Kanti Sweets Sales Data Analysis and Visualization using Python

Index

1. Description (Detailed)
2. Data within the Problem & Relevant Info
3. Output & Purpose
4. Outcome & Benefits
5. Plan
6. Design (Diagrams)
7. Implementation
8. Code & Explanation
9. Output Screenshots
10. Closure (Brief of the Project)
11. Bibliography

PROBLEM STATEMENT

Kanti Sweets, a growing retail chain, faces challenges in managing inventory, minimizing wastage, and understanding sales trends to optimize profits. Despite data collection, it's underutilized for decision-making. This project provides a professional, data-driven approach to analyze historical sales and stock data, empowering business decisions and improving resource management using Python and visualization tools.

DESCRIPTION

- This project is centered around the analysis of operational data collected across multiple branches of Kanti Sweets. The analysis includes identifying best-selling products, underperforming items, revenue patterns, and inventory mismatches across locations. It helps store managers and head office teams to:

- Determine product demand and customer preferences.
- Compare revenue performance month-wise and year-wise.
- Recognize surplus and shortage patterns in inventory.
- Predict high-sales periods such as weekends.
- The outcome is a holistic system that can assist in making well-informed decisions, improve profitability, and minimize spoilage of perishable items .

DATA ANALYSIS AND OUTPUT INSIGHTS :

Input Data:

File: kanti_sweets_multi_branch.csv (or raw_data.csv as per previous context, then cleaned to clean_data.csv)

Attributes: sweet_name, quantity_sold, revenue (or purchase_cost), month, year, branch (or branch_location), days (or date for weekday extraction).

Outputs Generated:

Graphs:

- Bar graphs showing top-selling and least-selling items (monthly).
- Line and bar charts for revenue comparisons across months and branches.
- Bar graph for total sales vs. month.

Textual Analysis:

Insights accompanying each graph (e.g., "Highest revenue: April with ₹924585.34.").

Objective:

- Forecast and optimize stock levels.
- Avoid loss due to excess production or low demand.
- Understand seasonal and daily purchase behavior.

Expected Benefits:

- Better branch coordination and logistics.
- Reduced operational costs.
- Improved customer satisfaction.
- Enhanced operational efficiency and optimized stock planning.

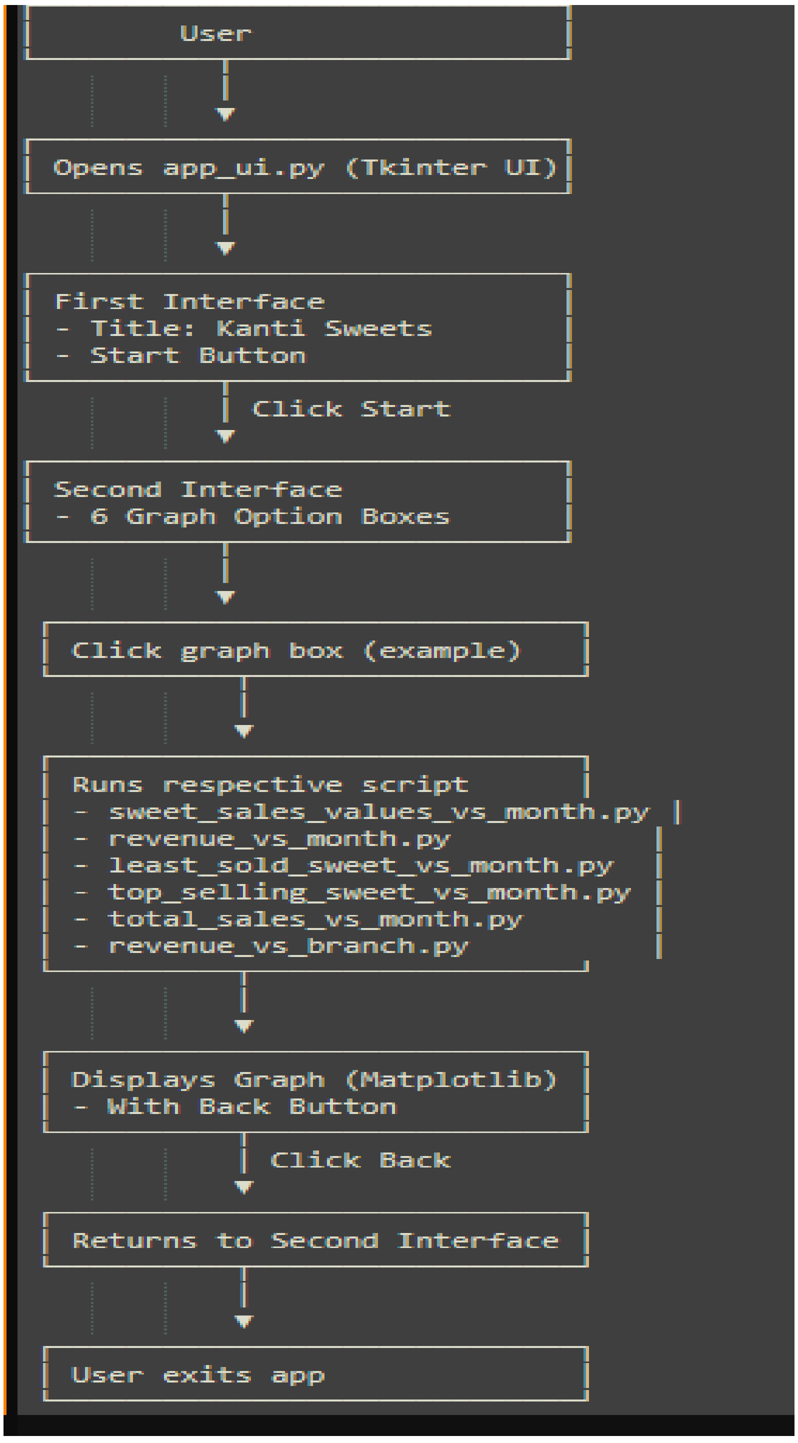
PROJECT PLAN: KANTI SWEETS SALES ANALYSIS

1. To provide a scalable and modular solution, the project plan included:
2. Data Collection & Cleaning: Importing CSV data using pandas and performing necessary cleaning steps (e.g., removing transaction_id, handling dates).
3. Data Analysis: Utilizing pandas' groupby, filters, and aggregate functions to extract meaningful insights from the cleaned data.
4. Visualization: Representing the analyzed data effectively via charts using Matplotlib.

5. Segmentation of Logic: Dividing responsibilities into different Python files (e.g., sales_analysis.py, revenue_analysis.py, inventory_analysis.py) for modularity and maintainability.
6. User Interface: Developing a Graphical User Interface (GUI) using Tkinter to enable interaction with non-technical users.

➢ DIAGRAM

User Flow Diagram


User
Opens app_ui.py (Tkinter UI)
First Interface
- Title: Kanti Sweets
- Start Button
Click Start
Second Interface
- 6 Graph Option Boxes
Click graph box (example)
Runs respective script
- sweet_sales_values_vs_month.py
- revenue_vs_month.py
- least_sold_sweet_vs_month.py
- top_selling_sweet_vs_month.py
- total_sales_vs_month.py
- revenue_vs_branch.py
Displays Graph (Matplotlib)
- With Back Button
Click Back
Returns to Second Interface
User exits app

IMPLEMENTATION

The project's implementation is modularized for clarity and maintainability, structured into three main stages:

Data Cleaning:

The initial stage involves preparing the raw dataset for analysis using clean_data.py. This script loads raw_data.csv, removes the transaction_id column (as it's not required for sales analysis), and saves the processed data to clean_data.csv. This ensures that only relevant data remains, optimizing subsequent analysis.

Analysis Scripts:

Following data cleaning, multiple individual Python scripts were created in the analyzed_data folder to generate specific visual insights. Each script:

Reads the clean_data.csv dataset.

Processes data to compute required metrics (e.g., grouping by month, summing sales/revenue).

Generates visualizations using Matplotlib and Seaborn.

These scripts include:
sweet_sales_values_vs_month.py (total sales value of each sweet), revenue_vs_month.py (overall monthly revenue), least_sold_sweet_vs_month.py (least sold sweets each month), top_selling_sweet_vs_month.py (highest-selling sweet each month), total_sales_vs_month.py (total sales count per month), and revenue_vs_branch.py (revenue per branch).

GUI (Graphical User Interface):

A Tkinter-based GUI application (app_ui.py) was developed to integrate all analysis scripts, providing an interactive platform.

First Interface (Home Page): Displays the application title and a "Start" button to navigate.

Second Interface (Graph Selection Page): Presents six distinct buttons, each representing one of the analysis graphs. Clicking a button generates and displays the corresponding graph.

Graph View: Displays the selected graph. A "Back" button is provided to return to the selection page, ensuring smooth navigation.

The system is designed with robust error handling for missing files and invalid inputs, and its modularity supports future enhancements

CODE & EXPLANATION

1.Cleaning of Data:

```
import pandas as pd

df = pd.read_csv('raw_data.csv')

df = df.drop('transaction_id', axis=1)

df.to_csv('clean_data.csv', index=False)
```

Step 1: Import the pandas library to handle data reading and cleaning tasks.

Step 2: Read the raw CSV data file (raw_data.csv) and load it into a pandas DataFrame named df.

Step 3: Drop the unneeded column transaction_id from the DataFrame since it is not required for sales analysis.

Step 4: Save the cleaned data into a new CSV file named clean_data.csv without including row indices.

2.top_selling_sweet_vs_month:

```
import pandas as pd

import matplotlib.pyplot as plt

def plot_top_selling_sweet_vs_month():

    """Identifies and plots the top-selling sweet for each month."""

    df = pd.read_csv('clean_data.csv')

    df['month'] = pd.to_datetime(df['date']).dt.strftime('%B') # Extract month
names
```

```
    # Group by month and sweet, then count sales
    monthly_sales = df.groupby(['month', 'item_name']).size()


    # Get top sweet name and its sales count per month
    top_sweets_name = monthly_sales.groupby(level=0).idxmax().apply(lambda x:
x[1])
    top_sweets_count = monthly_sales.groupby(level=0).max()
    plt.figure(figsize=(10, 6))
    plt.bar(top_sweets_name.index, top_sweets_count.values) # Plot bar chart
    plt.xlabel("Month")
    plt.ylabel("Top Sweet Sales Count")
    plt.title("Top Selling Sweet Each Month")
    plt.xticks(rotation=45, ha='right')
    plt.tight_layout()
    return plt.gcf() # Return figure for GUI
```

Explaination for top_selling_sweet_vs_month:

- Purpose: Identifies and visualizes the best-selling sweet item for each month.
- Key Code Snippets & What they do:
- df['month'] = ...strftime('%B'): Extracts full month names from the 'date' column.
- df.groupby(['month', 'item_name']).size(): Groups data by month and item name, then counts the occurrences (sales) of each sweet per month.
- monthly_sales.groupby(level=0).idxmax(): Finds the item_name with the maximum sales count for each month.

- plt.bar(...): Creates a bar chart showing the top-selling sweet's quantity for each month.
- Why it was used: Helps Kanti Sweets understand monthly product popularity, aiding in targeted marketing and procurement strategies.

3. revenue_vs_branch.py

```
import pandas as pd
import matplotlib.pyplot as plt
def plot_revenue_vs_branch():
    """Calculates and plots total revenue generated by each branch location."""
    df = pd.read_csv('clean_data.csv')


    # Calculate revenue per branch
    branch_revenue = df.groupby('branch_location')['purchase_cost'].sum()
    # Sort and get top 5 branches if desired, or plot all
    # For this example, plotting all as per main code, but friend's description mentions 'Top 5'
    # branch_revenue = branch_revenue.nlargest(5) # Uncomment for Top 5
    plt.figure(figsize=(10, 6))
    plt.bar(branch_revenue.index, branch_revenue.values) # Plot bar chart
    plt.xlabel("Branch Location")
    plt.ylabel("Revenue")
    plt.title("Revenue per Branch")
    plt.xticks(rotation=45, ha='right')
    plt.tight_layout()
    return plt.gcf() # Return figure for GUI
```

Explaination for revenue_vs_branch.py

- Import libraries ➔ For reading data and plotting.
- Read cleaned data ➔ Loads dataset.
- Group by branch and sum purchase cost ➔ Calculates total revenue for each branch.
- Plot bar chart of branch vs revenue ➔ Visualizes regional revenue performance.
- Add labels and title, show plot ➔ For clarity and final output.

---

4. revenue_vs_month.py

```
import pandas as pd
import matplotlib.pyplot as plt
def plot_revenue_vs_month():
    """Analyzes and plots total revenue generated per month."""
    df = pd.read_csv('clean_data.csv')

    df['month'] = pd.to_datetime(df['date']).dt.strftime('%B') # Extract
month names

    # Define a custom order for months

    month_order = ['January', 'February', 'March', 'April', 'May', 'June',

                   'July', 'August', 'September', 'October', 'November',
'December']

    df['month'] = pd.Categorical(df['month'], categories=month_order,
ordered=True)

    # Calculate revenue per month

    monthly_revenue =
df.groupby('month')['purchase_cost'].sum().reindex(month_order,
fill_value=0)

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

    plt.plot(monthly_revenue.index, monthly_revenue.values,
marker='o', color='green') # Plot line chart

    plt.xlabel("Month")

    plt.ylabel("Total Revenue")

    plt.title("Revenue vs. Month")

    plt.grid(True)

    plt.xticks(rotation=45, ha='right')

    plt.tight_layout()

    return plt.gcf() # Return figure for GUI
```

Explaination for revenue_vs_month.py

- Import pandas and matplotlib ➔ For processing and plotting.
- Read cleaned data ➔ Load dataset.

- Extract month names from dates ➔ Groups data by month.
- Group by month and sum purchase cost ➔ Calculates monthly revenue.
- Plot bar chart of month vs revenue ➔ Visualizes revenue trends.
- Add labels and title, display plot ➔ For clarity and output.

---

5. total_sales_vs_month.py

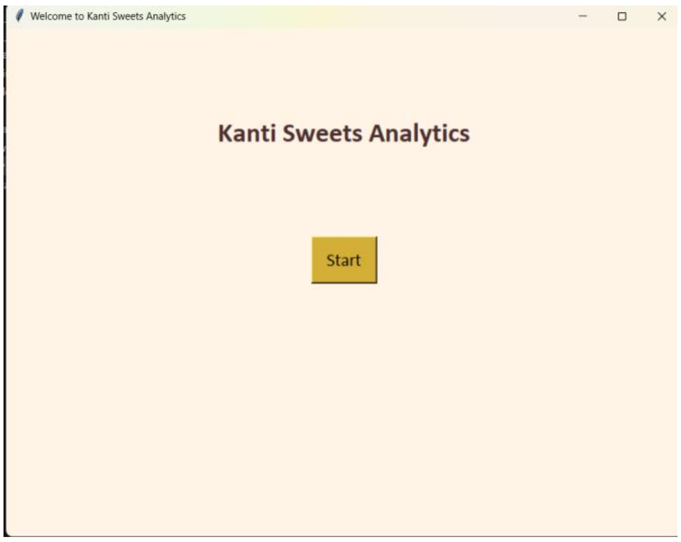
```
import pandas as pd

import matplotlib.pyplot as plt

def plot_total_sales_vs_month():

    """Counts and plots total sales transactions per month."""

    df = pd.read_csv('clean_data.csv')

    df['month'] = pd.to_datetime(df['date']).dt.strftime('%B') # Extract month
names

    # Define a custom order for months

    month_order = ['January', 'February', 'March', 'April', 'May', 'June',

                   'July', 'August', 'September', 'October', 'November', 'December']

    df['month'] = pd.Categorical(df['month'], categories=month_order,
ordered=True)

    # Count total sales per month

    monthly_sales_count = df.groupby('month').size().reindex(month_order,
fill_value=0)

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

    plt.plot(monthly_sales_count.index, monthly_sales_count.values, marker='o') #
Plot line chart

    plt.xlabel("Month")

    plt.ylabel("Total Sales Count")
    plt.title("Total Sales per Month")

    plt.grid(True)

    plt.xticks(rotation=45, ha='right')

    plt.tight_layout()

    return plt.gcf() # Return figure for GUI
```

Explaination for total_sales_vs_month.py

- Import pandas and matplotlib ➔ For data processing and plotting.
- Read cleaned data ➔ Load dataset.
- Extract month names ➔ For grouping data by month.
- Group by month and sweet name, sum purchase cost ➔ Calculates total sales value per sweet per month.
- Unstack data ➔ Converts it into table format for stacked plotting.
- Plot stacked bar chart ➔ Visualizes each sweet’s sales value contribution per month.
- Set labels, title, adjust legend, tight layout, display plot ➔ For clarity

OUTPUT SCREENSHOTS

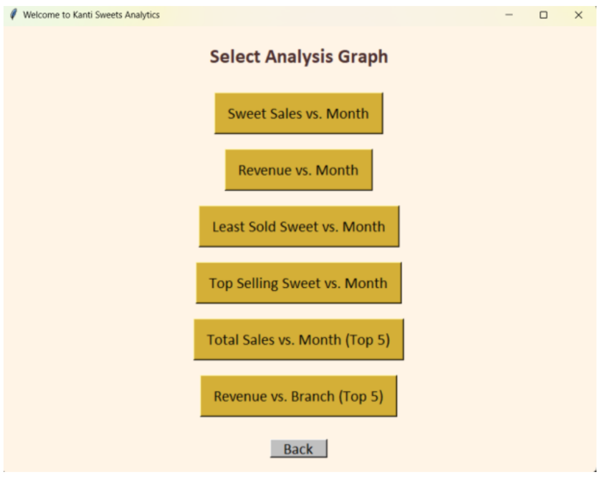
Welcome screen



The main screen of the Kanti Sweets Sales Analysis Dashboard serves as the entry point for users to access the sales

analysis features of the application. It is designed to be simple, intuitive, and welcoming.

Main Selection Screen

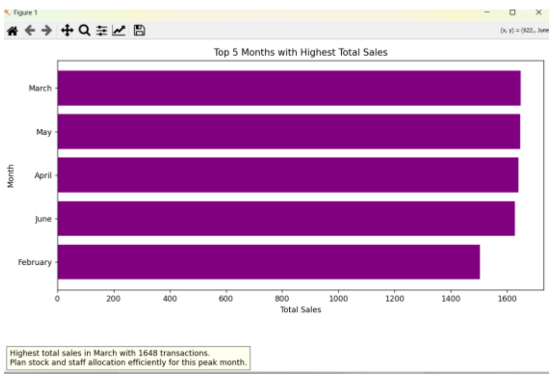

Welcome to Kanti Sweets Analytics
Select Analysis Graph
Sweet Sales vs. Month
Revenue vs. Month
Least Sold Sweet vs. Month
Top Selling Sweet vs. Month
Total Sales vs. Month (Top 5)
Revenue vs. Branch (Top 5)
Back

The main selection screen is the second interface of the Kanti Sweets Sales Analysis Dashboard. After the user clicks the “Start Analysis” button on the main screen, they are directed to this interface, where they can select the type of analysis they wish to view.

Graph Selection Options

- The interface contains six distinct buttons or boxes, each representing one of the analysis scripts available:
- Options include:
    - Total Sales vs Month
    - Revenue vs Month
    - Top Selling Sweet vs Month
    - Least Sold Sweet vs Month
    - Sweet Sales Values vs Month
    - Revenue vs Branch

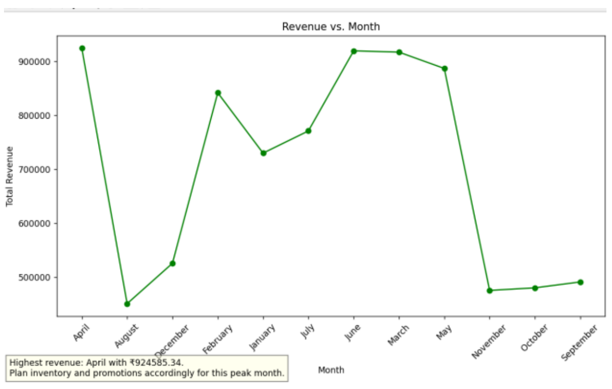
Horizontal bar chart - Total Sales vs Month



Highest total sales in March with 1648 transactions.
Plan stock and staff allocation efficiently for this peak month.

Description: Shows the total number of sales transactions per month with trends connected via lines for clear temporal analysis.

Line chart- Revenue vs Month

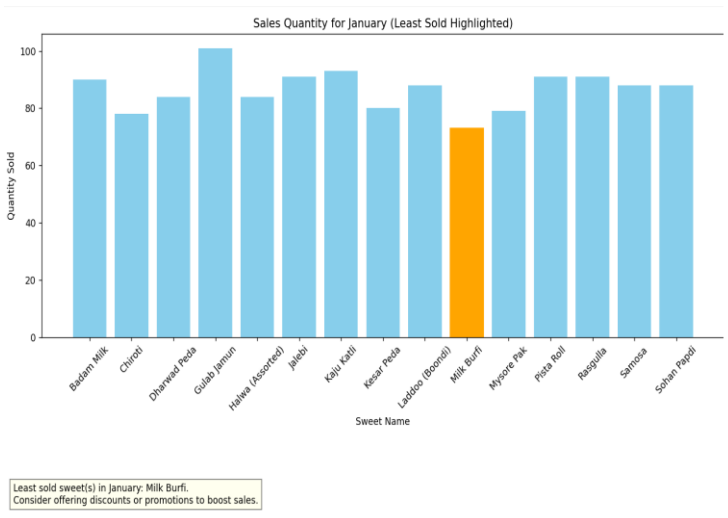


Highest revenue: April with ₹924585.34.
Plan inventory and promotions accordingly for this peak month.

Description:

The Revenue vs Month script analyzes the total revenue generated in each month throughout the dataset and visualizes it, helping Kanti Sweets identify peak revenue months for informed business decisions and strategic planning.

Bar chart- least_sold_sweet_vs_month

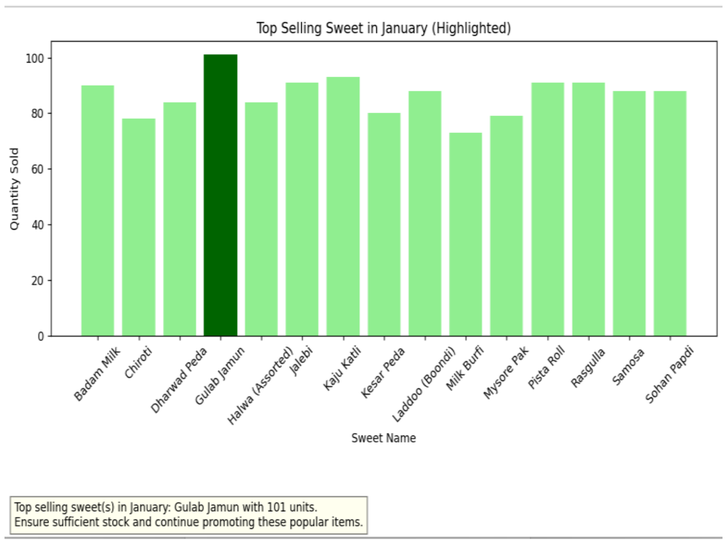


Least sold sweet(s) in January: Milk Burfi.
Consider offering discounts or promotions to boost sales.

Description:

The Least Sold Sweet vs Month script identifies the sweet item with the lowest sales in each month, helping Kanti Sweets understand which products are underperforming so that they can take strategic actions such as targeted promotions, discontinuation, or recipe improvements.

Bar chart- top_selling_sweet_vs_month

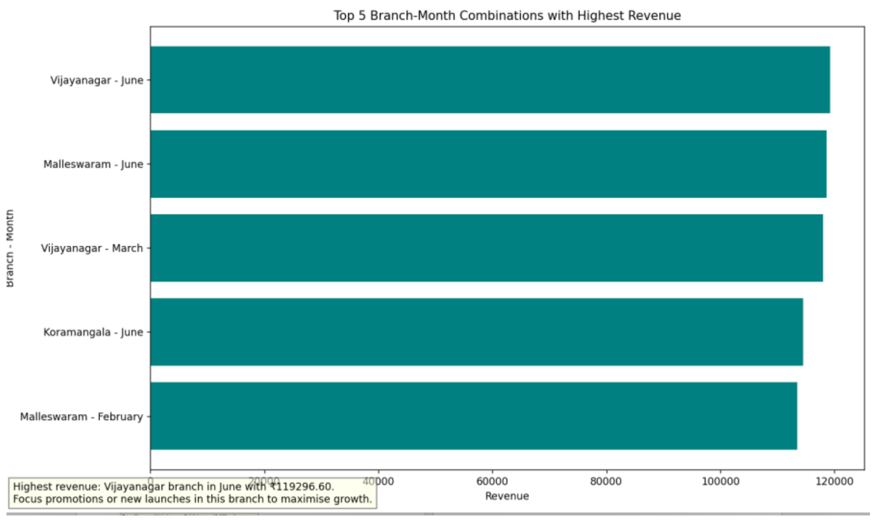


Top selling sweet(s) in January: Gulab Jamun with 101 units.
Ensure sufficient stock and continue promoting these popular items.

Description:

This script identifies the most sold sweet item in each month by analyzing transaction data. It helps Kanti Sweets understand which sweet is the top performer in each month, enabling targeted promotions, optimized production, and stock management based on seasonal demand patterns.

Horizontal bar graph- revenue_vs_branch



Description:

The Revenue vs Branch script analyzes the total revenue generated by each branch location. This helps Kanti Sweets identify high-performing branches and branches with lower revenue, supporting effective regional decision-making, targeted marketing, and resource allocation.

CLOSURE

The Kanti Sweets Sales Analysis Dashboard is a comprehensive and intuitive application that transforms raw sales transactions into meaningful business insights. It systematically cleans and prepares data, generates insightful visualizations highlighting sales trends, revenue distribution, and product performance, and integrates these analyses into a user-friendly graphical interface. By revealing peak sales months, top and least-selling sweets, and branch-wise revenue contributions, this tool equips Kanti Sweets management with the knowledge needed to make planning, optimises inventory and staffing, and supports targeted marketing efforts to maximise profitability. Its well-structured and modular design also ensures ease of maintenance and future enhancements, making it a powerful asset for continuous sales analysis and business growth.informed, strategic decisions. The application enhances operational

BIBLIOGRAPHY

Python Programming Language: https://www.python.org/

Pandas Documentation: https://pandas.pydata.org/docs/

Matplotlib Documentation: https://matplotlib.org/stable/contents.html

Seaborn Documentation: https://seaborn.pydata.org/

Tkinter Documentation (Python official docs): https://docs.python.org/3/library/tkinter.html

Matplotlib with Tkinter Integration (FigureCanvasTkAgg): Typically found within Matplotlib's backend documentation or examples.