```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
```

```
data= pd.read_csv("https://d2beiqkhq929f0.cloudfront.net/public_assets/assets/000/001/293/original/walmart_data.csv?1641285094")
data.head()
```

	User_ID	Product_ID	Gender	Age	Occupation	City_Category	Stay_In_Current_City_Yea
0	1000001	P00069042	F	0-17	10	A	
1	1000001	P00248942	F	0-17	10	A	
2	1000001	P00087842	F	0-17	10	A	

```
data.shape
```

```
(550068, 10)
```

```
data.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
```

```
data[['User_ID','Occupation', 'Marital_Status', 'Product_Category']]= data[['User_ID','Occupation', 'Marital_Status', 'Product_Category']].as
```

```
data.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  object
 1   Product_ID                  550068 non-null  object
 2   Gender                      550068 non-null  object
 3   Age                         550068 non-null  object
 4   Occupation                  550068 non-null  object
 5   City_Category               550068 non-null  object
 6   Stay_In_Current_City_Years  550068 non-null  object
 7   Marital_Status              550068 non-null  object
 8   Product_Category            550068 non-null  object
 9   Purchase                    550068 non-null  int64
dtypes: int64(1), object(9)
memory usage: 42.0+ MB
```

```
data.describe(include= "all")
```

	User_ID	Product_ID	Gender	Age	Occupation	City_Category	Stay_In_Curren
count	550068.0	550068	550068	550068	550068.0	550068	
unique	5891.0	3631	2	7	21.0	3	
top	1001680.0	P00265242	M	26-35	4.0	B	
freq	1026.0	1880	414259	219587	72308.0	231173	
mean	NaN	NaN	NaN	NaN	NaN	NaN	
std	NaN	NaN	NaN	NaN	NaN	NaN	
min	NaN	NaN	NaN	NaN	NaN	NaN	

```
data.User_ID.value_counts()
```

```
1001680    1026
1004277     979
1001941     898
1001181     862
1000889     823
           ...
1002690       7
1002111       7
1005810       7
1004991       7
1000708       6
Name: User_ID, Length: 5891, dtype: int64
```

```
data.Product_ID.value_counts()
```

```
P00265242    1880
P00025442    1615
P00110742    1612
P00112142    1562
P00057642    1470
             ...
P00314842       1
P00298842       1
P00231642       1
P00204442       1
P00066342       1
Name: Product_ID, Length: 3631, dtype: int64
```

```
data.Gender.value_counts(normalize= True)
```

```
M    0.753105
F    0.246895
Name: Gender, dtype: float64
```

```
data.Marital_Status.value_counts(normalize= True)
```

```
0    0.590347
1    0.409653
Name: Marital_Status, dtype: float64
```

```
data.Age.value_counts(normalize= True).sort_values(ascending= False)
```

```
26-35    0.399200
36-45    0.199999
18-25    0.181178
46-50    0.083082
51-55    0.069993
55+      0.039093
0-17     0.027455
Name: Age, dtype: float64
```

```
data.Occupation.value_counts(normalize= True).sort_values(ascending= False)
```

```
4     0.131453
0     0.126599
7     0.107501
1     0.086218
17    0.072796
20    0.061014
12    0.056682
14    0.049647
2     0.048336
16    0.046123
```

```
6     0.037005
3     0.032087
10    0.023506
5     0.022137
15    0.022115
11    0.021063
19    0.015382
13    0.014049
18    0.012039
9     0.011437
8     0.002811
Name: Occupation, dtype: float64
```

```
data.City_Category.value_counts(normalize= True).sort_values(ascending= False)
```

```
B    0.420263
C    0.311189
A    0.268549
Name: City_Category, dtype: float64
```

