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:

[10]:

	product_id	product_name	category	discounted_price	actual_price	discount_percentage	rating	rating_count	about_produ
0	B07JW9H4J1	Wayona Nylon Braided USB to Lightning Fast Cha...	Computers&Accessories|Accessories&Peripherals|...	₹399	₹1,099	64%	4.2	24,269	Hi Compatibilit Compatil With iPho 1
1	B098NS6PVG	Ambrane Unbreakable 60W / 3A Fast Charging 1.5...	Computers&Accessories|Accessories&Peripherals|...	₹199	₹349	43%	4.0	43,994	Compatil with all Type enabl devices, b
2	B096MSW6CT	Sounce Fast Phone Charging Cable & Data Sync U...	Computers&Accessories|Accessories&Peripherals|...	₹199	₹1,899	90%	3.9	7,928	【 F Charger& D Sync】 -W built-in safe

2. 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:

```
: Amazon.shape

: (1465, 16)

: Amazon.info()

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1465 entries, 0 to 1464
Data columns (total 16 columns):
 #   Column               Non-Null Count  Dtype
---  ------               --------------  -----
 0   product_id           1465 non-null   object
 1   product_name         1465 non-null   object
 2   category             1465 non-null   object
 3   discounted_price     1465 non-null   object
 4   actual_price         1465 non-null   object
 5   discount_percentage  1465 non-null   object
 6   rating               1465 non-null   object
 7   rating_count         1463 non-null   object
 8   about_product        1465 non-null   object
 9   user_id              1465 non-null   object
 10  user_name            1465 non-null   object
 11  review_id            1465 non-null   object
 12  review_title         1465 non-null   object
 13  review_content       1465 non-null   object
 14  img_link             1465 non-null   object
 15  product_link         1465 non-null   object
dtypes: object(16)
memory usage: 183.3+ KB
```

```
Amazon.describe()
```

	product_id	product_name	category	discounted_price	actual_price	discount_percentage	rating	rating_count	about_pro
count	1465	1465	1465	1465	1465	1465	1465	1463	
unique	1351	1337	211	550	449	92	28	1143	
top	B07JW9H4J1	Fire-Boltt Ninja Call Pro Plus 1.83" Smart Wat...	Computers&Accessories|Accessories&Peripherals|...	₹199	₹999	50%	4.1	9,378	[CHAR(S FUNCTI(This c comes
freq	3	5	233	53	120	56	244	9	

- Let’s understand the cells having Null entries.
  SYNTAX:
  Amazon.isnull().sum()

```
Amazon.isnull().sum()
```

```
product_id            0
product_name          0
category              0
discounted_price      0
actual_price          0
discount_percentage   0
rating                0
rating_count          2
about_product         0
user_id               0
user_name             0
review_id             0
review_title          0
review_content        0
img_link              0
product_link          0
dtype: int64
```

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()

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1465 entries, 0 to 1464
Data columns (total 16 columns):
 #   Column              Non-Null Count  Dtype
---  ------              --------------  -----
 0   product_id          1465 non-null   object
 1   product_name        1465 non-null   object
 2   category            1465 non-null   object
 3   discounted_price    1465 non-null   float64
 4   actual_price        1465 non-null   float64
 5   discount_percentage 1465 non-null   float64
 6   rating              1465 non-null   float64
 7   rating_count        1463 non-null   float64
 8   about_product       1465 non-null   object
 9   user_id             1465 non-null   object
 10  user_name           1465 non-null   object
 11  review_id           1465 non-null   object
 12  review_title        1465 non-null   object
 13  review_content      1465 non-null   object
 14  img_link            1465 non-null   object
 15  product_link        1465 non-null   object
dtypes: float64(5), object(11)
memory usage: 183.3+ KB
```

- 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()

```
Amazon['rating_count'] = Amazon.rating_count.fillna(value=Amazon['rating_count'].median())
```

```
Amazon.isnull().sum()
```

```
product_id            0
product_name          0
category              0
discounted_price      0
actual_price          0
discount_percentage   0
rating                0
rating_count          0
about_product         0
user_id               0
user_name             0
review_id             0
review_title          0
review_content        0
img_link              0
product_link          0
dtype: int64
```

- 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

3. <u>VALUE_COUNTS:</u>

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()

```
category
Computers&Accessories|Accessories&Peripherals|Cables&Accessories|Cables|USBCables                                  233
Electronics|WearableTechnology|SmartWatches                                                                        76
Electronics|Mobiles&Accessories|Smartphones&BasicMobiles|Smartphones                                               68
Electronics|HomeTheater,TV&Video|Televisions|SmartTelevisions                                                      63
Electronics|Headphones,Earbuds&Accessories|Headphones|In-Ear                                                       52
                                                                                                                 ...
Electronics|Cameras&Photography|Accessories|Batteries&Chargers|BatteryChargers                                      1
Computers&Accessories|NetworkingDevices|DataCards&Dongles                                                          1
Electronics|HomeAudio|Speakers|MultimediaSpeakerSystems                                                            1
OfficeProducts|OfficePaperProducts|Paper|Copy&PrintingPaper|ColouredPaper                                          1
Home&Kitchen|Kitchen&HomeAppliances|Vacuum,Cleaning&Ironing|Vacuums&FloorCare|VacuumAccessories|VacuumBags|HandheldBags    1
Name: count, Length: 211, dtype: int64
```

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

```
Amazon['category'].value_counts(normalize = True)
```

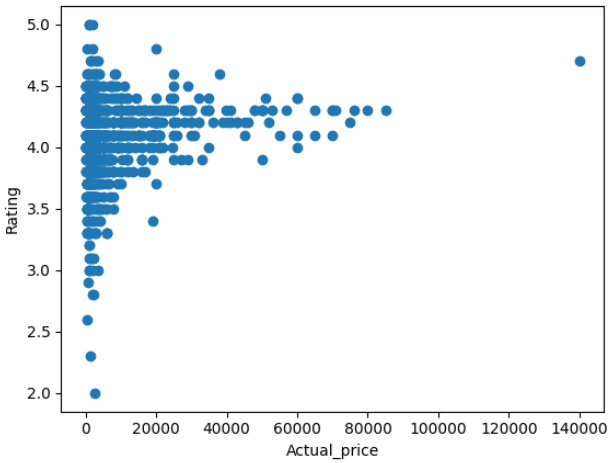
```
category
Computers&Accessories|Accessories&Peripherals|Cables&Accessories|Cables|USBCables                                  0.159044
Electronics|WearableTechnology|SmartWatches                                                                        0.051877
Electronics|Mobiles&Accessories|Smartphones&BasicMobiles|Smartphones                                               0.046416
Electronics|HomeTheater,TV&Video|Televisions|SmartTelevisions                                                      0.043003
Electronics|Headphones,Earbuds&Accessories|Headphones|In-Ear                                                       0.035495
                                                                                                                     ...
Electronics|Cameras&Photography|Accessories|Batteries&Chargers|BatteryChargers                                     0.000683
Computers&Accessories|NetworkingDevices|DataCards&Dongles                                                          0.000683
Electronics|HomeAudio|Speakers|MultimediaSpeakerSystems                                                            0.000683
OfficeProducts|OfficePaperProducts|Paper|Copy&PrintingPaper|ColouredPaper                                          0.000683
Home&Kitchen|Kitchen&HomeAppliances|Vacuum,Cleaning&Ironing|Vacuums&FloorCare|VacuumAccessories|VacuumBags|HandheldBags    0.000683
Name: proportion, Length: 211, dtype: float64
```

4. DATA VISUALIZATION

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

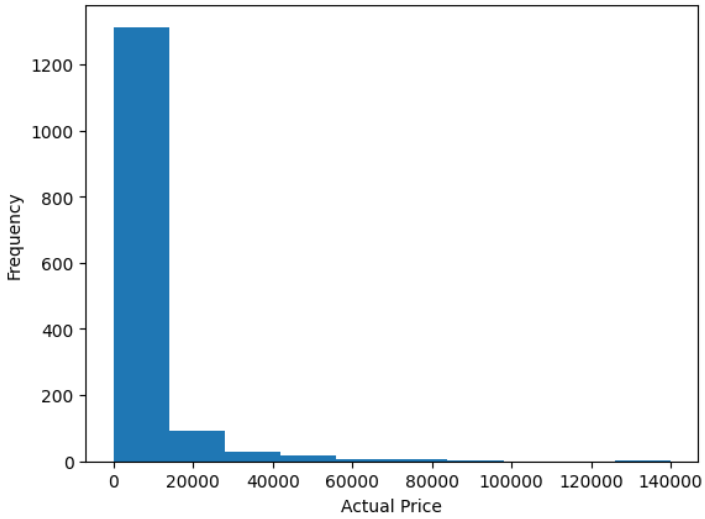
SCATTER PLOT: Actual price vs rating:
SYNTAX:

```
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:

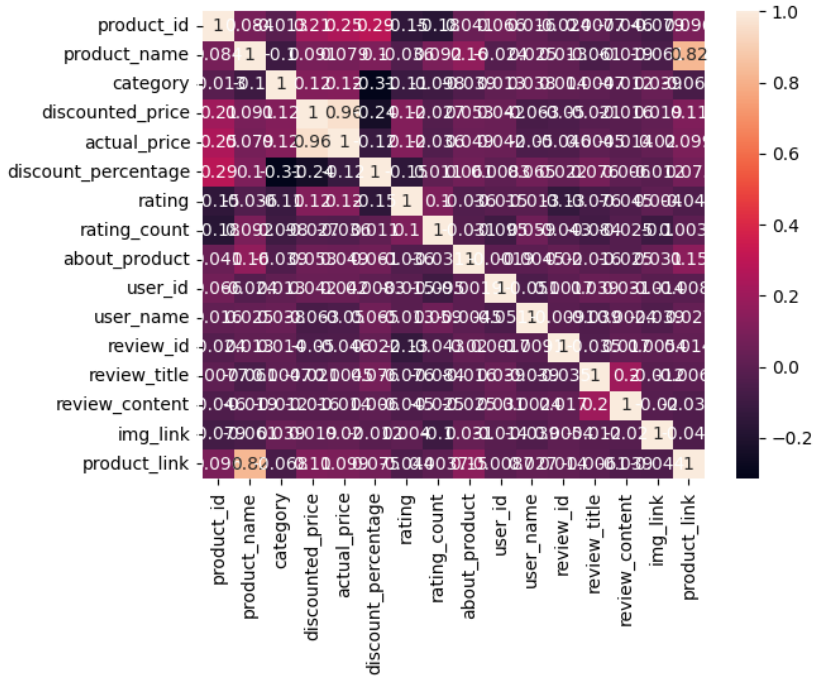
```
plt.hist(Amazon['actual_price'])
plt.xlabel('Actual Price')
plt.ylabel('Frequency')
plt.show()
```



HEATMAP:
SYNTAX:

```
correlation_matrix = Amazon.corr()
```

sns.heatmap(correlation_matrix, annot=True)
plt.show()



5. **CORRELATION ANALYSIS:**

SYNTAX:

 # Calculate Pearson correlation coefficients (default in Pandas)
correlation_matrix = Amazon.corr()

 # Print the correlation matrix
print(correlation_matrix)

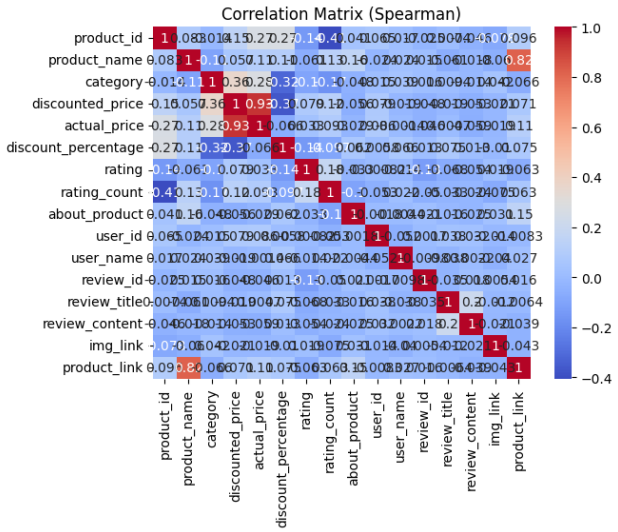
 # Create a heatmap to visualize the correlations
sns.heatmap(correlation_matrix, annot=True, cmap="coolwarm")
plt.title("Correlation Matrix (Pearson)")
plt.show()

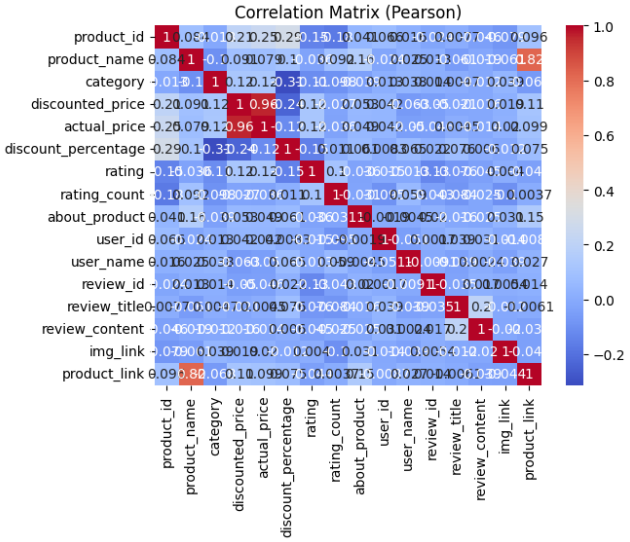
 # Calculate Spearman correlation coefficients (for non-linear relationships)
spearman_correlation_matrix = Amazon.corr(method="spearman")

 # 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()

	product_id	product_name	category	discounted_price
product_id	1.000000	0.084089	-0.012565	0.206448
product_name	0.084089	1.000000	-0.103778	0.090665
category	-0.012565	-0.103778	1.000000	0.119365
discounted_price	0.206448	0.090665	0.119365	1.000000
actual_price	0.246733	0.078567	0.122451	0.961915
discount_percentage	0.289514	0.101913	-0.314465	-0.242412
rating	-0.149105	-0.035592	-0.109424	0.120386





6. 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)

```
category
0      3.800000
1      4.150000
2      3.500000
3      3.600000
4      4.050000
         ...
206    4.250000
207    4.150000
208    4.300000
209    4.133333
210    4.300000
Name: rating, Length: 211, dtype: float64
```

7. **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:

rating	2	2.3	2.6	2.8	2.9	3	3.0	3.1	3.2	3.3	...	4.1	4.2	4.3	4.4	4.5	4.6	4.7	4.8	5.0	|
actual_price																					
₹1,000	0	0	0	0	0	0	0	0	0	0	...	1	1	0	2	0	0	0	0	1	0
₹1,010	0	0	0	0	0	0	0	0	0	0	...	1	0	0	0	0	0	0	0	0	0
₹1,020	0	0	0	0	0	0	0	0	0	0	...	2	0	0	0	0	0	0	0	0	0
₹1,052	0	0	0	0	0	0	0	0	0	0	...	0	0	1	0	0	0	0	0	0	0
₹1,075	0	0	0	0	0	0	0	0	0	0	...	1	0	0	0	0	0	0	0	0	0
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
₹980	0	0	0	0	0	0	0	0	0	0	...	0	1	0	0	0	0	0	0	0	0
₹99	0	0	0	0	0	0	0	0	0	0	...	0	1	2	0	0	0	0	0	0	0
₹990	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	1	0	0	0	0	0
₹995	0	0	0	0	0	0	0	0	0	0	...	0	2	0	0	1	0	0	0	0	0
₹999	0	0	0	0	0	1	1	1	0	5	...	18	13	27	4	3	1	0	0	1	0

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}")

```
Chi-square statistic: 9320.894838747996
p-value: 1.0
Degrees of freedom: 12096
Expected:
 [[0.00682594 0.00682594 0.00682594 ... 0.02047782 0.02047782 0.00682594]
 [0.00068259 0.00068259 0.00068259 ... 0.00204778 0.00204778 0.00068259]
 [0.00136519 0.00136519 0.00136519 ... 0.00409556 0.00409556 0.00136519]
 ...
 [0.00136519 0.00136519 0.00136519 ... 0.00409556 0.00409556 0.00136519]
 [0.00273038 0.00273038 0.00273038 ... 0.00819113 0.00819113 0.00273038]
 [0.08191126 0.08191126 0.08191126 ... 0.24573379 0.24573379 0.08191126]]
```

8. **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)

	category	rating
0	0	3.800000
1	1	4.150000
2	2	3.500000
3	3	3.600000
4	4	4.050000
..	...	...
206	206	4.250000
207	207	4.150000
208	208	4.300000
209	209	4.133333
210	210	4.300000

[211 rows x 2 columns]

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:

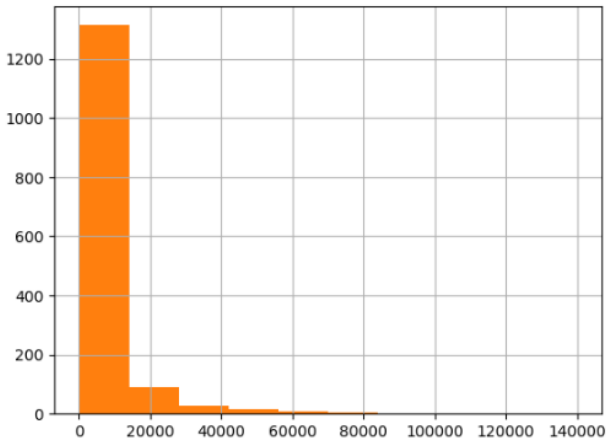
```
import pandas as pd

# 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:

<Axes: >



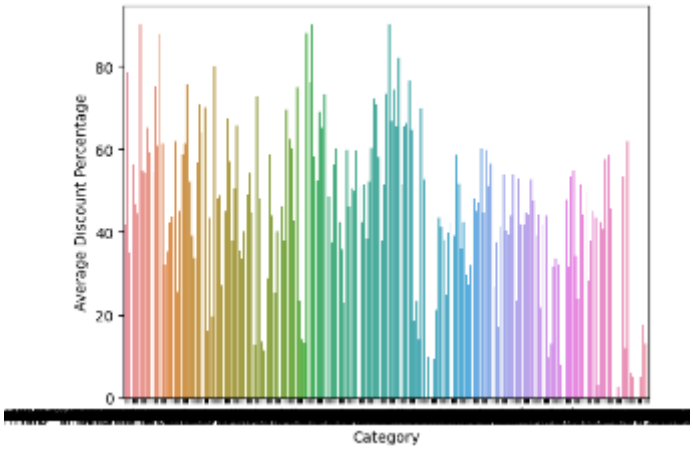
The output shows that discounted prices are generally lower than actual prices, with a median discounted price of $200 and a median actual price of $400.
The discount percentage distribution is skewed to the left, with most products having a discount of 30% or less.
The output suggests that there may 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?

```
# Count occurrences of product names
product_counts = Amazon["product_name"].value_counts()

# Sort in descending order and display top results
print(product_counts.sort_values(ascending=False).head(10))
```

```
product_name
384     5
386     4
121     3
820     3
834     3
1271    3
658     3
1328    3
829     3
1264    3
Name: count, dtype: int64
```

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.