Inventory Management Analysis Report

1. Introduction

Effective inventory management is a cornerstone of any successful business, helping to maintain optimal stock levels, reduce carrying costs, and ensure that customer demand is met without interruptions. This report presents a comprehensive analysis of sales and inventory data from Boxify, focusing on key inventory management metrics such as inventory turnover, stock-to-sales ratio, and reorder points. The insights and recommendations provided aim to enhance inventory management practices, improve profitability, and ensure operational efficiency.

1.2. Data Overview

The dataset consists of multiple columns that represent various aspects of the inventory and sales data, including:

- SKU_number: Unique identifier for each product.
- SoldCount: Number of units sold for each SKU.
- ItemCount: Available inventory count for each SKU.
- PriceReg: Regular price of the product.
- LowUserPrice and LowNetPrice: The lowest prices offered.
- ReleaseYear: Year the product was released.
- Reorder Points: Calculated to determine when a product should be restocked

1.3. Background

- Importance of inventory management in businesses.
- Overview of challenges like maintaining optimal stock levels, minimizing costs, and meeting customer demand.

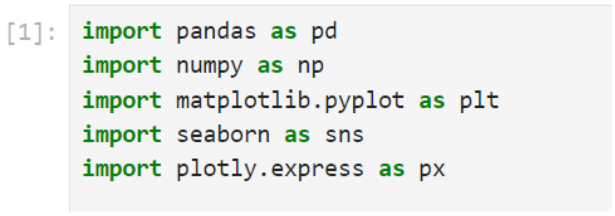
1.4. Objectives

- Analyze the sales dataset to understand sales trends, stock levels, and product performance.
- Identify popular products, low-stock items, and sales patterns over time.
- Generate actionable recommendations to enhance inventory management practices.

2. Methodology

2.1. Data Collection

- Source of the dataset: Sales Analysis Dataset.
- Description of the dataset structure and key variables.
- First, ensure we have all the necessary libraries installed. We can install any missing libraries using pip.
- Imported the required libraries in Jupyter Notebook.



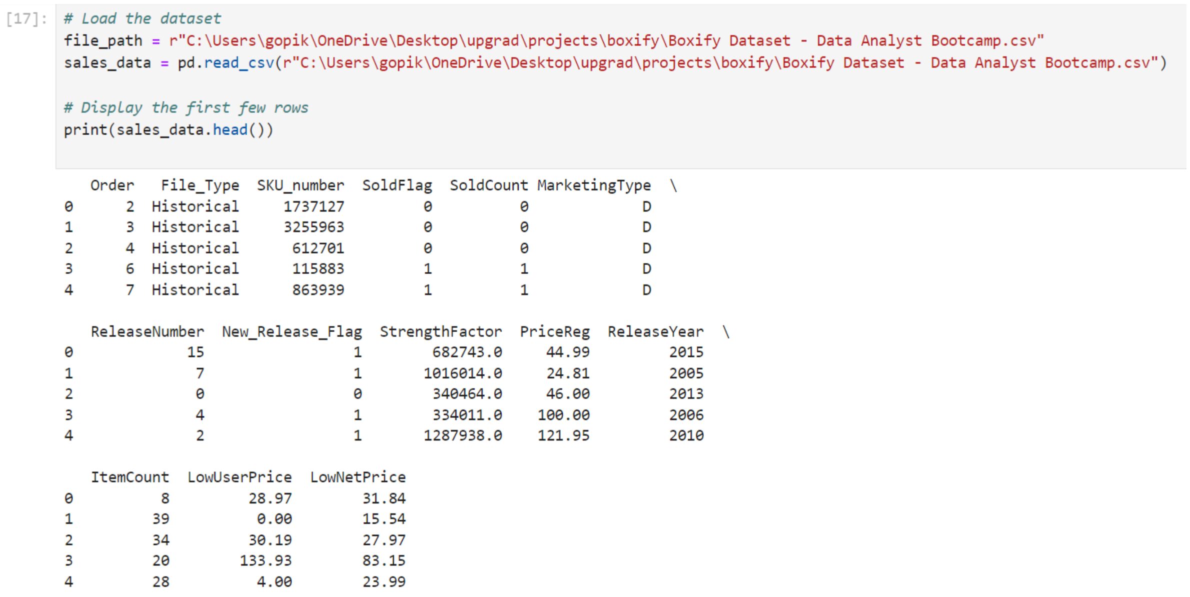
```
[1]: import pandas as pd
     import numpy as np
     import matplotlib.pyplot as plt
     import seaborn as sns
     import plotly.express as px
```

2.2. Data Preprocessing

- Steps taken to clean the data.
- Handling missing values and inconsistencies.
- Data transformation processes.

Data Collection and Preprocessing

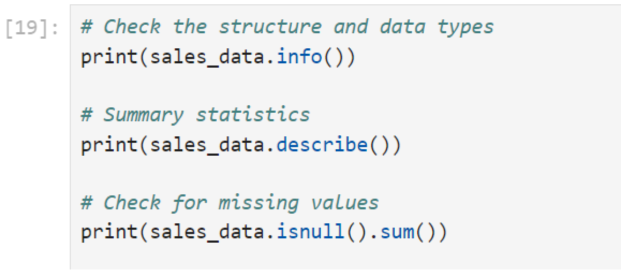
- **Load the Dataset**



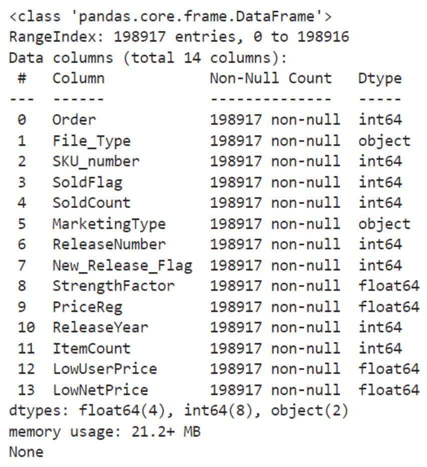
```
[17]: # Load the dataset
      file_path = r"C:\Users\gopik\OneDrive\Desktop\upgrad\projects\boxify\Boxify Dataset - Data Analyst Bootcamp.csv"
      sales_data = pd.read_csv(r"C:\Users\gopik\OneDrive\Desktop\upgrad\projects\boxify\Boxify Dataset - Data Analyst Bootcamp.csv")

      # Display the first few rows
      print(sales_data.head())
```

```
   Order   File_Type  SKU_number  SoldFlag  SoldCount MarketingType  \
0      2  Historical     1737127         0          0             D   
1      3  Historical     3255963         0          0             D   
2      4  Historical      612701         0          0             D   
3      6  Historical      115883         1          1             D   
4      7  Historical      863939         1          1             D   

   ReleaseNumber  New_Release_Flag  StrengthFactor  PriceReg  ReleaseYear  \
0             15                 1        682743.0     44.99         2015   
1              7                 1       1016014.0     24.81         2005   
2              0                 0        340464.0     46.00         2013   
3              4                 1        334011.0    100.00         2006   
4              2                 1       1287938.0    121.95         2010   

   ItemCount  LowUserPrice  LowNetPrice  
0          8         28.97        31.84  
1         39          0.00        15.54  
2         34         30.19        27.97  
3         20        133.93        83.15  
4         28          4.00        23.99  
```

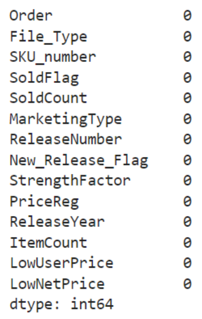
- **Inspect the Dataset:** Check the structure, data types, and summary statistics.



```
[19]: # Check the structure and data types
      print(sales_data.info())

      # Summary statistics
      print(sales_data.describe())

      # Check for missing values
      print(sales_data.isnull().sum())
```



```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 198917 entries, 0 to 198916
Data columns (total 14 columns):
 #   Column            Non-Null Count   Dtype  
---  ------            --------------   -----  
 0   Order             198917 non-null  int64  
 1   File_Type         198917 non-null  object 
 2   SKU_number        198917 non-null  int64  
 3   SoldFlag          198917 non-null  int64  
 4   SoldCount         198917 non-null  int64  
 5   MarketingType     198917 non-null  object 
 6   ReleaseNumber     198917 non-null  int64  
 7   New_Release_Flag  198917 non-null  int64  
 8   StrengthFactor    198917 non-null  float64
 9   PriceReg          198917 non-null  float64
 10  ReleaseYear       198917 non-null  int64  
 11  ItemCount         198917 non-null  int64  
 12  LowUserPrice      198917 non-null  float64
 13  LowNetPrice       198917 non-null  float64
dtypes: float64(4), int64(8), object(2)
memory usage: 21.2+ MB
None
```



```
Order               0
File_Type           0
SKU_number          0
SoldFlag            0
SoldCount           0
MarketingType       0
ReleaseNumber       0
New_Release_Flag    0
StrengthFactor      0
PriceReg            0
ReleaseYear         0
ItemCount           0
LowUserPrice        0
LowNetPrice         0
dtype: int64
```

- **Handle Missing Values and Inconsistencies:** Depending on the dataset, we might need to handle missing values or correct inconsistencies. Since here there were no missing values, we can skip that step.

2.3. Analytical Tools

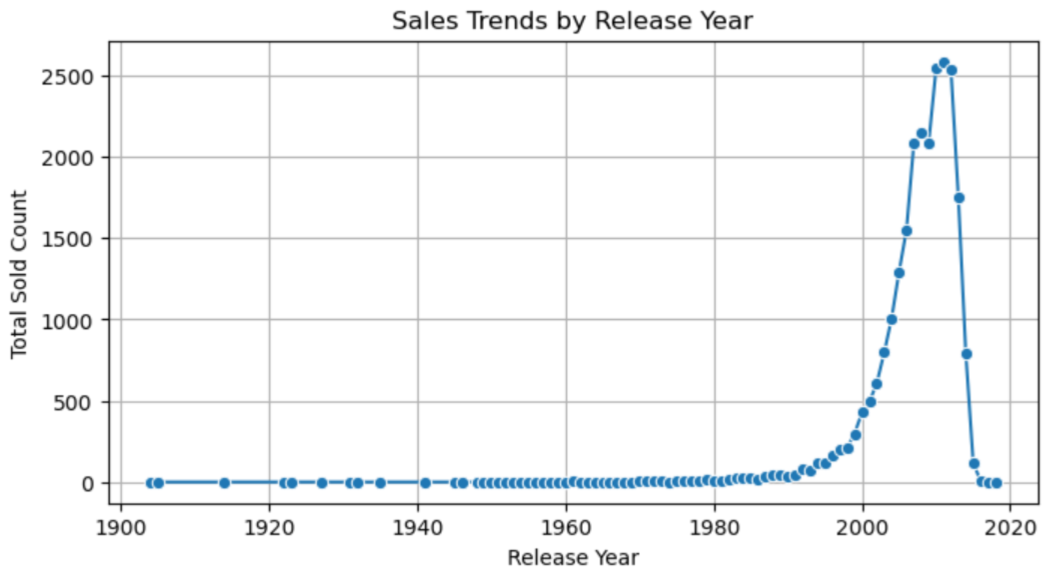
- Python libraries used: pandas, numpy, matplotlib, seaborn, plotly.
- Tools for visualization: Jupyter Notebook, Plotly for interactive charts.

3. Exploratory Data Analysis (EDA)

3.1. Sales Trends Over Time

- Analysis of how sales have evolved over the years.
- Insights on peak sales periods and declining trends.

```
[71]: # Filtering the data to only include reasonable ReleaseYear values (e.g., > 1900)
valid_sales_data = sales_data[sales_data['ReleaseYear'] > 1900]

# Re-plot the sales trends
import matplotlib.pyplot as plt
import seaborn as sns

# Grouping by ReleaseYear to sum SoldCount
sales_trends = valid_sales_data.groupby('ReleaseYear')['SoldCount'].sum().reset_index()

# Plotting the corrected sales trends
plt.figure(figsize=(8, 4))
sns.lineplot(data=sales_trends, x='ReleaseYear', y='SoldCount', marker='o')
plt.title('Sales Trends by Release Year')
plt.xlabel('Release Year')
plt.ylabel('Total Sold Count')
plt.grid(True)
plt.show()
```



- The chart shows a significant spike in sales between the late 1990s and early 2000s, indicating a peak in product demand during that period. After 2010, there is a sharp decline in sales, suggesting market saturation or a lack of product innovation.

Recommendations:

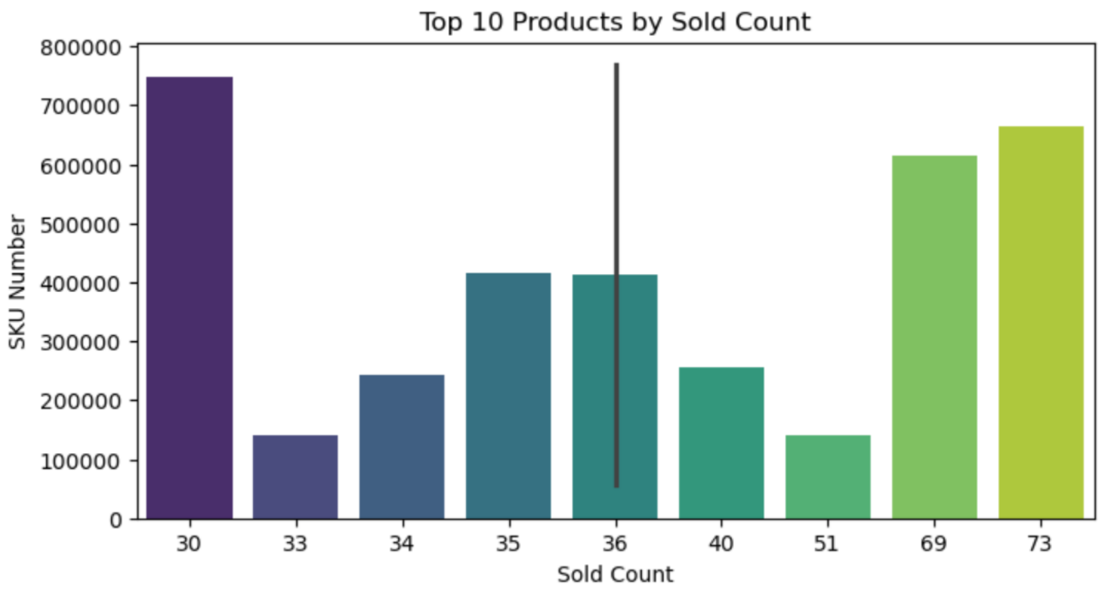
- The company should focus on reintroducing successful products from the peak years, explore new product lines, and strengthen marketing efforts to regain market momentum.
- Enhanced inventory management for high-demand products from the 2000s will also help meet potential resurgence in demand.

3.2. Top-Selling Products and Categories

- Identification of the most popular products and categories.

- Discussion on factors contributing to high sales.

```
[23]: # Top 10 products based on SoldCount
      top_products = sales_data.groupby('SKU_number')['SoldCount'].sum().sort_values(ascending=False).head(10).reset_index()

      # Plot top 10 products
      plt.figure(figsize=(10, 6))
      sns.barplot(data=top_products, x='SoldCount', y='SKU_number', palette='viridis')
      plt.title('Top 10 Products by Sold Count')
      plt.xlabel('Sold Count')
      plt.ylabel('SKU Number')
      plt.show()
```



- The highest-selling SKU is "30", with sales significantly higher than the others, followed by SKUs "73" and "69", which also have strong performance.
- SKU 30 is the clear leader with sales much higher than the rest, indicating that this product is a significant revenue driver for the company.
- There is a noticeable gap between the top-selling SKUs (30, 73, 69) and the rest of the products, indicating that a few products account for the majority of sales.

Recommendations:

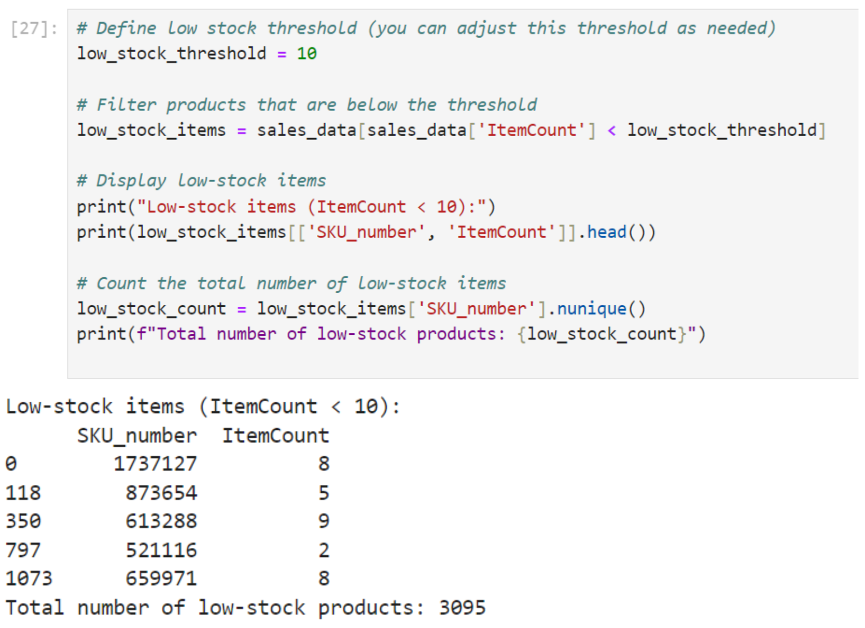
- Prioritize Restocking for Top Sellers: Ensure that high-selling products like SKU 30, 73, and 69 are always in stock to meet demand and avoid missed sales opportunities.

- Consider Expanding the Product Line for Best-Selling SKUs: Analyze customer preferences for these products and explore opportunities to expand or introduce similar products.

- Focus Marketing Efforts on Mid-Performing Products: For mid-performing SKUs like 35 and 36, consider running targeted promotions to boost their sales and bring them closer to the performance of the top sellers.

3.3. Stock Levels and Low-Stock Items

- Analysis of current stock levels.
- Identification of products that frequently run low on stock.
- To investigate stock levels and identify low-stock items, we can focus on the Item Count (which likely represents the stock level) and any potential thresholds to flag low-stock items.

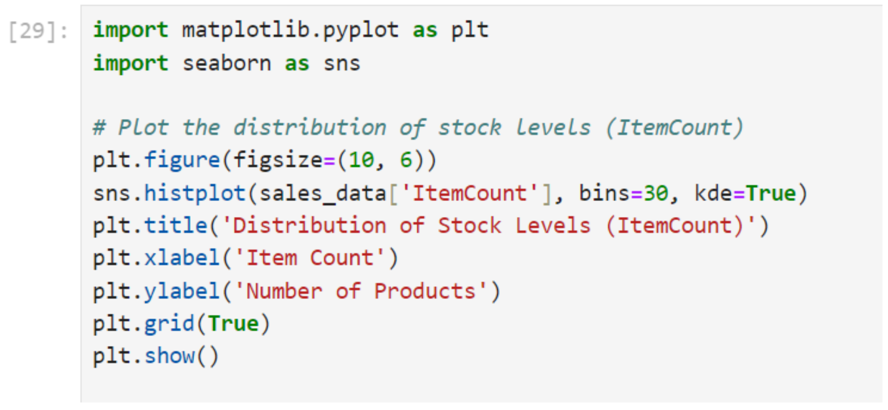


```
[27]: # Define low stock threshold (you can adjust this threshold as needed)
low_stock_threshold = 10

# Filter products that are below the threshold
low_stock_items = sales_data[sales_data['ItemCount'] < low_stock_threshold]

# Display low-stock items
print("Low-stock items (ItemCount < 10):")
print(low_stock_items[['SKU_number', 'ItemCount']].head())

# Count the total number of low-stock items
low_stock_count = low_stock_items['SKU_number'].nunique()
print(f"Total number of low-stock products: {low_stock_count}")
```

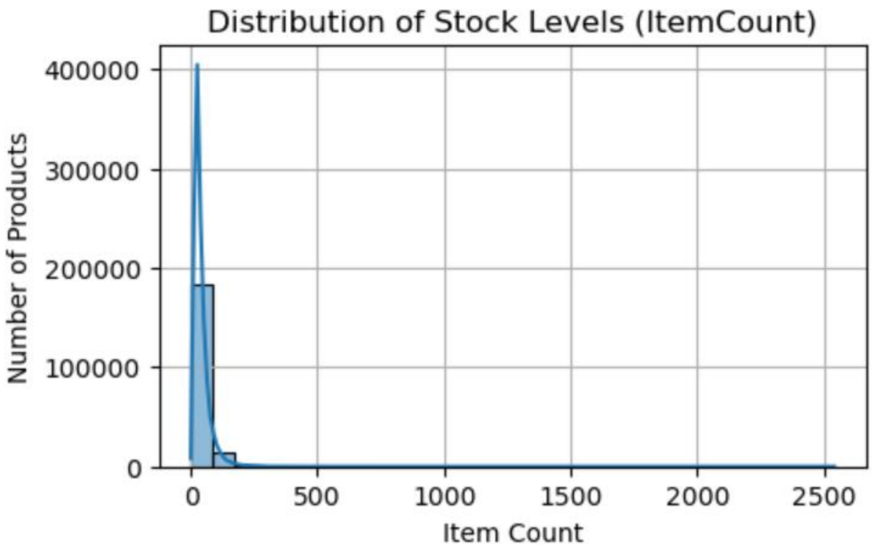
```
Low-stock items (ItemCount < 10):
      SKU_number  ItemCount
0        1737127          8
118       873654          5
350       613288          9
797       521116          2
1073      659971          8
Total number of low-stock products: 3095
```

3.3. 1 Visualizing Stock Levels

We can create a histogram or bar plot to visualize the distribution of stock levels across products



```
[29]: import matplotlib.pyplot as plt
import seaborn as sns

# Plot the distribution of stock levels (ItemCount)
plt.figure(figsize=(10, 6))
sns.histplot(sales_data['ItemCount'], bins=30, kde=True)
plt.title('Distribution of Stock Levels (ItemCount)')
plt.xlabel('Item Count')
plt.ylabel('Number of Products')
plt.grid(True)
plt.show()
```



- Highly Skewed Distribution: The majority of products have very low stock levels, with most items concentrated between 0 and 50 units. A few products have stock levels above 500, but these are rare.
- Outliers with High Stock Levels: A small number of products have exceptionally high stock levels (over 1000 units), indicating potential overstocking for these items.

Recommendations:

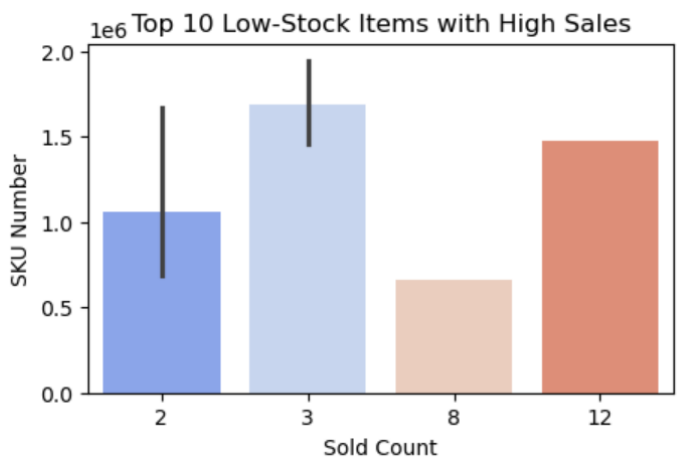
- Replenish Low-Stock Items: Since most products have low stock levels, prioritize restocking to avoid stockouts and meet customer demand, especially for high-demand items.
- Review Overstocked Items: Products with unusually high stock levels should be reviewed to avoid excess inventory costs. Implement strategies such as promotions or discounts to reduce the inventory for these overstocked items.
- Optimize Inventory Management: Use dynamic stock forecasting to ensure that stock levels align with demand trends, minimizing both stockouts and overstocking.

3.3.2 Identify Top Low-Stock Products with High Sales

This step will help us find products that have low stock but are in high demand (based on the SoldCount column).

```
[31]: # Filter products with low stock and high sales
      low_stock_high_sales = low_stock_items[low_stock_items['SoldCount'] > 0].sort_values(by='SoldCount', ascending=False)

      # Display top low-stock items with high sales
      print("Top low-stock items with high sales:")
      print(low_stock_high_sales[['SKU_number', 'ItemCount', 'SoldCount']].head())

      # Plot the top low-stock items with high sales
      plt.figure(figsize=(10, 6))
      sns.barplot(data=low_stock_high_sales.head(10), x='SoldCount', y='SKU_number', palette='coolwarm')
      plt.title('Top 10 Low-Stock Items with High Sales')
      plt.xlabel('Sold Count')
      plt.ylabel('SKU Number')
      plt.show()
```

```
Top low-stock items with high sales:
       SKU_number  ItemCount  SoldCount
26488     1475488          8         12
72303      664616          9          8
53028     1946541          8          3
19424     1453226          9          3
27786     1656831          9          3
```



- The chart highlights the Top 10 low-stock items with high sales. Each bar represents a product (SKU number) with high sales but low inventory levels. The x-axis shows the sold count, while the y-axis lists the SKU numbers.
- High Sales with Low Stock: SKU numbers 2, 3, 8, and 12 have significant sales but are likely at risk of stockouts due to their low stock levels.
- SKU 3 and 12 Stand Out: These SKU numbers have especially high sales, indicating strong demand, but their low stock levels suggest they may soon be unavailable if not restocked.

Recommendations:

- Immediate Restocking Required: SKU numbers 2, 3, 8, and 12 should be prioritized for restocking to meet continued demand and avoid missed sales opportunities due to stockouts.
- Automate Reorder Points: Implement automated reordering for these high-demand, low-stock items to prevent future stockouts and ensure they are always available for purchase.

- Monitor Sales Trends: Regularly review sales and stock levels for these SKUs to adjust inventory management strategies as needed, ensuring stock aligns with demand.

3.3.3 Recommendations Based on Stock Analysis:

1. Restocking Strategy:
    - Identify items that are frequently low in stock but in high demand. These products should be prioritized for restocking.
2. Automated Alerts:
    - Set up automated alerts when stock levels fall below a certain threshold to prevent stockouts, especially for high-selling products.
3. Inventory Replenishment Plan:
    - Develop a replenishment plan for items that consistently have low stock levels to maintain continuous availability.

3.4 Calculate Inventory Turnover:

To calculate the key performance indicators (KPIs) for inventory management such as inventory turnover, stock-to-sales ratio, and reorder points, we need to work with the following metrics:

1. Inventory Turnover Ratio:

- Formula:

  Inventory Turnover=Cost of Goods Sold (COGS) / Average Inventory

- The Inventory Turnover ratio measures how often the inventory is sold and replaced over a period of time.

2. Stock-to-Sales Ratio:

- Formula:

  Stock-to-Sales Ratio=Stock Level (Item Count) / Sales (SoldCount)

- This ratio helps to determine how much inventory is available relative to the sales. It can indicate overstocking or understocking.

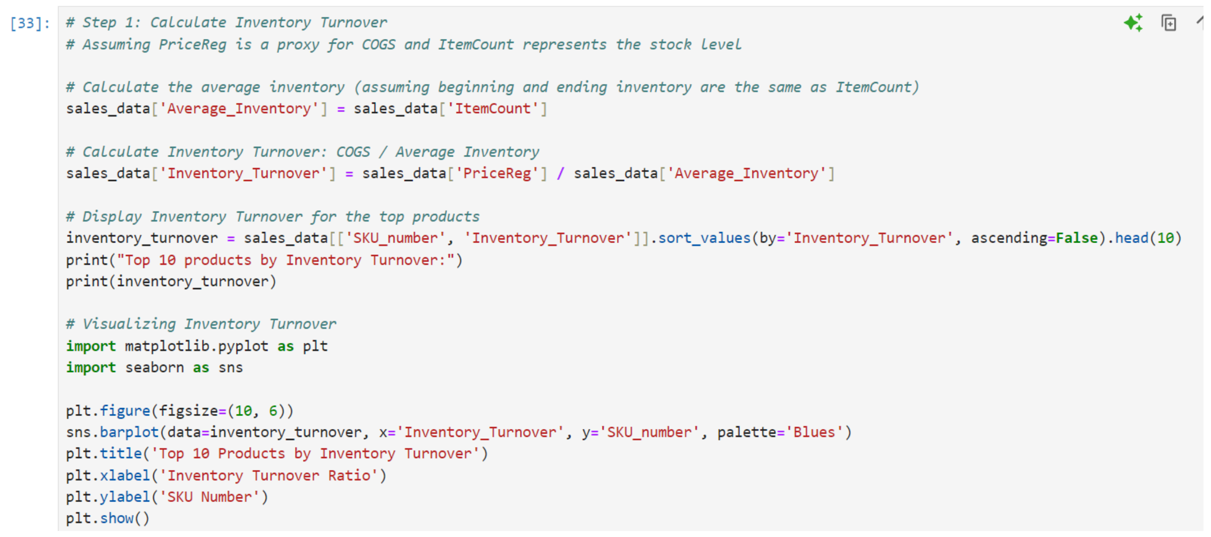
3. Reorder Point:

- Formula:

  Reorder Point= (Average Daily Usage × Lead Time) + Safety Stock

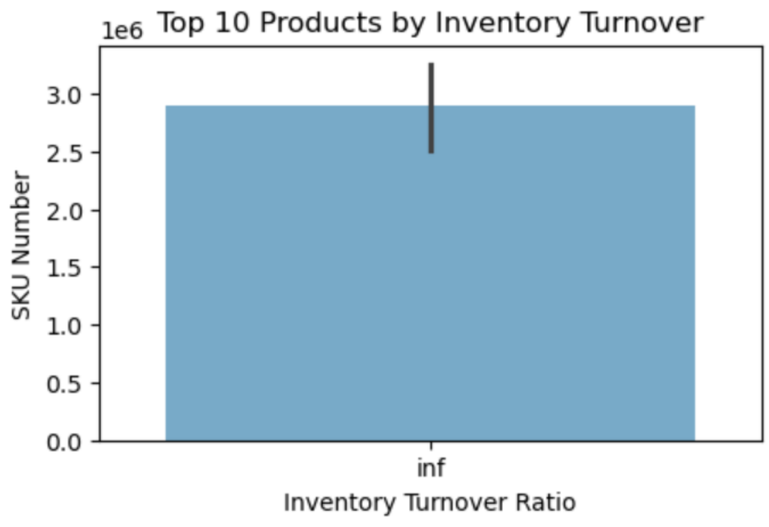
- The Reorder Point helps to determine when a new order should be placed to avoid stockouts.

Calculate Inventory Turnover:



```
[33]: # Step 1: Calculate Inventory Turnover
      # Assuming PriceReg is a proxy for COGS and ItemCount represents the stock level

      # Calculate the average inventory (assuming beginning and ending inventory are the same as ItemCount)
      sales_data['Average_Inventory'] = sales_data['ItemCount']

      # Calculate Inventory Turnover: COGS / Average Inventory
      sales_data['Inventory_Turnover'] = sales_data['PriceReg'] / sales_data['Average_Inventory']

      # Display Inventory Turnover for the top products
      inventory_turnover = sales_data[['SKU_number', 'Inventory_Turnover']].sort_values(by='Inventory_Turnover', ascending=False).head(10)
      print("Top 10 products by Inventory Turnover:")
      print(inventory_turnover)

      # Visualizing Inventory Turnover
      import matplotlib.pyplot as plt
      import seaborn as sns

      plt.figure(figsize=(10, 6))
      sns.barplot(data=inventory_turnover, x='Inventory_Turnover', y='SKU_number', palette='Blues')
      plt.title('Top 10 Products by Inventory Turnover')
      plt.xlabel('Inventory Turnover Ratio')
      plt.ylabel('SKU Number')
      plt.show()
```

```
Top 10 products by Inventory Turnover:
        SKU_number  Inventory_Turnover
41144      2751184                 inf
75467      3474212                 inf
89541      1855137                 inf
89657      1857970                 inf
102146     3136590                 inf
172734     3218991                 inf
145902     2485383                 inf
129808     3212060                 inf
52779      3635010                 inf
41858      3282329                 inf
```



- The table and bar chart display the Top 10 products (SKU numbers) by Inventory Turnover, which indicates how many times the inventory for a product is sold and replaced within a given period.

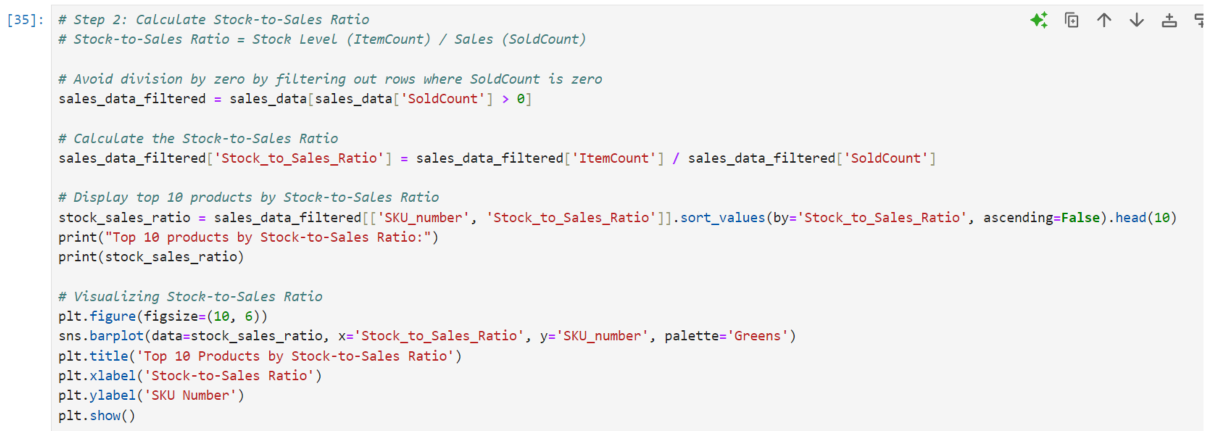
- Exceptionally High Inventory Turnover: The table shows several SKU numbers with extremely high inventory turnover ratios, marked as "inf" (infinite), meaning these products have either zero or negligible inventory but continue to have significant sales.

- Frequent Restocking Required: Products with such high turnover indicate that they are selling out very quickly and are likely in constant need of restocking to avoid stockouts.

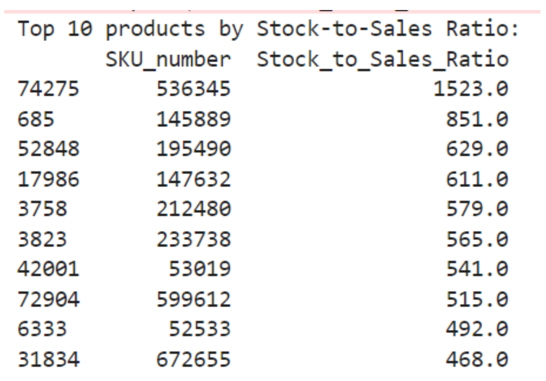
Recommendations:

- Urgent Restocking: Products like SKU 41144, 75467, and 89541, which have the highest turnover, must be restocked immediately to maintain sales and meet demand.
- Implement Automatic Reorder Points: For these high-turnover products, setting up automated reordering based on real-time stock levels will prevent potential stockouts and ensure product availability.
- Optimize Inventory Levels: Ensure that the stock levels for these fast-selling products are aligned with their high sales frequency to avoid the risk of frequent stock shortages, which can affect revenue.

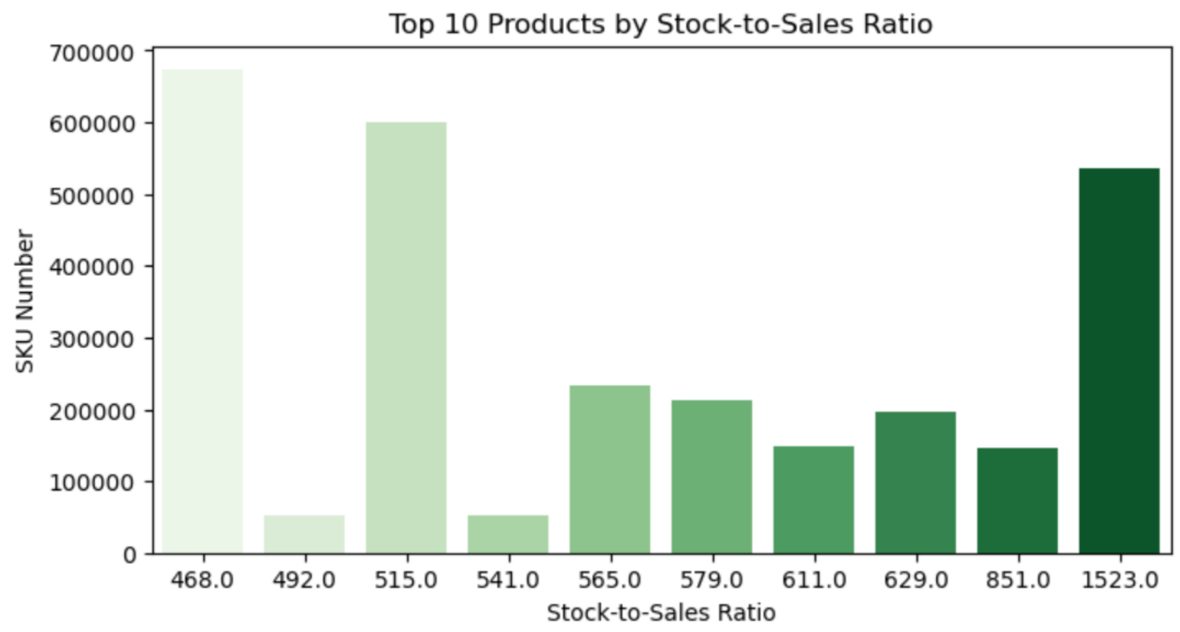
3.5. Calculate Stock-to-Sales Ratio:



```
[35]: # Step 2: Calculate Stock-to-Sales Ratio
      # Stock-to-Sales Ratio = Stock Level (ItemCount) / Sales (SoldCount)

      # Avoid division by zero by filtering out rows where SoldCount is zero
      sales_data_filtered = sales_data[sales_data['SoldCount'] > 0]

      # Calculate the Stock-to-Sales Ratio
      sales_data_filtered['Stock_to_Sales_Ratio'] = sales_data_filtered['ItemCount'] / sales_data_filtered['SoldCount']

      # Display top 10 products by Stock-to-Sales Ratio
      stock_sales_ratio = sales_data_filtered[['SKU_number', 'Stock_to_Sales_Ratio']].sort_values(by='Stock_to_Sales_Ratio', ascending=False).head(10)
      print("Top 10 products by Stock-to-Sales Ratio:")
      print(stock_sales_ratio)

      # Visualizing Stock-to-Sales Ratio
      plt.figure(figsize=(10, 6))
      sns.barplot(data=stock_sales_ratio, x='Stock_to_Sales_Ratio', y='SKU_number', palette='Greens')
      plt.title('Top 10 Products by Stock-to-Sales Ratio')
      plt.xlabel('Stock-to-Sales Ratio')
      plt.ylabel('SKU Number')
      plt.show()
```



```
Top 10 products by Stock-to-Sales Ratio:
       SKU_number  Stock_to_Sales_Ratio
74275      536345                1523.0
685        145889                 851.0
52848      195490                 629.0
17986      147632                 611.0
3758       212480                 579.0
3823       233738                 565.0
42001       53019                 541.0
72904      599612                 515.0
6333        52533                 492.0
31834      672655                 468.0
```



- The bar chart presents the Top 10 products based on their Stock-to-Sales Ratio, which measures how much stock is held relative to the sales volume. The higher the ratio, the more stock is available relative to the number of units sold.
- High Stock-to-Sales Ratios: Products with SKU numbers 468, 515, and 1523 show significantly high stock-to-sales ratios. This suggests that these products are overstocked compared to their actual sales.
- Overstocking Potential: These products have large amounts of inventory that are not selling quickly, which could result in high holding costs and potential obsolescence.

Recommendations:

- Reduce Inventory for High-Ratio Products: Products with high stock-to-sales ratios (e.g., SKU 468, 515, and 1523) should have their inventory levels reduced to avoid excess holding costs. Focus on adjusting stock levels to better align with sales demand.
- Promote Slow-Moving Items: Consider running targeted promotions or discounts on these overstocked products to clear inventory and improve the stock-to-sales balance.
- Dynamic Stock Adjustments: Continuously monitor stock-to-sales ratios and implement dynamic stock adjustments for better alignment between inventory levels and sales performance.

3.6 Calculate Reorder Point:

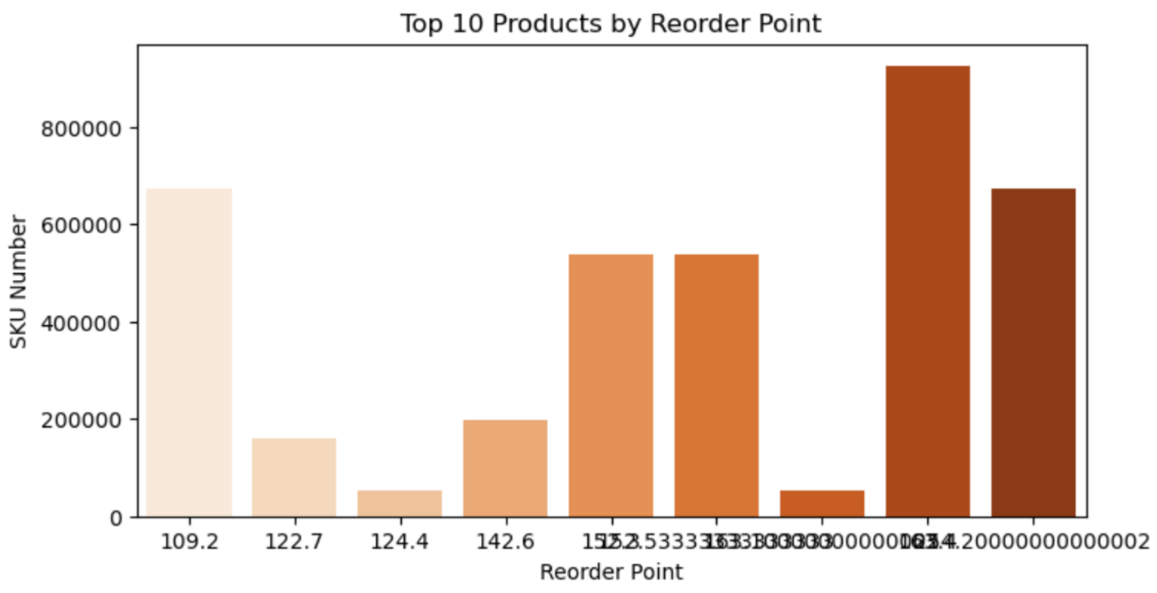
To calculate the reorder point, we need the following:

- Average Daily Usage: Total sales divided by the number of days.

- Lead Time: The time it takes to replenish inventory.
- Safety Stock: Extra stock to account for variability in demand.

```
[37]: # Step 3: Calculate Reorder Point
      # Assuming Average_Daily_Usage and Lead_Time are given or estimated

      # Example values for average daily usage and lead time
      sales_data['Average_Daily_Usage'] = sales_data['SoldCount'] / 30  # Assuming 30 days in a month
      sales_data['Lead_Time'] = 7  # Assuming a lead time of 7 days (this can be adjusted)

      # Assuming Safety Stock is 10% of the stock level (this can be adjusted)
      sales_data['Safety_Stock'] = 0.10 * sales_data['ItemCount']

      # Reorder Point = (Average Daily Usage * Lead Time) + Safety Stock
      sales_data['Reorder_Point'] = (sales_data['Average_Daily_Usage'] * sales_data['Lead_Time']) + sales_data['Safety_Stock']

      # Display top 10 products by Reorder Point
      reorder_point = sales_data[['SKU_number', 'Reorder_Point']].sort_values(by='Reorder_Point', ascending=False).head(10)
      print("Top 10 products by Reorder Point:")
      print(reorder_point)

      # Visualizing Reorder Points
      plt.figure(figsize=(10, 6))
      sns.barplot(data=reorder_point, x='Reorder_Point', y='SKU_number', palette='Oranges')
      plt.title('Top 10 Products by Reorder Point')
      plt.xlabel('Reorder Point')
      plt.ylabel('SKU Number')
      plt.show()
```

```
Top 10 products by Reorder Point:
        SKU_number  Reorder_Point
125452      672549     254.200000
98496       924198     163.400000
117615       53985     163.100000
74275       536345     152.533333
195045      536345     152.300000
75332       198645     142.600000
197557      198645     142.600000
114413       54246     124.400000
118411      160073     122.700000
123862      672653     109.200000
```



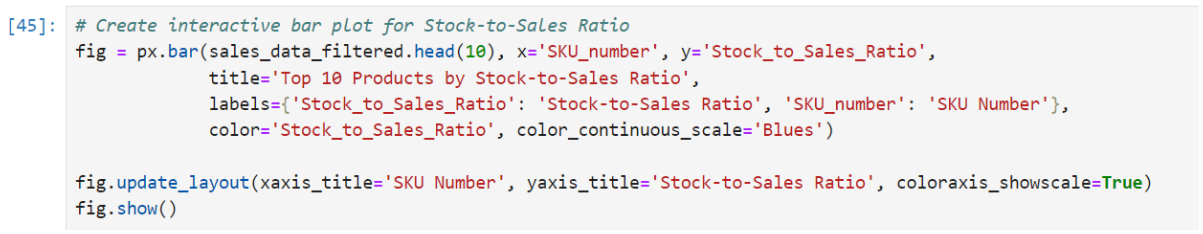
- The chart and table display the Top 10 products by Reorder Point, which indicates the stock level at which a new order should be placed to replenish the inventory. Products with higher reorder points require restocking more frequently due to higher demand or lead times.
- High Reorder Points for SKU 125452 and 98496: These SKUs have the highest reorder points, with SKU 125452 requiring restocking once inventory drops to 254 units and SKU 98496 needing restocking at 163 units. These high reorder points suggest that these products have high sales volumes or longer lead times.
- Moderate to Low Reorder Points for Other SKUs: Products like SKU 123862 and SKU 118411 have lower reorder points, indicating they need to be reordered less frequently.

Recommendations:

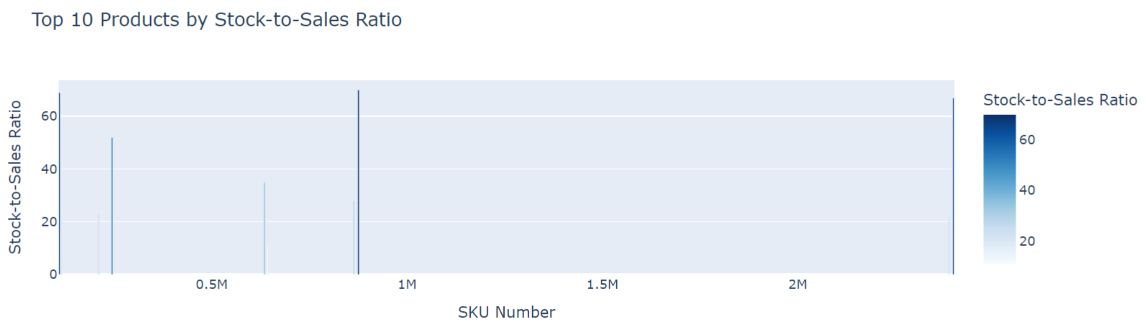
- Ensure Adequate Stock for High Reorder Point Products: Products with high reorder points (e.g., SKU 125452 and 98496) should be monitored closely, and restocking should occur promptly to prevent stockouts. Automate reorder processes for these SKUs to ensure continuous availability.
- Evaluate Lead Time and Demand: For products with moderate reorder points, regularly review lead times and sales trends to adjust the reorder point if necessary. This will help maintain optimal stock levels.
- Stock Optimization: For lower reorder point products, ensure that inventory does not exceed required levels to minimize holding costs, but still meets demand.

4. Stock-to-Sales Ratio (Interactive Bar Plot with Plotly)

This visualization shows the relationship between stock levels and sales. A high stock-to-sales ratio may indicate overstocking, while a low ratio may indicate understocking.



```
[45]: # Create interactive bar plot for Stock-to-Sales Ratio
      fig = px.bar(sales_data_filtered.head(10), x='SKU_number', y='Stock_to_Sales_Ratio',
                   title='Top 10 Products by Stock-to-Sales Ratio',
                   labels={'Stock_to_Sales_Ratio': 'Stock-to-Sales Ratio', 'SKU_number': 'SKU Number'},
                   color='Stock_to_Sales_Ratio', color_continuous_scale='Blues')

      fig.update_layout(xaxis_title='SKU Number', yaxis_title='Stock-to-Sales Ratio', coloraxis_showscale=True)
      fig.show()
```



- This chart illustrates the Stock-to-Sales Ratio for the top 10 products (SKU numbers). The stock-to-sales ratio measures the relationship between the available stock and the number of units sold. A higher ratio suggests overstocking, while a lower ratio indicates efficient stock utilization or potential understocking.
- High Stock-to-Sales Ratios: Some SKUs have extremely high stock-to-sales ratios, implying that these products are overstocked relative to their sales performance. This could lead to excess inventory holding costs and inefficient use of storage space.
- Imbalance in Stock Levels: The wide variation in the stock-to-sales ratios across SKUs suggests inconsistent stock management, with some products being significantly overstocked while others may be understocked.

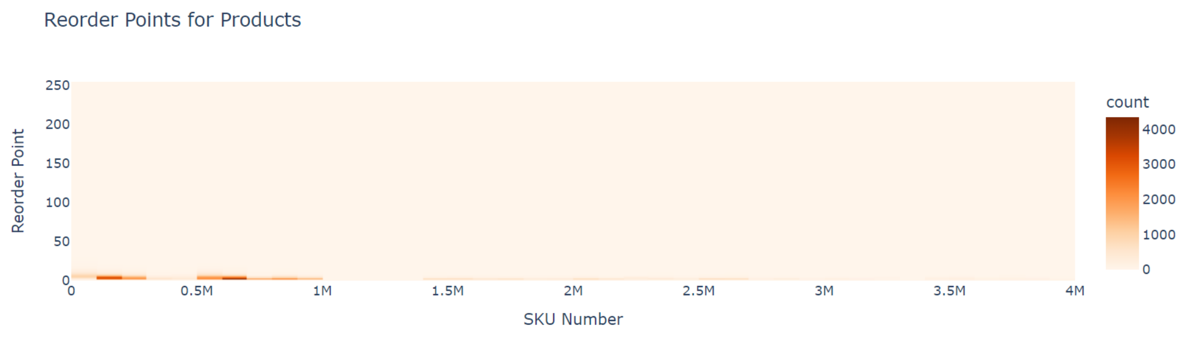
Recommendations:

- Reduce Overstocking: Focus on reducing the stock levels for products with high stock-to-sales ratios to avoid excess holding costs. Implement sales promotions or discounts to move the overstocked items.
- Monitor Stock Efficiency: Regularly track the stock-to-sales ratio to ensure that stock levels are aligned with demand. Adjust inventory replenishment strategies for products that consistently show a high ratio.
- Optimize Reordering Strategy: Use this analysis to adjust reorder points for overstocked products, reducing future restocking until sales pick up or stock levels normalize.

5. Reorder Points (Interactive Heatmap with Plotly)

This heatmap will help visualize reorder points for various products, highlighting the urgency for restocking.

```
[47]: # Create an interactive heatmap for Reorder Points
      fig = px.density_heatmap(sales_data, x='SKU_number', y='Reorder_Point', title='Reorder Points for Products',
                               labels={'Reorder_Point': 'Reorder Point', 'SKU_number': 'SKU Number'},
                               color_continuous_scale='Oranges')

      # Customize the chart layout
      fig.update_layout(xaxis_title='SKU Number', yaxis_title='Reorder Point')
      fig.show()
```



- This heatmap shows the Reorder Points for different products (SKU numbers). Reorder points indicate the stock level at which a product should be restocked to prevent stockouts. The colour intensity in the heatmap corresponds to the number of products that share the same reorder point, with darker areas representing more products at that specific reorder level.
- Concentration of Reorder Points at Lower Levels: The majority of the products have a reorder point below 50 units, indicating that they require restocking once inventory reaches a relatively low level.
- Few Products with High Reorder Points: Only a few products have higher reorder points, around 250 units. These products likely have higher demand or longer lead times that require larger reorder quantities.
- Skewed Distribution: Most products have reorder points concentrated between SKU numbers less than 1 million, with very few products having reorder points spread across higher SKU numbers.

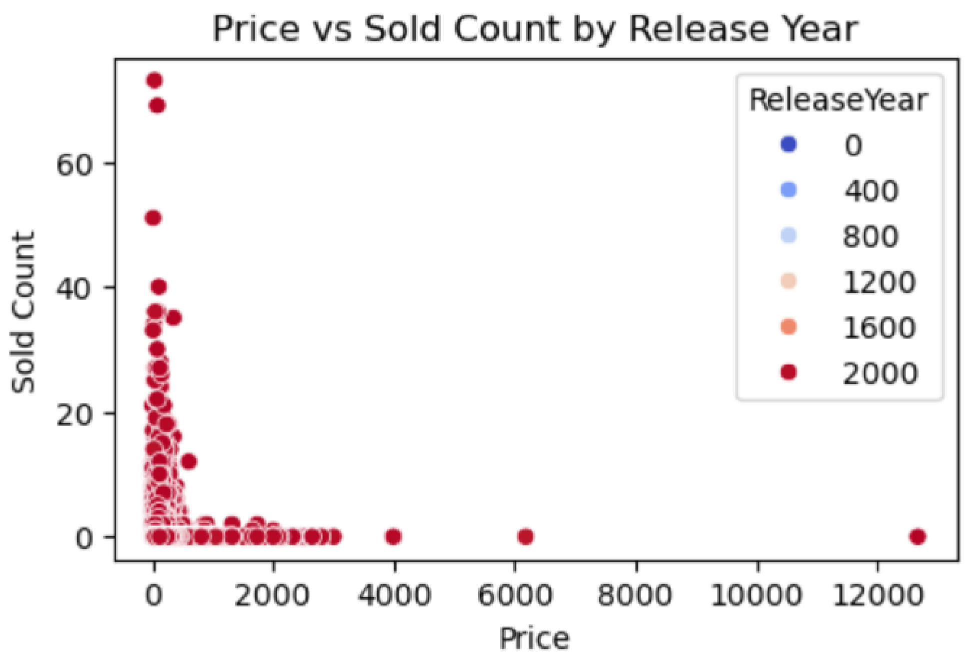
Recommendations:

- Ensure Timely Restocking for Products with High Reorder Points: Products with higher reorder points (around 250 units) should be closely monitored to ensure they are restocked promptly to prevent any interruptions in availability.
- Optimize Reorder Levels for Low Reorder Point Products: Reassess the reorder points for products with very low reorder levels (below 50) to ensure they are aligned with sales demand and stock turnover rates. This will help balance stock levels and prevent frequent reorders.

- Use Dynamic Reordering: Implement dynamic reordering systems that adjust reorder points based on real-time sales data, ensuring that stock levels stay aligned with demand fluctuations.

2.3. Price Analysis (PriceReg vs. SoldCount)

```
[25]: # Relationship between PriceReg and SoldCount
      plt.figure(figsize=(10, 6))
      sns.scatterplot(data=sales_data, x='PriceReg', y='SoldCount', hue='ReleaseYear', palette='coolwarm')
      plt.title('Price vs Sold Count by Release Year')
      plt.xlabel('Price')
      plt.ylabel('Sold Count')
      plt.show()
```



- This scatter plot compares Price with Sold Count for various products, differentiated by their Release Year. Each dot represents a product, with the x-axis showing the product price and the y-axis showing the number of units sold.
- Concentration of Sales at Lower Prices: The majority of products with high sold counts are priced below 2000, indicating that lower-priced products tend to sell in larger volumes.
- Few High-Priced Sales: There are very few products priced above 4000 that have notable sales. This suggests that products priced at higher levels have much lower demand.
- Older Products: Products released in earlier years (e.g., the year 2000, represented by darker red) dominate the sales, with relatively lower-priced items leading in terms of sold count.

Recommendations:

1. Focus on Lower-Priced Products: Since products priced below 2000 have the highest sales volumes, focus inventory and marketing efforts on these items to maximize revenue.
2. Reassess Pricing for High-Cost Items: Products priced above 4000 show minimal sales. Consider reducing the price or creating promotional bundles to boost demand for these items.
3. Continued Support for Popular Older Products: The high sales of older products released in 2000 indicate ongoing demand. Ensure sufficient stock and targeted marketing for these products to maintain their market performance.

Summary of Analysis, Inventory Insights, and Recommendations

1. Sales Trends by Release Year

- Findings: Sales peaked between the late 1990s and early 2000s, followed by a steep decline after 2010.
- Recommendations: Focus on restocking and reintroducing successful products from the 2000s. Implement product innovation and targeted marketing to revitalize declining sales.

2. Top 10 Products by Sold Count

- Findings: Products like SKU 30 and SKU 73 dominate sales, with a significant gap between top sellers and mid-performing products.
- Recommendations: Prioritize inventory for top-selling products to avoid stockouts. For mid-performing products, consider running promotions to boost sales.

3. Distribution of Stock Levels (ItemCount)

- Findings: Most products have low stock levels, with a few outliers showing high inventory.
- Recommendations: Replenish low-stock items to avoid stockouts. Review overstocked items to reduce holding costs, potentially through promotions or discounts.

4. Low-Stock Items with High Sales

- Findings: Several SKUs, such as 2, 3, and 12, have high sales but are at risk of stockouts due to low inventory.
- Recommendations: Immediately restock these high-demand, low-stock items. Automate reordering processes to maintain adequate inventory levels.

5. Top 10 Products by Inventory Turnover

- Findings: Products like SKU 41144 and SKU 75467 have very high inventory turnover, indicating rapid sales.
- Recommendations: Automate the reordering process for high-turnover products to prevent stockouts. Optimize stock levels to align with rapid sales frequencies.

6. Top 10 Products by Stock-to-Sales Ratio

- Findings: Some products have high stock-to-sales ratios, indicating potential overstocking.
- Recommendations: Reduce inventory levels for overstocked products to minimize holding costs. Focus on promotions or discounts to clear excess stock.

7. Top 10 Products by Reorder Point

- Findings: Products like SKU 125452 and 98496 have high reorder points, indicating frequent restocking needs.
- Recommendations: Closely monitor and promptly restock high-reorder-point products to avoid stockouts. Adjust reorder points dynamically based on real-time demand.

8. Price vs. Sold Count by Release Year

- Findings: Lower-priced products (below 2000) account for the majority of sales, while higher-priced items struggle to generate demand.
- Recommendations: Focus on lower-priced products to maximize revenue. For high-priced items, consider price reductions or bundling strategies to boost sales.

Inventory-Driven Insights

- Restocking High-Demand Products: Ensure that products with high sales volumes and low stock levels are restocked regularly to avoid stockouts. This will maximize sales potential and customer satisfaction.
- Reducing Overstocking: Identify and address overstocked items by lowering inventory levels or running promotional campaigns. Overstocked items tie up capital and increase holding costs, which can be minimized through effective stock control.
- Dynamic Reordering: Implement automated reorder points for products with high turnover rates and high demand. Dynamic reordering based on real-time sales data ensures optimal inventory levels and reduces the risk of stockouts.

- Product Lifecycle Management: For products that have reached their decline phase, review stock levels and reduce orders to prevent excess inventory. Focus on product innovation and introducing new products to maintain sales growth.
- Price Optimization: Since lower-priced items drive more sales, adjust pricing strategies to ensure that price-sensitive products remain competitive. For higher-priced items, consider reducing prices or offering promotional bundles to boost sales.

How Inventory-Focused Insights Benefit Businesses

1. Improved Stock Management: By identifying high-demand, low-stock items, businesses can focus on maintaining adequate inventory levels, reducing the risk of stockouts and maximizing sales potential.
2. Reduced Holding Costs: Addressing overstocked products through better stock control, price reductions, or promotions can reduce storage costs and free up capital for more profitable inventory.
3. Increased Revenue: Optimizing stock levels for high-turnover and high-demand products ensures that these products remain available, resulting in higher sales volumes and improved customer satisfaction.
4. Enhanced Inventory Efficiency: Automated and dynamic reordering systems ensure that businesses maintain the right balance between supply and demand, reducing excess inventory while ensuring that popular products are always available.
5. Better Decision-Making: Data-driven insights from sales and inventory metrics enable businesses to make informed decisions about pricing, stock levels, and product lifecycle management, ensuring that resources are allocated effectively.

Conclusion

This analysis has provided valuable insights into sales patterns, stock levels, and key inventory management metrics for Boxify's products. The findings highlight several areas where inventory management practices can be optimized to improve operational efficiency and profitability.

Key Takeaways:

- High-demand products should be prioritized for restocking, especially those with low stock levels to avoid missed sales opportunities. Automating reorder points for these products will ensure continuous availability.
- Overstocked products present an opportunity to reduce holding costs through dynamic stock adjustments, promotions, or discounts. Addressing overstocking will free up capital and storage space for more profitable items.

- Pricing strategies can be optimized to focus on lower-priced products, which are driving the majority of sales. For higher-priced items, strategic price reductions or bundling could stimulate demand.
- Dynamic inventory systems should be implemented to align stock levels with real-time demand, ensuring optimal inventory management across all product categories.

By addressing the issues of overstocking and stockouts, and by optimizing product lifecycle management, Boxify can reduce operational costs, improve customer satisfaction, and enhance overall profitability. These data-driven insights and recommendations will empower Boxify to make better inventory decisions, ensuring sustained growth and operational excellence.