Import the dataset and do usual data analysis steps like checking the structure & characteristics of the dataset.

In [33]:
```
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
from scipy.stats import norm
```

In [29]:
```
df = pd.read_excel(r"C:\Users\Hp\Downloads\walmart.xlsx")
```

In [35]:
```
df.head(20)
```

Out[35]:

	User_ID	Product_ID	Gender	Age	Occupation	City_Category	Stay_In_Current_City_Year
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	
5	1000003	P00193542	M	26-35	15	A	
6	1000004	P00184942	M	46-50	7	B	
7	1000004	P00346142	M	46-50	7	B	
8	1000004	P0097242	M	46-50	7	B	
9	1000005	P00274942	M	26-35	20	A	
10	1000005	P00251242	M	26-35	20	A	
11	1000005	P00014542	M	26-35	20	A	
12	1000005	P00031342	M	26-35	20	A	
13	1000005	P00145042	M	26-35	20	A	
14	1000006	P00231342	F	51-55	9	A	
15	1000006	P00190242	F	51-55	9	A	
16	1000006	P0096642	F	51-55	9	A	
17	1000006	P00058442	F	51-55	9	A	
18	1000007	P00036842	M	36-45	1	B	
19	1000008	P00249542	M	26-35	12	C	

In [31]:
```
df.describe()
```

Out[31]:

	User_ID	Occupation	Stay_In_Current_City_Years	Marital_Status	Product_Cate
count	5.500680e+05	550068.000000	550068.000000	550068.000000	550068.00
mean	1.003029e+06	8.076707	1.858418	0.409653	5.40
std	1.727592e+03	6.522660	1.289443	0.491770	3.93
min	1.000001e+06	0.000000	0.000000	0.000000	1.00
25%	1.001516e+06	2.000000	1.000000	0.000000	1.00
50%	1.003077e+06	7.000000	2.000000	0.000000	5.00
75%	1.004478e+06	14.000000	3.000000	1.000000	8.00
max	1.006040e+06	20.000000	4.000000	1.000000	20.00

In [34]:
```
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  int64 
 7   Marital_Status              550068 non-null  int64 
 8   Product_Category            550068 non-null  int64 
 9   Purchase                    550068 non-null  int64 
dtypes: int64(6), object(4)
memory usage: 42.0+ MB
```

```
-columns like Age, Gender,City_Category are Categorical.

-column Purchase is Continuous.

-City_Category is of nominal type (no proper order between the
categories).

-Age is of ordinal type (an order exists between the categories
```

Converting Marital_Status to categorical

In [36]:
```
# Get current data type of columns
df.dtypes
```

Out[36]:
```
User_ID                        int64
Product_ID                    object
Gender                        object
Age                           object
Occupation                     int64
City_Category                 object
Stay_In_Current_City_Years     int64
Marital_Status                 int64
Product_Category               int64
Purchase                       int64
dtype: object
```

In [37]:
```
df['Marital_Status'] = pd.Categorical(df.Marital_Status)
df.dtypes
```

Out[37]:
```
User_ID                          int64
Product_ID                      object
Gender                          object
Age                             object
Occupation                       int64
City_Category                   object
Stay_In_Current_City_Years       int64
Marital_Status                category
Product_Category                 int64
Purchase                         int64
dtype: object
```

Converting Occupation to categorical

In [49]:
```
df['Occupation'] = pd.Categorical(df.Occupation)
df.dtypes
```

Out[49]:
```
User_ID                          int64
Product_ID                      object
Gender                          object
Age                             object
Occupation                    category
City_Category                   object
Stay_In_Current_City_Years       int64
Marital_Status                category
Product_Category                 int64
Purchase                         int64
dtype: object
```

Shape of the data

In [42]:
```
#shape of data
shapeofdf = np.shape(df)
print(shapeofdf)
```

```
(550068, 10)
```

In [43]:
```
# The data set has 550068 rows and 10 columns
```

In [90]:
```
max(df.Purchase)
```

Out[90]:
```
23961
```

Value counts and unique attributes

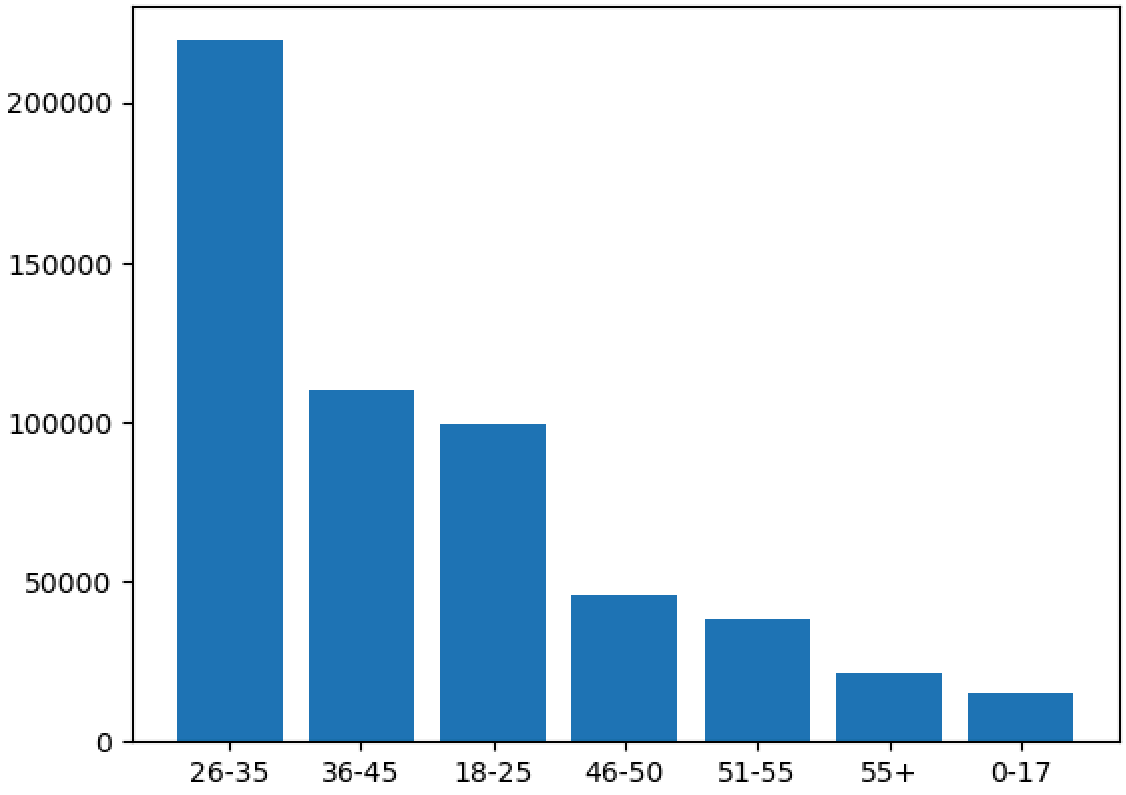
In [67]:
```
cat_Count_Age = df["Age"].value_counts()
print(cat_Count_Age)
```

```
Age
26-35    219587
36-45    110013
18-25     99660
46-50     45701
51-55     38501
55+       21504
0-17      15102
Name: count, dtype: int64
```

```
The count of purchase done by the age group 26-35 is highest
```

In [69]:
```
x_bar=cat_Count_Age.index
y_bar=cat_Count_Age
plt.bar(x_bar,y_bar)
```

Out[69]:
```
<BarContainer object of 7 artists>
```



In [51]:
```
df['Age'].value_counts(normalize=True)
```

Out[51]:
```
Age
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: proportion, dtype: float64
```

In [71]:
```
cat_count_ms = df["Marital_Status"].value_counts()
print(cat_count_ms)
```

```
Marital_Status
0    324731
1    225337
Name: count, dtype: int64
```

In [52]:
```
df['Marital_Status'].value_counts(normalize=True)
```

Out[52]:
```
Marital_Status
0    0.590347
1    0.409653
Name: proportion, dtype: float64
```

In [48]:
```
df["City_Category"].value_counts()
```

Out[48]:
```
City_Category
B    231173
C    171175
A    147720
Name: count, dtype: int64
```

In [53]:
```
df['City_Category'].value_counts(normalize=True)
```

Out[53]:
```
City_Category
B    0.420263
C    0.311189
A    0.268549
Name: proportion, dtype: float64
```

