In [21]:
```
# importing python libraries

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

In [4]:
```
# importing csv file

df_sales = pd.read_csv('Sales Data.csv', encoding = 'latin1')
```

In [7]:
```
#to know the rows and columns
df_sales.head()
```

Out[7]:

	User_ID	Cust_name	Product_ID	Gender	Age Group	Age	Marital_Status	State	Zone	Occupation	Product_Category	Orders	Amount	Status	unnamed1
0	1002903	Sanskriti	P00125942	F	26-35	28	0	Maharashtra	Western	Healthcare	Auto	1	23952.0	NaN	NaN
1	1000732	Kartik	P00110942	F	26-35	35	1	Andhra Pradesh	Southern	Govt	Auto	3	23934.0	NaN	NaN
2	1001990	Bindu	P00118542	F	26-35	35	1	Uttar Pradesh	Central	Automobile	Auto	3	23924.0	NaN	NaN
3	1001425	Sudevi	P00237842	M	0-17	16	0	Karnataka	Southern	Construction	Auto	2	23912.0	NaN	NaN
4	1000588	Joni	P00057942	M	26-35	28	1	Gujarat	Western	Food Processing	Auto	2	23877.0	NaN	NaN

In [45]:
```
#to know the rows and columns
df_sales.shape
```

Out[45]:
```
(11251, 15)
```

In [47]:
```
#information about the DataFrame,data types, memory usage, range index,
df_sales.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 11251 entries, 0 to 11250
Data columns (total 15 columns):
 #   Column            Non-Null Count  Dtype  
---  ------            --------------  -----  
 0   User_ID           11251 non-null  int64  
 1   Cust_name         11251 non-null  object 
 2   Product_ID        11251 non-null  object 
 3   Gender            11251 non-null  object 
 4   Age Group         11251 non-null  object 
 5   Age               11251 non-null  int64  
 6   Marital_Status    11251 non-null  int64  
 7   State             11251 non-null  object 
 8   Zone              11251 non-null  object 
 9   Occupation        11251 non-null  object 
 10  Product_Category  11251 non-null  object 
 11  Orders            11251 non-null  int64  
 12  Amount            11239 non-null  float64
 13  Status            0 non-null      float64
 14  unnamed1          0 non-null      float64
dtypes: float64(3), int64(4), object(8)
memory usage: 1.3+ MB
```

In [9]:
```
##droping unwanted rows and columns because there is no data that gives insights
df_sales.drop(columns=["Status","unnamed1"],inplace = True)
```

In [10]:
```
#check for null values
df_sales.isnull().sum().sort_values(ascending = False)
```

```
Out[10]: Amount             12
         User_ID             0
         Cust_name           0
         Product_ID          0
         Gender              0
         Age Group           0
         Age                 0
         Marital_Status      0
         State               0
         Zone                0
         Occupation          0
         Product_Category    0
         Orders              0
         dtype: int64
```

```
In [11]: #initiating null values by taking column values
         mode_value=df_sales['Amount'].mode()[0]
```

```
In [12]: #knowing null values
         mode_value
```

```
Out[12]: 7907.0
```

```
In [13]: #filling null values
         df_sales['Amount'].fillna(mode_value,inplace=True)
```

```
In [14]: #rechecking it for null values
         df_sales.isnull().sum()
```

```
Out[14]: User_ID             0
         Cust_name           0
         Product_ID          0
         Gender              0
         Age Group           0
         Age                 0
         Marital_Status      0
         State               0
         Zone                0
         Occupation          0
         Product_Category    0
         Orders              0
         Amount              0
         dtype: int64
```

```
In [15]: #Changing the data types
         df_sales['Amount'] = df_sales['Amount'].astype('int')
```

```
In [16]: #Rechecking
         df_sales['Amount'].dtype
```

```
Out[16]: dtype('int64')
```

```
In [17]: #checking the columns of the data frame
         df_sales.columns
```

```
Out[17]: Index(['User_ID', 'Cust_name', 'Product_ID', 'Gender', 'Age Group', 'Age',
                'Marital_Status', 'State', 'Zone', 'Occupation', 'Product_Category',
                'Orders', 'Amount'],
               dtype='object')
```

