Part 2

1. Design Schema

Design a normalized relational schema using SQL DDL-style text notation. Here's a proposed structure:

```
CREATE TABLE Companies (
    company_id INT PRIMARY KEY,
    name VARCHAR(255) NOT NULL
);

CREATE TABLE Warehouses (
    warehouse_id INT PRIMARY KEY,
    company_id INT NOT NULL,
    location VARCHAR(255),
    FOREIGN KEY (company_id) REFERENCES Companies(company_id)
);

CREATE TABLE Suppliers (
    supplier_id INT PRIMARY KEY,
    name VARCHAR(255) NOT NULL
);

CREATE TABLE Products (
    product_id INT PRIMARY KEY,
    name VARCHAR(255) NOT NULL,
    is_bundle BOOLEAN DEFAULT FALSE
);
```

```
CREATE TABLE ProductBundles (

    bundle_id INT,

    product_id INT,

    quantity INT NOT NULL,

    PRIMARY KEY (bundle_id, product_id),

    FOREIGN KEY (bundle_id) REFERENCES Products(product_id),

    FOREIGN KEY (product_id) REFERENCES Products(product_id)

);


CREATE TABLE SupplierProducts (

    supplier_id INT,

    product_id INT,

    PRIMARY KEY (supplier_id, product_id),

    FOREIGN KEY (supplier_id) REFERENCES Suppliers(supplier_id),

    FOREIGN KEY (product_id) REFERENCES Products(product_id)

);


CREATE TABLE Inventory (

    warehouse_id INT,

    product_id INT,

    quantity INT NOT NULL,

    PRIMARY KEY (warehouse_id, product_id),

    FOREIGN KEY (warehouse_id) REFERENCES Warehouses(warehouse_id),

    FOREIGN KEY (product_id) REFERENCES Products(product_id)

);


CREATE TABLE InventoryChanges (

    change_id INT PRIMARY KEY,
    warehouse_id INT,
    product_id INT,
    quantity_change INT,
    change_timestamp TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
    FOREIGN KEY (warehouse_id) REFERENCES Warehouses(warehouse_id),
    FOREIGN KEY (product_id) REFERENCES Products(product_id)
);
```

2. Identify Gaps

Here are some clarifying questions you'd ask the product team:

1. Are inventory changes caused by specific events (e.g., sales, restocks)? Should we capture event types?
2. Do suppliers supply directly to warehouses or only to companies in general?
3. Can a bundle include other bundles (nested bundles)?
4. Should inventory reflect bundles or just individual product components?
5. Should we store pricing info (product cost from supplier, selling price, etc.)?
6. Do warehouses have capacity limits or other constraints (e.g., temperature-controlled)?
7. Do we need to track stockouts, reorder levels, or alerts for inventory thresholds?

3. Explain Decisions

- Indexes:
- Primary keys and foreign keys implicitly create indexes.
- Additional indexes could be added on frequently queried fields like product_id in Inventory or change_timestamp in InventoryChanges.

- Constraints: Foreign keys maintain referential integrity. Use of is_bundle in Products and a separate ProductBundles table avoids circular dependency.
- Data Normalisation :Schema is normalized to avoid redundancy (e.g., supplier-product links, bundle composition).
- Flexibility:Supports multiple suppliers per product, multiple warehouses per company, and bundle logic.