Milestone 2: Database Design and CRUD Operations

CNIT 37200

Samuel Gomez

Noah Zhou

John Hsu

Nicholas Lauw

Submitted To: Dr. Tianyi Li

Date Submitted: 10/17/2024

Date Due: 10/27/2024

Insights Derived from Data

The insights we aim to derive include:

- Identification of patterns in product demand based on regions and seasons.
- Stock shortage predictions using historical data and trends.
- Identification of overstock and slow-moving items.
- Supplier lead time estimation for efficient replenishment.
- Reorder point optimization for different product categories.

Importance of These Insights

These insights are essential for efficient inventory management and business operations. By optimizing stock levels for each region, businesses can ensure products are readily available where they are most needed, minimizing the risk of stockouts that could lead to lost sales and dissatisfied customers.

- This helps optimize inventory levels by region, ensuring products are available when and where they are needed.
- Preventing stockouts is crucial to maintaining sales and customer satisfaction.
- Reducing excess stock saves costs on storage and decreases the chance of unsold items.

Tables Outline

- **Inventory**: information about products and their stock levels will be stored in each store.
- **Sales Transactions**: record each sale, linking it to the product and the store.
- **Suppliers**: will store supplier details and link to products to track supplier performance.
- **Products:** Stores information about all products, including unique identifiers and descriptive names. The Products table serves as the central reference point for other tables like Inventory and Suppliers, ensuring that each product has a unique identity throughout the system.

Database Design

In the following list, here are some of the column names, data types, and constraints that we might use in our database

1. **Products Table:**
    - Product_ID (INT, Primary Key, NOT NULL): Unique identifier for products.
    - Product_Name (VARCHAR(50), NOT NULL): Descriptive name of the product.
2. **Inventory Table**:
    - Product_ID (INT, Foreign Key, NOT NULL): Unique identifier for products (links to the Products table).
    - Store_ID (INT, Primary Key, NOT NULL): Unique identifier for stores.

    - Stock_Level (INT, NOT NULL, CHECK >= 0): Current stock count of the product.
    - Stock_Status_Date (DATE, NOT NULL): Tracks the date of stock status updates.
    - Stock_Status (VARCHAR(20), NOT NULL): Describes the stock status (e.g., "In Stock", "Low Stock").
3. **Sales Transactions Table**:
    - Transaction_ID (INT, Primary Key, NOT NULL): Unique identifier for each sale.
    - Product_ID (INT, Foreign Key, NOT NULL): Links to the product in the Inventory table.
    - Store_ID (INT, Foreign Key, NOT NULL): Links to the store in the Inventory table.
    - Quantity_Sold (INT, NOT NULL, CHECK > 0): Number of units sold.
    - Sale_Date (DATE, NOT NULL): Date of the transaction.
4. **Suppliers Table**:
    - Supplier_ID (INT, Primary Key, NOT NULL): Unique identifier for suppliers.
    - Product_ID (INT, Foreign Key, NOT NULL): Links to the product in the Products table.
    - Lead_Time (INT, NOT NULL, CHECK >= 0): Days required for delivery.
    - Delivery_History (VARCHAR(125)): Tracks delivery performance data.

Relationships Between Tables

Understanding relationships between tables in a database is crucial for managing and analyzing data effectively. These connections, often made through shared fields, allow us to link related information across different tables. Such relationships enable a comprehensive view of the data, supporting better decision-making and operational management.

- Inventory and Sales Transactions tables are linked via Product_ID and Store_ID. This allows tracking of stock levels relative to sales.
- Inventory and Suppliers tables are linked via Product_ID, allowing insights into how supplier lead times affect stock levels and reorder efficiency.
- Products serves as the central table, linking to both Inventory and Suppliers, ensuring that every product tracked in inventory or supplied has a unique identity in the system.

Normalization Process

Normalization is a key process in database design that ensures data is organized efficiently, minimizes redundancy, and improves data integrity. For our inventory management system, we applied the following normalization steps:

- **First Normal Form (1NF):** Each table has been structured to ensure that every column contains atomic values, and there are no repeating groups. For instance, in the **Products** table, each product has a unique Product_ID and a single Product_Name, ensuring that no column contains multiple values.
- **Second Normal Form (2NF):** All non-primary key attributes are fully dependent on the primary key. In the **Inventory** table, attributes such as Stock_Level, Stock_Status, and Stock_Status_Date are dependent on both the Product_ID and Store_ID (composite key), ensuring that each product in each store is uniquely identified and tracked without redundancy.
- **Third Normal Form (3NF):** We ensured that no transitive dependencies exist, meaning non-key attributes are not dependent on other non-key attributes. For example, in the **Sales Transactions** table, attributes like Quantity_Sold and Sale_Date are directly related to the Transaction_ID and not dependent on other fields such as Product_ID or Store_ID. This ensures that data is not duplicated across the system.

By normalizing the database, we aim to:

- Eliminate redundant data and avoid inconsistencies across tables.
- Facilitate efficient data retrieval for queries such as tracking stock levels, analyzing supplier performance, and identifying sales trends across stores.

This approach helps ensure the database structure is optimized for the specific needs of the inventory management system, allowing for accurate and reliable operations as the dataset grows.

Database ER Diagram

This Entity-Relationship Diagram (ERD) models the structure of a retail inventory management system, illustrating the relationships between four key entities: Products, Inventory, Suppliers, and Sales Transactions. The Products table serves as a central entity, uniquely identifying each product in the system. The Inventory table tracks product stock levels across multiple stores, linking products to specific stores. The Sales Transactions table records each sale made, detailing the product, store, quantity sold, and date. The Suppliers table stores information about suppliers and their relationship to products, including lead times and delivery history. The relationships are designed to ensure data integrity and efficient management of product stock, supplier performance, and sales activities within the system.

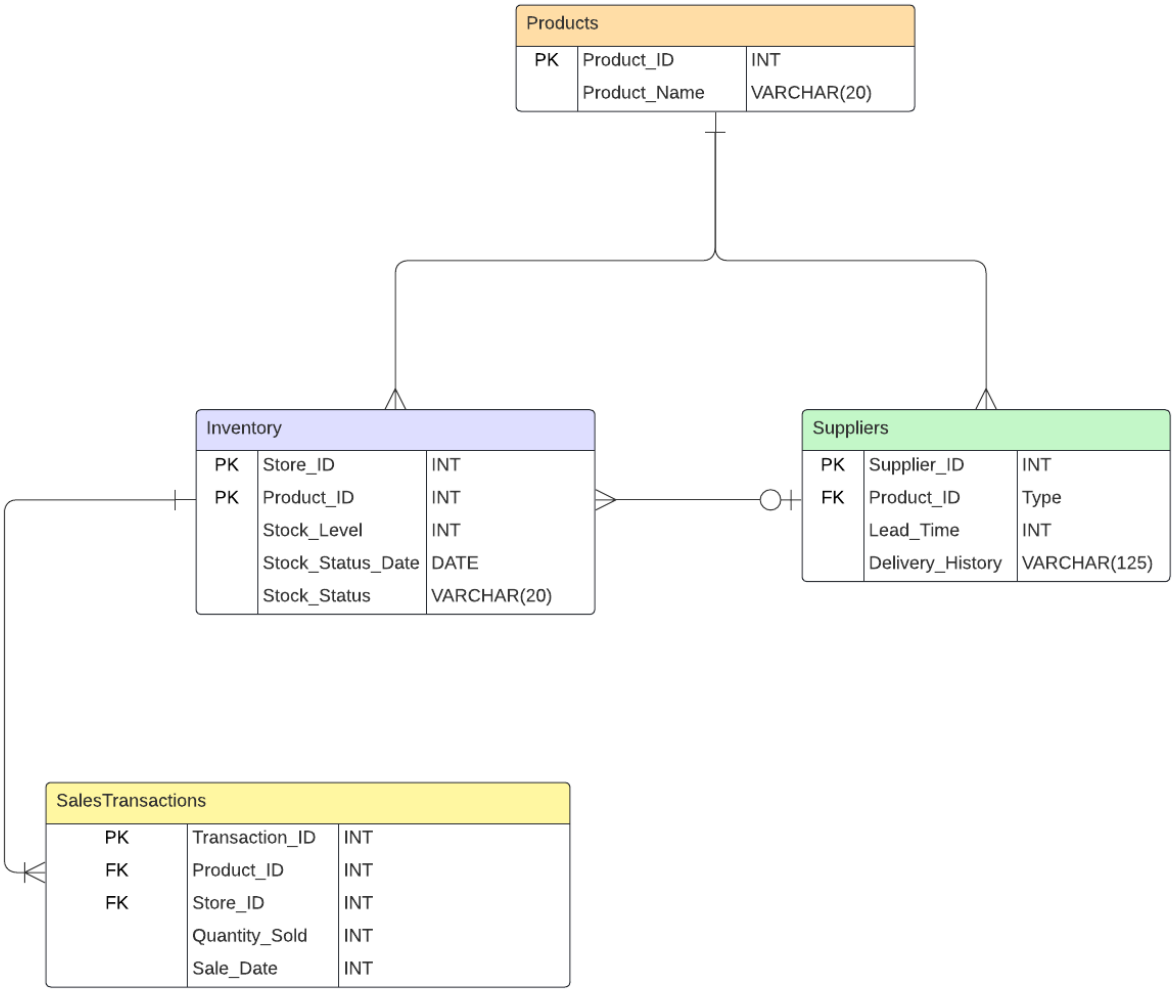

Table Structure

The following SQL script is designed to create and populate tables for a retail inventory management system. The database schema consists of four core tables: Products, Inventory, SalesTransactions, and Suppliers. These tables model the relationships between products, stock levels in stores, sales transactions, and suppliers. Each table is created with appropriate data types, primary keys, and foreign keys to ensure referential integrity. Following the table creation, a series of INSERT statements populate the tables with 50 rows of data, providing sufficient information to perform inventory tracking, demand forecasting, and supply chain analysis. These scripts are structured to be executed separately for ease of importing and initializing the database.