Exploratory Data Analysis using Python (Pandas library majorly)

```
import pandas as pd
```

Importing dataset which is in csv format

```
df= pd.read_csv('Diwali Sales Data.csv',encoding='unicode_escape')
df.head()
```

	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

Data frame structure:

```
df.shape
```

```
(11251, 15)
```

Check completeness of columns:

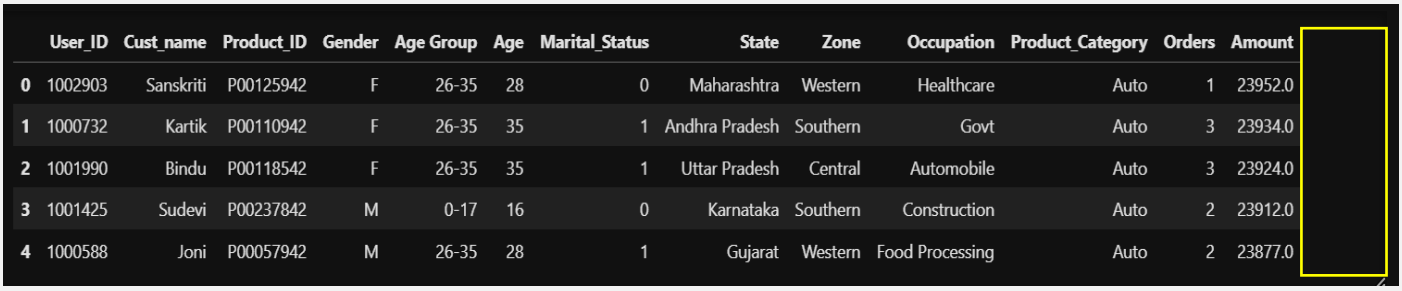
```
df.info()
```

```
<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
```

Data Transformation

```
#delete empty columns
```

```
df.drop(["Status","unnamed1"],axis=1,inplace=True)
df.head()
```



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

Let's look for NA Values:

```
# search for NA values
```

```
df.isnull().sum()
```

```
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
```

Amount is an important column, as there are 12 null entries only. Dropping the rows is feasible

```
#drop NA values because only 12
```

```
df.dropna(inplace=True)
```

Confirm the deletion of NA values:

```
df.isnull().sum()
```

```
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
```

Rows reduced from 11251 to 11239

```
#compare previous shape and this
df.shape
```

```
(11239, 13)
```

Standardization of data types:

```
df.dtypes
```

```
User_ID              int64
Cust_name           object
Product_ID          object
Gender              object
Age Group           object
Age                  int64
Marital_Status       int64
State               object
Zone                object
Occupation          object
Product_Category    object
Orders               int64
Amount             float64
dtype: object
```

```
#change amount data type to int
df['Amount']=df['Amount'].astype('int')
```

```
#check changes
df['Amount'].dtype
```

```
dtype('int32')
```

Exploring numeric columns

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

	Age	Orders	Amount
count	11239.000000	11239.000000	11239.000000
mean	35.410357	2.489634	9453.610553
std	12.753866	1.114967	5222.355168
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

Column Names:

```
df.columns
```

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

Data Exploration

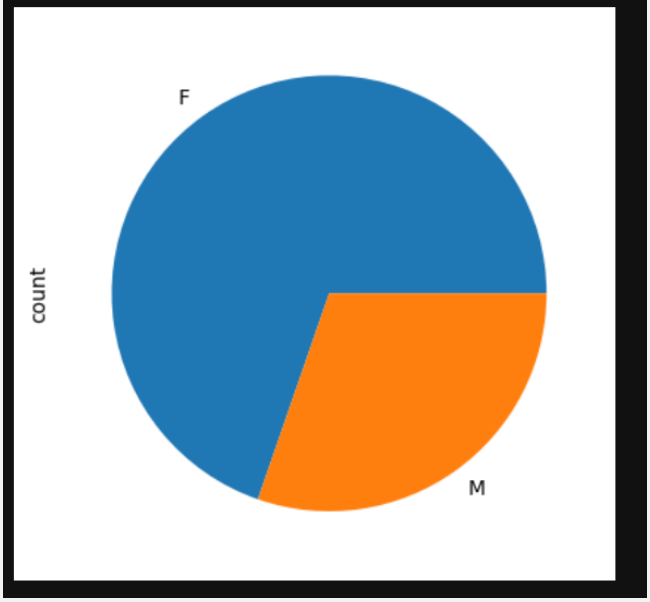
Gender analysis

```
df['Gender'].value_counts()
```

```
Gender
F    7832
M    3407
Name: count, dtype: int64
```

```
df['Gender'].value_counts().plot(kind='pie')
```

```
<Axes: ylabel='count'>
```



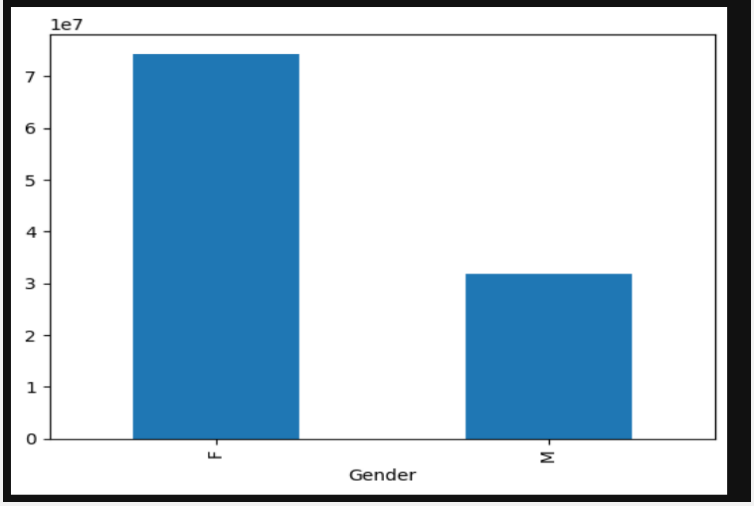
Female consumers are more in number and also contribute more in sales amount

```
df.groupby("Gender")["Amount"].sum()
```

```
Gender
F    74335853
M    31913276
Name: Amount, dtype: int32
```

```
df.groupby("Gender")["Amount"].sum().plot(kind='bar')
```

```
<Axes: xlabel='Gender'>
```



Age- Group analysis

```
df["Age Group"].value_counts().sort_index()
```

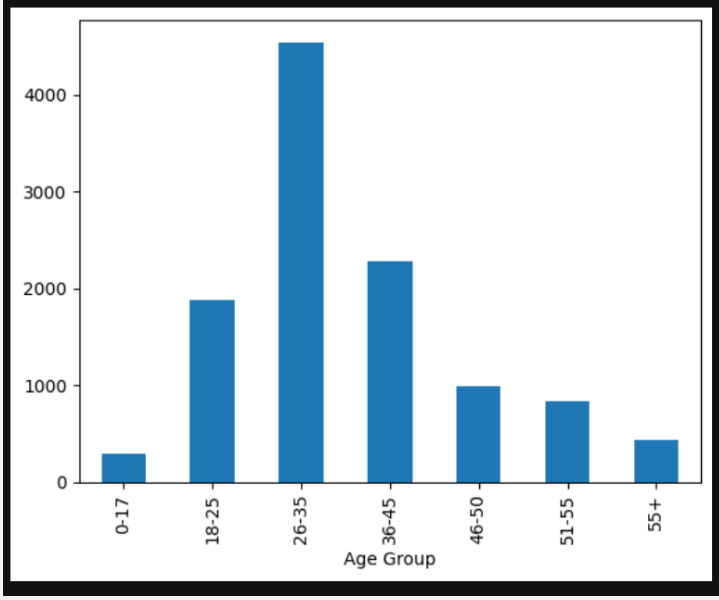
```
Age Group
0-17      296
18-25    1879
26-35    4541
36-45    2283
46-50     983
51-55     830
55+       427
Name: count, dtype: int64
```

```
df.groupby("Age Group")["Amount"].sum()
```

```
Age Group
0-17      2699653
18-25    17240732
26-35    42613442
36-45    22144994
46-50     9207844
51-55     8261477
55+       4080987
Name: Amount, dtype: int32
```

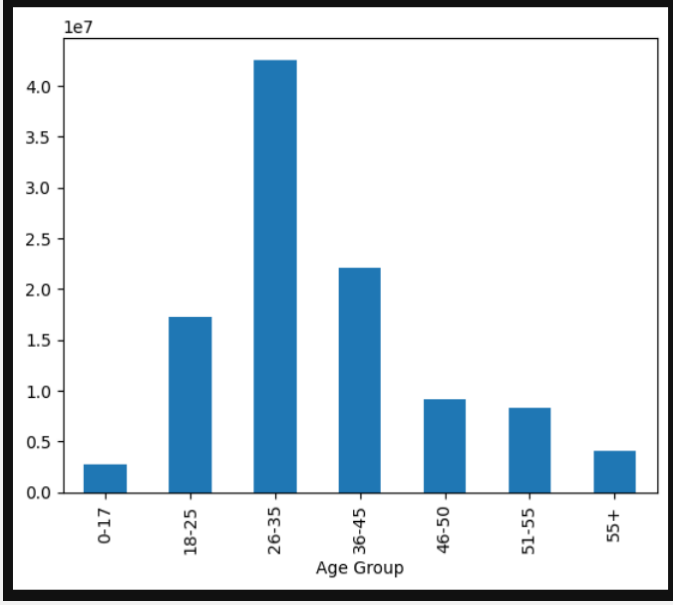
```
df["Age Group"].value_counts().sort_index().plot(kind='bar')
```

```
<Axes: xlabel='Age Group'>
```




```
df.groupby("Age Group")["Amount"].sum().plot(kind='bar')
```

```
<Axes: xlabel='Age Group'>
```



Age group 26-35 are spending more

Teenagers and Senior Adults spend less overall.

Marital Status wise analysis

```
df["Marital_Status"].value_counts()
```

```
Marital_Status
0    6518
1    4721
Name: count, dtype: int64
```

```
df.groupby(["Marital_Status","Gender"])[["Orders","Amount"]].sum()
```

		Orders	Amount
Marital_Status	**Gender**		
0	**F**	11393	43786646
	M	4856	18338738
1	**F**	8155	30549207
	M	3577	13574538

Single Women are spending the most while married men spend the least

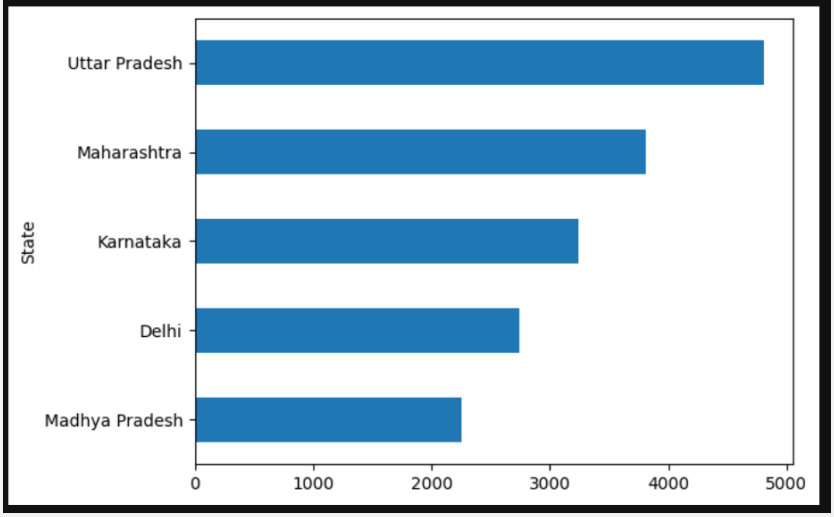
Top Ordered States

```
top_ordered_state=df.groupby("State")["Orders"].sum().nlargest(5)
top_ordered_state
```

```
State
Uttar Pradesh     4807
Maharashtra       3810
Karnataka         3240
Delhi             2740
Madhya Pradesh    2252
Name: Orders, dtype: int64
```

```
top_ordered_state.sort_values().plot(kind='barh')
```

```
<Axes: ylabel='State'>
```



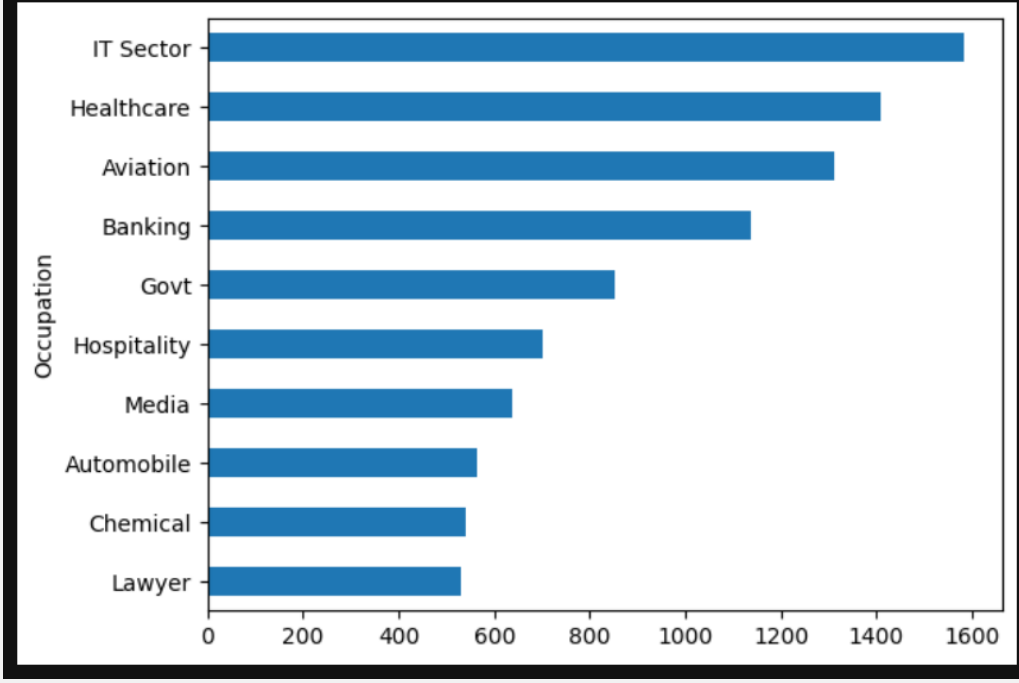
Orders by Occupation

```
Orders_by_occupation=df["Occupation"].value_counts().sort_values(ascending=False).head(10)
Orders_by_occupation
```

```
Occupation
IT Sector      1583
Healthcare     1408
Aviation       1310
Banking        1137
Govt            854
Hospitality     703
Media           637
Automobile      565
Chemical        541
Lawyer          531
Name: count, dtype: int64
```

```
Orders_by_occupation.sort_values().plot(kind='barh')
```

```
<Axes: ylabel='Occupation'>
```



Female in IT sector spend the most

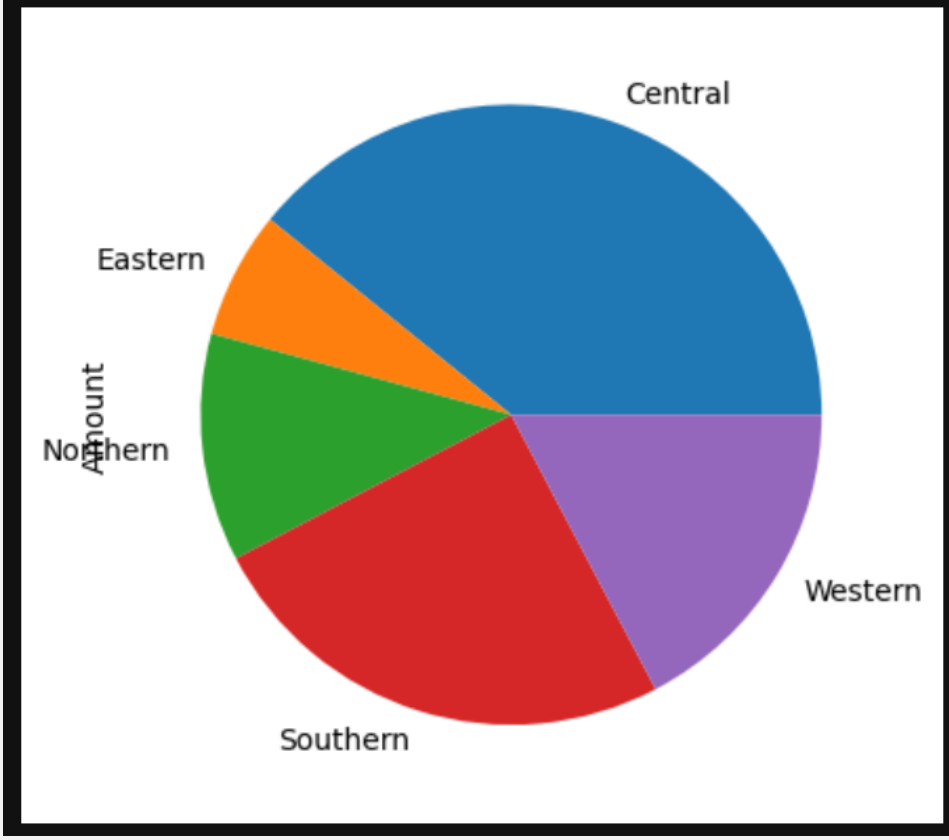
```
df[df["Occupation"]=="IT Sector"].groupby("Gender")["Amount"].sum()
#IT sector male spend only half of total
```

```
Gender
F    10184835
M     4570244
Name: Amount, dtype: int32
```

Sales division by Zone

```
df.groupby('Zone')["Amount"].sum().plot(kind='pie',)
```

```
<Axes: ylabel='Amount'>
```



Central Zone performs the best.

Let's look at the States in Central Zone

```
#which states are in central region
df[df["Zone"]=="Central"]["State"].unique()
```

```
array(['Uttar Pradesh', 'Delhi', 'Madhya Pradesh', 'Uttarakhand'],
      dtype=object)
```

Product Category

```
df["Product_Category"].unique()
```

```
array(['Auto', 'Hand & Power Tools', 'Stationery', 'Tupperware',
       'Footwear & Shoes', 'Furniture', 'Food', 'Games & Toys',
       'Sports Products', 'Books', 'Electronics & Gadgets', 'Decor',
       'Clothing & Apparel', 'Beauty', 'Household items', 'Pet Care',
       'Veterinary', 'Office'], dtype=object)
```

```
df.groupby('Product_Category')[["Orders","Amount"]].sum().sort_values(by="Orders",ascending=False)
```

Product_Category	Orders	Amount
Clothing & Apparel	6634	16495019
Food	6110	33933883
Electronics & Gadgets	5226	15643846
Footwear & Shoes	2646	15575209
Household items	1331	1569337
Beauty	1086	1959484

Clothing though has highest orders, revenue from food is twice the revenue from clothing

Top 10 Products

```
#top_10_selling products
top_10=df.groupby('Product_ID')['Orders'].sum().nlargest(10)
top_10
```

```
Product_ID
P00265242    127
P00110942    116
P00237542     91
P00184942     82
P00025442     79
P00114942     79
P00117942     76
P00145042     76
P00044442     75
P00110842     74
Name: Orders, dtype: int64
```