```
In [18]: df_sales[["Age","Marital_Status","Orders","Amount"]].describe()
```

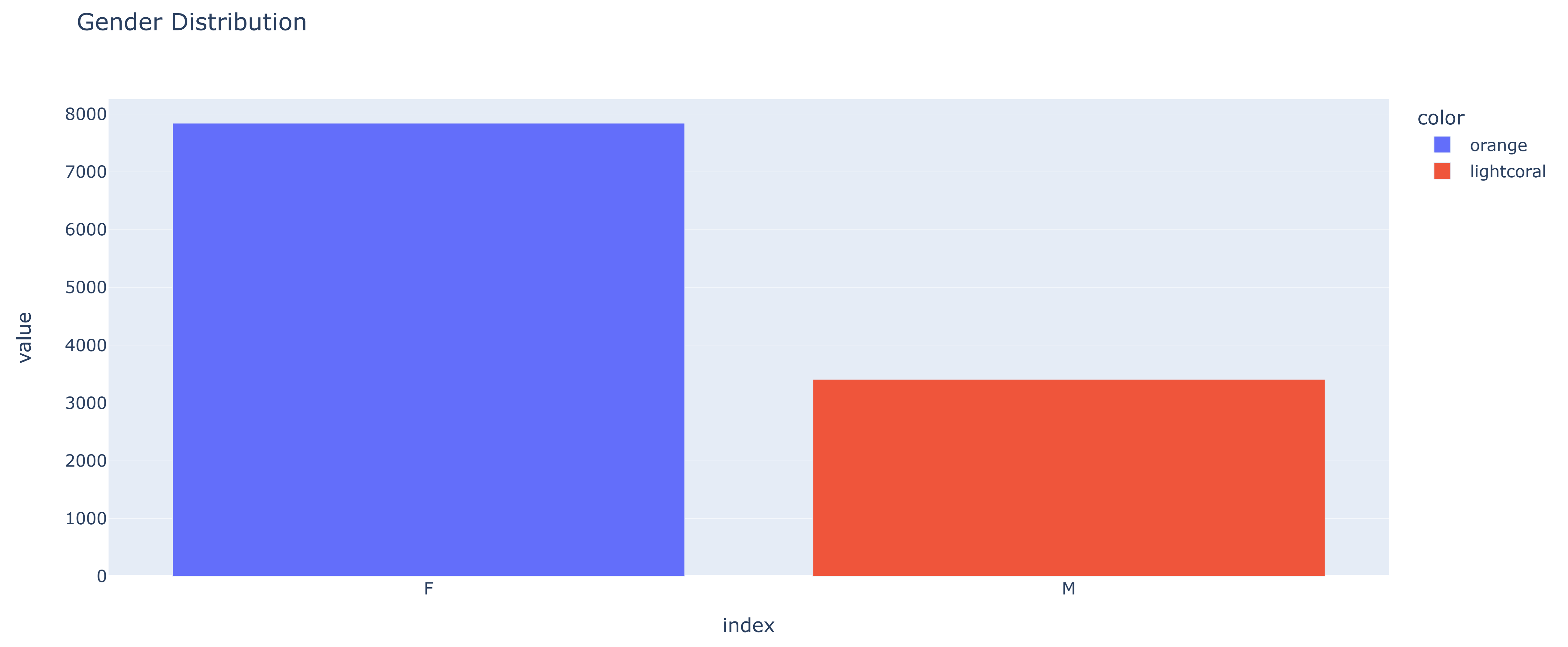
Out[18]:

	Age	Marital_Status	Orders	Amount
count	11251.000000	11251.000000	11251.000000	11251.000000
mean	35.421207	0.420318	2.489290	9451.960981
std	12.754122	0.493632	1.115047	5219.813316
min	12.000000	0.000000	1.000000	188.000000
25%	27.000000	0.000000	1.500000	5443.500000
50%	33.000000	0.000000	2.000000	8108.000000
75%	43.000000	1.000000	3.000000	12671.000000
max	92.000000	1.000000	4.000000	23952.000000

Exploratory data analysis

Visualization

In [22]:
```
## plotting a bar chart for gender
px.bar(df_sales["Gender"].value_counts(), color=['orange', 'lightcoral'], title="Gender Distribution")
```



In [160…
```
#Checking in the code
df_sales.groupby(['Gender'])["Amount"].sum()
```

Out[160]:
```
Gender
F    74414923
M    31929090
Name: Amount, dtype: int64
```

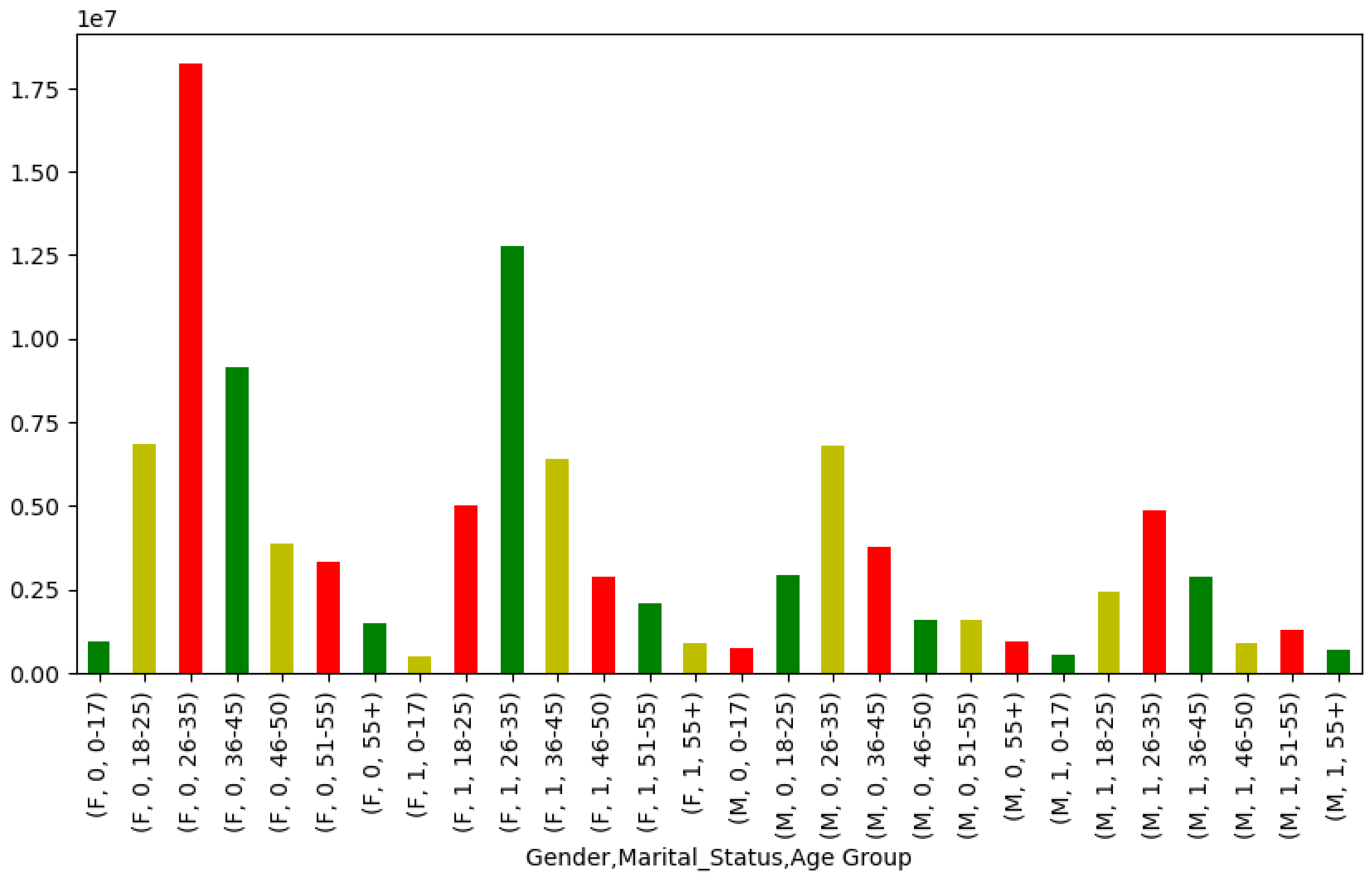
From above graphs we can see that most of the buyers are females

Age Analyse

In [28]:
```python
dfageplot = df_sales.groupby(['Gender','Marital_Status','Age Group'])['Amount'].sum()

