**Technology Stack**

To execute this data analytics solution effectively, a carefully selected technology stack was adopted. It balances low-code and pro-code components, ensuring flexibility for developers while remaining user-friendly for analysts and business users.

Overview of Tools & Technologies

|  |  |  |
| --- | --- | --- |
| Category | Tool/Technology | Purpose |
| Visualization Platform | Tableau Desktop | Build interactive dashboards, analyze trends, and generate stories from complex datasets using drag-and-drop capabilities. |
| Web Application Layer | Flask (Python) | Host dashboards securely on internal networks or web portals, offering seamless integration and user access management. |
| Data Input Formats | CSV, Excel, SQL | Ingest historical and real-time sales data, product placement records, customer types, and store layouts. |
| Preprocessing Scripts | Python (Pandas) | Clean, normalize, and structure data fields such as product category, sale timestamp, and placement type before Tableau use. |
| Automation | Tableau Scheduled Refresh | Enable periodic updates to dashboards without manual intervention by syncing with live or regularly uploaded data sources. |
| Dashboard Distribution | SMTP (Email Integration) | Automatically email dashboards to designated recipients or groups, especially for executive updates and end-of-week reviews. |
| Access Control | Tableau Permissions | Set up role-based permissions for each dashboard and data source to avoid unauthorized access. |
| QA and Data Validation | Tableau Prep / Filters | Clean anomalies, eliminate duplicates, validate field types, and maintain dataset quality before visualization. |
| Deployment Interface | HTML Embed Snippets | Embed dashboard iframe views inside Flask-based websites with authentication logic to prevent public access. |
| Version Control | Tableau Workbook History | Maintain change logs, enable rollback of visualizations, and manage collaborative edits. |

**Rationale for Choosing This Stack**

1. **Tableau for Visualization**  
   Tableau was chosen for its rich visualization options, easy integration with data sources, and ability to create both dashboards and data stories. It supports deep analytics without writing code and lets analysts create layered insights like sales heatmaps, time-series sales flows, and category-level comparisons.
2. **Flask for Lightweight Hosting**Flask is ideal for small-to-medium-scale internal deployments. It supports secure routing, easy embedding of dashboards, and offers scalability through extensions like Flask-Login and Flask-Mail. Its simplicity makes it excellent for managing multiple dashboards across different user groups.
3. **Python (Pandas) for Data Prep**Pandas is used for advanced data preparation tasks before data is loaded into Tableau. It handles missing values, reshapes datasets, applies calculations, and ensures field consistency—all essential in retail analytics.
4. **Scheduled Automation & Email Integration**Tableau’s refresh schedules and SMTP-based email reports make the solution dynamic and timely. Business users are notified with updated KPIs without logging in manually.
5. **Access Control and Security**Role-based access control using Tableau’s permission management system ensures sensitive sales data and strategic insights are available only to authorized individuals.

**Sample Workflow in Practice**

1. Import Data from regional sales spreadsheets or SQL exports.
2. Preprocess with Python to calculate key fields like average sales per shelf.
3. Load Data into Tableau and create initial dashboards by product category, placement zone, and time trend.
4. Build Storyboard that connects multiple dashboards to explain a narrative (e.g., why endcap sales outperform middle-shelf).
5. Embed Dashboards into a Flask portal for store managers and marketing executives.
6. Schedule Weekly Email Reports using Tableau’s in-built scheduling and SMTP linking.
7. Iterate on Feedback using change logs and comments collected from internal review meetings.