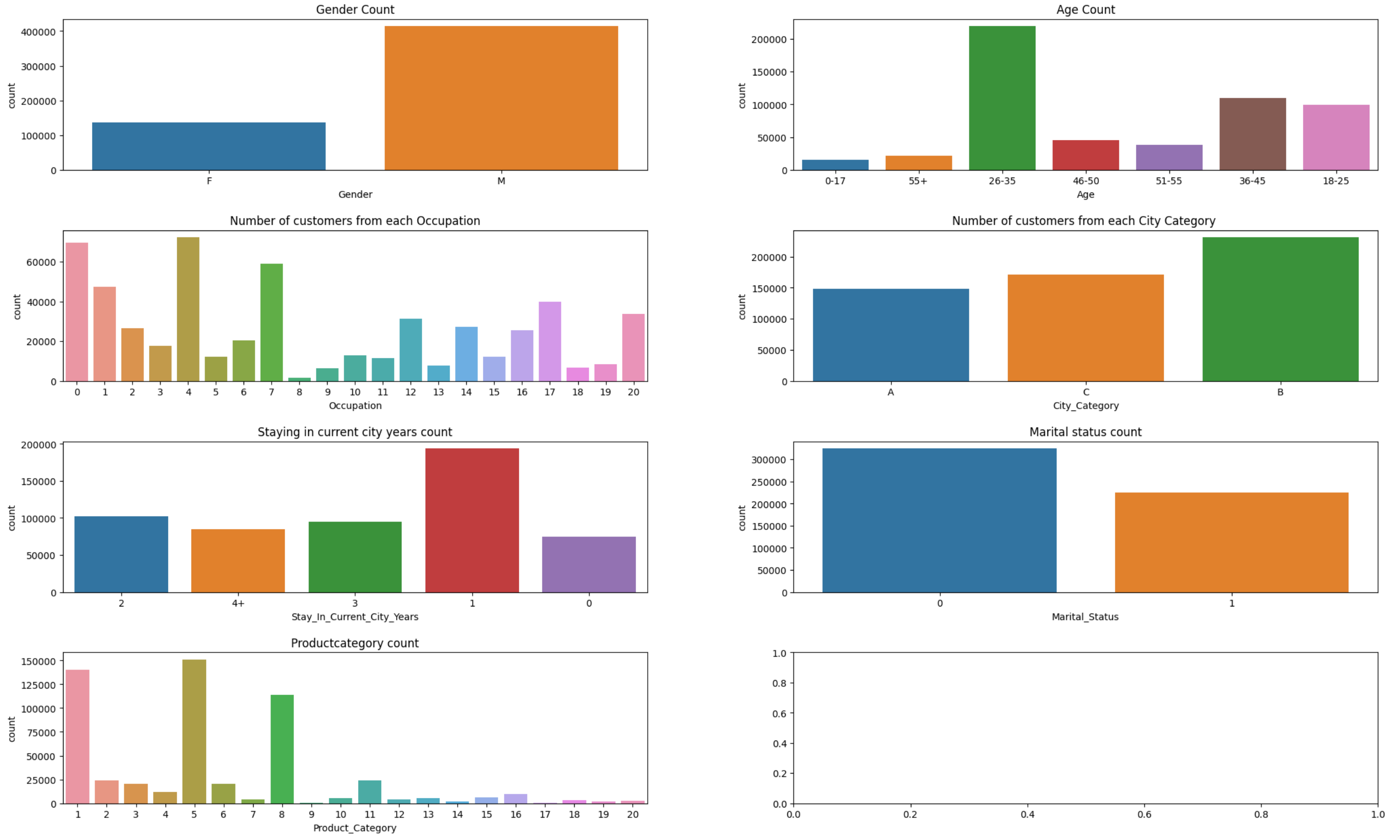
```
data.Stay_In_Current_City_Years.value_counts(normalize= True).sort_values(ascending= False)
```

```
1     0.352358
2     0.185137
3     0.173224
4+    0.154028
0     0.135252
Name: Stay_In_Current_City_Years, dtype: float64
```

```
data.Product_Category.value_counts(normalize= True).sort_values(ascending= False)
```

