Final Report - Group 5

Problem Statement

ABC Foodmart is a grocery chain with two stores in Queens, NY. It has been in business for over 30 years.

In mid-2025, a new management team took over. They opened three new stores in Brooklyn. This was the largest expansion in the company's history. It created new challenges in sales, inventory, staffing, and vendor management.

In the past, the company stored data in spreadsheets and paper binders. The records covered staffing, payroll, inventory, vendors, purchase orders, sales, refunds, and expenses. The old system worked for two stores, but it became slow and inefficient after the expansion.

Main problems included:

- Data was scattered in different files and formats.
- Manual entry caused delays and mistakes.
- Reports took days to prepare.
- No real-time tracking of sales, stock, or expenses.
- No clear access levels for analysts and executives.

With five stores, the amount of data grew quickly. Management needed a new system to:

1. Store and organize all historical and new data in one place.
2. Update sales and inventory in real time.
3. Let analysts run SQL and Python/R queries.
4. Give executives dashboards that update automatically.

The company hired our team to design and build this database system. It must be reliable, easy to use, and ready to support future growth.

Proposal

We plan to build a new relational database for ABC Foodmart to replace the existing Excel and paper records, centralizing all business data in a single location. This will allow both old and new data to be stored and managed in a unified format.

The database will contain primary tables such as stores, employees, products, suppliers, purchase orders, inventory, sales, refunds, and operating expenses, as well as join tables to link different pieces

of information. We will design it according to Third Normal Form (3NF) to reduce data duplication and maintain data consistency.

To automate data updates, we will implement automated import processes. For example, inventory will be reduced in real time after a sale is completed; inventory will be increased after receiving goods from a supplier; inventory will be restored upon a refund; and inventory arrival dates will be automatically updated. The system will also check expiration dates to prevent expired items from being entered or sold.

In terms of analytics, we perform common queries as follows:

- Which store has the highest monthly revenue and profit?
- Which product categories are the most profitable?
- Daily sales and refund amounts
- Refund rate and primary refund reasons
- Orders fulfilled and item quantity by supplier
- Items below the replenishment threshold
- Items nearing expiration and remaining quantity
- Employee hourly sales compared to store averages
- Commonly purchased item combinations
- Monthly operating costs and highest expense types for each store

For management, we create interactive dashboards in Metabase, including: total revenue, net profit, monthly sales trends, category revenue share, a list of low-inventory and near-expiration items, expense breakdown, purchasing trends and expenditures, peak sales hours, and primary refund reasons. These charts automatically update as new data is imported.

This system enables the company to make decisions quickly (real-time reporting and dashboards), reduce manual entry errors, build
all data in one place for better organization, support future expansion such as adding more stores or products, and ensure data security based on user permissions.

This solution will make ABC Foodmart more efficient in its daily operations and provide a constant overview of business performance, providing reliable data support for future growth and planning.

Team Contract

All team members will work collaboratively on all phases of the project. Each member is responsible for completing a subset of the work while actively supporting group progress and peer review.

1. Database Schema Design and Data Collection

- Design the Entity-Relationship Diagram (at least 15 tables in 3NF)
- Find datasets to support the schema
- Data cleaning & sample data creation

2. Data Plan Implementation and SQL Queries
   - Build the PostgreSQL database schema (DDL)
   - ETL process (using Python to automate the process)
   - Write and test insight queries

3. Dashboard and Presentation
   - Design and prototype real-time dashboards
   - Prepare presentation materials for final submission

Data Description & Database Design

We use a portion of the "Grocery Inventory and Sales Dataset" from Kaggle as our base data to build the database. Specifically, we extract the product SKU, product name, product category, and unit price, providing a reference for what real-world grocery store products look like. For store locations, we reference the "NYS Retail Food Stores" dataset from Kaggle, focusing on addresses to model store locations in New York City.

Our database consists of 16 tables, organized into several categories to accurately model the processes of a grocery retail business while preserving data integrity, enabling efficient querying, and supporting both operational and analytical needs:

1. Reference Data: stores, product_category, products, employees, vendors, vendor_product, payment_method
2. Purchasing Data: purchase_order, purchase_product
3. Sales Data: transactions, sales_detailed_item, refund
4. Logistics Data: shifts, operating_expenses
5. Inventory Management Data: inventory, inventory_lot

The database schema was developed to model the complete operational, transactional, and logistical processes of a grocery retail business while maintaining strong data integrity, analytical flexibility, and performance efficiency. The design follows a normalized form (3NF) in which it separates the referencing data of stores, products, categories, and vendors from operational fact tables that record purchasing, inventory movements, sales, and refunds.

The reference data tables define the core entities on which all transactional data depends. The **stores table** contains the primary key store_id, store name, address details, and open_date. Using the primary key ensures that each store can be uniquely identified regardless of changes to address or name. The

product_category table stores category_id and category_name, with categories obtained from the Kaggle dataset. Separating categories into their own table avoids repetition and ensures the naming consistency. The **products table** uses the natural key product_sku as the primary key, along with product_name, foreign key category_id referencing product_category, and unit_price. Using the SKU as the key aligns with real-world retail operations, where SKUs are consistently used in procurement, point-of-sale, and inventory systems which reduces the need for additional joins when integrating data across processes. The **vendors table** stores supplier details with vendor_id as the primary key, while the **vendor_product table** models the many-to-many relationship between vendors and products, ensuring that a product can be supplied by multiple vendors and a vendor can supply multiple products. A uniqueness constraint on (vendor_id, product_sku) prevents duplication. The **employees table** stores staff records, with employee_id as the primary key and a foreign key to stores to identify their work location. The **payment_method table** holds the payment_method_id as primary key and method_name which provides a standardized reference for payment types used in transactions.

The purchasing data tables begin with the **purchase_order table**, which holds order-level data including purchase_order_id (primary key), vendor_id, store_id, order_date, and status. This is linked to **the purchase_product table**, which stores each order's line items with purchase_product_id (primary key), purchase_order_id (foreign key), vendor_product_id (foreign key), quantity, and unit_cost. Check constraints enforce that quantities and unit costs are positive, preventing invalid purchasing data. Separating order into detailed items follows standard database design practices for procurement systems and allows detailed cost tracking and vendor performance measurement.

The sales and refund data tables capture retail transactions at two levels of granularity. The **transactions table** stores the transaction header, with transaction_id (primary key), store_id, employee_id, transaction_date, and payment_method_id. The **sales_detailed_item table** stores each sold product line, with detailed_id (primary key), transaction_id (foreign key), product_sku (foreign key), lot_id (foreign key), quantity, actual_unit_price, and discount_amount. Linking sales to specific lot IDs allows precise margin analysis based on the cost of the actual batch sold, as well as inventory tracking for expiration dates. The **refund table** records product returns, with refund_id (primary key), transaction_id, detailed_id, quantity, and amount. Linking refunds directly to the original sales line ensures accuracy and prevents duplicate or fraudulent refunds.

The store operations data tables support labor and cost management. The **shifts table** records employee work periods with shift_id (primary key), employee_id, store_id, start_time, and end_time. This enables calculation of labor productivity. The **operating_expenses table** contains expense_id (primary key), store_id, expense_date, expense_type, and amount, capturing non-COGS costs for profitability analysis and budgeting.

The inventory data tables consist of two tables serving complementary purposes. The **inventory_lot table** tracks stock at the batch level, with lot_id (primary key), store_id, purchase_product_id, received_quantity, and an optional expiration_date. This enables First-Expired–First-Out inventory

rotation, recall traceability, and lot-specific costing, which are essential for managing expiration dates of the products. The **inventory table** provides a summarized view of current stock at each store–product combination, using a composite primary key (store_id, product_sku) to prevent duplicate rows and ensure efficient lookups.

Several triggers are implemented to enforce business rules and maintain data integrity. Two helper functions, sku_for_pp and inv_upsert, centralize SKU from purchase records and updates to the inventory snapshot. The AFTER triggers, on sales_detailed_item decrease the inventory stock after selling, on inventory_lot increase the inventory stock after purchasing from vendors, on refunds restore the inventory stock after consumer returning products. This helps ensure real-time synchronization to inventory levels. The BEFORE trigger on sales_detailed_item prevents overselling by comparing the requested quantity against available lot balance (received − sold + refunded) and raising an exception if there is insufficient.

ETL Process

The ETL pipeline follows a sequential process that reflects the operational workflow of the retail grocery business. Data is extracted from source datasets or generated synthetically. And then transformed to match the schema's constraints in which column names were normalized and missing values were handled. Finally loaded into the PostgreSQL database via SQLAlchemy in the order that preserves referential integrity.

Reference tables including stores, product_category, products, vendors, vendor_product, employees, and payment_method, are populated before any transactional tables to ensure that all foreign key dependencies are met. These are derived directly from cleaned source datasets or generated using the Faker library in python with random seed set to ensure future replications.

The purchase_order table is populated first in the transactional chain. Each record represents a vendor's supply request from a specific store, containing the vendor_id, store_id, order date, and order status. Dates are assigned over a defined time range to simulate continuous procurement activity, and statuses are distributed across realistic states of completed and pending. During the data generation, this table follows the logic that all vendors are included, each supplying multiple stores, and stores could place multiple orders from the same vendor.

The purchase_product table details the contents of each purchase order. For every purchase_order_id, a selection of products supplied by the corresponding vendor is sampled. Each row specifies a vendor_product_id, the ordered quantity varies by category to reflect real-world ordering patterns, and the actual unit cost, calculated using mark up rate for each category with selling price divided by the one plus the mark up rate.

The inventory_lot table records the receipt of goods into stock. Each purchase product can generate one or more lot entries with the same expiration date. The table includes lot_id, store_id, purchase_product_id, and quantity specified. Receipt dates are derived from the purchase order date, and expiration dates are assigned based on product category shelf life, for example, shorter for bakery products, longer for canned items. These lots help track inventory movements and ensure the rule of First-Expired–First-Out.

The transactions table contains point-of-sale records, each representing a completed customer checkout. Each record associates the sale with a store_id, employee_id of the cashier, a transaction timestamp, and a payment method. Transactions are distributed throughout business hours, and payment methods reflect a realistic mix of cash, card, and mobile payment with setted probability.

The sales_detailed_item table records each transaction at the product level. Each record is linked to a specific lot_id to maintain traceability from sale back to purchase and supplier. The table includes quantity sold, actual_unit_price, and any discount_amount applied. Quantities and discounts are generated with category-specific probabilities, and lots are selected according to the principles to ensure that the oldest stock is sold first.

The refunds table captures returned merchandise. Each refund references a specific detailed_id from the sales detail table, ensuring direct link between the refund and the original sale. Refund dates are restricted to occur within a defined window of 30 days after the sale, and quantities cannot exceed the amount sold. Amounts are calculated based on the net price at the time of sale.

This ETL process preserves referential completeness by loading tables in a dependency-respecting sequence and by generating keys in a controlled manner. Each transactional record is tied back to its reference data through explicit foreign key relationships to ensure the data accuracy and integrity.

Analytical Procedures & Insights

One of the main goals of this project was to equip ABC Foodmart with the ability to turn its sales, inventory, and operations data into clear, actionable insights. Using the PostgreSQL database we built, we created and tested ten advanced analytical procedures that answer the most important business questions raised by management and store teams. These procedures combine data from multiple tables, apply detailed calculations, and use filtering and ranking logic that goes far beyond what could be done in spreadsheets. Together, they demonstrate how a well-structured relational database can support better decision-making, improve day-to-day operations, and help the company grow profitably.

1. Monthly Revenue and Profit by each store

   This analysis calculates both gross revenue and net profit for each store, aggregated by calendar month. We joined sales_detailed_item to transactions to obtain the sales quantity and unit price,

then linked sales_detailed_item to inventory_lot and subsequently to purchase_product to retrieve the actual unit cost for each sold batch. Profit was calculated by subtracting both the unit cost and any applicable discount from the selling price, multiplied by the quantity sold.

The results identified the top-performing store each month for both revenue and profit. For example, in January 2025, 101 AVE CONVENIENCE STO led in both revenue and profit. In June 2025, 1066 RUTLAND DELI achieved the highest monthly profit, while in March 2025, 1060 GREENE DELI topped both metrics. These patterns allow management to recognize high-performing locations each month and investigate the operational strategies contributing to their success.

2. Product Categories that generate the most revenue and profit

   We evaluated contribution by product category using batch-accurate costs and line-level discounts. Sales lines from sales_detailed_item were joined to products and their product_category for revenue, and then linked through inventory_lot to purchase_product to retrieve the actual unit cost of the specific lot sold. Profit was computed per line as quantity × (unit price − unit cost − discount), aggregated to the category level. This analysis surfaces volume vs. margin dynamics that are not visible in raw sales totals. In our data, Fruits & Vegetables is the top revenue driver ($482.6k) but ranks second on profit, while Beverages delivers the highest profit ($124.7k) despite lower revenue than produce. Seafood ranks third for both revenue and profit, and Bakery trails on both measures. These findings point to different strategies for each category: keep the high sales of Fruits & Vegetables by ensuring products are always in stock and reducing spoilage, grow Beverages further by offering bulk packs and placing them in high-visibility areas, and improve Bakery performance by reviewing suppliers, adjusting prices, and reducing waste.

3. Total Daily Sales and Refund Amounts Over Time

   This analysis tracks daily sales, refunds, and net sales over time to identify short-term performance shifts and anomalies. We calculated daily sales from sales_detailed_item by multiplying quantity by the actual unit price and subtracting any per-unit discount. These sales lines were joined to transactions to extract the transaction date. Daily refunds were computed from the refunds table, joined to sales_detailed_item and transactions to ensure accurate linkage to the original sale date. The two datasets were then joined on date, and net sales were derived as daily sales minus daily refunds.

   The resulting time series provides a clear view of both revenue and refund activity for each day between January 1, 2025, and July 31, 2025. Most days show modest refund levels relative to sales, but occasional spikes such as July 27, 30, and 31, 2025 indicate operational or quality issues that warrant further investigation. This procedure allows managers to quickly pinpoint problematic days, correlate them with product batches or store events, and implement

corrective measures. Over time, this analysis helps track whether such interventions reduce refund volumes and improve net revenue consistency.

4. Refund rate and amount by product and the top refund reason

   We evaluated refund exposure in two ways: by individual product and by reason for return. At the product level, we used sales_detailed_item to calculate the total units sold and sales value for each SKU. We then joined refunds back to the exact sold line through detailed_id to determine the total refunded units and refunded dollar amounts. By linking these results to the products table, we were able to present both the product name and SKU alongside a calculated refund rate, defined as refunded units divided by total units sold. This method ensures that every refund is tied to the exact product sold rather than aggregated into broad categories.

   The analysis revealed products with both high refund rates and high refund dollar losses. For example, Avocado Oil had a refund rate of 12.80%, which may point to a product quality or suitability issue, while Banana generated the largest dollar loss from refunds ($3,312.56) despite a lower rate of 10.76%, due to its high sales volume. Based on these results, managers can set priorities by focusing on high-rate items for quality and supplier reviews, while addressing high-dollar-loss products through improvements in handling, packaging, and pricing strategies.

   In addition to the product-level view, we also analyzed the top refund reasons by aggregating refund counts and total refund amounts from the reason field in the refunds table. The most common reason was Changed mind, with 2,367 refunds totaling $26,433.30. This was followed by Defective items (2,082 refunds totaling $22,618.14), Quality/Expired issues (1,615 refunds totaling $18,335.96), and Wrong item returns (1,189 refunds totaling $13,293.42). The “Other” category accounted for 825 refunds totaling $8,657.00.

   These insights make it possible to address the root causes of returns more effectively. High counts for “Defective” and “Quality/Expired” highlight areas for improvement in supplier quality control and in-store storage practices, while “Wrong item” suggests that process changes are needed at checkout or during online order fulfillment. Although “Changed mind” is often linked to customer behavior, clearer product descriptions and better merchandising could help reduce its occurrence. By monitoring both product-level refund metrics and reason-level trends over time, ABC Foodmart can measure the impact of corrective actions and reduce the overall financial burden of refunds.

5. Vendor Performance and Order Fulfillment

   We measure each vendor’s contribution to ABC Foodmart’s operations by evaluating both the range of products they supply and the number of completed orders they fulfill. Using purchase order records filtered to “completed” status, we linked vendors to their supplied SKUs and purchase orders. By counting distinct products and distinct fulfilled orders for each vendor, the

results highlight suppliers that provide a broad assortment, consistently meet order requirements, or excel in both areas.

In the current period, several vendors stand out for both order volume and product range. Edwards Ltd completed 112 orders covering 180 unique SKUs, while Matthews, Howard and King fulfilled 103 orders with an even broader assortment of 229 SKUs. Duran, Bass and Patton and Morton Group also performed well, completing 88 and 87 orders respectively. These insights enable the procurement team to focus on high-volume, broad-assortment suppliers during peak periods, negotiate favorable terms or rebates with top performers, and develop improvement or secondary sourcing strategies for vendors with limited assortments or lower fulfillment activity.

6. Inventory Below Reorder Points (Low-stock)

   This query identifies products that have dropped below their defined reorder thresholds. By comparing the quantity_in_stock against each item's reorder_threshold in the inventory table, we isolate SKUs at risk of stockouts for each store. In the current run, the output is empty which indicates no items are presently below their reorder point. When populated, this procedure is a key operational safeguard as it enables store managers and the procurement team to trigger timely replenishment orders, maintain product availability, and avoid lost sales.

7. Near Expiration Products in Inventory and Remaining Quantity

   We monitor inventory risk by identifying products that are set to expire within the current month and calculating the remaining sellable quantity for each lot. Using inventory lot records, we determine remaining stock by subtracting units sold from the original lot quantity and adding back any refunded units. By linking each lot through purchase and vendor product records, we ensure accurate SKU-level reporting across all stores.

   In the current period, 320 lots across multiple stores are approaching expiration, with remaining quantities ranging from fewer than 100 units to more than 300 units. This visibility enables store managers to take targeted action such as applying First-Expired, First-Out (FEFO) rotation, running short-term promotions, or adjusting shelf placement to clear at-risk stock before it expires. For the procurement team, the results provide early warning signals for seasonal overstock or slow-moving SKUs, helping to refine purchasing decisions and reduce future shrink.

8. Employee Revenue per Labor Hour each store vs. Store Average

   We evaluate individual performance by calculating monthly revenue per labor hour for each cashier and comparing it to their store's average. Using transaction records, we derive monthly revenue per employee, net of line-level discounts. We estimate labor hours by multiplying each employee's typical shift length (from shifts) by the number of distinct days they processed

transactions that month. We then compute each employee's revenue per hour and benchmark it against the store-level average for the same month, surfacing the top performer per month.

In the current period, several employees consistently exceed their store averages. For example, Nancy Miller at 1060 GREENE DELI posted $113.93/hour in January against a store average of $104.13. Sean Peters at 100-17 BEACH CHANNEL DR achieved $118.46/hour in February vs. $105.26, and Nicole Patterson at 1064 FUEL led both May ($111.27/hour) and July ($114.70/hour) relative to store benchmarks. These results will help managers identify coaching opportunities and staffing patterns that lift productivity. For instance, managers can schedule high performers during peak windows, pairing them with trainees, and replicating effective checkout practices across shifts.

9. Most Frequently Bought Together Items (In the same transaction)

   We identify products that are most frequently purchased together by analyzing transaction-level co-occurrences. First, each transaction is reduced to a distinct set of SKUs to avoid double-counting multiple units of the same item. We then generate all unique product pairs from these transactions and count the number of transactions in which each pair appears. Finally, we join the results to the products table to display readable product names alongside their pair counts.

   In the current period, the strongest product pairing signals come from combinations like Mushrooms with Bread Flour, Bread Flour with Powdered Sugar, and Blueberries with Anchovies, each appearing together in five separate transactions. Other notable pairings include Black Tea with Kale, Whole Wheat Bread with Sardines, and Olive Oil with Black Coffee also with five paired purchases each. These insights can guide in-store merchandising, endcap displays, and cross-promotional offers. For example, displaying Mushrooms and Bread Flour together, or bundling Olive Oil with Black Coffee in promotions, could capture natural purchase synergies and increase basket value.

10. Monthly Operating Expenses per store and The Highest Expense Type

    We track each store's monthly operating expenses by aggregating costs across all expense categories and identifying the highest expense type for each month. Using the operating_expenses table, monthly totals are calculated by summing expenses per store and month, while expense types are ranked by amount to isolate the largest contributor to operating costs.

    In the current period, rent consistently emerges as the top expense category for all stores, often representing the majority of monthly outflows. For example, in January, 1060 GREENE DELI incurred a total of $9,088.34 in expenses, with rent accounting for $7,811.62. Similarly, 1066 RUTLAND DELI posted $8,633.72 in January expenses, of which $7,538.16 was rent. This pattern persists throughout the year, highlighting rent as a stable, high-impact cost driver.

These insights will help store and finance managers in budget planning and cost control. By recognizing rent as the predominant expense, managers can better forecast fixed costs, evaluate lease agreements, and focus on optimizing controllable expense categories to improve profitability.

Customer Interaction & Performance Optimization

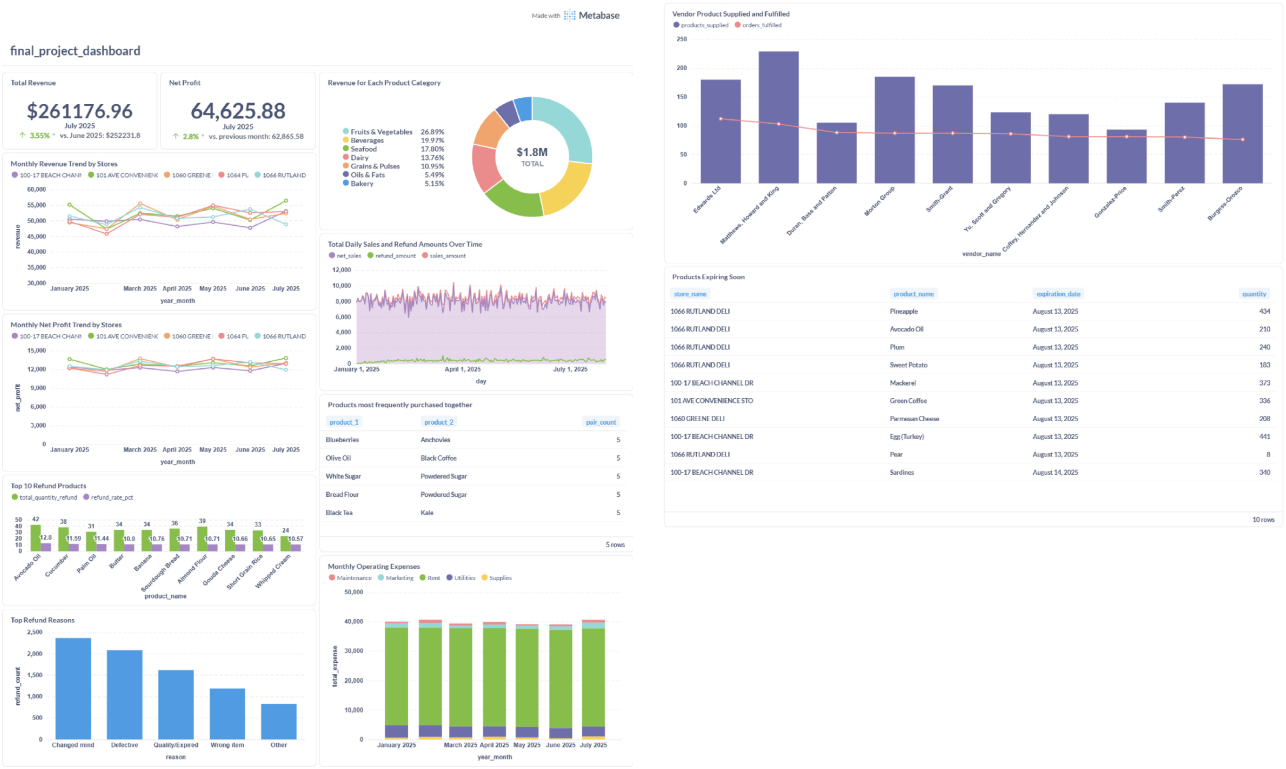
The database system for ABC Foodmart is designed to support both technical analysts and executive decision-makers, ensuring tailored access and actionable insights for each group. Analysts are provided direct access to the PostgreSQL database and Metabase. This allows them to run SQL queries and explore the data deeply. They could utilize pre-built queries, views, and parameterized dashboards to perform advanced analytics. This direct access accelerates data-driven decision-making and promotes flexibility in investigating emerging business questions. Executives receive curated and easy-to-understand reports and dashboards focusing on high-level KPIs, trends, and summaries of the stores. These dashboards highlight key business metrics such as revenue, profit, sales growth, and refund rates with visualizations optimized for quick understanding. In addition, the future step will be automating report delivery via scheduled emails to ensure timely information flow. Alerts are configured to notify executives of critical issues which allows immediate responses without requiring SQL knowledge or direct database access.

The integration of SQL and Python in the ETL and analytical pipeline brings multiple benefits. It helps reduce manual intervention and errors to ensure consistent data updates. Besides this, it also facilitates code version control and modular development, improving long-term system evolution. Non-technical personnel could interact with the system exclusively via interactive dashboards and reports which provides a user-friendly experience without exposure to raw data or query writing.

For optimal performance, we employ indexing on frequently queried columns, especially foreign keys and columns used in filter conditions, to accelerate data retrieval. Coupled with query optimization techniques, such as rewriting complex queries and using materialized views for expensive aggregations, these measures reduce latency and improve the user experience for both analysts and executives.

Regarding deployment, we recommend hosting the database in the cloud on managed services like AWS RDS, Microsoft Azure Database, or Google Cloud SQL. Cloud platforms provide significant advantages, including effortless scalability to handle growing data volumes and concurrent users, built-in security and compliance features.

Dashboard Showcase



The Metabase dashboard created for ABC Foodmart is a comprehensive tool designed to provide actionable insights across sales, inventory, refunds, vendor performance, and operational expenses.

The KPI metric cards display total revenue and net profit, alongside percentage comparisons to the previous month. This immediate snapshot helps executives quickly assess financial trends and identify any significant changes in performance. The dashboard also features a revenue breakdown by product category to highlight which categories drive sales and profitability which allows targeted merchandising and stocking strategies. The line charts present the trend analysis of monthly revenue and net profit. It illustrates performance dynamics over time to support forecasting and strategic planning. The dashboard also includes a time series of daily sales and refund amounts, enabling managers to detect irregularities or operational challenges on a more granular level. A detailed analysis of the top ten products by refund quantity and rate highlights potential quality or process issues requiring attention. Additionally, a market basket analysis of products frequently purchased together offers actionable insights for merchandising and cross-selling opportunities. The bar chart of top refund reasons presents customer behavior and product issues, allowing the stores to address root causes effectively. Operational expenses are broken down monthly by type, facilitating improved budget management and cost control. Vendor performance metrics provide visibility into supplier reliability and product range to assist procurement decisions. The table of products nearing expiration supports inventory managers in minimizing waste and optimizing stock rotation.

Together, this dashboard empowers executives with high-level overviews while equipping analysts and store managers with detailed and actionable data.

Conclusion

The goal of this project is to design and implement a robust relational database management system and an end-to-end ETL pipeline to empower ABC Foodmart with reliable, accurate, and actionable data. By structuring complex sales, inventory, and operational data into a well-designed PostgreSQL database, we established a solid foundation for efficient data management and advanced analytics.

The ETL process ensured that data was cleanly extracted, transformed to meet schema requirements, and loaded in a manner that preserved referential integrity and data quality. This seamless pipeline supports continuous data updates and reliable reporting. In addition, through designed analytical procedures, ABC Foodmart gained deep insights into revenue and profit drivers, inventory health, refund patterns, vendor performance, and employee productivity. These insights allow data-driven decision-making, targeted operational improvements, and strategic planning that were previously difficult with fragmented data or spreadsheet-based analysis.

In conclusion, the relational database management system provided significant benefits, including data consistency, integrity enforcement through relationships, scalability for growing data volumes, and flexible querying capabilities. The combination of ETL automation and powerful SQL analytics increased efficiency and reduced errors, allowing ABC Foodmart to respond rapidly to market changes and customer behaviors. Overall, the implementation delivered a comprehensive and user-friendly data infrastructure that supports both high-level executive dashboards and detailed analytical exploration.

Sample Data:

1. Stores

	store_id	store_name	address	city	state	zip	open_date
0	1	100-17 BEACH CHANNEL DR	100-17 BEACH CHANNEL DR	ROCKAWAY BEACH	NY	11694	1993-12-09
1	2	101 AVE CONVENIENCE STO	77-02 101ST AVE	OZONE PARK	NY	11416	1993-07-14
2	3	1060 GREENE DELI	1060 GREENE AVE	BROOKLYN	NY	11221	2025-05-03
3	4	1064 FUEL	1064 ATLANTIC AVE	BROOKLYN	NY	11238	2025-04-30
4	5	1066 RUTLAND DELI	1066 RUTLAND ROAD	BROOKLYN	NY	11212	2025-06-16

2. Product Categories

	category_id	category_name
0	1	Grains & Pulses
1	2	Beverages
2	3	Fruits & Vegetables
3	4	Oils & Fats
4	5	Dairy
5	6	Bakery
6	7	Seafood

3. Products

	product_sku	product_name	category_id
0	29-205-1132	Sushi Rice	1
1	40-681-9981	Arabica Coffee	2
2	06-955-3428	Black Rice	1
3	71-594-6552	Long Grain Rice	1
4	57-437-1828	Plum	3
...	...	...	...
985	82-977-7752	Spinach	3
986	62-393-9939	Cheddar Cheese	5
987	31-745-6850	Cabbage	3
988	86-692-2312	Avocado Oil	4
989	28-044-4102	Papaya	3

4. Employees

	employee_id	first_name	last_name	store_id	phone	email	salary	start_date	employment_status	job_title
0	1	Phillip	Molina	1	001-795-954-8054x8536	phillip.molina@abcfoodmart.com	36678.81	2017-11-20	Part-time	Stocker
1	2	Dana	Jackson	1	001-899-399-4324	dana.jackson@abcfoodmart.com	35596.32	2019-06-20	Part-time	Cleaner
2	3	Sean	Peters	1	(928)309-9837x0335	sean.peters@abcfoodmart.com	37683.33	2024-05-19	Part-time	Cashier
3	4	Henry	Pennington	1	001-638-578-8481x774	henry.pennington@abcfoodmart.com	36795.52	2018-08-01	On Leave	Security Guard
4	5	Lindsey	Kline	1	(712)394-3713x0998	lindsey.kline@abcfoodmart.com	37229.61	2016-11-21	Full-time	Stocker

5. Shifts

	shift_id	employee_id	store_id	schedule_start	schedule_end
0	1	1	1	08:00:00	16:00:00
1	2	2	1	08:00:00	16:00:00
2	3	3	1	08:00:00	16:00:00
3	4	4	1	08:00:00	16:00:00
4	5	5	1	08:00:00	16:00:00

6. Vendors

	vendor_id	vendor_name	vendor_phone	vendor_email
0	1	Smith-Grant	001-385-219-1524	huffmanbeth@knox.com
1	2	Matthews, Howard and King	432-545-2314	leonbeth@montes.com
2	3	Edwards Ltd	(430)278-0625	eugene94@campbell.com
3	4	Yu, Scott and Gregory	921.761.6940x21596	nwilson@willis.org
4	5	Gonzalez-Price	(528)649-5534x652	renee06@mccann.net
5	6	Harrison-Anderson	899.407.8922	juliemorton@wilson.com

7. Vendor Product

	vendor_product_id	vendor_id	product_sku
0	1	1	83-556-0996
1	2	1	80-441-7249
2	3	1	02-655-3240
3	4	1	84-624-0201
4	5	1	06-340-6856

8. Purchase Orders

	purchase_order_id	vendor_id	store_id	order_date	status
0	1	1	1	2025-01-04	Completed
1	2	1	1	2025-07-12	Completed
2	3	1	1	2025-05-23	Completed
3	4	1	1	2025-07-20	Completed

9. Purchase Products

	purchase_product_id	purchase_order_id	vendor_product_id	quantity_purchased	actual_unit_cost
0	1	1458	2123	303	3.21
1	2	1708	2654	500	12.50
2	3	1623	2487	391	4.29
3	4	751	1185	432	1.07

10. Inventory Lots

	lot_id	store_id	purchase_product_id	quantity	expiration_date	received_date
0	1	1	1	166	2025-11-02	2025-02-20
1	2	1	1	137	2025-11-11	2025-02-20
2	3	1	2	500	2026-01-15	2025-01-15

11. Inventory

	store_id [PK] integer	product_sku [PK] character varying (100)	quantity_in_stock integer	reorder_threshold integer
1	1	00-119-8780	380	35
2	1	00-215-7434	325	20
3	1	00-357-2313	219	20
4	1	00-366-9496	319	50
5	1	00-405-7428	390	50
6	1	00-440-9568	291	50

12. Payment Methods

	payment_method_id	method_name
0	1	Cash
1	2	Credit Card
2	3	Debit Card

13. Transactions

	transaction_id	store_id	employee_id	tran_time	payment_method_id
0	1	1	46	2025-02-07	2
1	2	1	24	2025-02-14	1
2	3	3	14	2025-03-31	4
3	4	2	3	2025-01-29	3

14. Sales Detailed Items

	detailed_id	transaction_id	product_sku	lot_id	quantity	actual_unit_price	discount_amount
0	1	1	42-175-1648	1279	2	6.00	0.83
1	2	1	80-371-3695	703	2	5.00	0.00
2	3	1	38-664-2155	1013	1	2.55	0.00
3	4	2	56-191-6497	436	2	1.50	0.23
4	5	3	72-810-9753	4632	2	6.50	0.00

15. Refunds

	refund_id	original_tran_id	refund_time	employee_id	reason	detailed_id	quantity	amount
0	1	11936	2025-07-23	40	Wrong item	32162	1.0	4.00
1	2	49076	2025-07-13	18	Changed mind	132124	2.0	30.00
2	3	16731	2025-02-23	47	Changed mind	45009	1.0	4.50
3	4	3080	2025-06-08	46	Defective	8220	1.0	4.74

16. Operating Expenses

	expense_id	store_id	expense_type	amount	expense_date
0	1	1	Rent	6679.50	2025-01-01
1	2	1	Utilities	664.34	2025-01-01
2	3	1	Rent	6679.50	2025-02-01
3	4	1	Utilities	771.59	2025-02-01
4	5	1	Rent	6679.50	2025-03-01

Reference

- Reference data: Grocery Inventory and Sales Dataset: https://www.kaggle.com/datasets/salahuddinahmedshuvo/grocery-inventory-and-sales-dataset
- Reference data: NYS Retail Food Stores: https://www.kaggle.com/datasets/new-york-state/nys-retail-food-stores?select=NYSDAM_RetailFoodStoresEstablishmentTypeCodes.pdf
- Link to full dataset: https://github.com/chenyifan20010324-pixel/APAN-5310-Final-Project-/tree/main/data
- Link to database design SQL code: https://github.com/chenyifan20010324-pixel/APAN-5310-Final-Project-/blob/main/table_schema.sql
- Link to ER Diagram: https://github.com/chenyifan20010324-pixel/APAN-5310-Final-Project-/blob/main/ER_Diagram.pdf
- Link to Lucidchart: https://lucid.app/lucidchart/cb032c61-17c8-4170-8384-4e7049fbbbe1/edit?beaconFlowId=4ECD4C79F0BBE82C&invitationId=inv_ebd50dd0-b3f4-47e8-8676-3a8be76bb0ba&page=0_0#
- Link to ETL process: https://github.com/chenyifan20010324-pixel/APAN-5310-Final-Project-/blob/main/data_generation.ipynb
- Link to analytical queries: https://github.com/chenyifan20010324-pixel/APAN-5310-Final-Project-/blob/main/project_queries.ipynb
- Link to dashboard: https://github.com/chenyifan20010324-pixel/APAN-5310-Final-Project-/blob/main/Metabase%20-dashboard.pdf