'])/np.sqrt(1000)
print('Standard of error:',male_Standard_of_error)
female_Standard_of_error=np.std(female_df['Purchase'])/np.sqrt(1000)
print('Standard of error:',female_Standard_of_error)
Upper_Limit_male=Z_value_99_perc*male_Standard_of_error+np.mean(male_means)
Lower_Limit_male=np.mean(male_means)-Z_value_99_perc*male_Standard_of_error
Upper_limit_female=Z_value_99_perc*female_Standard_of_error+np.mean(female_means)
Lower_limit_female=np.mean(female_means)-Z_value_99_perc*female_Standard_of_error
```

```
print('99% confidence interval for male population:',Lower_Limit_male,'to',Upper_Limit_male)
print('99% confidence interval for female population:',Lower_limit_female,'to',Upper_limit_female)
```

```
Avg spend of male population: 9437.526040472265
Avg spend of female population: 8734.565765155476
Standard deviation of male population: 5092.180063635943
Standard deviation of female population: 4767.215738016988
Standard of error: 161.0288725679074
Standard of error: 150.75259829534235
99% confidence interval for male population: 9064.974417642456 to 9814.192768357547
99% confidence interval for female population: 8381.598618449498 to 9083.004591550503
```

Insights

Now using the Confidence interval at 99%, we can say that

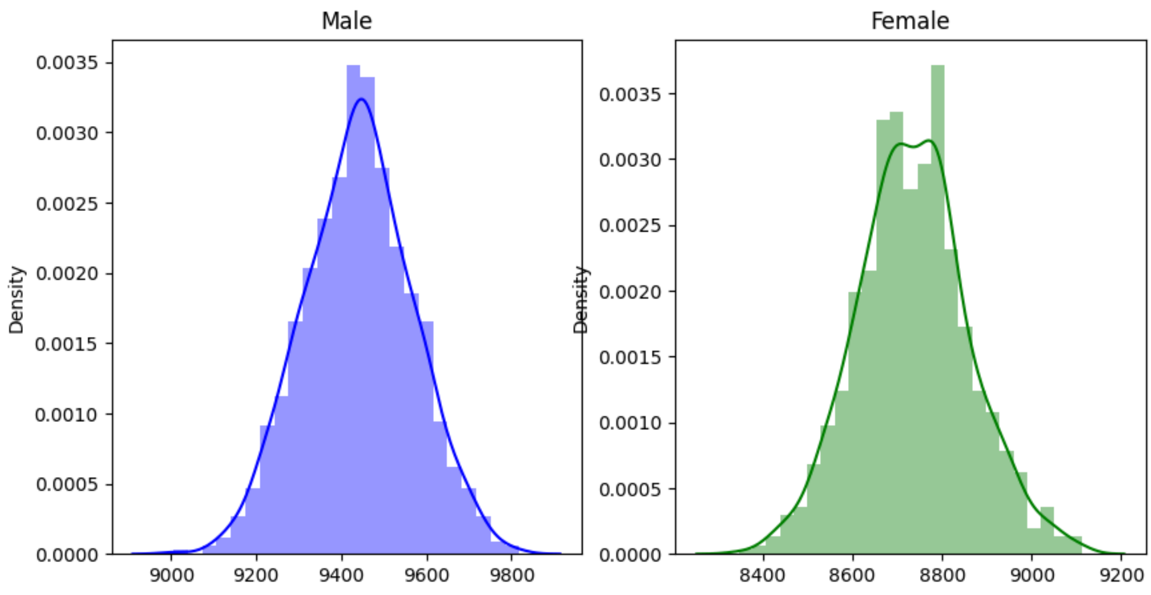
Average amount spend by male customers lie in the range 9058.518713642456 - 9807.737064357547

Average amount spend by female customers lie in range 8385.639530449498 - 9087.045503550504

Calculating 90% Confidence Interval for Sample Size 1500:

```
male_means_1500=[]
female_means_1500=[]
for i in range(1000):
 male_mean_1500=male_df['Purchase'].sample(1500).mean()
 female_mean_1500=female_df['Purchase'].sample(1500).mean()
 male_means_1500.append(male_mean_1500)
 female_means_1500.append(female_mean_1500)
```

