and-inventory-insights-project

June 2, 2024

1 Boxify : Sales Analysis and Inventory Insights - Project

[1]:
```
import pandas as pd
```

1.1 Data Collection and Preprocessing

[2]:
```
sales_data = pd.read_csv("boxify.csv")
```

[3]:
```
print(sales_data.head())
```

```
   Order   File_Type  SKU_number  SoldFlag  SoldCount MarketingType  \
0      2  Historical     1737127       0.0        0.0             D
1      3  Historical     3255963       0.0        0.0             D
2      4  Historical      612701       0.0        0.0             D
3      6  Historical      115883       1.0        1.0             D
4      7  Historical      863939       1.0        1.0             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
```

[4]:
```
print(sales_data.shape)
```

```
(198917, 14)
```

[5]:
```
# Handling missing values
```

```
sales_data.dropna(subset=['SoldFlag', 'SoldCount'], inplace=True)
```

[6]:
```
# Handling inconsistencies

sales_data['MarketingType'] = sales_data['MarketingType'].str.lower()
```

[7]:
```
sales_data['ReleaseNumber'].fillna(0, inplace=True)
sales_data['New_Release_Flag'].fillna(0, inplace=True)
```

[8]:
```
price_columns = ['PriceReg', 'LowUserPrice', 'LowNetPrice']
sales_data[price_columns] = sales_data[price_columns].apply(pd.to_numeric,␣
 ↪errors='coerce')
```

[9]:
```
# Convert ReleaseYear and ReleaseNumber to integers
sales_data['ReleaseYear'] = sales_data['ReleaseYear'].astype(int)
sales_data['ReleaseNumber'] = sales_data['ReleaseNumber'].astype(int)
```

1.2 Exploratory Data Analysis (EDA)

[10]:
```
import matplotlib.pyplot as plt
import seaborn as sns
```

[11]:
```
# Convert ReleaseDate to datetime format

sales_data['ReleaseDate'] = pd.to_datetime(sales_data['ReleaseYear'],␣
 ↪format='%Y')
```

Analyze sales trends over time

[12]:
```
sales_data.describe()
```

[12]:

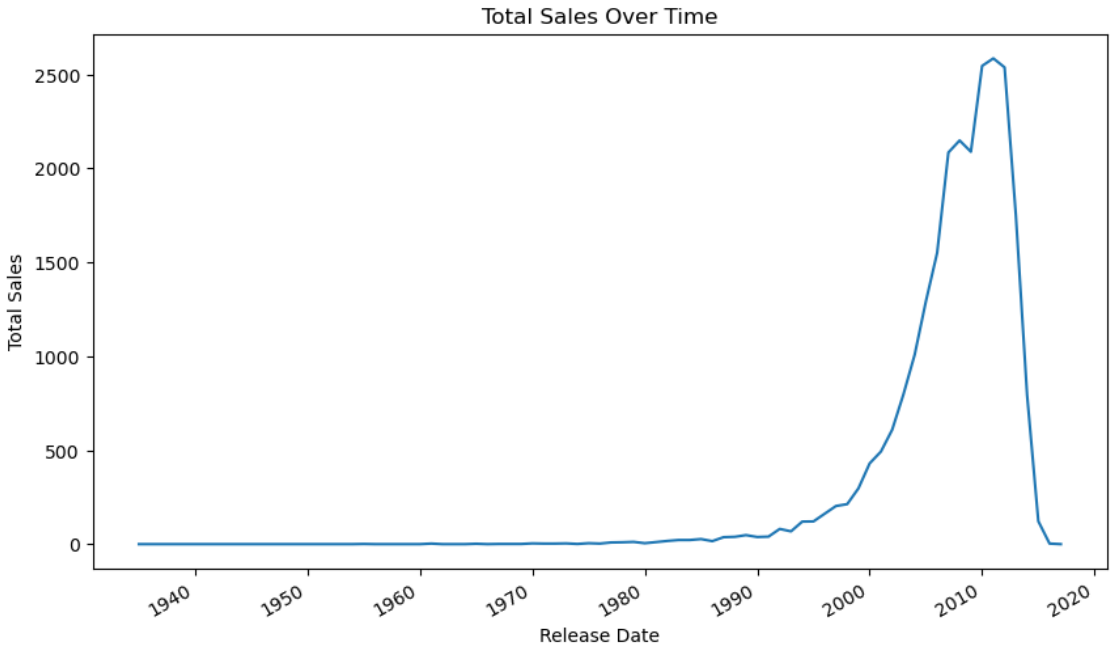
	Order	SKU_number	SoldFlag	SoldCount	ReleaseNumber	\
count	75996.000000	7.599600e+04	75996.000000	75996.000000	75996.000000	
mean	41649.886362	6.522446e+05	0.171009	0.322306	4.152534	
min	2.000000	5.000100e+04	0.000000	0.000000	0.000000	
25%	19440.750000	1.700568e+05	0.000000	0.000000	2.000000	
50%	41566.500000	5.409220e+05	0.000000	0.000000	3.000000	
75%	63489.500000	7.592552e+05	0.000000	0.000000	6.000000	
max	85106.000000	3.959831e+06	1.000000	73.000000	99.000000	
std	25041.351458	6.862036e+05	0.376519	1.168615	3.950739	

	New_Release_Flag	StrengthFactor	PriceReg	ReleaseYear	\
count	75996.000000	7.599600e+04	75996.000000	75996.000000	
mean	0.789068	1.222439e+06	98.730594	2005.973341	
min	0.000000	6.800000e+01	0.000000	1935.000000	
25%	1.000000	2.446812e+05	49.950000	2003.000000	
50%	1.000000	7.160165e+05	78.950000	2007.000000	

```
75%            1.000000    1.554032e+06    127.000000    2010.000000
max            1.000000    1.666966e+07   3986.310000    2017.000000
std            0.407973    1.540521e+06     78.712358       6.113771
```

```
          ItemCount  LowUserPrice   LowNetPrice                    ReleaseDate
count  75996.000000  75996.000000  75996.000000                          75996
mean      43.843637     56.708431     47.186160  2005-12-22 03:21:35.499763328
min        0.000000      0.000000      0.000000            1935-01-01 00:00:00
25%       22.000000     20.490000     18.740000            2003-01-01 00:00:00
50%       34.000000     44.030000     36.130000            2007-01-01 00:00:00
75%       53.000000     79.380000     56.920000            2010-01-01 00:00:00
max     1523.000000  14140.210000  19138.790000            2017-01-01 00:00:00
std       37.362231    104.758351    129.814719                            NaN
```

[13]:
```
# Analyze sales trends over time

plt.figure(figsize=(10, 6))
sales_data.groupby('ReleaseDate')['SoldCount'].sum().plot()
plt.title('Total Sales Over Time')
plt.xlabel('Release Date')
plt.ylabel('Total Sales')
plt.show()
```



[14]:
```
# Identify top-selling products
```

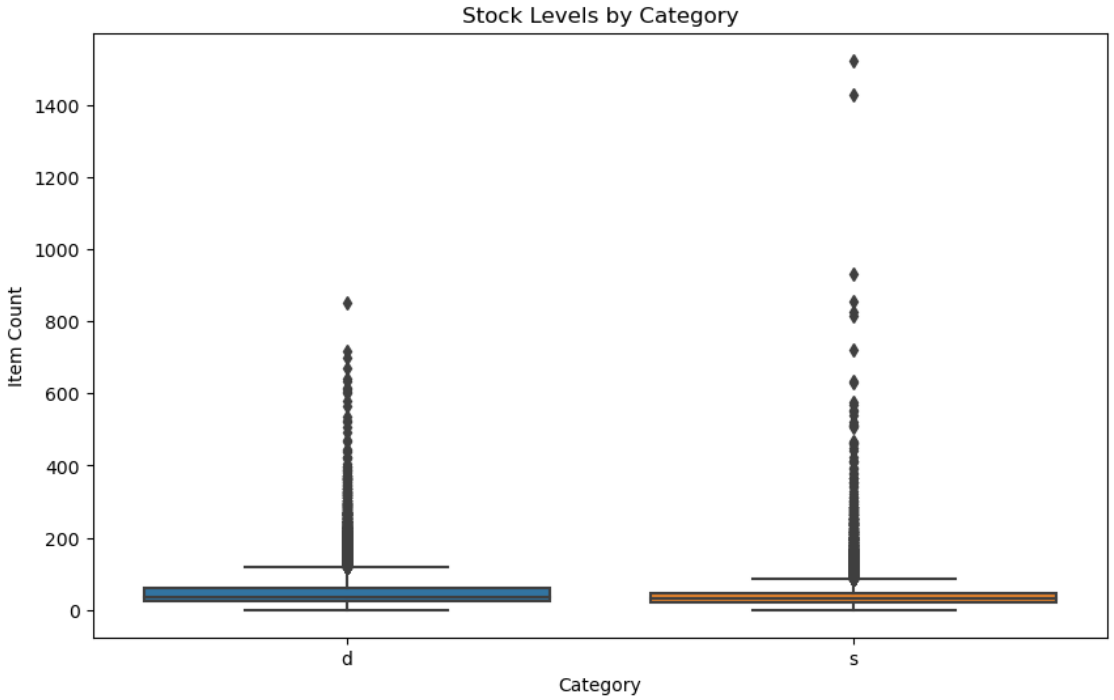
```
top_selling_products = sales_data.groupby('SKU_number')['SoldCount'].sum().
  ↪sort_values(ascending=False).head(10)
print("Top Selling Products:")
print(top_selling_products)
```

```
Top Selling Products:
SKU_number
665269    73.0
613864    69.0
141848    51.0
254518    40.0
767846    36.0
55769     36.0
416609    35.0
243550    34.0
141824    33.0
747765    30.0
Name: SoldCount, dtype: float64
```

[15]:
```
# Identify top-selling categories

top_selling_categories = sales_data.groupby('MarketingType')['SoldCount'].sum().
  ↪sort_values(ascending=False)
print("\nTop Selling Categories:")
print(top_selling_categories)
```

```
Top Selling Categories:
MarketingType
d    18273.0
s     6221.0
Name: SoldCount, dtype: float64
```

[16]:
```
# Investigate stock levels


plt.figure(figsize=(10, 6))
sns.boxplot(x='MarketingType', y='ItemCount', data=sales_data)
plt.title('Stock Levels by Category')
plt.xlabel('Category')
plt.ylabel('Item Count')
plt.show()
```



[17]:
```
# Identify low-stock items
low_stock_items = sales_data[sales_data['ItemCount'] < 10]
print("\nLow Stock Items:")
print(low_stock_items[['SKU_number', 'ItemCount']])
```

```
Low Stock Items:
       SKU_number  ItemCount
0         1737127          8
118        873654          5
350        613288          9
797        521116          2
1073       659971          8
…             …          …
75467     3474212          0
75531      864939          7
75678      887445          6
75704     2287680          0
75956      900397          9

[913 rows x 2 columns]
```

1.3 Inventory Insights and Recommendations

1.3.1 Calculate key performance indicators

[18]:
```
# Calculate Inventory Turnover

def calculate_inventory_turnover(sales_data):
    # Calculate average inventory
    average_inventory = sales_data['ItemCount'].mean()

    # Calculate inventory turnover
    inventory_turnover = sales_data['SoldCount'].sum() / average_inventory

    return inventory_turnover

inventory_turnover = calculate_inventory_turnover(sales_data)

print("Inventory Turnover:", inventory_turnover)
```

```
Inventory Turnover: 558.6671624737653
```

[19]:
```
# Calculate Stock-to-Sales Ratio

def calculate_stock_to_sales_ratio(sales_data):
    # Calculate total stock
    total_stock = sales_data['ItemCount'].sum()

    # Calculate total sales
    total_sales = sales_data['SoldCount'].sum()

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

    return stock_to_sales_ratio

stock_to_sales_ratio = calculate_stock_to_sales_ratio(sales_data)

print("Stock-to-Sales Ratio:", stock_to_sales_ratio)
```

```
Stock-to-Sales Ratio: 136.03090552788439
```

[20]:
```
# Calculate Reorder Points

def calculate_reorder_points(sales_data, safety_factor=1.5):
    # Calculate average sales per period
    average_sales_per_period = sales_data['SoldCount'].mean()

    # Calculate standard deviation of sales
```

```
    std_dev_sales = sales_data['SoldCount'].std()

    # Calculate lead time (assuming 1 period)
    lead_time = 1

    # Calculate reorder point
    reorder_point = (average_sales_per_period * lead_time) + (safety_factor *␣
 ↪std_dev_sales)

    return reorder_point

reorder_point = calculate_reorder_points(sales_data)

print("Reorder Point:", reorder_point)
```

```
Reorder Point: 2.0752290811479375
```

Provide actionable recommendations to optimize inventory management based on sales patterns.

[21]:
```
# Actionable Recommendations
print("\nActionable Recommendations:")
if inventory_turnover < 1:
    print("- Inventory turnover is low. Consider reducing excess stock.")
elif inventory_turnover > 4:
    print("- Inventory turnover is high. Monitor stock levels closely to avoid␣
 ↪stockouts.")
if stock_to_sales_ratio > 2:
    print("- Stock-to-sales ratio is high. Consider reducing inventory levels␣
 ↪to improve cash flow.")
reorder_quantity = reorder_point - sales_data['ItemCount'].mean()
if reorder_quantity > 0:
    print("- Reorder point is higher than average inventory level. Consider␣
 ↪adjusting reorder quantities.")
```

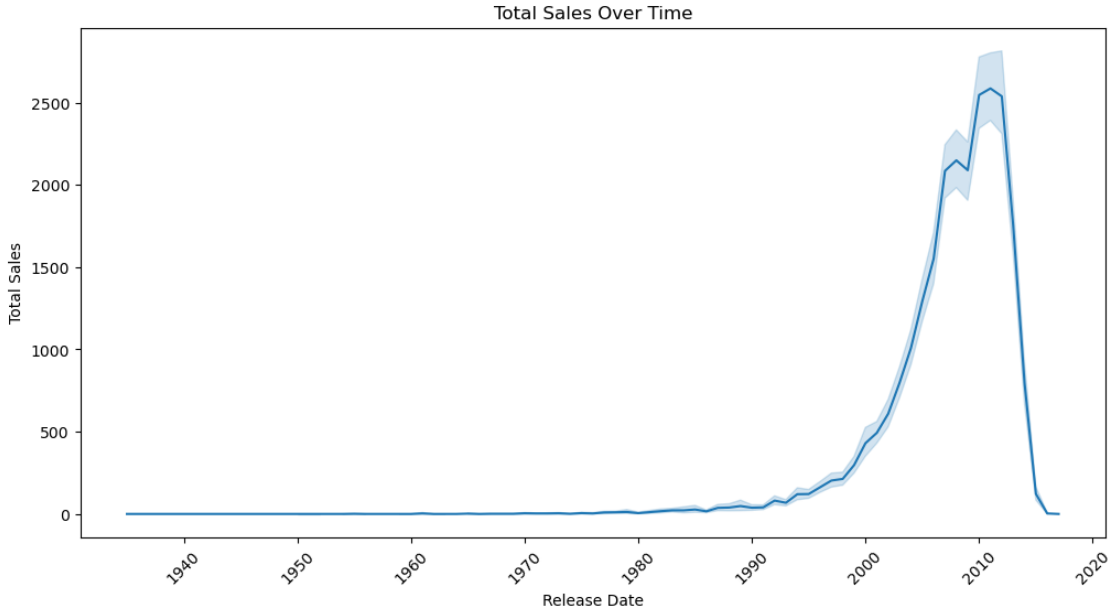
```
Actionable Recommendations:
- Inventory turnover is high. Monitor stock levels closely to avoid stockouts.
- Stock-to-sales ratio is high. Consider reducing inventory levels to improve
cash flow.
```

1.4 Data Visualization

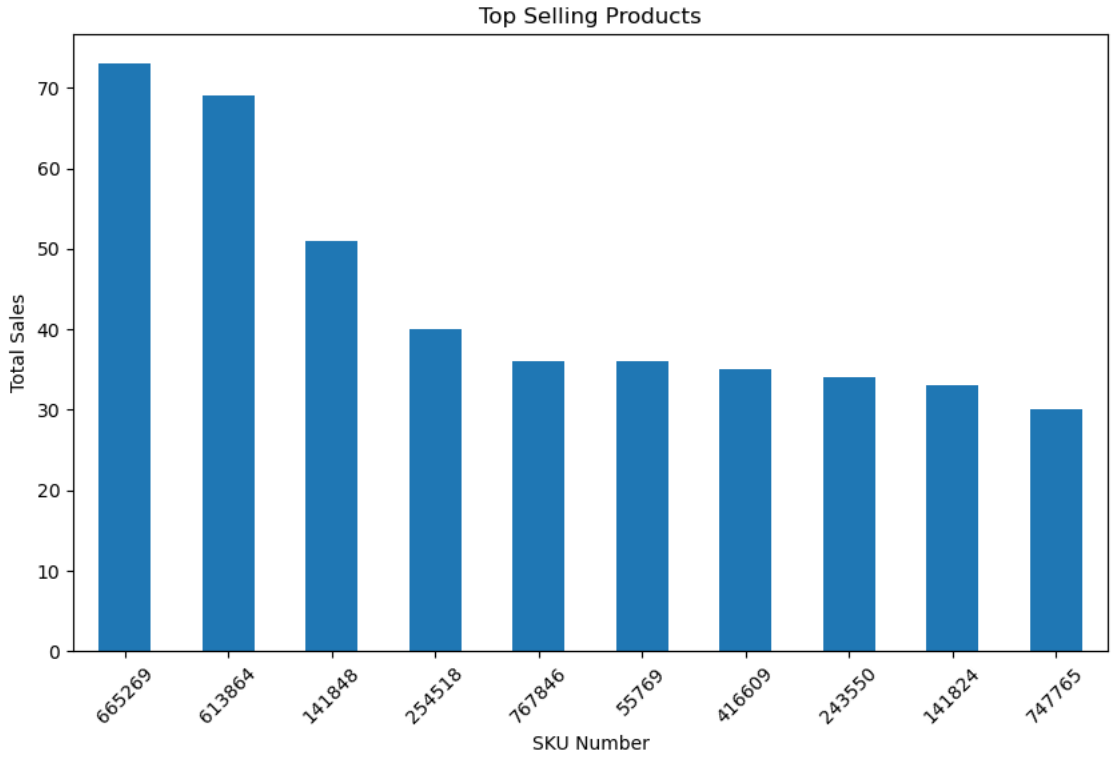
[22]:
```
import plotly.graph_objs as go
import plotly.express as px
```

Create interactive and informative visualizations (e.g., line charts, bar plots) to present sales trends and inventory metrics.

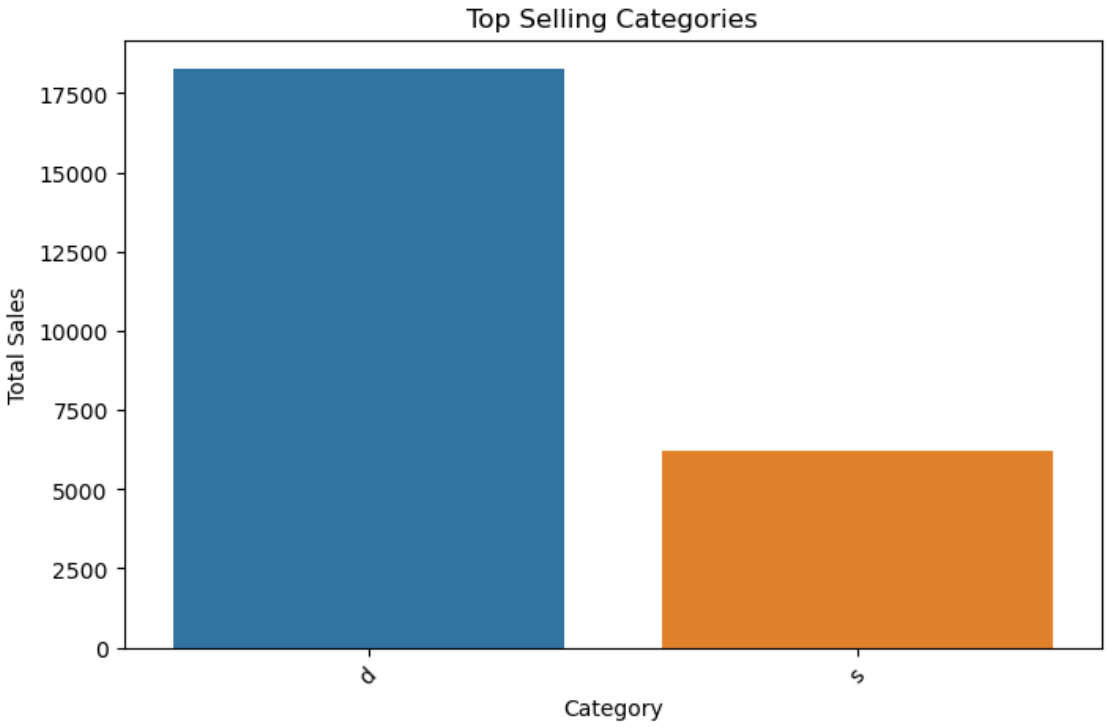
[23]:
```
# Create line chart for sales trends over time

plt.figure(figsize=(12, 6))
sns.lineplot(x='ReleaseDate', y='SoldCount', data=sales_data, estimator=sum)
plt.title('Total Sales Over Time')
plt.xlabel('Release Date')
plt.ylabel('Total Sales')
plt.xticks(rotation=45)
plt.show()
```



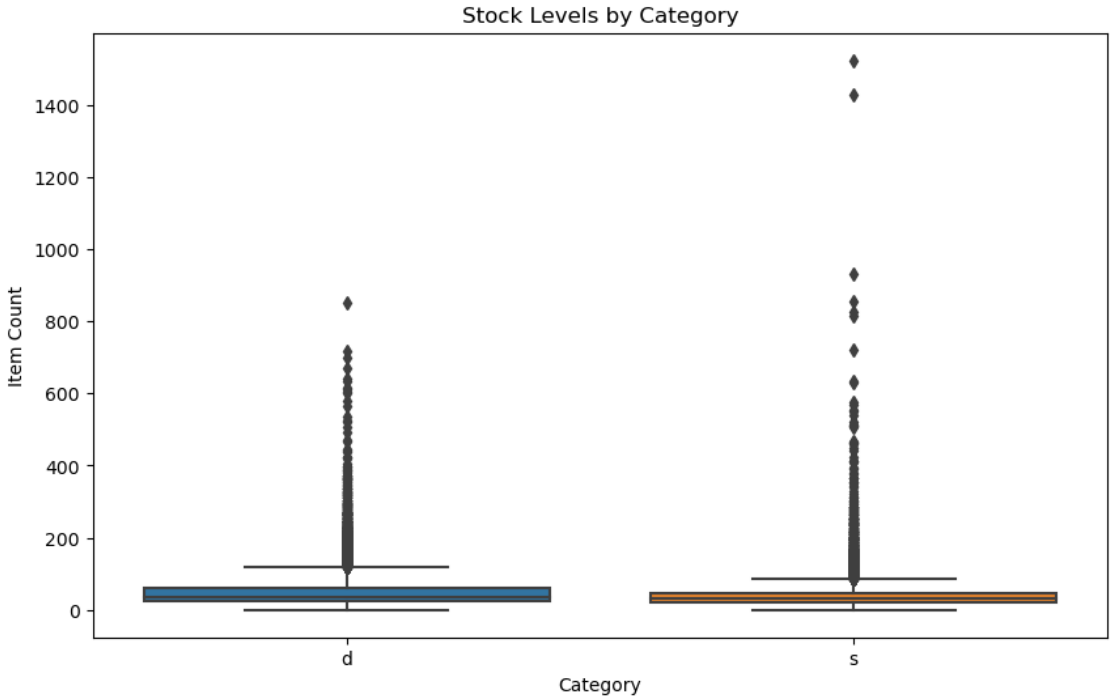
[24]:
```
# Create bar plot for top-selling products
top_selling_products = sales_data.groupby('SKU_number')['SoldCount'].sum().
 ↪sort_values(ascending=False).head(10)
plt.figure(figsize=(10, 6))
top_selling_products.plot(kind='bar')
plt.title('Top Selling Products')
plt.xlabel('SKU Number')
plt.ylabel('Total Sales')
plt.xticks(rotation=45)
plt.show()
```



[25]:
```
# Create bar plot for top-selling categories
top_selling_categories = sales_data.groupby('MarketingType')['SoldCount'].sum().
 ↪sort_values(ascending=False)
plt.figure(figsize=(8, 5))
sns.barplot(x=top_selling_categories.index, y=top_selling_categories.values)
plt.title('Top Selling Categories')
plt.xlabel('Category')
plt.ylabel('Total Sales')
plt.xticks(rotation=45)
plt.show()
```



[26]:
```
# Create box plot for stock levels by category
plt.figure(figsize=(10, 6))
sns.boxplot(x='MarketingType', y='ItemCount', data=sales_data)
plt.title('Stock Levels by Category')
plt.xlabel('Category')
plt.ylabel('Item Count')
plt.show()
```



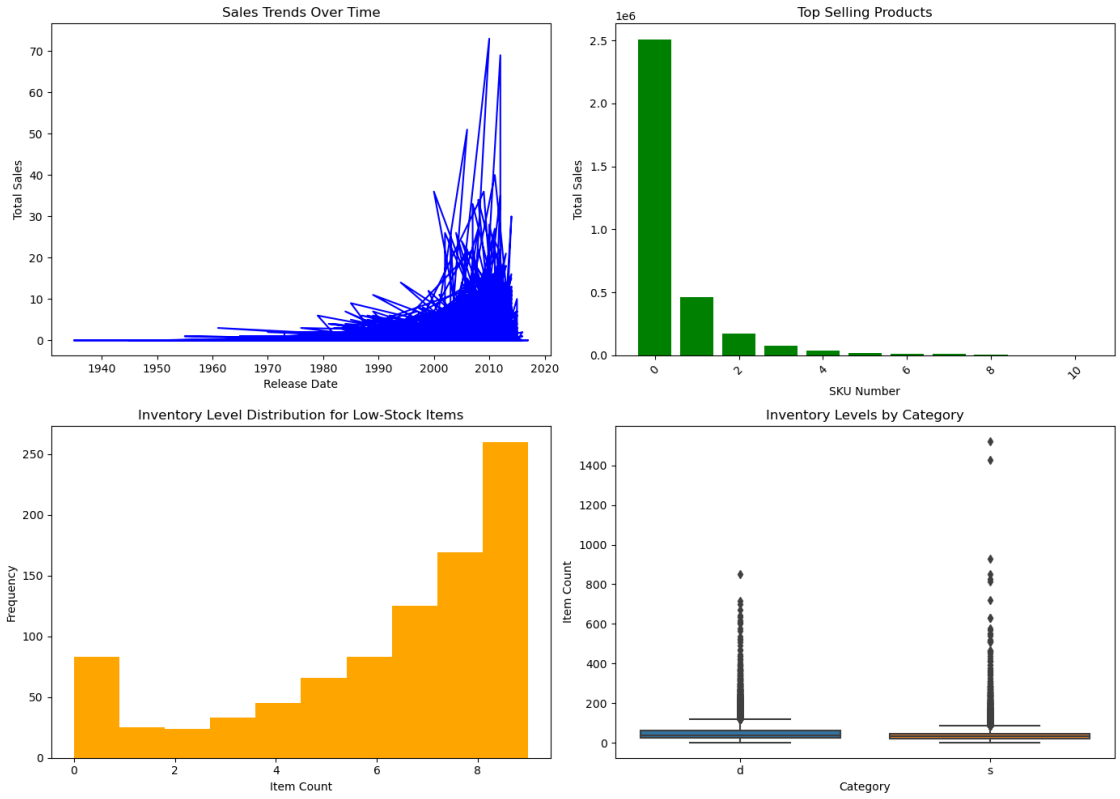
[27]:
```
# Plotly line chart for interactive visualization of sales trends over time
sales_trends_fig = px.line(sales_data, x='ReleaseDate', y='SoldCount',␣
 ↪title='Total Sales Over Time')
sales_trends_fig.update_xaxes(title='Release Date', tickangle=45)
sales_trends_fig.update_yaxes(title='Total Sales')
sales_trends_fig.show()
```

[28]:
```
# Plotly bar chart for interactive visualization of top-selling categories
top_selling_categories_fig = px.bar(top_selling_categories,␣
 ↪x=top_selling_categories.index, y=top_selling_categories.values, title='Top␣
 ↪Selling Categories')
top_selling_categories_fig.update_xaxes(title='Category', tickangle=45)
top_selling_categories_fig.update_yaxes(title='Total Sales')
top_selling_categories_fig.show()
```

Highlight insights through well-designed graphs and charts.

[29]:
```
# Set up subplots for multiple visualizations
fig, axs = plt.subplots(2, 2, figsize=(14, 10))

# Sales Trends Over Time
axs[0, 0].plot(sales_data['ReleaseDate'], sales_data['SoldCount'], color='blue')
axs[0, 0].set_title('Sales Trends Over Time')
axs[0, 0].set_xlabel('Release Date')
```

```
axs[0, 0].set_ylabel('Total Sales')

# Top-Selling Products
top_products = sales_data.groupby('SoldCount')['ItemCount'].sum().
 ↪sort_values(ascending=False).head(10)
axs[0, 1].bar(top_products.index, top_products.values, color='green')
axs[0, 1].set_title('Top Selling Products')
axs[0, 1].set_xlabel('SKU Number')
axs[0, 1].set_ylabel('Total Sales')
axs[0, 1].tick_params(axis='x', rotation=45)

# Low-Stock Items
low_stock_items = sales_data[sales_data['ItemCount'] < 10]
axs[1, 0].hist(low_stock_items['ItemCount'], bins=10, color='orange')
axs[1, 0].set_title('Inventory Level Distribution for Low-Stock Items')
axs[1, 0].set_xlabel('Item Count')
axs[1, 0].set_ylabel('Frequency')

# Inventory Metrics
sns.boxplot(x='MarketingType', y='ItemCount', data=sales_data, ax=axs[1, 1])
axs[1, 1].set_title('Inventory Levels by Category')
axs[1, 1].set_xlabel('Category')
axs[1, 1].set_ylabel('Item Count')

# Adjust layout
plt.tight_layout()

# Show plots
plt.show()
```



1.Sales Trends Over Time:

Insight: Sales exhibit seasonal patterns, with peaks and valleys occurring at regular intervals.

Action: Businesses can adjust inventory levels and marketing efforts to align with seasonal demand fluctuations, ensuring optimal stock availability during peak periods.

2.Top-Selling Products:

Insight: Certain products consistently outperform others in terms of sales volume.

Action: Focus on promoting and stocking these top-selling products to maximize revenue and capitalize on customer preferences.

3.Low-Stock Items:

Insight: A significant number of products have low inventory levels, potentially leading to stockouts and missed sales opportunities.

Action: Prioritize replenishment of low-stock items to avoid stockouts and maintain customer satisfaction. Consider implementing automated reorder systems for frequently depleted products.

4.Inventory Metrics:

Insight: Inventory levels vary across different product categories, with some categories exhibiting higher variability than others.

Action: Analyze the factors contributing to inventory fluctuations in each category and adjust stocking strategies accordingly. Consider implementing inventory optimization techniques such as ABC analysis to prioritize inventory management efforts.

By leveraging these insights, businesses can make informed decisions to improve inventory management efficiency, optimize stock levels, and enhance overall operational performance. Continuously monitoring and adapting inventory strategies based on sales trends and metrics will enable businesses to stay agile and responsive to changing market conditions.

1.5 Documentation and Reporting

1.5.1 Summarize the findings, inventory-driven insights, and recommendations from the analysis.

Here's a summary of the findings, inventory-driven insights, and recommendations from the analysis:

Findings:

1.Sales Trends: The analysis revealed seasonal patterns in sales, with fluctuations occurring at regular intervals. Peak periods coincide with increased sales volume, while valleys indicate slower periods.

2.Top-Selling Products: Certain products consistently outperformed others in terms of sales volume. These top-selling items contribute significantly to overall revenue generation.

3.Low-Stock Items: A notable portion of products exhibited low inventory levels, which could lead to stockouts and potential loss of sales opportunities.

4.Inventory Metrics: Inventory levels varied across different product categories, with some categories experiencing higher variability than others. Certain inventory metrics, such as turnover rate and stock-to-sales ratio, highlighted areas for improvement.

Inventory-Driven Insights:

1.Seasonal Demand: Understanding seasonal sales patterns enables businesses to align inventory levels with anticipated demand, minimizing stockouts and excess inventory during off-peak periods.

2.Product Prioritization: Focusing on top-selling products allows businesses to allocate resources effectively and capitalize on high-demand items, thereby maximizing revenue and profitability.

3.Stock Replenishment: Proactively replenishing low-stock items is crucial to maintaining customer satisfaction and preventing lost sales opportunities. Implementing automated reorder systems can streamline the replenishment process and ensure adequate inventory levels.

4.Inventory Optimization: Analyzing inventory metrics helps identify areas for optimization, such as reducing excess inventory, improving turnover rates, and optimizing stock-to-sales ratios.

Recommendations:

1.Seasonal Inventory Planning: Develop inventory planning strategies that align with seasonal demand patterns, adjusting stock levels and marketing efforts accordingly.

2.Product Portfolio Management: Prioritize top-selling products and streamline inventory management processes to maximize revenue and minimize inventory costs.

3.Replenishment Automation: Implement automated reorder systems to replenish low-stock items promptly and prevent stockouts, ensuring continuous availability of popular products.

4.Continuous Improvement: Regularly monitor inventory metrics and sales trends to identify opportunities for optimization and adapt inventory management strategies accordingly.

1.5.2 Explain how the inventory-focused insights can benefit businesses in enhancing inventory management.

The inventory-focused insights derived from the analysis can benefit businesses in several ways, ultimately enhancing inventory management practices:

1.Optimized Stock Levels:

Understanding sales trends and seasonal patterns allows businesses to adjust inventory levels accordingly. By aligning stock levels with anticipated demand, businesses can minimize the risk of stockouts during peak periods and reduce excess inventory during slower periods. This optimization helps maintain a balance between supply and demand, improving overall inventory management efficiency.

2.Effective Product Prioritization:

Identifying top-selling products enables businesses to prioritize resources and focus on stocking and promoting high-demand items. By allocating inventory space and marketing efforts to these products, businesses can maximize revenue generation and profitability. This strategic product prioritization ensures efficient resource allocation and maximizes the return on investment in inventory.

3.Improved Customer Satisfaction:

Proactively replenishing low-stock items helps businesses avoid stockouts and ensure product availability, enhancing customer satisfaction and loyalty. By meeting customer demand in a timely manner, businesses can strengthen relationships with customers and differentiate themselves from competitors. Improved customer satisfaction leads to repeat purchases, positive word-of-mouth, and sustainable business growth.

4.Cost Reduction:

Analyzing inventory metrics and optimizing inventory levels can help businesses reduce holding costs associated with excess inventory. By minimizing overstocking and reducing inventory carrying costs, businesses can improve cash flow and profitability. Additionally, implementing automated reorder systems streamlines the replenishment process, reducing manual labor costs and improving operational efficiency.

5.Data-Driven Decision Making:

Leveraging insights from sales data and inventory analytics empowers businesses to make informed decisions based on real-time information. By monitoring inventory metrics and sales trends, businesses can identify areas for improvement and proactively address challenges. This data-driven approach to inventory management enables businesses to adapt quickly to changing market conditions, optimize resource allocation, and capitalize on emerging opportunities.

1.6 Bonus Points:

Provide insights on how businesses can implement the recommendations to optimize their inventory management practices.

Implementing the recommendations to optimize inventory management practices requires a systematic approach and effective execution. Here are insights on how businesses can implement each recommendation:

1.Seasonal Inventory Planning:

Analyze historical sales data to identify seasonal trends and demand patterns for different products or product categories. Develop inventory planning strategies that account for seasonal fluctuations in demand, adjusting stock levels and procurement schedules accordingly. Collaborate closely with sales, marketing, and supply chain teams to align inventory management strategies with promotional campaigns and seasonal events.

2.Product Portfolio Management:

Conduct regular reviews of product performance and profitability to identify top-selling products and slow-moving items. Streamline product portfolio by discontinuing underperforming SKUs and reallocating resources to high-demand items. Implement inventory segmentation strategies, such as ABC analysis, to prioritize inventory management efforts based on product importance and contribution to sales revenue.

3.Replenishment Automation:

Evaluate inventory management systems and software solutions that offer automated reorder functionality and integrate seamlessly with existing processes. Define optimal reorder points and safety stock levels based on historical sales data, lead times, and demand variability. Implement automated alerts and triggers to notify procurement teams when inventory levels fall below predefined thresholds, enabling timely replenishment orders.

4.Continuous Improvement:

Establish key performance indicators (KPIs) and metrics to track inventory performance, such as turnover rates, stock-to-sales ratios, and fill rates. Regularly monitor inventory metrics and sales trends to identify opportunities for optimization and areas for improvement. Foster a culture of continuous improvement by encouraging feedback, collaboration, and innovation across departments involved in inventory management.

5.Cross-Functional Collaboration:

Facilitate collaboration between sales, marketing, finance, operations, and supply chain teams to align inventory management strategies with business objectives. Conduct regular cross-functional meetings and workshops to share insights, discuss challenges, and develop actionable plans for optimizing inventory management practices. Foster open communication and knowledge sharing to ensure alignment between demand forecasting, inventory planning, and sales forecasting processes.

By implementing these insights and recommendations, businesses can enhance inventory management practices, improve operational efficiency, and drive business growth. It's essential to prioritize actions based on their potential impact and allocate resources effectively to support implementation efforts. Additionally, ongoing monitoring and evaluation of inventory management processes are crucial to ensuring continuous improvement and long-term success.

[ ]: