# DIWALI SALES ANALYSIS USING PYTHON

## DATA IMPORTING

In [2]:
```
# In this step we are importing Python Libararies necessary for our Data Analysis

import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
%matplotlib inline
import seaborn as sns
```

In [3]:
```
# We are importing the csv file as a data frame

df = pd.read_csv('D:\Power BI Projects\Diwali sales analysis\Diwali Sales Data.csv', encoding= 'unicode_escape')
```

In [4]:
```
# Now We will check wheather the data is imported or not

df.shape

# This will show the number of rows and columns in the data set we imported
# and also if the dataset is not imported it will throw an error
```

Out[4]: (11251, 15)

## DATA CLEANING

In [5]:
```
# Now we need to Preview how the data Looks in the dataset

df.head()

# This will show us the first 5 rows of the dataset as a preview
```

Out[5]:

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

In [6]:
```
# Now we need to identify the number of rows and columns, their data types, values its containing and
# Values that must be cleaned.

df.info()

# This will help us identify all the basic details required for cleaning of the data set
```

```
<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]:
```
# From above info we can see that column 13 and 14 have all null values
# and these will not be useful for us, so we will remove these columns from dataset

df.drop(['Status','unnamed1'], axis = 1, inplace = True)

# This will delete the specified column, axis paramter specifies drop all the rows for entire column and
# Inplace is used to make the changes saved in the main data set
```

In [8]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 11251 entries, 0 to 11250
Data columns (total 13 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
dtypes: float64(1), int64(4), object(8)
memory usage: 1.1+ MB
```

In [9]:
```
# Now in the remaining datas we must identify is there any null values is there

pd.isnull(df)

# Now this will check the entire dataset for null values and shows true or false of whole table,
# shows True for null values and False for non null
```

Out[9]:

| | User_ID | Cust_name | Product_ID | Gender | Age Group | Age | Marital_Status | State | Zone | Occupation | Product_Category | Orders | Amount |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|

| | User_ID | Cust_name | Product_ID | Gender | Age Group | Age | Marital_Status | State | Zone | Occupation | Product_Category | Orders | Amount |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **0** | False | False | False | False | False | False | False | False | False | False | False | False | False |
| **1** | False | False | False | False | False | False | False | False | False | False | False | False | False |
| **2** | False | False | False | False | False | False | False | False | False | False | False | False | False |
| **3** | False | False | False | False | False | False | False | False | False | False | False | False | False |
| **4** | False | False | False | False | False | False | False | False | False | False | False | False | False |
| **...** | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... |
| **11246** | False | False | False | False | False | False | False | False | False | False | False | False | False |
| **11247** | False | False | False | False | False | False | False | False | False | False | False | False | False |
| **11248** | False | False | False | False | False | False | False | False | False | False | False | False | False |
| **11249** | False | False | False | False | False | False | False | False | False | False | False | False | False |
| **11250** | False | False | False | False | False | False | False | False | False | False | False | False | False |

11251 rows × 13 columns

In [10]:
```
# From the above table we can't conclude the number of null values, so we need to find other way

pd.isnull(df).sum()

# This will sum up all the null values and give it as a number column wise
```

Out[10]:
```
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 [11]:
```
# We can clearly see there are 12 null values in the amount column, Now
# we will delete the null value rows

df.dropna(inplace=True)

# This will delete all null value rows and save the changes
```

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

Out[12]:
```
(11239, 13)
```

## DATA ANALYSIS

In [13]:
```
# Now We have Cleaned the data and we have to Analyse the necesary data based on client Requirements
# Firstly we will get a brief analysis of the Datas

df.describe()
```

Out[13]:

| | 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 | 9453.610858 |
| **std** | 1.716039e+03 | 12.753866 | 0.493589 | 1.114967 | 5222.355869 |
| **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 |

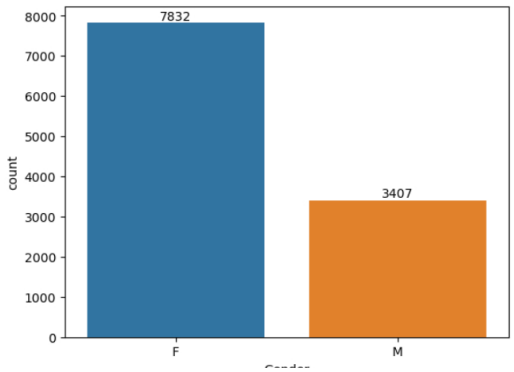
In [15]:
```
# We don't want these basic mathematical operations on userid and martial status

