```
import pandas as pd
pd.options.display.float_format = '{:,.2f}'.format
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from IPython.display import Image
from sklearn.tree import DecisionTreeClassifier
```

```
colors = ["#89CFF0", "#FF69B4", "#FFD700", "#7B68EE", "#FF4500",
          "#9370DB", "#32CD32", "#8A2BE2", "#FF6347", "#20B2AA",
          "#FF69B4", "#00CED1", "#FF7F50", "#7FFF00", "#DA70D6"]
```

+ Code + Text

```
df = pd.read_csv("/content/shopping_trends_updated.csv")
df.head(10)
```

	Customer ID	Age	Gender	Item Purchased	Category	Purchase Amount (USD)	Location	Size	Color	Season	Review Rating	Subscription Status	Shipping Type	Dis Ap
0	1	55	Male	Blouse	Clothing	53	Kentucky	L	Gray	Winter	3.10	Yes	Express	
1	2	19	Male	Sweater	Clothing	64	Maine	L	Maroon	Winter	3.10	Yes	Express	
2	3	50	Male	Jeans	Clothing	73	Massachusetts	S	Maroon	Spring	3.10	Yes	Free Shipping	
3	4	21	Male	Sandals	Footwear	90	Rhode Island	M	Maroon	Spring	3.50	Yes	Next Day Air	
4	5	45	Male	Blouse	Clothing	49	Oregon	M	Turquoise	Spring	2.70	Yes	Free Shipping	
5	6	46	Male	Sneakers	Footwear	20	Wyoming	M	White	Summer	2.90	Yes	Standard	
6	7	63	Male	Shirt	Clothing	85	Montana	M	Gray	Fall	3.20	Yes	Free Shipping	
7	8	27	Male	Shorts	Clothing	34	Louisiana	L	Charcoal	Winter	3.20	Yes	Free Shipping	
8	9	26	Male	Coat	Outerwear	97	West Virginia	L	Silver	Summer	2.60	Yes	Express	
9	10	57	Male	Handbag	Accessories	31	Missouri	M	Pink	Spring	4.80	Yes	2-Day Shipping	

```
df.shape
```

```
(3900, 18)
```

```
df.columns
```

```
Index(['Customer ID', 'Age', 'Gender', 'Item Purchased', 'Category',
       'Purchase Amount (USD)', 'Location', 'Size', 'Color', 'Season',
       'Review Rating', 'Subscription Status', 'Shipping Type',
       'Discount Applied', 'Promo Code Used', 'Previous Purchases',
       'Payment Method', 'Frequency of Purchases'],
      dtype='object')
```

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 3900 entries, 0 to 3899
Data columns (total 18 columns):
 #   Column                  Non-Null Count  Dtype  
---  ------                  --------------  -----  
 0   Customer ID             3900 non-null   int64  
 1   Age                     3900 non-null   int64  
 2   Gender                  3900 non-null   object 
 3   Item Purchased          3900 non-null   object 
 4   Category                3900 non-null   object 
 5   Purchase Amount (USD)   3900 non-null   int64  
 6   Location                3900 non-null   object 
 7   Size                    3900 non-null   object 
 8   Color                   3900 non-null   object 
 9   Season                  3900 non-null   object 
 10  Review Rating           3900 non-null   float64
 11  Subscription Status     3900 non-null   object 
 12  Shipping Type           3900 non-null   object 
 13  Discount Applied        3900 non-null   object 
 14  Promo Code Used         3900 non-null   object 
 15  Previous Purchases      3900 non-null   int64  
 16  Payment Method          3900 non-null   object 
 17  Frequency of Purchases  3900 non-null   object
dtypes: float64(1), int64(4), object(13)
memory usage: 548.6+ KB
```

```
df.describe()
```

	Customer ID	Age	Purchase Amount (USD)	Review Rating	Previous Purchases
count	3,900.00	3,900.00	3,900.00	3,900.00	3,900.00
mean	1,950.50	44.07	59.76	3.75	25.35
std	1,125.98	15.21	23.69	0.72	14.45
min	1.00	18.00	20.00	2.50	1.00
25%	975.75	31.00	39.00	3.10	13.00
50%	1,950.50	44.00	60.00	3.70	25.00
75%	2,925.25	57.00	81.00	4.40	38.00
max	3,900.00	70.00	100.00	5.00	50.00

```
df.describe()
```

	Customer ID	Age	Purchase Amount (USD)	Review Rating	Previous Purchases
count	3,900.00	3,900.00	3,900.00	3,900.00	3,900.00
mean	1,950.50	44.07	59.76	3.75	25.35
std	1,125.98	15.21	23.69	0.72	14.45
min	1.00	18.00	20.00	2.50	1.00
25%	975.75	31.00	39.00	3.10	13.00
50%	1,950.50	44.00	60.00	3.70	25.00
75%	2,925.25	57.00	81.00	4.40	38.00
max	3,900.00	70.00	100.00	5.00	50.00

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

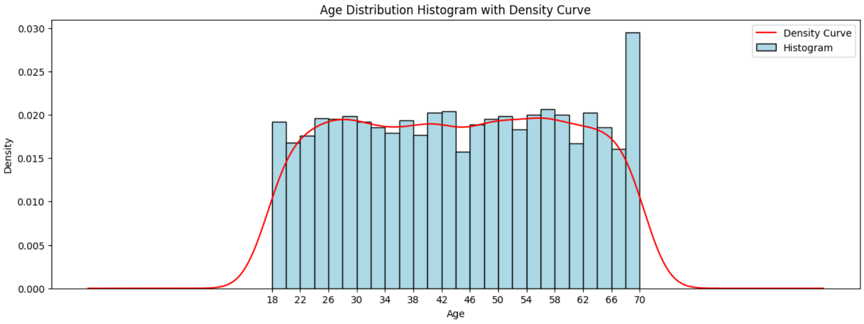
```
Customer ID               0
Age                       0
Gender                    0
Item Purchased            0
Category                  0
Purchase Amount (USD)     0
Location                  0
Size                      0
Color                     0
Season                    0
Review Rating             0
Subscription Status       0
Shipping Type             0
Discount Applied          0
Promo Code Used           0
Previous Purchases        0
Payment Method            0
Frequency of Purchases    0
dtype: int64
```