dfageplot.plot(kind='bar',figsize = (10,5),color = ['g','y','red'])
```

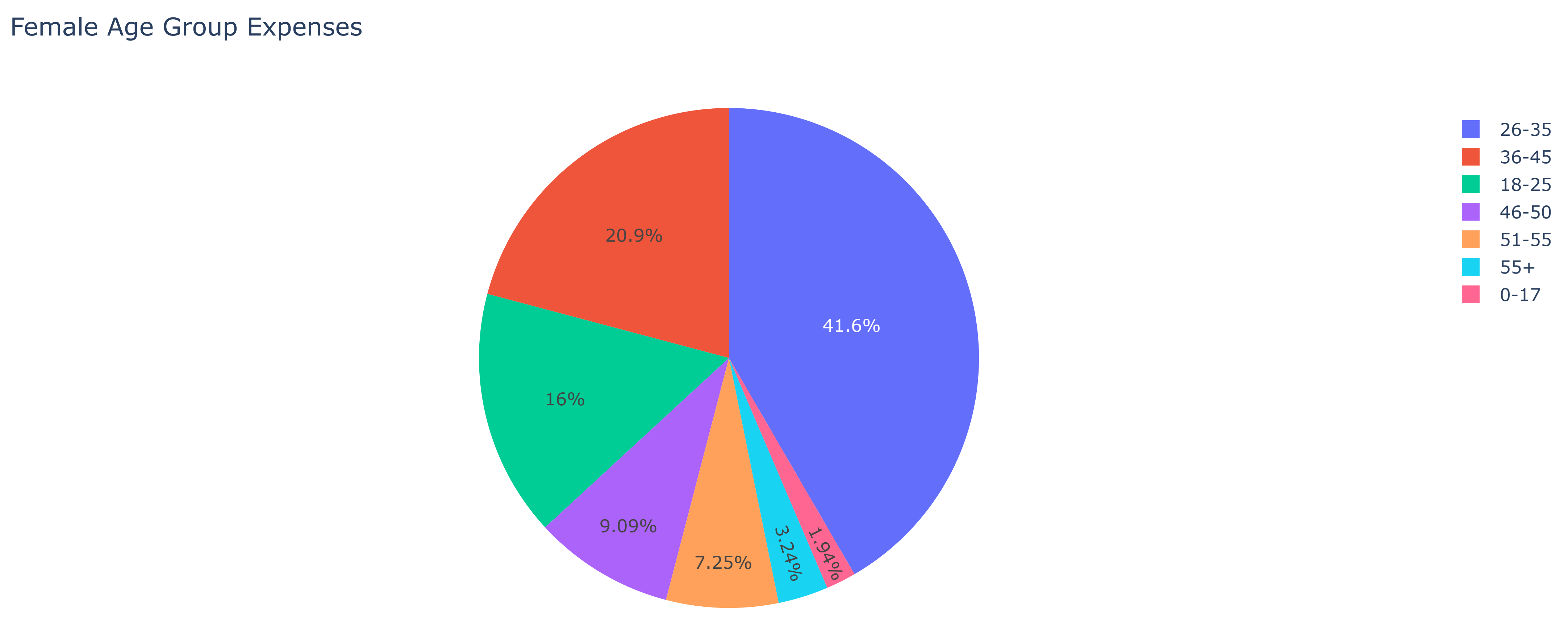
Out[28]:
```
<Axes: xlabel='Gender,Marital_Status,Age Group'>
```



In [29]:
```python
# Total Amount vs Age Group
femaleplot = df_sales[df_sales['Gender'].isin(['F'])].groupby(['Age Group'])['Amount'].sum().reset_index()

