Simple Sales Data Visualization – Analyze and plot revenue, product demand, and seasonal sales trends.

NAME :- SHIVAM KUMAR SINGH

BRANCH :- CSE(aiml)

ROLL NO. :- 202401100400176.

Sales Data Analysis and Visualization Report

1. Introduction

This report provides an analysis of the sales data for a hypothetical business using Python. The data contains daily sales information for three products across a year. The analysis focuses on two key aspects of sales performance:

1. **Monthly Revenue Trend** - How revenue fluctuates over the 12 months of the year.
2. **Product Demand** - The total sales for each of the three products.

The analysis is conducted using basic Python data structures (lists and defaultdict from the collections module) and visualization through the matplotlib library.

2. Methodology

2.1 Data Preparation

The dataset used for the analysis is generated with the following fields:

1. **Dates**: A list of 365 dates for the year 2023 (January 1 to December 31).
2. **Products**: A list of three products ("Product A", "Product B", "Product C"). Each product is repeated 120 times across the year to align with the daily sales data.
3. **Sales**: The sales numbers for each product for each day, with a repeating cycle of values.
4. **Prices**: The price associated with each product. Each product has a distinct price, repeating in a cycle.

2.2 Data Analysis

- **Revenue Calculation**: The revenue for each day is computed as the product of sales[i] and prices[i], which gives the total amount earned from sales on that day.
- **Monthly Revenue Aggregation**: The data is aggregated by month, summing up the revenue for each month (from January to December). This is done using the defaultdict data structure, where the key is the month number, and the value is the total revenue for that month.
- **Product Demand Aggregation**: The total sales for each product are calculated by summing the sales numbers for each product throughout the year.

2.3 Data Visualization

The results of the analysis are visualized using matplotlib in two ways:

1. **Monthly Revenue Trend**: A **line plot** is created to show the revenue trends over the months. This helps to understand how revenue fluctuates throughout the year.
2. **Product Demand**: A **bar plot** is used to display the total sales for each product. This shows the relative demand for each product over the year.

3. Results

3.1 Monthly Revenue Trend

The **monthly revenue trend** is represented in a line plot. This plot shows the total revenue for each of the 12 months in the year. The x-axis represents the months, and the y-axis represents the total revenue for each month.

- **Key Observations**:
  - The revenue fluctuates across months, with some months showing higher sales (possibly due to seasonality or promotions).
  - The plot allows for easy comparison of which months performed better and which months had lower sales.

Visualization:

(This is a placeholder link for illustrative purposes)

3.2 Product Demand

The **product demand** is shown in a bar plot. Each bar represents a product ("Product A", "Product B", and "Product C"), and the height of each bar represents the total sales for that product throughout the year.

- **Key Observations**:
  - The bar plot clearly shows the relative demand for each product. For example, if "Product A" has the tallest bar, it means that it had the highest total sales over the year.
  - This plot helps in identifying which products were the most popular or in-demand.

4. Conclusion

The analysis and visualizations provide valuable insights into the sales performance over the year. By looking at the **monthly revenue trend**, businesses can identify seasonal patterns and plan for months with expected lower or higher sales. The **product demand bar plot** helps identify the best-selling products, which could be useful for inventory and marketing decisions.

The code used for this analysis was simple yet effective, utilizing basic Python structures and libraries to perform aggregation and visualization. The results can be expanded to handle more complex datasets or real-world data with additional features, such as regions or customer segmentation.

6. Code Snippet

Below is the Python code used for the analysis:

```
import matplotlib.pyplot as plt

from collections import defaultdict

import datetime


# Sample Data: Dates, Products, Sales, Prices

dates = [datetime.date(2023, 1, i+1) for i in range(12)] * 30  # 12 months * 30 days (for simplicity)

products = ['Product A', 'Product B', 'Product C'] * 120

sales = [200, 150, 300, 250, 100, 200, 300, 350, 180, 220, 210, 240] * 30

prices = [20, 25, 30] * 120


# Initialize dictionaries for analysis

monthly_revenue = defaultdict(float)  # Month -> Revenue

product_sales = defaultdict(float)  # Product -> Total Sales


# Analyze data

for i in range(len(dates)):

    month = dates[i].month

    revenue = sales[i] * prices[i]
    # Aggregate monthly revenue

    monthly_revenue[month] += revenue


    # Aggregate product sales

    product_sales[products[i]] += sales[i]


# Sort months and products for consistent plotting

months = sorted(monthly_revenue.keys())

products_sorted = sorted(product_sales.keys())


# Plot Monthly Revenue Trend

plt.figure(figsize=(8, 5))

plt.plot(months, [monthly_revenue[month] for month in months], marker='o')

plt.title('Monthly Revenue Trend')

plt.xlabel('Month')

plt.ylabel('Revenue ($)')

plt.grid(True)

plt.show()


# Plot Product Demand (Total Sales)

plt.figure(figsize=(8, 5))

plt.bar(products_sorted, [product_sales[product] for product in products_sorted])

plt.title('Product Demand (Total Sales)')
plt.xlabel('Product')

plt.ylabel('Total Sales')

plt.show()
```

End of Report

This report summarizes the analysis of the sales data, providing insights through visualizations and explaining how the Python code performs the necessary calculations.