Normalization Process Documentation

Project: Pharmaceutical Inventory Management System (PIMS)
Objective: Ensure the relational database schema is normalized to improve data integrity, reduce redundancy, and support scalability.

🔷 Step 1: First Normal Form (1NF) – Eliminate Repeating Groups

Action Taken:

- Reviewed all attributes in each table to ensure atomicity.
- No columns contain lists, sets, or multi-valued fields.
- Each column stores only **one value per attribute**, per row (e.g., a drug has one batch number, one category, etc.).

Examples:

- Patient_Contact_Number stores a single contact number per patient.
- Drug_Name is a single atomic value (no combinations).
- Prescription_Drugs table ensures one drug per row per prescription.

Result:
All tables comply with 1NF. No repeating groups or non-atomic fields exist.

🔷 Step 2: Second Normal Form (2NF) – Remove Partial Dependencies

Action Taken:

- Verified that all tables in 1NF are also free from **partial dependencies**.
- Ensured that **composite keys**, if present, have non-key attributes fully dependent on the entire key.

Example:

- In the Prescription_Drugs table:
    - Prescription_Drugs_ID is used as the primary key.
    - Fields like Drugs_Drug_ID, Prescriptions_Prescription_ID, and Quantity are directly related to that key.
    - No non-key attribute is dependent on just a part of a composite key.

Result:
No partial dependencies. All non-key attributes are fully dependent on their primary key. The model is in 2NF.

🔷 Step 3: Third Normal Form (3NF) – Remove Transitive Dependencies

Action Taken:

- Analyzed all tables to detect **transitive dependencies** (i.e., non-key attributes depending on other non-key attributes).
- Ensured every non-primary-key column depends only on the primary key.

Examples:

- In the Drugs table:
  - Price, Batch_Number, and Stock_Quantity all depend only on Drug_ID.
  - No attribute like Price is dependent on Category_Name, avoiding transitive dependency.
- In the Users table:
  - Attributes like User_Email and User_Role directly depend on User_ID.

Result:
No transitive dependencies found. The model satisfies 3NF.

All tables in the PIMS database schema have been reviewed and validated to ensure they satisfy:

- **1NF**: No repeating groups or non-atomic fields
- **2NF**: No partial dependencies
- **3NF**: No transitive dependencies

As a result, the database model is **fully normalized** and adheres to the best practices of relational database design.

Business Rules, Validations, and Constraints

1. General Data Integrity Rules

Rule	Description
Email, First Name, and Last Name cannot be blank	Applies to Users, Patients, Doctors, and Suppliers. Enforced via NOT NULL and CHECK constraints.
Email must be unique	Ensures that no duplicate user, patient, doctor, or supplier accounts exist. Enforced via UNIQUE constraint.
Contact numbers must be valid (max 15 digits)	Enforced via VARCHAR(15) and proper validation.
Created_By and Updated_By must refer to valid User_IDs	Enforced through foreign key constraints referencing Users(User_ID) for audit tracking.
Timestamps must be auto-generated	Created_On and Updated_On should use default values and triggers.

---

2. Drug and Inventory Rules

Rule	Description
Drug stock quantity cannot be negative	Enforced via CHECK (Stock_Quantity >= 0) in the Drugs table.
Selling price cannot exceed MRP	If MRP is added, a CHECK (Price <= MRP) constraint should be enforced.
Batch Number must be unique for each drug	Ensures accurate tracking of expiry and supplier.
Expiry dates must be in the future at the time of insertion	Can be enforced via trigger or validation at the time of data entry.
Auto order placement if quantity falls below threshold	Can be implemented via scheduled PL/SQL job or trigger (future enhancement).

Category must be valid and from predefined list	Controlled via foreign key to Catagory(Catagory_ID) table.

3. Prescription and Medical Rules

Rule	Description
Each prescription must be linked to a valid doctor and patient	Enforced through foreign keys in Prescriptions table.
Each prescription must contain at least one drug	Ensured via presence of records in the Prescription_Drugs table.
Prescription drug quantity must be greater than zero	Enforced via CHECK (Quantity > 0) in the Prescription_Drugs table.
Expired drugs cannot be included in new prescriptions	Business logic to be implemented at the application layer or via a trigger.
Prescription date cannot be in the future	Enforced via a CHECK (Date_Issue <= SYSDATE) constraint.

4. Sales and Transaction Rules

Rule	Description
Sales quantity must be > 0 and ≤ available stock	CHECK (Quantity_Sold > 0) + trigger or procedure to validate against Stock_Quantity.
Each transaction must reference a valid drug, user, and payment method	Enforced through foreign keys.
Total price must match (unit price * quantity)	Validated via trigger or application logic.

Discounts must be applied before tax calculation	Business logic to be applied in billing procedures.
Only valid payment methods can be used	Enforced through foreign key to Payment_Method table.

---

5. Supplier and Procurement Rules

Rule	Description
Each drug must be linked to a supplier	Enforced via Suppliers_Supplier_ID foreign key in Drugs table.
Supplier email or phone must not be null	Enforced via NOT NULL on at least one contact column.
Suppliers must be created by an Admin	Can be enforced using triggers checking Created_By role.

---

6. User Roles and Access Control

Rule	Description
Only Admins can create/update/delete Users, Suppliers, Drugs	Enforced via application logic and/or stored procedures using Role.
Pharmacists can only manage prescriptions and sales	Role-based access control to be implemented at logic layer.
WHO columns must be maintained for audit trail	Created_On, Created_By, Updated_On, Updated_By to be auto-populated via triggers.

---

7. System and Automation Rules

Rule	Description
Auto-incrementing IDs	Implemented using sequences and triggers (e.g., Prescription_Drugs_ID).
Inventory logs must capture every stock change	All inserts/updates on Drugs table must result in a corresponding entry in Inventory_Logs.
Triggers must be rerunnable without errors	All triggers should be CREATE OR REPLACE, and scripts must handle drop-before-create.
Scripts must be idempotent	DDL scripts are designed to run multiple times without ORA errors using DROP IF EXISTS or exception-handling blocks.

The PIMS system enforces business rules and constraints through:

- **SQL constraints** (CHECK, NOT NULL, UNIQUE, FOREIGN KEY)
- **Data validations**
- **Triggers**
- **Procedures**
- **Role-based logic**

These rules ensure:

- Legal and regulatory compliance (drug expiry, prescription integrity)
- Operational efficiency (stock tracking, reorder automation)
- Data integrity and security (via WHO columns and role enforcement)