# Create a pie chart using Plotly Express
fig = px.pie(femaleplot, values='Amount', names='Age Group', title='Female Age Group Expenses')

# Show the plot
fig.show()
```



From above graphs we can see that most of the buyers are of age group B/W 26-35 year female

State

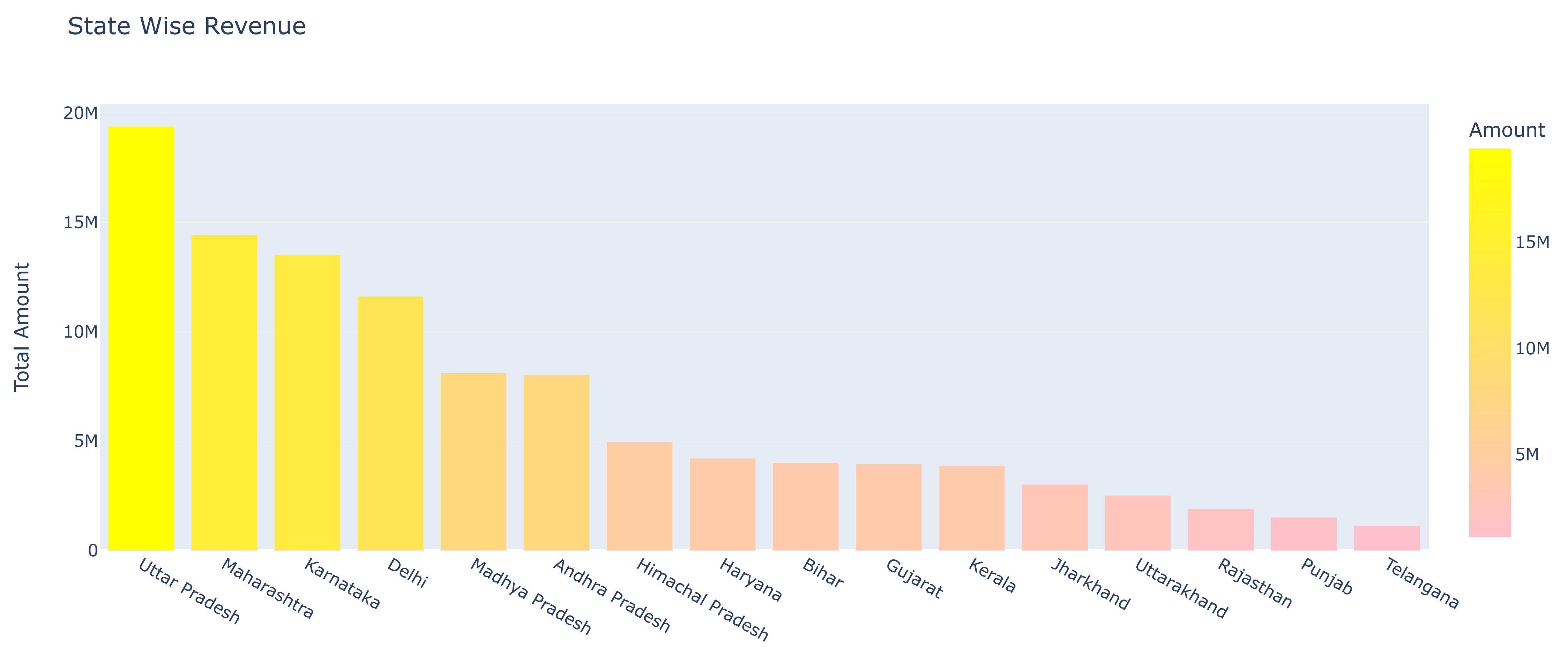
```
In [34]: # total amount of sales from states

stateplot = df_sales.groupby(['State'])['Amount'].sum().reset_index().sort_values(by='Amount', ascending=False)

# Create a bar chart using Plotly Express
fig = px.bar(stateplot, x='State', y='Amount', color='Amount',
             color_continuous_scale=['pink', 'yellow'],
             title='State Wise Revenue')