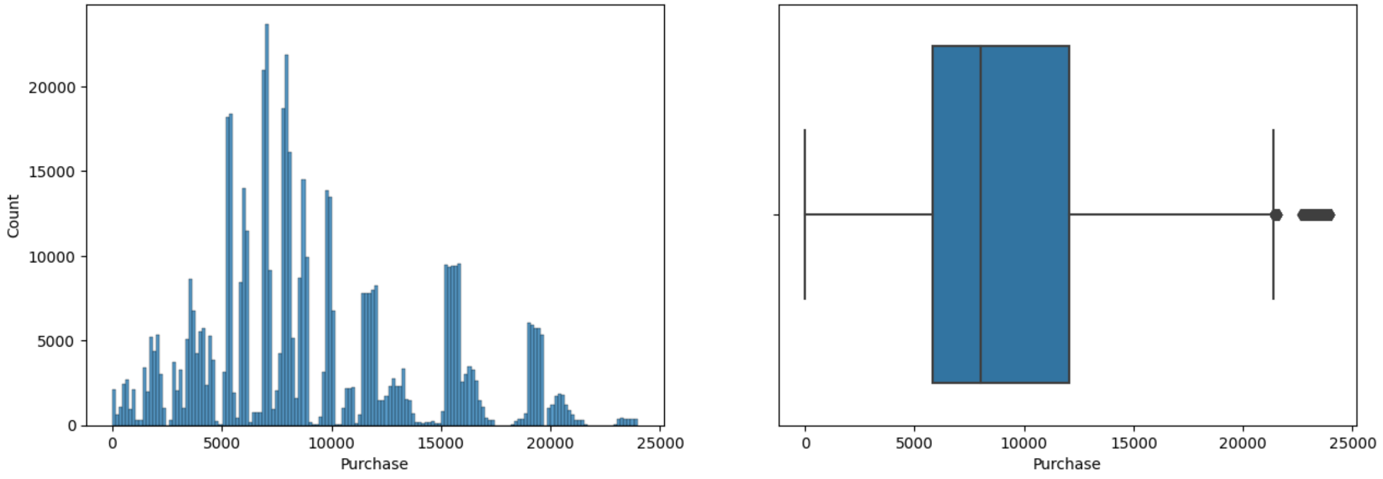
```
5     0.274390
1     0.255201
8     0.207111
11    0.044153
2     0.043384
6     0.037206
3     0.036746
4     0.021366
16    0.017867
15    0.011435
13    0.010088
10    0.009317
12    0.007175
7     0.006765
18    0.005681
20    0.004636
19    0.002914
14    0.002769
17    0.001051
9     0.000745
Name: Product_Category, dtype: float64
```

```
fig, ax = plt.subplots(4, 2, figsize=(25,15))
plt.subplots_adjust(wspace=0.25, hspace=0.4)

sns.countplot(x="Gender", data= data, ax= ax[0,0])
sns.countplot(x="Age", data= data, ax= ax[0,1])
sns.countplot(x="Occupation", data= data, ax= ax[1,0])
sns.countplot(x="City_Category", data= data, ax= ax[1,1])
sns.countplot(x="Stay_In_Current_City_Years", data= data, ax= ax[2,0])
sns.countplot(x="Marital_Status", data= data, ax= ax[2,1])
sns.countplot(x="Product_Category", data= data, ax= ax[3,0])

ax[0,0].set_title("Gender Count")
ax[0,1].set_title("Age Count")
ax[1,0].set_title("Number of customers from each Occupation")
ax[1,1].set_title("Number of customers from each City Category ")
ax[2,0].set_title("Staying in current city years count")
ax[2,1].set_title("Marital status count")
ax[3,0].set_title("Productcategory count")

plt.show()
```



```
fig, ax = plt.subplots(1, 2, figsize=(15, 5))
sns.histplot(x="Purchase", data= data, ax=ax[0])
sns.boxplot(x="Purchase", data= data, ax=ax[1], showfliers= True)
```

```
<Axes: xlabel='Purchase'>
```



```
data.groupby(by=["Gender"])["Purchase"].mean()
```

```
Gender
F    8734.565765
```

```
     M    9437.526040
     Name: Purchase, dtype: float64
```

