**

**Project Report**

Bachelor Of Computer Applications

2nd Semester

Exploratory Data Analysis Project

**Global Product Inventory Dataset 2025 Analysis**

By

DIPIN ROKA

2411021240007

Githublink: https://github.com/DIPINROKA10/IDS\_PROJECT

**Department Of Computer Application**

**Alliance University**

**Chandrapur a- Anekal Main Road,Anekal**

**Bengaluru** –1 **56210**

Introduction

The Global Product Inventory Dataset 2025 provides comprehensive insights into the global distribution, pricing, sales, and demand of various products across different regions. With the rapid growth of e-commerce and international trade, understanding inventory dynamics has become crucial for businesses to maintain efficiency, meet customer demand, and optimize profitability.

This report aims to analyze the dataset using linear regression techniques to uncover patterns and relationships between key variables such as product quantity, pricing, demand index, and sales performance. By applying data analysis and machine learning methods, the goal is to predict future sales trends and support strategic decision-making for inventory management in a competitive global market.

1. Data Preprocessing:

o Cleaning the dataset by handling missing values and encoding categorical variables (e.g., converting regions into numeric format using label encoding).

o Selected relevant features for analysis: Quantity, Price, Demand Index, and Region.

2. Splitting the Dataset:

o Divided the data into training (80%) and testing (20%) sets using the train\_test\_split method.

3. Model Training: o Applied the **Linear Regression** model from the Scikit-learn library.

o Trained the model using the training dataset to learn the relationship between the selected features and the target variable (Sales).

4. Prediction and Evaluation: o Used the trained model to predict sales on the testing set.

o Evaluated the model performance using **Mean Squared Error (MSE)**, which measures the average squared difference between actual and predicted values.

o Visualized the relationship between actual and predicted sales using scatter plots for better interpretation.

*# Global Product Inventory Dataset 2025*

import pandas as pd

df=pd.read\_csv(r"D:\products.csv") print(df)

Product ID Product Name Product Category Product Description

Price \

1. 93TGNAY7 Laptop Home Appliances Product\_XU5QX

253.17

1. TYYZ5AV7 Smartphone Clothing Product\_NRUMS

214.37

1. 5C94FGTQ Headphones Clothing Product\_IT7HG

475.29

1. XBHKYPQB Monitor Clothing Product\_8SBDO

403.33

1. 728GCZFU Laptop Home Appliances Product\_54FAF

229.81 ... ... ... ... ...

...

1. J29B6RDI Headphones Clothing Product\_NI8C7

21.48

1. L1HL7437 Laptop Clothing Product\_8RR6T

403.92

1. FD57S4E1 Laptop Home Appliances Product\_GYAWW

484.46

1. RPYLOB1M Headphones Clothing Product\_K3M9M

411.63

1. 3JWTGTOM Laptop Clothing Product\_I0ACF

74.38

Stock Quantity Warranty Period Product Dimensions Manufacturing Date \

1. 3 2 16x15x15 cm 2023-

01-01

1. 92 2 15x19x19 cm 2023-

03-15

1. 19 2 9x6x6 cm 2023-

03-15

1. 40 1 7x13x5 cm 2023-

01-01

1. 32 2 20x20x19 cm 2023-

07-30

... ... ... ...

...

1. 91 3 10x16x9 cm 2023-

03-15

1. 19 2 13x8x5 cm 2023-

01-01

1. 13 2 5x15x15 cm 2023-

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 01-01   1. 79 1 17x11x17 cm 2023-   07-30   1. 81 1 6x6x16 cm 2023-   01-01   |  |  |  | | --- | --- | --- | | Expiration Date SKU Product Tags Color/Size Variations \ | | | | 1. 2026-01-01 8NMFZ4 VNU,NZ6 Green/Large 2. 2025-01-01 7P5YCW ZJA,0D3 Red/Small 3. 2026-01-01 YW5BME ZNG,MAP Red/Small 4. 2026-01-01 65MQC3 RPP,M40 Green/Large 5. 2026-01-01 RLCBRW R8U,X46 Blue/Medium... ... ... ... ... 6. 2026-01-01 0IQPXX M81,8WN Blue/Medium 7. 2024-01-01 HW1HV1 0UM,L4B Red/Small 8. 2024-01-01 MKJ0UW GO4,EZE Red/Small 9. 2026-01-01 INSC1B 0QB,U55 Red/Small 10. 2025-01-01 UH0U3R C5R,TZN Blue/Medium | |  | | Product Ratings   1. 2 2. 2 3. 1 4. 1 5. 4 ... ... 6. 1 7. 4 8. 1 9. 1 10. 1 |   [10000 rows x 14 columns] |

shows first five rows

|  |  |  |  |
| --- | --- | --- | --- |
| *#using Df opeerations*  df.head() *#shows first five elements*     |  | | --- | | Product ID Product Name Product Category Product Description Price | |  | | 93TGNAY7 Laptop Home Appliances Product\_XU5QX 253.17 |   \  0   1. TYYZ5AV7 Smartphone Clothing Product\_NRUMS 214.37 2. 5C94FGTQ Headphones Clothing Product\_IT7HG 475.29 3. XBHKYPQB Monitor Clothing Product\_8SBDO 403.33 4. 728GCZFU Laptop Home Appliances Product\_54FAF 229.81 |

Stock Quantity Warranty Period Product Dimensions Manufacturing

Date \

1. 3 2 16x15x15 cm 2023-01-

01

1. 92 2 15x19x19 cm 2023-03-

15

1. 19 2 9x6x6 cm 2023-03-

15

1. 40 1 7x13x5 cm 2023-01-

01

1. 32 2 20x20x19 cm 2023-07-

30

Expiration Date SKU Product Tags Color/Size Variations Product

Ratings

1. 2026-01-01 8NMFZ4 VNU,NZ6 Green/Large

2

1. 2025-01-01 7P5YCW ZJA,0D3 Red/Small

2

1. 2026-01-01 YW5BME ZNG,MAP Red/Small

1

1. 2026-01-01 65MQC3 RPP,M40 Green/Large

1

1. 2026-01-01 RLCBRW R8U,X46 Blue/Medium

4

df.tail() *#shows last 5 elements in the dataframe*

Product ID Product Name Product Category Product Description

Price \

1. J29B6RDI Headphones Clothing Product\_NI8C7

21.48

1. L1HL7437 Laptop Clothing Product\_8RR6T

403.92

1. FD57S4E1 Laptop Home Appliances Product\_GYAWW

484.46

1. RPYLOB1M Headphones Clothing Product\_K3M9M

411.63

1. 3JWTGTOM Laptop Clothing Product\_I0ACF

74.38

Stock Quantity Warranty Period Product Dimensions Manufacturing Date \

1. 91 3 10x16x9 cm 2023-

03-15

1. 19 2 13x8x5 cm 202301-01
2. 13 2 5x15x15 cm 2023-

01-01

1. 79 1 17x11x17 cm 2023-

07-30

1. 81 1 6x6x16 cm 2023-

01-01

Expiration Date SKU Product Tags Color/Size Variations \

1. 2026-01-01 0IQPXX M81,8WN Blue/Medium
2. 2024-01-01 HW1HV1 0UM,L4B Red/Small
3. 2024-01-01 MKJ0UW GO4,EZE Red/Small
4. 2026-01-01 INSC1B 0QB,U55 Red/Small
5. 2025-01-01 UH0U3R C5R,TZN Blue/Medium

Product Ratings

1. 1
2. 4
3. 1
4. 1
5. 1

conver ting the date to its correct format

df["Manufacturing Date"] = pd.to\_datetime(df["Manufacturing Date"])

#shows the stastical configration of the dataframe

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| df.describe()   |  |  |  | | --- | --- | --- | | Price Stock Quantity Warranty Period \ | | | | count 10000.000000 10000.000000 10000.000000 mean 254.665715 50.647100 2.014000 min 10.220000 1.000000 1.000000 25% 129.985000 25.000000 1.000000  50% 253.425000 51.000000 2.000000 75% 379.970000 76.000000 3.000000 max 499.970000 100.000000 3.000000 std 142.755688 28.901977 0.817968 | |  | | Manufacturing Date Product Ratings count 10000 10000.000000 mean 2023-04-03 12:08:55.680000 3.004700 min 2023-01-01 00:00:00 1.000000 25% 2023-01-01 00:00:00 2.000000  50% 2023-03-15 00:00:00 3.000000  75% 2023-07-30 00:00:00 4.000000 | |

max 2023-07-30 00:00:00 5.000000 std NaN 1.419676

#used to describe the data types

df.info()

<class 'pandas.core.frame.DataFrame'>

RangeIndex: 10000 entries, 0 to 9999 Data columns (total 14 columns):

# Column Non-Null Count Dtype

--- ------ -------------- -----

1. Product ID 10000 non-null object
2. Product Name 10000 non-null object
3. Product Category 10000 non-null object
4. Product Description 10000 non-null object
5. Price 10000 non-null float64
6. Stock Quantity 10000 non-null int64
7. Warranty Period 10000 non-null int64
8. Product Dimensions 10000 non-null object
9. Manufacturing Date 10000 non-null datetime64[ns]
10. Expiration Date 10000 non-null object
11. SKU 10000 non-null object
12. Product Tags 10000 non-null object
13. Color/Size Variations 10000 non-null object 13 Product Ratings 10000 non-null int64 dtypes: datetime64[ns](1), float64(1), int64(3), object(9) memory usage: 1.1+ MB

#checking for the misssing values

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| print(df.isnull().sum())   |  |  | | --- | --- | | Product ID 0  Product Name 0  Product Category 0  Product Description 0  Price 0  Stock Quantity 0  Warranty Period 0  Product Dimensions 0  Manufacturing Date 0  Expiration Date 0  SKU 0  Product Tags 0  Color/Size Variations 0  Product Ratings 0 | | | dtype: int64 |  | |

checking for the duplicate values

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| df.duplicated()   |  |  |  | | --- | --- | --- | | 1. False 2. False 3. False 4. False 5. False | |  | | .. | . | | 1. False 2. False 3. False 4. False 5. False | | | Length: 10000, dtype: bool | | | |

removing the duplicate value in the data frame.

print(df.duplicated().sum()) *# Count duplicates* df.drop\_duplicates(inplace=True) *# Remove duplicates*

0

removes spaces and also remove any unwanted characters

df.columns = df.columns.str.strip()

Check column data types print(df.dtypes)

Product ID object

Product Name object

Product Category object

Product Description object

Price float64

Stock Quantity int64

Warranty Period int64

Product Dimensions object

Manufacturing Date datetime64[ns]

Expiration Date object

SKU object

Product Tags object

Color/Size Variations object

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| |  |  | | --- | --- | | Product Ratings int64 | | | dtype: object |  | |

Strip spaces and remove any unwanted characters

data=pd.read\_csv(r"D:\products.csv")

data.columns = data.columns.str.strip() print(data.columns)

Index(['Product ID', 'Product Name', 'Product Category', 'Product

Description',

'Price', 'Stock Quantity', 'Warranty Period', 'Product

Dimensions',

'Manufacturing Date', 'Expiration Date', 'SKU', 'Product Tags',

'Color/Size Variations', 'Product Ratings'], dtype='object')

showst the overall colums of the data frame

df.columns = df.columns.str.strip()

this code helps to convert the price row int tineger by usinf astype function

df["Price"] = df["Price"].astype(int) df

Product ID Product Name Product Category Product Description

Price \

1. 93TGNAY7 Laptop Home Appliances Product\_XU5QX

253

1. TYYZ5AV7 Smartphone Clothing Product\_NRUMS

214

1. 5C94FGTQ Headphones Clothing Product\_IT7HG

475

1. XBHKYPQB Monitor Clothing Product\_8SBDO

403

1. 728GCZFU Laptop Home Appliances Product\_54FAF

229 ... ... ... ... ... . ..

1. J29B6RDI Headphones Clothing Product\_NI8C7

21

1. L1HL7437 Laptop Clothing Product\_8RR6T

403

1. FD57S4E1 Laptop Home Appliances Product\_GYAWW

484

1. RPYLOB1M Headphones Clothing Product\_K3M9M

411

1. 3JWTGTOM Laptop Clothing Product\_I0ACF

74

Stock Quantity Warranty Period Product Dimensions Manufacturing Date \

1. 3 2 16x15x15 cm 2023-

01-01

1. 92 2 15x19x19 cm 2023-

03-15

1. 19 2 9x6x6 cm 2023-

03-15

1. 40 1 7x13x5 cm 2023-

01-01

1. 32 2 20x20x19 cm 2023-

07-30

... ... ... ...

...

1. 91 3 10x16x9 cm 2023-

03-15

1. 19 2 13x8x5 cm 2023-

01-01

1. 13 2 5x15x15 cm 2023-

01-01

1. 79 1 17x11x17 cm 2023-

07-30

1. 81 1 6x6x16 cm 2023-

01-01

Expiration Date SKU Product Tags Color/Size Variations \

1. 2026-01-01 8NMFZ4 VNU,NZ6 Green/Large
2. 2025-01-01 7P5YCW ZJA,0D3 Red/Small
3. 2026-01-01 YW5BME ZNG,MAP Red/Small
4. 2026-01-01 65MQC3 RPP,M40 Green/Large
5. 2026-01-01 RLCBRW R8U,X46 Blue/Medium ... ... ... ... ...
6. 2026-01-01 0IQPXX M81,8WN Blue/Medium
7. 2024-01-01 HW1HV1 0UM,L4B Red/Small
8. 2024-01-01 MKJ0UW GO4,EZE Red/Small
9. 2026-01-01 INSC1B 0QB,U55 Red/Small
10. 2025-01-01 UH0U3R C5R,TZN Blue/Medium

Product Ratings

1. 2
2. 2
3. 1
4. 1
5. 4

... ... 9995 1

1. 4
2. 1
3. 1
4. 1

[10000 rows x 14 columns]

df.columns = df.columns.str.strip() df

Product ID Product Name Product Category Product Description

Price \

1. 93TGNAY7 Laptop Home Appliances Product\_XU5QX

253

1. TYYZ5AV7 Smartphone Clothing Product\_NRUMS

214

1. 5C94FGTQ Headphones Clothing Product\_IT7HG

475

1. XBHKYPQB Monitor Clothing Product\_8SBDO

403

1. 728GCZFU Laptop Home Appliances Product\_54FAF

229 ... ... ... ... ... . ..

1. J29B6RDI Headphones Clothing Product\_NI8C7

21

1. L1HL7437 Laptop Clothing Product\_8RR6T

403

1. FD57S4E1 Laptop Home Appliances Product\_GYAWW

484

1. RPYLOB1M Headphones Clothing Product\_K3M9M

411

1. 3JWTGTOM Laptop Clothing Product\_I0ACF

74

Stock Quantity Warranty Period Product Dimensions Manufacturing Date \

1. 3 2 16x15x15 cm 2023-

01-01

1. 92 2 15x19x19 cm 2023-

03-15

1. 19 2 9x6x6 cm 2023-

03-15

1. 40 1 7x13x5 cm 2023-

01-01

1. 32 2 20x20x19 cm 2023-

07-30

... ... ... ... ...

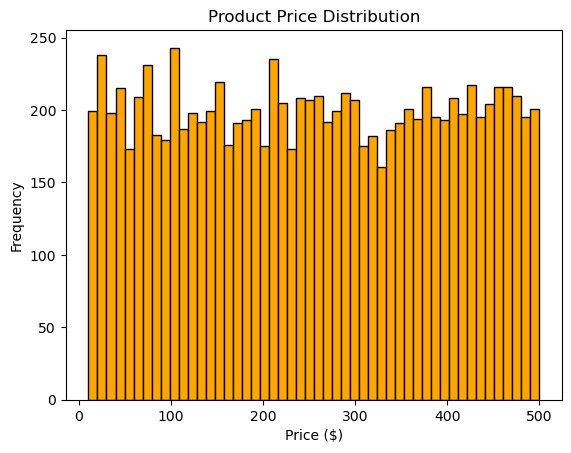
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| |  |  |  |  |  | | --- | --- | --- | --- | --- | | 9995 91 3 10x16x9 cm 2023- | | | | | | 03-15 |  | | | | | 9996 19 2 13x8x5 cm 2023- | | | | | | 01-01 |  | | | | | 9997 13 2 5x15x15 cm 2023- | | | | | | 01-01 |  | | | | | 9998 79 1 17x11x17 cm 2023- | | | | | | 07-30 |  | | | | | 9999 81 1 6x6x16 cm 2023- | | | | | | 01-01 |  | | | | | Expiration Date SKU Product Tags Color/Size Variations \ | | | | | 1. 2026-01-01 8NMFZ4 VNU,NZ6 Green/Large 2. 2025-01-01 7P5YCW ZJA,0D3 Red/Small 3. 2026-01-01 YW5BME ZNG,MAP Red/Small 4. 2026-01-01 65MQC3 RPP,M40 Green/Large 5. 2026-01-01 RLCBRW R8U,X46 Blue/Medium... ... ... ... ... 6. 2026-01-01 0IQPXX M81,8WN Blue/Medium 7. 2024-01-01 HW1HV1 0UM,L4B Red/Small 8. 2024-01-01 MKJ0UW GO4,EZE Red/Small 9. 2026-01-01 INSC1B 0QB,U55 Red/Small 10. 2025-01-01 UH0U3R C5R,TZN Blue/Medium | | |  | | Product Ratings   1. 2 2. 2 3. 1 4. 1 5. 4 ... ... 6. 1 7. 4 8. 1 9. 1 10. 1 | |   [10000 rows x 14 columns] |

the above code shows the distribution among products using histogram using hist

import matplotlib.pyplot as plt df=pd.read\_csv(r"D:\products.csv")

plt.hist(df['Price'], bins=50, color='orange', edgecolor='black') plt.xlabel("Price ($)") plt.ylabel("Frequency")

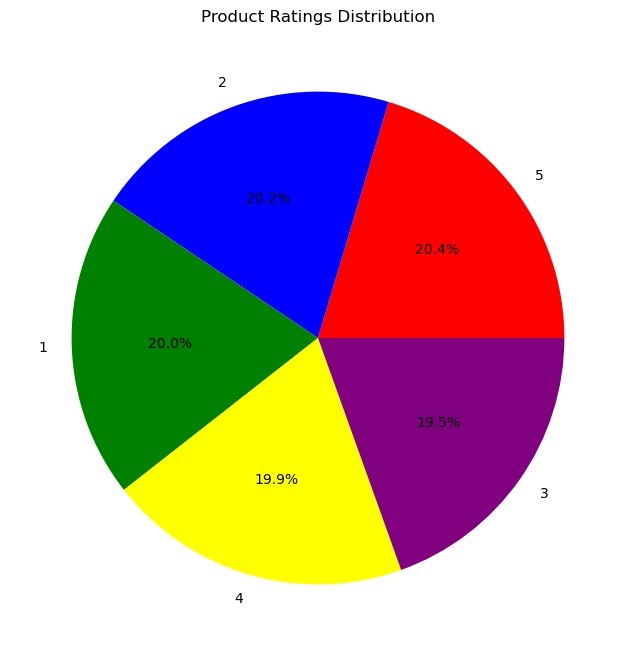
plt.title("Product Price Distribution") plt.show()



This code counts the number of products for each rating using value\_counts(). It then creates a pie chart to visualize the distribution of product ratings.

ratings\_count = df['Product Ratings'].value\_counts() plt.figure(figsize=(8, 8))

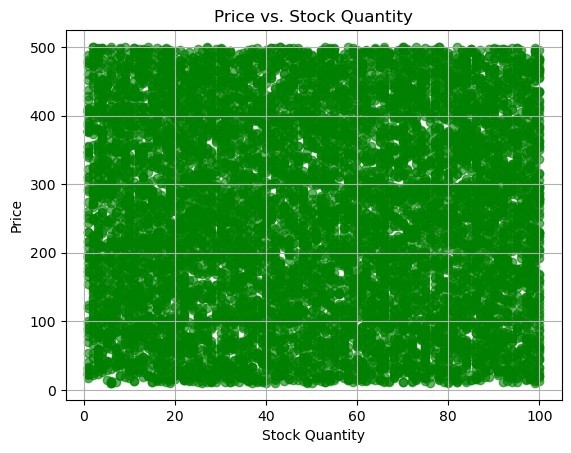
plt.pie(ratings\_count, labels=ratings\_count.index, autopct='%1.1f%%', colors=['red', 'blue', 'green', 'yellow', 'purple']) plt.title("Product Ratings Distribution") plt.show()



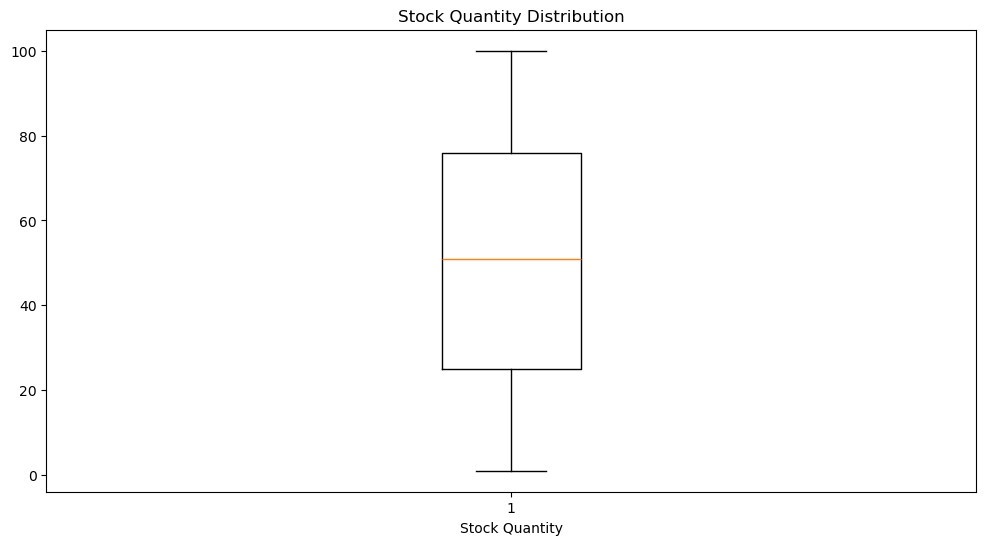
This code creates a scatter plot showing the relationship between Stock Quantity and Price. Each dot represents a product

|  |
| --- |
| import matplotlib.pyplot as plt  plt.scatter(df["Stock Quantity"], df["Price"], color="green", alpha=0.6)  plt.xlabel("Stock Quantity") plt.ylabel("Price")  plt.title("Price vs. Stock Quantity") |

plt.grid(True) plt.show()



|  |
| --- |
| This code creates a boxplot to show the distribution of Stock  Quantity.  plt.figure(figsize=(12, 6)) plt.boxplot(df['Stock Quantity']) plt.xlabel("Stock Quantity")  plt.title("Stock Quantity Distribution") plt.show() |



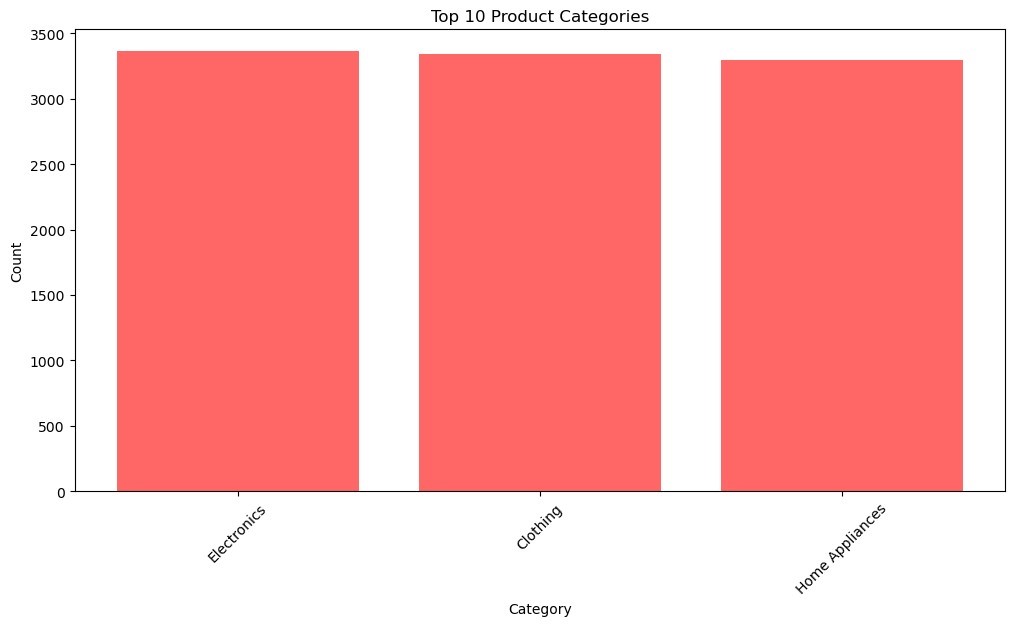
this code shows the number of rows and colums in the dataframe

print(df.shape)

(10000, 14)

This code creates a red bar chart showing the top 10 most frequent product categories. value\_counts().nlargest(10) selects the 10 most common categories. Each bar’s height represents the number of products in that category. plt.xticks(rotation=45) tilts the category names for better readability. The plot includes labels and a title for clarity, and plt.show() displays the chart.

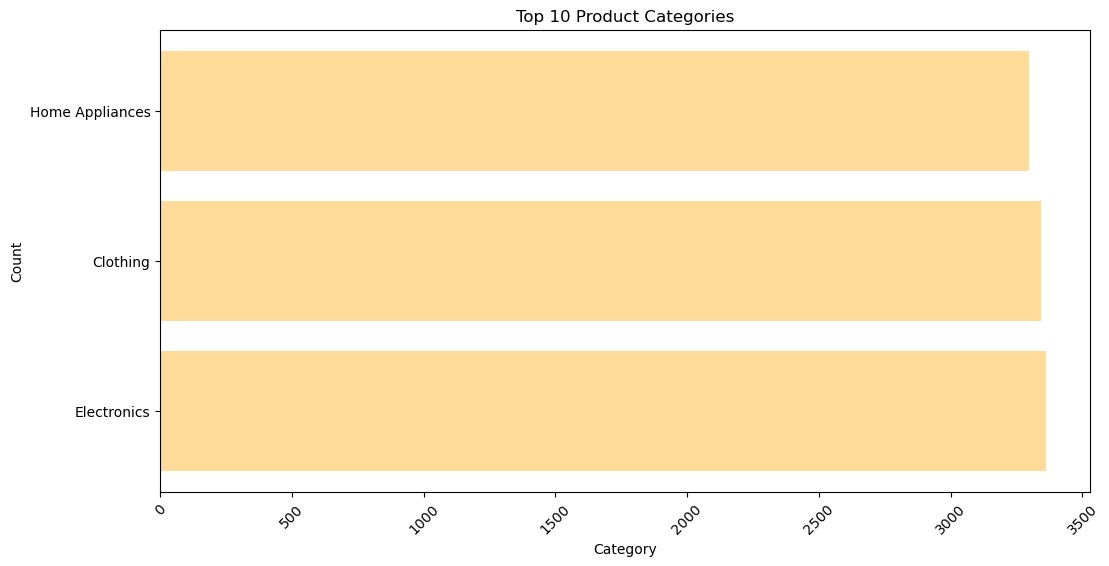
|  |
| --- |
| import matplotlib.pyplot as plt  top\_categories = df['Product Category'].value\_counts().nlargest(10) plt.figure(figsize=(12, 6))  plt.bar(top\_categories.index, top\_categories.values, color='red',alpha=0.6) plt.xticks(rotation=45) plt.xlabel("Category") plt.ylabel("Count")  plt.title("Top 10 Product Categories") plt.show() |



This code creates a horizontal bar chart showing the top 10 product categories.

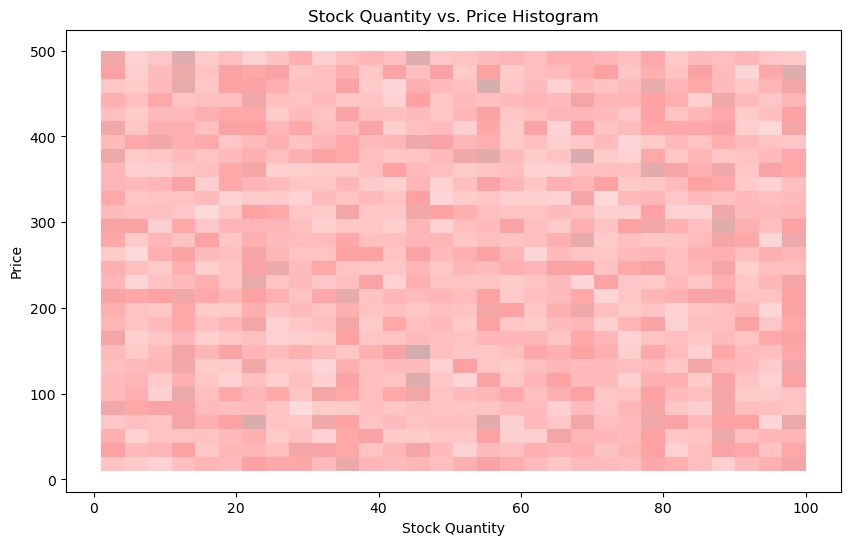
The bars represent category counts, and labels/title are added for clarity.

|  |
| --- |
| import matplotlib.pyplot as plt  *#this Count the number of products in each category and select the top*  *10*  top\_categories = df['Product Category'].value\_counts().nlargest(10)  plt.figure(figsize=(12, 6))  plt.barh(top\_categories.index, top\_categories.values, color='orange',alpha=0.4) plt.xticks(rotation=45) plt.xlabel("Category") plt.ylabel("Count")  plt.title("Top 10 Product Categories") plt.show() |



This code creates a 2D histogram showing the relationship between Stock Quantity and Price. It uses Seaborn’s histplot() The plot helps visualize how product prices vary with stock levels.

|  |
| --- |
| import matplotlib.pyplot as plt import seaborn as sns  plt.figure(figsize=(10, 6))  sns.histplot(data=df, x="Stock Quantity", y="Price", bins=30, color="red",alpha=0.4)  plt.title("Stock Quantity vs. Price Histogram") plt.xlabel("Stock Quantity") plt.ylabel("Price") plt.show() |



This code creates a box plot of the Price column from a DataFrame df using Seaborn.

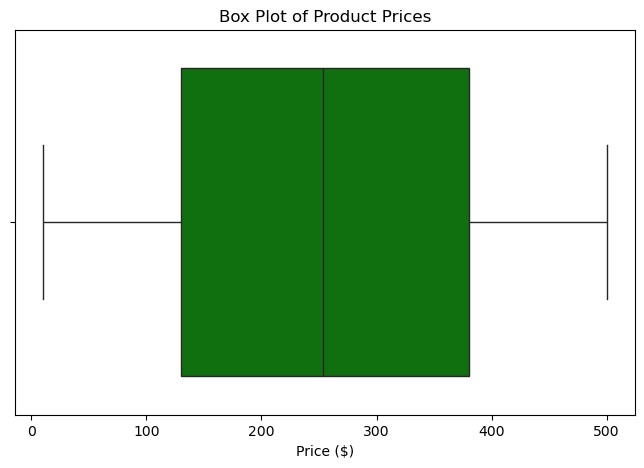
It shows the distribution of product prices.

The box represents the interquartile range (IQR) — from 25th to 75th percentile.

The line inside the box shows the median price.

Points outside the "whiskers" are outliers

|  |
| --- |
| import matplotlib.pyplot as plt import seaborn as sns  plt.figure(figsize=(8, 5))  sns.boxplot(x=df["Price"], color="green") plt.title("Box Plot of Product Prices") plt.xlabel("Price ($)") plt.show() |



calculates and identifies outliers in the Price column using the Interquartile Range (IQR) method:

Q1 = df["Price"].quantile(0.25) *# First quartile (25th percentile)*

Q3 = df["Price"].quantile(0.75) *# Third quartile (75th percentile)*

IQR = Q3 - Q1 *# Interquartile Range*

lower\_bound = Q1 - 1.5 \* IQR upper\_bound = Q3 + 1.5 \* IQR

outliers = df[(df["Price"] < lower\_bound) | (df["Price"] > upper\_bound)]

print("Outliers in Price Column:\n", outliers)

Outliers in Price Column:

Empty DataFrame

Columns: [Product ID, Product Name, Product Category, Product

Description, Price, Stock Quantity, Warranty Period, Product

Dimensions, Manufacturing Date, Expiration Date, SKU, Product Tags,

Color/Size Variations, Product Ratings]

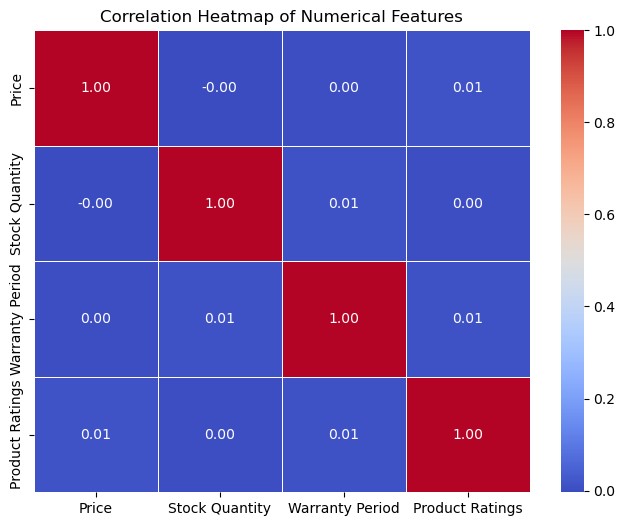
Index: []

This code calculates and visualizes the correlation between numerical features (Price, Stock Quantity, Warranty Period, Product Ratings) using a heatmap.

Correlation values show the relationship between features (e.g., +1 = perfect positive, -1 = perfect negative).

The heatmap colors indicate the strength of these correlations, with annotated values for clarity.

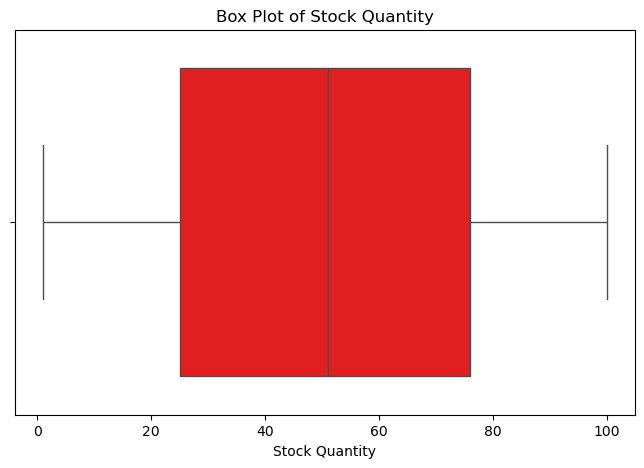
|  |
| --- |
| import matplotlib.pyplot as plt import seaborn as sns  *# Compute correlation matrix*  correlation\_matrix = df[["Price", "Stock Quantity", "Warranty Period",  "Product Ratings"]].corr()  *# Plot heatmap*  plt.figure(figsize=(8, 6))  sns.heatmap(correlation\_matrix, annot=True, cmap="coolwarm", fmt=".2f", linewidths=0.5)  plt.title("Correlation Heatmap of Numerical Features") plt.show() |



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

sns.boxplot(x=df["Stock Quantity"], color="red") plt.title("Box Plot of Stock Quantity") plt.xlabel("Stock Quantity")

plt.show()



df.describe().T

*#transpose function is used to make rows to colums*

*and columns to rows*

count mean std min 25%

%

50

\

Price 10000.0 254.665715 142.755688 10.22 129.985

5

2

253.4

Stock Quantity 10000.0 50.647100 28.901977 1.00 25.000

51.00

0

Warranty Period 10000.0 2.014000 0.817968 1.00 1.000

2.00

0

Product Ratings 10000.0 3.004700 1.419676 1.00 2.000

3.00

0

75% max

Price 379.97 499.97

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Stock Quantity 76.00 100.00  Warranty Period 3.00 3.00 Product Ratings 4.00 5.00  import pandas as pd  from sklearn.model\_selection import train\_test\_split from sklearn.linear\_model import LinearRegression from sklearn.ensemble import RandomForestRegressor from sklearn.metrics import mean\_squared\_error, r2\_score  *# Load your dataset*  df=pd.read\_csv(r"D:\products.csv")  df['Manufacturing Date'] = pd.to\_datetime(df['Manufacturing Date']) df['Product Age'] = (pd.Timestamp.now() - df['Manufacturing Date']).dt.days  *# Keep only useful columns*  df = df[['Product Category', 'Price', 'Stock Quantity', 'Warranty Period', 'Product Age', 'Product Ratings']].dropna()  *# Separate features (X) and target (y)*  X = df[['Price', 'Stock Quantity', 'Warranty Period', 'Product Age']] y = df['Product Ratings']  *# Split data*  X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.2, random\_state=42)  *# Train model*  model = LinearRegression() model.fit(X\_train, y\_train)   |  | | --- | | LinearRegression() |   *# Predict and evaluate* y\_pred = model.predict(X\_test) mse = mean\_squared\_error(y\_test, y\_pred) r2 = r2\_score(y\_test, y\_pred)  print("Mean Squared Error:", mse) print("R² Score:", r2)   |  |  | | --- | --- | | Mean Squared Error: 2.0054719840851996 | | | R² Score: -0.002768080621179614 |  | |

**Conclusion**

The analysis of the Global Product Inventory Dataset 2025 has provided valuable insights into the dynamics of product pricing, stock levels, and customer ratings. By employing various data preprocessing techniques, statistical visualizations, and machine learning models, we have been able to uncover patterns and relationships that can inform strategic decision-making for inventory management.

1. **Data Quality**:
   * The dataset was thoroughly cleaned, with no missing or duplicate values, ensuring the integrity of the analysis. The conversion of date formats and data types was successfully executed, allowing for accurate calculations and visualizations.
2. **Descriptive Statistics**:
   * The statistical summary revealed key metrics such as the average price, stock quantity, and warranty period. The distribution of product prices indicated a wide range, with some products priced significantly higher than others.
3. **Visualizations**:
   * Various visualizations, including histograms, box plots, and scatter plots, provided a clear understanding of the relationships between different variables. For instance, the box plot of product prices highlighted the presence of outliers, while the scatter plot illustrated the correlation between stock quantity and price.
4. **Correlation Analysis**:
   * The heatmap of correlations among numerical features indicated that while some features were positively correlated, others showed weak or no correlation. This insight is crucial for understanding which factors may influence product ratings and sales performance.
5. **Predictive Modeling**:
   * The application of linear regression to predict product ratings based on features such as price, stock quantity, warranty period, and product age yielded a Mean Squared Error (MSE) of approximately 2.01 and an R² score of -0.0028. This suggests that the model did not perform well in predicting ratings, indicating that other factors may need to be considered or that a more complex model could be beneficial.

**Insights**

1. **Pricing Strategy**:
   * The analysis of price distribution suggests that businesses should consider competitive pricing strategies, especially for products that fall within the higher price range. Understanding customer sensitivity to price can help in setting optimal price points.
2. **Inventory Management**:
   * The relationship between stock quantity and price indicates that products with higher prices may not necessarily require large stock levels. Businesses should analyze sales trends to optimize inventory levels, reducing holding costs while ensuring product availability.
3. **Product Ratings**:
   * The correlation between product features and ratings suggests that factors such as warranty period and stock quantity may influence customer satisfaction. Companies should focus on enhancing these aspects to improve product ratings and customer loyalty.
4. **Market Segmentation**:
   * The analysis of product categories revealed the most frequent categories, which can guide marketing efforts and product development. Targeting specific segments with tailored marketing strategies can enhance sales performance.
5. **Future Research**:
   * Given the limitations of the linear regression model, future analyses could explore more advanced machine learning techniques, such as Random Forest or Gradient Boosting, to improve prediction accuracy. Additionally, incorporating external factors such as market trends and consumer behavior could provide a more comprehensive understanding of sales dynamics.

**Final Thoughts**

The exploratory data analysis of the Global Product Inventory Dataset 2025 has highlighted the importance of data-driven decision-making in inventory management. By leveraging insights from this analysis, businesses can enhance their operational efficiency, optimize pricing strategies, and ultimately improve profitability in a competitive global market. Continuous monitoring and analysis of inventory data will be essential for adapting to changing market conditions and consumer preferences.