In [3]:
```
!pip install numpy
```

```
Requirement already satisfied: numpy in c:\users\tanushri mune\anaconda3\lib
\site-packages (1.24.3)
```

In [2]:
```
!pip install pandas
```

```
Requirement already satisfied: pandas in c:\users\tanushri mune\anaconda3\lib
\site-packages (1.5.3)
Requirement already satisfied: python-dateutil>=2.8.1 in c:\users\tanushri mu
ne\anaconda3\lib\site-packages (from pandas) (2.8.2)
Requirement already satisfied: pytz>=2020.1 in c:\users\tanushri mune\anacond
a3\lib\site-packages (from pandas) (2022.7)
Requirement already satisfied: numpy>=1.21.0 in c:\users\tanushri mune\anacon
da3\lib\site-packages (from pandas) (1.24.3)
Requirement already satisfied: six>=1.5 in c:\users\tanushri mune\anaconda3\l
ib\site-packages (from python-dateutil>=2.8.1->pandas) (1.16.0)
```

In [3]:
```
!pip install matplotlib
```

```
Requirement already satisfied: matplotlib in c:\users\tanushri mune\anaconda3
\lib\site-packages (3.7.1)
Requirement already satisfied: contourpy>=1.0.1 in c:\users\tanushri mune\ana
conda3\lib\site-packages (from matplotlib) (1.0.5)
Requirement already satisfied: cycler>=0.10 in c:\users\tanushri mune\anacond
a3\lib\site-packages (from matplotlib) (0.11.0)
Requirement already satisfied: fonttools>=4.22.0 in c:\users\tanushri mune\an
aconda3\lib\site-packages (from matplotlib) (4.25.0)
Requirement already satisfied: kiwisolver>=1.0.1 in c:\users\tanushri mune\an
aconda3\lib\site-packages (from matplotlib) (1.4.4)
Requirement already satisfied: numpy>=1.20 in c:\users\tanushri mune\anaconda
3\lib\site-packages (from matplotlib) (1.24.3)
Requirement already satisfied: packaging>=20.0 in c:\users\tanushri mune\anac
onda3\lib\site-packages (from matplotlib) (23.0)
Requirement already satisfied: pillow>=6.2.0 in c:\users\tanushri mune\anacon
da3\lib\site-packages (from matplotlib) (9.4.0)
Requirement already satisfied: pyparsing>=2.3.1 in c:\users\tanushri mune\ana
conda3\lib\site-packages (from matplotlib) (3.0.9)
Requirement already satisfied: python-dateutil>=2.7 in c:\users\tanushri mune
\anaconda3\lib\site-packages (from matplotlib) (2.8.2)
Requirement already satisfied: six>=1.5 in c:\users\tanushri mune\anaconda3\l
ib\site-packages (from python-dateutil>=2.7->matplotlib) (1.16.0)
```

In [4]:
```
!pip install seaborn
```

```
Requirement already satisfied: seaborn in c:\users\tanushri mune\anaconda3\li
b\site-packages (0.12.2)
Requirement already satisfied: numpy!=1.24.0,>=1.17 in c:\users\tanushri mune
\anaconda3\lib\site-packages (from seaborn) (1.24.3)
Requirement already satisfied: pandas>=0.25 in c:\users\tanushri mune\anacond
a3\lib\site-packages (from seaborn) (1.5.3)
Requirement already satisfied: matplotlib!=3.6.1,>=3.1 in c:\users\tanushri m
une\anaconda3\lib\site-packages (from seaborn) (3.7.1)
Requirement already satisfied: contourpy>=1.0.1 in c:\users\tanushri mune\ana
conda3\lib\site-packages (from matplotlib!=3.6.1,>=3.1->seaborn) (1.0.5)
Requirement already satisfied: cycler>=0.10 in c:\users\tanushri mune\anacond
a3\lib\site-packages (from matplotlib!=3.6.1,>=3.1->seaborn) (0.11.0)
Requirement already satisfied: fonttools>=4.22.0 in c:\users\tanushri mune\an
aconda3\lib\site-packages (from matplotlib!=3.6.1,>=3.1->seaborn) (4.25.0)
Requirement already satisfied: kiwisolver>=1.0.1 in c:\users\tanushri mune\an
aconda3\lib\site-packages (from matplotlib!=3.6.1,>=3.1->seaborn) (1.4.4)
Requirement already satisfied: packaging>=20.0 in c:\users\tanushri mune\anac
onda3\lib\site-packages (from matplotlib!=3.6.1,>=3.1->seaborn) (23.0)
Requirement already satisfied: pillow>=6.2.0 in c:\users\tanushri mune\anacon
da3\lib\site-packages (from matplotlib!=3.6.1,>=3.1->seaborn) (9.4.0)
Requirement already satisfied: pyparsing>=2.3.1 in c:\users\tanushri mune\ana
conda3\lib\site-packages (from matplotlib!=3.6.1,>=3.1->seaborn) (3.0.9)
Requirement already satisfied: python-dateutil>=2.7 in c:\users\tanushri mune
\anaconda3\lib\site-packages (from matplotlib!=3.6.1,>=3.1->seaborn) (2.8.2)
Requirement already satisfied: pytz>=2020.1 in c:\users\tanushri mune\anacond
a3\lib\site-packages (from pandas>=0.25->seaborn) (2022.7)
Requirement already satisfied: six>=1.5 in c:\users\tanushri mune\anaconda3\l
ib\site-packages (from python-dateutil>=2.7->matplotlib!=3.6.1,>=3.1->seabor
n) (1.16.0)
```

In [5]:
```
pip show numpy
```

```
Name: numpy
Version: 1.24.3
Summary: Fundamental package for array computing in Python
Home-page: https://www.numpy.org (https://www.numpy.org)
Author: Travis E. Oliphant et al.
Author-email:
License: BSD-3-Clause
Location: C:\Users\Tanushri Mune\anaconda3\Lib\site-packages
Requires:
Required-by: astropy, bokeh, Bottleneck, contourpy, daal4py, datashader, data
shape, gensim, h5py, holoviews, hvplot, imagecodecs, imageio, imbalanced-lear
n, matplotlib, mkl-fft, mkl-random, numba, numexpr, pandas, patsy, pyarrow, p
yerfa, PyWavelets, scikit-image, scikit-learn, scipy, seaborn, statsmodels, t
ables, tifffile, transformers, xarray
Note: you may need to restart the kernel to use updated packages.
```

In [6]:
```
pip show pandas
```

```
Name: pandas
Version: 1.5.3
Summary: Powerful data structures for data analysis, time series, and statist
ics
Home-page: https://pandas.pydata.org (https://pandas.pydata.org)
Author: The Pandas Development Team
Author-email: pandas-dev@python.org
License: BSD-3-Clause
Location: C:\Users\Tanushri Mune\anaconda3\Lib\site-packages
Requires: numpy, numpy, python-dateutil, pytz
Required-by: bokeh, datashader, holoviews, hvplot, panel, seaborn, statsmodel
s, xarray
Note: you may need to restart the kernel to use updated packages.
```

In [7]:
```
pip show seaborn
```

```
Name: seaborn
Version: 0.12.2
Summary: Statistical data visualization
Home-page:
Author:
Author-email: Michael Waskom <mwaskom@gmail.com>
License:
Location: C:\Users\Tanushri Mune\anaconda3\Lib\site-packages
Requires: matplotlib, numpy, pandas
Required-by:
Note: you may need to restart the kernel to use updated packages.
```

In [8]:
```
pip show matplotlib
```

```
Name: matplotlib
Version: 3.7.1
Summary: Python plotting package
Home-page: https://matplotlib.org (https://matplotlib.org)
Author: John D. Hunter, Michael Droettboom
Author-email: matplotlib-users@python.org
License: PSF
Location: C:\Users\Tanushri Mune\anaconda3\Lib\site-packages
Requires: contourpy, cycler, fonttools, kiwisolver, numpy, packaging, pillow,
pyparsing, python-dateutil
Required-by: seaborn
Note: you may need to restart the kernel to use updated packages.
```

In [37]:
```
import numpy as np
import pandas as pd
import matplotlib as plt
import seaborn as sns
```

In [11]:
```
df = pd.read_csv(r'C:\Users\Tanushri Mune\Downloads\Python_Diwali_Sales_Analys
```

In [12]:
```
df.shape
```

Out[12]:
```
(11251, 15)
```

In [13]:
```
df.head()
```

Out[13]:

	User_ID	Cust_name	Product_ID	Gender	Age Group	Age	Marital_Status	State	Zo
0	1002903	Sanskriti	P00125942	F	26-35	28	0	Maharashtra	Weste
1	1000732	Kartik	P00110942	F	26-35	35	1	Andhra Pradesh	Southe
2	1001990	Bindu	P00118542	F	26-35	35	1	Uttar Pradesh	Cent
3	1001425	Sudevi	P00237842	M	0-17	16	0	Karnataka	Southe
4	1000588	Joni	P00057942	M	26-35	28	1	Gujarat	Weste

In [14]:
```
df.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 [15]:
```
df.drop(['Status','unnamed1'],axis=1, inplace=True)
```

In [19]:
```python
pd.isnull(df)
```

Out[19]:

	User_ID	Cust_name	Product_ID	Gender	Age Group	Age	Marital_Status	State	Zone	Occ
0	False	False	False	False	False	False	False	False	False	
1	False	False	False	False	False	False	False	False	False	
2	False	False	False	False	False	False	False	False	False	
3	False	False	False	False	False	False	False	False	False	
4	False	False	False	False	False	False	False	False	False	
...	...	...	...	...	...	...	...	...	...	
11246	False	False	False	False	False	False	False	False	False	
11247	False	False	False	False	False	False	False	False	False	
11248	False	False	False	False	False	False	False	False	False	
11249	False	False	False	False	False	False	False	False	False	
11250	False	False	False	False	False	False	False	False	False	

11251 rows × 13 columns

In [21]:
```python
pd.isnull(df).sum()
```

Out[21]:
```
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              12
dtype: int64
```

In [22]:
```python
df.shape
```

Out[22]:
```
(11251, 13)
```

In [23]:
```python
df.dropna(inplace=True)
```

In [24]:
```python
df.shape
```

Out[24]:
```
(11239, 13)
```

In [25]:
```
df['Amount']=df['Amount'].astype('int')
```

In [26]:
```
df['Amount'].dtypes
```

Out[26]:
```
dtype('int32')
```

In [27]:
```
df.columns
```

Out[27]:
```
Index(['User_ID', 'Cust_name', 'Product_ID', 'Gender', 'Age Group', 'Age',
       'Marital_Status', 'State', 'Zone', 'Occupation', 'Product_Category',
       'Orders', 'Amount'],
      dtype='object')
```

In [29]:
```
df.rename(columns={'Marital_Status':'Shaadi'})
```

Out[29]:

	User_ID	Cust_name	Product_ID	Gender	Age Group	Age	Shaadi	State	Zone
0	1002903	Sanskriti	P00125942	F	26-35	28	0	Maharashtra	Westerr
1	1000732	Kartik	P00110942	F	26-35	35	1	Andhra Pradesh	Southerr
2	1001990	Bindu	P00118542	F	26-35	35	1	Uttar Pradesh	Centra
3	1001425	Sudevi	P00237842	M	0-17	16	0	Karnataka	Southerr
4	1000588	Joni	P00057942	M	26-35	28	1	Gujarat	Westerr
...	...	...	...	...	...	...	...	...	..
11246	1000695	Manning	P00296942	M	18-25	19	1	Maharashtra	Westerr
11247	1004089	Reichenbach	P00171342	M	26-35	33	0	Haryana	Northerr
11248	1001209	Oshin	P00201342	F	36-45	40	0	Madhya Pradesh	Centra
11249	1004023	Noonan	P00059442	M	36-45	37	0	Karnataka	Southerr
11250	1002744	Brumley	P00281742	F	18-25	19	0	Maharashtra	Westerr

11239 rows × 13 columns

In [30]:
```
df.describe()
```

Out[30]:

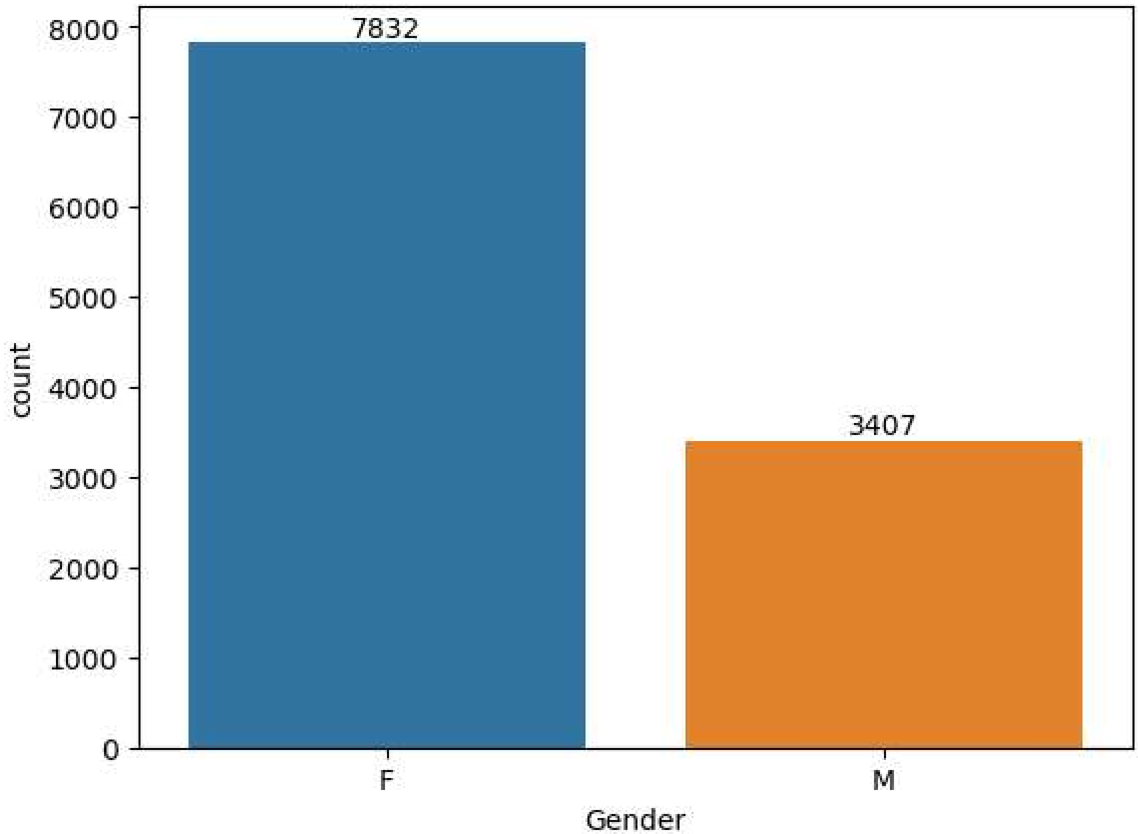
	User_ID	Age	Marital_Status	Orders	Amount
count	1.123900e+04	11239.000000	11239.000000	11239.000000	11239.000000
mean	1.003004e+06	35.410357	0.420055	2.489634	9453.610553
std	1.716039e+03	12.753866	0.493589	1.114967	5222.355168
min	1.000001e+06	12.000000	0.000000	1.000000	188.000000
25%	1.001492e+06	27.000000	0.000000	2.000000	5443.000000
50%	1.003064e+06	33.000000	0.000000	2.000000	8109.000000
75%	1.004426e+06	43.000000	1.000000	3.000000	12675.000000
max	1.006040e+06	92.000000	1.000000	4.000000	23952.000000

Gender:

In [32]:
```
df.columns
```

Out[32]:
```
Index(['User_ID', 'Cust_name', 'Product_ID', 'Gender', 'Age Group', 'Age',
       'Marital_Status', 'State', 'Zone', 'Occupation', 'Product_Category',
       'Orders', 'Amount'],
      dtype='object')
```

In [33]:
```
ax = sns.countplot(x = 'Gender',data = df)
for bars in ax.containers:
    ax.bar_label(bars)
```



The X-axis represents gender with two categories: 'F' for female and 'M' for male.

The Y-axis represents the shopping count, which is the number of shopping instances or items purchased.

The bar for 'F' reaches up to a count of 7832, indicating that females have a shopping count of 7832.

The bar for 'M' reaches up to a count of 3407, indicating that males have a shopping count of 3407.

From this chart, we can infer that females have a significantly higher shopping count than males according to the data represented.

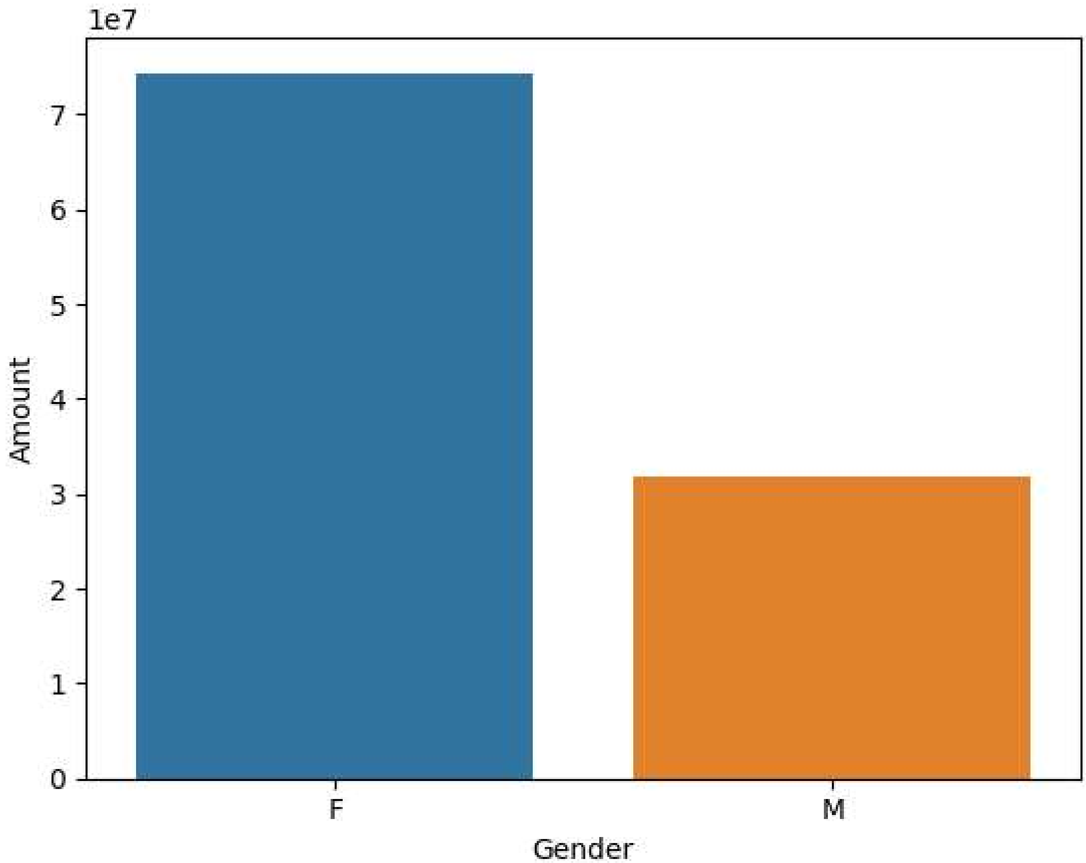
In [40]:
```
df.groupby(['Gender'],as_index= False)['Amount'].sum().sort_values(by='Amount
```

Out[40]:

	Gender	Amount
0	F	74335853
1	M	31913276

In [36]:
```
sales_gen = df.groupby(['Gender'],as_index= False)['Amount'].sum().sort_values
sns.barplot(x = 'Gender',y = 'Amount', data = sales_gen)
```

Out[36]: `<Axes: xlabel='Gender', ylabel='Amount'>`



The graphs indicate a predominant female buyer demographic, with a substantial count of 74,335,853. Furthermore, the purchasing power of females surpasses that of males.

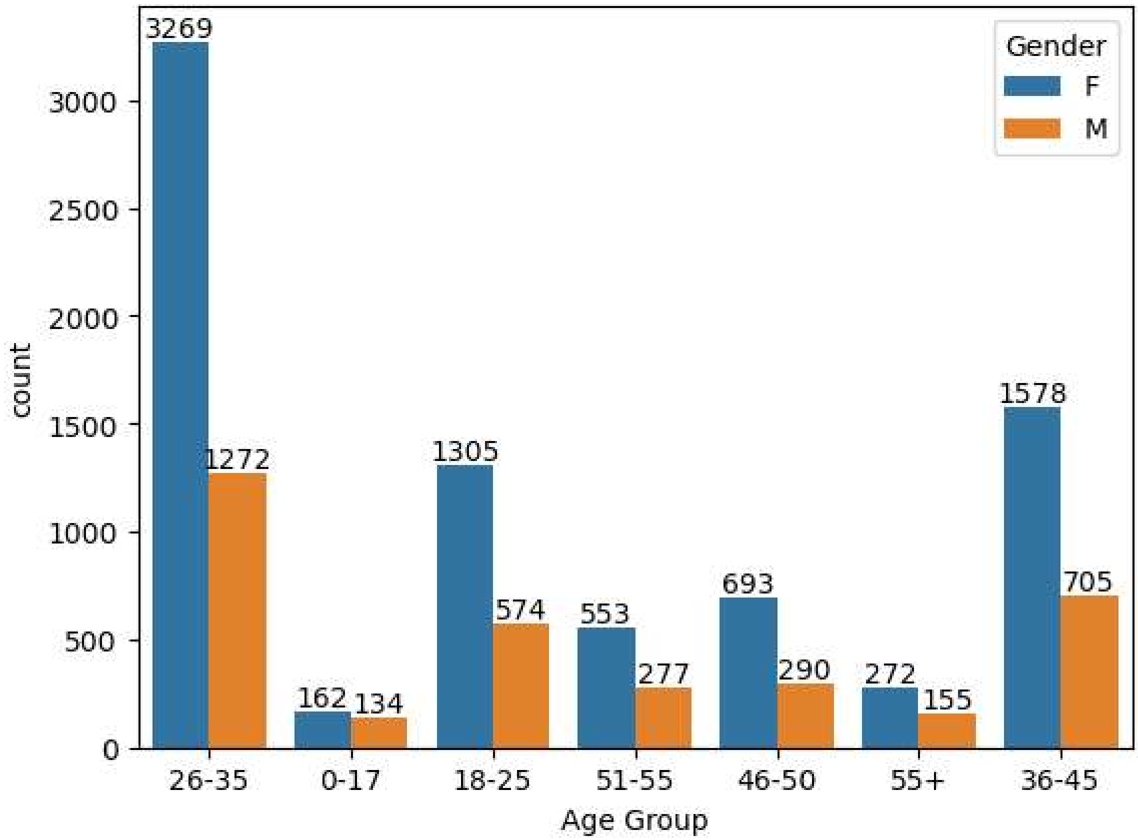
Age:

@Total Count Vs Age Group:

In [41]:
```
df.columns
```

Out[41]:
```
Index(['User_ID', 'Cust_name', 'Product_ID', 'Gender', 'Age Group', 'Age',
       'Marital_Status', 'State', 'Zone', 'Occupation', 'Product_Category',
       'Orders', 'Amount'],
      dtype='object')
```

In [45]:
```
ax = sns.countplot(data = df, x = 'Age Group', hue= 'Gender')
for bars in ax.containers:
    ax.bar_label(bars)
```



The 26-35 age group has the highest count for both categories, with the 'F' category (presumably 'Female') having a significantly higher count than the 'M' category (presumably 'Male').

The second-highest count for the 'F' category is in the 36-45 age group, while for the 'M' category, it is in the 18-25 age group.

The 0-17 age group has the lowest count for both categories.

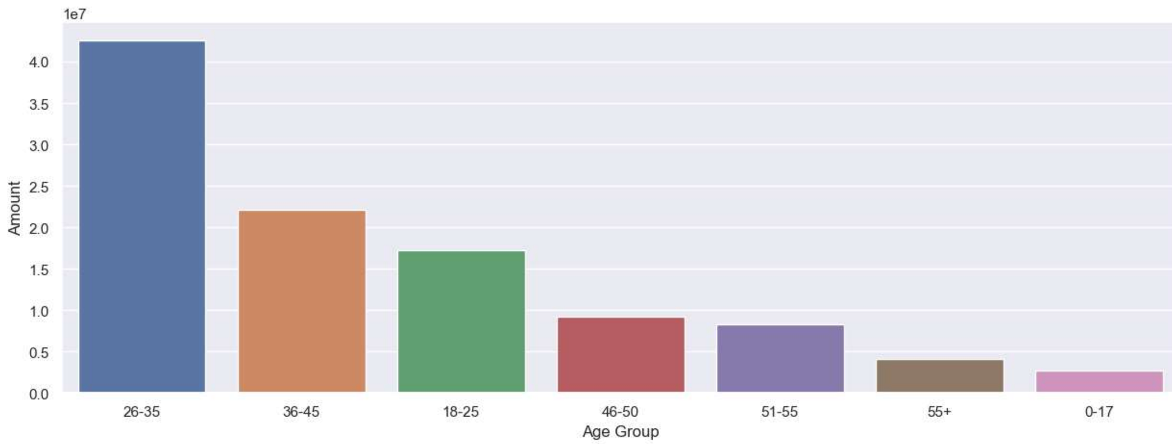
In all age groups except for 46-50, the 'F' category has a higher count than the 'M' category.

The 55+ age group has a similar count for both categories, with the 'M' category slightly higher.

@Total Amount Vs Age Group:

In [15]:
```
#Total Amount Vs Age Group
sales_age = df.groupby(['Age Group'],as_index = False)['Amount'].sum().sort_va
sns.barplot(x = 'Age Group', y = 'Amount',data = sales_age)
```

Out[15]: <Axes: xlabel='Age Group', ylabel='Amount'>



The following age group, 36-45, has roughly half the amount of the 26-35 Female group.

The 18-25 age group has a slightly lower amount than the 36-45 group.

The amounts decrease with the increasing age groups, with the 46-50 and 51-55 groups showing similar amounts.

The 55+ and 0-17 age groups have the lowest amounts, with the 0-17 group having the smallest bar.

This chart provides a visual representation of the distribution of a certain amount (possibly sales or revenue) across different age groups. The data suggests that the most significant amount is associated with the 26-35 females age group, indicating that this age group might be the most active or profitable segment.

State :

In [48]:
```
df.columns
```

Out[48]:
```
Index(['User_ID', 'Cust_name', 'Product_ID', 'Gender', 'Age Group', 'Age',
       'Marital_Status', 'State', 'Zone', 'Occupation', 'Product_Category',
       'Orders', 'Amount'],
      dtype='object')
```

@Total Number of orders from 10 states

In [9]:
```
sales_state = df.groupby(['State'], as_index=False)['Orders'].sum().sort_value

sns.set(rc={'figure.figsize':(15,5)})
sns.barplot(data = sales_state, x = 'State',y= 'Orders')
```

...

Uttar Pradesh leads in orders, nearing 5000 on the y-axis. Maharashtra follows closely. Karnataka and Delhi have comparable, slightly lower order counts.

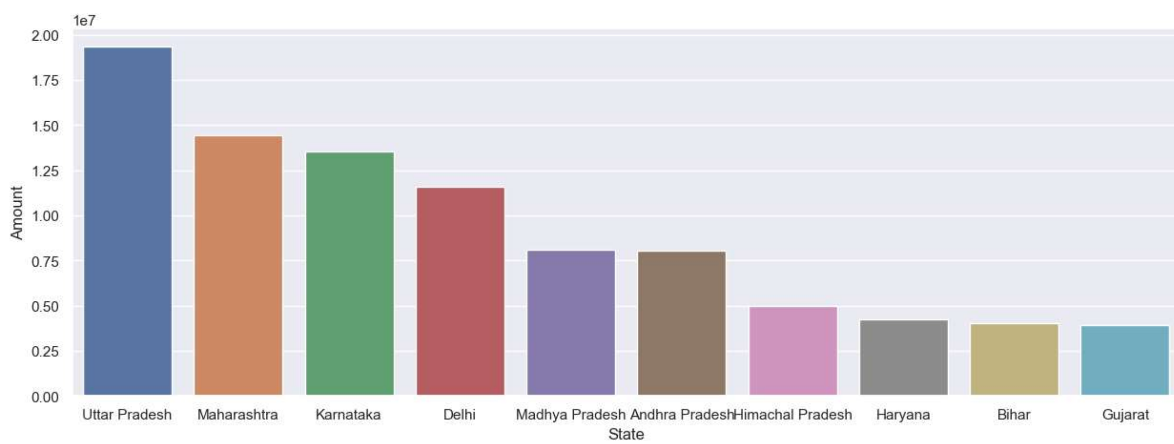
Madhya Pradesh, Andhra Pradesh, and Himachal Pradesh show moderate orders. Kerala, Haryana, and Gujarat have the fewest.

The x-axis denotes states, and y-axis shows order numbers. This chart aids quick state-wise order volume comparison, guiding marketing and operational decisions.

@ Total Amount/Sales from top 10 states

In [11]:
```
sales_state = df.groupby(['State'],as_index=False)['Amount'].sum().sort_values
sns.set(rc={'figure.figsize':(15,5)})
sns.barplot(data= sales_state,x = 'State',y='Amount')
```

Out[11]: <Axes: xlabel='State', ylabel='Amount'>



The bar chart depicts Uttar Pradesh with the highest amount, substantially exceeding other states. Maharashtra and Karnataka follow, with Maharashtra's amount slightly surpassing Karnataka's.

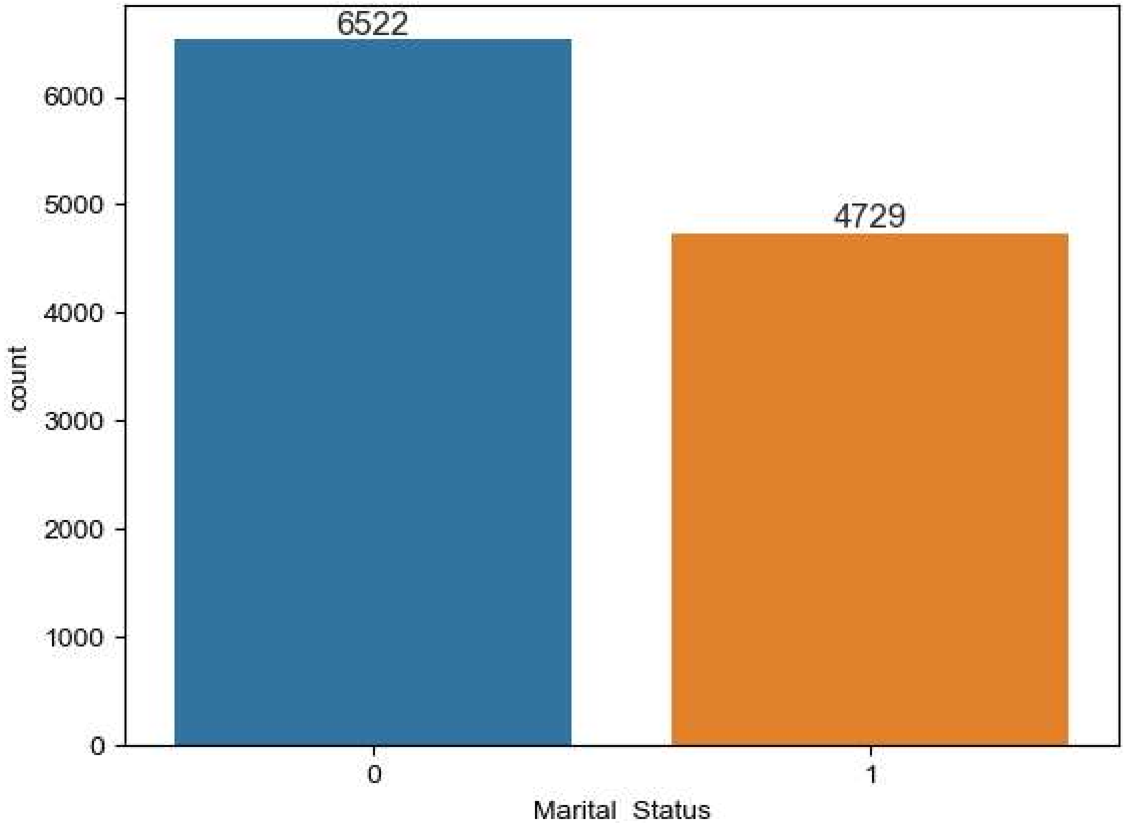
Delhi's amount is slightly lower than Karnataka's, while Madhya Pradesh and Andhra Pradesh have comparable amounts, both falling below Delhi's.

Marital status:

@Total Count Vs Marital Status

In [16]:
```
ax = sns.countplot(data = df, x = 'Marital_Status')

sns.set(rc={'figure.figsize':(7,5)})
for bars in ax.containers:
    ax.bar_label(bars)
```



The bar chart illustrates counts for a categorical variable labeled "Marital_Status" with two categories denoted as '0' and '1'. Category '0' exhibits a higher count of 6522, surpassing the count of category '1' which stands at 4729.

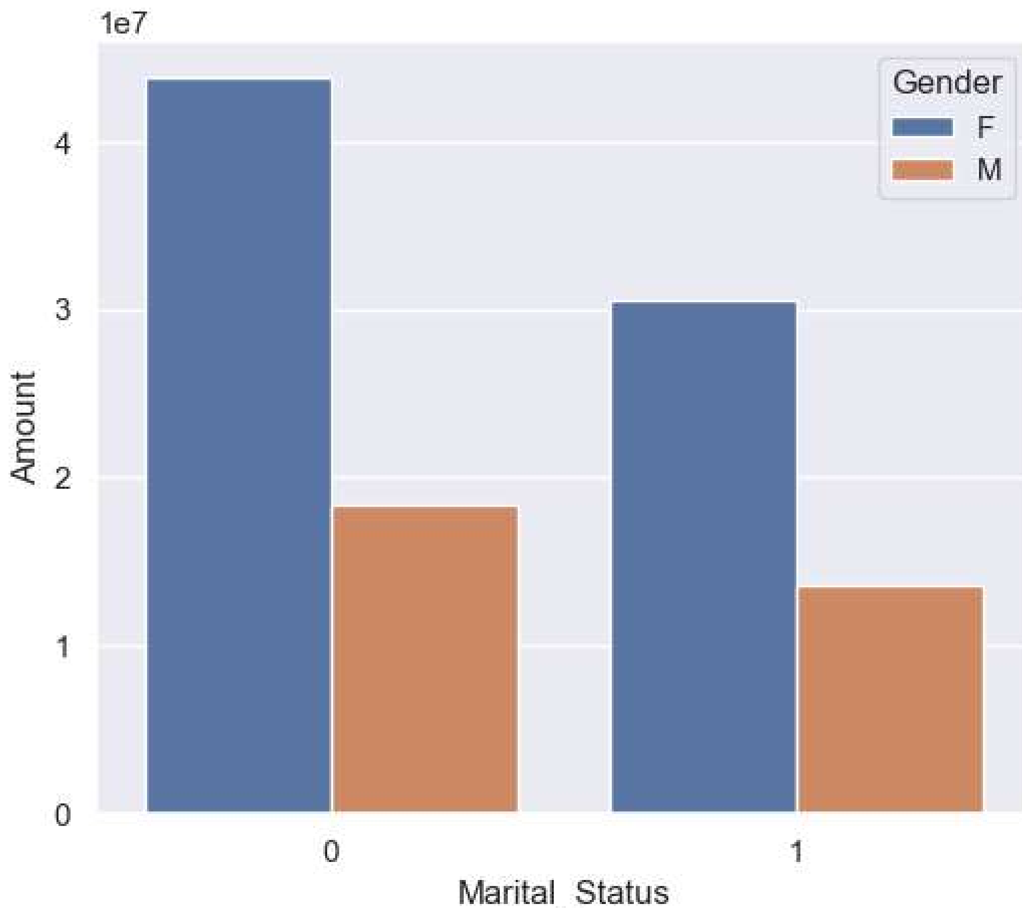
While the specific meanings of '0' and '1' are not explicitly defined in the chart, it is common in binary variables for '0' to represent 'Unmarried' and '1' to represent 'Married', or vice versa, depending on the coding scheme employed in the dataset.

The chart provides a straightforward overview of the distribution of marital status, indicating a higher prevalence of the category denoted by '0'.

@Total Amount Vs Marital Status

In [19]:
```
sales_state = df.groupby(['Marital_Status','Gender'],as_index=False)['Amount']
sns.set(rc={'figure.figsize':(6,5)})
sns.barplot(data= sales_state,x = 'Marital_Status',y='Amount',hue='Gender')
```

Out[19]: <Axes: xlabel='Marital_Status', ylabel='Amount'>



The bar chart illustrates data categorized by marital status and gender. Marital statuses '0' and '1' are present, potentially indicating 'single' and 'married'.

Two genders, 'F' for female and 'M' for male, are represented. The y-axis labeled 'Amount' implies counts of individuals. The tallest bar, for '0' marital status in males ('M'), has the highest count.

Females ('F') in both '0' and '1' marital statuses have lower counts than males ('M'). The gender count difference is more pronounced in the '0' marital status.

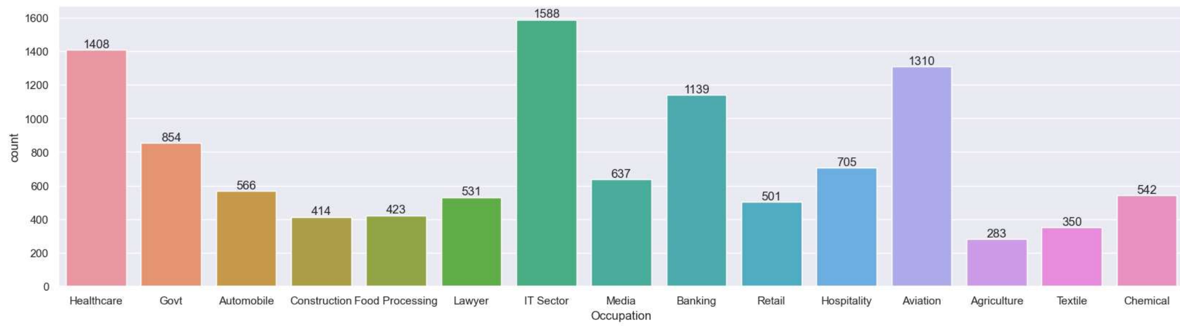
The y-axis scale suggests data in the tens of millions (1e7). Overall, the chart provides a high-level overview of population distribution in terms of marital status and gender.

Occupation:

Total Count Vs Occupation

In [20]:
```
sns.set(rc={'figure.figsize':(20,5)})
ax = sns.countplot(data = df, x='Occupation')

for bars in ax.containers:
    ax.bar_label(bars)
```



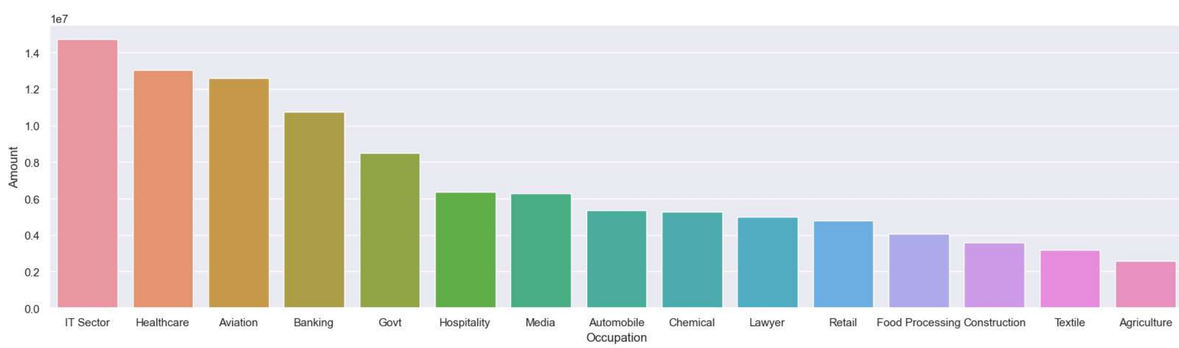
The bar chart depicts counts of various occupations across sectors. The IT sector leads with 1588, indicating its prominence in the dataset. Healthcare follows closely with 1408, and Aviation stands out with 1310. Sectors with the lowest counts include Agriculture (283), Textile (350), and Chemical (542). The Government sector is relatively high at 854, surpassing counts in Automobile (566), Construction (414), and Food Processing (423). Other notable counts include Media (637), Banking (1139), Retail (501), and Hospitality (705). These counts suggest the prevalence of occupations within each sector, possibly reflecting employee numbers or company representation.

Total Amount Vs Occupation

In [23]:
```
sales_state = df.groupby(['Occupation'], as_index=False)['Amount'].sum().sort_

sns.set(rc={'figure.figsize':(20,5)})
sns.barplot(data = sales_state, x = 'Occupation',y= 'Amount')
```

Out[23]: <Axes: xlabel='Occupation', ylabel='Amount'>



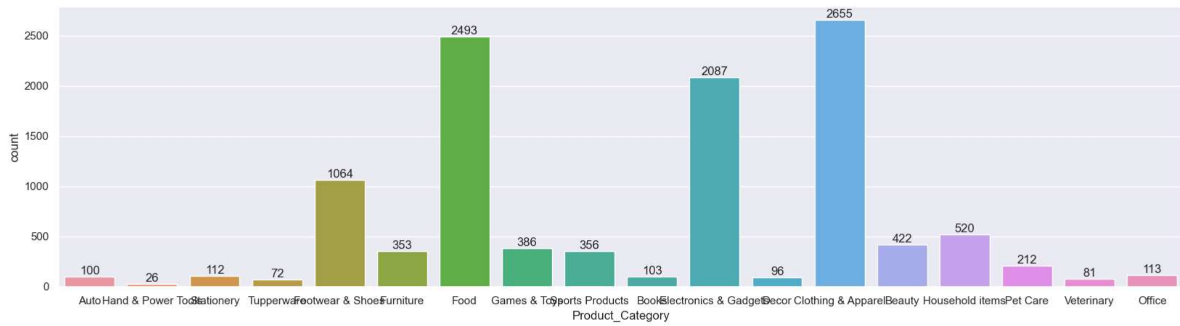
The bar chart clearly illustrates that the IT sector boasts the highest bar, signaling its substantial magnitude compared to other sectors. Conversely, the Agriculture sector exhibits the smallest bar, indicating its comparatively modest size in relation to the other sectors.

Product Category

@Total Count vs Product Category

In [25]:
```
sns.set(rc={'figure.figsize':(20,5)})
ax = sns.countplot(data = df, x = 'Product_Category')

for bars in ax.containers:
    ax.bar_label(bars)
```



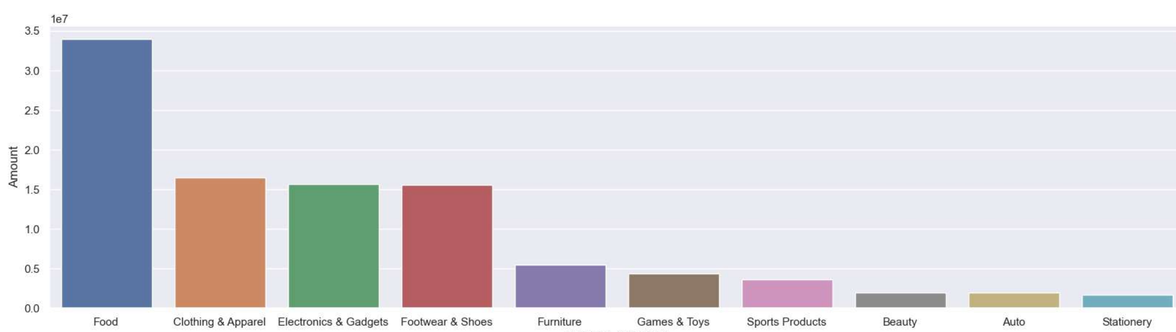
The bar chart highlights that Clothing & Apparel, Food, and Electronics & Gadgets are the categories with the highest counts, underscoring their popularity. Conversely, categories such as Hand & Power Tools, Decor, and Veterinary have the lowest counts, indicating lower engagement or demand in comparison.

@Total Amount Vs Product Category

In [26]:
```
sales_state = df.groupby(['Product_Category'], as_index=False)['Amount'].sum()

sns.set(rc={'figure.figsize':(20,5)})
sns.barplot(data = sales_state, x = 'Product_Category',y= 'Amount')
```

Out[26]: `<Axes: xlabel='Product_Category', ylabel='Amount'>`



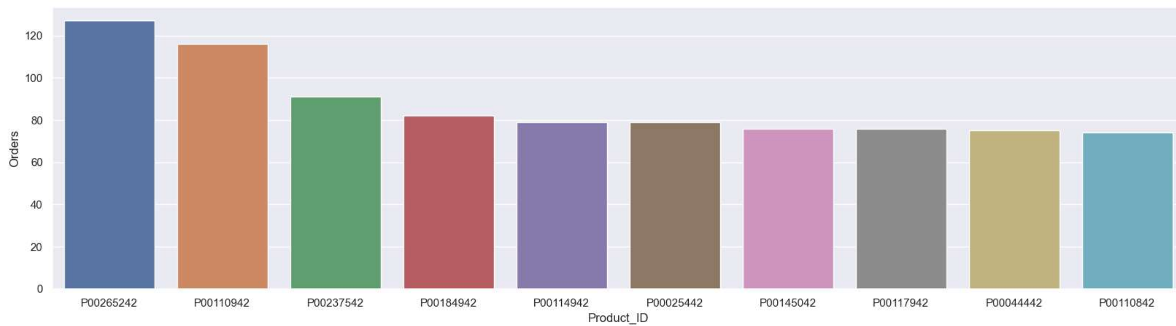
The bar chart highlights that Clothing & Apparel, Food, and Electronics & Gadgets are the categories with the highest counts, underscoring their popularity. Conversely, categories such as Hand & Power Tools, Decor, and Veterinary have the lowest counts, indicating lower engagement or demand in comparison.

@Product Id Vs Orders

In [27]:
```
sales_state = df.groupby(['Product_ID'], as_index=False)['Orders'].sum().sort_

sns.set(rc={'figure.figsize':(20,5)})
sns.barplot(data = sales_state, x = 'Product_ID',y= 'Orders')
```

Out[27]: <Axes: xlabel='Product_ID', ylabel='Orders'>



The bar chart reveals distinct order counts for different product IDs. Notably, product ID P00265242 stands out with the highest number of orders, surpassing 120. Conversely, product ID P00118442 has the lowest order count, hovering just below 60. The remaining products fall within this range, with none reaching the peak of P00265242 or the low of P00118442.

Conclusion :

In conclusion, the analysis reveals a predominant female buyer demographic, indicating higher purchasing power among women. The age group with the most buyers falls within 26-35 years, particularly among females. Geographically, the majority of orders and total sales originate from Uttar Pradesh, Maharashtra, and Karnataka. Married women emerge as the primary buyers with substantial purchasing influence. Additionally, buyers predominantly work in the IT, Healthcare, and Aviation sectors. The most sold product categories include Food, Clothing, and Electronics.