```
plt.figure(figsize=(10,5))
plt.subplot(1,2,1)
sns.distplot(male_means_1500,color ="Blue")
plt.title('Male')
plt.subplot(1,2,2)
sns.distplot(female_means_1500,color = "Green")
plt.title('Female')
plt.show()
```




```
Z_value_90_perc=norm.ppf(0.90)
Z_value_90_perc
```

```
1.2815515655446004
```

```
print('Avg spend of male population:',np.mean(male_df['Purchase']))
print('Avg spend of female population:',np.mean(female_df['Purchase']))
print('Standard deviation of male population:',np.std(male_df['Purchase']))
print('Standard deviation of female population:',np.std(female_df['Purchase']))
male_Standard_of_error=np.std(male_df['Purchase'])/np.sqrt(1500)
print('Standard of error:',male_Standard_of_error)
female_Standard_of_error=np.std(female_df['Purchase'])/np.sqrt(1500)
```

```
print('Standard of error:',female_Standard_of_error)
Upper_Limit_male=Z_value_90_perc*male_Standard_of_error+np.mean(male_means)
Lower_Limit_male=np.mean(male_means)-Z_value_90_perc*male_Standard_of_error
Upper_limit_female=Z_value_90_perc*female_Standard_of_error+np.mean(female_means)
Lower_limit_female=np.mean(female_means)-Z_value_90_perc*female_Standard_of_error
print('90% confidence interval for male population:',Lower_Limit_male,'to',Upper_Limit_male)
print('90% confidence interval for female population:',Lower_limit_female,'to',Upper_limit_female)
```

```
Avg spend of male population: 9437.526040472265
Avg spend of female population: 8734.565765155476
Standard deviation of male population: 5092.180063635943
Standard deviation of female population: 4767.215738016988
Standard of error: 131.47952388234287
Standard of error: 123.08898107411797
90% confidence interval for male population: 9271.085803331525 to 9608.081382668477
90% confidence interval for female population: 8574.556728603175 to 8890.046481396826
```

Calculating 95% confidence interval for sample size 1500:

```
Z_value_95_perc=norm.ppf(0.95)
Z_value_95_perc
```

```
1.6448536269514722
```

Calculating 99% confidence interval for sample size 1500:

```
Z_value_99_perc=norm.ppf(0.99)
Z_value_99_perc
```

```
2.3263478740408408
```

```
avgamt_age = df.groupby(['User_ID', 'Age'])[['Purchase']].sum()
avgamt_age = avgamt_age.reset_index()
avgamt_age['Age'].value_counts()
```

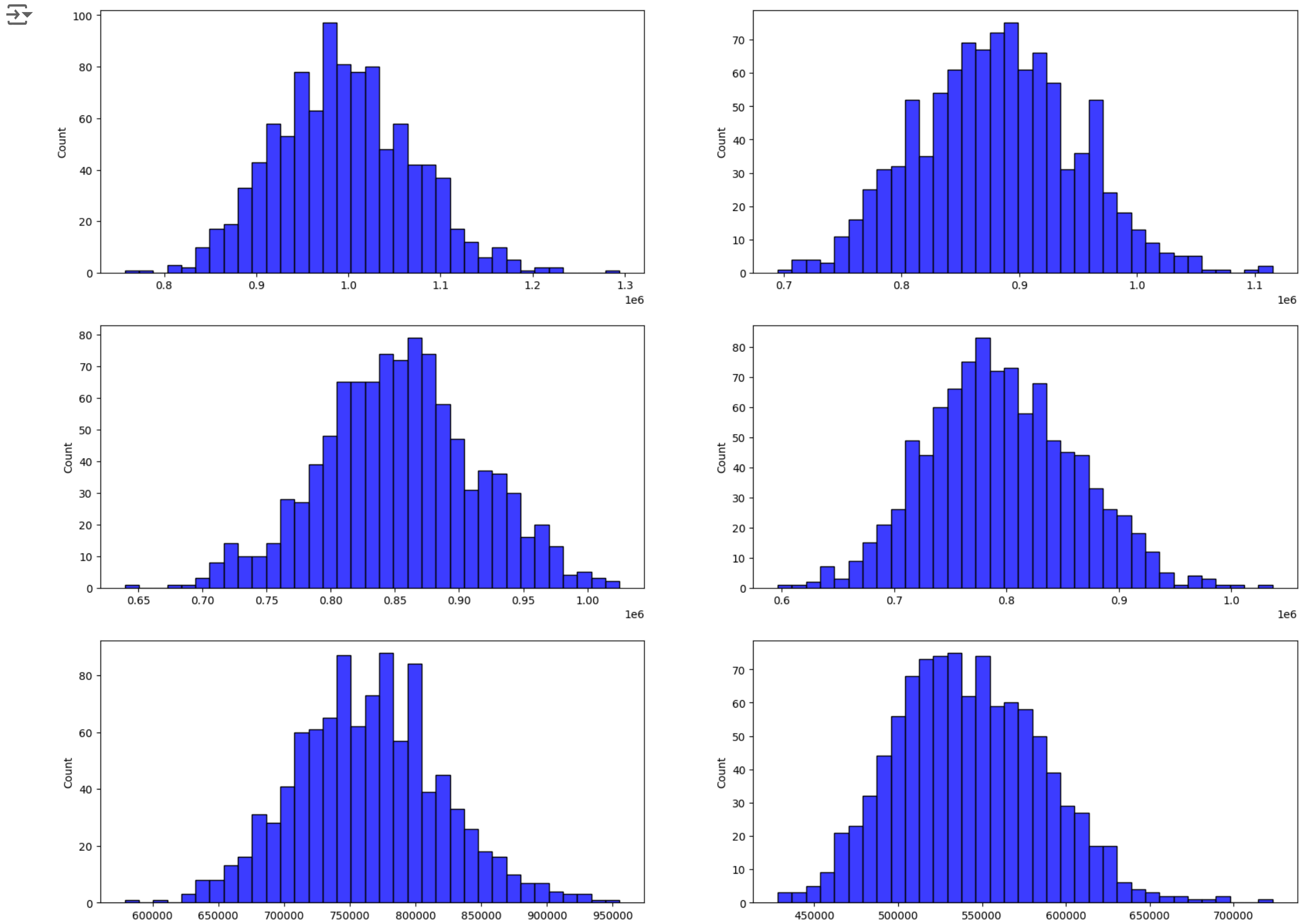
Age	count
26-35	2053
36-45	1167
18-25	1069
46-50	531
51-55	481
55+	372
0-17	218

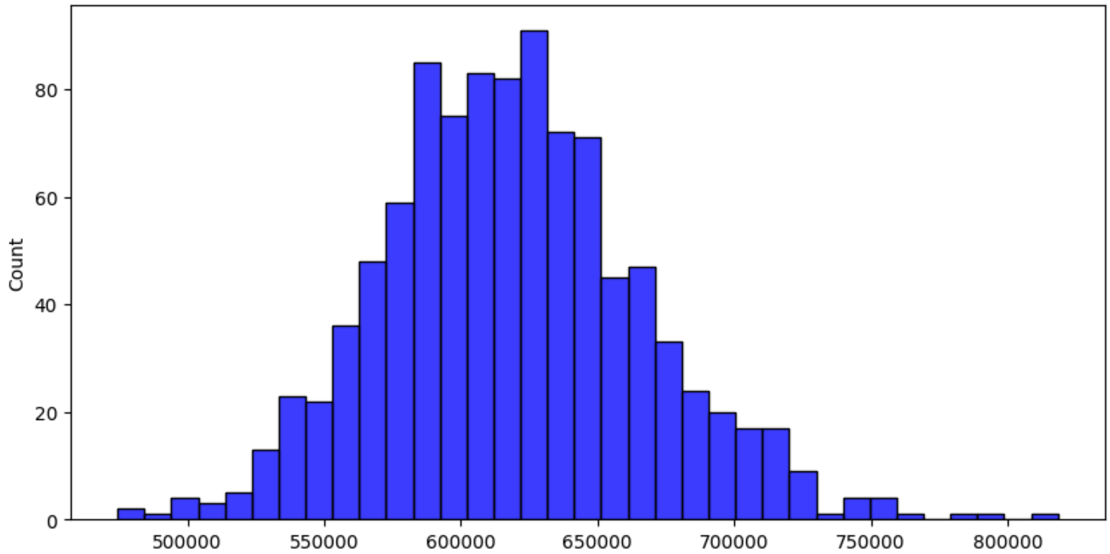
dtype: int64

```
sample_size = 200
num_repitions = 1000
all_sample_means = {}
age_intervals = ['26-35', '36-45', '18-25', '46-50', '51-55', '55+', '0-17']
for i in age_intervals:
 all_sample_means[i] = []


for i in age_intervals:
 for j in range(num_repitions):
  mean = avgamt_age[avgamt_age['Age']==i].sample(sample_size, replace=True)['Purchase'].mean()
  all_sample_means[i].append(mean)


fig, axis = plt.subplots(nrows=3, ncols=2, figsize=(20, 15))
sns.histplot(all_sample_means['26-35'],bins=35,ax=axis[0,0],color= "Blue")
sns.histplot(all_sample_means['36-45'],bins=35,ax=axis[0,1],color="Blue")
sns.histplot(all_sample_means['18-25'],bins=35,ax=axis[1,0],color="Blue")
sns.histplot(all_sample_means['46-50'],bins=35,ax=axis[1,1],color="Blue")
sns.histplot(all_sample_means['51-55'],bins=35,ax=axis[2,0],color="Blue")
sns.histplot(all_sample_means['55+'],bins=35,ax=axis[2,1],color="Blue")
plt.show()
plt.figure(figsize=(10, 5))
sns.histplot(all_sample_means['0-17'],bins=35,color="Blue")
plt.show()
```





Insights

Gender Distribution: 75% of the users are Male, and 25% are Female. Males make more purchases than females. Marital Status:

Marital status: 59% of customers are Single, while 41% are Married.

City Stay Duration: 35% of customers have been staying in the city for 1 year. 18% have been staying for 2 years. 17% have been staying for 3 years.

City Categories: Majority of customers are from City Category B. Customers from City Category C spend the most on average.

Age Distribution: The majority of customers are between the ages of 26 and 35. 60% of purchases are made by people between the ages of 26 and 45.

Recommendations

Gender-Based Targeting : Allocate marketing resources and campaigns based on the gender distribution. Tailor product recommendations and promotions to match the preferences of each gender group.

Marital Status Engagement : Create marketing messages that resonate with both single and married customers. Offer special promotions for couples or families to encourage shopping together

City Stay Duration Engagement : Develop loyalty programs or incentives for customers staying in the city for longer periods. Highlight the benefits of long-term customer relationships to encourage repeat business.

City Categories Strategy : Focus marketing efforts on City Category B, where more customers tend to make purchases. Analyze what attracts customers in City Category C and consider replicating successful strategies in other categories.

Customer Experience Improvement : Focus on enhancing the shopping experience for customers aged 26-35, as they form a significant customer base. Train staff to provide excellent customer service and address the specific needs of this age group.

Start coding or generate with AI.

Start coding or generate with AI.

Start coding or generate with AI.

Start coding or generate with AI.

Start coding or generate with AI.

Start coding or generate with AI.

Start coding or generate with AI.

Start coding or generate with AI.

Start coding or generate with AI.

Start coding or generate with AI.

Start coding or generate with AI.

Start coding or generate with AI.

Start coding or generate with AI.

Start coding or generate with AI.

Start coding or generate with AI.

Start coding or generate with AI.

Start coding or generate with AI.

Start coding or generate with AI.

Start coding or generate with AI.

Start coding or generate with AI.

Start coding or generate with AI.

Start coding or generate with AI.

Start coding or generate with AI.

Start coding or generate with AI.

Start coding or generate with AI.

Start coding or generate with AI.

Start coding or generate with AI.

Start coding or generate with AI.

Start coding or generate with AI.

Start coding or generate with AI.

Start coding or generate with AI.

Start coding or generate with AI.

Start coding or generate with AI.

Start coding or generate with AI.

Start coding or generate with AI.

Start coding or generate with AI.

Start coding or generate with AI.

Start coding or generate with AI.

Start coding or generate with AI.

Start coding or generate with AI.

Start coding or generate with AI.

Start coding or generate with AI.