```
# Exploratory Data Analysis of BigBasket Project
```

Problem Statement: Optimizing Delivery Operations and Enhancing Customer Experience for Bigbasket

Bigbasket faces challenges in ensuring timely deliveries, especially during peak hours and in regions with high demand. This affects customer satisfaction, increases operational costs, and leads to inefficiencies in resource allocation. The company seeks to address these issues by leveraging data analytics to optimize its delivery operations and improve the overall customer experience.

Goal for Bigbasket Data Analytics Project

Primary Goal: Leverage data analytics to enhance operational efficiency, improve customer satisfaction, and drive sustainable growth for Bigbasket through data-driven decision-making.

Dataset Description

This dataset contains 10 attributes with simple meaning and which are described as follows:

- **index** -- Simply the Index!
- **product** -- Title of the product ()
- **categroy** -- Category into which product has been classified
- **sub_category** -- Subcategory into which product has been kept
- **brand** -- Brand of the product
- **sale_price** -- Price at which product is being sold on the site
- **market_price** -- Market price of the product
- **type** -- Type into which product falls
- **rating** -- Rating the product has got from its consumers
- **description** -- Description of the dataset (in detail)

Import Libraries

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

```
import seaborn as sns
import plotly.express as px
import plotly.io as pio
```

Step 1: Load DataSet.

```
df = pd.read_csv('BigBasket Products.csv')
df
```

```
       index                                            product  \
0          1             Garlic Oil - Vegetarian Capsule 500 mg
1          2                               Water Bottle - Orange
2          3                        Brass Angle Deep - Plain, No.2
3          4  Cereal Flip Lid Container/Storage Jar - Assort...
4          5                    Creme Soft Soap - For Hands & Body
...      ...                                                ...
27550  27551         Wottagirl! Perfume Spray - Heaven, Classic
27551  27552                                           Rosemary
27552  27553                      Peri-Peri Sweet Potato Chips
27553  27554                         Green Tea - Pure Original
27554  27555                      United Dreams Go Far Deodorant

                     category              sub_category  \
0           Beauty & Hygiene                 Hair Care
1      Kitchen, Garden & Pets     Storage & Accessories
2        Cleaning & Household              Pooja Needs
3        Cleaning & Household      Bins & Bathroom Ware
4           Beauty & Hygiene          Bath & Hand Wash
...                      ...                       ...
27550       Beauty & Hygiene         Fragrances & Deos
27551  Gourmet & World Food    Cooking & Baking Needs
27552  Gourmet & World Food  Snacks, Dry Fruits, Nuts
27553             Beverages                       Tea
27554       Beauty & Hygiene            Men's Grooming

                          brand  sale_price  market_price  \
0              Sri Sri Ayurveda      220.00         220.0
1                     Mastercook     180.00         180.0
2                            Trm     119.00         250.0
3                         Nakoda     149.00         176.0
4                          Nivea     162.00         162.0
...                          ...        ...           ...
27550                     Layerr     199.20         249.0
27551                   Puramate      67.50          75.0
27552                     FabBox     200.00         200.0
27553                     Tetley     396.00         495.0
27554  United Colors Of Benetton     214.53         390.0
```

```
                          type  rating  \
0              Hair Oil & Serum     4.1
1        Water & Fridge Bottles     2.3
2                Lamp & Lamp Oil     3.4
3      Laundry, Storage Baskets     3.7
4          Bathing Bars & Soaps     4.4
...                         ...     ...
27550                  Perfume     3.9
27551  Herbs, Seasonings & Rubs     4.0
27552            Nachos & Chips     3.8
27553                  Tea Bags     4.2
27554          Men's Deodorants     4.5

                                              description
0      This Product contains Garlic Oil that is known...
1      Each product is microwave safe (without lid), ...
2      A perfect gift for all occasions, be it your m...
3      Multipurpose container with an attractive desi...
4      Nivea Creme Soft Soap gives your skin the best...
...                                                  ...
27550  Layerr brings you Wottagirl Classic fragrant b...
27551  Puramate rosemary is enough to transform a dis...
27552  We have taken the richness of Sweet Potatoes (...
27553  Tetley Green Tea with its refreshing pure, ori...
27554  The new mens fragrance from the United Dreams ...

[27555 rows x 10 columns]
```

Step 2: Look first 12 rows.

```
df.head(12)
```

```
    index                                            product  \
0       1            Garlic Oil - Vegetarian Capsule 500 mg
1       2                           Water Bottle - Orange
2       3                    Brass Angle Deep - Plain, No.2
3       4  Cereal Flip Lid Container/Storage Jar - Assort...
4       5                 Creme Soft Soap - For Hands & Body
5       6                  Germ - Removal Multipurpose Wipes
6       7                                      Multani Mati
7       8                  Hand Sanitizer - 70% Alcohol Base
8       9  Biotin & Collagen Volumizing Hair Shampoo + Bi...
9      10               Scrub Pad - Anti- Bacterial, Regular
10     11                          Wheat Grass Powder - Raw
11     12                    Butter Cookies Gold Collection

                  category            sub_category               brand
\
```

0	Beauty & Hygiene	Hair Care	Sri Sri Ayurveda	
1	Kitchen, Garden & Pets	Storage & Accessories	Mastercook	
2	Cleaning & Household	Pooja Needs	Trm	
3	Cleaning & Household	Bins & Bathroom Ware	Nakoda	
4	Beauty & Hygiene	Bath & Hand Wash	Nivea	
5	Cleaning & Household	All Purpose Cleaners	Nature Protect	
6	Beauty & Hygiene	Skin Care	Satinance	
7	Beauty & Hygiene	Bath & Hand Wash	Bionova	
8	Beauty & Hygiene	Hair Care	StBotanica	
9	Cleaning & Household	Mops, Brushes & Scrubs	Scotch brite	
10	Gourmet & World Food	Cooking & Baking Needs	NUTRASHIL	
11	Gourmet & World Food	Chocolates & Biscuits	Sapphire	

	sale_price	market_price	type	rating	\
0	220.0	220.0	Hair Oil & Serum	4.1	
1	180.0	180.0	Water & Fridge Bottles	2.3	
2	119.0	250.0	Lamp & Lamp Oil	3.4	
3	149.0	176.0	Laundry, Storage Baskets	3.7	
4	162.0	162.0	Bathing Bars & Soaps	4.4	
5	169.0	199.0	Disinfectant Spray & Cleaners	3.3	
6	58.0	58.0	Face Care	3.6	
7	250.0	250.0	Hand Wash & Sanitizers	4.0	
8	1098.0	1098.0	Shampoo & Conditioner	3.5	
9	20.0	20.0	Utensil Scrub-Pad, Glove	4.3	
10	261.0	290.0	Flours & Pre-Mixes	4.0	
11	600.0	600.0	Luxury Chocolates, Gifts	2.2	

	description
0	This Product contains Garlic Oil that is known...
1	Each product is microwave safe (without lid), ...
2	A perfect gift for all occasions, be it your m...
3	Multipurpose container with an attractive desi...
4	Nivea Creme Soft Soap gives your skin the best...
5	Stay protected from contamination with Multipu...
6	Satinance multani matti is an excellent skin t...
7	70%Alcohol based is gentle of hand leaves skin...
8	An exclusive blend with Vitamin B7 Biotin, Hyd...
9	Scotch Brite Anti- Bacterial Scrub Pad thoroug...

```
10  Wheatgrass is a superfood potent health food w...
11  Enjoy a tin full of delicious butter cookies m...
```

Step 3: Get Description of the data in the DataFrame.

```
df.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
```

Step 4: Find Information about the DataFrame.

```
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 and Least sold products.

```
# Top sold products
df[['product','category','sub_category']].value_counts().head(10)
```

```
product                                 category
sub_category
Turmeric Powder/Arisina Pudi            Foodgrains, Oil & Masala  Masalas
& Spices        25
Extra Virgin Olive Oil                  Gourmet & World Food      Oils &
Vinegar          14
Soft Drink                              Beverages                 Energy &
Soft Drinks    12
Cow Ghee/Tuppa                          Foodgrains, Oil & Masala  Edible
Oils & Ghee      12
Colorsilk Hair Colour With Keratin      Beauty & Hygiene          Hair
Care               12
Coriander Powder                        Foodgrains, Oil & Masala  Masalas
& Spices        11
Ghee/Tuppa                              Foodgrains, Oil & Masala  Edible
Oils & Ghee      11
Olive Oil - Extra Virgin                Gourmet & World Food      Oils &
Vinegar          11
Powder - Coriander                      Foodgrains, Oil & Masala  Masalas
& Spices        11
Casting Creme Gloss Hair Color          Beauty & Hygiene          Hair
Care               10
Name: count, dtype: int64
```

```
# Top Least sold products.
df[['product','category','sub_category']].value_counts().head(10)
```

```
product                                 category
sub_category
Turmeric Powder/Arisina Pudi            Foodgrains, Oil & Masala  Masalas
& Spices        25
Extra Virgin Olive Oil                  Gourmet & World Food      Oils &
Vinegar          14
Soft Drink                              Beverages                 Energy &
Soft Drinks    12
Cow Ghee/Tuppa                          Foodgrains, Oil & Masala  Edible
Oils & Ghee      12
Colorsilk Hair Colour With Keratin      Beauty & Hygiene          Hair
Care               12
Coriander Powder                        Foodgrains, Oil & Masala  Masalas
& Spices        11
Ghee/Tuppa                              Foodgrains, Oil & Masala  Edible
Oils & Ghee      11
Olive Oil - Extra Virgin                Gourmet & World Food      Oils &
Vinegar          11
```

```
Powder - Coriander                  Foodgrains, Oil & Masala  Masalas
& Spices        11
Casting Creme Gloss Hair Color      Beauty & Hygiene          Hair
Care               10
Name: count, dtype: int64
```

Step 6: Measuring discount on a certain item.

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

```
0          0.000000
1          0.000000
2         52.400000
3         15.340909
4          0.000000
             ...
27550     20.000000
27551     10.000000
27552      0.000000
27553     20.000000
27554     44.992308
Length: 27555, dtype: float64
```

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

```
# Total number of rows
total_rows = len(df)
total_rows
```

```
27555
```

```
# Count all missing values
missing_values = df.isnull().sum().sum()
missing_values
```

```
8759
```

```
# Count missing values for each column
missing_values = df.isnull().sum()
missing_values
```

```
index               0
product             1
```

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

```
# Calculate the percantage of missing values
missing_percentage = (missing_values/total_rows)*100
missing_percentage
```

```
index             0.000000
product           0.003629
category          0.000000
sub_category      0.000000
brand             0.003629
sale_price        0.021775
market_price      0.000000
type              0.000000
rating           31.340954
description       0.417347
dtype: float64
```

```
missing_percentage.apply(lambda x: '{:.2f}%'.format(x))
```

```
index             0.00%
product           0.00%
category          0.00%
sub_category      0.00%
brand             0.00%
sale_price        0.02%
market_price      0.00%
type              0.00%
rating           31.34%
description       0.42%
dtype: object
```

Cleaning missing values

```
missing_values = df.isnull().sum()
missing_values
```

```
index               0
product             1
category            0
sub_category        0
brand               1
```

```
sale_price         6
market_price       0
type               0
rating          8636
description      115
dtype: int64
```

```
df1 = pd.DataFrame(df)
```

```
df1.loc[df1['product'].isna(),'product']='Unknown'
df1['product']
```

```
0                    Garlic Oil - Vegetarian Capsule 500 mg
1                                    Water Bottle - Orange
2                             Brass Angle Deep - Plain, No.2
3        Cereal Flip Lid Container/Storage Jar - Assort...
4                          Creme Soft Soap - For Hands & Body
                               ...
27550           Wottagirl! Perfume Spray - Heaven, Classic
27551                                              Rosemary
27552                           Peri-Peri Sweet Potato Chips
27553                              Green Tea - Pure Original
27554                          United Dreams Go Far Deodorant
Name: product, Length: 27555, dtype: object
```

```
df1.loc[df1['brand'].isna(),'brand']='Unknown'
df1['brand']
```

```
0                  Sri Sri Ayurveda
1                         Mastercook
2                                Trm
3                             Nakoda
4                              Nivea
                     ...
27550                          Layerr
27551                        Puramate
27552                          FabBox
27553                          Tetley
27554    United Colors Of Benetton
Name: brand, Length: 27555, dtype: object
```

```
df1.loc[df1['sale_price'].isna(),'sale_price']='Unknown'
df1['sale_price']
```

```
C:\Users\g k\AppData\Local\Temp\ipykernel_9776\359104473.py:1:
FutureWarning: Setting an item of incompatible dtype is deprecated and
will raise an error in a future version of pandas. Value 'Unknown' has
dtype incompatible with float64, please explicitly cast to a
compatible dtype first.
  df1.loc[df1['sale_price'].isna(),'sale_price']='Unknown'
```

```
0         220.0
1         180.0
2         119.0
3         149.0
4         162.0
          ...
27550     199.2
27551      67.5
27552     200.0
27553     396.0
27554    214.53
Name: sale_price, Length: 27555, dtype: object
```

```
df1.loc[df1['rating'].isna(),'rating']='Unknown'
df1['rating']
```

```
C:\Users\g k\AppData\Local\Temp\ipykernel_9776\2186118322.py:1:
FutureWarning: Setting an item of incompatible dtype is deprecated and
will raise an error in a future version of pandas. Value 'Unknown' has
dtype incompatible with float64, please explicitly cast to a
compatible dtype first.
  df1.loc[df1['rating'].isna(),'rating']='Unknown'
```

```
0        4.1
1        2.3
2        3.4
3        3.7
4        4.4
        ...
27550    3.9
27551    4.0
27552    3.8
27553    4.2
27554    4.5
Name: rating, Length: 27555, dtype: object
```

```
df1.loc[df1['description'].isna(),'description']='Unknown'
df1['description']
```

```
0        This Product contains Garlic Oil that is known...
1        Each product is microwave safe (without lid), ...
2        A perfect gift for all occasions, be it your m...
3        Multipurpose container with an attractive desi...
4        Nivea Creme Soft Soap gives your skin the best...
                              ...
27550    Layerr brings you Wottagirl Classic fragrant b...
27551    Puramate rosemary is enough to transform a dis...
27552    We have taken the richness of Sweet Potatoes (...
27553    Tetley Green Tea with its refreshing pure, ori...
```

```
27554    The new mens fragrance from the United Dreams ...
Name: description, Length: 27555, dtype: object
```

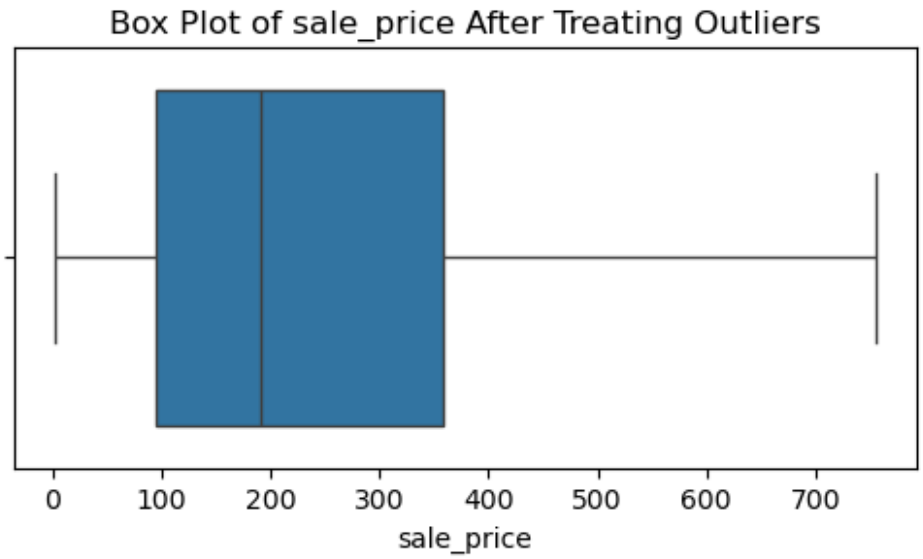
```
# check clear missing value
df1.isnull().sum()
```

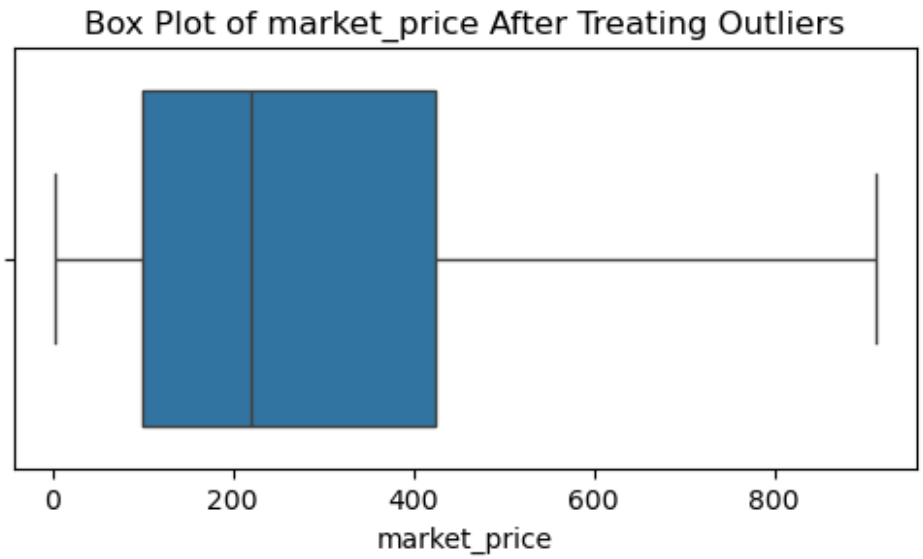
```
index           0
product         0
category        0
sub_category    0
brand           0
sale_price      0
market_price    0
type            0
rating          0
description     0
dtype: int64
```

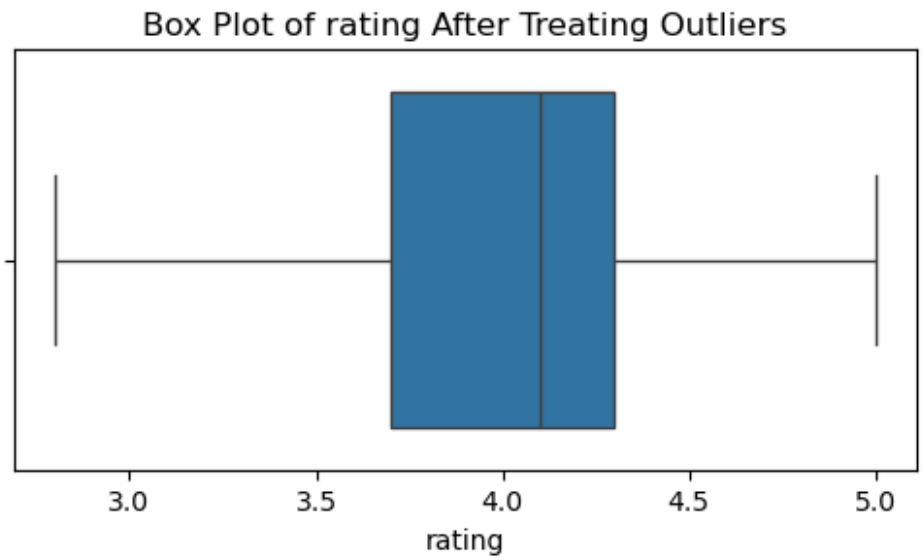
Step 8: Find out the outliers form the dataset according to the columns

```
def replace_outliers(df, column):
    Q1 = df[column].quantile(.25)
    Q3 = df[column].quantile(.75)
    IQR = Q3-Q1

    # Define lower and upper bounds
    lower_bound = max(Q1 - 1.5 * IQR,0)
    upper_bound = Q3 + 1.5 * IQR

    # Replace outliers with the column median
    df[column]=df[column].apply(lambda x: lower_bound if x <
lower_bound else (upper_bound if x > upper_bound else x))

    return df

df = replace_outliers(df, 'sale_price')
df = replace_outliers(df, 'market_price')
df = replace_outliers(df, 'rating')

# List of specific columns to plot
df3 = ['sale_price', 'market_price', 'rating']

# Plot box plots for the selected columns
plt.figure(figsize=(5, len(df3) * 3))  # Adjust figure size
for i, column in enumerate(df3):
    plt.subplot(len(df3), 1, i + 1)
    sns.boxplot(x=df[column])
```

```
    plt.title(f'Box Plot of {column} After Treating Outliers')
    plt.tight_layout()

plt.show()
```


Box Plot of sale_price After Treating Outliers
0
100
200
300
400
500
600
700
sale_price


Box Plot of market_price After Treating Outliers
0
200
400
600
800
market_price


Box Plot of rating After Treating Outliers
3.0
3.5
4.0
4.5
5.0
rating

Findings and Insights

1st box plot represents the sale_price distribution after treating outliers, showing most data within the interquartile range (100–400) and a total range of approximately 0 to 700. The absence of points beyond the whiskers indicates successful outlier removal or adjustment

2nd box plot represents the market_price distribution after treating outliers, showing most data within the interquartile range (0-600) and a total range of approximately 0 to 800. The absence of points beyond the whiskers indicates successful outlier removal or adjustment

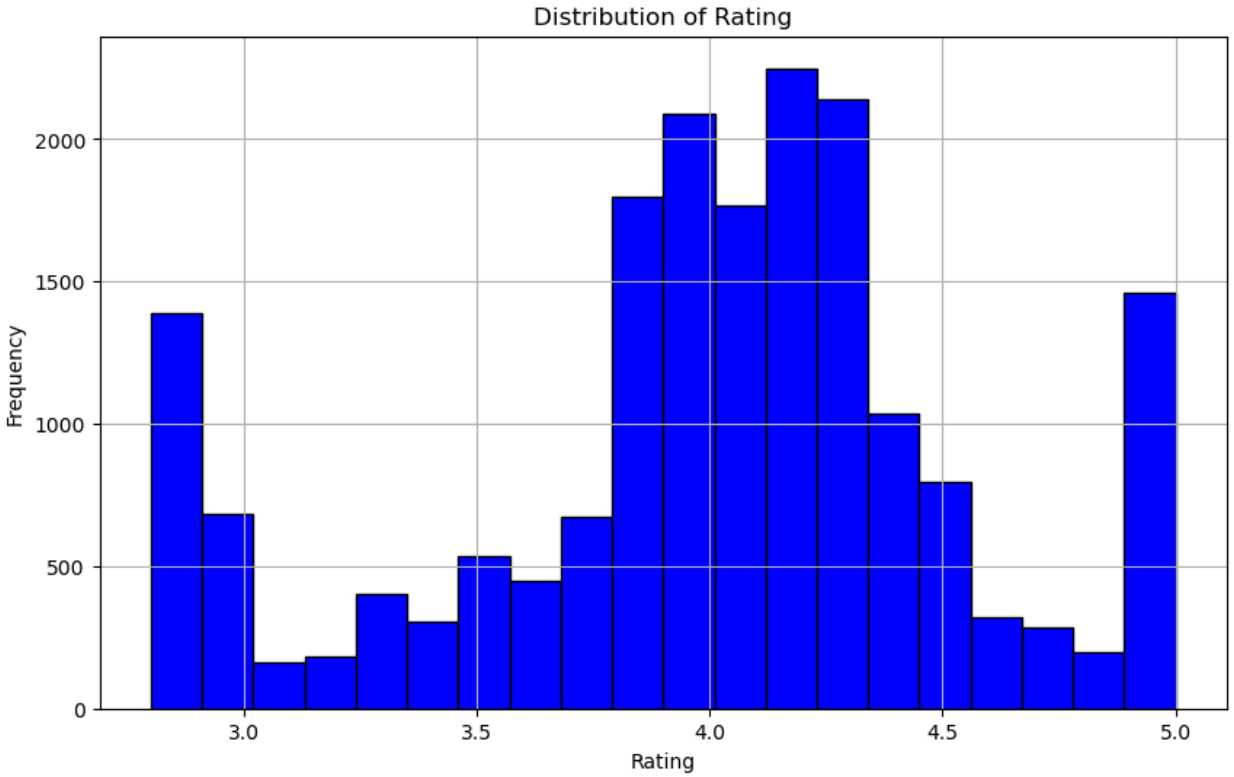
2nd box plot represents the market_price distribution after treating outliers, showing most data within the interquartile range (3.5-4.5) and a total range of approximately 0 to 5.0. The absence of points beyond the whiskers indicates successful outlier removal or adjustment

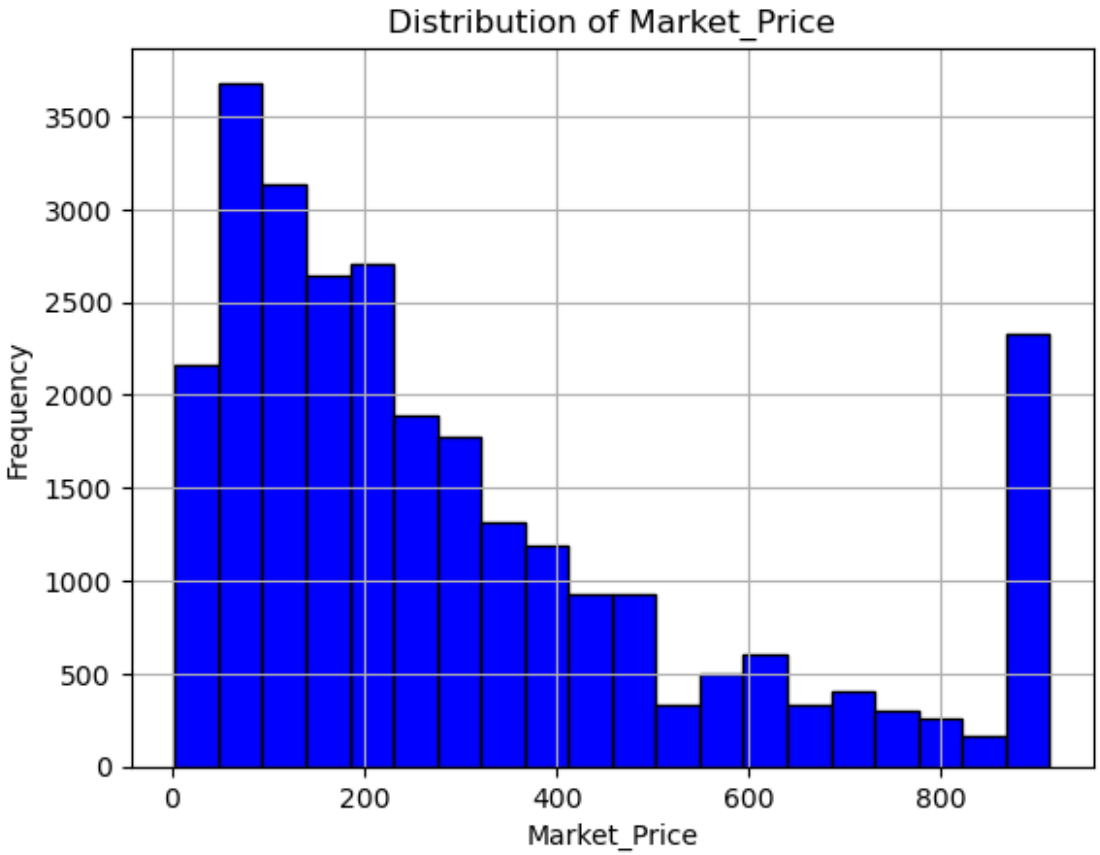
Step 9: Create Plots or visualizations.

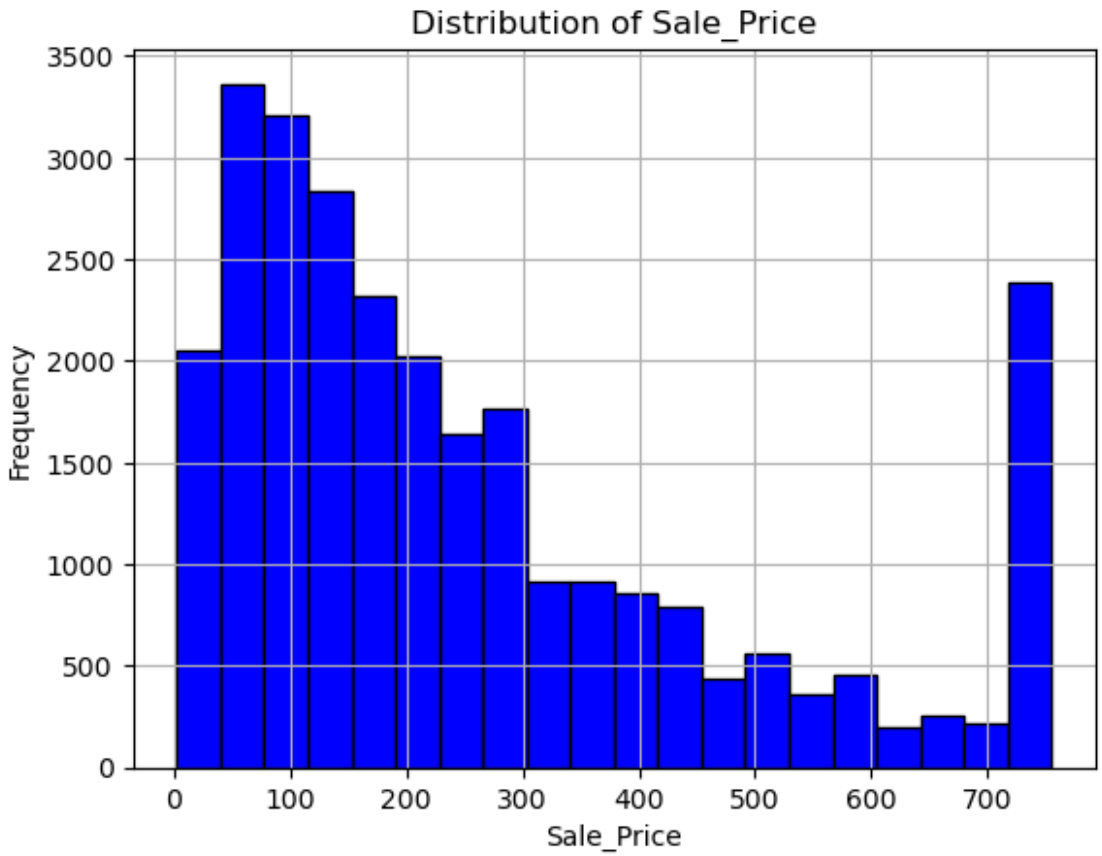
Histograms for Numeric Vlaues

- In this dataset, there are Numerice Values like Sale_price,Market_price and Rating

```
# Plotting histograms for each column separately
plt.figure(figsize=(10,6))

# Histogram for 'rating'
plt.subplot(1,1,1)
plt.hist(df['rating'], bins=20, color='Blue', edgecolor='black')
plt.title('Distribution of Rating')
plt.xlabel('Rating')
plt.ylabel('Frequency')
plt.grid(True)
plt.show()

# Histogram for 'market_price'
plt.subplot(1,1,1)
plt.hist(df['market_price'], bins=20, color='Blue', edgecolor='black')
plt.title('Distribution of Market_Price')
plt.xlabel('Market_Price')
plt.ylabel('Frequency')
plt.grid(True)
plt.show()

# Histogram for 'sale_price'
plt.subplot(1,1,1)
plt.hist(df['sale_price'], bins=20, color='Blue', edgecolor='black')
plt.title('Distribution of Sale_Price')
plt.xlabel('Sale_Price')
plt.ylabel('Frequency')
plt.grid(True)
plt.show()
```


Distribution of Rating
Frequency
Rating
0
500
1000
1500
2000
3.0
3.5
4.0
4.5
5.0


Distribution of Market_Price
Frequency
Market_Price
0
500
1000
1500
2000
2500
3000
3500
0
200
400
600
800



Findings and insights

1st histogram Plot displays the distribution of ratings, showing a peak concentration between 4.0 and 4.5, with fewer ratings at the extremes (around 3.0 and 5.0). The data indicates a generally high tendency toward favorable ratings.

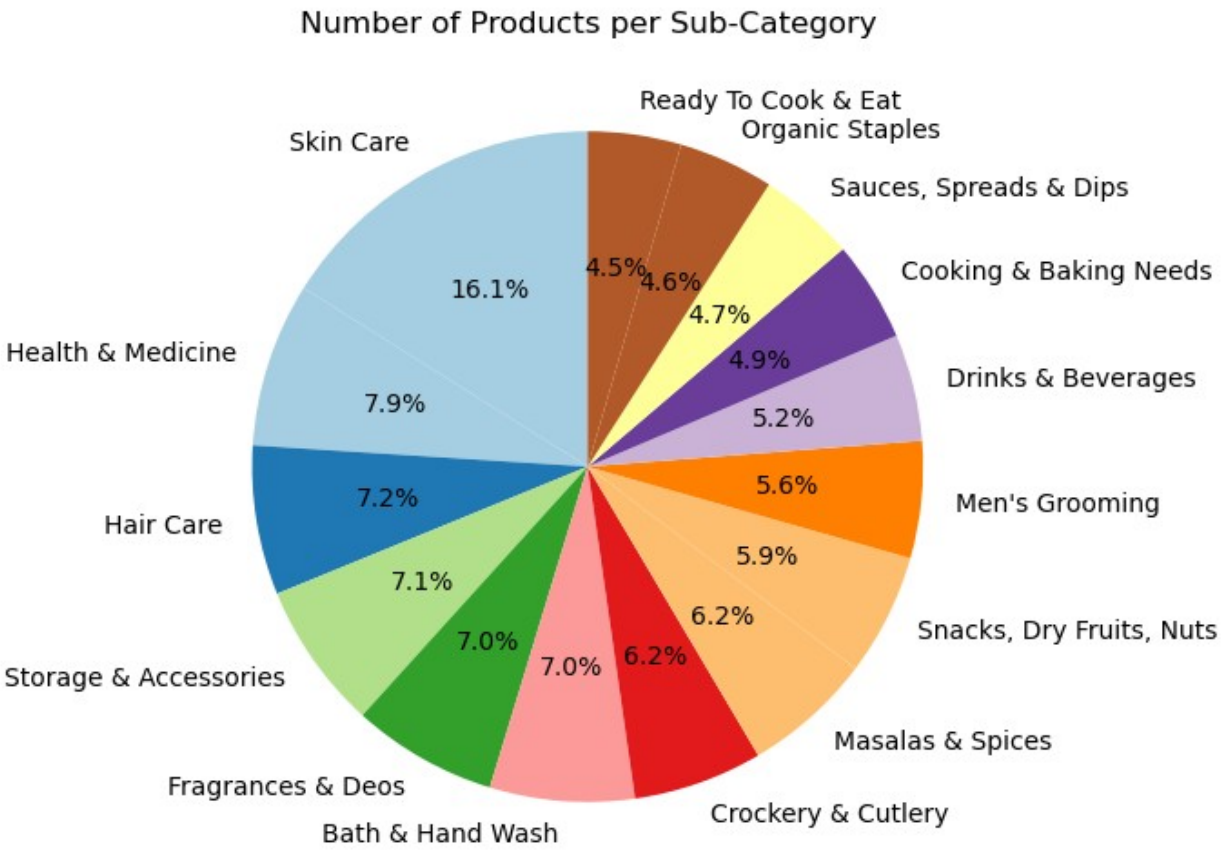
2nd histogram Plot shows the distribution of market prices. Most values are concentrated below 200, indicating that lower prices are more common. There is also a noticeable spike at around 800, suggesting a specific price range with higher frequency.

3rd histogram plot illustrates the distribution of sale prices. The majority of sale prices are concentrated below 200, indicating that lower-priced items are more common. There is a steady decline in frequency as the sale price increases beyond 200. A significant spike is observed around 700, which could indicate a common price point for certain products. The distribution highlights a right-skewed trend with an outlier range.

Pie Plot for Categorical Variables

```
# Get top 15 sub_category counts
sub_category_counts = df['sub_category'].value_counts().head(15)

# Plotting the number of products per sub_category as a pie chart
```

```
plt.figure(figsize=(12, 6))
sub_category_counts.plot(
    kind='pie',
    autopct='%1.1f%%',
    colors=plt.cm.Paired(np.linspace(0, 1, len(sub_category_counts))),
# Generate colors
    startangle=90
)

# Add title and remove y-label
plt.title('Number of Products per Sub-Category')
plt.ylabel('')  # Remove the default y-label
plt.show()
```



Findings and insights

This is a pie chart showing the distribution of the number of products across various sub-categories. Each slice of the pie represents a specific sub-category, with its size proportional to the percentage of products it comprises.

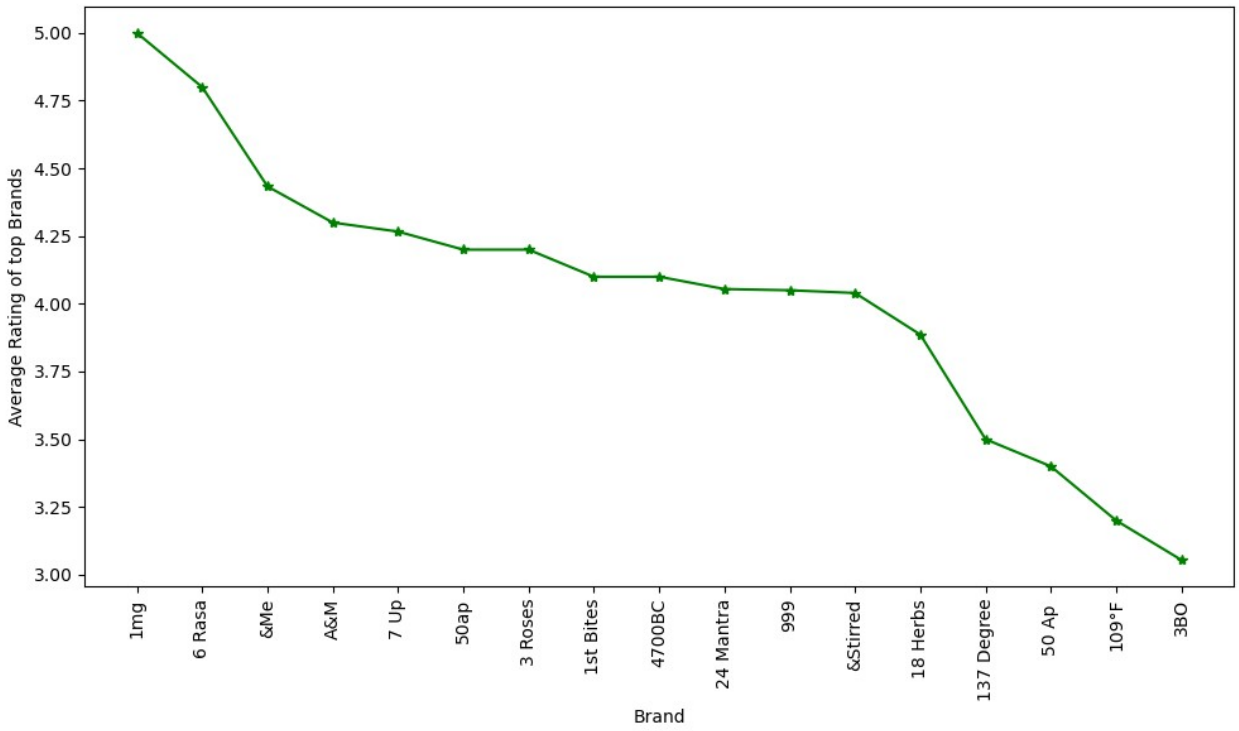
Key observations:

1. **Skin Care** is the largest sub-category, making up 16.1% of the total products.
2. **Health & Medicine** accounts for 7.9%, followed by **Hair Care** (7.2%) and **Storage & Accessories** (7.0%).
3. Smaller sub-categories include **Ready To Cook & Eat** (4.5%), **Organic Staples** (4.6%), and **Cooking & Baking Needs** (4.7%).

Other notable percentages:

- Drinks & Beverages: 5.2%
- Men's Grooming: 5.6%
- Snacks, Dry Fruits, Nuts: 5.9%
- Masalas & Spices: 6.2%

