In [1]:
```
# importing the necessary libraries

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

```
/Users/pushpa/opt/anaconda3/lib/python3.9/site-packages/scipy/__init__.py
:155: UserWarning: A NumPy version >=1.18.5 and <1.25.0 is required for t
his version of SciPy (detected version 1.25.2
  warnings.warn(f"A NumPy version >={np_minversion} and <{np_maxversion}"
```

In [2]:
```
#loading the data set

df = pd.read_csv('Desktop/Wallmart.csv_1641285094.txt')
df
```

Out[2]:

	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

Basic details of the DataFrame

In [3]:
```
df.head()
```

Out[3]:

	User_ID	Product_ID	Gender	Age	Occupation	City_Category	Stay_In_Current_City_
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	

In [4]:
```
df.tail()
```

Out[4]:

	User_ID	Product_ID	Gender	Age	Occupation	City_Category	Stay_In_Current
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	

In [5]:
```
df.describe()
```

Out[5]:

	User_ID	Occupation	Marital_Status	Product_Category	Purchase
count	5.500680e+05	550068.000000	550068.000000	550068.000000	550068.000000
mean	1.003029e+06	8.076707	0.409653	5.404270	9263.968713
std	1.727592e+03	6.522660	0.491770	3.936211	5023.065394
min	1.000001e+06	0.000000	0.000000	1.000000	12.000000
25%	1.001516e+06	2.000000	0.000000	1.000000	5823.000000
50%	1.003077e+06	7.000000	0.000000	5.000000	8047.000000
75%	1.004478e+06	14.000000	1.000000	8.000000	12054.000000
max	1.006040e+06	20.000000	1.000000	20.000000	23961.000000

In [6]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 550068 entries, 0 to 550067
Data columns (total 10 columns):
 #   Column                      Non-Null Count   Dtype 
---  ------                      --------------   ----- 
 0   User_ID                     550068 non-null  int64 
 1   Product_ID                  550068 non-null  object
 2   Gender                      550068 non-null  object
 3   Age                         550068 non-null  object
 4   Occupation                  550068 non-null  int64 
 5   City_Category               550068 non-null  object
 6   Stay_In_Current_City_Years  550068 non-null  object
 7   Marital_Status              550068 non-null  int64 
 8   Product_Category            550068 non-null  int64 
 9   Purchase                    550068 non-null  int64 
dtypes: int64(5), object(5)
memory usage: 42.0+ MB
```

In [7]:
```
df.columns
```

Out[7]:
```
Index(['User_ID', 'Product_ID', 'Gender', 'Age', 'Occupation', 'City_Cate
gory',
       'Stay_In_Current_City_Years', 'Marital_Status', 'Product_Category'
,
       'Purchase'],
      dtype='object')
```

In [8]:
```
# Convert columns to the categorical data type
df['Gender'] = df['Gender'].astype('category')
df['City_Category'] = df['City_Category'].astype('category')
```

In [9]:
```
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  category
 3   Age                         550068 non-null  object  
 4   Occupation                  550068 non-null  int64   
 5   City_Category               550068 non-null  category
 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: category(2), int64(5), object(3)
memory usage: 34.6+ MB
```

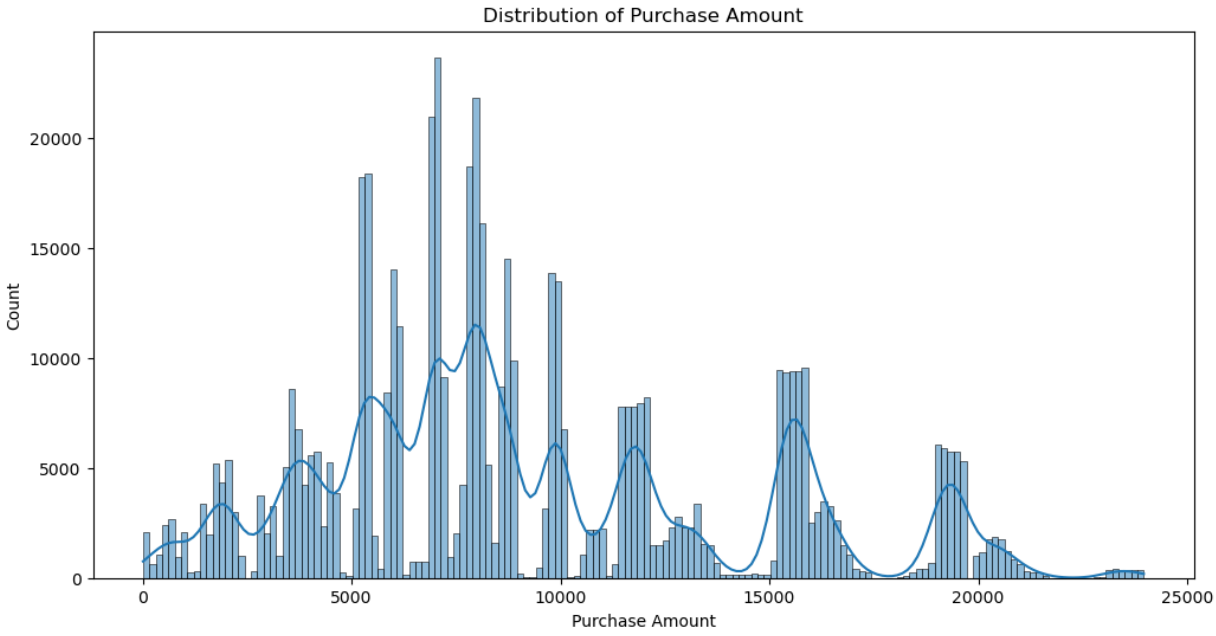
In [10]:
```
# Counting Genderwise
df['Gender'].value_counts()
```

Out[10]:
```
M    414259
F    135809
Name: Gender, dtype: int64
```

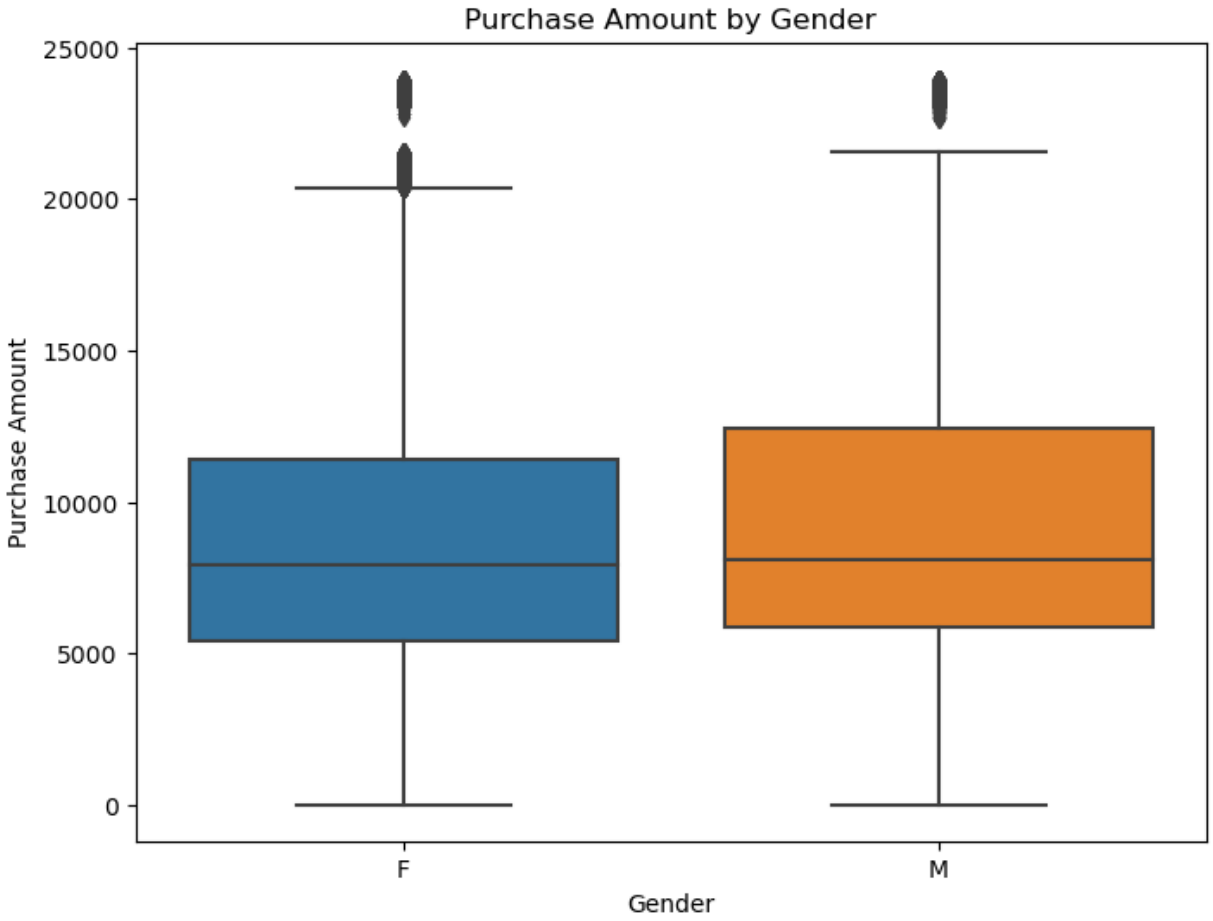
In [11]:
```
# Number of unique values
df.nunique()
```

Out[11]:
```
User_ID                       5891
Product_ID                    3631
Gender                           2
Age                              7
Occupation                      21
City_Category                    3
Stay_In_Current_City_Years       5
Marital_Status                   2
Product_Category                20
Purchase                     18105
dtype: int64
```

In [12]:
```
# Visual Analysis
# Univariate analysis for 'Purchase'
plt.figure(figsize=(12, 6))
sns.histplot(df['Purchase'], kde=True)
plt.title('Distribution of Purchase Amount')
plt.xlabel('Purchase Amount')
plt.ylabel('Count')
plt.show()
```

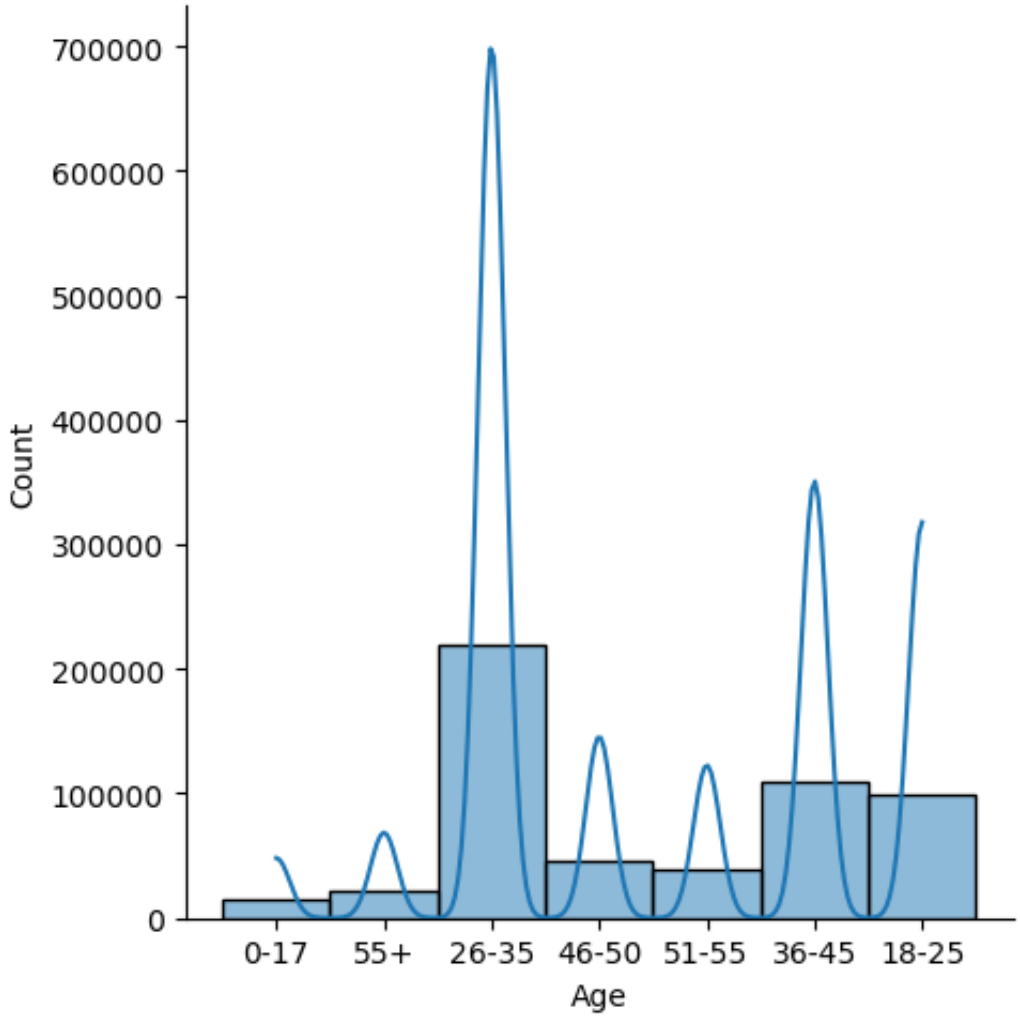


In [13]:
```
# Bivariate analysis: Boxplot for 'Gender' vs 'Purchase'
plt.figure(figsize=(8, 6))
sns.boxplot(x='Gender', y='Purchase', data=df)
plt.title('Purchase Amount by Gender')
plt.xlabel('Gender')
plt.ylabel('Purchase Amount')
plt.show()
```



In [14]:
```
sns.displot(data=df, x="Age", kde=True) # Majority of users are in the ag
```

Out[14]:
```
<seaborn.axisgrid.FacetGrid at 0x7fb59880ef70>
```



In [15]:
```
df.isna().sum()
```

Out[15]:
```
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
```

In [16]:
```
x = df[df['Gender']=='M'].count()
x
```

Out[16]:
```
User_ID                       414259
Product_ID                    414259
Gender                        414259
Age                           414259
Occupation                    414259
City_Category                 414259
Stay_In_Current_City_Years    414259
Marital_Status                414259
Product_Category              414259
Purchase                      414259
dtype: int64
```

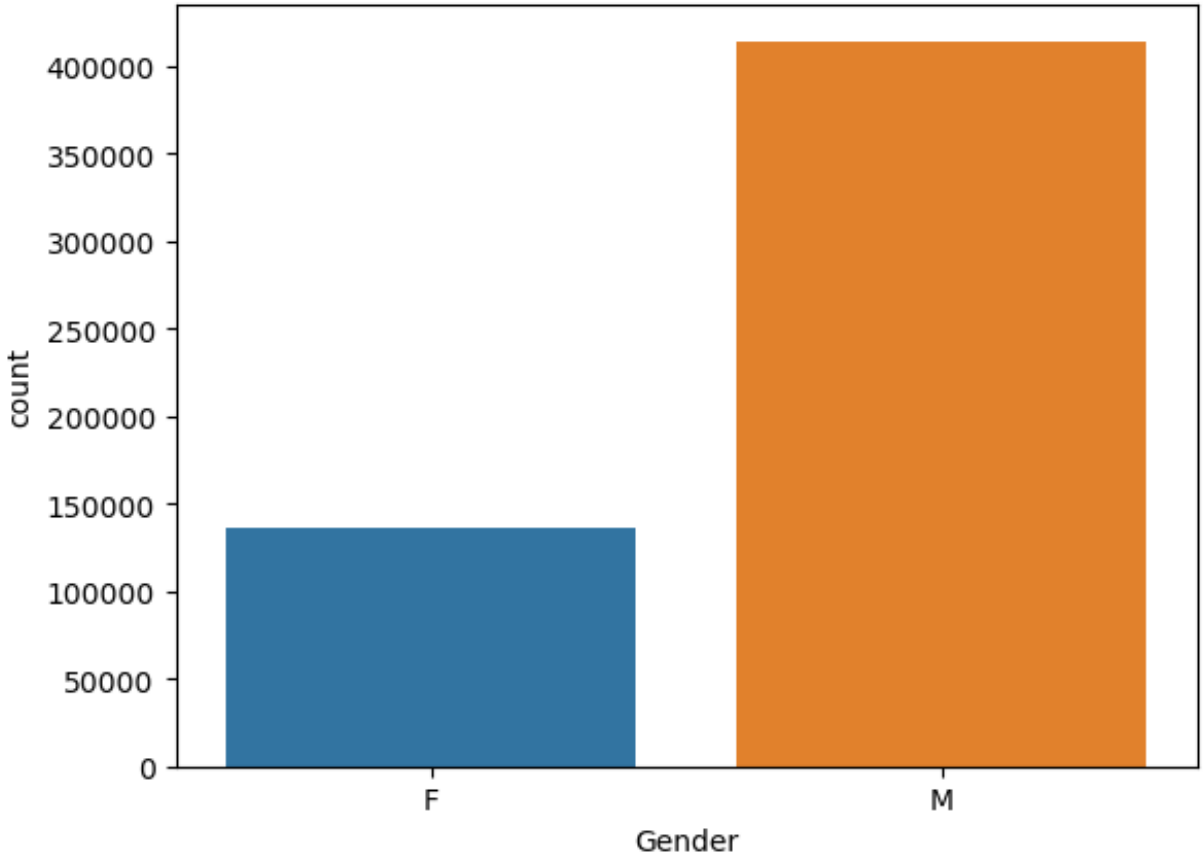
In [17]:
```
y = df[df['Gender']=='F'].count()
y
```

Out[17]:
```
User_ID                       135809
Product_ID                    135809
Gender                        135809
Age                           135809
Occupation                    135809
City_Category                 135809
Stay_In_Current_City_Years    135809
Marital_Status                135809
Product_Category              135809
Purchase                      135809
dtype: int64
```

In [18]:
```
sns.countplot(df['Gender']) # Majority of the users are Males
```

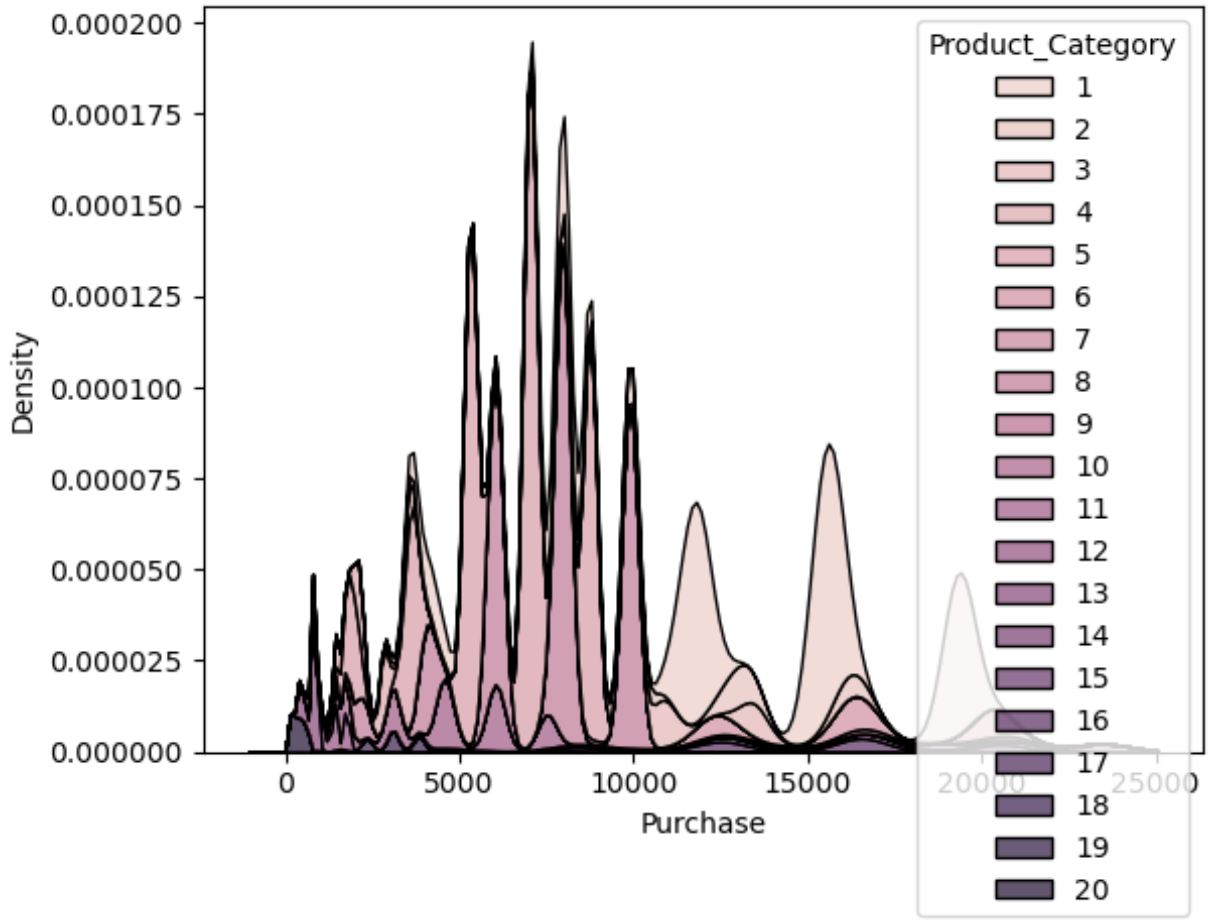
```
/Users/pushpa/opt/anaconda3/lib/python3.9/site-packages/seaborn/_decorato
rs.py:36: FutureWarning: Pass the following variable as a keyword arg: x.
From version 0.12, the only valid positional argument will be `data`, and
passing other arguments without an explicit keyword will result in an err
or or misinterpretation.
  warnings.warn(
```

Out[18]:
```
<AxesSubplot:xlabel='Gender', ylabel='count'>
```



In [19]:
```
sns.kdeplot(data=df, x='Purchase', hue='Product_Category', multiple="stac
```

Out[19]:
```
<AxesSubplot:xlabel='Purchase', ylabel='Density'>
```


0.000200
0.000175
0.000150
0.000125
0.000100
0.000075
0.000050
0.000025
0.000000
Density
0
5000
10000
15000
20000
25000
Purchase
Product_Category
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20

In [20]:
```
df
```

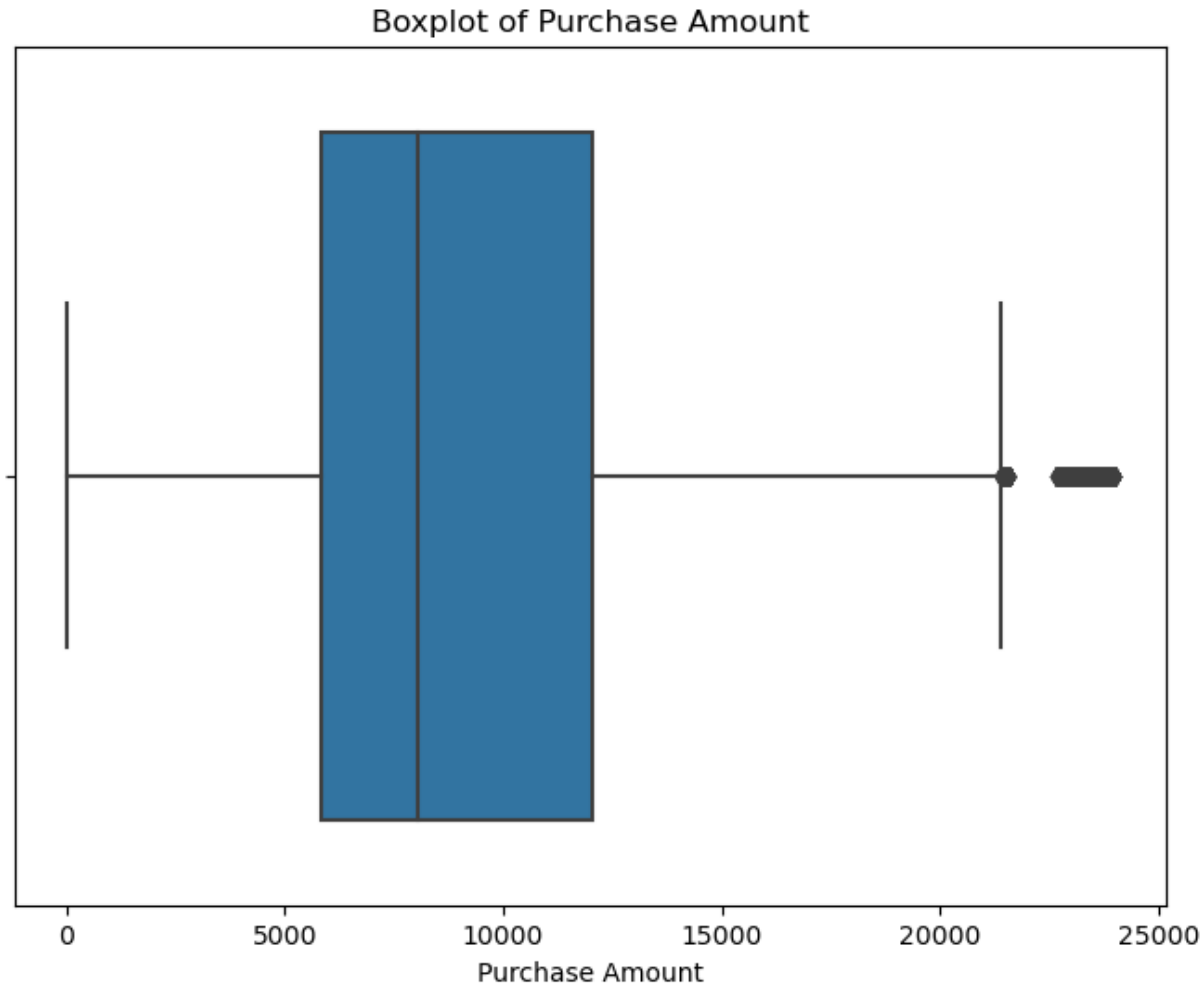
Out[20]:

	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

In [21]:

```
# Outlier detection using boxplot
plt.figure(figsize=(8, 6))
sns.boxplot(x='Purchase', data=df)
plt.title('Boxplot of Purchase Amount')
plt.xlabel('Purchase Amount')
plt.show()
```



In [22]:
```
# Detect outliers using IQR method
Q1 = df['Purchase'].quantile(0.25)
Q3 = df['Purchase'].quantile(0.75)
IQR = Q3 - Q1
lower_bound = Q1 - 1.5 * IQR
upper_bound = Q3 + 1.5 * IQR
outliers = df[(df['Purchase'] < lower_bound) | (df['Purchase'] > upper_bo
print("\nOutliers:")
print(outliers)
```

```
Outliers:
        User_ID Product_ID Gender    Age  Occupation City_Category  \
343     1000058  P00117642      M  26-35           2             B
375     1000062  P00119342      F  36-45           3             A
652     1000126  P00087042      M  18-25           9             B
736     1000139  P00159542      F  26-35          20             C
1041    1000175  P00052842      F  26-35           2             B
...         ...        ...    ...    ...         ...           ...
544488  1005815  P00116142      M  26-35          20             B
544704  1005847  P00085342      F  18-25           4             B
544743  1005852  P00202242      F  26-35           1             A
545663  1006002  P00116142      M  51-55           0             C
545787  1006018  P00052842      M  36-45           1             C

       Stay_In_Current_City_Years  Marital_Status  Product_Category  Purc
hase
343                             3               0                10     2
3603
375                             1               0                10     2
3792
652                             1               0                10     2
3233
736                             2               0                10     2
3595
1041                            1               0                10     2
3341
...                           ...             ...               ...
...
544488                          1               0                10     2
3753
544704                          2               0                10     2
3724
544743                          0               1                10     2
3529
545663                          1               1                10     2
3663
545787                          3               0                10     2
3496

[2677 rows x 10 columns]
```

In [23]:
```
# Calculate average spending for male and female customers
average_purchase_by_gender = df.groupby('Gender')['Purchase'].mean()
print("\nAverage spending by gender:")
print(average_purchase_by_gender)
```

```
Average spending by gender:
Gender
F    8734.565765
M    9437.526040
Name: Purchase, dtype: float64
```

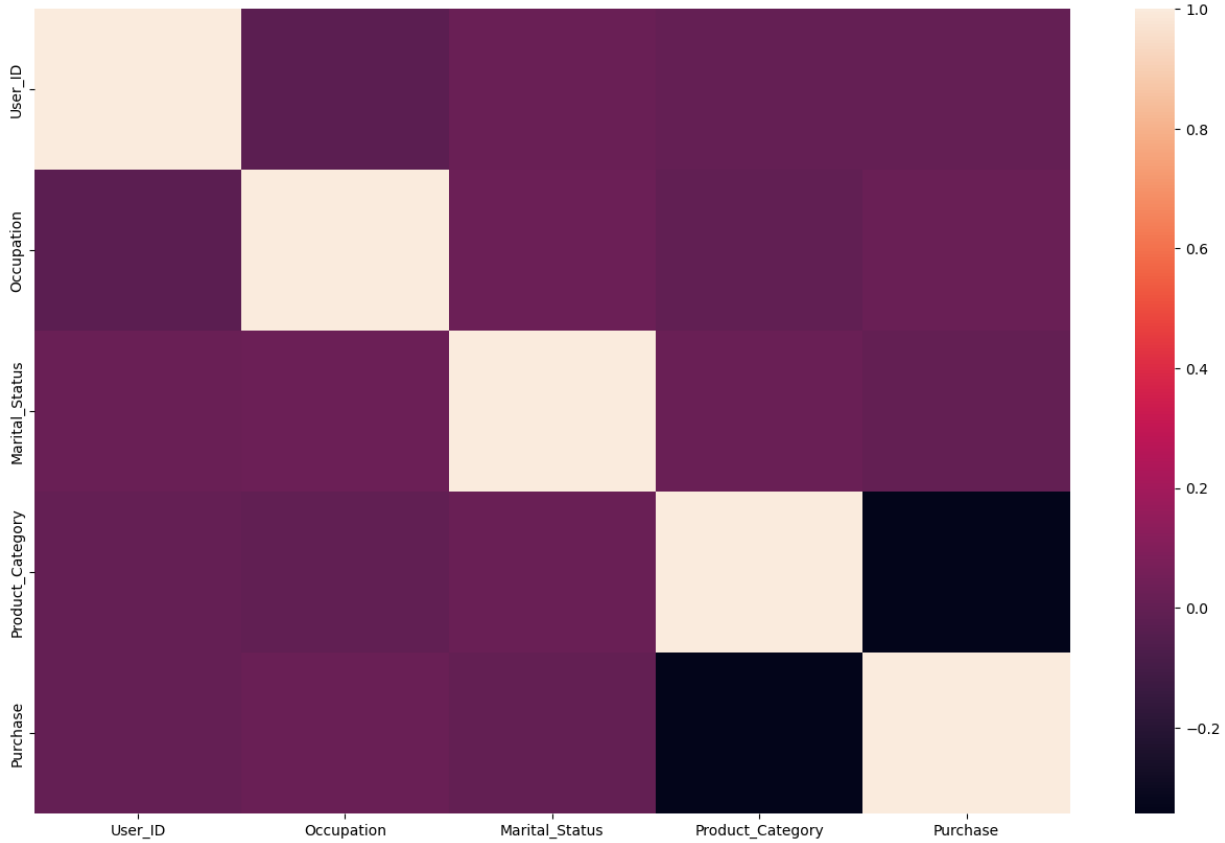
In [24]:
```
a = df.corr()
a
```

Out[24]:

	User_ID	Occupation	Marital_Status	Product_Category	Purchase
User_ID	1.000000	-0.023971	0.020443	0.003825	0.004716
Occupation	-0.023971	1.000000	0.024280	-0.007618	0.020833
Marital_Status	0.020443	0.024280	1.000000	0.019888	-0.000463
Product_Category	0.003825	-0.007618	0.019888	1.000000	-0.343703
Purchase	0.004716	0.020833	-0.000463	-0.343703	1.000000

In [25]:
```
#Corelation
cor = df.corr()
plt.figure(figsize=(16,10))
sns.heatmap(cor)
```

Out[25]:
```
<AxesSubplot:>
```

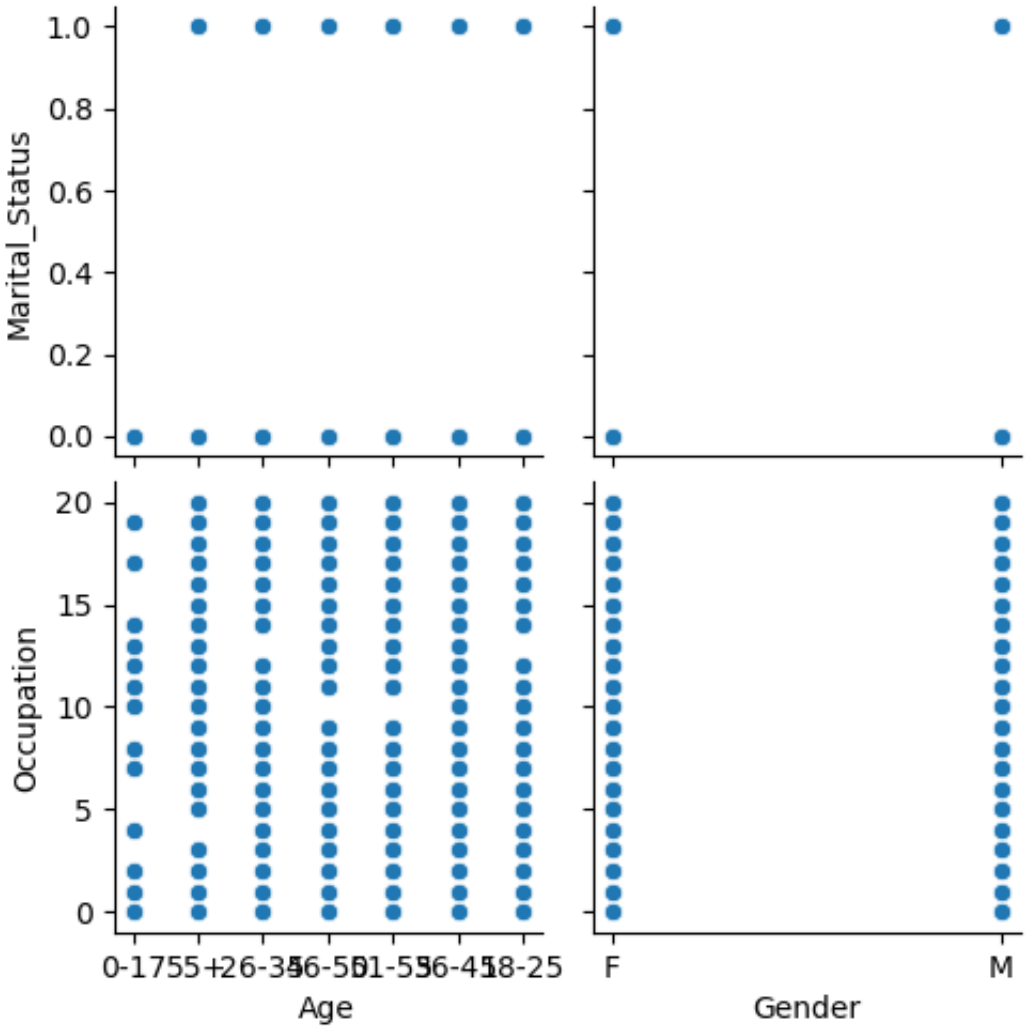


In [26]:
```
df['Product_Category'].value_counts(normalize=True)
```

Out[26]:
```
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
```

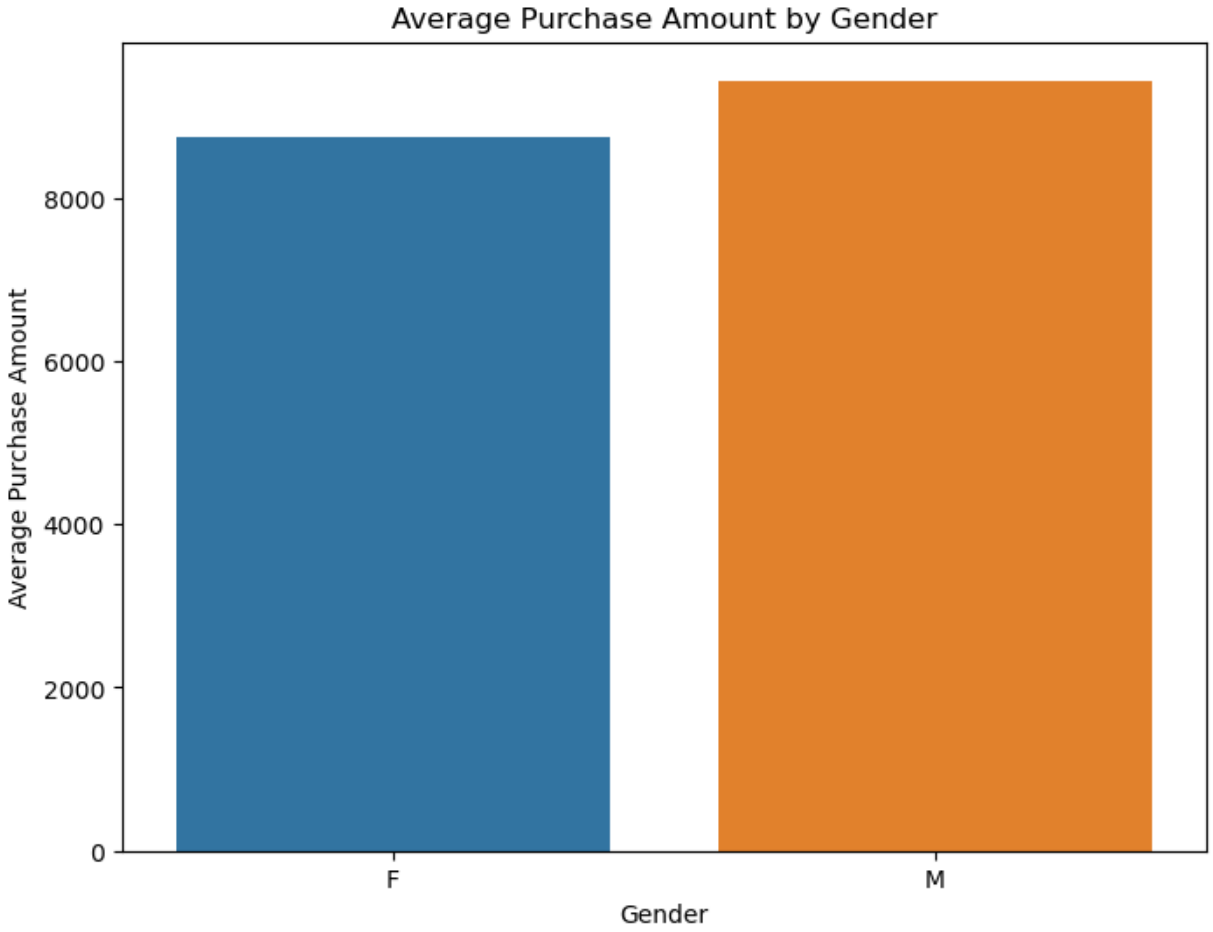
In [27]:
```
sns.pairplot(df, x_vars=['Age','Gender'], y_vars=['Marital_Status', 'Occu
```

Out[27]:
```
<seaborn.axisgrid.PairGrid at 0x7fb5a869b610>
```



In [28]:
```
# Calculate average spending for male and female customers
average_purchase_by_gender = df.groupby('Gender')['Purchase'].mean()

# Print the average spending by gender
print("Average spending by gender:")
print(average_purchase_by_gender)

# Visualize the average spending by gender
plt.figure(figsize=(8, 6))
sns.barplot(x=average_purchase_by_gender.index, y=average_purchase_by_gen
plt.xlabel('Gender')
plt.ylabel('Average Purchase Amount')
plt.title('Average Purchase Amount by Gender')
plt.show()
```

```
Average spending by gender:
Gender
F    8734.565765
M    9437.526040
Name: Purchase, dtype: float64
```