df[['Age','Orders','Amount']].describe()
```

Out[15]:

| | Age | Orders | Amount |
|---|---|---|---|
| **count** | 11239.000000 | 11239.000000 | 11239.000000 |
| **mean** | 35.410357 | 2.489634 | 9453.610858 |
| **std** | 12.753866 | 1.114967 | 5222.355869 |
| **min** | 12.000000 | 1.000000 | 188.000000 |
| **25%** | 27.000000 | 2.000000 | 5443.000000 |
| **50%** | 33.000000 | 2.000000 | 8109.000000 |
| **75%** | 43.000000 | 3.000000 | 12675.000000 |
| **max** | 92.000000 | 4.000000 | 23952.000000 |

### 1) Gender Analysis

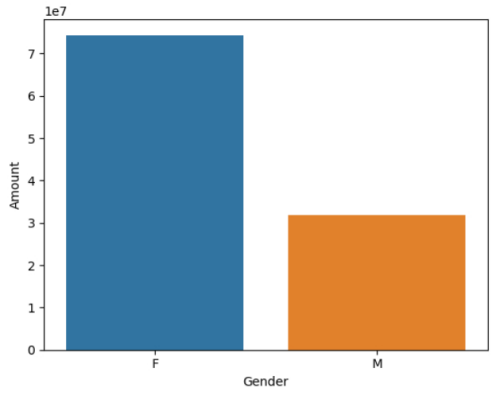
In [17]:
```
# We will plot a bar bar chart for Gender Analysis

ax = sns.countplot(x = 'Gender',data = df)
# seaborn used to plot barchart('countplot'), where x axis has Gender and chart based on dataframe df

for bars in ax.containers: # we are marking the values on the bars by this line
    ax.bar_label(bars)
```



Gender

In [18]:
```
# plotting a bar chart for gender vs total amount

sales_gen = df.groupby(['Gender'], as_index=False)['Amount'].sum().sort_values(by='Amount', ascending=False)
# we are groping Gender and amount and plotting bar graph

sns.barplot(x = 'Gender',y= 'Amount' ,data = sales_gen)
```

Out[18]: <Axes: xlabel='Gender', ylabel='Amount'>

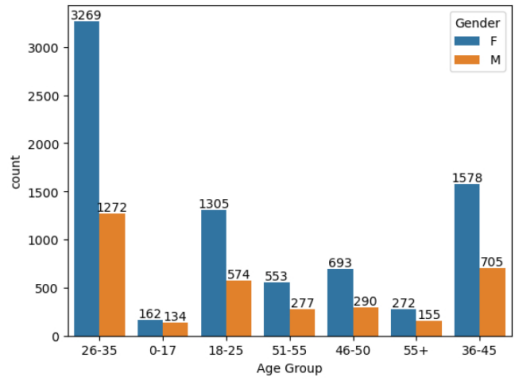



**We can see that In both cases(i.e) Total number of orders are made by Women and Total amount that Women Bought our product**

**is also High**

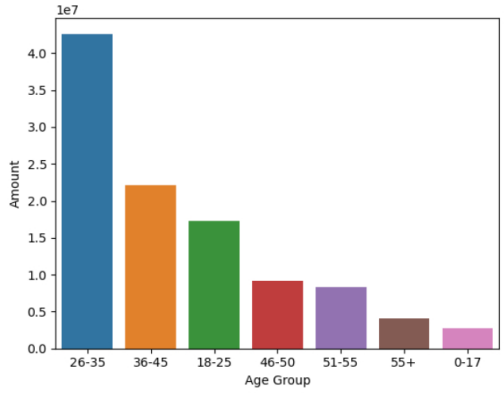
## 2) Age Analysis

In [19]:
```
ax = sns.countplot(data = df, x = 'Age Group', hue = 'Gender')

