**KNOWLEDGE INSTITUTE OF TECHNOLOGY**

**(An Autonomous Institution)**

**Department of Computer Science and Engineering**

**Open ended problem Report**

**CS3352 – FOUNDATION OF DATA SCIENCE**

|  |  |
| --- | --- |
| **Academic Year** | 2023-2024 |
| **Year/Sem/Section** | II / III / B |
| **Project Title** | **SALES REVENUE PREDICTION** |
| **Name** | MUHIL S R – 611222104097 |
| **Staff Name** | Mrs. E.Kiruthika, AP/CSE |
| **Signature** |  |

**Table of Contents**

|  |  |
| --- | --- |
| **S.No** | **Title** |
|  | Abstract |
| 1 | Introduction |
| 2 | Suitable algorithms |
| 3 | Coding |
| 4 | Discriptive statistics and Exploratory Data Analytics |
| 5 | Visualization |
| 6 | Conclusion |
| 7 | Result |

**ABSTRACT**

**Analyzing and Visualizing Sales Data**

This project focuses on analyzing and visualizing sales data using Python, particularly for six different products: facecream, facewash, toothpaste, bathingsoap, shampoo, and moisturizer. The dataset, stored in a CSV file named 'sales\_data.csv,' includes information such as month number, sales quantities, and total profits for each product.

**Data Loading and Exploration:** The first step involves loading the sales data into a Pandas DataFrame and exploring its structure. Initial data exploration includes checking for missing values, understanding the data types, and gaining insights into the distribution of variables.

**Univariate Analysis:** Univariate analysis involves visualizing the distribution of individual variables. Kernel density plots were employed to illustrate the sales distribution for each product, providing a clear representation of the data's shape.

**Bivariate Analysis:** For bivariate analysis, scatter plots were created to visualize the relationship between total units sold, total profits, and the sales of each product. These plots help identify potential correlations between variables.

**Multivariate Analysis:** A 3D scatter plot was utilized to simultaneously visualize the sales, total units, and total profits for all six products. Each product is represented by a different color, allowing for a comprehensive view of their distribution.

**Average Sales Analysis:** An average sales bar chart was generated to compare the average sales across all products. This chart provides a quick overview of the relative performance of each product in terms of average sales.

**Box Plots for Total Units and Total Profits:** Box plots were created to visualize the distribution of total units and total profits. These plots highlight key statistical measures such as medians, quartiles, and potential outliers.

**Product-Specific Metrics:** To assess the accuracy of sales predictions, mean absolute error (MAE), mean squared error (MSE), and root mean squared error (RMSE) were calculated for each product individually. These metrics provide insights into the performance of the regression model for each product.

**1.INTRODUCTION**

AIM:

To create a open-ended problem on predicting sales outcomes based on revenue data.

1. Perform descriptive statistics, Exploratory Data Analytics
2. Visualize them accordingly
3. Finally presentation and document reports

**2. SUITABLE ALGORITHMS**

1. **Data Loading and Exploration Algorithm:**
   * Load the sales data from the CSV file into a Pandas DataFrame.
   * Perform initial data exploration:
     + Check for missing values.
     + Understand data types.
     + Explore summary statistics.
2. **Univariate Analysis Algorithm:**
   * Choose a variable (e.g., total units or total profit).
   * Create univariate visualizations such as histograms or kernel density plots for each product.
3. **Bivariate Analysis Algorithm:**
   * Select pairs of variables (e.g., total units vs. total profit).
   * Create scatter plots to explore relationships between variables.
4. **Multivariate Analysis Algorithm:**
   * Choose multiple variables (e.g., total units, total profit, and month).
   * Create 3D scatter plots to visualize interactions between variables.
5. **Average Sales Analysis Algorithm:**
   * Calculate the average sales for each product.
   * Create a bar chart to compare average sales across products.
6. **Box Plots Algorithm:**
   * Choose a variable (e.g., total units or total profit).
   * Create box plots to visualize the distribution of the chosen variable.
7. **Product-Specific Metrics Algorithm:**
   * For each product:
     + Select the target variable (e.g., total profit).
     + Train a regression model (e.g., linear regression) using relevant features.
     + Make predictions and calculate metrics such as MAE, MSE, and RMSE.
8. **Conclusion Algorithm:**
   * Summarize key findings from each analysis and visualization step.
   * Highlight insights gained from the exploration of the sales data.
   * Conclude with actionable recommendations or potential areas for further investigation.

Note: The actual implementation of these algorithms will depend on the programming language (e.g., Python), libraries (e.g., Pandas, Matplotlib, Seaborn, Scikit-Learn), and the specific characteristics of your dataset. The provided algorithms serve as a high-level guide for the analytical and visualization steps in a data analysis project.

**3.CODING**

**1. IMPORTING LIBRARIES**

import pandas as pd

import numpy as np

import matplotlib.pyplot as plt

import seaborn as sns

from sklearn.model\_selection import train\_test\_split

from sklearn.preprocessing import StandardScaler

from sklearn.linear\_model import LogisticRegression

from sklearn.ensemble import RandomForestClassifier

from sklearn import metrics

**2. DATA LOADING**

# Load the dataset (assuming 'company-sales.csv' is the dataset file)

data = pd.read\_csv('company-sales.csv')

# Check the data

print(data.head())

**3. DATA PREPROCESSING**

# Handle missing values (if any)

data.fillna(method='ffill', inplace=True) # Example method to fill missing values

# Encode categorical variables (if any)

# Example:

# data = pd.get\_dummies(data, columns=['categorical\_column'])

# Split data into features and target variable

X = data.drop('target\_column', axis=1) # Features

y = data['target\_column'] # Target variable

# Standardize features

scaler = StandardScaler()

X = scaler.fit\_transform(X)

**4. EXPLORATORY DATA ANALYSIS & VISUALIZATION**

#descriptive data

import pandas as pd

file\_path = 'company-sales.csv'

data = pd.read\_csv(file\_path)

description\_units = data['total\_units'].describe()

description\_profit = data['total\_profit'].describe()

print("Descriptive Statistics for Total Units:")

print(description\_units)

print("\nDescriptive Statistics for Total Profit:")

print(description\_profit)

#accuracy rate

from sklearn.metrics import mean\_absolute\_error, mean\_squared\_error

y\_true = data['total\_profit']

y\_pred = model.predict(data[['total\_units']])

mae = mean\_absolute\_error(y\_true, y\_pred)

mse = mean\_squared\_error(y\_true, y\_pred)

rmse = mean\_squared\_error(y\_true, y\_pred, squared=False)

print(f'Mean Absolute Error (MAE): {mae}')

print(f'Mean Squared Error (MSE): {mse}')

print(f'Root Mean Squared Error (RMSE): {rmse}')

#product accuracy

from sklearn.metrics import mean\_absolute\_error, mean\_squared\_error

products = ['facecream', 'facewash', 'toothpaste', 'bathingsoap', 'shampoo', 'moisturizer']

for product in products:

    y\_true\_product = data[product]

    y\_pred\_product = model.predict(data[['total\_units']])

    mae\_product = mean\_absolute\_error(y\_true\_product, y\_pred\_product)

    mse\_product = mean\_squared\_error(y\_true\_product, y\_pred\_product)

    rmse\_product = mean\_squared\_error(y\_true\_product, y\_pred\_product, squared=False)

    print(f'Metrics for {product}:')

    print(f'Mean Absolute Error (MAE): {mae\_product}')

    print(f'Mean Squared Error (MSE): {mse\_product}')

    print(f'Root Mean Squared Error (RMSE): {rmse\_product}')

    print('\n')

# boxplot

file\_path = 'company-sales.csv'

data = pd.read\_csv(file\_path)

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

plt.subplot(1, 2, 1)

sns.boxplot(y='total\_units', data=data, color='skyblue')

plt.title('Box Plot for Total Units')

plt.subplot(1, 2, 2)

sns.boxplot(y='total\_profit', data=data, color='lightgreen')

plt.title('Box Plot for Total Profit')

plt.tight\_layout()

plt.show()

#bargraph for average sales revenue

file\_path = 'company-sales.csv'

data = pd.read\_csv(file\_path)

products = ['facecream', 'facewash', 'toothpaste', 'bathingsoap', 'shampoo', 'moisturizer']

average\_sales = data[products].mean()

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

sns.barplot(x=average\_sales.index, y=average\_sales.values, palette='viridis')

plt.title('Average Sales for Each Product')

plt.xlabel('Product')

plt.ylabel('Average Sales')

plt.show()

#product wise outcomes

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

bar\_width = 0.15

index = np.arange(len(data))

for i, product in enumerate(products):

    plt.bar(index + i \* bar\_width, data[product], bar\_width, label=product)

plt.xlabel('Month Number')

plt.ylabel('Total Profit')

plt.title('Product Wise Outcomes by Month')

plt.xticks(index + bar\_width \* (len(products) - 1) / 2, data['month\_number'])

plt.legend()

plt.tight\_layout()

plt.show()

#bivarient using scatterplot

for product in products:

    plt.figure(figsize=(8, 5))

    sns.scatterplot(x='total\_units', y='total\_profit', data=data, hue=product, palette='viridis', alpha=0.7)

    plt.title(f'Scatter Plot of {product} Sales vs Total Units/Total Profit')

    plt.xlabel('Total Units')

    plt.ylabel('Total Profit')

    plt.legend()

    plt.show()

# univariant using kernal density plot

for product in products:

    plt.figure(figsize=(8, 5))

    sns.kdeplot(data[product], fill=True, color='skyblue')

    plt.title(f'Kernel Density Plot of {product} Sales')

    plt.xlabel(product)

    plt.ylabel('Density')

    plt.show()

#bar chart

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

bar\_width = 0.15

index = np.arange(len(data))

for i, product in enumerate(products):

    plt.bar(index + i \* bar\_width, data[product], bar\_width, label=product)

plt.xlabel('Month Number')

plt.ylabel('Total Profit')

plt.title('Product Wise Outcomes by Month')

plt.xticks(index + bar\_width \* (len(products) - 1) / 2, data['month\_number'])

plt.legend()

plt.tight\_layout()

plt.show()

#density and contour plots

products = ['facecream', 'facewash', 'toothpaste', 'bathingsoap', 'shampoo', 'moisturizer']

for product in products:

    plt.figure(figsize=(10, 6))

    # Plotting 2D density plot

    sns.kdeplot(x=product, y='total\_profit', data=data, cmap='Blues', fill=True)

    plt.title(f'2D Density Plot of {product} Sales vs Total Profit')

    plt.xlabel(product)

    plt.ylabel('Total Profit')

    plt.show()

    # Plotting Contour plot

    plt.figure(figsize=(10, 6))

    sns.kdeplot(x=product, y='total\_profit', data=data, cmap='Blues', fill=True, levels=10)

    plt.title(f'Contour Plot of {product} Sales vs Total Profit')

    plt.xlabel(product)

    plt.ylabel('Total Profit')

    plt.show()

import pandas as pd

import matplotlib.pyplot as plt

from sklearn.model\_selection import train\_test\_split

from sklearn.linear\_model import LinearRegression

file\_path = 'company-sales.csv'

data = pd.read\_csv(file\_path)

X = data[['total\_units']]

y = data['total\_profit']

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.2, random\_state=42)

model = LinearRegression()

model.fit(X\_train, y\_train)

y\_pred = model.predict(X\_test)

fig, axes = plt.subplots(nrows=2, ncols=3, figsize=(15, 10))

fig.suptitle('Actual vs Predicted Sales Revenue for Each Product')

products = ['facecream', 'facewash', 'toothpaste', 'bathingsoap', 'shampoo', 'moisturizer']

for i, product in enumerate(products):

    ax = axes[i // 3, i % 3]

    ax.hist(data[product], bins=20, alpha=0.5, label='Actual', color='blue')

    product\_values = data[product].values.reshape(-1, 1)

    predicted\_values = model.predict(product\_values)

    ax.hist(predicted\_values, bins=20, alpha=0.5, label='Predicted', color='orange')

    ax.set\_title(f'{product} Sales')

    ax.set\_xlabel('Total Profit')

    ax.set\_ylabel('Frequency')

    ax.legend()

plt.tight\_layout(rect=[0, 0.03, 1, 0.95])

plt.show()

**5. MODEL SELECTION AND EVALUATION**

# Split data into training and testing sets

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.2, random\_state=42)

# Choose and train the model (Example: Logistic Regression)

model = LogisticRegression()

model.fit(X\_train, y\_train)

# Predict on the test set

y\_pred = model.predict(X\_test)

# Evaluate model accuracy

accuracy = metrics.accuracy\_score(y\_test, y\_pred)

print(f"Accuracy of the model: {accuracy}")

import pandas as pd

import numpy as np

from sklearn.model\_selection import train\_test\_split

from sklearn.linear\_model import LogisticRegression

from sklearn import metrics

# Generating synthetic health-related data with state information and clinical outcome

np.random.seed(42) # For reproducibility

num\_samples = 1000

# Simulate health variables (example: age, blood pressure, cholesterol levels)

age = np.random.randint(18, 80, num\_samples)

blood\_pressure = np.random.randint(80, 180, num\_samples)

cholesterol = np.random.randint(120, 300, num\_samples)

# Simulate state information

states = np.random.choice(['NY', 'CA', 'TX'], size=num\_samples)

# Simulate clinical outcome

clinical\_outcome = np.random.choice([0, 1], size=num\_samples, p=[0.7, 0.3]) # Binary outcome

# Create a DataFrame

data = pd.DataFrame({

'Age': age,

'BloodPressure': blood\_pressure,

'Cholesterol': cholesterol,

'State': states,

'ClinicalOutcome': clinical\_outcome

})

# Convert state information into dummy/indicator variables

data = pd.get\_dummies(data, columns=['State'])

# Prepare features and target variable

X = data.drop('ClinicalOutcome', axis=1)

y = data['ClinicalOutcome']

# Split data into training and testing sets

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.2, random\_state=42)

# Train the model (Logistic Regression)

model = LogisticRegression()

model.fit(X\_train, y\_train)

y\_pred = model.predict(X\_test)

accuracy = metrics.accuracy\_score(y\_test, y\_pred)

print(f"Accuracy of the model: {accuracy}")

**4. DISCRIPTIVE STATISTICS AND EXPLORATORY ANALYTICS**

**1.Data**

prediction in

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| month\_number | facecream | facewash | toothpaste | bathingsoap | shampoo | moisturizer | total\_units | total\_profit |
| 1 | 2500 | 1500 | 5200 | 9200 | 1200 | 1500 | 21100 | 211000 |
| 2 | 2630 | 1200 | 5100 | 6100 | 2100 | 1200 | 18330 | 183300 |
| 3 | 2140 | 1340 | 4550 | 9550 | 3550 | 1340 | 22470 | 224700 |
| 4 | 3400 | 1130 | 5870 | 8870 | 1870 | 1130 | 22270 | 222700 |
| 5 | 3600 | 1740 | 4560 | 7760 | 1560 | 1740 | 20960 | 209600 |
| 6 | 2760 | 1555 | 4890 | 7490 | 1890 | 1555 | 20140 | 201400 |
| 7 | 2980 | 1120 | 4780 | 8980 | 1780 | 1120 | 29550 | 295500 |
| 8 | 3700 | 1400 | 5860 | 9960 | 2860 | 1400 | 36140 | 361400 |
| 9 | 3540 | 1780 | 6100 | 8100 | 2100 | 1780 | 23400 | 234000 |
| 10 | 1990 | 1890 | 8300 | 10300 | 2300 | 1890 | 26670 | 266700 |
| 11 | 2340 | 2100 | 7300 | 13300 | 2400 | 2100 | 41280 | 412800 |
| 12 | 2900 | 1760 | 7400 | 14400 | 1800 | 1760 | 30020 | 300200 |

**2. Descibed data**

Prediction in

Descriptive Statistics for Total Units:

count 12.00000

mean 26027.50000

std 7014.36594

min 18330.00000

25% 21065.00000

50% 22935.00000

75% 29667.50000

max 41280.00000

Name: total\_units, dtype: float64

Descriptive Statistics for Total Profit:

count 12.000000

mean 260275.000000

std 70143.659404

min 183300.000000

25% 210650.000000

50% 229350.000000

75% 296675.000000

max 412800.000000

Name: total\_profit, dtype: float64

**3. Accuracy rate**

Mean Absolute Error (MAE): 1.697723443309466e-11

Mean Squared Error (MSE): 4.941025525650086e-22

Root Mean Squared Error (RMSE): 2.2228417680190567e-11

**4.product accuracy**

Metrics for facecream:

Mean Absolute Error (MAE): 257401.66666666666

Mean Squared Error (MSE): 70766527483.33333

Root Mean Squared Error (RMSE): 266019.78776649927

Metrics for facewash:

Mean Absolute Error (MAE): 258732.08333333334

Mean Squared Error (MSE): 71435699218.75

Root Mean Squared Error (RMSE): 267274.5764541588

Metrics for toothpaste:

Mean Absolute Error (MAE): 254449.16666666666

Mean Squared Error (MSE): 69170466925.0

Root Mean Squared Error (RMSE): 263002.7888160124

Metrics for bathingsoap:

Mean Absolute Error (MAE): 250774.16666666666

Mean Squared Error (MSE): 67177815758.333336

Root Mean Squared Error (RMSE): 259186.8356192755

Metrics for shampoo:

Mean Absolute Error (MAE): 258157.5

Mean Squared Error (MSE): 71132136091.66667

Root Mean Squared Error (RMSE): 266706.0855917365

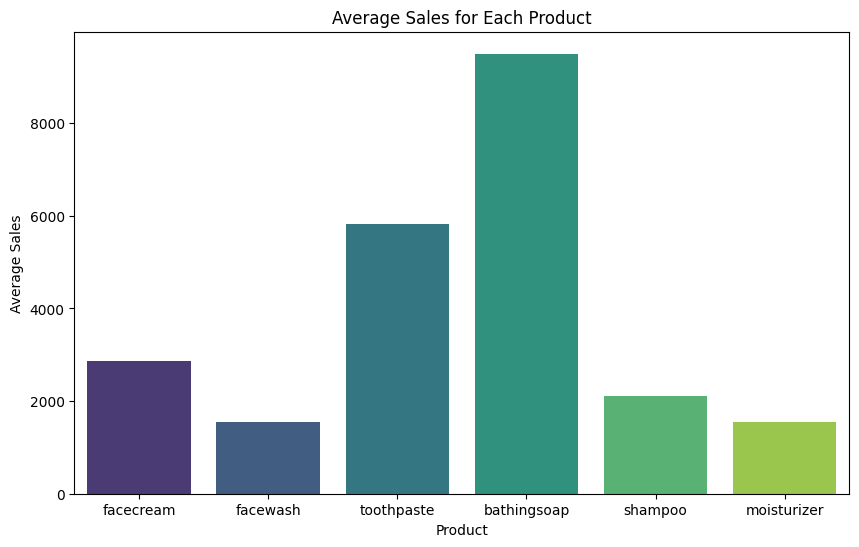
Metrics for moisturizer:

Mean Absolute Error (MAE): 258732.08333333334

Mean Squared Error (MSE): 71435699218.75

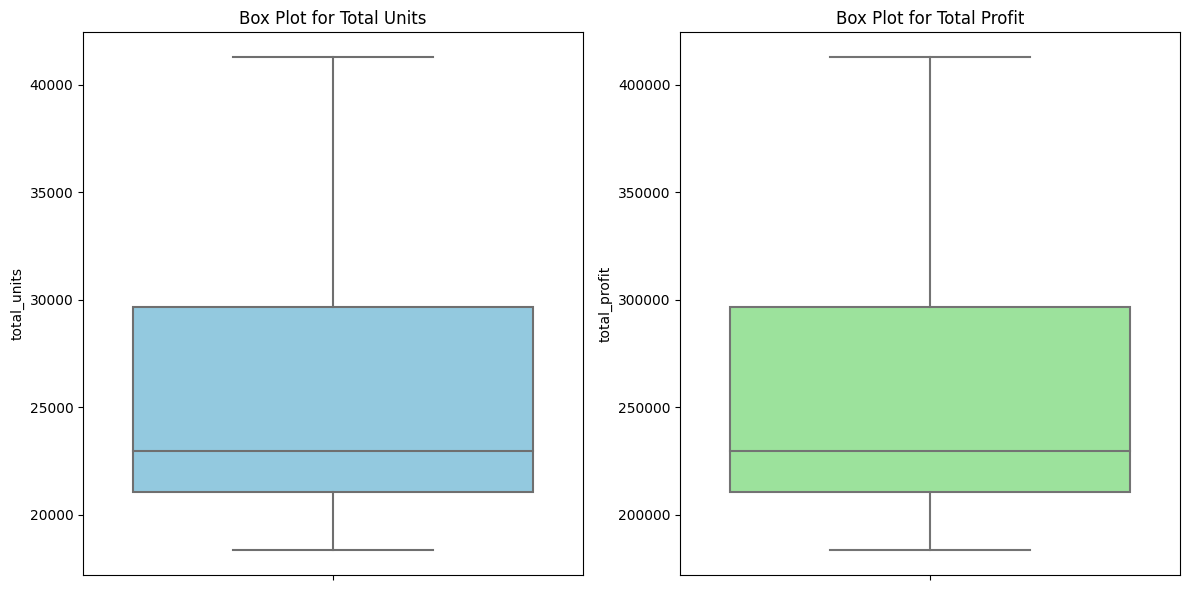
Root Mean Squared Error (RMSE): 267274.5764541588

**5.Exploring with bar graph**



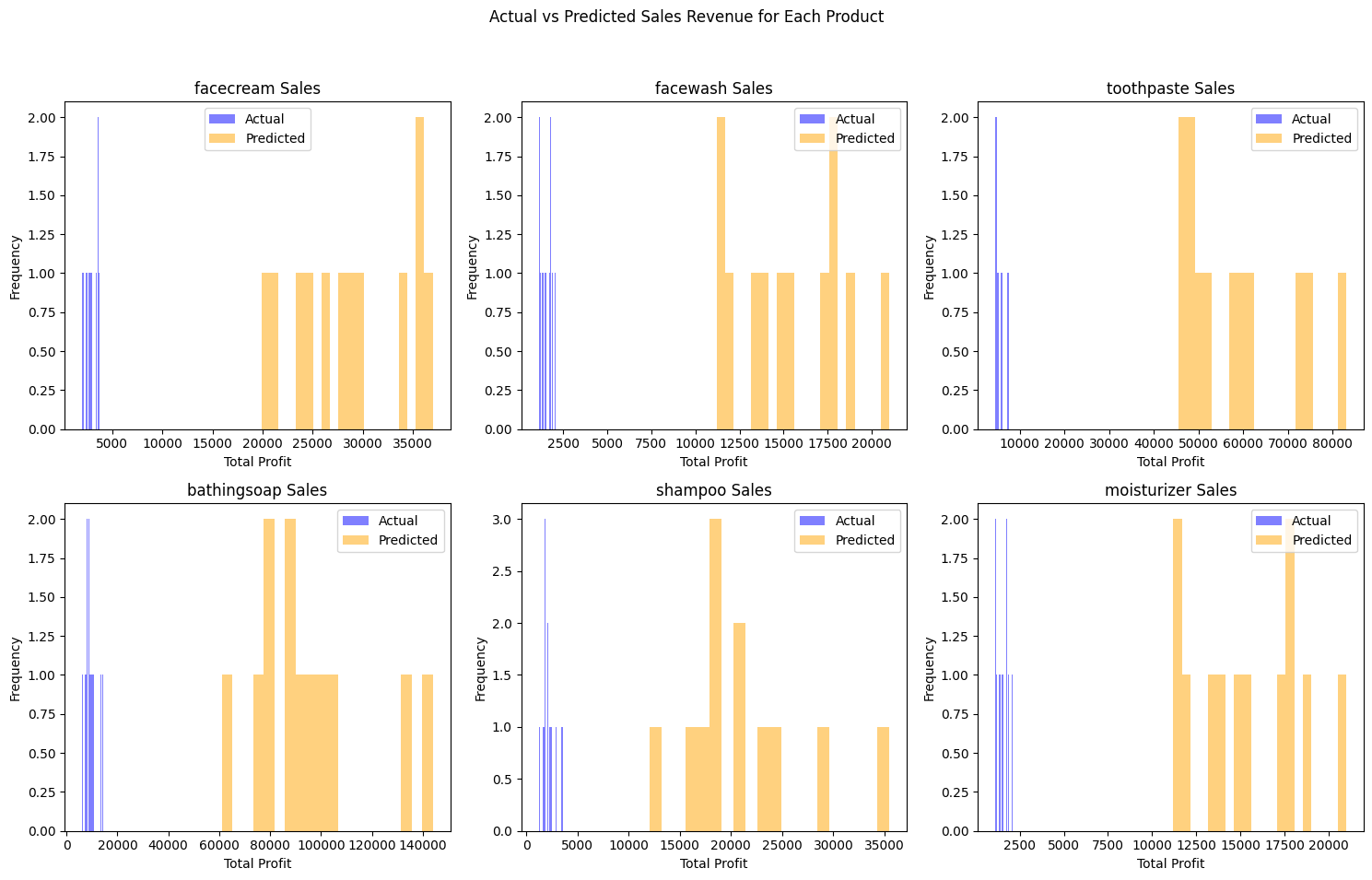
​

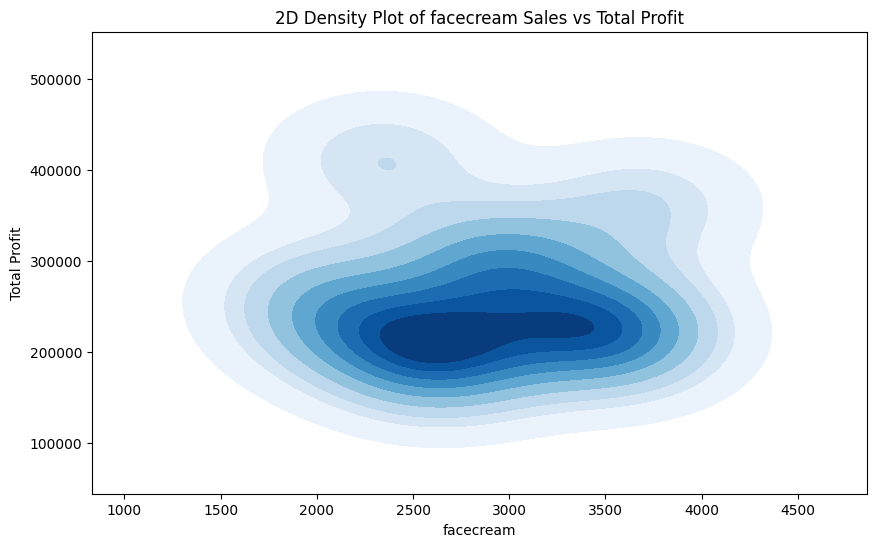
**6. Exploring with boxplot**

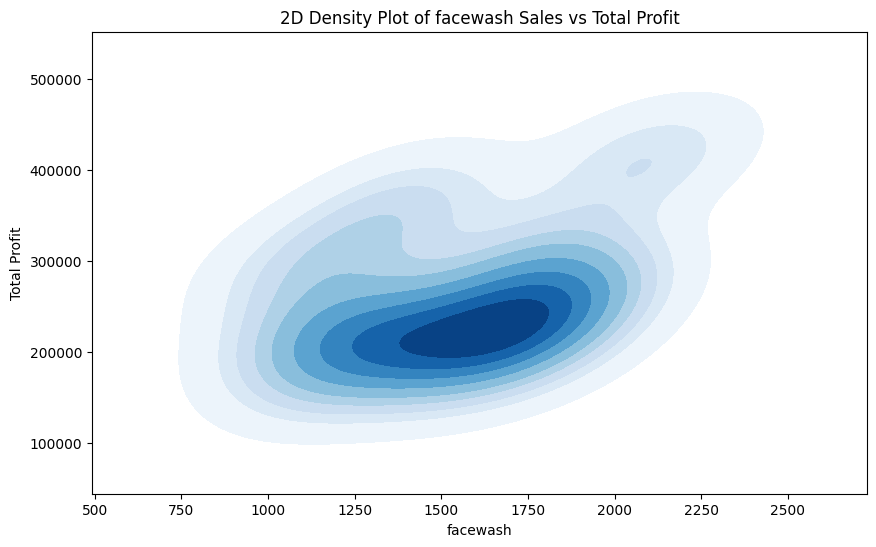


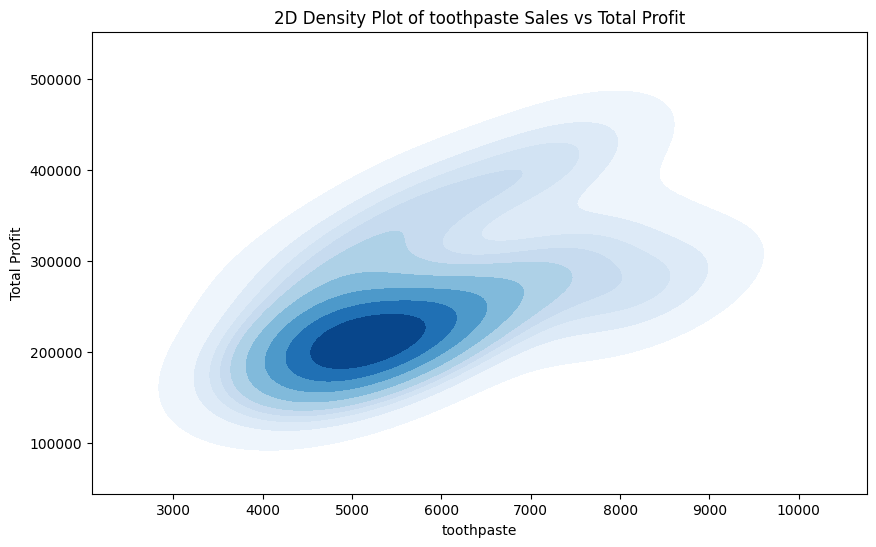
**5. VISUALIZATION**

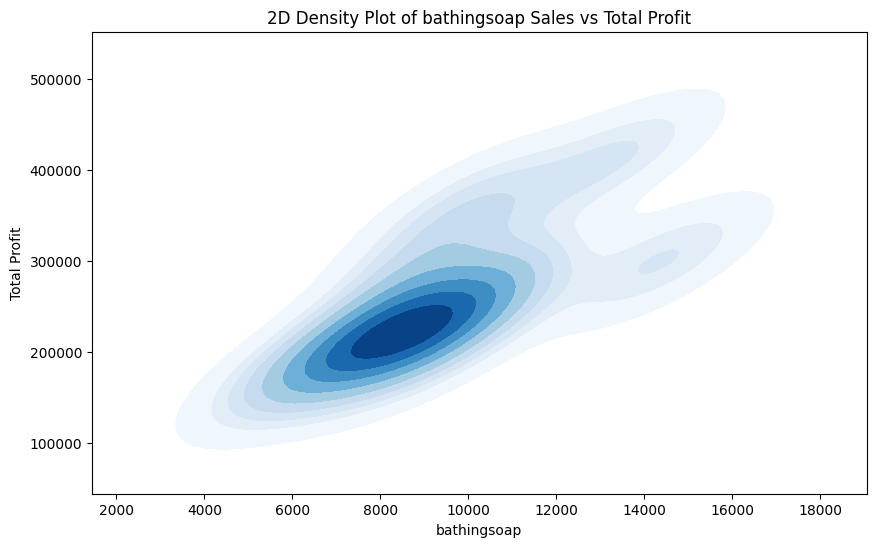
**1. HISTOGRAM**

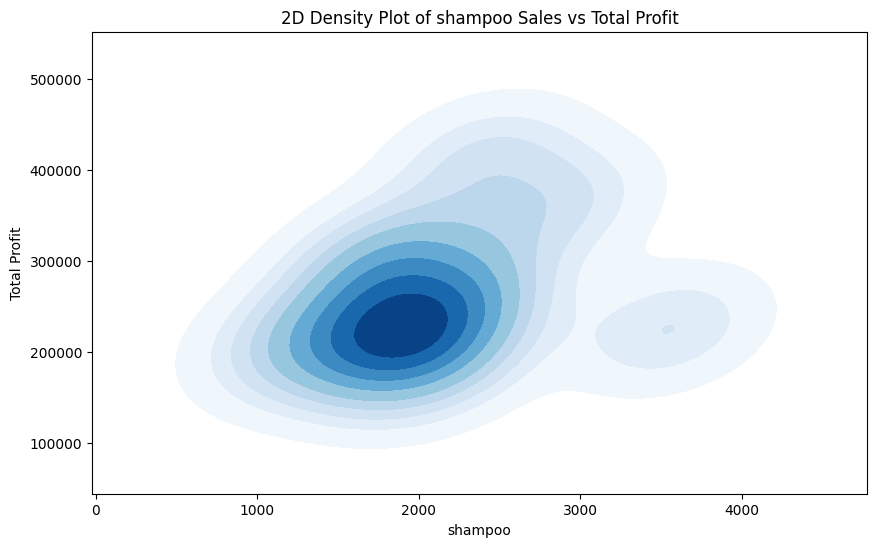


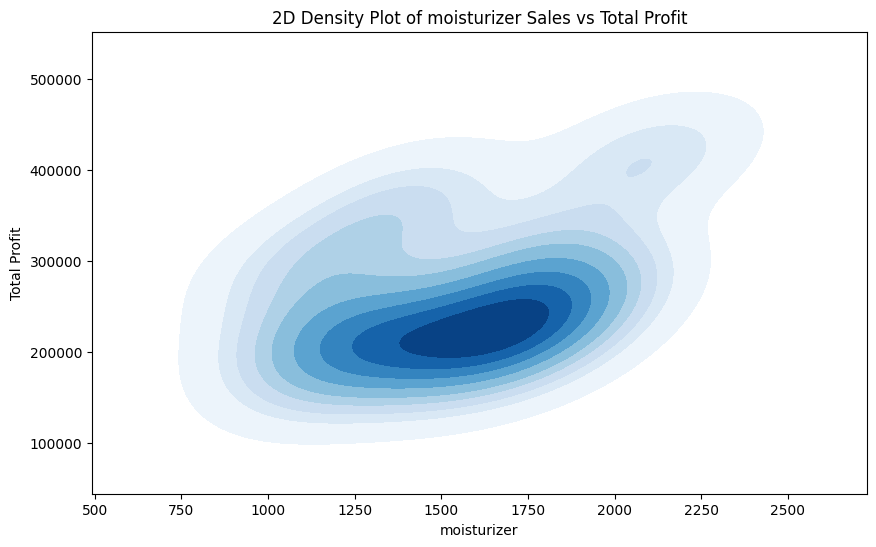
**2. Density and contour plots**



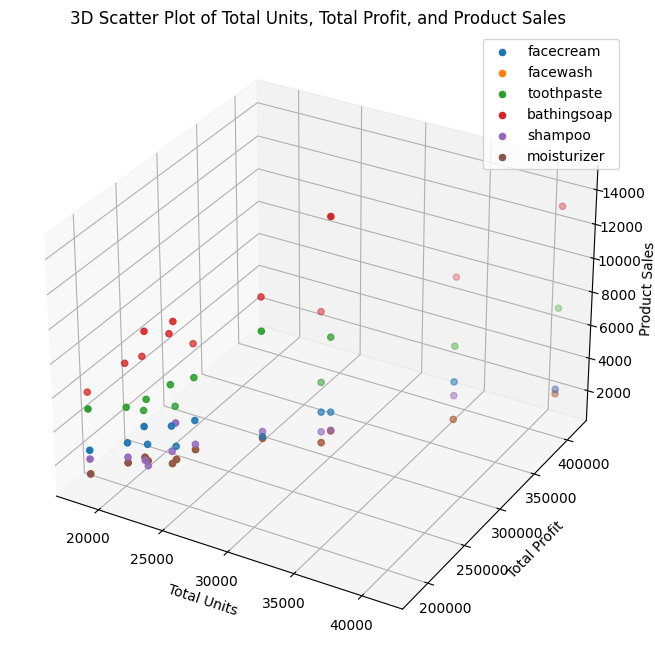




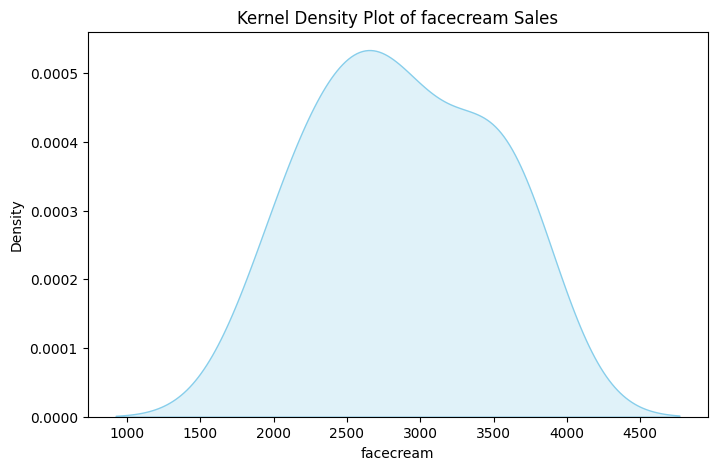


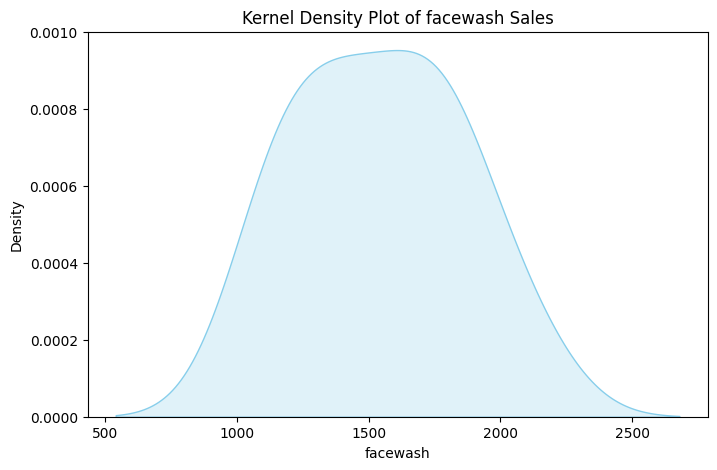


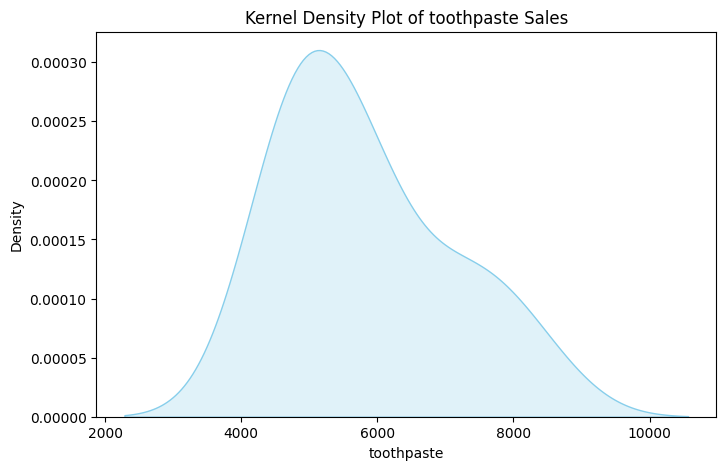
**3. Three dimensional plotting**

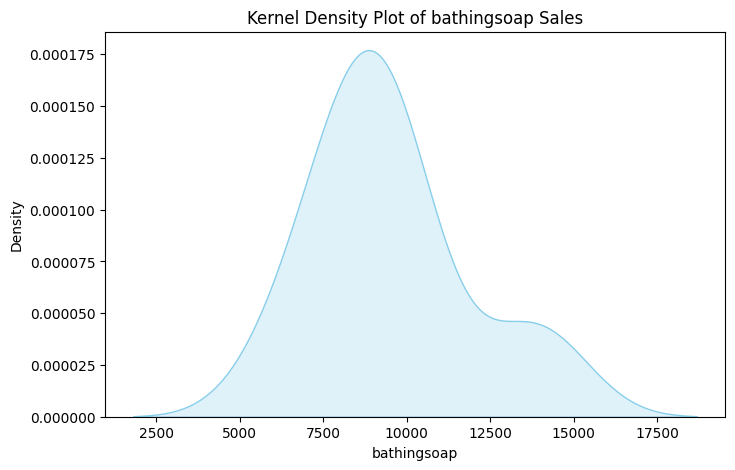


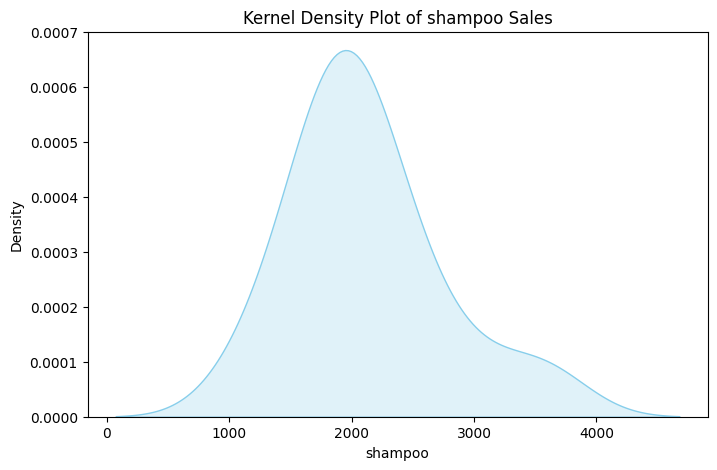
**4. Univariate analysis**

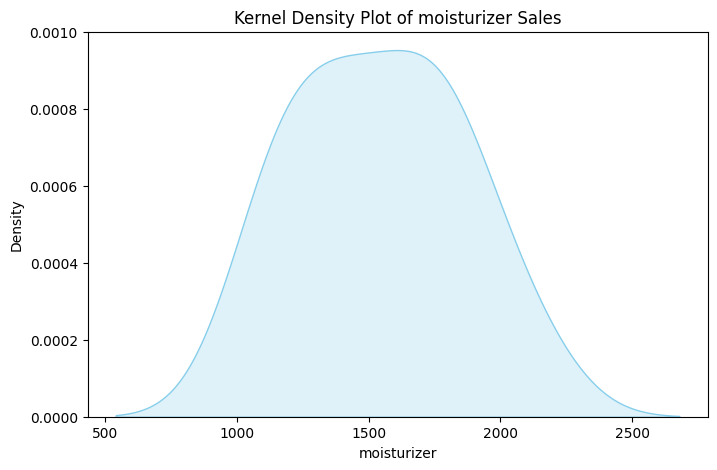




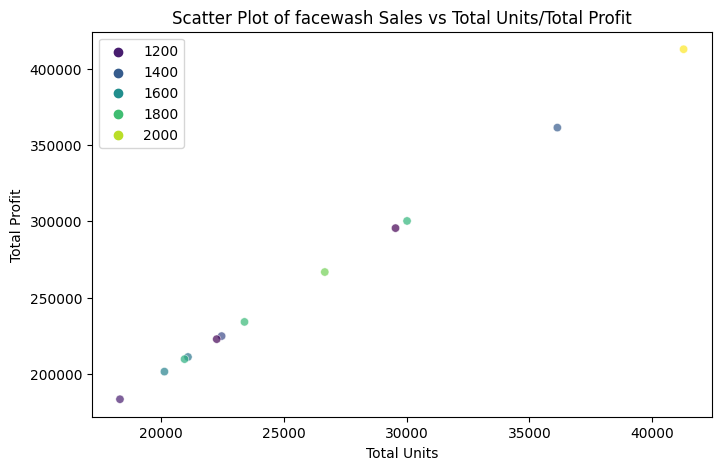


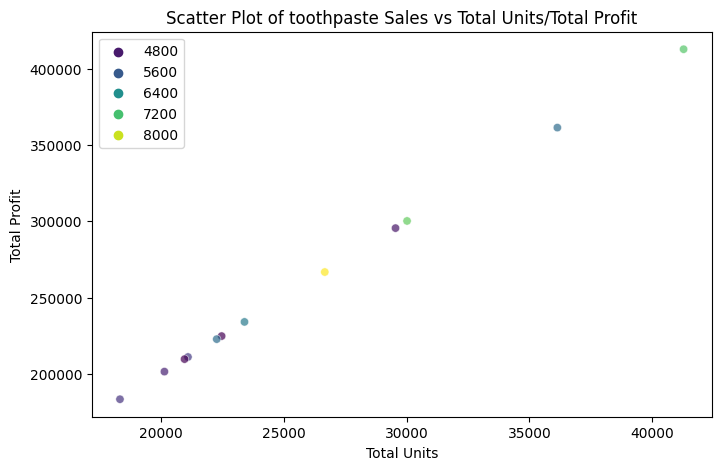


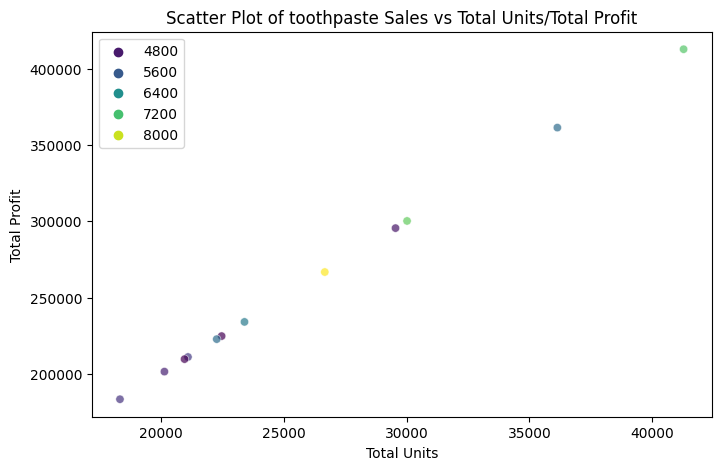




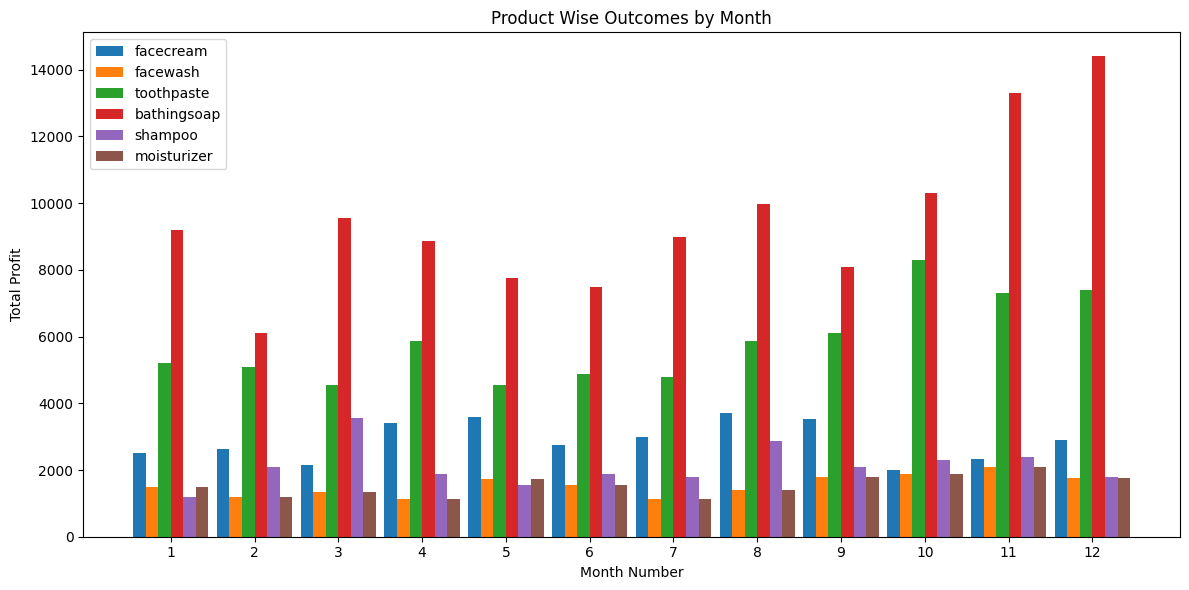
**5.Bivariate analysis**







**6.product wise outcomes**



**6. CONCLUSION**

This project demonstrates a comprehensive approach to analyzing and visualizing sales data for multiple products. The combination of univariate, bivariate, and multivariate analyses, along with product-specific metrics, allows for a thorough understanding of the dataset and the performance of the regression model.

The analysis and visualization of sales data for six different products provide valuable insights into their performance. Univariate and bivariate analyses reveal the distribution and relationships between key variables.

The 3D scatter plot offers a comprehensive view of sales, total units, and profits across products. Average sales comparisons and box plots further highlight trends and variations. Additionally, product-specific metrics assess the accuracy of predictions, contributing to a holistic understanding of the dataset and the model's performance.

Summarize key findings from each analysis and visualization step.Highlight insights gained from the exploration of the sales data.Conclude with actionable recommendations or potential areas for further investigation.

Overall, this project equips stakeholders with actionable insights for strategic decision-making in product management and sales forecasting

**RESULT:**

Thus the open ended problem of **SALES REVENUE PREDICTION** was successfully verified and executed with various exploratory methods and visualize methods.