In [2]:
```
!pip install pandas matplotlib seaborn numpy
```

```
Requirement already satisfied: pandas in c:\users\pc-udaya\appdata\local\programs\python\python312\lib\sit
e-packages (2.2.2)
Requirement already satisfied: matplotlib in c:\users\pc-udaya\appdata\local\programs\python\python312\lib
\site-packages (3.9.1)
Requirement already satisfied: seaborn in c:\users\pc-udaya\appdata\local\programs\python\python312\lib\si
te-packages (0.13.2)
Requirement already satisfied: numpy in c:\users\pc-udaya\appdata\local\programs\python\python312\lib\site
-packages (2.0.0)
Requirement already satisfied: python-dateutil>=2.8.2 in c:\users\pc-udaya\appdata\local\programs\python\p
ython312\lib\site-packages (from pandas) (2.9.0.post0)
Requirement already satisfied: pytz>=2020.1 in c:\users\pc-udaya\appdata\local\programs\python\python312\l
ib\site-packages (from pandas) (2024.1)
Requirement already satisfied: tzdata>=2022.7 in c:\users\pc-udaya\appdata\local\programs\python\python312
\lib\site-packages (from pandas) (2024.1)
Requirement already satisfied: contourpy>=1.0.1 in c:\users\pc-udaya\appdata\local\programs\python\python3
12\lib\site-packages (from matplotlib) (1.2.1)
Requirement already satisfied: cycler>=0.10 in c:\users\pc-udaya\appdata\local\programs\python\python312\l
ib\site-packages (from matplotlib) (0.12.1)
Requirement already satisfied: fonttools>=4.22.0 in c:\users\pc-udaya\appdata\local\programs\python\python
312\lib\site-packages (from matplotlib) (4.53.1)
Requirement already satisfied: kiwisolver>=1.3.1 in c:\users\pc-udaya\appdata\local\programs\python\python
312\lib\site-packages (from matplotlib) (1.4.5)
Requirement already satisfied: packaging>=20.0 in c:\users\pc-udaya\appdata\local\programs\python\python31
2\lib\site-packages (from matplotlib) (24.1)
Requirement already satisfied: pillow>=8 in c:\users\pc-udaya\appdata\local\programs\python\python312\lib
\site-packages (from matplotlib) (10.4.0)
Requirement already satisfied: pyparsing>=2.3.1 in c:\users\pc-udaya\appdata\local\programs\python\python3
12\lib\site-packages (from matplotlib) (3.1.2)
Requirement already satisfied: six>=1.5 in c:\users\pc-udaya\appdata\local\programs\python\python312\lib\s
ite-packages (from python-dateutil>=2.8.2->pandas) (1.16.0)
```

```
[notice] A new release of pip is available: 23.2.1 -> 24.2
[notice] To update, run: C:\Users\PC-Udaya\AppData\Local\Programs\Python\Python312\python.exe -m pip insta
ll --upgrade pip
```

In [3]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
```

In [4]:
```
df = pd.read_csv(r'C:\Users\PC-Udaya\Downloads\Data Source (sales_transactions).csv')
```

In [5]:
```
df.head()
```

Out[5]:

	TransactionID	CustomerID	TransactionDate	ProductID	ProductCategory	Quantity	PricePerUnit	TotalAmount
0	1	1002.0	08/08/24 22:00	2008	Grocery	1	10.0	10.0
1	2	NaN	07/08/24 1:00	2004	Home Decor	1	10.0	10.0
2	3	1004.0	02/08/24 19:00	2002	Grocery	3	30.0	90.0
3	2	1003.0	07/08/24 17:00	2001	Toys	2	30.0	60.0
4	5	1001.0	09/08/24 9:00	2008	Grocery	1	NaN	NaN

In [6]:
```
pd.set_option('display.max_rows',None)
```

In [7]:
```
df_original = df.copy()
```

In [8]:
```
df.isnull().sum()
```

Out[8]:
```
TransactionID       0
CustomerID          5
TransactionDate     1
ProductID           0
ProductCategory     0
Quantity            0
PricePerUnit       14
TotalAmount        14
TrustPointsUsed     0
PaymentMethod      10
DiscountApplied     5
dtype: int64
```

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

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 50 entries, 0 to 49
Data columns (total 11 columns):
 #   Column           Non-Null Count  Dtype
---  ------           --------------  -----
 0   TransactionID    50 non-null     int64
 1   CustomerID       45 non-null     float64
 2   TransactionDate  49 non-null     object
 3   ProductID        50 non-null     int64
 4   ProductCategory  50 non-null     object
 5   Quantity         50 non-null     int64
 6   PricePerUnit     36 non-null     float64
 7   TotalAmount      36 non-null     float64
 8   TrustPointsUsed  50 non-null     int64
 9   PaymentMethod    40 non-null     object
 10  DiscountApplied  45 non-null     float64
dtypes: float64(4), int64(4), object(3)
memory usage: 4.4+ KB
```

