PROJECT REPORT

Dissertation submitted in fulfilment of the requirements for the Degree of

BACTHELOR OF TECHNOLOGY

in

COMPUTER SCIENCE AND ENGINEERING

By

Avula Karthik

Registration number

12212224

K22UGB45

Submitted to – Ved Prakash Chaubey

School of Computer Science and Engineering

Lovely Professional University

Phagwara, Punjab (India)

Sep,2024

Supervisor Certificate

Lovely Professional University
School of Computer Science and Engineering

Certificate of Supervision

This is to certify that the project report titled "Amazon Product Sales Analysis" has been carried out by Avula Karthik (Registration No. 12212224) under my supervision in partial fulfillment of the requirements for the award of the degree of Bachelor of Technology in Computer Science and Engineering at Lovely Professional University, Phagwara, Punjab.

The project embodies the student's original work and has not been submitted to any other institution or university for the award of any degree or diploma. This report provides an insightful analysis of Amazon's product sales data, with particular emphasis on understanding consumer behavior, product categories, pricing strategies, and customer satisfaction, based on exploratory data analysis (EDA) and visualization.

I hereby approve this project report and consider it worthy of submission for evaluation.

Ved Prakash Chaubey.
School of Computer Science and Engineering
Lovely Professional University
Signature:________

Acknowledgment

I am grateful to many people who contributed to the successful completion of this project report, "Amazon Product Sales Analysis."

First, I would like to express my sincere gratitude to Ved Prakash Chaubey, my project supervisor, for their guidance, valuable feedback, and continuous support throughout the development of this project. Their expertise and encouragement were instrumental in overcoming challenges and enhancing the quality of my analysis.

I am also thankful to the faculty and staff of the School of Computer Science and Engineering, Lovely Professional University, for their assistance and for creating a learning environment that allowed me to carry out this project successfully. Additionally, I am grateful for the online resources and datasets available on platforms like Kaggle, which provided me with the necessary data to conduct meaningful analysis.

Finally, I extend my appreciation to my family and friends for their patience, encouragement, and support during the completion of this project.

Thank you all for your invaluable support and assistance.
Avula Karthik
Registration No. 12212224
Bachelor of Technology, Computer Science and Engineering

Table of Contents

Title: Amazon product sales Analysis

Abstract:

This project involves analyzing Amazon product data to extract meaningful insights regarding product categories, pricing, and ratings, with a focus on understanding consumer behavior. The dataset contains over 550,000 product entries with information such as product names, categories, prices (actual and discounted), and customer ratings.
The analysis explores price variations across product categories, identifies pattens in customer ratings, and investigates the relationship between product prices and ratings. Data cleaning includes handling missing values and converting price and rating information into numeric formats.
Exploratory data analysis (EDA) is conducted to visualize the distribution of prices, ratings, and the frequency of different categories. Key findings include significant price differences between product categories, variations in customer rating distributions, and interesting patterns between product price reductions and customer sentiment as reflected in ratings.
This analysis helps in uncovering trends and patterns in the Amazon marketplace, providing valuable insights into consumer preferences, pricing strategies, and product performance.

The analysis begins with data cleaning and preprocessing to ensure quality and consistency in the dataset, addressing missing values and standardizing numerical formats for prices and ratings. Exploratory Data Analysis (EDA) is then conducted to gain an overview of the distribution and patterns in the data. Key aspects include examining the price range across different categories, identifying highly-rated products, and exploring the impact of discounts on customer ratings. Statistical tests, such as correlation analysis and ANOVA, are employed to identify significant relationships between variables like price, discount, and ratings.

Through various visualizations and statistical analyses, this project uncovers actionable insights, such as the categories with the highest customer satisfaction and the pricing strategies that best resonate with consumers. The findings indicate that specific product categories, like electronics and apparel, demonstrate unique trends in customer ratings and price sensitivity. Additionally, certain brands show consistently high ratings, providing potential benchmarks for customer satisfaction.

In conclusion, the analysis reveals valuable trends and patterns that can aid Amazon sellers and other e-commerce businesses in understanding consumer preferences, fine-tuning pricing strategies, and enhancing customer satisfaction. Future research could expand upon this analysis by incorporating textual sentiment analysis of product reviews or by exploring sales performance across different time periods, helping businesses to keep pace with changing consumer trends.

Problem Statement

In the competitive e-commerce landscape, understanding customer preferences, pricing strategies, and product performance is essential for business success. Amazon, one of the largest global online marketplaces, offers a wealth of product and customer feedback data. By analyzing this data, businesses can gain insights into consumer behavior, price trends, product popularity, and customer satisfaction.

The primary goal of this project is to explore and analyze a large dataset of Amazon product listings to:

Identify price and rating distributions across various product categories.
Uncover patterns and trends in customer ratings and the effect of discounts on product ratings.
Analyze the correlation between product prices, ratings, and customer preferences to provide actionable insights for improving business strategy, product positioning, and pricing optimization.
This analysis will help businesses to better understand market dynamics, optimize their product offerings, and develop effective pricing and promotion strategies tailored to customer expectations.

Data Description

The dataset used for this analysis is a comprehensive collection of Amazon product information, consisting of over 550,000 entries. Each entry includes various product attributes that provide insight into the characteristics and performance of products on the platform. The main features in the dataset include:

Product Name: The name of the product listed on Amazon.
Category: Main category and subcategory of the product, offering a breakdown of the product's classification (e.g., electronics, fashion).
Actual Price: The original price of the product before any discounts.
Discounted Price: The price after any discounts applied, if available.
Ratings: Customer ratings for the product, indicating overall customer satisfaction.
Number of Ratings: The total number of customer ratings submitted for the product, providing a sense of popularity and engagement.
Brand Name: The brand or manufacturer of the product, useful for analyzing brand performance and popularity.

Data Source

The dataset was sourced from [specific source or platform, e.g., Kaggle] and is intended for academic and analytical use, allowing us to explore trends in consumer behavior and pricing across a vast array of products.

Data Preprocessing Steps

The raw dataset contains some inconsistencies and missing values, particularly in pricing and ratings.

Preprocessing steps included:

Handling missing values in key columns such as price and ratings.
Standardizing formats for price columns, removing currency symbols, and converting strings to numeric types.
Removing duplicates to ensure data quality and integrity.
This cleaned and transformed dataset enables meaningful analysis and insight extraction to achieve the project objectives.

SOLUTION APPORACH:

1. Data Collection and Inspection

- **Load the Dataset**: Import the Amazon product data into the environment (e.g., using pandas to read a CSV file).

- **Initial Inspection**: Review the dataset's shape, column names, data types, and sample entries. Use methods like .head(), .info(), and .describe() to understand the basic structure of the dataset.

- **Missing Value Analysis**: Check for missing values across columns, assess their distribution, and determine if any columns or rows should be removed based on the extent of missing data.

2. Data Cleaning and Preprocessing

- **Handle Missing Values**: For essential columns like actual_price, discounted_price, and ratings, handle missing data by filling values with appropriate statistics (e.g., median or mean) or removing rows if the missing data is substantial.

- **Data Type Conversion**: Convert text-based numerical data (e.g., price with currency symbols) to numerical types, allowing for mathematical operations.

- **Outlier Detection and Handling**: Identify and address outliers, particularly in numerical fields like price and ratings, using techniques such as interquartile range (IQR) filtering.

- **Data Standardization**: Standardize key fields (e.g., price in a consistent currency format) and remove unnecessary columns (e.g., image URLs) for a cleaner dataset.

3. Exploratory Data Analysis (EDA)

- **Descriptive Statistics**: Generate summary statistics (mean, median, standard deviation) for numerical columns to understand overall distributions.

- **Visual Analysis**:
    - **Distribution Analysis**: Visualize the distribution of actual_price, discounted_price, and ratings using histograms and box plots.
    - **Category-Wise Analysis**: Use bar charts to show product counts across categories and subcategories, identifying the most popular segments.
    - **Price and Rating Trends**: Analyze the relationship between product prices and customer ratings with scatter plots to find potential correlations.
    - **Discount Impact**: Assess the relationship between discounts (actual vs. discounted price) and customer satisfaction (ratings) to understand how discounts influence consumer perception.

4. Feature Engineering

- **New Feature Creation**:
    - **Discount Percentage**: Calculate the discount percentage for each product as a new feature.
    - **Rating Level**: Segment products based on ratings into categories like "Top-rated," "Average," and "Low-rated."
    - **Sales Performance Metric**: If possible, create a metric for sales performance using available data on ratings and discount levels.
- **Normalization and Scaling**: Normalize numerical features (e.g., actual_price, discounted_price) for better visualizations and analysis consistency, especially if machine learning models are applied.

5. Statistical Analysis

- **Correlation Analysis**: Compute the correlation matrix for numerical columns to detect linear relationships, particularly between pricing, discounts, and ratings.
- **Hypothesis Testing**:
  - **ANOVA and T-tests**: Compare means across product categories or brands to see if significant differences exist in pricing or ratings.
  - **Correlation Testing**: Use Pearson or Spearman correlation tests to statistically assess the relationship between discounts and sales or ratings.
- **Outlier Analysis**: Use methods like Z-score or IQR to examine the impact of outliers and decide on handling methods.

6. Visualization and Interpretation

- **Graphical Insights**:
  - Create visualizations to interpret data trends effectively, including heatmaps, box plots, scatter plots, and bar charts.
  - Generate category-wise insights, e.g., the distribution of high-rating products across categories, and visualize sales trends across discount levels.
- **Dashboard Creation (Optional)**: Use libraries like Plotly to create interactive dashboards for dynamic exploration of product categories, pricing, and ratings.

7. Result Analysis and Business Insights

- **Insight Extraction**: Analyze EDA results and statistical tests to draw actionable conclusions about consumer behavior, product positioning, and pricing strategies.
- **Recommendation Development**: Based on findings, provide recommendations on optimal pricing strategies, potential areas for discounting, and popular product categories for targeted marketing.

8. Conclusion and Future Work

- **Summarize Findings**: Provide a summary of key insights, including trends in customer ratings, pricing patterns, and product popularity.
- **Future Research**: Suggest areas for further research, such as sentiment analysis on customer reviews or expanding analysis to include more recent or specialized datasets.

Required Libraries or used libarires

1. Data Manipulation and Analysis

- **Pandas**: Used for data manipulation, cleaning, and organizing structured data into dataframes. Essential functions include reading the CSV file, handling missing values, filtering, and aggregating data.

Code: import pandas as pd

- **NumPy**: Utilized for numerical operations and handling arrays. It's often used to calculate statistics like mean, median, and for handling data transformations.

Code : import numpy as np

2. Statistical Analysis

- **SciPy**: Offers statistical functions and tests, such as correlation tests and ANOVA, to understand relationships and differences between groups.

Code: from scipy import stats

3. Data Visualization

- **Matplotlib**: A foundational plotting library for creating static, publication-quality visualizations. It's useful for line plots, histograms, and scatter plots.

Code : import matplotlib.pyplot as plt

- **Seaborn**: Built on top of Matplotlib, it provides more complex visualizations and easier syntax for statistical plots like box plots, pair plots, and heatmaps.

Code: import seaborn as sns

- **Plotly** (Optional): A library for interactive visualizations, suitable for creating dynamic dashboards and visuals that allow for interactive exploration.

Code : import plotly.express as px

4. Machine Learning and Feature Scaling (Optional for advanced analysis)

- **Scikit-Learn**: Provides tools for feature scaling, outlier detection, clustering, and dimensionality reduction, helpful for analyzing relationships and grouping data.

Code:

```
from sklearn.preprocessing import StandardScaler

from sklearn.decomposition import PCA
```

Introduction:

The e-commerce industry has witnessed tremendous growth over the past decade, fueled by the convenience, variety, and competitive pricing offered by online marketplaces. Among these, Amazon stands out as one of the largest and most influential platforms, hosting millions of products across countless categories. With its vast customer base, Amazon generates extensive data that reflects customer preferences, product performance, and pricing trends. This data holds immense value for businesses seeking to understand and adapt to consumer behavior. However, the vast quantity of data also presents a challenge in extracting meaningful insights without sophisticated data analysis techniques.

The purpose of this project is to analyze Amazon's product data to identify trends in customer preferences, pricing patterns, and product popularity. By leveraging a dataset containing over 550,000 Amazon product listings, we aim to uncover insights into how different factors—such as product categories, price discounts, and ratings—affect consumer behavior and satisfaction. Our primary objective is to gain a better understanding of customer priorities and preferences, enabling businesses to optimize their offerings and pricing strategies.

The project follows a structured approach that begins with data cleaning to ensure a reliable dataset for analysis. Following this, Exploratory Data Analysis (EDA) is conducted to identify trends in product categories, ratings, and price distribution. Visualizations and statistical tests are used to examine relationships between key variables, such as the correlation between price and customer ratings, as well as the impact of discounts on customer sentiment. By exploring these aspects, the project provides a holistic view of product performance on Amazon, revealing insights that can help sellers make data-driven decisions.

This analysis not only enhances understanding of Amazon's marketplace dynamics but also provides valuable insights for e-commerce businesses aiming to improve customer satisfaction, increase sales, and optimize product positioning. The findings have broad applications, from guiding marketing strategies to informing inventory management. In the long term, such analyses can support businesses in remaining competitive in a rapidly evolving e-commerce landscape. Future iterations of this project could benefit from integrating textual reviews and sales volume data, providing a more comprehensive view of customer sentiment and product performance over time.

.

Objective:

The objectives of this project are:

Data Exploration and Cleaning:

- Examine the Amazon product dataset to identify any missing, inconsistent, or erroneous values.
- Clean the dataset by handling missing values, standardizing formats for pricing and ratings, and ensuring data quality for analysis.

Exploratory Data Analysis (EDA):

- Analyze the distribution of product prices (both actual and discounted) across different categories and subcategories.
- Investigate the relationship between customer ratings and product prices to uncover patterns or trends in customer satisfaction.
- Visualize key insights using graphs and charts to explore the most popular product categories, pricing ranges, and rating distributions.

Price Analysis:

- Compare actual prices with discounted prices to assess the extent of price reductions across different product categories.
- Analyze pricing trends and their correlation with customer ratings to understand if higher-rated products tend to have higher or lower prices.

Customer Rating Insights:

- Study the distribution of customer ratings across various products to understand patterns in customer satisfaction.
- Explore the relationship between product ratings and the number of customer reviews to identify whether highly-rated products also attract more reviews.

Business Strategy and Market Insights:

- Provide actionable insights for businesses to optimize their pricing strategies based on product category and customer feedback.
- Identify potential areas for improvement in product offerings based on customer ratings and pricing trends.
- Suggest data-driven approaches for improving customer satisfaction and increasing product competitiveness in the e-commerce marketplace.

Future Research and Applications:

- Highlight areas for further analysis, such as applying sentiment analysis on product reviews to complement the numerical analysis of ratings and prices.
- Propose enhancements to data collection and analysis techniques to provide a more comprehensive understanding of customer behavior and market dynamics.

Methodology:

The methodology followed in this project consists of a structured workflow starting from data collection to visualization. The steps involved are:

1. **Data Collection:**

Data Inspection – (shape,dtypes,content) for basic understanding – checking for missing values,outliers and any data quality issues.

- **Data Cleaning**

  Before analysis, it was important to clean the dataset to address missing or incomplete data and format inconsistencies:

  - **Handling Missing Values:**
  - Remove rows/columns with grater than 70% missing values if neccessary
  - ∗ Fill null values with 0
  - ∗ Impute/replace null values using methods like mean, median,mode or more advanced techniques
  - **Data Type Conversion:** Columns like Ratings column and No_of_Ratings Column, w Discount_price column and Actual_price column which were stored as strings, were converted to numeric types toenable meaningful analysis.
  - **Duplicate Removal:** Any duplicates in the data were removed using the drop_duplicates() function to ensure accurate analysis.

2. **Exploratory Data Analysis (EDA):**

Once the data was clean, exploratory analysis was conducted to identify key trends and patterns:

- **Descriptive Statistics:** Summary statistics were calculated using .describe(), showing count, mean, and standard deviation across numeric columns like release year and duration.
- **Data Visualization:** Using tools like Matplotlib, Seaborn, and Plotly, charts were created to visualize the content distribution by year, country, and genre.
- **Correlation Matrix:** A correlation matrix was generated to find relationships between different features

3. **Tools and Libraries:**

Pandas: Used for data manipulation and cleaning.