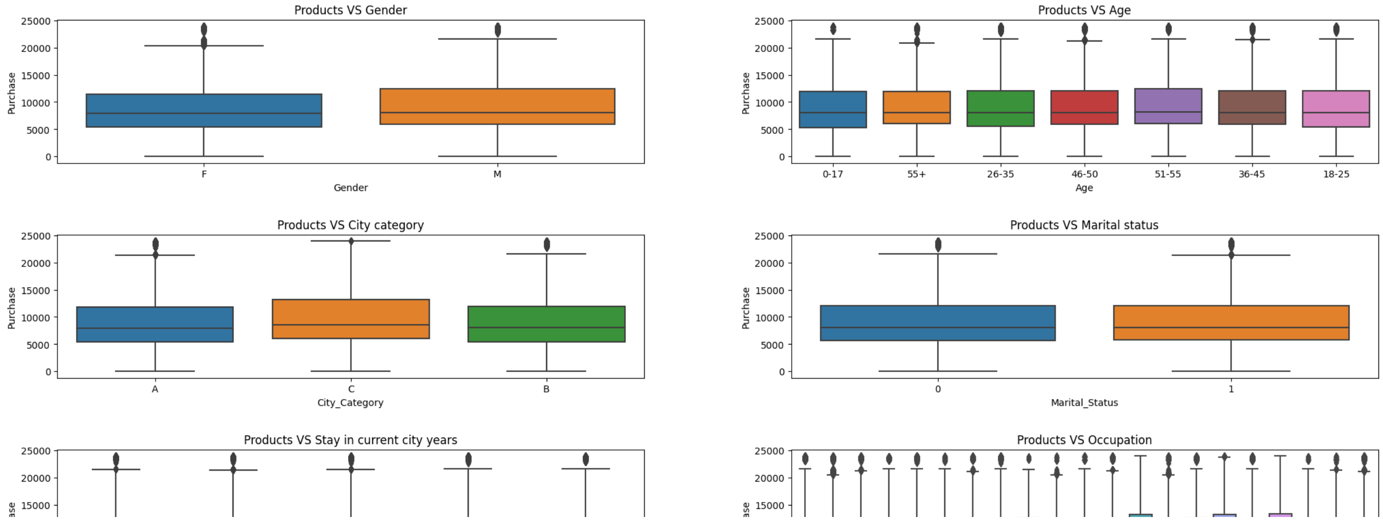
```
data.groupby(by=["Age","Gender"])["Purchase"].sum()
```

```
     Age    Gender
     0-17   F           42385978
            M           92527205
     18-25  F          205475842
            M          708372833
     26-35  F          442976233
            M         1588794345
     36-45  F          243438963
            M          783130921
     46-50  F          116706864
            M          304136539
     51-55  F           89465997
            M          277633647
     55+    F           45782765
            M          154984610
     Name: Purchase, dtype: int64
```

```
data.groupby(by=["Marital_Status","Gender"])["Purchase"].mean()
```

```
     Marital_Status  Gender
     0               F         8679.845815
                     M         9453.756740
     1               F         8810.249789
                     M         9413.817605
     Name: Purchase, dtype: float64
```

```
fig, ax = plt.subplots(4, 2, figsize=(25, 15))
plt.subplots_adjust(wspace=0.25, hspace=0.5)

sns.boxplot(x="Gender", y="Purchase", data=data, ax=ax[0,0])
sns.boxplot(x="Age", y="Purchase", data=data, ax=ax[0,1])
sns.boxplot(x="City_Category", y="Purchase", data=data, ax=ax[1,0])
sns.boxplot(x="Marital_Status", y="Purchase", data=data, ax=ax[1,1])
sns.boxplot(x="Stay_In_Current_City_Years", y="Purchase", data=data, ax=ax[2,0])
sns.boxplot(x="Occupation", y="Purchase", data=data, ax=ax[2,1])
sns.boxplot(x="Product_Category", y="Purchase", data=data, ax=ax[3,0])

ax[0,0].set_title("Products VS Gender")
ax[0,1].set_title("Products VS Age")
ax[1,0].set_title("Products VS City category")
ax[1,1].set_title("Products VS Marital status")
ax[2,0].set_title("Products VS Stay in current city years")
ax[2,1].set_title("Products VS Occupation")
ax[3,0].set_title("Products VS Productcategory")

plt.show()
```



```
from IPython.display import display

cols= ["Gender", "Age", "City_Category", "Marital_Status", "Stay_In_Current_City_Years", "Occupation", "Product_Category"]

for i in cols:
  display(data.groupby(by=i)["Purchase"].describe())
```

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

	count	mean	std	min	25%	50%	75%	max
Age								
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

	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

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

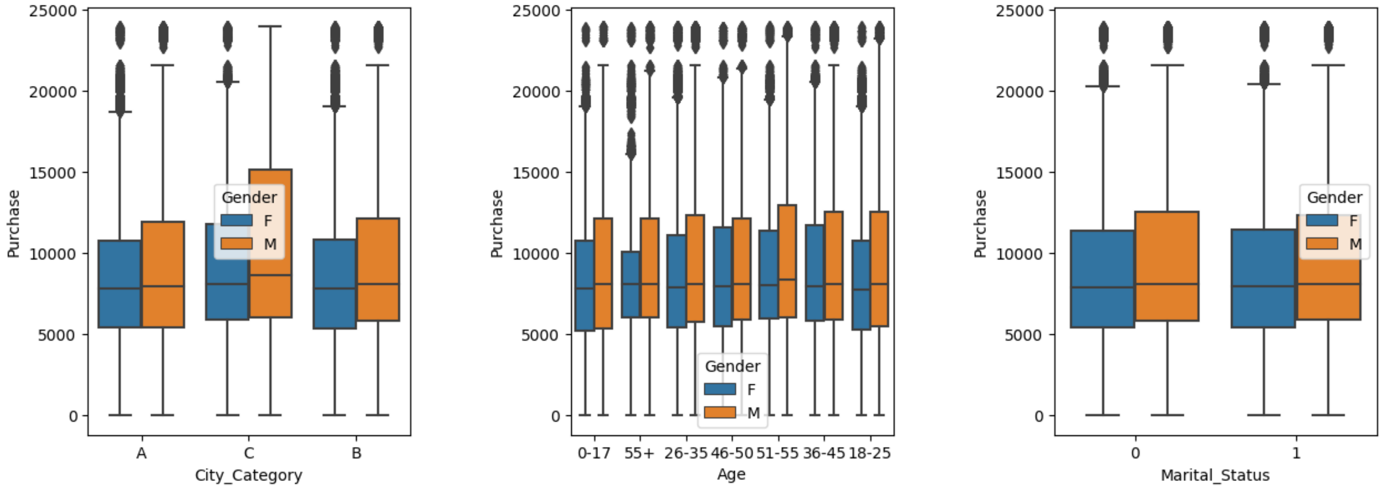
	count	mean	std	min	25%	50%	75%	max
Stay_In_Current_City_Years								
0	74398.0	9180.075123	4990.479940	12.0	5480.0	8025.0	11990.0	23960.0
1	193821.0	9250.145923	5027.476933	12.0	5500.0	8041.0	12042.0	23961.0
2	101838.0	9320.429810	5044.588224	12.0	5846.0	8072.0	12117.0	23961.0
3	95285.0	9286.904119	5020.343541	12.0	5832.0	8047.0	12075.0	23961.0
4+	84726.0	9275.598872	5017.627594	12.0	5844.0	8052.0	12038.0	23958.0

	count	mean	std	min	25%	50%	75%	max
Occupation								
0	69638.0	9124.428588	4971.757402	12.0	5445.00	8001.0	11957.00	23961.0

```
fig, ax = plt.subplots(1, 3, figsize=(15, 5))
plt.subplots_adjust(wspace=0.5, hspace=0.5)
```

```
sns.boxplot(x="Age", y="Purchase", hue= "Gender", data= data, ax=ax[1])
sns.boxplot(x="Marital_Status", y="Purchase", hue= "Gender", data= data, ax=ax[2])
sns.boxplot(x="City_Category", y="Purchase", hue= "Gender", data= data, ax=ax[0])
```

```
<Axes: xlabel='City_Category', ylabel='Purchase'>
```



```
from IPython.display import display

cols2= ["City_Category", "Age", "Marital_Status"]

for i in cols2:
  display(data.groupby([i, "Gender"])["Purchase"].describe())
```

		count	mean	std	min	25%	50%	75%	max
City_Category	Gender								
A	F	35704.0	8579.708576	4670.230320	12.0	5413.0	7847.0	10728.25	23948.0
	M	112016.0	9017.834470	4956.095263	12.0	5399.0	7963.0	11908.00	23961.0
B	F	57796.0	8540.677694	4682.803540	12.0	5376.0	7839.0	10847.00	23959.0
	M	173377.0	9354.854433	5026.679086	12.0	5826.0	8065.0	12134.00	23960.0
C	F	42309.0	9130.107518	4935.788374	12.0	5919.0	8077.0	11765.00	23951.0
	M	128866.0	9913.567248	5255.694667	12.0	6071.0	8655.0	15161.00	23961.0

```
# Average amount by gender

df1= data.groupby(by=["User_ID", "Gender"])["Purchase"].sum().reset_index()
df1.groupby(by="Gender")["Purchase"].mean()
```

```
Gender
F    712024.394958
M    925344.402367
Name: Purchase, dtype: float64
```

