EE-436L Database Engineering

Project Report

ZAIN MUJAHID	(2016-EE-304)
SAMIULLAH ARSHAD	(2016-EE-276)
FATIMAH LYBA KHAN	(2016-EE-342)
ANNAS BIN MALIK	(2016-EE-337)

June 15, 2020

EE-436L Database Engineering

Project Report

Introduction

Pharmacies are essential component of healthcare in the Pakistan and handle the function of selling medical drugs. Even though the pharmacies do not seem different than any other shop, their functioning is very different due to various laws regarding drugs.

For example, most of the drugs available in a pharmacy cannot be purchased without a prescription. Even with a signed prescription, there is a limit on the quantity that can be purchased. Additionally, pharmacist can do a background check on customer's medical history to ensure that they are not involved in drug abuse.

In addition, there are other laws on the operations of pharmacy like requirement for safe disposal of expired medicine and requirement of license for employees that mix/prepare the drugs.

Thus, preparing a Database Management System for a pharmacy not only requires study of how things are handled from a customer or employee point of view but also the relevant laws. With this project, our aim was to develop a comprehensive system that could deal with challenges faced in day to day operation of a modern pharmacy. We studied the relevant laws and prepared a system that complies with the required Federal and State laws.

Requirements

During research phase, we arrived at following requirements based on the pharmacy flow:

- ### Customer

When a customer arrives in the pharmacy, we identify them based on their SSN. If they are a new customer, they are asked for their name, date of birth, phone number, gender and address.

- ### Insurance

20-25% of Pakistan population has health insurance coverage. If a customer has health insurance, we store the insurance ID (unique for each customer), company name, start date, end date and Co-Insurance. Co-Insurance is a percentage amount that insurance company pays for a medicinal purchase (Managing your healthcare costs, n.d.). Given the customer SSN and insurance ID, the system should be able to automatically calculate the amount paid by insurance company and customer.

- ### Employee

An employee has same details as a customer but they are also given a company ID, that is unique for them. An employee has to have one of the following roles:

1. Pharmacist
2. CPhT (Certified Pharmacy Technician)
3. Intern (can work in the pharmacy part time)
4. Cashier

Apart from cashier, all other roles require a license from Pakistan Medical Commission as they directly deal with mixing and preparation of drugs.

- **Prescription**

Most of the drugs in the pharmacy can only be sold with a prescription. A prescription contains customer's SSN, the prescribing Doctor's ID (required by law) and when the prescription was prescribed.

Each prescription contains a number of prescribed drugs with drug name, quantity and refill limit of each of them. By law, a pharmacy cannot sell more than prescribed quantity or anything that is not listed on prescription.

- **Order**

An order is created from the prescription. This data has to be stored separately because customer may:

1. Buy less medicine than prescription specifies
2. Come back for refills based on same prescription

Each order has a unique Order ID that is automatically assigned by the system. Each order can have multiple drugs, each with their ordered quantity and price. We also record batch number of the drug. This data can be requested by the government and has to be stored.

- **Bill**

Once an order has been completed, a bill is generated by the system. This bill is handed over to the customer and contains order information, insurance information as well as breakdown of amount paid.

The breakdown should be automatically calculated by the system based on insurance, customer and medicine data.

- **Medicine(Inventory)**

Drugs are divided into "over the counter", "restricted" and "prescription only". Federal Law only divides restricted drugs into 5 schedules and require "readily accessible" inventory for schedule 2 drugs.

While not needed by law everywhere, it is beneficial to store an up to date inventory for record keeping as well knowing when we run out of stock.

- **Notifications**

The system should be able to generate notifications based on the following four events:

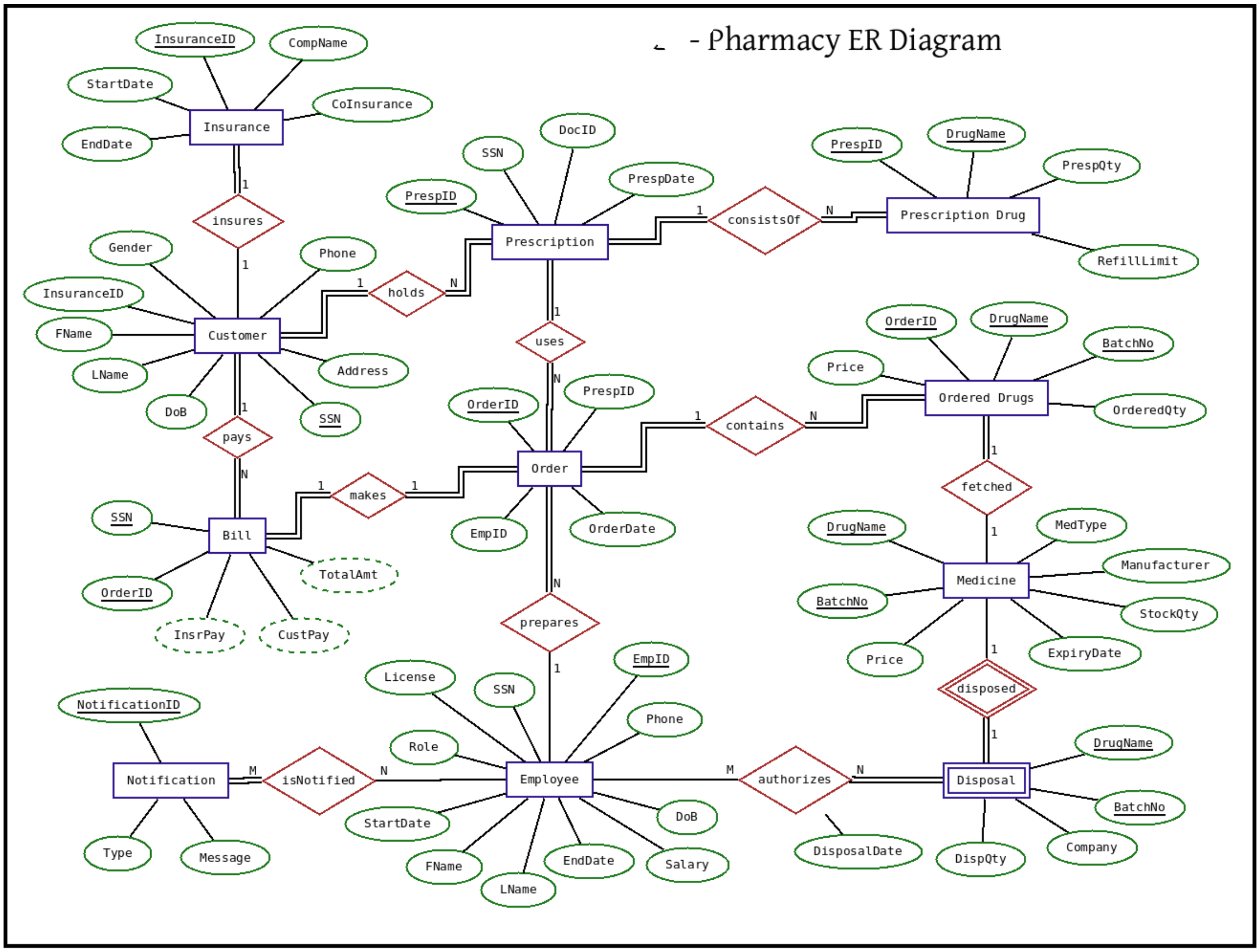
1. Stock for a medicine is low (less than 100 tablets)
2. Some medicine will expire in next 60 days
3. Drugs are marked for disposal
4. Drugs are successfully disposed

The notifications are sent to all the employees who are Pharmacists.

