ADDIS ABABA SCIENCE AND TECHNOLOGY UNIVERSITY COLLEGE OF ENGINEERING

Department of Software Engineering

Software component Design assignment

Group members	Id
Ruth Wossen	1113/13
Saron Meseret	1149/13
Yidnekachew Tebeje	1314/13
Yohannes Getachew	1339/13
Samuel Amsalu	1129/13

Overview

The Sales Dashboard is an interactive web application designed to provide a comprehensive visualization of sales data. Built using Streamlit and Python, it allows users to analyze key performance indicators (KPIs) related to sales, such as total sales, margins, and transaction counts. Users can filter the data by various parameters, including financial year, retailer, company, and month, to gain insights into sales performance and trends over time.

The dashboard aims to facilitate data-driven decision-making for stakeholders by presenting critical information in a user-friendly interface. Through various visualization options, including line charts, area charts, and bar charts, users can easily interpret data and identify patterns.

Key Features

Data Import and Preprocessing

- **Data Loading**: Ability to load sales data from a CSV file.
- **Feature Extraction**: Creation of time-related features (financial year and month) for better analysis.

User Interaction

- **Sidebar Filters**: Users can filter data based on: Financial year, Retailer, company, financial month
- **Chart Selection**: Users can choose the type of chart to visualize the data (e.g., Line, Area, Bar).

Data Visualization

- Dynamic charts that update based on user-selected filters:
    - **Line Chart**: Visualizes trends in sales over time.
    - **Area Chart**: Shows cumulative sales data.
    - **Bar Chart**: Compares sales across different categories (retailers and companies).

User Interface Design

- Responsive and intuitive layout designed for ease of use across devices.
- Consistent styling and branding, including a footer with developer information.

Tools used

Python: The programming language used for development.

Streamlit: Framework for building web applications and creating interactive user interfaces.

CSV File: The data source (e.g., data.csv) used for loading sales data.

Component

User Interface Components

1. **Main Application Component**
    - **Responsibilities**:
        - Initialize the application and configure the layout.
        - Load and preprocess data.
        - Manage user interactions via the sidebar.
2. **Sidebar Component**
    - **Responsibilities**:
        - Provide filters for user input (Year, Retailer, Company, Month).
        - Allow users to select the type of chart to display.
3. **Metrics Display Component**
    - **Responsibilities**:
        - Display key performance indicators (KPIs) like Total Sales, Total Margin, etc.
4. **Chart Visualization Component**

- **Responsibilities**:
    - Render different chart types based on user selection.

5. **Revenue Insights Component**
    - **Responsibilities**:
        - Display insights on retailer and company performance based on revenue.

Data Processing Components

1. **Preprocessor Module**
    - **Responsibilities**:
        - Handle data loading, transformation, and feature extraction.
    - **Key Functions**:
        - fetch_time_features(): Extracts date-related features.
        - fetch_top_revenue_retailers(): Calculates top revenue-generating retailers.
        - fetch_top_revenue_companies(): Calculates top revenue-generating companies.

Component Interaction

- **Main Application Component** interacts with **Preprocessor Module** to fetch and preprocess data before displaying it.
- **Sidebar Component** gathers user input, which the **Main Application Component** uses to filter data.
- **Metrics Display Component** receives filtered data from the **Main Application Component** and computes key metrics.

- **Chart Visualization Component** uses the filtered data to generate visualizations based on the selected chart type.
- **Revenue Insights Component** displays additional insights by calling functions from the **Preprocessor Module**.
- **Footer Component** is static and appears across all pages, providing consistent **branding and links.**

Conclusion

The Sales Dashboard application serves as a powerful tool for analyzing and visualizing sales data, enabling users to derive actionable insights through an intuitive interface. By leveraging Python, Streamlit, and Pandas, the dashboard allows for seamless interaction with sales data, providing users with the ability to filter information by key parameters such as year, retailer, company, and month.

The incorporation of a dedicated Preprocessor Module ensures that the data is clean, well-structured, and ready for analysis, facilitating feature extraction and aggregation processes. The dashboard's design emphasizes user experience, offering dynamic visualizations that update in real time based on user selections. This interactivity enhances the decision-making process, allowing stakeholders to identify trends, monitor performance, and make informed choices regarding sales strategies.

Overall, the Sales Dashboard not only simplifies the complexities of data analysis but also empowers users to explore their data effectively. Future enhancements could include advanced analytics features, user authentication, and additional visualization options, further expanding its capabilities. This application stands as a testament to the potential of data-driven decision-making in sales and business management.