In [41]:
```
#Importing Modules
import numpy as np # linear algebra
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns # data processing
```

In [4]:
```
import os
for dirname, _, filenames in os.walk('retail_sales_dataset.csv'):
    for filename in filenames:
        print(os.path.join(dirname, filename))
```

In [5]:
```
sales_df=pd.read_csv("retail_sales_dataset.csv")
```

In [7]:
```
#Exploratory Data Analysis (EDA)
# Display the first few rows of the dataset
sales_df.head()
```

Out[7]:

	Transaction ID	Date	Customer ID	Gender	Age	Product Category	Quantity	Price per Unit	Total Amount
0	1	2023-11-24	CUST001	Male	34	Beauty	3	50	150
1	2	2023-02-27	CUST002	Female	26	Clothing	2	500	1000
2	3	2023-01-13	CUST003	Male	50	Electronics	1	30	30
3	4	2023-05-21	CUST004	Male	37	Clothing	1	500	500
4	5	2023-05-06	CUST005	Male	30	Beauty	2	50	100

In [8]:
```
# Display the first few rows of the dataset
sales_df.tail()
```

Out[8]:

	Transaction ID	Date	Customer ID	Gender	Age	Product Category	Quantity	Price per Unit	Total Amount
995	996	2023-05-16	CUST996	Male	62	Clothing	1	50	50
996	997	2023-11-17	CUST997	Male	52	Beauty	3	30	90
997	998	2023-10-29	CUST998	Female	23	Beauty	4	25	100
998	999	2023-12-05	CUST999	Female	36	Electronics	3	50	150
999	1000	2023-04-12	CUST1000	Male	47	Electronics	4	30	120

In [9]:
```
sales_df.info()
```

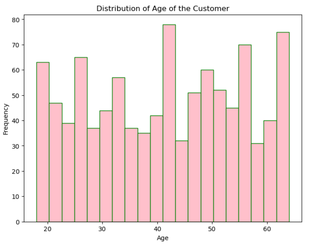
```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1000 entries, 0 to 999
Data columns (total 9 columns):
 #   Column             Non-Null Count  Dtype 
---  ------             --------------  ----- 
 0   Transaction ID     1000 non-null   int64 
 1   Date               1000 non-null   object
 2   Customer ID        1000 non-null   object
 3   Gender             1000 non-null   object
 4   Age                1000 non-null   int64 
 5   Product Category   1000 non-null   object
 6   Quantity           1000 non-null   int64 
 7   Price per Unit     1000 non-null   int64 
 8   Total Amount       1000 non-null   int64 
dtypes: int64(5), object(4)
memory usage: 70.4+ KB
```

In [10]:
```
# Summary statistics of desc analysis
sales_df.describe()
```

Out[10]:

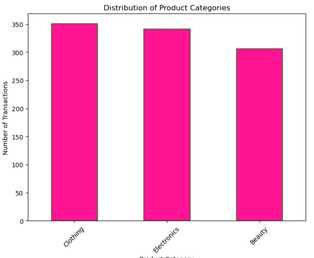
	Transaction ID	Age	Quantity	Price per Unit	Total Amount
count	1000.000000	1000.000000	1000.000000	1000.000000	1000.000000
mean	500.500000	41.392000	2.514000	179.890000	456.000000
std	288.819436	13.68143	1.132734	189.681356	559.997632
min	1.000000	18.000000	1.000000	25.000000	25.000000
25%	250.750000	29.000000	1.000000	30.000000	60.000000
50%	500.500000	42.000000	3.000000	50.000000	135.000000
75%	750.250000	53.000000	4.000000	300.000000	900.000000
max	1000.000000	64.000000	4.000000	500.000000	2000.000000

In [16]:
```
#Data Visualization
# Distribution of customer age
plt.figure(figsize=(8, 6))
plt.hist(sales_df['Age'], bins=20, color='pink', edgecolor='green')
plt.xlabel('Age')
plt.ylabel('Frequency')
plt.title('Distribution of Age of the Customer')
plt.show()
```



In [28]:
```
#Time Series Analysis
#Sales over time
plt.figure(figsize=(10, 5))
sales_df['Date'] = pd.to_datetime(sales_df['Date'])
monthly_sales = sales_df.groupby(sales_df['Date'].dt.to_period('M'))['Total Amount'].sum()
monthly_sales.plot(kind='line', marker='*')
plt.xlabel('Date')
plt.ylabel('Total Sales Amount')
plt.title('Monthly Sales Trend')
plt.xticks(rotation=45)
plt.show()
```

In [31]:
```
# Product category distribution
plt.figure(figsize=(8, 6))
product_counts = sales_df['Product Category'].value_counts()
product_counts.plot(kind='bar', color='deeppink', edgecolor='green')
plt.xlabel('Product Category')
plt.ylabel('Number of Transactions')
plt.title('Distribution of Product Categories')
plt.xticks(rotation=45)
plt.show()
```



In [34]:
```
# Correlation Analysis using Scatterplot in between age and total spending
plt.figure(figsize=(8, 6))
plt.scatter(sales_df['Age'], sales_df['Total Amount'], alpha=0.5)
plt.xlabel('Age')
plt.ylabel('Total Amount')
plt.title('Correlation between Age and Total Spending')
plt.show()
```

In [42]:
```
# Data Cleaning
sales_df=sales_df.drop(columns=['Date','Customer ID','Gender','Product Category'])
print('Modified DataFrame:\n',sales_df)
```

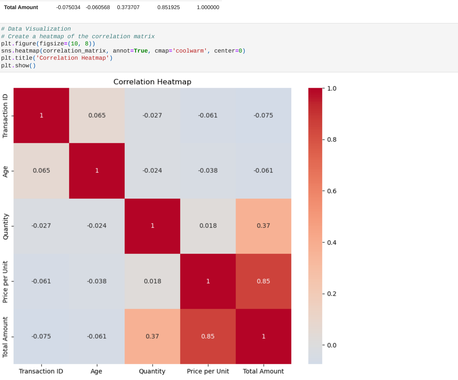
```
Modified DataFrame:
      Transaction ID  Age  Quantity  Price per Unit  Total Amount
0                1   34         3              50           150
1                2   26         2             500          1000
2                3   50         1              30            30
3                4   37         1             500           500
4                5   30         2              50           100
..             ...  ...       ...             ...           ...
995            996   62         1              50            50
996            997   52         3              30            90
997            998   23         4              25           100
998            999   36         3              50           150
999           1000   47         4              30           120

[1000 rows x 5 columns]
```

In [43]:
```
# Calculate correlation matrix
correlation_matrix = sales_df.corr()
sales_df.corr()
```

Out[43]:

	Transaction ID	Age	Quantity	Price per Unit	Total Amount
Transaction ID	1.000000	0.065191	-0.026623	-0.060837	-0.075034
Age	0.065191	1.000000	-0.023737	-0.038423	-0.060568
Quantity	-0.026623	-0.023737	1.000000	0.017501	0.373707
Price per Unit	-0.060837	-0.038423	0.017501	1.000000	0.851925
Total Amount	-0.075034	-0.060568	0.373707	0.851925	1.000000

In [44]:
```
# Data Visualization
# Create a heatmap of the correlation matrix
plt.figure(figsize=(10, 8))
sns.heatmap(correlation_matrix, annot=True, cmap='coolwarm', center=0)
plt.title('Correlation Heatmap')
plt.show()
```



In [ ]:
```
#Process to read Heatmap
#Look at the color of each cell to see the strength and direction of the correlation.
#Darker colors indicate stronger correlations, while lighter colors indicate weaker correlations.
#Positive correlations (when one variable increases, the other variable tends to increase) are usually represented by warm colors, such as red or orange.
#Negative correlations (when one variable increases, the other variable tends to decrease) are usually represented by cool colors, such as blue or green.

#Here, highest correlation between 'Total Amount' and 'Price per Unit' is present with 0.85 value, by following the next correlation between 'Total Amount' and
```