Diwali Sales Analysis

Wasay Ahmed Shaikh

Project 4

Kindly let me know your feedback & feel free to connect

Import Libraries

In [1]:
```
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
```

Import dataset

In [2]:
```
data = pd.read_csv('Diwali Sales Data.csv', encoding= 'unicode_escape')
```

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

Out[3]:

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

In [4]:
```
data.shape
```

Out[4]:
```
(11251, 15)
```

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

Out[5]:

	User_ID	Age	Marital_Status	Orders	Amount	Status	unnamed
count	1.125100e+04	11251.000000	11251.000000	11251.000000	11239.000000	0.0	0.
mean	1.003004e+06	35.421207	0.420318	2.489290	9453.610858	NaN	Na
std	1.716125e+03	12.754122	0.493632	1.115047	5222.355869	NaN	Na
min	1.000001e+06	12.000000	0.000000	1.000000	188.000000	NaN	Na
25%	1.001492e+06	27.000000	0.000000	1.500000	5443.000000	NaN	Na
50%	1.003065e+06	33.000000	0.000000	2.000000	8109.000000	NaN	Na
75%	1.004430e+06	43.000000	1.000000	3.000000	12675.000000	NaN	Na
max	1.006040e+06	92.000000	1.000000	4.000000	23952.000000	NaN	Na

In [6]:
```
data.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 [7]:
```
data.drop(['unnamed1','Status'], axis = 1, inplace = True)
```

In [8]:
```
pd.isnull(data).sum()
```

Out[8]:
```
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 [9]:
```
data.dropna(inplace = True)
```

In [10]:
```
data.rename(columns = {'Occupation': 'Services'})
```

Out[10]:

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

11239 rows × 13 columns

In [11]:
```
data.columns
```

Out[11]:
```
Index(['User_ID', 'Cust_name', 'Product_ID', 'Gender', 'Age Group', 'Age',
       'Marital_Status', 'State', 'Zone', 'Occupation', 'Product_Category',
       'Orders', 'Amount'],
      dtype='object')
```

In [12]:
```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 11239 entries, 0 to 11250
Data columns (total 13 columns):
 #   Column            Non-Null Count  Dtype  
---  ------            --------------  -----  
 0   User_ID           11239 non-null  int64  
 1   Cust_name         11239 non-null  object 
 2   Product_ID        11239 non-null  object 
 3   Gender            11239 non-null  object 
 4   Age Group         11239 non-null  object 
 5   Age               11239 non-null  int64  
 6   Marital_Status    11239 non-null  int64  
 7   State             11239 non-null  object 
 8   Zone              11239 non-null  object 
 9   Occupation        11239 non-null  object 
 10  Product_Category  11239 non-null  object 
 11  Orders            11239 non-null  int64  
 12  Amount            11239 non-null  float64
dtypes: float64(1), int64(4), object(8)
memory usage: 1.2+ MB
```

In [13]:
```
data['Amount'] = data['Amount'].astype('int')
```

In [14]:
```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 11239 entries, 0 to 11250
Data columns (total 13 columns):
 #   Column            Non-Null Count  Dtype  
---  ------            --------------  -----  
 0   User_ID           11239 non-null  int64  
 1   Cust_name         11239 non-null  object 
 2   Product_ID        11239 non-null  object 
 3   Gender            11239 non-null  object 
 4   Age Group         11239 non-null  object 
 5   Age               11239 non-null  int64  
 6   Marital_Status    11239 non-null  int64  
 7   State             11239 non-null  object 
 8   Zone              11239 non-null  object 
 9   Occupation        11239 non-null  object 
 10  Product_Category  11239 non-null  object 
 11  Orders            11239 non-null  int64  
 12  Amount            11239 non-null  int32  
dtypes: int32(1), int64(4), object(8)
memory usage: 1.2+ MB
```

In [15]:
```
data.rename(columns = {'Marital_Status': 'Married'}, inplace = 'True')
```