```
brand_ratings = df.groupby('brand')
['rating'].mean().reset_index().head(20)
brand_ratings_sorted = brand_ratings.sort_values(by =
'rating',ascending=False)
plt.figure(figsize=(10,6))
plt.plot(brand_ratings_sorted['brand'],brand_ratings_sorted['rating'],
marker='*',color='green')
plt.xticks(rotation=90)
plt.xlabel('Brand')
plt.ylabel('Average Rating of top Brands')
plt.tight_layout()
plt.show()
```



Key Observations:

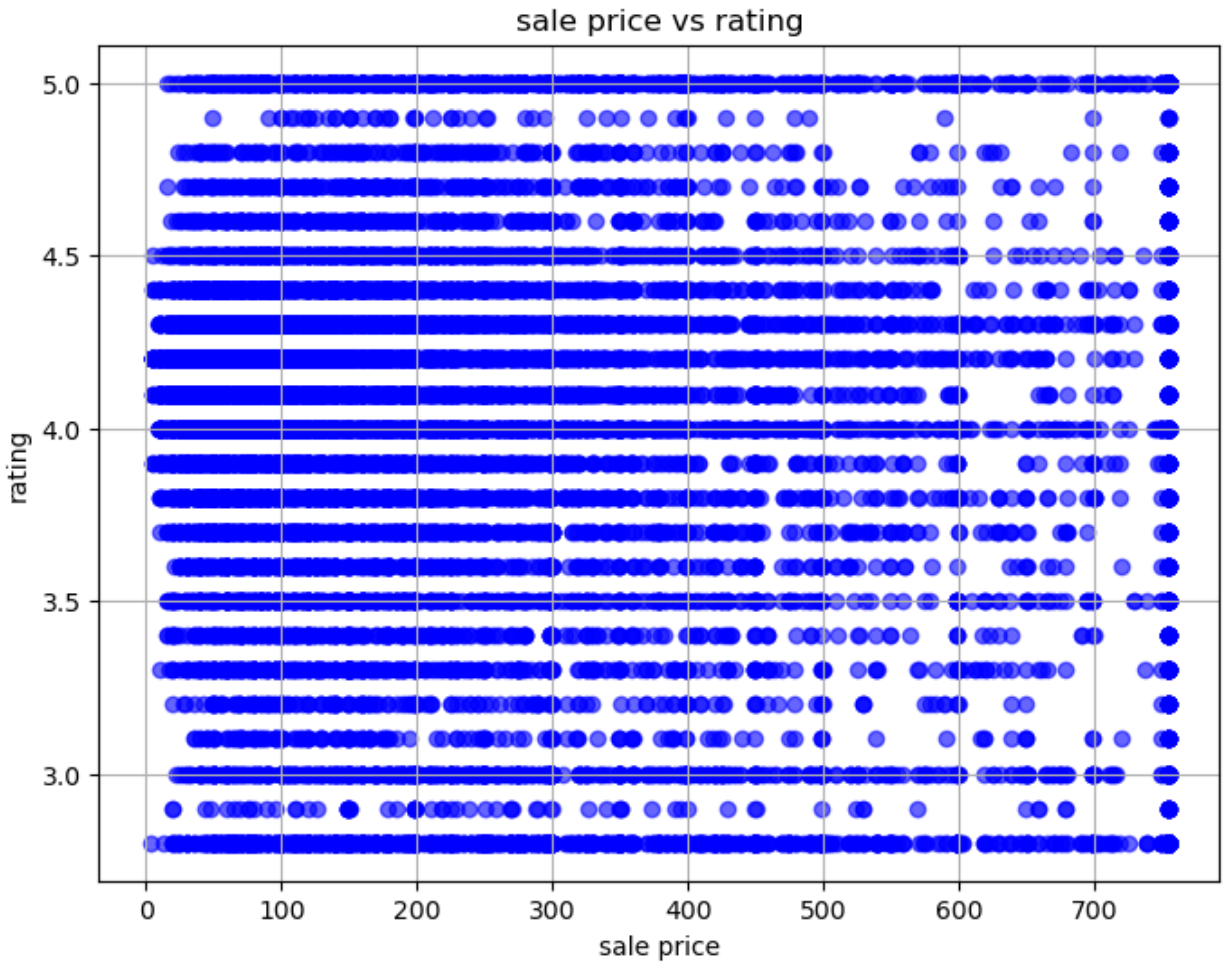
1. **Highest Rated Brands**:
   - The brand "1mg" has the highest average rating, close to 5.0.
   - "6 Rasa" follows with a slightly lower rating.
2. **Steady Decline**:
   - The ratings gradually decline across the brands as we move along the x-axis.
   - Brands like "6Me", "A&M", and "7Up" occupy the middle range of the chart with ratings around 4.0–4.5.
3. **Lowest Rated Brands**:
   - Brands such as "50 Ap", "109°F", and "3BO" have the lowest ratings, around or below 3.5.

Insights:

- Brands with higher ratings (e.g., "1mg" and "6 Rasa") are likely associated with better customer satisfaction or quality.
- The steady decline in ratings highlights a notable variation in customer experiences among these brands

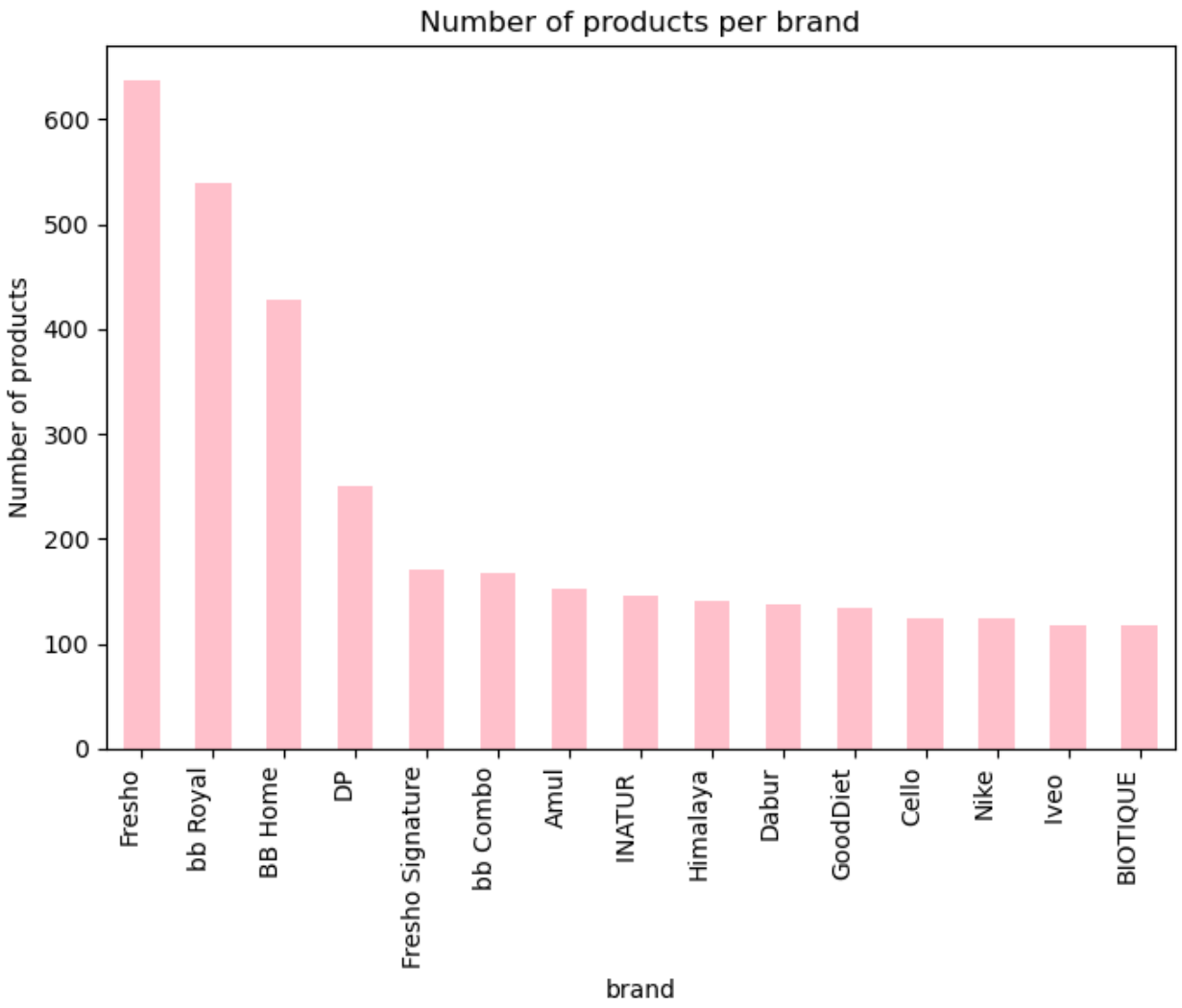
Scatter plot of sale price vs rating

```
plt.figure(figsize = (8,6))
plt.scatter(df['sale_price'],df['rating'],color='blue',alpha=0.6)
plt.title('sale price vs rating')
plt.xlabel('sale price')
plt.ylabel('rating')
plt.grid(True)
plt.show()
```



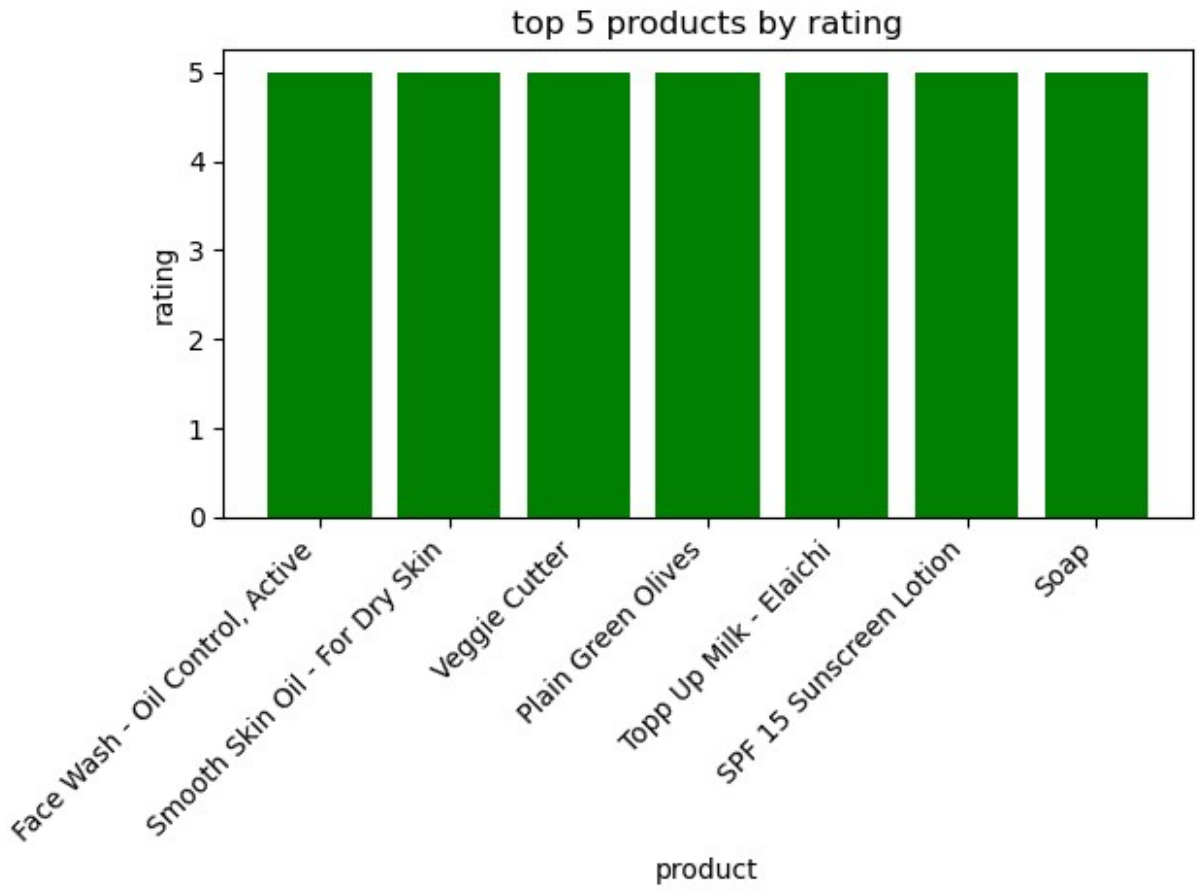
Count the number of products for top 15 brands

```
brand_counts = df['brand'].value_counts().head(15)
# plotting the number of products per brand
plt.figure(figsize=(7,6))
brand_counts.plot(kind='bar',color='pink')
plt.title('Number of products per brand')
plt.xlabel('brand')
plt.ylabel('Number of products')
plt.xticks(rotation=90, ha='right')
plt.tight_layout()
plt.show()
```



Top 5 Products by Ratin

```
top_5_products = df.nlargest(7, 'rating')[['product','rating']]
plt.bar(top_5_products['product'],top_5_products['rating'],color='gree
n')
plt.xlabel('product')
plt.ylabel('rating')
plt.title('top 5 products by rating')
plt.xticks(rotation=45, ha='right')
plt.tight_layout()
plt.show()
```




```
# Plot density plots for sale_price
plt.figure(figsize=(10,6))
sns.kdeplot(df['sale_price'], shade=True, color='blue',
label='sale_price')

# Adding title and labels
plt.title('Density plot of sale_price', fontsize=16)
plt.xlabel('Value', fontsize=12)
plt.ylabel('Density', fontsize=12)
# Adding a legend
plt.legend()

# Show the plot
plt.tight_layout()
plt.show()
```

```
C:\Users\g k\AppData\Local\Temp\ipykernel_9776\1216780435.py:3:
FutureWarning:

`shade` is now deprecated in favor of `fill`; setting `fill=True`.
This will become an error in seaborn v0.14.0; please update your code.
```

```
  sns.kdeplot(df['sale_price'], shade=True, color='blue',
label='sale_price')
```

Key Observations:

1. **Primary Peak**:

- The highest density is observed around a sale price close to 100-200. This suggests that most items are sold in this price range.

1. **Right Skewness**:

- The plot shows a long tail toward higher sale prices, indicating fewer high-priced items. This is typical in datasets where most products are moderately priced, with fewer premium products.

1. **Secondary Peak**:

- A smaller peak is observed around the sale price of 800. This might represent a specific category of premium or high-value items sold at a higher price point.

1. **Low Sale Prices**:

The density is low near a sale price of 0, indicating very few items are sold at negligible prices.

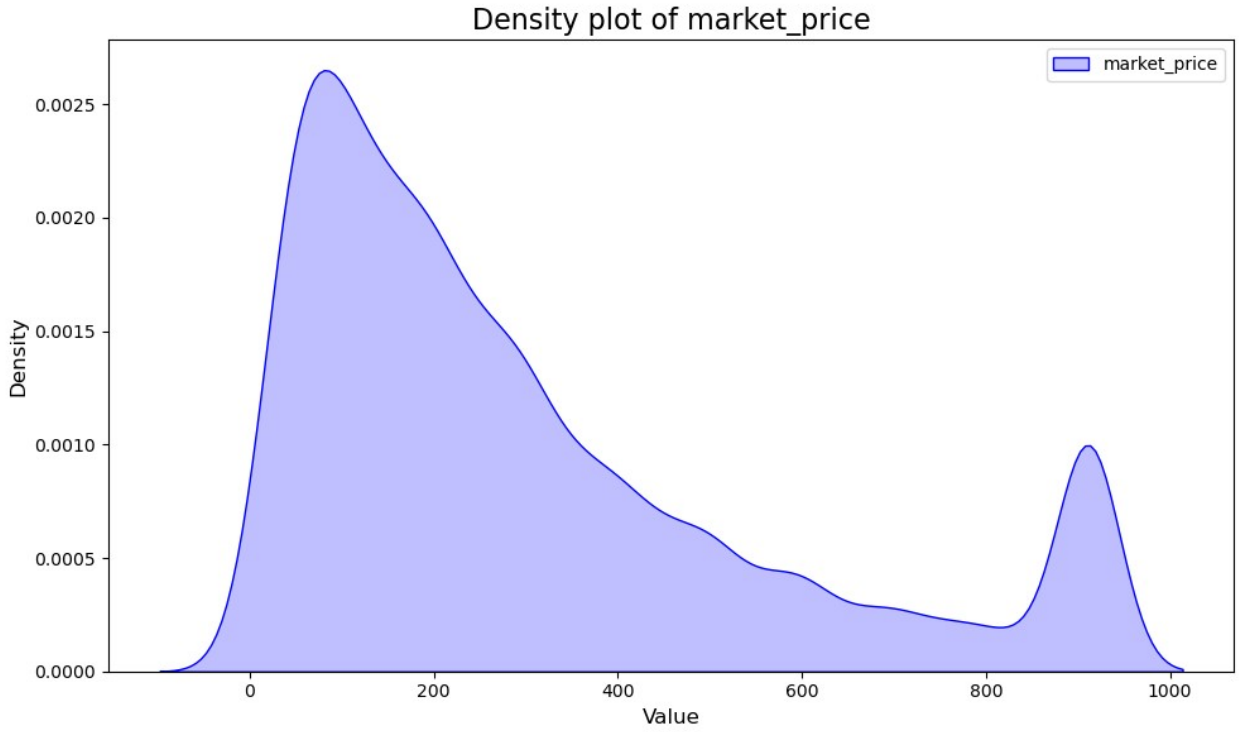
Insights:

This plot is useful for understanding pricing strategies or inventory distribution:

- The primary peak suggests the optimal pricing range for maximum sales volume.
- The secondary peak could be investigated for premium products, which might offer higher profit margins. Would you like to dive deeper into specific areas or correlations with other variables?

```
# Plot density plots for sale_price
plt.figure(figsize=(10,6))
sns.kdeplot(df['market_price'], shade=True, color='blue',
label='market_price')

# Adding title and labels
plt.title('Density plot of market_price', fontsize=16)
plt.xlabel('Value', fontsize=12)
plt.ylabel('Density', fontsize=12)
# Adding a legend
plt.legend()

# Show the plot
plt.tight_layout()
plt.show()
```

```
C:\Users\g k\AppData\Local\Temp\ipykernel_9776\1451019416.py:3:
FutureWarning:

`shade` is now deprecated in favor of `fill`; setting `fill=True`.
This will become an error in seaborn v0.14.0; please update your code.

  sns.kdeplot(df['market_price'], shade=True, color='blue',
label='market_price')
```



Key Observations:

1. **Primary Peak**:

- The highest density is observed around a sale price close to 100-200. This suggests that most items are sold in this price range.

1. **Right Skewness**:

- The plot shows a long tail toward higher sale prices, indicating fewer high-priced items. This is typical in datasets where most products are moderately priced, with fewer premium products.

1. **Secondary Peak**:

- A smaller peak is observed around the sale price of 800. This might represent a specific category of premium or high-value items sold at a higher price point.

1. **Low Sale Prices**:

The density is low near a sale price of 0, indicating very few items are sold at negligible prices.

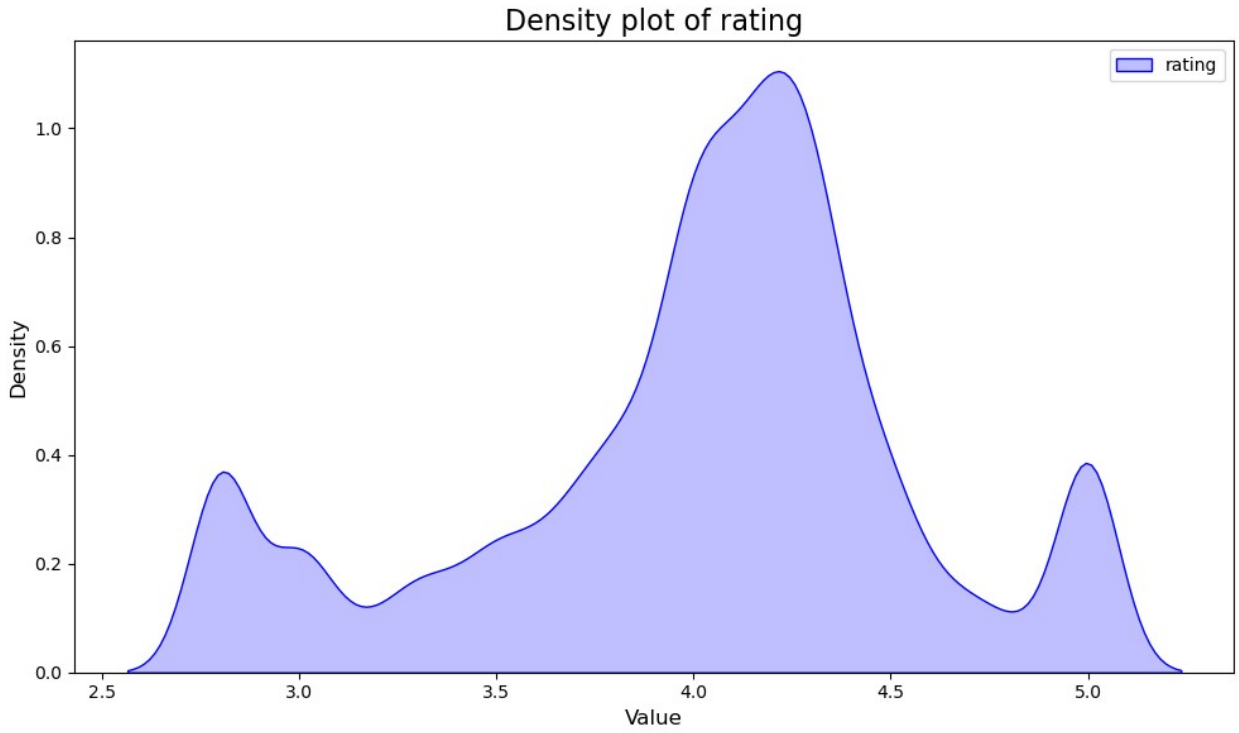
Insights:

This plot is useful for understanding pricing strategies or inventory distribution:

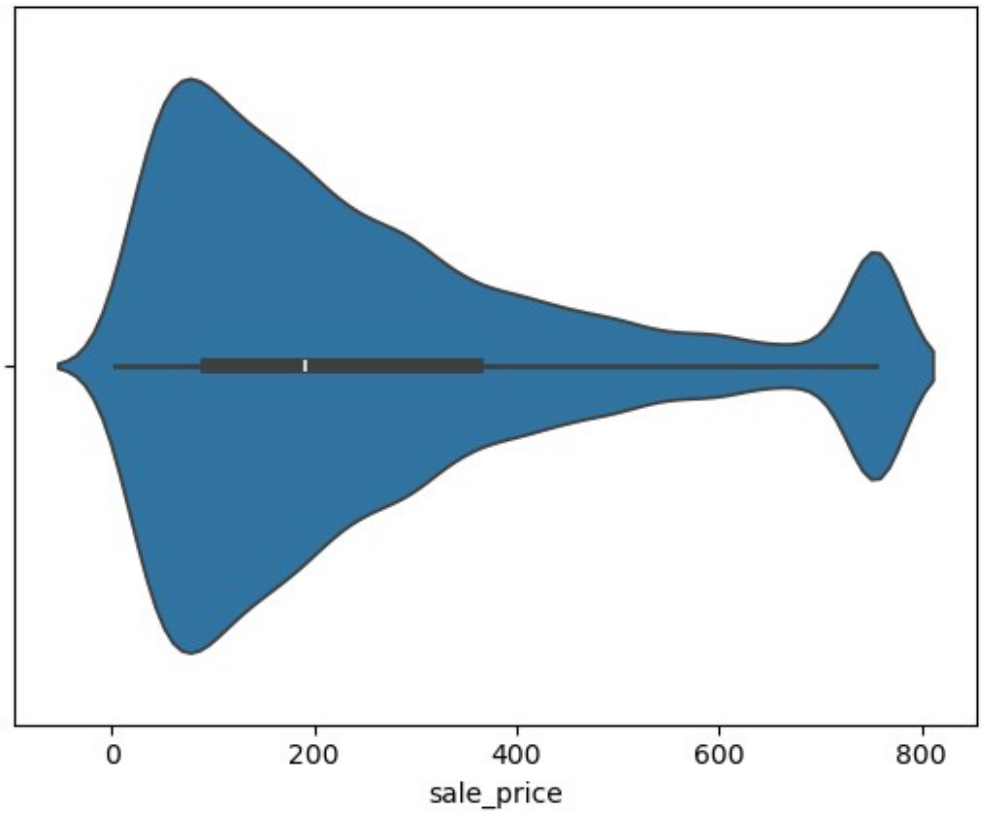
- The primary peak suggests the optimal pricing range for maximum sales volume.
- The secondary peak could be investigated for premium products, which might offer higher profit margins. Would you like to dive deeper into specific areas or correlations with other variables?

```
# Plot density plots for sale_price
plt.figure(figsize=(10,6))
sns.kdeplot(df['rating'], shade=True, color='blue', label='rating')

# Adding title and labels
plt.title('Density plot of rating', fontsize=16)
plt.xlabel('Value', fontsize=12)
plt.ylabel('Density', fontsize=12)
# Adding a legend
plt.legend()

# Show the plot
plt.tight_layout()
plt.show()
```

```
C:\Users\g k\AppData\Local\Temp\ipykernel_9776\176335829.py:3:
FutureWarning:

`shade` is now deprecated in favor of `fill`; setting `fill=True`.
This will become an error in seaborn v0.14.0; please update your code.

  sns.kdeplot(df['rating'], shade=True, color='blue', label='rating')
```



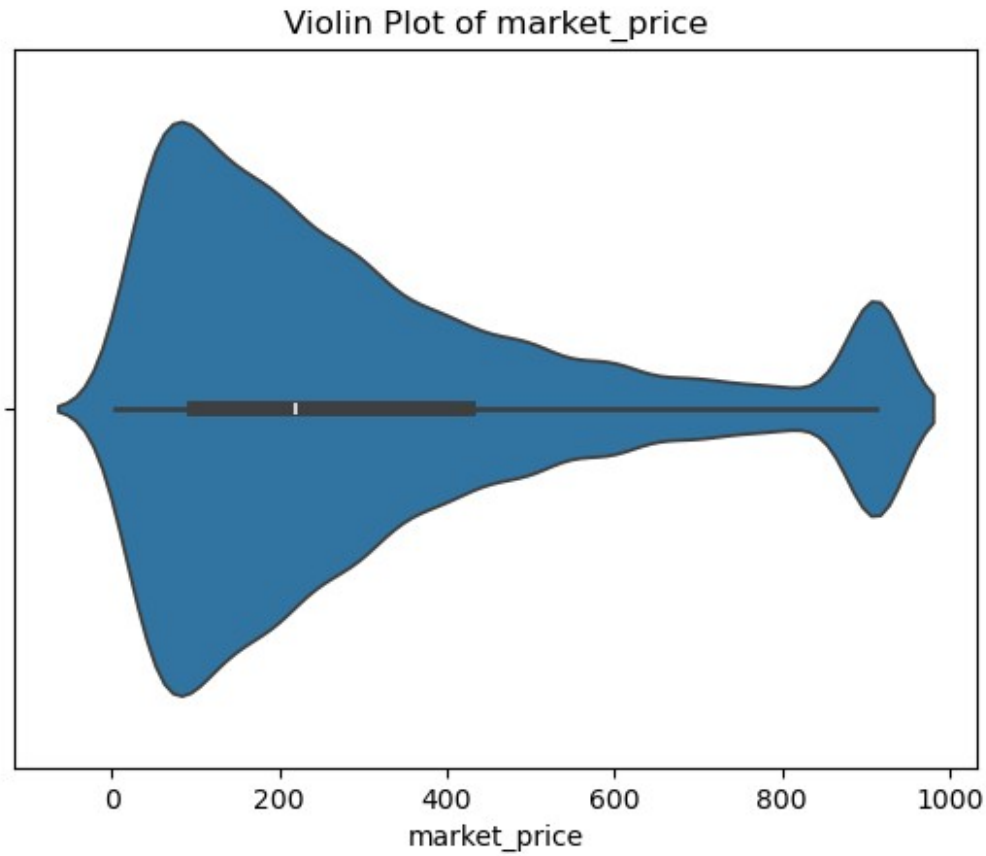

```
sns.violinplot(x=df['sale_price'])
plt.show()
```



This is a violin plot, which combines a boxplot and a kernel density estimate to provide insights into the distribution of a dataset. The plot seems to represent the distribution of sale prices:

X-axis (sale price): Indicates the numerical values of the sale price. **Shape (violin shape)**: Shows the probability density of the data at different sale price values. Wider sections represent more data points at those values, while narrower sections represent fewer data points. **Boxplot in the center**: Highlights key summary statistics: - The white dot represents the median. - The thick bar around the median shows the interquartile range (IQR). - The thin lines (whiskers) extend to the range of the data, excluding potential outliers. This visualization suggests that the sale price has an uneven distribution, with some skewness or outliers present, as indicated by the density spread and the whiskers' extension. The tail on the right might represent higher-priced items or properties

```
sns.violinplot(x=df['market_price'])
plt.title('Violin Plot of market_price')
plt.show()
```

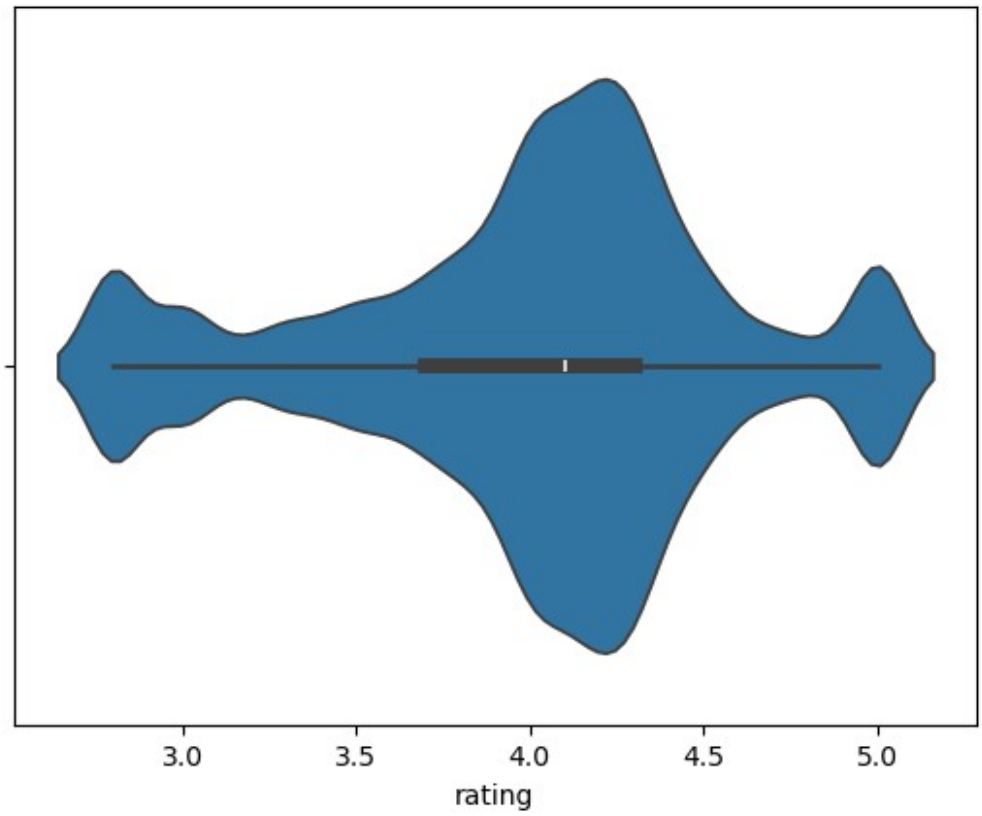


This is a violin plot, which combines a boxplot and a kernel density estimate to provide insights into the distribution of a dataset. The plot seems to represent the distribution of sale prices:

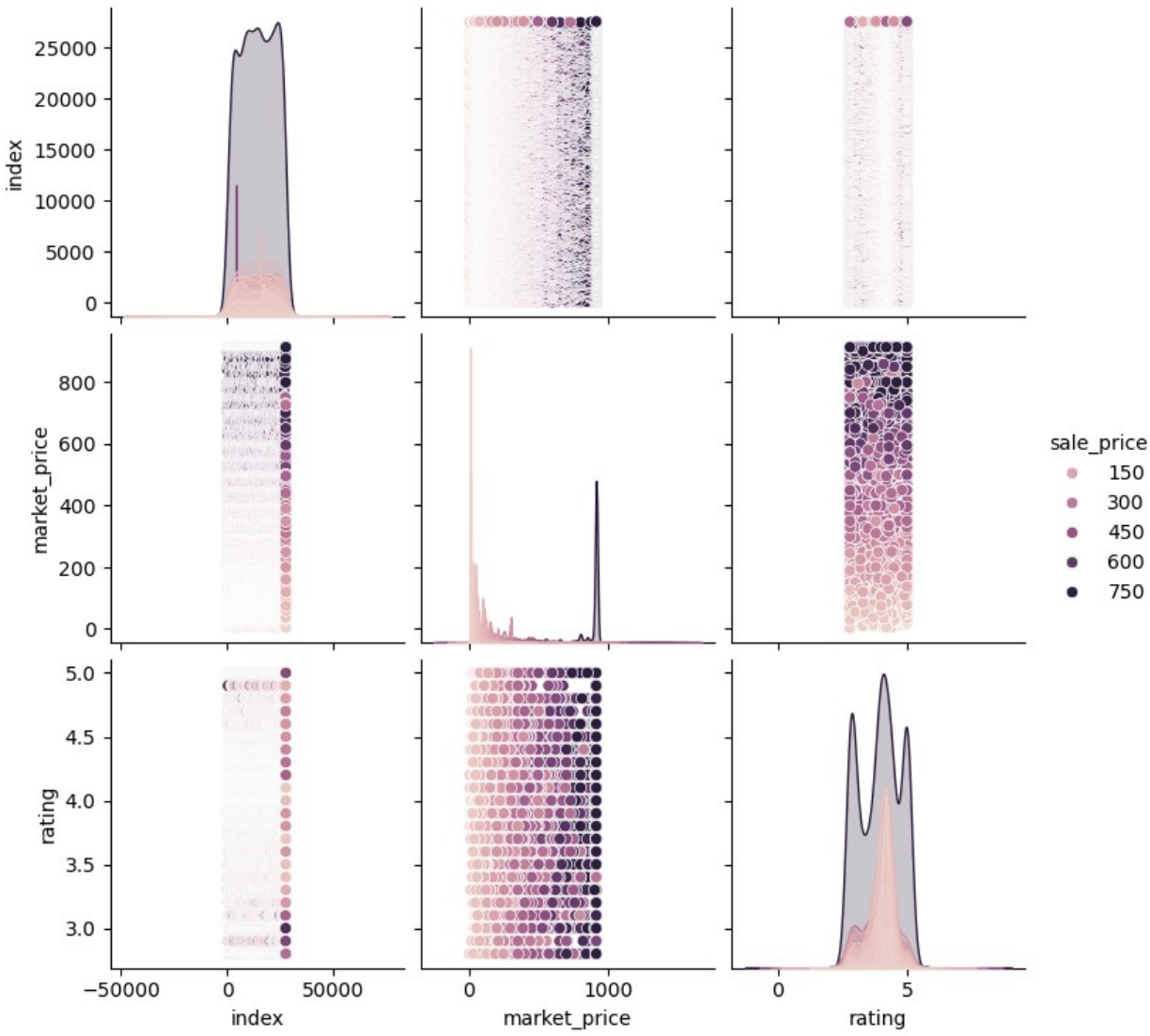
- **X-axis (sale price)**: Indicates the numerical values of the sale price.
- **Shape (violin shape)**: Shows the probability density of the data at different sale price values. Wider sections represent more data points at those values, while narrower sections represent fewer data points.
- **Boxplot in the center**: Highlights key summary statistics:
  - The white dot represents the median.
  - The thick bar around the median shows the interquartile range (IQR).
  - The thin lines (whiskers) extend to the range of the data, excluding potential outliers. This visualization suggests that the sale price has an uneven distribution, with some skewness or outliers present, as indicated by the density spread and the whiskers' extension. The tail on the right might represent higher-priced items or properties

```
sns.violinplot(x=df['rating'])
```

```
<Axes: xlabel='rating'>
```




```
# Pair Plot
sns.pairplot(df, hue='sale_price')
plt.show()
```



This image is a **pair plot**, which visualizes relationships between multiple variables in a dataset. Here's a breakdown of its components:

Axes and Variables

- The plot includes three variables: index, market_price, and rating.
- Each diagonal plot represents the distribution (e.g., histograms or KDE) of a single variable.
- Off-diagonal scatterplots show pairwise relationships between variables.

Observations

1. Diagonal Plots:
   - index: The distribution is dense within a certain range.
   - market_price: There's a sharp spike, likely indicating a concentrated price range.
   - rating: Bimodal distribution, suggesting two prominent rating clusters.

2. **Scatterplots**:
    - index vs. market_price: The data is spread vertically, indicating little correlation.
    - index vs. rating: Points are evenly spread with no strong relationship.
    - market_price vs. rating: There's some clustering, possibly suggesting relationships between high market prices and certain ratings.
3. **Color Coding**:

- Points are colored by sale_price values, with darker colors representing higher sale prices.
- Higher sale_price appears clustered in specific regions of the scatterplots, indicating potential interactions with other variables.

Insights
This pair plot is useful for:

- Identifying correlations or lack thereof between variables. Detecting patterns, clusters, or outliers in the data. Observing how sale_price (as the color intensity) varies with the other variables

Insights:¶

- Stable claim patterns over time: The frequency of claims remains relatively consistent across the "vintage" period (0–300 days), with only minor fluctuations.
- Low response for claims: Very few claims are marked "Yes," while the majority are "No," indicating a low claim response rate throughout the period.

Recommendations:¶

- Engagement strategies: Introduce initiatives to encourage more claim responses over time, possibly through customer engagement or outreach.
- Monitor early claims: There is a slight increase in claims around the early stages (first 50 days). Investigate why customers are more likely to claim during this period for potential improvements

Conclusion

Through the EDA of the vehicle insurance dataset, we uncovered actionable insights into policyholder demographics, driving behavior, and insurance claim trends. These insights can be utilized to design data-driven strategies, enhance customer segmentation, and mitigate risks effectively. This analysis provides a foundation for further predictive modeling and decision-making processes within the organization