loading packages

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

New Section

```
from google.colab import drive
drive.mount('/content/drive')
```

```
Drive already mounted at /content/drive; to attempt to forcibly remount, call drive.mount("/con
```

loading dataset

```
df= pd.read_excel("/content/drive/MyDrive/Medical Inventory Optimization Dataset.xlsx")
```

data preprocessing

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 14218 entries, 0 to 14217
Data columns (total 14 columns):
 #   Column          Non-Null Count  Dtype  
---  ------          --------------  -----  
 0   Typeofsales     14218 non-null  object 
 1   Patient_ID      14218 non-null  int64  
 2   Specialisation  14218 non-null  object 
 3   Dept            14218 non-null  object 
 4   Dateofbill      14218 non-null  object 
 5   Quantity        14218 non-null  int64  
 6   ReturnQuantity  14218 non-null  int64  
 7   Final_Cost      14218 non-null  float64
 8   Final_Sales     14218 non-null  float64
 9   RtnMRP          14218 non-null  float64
 10  Formulation     13565 non-null  object 
 11  DrugName        12550 non-null  object 
 12  SubCat          12550 non-null  object 
 13  SubCat1         12526 non-null  object 
dtypes: float64(3), int64(3), object(8)
memory usage: 1.5+ MB
```

```
df.head()
```

	Typeofsales	Patient_ID	Specialisation	Dept	Dateofbill	Quantity	ReturnQuantity
0	Sale	12018098765	Specialisation6	Department1	2022-01-06 00:00:00	1	0
1	Sale	12018103897	Specialisation7	Department1	7/23/2022	1	0
2	Sale	12018101123	Specialisation2	Department3	6/23/2022	1	0
3	Sale	12018079281	Specialisation40	Department1	3/17/2022	2	0
4	Sale	12018117928	Specialisation5	Department1	12/21/2022	1	0

Next steps: Generate code with df | View recommended plots | New interactive sheet

```
#display column names
df_columns =df.columns
print(df_columns)
```

```
Index(['Typeofsales', 'Patient_ID', 'Specialisation', 'Dept', 'Dateofbill',
       'Quantity', 'ReturnQuantity', 'Final_Cost', 'Final_Sales', 'RtnMRP',
       'Formulation', 'DrugName', 'SubCat', 'SubCat1'],
      dtype='object')
```

```
#display shape of the data
df.shape
```

```
(14218, 14)
```

```
df.describe()
```

	Patient_ID	Quantity	ReturnQuantity	Final_Cost	Final_Sales	RtnMRP
count	1.421800e+04	14218.000000	14218.000000	14218.000000	14218.000000	14218.000000
mean	1.201809e+10	2.231748	0.291954	124.823957	234.038300	29.126755
std	2.810229e+04	5.132043	1.643322	464.782794	671.261572	182.262335
min	1.201800e+10	0.000000	0.000000	40.000000	0.000000	0.000000
25%	1.201808e+10	1.000000	0.000000	44.928000	47.815000	0.000000
50%	1.201809e+10	1.000000	0.000000	53.650000	86.424000	0.000000
75%	1.201811e+10	2.000000	0.000000	77.800000	181.000000	0.000000
max	1.201813e+10	150.000000	50.000000	33178.000000	39490.000000	8014.000000

HANDLING MISSING VALUES

```
df.isnull().sum()
```

	0
Typeofsales	0
Patient_ID	0
Specialisation	0
Dept	0
Dateofbill	0
Quantity	0
ReturnQuantity	0
Final_Cost	0
Final_Sales	0
RtnMRP	0
Formulation	653
DrugName	1668
SubCat	1668
SubCat1	1692

```
#display data types
df.dtypes
```

	0
Typeofsales	object
Patient_ID	int64
Specialisation	object
Dept	object
Dateofbill	object
Quantity	int64
ReturnQuantity	int64
Final_Cost	float64
Final_Sales	float64
RtnMRP	float64
Formulation	object
DrugName	object
SubCat	object
SubCat1	object

```
#mode imputation
categorical_vars = ['Formulation', 'DrugName', 'SubCat', 'SubCat1']
for col in categorical_vars:
    mode_val = df[col].mode()[0]  # Calculate mode
    df[col].fillna(mode_val, inplace=True)
```

```
<ipython-input-20-6472e8fca314>:5: FutureWarning: A value is trying to be set on a copy of a Da
The behavior will change in pandas 3.0. This inplace method will never work because the interme

For example, when doing 'df[col].method(value, inplace=True)', try using 'df.method({col: value


  df[col].fillna(mode_val, inplace=True)
```

```
df.isnull().sum()
```

	0
Typeofsales	0
Patient_ID	0
Specialisation	0
Dept	0
Dateofbill	0
Quantity	0
ReturnQuantity	0
Final_Cost	0
Final_Sales	0
RtnMRP	0
Formulation	0
DrugName	0
SubCat	0
SubCat1	0

handling duplicates


```
Start coding or generate with AI.
```

```
# Check for duplicates
num_duplicates = df.duplicated().sum()

# Display the number of duplicates
print("Number of duplicates:", num_duplicates)
```

```
Number of duplicates: 26
```

```
#display the duplicate rows
duplicate_rows = df[df.duplicated()]
print("Duplicate rows:")
print(duplicate_rows)
```

```
13966  2022-10-12 00:00:00         1               0      49.956       62.800
14204  2022-12-04 00:00:00         0               2      43.176        0.000

       RtnMRP Formulation                           DrugName  \
2706    0.000       Form1    LIPOSOMAL AMPHOTERICIN B 50MG INJ
4444    0.000       Form1                   ESOMEPRAZOLE 40MG
4509    0.000       Form2                   MEROPENEM 1GM INJ
5539    0.000       Form1                     CEFTRIAXONE 1GM
6254    0.000       Form1  PIPERACILLIN 1GM + TAZOBACTAM 125MG
6499    0.000       Form1             DEXTROSE 5% W/V IV FLUID
6731   91.600       Form1            SODIUM CHLORIDE IVF 100ML
7664    0.000       Form1            SODIUM CHLORIDE IVF 100ML
7950    0.000       Form1            SODIUM CHLORIDE IVF 100ML
7968    0.000       Form1        NUTRITIONAL SUPPLEMENT POWDER
8661    0.000       Form1            SODIUM CHLORIDE IVF 100ML
8973    0.000       Form1            SODIUM CHLORIDE IVF 100ML
9470   57.568       Form1                 SODIUM CHLORIDE 0.9%
9643    0.000       Form1            SODIUM CHLORIDE IVF 100ML
10990   0.000       Form1                HUMAN ALBUMIN 25% INJ
11590   0.000       Form1                 DAPTOMYCIN 350MG INJ
12923   0.000       Form1                MINOCYCLINE INJ 100MG
12925   0.000       Form1            SODIUM CHLORIDE IVF 100ML
13008   0.000       Form1           MAGNESIUM SULPHATE 2ML INJ
13139  63.944       Form2                 SODIUM CHLORIDE 0.45%
13361   0.000       Form1                HUMAN ALBUMIN 25% INJ
13576   0.000       Form1                   ONDANSETRON 2MG/ML
13748   0.000       Form1                    CALCIUM GLUCONATE
13963  60.800       Form1            SODIUM CHLORIDE IVF 100ML
13966   0.000       Form1            SODIUM CHLORIDE IVF 100ML
14204  85.960       Form1                     PARACETAMOL 150MG

                            SubCat                                  SubCat1
2706                     INJECTIONS                        ANTI-INFECTIVES
4444                     INJECTIONS  GASTROINTESTINAL & HEPATOBILIARY SYSTEM
4509                     INJECTIONS                        ANTI-INFECTIVES
5539                     INJECTIONS                        ANTI-INFECTIVES
6254                     INJECTIONS                        ANTI-INFECTIVES
6499   IV FLUIDS, ELECTROLYTES, TPN    INTRAVENOUS & OTHER STERILE SOLUTIONS
6731                     INJECTIONS    INTRAVENOUS & OTHER STERILE SOLUTIONS
7664                     INJECTIONS    INTRAVENOUS & OTHER STERILE SOLUTIONS
7950                     INJECTIONS    INTRAVENOUS & OTHER STERILE SOLUTIONS
7968          NUTRITIONAL SUPPLEMENTS    INTRAVENOUS & OTHER STERILE SOLUTIONS
8661                     INJECTIONS    INTRAVENOUS & OTHER STERILE SOLUTIONS
8973                     INJECTIONS    INTRAVENOUS & OTHER STERILE SOLUTIONS
9470   IV FLUIDS, ELECTROLYTES, TPN    INTRAVENOUS & OTHER STERILE SOLUTIONS
9643                     INJECTIONS    INTRAVENOUS & OTHER STERILE SOLUTIONS
10990  IV FLUIDS, ELECTROLYTES, TPN    INTRAVENOUS & OTHER STERILE SOLUTIONS
11590                    INJECTIONS                        ANTI-INFECTIVES
12923                    INJECTIONS                        ANTI-INFECTIVES
12925                    INJECTIONS    INTRAVENOUS & OTHER STERILE SOLUTIONS
13008                    INJECTIONS                 VITAMINS & MINERALS
13139  IV FLUIDS, ELECTROLYTES, TPN    INTRAVENOUS & OTHER STERILE SOLUTIONS
13361  IV FLUIDS, ELECTROLYTES, TPN    INTRAVENOUS & OTHER STERILE SOLUTIONS
13576                    INJECTIONS  GASTROINTESTINAL & HEPATOBILIARY SYSTEM
13748                    INJECTIONS    INTRAVENOUS & OTHER STERILE SOLUTIONS
13963                    INJECTIONS    INTRAVENOUS & OTHER STERILE SOLUTIONS
13966                    INJECTIONS    INTRAVENOUS & OTHER STERILE SOLUTIONS
14204                    INJECTIONS                 CENTRAL NERVOUS SYSTEM
```

```
# Remove duplicate rows
df_cleaned = df.drop_duplicates()

# Check the shape of the cleaned DataFrame to verify that duplicates were removed
print("Shape of cleaned DataFrame:", df_cleaned.shape)
```

```
Shape of cleaned DataFrame: (14192, 14)
```

SUMMARY STATISTICS

```
# Summary statistics for relevant columns
print(df[['Quantity', 'ReturnQuantity', 'Final_Cost', 'Final_Sales', 'RtnMRP']].describe())
```

```
           Quantity  ReturnQuantity    Final_Cost   Final_Sales        RtnMRP
count  14218.000000    14218.000000  14218.000000  14218.000000  14218.000000
mean       2.231748        0.291954    124.823957    234.038300     29.126755
std        5.132043        1.643322    464.782794    671.261572    182.262335
min        0.000000        0.000000     40.000000      0.000000      0.000000
25%        1.000000        0.000000     44.928000     47.815000      0.000000
50%        1.000000        0.000000     53.650000     86.424000      0.000000
75%        2.000000        0.000000     77.800000    181.000000      0.000000
max      150.000000       50.000000  33178.000000  39490.000000   8014.000000
```

```
df['BounceRate'] = np.where(df['Quantity'] != 0, (df['ReturnQuantity'] / df['Quantity']) * 100, 0)
# Calculate average bounce rate
average_bounce_rate = df['BounceRate'].mean()
print(average_bounce_rate)
```

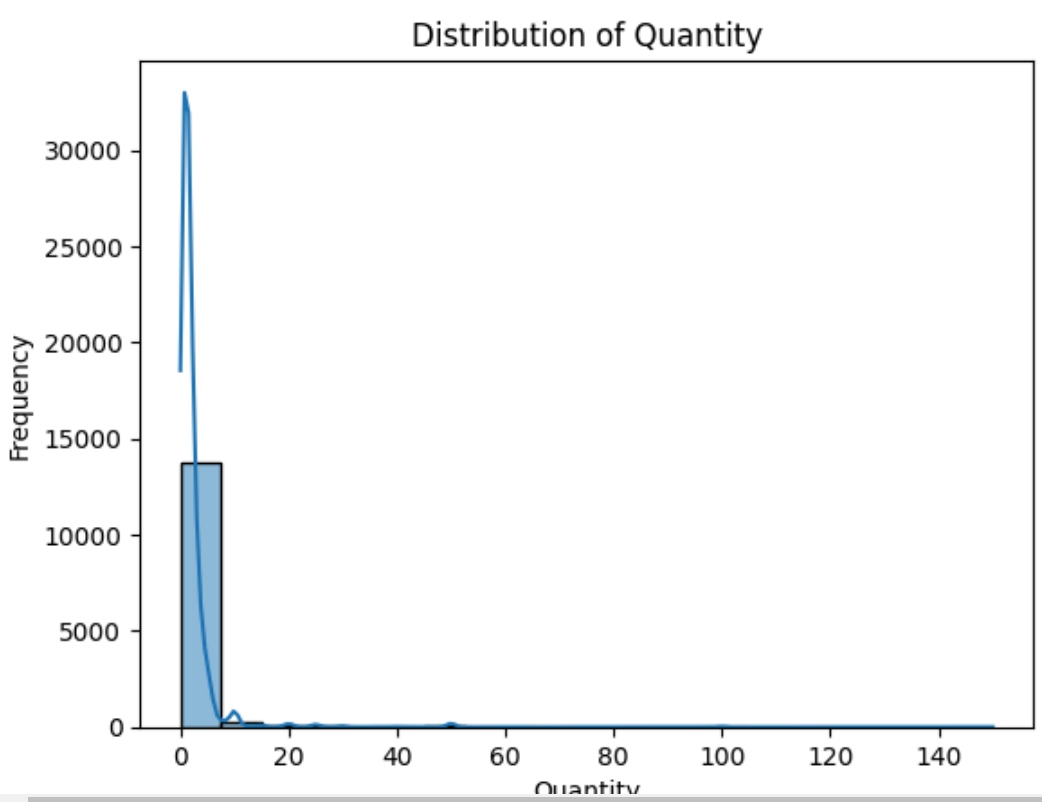
```
0.0
```

```
# Calculate median and variance for multiple columns
columns_of_interest = ['BounceRate', 'Quantity', 'ReturnQuantity', 'Final_Cost', 'Final_Sales', 'RtnMRP']
summary_statistics = df[columns_of_interest].agg(['median', 'var'])
print(summary_statistics)
```

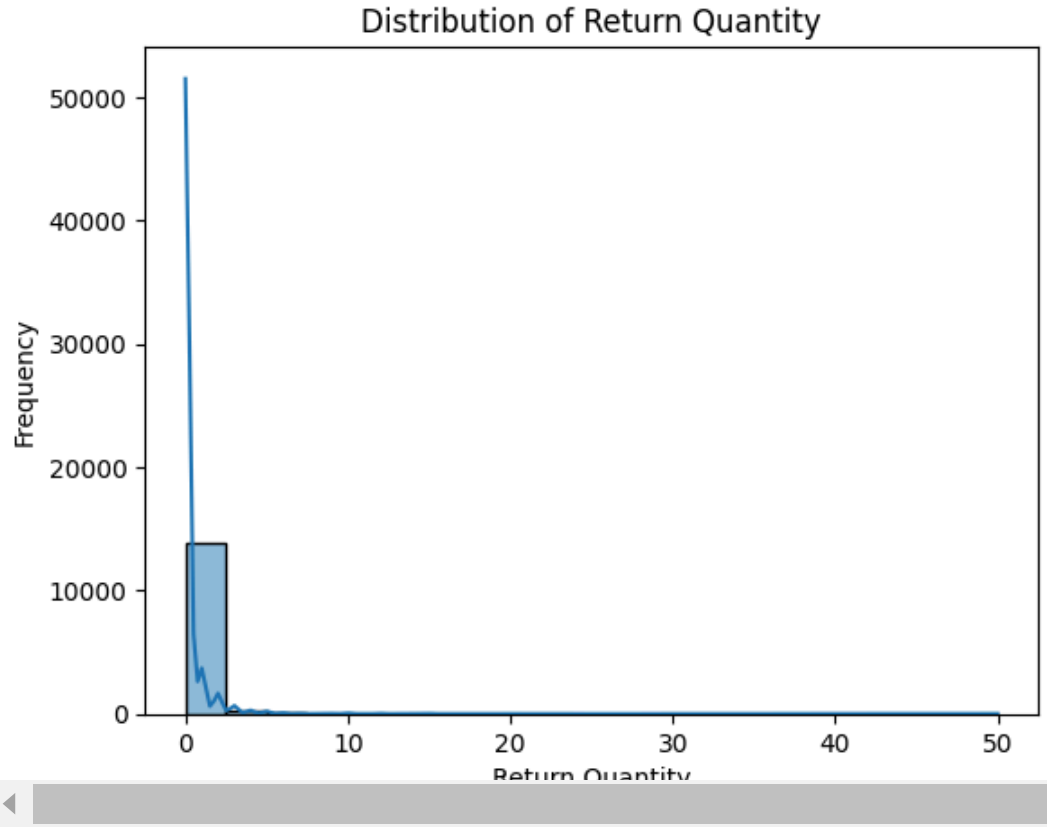
```
        BounceRate   Quantity  ReturnQuantity     Final_Cost    Final_Sales  \
median         0.0   1.000000        0.000000      53.650000      86.424000
var            0.0  26.337862        2.700506  216023.045394  450592.097666

              RtnMRP
median      0.000000
var     33219.558938
```

bars

```
# Distribution visualization for relevant columns
sns.histplot(data=df, x='Quantity', bins=20, kde=True)
plt.title('Distribution of Quantity')
plt.xlabel('Quantity')
plt.ylabel('Frequency')
plt.show()
```

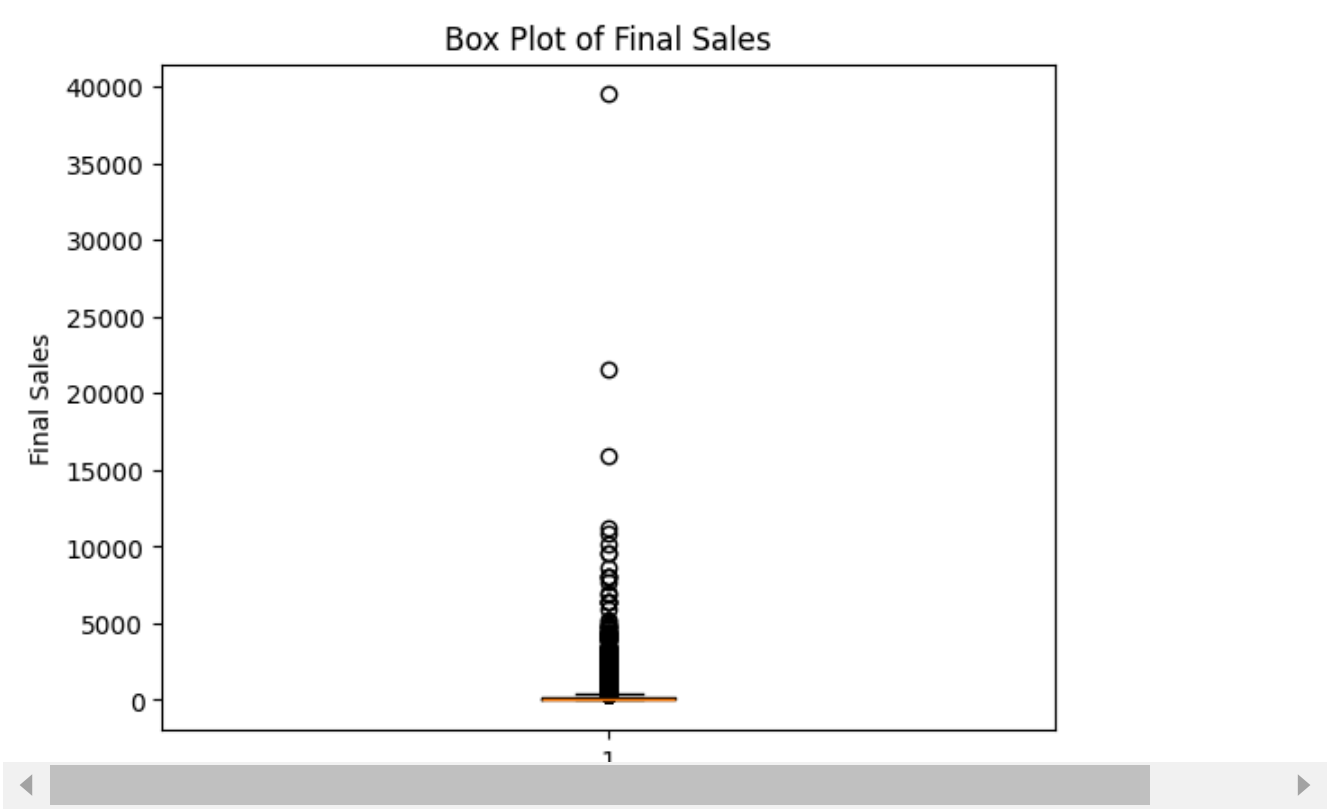



```
sns.histplot(data=df, x='ReturnQuantity', bins=20, kde=True)
plt.title('Distribution of Return Quantity')
plt.xlabel('Return Quantity')
plt.ylabel('Frequency')
plt.show()
```

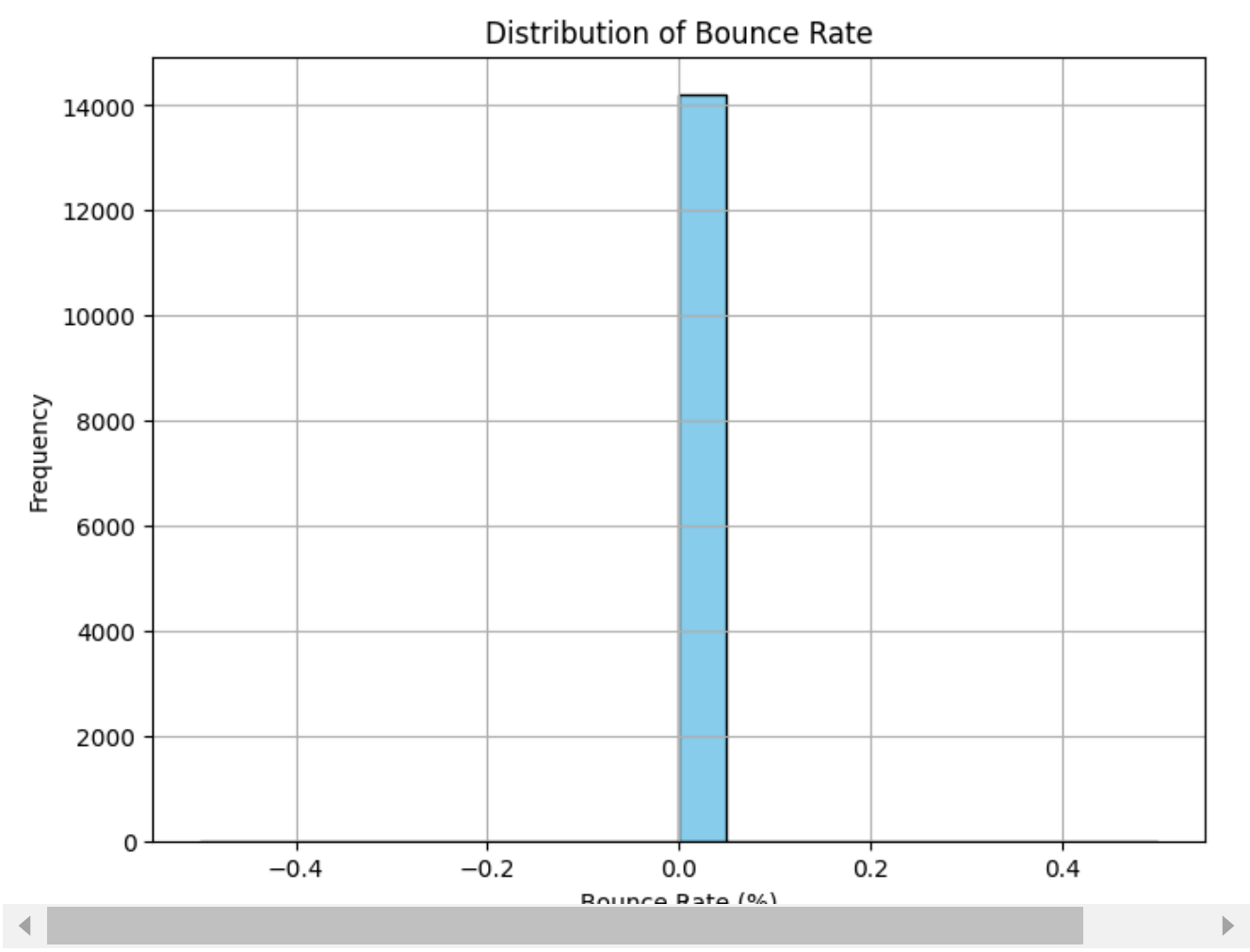



```
# Create a box plot of a numerical variable
plt.boxplot(df['Final_Sales'])
plt.title('Box Plot of Final Sales')
```

```
plt.ylabel('Final Sales')
plt.show()
```




```
# Visualize bounce rate distribution
plt.figure(figsize=(8, 6))
plt.hist(df['BounceRate'], bins=20, color='skyblue', edgecolor='black')
plt.title('Distribution of Bounce Rate')
plt.xlabel('Bounce Rate (%)')
plt.ylabel('Frequency')
plt.grid(True)
plt.show()
```




```
# Group by SubCat and count returned drug names
return_counts_by_subcat = df[df['Typeofsales'] == 'Return'].groupby('SubCat')['DrugName'].count().sort_valu
print("Returned Drug Names by Subcategory:")
print(return_counts_by_subcat)
```

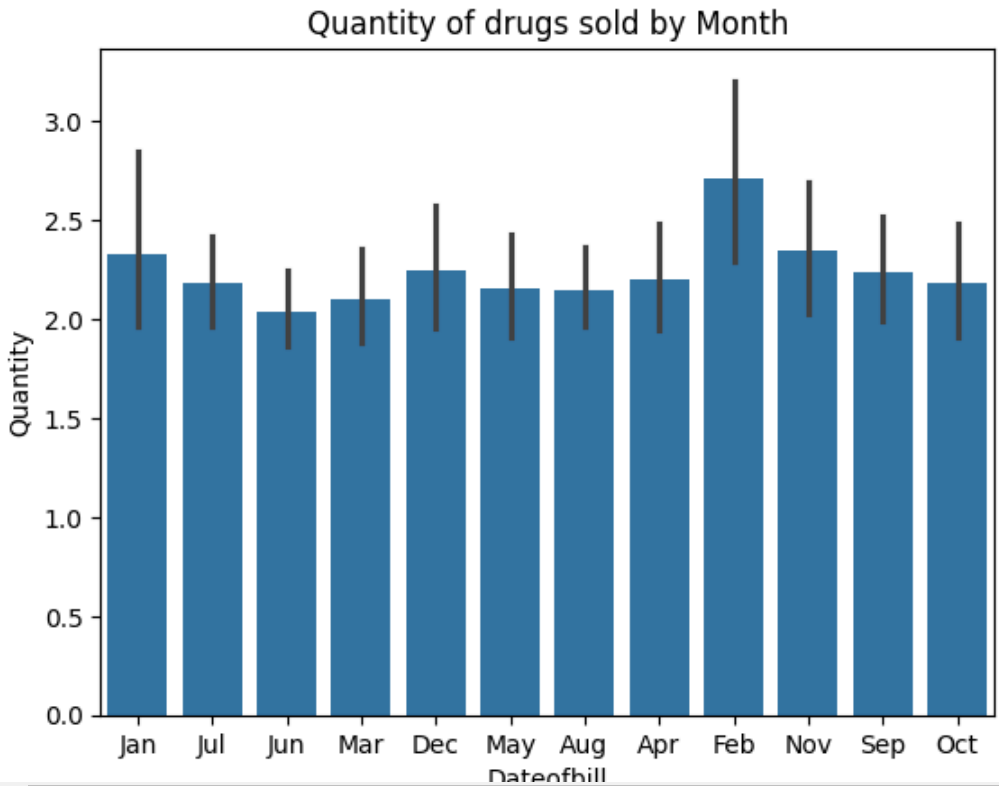
```
Returned Drug Names by Subcategory:
SubCat
INJECTIONS                      926
IV FLUIDS, ELECTROLYTES, TPN    475
TABLETS & CAPSULES               94
INHALERS & RESPULES              71
POWDER                           31
LIQUIDS & SOLUTIONS              24
OINTMENTS, CREAMS & GELS         15
SYRUP & SUSPENSION               15
PESSARIES & SUPPOSITORIES        11
NUTRITIONAL SUPPLEMENTS           8
DROPS                             7
VACCINE                           2
LOTIONS                           1
PATCH                             1
Name: DrugName, dtype: int64
```

```
# Convert date formate to month
df['Dateofbill'] = pd.to_datetime(df['Dateofbill'])
df['Dateofbill'] = df['Dateofbill'].dt.strftime('%b')
df.head()
```

	Typeofsales	Patient_ID	Specialisation	Dept	Dateofbill	Quantity	ReturnQuantity
0	Sale	12018098765	Specialisation6	Department1	Jan	1	0
1	Sale	12018103897	Specialisation7	Department1	Jul	1	0
2	Sale	12018101123	Specialisation2	Department3	Jun	1	0
3	Sale	12018079281	Specialisation40	Department1	Mar	2	0
4	Sale	12018117928	Specialisation5	Department1	Dec	1	0

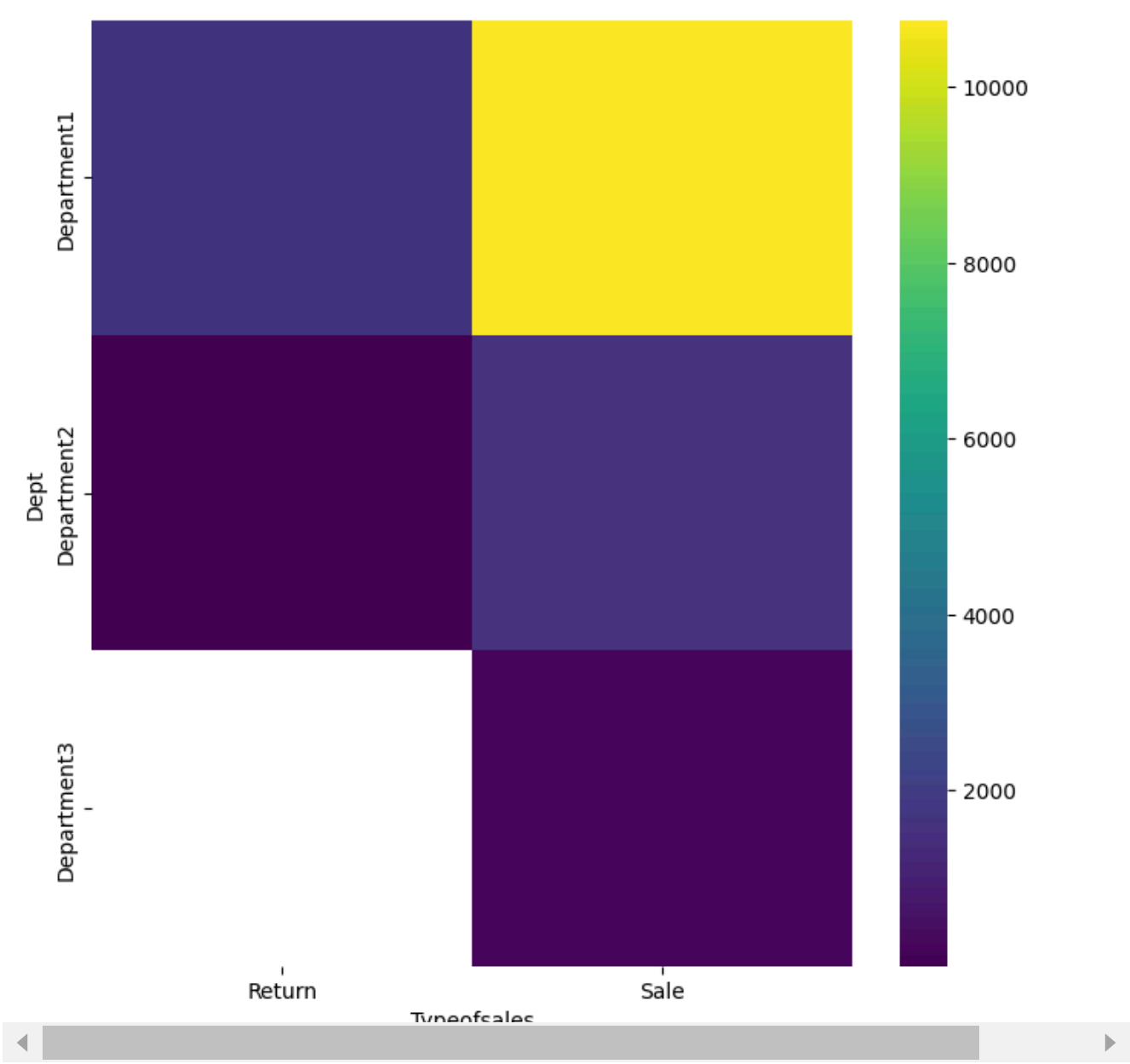
Next steps: Generate code with df | View recommended plots | New interactive sheet

```
sns.barplot(data = df, x = 'Dateofbill', y = 'Quantity')
plt.title('Quantity of drugs sold by Month')
plt.show()
```

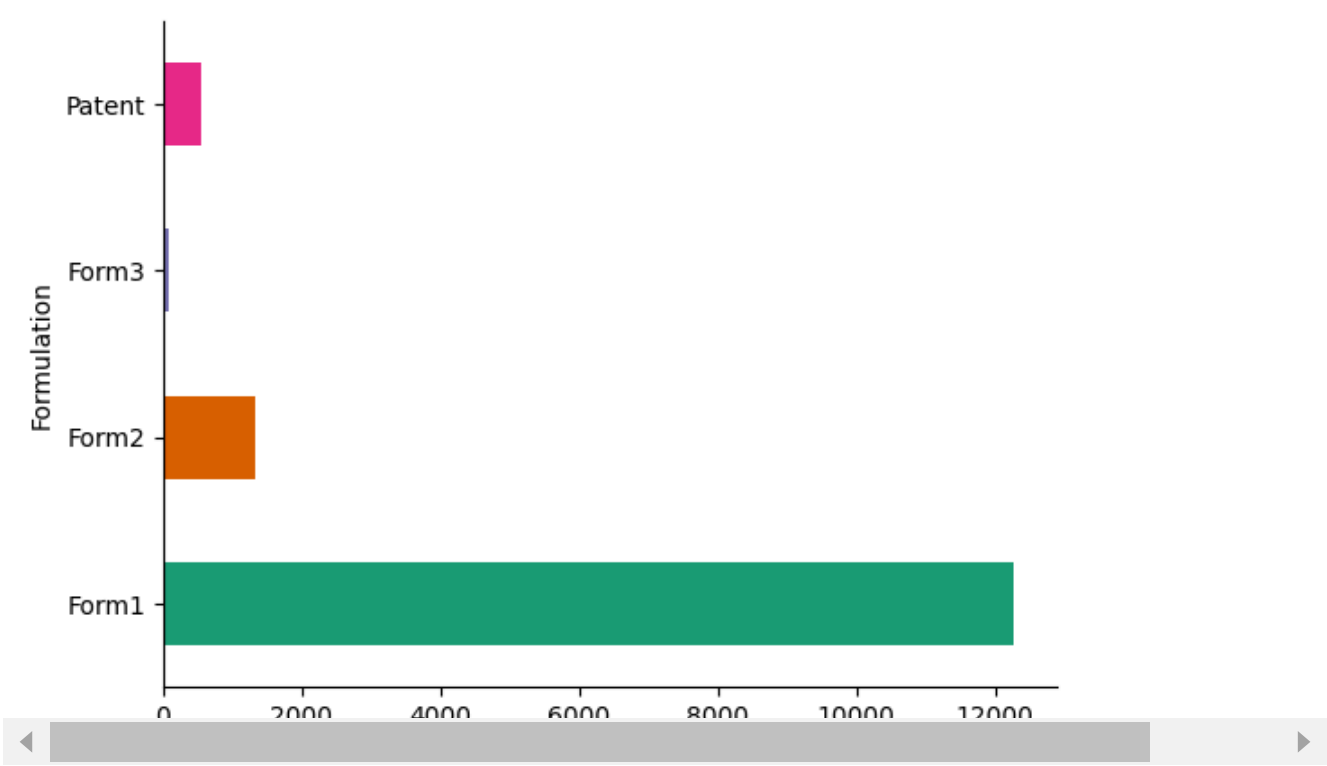



```
#  Typeofsales vs Dept

from matplotlib import pyplot as plt
import seaborn as sns
import pandas as pd
plt.subplots(figsize=(8, 8))
df_2dhist = pd.DataFrame({
```

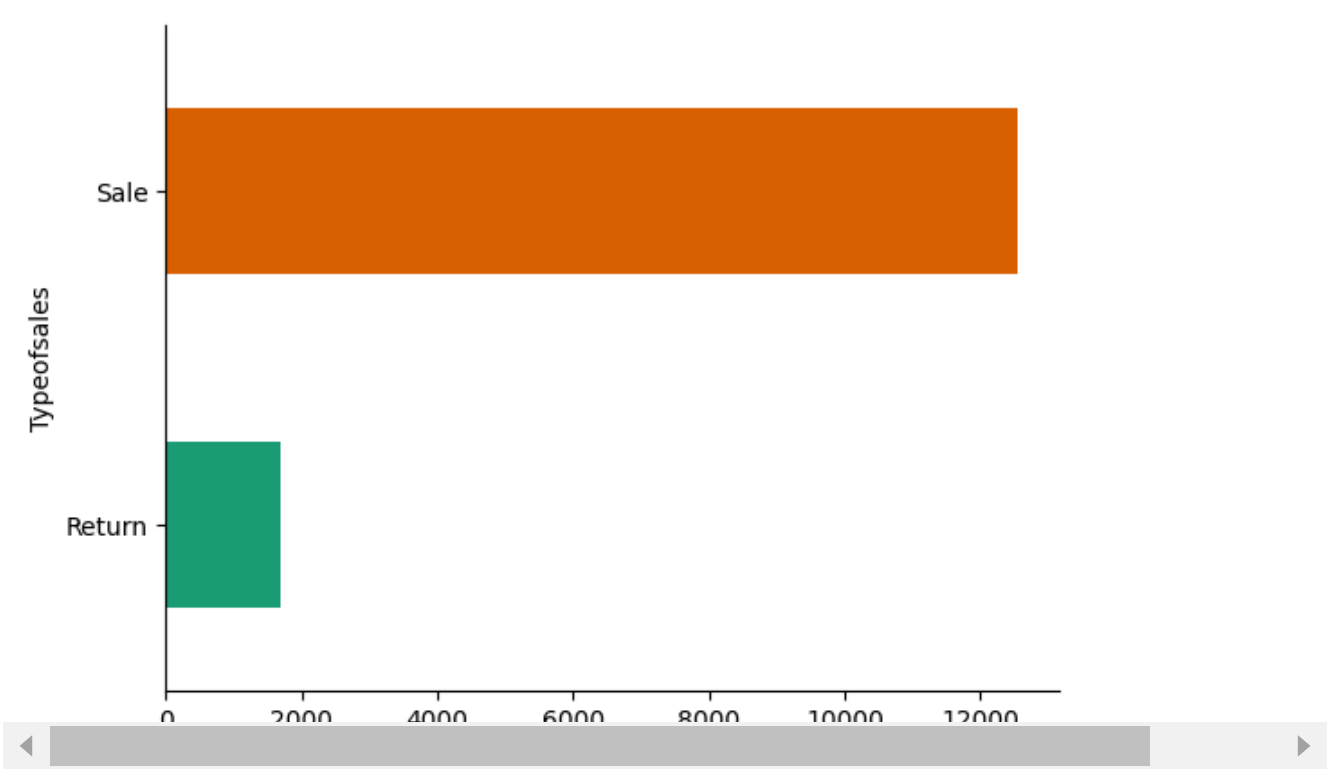
```
    x_label: grp['Dept'].value_counts()
    for x_label, grp in df.groupby('Typeofsales')
})
sns.heatmap(df_2dhist, cmap='viridis')
plt.xlabel('Typeofsales')
_ = plt.ylabel('Dept')
```




```
# Formulation

from matplotlib import pyplot as plt
import seaborn as sns
df.groupby('Formulation').size().plot(kind='barh', color=sns.palettes.mpl_palette('Dark2'))
plt.gca().spines[['top', 'right',]].set_visible(False)
```



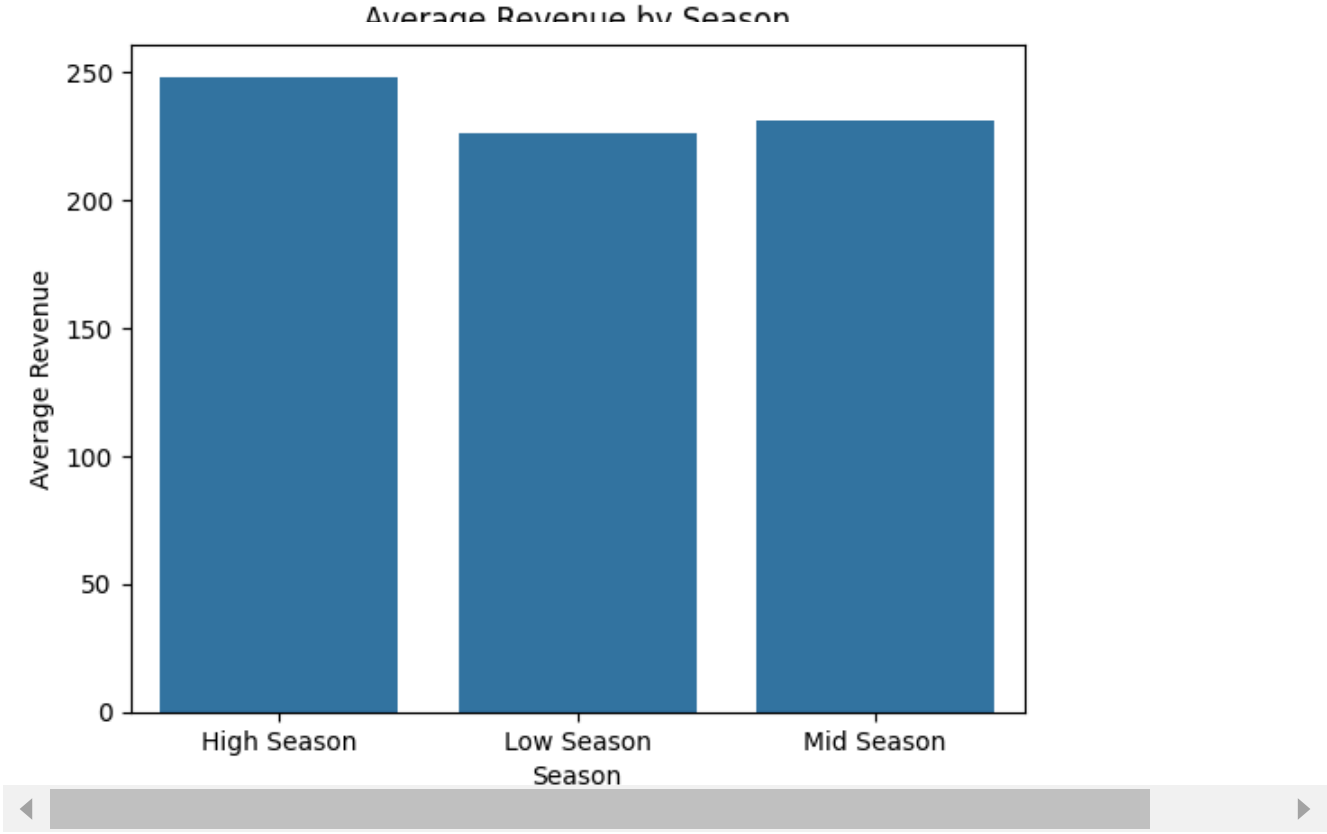

```
# Typeofsales

from matplotlib import pyplot as plt
import seaborn as sns
df.groupby('Typeofsales').size().plot(kind='barh', color=sns.palettes.mpl_palette('Dark2'))
plt.gca().spines[['top', 'right',]].set_visible(False)
```



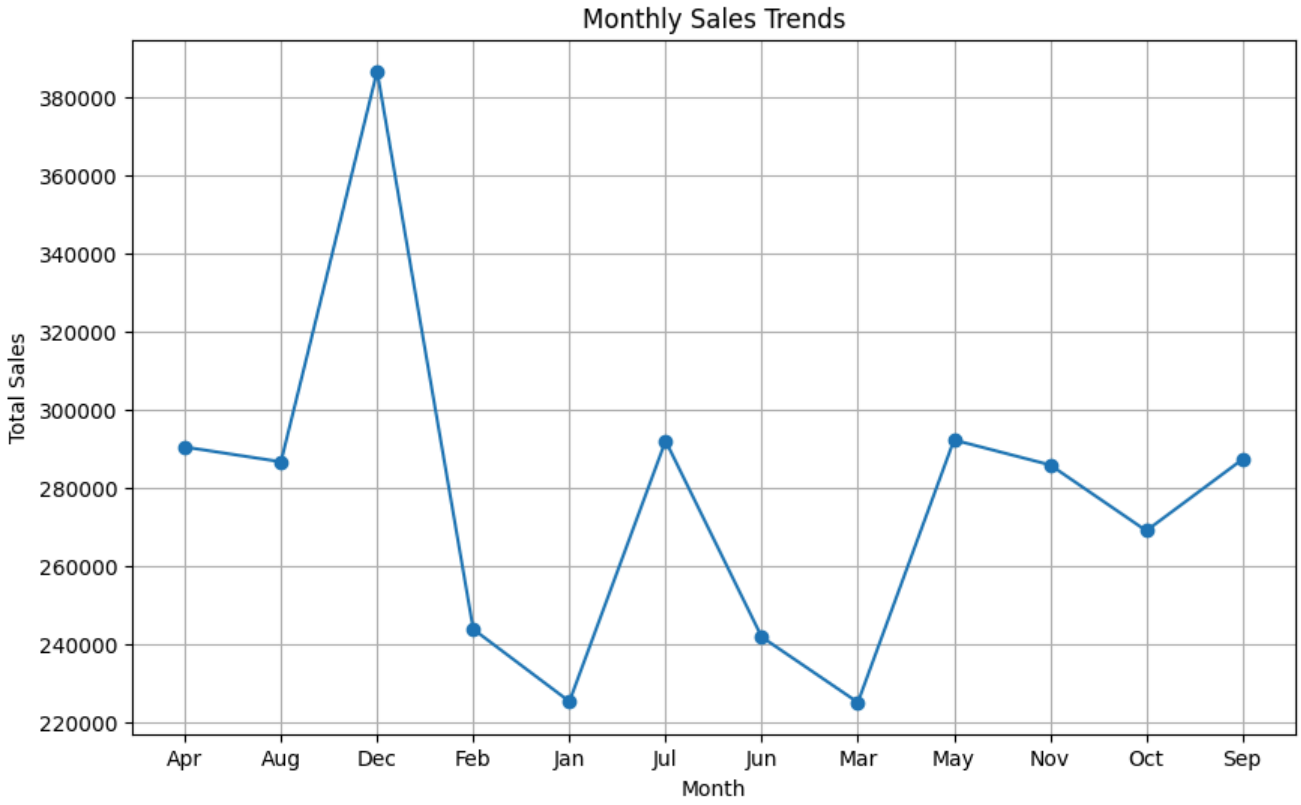

```
# Create a new column to represent seasons
df['Season'] = np.select(
    [df['Dateofbill'].isin(['Apr', 'Aug', 'Dec']), df['Dateofbill'].isin(['Jan', 'Jun', 'Sep', 'Nov']), df[
    ['High Season', 'Low Season', 'Mid Season'], default='Unknown Season')

# Group data by season and calculate average revenue
```

```
average_revenue_by_season = df.groupby('Season')['Final_Sales'].mean()

# Print the average revenue for each season
print("Average Revenue by Season:")
print(average_revenue_by_season)

# Create a bar plot to visualize average revenue by season
sns.barplot(x=average_revenue_by_season.index, y=average_revenue_by_season.values)
plt.title('Average Revenue by Season')
plt.xlabel('Season')
plt.ylabel('Average Revenue')
plt.show()
```

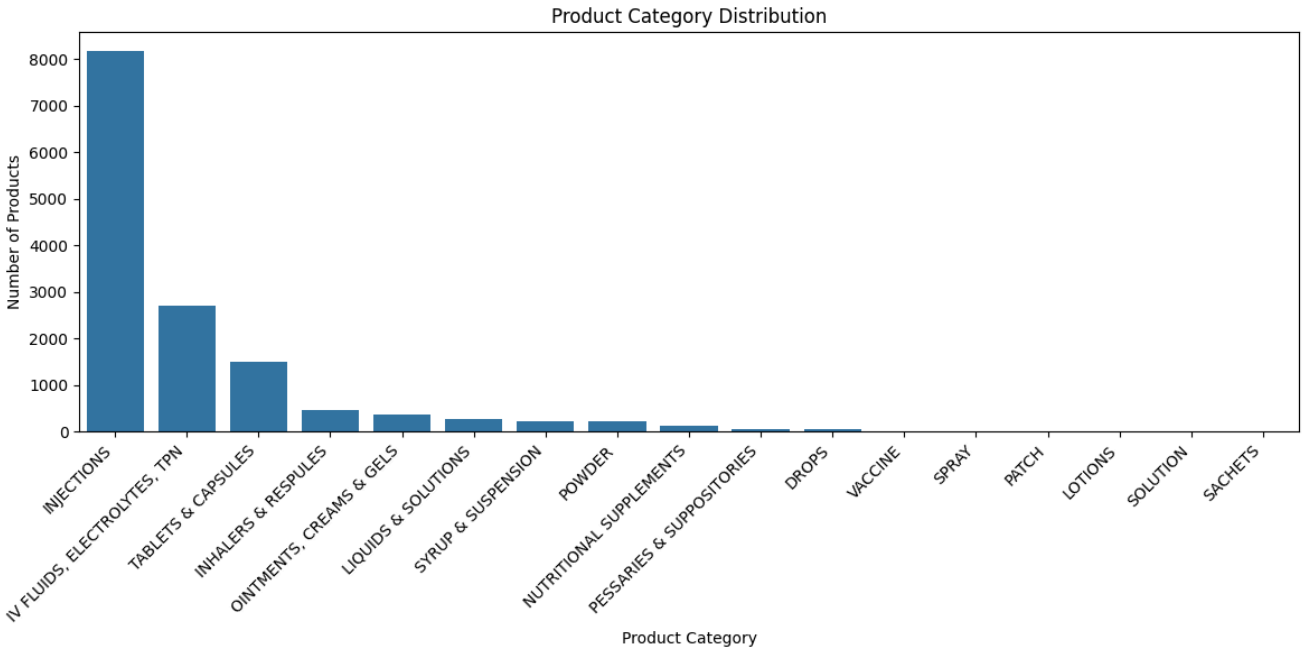
```
Average Revenue by Season:
Season
High Season    248.009715
Low Season     226.299938
Mid Season     230.772746
Name: Final_Sales, dtype: float64
```



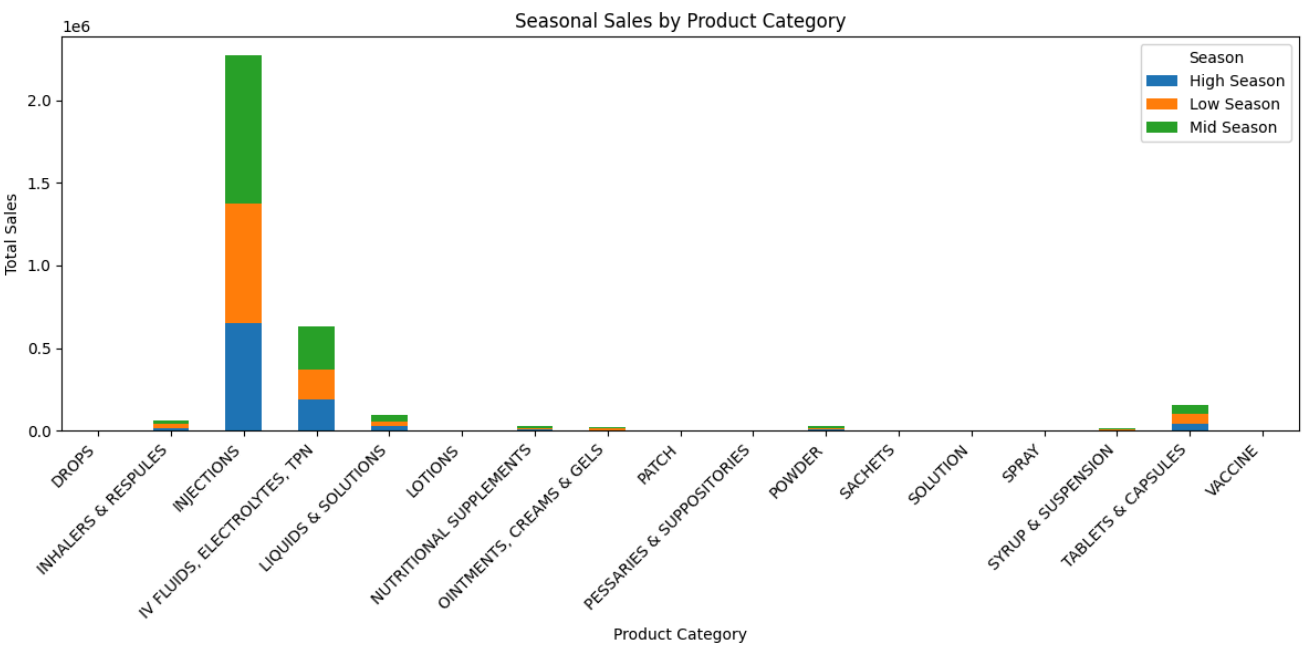

```
# Group data by month and calculate total sales
monthly_sales = df.groupby('Dateofbill')['Final_Sales'].sum()

# Create a line plot to visualize monthly sales trends
plt.figure(figsize=(10, 6))
plt.plot(monthly_sales.index, monthly_sales.values, marker='o', linestyle='-')
plt.title('Monthly Sales Trends')
plt.xlabel('Month')
plt.ylabel('Total Sales')
plt.grid(True)
plt.show()
```




```
# Group by product category (e.g., 'SubCat' or 'SubCat1') and count occurrences
product_category_distribution = df['SubCat'].value_counts()

# Visualize the distribution using a bar plot
plt.figure(figsize=(12, 6))
sns.barplot(x=product_category_distribution.index, y=product_category_distribution.values)
plt.title('Product Category Distribution')
plt.xlabel('Product Category')
plt.ylabel('Number of Products')
plt.xticks(rotation=45, ha='right') # Rotate x-axis labels for better readability
plt.tight_layout()
plt.show()
```

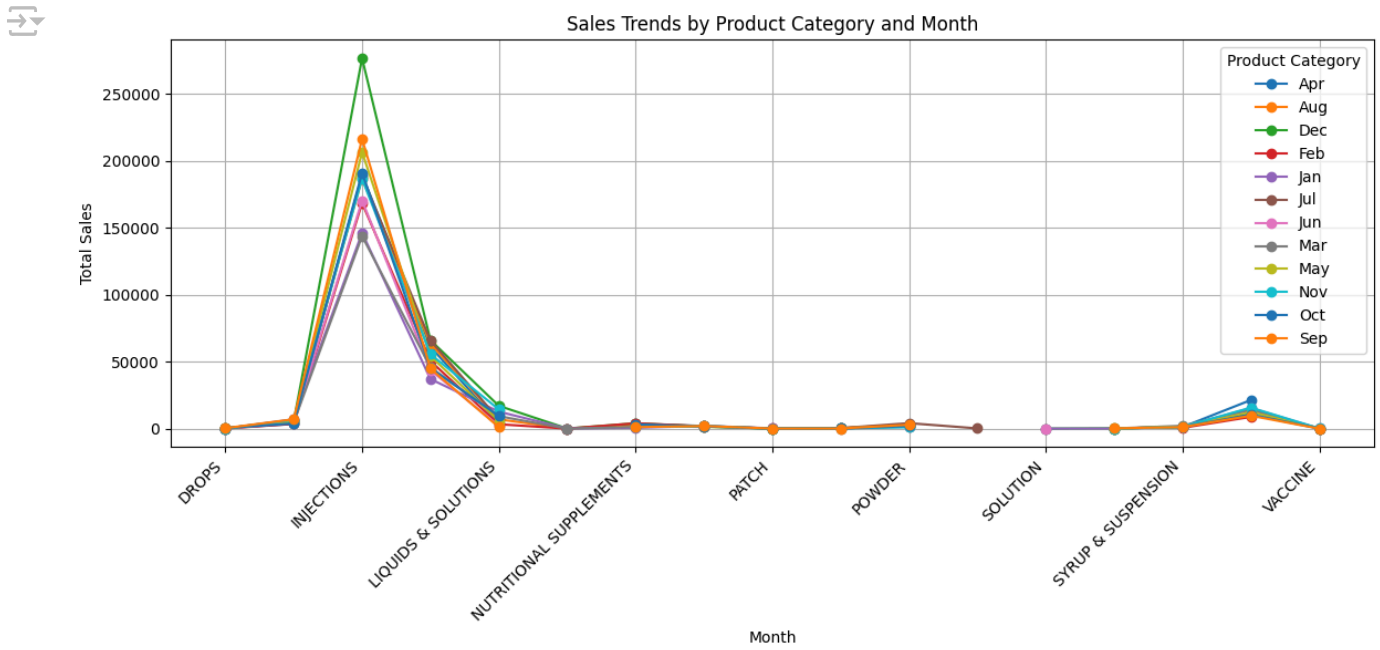



```
# Analyze sales by product category and season
seasonal_sales_by_category = df.groupby(['SubCat', 'Season'])['Final_Sales'].sum().unstack()
seasonal_sales_by_category.plot(kind='bar', stacked=True, figsize=(12, 6))
plt.title('Seasonal Sales by Product Category')
plt.xlabel('Product Category')
plt.ylabel('Total Sales')
plt.xticks(rotation=45, ha='right')
plt.legend(title='Season')
plt.tight_layout()
plt.show()
```




```
# Group by product category and month to analyze sales trends
sales_by_category_and_month = df.groupby(['SubCat', 'Dateofbill'])['Final_Sales'].sum().unstack()

# Visualize the trends using a line plot
sales_by_category_and_month.plot(kind='line', figsize=(12, 6), marker='o')
plt.title('Sales Trends by Product Category and Month')
```

```
plt.title('Sales Trends by Product Category and Month')
plt.xlabel('Month')
plt.ylabel('Total Sales')
plt.xticks(rotation=45, ha='right')
plt.legend(title='Product Category')
plt.grid(True)
plt.tight_layout()
plt.show()
```




```
# Filter data for Department1 and count return occurrences by Specialisation
return_counts_by_specialisation = df[(df['Typeofsales'] == 'Return') & (df['Dept'] == 'Department1')].group
print("Return Occurrences by Specialisation in Department1:")
print(return_counts_by_specialisation)
```