Data Cleaning

In [10]:
```
df['PaymentMethod'].fillna(df['PaymentMethod'].mode()[0],inplace=True)
```

```
C:\Users\PC-Udaya\AppData\Local\Temp\ipykernel_9760\3440700897.py:1: FutureWarning: A value is trying to b
e set on a copy of a DataFrame or Series through chained assignment using an inplace method.
The behavior will change in pandas 3.0. This inplace method will never work because the intermediate objec
t on which we are setting values always behaves as a copy.

For example, when doing 'df[col].method(value, inplace=True)', try using 'df.method({col: value}, inplace=
True)' or df[col] = df[col].method(value) instead, to perform the operation inplace on the original objec
t.


  df['PaymentMethod'].fillna(df['PaymentMethod'].mode()[0],inplace=True)
```

In [11]:
```
df['DiscountApplied'].fillna(0,inplace=True)
```

```
C:\Users\PC-Udaya\AppData\Local\Temp\ipykernel_9760\65426650.py:1: FutureWarning: A value is trying to be
set on a copy of a DataFrame or Series through chained assignment using an inplace method.
The behavior will change in pandas 3.0. This inplace method will never work because the intermediate objec
t on which we are setting values always behaves as a copy.

For example, when doing 'df[col].method(value, inplace=True)', try using 'df.method({col: value}, inplace=
True)' or df[col] = df[col].method(value) instead, to perform the operation inplace on the original objec
t.


  df['DiscountApplied'].fillna(0,inplace=True)
```

There is no fixed or consistent value of ProductCategory as compare to PricePerUnit so i will fill 0 to NaN in PricePerUnit and TotalAmount.

In [12]:
```
df['PricePerUnit'].fillna(0,inplace=True)
```

```
C:\Users\PC-Udaya\AppData\Local\Temp\ipykernel_9760\1992857639.py:1: FutureWarning: A value is trying to b
e set on a copy of a DataFrame or Series through chained assignment using an inplace method.
The behavior will change in pandas 3.0. This inplace method will never work because the intermediate objec
t on which we are setting values always behaves as a copy.

For example, when doing 'df[col].method(value, inplace=True)', try using 'df.method({col: value}, inplace=
True)' or df[col] = df[col].method(value) instead, to perform the operation inplace on the original objec
t.


  df['PricePerUnit'].fillna(0,inplace=True)
```

In [13]:
```
df['TotalAmount'].fillna(0,inplace=True)
```

```
C:\Users\PC-Udaya\AppData\Local\Temp\ipykernel_9760\1543364000.py:1: FutureWarning: A value is trying to b
e set on a copy of a DataFrame or Series through chained assignment using an inplace method.
The behavior will change in pandas 3.0. This inplace method will never work because the intermediate objec
t on which we are setting values always behaves as a copy.

For example, when doing 'df[col].method(value, inplace=True)', try using 'df.method({col: value}, inplace=
True)' or df[col] = df[col].method(value) instead, to perform the operation inplace on the original objec
t.


  df['TotalAmount'].fillna(0,inplace=True)
```

In [14]:
```
df['TransactionDate'] = pd.to_datetime(df['TransactionDate'])
```

```
C:\Users\PC-Udaya\AppData\Local\Temp\ipykernel_9760\390071961.py:1: UserWarning: Could not infer format, s
o each element will be parsed individually, falling back to `dateutil`. To ensure parsing is consistent an
d as-expected, please specify a format.
  df['TransactionDate'] = pd.to_datetime(df['TransactionDate'])
```

Note-Ignoring NaN in CustomerID focusing on product sales or payment methods.

This can provide insights into how many transactions occur without customer identification. Otherwise, i will remove or replace NaN values with "unknown".

In [15]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 50 entries, 0 to 49
Data columns (total 11 columns):
 #   Column           Non-Null Count  Dtype
---  ------           --------------  -----
 0   TransactionID    50 non-null     int64
 1   CustomerID       45 non-null     float64
 2   TransactionDate  49 non-null     datetime64[ns]
 3   ProductID        50 non-null     int64
 4   ProductCategory  50 non-null     object
 5   Quantity         50 non-null     int64
 6   PricePerUnit     50 non-null     float64
 7   TotalAmount      50 non-null     float64
 8   TrustPointsUsed  50 non-null     int64
 9   PaymentMethod    50 non-null     object
 10  DiscountApplied  50 non-null     float64
dtypes: datetime64[ns](1), float64(4), int64(4), object(2)
memory usage: 4.4+ KB
```

In [16]:
```
df['Quantity'].replace(to_replace=[-1],value=1,inplace=True)
```

```
C:\Users\PC-Udaya\AppData\Local\Temp\ipykernel_9760\7475304.py:1: FutureWarning: A value is trying to be s
et on a copy of a DataFrame or Series through chained assignment using an inplace method.
The behavior will change in pandas 3.0. This inplace method will never work because the intermediate objec
t on which we are setting values always behaves as a copy.