```
df.duplicated().sum()
```

```
0
```

Visualization

```
fig, ax = plt.subplots(figsize = (15, 5))

ax.hist(df['Age'], bins = 26, edgecolor = 'black', color = 'lightblue', density = True)
df['Age'].plot(kind = 'kde', color = 'red', ax = ax)

ax.set_xlabel('Age')
ax.set_ylabel('Density')
ax.set_title('Age Distribution Histogram with Density Curve')
ax.legend(['Density Curve', 'Histogram'])
step = 4
plt.xticks(range(int(df['Age'].min()), int(df['Age'].max()) + 1, step))

plt.show()
```

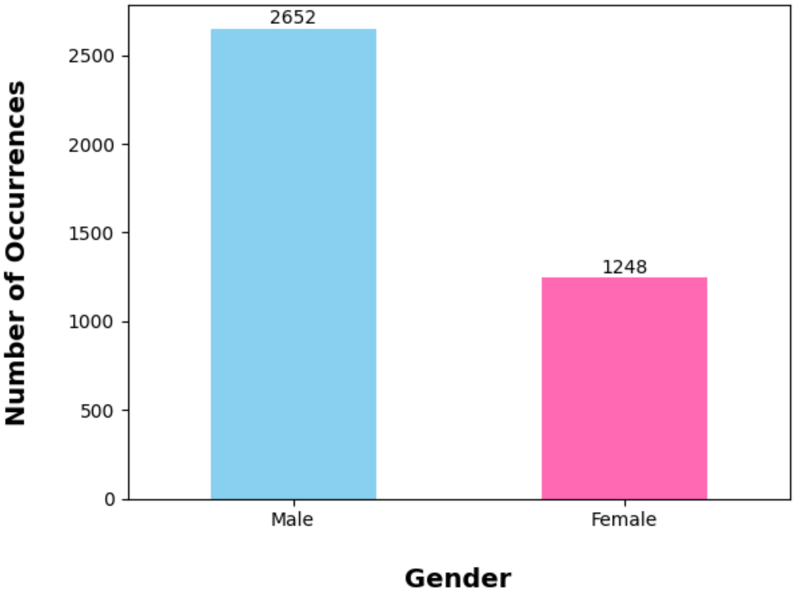


1.GENDER

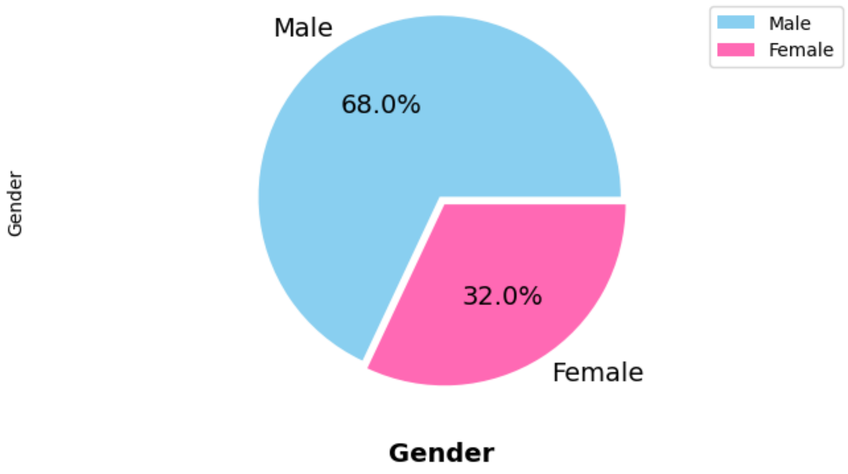
```
df["Gender"].value_counts()
```

```
    Male      2652
    Female    1248
    Name: Gender, dtype: int64
```

```
ax = df["Gender"].value_counts().plot(kind = 'bar', color = colors, rot=0)
ax.set_xticklabels(('Male', 'Female'))

for p in ax.patches:
    ax.annotate(int(p.get_height()), (p.get_x() + 0.25, p.get_height() + 1), ha = 'center', va = 'bottom', color = 'black')
plt.xlabel('Gender', weight = "bold", fontsize = 14, labelpad = 20)
plt.ylabel('Number of Occurrences', weight = "bold", fontsize = 14, labelpad = 20);
```



```
plt.figure(figsize = (8, 4))
counts = df["Gender"].value_counts()
explode = (0, 0.05)

