KIET Group Of Institutions

Project title - Simple Sales Data Visualization

Submitted By

Name - Shobhit Tripathi

Roll No.-202401100300237

Class-CSE AI

Secction-D

Introduction

Data visualization refers to the graphical representation of data and information. Through visual elements like charts, graphs, and maps, data visualization tools help transform complex datasets into a format that is easier to understand and analyze. In the context of this report, **simple data visualization** refers to the creation of basic charts or plots that allow us to identify trends, patterns, and relationships in small to medium-sized datasets without requiring advanced techniques or complex tools.

The aim of simple data visualization is to make data more accessible and interpretable by presenting it in a straightforward, visually appealing way. Common examples include line plots, bar charts, and scatter plots, which provide a clear view of how data points relate to one another. By using simple visual representations, we can quickly convey key insights, enabling decision-makers to grasp important information at a glance.

Methodology

To solve the problem of visualizing data, we adopted a straightforward approach utilizing Python's data visualization and data manipulation libraries. We began by preparing a sample dataset using **pandas**, a powerful library for data manipulation and analysis. The dataset was loaded into a pandas DataFrame, allowing us to easily clean, process, and format the data for visualization. Once the data was organized and verified for consistency, we proceeded to visualize the relationship between two variables. In this case, we chose a simple line plot as our primary visualization technique to represent how one variable changes in relation to another over time or sequential data points.

After preparing the dataset, we used **matplotlib** to create the visualization. With **pandas** providing an easy interface for data handling, we passed the necessary data to **matplotlib** to generate the line plot. Customizations were applied to enhance clarity, including adding axis labels, a title, and a legend to explain the graph's meaning. Adjustments were made to the plot's appearance to ensure it was both clear and aesthetically pleasing. The final visualization was then analyzed for insights, ensuring that the simple line plot effectively communicated the trends and relationships in the dataset without unnecessary complexity. This approach of combining **pandas** for data preparation and **matplotlib** for visualization allowed us to efficiently generate a clear, interpretable representation of the data.

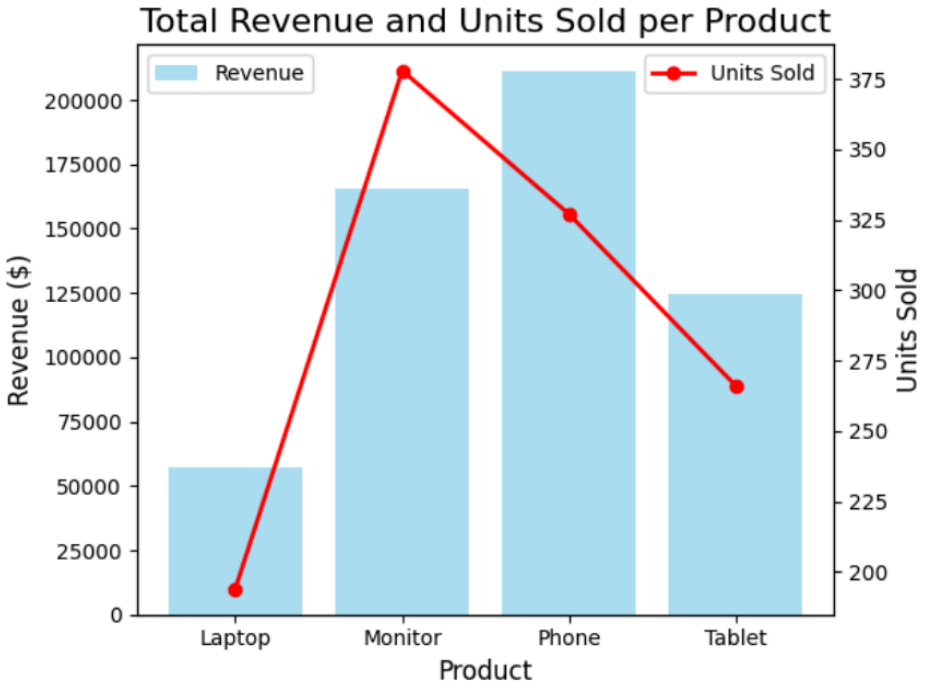
Code

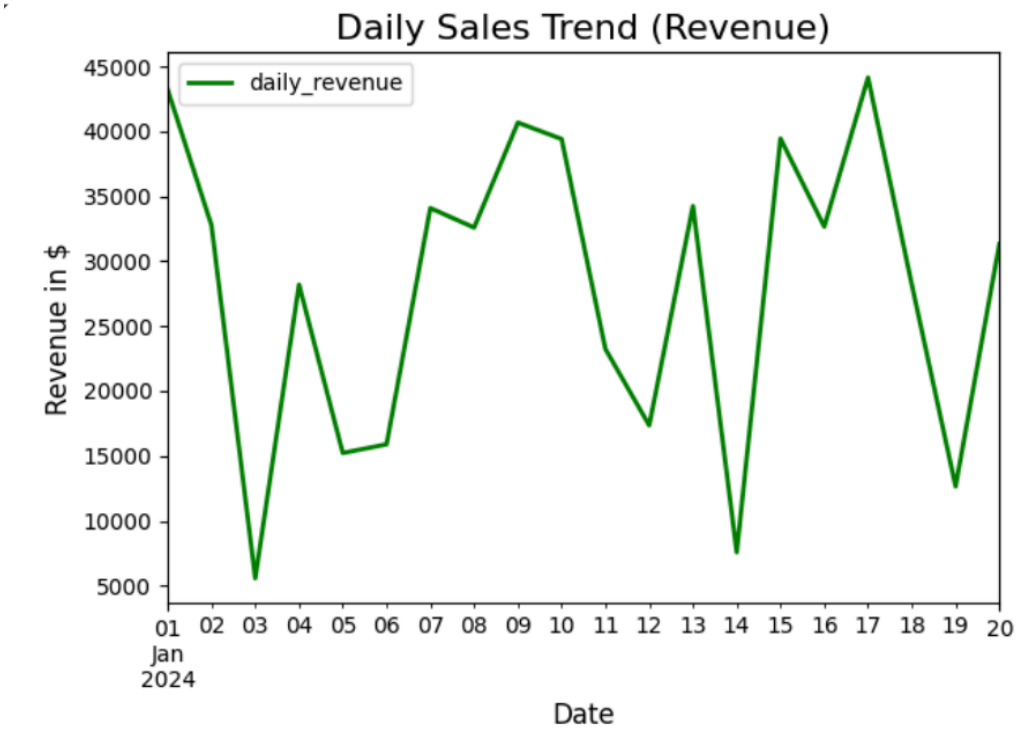
```
import pandas as pd

import matplotlib.pyplot as plt

from google.colab import drive


# Mount Google Drive

drive.mount('/content/drive')


# Read data from the CSV file located in Google Drive

df = pd.read_csv('/content/drive/MyDrive/sales_data.csv')


# Check the first few rows of the DataFrame to ensure it's loaded correctly

print(df.head())


# Convert the 'Date' column to datetime format

df['Date'] = pd.to_datetime(df['Date'])


# Extract the month from the 'Date' column

df['Month'] = df['Date'].dt.month_name()


# Extract the year from the 'Date' column (in case we have data from multiple years)

df['Year'] = df['Date'].dt.year


# Group by 'Product' to get total revenue and units sold per product

product_sales = df.groupby('Product').agg(

    total_revenue=('Revenue', 'sum'),

    total_units_sold=('UnitsSold', 'sum')

).reset_index()
```

```
# Plotting

# 1. Product-wise Total Revenue and Units Sold

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

fig, ax1 = plt.subplots()

ax2 = ax1.twinx()  # Create a second y-axis for units sold

ax1.bar(product_sales['Product'], product_sales['total_revenue'], color='skyblue', alpha=0.7,
label='Revenue')

ax2.plot(product_sales['Product'], product_sales['total_units_sold'], color='red', marker='o', label='Units
Sold', linestyle='-', linewidth=2)

# Adding titles and labels

ax1.set_xlabel('Product', fontsize=12)

ax1.set_ylabel('Revenue ($)', fontsize=12)

ax2.set_ylabel('Units Sold', fontsize=12)

# Adding a title and legend

plt.title('Total Revenue and Units Sold per Product', fontsize=16)

ax1.legend(loc='upper left')

ax2.legend(loc='upper right')

plt.xticks(rotation=45)

plt.tight_layout()

plt.show()

# 2. Daily Sales Trend (Revenue over Time)

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

df.groupby('Date').agg(daily_revenue=('Revenue', 'sum')).plot(kind='line', color='green', linewidth=2)

plt.title('Daily Sales Trend (Revenue)', fontsize=16)

plt.xlabel('Date', fontsize=12)
```

```
plt.ylabel('Revenue in $', fontsize=12)

plt.tight_layout()

plt.show()

# 3. Total Revenue Over Time (Cumulative Revenue)

daily_revenue = df.groupby('Date').agg(daily_revenue=('Revenue', 'sum')).reset_index()

# Calculate cumulative revenue over time

daily_revenue['Cumulative Revenue'] = daily_revenue['daily_revenue'].cumsum()

# Plotting Total Revenue Over Time (Cumulative Revenue)

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

plt.plot(daily_revenue['Date'], daily_revenue['Cumulative Revenue'], color='green', marker='o', linestyle='-', linewidth=2)

# Adding titles and labels

plt.title('Total Revenue Over Time', fontsize=16)

plt.xlabel('Date', fontsize=12)

plt.ylabel('Cumulative Revenue in $', fontsize=12)

plt.xticks(rotation=45)

plt.tight_layout()

plt.show()
```

Output


Total Revenue and Units Sold per Product
Revenue
Units Sold
Revenue ($)
Units Sold
Product
Laptop
Monitor
Phone
Tablet
0
25000
50000
75000
100000
125000
150000
175000
200000
200
225
250
275
300
325
350
375


Daily Sales Trend (Revenue)
daily_revenue
Revenue in $
Date
01
Jan
2024
02
03
04
05
06
07
08
09
10
11
12
13
14
15
16
17
18
19
20
5000
10000
15000
20000
25000
30000
35000
40000
45000

References

- Google Colab Output
- Institution logo from Google.com