For example, when doing 'df[col].method(value, inplace=True)', try using 'df.method({col: value}, inplace=
True)' or df[col] = df[col].method(value) instead, to perform the operation inplace on the original objec
t.


  df['Quantity'].replace(to_replace=[-1],value=1,inplace=True)
```

In [17]:
```
df['TotalAmount'] = df['TotalAmount'].abs()
```

Data Analysis

In [18]:
```
df
```

Out[18]:

	TransactionID	CustomerID	TransactionDate	ProductID	ProductCategory	Quantity	PricePerUnit	TotalAmount
0	1	1002.0	2024-08-08 22:00:00	2008	Grocery	1	10.0	10.0
1	2	NaN	2024-07-08 01:00:00	2004	Home Decor	1	10.0	10.0
2	3	1004.0	2024-02-08 19:00:00	2002	Grocery	3	30.0	90.0
3	2	1003.0	2024-07-08 17:00:00	2001	Toys	2	30.0	60.0
4	5	1001.0	2024-09-08 09:00:00	2008	Grocery	1	0.0	0.0
5	6	1001.0	NaT	2007	Home Decor	1	0.0	0.0
6	7	1001.0	2024-01-08 13:00:00	2007	Home Decor	1	30.0	30.0
7	8	1005.0	2024-04-08 22:00:00	2006	Toys	1	50.0	50.0
8	9	1004.0	2024-02-08 23:00:00	2008	Fashion	1	0.0	0.0
9	10	1004.0	2024-01-08 14:00:00	2005	Fashion	2	500.0	1000.0
10	11	1001.0	2024-09-08 07:00:00	2003	Grocery	5	0.0	0.0
11	12	NaN	2024-09-08 13:00:00	2004	Electronics	1	10.0	10.0
12	13	1005.0	2024-09-08 22:00:00	2008	Grocery	3	0.0	0.0

	TransactionID	CustomerID	TransactionDate	ProductID	ProductCategory	Quantity	PricePerUnit	TotalAmount
13	14	1002.0	2024-08-08 21:00:00	2006	Toys	3	50.0	150.0
14	15	1001.0	2024-02-08 15:00:00	2003	Toys	1	30.0	30.0
15	16	1001.0	2024-09-08 20:00:00	2003	Toys	3	10.0	30.0
16	17	1002.0	2024-09-08 15:00:00	2001	Toys	1	0.0	0.0
17	18	1003.0	2024-04-08 09:00:00	2003	Toys	3	50.0	150.0
18	19	1003.0	2024-05-08 14:00:00	2005	Home Decor	3	500.0	1500.0
19	20	1002.0	2024-03-08 04:00:00	2007	Fashion	2	0.0	0.0
20	21	1004.0	2024-01-08 23:00:00	2006	Toys	1	50.0	50.0
21	22	1001.0	2024-07-08 09:00:00	2003	Electronics	0	500.0	0.0
22	23	1002.0	2024-10-08 00:00:00	2001	Electronics	1	30.0	30.0
23	24	1002.0	2024-08-08 19:00:00	2005	Electronics	1	0.0	0.0
24	25	1003.0	2024-06-08 03:00:00	2002	Electronics	1	500.0	500.0
25	26	1004.0	2024-02-08 16:00:00	2007	Home Decor	2	10.0	20.0

	TransactionID	CustomerID	TransactionDate	ProductID	ProductCategory	Quantity	PricePerUnit	TotalAmount
26	27	1001.0	2024-07-08 12:00:00	2007	Electronics	3	30.0	90.0
27	28	1003.0	2024-01-08 14:00:00	2006	Toys	1	10.0	10.0
28	29	1003.0	2024-02-08 20:00:00	2007	Grocery	2	100.0	200.0
29	30	1001.0	2024-03-08 16:00:00	2003	Electronics	1	20.0	20.0
30	31	1004.0	2024-04-08 16:00:00	2001	Home Decor	3	0.0	0.0
31	32	1001.0	2024-03-08 22:00:00	2007	Home Decor	1	500.0	500.0
32	33	1001.0	2024-01-08 08:00:00	2007	Electronics	3	20.0	60.0
33	34	NaN	2024-04-08 15:00:00	2002	Grocery	1	20.0	20.0
34	35	NaN	2024-06-08 08:00:00	2002	Electronics	1	500.0	500.0
35	36	1005.0	2024-06-08 15:00:00	2004	Home Decor	1	30.0	30.0
36	37	1002.0	2024-09-08 23:00:00	2005	Fashion	1	0.0	0.0
37	38	1001.0	2024-03-08 14:00:00	2003	Toys	1	100.0	100.0
38	39	1004.0	2024-06-08 18:00:00	2007	Home Decor	1	10.0	10.0

	TransactionID	CustomerID	TransactionDate	ProductID	ProductCategory	Quantity	PricePerUnit	TotalAmount
39	40	1003.0	2024-04-08 08:00:00	2008	Electronics	0	50.0	0.0
40	41	1001.0	2024-06-08 15:00:00	2007	Toys	1	10.0	10.0
41	42	1003.0	2024-07-08 18:00:00	2001	Electronics	3	50.0	150.0
42	43	1001.0	2024-07-08 18:00:00	2004	Electronics	0	10.0	0.0
43	44	NaN	2024-02-08 08:00:00	2005	Fashion	2	500.0	1000.0
44	45	1002.0	2024-06-08 02:00:00	2008	Toys	2	100.0	200.0
45	46	1004.0	2024-01-08 04:00:00	2004	Toys	1	0.0	0.0
46	47	1002.0	2024-02-08 16:00:00	2006	Fashion	1	50.0	50.0
47	48	1003.0	2024-02-08 03:00:00	2005	Home Decor	0	0.0	0.0
48	49	1003.0	2024-06-08 14:00:00	2007	Electronics	1	0.0	0.0
49	50	1001.0	2024-09-08 08:00:00	2007	Grocery	3	0.0	0.0

In [19]:
```
df.describe()
```

Out[19]:

	TransactionID	CustomerID	TransactionDate	ProductID	Quantity	PricePerUnit	TotalAmount	TrustPoin
count	50.000000	45.000000	49	50.000000	50.000000	50.00000	50.000000	50.
mean	25.460000	1002.444444	2024-05-11 17:58:46.530612480	2004.920000	1.580000	90.20000	133.400000	28.
min	1.000000	1001.000000	2024-01-08 04:00:00	2001.000000	0.000000	0.00000	0.000000	-10.
25%	13.250000	1001.000000	2024-02-08 20:00:00	2003.000000	1.000000	0.00000	0.000000	0.
50%	25.500000	1002.000000	2024-06-08 02:00:00	2005.000000	1.000000	20.00000	20.000000	20.
75%	37.750000	1003.000000	2024-07-08 18:00:00	2007.000000	2.000000	50.00000	90.000000	50.
max	50.000000	1005.000000	2024-10-08 00:00:00	2008.000000	5.000000	500.00000	1500.000000	100.
std	14.640188	1.306549	NaN	2.284285	1.051529	168.97639	296.404371	39.

In [20]:
```
sales_by_category = df.groupby('ProductCategory')['TotalAmount'].sum()
```

In [21]:
```
price_per_category = df.groupby('ProductCategory')['PricePerUnit'].mean()
```

In [35]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 50 entries, 0 to 49
Data columns (total 11 columns):
 #   Column           Non-Null Count  Dtype
---  ------           --------------  -----
 0   TransactionID    50 non-null     int64
 1   CustomerID       45 non-null     float64
 2   TransactionDate  49 non-null     datetime64[ns]
 3   ProductID        50 non-null     int64
 4   ProductCategory  50 non-null     object
 5   Quantity         50 non-null     int64
 6   PricePerUnit     50 non-null     float64
 7   TotalAmount      50 non-null     float64
 8   TrustPointsUsed  50 non-null     int64
 9   PaymentMethod    50 non-null     object
 10  DiscountApplied  50 non-null     float64
dtypes: datetime64[ns](1), float64(4), int64(4), object(2)
memory usage: 4.4+ KB
```