counts.plot(kind = 'pie', fontsize = 14, colors = colors, explode = explode, autopct = '%1.1f%%')
plt.xlabel('Gender', weight = "bold", fontsize = 14, labelpad = 20)
plt.axis('equal')
plt.legend(labels = counts.index, loc = "best")
plt.show()
```

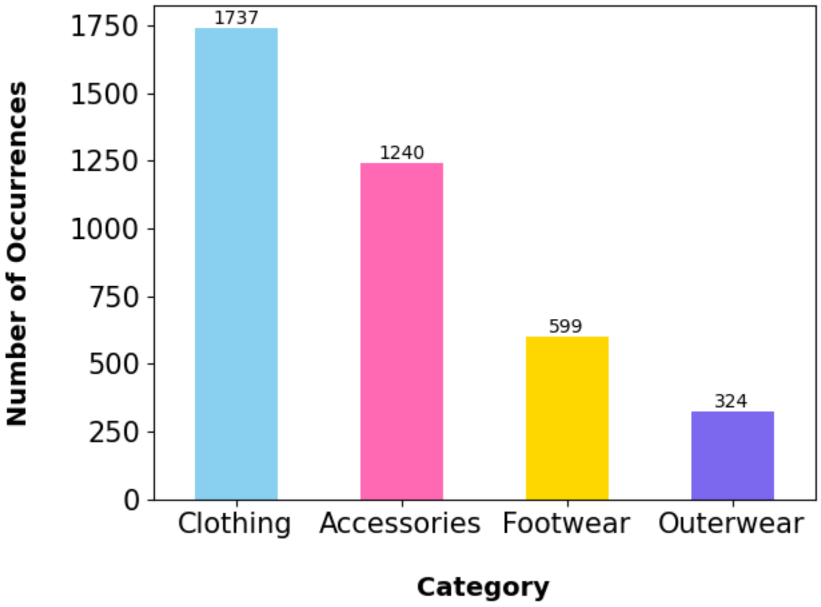


2. Category

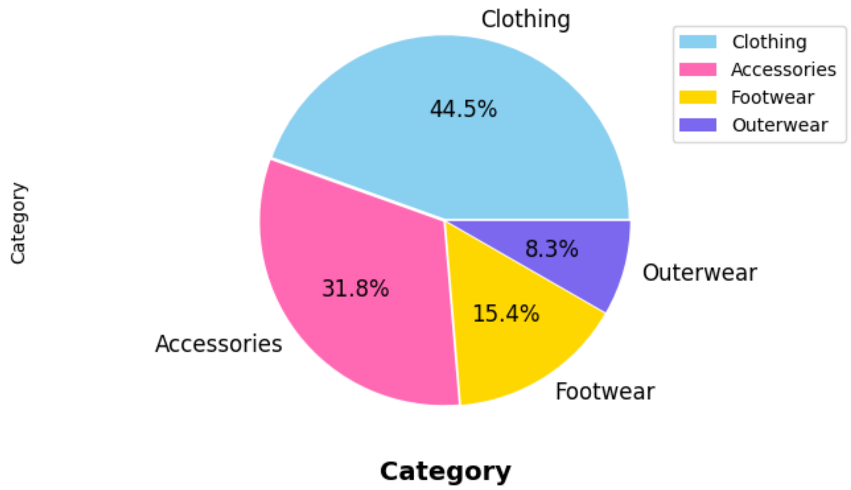
```
df["Category"].value_counts()
```

```
     Clothing       1737
     Accessories    1240
     Footwear        599
     Outerwear       324
     Name: Category, dtype: int64
```

```
ax = df["Category"].value_counts().plot(kind = 'bar',color = colors, rot = 0)
ax.set_xticklabels(('Clothing', 'Accessories', 'Footwear', 'Outerwear'))

for p in ax.patches:
    ax.annotate(int(p.get_height()), (p.get_x() + 0.25, p.get_height() + 1), ha = 'center', va = 'bottom', color = 'black')
    ax.tick_params(axis = 'both', labelsize = 15)
plt.xlabel('Category', weight = "bold", fontsize = 14, labelpad = 20)
plt.ylabel('Number of Occurrences', weight = "bold", fontsize = 14, labelpad = 20);
```

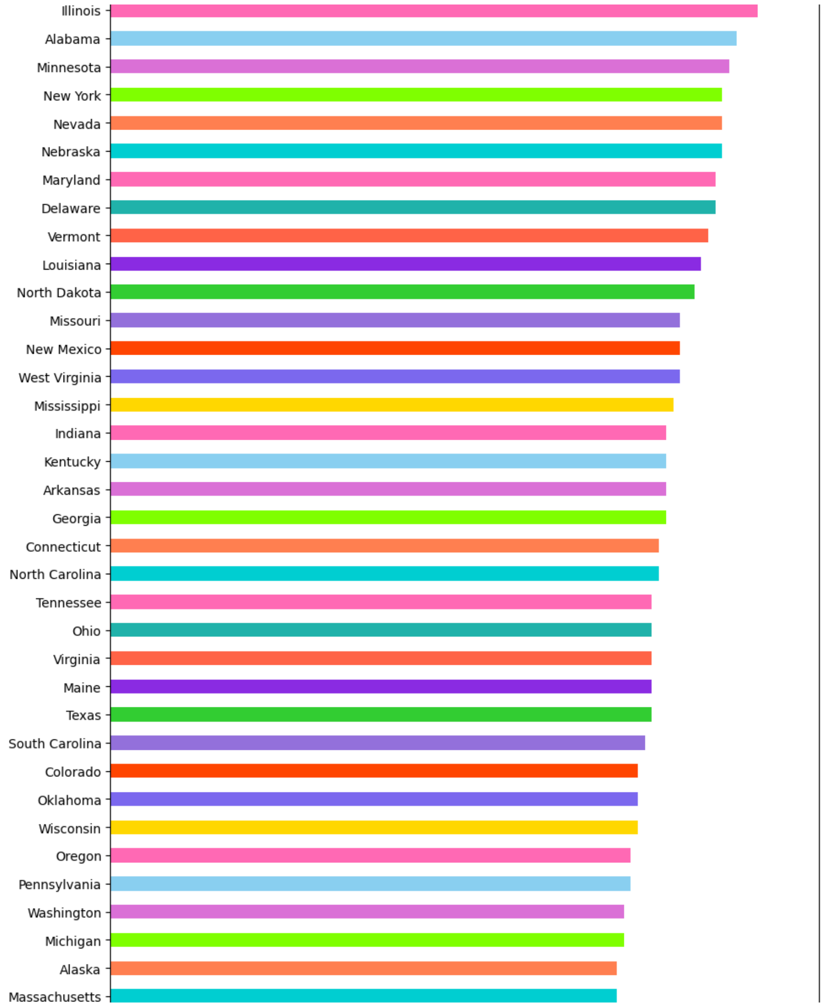


```
plt.figure(figsize = (8, 4))
counts = df["Category"].value_counts()

counts.plot(kind='pie', fontsize=12, colors=colors, explode=(0.01, 0.01, 0.01, 0.01), autopct='%1.1f%%')
plt.xlabel('Category', weight="bold", fontsize=14, labelpad=20)
plt.axis('equal')
plt.legend(labels=counts.index, loc="best")
plt.show()
```



3.LOCATION