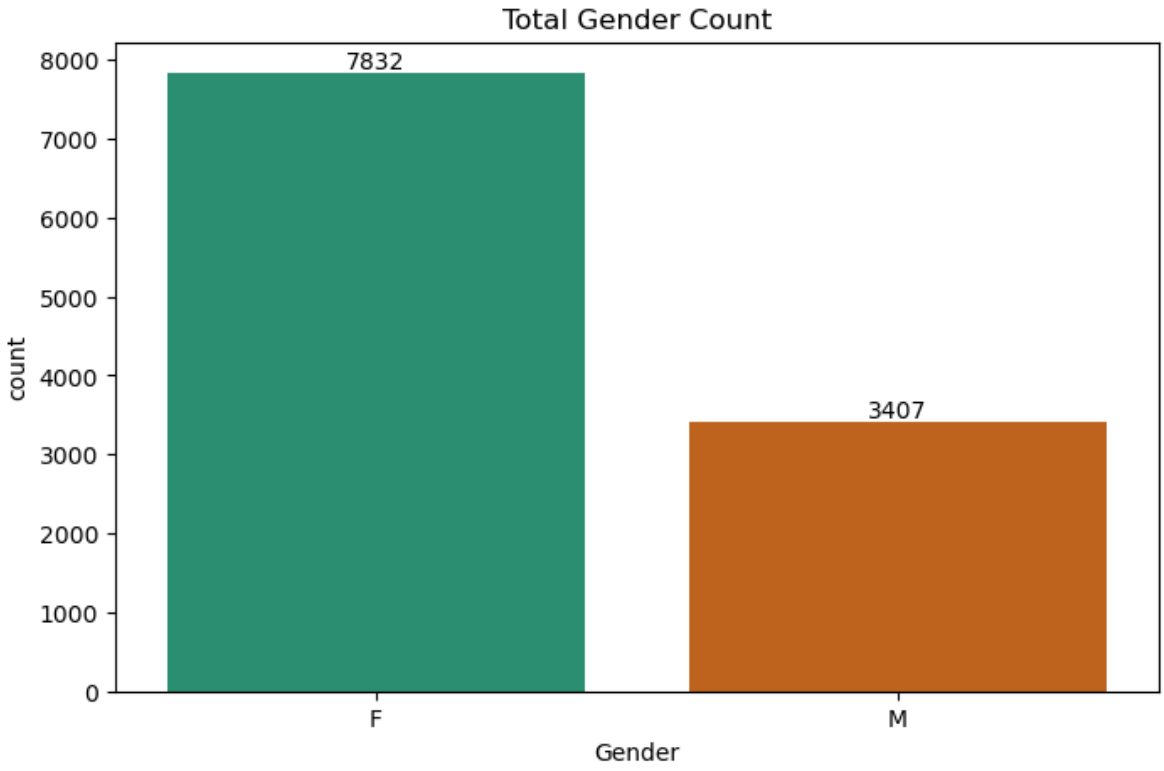
In [16]:
```
numarical = data[['Orders', 'Amount', 'Age']].describe()
numarical
```

Out[16]:

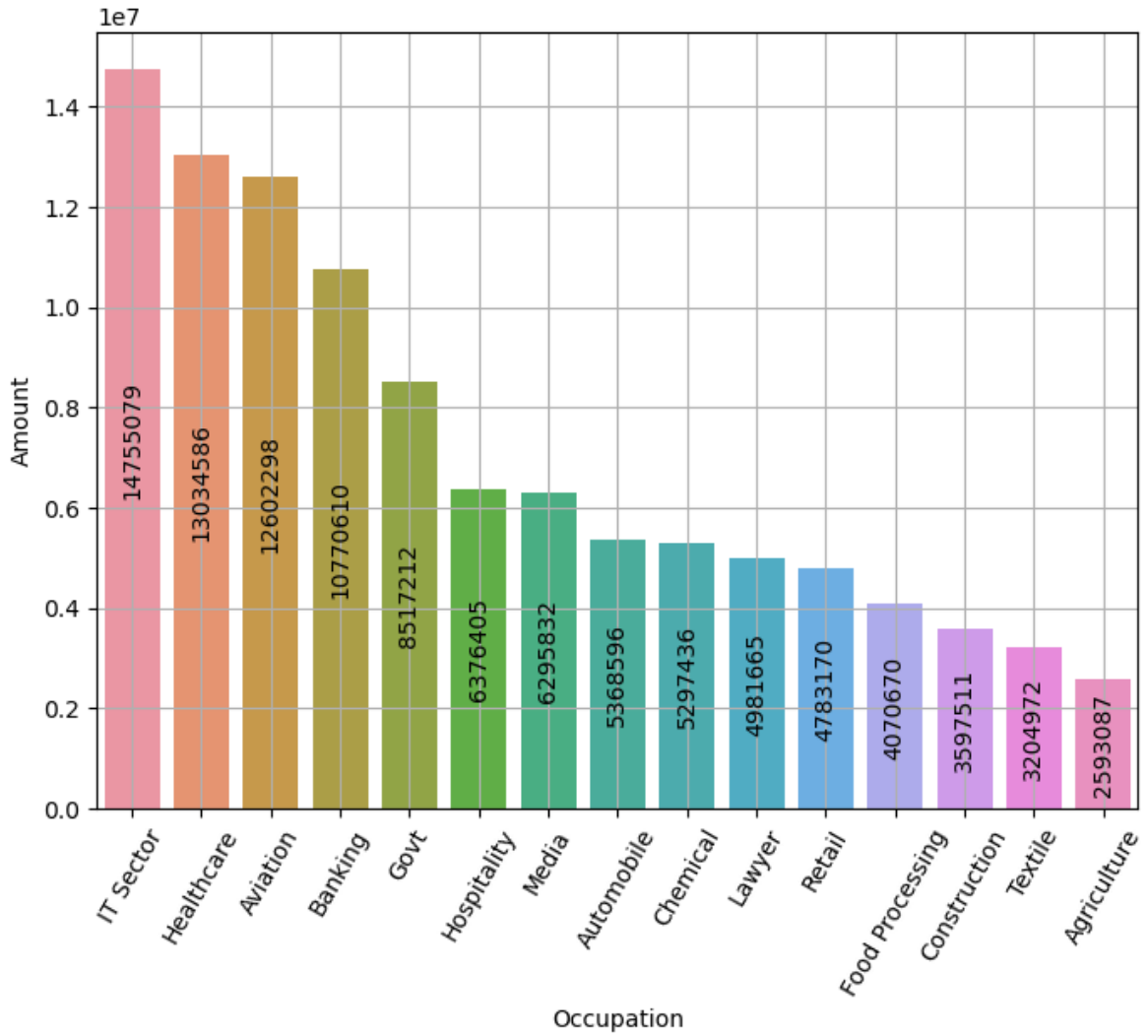
	Orders	Amount	Age
count	11239.000000	11239.000000	11239.000000
mean	2.489634	9453.610553	35.410357
std	1.114967	5222.355168	12.753866
min	1.000000	188.000000	12.000000
25%	2.000000	5443.000000	27.000000
50%	2.000000	8109.000000	33.000000
75%	3.000000	12675.000000	43.000000
max	4.000000	23952.000000	92.000000

In [17]:
```
#Import Libraries
plt.figure(figsize =(8,5))
ax= sns.countplot(x='Gender', data = data, palette='Dark2')
for bars in ax.containers:
    ax.bar_label(bars)

plt.title('Total Gender Count')
```

Out[17]:
```
Text(0.5, 1.0, 'Total Gender Count')
```



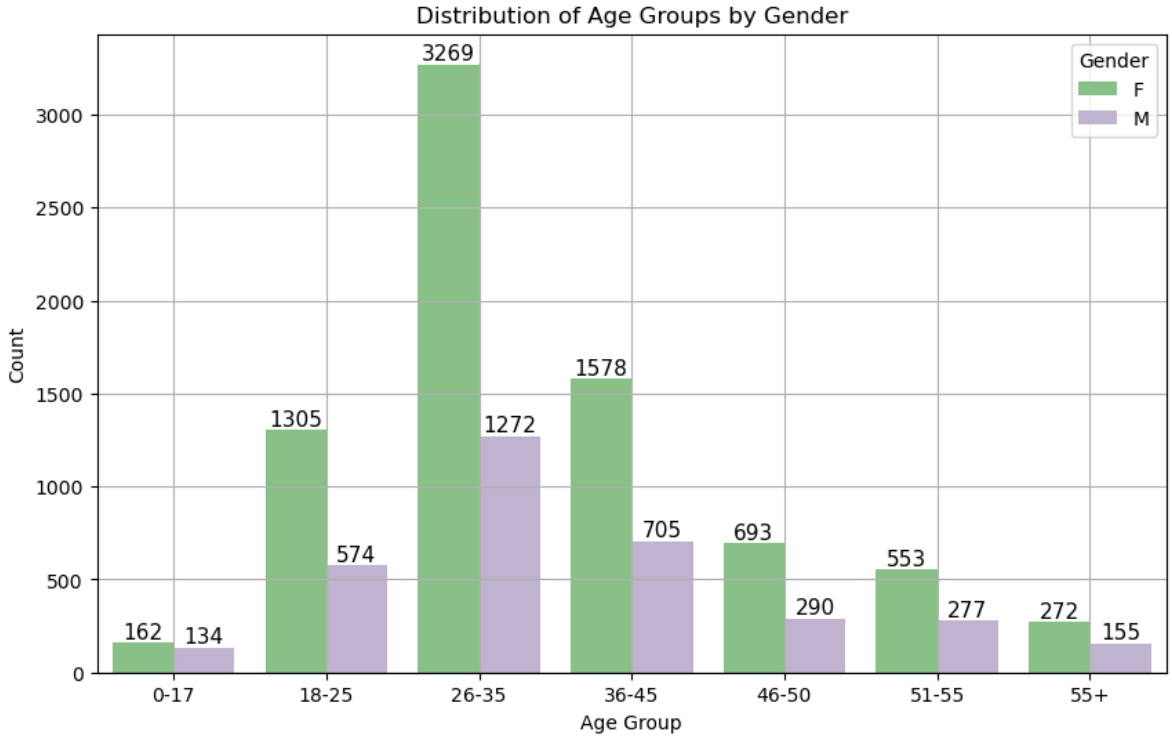
In [18]:
```
df = data.groupby('Occupation')['Amount'].sum().reset_index().sort_values('Amc

plt.figure(figsize=(8, 6))
bx = sns.barplot(x='Occupation', y='Amount', data=df)
plt.xticks(rotation=60, fontsize=10)

plt.grid(True)

for p in bx.patches:
    height = p.get_height()
    width = p.get_width()
    x, y = p.get_x(), p.get_y()

    label_x = x + width / 2
    label_y = y + height / 2

    bx.annotate(f'{int(height)}', (label_x, label_y), ha='center', va='center'

plt.show()
```



In [19]:
```
data.columns
```

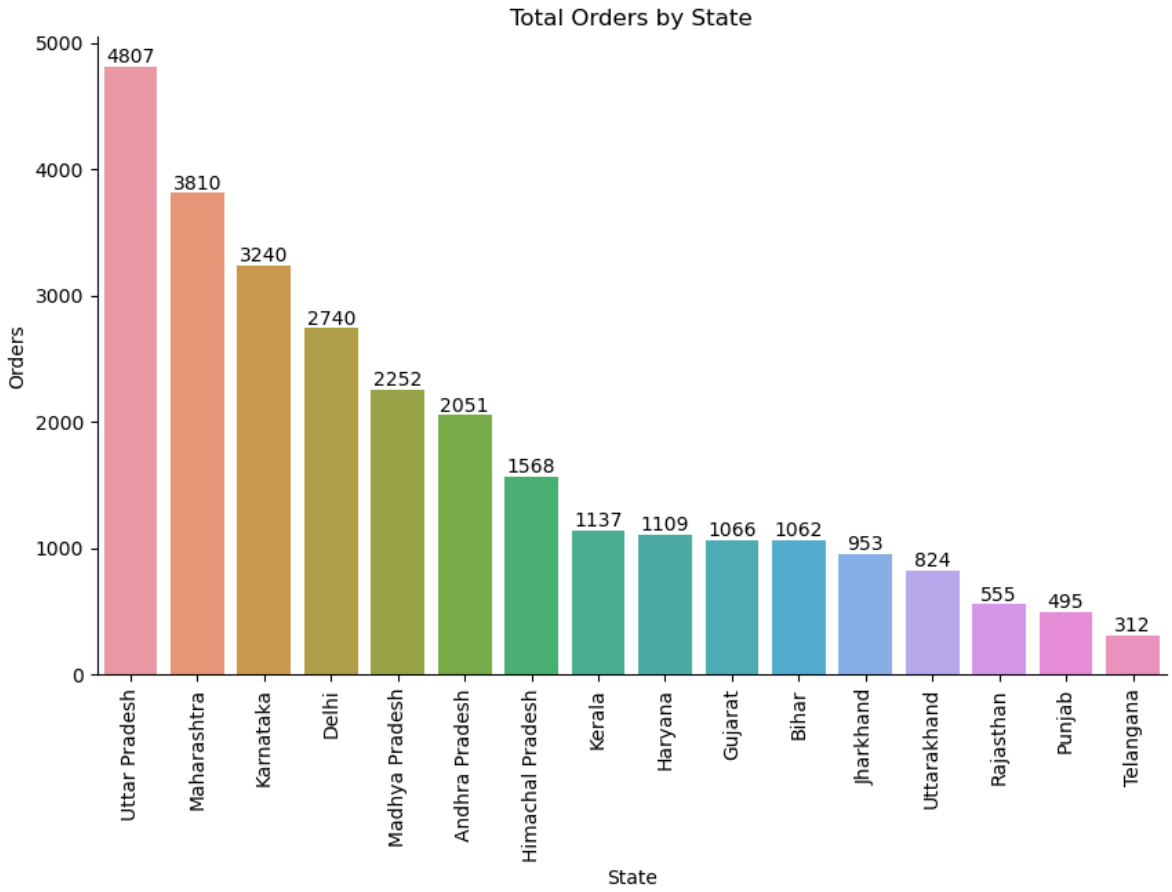
Out[19]:
```
Index(['User_ID', 'Cust_name', 'Product_ID', 'Gender', 'Age Group', 'Age',
       'Married', 'State', 'Zone', 'Occupation', 'Product_Category', 'Order
s',
       'Amount'],
      dtype='object')
```

In [27]:
```
plt.figure(figsize=(10, 6))

# Define the order of age groups in ascending order
age_group_order = sorted(data['Age Group'].unique())

sns.countplot(data=data, x='Age Group', hue='Gender', order=age_group_order)

# Add bar labels
ax = plt.gca()
for p in ax.patches:
    ax.annotate(f'{int(p.get_height())}', (p.get_x() + p.get_width() / 2, p.ge
                ha='center', va='bottom', fontsize=11)

plt.title('Distribution of Age Groups by Gender')
plt.xlabel('Age Group')
plt.ylabel('Count')
plt.legend(title='Gender', loc='upper right')

sns.set_palette('Accent')
plt.grid(True)
plt.show()
```



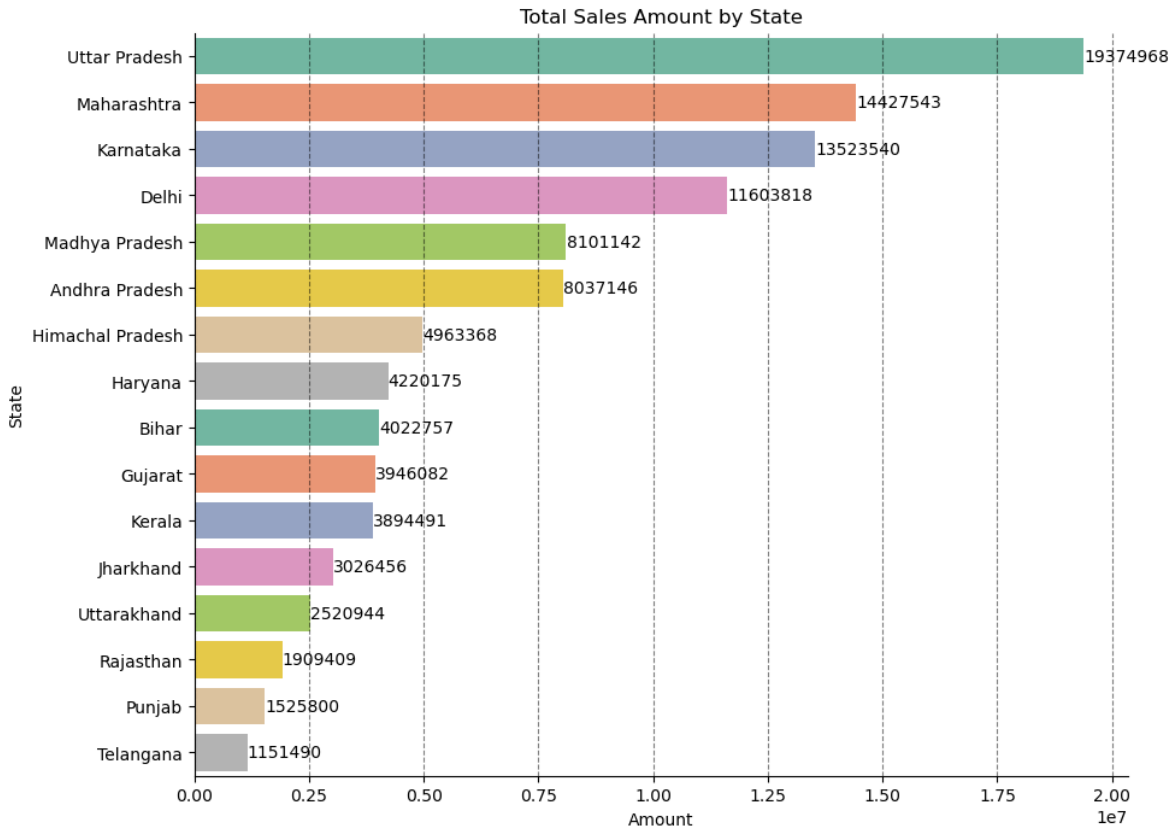
In [21]:

```
# Set the figure size
plt.figure(figsize=(10, 6))

# Group and sort the data
sales_state = data.groupby('State')['Orders'].sum().reset_index().sort_values(

# Plot the bar chart
sns.barplot(data=sales_state, x='State', y='Orders', order=sales_state['State'

# Add labels to the bars
ax = plt.gca()
for p in ax.patches:
    height = p.get_height()
    ax.annotate(f'{int(height)}', (p.get_x() + p.get_width() / 2, height),
                ha='center', va='bottom', fontsize=10)

# Set the x-axis label rotation
plt.xticks(rotation=90)

# Set the x-axis and y-axis labels
plt.xlabel('State')
plt.ylabel('Orders')

# Set the title
plt.title('Total Orders by State')

# Remove spines
ax.spines['right'].set_visible(False)
ax.spines['top'].set_visible(False)

# Display the plot
plt.show()
```


Total Orders by State
Orders
State
5000
4000
3000
2000
1000
0
4807
3810
3240
2740
2252
2051
1568
1137
1109
1066
1062
953
824
555
495
312
Uttar Pradesh
Maharashtra
Karnataka
Delhi
Madhya Pradesh
Andhra Pradesh
Himachal Pradesh
Kerala
Haryana
Gujarat
Bihar
Jharkhand
Uttarakhand
Rajasthan
Punjab
Telangana

In [22]:

```
# Increase the figure size
plt.figure(figsize=(10, 8))

# Group and sort the data
sales_state_amount = data.groupby('State')['Amount'].sum().reset_index().sort_

# Plot the bar chart
ax = sns.barplot(data=sales_state_amount, y='State', x='Amount', order=sales_s

# Set the y-axis label rotation
plt.yticks(rotation=0)

# Set the x-axis and y-axis labels
plt.xlabel('Amount')
plt.ylabel('State')

# Set the title
plt.title('Total Sales Amount by State')

# Remove spines
ax.spines['right'].set_visible(False)
ax.spines['top'].set_visible(False)

# Add bar labels
for p in ax.patches:
    width = p.get_width()
    label_x = p.get_x() + width + 5000
    label_y = p.get_y() + p.get_height() / 2
    value = int(width)
    ax.annotate(f'{value}', (label_x, label_y), ha='left', va='center')

# Add gridlines
plt.grid(axis='x', color='black', linestyle='--', alpha=0.5)

# Display the plot
plt.show()
```

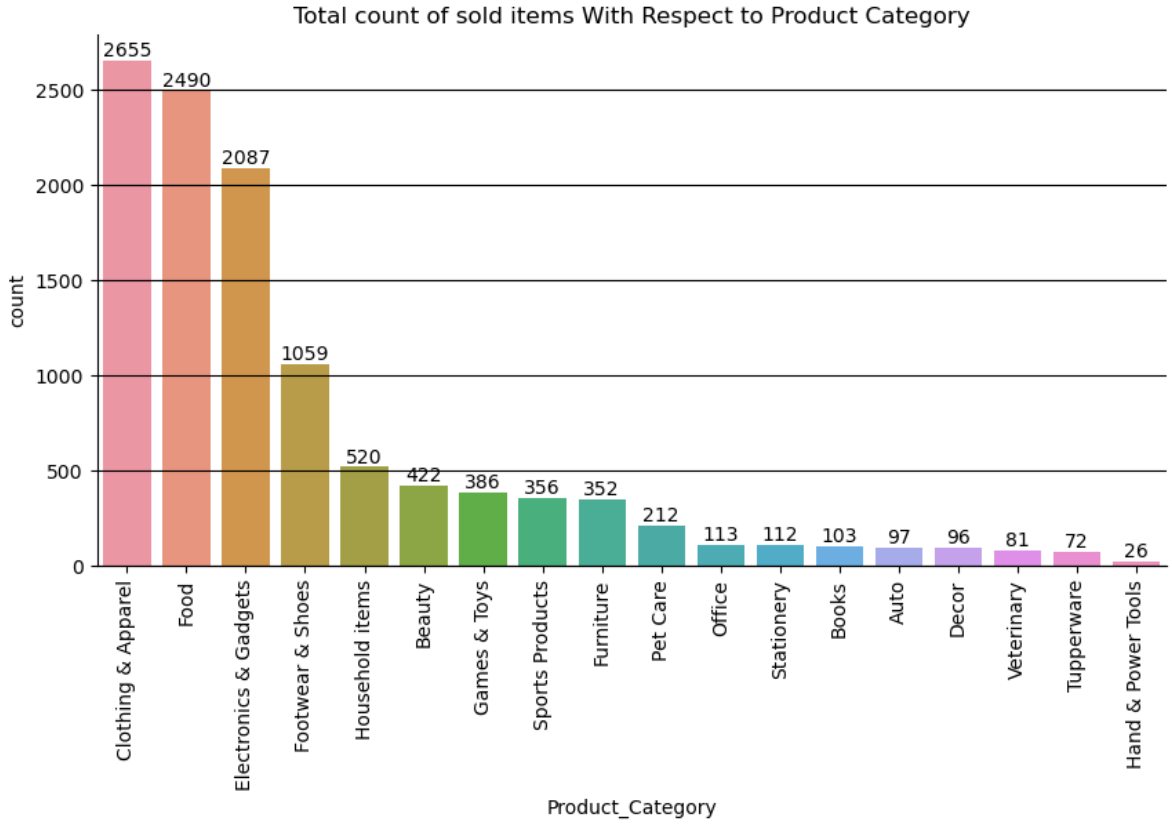


In [23]:
```
data.head()
```

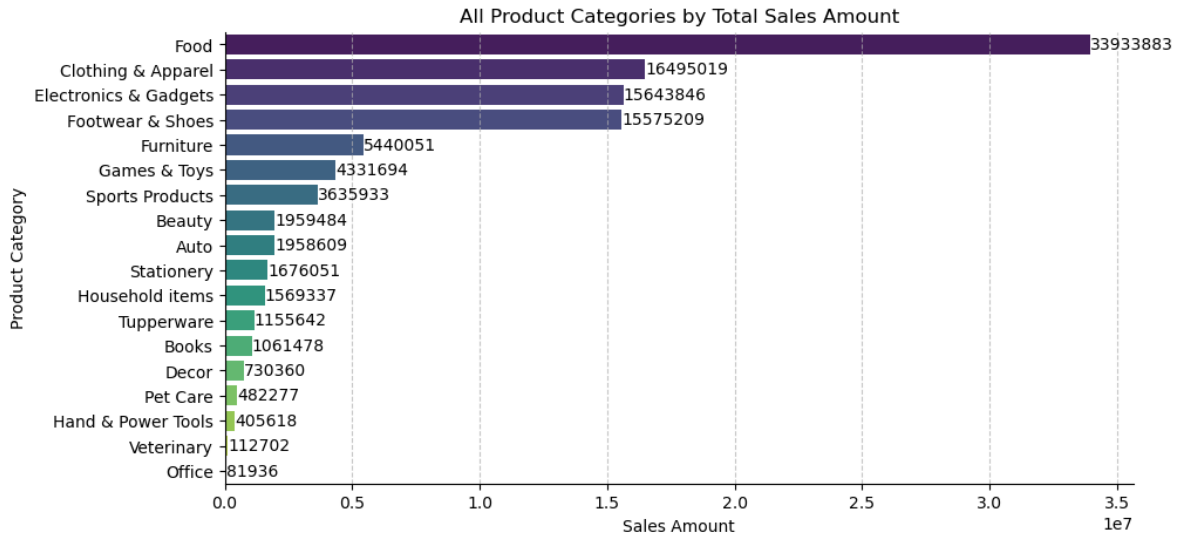
Out[23]:

	User_ID	Cust_name	Product_ID	Gender	Age Group	Age	Married	State	Zone	O
0	1002903	Sanskriti	P00125942	F	26-35	28	0	Maharashtra	Western	I
1	1000732	Kartik	P00110942	F	26-35	35	1	Andhra Pradesh	Southern	
2	1001990	Bindu	P00118542	F	26-35	35	1	Uttar Pradesh	Central	A
3	1001425	Sudevi	P00237842	M	0-17	16	0	Karnataka	Southern	Co
4	1000588	Joni	P00057942	M	26-35	28	1	Gujarat	Western	

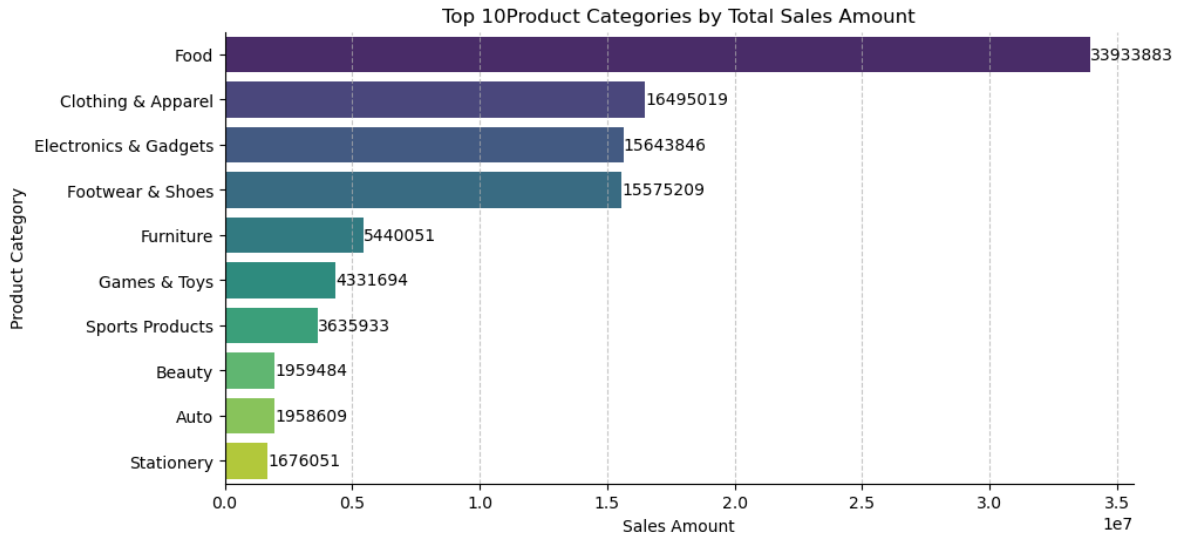
In [24]:
```
plt.figure(figsize=(10, 5))
ax = sns.countplot(data=data, x='Product_Category', order=data['Product_Catego
for bars in ax.containers:
    ax.bar_label(bars)

# Set the title
plt.title('Total count of sold items With Respect to Product Category')

# Remove spines
ax.spines['right'].set_visible(False)
ax.spines['top'].set_visible(False)


plt.grid(axis = 'y', color = 'Black')

plt.xticks(rotation=90)
plt.show()
```



In [25]:
```
sales_state = data.groupby('Product_Category')['Amount'].sum().reset_index().s

plt.figure(figsize=(10, 5))

# Plot the bar chart
sns.barplot(data=sales_state, x='Amount', y='Product_Category', palette='viric

# Add labels to the bars
ax = plt.gca()
for p in ax.patches:
    width = p.get_width()
    ax.annotate(f'{int(width)}', (width, p.get_y() + p.get_height() / 2),
                ha='left', va='center', fontsize=10)

# Set the title
plt.title('All Product Categories by Total Sales Amount')

# Set the x-axis label
plt.xlabel('Sales Amount')

# Set the y-axis label
plt.ylabel('Product Category')

# Remove spines
ax.spines['right'].set_visible(False)
ax.spines['top'].set_visible(False)

# Show gridlines
plt.grid(True, axis='x', linestyle='--', alpha=0.7)

plt.show()
```



In [26]:
```
sales_state = data.groupby('Product_Category')['Amount'].sum().nlargest(10).re

plt.figure(figsize=(10, 5))

# Plot the bar chart
sns.barplot(data=sales_state, x='Amount', y='Product_Category', palette='virid

# Add labels to the bars
ax = plt.gca()
for p in ax.patches:
    width = p.get_width()
    ax.annotate(f'{int(width)}', (width, p.get_y() + p.get_height() / 2),
                ha='left', va='center', fontsize=10)

# Set the title
plt.title('Top 10Product Categories by Total Sales Amount')

# Set the x-axis label
plt.xlabel('Sales Amount')

# Set the y-axis label
plt.ylabel('Product Category')

# Remove spines
ax.spines['right'].set_visible(False)
ax.spines['top'].set_visible(False)

# Show gridlines
plt.grid(True, axis='x', linestyle='--', alpha=0.7)

plt.show()
```



Thanks for your time