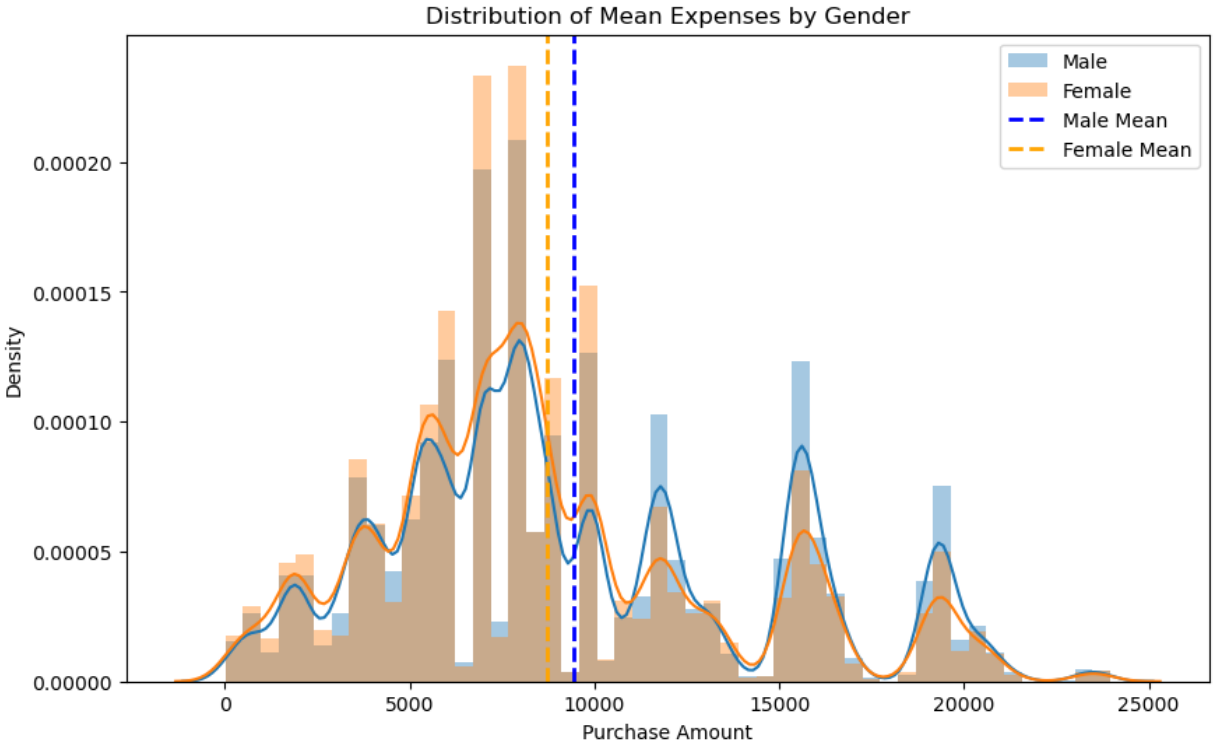


Males are spending more than thier counterparts in terms of average spend

In [29]:
```
# Calculate mean and standard deviation for male and female customers


# Calculate mean and standard deviation for male and female customers
mean_male = df[df['Gender'] == 'M']['Purchase'].mean()
mean_female = df[df['Gender'] == 'F']['Purchase'].mean()
std_male = df[df['Gender'] == 'M']['Purchase'].std()
std_female = df[df['Gender'] == 'F']['Purchase'].std()

# Sample sizes (assuming equal sample sizes)
n_male = len(df[df['Gender'] == 'M'])
n_female = len(df[df['Gender'] == 'F'])

# Calculate standard error for male and female customers
se_male = std_male / np.sqrt(n_male)
se_female = std_female / np.sqrt(n_female)

# Z-score for a 95% confidence interval (assuming normal distribution)
z_score = stats.norm.ppf(0.975)  # for a two-tailed 95% confidence interv

# Calculate confidence intervals
ci_male = (mean_male - z_score * se_male, mean_male + z_score * se_male)
ci_female = (mean_female - z_score * se_female, mean_female + z_score * s

# Print the confidence intervals
print("Confidence Interval for Male Mean:", ci_male)
print("Confidence Interval for Female Mean:", ci_female)

# Visualize the distribution of the mean expenses by gender
plt.figure(figsize=(10, 6))
sns.distplot(df[df['Gender'] == 'M']['Purchase'], label='Male')
sns.distplot(df[df['Gender'] == 'F']['Purchase'], label='Female')
plt.axvline(mean_male, color='blue', linestyle='dashed', linewidth=2, lab
plt.axvline(mean_female, color='orange', linestyle='dashed', linewidth=2,
plt.xlabel('Purchase Amount')
plt.ylabel('Density')
plt.legend()
plt.title('Distribution of Mean Expenses by Gender')
plt.show()
```

```
Confidence Interval for Male Mean: (9422.01944736257, 9453.032633581959)
Confidence Interval for Female Mean: (8709.21154714068, 8759.919983170272
)
```

```
/Users/pushpa/opt/anaconda3/lib/python3.9/site-packages/seaborn/distribut
ions.py:2619: FutureWarning: `distplot` is a deprecated function and will
be removed in a future version. Please adapt your code to use either `dis
plot` (a figure-level function with similar flexibility) or `histplot` (a
n axes-level function for histograms).
  warnings.warn(msg, FutureWarning)
/Users/pushpa/opt/anaconda3/lib/python3.9/site-packages/seaborn/distribut
ions.py:2619: FutureWarning: `distplot` is a deprecated function and will
be removed in a future version. Please adapt your code to use either `dis
plot` (a figure-level function with similar flexibility) or `histplot` (a
n axes-level function for histograms).
  warnings.warn(msg, FutureWarning)
```

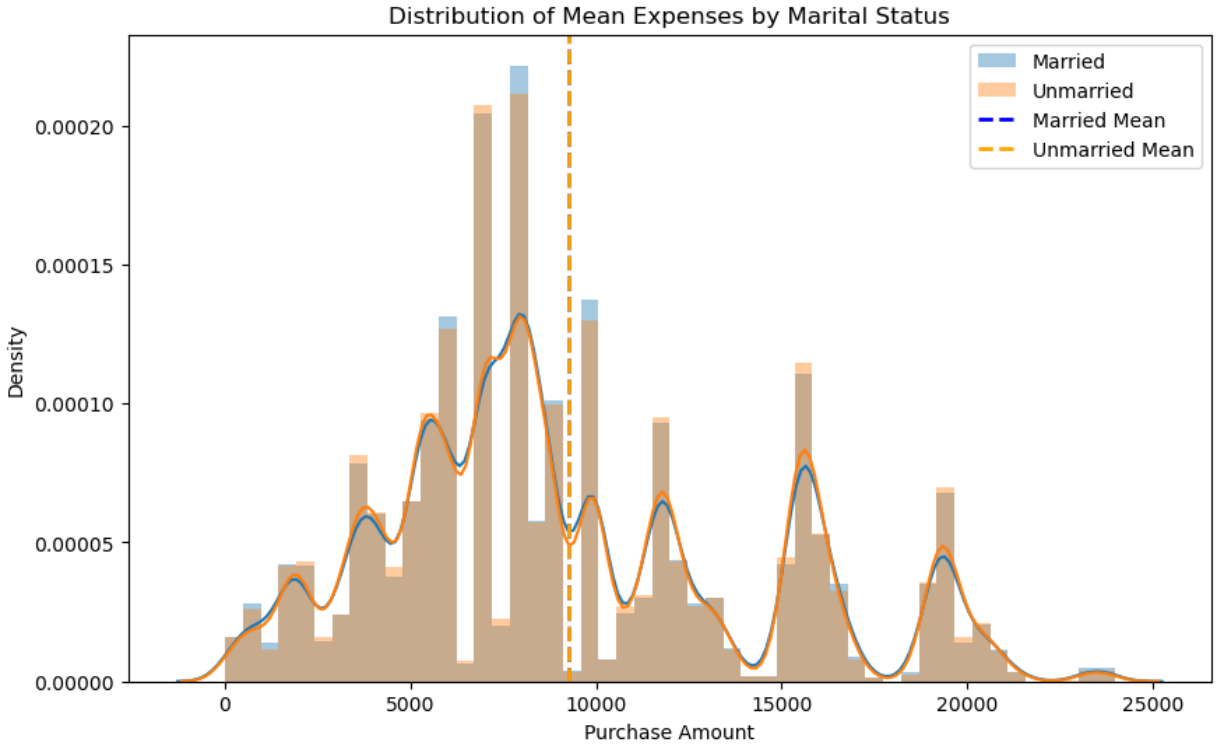


In [30]:
```
# Check for overlap
confidence_intervals_overlap = ci_male[1] >= ci_female[0] and ci_female[1

# Print the result
if confidence_intervals_overlap:
    print("Confidence intervals of average male and female spending overl
else:
    print("Confidence intervals of average male and female spending do no

# Provide insights and recommendations based on the comparison
print("\nInsights and Recommendations:")
if confidence_intervals_overlap:
    print("Since the CI of average spending for male and female customers
          "there is no significant statistical evidence")

else:
    print("The CI of average spending for male and female customers do no
          "indicating a potential difference in spending habits between g
    print("Walmart may consider tailored marketing strategies and promoti
```

```
Confidence intervals of average male and female spending do not overlap.

Insights and Recommendations:
The CI of average spending for male and female customers do not overlap,
indicating a potential difference in spending habits between genders.
Walmart may consider tailored marketing strategies and promotions to targ
et each gender specifically,
```

In [31]:
```
# Calculate mean and standard deviation for married and unmarried custome
mean_married = df[df['Marital_Status'] == 1]['Purchase'].mean()
mean_unmarried = df[df['Marital_Status'] == 0]['Purchase'].mean()
std_married = df[df['Marital_Status'] == 1]['Purchase'].std()
std_unmarried = df[df['Marital_Status'] == 0]['Purchase'].std()

# Sample sizes (assuming equal sample sizes)
n_married = len(df[df['Marital_Status'] == 1])
n_unmarried = len(df[df['Marital_Status'] == 0])

# Calculate standard error for married and unmarried customers
se_married = std_married / np.sqrt(n_married)
se_unmarried = std_unmarried / np.sqrt(n_unmarried)

# Z-score for a 95% confidence interval (assuming normal distribution)
z_score = stats.norm.ppf(0.975)  # for a two-tailed 95% confidence interv

# Calculate confidence intervals
ci_married = (mean_married - z_score * se_married, mean_married + z_score
ci_unmarried = (mean_unmarried - z_score * se_unmarried, mean_unmarried +

# Print the confidence intervals
print("Confidence Interval for Married Mean:", ci_married)
print("Confidence Interval for Unmarried Mean:", ci_unmarried)

# Visualize the distribution of the mean expenses by marital status
plt.figure(figsize=(10, 6))
sns.distplot(df[df['Marital_Status'] == 1]['Purchase'], label='Married')
sns.distplot(df[df['Marital_Status'] == 0]['Purchase'], label='Unmarried'
plt.axvline(mean_married, color='blue', linestyle='dashed', linewidth=2,
plt.axvline(mean_unmarried, color='orange', linestyle='dashed', linewidth
plt.xlabel('Purchase Amount')
plt.ylabel('Density')
plt.legend()
plt.title('Distribution of Mean Expenses by Marital Status')
plt.show()
```

```
Confidence Interval for Married Mean: (9240.460427057078, 9281.8887211076
69)
Confidence Interval for Unmarried Mean: (9248.61641818668, 9283.198819656
332)
/Users/pushpa/opt/anaconda3/lib/python3.9/site-packages/seaborn/distribut
ions.py:2619: FutureWarning: `distplot` is a deprecated function and will
be removed in a future version. Please adapt your code to use either `dis
plot` (a figure-level function with similar flexibility) or `histplot` (a
n axes-level function for histograms).
  warnings.warn(msg, FutureWarning)
/Users/pushpa/opt/anaconda3/lib/python3.9/site-packages/seaborn/distribut
ions.py:2619: FutureWarning: `distplot` is a deprecated function and will
be removed in a future version. Please adapt your code to use either `dis
plot` (a figure-level function with similar flexibility) or `histplot` (a
n axes-level function for histograms).
  warnings.warn(msg, FutureWarning)
```



In [32]:
```
# Check if confidence intervals overlap
confidence_intervals_overlap = ci_married[1] >= ci_unmarried[0] and ci_un

# Print the result
if confidence_intervals_overlap:
    print("Confidence intervals of average married and unmarried spending
else:
    print("Confidence intervals of average married and unmarried spending

# insights and recommendations based on the comparison
print("\nInsights and Recommendations:")
if confidence_intervals_overlap:
    print("Since the CI of average spending for married and unmarried cus
          "there is no significant statistical evidence")
    print("Walmart can focus on normal marketing strategies that target a
else:
    print("The CI of average spending for married and unmarried customers
          "indicating a potential difference in spending habits between t
    print("Walmart may consider tailored marketing strategies and promoti
```

```
Confidence intervals of average married and unmarried spending overlap.

Insights and Recommendations:
Since the CI of average spending for married and unmarried customers over
lap, there is no significant statistical evidence
Walmart can focus on normal marketing strategies that target a wider audi
ence regardless of marital status.
```

General Insights:

By Gender:

The distribution of purchase amounts across the dataset is right-skewed.

The analysis comparing male and female spending showed no overlapping confidence intervals, implying that there is significant statistical difference in spending habits between genders. Walmart can devise marketing strategies specifically concentrating on genderwise plans.

By Marital Status:

Unlike gender, the analysis for married and unmarried customers resulted in overlapping confidence intervals. This suggests that there's no significant statistical difference in spending habits based on marital status there by implying there is no need of tailor-made marketing strategies in terms of customers maritial status.

CONCLUSION:

* Majority of customers are Men so Walmart can give more preference to catering thier needs or alternatively efforts can also be made to even the gender inequality in terms of customers.
* Males are spending more than thier counterparts in terms of average spend
* The CI of average spending for male and female customers do not overlap, indicating a potential difference in spending habits between genders. Walmart may consider tailored marketing strategies and promotions to target each gender specifically,
* the analysis for married and unmarried customers resulted in overlapping confidence intervals. This suggests that there's no significant statistical difference in spending habits based on marital status there by implying there is no need of tailor-made marketing strategies in terms of customers maritial status.