Pivot tables

Understand which customers contribute the most to sales.

In [23]:
```
sale_by_customer = pd.pivot_table(df,values='TotalAmount',index='CustomerID',aggfunc=np.sum)
```

```
C:\Users\PC-Udaya\AppData\Local\Temp\ipykernel_9760\1063842485.py:1: FutureWarning: The provided callable
<function sum at 0x000001E4BF2CDB20> is currently using DataFrameGroupBy.sum. In a future version of panda
s, the provided callable will be used directly. To keep current behavior pass the string "sum" instead.
  sale_by_customer = pd.pivot_table(df,values='TotalAmount',index='CustomerID',aggfunc=np.sum)
```

In [24]:
```
sale_by_customer
```

Out[24]:

CustomerID	TotalAmount
1001.0	870.0
1002.0	440.0
1003.0	2570.0
1004.0	1170.0
1005.0	80.0

payment methods receive the highest average discounts.

In [25]:
```
Discount_by_payment = pd.pivot_table(df, values='DiscountApplied', index='PaymentMethod', aggfunc=np.mean)
```

```
C:\Users\PC-Udaya\AppData\Local\Temp\ipykernel_9760\2706009581.py:1: FutureWarning: The provided callable
<function mean at 0x000001E4BF2CEC00> is currently using DataFrameGroupBy.mean. In a future version of pan
das, the provided callable will be used directly. To keep current behavior pass the string "mean" instead.
  Discount_by_payment = pd.pivot_table(df, values='DiscountApplied', index='PaymentMethod', aggfunc=np.mea
n)
```

Track sales trends over time for different product categories.