# Adjust the layout
fig.update_layout(xaxis_title="State", yaxis_title="Total Amount")

# Show the plot
fig.show()
```



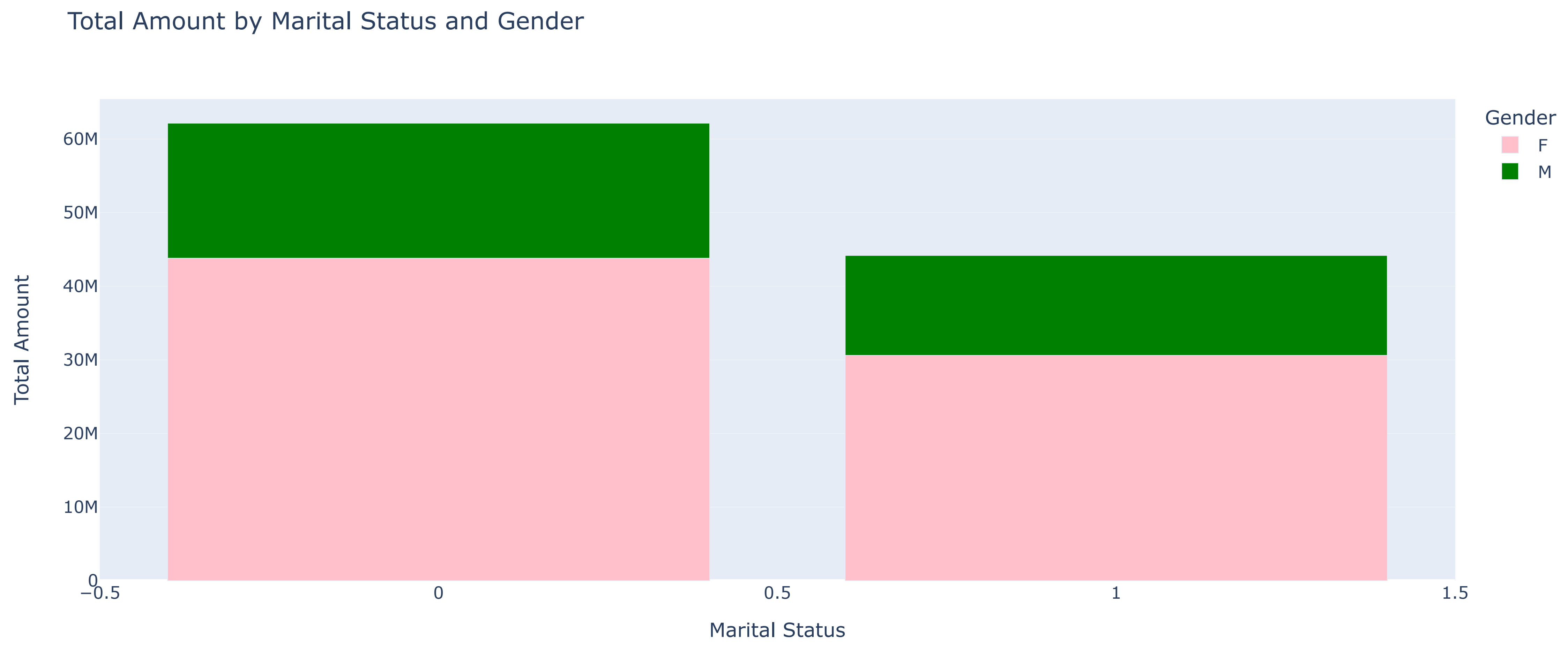
Marital Status

In [43]:
```
MaritalStatus = df_sales.groupby(["Marital_Status", 'Gender'])['Amount'].sum().reset_index()

# Create a bar chart using Plotly Express
fig = px.bar(MaritalStatus, x='Marital_Status', y='Amount', color='Gender',
             color_discrete_map={'M': 'green', 'F': 'pink'},
             title='Total Amount by Marital Status and Gender')

# Adjust the layout
fig.update_layout(xaxis_title="Marital Status", yaxis_title="Total Amount")

# Show the plot
fig.show()
```

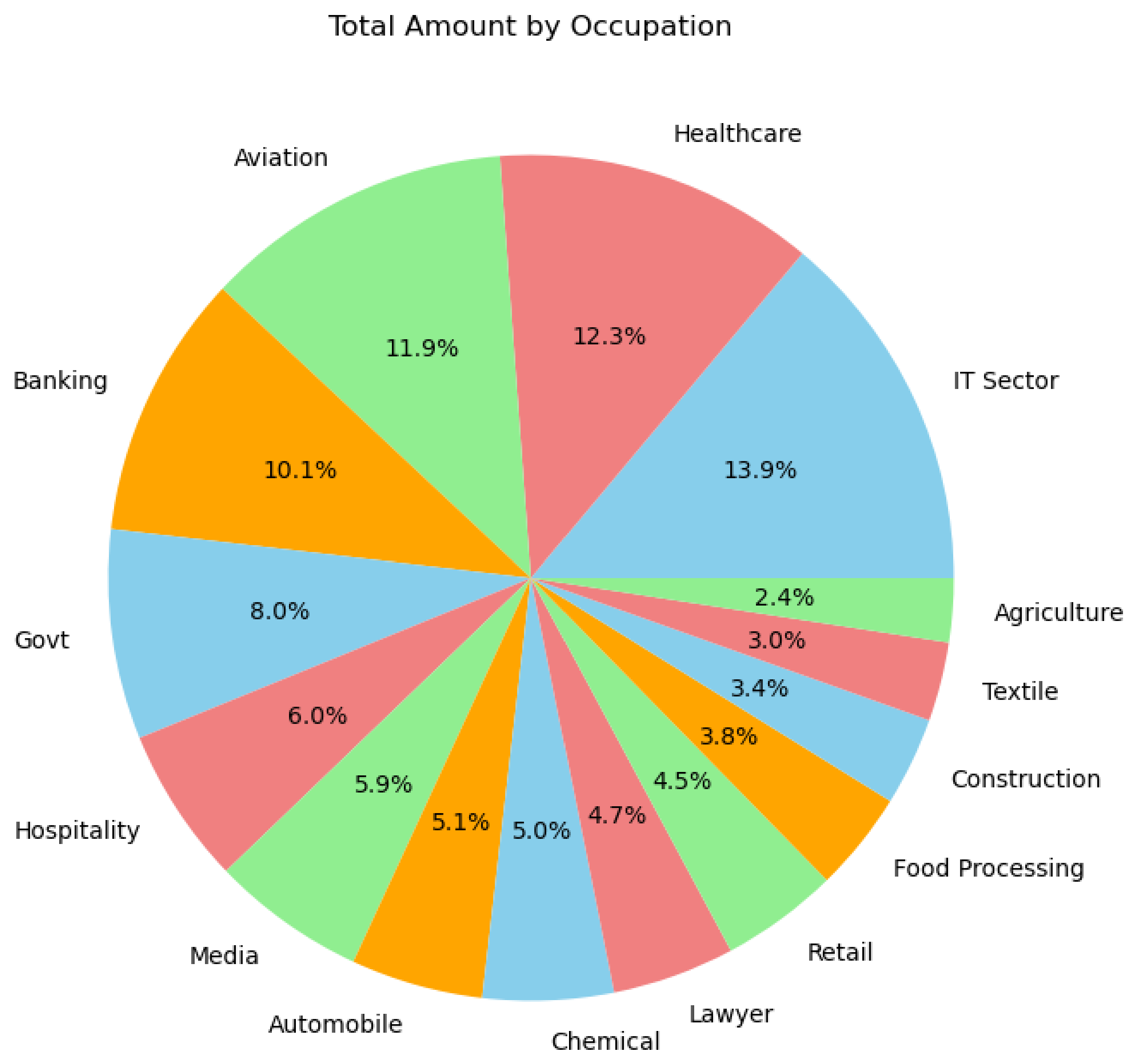


From above graphs we can see that most of the buyers are married (women) and they have high purchasing power

Occupation

In [46]:
```
Occupationplot = df_sales.groupby(['Occupation'])["Amount"].sum().sort_values(ascending=False)

# Create a pie chart using Matplotlib
plt.figure(figsize=(8, 8))
plt.pie(Occupationplot, labels=Occupationplot.index, autopct='%1.1f%%', colors=['skyblue', 'lightcoral', 'lightgreen', 'orange'])
plt.title('Total Amount by Occupation')
plt.show()
```



From above chart we can see that most of the buyers are working in IT, Healthcare and Aviation sector

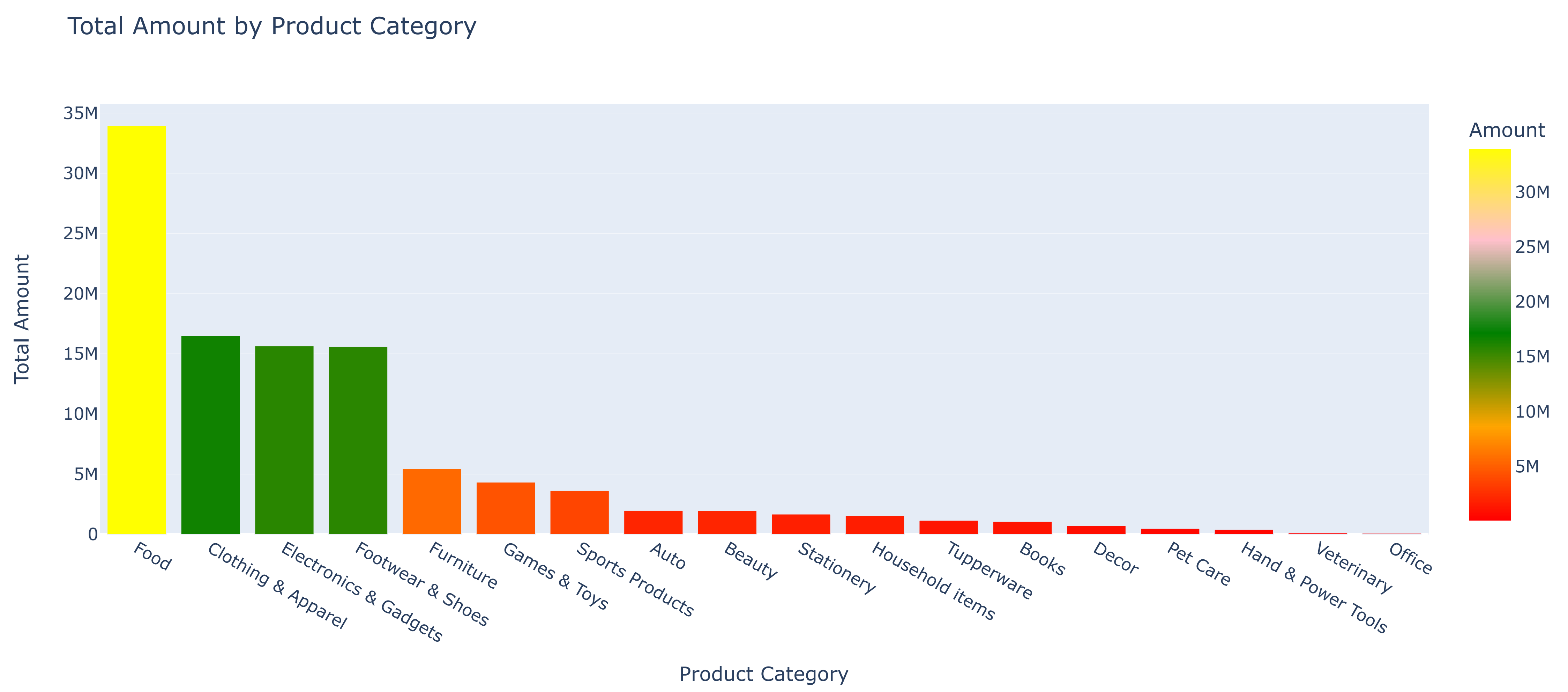
Product Category

In [51]:
```
ProductCategory = df_sales.groupby(["Product_Category"])['Amount'].sum().reset_index()

# Create a bar chart using Plotly Express
fig = px.bar(ProductCategory, x='Product_Category', y='Amount', color='Amount',
             color_continuous_scale=['red', 'orange', 'green', 'pink', 'yellow'],
             title='Total Amount by Product Category',
             category_orders={'Product_Category': ProductCategory.sort_values(by='Amount', ascending=False)['Product_Category']})

fig.update_layout(xaxis_title="Product Category", yaxis_title="Total Amount")

# Show the plot
fig.show()
```



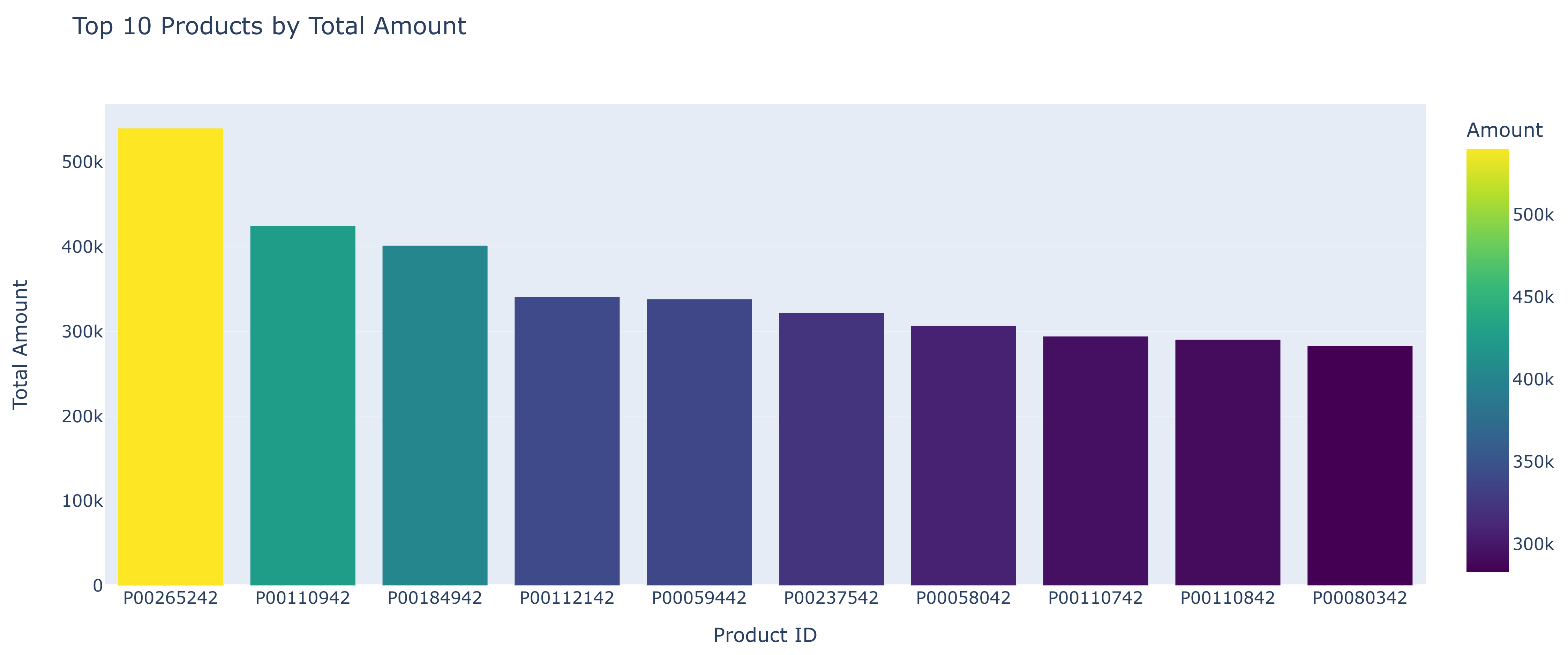
From above graphs we can see that most of the sold products are from Food, Clothing and Electronics category

