# Medicine Management System Project Report

**Table of Contents :**

# ***MEDICINE MANAGEMENT SYSTEM PROJECT***

## 1. Introduction :

The introduction will provide an overview of what a Medicine Management System is, why it's important, and how DBMS can be utilized to optimize the management of medicines in hospitals or pharmacies.

### What is Medicine Management?

Medicine management involves maintaining accurate records of medicines, including their names, quantities, dosages, suppliers, and expiry dates. The goal is to ensure that medicines are efficiently tracked and used in healthcare systems.

### Importance of Medicine Management

- Ensures medicines are readily available when needed.
- Prevents overstocking and understocking of medicines.
- Helps in tracking expiration dates and batch numbers to avoid the use of expired or faulty medicines.
- Enhances the operational efficiency of pharmacies, hospitals, and clinics.

### Purpose of the DBMS

A Database Management System (DBMS) is used to efficiently store, retrieve, and manage data in structured formats. In the context of medicine management, it allows for real-time tracking and monitoring of inventory, minimizing human error and optimizing resources.

## . Objectives :

The primary objectives of this project are:

- To design and implement a Medicine Management System using a DBMS.
- To demonstrate the effectiveness of relational databases in managing large sets of medical data.
- To create an efficient database structure for storing and retrieving medicine-related information.
- To explore SQL queries, including CRUD operations (Create, Read, Update, Delete), joins, nested queries, and transactions .
- Efficient Inventory Management: Automate the tracking of medicine stocks, expiry dates, and reorder levels.
- Streamlined Sales & Prescription Handling: Facilitate quick order processing and maintain accurate records of prescriptions.
- Supplier & Customer Management: Maintain detailed records of suppliers and customers to enhance relationships and improve service quality.
- Data Integrity and Reporting: Ensure reliable data storage with consistent relationships among tables for effective reporting and analysis.
- Enhanced Query Capability: Allow complex SQL queries (including joins and nested queries) for advanced data retrieval and decision support.

## 3. System design :

A robust system design is key to ensuring that the database supports all necessary operations and maintains data integrity. In this project, we start with an ER diagram that maps out the core entities and their relationships, then translate that into a relational schema.

### a. ER Diagram :

The Entity-Relationship (ER) diagram models the key entities and relationships in the system. Here is a textual description:

- **Medicine :**

- **Attributes:** Medicine_ID (PK), Name, Type, Manufacturer, Expiry_Date, Price, Stock_Quantity
- **Description**: Contains details about each medicine.

- **Supplier :**
  - **Attributes**: Supplier_ID (PK), Supplier_Name, Contact_Info, Address
  - **Description**: Stores information about the suppliers of medicines.

- **Customer :**
  - **Attributes:** Customer_ID (PK), Customer_Name, Contact_Info, Address
  - **Description**: Maintains customer profiles who purchase medicines or require prescriptions.

- **Order** :
  - **Attributes:** Order_ID (PK), Order_Date, Customer_ID (FK), Total_Amount
  - **Description:** Records individual sales transactions.

- **Order_Details :**
  - **Attributes:** Order_ID (FK), Medicine_ID (FK), Quantity, Price
  - **Description**: Contains details of medicines included in each order. The combination of Order_ID and Medicine_ID serves as a composite primary key.

- **Prescription :**

- **Attributes**: Prescription_ID (PK), Customer_ID (FK), Doctor_Name, Date, Diagnosis

- **Description**: Records prescriptions provided to customers by healthcare professionals.

- **Medicine_Prescription :**

- **Attributes**: Prescription_ID (FK), Medicine_ID (FK), Dosage, Duration

- **Description**: Maps medicines to a given prescription with dosage and duration details.

**Relationships:**

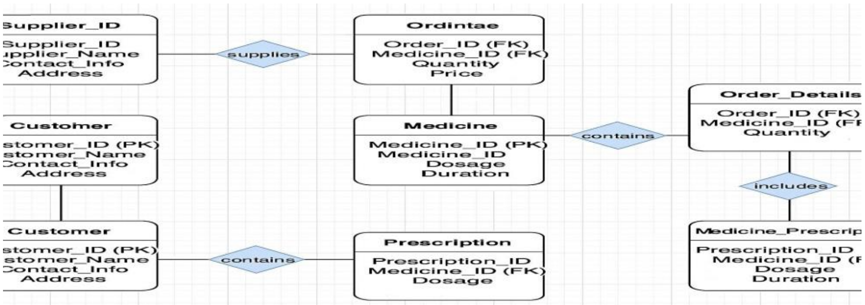
- A Supplier can supply multiple Medicines.

- A Medicine is supplied by one Supplier

- A Customer can place multiple Orders and have multiple Prescriptions.

- An Order may include one or more Medicines (via Order_Details).

- A Prescription may include multiple Medicines (via Medicine_Prescription).



**Relational Diagram :**

**1. Supplier (Supplier_ID, Supplier_Name, Contact_Info, Address)**

**2. Medicine (Medicine_ID, Name, Type, Manufacturer, Expiry_Date, Price, Stock_Quantity, Supplier_ID)**

**3. Customer (Customer_ID, Customer_Name, Contact_Info, Address)**

**4. Orders (Order_ID, Order_Date, Customer_ID, Total_Amount)**

**5. Order_Details (Order_ID, Medicine_ID, Quantity, Price)**

**6. Prescription (Prescription_ID, Customer_ID, Doctor_Name, Date, Diagnosis)**

**7. Medicine_Prescription (Prescription_ID, Medicine_ID, Dosage, Duration)**

Each table enforces foreign key constraints to maintain referential integrity across the database

## 4. Database Implementation :

This section includes the SQL code for creating the tables (DDL) and inserting sample data (DML).

### *DDL – Table Creation*

Below is the complete SQL script for creating the database tables.

SQL :

```
CREATE TABLE Medicines (

    MedicineID INT PRIMARY KEY,
    Name VARCHAR(255),

    Type VARCHAR(100),

    Dosage VARCHAR(100),

    Quantity INT,

    ExpiryDate DATE

);

CREATE TABLE Suppliers (

    SupplierID INT PRIMARY KEY,

    Name VARCHAR(255),

    ContactInfo VARCHAR(255)

);

CREATE TABLE Inventory (

    InventoryID INT PRIMARY KEY,

    MedicineID INT,

    Quantity INT,

    DateReceived DATE,

    FOREIGN KEY (MedicineID) REFERENCES
Medicines(MedicineID)

);
```

```
CREATE TABLE Sales (

    SaleID INT PRIMARY KEY,

    MedicineID INT,

    QuantitySold INT,

    SaleDate DATE,

    FOREIGN KEY (MedicineID) REFERENCES
Medicines(MedicineID)

);

CREATE TABLE Purchases (

    PurchaseID INT PRIMARY KEY,

    SupplierID INT,

    MedicineID INT,

    PurchaseDate DATE,

    Quantity INT,

    FOREIGN KEY (SupplierID) REFERENCES Suppliers(SupplierID),

    FOREIGN KEY (MedicineID) REFERENCES
Medicines(MedicineID)

);
```

**Sample Data Insertion :**

Insert sample data into the tables using DML commands:

```
sql
Copy
INSERT INTO Medicines (MedicineID, Name, Type, Dosage,
Quantity, ExpiryDate)
VALUES (1, 'Paracetamol', 'Tablet', '500mg', 100, '2026-
12-31');

INSERT INTO Suppliers (SupplierID, Name, ContactInfo)
VALUES (1, 'ABC Pharmaceuticals', '123-456-7890');

INSERT INTO Inventory (InventoryID, MedicineID, Quantity,
DateReceived)
VALUES (1, 1, 100, '2025-04-01');
```

## 5. SQL OPERATIONS :

### CRUD Operations (Insert, Update, Delete)

- **Insert**:

  ```
  sql
  Copy
  INSERT INTO Medicines (MedicineID, Name, Type,
  Dosage, Quantity, ExpiryDate)
  VALUES (2, 'Ibuprofen', 'Tablet', '200mg', 150,
  '2025-12-31');
  ```

- **Update**:

  ```
  sql
  Copy
  UPDATE Medicines
  SET Quantity = 120
  WHERE MedicineID = 1;
  ```

- **Delete**:

  ```
  sql
  Copy
  DELETE FROM Medicines
  ```

```
WHERE MedicineID = 2;
```

## Join Operations

- **Inner Join** (Medicine sales information with medicine details):

```
sql
Copy
SELECT Medicines.Name, Sales.QuantitySold,
Sales.SaleDate
FROM Sales
INNER JOIN Medicines ON Sales.MedicineID =
Medicines.MedicineID;
```

- **Left Join** (Medicine purchases and their suppliers):

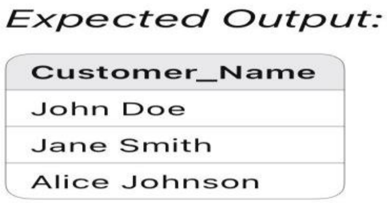
```
sql
Copy
SELECT Medicines.Name, Purchases.Quantity,
Suppliers.Name
FROM Purchases
LEFT JOIN Medicines ON Purchases.MedicineID =
Medicines.MedicineID
LEFT JOIN Suppliers ON Purchases.SupplierID =
Suppliers.SupplierID;
```

## Nested Queries

- Example of a nested query to find medicines that have been sold more than 100 times:

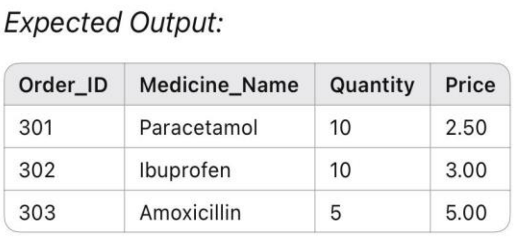
```
sql
Copy
SELECT Name
FROM Medicines
WHERE MedicineID IN (
    SELECT MedicineID
    FROM Sales
    GROUP BY MedicineID
    HAVING SUM(QuantitySold) > 100
);
```

## 6. OUTPUT :



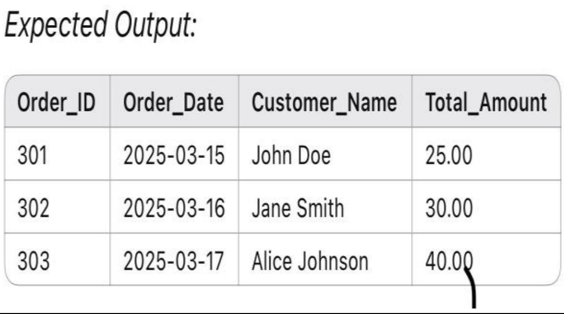
*Expected Output:*

| Customer_Name |
|---|
| John Doe |
| Jane Smith |
| Alice Johnson |



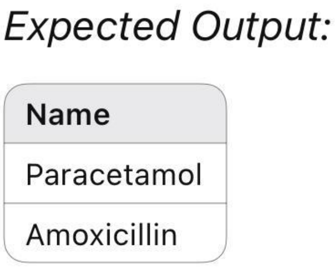
*Expected Output:*

| Order_ID | Medicine_Name | Quantity | Price |
|---|---|---|---|
| 301 | Paracetamol | 10 | 2.50 |
| 302 | Ibuprofen | 10 | 3.00 |
| 303 | Amoxicillin | 5 | 5.00 |



*Expected Output:*

| Order_ID | Order_Date | Customer_Name | Total_Amount |
|---|---|---|---|
| 301 | 2025-03-15 | John Doe | 25.00 |
| 302 | 2025-03-16 | Jane Smith | 30.00 |
| 303 | 2025-03-17 | Alice Johnson | 40.00 |



*Expected Output:*

| Name |
|---|
| Paracetamol |
| Amoxicillin |

## 7. Learning Outcomes :

- **Understanding the Design Process**: I have learned how to design a database schema based on real-world needs and how to create an efficient structure for data storage.
- **Mastery of SQL**: The project has helped in understanding various SQL operations like JOINs, Nested Queries, and CRUD operations, which are essential for manipulating data.
- **Real-World Application**: I now understand how to apply DBMS concepts to manage an essential service like medicine inventory, which can improve efficiency in healthcare.

## 8. Conclusion :

The Medicine Management System project exemplifies the practical application of DBMS concepts in a healthcare context. The project's comprehensive design—from conceptual modeling (ER diagram) to a fully implemented relational schema—ensures that critical data such as medicine inventories, supplier information, customer orders, and prescriptions are stored and managed effectively.

Key conclusions include:

- **Enhanced Efficiency:**

Automating the tracking and management of medicines reduces manual errors and streamlines operations.

- **Data Integrity:**

Well-defined relationships and constraints maintain data consistency and support accurate reporting.

- **Advanced Query Capabilities:**

The system supports complex SQL operations (joins and nested queries) that facilitate robust data retrieval, essential for decision-making and strategic planning.

- **Scalability:**

The design can be extended to incorporate additional functionalities such as detailed billing, advanced analytics, and integration with electronic health records (EHR).

Overall, this project serves as a solid foundation for a comprehensive medicine management solution. It not only meets the basic requirements of a DBMS project but also provides insights into how database technologies can be leveraged to improve operational efficiency in the healthcare sector.