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




In [20]:
```
# Total Amount vs Age Group
sales_age = df.groupby(['Age Group'], as_index=False)['Amount'].sum().sort_values(by='Amount', ascending=False)

sns.barplot(x = 'Age Group',y= 'Amount' ,data = sales_age)
```

Out[20]: <Axes: xlabel='Age Group', ylabel='Amount'>

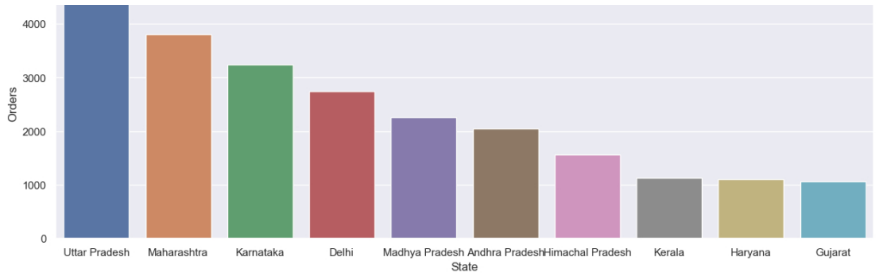



**From above we can see that Most of the buyers are from age group 26-35 and among them Women at these age are major buyers**

## 3) State Analysis

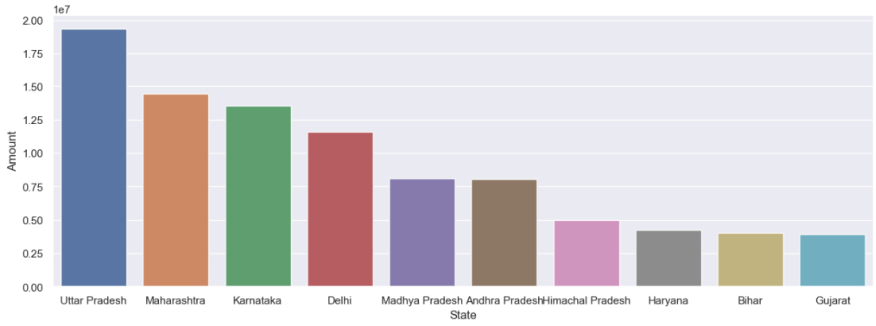
In [21]:
```
# total number of orders from top 10 states

sales_state = df.groupby(['State'], as_index=False)['Orders'].sum().sort_values(by='Orders', ascending=False).head(10)

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

Out[21]: <Axes: xlabel='State', ylabel='Orders'>




In [22]:
```
# total amount/sales from top 10 states

sales_state = df.groupby(['State'], as_index=False)['Amount'].sum().sort_values(by='Amount', ascending=False).head(10)

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

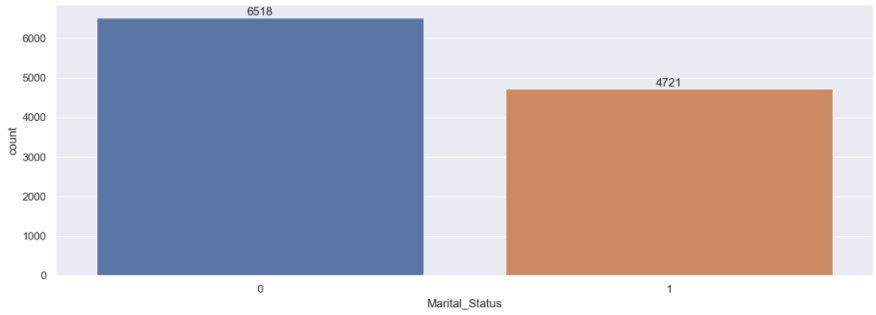
Out[22]: <Axes: xlabel='State', ylabel='Amount'>




**We can see that most of the orders and total amount gained are from UP, Maharashtra and Karnataka**

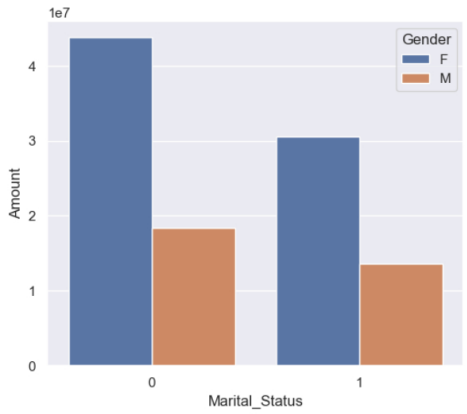
## 4) Marital Status

In [23]:
```
ax = sns.countplot(data = df, x = 'Marital_Status')

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




In [24]:
```
sales_state = df.groupby(['Marital_Status', 'Gender'], as_index=False)['Amount'].sum().sort_values(by='Amount', ascending=False)

sns.set(rc={'figure.figsize':(6,5)})
sns.barplot(data = sales_state, x = 'Marital_Status',y= 'Amount', hue='Gender')
```

Out[24]: <Axes: xlabel='Marital_Status', ylabel='Amount'>

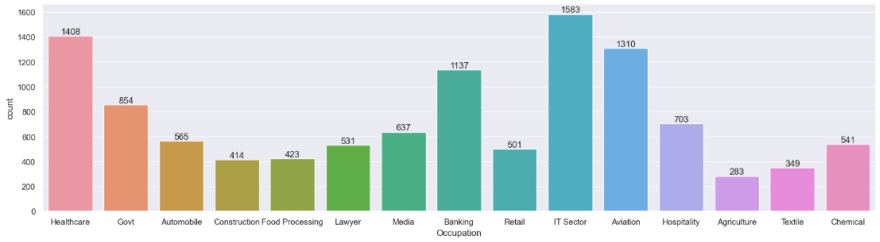



**We can see that most purchase power is from married woman**

## 5) Occupation Analysis

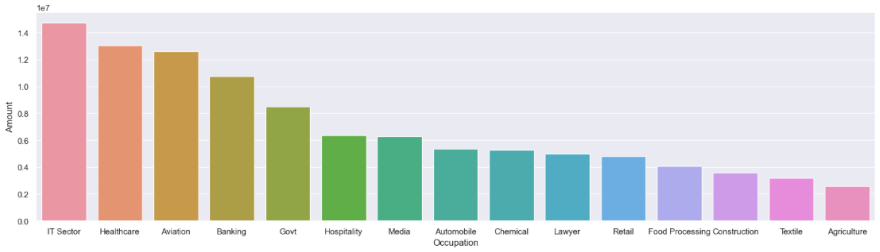
In [25]:
```
sns.set(rc={'figure.figsize':(20,5)})
ax = sns.countplot(data = df, x = 'Occupation')

for bars in ax.containers:
```

```
    ax.bar_label(bars)
```




In [26]:
```
sales_state = df.groupby(['Occupation'], as_index=False)['Amount'].sum().sort_values(by='Amount', ascending=False)

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

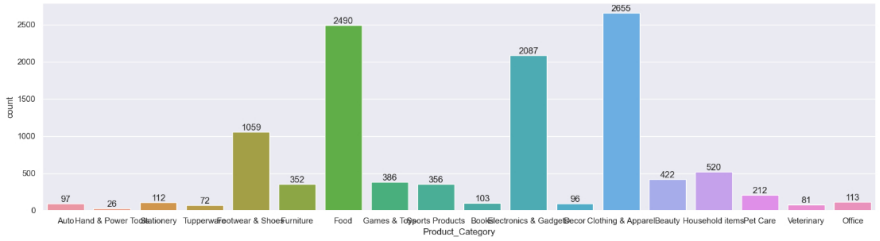
Out[26]: <Axes: xlabel='Occupation', ylabel='Amount'>




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

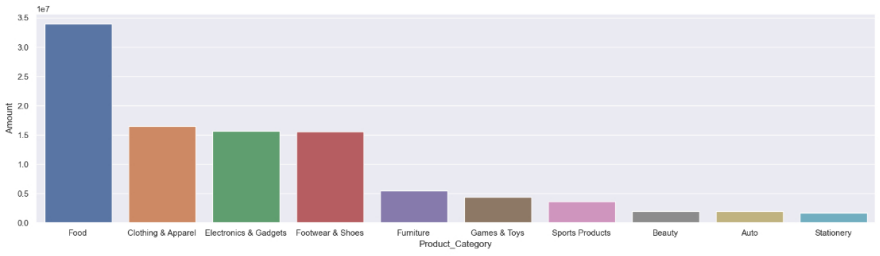
## 6) Product Category Analysis

In [27]:
```
sns.set(rc={'figure.figsize':(20,5)})
ax = sns.countplot(data = df, x = 'Product_Category')

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




In [28]:
```
sales_state = df.groupby(['Product_Category'], as_index=False)['Amount'].sum().sort_values(by='Amount', ascending=False).head(10)

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

Out[28]: <Axes: xlabel='Product_Category', ylabel='Amount'>

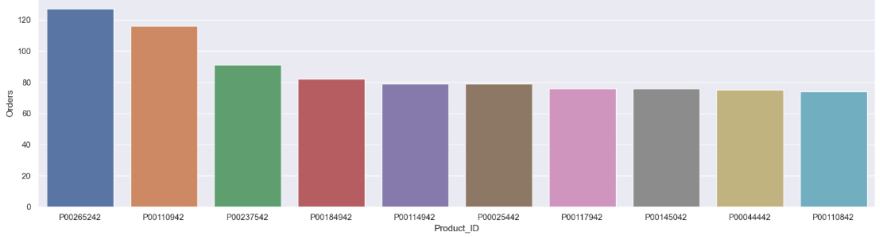



**we can see that most of the sold products are from Food, Clothing and Electronics category**

## 7) Individual Product Analysis

In [29]:
```
sales_state = df.groupby(['Product_ID'], as_index=False)['Orders'].sum().sort_values(by='Orders', ascending=False).head(10)

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

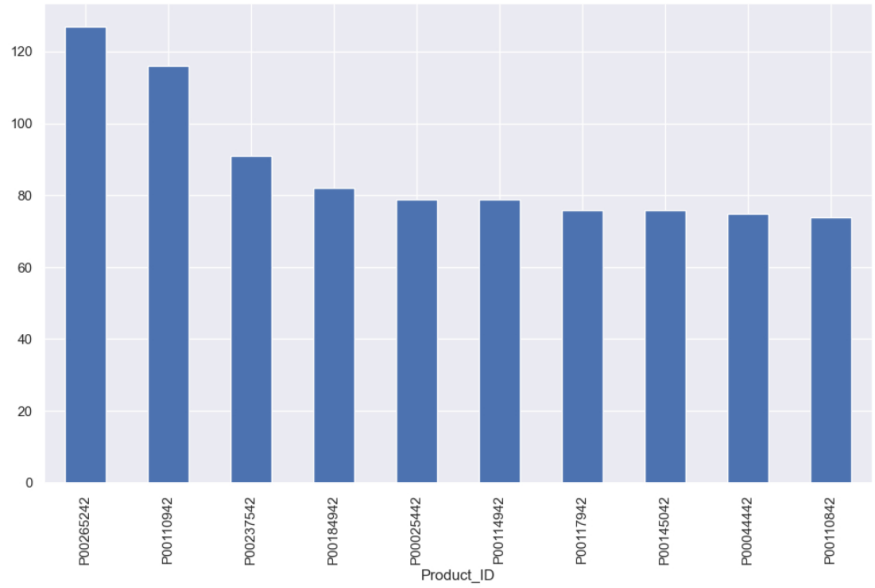
Out[29]: <Axes: xlabel='Product_ID', ylabel='Orders'>




In [30]:
```
# top 10 most sold products (same thing as above)
```

```
fig1, ax1 = plt.subplots(figsize=(12,7))
df.groupby('Product_ID')['Orders'].sum().nlargest(10).sort_values(ascending=False).plot(kind='bar')
```

Out[30]: <Axes: xlabel='Product_ID'>




**CONCLUSION :**

**Married women age group 26-35 yrs from UP, Maharastra and Karnataka working in IT, Healthcare and Aviation are more likely to buy products from Food, Clothing and Electronics category**