- **Matplotlib & Seaborn:** Used for creating visualizations like bar charts, line plots, and histograms.
- **Plotly:** For building interactive dashboards and visualizations.

Python: Used for coding and executing the analysis

Statistical Analysis:

A range of statistical techniques was used to uncover the relationships andpatterns in the Amazon product sales dataset:

1. **Descriptive Statistics:**

   Descriptive statistics were calculated for important features like content release years, duration, and ratings. For example, the .describe() function was used to provide a summary of these features, including measures like mean, median, and standard deviation.

2. **Correlation Analysis:**

A correlation matrix was computed to explore how different variables relate to each other. For instance, we examined numerical columns

3.Outlier Detection:

Using box plots, outliers in variables like content duration were identified. Outliers were handled by capping or removing extreme values,depending on their relevance to the overall analysis

Graphical Visualizations for Amazon Products Dataset:

1. **Content by Category and Subcategory:**
   - **Bar charts** can be used to show the distribution of products across different categories and subcategories. This would help highlight which categories have the highest product count, such as "Appliances" or "Men's Clothing."
2. **Price Distribution Across Categories:**
   - A **box plot** or **violin plot** can illustrate the range of prices (both actual and discounted) within each category. This would allow us to identify which categories tend to have higher or lower pricing.
3. **Ratings Distribution:**
   - A **histogram** of customer ratings would show how ratings are distributed across the products. We can visualize whether most products are highly rated or whether there are outliers with poor ratings.
4. **Correlation Between Discount and Sales:**
   - A **scatter plot** can show the relationship between discount percentage and sales performance. This visualization can reveal whether a higher discount leads to more sales.
5. **Top Products by Rating:**
   - A **bar chart** showcasing the top-rated products can provide insights into which items receive the best feedback from customers.
6. **Price vs. Rating:**

- A **scatter plot** comparing product prices to their ratings can help identify whether there’s a trend between how much a product costs and how well it is rated by customers.

REQUIRED LIBRARIES

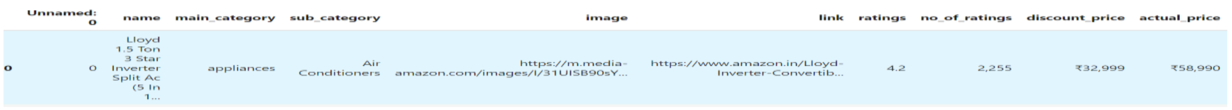
In [203]:

```
import pandas as pd # used for Data Manuplation and Handling
import numpy as np # used for Numerical operations
import statistics as stat # used for Statistical Calculations
import matplotlib.pyplot as plt # used for plotting graphs(python plotting
package)
import seaborn as sb # used for visualization
```

IMPORTING DATA SET

data = pd.read_csv("Amazon-product.csv")

data.head(5)



	Unnamed: 0	name	main_category	sub_category	image	link	ratings	no_of_ratings	discount_price	actual_price
0	0	Lloyd 1.5 Ton 3 Star Inverter Split Ac (5 In 1...	appliances	Air Conditioners	https://m.media-amazon.com/images/I/31UISB90sY...	https://www.amazon.in/Lloyd-Inverter-Convertib...	4.2	2,255	₹32,999	₹58,990

data.shape[0]

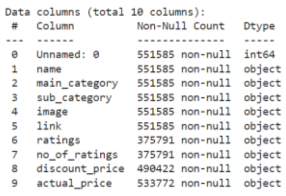
551585

data.shape[1]

10

What is the structure and integrity of the dataset?

data.info()



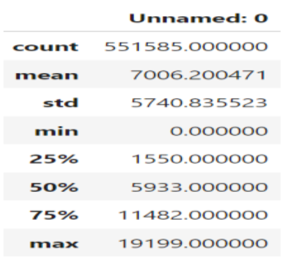
```
Data columns (total 10 columns):
 #   Column          Non-Null Count   Dtype
---  ------          --------------   -----
 0   Unnamed: 0      551585 non-null  int64
 1   name            551585 non-null  object
 2   main_category   551585 non-null  object
 3   sub_category    551585 non-null  object
 4   image           551585 non-null  object
 5   link            551585 non-null  object
 6   ratings         375791 non-null  object
 7   no_of_ratings   375791 non-null  object
 8   discount_price  490422 non-null  object
 9   actual_price    533772 non-null  object
```

To display column names of the dataset

data.columns

To display statistical values of the dataset

data.describe()



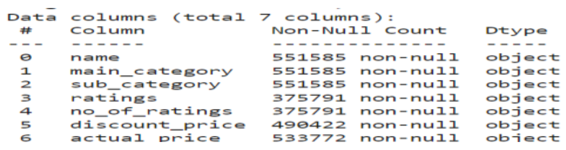
	Unnamed: 0
count	551585.000000
mean	7006.200471
std	5740.835523
min	0.000000
25%	1550.000000
50%	5933.000000
75%	11482.000000
max	19199.000000

Dropping unneccesary columns

columns_to_drop = ['image','link']

data =data.drop(columns=columns_to_drop)

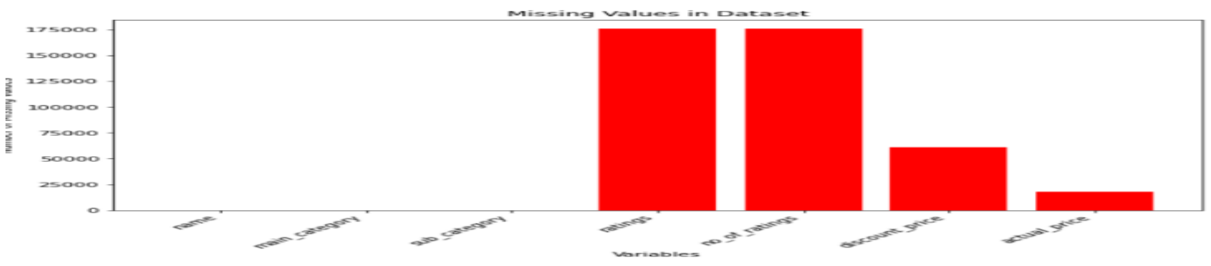
del data['Unnamed: 0'] *# data.drop(columns='Unnamed: 0',inplace= True)*



```
Data columns (total 7 columns):
 #   Column          Non-Null Count   Dtype
---  ------          --------------   -----
 0   name            551585 non-null  object
 1   main_category   551585 non-null  object
 2   sub_category    551585 non-null  object
 3   ratings         375791 non-null  object
 4   no_of_ratings   375791 non-null  object
 5   discount_price  490422 non-null  object
 6   actual_price    533772 non-null  object
```

Displaying Missing values

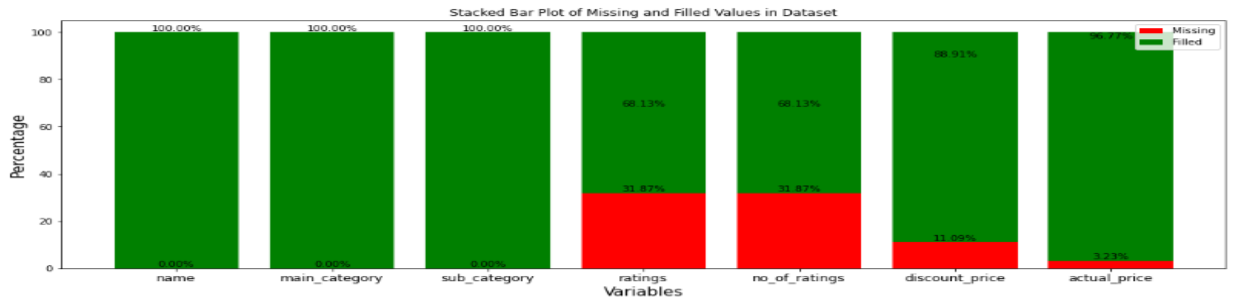
data.isnull().sum()



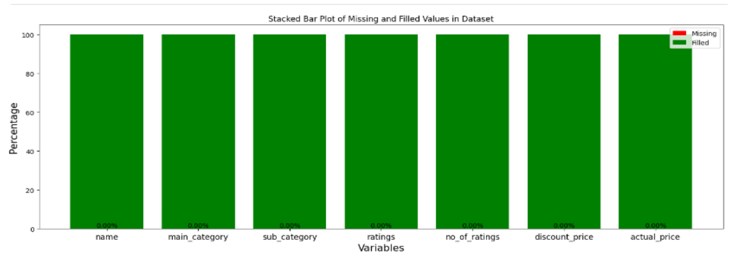
Missing percentage and filled percentage graphs:

missing_percentage = (data.isnull().sum() / len(data)) * 100

filled_percentage = 100 - missing_percentage

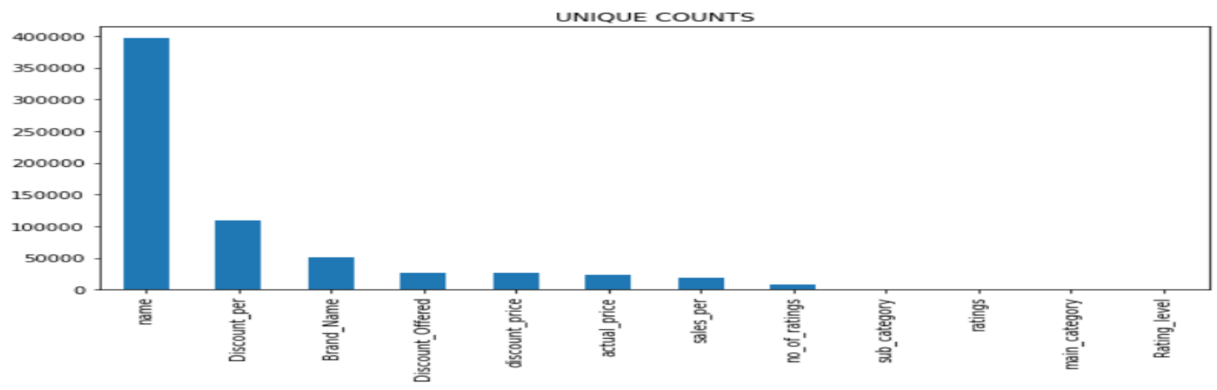


After data cleaning of missing vaues:



```
Data columns (total 7 columns):
 #   Column          Non-Null Count   Dtype
---  ------          --------------   -----
 0   name            551585 non-null  object
 1   main_category   551585 non-null  object
 2   sub_category    551585 non-null  object
 3   ratings         551585 non-null  object
 4   no_of_ratings   551585 non-null  object
 5   discount_price  551585 non-null  object
 6   actual_price    551585 non-null  object
```

Generation of bar chart showing the number of unique values for each column in the DataFrame.

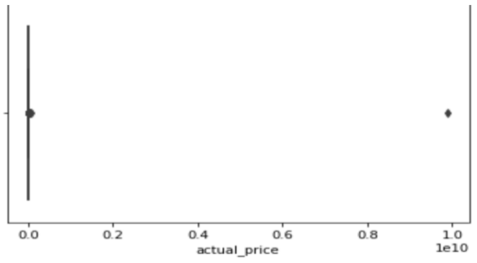


OUTLIERS DETECTION

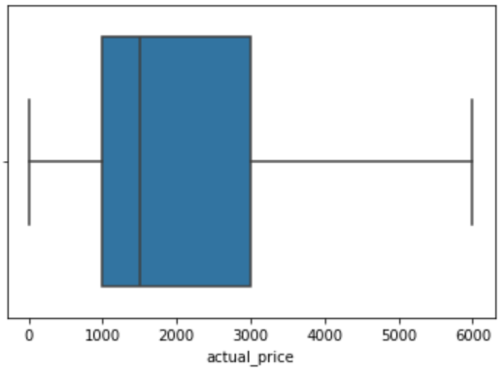
FOR Actual price

sb.boxplot(data=data,x='actual_price')

plt.show()



After IQR Boxplot for actual price

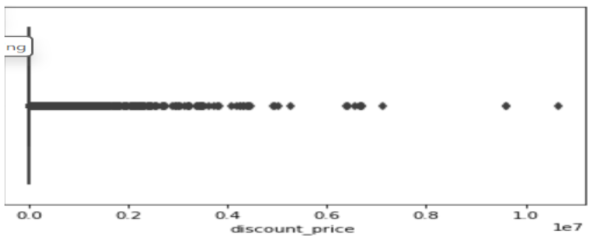


OUTLIER DETECTION

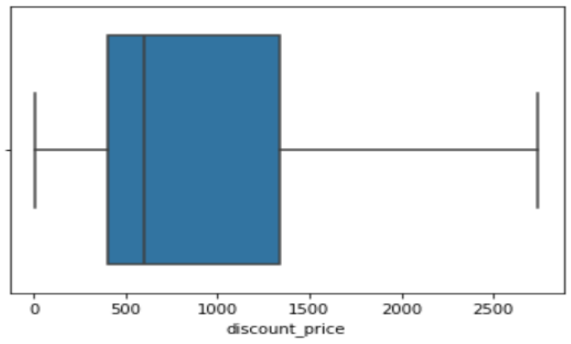
For Discount price

sb.boxplot(data=data,x='discount_price')

plt.show()



After IQR Boxplot for discount price

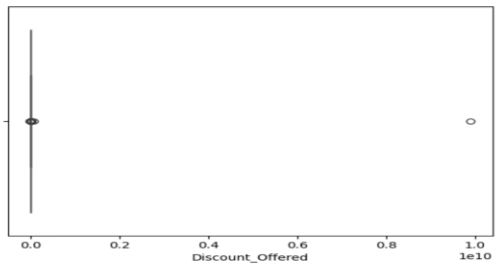


OUTLIER DETECTION

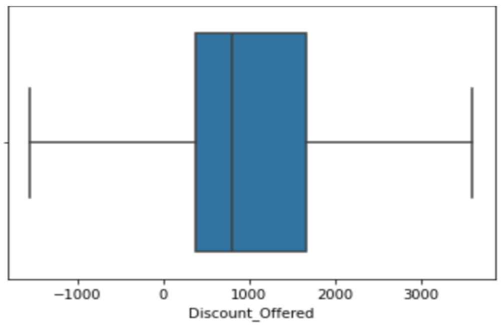
FOR DISCOUNT_OFFERED

sb.boxplot(data=data,x='Discount_Offered')

plt.show()



After IQR Boxplot for discount_offered

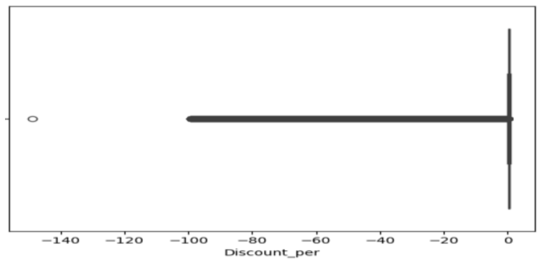


OUTLIER DETECTION

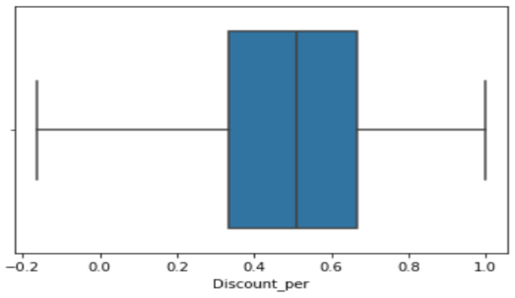
FOR Discoutn_per

sb.boxplot(data=data,x='Discount_per')

plt.show()



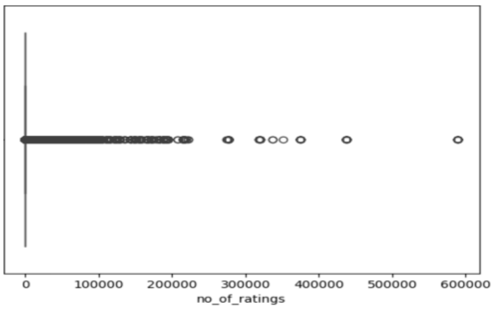
After IQR Boxplot for discount_per



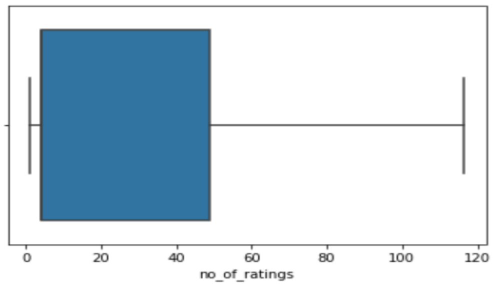
OUTLIER DETECTION

FOR no_of_ratings

```
sb.boxplot(data=data, x='no_of_ratings')
plt.show()
```



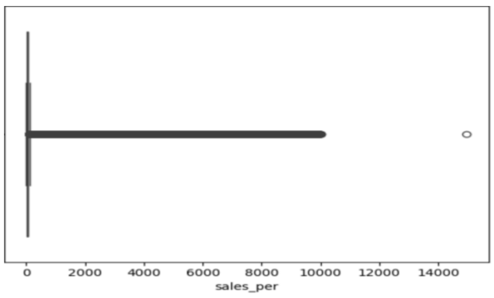
After IQR Boxplot for no_of_ratings



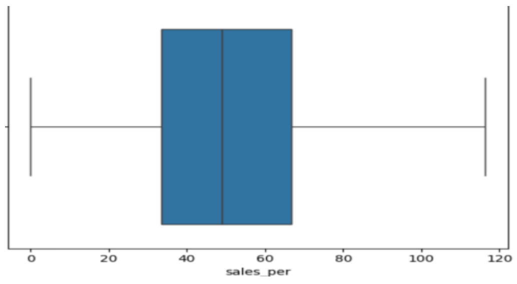
OUTLIER DETECCTION

FOR sales_per

```
sb.boxplot(data=data,x='sales_per')
plt.show()
```



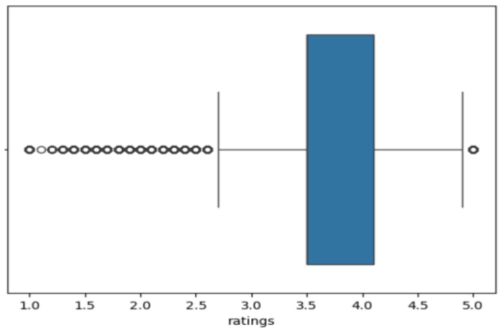
After IQR Boxplot for no_of_ratings



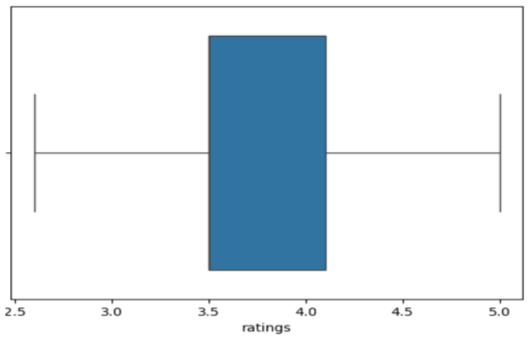
OUTLIER DETECTION FOR

Ratings

```
sb.boxplot(data=data, x='ratings')
plt.show()
```



After IQR Boxplot for ratings



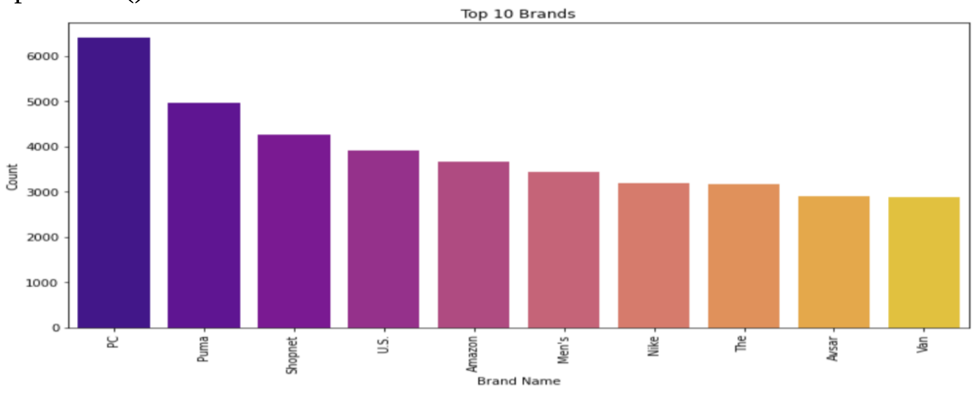
UNI-VARiATE ANALYSIS OF BRAND_NAME

Univariate analysis examines a single variable to understand its distribution, central tendency, and spread. It helps summarize the data using descriptive statistics like mean, median, mode, variance, and range. Common types of graphs and plots for univariate analysis include:

1. Histogram – Shows the frequency distribution of a continuous variable.
2. Box plot – Displays the spread, median, and outliers of the data.
3. Bar chart – Used for categorical data to display frequency.
4. Pie chart – Represents proportions of categorical data.
5. Density plot – Smoothed version of a histogram to show the distribution's shape.

What are the top 10 brands with respective to their product counts?

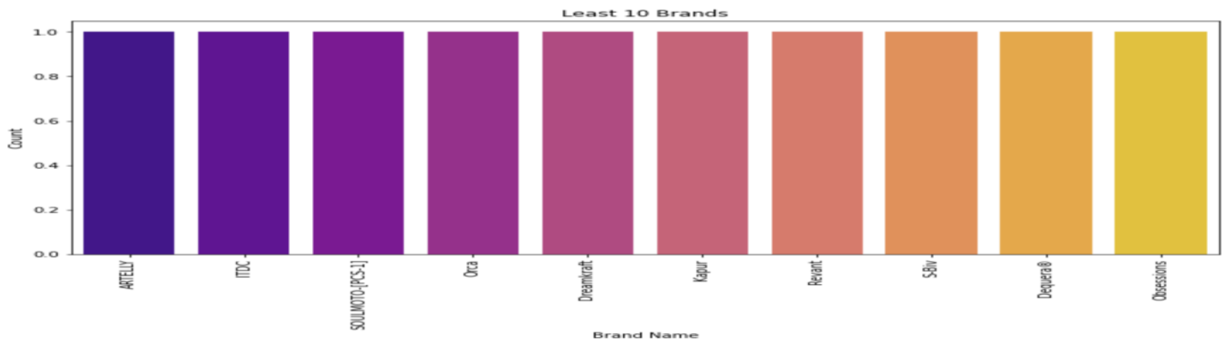
```
brand_counts = data['Brand_Name'].value_counts()
top_brands = brand_counts.head(10)  # Display top 10 brands
plt.figure(figsize=(12, 6))
sb.barplot(x=top_brands.index, y=top_brands.values, palette='plasma')
plt.show()
```



- The given barchart demonstrates the top 10 brands which have high product count

What are the bottom 10 brands with respective to thier Product counts?

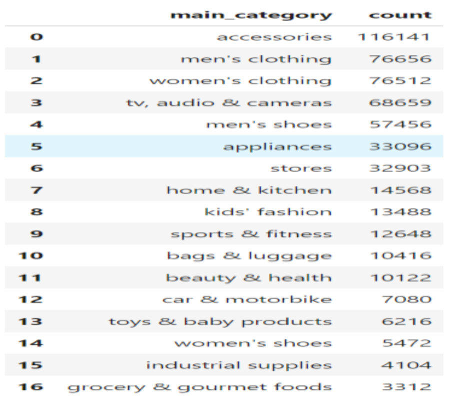
```
brand_counts = data['Brand_Name'].value_counts()
low_brands = brand_counts.tail(10)  # Display top 10 brands
plt.figure(figsize=(12, 6))
sb.barplot(x=low_brands.index, y=low_brands.values, palette='plasma')
plt.show()
```



. The given chart demonstares the least 10 brands which have low product count

What are the main categories of the products in terms of their counts?

data['main_category'].value_counts().reset_index()



	main_category	count
0	accessories	116141
1	men's clothing	76656
2	women's clothing	76512
3	tv, audio & cameras	68659
4	men's shoes	57456
5	appliances	33096
6	stores	32903
7	home & kitchen	14568
8	kids' fashion	13488
9	sports & fitness	12648
10	bags & luggage	10416
11	beauty & health	10122
12	car & motorbike	7080
13	toys & baby products	6216
14	women's shoes	5472
15	industrial supplies	4104
16	grocery & gourmet foods	3312

Distribution of main categeroies

# Column: 'main_category' - Main Product Category

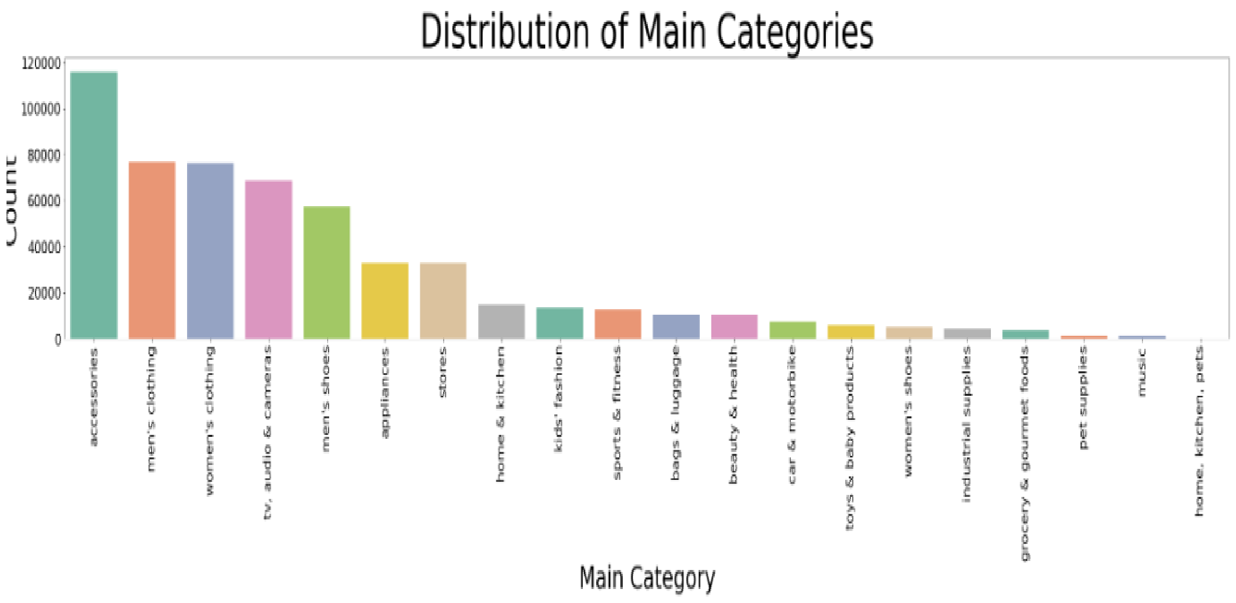
# Univariate analysis for categorical data

category_counts = data['main_category'].value_counts()

plt.figure(figsize=(40, 6))

sb.barplot(x=category_counts.index, y=category_counts.values, palette='Set2')

plt.show()

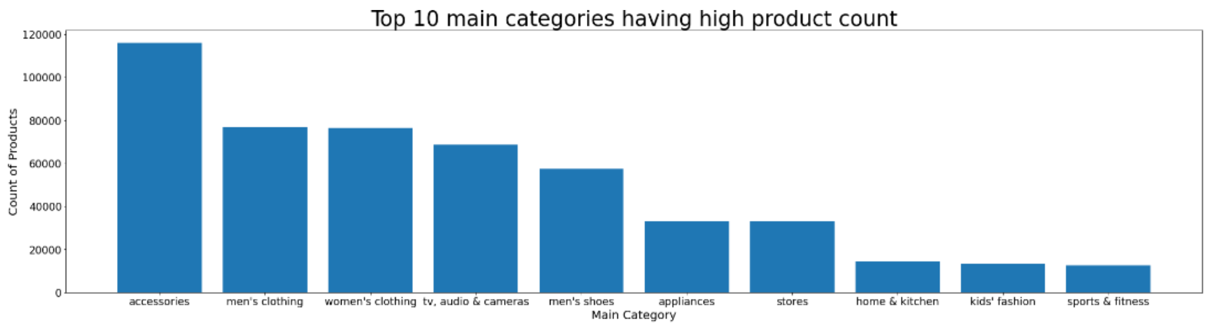


Product count of top 10 categories:

plt.figure(figsize=(35,8))

plt.bar(main_cat['main_category'],main_cat['count'])

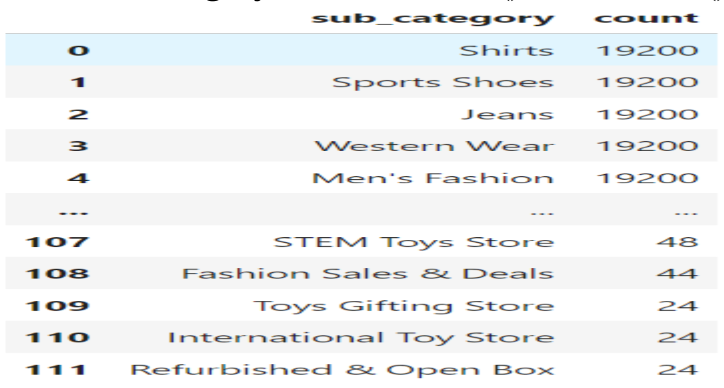
plt.show()



- Accessories have more no.of products in Main category

What are the sub-categories of the products with respect to their product count?

data['sub_category'].value_counts().reset_index()



	sub_category	count
0	Shirts	19200
1	Sports Shoes	19200
2	Jeans	19200
3	Western Wear	19200
4	Men's Fashion	19200
...	...	...
107	STEM Toys Store	48
108	Fashion Sales & Deals	44
109	Toys Gifting Store	24
110	International Toy Store	24
111	Refurbished & Open Box	24

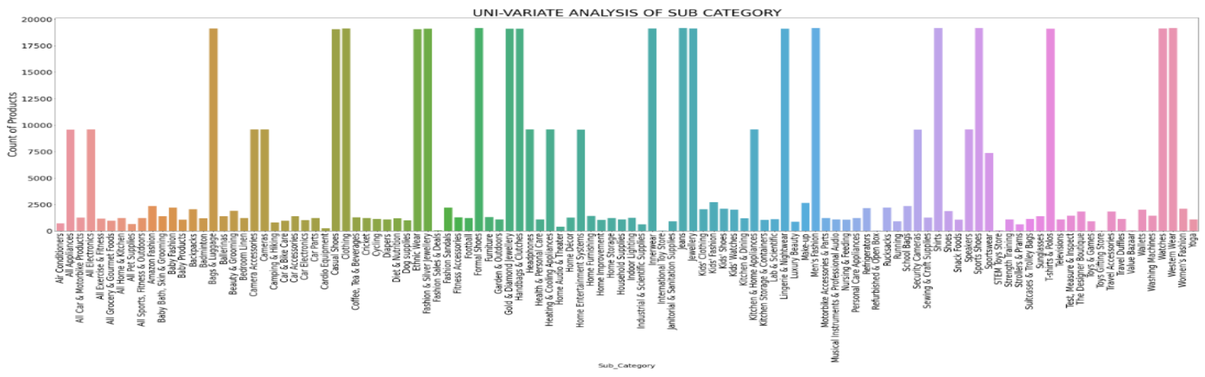
Univariate anaylsis of sub_category

plt.figure(figsize=(50,15))

sb.countplot(data=data,x='sub_category')

plt.title("UNI-VARIATE ANALYSIS OF SUB CATEGORY",fontsize=36)

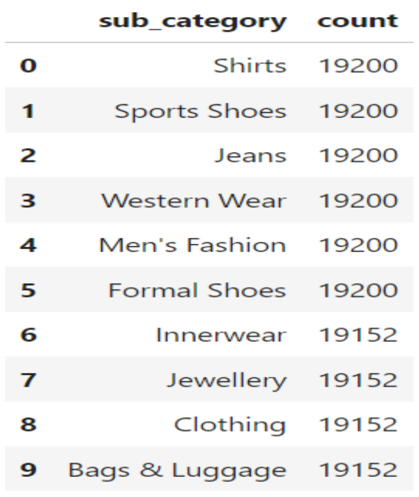
plt.show()



Top 10 sub categories

sub_cat=data['sub_category'].value_counts().reset_index().head(10)

sub_cat



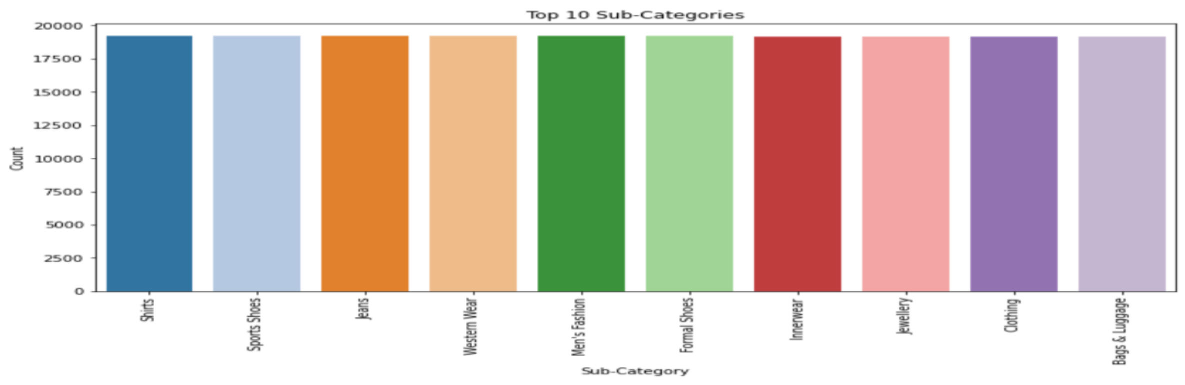
	sub_category	count
0	Shirts	19200
1	Sports Shoes	19200
2	Jeans	19200
3	Western Wear	19200
4	Men's Fashion	19200
5	Formal Shoes	19200
6	Innerwear	19152
7	Jewellery	19152
8	Clothing	19152
9	Bags & Luggage	19152

What are the top 10 sub categories with respect to product count?

```
# Column: 'sub_category' - Sub-Category
# Univariate analysis for categorical data
sub_category_counts = data['sub_category'].value_counts()
top_sub_categories = sub_category_counts.head(10)  # Display top 10 sub-categories
```

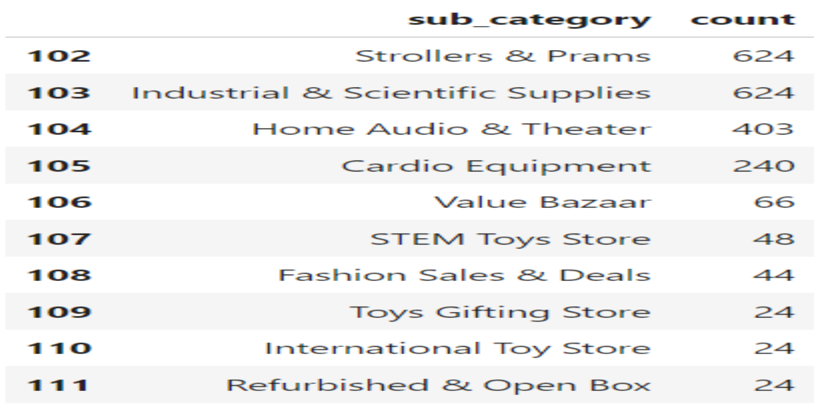
Top 10 sub categories

```
plt.figure(figsize=(12, 6))
sb.barplot(x=top_sub_categories.index, y=top_sub_categories.values, palette='tab20')
plt.show()
```



Bottom 10 sub categories

```
sub_cat2=data['sub_category'].value_counts().reset_index().tail(10)
sub_cat2
```



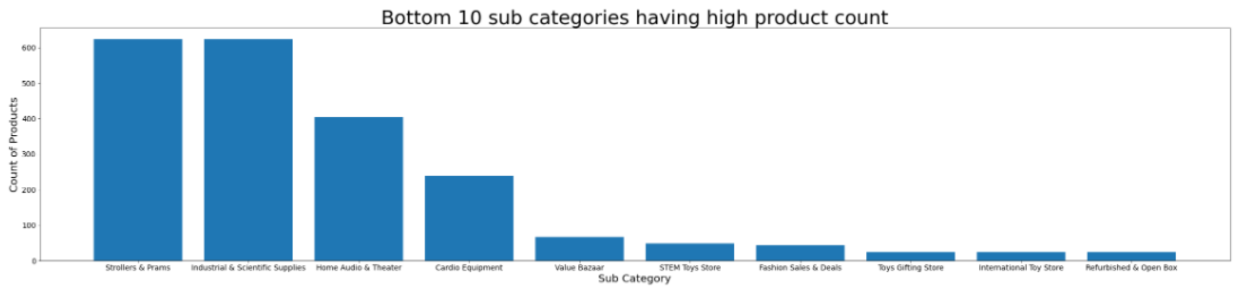
	sub_category	count
102	Strollers & Prams	624
103	Industrial & Scientific Supplies	624
104	Home Audio & Theater	403
105	Cardio Equipment	240
106	Value Bazaar	66
107	STEM Toys Store	48
108	Fashion Sales & Deals	44
109	Toys Gifting Store	24
110	International Toy Store	24
111	Refurbished & Open Box	24

What are the bottom 10 sub categories with respect to product count?

plt.figure(figsize=(40,8))

plt.bar(sub_cat2['sub_category'],sub_cat2['count'])

plt.show()

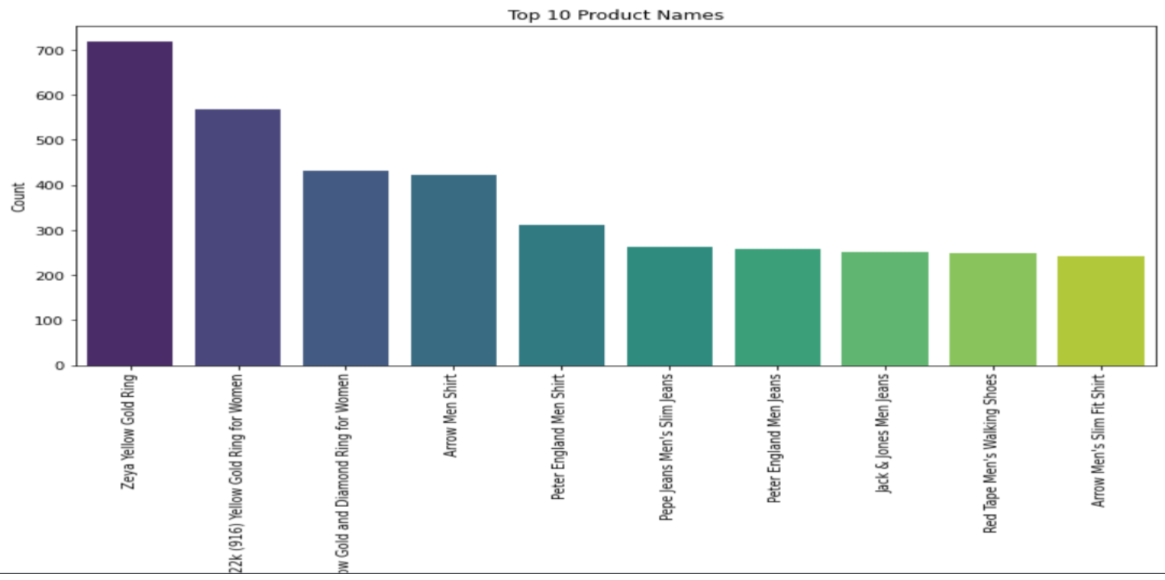


Top 10 produts names:

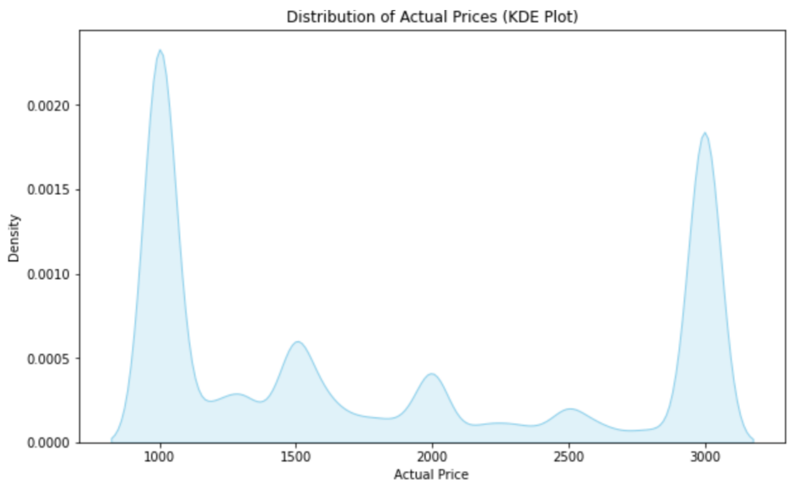
plt.figure(figsize=(12, 6))

sb.barplot(x=top_names.index, y=top_names.values, palette='viridis')

plt.show()

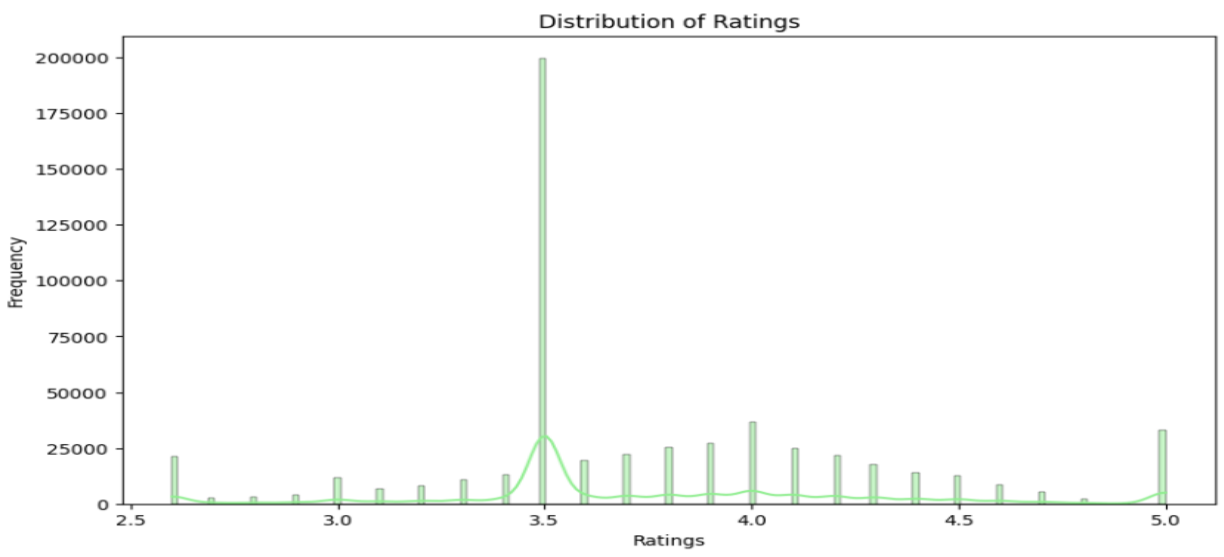


```
# Column: 'actual_price' - Actual Price
# Univariate analysis for numerical data (KDE plot)
plt.figure(figsize=(10, 6))
sb.kdeplot(data['actual_price'], color='skyblue', fill=True)
plt.show()
```



The KDE plot of the 'actual_price' column shows that the distribution of actual prices appears to be bimodal, with two peaks around 1000 and 2500. This suggests that there are two groups in the data with different price ranges. The density is highest at these two peaks, indicating that most of the prices are around these values. The density decreases as the price increases or decreases from these values, indicating fewer data points in those ranges. This could imply that items priced around 1000 and 2500 are more common

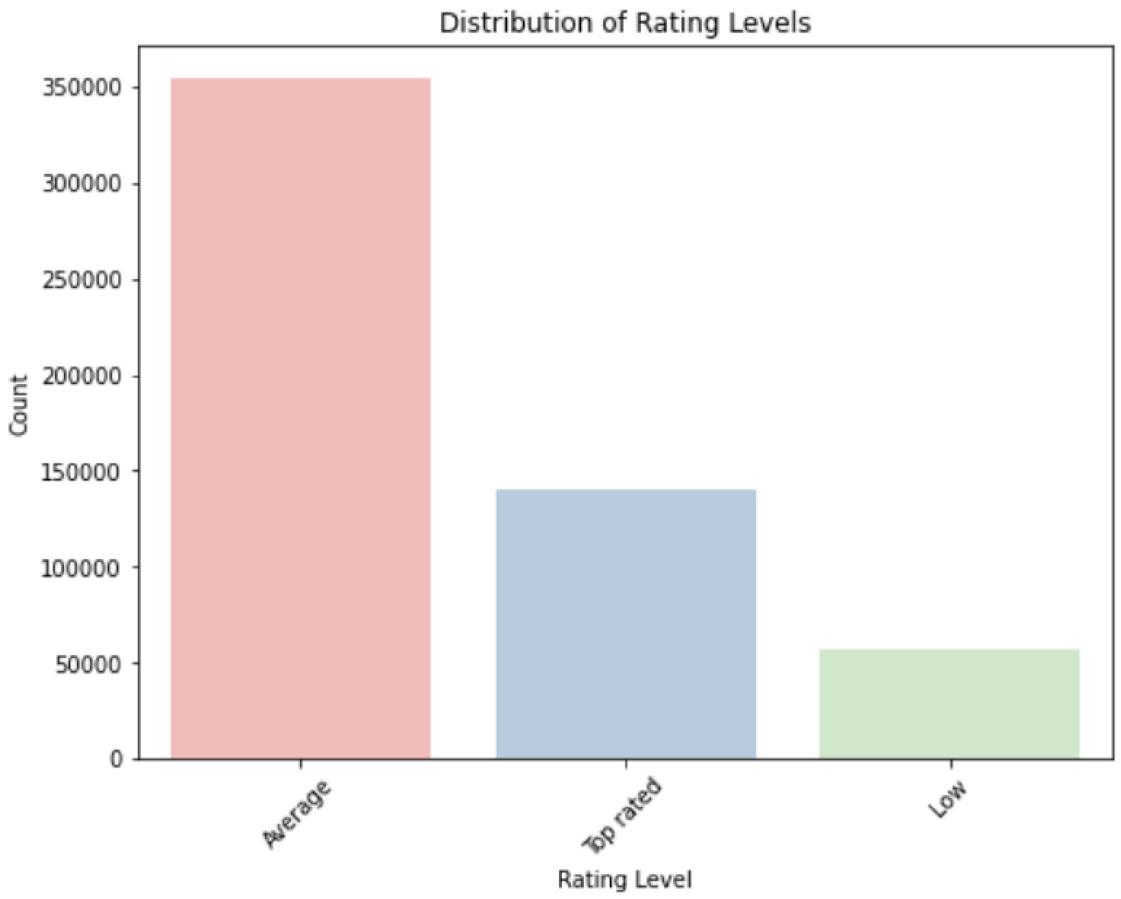
```
# Column: 'ratings' - Ratings (Numerical)
# Univariate analysis for numerical data (histogram)
#distributon of ratings high was 3.5
plt.figure(figsize=(10, 6))
sb.histplot(data['ratings'], kde=True, color='lightgreen')
plt.show()
```



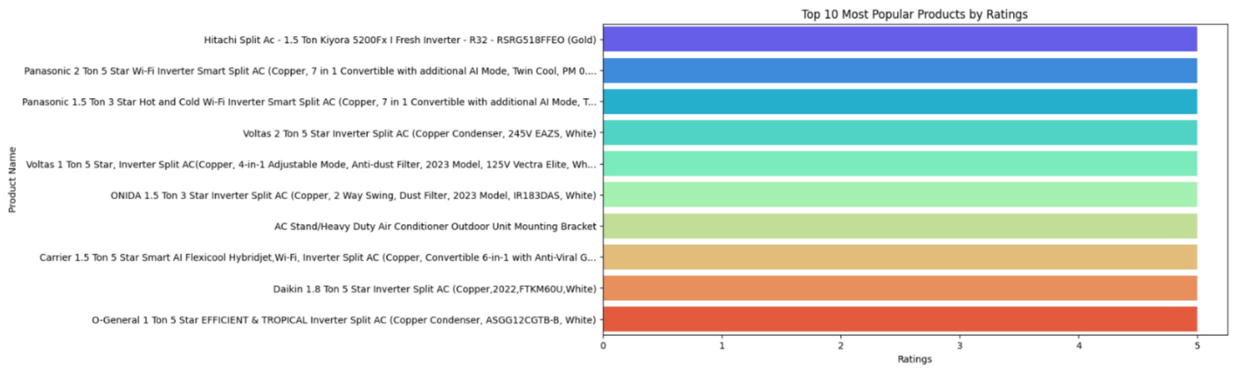
# Column: 'Rating_level' - Rating Level (Categorical)

# Univariate analysis for categorical data

```
rating_level_counts = data['Rating_level'].value_counts()
plt.figure(figsize=(8, 6))
sb.barplot(x=rating_level_counts.index, y=rating_level_counts.values,
palette='Pastel1')
plt.show()
```



```
top_rated_products = data.nlargest(10, 'ratings')
plt.figure(figsize=(12, 6))
sb.barplot(data=top_rated_products, x='ratings', y='name',palette='rainbow')
plt.title('Top 10 Most Popular Products by Ratings')
plt.xlabel('Ratings')
plt.ylabel('Product Name')
plt.show()
```



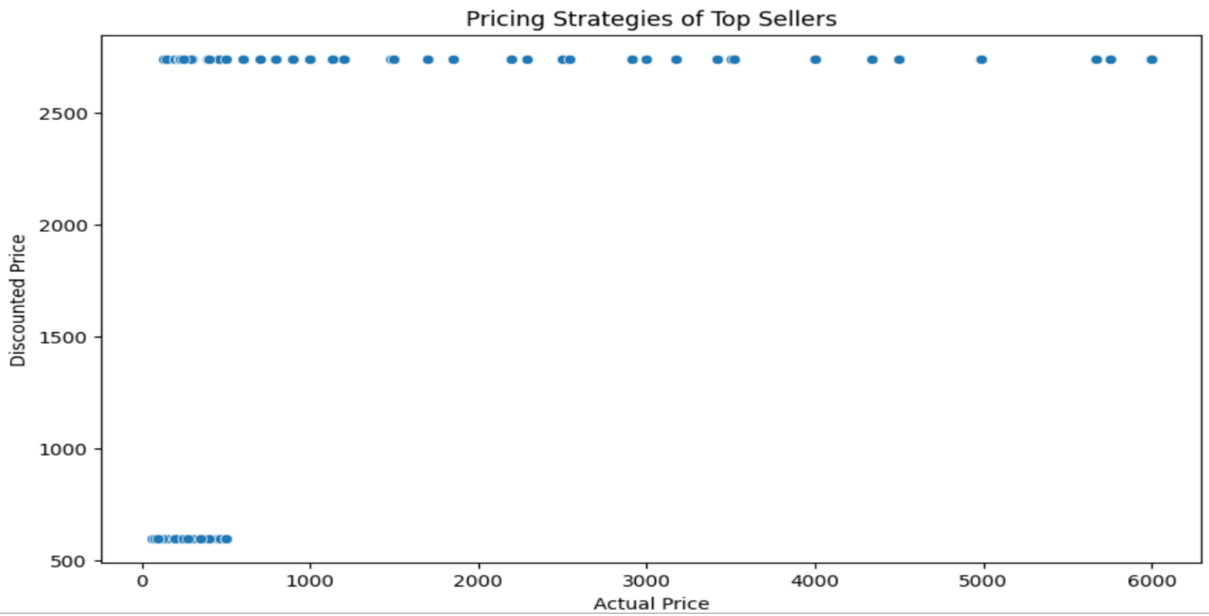
# Create a scatter plot to compare actual_price and discount_price for top-selling products

top_selling_products = data.nlargest(100, 'sales_per') # Assuming you consider the top 100 products as top sellers

sb.scatterplot(data=top_selling_products, x='actual_price', y='discount_price')

plt.show()

# Explain the pricing strategies you observe among top sellers.



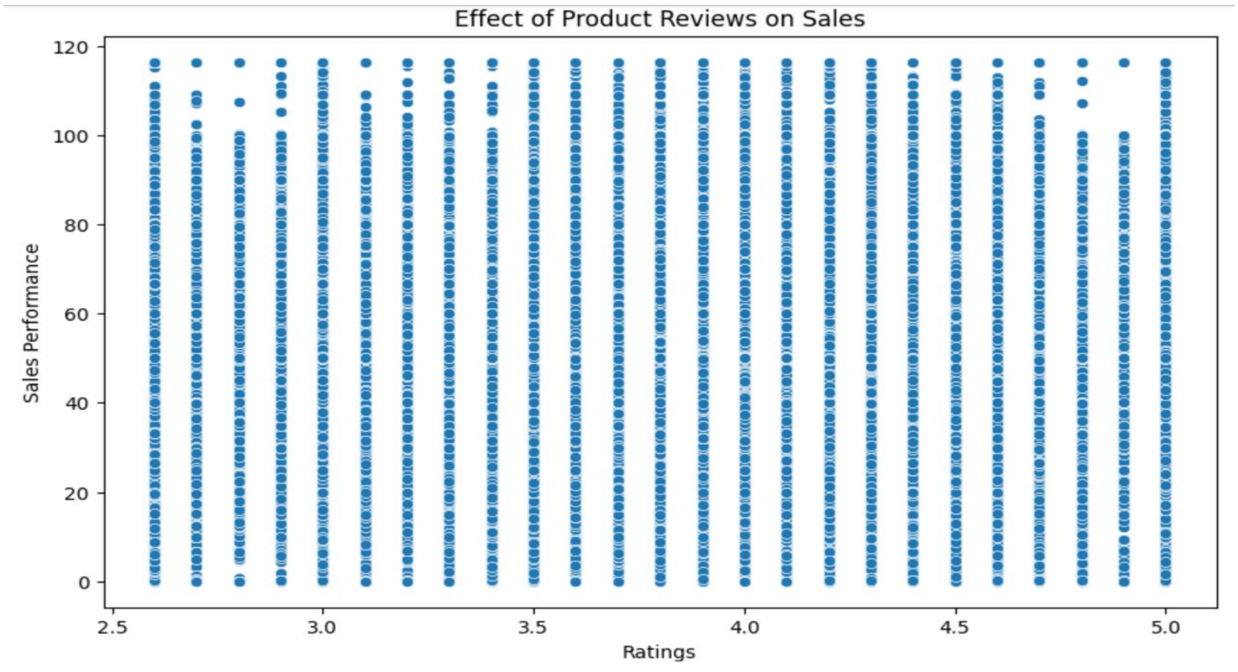
# Create a scatter plot to analyze the correlation between ratings and sales_per

plt.figure(figsize=(10, 6))

sb.scatterplot(data=data, x='ratings', y='sales_per')

plt.show()

# Explain the relationship between ratings and sales performance.



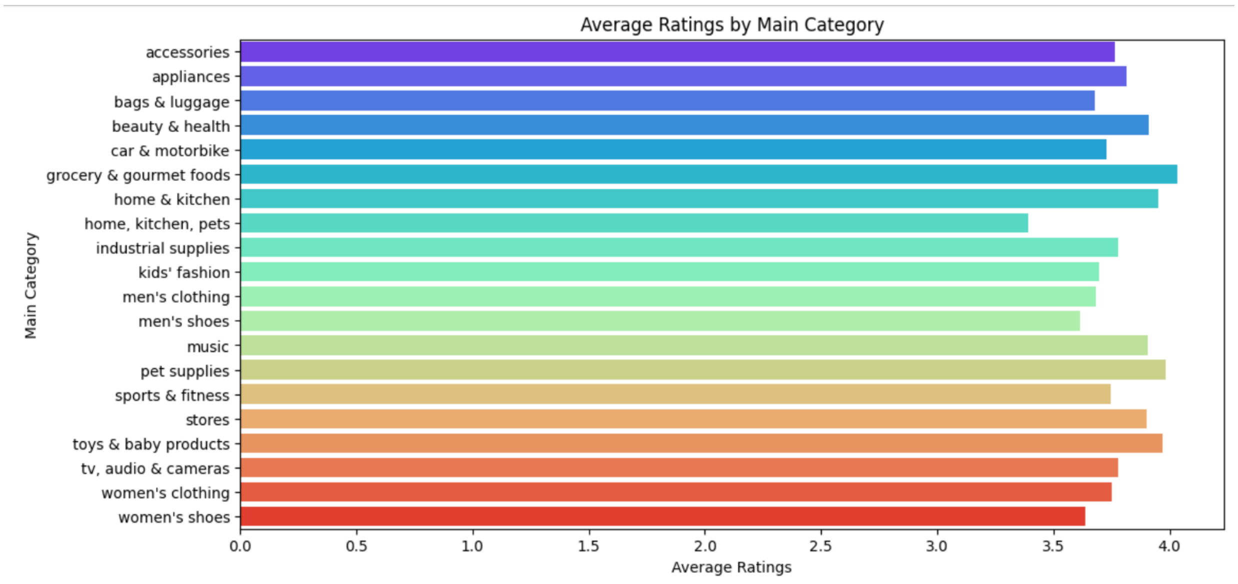
# Create a bar chart to show the average ratings by main_category

avg_ratings_by_category=data.groupby('main_category')['ratings'].mean().reset_index()

sb.barplot(data=avg_ratings_by_category,x='ratings',y='main_category',palette='rainbow')

plt.show()

# Explain which main categories have high average ratings.



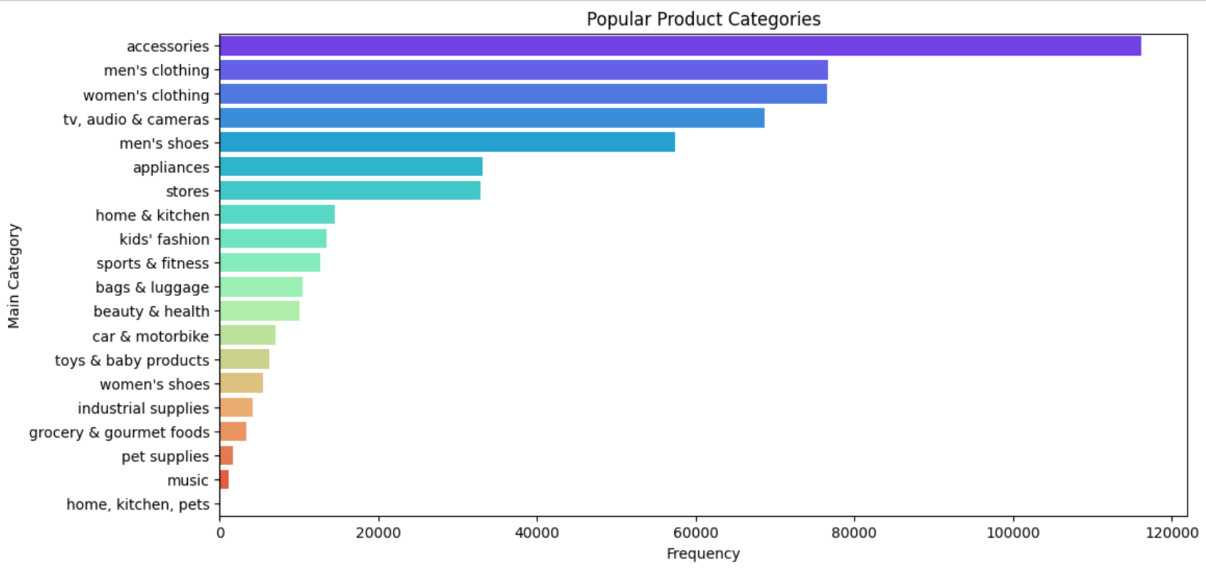
# Create a count plot to show the frequency of each main_category

plt.figure(figsize=(12, 6))

sb.countplot(data=data,y='main_category',palette='rainbow',order=data['main_category'].value_counts().index)

plt.show()

# Explain which main categories are the most popular based on frequency.



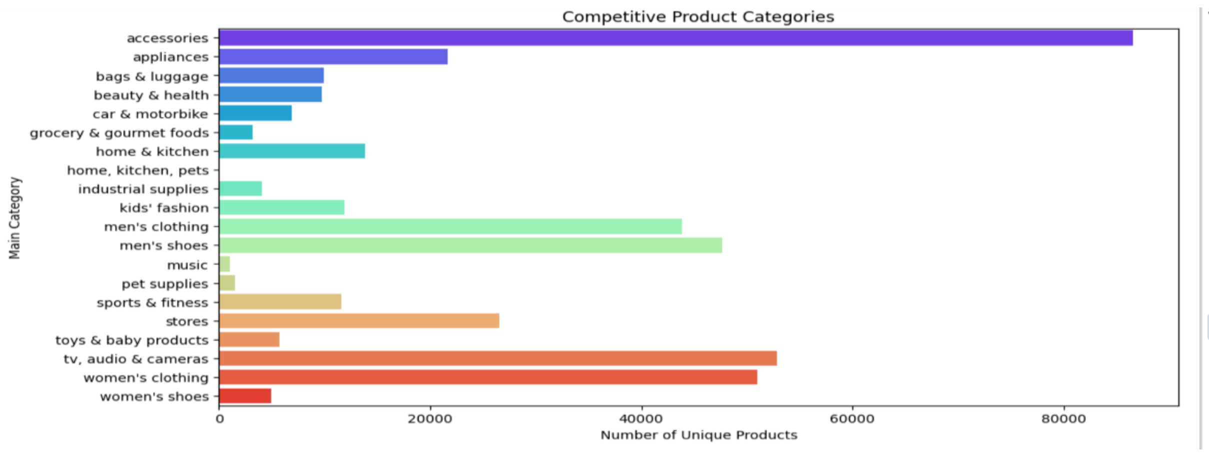
# Create a count plot to show the number of unique products or sellers within each main_category

unique_products_per_category=data.groupby('main_category')['name'].nunique().reset_index()

plt.figure(figsize=(12, 6))

sb.barplot(data=unique_products_per_category,x='name',y='main_category',palette='rainbow')

# Explain which main categories are the most competitive based on the number of unique products.



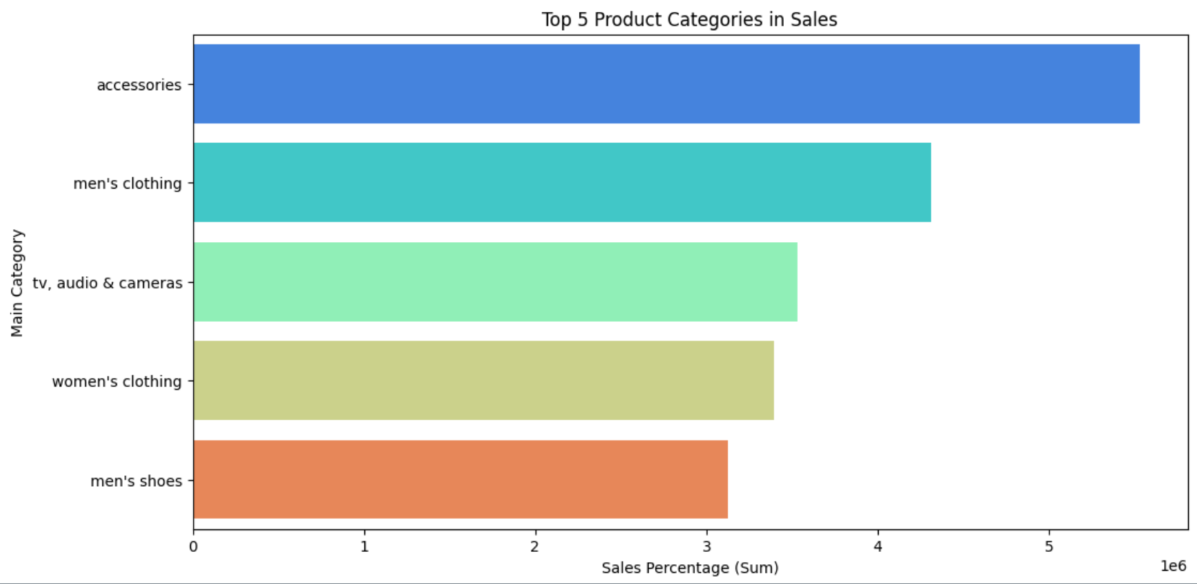
# Create a bar chart to show the top 5 main_category by sales_metric

top_5_categories=data.groupby('main_category')['sales_per'].sum().nlargest(5).reset_index()

sb.barplot(data=top_5_categories,x='sales_per',y='main_category',palette='rainbow')

plt.show()

# Explain the top 5 main categories with the highest sales_metric.



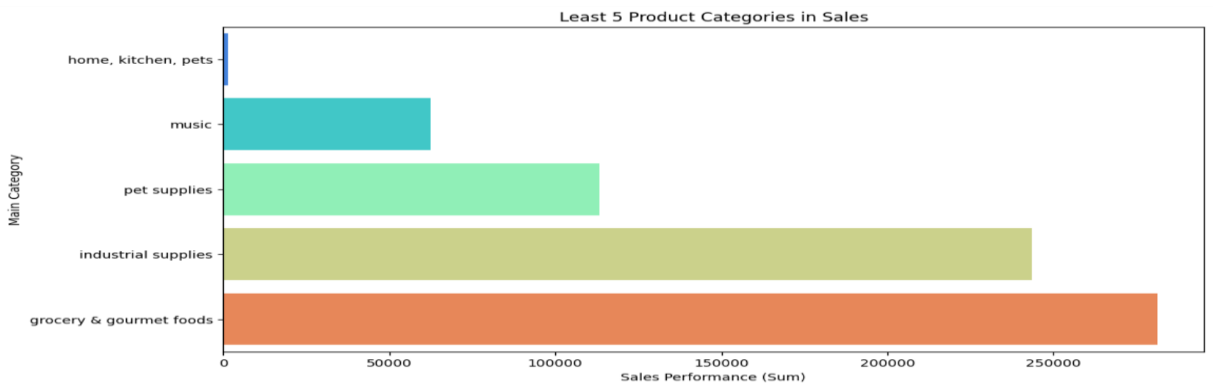
# Create a bar chart to show the least 5 main_category by sales_metric

least_5_categories=data.groupby('main_category')['sales_per'].sum().nsmallest(5).reset_index()

sb.barplot(data=least_5_categories,x='sales_per',y='main_category',palette='rainbow')

plt.show()

# Explain the least 5 main categories with the lowest sales_metric.



# Calculate profit margins and create a bar chart or list products with the highest and lowest margins

data['profit_margin'] = data['actual_price'] - data['discount_price'] highest_margin_products = data.nlargest(10, 'profit_margin')

lowest_margin_products = data.nsmallest(10, 'profit_margin')

plt.figure(figsize=(12, 6))

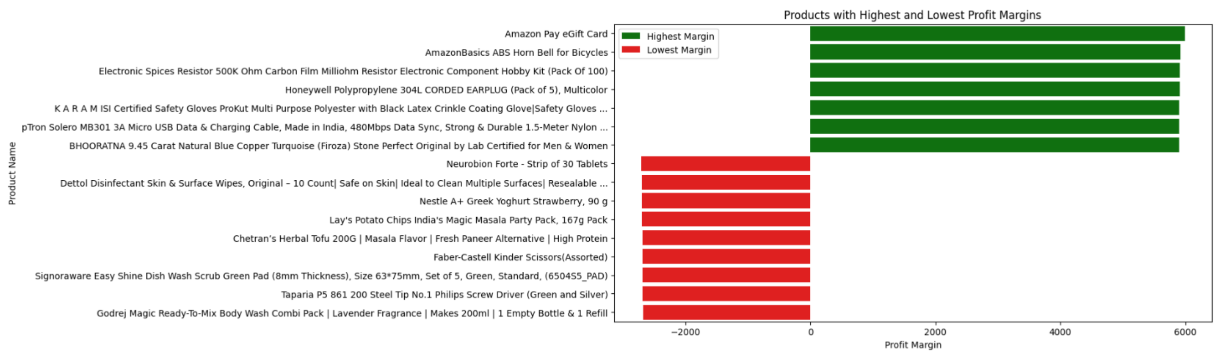
sb.barplot(data=highest_margin_products, x='profit_margin', y='name', color='green', label='Highest Margin')

sb.barplot(data=lowest_margin_products, x='profit_margin', y='name', color='red', label='Lowest Margin')

plt.legend()

plt.show()

# Explain the products with the highest and lowest profit margins.



# Calculate average discount percentage for products and analyze its impact on sales_metric

avg_discount_percentage = (data['discount_price'] / data['actual_price']).mean() * 100

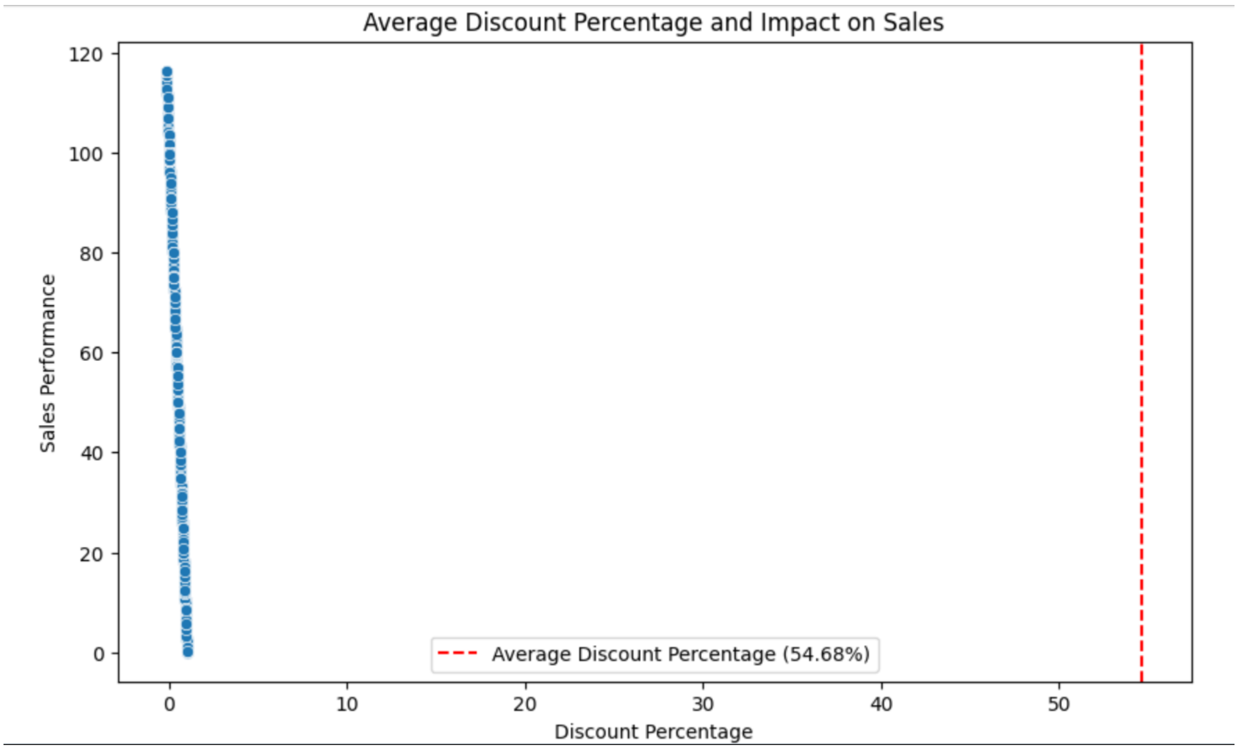
plt.figure(figsize=(10, 6))

sb.scatterplot(data=data, x='Discount_per', y='sales_per')

plt.axvline(x=avg_discount_percentage, color='red', linestyle='--', label=f'Average Discount Percentage ({avg_discount_percentage:.2f}%)')

plt.show()

# Explain the relationship between discount percentage and sales_metric.



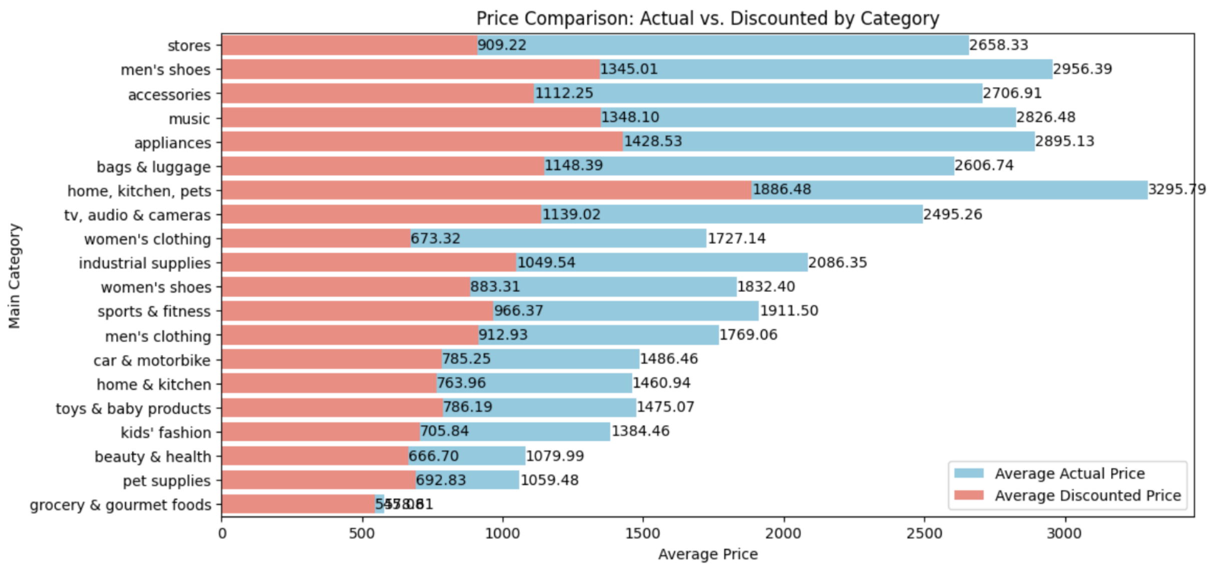
Price Comparison: Actual vs. Discounted by Category

```
avg_actual_prices = data.groupby('main_category')['actual_price'].mean().reset_index()
avg_discounted_prices=data.groupby('main_catecgory')['discount_price'].mean().reset_
index()
avg_price_diff = avg_actual_prices['actual_price'] -
avg_discounted_prices['discount_price']
avg_price_diff_df=pd.DataFrame({'main_category':avg_actual_prices['main_category']
, 'avg_price_diff': avg_price_diff})

max_diff_category=avg_price_diff_df.loc[avg_price_diff_df['avg_price_diff'].idxmax(
), 'main_category']
plt.figure(figsize=(12, 6))

# Define the order of categories based on average price difference
order = avg_price_diff_df.sort_values(by='avg_price_diff',
ascending=False)['main_category']

# Plotting average actual prices
ax = sb.barplot(data=avg_actual_prices, x='actual_price',
y='main_category',order=order ,label='Average Actual Price', color='skyblue')

# Plotting average discounted prices
```

```
sb.barplot(data=avg_discounted_prices, x='discount_price', y='main_category',
order=order,label='Average Discounted Price', color='salmon')
```

# Adding annotations for actual prices

```
for p in ax.patches:
    ax.annotate(f'{p.get_width():.2f}', (p.get_width(), p.get_y() + p.get_height() / 2),
                ha='left', va='center', color='black', fontsize=10)

plt.title('Price Comparison: Actual vs. Discounted by Category')
plt.xlabel('Average Price')
plt.ylabel('Main Category')
plt.legend()
plt.show()
```



Multi variate analysis

```
numeric_columns=data.select_dtypes(include="number").columns

sb.pairplot(data[numeric_columns])

numeric_columns = data.select_dtypes(include="number").columns
```

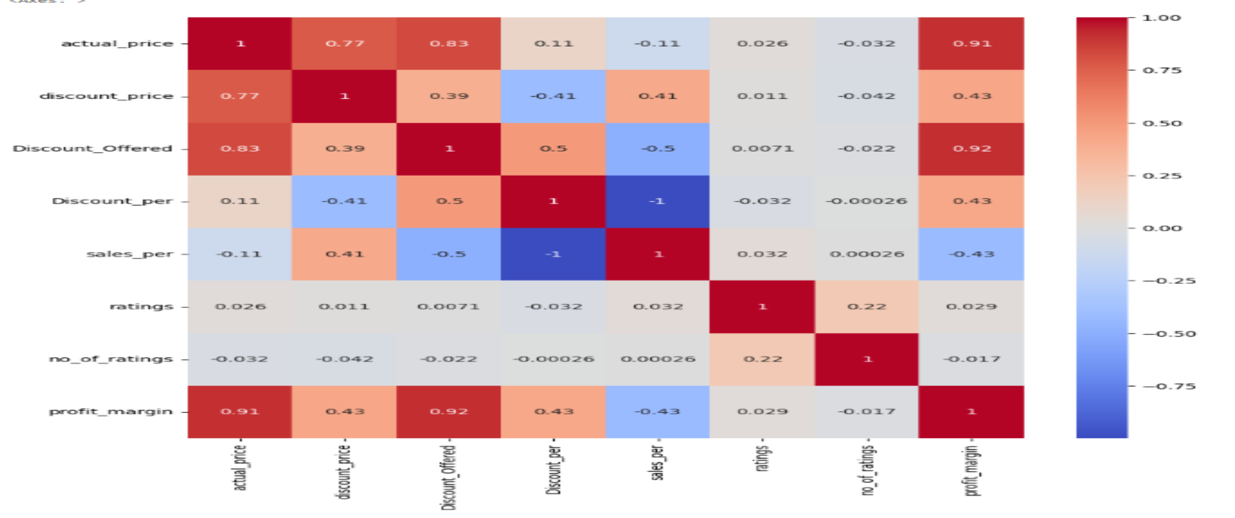
```
print(numeric_columns)
```

output:
```
Index(['actual_price', 'discount_price', 'Discount_Offered', 'Discount_per',
       'sales_per', 'ratings', 'no_of_ratings', 'profit_margin'], dtype='object')
```

#Correlation

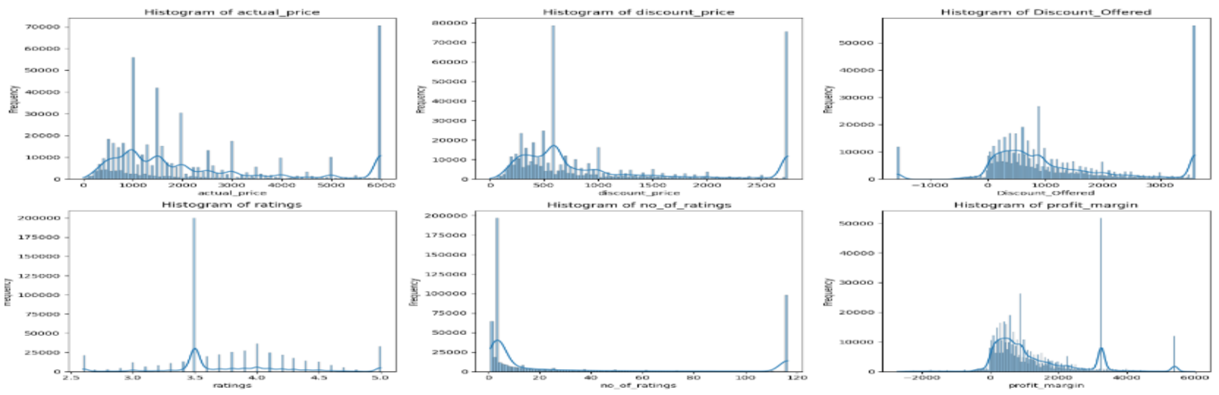
```
corr = data[numeric_columns].corr()
plt.figure(figsize=(10,10))
sb.heatmap(corr, annot=True, cmap='coolwarm')
```

```
<Axes: >
```



Distribution

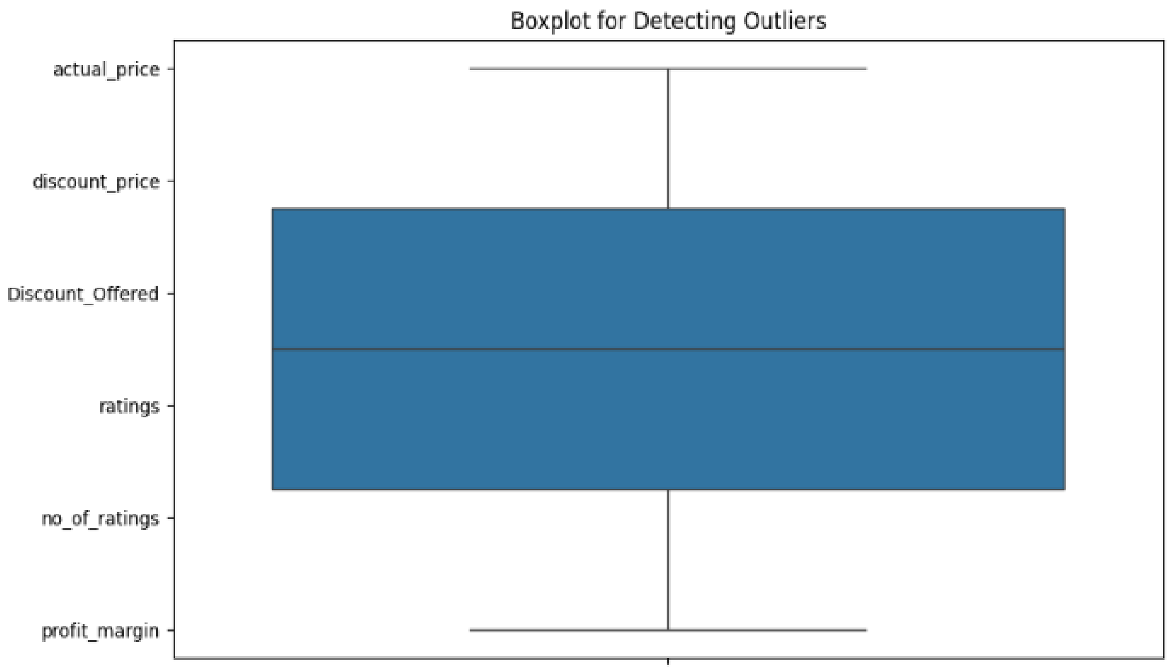
```
numerical_columns = ['actual_price', 'discount_price', 'Discount_Offered', 'ratings', 'no_of_ratings', 'profit_margin']
fig, axes = plt.subplots(nrows=2, ncols=3, figsize=(15, 10))
```



Log Transformation

```
from sklearn.preprocessing import StandardScaler
import numpy as np
# Log transformation for skewed numerical columns
data['log_ratings'] = data['ratings'].apply(lambda x: np.log(x) if x > 0 else 0)
data['log_discount_price'] = data['discount_price'].apply(lambda x: np.log(x) if x > 0 else 0)
# Scaling features for analysis (important for PCA, t-SNE, clustering)
scaler = StandardScaler()
scaled_data = scaler.fit_transform(data[['log_ratings', 'log_discount_price', 'actual_price', 'Discount_Offered', 'profit_margin']])
print(vif_data)  # Higher VIF (> 10) suggests multicollinearity
vif = 1. / (1. - r_squared_i)
Explained variance by components: [0.63004284 0.19990952]
import scipy.stats as stats
# Normality test for log-transformed ratings and discount price
print("Normality test for log_ratings:", stats.normaltest(data['log_ratings']))
print("Normalitytestforlog_discount_price:",stats.normaltest(data['log_discount_price']))
# Other features like actual_price can be similarly tested
Normalitytestforlog_ratings:
NormaltestResult(statistic=np.float64(3770.009681021775), pvalue=np.float64(0.0))
Normalitytestforlog_discount_price:
NormaltestResult(statistic=np.float64(22217.72913214297), pvalue=np.float64(0.0))
```

# Box plot to visualize outliers



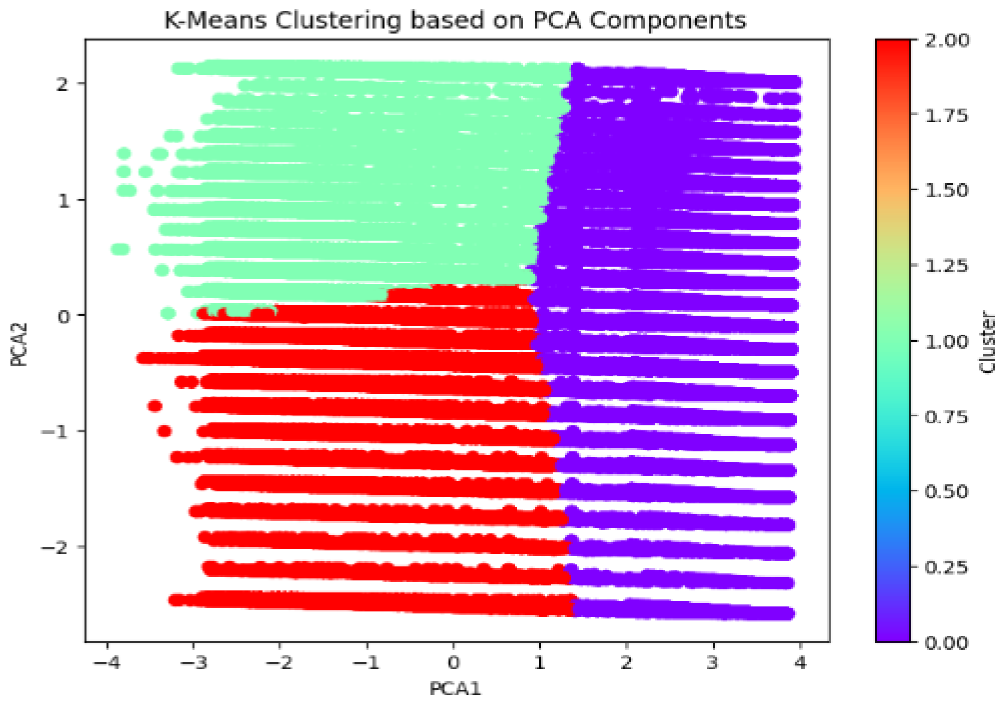
from sklearn.cluster import KMeans

# Applying K-Means clustering

kmeans = KMeans(n_clusters=3, random_state=42)

data['Cluster'] = kmeans.fit_predict(scaled_data)

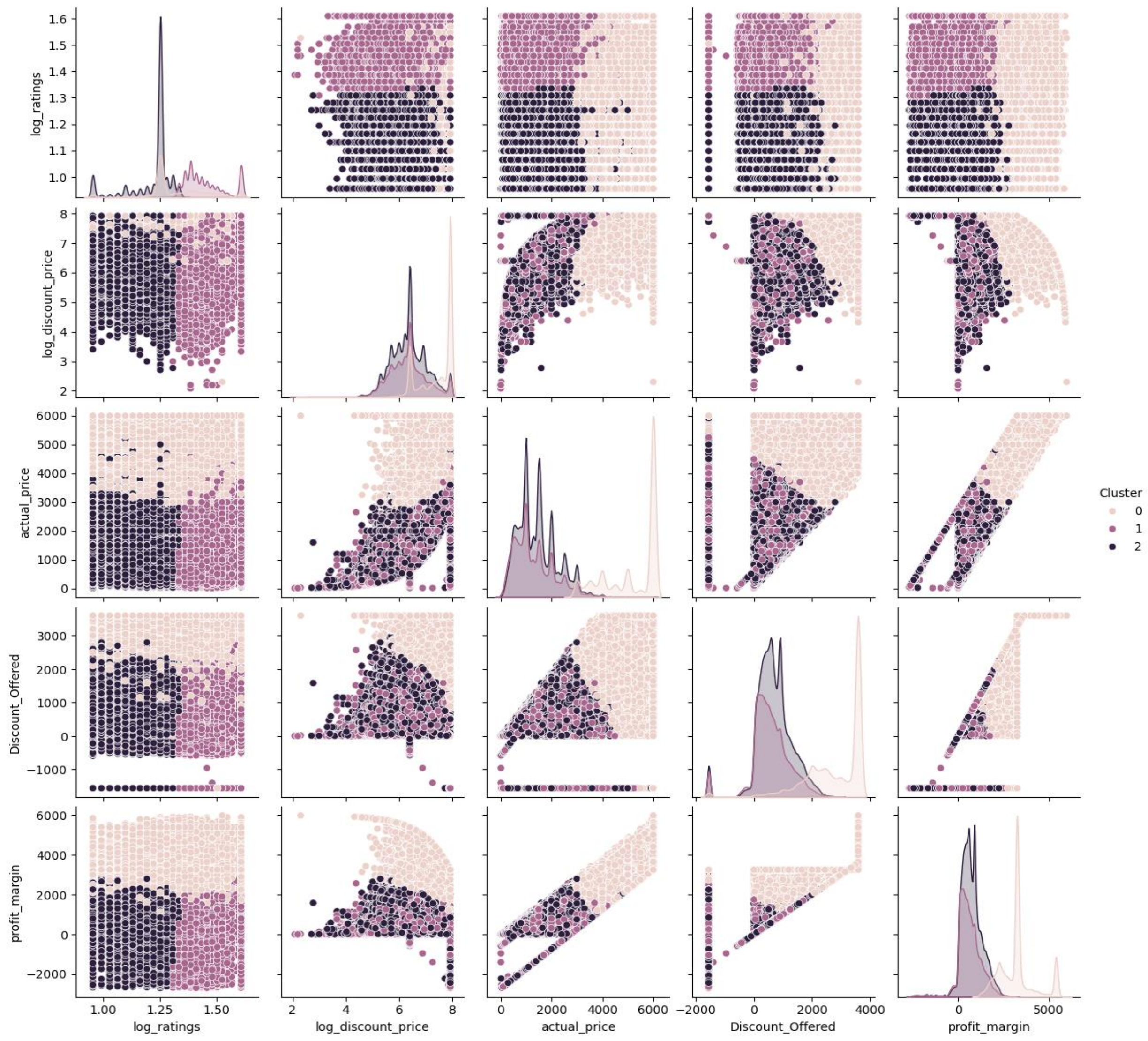
# Visualizing clusters with PCA components



# Pair plot to visualize relationships and interactions

sb.pairplot(data[['log_ratings', 'log_discount_price', 'actual_price', 'Discount_Offered', 'profit_margin', 'Cluster']], hue='Cluster')

plt.show()



Insight

Here by seeing the graph, we can say that non of the graph is normally distributed

- Actual_price, Discount_price, Discount_Offered, Ratings, No.of_ratings have right skewed data distribution because, we can observe that the frequency of the data is high in right hand side of graph

```
for column in numerical_columns:
    print(f"Statistics for {column}:")
    print(f"Mean: {data[column].mean()}")
    print(f"Median: {data[column].median()}")
    print(f"Standard Deviation: {data[column].std()}\n")
```

Statistics for actual_price:

Mean: 2273.2389967094828

Median: 1499.0

Standard Deviation: 1816.4056513326805

Statistics for discount_price:

Mean: 1013.372029696239

Median: 599.0

Standard Deviation: 842.3925436191876

Statistics for Discount_Offered:

Mean: 1148.4944043075864

Median: 802.0

Standard Deviation: 1165.1085040149635

Statistics for ratings:

Mean: 3.755463981072726

Median: 3.5

Standard Deviation: 0.5374235801895199

Statistics for no_of_ratings:

Mean: 32.35141637281652

Median: 4.0

Standard Deviation: 44.04783396922009

Statistics for profit_margin:

Mean: 1259.8669670132438

Median: 850.0

Standard Deviation: 1286.9314144607563

Insight

1. By using these statistical measures, we surely conclude that the data is distributes positive in the entire data set.
2. Because we observe that mean value is greater than median
3. According to positive skewness order of statistical measures
   - mode< median< mean

import scipy.stats as stats

# Identify distribution types

for column in numerical_columns:

print(f"Distribution type for {column}: {stats.normaltest(data[column])}")

Distributiontypeforactual_price:NormaltestResult(statistic=np.float64(71328.19009633732), pvalue=np.float64(0.0))

Distributiontypefordiscount_price:NormaltestResult(statistic=np.float64(80501.54805452723), pvalue=np.float64(0.0))

DistributiontypeforDiscount_Offered:NormaltestResult(statistic=np.float64(43129.065936540464), pvalue=np.float64(0.0))

Distributiontypeforratings: NormaltestResult(statistic=np.float64(17967.26587022675), pvalue=np.float64(0.0))

```
Distributiontypeforno_of_ratings:NormaltestResult(statistic=np.float64(88277.431347
58572), pvalue=np.float64(0.0))
Distributiontypeforprofit_margin:NormaltestResult(statistic=np.float64(96058.199145
22018), pvalue=np.float64(0.0))
```

AS p_val is less than 0.05 we reject the null hypothesis.so the data doesnot follow a normal distribution in of these columns

```
data['normalized_ratings'] = data['ratings'].apply(lambda x: np.log(x) if x > 0 else 0)
print(f"Distributiontypefornormalized_ratings:{stats.normaltest(data['normalized_ratin
gs'])}")
```

```
Distributiontypefornormalized_ratings:NormaltestResult(statistic=np.float64(3770.009
681021775), pvalue=np.float64(0.0))
```

```
print(f"Distributiontypefornormalized_ratings:{stats.normaltest(data['normalized_disco
unt_price'])}")
```

```
Distributiontypefornormalized_ratings:NormaltestResult(statistic=np.float64(22217.72
913214297), pvalue=np.float64(0.0))
```

Hypothesis Testing

1. ANOVA TESTING

The ANOVA (Analysis of Variance) test is a way to find out if survey or experiment results are significant

Outputs

F-statistics: Ratio of the variance between the groups to the variance within the groups

P-value: Probability of obtainging an F-statistic extreme than the observed value

A comparison of average sales percentages across different main categories using ANOVA test.

- we only get the result for "Is there any significant differance betweeen each group"
- but we cant specify which two group differ from each other **-** for this we need to perform post hoc test

Assumptions

Null hypothesis: There's no difference in the means of brands

Alternative hypothesis: Theres is a differnace in the means of brands

# Group the data by main category and calculate the mean sales percentage by each main category

```
main_cat_groups = data.groupby('main_category')['sales_per'].mean()
```

# Perform ANOVA test

```
f_val, p_val = stats.f_oneway(*[data.loc[data['main_category'] == category, 'sales_per'] for category in main_cat_groups.index])
```

# "*" is used for unpacking the list of series into separate arguments

# here we are only selecting the rows that is equal to category(loc used to select a subset of the dataframe based on certain condition)

```
f_val, p_val(np.float64(1797.4265173063473), np.float64(0.0))
```

Here we perform the one way ANOVA test, which tests the null hypothesis that two or more groups have the same populaltion mean.

```
print(p_val)
```

OUTPUT: 0.0

```
if p_val<0.05:
    print("reject null hypothesis")
else:
    print("accept null hypothesis")
```

Output:reject null hypothesis

So , we conclude that there is a differance in the means of the main categories

2. T-TEST

performing a two-sample t-test to compare the average discount offered on products between Brand A and Brand B

```
brand_list=list(data['Brand_Name'].unique())
['Lloyd', 'LG', 'Carrier', 'Voltas', 'Daikin', 'Panasonic', 'Whirlpool', 'Samsung', 'Godrej', 'Blue', 'IFB', 'Cruise', 'AmazonBasics', 'Haier', 'Hitachi', 'Amazon',
 'OGENERAL', 'Small', 'ALLWIN', 'Hexzone', 'Candy', 'O-General', 'ONIDA']
# print(brand_list)
print("Select any two from above")
Brand_A = input("Enter Brand A name(Check Capitals and small letters): ")
Brand_B = input("Enter Brand B name(Check Capitals and small letters): ")
if Brand_A in brand_list and Brand_B in brand_list:
    brand_a_data = data[data['Brand_Name'] == Brand_A]
    brand_b_data = data[data['Brand_Name'] == Brand_B]

    # Extract the discount offered for each brand
    discount_offered_a = brand_a_data['Discount_Offered']
    discount_offered_b = brand_b_data['Discount_Offered']
    # Perform the two-sample t-test
    t_statistic, p_value =stats.ttest_ind(discount_offered_a, discount_offered_b)

    # Print the results
    print('t-statistic:', t_statistic)
    print('p-value:', p_value)

    # Interpret the results
    if p_value < 0.05:
        print('There is a significant difference in the average discount offered on products between Brand A ({}) and Brand B ({}).'.format(Brand_A, Brand_B))
    else:
        print('There is no significant difference in the average discount offered on products between Brand A ({}) and Brand B ({}).'.format(Brand_A, Brand_B))
else:
   print("Give the brand names correctly, please go throught the list once again")
```

```
Select any two from above
Enter Brand A name(Check Capitals and small letters):  Pc
Enter Brand B name(Check Capitals and small letters):  Puma
t-statistic: -0.9164888334793503
p-value: 0.35945502023023657

There is no significant difference in the average discount offered on products between Brand A (Pc) and Brand B (Puma).
```

Conclusion

- if the p_val is less than 0.05 then statistically, the two brands offer similar discounts on their products
- else statistically, the two brands does not offer similar discounts on their products
-

3. PEARSON CORRELATION

```
# Perform the Pearson correlation test
correlation_coefficient, p_value = stats.pearsonr(data['sales_per'], data['no_of_ratings'])
# Print the results
print('Correlation coefficient:', correlation_coefficient)
print('p-value:', p_value)
if abs(correlation_coefficient) > 0.3 and p_value < 0.05:
   print('There is a weak to moderate significant relationship between the average sales percentage product and the number of ratings a product has.')
elif abs(correlation_coefficient) > 0.5 and p_value < 0.05:
   print('There is a moderate to strong significant relationship between the average sales percentage product and the number of ratings a product has.')
elif abs(correlation_coefficient) > 0.8 and p_value < 0.05:
   print('There is a strong significant relationship between the average sales percentage product and the number of ratings a product has.')
else:
   print('There is no significant relationship between the average sales percentage product and the number of ratings a product has.')
```

```
Correlation coefficient: 0.0002578443267975152
p-value: 0.8481359362952365
```

There is no significant relationship between the average sales percentage product and the number of ratings a product has.

Correlation coefficient: 0.0002578443267975152

p-value: 0.8481359362952365

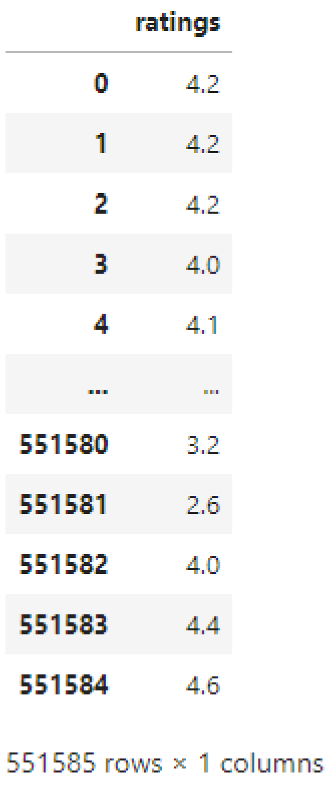
There is no significant relationship between the average sales percentage product and the number of ratings a product has.

The correlation coefficient measures the strength and direction of the linear relationship between two variables

- 0.0002 indicate a very weak correlation is statistically significant

CENTRAL LIMIT THEOREM

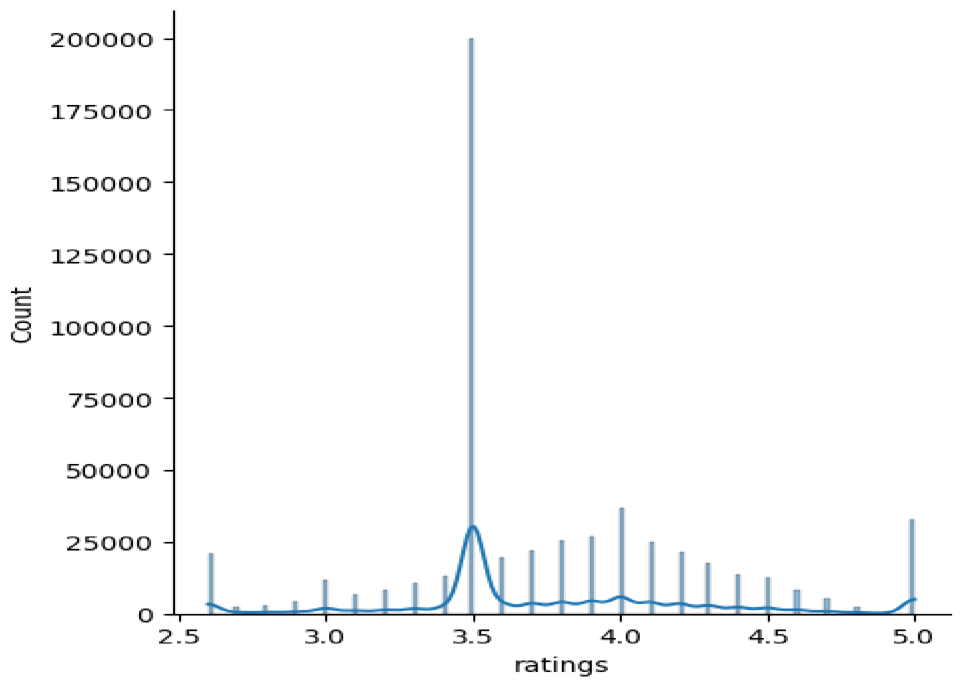
- df= data[['ratings']]
- df



	ratings
0	4.2
1	4.2
2	4.2
3	4.0
4	4.1
...	...
551580	3.2
551581	2.6
551582	4.0
551583	4.4
551584	4.6

551585 rows × 1 columns

```
sb.displot(df.ratings,kde=True)
plt.show()
```



Population Mean

```
df.ratings.mean()
(3.755463981072726)
```

This is the true mean weight of the population. This is the population parameter, the ground truth.

Let's take a random sample from this data, and see what mean we get.

Sample Mean

```
sample_size= 30
df.ratings.sample(sample_size).mean()
```

(3.6933333333333333)

Slight different from the population mean, right?

Let's take another sample.

```
df.ratings.sample(sample_size).mean()
```

(3.7866666666666666)

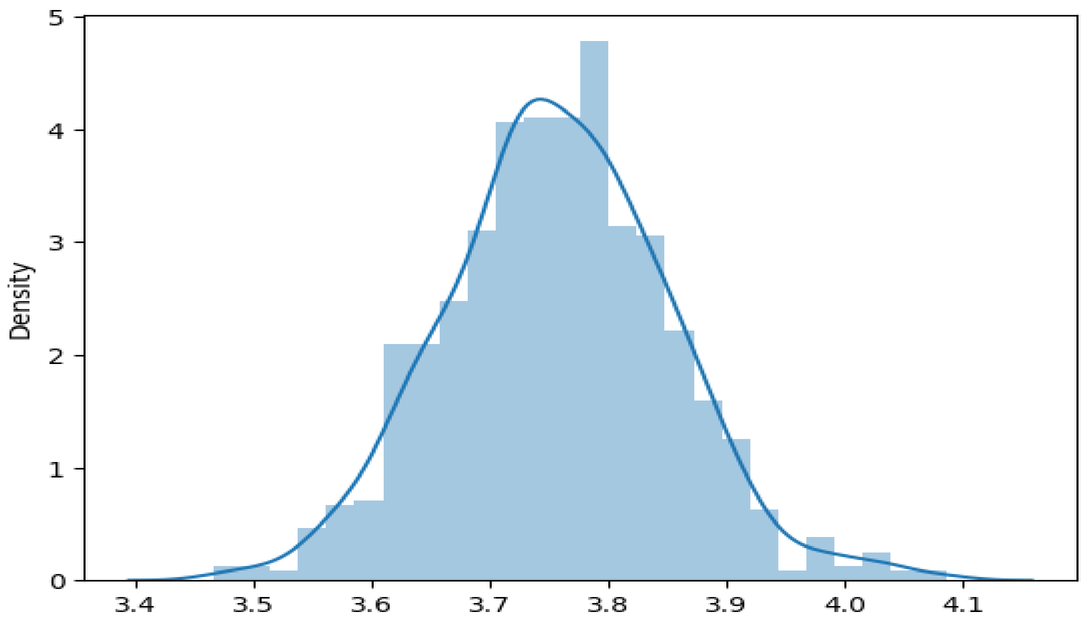
```
df.ratings.sample(sample_size).mean()
```

(3.7333333333333334)

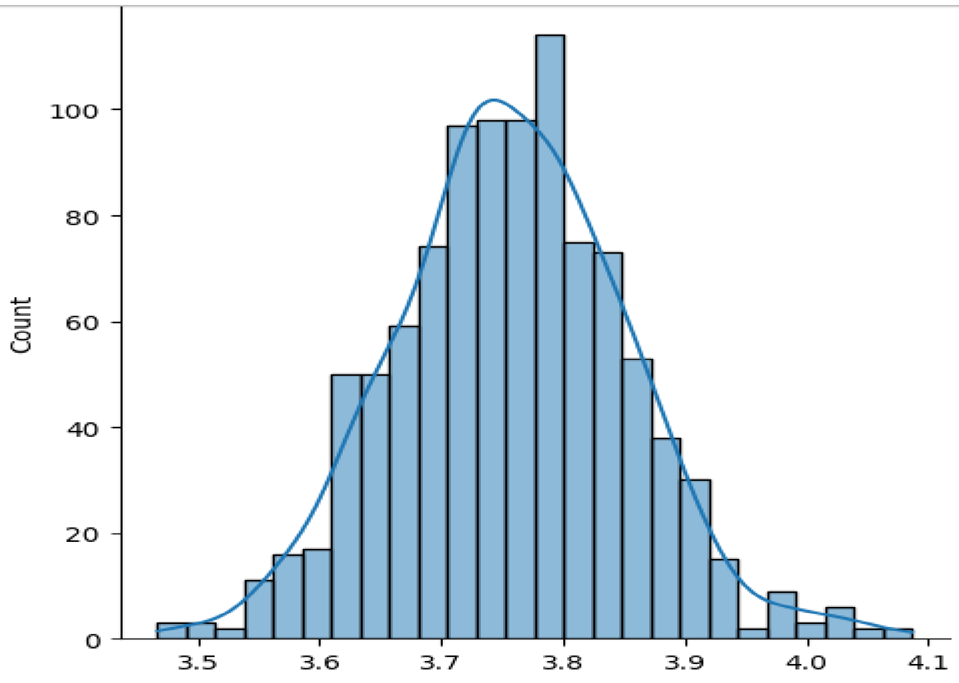
Each time we take a sample, our mean value is different. There is variability in the sample mean itself. Does the sample mean itself follow a distribution? Let'sassessthis.

We'll take many samples from the data, and plot a histogram of the same.

```
sample_means = [df.ratings.sample(sample_size).mean() for i in range(1000)]
sample_means = pd.Series(sample_means)
sb.distplot(sample_means)
plt.show()
```



```
sb.displot(sample_means,kde=True)

plt.show()
```



It is a distribution of sample means, which is a histogram with a Kernel Density Estimate (KDE) overlay. This type of distribution is known as a Sampling Distribution.

From the plot, it appears that the distribution of sample means follows a Normal Distribution (also known as a Gaussian Distribution), which is characterized by its bell shape and symmetry around the mean. This is in line with the Central Limit Theorem, which states that the distribution of sample means will approximate a normal distribution as the sample size becomes large, regardless of the shape of the population distribution.

The mean of this distribution is approximately 3.75, which is the peak of the bell curve. This suggests that the average rating across all samples is around 3.75. The spread of the distribution gives us an idea about the variability of the ratings. The narrower the bell curve, the less variability there is in the ratings.

Mean of the sample means

sample_means.mean()

(3.75751)

Standard deviation of your sample means

sample_means.std()

(0.09504537027809502)

standard error of the population

Population std vs. std of sampling mean

df.ratings.std()/np.sqrt(sample_size)

(0.09811967260166218)

conclusion:

This exploratory data analysis has provided valuable insights into the Amazon product sales data. The top selling brands are PC Company, Adidas, and LG, with accessories, men's clothing, and women's clothing being the most popular main categories. The sub-categories with the highest product counts are Shirts, Sports Shoes, Jeans, Western Wear, and Men's Fashion.Strategic pricing and discounting strategies are evident, with actual prices scattered across a wide range and discounted prices clustering around certain price points. The average discount percentage across all products is around 50%. There is a positive correlation between product ratings and sales performance, indicating that higher rated products tend to sell better.The most competitive main categories in terms of number of unique products are men's clothing, women's clothing, and accessories. The least competitive categories with the lowest sales performance are music, home, kitchen, pets, and industrial supplies. Appliances and sports & fitness have higher average ratings compared to other categories.While the dataset provides a comprehensive view of the Amazon product landscape, it lacks information about actual sales volume or revenue, which would provide a more accurate measure of sales performance. Incorporating additional features like product descriptions, images, and seller information could yield further insights.Applying machine learning techniques to predict sales based on product attributes and customer preferences could help optimize product selection and pricing strategies. Analyzing the impact of discounts on sales and identifying the optimal discount levels for different product categories could also provide valuable insights.Overall, this exploratory data analysis has uncovered valuable insights about the Amazon product sales landscape, which can inform business decisions and guide future research directions. The success of this project lies in the ability to clean and preprocess a messy dataset and use data visualization tools to create clear, insightful graphic

REFERENCES:

McKinney, W. (2017). Python for Data Analysis: Data Wrangling with Pandas, NumPy, and IPython. O'Reilly Media.

This book offers an introduction to using Python for data manipulation and analysis, with a focus on Pandas and NumPy.

Seaborn Documentation. Retrieved from https://seaborn.pydata.org/

Official documentation for Seaborn, a Python library for creating visualizations, ideal for EDA in Python.

Python Crash Course for Data Science by Corey Schafer (YouTube series).

YouTube tutorials explaining data analysis concepts using Python, covering Pandas, Seaborn, and Matplotlib for beginners.

Chawla, D., & Singh, D. (2019). Consumer behavior towards online shopping: An empirical study of Delhi. Journal of Business and Retail Management Research, 13(2), 46-61.

This study examines factors influencing consumer satisfaction and behavior in online shopping, which can help in interpreting customer ratings and reviews.

DataCamp's "Exploratory Data Analysis with Python" Course. Retrieved from https://www.datacamp.com/

An interactive course for students to learn data analysis techniques in Python, including data cleaning, transformation, and visualization.

Matplotlib and Seaborn Tutorial on Real Python. Retrieved from https://realpython.com/

Comprehensive tutorials on creating data visualizations with Python, useful for representing trends in sales data.

Kaggle (Dataset Resource). https://www.kaggle.com/

Kaggle offers open-source datasets, including the Amazon product dataset, and community notebooks that demonstrate data cleaning and analysis techniques.

Pandas Documentation. Retrieved from https://pandas.pydata.org/

Official documentation for Pandas, covering data handling, cleaning, and basic analysis tasks for beginners.

Statistics for Data Science Tutorial on Dataquest. Retrieved from https://www.dataquest.io/

This tutorial introduces basic statistical techniques for data analysis, covering concepts like central tendency and correlations.

"A Gentle Introduction to Exploratory Data Analysis" on Towards Data Science. Retrieved from https://towardsdatascience.com/

A beginner-friendly guide to EDA, with a step-by-step approach to data inspection, cleaning, and visualization.

Field, A. (2013). Discovering Statistics Using SPSS. Sage Publications.
Introduces students to statistical methods like correlation and ANOVA in an approachable way, which is useful for analyzing relationships in datasets.

YouTube: "Data Cleaning and Preprocessing with Python" by freeCodeCamp.
A hands-on tutorial that walks students through data cleaning and preprocessing using Python, including handling missing data and outliers.
Jupyter Notebooks on Google Colab. Retrieved from https://colab.research.google.com/
An accessible platform for students to practice coding and data analysis, especially helpful for running EDA tasks in a browser environment.
Few, S. (2012). Show Me the Numbers: Designing Tables and Graphs to Enlighten. Analytics Press.
An introductory guide to creating effective graphs and tables, emphasizing clarity and insight in visualizations.
"Introduction to Data Visualization with Python" by DataCamp. Retrieved from https://www.datacamp.com/
Covers different types of visualizations, ideal for students looking to understand and represent data patterns through graphs and charts.
Google,Kaggle,chatgpt.

GITHUB REPOSITORY LINK: