BOXIFY: SALES ANALYSIS AND INVENTORY INSIGHTS

By

Shilpi Verma

A Project Report

Submitted to

UpGrad- Data Analyst Bootcamp

On

30 July 2024

Table of Contents

Abstract

Effective inventory management is crucial for businesses to optimize stock levels, minimize costs, and meet customer demand. This project employs data analysis techniques to explore a sales dataset, aiming to uncover sales trends, product performance insights, and inventory management recommendations. The methodology involves data collection, preprocessing to ensure data integrity, and extensive exploratory data analysis (EDA) to identify top-selling products, analyze sales patterns over time, and assess inventory levels. Key performance indicators such as inventory turnover and stock-to-sales ratio are calculated to provide actionable recommendations for enhancing inventory efficiency. Visualization tools are utilized to present findings interactively, aiding in the understanding and implementation of proposed strategies. The project culminates in a detailed report that outlines insights and practical recommendations, empowering businesses to optimize their inventory management practices effectively.

Introduction

In today's competitive business landscape, efficient inventory management is pivotal for companies aiming to balance stock levels, minimize costs, and meet customer expectations seamlessly. This project focuses on leveraging data analytics to enhance inventory management practices through a detailed analysis of a sales dataset. By exploring sales trends, identifying top-performing products, and evaluating inventory levels, this analysis aims to uncover actionable insights that can drive strategic decisions.

The project begins with meticulous data collection and preprocessing to ensure data accuracy and completeness. Through exploratory data analysis (EDA), the study examines patterns in sales volumes over time, identifies seasonal fluctuations, and assesses the performance of different product categories. Key metrics such as inventory turnover and stock-to-sales ratios will be calculated to provide quantitative benchmarks for inventory efficiency.

Furthermore, the analysis will delve into identifying low-stock items that require attention and propose strategies to optimize inventory replenishment processes. Visualizations created from the data will offer clear insights into sales dynamics and inventory metrics, facilitating effective communication of findings.

Ultimately, the findings of this analysis will be distilled into actionable recommendations aimed at optimizing inventory levels, streamlining replenishment processes, and improving overall operational efficiency. By implementing these recommendations, businesses can expect to enhance their inventory management strategies, reduce costs, and better meet the dynamic demands of their markets, ultimately driving growth and profitability.

Purpose Statement

This project aims to revolutionize inventory management practices through comprehensive analysis of a sales dataset using advanced data analytics techniques. By delving into historical sales data, the project seeks to uncover intricate sales trends, seasonal variations, and product performance metrics that will inform strategic decisions. Key objectives include optimizing inventory levels by calculating essential metrics like inventory turnover and stock-to-sales ratios, identifying top-performing products and categories, and pinpointing low-stock items needing attention. The project will innovate through:

i. **Advanced Analytics Techniques**: Utilizing sophisticated algorithms and methodologies to extract deeper insights from the dataset, enhancing the granularity and accuracy of inventory management strategies.
ii. **Interactive Visualization Tools**: Employing interactive visualization tools such as Plotly and Tableau to create dynamic visual representations of sales trends and inventory metrics, facilitating intuitive understanding and actionable decision-making.
iii. **Actionable Recommendations**: Providing actionable recommendations tailored to improve operational efficiency, reduce carrying costs, and optimize stock replenishment processes based on data-driven insights.

By implementing these novel approaches, this project aims to empower businesses to achieve optimal inventory management, thereby enhancing competitiveness and customer satisfaction in a dynamic market environment.

Project Description

This project entails analyzing a sales dataset to uncover insights that optimize inventory management. By examining sales trends, identifying top-selling products, and evaluating inventory levels, the analysis aims to enhance operational efficiency. Key metrics such as inventory turnover and stock-to-sales ratios will guide recommendations for improving stock replenishment strategies. Visualizations will illustrate findings, aiding in decision-making. Ultimately, the project aims to reduce carrying costs, minimize stockouts, and meet customer demand effectively through informed inventory management practices.

Dataset Analysis

The dataset consists of two parts, historical and active SKUs. The historical dataset contains historical sales data for SKUs. The active dataset contains SKUs that are in the active inventory. Although the dataset was uploaded to Kaggle years ago, we will treat the active data as the current inventory. Our eventual goal is to determine a probability of sale for active SKUs after analyzing the historical SKU sales data. Many of the historical SKUs are also present in the active inventory. Each row in the datasets represent a distinct SKU. Below is breakdown of all relevant variables in the datasets.

SKU_number: This is the unique identifier for each SKU.

SoldFlag: Category, 1 = sold in past 6 months, 0 = Not sold in the past 6 months. When SoldFlagis zero, SoldCount always zero. SoldFlag can be Null. When it is Null, SoldCount will be NA too. Both SoldFlag and SoldCount are Null in the active dataset as these SKUs have not been sold and this is the target variable we are trying to predict.

SoldCount: Count how many items were sold for this item in the last 6 months.

MarketingType: Two categories of how we market the product ("D" and "S"), each type should be considered independently.

ReleaseNumber: This will count the number of items that are going to be released in the future.

New_Release_Flag: Any product that has had a future release (i.e., Release Number > 1). When New_Release_Flag is zero, ReleaseNumber will be zero also.

StrengthFactor: Represents the performance of the items depending on ItemCount, SoldFlag and Price.

ReleaseYear: When the SKU was originally released (year).

ItemCount: Represents the number of items in the inventory per SKU. Many values in ItemCount are 0.

PriceReg: PriceReg we assume stands for regular price of the SKU.

LowNetPrice: Another price metric per SKU, we assume stands for the net price to sell SKU after expenses.

LowUserPrice: Another price metric per SKU. It is worth noting that mdneuzerling's analysis treats LowUserPrice as the primary SKU price since it has the fewest count of 0-dollar values. In general, we will follow this guideline, however, we will also look at how the other two price variables (PriceReg & LowNetPrice) vary across SKU's and SKU sales performance.

Methodology

The below structured methodology ensures a systematic approach to analyzing the sales dataset, deriving actionable insights, and proposing tailored recommendations to optimize inventory management practices effectively.

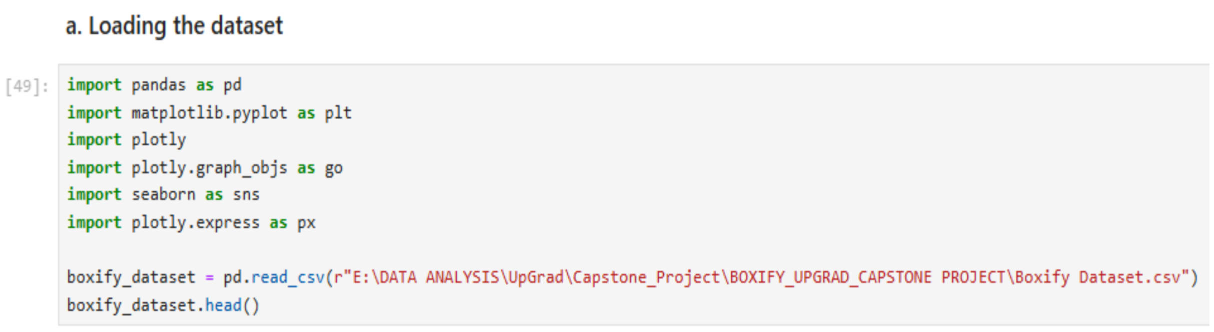
1. **Data Collection**:
   a. Obtain the sales dataset from the provided source or database.
   b. Ensure completeness and accuracy of the dataset by validating against primary records.
2. **Data Preprocessing**:
   a. Clean the data to handle missing values, duplicates, and inconsistencies.
   b. Standardize formats (e.g., dates, currencies) for consistency in analysis.
3. **Exploratory Data Analysis (EDA)**:
   a. **Sales Trends Analysis**:
      i. Explore sales patterns over different time intervals (daily, weekly, monthly).
      ii. Identify seasonal variations and trends using time series analysis techniques.

   b. **Product Performance Analysis**:
      i. Determine top-selling products and categories based on revenue and quantity sold.
      ii. Calculate metrics such as average selling price and contribution margin.
   c. **Inventory Evaluation**:
      i. Assess current stock levels and identify low-stock items.
      ii. Calculate inventory turnover rate and stock-to-sales ratios to gauge efficiency.
4. **Key Performance Indicators (KPIs) Calculation**:
   a. Compute KPIs such as:
      i. Inventory turnover rate: Measure how often inventory is sold and replaced over a specific period.
      ii. Stock-to-sales ratio: Assess the relationship between inventory levels and sales volume.
      iii. Reorder points: Determine optimal inventory levels that trigger reorder actions based on demand forecasts.
5. **Data Visualization**:
   a. Create visual representations of insights using tools like Matplotlib and Plotly:
      i. Line charts to visualize sales trends over time.
      ii. Bar plots to compare performance across different product categories.
      iii. Interactive dashboards (if using tools like Tableau or Power BI) for dynamic exploration of data.
6. **Insights Generation**:
   a. Summarize findings from EDA and KPI calculations.
   b. Identify actionable insights and recommendations for inventory management improvement.
7. **Recommendations**:
   a. Provide actionable strategies for optimizing inventory levels, reducing carrying costs, and improving stock availability.
   b. Include suggestions for implementing automated inventory replenishment systems or adjusting stock allocation based on demand patterns.

Exploratory Data Analysis and Insights

1. Data Collection

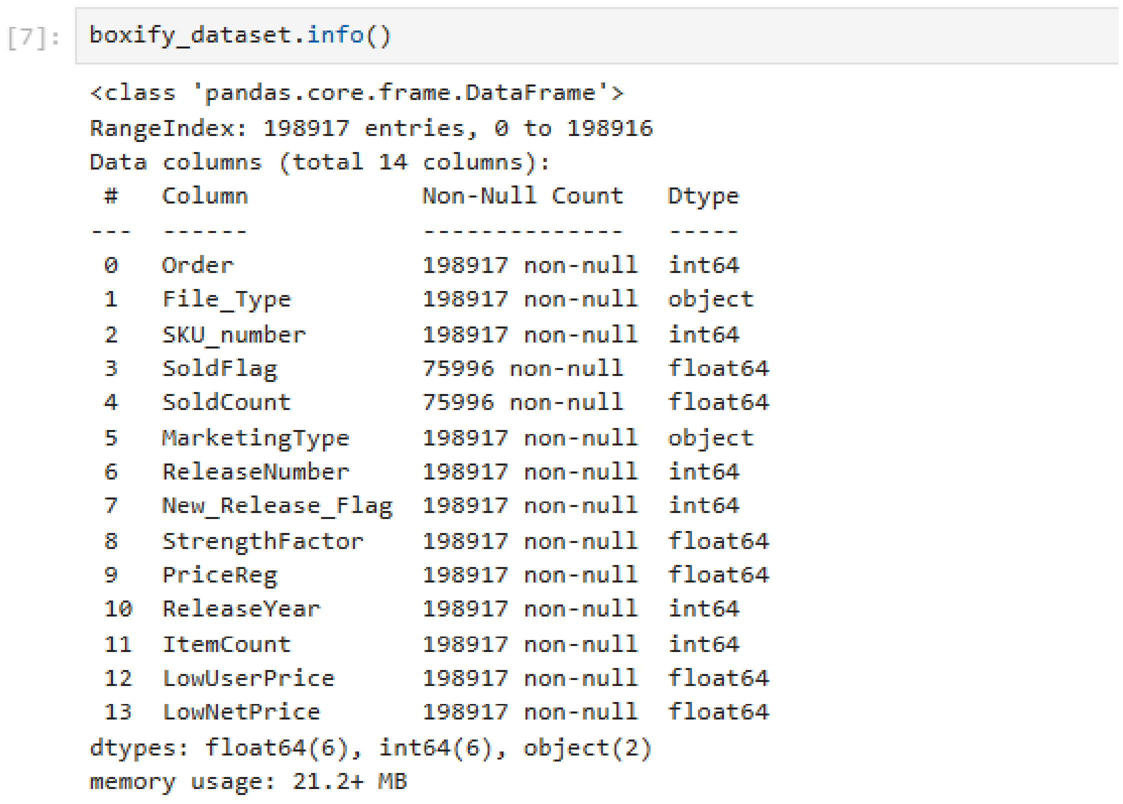
The dataset is loaded using Panda's library and the required libraries have been imported.



a. Loading the dataset

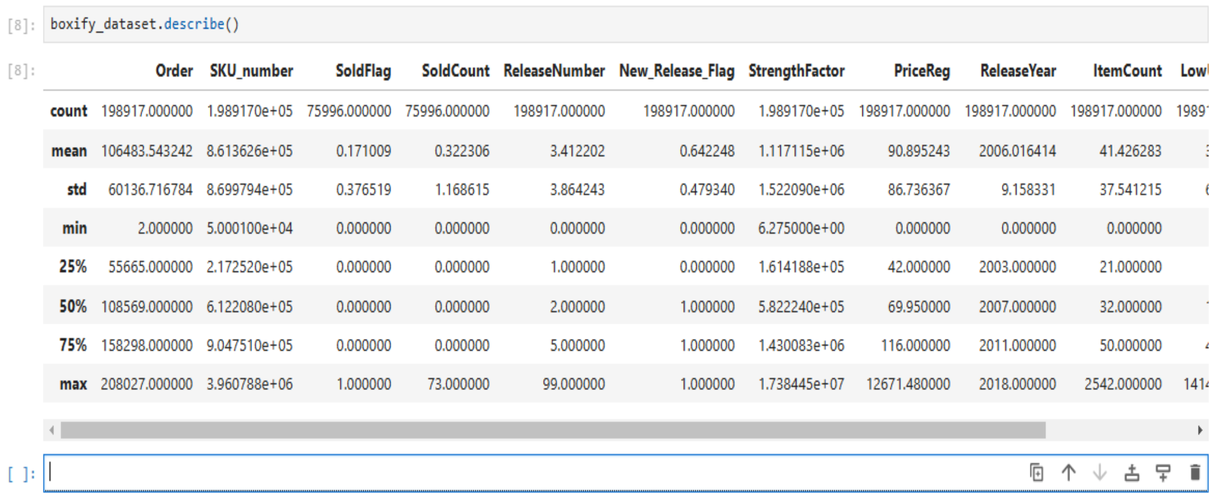
```
[49]: import pandas as pd
      import matplotlib.pyplot as plt
      import plotly
      import plotly.graph_objs as go
      import seaborn as sns
      import plotly.express as px

      boxify_dataset = pd.read_csv(r"E:\DATA ANALYSIS\UpGrad\Capstone_Project\BOXIFY_UPGRAD_CAPSTONE PROJECT\Boxify Dataset.csv")
      boxify_dataset.head()
```

The summary of the data frame's structure and information is analyzed using 'info()' method.



```
[7]: boxify_dataset.info()

<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           75996 non-null   float64
 4   SoldCount          75996 non-null   float64
 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(6), int64(6), object(2)
memory usage: 21.2+ MB
```

The statistical summary of dataset is analyzed by 'describe ()' method.



```
[8]: boxify_dataset.describe()
```

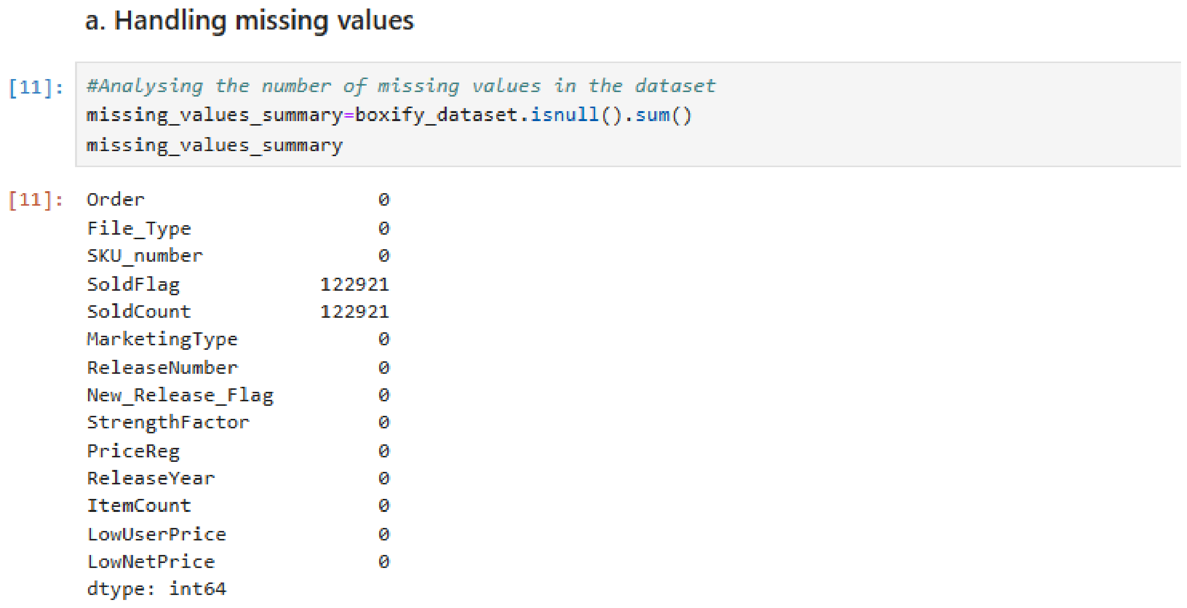
[8]:

	Order	SKU_number	SoldFlag	SoldCount	ReleaseNumber	New_Release_Flag	StrengthFactor	PriceReg	ReleaseYear	ItemCount	Low
count	198917.000000	1.989170e+05	75996.000000	75996.000000	198917.000000	198917.000000	1.989170e+05	198917.000000	198917.000000	198917.000000	1989
mean	106483.543242	8.613626e+05	0.171009	0.322306	3.412202	0.642248	1.117115e+06	90.895243	2006.016414	41.426283	3
std	60136.716784	8.699794e+05	0.376519	1.168615	3.864243	0.479340	1.522090e+06	86.736367	9.158331	37.541215	6
min	2.000000	5.000100e+04	0.000000	0.000000	0.000000	0.000000	6.275000e+00	0.000000	0.000000	0.000000	
25%	55665.000000	2.172520e+05	0.000000	0.000000	1.000000	0.000000	1.614188e+05	42.000000	2003.000000	21.000000	
50%	108569.000000	6.122080e+05	0.000000	0.000000	2.000000	1.000000	5.822240e+05	69.950000	2007.000000	32.000000	1
75%	158298.000000	9.047510e+05	0.000000	0.000000	5.000000	1.000000	1.430083e+06	116.000000	2011.000000	50.000000	4
max	208027.000000	3.960788e+06	1.000000	73.000000	99.000000	1.000000	1.738445e+07	12671.480000	2018.000000	2542.000000	1414

```
[ ]:
```

2. Data Preprocessing

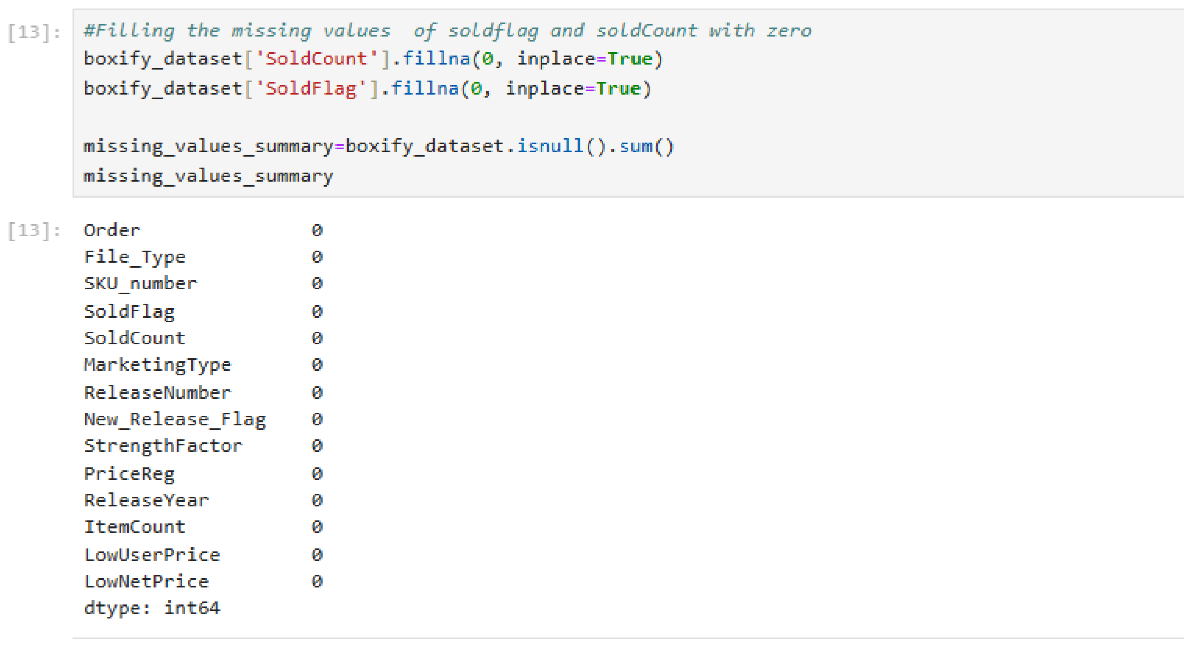
To handle the missing values in the dataset and to organize the format of dataset, we follow data pre-processing. From the summary of the data frame, it has been observed that sold count and sold flag columns have missing values. We will replace the missing values of both these columns with zero indicating no sales.



a. Handling missing values

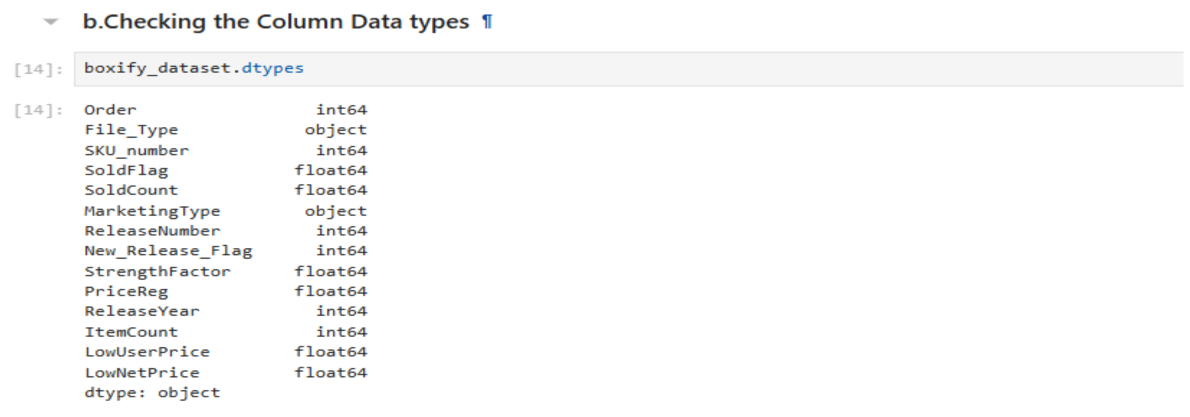
```
[11]: #Analysing the number of missing values in the dataset
      missing_values_summary=boxify_dataset.isnull().sum()
      missing_values_summary
```

```
[11]: Order                    0
      File_Type                0
      SKU_number               0
      SoldFlag            122921
      SoldCount           122921
      MarketingType            0
      ReleaseNumber            0
      New_Release_Flag         0
      StrengthFactor           0
      PriceReg                 0
      ReleaseYear              0
      ItemCount                0
      LowUserPrice             0
      LowNetPrice              0
      dtype: int64
```



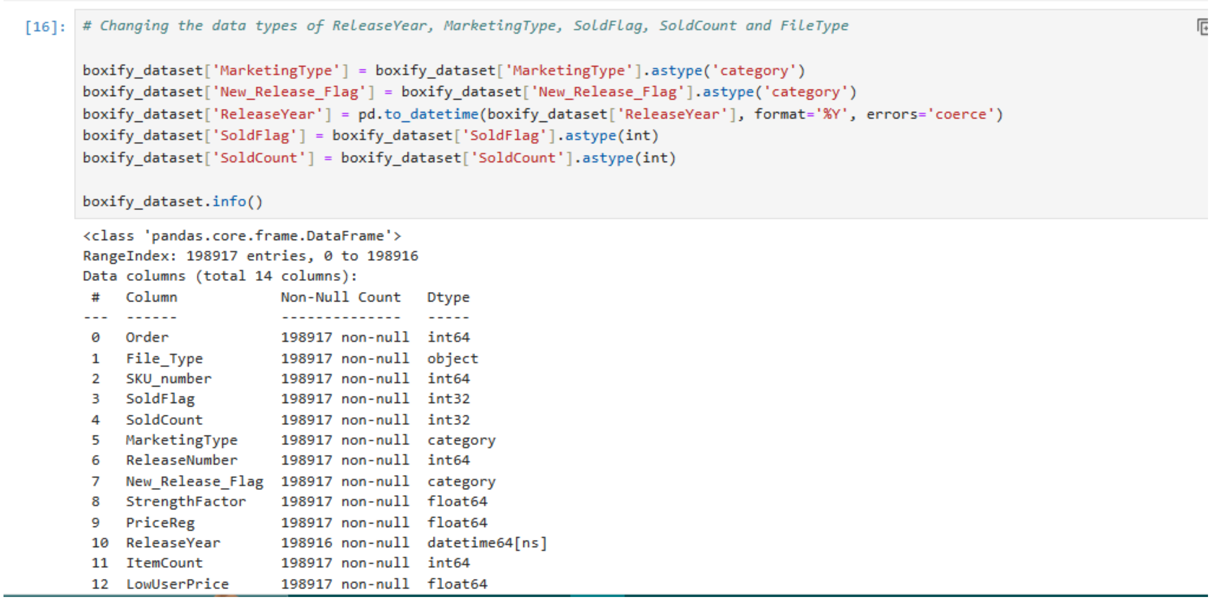
```
[13]: #Filling the missing values  of soldflag and soldCount with zero
      boxify_dataset['SoldCount'].fillna(0, inplace=True)
      boxify_dataset['SoldFlag'].fillna(0, inplace=True)

      missing_values_summary=boxify_dataset.isnull().sum()
      missing_values_summary

[13]: 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
```

To check whether the data types of each Column is correct, we need to verify it. And hence we will use 'dtype' object. The data type of ReleaseNumber, SoldFlag, SoldCount, MarketingType and New_release_flag have been changed for correct analysis.



b.Checking the Column Data types ¶

```
[14]: boxify_dataset.dtypes

[14]: Order                 int64
      File_Type            object
      SKU_number            int64
      SoldFlag            float64
      SoldCount           float64
      MarketingType        object
      ReleaseNumber         int64
      New_Release_Flag      int64
      StrengthFactor      float64
      PriceReg            float64
      ReleaseYear           int64
      ItemCount             int64
      LowUserPrice        float64
      LowNetPrice         float64
      dtype: object
```



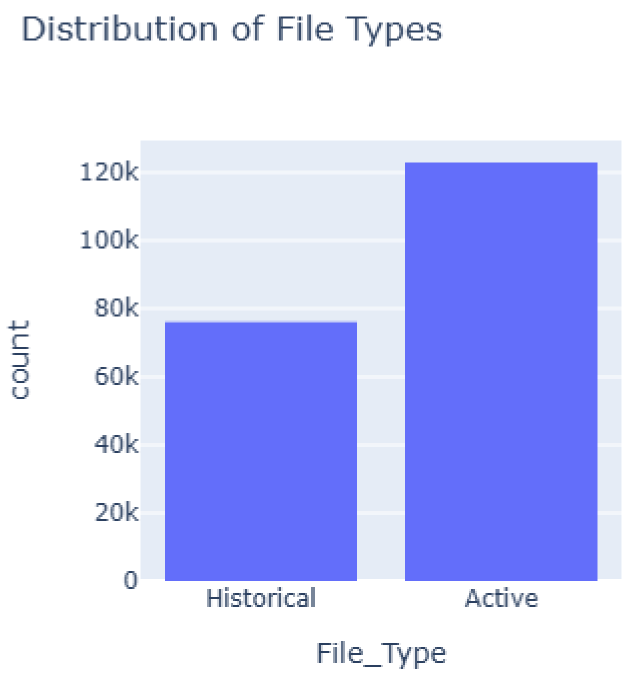
```
[16]: # Changing the data types of ReleaseYear, MarketingType, SoldFlag, SoldCount and FileType

      boxify_dataset['MarketingType'] = boxify_dataset['MarketingType'].astype('category')
      boxify_dataset['New_Release_Flag'] = boxify_dataset['New_Release_Flag'].astype('category')
      boxify_dataset['ReleaseYear'] = pd.to_datetime(boxify_dataset['ReleaseYear'], format='%Y', errors='coerce')
      boxify_dataset['SoldFlag'] = boxify_dataset['SoldFlag'].astype(int)
      boxify_dataset['SoldCount'] = boxify_dataset['SoldCount'].astype(int)

      boxify_dataset.info()

      <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  int32
       4   SoldCount         198917 non-null  int32
       5   MarketingType     198917 non-null  category
       6   ReleaseNumber     198917 non-null  int64
       7   New_Release_Flag  198917 non-null  category
       8   StrengthFactor    198917 non-null  float64
       9   PriceReg          198917 non-null  float64
       10  ReleaseYear       198916 non-null  datetime64[ns]
       11  ItemCount         198917 non-null  int64
       12  LowUserPrice      198917 non-null  float64
```

3. Exploratory Data Analysis (EDA)

(A) Frequency Distribution of File Types and Marketing Types

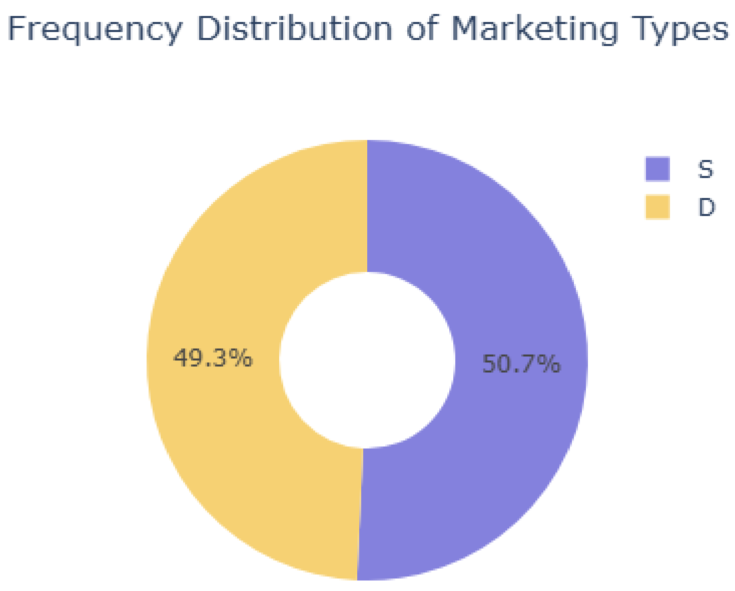
Analysis: Initially we need to find the frequency of File_types and Marketing_Types using Plotly.

```
[36]: fig = px.histogram(boxify_dataset.File_Type, x="File_Type",title="Distribution of File Types")
      fig.update_layout(autosize=False,
          width=400,
          height=400)
      fig.show()
```



Frequency distribution by Marketing Types

```
[39]: custom_colors = ['#8481DD', '#F6D173']  # Add more colors if you have more categories
      fig = px.pie(
          boxify_dataset,
          names='MarketingType',
          title='Frequency Distribution of Marketing Types',
          hole=0.4,  # Optional: makes it a donut chart for better readability
          color_discrete_sequence=custom_colors
      )
      fig.update_traces(textinfo='percent')

      fig.update_layout(
          autosize=False,
          width=400,
          height=400
      )

      fig.show()
```

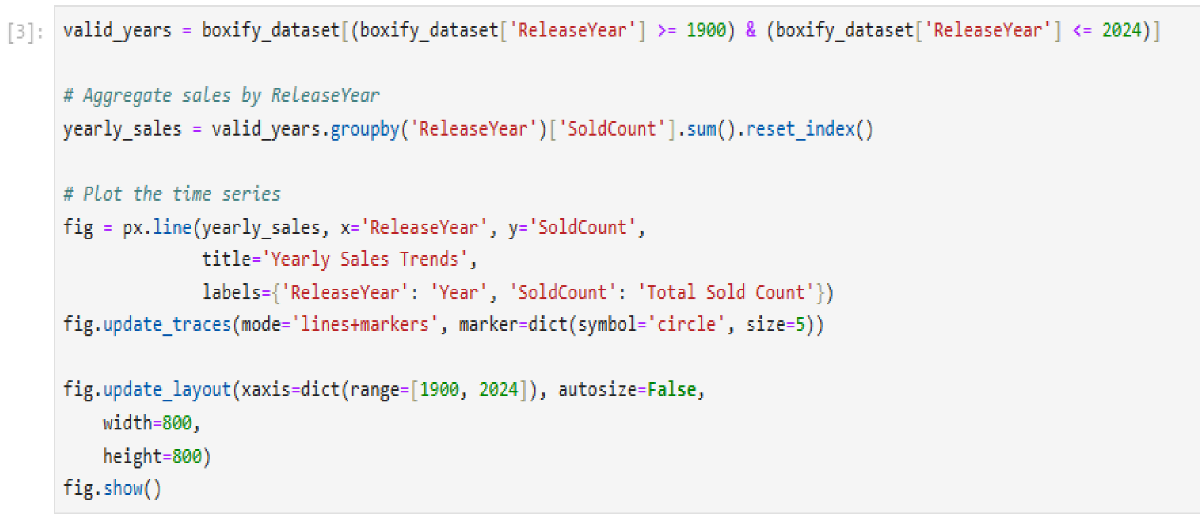


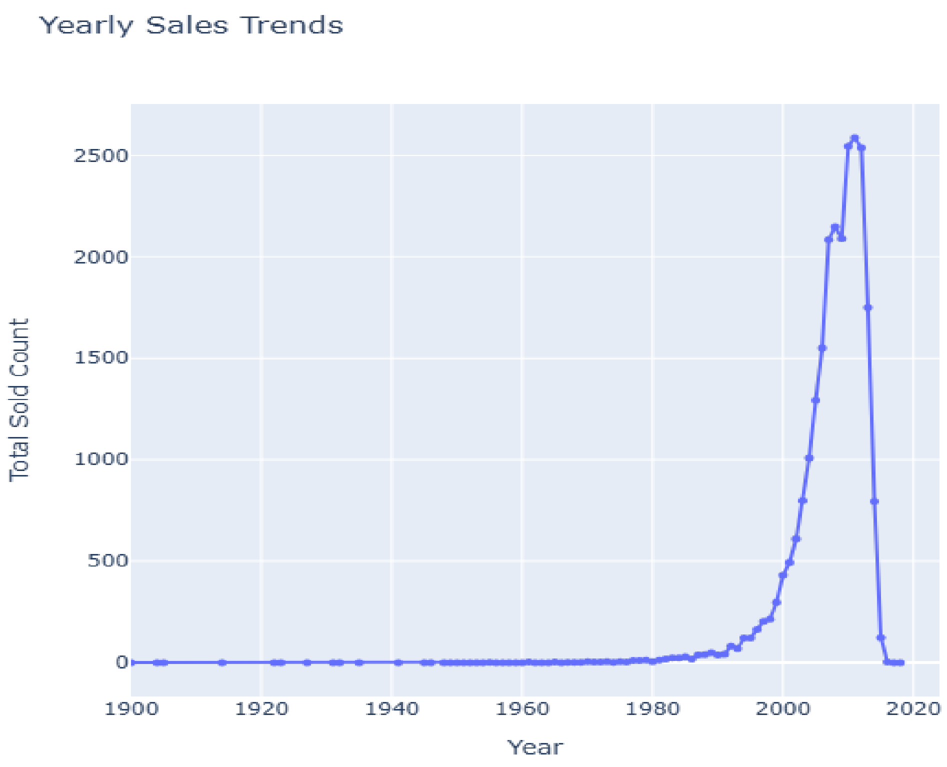
Insights:

- Active files constitute a larger portion of the total dataset.
- The higher number of Active files might indicate a need for more resources to manage and maintain current data.
- The near-equal split of Marketing Types suggests that both marketing strategies (S and D) are being used almost equally to reach different segments of the market or to diversify marketing efforts.

(B) Yearly sales Trend

Analysis: To view the sales trend over the time, we need to first group the SoldCount according to ReleaseYear.


```
[3]: valid_years = boxify_dataset[(boxify_dataset['ReleaseYear'] >= 1900) & (boxify_dataset['ReleaseYear'] <= 2024)]

     # Aggregate sales by ReleaseYear
     yearly_sales = valid_years.groupby('ReleaseYear')['SoldCount'].sum().reset_index()

     # Plot the time series
     fig = px.line(yearly_sales, x='ReleaseYear', y='SoldCount',
                   title='Yearly Sales Trends',
                   labels={'ReleaseYear': 'Year', 'SoldCount': 'Total Sold Count'})
     fig.update_traces(mode='lines+markers', marker=dict(symbol='circle', size=5))

     fig.update_layout(xaxis=dict(range=[1900, 2024]), autosize=False,
         width=800,
         height=800)
     fig.show()
```



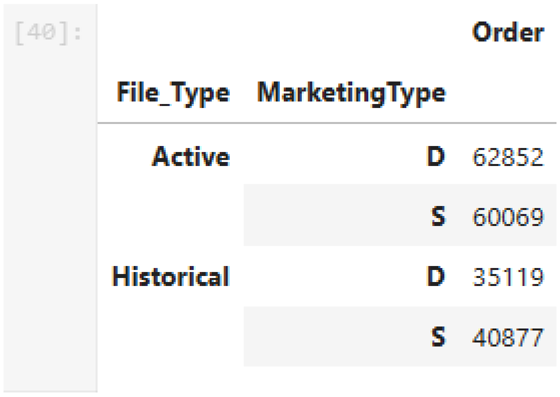
Insights:

- There is a significant increase in the total sold count starting around the 1980s, peaking in the late 1990s to early 2000s.
- After reaching the peak, there is a sharp decline in the total sold count post-2000s.

(C) Order Count of Active and Historical Products

Analysis: Sales By Marketing Types- To see how many SKUs with marketing types "D" and "S" have been sold in the last six months.

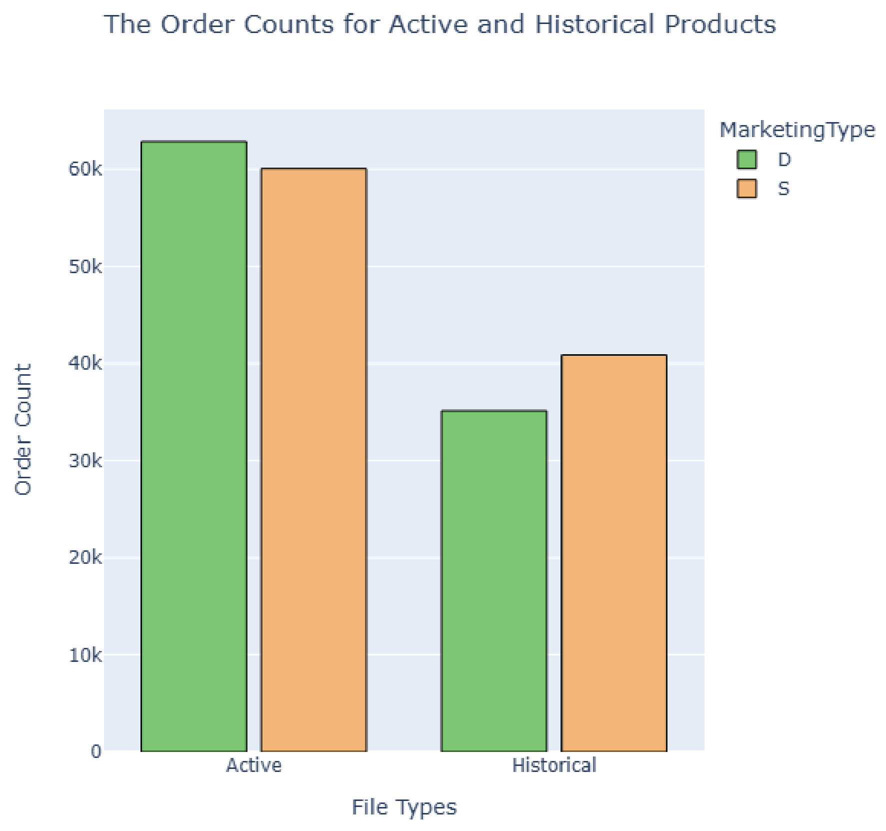
```
[40]: market_compare = boxify_dataset.groupby(['File_Type','MarketingType'])[['Order']].count()
      market_compare
```



[40]:

File_Type	MarketingType	Order
Active	D	62852
	S	60069
Historical	D	35119
	S	40877

```python
market_com = market_compare.reset_index()
color_map = {
    'D': '#7CC674',
    'S': '#F4B678'
}
fig = px.bar(
    market_com,
    x='File_Type',
    y='Order',
    color='MarketingType',
    title='Top Selling Products',
    labels={'File_Type': 'File Types', 'Order': 'Order Count'},
    barmode='group',
    color_discrete_map=color_map
)

fig.update_traces(marker_line_color='black', marker_line_width=1, width=0.35)

fig.update_layout(title_text='The Order Counts for Active and Historical Products', title_x=0.5, autosize=False, width=600, height=600)
fig.show()
```



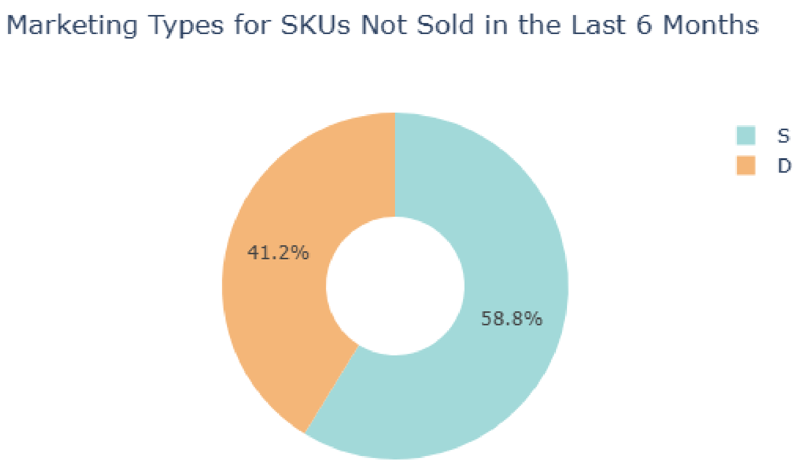
Insights:

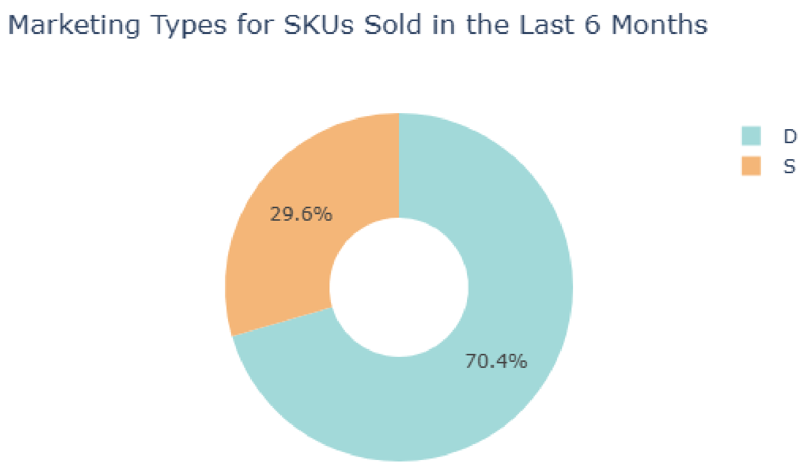
- For historical products, the market preference for type S products is higher than type D, on the contrary, type D take a larger proportion in active products rather than type S.
- Next, we explore the sold count and not sold count for type D and type S products in the last 6 months, in order to find the reason why the proportion of these two type goods are different in older and current inventories.

(D) Marketing Type proportion for Sold/Not sold SKUs

Analysis: Marketing type counts for SKUs that have and have not been sold in the last 6 months

```
marketing_counts = boxify_dataset.groupby(['SoldFlag', 'MarketingType']).size().reset_index(name='count')
custom_colors = ['#A2D9D9', '#F4B678']
fig0 = px.pie(marketing_counts[marketing_counts['SoldFlag'] == 0],
              names='MarketingType', values='count',
              title='Marketing Types for SKUs Not Sold in the Last 6 Months', hole = 0.4,
             color_discrete_sequence=custom_colors)

# Filter data for SoldFlag = 1
fig1 = px.pie(marketing_counts[marketing_counts['SoldFlag'] == 1],
              names='MarketingType', values='count',
              title='Marketing Types for SKUs Sold in the Last 6 Months',hole = 0.4,
              color_discrete_sequence=custom_colors)
fig0.update_layout(
    autosize=False,
    width=550,
    height=400
)
fig1.update_layout(
    autosize=False,
    width=550,
    height=400
)
fig0.show()
fig1.show()
```





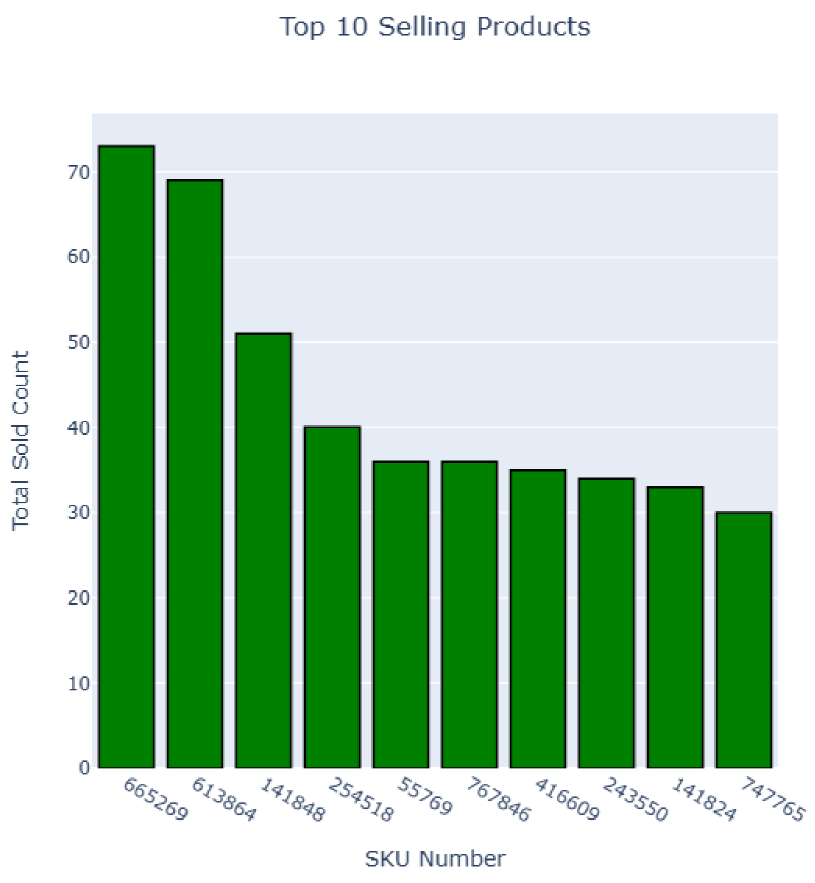
Insights:

- We already know that about 80% of historical SKUs have not been sold in the last six months, which we can clearly see in the larger bars where SoldFlag=0.
- However, within the ~20% of SKUs that have been sold in the last six months, we can see that SKUs with MarketingType "D" historically sold double than that of MarketingType "S".
- In other words, it looks like MarketingType "D" SKUs exhibit better sales performance. Let's take a look at how price varies across different marketing types.

(E) Top 10 selling SKUs

Analysis: Top selling products by SKU_number and SoldCount

```
top_selling_products = boxify_dataset.groupby('SKU_number')['SoldCount'].sum().reset_index()

# Sorting SoldCount in descending order and considering top 10 products
top_selling_products = top_selling_products.sort_values(by='SoldCount', ascending=False).head(10)

# Plotting the top-selling products
fig = px.bar(top_selling_products, x='SKU_number', y='SoldCount',
             title='Top Selling Products',
             labels={'SKU_number': 'SKU Number', 'SoldCount': 'Total Sold Count'})

fig.update_traces(marker_color='green', marker_line_color='black', marker_line_width=1.5)
fig.update_layout(title_text='Top 10 Selling Products', title_x=0.5, autosize=False, width=600, height=600)
fig.show()
```

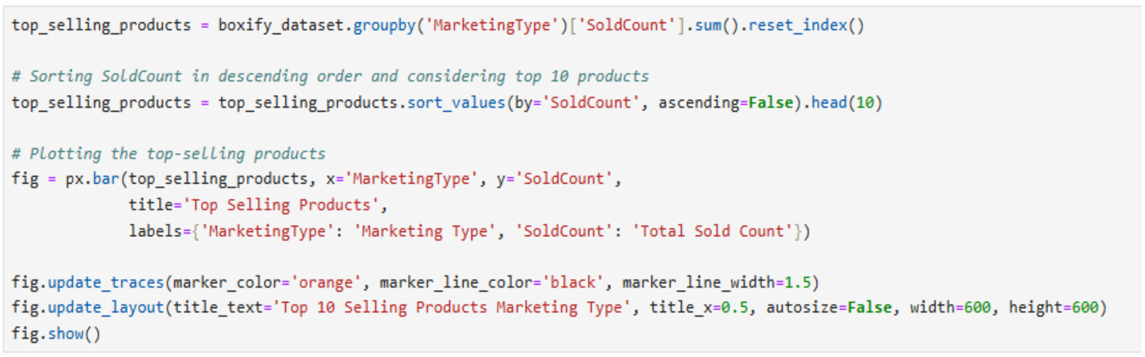


Insights:

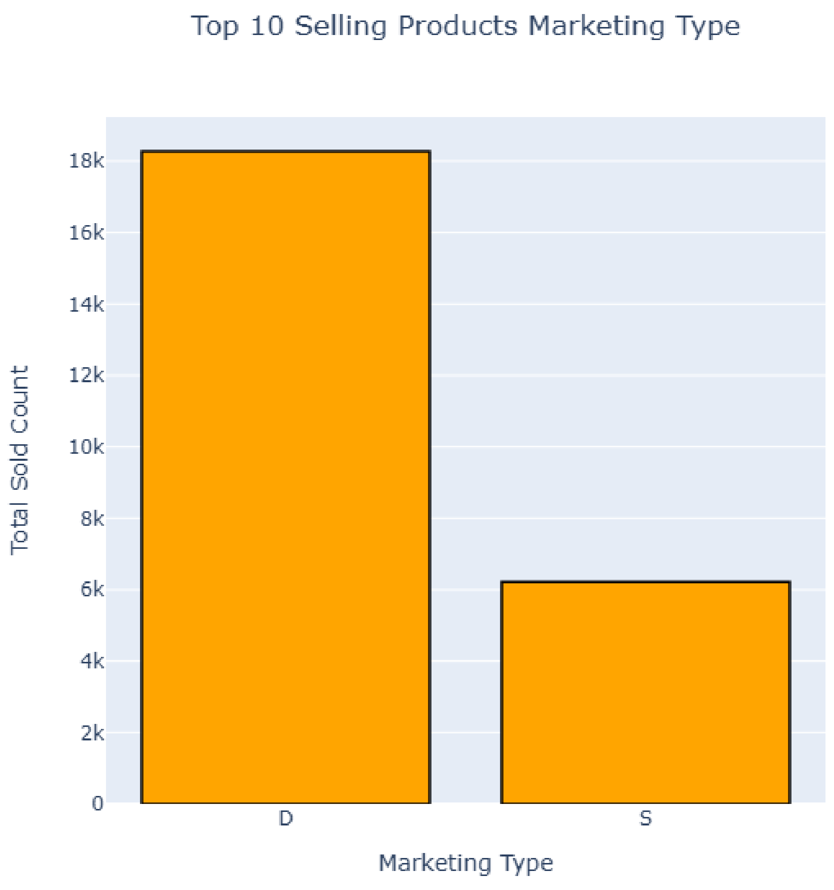
- Throughout the time period, the SKU_number ‘665269’has been sold the most with 73 counts.
- The top three SKUs (665269, 613864, and 141848) have sold counts above 50. These products are the most popular among customers.
- While there is a clear leader, the top 10 SKUs all have a sold count above 20, indicating a fairly consistent demand for these items.

(F) Top 10 selling products by Marketing Types

Analysis: We need to find the top selling products by marketing types also.



```
top_selling_products = boxify_dataset.groupby('MarketingType')['SoldCount'].sum().reset_index()

# Sorting SoldCount in descending order and considering top 10 products
top_selling_products = top_selling_products.sort_values(by='SoldCount', ascending=False).head(10)

# Plotting the top-selling products
fig = px.bar(top_selling_products, x='MarketingType', y='SoldCount',
             title='Top Selling Products',
             labels={'MarketingType': 'Marketing Type', 'SoldCount': 'Total Sold Count'})

fig.update_traces(marker_color='orange', marker_line_color='black', marker_line_width=1.5)
fig.update_layout(title_text='Top 10 Selling Products Marketing Type', title_x=0.5, autosize=False, width=600, height=600)
fig.show()
```



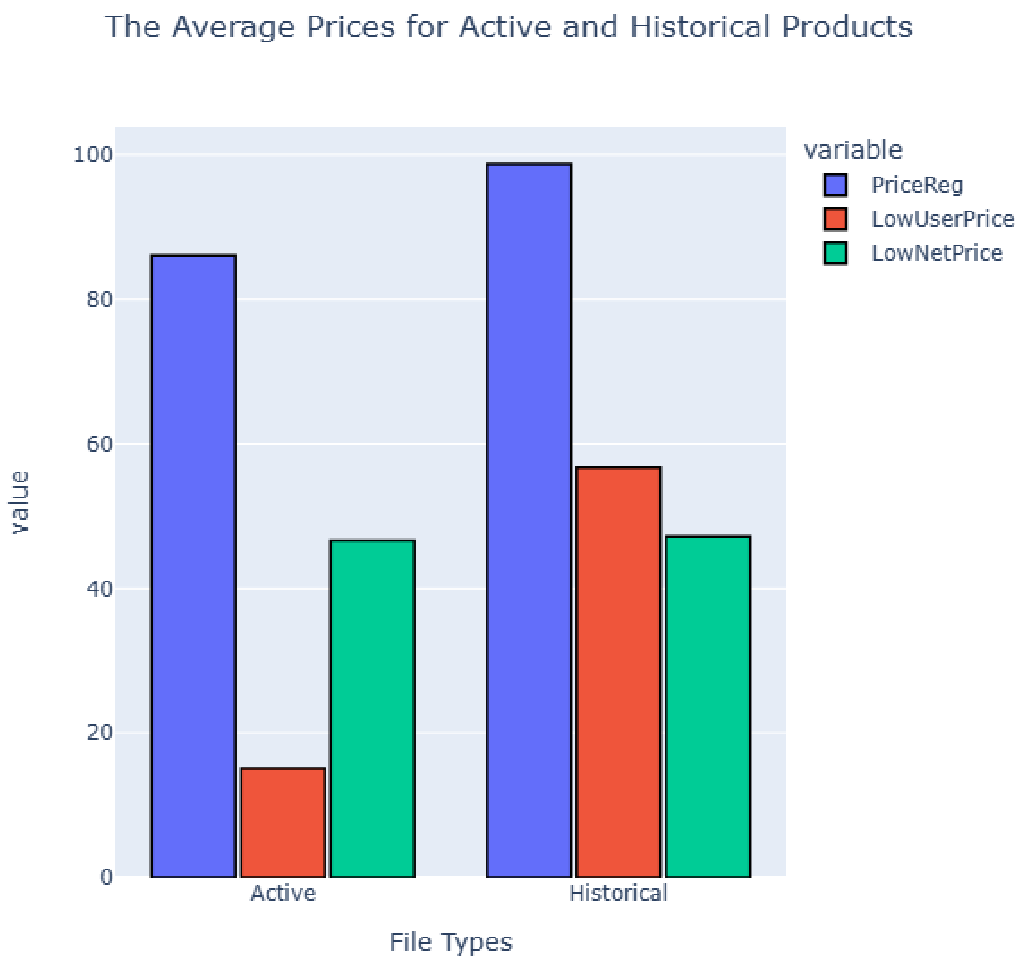
Insights:

- Marketing Type "D" (Denoted as "D") has a significantly higher total sold count compared to Marketing Type "S" (Denoted as "S"). This indicates that products marketed under type "D" are more successful in terms of sales.
- The total sold count for Marketing Type "D" is above 18,000 units.
- The total sold count for Marketing Type "S" is around 6,000 units.

(G) The average prices for the historical and active products

Analysis: Comparison of Prices for the historical and active products

```
price_comparison = boxify_dataset.groupby('File_Type')[['PriceReg', 'LowUserPrice', 'LowNetPrice']].mean().reset_index()
fig = px.bar(price_comparison, x='File_Type', y=['PriceReg', 'LowUserPrice', 'LowNetPrice'],
             title='Top Selling Products',
             labels={'File_Type': 'File Types'}, barmode='group')

fig.update_traces(marker_line_color='black', marker_line_width=1.5, width=0.25)
fig.update_layout(title_text='Top 10 Selling Products', title_x=0.5, autosize=False, width=600, height=600)
fig.show()
```



Insights:

- The average PriceReg for active products is around 80. The average PriceReg for historical products is slightly higher, close to 100. Hence, Historical products tend to have a higher regular price compared to active products.
- Active products are offered at much lower prices to users compared to historical products.
- Historical products have a slightly higher net price compared to active products, but the difference is not as pronounced as with the LowUserPrice.

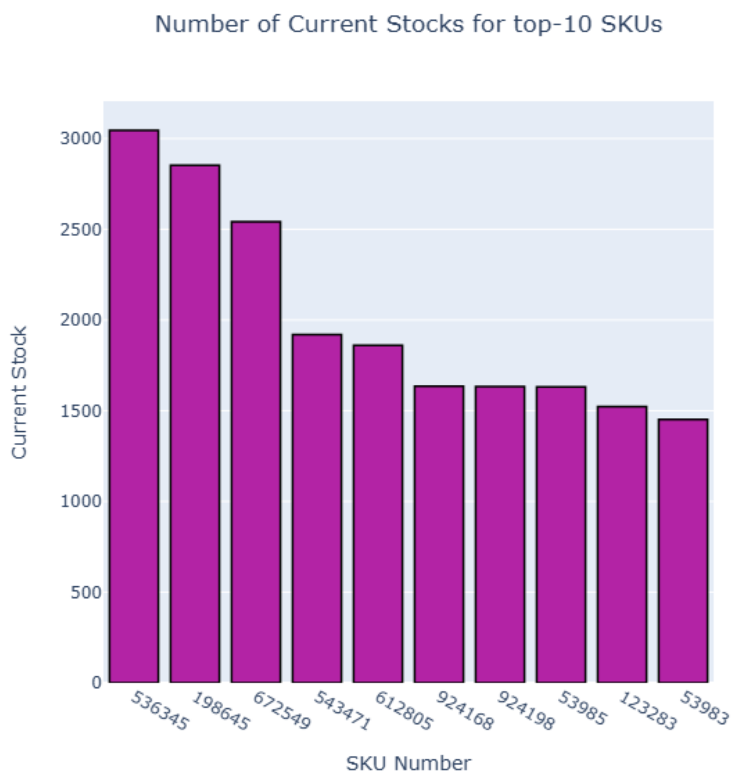
(H)Count of Stocks

Analysis: Calculation of Stock Levels

```
current_stock_items = boxify_dataset.groupby('SKU_number')['ItemCount'].sum().reset_index()
current_stock_items

#To identify the stock_count below defined threshold value
low_stock_threshold = 10
low_stock_items = boxify_dataset[boxify_dataset['ItemCount'] < low_stock_threshold]
low_stocks_count = low_stock_items.shape[0]
low_stocks_count_percentage = (low_stocks_count/boxify_dataset.shape[0])*100
print(f"Low stock item count: {low_stocks_count}")
print(f"Low stock item count percentage: {low_stocks_count_percentage:.2f}%")
```

```
Low stock item count: 3379
Low stock item count percentage: 1.70%
```

```
#Plotting the number of current stocks
initial_stock = boxify_dataset.groupby('SKU_number')['ItemCount'].sum().reset_index()

# Calculate the total sold count for each SKU_number
total_sold = boxify_dataset.groupby('SKU_number')['SoldCount'].sum().reset_index()

# Merge initial stock and total sold count
stock_data = initial_stock.merge(total_sold, on='SKU_number', how='left')

stock_data['CurrentStock'] = stock_data['ItemCount'] - stock_data['SoldCount']
stock_data = stock_data.sort_values(by='CurrentStock', ascending=False).head(10)
fig = px.bar(stock_data, x='SKU_number', y='CurrentStock', title='Current Stock by SKU Number',
             labels={'SKU_number': 'SKU Number', 'CurrentStock': 'Current Stock'})

# Show the plot
fig.update_traces(marker_color='#B323A5', marker_line_color='black', marker_line_width=1.5)
fig.update_layout(title_text='Number of Current Stocks for top-10 SKUs', title_x=0.5, autosize=False, width=600, height=600)
fig.show()
```



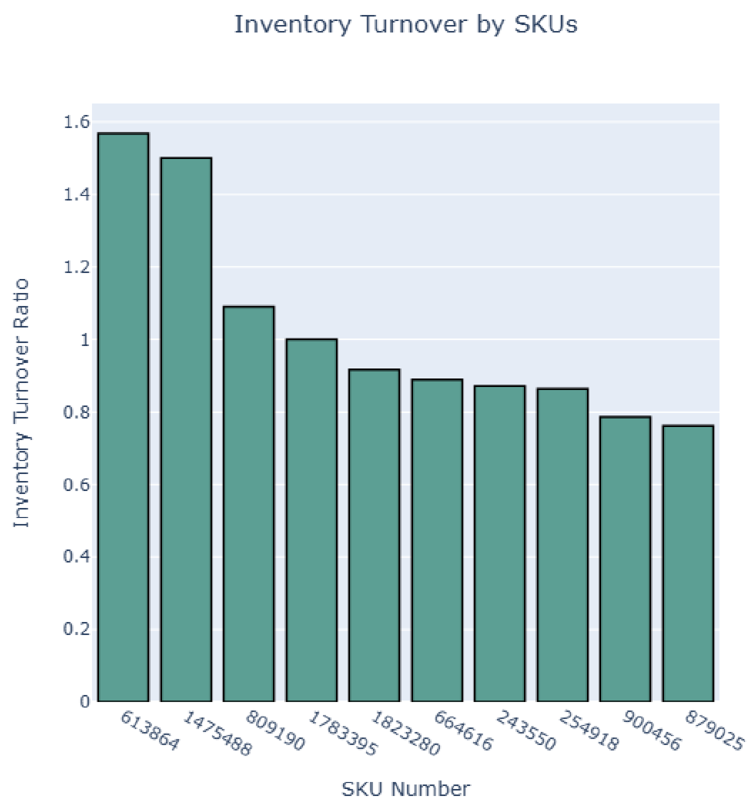
Insights:

- The script calculates the total number of items in stock for each SKU (Stock Keeping Unit) by grouping the dataset by SKU_number and summing the ItemCount. It then identifies SKUs with low stock by comparing the ItemCount to a predefined threshold value (10 in this case).
- Although the percentage of SKUs with low stock is relatively small (1.70%), the absolute count (3379 SKUs) indicates a significant number of items that may need restocking.
- The top 10 SKUs have relatively high stock levels, indicating a concentration of inventory in these items. This could suggest that these SKUs are either highly popular or have a slower turnover rate, necessitating higher stock levels.

(I) Inventory Turnover

Analysis: Calculating Inventory turn-over

```
boxify_dataset['InventoryTurnover'] = boxify_dataset['SoldCount'] / boxify_dataset['ItemCount']

boxify_dataset.replace([float('inf'), -float('inf')], float('nan'), inplace=True)
boxify_dataset.dropna(subset=['InventoryTurnover'], inplace=True)
boxify_turnOver_TopSKU=boxify_dataset.sort_values(by='InventoryTurnover', ascending=False).head(10)

# Plotting Inventory Turnover using Plotly
fig = px.bar(boxify_turnOver_TopSKU, x='SKU_number', y='InventoryTurnover',
             title='Inventory Turnover by SKUs',
             labels={'SKU_number': 'SKU Number', 'InventoryTurnover': 'Inventory Turnover Ratio'})

# Update the layout for better readability
fig.update_layout(xaxis={'categoryorder': 'total descending'}, showlegend=False,
                  title_x=0.5, autosize=False, width=600, height=600)
fig.update_traces(marker_color='#5c9f94', marker_line_color='black', marker_line_width=1.5)

# Show the plot
fig.show()
```



Insights:

- The inventory turnover ratio is a measure of how frequently inventory is sold and replaced over a specific period. It is calculated as the cost of goods sold divided by the average inventory.
- The SKU with the number `613864` has the highest inventory turnover ratio, just above 1.5. This indicates that this SKU is being sold and replenished more frequently compared to others.
- SKUs with high turnover ratios indicate strong sales performance and efficient inventory management. These SKUs may need to be prioritized for restocking to prevent stockouts.

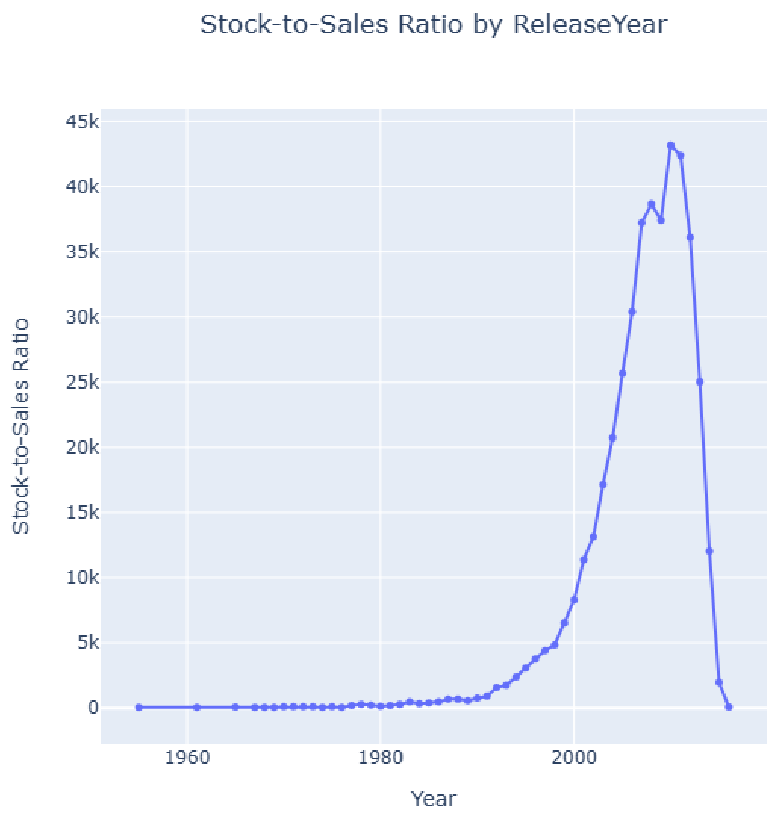
(J) Stock to Sales Ratio

Analysis: Stock to Sales Ratio

```python
# Calculate total inventory and total sales
total_inventory = boxify_dataset['ItemCount'].sum()
total_sales = boxify_dataset['SoldCount'].sum()

# Calculate stock-to-sales ratio
stock_to_sales_ratio = total_inventory / total_sales

print(f"Stock-to-Sales Ratio: {stock_to_sales_ratio:.2f}")
```

```
Stock-to-Sales Ratio: 33.60
```

```
boxify_dataset['stock_to_sales_ratio'] = boxify_dataset['ItemCount'].sum() / boxify_dataset['SoldCount'].sum()

stock_to_sales = boxify_dataset.groupby('ReleaseYear')['stock_to_sales_ratio'].sum().reset_index()
stock_to_sales

fig = px.line(stock_to_sales, x='ReleaseYear', y='stock_to_sales_ratio',
              title='Stock-to-Sales Ratio by ReleaseYear',
              labels={'ReleaseYear': 'Year', 'stock_to_sales_ratio': 'Stock-to-Sales Ratio'})

# Update the layout for better readability
fig.update_layout(xaxis= {'categoryorder': 'total descending'} , showlegend=False,
                  title_x=0.5, autosize=False, width=600, height=600)
fig.update_traces(mode='lines+markers', marker=dict(symbol='circle', size=5))
# Show the plot
fig.show()
```



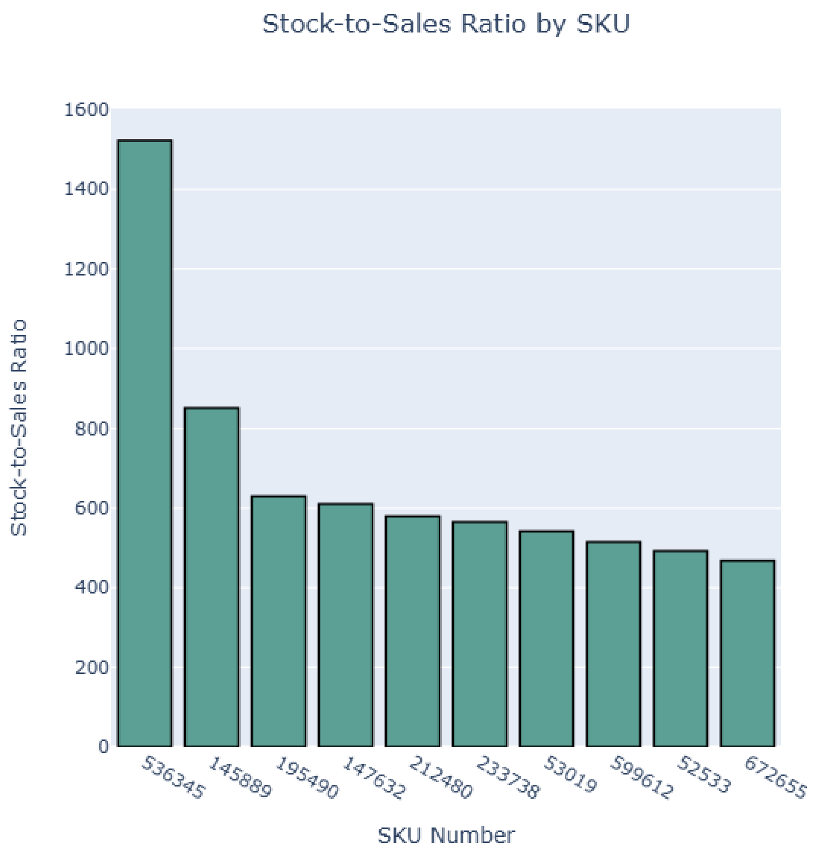
Insights:

- The Stock-to-Sales Ratio remains relatively stable and low from the 1950s to the early 1980s.
- Starting around the mid-1980s, the Stock-to-Sales Ratio begins to increase gradually, indicating an increase in stock levels relative to sales.
- he Stock-to-Sales Ratio experiences a sharp and rapid increase in the late 1990s, peaking around the year 2000. This could indicate a period of significant overstocking or a boom in production that outpaced sales.

(K) Stock to Sales Ration by top SKUs

Analysis: Stock to Sales Ration by top SKUs

```
boxify_dataset['StockToSalesRatio'] = boxify_dataset['ItemCount'] / boxify_dataset['SoldCount'].replace(0, pd.NA)

# Handle infinite or NaN values
boxify_dataset.replace([float('inf'), -float('inf')], float('nan'), inplace=True)
boxify_dataset.dropna(subset=['StockToSalesRatio'], inplace=True)
boxify_stocksRatio = boxify_dataset.sort_values(by='StockToSalesRatio', ascending=False).head(10)

# Plotting Stock-to-Sales Ratio using Plotly
fig = px.bar(boxify_stocksRatio, x='SKU_number', y='StockToSalesRatio',
             title='Stock-to-Sales Ratio by SKU',
             labels={'SKU_number': 'SKU Number', 'StockToSalesRatio': 'Stock-to-Sales Ratio'})

# Update the layout for better readability
fig.update_layout(xaxis={'categoryorder': 'total descending'}, showlegend=False, autosize=False, width=600, height=600)
fig.update_traces(marker_color='#c7728c', marker_line_color='black', marker_line_width=1.5)
# Show the plot
fig.show()
```



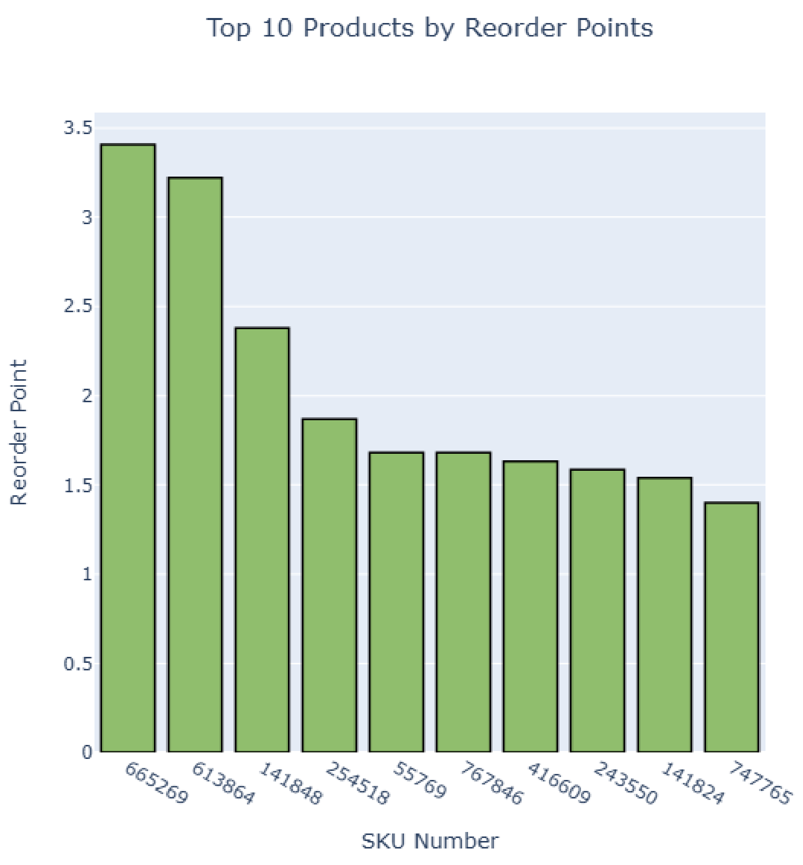
Insights:

- SKU 536345 has the highest tock to sales ratio indicating this product was overstocked due to demand among the customers.

(L) Reordering Points

Analysis: Calculating top 10 SKUs by Reordering Points

```
lead_time_days = 7
safety_stock_percentage = 0.2

# Calculate average daily demand
boxify_dataset['Average_Daily_Demand'] = boxify_dataset['SoldCount'] / 180  # the sales count is for six months

# Calculate demand during lead time
boxify_dataset['Demand_During_Lead_Time'] = boxify_dataset['Average_Daily_Demand'] * lead_time_days

# Calculate safety stock
boxify_dataset['Safety_Stock'] = boxify_dataset['Demand_During_Lead_Time'] * safety_stock_percentage

# Calculate reorder points
boxify_dataset['Reorder_Point'] = boxify_dataset['Demand_During_Lead_Time'] + boxify_dataset['Safety_Stock']

# Get the top 10 products by reorder points
top_10_reorder_points = boxify_dataset.nlargest(10, 'Reorder_Point')
fig = px.bar(top_10_reorder_points, x='SKU_number', y='Reorder_Point',
             title='Top 10 Products by Reorder Points',
             labels={'SKU_number':'SKU Number', 'Reorder_Point':'Reorder Point'})
fig.update_layout(xaxis={'categoryorder': 'total descending'}, showlegend=False,
                  title_x=0.5, autosize=False, width=600, height=600)
fig.update_traces(marker_color='#90be6d', marker_line_color='black', marker_line_width=1.5)
fig.show()
```



Insights:

- The SKUs 665269 and 613864 have the highest reorder points, indicating they may have higher sales volume or longer lead times, necessitating larger safety stock. The graph shows the reorder points on the basis of average daily demand.

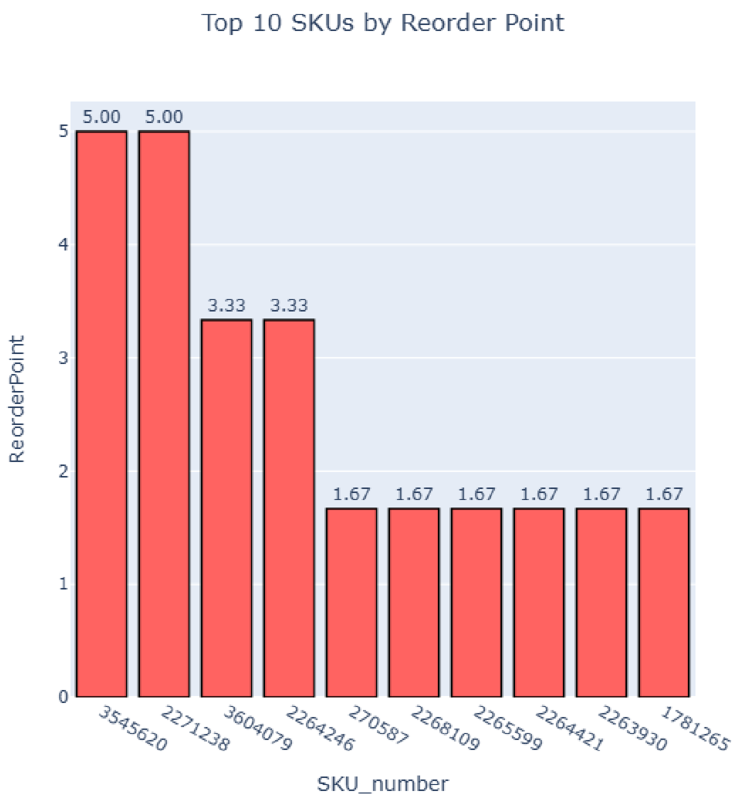
- SKUs like 141848 and 254518 have moderate reorder points, suggesting balanced demand and supply management.
- The SKUs 55769, 767846, 416609, 243350, 141824, and 747765 have lower reorder points, indicating either lower sales volumes, more frequent restocking, or shorter lead times.

(M) Top SKUs by Yearly Reorder Point

Analysis: Top SKUs by Yearly Reorder Point

```
# Calculate yearly sales for each SKU
yearly_sales = boxify_dataset.groupby(['SKU_number', 'ReleaseYear']).agg({'SoldCount': 'sum'}).reset_index()

# Calculate average yearly sales per SKU
average_sales = yearly_sales.groupby('SKU_number').agg({'SoldCount': 'mean'}).reset_index()
average_sales.rename(columns={'SoldCount': 'AvgYearlySales'}, inplace=True)

# Assume lead time in years (for simplicity, assume 1 year lead time)
lead_time = 1

# Calculate lead time demand
average_sales['LeadTimeDemand'] = average_sales['AvgYearlySales'] * lead_time

# Calculate standard deviation of sales per SKU to estimate variability
sales_variability = yearly_sales.groupby('SKU_number').agg({'SoldCount': 'std'}).reset_index()
sales_variability.rename(columns={'SoldCount': 'SalesStdDev'}, inplace=True)

reorder_data = pd.merge(average_sales, sales_variability, on='SKU_number')

# Assume a safety stock level (e.g., 1.65 for 95% service level)
safety_factor = 1.65
reorder_data['SafetyStock'] = safety_factor * reorder_data['SalesStdDev']

# Calculate reorder points
reorder_data['ReorderPoint'] = reorder_data['LeadTimeDemand'] + reorder_data['SafetyStock']
```

```
# Plotting reorder points for top 10 SKUs with highest reorder points
top_10_reorder_points = reorder_data.nlargest(10, 'ReorderPoint')

fig = px.bar(top_10_reorder_points, x='SKU_number', y='ReorderPoint', title='Top 10 SKUs by Yearly Reorder Point',
             text=top_10_reorder_points['ReorderPoint'].map(lambda x: f'{x:.2f}'))

fig.update_layout(xaxis={'categoryorder': 'total descending'}, showlegend=False,
                  title_x=0.5, autosize=False, width=600, height=600)
fig.update_traces(marker_color='#ff6361', marker_line_color='black', marker_line_width=1.5, textposition='outside' )
fig.show()

top_10_reorder_points = top_10_reorder_points.merge(yearly_sales[['SKU_number', 'ReleaseYear']], on='SKU_number', how='left')
top_10_reorder_points
```



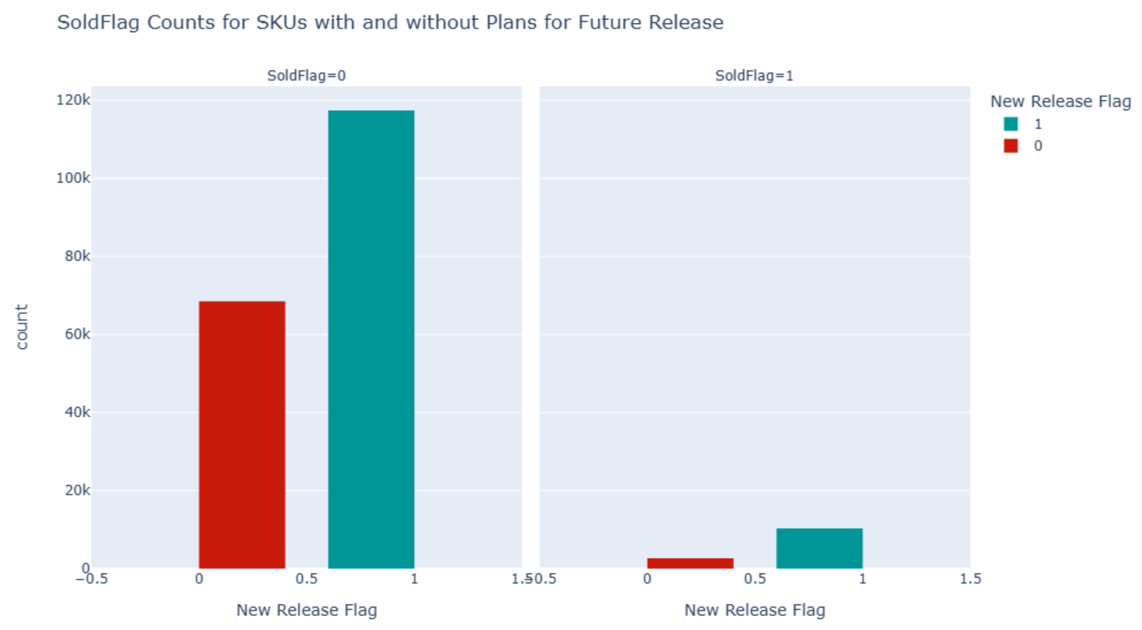
Insights:

- The graph shows the reorder points on the basis of yearly demand. SKU 3545620 has the maximum reorder point throughout the given period in the dataset.
- This indicates that this SKU number was most popular among the customers.

(N) SKUs with a planned future release

Analysis: Sold Flag counts for SKUs with and without future release plans

```
fig = px.histogram(boxify_dataset,
                   x="New_Release_Flag",
                   color="New_Release_Flag",
                   facet_col="SoldFlag",
                   title="SoldFlag Counts for SKUs with and without Plans for Future Release",
                   labels={"New_Release_Flag": "New Release Flag", "count": "Count"},
                   category_orders={"SoldFlag": [0, 1]},
                   color_discrete_sequence=["#009596", "#C9190B"])

# Update layout for better visualization
fig.update_layout(
    barmode='group',
    height=600,
    width=1000)
```



Insights:

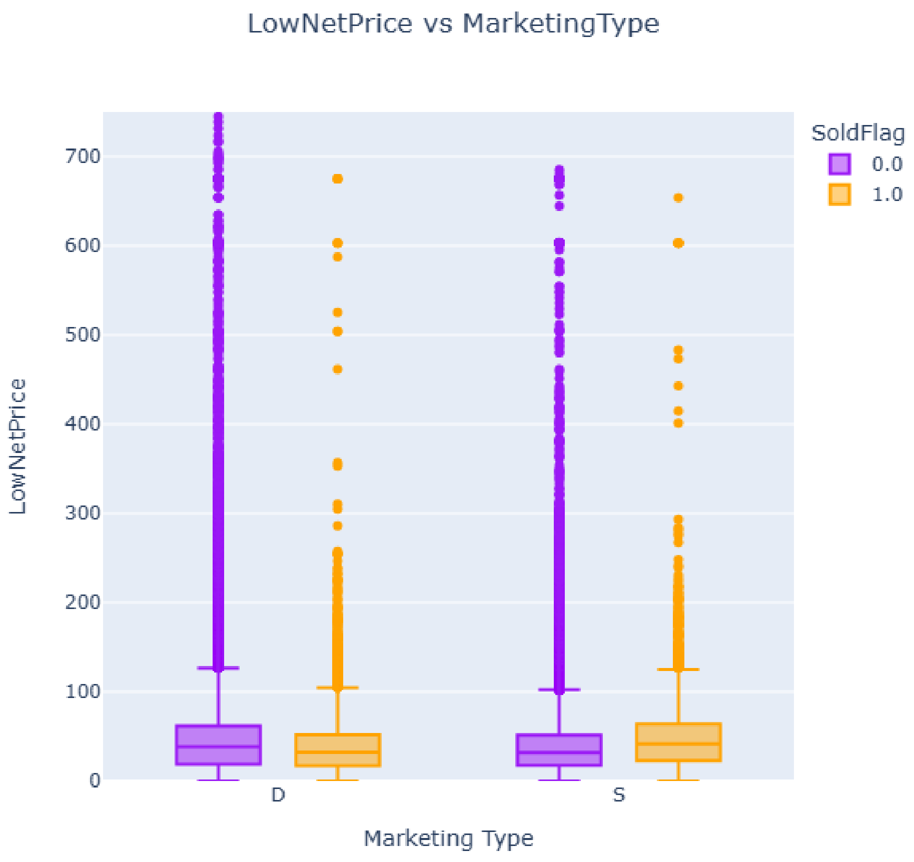
- We can conclude two statements from this visualization. First, most historical SKUs have a planned future release.
- Second, SKUs with a planned future release exhibit better sales performance. We take this second point with a grain of salt since most SKUs have planned future release.
- While this important to take note of, we can also conclude from this that Marketing Type may be a better indicator of sales performance than New Release Flag.

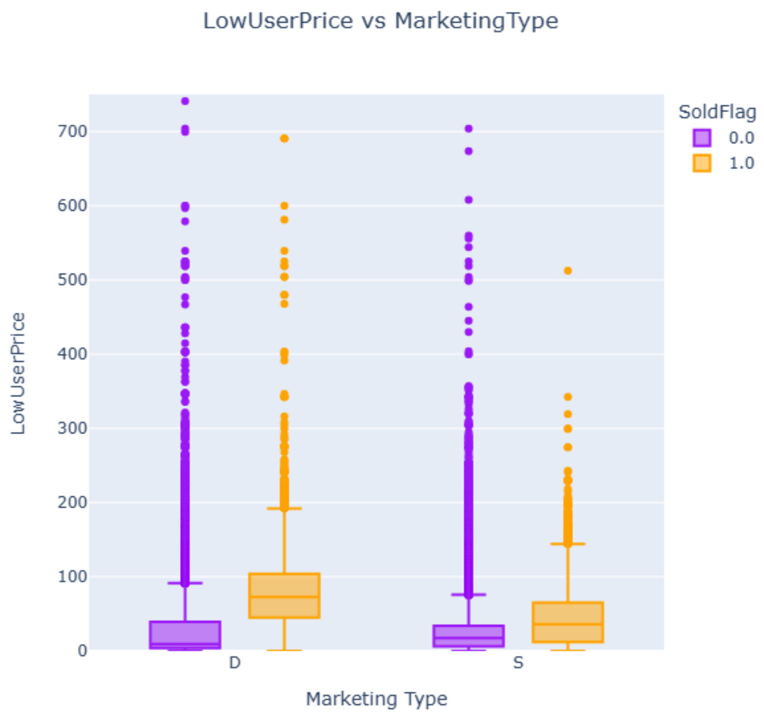
(O)Prices relation by Sales Performance

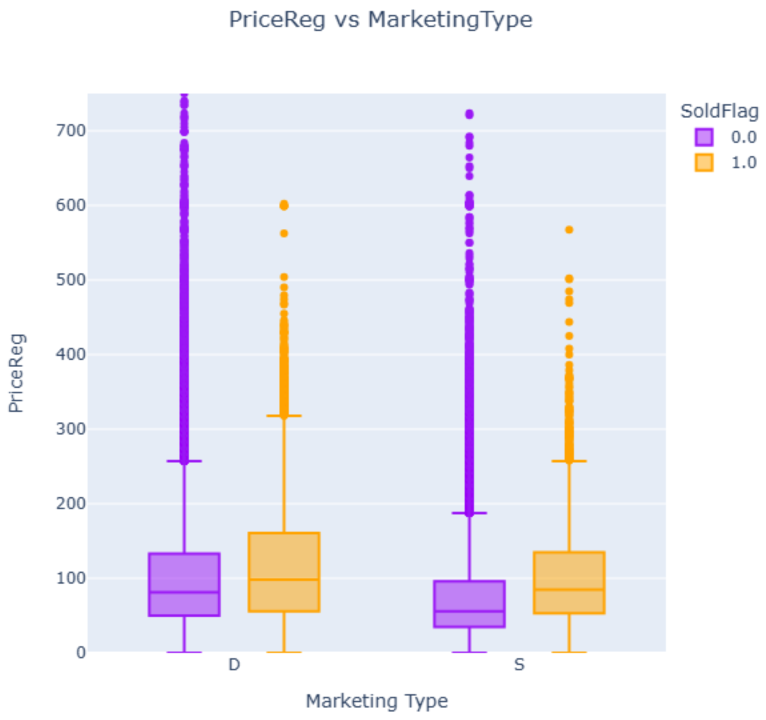
Analysis: Analysing relation between Marketing Type and Price by Sales Performance

```
color_map = {
    0: '#9b19f5',
    1: '#ffa300'
}

# Plotting PriceReg vs MarketingType
fig_price_reg = px.box(boxify_dataset,
                       x="MarketingType",
                       y="PriceReg",
                       color="SoldFlag",
                       title="PriceReg vs MarketingType",
                       labels={"MarketingType": "Marketing Type", "PriceReg": "PriceReg"},
                       category_orders={"SoldFlag": [0, 1]},
                      color_discrete_map=color_map)

fig_price_reg.update_yaxes(range=[0, 750])
fig_price_reg.update_layout(title_x=0.5, autosize=False, width=600, height=600)
fig_price_reg.show()
```

```
# Plotting LowUserPrice vs MarketingType
fig_low_user_price = px.box(boxify_dataset,
                             x="MarketingType",
                             y="LowUserPrice",
                             color="SoldFlag",
                             title="LowUserPrice vs MarketingType",
                             labels={"MarketingType": "Marketing Type", "LowUserPrice": "LowUserPrice"},
                             category_orders={"SoldFlag": [0, 1]},
                            color_discrete_map=color_map)
fig_low_user_price.update_yaxes(range=[0, 750])
fig_low_user_price.update_layout(title_x=0.5, autosize=False, width=600, height=600)
fig_low_user_price.show()

# Plotting LowNetPrice vs MarketingType
fig_low_net_price = px.box(boxify_dataset,
                            x="MarketingType",
                            y="LowNetPrice",
                            color="SoldFlag",
                            title="LowNetPrice vs MarketingType",
                            labels={"MarketingType": "Marketing Type", "LowNetPrice": "LowNetPrice"},
                            category_orders={"SoldFlag": [0, 1]},
                           color_discrete_map=color_map)
fig_low_net_price.update_yaxes(range=[0, 750])
fig_low_net_price.update_layout(title_x=0.5, autosize=False, width=600, height=600)
fig_low_net_price.show()
```







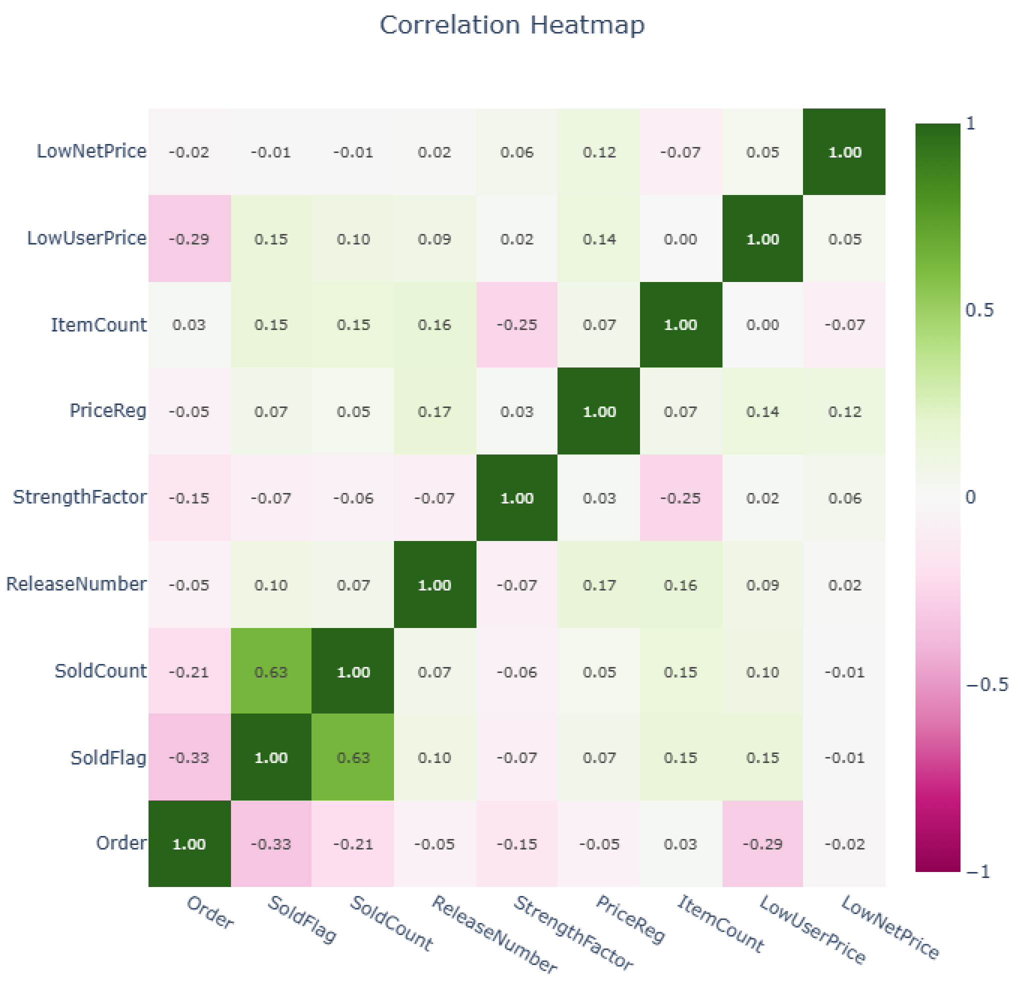
Insights:

- For "PriceReg" and "LowUserPrice", median price is higher for SKUs that have been sold in the last six months within both MartketingType buckets and MarketingType "D" exhibits more sales.
- Again, we can see that "LowNetPrice" shows a discrepancy in this trend.

(P) Correlation Matrix

Analysis: Deriving relationships among each variable

```
# Select only numeric columns from the DataFrame
numeric_data = boxify_dataset.select_dtypes(include=['float64', 'int64'])

# Calculate the correlation matrix
corr_matrix = numeric_data.corr()

# Create a Plotly heatmap
fig = go.Figure(data=go.Heatmap(
    z=corr_matrix.values,
    x=corr_matrix.columns,
    y=corr_matrix.columns,
    colorscale='PiYG',
    zmin=-1, zmax=1,
    text=corr_matrix.values,
    texttemplate="%{text:.2f}",
    textfont={"size":10},
))

# Update layout
fig.update_layout(
    title='Correlation Heatmap',
    xaxis_nticks=36,
    yaxis_nticks=36,
    title_x=0.5, autosize=False, width=700, height=700
)

fig.show()
```



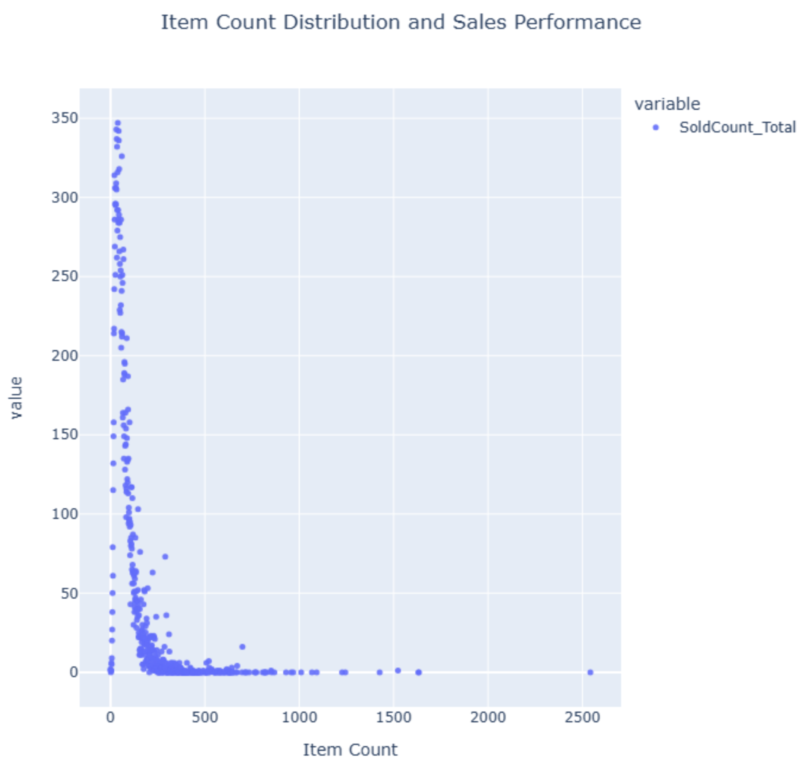
Insights:

- Order and SoldFlag (0.63): Orders are strongly correlated with the SoldFlag, indicating that as orders increase, the sold flag (indicating items are sold) also increases.
- SoldCount and SoldFlag (0.63): SoldCount is strongly correlated with the SoldFlag, meaning that higher sold counts are associated with a higher likelihood of an item being sold.
- Most other variables such as LowNetPrice, ItemCount, PriceReg, StrengthFactor, and ReleaseNumber show weak or negligible correlations with each other and with Order and SoldFlag.
- SoldFlag and LowUserPrice (-0.33): There is a moderate negative correlation between the SoldFlag and LowUserPrice, suggesting that higher user prices might reduce the likelihood of items being sold.
- Order and SoldFlag (-0.33): Orders show a moderate negative correlation with LowUserPrice, suggesting that higher user prices may lead to fewer orders.

(Q) Item Count Distribution

Analysis: Item Count Distribution by average of sold counts of SKUs

```python
item_count_sales = boxify_dataset.groupby('ItemCount')['SoldCount'].sum().reset_index()
item_count_avg_sales = boxify_dataset.groupby('ItemCount')['SoldCount'].mean().reset_index()

# Merge the two dataframes
item_count_summary = pd.merge(item_count_sales, item_count_avg_sales, on='ItemCount', suffixes=('_Total', '_Average'))


# Plot the distribution and average of ItemCount in relation to sales performance
fig = px.scatter(
    item_count_summary,
    x='ItemCount',
    y=['SoldCount_Total'],
    title='Item Count Distribution and Sales Performance',
    labels={'ItemCount': 'Item Count', 'SoldCount': 'Sales Count'},
    opacity=0.9
)
fig.update_layout(title_x=0.5, autosize=False, width=700, height=700)
fig.update_traces(marker=dict(size=5))
fig.show()
```



Insights:

- The majority of data points are concentrated at lower values of `ItemCount`. This indicates that most of the items have relatively low counts.
- This is consistent with typical sales data where a large number of items sell in smaller quantities.
- The blue trend line, representing `SoldCount_Total`, shows a downward slope. This suggests that as `ItemCount` increases, the total sales tend to decrease.
- This could indicate that items with higher counts do not necessarily translate to higher total sales, possibly because of decreased demand or lower popularity of bulk items.

Actionable Recommendations to Optimize Inventory Management Based on Sales Patterns

1. Implications for Marketing Strategies and Recommendations

- The company might consider investing more resources into Marketing Type "D" as it clearly drives higher sales.
- Further analysis could be conducted to understand why Marketing Type "D" is more effective and if there are specific aspects of this marketing type that can be applied to Marketing Type "S" to boost its performance.

- If the types S and D correspond to specific marketing channels, campaigns, or target audiences, this balanced distribution could be part of a broader strategy to maximize coverage and impact.
- Implement targeted promotions and discounts during periods of low sales to boost demand.
- Analyse the effectiveness of promotions through the impact on Order and SoldFlag variables and to refine future marketing strategies.
- Further analysis could explore the effectiveness of each type in terms of conversion rates, customer acquisition, and ROI.
- Conduct regular market research to understand changing consumer preferences.
- Gather feedback from customers to identify areas for improvement and innovation.

2. Inventory Turnover Recommendations

- Ensure that SKUs with high turnover ratios are always adequately stocked to meet demand. Implement efficient restocking processes for these items.
- Consider marketing or promotional strategies to boost the sales of SKUs with moderate turnover ratios to improve their performance.
- Analyse the reasons behind the low turnover ratios for certain SKUs. Consider strategies such as discounts, bundling, or phasing out underperforming products to optimize inventory.
- Regularly analyze inventory turnover ratios to identify slow-moving or obsolete stock. High turnover indicates efficient inventory management, while low turnover may signal overstocking issues.
- For SKUs with low reorder points, analyze if they are slow-moving items. Consider strategies to increase their turnover, such as promotions or bundling with faster-moving items.
- If these SKUs are indeed slow-moving, evaluate the possibility of reducing their reorder points further to minimize holding costs.

3. Inventory and Supply Chain Considerations

- Given the higher sales of products under Marketing Type "D", the company should ensure that there is sufficient inventory to meet the demand.
- Marketing efforts and budget allocations should be optimized to capitalize on the success of Marketing Type "D".
- Strengthen relationships with suppliers to ensure a reliable and flexible supply chain. Good supplier communication can help adjust orders quickly in response to changing demand.
- Consider collaborating with suppliers for Vendor Managed Inventory (VMI) arrangements for SKUs with high variability in demand to ensure smoother stock replenishments.

4. Inventory Optimization

- Use the insights from the turnover ratios to refine inventory management practices, ensuring a balance between sufficient stock levels and minimizing excess inventory.
- Focus on the 3379 SKUs that are identified as low stock to prevent potential stockouts.

- Analyse the sales patterns of the top 10 SKUs to ensure the high stock levels are justified and not leading to excess inventory.
- Consider adjusting the low stock threshold based on sales velocity and lead time for restocking to optimize inventory levels further.
- Products with higher sold counts should be prioritized in inventory management to ensure they are always in stock, while those with lower counts may need re-evaluation to understand if they should continue to be offered or need marketing support.
- Based on the correlation between Order and SoldFlag, ensure sufficient inventory levels for items with high order volumes to avoid stockouts.
- For items with low correlation between SoldFlag and other variables, consider implementing just-in-time (JIT) inventory practices to minimize holding costs.

5. Implications of Pricing Strategies

- The significant difference in Low User Price between active and historical products suggests a strategic pricing approach to attract users with lower prices for active products.
- The relatively higher regular and net prices for historical products may indicate their higher perceived value or rarity compared to active products.
- Adjust pricing strategies to optimize orders and sales. Given the moderate correlation between LowUserPrice and Order, consider offering competitive pricing to boost order volumes.
- Monitor the impact of pricing on sales and sold flags to ensure that price reductions do not excessively erode profit margins.
- Implement dynamic pricing strategies that adjust prices based on demand, competition, and other market factors. Utilize the correlation data to identify optimal price points that maximize orders and sales.

6. Recommendations for demand forecasting and improving sales

- Investigate the factors that contributed to the peak sales period in the late 1990s and early 2000s.
- Understand the marketing strategies, product offerings, and external factors that drove high sales during this time.
- Use historical sales data to forecast future demand more accurately. This will help avoid the pitfalls of overstocking or understocking, as seen in the sharp rise and fall around the year 2000.
- Implement predictive analytics to anticipate sales trends and adjust inventory levels accordingly.
- Continuously monitor sales data and market trends to adapt quickly to changes.
- Perform a detailed demand analysis for SKUs 665269 and 613864 to ensure that their high reorder points are justified by their sales patterns.
- Evaluate and possibly optimize the lead times for these SKUs to see if reorder points can be reduced without risking stockouts.

- For SKUs with moderate reorder points, ensure the safety stock levels are optimized to balance carrying costs and service levels.
- Continuously analyse historical sales and inventory data to identify long-term trends and patterns. This can provide valuable insights for strategic planning and inventory optimization.
- Utilize the strong correlation between SoldCount and SoldFlag to improve demand forecasting models. Accurate predictions of sold items can help in planning inventory levels more effectively.

Conclusion

The analysis of sales data has yielded valuable insights into product performance, marketing efficacy, and inventory management. The top-selling products, particularly SKUs 665269 and 613864, demonstrate a high demand, underscoring the importance of maintaining adequate stock levels for these items. Additionally, products marketed under type 'D' significantly outperform those under type 'S,' suggesting the superior effectiveness of this marketing strategy. Key inventory performance metrics, including a calculated reorder point of 1127.40 units, highlight the need for strategic inventory planning to prevent stockouts and ensure a smooth supply chain. Furthermore, pricing analysis indicates that historical products tend to have higher average prices compared to active ones, providing a basis for strategic pricing decisions. These insights collectively offer actionable recommendations to optimize sales, enhance marketing strategies, and improve inventory management, ultimately contributing to better business performance. The data-driven recommendations foster a proactive approach to business management, leading to optimized sales strategies, improved customer service, and ultimately, a stronger market position.

Repository link

https://github.com/shilpi-verma/Boxify_CapstoneProject.git