**Sales Analysis Dashboard Report**

Subject: Python Programming(24CAH-606)

Submitted by: Ashi Mittal (24MCI10015)

Section\group:24MAM1(A)

Branch:MCA(AIML)

Github:- https://github.com/ashimittal5/mini-python.git

**Introduction**

The Sales Analysis Dashboard is a graphical user interface (GUI) application developed using Python's Tkinter library. This application allows users to load sales data from a CSV file, visualize sales trends by month and product, and calculate total sales. The dashboard utilizes the Pandas library for data manipulation and Matplotlib for data visualization.

**Objectives**

1. **Load sales data from CSV files**.
2. **Visualize sales by month and by product**.
3. **Calculate and display total sales**.
4. **Provide an interactive and user-friendly interface**.

**Technical Overview**

**Tools and Libraries Used**

* **Python**: The programming language used to develop the application.
* **Tkinter**: A standard GUI toolkit in Python for building desktop applications.
* **Pandas**: A data analysis library that provides data structures for efficiently manipulating large datasets.
* **Matplotlib**: A plotting library that provides an object-oriented API for embedding plots into applications.

**Installation Requirements**

To run the application, the following packages should be installed:

* Tkinter (comes with standard Python installations)
* Pandas
* Matplotlib

These packages can be installed using pip:

pip install pandas matplotlib

**Application Structure**

The application consists of the following components:

1. **Main Window**: The primary interface where buttons for various functionalities are located.
2. **Buttons**: Interactive elements for loading data, plotting sales by month or product, and displaying total sales.
3. **Plots**: Visual representations of sales data generated dynamically based on user actions.

**Code Explanation**

Below is the complete code of the application, followed by detailed explanations of each section

import tkinter as tk

from tkinter import filedialog, messagebox

import pandas as pd

import matplotlib.pyplot as plt

from matplotlib.backends.backend\_tkagg import FigureCanvasTkAgg

# Initialize the main application window

root = tk.Tk()

root.title("Sales Analysis Dashboard")

root.geometry("800x600")

# Define global variable for data

data = None

# Function to load CSV file

def load\_data():

global data

file\_path = filedialog.askopenfilename(filetypes=[("CSV Files", "\*.csv")])

if file\_path:

data = pd.read\_csv(file\_path)

# Ensure 'Sales' column is numeric

data['Sales'] = pd.to\_numeric(data['Sales'], errors='coerce')

messagebox.showinfo("Data Load", "Data loaded successfully!")

else:

messagebox.showerror("Data Load Error", "No file selected or file could not be loaded.")

# Function to clear previous plots

def clear\_plots():

for widget in root.pack\_slaves():

if isinstance(widget, FigureCanvasTkAgg):

widget.get\_tk\_widget().destroy()

# Function to plot sales by month

def plot\_sales\_by\_month():

if data is None:

messagebox.showerror("Data Error", "No data loaded.")

return

data['Date'] = pd.to\_datetime(data['Date'])

data['Month'] = data['Date'].dt.month

monthly\_sales = data.groupby('Month')['Sales'].sum()

clear\_plots()

fig, ax = plt.subplots()

monthly\_sales.plot(kind='bar', ax=ax, color='skyblue')

ax.set\_title("Sales by Month")

ax.set\_xlabel("Month")

ax.set\_ylabel("Sales")

canvas = FigureCanvasTkAgg(fig, master=root)

canvas.draw()

canvas.get\_tk\_widget().pack()

# Function to plot sales by product

def plot\_sales\_by\_product():

if data is None:

messagebox.showerror("Data Error", "No data loaded.")

return

product\_sales = data.groupby('Product')['Sales'].sum()

clear\_plots()

fig, ax = plt.subplots()

product\_sales.plot(kind='bar', ax=ax, color='lightgreen')

ax.set\_title("Sales by Product")

ax.set\_xlabel("Product")

ax.set\_ylabel("Sales")

canvas = FigureCanvasTkAgg(fig, master=root)

canvas.draw()

canvas.get\_tk\_widget().pack()

# Function to display total sales

def display\_total\_sales():

if data is None:

messagebox.showerror("Data Error", "No data loaded.")

return

total\_sales = data['Sales'].sum()

messagebox.showinfo("Total Sales", f"Total Sales: ${total\_sales:.2f}")

# Buttons for the dashboard

btn\_load = tk.Button(root, text="Load Data", command=load\_data)

btn\_load.pack(pady=10)

btn\_sales\_by\_month = tk.Button(root, text="Plot Sales by Month", command=plot\_sales\_by\_month)

btn\_sales\_by\_month.pack(pady=10)

btn\_sales\_by\_product = tk.Button(root, text="Plot Sales by Product", command=plot\_sales\_by\_product)

btn\_sales\_by\_product.pack(pady=10)

btn\_total\_sales = tk.Button(root, text="Display Total Sales", command=display\_total\_sales)

btn\_total\_sales.pack(pady=10)

# Run the application

root.mainloop()

**Code Breakdown**

1. **Imports**:
   * The necessary libraries are imported at the start. Tkinter is used for GUI elements, Pandas for data manipulation, and Matplotlib for plotting.
2. **Main Window Initialization**:
   * A Tkinter window is created, titled "Sales Analysis Dashboard", with a specified size.
3. **Global Variable**:
   * A global variable data is defined to hold the loaded CSV data.
4. **Loading Data**:
   * The load\_data function allows users to select a CSV file. The data is read using Pandas, and the 'Sales' column is converted to numeric. A success or error message is displayed based on the outcome.
5. **Clearing Previous Plots**:
   * The clear\_plots function removes any previously displayed plots to ensure the new plot does not overlap with the old one.
6. **Plotting Functions**:
   * The plot\_sales\_by\_month and plot\_sales\_by\_product functions generate bar charts based on monthly sales and product sales, respectively. They clear previous plots before drawing new ones.
7. **Displaying Total Sales**:
   * The display\_total\_sales function calculates the total sales from the data and shows it in a message box.
8. **Buttons**:
   * Buttons are created for each functionality, allowing users to interact with the application.
9. **Main Loop**:
   * The application enters the Tkinter main loop, waiting for user input.

**User Instructions**

**How to Use the Application**

1. **Load Data**: Click the "Load Data" button to select a CSV file containing sales data.
2. **Plot Sales by Month**: Click the "Plot Sales by Month" button to visualize sales distribution across different months.
3. **Plot Sales by Product**: Click the "Plot Sales by Product" button to visualize sales distribution for each product.
4. **Display Total Sales**: Click the "Display Total Sales" button to view the total sales amount.

**Data Format**

The CSV file should contain the following columns:

* **Date**: The date of the sales transaction.
* **Sales**: The amount of sales made.
* **Product**: The name of the product sold.

**Example CSV Formal**

Date,Sales,Product

2023-01-01,100,Product A

2023-01-02,150,Product B

2023-02-01,200,Product A

**Conclusion**

The Sales Analysis Dashboard provides a simple yet effective way to analyze sales data. By leveraging the capabilities of Tkinter, Pandas, and Matplotlib, the application enables users to make data-driven decisions based on visual insights. This report outlines the application's structure, functionality, and usage, making it easy for users to understand and operate the tool effectively.

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

A graph with green bars

Description automatically generated