Automated Inventory Analysis and Reporting Platform

To reduce the complexity and operational burden associated with relying on Excel spreadsheets and static Power BI reports, I designed and developed a custom Python-based analytical application tailored specifically for warehouse inventory analysis. The primary objective of this solution was to provide my manager and the planning team with a more intuitive, interactive, and automated platform for exploring inventory data across multiple warehouses without requiring advanced technical skills or manual data manipulation.

The application was implemented as a web-based graphical interface using Python's Streamlit framework, enabling seamless access through a browser-based environment. Users can easily upload Excel files containing stock source data and fabric inventory data as shown in Fig. 1, after which the system automatically processes, cleans, and integrates the datasets. This automated pipeline eliminates repetitive manual preprocessing steps, significantly reducing the time and effort required to prepare inventory reports while minimizing the risk of human error.

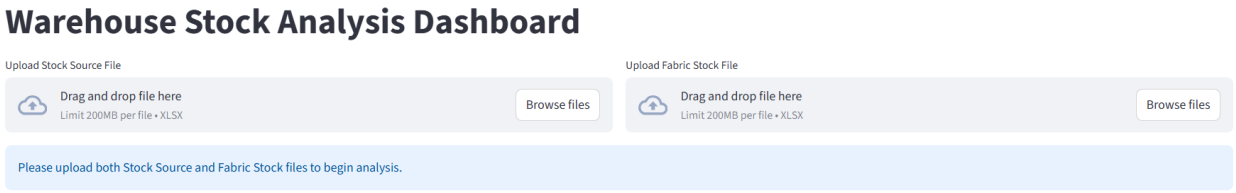

Figure 1: Python-based Inventory Analysis Application - Data Upload Interface

Once the data is uploaded, the application performs a series of analytical computations to generate meaningful insights. Inventory quantities are aggregated across warehouses and visually presented using dynamic bar charts, ranking cards, and pivot tables. These visual elements provide a clear overview of stock distribution as illustrated in Fig. 2, allowing users to quickly compare inventory levels across warehouses and identify high-volume or underutilized storage locations. Interactive filters enable users to focus on specific warehouses, time ranges, or inventory categories, thereby supporting exploratory analysis and targeted decision-making.

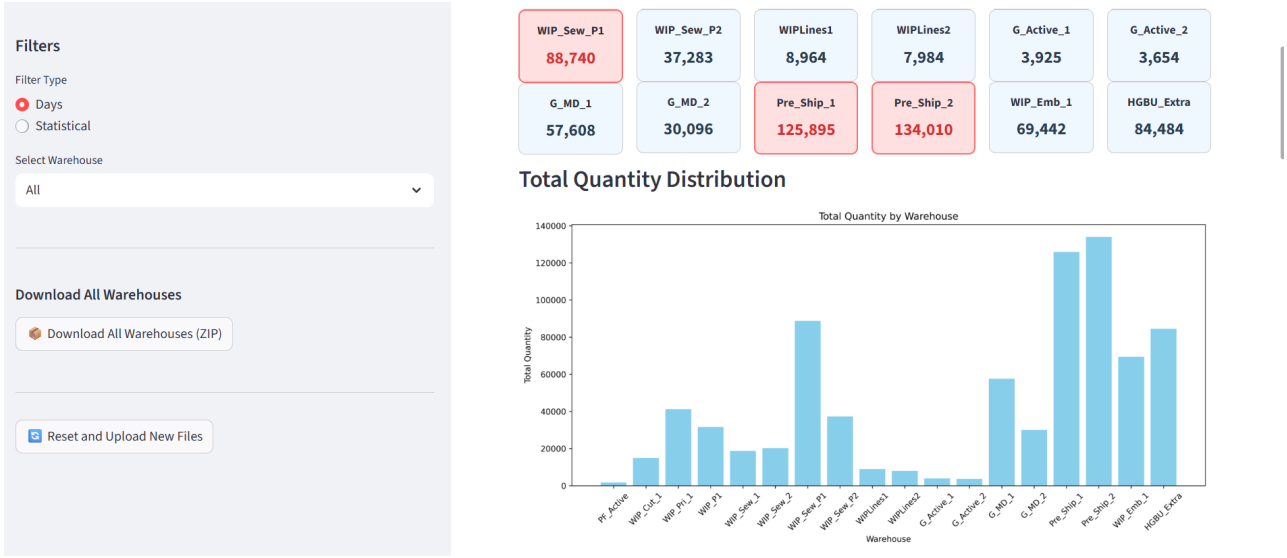

Figure 2: Python-based Inventory Analysis Application - Data Visualization Interface

To enhance temporal analysis, the application calculates the number of days each project or inventory item has remained idle based on its last recorded movement or transaction date. Items are then categorized into predefined time intervals (e.g., 0–15 days, 16–30 days, up to 180+ days) (Fig. 3), enabling planners to assess inventory aging patterns and detect

prolonged inactivity. This time-based segmentation helps identify potential delays, inefficiencies, or blocked projects within the warehouse network.

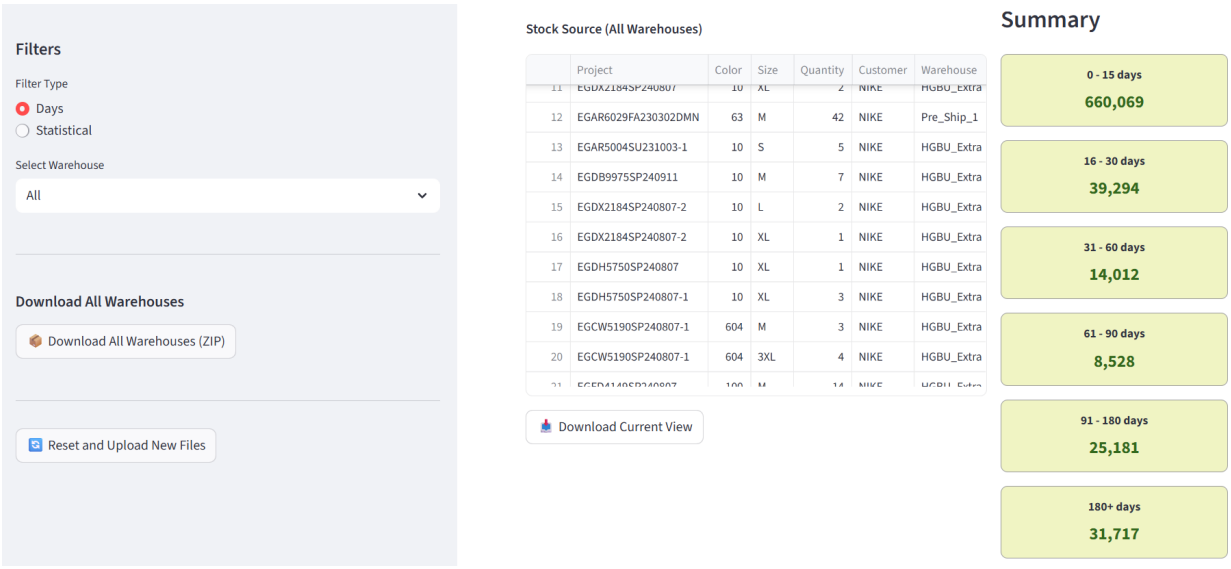

Figure 3: Python-based Inventory Analysis Application - Idle Time Categorization

In addition to descriptive analysis, I incorporated a statistical measure to objectively identify projects that have remained idle for a significantly longer duration than usual. This was achieved by computing warehouse-specific confidence intervals for inactivity duration and flagging items that exceed statistically significant thresholds. Projects classified as critical under this criterion are highlighted within the dashboard (Fig. 4), allowing planners to distinguish between normal operational delays and abnormal stagnation that may indicate bottlenecks or process failures.

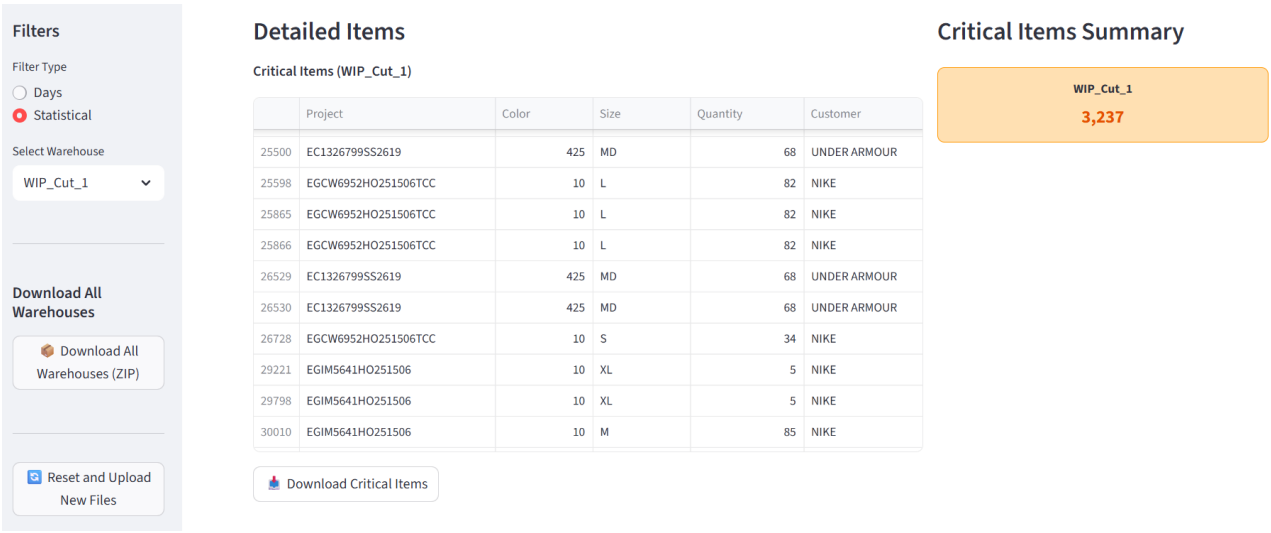

Figure 4: Python-based Inventory Analysis Application - Critical Projects Identification

The application further supports proactive inventory management by emphasizing warehouses with critical inventory levels and delayed projects. Dedicated summary panels and visual indicators draw attention to warehouses where stock accumulation or inactivity is most severe, enabling users to quickly pinpoint operational bottlenecks. This functionality is particularly valuable for identifying which warehouses are associated with projects that have not progressed for an extended period, thereby supporting root-cause analysis and corrective action planning.

To facilitate collaboration and reporting, I implemented comprehensive data export and communication features. Users can download filtered views of the data in CSV or Excel format, either for individual warehouses or for all warehouses

collectively. Additionally, the application includes an integrated email reporting module that allows users to automatically generate and send summarized inventory reports directly from the interface. Based on selected filters and departmental mappings, the system dynamically prepares warehouse-specific Excel attachments and composes a structured email highlighting key statistics, critical projects, and delayed items as shown in Fig. 5. This automation significantly improves reporting efficiency and ensures timely dissemination of actionable information to relevant stakeholders.

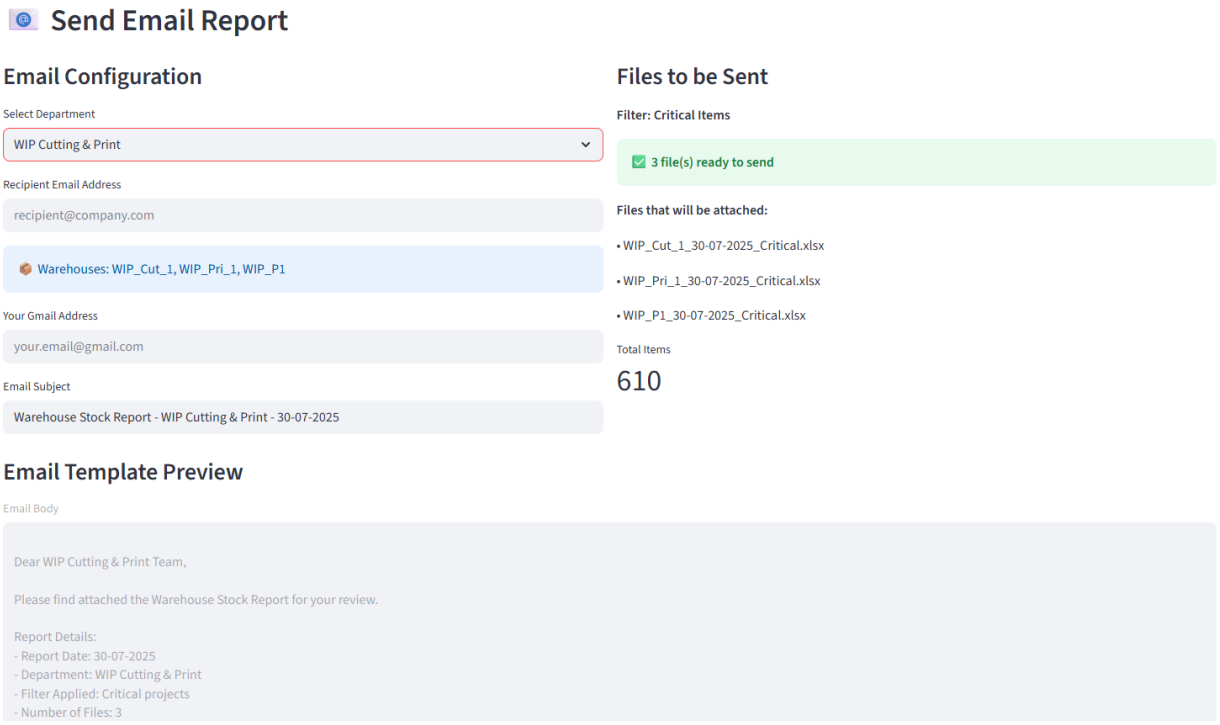

Figure 5: Python-based Inventory Analysis Application - Automated Email Reporting

Overall, this Python-based application provides a scalable, user-friendly alternative to traditional spreadsheet-driven analysis. By combining automated data processing, interactive visualization, statistical anomaly detection, and integrated reporting, the solution enhances visibility across warehouse operations and supports more effective, data-driven planning and inventory management decisions.