1. DATA PREPROCESSING

In [2]:
```
import pandas as pd
import numpy as np
df=pd.read_csv('C:\\Users\\varku\\Downloads\\DataSource.csv')
print(df)
```

	TransactionID	CustomerID	TransactionDate	ProductID	ProductCategory
0	1	1002.0	08-08-2024 22:00	2008	Grocery
1	2	NaN	07-08-2024 01:00	2004	Home Decor
2	3	1004.0	02-08-2024 19:00	2002	Grocery
3	2	1003.0	07-08-2024 17:00	2001	Toys
4	5	1001.0	09-08-2024 09:00	2008	Grocery
5	6	1001.0	NaN	2007	Home Decor
6	7	1001.0	01-08-2024 13:00	2007	Home Decor
7	8	1005.0	04-08-2024 22:00	2006	Toys
8	9	1004.0	02-08-2024 23:00	2008	Fashion
9	10	1004.0	01-08-2024 14:00	2005	Fashion
10	11	1001.0	09-08-2024 07:00	2003	Grocery
11	12	NaN	09-08-2024 13:00	2004	Electronics
12	13	1005.0	09-08-2024 22:00	2008	Grocery
13	14	1002.0	08-08-2024 21:00	2006	Toys
14	15	1001.0	02-08-2024 15:00	2003	Toys
15	16	1001.0	09-08-2024 20:00	2003	Toys
16	17	1002.0	09-08-2024 15:00	2001	Toys
17	18	1003.0	04-08-2024 09:00	2003	Toys
18	19	1003.0	05-08-2024 14:00	2005	Home Decor
19	20	1002.0	03-08-2024 04:00	2007	Fashion
20	21	1004.0	01-08-2024 23:00	2006	Toys
21	22	1001.0	07-08-2024 09:00	2003	Electronics
22	23	1002.0	10-08-2024 00:00	2001	Electronics
23	24	1002.0	08-08-2024 19:00	2005	Electronics
24	25	1003.0	06-08-2024 03:00	2002	Electronics
25	26	1004.0	02-08-2024 16:00	2007	Home Decor
26	27	1001.0	07-08-2024 12:00	2007	Electronics
27	28	1003.0	01-08-2024 14:00	2006	Toys
28	29	1003.0	02-08-2024 20:00	2007	Grocery
29	30	1001.0	03-08-2024 16:00	2003	Electronics
30	31	1004.0	04-08-2024 16:00	2001	Home Decor
31	32	1001.0	03-08-2024 22:00	2007	Home Decor
32	33	1001.0	01-08-2024 08:00	2007	Electronics
33	34	NaN	04-08-2024 15:00	2002	Grocery
34	35	NaN	06-08-2024 08:00	2002	Electronics
35	36	1005.0	06-08-2024 15:00	2004	Home Decor
36	37	1002.0	09-08-2024 23:00	2005	Fashion
37	38	1001.0	03-08-2024 14:00	2003	Toys
38	39	1004.0	06-08-2024 18:00	2007	Home Decor
39	40	1003.0	04-08-2024 08:00	2008	Electronics
40	41	1001.0	06-08-2024 15:00	2007	Toys
41	42	1003.0	07-08-2024 18:00	2001	Electronics
42	43	1001.0	07-08-2024 18:00	2004	Electronics

43	44	NaN	02-08-2024 08:00	2005	Fashion
44	45	1002.0	06-08-2024 02:00	2008	Toys
45	46	1004.0	01-08-2024 04:00	2004	Toys
46	47	1002.0	02-08-2024 16:00	2006	Fashion
47	48	1003.0	02-08-2024 03:00	2005	Home Decor
48	49	1003.0	06-08-2024 14:00	2007	Electronics
49	50	1001.0	09-08-2024 08:00	2007	Grocery

	Quantity	PricePerUnit	TotalAmount	TrustPointsUsed	PaymentMethod	\
0	1	10.0	10.0	20	Trust Points	
1	1	10.0	10.0	0	Credit Card	
2	3	30.0	90.0	0	Credit Card	
3	2	30.0	60.0	50	NaN	
4	1	NaN	NaN	20	Trust Points	
5	1	NaN	NaN	20	Credit Card	
6	-1	30.0	-30.0	-10	NaN	
7	1	50.0	50.0	-10	Trust Points	
8	1	NaN	NaN	-10	NaN	
9	2	500.0	1000.0	-10	Cash	
10	5	NaN	NaN	-10	Credit Card	
11	1	10.0	10.0	-10	Cash	
12	3	NaN	NaN	100	Trust Points	
13	3	50.0	150.0	20	Cash	
14	1	30.0	30.0	-10	Cash	
15	3	10.0	30.0	20	Trust Points	
16	1	NaN	NaN	20	NaN	
17	3	50.0	150.0	10	Cash	
18	3	500.0	1500.0	100	NaN	
19	2	NaN	NaN	-10	Cash	
20	1	50.0	50.0	0	NaN	
21	0	500.0	0.0	50	Trust Points	
22	1	30.0	30.0	0	Cash	
23	1	NaN	NaN	100	Trust Points	
24	1	500.0	500.0	50	Cash	
25	2	10.0	20.0	100	Trust Points	
26	3	30.0	90.0	20	Trust Points	
27	1	10.0	10.0	50	Cash	
28	2	100.0	200.0	20	Cash	
29	1	20.0	20.0	0	Cash	
30	3	NaN	NaN	0	Credit Card	
31	1	500.0	500.0	50	NaN	
32	3	20.0	60.0	50	Credit Card	
33	1	20.0	20.0	-10	Credit Card	
34	-1	500.0	-500.0	100	Trust Points	

35	1	30.0	30.0	-10	Cash
36	1	NaN	NaN	100	NaN
37	1	100.0	100.0	-10	Trust Points
38	-1	10.0	-10.0	20	Credit Card
39	0	50.0	0.0	50	Cash
40	1	10.0	10.0	0	Trust Points
41	3	50.0	150.0	100	Cash
42	0	10.0	0.0	100	Trust Points
43	2	500.0	1000.0	0	Cash
44	2	100.0	200.0	-10	Credit Card
45	1	NaN	NaN	100	NaN
46	1	50.0	50.0	20	Credit Card
47	0	NaN	NaN	50	NaN
48	-1	NaN	NaN	0	Cash
49	3	NaN	NaN	50	Trust Points

	DiscountApplied
0	5.0
1	20.0
2	25.0
3	20.0
4	5.0
5	NaN
6	NaN
7	30.0
8	NaN
9	30.0
10	20.0
11	15.0
12	50.0
13	50.0
14	20.0
15	10.0
16	10.0
17	NaN
18	50.0
19	50.0
20	30.0
21	15.0
22	5.0
23	10.0
24	50.0
25	30.0
26	25.0

```
27    30.0
28    10.0
29    10.0
30    30.0
31     5.0
32    20.0
33    10.0
34    30.0
35    50.0
36    30.0
37    15.0
38    20.0
39    25.0
40     5.0
41     NaN
42    25.0
43    20.0
44    20.0
45    20.0
46    25.0
47    50.0
48    20.0
49    30.0
```

In [3]:
```
df.head()
```

Out[3]:

	TransactionID	CustomerID	TransactionDate	ProductID	ProductCategory	Quantity	PricePerUnit	TotalAmount	TrustPointsUsed	PaymentMethod	Dis
0	1	1002.0	08-08-2024 22:00	2008	Grocery	1	10.0	10.0	20	Trust Points	
1	2	NaN	07-08-2024 01:00	2004	Home Decor	1	10.0	10.0	0	Credit Card	
2	3	1004.0	02-08-2024 19:00	2002	Grocery	3	30.0	90.0	0	Credit Card	
3	2	1003.0	07-08-2024 17:00	2001	Toys	2	30.0	60.0	50	NaN	
4	5	1001.0	09-08-2024 09:00	2008	Grocery	1	NaN	NaN	20	Trust Points	

In [4]:
```
df.isnull().sum()
```

Out[4]:
```
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 [5]:
```
df=df.dropna()
print(df)
```

	TransactionID	CustomerID	TransactionDate	ProductID	ProductCategory
0	1	1002.0	08-08-2024 22:00	2008	Grocery
2	3	1004.0	02-08-2024 19:00	2002	Grocery
7	8	1005.0	04-08-2024 22:00	2006	Toys
9	10	1004.0	01-08-2024 14:00	2005	Fashion
13	14	1002.0	08-08-2024 21:00	2006	Toys
14	15	1001.0	02-08-2024 15:00	2003	Toys
15	16	1001.0	09-08-2024 20:00	2003	Toys
21	22	1001.0	07-08-2024 09:00	2003	Electronics
22	23	1002.0	10-08-2024 00:00	2001	Electronics
24	25	1003.0	06-08-2024 03:00	2002	Electronics
25	26	1004.0	02-08-2024 16:00	2007	Home Decor
26	27	1001.0	07-08-2024 12:00	2007	Electronics
27	28	1003.0	01-08-2024 14:00	2006	Toys
28	29	1003.0	02-08-2024 20:00	2007	Grocery
29	30	1001.0	03-08-2024 16:00	2003	Electronics
32	33	1001.0	01-08-2024 08:00	2007	Electronics
35	36	1005.0	06-08-2024 15:00	2004	Home Decor
37	38	1001.0	03-08-2024 14:00	2003	Toys
38	39	1004.0	06-08-2024 18:00	2007	Home Decor
39	40	1003.0	04-08-2024 08:00	2008	Electronics
40	41	1001.0	06-08-2024 15:00	2007	Toys
42	43	1001.0	07-08-2024 18:00	2004	Electronics
44	45	1002.0	06-08-2024 02:00	2008	Toys
46	47	1002.0	02-08-2024 16:00	2006	Fashion

	Quantity	PricePerUnit	TotalAmount	TrustPointsUsed	PaymentMethod
0	1	10.0	10.0	20	Trust Points
2	3	30.0	90.0	0	Credit Card
7	1	50.0	50.0	-10	Trust Points
9	2	500.0	1000.0	-10	Cash
13	3	50.0	150.0	20	Cash
14	1	30.0	30.0	-10	Cash
15	3	10.0	30.0	20	Trust Points
21	0	500.0	0.0	50	Trust Points
22	1	30.0	30.0	0	Cash
24	1	500.0	500.0	50	Cash
25	2	10.0	20.0	100	Trust Points
26	3	30.0	90.0	20	Trust Points
27	1	10.0	10.0	50	Cash
28	2	100.0	200.0	20	Cash
29	1	20.0	20.0	0	Cash
32	3	20.0	60.0	50	Credit Card
35	1	30.0	30.0	-10	Cash

```
37         1         100.0        100.0              -10  Trust Points
38        -1          10.0        -10.0               20   Credit Card
39         0          50.0          0.0               50          Cash
40         1          10.0         10.0                0  Trust Points
42         0          10.0          0.0              100  Trust Points
44         2         100.0        200.0              -10   Credit Card
46         1          50.0         50.0               20   Credit Card
```

```
    DiscountApplied
0               5.0
2              25.0
7              30.0
9              30.0
13             50.0
14             20.0
15             10.0
21             15.0
22              5.0
24             50.0
25             30.0
26             25.0
27             30.0
28             10.0
29             10.0
32             20.0
35             50.0
37             15.0
38             20.0
39             25.0
40              5.0
42             25.0
44             20.0
46             25.0
```

In [6]:
```
num_rows=len(df)
print(num_rows)
```

```
24
```

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

Out[7]:
```
TransactionID      0
CustomerID         0
TransactionDate    0
ProductID          0
ProductCategory    0
Quantity           0
PricePerUnit       0
TotalAmount        0
TrustPointsUsed    0
PaymentMethod      0
DiscountApplied    0
dtype: int64
```

In [8]:
```
df['TransactionDate']=pd.to_datetime(df['TransactionDate'])
print(df['TransactionDate'])
```

```
0    2024-08-08 22:00:00
2    2024-02-08 19:00:00
7    2024-04-08 22:00:00
9    2024-01-08 14:00:00
13   2024-08-08 21:00:00
14   2024-02-08 15:00:00
15   2024-09-08 20:00:00
21   2024-07-08 09:00:00
22   2024-10-08 00:00:00
24   2024-06-08 03:00:00
25   2024-02-08 16:00:00
26   2024-07-08 12:00:00
27   2024-01-08 14:00:00
28   2024-02-08 20:00:00
29   2024-03-08 16:00:00
32   2024-01-08 08:00:00
35   2024-06-08 15:00:00
37   2024-03-08 14:00:00
38   2024-06-08 18:00:00
39   2024-04-08 08:00:00
40   2024-06-08 15:00:00
42   2024-07-08 18:00:00
44   2024-06-08 02:00:00
46   2024-02-08 16:00:00
Name: TransactionDate, dtype: datetime64[ns]
```

In [9]:
```
df['TransactionMonth']=df['TransactionDate'].dt.month
df['TransactionYear']=df['TransactionDate'].dt.year
```

```
print(df['TransactionMonth'])
print(df['TransactionYear'])
```

```
0      8
2      2
7      4
9      1
13     8
14     2
15     9
21     7
22    10
24     6
25     2
26     7
27     1
28     2
29     3
32     1
35     6
37     3
38     6
39     4
40     6
42     7
44     6
46     2
Name: TransactionMonth, dtype: int64
0     2024
2     2024
7     2024
9     2024
13    2024
14    2024
15    2024
21    2024
22    2024
24    2024
25    2024
26    2024
27    2024
28    2024
29    2024
32    2024
35    2024
37    2024
38    2024
```

```
39    2024
40    2024
42    2024
44    2024
46    2024
Name: TransactionYear, dtype: int64
```

In [10]:
```
from sklearn.preprocessing import StandardScaler
scaler = StandardScaler()
df[['Quantity','PricePerUnit','TotalAmount']]=scaler.fit_transform(df[['Quantity','PricePerUnit','TotalAmount']])
```

In [11]:
```
df.head()
```

Out[11]:

	TransactionID	CustomerID	TransactionDate	ProductID	ProductCategory	Quantity	PricePerUnit	TotalAmount	TrustPointsUsed	PaymentMethod	I
0	1	1002.0	2024-08-08 22:00:00	2008	Grocery	-0.349531	-0.539937	-0.474003	20	Trust Points	
2	3	1004.0	2024-02-08 19:00:00	2002	Grocery	1.514634	-0.411635	-0.099482	0	Credit Card	
7	8	1005.0	2024-04-08 22:00:00	2006	Toys	-0.349531	-0.283333	-0.286743	-10	Trust Points	
9	10	1004.0	2024-01-08 14:00:00	2005	Fashion	0.582552	2.603459	4.160694	-10	Cash	
13	14	1002.0	2024-08-08 21:00:00	2006	Toys	1.514634	-0.283333	0.181409	20	Cash	

2. WRITE THE STEPS YOU FOLLOWED

1. Imported Necessary Libraries.
2. Loaded the csv file into a 2D DataFrame.
3. viewed the very first rows.
4. Checking for missing values in the DataFrame.
5. Droping Missing values.

6. Checking no.of rows present after droping missing values.
7. Converted 'TransactionDate' to datetime.
8. Gets new features like 'TrasactionMonth' and 'TransactionYear'.
9. Standardized numerical features for consistency.
10. Reviewed and saved the cleaned data.

3. WHAT WAS YOUR FIRST THOUGHT PROCESS WHEN YOU FIRST SAW THE DATA

When I first saw the data , My step is to understand it's structure and identify key columns . After that checking for missing values and inconsistencies in the dataset,particularly in critical fields like prices and totals. I also looked for creating useful features like extracting Month & Year from TransactionDate.

DATA AGGREGATION AND GROUPING

1. WHAT ALL FIELDS AMONG THEM YOU THINK CAN BE AGGREGATED? NAME THEM.

Among the fields in the dataset, the following can be aggregated:

1. Quantity.
2. PricePerUnit.
3. TotalAmount.
4. TrustPointsUsed.
5. DiscountApplied.

WHAT KIND OF AGGREGATION (FOR EVERY COLUMN) WOULD MAKE SENSE AND WHY?

1. Quantity : Indicates total volume of products sold.
2. PricePerUnit : Indicates the average selling price of products.
3. TotalAmount : Shows the total revenue generated from all transactions.
4. TrustPointsUsed : Shows the total number of trust points used across transactions.
5. DiscountApplied : Indicates the total value of discounts given. Provides insight into the typical discount amount applied.

DATA VALIDATION

1. HOW DO YOU KNOW , YOUR PREPROCESSING WAS CORRECT?

1. Validate that all columns have the correct data types and no unexpected nulls remain.
2. Use visualizations to confirm the data looks as expected without any strange patterns.
3. Check for logical consistency, like verifying TotalAmount = Quantity * PricePerUnit.
4. Apply preprocessing to a small sample of the data first to catch any potential issues before processing the entire dataset.

2. HOW WILL YOU VALIDATE YOUR RESULTS?

1. Compare the preprocessed data with the raw data.
2. Manually inspect some rows before and after preprocessing to verify that specific changes were actually applied.
3. Validate that aggregated values are logical and match previous trends.

4. Apply preprocessing on different subsets of data to check if the results are consistent.

3. DO YOU FOLLOW ANY SPECIFIC VALIDATION PROCESS FOR ALL QUESTIONS? EXPLAIN.

1. Outline the steps needed to achieve the objective, considering potential challenges and validation points along the way.
2. Begin by exploring the data to identify any issues like missing values, inconsistencies, or outliers.
3. After each preprocessing step , validate the results by comparing before and after.

4. WHAT ARE THE EDGE CASES YOU CAN THINK OF?

1. Key fields like 'TransactionID' or 'ProductID' having missing values.
2. Missing components in date fields can lead to incorrect time-based analysis.
3. Outliers in 'PricePerUnit','Quantity or 'TotalAmount' can lead to false aggregate values.
4. Field component with zero might indicates a error.
5. Duplicate Data leads to incorrect conclusions if not removed.

5. WHAT ALL DATA INTEGRITY POINTS YOU WANT TO MENTION FOR THE GIVEN SCENARIO?

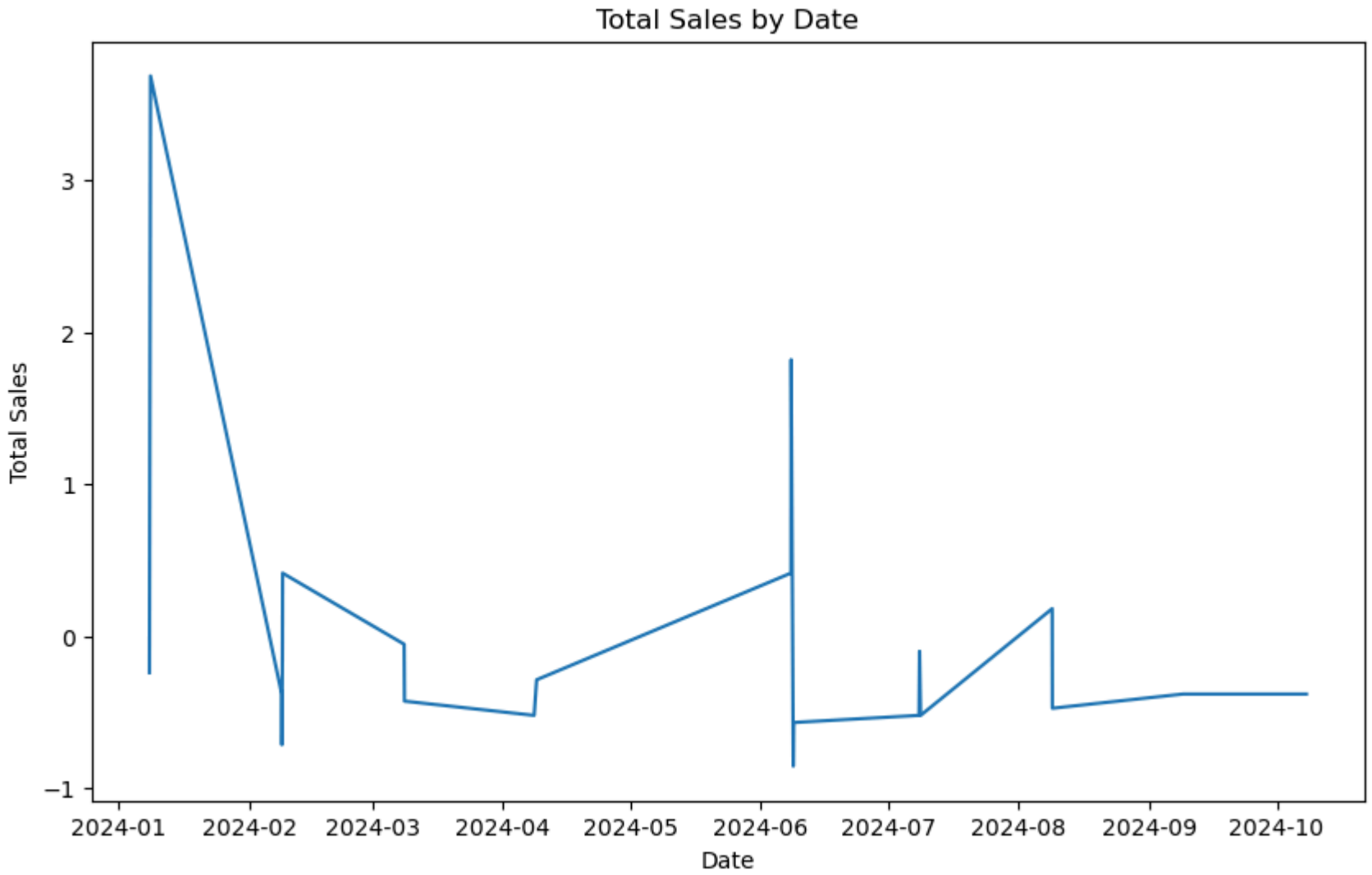
1. Accuracy.
2. Completeness.

3. Consistency.
4. Validation

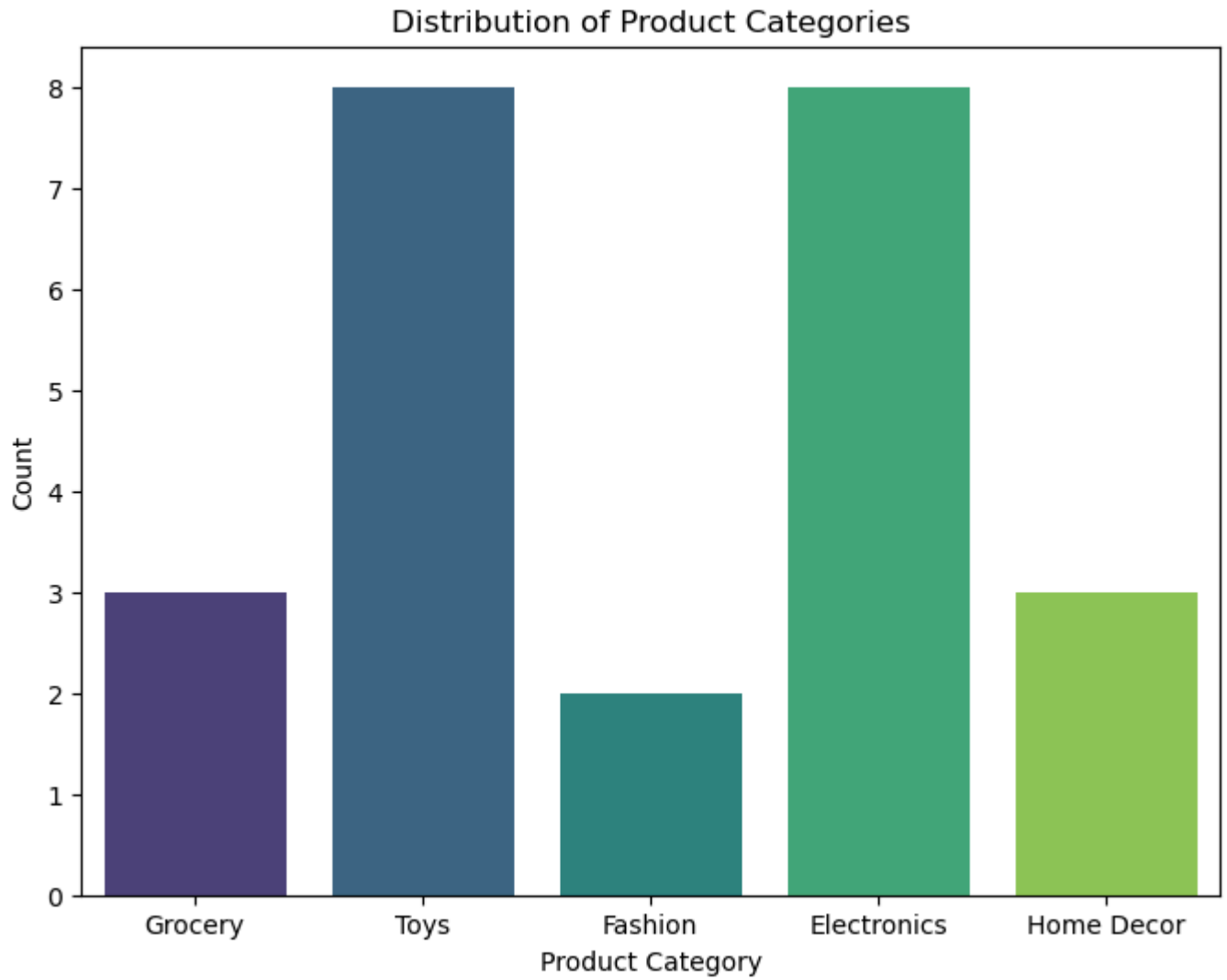
DATA VISUALIZATION

1. What all projections are possible out of the data.

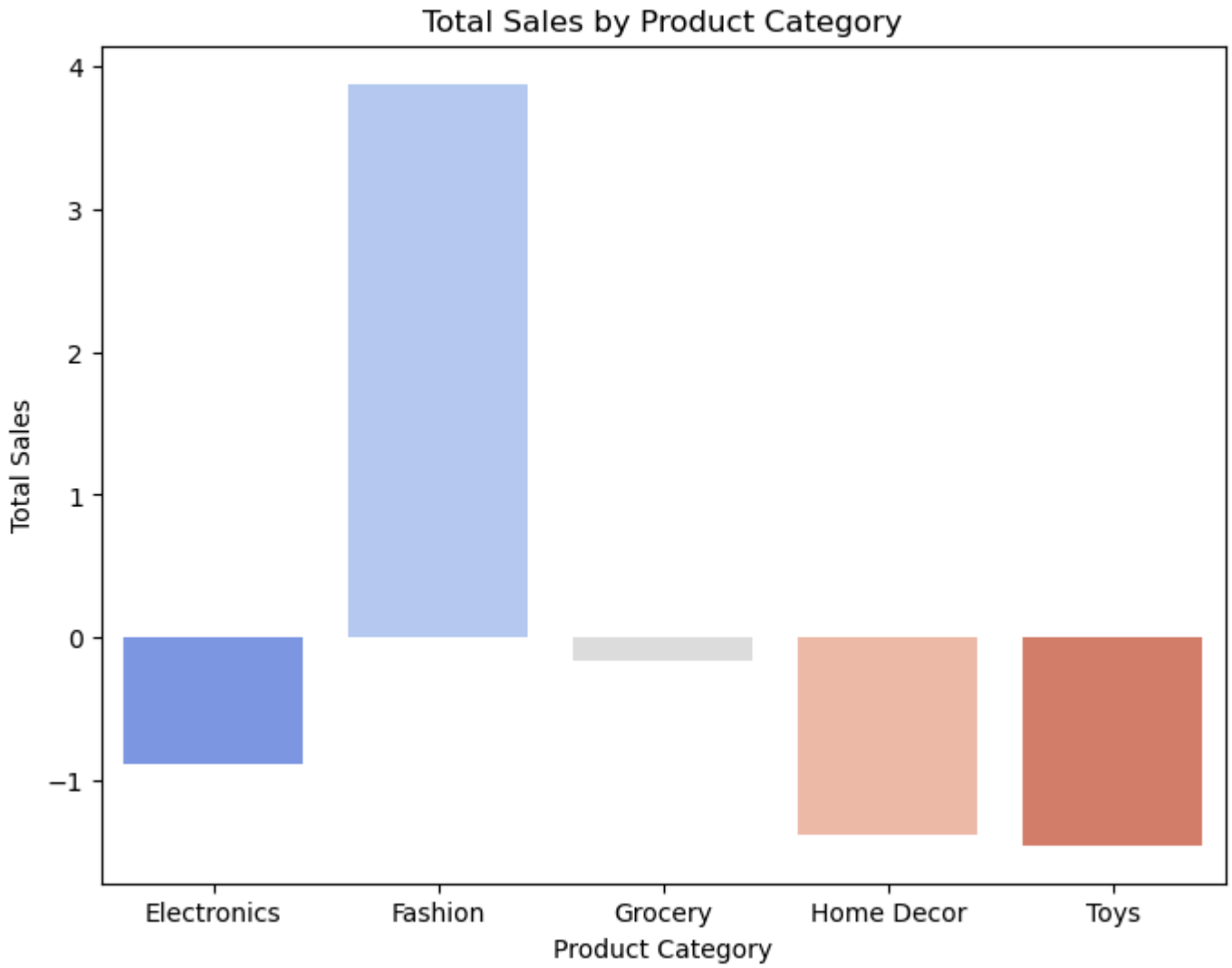
In [16]:
```
import matplotlib.pyplot as plt
import seaborn as sns

sales_by_date = df.groupby('TransactionDate')['TotalAmount'].sum().reset_index()

plt.figure(figsize=(10, 6))
sns.lineplot(x='TransactionDate', y='TotalAmount', data=sales_by_date)
plt.title('Total Sales by Date')
plt.xlabel('Date')
plt.ylabel('Total Sales')
plt.show()
```



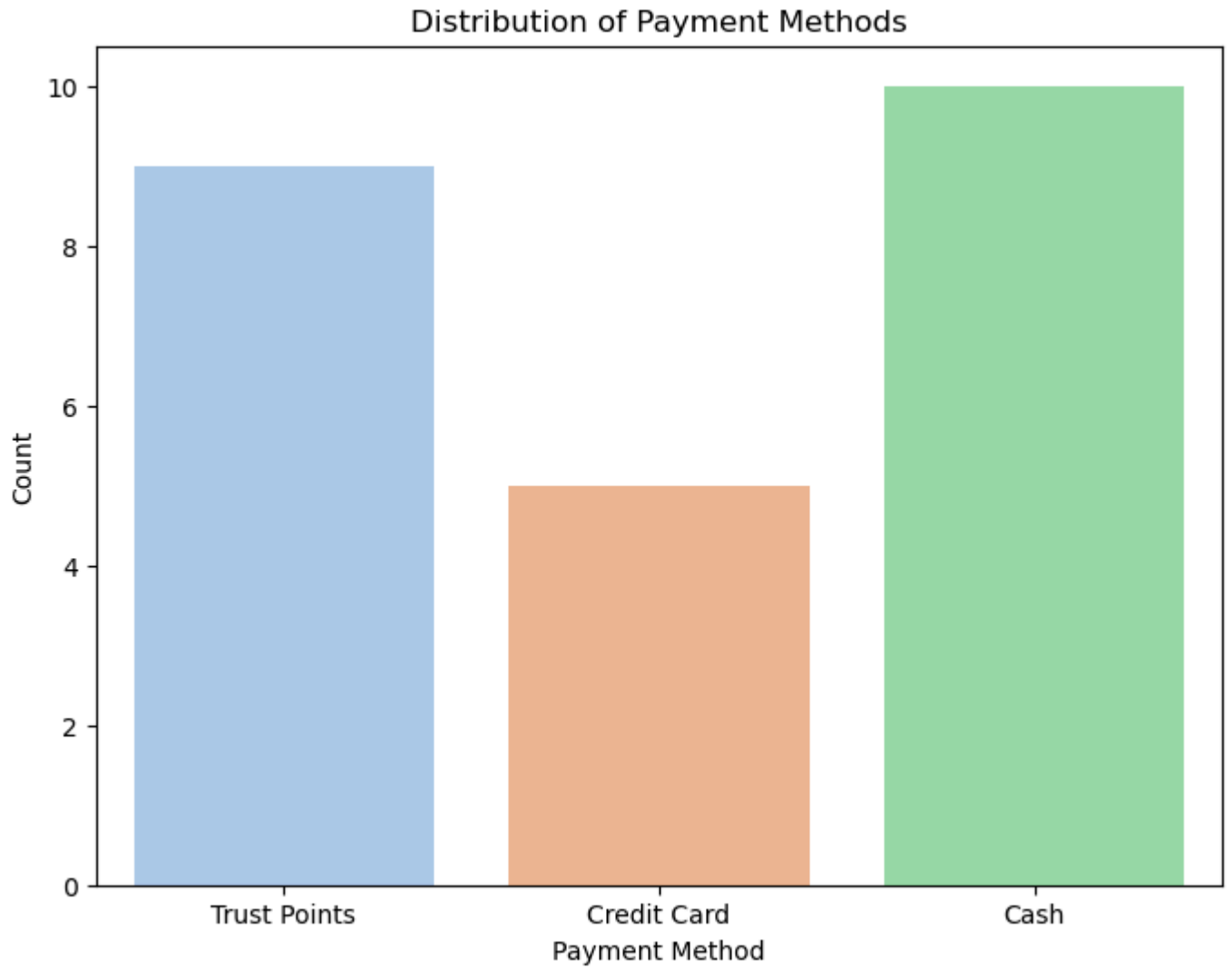
In [17]:
```
plt.figure(figsize=(8, 6))
sns.countplot(x='ProductCategory', data=df, palette='viridis')
plt.title('Distribution of Product Categories')
plt.xlabel('Product Category')
plt.ylabel('Count')
plt.show()
```



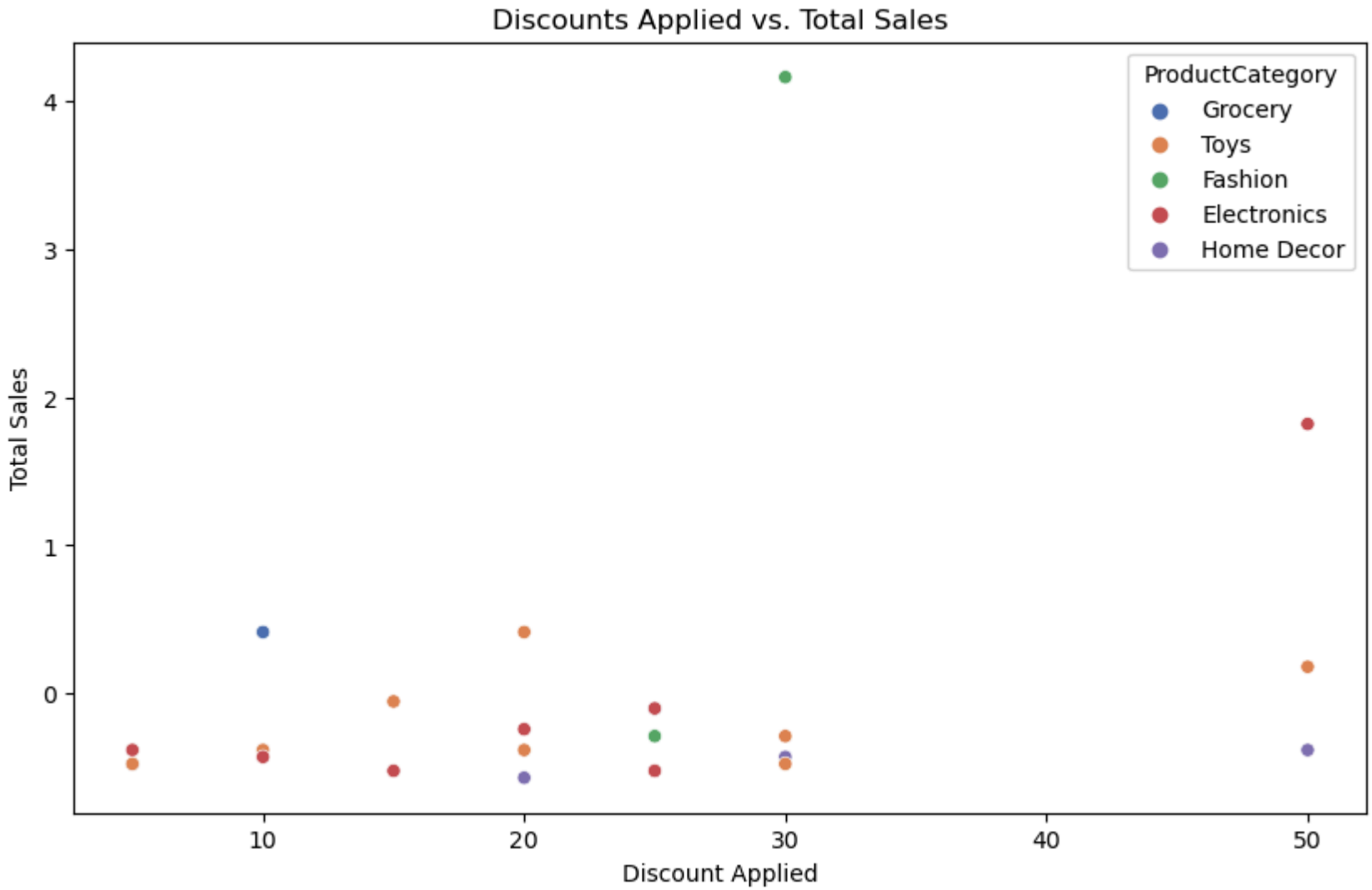
In [18]:
```
sales_by_category = df.groupby('ProductCategory')['TotalAmount'].sum().reset_index()
plt.figure(figsize=(8, 6))
sns.barplot(x='ProductCategory', y='TotalAmount', data=sales_by_category, palette='coolwarm')
plt.title('Total Sales by Product Category')
plt.xlabel('Product Category')
plt.ylabel('Total Sales')
plt.show()
```



In [19]:
```
plt.figure(figsize=(8, 6))
sns.countplot(x='PaymentMethod', data=df, palette='pastel')
plt.title('Distribution of Payment Methods')
plt.xlabel('Payment Method')
plt.ylabel('Count')
plt.show()
```



In [20]:
```
plt.figure(figsize=(10, 6))
sns.scatterplot(x='DiscountApplied', y='TotalAmount', data=df, hue='ProductCategory', palette='deep')
plt.title('Discounts Applied vs. Total Sales')
plt.xlabel('Discount Applied')
plt.ylabel('Total Sales')
plt.show()
```



How would be know if the data is linearly projected?

In [ ]:
```
To determine if data is linearly projected, you can check for a linear relationship between the variables:

-- A scatter plot allows you to visualize the relationship between two continuous variables.
-- Plotting a regression line on the scatter plot can help visualize how well the data points fit a linear model.
```

For all the different combinations of possible projections, what are the suitable graphical representation? (Eg: Line Chart or Bar Graph)

TOTAL SALES VS DATE -- LINE CHART

COUNT VS PRODUCT CATEGORY -- BAR CHART

TOTAL SALES VS PRODUCT CATEGORY -- BAR CHART

COUNT VS PAYMENT METHOD -- BAR CHART

TOTAL SALES VS DISCOUNT APPLIED -- SCATTER PLOT.