```
In [56]: ProductIDplot = df_sales.groupby(['Product_ID'])['Amount'].sum().reset_index().sort_values(by='Amount', ascending=False).head(10)

         fig = px.bar(ProductIDplot, x='Product_ID', y='Amount', color='Amount',
                      color_continuous_scale='viridis',  # You can change this to any valid color scale
                      title='Top 10 Products by Total Amount')

         fig.update_layout(xaxis_title="Product ID", yaxis_title="Total Amount")

         fig.show()
```



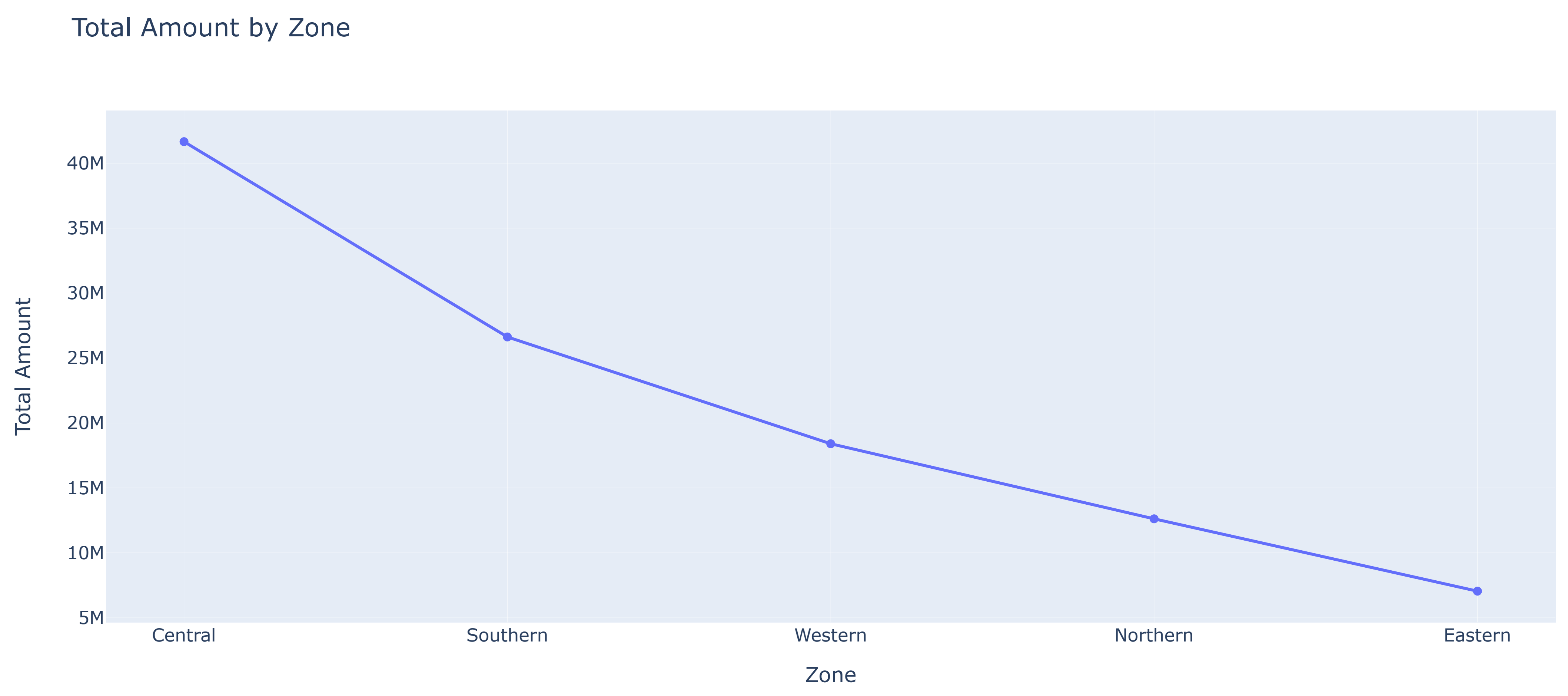
Zone

In [58]:
```
Zoneplot = df_sales.groupby(["Zone"])['Amount'].sum().reset_index().sort_values(by='Amount', ascending=False)

# Create a line plot using Plotly Express
fig = px.line(Zoneplot, x='Zone', y='Amount', markers=True, title='Total Amount by Zone')

# Adjust the layout
fig.update_layout(xaxis_title="Zone", yaxis_title="Total Amount")

# Show the plot
fig.show()
```



From above graphs we can see that most of the revenue is genrating from Central zone

Conclusion

Based on the analysis and visualizations of the sales data we have provided, here are some key conclusions and insights:

1] **Gender Analysis**: Most of the buyers are females. Females contribute significantly more to the total sales amount compared to males.

2] **Age Group Analysis**: The age group between 25-35 years appears to be the primary customer segment with the highest spending. Age group B/W 25-35 years contributes the most to the total sales amount among females.

3] **State Analysis**: Maharashtra is the top-performing state in terms of revenue. Southern and Western regions seem to have higher sales.

4] **Marital Status Analysis**: Married women have higher purchasing power compared to single women.

5] **Occupation Analysis**: Buyers working in IT, Healthcare, and Aviation sectors are the top contributors to sales.

6] **Product Category Analysis**: The most sold product categories are Food, Clothing, and Electronics.

7] **Product ID Analysis**: The top 10 most sold product IDs have been identified, which can help in focusing on popular products.

8] **Zone Analysis**: The Central zone generates the highest revenue.