Importing Libraries

In [20]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
```

STEP-1: Loading Dataset.

In [21]:
```
df = pd.read_csv(r'C:\Users\aryan\OneDrive\Desktop\DS PROJECTS\MINI PROJECT\BigBasket Products(1).csv')
df
```

Out[21]:

	index	product	category	sub_category	brand	sale_price	market_price	type	rating	descripti
0	1	Garlic Oil - Vegetarian Capsule 500 mg	Beauty & Hygiene	Hair Care	Sri Sri Ayurveda	220.00	220.0	Hair Oil & Serum	4.1	This Prod conta Garlic tha know
1	2	Water Bottle - Orange	Kitchen, Garden & Pets	Storage & Accessories	Mastercook	180.00	180.0	Water & Fridge Bottles	2.3	Ea produc microwa safe (with lid),
2	3	Brass Angle Deep - Plain, No.2	Cleaning & Household	Pooja Needs	Trm	119.00	250.0	Lamp & Lamp Oil	3.4	A perfect for occasio be it yo n
3	4	Cereal Flip Lid Container/Storage Jar - Assort...	Cleaning & Household	Bins & Bathroom Ware	Nakoda	149.00	176.0	Laundry, Storage Baskets	3.7	Multipurpo contai with attract des
4	5	Creme Soft Soap - For Hands & Body	Beauty & Hygiene	Bath & Hand Wash	Nivea	162.00	162.0	Bathing Bars & Soaps	4.4	Niv Creme S Soap giv your skin bes
...	...	...	...	...	...	...	...	...	...	
27550	27551	Wottagirl! Perfume Spray - Heaven, Classic	Beauty & Hygiene	Fragrances & Deos	Layerr	199.20	249.0	Perfume	3.9	Layerr brin y Wotta Clas fragrant b
27551	27552	Rosemary	Gourmet & World Food	Cooking & Baking Needs	Puramate	67.50	75.0	Herbs, Seasonings & Rubs	4.0	Puram rosemary enough transforn di
27552	27553	Peri-Peri Sweet Potato Chips	Gourmet & World Food	Snacks, Dry Fruits, Nuts	FabBox	200.00	200.0	Nachos & Chips	3.8	We ha taken richness Sw Potatoes
27553	27554	Green Tea - Pure Original	Beverages	Tea	Tetley	396.00	495.0	Tea Bags	4.2	Tetley Gre Tea with refresh pure, or
27554	27555	United Dreams Go Far Deodorant	Beauty & Hygiene	Men's Grooming	United Colors Of Benetton	214.53	390.0	Men's Deodorants	4.5	The n m fragran from Unit Dreams

27555 rows × 10 columns

Step 2: Use head function to look for first 12 rows.

In [22]:
df.head(12)

Out[22]:

	index	product	category	sub_category	brand	sale_price	market_price	type	rating	description
0	1	Garlic Oil - Vegetarian Capsule 500 mg	Beauty & Hygiene	Hair Care	Sri Sri Ayurveda	220.0	220.0	Hair Oil & Serum	4.1	This Produc contain Garlic O that i known..
1	2	Water Bottle - Orange	Kitchen, Garden & Pets	Storage & Accessories	Mastercook	180.0	180.0	Water & Fridge Bottles	2.3	Each produc is microwav safe (withou lid), ..
2	3	Brass Angle Deep - Plain, No.2	Cleaning & Household	Pooja Needs	Trm	119.0	250.0	Lamp & Lamp Oil	3.4	A perfect gi for a occasions, b it your m..
3	4	Cereal Flip Lid Container/Storage Jar - Assort...	Cleaning & Household	Bins & Bathroom Ware	Nakoda	149.0	176.0	Laundry, Storage Baskets	3.7	Multipurpos containe with a attractiv desi..
4	5	Creme Soft Soap - For Hands & Body	Beauty & Hygiene	Bath & Hand Wash	Nivea	162.0	162.0	Bathing Bars & Soaps	4.4	Nivea Crem Soft Soa gives you skin th best..
5	6	Germ - Removal Multipurpose Wipes	Cleaning & Household	All Purpose Cleaners	Nature Protect	169.0	199.0	Disinfectant Spray & Cleaners	3.3	Sta protecte fro contaminatio wit Multipu..
6	7	Multani Mati	Beauty & Hygiene	Skin Care	Satinance	58.0	58.0	Face Care	3.6	Satinanc multani mat is an excelle skin t..
7	8	Hand Sanitizer - 70% Alcohol Base	Beauty & Hygiene	Bath & Hand Wash	Bionova	250.0	250.0	Hand Wash & Sanitizers	4.0	70%Alcoh based i gentle o hand leave skin..
8	9	Biotin & Collagen Volumizing Hair Shampoo + Bi..	Beauty & Hygiene	Hair Care	StBotanica	1098.0	1098.0	Shampoo & Conditioner	3.5	An exclusiv blend wit Vitamin B Biotin, Hyd..
9	10	Scrub Pad - Anti-Bacterial, Regular	Cleaning & Household	Mops, Brushes & Scrubs	Scotch brite	20.0	20.0	Utensil Scrub-Pad, Glove	4.3	Scotch Brit Anti Bacteri Scrub Pa thoroug..
10	11	Wheat Grass Powder - Raw	Gourmet & World Food	Cooking & Baking Needs	NUTRASHIL	261.0	290.0	Flours & Pre-Mixes	4.0	Wheatgras is superfoo potent healt

	index	product	category	sub_category	brand	sale_price	market_price	type	rating	description
11	12	Butter Cookies Gold Collection	Gourmet & World Food	Chocolates & Biscuits	Sapphire	600.0	600.0	Luxury Chocolates, Gifts	2.2	Enjoy a ti full o deliciou butte cookies m..

Step 3: Get Description of the data in the DataFrame

In [23]:
df.describe()

Out[23]:

	index	sale_price	market_price	rating
count	27555.00000	27549.000000	27555.000000	18919.000000
mean	13778.00000	334.648391	382.056664	3.943295
std	7954.58767	1202.102113	581.730717	0.739217
min	1.00000	2.450000	3.000000	1.000000
25%	6889.50000	95.000000	100.000000	3.700000
50%	13778.00000	190.320000	220.000000	4.100000
75%	20666.50000	359.000000	425.000000	4.300000
max	27555.00000	112475.000000	12500.000000	5.000000

Step 4: Find Information about the DataFrame.

In [27]:
df.info()

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 27555 entries, 0 to 27554
Data columns (total 10 columns):
 #   Column       Non-Null Count  Dtype  
---  ------       --------------  -----  
 0   index        27555 non-null  int64  
 1   product      27554 non-null  object 
 2   category     27555 non-null  object 
 3   sub_category 27555 non-null  object 
 4   brand        27554 non-null  object 
 5   sale_price   27549 non-null  float64
 6   market_price 27555 non-null  float64
 7   type         27555 non-null  object 
 8   rating       18919 non-null  float64
 9   description  27440 non-null  object 
dtypes: float64(3), int64(1), object(6)
memory usage: 2.1+ MB
```

Step 5: Find out Top & least sold products.

In [38]:

#Sort by rating to find Top & Least sold

```
top_sold = df.sort_values(by = 'rating', ascending = False).head(10)
least_sold = df.sort_values(by = 'rating', ascending =True).head(10)
```

 # Display the results

```
print(" Top 10 Sold (Most Popular) Products:\n")
print(top_sold[['product', 'brand', 'category', 'rating', 'sale_price']])

print("\n Least 10 Sold (Least Popular) Products:\n")
print(least_sold[['product', 'brand', 'category', 'rating', 'sale_price']])
```

```
 Top 10 Sold (Most Popular) Products:

                                  product          brand  \
12416  Single Line Soft Bound Long Book Feminine Seri...   Navneet Youva
12413                          Squash - Kokum Kadi      NaturoBell
27502                          Forest Honey - Wild   Organic Nation
20437  MaxFresh Blue Spicy Fresh Gel Toothpaste - Mum...        Colgate
5985                         Tea Tree Essential Oil    Kama Ayurveda
27512          Water Bottle - Fridge, Tulip, Dark Blue          Cello
20315            Cleaner - Stainless Steel & Aluminum          Big D
20327                Smooth Skin Oil - For Dry Skin  Aroma Treasures
27508                  Palm Jaggery/Bella Crystals          Draft
27507                       Extra Crisp Sweet Corn          Daucy

                category  rating  sale_price
12416    Cleaning & Household     5.0      53.00
12413               Beverages     5.0     110.00
27502   Snacks & Branded Foods     5.0     350.00
20437        Beauty & Hygiene     5.0      89.25
5985         Beauty & Hygiene     5.0     750.00
27512   Kitchen, Garden & Pets     5.0     109.00
20315    Cleaning & Household     5.0     249.00
20327        Beauty & Hygiene     5.0     324.00
27508  Foodgrains, Oil & Masala     5.0     199.00
27507    Gourmet & World Food     5.0     202.50

 Least 10 Sold (Least Popular) Products:

                                  product          brand  \
11219                              Argan Oil       Aloe Veda
11175                  Vaginal Tightening Gel        Everteen
14265            Ramona - Eau De Parfum, For Her         Maryaj
16863                      Piping Gel, Neutral        Foodecor
4657          Black Gram Lentils Curry-Basmati Rice    Fazlani Foods
10557                      Jasmine Potpourri       Soulflower
24261  Hibiscus & Red Melon Green Tea - 2 In 1 For Ho...           Care
15172          Blue Incense Sticks - Economy Pack         Liberty
14251                      Love & Joy EDP  All Good Scents
1970                OH! Pleasure Gel for Women           Skore

                category  rating  sale_price
11219        Beauty & Hygiene     1.0     1755.0
11175        Beauty & Hygiene     1.0     1999.0
14265        Beauty & Hygiene     1.0      834.5
16863    Gourmet & World Food     1.0      104.5
4657   Snacks & Branded Foods     1.0      150.0
10557    Cleaning & Household     1.0      250.0
24261    Gourmet & World Food     1.0      250.0
15172    Cleaning & Household     1.0       54.0
14251        Beauty & Hygiene     1.0     1200.0
1970         Beauty & Hygiene     1.0      500.0
```

Step 6: Measuring discount on a certain item.

In [40]:

 # Create a new column for discount percentage

```
df['discount_percent'] = ((df['market_price'] - df['sale_price']) / df['market_price']) * 100
```

 # Show a sample of the discount values

```
print(df[['product', 'market_price', 'sale_price', 'discount_percent']].head(10))
```

```
                                product  market_price  \
0          Garlic Oil - Vegetarian Capsule 500 mg         220.0
1                       Water Bottle - Orange         180.0
2                 Brass Angle Deep - Plain, No.2         250.0
3  Cereal Flip Lid Container/Storage Jar - Assort...         176.0
4               Creme Soft Soap - For Hands & Body         162.0
5               Germ - Removal Multipurpose Wipes         199.0
6                                 Multani Mati          58.0
7               Hand Sanitizer - 70% Alcohol Base         250.0
8  Biotin & Collagen Volumizing Hair Shampoo + Bi...        1098.0
9             Scrub Pad - Anti- Bacterial, Regular          20.0

   sale_price  discount_percent
0       220.0         0.000000
1       180.0         0.000000
2       119.0        52.400000
3       149.0        15.340909
4       162.0         0.000000
5       169.0        15.075377
6        58.0         0.000000
7       250.0         0.000000
8      1098.0         0.000000
9        20.0         0.000000
```

In [41]:

```
item = "Tata Salt"   # ← type any product name here

item_discount = df[df['product'].str.contains(item, case=False, na=False)][
    ['product', 'market_price', 'sale_price', 'discount_percent']
]

print(item_discount)
```

```
Empty DataFrame
Columns: [product, market_price, sale_price, discount_percent]
Index: []
```

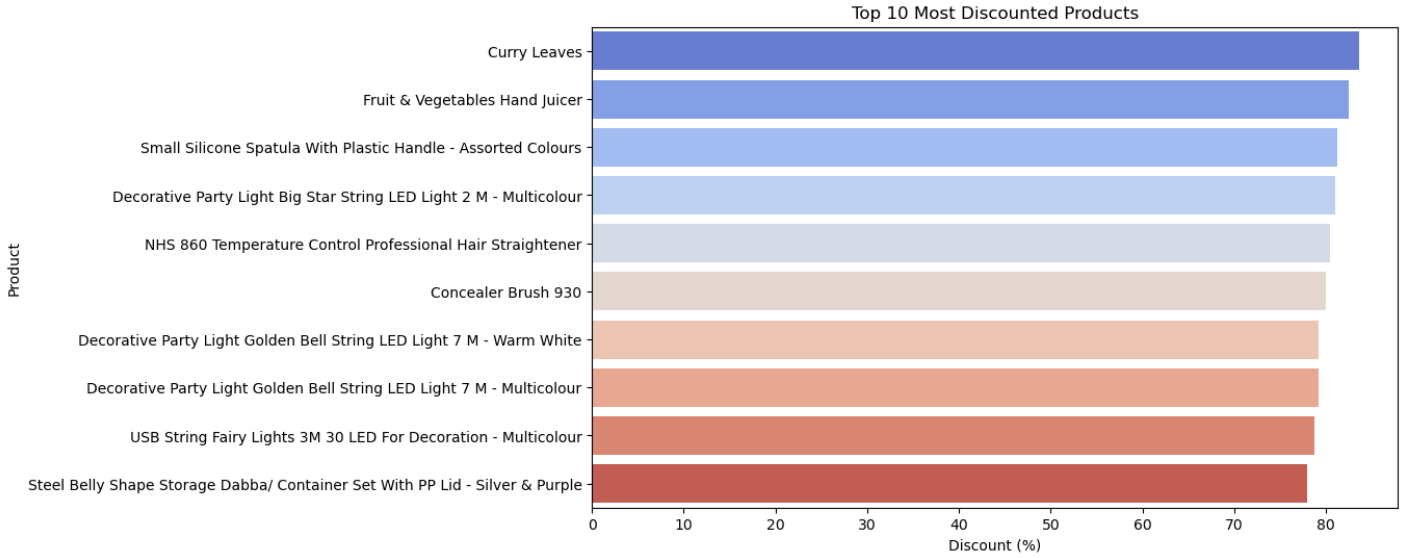
In [42]:

```
df['discount_percent'] = df['discount_percent'].round(2)
```

In [44]:

```
top_discount = df.sort_values(by='discount_percent', ascending=False).head(10)

plt.figure(figsize=(10,6))
sns.barplot(x='discount_percent', y='product', data=top_discount, palette='coolwarm')
plt.title('Top 10 Most Discounted Products')
plt.xlabel('Discount (%)')
plt.ylabel('Product')
plt.show()
```

```
C:\Users\aryan\AppData\Local\Temp\ipykernel_14412\1491855841.py:4: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in v0.14.0. Assign the `y` variable to `hue` and set `legend=False` for the same effect.

  sns.barplot(x='discount_percent', y='product', data=top_discount, palette='coolwarm')
```



Step 7: Find out the Missing Values from the Dataset.

In [45]:

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

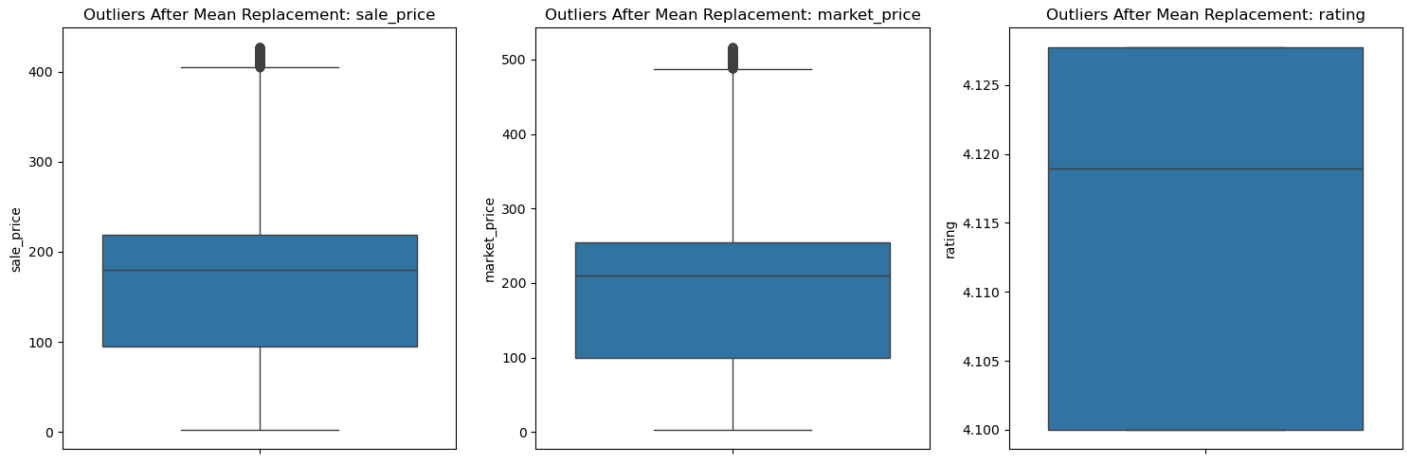
Out[45]:
```
index              0
product            1
category           0
sub_category       0
brand              1
sale_price         6
market_price       0
type               0
rating          8636
description      115
discount_percent   6
dtype: int64
```

Step 8: Find out the outliers from the dataset according to the columns and fill them with the mean.

In [57]:
```
# Choose numeric columns to check
numeric_cols = ['sale_price', 'market_price', 'rating']

# Detect and replace outliers with mean
for col in numeric_cols:
    Q1 = df[col].quantile(0.25)
    Q3 = df[col].quantile(0.75)
    IQR = Q3 - Q1

    # Define bounds
    lower_limit = Q1 - 1.5 * IQR
    upper_limit = Q3 + 1.5 * IQR

    # Calculate mean of the column
    mean_value = df[col].mean()

    # Replace outliers with mean
    df.loc[(df[col] < lower_limit) | (df[col] > upper_limit), col] = mean_value

print("Outliers have been replaced with mean successfully!")

# Step 4: Check updated data
print(df[numeric_cols].describe())
```

```
Outliers have been replaced with mean successfully!
       sale_price  market_price       rating
count  27549.000000  27555.000000  18919.000000
mean     172.957177    203.250192      4.113813
std       96.870779    118.958001      0.012436
min        2.450000      3.000000      4.100000
25%       95.000000    100.000000      4.100000
50%      180.000000    210.000000      4.118949
75%      219.000000    255.000000      4.127758
max      427.500000    517.000000      4.127758
```

In [58]:
```
import seaborn as sns
import matplotlib.pyplot as plt

plt.figure(figsize=(15,5))
for i, col in enumerate(numeric_cols, 1):
    plt.subplot(1, 3, i)
    sns.boxplot(y=df[col])
    plt.title(f'Outliers After Mean Replacement: {col}')

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



Step 9: Create Plots or visualizations.

Countplot – Category Distribution

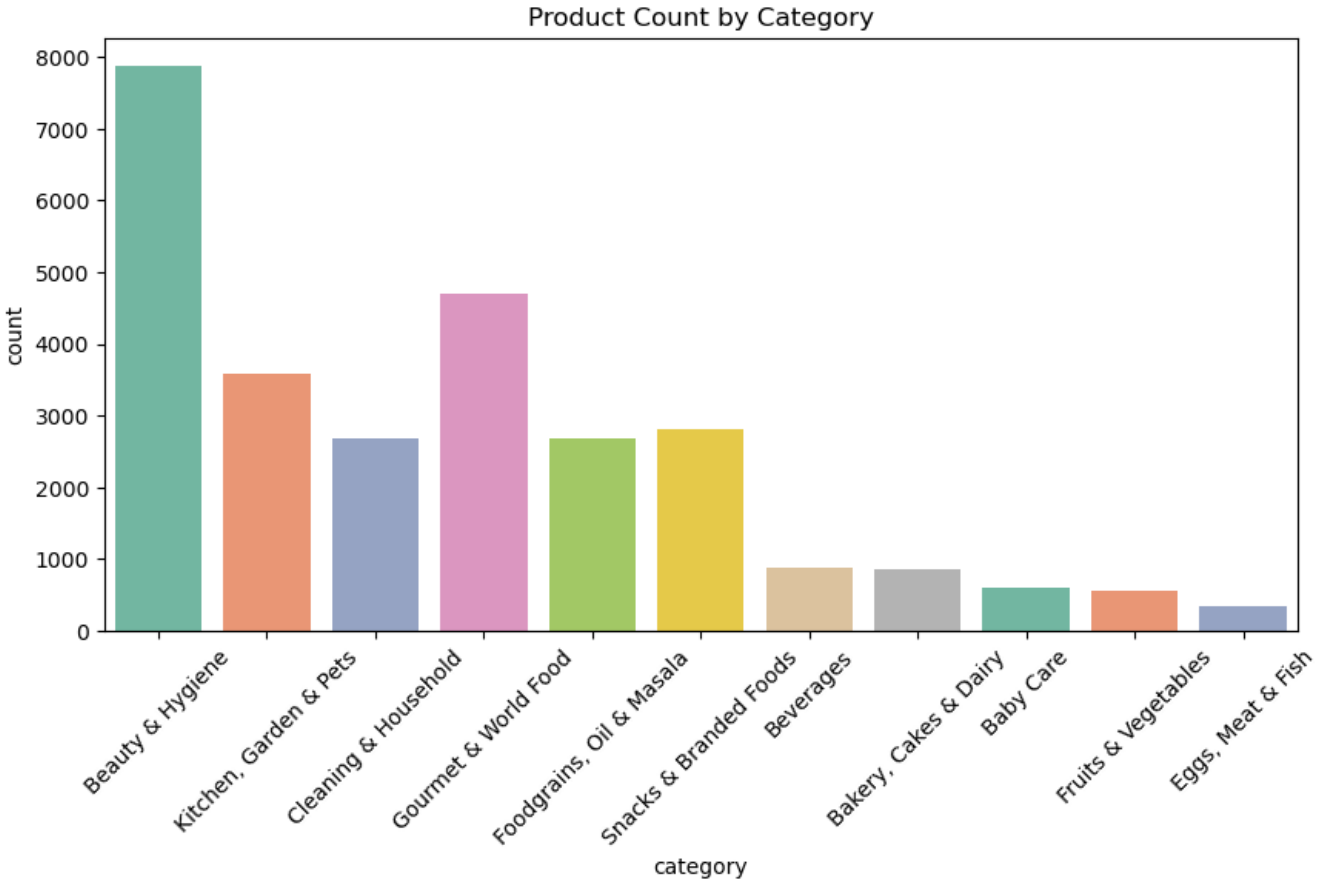
In [59]:

```
plt.figure(figsize=(10,5))
sns.countplot(x='category', data=df, palette='Set2')
plt.title('Product Count by Category')
plt.xticks(rotation=45)
plt.show()
```

C:\Users\aryan\AppData\Local\Temp\ipykernel_14412\497824856.py:2: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in v0.14.0. Assign the `x` variable to `hue` and set `legend=False` for the same effect.

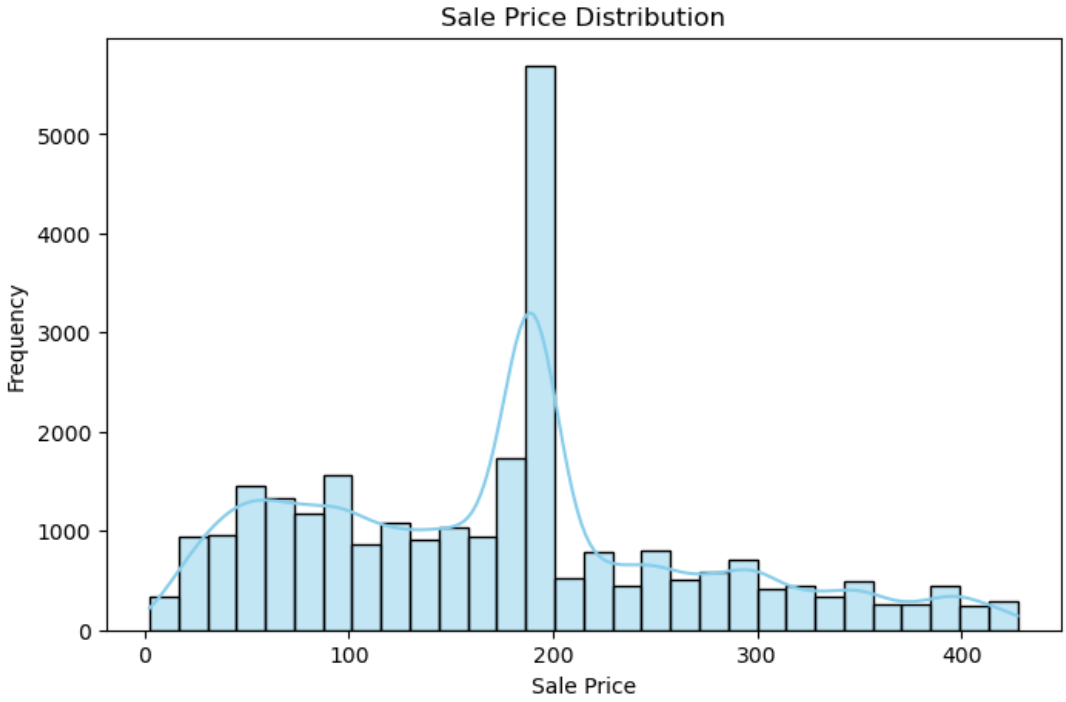
sns.countplot(x='category', data=df, palette='Set2')



Histogram – Price Distribution

In [61]:

```
plt.figure(figsize=(8,5))
sns.histplot(df['sale_price'], bins=30, kde=True, color='skyblue')
plt.title('Sale Price Distribution')
plt.xlabel('Sale Price')
plt.ylabel('Frequency')
plt.show()
```



Boxplot – Category vs Sale Price

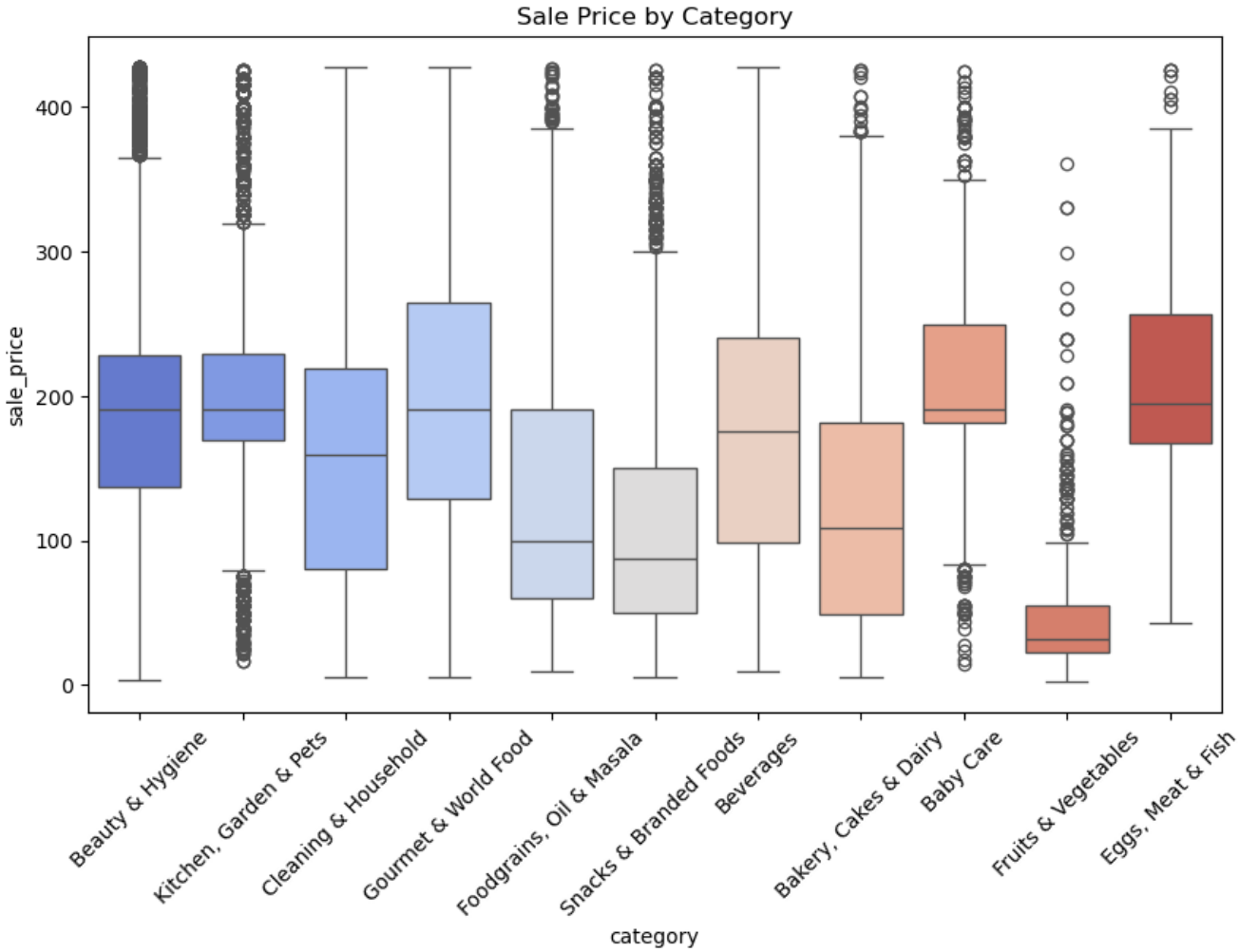
In [62]:

```
plt.figure(figsize=(10,6))
sns.boxplot(x='category', y='sale_price', data=df, palette='coolwarm')
plt.title('Sale Price by Category')
plt.xticks(rotation=45)
plt.show()
```

C:\Users\aryan\AppData\Local\Temp\ipykernel_14412\1584097190.py:2: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in v0.14.0. Assign the `x` variable to `hue` and set `legend=False` for the same effect.

```
sns.boxplot(x='category', y='sale_price', data=df, palette='coolwarm')
```



Barplot – Average Price by Brand

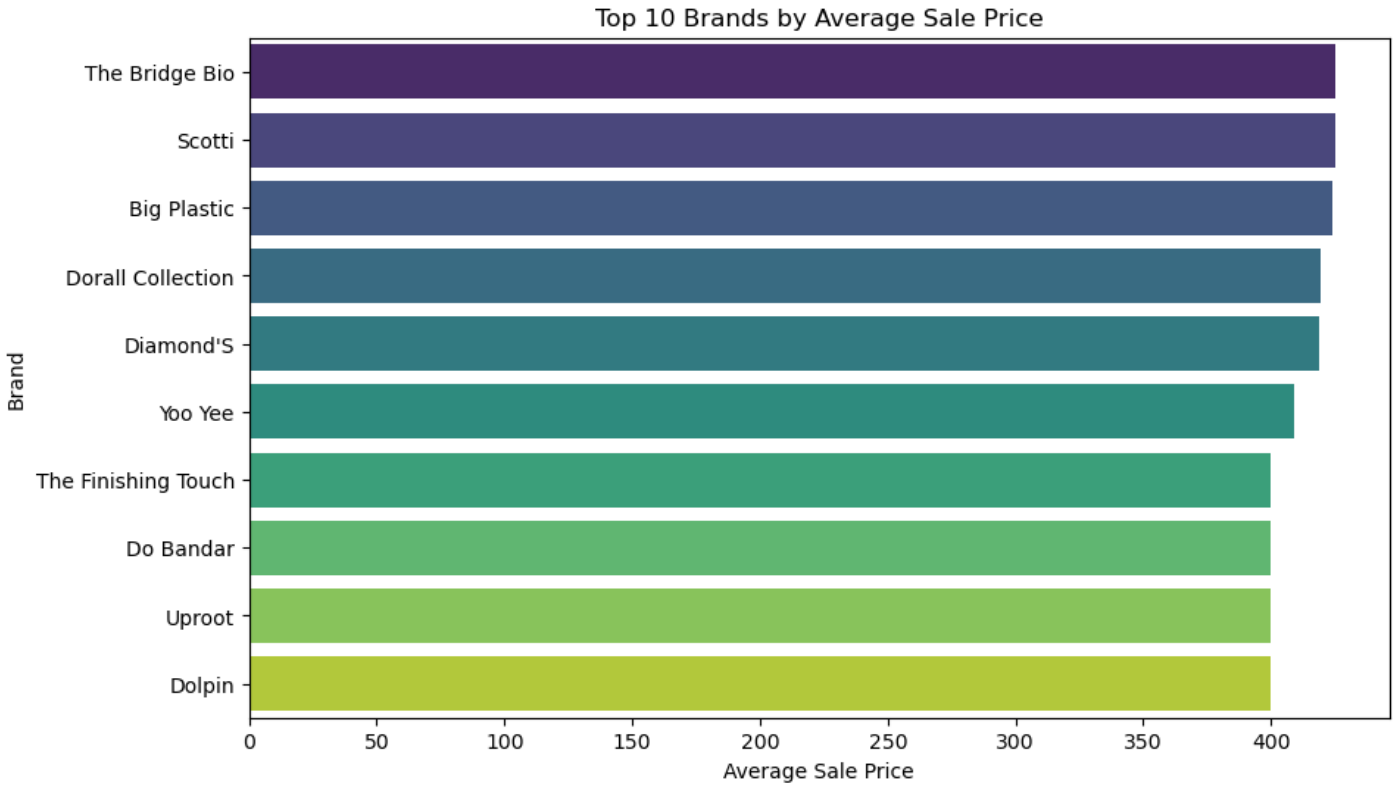
In [63]:

```
plt.figure(figsize=(10,6))
avg_price = df.groupby('brand')['sale_price'].mean().sort_values(ascending=False).head(10)
sns.barplot(x=avg_price.values, y=avg_price.index, palette='viridis')
plt.title('Top 10 Brands by Average Sale Price')
plt.xlabel('Average Sale Price')
plt.ylabel('Brand')
plt.show()
```

C:\Users\aryan\AppData\Local\Temp\ipykernel_14412\2920604037.py:3: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in v0.14.0. Assign the `y` variable to `hue` and set `legend=False` for the same effect.

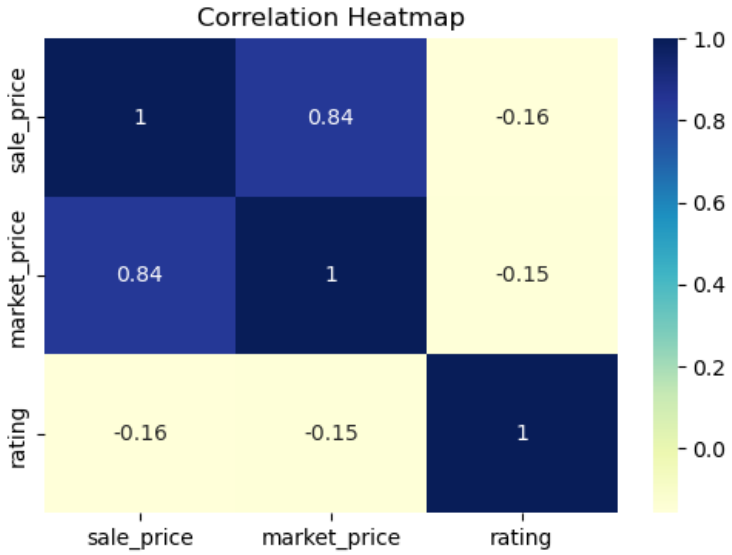
```
sns.barplot(x=avg_price.values, y=avg_price.index, palette='viridis')
```



Heatmap – Correlation Between Numeric Columns

In [64]:

```
plt.figure(figsize=(6,4))
sns.heatmap(df[['sale_price', 'market_price', 'rating']].corr(), annot=True, cmap='YlGnBu')
plt.title('Correlation Heatmap')
plt.show()
```



Scatter Plot – Sale Price vs Rating

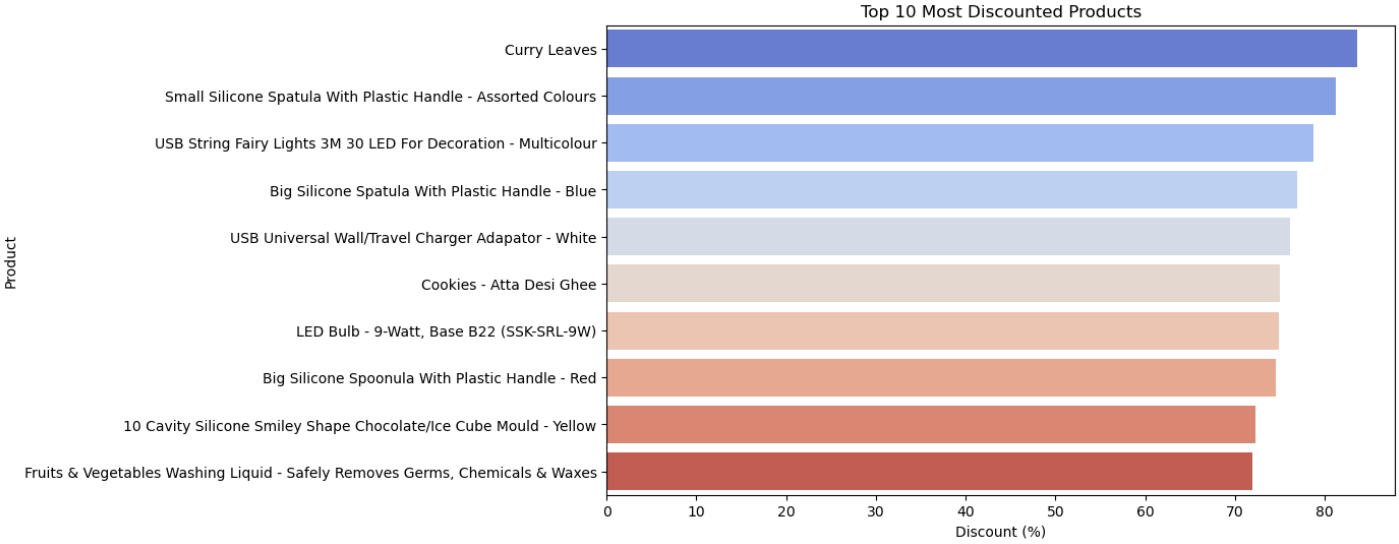
In [65]:

```
plt.figure(figsize=(8,5))
sns.scatterplot(x='sale_price', y='rating', data=df, color='purple')
plt.title('Sale Price vs Rating')
plt.show()
```

Discount Plot – Top 10 Most Discounted Products

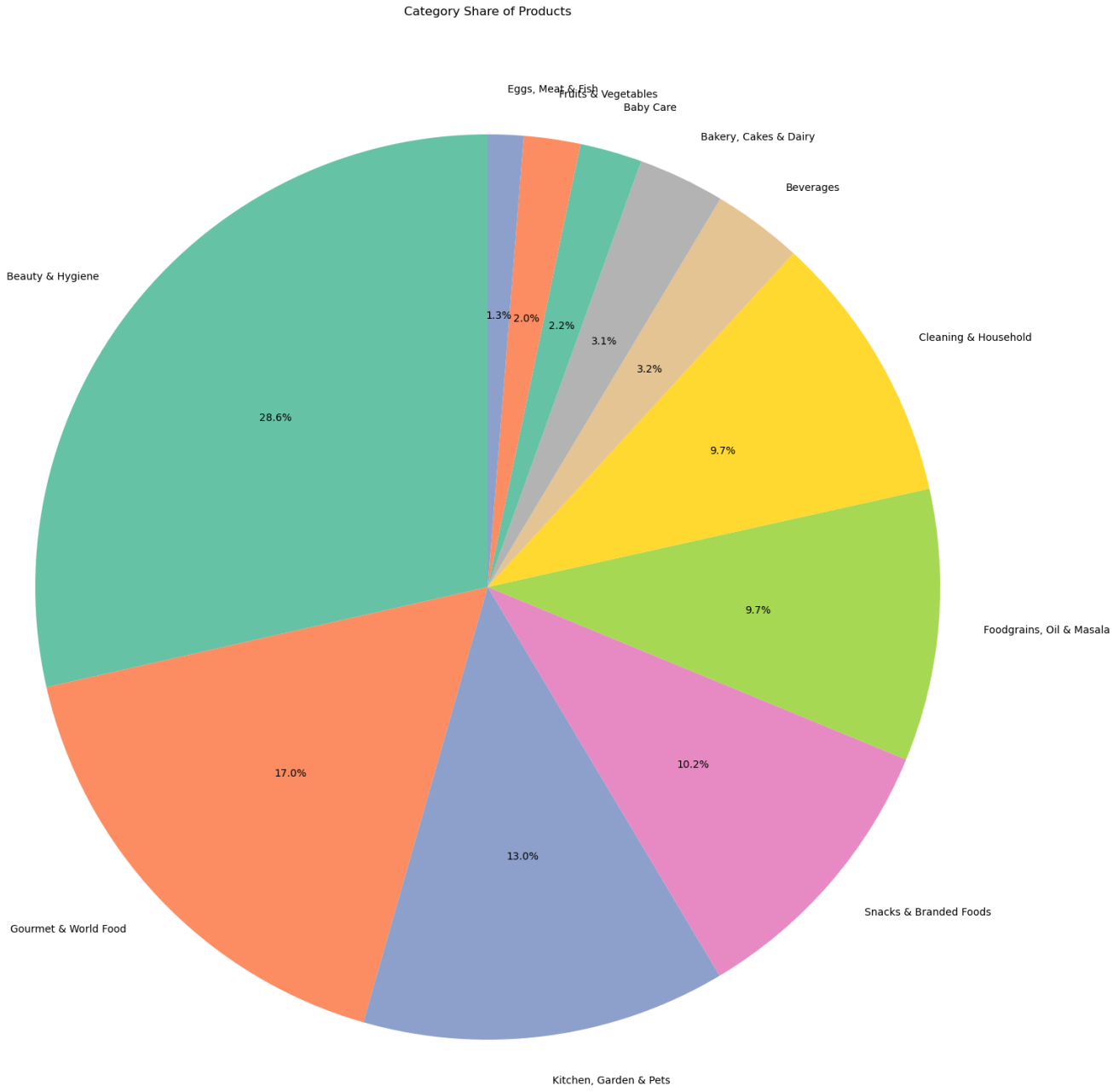
In [66]:
```
df['discount_percent'] = ((df['market_price'] - df['sale_price']) / df['market_price']) * 100
top_discount = df.sort_values(by='discount_percent', ascending=False).head(10)

plt.figure(figsize=(10,6))
sns.barplot(x='discount_percent', y='product', data=top_discount, palette='coolwarm')
plt.title('Top 10 Most Discounted Products')
plt.xlabel('Discount (%)')
plt.ylabel('Product')
plt.show()
```

```
C:\Users\aryan\AppData\Local\Temp\ipykernel_14412\3573293495.py:5: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in v0.14.0. Assign the `y` variable to `hue` and set `legend=False` for the same effect.

  sns.barplot(x='discount_percent', y='product', data=top_discount, palette='coolwarm')
```



Pie Chart – Share of Categories

In [71]:
```
category_counts = df['category'].value_counts()
plt.figure(figsize=(20,20))
plt.pie(category_counts, labels=category_counts.index, autopct='%1.1f%%', startangle=90, colors=sns.color_palette('Set2'))
plt.title('Category Share of Products')
plt.show()
```



Countplot – Top 10 Brands by Number of Products

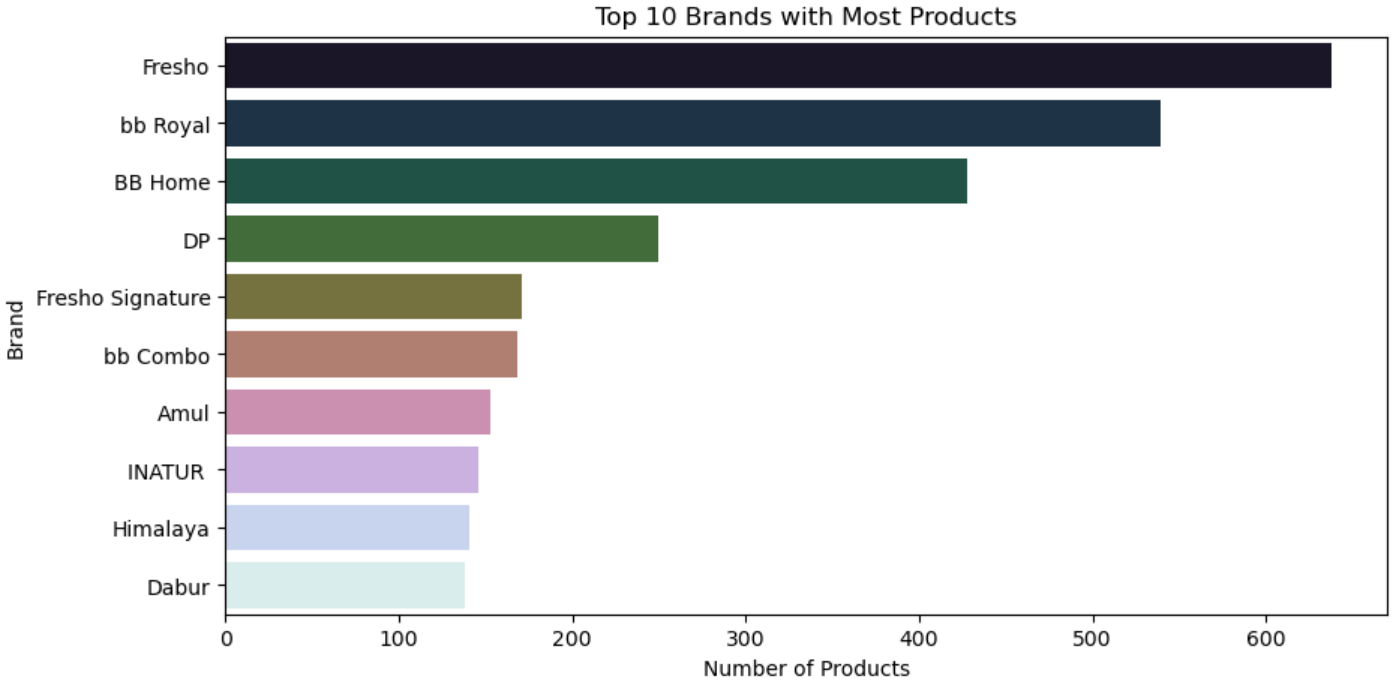
In [72]:

```
top_brands = df['brand'].value_counts().head(10)
plt.figure(figsize=(10,5))
sns.barplot(x=top_brands.values, y=top_brands.index, palette='cubehelix')
plt.title('Top 10 Brands with Most Products')
plt.xlabel('Number of Products')
plt.ylabel('Brand')
plt.show()
```

C:\Users\aryan\AppData\Local\Temp\ipykernel_14412\4112046618.py:3: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in v0.14.0. Assign the `y` variable to `hue` and set `legend=False` for the same effect.

```
sns.barplot(x=top_brands.values, y=top_brands.index, palette='cubehelix')
```



Violin Plot – Price Distribution by Category

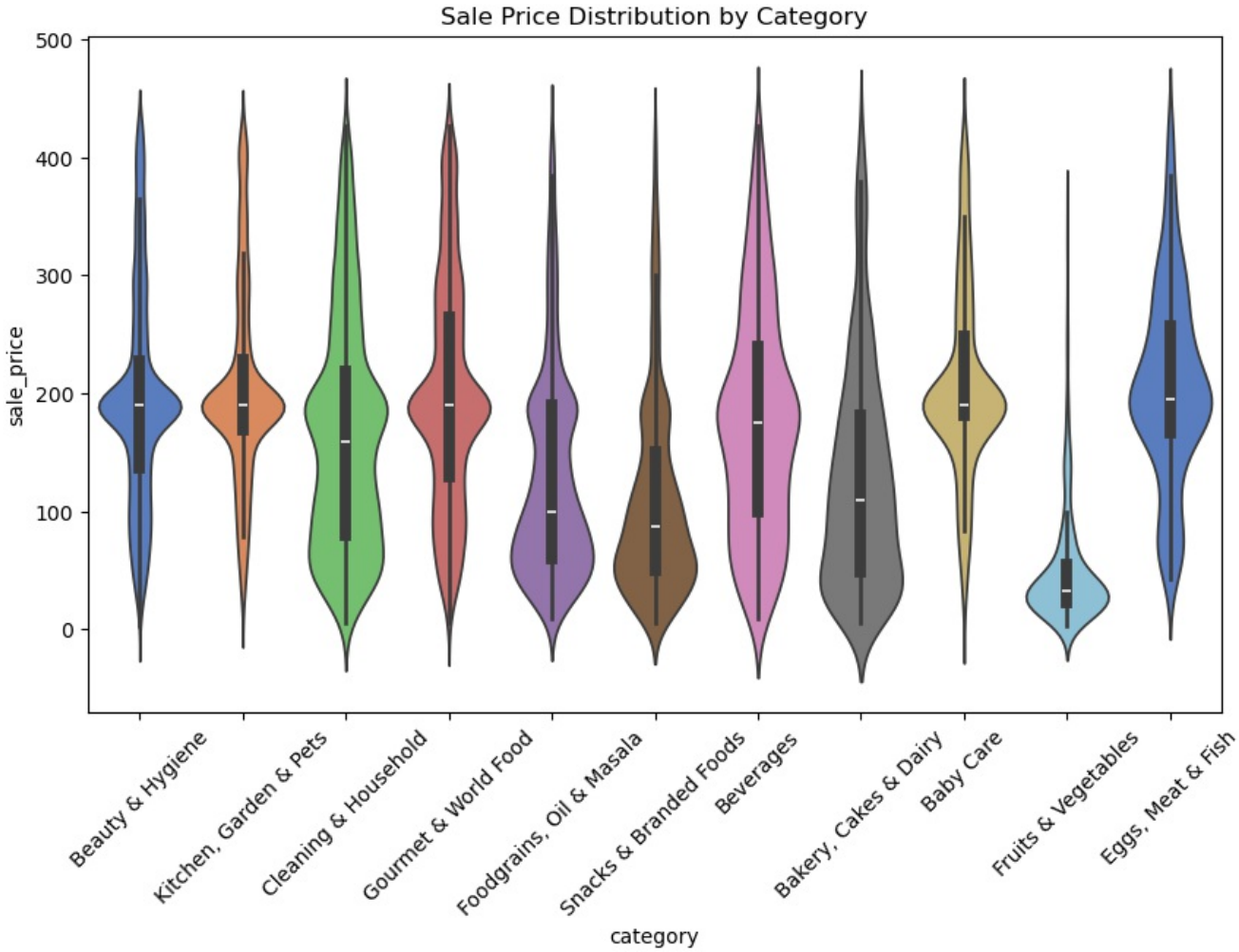
In [73]:

```
plt.figure(figsize=(10,6))
sns.violinplot(x='category', y='sale_price', data=df, palette='muted')
plt.title('Sale Price Distribution by Category')
plt.xticks(rotation=45)
plt.show()
```

C:\Users\aryan\AppData\Local\Temp\ipykernel_14412\3527681912.py:2: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in v0.14.0. Assign the `x` variable to `hue` and set `legend=False` for the same effect.

```
sns.violinplot(x='category', y='sale_price', data=df, palette='muted')
```

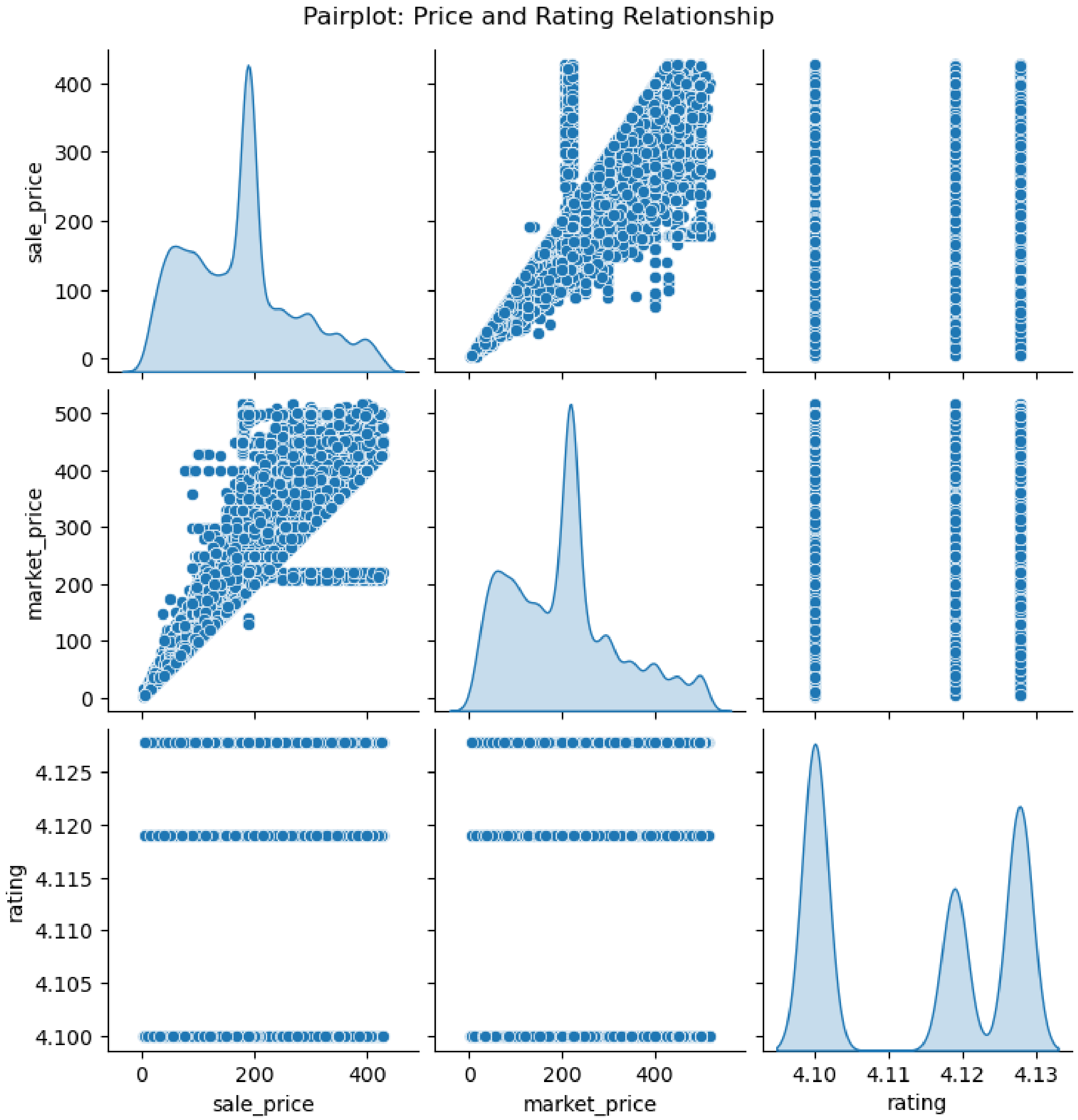


Discount vs Rating (Scatter Plot)

In [74]:

```
sns.pairplot(df[['sale_price', 'market_price', 'rating']], diag_kind='kde', palette='coolwarm')
plt.suptitle('Pairplot: Price and Rating Relationship', y=1.02)
plt.show()
```

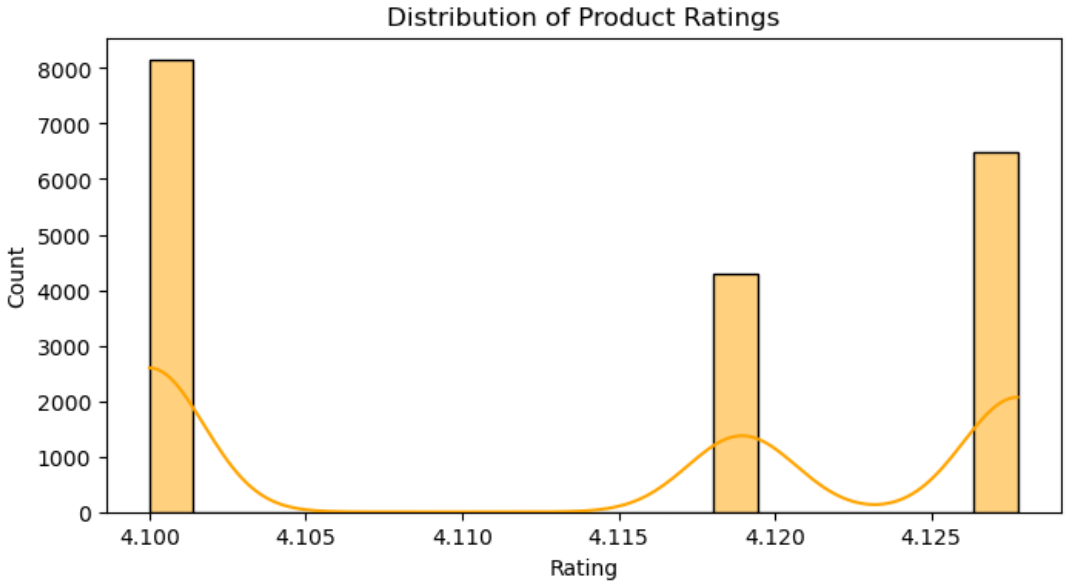
C:\Users\aryan\anaconda3\Lib\site-packages\seaborn\axisgrid.py:1513: UserWarning: Ignoring `palette` because no `hue` variable has been assigned.
  func(x=vector, **plot_kwargs)
C:\Users\aryan\anaconda3\Lib\site-packages\seaborn\axisgrid.py:1513: UserWarning: Ignoring `palette` because no `hue` variable has been assigned.
  func(x=vector, **plot_kwargs)
C:\Users\aryan\anaconda3\Lib\site-packages\seaborn\axisgrid.py:1513: UserWarning: Ignoring `palette` because no `hue` variable has been assigned.
  func(x=vector, **plot_kwargs)
C:\Users\aryan\anaconda3\Lib\site-packages\seaborn\axisgrid.py:1615: UserWarning: Ignoring `palette` because no `hue` variable has been assigned.
  func(x=x, y=y, **kwargs)
C:\Users\aryan\anaconda3\Lib\site-packages\seaborn\axisgrid.py:1615: UserWarning: Ignoring `palette` because no `hue` variable has been assigned.
  func(x=x, y=y, **kwargs)
C:\Users\aryan\anaconda3\Lib\site-packages\seaborn\axisgrid.py:1615: UserWarning: Ignoring `palette` because no `hue` variable has been assigned.
  func(x=x, y=y, **kwargs)
C:\Users\aryan\anaconda3\Lib\site-packages\seaborn\axisgrid.py:1615: UserWarning: Ignoring `palette` because no `hue` variable has been assigned.
  func(x=x, y=y, **kwargs)
C:\Users\aryan\anaconda3\Lib\site-packages\seaborn\axisgrid.py:1615: UserWarning: Ignoring `palette` because no `hue` variable has been assigned.
  func(x=x, y=y, **kwargs)
C:\Users\aryan\anaconda3\Lib\site-packages\seaborn\axisgrid.py:1615: UserWarning: Ignoring `palette` because no `hue` variable has been assigned.
  func(x=x, y=y, **kwargs)



Distribution Plot – Ratings

In [75]:

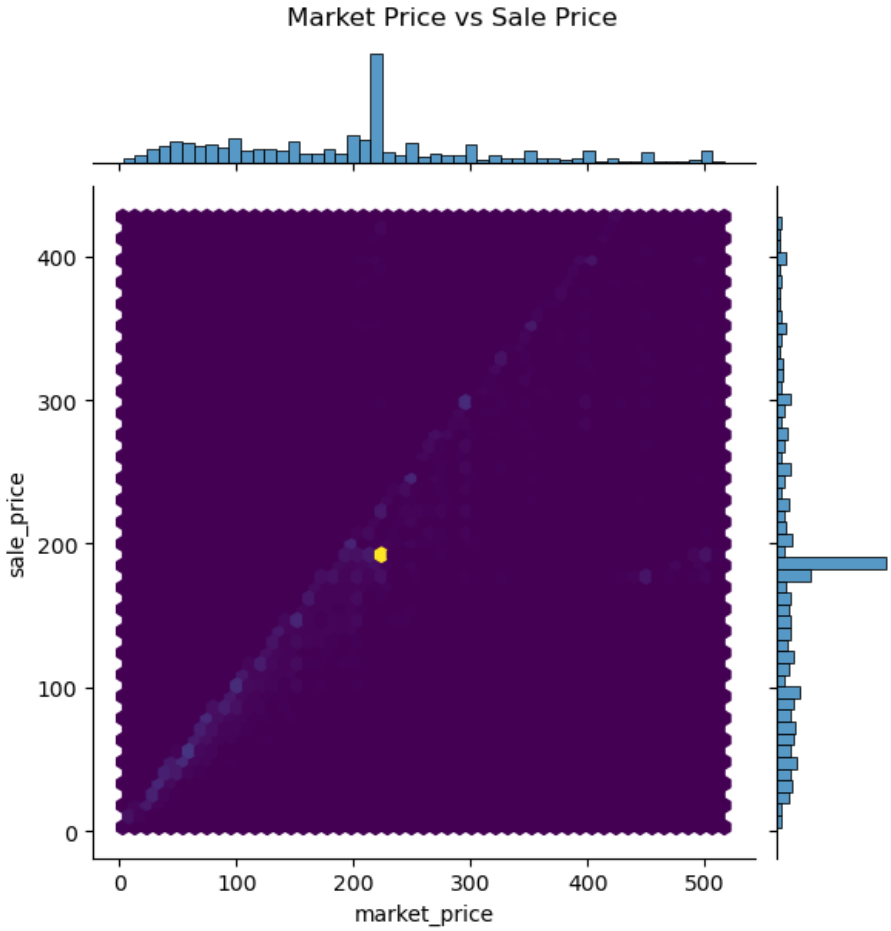
```
plt.figure(figsize=(8,4))
sns.histplot(df['rating'], bins=20, kde=True, color='orange')
plt.title('Distribution of Product Ratings')
plt.xlabel('Rating')
plt.ylabel('Count')
plt.show()
```



Joint Plot – Sale Price vs Market Price

In [76]:

```
sns.jointplot(x='market_price', y='sale_price', data=df, kind='hex', cmap='viridis')
plt.suptitle('Market Price vs Sale Price', y=1.02)
plt.show()
```



Barplot – Average Discount by Category

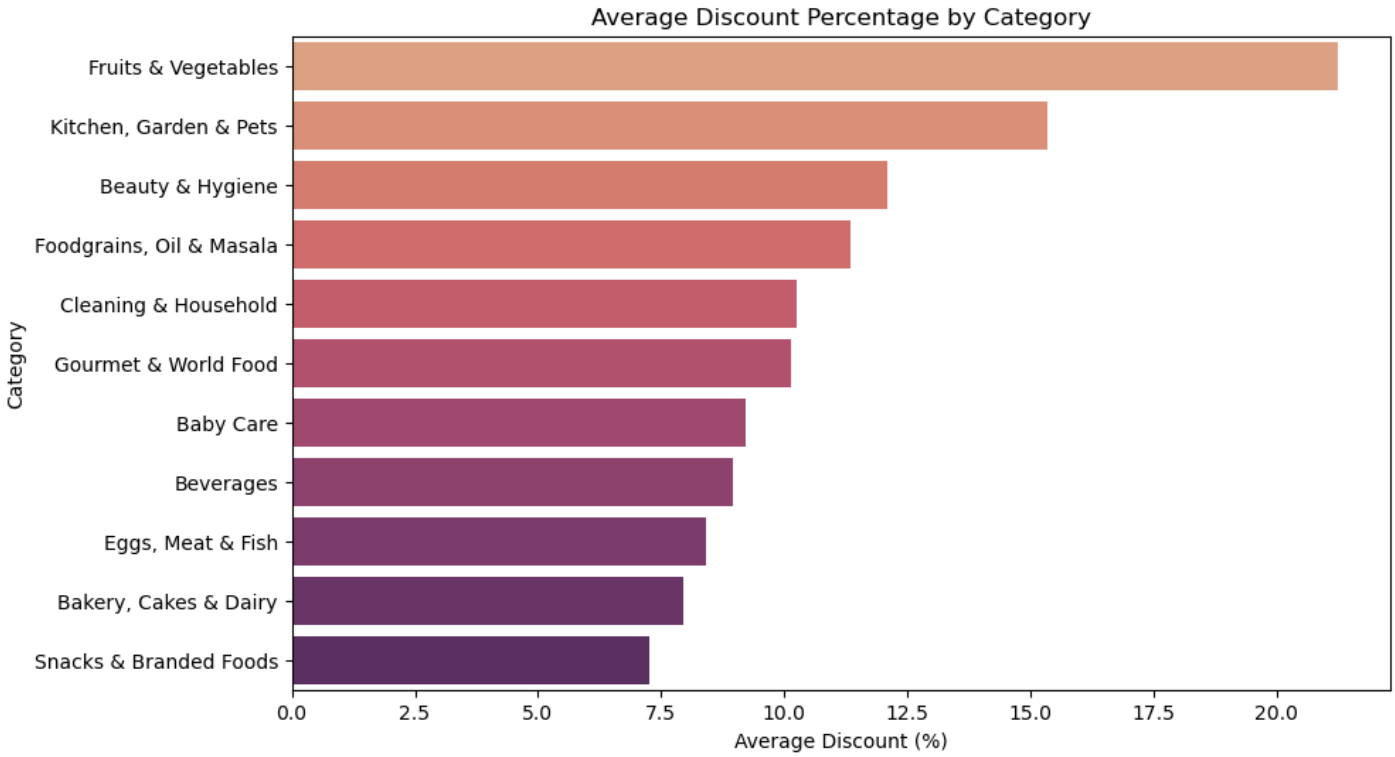
In [77]:

```
df['discount_percent'] = ((df['market_price'] - df['sale_price']) / df['market_price']) * 100
plt.figure(figsize=(10,6))
avg_discount = df.groupby('category')['discount_percent'].mean().sort_values(ascending=False)
sns.barplot(x=avg_discount.values, y=avg_discount.index, palette='flare')
plt.title('Average Discount Percentage by Category')
plt.xlabel('Average Discount (%)')
plt.ylabel('Category')
plt.show()
```

C:\Users\aryan\AppData\Local\Temp\ipykernel_14412\2190907661.py:4: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in v0.14.0. Assign the `y` variable to `hue` and set `legend=False` for the same effect.

```
sns.barplot(x=avg_discount.values, y=avg_discount.index, palette='flare')
```



Conclusion

- Category Distribution: Certain categories such as Foodgrains, Oils & Masala and Beverages dominate the product range, showing Big Basket's focus on daily essentials.
- Price Insights: Most products are priced under ₹500, with only a few premium items above ₹1000, indicating Big Basket's wide appeal to middle-income households.
- Brand Analysis: A few brands contribute a large portion of products and sales, highlighting brand loyalty and strong supplier relationships.
- Discount Trends: The discount percentage varies widely across categories — packaged food and household essentials often have the highest discounts to attract repeat buyers.
- Ratings: Most products have ratings between 3.5 and 5, indicating generally positive customer feedback and trust in Big Basket's product quality.
- Outliers: A small number of products had unusually high or low prices; these were identified and treated (replaced with mean values) to improve data accuracy.

The Exploratory Data Analysis of the Big Basket dataset revealed key insights into pricing, brand performance, and customer preferences. Most products are moderately priced, with essentials like groceries and beverages dominating the inventory. Discounts are highest in everyday-use categories, helping attract frequent buyers. Ratings indicate that customers are generally satisfied, with most products rated above 3.5. A strong correlation between market and sale prices confirms consistent pricing strategies, while outlier treatment and visualizations improved data quality and understanding. Overall, the analysis highlights Big Basket's effective product management and competitive pricing approach.

In [ ]:

In [ ]: