**Project Title:** Retail Discount Analysis

**Group Member**: Rence Vithlani ku2407u705

Aksh Shah ku2407u011

Neel Patel ku2407u685

Dhyey Thakkar ku2407u687

Ayushi Jha ku2407u826

**Objective of The Project**:

The objective of a \*Retail Discount Analysis Project\* is to analyze and evaluate the impact of discounting strategies on sales performance, profitability, and customer behavior within a retail environment. This project typically aims to:

1. \*Measure the effectiveness of discounts\*: Assess how different discount strategies (e.g., percentage off, buy one get one free, seasonal promotions) influence customer purchases, sales volume, and revenue.

2. \*Identify optimal discount levels\*: Analyze various discount levels to determine which offers maximize sales while maintaining healthy profit margins.

3. \*Understand customer response\*: Study how different customer segments (e.g., age, location, purchasing history) respond to discounts to tailor future promotional efforts.

4. \*Evaluate profitability\*: Determine whether the revenue generated from increased sales offsets the reduced price from the discount, ensuring that the company maintains profitability.

5. \*Forecast future promotions\*: Use historical data to predict the impact of future discounts on sales, helping retail managers plan more effective promotional campaigns.

6. \*Improve inventory management\*: Assess how discounts affect inventory turnover and adjust stock levels based on customer demand driven by promotions.

Overall, the project aims to provide data-driven insights that can guide retail pricing and discount strategies, improving sales outcomes and enhancing customer loyalty.

**Tools & Library used:**

The code uses the following tools and libraries:

1. Libraries for Data Manipulation and Analysis:
   * pandas:
     + Used to load and manipulate the dataset from an Excel file (pd.read\_excel).
     + Provides functionality for working with data in a tabular format.
2. Libraries for Visualization:
   * matplotlib:
     + Specifically uses matplotlib.pyplot for creating visualizations.
     + The Agg backend (matplotlib.use('Agg')) is used to save figures without displaying them, which is often used in non-interactive environments.
3. Tools for Handling Excel Files:
   * Excel File: The dataset is stored in an Excel file (updated\_retail\_data.xlsx), and the code uses the pandas library to load the data from a specific sheet.

**Data Sources:**

The current dataset used in the script is synthetic data created for demonstration purposes.

**Execution Step:**

1. Import Required Libraries

* The code begins by importing the necessary Python libraries:
  + pandas: For loading and processing data.
  + matplotlib: For creating the graph.
  + matplotlib.use('Agg'): Specifies the "Agg" backend for matplotlib, which is used for generating images without displaying them interactively (e.g., for saving to a file).

**2. Load the Excel File**

* The file\_path variable stores the path to the Excel file, and the sheet\_name specifies the sheet to load.
* The pd.read\_excel() function loads the data into a pandas DataFrame.

**3. Display Dataset Overview**

* The code prints the first few rows of the dataset using data.head(), allowing you to quickly inspect the data.
* It also prints the column names using data.columns to show what columns are present in the dataset.

**4. Create the Bar Chart**

* The code creates a bar chart with matplotlib.
* It specifies the size of the figure using plt.figure(figsize=(15, 7)).
* It plots the "Product Name" on the x-axis and the "Total Sales After Discount" on the y-axis using plt.bar().
* Additional properties are set, such as:
  + Labels for the x and y axes (plt.xlabel() and plt.ylabel()).
  + A title for the chart (plt.title()).
  + Rotation for the x-axis labels (plt.xticks(rotation=90)), which helps when there are many product names.
  + plt.tight\_layout() is used to ensure that the labels and title are correctly positioned in the plot.

**5. Save and Display the Graph**

* The bar chart is saved to a file (total\_sales\_after\_discount\_graph.png) using plt.savefig().
* The plt.show() function is used to display the graph if you are running the script in an interactive environment.
* Finally, a message is printed confirming the graph file location

**6. Execution Workflow:**

1. **Import Libraries**: Import pandas for data handling and matplotlib for plotting.
2. **Load Data**: Load data from the specified Excel file and sheet into a DataFrame.
3. **Inspect Data**: Print the first few rows and columns of the dataset for review.
4. **Generate Bar Chart**: Create a bar chart that visualizes the Total Sales After Discount for each Product Name.
5. **Save and Show Graph**: Save the bar chart as an image and optionally display it in an interactive environment.

**Summary of Results:**

1. Data Loading and Inspection:

* The dataset is successfully loaded from the Excel file located at C:/Users/undha/Downloads/updated\_retail\_data.xlsx (or another file path you specify).
* The first few rows of the dataset are displayed, providing an overview of the data structure.
* The columns in the dataset are printed, which helps confirm the available data and ensure that relevant columns like "Product Name" and "Total Sales After Discount" are present.

2. Dataset Overview:

* A preview of the first few rows of the dataset is printed to the console using data.head(). This gives you a snapshot of the data, including product names and corresponding sales figures.
* The column names are also displayed using data.columns, showing you what data attributes are available in the dataset (e.g., "Product Name", "Total Sales After Discount").

3. Bar Chart Creation:

* A bar chart is created that visualizes the total sales after discount for each product.
* The chart has:
  + X-axis: Product names (labelled on the x-axis).
  + Y-axis: Total Sales After Discount for each product.
  + The bars are colored with a sky blue shade.
* Labels are added for both the x and y axes, and the chart is titled "Total Sales After Discount for Each Product."
* The x-axis labels (product names) are rotated 90 degrees for better readability.

4. Graph Saving:

* The bar chart is saved as an image file (total\_sales\_after\_discount\_graph.png) in the current working directory (or the directory where the script is executed).
* The graph is displayed if you run the script in an interactive environment, like Jupyter notebooks or a Python GUI, though this is not necessary in a headless environment where the backend "Agg" is used.

5. Confirmation:

* After saving the graph, a message is printed to the console confirming the location and name of the saved file.

**Challenges Faced:**

**1.** **File Path Issues**: Ensure the file path is correct and the Excel file exists at the specified location.

**2.** **Incorrect Column Names**: Verify that the column names (e.g., Product Name, Total Sales After Discount) match exactly and handle any extra spaces.

**3.** **Data Types Mismatch**: Make sure that numeric columns are properly formatted. Convert them to numeric if necessary using pd.to\_numeric().

**4.** **Missing Values**: Handle missing data (NaN) by dropping rows or filling in missing values with default values like 0.

**5.** **Excel Formatting Issues**: Ensure the Excel file is well-structured, with clean data and no merged cells.

**6.** **Plotting Issues**: If product names are too long or numerous, rotate x-axis labels and adjust the figure size or font size for readability.

**7.** **Graph Not Displaying**: In headless environments, ensure the graph is saved (with Agg backend) and adjust settings for display if necessary.

**8.** **Permissions and Save Issues**: Ensure the save path is valid and that you have the necessary permissions to write to the directory.