KANTI SWEETS SALES DATA ANALYTICS SYSTEM

The Kanti Sweets Sales Data Analytics System is a project designed to help analyse sales data for a popular Indian sweets shop called ***Kanti Sweets***. The main goal of this system is to understand customer buying behaviour and sales trends in a simple and visual way. With multiple outlets across Bengaluru and a wide range of traditional Indian sweets, it's important for the business to understand what's selling, when, and where. This system analyses transactional data collected from different branches of Kanti Sweets. It includes key information such as:

- The sweets sold
- The branch where the purchase was made
- The date and time of the transaction
- The cost of the item
- The period of the day (morning, afternoon, evening, night)

To make the analysis simple and accessible, the system has a user-friendly desktop interface (GUI) built using tkinter. Users (such as branch managers or head office staff) can select specific date, then choose a sweet (or analyse all items), then pick a branch location (or include all branches).

Then the system generates a detailed report with:

- Top and least-selling items of the day
- Total revenue generated on that date
- Average sales per transaction
- Most popular time of the day for sales
- Most frequently purchased item combination.

This helps Kanti Sweets monitor daily sales patterns, identify fast- and slow-moving items, understand customer preferences, improve inventory, staffing, and marketing decisions. This analytics system brings a data-driven approach to managing and growing Kanti Sweets, making it easier for the business to respond to changing customer demands and improve overall performance.

Problem Statement:

Kanti Sweets currently lacks an efficient system to analyse daily sales data across its multiple branches. This limits the management's ability to make informed decisions based on product performance and customer behaviour.

- No centralized platform to track daily sales and revenue trends.
- Inability to identify top-performing and underperforming sweets.
- Limited insight into peak sales periods and customer buying patterns.
- Challenges in optimizing inventory, staffing, and promotions.
- Need for a simple, visual, and interactive analytics solution.

Relevant Information:

- **Branches:** 9 locations (e.g., Jayanagar, Malleswaram, Koramangala)
- **Items:** 15 popular Indian sweets & snacks
- **Transaction Span:** Approx. 1.5 years of simulated daily sales
- **Time Period Segmentation:**
    - Morning (6 AM – 12 PM)
    - Afternoon (12 PM – 5 PM)
    - Evening (5 PM – 10 PM)
    - Night (Others)
- **Sweet Prices:** Randomly varied using a multiplier

Output (O/P):

The system provides the following outputs:

- Display summary results like top seller, peak hour, total revenue
- Sales by hour and by time period
- Top 5 selling items by revenue share

- Entire dataset is saved for analysis/reporting
- Shows top 5 most frequent item combinations sold together

Purpose:

- To assist Kanti Sweets' management in identifying sales patterns and performance across locations
- Provide a daily analytics tool that is:
    - Interactive (via GUI)
    - Insightful (via visualizations)
- Enable planning decisions around:
    - Inventory management
    - Promotions for low-selling items
    - Staffing during peak hours

Outcome:

- Clear trends per sweet, branch, and hour.
- Focused marketing & supply decisions.
- Understand favourite item combos.
- Quickly compare performance across locations.

Benefits:

- Helps make data-driven choices for pricing, stocking, promotions
- Identifies customer favourite to ensure stock availability

- Know when and where sales peak—optimize staff and logistics accordingly and GUI simplifies data analysis for non-technical staff.

Plan:

- **Phase 1:** Generate realistic synthetic data for transactions
- **Phase 2:** Create analysis functions (top seller, hourly trends, etc.)
- **Phase 3:** Build GUI using Tkinter
- **Phase 4:** Integrate graphs and tables
- **Phase 5:** Export data to CSV for backup.

Design:

The system is designed with four main components: data generation, GUI interface, analytics engine, and visualization. It begins by generating synthetic transaction data and saving it to a CSV file. Users interact through a Tkinter GUI, selecting date, sweet, and branch filters. The backend processes this data to calculate key sales metrics. Results are shown in a table and visualized using bar and pie charts. The modular structure ensures easy maintenance and future scalability.