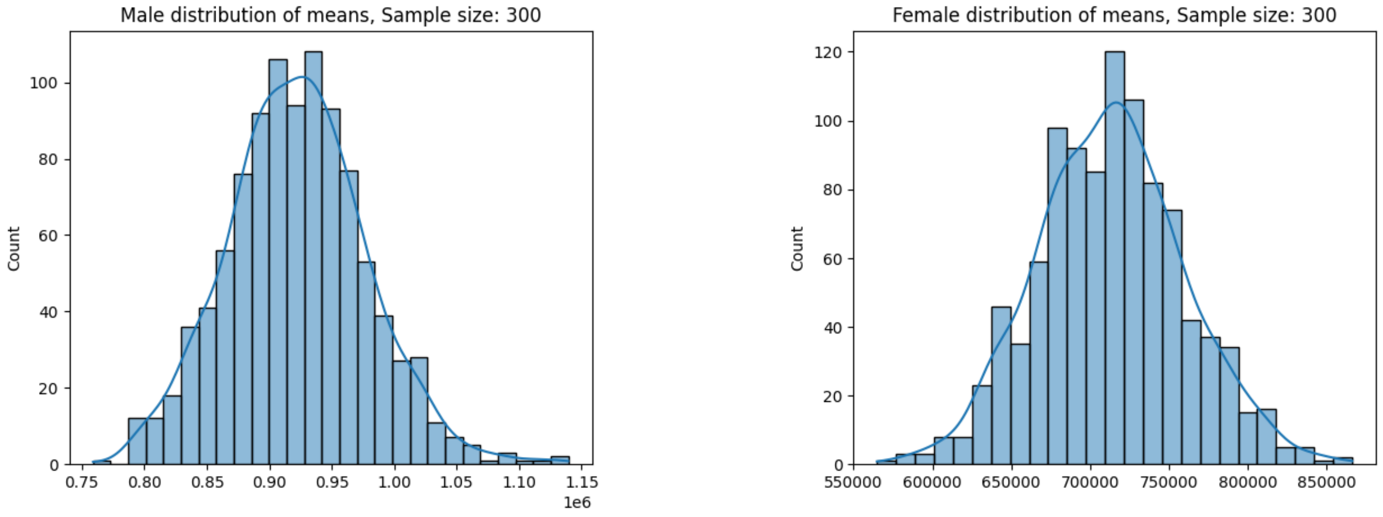
26-35 F 50752.0 8728.251754 4718.826059 12.0 5442.0 7886.0 11101.25 23955.0

```
# Confidence intervals and distribution of the mean of the expenses by female and male customers for sample size of 300

male_data= df1[df1.Gender=="M"]
female_data= df1[df1.Gender=="F"]

male_avg=[male_data.sample(300, replace=True)["Purchase"].mean() for i in range(1000)]
female_avg=[female_data.sample(300, replace=True)["Purchase"].mean() for i in range(1000)]
```

M 32502.0 9357.471509 5027.390264 12.0 5921.0 8074.5 12110.00 23960.0

```
fig, ax = plt.subplots(1, 2, figsize=(15, 5))
plt.subplots_adjust(wspace=0.5, hspace=0.5)

sns.histplot(data=male_avg, ax=ax[0], kde= True)
sns.histplot(data=female_avg, ax=ax[1], kde= True)

ax[0].set_title("Male distribution of means, Sample size: 300")
ax[1].set_title("Female distribution of means, Sample size: 300")
```

```
Text(0.5, 1.0, 'Female distribution of means, Sample size: 300')
```



```
male_pop_mean= np.round(np.mean(male_data["Purchase"]),2)
female_pop_mean= np.round(np.mean(female_data["Purchase"]),2)

male_samp_mean= np.round(np.mean(male_avg),2)
female_samp_mean= np.round(np.mean(female_avg),2)

male_std= np.std(male_avg)
```

```
female_std= np.std(female_avg)

print("Male Population average:", male_pop_mean)
print("Male Population average:", female_pop_mean)
print("")
print("Male Sample average:", male_samp_mean)
print("Female Sample average:", female_samp_mean)
print("")
print("Male standard deviation:", male_std)
print("Male standard deviation:", female_std)
```

```
    Male Population average: 925344.4
    Male Population average: 712024.39

    Male Sample average: 922198.49
    Female Sample average: 712928.06

    Male standard deviation: 55215.58693568351
    Male standard deviation: 46369.89686685192
```

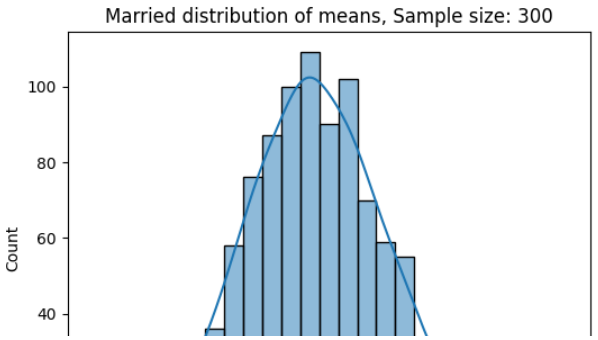
```
# Confidence interval of average expense of male customer

ci={"90%": 1.645, "95%": 1.960, "99%": 2.567}

for i in ci:
  print(f"{i} Confidece interval of male:", (np.round(male_samp_mean - (ci[i]*(male_std/300**0.5)),2), np.round(male_samp_mean + (ci[i]*(male
```

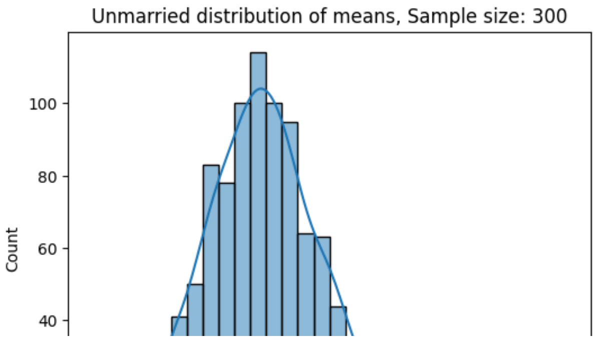
```
    90% Confidece interval of male: (916954.44, 927442.54)
    95% Confidece interval of male: (915950.26, 928446.72)
    99% Confidece interval of male: (914015.22, 930381.76)
```

```
# Confidence interval of avegrage expense of female expense

ci={"90%": 1.645, "95%": 1.960, "99%": 2.567}

for i in ci:
  print(f"{i} Confidece interval of female:", (np.round(female_samp_mean - (ci[i]*(female_std/300**0.5)),2), np.round(female_samp_mean + (ci[
```

```
    90% Confidece interval of female: (708524.12, 717332.0)
    95% Confidece interval of female: (707680.81, 718175.31)
    99% Confidece interval of female: (706055.77, 719800.35)
```

```
# Confidence intervals and distribution of the mean of the expenses by customers based on Marital status

df2= data.groupby(by=["User_ID", "Marital_Status"])["Purchase"].sum().reset_index()
df2.groupby(by="Marital_Status")["Purchase"].mean()

married_data= df2[df2.Marital_Status==1]
unmarried_data= df2[df2.Marital_Status==0]

married_avg=[married_data.sample(300, replace=True)["Purchase"].mean() for i in range(1000)]
unmarried_avg=[unmarried_data.sample(300, replace=True)["Purchase"].mean() for i in range(1000)]


fig, ax = plt.subplots(1, 2, figsize=(15, 5))
plt.subplots_adjust(wspace=0.5, hspace=0.5)

sns.histplot(data=married_avg, ax=ax[0], kde= True)
sns.histplot(data=unmarried_avg, ax=ax[1], kde= True)

ax[0].set_title("Married distribution of means, Sample size: 300")
ax[1].set_title("Unmarried distribution of means, Sample size: 300")

plt.show()
```





```
married_pop_mean= np.round(np.mean(married_data["Purchase"]),2)
unmarried_pop_mean= np.round(np.mean(unmarried_data["Purchase"]),2)

married_samp_mean= np.round(np.mean(married_avg),2)
unmarried_samp_mean= np.round(np.mean(unmarried_avg),2)

married_std= np.round(np.std(married_avg),2)
unmarried_std= np.round(np.std(unmarried_avg),2)

print("Married Population average:", married_pop_mean)
print("Unmarried Population average:", unmarried_pop_mean)
print("")
print("Married Sample average:", married_samp_mean)
print("Unmarried Sample average:", unmarried_samp_mean)
print("")
print("Married standard deviation:", married_std)
print("Unmarried standard deviation:", unmarried_std)
```

```
    Married Population average: 843526.8
    Unmarried Population average: 880575.78

    Married Sample average: 844023.15
    Unmarried Sample average: 876984.19

    Married standard deviation: 52386.17
    Unmarried standard deviation: 56124.66
```

```
# confidence interval of average expense of married customer

ci={"90%": 1.645, "95%": 1.960, "99%": 2.567}

for i in ci:
  print(f"{i} Confidence interval of average expence of married customers:", (np.round(married_samp_mean - (ci[i]*(married_std/300**0.5)),2),
```

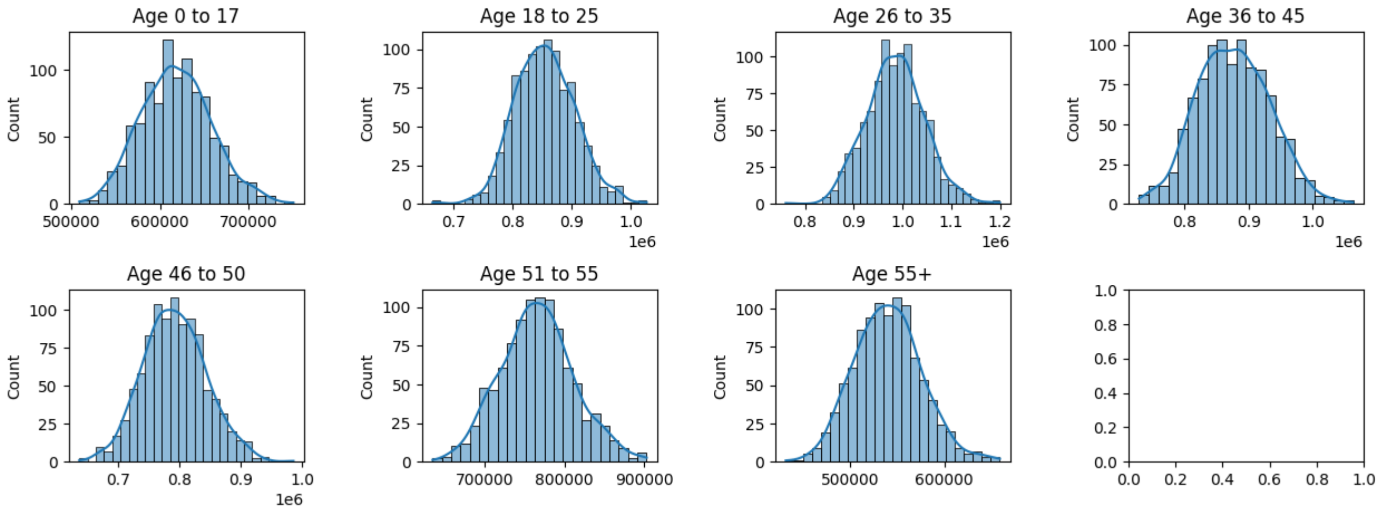
```
    90% Confidence interval of average expence of married customers: (839047.82, 848998.48)
    95% Confidence interval of average expence of married customers: (838095.1, 849951.2)
    99% Confidence interval of average expence of married customers: (836259.22, 851787.08)
```

```
# confidence interval of average expense of unmarried customer

ci={"90%": 1.645, "95%": 1.960, "99%": 2.567}

for i in ci:
  print(f"{i} Confidence interval of average expence of unmarried customers:", (np.round(unmarried_samp_mean - (ci[i]*(unmarried_std/300**0.5
```

```
    90% Confidence interval of average expence of unmarried customers: (871653.8, 882314.58)
    95% Confidence interval of average expence of unmarried customers: (870633.09, 883335.29)
    99% Confidence interval of average expence of unmarried customers: (868666.19, 885302.19)
```

```
# Confidence intervals and distribution of the mean of the expenses by customers of diffent age groups

df3= data.groupby(by=["User_ID", "Age"])["Purchase"].sum().reset_index()
df3.groupby(by="Age")["Purchase"].mean()

age_intervals= data.Age.unique()

samp_avg={}
samp_data={}

for i in age_intervals:
  samp_data[i]= df3[df3.Age==i]
```

```
for i in age_intervals:
  samp_avg[i]=[ df3[df3.Age==i].sample(300, replace=True)["Purchase"].mean() for j in range(1000)]


fig, ax = plt.subplots(2, 4, figsize=(15, 5))
plt.subplots_adjust(wspace=0.5, hspace=0.5)

sns.histplot(data=samp_avg['0-17'], ax=ax[0,0], kde= True)
sns.histplot(data=samp_avg['18-25'], ax=ax[0,1], kde= True)
sns.histplot(data=samp_avg['26-35'], ax=ax[0,2], kde= True)
sns.histplot(data=samp_avg['36-45'], ax=ax[0,3], kde= True)
sns.histplot(data=samp_avg['46-50'], ax=ax[1,0], kde= True)
sns.histplot(data=samp_avg['51-55'], ax=ax[1,1], kde= True)
sns.histplot(data=samp_avg['55+'], ax=ax[1,2], kde= True)

ax[0,0].set_title("Age 0 to 17")
ax[0,1].set_title("Age 18 to 25")
ax[0,2].set_title("Age 26 to 35")
ax[0,3].set_title("Age 36 to 45")
ax[1,0].set_title("Age 46 to 50")
ax[1,1].set_title("Age 51 to 55")
ax[1,2].set_title("Age 55+")

plt.show()
```



```
samp_pop_mean={}
samp_mean={}
samp_std={}
ul90={}
ul95={}
ul99={}
ll90={}
ll95={}
ll99={}

for i in age_intervals:

  samp_pop_mean[i]= np.round(np.mean(samp_data[i].Purchase),2)
  samp_mean[i]= np.round(np.mean(samp_avg[i]),2)
  samp_std[i]= np.round(np.std(samp_avg[i]),2)

#90%
for i in age_intervals:
  ll90[i]= np.round(samp_mean[i] - (1.645*(samp_std[i]/300**0.5)),2)
  ul90[i]= np.round(samp_mean[i] + (1.645*(samp_std[i]/300**0.5)),2)
#95%
for i in age_intervals:
  ll95[i]= np.round(samp_mean[i] - (1.960*(samp_std[i]/300**0.5)),2)
  ul95[i]= np.round(samp_mean[i] + (1.960*(samp_std[i]/300**0.5)),2)

#99%
```