In [26]:
```
sales_by_date_category = pd.pivot_table(df, values='TotalAmount', index='TransactionDate', columns='Produ
```

```
C:\Users\PC-Udaya\AppData\Local\Temp\ipykernel_9760\4072295229.py:1: FutureWarning: The provided callable
<function sum at 0x000001E4BF2CDB20> is currently using DataFrameGroupBy.sum. In a future version of panda
s, the provided callable will be used directly. To keep current behavior pass the string "sum" instead.
  sales_by_date_category = pd.pivot_table(df, values='TotalAmount', index='TransactionDate', columns='Prod
uctCategory', aggfunc=np.sum)
```

Trust points are being used across different product categories.

In [27]:
```
Trustpoint_by_category = pd.pivot_table(df,values='TrustPointsUsed',index='ProductCategory',aggfunc=np.sur
```

```
C:\Users\PC-Udaya\AppData\Local\Temp\ipykernel_9760\1919594437.py:1: FutureWarning: The provided callable
<function sum at 0x000001E4BF2CDB20> is currently using DataFrameGroupBy.sum. In a future version of panda
s, the provided callable will be used directly. To keep current behavior pass the string "sum" instead.
  Trustpoint_by_category = pd.pivot_table(df,values='TrustPointsUsed',index='ProductCategory',aggfunc=np.s
um)
```

Identify which product categories are most frequently purchased.

In [28]:
```
Quantity_by_category = pd.pivot_table(df, values='Quantity', index='ProductCategory', aggfunc=np.sum)
```

```
C:\Users\PC-Udaya\AppData\Local\Temp\ipykernel_9760\193182128.py:1: FutureWarning: The provided callable <
function sum at 0x000001E4BF2CDB20> is currently using DataFrameGroupBy.sum. In a future version of panda
s, the provided callable will be used directly. To keep current behavior pass the string "sum" instead.
  Quantity_by_category = pd.pivot_table(df, values='Quantity', index='ProductCategory', aggfunc=np.sum)
```

Average transaction value for each product category.

In [29]:
```
Avg_sales_by_category = pd.pivot_table(df, values='TotalAmount', index='ProductCategory', aggfunc=np.mean)
```

```
C:\Users\PC-Udaya\AppData\Local\Temp\ipykernel_9760\3639401149.py:1: FutureWarning: The provided callable
<function mean at 0x000001E4BF2CEC00> is currently using DataFrameGroupBy.mean. In a future version of pan
das, the provided callable will be used directly. To keep current behavior pass the string "mean" instead.
  Avg_sales_by_category = pd.pivot_table(df, values='TotalAmount', index='ProductCategory', aggfunc=np.mea
n)
```

