Interactive Sales Dashboard Documentation

# Overview

The **Interactive Sales Dashboard** is a Python-based application designed to analyze and visualize sales data. It leverages Tkinter for the graphical user interface (GUI) and Matplotlib for generating visual charts. Users can explore various aspects of the sales data through different chart types, including bar charts, pie charts, and line graphs. This dashboard is interactive, allowing users to sort data, select different views, and visualize trends over time.

**Repository:** [**Data Factz Project**](https://github.com/rakesh-madadi/Datafactz-Project)

# Project Dependencies

To run this project, ensure the following Python libraries are installed:

1**. tkinter:** For creating the GUI  
2. **pandas:** For data manipulation and analysis  
3. **matplotlib:** For generating charts and plots  
4. **ttkbootstrap:** For enhanced styling and additional widgets

# Project Structure

The project is structured around a single main class:

* **InteractiveSalesDashboard Class**: This is the core class responsible for creating the GUI, loading the data, and displaying various visualizations.

## Attributes:

1. **master:** The main window for the GUI.  
2. **df:** The sales data loaded from an Excel file.  
3. **summary\_df:** The summary data processed from the sales data.  
4. **quantity\_sort\_order:** Controls the sorting order for quantity charts.  
5. **revenue\_sort\_order:** Controls the sorting order for revenue charts.  
6. **current\_chart:** Tracks the currently displayed chart.

## Methods:

**\_\_init\_\_(self, master):** Initializes the GUI, loads the data, and sets up the layout.

**load\_and\_process\_data(self):** Reads and preprocesses the sales data from an Excel file. Handles missing values, converts columns to the appropriate data types, and formats the data for analysis.

**clean\_data(self, df)**: Cleans the data by removing missing values and converting columns.

**format\_data(self, df):** Processes the data for summary statistics and visualization.

**display\_data(self)**: Displays the processed summary data in a Treeview widget.

**show\_pie\_chart(self):** Generates and displays a pie chart of total revenue by product.

**show\_quantity\_bar\_chart(self)**: Generates and displays a bar chart of total quantity sold by product.

**show\_revenue\_bar\_chart(self):** Generates and displays a bar chart of total revenue by product.

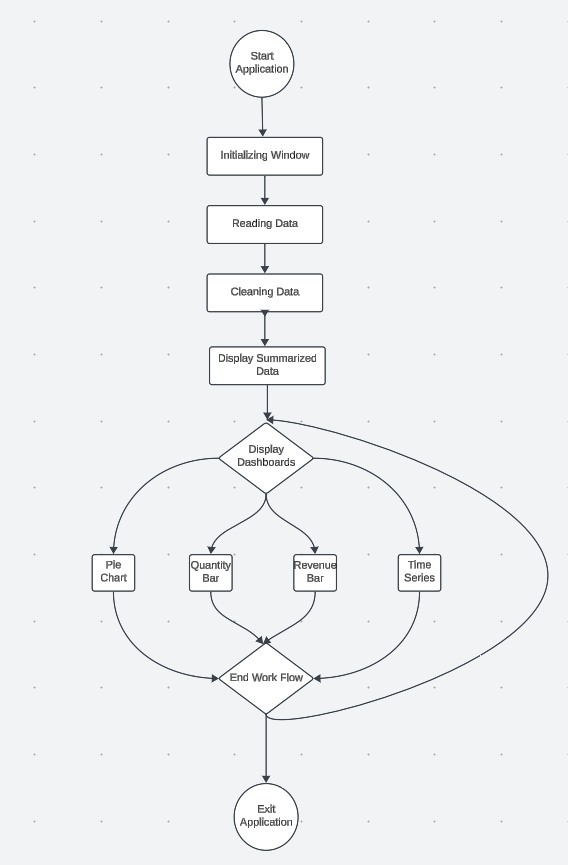
**show\_quantity\_line\_graph(self):** Generates and displays a line graph of total quantity sold over time.

# Data Preprocessing

The data is loaded using the pandas library in the load\_and\_process\_data() method. It performs the following steps:

**1. Loading the Excel file**: Attempts to load the specified Excel file.  
2**. Data Cleaning**: Removes missing values, converts columns to appropriate data types, and formats date columns.  
3. **Data Formatting**: Groups the data by product name, calculates total quantities and revenue, and formats the data for display.

**Data Flow Diagram**:



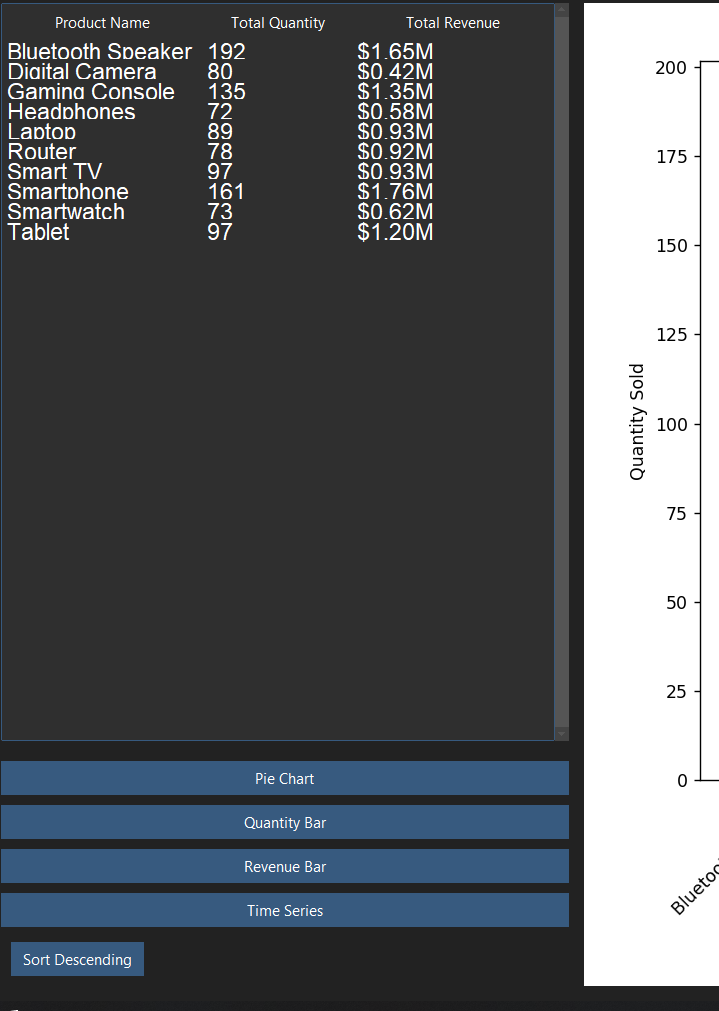
Here is a simple explanation of the data flow diagram:

1. **Start of the Application**: The process begins when the application is initiated, leading to the window initialization where the main UI components are set up.
2. **Reading and Cleaning Data**: Once the window is initialized, the application reads sales data (from an Excel file) and cleans it by handling missing or erroneous data points.
3. **Display Summarized Data**: After the data is cleaned, it is summarized to show total quantities and revenues per product, and displayed in the treeview in the application.
4. **Display Dashboards**: Based on user input, the dashboard displays different types of visualizations such as Pie Charts, Quantity Bar Charts, Revenue Bar Charts, or Time Series Graphs, each using the summarized data.
5. **End Workflow**: After the user completes their analysis, they can exit the application, which ends the workflow, closing the application.
6. **Interaction with Buttons**: The user can interact with buttons on the dashboard (e.g., Pie Chart, Quantity Bar, Revenue Bar, Time Series) to switch between different visual representations of the sales data.
7. **Data Sorting**: When viewing bar charts, the user can toggle between ascending and descending sorting of the quantity or revenue data using sort buttons. These buttons dynamically adjust the chart presentation based on the chosen sort order.
8. **Real-Time Chart Updates**: Each time the user selects a different chart type or sorting option, the dashboard dynamically updates the visualizations to reflect the corresponding dataset or sort order in real-time.
9. **Chart Labels**: In the bar and line charts, labels are automatically added to provide clear visibility of data points, such as total quantity sold or revenue for each product, or specific values on time-series graphs.
10. **Error Handling and User Notifications**: If there are issues with data loading or cleaning (e.g., missing data or incorrect formatting), error handling mechanisms trigger pop-up messages to notify the user, ensuring a smooth and informative user experience.

# Dashboard Layout

The dashboard is divided into two main sections:

1. **Left Frame**: Contains the summary data and control buttons for selecting different chart types.

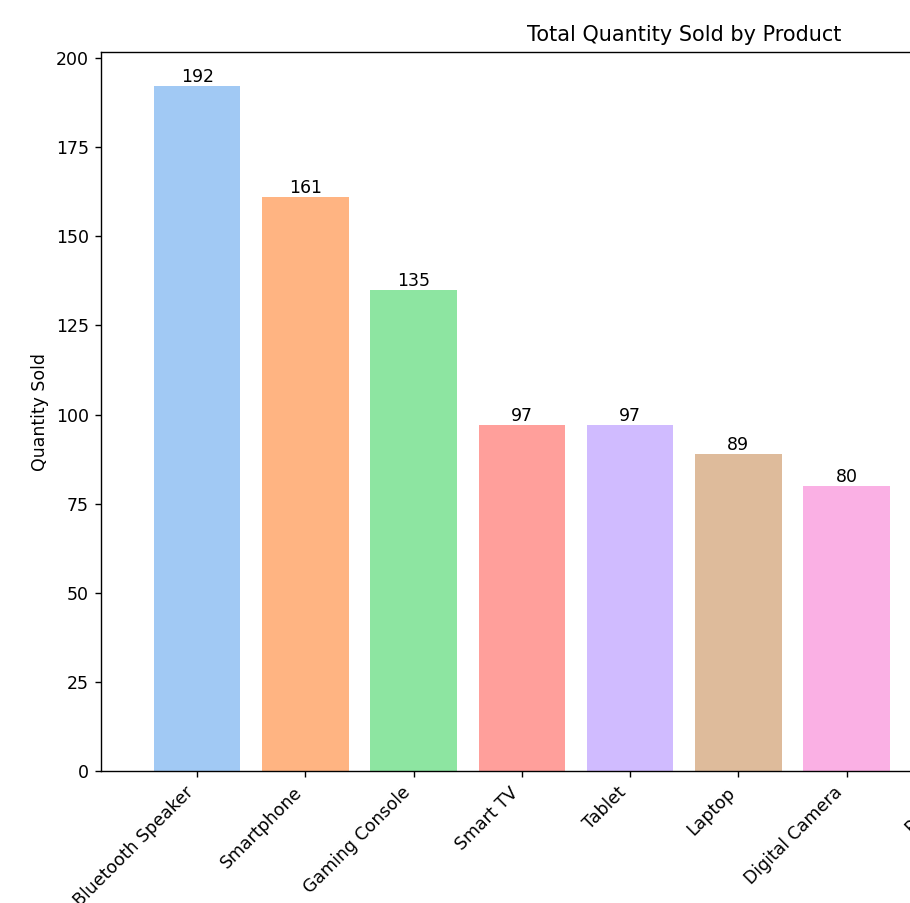


CONTROL BUTTONS

SUMMARY DATA

1. **Right Frame**: Displays the selected chart.

DISPLAY OF THE SELECTED CHARTS



# Functionality and Visualization

The dashboard supports the following visualizations:

1. **Pie Chart**: Shows the distribution of total revenue by product.  
2. **Quantity Bar Chart**: Displays total quantity sold by product, with sorting options.  
3. **Revenue Bar Chart**: Displays total revenue by product, with sorting options.  
4. **Quantity Line Graph:** Shows the trend of total quantity sold over time.

# Error Handling

The application includes basic error handling in methods such as load\_and\_process\_data() and clean\_data(). If the Excel file is not found or contains invalid data, appropriate error messages are displayed.

# Execution

To execute the project, follow these steps:

1. Clone the repository from GitHub.  
2. Install the required dependencies using pip.  
3. Run the script using the command: python sales\_dashboard.py