Mini project report on

Pharmacy Stock and Sales Management

Submitted in partial fulfilment of the requirements for the award of degree of

Bachelor of Technology

in

Computer Science & Engineering

UE23CS351A – DBMS Project

Submitted by:

B Neeharika	**PES2UG23CS114**
Bhoomika H B	**PES2UG23CS126**

under the guidance of

Dr. Gamini Joshi

Professor

PES University

AUG - DEC 2025

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

FACULTY OF ENGINEERING

PES UNIVERSITY

(Established under Karnataka Act No. 16 of 2013)

Electronic City, Hosur Road, Bengaluru – 560 100, Karnataka, India

PES UNIVERSITY

(Established under Karnataka Act No. 16 of 2013)
Electronic City, Hosur Road, Bengaluru – 560 100, Karnataka, India

CERTIFICATE

This is to certify that the mini project entitled

Pharmacy Stock and Sales Management

is a bonafide work carried out by

B Neeharika	**PES2UG23CS114**
Bhoomika H B	**PES2UG23CS126**

In partial fulfilment for the completion of fifth semester DBMS Project in the Program of Study -Bachelor of Technology in Computer Science and Engineering under rules and regulations of PES University, Bengaluru during the period AUG. 2025 – DEC. 2025. It is certified that all corrections / suggestions indicated for internal assessment have been incorporated in the report. The project has been approved as it satisfies the 5th semester academic requirements in respect of project work.

Signature

Dr. Gamini Joshi

DECLARATION

We hereby declare that the DBMS Project entitled **Pharmacy Stock and Sales Management** has been carried out by us under the guidance of **Dr. Gamini Joshi, Professor** and submitted in partial fulfilment of the course requirements for the award of degree of **Bachelor of Technology** in **Computer Science and Engineering** of **PES University, Bengaluru** during the academic semester AUG – DEC 2025.

B Neeharika	**PES2UG23CS114**
Bhoomika H B	**PES2UG23CS126**

ACKNOWLEDGEMENT

I would like to express my gratitude to Dr. Gamini Joshi, Professor, Department of Computer Science and Engineering, PES University, for her continuous guidance, assistance, and encouragement throughout the development of this DBMS Project.

I take this opportunity to thank Dr. Sandesh B J, C, Professor, ChairPerson, Department of Computer Science and Engineering, PES University, for all the knowledge and support I have received from the department.

I am deeply grateful to Dr. M. R. Doreswamy, Chancellor, PES University, Prof. Jawahar Doreswamy, Pro Chancellor – PES University, Dr. Suryaprasad J, Vice-Chancellor, PES University for providing to me various opportunities and enlightenment every step of the way. Finally, this DBMS Project could not have been completed without the continual support and encouragement I have received from my family and friends.

ABSTRACT

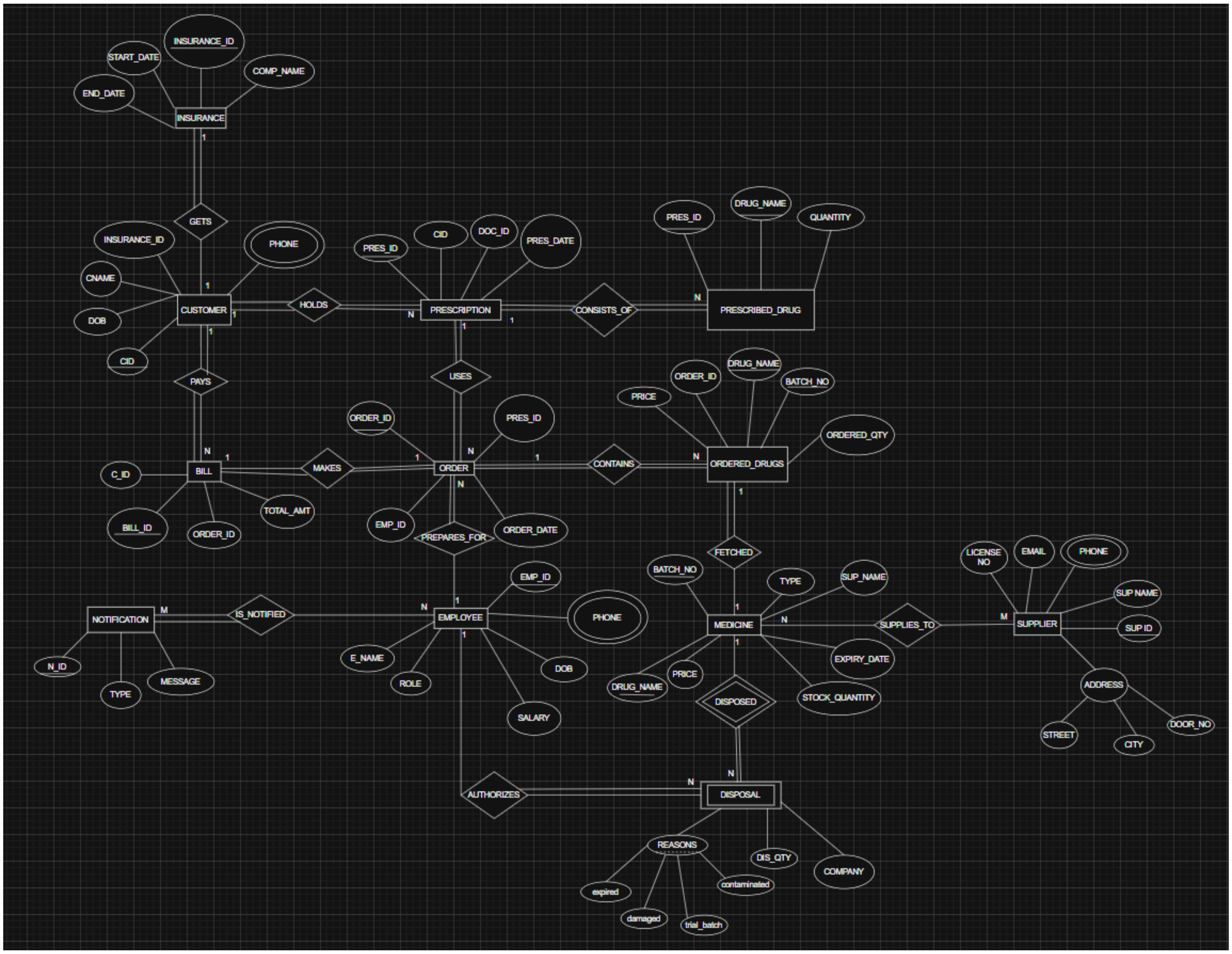
The **Pharmacy Stock and Sales Management System** is a comprehensive desktop application designed to automate and streamline pharmacy operations, ensuring efficient management of medicines, suppliers, customers, orders, prescriptions, and billing. Developed using **Python Tkinter** for the frontend and **MySQL** for the backend, the system incorporates a robust **Role-Based Access Control (RBAC)** mechanism, supporting five user roles—Admin, Supervisor, Manager, Pharmacist, and Cashier—with granular permissions for secure and controlled access.

The system provides real-time **inventory management**, tracking medicines by batch numbers, monitoring expiry dates, and generating automatic notifications for low stock or near-expiry items. Advanced database features, including **triggers, stored procedures, and user-defined functions**, automate critical operations such as stock updates, order processing, and bill generation. Triggers prevent the sale of expired medicines and insufficient stock, while functions calculate total stock value and check for expired batches.

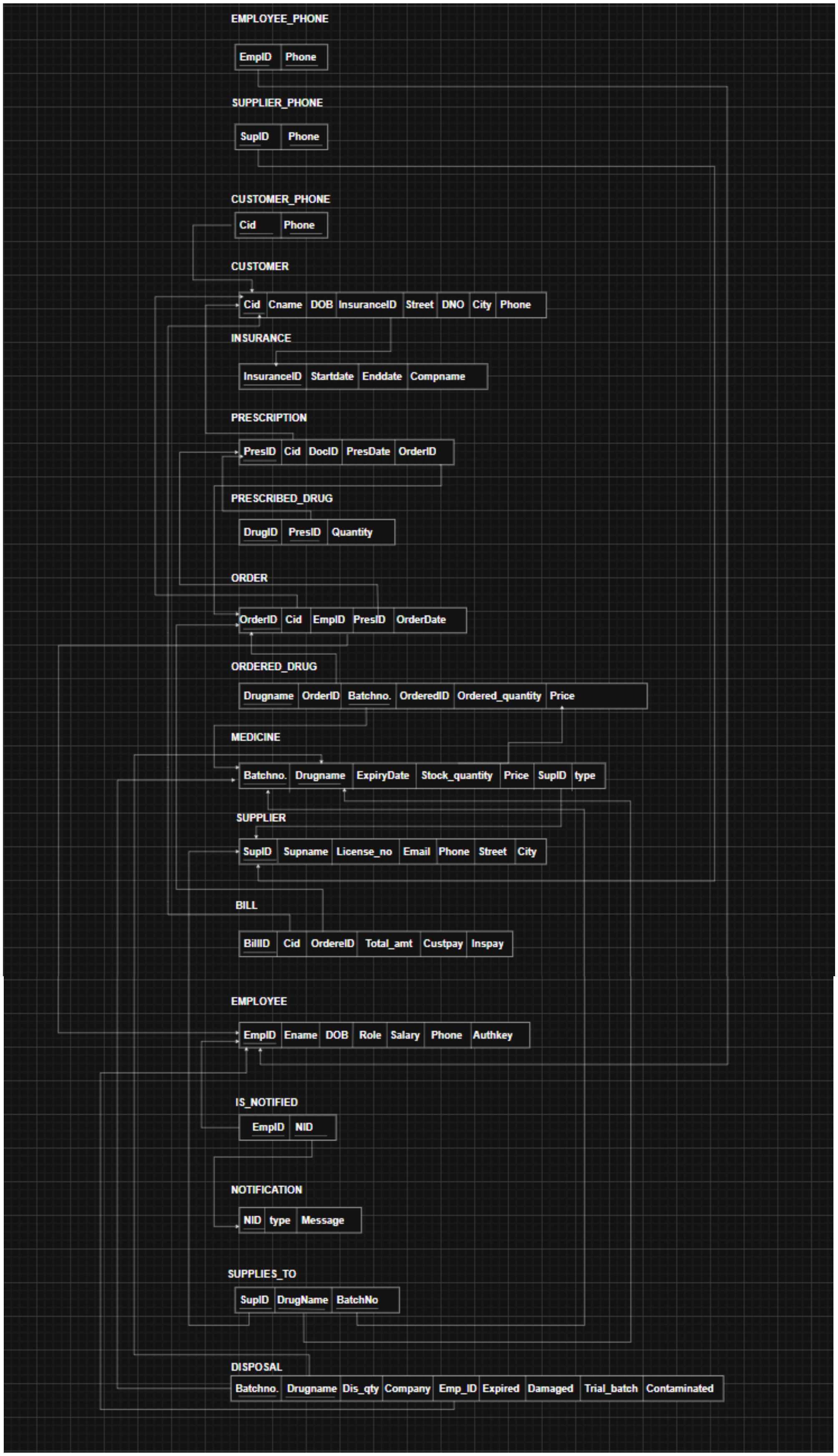
Customer and order management is fully integrated, supporting multi-drug prescriptions linked to doctor records, order tracking, and automatic notifications. The **billing system** generates accurate bills reflecting customer and insurance payments. The application also provides analytical queries and reports, including stock valuation, expiry tracking, insurance tracking, and sales history.

By combining **secure access control, real-time inventory monitoring, automated workflows, and advanced reporting**, this system reduces manual errors, ensures regulatory compliance, and enhances operational efficiency. It serves as a complete solution for modern pharmacies seeking to maintain accurate stock, streamline sales processes, and improve overall service delivery.

ER DIAGRAM:


INSURANCE_ID
START_DATE
COMP_NAME
END_DATE
INSURANCE
1
GETS
INSURANCE_ID
PHONE
CNAME
1
CUSTOMER
1
HOLDS
N
PRESCRIPTION
1
DOB
CID
1
PAYS
PRES_ID
CID
DOC_ID
PRES_DATE
1
CONSISTS_OF
N
PRESCRIBED_DRUG
PRES_ID
DRUG_NAME
QUANTITY
USES
ORDER_ID
PRES_ID
N
1
BILL
MAKES
1
N
ORDER
1
CONTAINS
N
ORDERED_DRUGS
1
C_ID
BILL_ID
ORDER_ID
TOTAL_AMT
N
EMP_ID
PREPARES_FOR
ORDER_DATE
PRICE
ORDER_ID
DRUG_NAME
BATCH_NO
ORDERED_QTY
FETCHED
EMP_ID
1
NOTIFICATION
M
IS_NOTIFIED
N
EMPLOYEE
1
PHONE
N_ID
TYPE
MESSAGE
E_NAME
ROLE
DOB
SALARY
BATCH_NO
TYPE
SUP_NAME
1
MEDICINE
N
1
SUPPLIES_TO
M
SUPPLIER
LICENSE NO
EMAIL
PHONE
SUP NAME
SUP ID
ADDRESS
STREET
CITY
DOOR_NO
EXPIRY_DATE
PRICE
DRUG_NAME
STOCK_QUANTITY
DISPOSED
N
AUTHORIZES
N
DISPOSAL
REASONS
DIS_QTY
COMPANY
expired
damaged
trial_batch
contaminated

RELATIONAL SCHEMA



EMPLOYEE_PHONE

EmpID	Phone

SUPPLIER_PHONE

SupID	Phone

CUSTOMER_PHONE

Cid	Phone

CUSTOMER

Cid	Cname	DOB	InsuranceID	Street	DNO	City	Phone

INSURANCE

InsuranceID	Startdate	Enddate	Compname

PRESCRIPTION

PresID	Cid	DocID	PresDate	OrderID

PRESCRIBED_DRUG

DrugID	PresID	Quantity

ORDER

OrderID	Cid	EmpID	PresID	OrderDate

ORDERED_DRUG

Drugname	OrderID	Batchno.	OrderedID	Ordered_quantity	Price

MEDICINE

Batchno.	Drugname	ExpiryDate	Stock_quantity	Price	SupID	type

SUPPLIER

SupID	Supname	License_no	Email	Phone	Street	City

BILL

BillID	Cid	OrdereID	Total_amt	Custpay	Inspay

EMPLOYEE

EmpID	Ename	DOB	Role	Salary	Phone	Authkey

IS_NOTIFIED

EmpID	NID

NOTIFICATION

NID	type	Message

SUPPLIES_TO

SupID	DrugName	BatchNo

DISPOSAL

Batchno.	Drugname	Dis_qty	Company	Emp_ID	Expired	Damaged	Trial_batch	Contaminated

User Requirement Specification

(A) Functional Requirements

- **User Authentication:**
  - Staff must be able to create a new account (Sign Up).
  - Staff must be able to log in with their Staff ID and Password.
  - The system must validate credentials against the EMPLOYEE table.
- **Role-Based Access Control (RBAC):**
  - The system must restrict access to features based on staff roles.
  - Admin: Full access (View, Add, Edit, Delete, Employee Management, Analytics).
  - Supervisor: Access to View, Add, Edit, Delete, Analytics.
  - Manager: Access to View, Add, Edit, Delete, Employee Management.
  - Pharmacist: Access to View, Add, Edit (limited), Stock Management.
  - Cashier: Access to View and Add only (limited billing and order creation).
- **Medicine Management (Pharmacist, Supervisor, Admin):**
  - Track medicines with batch numbers, type, supplier, and price.
  - Monitor expiry dates with automatic notifications.
  - Manage stock quantities and update on sales.
  - Track disposal of expired, damaged, trial, or contaminated medicines.
- **Customer & Order Management (Cashier, Pharmacist, Supervisor, Admin):**
  - Staff must be able to register new customers.
  - The system must prevent duplicate customers based on Phone or Customer ID.
  - Staff must be able to update or delete customer records.
  - Staff must be able to create orders linked to customers and prescriptions.
  - Multi-drug prescriptions must be supported, with automatic link to orders.
- **Billing System (Cashier, Supervisor, Admin):**
  - The system must generate bills for customer orders, including insurance payments.
  - Bills must reflect correct pricing and quantities.
  - Billing calculations may be automated via stored procedures.
- **Database Automation (Triggers, Functions & Procedures)**

The system uses database automation to ensure accurate stock management, prevent errors, and streamline operations.

Triggers:

- trg_reduce_stock – Automatically reduces medicine stock after an order is placed.
- trg_check_stock_before_order – Prevents orders exceeding available stock.
- trg_block_expired – Blocks sale of expired medicines.

Functions:

- TotalStockValue() – Returns the total monetary value of all medicines in stock.
- IsExpired(batch, name) – Checks whether a given medicine batch has expired.

Stored Procedures:

- AddMedicine – Adds a new medicine batch to the inventory.
- CreateOrder – Creates a new customer order.
- GenerateBill – Calculates total bill and inserts a record into the BILL table.

Purpose:

These automated routines ensure real-time inventory updates, accurate billing, and enforcement of business rules, reducing manual effort and errors.

- **Notifications & Alerts:**
  - The system must notify employees about low stock and near-expiry medicines.
  - Prescription alerts and order updates must be recorded in the IS_NOTIFIED table.
  - Notifications can be marked as seen to track acknowledgment.
- **Analytics & Reports (Supervisor, Admin):**
  - The system must provide stock valuation reports.
  - Expiry tracking and insurance coverage reports must be generated.
  - Historical sales and order reports must be viewable.
  - Custom SQL query execution for advanced analysis must be supported.

(B) Non-Functional Requirements

- **Security:**
  - Role-based access ensures only authorized staff can access sensitive information.
  - Database access must be secured with MySQL user roles and privileges.
- **Data Integrity:**
  - Primary Key, Foreign Key, UNIQUE, and NOT NULL constraints must enforce consistent and valid data.
  - Triggers and procedures must maintain automatic data consistency during stock updates and sales.
- **Usability:**
  - The system must have a tab-based, intuitive GUI for non-technical pharmacy staff.
  - Alerts and notifications must be visually clear and easy to understand.
- **Reliability:**
  - Database operations must be transactional to avoid partial updates.

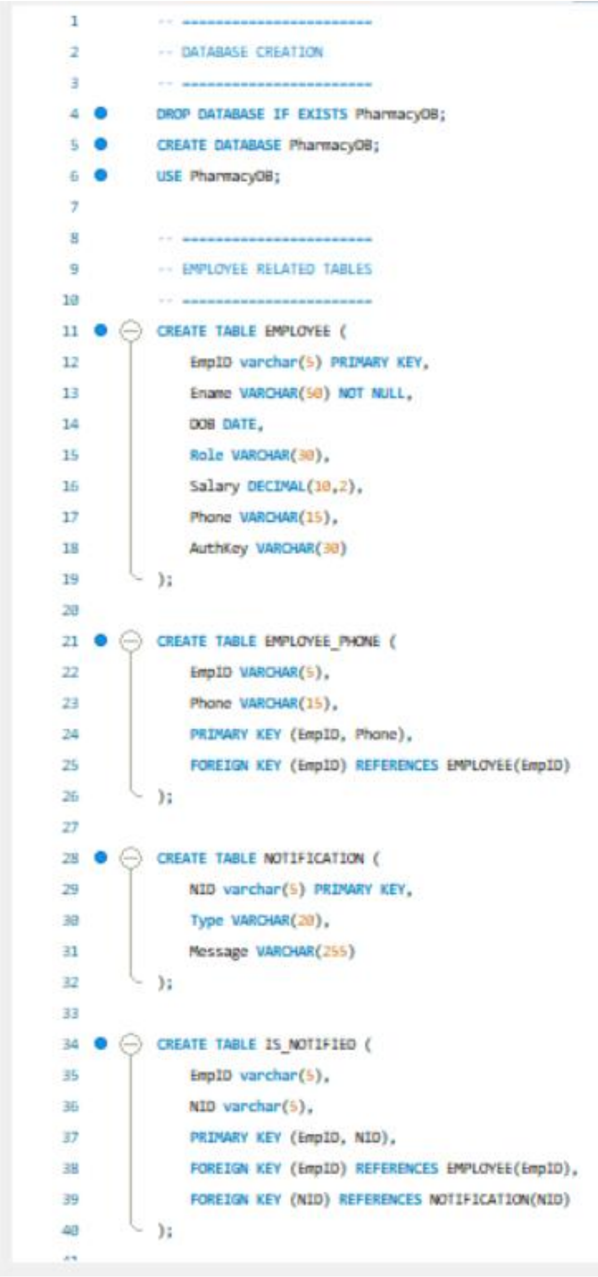
    - Error handling in triggers and procedures must prevent invalid orders or stock inconsistencies.

Software/Tools/Programming Languages Used

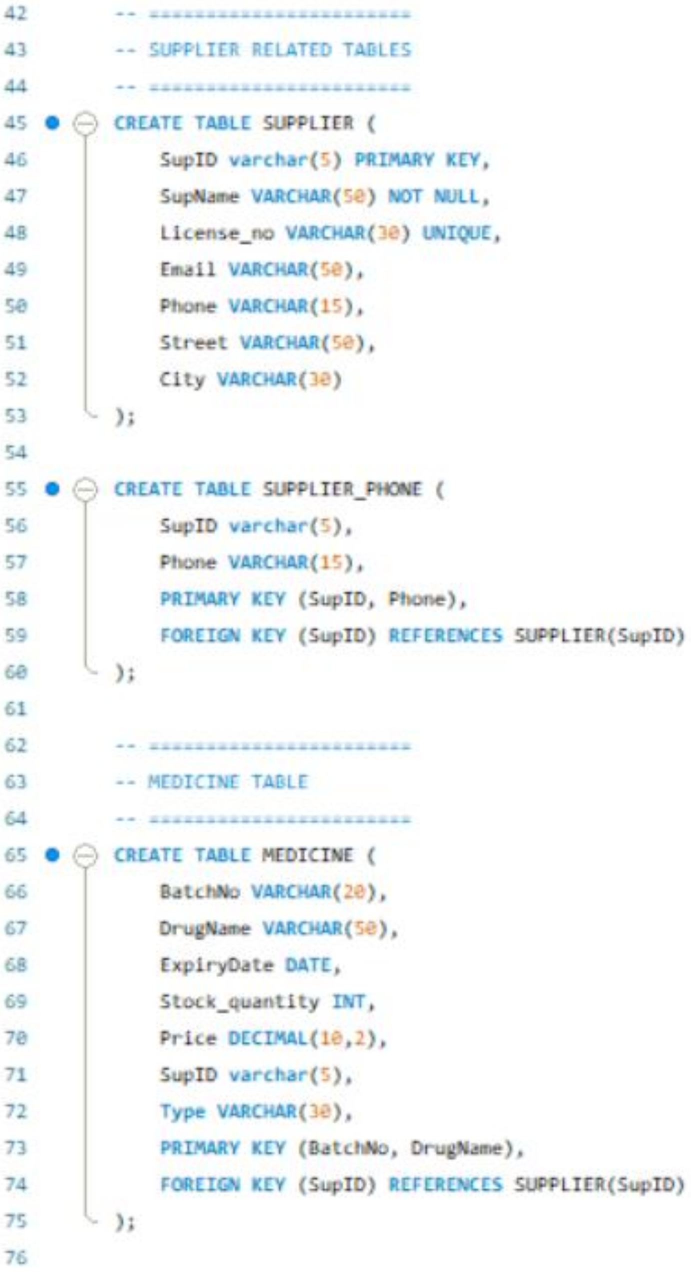
- **Programming Language:** Python 3.7+
- **Database:** MySQL Workbench
- **GUI Library:** Tkinter (Python's standard GUI package)
- **Analytics & Reporting:** Custom SQL queries, optional integration with Matplotlib for charts
- **Database Connector:** mysql-connector-python
- **Version Control:** Git and GitHub
- **IDE:** VS Code (or any Python-compatible IDE)
- **Database Management:** MySQL Workbench (or any SQL client)
- **Design Pattern:** MVC-inspired architecture for clean separation of GUI, business logic, and database operations

BACKEND:

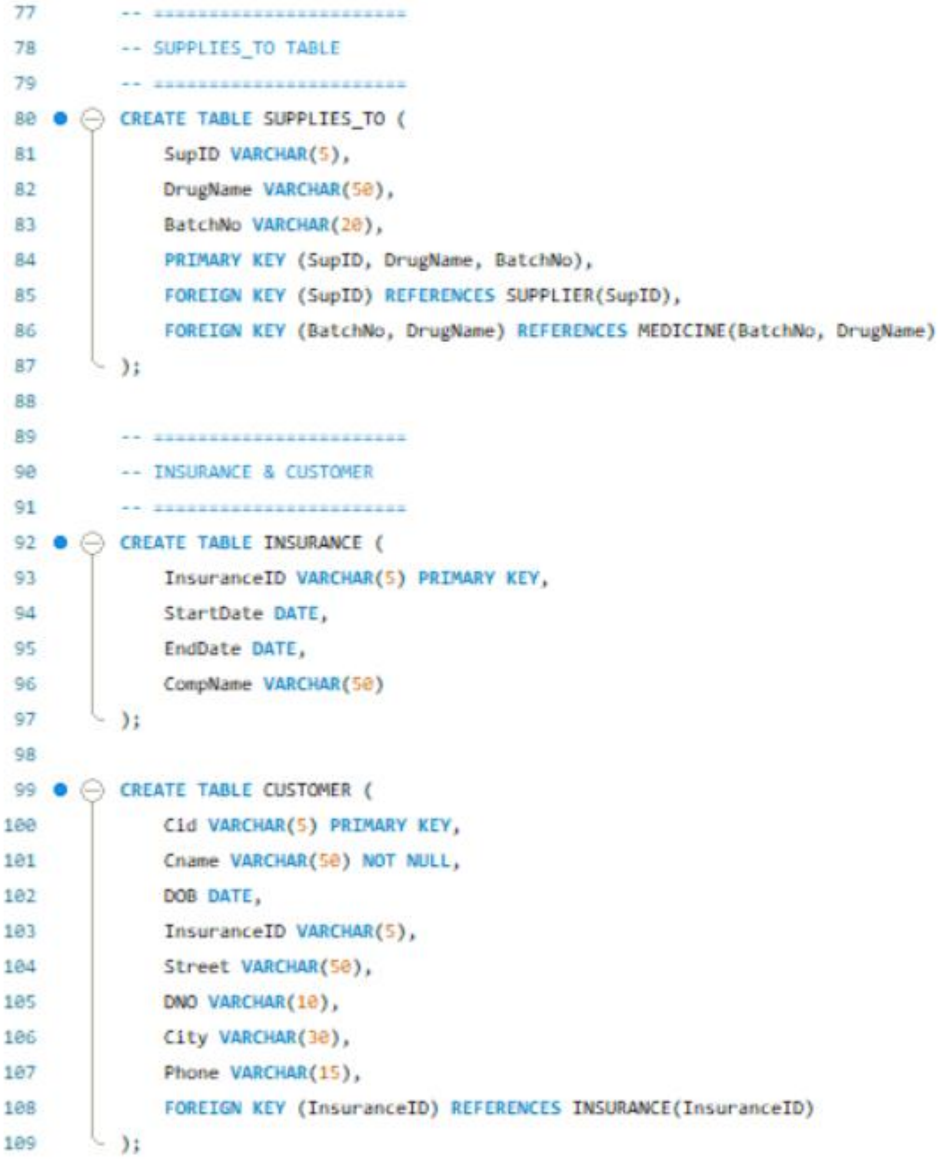
DDL COMMANDS:



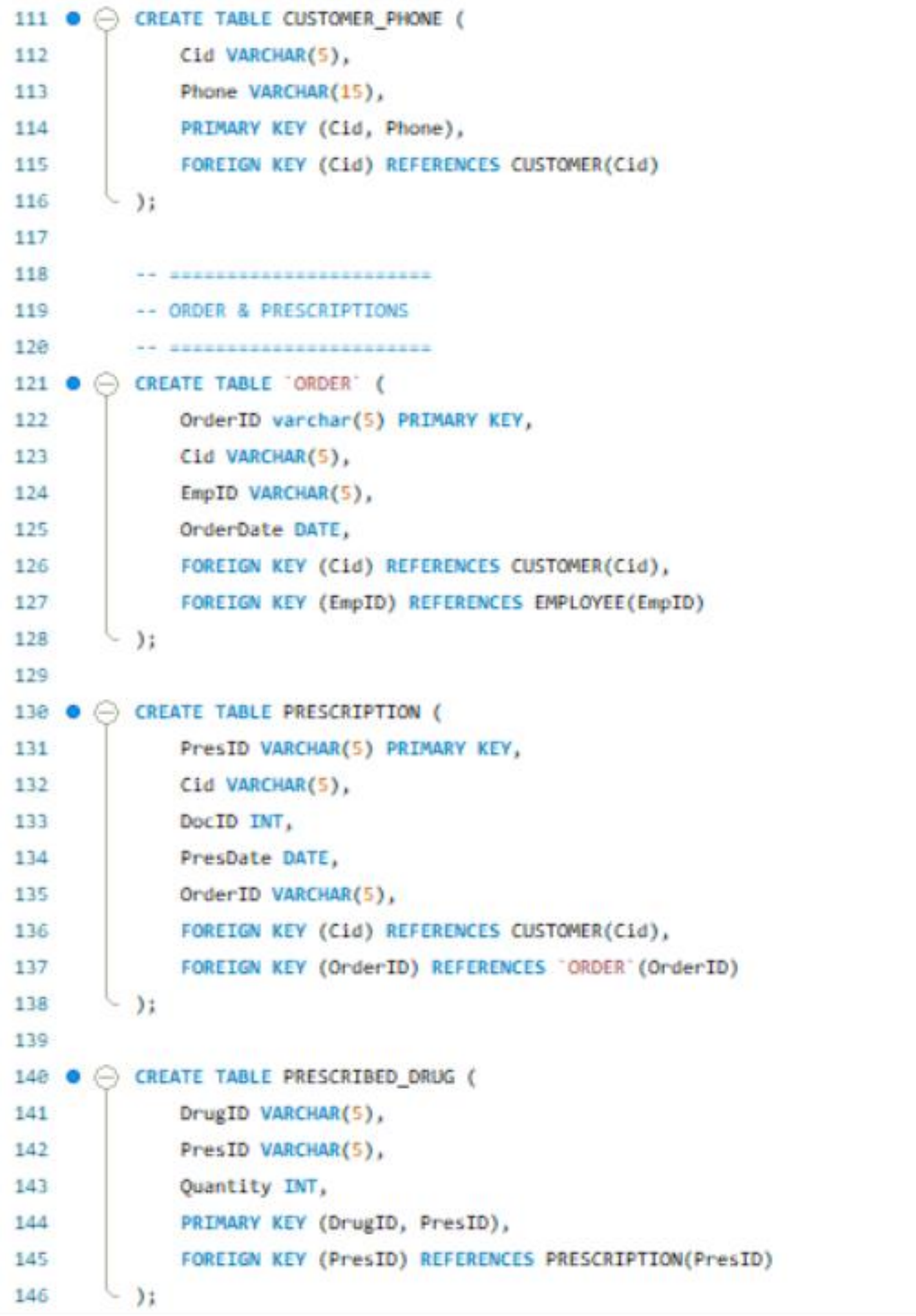
```sql
-- ========================
-- DATABASE CREATION
-- ========================
DROP DATABASE IF EXISTS PharmacyDB;
CREATE DATABASE PharmacyDB;
USE PharmacyDB;

-- ========================
-- EMPLOYEE RELATED TABLES
-- ========================
CREATE TABLE EMPLOYEE (
    EmpID varchar(5) PRIMARY KEY,
    Ename VARCHAR(50) NOT NULL,
    DOB DATE,
    Role VARCHAR(30),
    Salary DECIMAL(10,2),
    Phone VARCHAR(15),
    AuthKey VARCHAR(30)
);

CREATE TABLE EMPLOYEE_PHONE (
    EmpID VARCHAR(5),
    Phone VARCHAR(15),
    PRIMARY KEY (EmpID, Phone),
    FOREIGN KEY (EmpID) REFERENCES EMPLOYEE(EmpID)
);

CREATE TABLE NOTIFICATION (
    NID varchar(5) PRIMARY KEY,
    Type VARCHAR(20),
    Message VARCHAR(255)
);

CREATE TABLE IS_NOTIFIED (
    EmpID varchar(5),
    NID varchar(5),
    PRIMARY KEY (EmpID, NID),
    FOREIGN KEY (EmpID) REFERENCES EMPLOYEE(EmpID),
    FOREIGN KEY (NID) REFERENCES NOTIFICATION(NID)
);
```



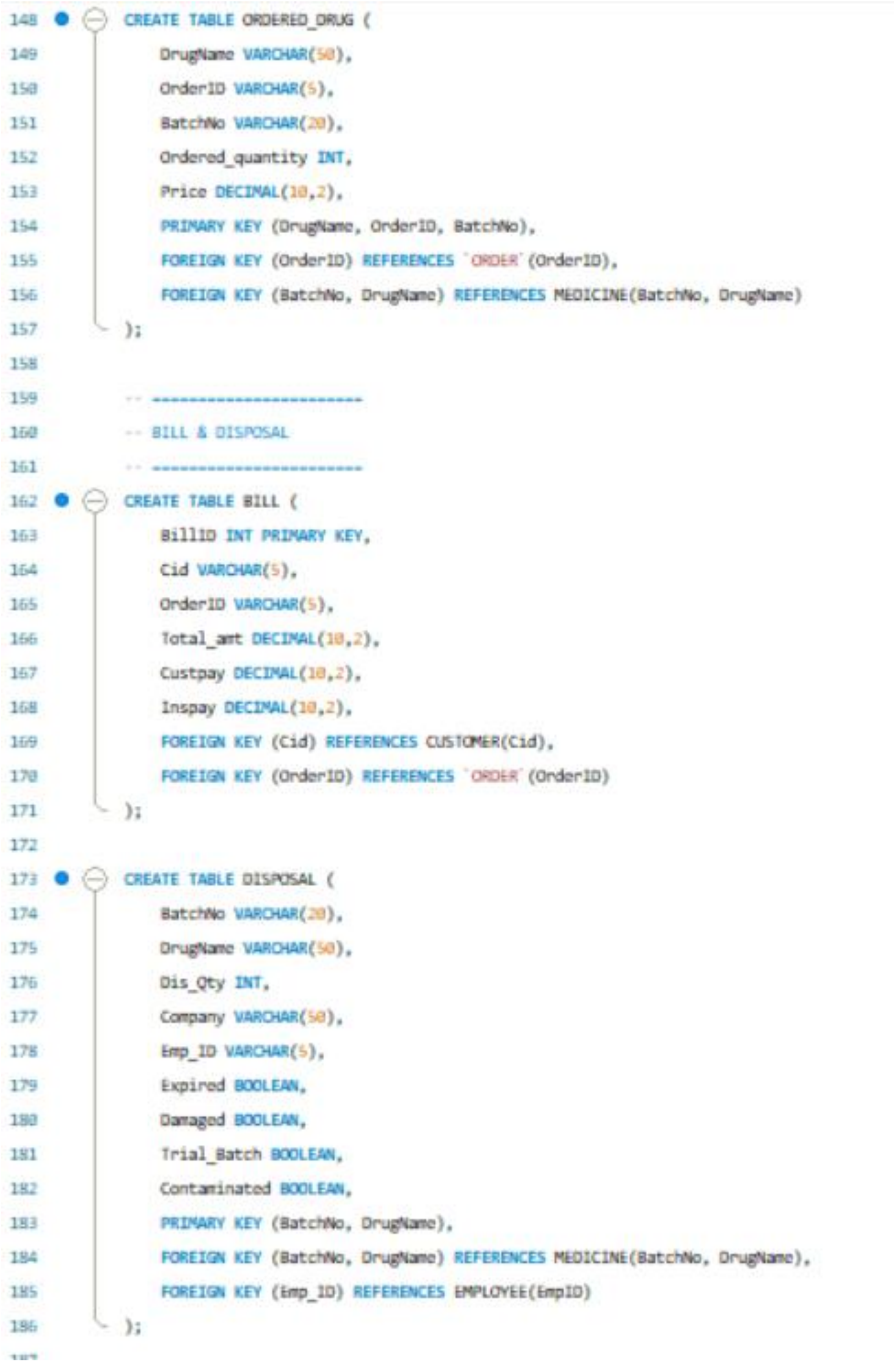
```sql
-- ========================
-- SUPPLIER RELATED TABLES
-- ========================
CREATE TABLE SUPPLIER (
    SupID varchar(5) PRIMARY KEY,
    SupName VARCHAR(50) NOT NULL,
    License_no VARCHAR(30) UNIQUE,
    Email VARCHAR(50),
    Phone VARCHAR(15),
    Street VARCHAR(50),
    City VARCHAR(30)
);

CREATE TABLE SUPPLIER_PHONE (
    SupID varchar(5),
    Phone VARCHAR(15),
    PRIMARY KEY (SupID, Phone),
    FOREIGN KEY (SupID) REFERENCES SUPPLIER(SupID)
);

-- ========================
-- MEDICINE TABLE
-- ========================
CREATE TABLE MEDICINE (
    BatchNo VARCHAR(20),
    DrugName VARCHAR(50),
    ExpiryDate DATE,
    Stock_quantity INT,
    Price DECIMAL(10,2),
    SupID varchar(5),
    Type VARCHAR(30),
    PRIMARY KEY (BatchNo, DrugName),
    FOREIGN KEY (SupID) REFERENCES SUPPLIER(SupID)
);
```



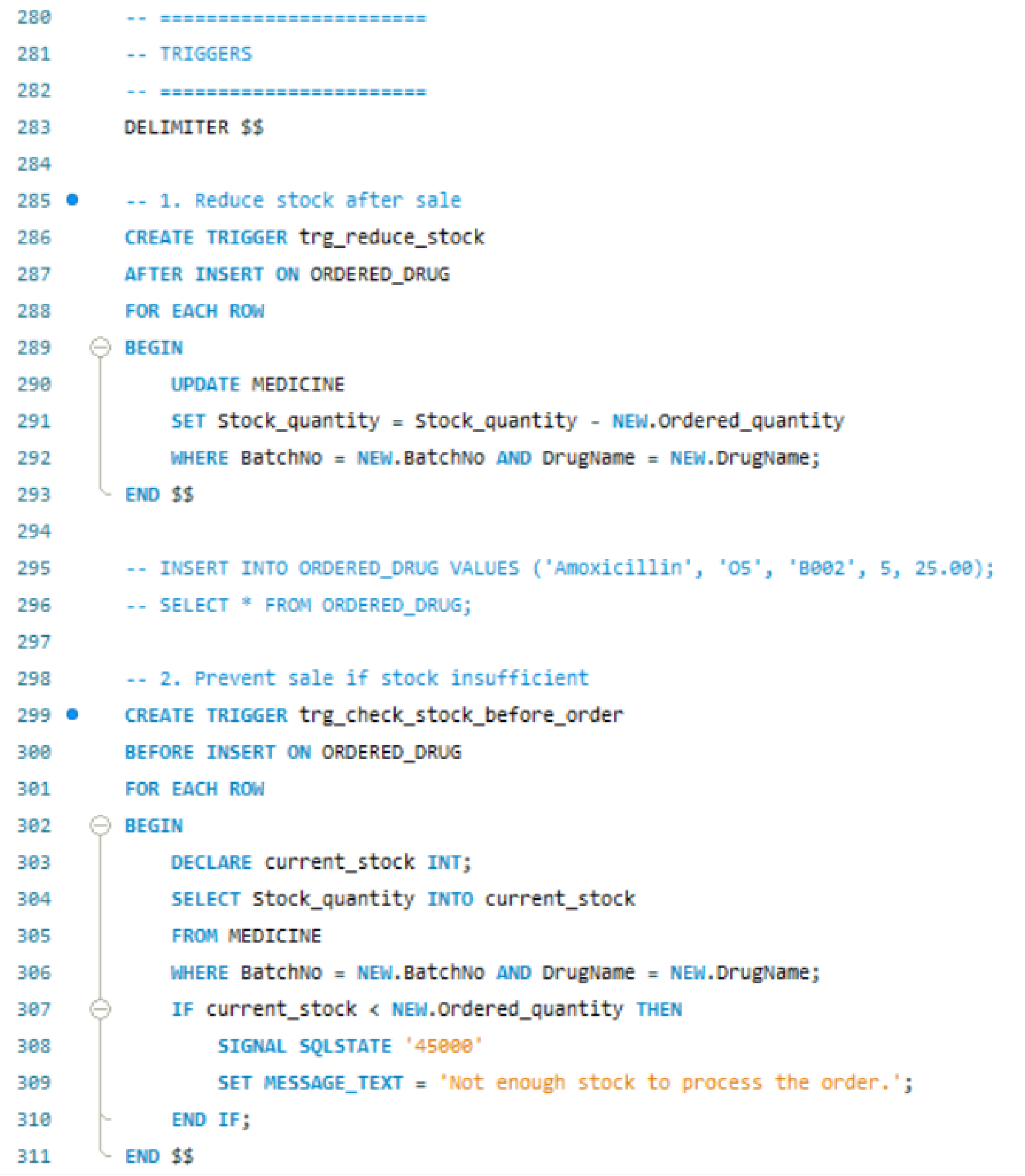
```sql
-- ========================
-- SUPPLIES_TO TABLE
-- ========================
CREATE TABLE SUPPLIES_TO (
    SupID VARCHAR(5),
    DrugName VARCHAR(50),
    BatchNo VARCHAR(20),
    PRIMARY KEY (SupID, DrugName, BatchNo),
    FOREIGN KEY (SupID) REFERENCES SUPPLIER(SupID),
    FOREIGN KEY (BatchNo, DrugName) REFERENCES MEDICINE(BatchNo, DrugName)
);

-- ========================
-- INSURANCE & CUSTOMER
-- ========================
CREATE TABLE INSURANCE (
    InsuranceID VARCHAR(5) PRIMARY KEY,
    StartDate DATE,
    EndDate DATE,
    CompName VARCHAR(50)
);

CREATE TABLE CUSTOMER (
    Cid VARCHAR(5) PRIMARY KEY,
    Cname VARCHAR(50) NOT NULL,
    DOB DATE,
    InsuranceID VARCHAR(5),
    Street VARCHAR(50),
    DNO VARCHAR(10),
    City VARCHAR(30),
    Phone VARCHAR(15),
    FOREIGN KEY (InsuranceID) REFERENCES INSURANCE(InsuranceID)
);
```



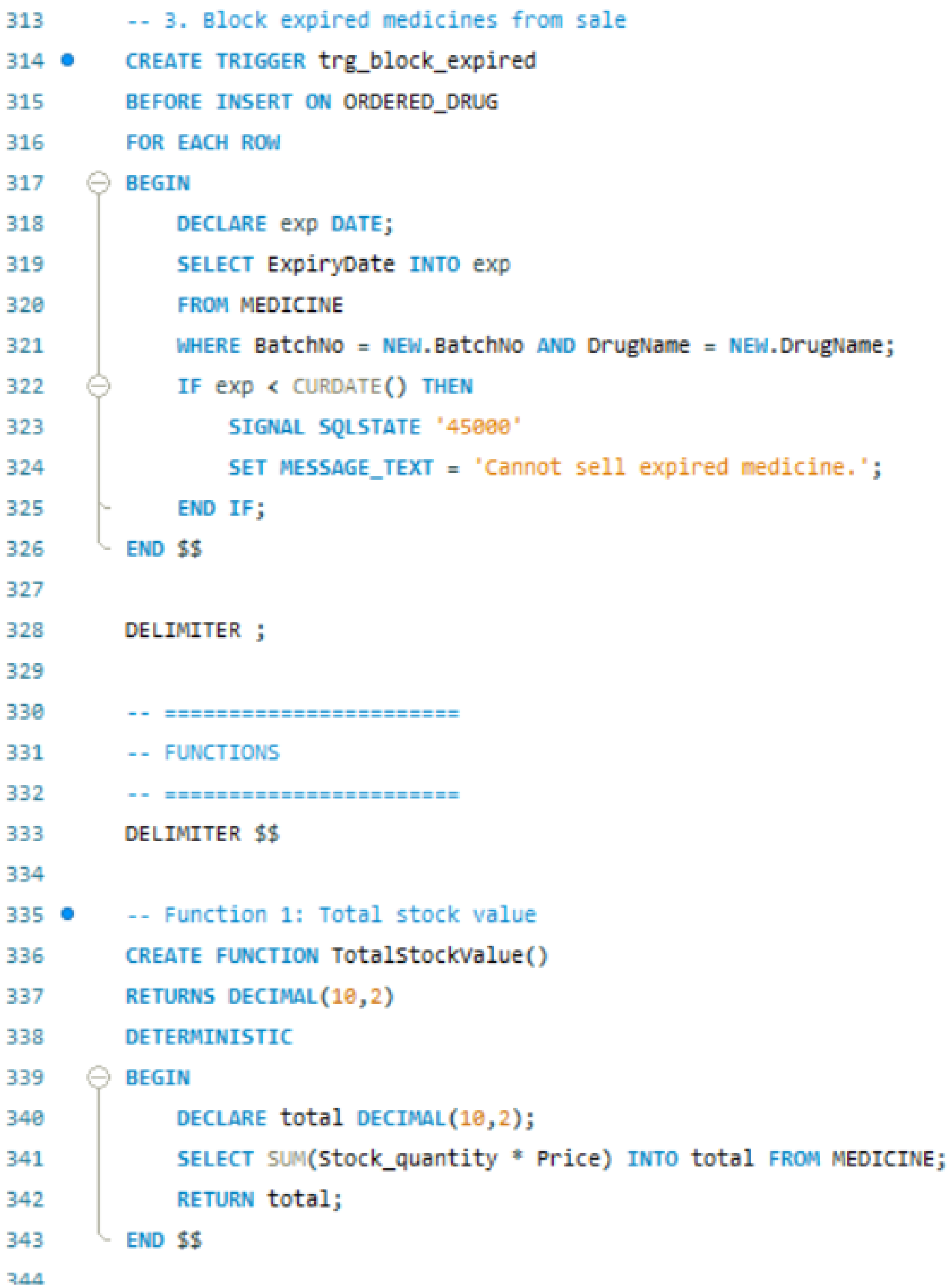
```sql
CREATE TABLE CUSTOMER_PHONE (
    Cid VARCHAR(5),
    Phone VARCHAR(15),
    PRIMARY KEY (Cid, Phone),
    FOREIGN KEY (Cid) REFERENCES CUSTOMER(Cid)
);

-- =======================
-- ORDER & PRESCRIPTIONS
-- =======================
CREATE TABLE `ORDER` (
    OrderID varchar(5) PRIMARY KEY,
    Cid VARCHAR(5),
    EmpID VARCHAR(5),
    OrderDate DATE,
    FOREIGN KEY (Cid) REFERENCES CUSTOMER(Cid),
    FOREIGN KEY (EmpID) REFERENCES EMPLOYEE(EmpID)
);

CREATE TABLE PRESCRIPTION (
    PresID VARCHAR(5) PRIMARY KEY,
    Cid VARCHAR(5),
    DocID INT,
    PresDate DATE,
    OrderID VARCHAR(5),
    FOREIGN KEY (Cid) REFERENCES CUSTOMER(Cid),
    FOREIGN KEY (OrderID) REFERENCES `ORDER`(OrderID)
);

CREATE TABLE PRESCRIBED_DRUG (
    DrugID VARCHAR(5),
    PresID VARCHAR(5),
    Quantity INT,
    PRIMARY KEY (DrugID, PresID),
    FOREIGN KEY (PresID) REFERENCES PRESCRIPTION(PresID)
);
```



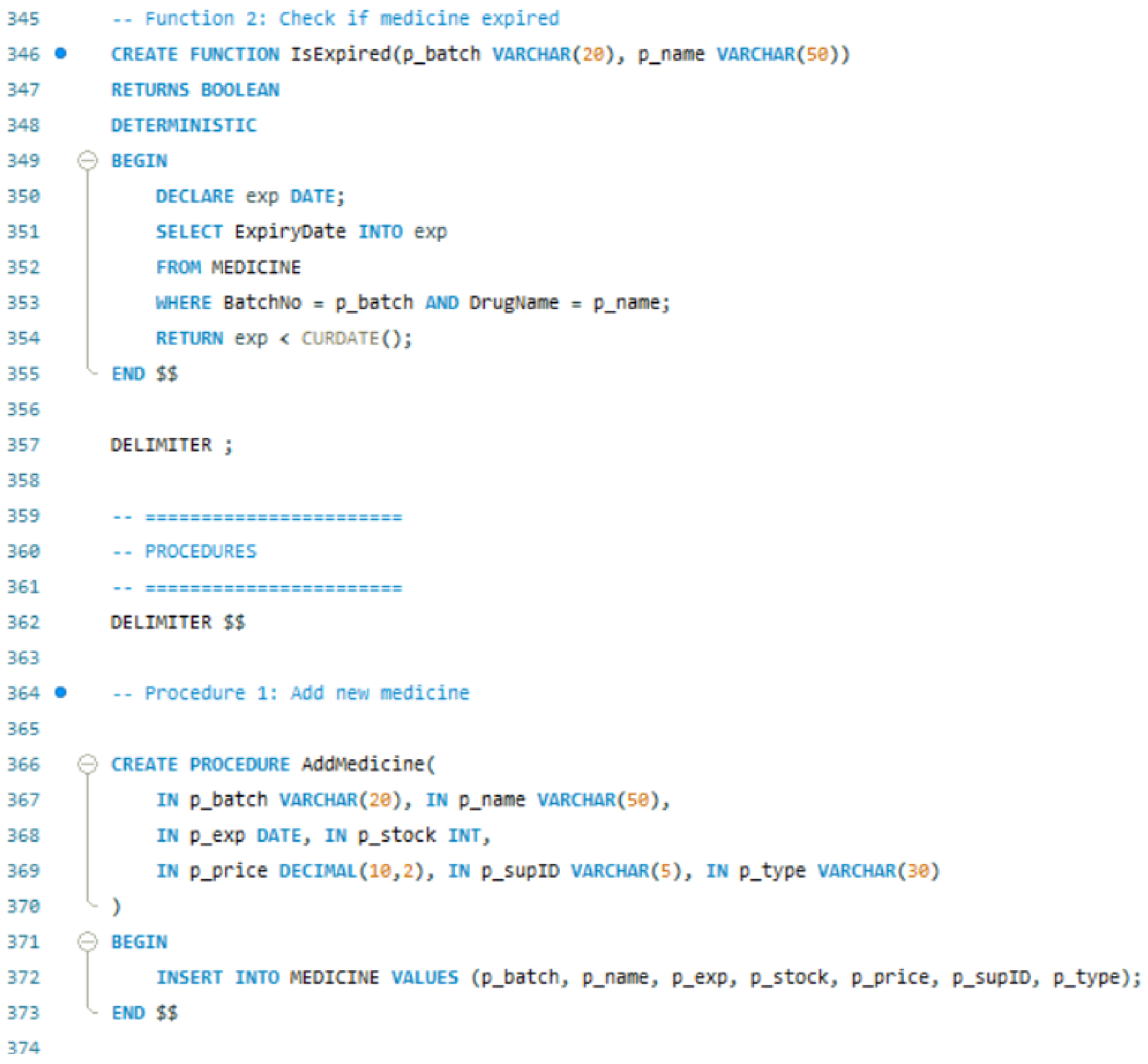
```
CREATE TABLE ORDERED_DRUG (
    DrugName VARCHAR(50),
    OrderID VARCHAR(5),
    BatchNo VARCHAR(20),
    Ordered_quantity INT,
    Price DECIMAL(10,2),
    PRIMARY KEY (DrugName, OrderID, BatchNo),
    FOREIGN KEY (OrderID) REFERENCES `ORDER` (OrderID),
    FOREIGN KEY (BatchNo, DrugName) REFERENCES MEDICINE(BatchNo, DrugName)
);

-- -----------------------
-- BILL & DISPOSAL
-- -----------------------
CREATE TABLE BILL (
    BillID INT PRIMARY KEY,
    Cid VARCHAR(5),
    OrderID VARCHAR(5),
    Total_amt DECIMAL(10,2),
    Custpay DECIMAL(10,2),
    Inspay DECIMAL(10,2),
    FOREIGN KEY (Cid) REFERENCES CUSTOMER(Cid),
    FOREIGN KEY (OrderID) REFERENCES `ORDER` (OrderID)
);

CREATE TABLE DISPOSAL (
    BatchNo VARCHAR(20),
    DrugName VARCHAR(50),
    Dis_Qty INT,
    Company VARCHAR(50),
    Emp_ID VARCHAR(5),
    Expired BOOLEAN,
    Damaged BOOLEAN,
    Trial_Batch BOOLEAN,
    Contaminated BOOLEAN,
    PRIMARY KEY (BatchNo, DrugName),
    FOREIGN KEY (BatchNo, DrugName) REFERENCES MEDICINE(BatchNo, DrugName),
    FOREIGN KEY (Emp_ID) REFERENCES EMPLOYEE(EmpID)
);
```



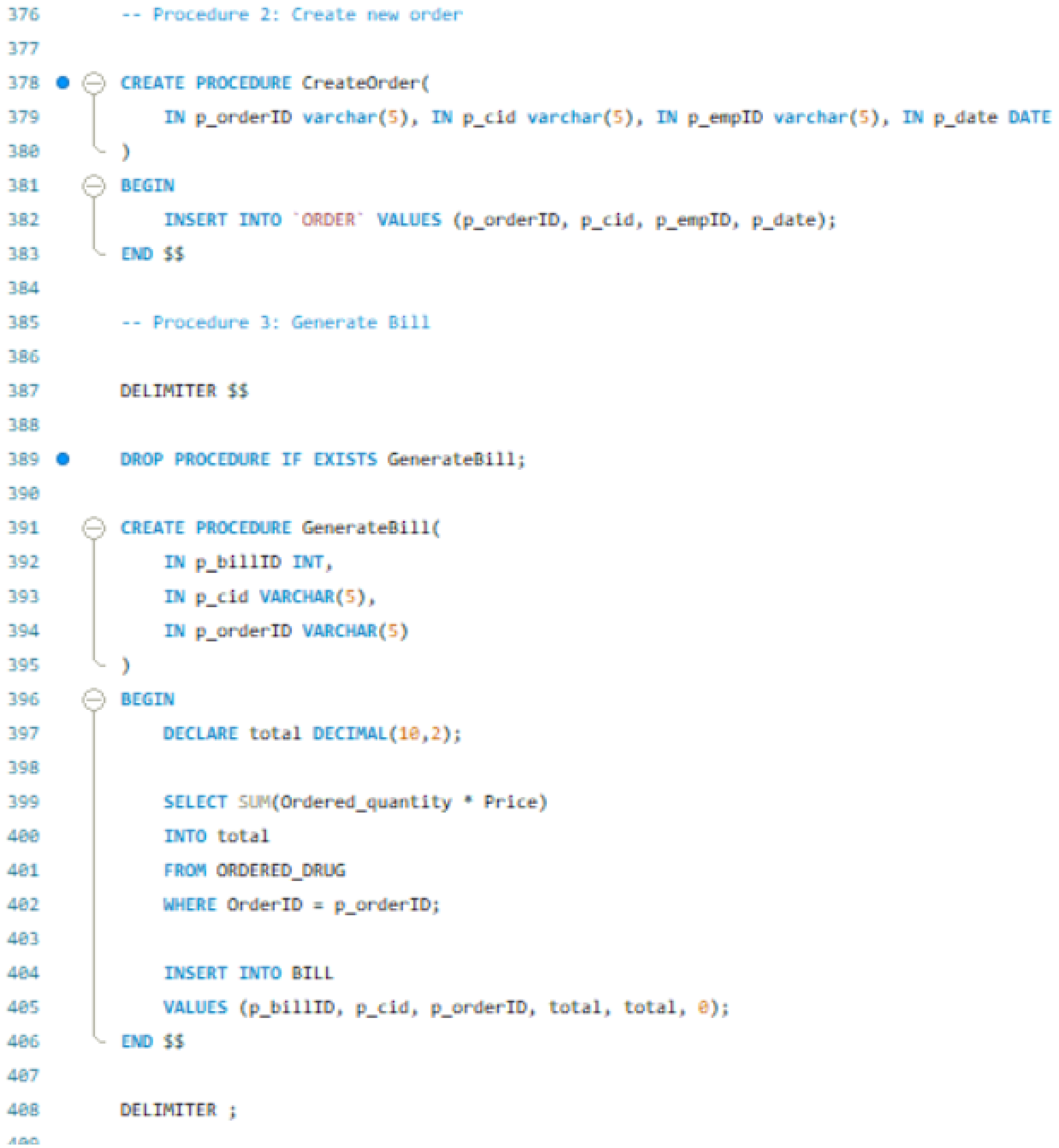
```sql
-- =======================
-- TRIGGERS
-- =======================
DELIMITER $$

-- 1. Reduce stock after sale
CREATE TRIGGER trg_reduce_stock
AFTER INSERT ON ORDERED_DRUG
FOR EACH ROW
BEGIN
    UPDATE MEDICINE
    SET Stock_quantity = Stock_quantity - NEW.Ordered_quantity
    WHERE BatchNo = NEW.BatchNo AND DrugName = NEW.DrugName;
END $$

-- INSERT INTO ORDERED_DRUG VALUES ('Amoxicillin', 'O5', 'B002', 5, 25.00);
-- SELECT * FROM ORDERED_DRUG;

-- 2. Prevent sale if stock insufficient
CREATE TRIGGER trg_check_stock_before_order
BEFORE INSERT ON ORDERED_DRUG
FOR EACH ROW
BEGIN
    DECLARE current_stock INT;
    SELECT Stock_quantity INTO current_stock
    FROM MEDICINE
    WHERE BatchNo = NEW.BatchNo AND DrugName = NEW.DrugName;
    IF current_stock < NEW.Ordered_quantity THEN
        SIGNAL SQLSTATE '45000'
        SET MESSAGE_TEXT = 'Not enough stock to process the order.';
    END IF;
END $$
```



```sql
-- 3. Block expired medicines from sale
CREATE TRIGGER trg_block_expired
BEFORE INSERT ON ORDERED_DRUG
FOR EACH ROW
BEGIN
    DECLARE exp DATE;
    SELECT ExpiryDate INTO exp
    FROM MEDICINE
    WHERE BatchNo = NEW.BatchNo AND DrugName = NEW.DrugName;
    IF exp < CURDATE() THEN
        SIGNAL SQLSTATE '45000'
        SET MESSAGE_TEXT = 'Cannot sell expired medicine.';
    END IF;
END $$

DELIMITER ;

-- ========================
-- FUNCTIONS
-- ========================
DELIMITER $$

-- Function 1: Total stock value
CREATE FUNCTION TotalStockValue()
RETURNS DECIMAL(10,2)
DETERMINISTIC
BEGIN
    DECLARE total DECIMAL(10,2);
    SELECT SUM(Stock_quantity * Price) INTO total FROM MEDICINE;
    RETURN total;
END $$
```

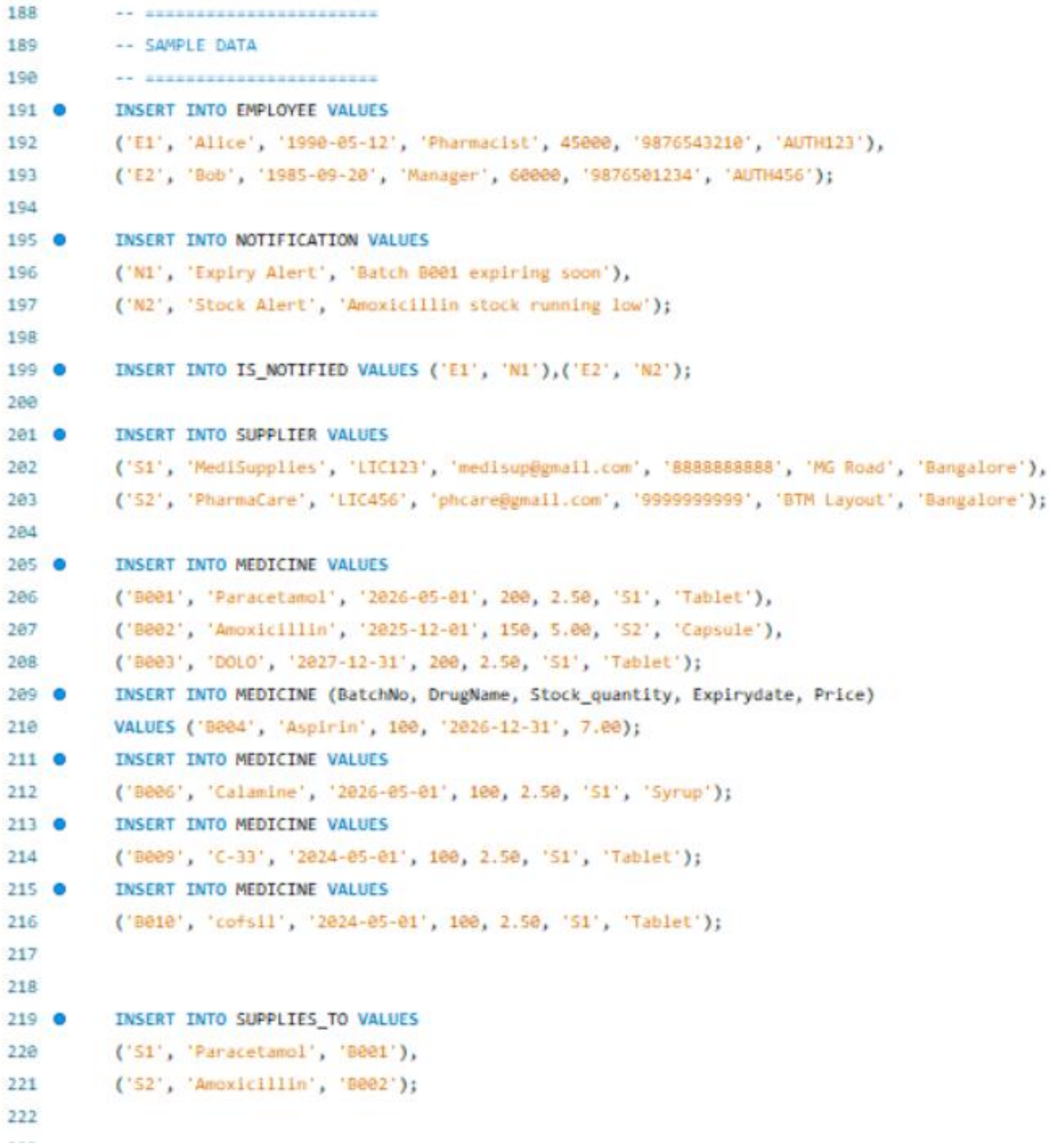


```sql
-- Function 2: Check if medicine expired
CREATE FUNCTION IsExpired(p_batch VARCHAR(20), p_name VARCHAR(50))
RETURNS BOOLEAN
DETERMINISTIC
BEGIN
    DECLARE exp DATE;
    SELECT ExpiryDate INTO exp
    FROM MEDICINE
    WHERE BatchNo = p_batch AND DrugName = p_name;
    RETURN exp < CURDATE();
END $$

DELIMITER ;

-- ======================
-- PROCEDURES
-- ======================
DELIMITER $$

-- Procedure 1: Add new medicine

CREATE PROCEDURE AddMedicine(
    IN p_batch VARCHAR(20), IN p_name VARCHAR(50),
    IN p_exp DATE, IN p_stock INT,
    IN p_price DECIMAL(10,2), IN p_supID VARCHAR(5), IN p_type VARCHAR(30)
)
BEGIN
    INSERT INTO MEDICINE VALUES (p_batch, p_name, p_exp, p_stock, p_price, p_supID, p_type);
END $$

```

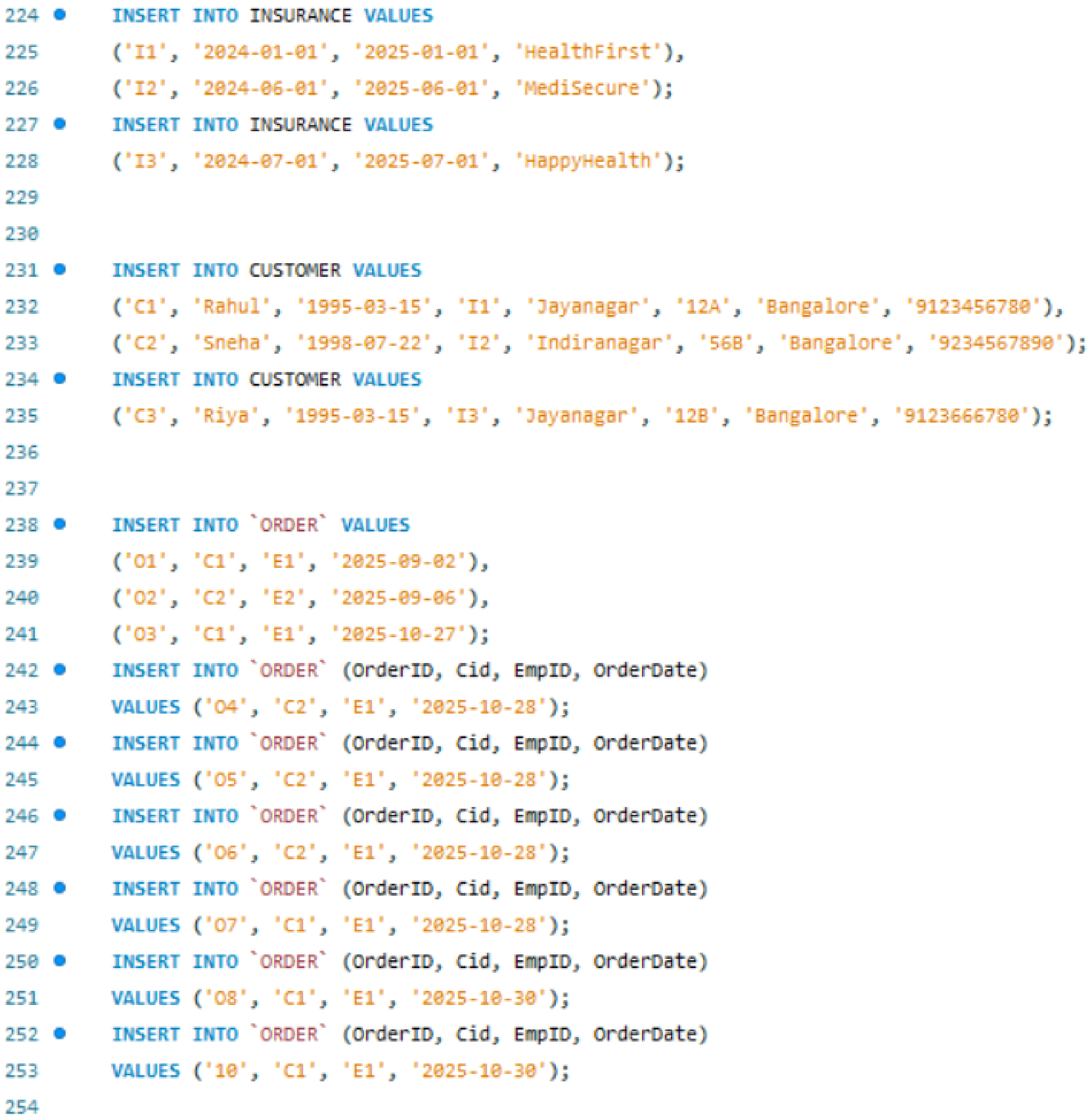


```
-- Procedure 2: Create new order

CREATE PROCEDURE CreateOrder(
    IN p_orderID varchar(5), IN p_cid varchar(5), IN p_empID varchar(5), IN p_date DATE
)
BEGIN
    INSERT INTO `ORDER` VALUES (p_orderID, p_cid, p_empID, p_date);
END $$

-- Procedure 3: Generate Bill

DELIMITER $$

DROP PROCEDURE IF EXISTS GenerateBill;

CREATE PROCEDURE GenerateBill(
    IN p_billID INT,
    IN p_cid VARCHAR(5),
    IN p_orderID VARCHAR(5)
)
BEGIN
    DECLARE total DECIMAL(10,2);

    SELECT SUM(Ordered_quantity * Price)
    INTO total
    FROM ORDERED_DRUG
    WHERE OrderID = p_orderID;

    INSERT INTO BILL
    VALUES (p_billID, p_cid, p_orderID, total, total, 0);
END $$

DELIMITER ;
```

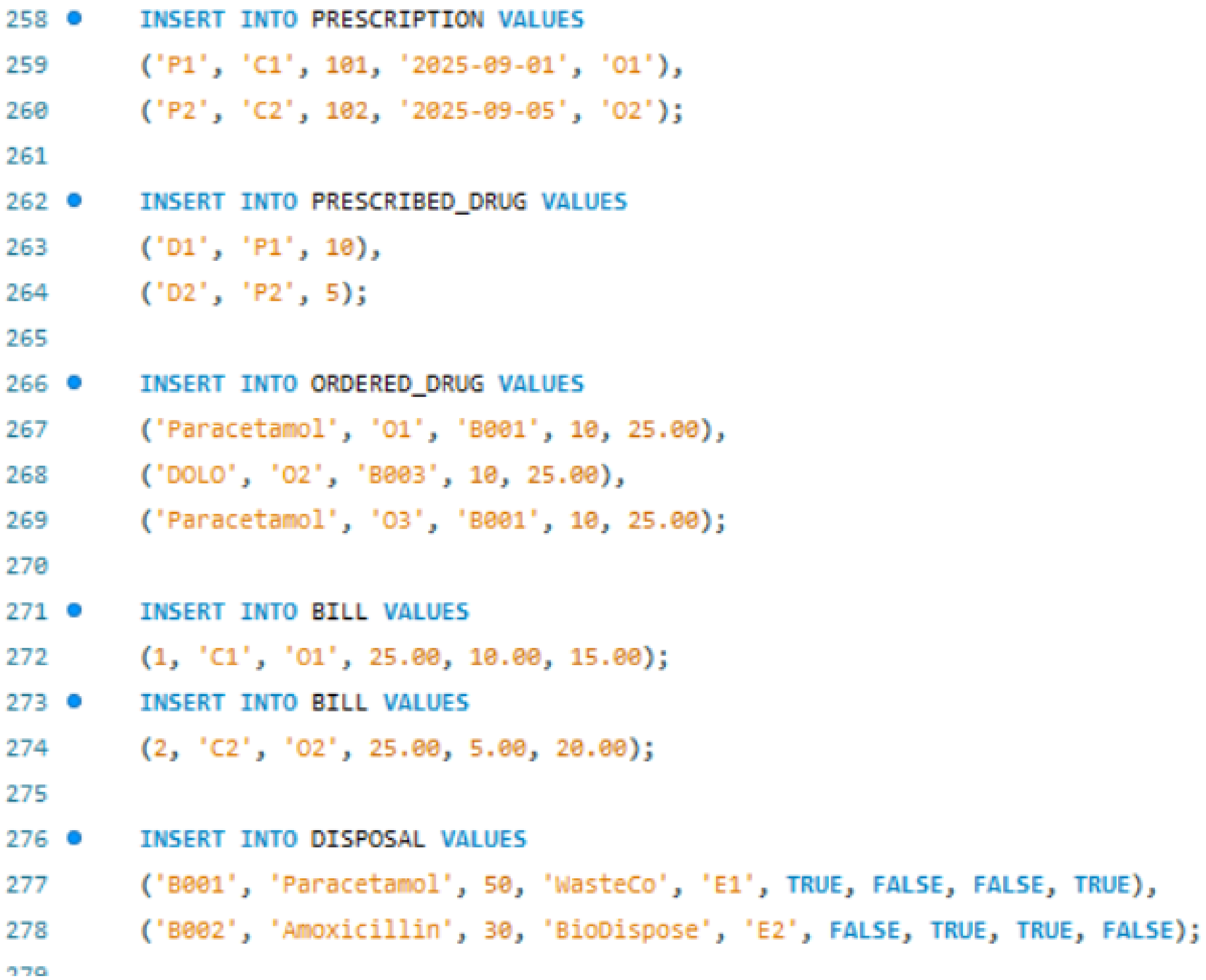
DML COMMANDS:



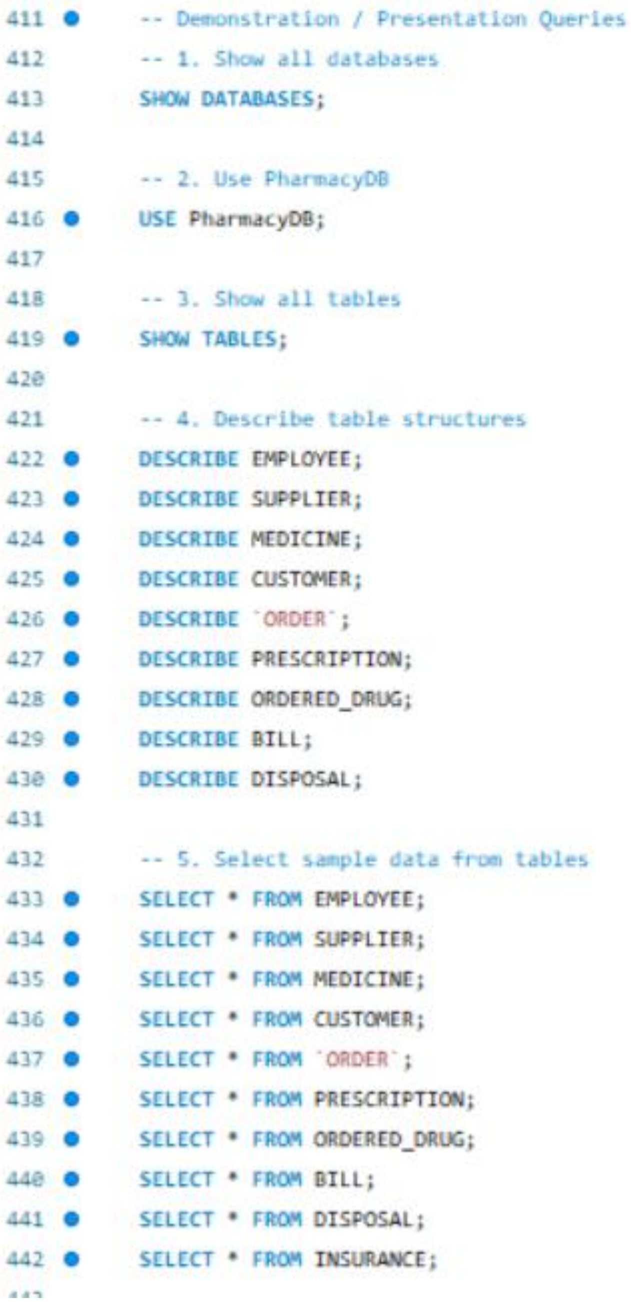
```sql
-- ========================
-- SAMPLE DATA
-- ========================
INSERT INTO EMPLOYEE VALUES
('E1', 'Alice', '1990-05-12', 'Pharmacist', 45000, '9876543210', 'AUTH123'),
('E2', 'Bob', '1985-09-20', 'Manager', 60000, '9876501234', 'AUTH456');

INSERT INTO NOTIFICATION VALUES
('N1', 'Expiry Alert', 'Batch B001 expiring soon'),
('N2', 'Stock Alert', 'Amoxicillin stock running low');

INSERT INTO IS_NOTIFIED VALUES ('E1', 'N1'),('E2', 'N2');

INSERT INTO SUPPLIER VALUES
('S1', 'MediSupplies', 'LIC123', 'medisup@gmail.com', '8888888888', 'MG Road', 'Bangalore'),
('S2', 'PharmaCare', 'LIC456', 'phcare@gmail.com', '9999999999', 'BTM Layout', 'Bangalore');

INSERT INTO MEDICINE VALUES
('B001', 'Paracetamol', '2026-05-01', 200, 2.50, 'S1', 'Tablet'),
('B002', 'Amoxicillin', '2025-12-01', 150, 5.00, 'S2', 'Capsule'),
('B003', 'DOLO', '2027-12-31', 200, 2.50, 'S1', 'Tablet');
INSERT INTO MEDICINE (BatchNo, DrugName, Stock_quantity, Expirydate, Price)
VALUES ('B004', 'Aspirin', 100, '2026-12-31', 7.00);
INSERT INTO MEDICINE VALUES
('B006', 'Calamine', '2026-05-01', 100, 2.50, 'S1', 'Syrup');
INSERT INTO MEDICINE VALUES
('B009', 'C-33', '2024-05-01', 100, 2.50, 'S1', 'Tablet');
INSERT INTO MEDICINE VALUES
('B010', 'cofsil', '2024-05-01', 100, 2.50, 'S1', 'Tablet');


INSERT INTO SUPPLIES_TO VALUES
('S1', 'Paracetamol', 'B001'),
('S2', 'Amoxicillin', 'B002');

```



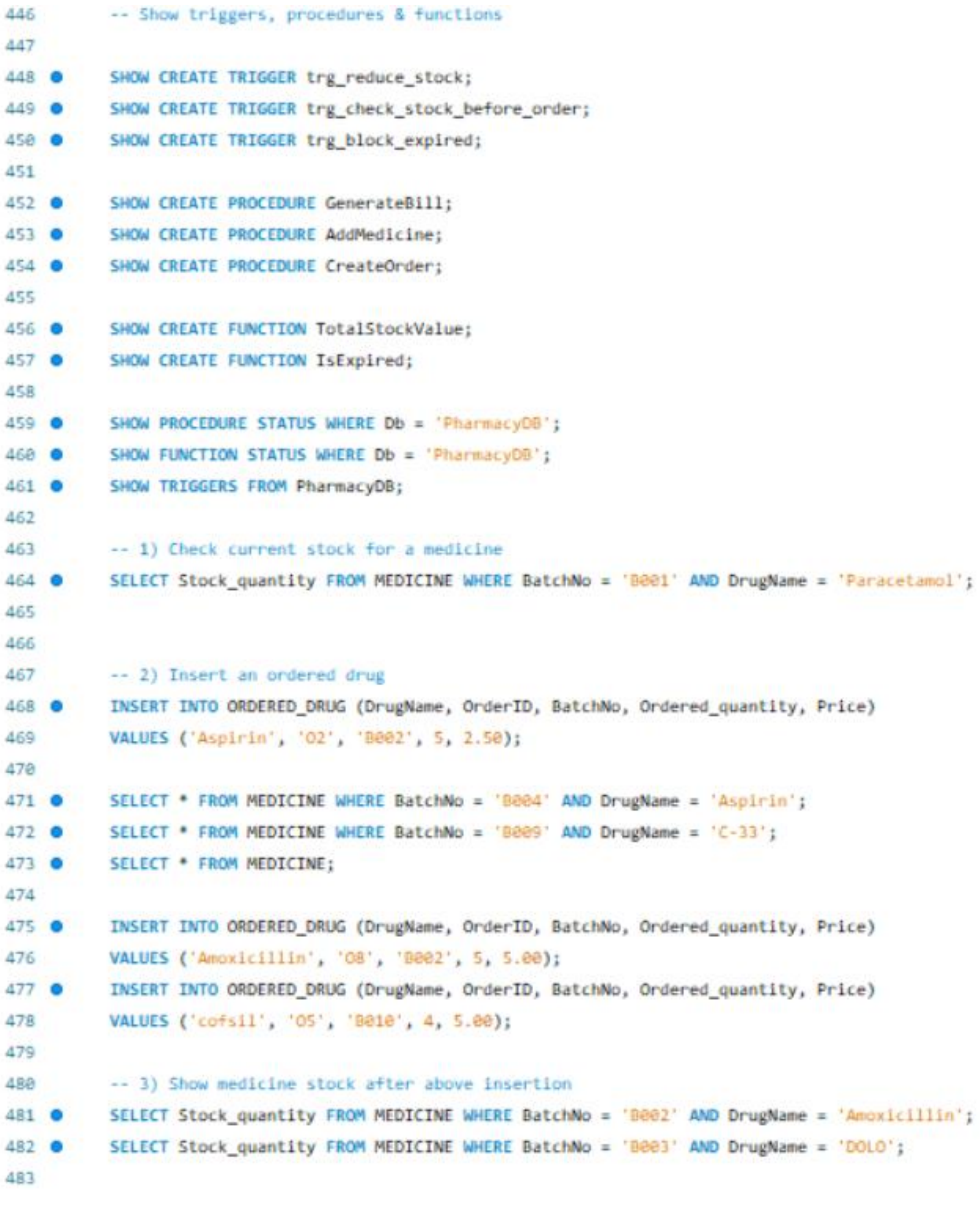
```sql
INSERT INTO INSURANCE VALUES
('I1', '2024-01-01', '2025-01-01', 'HealthFirst'),
('I2', '2024-06-01', '2025-06-01', 'MediSecure');
INSERT INTO INSURANCE VALUES
('I3', '2024-07-01', '2025-07-01', 'HappyHealth');


INSERT INTO CUSTOMER VALUES
('C1', 'Rahul', '1995-03-15', 'I1', 'Jayanagar', '12A', 'Bangalore', '9123456780'),
('C2', 'Sneha', '1998-07-22', 'I2', 'Indiranagar', '56B', 'Bangalore', '9234567890');
INSERT INTO CUSTOMER VALUES
('C3', 'Riya', '1995-03-15', 'I3', 'Jayanagar', '12B', 'Bangalore', '9123666780');


INSERT INTO `ORDER` VALUES
('O1', 'C1', 'E1', '2025-09-02'),
('O2', 'C2', 'E2', '2025-09-06'),
('O3', 'C1', 'E1', '2025-10-27');
INSERT INTO `ORDER` (OrderID, Cid, EmpID, OrderDate)
VALUES ('O4', 'C2', 'E1', '2025-10-28');
INSERT INTO `ORDER` (OrderID, Cid, EmpID, OrderDate)
VALUES ('O5', 'C2', 'E1', '2025-10-28');
INSERT INTO `ORDER` (OrderID, Cid, EmpID, OrderDate)
VALUES ('O6', 'C2', 'E1', '2025-10-28');
INSERT INTO `ORDER` (OrderID, Cid, EmpID, OrderDate)
VALUES ('O7', 'C1', 'E1', '2025-10-28');
INSERT INTO `ORDER` (OrderID, Cid, EmpID, OrderDate)
VALUES ('O8', 'C1', 'E1', '2025-10-30');
INSERT INTO `ORDER` (OrderID, Cid, EmpID, OrderDate)
VALUES ('10', 'C1', 'E1', '2025-10-30');
```



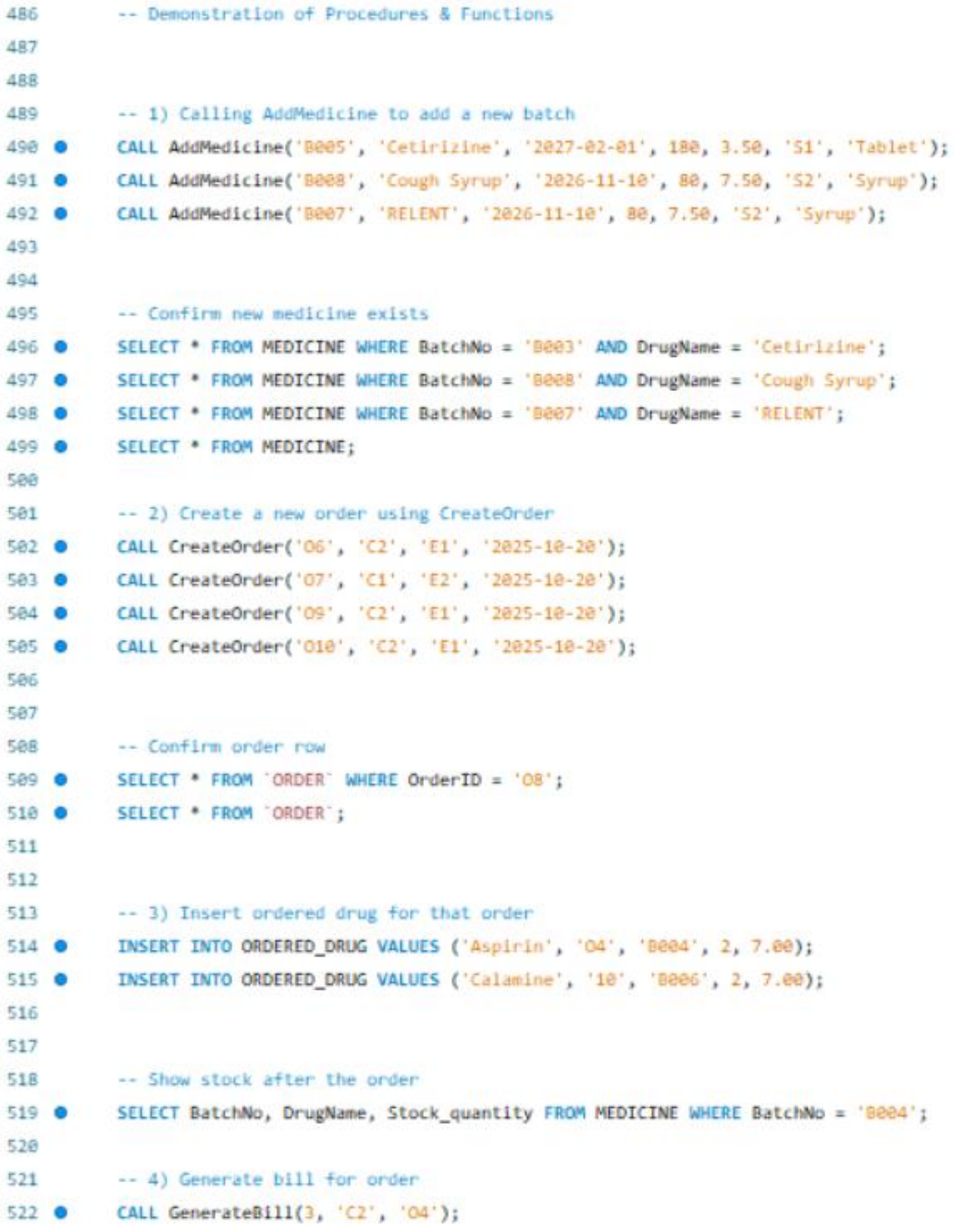
```sql
INSERT INTO PRESCRIPTION VALUES
('P1', 'C1', 101, '2025-09-01', 'O1'),
('P2', 'C2', 102, '2025-09-05', 'O2');

INSERT INTO PRESCRIBED_DRUG VALUES
('D1', 'P1', 10),
('D2', 'P2', 5);

INSERT INTO ORDERED_DRUG VALUES
('Paracetamol', 'O1', 'B001', 10, 25.00),
('DOLO', 'O2', 'B003', 10, 25.00),
('Paracetamol', 'O3', 'B001', 10, 25.00);

INSERT INTO BILL VALUES
(1, 'C1', 'O1', 25.00, 10.00, 15.00);
INSERT INTO BILL VALUES
(2, 'C2', 'O2', 25.00, 5.00, 20.00);

INSERT INTO DISPOSAL VALUES
('B001', 'Paracetamol', 50, 'WasteCo', 'E1', TRUE, FALSE, FALSE, TRUE),
('B002', 'Amoxicillin', 30, 'BioDispose', 'E2', FALSE, TRUE, TRUE, FALSE);
```



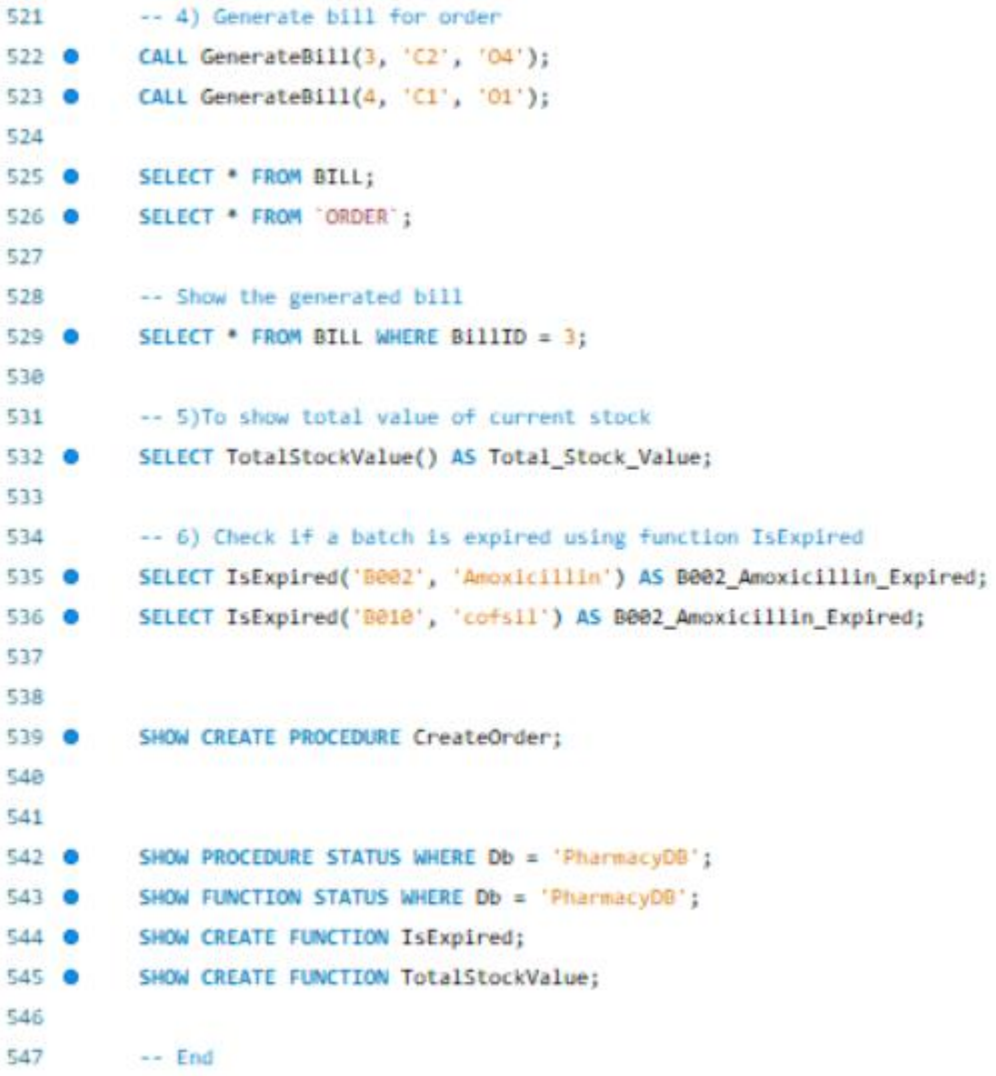
```sql
-- Demonstration / Presentation Queries
-- 1. Show all databases
SHOW DATABASES;

-- 2. Use PharmacyDB
USE PharmacyDB;

-- 3. Show all tables
SHOW TABLES;

-- 4. Describe table structures
DESCRIBE EMPLOYEE;
DESCRIBE SUPPLIER;
DESCRIBE MEDICINE;
DESCRIBE CUSTOMER;
DESCRIBE `ORDER`;
DESCRIBE PRESCRIPTION;
DESCRIBE ORDERED_DRUG;
DESCRIBE BILL;
DESCRIBE DISPOSAL;

-- 5. Select sample data from tables
SELECT * FROM EMPLOYEE;
SELECT * FROM SUPPLIER;
SELECT * FROM MEDICINE;
SELECT * FROM CUSTOMER;
SELECT * FROM `ORDER`;
SELECT * FROM PRESCRIPTION;
SELECT * FROM ORDERED_DRUG;
SELECT * FROM BILL;
SELECT * FROM DISPOSAL;
SELECT * FROM INSURANCE;
```



```sql
-- Show triggers, procedures & functions

SHOW CREATE TRIGGER trg_reduce_stock;
SHOW CREATE TRIGGER trg_check_stock_before_order;
SHOW CREATE TRIGGER trg_block_expired;

SHOW CREATE PROCEDURE GenerateBill;
SHOW CREATE PROCEDURE AddMedicine;
SHOW CREATE PROCEDURE CreateOrder;

SHOW CREATE FUNCTION TotalStockValue;
SHOW CREATE FUNCTION IsExpired;

SHOW PROCEDURE STATUS WHERE Db = 'PharmacyDB';
SHOW FUNCTION STATUS WHERE Db = 'PharmacyDB';
SHOW TRIGGERS FROM PharmacyDB;

-- 1) Check current stock for a medicine
SELECT Stock_quantity FROM MEDICINE WHERE BatchNo = 'B001' AND DrugName = 'Paracetamol';


-- 2) Insert an ordered drug
INSERT INTO ORDERED_DRUG (DrugName, OrderID, BatchNo, Ordered_quantity, Price)
VALUES ('Aspirin', 'O2', 'B002', 5, 2.50);

SELECT * FROM MEDICINE WHERE BatchNo = 'B004' AND DrugName = 'Aspirin';
SELECT * FROM MEDICINE WHERE BatchNo = 'B009' AND DrugName = 'C-33';
SELECT * FROM MEDICINE;

INSERT INTO ORDERED_DRUG (DrugName, OrderID, BatchNo, Ordered_quantity, Price)
VALUES ('Amoxicillin', 'O8', 'B002', 5, 5.00);
INSERT INTO ORDERED_DRUG (DrugName, OrderID, BatchNo, Ordered_quantity, Price)
VALUES ('cofsil', 'O5', 'B010', 4, 5.00);

-- 3) Show medicine stock after above insertion
SELECT Stock_quantity FROM MEDICINE WHERE BatchNo = 'B002' AND DrugName = 'Amoxicillin';
SELECT Stock_quantity FROM MEDICINE WHERE BatchNo = 'B003' AND DrugName = 'DOLO';
```

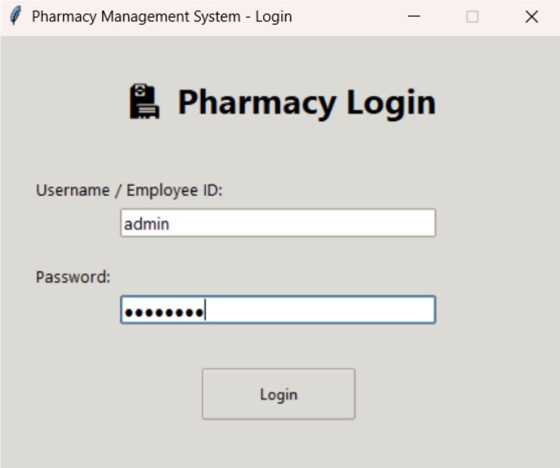


```sql
-- Demonstration of Procedures & Functions


-- 1) Calling AddMedicine to add a new batch
CALL AddMedicine('B005', 'Cetirizine', '2027-02-01', 100, 3.50, 'S1', 'Tablet');
CALL AddMedicine('B008', 'Cough Syrup', '2026-11-10', 80, 7.50, 'S2', 'Syrup');
CALL AddMedicine('B007', 'RELENT', '2026-11-10', 80, 7.50, 'S2', 'Syrup');


-- Confirm new medicine exists
SELECT * FROM MEDICINE WHERE BatchNo = 'B003' AND DrugName = 'Cetirizine';
SELECT * FROM MEDICINE WHERE BatchNo = 'B008' AND DrugName = 'Cough Syrup';
SELECT * FROM MEDICINE WHERE BatchNo = 'B007' AND DrugName = 'RELENT';
SELECT * FROM MEDICINE;

-- 2) Create a new order using CreateOrder
CALL CreateOrder('O6', 'C2', 'E1', '2025-10-20');
CALL CreateOrder('O7', 'C1', 'E2', '2025-10-20');
CALL CreateOrder('O9', 'C2', 'E1', '2025-10-20');
CALL CreateOrder('O10', 'C2', 'E1', '2025-10-20');


-- Confirm order row
SELECT * FROM `ORDER` WHERE OrderID = 'O8';
SELECT * FROM `ORDER`;


-- 3) Insert ordered drug for that order
INSERT INTO ORDERED_DRUG VALUES ('Aspirin', 'O4', 'B004', 2, 7.00);
INSERT INTO ORDERED_DRUG VALUES ('Calamine', '10', 'B006', 2, 7.00);


-- Show stock after the order
SELECT BatchNo, DrugName, Stock_quantity FROM MEDICINE WHERE BatchNo = 'B004';

-- 4) Generate bill for order
CALL GenerateBill(3, 'C2', 'O4');
```



```sql
-- 4) Generate bill for order
CALL GenerateBill(3, 'C2', 'O4');
CALL GenerateBill(4, 'C1', 'O1');

SELECT * FROM BILL;
SELECT * FROM `ORDER`;

-- Show the generated bill
SELECT * FROM BILL WHERE BillID = 3;

-- 5)To show total value of current stock
SELECT TotalStockValue() AS Total_Stock_Value;

-- 6) Check if a batch is expired using function IsExpired
SELECT IsExpired('B002', 'Amoxicillin') AS B002_Amoxicillin_Expired;
SELECT IsExpired('B010', 'cofsil') AS B002_Amoxicillin_Expired;


SHOW CREATE PROCEDURE CreateOrder;


SHOW PROCEDURE STATUS WHERE Db = 'PharmacyDB';
SHOW FUNCTION STATUS WHERE Db = 'PharmacyDB';
SHOW CREATE FUNCTION IsExpired;
SHOW CREATE FUNCTION TotalStockValue;

-- End
```

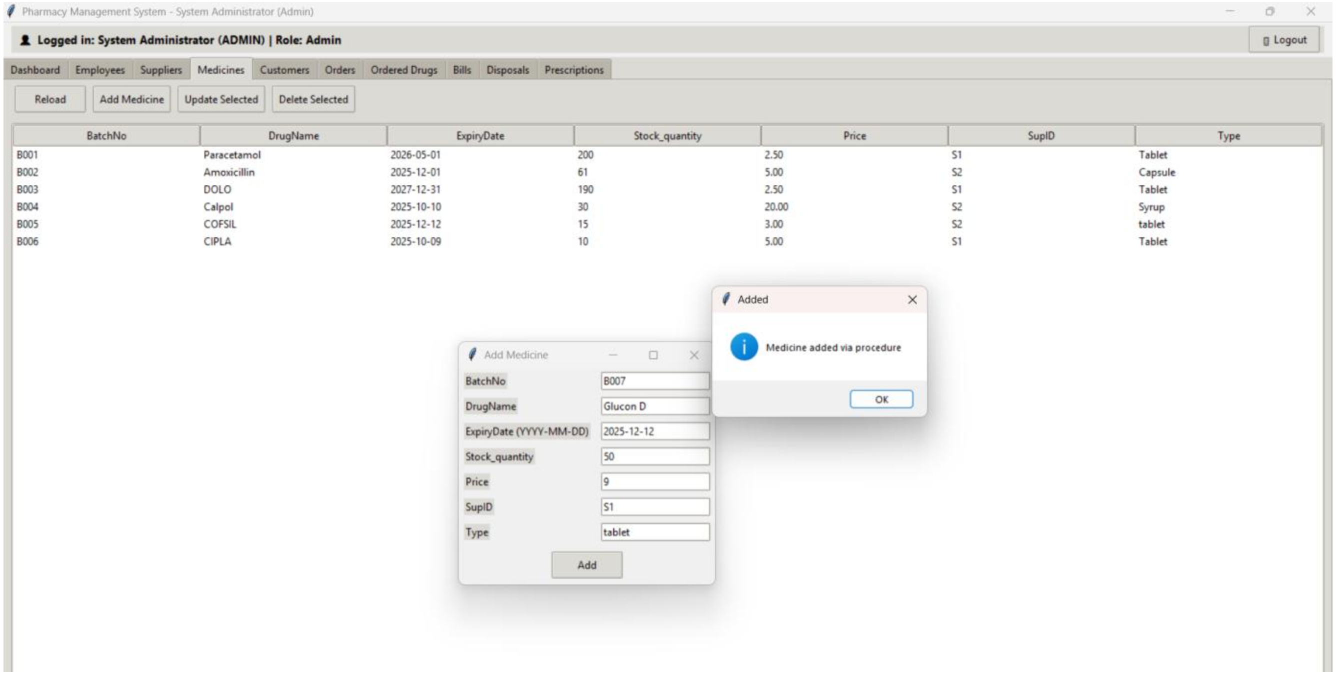
FRONTEND: (few screenshots)



Pharmacy Login

Username / Employee ID: admin

Password: ••••••••

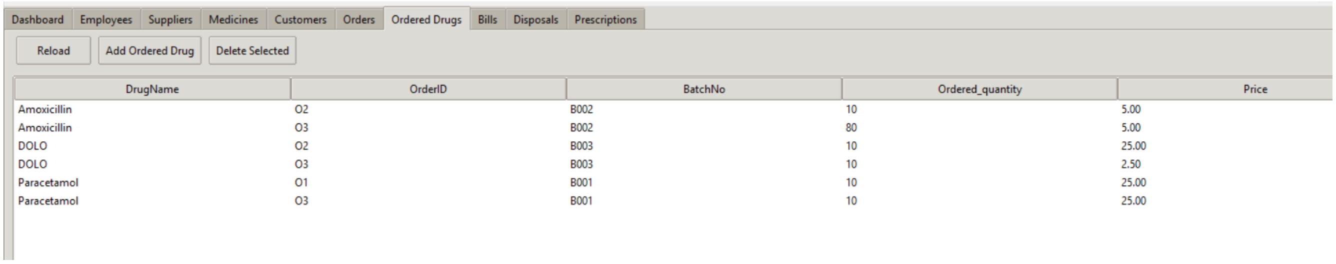


Logged in: System Administrator (ADMIN) | Role: Admin

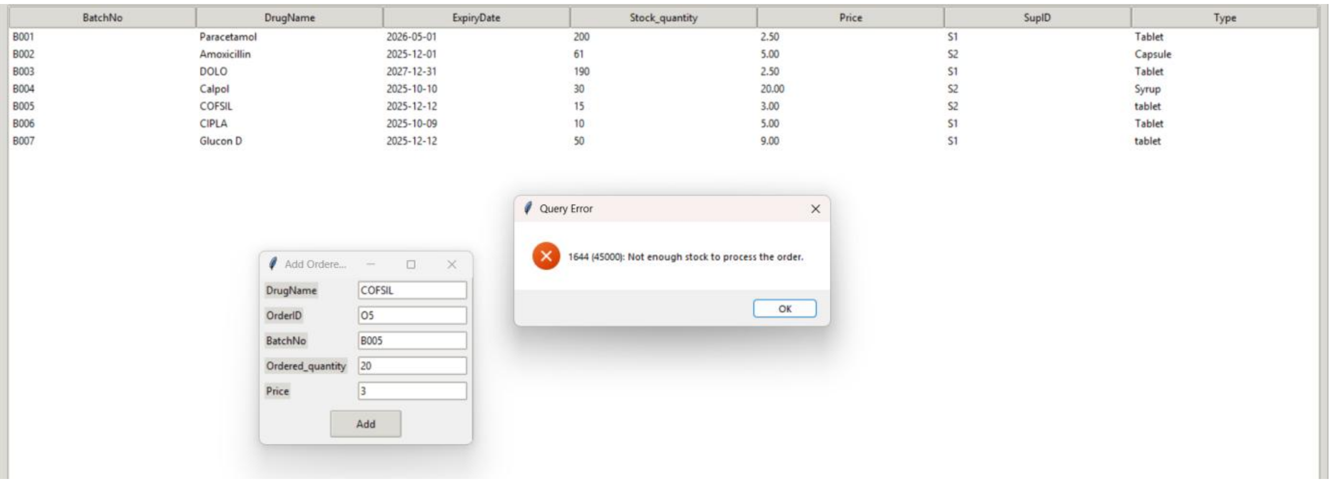
BatchNo	DrugName	ExpiryDate	Stock_quantity	Price	SupID	Type
B001	Paracetamol	2026-05-01	200	2.50	S1	Tablet
B002	Amoxicillin	2025-12-01	61	5.00	S2	Capsule
B003	DOLO	2027-12-31	190	2.50	S1	Tablet
B004	Calpol	2025-10-10	30	20.00	S2	Syrup
B005	COFSIL	2025-12-12	15	3.00	S2	tablet
B006	CIPLA	2025-10-09	10	5.00	S1	Tablet

BatchNo	B007
DrugName	Glucon D
ExpiryDate (YYYY-MM-DD)	2025-12-12
Stock_quantity	50
Price	9
SupID	S1
Type	tablet

Medicine added via procedure



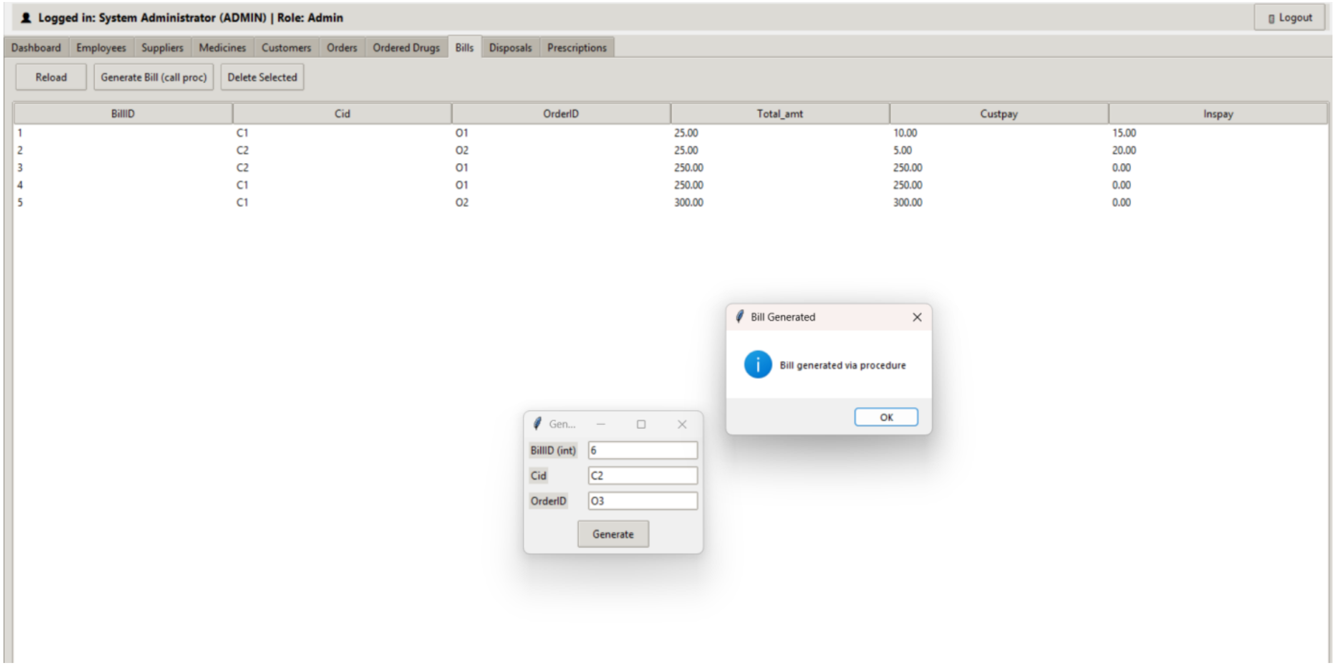
DrugName	OrderID	BatchNo	Ordered_quantity	Price
Amoxicillin	O2	B002	10	5.00
Amoxicillin	O3	B002	80	5.00
DOLO	O2	B003	10	25.00
DOLO	O3	B003	10	2.50
Paracetamol	O1	B001	10	25.00
Paracetamol	O3	B001	10	25.00



BatchNo	DrugName	ExpiryDate	Stock_quantity	Price	SupID	Type
B001	Paracetamol	2026-05-01	200	2.50	S1	Tablet
B002	Amoxicillin	2025-12-01	61	5.00	S2	Capsule
B003	DOLO	2027-12-31	190	2.50	S1	Tablet
B004	Calpol	2025-10-10	30	20.00	S2	Syrup
B005	COFSIL	2025-12-12	15	3.00	S2	tablet
B006	CIPLA	2025-10-09	10	5.00	S1	Tablet
B007	Glucon D	2025-12-12	50	9.00	S1	tablet

1644 (45000): Not enough stock to process the order.

DrugName: COFSIL
OrderID: O5
BatchNo: B005
Ordered_quantity: 20
Price: 3

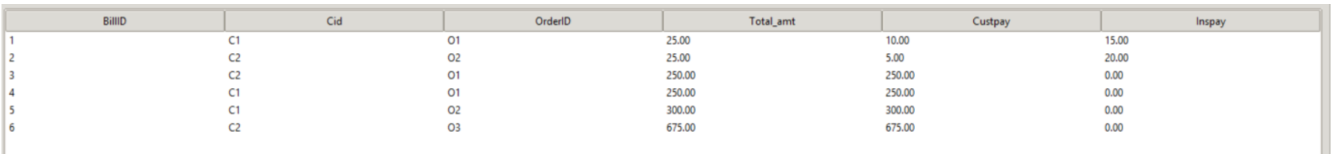


Logged in: System Administrator (ADMIN) | Role: Admin

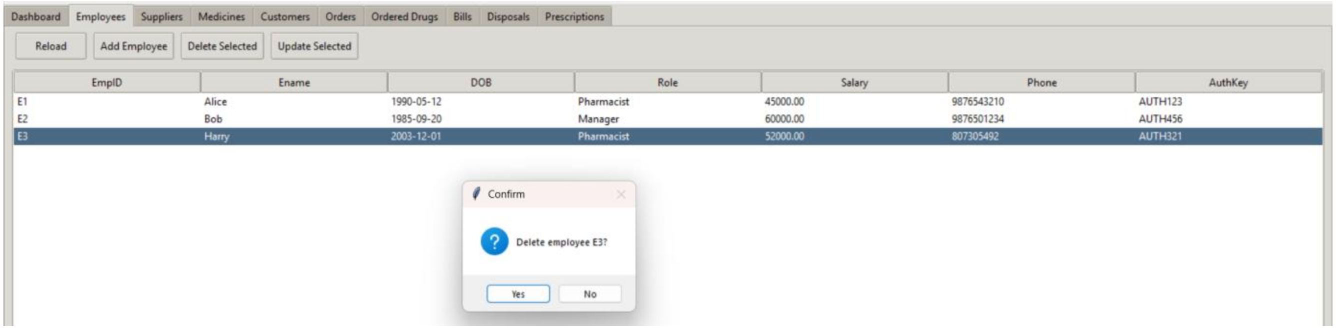
BillID	Cid	OrderID	Total_amt	Custpay	Inspay
1	C1	O1	25.00	10.00	15.00
2	C2	O2	25.00	5.00	20.00
3	C2	O1	250.00	250.00	0.00
4	C1	O1	250.00	250.00	0.00
5	C1	O2	300.00	300.00	0.00

Bill generated via procedure

BillID (int): 6
Cid: C2
OrderID: O3



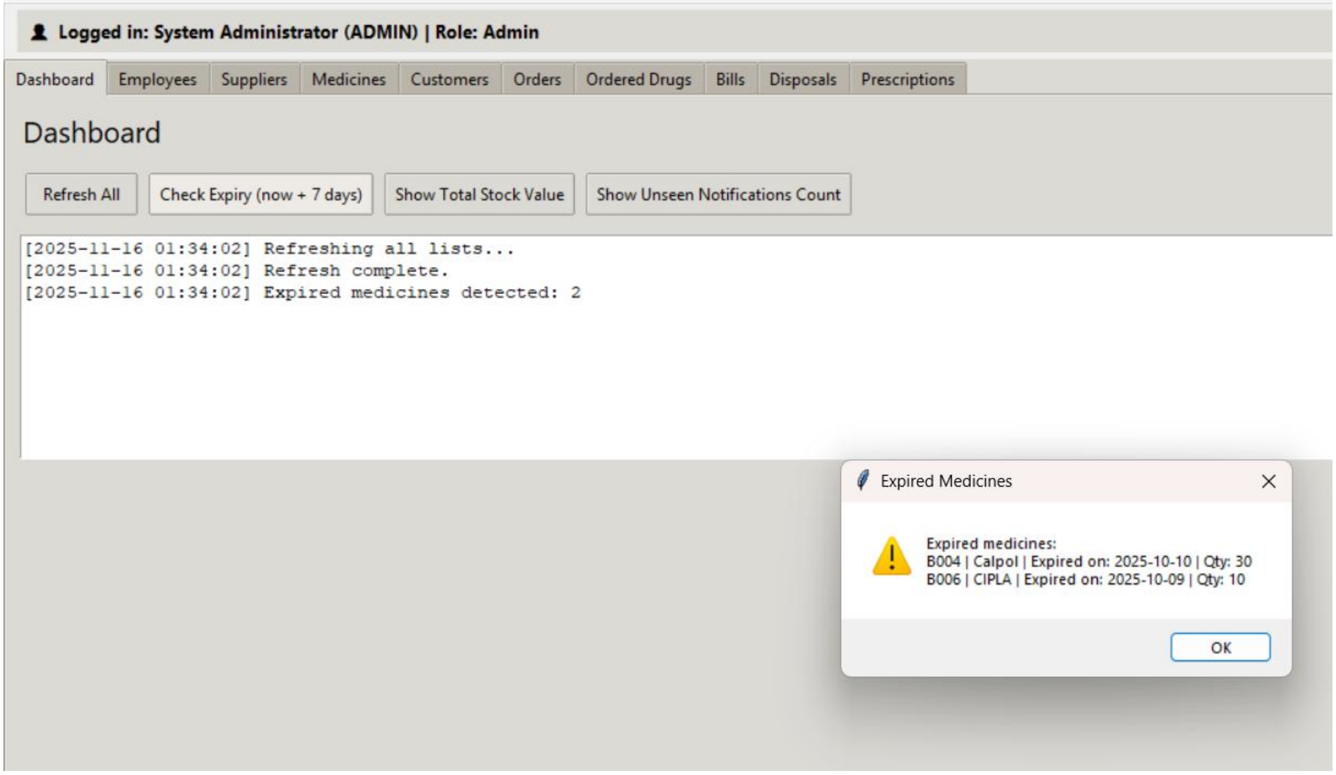
BillID	Cid	OrderID	Total_amt	Custpay	Inspay
1	C1	O1	25.00	10.00	15.00
2	C2	O2	25.00	5.00	20.00
3	C2	O1	250.00	250.00	0.00
4	C1	O1	250.00	250.00	0.00
5	C1	O2	300.00	300.00	0.00
6	C2	O3	675.00	675.00	0.00



EmpID	Ename	DOB	Role	Salary	Phone	AuthKey
E1	Alice	1990-05-12	Pharmacist	45000.00	9876543210	AUTH123
E2	Bob	1985-09-20	Manager	60000.00	9876501234	AUTH456
E3	Harry	2003-12-01	Pharmacist	52000.00	807305492	AUTH321

Confirm

Delete employee E3?

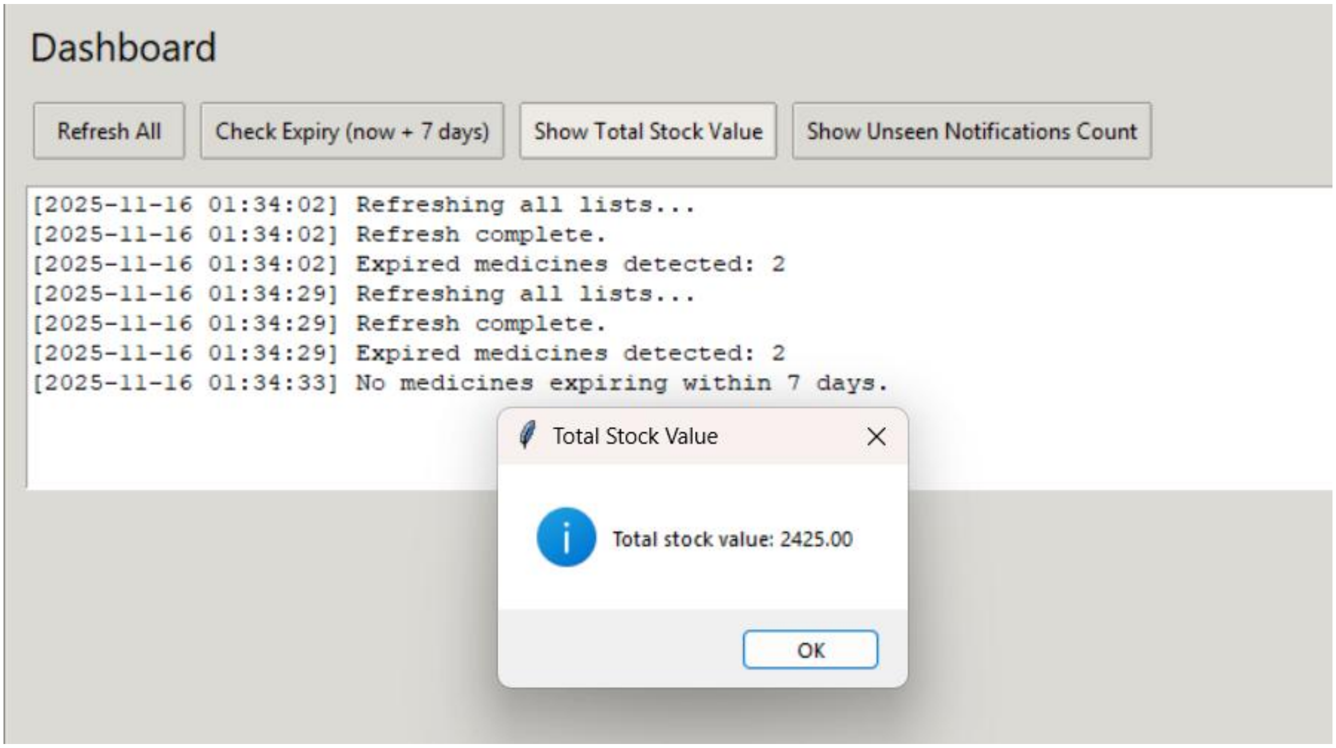

Logged in: System Administrator (ADMIN) | Role: Admin

Dashboard

```
[2025-11-16 01:34:02] Refreshing all lists...
[2025-11-16 01:34:02] Refresh complete.
[2025-11-16 01:34:02] Expired medicines detected: 2
```

Expired Medicines

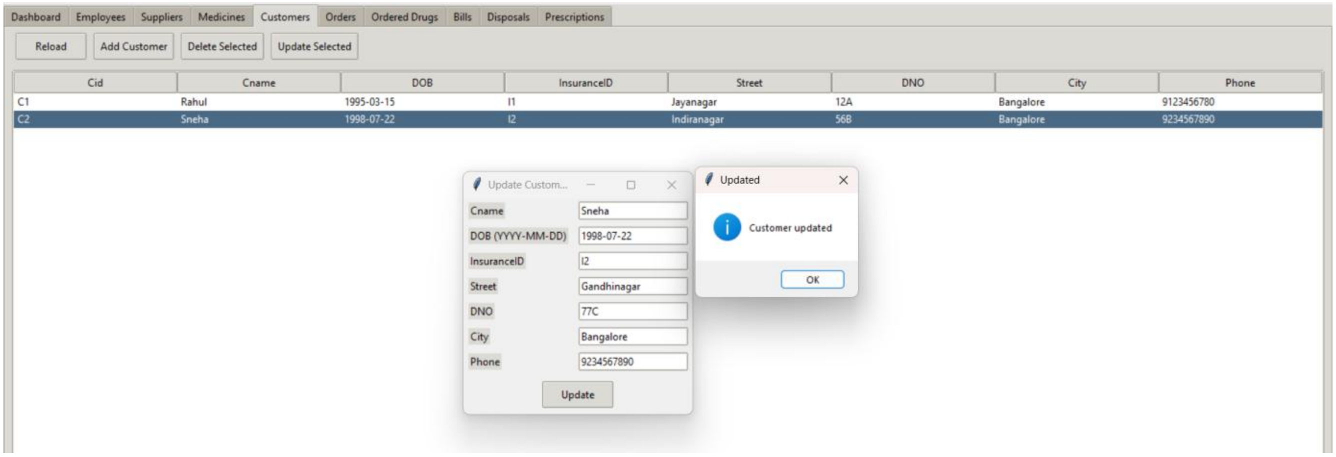
Expired medicines:
B004 | Calpol | Expired on: 2025-10-10 | Qty: 30
B006 | CIPLA | Expired on: 2025-10-09 | Qty: 10



```
[2025-11-16 01:34:02] Refreshing all lists...
[2025-11-16 01:34:02] Refresh complete.
[2025-11-16 01:34:02] Expired medicines detected: 2
[2025-11-16 01:34:29] Refreshing all lists...
[2025-11-16 01:34:29] Refresh complete.
[2025-11-16 01:34:29] Expired medicines detected: 2
[2025-11-16 01:34:33] No medicines expiring within 7 days.
```

Total Stock Value

Total stock value: 2425.00



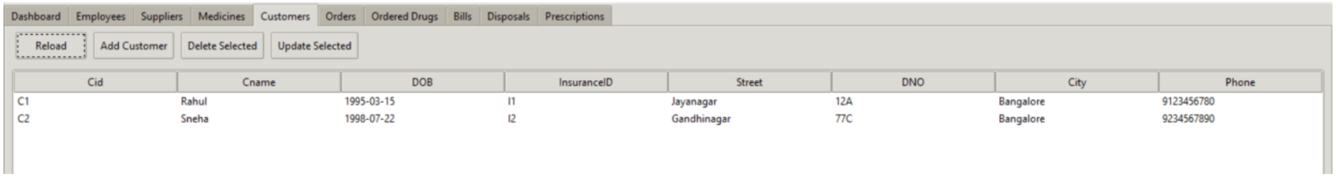
Cid	Cname	DOB	InsuranceID	Street	DNO	City	Phone
C1	Rahul	1995-03-15	I1	Jayanagar	12A	Bangalore	9123456780
C2	Sneha	1998-07-22	I2	Indiranagar	56B	Bangalore	9234567890

Update Custom...

Cname	Sneha
DOB (YYYY-MM-DD)	1998-07-22
InsuranceID	I2
Street	Gandhinagar
DNO	77C
City	Bangalore
Phone	9234567890

Updated

Customer updated



Cid	Cname	DOB	InsuranceID	Street	DNO	City	Phone
C1	Rahul	1995-03-15	I1	Jayanagar	12A	Bangalore	9123456780
C2	Sneha	1998-07-22	I2	Gandhinagar	77C	Bangalore	9234567890

GITHUB REPO LINK:

https://github.com/DBMS-mini-project-pes/Pharmacy-Stock-and-Sales-Management