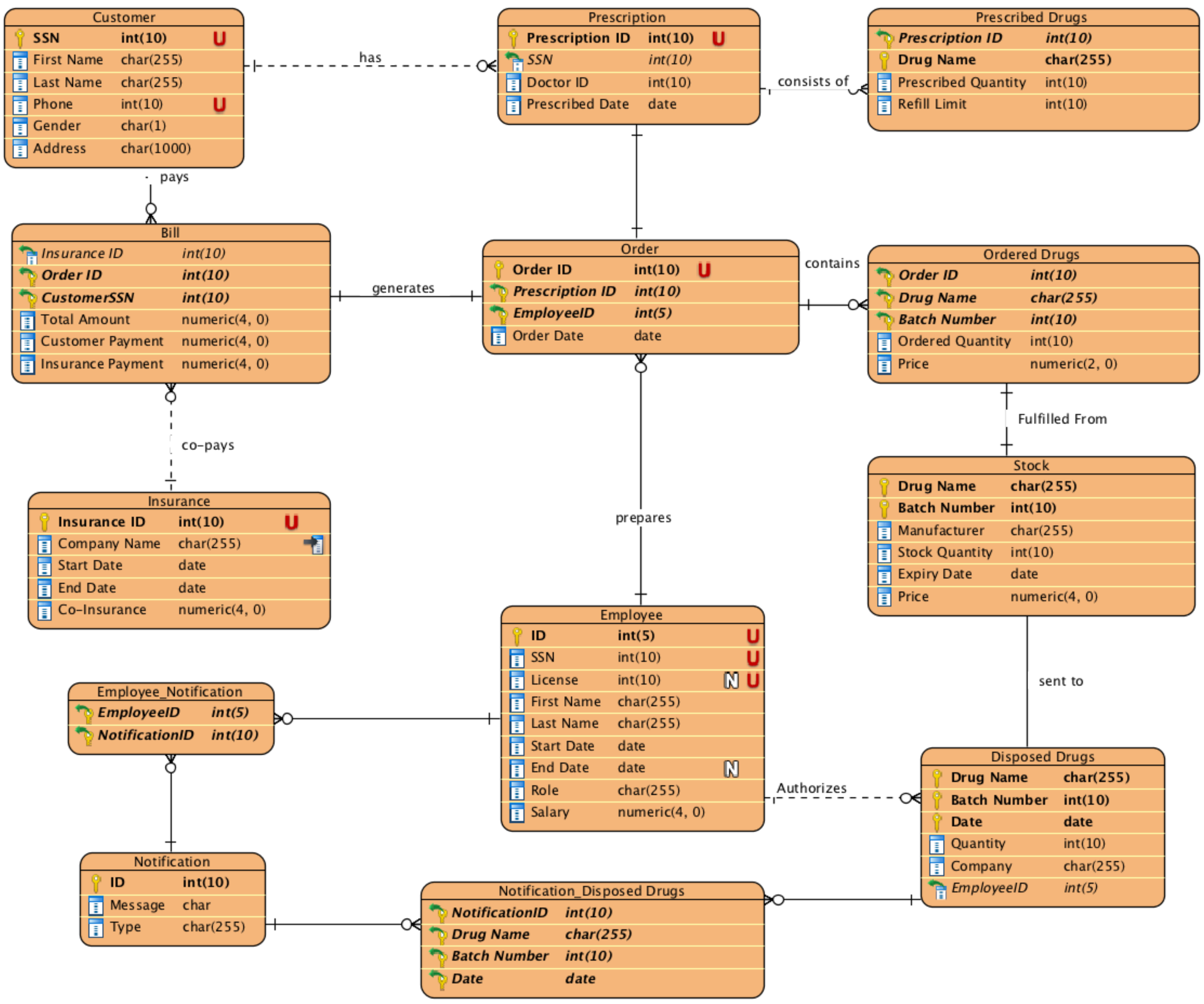
ER Modeling

The final ER diagram and UML diagram are shown below with explanations.

- ### ER Diagram



• Relational Schema



1. A single customer can have multiple prescriptions. Thus, the relation between them is one to many.
2. A prescription consists of multiple drugs, so the relation is one to many. In case of refills, a prescription can generate multiple orders. So, this relation is one to many as well.
3. A single order can contain multiple drugs, thus relationship is one to many. One order, however, can generate only one bill. Thus, the relation between bill and order is one to one.
4. A customer can make multiple purchases and hence, the relation between customer and bill is one to many. This is due to the fact that every bill has only one customer.
5. In medicine table (stock), drug name and batch number can uniquely identify every drug we have in inventory. Batch number is assumed to be unique among manufacturers.
6. Disposed drugs are weak entity and use foreign key Drug Name and Batch Number as their primary key.
7. One employee can receive multiple notifications and one notification can be sent to multiple employees, thus relationship is many to many.
8. Multiple employees can dispose same drug. Similarly, one employee can dispose multiple drugs. Hence, relationship is many to many.
9. One employee can prepare multiple orders. However, a specific order can only be prepared by one employee. Thus, relationship is one to many.

Relations:

The final relations are listed below:

Customer

<u>SSN</u>	First Name	Last Name	Phone	Gender	Address	Date of Birth	Insurance ID

Primary Key: SSN

Foreign Key: Customer(Insurance ID) → Insurance(Insurance ID)

Insurance

<u>Insurance ID</u>	Company Name	Start Date	End Date	Co-Insurance

Primary Key: Insurance ID

Employee

<u>ID</u>	SSN	License	First Name	Last Name	Start Date	End Date	Role
Salary	Phone Number	Date of Birth					

Primary Key: ID

Prescription

<u>Prescription ID</u>	SSN	Doctor ID	Prescription Date

Primary Key: Prescription ID

Foreign Key: Prescription(SSN) → Customer(SSN)

Prescribed Drugs

<u>Prescription ID</u>	<u>Drug Name</u>	Prescribed Quantity	Refill Limit

Primary Key: Prescription ID, Drug Name

Foreign Key: Prescribed Drugs(Prescription ID) → Prescription(Prescription ID)

Order

<u>Order ID</u>	Prescription ID	EmployeeID	Order Date

Primary Key: Order ID

Foreign Key: Order(Prescription ID) → Prescription(Prescription ID), Order(Employee ID) → Employee(ID)

Ordered Drugs

Order ID	Drug Name	Batch Number	Quantity	Price

Primary Key: Order ID, Drug Name, Batch Number
Foreign Key: Ordered Drugs(Order ID) → Order(Order ID), Ordered Drugs(Drug Name, Batch Number) → Medicine(Drug Name, Batch Number)

