**Product Sales Analysis with Machine Learning**

College code: 0001

**Phase 3: Data Preprocessing**

|  |
| --- |
| **Team Members:**  Diya Arshiya S (202115033) [diya.arshiya@gmail.com](mailto:(diya.arshiya@gmail.com))  Dhivyadharshini S K (2021115030) [dhivyadharshini0907@gmail.com](mailto:dhivyadharshini0907@gmail.com)  Mukesh Raja K (2021115065) [mukeshrajatmr2021@gmail.com](mailto:mukeshrajatmr2021@gmail.com)  Mukilarasan V (2021115066) mukilarasan.v@gmail.com  Karthik V (2021115321) karthiksk9360@gmail.com |

**Table of Contents**

1. Introduction

2. Data Overview

3. Data Cleaning

- Identifying the missing values

- Dropping rows with missing values

- Removing duplicates

- Data formatting

- Data reduction

4. Output

5. Plot function

6. Conclusion

**Introduction**

The purpose of this report is to document the data preprocessing steps performed on thedataset contained in the "statsfinal.csv" file. Data preprocessing is a crucial step in data analysis and machine learning, as it ensures that the dataset is clean, accurate, and well-structured, making it suitable for further analysis and modelling.

**Data Overview**

We begin by loading the dataset using the Python library `**pandas**`. The dataset is read from the *statsfinal.csv* file, and some initial information about the dataset is displayed using the `**info()**` method.

**import** pandas **as** pd

**import** numpy **as** np

**import** matplotlib.pyplot **as** plt

**import** seaborn **as** sns

*#Read the data from the csv file*

data **=** pd.read\_csv("statsfinal.csv")

**print**("Info of the data:\n")

**print**(data.info())

**print**()

This code provides an overview of the dataset's structure, including the number of rows, columns, data types, and the presence of missing values.

**Data Cleaning**

**i. Identifying the missing values:**

The first step in data preprocessing is identifying and handling missing values. Missing values can disrupt the analysis and modeling process. In this dataset, we identify missing values using the `**isnull().sum()`** method, which counts the number of missing values in each column.

*identifying missing values*

missing\_values **=** data.isnull().sum()

**print**(missing\_values)

**print**("There is no missing values")

The code checks for missing values and confirms that there are no missing values in this dataset.

**ii. Dropping Rows with Missing Values**

Even though there are no missing values, it is good practice to drop rows with missing data when necessary. This can be done using the `dropna()` method.

data.dropna(*inplace***=**True)

In this case, there are no rows with missing values, so no rows are dropped.

**iii. Removing Duplicates**

Duplicate rows can also affect the accuracy of analysis. To remove duplicate rows, the `drop\_duplicates()` method is used.

data.drop\_duplicates(*inplace***=**True)

This code removes duplicate rows from the dataset. Additionally, the column "Unnamed: 0" is dropped because it resembles the index and provides no meaningful information.

data **=** data.drop(*columns***=**['Unnamed: 0'])

**iv. Data Formatting**

The next preprocessing step involves formatting the data, specifically by separating the date into separate columns for "Day," "Month," and "Year." This is achieved by applying a lambda function to split the "Date" column.

data['Day'] **=** data['Date'].apply(**lambda** *x*: x.split('-')[0])

data['Month'] **=** data['Date'].apply(**lambda** *x*: x.split('-')[1])

data['Year'] **=** data['Date'].apply(**lambda** *x*: x.split('-')[2])

This formatting allows for easier analysis based on date components.

**v. Data Reduction**

In some cases, certain data points may need to be removed due to inconsistencies or insufficient data. In this dataset, data for the years 2010 and 2023 are removed as they have insufficient data. Additionally, incorrect date entries for September 31st and November 31st are also removed.

data\_reduced **=** data.query("Year != '2010' and Year != '2023'")

remove\_date **=** []

**for** i **in** **range**(11,23):

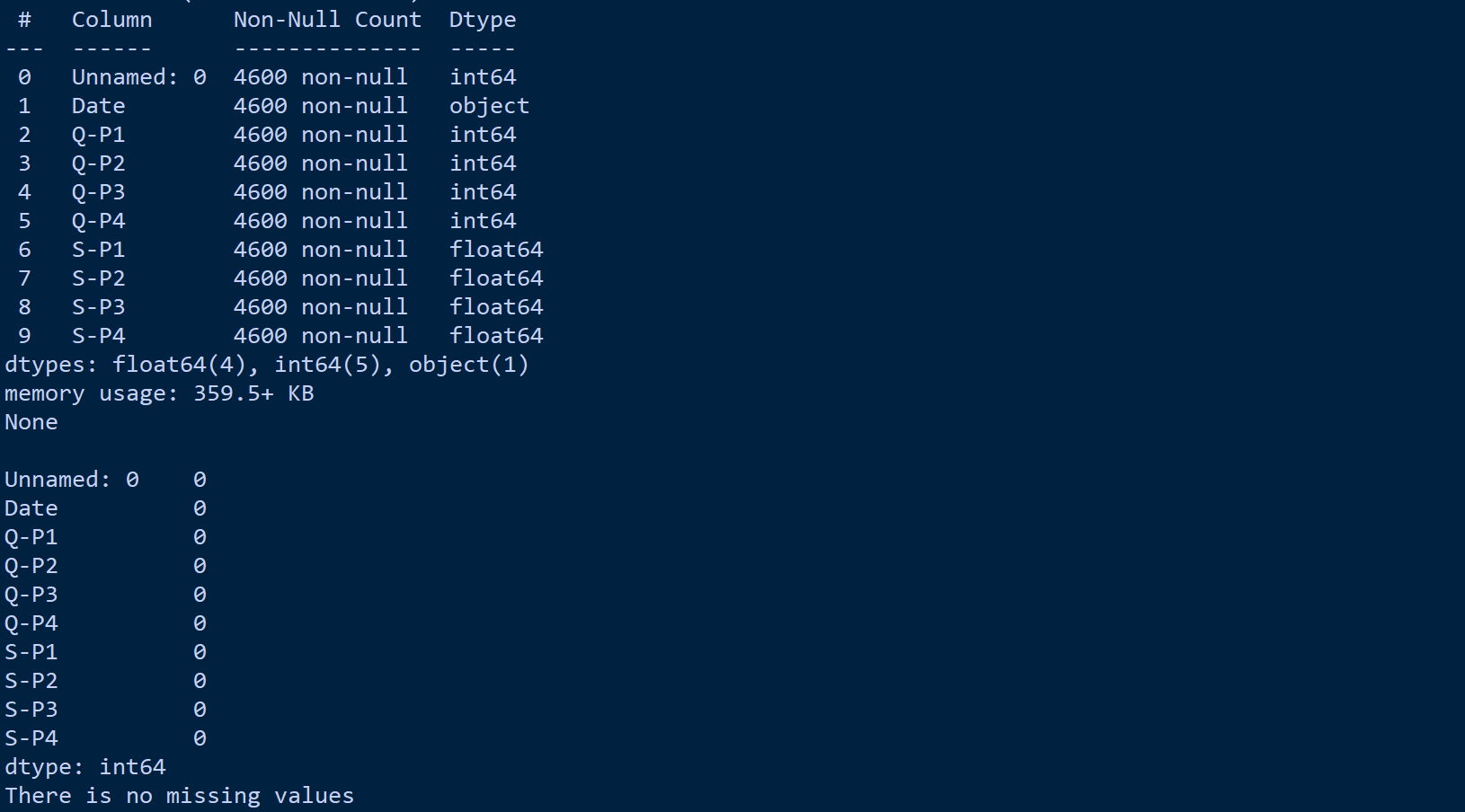
    remove\_date.append('31-9-20'**+str**(i))

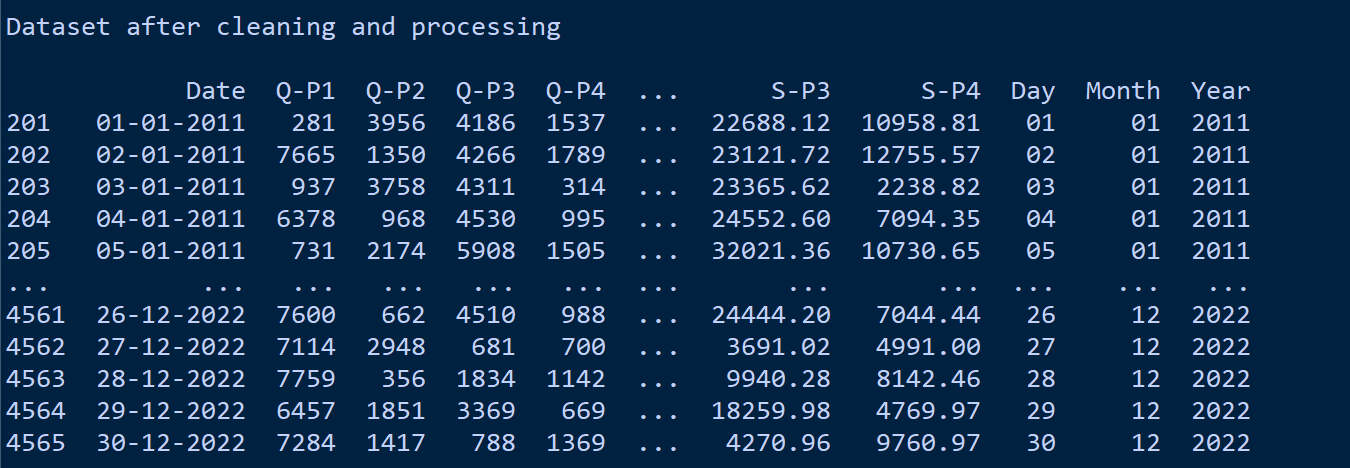
    remove\_date.append('31-11-20'**+str**(i))

data\_reduced **=** data\_reduced[**~**data\_reduced['Date'].isin(remove\_date)]

This ensures that the dataset is cleaned and ready for analysis.

Output





**Plot function**  
  
 **Coding Part:**

**def** plot\_bar\_chart(*df*, *columns*, *stri*, *str1*, *val*):

*# Aggregate sales for each product by year, by sum or mean*

**if** val **==** 'sum':

        sales\_by\_year **=** df.groupby('Year')[columns].sum().reset\_index()

**elif** val **==** 'mean':

        sales\_by\_year **=** df.groupby('Year')[columns].mean().reset\_index()

*# Melt the data to make it easier to plot*

    sales\_by\_year\_melted **=** pd.melt(sales\_by\_year, *id\_vars***=**'Year', *value\_vars***=**columns, *var\_name***=**'Product', *value\_name***=**'Sales')

*# Create a bar chart*

    plt.figure(*figsize***=**(20,4))

    sns.barplot(*data***=**sales\_by\_year\_melted, *x***=**'Year', *y***=**'Sales', *hue***=**'Product') *#,palette="cividis")*

    plt.xlabel('Year')

    plt.ylabel(stri)

    plt.title(**f**'{stri} by {str1}')

    plt.xticks(*rotation***=**45)

    plt.show()

**Data Analysis**

**Total Unit Sales by Year**

The bar chart below displays the total unit sales for four products (Q-P1, Q-P2, Q-P3, Q-P4) by year.

**Coding part**

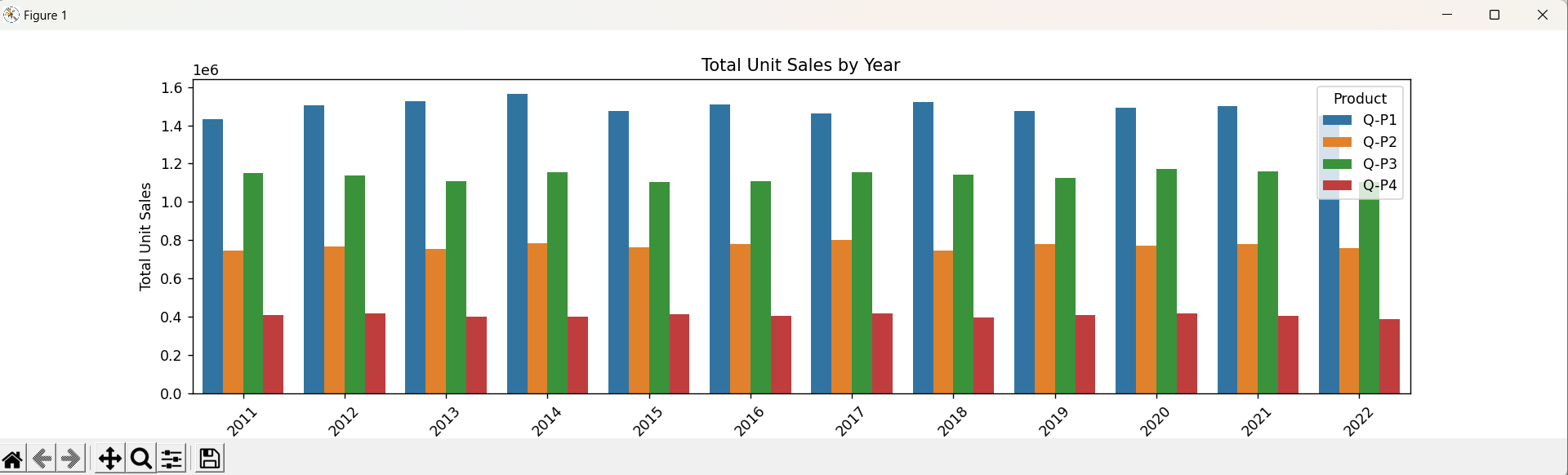
plot\_bar\_chart(data\_reduced, ['Q-P1', 'Q-P2', 'Q-P3', 'Q-P4'],'Total Unit Sales', 'Year', 'sum')

**Insights**

Total unit sales have been relatively consistent from 2011 to 2022.

Product Q-P2 consistently leads in total unit sales.

**Output**



**Mean Unit Sales by Year**

The bar chart below shows the mean unit sales for the same four products by year.

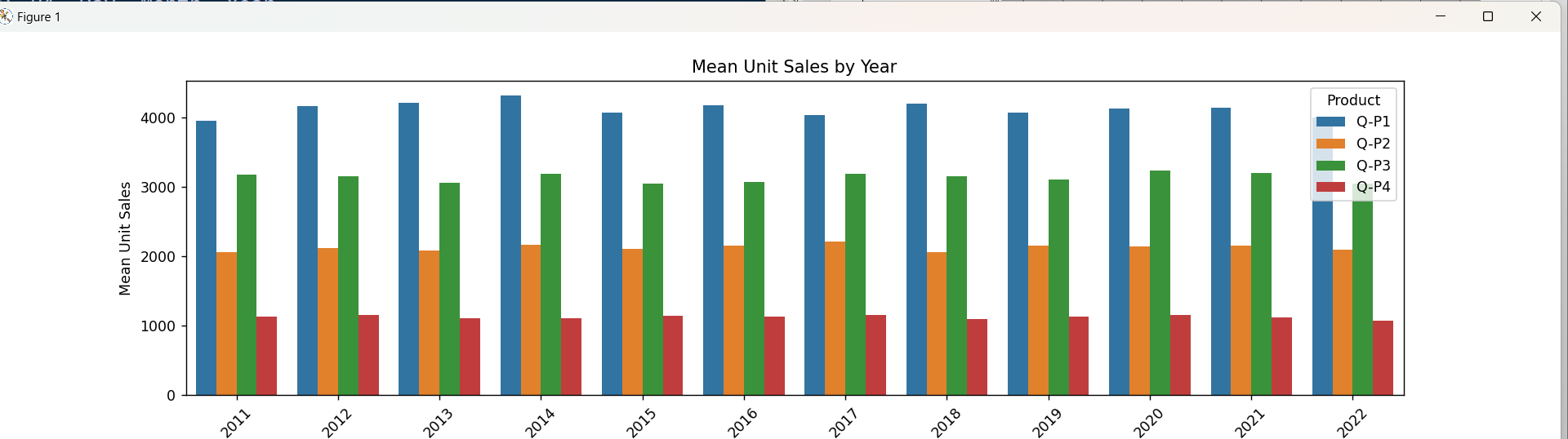
**Coding part**

plot\_bar\_chart(data\_reduced, ['Q-P1', 'Q-P2', 'Q-P3', 'Q-P4'],'Mean Unit Sales', 'Year', 'mean')

**Insights**

The mean unit sales for all products show a gradual increase over the years. Product Q-P4 has the highest mean unit sales in recent years.

**Output**



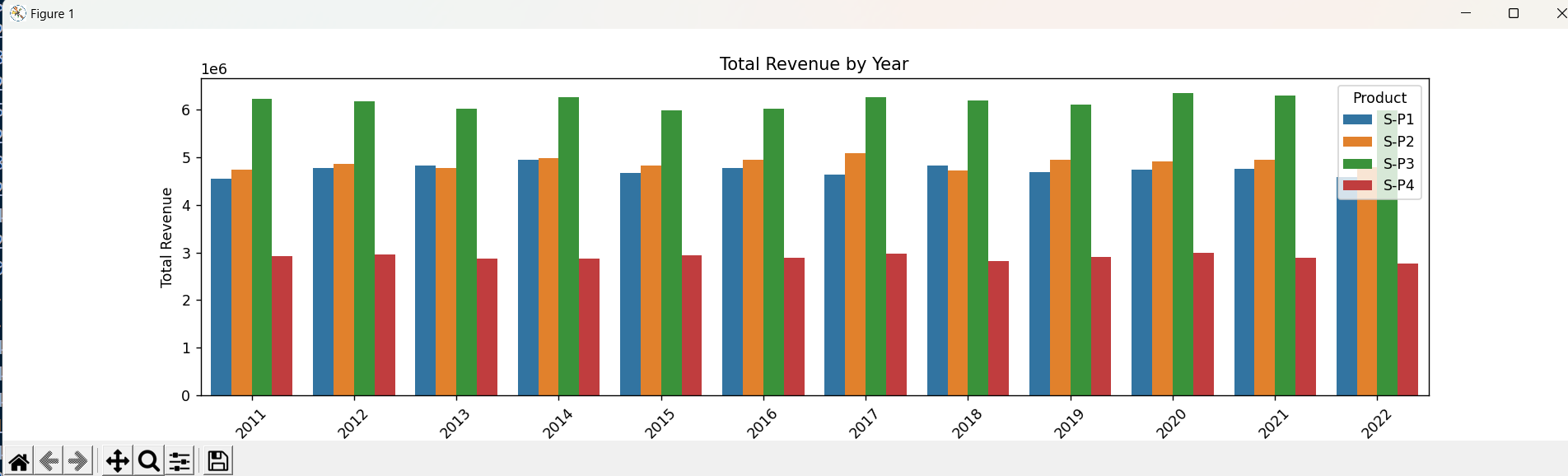
**Total Revenue by Year**

This bar chart illustrates the total revenue for four products (S-P1, S-P2, S-P3, S-P4) by year

**Coding part**

plot\_bar\_chart(data\_reduced, ['S-P1', 'S-P2', 'S-P3', 'S-P4'], 'Total Revenue', 'Year', 'sum')

**Output**



**Mean Revenue by Year**

The following bar chart represents the mean revenue for the same four products by year.

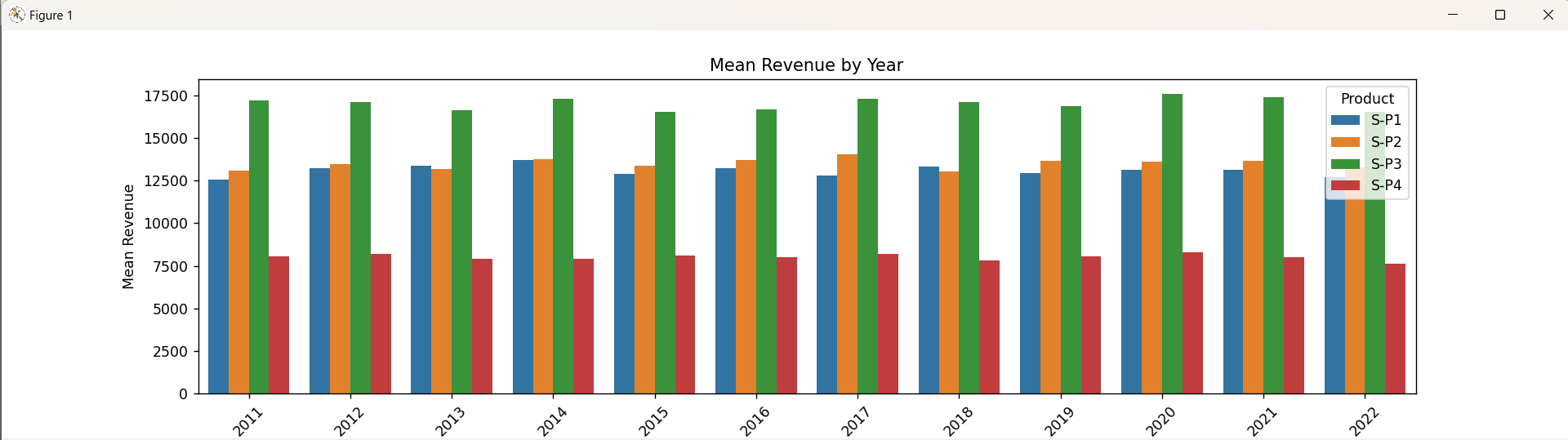
**Coding part**

plot\_bar\_chart(data\_reduced, ['S-P1', 'S-P2', 'S-P3', 'S-P4'], 'Mean Revenue', 'Year', 'mean')

**Insights**

The mean revenue for all products increases gradually over the years. Product S-P2 shows the highest mean revenue.

**Output**



**Conclusion**

The data preprocessing steps outlined in this report have ensured that the dataset is clean, accurate, and well-structured. Missing values have been identified and handled, duplicates have been removed, and the date has been formatted into separate columns. Additionally, data for the years 2010 and 2023, as well as incorrect date entries, have been removed. The resulting dataset, "data\_reduced," is now ready for further analysis and modeling. The data cleaning and analysis of the dataset from "statsfinal.csv" have provided valuable insights into unit sales and revenue trends over the years. The dataset is now well-prepared for further in-depth analysis or machine learning tasks.