**Pharmacy Management System**



**Project Report**

Database Management System Lab

Division 4, G8

**Submitted By -**

Pearl Kalariya(23BCP248)

Maan Patel(23BCP253)

Department of Computer Science Engineering

School of Technology

Pandit Deendayal Energy University (PDEU)

Gandhinagar, INDIA-382426

Project Report

# 1. Introduction

The Pharmacy Management System (PMS) is a relational database solution designed to streamline and optimize pharmacy operations. It manages key tasks such as inventory control, sales tracking, prescription handling, and stakeholder management, including customers, doctors, suppliers, and pharmacists. Built using MySQL, the PMS ensures data accuracy, consistency, and efficient access for users.

# 2. Requirements Analysis

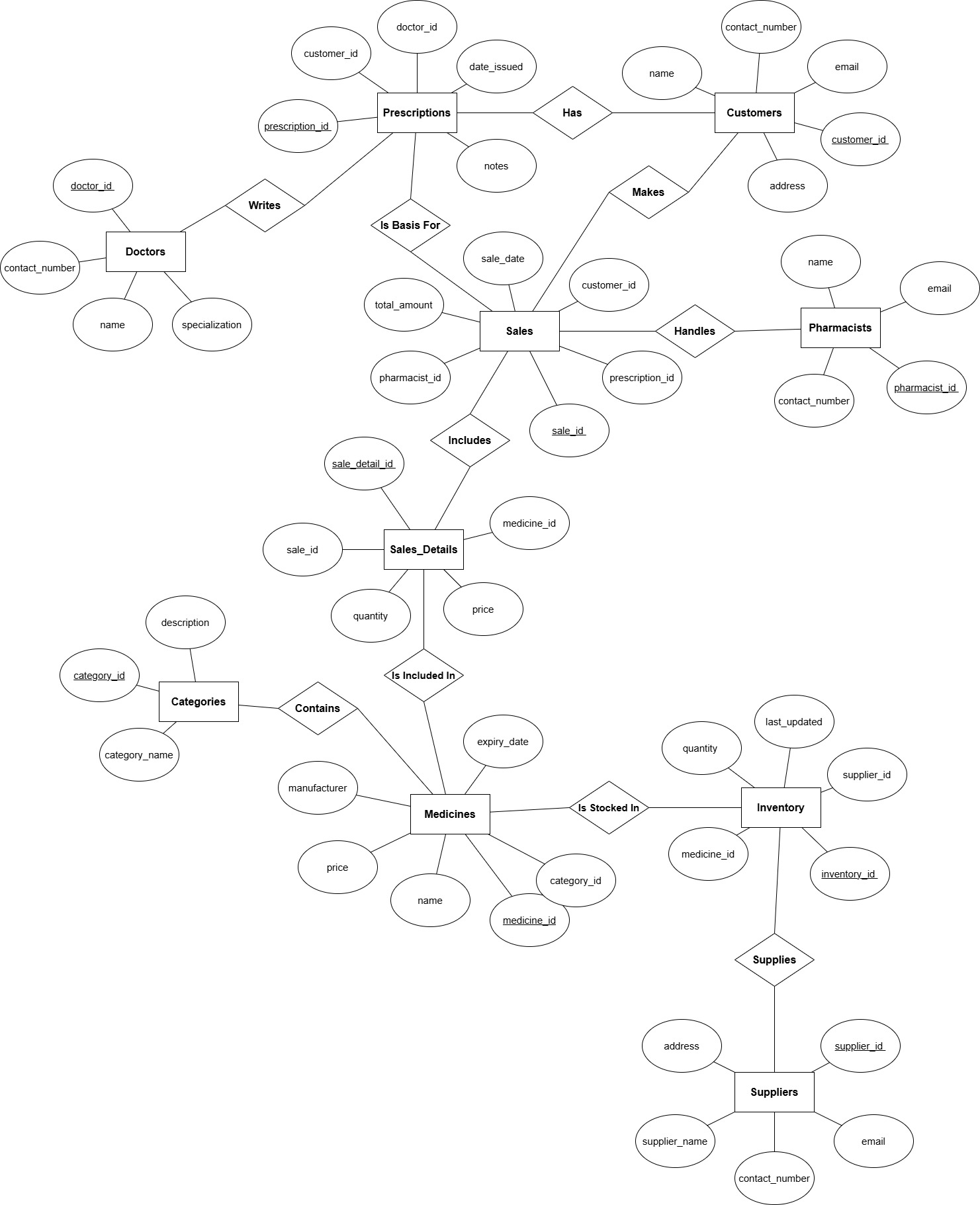
## Functional Requirements

- Manage medicine data including name, category, price, manufacturer, and expiry  
- Maintain records of suppliers and manage medicine inventory  
- Store and update customer and doctor information  
- Record prescriptions issued by doctors to customers  
- Track sales transactions with itemized details  
- Enforce relational integrity with primary and foreign keys

## Typical User Queries

- Identify expired or low-stock medicines  
- View total sales per customer or pharmacist  
- Determine top-selling medicines  
- Review prescription history of customers  
- List medicines supplied by specific suppliers

# 3. ER Diagram



Entities in the ER Diagram:  
- Categories  
- Medicines  
- Suppliers  
- Inventory  
- Customers  
- Doctors  
- Prescriptions  
- Pharmacists  
- Sales  
- Sales\_Details

# 4. Relational Schema

* **Categories**(category\_id INT, category\_name VARCHAR(100), description TEXT)
* **Medicines**(medicine\_id INT, name VARCHAR(100), category\_id INT, manufacturer VARCHAR(100), price DECIMAL(10,2), expiry\_date DATE)
* **Suppliers**(supplier\_id INT, supplier\_name VARCHAR(100), contact\_number VARCHAR(15), email VARCHAR(100), address TEXT)
* **Inventory**(inventory\_id INT, medicine\_id INT, supplier\_id INT, quantity INT, last\_updated DATE)
* **Customers**(customer\_id INT, name VARCHAR(100), contact\_number VARCHAR(15), email VARCHAR(100), address TEXT)
* **Doctors**(doctor\_id INT, name VARCHAR(100), specialization VARCHAR(100), contact\_number VARCHAR(15))
* **Prescriptions**(prescription\_id INT, customer\_id INT, doctor\_id INT, date\_issued DATE, notes TEXT)
* **Pharmacists**(pharmacist\_id INT, name VARCHAR(100), contact\_number VARCHAR(15), email VARCHAR(100))
* **Sales**(sale\_id INT, customer\_id INT, pharmacist\_id INT, prescription\_id INT, sale\_date DATE, total\_amount DECIMAL(10,2))
* **Sales\_Details**(sale\_detail\_id INT, sale\_id INT, medicine\_id INT, quantity INT, price DECIMAL(10,2))

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Table Name** | **Purpose** | **Primary Key (PK)** | **Foreign Keys (FK)** | **Normal Keys (NK)** |
| **Categories** | Stores different categories of medicines. | category\_id | – | category\_name, description |
| **Medicines** | Contains details of all medicines available. | medicine\_id | category\_id → Categories(category\_id) | name, manufacturer, price, expiry\_date |
| **Suppliers** | Holds information about medicine suppliers. | supplier\_id | – | supplier\_name, contact\_number, email, address |
| **Inventory** | Tracks stock quantity of medicines from suppliers. | inventory\_id | medicine\_id → Medicines(medicine\_id) supplier\_id → Suppliers(supplier\_id) | quantity, last\_updated |
| **Customers** | Maintains records of pharmacy customers. | customer\_id | – | name, contact\_number, email, address |
| **Doctors** | Stores doctor information who issue prescriptions. | doctor\_id | – | name, specialization, contact\_number |
| **Prescriptions** | Contains prescriptions issued by doctors to customers. | prescription\_id | customer\_id → Customers(customer\_id) doctor\_id → Doctors(doctor\_id) | date\_issued, notes |
| **Pharmacists** | Lists pharmacists who process and record sales. | pharmacist\_id | – | name, contact\_number, email |
| **Sales** | Records overall sales transactions. | sale\_id | customer\_id → Customers(customer\_id) pharmacist\_id → Pharmacists(pharmacist\_id) prescription\_id → Prescriptions(prescription\_id) | sale\_date, total\_amount |
| **Sales\_Details** | Contains line-item details of medicines sold in each sale transaction. | sale\_detail\_id | sale\_id → Sales(sale\_id) medicine\_id → Medicines(medicine\_id) | quantity, price |

All tables use primary keys for unique identification and foreign keys to enforce relationships. Constraints such as NOT NULL, UNIQUE, and CHECK help maintain data validity.

# 6. Normalization

1. **First Normal Form (1NF) – *Atomic values & unique rows***

* All columns must contain only atomic (indivisible) values. Each record must be unique.

1. **Second Normal Form (2NF) – *1NF + full functional dependency***

* Every non-prime attribute must depend on the whole primary key (for composite keys).

1. **Third Normal Form (3NF) – *2NF + no transitive dependency***

* Non-prime attributes should not depend on other non-prime attributes.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Table** | **1NF** | **2NF** | **3NF** | **Notes** |
| Categories | ✔️ | ✔️ | ✔️ | Basic reference table |
| Medicines | ✔️ | ✔️ | ✔️ | Good FK usage |
| Suppliers | ✔️ | ✔️ | ✔️ | No redundancy |
| Inventory | ✔️ | ✔️ | ✔️ | Junction table with quantities |
| Customers | ✔️ | ✔️ | ✔️ | Well structured |
| Doctors | ✔️ | ✔️ | ✔️ | Independent entity |
| Prescriptions | ✔️ | ✔️ | ✔️ | Clean relationships |
| Pharmacists | ✔️ | ✔️ | ✔️ | Clean structure |
| Sales | ✔️ | ✔️ | ✔️ | Well-normalized transactional data |
| Sales\_Details | ✔️ | ✔️ | ✔️ | Correct many-to-many normalization |

# 7. Query Examples

**1.Medicines with Category Names**

SELECT m.name AS medicine, c.category\_name

FROM Medicines m

JOIN Categories c ON m.category\_id = c.category\_id;

**2.Total Sales per Customer**

SELECT cu.name, SUM(s.total\_amount) AS total\_spent

FROM Sales s

JOIN Customers cu ON s.customer\_id = cu.customer\_id

GROUP BY s.customer\_id;

**3.Expired Medicines**

SELECT name, expiry\_date

FROM Medicines

WHERE expiry\_date < CURDATE();

**4.Low Stock Inventory**

SELECT m.name, i.quantity

FROM Inventory i

JOIN Medicines m ON i.medicine\_id = m.medicine\_id

WHERE i.quantity < 10;

**5.Doctors and Number of Prescriptions Issued**

SELECT d.name, COUNT(p.prescription\_id) AS prescriptions\_issued

FROM Doctors d

JOIN Prescriptions p ON d.doctor\_id = p.doctor\_id

GROUP BY d.doctor\_id;

**6.Top 5 Most Sold Medicines**

SELECT m.name, SUM(sd.quantity) AS total\_sold

FROM Sales\_Details sd

JOIN Medicines m ON sd.medicine\_id = m.medicine\_id

GROUP BY sd.medicine\_id

ORDER BY total\_sold DESC

LIMIT 5;

**7.Detailed Sale View**

SELECT s.sale\_id, s.sale\_date, m.name, sd.quantity, sd.price

FROM Sales s

JOIN Sales\_Details sd ON s.sale\_id = sd.sale\_id

JOIN Medicines m ON sd.medicine\_id = m.medicine\_id

WHERE s.sale\_id = 1;

**8.Customer Prescription History**

SELECT p.prescription\_id, p.date\_issued, d.name AS doctor

FROM Prescriptions p

JOIN Doctors d ON p.doctor\_id = d.doctor\_id

WHERE p.customer\_id = 1;

**9.Total Inventory by Medicine**

SELECT m.name, SUM(i.quantity) AS total\_stock

FROM Inventory i

JOIN Medicines m ON i.medicine\_id = m.medicine\_id

GROUP BY m.medicine\_id;

**10.Sales by Pharmacist**

SELECT ph.name, COUNT(s.sale\_id) AS total\_sales

FROM Pharmacists ph

JOIN Sales s ON ph.pharmacist\_id = s.pharmacist\_id

GROUP BY ph.pharmacist\_id;

**11.Supplier-wise Medicines**

SELECT sup.supplier\_name, m.name AS medicine

FROM Inventory i

JOIN Suppliers sup ON i.supplier\_id = sup.supplier\_id

JOIN Medicines m ON i.medicine\_id = m.medicine\_id;

**12.High-Price Medicines (₹500+)**

SELECT name, price

FROM Medicines

WHERE price > 500;

**13.Current Month’s Sales**

SELECT \* FROM Sales

WHERE MONTH(sale\_date) = MONTH(CURDATE()) AND YEAR(sale\_date) = YEAR(CURDATE());

**14.Average Sale per Pharmacist**

SELECT ph.name, AVG(s.total\_amount) AS avg\_sale

FROM Pharmacists ph

JOIN Sales s ON ph.pharmacist\_id = s.pharmacist\_id

GROUP BY ph.pharmacist\_id;

**15.Customers Served by Pharmacist**

SELECT ph.name, COUNT(DISTINCT s.customer\_id) AS customers\_served

FROM Pharmacists ph

JOIN Sales s ON ph.pharmacist\_id = s.pharmacist\_id

GROUP BY ph.pharmacist\_id;

# 8. Testing and Evaluation

* **Data Integrity:** Validated referential integrity and cascading relationships.
* **Performance:** Indexed foreign keys and search-heavy columns.
* **Realism:** Inserted realistic sample data exceeding 30 entries per table.
* **Edge Cases:** Tested NULL values, constraint violations, and zero-record conditions.

# 9. Conclusion

The Pharmacy Management System database is a comprehensive, normalized, and robust solution for pharmacy operations. It effectively demonstrates the use of ER modeling, relational schema design, SQL scripting, and query optimization. With scope for front-end integration and advanced analytics, this project lays the groundwork for a full-scale pharmacy management application.