Bill

Order ID	CustomerSSN	Total Amount	Customer Payment	Insurance Payment

Primary Key: Order ID, Customer SSN
Foreign Key: Bill(Order ID) → Order(Order ID), Bill(Customer SSN) → Customer(SSN)

Medicine

Drug Name	Batch Number	Medicine Type	Manufacturer	Quantity	Expiry Date	Price

Primary Key: Drug Name,
Batch Number

Disposed Drugs

Drug Name	Batch Number	Quantity	Company

Primary Key: Drug Name, Batch Number
Foreign Key: Disposed Drugs(Drug Name, Batch Number) → Medicine(Drug Name, Batch Number)

Notification

ID	Message	Type

Primary Key: ID

Employee_Disposed Drugs

Employee ID	Drug Name	Batch Number	Disposal Date

Primary Key: Employee ID, Drug Name, Batch Number, Disposal Date
Foreign Key: Employee_Disposed Drugs(Employee ID) → Employee (Employee ID), Employee_Disposed Drugs(Drug Name, Batch Number) → Disposed Drugs(Drug Name, Batch Number)

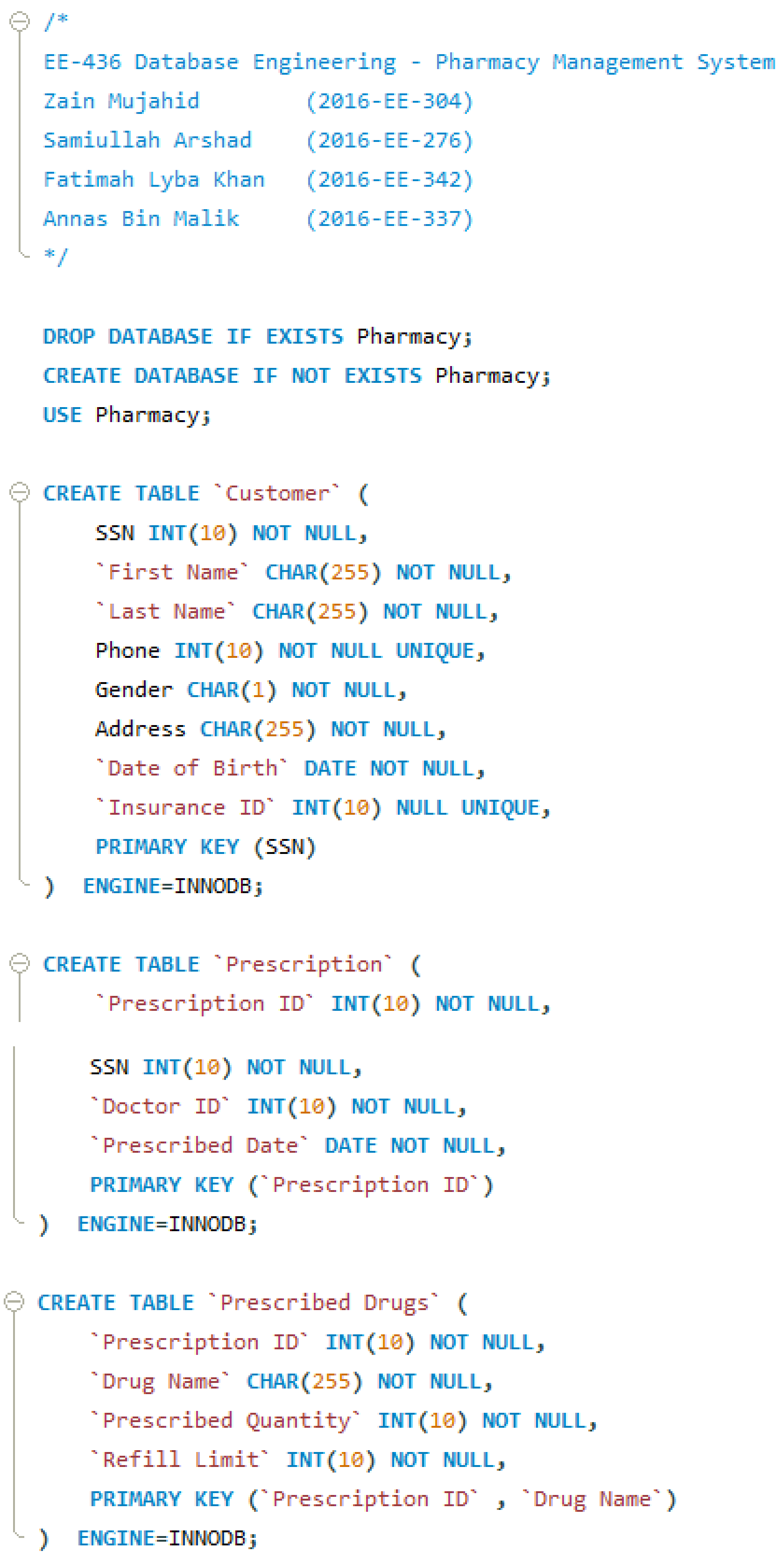
Employee Notification

Employee ID	Notification ID

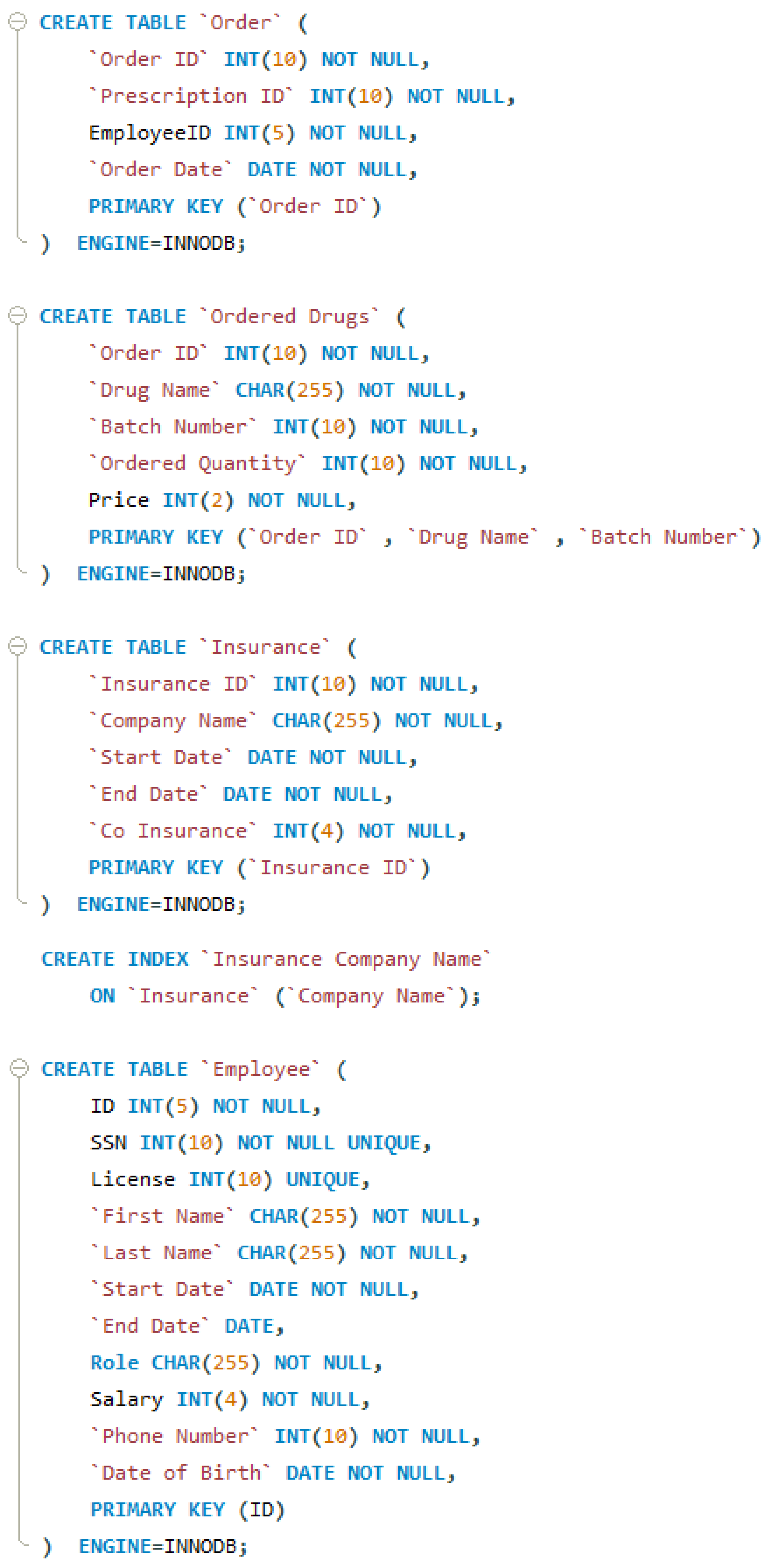
Primary: Employee ID, Notification ID
Foregin Key: Employee Notification(Employee ID) → Employee(ID), Employee Notification(Notification ID) → Notification(Notification ID)

Database Creation

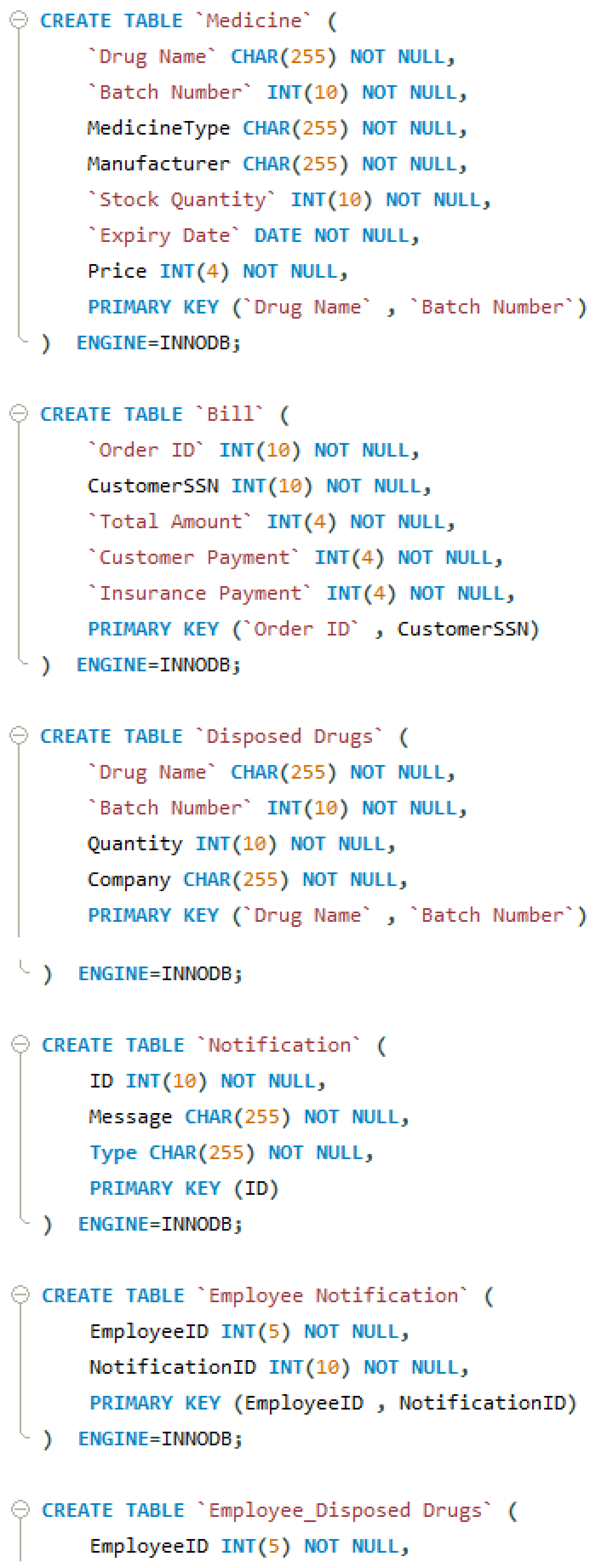
SQL commands for creating the tables in our database:



```
/*
EE-436 Database Engineering - Pharmacy Management System
Zain Mujahid        (2016-EE-304)
Samiullah Arshad    (2016-EE-276)
Fatimah Lyba Khan   (2016-EE-342)
Annas Bin Malik     (2016-EE-337)
*/

DROP DATABASE IF EXISTS Pharmacy;
CREATE DATABASE IF NOT EXISTS Pharmacy;
USE Pharmacy;

CREATE TABLE `Customer` (
    SSN INT(10) NOT NULL,
    `First Name` CHAR(255) NOT NULL,
    `Last Name` CHAR(255) NOT NULL,
    Phone INT(10) NOT NULL UNIQUE,
    Gender CHAR(1) NOT NULL,
    Address CHAR(255) NOT NULL,
    `Date of Birth` DATE NOT NULL,
    `Insurance ID` INT(10) NULL UNIQUE,
    PRIMARY KEY (SSN)
)  ENGINE=INNODB;

CREATE TABLE `Prescription` (
    `Prescription ID` INT(10) NOT NULL,

    SSN INT(10) NOT NULL,
    `Doctor ID` INT(10) NOT NULL,
    `Prescribed Date` DATE NOT NULL,
    PRIMARY KEY (`Prescription ID`)
)  ENGINE=INNODB;

CREATE TABLE `Prescribed Drugs` (
    `Prescription ID` INT(10) NOT NULL,
    `Drug Name` CHAR(255) NOT NULL,
    `Prescribed Quantity` INT(10) NOT NULL,
    `Refill Limit` INT(10) NOT NULL,
    PRIMARY KEY (`Prescription ID` , `Drug Name`)
)  ENGINE=INNODB;
```



```sql
CREATE TABLE `Order` (
    `Order ID` INT(10) NOT NULL,
    `Prescription ID` INT(10) NOT NULL,
    EmployeeID INT(5) NOT NULL,
    `Order Date` DATE NOT NULL,
    PRIMARY KEY (`Order ID`)
) ENGINE=INNODB;

CREATE TABLE `Ordered Drugs` (
    `Order ID` INT(10) NOT NULL,
    `Drug Name` CHAR(255) NOT NULL,
    `Batch Number` INT(10) NOT NULL,
    `Ordered Quantity` INT(10) NOT NULL,
    Price INT(2) NOT NULL,
    PRIMARY KEY (`Order ID` , `Drug Name` , `Batch Number`)
) ENGINE=INNODB;

CREATE TABLE `Insurance` (
    `Insurance ID` INT(10) NOT NULL,
    `Company Name` CHAR(255) NOT NULL,
    `Start Date` DATE NOT NULL,
    `End Date` DATE NOT NULL,
    `Co Insurance` INT(4) NOT NULL,
    PRIMARY KEY (`Insurance ID`)
) ENGINE=INNODB;

CREATE INDEX `Insurance Company Name`
    ON `Insurance` (`Company Name`);

CREATE TABLE `Employee` (
    ID INT(5) NOT NULL,
    SSN INT(10) NOT NULL UNIQUE,
    License INT(10) UNIQUE,
    `First Name` CHAR(255) NOT NULL,
    `Last Name` CHAR(255) NOT NULL,
    `Start Date` DATE NOT NULL,
    `End Date` DATE,
    Role CHAR(255) NOT NULL,
    Salary INT(4) NOT NULL,
    `Phone Number` INT(10) NOT NULL,
    `Date of Birth` DATE NOT NULL,
    PRIMARY KEY (ID)
) ENGINE=INNODB;
```



```sql
CREATE TABLE `Medicine` (
    `Drug Name` CHAR(255) NOT NULL,
    `Batch Number` INT(10) NOT NULL,
    MedicineType CHAR(255) NOT NULL,
    Manufacturer CHAR(255) NOT NULL,
    `Stock Quantity` INT(10) NOT NULL,
    `Expiry Date` DATE NOT NULL,
    Price INT(4) NOT NULL,
    PRIMARY KEY (`Drug Name` , `Batch Number`)
) ENGINE=INNODB;

CREATE TABLE `Bill` (
    `Order ID` INT(10) NOT NULL,
    CustomerSSN INT(10) NOT NULL,
    `Total Amount` INT(4) NOT NULL,
    `Customer Payment` INT(4) NOT NULL,
    `Insurance Payment` INT(4) NOT NULL,
    PRIMARY KEY (`Order ID` , CustomerSSN)
) ENGINE=INNODB;

CREATE TABLE `Disposed Drugs` (
    `Drug Name` CHAR(255) NOT NULL,
    `Batch Number` INT(10) NOT NULL,
    Quantity INT(10) NOT NULL,
    Company CHAR(255) NOT NULL,
    PRIMARY KEY (`Drug Name` , `Batch Number`)

) ENGINE=INNODB;

CREATE TABLE `Notification` (
    ID INT(10) NOT NULL,
    Message CHAR(255) NOT NULL,
    Type CHAR(255) NOT NULL,
    PRIMARY KEY (ID)
) ENGINE=INNODB;

CREATE TABLE `Employee Notification` (
    EmployeeID INT(5) NOT NULL,
    NotificationID INT(10) NOT NULL,
    PRIMARY KEY (EmployeeID , NotificationID)
) ENGINE=INNODB;

CREATE TABLE `Employee_Disposed Drugs` (
    EmployeeID INT(5) NOT NULL,
    `Drug Name` CHAR(255) NOT NULL,
    `Batch Number` INT(10) NOT NULL,
    `Disposal Date` DATE NOT NULL,
    PRIMARY KEY (EmployeeID , `Drug Name` , `Batch Number` , `Disposal Date`)
) ENGINE=INNODB;


-- Adding Foreign Keys --

ALTER TABLE `Customer` ADD CONSTRAINT insures FOREIGN KEY (`Insurance ID`)
    REFERENCES `Insurance` (`Insurance ID`) ON DELETE Set null;

ALTER TABLE `Prescription` ADD CONSTRAINT holds FOREIGN KEY (SSN)
    REFERENCES `Customer` (SSN);

ALTER TABLE `Prescribed Drugs` ADD CONSTRAINT `consists of` FOREIGN KEY (`Prescription ID`)
    REFERENCES `Prescription` (`Prescription ID`) ON DELETE Cascade;

ALTER TABLE `Order` ADD CONSTRAINT prepares FOREIGN KEY (EmployeeID)
    REFERENCES `Employee` (ID);
ALTER TABLE `Order` ADD CONSTRAINT uses FOREIGN KEY (`Prescription ID`)
    REFERENCES `Prescription` (`Prescription ID`);

ALTER TABLE `Ordered Drugs` ADD CONSTRAINT contains FOREIGN KEY (`Order ID`)
    REFERENCES `Order` (`Order ID`) ON DELETE Cascade;
ALTER TABLE `Ordered Drugs` ADD CONSTRAINT `Fulfilled_From` FOREIGN KEY (`Drug Name`, `Batch Number`)

    REFERENCES `Medicine` (`Drug Name`, `Batch Number`);

ALTER TABLE `Bill` ADD CONSTRAINT makes FOREIGN KEY (`Order ID`)
    REFERENCES `Order` (`Order ID`);
ALTER TABLE `Bill` ADD CONSTRAINT pays FOREIGN KEY (CustomerSSN)
    REFERENCES `Customer` (SSN);

ALTER TABLE `Disposed Drugs` ADD CONSTRAINT disposed FOREIGN KEY (`Drug Name`, `Batch Number`)
    REFERENCES `Medicine` (`Drug Name`, `Batch Number`);

ALTER TABLE `Employee Notification` ADD CONSTRAINT FKEmployee_N849182 FOREIGN KEY (EmployeeID)
    REFERENCES `Employee` (ID) ON DELETE Cascade;
ALTER TABLE `Employee Notification` ADD CONSTRAINT FKEmployee_N664471 FOREIGN KEY (NotificationID)
    REFERENCES `Notification` (ID) ON DELETE Cascade;

ALTER TABLE `Employee_Disposed Drugs` ADD CONSTRAINT FKEmployee_D470142 FOREIGN KEY (EmployeeID)
    REFERENCES `Employee` (ID);
ALTER TABLE `Employee_Disposed Drugs` ADD CONSTRAINT FKEmployee_D990025 FOREIGN KEY (`Drug Name`, `Batch Number`)
    REFERENCES `Disposed Drugs` (`Drug Name`, `Batch Number`);
```

Procedures

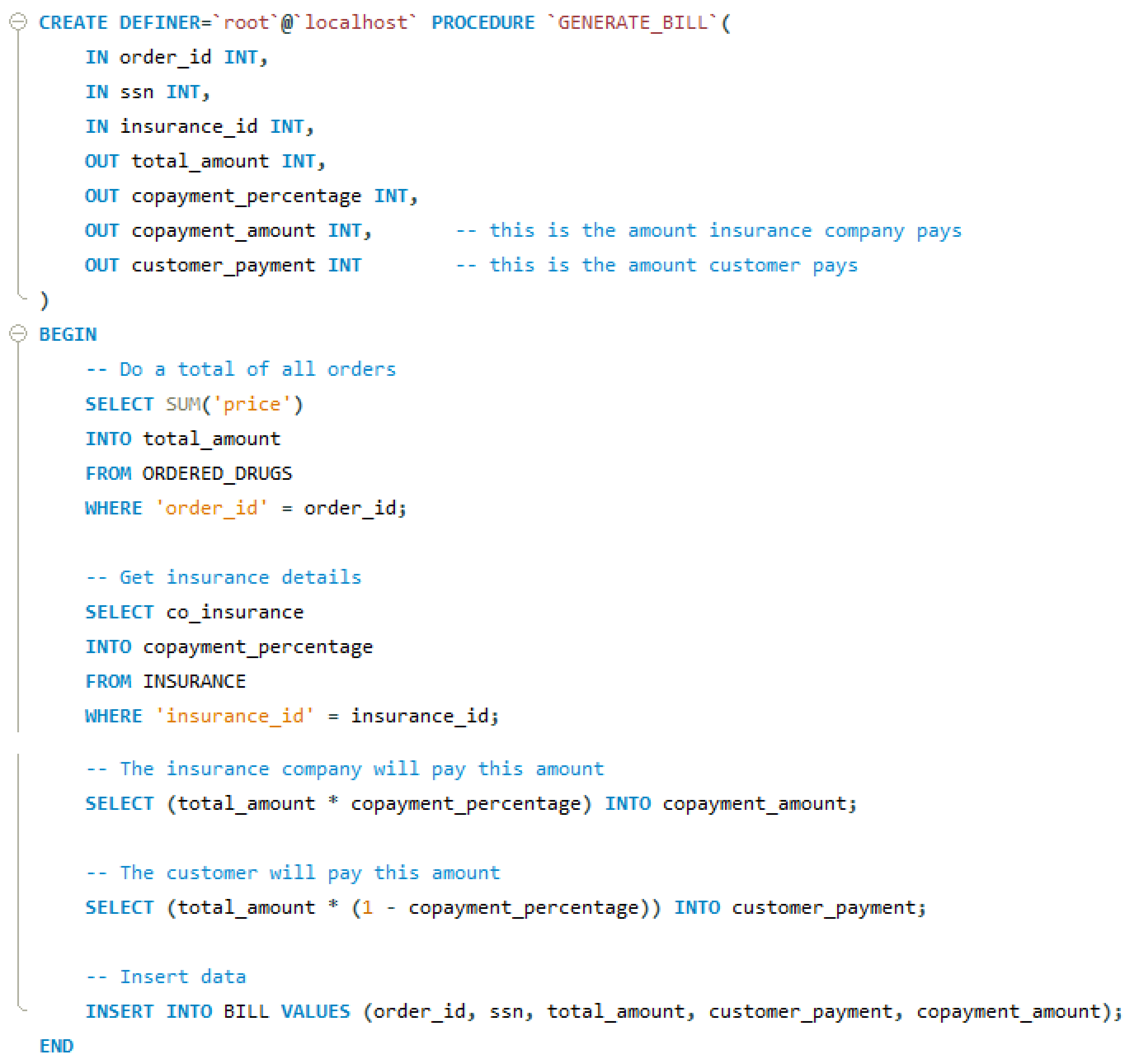
We have 3 stored procedure

1. Generate Bill
2. Report Expiring Drugs
3. Send Notifications

- ### Generate Bill

Making payments is slightly different in the pharmacy as customers may have insurance that supports co-payment. It means that insurance company pays a part of the bill while the patient pays rest of it. The data is stored in the database in the insurance table.

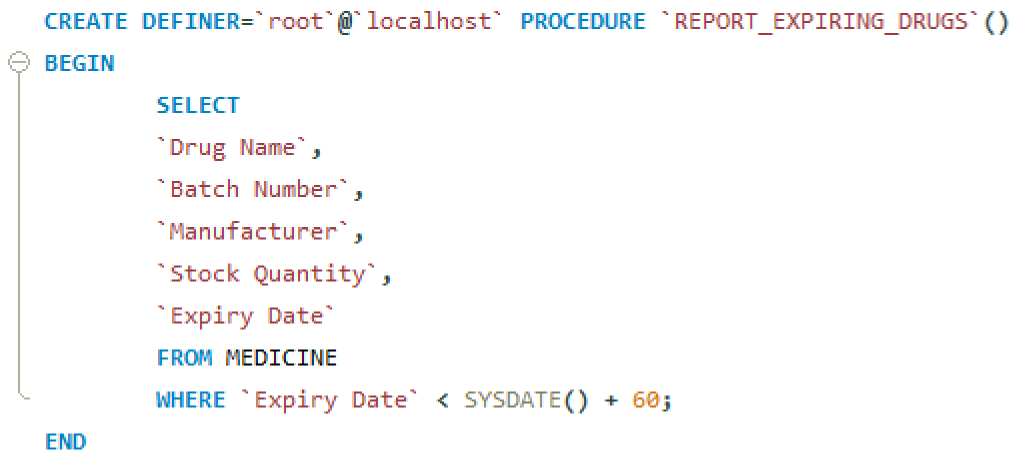
When the bill is generated, total amount is calculated based on ordered drugs and then copayment and customer payment is automatically calculated.



```sql
CREATE DEFINER=`root`@`localhost` PROCEDURE `GENERATE_BILL`(
    IN order_id INT,
    IN ssn INT,
    IN insurance_id INT,
    OUT total_amount INT,
    OUT copayment_percentage INT,
    OUT copayment_amount INT,        -- this is the amount insurance company pays
    OUT customer_payment INT         -- this is the amount customer pays
)
BEGIN
    -- Do a total of all orders
    SELECT SUM('price')
    INTO total_amount
    FROM ORDERED_DRUGS
    WHERE 'order_id' = order_id;

    -- Get insurance details
    SELECT co_insurance
    INTO copayment_percentage
    FROM INSURANCE
    WHERE 'insurance_id' = insurance_id;

    -- The insurance company will pay this amount
    SELECT (total_amount * copayment_percentage) INTO copayment_amount;

    -- The customer will pay this amount
    SELECT (total_amount * (1 - copayment_percentage)) INTO customer_payment;

    -- Insert data
    INSERT INTO BILL VALUES (order_id, ssn, total_amount, customer_payment, copayment_amount);
END
```

- **Report Expiring Drugs**

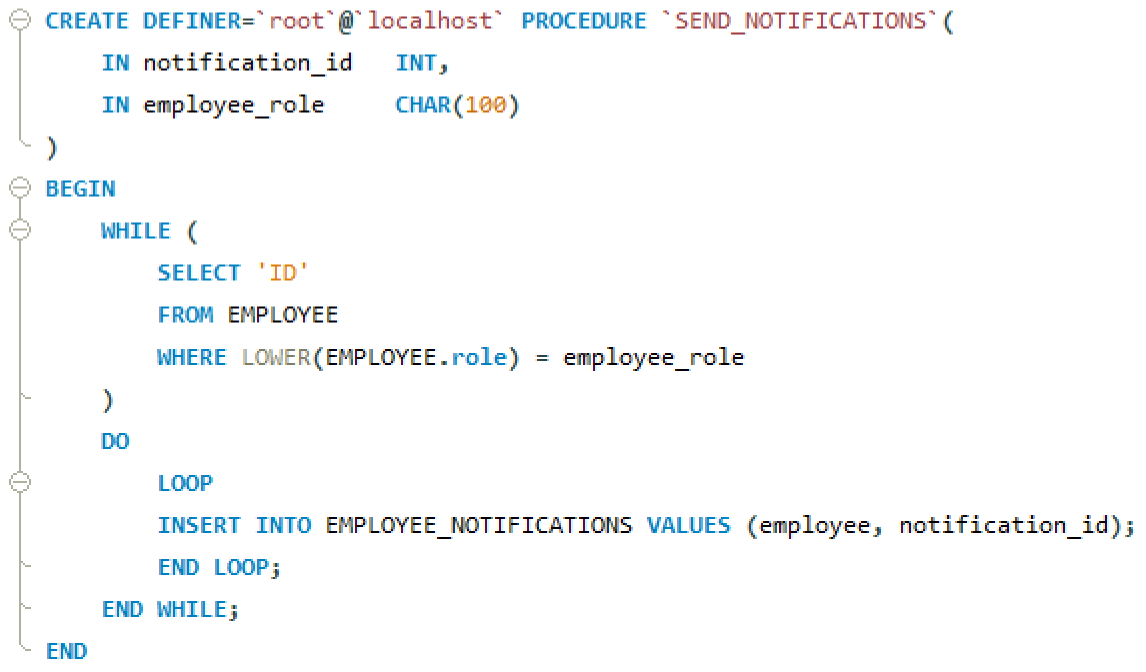
Any drugs that are going to expire within 60 days are displayed on screen along with their quantity and batch number.



```
CREATE DEFINER=`root`@`localhost` PROCEDURE `REPORT_EXPIRING_DRUGS`()
BEGIN
        SELECT
        `Drug Name`,
        `Batch Number`,
        `Manufacturer`,
        `Stock Quantity`,
        `Expiry Date`
        FROM MEDICINE
        WHERE `Expiry Date` < SYSDATE() + 60;
END
```

- **Send Notifications**

Sending of notifications is recorded in “EMPLOYEE_NOTIFICATION” relation. We are assuming that a third party tool will pick notifications from there and send to the relevant employees. Email/Text is out of scope for the project.



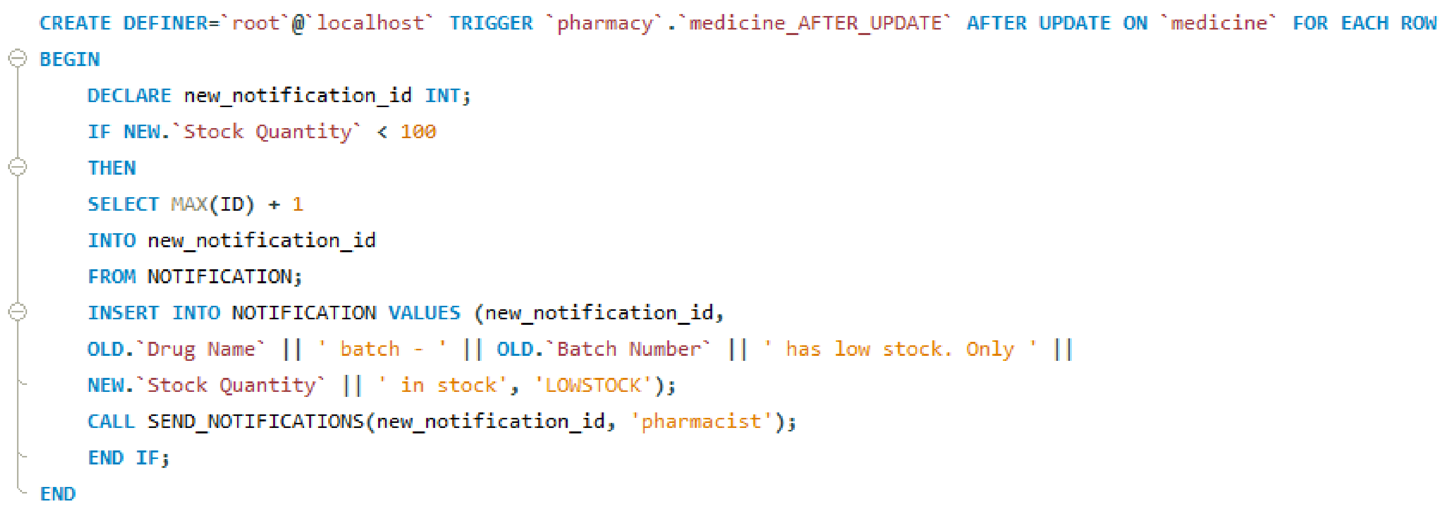
```
CREATE DEFINER=`root`@`localhost` PROCEDURE `SEND_NOTIFICATIONS`(
    IN notification_id   INT,
    IN employee_role     CHAR(100)
)
BEGIN
    WHILE (
        SELECT 'ID'
        FROM EMPLOYEE
        WHERE LOWER(EMPLOYEE.role) = employee_role
    )
    DO
        LOOP
        INSERT INTO EMPLOYEE_NOTIFICATIONS VALUES (employee, notification_id);
        END LOOP;
    END WHILE;
END
```

Triggers

We have a total of 3 triggers for following 2 tasks:

- ### Low Stock Alert

Low stock notification can be automatically sent when adding a medicine to the order makes the quantity too low for a medicine.



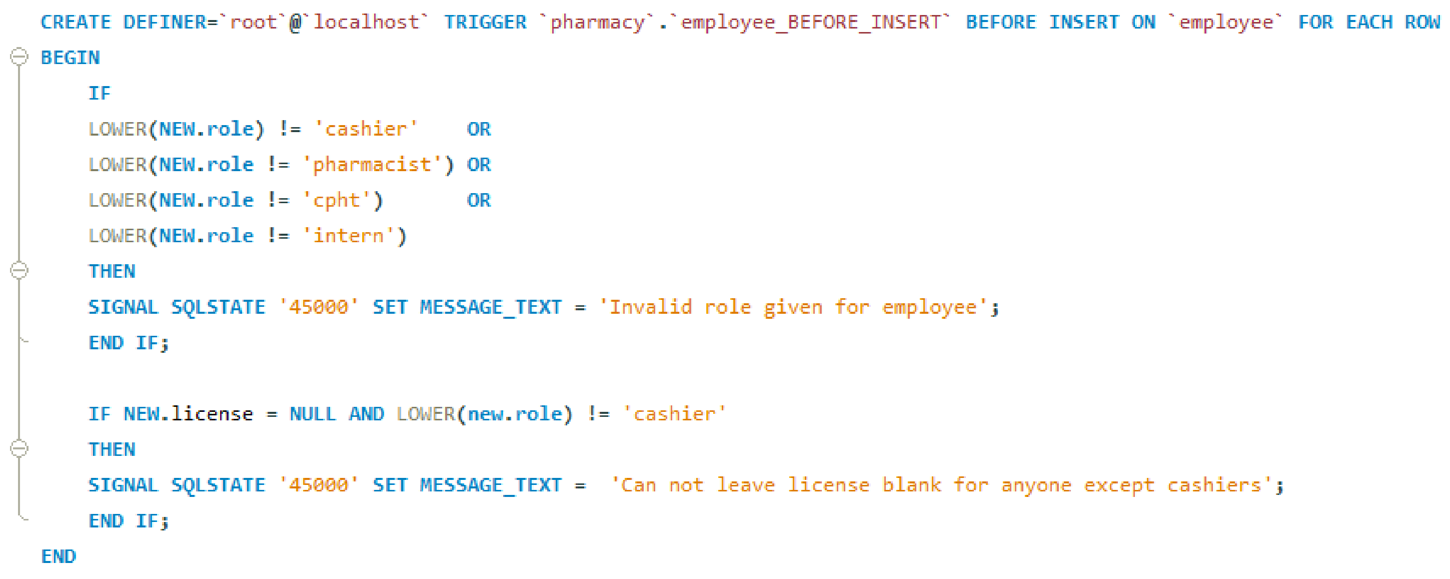
```
CREATE DEFINER=`root`@`localhost` TRIGGER `pharmacy`.`medicine_AFTER_UPDATE` AFTER UPDATE ON `medicine` FOR EACH ROW
BEGIN
    DECLARE new_notification_id INT;
    IF NEW.`Stock Quantity` < 100
    THEN
    SELECT MAX(ID) + 1
    INTO new_notification_id
    FROM NOTIFICATION;
    INSERT INTO NOTIFICATION VALUES (new_notification_id,
    OLD.`Drug Name` || ' batch - ' || OLD.`Batch Number` || ' has low stock. Only ' ||
    NEW.`Stock Quantity` || ' in stock', 'LOWSTOCK');
    CALL SEND_NOTIFICATIONS(new_notification_id, 'pharmacist');
    END IF;
END
```

- ### Employee Validation

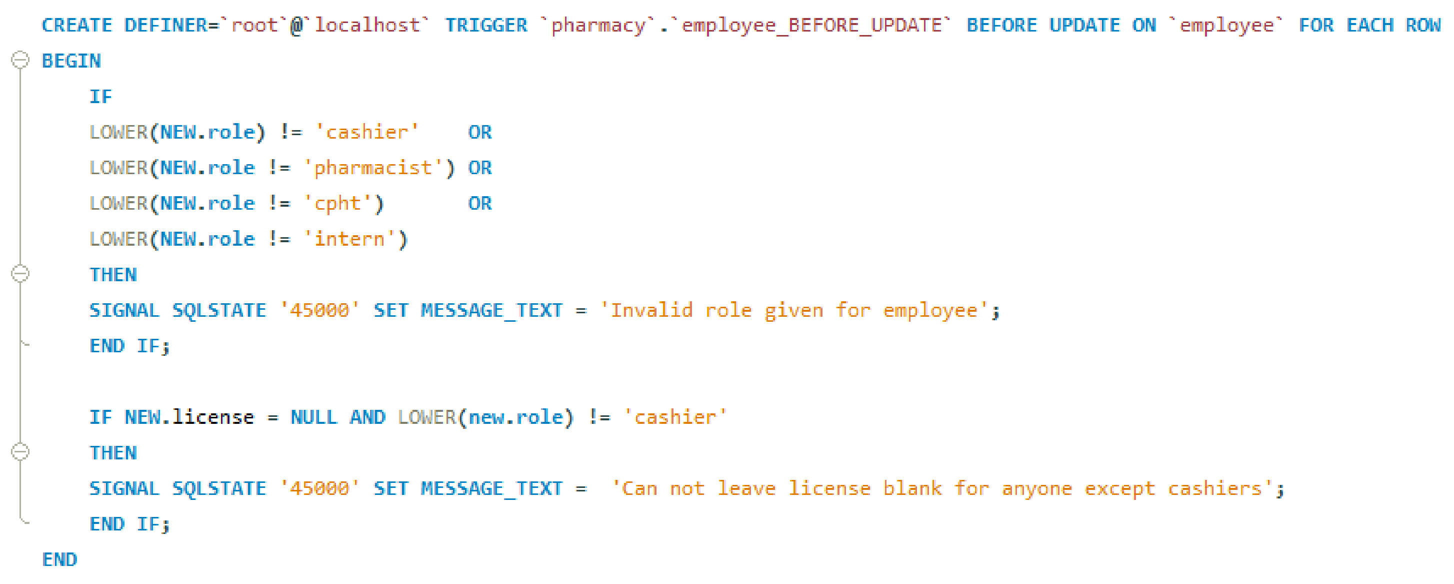
When an employee is added, the employee type should be one of:

1. Pharmacist
2. CPhT
3. Intern
4. Cashier

Except cashier, every other role requires a license.



```
CREATE DEFINER=`root`@`localhost` TRIGGER `pharmacy`.`employee_BEFORE_INSERT` BEFORE INSERT ON `employee` FOR EACH ROW
BEGIN
    IF
    LOWER(NEW.role) != 'cashier'    OR
    LOWER(NEW.role != 'pharmacist') OR
    LOWER(NEW.role != 'cpht')       OR
    LOWER(NEW.role != 'intern')
    THEN
    SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT = 'Invalid role given for employee';
    END IF;

    IF NEW.license = NULL AND LOWER(new.role) != 'cashier'
    THEN
    SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT =  'Can not leave license blank for anyone except cashiers';
    END IF;
END
```



```
CREATE DEFINER=`root`@`localhost` TRIGGER `pharmacy`.`employee_BEFORE_UPDATE` BEFORE UPDATE ON `employee` FOR EACH ROW
BEGIN
    IF
    LOWER(NEW.role) != 'cashier'    OR
    LOWER(NEW.role != 'pharmacist') OR
    LOWER(NEW.role != 'cpht')       OR
    LOWER(NEW.role != 'intern')
    THEN
    SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT = 'Invalid role given for employee';
    END IF;

    IF NEW.license = NULL AND LOWER(new.role) != 'cashier'
    THEN
    SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT =  'Can not leave license blank for anyone except cashiers';
    END IF;
END
```

Conclusion

The pharmacy project was a good learning experience for implementing a real world DBMS and helped us understand the nuances of a full implementation.

The most interesting part was the experience of starting from real world and then translating the concepts into the terms of a DBMS. The final implementation is robust and can handle various edge cases and scenarios. Paired with a capable application front end, it can handle day to day operations for a pharmacy.

Acknowledgement

Above all the praise goes to ALMIGHTY ALLAH with whom all the sovereignty, WHO bestowed us the essence of knowledge and the sight to differentiate between the right & wrong.

We will greatly like to thank our teacher, **Dr. Farrukh Arslan** who assigned us this project to enhance our knowledge and information regarding this course “Database Engineering”.

Thank You