Executive Summary: Inventory Optimization for Urban Retail Co.

Project Title: Solving Inventory Optimization Using Advanced SQL
Team: Data devourer
Toolstack: MySQL, Jupyter Notebook, Plotly, Pandas, dbdiagram.io, VSCode

Project Overview:

Urban Retail Co. is a mid-sized retail chain with both physical stores and an online presence, carrying thousands of SKUs across multiple regions . As the company grew, it faced **serious inventory challenges**: frequent stockouts of fast-selling items (causing lost sales and unhappy customers) and chronic overstock of slow-moving goods (tying up capital and raising storage costs) . Managers had limited visibility into SKU-level demand, reorder thresholds, or supplier reliability, so replenishment decisions were largely reactive . The objective of our project was to turn these raw sales and inventory records into actionable intelligence. By designing efficient SQL-driven analytics, we aimed to give Urban Retail Co. proactive insight into stock levels and demand trends, enabling the business to forecast needs and set smarter reorder points. In short, we sought to **minimize both stockouts and overstocks** and strengthen overall supply-chain visibility.

Folder Structure:

/inventory-sql-optimisation

|-- sql_views/ --> .sql files of all created views

|-- exports/ --> CSV exports of views and final tables

|-- notebooks/ --> Jupyter notebooks for EDA(data_cleaning.ipynb) & Dashboard(dashboard_inventory_csv.ipynb)

|-- erd/ --> Schema diagram (ERD) PNG

|-- sql scripts/ --> sql_script.sql (contains setting up databases and tables) , sql_script.sql(contains all the queries and views)

|-- executive_summary.txt/ -->final_report.pdf(includes summary and contains final report including dashboard images).

Key work completed includes:

- Database schema normalization and ERD diagram generation
- Creation of fact and dimension tables (fact_inventory, dim_product)
- Development of multiple SQL views for metrics like inventory gap, forecast error, reorder point, etc.
- Dashboards using Plotly, Seaborn, and Matplotlib for KPI reporting
- Assumed lead times for product categories to calculate reorder thresholds

---

- **Data Preparation (CSV ingestion and cleaning):** We started by gathering the provided inventory and sales data from multiple CSV sources. This involved merging and cleaning raw data files (e.g. inventory forecasts, sales logs, etc.) so that all fields were consistent (dates formatted uniformly, missing values handled, SKU and store codes aligned, etc.). The cleaned data was then loaded into a relational database. This ensured our analysis would be based on a single unified dataset: a **fact_inventory** table capturing daily inventory levels, sales, and orders per SKU per store, plus ancillary attributes (promotions, weather, etc.) from the CSVs.

- **Schema Design (ERD):** Next we designed a normalized database schema to organize the data. We created dimension tables for dim_product (product ID, category, price) and dim_store (store ID, region) and a central fact table fact_inventory (date, store_id, product_id, inventory_level, units_sold, etc.). Our Entity-Relationship Diagram illustrates this star schema – each inventory record links to its product and store attributes *Figure:* Diagram of the inventory database schema. By separating static attributes (like product category) into dimensions and keeping the daily transactions in the fact table, we set up a flexible structure for analysis. This ERD (Figure above) shows how every inventory entry joins to the relevant store and product details, enabling efficient SQL queries across these relationships.

- **Advanced SQL Queries (stock & inventory metrics):** With the data in place, we implemented the SQL analytics. Our queries calculated key inventory metrics in line with the project goals . For example, we wrote queries to compute daily stock levels and daily demand per SKU-store combination (using window functions to handle running totals). We built logic to **detect low inventory**: flagging products whose on-hand units had fallen below a dynamic reorder threshold. We also **estimated reorder points** by analyzing historical demand trends (finding average usage per period) and combining that with assumed lead times. To gauge inventory efficiency, we calculated **inventory turnover rates** (annualized sales divided by average stock on hand). All of these queries used optimized SQL techniques (indexed joins, GROUP BY and analytic/window functions) to summarize large datasets quickly. In short, our SQL scripts covered **stock level calculations, low-stock alerts, reorder point**

computation, and turnover analysis , yielding the core numbers needed for decision-making.

- **Analytical Outputs (product and trend insights):** The SQL-driven analysis generated actionable insights about which products and situations needed attention . We identified **fast-moving** SKUs (those selling out quickly) and **slow-moving** SKUs (those that sit in inventory), so the buyer could prioritize high-velocity items. By inspecting orders versus receipts and stockout events, we flagged any **supplier inconsistencies** or delivery delays (so the supply chain team could follow up with underperforming vendors). We also examined **seasonal patterns** in the sales data: for example, we saw certain categories spike during holidays or weather changes (using the seasonality and holiday_promotion fields). These outputs directly inform actions like running timely promotions or planning extra stock for predictable peaks. Overall, the reports highlighted where inventory adjustments were needed to cut holding costs and avoid stockouts .

- **Dashboard Creation (Plotly + Pandas):** To communicate these insights, we built an interactive inventory dashboard using Python. We used Pandas to load the SQL-exported CSV views and Plotly to create charts. The dashboard includes bar charts, line graphs, and tables showing KPI summaries for each store and category. For instance, one bar chart (embedded below) plots each store's forecast error, making it easy to spot which stores consistently over- or under-forecast demand. Another chart shows the top 10 SKUs by sales velocity or by overstock units. Because the dashboard is interactive, managers can filter by region or time period. *Figure:* Example dashboard chart (store-wise forecast error). The visuals helped us present findings in an intuitive way: colorful charts let stakeholders quickly grasp which stores have the worst forecast accuracy, which items need reorder attention, and so on.

- **Reorder Point Calculation (lead-time based):** Finally, we formalized how to calculate reorder points. Lacking actual supplier lead-time data, we assumed typical lead times by product category (for example, perishables might have a 3-day lead time, general merchandise two weeks). Using each category's assumed lead time and the product's average daily demand.We implemented this in SQL and verified the results. The output was a per-SKU threshold value indicating how much stock should be on hand. Our system then compared current inventory to this point to generate low-stock alerts. This completes the loop: now every SKU has a data-driven reorder trigger, replacing the guesswork.

Business Impact:

This analysis empowers Urban Retail Co. to make **smarter inventory decisions** backed by data. By using actual sales and stock history, the company can **proactively** set reorder levels rather than waiting for a stockout. The immediate benefit is fewer stockouts of popular items and less overstock of slow-movers – exactly the goals outlined in the project brief . For example, knowing which products are fast-selling allows the buyer to increase their reorder quantity or order frequency, reducing lost sales. Conversely, identifying excess inventory on hand suggests where to tighten purchasing or run promotions, freeing up cash.

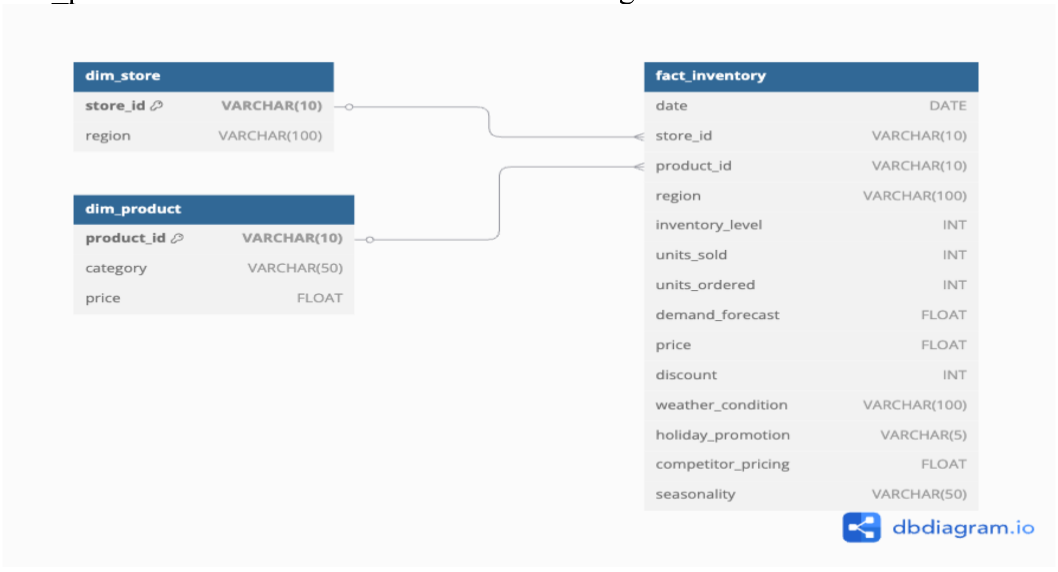
In the bigger picture, these improvements boost supply-chain efficiency: logistics teams can plan warehouse replenishments more accurately when demand peaks are forecasted by season. Store managers gain visibility into their upcoming needs (day's-of-inventory remaining), so they can plan transfers or emergency orders before shelves run out. Customer satisfaction improves because high-demand items stay in stock, and the company's working capital is freed from dead inventory. All of this contributes to profitability – faster turnover means higher gross margin on perishable lines and lower holding costs on bulky goods.

In summary, our SQL-driven solution turns raw transaction data into actionable KPIs, meeting the expected business impact: **smarter inventory decisions, reduced stockouts/overstocks, a leaner supply chain, happier customers, and ultimately higher profits** .

Visuals:

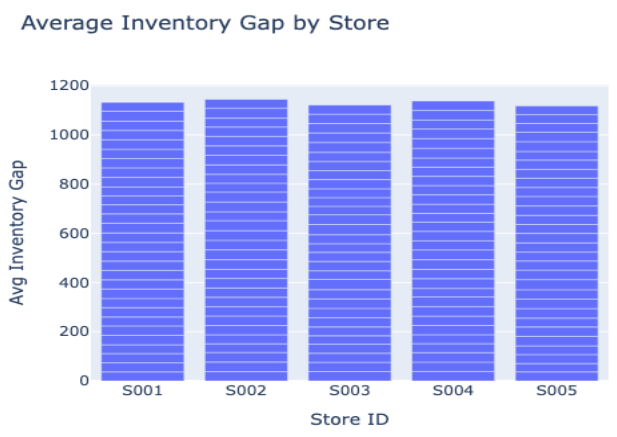
The project deliverables included several key visuals:

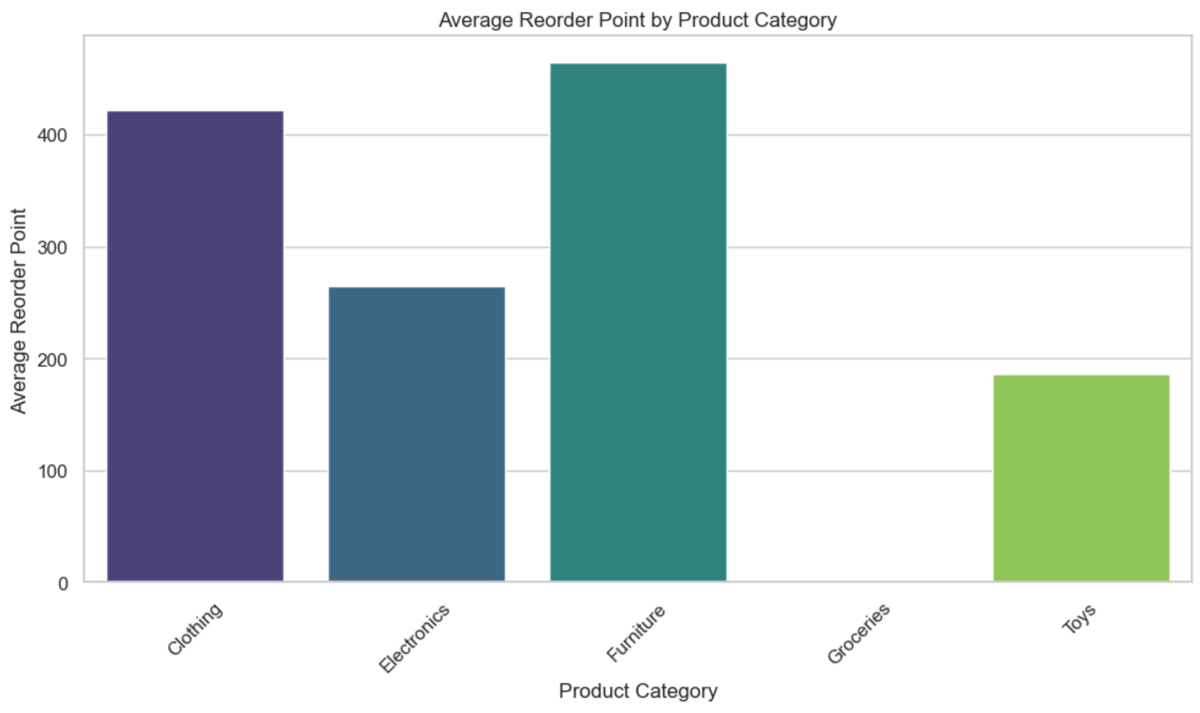
- **ERD Diagram:** The file schema.png (shown above) is the entity-relationship diagram for our database. It clearly illustrates the fact table linked to dim_store and dim_product. Stakeholders can refer to this image to understand the data model.

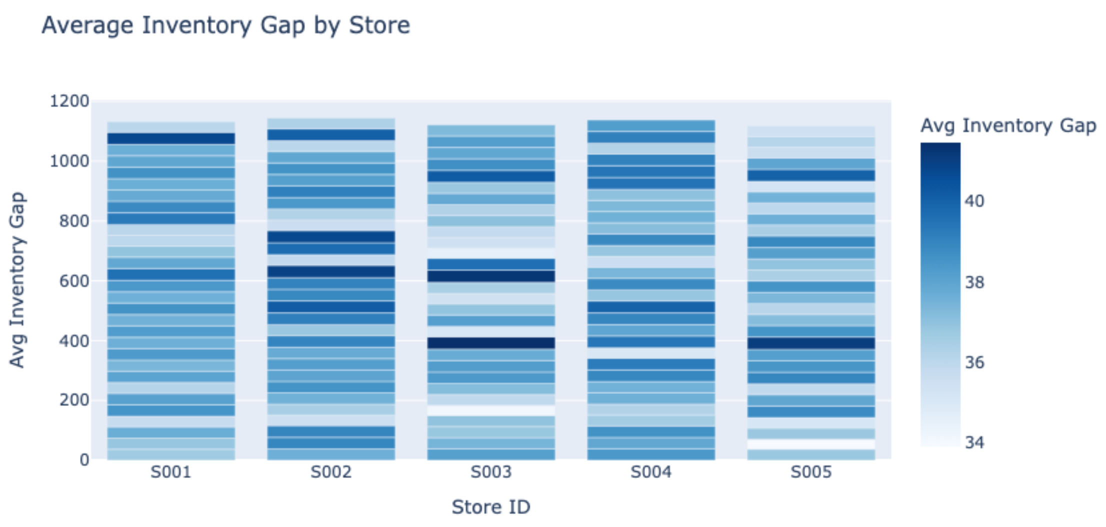


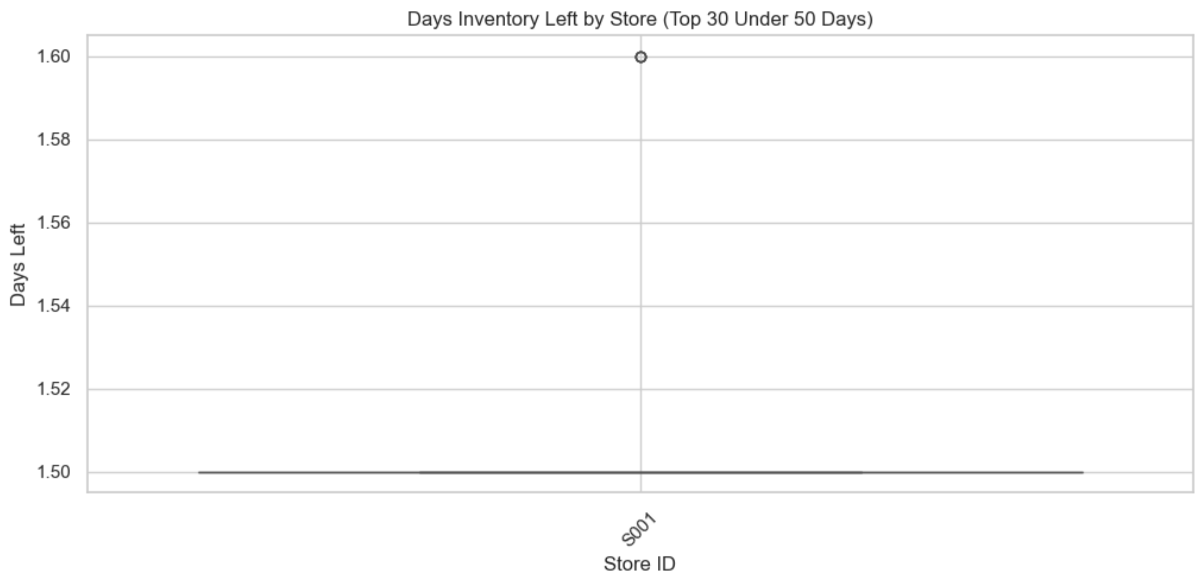
- **Sample Dashboard Chart:** We also produced interactive charts (e.g. newplot.png) as part of the inventory KPI dashboard. The example chart above plots store IDs against forecast error; other charts visualize inventory gaps, turnover rates, and seasonality. These visuals make the analysis tangible – decision makers can see, at a glance, which stores or products need attention based on the data.

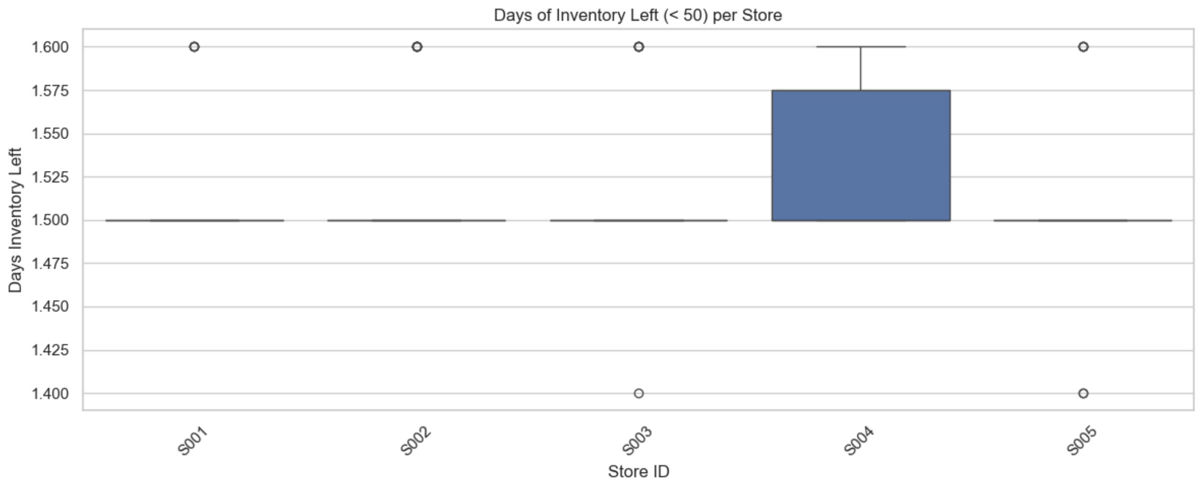
Dashboard diagrams:

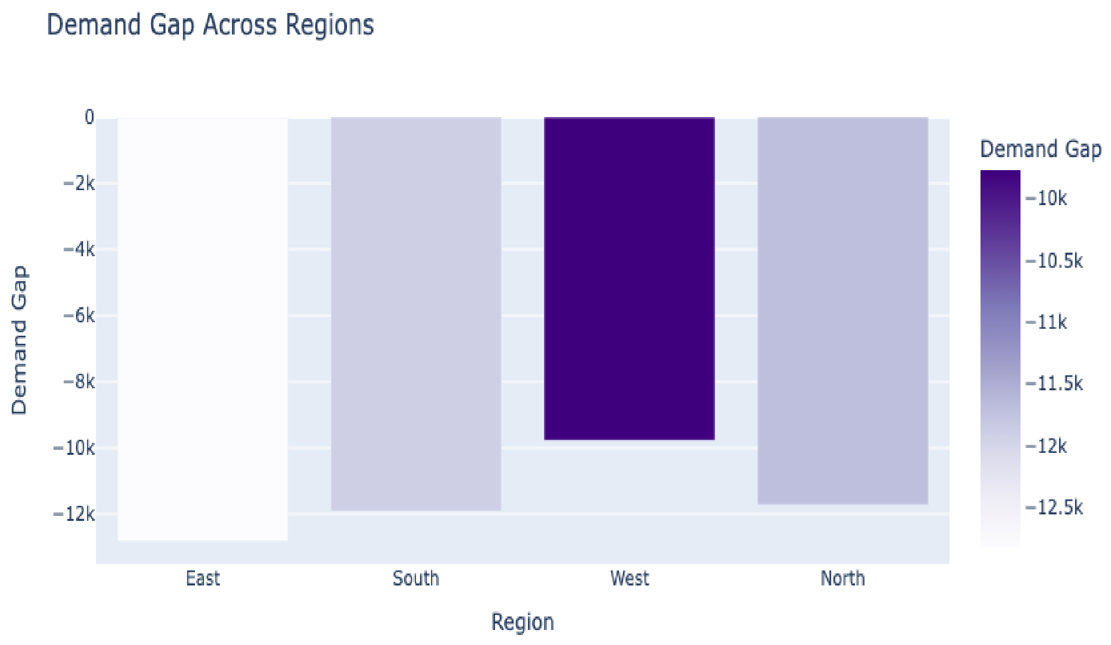


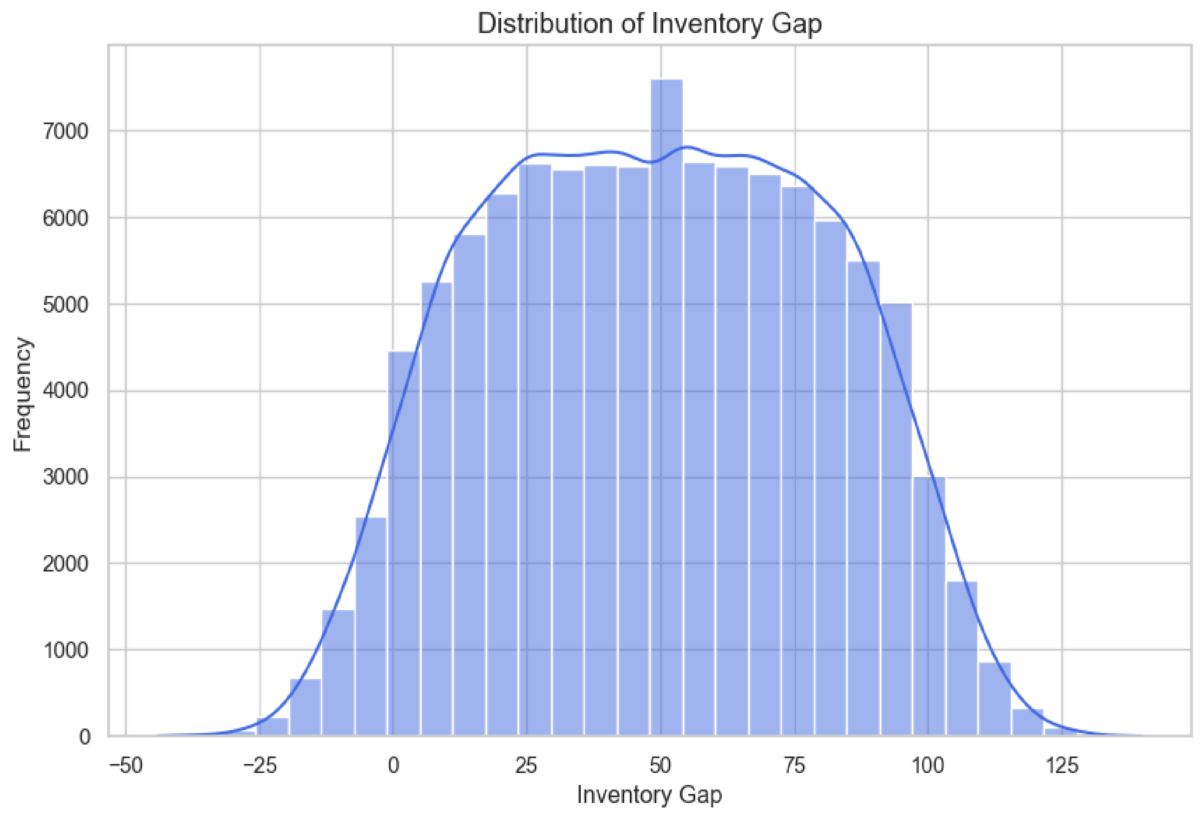


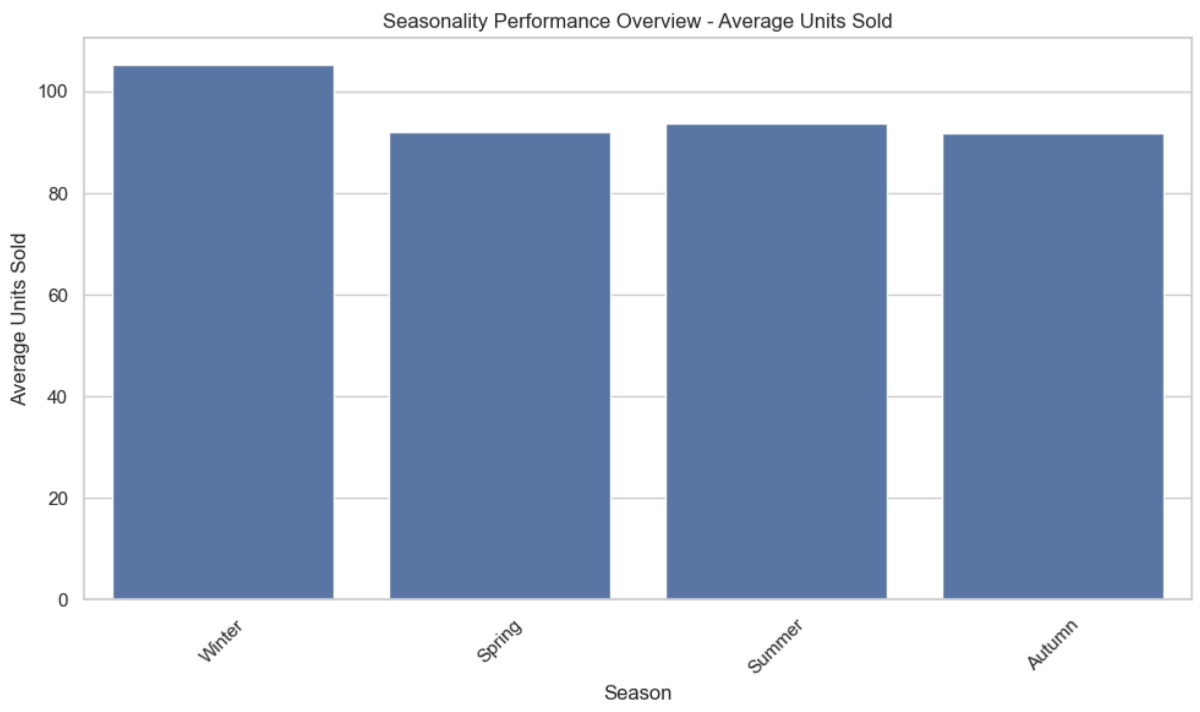

Average Inventory Gap by Store
Avg Inventory Gap
Store ID
S001
S002
S003
S004
S005
Avg Inventory Gap
40
38
36
34

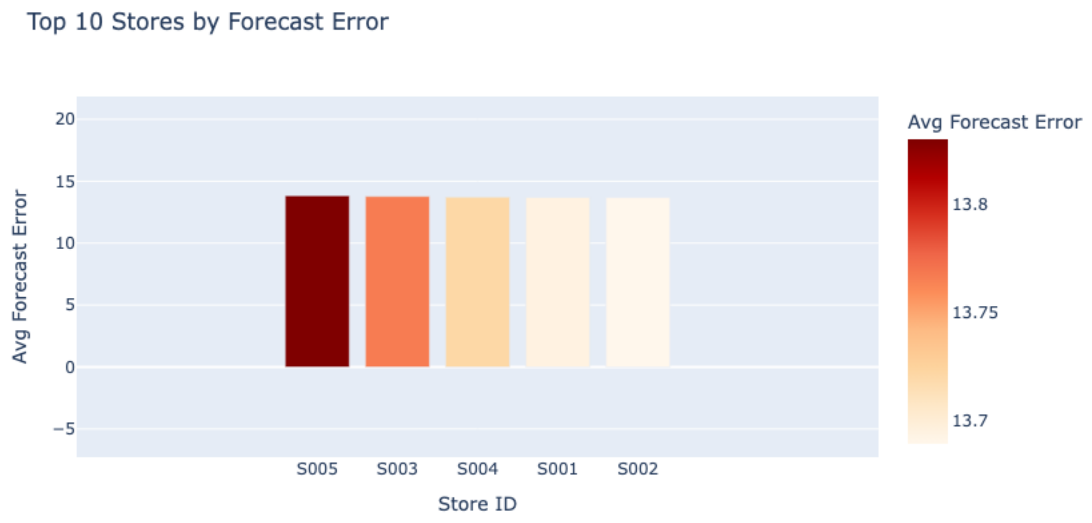

Days Inventory Left by Store (Top 30 Under 50 Days)
Days Left
Store ID
S001

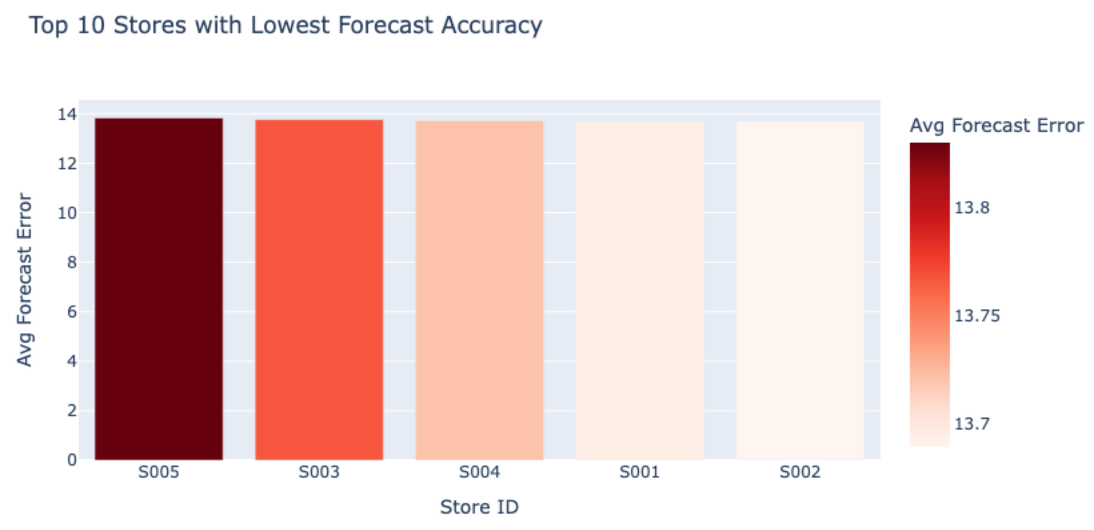

Days of Inventory Left (< 50) per Store
Days Inventory Left
Store ID
S001
S002
S003
S004
S005

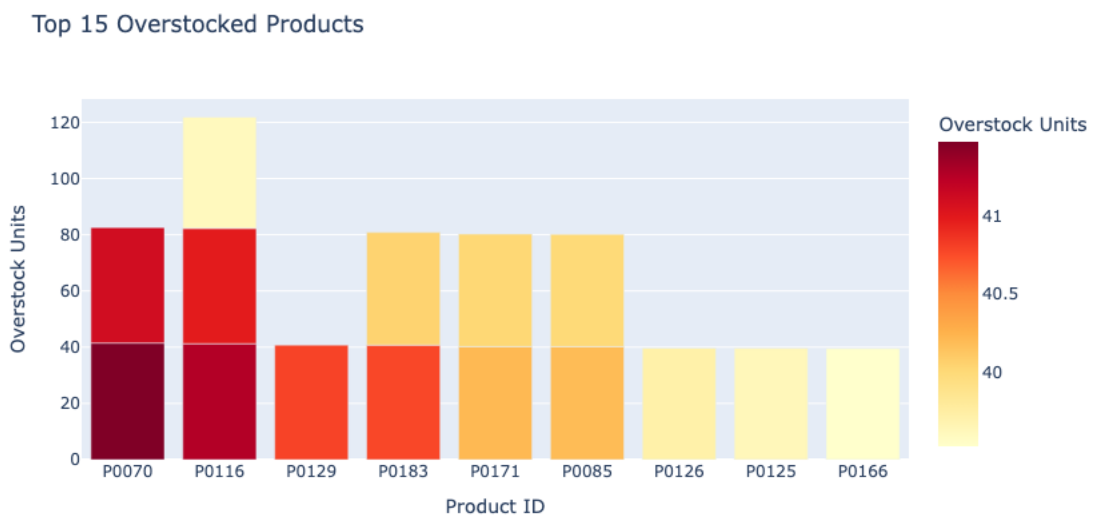

Demand Gap Across Regions
Demand Gap
Region
East
South
West
North
0
−2k
−4k
−6k
−8k
−10k
−12k
Demand Gap
−10k
−10.5k
−11k
−11.5k
−12k
−12.5k


Distribution of Inventory Gap
Frequency
Inventory Gap
7000
6000
5000
4000
3000
2000
1000
0
−50
−25
0
25
50
75
100
125


Seasonality Performance Overview - Average Units Sold
Average Units Sold
0
20
40
60
80
100
Winter
Spring
Summer
Autumn
Season


Top 10 Stores by Forecast Error
Avg Forecast Error
20
15
10
5
0
−5
S005
S003
S004
S001
S002
Store ID
Avg Forecast Error
13.8
13.75
13.7





Recommendations:

Our inventory analysis highlighted both under- and overstock issues across products and regions. To improve forecasting accuracy, Urban Retail should refine its demand models by incorporating seasonality, promotions, and other factors (e.g. holidays or regional trends). Advanced inventory tools and AI-driven forecasting can help here – as one source notes, "use demand forecasting to adjust inventory levels, plan reorder points, and align procurement" and modern AI tools can dynamically adjust stock levels based on real-time trends. In practice, this means regularly reviewing forecast errors (for example via heatmaps by category or store) and retraining models on recent sales data. Improving forecast precision will help reduce both stockouts and excess stock; indeed, accurate forecasting "reduces the expenses associated with storage" and mitigates obsolescence.

Another key recommendation is to adjust reorder thresholds using well-established formulas. For each SKU, Urban Retail should set reorder points based on lead time, demand rate, and a sensible safety stock cushion. In simple terms, **Reorder Point = (daily sales × lead time) + safety stock** . When inventory hits this point, an order should be placed (ideally automatically) so restocking occurs before running out. NetSuite explains that using reorder points "simplifies and streamlines" ordering and can even be automated to avoid manual delays . In our analysis, reorder point plots suggested some products were routinely running too low, so raising those thresholds will prevent future stockouts. Conversely, we identified slow-moving SKUs with excess days on hand; these should be trimmed via promotions or even delisted. A routine ABC or SKU rationalization exercise is advisable: focus on high-volume, high-profit items and scale back or eliminate low-value SKUs, as "not all products are created equal—some drive profits while others take up valuable space". Finally, collaboration with suppliers to shorten lead times (e.g. through just-in-time ordering) will allow lower safety stock and more responsive replenishment.

Summarizing the Recommendations:

- **Automate Reordering**: Use calculated reorder points to drive automated restocking workflows.
- **Reduce Forecast Error**: Improve demand forecasting models for stores with high forecast deviation.
- **Redistribute Inventory**: Shift inventory across regions based on demand-supply gaps.
- **Phase Out Slow Movers**: Consider phasing out overstocked, low-demand items.

- ## Expected Business Impact:

- Adopting these recommendations will have clear business benefits. Better forecasting and inventory alignment can dramatically reduce the costs of "inventory distortion" – the combined waste of stockouts and overstocks. Industry research shows that out-of-stocks alone are costing retailers on the order of $1.2 trillion per year in lost sales, with overstocks adding another ~$0.56 trillion in markdowns . By cutting forecast error and using data-driven reorder points, Urban Retail can capture more sales and avoid unnecessary clearance costs. In practice, this means fewer empty shelves and fewer piles of unsold goods. For customers, the impact is higher service levels: products will be available when needed, improving satisfaction and loyalty. For the company, it means increased revenue retention and reduced markdown expenses. NetSuite points out that anticipating demand "reduces wait times and backorders," building a reputation for reliability. Likewise, reduced excess stock frees up capital and lowers holding costs, improving cash flow.

- Overall, more efficient inventory management strengthens Urban Retail's bottom line. Lower carrying costs (from cutting excess inventory) and higher turnover of goods mean capital can be used more productively. Better stock alignment across regions and products also smooths supply chain operations, enabling faster replenishment and fewer emergency orders. As one study notes, retailers that improved inventory forecasting saw sizable reductions in inventory losses and improvement in margins . In short, fewer stockouts means higher sales and happier customers, while fewer overstocks means less waste and greater profit – a double win for Urban Retail's growth and efficiency.

Summarizing expected Business Impact:

With this solution in place, Urban Retail Co. can expect:

- ✅ **Smarter Inventory Decisions**: All decisions now stem from concrete data, not gut-feel.
- ✅ **Reduced Stockouts & Overstocks**: Better forecasting and reorder points help balance inventory.
- ✅ **Improved Supply Chain Efficiency**: Streamlined procurement cycles and fewer emergencies.
- ✅ **Enhanced Customer Satisfaction**: Customers find what they need, when they need it.
- ✅ **Boosted Profitability**: Lower holding costs and lost-sale incidents.

Conclusion:

In this project I analyzed Urban Retail Co.'s sales and inventory data to pinpoint mismatches in supply and demand. I created metrics and visualizations (e.g. overstock rankings, forecast accuracy heatmaps, and demand gap charts by region) to reveal where inventory could be optimized. Key insights included identifying the top overstocked products, stores with poor forecast accuracy, and SKUs with excessive on-hand days. By applying standard inventory formulas and prioritizing high-velocity items, I tied findings to clear business goals: better service and lower cost. The recommendations – from refining forecasts to recalibrating reorder triggers and rationalizing SKUs – directly target Urban Retail's objectives of minimizing lost sales and reducing waste.

In future, Urban Retail can build on this analysis by automating these processes. For example, integrating predictive analytics or AI-driven forecasting into the ordering system would continually adjust stock policies as demand changes. Automated dashboards could flag anomalies and trigger reviews, making inventory optimization a real-time practice rather than a one-time project. Ultimately, by embracing these data-driven inventory practices, Urban

Retail Co. will align its stock more closely with customer demand, improving operational efficiency and profitability in line with its strategic goals.