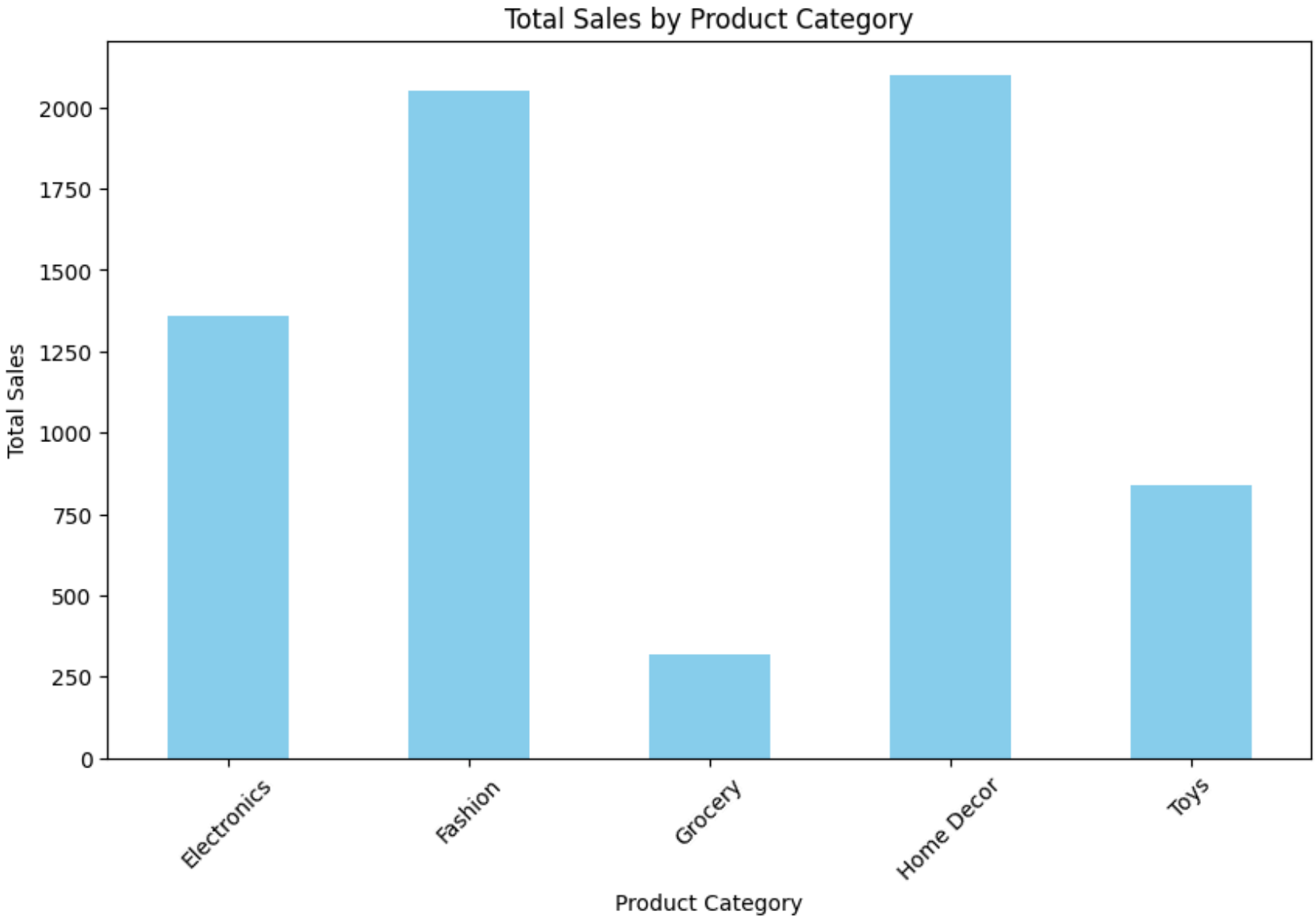
Data Visualization

Total Sales by Product Category

In [55]:
```
category_sales = df.groupby('ProductCategory')['TotalAmount'].sum()
```

In [60]:
```
plt.figure(figsize=(10,6))
category_sales.plot(kind='bar', color='skyblue')
```

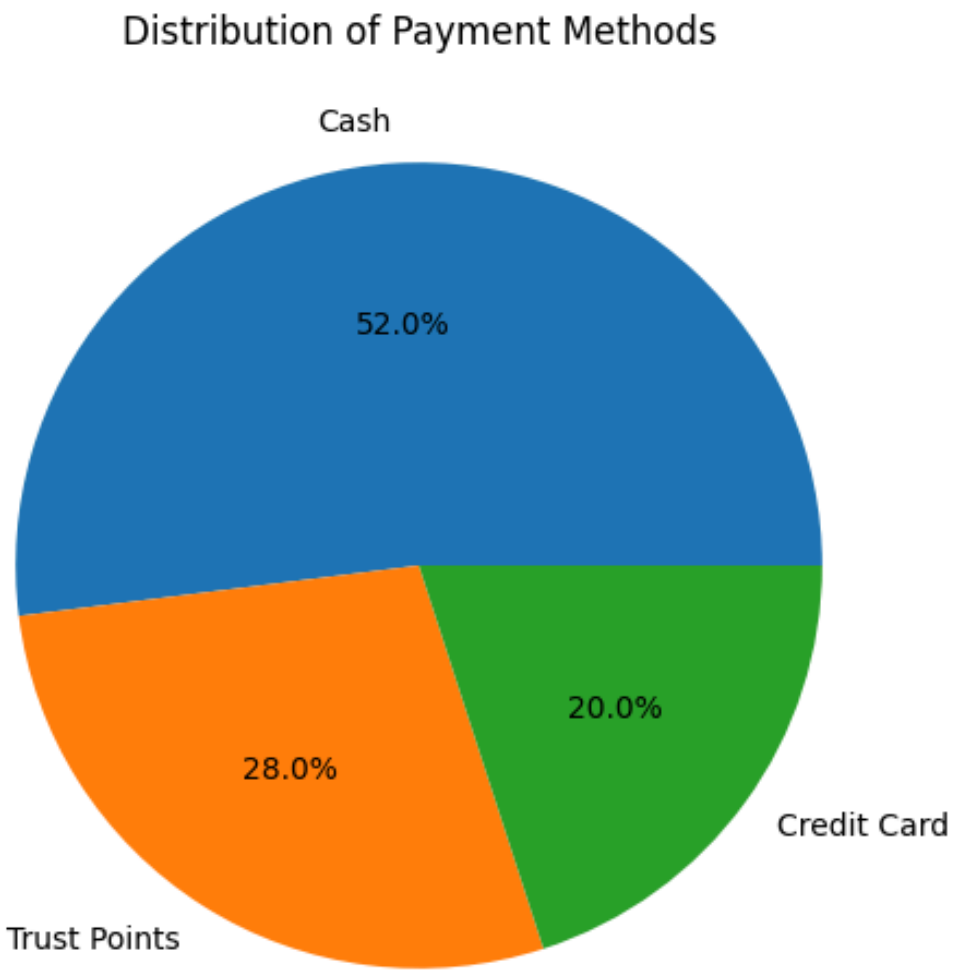
```
plt.title('Total Sales by Product Category')
plt.xlabel('Product Category')
plt.ylabel('Total Sales')
plt.xticks(rotation=45)
plt.show()
```



Distribution of Payment Methods

In [63]:
```
payment_counts = df['PaymentMethod'].value_counts()
```

In [64]:
```
plt.figure(figsize=(10,6))
payment_counts.plot(kind='pie',autopct='%1.1f%%')
plt.title('Distribution of Payment Methods')
plt.ylabel('')
plt.show()
```

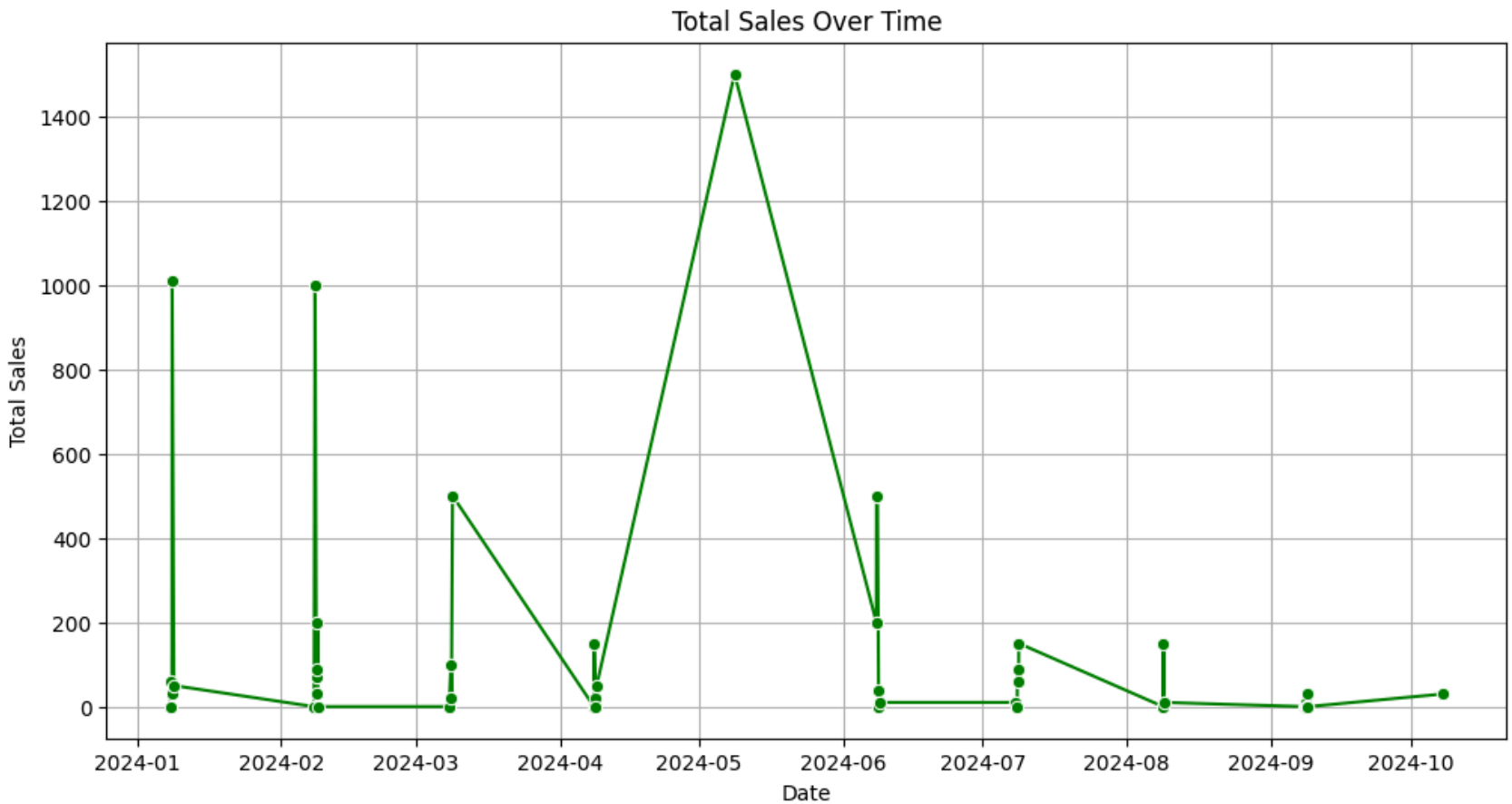


visualize trends in sales over time

In [72]:
```
sales_over_time = df.groupby('TransactionDate')['TotalAmount'].sum()
```

In [75]:
```
sales_over_time = df.groupby('TransactionDate')['TotalAmount'].sum().reset_index()
plt.figure(figsize=(12, 6))
sns.lineplot(x='TransactionDate', y='TotalAmount', data=sales_over_time, marker='o', color='green')
```

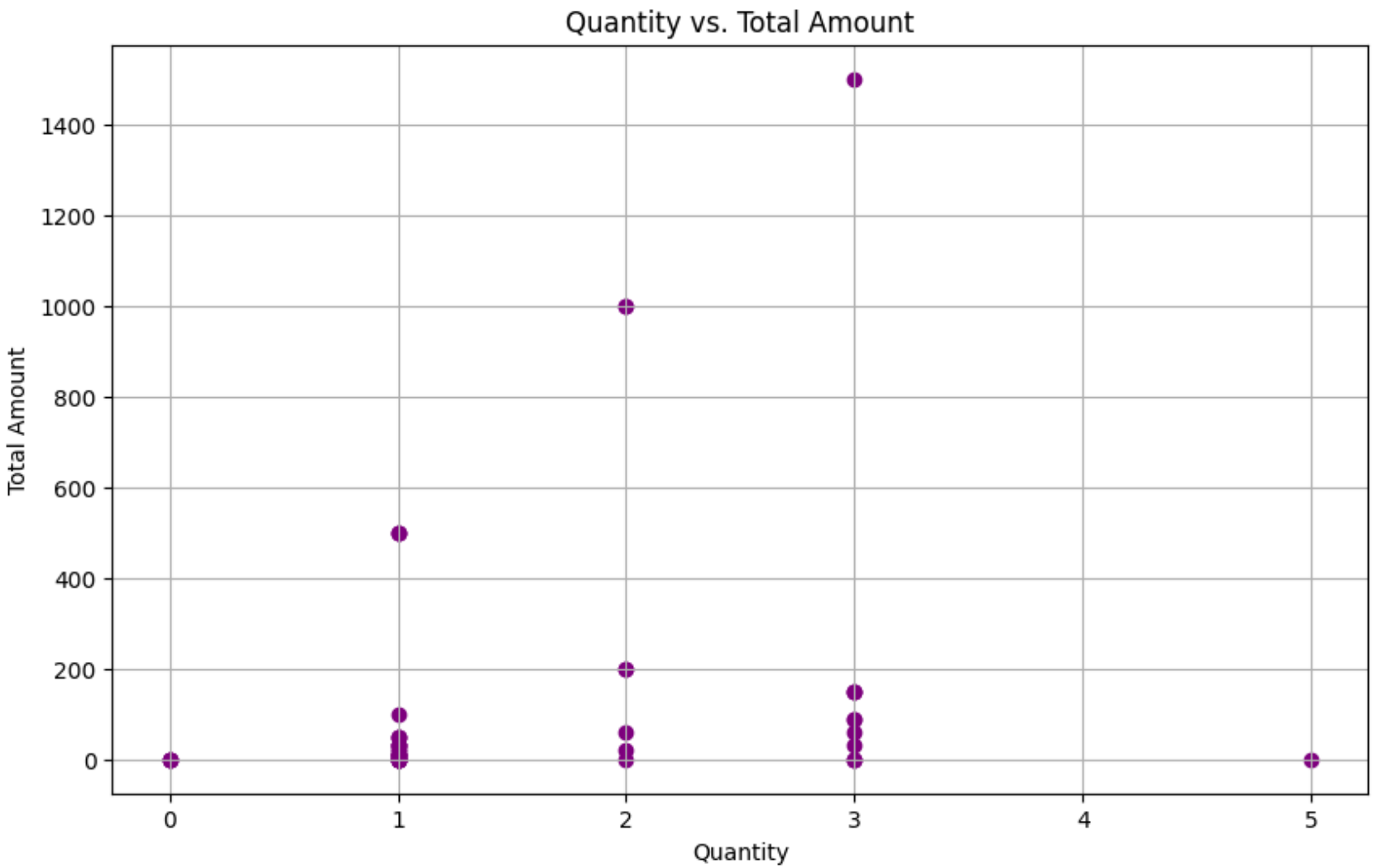
```
plt.title('Total Sales Over Time')
plt.xlabel('Date')
plt.ylabel('Total Sales')
plt.grid()
plt.show()
```



Shows the relationship between the quantity of items sold and the total amount, highlighting trends such as bulk purchases.

In [68]:
```
plt.figure(figsize=(10, 6))
plt.scatter(df['Quantity'], df['TotalAmount'], color='purple')
plt.title('Quantity vs. Total Amount')
plt.xlabel('Quantity')
```

```
plt.ylabel('Total Amount')
plt.grid(True)
plt.show()
```



In [78]:

In [ ]: