Data Loading

```
import os

# Check current working directory
os.getcwd()
```

```
'c:\\Users\\rapol\\OneDrive\\Desktop\\Data Visualization'
```

Importing liberies

```
# Data handling
import pandas as pd
import numpy as np

# Visualization
import matplotlib.pyplot as plt
import seaborn as sns

# Date & time handling
from datetime import datetime

# System utilities
import os

# Plot settings
plt.style.use("seaborn-v0_8")
plt.rcParams["figure.figsize"] = (12, 6)

# Pandas display settings
pd.set_option("display.max_columns", None)
pd.set_option("display.width", 120)
```

Loading the dataset and preview of dataset

```
# Load dataset (same folder)
df = pd.read_csv("retail_store_inventory.csv")

# Preview
df.head()
```

```
         Date Store ID Product ID     Category Region  Inventory Level
\
0  2022-01-01     S001      P0001    Groceries  North              231

1  2022-01-01     S001      P0002         Toys  South              204

2  2022-01-01     S001      P0003         Toys   West              102

3  2022-01-01     S001      P0004         Toys  North              469
```

```
4  2022-01-01     S001      P0005  Electronics   East              166


   Units Sold  Units Ordered  Demand Forecast  Price  Discount  \
0         127             55           135.47  33.50        20
1         150             66           144.04  63.01        20
2          65             51            74.02  27.99        10
3          61            164            62.18  32.72        10
4          14            135             9.26  73.64         0

  Weather Condition  Holiday/Promotion  Competitor Pricing Seasonality

0             Rainy                  0               29.69      Autumn

1             Sunny                  0               66.16      Autumn

2             Sunny                  1               31.32      Summer

3            Cloudy                  1               34.74      Autumn

4             Sunny                  0               68.95      Summer
```

Load the Dataset

```
# Load dataset
df = pd.read_csv("retail_store_inventory.csv")

# Preview the dataset
df.head()
```

Dataset Overview

```
# Check number of rows and columns
df.shape
# View column names
df.columns
# Dataset structure and missing values
df.info()
# Summary statistics of numerical columns
df.describe()




<class 'pandas.core.frame.DataFrame'>
RangeIndex: 73100 entries, 0 to 73099
Data columns (total 15 columns):
 #   Column              Non-Null Count  Dtype
```

```
---  ------              --------------  -----
 0   Date                73100 non-null  object
 1   Store ID            73100 non-null  object
 2   Product ID          73100 non-null  object
 3   Category            73100 non-null  object
 4   Region              73100 non-null  object
 5   Inventory Level     73100 non-null  int64
 6   Units Sold          73100 non-null  int64
 7   Units Ordered       73100 non-null  int64
 8   Demand Forecast     73100 non-null  float64
 9   Price               73100 non-null  float64
 10  Discount            73100 non-null  int64
 11  Weather Condition   73100 non-null  object
 12  Holiday/Promotion   73100 non-null  int64
 13  Competitor Pricing  73100 non-null  float64
 14  Seasonality         73100 non-null  object
dtypes: float64(3), int64(5), object(7)
memory usage: 8.4+ MB

       Inventory Level    Units Sold  Units Ordered  Demand Forecast
Price      Discount  Holiday/Promotion  \
count     73100.000000  73100.000000   73100.000000     73100.000000
73100.000000  73100.000000       73100.000000
mean        274.469877    136.464870     110.004473       141.494720
55.135108     10.009508           0.497305
std         129.949514    108.919406      52.277448       109.254076
26.021945      7.083746           0.499996
min          50.000000      0.000000      20.000000        -9.990000
10.000000      0.000000           0.000000
25%         162.000000     49.000000      65.000000        53.670000
32.650000      5.000000           0.000000
50%         273.000000    107.000000     110.000000       113.015000
55.050000     10.000000           0.000000
75%         387.000000    203.000000     155.000000       208.052500
77.860000     15.000000           1.000000
max         500.000000    499.000000     200.000000       518.550000
100.000000     20.000000           1.000000

       Competitor Pricing
count        73100.000000
mean            55.146077
std             26.191408
min              5.030000
25%             32.680000
50%             55.010000
75%             77.820000
max            104.940000
```

Date Handling & Time-Based Feature Engineering

```
# Convert Date column to datetime
df['Date'] = pd.to_datetime(df['Date'])

# Verify conversion
df.dtypes
```

```
Date                      datetime64[ns]
Store ID                          object
Product ID                        object
Category                          object
Region                            object
Inventory Level                    int64
Units Sold                         int64
Units Ordered                      int64
Demand Forecast                  float64
Price                            float64
Discount                           int64
Weather Condition                 object
Holiday/Promotion                  int64
Competitor Pricing               float64
Seasonality                       object
dtype: object
```

Sorting Data by Date for Time-series analysus

```
# Sort dataset by date
df = df.sort_values('Date').reset_index(drop=True)

# Check earliest and latest dates
df['Date'].min(), df['Date'].max()
```

```
(Timestamp('2022-01-01 00:00:00'), Timestamp('2024-01-01 00:00:00'))
```

Create Time-Based Features

```
# Create time-based features
df['Year'] = df['Date'].dt.year
df['Month'] = df['Date'].dt.month
df['Month_Name'] = df['Date'].dt.month_name()
df['Weekday'] = df['Date'].dt.day_name()
```

Quick Check

```
df[['Date', 'Year', 'Month', 'Month_Name', 'Weekday']].head()
```

```
        Date  Year  Month Month_Name   Weekday
0 2022-01-01  2022      1    January  Saturday
1 2022-01-01  2022      1    January  Saturday
2 2022-01-01  2022      1    January  Saturday
```

```
3 2022-01-01  2022      1    January  Saturday
4 2022-01-01  2022      1    January  Saturday
```

1Q.) How does daily sales volume change over time across different product categories?

```
# Aggregate daily sales by category
category_sales = (
    df.groupby(['Date', 'Category'])['Units Sold']
      .sum()
      .reset_index()
)

category_sales.head()
```

```
        Date     Category  Units Sold
0 2022-01-01     Clothing        3784
1 2022-01-01  Electronics        3440
2 2022-01-01    Furniture        1738
3 2022-01-01    Groceries        3112
4 2022-01-01         Toys        2410
```

```
# Create Year-Month column
category_sales['Year_Month'] = 
category_sales['Date'].dt.to_period('M')

# Aggregate monthly sales
monthly_category_sales = (
    category_sales.groupby(['Year_Month', 'Category'])['Units Sold']
    .sum()
    .reset_index()
)

# Convert back to timestamp for plotting
monthly_category_sales['Year_Month'] = 
monthly_category_sales['Year_Month'].dt.to_timestamp()

monthly_category_sales.head()
```

```
  Year_Month     Category  Units Sold
0 2022-01-01     Clothing       77477
1 2022-01-01  Electronics       90499
2 2022-01-01    Furniture       87502
3 2022-01-01    Groceries       82139
4 2022-01-01         Toys       82321
```

```
# Count number of days per month
days_per_month = (
    df.groupby(df['Date'].dt.to_period('M'))['Date']
      .nunique()
)
```

```
days_per_month
```

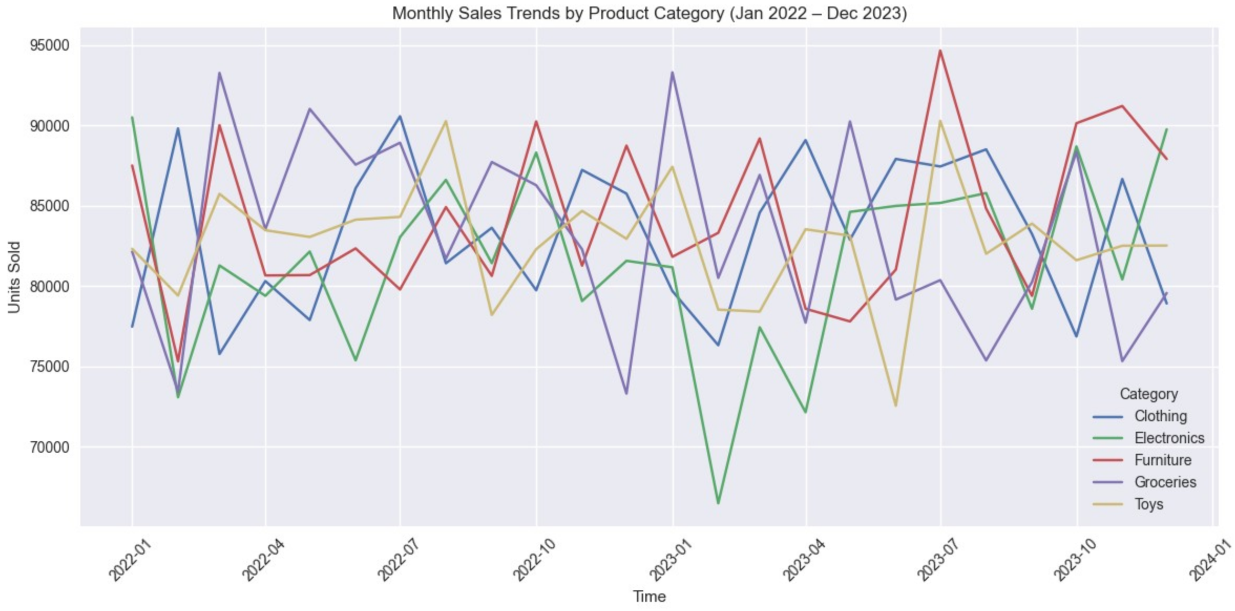
```
Date
2022-01    31
2022-02    28
2022-03    31
2022-04    30
2022-05    31
2022-06    30
2022-07    31
2022-08    31
2022-09    30
2022-10    31
2022-11    30
2022-12    31
2023-01    31
2023-02    28
2023-03    31
2023-04    30
2023-05    31
2023-06    30
2023-07    31
2023-08    31
2023-09    30
2023-10    31
2023-11    30
2023-12    31
2024-01     1
Freq: M, Name: Date, dtype: int64
```

```
# Keep data only till December 2023
df_clean = df[df['Date'] <= '2023-12-31']

# Verify date range
df_clean['Date'].min(), df_clean['Date'].max()
```

```
(Timestamp('2022-01-01 00:00:00'), Timestamp('2023-12-31 00:00:00'))
```

```
# Aggregate daily sales by category
category_sales_clean = (
    df_clean.groupby(['Date', 'Category'])['Units Sold']
    .sum()
    .reset_index()
)

# Create Year-Month
category_sales_clean['Year_Month'] =
category_sales_clean['Date'].dt.to_period('M')

# Monthly aggregation
```

```python
monthly_category_sales_clean = (
    category_sales_clean.groupby(['Year_Month', 'Category'])['Units Sold']
    .sum()
    .reset_index()
)

# Convert to timestamp
monthly_category_sales_clean['Year_Month'] = (
    monthly_category_sales_clean['Year_Month'].dt.to_timestamp()
)

plt.figure()

sns.lineplot(
    data=monthly_category_sales_clean,
    x='Year_Month',
    y='Units Sold',
    hue='Category'
)

plt.title('Monthly Sales Trends by Product Category (Jan 2022 – Dec 2023)')
plt.xlabel('Time')
plt.ylabel('Units Sold')
plt.xticks(rotation=45)
plt.tight_layout()
plt.show()
```



```
# Aggregate sales by Year and Category
yearly_category_sales = (
    df_clean.groupby(['Year', 'Category'])['Units Sold']
    .sum()
    .reset_index()
)

# Pivot for stacked bar chart
pivot_year_category = yearly_category_sales.pivot(
    index='Year',
    columns='Category',
    values='Units Sold'
)

pivot_year_category
```

```
Category  Clothing  Electronics  Furniture  Groceries     Toys
Year
2022        995680       981838    1002118    1011162  1000826
2023       1002096       975260    1019846     987093   986419
```

which product categories contribute the most to total sales volume?

```
# Sort categories by total sales
category_total_sales = category_total_sales.sort_values(
    by='Units Sold',
    ascending=False
)

category_total_sales
```

```
      Category  Units Sold
2    Furniture     2021964
3    Groceries     1998255
0     Clothing     1997776
4         Toys     1987245
1  Electronics     1957098
```

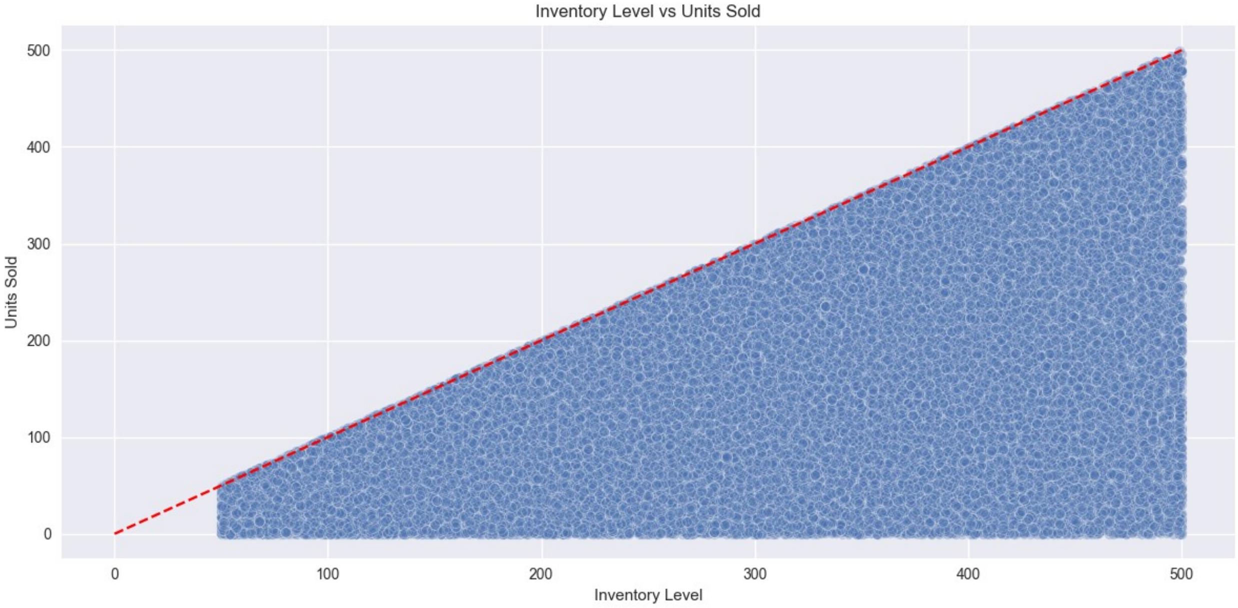
QUESTION 3: Inventory Level vs Units Sold

```
# Select relevant columns
inventory_sales = df_clean[['Inventory Level', 'Units Sold']]

inventory_sales.head()
```

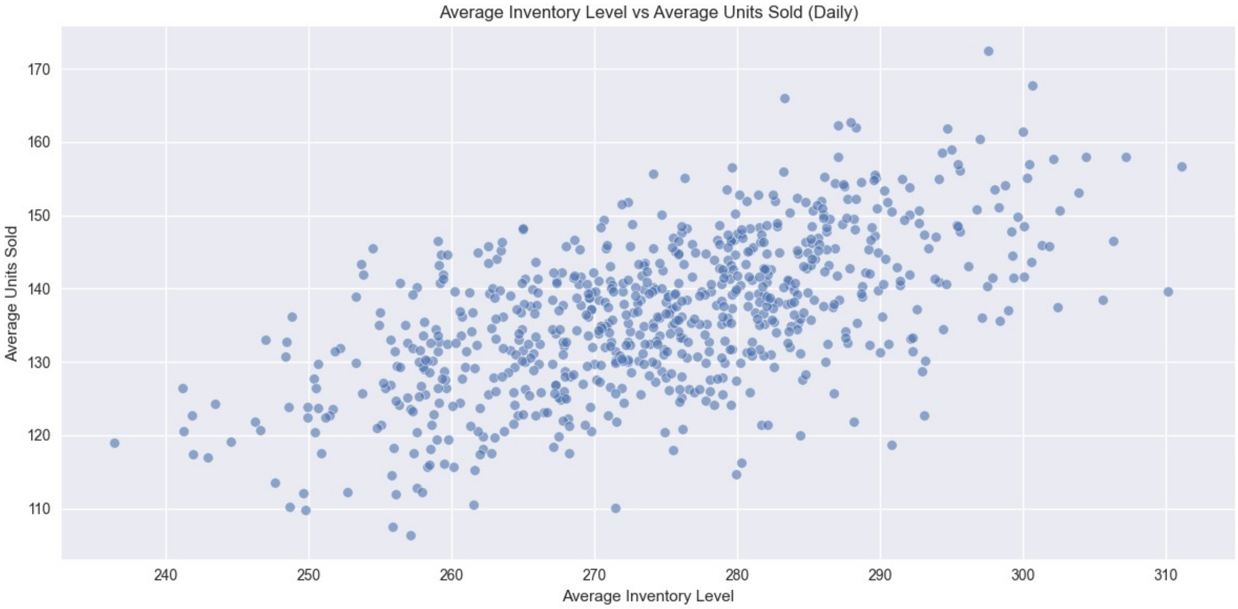
```
   Inventory Level  Units Sold
0              231         127
1              191          56
2              349           9
```

```
3                205          46
4                447         104
```

```python
plt.figure()

sns.scatterplot(
    data=inventory_sales,
    x='Inventory Level',
    y='Units Sold',
    alpha=0.3
)

# Reference line: Units Sold = Inventory Level
max_val = inventory_sales['Inventory Level'].max()
plt.plot([0, max_val], [0, max_val], linestyle='--', color='red')

plt.title('Inventory Level vs Units Sold')
plt.xlabel('Inventory Level')
plt.ylabel('Units Sold')
plt.tight_layout()
plt.show()
```



```python
inventory_daily = (
    df_clean.groupby('Date')[['Inventory Level', 'Units Sold']]
    .mean()
    .reset_index()
)

plt.figure()

sns.scatterplot(
    data=inventory_daily,
    x='Inventory Level',
    y='Units Sold',
    alpha=0.6
)

plt.title('Average Inventory Level vs Average Units Sold (Daily)')
plt.xlabel('Average Inventory Level')
plt.ylabel('Average Units Sold')
plt.tight_layout()
plt.show()
```



QUESTION 4:Which products experience the highest frequency of stockouts?

```
# Identify stockout risk situations (sold almost all inventory)
stockouts = df_clean[df_clean['Units Sold'] >= df_clean['Inventory
Level'] * 0.95]

stockouts.head()

          Date Store ID Product ID     Category Region  Inventory
Level  Units Sold  Units Ordered  Demand Forecast  \
29  2022-01-01     S005      P0017  Electronics   East
388         371             98          390.04
40  2022-01-01     S005      P0006    Groceries  North
185         176            163          174.06
60  2022-01-01     S001      P0015     Clothing  North
379         369            154          363.46
65  2022-01-01     S001      P0009  Electronics   West
183         175            135          174.15
171 2022-01-02     S001      P0003     Clothing  South
```

```
488         464              163           463.12

     Price  Discount Weather Condition  Holiday/Promotion  Competitor
Pricing Seasonality  Year  Month Month_Name  \
29   98.31        15             Sunny                  1
102.42      Spring  2022      1    January
40   60.76        20            Cloudy                  0
61.15      Summer  2022      1    January
60   92.99        15             Snowy                  0
95.80      Winter  2022      1    January
65   20.74        10            Cloudy                  0
17.66      Autumn  2022      1    January
171  70.99        10             Snowy                  0
72.93      Summer  2022      1    January

      Weekday
29   Saturday
40   Saturday
60   Saturday
65   Saturday
171    Sunday
```

```
product_stockouts = (
    stockouts.groupby('Product ID')
    .size()
    .reset_index(name='Stockout Risk Count')
    .sort_values(by='Stockout Risk Count', ascending=False)
)

product_stockouts.head(10)
```

```
   Product ID  Stockout Risk Count
19      P0020                  224
18      P0019                  206
6       P0007                  200
0       P0001                  198
9       P0010                  195
4       P0005                  191
3       P0004                  191
7       P0008                  190
8       P0009                  189
12      P0013                  188
```

```
# Count stockouts by Product ID
product_stockouts = (
    stockouts.groupby('Product ID')
    .size()
    .reset_index(name='Stockout Count')
    .sort_values(by='Stockout Count', ascending=False)
)
```

```
product_stockouts.head(10)
```

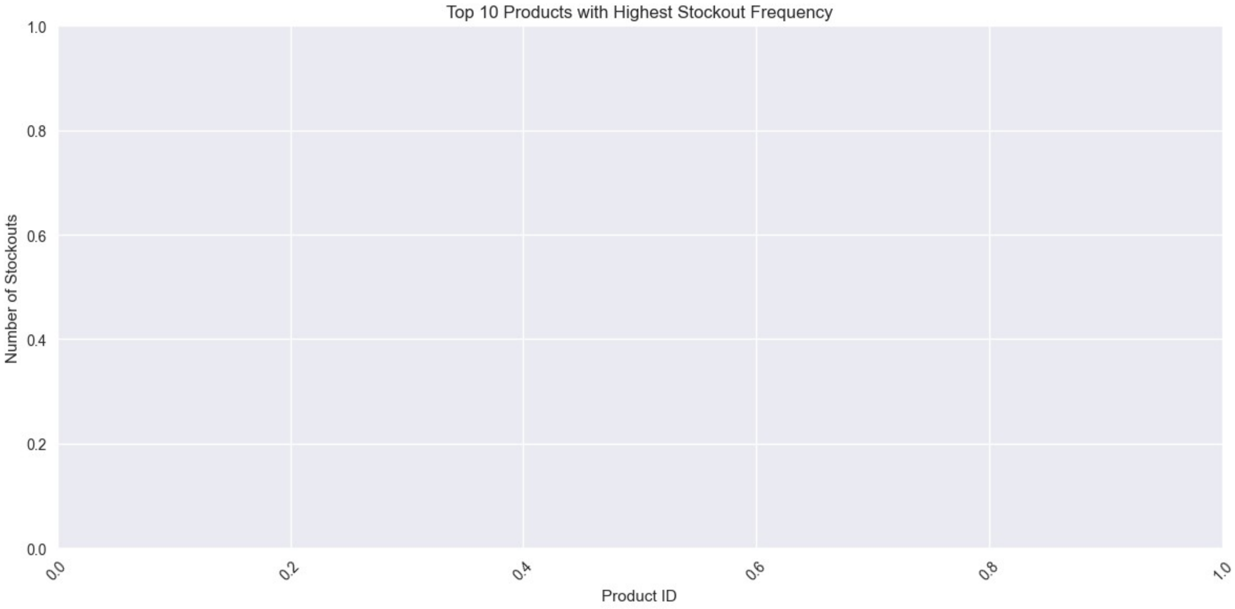
```
Empty DataFrame
Columns: [Product ID, Stockout Count]
Index: []
```

```
# Select top 10 products with most stockouts
top_stockout_products = product_stockouts.head(10)
```

```
top_stockout_products
```

```
Empty DataFrame
Columns: [Product ID, Stockout Count]
Index: []
```

```
plt.figure()

sns.barplot(
    data=top_stockout_products,
    x='Product ID',
    y='Stockout Count'
)

plt.title('Top 10 Products with Highest Stockout Frequency')
plt.xlabel('Product ID')
plt.ylabel('Number of Stockouts')
plt.xticks(rotation=45)
plt.tight_layout()
plt.show()
```



```
df_clean['Inventory Level'].min()
```

```
np.int64(50)
```

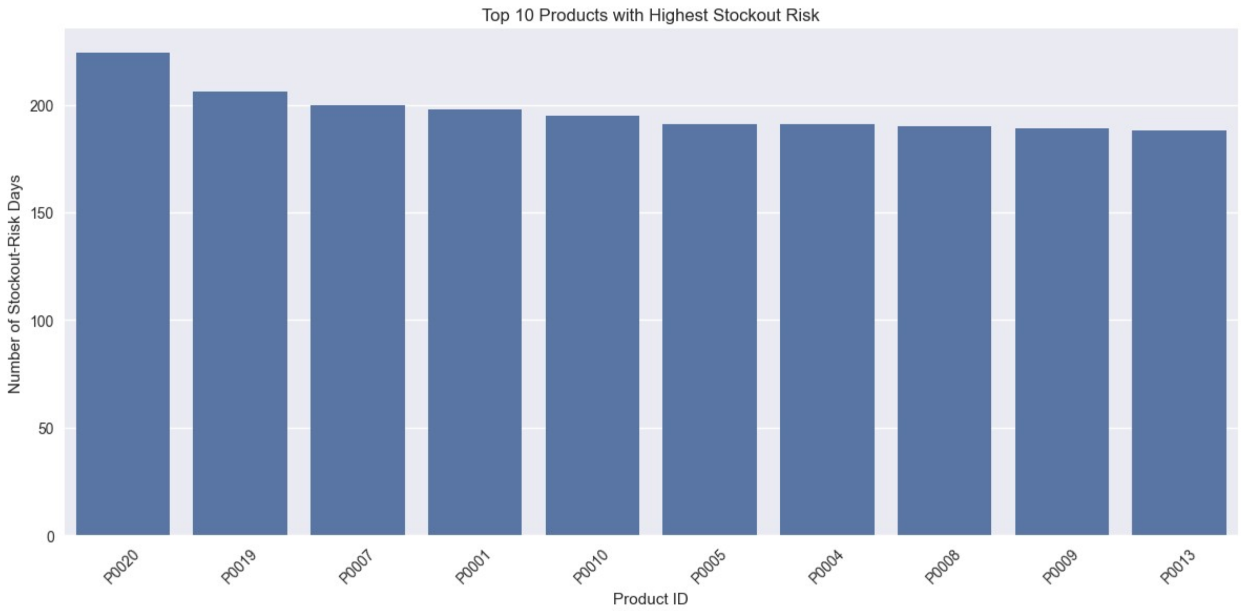
```
# Identify stockout risk situations (sold almost all inventory)
stockouts = df_clean[df_clean['Units Sold'] >= df_clean['Inventory
Level'] * 0.95]

stockouts.head()
```

```
          Date Store ID Product ID     Category Region  Inventory
Level  Units Sold  Units Ordered  Demand Forecast  \
29  2022-01-01     S005      P0017  Electronics   East
388         371             98           390.04
40  2022-01-01     S005      P0006    Groceries  North
185         176            163           174.06
60  2022-01-01     S001      P0015     Clothing  North
379         369            154           363.46
65  2022-01-01     S001      P0009  Electronics   West
183         175            135           174.15
171 2022-01-02     S001      P0003     Clothing  South
488         464            163           463.12

     Price  Discount Weather Condition  Holiday/Promotion  Competitor
Pricing Seasonality  Year  Month Month_Name  \
29   98.31        15             Sunny                  1
102.42      Spring  2022      1    January
40   60.76        20            Cloudy                  0
61.15       Summer  2022     1    January
60   92.99        15             Snowy                  0
95.80       Winter  2022     1    January
65   20.74        10            Cloudy                  0
17.66       Autumn  2022     1    January
171  70.99        10             Snowy                  0
72.93       Summer  2022     1    January

      Weekday
29   Saturday
40   Saturday
60   Saturday
65   Saturday
171    Sunday
```

```
product_stockouts = (
    stockouts.groupby('Product ID')
    .size()
    .reset_index(name='Stockout Risk Count')
    .sort_values(by='Stockout Risk Count', ascending=False)
)

product_stockouts.head(10)
```

```
   Product ID  Stockout Risk Count
19      P0020                  224
18      P0019                  206
6       P0007                  200
0       P0001                  198
9       P0010                  195
4       P0005                  191
3       P0004                  191
7       P0008                  190
8       P0009                  189
12      P0013                  188
```

```
plt.figure()

sns.barplot(
    data=top_stockout_products,
    x='Product ID',
    y='Stockout Risk Count'
)

plt.title('Top 10 Products with Highest Stockout Risk')
plt.xlabel('Product ID')
plt.ylabel('Number of Stockout-Risk Days')
plt.xticks(rotation=45)
plt.tight_layout()
plt.show()
```



Question 5: Which products are most frequently overstocked?

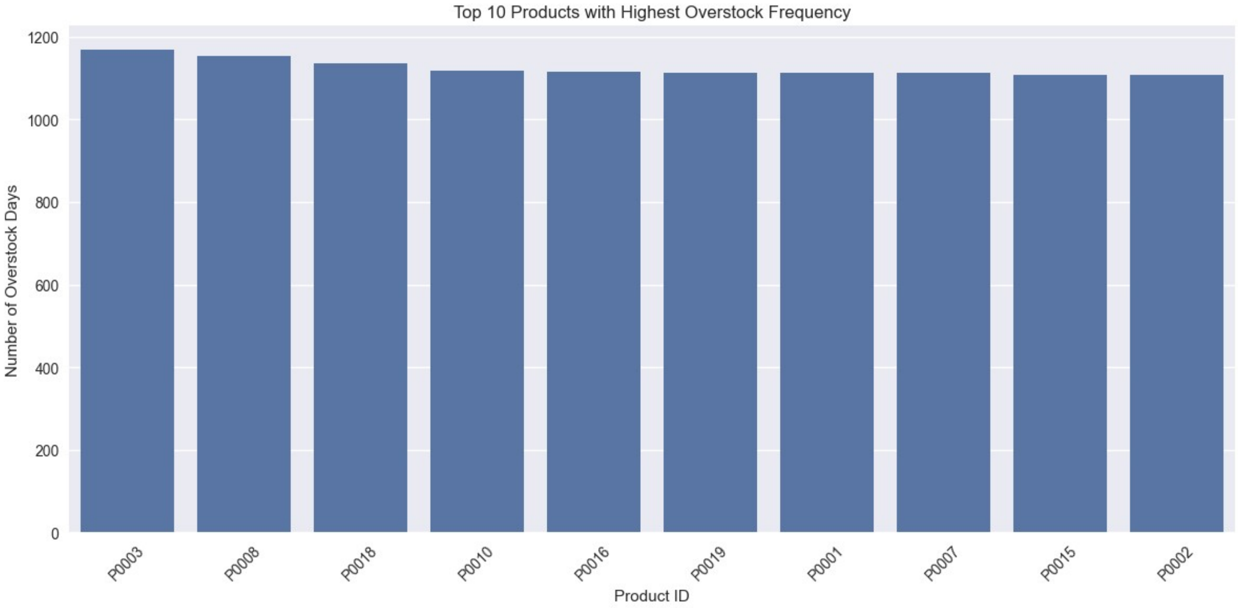
```
# Define overstock condition
overstock = df_clean[df_clean['Units Sold'] <= df_clean['Inventory
Level'] * 0.30]

overstock.head()

        Date Store ID Product ID     Category Region  Inventory Level
Units Sold  Units Ordered  Demand Forecast  \
1 2022-01-01     S004       P0013    Furniture    East              191
56             65              54.47
2 2022-01-01     S004       P0012  Electronics   North              349
9            165               0.95
3 2022-01-01     S004       P0011  Electronics    West              205
46             27              46.65
4 2022-01-01     S004       P0010    Groceries    East              447
104             96             115.03
6 2022-01-01     S004       P0008    Furniture   South              250
51            137              54.98

   Price  Discount Weather Condition  Holiday/Promotion  Competitor
Pricing Seasonality  Year  Month Month_Name  \
1  61.81         0             Sunny                  0
63.92      Autumn  2022      1    January
2  14.25         5             Rainy                  1
18.56      Spring  2022      1    January
3  54.84         0             Sunny                  1
57.76      Spring  2022      1    January
4  33.48        15            Cloudy                  0
37.15      Summer  2022      1    January
6  85.88        20             Snowy                  1
86.14      Winter  2022      1    January

    Weekday
1  Saturday
2  Saturday
3  Saturday
4  Saturday
6  Saturday

product_overstock = (
    overstock.groupby('Product ID')
    .size()
    .reset_index(name='Overstock Count')
    .sort_values(by='Overstock Count', ascending=False)
)

product_overstock.head(10)

   Product ID  Overstock Count
2       P0003             1168
7       P0008             1153
17      P0018             1135
```

```
9       P0010             1119
15      P0016             1116
18      P0019             1114
0       P0001             1112
6       P0007             1112
14      P0015             1109
1       P0002             1108
```

```
top_overstock_products = product_overstock.head(10)
top_overstock_products
```

```
   Product ID  Overstock Count
2       P0003             1168
7       P0008             1153
17      P0018             1135
9       P0010             1119
15      P0016             1116
18      P0019             1114
0       P0001             1112
6       P0007             1112
14      P0015             1109
1       P0002             1108
```

```
plt.figure()

sns.barplot(
    data=top_overstock_products,
    x='Product ID',
    y='Overstock Count'
)

plt.title('Top 10 Products with Highest Overstock Frequency')
plt.xlabel('Product ID')
plt.ylabel('Number of Overstock Days')
plt.xticks(rotation=45)
plt.tight_layout()
plt.show()
```


Top 10 Products with Highest Overstock Frequency
Number of Overstock Days
0
200
400
600
800
1000
1200
P0003
P0008
P0018
P0010
P0016
P0019
P0001
P0007
P0015
P0002
Product ID