📄 ***Project Objective*** To analyze BigBasket's product data for actionable insights that can support a grocery or e-commerce startup in:

.Inventory planning

..Product categorization

...Price optimization

....Understanding customer preferences

📦 ***Dataset Summary*** The dataset contains information about various products listed on BigBasket. Columns may described as follows:

1. index - Simply the Index!
1. product - Title of the product (as they're listed)
2. category - Category into which product has been classified.
3. sub_category - Subcategory into which product has been kept brand - Brand of the product.
4. sale_price - Price at which product is being sold on the site.
5. market_price - Market price of the product.
6. type - Type into which product falls.
7. rating - Rating the product has got from its consumers.
8. description - Description of the dataset (in detail).

💼 ***BigBasket Product Data Helps in following ways***:

📌 1. Understand Market Demand Analyzing BigBasket's product catalog reveals which categories are most in demand (e.g., dairy, fruits, snacks).

Shows current consumer shopping patterns.

📌 2. Identify Profitable Categories Helps you identify top-selling and low-competition niches.

Useful for building your initial product line or MVP.

📌 3. Optimize Pricing Strategy Product prices across categories let you see price ranges, premium vs. budget trends, and discount patterns.

Useful for setting your own pricing tiers competitively.

📌 4. Learn from Brand Positioning Find which brands dominate in each category.

Decide whether to compete, partner, or offer alternatives (e.g., local brands, private labels).

Import Libraries

```
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
```

Step 1: Load DataSet.

```
data=pd.read_csv("BigBasket Products.csv")
data
```

	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	220.00	220.0	Hair Oil & Serum	4.1	This Product contains Garlic Oil that is known...
1	2	Water Bottle - Orange	Kitchen, Garden & Pets	Storage & Accessories	Mastercook	180.00	180.0	Water & Fridge Bottles	2.3	Each product is microwave safe (without 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 gift for all occasions, be it your m...
3	4	Cereal Flip Lid Container/Storage Jar - Assort...	Cleaning & Household	Bins & Bathroom Ware	Nakoda	149.00	176.0	Laundry, Storage Baskets	3.7	Multipurpose container with an attractive desi...
4	5	Creme Soft Soap - For Hands & Body	Beauty & Hygiene	Bath & Hand Wash	Nivea	162.00	162.0	Bathing Bars & Soaps	4.4	Nivea Creme Soft Soap gives your skin the best...
...	...	...	...	...	...	...	...	...	...	...
27550	27551	Wottagirl! Perfume Spray - Heaven, Classic	Beauty & Hygiene	Fragrances & Deos	Layerr	199.20	249.0	Perfume	3.9	Layerr brings you Wottagirl Classic fragrant b...
27551	27552	Rosemary	Gourmet & World Food	Cooking & Baking Needs	Puramate	67.50	75.0	Herbs, Seasonings & Rubs	4.0	Puramate rosemary is enough to transform a dis...
27552	27553	Peri-Peri Sweet Potato Chips	Gourmet & World Food	Snacks, Dry Fruits, Nuts	FabBox	200.00	200.0	Nachos & Chips	3.8	We have taken the richness of Sweet Potatoes (...
27553	27554	Green Tea - Pure Original	Beverages	Tea	Tetley	396.00	495.0	Tea Bags	4.2	Tetley Green Tea with its refreshing pure, ori...
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 new mens fragrance from the United Dreams ...

27555 rows × 10 columns

Step 2. Head function to look for first 12 rows

data.head(12)

	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 Product contains Garlic Oil that is known...
1	2	Water Bottle - Orange	Kitchen, Garden & Pets	Storage & Accessories	Mastercook	180.0	180.0	Water & Fridge Bottles	2.3	Each product is microwave safe (without 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 gift for all occasions, be 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	Multipurpose container with an attractive 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 Creme Soft Soap gives your skin the best...
5	6	Germ - Removal Multipurpose Wipes	Cleaning & Household	All Purpose Cleaners	Nature Protect	169.0	199.0	Disinfectant Spray & Cleaners	3.3	Stay protected from contamination with Multipu...
6	7	Multani Mati	Beauty & Hygiene	Skin Care	Satinance	58.0	58.0	Face Care	3.6	Satinance multani matti is an excellent 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%Alcohol based is gentle of hand leaves skin...
8	9	Biotin & Collagen Volumizing Hair Shampoo + Bi...	Beauty & Hygiene	Hair Care	StBotanica	1098.0	1098.0	Shampoo & Conditioner	3.5	An exclusive blend with Vitamin B7 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 Brite Anti- Bacterial Scrub Pad thoroug...
10	11	Wheat Grass Powder - Raw	Gourmet & World Food	Cooking & Baking Needs	NUTRASHIL	261.0	290.0	Flours & Pre-Mixes	4.0	Wheatgrass is a superfood potent health food w...
11	12	Butter Cookies Gold Collection	Gourmet & World Food	Chocolates & Biscuits	Sapphire	600.0	600.0	Luxury Chocolates, Gifts	2.2	Enjoy a tin full of delicious butter cookies m...

Firstly,check for the duplicated row if existed.

```
data.duplicated().sum()
```

```
np.int64(0)
```

Step 3. Description of the data in the DataFrame

data.describe()

	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

Step4. Information about the DataFrame.

```
data.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
```

Step5. Top & least sold products

```
data["product"].unique()
```

```
array(['Garlic Oil - Vegetarian Capsule 500 mg', 'Water Bottle - Orange',
       'Brass Angle Deep - Plain, No.2', ...,
       'Wottagirl! Perfume Spray - Heaven, Classic',
       'Peri-Peri Sweet Potato Chips', 'Green Tea - Pure Original'],
      dtype=object)
```

```
data["product"].value_counts()
```

```
product
Turmeric Powder/Arisina Pudi                                26
Extra Virgin Olive Oil                                      15
Cow Ghee/Tuppa                                              14
Olive Oil - Extra Virgin                                    12
Soft Drink                                                  12
                                                            ..
Opalware Classique Serving Bowl - Medium, Royal Irish        1
Lavangadi Vati - Respiratory Conditions, 300mg               1
Pomegranate - Small                                          1
Butter - Cashew, Smooth                                      1
Powder - Pepper                                              1
Name: count, Length: 23540, dtype: int64
```

Top Products

```
data["product"].value_counts().head()
```

```
product
Turmeric Powder/Arisina Pudi    26
Extra Virgin Olive Oil          15
Cow Ghee/Tuppa                  14
Olive Oil - Extra Virgin        12
Soft Drink                      12
Name: count, dtype: int64
```

Least Products

```
data["product"].value_counts().tail()
```

```
product
Opalware Classique Serving Bowl - Medium, Royal Irish    1
```

```
Lavangadi Vati - Respiratory Conditions, 300mg          1
Pomegranate - Small                                     1
Butter - Cashew, Smooth                                 1
Powder - Pepper                                         1
Name: count, dtype: int64
```

