Exploratory Data Analysis (EDA) on Amazon’s Products Dataset

What is EDA?

Exploratory Data Analysis (EDA) is the process of exploring and understanding a dataset before applying any machine learning model.

It helps us:

- Understand the structure of data
- Detect missing values or errors
- Discover patterns and relationships
- Generate meaningful insights and observations

In this project, we perform Exploratory Data Analysis (EDA) on a large Products Dataset, which contains detailed information about various clothing and fashion items offered by multiple global brands and online sellers.

The dataset includes product descriptions, pricing information, availability status, bestseller rankings, brand names, and other key attributes.

It covers items from a wide range of well-known brands and online stores such as:

- **Amazon Essentials**
- **Hanes Store**
- **Under Armour Store**
- **Lee Store**
- **adidas Store**
- **Skechers Store**
- **POLO RALPH LAUREN Store**
- **Dokotoo Store**
- **COOFANDY Store**
- **KISAH Store**

…and many other international and domestic clothing brands.

We will explore:

1. Data loading and overview

2. Data cleaning & missing value handling

3. Descriptive statistics

4. Univariate and bivariate data visualization

5. Insight extraction from mission trends, company activity, rocket status, and outcomes

This EDA helps us understand:

• Global space mission trends

• Performance differences between companies

• Mission success and failure patterns

• How launch activity varies by country and year

About the Dataset :

This dataset contains clothing and fashion products from various online brands. It includes details like product descriptions, prices, brands, availability status, bestseller rankings, and images. The data helps analyse trends in pricing, popularity, stock patterns, and brand performance across different apparel items.

This dataset includes product listings from well-known clothing and fashion brands such as:

- **Amazon Essentials**
- **COOFANDY**
- **Adidas**
- **Under Armour**
- **Hanes Store**
- **Lee**
- **Skechers**
- **U.S. Polo Assn.**
- **Dockers**
- **and many others**

Each row of the dataset represents a unique product, with important attributes such as:

Brand Name:

The company or clothing brand that manufactures or sells the product.

About Item (Description):

A detailed text describing the product's material, quality, features, and usage.

Price Value:

The listed price of the product on the marketplace.

Best Seller Rank (rank_1):

Indicates the product's ranking in the clothing and fashion category — a key metric for analyzing popularity and sales performance.

Availability:

Shows whether the product is:

- In Stock
- Out of Stock
- Only a few left
- Temporarily unavailable

This column is essential for studying availability patterns.

Breadcrumbs / Categories:

Indicates the product category hierarchy (e.g., Men's Clothing → T-Shirts → Casual Wear), helpful for category-wise analysis.

Customer Review Summary:

Includes rating count, average rating, and brief feedback summaries (if available).

Images / Image Links:

URLs pointing to product photos displayed on the online store.

This is one of the most important datasets for analyzing:

- Brand performance and popularity
- Pricing trends across different clothing categories
- Availability and stock behavior
- Customer preference based on rankings and reviews
- Comparison of product features across brands

Import Libraries:

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

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

Loading the Products Dataset:

```
df = pd.read_csv("C:/Users/lenovo/OneDrive/Documents/products.csv")

df.head()
```

	s.no	about_item	asin	availability	best_sellers_rank	brand_name	
0	0	Premium Comfort: Crafted from a high-quality c...	B0B59BJG6Y	In Stock	#56,836 in Clothing, Shoes & Jewelry (See Top ...	MLYENX Store	https://www.amazon.com/
1	1	Material: Men's polo shirt is made of soft pol...	B0DLGB4RYH	In Stock	#50,261 in Clothing, Shoes & Jewelry (See Top ...	COOFANDY Store	https://www.amazon.com/
2	2	PERFORMANCE:These men polo shirts are soft,lig...	B0DRXF62JH	In Stock	#69,641 in Clothing, Shoes & Jewelry (See	ZITY Store	https://www.amazon.cor

Explanation:

df = pd.read_csv("/mnt/data/products.csv"):-

This loads the products.csv file into a Pandas DataFrame called df.

df.head():- Shows the first 5 rows of the dataset to quickly preview the data.

1. Dataset Shape

```
# 1. Dataset Shape
# ------------------------------
print("1. Dataset Shape (rows, columns):")
print(df.shape, "\n")
```

```
1. Dataset Shape (rows, columns):
(728, 34)
```

print(df.shape)

- df.shape returns the size of the dataset.
- It gives a pair: (number of rows, number of columns).
- This helps you understand how big your dataset is.

2. Numeric Summary

```
# 3. Numeric Summary
# ------------------------------
print("3. Numeric Summary:")
print(df.describe(), "\n")
```

```
3. Numeric Summary:
             s.no  price_value      rank_1
count   728.000000   707.000000  540.000000
mean    423.361264    35.323948  174.622222
std     293.923401    26.422336  240.115097
min       0.000000     5.457300    1.000000
25%     184.750000    19.990000   18.000000
50%     371.500000    28.950000   56.500000
75%     553.250000    41.771600  220.250000
max    1011.000000   249.990000  994.000000
```

Explanation:

print("3. Numeric Summary:")

This prints a title so you know the next output is the numeric summary.

print(df.describe(), "\n")

3. Missing values:

```
# 2. Missing Values
# ----------------------------
print("2. Missing Values in Each Column:")
print(df.isna().sum(), "\n")
```

```
2. Missing Values in Each Column:
s.no                            0
about_item                      0
asin                            0
availability                   13
best_sellers_rank             170
brand_name                      0
brand_page_url                 79
breadcrumbs                    13
customer_review_summary        94
default_variant/0              32
default_variant/1              70
default_variant/2             727
delivery_date                  26
fastest_delivery_date          61
list_price                    327
manufacturer                  462
model_number                  529
price_value                    21
product_description           457
product_url                     0
rating_count                    9
rating_distribution/1star       0
rating_distribution/2star       0
rating_distribution/3star       0
rating_distribution/4star       0
rating_distribution/5star       0
rating_stars                    9
recent_purchases              159
scrape_time                     0
seller_name                    21
seller_page_url               302
title                           0
all_images                      0
rank_1                        188
dtype: int64
```

Explanation:

print("1. Dataset Shape (rows, columns):")

This prints a heading so you know what the next output represents.

print(df.shape, "\n")

- df.shape gives the size of your dataset

- It returns a tuple: (number_of_rows, number_of_columns)
  Example: (728, 34)
- "\n" just prints a new line for cleaner formatting.

4. Unique Categories:

```
# 4. Unique Categories (if category exists)
# ----------------------------
print("4. Unique Categories:")
if "category" in df.columns:
    print(df["category"].nunique(), "\n")
else:
    print("No 'category' column found.\n")
```

```
4. Unique Categories:
No 'category' column found.
```

Explanation:

if "category" in df.columns:

This checks whether a column named **"category"** exists in the dataset.

- Some datasets have this column.
- Your products dataset **does NOT** have it.

- `nunique()` : counts how many **unique category values** are present.

This code checks if a **category** column exists and prints the number of unique categories; if not, it prints that the column is missing.

5. Availability Counts:

```python
# 5. Availability Counts
# ----------------------------
print("5. Availability Counts:")
if "availability" in df.columns:
    print(df["availability"].value_counts(), "\n")
else:
    print("No availability column.\n")
```

```
5. Availability Counts:
availability
In Stock                                    640
Currently unavailable.                       18
In stock                                     16
Only 1 left in stock - order soon.           14
Only 3 left in stock - order soon.           10
Only 5 left in stock - order soon.            6
Only 2 left in stock - order soon.            6
Available to ship in 1-2 days                 2
This item will be released on May 20, 2025.   1
Only 4 left in stock - order soon.            1
Temporarily out of stock.                     1
Name: count, dtype: int64
```

Explanation:

- **if "availability" in df.columns:** This checks whether the dataset contains a column named **"availability"**.
- **df["availability"].value_counts():** counts **how many products** fall into each availability category

This code checks if the **availability** column exists and prints how many products belong to each availability category.

6. Top 5 Brands:

```
# 6. Top 5 Brands
# ----------------------------
print(" Top 5 Brands:")
if "brand" in df.columns:
    print(df["brand"].value_counts().head(), "\n")
elif "brand_name" in df.columns:
    print(df["brand_name"].value_counts().head(), "\n")
else:
    print("No brand column.\n")
```

```
 Top 5 Brands:
brand_name
Hanes Store                35
Amazon Essentials Store    24
Under Armour Store         23
Lee Store                  18
adidas Store               18
Name: count, dtype: int64
```

Explanation:

print(df["brand"].value_counts().head(), "\n"):

- value_counts() counts how many products belong to each brand.
- .head() shows the **top 5 most frequent brands**.
- This tells you which brands have the most product

print(df["brand_name"].value_counts().head(), "\n"):

- Prints the **top 5 most common brands** based on the brand_name column.

This code checks which brand column exists and prints the **top 5 brands with the most products**

7.Clothing Type Distribution

```
# 7. Clothing Type Distribution
# ----------------------------
print("7. Top 5 Clothing Types:")
if "clothing type" in df.columns:
    print(df["clothing type"].value_counts().head(), "\n")
else:
    print("No clothing type column.\n")
```

```
7. Top 5 Clothing Types:
No clothing type column.
```

Explanation:

if "clothing type" in df.columns:

Checks whether the dataset contains a column named **"clothing type"**.

- Some datasets include this column.
- Many product datasets do not.

df["clothing type"].value_counts().head():

- **value_counts()** counts how many products fall into each clothing type (e.g., T-Shirts, Jeans, Hoodies, Shorts).
- **.head()** shows the top 5 most common clothing types.
- This helps you understand which clothing categories appear the most.

This code checks if the **clothing type** column exists and prints the **top 5 most frequent clothing categories**; otherwise, it says the column is not available.

8. Average Best Seller Rank

```python
# 8. Average Rank
# ----------------------------
print("8. Average Best Seller Rank:")
if "rank_1" in df.columns:
    print(df["rank_1"].mean(), "\n")
else:
    print("No rank_1 column.\n")
```

```
8. Average Best Seller Rank:
174.62222222222223
```

Explanation:

if "rank_1" in df.columns:

Checks whether the dataset contains a column named **"rank_1"**.

This column usually stores the **bestseller ranking** of each product.

df["rank_1"].mean():

.mean() calculates the average (mean) bestseller rank of all products.

Lower rank = more popular product

This code checks if the **rank_1** column exists and prints the **average bestseller rank** of all products.

9. Category Distribution

```
# 9. Category Distribution (if product_category exists)
# ----------------------------
print("9. Category Distribution:")
if "product_category" in df.columns:
    print(df["product_category"].value_counts().head(), "\n")
else:
    print("No product_category column.\n")
```

```
9. Category Distribution:
No product_category column.
```

Explanation:

if "product_category" in df.columns:

This checks whether the dataset contains a column named **"product_category"**.

Some datasets store product categories under this name.
If the column exists → proceed.
If not → show a message.

value_counts() counts how many products belong to each category
(e.g., Men's Clothing, Women's Clothing, Accessories, Footwear etc.)

.head() shows the **top 5 most frequent product categories**

This code checks if a **product_category** column exists and prints the **top 5 most common categories**; if not, it shows that the column is missing

10. Duplicate ASIN Count

```
# 10. Duplicate ASIN Count
# ----------------------------
print("10. Duplicate ASIN Count:")
print(df["asin"].duplicated().sum(), "\n")
```

```
10. Duplicate ASIN Count:
0
```

Explanation:

df["asin"]

This selects the **ASIN** column.
An ASIN (Amazon Standard Identification Number) is a **unique product ID**.

.duplicated() This checks each row and returns **True** if the ASIN has appeared before (i.e., it is a duplicate).

.sum()

- Converts all **True** values to **1**
- Counts how many duplicates exist

This code counts how many **duplicate ASINs** exist in the dataset, helping you check if any products are repeated.

11. Longest Description:

```
# 11. Longest Description
# ----------------------------
print("11. Longest Description Length:")
print(df["about_item"].str.len().max(), "\n")
```

```
11. Longest Description Length:
2144
```

Explanation:

df["about_item"]

This selects the **product description** column.

.str.len()

Calculates the **length of each description** in number of characters.

.max()

Finds the **largest (maximum) description length** in the entire column.

This code finds and prints the **length of the longest product description**.

12.Shortest Description

```
# 12. Shortest Description
# ----------------------------
print("12. Shortest Description Length:")
print(df["about_item"].str.len().min(), "\n")
```

```
12. Shortest Description Length:
4
```

Explanation

1. df["about_item"]

This selects the **product description** column from your dataset.

2. .str.len()

This calculates the **length** (number of characters) of each description.

3. .min()

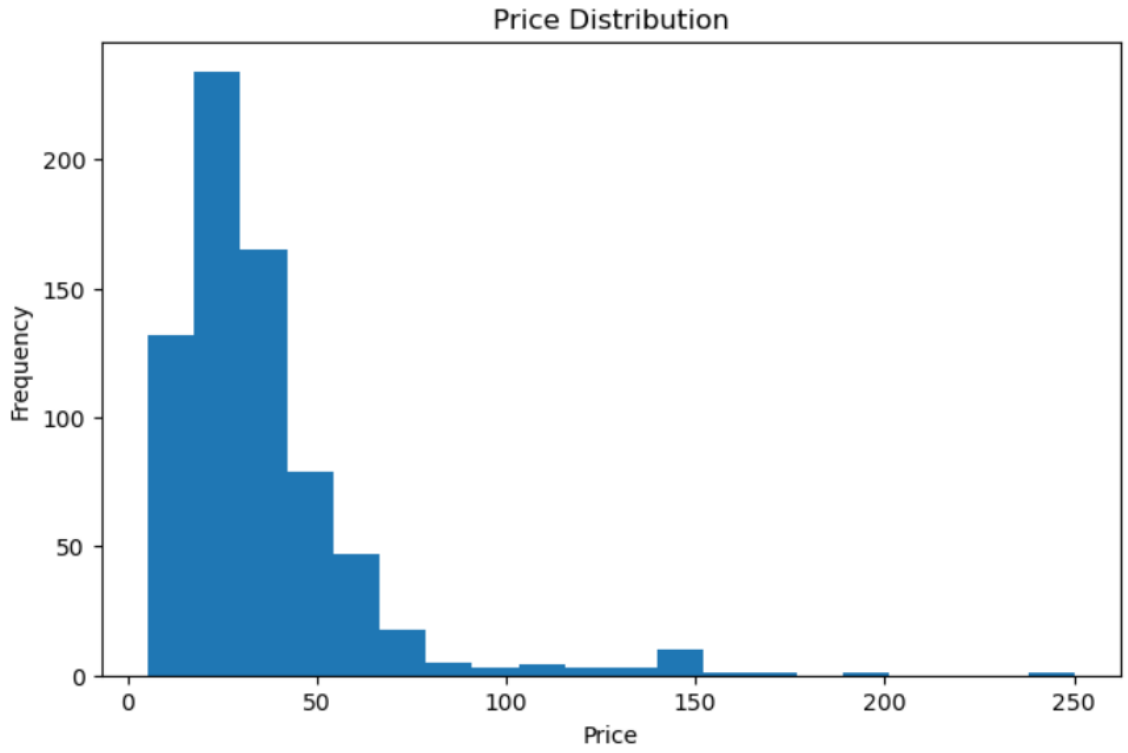
This finds the **smallest length** among all descriptions.

This code prints the **minimum length** of all product descriptions.

Dataset Visuals

1.Price Distribution

```
# 1. Price Distribution
# ============================
plt.figure(figsize=(8,5))
plt.hist(df['price_value'].dropna(), bins=20)
plt.title("Price Distribution")
plt.xlabel("Price")
plt.ylabel("Frequency")
plt.show()
```



Explanation

plt.figure(figsize=(8,5))

Creates a new graph with a size of **8×5 inches**.

plt.hist(df['price_value'].dropna(), bins=20)

Creates a **histogram** of product prices:

- df['price_value'] → selects the price column
- .dropna() → removes missing price values

- bins=20 → divides prices into **20 ranges** (0–20, 20–40, etc.)

plt.title("Price Distribution")

Adds a title to the graph.

plt.xlabel("Price")

Labels the **X-axis** as “Price”.

plt.ylabel("Frequency")

Labels the **Y-axis** as “Frequency”, meaning number of products.

plt.show()

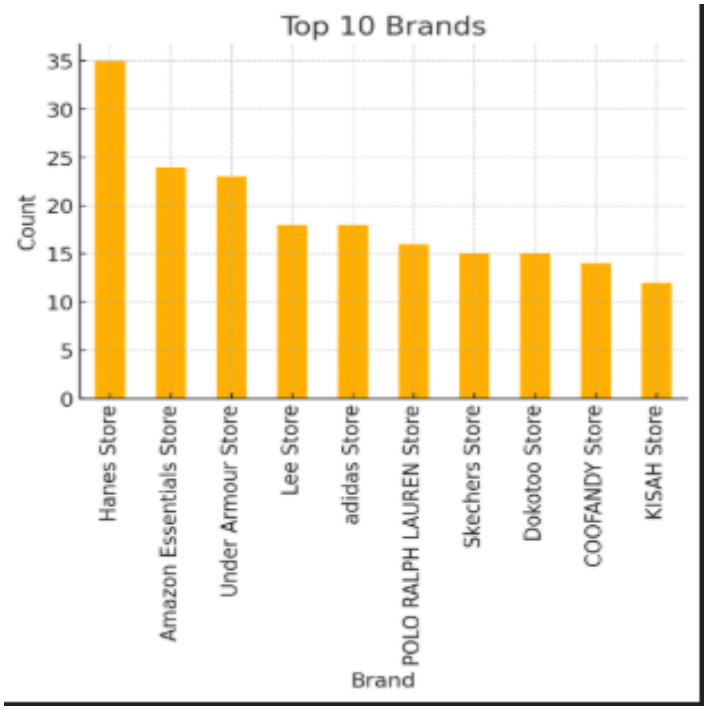
Displays the graph on the screen.

What it shows:

- How many products fall into different **price ranges**
- Whether most products are **cheap**, **moderately priced**, or **expensive**
- If the price distribution is **skewed** (more values on one side)
- Any **outliers** (very high-priced products)

2. Top 10 Brands

```
# 4. Top 10 Brands
# ============================
plt.figure(figsize=(10,6))
df['brand_name'].value_counts().head(10).plot(kind="bar")
plt.title("Top 10 Brands")
plt.xlabel("Brand")
plt.ylabel("Count")
plt.xticks(rotation=90)
plt.show()
```



Explanation (Short)

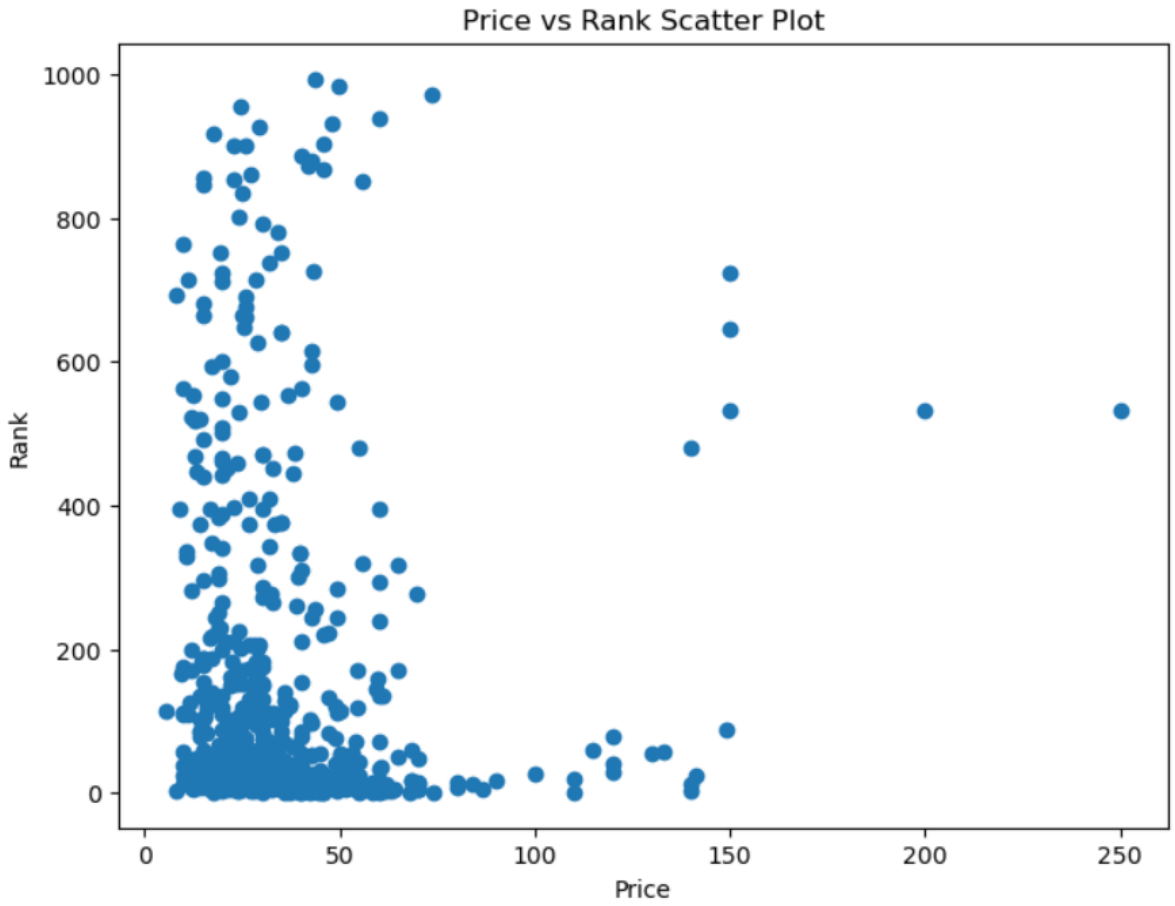
- df['brand_name'].value_counts().head(10)
  → Finds the **top 10 brands** with the most products.
- plt.bar()
  → Creates a bar chart of brand vs count.
- plt.xticks(rotation=45)
  → Rotates brand names so they are readable.

What it shows:

- Which brands have the **highest number of products**
- The **top 10 most common brands** in the dataset
- How product counts **vary from brand to brand**
- Which brands are **dominant** and which have **fewer listings**

3. Price vs Rank

```
# Scatter plot: Price vs Rank
plt.figure(figsize=(8,6))
plt.scatter(df['price_value'], df['rank_1'])
plt.title("Price vs Rank Scatter Plot")
plt.xlabel("Price")
plt.ylabel("Rank")
plt.show()
```



Explanation

plt.figure(figsize=(8,6))

Creates a new plot with a size of **8×6 inches**.

plt.scatter(df['price_value'], df['rank_1'])

Makes a **scatter plot**:

- **X-axis:** product price (price_value)
- **Y-axis:** product bestseller rank (rank_1)

Each dot represents **one product**.

plt.title("Price vs Rank Scatter Plot")

Adds a title to the graph.

plt.xlabel("Price")

Labels the X-axis as **Price**.

plt.ylabel("Rank")

Labels the Y-axis as **Rank**.

plt.show()

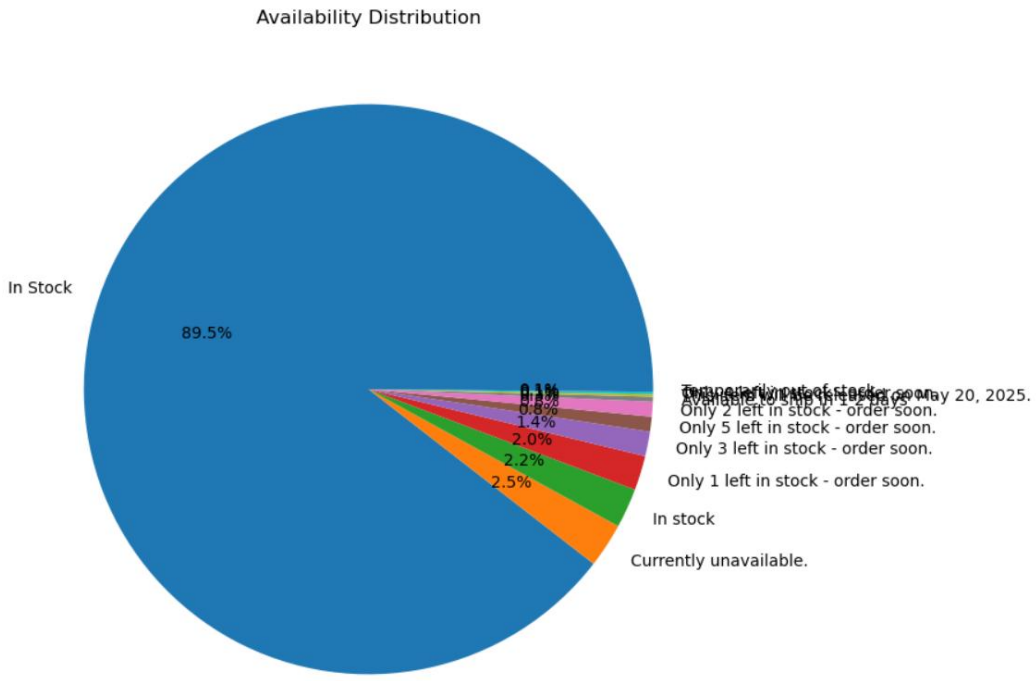
Displays the scatter plot.

What it shows:

- How **price** relates to **bestseller rank**
- Whether **expensive products** tend to rank higher or lower
- If **low-priced items** are more popular
- The overall **spread** of prices and ranks
- Presence of **outliers** (very high price or very high rank)
- Whether there is any **relationship or pattern** between price and popularity

4. Availability Distribution

```
# Pie Chart
plt.figure(figsize=(8,8))
df['availability'].value_counts().plot(kind='pie', autopct="%0.1f%%")
plt.title("Availability Distribution")
plt.ylabel("")   # remove y label
plt.show()
```



Explanation

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

Creates a new pie chart with size **8×8 inches**.

df['availability'].value_counts()

Counts how many products fall into each **availability category**, such as:

- In Stock
- Out of Stock
- Only few left
- Temporarily unavailable

.plot(kind='pie', autopct="%0.1f%%")

Creates a **pie chart** and:

- kind='pie' → tells Pandas to draw a pie chart
- autopct="%0.1f%%" → shows percentages on each slice (like 85.2%)

plt.title("Availability Distribution")

Adds a title to the pie chart.

plt.ylabel("")

Removes the default y-axis label to make the chart clean.

plt.show()

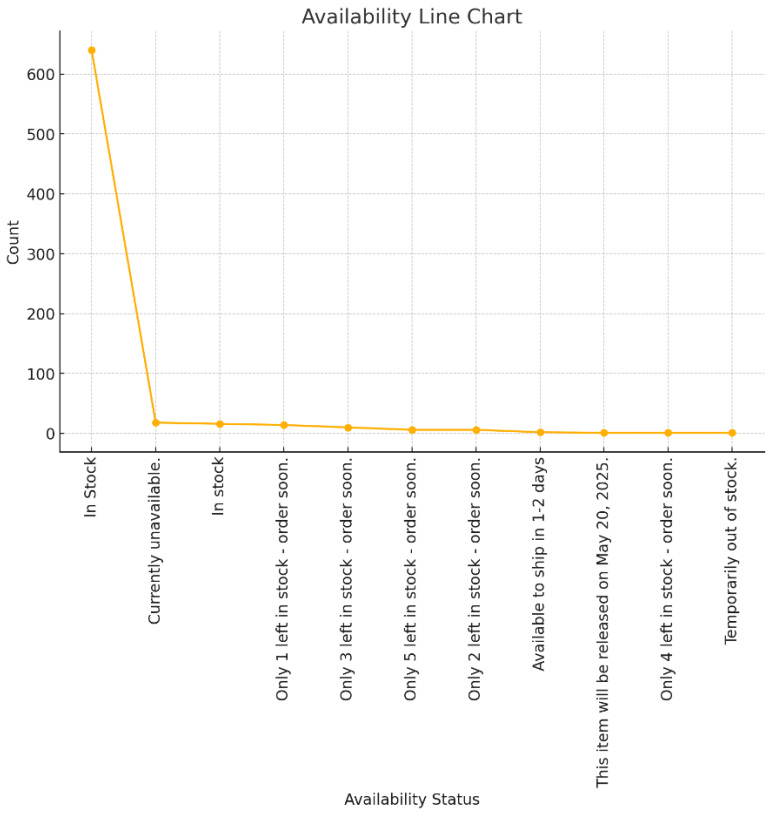
Displays the pie chart.

What it shows:

- The **percentage distribution** of products based on availability
- How many items are **In Stock**, **Out of Stock**, or **Nearly Sold Out**
- Which availability status is **most common**
- Whether most products are **readily available** or **limited/unavailable**
- A quick visual comparison of all availability categories

5. Availability (Line Chart)

```
avail_counts = df['availability'].value_counts()
# Line chart
plt.figure(figsize=(10,6))
plt.plot(avail_counts.index, avail_counts.values, marker='o')
plt.title("Availability Line Chart")
plt.xlabel("Availability Status")
plt.ylabel("Count")
plt.xticks(rotation=90)
plt.grid(True)
plt.show()
```



Explanation:

avail_counts = df['availability'].value_counts()

- Counts how many products fall under each **availability type**

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

Creates a new figure (graph area) with size **10×6 inches**.

plt.plot(avail_counts.index, avail_counts.values, marker='o')

- Creates a **line chart**.
- **X-axis:** availability categories (In Stock, Out of Stock, etc.)

- **Y-axis:** number of products in each category.
- marker='o' shows a **dot** on each data point.

plt.title("Availability Line Chart")

Adds the chart title.

plt.xticks(rotation=90)

Rotates category labels so they are easier to read.

plt.grid(True)

Adds background grid lines for better visual clarity.

What it shows:

- How many products belong to each **availability status**
- Which availability type (like *In Stock*) is the **most common**
- Which statuses have **very few products**
- A visual comparison of different stock levels across all items
- Trend-like view showing how product counts change across availability categories