A yellow store front with people walking past

Description automatically generated

Created by

Denso Daniel

**TABLE OF CONTENTS**

1. Introduction 3

[2. Purpose](#_30j0zll) 4

3. Goals 5

4. Business Rules 5

[5. Subjects](#_1fob9te) 6

6. List of Tables 6

7. List of Attributes 7

8. Entity Relationship 9

9. Conclusion 10

Appendix11

1. **Introduction**

FreshCo is a modern, intuitive grocery delivery service that prioritizes freshness, convenience, and sustainability. We partner with local farmers and suppliers to bring you the best products while supporting the community. This repository houses the code, documentation, and resources to enhance and expand FreshCo's capabilities. To support its operations, FreshCo requires a robust database design that ensures data integrity, performance, scalability, and security. This case study explores the steps FreshCo took to achieve an optimal database design.

**2. Purpose**

Develop a single, unified database system that serves as the primary repository for all data generated by FreshCo locations. This centralization ensures that all data is stored in one place, making it easier to manage, access, and analyze. By centralizing the database, data becomes readily accessible to authorized users across the organization. Staff at different FreshCo locations can retrieve up-to-date information quickly and efficiently, leading to better decision-making and improved operational efficiency. Aggregate data from multiple FreshCo locations into this centralized database. This includes sales data, inventory levels, customer information, supplier details, and other operational metrics. Consolidating data eliminates silos and provides a comprehensive view of the entire business.

**3. Goals**

Implement a centralized inventory management system that provides real-time visibility into stock levels across all stores. This ensures that each store can access the same information, reducing discrepancies and enabling better decision-making. Develop and enforce standardized inventory management procedures, including guidelines for ordering, receiving, storing, and tracking inventory. Consistency in these processes helps prevent stockouts, overstock, and discrepancies. Conduct regular inventory audits in all stores to verify accuracy and identify any issues early. These audits help maintain the integrity of inventory records and ensure that the physical stock matches the system records. Provide comprehensive training for staff at all stores on inventory management best practices and the use of the centralized system. Ongoing support and refresher training help maintain consistency and address any challenges.

**4. Business Rules**

* Data should be entered into the system within 24 hours of its generation to ensure the database remains current.
* Each supplier must have a unique identifier and provide valid contact information.
* Data should be structured to facilitate advanced analytics, enabling the use of business intelligence tools for insights and decision-making.
* Every inventory item must have a unique identifier and real-time quantity updates.
* Each customer must have a unique identifier and complete contact details.
* Each employee must have a unique identifier and accurate role information.
* Each sales transaction must have a unique identifier and link to a specific property.
* All data entries must be validated and updated in real-time.
* Database access must be role-based and logged for critical changes.
* The system must generate regular reports for inventory, sales, and supplier efficiency.
* The database should support comprehensive reporting capabilities, allowing users to generate customized reports for various business needs.

**5. Subjects**

* Stockouts: Stockouts occur when inventory runs out, leading to missed sales and dissatisfied customers.
* Overstocking: Overstocking happens when excess inventory is held, causing increased storage costs and potential waste.
* Centralized System: A centralized system consolidates inventory data into one location, improving consistency and coordination but can be complex to implement.
* Record Accuracy: Record accuracy ensures that inventory records match actual stock levels, crucial for efficient operations and correct financial reporting.
* Decision-Making: Decision-making in inventory management involves using data and strategic planning to maintain optimal stock levels and ensure efficient operations.

**6. List of Tables**

* **Product Table:** Stores information about ProductID including ProductName, Category, Price, SupplierID and StockQuantity
* **Order Table**: Stores information about Order ID including CustomerID

OrderDate, Ship Date, Status and Total Amount.

* **Customers Table**: Stores information about customers including their unique identifier, name, contact information, and address.
* **Store Table:** Stores information about StoreID, StoreName, Address, City and State
* **Supplier Table:** Stores information about SupplierID , SupplierName, Contact Name, ContactEmail, ContactPhone and Address
* **Employees Table**: Stores information about employees including their unique identifier, name, role, contact information, and hire date
* **Inventory Table**: Manages inventory items including their unique identifier, product name, available quantity, unit price, supplier information, date received, and the employee responsible for managing the item.
* **Order Details Table**: Manages details of Order DetailID , OrderID , ProductID , Quantity, UnitPrice and TotalPrice

**7.List of Attributes**

### **Product Table**

* **product\_id**: Unique identifier for each product.
* **product\_name**: Name of the product.
* **Price:** Price of the product
* **contact\_info**: Contact information of the employee.
* **Stock\_quantity**: stock of the product available.

### **Order Table**

* **order\_id**: Unique identifier for each order.
* **Customer\_id**: Unique identifier for each customer.
* **Order\_date:** date of purchase
* **Status:** Status of the item (e.g., available, sold).
* **Store\_id:** Unique identifier for each store.

### **Customers Table**

* **customer\_id**: Unique identifier for each customer.
* **customer\_name**: Name of the customer.
* **contact\_info**: Contact information of the customer.
* **address**: Address of the customer.

### **Store Table**

* **store\_id**: Unique identifier for each store.
* **store\_name**: Name of the store.
* **address**: Address of the customer.

### **Suppliers Table**

* **supplier\_id**: Unique identifier for each supplier.
* **supplier\_name**: Name of the supplier.
* **contact\_info**: Contact information of the supplier.
* **address**: Address of the supplier.

### **Employees Table**

* **employee\_id**: Unique identifier for each employee.
* **employee\_name**: Name of the employee.
* **employee\_role**: Role of the employee.
* **contact\_info**: Contact information of the employee.
* **hire\_date**: Date when the employee was hired.

### **Inventory Table**

* **inventory\_id**: Unique identifier for each inventory item.
* **product\_name**: Name of the product.
* **quantity\_instock**: Quantity of the product available.
* **unit\_price**: Price per unit of the product.

### **Order Table**

* **orderdetails\_id**: Unique identifier for each order.
* **Product\_id**: Unique identifier for each product.
* **quantity**: Quantity of the product available.
* **Unit\_price**: Pricing of each product.
* **Total\_price**: Total pricing on the bill generated.

**8. Entity Relationship**

* **Sales and Employees**
  + Relationship (One-to-Many)
* **Employee and store**: many to one relationship
  + Relationship (Many-to-One)
* **Customer and order**:
  + Relationship (One-to-Many)
* **Order and Product**
  + Relationship (Many-to-Many)
* **Product and Supplier**:
  + Relationship (Many-to-One)
* **Store and Product**:
  + Relationship (Many-to-Many)

**9. Conclusion**

By following best practices in database design, FreshCo successfully created a robust, scalable, and secure database that met its operational needs. The well-structured schema ensured data integrity, optimized performance, and supported future growth.

Key takeaways include the importance of maintaining data integrity, organizing data effectively, and establishing clear relationships between tables. By following these guidelines, you can create a database that not only stores information efficiently but also enhances business operations and decision-making processes.

This case study highlights the importance of thoughtful database design in achieving business objectives and enhancing overall efficiency.

**Appendix**

# **Data Dictionary**

**Products Table:**

|  |  |  |  |
| --- | --- | --- | --- |
| **Column Name** | **Data Type** | **Description** | **Constraints** |
| product\_id | INTEGER | Unique identifier for each product | Primary Key |
| product\_name | VARCHAR | Name of the product | Not Null |
| category | VARCHAR | Role of the employee | Not Null |
| price | DECIMAL | Price per unit of the product | Not Null |
| supplier\_id | INTEGER | Identifier for the supplier | Foreign Key (suppliers.supplier\_id), Not Null |

**Employees Table:**

|  |  |  |  |
| --- | --- | --- | --- |
| **Column Name** | **Data Type** | **Description** | **Constraints** |
| employee\_id | INTEGER | Unique identifier for each employee | Primary Key |
| employee\_name | VARCHAR | Name of the employee | Not Null |
| employee\_role | VARCHAR | Role of the employee | Not Null |
| contact\_info | VARCHAR | Contact information of the employee | Not Null |
| hire\_date | DATE | Date when the employee was hired | Not Null |

**Customers Table:**

|  |  |  |  |
| --- | --- | --- | --- |
| **Column Name** | **Data Type** | **Description** | **Constraints** |
| customer\_id | INTEGER | Unique identifier for each customer | Primary Key |
| customer\_name | VARCHAR | Name of the customer | Not Null |
| contact\_info | VARCHAR | Contact information of the customer | Not Null |
| address | VARCHAR | Address of the customer | Not Null |

**Inventory Table:**

|  |  |  |  |
| --- | --- | --- | --- |
| **Column Name** | **Data Type** | **Description** | **Constraints** |
| inventory\_id | INTEGER | Unique identifier for each inventory item | Primary Key |
| product\_name | VARCHAR | Name of the product | Not Null |
| quantity\_available | INTEGER | Quantity of the product available | Not Null |
| unit\_price | DECIMAL | Price per unit of the product | Not Null |
| supplier\_id | INTEGER | Identifier for the supplier | Foreign Key (suppliers.supplier\_id), Not Null |
| date\_received | DATE | Date when the inventory item was received | Not Null |
| employee\_id | INTEGER | Identifier for the employee managing the inventory | Foreign Key (employees.employee\_id), Not Null |

**Suppliers Table:**

|  |  |  |  |
| --- | --- | --- | --- |
| **Column Name** | **Data Type** | **Description** | **Constraints** |
| supplier\_id | INTEGER | Unique identifier for each supplier | Primary Key |
| supplier\_name | VARCHAR | Name of the supplier | Not Null |
| contact\_info | VARCHAR | Contact information of the supplier | Not Null |
| address | VARCHAR | Address of the supplier | Not Null |