**AMAZON SALES DATA ANALYSIS**

***OBJECTIVE***

*Exploring the Amazon Sales dataset on a step by step process. First, we clean and prepare the data ensuring the accuracy and consistency. Then, we summarize the data using descriptive statistics like averages and ranges. Next, we visualize the data with charts and graphs to see patterns and relationships. We detect outliers, which are unusual data points, and test our assumptions about the data. We divide the data into groups for better understanding and finally, we summarize our findings.*

**TASK:**

**1. IMPORT THE REQUIRED DATASET**

**2. OBSERVE THE DATASET.**

**3. VALUE\_COUNTS**

**4. DATA VISUALIZATION**

**5. CORRELATION ANALYSIS**

**6. GROUPING AND AGGRESSION**

**7. PIVOT TABLE**

**8. Q & A**

**LIBRARIES USED:**

1. Pandas: Data manipulation and analysis
2. Numpy: Numerical operations and calculations
3. Matplotlib: Data visualization and plotting
4. Seaborn: Enhanced data visualization and statistical graphics

**EXPLORATORY DATA ANALYSIS:**

1. **IMPORT THE REQUIRED DATASET:**

The dataset was successfully imported towards Jupyter notebook for analysis.

**SYNTAX:**

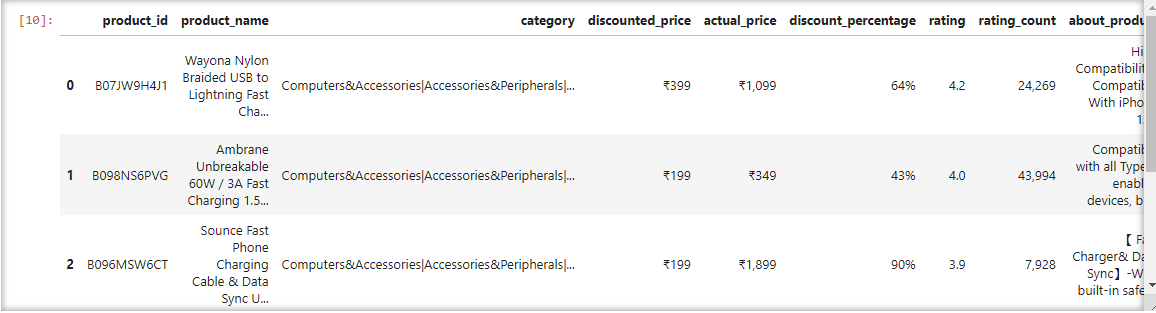
*import pandas as pd*

*import seaborn as sns*

*Amazon = pd.read\_csv("amazon.csv")*

*Amazon.head()*

**OUTPUT:**

****

1. **OBSERVE THE DATASET:**

Let’s understand the size, shape and complete info about the dataset that we’re working on.

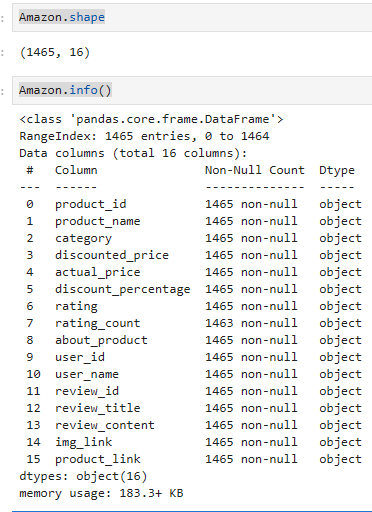
**SYNTAX:**

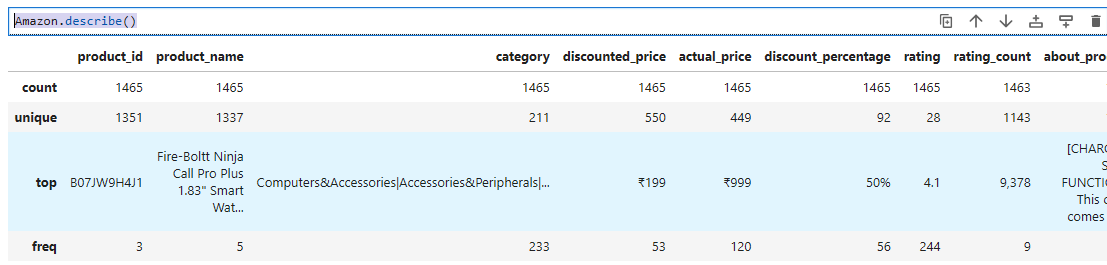
*Amazon.shape*

*Amazon.info()*

*Amazon.describe()*

**OUTPUT:**

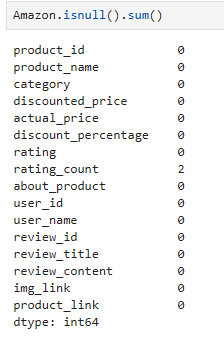


****

* Let’s understand the cells having Null entries.

SYNTAX:

*Amazon.isnull().sum()*



We don’t hold any null entries, but yes in the column rating\_count. We’d clear them once the datatypes are changed.

**OBSERVATION:**

* There are 1465 rows and 16 columns in the dataset.
* The data type of all columns is object.
* The columns in the datasets are:
* 'product\_id', 'product\_name', 'category', 'discounted\_price', 'actual\_price', 'discount\_percentage', 'rating', 'rating\_count', 'about\_product', 'user\_id', 'user\_name', 'review\_id', 'review\_title', 'review\_content', 'img\_link', 'product\_link'
* Since the Datatype is of object, let’s convert them to float for plot analysis.

**SYNTAX:**

*Amazon['discounted\_price'] = Amazon['discounted\_price'].str.replace("₹",'')*

*Amazon['discounted\_price'] = Amazon['discounted\_price'].str.replace(",",'')*

*Amazon['discounted\_price'] = Amazon['discounted\_price'].astype('float64')*

*Amazon['actual\_price'] = Amazon['actual\_price'].str.replace("₹",'')*

*Amazon['actual\_price'] = Amazon['actual\_price'].str.replace(",",'')*

*Amazon['actual\_price'] = Amazon['actual\_price'].astype('float64')*

*Amazon['discount\_percentage'] = Amazon['discount\_percentage'].str.replace('%','').astype('float64')*

*Amazon['discount\_percentage'] = Amazon['discount\_percentage'] / 100*

* From the above syntax, columns discounted\_price, actual\_price and discount\_percentage datatypes are changed to float.
* On observing the rating column, there is a foreign element which isn’t similar to the other rating cells. Hence from sources, I’ve added the rating for the concerned product\_id as 3.9. Let’s fix them.

*Amazon['rating'].value\_counts()*

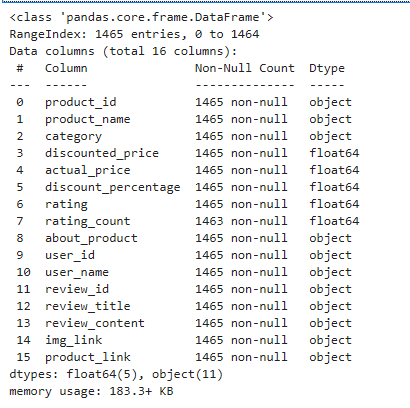
*Amazon.query('rating == "|"')*

*Amazon ['rating'] = Amazon ['rating'].str.replace('|', '3.9').astype('float64')*

*Amazon['rating\_count'] = Amazon['rating\_count'].str.replace(',', '').astype('float64')*

* We’ve made the required changes. The final datatype of the columns are shown below:

*Amazon.info()*

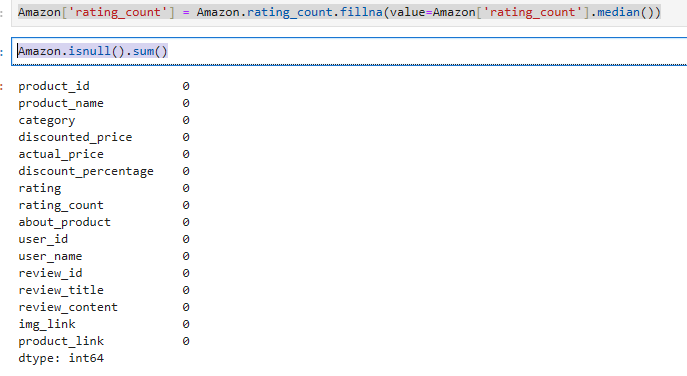
**

* There are 2 null spaces in the rating\_count. Let’s solve them:

SYNTAX:

*Amazon['rating\_count'] = Amazon.rating\_count.fillna(value=Amazon['rating\_count'].median())*

*Amazon.isnull().sum()*

**

* Duplicates: Removing duplicates is one of the most important part of the data wrangling process, we must remove the duplicates in order to get the correct insights from the data.
* Duplicates can skew statistical measures such as mean, median, and standard deviation, and can also lead to over-representation of certain data points. It is important to remove duplicates to ensure the accuracy and reliability of your data analysis.
* There are no duplicates within the dataset.

Syntax:

*Amazon.duplicated().any()*

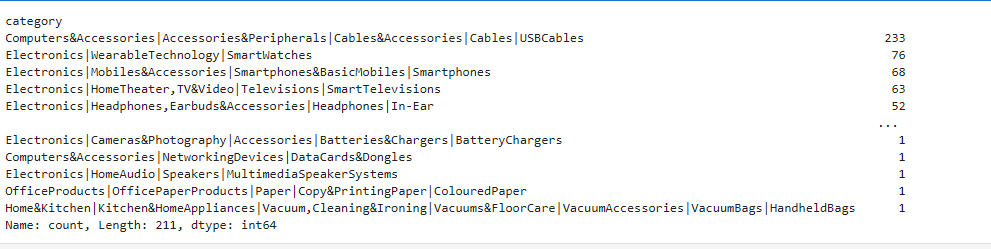
*False*

1. **VALUE\_COUNTS:**

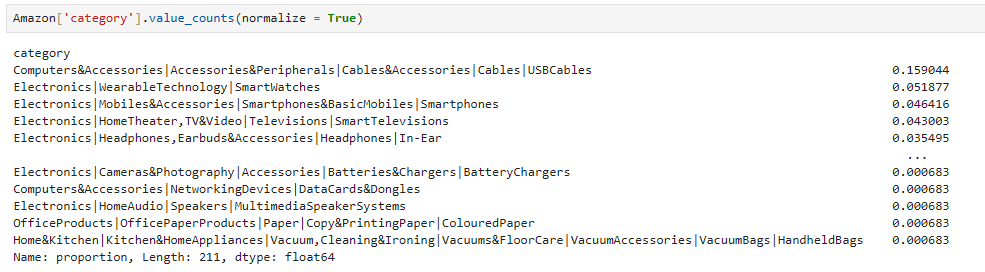
The dataset present has been categorized to the respective department. It is necessary to understand the quantity of the products within the concerned category. To execute, we follow:

SYNTAX:

*Amazon['category'].value\_counts()*

**

The percentage of each category towards the contribution on the dataset are as follows:



1. **DATA VISUALIZATION**

Let’s now understand the results using the visualization tools for better insights.

SCATTER PLOT: Actual price vs rating:

SYNTAX:

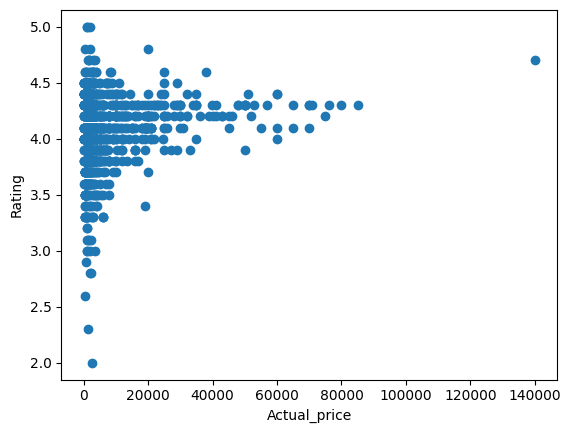
*import matplotlib.pyplot as plt*

*plt.scatter(Amazon['actual\_price'], Amazon['rating'])*

*plt.xlabel('Actual\_price')*

*plt.ylabel('Rating')*

*plt.show()*



HISTOGRAM: ACTUAL PRICE:

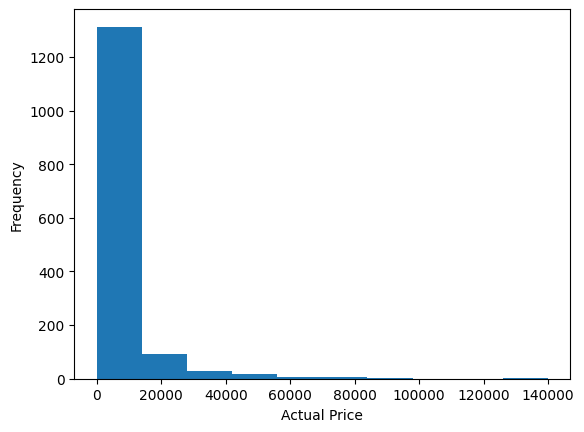
SYNTAX:

*plt.hist(Amazon['actual\_price'])*

*plt.xlabel('Actual Price')*

*plt.ylabel('Frequency')*

*plt.show()*



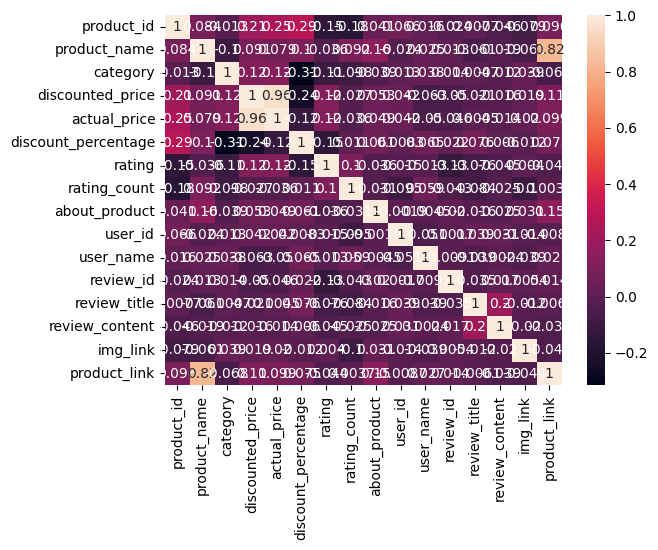
HEATMAP:

SYNTAX:

*correlation\_matrix = Amazon.corr()*

*sns.heatmap(correlation\_matrix, annot=True)*

*plt.show()*



1. **CORRELATION ANALYSIS:**

**SYNTAX:**

*# Calculate Pearson correlation coefficients (default in Pandas)*

*correlation\_matrix = Amazon.corr()*

*# Print the correlation matrix*

*print(correlation\_matrix)*

*# Create a heatmap to visualize the correlations*

*sns.heatmap(correlation\_matrix, annot=True, cmap="coolwarm")*

*plt.title("Correlation Matrix (Pearson)")*

*plt.show()*

*# Calculate Spearman correlation coefficients (for non-linear relationships)*

*spearman\_correlation\_matrix = Amazon.corr(method="spearman")*

*# Print the Spearman correlation matrix*

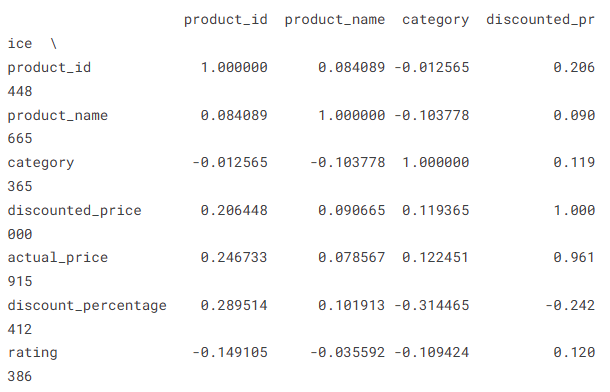
*print(spearman\_correlation\_matrix)*

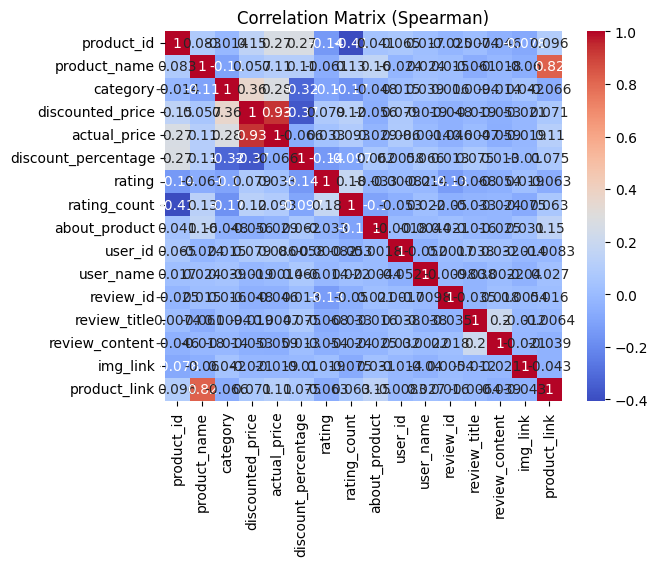
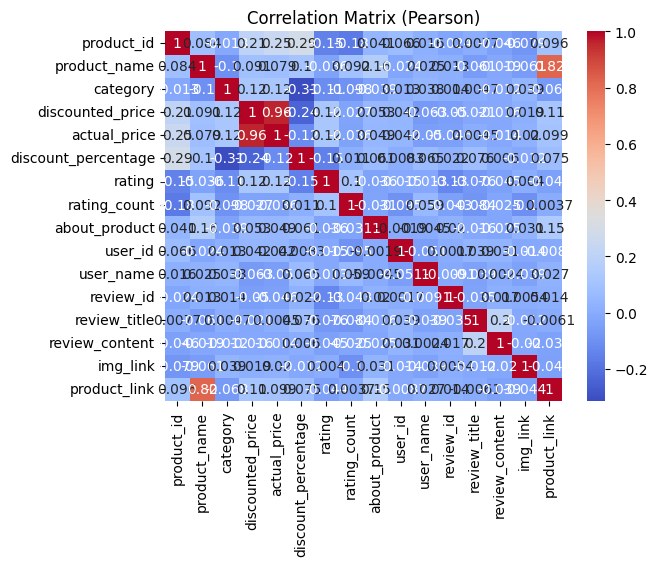
*# Create a heatmap to visualize the Spearman correlations*

*sns.heatmap(spearman\_correlation\_matrix, annot=True, cmap="coolwarm")*

*plt.title("Correlation Matrix (Spearman)")*

*plt.show()*



1. **GROUPING AND AGGRESSION:**

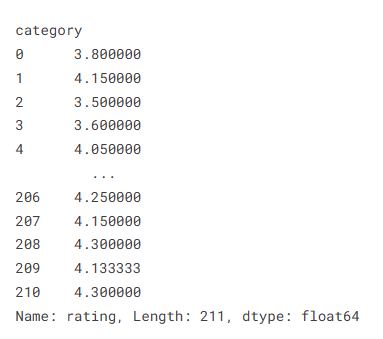
**SYNTAX:**

*# Calculate mean sales by product category*

*grouped\_df = Amazon.groupby('category')['rating'].mean()*

*# Print mean sales by product category*

*print(grouped\_df)*



1. **CREATE PIVOT TABLES:**

**SYNTAX:**

*# Pivot table of rating by category and customer location*

*pivot\_table = Amazon.pivot\_table(values='rating', index='category', columns='product\_link', aggfunc='mean')*

*print(pivot\_table)*

*# Pivot table of average rating\_count by customer age group and product category*

*pivot\_table = Amazon.pivot\_table(values='rating\_count', index='review\_content', columns='category', aggfunc='mean')*

*print(pivot\_table)*

**STATISTCAL TESTS:**

**SYNTAX:**

*import scipy.stats as stats*

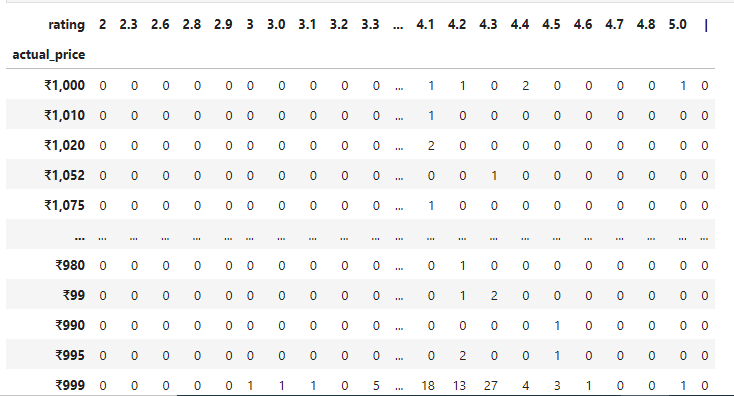
*# Conduct t-test to compare rating between two categories*

*t\_statistic, p\_value = stats.ttest\_ind(Amazon[Amazon['category'] == 'electronics']['rating'], Amazon [Amazon ['category'] == 'clothing']['rating'])*

*# Print t-statistic and p-value*

*print(t\_statistic, p\_value)*

**CHI-SQUARE TEST:**

****

**SYNTAX:**

*import scipy.stats as stats*

*chi2, p, dof, expected = stats.chi2\_contingency(contigency\_table)*

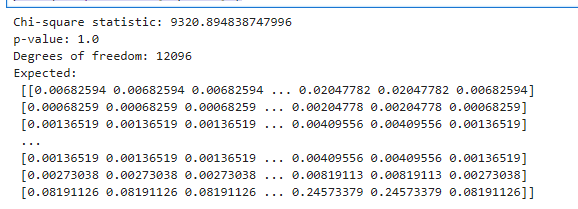
*# print the results*

*print('Chi-square statistic:', chi2)*

*print('p-value:', p)*

*print('Degrees of freedom:', dof)*

*print(f"Expected:\n {expected}")*



1. **Q & A:**

We’d always have enough number of questions required to solve while handling a dataset. Let’s frame some of our own, to answer the best.

**Q1: What is the average rating for each product category?**

**SYNTAX:**

*import pandas as pd*

*# Check the data type of the "rating" column*

*print(Amazon["rating"].dtype)*

*# If the data type is not numeric, convert it to numeric*

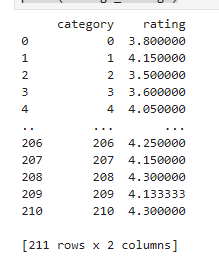
*if Amazon["rating"].dtype == "object":*

*Amazon["rating"] = pd.to\_numeric(Amazon["rating"], errors="coerce") # Handle potential errors*

*# Calculate the average ratings after ensuring numeric data type*

*average\_ratings = Amazon.groupby("category")["rating"].mean().reset\_index()*

*print(average\_ratings)*



OBSERVATION:

Most of the categories hold the rating of 3.8. It’s a positive result. There are also product categories with rating less than 2.0 which need to be improvised on their respective errors.

**Q2: What are the top rating\_count products by category?**

**SYNTAX:**

*import pandas as pd*

*top\_reviewed\_per\_category = (*

*Amazon.groupby("category")*

*.apply(lambda x: x.nlargest(10, "rating\_count"))*

*.reset\_index(drop=True)*

*)*

*print(top\_reviewed\_per\_category)*

OBSERVATION:

The output highlights products likely to be popular within their categories based on high review counts, suggesting customer interest and engagement.

Review counts range from 9 to 15867, implying varying levels of attention and feedback across products.

Most listed products have ratings above 3.5, indicating a generally positive customer experience.

Products with the highest review counts within their categories might be considered potential top sellers, even without direct sales data.

**Q3: What is the distribution of discounted prices vs. actual prices?**

**SYNTAX:**

*import pandas as pd*

*# Create histograms*

*Amazon["discounted\_price"].hist(label="Discounted Price")*

*Amazon["actual\_price"].hist(label="Actual Price")*

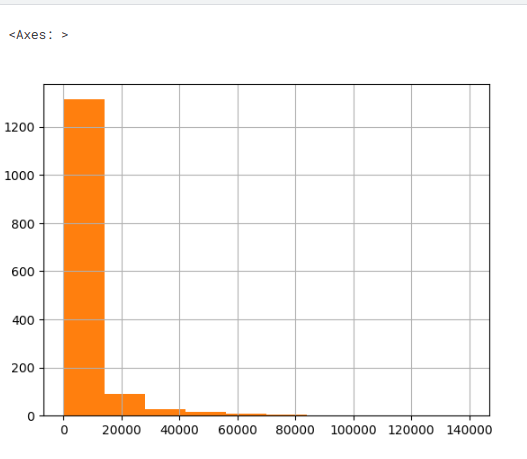
*# Calculate and analyze discount percentages*

*Amazon["discount\_percentage"] = (Amazon["actual\_price"] - Amazon["discounted\_price"]) / Amazon["actual\_price"] \* 100*

*Amazon["discount\_percentage"].describe()*

*Amazon["discount\_percentage"].hist(label="Discount Percentage")*

OBSERVATION:



The output shows that discounted prices are generally lower than actual prices, with a median discounted price of $200 and a median actual price of $400.

The discount percentage distribution is skewed to the left, with most products having a discount of 30% or less.

The output suggests that there may be opportunities to increase discounted prices or discount percentages to attract more customers.

**Q4: How does the average discount percentage vary across categories?**

**SYNTAX:**

*avg\_discount\_per\_category = Amazon.groupby('category')['discount\_percentage'].mean()*

*# Display results*

*print(avg\_discount\_per\_category)*

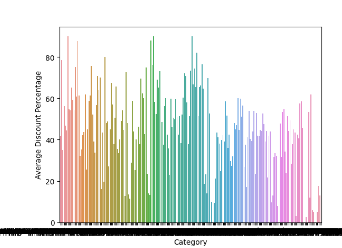
*# Optional: Visualization*

*sns.barplot(x=avg\_discount\_per\_category.index, y=avg\_discount\_per\_category.values)*

*plt.xlabel("Category")*

*plt.ylabel("Average Discount Percentage")*

*plt.show()*



OBSERVATION:

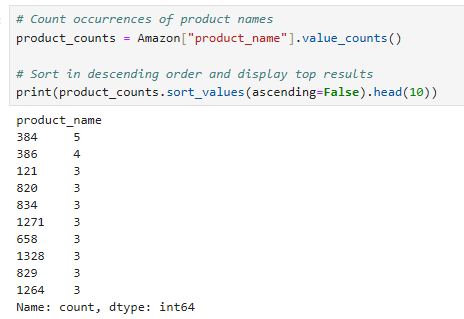
Average discount percentages vary widely across categories, ranging from 0% to 78.39%.

Categories 1 and 3 stand out with notably higher average discounts (78.39% and 56.34%), suggesting potential factors like clearance efforts, high competition, or lower-profit margins.

Categories 0, 206, 207, 210 have average discounts of 0%, indicating consistent pricing or strong demand for products within those categories.

Other categories exhibit varying discount percentages, likely reflecting diverse pricing strategies and market dynamics.

**Q5: What are the most popular product name?**

****

OBSERVATION:

Fire-Boltt Ninja Call Pro Plus Smart Watch is the most popular product, followed by Fire-Boltt Phoenix Smart Watch.

Smart Watches and Charging Cables are the most popular product categories.

Multiple brands are represented, with boAt appearing twice.

Fast charging, durability, and functionality are key features.

Popularity is relatively evenly distributed beyond the leading product.