Step6. Measuring discount on a certain items.

```
data["discount"]=data["market_price"]-data["sale_price"]
data["discount"]
```

```
0          0.00
1          0.00
2        131.00
3         27.00
4          0.00
          ...
27550     49.80
27551      7.50
27552      0.00
27553     99.00
27554    175.47
Name: discount, Length: 27555, dtype: float64
```

Step7. To find the Missing Values from the Dataset

data.isnull().sum()

```
index              0
product            1
category           0
sub_category       0
brand              1
sale_price         6
market_price       0
type               0
rating          8636
description      115
discount           6
dtype: int64
```

data.info()

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 27555 entries, 0 to 27554
Data columns (total 11 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
 10  discount      27549 non-null  float64
dtypes: float64(4), int64(1), object(6)
memory usage: 2.3+ MB
```

Filling of missing values

```
data["product"].mode()[0]
```

```
{"type":"string"}
```

```
data["product"]=data["product"].fillna(data["product"].mode()[0])
```

```
data["brand"].mode()[0]
```

```
{"type":"string"}
```

```
data["brand"]=data["brand"].fillna(data["brand"].mode()[0])
```

```
data["sale_price"].mean()
```

```
np.float64(334.6483912301717)
```

```
data["sale_price"]=data["sale_price"].fillna(data["sale_price"].mean())
```

```
data["rating"].mean()
```

```
np.float64(3.9432951001638563)
```

```
data["rating"]=data["rating"].fillna(data["rating"].mean())
```

```
data["description"].mode()[0]
```

```
{"type":"string"}
```

```
data["description"]=data["description"].fillna(data["description"].mode()[0])
```

```
data["discount"].mean()
```

```
np.float64(47.469267850012706)
```

```
data["discount"]=data["discount"].fillna(data["discount"].mean())
```

```
data.isnull().sum()/len(data)*100
```

```
index           0.0
product         0.0
```

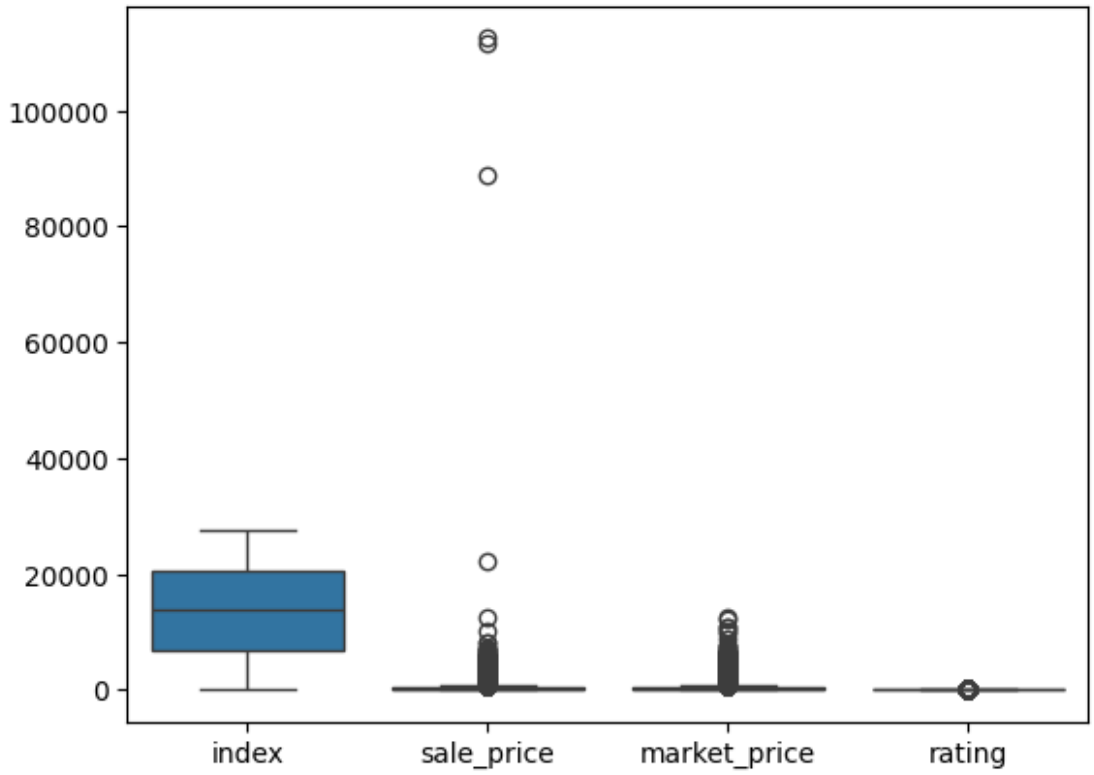
```
category         0.0
sub_category     0.0
brand            0.0
sale_price       0.0
market_price     0.0
type             0.0
rating           0.0
description      0.0
discount         0.0
dtype: float64
```

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

Firstly we check the outliers.

```
import seaborn as sns

sns.boxplot(data=data[["index","sale_price","market_price","rating"]])
```

```
<Axes: >
```



Removal of outliers according to columns.

Sale_price

```
## calculate iqr and outliers fence
```

```
q1=data["sale_price"].quantile(0.25)
q3=data["sale_price"].quantile(0.75)
iqr=q3-q1
iqr
```

```
np.float64(264.0)
```

```
lower_fence=max(q1-1.5*iqr,0)
upper_fence=q3+1.5*iqr
```

```
lower_fence
```

```
0
```

```
upper_fence
```

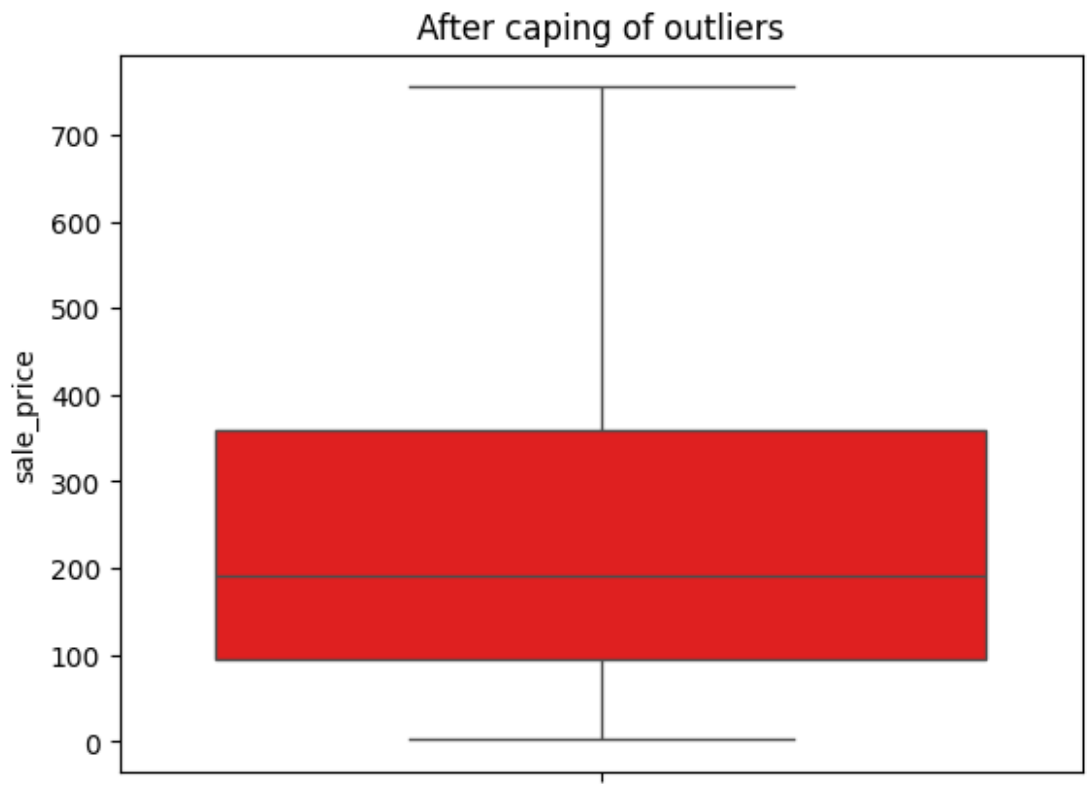
```
np.float64(755.0)
```

```
# To detect the number of outliers present in sale_price
outliers=data[(data["sale_price"]< lower_fence) | (data["sale_price"] >
upper_fence)]
print("Outliers_count:", len(outliers))
```

```
Outliers_count: 2205
```

```
# removing the outliers
data['sale_price']=np.where(data['sale_price']<=lower_fence,lower_fence,data[
'sale_price'])
data['sale_price']=np.where(data['sale_price']>=upper_fence,upper_fence,data[
'sale_price'])
```

```
#Boxplot after caping of outliers
sns.boxplot(data['sale_price'],color="red")
plt.title("After caping of outliers")
plt.show()
```




```
#checking of outliers after caping in sale_price
outliers_after_caping = data[(data["sale_price"] < lower_fence) &
(data["sale_price"] > upper_fence)]
print("outliers left after caping in
sale_price:",outliers_after_caping.shape[0])
```

```
outliers left after caping in sale_price: 0
```

Market_price

```
# Calculate outliers and outer fence
q1=data["market_price"].quantile(0.25)
q3=data["market_price"].quantile(0.75)
iqr=q3-q1
iqr
```

```
np.float64(325.0)
```

```
lower_fence=max((q1-1.5*iqr,0))
upper_fence=q3+1.5*iqr
```

```
lower_fence
```

```
0
```

```
upper_fence
```

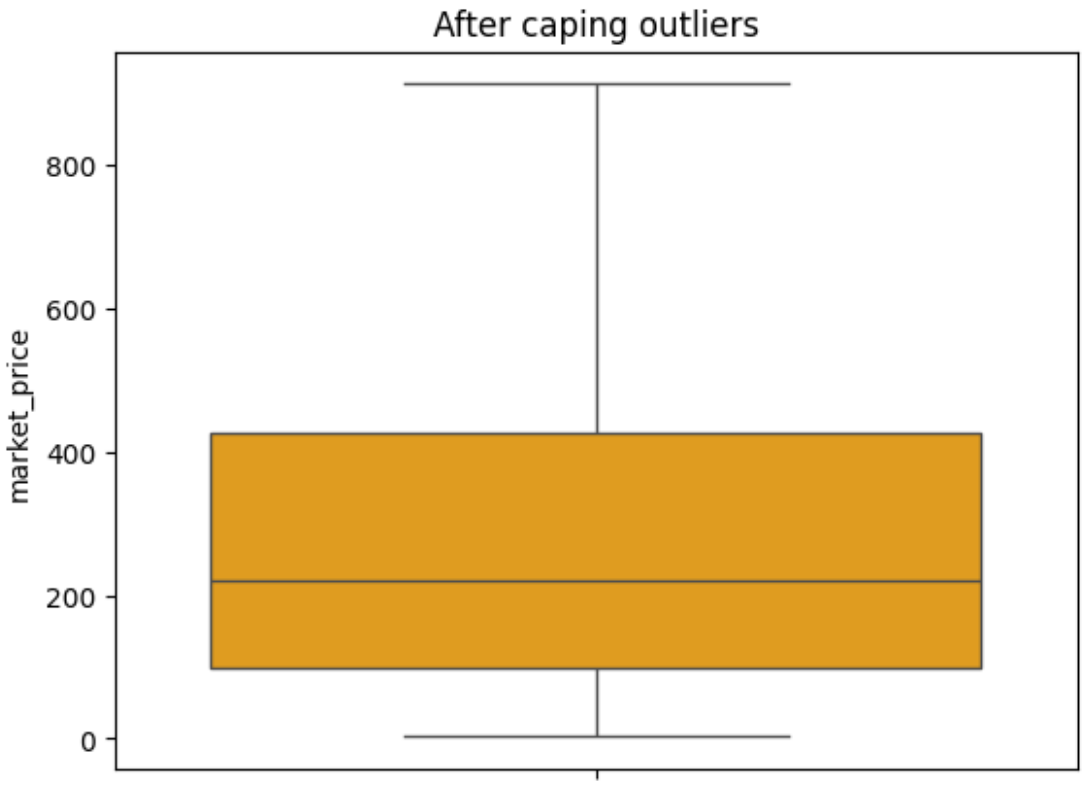
```
np.float64(912.5)
```

```
# To detect the number of outliers present in market_price
outliers=data[(data["market_price"]< lower_fence) | (data["market_price"] >
upper_fence)]
print("Outliers_count:", len(outliers))
```

```
Outliers_count: 2147
```

```
# Caping of outliers
data['market_price']=np.where(data['market_price']<=lower_fence,lower_fence,d
ata['market_price'])
data['market_price']=np.where(data['market_price']>=upper_fence,upper_fence,d
ata['market_price'])
```

```
# Showing boxplot after caping of outliers
sns.boxplot(data["market_price"], color="orange")
plt.title("After caping outliers")
plt.show()
```




```
#checking of outliers after caping in market_price
outliers_after_caping = data[(data["market_price"] < lower_fence) &
(data["market_price"] > upper_fence)]
print("outliers left after caping in
market_price:",outliers_after_caping.shape[0])
```

```
outliers left after caping in market_price: 0
```

RATING

```
# Calculate outliers and outer fence of rating
q1=data["rating"].quantile(0.25)
q3=data["rating"].quantile(0.75)
iqr=q3-q1
iqr
```

```
np.float64(0.25670489983614386)
```

```
lower_fence=q1-1.5*iqr
upper_fence=q3+1.5*iqr
```

```
lower_fence
```

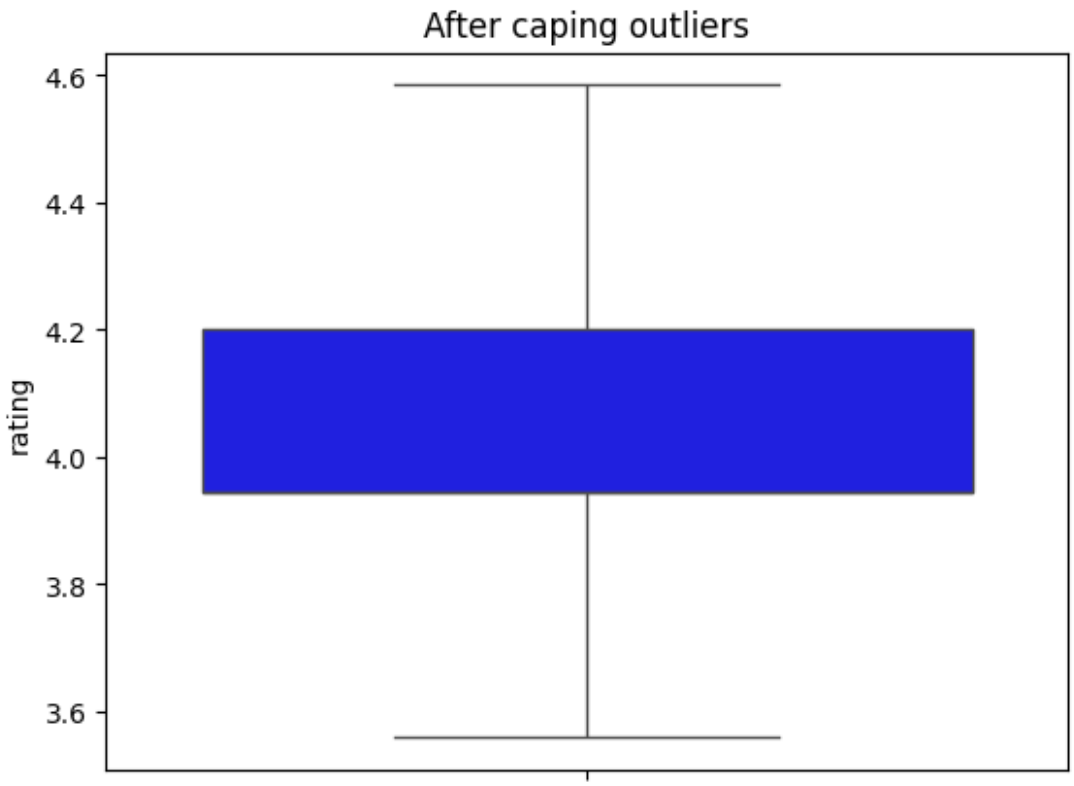
```
np.float64(3.5582377504096403)
```

```
upper_fence
```

```
np.float64(4.585057349754216)
```

```
# Caping of outliers
data["rating"]=np.where(data["rating"]<lower_fence,lower_fence,data["rating"]
)
data["rating"]=np.where(data["rating"]>upper_fence,upper_fence,data["rating"]
)
```

```
# Boxplot for rating after caping outliers
sns.boxplot(data["rating"], color="blue")
plt.title("After caping outliers")
plt.show()
```

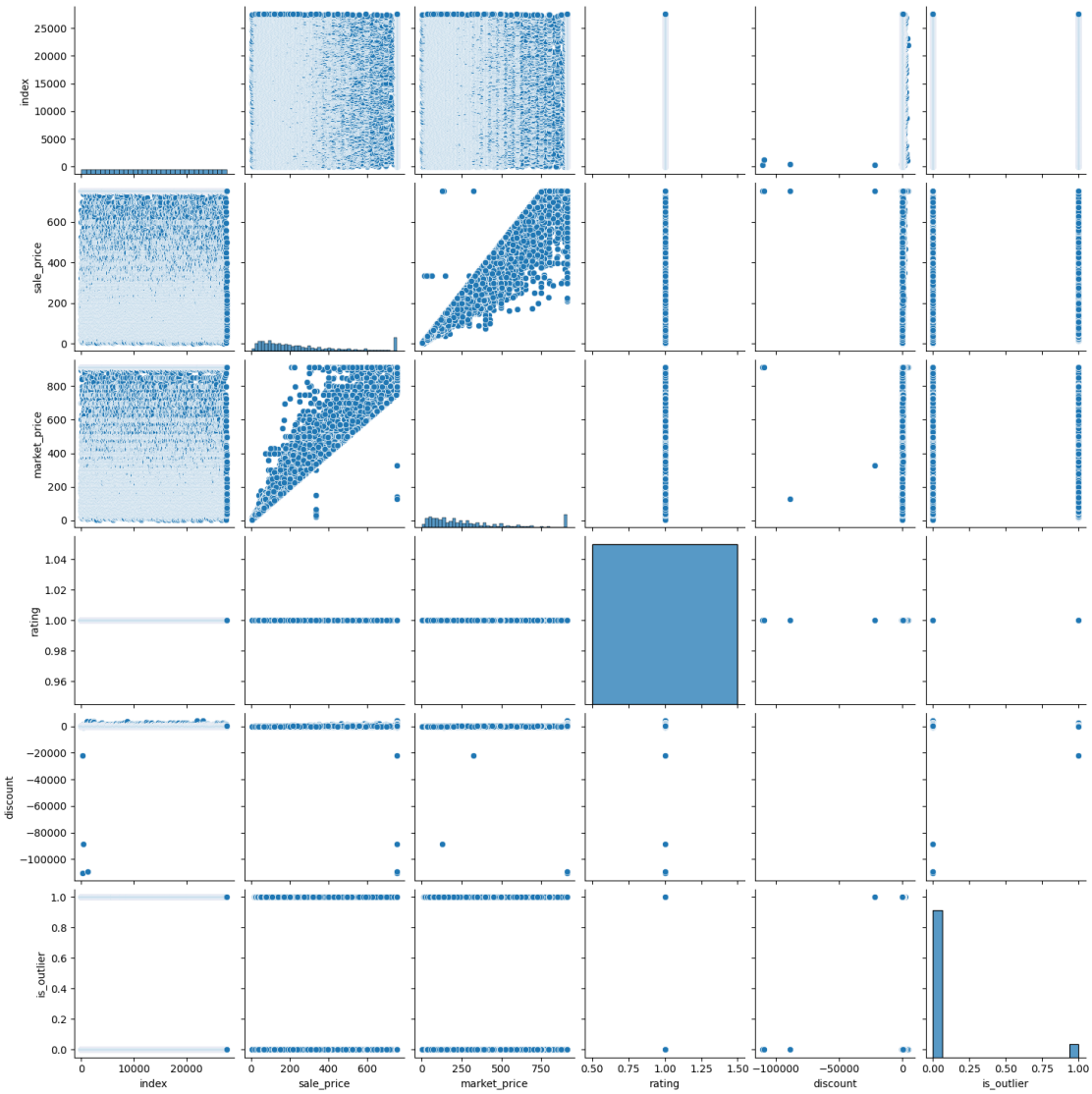



```
#checking of outliers after caping in rating
outliers_after_caping = data[(data["rating"] < lower_fence) & (data["rating"]
> upper_fence)]
print("outliers left after caping in rating:",outliers_after_caping.shape[0])
```

```
outliers left after caping in rating: 0
```

Step 9. Create Plots and Visualizations

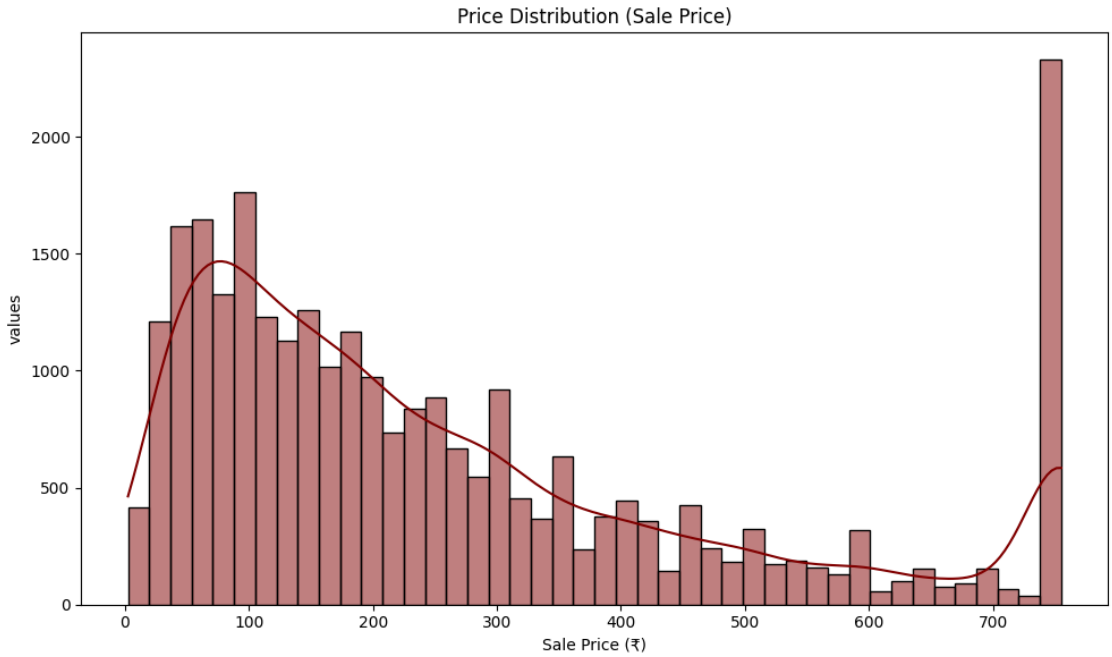
```
#pairplot of all data
sns.pairplot(data)
plt.show()
```



Insights for pairplot:

It automatically selects numerical columns from data and, plots scatter plots between each pair of variables.

```
#Price Distribution of Sale price
plt.figure(figsize=(10, 6))
sns.histplot(data["sale_price"],kde=True,color='maroon')
plt.title("Price Distribution (Sale Price)")
plt.xlabel("Sale Price (₹)")
plt.ylabel("values")
plt.tight_layout()
plt.show()
```



📊 **Insights for Price Distribution of Sale_price:** This histplot shows that most products are concentrated on the lower price range (₹300 – ₹7500).

1. The density decreases gradually as price increases, indicating fewer high-priced items.

1. The long tail on the right side suggests the presence of premium or bulk-packaged products priced much higher than average.

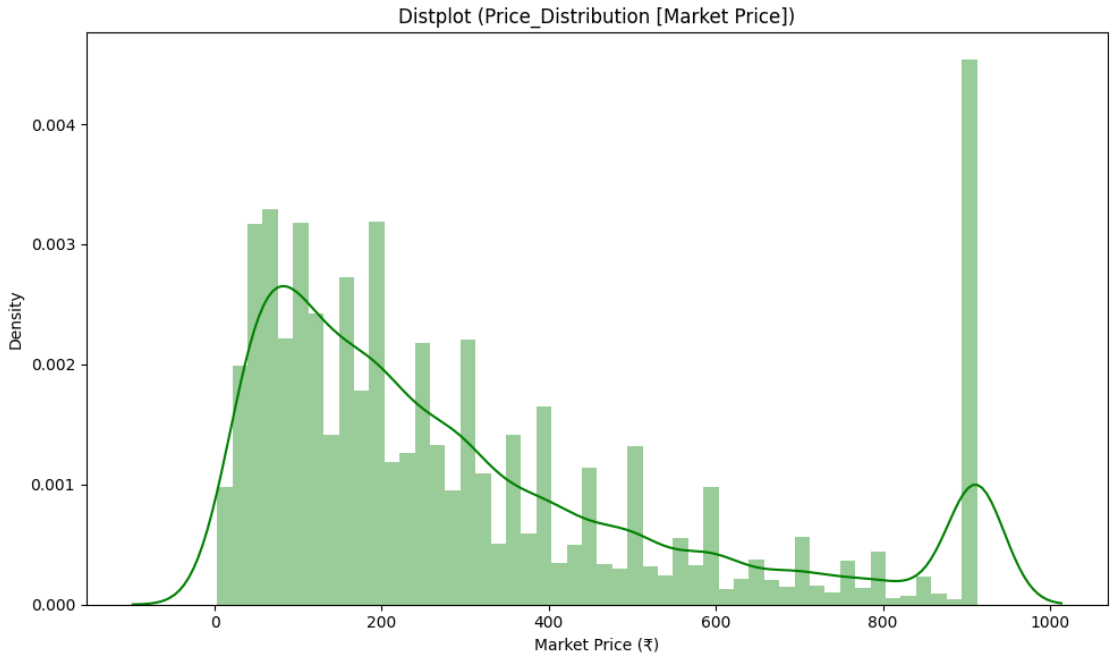
2. The Kernel Density Estimate (KDE) line helps visualize the smooth probability distribution of sale prices.

```
#Price Distribution of market price
plt.figure(figsize=(10, 6))
sns.distplot(data["market_price"], color='green',bins=50, kde=True,
hist=True)
plt.title("Distplot (Price_Distribution [Market Price])")
plt.xlabel("Market Price (₹)")
plt.ylabel("Density")
plt.tight_layout()
plt.show()
```

```
/tmp/ipython-input-242-2051947461.py:3: UserWarning:

`distplot` is a deprecated function and will be removed in seaborn v0.14.0.

Please adapt your code to use either `displot` (a figure-level function with
similar flexibility) or `histplot` (an axes-level function for histograms).

For a guide to updating your code to use the new functions, please see
```

https://gist.github.com/mwaskom/de44147ed2974457ad6372750bbe5751

```
  sns.distplot(data["market_price"], color='green',bins=50, kde=True,
hist=True)
```

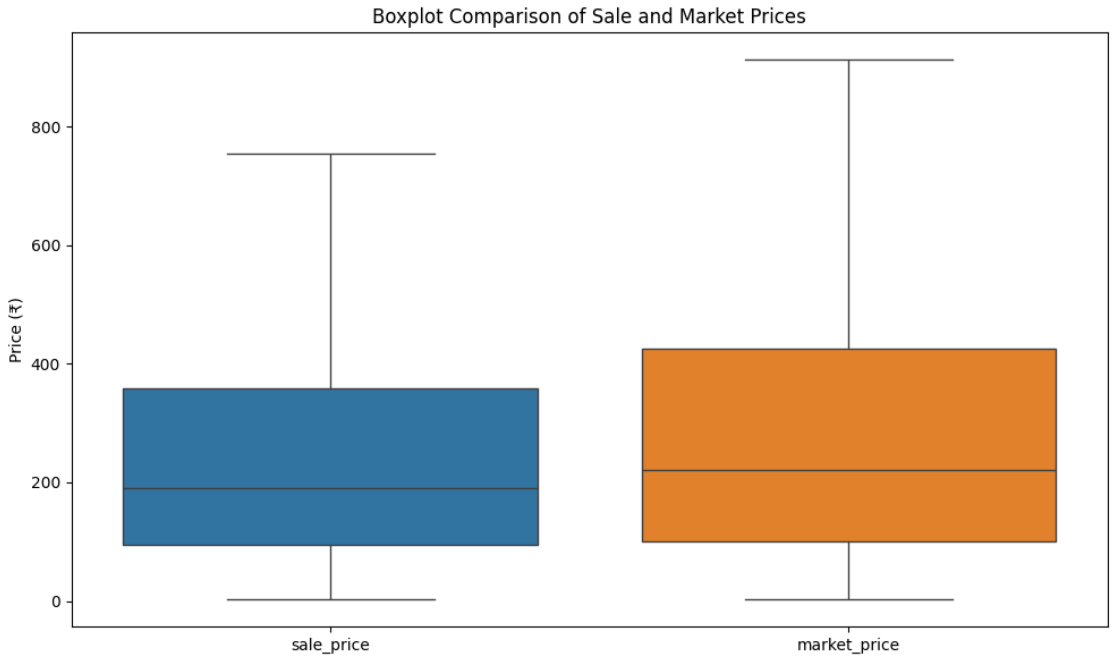


🛒 Insights: Market Price Distribution:

1. Similar to sale_price, market_price is heavily right-skewed. Most values lie in the low to mid price range, typically under ₹500.

1. The KDE curve might show multiple peaks, suggesting: Popular market pricing tiers like ₹100, ₹250, ₹500 are commonly used.

```
#Comparison between sale price and market price
plt.figure(figsize=(10, 6))
sns.boxplot(data=data[['sale_price', 'market_price']])
plt.title("Boxplot Comparison of Sale and Market Prices")
plt.ylabel("Price (₹)")
plt.tight_layout()
plt.show()
```



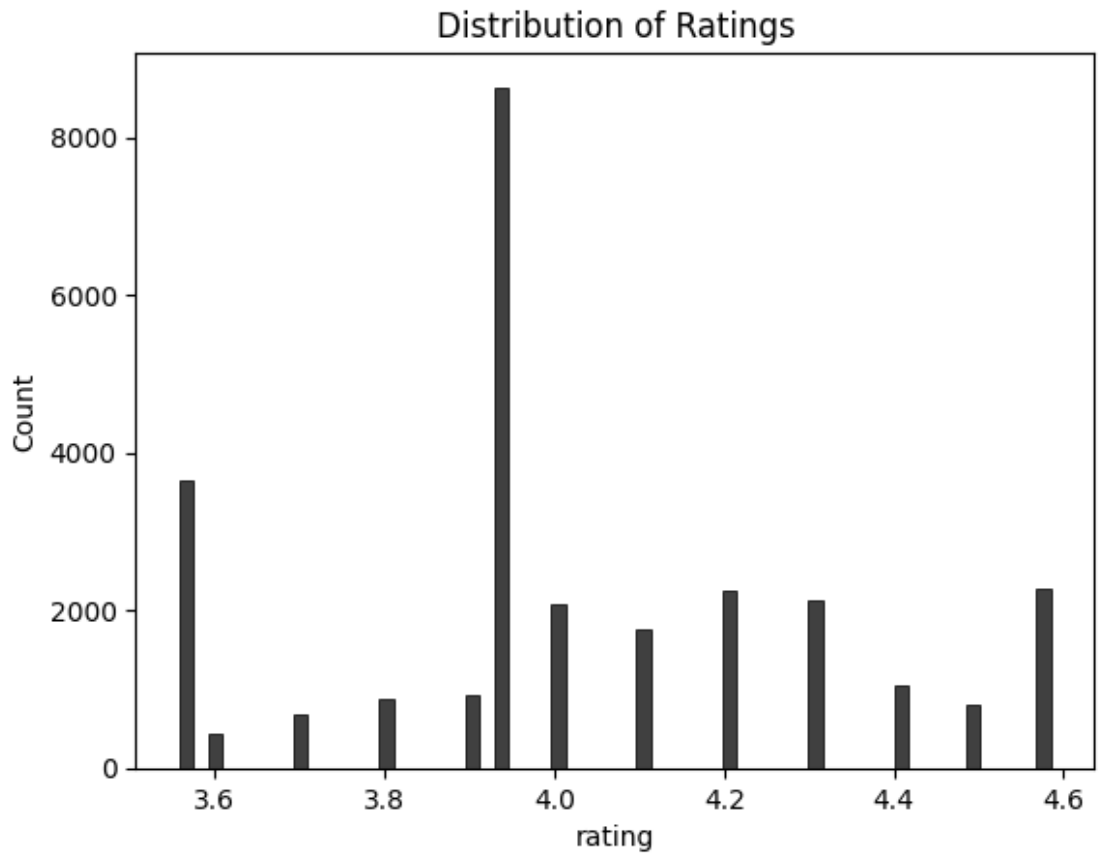
📦 💰 ***Insights: Boxplot*** – Sale Price vs Market Price

1. 🔻 Sale Prices Are Generally Lower The median of sale_price is clearly lower than that of market_price, confirming that products are commonly sold at a discount. This validates the presence of promotional pricing strategies.

1. 📈 Market Price Has Higher Spread and Variability The interquartile range (IQR) for market_price is wider.Some products are priced much higher, possibly due to category, size, or brand premium.

2. The consistent gap between sale_price and market_price supports: An intentional pricing policy to show savings. Creation of a perceived value proposition that encourages purchases.

```
# Histogram for distribution of ratings
sns.histplot(data['rating'],color="black")
plt.title("Distribution of Ratings")
plt.show()
```



⭐ ***Insights***: Distribution of Ratings

1. Most products have high ratings. This indicates generally good product quality or strong customer satisfaction.

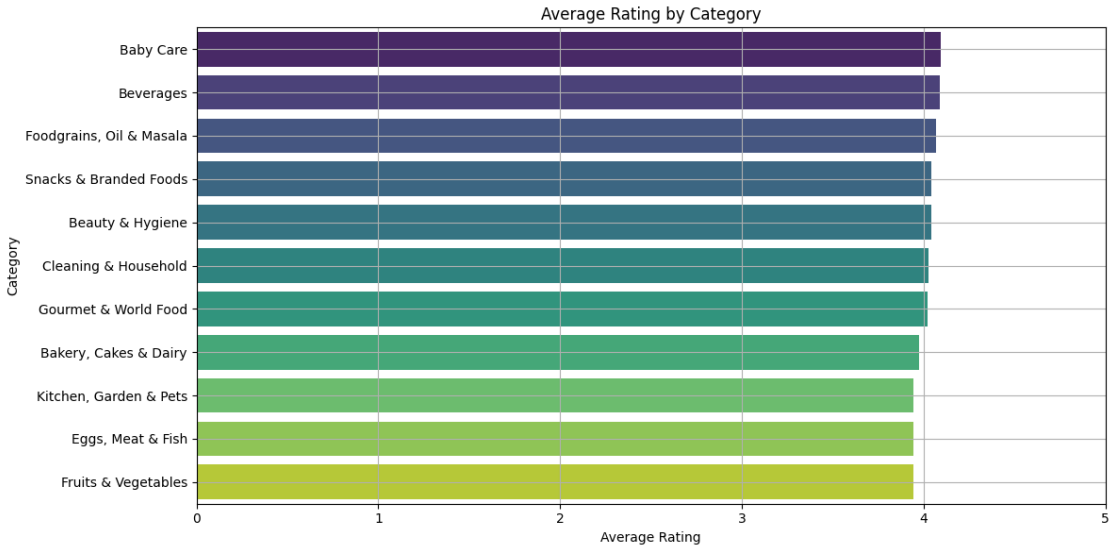
1. Low-rated products are either poor performers or less reviewed, and could benefit from: Improved descriptions, packaging, or delivery experience. Review solicitation or product replacement.

2. The abundance of higher ratings builds trust for new customers and increases conversion likelihood.Consider highlighting top-rated products in marketing sections.

```
#To find avg rating by catagories
rating_by_category = data[['category', 'rating']].dropna()

avg_rating =
rating_by_category.groupby('category')['rating'].mean().sort_values(ascending
=False)

#Bar plot of avg rating by category
plt.figure(figsize=(12, 6))
sns.barplot(x=avg_rating.values, y=avg_rating.index, palette='viridis')
plt.title("Average Rating by Category")
```

```
plt.xlabel("Average Rating")
plt.ylabel("Category")
plt.xlim(0, 5)
plt.grid(True)
plt.tight_layout()
plt.show()
```

```
/tmp/ipython-input-247-2541320514.py:2: 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_rating.values, y=avg_rating.index, palette='viridis')
```



⭐📊 ***Insights:*** Average Rating by Category

1. Categories at the top of the chart represent products that are: High in customer satisfaction. Consistently well-reviewed in quality, packaging, or delivery.

1. Categories with average ratings below 3.5 may need attention:

Could indicate product quality issues, misleading descriptions, or stock problems. Suggest action like supplier review, product delisting, or customer feedback analysis.

```
data.columns
```

```
Index(['index', 'product', 'category', 'sub_category', 'brand', 'sale_price',
       'market_price', 'type', 'rating', 'description', 'discount'],
      dtype='object')
```

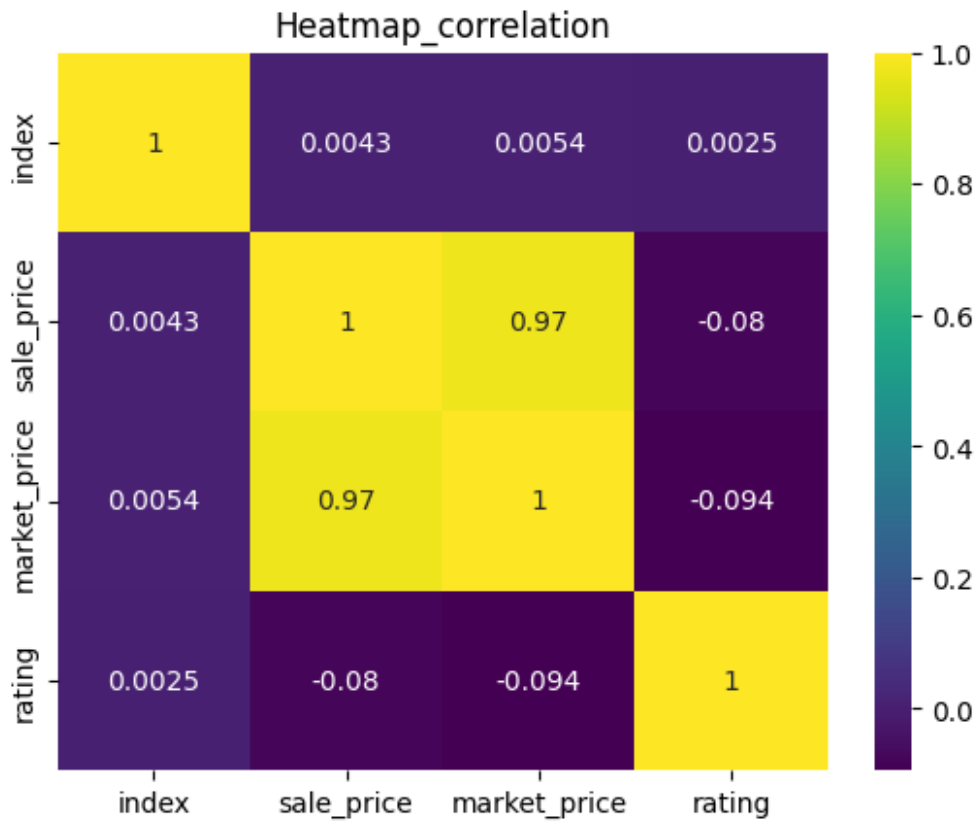
```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 27555 entries, 0 to 27554
Data columns (total 11 columns):
 #   Column         Non-Null Count  Dtype  
---  ------         --------------  -----  
 0   index          27555 non-null  int64  
 1   product        27555 non-null  object 
 2   category       27555 non-null  object 
 3   sub_category   27555 non-null  object 
 4   brand          27555 non-null  object 
 5   sale_price     27555 non-null  float64
 6   market_price   27555 non-null  float64
 7   type           27555 non-null  object 
 8   rating         27555 non-null  float64
 9   description    27555 non-null  object 
 10  discount       27555 non-null  float64
dtypes: float64(4), int64(1), object(6)
memory usage: 2.3+ MB
```

```
# To find the heapmap correlation.
df=data[["index","sale_price","market_price","rating"]]

df.corr()
```

	index	sale_price	market_price	rating
index	1.000000	0.004277	0.005392	0.002550
sale_price	0.004277	1.000000	0.971350	-0.079937
market_price	0.005392	0.971350	1.000000	-0.094196
rating	0.002550	-0.079937	-0.094196	1.000000

```
sns.heatmap(df.corr(),annot=True,cmap="viridis")
plt.title("Heatmap_correlation")
plt.show()
```



🔥 📈 ***Insights from Correlation Heatmap***

1. 🔢 Interpretation of Values Correlation values range from -1 to +1:

+1: Perfect positive correlation

-1: Perfect negative correlation

0: No correlation

1. 🔗 Highly Correlated Features Identify any cells with high positive correlation (e.g. > 0.75): Example: market_price vs sale_price is usually highly correlated — which is expected.

Strong correlation may indicate redundancy, useful for feature reduction in modeling.

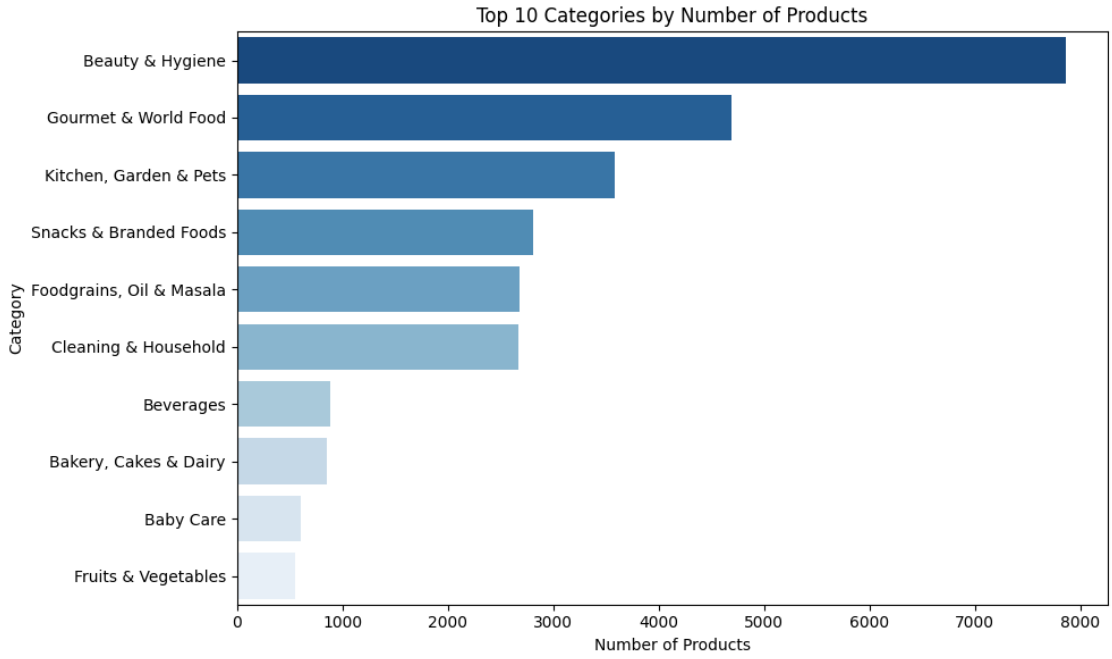
1. ⚠️ Low or No Correlation Features with correlations near 0 have no linear relationship. This doesn't mean they're not useful—some models (like tree-based) capture non-linear relationships.

1. 📉 Negative Correlation Look for values < 0: These show inverse relationships (e.g., higher discounts might correlate negatively with ratings in some edge cases).

.

```
# Barplot of top 10 catagories
category_counts = data['category'].value_counts().nlargest(10)
plt.figure(figsize=(10, 6))
sns.barplot(x=category_counts.values, y=category_counts.index,
palette='Blues_r')
plt.title("Top 10 Categories by Number of Products")
plt.xlabel("Number of Products")
plt.ylabel("Category")
plt.tight_layout()
plt.show()
```

```
/tmp/ipython-input-253-2296679758.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=category_counts.values, y=category_counts.index,
palette='Blues_r')
```



📊🧺 *Insights: Top 10 Product Categories*

1. 🥇 Most Populated Categories The top 1–3 categories are:

Beauty & Hygiene

Gourmet & World Food

Kitchen,Garden &Pets

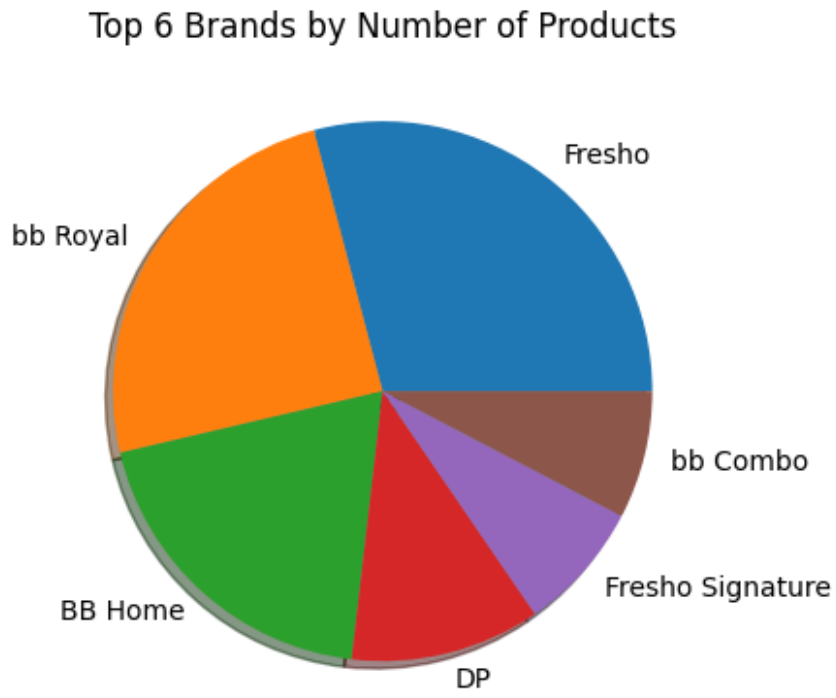
These categories dominate inventory and reflect core customer demand.

1. The fact that just 10 categories account for a large number of products suggests:

A long-tail strategy where fewer products exist in niche categories.

Top categories may receive more frequent updates, offers, and shelf space.

```
# Piechart of Top 6 brands
plt.figure(figsize=(8, 4))
top_brands = data['brand'].value_counts().nlargest(6)
plt.pie(top_brands,labels=top_brands.index,shadow=True)
plt.title("Top 6 Brands by Number of Products")
plt.tight_layout()
plt.show()
```



🧠📈 ***Insights: Top 6 Brands by Product Count (Pie Chart)***

1. One or two slices are likely significantly larger—these brands dominate the catalog.

1. 🧃 High Product Variety These top brands offer many SKUs, possibly due to:

Variants (flavors, sizes, packaging)

Serving multiple categories (e.g., beverages, staples, and cleaning)

1. 🎯 Strategic Partners These brands are key for partnerships, promotions, and visibility on the platform.

Consider featuring them in:

Deals of the day

Category highlights

Brand stores

📄 Final Project Conclusion:

Through detailed exploration of product categories, ratings, pricing, and brands, the project uncovered valuable insights into consumer trends, market positioning, and product assortment strategies.

Key takeaways include:

High product concentration in essential categories such as fruits, vegetables, dairy, and snacks.

Opportunities to target underserved or high-demand niches identified through frequency analysis.

This analysis not only deepens understanding of the online grocery ecosystem but also provides actionable intelligence for:

Product catalog planning

Pricing strategies

Inventory decisions

Launching a Minimum Viable Product (MVP)

In conclusion, the BigBasket product data offers a practical foundation for startups to design a data-driven, customer-oriented, and competitive business model in the digital retail space.