Key Components:

- Data Generator: Creates realistic sweet sale transactions.
- GUI: User-friendly input for analysis filters.
- Analytics Engine: Computes KPIs like top seller, peak hour, combos.
- Visualization: Graphical insights using matplotlib.

Implementation:

TECHNOLOGY	ROLE
• Pandas	Data creation, manipulation, filtering, grouping, and aggregation of transactional data

• Random	randomness in item selection, costs and timestamps.
• Matplotlib	Generates visual charts (bar and pie) for sales trends and sweet performance.
• Collections and itertools	Identifies most common sweet combinations purchased together and also combo analysis.
• tkinter	GUI interface for user interaction and displays result.
• datetime	Handles date and time for simulating and analysing transactions.

The implementation of the *Kanti Sweets Sales Analytics System* is modular and built entirely in Python. It begins with synthetic data generation using pandas, numpy, random, and datetime, simulating 15,000 transactions across sweets, branches, and dates. The data is saved as a CSV file on the user's desktop. A tkinter GUI allows users to input a date, select a sweet or branch, and trigger analysis. Based on the selection, the system filters the data, calculates key metrics (like top/least-selling sweets, peak hour, total revenue, and frequent combos using collections and itertools), and displays results in a table. Visual insights are generated using matplotlib with bar and pie charts. The entire setup is interactive, easy to use, and scalable for future enhancements.

Code:

```
import pandas as pd, numpy as np, datetime, random, os, tkinter as tk

from tkinter import ttk, messagebox

import matplotlib.pyplot as plt

from collections import Counter

from itertools import combinations


def generate_kanti_sweets_data(num_records=10000):
```

```
    items = ["Mysore Pak", "Dharwad Peda", "Badam Milk", "Kaju Katli", "Laddoo (Boondi)",
             "Jalebi", "Samosa", "Gulab Jamun", "Rasgulla", "Milk Burfi", "Pista Roll",
             "Kesar Peda", "Sohan Papdi", "Chiroti", "Halwa (Assorted)"]
    branches = ["Jayanagar", "Malleswaram", "Indiranagar", "Koramangala", "Basavanagudi",
                "Vijayanagar", "RR Nagar", "HSR Layout", "Electronic City"]
    base_prices = {
        "Mysore Pak": 400, "Dharwad Peda": 450, "Badam Milk": 80, "Kaju Katli": 800,
        "Laddoo (Boondi)": 300, "Jalebi": 250, "Samosa": 30, "Gulab Jamun": 50,
        "Rasgulla": 55, "Milk Burfi": 380, "Pista Roll": 750, "Kesar Peda": 500,
        "Sohan Papdi": 280, "Chiroti": 150, "Halwa (Assorted)": 350
    }

    start, end = datetime.date.today() - datetime.timedelta(days=550), datetime.date.today()
    data = []
    for i in range(num_records):
        txn = f"TXN{100000 + i}"
        dt = start + datetime.timedelta(days=random.randint(0, (end - start).days))
        t = datetime.time(*(random.randint(a, b) for a, b in [(8,22), (0,59), (0,59)]))
        hour = t.hour
        period = "Morning" if 6 <= hour < 12 else "Afternoon" if 12 <= hour < 17 else "Evening" if 17 <= hour < 22 else "Night"
        selected_items = random.sample(items, random.choices([1,2,3], [70,20,10])[0])
        branch = random.choice(branches)
```

```
        for item in selected_items:
            cost = round(base_prices[item] * random.uniform(0.5, 3.0), 2)
            cost = cost if cost >= 20 else random.uniform(20, 100)
            data.append({"transaction_id": txn, "item_name": item, "date":
dt.strftime("%Y-%m-%d"),
                         "time": t.strftime("%H:%M:%S"), "period": period,
"purchase_cost": cost,
                         "branch_location": branch})
    return pd.DataFrame(data)


def display_table(tree, data, cols):
    tree.delete(*tree.get_children())
    tree["columns"], tree["show"] = cols, "headings"
    [tree.heading(c, text=c) or tree.column(c, width=200, anchor="center") for c
in cols]
    [tree.insert("", "end", values=r) for r in data]


def plot_bar(data, title, xlabel, ylabel, color='skyblue'):
    plt.figure(figsize=(8, 5))
    data.plot(kind='bar', color=color)
    plt.title(title), plt.xlabel(xlabel), plt.ylabel(ylabel)
    plt.grid(axis='y'), plt.tight_layout(), plt.show()


def plot_hourly_sales(df, day):
    df['hour'] = pd.to_datetime(df['time']).dt.hour
```

```
    plot_bar(df.groupby('hour')['purchase_cost'].sum(), f"Hourly Sales on {day}",
"Hour", "Sales (₹)")


def plot_sweet_sales_pie(df):
    s =
df.groupby('item_name')['purchase_cost'].sum().sort_values(ascending=False).
head(5)
    plt.figure(figsize=(6, 6))
    plt.pie(s, labels=s.index, autopct='%1.1f%%', startangle=140)
    plt.title("Top 5 Sweets by Sales"), plt.tight_layout(), plt.show()


def plot_period_sales_bar(df):
    plot_bar(df.groupby('period')['purchase_cost'].sum(), "Sales by Period",
"Period", "Sales (₹)", color='orange')


def find_favorite_combos(df):
    grouped = df.groupby('transaction_id')['item_name'].apply(list)
    combos = [pair for items in grouped if len(items) > 1 for pair in
combinations(sorted(items), 2)]
    return Counter(combos).most_common(5)


def perform_custom_daily_analysis(df, date_str, sweet, branch, tree):
    try:
        sel_day = pd.to_datetime(date_str).date()
    except ValueError:
        return messagebox.showerror("Invalid Date", "Use YYYY-MM-DD format.")


    df['date'] = pd.to_datetime(df['date'])
```

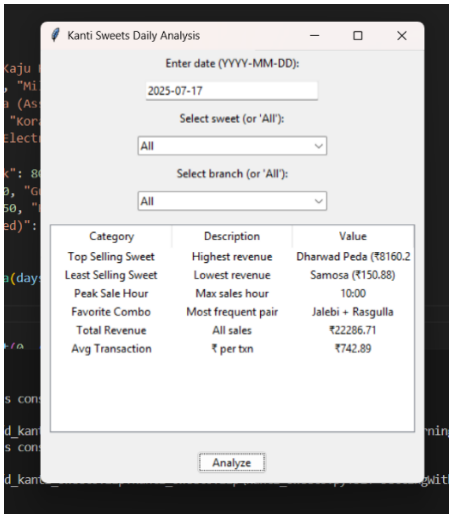
```
    day_df = df[df['date'].dt.date == sel_day]

    if branch != "All":
        day_df = day_df[day_df['branch_location'] == branch]
        if day_df.empty:
            return messagebox.showinfo("No Data", f"No transactions at '{branch}' on {sel_day}.")

    if day_df.empty:
        return messagebox.showinfo("No Data", f"No transactions for {sel_day}.")

    output = []
    total_sales = day_df['purchase_cost'].sum()
    avg_txn = day_df.groupby('transaction_id')['purchase_cost'].sum().mean()

    if sweet == "All":
        sales = day_df.groupby('item_name')['purchase_cost'].sum()
        top, least = sales.idxmax(), sales.idxmin()
        peak_hour = pd.to_datetime(day_df['time']).dt.hour
        peak_hour_val = day_df.groupby(peak_hour)['purchase_cost'].sum().idxmax()
        combo = find_favorite_combos(day_df)
        combo_text = f"{combo[0][0][0]} + {combo[0][0][1]}" if combo else "N/A"

        output = [
            ("Top Selling Sweet", "Highest revenue", f"{top} (₹{sales[top]:.2f})"),
            ("Least Selling Sweet", "Lowest revenue", f"{least} (₹{sales[least]:.2f})"),
```

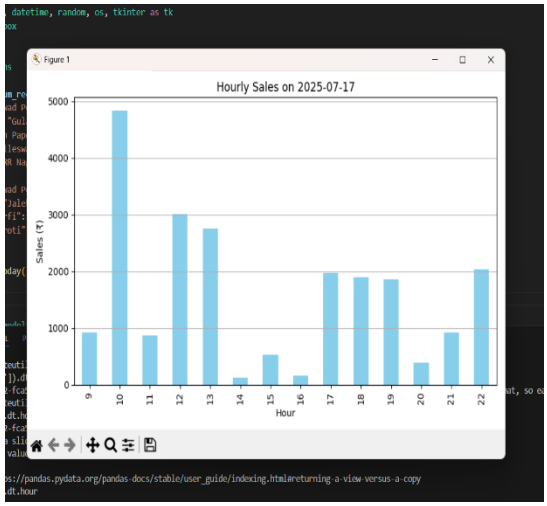
```
            ("Peak Sale Hour", "Max sales hour", f"{peak_hour_val}:00"),
            ("Favorite Combo", "Most frequent pair", combo_text),
            ("Total Revenue", "All sales", f"₹{total_sales:.2f}"),
            ("Avg Transaction", "₹ per txn", f"₹{avg_txn:.2f}")
        ]

        display_table(tree, output, ["Category", "Description", "Value"])
        plot_hourly_sales(day_df, sel_day)
        plot_sweet_sales_pie(day_df)
        plot_period_sales_bar(day_df)
    else:
        s_df = day_df[day_df['item_name'] == sweet]
        if s_df.empty:
            return messagebox.showinfo("No Sweet Data", f"No sales of '{sweet}' on {sel_day}.")
        time_summary = s_df.groupby('period')['purchase_cost'].sum().reset_index()
        output = [(f"{sweet} - {row['period']}", "Sales", f"₹{row['purchase_cost']:.2f}") for _, row in time_summary.iterrows()]
        display_table(tree, output, ["Category", "Description", "Value"])
        plot_hourly_sales(s_df, sel_day)
        plot_period_sales_bar(s_df)

def run_gui():
    df = generate_kanti_sweets_data(15000)
    sweets = ["All"] + sorted(df['item_name'].unique().tolist())
    branches = ["All"] + sorted(df['branch_location'].unique().tolist())
```

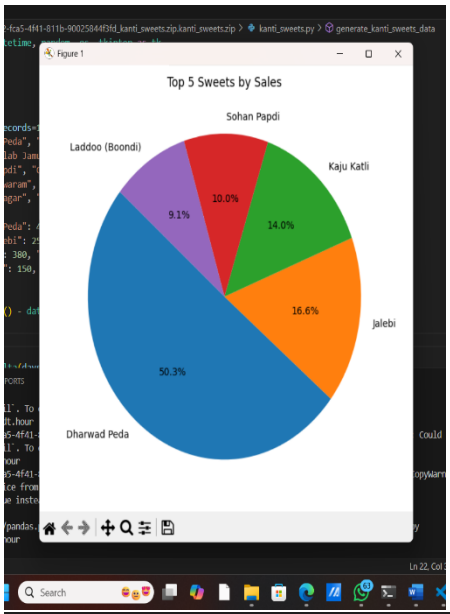
```
    path = os.path.join(os.path.expanduser("~"), "Desktop",
"kanti_sweets_transactions.csv")
    try: df.to_csv(path, index=False); print(f"Saved to {path}")
    except: df.to_csv("kanti_sweets_transactions.csv", index=False); print("Saved
locally due to error.")

    root = tk.Tk(); root.title("Kanti Sweets Daily Analysis")
    ttk.Label(root, text="Enter date (YYYY-MM-DD):").pack(pady=5)
    date_entry = ttk.Entry(root, width=30); date_entry.pack(pady=5)

    ttk.Label(root, text="Select sweet (or 'All'):").pack(pady=5)
    sweet_var = tk.StringVar(); sweet_combo = ttk.Combobox(root,
textvariable=sweet_var, values=sweets, width=30)
    sweet_combo.set("All"); sweet_combo.pack(pady=5)

    ttk.Label(root, text="Select branch (or 'All'):").pack(pady=5)
    branch_var = tk.StringVar(); branch_combo = ttk.Combobox(root,
textvariable=branch_var, values=branches, width=30)
    branch_combo.set("All"); branch_combo.pack(pady=5)

    tree = ttk.Treeview(root); tree.pack(padx=10, pady=10, fill='x')
    ttk.Button(root, text="Analyze", command=lambda:
perform_custom_daily_analysis(
        df, date_entry.get(), sweet_var.get(), branch_var.get(),
tree)).pack(pady=10)

    root.mainloop()
```

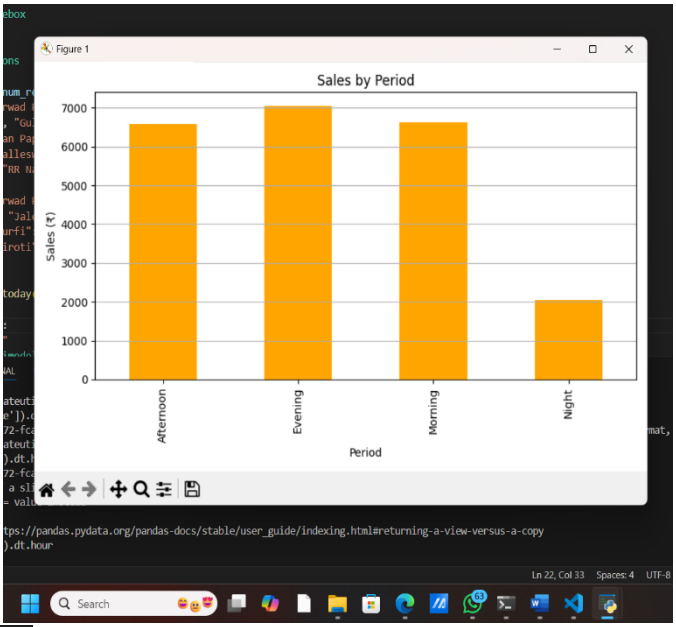
```
if __name__ == "__main__":
    run_gui()
```

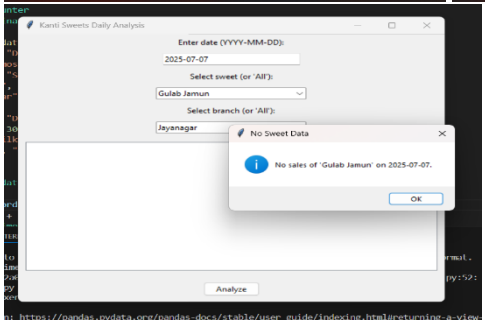
Output:











Closure:

The Kanti Sweets Sales Analytics System is a Python-based desktop application that simulates and analyses daily sales data of sweets across multiple branches. Using a combination of data generation, GUI interaction, and graphical visualization, the system helps identify key business insights such as top-selling products, peak sales hours, and customer purchasing patterns. The project demonstrates how simple technologies like pandas, tkinter, and matplotlib can be effectively used to build an interactive and insightful analytics tool. This system can support real-world decision-making in inventory planning, marketing, and operational efficiency for retail sweet businesses. It also lays the foundation for future integration with real-time databases and advanced predictive models.

Bibliography:

A list of resources and references used during the development of the Kanti Sweets Sales Analytics System project are:

1. Kanti Sweets – Official Website (For Domain Context) https://www.kantisweets.com/ (Referenced for understanding the variety of sweets and retail context)
2. Pandas Documentation - https://pandas.pydata.org/docs/ (Used extensively for data creation, manipulation, grouping, and filtering)
3. NumPy Documentation - https://numpy.org/doc/ (Used for numerical operations and random value generation)
4. Tkinter GUI Programming - https://tkdocs.com/ (Comprehensive guide for designing user interfaces using tkinter and ttk)
5. Matplotlib Documentation - https://matplotlib.org/stable/contents.html (Used for generating bar charts and pie charts for data visualization)
6. Collections and itertools – Python Standard Library https://docs.python.org/3/library/collections.html https://docs.python.org/3/library/itertools.html (Utilized for counting item combinations and analysing customer purchase patterns)