Importing Libraries

In [1]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import warnings
warnings.filterwarnings("ignore")
```

Loading the Dataset

In [2]:
```
data = pd.read_csv(r"C:\Users\nandh\Downloads\shopping data set\shopping_trends_update
data.sample(5)
```

Out[2]:

	Customer ID	Age	Gender	Item Purchased	Category	Purchase Amount (USD)	Location	Size	Color	Season	R
2983	2984	60	Female	Pants	Clothing	44	West Virginia	M	Charcoal	Winter	
2271	2272	29	Male	Dress	Clothing	99	Alaska	M	Green	Summer	
3814	3815	45	Female	Dress	Clothing	95	Michigan	M	Orange	Winter	
3287	3288	24	Female	Skirt	Clothing	63	West Virginia	L	Black	Spring	
3567	3568	32	Female	Scarf	Accessories	56	New Mexico	S	Silver	Summer	

Checking the shape of the Dataset

In [3]:
```
data.shape
```

Out[3]:
```
(3900, 18)
```

Checking the number of Columns in the Dataset

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

Out[4]:
```
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')
```

Information about the Dataset

In [5]:
```
data.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
```

In [6]:
```
data.describe()
```

Out[6]:

	Customer ID	Age	Purchase Amount (USD)	Review Rating	Previous Purchases
count	3900.000000	3900.000000	3900.000000	3900.000000	3900.000000
mean	1950.500000	44.068462	59.764359	3.749949	25.351538
std	1125.977353	15.207589	23.685392	0.716223	14.447125
min	1.000000	18.000000	20.000000	2.500000	1.000000
25%	975.750000	31.000000	39.000000	3.100000	13.000000
50%	1950.500000	44.000000	60.000000	3.700000	25.000000
75%	2925.250000	57.000000	81.000000	4.400000	38.000000
max	3900.000000	70.000000	100.000000	5.000000	50.000000

Checking if there are any null values present in the dataset or not ?

In [7]:
```
data.isnull().sum()
```

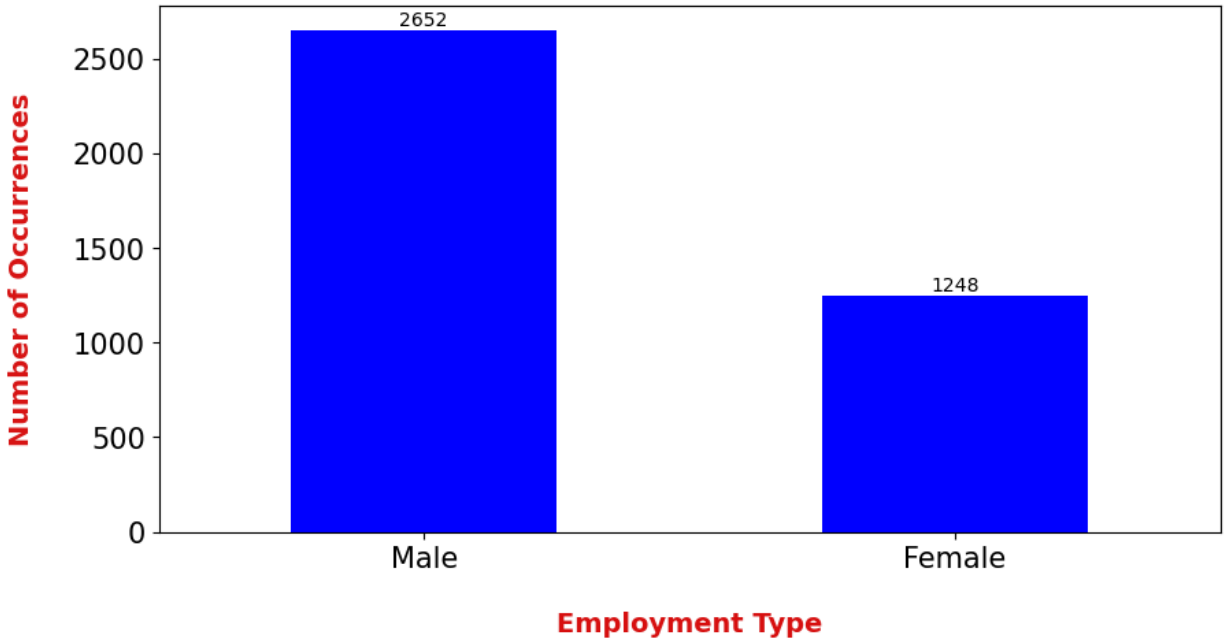
Out[7]:
```
Customer ID               0
Age                       0
Gender                    0
Item Purchased            0
Category                  0
Purchase Amount (USD)     0
Location                  0
Size                      0
Color                     0
Season                    0
Review Rating             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
```

Checking if there are any duplicate values present in the dataset or not ?

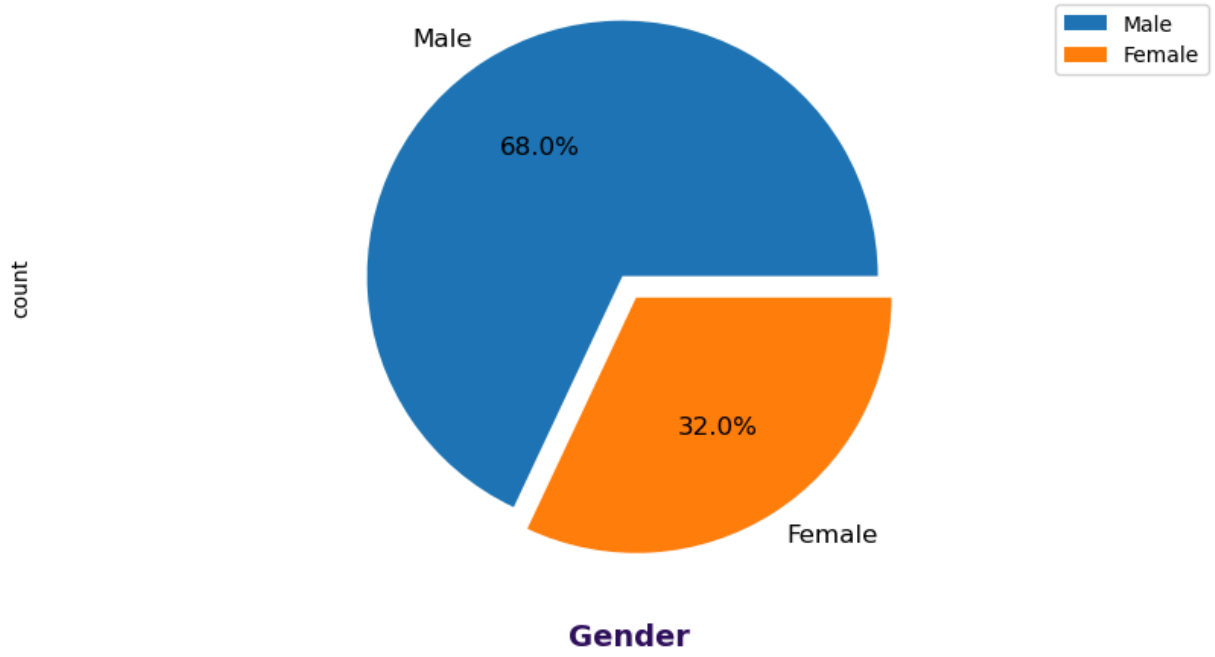
In [8]:
```
data.duplicated().sum()
```

Out[8]:
```
0
```

In [9]:
```
plt.figure(figsize = (10, 5))
ax = data["Gender"].value_counts().plot(kind = 'bar', color = 'Blue', 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 = 'cer
    ax.tick_params(axis = 'both', labelsize = 15)
plt.xlabel('Employment Type', weight = "bold", color = "#D71313", fontsize = 14, label
plt.ylabel('Number of Occurrences', weight = "bold", color = "#D71313", fontsize = 14,
```

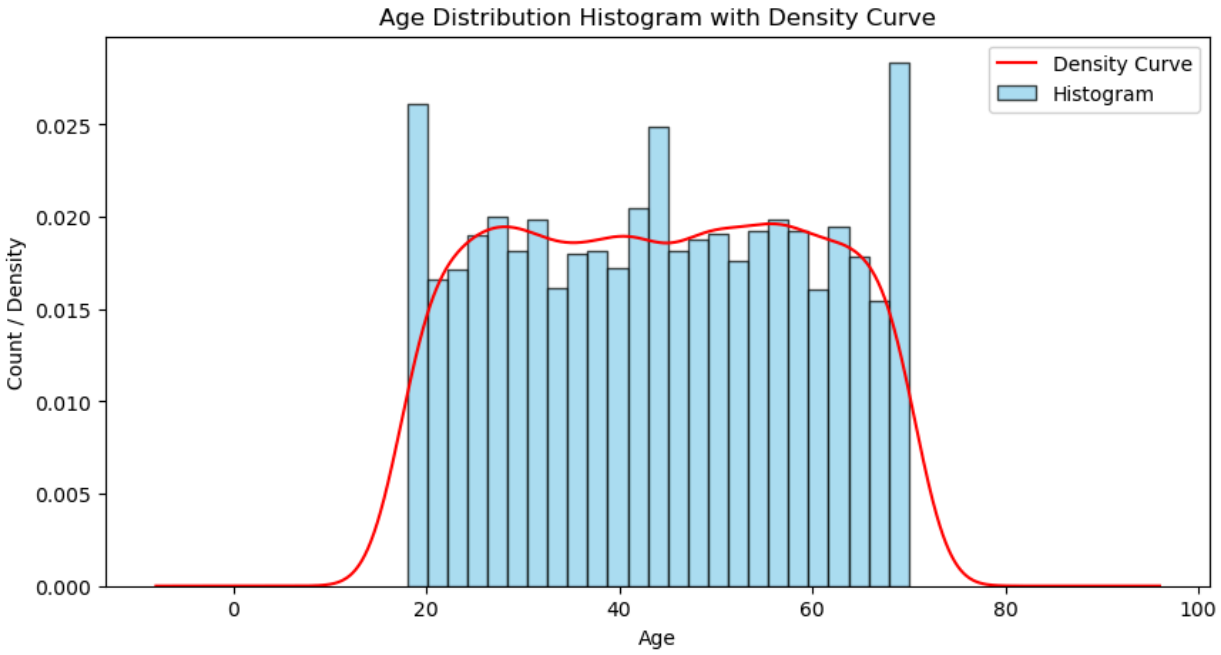


In [10]:
```
plt.figure(figsize = (10, 5))

counts = data["Gender"].value_counts()
explode = (0, 0.1)

counts.plot(kind = 'pie', fontsize = 12, explode = explode, autopct = '%1.1f%%')
plt.xlabel('Gender', weight = "bold", color = "#2F0F5D", fontsize = 14, labelpad = 20)
plt.axis('equal')
plt.legend(labels = counts.index, loc = "best")
plt.show()
```



In [11]:
```
fig, ax = plt.subplots(figsize = (10, 5))

ax.hist(data['Age'], bins = 25, edgecolor = 'black', alpha = 0.7, color = 'skyblue', c
data['Age'].plot(kind = 'kde', color = 'red')
```

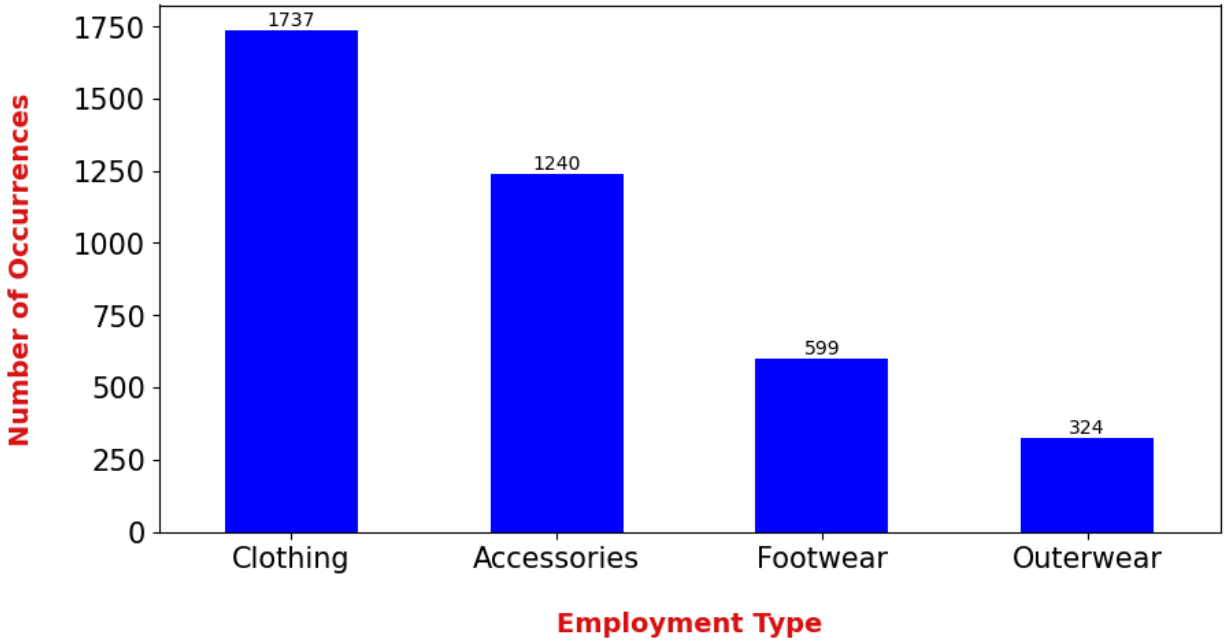
```
ax.set_xlabel('Age')
ax.set_ylabel('Count / Density')
ax.set_title('Age Distribution Histogram with Density Curve')
ax.legend(['Density Curve', 'Histogram'])
plt.show()
```



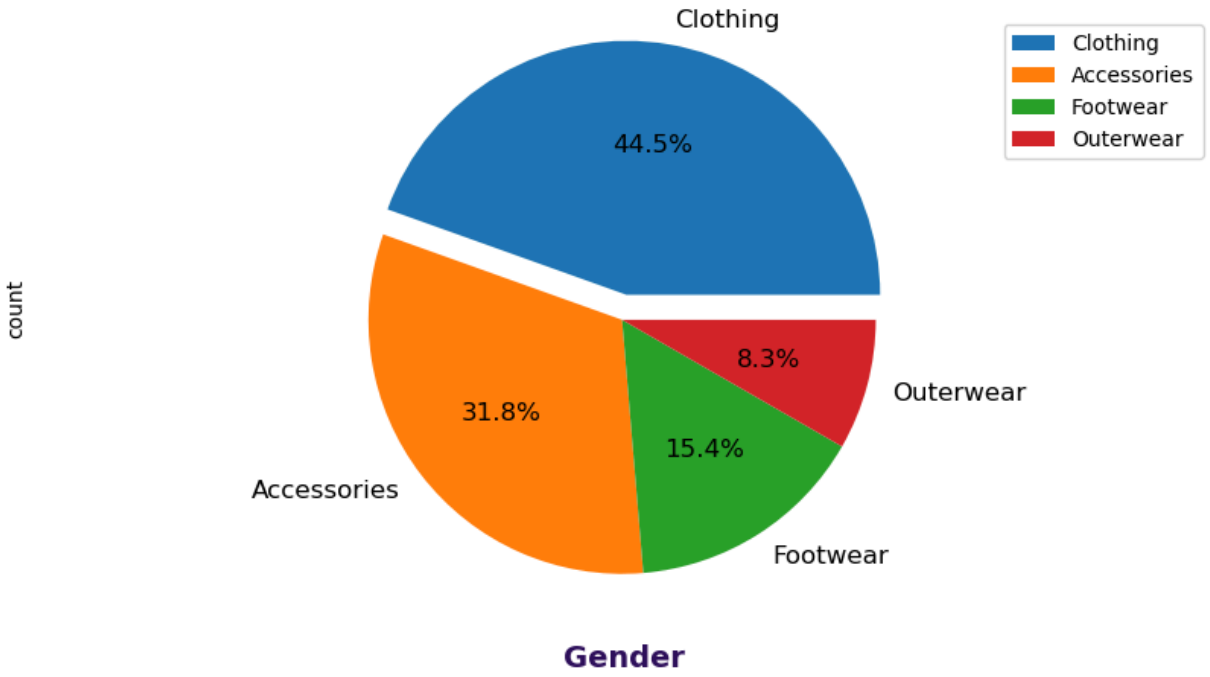
In [12]:
```
data["Category"].value_counts()
```

Out[12]:
```
Category
Clothing       1737
Accessories    1240
Footwear        599
Outerwear       324
Name: count, dtype: int64
```

In [13]:
```
plt.figure(figsize = (10, 5))
ax = data["Category"].value_counts().plot(kind = 'bar', color = 'Blue', 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 = 'cer
    ax.tick_params(axis = 'both', labelsize = 15)
plt.xlabel('Employment Type', weight = "bold", color = "#D71313", fontsize = 14, label
plt.ylabel('Number of Occurrences', weight = "bold", color = "#D71313", fontsize = 14,
```



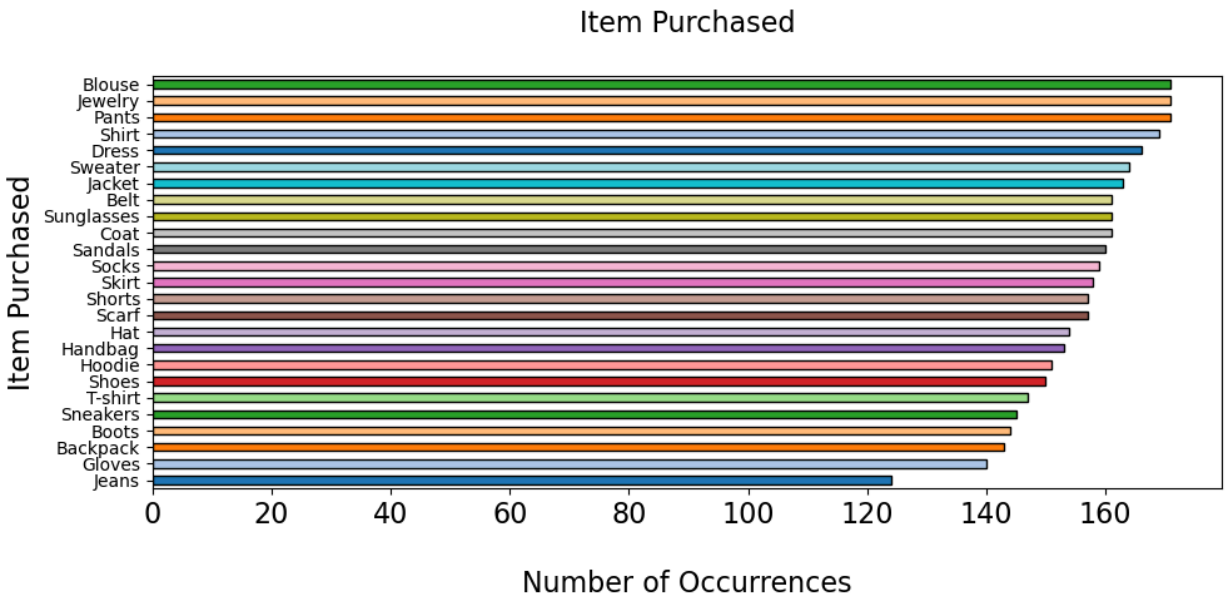
In [14]:
```
plt.figure(figsize = (10, 5))

counts = data["Category"].value_counts()
explode = (0.1, 0.0, 0.0, 0.0)

counts.plot(kind = 'pie', fontsize = 12, explode = explode, autopct = '%1.1f%%')
plt.xlabel('Gender', weight = "bold", color = "#2F0F5D", fontsize = 14, labelpad = 20)
plt.axis('equal')
plt.legend(labels = counts.index, loc = "best")
plt.show()
```



In [15]:
```
data["Item Purchased"].value_counts()
```

Out[15]:
```
Item Purchased
Blouse        171
Jewelry       171
Pants         171
Shirt         169
Dress         166
Sweater       164
Jacket        163
Belt          161
Sunglasses    161
Coat          161
Sandals       160
Socks         159
Skirt         158
Shorts        157
Scarf         157
Hat           154
Handbag       153
Hoodie        151
Shoes         150
T-shirt       147
Sneakers      145
Boots         144
Backpack      143
Gloves        140
Jeans         124
Name: count, dtype: int64
```

In [16]:
```
plt.figure(figsize = (10, 5))
data["Item Purchased"].value_counts().sort_values(ascending = True).plot(kind = 'barh'
plt.ylabel('Item Purchased', fontsize = 16)
plt.xlabel('\nNumber of Occurrences', fontsize = 16)
plt.title('Item Purchased\n', fontsize = 16)
plt.xticks(rotation = 0, ha = 'center', fontsize = 16)
plt.tight_layout()
plt.show()
```

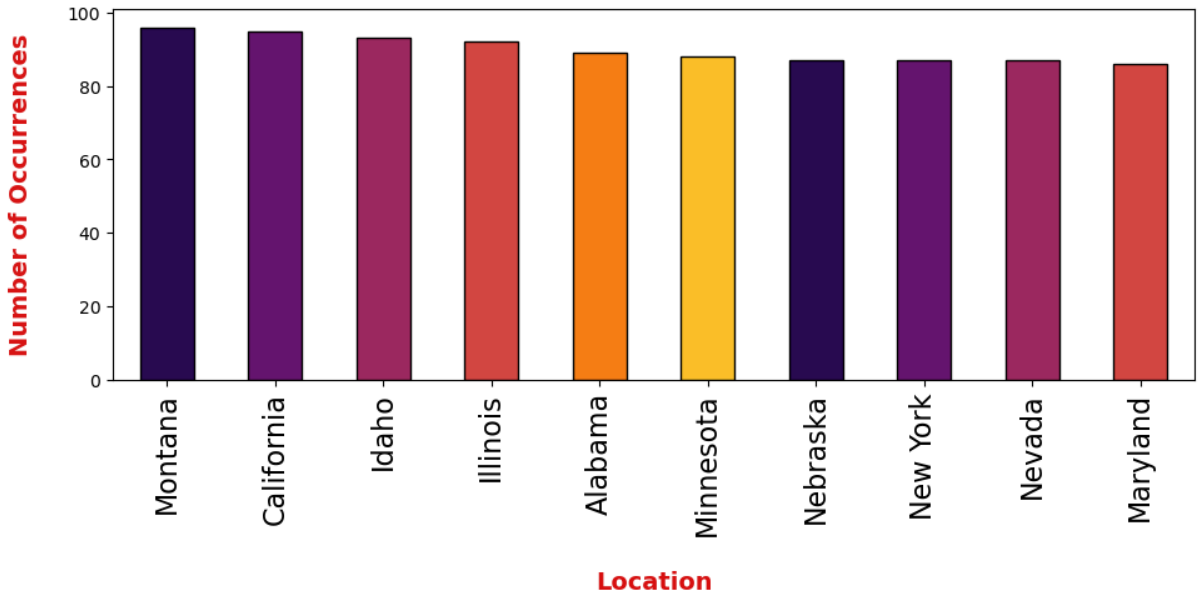


In [17]:
```
data["Location"].value_counts()
```

Out[17]:
```
Location
Montana           96
California        95
Idaho             93
Illinois          92
Alabama           89
Minnesota         88
Nebraska          87
New York          87
Nevada            87
Maryland          86
Delaware          86
Vermont           85
Louisiana         84
North Dakota      83
Missouri          81
West Virginia     81
New Mexico        81
Mississippi       80
Indiana           79
Georgia           79
Kentucky          79
Arkansas          79
North Carolina    78
Connecticut       78
Virginia          77
Ohio              77
Tennessee         77
Texas             77
Maine             77
South Carolina    76
Colorado          75
Oklahoma          75
Wisconsin         75
Oregon            74
Pennsylvania      74
Washington        73
Michigan          73
Alaska            72
Massachusetts     72
Wyoming           71
Utah              71
New Hampshire     71
South Dakota      70
Iowa              69
Florida           68
New Jersey        67
Hawaii            65
Arizona           65
Kansas            63
Rhode Island      63
Name: count, dtype: int64
```

In [18]:
```
plt.figure(figsize = (10, 5))
data["Location"].value_counts()[:10].sort_values(ascending = False).plot(kind = 'bar',
plt.xlabel('Location', weight = "bold", color = "#D71313", fontsize = 14, labelpad = 2
plt.ylabel('\nNumber of Occurrences', weight = "bold", color = "#D71313", fontsize = 1
plt.xticks(rotation = 90, ha = 'center', fontsize = 16)
```

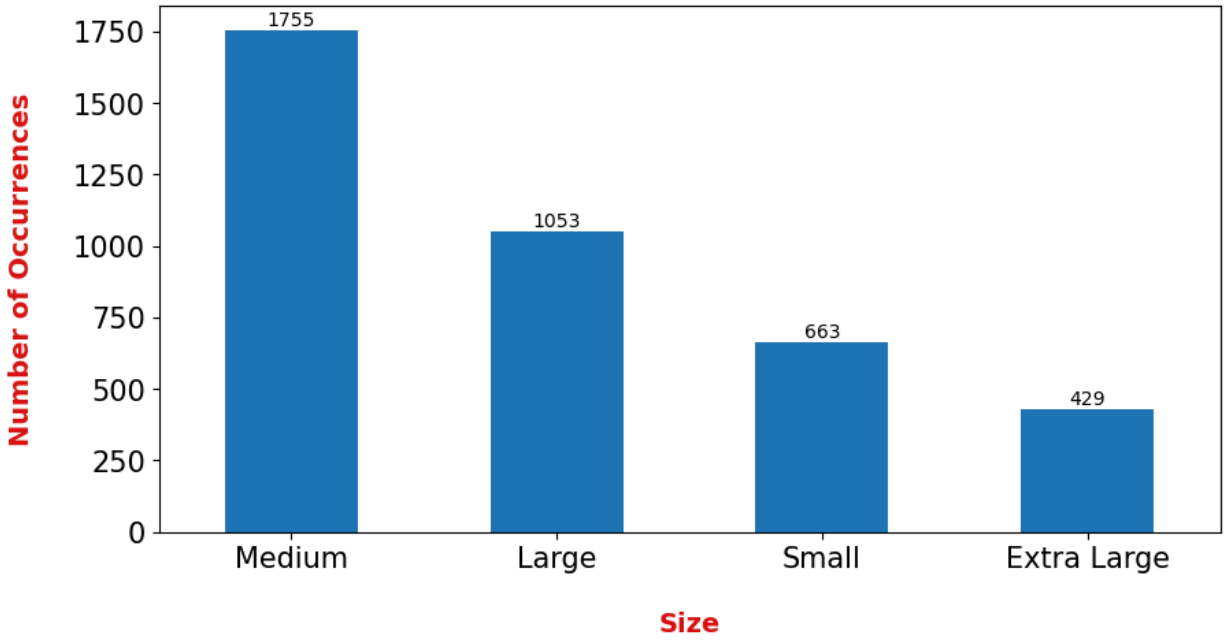
```
plt.tight_layout()
plt.show()
```



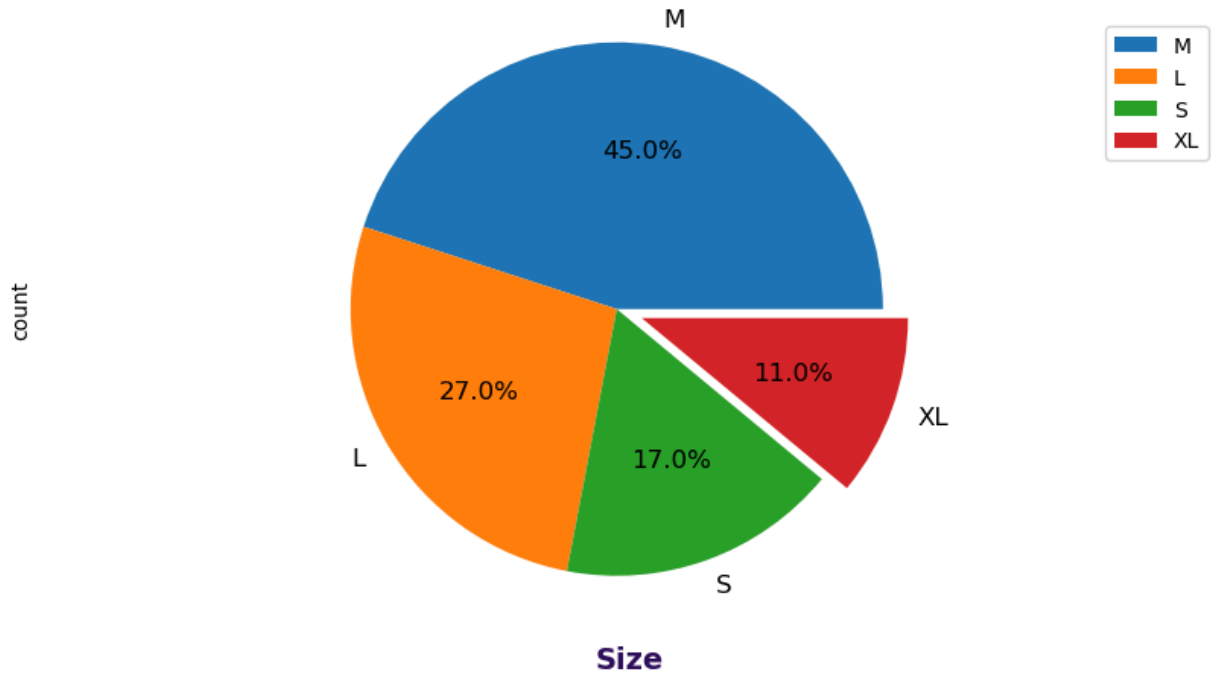
In [19]:
```
data["Size"].value_counts()
```

Out[19]:
```
Size
M     1755
L     1053
S      663
XL     429
Name: count, dtype: int64
```

In [20]:
```
plt.figure(figsize = (10, 5))
ax = data["Size"].value_counts().plot(kind = 'bar', rot = 0)
ax.set_xticklabels(('Medium', 'Large', 'Small', 'Extra Large'))

for p in ax.patches:
    ax.annotate(int(p.get_height()), (p.get_x() + 0.25, p.get_height() + 1), ha = 'cer
    ax.tick_params(axis = 'both', labelsize = 15)
plt.xlabel('Size', weight = "bold", color = "#D71313", fontsize = 14, labelpad = 20)
plt.ylabel('Number of Occurrences', weight = "bold", color = "#D71313", fontsize = 14,
```



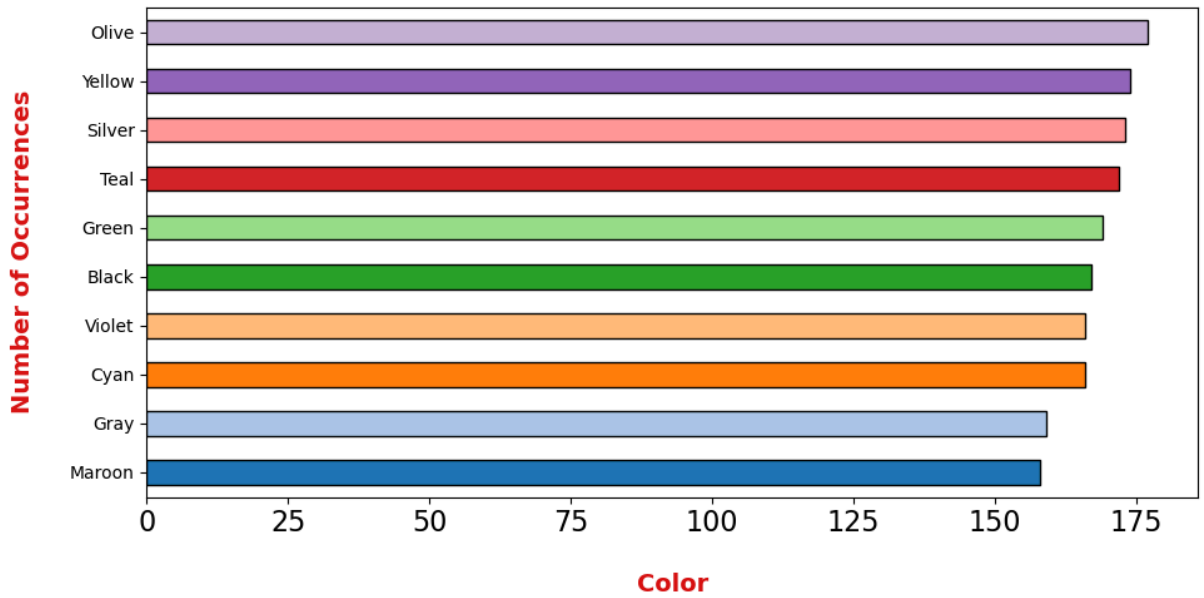
In [21]:
```
plt.figure(figsize = (10, 5))

counts = data["Size"].value_counts()
explode = (0, 0.0, 0.0, 0.1)

counts.plot(kind = 'pie', fontsize = 12, explode = explode, autopct = '%1.1f%%')
plt.xlabel('Size', weight = "bold", color = "#2F0F5D", fontsize = 14, labelpad = 20)
plt.axis('equal')
plt.legend(labels = counts.index, loc = "best")
plt.show()
```



In [22]:
```
data["Color"].value_counts()
```

Out[22]:
```
Color
Olive        177
Yellow       174
Silver       173
Teal         172
Green        169
Black        167
Cyan         166
Violet       166
Gray         159
Maroon       158
Orange       154
Charcoal     153
Pink         153
Magenta      152
Blue         152
Purple       151
Peach        149
Red          148
Beige        147
Indigo       147
Lavender     147
Turquoise    145
White        142
Brown        141
Gold         138
Name: count, dtype: int64
```

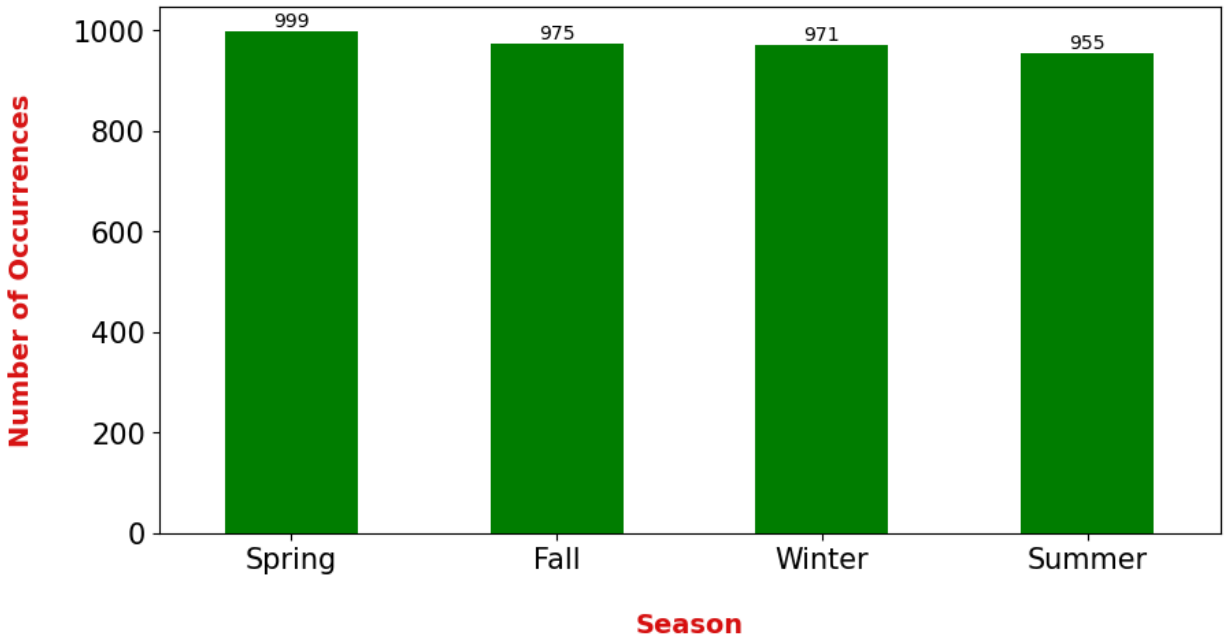
In [23]:
```
plt.figure(figsize = (10, 5))
data["Color"].value_counts()[:10].sort_values(ascending = True).plot(kind = 'barh', co
plt.xlabel('Color', weight = "bold", color = "#D71313", fontsize = 14, labelpad = 20)
plt.ylabel('\nNumber of Occurrences', weight = "bold", color = "#D71313", fontsize = 1
plt.xticks(rotation = 0, ha = 'center', fontsize = 16)
plt.tight_layout()
plt.show()
```



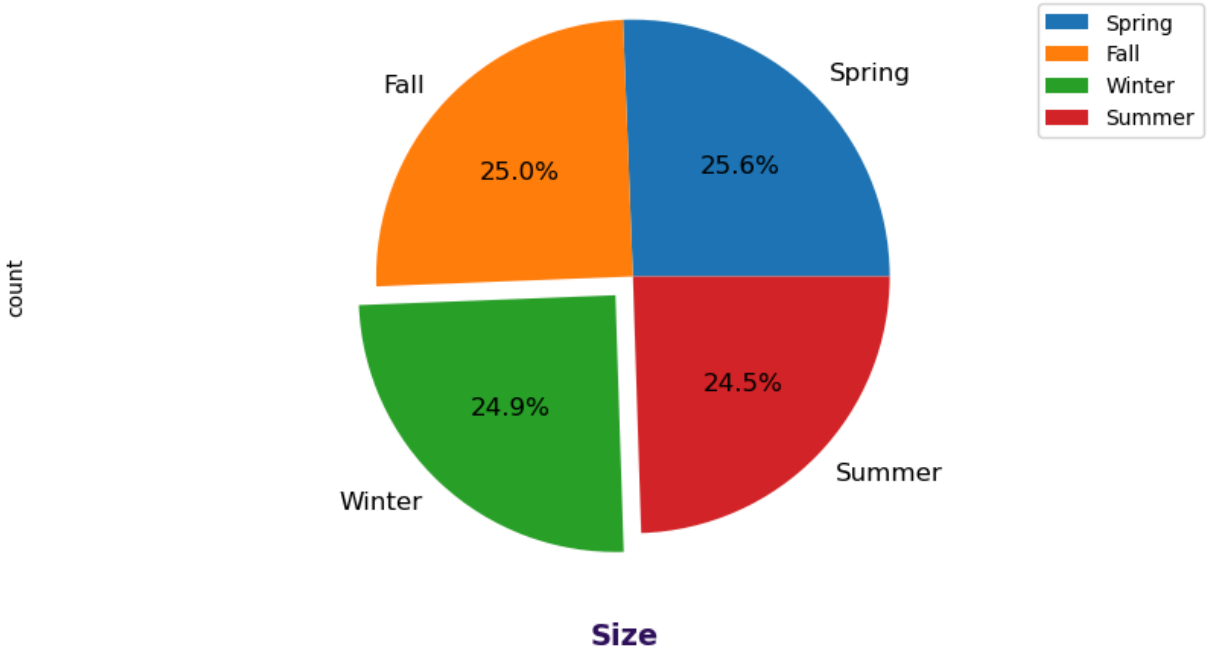
In [24]:
```
data["Season"].value_counts()
```

Out[24]:
```
Season
Spring    999
Fall      975
Winter    971
Summer    955
Name: count, dtype: int64
```

In [25]:
```
plt.figure(figsize = (10, 5))
ax = data["Season"].value_counts().plot(kind = 'bar', color='Green', rot = 0)
ax.set_xticklabels(('Spring', 'Fall', 'Winter', 'Summer'))

for p in ax.patches:
    ax.annotate(int(p.get_height()), (p.get_x() + 0.25, p.get_height() + 1), ha = 'cer
    ax.tick_params(axis = 'both', labelsize = 15)
plt.xlabel('Season', weight = "bold", color = "#D71313", fontsize = 14, labelpad = 20)
plt.ylabel('Number of Occurrences', weight = "bold", color = "#D71313", fontsize = 14,
```



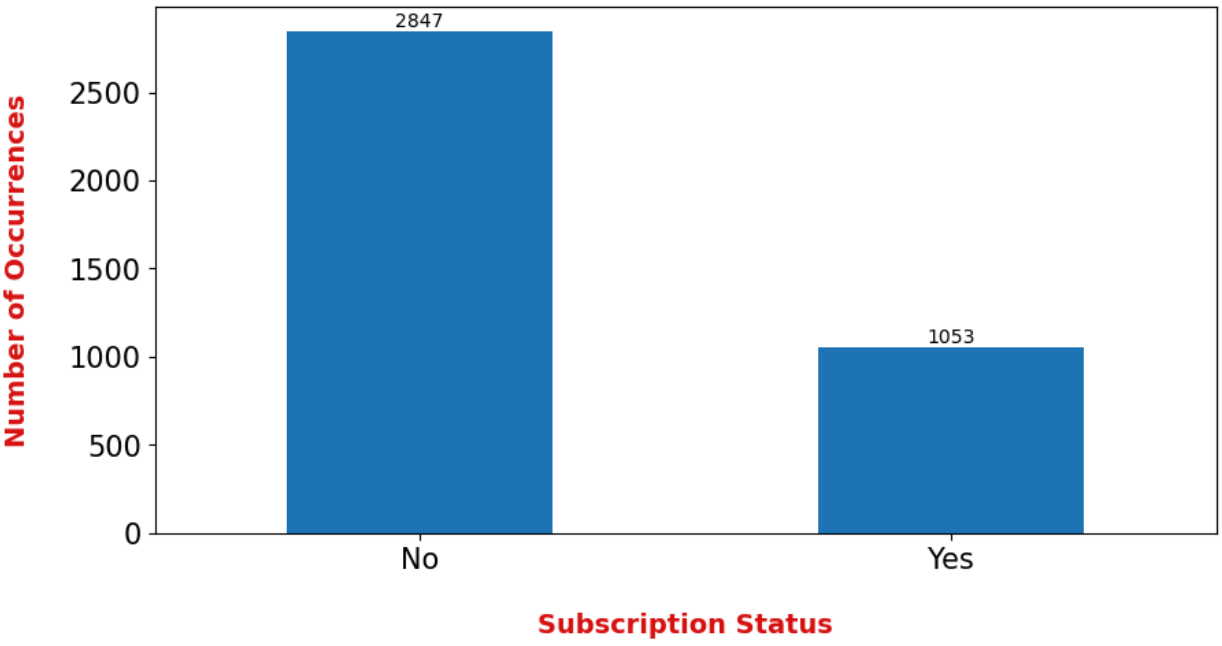
In [26]:
```
plt.figure(figsize = (10, 5))

counts = data["Season"].value_counts()
explode = (0, 0, 0.1, 0)

counts.plot(kind = 'pie', fontsize = 12, explode = explode, autopct = '%1.1f%%')
plt.xlabel('Size', weight = "bold", color = "#2F0F5D", fontsize = 14, labelpad = 20)
plt.axis('equal')
plt.legend(labels = counts.index, loc = "best")
plt.show()
```



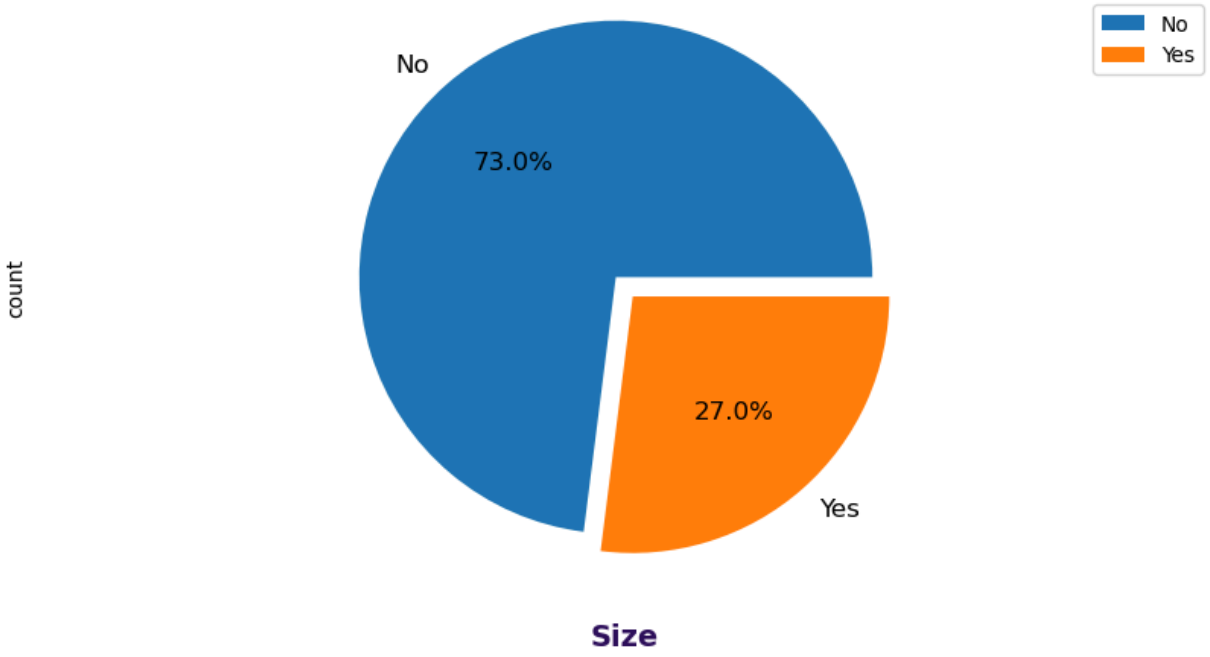
In [27]:
```
data["Subscription Status"].value_counts()
```

Out[27]:
```
Subscription Status
No     2847
Yes    1053
Name: count, dtype: int64
```

In [28]:
```
plt.figure(figsize = (10, 5))
ax = data["Subscription Status"].value_counts().plot(kind = 'bar', rot = 0)
ax.set_xticklabels(('No', 'Yes'))

for p in ax.patches:
    ax.annotate(int(p.get_height()), (p.get_x() + 0.25, p.get_height() + 1), ha = 'cer
    ax.tick_params(axis = 'both', labelsize = 15)
plt.xlabel('Subscription Status', weight = "bold", color = "#D71313", fontsize = 14, l
plt.ylabel('Number of Occurrences', weight = "bold", color = "#D71313", fontsize = 14,
```



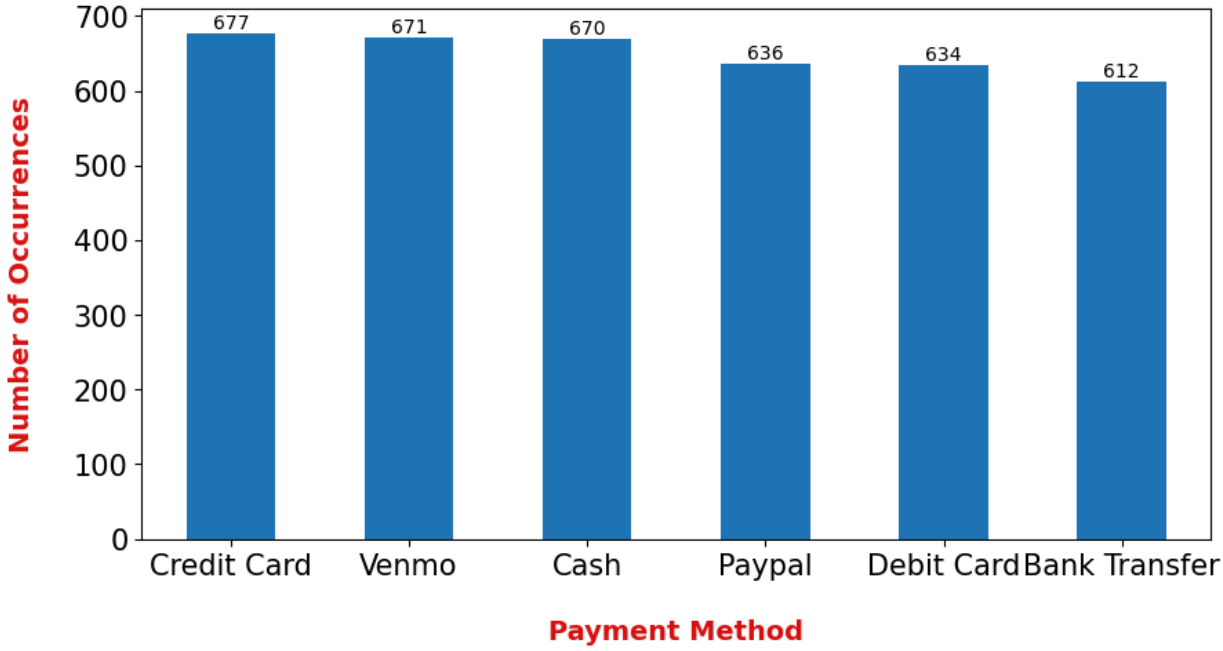
In [29]:
```
plt.figure(figsize = (10, 5))

counts = data["Subscription Status"].value_counts()
explode = (0, 0.1)

counts.plot(kind = 'pie', fontsize = 12, explode = explode, autopct = '%1.1f%%')
plt.xlabel('Size', weight = "bold", color = "#2F0F5D", fontsize = 14, labelpad = 20)
plt.axis('equal')
plt.legend(labels = counts.index, loc = "best")
plt.show()
```



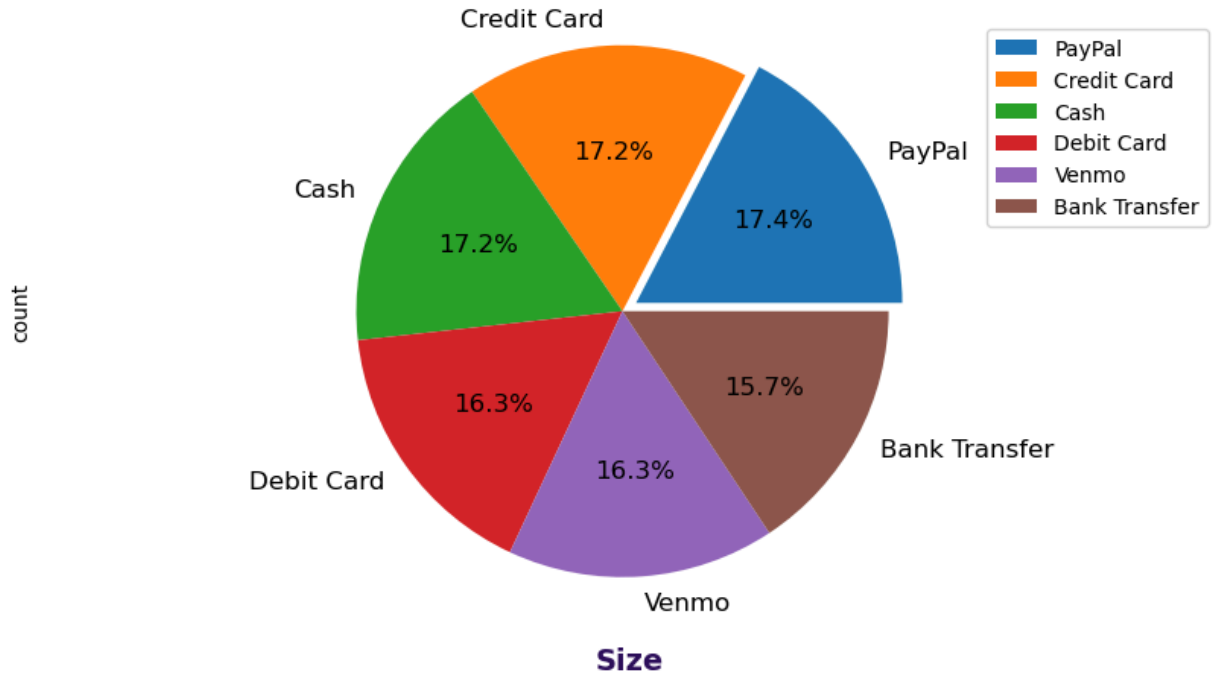
In [30]:
```
data["Payment Method"].value_counts()
```

Out[30]:
```
Payment Method
PayPal           677
Credit Card      671
Cash             670
Debit Card       636
Venmo            634
Bank Transfer    612
Name: count, dtype: int64
```

In [31]:
```
plt.figure(figsize = (10, 5))
ax = data["Payment Method"].value_counts().plot(kind = 'bar' , rot = 0)
ax.set_xticklabels(('Credit Card', 'Venmo', 'Cash', 'Paypal', 'Debit Card', 'Bank Trar

for p in ax.patches:
    ax.annotate(int(p.get_height()), (p.get_x() + 0.25, p.get_height() + 1), ha = 'cer
    ax.tick_params(axis = 'both', labelsize = 15)
plt.xlabel('Payment Method', weight = "bold", color = "#D71313", fontsize = 14, labelp
plt.ylabel('Number of Occurrences', weight = "bold", color = "#D71313", fontsize = 14,
```



In [32]:
```
plt.figure(figsize = (10, 5))

counts = data["Payment Method"].value_counts()
explode = (0.06, 0, 0, 0, 0.0, 0.0)

counts.plot(kind = 'pie', fontsize = 12, explode = explode, autopct = '%1.1f%%')
plt.xlabel('Size', weight = "bold", color = "#2F0F5D", fontsize = 14, labelpad = 20)
plt.axis('equal')
plt.legend(labels = counts.index, loc = "best")
plt.show()
```

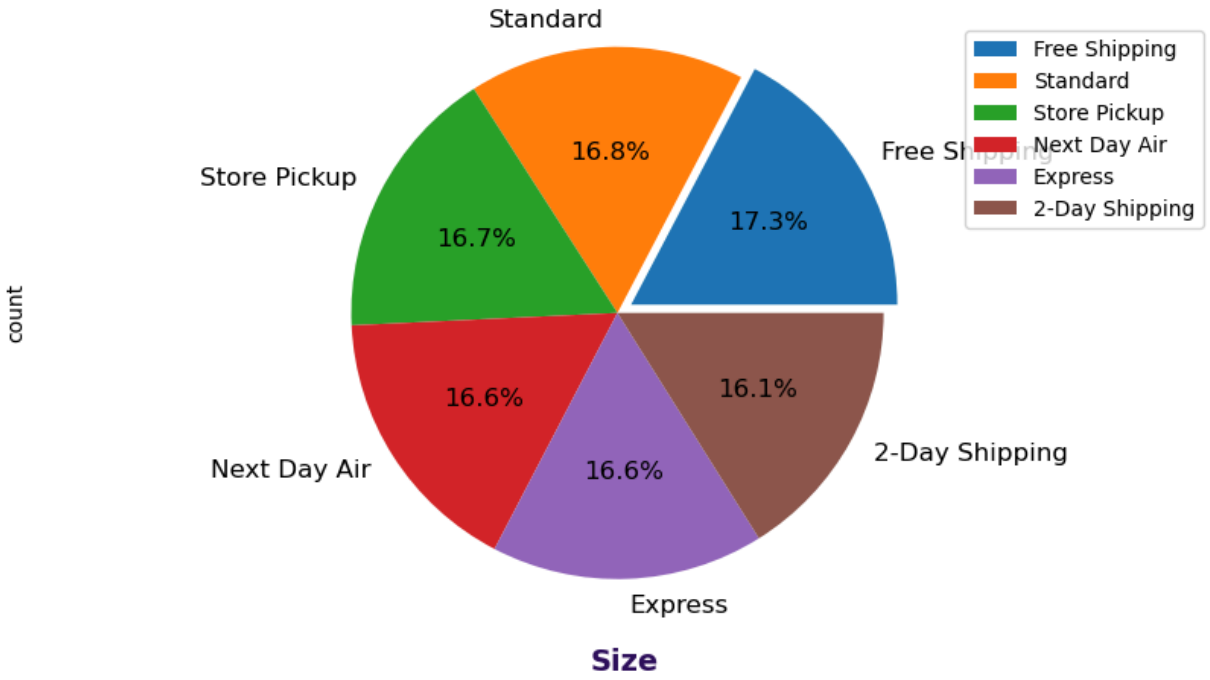


In [33]:
```
data["Shipping Type"].value_counts()
```

Out[33]:
```
Shipping Type
Free Shipping     675
Standard          654
Store Pickup      650
Next Day Air      648
Express           646
2-Day Shipping    627
Name: count, dtype: int64
```

In [34]:
```
plt.figure(figsize = (10, 5))
ax = data["Shipping Type"].value_counts().plot(kind = 'bar', color = 'Darkblue', rot =
ax.set_xticklabels(('Free Shipping', 'Standard', 'Store Pickup', 'Next Day Air', 'Expr

for p in ax.patches:
    ax.annotate(int(p.get_height()), (p.get_x() + 0.25, p.get_height() + 1), ha = 'cer
    ax.tick_params(axis = 'both', labelsize = 15)
plt.xlabel('Shipping Type', weight = "bold", color = "#D71313", fontsize = 14, labelpa
plt.ylabel('Number of Occurrences', weight = "bold", color = "#D71313", fontsize = 14,
```

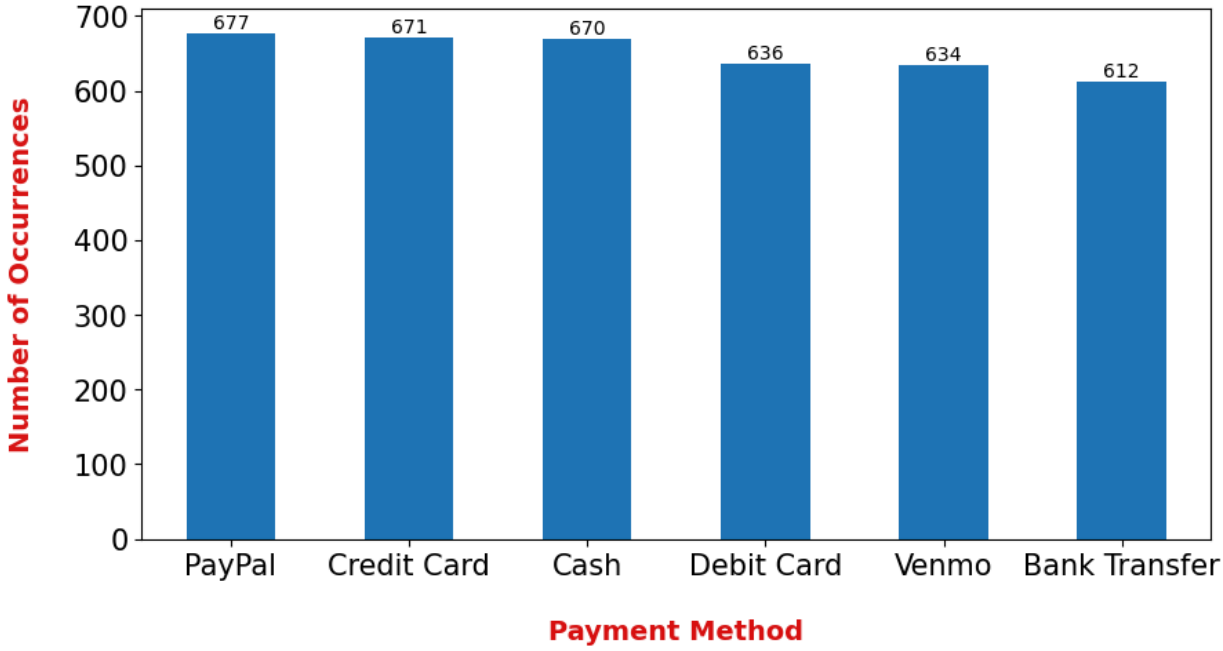
In [35]:
```
plt.figure(figsize = (10, 5))

counts = data["Shipping Type"].value_counts()
explode = (0.06, 0, 0, 0, 0.0, 0.0)

counts.plot(kind = 'pie', fontsize = 12, explode = explode, autopct = '%1.1f%%')
plt.xlabel('Size', weight = "bold", color = "#2F0F5D", fontsize = 14, labelpad = 20)
plt.axis('equal')
plt.legend(labels = counts.index, loc = "best")
plt.show()
```



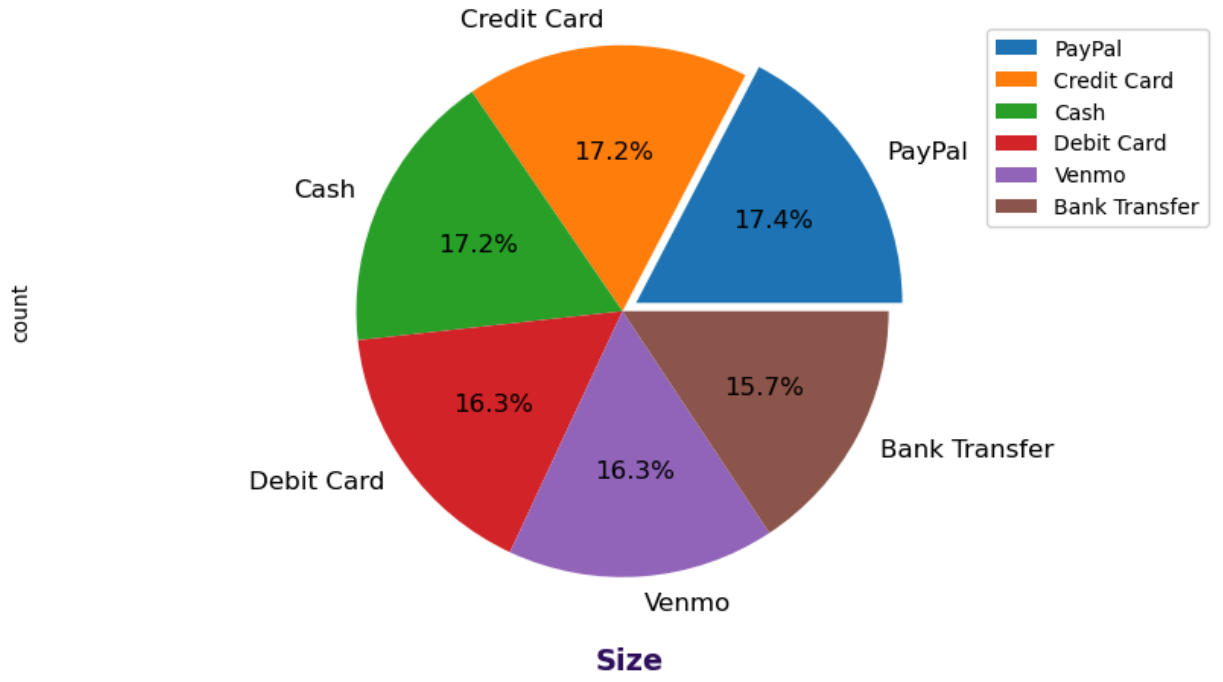
In [36]:
```
data["Payment Method"].value_counts()
```

Out[36]:
```
Payment Method
PayPal           677
Credit Card      671
Cash             670
Debit Card       636
Venmo            634
Bank Transfer    612
Name: count, dtype: int64
```

In [37]:
```
plt.figure(figsize = (10, 5))
ax = data["Payment Method"].value_counts().plot(kind = 'bar', rot = 0)
ax.set_xticklabels(('PayPal', 'Credit Card', 'Cash', 'Debit Card', 'Venmo', 'Bank Trar

for p in ax.patches:
    ax.annotate(int(p.get_height()), (p.get_x() + 0.25, p.get_height() + 1), ha = 'cer
    ax.tick_params(axis = 'both', labelsize = 15)
plt.xlabel('Payment Method', weight = "bold", color = "#D71313", fontsize = 14, labelp
plt.ylabel('Number of Occurrences', weight = "bold", color = "#D71313", fontsize = 14,
```



In [38]:
```
plt.figure(figsize = (10, 5))

counts = data["Payment Method"].value_counts()
explode = (0.06, 0, 0, 0, 0.0, 0.0)

counts.plot(kind = 'pie', fontsize = 12, explode = explode, autopct = '%1.1f%%')
plt.xlabel('Size', weight = "bold", color = "#2F0F5D", fontsize = 14, labelpad = 20)
plt.axis('equal')
plt.legend(labels = counts.index, loc = "best")
plt.show()
```



What is the average age of customers in the dataset ?

In [39]:
```
average_age = data['Age'].mean()
print("Average Age:", average_age)
```

```
Average Age: 44.06846153846154
```

What is the most common item purchased ?

In [40]:
```
most_common_item = data['Item Purchased'].mode()[0]
print("Most Common Item Purchased:", most_common_item)
```

```
Most Common Item Purchased: Blouse
```

What is the total purchase amount for each category ?

In [41]:
```
total_purchase_by_category = data.groupby('Category')['Purchase Amount (USD)'].sum()
print("Total Purchase Amount by Category:")
print(total_purchase_by_category)
```

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

What is the average review rating for male customers and female customers separately ?

In [42]:
```
average_rating_male = data[data['Gender'] == 'Male']['Review Rating'].mean()
average_rating_female = data[data['Gender'] == 'Female']['Review Rating'].mean()
print("Average Review Rating for Male Customers:", average_rating_male)
print("Average Review Rating for Female Customers:", average_rating_female)
```

```
Average Review Rating for Male Customers: 3.7539592760180995
Average Review Rating for Female Customers: 3.741426282051282
```

What is the most common payment method used by customers ?

In [43]:
```
most_common_payment_method = data['Payment Method'].mode()[0]
print("Most Common Payment Method:", most_common_payment_method)
```

```
Most Common Payment Method: PayPal
```

What is the median purchase amount (USD) ?

In [44]:
```
median_purchase_amount = data['Purchase Amount (USD)'].median()
print("Median Purchase Amount (USD):", median_purchase_amount)
```

```
Median Purchase Amount (USD): 60.0
```

How many customers have opted for the Subscription and How many did not subscribed ?

In [45]:
```
subscription_count = data[data['Subscription Status'] == 'Yes']['Customer ID'].count()
not_subscribed_count = data[data['Subscription Status'] == 'No']['Customer ID'].count(
print("Number of Customers with Subscription: ", subscription_count)
print("Number of Customers Not subscribed : ", not_subscribed_count)
```

```
Number of Customers with Subscription:  1053
Number of Customers Not subscribed :  2847
```

What is the average purchase amount for customers with a subscription status of 'Yes' and 'No' ?

In [46]:
```
avg_purchase_subscription_yes = data[data['Subscription Status'] == 'Yes']['Purchase A
avg_purchase_subscription_no = data[data['Subscription Status'] == 'No']['Purchase Amo
print("Average Purchase Amount for Subscription 'Yes':", avg_purchase_subscription_yes
print("Average Purchase Amount for Subscription 'No':", avg_purchase_subscription_no)
```

```
Average Purchase Amount for Subscription 'Yes': 59.49192782526116
Average Purchase Amount for Subscription 'No': 59.865121180189675
```

What is the most common season for purchases ?

In [47]:
```
most_common_season = data['Season'].mode()[0]
print("Most Common Season for Purchases:", most_common_season)
```

```
Most Common Season for Purchases: Spring
```

What is the total purchase amount for each gender ?

In [48]:
```
total_purchase_by_gender = data.groupby('Gender')['Purchase Amount (USD)'].sum()
print("Total Purchase Amount by Gender:")
print(total_purchase_by_gender)
```

```
Total Purchase Amount by Gender:
Gender
Female     75191
Male      157890
Name: Purchase Amount (USD), dtype: int64
```

What is the average age of customers who made purchases in the Summer season ?

In [49]:
```
average_age_summer = data[data['Season'] == 'Summer']['Age'].mean()
print("Average Age of Customers in the Summer Season:", average_age_summer)
```

```
Average Age of Customers in the Summer Season: 43.973821989528794
```

How many customers used a promo code for their purchase ?

In [50]:
```
promo_code_count = data[data['Promo Code Used'] == 'Yes']['Customer ID'].count()
print("Number of Customers who used Promo Code:", promo_code_count)
```

```
Number of Customers who used Promo Code: 1677
```

What is the maximum and minimum review rating in the dataset ?

In [51]:
```
max_review_rating = data['Review Rating'].max()
min_review_rating = data['Review Rating'].min()
print("Maximum Review Rating:", max_review_rating)
print("Minimum Review Rating:", min_review_rating)
```

```
Maximum Review Rating: 5.0
Minimum Review Rating: 2.5
```

What is the most common shipping type for customers with a review rating above 4 ?

In [52]:
```
common_shipping_high_rating = data[data['Review Rating'] > 4]['Shipping Type'].mode()[
print("Most Common Shipping Type for High Review Ratings:", common_shipping_high_ratir
```

```
Most Common Shipping Type for High Review Ratings: Standard
```

How many customers have made more than 30 previous purchases ?

In [53]:
```
customers_above_30_previous_purchases = data[data['Previous Purchases'] > 30]['Custome
print("Number of Customers with more than 30 Previous Purchases:", customers_above_30_
```

```
Number of Customers with more than 30 Previous Purchases: 1549
```

What is the average purchase amount for customers who have made more than 30 previous purchases ?

In [54]:
```
avg_purchase_above_30_previous_purchases = data[data['Previous Purchases'] > 30]['Purc
print("Average Purchase Amount for Customers with more than 30 Previous Purchases:", a
```

```
Average Purchase Amount for Customers with more than 30 Previous Purchases: 60.028405
42285345
```

What is the most common payment method for customers who shop in the Winter season ?

In [78]:
```
total_purchase_free_shipping = data[data['Shipping Type'] == 'Free Shipping']['Purchas
print("Total Purchase Amount for 'Free Shipping' Shipping Type:", total_purchase_free_
```

```
Total Purchase Amount for 'Free Shipping' Shipping Type: 40777
```

What is the total purchase amount for customers with a 'Free Shipping' shipping type ?

In [56]:
```
total_purchase_free_shipping = data[data['Shipping Type'] == 'Free Shipping']['Purchas
print("Total Purchase Amount for 'Free Shipping' Shipping Type:", total_purchase_free_
```

```
Total Purchase Amount for 'Free Shipping' Shipping Type: 40777
```

What is the average purchase amount for customers who used a discount ?

In [57]:
```
avg_purchase_with_discount = data[data['Discount Applied'] == 'Yes']['Purchase Amount
print("Average Purchase Amount for Customers with Discount Applied:", avg_purchase_wit
```

```
Average Purchase Amount for Customers with Discount Applied: 59.27906976744186
```

What is the most common category of items purchased by female customers with a review rating below 3 ?

In [58]:
```
common_category_low_rating_female = data[(data['Gender'] == 'Female') & (data['Review
print("Most Common Category for Low Review Rating Female Customers:", common_category_
```

```
Most Common Category for Low Review Rating Female Customers: Clothing
```

What is the average age of customers who made purchases with a review rating above 4 and used a promo code?

In [59]:
```
average_age_high_rating_promo = data[(data['Review Rating'] > 4) & (data['Promo Code U
print("Average Age of Customers with High Review Ratings and Promo Code Used:", averag
```

```
Average Age of Customers with High Review Ratings and Promo Code Used: 43.9872
```

What is the total purchase amount for customers in each location ?

In [60]:
```
total_purchase_by_location = data.groupby('Location')['Purchase Amount (USD)'].sum()
print("Total Purchase Amount by Location:")
print(total_purchase_by_location)
```

```
Total Purchase Amount by Location:
Location
Alabama           5261
Alaska            4867
Arizona           4326
Arkansas          4828
California        5605
Colorado          4222
Connecticut       4226
Delaware          4758
Florida           3798
Georgia           4645
Hawaii            3752
Idaho             5587
Illinois          5617
Indiana           4655
Iowa              4201
Kansas            3437
Kentucky          4402
Louisiana         4848
Maine             4388
Maryland          4795
Massachusetts     4384
Michigan          4533
Minnesota         4977
Mississippi       4883
Missouri          4691
Montana           5784
Nebraska          5172
Nevada            5514
New Hampshire     4219
New Jersey        3802
New Mexico        5014
New York          5257
North Carolina    4742
North Dakota      5220
Ohio              4649
Oklahoma          4376
Oregon            4243
Pennsylvania      4926
Rhode Island      3871
South Carolina    4439
South Dakota      4236
Tennessee         4772
Texas             4712
Utah              4443
Vermont           4860
Virginia          4842
Washington        4623
West Virginia     5174
Wisconsin         4196
Wyoming           4309
Name: Purchase Amount (USD), dtype: int64
```

What is the average purchase amount for customers who have a subscription and used Venmo as the payment method ?

In [61]:
```
avg_purchase_subscription_venmo = data[(data['Subscription Status'] == 'Yes') & (data[
print("Average Purchase Amount for Customers with Subscription and Venmo Payment Methc
```

```
Average Purchase Amount for Customers with Subscription and Venmo Payment Method: 57.
51149425287356
```

What is the frequency distribution of the 'Frequency of Purchases' column ?

In [62]:
```
purchase_frequency_distribution = data['Frequency of Purchases'].value_counts()
print("Frequency Distribution of Purchase Frequency:")
print(purchase_frequency_distribution)
```

```
Frequency Distribution of Purchase Frequency:
Frequency of Purchases
Every 3 Months    584
Annually          572
Quarterly         563
Monthly           553
Bi-Weekly         547
Fortnightly       542
Weekly            539
Name: count, dtype: int64
```

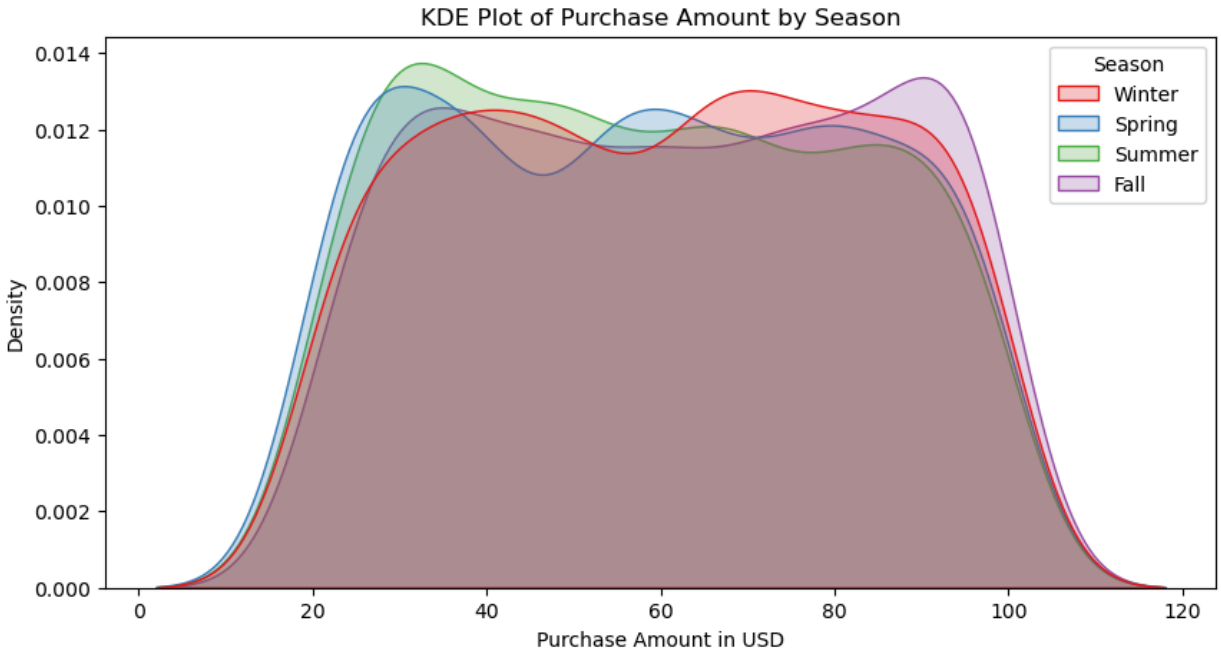
What is the average purchase amount for each color of items ?

In [63]:
```
avg_purchase_by_color = data.groupby('Color')['Purchase Amount (USD)'].mean()
print("Average Purchase Amount by Color:")
print(avg_purchase_by_color)
```

```
Average Purchase Amount by Color:
Color
Beige        60.414966
Black        58.401198
Blue         56.953947
Brown        59.063830
Charcoal     60.633987
Cyan         61.891566
Gold         61.007246
Gray         62.490566
Green        65.704142
Indigo       56.251701
Lavender     59.129252
Magenta      57.131579
Maroon       59.525316
Olive        58.146893
Orange       60.889610
Peach        59.187919
Pink         60.588235
Purple       60.013245
Red          59.317568
Silver       56.832370
Teal         60.808140
Turquoise    55.613793
Violet       61.716867
White        62.640845
Yellow       59.241379
Name: Purchase Amount (USD), dtype: float64
```

KDE plot for Purchase Amount by Season

In [64]:
```
plt.figure(figsize=(10, 5))
sns.kdeplot(data , x = 'Purchase Amount (USD)', hue = 'Season', common_norm = False, f
plt.title('KDE Plot of Purchase Amount by Season')
plt.xlabel('Purchase Amount in USD')
plt.show()
```



What is the most common payment method for customers who purchased items in the Fall season ?

In [65]:
```
common_payment_fall = data[data['Season'] == 'Fall']['Payment Method'].mode()[0]
print("Most Common Payment Method for Fall Season Purchases:", common_payment_fall)
```

```
Most Common Payment Method for Fall Season Purchases: Cash
```

How many customers have made a purchase in each category ?

In [66]:
```
purchase_count_by_category = data['Category'].value_counts()
print("Purchase Count by Category:")
print(purchase_count_by_category)
```

```
Purchase Count by Category:
Category
Clothing       1737
Accessories    1240
Footwear        599
Outerwear       324
Name: count, dtype: int64
```

What is the total purchase amount for each size of clothing items (XL, L, M, S) ?

In [67]:
```
total_purchase_by_size = data[data['Category'] == 'Clothing'].groupby('Size')['Purchas
print("Total Purchase Amount by Size for Clothing Items:")
```

```
print(total_purchase_by_size)
```

```
Total Purchase Amount by Size for Clothing Items:
Size
L     27864
M     47041
S     17416
XL    11943
Name: Purchase Amount (USD), dtype: int64
```

What is the most common item purchased by customers in Louisiana with a review rating of 4 or higher ?

In [68]:
```
common_item_high_rating_louisiana = data[(data['Location'] == 'Louisiana') & (data['Re
print("Most Common Item Purchased by High-Rating Customers in Louisiana:", common_item
```

```
Most Common Item Purchased by High-Rating Customers in Louisiana: Sweater
```

What is the most common category of items purchased by male customers in the Winter season with a review rating below 3 ?

In [69]:
```
common_category_low_rating_male_winter = data[(data['Gender'] == 'Male') & (data['Seas
print("Most Common Category for Low-Rating Male Customers in Winter Season:", common_c
```

```
Most Common Category for Low-Rating Male Customers in Winter Season: Clothing
```

How many customers have a subscription status of 'Yes' and used a promo code for their purchase ?

In [70]:
```
subscription_promo_count = data[(data['Subscription Status'] == 'Yes') & (data['Promo
print("Number of Customers with Subscription and Promo Code Used: ", subscription_prom
```

```
Number of Customers with Subscription and Promo Code Used:  1053
```

Histogram of Age Distribution

In [71]:
```
plt.figure(figsize = (10, ))
plt.hist(data['Age'], bins = 20, edgecolor = 'k')
plt.title('Age Distribution')
plt.xlabel('Age')
plt.ylabel('Frequency')
plt.show()
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
Cell In[71], line 1
----> 1 plt.figure(figsize = (10, ))
      2 plt.hist(data['Age'], bins = 20, edgecolor = 'k')
      3 plt.title('Age Distribution')

File ~\anaconda3\Lib\site-packages\matplotlib\_api\deprecation.py:454, in make_keyword_only.<locals>.wrapper(*args, **kwargs)
    448 if len(args) > name_idx:
    449     warn_deprecated(
    450         since, message="Passing the %(name)s %(obj_type)s "
    451         "positionally is deprecated since Matplotlib %(since)s; the "
    452         "parameter will become keyword-only %(removal)s.",
    453         name=name, obj_type=f"parameter of {func.__name__}()")
--> 454 return func(*args, **kwargs)

File ~\anaconda3\Lib\site-packages\matplotlib\pyplot.py:840, in figure(num, figsize, dpi, facecolor, edgecolor, frameon, FigureClass, clear, **kwargs)
    830 if len(allnums) == max_open_warning >= 1:
    831     _api.warn_external(
    832         f"More than {max_open_warning} figures have been opened. "
    833         f"Figures created through the pyplot interface "
   (...)
    837         f"Consider using `matplotlib.pyplot.close()`.",
    838         RuntimeWarning)
--> 840 manager = new_figure_manager(
    841     num, figsize=figsize, dpi=dpi,
    842     facecolor=facecolor, edgecolor=edgecolor, frameon=frameon,
    843     FigureClass=FigureClass, **kwargs)
    844 fig = manager.canvas.figure
    845 if fig_label:

File ~\anaconda3\Lib\site-packages\matplotlib\pyplot.py:384, in new_figure_manager(*args, **kwargs)
    382 """Create a new figure manager instance."""
    383 _warn_if_gui_out_of_main_thread()
--> 384 return _get_backend_mod().new_figure_manager(*args, **kwargs)

File ~\anaconda3\Lib\site-packages\matplotlib_inline\backend_inline.py:27, in new_figure_manager(num, FigureClass, *args, **kwargs)
     21 def new_figure_manager(num, *args, FigureClass=Figure, **kwargs):
     22     """
     23     Return a new figure manager for a new figure instance.
     24
     25     This function is part of the API expected by Matplotlib backends.
     26     """
---> 27     return new_figure_manager_given_figure(num, FigureClass(*args, **kwargs))

File ~\anaconda3\Lib\site-packages\matplotlib\_api\deprecation.py:454, in make_keyword_only.<locals>.wrapper(*args, **kwargs)
    448 if len(args) > name_idx:
    449     warn_deprecated(
    450         since, message="Passing the %(name)s %(obj_type)s "
    451         "positionally is deprecated since Matplotlib %(since)s; the "
    452         "parameter will become keyword-only %(removal)s.",
    453         name=name, obj_type=f"parameter of {func.__name__}()")
--> 454 return func(*args, **kwargs)

File ~\anaconda3\Lib\site-packages\matplotlib\figure.py:2571, in Figure.__init__(sel
```

```
f, figsize, dpi, facecolor, edgecolor, linewidth, frameon, subplotpars, tight_layout,
constrained_layout, layout, **kwargs)
   2568 if not np.isfinite(figsize).all() or (np.array(figsize) < 0).any():
   2569     raise ValueError('figure size must be positive finite not '
   2570                      f'{figsize}')
-> 2571 self.bbox_inches = Bbox.from_bounds(0, 0, *figsize)
   2573 self.dpi_scale_trans = Affine2D().scale(dpi)
   2574 # do not use property as it will trigger

TypeError: Bbox.from_bounds() missing 1 required positional argument: 'height'
```

Box Plot of Purchase Amount by Gender

In [ ]:
```
plt.figure(figsize = (10, 5))
sns.boxplot(data, x='Gender', y='Purchase Amount (USD)')
plt.title('Purchase Amount by Gender')
plt.xlabel('Gender')
plt.ylabel('Purchase Amount (USD)')
plt.show()
```

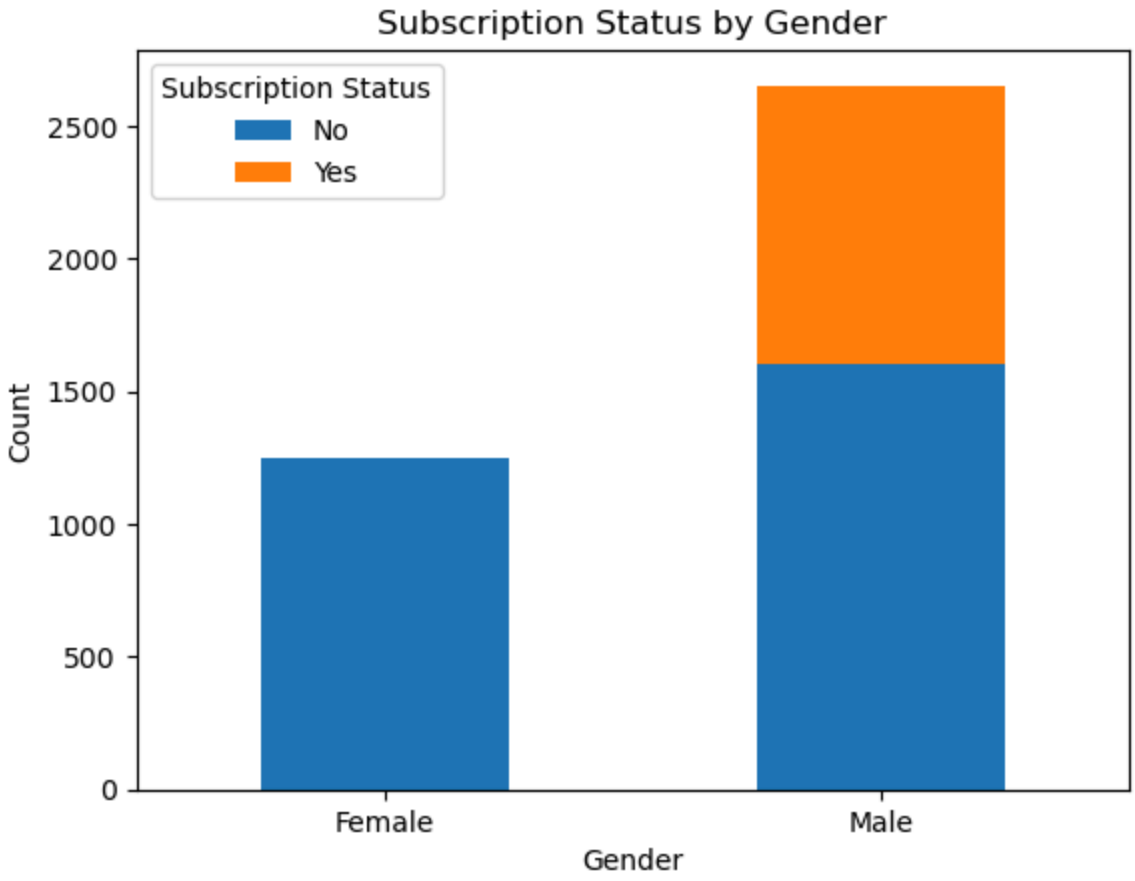
Scatter Plot of Age vs Review rating

In [ ]:
```
plt.figure(figsize = (10, 5))
plt.scatter(data['Age'], data['Review Rating'], alpha=0.5)
plt.title('Age vs. Review Rating')
plt.xlabel('Age')
plt.ylabel('Review Rating')
plt.show()
```

Stacked Bar Chart of Subscription Status by Gender

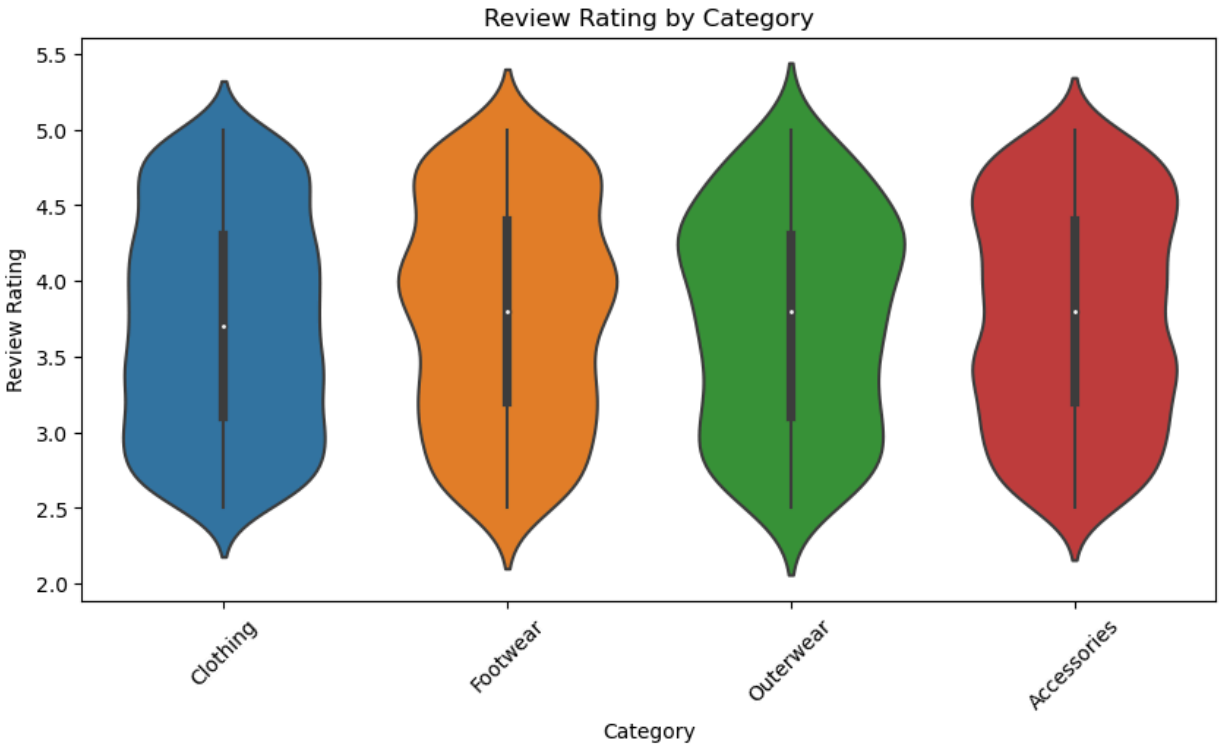
In [72]:
```
subscription_gender_counts = data.groupby(['Gender', 'Subscription Status']).size().ur
plt.figure(figsize = (10, 5))
subscription_gender_counts.plot(kind='bar', stacked=True, rot = 0);
plt.title('Subscription Status by Gender')
plt.xlabel('Gender')
plt.ylabel('Count')
plt.show();
```

```
<Figure size 1000x500 with 0 Axes>
```



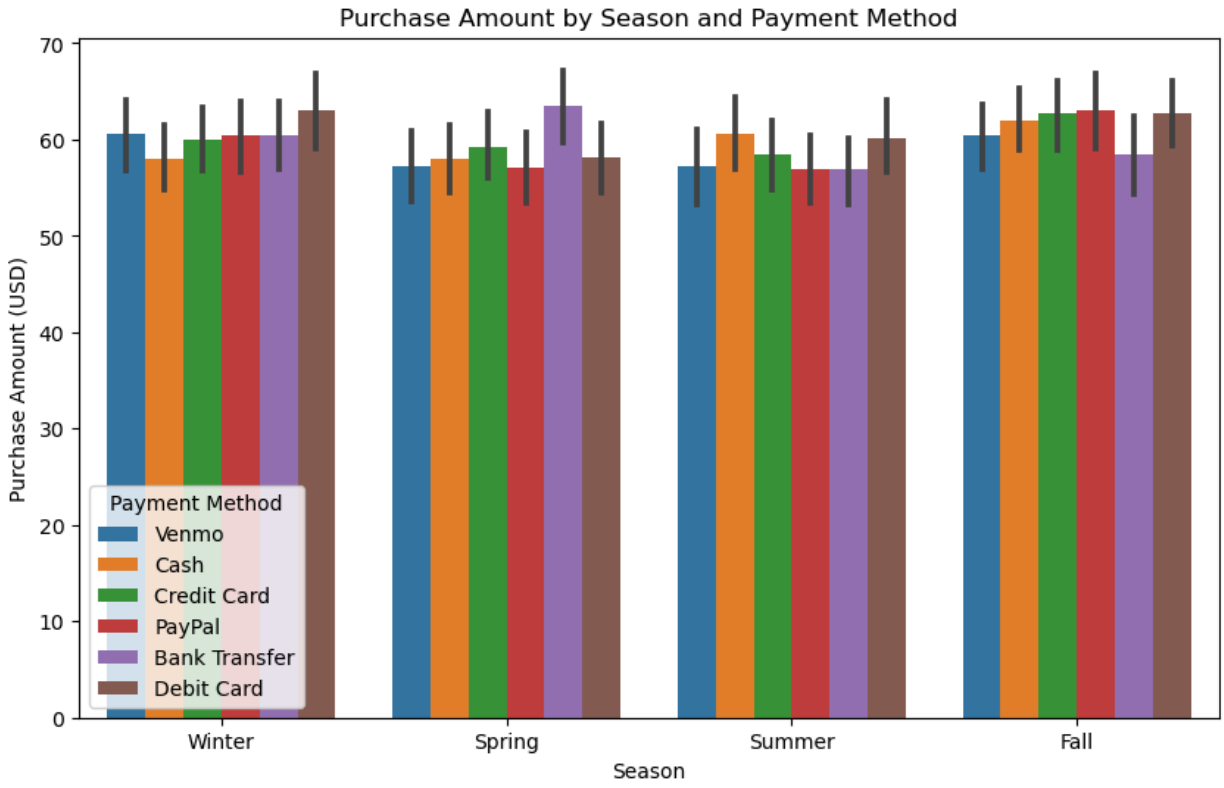
Violin Plot of review rating by Category

In [73]:
```
plt.figure(figsize=(10, 5))
sns.violinplot(data, x='Category', y='Review Rating')
plt.title('Review Rating by Category')
plt.xlabel('Category')
plt.ylabel('Review Rating')
plt.xticks(rotation=45)
plt.show()
```



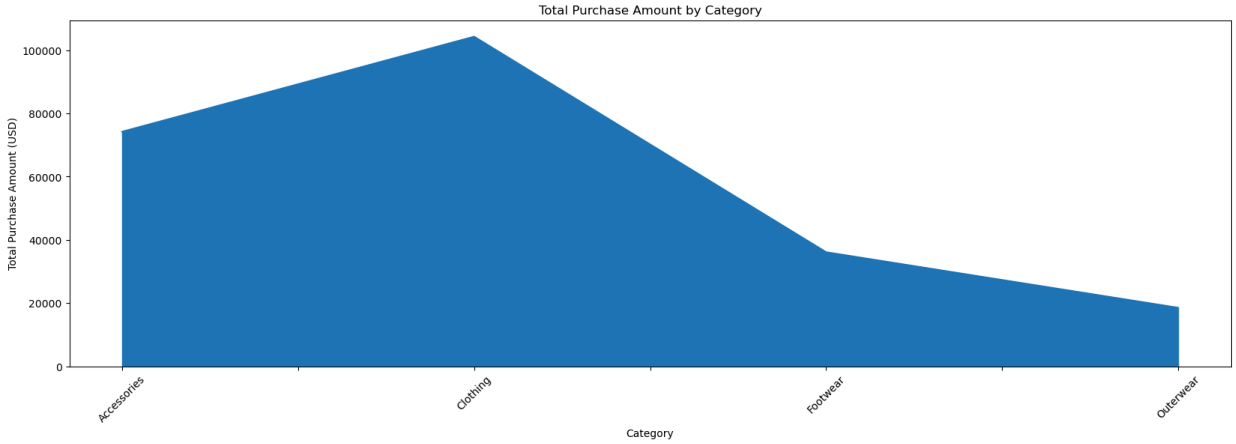
Purchase Amount by Season and Pyament Method

In [74]:
```
plt.figure(figsize=(10, 6))
sns.barplot(data, x='Season', y='Purchase Amount (USD)', hue='Payment Method')
plt.title('Purchase Amount by Season and Payment Method')
plt.xlabel('Season')
plt.ylabel('Purchase Amount (USD)')
plt.xticks(rotation = 0)
plt.show()
```



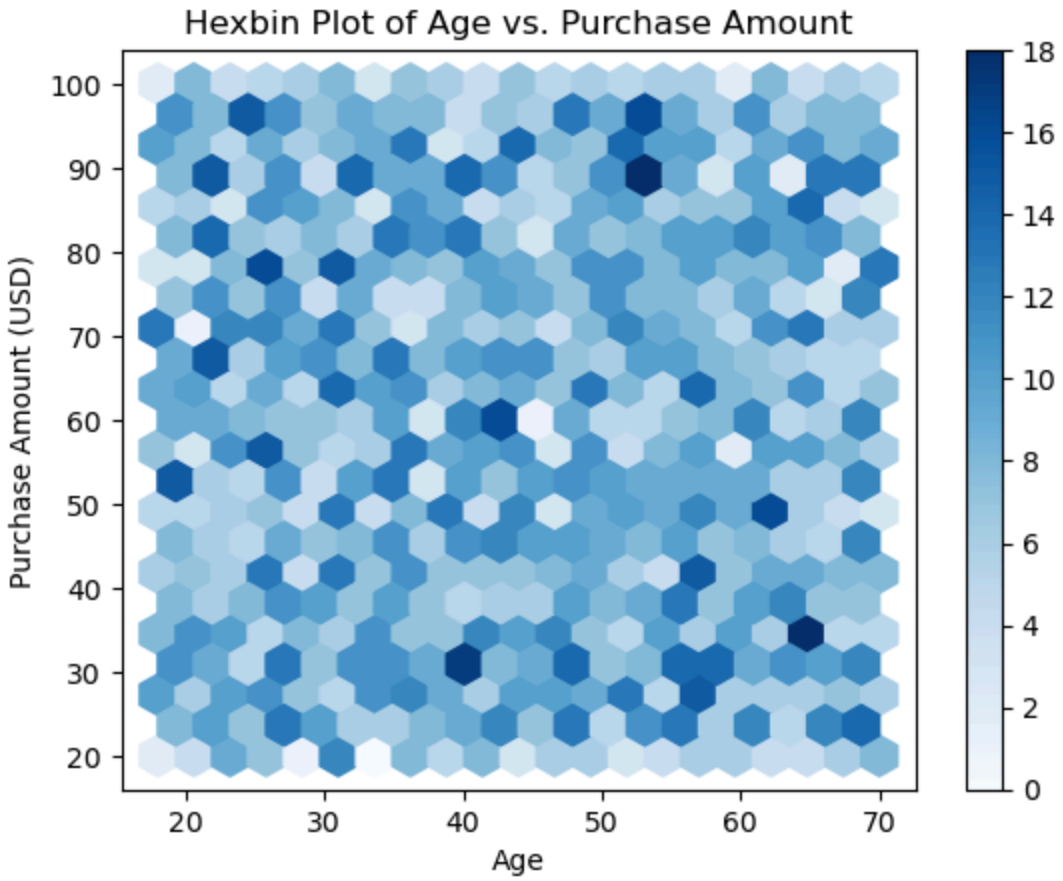
Area Plot of Total Purchase Amount by Category ?

In [75]:
```
plt.figure(figsize=(20, 6))
category_purchase_total = data.groupby('Category')['Purchase Amount (USD)'].sum()

category_purchase_total.plot(kind='area')
plt.title('Total Purchase Amount by Category')
plt.xlabel('Category')
plt.ylabel('Total Purchase Amount (USD)')
plt.xticks(rotation=45)
plt.show()
```



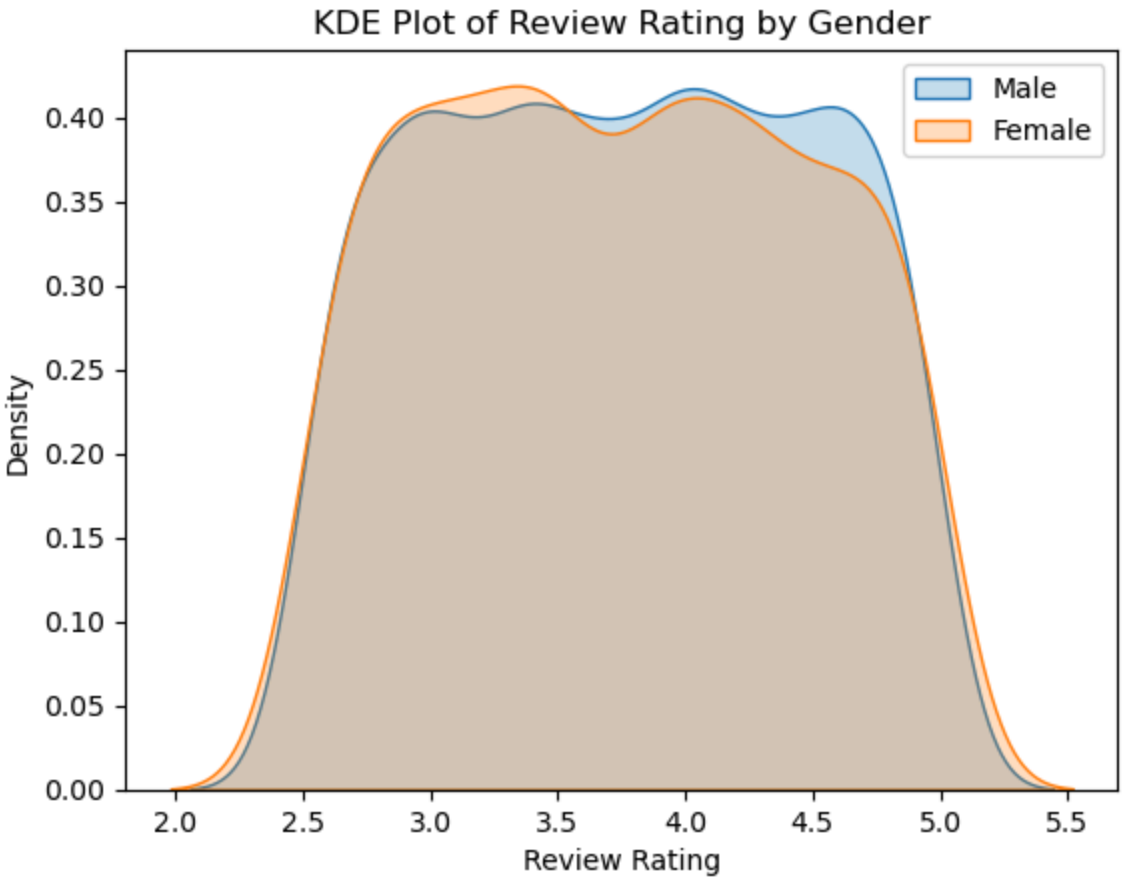
Hexbin Plot of Age vs Purchase Amount?

In [76]:
```
plt.hexbin(data['Age'], data['Purchase Amount (USD)'], gridsize=20, cmap='Blues')
plt.title('Hexbin Plot of Age vs. Purchase Amount')
plt.xlabel('Age')
plt.ylabel('Purchase Amount (USD)')
plt.colorbar()
plt.show()
```



KDE Plot of Review Rating by Gender

In [77]:
```
sns.kdeplot(data[data['Gender'] == 'Male']['Review Rating'], label='Male', shade=True)
sns.kdeplot(data[data['Gender'] == 'Female']['Review Rating'], label='Female', shade=T
plt.title('KDE Plot of Review Rating by Gender')
plt.xlabel('Review Rating')
plt.ylabel('Density')
plt.legend()
plt.show()
```



Thank You For Reviewing the project !! Please share feedback