# Section 1. Database Schema Design:

**Entities:**

* **Companies**
* **Warehouses**
* **Products**
* **Inventory**
* **InventoryChangeLog**
* **Suppliers**
* **SupplierProducts**
* **ProductBundles**

**SQL Representation:**

1. **Companies:**

CREATE TABLE Companies (

id SERIAL PRIMARY KEY,

name VARCHAR(255) NOT NULL UNIQUE,

address TEXT,

created\_at TIMESTAMP DEFAULT CURRENT\_TIMESTAMP

);

1. **Warehouses:**

CREATE TABLE Warehouses (

id SERIAL PRIMARY KEY,

company\_id INT NOT NULL,

name VARCHAR(255) NOT NULL,

location TEXT,

created\_at TIMESTAMP DEFAULT CURRENT\_TIMESTAMP,

FOREIGN KEY (company\_id) REFERENCES Companies(id) ON DELETE CASCADE

);

1. **Products**:

CREATE TABLE Products (

id SERIAL PRIMARY KEY,

name VARCHAR(255) NOT NULL,

sku VARCHAR(100) NOT NULL UNIQUE,

price DECIMAL(10,2) NOT NULL,

is\_bundle BOOLEAN DEFAULT FALSE,

created\_at TIMESTAMP DEFAULT CURRENT\_TIMESTAMP

);

1. **Inventory**:

CREATE TABLE Inventory (

id SERIAL PRIMARY KEY,

product\_id INT NOT NULL,

warehouse\_id INT NOT NULL,

quantity INT DEFAULT 0,

last\_updated TIMESTAMP DEFAULT CURRENT\_TIMESTAMP,

UNIQUE(product\_id, warehouse\_id),

FOREIGN KEY (product\_id) REFERENCES Products(id) ON DELETE CASCADE,

FOREIGN KEY (warehouse\_id) REFERENCES Warehouses(id) ON DELETE CASCADE

);

1. **InventoryChangeLog**:

CREATE TABLE InventoryChangeLog (

id SERIAL PRIMARY KEY,

inventory\_id INT NOT NULL,

change\_amount INT NOT NULL,

change\_type VARCHAR(50) NOT NULL, -- 'IN', 'OUT', 'ADJUSTMENT'

reason TEXT,

created\_at TIMESTAMP DEFAULT CURRENT\_TIMESTAMP,

FOREIGN KEY (inventory\_id) REFERENCES Inventory(id) ON DELETE CASCADE

);

1. **Suppliers**:

CREATE TABLE Suppliers (

id SERIAL PRIMARY KEY,

name VARCHAR(255) NOT NULL UNIQUE,

contact\_info TEXT,

created\_at TIMESTAMP DEFAULT CURRENT\_TIMESTAMP

);

1. **SupplierProducts:**

CREATE TABLE SupplierProducts (

id SERIAL PRIMARY KEY,

supplier\_id INT NOT NULL,

product\_id INT NOT NULL,

supply\_price DECIMAL(10,2),

lead\_time\_days INT,

FOREIGN KEY (supplier\_id) REFERENCES Suppliers(id) ON DELETE CASCADE,

FOREIGN KEY (product\_id) REFERENCES Products(id) ON DELETE CASCADE,

UNIQUE(supplier\_id, product\_id)

);

1. **ProductBundles**:

CREATE TABLE ProductBundles (

id SERIAL PRIMARY KEY,

bundle\_id INT NOT NULL,

component\_id INT NOT NULL,

quantity INT DEFAULT 1,

FOREIGN KEY (bundle\_id) REFERENCES Products(id) ON DELETE CASCADE,

FOREIGN KEY (component\_id) REFERENCES Products(id) ON DELETE CASCADE,

UNIQUE(bundle\_id, component\_id)

);

**Relationships Explained:**

* Companies → Warehouses  
   One company can have many warehouses.
* Products → Warehouses (via Inventory)  
   A product can be stored in multiple warehouses with different quantities.
* Inventory → InventoryChangeLog  
   Tracks changes in inventory levels over time.
* Suppliers → Products (via SupplierProducts)  
   Suppliers provide products, potentially with different prices or lead times.
* Products → Bundles (via ProductBundles)  
   A product can be a bundle of other products.

**Identified Gaps – Questions for Product Team:**

**User Access & Permissions**

* Who manages warehouses, products, and inventory?
* Are there roles like admin, warehouse manager, or supplier manager?

**Inventory Operations**

* How should stock adjustments be tracked (manual adjustments, automated replenishments)?
* Do we need audit logs for who performed changes?

**Bundled Products**

* Can bundles be nested (bundles inside bundles)?
* How do we price bundles? Based on sum of components or separate pricing?

**Suppliers**

* Can multiple suppliers supply the same product at different rates?
* Are lead times mandatory or optional?

**Reporting**

* What metrics and reports are required (stock levels, turnover rates, reorder alerts)?

**Warehouses**

* Are warehouses always tied to companies or can they be shared?

**Data Volume**

* What’s the expected scale (number of products, warehouses, transactions)?

**Pricing**

* Is there support for tiered pricing, discounts, or taxes?

**Design Choices (Reason):**

* Normalization – Data redundancy is minimized by using join tables like Inventory, SupplierProducts, and ProductBundles.
* Flexibility – Supports products in multiple warehouses and complex product bundles.
* Data Integrity – Foreign keys ensure relationships are enforced.
* Scalability – Using indexes on key relationships allows queries to scale efficiently.
* Auditability – Inventory changes are tracked with timestamps and reasons.
* Extensibility – Additional features like pricing models or reporting dashboards can be built without schema redesign.