**Project Report:Sales Data Visualization**

**Project Name:** Sales Data Visualization

**Student Name:** Shreya Yadav

**Roll Number:** 202401100400182

**Date:** 10/03/2025

**Introduction**

This project focuses on Simple Sales Data Visualization, aiming to analyze and visualize key aspects of sales data, including revenue trends, product demand, and seasonal sales patterns. Using Python libraries like pandas for data manipulation and matplotlib for visualization, the project provides a beginner-friendly approach to understanding sales data. It demonstrates how to analyze revenue over time, compare product demand, and identify seasonal trends, offering a foundational skill set for data-driven decision-making in business and analytics.

**Methodology**

The project follows a structured approach to analyze and visualize sales data. The methodology is divided into the following steps:

**1. Data Preparation**

* A sample dataset is created to simulate sales data for three products (Product\_A, Product\_B, Product\_C) and revenue over 12 months.
* The dataset includes:
  + **Date**: Monthly timestamps for a year.
  + **Product Sales**: Sales data for each product.
  + **Revenue**: Total revenue generated each month.

**2. Data Analysis**

* The dataset is loaded into a pandas DataFrame for easy manipulation.
* The Date column is set as the index to facilitate time-series analysis.

**3. Data Visualization**

* Three key visualizations are created using matplotlib:
  1. **Revenue Over Time**: A line plot showing monthly revenue trends.
  2. **Product Demand**: A line plot comparing the sales of the three products over time.
  3. **Seasonal Sales Trends**: A line plot highlighting seasonal patterns in product sales.

**4. Insights and Interpretation**

* The visualizations are analyzed to:
  + Identify trends in revenue growth or decline.
  + Compare the performance of different products.
  + Detect seasonal patterns in sales data.

**Code Used**

# Import necessary libraries

import pandas as pd  # For data manipulation

import matplotlib.pyplot as plt  # For data visualization

# Sample Data Creation

# We create a dictionary to simulate sales data for three products and revenue over 12 months

data = {

    'Date': pd.date\_range(start='2023-01-01', periods=12, freq='ME'),  # Monthly dates for a year

    'Product\_A\_Sales': [100, 120, 130, 150, 200, 250, 300, 280, 230, 200, 180, 160],  # Sales data for Product A

    'Product\_B\_Sales': [80, 90, 100, 110, 120, 130, 140, 130, 120, 110, 100, 90],  # Sales data for Product B

    'Product\_C\_Sales': [50, 60, 70, 80, 90, 100, 110, 100, 90, 80, 70, 60],  # Sales data for Product C

    'Revenue': [1000, 1200, 1300, 1500, 2000, 2500, 3000, 2800, 2300, 2000, 1800, 1600]  # Revenue data

}

# Convert the dictionary to a pandas DataFrame

df = pd.DataFrame(data)

# Set the 'Date' column as the index for easier time-series plotting

df.set\_index('Date', inplace=True)

# 1. Revenue Analysis

# Plot the revenue over time using a line plot

plt.figure(figsize=(10, 5))  # Set the figure size

plt.plot(df.index, df['Revenue'], marker='o', color='b', label='Revenue')  # Plot revenue data

plt.title('Revenue Over Time')  # Add a title to the plot

plt.xlabel('Date')  # Label the x-axis

plt.ylabel('Revenue ($)')  # Label the y-axis

plt.legend()  # Add a legend

plt.grid(True)  # Add a grid for better readability

plt.show()  # Display the plot

# 2. Product Demand

# Plot the demand for each product over time

plt.figure(figsize=(10, 5))  # Set the figure size

plt.plot(df.index, df['Product\_A\_Sales'], marker='o', label='Product A Sales')  # Plot Product A sales

plt.plot(df.index, df['Product\_B\_Sales'], marker='o', label='Product B Sales')  # Plot Product B sales

plt.plot(df.index, df['Product\_C\_Sales'], marker='o', label='Product C Sales')  # Plot Product C sales

plt.title('Product Demand Over Time')  # Add a title to the plot

plt.xlabel('Date')  # Label the x-axis

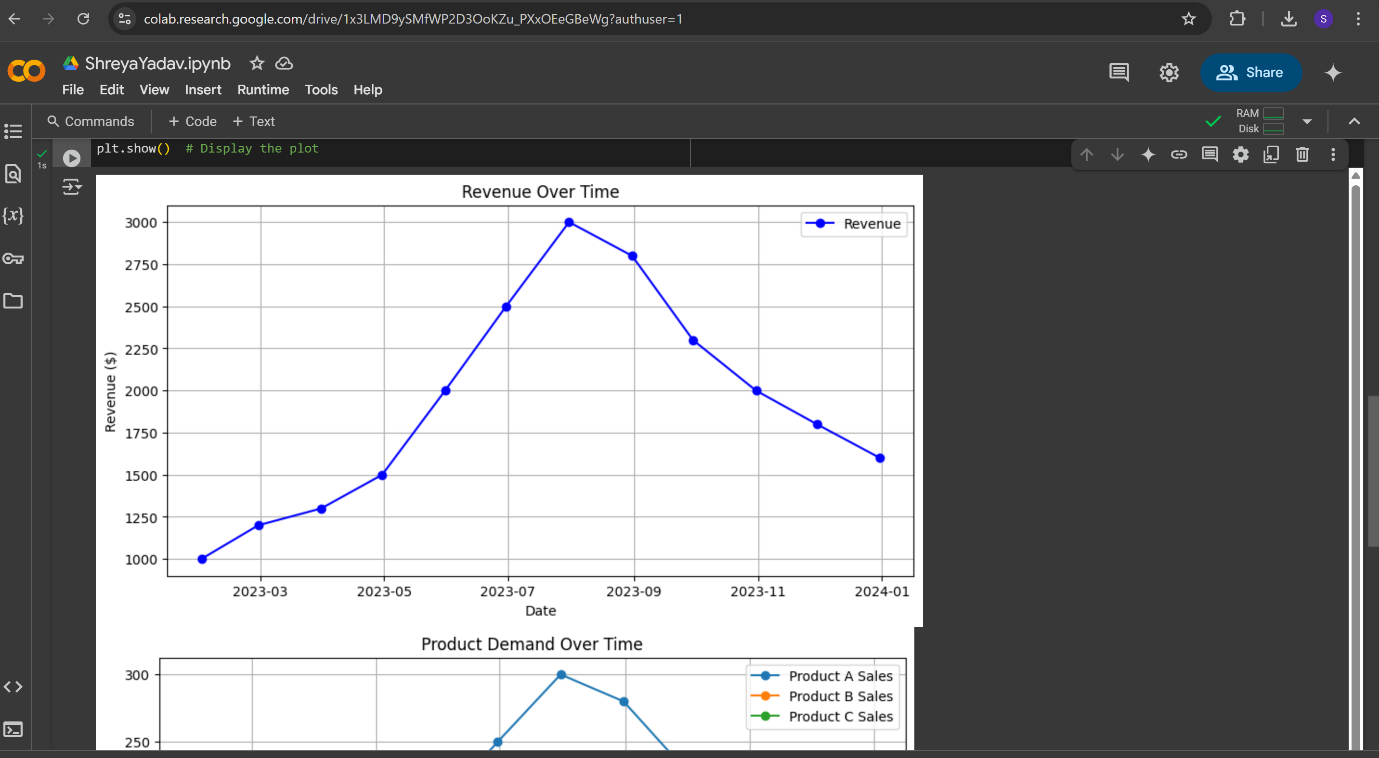
plt.ylabel('Units Sold')  # Label the y-axis

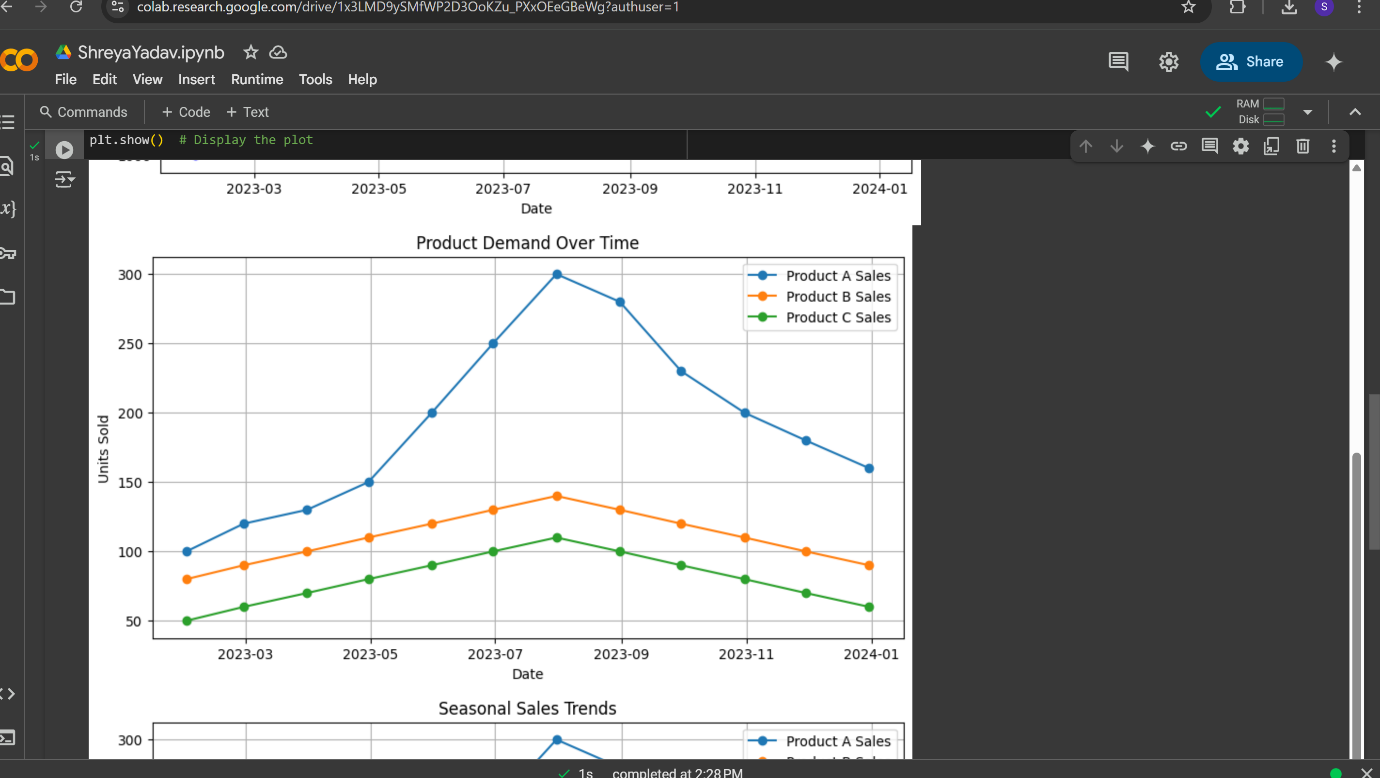
plt.legend()  # Add a legend

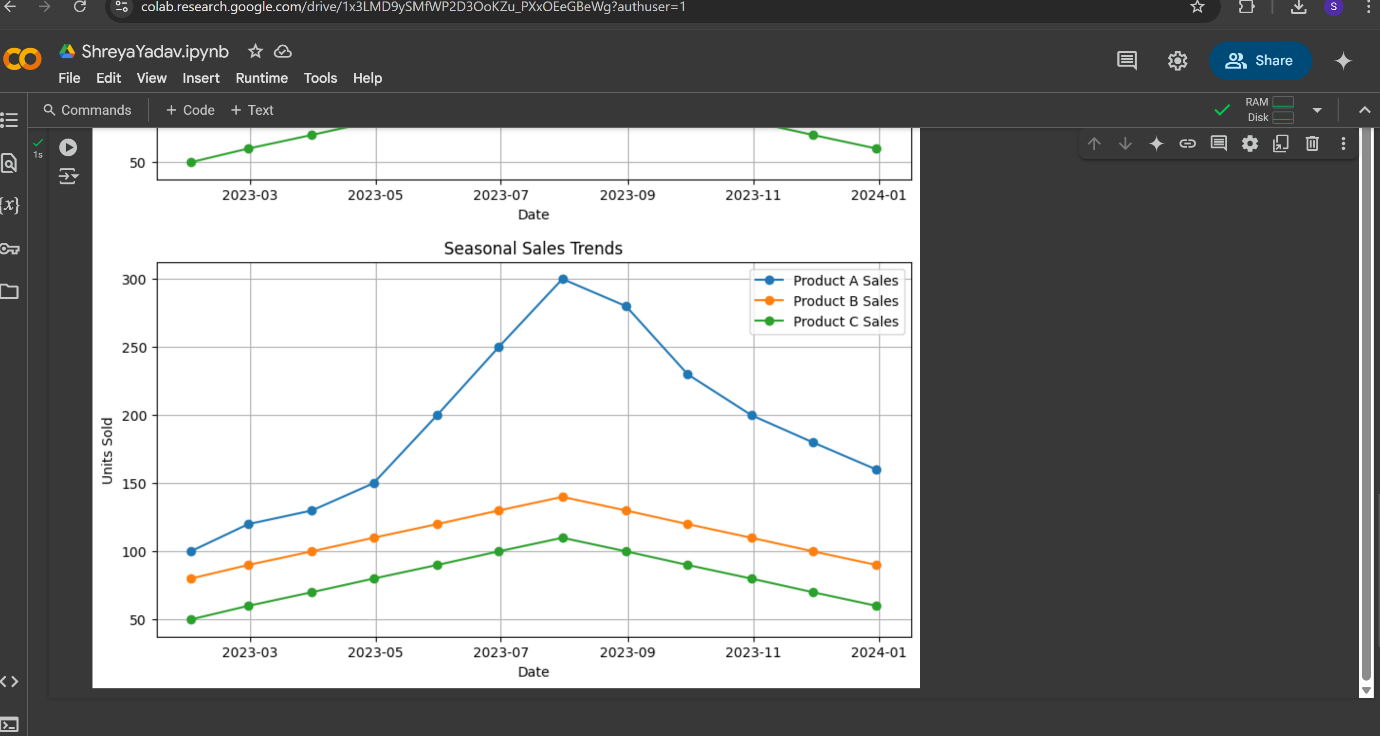
plt.grid(True)  # Add a grid for better readability

plt.show()  # Display the plot

**Screenshots**

****

****

****