Mini project report on

Medicine Supply Management System

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

Bachelor of Technology

in

Computer Science & Engineering

UE22CS351A – DBMS Project

Submitted by:

Tejas A Hombal	**PES2UG23CS824**
Veeresha	**PES2UG23CS825**

Under the guidance of
Prof. Nivedita Kasturi
Assistant Professor
PES University
AUG - DEC 2024

DEPARTMENT OF COMPUTER SCIENCE AND 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

Medicine Supply Management System

is a bonafide work carried out by

Tejas A Hombal	**PES2UG23CS824**
Veeresha	**PES2UG23CS825**

In partial fulfilment for the completion of fifth semester DBMS Project (UE22CS351A) 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. 2024 – DEC. 2024. 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
Prof. Nivedita Kasturi
Assistant Professor

DECLARATION

We hereby declare that the DBMS Project entitled **University Fest Management System** has been carried out by us under the guidance of **Prof. Nivedita Kasturi, Assistant 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 2024.

Tejas A Hombal	**PES2UG23CS824**	**<Signature>**
Veeresha	**PES2UG23CS825**	**<Signature>**

ABSTRACT

The **Medicine Supply Management System** is a comprehensive database-driven solution designed to streamline and optimize the supply chain of medicines for an online pharmacy platform. The system addresses crucial challenges in medicine inventory management, from maintaining accurate stock levels to ensuring timely processing of orders, minimizing expired stock, and supporting efficient interactions with suppliers and customers. This project offers an intuitive platform for tracking and managing the end-to-end journey of medicines—from suppliers to customers—ensuring reliability, accuracy, and efficiency in the supply chain.

The system's database is implemented using **MySQL**, with structured tables representing core entities, such as medicines, suppliers, products, customers, and purchase orders. These entities are managed through **CRUD operations** (Create, Read, Update, Delete), allowing users to perform essential tasks like adding new medicines, updating stock levels, handling customer orders, and tracking supplier details. **Python** is used for server-side operations, offering an easy-to-navigate platform for users to interact with the database.

Advanced features are integrated through **stored procedures, triggers, and functions** within MySQL, which automate critical tasks and ensure data integrity. For instance, triggers automatically update stock quantities when a purchase order is processed, while functions and stored procedures facilitate complex queries, such as calculating the total value of medicines in stock, identifying suppliers with expired stock, and generating summaries of customer orders. These features reduce the risk of manual errors, ensure real-time data accuracy, and enhance system efficiency by automating repetitive tasks.

In addition to practical functionality, the project report provides detailed documentation on database design and implementation. The **Entity-Relationship (ER) Model** captures the relationships between key entities, guiding the translation into **DDL (Data Definition Language) statements** for creating the database schema. The report also includes **DML (Data Manipulation Language) statements** for managing records, demonstrating best practices in structured data handling, and ensuring adherence to normalization principles.

This system is designed with scalability in mind, supporting potential growth to accommodate larger user bases and more extensive product inventories, while ensuring security through access controls that limit operations to authorized personnel. Such scalability and security make the platform suitable for use by both small-scale pharmacies and larger medical supply distributors, as it offers tailored solutions for a range of operational needs.

In conclusion, the Medicine Supply Management System provides an integrated solution to the complexities of pharmaceutical inventory management. By leveraging the power of a relational database, the project enhances the efficiency and reliability of supply chain operations. This platform demonstrates the potential of database management systems in the healthcare sector, contributing to more effective medicine distribution practices.

TABLE OF CONTENTS

Chapter 1:
INTRODUCTION

The **Medicine Supply Management System** is designed to address the critical needs of managing pharmaceutical inventory in a streamlined, efficient, and accessible way. With the continuous growth of online pharmacies and the increasing demand for fast, reliable, and safe access to medicines, effective supply chain management in the pharmaceutical sector has become more vital than ever. This project tackles the complex requirements of inventory control, order management, supplier and customer tracking, and real-time stock monitoring, all of which are essential for a well-functioning medical supply system.

This project's purpose is to create a centralized database solution that manages the entire workflow, from acquiring medicine stock from suppliers to distributing orders to customers. With a focus on accuracy, automation, and user accessibility, this system aims to replace or supplement traditional, error-prone manual processes with a digital platform that ensures data consistency, minimizes stock-outs, reduces expired stock, and supports quick access to essential records. Leveraging MySQL for the database, Python for backend operations, , this system allows for seamless interactions and data flow, thereby optimizing each stage of the medicine supply process.

The significance of the Medicine Supply Management System lies in its potential to improve healthcare operations by ensuring the timely availability of medicines and reducing the chances of stock wastage due to expired products. This solution is especially relevant in today's digital healthcare landscape, where customers increasingly rely on online platforms to meet their medical needs. The ability to automate and streamline inventory management, including real-time stock tracking and low-stock alerts, means that suppliers can proactively manage their inventory and customers can reliably access the medicines they need.

The primary objectives of this project are:

1. **Enhancing Operational Efficiency**: Through automated processes for managing stock levels, orders, and supplier interactions, the system reduces manual workload and operational costs.
2. **Improving Inventory Control**: By enabling real-time updates on stock levels and tracking of expiry dates, the system ensures that adequate inventory is maintained and prevents wastage due to expired medicines.
3. **Streamlining Order Management**: Automated order processing from customers ensures prompt and accurate fulfillment, leading to better customer satisfaction and trust.
4. **Ensuring Data Accuracy and Consistency**: Centralized data management eliminates errors associated with manual record-keeping, ensuring that all stakeholders have access to accurate information.
5. **Supporting Scalability and Adaptability**: The system is designed to be scalable, capable of handling growing data as the business expands, and adaptable to future upgrades or integrations with other software systems.

Overall, this project not only addresses the specific challenges in medicine supply chain management but also aligns with the broader objectives of enhancing healthcare access and operational efficiency. By developing a robust, database-driven management system, this project demonstrates the practical applications of database systems in the healthcare sector and showcases the benefits of leveraging technology to improve critical processes in medical supply management. The following chapters provide a detailed breakdown of the system's design, implementation, and functionalities, documenting the systematic approach taken to develop a reliable and user-friendly medicine supply management solution.

Chapter 2:
PROBLEM DEFINITION WITH USER REQUIREMENT SPECIFICATIONS

The **Medicine Supply Management System** is designed to resolve challenges and inefficiencies prevalent in traditional medicine supply chains. In a typical scenario, managing pharmaceutical inventory requires tracking large volumes of data on stock levels, supplier orders, expiry dates, and customer transactions. Manual or semi-automated systems often lead to issues such as stockouts, overstocking, delayed order processing, and inconsistent data tracking, ultimately impacting customer satisfaction and operational efficiency.

Problem Definition

The key issues identified in the existing medicine supply processes include:

1. **Stockouts and Excess Inventory**: Without real-time tracking and forecasting capabilities, supply chains may either run out of critical medicines or accumulate excess stock, leading to expired products and financial loss.
2. **Inaccurate Order Tracking**: Manual systems make it difficult to track orders accurately, often leading to delays, missed orders, or incorrect deliveries, which impacts both suppliers and customers.
3. **Lack of Inventory Visibility**: Current systems often lack a centralized database, making it challenging to monitor stock levels across different locations and suppliers. This results in inefficiencies in stock allocation and replenishment.
4. **Expiration Management**: Pharmaceuticals have strict expiration timelines, yet many systems fail to monitor expiration dates effectively, leading to wastage and potential health risks if expired products reach customers.
5. **Inefficient Report Generation**: Gathering insights into inventory levels, order fulfillment, and supplier performance is crucial for strategic planning. However, traditional methods require extensive manual work and are prone to errors.

User Requirement Specifications

To address these issues, the system must meet specific requirements for different users: suppliers, customers, and administrators. Each requirement is designed to ensure the efficient and seamless management of the medicine supply chain.

1. **User Authentication and Role-Based Access Control**
   - Each user (supplier, customer, administrator) should have a secure login with role-based access to relevant functions, ensuring data privacy and security.
   - Administrators will have full access to manage inventory, suppliers, orders, and customer data, while suppliers and customers can only view or modify data related to their specific roles.
2. **Inventory Management and Stock Updates**
   - The system should allow administrators to add, update, or remove products, with automatic updates to stock levels when orders are placed.
   - Stock alerts must notify administrators of low inventory or expiring products, enabling proactive restocking or discounting of soon-to-expire items.
3. **Order Processing and Tracking**
   - Customers should be able to place orders online, which are then tracked through each stage, from processing to fulfillment.
   - Order status updates should be available to both customers and administrators, providing transparency and ensuring timely delivery.
4. **Supplier Management**

   - Suppliers should be able to view and manage their product contributions, including stock quantities and product details, to ensure availability and alignment with the platform's stock requirements.
   - The system should support supplier reporting and order history, allowing for easy coordination with administrators on inventory needs.
5. **Expiry and Waste Management**
   - The system must provide functionality to track the expiry dates of all medicines, with automated notifications to administrators when stock is nearing its expiry.
   - Reporting features for expired stock should enable administrators to take actions, such as removing products from the available inventory.
6. **Data Reporting and Analytics**
   - Administrators require access to reports on inventory levels, sales, supplier performance, and customer purchasing trends. Automated reports should be generated at scheduled intervals and upon request.
   - Analyzing data trends will support better decision-making in purchasing, inventory planning, and marketing strategies.
7. **User-Friendly Interface**
   - The system interface should be intuitive, with clear navigation and easy-to-use forms for order placement, inventory updates, and report generation.
   - Clear instructions and validations should ensure a smooth experience for all users, minimizing errors and maximizing productivity.

The specifications in this chapter outline a robust set of requirements that address the critical aspects of medicine supply chain management. By meeting these requirements, the Medicine Supply Management System will enhance operational efficiency, reduce stock-related issues, and improve the overall experience for suppliers, customers, and administrators alike.

Chapter 3:
LIST OF SOFTWARES/TOOLS/PROGRAMMING LANGUAGES USED

This chapter presents the main software tools and programming languages used to develop the Medicine Supply Management System. Each tool is integral to the system's functionality, from database management to user interface creation.

1. **MySQL**: MySQL serves as the database management system for this project. It is used to store, manage, and organize data related to medicines, suppliers, customers, products, and orders. MySQL provides a robust platform for handling large volumes of data, ensuring data integrity and supporting essential database operations like queries, triggers, and stored procedures.

2. **MySQL Workbench**: MySQL Workbench is utilized as a visual tool for database design, modeling, and administration. It provides a user-friendly interface for creating and managing the database schema, enabling the team to visualize relationships, create ER diagrams, and easily perform SQL operations.

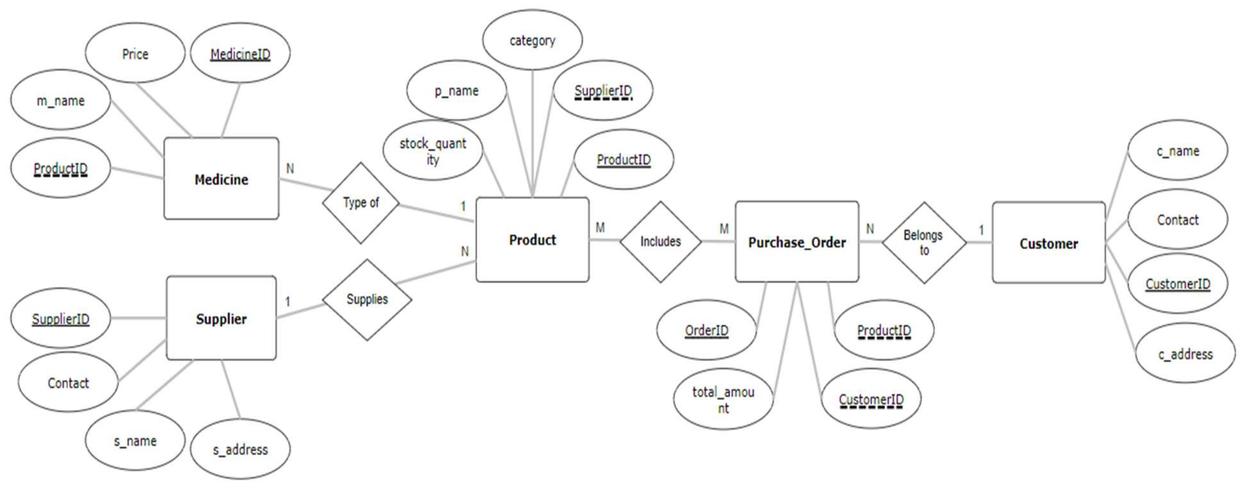
3. **Python**: Python is the primary programming language for backend development in this project. It handles server-side operations, such as executing SQL commands, managing CRUD operations, and integrating the database with the front end. Python's extensive libraries for database handling and data processing make it an ideal choice for backend development.

4. **Streamlit**: Streamlit is used to develop the front-end user interface for the system. This Python-based framework allows rapid prototyping and deployment of web applications. In this project, Streamlit is used to create intuitive user interfaces for CRUD operations, ensuring users can seamlessly interact with the database to manage medicines, orders, and other system entities.

Each of these tools is selected based on its unique capabilities and compatibility with the system's goals. MySQL and MySQL Workbench ensure database reliability and effective data modeling, while Python and Streamlit enable efficient backend processes and user-friendly front-end interactions, creating a cohesive and functional management system for the medicine supply chain.

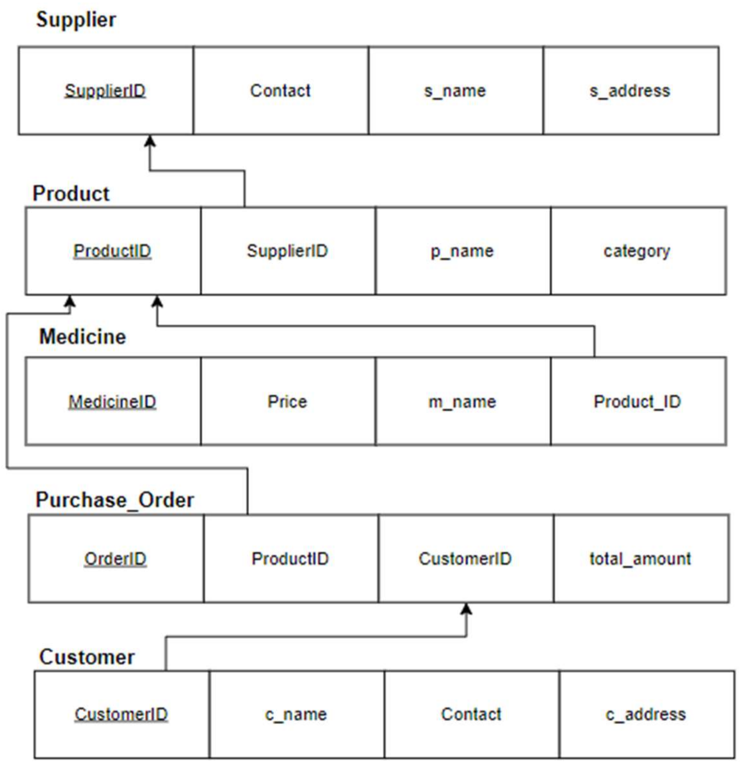
Chapter 4:
ER MODEL

The Entity-Relationship (ER) model visually represents the primary entities (Product, Medicine, Supplier, Customer, Purchase Order) and their relationships within the Medicine Supply Management System. This chapter includes a detailed diagram and descriptions of each entity, along with attributes and their connections to other entities.



Chapter 5:
ER TO RELATIONAL MAPPING

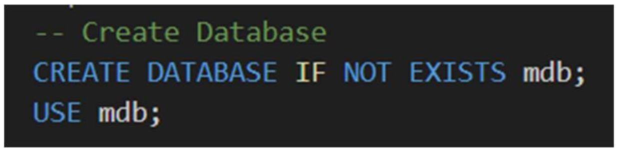
This chapter covers the transformation of the ER model into relational tables. It includes the mapping of entities to relations, detailing the primary keys, foreign keys, and constraints. This ensures data integrity and a well-structured database schema, setting the foundation for efficient data retrieval and manipulation.



Chapter 6:
DDL STATEMENTS

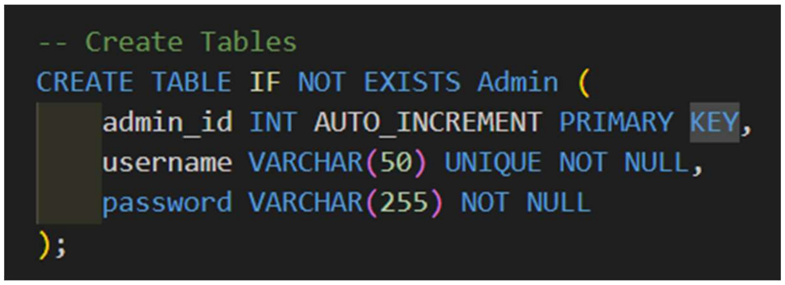
The DDL (Data Definition Language) chapter includes the SQL statements used to define the database schema. This section details the creation of tables, primary and foreign keys, indexes, and constraints necessary for establishing the database structure.

Database Name: mdb
The database mdb represents a medicine supply management system. It is structured to handle the administration, supplier details, products, medicines, customer data, purchase orders, and a medicine log for tracking changes.



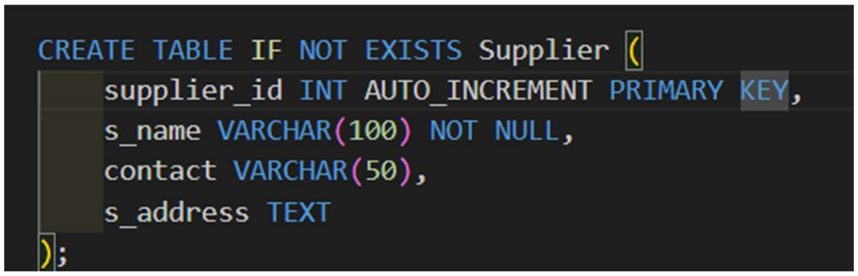
```
-- Create Database
CREATE DATABASE IF NOT EXISTS mdb;
USE mdb;
```

Admin Table:



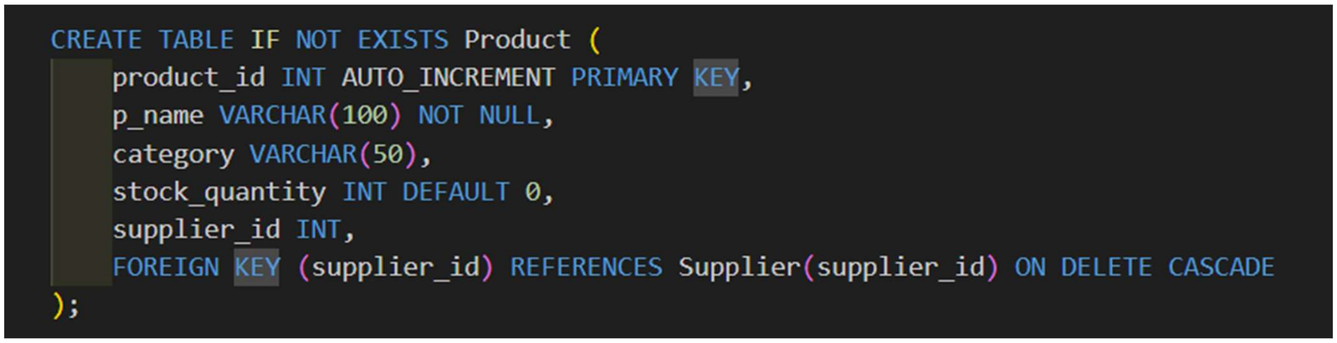
```
-- Create Tables
CREATE TABLE IF NOT EXISTS Admin (
    admin_id INT AUTO_INCREMENT PRIMARY KEY,
    username VARCHAR(50) UNIQUE NOT NULL,
    password VARCHAR(255) NOT NULL
);
```

Supplier Table:



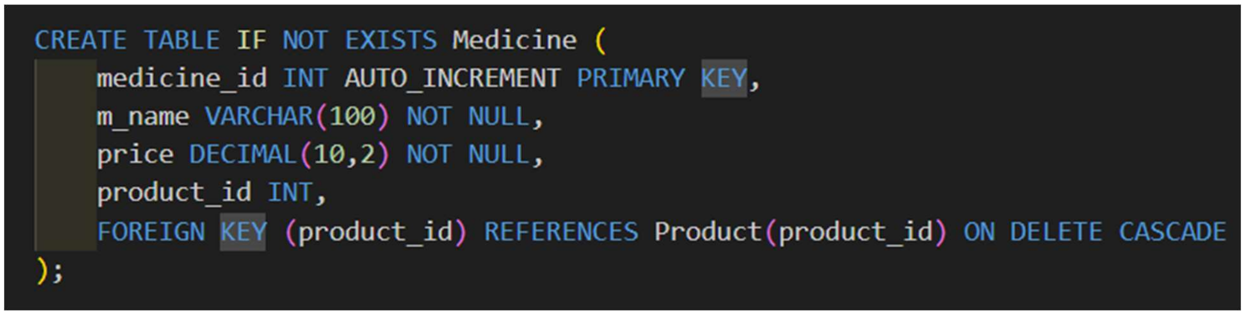
```
CREATE TABLE IF NOT EXISTS Supplier (
    supplier_id INT AUTO_INCREMENT PRIMARY KEY,
    s_name VARCHAR(100) NOT NULL,
    contact VARCHAR(50),
    s_address TEXT
);
```

Product Table:



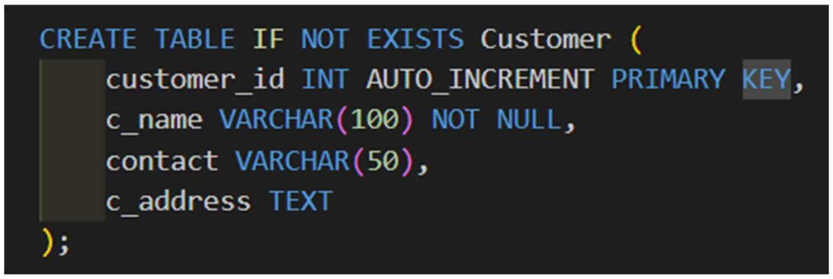
```
CREATE TABLE IF NOT EXISTS Product (
    product_id INT AUTO_INCREMENT PRIMARY KEY,
    p_name VARCHAR(100) NOT NULL,
    category VARCHAR(50),
    stock_quantity INT DEFAULT 0,
    supplier_id INT,
    FOREIGN KEY (supplier_id) REFERENCES Supplier(supplier_id) ON DELETE CASCADE
);
```

Medicine Table:



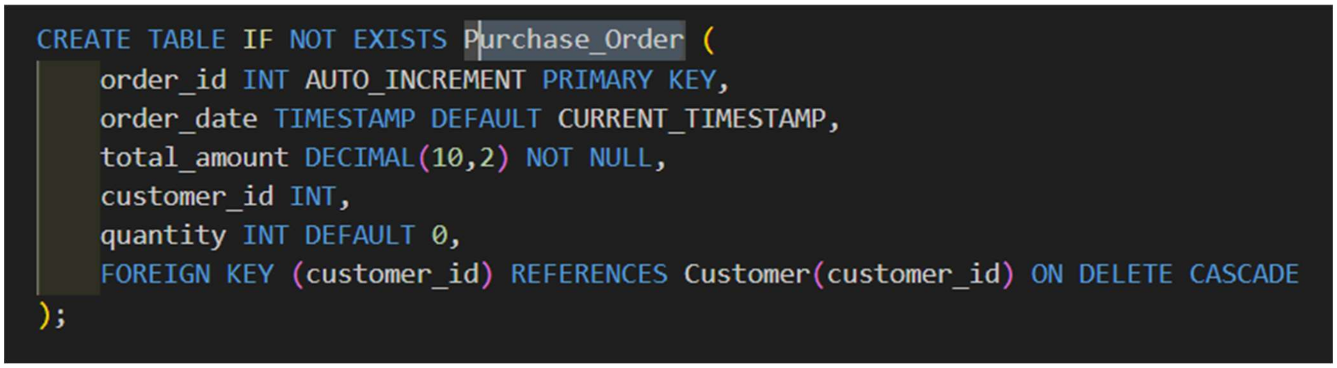
```
CREATE TABLE IF NOT EXISTS Medicine (
    medicine_id INT AUTO_INCREMENT PRIMARY KEY,
    m_name VARCHAR(100) NOT NULL,
    price DECIMAL(10,2) NOT NULL,
    product_id INT,
    FOREIGN KEY (product_id) REFERENCES Product(product_id) ON DELETE CASCADE
);
```

Customer Table:



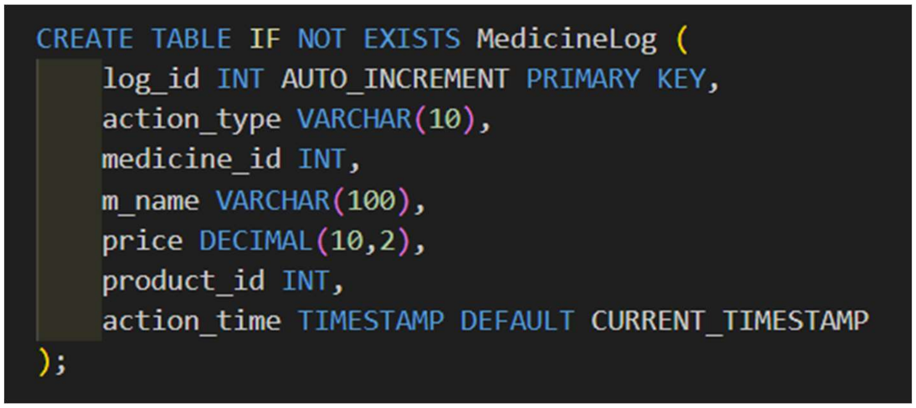
```
CREATE TABLE IF NOT EXISTS Customer (
    customer_id INT AUTO_INCREMENT PRIMARY KEY,
    c_name VARCHAR(100) NOT NULL,
    contact VARCHAR(50),
    c_address TEXT
);
```

Purchase_Order Table:



```
CREATE TABLE IF NOT EXISTS Purchase_Order (
    order_id INT AUTO_INCREMENT PRIMARY KEY,
    order_date TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
    total_amount DECIMAL(10,2) NOT NULL,
    customer_id INT,
    quantity INT DEFAULT 0,
    FOREIGN KEY (customer_id) REFERENCES Customer(customer_id) ON DELETE CASCADE
);
```

MedicineLog tables:

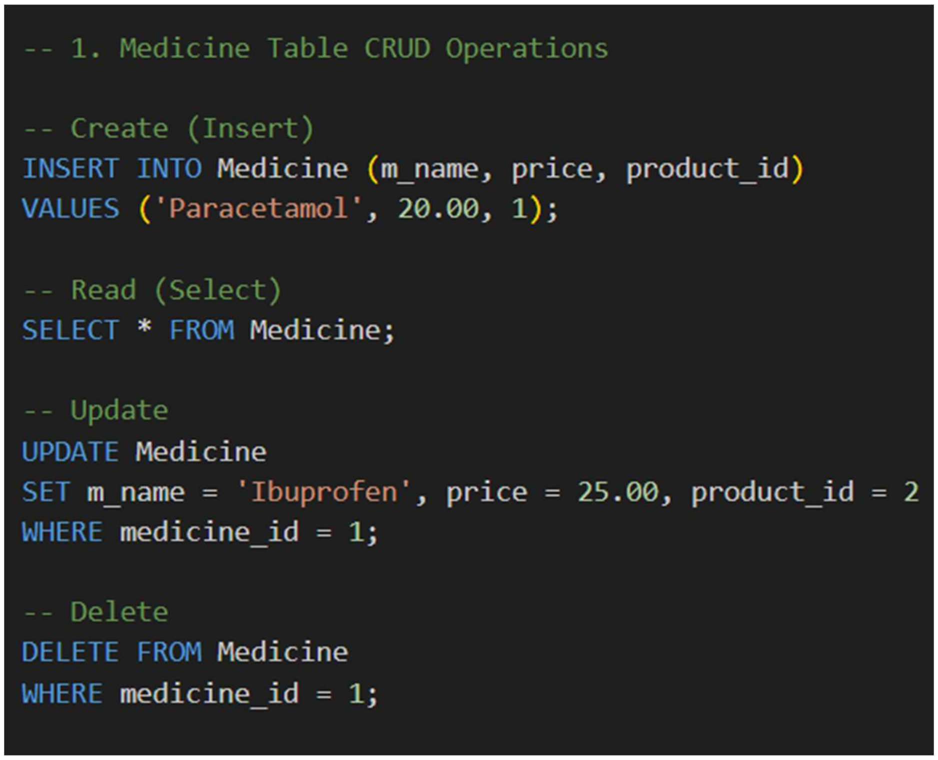


```sql
CREATE TABLE IF NOT EXISTS MedicineLog (
    log_id INT AUTO_INCREMENT PRIMARY KEY,
    action_type VARCHAR(10),
    medicine_id INT,
    m_name VARCHAR(100),
    price DECIMAL(10,2),
    product_id INT,
    action_time TIMESTAMP DEFAULT CURRENT_TIMESTAMP
);
```

Chapter 7:
DML STATEMENTS (CRUD OPERATION SCREENSHOTS)

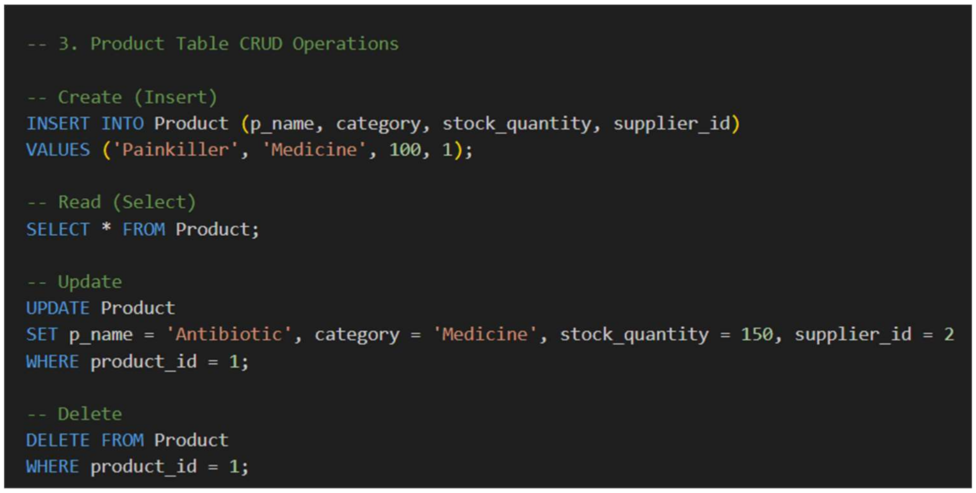
This chapter demonstrates the use of DML (Data Manipulation Language) statements for performing CRUD (Create, Read, Update, Delete) operations. Screenshots of these operations, such as adding new medicines, updating stock, and deleting outdated records, are provided.

Medicine Table CRUD Operations:



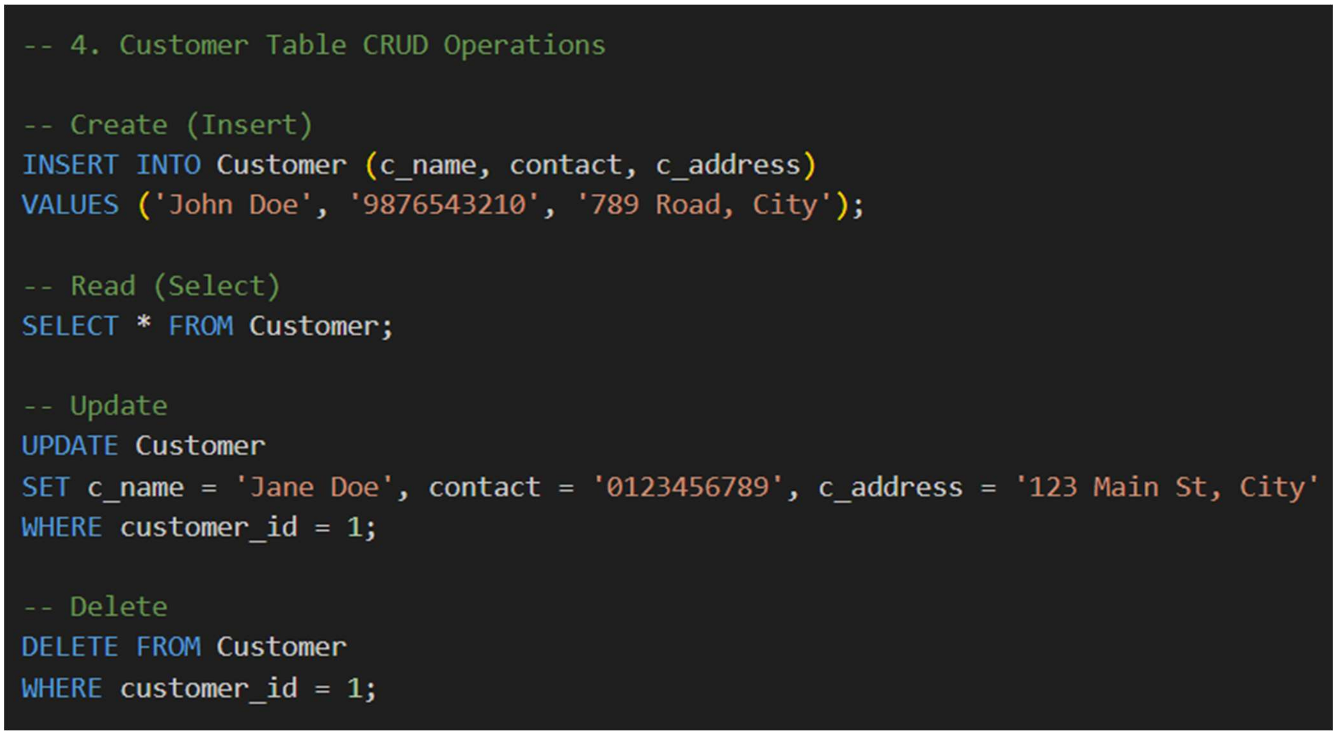
```sql
-- 1. Medicine Table CRUD Operations

-- Create (Insert)
INSERT INTO Medicine (m_name, price, product_id)
VALUES ('Paracetamol', 20.00, 1);

-- Read (Select)
SELECT * FROM Medicine;

-- Update
UPDATE Medicine
SET m_name = 'Ibuprofen', price = 25.00, product_id = 2
WHERE medicine_id = 1;

-- Delete
DELETE FROM Medicine
WHERE medicine_id = 1;
```

Supplier Table CRUD Operations:



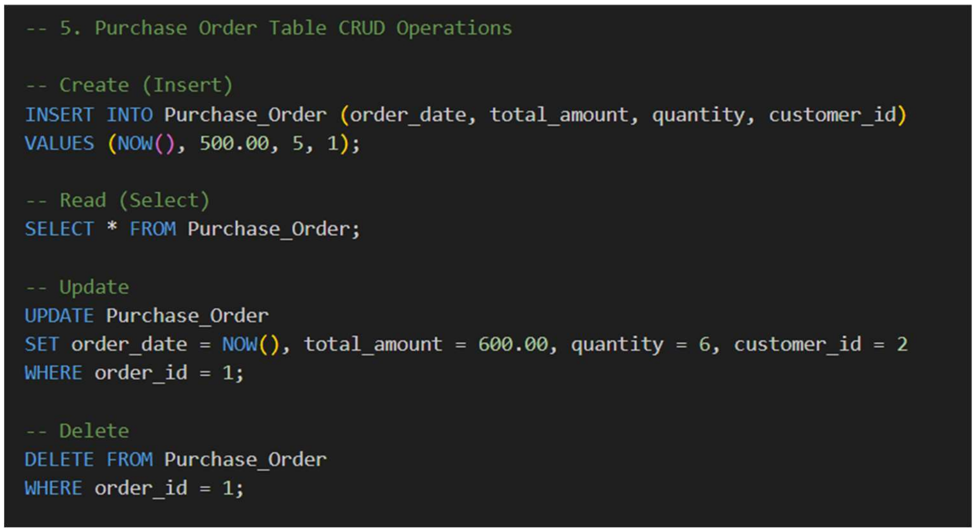
```sql
-- 3. Product Table CRUD Operations

-- Create (Insert)
INSERT INTO Product (p_name, category, stock_quantity, supplier_id)
VALUES ('Painkiller', 'Medicine', 100, 1);

-- Read (Select)
SELECT * FROM Product;

-- Update
UPDATE Product
SET p_name = 'Antibiotic', category = 'Medicine', stock_quantity = 150, supplier_id = 2
WHERE product_id = 1;

-- Delete
DELETE FROM Product
WHERE product_id = 1;
```

Product Table CRUD Operations:



```
-- 4. Customer Table CRUD Operations

-- Create (Insert)
INSERT INTO Customer (c_name, contact, c_address)
VALUES ('John Doe', '9876543210', '789 Road, City');

-- Read (Select)
SELECT * FROM Customer;

-- Update
UPDATE Customer
SET c_name = 'Jane Doe', contact = '0123456789', c_address = '123 Main St, City'
WHERE customer_id = 1;

-- Delete
DELETE FROM Customer
WHERE customer_id = 1;
```

Purchase Order Table CRUD Operations:

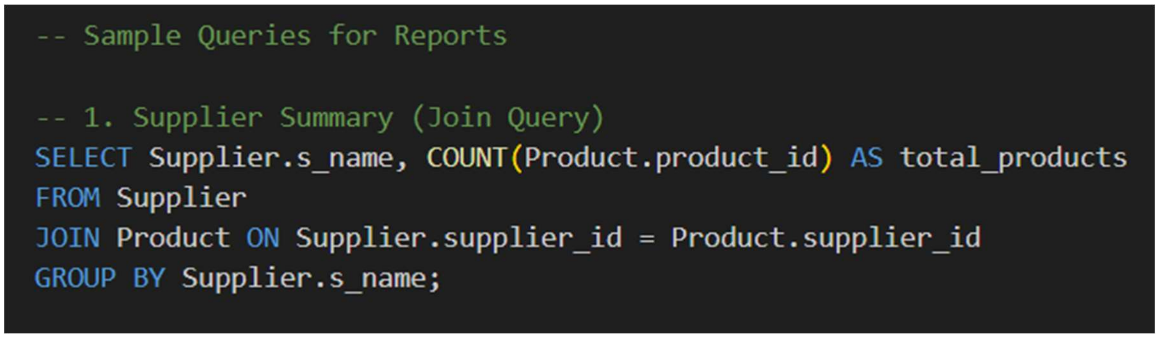


```
-- 5. Purchase Order Table CRUD Operations

-- Create (Insert)
INSERT INTO Purchase_Order (order_date, total_amount, quantity, customer_id)
VALUES (NOW(), 500.00, 5, 1);

-- Read (Select)
SELECT * FROM Purchase_Order;

-- Update
UPDATE Purchase_Order
SET order_date = NOW(), total_amount = 600.00, quantity = 6, customer_id = 2
WHERE order_id = 1;

-- Delete
DELETE FROM Purchase_Order
WHERE order_id = 1;
```

Chapter 8:
QUERIES (JOIN QUERY, AGGREGATE FUNCTION QUERIES, AND NESTED QUERY)

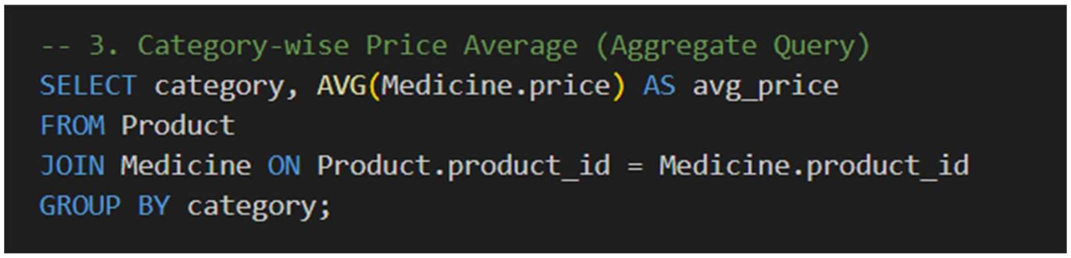
This section includes key SQL queries to retrieve and analyze data effectively. Examples include join queries to link data across tables, aggregate functions for inventory analysis, and nested queries for complex filtering and sorting tasks.

Join query:



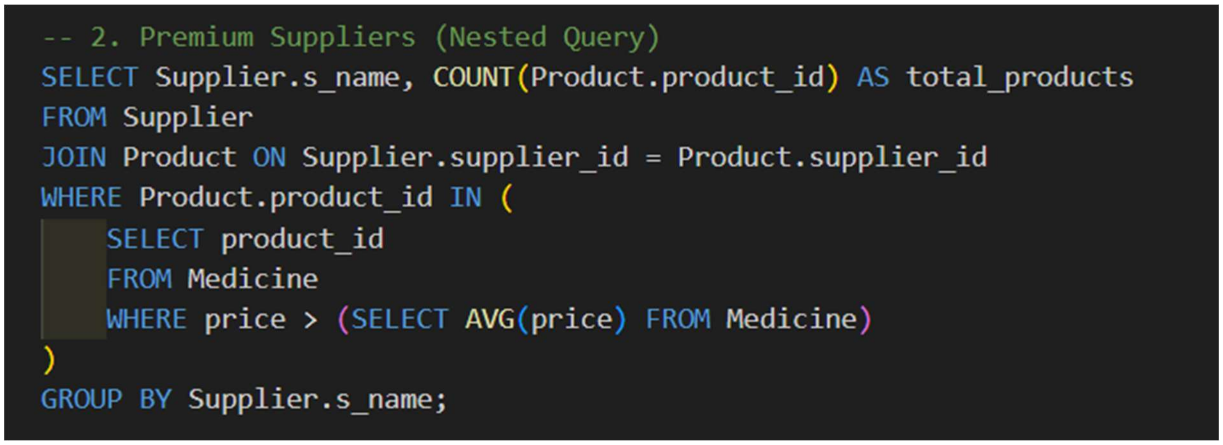
```sql
-- Sample Queries for Reports

-- 1. Supplier Summary (Join Query)
SELECT Supplier.s_name, COUNT(Product.product_id) AS total_products
FROM Supplier
JOIN Product ON Supplier.supplier_id = Product.supplier_id
GROUP BY Supplier.s_name;
```

Aggregate query:



```sql
-- 3. Category-wise Price Average (Aggregate Query)
SELECT category, AVG(Medicine.price) AS avg_price
FROM Product
JOIN Medicine ON Product.product_id = Medicine.product_id
GROUP BY category;
```

Nested query:

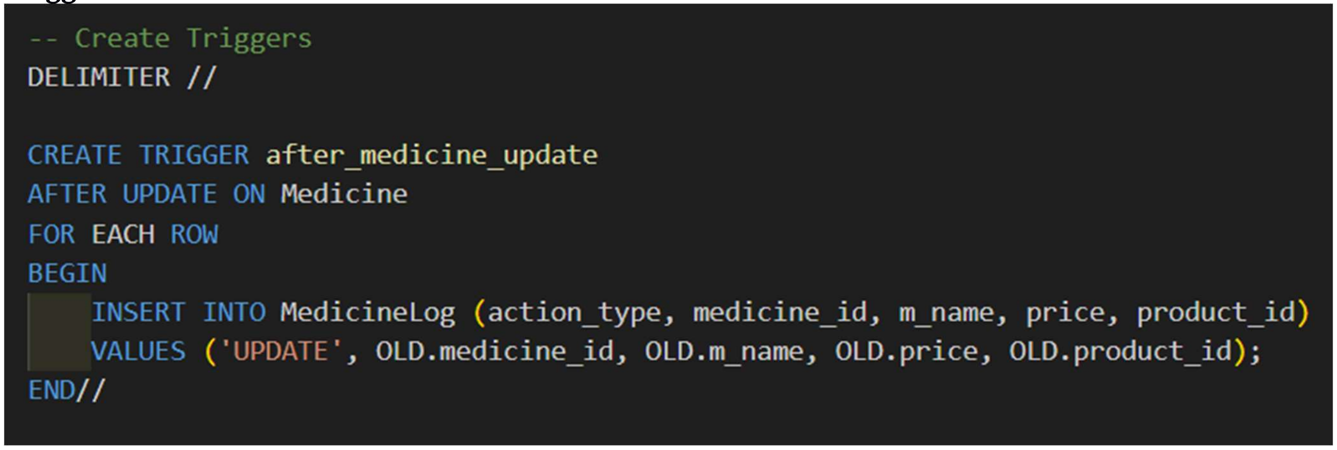


```sql
-- 2. Premium Suppliers (Nested Query)
SELECT Supplier.s_name, COUNT(Product.product_id) AS total_products
FROM Supplier
JOIN Product ON Supplier.supplier_id = Product.supplier_id
WHERE Product.product_id IN (
    SELECT product_id
    FROM Medicine
    WHERE price > (SELECT AVG(price) FROM Medicine)
)
GROUP BY Supplier.s_name;
```

Chapter 9:
STORED PROCEDURES, FUNCTIONS, AND TRIGGERS

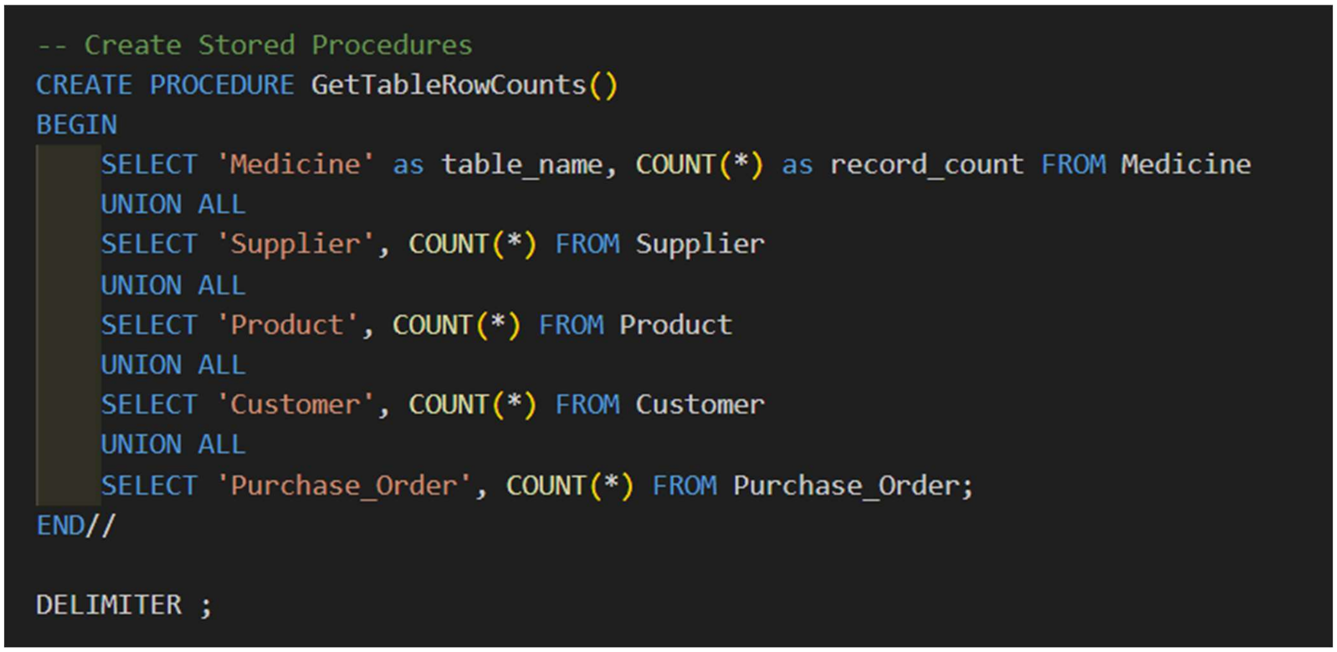
Stored procedures, functions, and triggers are implemented to automate tasks like stock updates, order processing, and alerts for low inventory. The chapter provides definitions, code, and descriptions of each procedure and function, highlighting the importance of database automation in maintaining an efficient supply chain.

Trigger function:



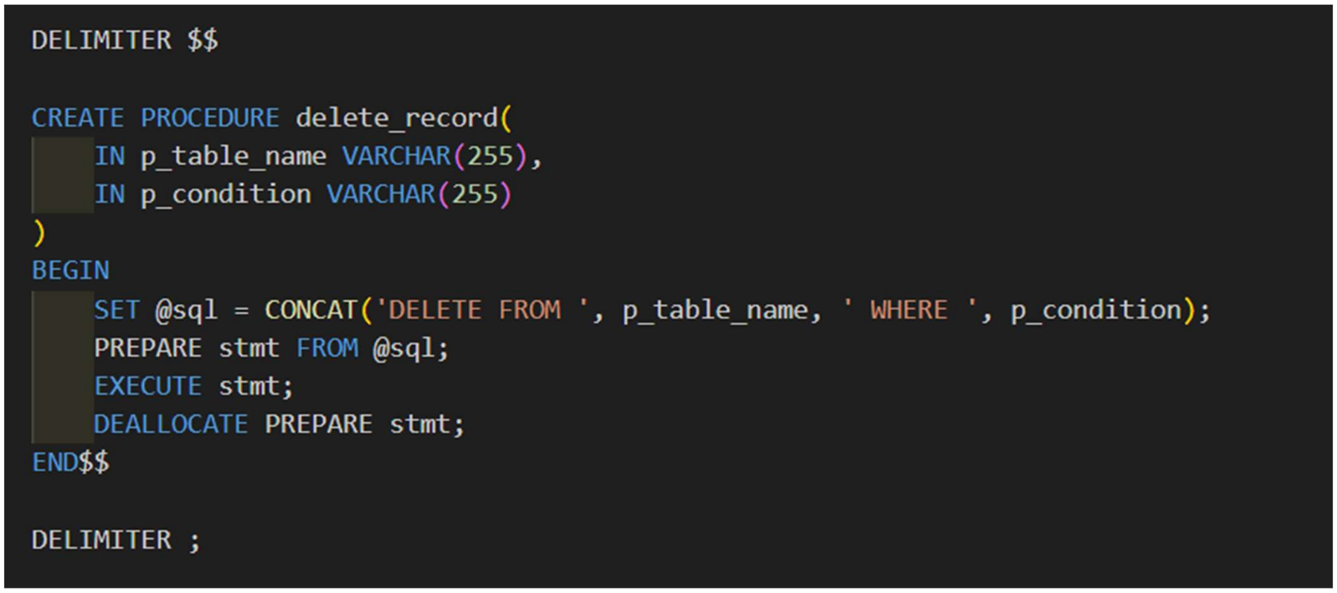
```
-- Create Triggers
DELIMITER //

CREATE TRIGGER after_medicine_update
AFTER UPDATE ON Medicine
FOR EACH ROW
BEGIN
    INSERT INTO MedicineLog (action_type, medicine_id, m_name, price, product_id)
    VALUES ('UPDATE', OLD.medicine_id, OLD.m_name, OLD.price, OLD.product_id);
END//
```

Procedure query:



```
-- Create Stored Procedures
CREATE PROCEDURE GetTableRowCounts()
BEGIN
    SELECT 'Medicine' as table_name, COUNT(*) as record_count FROM Medicine
    UNION ALL
    SELECT 'Supplier', COUNT(*) FROM Supplier
    UNION ALL
    SELECT 'Product', COUNT(*) FROM Product
    UNION ALL
    SELECT 'Customer', COUNT(*) FROM Customer
    UNION ALL
    SELECT 'Purchase_Order', COUNT(*) FROM Purchase_Order;
END//

DELIMITER ;
```

Function quey:



```sql
DELIMITER $$

CREATE PROCEDURE delete_record(
    IN p_table_name VARCHAR(255),
    IN p_condition VARCHAR(255)
)
BEGIN
    SET @sql = CONCAT('DELETE FROM ', p_table_name, ' WHERE ', p_condition);
    PREPARE stmt FROM @sql;
    EXECUTE stmt;
    DEALLOCATE PREPARE stmt;
END$$

DELIMITER ;
```

Chapter 10
FRONT-END DEVELOPMENT (FUNCTIONALITIES/FEATURES OF THE APPLICATION)

10.1 Overview

The front end of the Medicine Supply Management System is designed using **Streamlit**, a lightweight and efficient Python framework that allows developers to quickly create interactive web applications. Streamlit's simplicity and real-time updating make it suitable for managing records in our system, offering an intuitive and organized way for users to interact with the database.

10.2 User Interface and Layout

The Streamlit application is structured with a main title and a sidebar, allowing users to easily navigate between different sections. The sidebar includes a drop-down menu with options for each category:

- **Medicine**
- **Supplier**
- **Product**
- **Customer**
- **Purchase Order**

Selecting any of these categories displays the respective CRUD (Create, Read, Update, Delete) operations on the main page, allowing users to view, add, update, or delete records.

10.3 Functionalities

10.3.1 Medicine Management

Users can perform the following actions under the "Medicine Management" section:

- **Add Medicine**: Enter medicine details, including name, price, product ID, type, expiry date, and quantity.
- **View Medicines**: Display all medicine records in a tabular format.
- **Update Medicine**: Modify existing medicine records by specifying the medicine ID and updating relevant fields.
- **Delete Medicine**: Remove a specific medicine record by entering the medicine ID.

10.3.2 Supplier Management

In the "Supplier Management" section, the following actions are available:

- **Add Supplier**: Input supplier details such as name, contact information, and address.
- **View Suppliers**: Display a table of all supplier records.
- **Update Supplier**: Update supplier details by entering the supplier ID and modifying relevant fields.
- **Delete Supplier**: Delete a supplier record by specifying the supplier ID.

10.3.3 Product Management

For "Product Management," users can manage product records with the following operations:

- **Add Product**: Enter details like product name, category, stock quantity, and supplier ID.
- **View Products**: Display all products in a tabular format.
- **Update Product**: Update product details by specifying the product ID and making necessary changes.
- **Delete Product**: Remove a product record by entering the product ID.

10.3.4 Customer Management

The "Customer Management" section provides options to manage customer records:

- **Add Customer**: Enter customer details, including name, contact information, and address.
- **View Customers**: Display a list of all customers in table format.
- **Update Customer**: Modify customer records by specifying the customer ID and making changes to relevant fields.
- **Delete Customer**: Delete a customer record by entering the customer ID.

10.3.5 Purchase Order Management

In the "Purchase Order Management" section, users can manage purchase orders with these options:

- **Add Purchase Order**: Input order details, such as total amount and customer ID.
- **View Purchase Orders**: Display all purchase orders in a table.
- **Update Purchase Order**: Update order details by entering the order ID and modifying the fields.
- **Delete Purchase Order**: Remove an order record by specifying the order ID.

10.4 Design and User Experience Considerations

- **Simple and Intuitive Navigation**: The sidebar menu allows for quick access to different sections of the management system.
- **Form Inputs and Buttons**: Each CRUD operation is facilitated by text inputs, number inputs, and buttons, making data entry straightforward.
- **Real-Time Updates**: Changes in the data are instantly reflected due to Streamlit's real-time updating, improving the user experience.
- **Error Handling**: Each action, such as adding, updating, or deleting records, includes a button confirmation to ensure the user's intent, minimizing errors.

10.5 Conclusion

The front-end interface developed using Streamlit provides a clean, interactive, and efficient way to manage the data within the Medicine Supply Management System. By leveraging Streamlit's components, we created a responsive application that meets the requirements of the system and supports seamless CRUD operations. This interface plays a vital role in managing the supply chain efficiently by allowing easy access and control over the database.

REFERENCES/BIBLIOGRAPHY

This section lists all references and sources used throughout the project, including research papers, textbooks, and online resources related to database design, web development, and supply chain management.

This section lists all sources and materials referred to or consulted during the development of the Medicine Supply Management System. Proper citation of resources not only acknowledges the authors' contributions but also provides credibility and a foundation for the project's methodologies and functionalities. Here are examples of potential references to include:

1. **Books and Textbooks**
    - Elmasri, R., & Navathe, S. (2015). *Fundamentals of Database Systems*. Pearson.
    - Connolly, T., & Begg, C. (2014). *Database Systems: A Practical Approach to Design, Implementation, and Management*. Pearson Education.

2. **Websites and Documentation**
    - MySQL Documentation. (n.d.). Retrieved from https://dev.mysql.com/doc/
    - Streamlit Documentation. (n.d.). Retrieved from https://docs.streamlit.io/
    - Python Documentation. (n.d.). Retrieved from https://docs.python.org/

3. **Research Papers and Articles**
    - Nadeem, S., & Alshammari, A. (2021). "A Study on Inventory Management Techniques in Pharmacy Management Systems." *International Journal of Scientific & Engineering Research*, 12(5), 10-15.

4. **Software and Libraries**
    - MySQL: Open-source relational database management system. Oracle Corporation.
    - Streamlit: Python library for creating web applications.
    - Python: Programming language used for back-end development.

5. **Online Tutorials**
    - DataCamp. (n.d.). *SQL Tutorial: Learn SQL Basics for Data Science*. Retrieved from https://www.datacamp.com/
    - Real Python. (n.d.). *How to Build a CRUD Application with Streamlit*. Retrieved from https://realpython.com/

APPENDIX A: DEFINITIONS, ACRONYMS, AND ABBREVIATIONS

This appendix provides definitions for technical terms, acronyms, and abbreviations used in the report to ensure clarity for readers.

A.1 Definitions

- **Medicine Supply Management System**: A system designed to manage and streamline the inventory, purchasing, and sale of medicines and related products. It includes functionalities for tracking products, suppliers, customers, and purchase orders.
- **CRUD Operations**: Refers to the four basic operations for managing data in a database: **Create**, **Read**, **Update**, and **Delete**. These operations are fundamental to database management systems and form the core functionality of the user interface.
- **Medicine**: A pharmaceutical product used for medical treatment. In this project, medicine refers to the various drugs managed by the system, including information such as name, type, price, and quantity.
- **Supplier**: An entity that provides medicines and products to the system. Suppliers are responsible for supplying medicines to pharmacies, healthcare institutions, or customers.
- **Product**: In this context, a product refers to any medicine or item that is part of the system's inventory. A product is linked to a specific medicine and supplier.
- **Customer**: An individual or organization that purchases medicines or products from the system. Customers can place orders for products, and their details are stored in the system.
- **Purchase Order**: A record in the system that tracks customer orders, including details like the order date, status, and total amount.
- **Trigger**: A set of instructions in a database that automatically executes in response to certain events, such as inserting a new record or updating a field. In this system, triggers are used to automatically update stock levels or perform other actions when an order is placed.
- **Stored Procedure**: A precompiled collection of one or more SQL statements stored in the database. Stored procedures can be executed as a unit to perform operations like adding data or calculating totals.
- **Function**: A stored program in SQL that returns a value. Functions are used in the system to calculate data, such as the total value of stock or expired stock.

A.2 Acronyms

- **CRUD**: Create, Read, Update, Delete
- **SQL**: Structured Query Language
- **DBMS**: Database Management System
- **UI**: User Interface
- **API**: Application Programming Interface
- **HTTP**: HyperText Transfer Protocol
- **HTTPS**: HyperText Transfer Protocol Secure

A.3 Abbreviations

- **DB**: Database
- **ID**: Identifier
- **Qty**: Quantity
- **Stock Qty**: Stock Quantity
- **Price**: Price
- **Exp**: Expiry (Expiry Date of the product)
- **Qty Sold**: Quantity Sold
- **Total Amt**: Total Amount

A.4 System-Specific Terms

- **Streamlit**: A Python library used for building web applications. Streamlit is used in this project to create the front-end user interface for managing CRUD operations.
- **MySQL**: An open-source relational database management system (RDBMS) that is used to store and manage the database for the Medicine Supply Management System.
- **Python**: A high-level programming language used for back-end development in this project. It interacts with the MySQL database to perform CRUD operations.