```
plt.figure(figsize=(10, 20))
df.Location.value_counts(ascending=True).plot(kind='barh',color=colors)
plt.show()
```

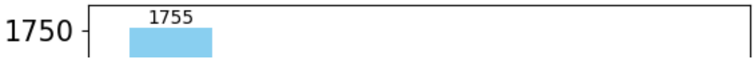


4.Size

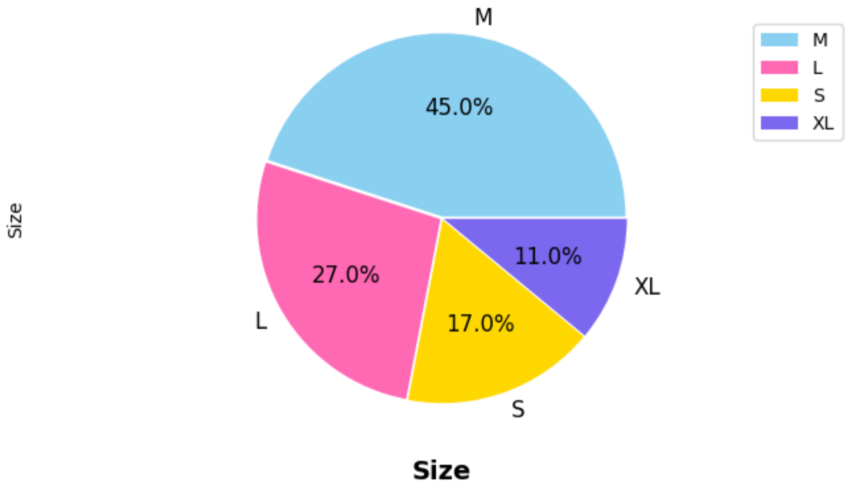
```
df["Size"].value_counts()
```

```
    M     1755
    L     1053
    S      663
    XL     429
    Name: Size, dtype: int64
```

```
ax = df["Size"].value_counts().plot(kind = 'bar', color = colors, rot = 0)
ax.set_xticklabels(('M', 'L', 'S', 'XL'))

for p in ax.patches:
    ax.annotate(int(p.get_height()), (p.get_x() + 0.25, p.get_height() + 1), ha = 'center', va = 'bottom', color = 'black')
    ax.tick_params(axis = 'both', labelsize = 15)
plt.xlabel('Size', weight = "bold", fontsize = 14, labelpad = 20)
plt.ylabel('Number of Occurrences', weight = "bold", fontsize = 14, labelpad = 20);
```



```
plt.figure(figsize = (8, 4))

counts = df["Size"].value_counts()

counts.plot(kind = 'pie', fontsize = 12, colors = colors, explode = (0.01, 0.01, 0.01, 0.01), autopct = '%1.1f%%')
plt.xlabel('Size', weight = "bold", fontsize = 14, labelpad = 20)
plt.axis('equal')
plt.legend(labels = counts.index, loc = "best")
plt.show()
```



5.Seasons

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

```
    Spring    999
    Fall      975
    Winter    971
    Summer    955
    Name: Season, dtype: int64
```

Analyse

```
average_age = df['Age'].mean()
print("Average age:", average_age)
```

```
    Average age: 44.06846153846154
```

```
total_purchase_by_category = df.groupby('Category')['Purchase Amount (USD)'].sum()
print("total purchaseby categories:")
print(total_purchase_by_category)
```

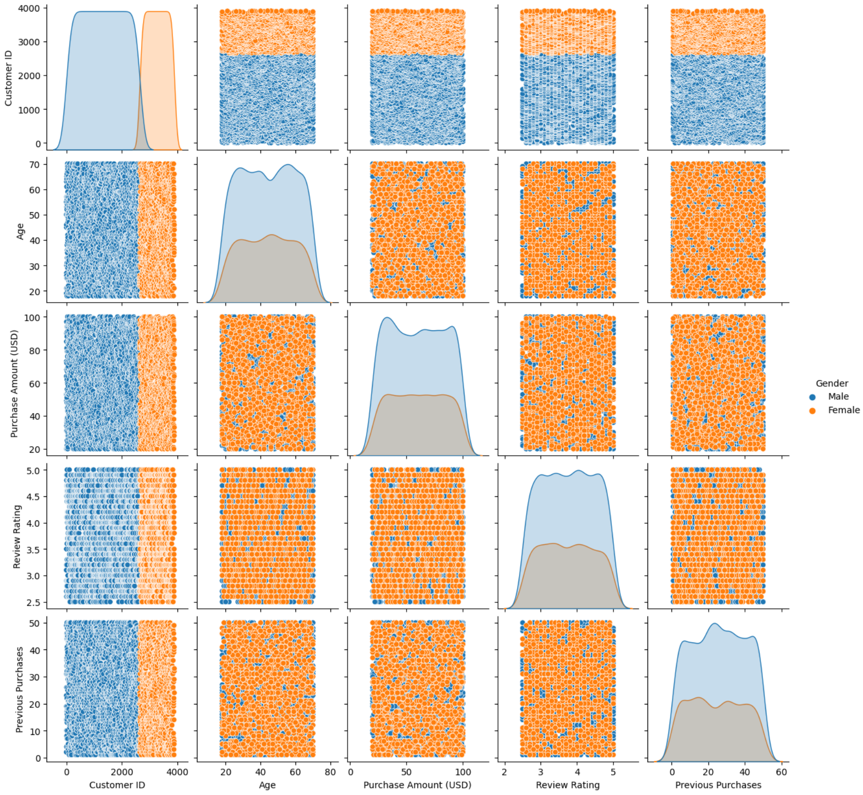
```
    total purchaseby categories:
    Category
    Accessories     74200
    Clothing       104264
    Footwear        36093
    Outerwear       18524
    Name: Purchase Amount (USD), dtype: int64
```

```
most_common_payment_method = df['Payment Method'].mode()[0]
print("most common payement method:", most_common_payment_method)
```

```
    most common payement method: PayPal
```

```
sns.pairplot(df,hue='Gender')
```

```
<seaborn.axisgrid.PairGrid at 0x7adf13b65030>
```



Analyse

```
average_age = df['Age'].mean()
print("Average Age:", average_age)
```

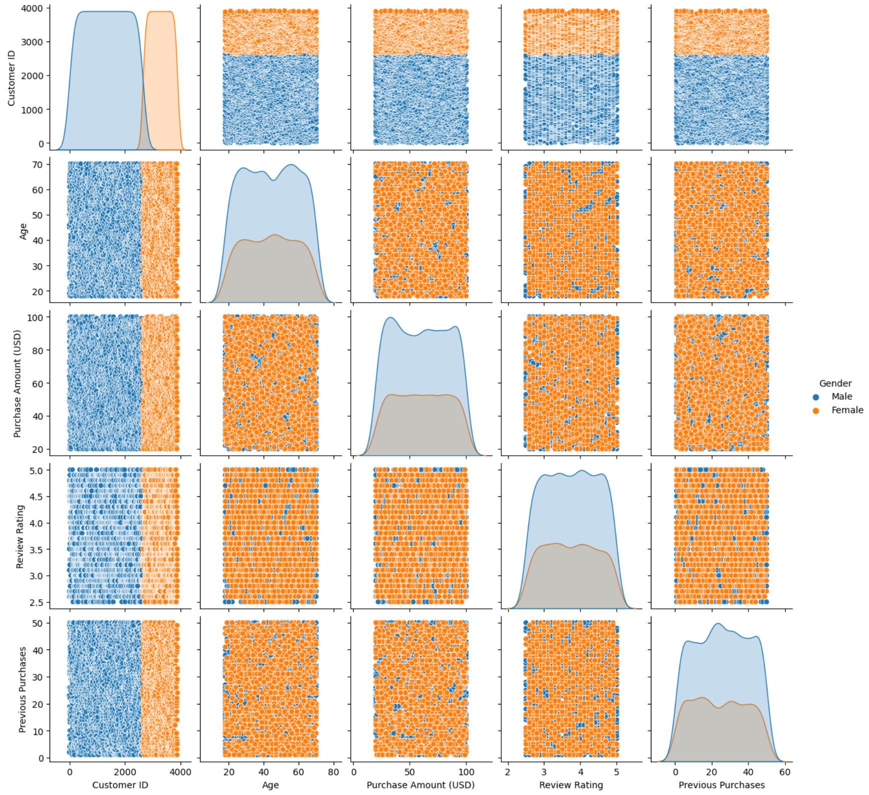
```
    Average Age: 44.06846153846154
```

```
most_common_payment_method = df['Payment Method'].mode()[0]
print("Most common payement method:", most_common_payment_method)
```

```
    Most common payement method: PayPal
```

```
sns.pairplot(df,hue='Gender')
```

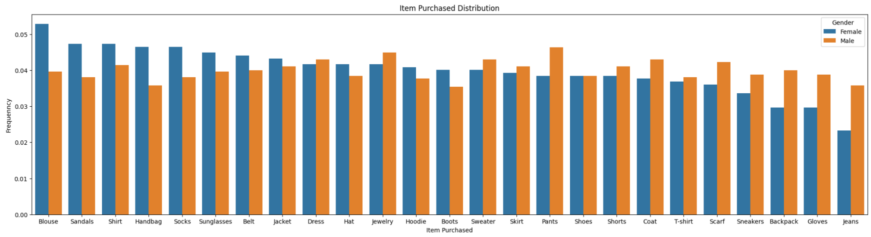
```
<seaborn.axisgrid.PairGrid at 0x7adf13e60250>
```



```
df_item = df['Item Purchased'].groupby(df['Gender']).value_counts(normalize= True).rename('frequency').to_frame().reset_index()
df_item
```

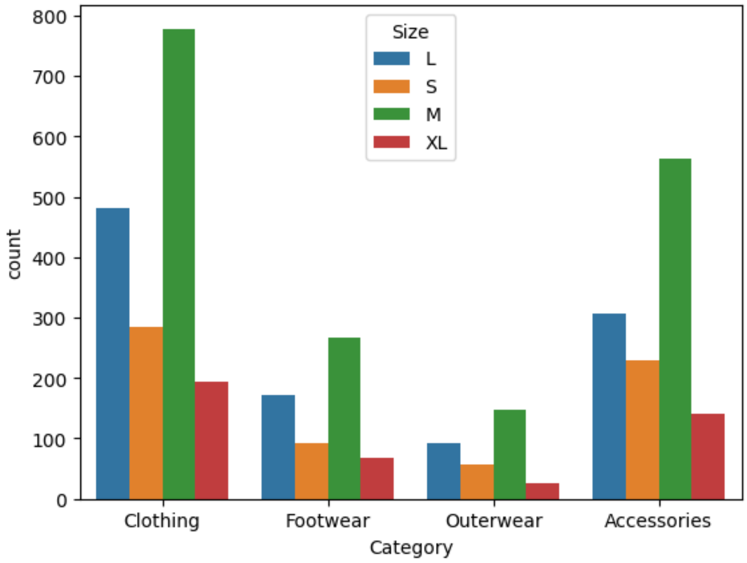
	Gender	Item Purchased	frequency
0	Female	Blouse	0.05
1	Female	Sandals	0.05
2	Female	Shirt	0.05
3	Female	Handbag	0.05
4	Female	Socks	0.05
5	Female	Sunglasses	0.04
6	Female	Belt	0.04
7	Female	Jacket	0.04
8	Female	Dress	0.04
9	Female	Hat	0.04
10	Female	Jewelry	0.04
11	Female	Hoodie	0.04
12	Female	Boots	0.04
13	Female	Sweater	0.04
14	Female	Skirt	0.04
15	Female	Pants	0.04
16	Female	Shoes	0.04
17	Female	Shorts	0.04
18	Female	Coat	0.04
19	Female	T-shirt	0.04
20	Female	Scarf	0.04
21	Female	Sneakers	0.03
22	Female	Backpack	0.03
23	Female	Gloves	0.03
24	Female	Jeans	0.02
25	Male	Pants	0.05
26	Male	Jewelry	0.04
27	Male	Coat	0.04
28	Male	Dress	0.04
29	Male	Sweater	0.04
30	Male	Scarf	0.04
31	Male	Shirt	0.04
32	Male	Jacket	0.04
33	Male	Shorts	0.04
34	Male	Skirt	0.04
35	Male	Backpack	0.04
36	Male	Belt	0.04
37	Male	Blouse	0.04

```
plt.figure(figsize = (25, 6))
sns.barplot(data = df_item,x='Item Purchased',y='frequency',hue='Gender')
plt.xlabel('Item Purchased')
plt.ylabel('Frequenncy')
plt.title("Item Purchased Distribution");
```



```
sns.countplot(x='Category', hue='Size', data=df)
```

```
<Axes: xlabel='Category', ylabel='count'>
```



```
cross_tab = pd.crosstab(df['Payment Method'], df['Age'])
print(cross_tab)
```

```
Age             18  19  20  21  22  23  24  25  26  27  ...  61  62  63  64  \
Payment Method                                          ...                   
Bank Transfer   10  20  10   9  12  11  11  13  11  15  ...   5  12   7  11   
Cash            17  12   7  18  19   8  12  14  12  17  ...  17  13  12  14   
Credit Card     14  16  11   8   8  13  11  14  15  12  ...  11  12  21  13   
Debit Card       6   7  14  12  10  15  11  14  17  14  ...  12   8  12  15   
PayPal          11  13  10   8  13  12  10  18   8  13  ...   8  19  13   8   
Venmo           11  13  10  14   4  12  13  12   6  12  ...  12  19  10  12   

Age             65  66  67  68  69  70  
Payment Method                          
Bank Transfer   11  15   7  14  18  15  
Cash            15   9   8  18  16  10  
Credit Card     16  17  14   9  14  12  
Debit Card       6   9   9  10  22  10  
PayPal          14  11  11   8   7   9  
Venmo           10  10   5  16  11  11  

[6 rows x 53 columns]
```

Prediction

```
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.metrics import accuracy_score, classification_report
```

```
features = ["Age", "Gender", "Category", "Purchase Amount (USD)", "Location", "Shipping Type"]
X = df[features]
y = df["Discount Applied"]
```

```
X = pd.get_dummies(X, columns=["Age", "Gender", "Category", "Purchase Amount (USD)", "Location", "Shipping Type"], drop_first=True)
```

```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
```

```
scaler = StandardScaler()
X_train = scaler.fit_transform(X_train)
X_test = scaler.transform(X_test)
```

```
from sklearn.linear_model import LogisticRegression
classifier = LogisticRegression(random_state = 0, solver='lbfgs', multi_class='auto')
classifier.fit(X_train, y_train)
```

```
▾       LogisticRegression
LogisticRegression(random_state=0)
```

```
y_pred = classifier.predict(X_test)
y_pred
```

```
array(['Yes', 'Yes', 'No', 'No', 'Yes', 'Yes', 'Yes', 'Yes', 'No', 'No',
       'Yes', 'Yes', 'No', 'No', 'No', 'No', 'No', 'Yes', 'No', 'No',
       'Yes', 'Yes', 'Yes', 'Yes', 'Yes', 'No', 'Yes', 'No', 'Yes', 'No',
       'Yes', 'Yes', 'Yes', 'Yes', 'Yes', 'No', 'Yes', 'No', 'No', 'No',
       'No', 'No', 'Yes', 'Yes', 'No', 'Yes', 'Yes', 'Yes', 'Yes', 'Yes',
       'No', 'Yes', 'No', 'Yes', 'Yes', 'Yes', 'Yes', 'No', 'No', 'Yes',
       'No', 'Yes', 'No', 'Yes', 'Yes', 'Yes', 'Yes', 'No', 'Yes', 'No',
       'No', 'No', 'No', 'No', 'Yes', 'No', 'No', 'No', 'Yes', 'No',
       'Yes', 'Yes', 'Yes', 'No', 'Yes', 'Yes', 'Yes', 'No', 'No', 'No',
       'Yes', 'Yes', 'No', 'No', 'Yes', 'No', 'Yes', 'No', 'No', 'Yes',
       'Yes', 'No', 'Yes', 'Yes', 'Yes', 'Yes', 'No', 'Yes', 'No', 'No',
       'Yes', 'No', 'Yes', 'Yes', 'Yes', 'Yes', 'Yes', 'Yes', 'No', 'Yes',
       'Yes', 'Yes', 'Yes', 'No', 'Yes', 'Yes', 'Yes', 'Yes', 'Yes', 'No',
       'Yes', 'Yes', 'Yes', 'No', 'Yes', 'Yes', 'No', 'No', 'Yes', 'Yes',
       'Yes', 'No', 'No', 'Yes', 'Yes', 'No', 'Yes', 'Yes', 'No', 'Yes',
       'Yes', 'Yes', 'Yes', 'No', 'No', 'Yes', 'Yes', 'Yes', 'Yes', 'Yes',
       'No', 'Yes', 'Yes', 'No', 'Yes', 'No', 'No', 'Yes', 'No', 'Yes',
       'No', 'Yes', 'No', 'Yes', 'Yes', 'No', 'No', 'Yes', 'No', 'No',
       'Yes', 'Yes', 'No', 'Yes', 'Yes', 'Yes', 'No', 'Yes', 'Yes', 'No',
       'Yes', 'No', 'Yes', 'No', 'Yes', 'No', 'No', 'Yes', 'Yes', 'No',
       'No', 'Yes', 'Yes', 'No', 'Yes', 'No', 'Yes', 'No', 'No', 'Yes',
       'Yes', 'Yes', 'Yes', 'Yes', 'Yes', 'No', 'No', 'Yes', 'Yes', 'Yes',
       'Yes', 'Yes', 'No', 'No', 'No', 'Yes', 'No', 'Yes', 'No', 'Yes',
       'Yes', 'No', 'No', 'No', 'No', 'No', 'No', 'Yes', 'Yes', 'Yes',
       'No', 'Yes', 'Yes', 'Yes', 'No', 'No', 'Yes', 'No', 'Yes', 'Yes',
       'Yes', 'No', 'No', 'No', 'Yes', 'Yes', 'No', 'Yes', 'No', 'No',
       'No', 'Yes', 'No', 'Yes', 'No', 'No', 'Yes', 'No', 'Yes', 'Yes',
       'Yes', 'No', 'Yes', 'Yes', 'No', 'Yes', 'No', 'Yes', 'No', 'Yes',
       'No', 'Yes', 'Yes', 'No', 'No', 'No', 'Yes', 'No', 'Yes', 'Yes',
       'No', 'Yes', 'No', 'Yes', 'No', 'Yes', 'Yes', 'Yes', 'Yes', 'Yes',
       'No', 'Yes', 'Yes', 'No', 'Yes', 'No', 'No', 'Yes', 'No', 'No',
       'Yes', 'Yes', 'No', 'Yes', 'Yes', 'Yes', 'No', 'Yes', 'Yes', 'Yes',
       'Yes', 'No', 'No', 'Yes', 'No', 'No', 'No', 'No', 'Yes', 'Yes',
       'No', 'No', 'Yes', 'No', 'Yes', 'No', 'No', 'No', 'No', 'Yes',
       'No', 'Yes', 'Yes', 'Yes', 'Yes', 'No', 'Yes', 'Yes', 'No', 'No',
       'No', 'Yes', 'Yes', 'Yes', 'Yes', 'Yes', 'No', 'Yes', 'Yes', 'Yes',
       'Yes', 'Yes', 'No', 'Yes', 'Yes', 'Yes', 'No', 'Yes', 'Yes', 'No',
       'No', 'Yes', 'No', 'No', 'Yes', 'No', 'Yes', 'Yes', 'No', 'No',
       'No', 'Yes', 'No', 'No', 'Yes', 'Yes', 'No', 'Yes', 'No', 'Yes',
       'Yes', 'Yes', 'Yes', 'No', 'Yes', 'Yes', 'No', 'Yes', 'Yes', 'No',
       'Yes', 'Yes', 'No', 'No', 'No', 'Yes', 'Yes', 'No', 'Yes', 'Yes',
       'Yes', 'Yes', 'No', 'Yes', 'Yes', 'Yes', 'Yes', 'Yes', 'Yes', 'No',
       'No', 'No', 'No', 'Yes', 'No', 'Yes', 'Yes', 'Yes', 'No', 'No',
       'Yes', 'No', 'Yes', 'Yes', 'Yes', 'No', 'Yes', 'Yes', 'No', 'Yes',
       'No', 'Yes', 'Yes', 'No', 'Yes', 'No', 'Yes', 'Yes', 'Yes', 'Yes',
       'No', 'No', 'No', 'No', 'No', 'No', 'No', 'No', 'No', 'No', 'No',
       'No', 'No', 'Yes', 'Yes', 'Yes', 'No', 'Yes', 'No', 'Yes', 'No',
       'Yes', 'Yes', 'Yes', 'Yes', 'Yes', 'No', 'No', 'No', 'Yes', 'Yes',
       'Yes', 'Yes', 'Yes', 'Yes', 'Yes', 'Yes', 'Yes', 'No', 'No', 'Yes',
       'Yes', 'Yes', 'Yes', 'No', 'Yes', 'Yes', 'Yes', 'No', 'No', 'Yes',
       'No', 'Yes', 'No', 'No', 'Yes', 'Yes', 'No', 'Yes', 'Yes', 'Yes',
       'Yes', 'Yes', 'No', 'No', 'Yes', 'No', 'Yes', 'No', 'Yes', 'No',
       'No', 'Yes', 'No', 'Yes', 'Yes', 'Yes', 'No', 'Yes', 'No', 'No',
       'No', 'Yes', 'No', 'Yes', 'No', 'No', 'Yes', 'No', 'Yes', 'No',
       'No', 'No', 'Yes', 'No', 'Yes', 'Yes', 'Yes', 'Yes', 'Yes', 'No',
       'Yes', 'Yes', 'Yes', 'No', 'Yes', 'No', 'No', 'No', 'Yes', 'No',
       'Yes', 'Yes', 'No', 'No', 'No', 'Yes', 'Yes', 'Yes', 'Yes', 'Yes',
       'Yes', 'Yes', 'Yes', 'Yes', 'No', 'Yes', 'Yes', 'Yes', 'Yes', 'No',
```

```
probs_y=classifier.predict_proba(X_test)
probs_y
```

```
array([[4.20313584e-01, 5.79686416e-01],
       [2.63196904e-01, 7.36803096e-01],
       [5.49545065e-01, 4.50454935e-01],
       ...,
       [9.99681770e-01, 3.18230008e-04],
       [4.09163171e-01, 5.90836829e-01],
       [5.67828894e-01, 4.32171106e-01]])
```

```
probs_y = np.round(probs_y, 2)


accuracy = accuracy_score(y_test, y_pred)
print("Accuracy:", accuracy)
print(classification_report(y_test, y_pred))
```

```
Accuracy: 0.6833333333333333
              precision    recall  f1-score   support

          No       0.75      0.62      0.68       422
         Yes       0.63      0.76      0.69       358

    accuracy                           0.68       780
   macro avg       0.69      0.69      0.68       780
weighted avg       0.70      0.68      0.68       780
```

```
res = "{:<10} | {:<10} | {:<10} | {:<13} | {:<5}".format("y_test", "y_pred", "Setosa(%)", "versicolor(%)", "virginica(%)\n")
res += "-"*65+"\n"
res += "\n".join("{:<10} | {:<10} | {:<10} | {:<13} | {:<10}".format(x, y, a, b, c) for x, y, a, b, c in zip(y_test, y_pred, probs_y[:,(
res += "\n"+"-"*65+"\n"
print(res)
```

```
No         | No         | 1.0        | 0.0           | 0.0
Yes        | Yes        | 0.47       | 0.53          | 0.53
No         | No         | 1.0        | 0.0           | 0.0
Yes        | Yes        | 0.36       | 0.64          | 0.64
No         | No         | 0.52       | 0.48          | 0.48
Yes        | No         | 0.61       | 0.39          | 0.39
Yes        | Yes        | 0.42       | 0.58          | 0.58
Yes        | No         | 0.7        | 0.3           | 0.3
No         | Yes        | 0.3        | 0.7           | 0.7
Yes        | Yes        | 0.44       | 0.56          | 0.56
Yes        | Yes        | 0.35       | 0.65          | 0.65
No         | No         | 1.0        | 0.0           | 0.0
No         | Yes        | 0.21       | 0.79          | 0.79
No         | Yes        | 0.12       | 0.88          | 0.88
No         | No         | 1.0        | 0.0           | 0.0
Yes        | Yes        | 0.38       | 0.62          | 0.62
No         | No         | 1.0        | 0.0           | 0.0
No         | Yes        | 0.39       | 0.61          | 0.61
No         | No         | 1.0        | 0.0           | 0.0
Yes        | Yes        | 0.44       | 0.56          | 0.56
No         | No         | 1.0        | 0.0           | 0.0
Yes        | Yes        | 0.41       | 0.59          | 0.59
No         | Yes        | 0.31       | 0.69          | 0.69
Yes        | No         | 0.55       | 0.45          | 0.45
No         | No         | 1.0        | 0.0           | 0.0
Yes        | No         | 0.54       | 0.46          | 0.46
Yes        | Yes        | 0.39       | 0.61          | 0.61
No         | No         | 0.64       | 0.36          | 0.36
No         | Yes        | 0.38       | 0.62          | 0.62
No         | Yes        | 0.41       | 0.59          | 0.59
No         | No         | 1.0        | 0.0           | 0.0
Yes        | Yes        | 0.4        | 0.6           | 0.6
No         | No         | 1.0        | 0.0           | 0.0
Yes        | Yes        | 0.23       | 0.77          | 0.77
No         | No         | 1.0        | 0.0           | 0.0
Yes        | Yes        | 0.49       | 0.51          | 0.51
No         | Yes        | 0.15       | 0.85          | 0.85
Yes        | Yes        | 0.27       | 0.73          | 0.73
Yes        | Yes        | 0.44       | 0.56          | 0.56
No         | Yes        | 0.24       | 0.76          | 0.76
No         | No         | 1.0        | 0.0           | 0.0
No         | Yes        | 0.48       | 0.52          | 0.52
Yes        | Yes        | 0.16       | 0.84          | 0.84
No         | No         | 1.0        | 0.0           | 0.0
Yes        | Yes        | 0.21       | 0.79          | 0.79
Yes        | No         | 0.59       | 0.41          | 0.41
Yes        | No         | 0.71       | 0.29          | 0.29
No         | Yes        | 0.34       | 0.66          | 0.66
No         | No         | 1.0        | 0.0           | 0.0
No         | No         | 1.0        | 0.0           | 0.0
Yes        | Yes        | 0.43       | 0.57          | 0.57
No         | Yes        | 0.23       | 0.77          | 0.77
No         | No         | 0.59       | 0.41          | 0.41
Yes        | Yes        | 0.37       | 0.63          | 0.63
Yes        | Yes        | 0.48       | 0.52          | 0.52
No         | Yes        | 0.43       | 0.57          | 0.57
No         | No         | 1.0        | 0.0           | 0.0
Yes        | Yes        | 0.39       | 0.61          | 0.61
No         | Yes        | 0.48       | 0.52          | 0.52
```

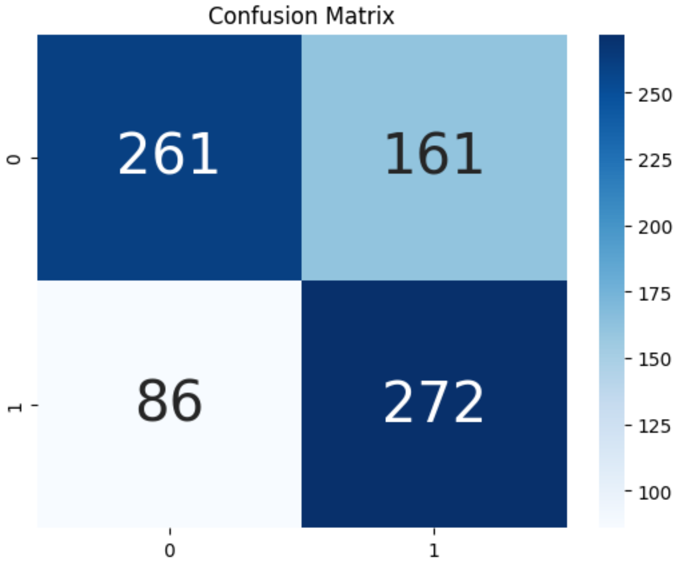
Making the Confusion Matrix :

```
from sklearn.metrics import confusion_matrix
cm = confusion_matrix(y_test, y_pred)
print(cm)
```

```
[[261 161]
 [ 86 272]]
```

Plot confusion matrix :

```
import seaborn as sns
import pandas as pd
ax = plt.axes()
df_cm = cm
sns.heatmap(df_cm, annot=True, annot_kws={"size": 30}, fmt='d',cmap="Blues", ax = ax )
ax.set_title('Confusion Matrix')
plt.show()
```



```
y_prob = classifier.predict_proba(X_test)[:, 1]
```

```
plt.figure(figsize=(8, 6))

plt.plot([0, 1], [0, 1], color='navy', lw=2, linestyle='--')
plt.xlim([0.0, 1.0])
plt.ylim([0.0, 1.05])
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.title('Receiver Operating Characteristic')
plt.legend(loc="lower right")
plt.show()
```