Data Cleaning and EDA using Python:

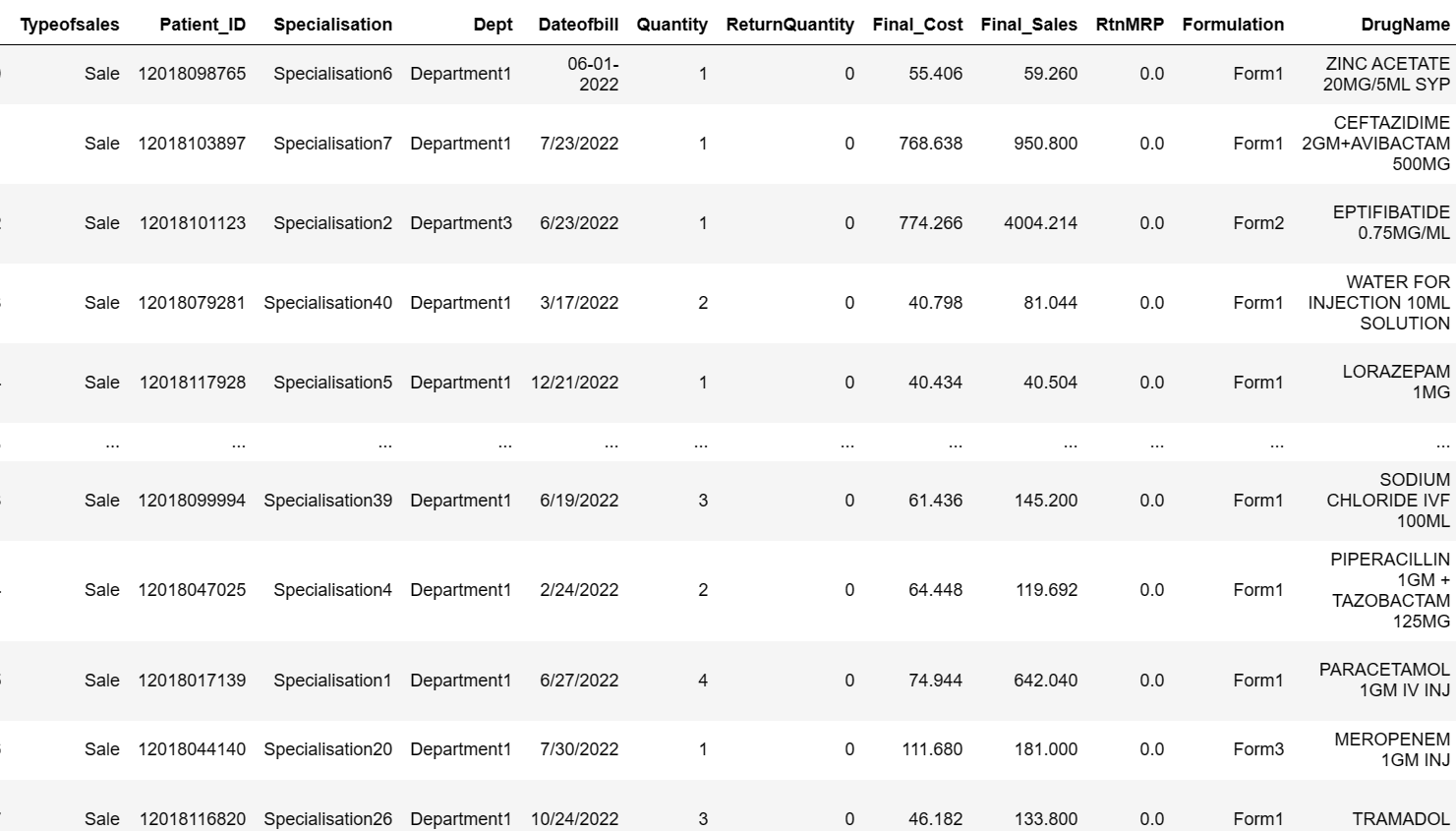
**Data Cleaning:**

Importing Medical inventory dataset and viewing the data

import pandas as pd

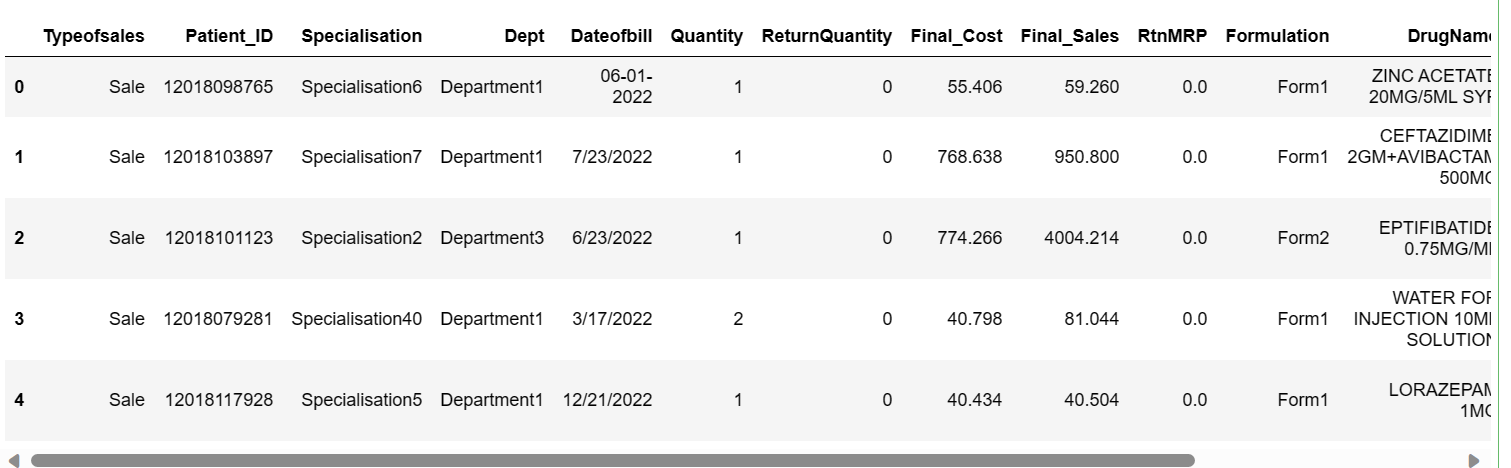
df = pd.read\_csv("Medical Inventory Optimaization Dataset.csv")

df



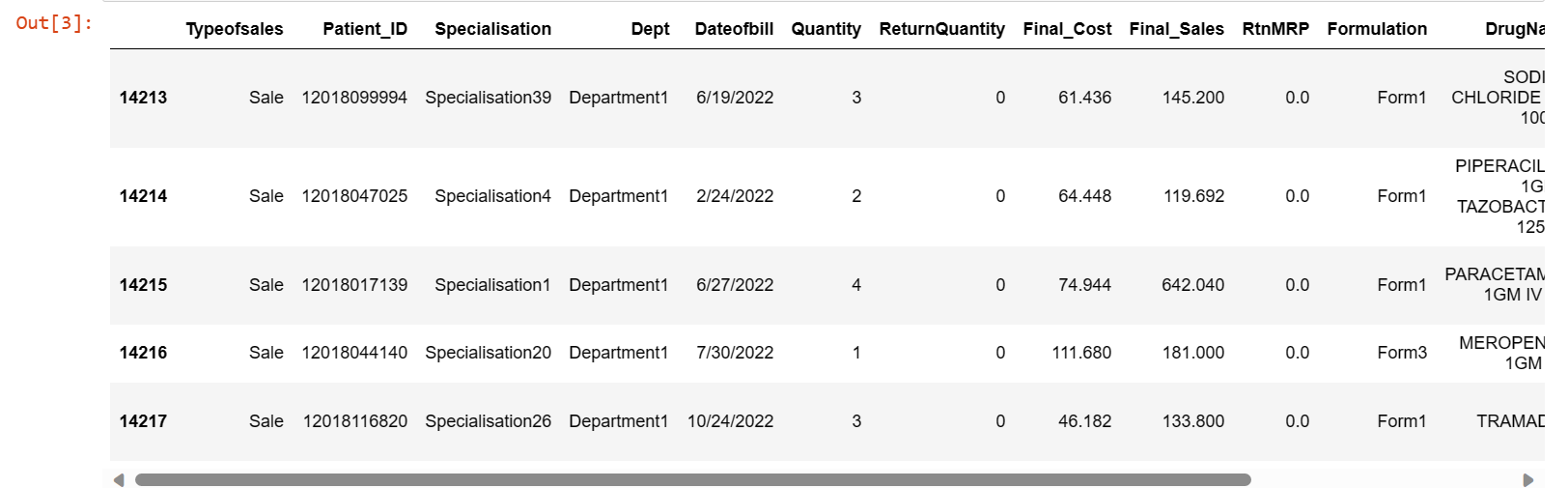
Displaying only 5 rows from top

df.head()



Displaying last 5 rows of dataset

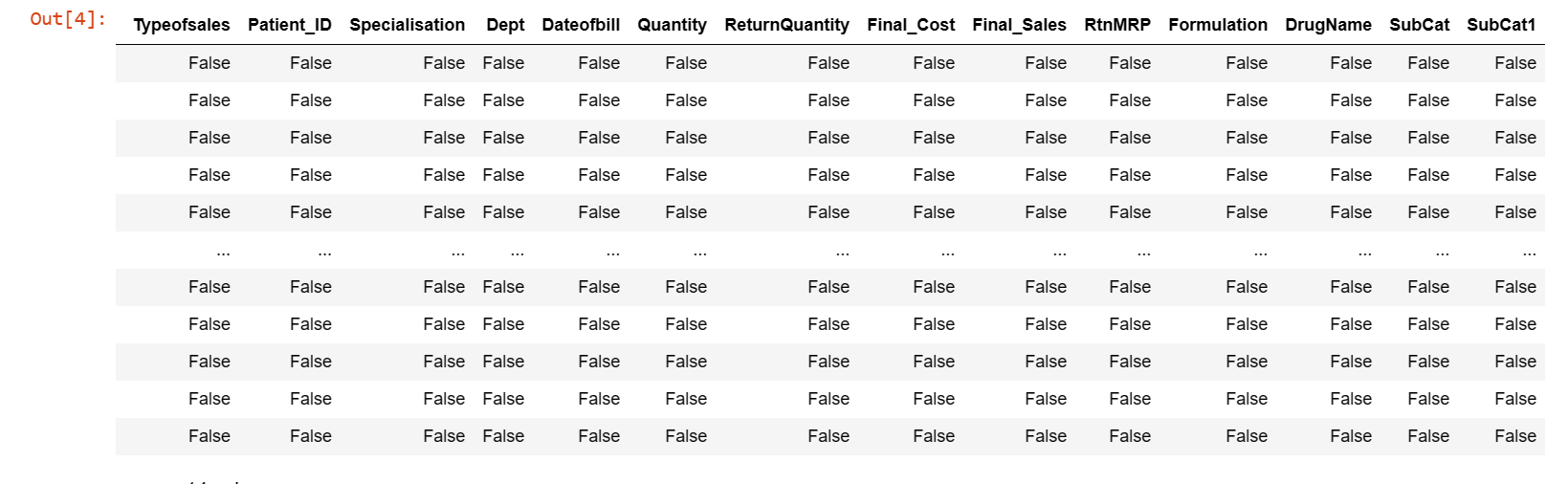
df.tail()



Checking for the null values

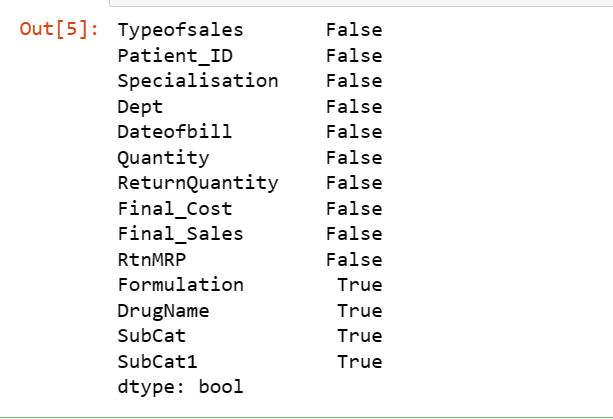
# checking null values

df.isnull()



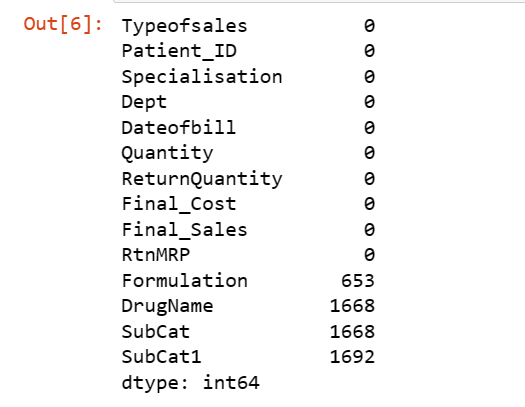
#returns boolean value if null values is present in data

df.isna().any()



#identifying total null values

df.isna().sum()



# number of columns contining null values

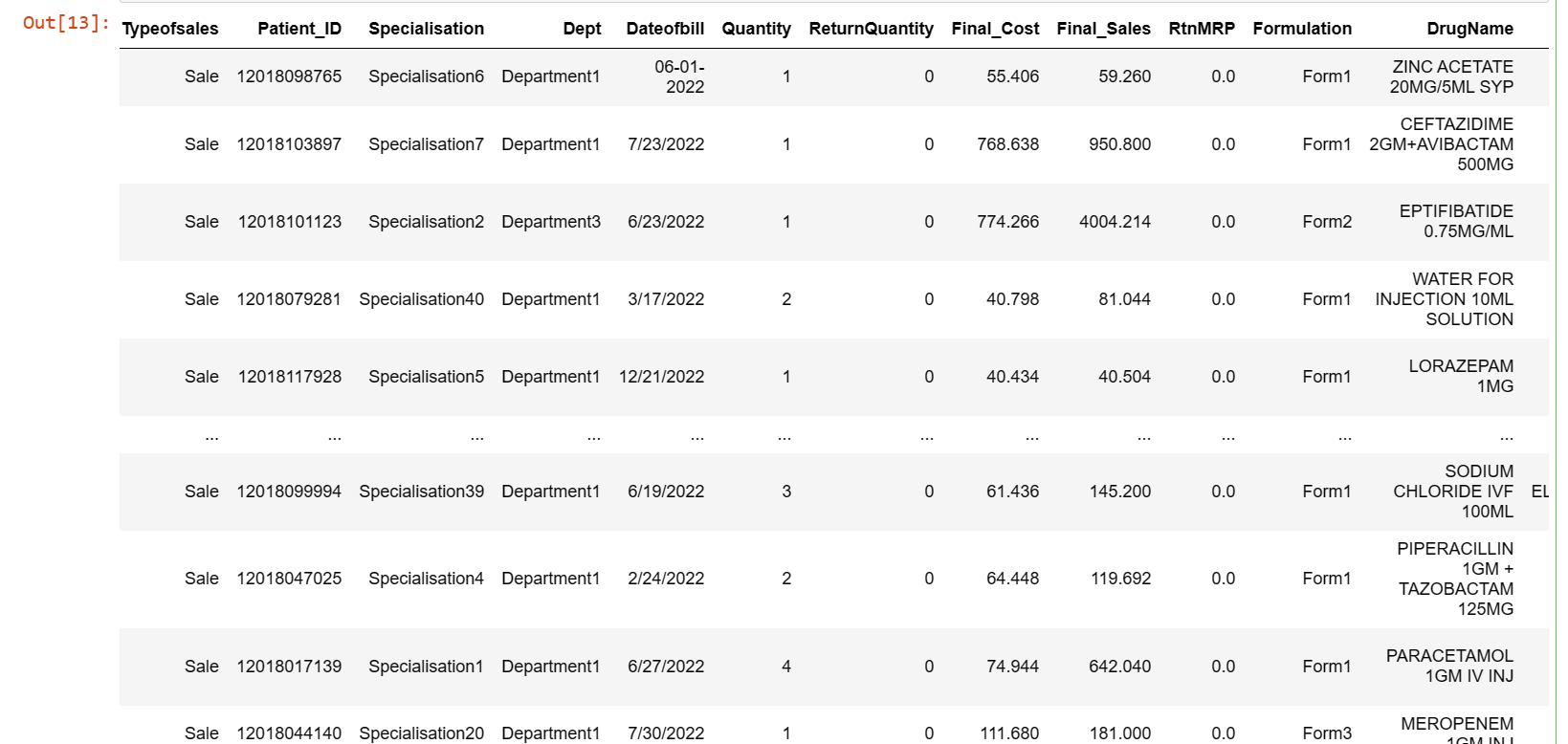
df.isna().any().sum()



From the above code we can see we have null values in 4 different column namely Formulation, Drugname, Subcat, Subcat1. Here Formulation contains 653 values, Drugname contains 1668, Subcat contains 1688, Subcat1 contains 1692 null values.

Replacing null values with NA. So the sum of columns will remain same

df.fillna('NA')



**Formating Date values:**

From the observation in date column, the format is in consistant. 08-06-2022 and 7/15/2022 are the two formats we can see in the date column. So we are changing the format uniformly 7/15/2022 and convert the format from m/d/y to d/m/y

import pandas as pd

from dateutil import parser

# Function to convert different date formats to dd/mm/yyyy

def convert\_to\_dd\_mm\_yyyy(date\_str):

try:

# Try parsing as mm/dd/yy format

date\_parsed = parser.parse(date\_str, dayfirst=False, yearfirst=False)

except ValueError:

# If parsing fails, try parsing as dd-mm-yyyy format

date\_parsed = parser.parse(date\_str, dayfirst=True, yearfirst=False)

# Convert the date to the desired format 'dd/mm/yyyy'

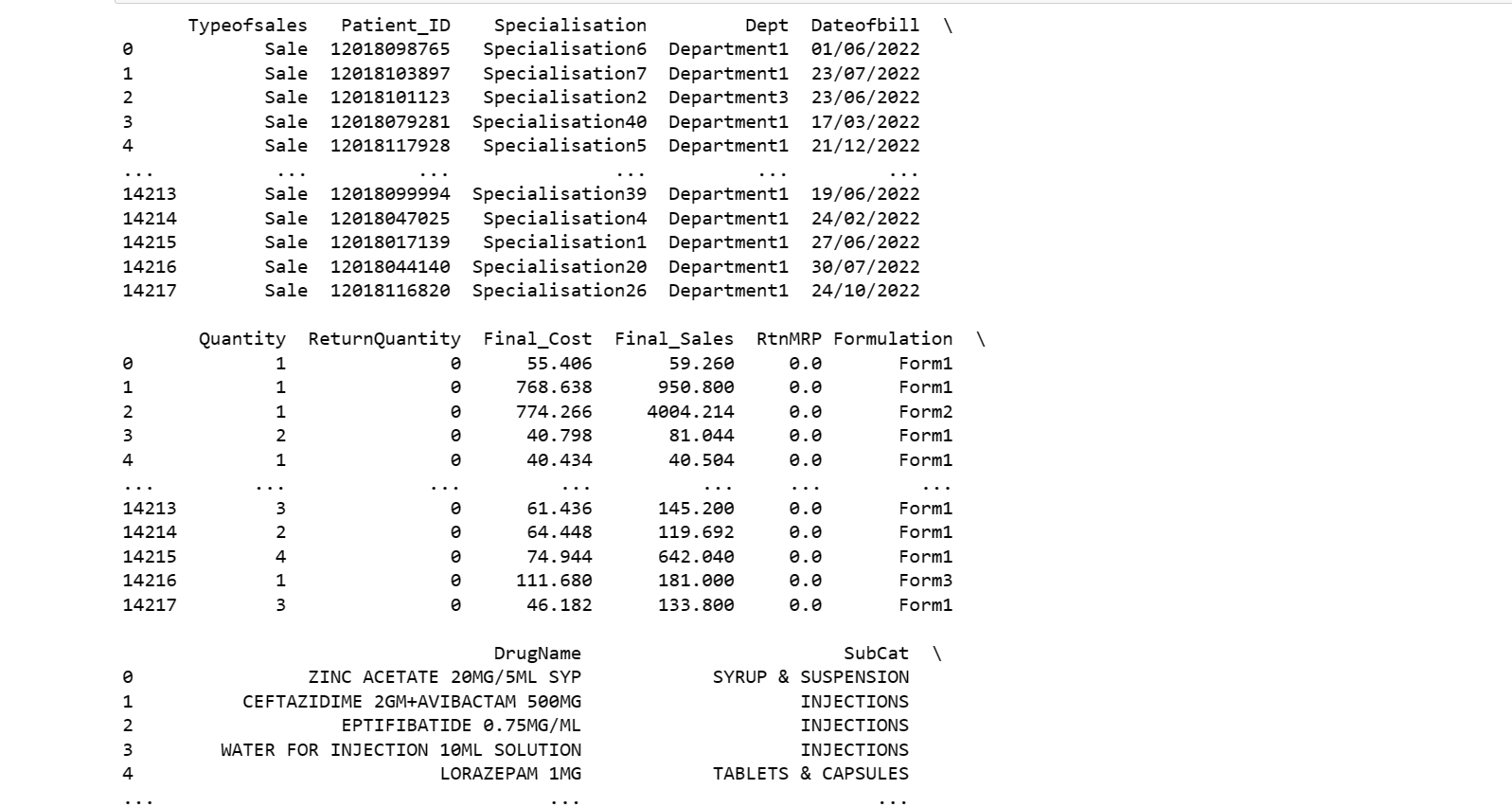
return date\_parsed.strftime('%d/%m/%Y')

# Apply the conversion function to the 'dateofbill' column

df['Dateofbill'] = df['Dateofbill'].apply(convert\_to\_dd\_mm\_yyyy)

# Display the updated DataFrame

print(df)



**Feature engineering:**

Create column with month name from date and add that column and remaning cleaned data from MioView to other view name MioView1

# Feature Engineering

df['month'] = pd.to\_datetime(df['Dateofbill'], format='%d/%m/%Y').dt.strftime('%b')

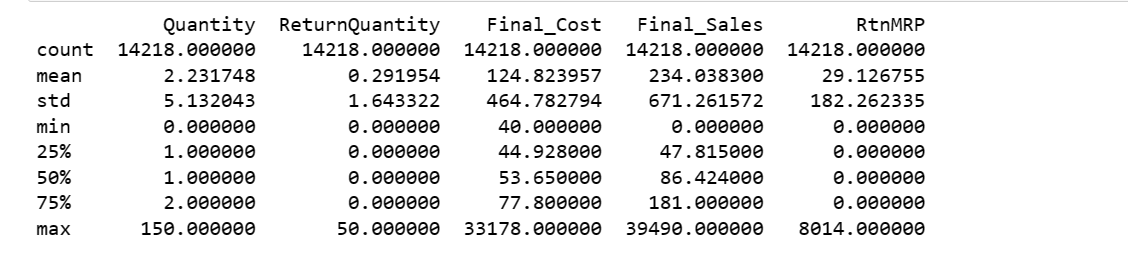
df

**EDA Analysis:**

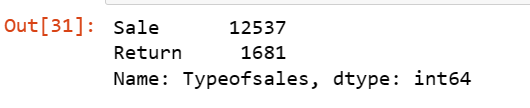
# Summary statistics for all numerical columns

numerical\_summary = df[['Quantity', 'ReturnQuantity', 'Final\_Cost', 'Final\_Sales', 'RtnMRP']].describe()

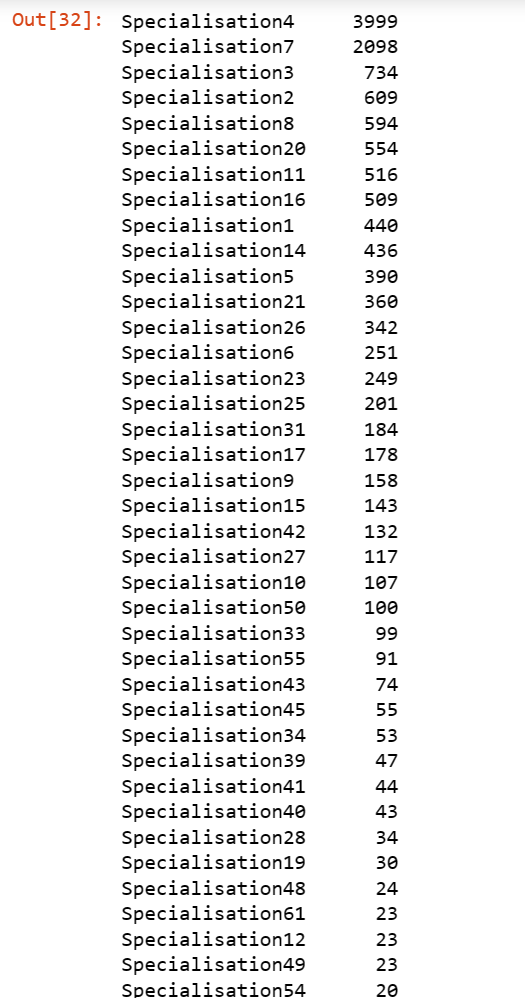
print(numerical\_summary)



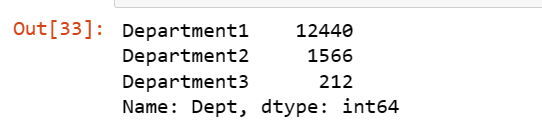
df['Typeofsales'].value\_counts()



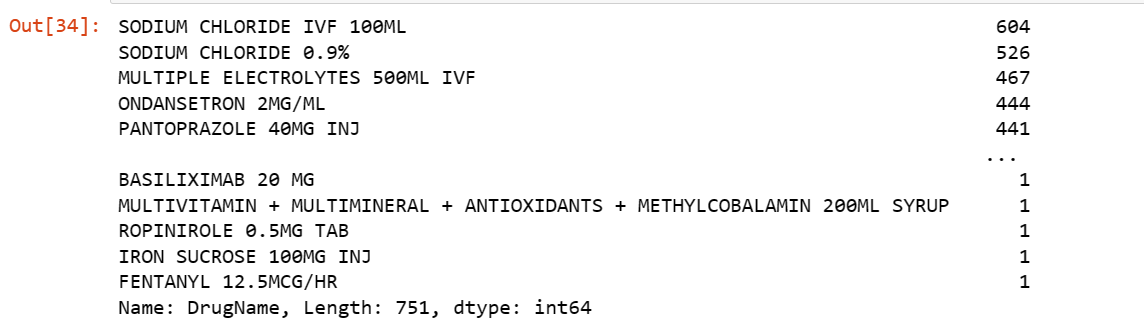
df['Specialisation'].value\_counts()



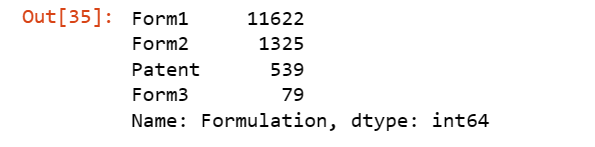
df['Dept'].value\_counts()



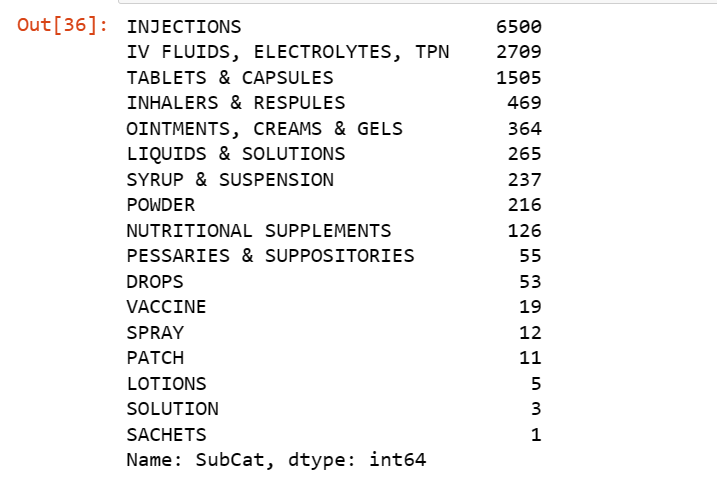
df['DrugName'].value\_counts()



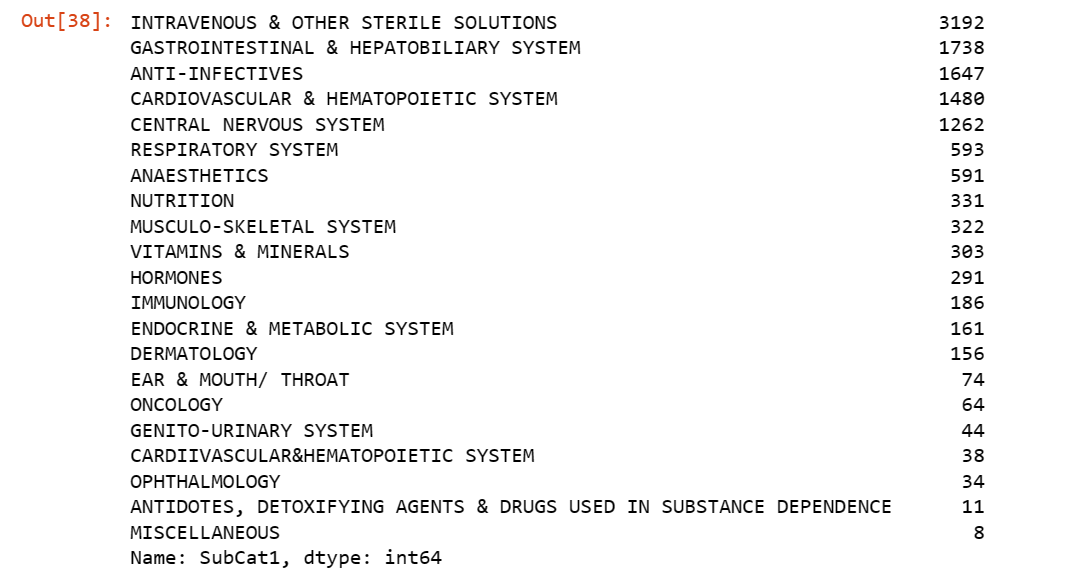
df['Formulation'].value\_counts()



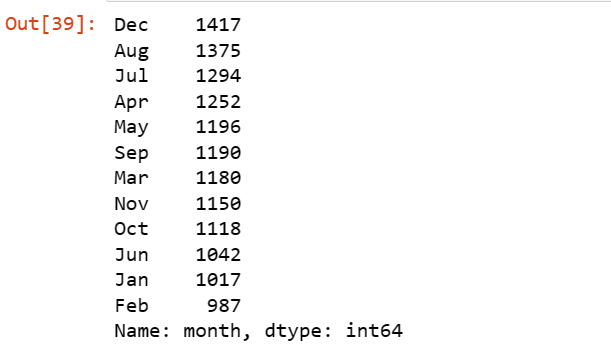
df['SubCat'].value\_counts()



df['SubCat1'].value\_counts()



df['month'].value\_counts()



First moment bussiness decision: Mean,Median,Mode

# Calculate mean, median, and mode for numerical columns

numerical\_columns = ['Quantity', 'ReturnQuantity', 'Final\_Cost', 'Final\_Sales', 'RtnMRP']

# Mean

mean\_values = df[numerical\_columns].mean()

print("Mean Values:")

print(mean\_values)

# Median

median\_values = df[numerical\_columns].median()

print("\nMedian Values:")

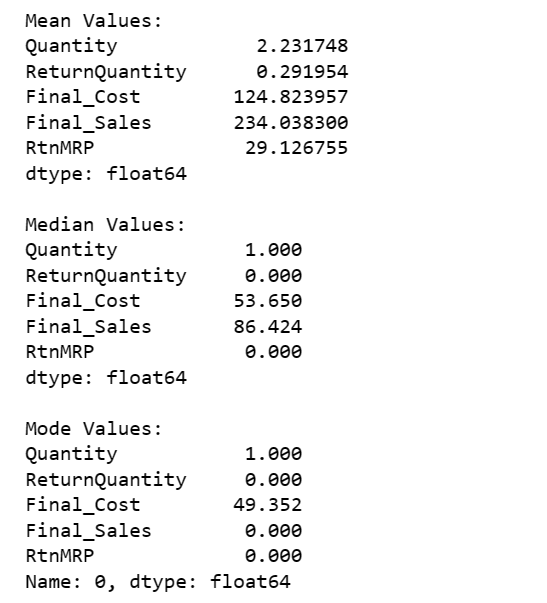
print(median\_values)

# Mode

mode\_values = df[numerical\_columns].mode().iloc[0]

print("\nMode Values:")

print(mode\_values)



Second Moment Business Decision: Variance, Standard Deviation, Range

import pandas as pd

# Compute variance for numerical columns

variance\_df = df[['Quantity', 'ReturnQuantity', 'Final\_Cost', 'Final\_Sales', 'RtnMRP']].var()

# Compute standard deviation for numerical columns

std\_dev\_df = df[['Quantity', 'ReturnQuantity', 'Final\_Cost', 'Final\_Sales', 'RtnMRP']].std()

# Compute range for numerical columns

range\_df = df[['Quantity', 'ReturnQuantity', 'Final\_Cost', 'Final\_Sales', 'RtnMRP']].max() - df[['Quantity', 'ReturnQuantity', 'Final\_Cost', 'Final\_Sales', 'RtnMRP']].min()

# Display the results

print("Variance:")

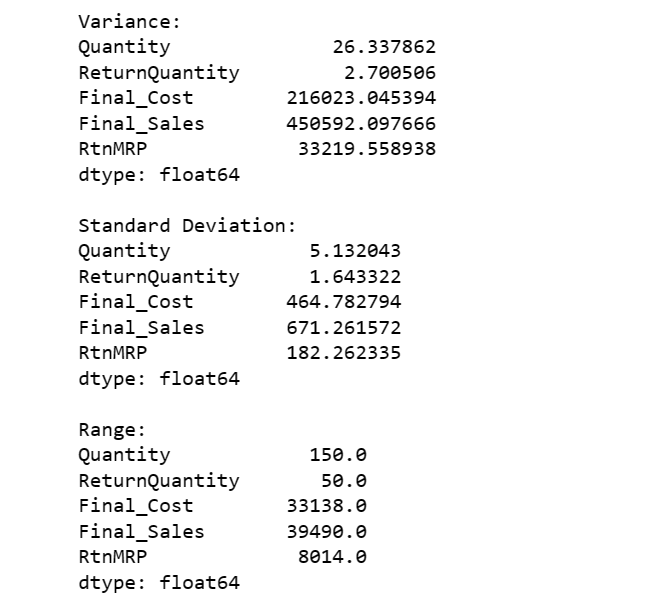
print(variance\_df)

print("\nStandard Deviation:")

print(std\_dev\_df)

print("\nRange:")

print(range\_df)



Third Moment Business Decision: skewness, Kurtosis

from scipy.stats import skew, kurtosis

numerical\_columns = ['Quantity', 'ReturnQuantity', 'Final\_Cost', 'Final\_Sales', 'RtnMRP']

for column in numerical\_columns:

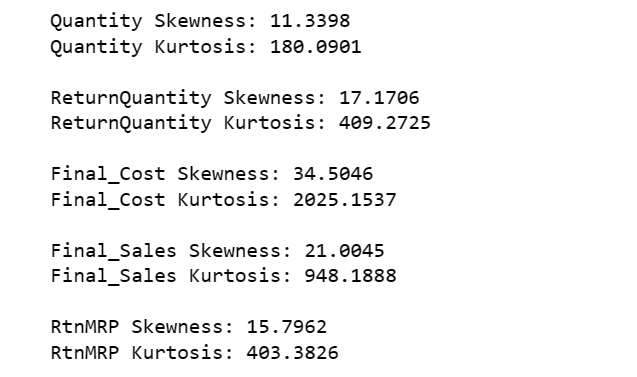
col\_data = df[column].dropna() # Remove NaN values

skew\_val = skew(col\_data)

kurt\_val = kurtosis(col\_data)

print(f"\n{column} Skewness: {skew\_val:.4f}")

print(f"{column} Kurtosis: {kurt\_val:.4f}")



Analysis:

import pandas as pd

import seaborn as sns

import matplotlib.pyplot as plt

# Q1) Total Distinct Patients

total\_distinct\_patients = df['Patient\_ID'].nunique()

print(f"Q1) Total Distinct Patients: {total\_distinct\_patients}")

# Q2) Patient\_ID count where type of sale is return

return\_patients\_count = df[df['Typeofsales'] == 'Return']['Patient\_ID'].nunique()

print(f"Q2) Patient\_ID count where type of sale is return: {return\_patients\_count}")

# Q3) Patient\_ID count where type of sale is sales

sales\_patients\_count = df[df['Typeofsales'] == 'Sale']['Patient\_ID'].nunique()

print(f"Q3) Patient\_ID count where type of sale is sales: {sales\_patients\_count}")

# Q4) Overall Bounce Rate

overall\_bounce\_rate = (return\_patients\_count / total\_distinct\_patients) \* 100

print(f"Q4) Overall Bounce Rate: {overall\_bounce\_rate:.2f}%")

# Q5) Bounce rate by Specialization

bounce\_rate\_by\_specialization = df[df['Typeofsales'] == 'Return'].groupby('Specialisation')['Patient\_ID'].nunique() / df.groupby('Specialisation')['Patient\_ID'].nunique() \* 100

print(f"Q5) Bounce Rate by Specialization:")

print(bounce\_rate\_by\_specialization)

# Q6) Total cost of purchase that returns from SubCat

total\_cost\_return\_subcat = df[df['Typeofsales'] == 'Return']['Final\_Cost'].sum()

print(f"Q6) Total cost of purchase that returns from SubCat: {total\_cost\_return\_subcat:.2f}")

# Q7) Count of drugs returned without sales

drugs\_returned\_without\_sales\_count = df[(df['Typeofsales'] == 'Return') & (df['Final\_Sales'] == 0)]['DrugName'].nunique()

print(f"Q7) Count of drugs returned without sales: {drugs\_returned\_without\_sales\_count}")

# Q8) Return items based on month

return\_items\_by\_month = df[df['Typeofsales'] == 'Return'].groupby('month')['Patient\_ID'].count()

print(f"Q8) Return items based on month:")

print(return\_items\_by\_month)

# Q9) Total sales when sales is return or sale

total\_sales\_return\_sale = df[df['Typeofsales'].isin(['Return', 'Sale'])]['Final\_Sales'].sum()

print(f"Q9) Total sales when sales is return or sale: {total\_sales\_return\_sale:.2f}")

# Q10) Drugs which are mostly returned

mostly\_returned\_drugs = df[df['Typeofsales'] == 'Return']['DrugName'].value\_counts().idxmax()

print(f"Q10) Drugs which are mostly returned: {mostly\_returned\_drugs}")

# Q11) Total sales based on month

total\_sales\_by\_month = df.groupby('month')['Final\_Sales'].sum()

print(f"Q11) Total sales based on month:")

print(total\_sales\_by\_month)

# Q12) Average quantity of drug purchases

average\_quantity\_purchases = df.groupby('DrugName')['Quantity'].mean()

print(f"Q12) Average quantity of drug purchases:")

print(average\_quantity\_purchases)

# Q13) Relation between quantity and total sales

plt.figure(figsize=(10, 6))

sns.scatterplot(x='Quantity', y='Final\_Sales', data=df)

plt.title('Q13) Relation between quantity and total sales')

plt.show()

# Q14) Average Sales based on Specialisation

average\_sales\_by\_specialization = df.groupby('Specialisation')['Final\_Sales'].mean()

print(f"Q14) Average Sales based on Specialisation:")

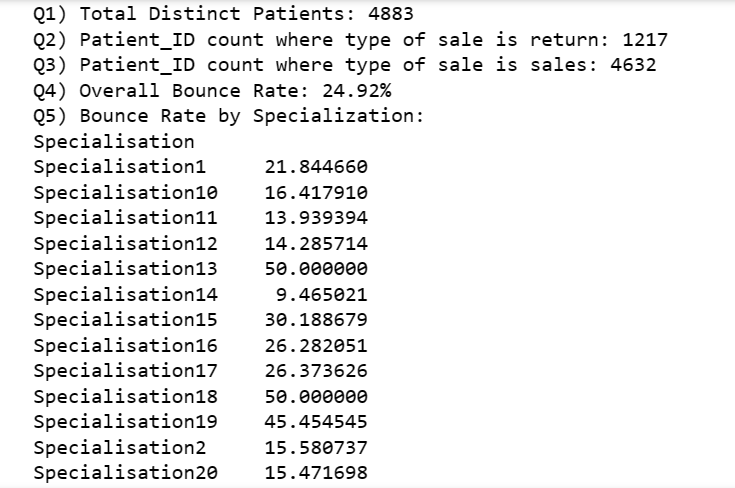
print(average\_sales\_by\_specialization)

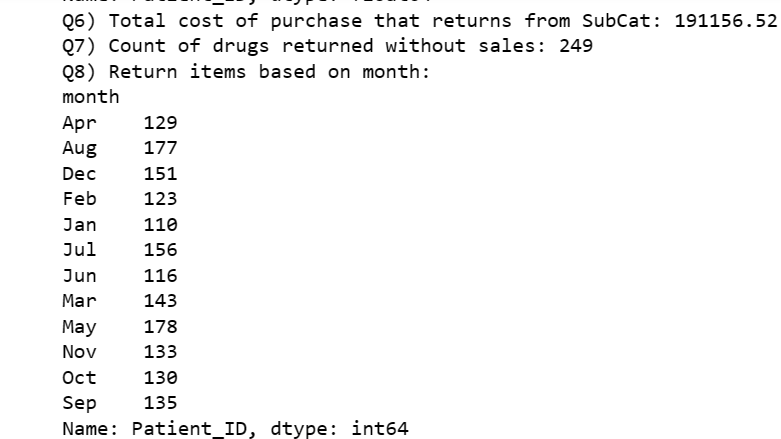
# Q15) Frequency of return quantity

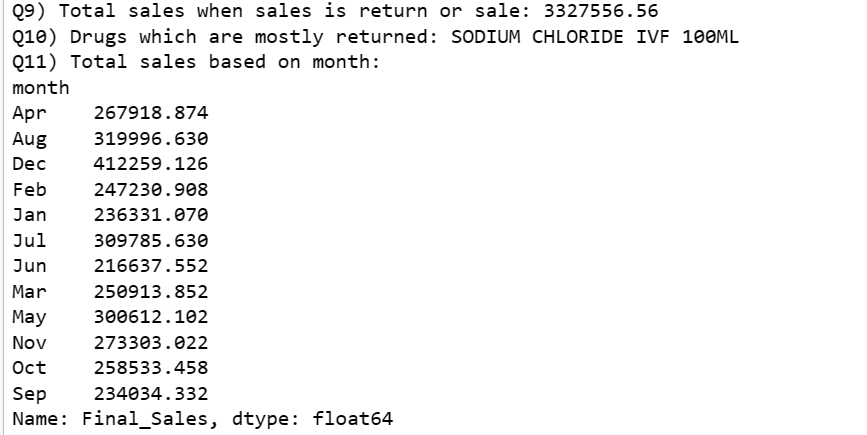
return\_quantity\_frequency = df[df['Typeofsales'] == 'return']['ReturnQuantity'].value\_counts()

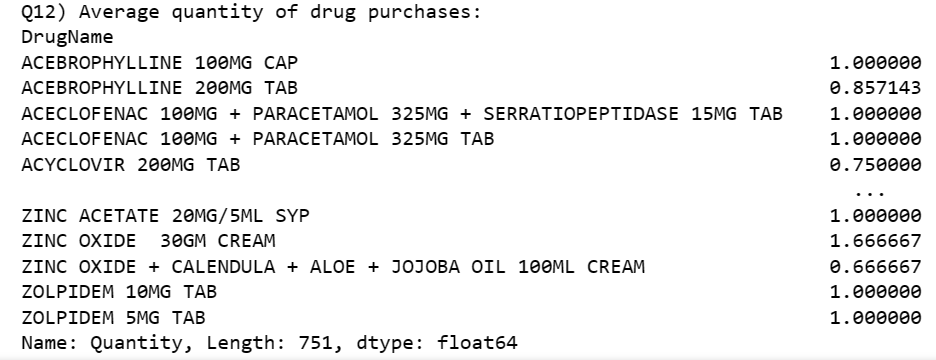
print(f"Q15) Frequency of return quantity:")

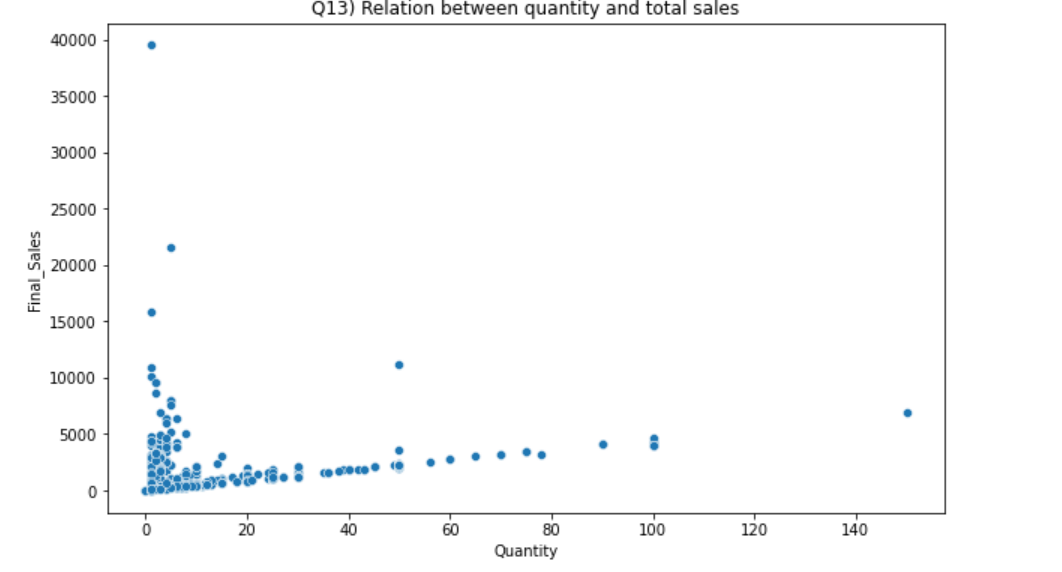
print(return\_quantity\_frequency)

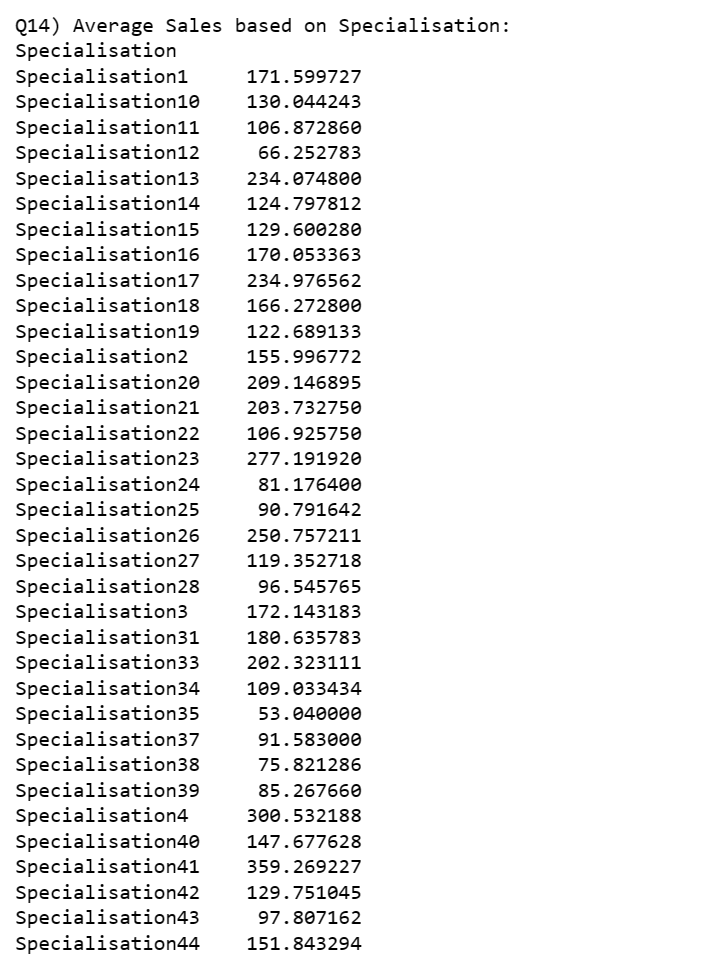


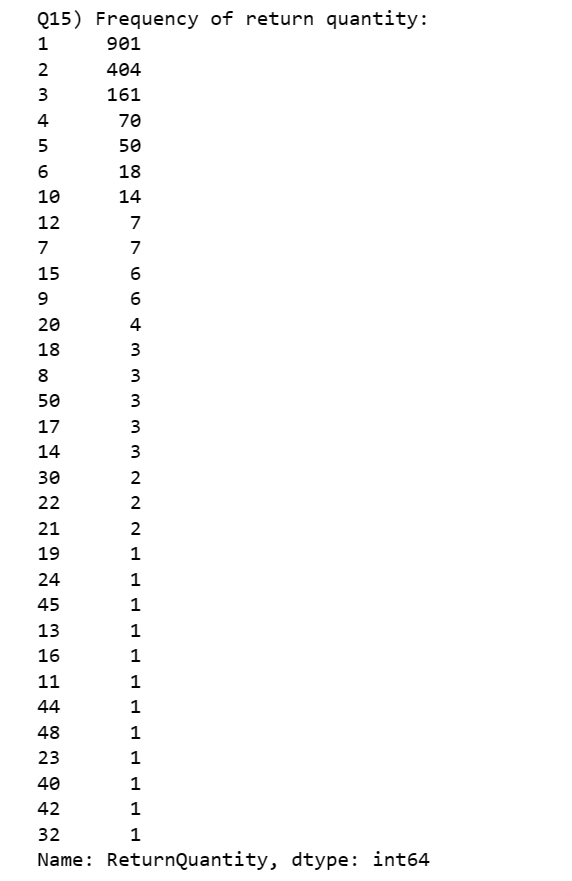












**Univariant Analysis:**

How does the 'Final\_Sales' vary over different 'month' values?

import pandas as pd

import matplotlib.pyplot as plt

import seaborn as sns

# Univariate chart for month and sales using a line plot

plt.figure(figsize=(12, 6))

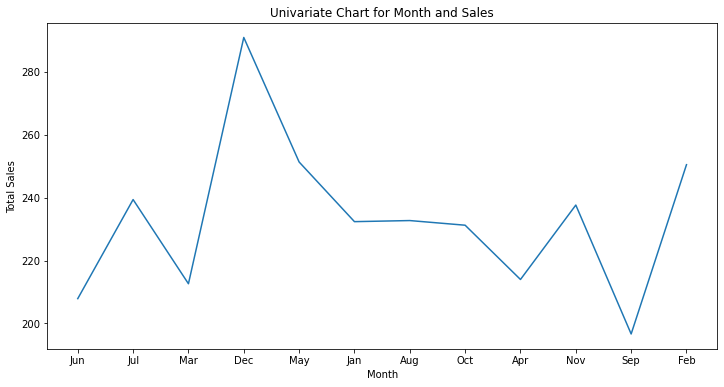
sns.lineplot(x='month', y='Final\_Sales', data=df, ci=None)

plt.title('Univariate Chart for Month and Sales')

plt.xlabel('Month')

plt.ylabel('Total Sales')

plt.show()



Is the distribution of 'Final\_cost' skewed or symmetric?

# Is the distribution of 'Final\_cost' skewed or symmetric?

import pandas as pd

import seaborn as sns

import matplotlib.pyplot as plt

# Plot a histogram and kernel density plot for 'Final\_cost'

plt.figure(figsize=(10, 6))

sns.histplot(df['Final\_Cost'], kde=True)

plt.title('Distribution of Final\_Cost')

plt.xlabel('Final\_Cost')

plt.show()

# Calculate skewness

skewness = df['Final\_Cost'].skew()

# Assess skewness

if skewness > 0:

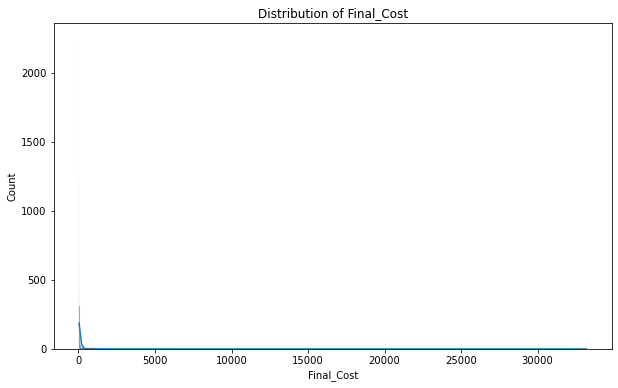
print(f"The distribution is right-skewed (positively skewed) with skewness value: {skewness:.2f}")

elif skewness < 0:

print(f"The distribution is left-skewed (negatively skewed) with skewness value: {skewness:.2f}")

else:

print("The distribution is approximately symmetric.")



Are there extreme values in the 'Quantity' column?

import pandas as pd

import matplotlib.pyplot as plt

# Check for extreme values using a boxplot

plt.figure(figsize=(8, 6))

sns.boxplot(x='Quantity', data=df)

plt.title('Boxplot of Quantity')

plt.show()

# Calculate the interquartile range (IQR)

Q1 = df['Quantity'].quantile(0.25)

Q3 = df['Quantity'].quantile(0.75)

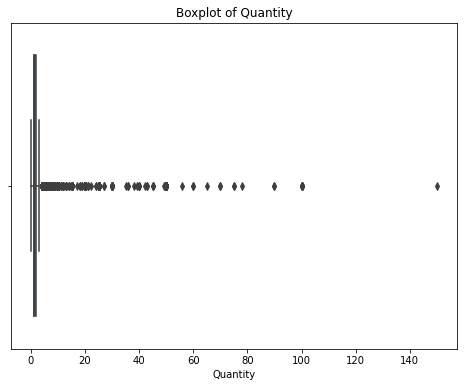
IQR = Q3 - Q1

# Identify potential outliers using the IQR method

outliers = (df['Quantity'] < (Q1 - 1.5 \* IQR)) | (df['Quantity'] > (Q3 + 1.5 \* IQR))

# Display the number of potential outliers

print(f"Number of potential outliers in Quantity: {outliers.sum()}")



What are the 25th, 50th, and 75th percentiles of 'Final\_Sales'?

import pandas as pd

import seaborn as sns

import matplotlib.pyplot as plt

# Calculate percentiles using describe method

percentiles = df['Final\_Sales'].describe(percentiles=[.25, .5, .75])[['25%', '50%', '75%']]

print("Percentiles of Final\_Sales:")

print(percentiles)

# Create a boxplot for 'Final\_Sales'

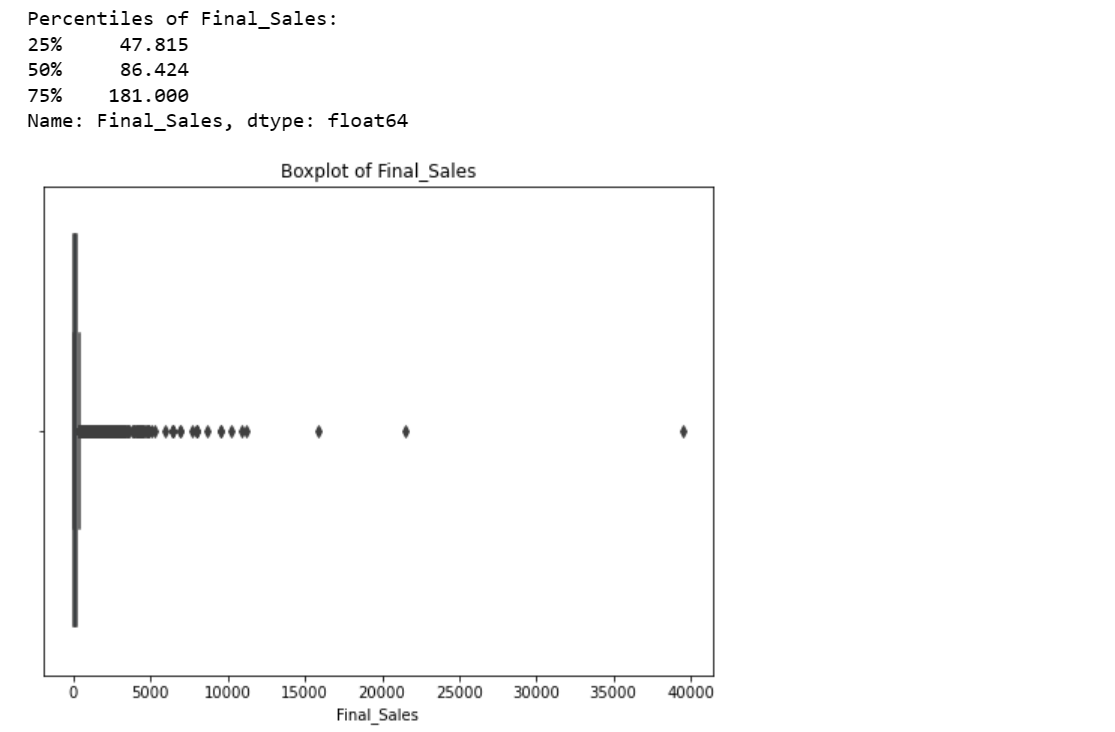
plt.figure(figsize=(8, 6))

sns.boxplot(x='Final\_Sales', data=df)

plt.title('Boxplot of Final\_Sales')

plt.xlabel('Final\_Sales')

plt.show()



**Bivariant Analysis:**

Return quantity by drug formulation

import pandas as pd

import matplotlib.pyplot as plt

import seaborn as sns

# Analysis of Return Quantity by Drug Formulation

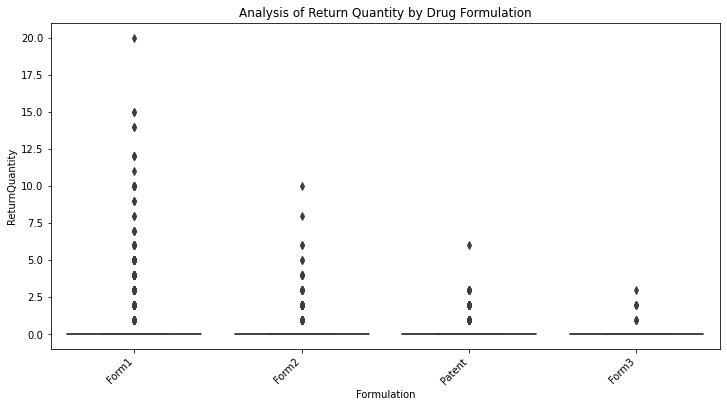
plt.figure(figsize=(12, 6))

sns.boxplot(x='Formulation', y='ReturnQuantity', data=df)

plt.title('Analysis of Return Quantity by Drug Formulation')

plt.xticks(rotation=45, ha='right')

plt.show()



How Month Effect Final sales

# Time Series Plot for Final Sales

df['Dateofbill'] = pd.to\_datetime(df['Dateofbill']) # Convert Dateofbill to datetime if it's not already

df\_time\_series = df.set\_index('Dateofbill')

plt.figure(figsize=(12, 6))

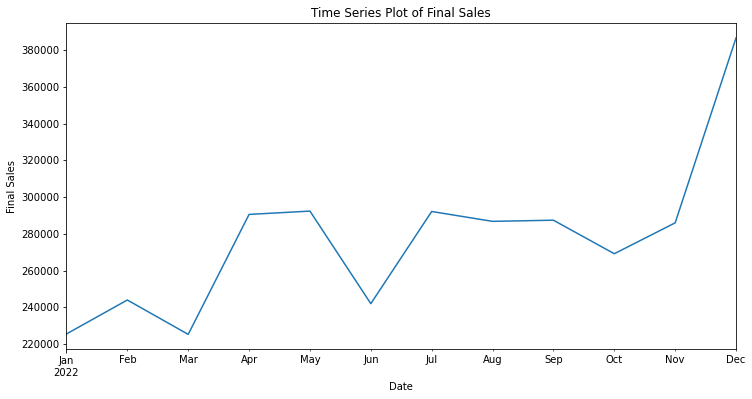
df\_time\_series['Final\_Sales'].resample('M').sum().plot()

plt.title('Time Series Plot of Final Sales')

plt.xlabel('Date')

plt.ylabel('Final Sales')

plt.show()



Here we can see we have more sales on December

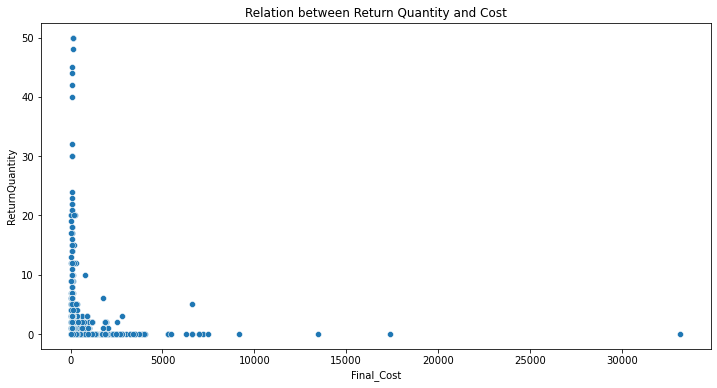
Explore how Return Quantity relates to the cost of the products

plt.figure(figsize=(12, 6))

sns.scatterplot(x='Final\_Cost', y='ReturnQuantity', data=df)

plt.title('Relation between Return Quantity and Cost')

plt.show()

Differences in Return Quantity and Final Sales across different Subcategories

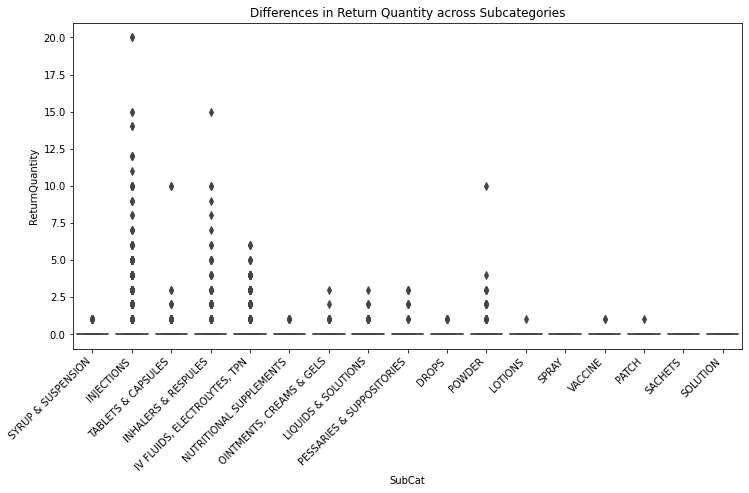
plt.figure(figsize=(12, 6))

sns.boxplot(x='SubCat', y='ReturnQuantity', data=df)

plt.title('Differences in Return Quantity across Subcategories')

plt.xticks(rotation=45, ha='right')

plt.show()



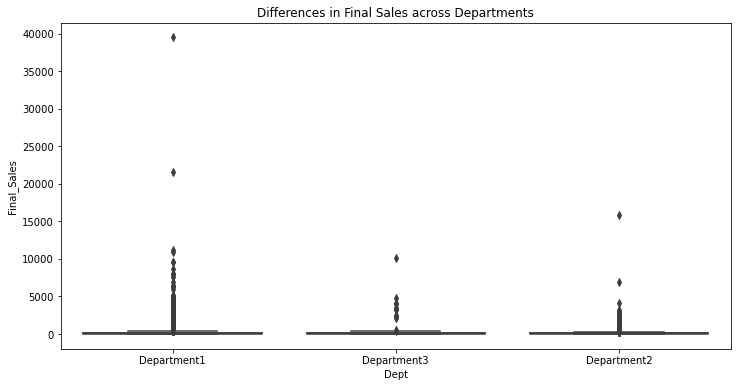
Differences in Final Sales across different Departments

plt.figure(figsize=(12, 6))

sns.boxplot(x='Dept', y='Final\_Sales', data=df)

plt.title('Differences in Final Sales across Departments')

plt.show()



**Multi Variant Analysis:**

Correlation heatmap for numerical columns

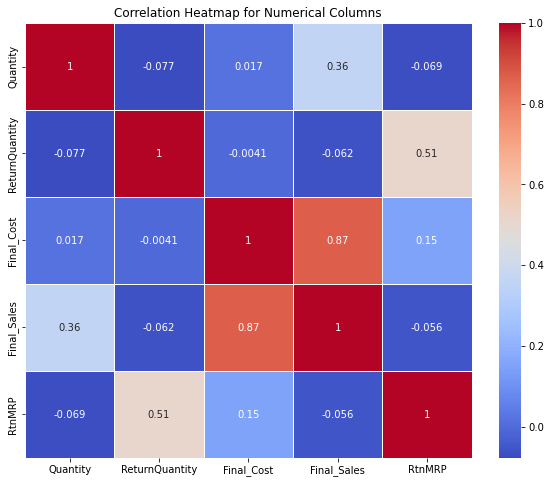
correlation\_matrix = df[['Quantity', 'Returnquantity', 'Final\_cost', 'Final\_Sales', 'RtnMRP']].corr()

plt.figure(figsize=(10, 8))

sns.heatmap(correlation\_matrix, annot=True, cmap='coolwarm', linewidths=0.5)

plt.title('Correlation Heatmap for Numerical Columns')

plt.show()



How does the Specialisation or Dept impact the Final\_Sales or Returnquantity?

# Bar plot for average Final\_Sales by Specialisation

avg\_sales\_by\_specialisation = df.groupby('Specialisation')['Final\_Sales'].mean().sort\_values(ascending=False)

plt.figure(figsize=(10, 6))

sns.barplot(x=avg\_sales\_by\_specialisation.index, y=avg\_sales\_by\_specialisation.values)

plt.title('Average Final\_Sales by Specialisation')

plt.xticks(rotation=45, ha='right')

plt.show()

# Bar plot for average Returnquantity by Dept

avg\_return\_quantity\_by\_dept = df.groupby('Dept')['ReturnQuantity'].mean().sort\_values(ascending=False)

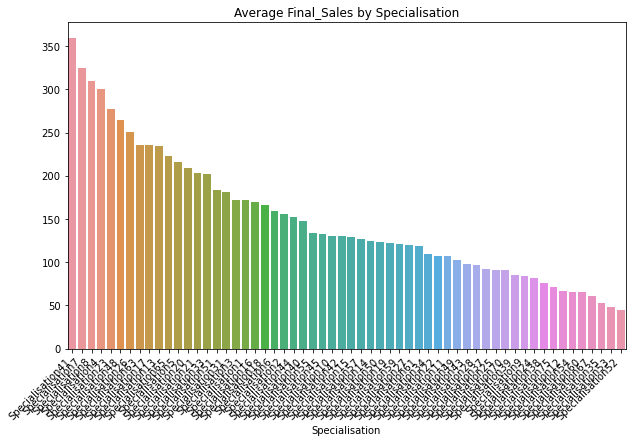
plt.figure(figsize=(10, 6))

sns.barplot(x=avg\_return\_quantity\_by\_dept.index, y=avg\_return\_quantity\_by\_dept.values)

plt.title('Average Returnquantity by Dept')

plt.xticks(rotation=45, ha='right')

plt.show()





Department 1 has highest average returns and specialization41 has highest average sales

ANOVA Analysis to check if month effect sales

import pandas as pd

from scipy.stats import f\_oneway

# One-way ANOVA

dept\_groups = [df['Final\_Sales'][df['month'] == dept] for dept in df['month'].unique()]

# Perform one-way ANOVA

anova\_result = f\_oneway(\*dept\_groups)

# Display the ANOVA result

print("ANOVA Result:")

print(anova\_result)

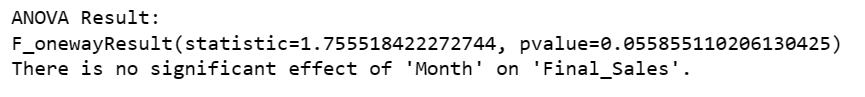
# Check if the p-value is less than a significance level (e.g., 0.05) to determine significance

if anova\_result.pvalue < 0.05:

print("There is a significant effect of 'Month' on 'Final\_Sales'.")

else:

print("There is no significant effect of 'Month' on 'Final\_Sales'.")



Which subcategory has more return items

import pandas as pd

import seaborn as sns

import matplotlib.pyplot as plt

# Filter rows where Typeofsales is 'return'

returns\_df = df[df['Typeofsales'] == 'Return']

# Count the returns for each subcategory

returns\_by\_subcategory = returns\_df['SubCat'].value\_counts()

# Plot a bar chart

plt.figure(figsize=(12, 6))

sns.barplot(x=returns\_by\_subcategory.index, y=returns\_by\_subcategory.values, palette="viridis")

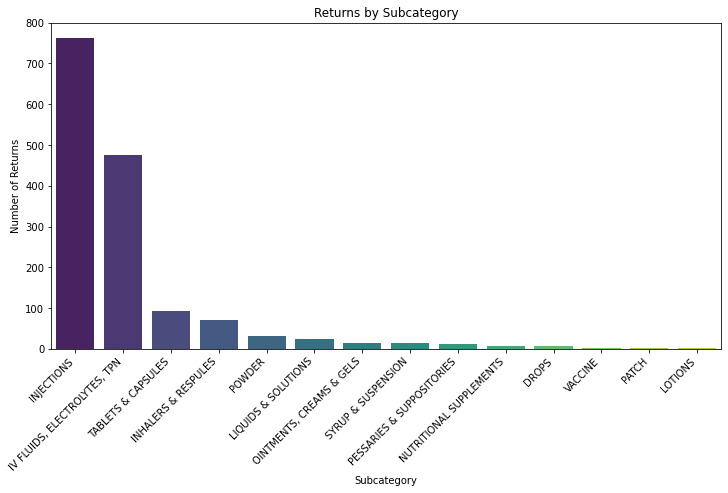
plt.title('Returns by Subcategory')

plt.xlabel('Subcategory')

plt.ylabel('Number of Returns')

plt.xticks(rotation=45, ha='right') # Rotate x-axis labels for better visibility

plt.show()



**insights:**

From Sub category Injections,tablets and IV fuilds, electrolytes,TPN has highest number of returns

Department 1 has highest average returns and specialization41 has highest average sales

Department 1 contains highest average sales in department compared to other two departments

Form1 has more return quantity , form3 has lowest return quantities

In sales we can see dec month has highest sales compared to other months

**Conclusion**

From the above analysis we can see the sub categories Injections and tablets and IV fuilds were returned frequently , so there may be some dissatisfaction with this products we need to check for those reasons for the items returned, the data we analyse does not consists that field so there is no correct finding for the reasons

We can these two categories injections, tablets, IV fluids return frequently and it costs lots of money, finding the reason for these returns we can reduce the amount of money lose to inventory

We can find highest average sales in December month and the highest returns in may month. Finding the reason, we can increase the sales in other months.