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).

Dataset

The company collected the transactional data of customers who purchased products from the Walmart Stores during Black Friday. The dataset has the following features: Dataset link: Walmart_data.csv

User_ID: User ID Product_ID: Product ID Gender: Sex of User Age: Age in bins Occupation: Occupation(Masked) City_Category: Category of the City (A,B,C) StayInCurrentCityYears: Number of years stay in current city Marital_Status: Marital Status ProductCategory: Product Category (Masked) Purchase: Purchase Amount

Importing Libraries and Performing Basic EDA

In [1]:
```
import pandas as pd
import numpy as np
%matplotlib inline
import pandas as pd
from matplotlib import pyplot as plt
import matplotlib as m
import seaborn as sns
m.style.use('ggplot')
import math
import os
import plotly.express as px
from wordcloud import wordcloud
from datetime import datetime
from scipy import stats
from scipy.stats import norm
```

```
import warnings
warnings.filterwarnings('ignore')
df = pd.read_csv("https://d2beiqkhq929f0.cloudfront.net/public_assets/assets/000
print(df.shape)
df.head()
df
```

(550068, 10)

Out[1]:

	User_ID	Product_ID	Gender	Age	Occupation	City_Category	Stay_In_Current
0	1000001	P00069042	F	0-17	10	A	
1	1000001	P00248942	F	0-17	10	A	
2	1000001	P00087842	F	0-17	10	A	
3	1000001	P00085442	F	0-17	10	A	
4	1000002	P00285442	M	55+	16	C	
...	...	...	...	...	...	...	
550063	1006033	P00372445	M	51-55	13	B	
550064	1006035	P00375436	F	26-35	1	C	
550065	1006036	P00375436	F	26-35	15	B	
550066	1006038	P00375436	F	55+	1	C	
550067	1006039	P00371644	F	46-50	0	B	

550068 rows × 10 columns

1.Import the Dataset and do Usual data analysis steps like checking the structure & characteristics of the dataset.

BASIC DATA ANALYSIS

In [ ]:

```
print(df.shape)
print(df.size)
print(df.ndim)
print(df.info())
```

```
print(df.describe())
print(df.columns)
```

```
(550068, 10)
5500680
2
<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
None
            User_ID     Occupation  Marital_Status  Product_Category  \
count  5.500680e+05  550068.000000   550068.000000     550068.000000   
mean   1.003029e+06       8.076707        0.409653          5.404270   
std    1.727592e+03       6.522660        0.491770          3.936211   
min    1.000001e+06       0.000000        0.000000          1.000000   
25%    1.001516e+06       2.000000        0.000000          1.000000   
50%    1.003077e+06       7.000000        0.000000          5.000000   
75%    1.004478e+06      14.000000        1.000000          8.000000   
max    1.006040e+06      20.000000        1.000000         20.000000   

            Purchase  
count  550068.000000  
mean     9263.968713  
std      5023.065394  
min        12.000000  
25%      5823.000000  
50%      8047.000000  
75%     12054.000000  
max     23961.000000  
Index(['User_ID', 'Product_ID', 'Gender', 'Age', 'Occupation', 'City_Category',
       'Stay_In_Current_City_Years', 'Marital_Status', 'Product_Category',
       'Purchase'],
      dtype='object')
```

INSIGHTS:

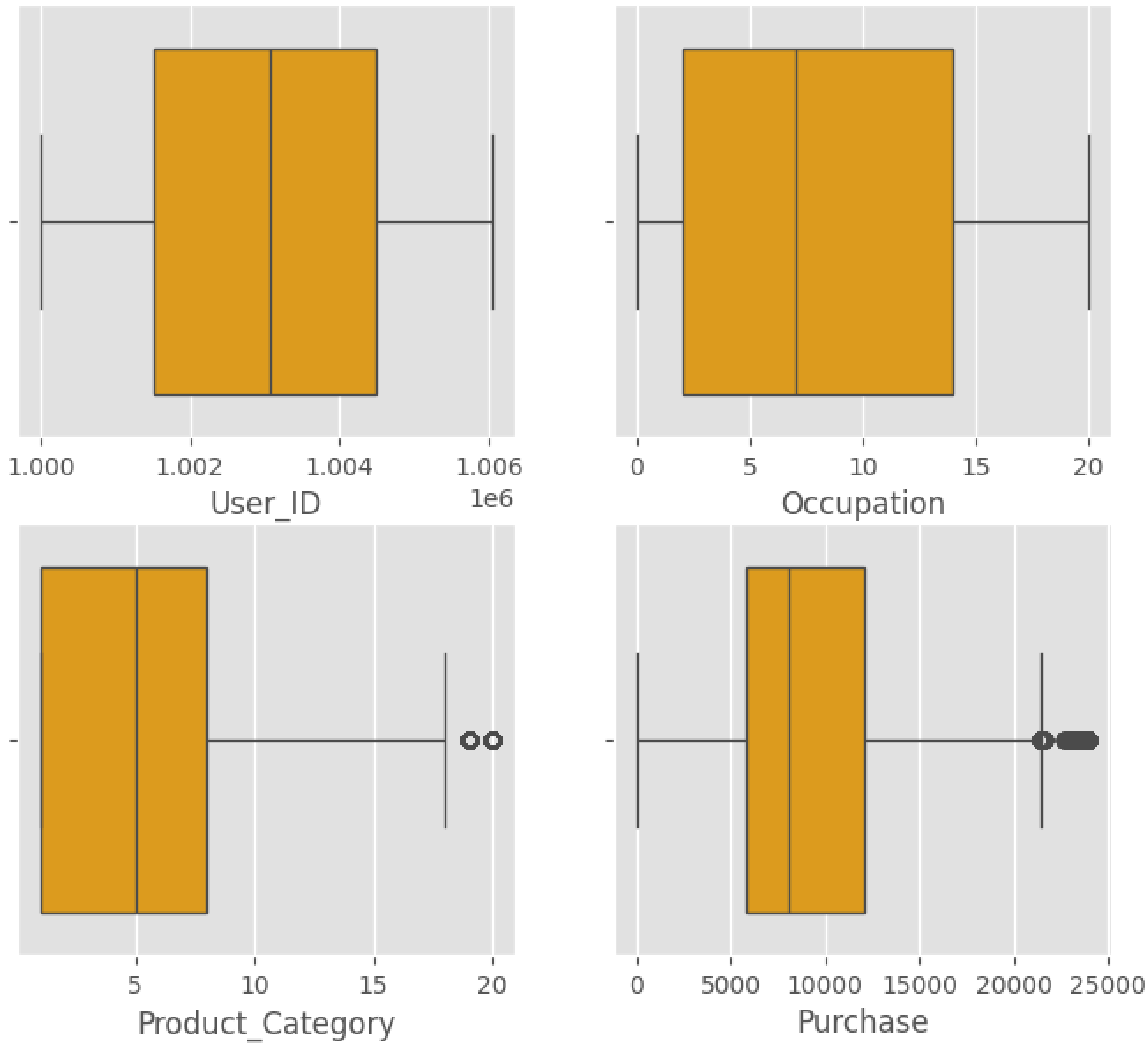
1. Data contains 550068 rows and 10 columns
2. No Missing Values
3. Unique 5891 user ids with 1001680 id with maximum transactions
4. Unique 3631 Product ids with P00265242 as the most selling product
5. Males are dominating the purchase with huge 414259 numbers
6. With 7 unique age groups, 26-35 is the group with maximum purchases

7. 21 unique occupations with 4 at the top
8. 3 unique city categories with B at the top
9. Customers with 1 year of stay in current city are the customers with maximum purchases
10. Customers with marital status 0 are the customers with most purchases
11. With 20 unique product categories 5 is at the top

2 .Detect Null values & Outliers (using boxplot, “describe” method by checking the difference between mean and median, isnull etc.)

Used Boxplot To Detect Outliers

In [3]:
```
fig, axis = plt.subplots(2, 2, figsize=(8, 7))
#fig.subplots_adjust(top=1.0)
sns.boxplot(data=df, x="User_ID", orient='h',
ax=axis[0,0],color='orange')
sns.boxplot(data=df, x="Occupation", orient='h',
ax=axis[0,1],color='orange')
sns.boxplot(data=df, x="Product_Category", orient='h',
ax=axis[1,0],color='orange')
sns.boxplot(data=df, x="Purchase", orient='h',
ax=axis[1,1],color='orange')
plt.show()
```



INSIGHTS:

- User Id and Occupation have no outliers
- Purchase have got significant number of outliers
- Product Category has got a couple of outliers

Clipping Data between 5 percentile and 95 percentile to handle outliers*

In [4]:
```
percentile_5=df['Purchase'].quantile(0.05)
percentile_95=df['Purchase'].quantile(0.95)
df['Purchase']=np.clip(df['Purchase'],percentile_5,percentile_95)
df.describe(include='all')
```

Out[4]:

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

Insights

Clipping data between 5 percentile and 95 percentile has modified the data within this range to handle outliers for accurate representation of majority of data

1. In Purchase, maximum value is reduced from 23961 to 19336
2. Minimum value is changed to 1984 from 12
3. Standard Deviation is reduced to 4855 from 5023

3.Do some data exploration steps like:

1.Tracking the amount spent per transaction of all the 50 million female customers, and all the 50 million male customers, calculate the average, and conclude the results. 2.Inference after computing the average female and male expenses. 3.Use the sample average to find out an interval within which the population average will lie. Using the sample of female customers you will calculate the interval within which the average spending of 50 million male and female customers may lie.

In [53]:

```
import numpy as np
import pandas as pd
from scipy import stats

# Simulate data (in dollars)
np.random.seed(42)
female_spend = np.random.normal(loc=52, scale=15, size=50000000)  # mean=52, sd=
male_spend = np.random.normal(loc=47, scale=18, size=50000000)    # mean=47, sd=
```

```python
# Calculate averages
avg_female = np.mean(female_spend)
avg_male = np.mean(male_spend)

avg_female, avg_male
```

Out[53]: (np.float64(51.99794452244972), np.float64(46.997239175093625))

In [54]:
```python
# Compute sample std deviations
std_female = np.std(female_spend, ddof=1)
std_male = np.std(male_spend, ddof=1)

# Confidence Intervals (95%)
z = 1.96
ci_female = (avg_female - z * std_female / np.sqrt(len(female_spend)),
             avg_female + z * std_female / np.sqrt(len(female_spend)))

ci_male = (avg_male - z * std_male / np.sqrt(len(male_spend)),
           avg_male + z * std_male / np.sqrt(len(male_spend)))

ci_female, ci_male
```

Out[54]:
```
((np.float64(51.99378636938585), np.float64(52.00210267551359)),
 (np.float64(46.99224954429444), np.float64(47.00222880589281)))
```

In [58]:
```python
#  Gender-based filtering
female_df = df[df["Gender"] == "F"]
male_df   = df[df["Gender"] == "M"]

#  Calculate averages
female_avg = female_df["Purchase"].mean()
male_avg   = male_df["Purchase"].mean()

print(f"\nAverage purchase - Female: {female_avg:.2f}")
print(f"Average purchase - Male   : {male_avg:.2f}")

# Inference
if male_avg > female_avg:
    print("\nInference: Male customers spend more on average per transaction.")
else:
    print("\nInference: Female customers spend more on average per transaction."

# 9. Confidence Interval function
def confidence_interval(data, confidence=0.95):
    n = len(data)
    mean = np.mean(data)
    sem = stats.sem(data, nan_policy='omit')
    h = sem * stats.t.ppf((1 + confidence) / 2, n - 1)
    return mean - h, mean + h

# 10. Compute 95% CI for both groups
female_ci = confidence_interval(female_df["Purchase"])
male_ci   = confidence_interval(male_df["Purchase"])

print(f"\n95% CI for Female Average Spend: ({female_ci[0]:.2f}, {female_ci[1]:.2
print(f"95% CI for Male Average Spend  : ({male_ci[0]:.2f}, {male_ci[1]:.2f})")
```

```
Average purchase - Female: 8736.54
Average purchase - Male   : 9427.24

Inference: Male customers spend more on average per transaction.

95% CI for Female Average Spend: (8712.09, 8760.99)
95% CI for Male Average Spend   : (9412.24, 9442.24)
```

Age Effect on Purchases - Confidence Interval / CLT

In [62]:
```
df['Age'].value_counts()
```

Out[62]:

Age	count
26-35	219587
36-45	110013
18-25	99660
46-50	45701
51-55	38501
55+	21504
0-17	15102

dtype: int64

Age Group 0-17

In [63]:
```
g17=df[df['Age']=='0-17']['Purchase']
mn_17=np.mean(g17)
std_17=np.std(g17)
n17=len(g17)
mn_17,std_17,n17
```

Out[63]:
```
(np.float64(8940.64905310555), 4940.43367702637, 15102)
```

In [64]:
```
norm.interval(confidence=0.95,loc=mn_17,scale=std_17/np.sqrt(n17))
```

Out[64]:
```
(np.float64(8861.854548111229), np.float64(9019.44355809987))
```

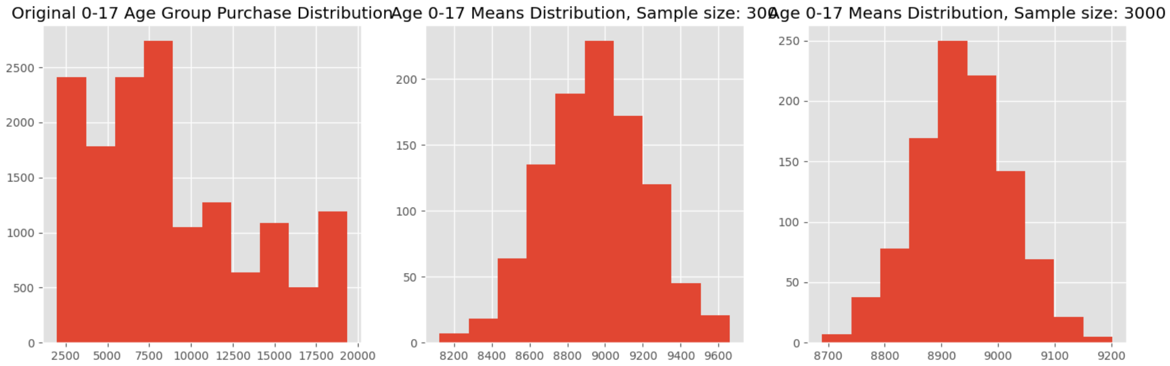
In [66]:
```
n17_300=300
n17_3000=3000
print(norm.interval(confidence=0.95,loc=mn_17,scale=std_17/np.sqrt(n17_300)))
print(norm.interval(confidence=0.95,loc=mn_17,scale=std_17/np.sqrt(n17_3000)))
```

```
(np.float64(8381.596626198298), np.float64(9499.701480012802))
(np.float64(8763.861153058378), np.float64(9117.43695315272))
```

In [68]:
```
sample_g17_300 = [np.mean(g17.sample(300)) for i in range(1000)]
sample_g17_3000 = [np.mean(g17.sample(3000)) for i in range(1000)]
print(np.mean(sample_g17_300))
print(np.mean(sample_g17_3000))
```

```
8949.534836666668
8938.117919333334
```

In [69]:
```
fig, axis = plt.subplots(nrows=1, ncols=3, figsize=(17, 5))
axis[0].hist(g17)
axis[1].hist(sample_g17_300)
axis[2].hist(sample_g17_3000)
axis[0].set_title('Original 0-17 Age Group Purchase Distribution')
axis[1].set_title("Age 0-17 Means Distribution, Sample size: 300")
axis[2].set_title("Age 0-17 Means Distribution, Sample size: 3000")
plt.show()
```



Age Group 18-25

In [71]:
```
g25=df[df['Age']=='18-25']['Purchase']
mn_25=np.mean(g25)
std_25=np.std(g25)
n25=len(g25)
mn_25,std_25,n25
```

Out[71]:
```
(np.float64(9169.010977322898), 4889.406153689914, 99660)
```

In [72]:
```
norm.interval(confidence=0.95,loc=mn_25,scale=std_25/np.sqrt(n25))
```

Out[72]:
```
(np.float64(9138.655031826722), np.float64(9199.366922819074))
```

In [74]:
```
n25_3000=3000
n25_30000=30000
print(norm.interval(confidence=0.95,loc=mn_25,scale=std_25/np.sqrt(n25_3000)))
print(norm.interval(confidence=0.95,loc=mn_25,scale=std_17/np.sqrt(n25_30000)))
```

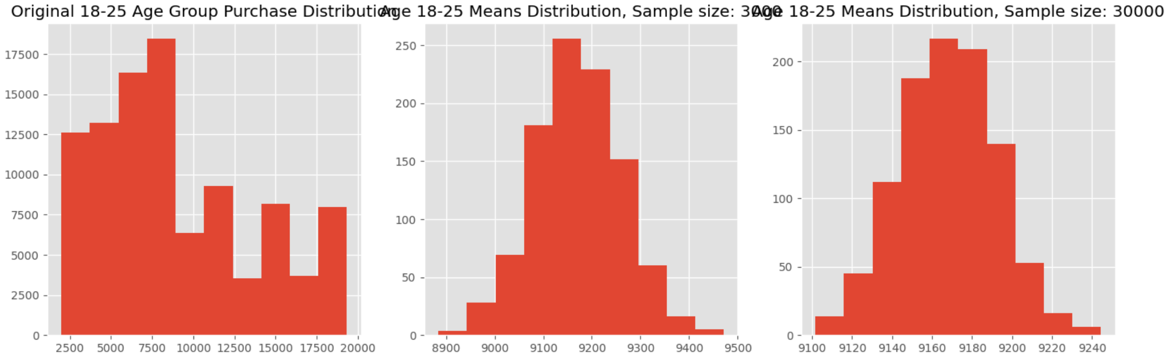
```
(np.float64(8994.049040194166), np.float64(9343.97291445163))
(np.float64(9113.105734632172), np.float64(9224.916220013623))
```

In [75]:
```
sample_g25_3000 = [np.mean(g25.sample(3000)) for i in range(1000)]
sample_g25_30000 = [np.mean(g25.sample(30000)) for i in range(1000)]
print(np.mean(sample_g25_3000))
print(np.mean(sample_g25_30000))
```

```
9170.489800000001
9168.111591733334
```

In [76]:
```
fig, axis = plt.subplots(nrows=1, ncols=3, figsize=(17, 5))
axis[0].hist(g25)
axis[1].hist(sample_g25_3000)
axis[2].hist(sample_g25_30000)
axis[0].set_title('Original 18-25 Age Group Purchase Distribution')
axis[1].set_title("Age 18-25 Means Distribution, Sample size: 3000")
```

```
axis[2].set_title("Age 18-25 Means Distribution, Sample size: 30000")
plt.show()
```



Age Group 26-35

In [77]:
```
g35=df[df['Age']=='26-35']['Purchase']
mn_35=np.mean(g35)
std_35=np.std(g35)
n35=len(g35)
mn_35,std_35,n35
```

Out[77]: (np.float64(9243.780119041656), 4855.1809978569545, 219587)

In [78]:
```
norm.interval(confidence=0.95,loc=mn_35,scale=std_35/np.sqrt(n35))
```

Out[78]: (np.float64(9223.472911701543), np.float64(9264.087326381768))

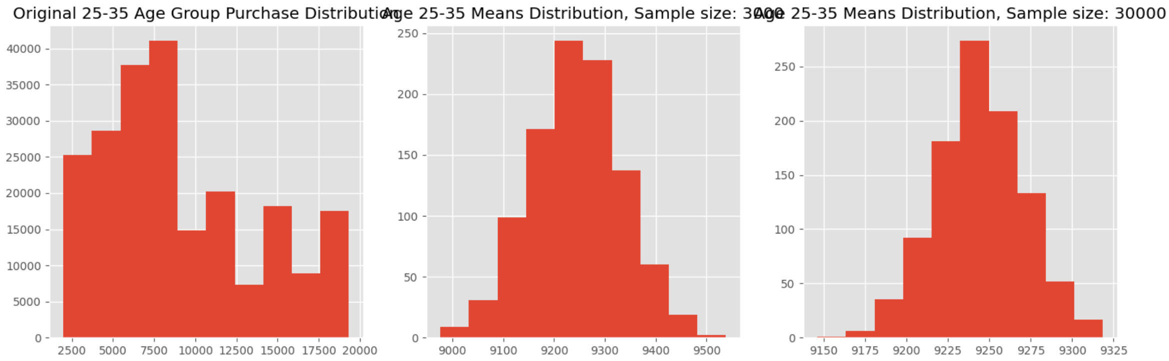
In [79]:
```
n35_3000=3000
n35_30000=30000
print(norm.interval(confidence=0.95,loc=mn_35,scale=std_35/np.sqrt(n35_3000)))
print(norm.interval(confidence=0.95,loc=mn_35,scale=std_35/np.sqrt(n35_30000)))
```

```
(np.float64(9070.042890880606), np.float64(9417.517347202705))
(np.float64(9188.83958350633), np.float64(9298.720654576982))
```

In [80]:
```
sample_g35_3000 = [np.mean(g35.sample(3000)) for i in range(1000)]
sample_g35_30000 = [np.mean(g35.sample(30000)) for i in range(1000)]
print(np.mean(sample_g35_3000))
print(np.mean(sample_g35_30000))
```

```
9245.009237
9244.271427099999
```

In [81]:
```
fig, axis = plt.subplots(nrows=1, ncols=3, figsize=(17, 5))
axis[0].hist(g35)
axis[1].hist(sample_g35_3000)
axis[2].hist(sample_g35_30000)
axis[0].set_title('Original 25-35 Age Group Purchase Distribution')
axis[1].set_title("Age 25-35 Means Distribution, Sample size: 3000")
axis[2].set_title("Age 25-35 Means Distribution, Sample size: 30000")
plt.show()
```



Age Group 36-45

In [82]:
```
g45=df[df['Age']=='36-45']['Purchase']
mn_45=np.mean(g45)
std_45=np.std(g45)
n45=len(g45)
mn_45,std_45,n45
```

Out[82]: `(np.float64(9322.92190922891), 4847.575809950491, 110013)`

In [83]:
```
norm.interval(confidence=0.95,loc=mn_45,scale=std_45/np.sqrt(n45))
```

Out[83]: `(np.float64(9294.276785879065), np.float64(9351.567032578754))`

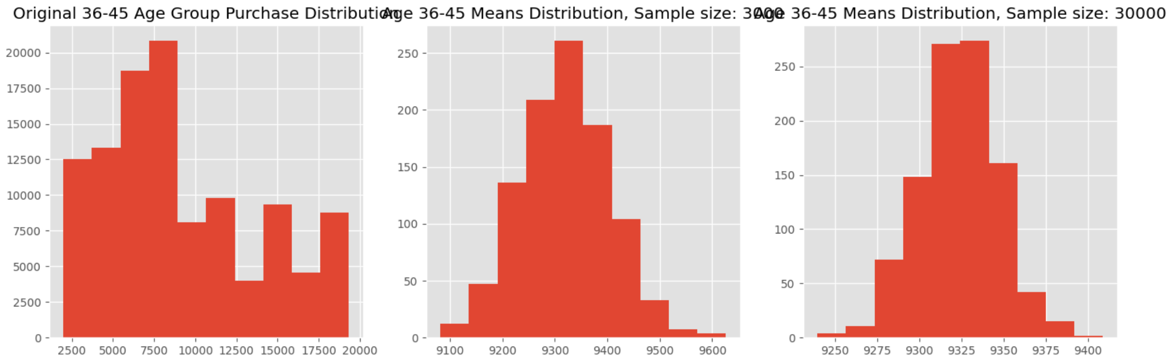
In [84]:
```
n45_3000=3000
n45_30000=30000
print(norm.interval(confidence=0.95,loc=mn_45,scale=std_45/np.sqrt(n45_3000)))
print(norm.interval(confidence=0.95,loc=mn_45,scale=std_45/np.sqrt(n45_30000)))
```

```
(np.float64(9149.456824221143), np.float64(9496.386994236676))
(np.float64(9268.067432914982), np.float64(9377.776385542837))
```

In [85]:
```
sample_g45_3000 = [np.mean(g45.sample(3000)) for i in range(1000)]
sample_g45_30000 = [np.mean(g45.sample(30000)) for i in range(1000)]
print(np.mean(sample_g45_3000))
print(np.mean(sample_g45_30000))
```

```
9319.89306
9323.294276133332
```

In [86]:
```
fig, axis = plt.subplots(nrows=1, ncols=3, figsize=(17, 5))
axis[0].hist(g45)
axis[1].hist(sample_g45_3000)
axis[2].hist(sample_g45_30000)
axis[0].set_title('Original 36-45 Age Group Purchase Distribution')
axis[1].set_title("Age 36-45 Means Distribution, Sample size: 3000")
axis[2].set_title("Age 36-45 Means Distribution, Sample size: 30000")
plt.show()
```



Age Group 46-50

In [87]:
```python
g50=df[df['Age']=='46-50']['Purchase']
mn_50=np.mean(g50)
std_50=np.std(g50)
n50=len(g50)
mn_50,std_50,n50
```

Out[87]:
```
(np.float64(9204.211483337345), 4785.889795462206, 45701)
```

In [88]:
```python
norm.interval(confidence=0.95,loc=mn_50,scale=std_50/np.sqrt(n50))
```

Out[88]:
```
(np.float64(9160.333371235685), np.float64(9248.089595439005))
```

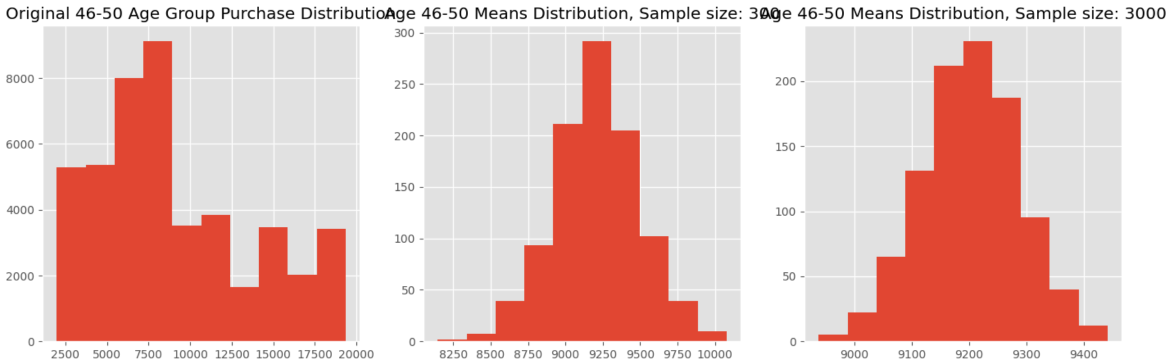
In [91]:
```python
n50_300=300
n50_3000=3000
print(norm.interval(confidence=0.95,loc=mn_50,scale=std_50/np.sqrt(n50_300)))
print(norm.interval(confidence=0.95,loc=mn_50,scale=std_50/np.sqrt(n50_3000)))
```

```
(np.float64(8662.64702159677), np.float64(9745.775945077921))
(np.float64(9032.95376344701), np.float64(9375.46920322768))
```

In [92]:
```python
sample_g50_300 = [np.mean(g50.sample(300)) for i in range(1000)]
sample_g50_3000 = [np.mean(g50.sample(3000)) for i in range(1000)]
print(np.mean(sample_g50_300))
print(np.mean(sample_g50_3000))
```

```
9210.210676666666
9202.833570666668
```

In [93]:
```python
fig, axis = plt.subplots(nrows=1, ncols=3, figsize=(17, 5))
axis[0].hist(g50)
axis[1].hist(sample_g50_300)
axis[2].hist(sample_g50_3000)
axis[0].set_title('Original 46-50 Age Group Purchase Distribution')
axis[1].set_title("Age 46-50 Means Distribution, Sample size: 300")
axis[2].set_title("Age 46-50 Means Distribution, Sample size: 3000")
plt.show()
```



Age Group 51-55

In [96]:
```
g55=df[df['Age']=='51-55']['Purchase']
mn_55=np.mean(g55)
std_55=np.std(g55)
n55=len(g55)
mn_55,std_55,n55
```

Out[96]: (np.float64(9514.863250305187), 4873.566375511186, 38501)

In [97]:
```
norm.interval(confidence=0.95,loc=mn_55,scale=std_55/np.sqrt(n55))
```

Out[97]: (np.float64(9466.182308527832), np.float64(9563.544192082541))

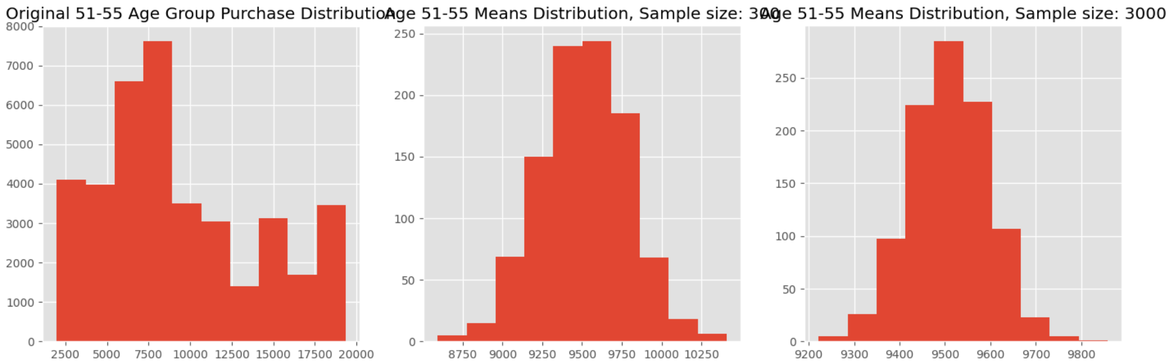
In [98]:
```
n55_300=300
n55_3000=3000
print(norm.interval(confidence=0.95,loc=mn_55,scale=std_55/np.sqrt(n55_300)))
print(norm.interval(confidence=0.95,loc=mn_55,scale=std_55/np.sqrt(n55_3000)))
```

```
(np.float64(8963.377431845009), np.float64(10066.349068765365))
(np.float64(9340.468121943557), np.float64(9689.258378666816))
```

In [99]:
```
sample_g55_300 = [np.mean(g55.sample(300)) for i in range(1000)]
sample_g55_3000 = [np.mean(g55.sample(3000)) for i in range(1000)]
print(np.mean(sample_g55_300))
print(np.mean(sample_g55_3000))
```

```
9514.603576666668
9509.267209
```

In [100…
```
fig, axis = plt.subplots(nrows=1, ncols=3, figsize=(17, 5))
axis[0].hist(g55)
axis[1].hist(sample_g55_300)
axis[2].hist(sample_g55_3000)
axis[0].set_title('Original 51-55 Age Group Purchase Distribution')
axis[1].set_title("Age 51-55 Means Distribution, Sample size: 300")
axis[2].set_title("Age 51-55 Means Distribution, Sample size: 3000")
plt.show()
```

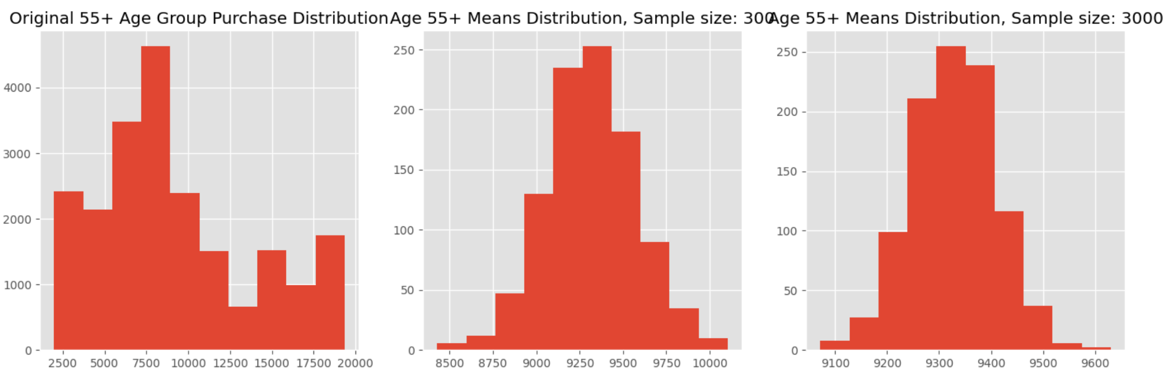


Age Group 55+

In [102…

```
g55p=df[df['Age']=='55+']['Purchase']
mn_55p=np.mean(g55p)
std_55p=np.std(g55p)
n55p=len(g55p)
mn_55p,std_55p,n55p

norm.interval(confidence=0.95,loc=mn_55p,scale=std_55p/np.sqrt(n55p))

n55p_300=300
n55p_3000=3000
print(norm.interval(confidence=0.95,loc=mn_55p,scale=std_55p/np.sqrt(n55p_300)))
print(norm.interval(confidence=0.95,loc=mn_55p,scale=std_55p/np.sqrt(n55p_3000)))

sample_g55p_300 = [np.mean(g55p.sample(300)) for i in range(1000)]
sample_g55p_3000 = [np.mean(g55p.sample(3000)) for i in range(1000)]
print(np.mean(sample_g55p_300))
print(np.mean(sample_g55p_3000))

fig, axis = plt.subplots(nrows=1, ncols=3, figsize=(17, 5))
axis[0].hist(g55p)
axis[1].hist(sample_g55p_300)
axis[2].hist(sample_g55p_3000)
axis[0].set_title('Original 55+ Age Group Purchase Distribution')
axis[1].set_title("Age 55+ Means Distribution, Sample size: 300")
axis[2].set_title("Age 55+ Means Distribution, Sample size: 3000")
plt.show()
```

```
(np.float64(8786.918400003362), np.float64(9868.67469895497))
(np.float64(9156.755860583111), np.float64(9498.837238375221))
9317.047166666665
9328.522041333332
```



INSIGHTS:

1. Age Group 0-17: Confidence Interval- (8861.85, 9019.44) Avg. Purchase- 8940
2. Age Group 18-25: Confidence Interval- (9138.65, 9199.36) Avg. Purchase- 9169
3. Age Group 26-35: Confidence Interval- (9223.78, 9264.08) Avg. Purchase- 9243
4. Age Group 36-45: Confidence Interval- (9294.27, 9351.56) Avg. Purchase- 9322
5. Age Group 46-50: Confidence Interval- (9160.33, 9248.08) Avg. Purchase- 9204
6. Age Group 51-55: Confidence Interval- (9466.18, 9563.54) Avg. Purchase- 9514
7. Age Group 55+: Confidence Interval- (9263.91, 9391.68) Avg. Purchase- 9327
8. Avg. Purchase is the highest for 51-55 age group
9. It is observed that as the sample size increases, width of the confidence interval decreases
10. In most of the cases of different sample sizes, confidence intervals are overlapping
11. As the sample size increases, the sample mean gets closer to the population mean and the shape of the distribution of the means get narrower

In [ ]:

```

```

4.Use the Central limit theorem to compute the interval. Change the sample size to observe the distribution of the mean of the expenses by female and male customers.

The interval that you calculated is called Confidence Interval. The width of the interval is mostly decided by the business: Typically 90%, 95%, or 99%. Play around with the width parameter and report the observations.

Gender vs Purchase

In [ ]:

```
# gender v/s purchase
df.groupby('Gender')['Purchase'].value_counts()
```

Out[ ]:

		count
Gender	**Purchase**	
F	**7108**	68
	6856	65
	6879	65
	6938	63
	7060	63
...	**...**	...
M	**23943**	1
	23945	1
	23952	1
	23956	1
	23959	1

32251 rows × 1 columns

dtype: int64

In [ ]:

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

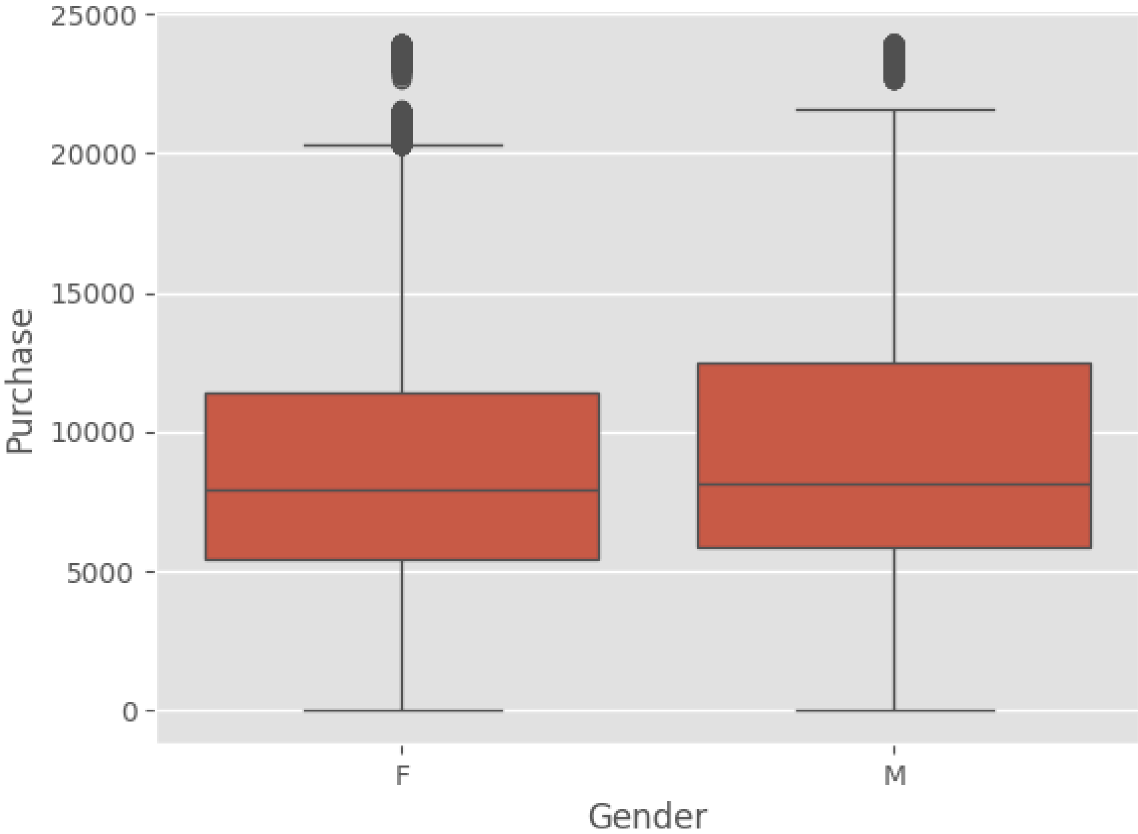
Out[ ]:

	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

In [ ]:

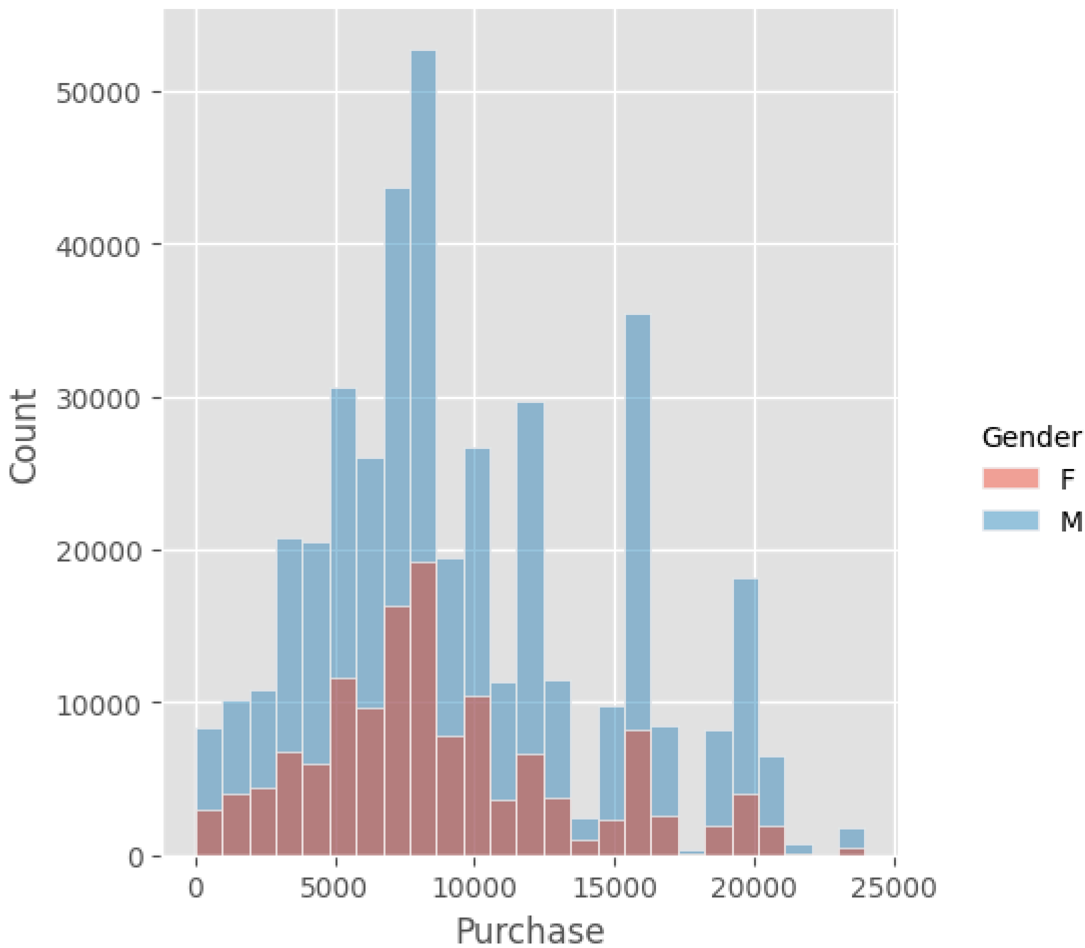
```
sns.boxplot(x='Gender',y='Purchase',data=df)
```

Out[ ]: <Axes: xlabel='Gender', ylabel='Purchase'>



In [ ]:
```
sns.displot(x='Purchase', hue='Gender', data=df, bins=25)
```

Out[ ]: <seaborn.axisgrid.FacetGrid at 0x7f493b92e960>



In [ ]:
```
# CLT
# sample> mean of sample> repeat


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

Out[ ]:

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

In [ ]:
```
df.sample(300).groupby('Gender')['Purchase'].describe()
```

Out[ ]:

Gender	count	mean	std	min	25%	50%	75%	max
F	86.0	8906.697674	4895.431462	2048.0	5458.00	7490.0	10075.50	23664.0
M	214.0	9662.364486	5172.259200	701.0	5460.75	8220.5	12920.75	23827.0

In [ ]:
```
sample_size = 300
iterations = 1000
```

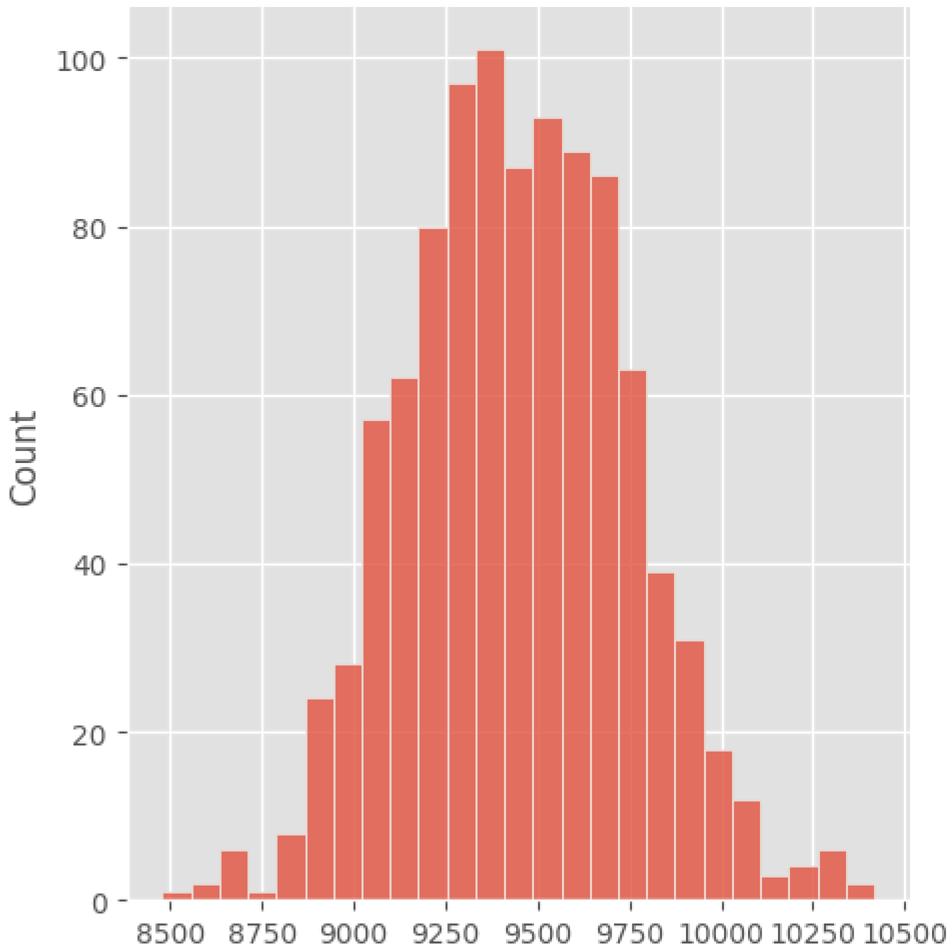
In [ ]:
```
df_males = df[df.Gender=='M']
male_spends = []
for iter in range(iterations):
    male_spends.append(
    df_males.sample(sample_size)['Purchase'].mean())
```

In [ ]:
```
df_females = df[df.Gender=='F']
female_spends = []
for iter in range(iterations):
    female_spends.append(
    df_females.sample(sample_size)['Purchase'].mean())
```

In [ ]:
```
print(np.mean(male_spends))
sns.displot(male_spends, bins=25)
```

```
9444.826949999999
```

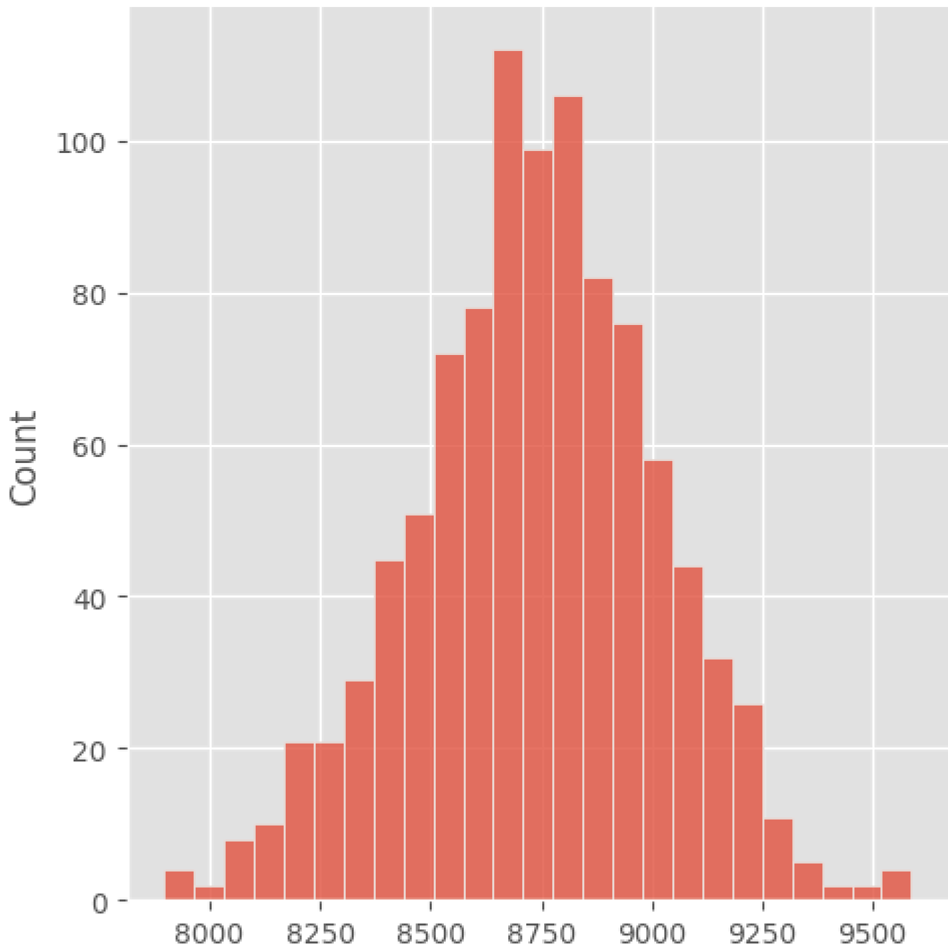
Out[ ]:
```
<seaborn.axisgrid.FacetGrid at 0x7fe0d2be73b0>
```



In [ ]:
```
print(np.mean(female_spends))
sns.displot(female_spends, bins=25)
```

```
8736.381166666668
```

Out[ ]:
```
<seaborn.axisgrid.FacetGrid at 0x7fe0d295f470>
```



z-score

95% confidence interval

min = mean-1.96*std_error==stdv

max = mean+1.96*std_error

For Males

In [ ]:
```
min_male = np.mean(male_spends) - 1.96*np.std(male_spends)
max_male = np.mean(male_spends) + 1.96*np.std(male_spends)
print(min_male, max_male)
```

```
8858.266247256119 10031.387652743879
```

For Females

In [ ]:
```
min_female = np.mean(female_spends) - 1.96*np.std(female_spends)
max_female = np.mean(female_spends) + 1.96*np.std(female_spends)
print(min_female, max_female)
```

```
8199.783265917285 9272.97906741605
```

Percentiles

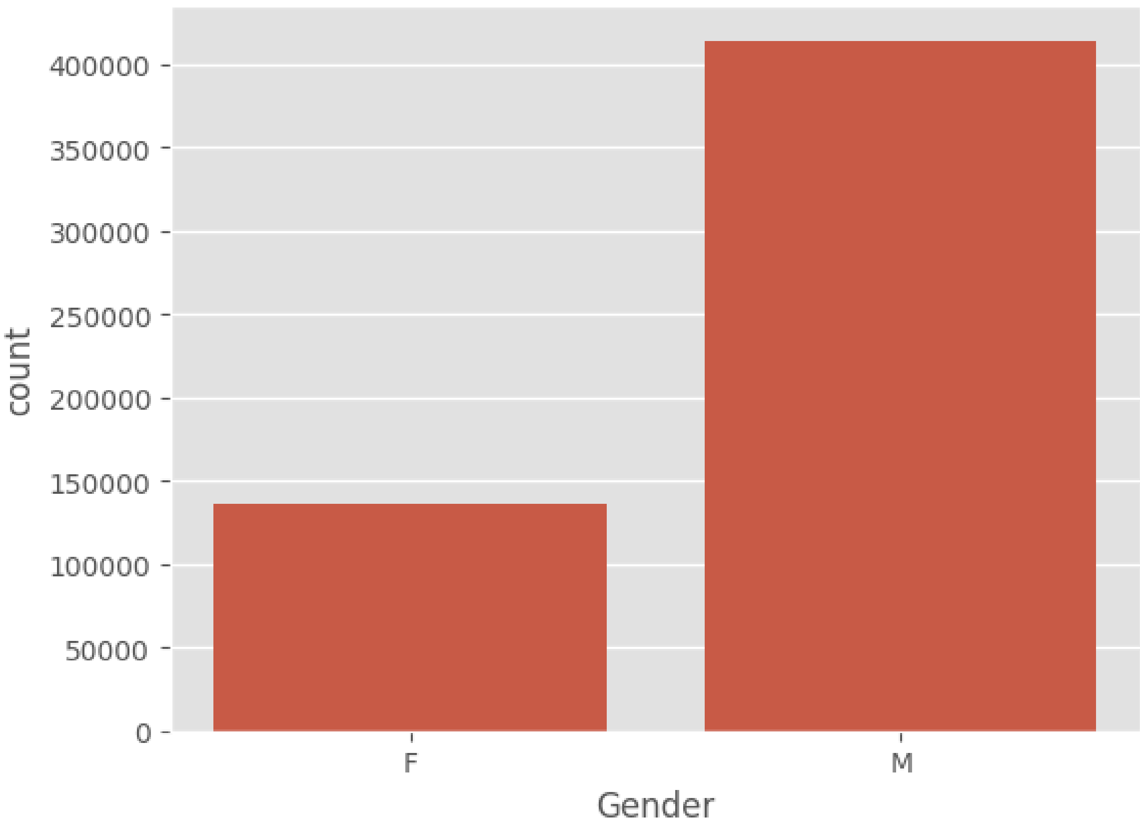
In [ ]:
```
print(np.percentile(male_spends,[2.5,97.5]))
print(np.percentile(female_spends,[2.5,97.5]))
```

```
[ 8922.09191667 10030.65258333]
[8171.21208333 9241.174     ]
```

The below code generates a visually appealing count plot to showcase the distribution of gender in the dataset

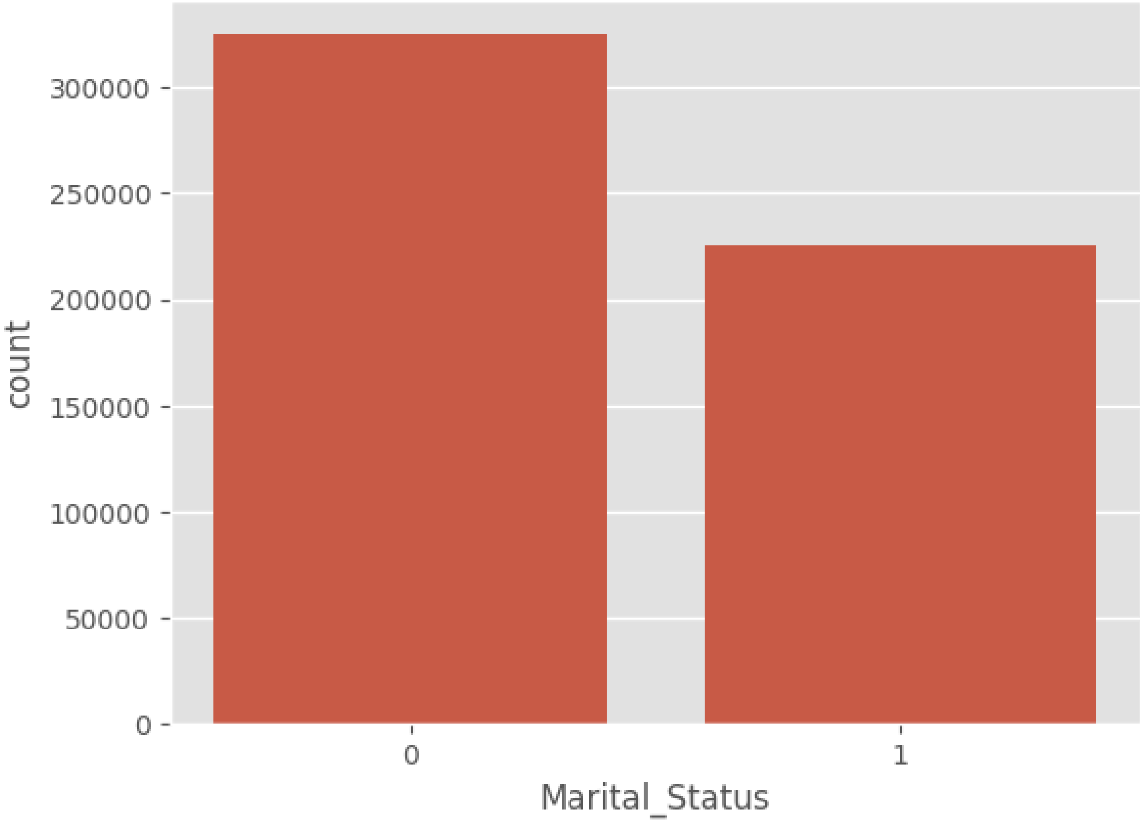
In [ ]:
```
sns.countplot(data = df, x = 'Gender')
plt.plot()   # displaying the plot
```

Out[ ]: []



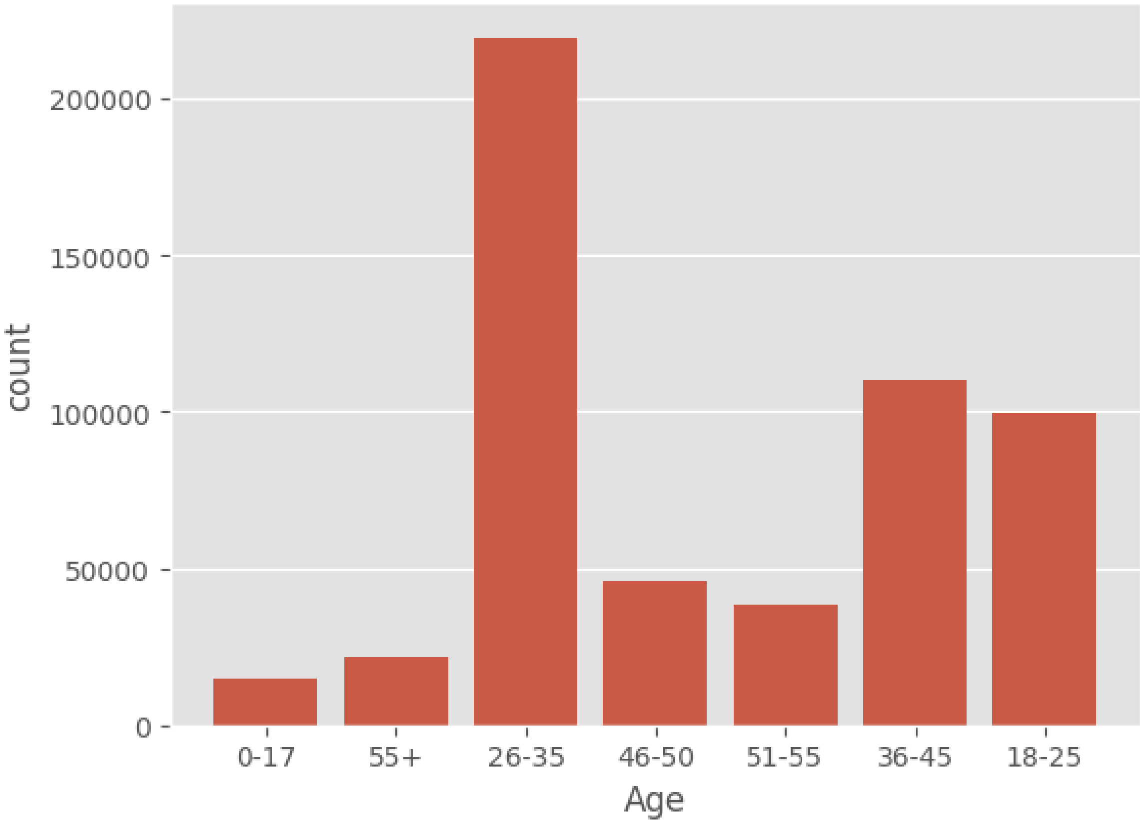
In [ ]:
```
sns.countplot(data = df, x = 'Marital_Status')
plt.plot()   # displaying the plot
```

Out[ ]: []



In [ ]:
```
sns.countplot(data = df, x = 'Age')
plt.plot()    # displaying the plot
```

Out[ ]: []



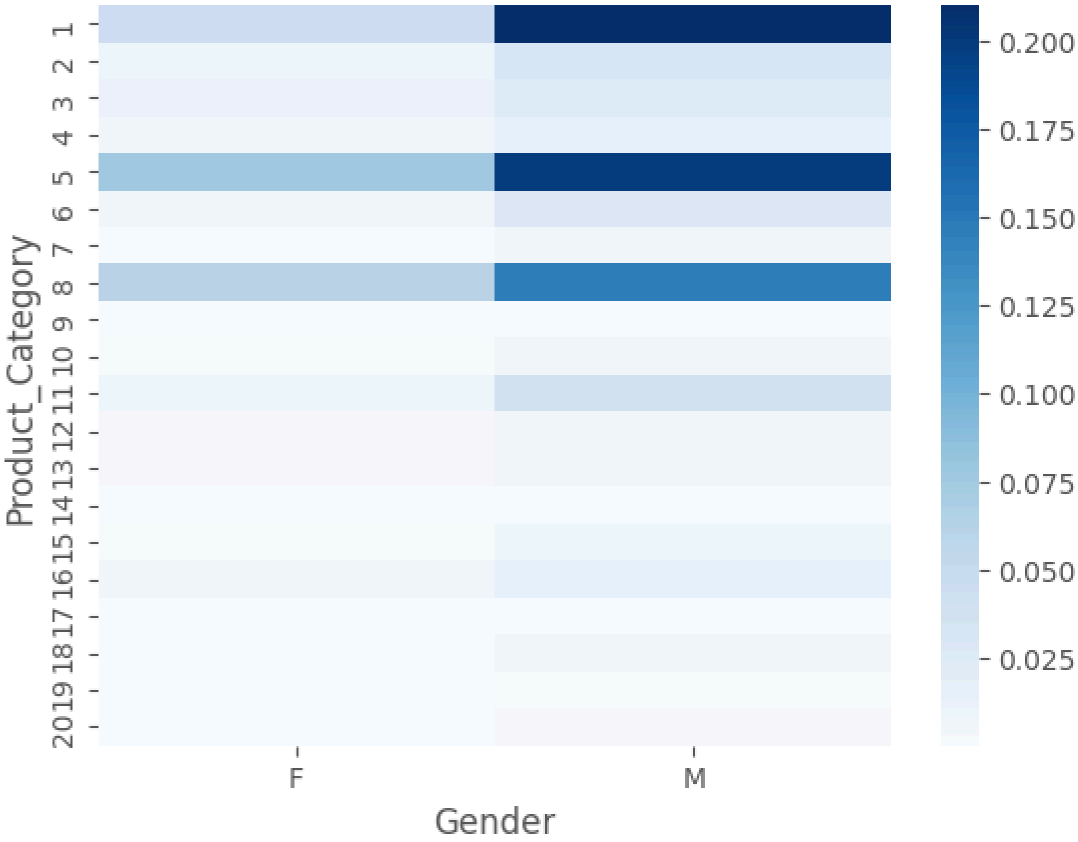
Product Categories preferred by different Genders

In [107…
```
pd.crosstab(df['Product_Category'], df['Gender'], normalize=True)
```

Out[107…

Gender	F	M
Product_Category		
1	0.045142	0.210059
2	0.010286	0.033098
3	0.010919	0.025828
4	0.006616	0.014751
5	0.076283	0.198106
6	0.008288	0.028918
7	0.001714	0.005050
8	0.061007	0.146104
9	0.000127	0.000618
10	0.002112	0.007205
11	0.008615	0.035537
12	0.002785	0.004390
13	0.002658	0.007430
14	0.001133	0.001636
15	0.001902	0.009533
16	0.004367	0.013500
17	0.000113	0.000938
18	0.000694	0.004987
19	0.000820	0.002094
20	0.001314	0.003321

In [109…
```
sns.heatmap(pd.crosstab(df['Product_Category'], df['Gender'],
normalize=True),
 cmap='Blues')
plt.show()
```



In [141…

```
df.dtypes
```

Out[141…

	0
User_ID	int64
Product_ID	object
Gender	object
Age	object
Occupation	int64
City_Category	object
Stay_In_Current_City_Years	object
Marital_Status	int64
Product_Category	int64
Purchase	int64

dtype: object

In [147…

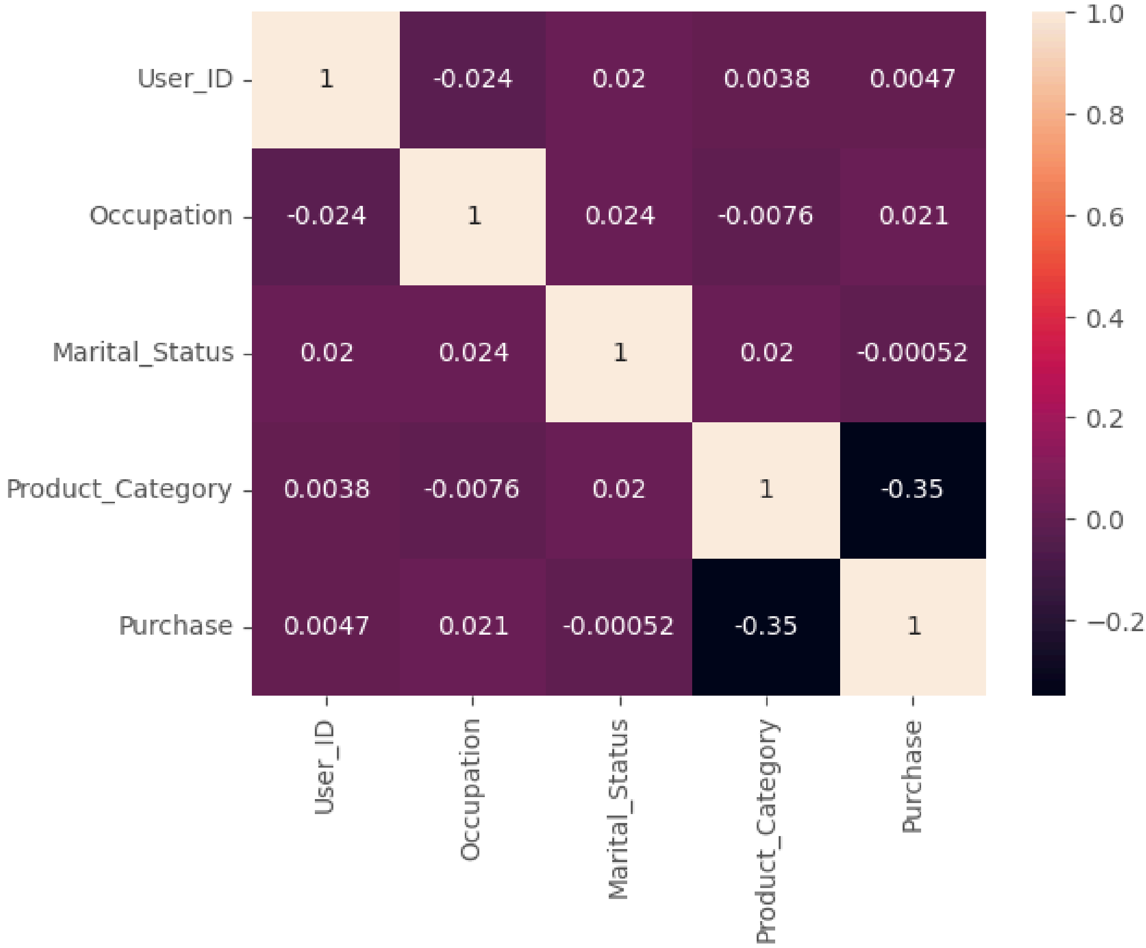
```
df.corr(numeric_only=True)
```

Out[147…

	User_ID	Occupation	Marital_Status	Product_Category	Purchase
User_ID	1.000000	-0.023971	0.020443	0.003825	0.004658
Occupation	-0.023971	1.000000	0.024280	-0.007618	0.021220
Marital_Status	0.020443	0.024280	1.000000	0.019888	-0.000522
Product_Category	0.003825	-0.007618	0.019888	1.000000	-0.347437
Purchase	0.004658	0.021220	-0.000522	-0.347437	1.000000

In [149…

```
sns.heatmap(df.corr(numeric_only=True),annot=True)
plt.show()
```

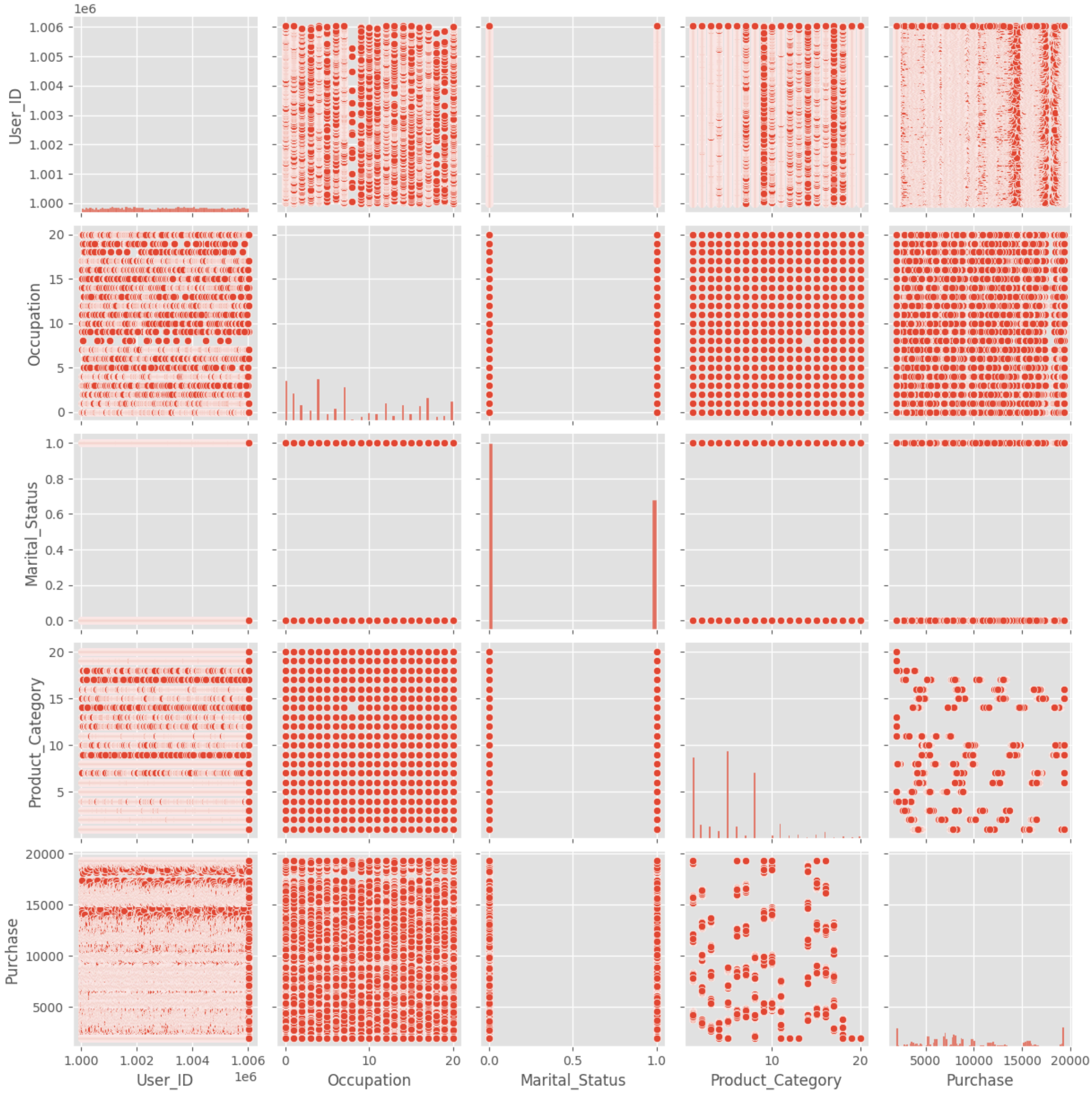


In [106…

```
sns.pairplot(data=df)
```

Out[106…

```
<seaborn.axisgrid.PairGrid at 0x7a6849faf800>
```



INSIGHTS:

1. From correlation function and the heatmap , it is observed that there is no significant correlation among any pair of attributes.
2. Pairplot and Heatmap show some correlation among few attributes but it is not coming out significantly.

5.Conclude the results and check if the confidence intervals of average male and female spends are overlapping or not overlapping. How can Walmart leverage this conclusion to make changes or improvements?

Gender Effect on Purchases - 95% Confidence Interval / CLT

In [15]:
```
dfmen=df[df['Gender']=='M']['Purchase']
dfmen
```

Out[15]:

	Purchase
4	7969
5	15227
6	19215
7	15854
8	15686
...	...
550057	1984
550058	1984
550060	1984
550062	1984
550063	1984

414259 rows × 1 columns

dtype: int64

In [16]:
```
dfwomen=df[df['Gender']=='F']['Purchase']
dfwomen
```

Out[16]:

	Purchase
0	8370
1	15200
2	1984
3	1984
14	5378
...	...
550061	1984
550064	1984
550065	1984
550066	1984
550067	1984

135809 rows × 1 columns

dtype: int64

In [17]:
```
m_mean=round(np.mean(dfmen),2)
f_mean=round(np.mean(dfwomen),2)
m_mean, f_mean
```

Out[17]: (np.float64(9427.24), np.float64(8736.54))

In [18]:
```
m_std=round(np.std(dfmen),2)
f_std=round(np.std(dfwomen),2)
m_std, f_std
```

Out[18]: (4925.95, 4596.97)

In [19]:
```
mn=len(dfmen)
fn=len(dfwomen)
mn, fn
```

Out[19]: (414259, 135809)

Male Data Confidence Interval & Distribution of Means

In [20]:
```
norm.interval(confidence=0.95, loc=m_mean, scale=m_std/np.sqrt(mn))
```

Out[20]: (np.float64(9412.239625076156), np.float64(9442.240374923844))

In [21]:
```
mn1=300
norm.interval(confidence=0.95, loc=m_mean, scale=m_std/np.sqrt(mn1))
```

Out[21]: (np.float64(8869.826525322747), np.float64(9984.653474677252))

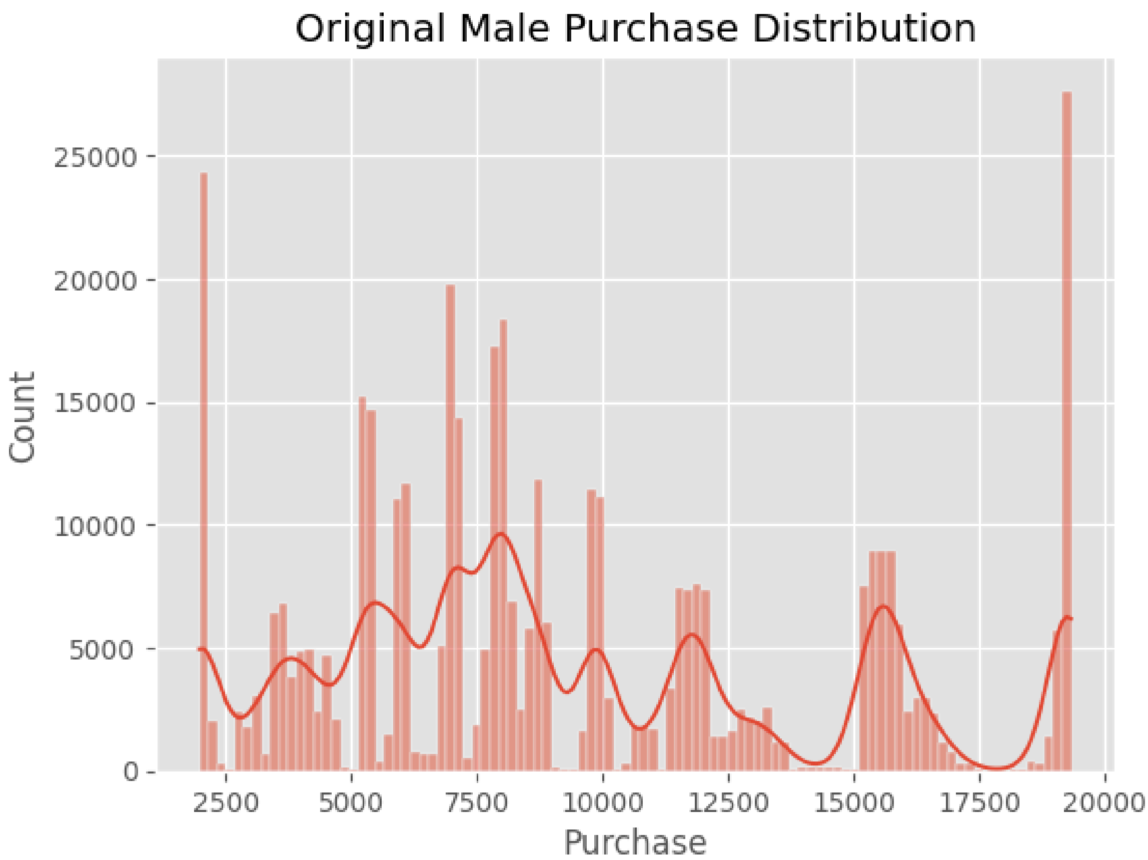
In [22]:
```
mn2=3000
norm.interval(confidence=0.95, loc=m_mean, scale=m_std/np.sqrt(mn2))
```

Out[22]: (np.float64(9250.970382155128), np.float64(9603.509617844871))

In [23]:
```
mn3=30000
norm.interval(confidence=0.95, loc=m_mean, scale=m_std/np.sqrt(mn3))
```

Out[23]: (np.float64(9371.498652532275), np.float64(9482.981347467725))

In [25]:
```
sns.histplot(data=dfmen,kde=True).set_title("Original Male Purchase Distribution
plt.show()
```



In [26]:
```
sample_mean_300 = [np.mean(dfmen.sample(300)) for i in range(1000)]
np.mean(sample_mean_300)
```

Out[26]: np.float64(9444.00903)

In [27]:
```
sample_mean_3000 = [np.mean(dfmen.sample(3000)) for i in range(1000)]
np.mean(sample_mean_3000)
```

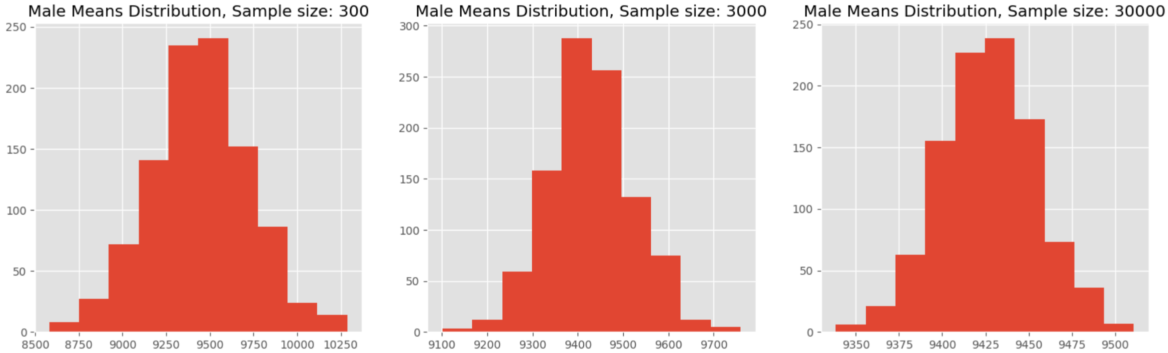
Out[27]: np.float64(9429.726053000002)

In [28]:
```
sample_mean_30000 = [np.mean(dfmen.sample(30000)) for i in
range(1000)]
np.mean(sample_mean_30000)
```

Out[28]: np.float64(9426.867031333333)

In [29]:
```
fig, axis = plt.subplots(nrows=1, ncols=3, figsize=(18, 5))
axis[0].hist(sample_mean_300)
```

```
axis[1].hist(sample_mean_3000)
axis[2].hist(sample_mean_30000)
axis[0].set_title("Male Means Distribution, Sample size: 300")
axis[1].set_title("Male Means Distribution, Sample size: 3000")
axis[2].set_title("Male Means Distribution, Sample size: 30000")
plt.show()
```



Female Data Confidence Interval & Distribution of Means

In [30]:
```
norm.interval(confidence=0.95, loc=f_mean, scale=f_std/np.sqrt(fn))
```

Out[30]: (np.float64(8712.09131613775), np.float64(8760.988683862251))

In [31]:
```
fn1=300
norm.interval(confidence=0.95, loc=f_mean, scale=f_std/np.sqrt(fn1))
```

Out[31]: (np.float64(8216.353432802387), np.float64(9256.726567197615))

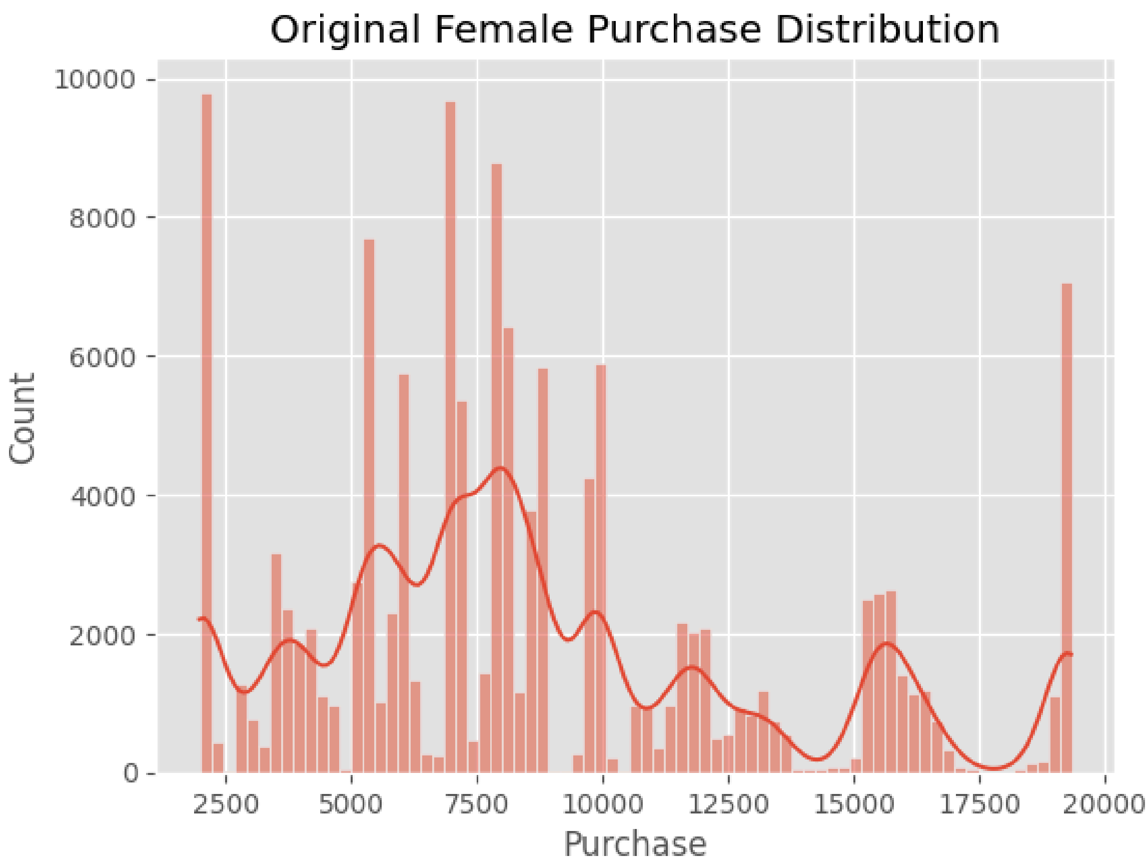
In [32]:
```
fn2=3000
norm.interval(confidence=0.95, loc=f_mean, scale=f_std/np.sqrt(fn2))
```

Out[32]: (np.float64(8572.042563943132), np.float64(8901.03743605687))

In [33]:
```
fn3=30000
norm.interval(confidence=0.95, loc=f_mean, scale=f_std/np.sqrt(fn3))
```

Out[33]: (np.float64(8684.52134328024), np.float64(8788.558656719762))

In [35]:
```
sns.histplot(data=dfwomen,kde=True).set_title("Original Female Purchase Distribu
plt.show()
```



In [36]:
```
sample_wmean_300 = [np.mean(dfwomen.sample(300)) for i in range(1000)]
np.mean(sample_wmean_300)
```

Out[36]: np.float64(8735.129276666667)

In [37]:
```
sample_wmean_3000 = [np.mean(dfwomen.sample(3000)) for i in
range(1000)]
np.mean(sample_wmean_3000)
```

Out[37]: np.float64(8735.554777000001)

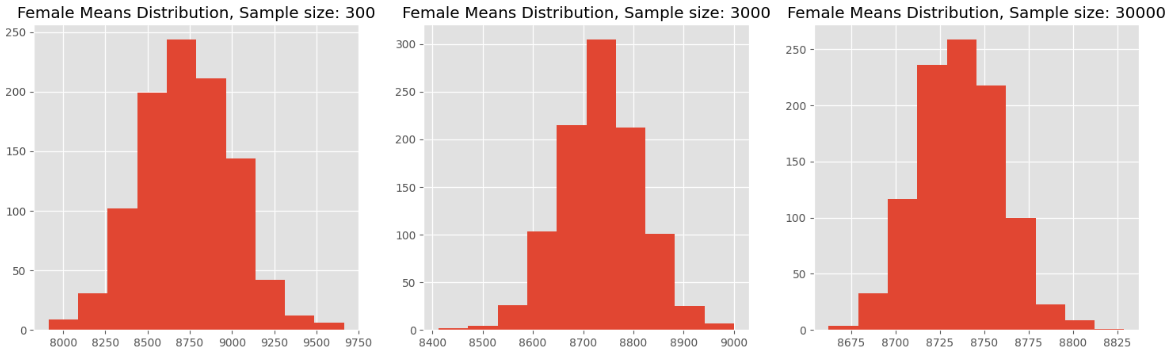
In [38]:
```
sample_wmean_30000 = [np.mean(dfwomen.sample(30000)) for i in
range(1000)]
np.mean(sample_wmean_30000)
```

Out[38]: np.float64(8735.080013266666)

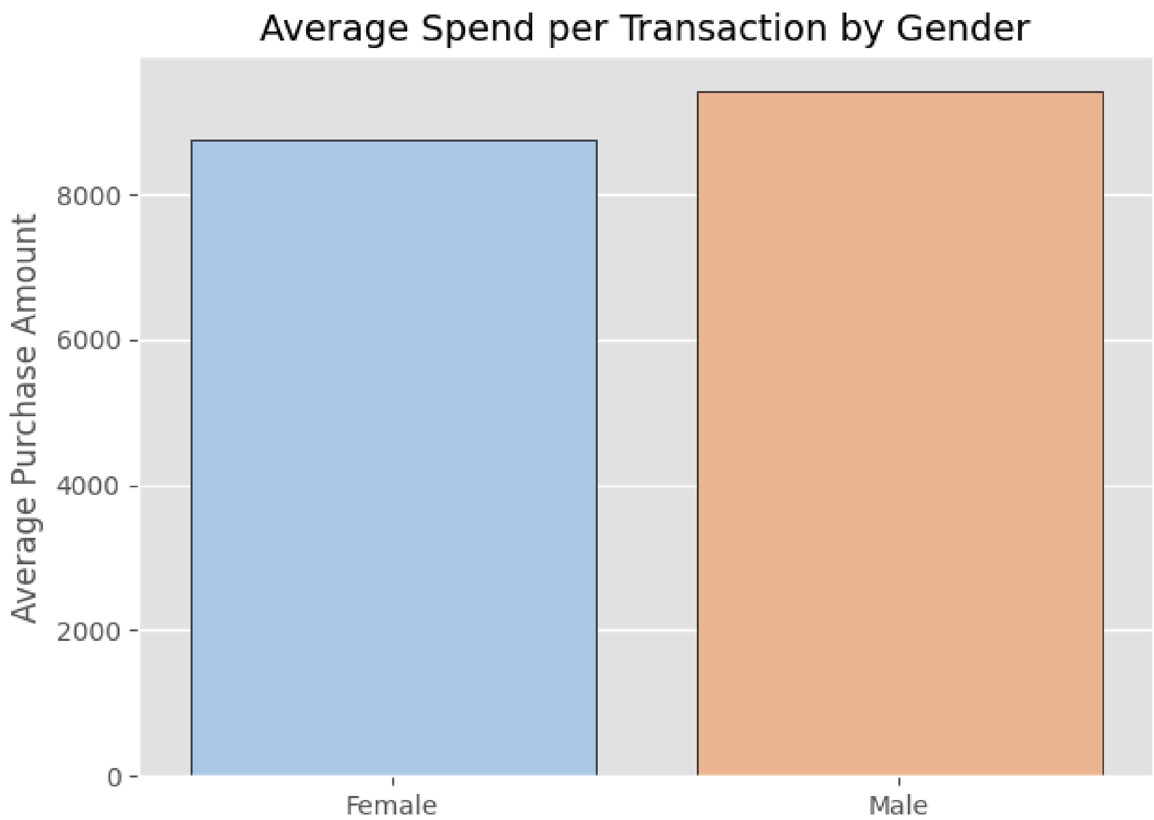
In [39]:
```
sample_wmean_30000 = [np.mean(dfwomen.sample(30000)) for i in
range(1000)]
np.mean(sample_wmean_30000)
```

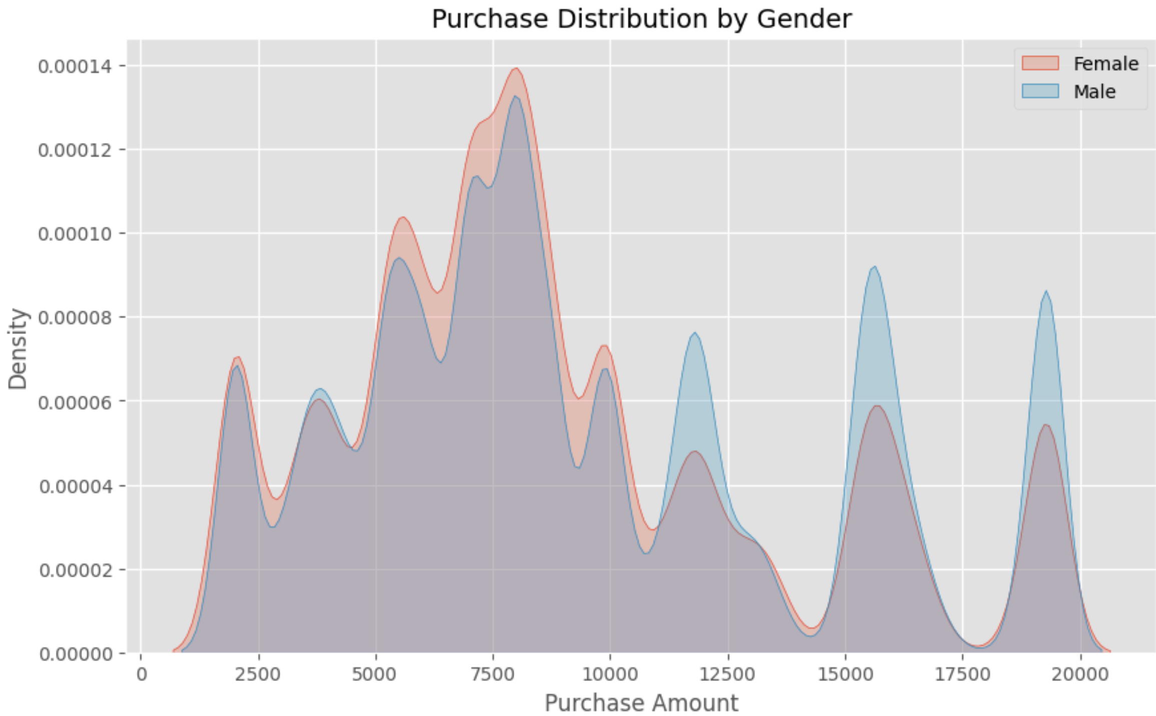
Out[39]: np.float64(8736.356016733333)

In [40]:
```
fig, axis = plt.subplots(nrows=1, ncols=3, figsize=(18, 5))
axis[0].hist(sample_wmean_300)
axis[1].hist(sample_wmean_3000)
axis[2].hist(sample_wmean_30000)
axis[0].set_title("Female Means Distribution, Sample size: 300")
axis[1].set_title("Female Means Distribution, Sample size: 3000")
axis[2].set_title("Female Means Distribution, Sample size: 30000")
plt.show()
```



In [61]:
```
# Visualization: Average bar chart
plt.figure(figsize=(7,5))
sns.barplot(x=["Female","Male"], y=[female_avg, male_avg], palette="pastel", edg
plt.title("Average Spend per Transaction by Gender", fontsize=14)
plt.ylabel("Average Purchase Amount")
plt.show()

# Visualization: Purchase distribution
plt.figure(figsize=(10,6))
sns.kdeplot(female_df["Purchase"], label="Female", shade=True)
sns.kdeplot(male_df["Purchase"], label="Male", shade=True)
plt.title("Purchase Distribution by Gender", fontsize=14)
plt.xlabel("Purchase Amount")
plt.ylabel("Density")
plt.legend()
plt.show()

# Summary Report
print("\n=== SUMMARY ===")
print(f"Sample Size (Females): {len(female_df)}")
print(f"Sample Size (Males)  : {len(male_df)}")
print(f"Mean Female Purchase : {female_avg:.2f}")
print(f"Mean Male Purchase   : {male_avg:.2f}")
print(f"95% CI Female        : {female_ci}")
print(f"95% CI Male          : {male_ci}")
```





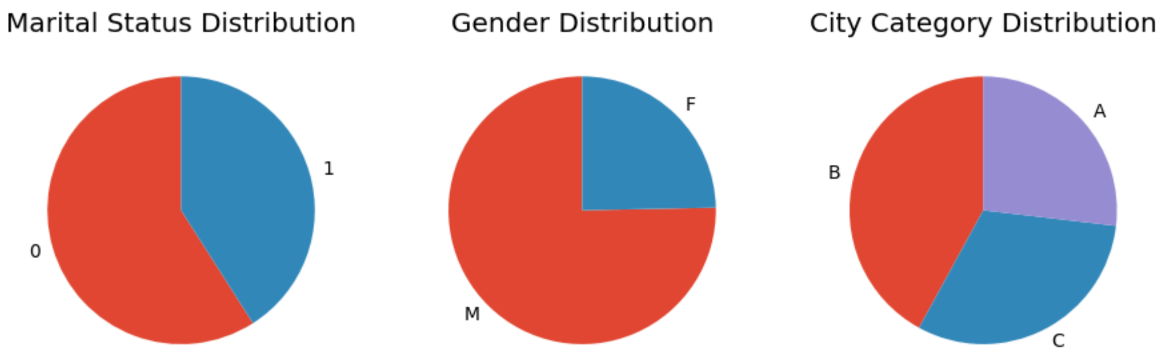

```
=== SUMMARY ===
Sample Size (Females): 135809
Sample Size (Males)  : 414259
Mean Female Purchase : 8736.54
Mean Male Purchase   : 9427.24
95% CI Female        : (np.float64(8712.091286628549), np.float64(8760.9892455894
93))
95% CI Male          : (np.float64(9412.240567188413), np.float64(9442.241425960
8))
```

INSIGHTS:

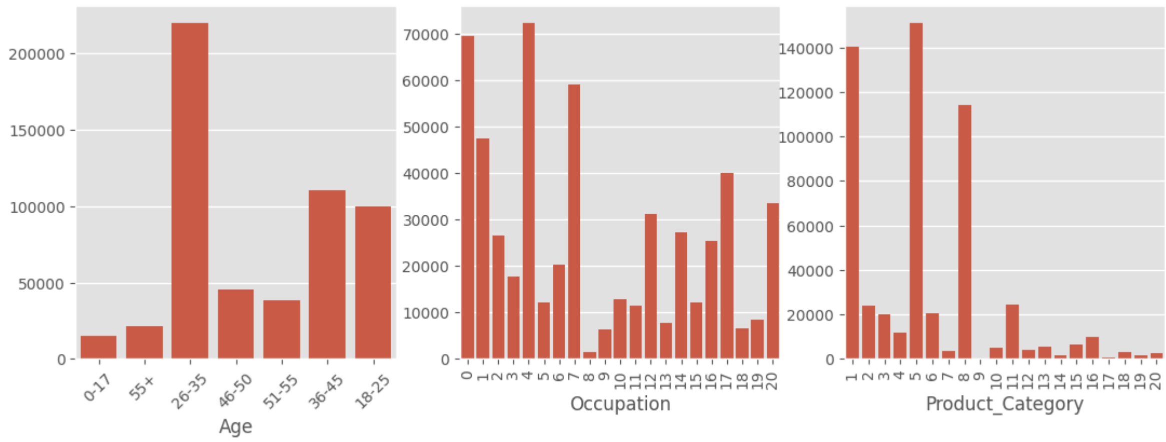
1. Male Population Purchase mean is 9427 and Female Population Purchase mean is 8736
2. Male Population 95% Confidence Interval:(9412.239625076156, 9442.240374923844) Female Population 95% Confidence Interval:(8712.09131613775, 8760.988683862251)
3. Male Confidence Interval with different Sample sizes: • Sample size of 300 - (9412.239625076156, 9442.240374923844) • Sample size of 3000 - (9250.970382155128, 9603.509617844871) • Sample size of 30000 - (9371.498652532275, 9482.981347467725) It is Observed that as the Sample size increases width of the Confidence Interval decreases
4. Confidence Intervals in above case are overlapping. It is clearly inferred by comparing lower bound of one interval with upper bound on other interval
5. Variation of Male Distribution means w.r.t Sample sizes: Sample size - 300: 9449 Sample size - 3000: 9428 Sample size- 30000: 9427 As the sample size increases the sample distribution mean comes closer to population mean.
6. As the sample size increases the sample dsitribution plot becomes narrower as shown above
7. Female Confidence Interval with different Sample sizes: • Sample size of 300 - (8216.353432802387, 9256.726567197615) • Sample size of 3000 - (8572.042563943132, 8901.03743605687) • Sample size of 30000 - (8684.52134328024, 8788.558656719762) It is Observed that as the Sample size increases width of the Confidence Interval decreases
8. Confidence Intervals in above case are overlapping. It is clearly inferred by comparing lower bound of one interval with upper bound on other interval
9. Variation of Female Distribution means w.r.t Sample sizes: Sample size - 300: 8748 Sample size - 3000: 8735 Sample size- 30000: 8736 As the sample size increases the sample distribution mean comes closer to population mean.
10. As the sample size increases the sample dsitribution plot becomes narrower as shown above
11. Female Population CI is wider than Male's which signifies lower precision in the estimate and greater uncertainity about the true population parameter

Univariate Analysis / Marginal Probability

In [42]:
```
fig, ax = plt.subplots(1, 3, figsize = (11, 4))
ax[0].pie(df['Marital_Status'].value_counts(),labels=df['Marital_Status'].value_
ax[0].set_title("Marital Status Distribution")
ax[1].pie(df['Gender'].value_counts(),labels=df['Gender'].value_counts().index,s
ax[1].set_title("Gender Distribution")
ax[2].pie(df['City_Category'].value_counts(),labels=df['City_Category'].value_cc
ax[2].set_title("City Category Distribution")
plt.show()
```



In [43]:
```
fig, axis = plt.subplots(nrows=1, ncols=3, figsize=(13, 3))
fig.subplots_adjust(top=1.2)
axis = axis.flatten()
sns.countplot(data=df, x="Age", ax=axis[0])
axis[0].set_ylabel('')
axis[0].tick_params(axis='x', rotation=45)
sns.countplot(data=df, x="Occupation", ax=axis[1])
axis[1].set_ylabel('')
axis[1].tick_params(axis='x', rotation=90)
sns.countplot(data=df, x="Product_Category", ax=axis[2])
axis[2].set_ylabel('')
axis[2].tick_params(axis='x', rotation=90)
plt.show()
```



In [44]:
```
df['Gender'].value_counts(normalize=True)
```

Out[44]:

Gender	proportion
M	0.753105
F	0.246895

dtype: float64

In [45]:
```
df['Age'].value_counts(normalize=True)
```

Out[45]:

Age	proportion
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

dtype: float64

In [46]:
```
df['Occupation'].value_counts(normalize=True)
```

Out[46]:

	proportion
Occupation	
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

dtype: float64

In [47]:
```
df['City_Category'].value_counts(normalize=True)
```

Out[47]:

	proportion
City_Category	
B	0.420263
C	0.311189
A	0.268549

dtype: float64

In [48]:
```
df['Stay_In_Current_City_Years'].value_counts(normalize=True)
```

Out[48]:

	proportion
Stay_In_Current_City_Years	
1	0.352358
2	0.185137
3	0.173224
4+	0.154028
0	0.135252

dtype: float64

In [49]:
```
df['Marital_Status'].value_counts(normalize=True)
```

Out[49]:

	proportion
Marital_Status	
0	0.590347
1	0.409653

dtype: float64

In [50]:
```
df['Product_Category'].value_counts(normalize=True)
```

Out[50]:

Product_Category	proportion
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

dtype: float64

INSIGHTS:

1. Customers with marital status 0 at 59% are higher than 1
2. 26-35 is the maximum buying age group with 40% share
3. Customers with occupation 4 are the maximum buyers (13%) followed by 0 and 7
4. Highest sold product category is 5 (27%) followed by 1 and 8
5. Males are clearly dominating the data with 75% of the purchases
6. Customers belonging to City catgeory B are at the top with 42%
7. Most of the customers are staying in the city for 1 year with 35%

Multivariate Analysis / Conditional Probability

In [52]:
```
pd.crosstab(df['Product_Category'], df['Age'], normalize=True)
```

Out[52]:

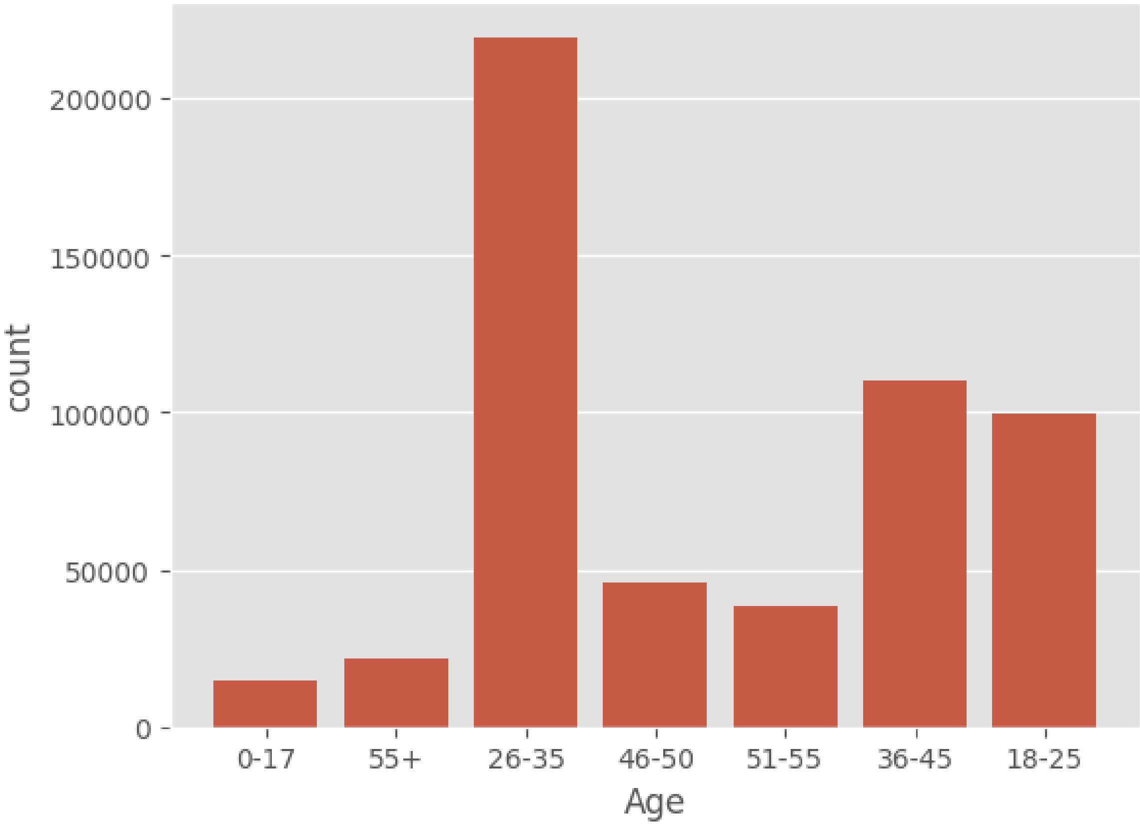
Age Product_Category	0-17	18-25	26-35	36-45	46-50	51-55	55+
1	0.006517	0.049016	0.105894	0.050263	0.019041	0.016451	0.008019
2	0.001463	0.008050	0.016231	0.008930	0.003827	0.003238	0.001645
3	0.002182	0.008563	0.013929	0.007006	0.002502	0.001680	0.000885
4	0.001378	0.004478	0.007621	0.004279	0.001800	0.001233	0.000578
5	0.007872	0.051852	0.111755	0.053406	0.021763	0.017985	0.009757
6	0.000725	0.006816	0.015425	0.007088	0.002949	0.002636	0.001567
7	0.000096	0.000874	0.003001	0.001471	0.000594	0.000484	0.000244
8	0.004105	0.032561	0.080456	0.042351	0.019372	0.016980	0.011286
9	0.000029	0.000115	0.000280	0.000195	0.000060	0.000053	0.000015
10	0.000202	0.001096	0.003249	0.002245	0.000945	0.000944	0.000636
11	0.001345	0.008357	0.017951	0.009004	0.003825	0.002651	0.001020
12	0.000227	0.000798	0.001992	0.001807	0.000945	0.000787	0.000618
13	0.000204	0.001374	0.003810	0.002272	0.001002	0.000878	0.000547
14	0.000071	0.000418	0.001025	0.000567	0.000271	0.000280	0.000136
15	0.000291	0.001862	0.004312	0.002536	0.001094	0.000924	0.000416
16	0.000416	0.002905	0.007486	0.003554	0.001598	0.001222	0.000685
17	0.000011	0.000075	0.000231	0.000245	0.000173	0.000195	0.000122
18	0.000049	0.000616	0.001894	0.001276	0.000638	0.000769	0.000438
19	0.000107	0.000500	0.001024	0.000582	0.000271	0.000244	0.000187
20	0.000164	0.000853	0.001633	0.000920	0.000413	0.000364	0.000291

6.Perform the same activity for Married vs Unmarried and Age For Age, you can try bins based on life stages: 0-17, 18-25, 26-35, 36-50, 51+ years.

Products Preferred By Different Age Groups

In [11]:
```
sns.countplot(data = df, x = 'Age')
plt.plot()   # displaying the plot
```

Out[11]: []



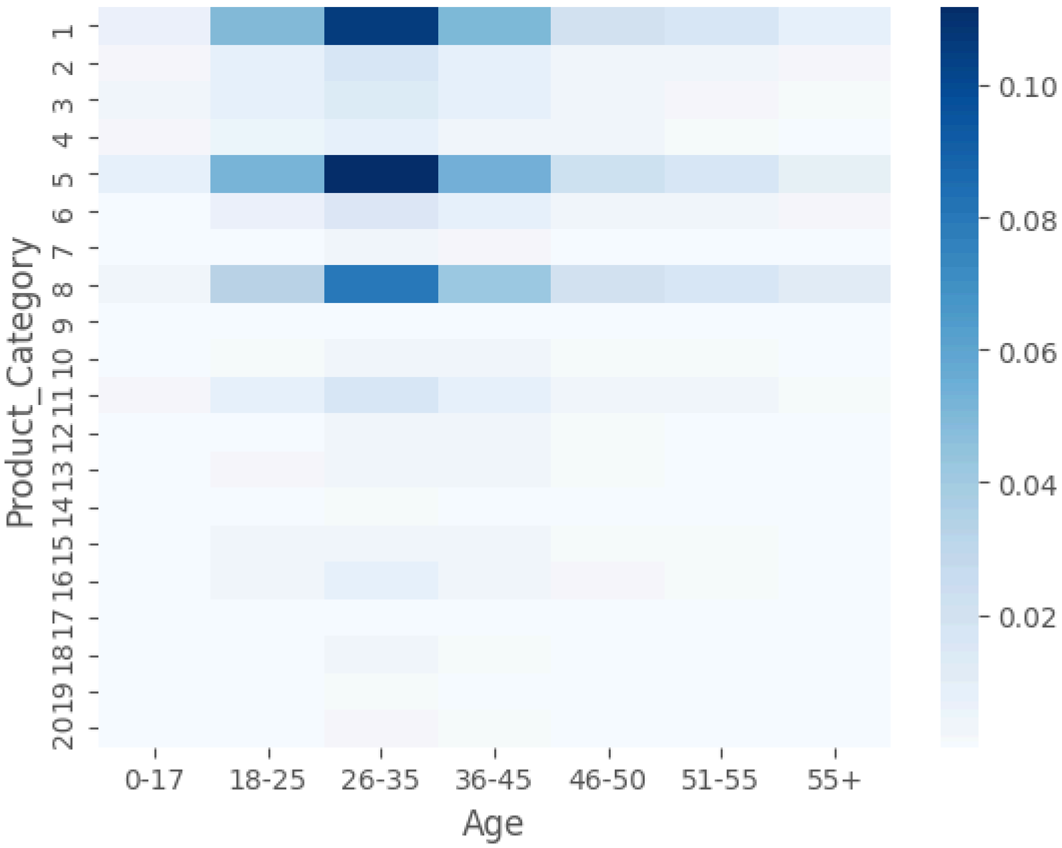
In [6]:
```
pd.crosstab(df['Product_Category'], df['Age'], normalize=True)
```

Out[6]:

Age Product_Category	0-17	18-25	26-35	36-45	46-50	51-55	55+
1	0.006517	0.049016	0.105894	0.050263	0.019041	0.016451	0.008019
2	0.001463	0.008050	0.016231	0.008930	0.003827	0.003238	0.001645
3	0.002182	0.008563	0.013929	0.007006	0.002502	0.001680	0.000885
4	0.001378	0.004478	0.007621	0.004279	0.001800	0.001233	0.000578
5	0.007872	0.051852	0.111755	0.053406	0.021763	0.017985	0.009757
6	0.000725	0.006816	0.015425	0.007088	0.002949	0.002636	0.001567
7	0.000096	0.000874	0.003001	0.001471	0.000594	0.000484	0.000244
8	0.004105	0.032561	0.080456	0.042351	0.019372	0.016980	0.011286
9	0.000029	0.000115	0.000280	0.000195	0.000060	0.000053	0.000015
10	0.000202	0.001096	0.003249	0.002245	0.000945	0.000944	0.000636
11	0.001345	0.008357	0.017951	0.009004	0.003825	0.002651	0.001020
12	0.000227	0.000798	0.001992	0.001807	0.000945	0.000787	0.000618
13	0.000204	0.001374	0.003810	0.002272	0.001002	0.000878	0.000547
14	0.000071	0.000418	0.001025	0.000567	0.000271	0.000280	0.000136
15	0.000291	0.001862	0.004312	0.002536	0.001094	0.000924	0.000416
16	0.000416	0.002905	0.007486	0.003554	0.001598	0.001222	0.000685
17	0.000011	0.000075	0.000231	0.000245	0.000173	0.000195	0.000122
18	0.000049	0.000616	0.001894	0.001276	0.000638	0.000769	0.000438
19	0.000107	0.000500	0.001024	0.000582	0.000271	0.000244	0.000187
20	0.000164	0.000853	0.001633	0.000920	0.000413	0.000364	0.000291

In [7]:

```
sns.heatmap(pd.crosstab(df['Product_Category'], df['Age'],
normalize=True),
 cmap='Blues')
plt.show()
```

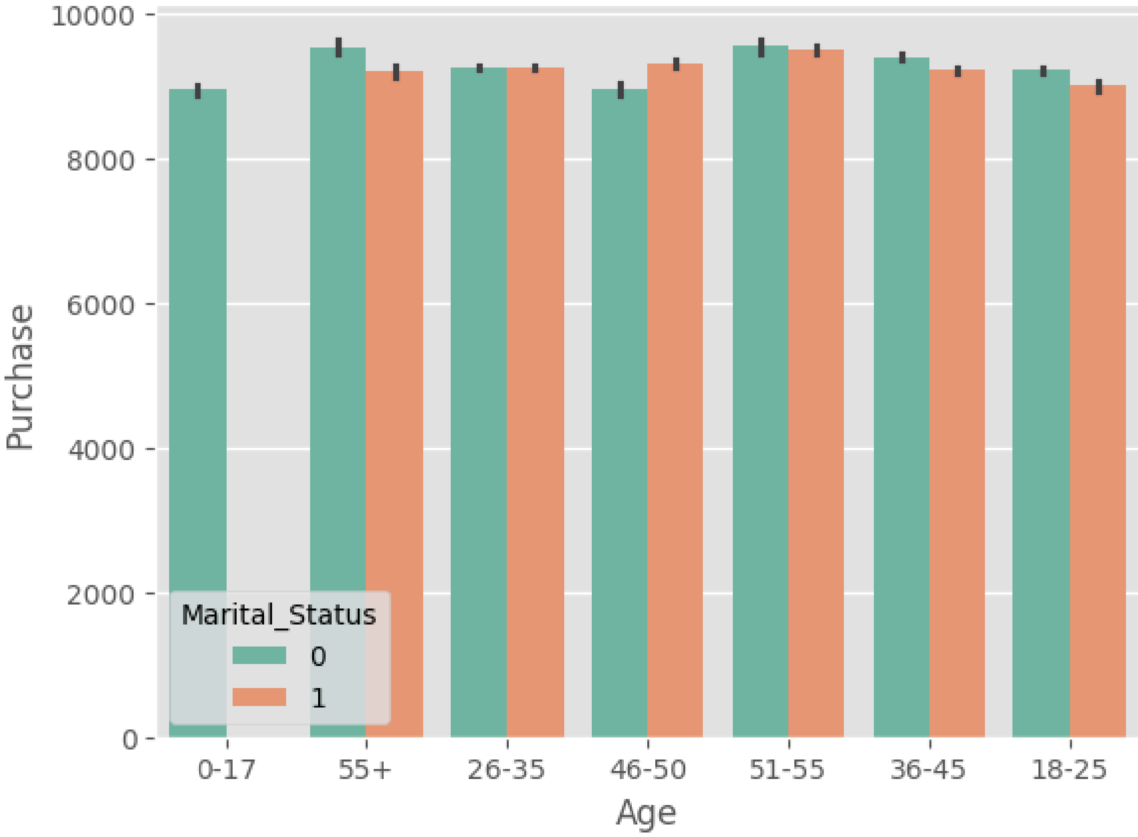


INSIGHTS:

1. Age group 26-35 buys maximum products from category 5 followed by 1 and 8
2. Age group 36-45 buys most of the products from category 5 followed by 1 and 8
3. Age group 18-25 shows similar buying pattern as age group 36-45

Relationship Between Age , Marital Status and Purchases

In [12]:
```
sns.barplot(data=df,x='Age',y='Purchase',hue='Marital_Status',palette
='Set2')
#plt.legend(bbox_to_anchor=(1.05, 0.5), loc='center left')
plt.show()
```



INSIGHTS:

1. All the age groups show similar buying behaviour for males and females.
2. Age group 0-17 have only 0 bar which denotes singles for obvious reasons

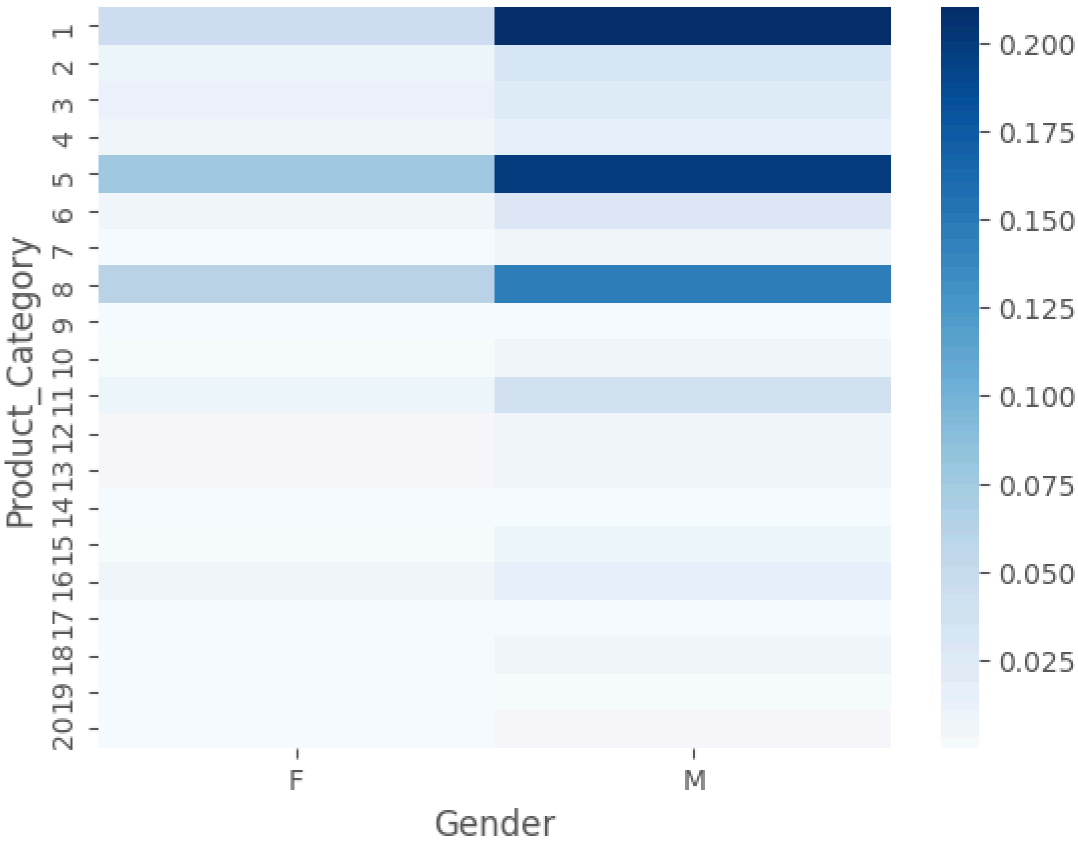
Product Categories preferred by different Genders

In [13]:
```
pd.crosstab(df['Product_Category'], df['Gender'], normalize=True)
```

Out[13]:

Gender	F	M
Product_Category		
1	0.045142	0.210059
2	0.010286	0.033098
3	0.010919	0.025828
4	0.006616	0.014751
5	0.076283	0.198106
6	0.008288	0.028918
7	0.001714	0.005050
8	0.061007	0.146104
9	0.000127	0.000618
10	0.002112	0.007205
11	0.008615	0.035537
12	0.002785	0.004390
13	0.002658	0.007430
14	0.001133	0.001636
15	0.001902	0.009533
16	0.004367	0.013500
17	0.000113	0.000938
18	0.000694	0.004987
19	0.000820	0.002094
20	0.001314	0.003321

In [14]:
```
sns.heatmap(pd.crosstab(df['Product_Category'], df['Gender'], normalize=True),
 cmap='Blues')
plt.show()
```



INSIGHTS:

1. Males are mostly purchasing product category 1 followed by 5 and 8
2. Females show preference to product category 5 followed by 8 and 1

Marital Status Effect on Purchases - 95% Confidence Interval / Bootstrapping

In [110…

```
dfs=df[df['Marital_Status']==0]['Purchase']
dfs
```

Out[110…

	Purchase
0	8370
1	15200
2	1984
3	1984
4	7969
...	...
550056	1984
550059	1984
550062	1984
550064	1984
550066	1984

324731 rows × 1 columns

dtype: int64

In [111…

```
s_mean=round(dfs.mean(),2)
s_mean
```

Out[111…

```
np.float64(9258.82)
```

In [112…

```
s_std=round(dfs.std(),2)
s_std
```

Out[112…

```
4864.58
```

In [113…

```
sn=len(dfs)
sn
```

Out[113…

```
324731
```

In [114…

```
norm.interval(confidence=0.95, loc=s_mean, scale=s_std/np.sqrt(sn))
```

Out[114…

```
(np.float64(9242.08862758751), np.float64(9275.55137241249))
```

In [115…

```
bootstrapped_mean_300 = []
for i in range(1000):
 bootstrapped_sample_300 = np.random.choice(dfs, size=300)
 bootstrapped_mean = np.mean(bootstrapped_sample_300)
 bootstrapped_mean_300.append(bootstrapped_mean)
x1 = np.percentile(bootstrapped_mean_300, 2.5)
x2 = np.percentile(bootstrapped_mean_300, 97.5)
x1, x2
```

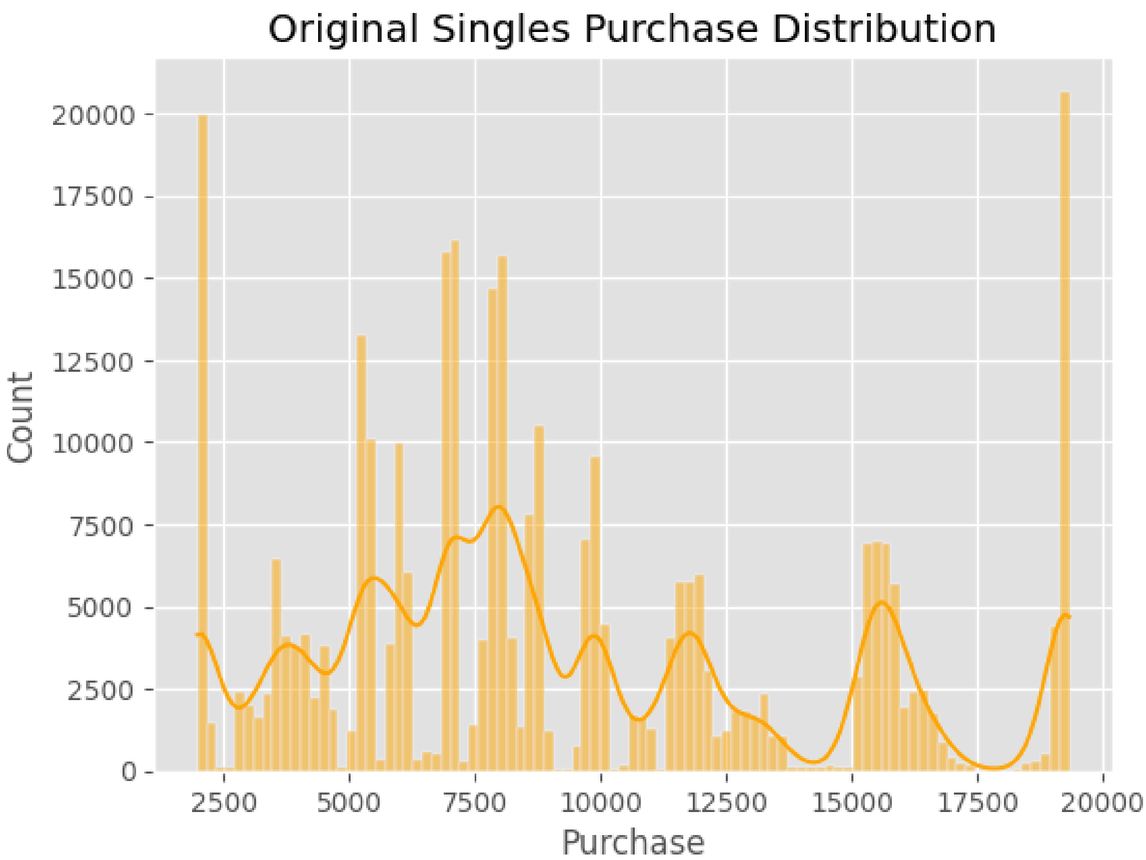
Out[115…

```
(np.float64(8689.06275), np.float64(9848.14825))
```

In [116…
```
bootstrapped_mean_3000 = []
for i in range(1000):
 bootstrapped_sample_3000 = np.random.choice(dfs, size=3000)
 bootstrapped_mean = np.mean(bootstrapped_sample_3000)
 bootstrapped_mean_3000.append(bootstrapped_mean)
a1 = np.percentile(bootstrapped_mean_3000, 2.5)
a2 = np.percentile(bootstrapped_mean_3000, 97.5)
a1, a2
```
Out[116…
```
(np.float64(9065.549058333332), np.float64(9433.374916666666))
```

In [117…
```
bootstrapped_mean_30000 = []
for i in range(1000):
 bootstrapped_sample_30000 = np.random.choice(dfs, size=30000)
 bootstrapped_mean = np.mean(bootstrapped_sample_30000)
 bootstrapped_mean_30000.append(bootstrapped_mean)
b1 = np.percentile(bootstrapped_mean_30000, 2.5)
b2 = np.percentile(bootstrapped_mean_30000, 97.5)
b1, b2
```
Out[117…
```
(np.float64(9203.9450575), np.float64(9312.599580833334))
```

In [119…
```
sns.histplot(data=dfs,color='orange',kde=True).set_title("Original Singles Purch
plt.show()
```



In [122…
```
np.mean(bootstrapped_mean_300)
```
Out[122…
```
np.float64(9261.804323333334)
```

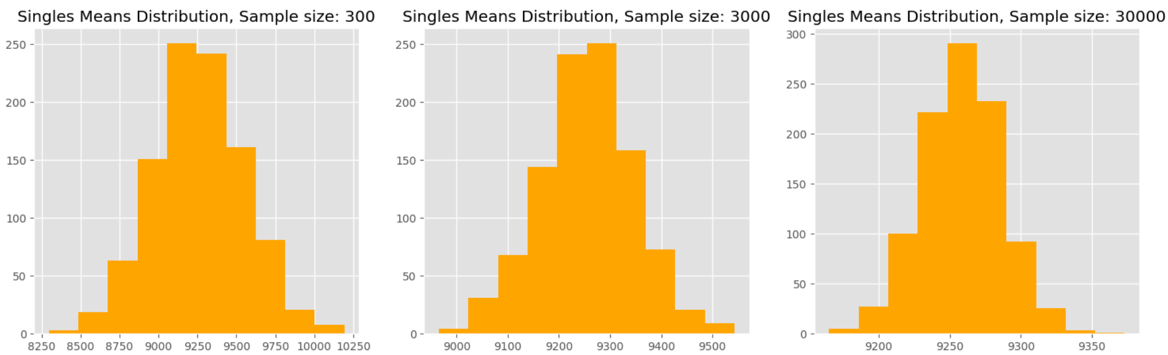
In [123…
```
np.mean(bootstrapped_mean_3000)
```

Out[123…
```
np.float64(9256.422206000001)
```

In [124…
```
np.mean(bootstrapped_mean_30000)
```

Out[124…
```
np.float64(9258.741952133334)
```

In [125…
```
fig, axis = plt.subplots(nrows=1, ncols=3, figsize=(18, 5))
axis[0].hist(bootstrapped_mean_300,color='orange')
axis[1].hist(bootstrapped_mean_3000,color='orange')
axis[2].hist(bootstrapped_mean_30000,color='orange')
axis[0].set_title("Singles Means Distribution, Sample size: 300")
axis[1].set_title("Singles Means Distribution, Sample size: 3000")
axis[2].set_title("Singles Means Distribution, Sample size: 30000")
plt.show()
```



Married Data Confidence Interval & Distribution of Means

In [126…
```
dfm=df[df['Marital_Status']==1]['Purchase']
dfm
```

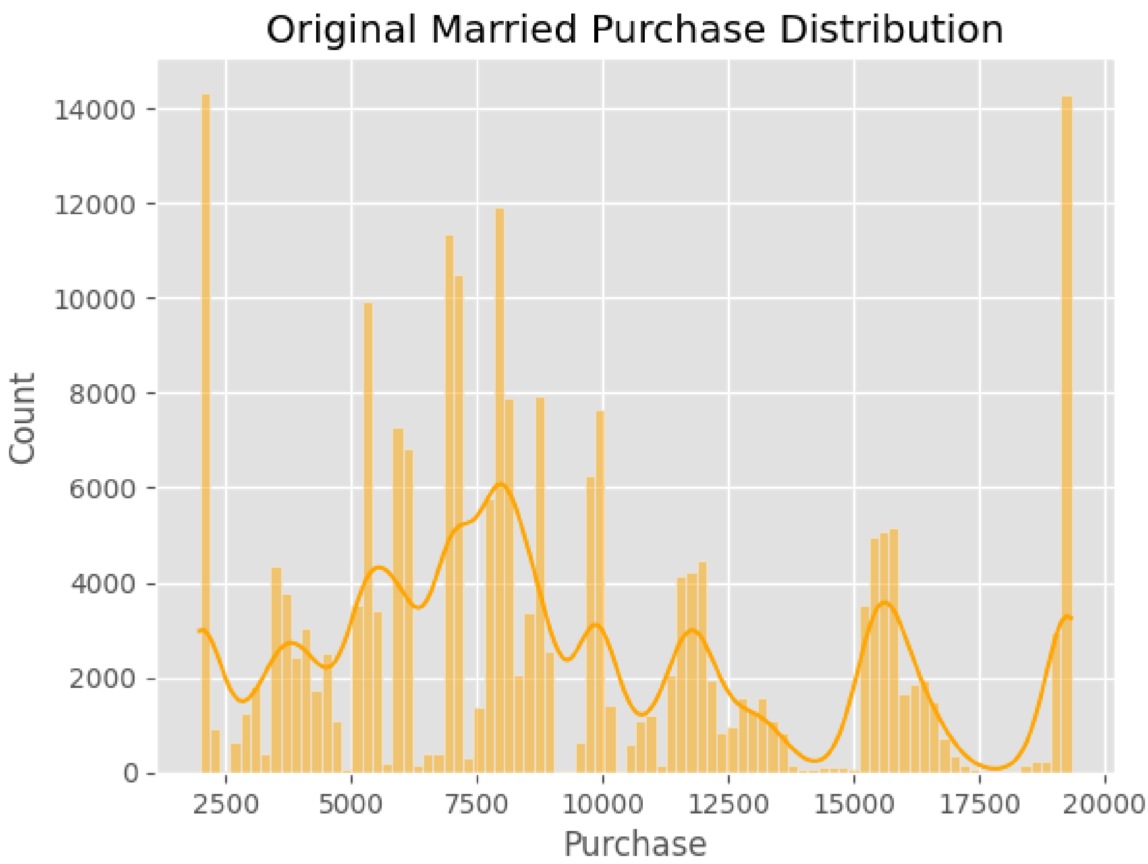
Out[126…

	Purchase
6	19215
7	15854
8	15686
9	7871
10	5254
...	...
550060	1984
550061	1984
550063	1984
550065	1984
550067	1984

225337 rows × 1 columns

dtype: int64

In [127…
```
m_mean=round(dfm.mean(),2)
m_mean
```
Out[127…
```
np.float64(9253.67)
```
In [128…
```
m_std=round(dfm.std(),2)
m_std
```
Out[128…
```
4843.49
```
In [129…
```
mn=len(dfm)
mn
```
Out[129…
```
225337
```
In [130…
```
norm.interval(confidence=0.95, loc=m_mean, scale=m_std/np.sqrt(mn))
```
Out[130…
```
(np.float64(9233.671830529833), np.float64(9273.668169470167))
```
In [131…
```
bootstrapped_m_mean_300 = []
for i in range(1000):
 bootstrapped_m_sample_300 = np.random.choice(dfs, size=300)
 bootstrapped_mean = np.mean(bootstrapped_m_sample_300)
 bootstrapped_m_mean_300.append(bootstrapped_mean)
y1 = np.percentile(bootstrapped_m_mean_300, 2.5)
y2 = np.percentile(bootstrapped_m_mean_300, 97.5)
y1, y2
```
Out[131…
```
(np.float64(8714.869583333333), np.float64(9853.2115))
```
In [132…
```
bootstrapped_m_mean_3000 = []
for i in range(1000):
 bootstrapped_m_sample_3000 = np.random.choice(dfs, size=3000)
 bootstrapped_mean = np.mean(bootstrapped_m_sample_3000)
 bootstrapped_m_mean_3000.append(bootstrapped_mean)
c1 = np.percentile(bootstrapped_m_mean_3000, 2.5)
c2 = np.percentile(bootstrapped_m_mean_3000, 97.5)
c1, c2
```
Out[132…
```
(np.float64(9083.9437), np.float64(9432.156016666666))
```
In [133…
```
bootstrapped_m_mean_30000 = []
for i in range(1000):
 bootstrapped_m_sample_30000 = np.random.choice(dfs, size=30000)
 bootstrapped_mean = np.mean(bootstrapped_m_sample_30000)
 bootstrapped_m_mean_30000.append(bootstrapped_mean)
d1 = np.percentile(bootstrapped_m_mean_30000, 2.5)
d2 = np.percentile(bootstrapped_m_mean_30000, 97.5)
d1, d2
```
Out[133…
```
(np.float64(9204.1013575), np.float64(9314.777866666667))
```
In [135…
```
sns.histplot(data=dfm,color='orange',kde=True).set_title("Original Married Purch
plt.show()
```

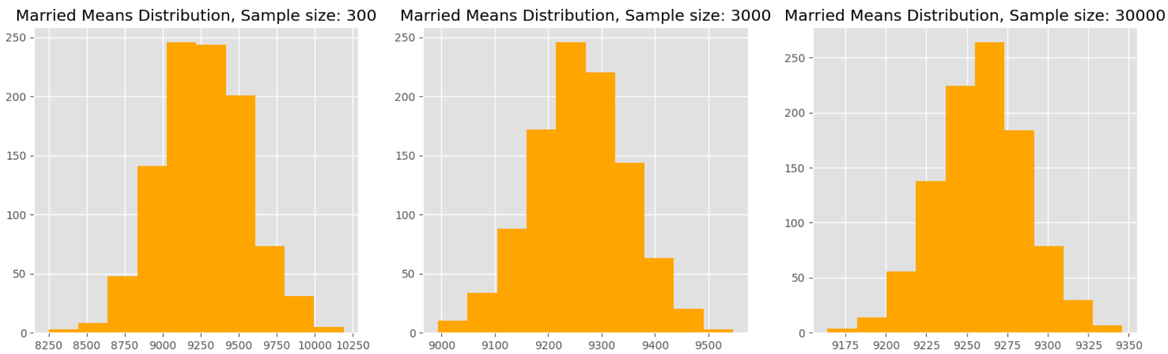


In [137…
```
np.mean(bootstrapped_m_mean_300)
```
Out[137… np.float64(9264.49434)

In [136…
```
np.mean(bootstrapped_m_mean_3000)
```
Out[136… np.float64(9258.780232)

In [138…
```
np.mean(bootstrapped_m_mean_30000)
```
Out[138… np.float64(9258.9378957)

In [139…
```
fig, axis = plt.subplots(nrows=1, ncols=3, figsize=(18, 5))
axis[0].hist(bootstrapped_m_mean_300,color='orange')
axis[1].hist(bootstrapped_m_mean_3000,color='orange')
axis[2].hist(bootstrapped_m_mean_30000,color='orange')
axis[0].set_title("Married Means Distribution, Sample size: 300")
axis[1].set_title("Married Means Distribution, Sample size: 3000")
axis[2].set_title("Married Means Distribution, Sample size: 30000")
plt.show()
```



INSIGHTS:

1. Singles Population Purchase mean is 9258 and Married Population Purchase mean is 9253
2. Singles Population Confidence 95% Interval:(9242.08862758751, 9275.55137241249) Married Population Confidence 95% Interval:(9233.671830529833, 9273.668169470167)
3. Singles Confidence Interval with different Sample sizes: • Sample size of 300 - (8691.985083333333, 9805.97075) • Sample size of 3000 - (9090.470683333333, 9423.650783333334) • Sample size of 30000 - (9202.758979166667, 9309.495452500001) It is Observed that as the Sample size increases width of the Confidence Interval decreases
4. Confidence Intervals in above case are overlapping. It is clearly inferred by comparing lower bound of one interval with upper bound on other interval
5. Variation of Singles Distribution means w.r.t Sample sizes: Sample size - 300: 9253 Sample size - 3000: 9255 Sample size- 30000: 9257 As the sample size increases, the sample distribution mean comes closer to population mean.
6. As the sample size increases, the sample dsitribution plot becomes narrower as shown above
7. Married Confidence Interval with different Sample sizes: • Sample size of 300 - (8740.33475, 9805.124333333333) • Sample size of 3000 - (9087.528275, 9427.935141666667) • Sample size of 30000 - (9202.58072, 9315.308989166668) It is Observed that as the Sample size increases width of the Confidence Interval decreases
8. Confidence Intervals in above case are overlapping. It is clearly inferred by comparing lower bound of one interval with upper bound on other interval
9. Variation of Married Distribution means w.r.t Sample sizes: Sample size - 300: 9265 Sample size - 3000: 9254 Sample size- 30000: 9258 As the sample size increases the sample distribution mean comes closer to population mean.
10. As the sample size increases the sample dsitribution plot becomes narrower as shown above
11. Married Population CI is slightly wider than Single's which signifies lower precision in the estimate and greater uncertainity about the true population parameter

7. Give recommendations and Action Items to Walmart.

Actions

Are women spending more money per transaction than men? Why or Why not? No,Women on an average spending 8736 which is less w.r.t Men who are spending 9427 on an average The data shows that 75% of the orders are purchased by Men and 25% by

Women. Possible reasons can be that Walmart have got more products which male dominating or Men at that :location have more purchasing power than Women. Confidence intervals and distribution of the mean of the expenses by female and male customers Male Population 95% Confidence Interval:(9412.23, 9442.24) Female Population 95% Confidence Interval:(8712.09, 8760.98) Distribution of means in case of various samples sizes alongwith insights is shared in detail above Are confidence intervals of average male and female spending overlapping? How can Walmart leverage this conclusion to make changes or improvements? No, CI of average male and female spending is not overlapping. The average spending of men per transaction is 9427 and for women its 8736. This information can be leveraged the following way:

1. We can look to increase this amount individually for men and women by introducing relevant products at attractive prices
2. Since women average purchase amount is less, we can introduce more women centric products to increase the average spending
3. The average spending by men and women give important information on the purchasing power in that location. Depending on this information we can introduce products at a target price which will definitely help increase revenue Results when the same activity is performed for Married vs Unmarried Singles Population Purchase mean is 9258 and Married Population Purchase mean is 9253 Singles Population 95% Confidence Interval:(9242.08, 9275.55) Married Population 95% Confidence Interval:(9233.67, 9273.66) CI are overlapping in this case and average purchase per transaction is almost same which infers similarity in buying behaviour of singles and married Results when the same activity is performed for Age
4. Age Group 0-17: (8861.85, 9019.44) Avg. Purchase- 8940
5. Age Group 18-25:(9138.65, 9199.36) Avg. Purchase- 9169
6. Age Group 26-35:(9223.78, 9264.08) Avg. Purchase- 9243
7. Age Group 36-45:(9294.27, 9351.56) Avg. Purchase- 9322
8. Age Group 46-50:(9160.33, 9248.08) Avg. Purchase- 9204
9. Age Group 51-55:(9466.18, 9563.54) Avg. Purchase- 9514
10. Age Group 55+: (9263.91, 9391.68) Avg. Purchase- 9327

Recommendations

1. Currently there are unique 5891 customers. We need to look into our marketing efforts to increase this customer base w.r.t population of that location and potential in that region
2. 75% of the orders are coming from male population and 25% from female. Males mostly buying product category 1 followed by 5 and 8. This figure shows the potential to bring in more variety of male centric products so as to retain our customer base. Definitely, improvement needed for female category who is currently buying mostly product category 5 followed by 8 and 1, in terms of identification of right products and marketing strategy for them so that this share can also increase
3. 40% of the orders are from the age group 26-35 who is mostly buying product category 5, which directly indicates how critical it is to keep supply of relevant

products for this group at attractive pricing so that it continues to be the revenue generator for the company

4. Customers with occupation no. 4 are 13% closely followed by 0 and 7. It is an indicator of what kind of corporate offers and product range to have to increase revenue from this section of customer
5. We need to devise a marketing strategy targeting customers from each category of city. Current data shows City category B is leading with 42% followed by C and A. We need to understand the demography of that region and plan marketing efforts accordingly to increase revenue
6. Singles are buying more with 59% followed by married with 41%. We can introduce more products targeting Singles in a specific price w.r.t purchasing power
7. Top of the product category is 5 followed by 1. It is an indicator of the demography of that region and the likes of that region. For ex. If category including books are selling most, we can introduce more products near that category like stationary products which definitely find a pull in that market and will help increase revenue as well
8. The average spending of men per transaction is 9427 and for women its 8736. We can look to increase this amount individually for men and women by introducing relevant products at attractive prices