In [50]:
```
df["Occupation"].value_counts()
```

Out[50]:
```
Occupation
4     72308
0     69638
7     59133
1     47426
17    40043
20    33562
12    31179
14    27309
2     26588
16    25371
6     20355
3     17650
10    12930
5     12177
15    12165
11    11586
19     8461
13     7728
18     6622
9      6291
8      1546
Name: count, dtype: int64
```

In [54]:
```
df['Occupation'].value_counts(normalize=True)
```

Out[54]:
```
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
Name: proportion, dtype: float64
```

In [55]:
```
df.Age.unique()
```

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

In [56]:
```
df.head()
```

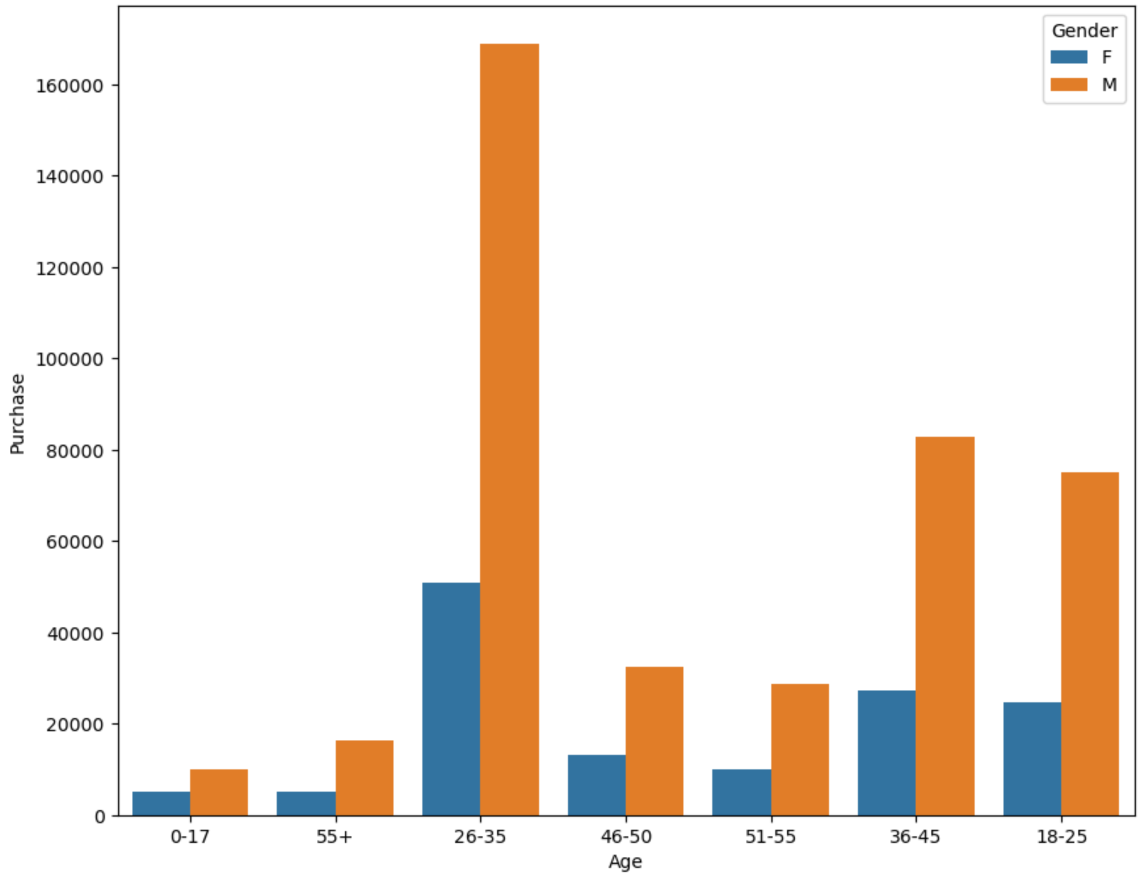
Out[56]:

	User_ID	Product_ID	Gender	Age	Occupation	City_Category	Stay_In_Current_City_Years
0	1000001	P00069042	F	0-17	10	A	2
1	1000001	P00248942	F	0-17	10	A	2
2	1000001	P00087842	F	0-17	10	A	2
3	1000001	P00085442	F	0-17	10	A	2
4	1000002	P00285442	M	55+	16	C	4

Visual Analysis - Univariate & Bivariate

In [58]:
```
plt.figure(figsize=(10,8))
sns.countplot(x='Age',hue='Gender',data=df)
plt.ylabel('Purchase')
```

Out[58]:
```
Text(0, 0.5, 'Purchase')
```

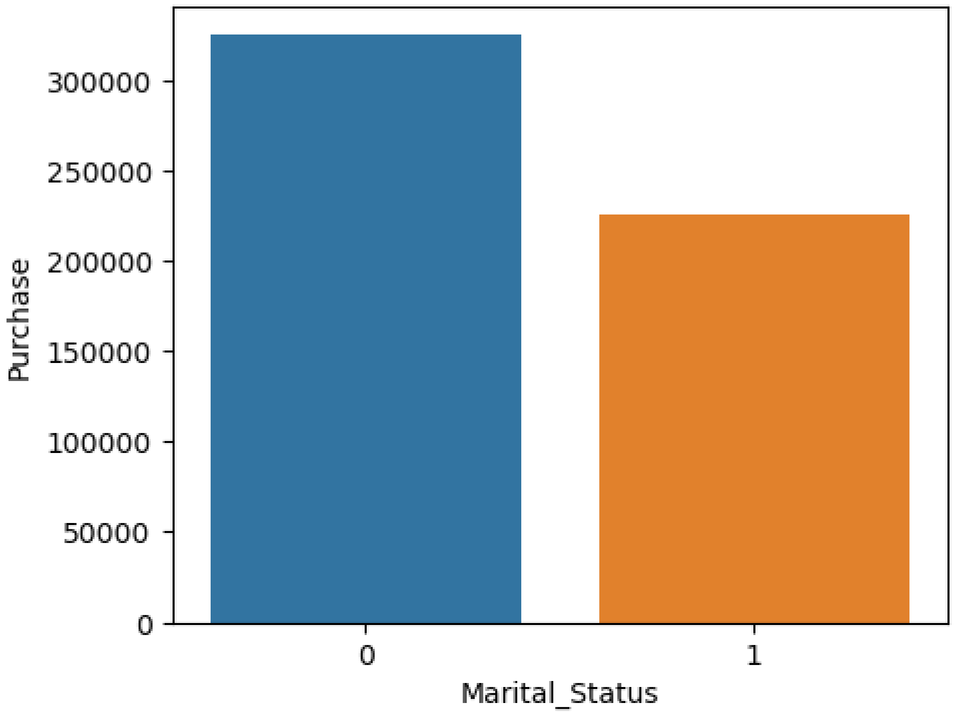


```
We can infer from above countplot:

The age group between 0 to 17 and above 55 there is less number of
purchase at Walmart irrrespective of the gender.
The number of purchase made by males is higher than females in all the age
groups.
The highest number of purchase is done by the age group 26-35 males.
```

In [62]:
```
plt.figure(figsize=(5,4))
sns.countplot(x='Marital_Status',data=df)
plt.ylabel('Purchase')
```

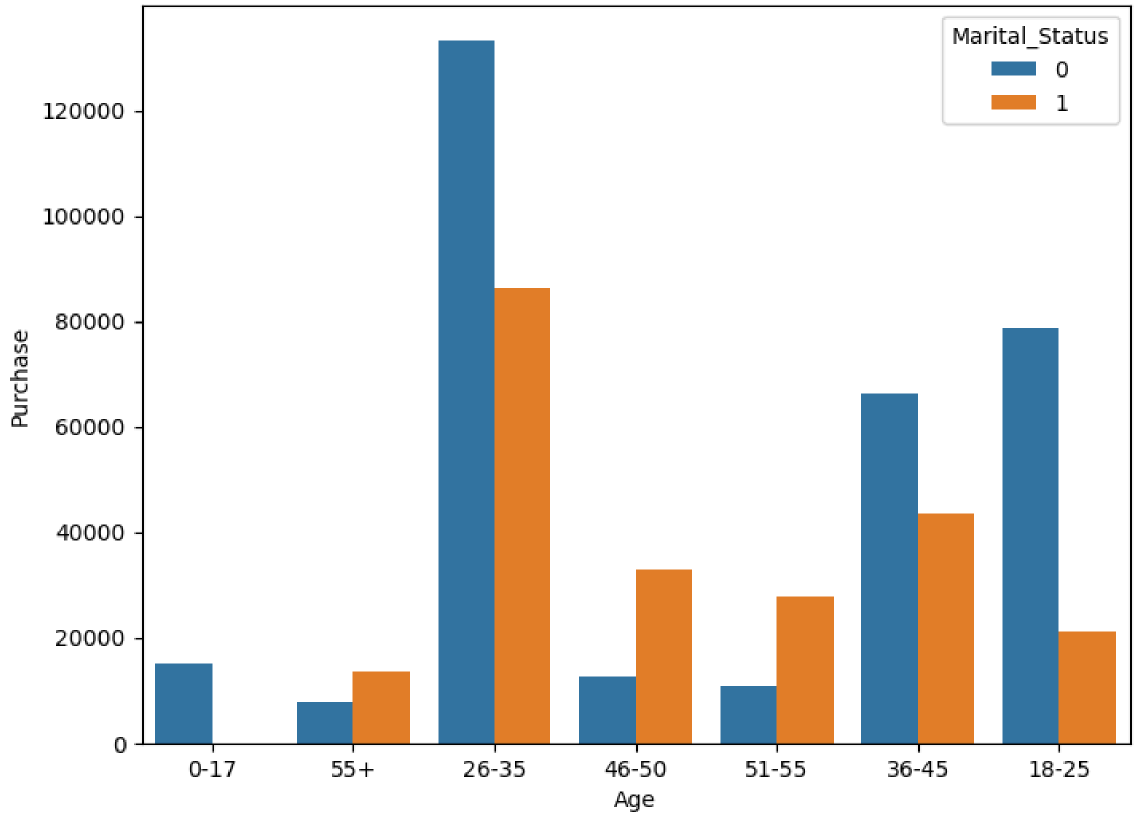
Out[62]:
```
Text(0, 0.5, 'Purchase')
```



```
From the above countplot it can be seen that Purchase done by non married
is more than the married people.
```

```
In [65]: plt.figure(figsize=(8,6))
         sns.countplot(x='Age',hue='Marital_Status',data=df)
         plt.ylabel('Purchase')

Out[65]: Text(0, 0.5, 'Purchase')
```

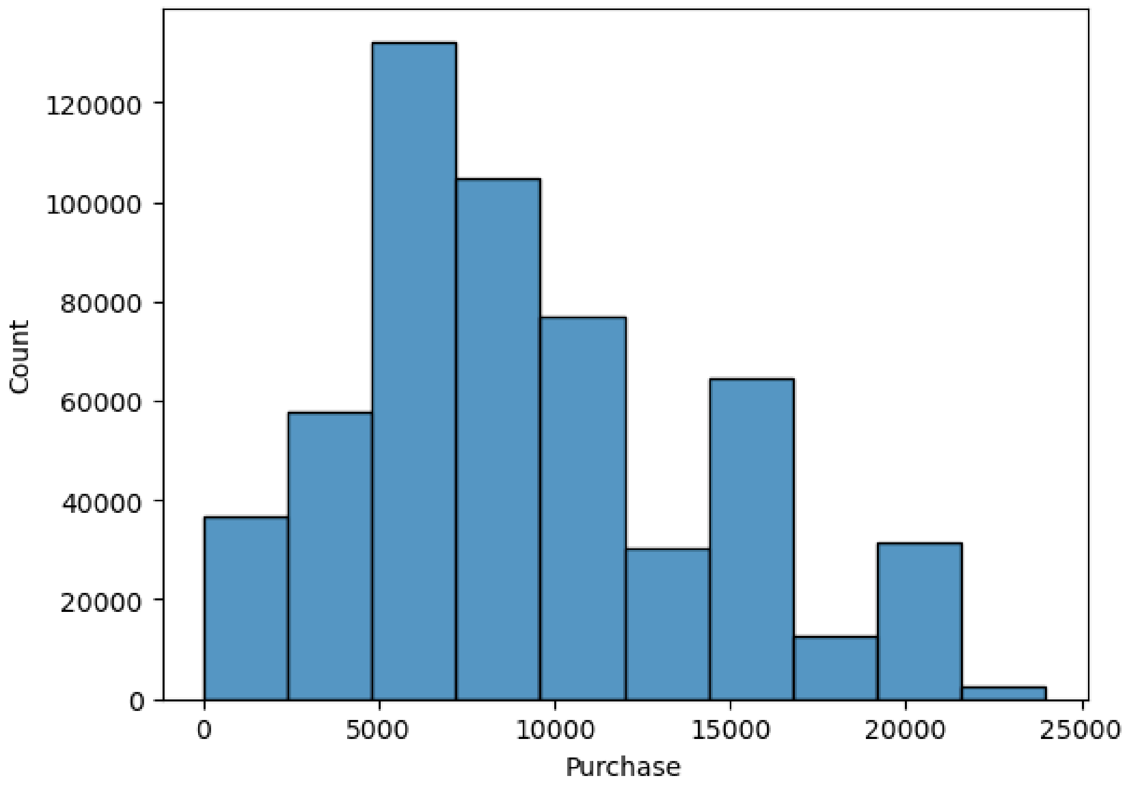


```
It can be infered from the above countplot that highest number of purchase
done by non married people in the age group 26-35.And the least by the non
married of age of age group above 55 and below 17.
```

Histplot

In [86]:
```
sns.histplot(df['Purchase'],bins = 10)
```

Out[86]:
```
<Axes: xlabel='Purchase', ylabel='Count'>
```

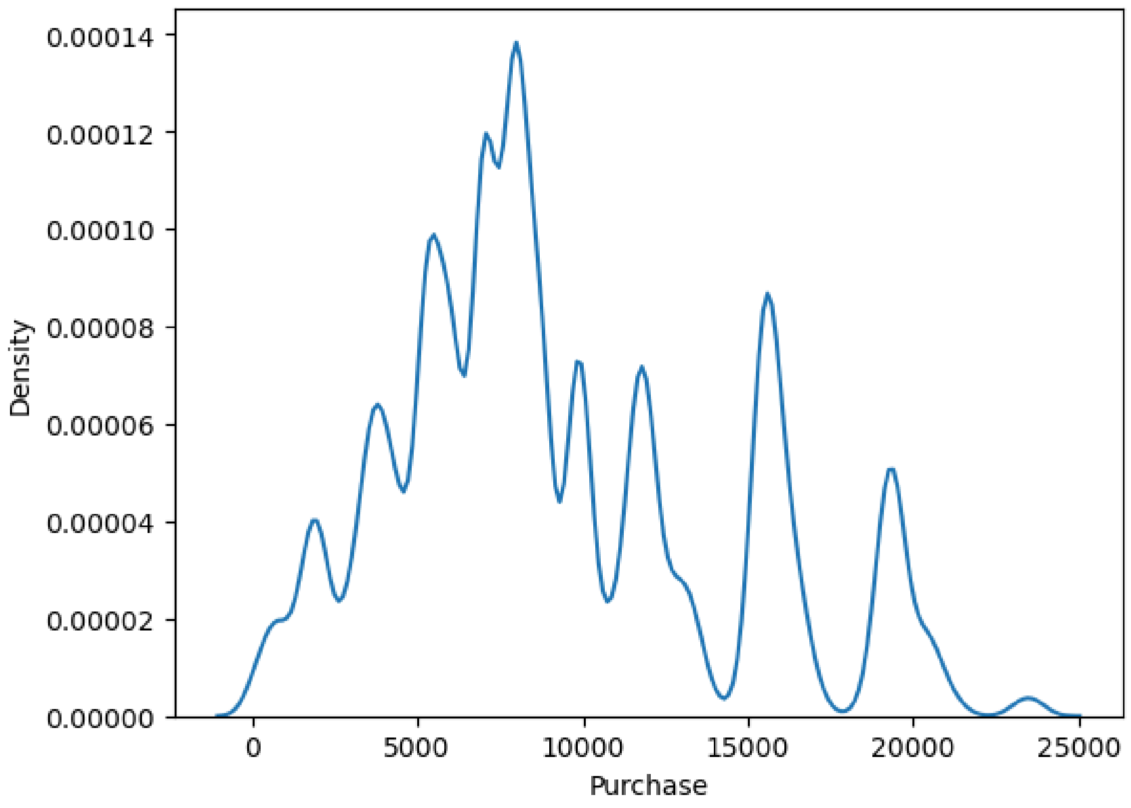


```
From the histplot it can be seen that the number of purchase made in the
range of 5000 to 10000 is highert
```

