ABC Foodmart Project Final Report - Group 7

GitHub Link to all documents: https://github.com/7effreyzzZ/SQL-Final-Project

Proposal

ABC Foodmart, a well-established neighborhood grocery chain with two locations in Queens, NY, has achieved steady growth over the past 30 years. However, as industries increasingly adopt digital solutions to streamline operations, ABC Foodmart has yet to modernize its data management processes. The company continues to rely on an outdated system of spreadsheets and paper binders to manage critical business data, including staffing, inventory, vendors, deliveries, sales, and accounting. While this approach was manageable for two locations, it has led to inefficiencies, errors, and slow decision-making.

With plans to expand into Brooklyn, NY, by opening three additional locations, the limitations of the current system have become a significant obstacle. The increased scale and complexity of operations demand a modern, efficient, and scalable solution to manage data effectively.

To address these challenges, ABC Foodmart has engaged our team to design and implement a relational database system. This system will modernize data management, enhance decision-making capabilities, and improve operational efficiency. The solution will include interactive dashboards and data visualizations, offering insights such as store profitability and product popularity to support business growth. The ultimate goal is to provide a robust foundation for seamless data management and operational success as ABC Foodmart scales its operations across multiple locations.

To address the challenges faced by ABC Foodmart, we propose the following solution:

Relational Database Design

Our team will design a normalized relational database with 18 tables in Third Normal Form (3NF). Additionally, an ER-diagram will also be generated according to the relational database schema. This structure will effectively organize and manage all critical aspects of the business, including inventory, staffing, vendor information, deliveries, sales transactions, and accounting. By eliminating redundancy and ensuring data integrity, the database will enable complex querying and reporting to support better decision-making and streamlined operations. Main Analytical tools at this phase are SQL and Lucidchart.

Data Migration

A critical step in this project will be the migration of existing data from spreadsheets into the new database system. We will perform thorough data validation and cleanup during this process to ensure accuracy and consistency. We then will begin the ETL process to enter the data into the relational database we designed in the previous phase. This integration will create a centralized repository for all business data, eliminating

inefficiencies and errors caused by scattered or outdated information. Main analytical tools for this phase are Python and SQL.

Analysis and Support

After completing the previous steps, our team will conduct business analysis to provide business insights for ABC Foodmart, including but not limited to the trends of profitability of each product in each store and average customer spending for each location. Additionally, our team will offer ongoing technical support during and after the implementation phase to address any challenges and ensure long-term success.

This comprehensive solution will modernize ABC Foodmart's data management, improve decision-making capabilities, and support the company's expansion into new markets.

Team Structure

Miaoer Mo - Research, Data extraction & preparation, Database Schema Design, ETL, Report

Isabella Shen - Research, Data extraction & preparation, Database Schema Design, Report

Yifan Lei - Research, Data extraction & preparation, Database Schema Design, Report

Yifei Wan - Research, Data extraction & preparation, Database Schema Design, Report

Zijian Li - Research, Data extraction & preparation, Database Schema Design, Report

Rui Yan - Research, Data extraction & preparation, Database Schema Design, Report

Timeline

Week 8 - Rough plan of actions about the project, initial phase research behind decision making

Week 9 - Develop team contract, brainstorm of a list of business requirements to address the stakeholders' needs

Week 10 - Database schema design and ER-Diagram

Week 11 - ETL process design to insert the data

Week 12 - Business analysis and customer interaction plan design

Week 13 - Presentation and showcase of our results and recommendations, final report

Dataset Overview and Selection

Our dataset consists of six tables retrieved from Kaggle, representing key aspects of ABC Foodmart's operations. These tables serve as the foundation for designing a comprehensive relational database system to streamline business processes. However, these tables are not in Third Normal Form (3NF), so they will need to be normalized and reorganized to eliminate redundancies, maintain data integrity, and enable efficient querying. Below is a brief description of each table and why it was selected:

1. **Customers_Table:** This table contains details about customers, including their contact information, addresses, and possibly loyalty program data. This data is critical for tracking customer interactions, purchase history, and loyalty program participation.
2. **Employees_Table:** This table includes information about employees, such as their positions, contact details, and potentially their assigned store locations. Employee data is crucial for managing staffing, scheduling, and operational efficiency.
3. **Product_Stock_Table:** This table captures stock information, including product IDs, quantities, and store locations. It provides essential insights into inventory levels, helping to manage stock-outs or overstock situations.
4. **Products_Table:** This table contains information about products, such as product names, descriptions, and prices. This data is vital for inventory tracking, sales analysis, and vendor management.
5. **Stores_table:** This table stores details about the grocery stores, including their addresses and contact information. It is critical for managing multiple store locations and linking transactions or stock data to specific stores.
6. **Transactions_Table:** This table logs sales transactions, capturing data such as transaction IDs, product IDs, quantities sold, and prices. It is crucial for analyzing sales trends, profitability, and customer purchase patterns.

These six tables were chosen because they comprehensively cover the core domains of ABC Foodmart's business operations, including customers, employees, inventory, products, stores, and transactions. Together, they offer a holistic view of the data required to optimize operations, improve decision-making, and support the company's expansion plans. However, as these tables are not normalized, breaking them down into 3NF will ensure data accuracy, minimize redundancy, and enhance query performance for analytical and reporting purposes. This normalization will prepare the data for integration into a relational database, enabling the creation of dashboards and reports that offer actionable insights into the company's operations.

Database Design

The database design for ABC Foodmart is structured with 16 normalized tables, adhering to Third Normal Form (3NF) to ensure data integrity, eliminate redundancy, and enhance query efficiency. The design follows a logical flow and sequence, starting with foundational tables that define core business entities like employees, stores, customers, products, and categories. These foundational tables support dependent tables that expand on operational and transactional details, followed by supporting tables that add analytical depth to the system. The relationships between these tables are defined through primary and foreign keys, reflecting a mix of one-to-one, one-to-many, and many-to-many cardinalities. This structure ensures scalability, robustness, and comprehensive data management across all business operations.

The database design is organized sequentially to establish foundational entities first, ensuring that all dependent tables have consistent and valid references:

1. Core Tables: Tables such as employee, store, customer, product, and category are created first to define the primary entities of the business. These tables store essential information, such as employee details, store locations, customer records, and product categories.
2. Dependent Tables: Tables like employee_detail, product_stock, and customer_transactions expand on the foundational entities by adding specific operational details, such as employee assignments, inventory levels, and sales transactions.
3. Supporting Tables: Supporting tables such as loyalty_program, operating_costs, payment, and transaction_details enhance the database by storing ancillary data, such as financial details, customer engagement metrics, and transaction-specific product data.

This structured approach ensures a clear hierarchy of data, enabling efficient data integration and seamless query execution for both operational and analytical needs.

Key Relationships and Cardinalities

1. Employee and Store Management

The employee table serves as the foundation for workforce data, capturing details such as names, positions, and employment statuses. It is linked to the employee_detail table in a one-to-one relationship, where additional data, such as salaries, shift types, and store assignments, is stored. Each employee has exactly one corresponding record in employee_detail, ensuring clarity and avoiding redundancy. The employee_detail table is connected to the store table in a many-to-one relationship, as multiple employees can be assigned to the same store, but each employee belongs to only one store. Additionally, the management table links to the store table in a one-to-many relationship, where a store can have multiple managers overseeing different aspects of operations.

2. Customer Management

Customer data is captured in the customer table, which serves as the central repository for customer information, including contact details and addresses. This table connects to the loyalty_program table in a one-to-one relationship, where each customer is associated with a single loyalty program record, tracking their membership tier and accumulated points. The customer_feedback table is linked to the customer table in a one-to-many relationship, allowing a single customer to leave multiple feedback entries tied to different transactions. These relationships enable detailed tracking of customer engagement and satisfaction.

3. Product and Inventory Management

The product table is central to inventory and sales operations, capturing details such as product names, descriptions, and unit prices. Each product is categorized using the category table, forming a one-to-many relationship, where a category can contain multiple products, but each product belongs to only one category. Inventory levels are managed through the product_stock table, which connects to both the product and store tables in one-to-many relationships, allowing precise tracking of quantities by product and location. Vendor relationships are managed through the vendor table, with the product_vendor table establishing a many-to-many relationship between products and vendors, capturing details like supply frequency.

4. Financial and Transaction Management

The customer_transactions table records sales transactions, linking each transaction to a specific store and customer in one-to-many relationships. Payments are tracked in the payment table, which connects to customer_transactions in a one-to-one relationship, ensuring each transaction has an associated payment record. The transaction_details table provides a detailed view of each transaction, capturing the products involved and their quantities. It connects to customer_transactions in a one-to-many relationship, as a single transaction can include multiple products.

5. Operational Costs

The operating_costs table tracks expenses incurred by each store, such as rent, utilities, and wages. It links to the store table in a one-to-many relationship, as a single store can have multiple expense entries over time. This table supports profitability analysis and operational cost tracking.

This database design ensures seamless integration of ABC Foodmart's core business domains, supporting efficient data management and robust analytics. The logical sequence and well-defined relationships between tables provide flexibility and scalability for current operations and future growth. By leveraging a mix of relationship types, the design enables detailed reporting, real-time insights, and data-driven decision-making to enhance operational efficiency and support expansion.

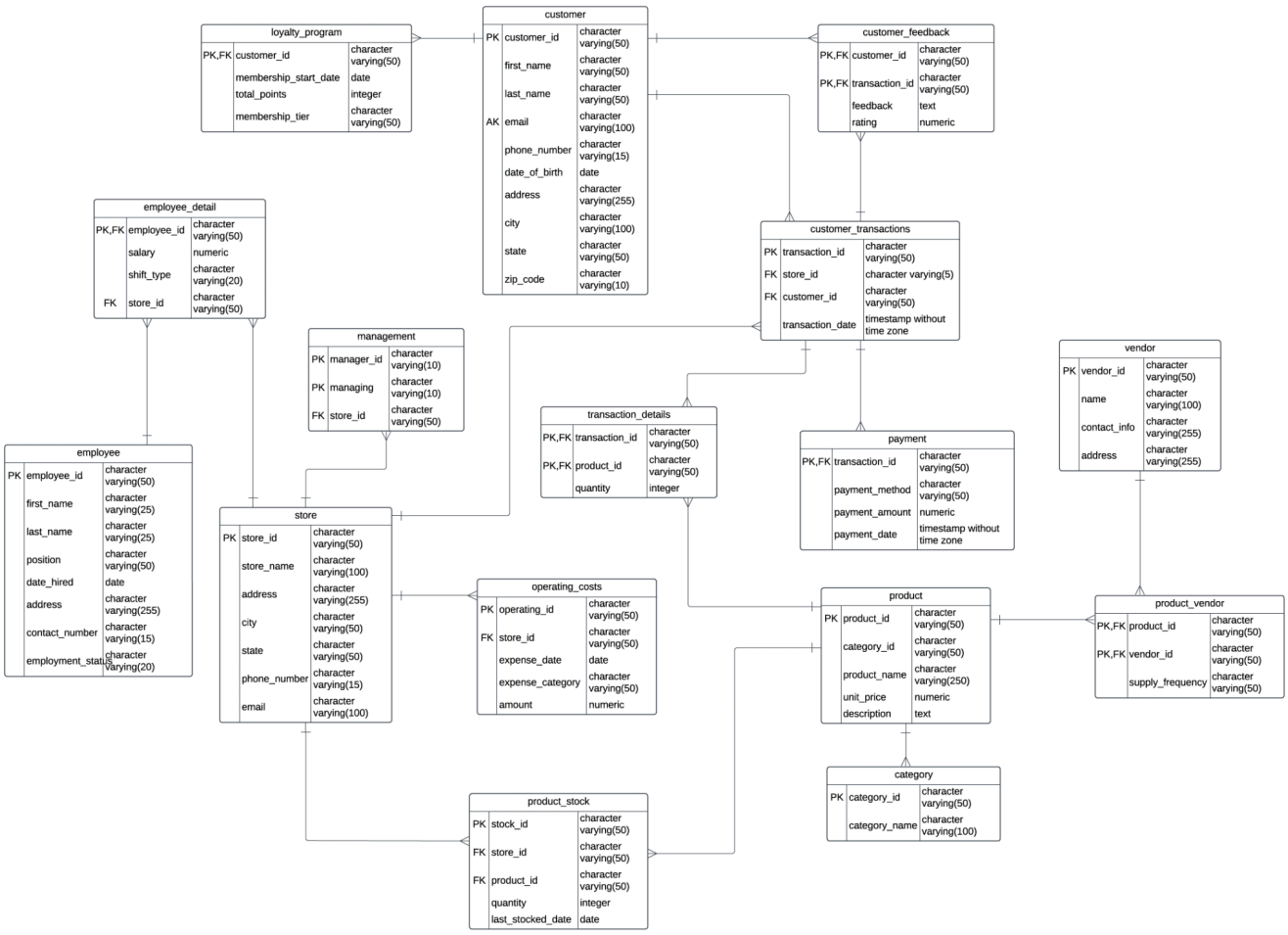


Figure. Entity-Relationship Diagram for ABC Foodmart

The ERD for ABC Foodmart visualizes the relational database schema, highlighting key entities like Customer, Store, Product, and Employee, along with supporting entities such as Product Stock, Product Vendor, and Transaction Details. Relationships are defined through primary and foreign keys, ensuring data consistency and referential integrity. The diagram captures practical applications, such as linking customers to sales transactions, tracking inventory by location, and mapping products to vendors for supply chain management.

Temporal attributes, like price_date in Product Pricing and expense_date in Operating Costs, enable historical data analysis, enhancing financial and operational insights. The ERD also demonstrates connections between business processes, such as linking sales to customer loyalty, monitoring store profitability, and analyzing inventory turnover. This well-structured design supports complex queries and analytics, laying a scalable foundation for ABC Foodmart's modernization and growth.

ETL Process GitHub Link: https://github.com/7effreyzzZ/SQL-Final-Project

Refer to File Name: 5310_Project_ETL.ipynb

ETL Process

Before conducting the ETL process, we have already designed and implemented a relational database schema to ensure proper data organization, integrity, and seamless insertion. Adhering to 3NF principles, the schema separates entities into individual tables to avoid redundancy, support efficient querying, and maintain data consistency through well-defined relationships such as foreign keys linking transactions to customers or products to vendors. This foundation allowed us to transform six original datasets into 16 normalized tables.

Data Manipulation 1 - ALL_Store Table

The All_Store dataset was split into two normalized data frames: store and operating_costs. First, we extracted unique store details like store_name, address, city, state, phone_number, and email into the store data frame. Each store was assigned a unique store_id by mapping the last letter of the store name to a number, ensuring a consistent identifier. Duplicate rows were removed based on store_id, and the resulting data was sorted to create a clean, organized table for static store information.

```
   store_id      store_name                      address      city state  \
0         1  ABC FoodMart A      123 Maple Avenue, Queens  New York    NY
1         2  ABC FoodMart B        456 Elm Street, Queens  New York    NY
2         3  ABC FoodMart C      101 Cedar Road, Brooklyn  New York    NY
3         4  ABC FoodMart D  789 Pine Boulevard, Brooklyn  New York    NY
4         5  ABC FoodMart E        202 Oak Lane, Brooklyn  New York    NY

   phone_number                  email
0  664-713-7151  ABCFoodMartA@gmail.com
1  881-729-5270  ABCFoodMartB@gmail.com
2  881-751-7981  ABCFoodMartC@gmail.com
3  881-729-5272  ABCFoodMartD@gmail.com
4  881-729-5273  ABCFoodMartE@gmail.com
Out[6]: 5
```

Next, operational costs were isolated into the operating_costs data frame by selecting columns like expense_date, expense_category, and amount. A unique operating_id was generated for each expense record, ensuring traceability. This table links to the store table using the store_id, enabling efficient tracking of each store's financial transactions. The separation into these two tables ensures data normalization, reduces redundancy, and facilitates scalable analysis of store performance and expenses.

```
In [7]: # Create opr_cost_df with the selected columns
opr_cost_df = df1[['store_id', 'expense_date', 'expense_category', 'amount']]

# Add Operating_id column with values starting from O1
opr_cost_df.insert(0, 'operating_id', ['O' + str(i) for i in range(1, len(opr_cost_df) + 1)])

# Display the result
print(opr_cost_df.head())
opr_cost_df.to_sql(name='operating_costs', con=engine, if_exists='append', index=False)
```

```
  operating_id  store_id expense_date expense_category   amount
0           O1         2    2022/4/29             Rent  3729.39
1           O2         2     2023/5/3      Maintenance  2210.33
2           O3         3    2023/8/19             Rent  3208.38
3           O4         1    2024/7/25      Maintenance  1135.39
4           O5         4    2022/1/21        Utilities   904.52
```

Data Manipulation 2 - ALL_Product Table

The ALL_Product table was transformed into four normalized data frames: product, category, vendor, and product_vendor, to organize and separate product-related information efficiently.

```
#Insert into product table
# Add `product_id` column based on unique `product_name`
product_mapping = {name: f'P{i+1}' for i, name in enumerate(df2['product_name'].unique())}
df2['product_id'] = df2['product_name'].map(product_mapping)

#  Add `category_id` column based on unique `Category`
category_mapping = {name: f'CE{i+1}' for i, name in enumerate(df2['Category'].unique())}
df2['category_id'] = df2['Category'].map(category_mapping)

# Add `vendor_id` column based on unique `vendor_name`
vendor_mapping = {name: f'V{i+1}' for i, name in enumerate(df2['vendor_name'].unique())}
df2['vendor_id'] = df2['vendor_name'].map(vendor_mapping)
```

First, the product table was created to store essential product details such as product_name, unit_price, and description. A unique product_id was generated for each product, ensuring clear identification. Next, the category table was derived by extracting unique Category values and assigning a category_id to each, linking products to their respective categories. This separation allows for better classification and analysis of products by category.

The vendor table was created to store details about suppliers, including vendor_name, contact_info, and address. Each vendor was assigned a unique vendor_id to establish a distinct identity. Finally, the product_vendor table captured the many-to-many relationship between products and vendors, including details like supply_frequency. This normalization process not only reduced redundancy but also established clear relationships between products, their categories, and vendors, facilitating efficient querying and analysis of product-related data.

Data Manipulation 3 - ALL_Employee Table

To transform the ALL_Employee table into the employee, employee_detail, and management data frames, we began by separating core employee information. The employee table includes fields such as employee_id, first_name, last_name and employment_status. We ensured that each employee was uniquely identified and removed duplicates based on employee_id. The data was also sorted using the numeric portion of employee_id to maintain consistency in organization. This table serves as the foundation for storing key employee details.

Next, we constructed the employee_detail table to capture job-specific attributes like salary, shift_type, and store_id. Using a mapping derived from the ALL_Store table, we linked store_name to store_id to ensure consistency across data sets. The original store_name column was removed after mapping, and duplicates were eliminated to create a clean data frame. Sorting was again performed using the numeric part of employee_id to keep the data well-structured.

```
temp_employee_detail_df.to_sql(name='employee_detail', con=engine, if_exists='append', index=False)
```

```
  employee_id  salary shift_type  store_id
0          E1   89000    Daytime         1
1          E2   89000    Daytime         2
2          E3   89000    Daytime         3
3          E4   89000    Daytime         4
4          E5   89000    Daytime         5
```

Out[24]: 25

Finally, we created the management table to track hierarchical relationships within the organization. The table includes manager_id, managing, and store_id. We iterated over the rows of the ALL_Employee table to parse the management column, splitting it into individual relationships. Each manager_id was mapped to its corresponding managing employees, and the store_name was linked to store_id using the same store mapping as before. Rows with missing store_id were dropped to maintain integrity. This table enables a clear representation of managerial assignments and reporting structures, essential for understanding team hierarchies.

```
# Iterate through the rows of the DataFrame
for _, row in df3.iterrows():
    # Check if 'management' is a valid string before splitting
    if isinstance(row['management'], str):
        manager_ids = row['management'].split(',')
        for manager_id in manager_ids:
            # Append data to the manager_table DataFrame
            manager_table = pd.concat([manager_table, pd.DataFrame({
                'manager_id': [row['employee_id']],  # Now `employee_id` is the manager
                'managing': [manager_id],           # `manager_id` becomes the managed employee
                'store_id': [store_mapping.get(row['store_name'], None)]  # Map store_name to store_id
            })], ignore_index=True)

# Drop rows with missing store_id
manager_table = manager_table.dropna(subset=['store_id'])

# Display the first few rows of the manager_table
print(manager_table.head())
manager_table.to_sql(name='management', con=engine, if_exists='append', index=False)
```

```
  manager_id managing store_id
0         E1       E6        1
1         E1      E11        1
2         E1      E16        1
3         E1      E21        1
4         E2       E7        2
```

Out[27]: 20

Data Manipulation 4 - ALL_Product_stock Table

To transform the ALL_Product_stock dataset into the product_stock data frame, we started by generating a unique stock_id for each record using a sequential format (S1, S2, etc.). We then mapped store_name to store_id using a mapping derived from the ALL_Store table to ensure consistency across data sets. Similarly, product_name was mapped to product_id using a mapping from the ALL_Products table. After applying these mappings, we extracted and organized the relevant columns: stock_id, store_id, product_id, quantity, and last_stocked_date. This data frame provides a clear view of inventory levels by linking products and stores, enabling efficient stock management and analysis.

```
df4 = df4[['stock_id', 'store_id', 'product_id', 'quantity', 'last_stocked_date']]

print(df4.head())
```

```
  stock_id  store_id product_id  quantity last_stocked_date
0       S1         1         P1       247         5/22/2023
1       S2         2         P1       920          7/8/2023
2       S3         3         P1       696         8/30/2023
3       S4         4         P1       772        11/12/2024
4       S5         5         P1       980         7/16/2023
```

Data Manipulation 5 - ALL_Customer Table

To transform the ALL_Customer dataset into the customer table, we started by assigning unique customer_id values to each record using a sequential format (C1, C2, etc.). The dataset was cleaned and renamed to align column names, such as converting zipcode to zip_code, ensuring consistency with the database schema. We then selected the relevant columns (customer_id, first_name, last_name, phone_number, email, date_of_birth, address, city, state, and zip_code) to create a clean and normalized customer DataFrame, storing static customer details.

```
# Insert the data into the customer table in the database
customer_df.to_sql(name='customer', con=engine, if_exists='append', index=False)
```

```
  customer_id first_name last_name    phone_number  \
0          C1      Uriah   Bridges  (401) 552-4059
1          C2      Paula     Small  (688) 210-1295
2          C3     Edward      Buck  (434) 593-6233
3          C4    Michael   Riordan  (517) 270-8979
4          C5    Jasmine     Onque  (214) 940-9676
```

The loyalty_program table was derived by extracting columns specific to the loyalty program: membership_startdate, total_points, and membership_tier. Using the email field as a unique identifier, we mapped customer_id values from the customer DataFrame into the loyalty data, ensuring referential integrity. After reordering the columns to match the database schema and removing records with missing customer_id, the loyalty_program DataFrame was finalized.

Data Manipulation 6 - ALL_Transaction Table

The customer_transactions and customer_feedback tables were derived from the ALL_Transaction dataset to capture high-level transaction details and customer reviews. For the customer_transactions table, store names were mapped to store_id using the store mapping, and phone numbers were used to retrieve customer_id from the customer dataset. Duplicate transaction entries were removed, ensuring each transaction_id appeared only once, along with associated store, customer, and date details. The customer_feedback table was then created by merging transaction data with customer_transactions to link each review and rating to a specific customer_id and transaction_id. Feedback records were cleaned by retaining the first occurrence of each transaction, supporting the analysis of user sentiment and transaction-level feedback.

The transaction_details table focuses on capturing the specific products purchased in each transaction and their quantities. Product names from the ALL_Transaction dataset were mapped to their respective product_id using the product mapping derived from the product table. The dataset was cleaned to drop rows with missing product_id and organized by sorting on transaction_id and product_id. Duplicates in the combination of

transaction_id and product_id were identified and removed to ensure data integrity. This table provides granular insights into transaction compositions, supporting analysis of product-level sales trends.

```
# Sort the data by transaction_id and product_id to organize the table
transaction_details_df = transaction_details_df.sort_values(by=['transaction_id', 'product_id']).reset_index(drop=True)

# Ensure no duplicates in transaction_id and product_id pairs (use for validation, optional)
duplicate_check = transaction_details_df.duplicated(subset=['transaction_id', 'product_id'], keep=False)
if duplicate_check.any():
    print("Warning: Duplicates detected in transaction_id and product_id pairs!")

print(transaction_details_df.head(20))
```

```
    transaction_id product_id  quantity
0        123000012       P112         2
1        123000012       P125         3
2        123000012        P74         2
3        123000012       P885         2
4        123000012       P911         2
5        123000012        P98         2
6        123000064      P1276         2
7        123000064       P205         2
8        123000064       P291         3
9        123000064        P42         3
```

The payment table consolidates financial data for each transaction. The payment_amount was calculated as the sum of the product of unit_price and quantity for all products in a transaction, grouped by transaction_id. Additional fields, such as the earliest payment_date and the primary payment_method for each transaction, were extracted. Data was further cleaned to ensure accurate calculations and consistent formats, with amounts rounded to two decimal places. The resulting table links financial details to transactions, enabling revenue analysis and tracking payment behaviors effectively.

```
# Ensure payment_amount is rounded to 2 decimal places
payment_df['payment_amount'] = payment_df['payment_amount'].round(2)

# Display the resulting DataFrame
print(payment_df.head())
```

```
   transaction_id  payment_amount      payment_date  payment_method
0       123000012         1215.87    2023/6/11 4:52  Mobile Payment
1       123000064          474.87    2023/9/17 0:01  Mobile Payment
2       123000121         2032.23  2023/12/25 16:16  Digital Wallet
3       123000136           55.97   2023/7/23 15:49     Credit Card
4       123000161          982.86  2023/11/27 21:38  Mobile Payment
```

Analytical Procedures : Insights and Business Value

Q1. What are the best-selling products?

```
SELECT
    p.product_id,
    p.product_name,
    c.category_name,
    SUM(td.quantity) AS total_sold
FROM
    transaction_details td
JOIN
    product p ON td.product_id = p.product_id
JOIN
    category c ON p.category_id = c.category_id
GROUP BY
    p.product_id, p.product_name, c.category_name
ORDER BY
    total_sold DESC
    Limit 10;
```

	product_id character varying (50)	product_name character varying (250)	category_name character varying (100)	total_sold bigint
1	P40	Kirkland Signature, Organic Almond Beverage, Vanilla, 32 fl oz, 6-Count	Beverages & Water	353
2	P112	Kirkland Signature, Organic Soy Beverage, Plain, 32 fl oz, 12-Count	Beverages & Water	349
3	P99	Kirkland Signature, Organic Soy Beverage, Vanilla, 32 fl oz, 12-Count	Beverages & Water	313
4	P161	Kirkland Signature, Organic 100% Juice, Variety Pack, 6.75 fl oz, 40-Count	Beverages & Water	287
5	P799	Kirkland Signature, Organic Chicken Stock, 32 fl oz, 6-Count	Kirkland Signature Grocery	255
6	P795	Kirkland Signature, Chicken Breast, 12.5 oz, 6-Count	Kirkland Signature Grocery	177
7	P116	Red Bull Energy Drink, Dragon Fruit, 8.4 fl oz, 24-count	Beverages & Water	171
8	P883	Crescent Foods Halal Hand-Cut Beef, Chicken Combo Pack - 14 Total Packs, 13.5 Lbs. Total	Meat & Seafood	162
9	P911	Wild Alaska Snow Crab Meat (Bairdi Crab 8 oz. Pack) 12 Total Packs, 6 Lbs. Total	Meat & Seafood	159
10	P793	Kirkland Signature Chewy Protein Bar, Peanut Butter & Semisweet Chocolate Chip, 1.41 oz, 42-Co…	Kirkland Signature Grocery	159

The first step was to integrate data from multiple tables to ensure a comprehensive understanding of the products and their categories. By using the SUM() function in SQL, we summed up the quantity column for each product across all transactions. From the ranked data, we observed that the top products were largely from the "Beverages & Water" category, with significant contributions from Kirkland Signature branded items. This suggests a strong demand for private-label products and beverages, which aligns with typical consumer behavior in grocery stores.

Q2. Which stores have the highest sales revenue?

```
SELECT
    s.store_id,
    s.store_name,
    s.address,
    SUM(td.quantity * p.unit_price) AS total_revenue
FROM
    transaction_details td
JOIN
    product p ON td.product_id = p.product_id
JOIN
    customer_transactions ct ON td.transaction_id = ct.transaction_id
JOIN
    store s ON ct.store_id = s.store_id
GROUP BY
    s.store_id, s.store_name
ORDER BY
    total_revenue DESC;
```

store_id [PK] character varying (50)	store_name character varying (100)	address character varying (255)	total_revenue numeric
1	ABC FoodMart A	123 Maple Avenue, Queens	474488.74
4	ABC FoodMart D	789 Pine Boulevard, Brooklyn	434985.93
2	ABC FoodMart B	456 Elm Street, Queens	373279.99
3	ABC FoodMart C	101 Cedar Road, Brooklyn	315220.83
5	ABC FoodMart E	202 Oak Lane, Brooklyn	238381.51

To determine store-level revenue, we first combined data to ensure that each transaction was tied to both the product's revenue contribution and the store where it occurred. Revenue for each product in a transaction was calculated as: Revenue=Quantity Sold×Unit Price. This calculation was performed for all transactions, and the results were aggregated by store using the SUM() function in SQL. ABC FoodMart A in Queens generated the highest revenue, totaling $474,488.74. The company could enhance operations and expand product offerings in the top-performing stores (ABC FoodMart A and D).

Q3. What is the average customer rating for each store?

```
SELECT
    s.store_id,
    s.store_name,
    ROUND(AVG(cf.rating),2) AS average_rating
FROM
    customer_feedback cf
JOIN
    customer_transactions ct ON cf.transaction_id = ct.transaction_id
JOIN
    store s ON ct.store_id = s.store_id
GROUP BY
    s.store_id, s.store_name
ORDER BY
    average_rating DESC;
```

store_id [PK] character varying (50)	store_name character varying (100)	average_rating numeric
1	ABC FoodMart A	3.77
5	ABC FoodMart E	3.73
3	ABC FoodMart C	3.68
2	ABC FoodMart B	3.62
4	ABC FoodMart D	3.58

For each store, the ratings were aggregated using the AVG() function. This step calculated the average rating by dividing the sum of all ratings by the number of ratings for each store. ABC FoodMart A received the highest average rating of 3.77, indicating strong customer satisfaction and likely high service quality or product availability. The store should continue to uphold excellent customer service and operational efficiency at ABC

FoodMart A to retain its top position. In addition, the store should conduct a deeper analysis of customer feedback for ABC FoodMart D to identify and address issues leading to lower ratings.

Q4. Which customers spend the most?

```sql
SELECT
    ct.customer_id,
    CONCAT(c.first_name, ' ', c.last_name) AS customer_name,
    SUM(td.quantity * p.unit_price) AS total_spent
FROM
    customer_transactions ct
JOIN
    transaction_details td ON ct.transaction_id = td.transaction_id
JOIN
    product p ON td.product_id = p.product_id
JOIN
    customer c ON ct.customer_id = c.customer_id
GROUP BY
    ct.customer_id, customer_name
ORDER BY
    total_spent DESC
    LIMIT 10;
```

customer_id character varying (50)	customer_name text	total_spent numeric
C776	Shari Ngodup	5235.78
C196	Harrison Hudson	4886.87
C429	Marlee Stevens	4834.63
C1001	Mylee Snow	4792.15
C679	Sincere Rivera	4759.56
C1401	Ellie Ortega	4753.11
C1169	Lamont Cook	4658.09
C1741	Valentino Myers	4605.54
C1562	Terrell Guerra	4484.28
C1209	Lilliana Walker	4101.56

This table highlights the top customers of ABC Foodmart based on their total spending. These transaction-level revenues were summed up for each customer across all their transactions using the SUM() function. We used CONCAT() to combine the customers' first name and last name. The top 10 customers all spent over $4,100, representing a valuable segment of loyal or high-value customers. The store could introduce exclusive benefits for top spenders to enhance customer retention and satisfaction.

Q5. What are the monthly sales trends?

```sql
SELECT
    TO_CHAR(ct.transaction_date, 'YYYY-MM') AS month,
    SUM(td.quantity * p.unit_price) AS total_revenue
FROM
    transaction_details td
JOIN
    customer_transactions ct ON td.transaction_id = ct.transaction_id
JOIN
    product p ON td.product_id = p.product_id
GROUP BY
    TO_CHAR(ct.transaction_date, 'YYYY-MM')
ORDER BY
    month;
```

month text	total_revenue numeric
2023-01	254716.59
2023-02	82461.93
2023-03	45469.45
2023-04	101198.62
2023-05	128484.98
2023-06	157087.87
2023-07	148197.08
2023-08	115754.82
2023-09	93600.53
2023-10	91781.15
2023-11	295536.07
2023-12	322067.91

The table displays total revenue generated by ABC Foodmart for each month in 2023. The TO_CHAR() function was used to format transaction_date into the YYYY-MM format, grouping all transactions by month. December recorded the highest revenue ($322,067.91), likely driven by holiday shopping and end-of-year promotions. Revenue from May to August was stable, with consistent sales in the range of $115,754.82 to $157,087.87. The stores should introduce promotions or loyalty incentives during Q1 to drive customer

engagement and boost revenue in slower months like February and March. Also, they could maintain consistent marketing efforts during the summer to ensure steady revenue.

Q6. Which product categories contribute the most revenue?

```sql
SELECT
    c.category_name,
    SUM(td.quantity * p.unit_price) AS total_revenue
FROM
    transaction_details td
JOIN
    product p ON td.product_id = p.product_id
JOIN
    category c ON p.category_id = c.category_id
GROUP BY
    c.category_name
ORDER BY
    total_revenue DESC;
```

	category_name character varying (100)	total_revenue numeric
1	Meat & Seafood	710897.32
2	Beverages & Water	310840.47
3	Candy	200130.57
4	Bakery & Desserts	186118.52
5	Gift Baskets	112706.79
6	Snacks	98675.01
7	Coffee	94490.61
8	Kirkland Signature Grocery	39418.65

The query integrates data from multiple tables to calculate total revenue by product category, using the SUM() function to aggregate the product of quantity and unit_price. Results show "Meat & Seafood" as the top contributor ($710,897.32), followed by "Beverages & Water" ($310,840.47) and "Candy" ($200,130.57), reflecting consumer preferences for high-protein and convenience products. To capitalize on these trends, we could expand offerings and promotions in these top-performing categories while using strategic placement or pricing to boost sales in lower-performing ones like "Kirkland Signature Grocery" and "Pantry & Dry Goods."

Q7. What are the three most profitable products in each store?

```sql
WITH RankedProducts AS (
    SELECT
        s.store_id,
        s.store_name,
        p.product_id,
        p.product_name,
        SUM(td.quantity * p.unit_price) AS total_profit,
        ROW_NUMBER() OVER (PARTITION BY s.store_id ORDER BY SUM(td.quantity * p.unit_price) DESC) AS rank
    FROM
        transaction_details td
    JOIN
        product p ON td.product_id = p.product_id
    JOIN
        customer_transactions ct ON td.transaction_id = ct.transaction_id
    JOIN
        store s ON ct.store_id = s.store_id
    GROUP BY
        s.store_id, s.store_name, p.product_id, p.product_name
)
SELECT
    store_id,
    store_name,
    product_id,
    product_name,
    total_profit
FROM
    RankedProducts
WHERE
    rank <= 3
ORDER BY
    store_id, rank;
```

store_id character varying (50)	store_name character varying (100)	product_id character varying (50)	product_name character varying (250)	total_profit numeric
1	ABC FoodMart A	P908	Rastelli Petite Filet Mignon & Jumbo Lump Crab Cake Surf & Turf, 24 Total Packs, 6.75 Total Lbs.	14699.58
1	ABC FoodMart A	P891	Chicago Steak - Steak & Cake - Filet Mignon, Crab Cakes, and Steak Burgers, Total 13 Packs, 6.5 Lbs. T...	8799.56
1	ABC FoodMart A	P925	A5 Wagyu Surf & Turf Pack, Cold Water Lobster Tails & Japanese A5 Wagyu Filet Mignon, Total 4 Packs	7219.81
2	ABC FoodMart B	P908	Rastelli Petite Filet Mignon & Jumbo Lump Crab Cake Surf & Turf, 24 Total Packs, 6.75 Total Lbs.	7699.78
2	ABC FoodMart B	P928	Japanese A5 Wagyu Ribeye Steaks, (3/16 Oz. Steaks), 3 Total Packs, 3 Lbs. Total	7049.85
2	ABC FoodMart B	P911	Wild Alaska Snow Crab Meat (Bairdi Crab 8 oz. Pack) 12 Total Packs, 6 Lbs. Total	5599.80
3	ABC FoodMart C	P911	Wild Alaska Snow Crab Meat (Bairdi Crab 8 oz. Pack) 12 Total Packs, 6 Lbs. Total	12319.56
3	ABC FoodMart C	P908	Rastelli Petite Filet Mignon & Jumbo Lump Crab Cake Surf & Turf, 24 Total Packs, 6.75 Total Lbs.	7349.79
3	ABC FoodMart C	P885	D'Artagnan Green Circle Chicken - Boneless & Skinless Breasts, 12 Total Packs, 11 Lbs. Total	6299.55

The query identifies the top three most profitable products in each store by calculating total profit as the difference between total revenue (quantity × unit price) and total cost (quantity × cost price). The results were grouped by store_id and product_id and ranked to select the top three products for each store.

From the analysis, premium meat and seafood products, such as "Rastelli Petite Filet Mignon & Jumbo Lump Crab Cake Surf & Turf" and "Wild Alaska Snow Crab Meat," dominate the profitability rankings across all

stores. For instance, at ABC FoodMart A, the former generated the highest profit of $14,699.58. This trend is consistent in other stores, where similar products rank among the top contributors. These results suggest that customers prioritize high-quality protein items, making them critical drivers of profitability. Therefore, we should focus on maintaining inventory and marketing these premium products while exploring opportunities to introduce new high-margin items within the same category.

Q8. Which 10 vendors contribute most to the supply chain?

```sql
SELECT
    v.vendor_id,
    v.name AS vendor_name,
    pv.supply_frequency,
    COUNT(DISTINCT pv.product_id) AS total_products_supplied,
    SUM(ps.quantity) AS total_stock_quantity
FROM product_vendor pv
JOIN product_stock ps ON pv.product_id = ps.product_id
JOIN vendor v ON pv.vendor_id = v.vendor_id
GROUP BY v.vendor_id, v.name, pv.supply_frequency
ORDER BY total_stock_quantity DESC;
```

	vendor_id character varying (50)	vendor_name character varying (100)	supply_frequency character varying (50)	total_products_supplied bigint	total_stock_quantity bigint
1	V7	Happy Snacks Inc.	Monthly	20	58676
2	V8	Premium Meat Suppliers	Quarterly	19	55559
3	V4	Fresh Catch Seafood	Daily	14	49482
4	V1	Dairy Delight Distributors	Weekly	13	46045
5	V7	Happy Snacks Inc.	Weekly	13	44460
6	V17	Beverage World	Quarterly	14	44449
7	V26	Natural Grocers Co.	Weekly	14	44185
8	V29	Healthy Living Organics	Quarterly	12	44153
9	V14	Super Sweets Distributors	Monthly	12	44068
10	V3	Refreshment Depot	Quarterly	15	43994

This analysis combines vendor supply frequency, the number of products supplied, and the total stock quantity to evaluate vendor performance. "Happy Snacks Inc." leads with the highest total stock quantity of 58,676 across its 20 products, supplied on a monthly basis. Vendors with higher stock quantities and frequent supplies, such as "Dairy Delight Distributors," demonstrate strong support in maintaining inventory consistency. The results highlight that vendors with frequent supply schedules ensure a steady inventory flow, but those with monthly or quarterly schedules, such as "Happy Snacks Inc." still manage to achieve significant stock contributions. We could strengthen partnerships with top vendors and strategize inventory planning based on their supply frequency and total stock contributions to avoid stockouts and optimize product availability.

Q9. Who are the 10 most loyal customers?

```sql
SELECT
    c.customer_id,
    c.first_name || ' ' || c.last_name AS full_name,
    lp.membership_start_date,
    lp.total_points,
    lp.membership_tier,
    c.city
FROM
    loyalty_program lp
JOIN
    customer c ON lp.customer_id = c.customer_id
ORDER BY
    lp.total_points DESC,
    lp.membership_start_date ASC;
```

	customer_id character varying (50)	full_name text	membership_start_date date	total_points integer	membership_tier character varying (50)
1	C1629	Peter Monroe	2005-05-07	9972	Gold
2	C844	Elena Gilmore	2022-12-11	9970	Gold
3	C850	Zoie Mercado	2008-02-11	9959	Gold
4	C762	Larissa Douglas	2023-08-21	9913	Gold
5	C297	Nancy Short	2020-10-12	9884	Gold
6	C1179	Rebecca Craig	2018-07-22	9883	Gold
7	C1210	Kristin Mccarty	2008-09-07	9880	Gold
8	C658	Aimee Terry	2008-11-20	9856	Gold
9	C67	Deborah Love	2006-10-10	9844	Gold
10	C105	Susan Lundy	2008-10-08	9837	Gold

The table highlights the top 10 most loyal customers of the company based on total points accumulated in the loyalty program. All of the top customers belong to the "Gold" membership tier, indicating their high engagement and long-term value to the business. Among these, Peter Monroe from Paterson leads with 9,972 points, followed by Elena Gilmore and Zoie Mercado, both from New York, with 9,970 and 9,959 points, respectively. Additionally, earlier joiners like Peter Monroe demonstrate the effectiveness of long-term

membership in building loyalty. Offering incentives to recent members like Elena Gilmore could encourage sustained engagement and increased spending over time. For instance, we could provide personalized discounts based on their purchase history or favorite products, or offer early access to sales or special events.

Q10. Store Expense Analysis

```
SELECT
    store_id,
    expense_category,
    SUM(amount) AS total_expense,
    (SUM(amount) * 100.0) /
    (SELECT SUM(amount)
    FROM operating_costs WHERE store_id = o.store_id) AS percentage_share
FROM operating_costs o
GROUP BY store_id, expense_category
ORDER BY store_id, percentage_share DESC;
```

store_id character varying (50)	expense_category character varying (50)	total_expense numeric	percentage_share numeric
1	Rent	145920.23	42.3748197323357146
1	Maintenance	86870.71	25.2270070864746683
1	Advertising	55506.46	16.1189180998419688
1	Utilities	39287.88	11.4090886004335223
1	Cleaning Services	16770.71	4.8701664809141261
2	Rent	129690.04	39.5697652060406711
2	Maintenance	79016.63	24.1087865843251299
2	Advertising	64874.76	19.7939565829283350
2	Cleaning Services	27158.99	8.2864872085258554
2	Utilities	27009.92	8.2410044181800086
3	Rent	160900.35	48.2351345274675860
3	Maintenance	65284.02	19.5710170126658173
3	Advertising	54512.46	16.3418901296559994

The analysis provides an overview of operating expenses for each store, categorized into Rent, Maintenance, Advertising, Utilities, and Cleaning Services. Rent consistently accounts for the highest share of expenses across all stores, while Maintenance expenses show notable variability, comprising 24.11% of total expenses in Store 2 but dropping to 20.64% in Store 5. Cleaning Services generally represent the smallest expense category, with an average share below 10%. This data suggests potential areas for cost optimization, particularly in Rent and Maintenance, as they dominate overall expenses. Targeted strategies, such as negotiating rental agreements or improving maintenance efficiency, could significantly impact profitability.

Database Interaction Plan and User Workflow

For Analysts: Direct Querying

Analysts will interact with the database through SQL interfaces and tools such as pgAdmin to query data directly. To streamline operations, predefined SQL queries will be developed for common use cases. For instance, sales analysis queries will provide insights into total sales by store, product category, or time period. Inventory tracking queries will help identify products with low stock levels or those that are out of stock. Customer insights queries will evaluate the performance of loyalty programs and feedback trends, while operating cost queries will break down expenses by store and category. In addition to these predefined queries, analysts will have the flexibility to write custom queries for exploring specific scenarios and addressing unique business questions.

The implementation will leverage PostgreSQL as the primary database management system for relational data querying and storage. For advanced data manipulation and reporting, Python will be used, specifically employing the Pandas library to process data and export results into Excel or CSV formats. This approach

ensures that data analysis is both robust and accessible, enabling further examination and presentation in widely used tools like spreadsheets.

For “C” level officers

For C-Level officers, high-level reports and dashboards will be implemented to provide real-time insights and visualizations tailored to their strategic decision-making needs. Automated reports will be designed to highlight key performance indicators (KPIs) such as profitability metrics (e.g., total revenue, operating costs), sales trends (e.g., top-performing stores and products), and customer engagement (e.g., loyalty program participation and feedback analysis). These insights will empower leadership to monitor business performance and make data-driven decisions effectively.

The primary tools used will include Matabase for creating interactive and visually appealing dashboards. Scheduled reports will be automated to generate and email PDF or Excel summaries periodically, such as weekly or monthly updates, using Python scripts or built-in Power BI features. Example reports include a Profitability Report, which compares revenue and operating costs by store and month, and a Customer Feedback Report, which tracks ratings and feedback trends to identify areas for improvement.

Dashboards will feature top-level tabs such as "Sales Overview," "Inventory Insights," "Customer Analytics," "Operating Costs," and "Vendor Performance" to provide a comprehensive overview of business metrics. Interactive visuals will include line charts for monthly revenue trends, heatmaps to visualize store performance by revenue, and pie charts to display payment method distribution. These features will enable C-Level officers to gain actionable insights at a glance, improving operational efficiency and driving strategic growth.

Infrastructure Planning: Redundancy, Performance, and Hosting Strategy

To ensure a robust and reliable database system capable of handling large-scale operations, redundancy and performance measures are in place. Automated daily backups using PostgreSQL tools are stored locally and in cloud platforms like AWS S3. Streaming replication ensures high availability with a real-time hot standby database, and a disaster recovery plan enables full data restoration within 1–2 hours. Weekly off-site backups mitigate risks from localized failures.

For optimal performance, indexing on key fields like store_id and customer_id speeds up queries, while partitioning large tables by date enhances time-based query efficiency. Query caching tools, such as pgbouncer or Redis, reduce database load, and connection pooling optimizes concurrent access. SQL queries are analyzed with PostgreSQL’s EXPLAIN and ANALYZE tools, and the database is hosted on scalable platforms like AWS RDS to handle peak usage seamlessly.

Data Visualization and Dashboard Insights

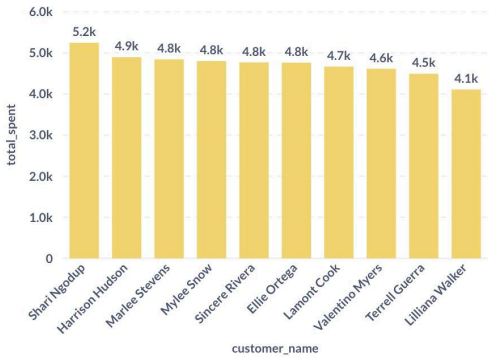
http://localhost:3000/public/dashboard/7f183dff-78cd-4398-96ee-22c60993f34b

This dashboard effectively visualizes data sets and provides actionable insights using three types of charts. By combining line charts, pie charts, and bar charts, the dashboard provides a comprehensive view of sales performance, product distribution, and customer satisfaction across multiple dimensions. Each chart is customized to address a specific aspect of the data, ensuring that the insights presented are clear and relevant.

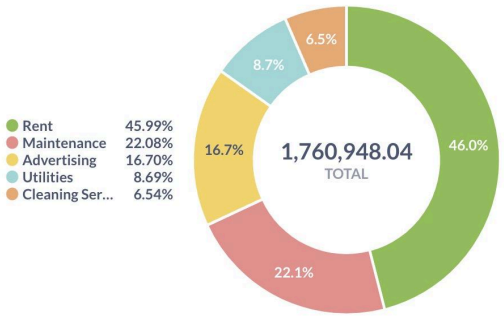
Each chart serves a distinct purpose: line charts excel at illustrating trends over time, pie charts effectively demonstrate proportions within a whole, and bar charts are ideal for comparing categories. For example, we use a line chart to visualize monthly sales trends, showing how total revenue changes throughout the year. This chart highlights key revenue peaks in January and November, likely tied to seasonal or promotional periods, as well as dips like the one in March, which may indicate areas for improvement. Similarly, the pie chart is used to display the proportion of sales contributed by different product categories. For instance, it reveals that "Beverages & Water" account for 61.8% of sales, making it the most significant contributor.

Lastly, we use a bar chart to compare the average customer ratings across various stores. Highlighting the best-rated store, "ABC FoodMart A," and the lowest-rated store, "ABC FoodMart D," helps identify operational strengths and areas requiring attention.

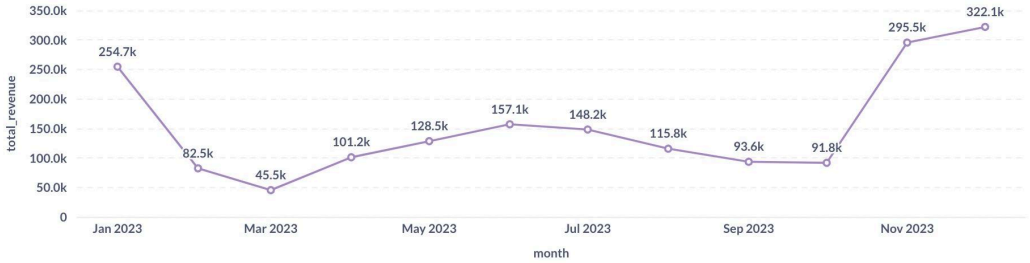
Customers who spend the most



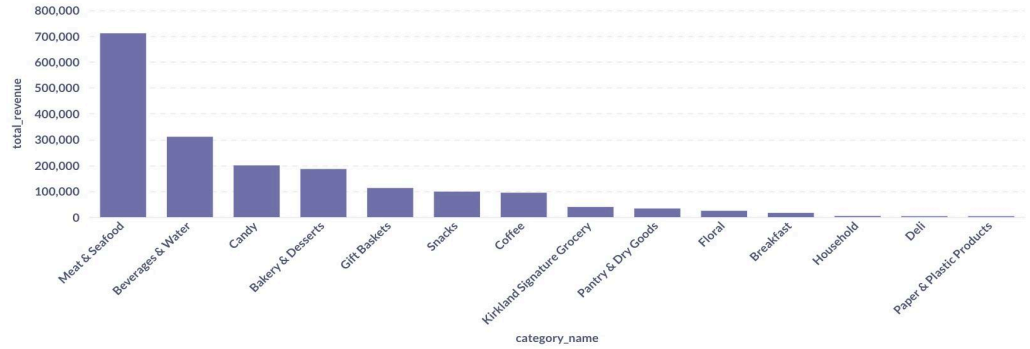
Store Expense Analysis



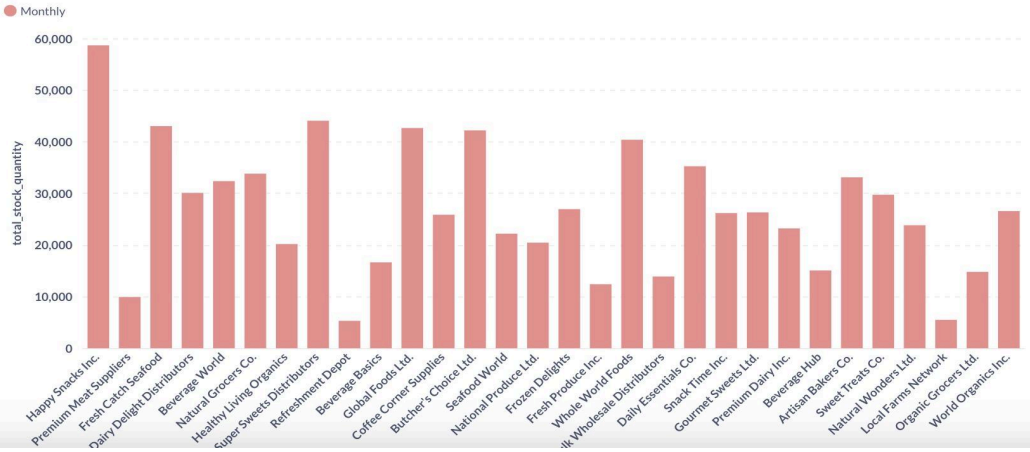
The monthly sales trends

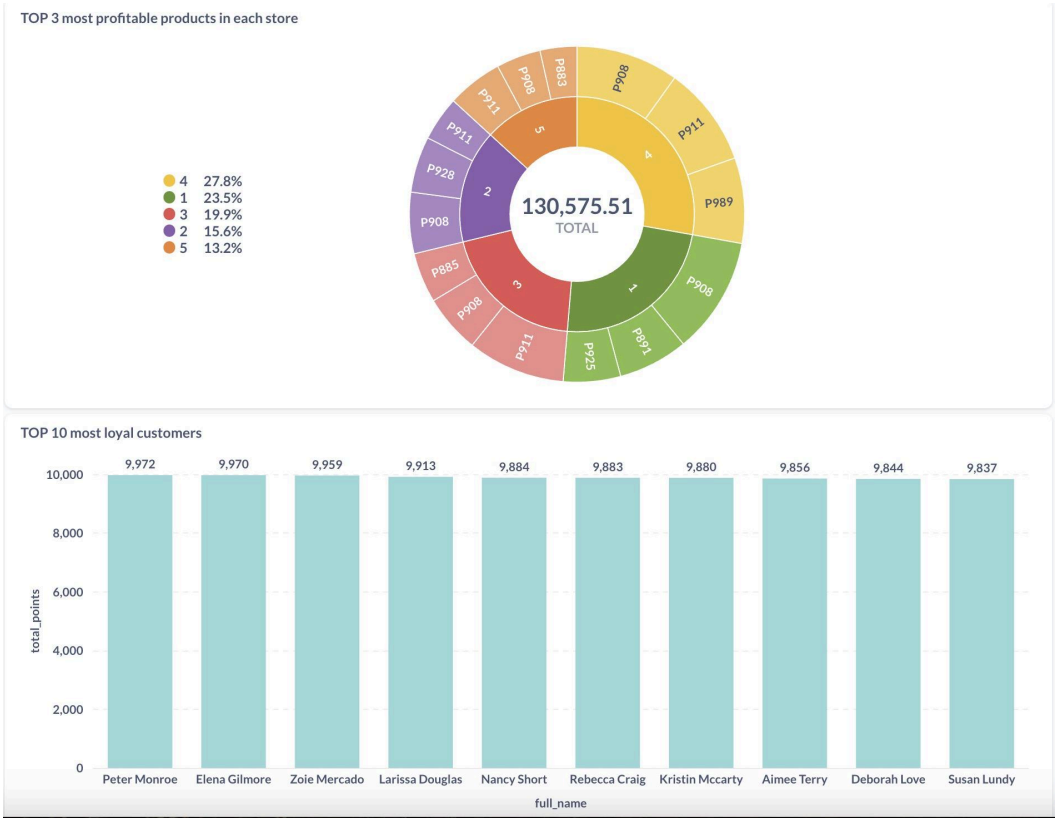


Product categories that contribute the most revenue



Vendor with the highest contribution to the supply chain





Conclusion

We addressed ABC Foodmart's operational inefficiencies by implementing a robust RDMS, efficient ETL processes, and comprehensive data analysis. The RDMS provided a scalable, normalized structure that ensured data integrity and streamlined key business areas like inventory, sales, and customer engagement. ETL processes ensured clean and consistent data migration, while analysis delivered actionable insights, such as identifying top-performing stores, best-selling products, peak shopping hours, and cost-saving opportunities. These insights informed strategic decisions, including targeted promotions, optimized inventory management, and enhanced customer loyalty programs. The solution improved decision-making with real-time data, increased operational efficiency, supported scalability for market expansion, and boosted profitability through better cost management and customer engagement.