KDE Plot

In [94]:
```
sns.kdeplot(df['Purchase'])
```

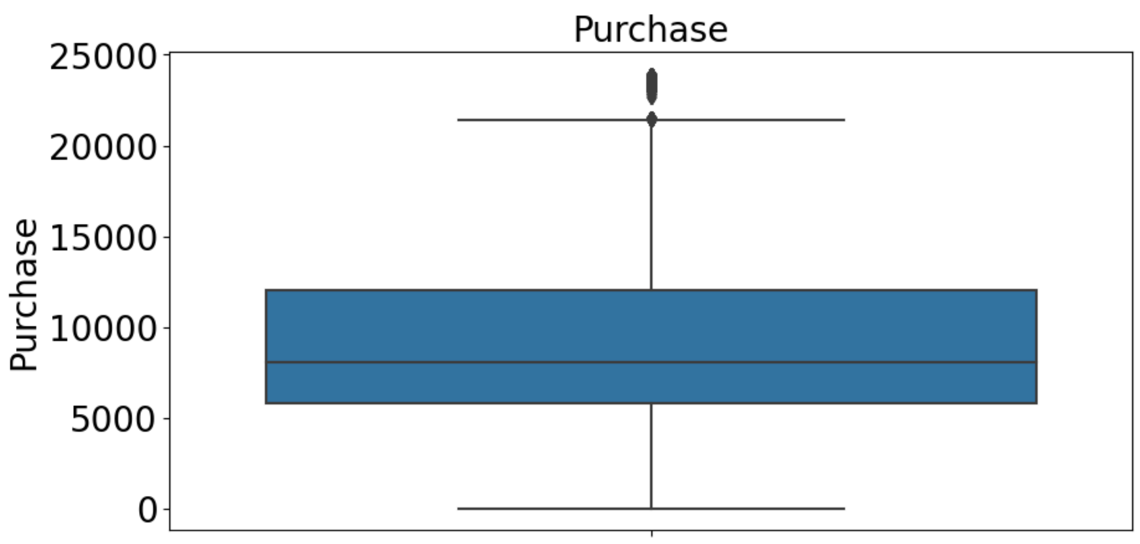
Out[94]:
```
<Axes: xlabel='Purchase', ylabel='Density'>
```



BoxPlot

In [82]:
```
plt.figure(figsize=(10,5))
sns.boxplot(y = df["Purchase"])
plt.yticks(fontsize=20)
plt.ylabel('Purchase', fontsize=20)
plt.title('Purchase', fontsize=20)
```

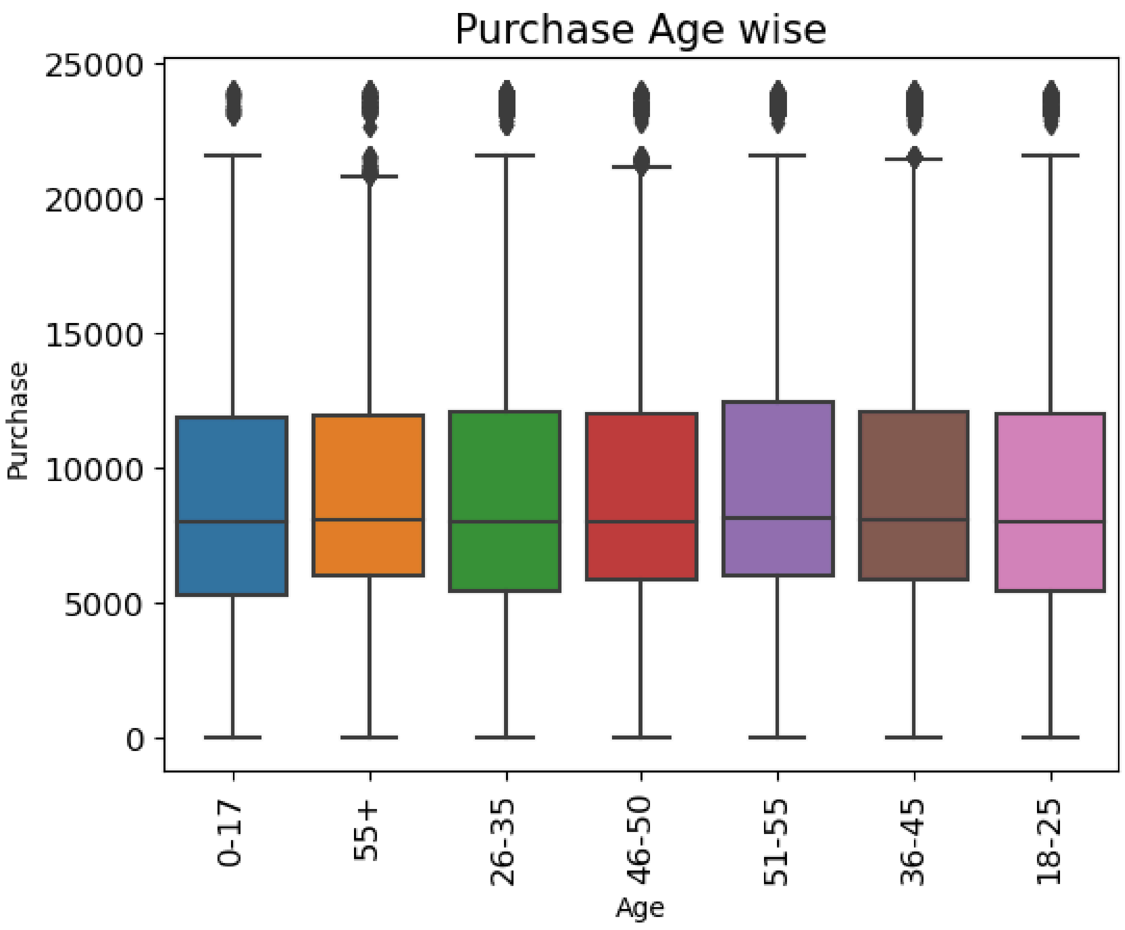
Out[82]:
```
Text(0.5, 1.0, 'Purchase')
```



```
Minimum, excluding outliers: 0
Maximum, excluding outliers: 20,000 above
25th Quantile: 5000
Median: around 7000
75th Quantile: 12500
```

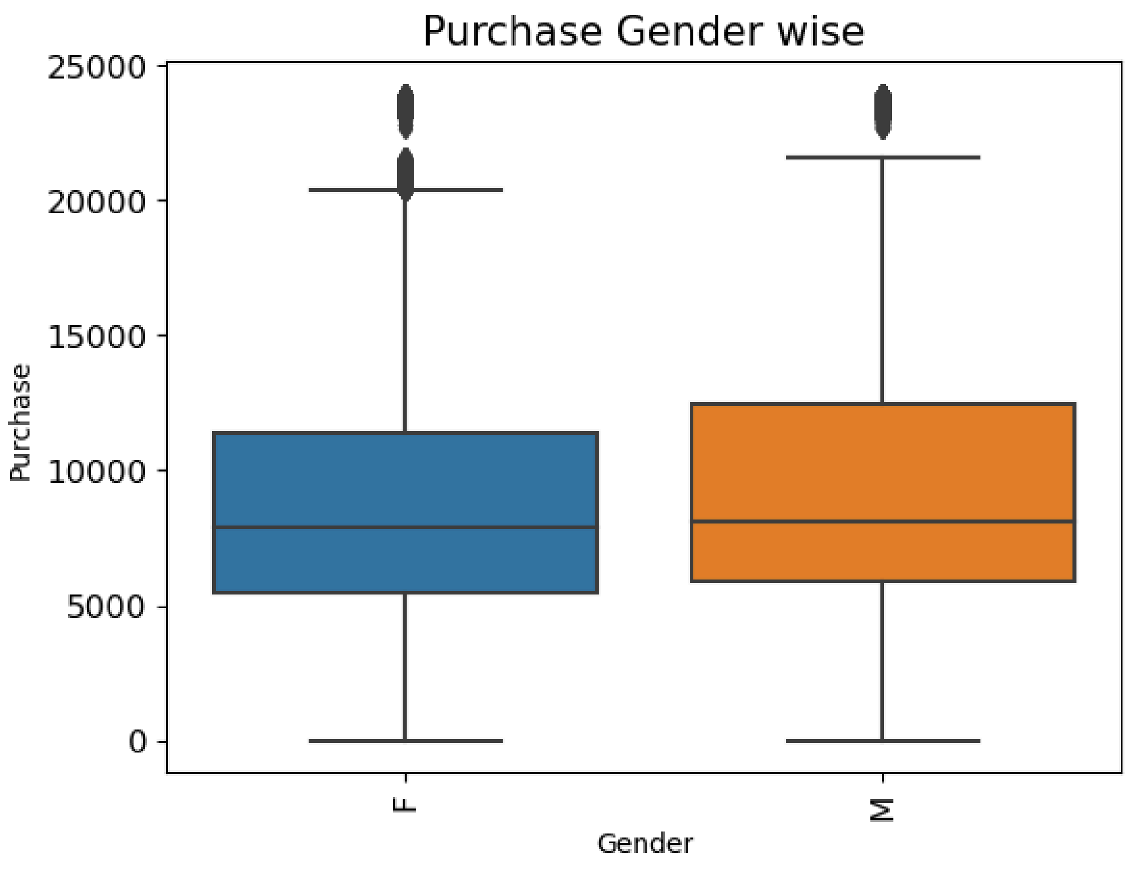
```
There are few outliers
```

In [88]:
```
sns.boxplot(x='Age', y='Purchase', data=df)
plt.xticks(rotation=90,fontsize=12)
plt.yticks(fontsize=12)
plt.title('Purchase Age wise', fontsize=15)
plt.show()
```



```
The overall Purchase of all the age category , has similar spread.(5000 to
12000)
All the age group have many outliers.
the median is almost the same for all the age group.
```

In [92]:
```
sns.boxplot(x='Gender', y='Purchase', data=df)
plt.xticks(rotation=90,fontsize=12)
plt.yticks(fontsize=12)
plt.title('Purchase Gender wise', fontsize=15)
plt.show()
```

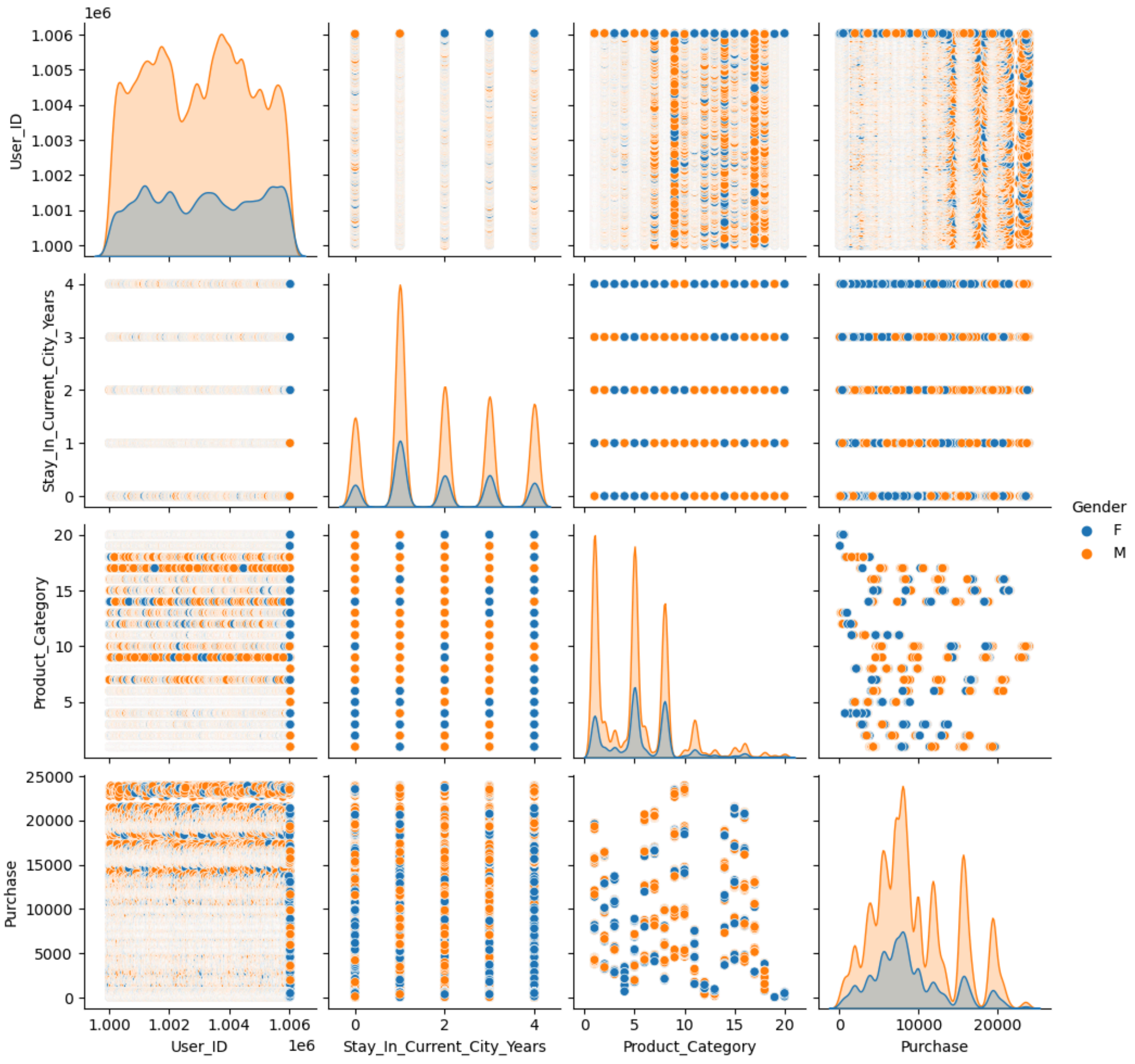


```
The overall Purchase of both the gender , has similar spread.(5000 to
12000)
Both the gender have many outliers.
the median is almost the same for both the gender.
```

Pairplot

In [99]:
```
sns.pairplot(data=df, hue='Gender')
plt.show()
```

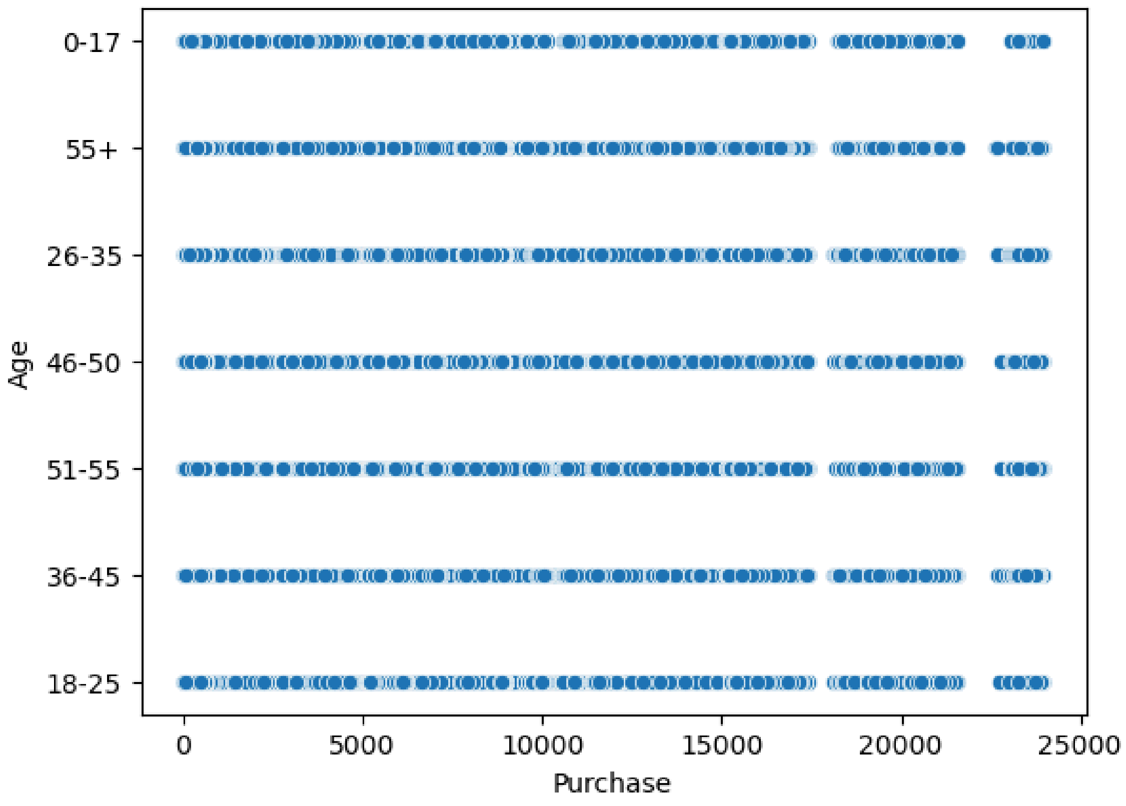
```
C:\Users\Hp\anaconda3\Lib\site-packages\seaborn\axisgrid.py:118: UserWarni
ng: The figure layout has changed to tight
  self._figure.tight_layout(*args, **kwargs)
```



Correlation Matrix

In [100]:
```
#to find the level of correlation b/w different attributes (variables)
```

In [102]:
```
sns.scatterplot(x= 'Purchase', y= 'Age', data = df)
plt.show()
```



Heatmap

Missing Value & Outlier Detection

In [104]:
```
df.isnull().values.any()
```

Out[104]:
```
False
```

In [105]:
```
df.isnull().sum()
```

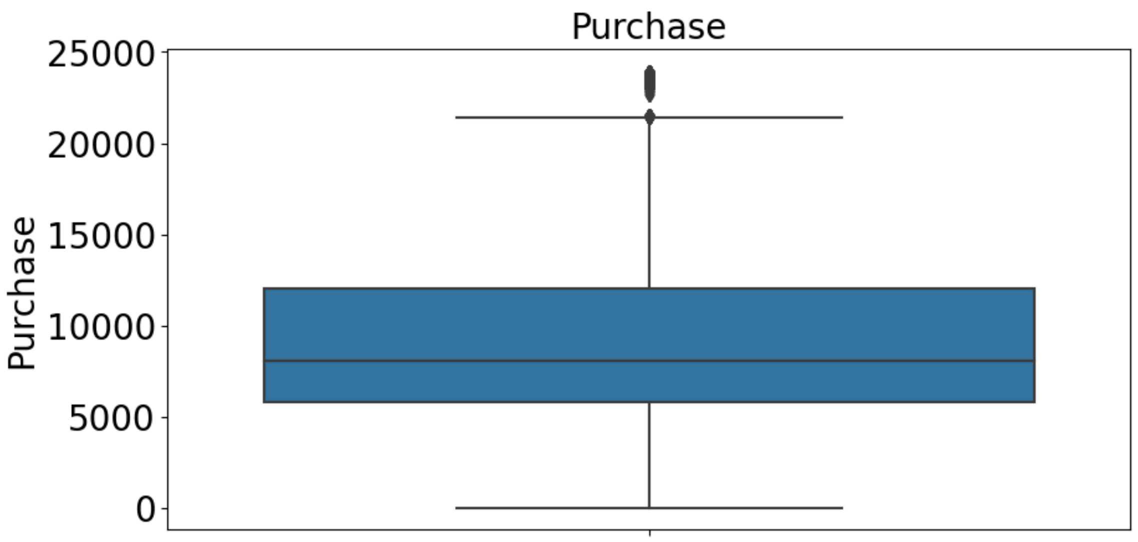
Out[105]:
```
User_ID                       0
Product_ID                    0
Gender                        0
Age                           0
Occupation                    0
City_Category                 0
Stay_In_Current_City_Years    0
Marital_Status                0
Product_Category              0
Purchase                      0
dtype: int64
```

There are no missing values

In [108]:
```
#Outliers can be detected using boxplot
```

In [106]:
```
plt.figure(figsize=(10,5))
sns.boxplot(y = df["Purchase"])
plt.yticks(fontsize=20)
plt.ylabel('Purchase', fontsize=20)
plt.title('Purchase', fontsize=20)
```

Out[106]:
```
Text(0.5, 1.0, 'Purchase')
```



There are many outliers and they can be treated using various techniques

1. Minimum Score It is the lowest value 5000.
2. Lower Quartile 25% of values fall below the lower quartile value. It is also known as the first quartile.
3. Median Median marks the mid-point of the data. It is shown by the line that divides the box into two parts. Half the scores are greater than or equal to this value and half are less. It is sometimes known as the second quartile.
4. Upper Quartile 75% of values fall below the upper quartile value. It is also known as the third quartile. Maximum Score It is the highest value, excluding outliers. It is shown at the end of upper whisker. Whiskers The upper and lower whiskers represent values outside the middle 50%. That is, the lower 25% of values and the upper 25% of values. Interquartile Range (or IQR) This is the box plot showing the middle 50% of scores. It is the range between the 25th and 75th percentile.

Do some data exploration steps like:

-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.

-Inference after computing the average female and male expenses.

-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 [112]:
```
grouped = df.groupby('Gender')['Purchase'].sum()
print(grouped)
```

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

In [113]:
```
df["Purchase"].sum()
```

Out[113]:
```
5095812742
```

In [117]:
```
total_purchase_male = 3909580100
total_purchase_female = 1186232642

total_purchase = 5095812742

# Calculate percent  purchase for female customers
percentage_female = total_purchase_female / total_purchase

# Calculate percent amount purchase for male customers
percentage_male = total_purchase_male / total_purchase
```

In [118]:
```
print(percentage_female)
```

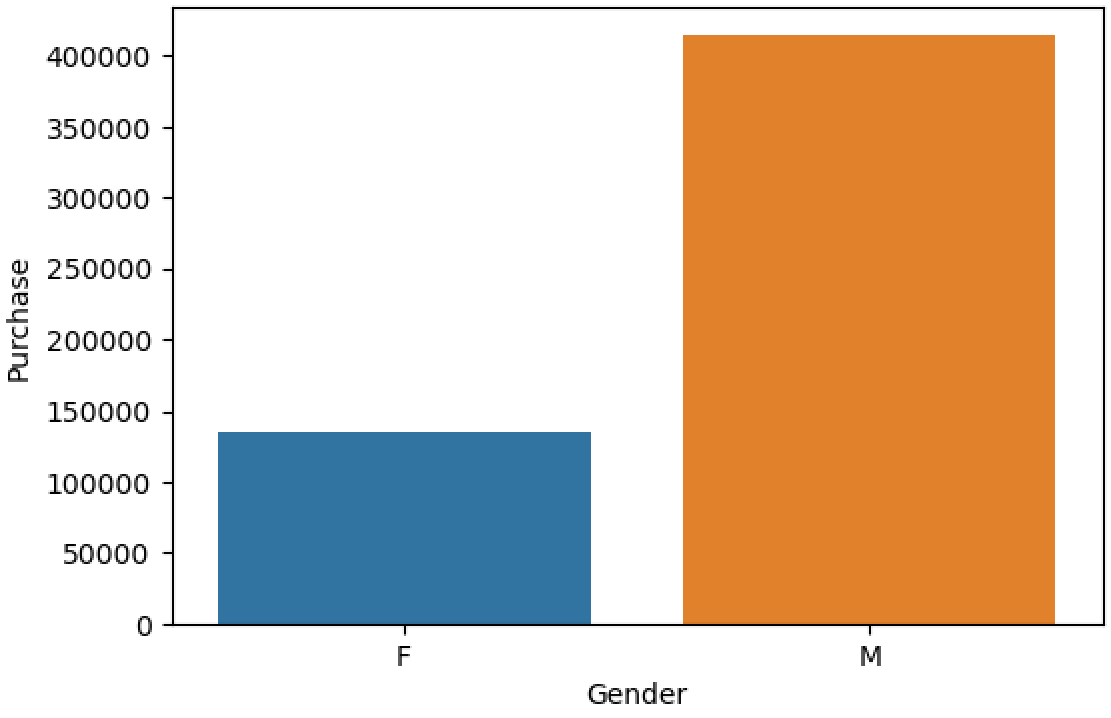
```
0.2327857600070344
```

In [119]:
```
print(percentage_male)
```

```
0.7672142399929656
```

In [121]:
```
plt.figure(figsize=(6,4))
sns.countplot(x='Gender',data=df)
plt.ylabel('Purchase')
```

Out[121]:
```
Text(0, 0.5, 'Purchase')
```



In [122]:
```
count1 = df.groupby('Gender')['Purchase'].count()
print(count1)
```

```
Gender
F    135809
M    414259
Name: Purchase, dtype: int64
```

In [123]:
```
num_transactions_female = 135809
num_transactions_male = 414259
```

In [124]:
```
#Calculate the average amount spent per transaction
average_female = total_purchase_female / num_transactions_female
average_male = total_purchase_male / num_transactions_male
```

In [127]:
```
print(average_female,average_male)
```

```
8734.565765155476 9437.526040472265
```

women are not spending more money per transaction than men.

Since the number of transactions by women and also the amount spent on the purchase is less compared to men

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.

```
To calculate the confidence interval for the population average spending
of female customers and male customers based on the sample averages, we
can use the t-distribution since we are dealing with sample averages and
want to estimate the population mean.
```

```
z1=norm.ppf(0.025)  as we have 2.5% data till z1
Similarly to calculate z2 will use

z2=norm.ppf(1-0.025)  as we have 2.5% data remaining after z2
We can also represent the z1 points as 2.5th percentile and z2 as 97.5th
percentile (1-0.025)
```

In [194]:
```
# z1 will be
z1 = norm.ppf(0.025)
z1
```

Out[194]:
```
-1.9599639845400545
```

In [195]:
```
# z2 will be
z2 = norm.ppf(1 - 0.025)  # we can also use norm.ppf(0.975)
z2
```

Out[195]:
```
1.959963984540054
```

Z=X¯−μσn√

Where,

X¯ = sample mean μ - population mean σ/n−−√ = standard error Here in the above example our sample S, the population mean will lie between z scores i.e. z1 and z2 which is -1.96 and 1.96.

Between these two points, we will have our 95% confidence interval.

In [ ]:
```
## Female CI
```

In [137]:
```
import math
```

In [229]:
```
# Calculate the mean
mean_female = 1186232642 / 135809



# Calculate the standard deviation
standard_deviation_female = (math.sqrt(mean_female / 135809))
```

```
In [230]: print(mean_female)

          8734.565765155476

In [231]: print(standard_deviation_female)

          0.25360416620895176

In [232]: # std deviation  sigma/sqrt(n)
          std_error_female = .25/np.sqrt(135809)
          std_error_female

Out[232]: 0.0006783842134393103
```

to get the data points for z1 which is on left side and z2 which is on right side,

X1=μ+Z1∗σ/n−−√ and,

X2=μ+Z2∗σ/n−−√

```
In [233]: x1 = 8734.56 + z1 * std_error_female
          x1

Out[233]: 8734.558670391374

In [234]: x2 = 8734.56 + z2 * std_error_female
          x2

Out[234]: 8734.561329608625
```

So the range of 95% confidence interval --> [8734.42, 8734.69]

```
In [235]: ## Using norm.interval()

In [236]: norm.interval(0.95, loc=8734.56, scale=std_error)

Out[236]: (8734.559558449768, 8734.560441550231)

In [237]: ## male CI

In [238]: # Calculate the mean
          mean_male = 3909580100 / 414259

In [239]: # Calculate the standard deviation
          standard_deviation_male = (math.sqrt(mean_female / 414259))

In [240]: print(mean_male)

          9437.526040472265

In [241]: print(standard_deviation_male)

          0.14520604054667763
```

In [203]:
```
# std deviation  sigma/sqrt(n)
std_error_male = 0.145/np.sqrt(414259)
std_error_male
```

Out[203]:
```
0.00022528487013342337
```

to get the data points for z1 which is on left side and z2 which is on right side,

X1=μ+Z1∗σ/n−−√ and,

X2=μ+Z2∗σ/n−−√

In [204]:
```
xm1 = 9437.5 + z1 * std_error_male
xm1
```

Out[204]:
```
9437.499558449768
```

In [205]:
```
xm2 = 9437.5 + z2 * std_error_male
xm2
```

Out[205]:
```
9437.500441550232
```

In [ ]:
```
So the range of 95% confidence interval --> [9437.49, 9437.500]
```

In [ ]:
```
## Using norm.interval()
```

In [170]:
```
norm.interval(0.95, loc=9437.5, scale=std_error)
```

Out[170]:
```
(9437.499558449768, 9437.500441550232)
```

The confidence interval (CI) gives us a range of values within which we are confident that the average spending per transaction lies. It provides measure of the uncertainty.

Based on the confidence intervals for the average spending of both female and male customers, we can make some inferences based on these intervals. Here are a few points of inference:

Female customers: 8734.42, 8734.69

Male customers: 9437.49,9437.50

Since there is no overlapping in the confidence intervals for the two groups,it suggests that there no is a significant difference in the average spending between the two groups.We can conclude with 95% confidence that there is no a statistically significant difference in the average spending per transaction between female and male customers. It might be due to the variability within the samples.

It's also essential to consider other factors that might influence spending behavior, such as age, location, type of products purchased, promotions, etc.

In [ ]:
```

```

Same way fro 99% Confident Interval

In [172]:
```
z1=norm.ppf(0.005) #as we have 0.5% data till z1
```

In [173]:
```

#Similarly to calculate z2 will use z2=norm.ppf(1-0.005)
```

as we have 0.5% data remaining after z2 We can also represent the z1 points as 0.5th percentile and z2 as 99.5th percentile (1-0.005)

In [188]:
```
# z1 will be
zz1 = norm.ppf(0.005)
zz1
```

Out[188]:
```
-2.575829303548901
```

In [189]:
```
# z2 will be
zz2 = norm.ppf(1 - 0.005)  # we can also use norm.ppf(0.975)
zz2
```

Out[189]:
```
2.5758293035489004
```

```
Z=X¯−μσn√

Where,

X¯ = sample mean
μ - population mean
σ/n−−√ = standard error
```

Here in the above example our sample S, the population mean will lie between z scores i.e. z1 and z2 which is -2.57 and 2.57.

Between these two points, we will have our 99% confidence interval.

In [177]:
```
## Female CI

import math
```

In [178]:
```
# Calculate the mean
mean_female = 1186232642 / 135809
```

In [179]:
```
# Calculate the standard deviation
standard_deviation_female = (math.sqrt(mean_female / 135809))*100
```

In [180]:
```
print(mean_female)
```

```
8734.565765155476
```

In [181]:
```
print(standard_deviation_female)
```

```
25.360416620895176
```

```
In [209]: # std deviation  sigma/sqrt(n)
          std_error_female = 25.3/np.sqrt(135809)
          std_error_female

Out[209]: 0.0686524824000582
```

to get the data points for z1 which is on left side and z2 which is on right side,

```
X1=μ+Z1∗σ/n−−√  and,

X2=μ+Z2∗σ/n−−√
```

```
In [210]: x1 = 8734.56 + zz1 * std_error_female
          x1

Out[210]: 8734.383162924072
```

```
In [211]: x2 = 8734.56 + zz2 * std_error_female
          x2

Out[211]: 8734.736837075927
```

```
In [ ]: So the range of 99% confidence interval --> [8734.38, 8734.73]
```

```
In [226]: ## Using norm.interval()

          norm.interval(0.99, loc=8734.56, scale=std_error)

Out[226]: (8734.559419704628, 8734.56058029537)
```

male CI

```
In [213]: # Calculate the mean
          mean_male = 3909580100 / 414259
```

```
In [214]: # Calculate the standard deviation
          standard_deviation_male = (math.sqrt(mean_female / 414259))
```

```
In [215]: print(mean_male)

          9437.526040472265
```

```
In [216]: print(standard_deviation_male)

          0.14520604054667763
```

```
In [217]: # std deviation  sigma/sqrt(n)
          std_error_male = 0.145/np.sqrt(414259)
          std_error_male

Out[217]: 0.00022528487013342337
```

to get the data points for z1 which is on left side and z2 which is on right side, X1=μ+Z1∗σ/n−−√ and,

X2=μ+Z2∗σ/n−−√

In [222]:
```
xm1 = 9437.5 + zz1 * std_error_male
xm1
```
Out[222]:
```
9437.499419704629
```

In [223]:
```
xm2 = 9437.5 + zz2 * std_error_male
xm2
```
Out[223]:
```
9437.500580295371
```

In [ ]:
```
So the range of 95% confidence interval --> [9437.49, 9437.500]
```

In [224]:
```
## Using norm.interval()

norm.interval(0.99, loc=9437.5, scale=std_error)
```
Out[224]:
```
(9437.499419704629, 9437.500580295371)
```

```
The confidence interval (CI) gives us a range of values within which we
are confident that the average spending per transaction lies. It provides
measure of the uncertainty.

Based on the confidence intervals for the average spending of both female
and male customers, we can make some inferences based on these intervals.
Here are a few points of inference:
```

In [ ]:
```
Female customers: 8734.38, 8734.73

Male customers: 9437.49,9437.50
```

Since there is no overlapping in the confidence intervals for the two groups,it suggests that there no is a significant difference in the average spending between the two groups.We can conclude with 95% confidence that there is no a statistically significant difference in the average spending per transaction between female and male customers. It might be due to the variability within the samples.

It's also essential to consider other factors that might influence spending behavior, such as age, location, type of products purchased, promotions, etc.

```
There is no differnce when confident interval is changed from 95 to 99.
```

the results and check if the confidence intervals of married and non married spends are overlapping or not overlapping. How can Walmart leverage this conclusion to make changes or improvements?

In [227]:
```
df.head()
```

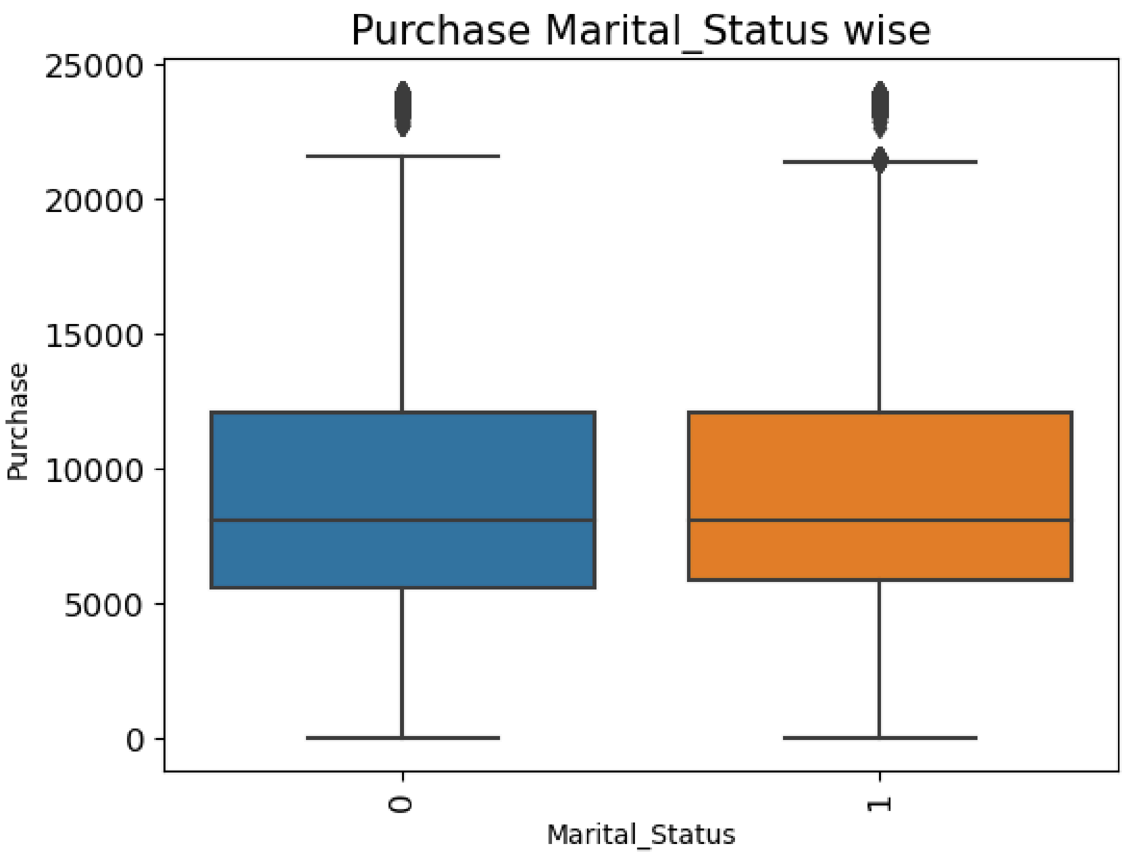
Out[227]:

	User_ID	Product_ID	Gender	Age	Occupation	City_Category	Stay_In_Current_City_Years
0	1000001	P00069042	F	0-17	10	A	2
1	1000001	P00248942	F	0-17	10	A	2
2	1000001	P00087842	F	0-17	10	A	2
3	1000001	P00085442	F	0-17	10	A	2
4	1000002	P00285442	M	55+	16	C	4

In [228]:
```
count_MS = df.groupby('Marital_Status')['Purchase'].count()
print(count_MS)
```

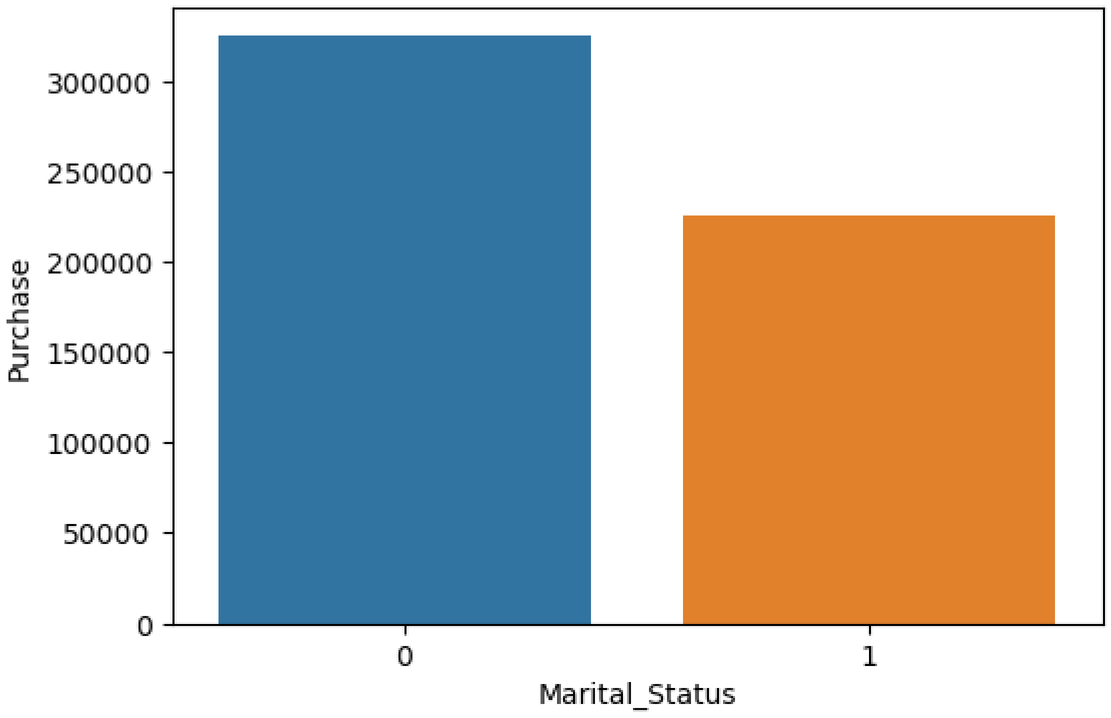
```
Marital_Status
0    324731
1    225337
Name: Purchase, dtype: int64
```

In [243]:
```
sns.boxplot(x='Marital_Status', y='Purchase', data=df)
plt.xticks(rotation=90,fontsize=12)
plt.yticks(fontsize=12)
plt.title('Purchase Marital_Status wise', fontsize=15)
plt.show()
```



In [244]:
```
plt.figure(figsize=(6,4))
sns.countplot(x='Marital_Status',data=df)
plt.ylabel('Purchase')
```

Out[244]:
```
Text(0, 0.5, 'Purchase')
```



In [ ]: