PHARMACY MANAGEMENT SYSTEM

DBMS PROJECT REPORT

Submitted by

PRIYANSHU DARSHAN [RA231003011372]
AASTHA SINGH [RA2311003011373]

Under the Guidance of

(DR. RAMA P)

Assistant Professor

DEPARTMENT OF COMPUTING TECHNOLOGIES

in partial fulfillment of the requirements for the degree of

BACHELOR OF TECHNOLOGY
in
COMPUTER SCIENCE ENGINEERING

DEPARTMENT OF COMPUTING
TECHNOLOGIES, COLLEGE OF ENGINEERING
AND TECHNOLOGY
SRM INSTITUTE OF SCIENCE ANDTECHNOLOGY

KATTANKULATHUR- 603 203

MAY 2025

Department of Computational Intelligence
SRM Institute of Science & Technology
Own Work Declaration Form

This sheet must be filled in (each box ticked to show that the condition has been met). It must besigned and dated along with your student registration number and included with all assignments you submit – work will not be marked unless this is done.

To be completed by the student for all assessments

Degree/ Course : **BTech/ 21CSC205P (Database Management Systems)**

Student Name : **PRIYANSHU DARSHAN & AASTHA SINGH**

Registration Number : **RA2311003011372 & RA2311003011373**

Title of Work : **Pharmacy Management System**

I / We hereby certify that this assessment compiles with the University's Rules and Regulations relating to Academic misconduct and plagiarism, as listed in the University Website, Regulations, and the Education Committee guidelines.

I / We confirm that all the work contained in this assessment is my / our own except where indicated, and that I / We have met the following conditions:

- Clearly referenced / listed all sources as appropriate
- Referenced and put in inverted commas all quoted text (from books, web, etc)
- Given the sources of all pictures, data etc. that are not my own
- Not made any use of the report(s) or essay(s) of any other student(s) either past or present
- Acknowledged in appropriate places any help that I have received from others (e.g.fellow students, technicians, statisticians, external sources)
- Compiled with any other plagiarism criteria specified in the Course handbook /University website

I understand that any false claim for this work will be penalized in accordance with the University policies and regulations.

DECLARATION:
I am aware of and understand the University's policy on Academic misconduct and plagiarism and I certify that this assessment is my / our own work, except where indicated by referring, and that I have followed the good academic practices noted above.
If you are working in a group, please write your registration numbers and sign with the date forevery student in your group.

SRM INSTITUTE OF SCIENCE AND TECHNOLOGY
KATTANKULATHUR – 603 203

BONAFIDE CERTIFICATE

Certified that 21CSC205P – Database Management System Project report titled "**PHARMACY MANAGEMENT SYSTEM**" is the bonafide work of "**PRIYANSHU DARSHAN [RA2311003011372], AASTHA SINGH [RA2311003011373]** who carried out the project work under my supervision. Certified further, that to the best of my knowledge the work reported herein does not form any other project report or dissertation on the basis of which a degree or award was conferred on an earlier occasion on this or any other candidate.

SIGNATURE

Dr. Rama P.

ASSISSTANT PROFESSOR
DEPARTMENT OF
COMPUTING TECHNOLOGIES

SIGNATURE

Dr. G. NIRANJANA

PROFESSOR &HEAD
DEPARTMENT OF
COMPUTING TECHNOLOGIES

ACKNOWLEDGEMENTS

We express our humble gratitude to **Dr. C. Muthamizhchelvan**, Vice-Chancellor, SRM Institute of Science and Technology, for the facilities extended for the project work and his continued support.

We extend our sincere thanks to **Dr. Leenus Jesu Martin M,** Dean-CET, SRM Institute of Science and Technology, for his invaluable support.

We wish to thank **Dr. Revathi Venkataraman**, Professor and Chairperson, School of Computing, SRM Institute of Science and Technology, for her support throughout the project work.

We encompass our sincere thanks to, **Dr. M. Pushpalatha**, Professor and Associate Chairperson, School of Computing and **Dr. C. Lakshmi,** Professor and Associate Chairperson, School of Computing, SRM Institute of Science and Technology, for their invaluable support.

We are incredibly grateful to our Head of the Department, **Dr. G. Niranjana**, Professor, Department of Computing Technologies, SRM Institute of Science and Technology, for her suggestions and encouragement at all the stages of the project work.

We want to convey our thanks to our DBMS Coordinators, **Dr. Muthu Kumaran A M J** and **Dr. Jayapradha J** Department of Computing Technologies, and our Audit Professor **Dr. Malathy C,** Department of Networking And Communications, SRM Institute of Science and Technology, for their inputs during the project reviews and support.

We register our immeasurable thanks to our Faculty Advisors, **Dr. Shiju Kumar P S**, Department of Computing Technologies, SRM Institute of Science and Technology, for leading and helping us to complete our course.

Our inexpressible respect and thanks to our faculty, **Dr. Rama P,** Department of Computing Technologies, S.R.M Institute of Science and Technology, for providing us with an opportunity to pursue our project under her mentorship. She provided us with the freedom and support to explore the research topics of our interest. Her passion for solving problems and making a difference in the world has always been inspiring.

We sincerely thank all the staff and students of Computing Technologies, School of Computing, S.R.M Institute of Science and Technology, for their help during our project. Finally, we would like to thank our parents, family members, and friends for their unconditional love, constant support and encouragement

PRIYANSHU DARSHAN[RA2311003011372]

AASTHA SINGH[RA2311003011373]

ABSTRACT

The **Pharmacy Management System (PMS)** is a comprehensive, centralized application developed to streamline and automate the core operations of modern pharmaceutical outlets. As pharmacies evolve to meet the growing healthcare demands, traditional methods of record-keeping and inventory tracking become increasingly inefficient and prone to error. PMS addresses these limitations by digitizing critical processes such as medicine inventory management, supplier coordination, sales and billing, prescription tracking, and customer information handling.

Built using a **Relational Database Management System (RDBMS)** and implemented through **Structured Query Language (SQL)**, the system ensures robust data handling capabilities. Its design is rooted in an **Entity-Relationship (ER) model**, accurately capturing the interrelations among key entities like Medicines, Customers, Suppliers, Pharmacists, Prescriptions, and Sales Transactions. These are translated into relational schemas and normalized up to **Third Normal Form (3NF)** to maintain data consistency, reduce redundancy, and support efficient data operations.

Security and controlled access are integral to the PMS architecture. The system incorporates **role-based access controls**, allowing different levels of access for pharmacists, administrators, and other personnel, ensuring that sensitive data such as prescription history and customer details are accessed only by authorized users. Additionally, authentication and authorization protocols safeguard against unauthorized data access.

The system offers real-time inventory updates, rapid query processing, and dynamic report generation, assisting pharmacy managers in making informed decisions. From stock alerts to sales summaries, the PMS supports **data-driven workflows** that enhance operational accuracy and transparency. The intuitive dashboard and user-friendly interface make the system accessible to users regardless of their technical proficiency.

A notable strength of the PMS is its **scalability and adaptability**. It is built to accommodate increasing transactional volumes and can be extended to support advanced features such as **cloud-based access, mobile interfaces, AI-powered analytics, and automated expiry alerts**. These enhancements not only future-proof the system but also improve the efficiency and responsiveness of pharmacy operations.

This project report documents the complete development lifecycle of the Pharmacy Management System, encompassing initial requirement analysis, database design, system implementation, and testing. It also addresses challenges such as **concurrent data access**, **performance tuning**, and **user training**. Evaluation metrics affirm the system's ability to minimize manual workload, reduce stock mismanagement, and significantly improve the reliability of pharmacy services.

In conclusion, the Pharmacy Management System stands as a robust solution for modernizing pharmaceutical management. It empowers pharmacies to transition from outdated processes to a streamlined, digital environment, thereby enhancing service quality, operational efficiency, and readiness for future technological integration.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

ABBREVIATIONS

Abbreviations	Full Form
PMS	**Pharmacy Management System**
UI	**User Interface**
UX	**User Experience**
DB	**Database**
SQL	**Structured Query Language**
API	**Application Programming Interface**
JWT	**JSON Web Token**
CRUD	**Create, Read, Update, Delete**
DBMS	**Database Management System**
HTML	**Hypertext Markup Language**
BMI	**Body Mass Index**
PHP	**Hypertext Preprocessor**
RDBMS	**Relational Database Management System**
MERN	**MongoDB, Express.js, React.js, Node.js**
JDBC	**Java Database Connectivity**
IDE	**Integrated Development Environment**
REST	**Representational State Transfer**
JSON	**JavaScript Object Notation**
HTTPS	**HyperText Transfer Protocol Secure**
MVC	**Model-View-Controller**
DBMS	**Database Management System**
PWA	**Progressive Web App**
KPI	**Key Performance Indicator**
ERP	**Enterprise Resource Planning**
QA	**Quality Assurance**
ROI	**Return on Investment**
UI/UX	**User Interface/User Experience**

CHAPTER 1: INTRODUCTION

1.1 General

The Pharmacy Management System is a MySQL-backed application designed to automate and centralize pharmacy operations. It handles medicines, customers, suppliers, billing, and inventory through a structured database and Java-based frontend.

1.2 Motivation

Manual systems often lead to data loss, redundancy, and delays. This system replaces outdated methods with a reliable, automated solution to improve efficiency, accuracy, and speed.

1.3 Key Requirements

- **Login System:** Basic role-based authentication
- **Inventory Control:** Real-time tracking of stock and expiry
- **Billing Module:** Automated receipt generation and transaction logs
- **Customer/Supplier Management:** Record and manage details
- **Search & Reports:** Quick lookup and report generation

1.4 Challenges

Managing foreign key relations, ensuring validation, and maintaining consistency across transactions were primary challenges, along with building a clean, responsive GUI.

1.5 Design Principles

- Modular and normalized schema (up to 3NF)
- Secure access with input validation
- Maintainable structure with logical table grouping
- Extendable for future modules (e.g., alerts, online access)

1.6 Technologies Used

- **Frontend:** PHP
- **Backend:** MySQL
- **Tools:** phpMyAdmin, VS Code

1.7 Conclusion

This system provides a streamlined approach to pharmacy management, replacing manual tasks with efficient digital processes. Further chapters cover schema, implementation, testing, and proposed enhancements.

CHAPTER 2: 1B EXPERIMENT

1B: Problem understanding, Identification of Entity and Relationships, Construction of DB using ER Model for their project

2.1 Problem Understanding

- Managing a pharmacy's daily operations involves handling a wide range of interconnected entities such as medicines, suppliers, customers, sales transactions, and stock levels. Each component is tightly linked—for instance, medicines are supplied by vendors, sold to customers, tracked for expiry, and must be restocked based on demand. In a manual or loosely organized system, maintaining accuracy, preventing stockouts, avoiding expired drug sales, and ensuring billing transparency becomes difficult and error-prone.

Identification of Entities and Relationships

2.2: Entities and Attributes:

1. **Sales**

   Tracks overall transactions made in the pharmacy, including date, total amount, and links to both employee and customer involved.

2. **Medicines**

   Core inventory of the pharmacy; stores info like name, price, quantity, expiry, category (tablet, syrup, etc.), and location rack.

3. **Employee**

   Represents the staff working in the pharmacy, storing details like ID, contact number, email, DOB, salary, and job type.

4. **Customer**

   Contains info about the buyers—name, contact details, age, and email—for reference in billing and sales tracking.

5. **Sales_Items**

   Junction entity between **Sales** and **Medicines**, showing which medicine was sold in what quantity and at what price.

6. **Admin**

   Users with system-level access; stores credentials and is linked to which supplier they manage.

7. **Supplier**

   External providers who supply medicines to the pharmacy. Tracks name, ID, address, and contact number.

2.3 Relationships:

1. **Employee → Sales** (One-to-Many)

   Each employee may perform multiple sales; each sale is done by one employee.

   (Sales.emp_id → Employee.e_id)

2. **Customer → Sales** (One-to-Many)

   Each customer may make multiple purchases; each sale is linked to one customer.

   (Sales.c_id → Customer.c_id)

3. **Sales → Sales_Items** (One-to-Many)

   Each sale can include multiple medicine items; each Sales_Item entry belongs to one sale.

   (Sales_Items.sale_id → Sales.sales_id)

4. **Medicines → Sales_Items** (One-to-Many)

   Each medicine can appear in multiple sales; each Sales_Item references one medicine.

   (Sales_Items.med_id → Medicines.med_id)

5. **Admin → Supplier** (One-to-Many)

   Each Admin manages multiple suppliers; each supplier entry is linked to one admin.

   (Supplier.admin_id → Admin.admin_id)

6. **Supplier → Medicines** (One-to-Many, via 'Supply')

   Each supplier can supply many medicines; each medicine comes from one supplier.

   (Inferred via 'Supply' relationship)

7. **Customer → Purchase** (One-to-Many)

   Each customer can make multiple purchases; each purchase relates to one customer.

   (Sales relationship implies this via c_id)

8. **Employee → Done By (Sales)** (One-to-Many)

   Each sale is done by one employee; employees can have multiple sales entries.

   (done by: Sales.emp_id → Employee.e_id)

9. **Employee → Gross Salary** (Derived/Composite)

   Gross salary is derived from basic pay; part of employee's salary structure.

   (Gross_salary = f(Basic_pay))

10. **Employee → Phone No** (One-to-One or One-to-Many)

    Each employee has one phone number (though this could be modeled for multiple).

    (Employee.phn_no)

11. **Customer → Email / Contact** (One-to-One)

    Each customer has one email and contact number.

    (Customer.e_mail, Customer.c_contact)

12. **Employee → Email / Type / DOB / Age** (One-to-One)

    These are employee attributes for tracking identity and employment type.

    (Employee.e_mail, e_type, e_dob, e_age)

13. **Admin → Email / Password** (One-to-One)

    Each Admin is uniquely identified by their email and password.

    (Admin.email, Admin.password)

14. **Medicines → Category (Syrup, Tablet, Capsule)** (One-to-One Multivalue)

    Each medicine belongs to one category; types are Syrup, Tablet, or Capsule.

    (Medicines.category → {Syrup, Tablet, Capsule})

15. **Medicines → Supplier (via Supply)**

    Medicines are supplied by a supplier; 'Supply' is the relation connecting them.

    (Supplier.sup_id → Medicine via Supply)

ER Diagram:

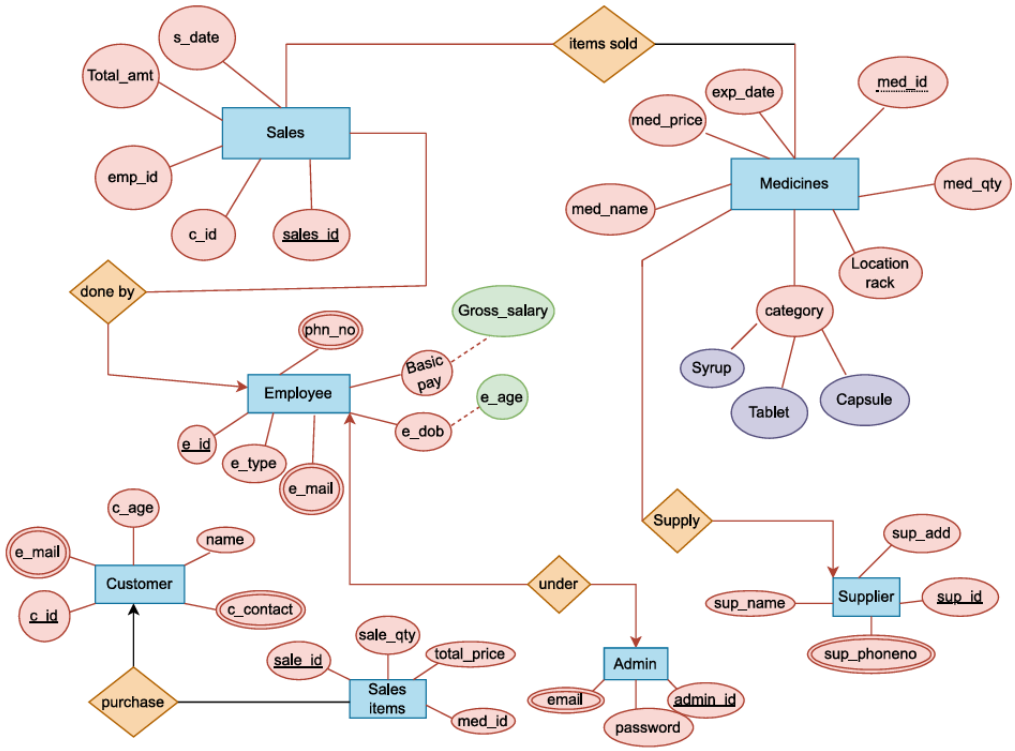
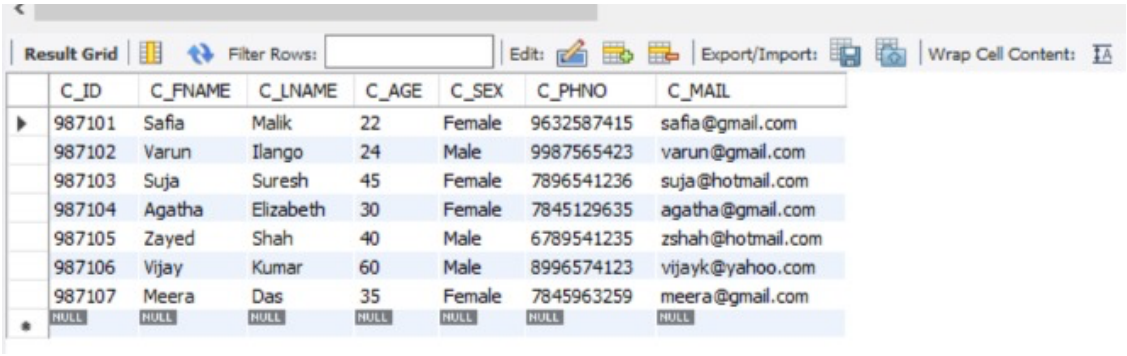


Fig: 2.1

2.5: DATABASE Tables:

I. **CUSTOMER –**

```
CREATE TABLE `customer` (
 `C_ID` decimal(6,0) NOT NULL,
 `C_FNAME` varchar(30) NOT NULL,
 `C_LNAME` varchar(30) DEFAULT NULL,
 `C_AGE` int NOT NULL,
 `C_SEX` varchar(6) NOT NULL,
 `C_PHNO` decimal(10,0) NOT NULL,
 `C_MAIL` varchar(40) DEFAULT NULL,
 PRIMARY KEY (`C_ID`),
 UNIQUE KEY `C_PHNO` (`C_PHNO`),
 UNIQUE KEY `C_MAIL` (`C_MAIL`)
)
```

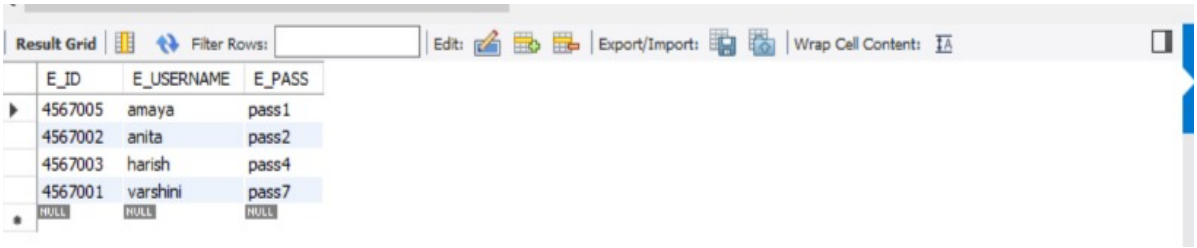


C_ID	C_FNAME	C_LNAME	C_AGE	C_SEX	C_PHNO	C_MAIL
987101	Safia	Malik	22	Female	9632587415	safia@gmail.com
987102	Varun	Ilango	24	Male	9987565423	varun@gmail.com
987103	Suja	Suresh	45	Female	7896541236	suja@hotmail.com
987104	Agatha	Elizabeth	30	Female	7845129635	agatha@gmail.com
987105	Zayed	Shah	40	Male	6789541235	zshah@hotmail.com
987106	Vijay	Kumar	60	Male	8996574123	vijayk@yahoo.com
987107	Meera	Das	35	Female	7845963259	meera@gmail.com

Fig: 2.2

II. EMPLOGIN –

```
CREATE TABLE `emplogin` (
 `E_ID` decimal(7,0) NOT NULL,
 `E_USERNAME` varchar(20) NOT NULL,
 `E_PASS` varchar(30) NOT NULL,
 PRIMARY KEY (`E_USERNAME`),
 KEY `E_ID` (`E_ID`),
 CONSTRAINT `emplogin_ibfk_1` FOREIGN KEY (`E_ID`) REFERENCES `employee`
(`E_ID`)
)
```



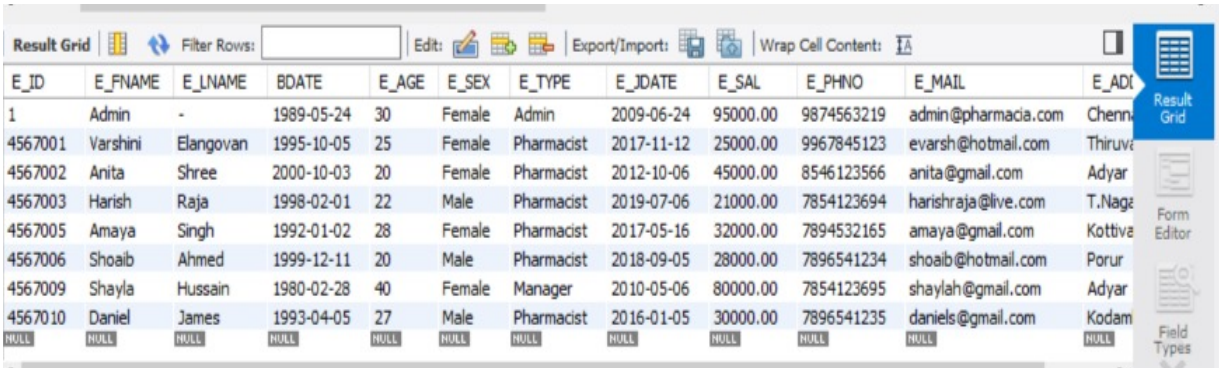
E_ID	E_USERNAME	E_PASS
4567005	amaya	pass1
4567002	anita	pass2
4567003	harish	pass4
4567001	varshini	pass7

Fig: 2.3

III. EMPLOYEE –

```
CREATE TABLE `employee` (
 `E_ID` decimal(7,0) NOT NULL,
 `E_FNAME` varchar(30) NOT NULL,
 `E_LNAME` varchar(30) DEFAULT NULL,
 `BDATE` date NOT NULL,
 `E_AGE` int NOT NULL,
 `E_SEX` varchar(6) NOT NULL,
 `E_TYPE` varchar(20) NOT NULL,
 `E_JDATE` date NOT NULL,
 `E_SAL` decimal(8,2) NOT NULL,
 `E_PHNO` decimal(10,0) NOT NULL,
 `E_MAIL` varchar(40) DEFAULT NULL,
 `E_ADD` varchar(40) DEFAULT NULL,
 PRIMARY KEY (`E_ID`)
```

)

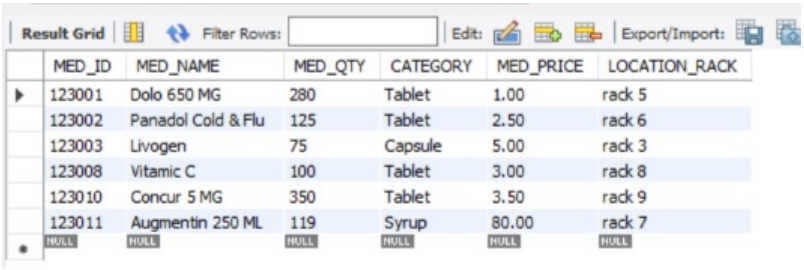


E_ID	E_FNAME	E_LNAME	BDATE	E_AGE	E_SEX	E_TYPE	E_JDATE	E_SAL	E_PHNO	E_MAIL	E_ADD
1	Admin	-	1989-05-24	30	Female	Admin	2009-06-24	95000.00	9874563219	admin@pharmacia.com	Chenn
4567001	Varshini	Elangovan	1995-10-05	25	Female	Pharmacist	2017-11-12	25000.00	9967845123	evarsh@hotmail.com	Thiruv
4567002	Anita	Shree	2000-10-03	20	Female	Pharmacist	2012-10-06	45000.00	8546123566	anita@gmail.com	Adyar
4567003	Harish	Raja	1998-02-01	22	Male	Pharmacist	2019-07-06	21000.00	7854123694	harishraja@live.com	T.Naga
4567005	Amaya	Singh	1992-01-02	28	Female	Pharmacist	2017-05-16	32000.00	7894532165	amaya@gmail.com	Kottiva
4567006	Shoaib	Ahmed	1999-12-11	20	Male	Pharmacist	2018-09-05	28000.00	7896541234	shoaib@hotmail.com	Porur
4567009	Shayla	Hussain	1980-02-28	40	Female	Manager	2010-05-06	80000.00	7854123695	shaylah@gmail.com	Adyar
4567010	Daniel	James	1993-04-05	27	Male	Pharmacist	2016-01-05	30000.00	7896541235	daniels@gmail.com	Kodam

Fig: 2.4

IV. MEDS –

```
CREATE TABLE `meds` (
 `MED_ID` decimal(6,0) NOT NULL,
 `MED_NAME` varchar(50) NOT NULL,
 `MED_QTY` int NOT NULL,
 `CATEGORY` varchar(20) DEFAULT NULL,
 `MED_PRICE` decimal(6,2) NOT NULL,
 `LOCATION_RACK` varchar(30) DEFAULT NULL,
 PRIMARY KEY (`MED_ID`)
```



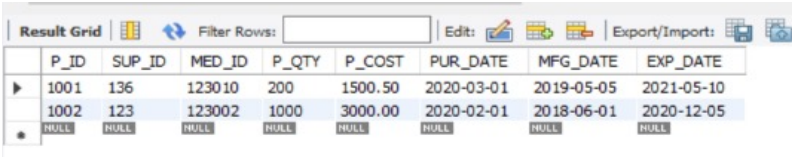
MED_ID	MED_NAME	MED_QTY	CATEGORY	MED_PRICE	LOCATION_RACK
123001	Dolo 650 MG	280	Tablet	1.00	rack 5
123002	Panadol Cold & Flu	125	Tablet	2.50	rack 6
123003	Livogen	75	Capsule	5.00	rack 3
123008	Vitamic C	100	Tablet	3.00	rack 8
123010	Concur 5 MG	350	Tablet	3.50	rack 9
123011	Augmentin 250 ML	119	Syrup	80.00	rack 7

Fig: 2.5

V. PURCHASE –

```
CREATE TABLE `purchase` (
 `P_ID` decimal(4,0) NOT NULL,
 `SUP_ID` decimal(3,0) NOT NULL,
 `MED_ID` decimal(6,0) NOT NULL,
 `P_QTY` int NOT NULL,
 `P_COST` decimal(8,2) NOT NULL,
 `PUR_DATE` date NOT NULL,
 `MFG_DATE` date NOT NULL,
 `EXP_DATE` date NOT NULL,
 PRIMARY KEY (`P_ID`,`MED_ID`),
 KEY `SUP_ID` (`SUP_ID`),
 KEY `MED_ID` (`MED_ID`),
```

CONSTRAINT `purchase_ibfk_1` FOREIGN KEY (`SUP_ID`) REFERENCES `suppliers` (`SUP_ID`),
CONSTRAINT `purchase_ibfk_2` FOREIGN KEY (`MED_ID`) REFERENCES `meds` (`MED_ID`)
)

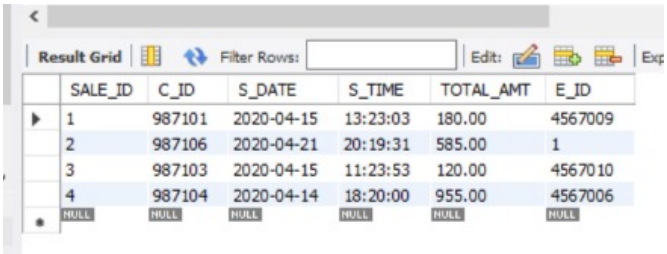


P_ID	SUP_ID	MED_ID	P_QTY	P_COST	PUR_DATE	MFG_DATE	EXP_DATE
1001	136	123010	200	1500.50	2020-03-01	2019-05-05	2021-05-10
1002	123	123002	1000	3000.00	2020-02-01	2018-06-01	2020-12-05
NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL

Fig: 2.6

VI. SALES –

CREATE TABLE `sales` (
`SALE_ID` int NOT NULL AUTO_INCREMENT,
`C_ID` decimal(6,0) NOT NULL,
`S_DATE` date DEFAULT NULL,
`S_TIME` time DEFAULT NULL,
`TOTAL_AMT` decimal(8,2) DEFAULT NULL,
`E_ID` decimal(7,0) NOT NULL,
PRIMARY KEY (`SALE_ID`),
KEY `C_ID` (`C_ID`),
KEY `E_ID` (`E_ID`),
CONSTRAINT `sales_ibfk_1` FOREIGN KEY (`C_ID`) REFERENCES `customer` (`C_ID`),
CONSTRAINT `sales_ibfk_2` FOREIGN KEY (`E_ID`) REFERENCES `employee` (`E_ID`)
)



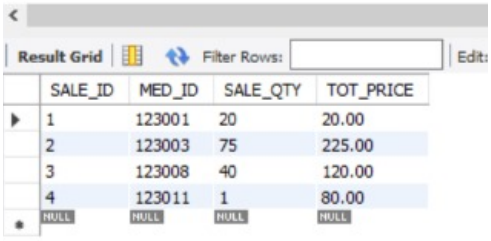
SALE_ID	C_ID	S_DATE	S_TIME	TOTAL_AMT	E_ID
1	987101	2020-04-15	13:23:03	180.00	4567009
2	987106	2020-04-21	20:19:31	585.00	1
3	987103	2020-04-15	11:23:53	120.00	4567010
4	987104	2020-04-14	18:20:00	955.00	4567006
NULL	NULL	NULL	NULL	NULL	NULL

Fig 2.7:

VII. SALES ITEMS –

CREATE TABLE `sales_items` (
`SALE_ID` int NOT NULL,
`MED_ID` decimal(6,0) NOT NULL,
`SALE_QTY` int NOT NULL,
`TOT_PRICE` decimal(8,2) NOT NULL,
PRIMARY KEY (`SALE_ID`,`MED_ID`),
KEY `MED_ID` (`MED_ID`),

```
 CONSTRAINT `sales_items_ibfk_1` FOREIGN KEY (`SALE_ID`) REFERENCES `sales`
(`SALE_ID`),
 CONSTRAINT `sales_items_ibfk_2` FOREIGN KEY (`MED_ID`) REFERENCES `meds`
(`MED_ID`)
)
```



SALE_ID	MED_ID	SALE_QTY	TOT_PRICE
1	123001	20	20.00
2	123003	75	225.00
3	123008	40	120.00
4	123011	1	80.00
NULL	NULL	NULL	NULL

Fig: 2.8

VIII. SUPPLIERS –

```
CREATE TABLE `suppliers` (
 `SUP_ID` decimal(3,0) NOT NULL,
 `SUP_NAME` varchar(25) NOT NULL,
 `SUP_ADD` varchar(30) NOT NULL,
 `SUP_PHNO` decimal(10,0) NOT NULL,
 `SUP_MAIL` varchar(40) NOT NULL,
 PRIMARY KEY (`SUP_ID`)
)
```

CHAPTER 3: 2B EXPERIMENT

2B: Design of Relational Schemas, Creation of Database and Tables for their project

3.1 Schemas:

- CUSTOMER TABLE:

schema = customer(c_id, c_fname, c_lname, c_age, c_sex, c_phno, c_mail)

- EMPLOGIN TABLE:

schema = emplogin(e_id, e_username, e_pass)

- EMPLOYEE TABLE:

schema = employee(e_id, e_fname, e_lname, bdate, e_age, e_sex, e_type, e_jdate, e_sal, e_phno, e_mail, e_add)

- MEDS TABLE:

schema = meds(med_id, med_name, med_qty, category, med_price, location_rack)

- PURCHASE TABLE:

schema = purchase(p_id, sup_id, med_id, p_qty, p_cost, pur_date, mfg_date, exp_date)

- SALES TABLE:

schema = sales(sale_id, c_id, s_date, s_time, total_amt, e_id)

- SALES_ITEMS TABLE:

schema = sales_items(sale_id, med_id, sale_qty, tot_price)

- SUPPLIERS TABLE:

schema = suppliers(sup_id, sup_name, sup_add, sup_phno, sup_mail)

RELATIONAL DIAGRAM:

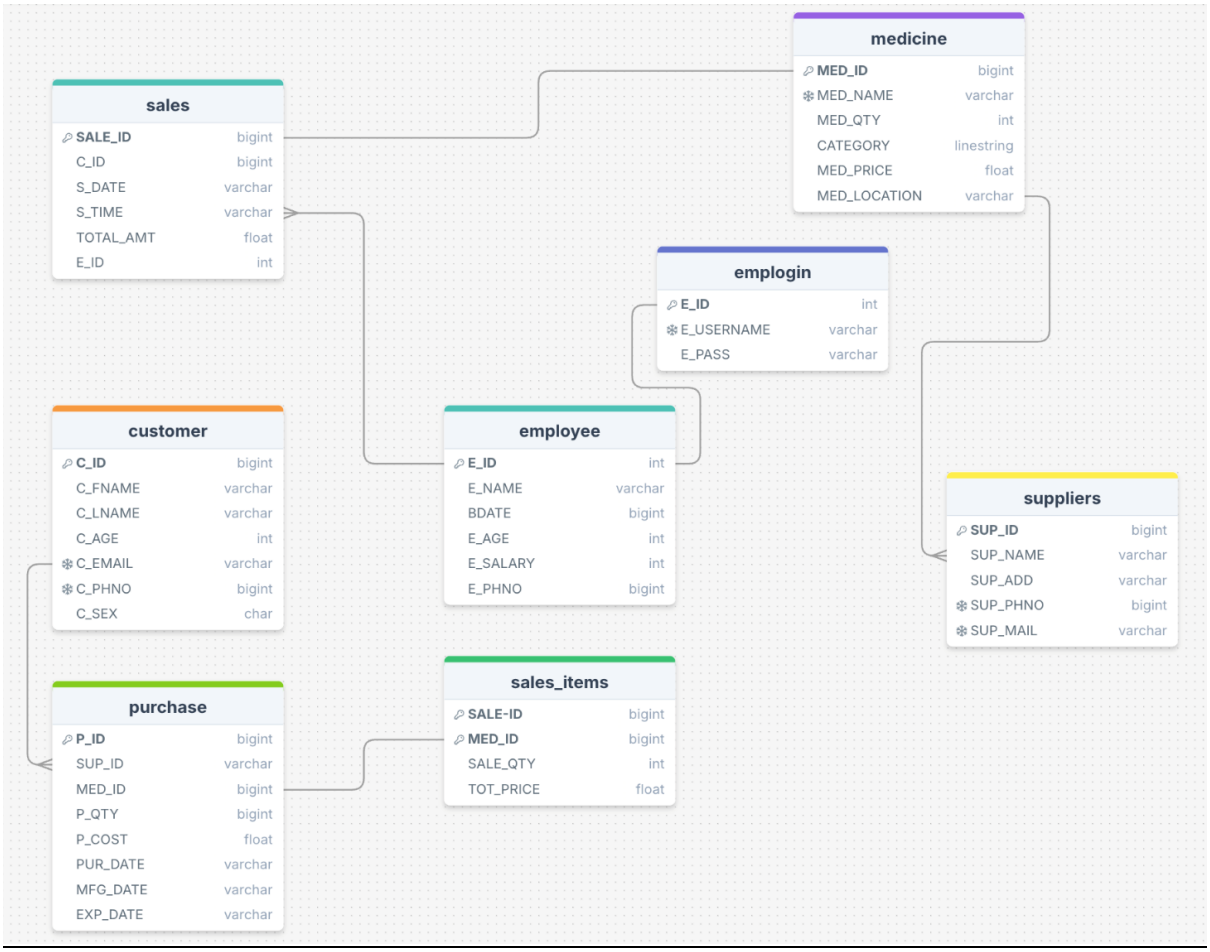


Fig: 3.1

CHAPTER 4 : 3B EXPERIMENT

3B: Writing complex queries based on the concept of constraints, sets, joins, views, triggers, and cursors.

4.1.1 CONSTRAINTS:

-- Step 1: Modify column definitions

ALTER TABLE purchase

MODIFY P_QTY INT NOT NULL DEFAULT 1,

MODIFY P_COST DECIMAL(8,2) NOT NULL DEFAULT 0.00,

MODIFY EXP_DATE DATE NOT NULL;

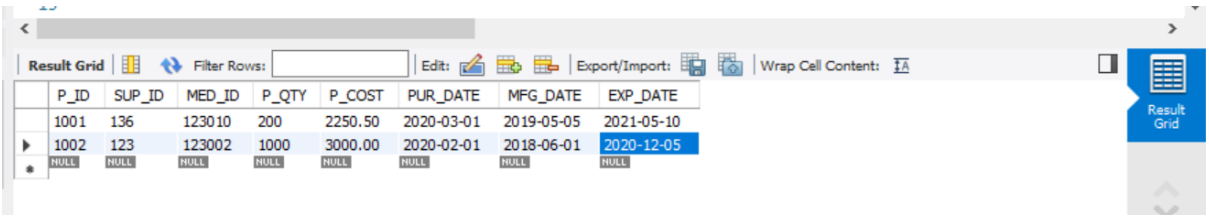
-- Step 2: Add check constraints

ALTER TABLE purchase

ADD CONSTRAINT chk_pur_qty CHECK (P_QTY > 2200),

ADD CONSTRAINT chk_pur_cost CHECK (P_COST >= 0),

ADD CONSTRAINT chk_exp_gt_mfg CHECK (EXP_DATE > MFG_DATE);



P_ID	SUP_ID	MED_ID	P_QTY	P_COST	PUR_DATE	MFG_DATE	EXP_DATE
1001	136	123010	200	2250.50	2020-03-01	2019-05-05	2021-05-10
1002	123	123002	1000	3000.00	2020-02-01	2018-06-01	2020-12-05
NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL

Fig: 4.1

4.1.2 SETS:

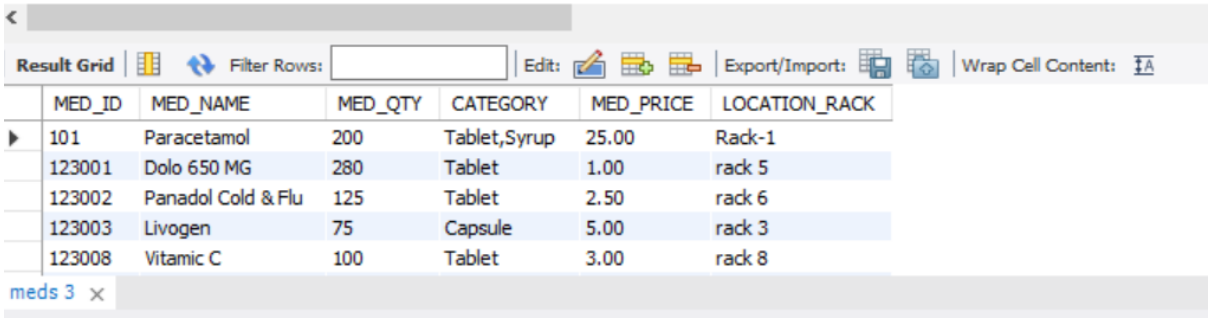
SELECT * FROM pharmacy.meds;

ALTER TABLE med

MODIFY CATEGORY SET('Tablet', 'Syrup', 'Injection', 'Capsule', 'Ointment') DEFAULT NULL;

INSERT INTO meds (MED_ID, MED_NAME, MED_QTY, CATEGORY, MED_PRICE, LOCATION_RACK)

VALUES (101, 'Paracetamol', 200, 'Tablet,Syrup', 25.00, 'Rack-1');



Result Grid | Filter Rows: | Edit: | Export/Import: | Wrap Cell Content:

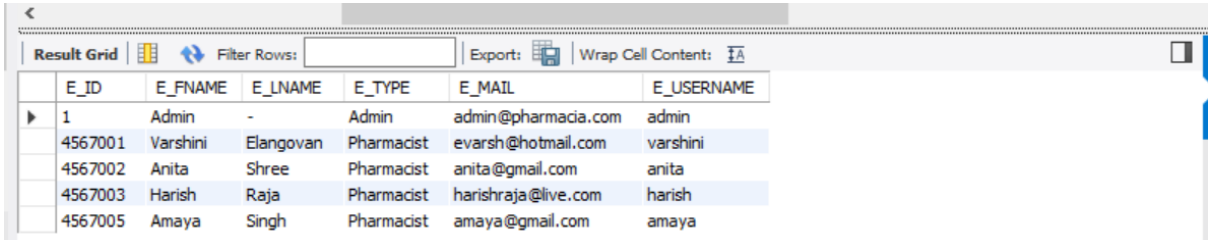
MED_ID	MED_NAME	MED_QTY	CATEGORY	MED_PRICE	LOCATION_RACK
101	Paracetamol	200	Tablet,Syrup	25.00	Rack-1
123001	Dolo 650 MG	280	Tablet	1.00	rack 5
123002	Panadol Cold & Flu	125	Tablet	2.50	rack 6
123003	Livogen	75	Capsule	5.00	rack 3
123008	Vitamic C	100	Tablet	3.00	rack 8

meds 3

Fig: 4.2

4.1.3 JOIN:

```
SELECT
  e.E_ID,
  e.E_FNAME,
  e.E_LNAME,
  e.E_TYPE,
  e.E_MAIL,
  l.E_USERNAME
FROM
  employee e
JOIN
  emplogin l ON e.E_ID = l.E_ID;
```



Result Grid | Filter Rows: | Export: | Wrap Cell Content:

E_ID	E_FNAME	E_LNAME	E_TYPE	E_MAIL	E_USERNAME
1	Admin	-	Admin	admin@pharmacia.com	admin
4567001	Varshini	Elangovan	Pharmacist	evarsh@hotmail.com	varshini
4567002	Anita	Shree	Pharmacist	anita@gmail.com	anita
4567003	Harish	Raja	Pharmacist	harishraja@live.com	harish
4567005	Amaya	Singh	Pharmacist	amaya@gmail.com	amaya

Fig: 4.3

4.1.4. VIEW:

```
SELECT * FROM emp_login_view;

CREATE VIEW emp_login_view AS
SELECT
  e.E_ID,
  e.E_FNAME,
  e.E_LNAME,
```

```
  e.E_TYPE,
  e.E_MAIL,
  l.E_USERNAME
FROM
  employee e
JOIN
  emplogin l ON e.E_ID = l.E_ID;
```

```
SELECT * FROM emp_login_view;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

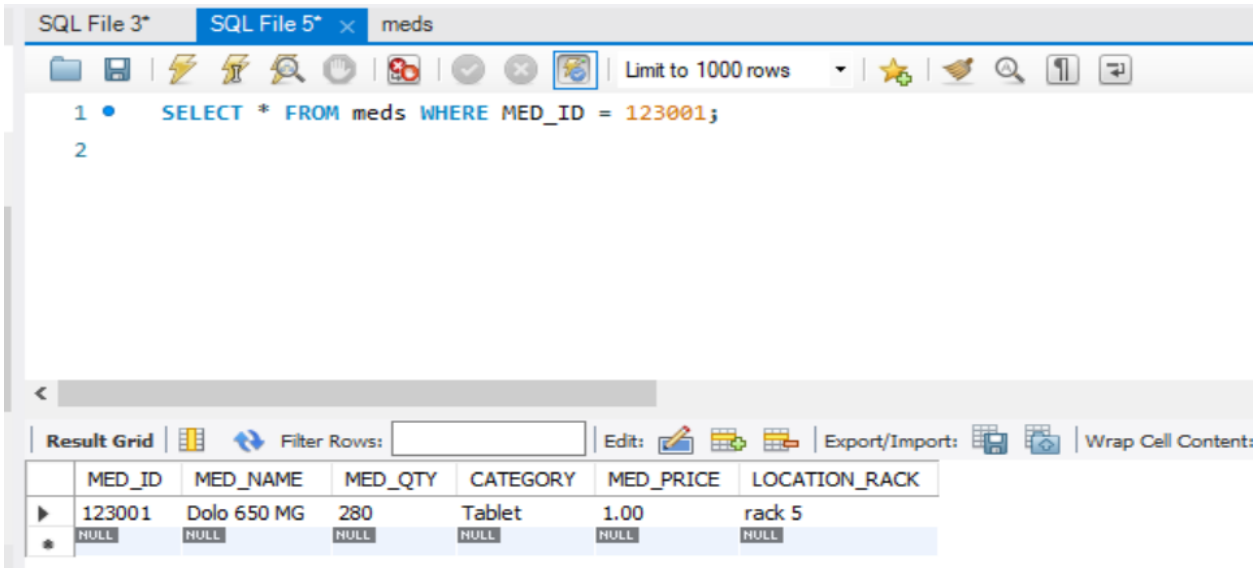
E_ID	E_FNAME	E_LNAME	E_TYPE	E_MAIL	E_USERNAME
1	Admin	-	Admin	admin@pharmacia.com	admin
4567001	Varshini	Elangovan	Pharmacist	evarsh@hotmail.com	varshini
4567002	Anita	Shree	Pharmacist	anita@gmail.com	anita
4567003	Harish	Raja	Pharmacist	harishraja@live.com	harish
4567005	Amaya	Singh	Pharmacist	amaya@gmail.com	amaya

Fig: 4.4

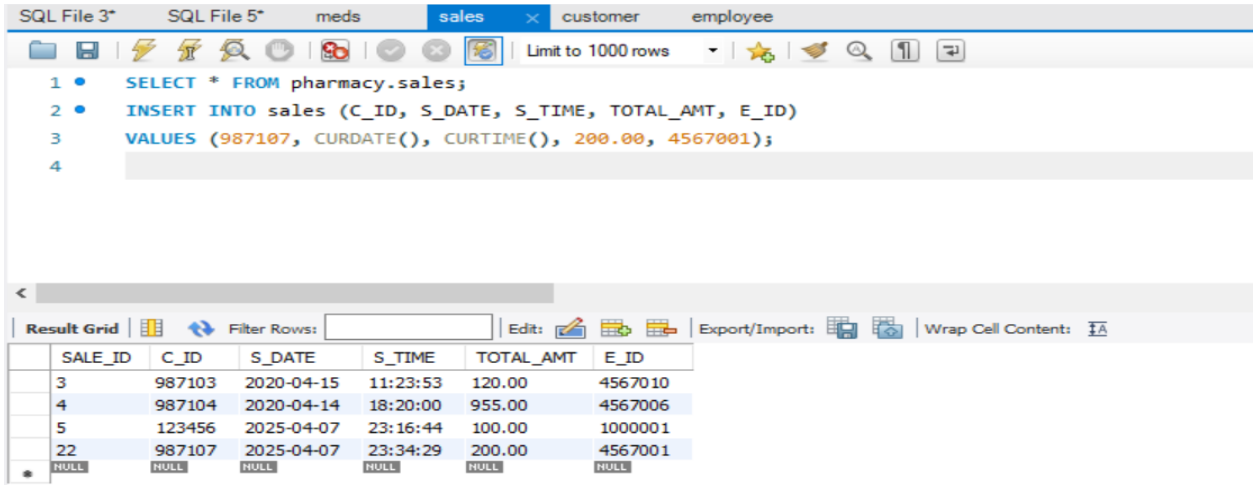
4.1.5 TRIGGER:

```
DELIMITER $$
CREATE TRIGGER after_sale_update_stock
AFTER INSERT ON sales_items
FOR EACH ROW
BEGIN
  UPDATE meds
  SET MED_QTY = MED_QTY - NEW.SALE_QTY
  WHERE MED_ID = NEW.MED_ID;
END$$
DELIMITER ;
```

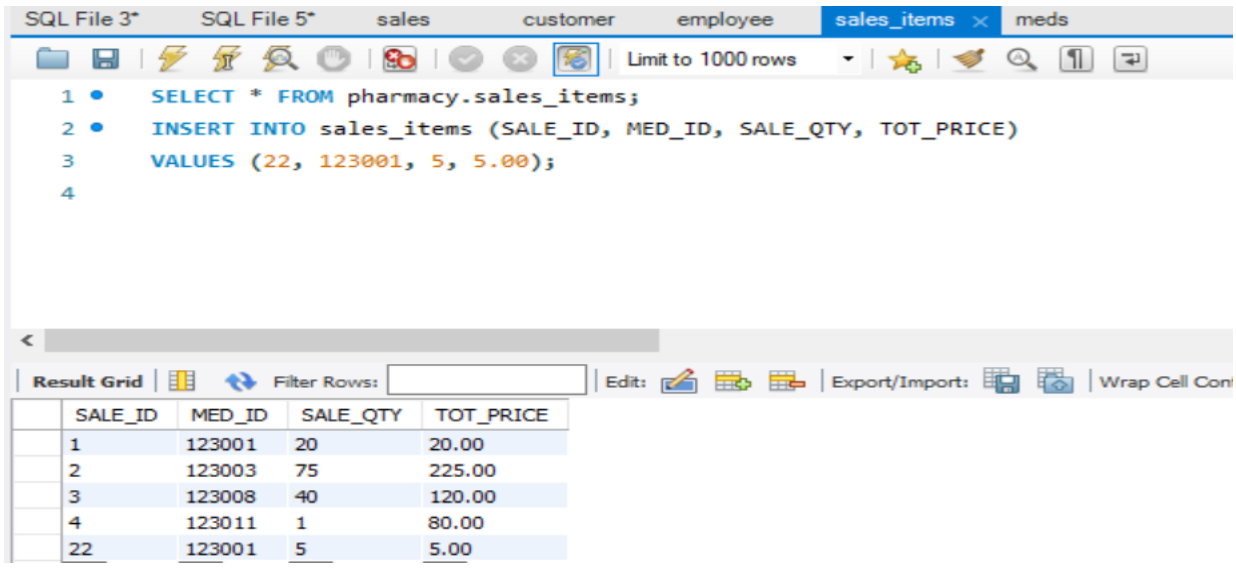
Step 1) Before Inserting any sales items



Step 2) Inserting data in sales table



Step 3) Inserting data in sales items table



Step 4) Trigger reflected in the med table.

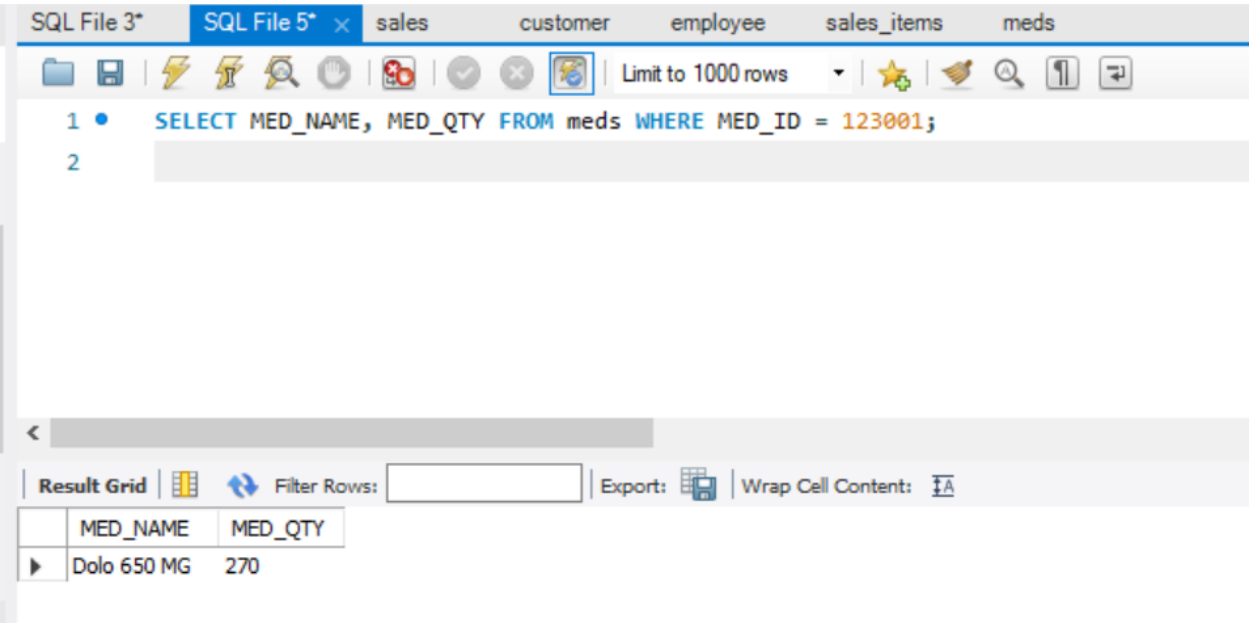


Fig: 4.5 to 4.8

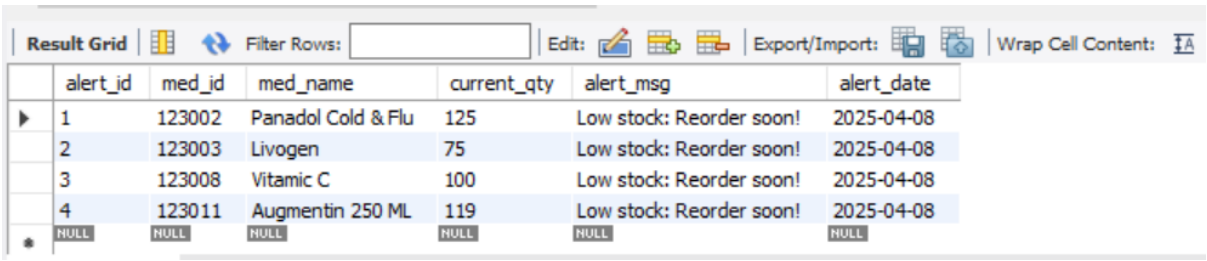
4.1.6 CURSOR:

```
CREATE TABLE low_stock_alerts (
    alert_id INT AUTO_INCREMENT PRIMARY KEY,
    med_id DECIMAL(6,0),
    med_name VARCHAR(50),
    current_qty INT,
    alert_msg VARCHAR(100),
    alert_date DATE
);
DELIMITER $$
CREATE PROCEDURE generate_low_stock_alerts()
BEGIN
  DECLARE done INT DEFAULT FALSE;
  DECLARE v_med_id DECIMAL(6,0);
  DECLARE v_med_name VARCHAR(50);
  DECLARE v_qty INT;
  DECLARE med_cursor CURSOR FOR
    SELECT MED_ID, MED_NAME, MED_QTY FROM meds;
```

```
  DECLARE CONTINUE HANDLER FOR NOT FOUND SET done = TRUE;
  OPEN med_cursor;
  read_loop: LOOP
    FETCH med_cursor INTO v_med_id, v_med_name, v_qty;
    IF done THEN
      LEAVE read_loop;
    END IF;

    IF v_qty <= 150 THEN
      INSERT INTO low_stock_alerts (med_id, med_name, current_qty, alert_msg, alert_date)
      VALUES (v_med_id, v_med_name, v_qty, 'Low stock: Reorder soon!', CURDATE());
    END IF;
  END LOOP;
  CLOSE med_cursor;
END$$
DELIMITER ;
```

CALL generate_low_stock_alerts();



alert_id	med_id	med_name	current_qty	alert_msg	alert_date
1	123002	Panadol Cold & Flu	125	Low stock: Reorder soon!	2025-04-08
2	123003	Livogen	75	Low stock: Reorder soon!	2025-04-08
3	123008	Vitamic C	100	Low stock: Reorder soon!	2025-04-08
4	123011	Augmentin 250 ML	119	Low stock: Reorder soon!	2025-04-08
NULL	NULL	NULL	NULL	NULL	NULL

Fig: 4.9

CHAPTER 5: 4B EXPERIMENT

4B: Analyzing the pitfalls, identifying the dependencies, and applying normalizations

5.1: Pitfalls:

1. Customer

i. Unnormalized form (UNF)

C_ID	C_FNAME	C_LNAME	C_AGE	C_SEX	C_PHONE	C_EMAIL	C_ORDERS
123456	Test	User	30	Male	9876543210, 9876543211	testuser@example.com, testuser2@example.com	Order1, Order2
987101	Safia	Malik	22	Female	9632587415	safia@gmail.com	Order1
987102	Varun	Ilango	24	Male	9987565423	varun@gmail.com	Order1, Order2
987103	Suja	Suresh	45	Female	7896541236	suja@hotmail.com	Order1
987104	Agatha	Elizabeth	30	Female	7845129635	agatha@gmail.com	Order2
987105	Zayed	Shah	40	Male	6789541235	zshah@hotmail.com	Order1, Order3
987106	Vijay	Kumar	60	Male	8996574123	vijayk@yahoo.com	Order1
987107	Meera	Das	35	Female	7845963259	meera@gmail.com	Order2
NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL

Fig: 5.1

Data Anomalies in UNF:

- Repeating groups (multiple phone numbers, emails, and orders).
- Data redundancy and inconsistency.
- Difficult to perform search, updates, and inserts.

Functional Dependencies:

- C_ID → C_FNAME, C_LNAME, C_AGE, C_SEX, C_PHONE, C_EMAIL, C_ORDERS.

ii. First Normal Form (1NF)

```
CREATE TABLE customer_1NF (
    C_ID INT,
    C_FNAME VARCHAR(50),
    C_LNAME VARCHAR(50),
    C_AGE INT,
    C_SEX VARCHAR(10),
```

```
    C_PHONE VARCHAR(15),
    C_EMAIL VARCHAR(100),
    C_ORDER VARCHAR(50)
);

INSERT INTO customer_1nf (C_ID, C_FNAME, C_LNAME, C_AGE, C_SEX,
C_PHONE, C_EMAIL, C_ORDER) VALUES
(123456, 'Test', 'User', 30, 'Male', '9876543210', 'testuser@example.com', 'Order1'),
(123456, 'Test', 'User', 30, 'Male', '9876543211', 'testuser2@example.com', 'Order2'),
(987101, 'Safia', 'Malik', 22, 'Female', '9632587415', 'safia@gmail.com', 'Order1'),
(987102, 'Varun', 'Ilango', 24, 'Male', '9987565423', 'varun@gmail.com', 'Order1'),
(987102, 'Varun', 'Ilango', 24, 'Male', '9987565423', 'varun@gmail.com', 'Order2'),
(987103, 'Suja', 'Suresh', 45, 'Female', '7896541236', 'suja@hotmail.com', 'Order1');
```

Justification:

- Repeating groups split into separate rows.
- Each attribute has atomic values.

C_ID	C_FNAME	C_LNAME	C_AGE	C_SEX	C_PHONE	C_EMAIL	C_ORDER
123456	Test	User	30	Male	9876543210	testuser@example.com	Order1
123456	Test	User	30	Male	9876543211	testuser2@example.com	Order2
123456	Test	User	30	Male	9876543210	testuser@example.com	Order1
123456	Test	User	30	Male	9876543211	testuser2@example.com	Order2
987101	Safia	Malik	22	Female	9632587415	safia@gmail.com	Order1
987102	Varun	Ilango	24	Male	9987565423	varun@gmail.com	Order1
987102	Varun	Ilango	24	Male	9987565423	varun@gmail.com	Order2
987103	Suja	Suresh	45	Female	7896541236	suja@hotmail.com	Order1

Fig: 5.2

iii. Second Normal Form (2NF)

```
CREATE TABLE customer_2nf (
    C_ID INT PRIMARY KEY,
```

C_FNAME VARCHAR(50),

C_LNAME VARCHAR(50),

C_AGE INT,

C_SEX VARCHAR(10)

);

C_ID	C_FNAME	C_LNAME	C_AGE	C_SEX
123456	Test	User	30	Male
987101	Safia	Malik	22	Female
987102	Varun	Ilango	24	Male
987103	Suja	Suresh	45	Female
987104	Agatha	Elizabeth	30	Female
987105	Zayed	Shah	40	Male
987106	Vijay	Kumar	60	Male
987107	Meera	Das	35	Female

Fig 5.3

CREATE TABLE **customer_contact** (

C_ID INT,

C_PHONE VARCHAR(20),

C_EMAIL VARCHAR(100),

FOREIGN KEY (C_ID) REFERENCES customer_2nf(C_ID));

C_ID	C_PHONE	C_EMAIL
123456	9876543210	testuser@example.com
123456	9876543211	testuser2@example.com
987101	9632587415	safia@gmail.com
987102	9987565423	varun@gmail.com
987103	7896541236	suja@hotmail.com
987104	7845129635	agatha@gmail.com
987105	6789541235	zshah@hotmail.com
987106	8996574123	vijayk@yahoo.com

Fig: 5.4

CREATE TABLE **customer_orders** (

C_ID INT,

C_ORDERS VARCHAR(50),

FOREIGN KEY (C_ID) REFERENCES customer_2nf(C_ID)

);

C_ID	C_ORDERS
123456	Order1
123456	Order2
987101	Order1
987102	Order1
987102	Order2
987103	Order1
987104	Order2
987105	Order1

Fig: 5.5

INSERT INTO customer_2nf (C_ID, C_FNAME, C_LNAME, C_AGE, C_SEX) VALUES

(123456, 'Test', 'User', 30, 'Male'),

(987101, 'Safia', 'Malik', 22, 'Female'),

(987102, 'Varun', 'Ilango', 24, 'Male');

INSERT INTO customer_contact (C_ID, C_PHONE, C_EMAIL) VALUES

(123456, '9876543210', 'testuser@example.com'),

(987105, '6789541235', 'zshah@hotmail.com'),

(987106, '8996574123', 'vijayk@yahoo.com'),

(123456, '9876543211', 'testuser2@example.com');

INSERT INTO customer_orders (C_ID, C_ORDERS) VALUES

(123456, 'Order1'),

(123456, 'Order2'),

(987101, 'Order1'),

(987102, 'Order1');

Justification for 2NF:

- **Eliminated Partial Dependency**: The customer contact and order information is now separated into different tables.
- **Full Dependency**: All non-key attributes (C_PHONE, C_EMAIL, C_ORDERS) now depend entirely on the primary key (C_ID).
- **Reduced Redundancy**: Customer details are stored only once in the Customer table, and contact/order info is stored in separate tables.

Justification for 3NF:

- Separated data into three tables to remove partial and transitive dependencies.
- Each table now contains attributes directly dependent on the primary key.
- Structure is in **3NF**, no further changes needed.

2. EMPLOGIN -

E_ID	E_USERNAME	E_PASS
1	admin	adminpass
4567005	amaya	pass1
4567002	anita	pass2
4567003	harish	pass4
4567001	varshini	pass7
NULL	NULL	NULL

Fig: 5.6

Justification :

The emplogin table is already in 3NF: All attributes are atomic (**1NF**), fully dependent on the primary key E_ID (**2NF**), and no transitive dependencies exist (3NF).

- **Functional Dependency**: E_ID → Username, Password
- **Conclusion**: No further normalization needed. Table is well-structured.

- **Data Anomalies**: None – table is already normalized.

3. MEDS -

i. Unnormalized form (UNF)

- Multiple forms in a single column (e.g., "Tablet, Syrup")
- Data anomalies possible - (insertion, update, deletion)

MED_ID	MED_NAME	MED_QTY	CATEGORY	MED_PRICE	LOCATION_RACK
101	Paracetamol	180	Tablet,Syrup	25.00	Rack-1
123001	Dolo 650 MG	270	Tablet	1.00	rack 5
123002	Panadol Cold & Flu	125	Tablet	2.50	rack 6
123003	Livogen	75	Capsule	5.00	rack 3
123008	Vitamic C	100	Tablet	3.00	rack 8
123010	Concur 5 MG	350	Tablet	3.50	rack 9
123011	Augmentin 250 ML	119	Syrup	80.00	rack 7

Fig: 5.7

ii. First Normal form (1NF)

- Remove multivalued attributes
- Create separate rows for each form
- **Functional Dependencies** - M_ID → M_NAME, M_QUANTITY, M_TYPE, M_PRICE, M_LOCATION

```
CREATE TABLE meds_1nf (
    M_ID INT,
    M_NAME VARCHAR(100),
    M_QUANTITY INT,
    M_TYPE VARCHAR(50),
    M_PRICE DECIMAL(10, 2),
    M_LOCATION VARCHAR(50)
);
-- Insert normalized rows
INSERT INTO meds_1nf VALUES
(101, 'Paracetamol', 180, 'Tablet', 25.00, 'Rack-1'),
(101, 'Paracetamol', 180, 'Syrup', 25.00, 'Rack-1'),
(123001, 'Dolo 650 MG', 270, 'Tablet', 1.00, 'rack 5'),
(123002, 'Panadol Cold & Flu', 125, 'Tablet', 2.50, 'rack 6'),
(123003, 'Livogen', 75, 'Capsule', 5.00, 'rack 3');
```

M_ID	M_NAME	M_QUANTITY	M_TYPE	M_PRICE	M_LOCATION
101	Paracetamol	180	Tablet	25.00	Rack-1
101	Paracetamol	180	Syrup	25.00	Rack-1
123001	Dolo 650 MG	270	Tablet	1.00	rack 5
123002	Panadol Cold & Flu	125	Tablet	2.50	rack 6
123003	Livogen	75	Capsule	5.00	rack 3
123008	Vitamic C	100	Tablet	3.00	rack 8
123010	Concur 5 MG	350	Tablet	3.50	rack 9
123011	Augmentin 250 ML	119	Syrup	80.00	rack 7

Fig: 5.8

iii. Second Normal form (2NF)

Issues in 1NF:

- **Partial Dependency:** Attributes like M_NAME, M_QUANTITY, M_PRICE, and M_LOCATION depend only on **M_ID**, not on **M_TYPE**.

Functional Dependencies Identified:

- **M_ID → M_NAME, M_QUANTITY, M_PRICE, M_LOCATION** (M_ID determines all other attributes except M_TYPE).
- **(M_ID, M_TYPE) → M_ID** (M_ID is the only key needed for other attributes).

```
CREATE TABLE meds_master (
M_ID INT PRIMARY KEY,
M_NAME VARCHAR(100),
M_QUANTITY INT,
M_PRICE DECIMAL(10, 2),
M_LOCATION VARCHAR(50)
);
```

M_ID	M_NAME	M_QUANTITY	M_PRICE	M_LOCATION
101	Paracetamol	180	25.00	Rack-1
123001	Dolo 650 MG	270	1.00	rack 5
123002	Panadol Cold & Flu	125	2.50	rack 6
123003	Livogen	75	5.00	rack 3
123008	Vitamic C	100	3.00	rack 8
123010	Concur 5 MG	350	3.50	rack 9
123011	Augmentin 250 ML	119	80.00	rack 7
NULL	NULL	NULL	NULL	NULL

Fig: 5.9

```
CREATE TABLE meds_type (
    M_ID INT,
    M_TYPE VARCHAR(50),
    FOREIGN KEY (M_ID) REFERENCES meds_master(M_ID)
);
```

M_ID	M_TYPE
101	Tablet
101	Syrup
123001	Tablet
123002	Tablet
123003	Capsule
123008	Tablet
123010	Tablet
123011	Syrup

Fig: 5.10

```
-- meds_master
INSERT INTO meds_master VALUES
(101, 'Paracetamol', 180, 25.00, 'Rack-1'),
(123001, 'Dolo 650 MG', 270, 1.00, 'rack 5'),
(123002, 'Panadol Cold & Flu', 125, 2.50, 'rack 6'),
```

(123003, 'Livogen', 75, 5.00, 'rack 3');

-- meds_type

INSERT INTO meds_type VALUES

(101, 'Tablet'),

(101, 'Syrup'),

(123001, 'Tablet'),

(123002, 'Tablet'),

(123003, 'Capsule'),

Third Normal Form (3NF)

- No attribute is dependent on another non-key attribute.
- M_LOCATION is not dependent on M_NAME or M_TYPE, only on M_ID.
- Already in 3NF.

4. SALES –

- **1NF:** The sales table satisfies 1NF by having atomic values (no repeating groups or arrays).
- **2NF:** The table satisfies 2NF because there are no partial dependencies, i.e., all non-key columns depend on the full primary key.
- **3NF:** This structure ensures that the sales table only contains necessary foreign keys and eliminates unnecessary duplication of data.

The **functional dependencies** in the table are as follows:

- SALE_ID → C_ID, S_DATE, S_TIME, TOTAL_AMT, E_ID
- C_ID → C_NAME, C_ADDRESS (from the customer table)
- E_ID → E_NAME, E_ROLE (from the employee table)

SALE_ID	C_ID	S_DATE	S_TIME	TOTAL_AMT	E_ID
1	987101	2020-04-15	13:23:03	180.00	4567009
2	987106	2020-04-21	20:19:31	585.00	1
3	987103	2020-04-15	11:23:53	120.00	4567010
4	987104	2020-04-14	18:20:00	955.00	4567006
5	123456	2025-04-07	23:16:44	100.00	1000001
22	987107	2025-04-07	23:34:29	200.00	4567001
NULL	NULL	NULL	NULL	NULL	NULL

Fig:5.11

5. SALES ITEMS –

The table with attributes SALE_ID, MED_ID, SALE_QTY, TOT_PRICE is already in **1NF**, **2NF**, **3NF**, and **BCNF** based on the following reasons:

The **functional dependencies** in the table are:
SALE_ID, MED_ID → SALE_QTY, TOT_PRICE.

- **1NF**: The table contains atomic values.
- **2NF**: No partial dependencies.
- **3NF**: No transitive dependencies.
- **BCNF**: Every determinant is a superkey.

SALE_ID	MED_ID	SALE_QTY	TOT_PRICE
1	101	10	250.00
1	123001	20	20.00
2	123003	75	225.00
3	123008	40	120.00
4	123011	1	80.00
22	123001	5	5.00

Fig 5.12

6. SUPPLIERS–

i. Unnormalized form (UNF)

- Multiple dates stored together with potential redundancy.

SUP_ID	SUP_NAME	SUP_ADD	SUP_PHNO	SUP_MAIL
123	XYZ Pharmaceuticals	Chennai, Tamil Nadu	8745632145	xyz@xyzpharma.com
136	ABC PharmaSupply	Trichy, Tamil Nadu	7894561235	abc@pharmsupp.com
145	Daily Pharma Ltd	Hyderabad, Telangana	7854699321	daily@dpharma.com
156	MedAll	Chennai, Tamil Nadu	9874585236	mainid@medall.com
157	CareChem Pharma	Chennai, Tamil Nadu	8899001122	contact@carechem.com
158	NewGen Meds	Hyderabad, Telangana	7854699321	newgen@meds.com
159	LifeSaver Drugs	Trichy, Tamil Nadu	8123456789	lifesaver@pharmsupp.com
NULL	NULL	NULL	NULL	NULL

Fig 5.13

ii. First Normal Form (1NF)

- Atomic values in all fields
- No repeating groups or multivalued attributes

iii. Second Normal Form (2NF)

Functional Dependencies:

- SUP_ID → SUP_NAME, SUP_ADD, SUP_PHNO, SUP_MAIL

Since SUP_ID is the primary key, and all non-key attributes fully depend on it, **2NF is also satisfied**.

iv. Third Normal Form (3NF)

Transitive Dependency Exists:

- SUP_ID → SUP_ADD
- SUP_ADD → SUP_PHNO, SUP_MAIL

SUP_PHNO and SUP_MAIL are **indirectly dependent** on SUP_ID → violates **3NF**.

```
CREATE TABLE supplier_main (

   SUP_ID INT PRIMARY KEY,

   SUP_NAME VARCHAR(100),

   ADD_ID INT

);
```

SUP_ID	SUP_NAME	ADD_ID
123	XYZ Pharmaceuticals	1
136	ABC PharmaSupply	99
145	Daily Pharma Ltd	36
156	MedAll	81
157	CareChem Pharma	25
158	NewGen Meds	6
159	LifeSaver Drugs	17

Fig: 5.14

```
CREATE TABLE supplier_address (
   ADD_ID INT PRIMARY KEY,
   SUP_ADD VARCHAR(100),
   SUP_PHNO VARCHAR(15),
   SUP_MAIL VARCHAR(100)
);
```

ADD_ID	SUP_ADD	SUP_PHNO	SUP_MAIL
1	Chennai, Tamil Nadu	8745632145	xyz@xyzpharma.com
6	Hyderabad, Telangana	7854699321	newgen@meds.com
17	Trichy, Tamil Nadu	8123456789	lifesaver@pharmsupp.com
25	Chennai, Tamil Nadu	8899001122	contact@carechem.com
36	Hyderabad, Telangana	7854699321	daily@dpharma.com
81	Chennai, Tamil Nadu	9874585236	mainid@medall.com
99	Trichy, Tamil Nadu	7894561235	abc@pharmsupp.com

Fig: 5.15

```
INSERT INTO supplier_address VALUES
(1, 'Chennai, Tamil Nadu', '8745632145', 'xyz@xyzpharma.com'),
(99, 'Trichy, Tamil Nadu', '7894561235', 'abc@pharmsupp.com'),
(36, 'Hyderabad, Telangana', '7854699321', 'daily@dpharma.com'),
(81, 'Chennai, Tamil Nadu', '9874585236', 'mainid@medall.com');
```

INSERT INTO supplier_main VALUES

(123, 'XYZ Pharmaceuticals', 1),

(136, 'ABC PharmaSupply', 99),

(145, 'Daily Pharma Ltd', 36),

(156, 'MedAll', 81);

Functional Dependencies -

- SUP_ID → SUP_NAME, ADD_ID (in supplier_main)
- ADD_ID → SUP_ADD, SUP_PHNO, SUP_MAIL (in supplier_address)

All non-key attributes now depend **only on the key**, and **no transitive dependencies exist**.

5.2: NORMALISATION FLOW CHART

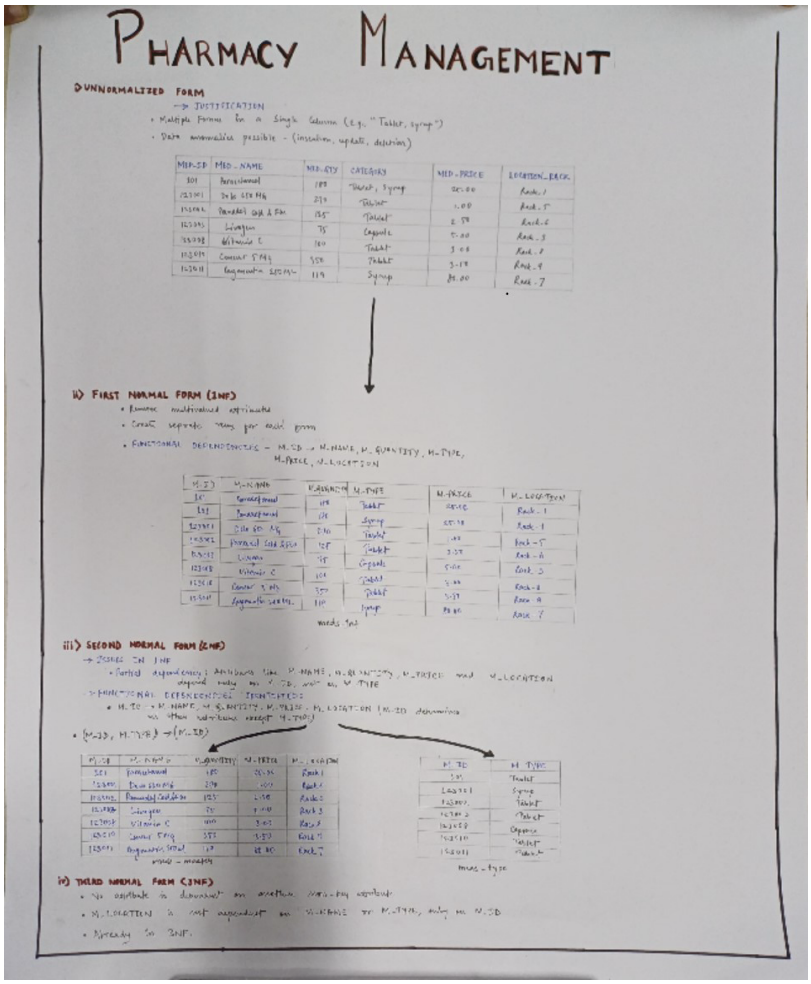


Fig: 5.1

CHAPTER 6: 5B EXPERIMENT

6.1: CONCURRENCY CONTROL

Concurrency control ensures that multiple pharmacy staff or customers can access or modify data like medicine stock, sales, or purchases simultaneously without causing inconsistencies or data conflicts.

Techniques to Implement:

1. **Locking Mechanisms**

   • *Row-Level Locking:* Locks specific rows being updated—for example, when updating medicine stock during a sale.

   • *Table-Level Locking:* Locks the entire table (e.g., locking the meds table during bulk inventory update).

2. **Timestamp Ordering**

   • Assigns a timestamp to each transaction in the system.

   • Ensures transactions like inventory update and purchase record entry are handled in the correct order to prevent conflict.

3. **Optimistic Concurrency**

   • Assumes low contention—for example, while updating employee profiles.

   • Checks for conflicts before committing using a version number or timestamp mechanism.

4. **Multiversion Concurrency Control (MVCC)**

   • Useful when staff access customer or medicine details simultaneously.

   • Maintains multiple versions of data to allow concurrent reads and writes without locking.

6.2: RECOVERY MECHANISM

Recovery ensures that the pharmacy database remains consistent and can recover gracefully after events like system crashes, power failures, or hardware issues.

Techniques to Implement:

1. **Transaction Logging (Write-Ahead Logging - WAL)**

   • All operations, such as sales or purchases, are logged before execution.

   • After a crash, the system uses the log to replay or undo transactions.

2. **Commit and Rollback**

   • Transactions like billing or inventory updates are grouped.

   • If failure occurs mid-operation, changes are rolled back to prevent corruption.

3. **Checkpointing**

   • Periodic snapshots of critical tables (e.g., sales, purchase) help reduce recovery time.

   • Reduces need to reprocess entire log.

4. **Shadow Paging**

   • Maintains a shadow copy of data during updates (e.g., stock updates).

   • Original data is preserved until transaction successfully commits.

Example Queries and Output :

1. use pharmacy_v1;

```
START TRANSACTION;

-- Lock the row for 'Piliton' so no one else can modify it

SELECT pharmacy_Qty

FROM pharmacy_stock

WHERE medicine_name = 'Piliton'

FOR UPDATE;

-- Simulate low stock scenario

-- Check if enough stock exists (example threshold = 5)

SELECT pharmacy_Qty INTO @available

FROM pharmacy_stock

WHERE medicine_name = 'Piliton';

-- If not enough stock, raise error

IF @available < 5 THEN

  SIGNAL SQLSTATE '45000'

  SET MESSAGE_TEXT = 'Not enough stock for Piliton';

END IF;

-- Otherwise, proceed to deduct stock
```

```
UPDATE pharmacy_stock

SET pharmacy_Qty = pharmacy_Qty - 5

WHERE medicine_name = 'Piliton';

COMMIT;
```

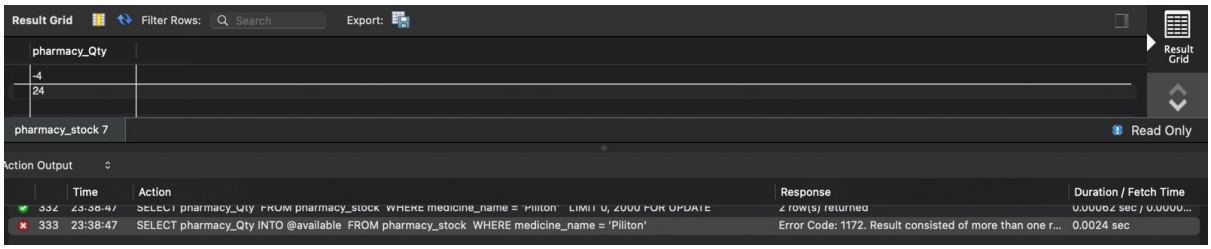


Fig: 6.1

2.

```
use pharmacy_v1;

START TRANSACTION;

-- Lock the row you want to modify

SELECT * FROM pharmacy_stock

WHERE medicine_name = 'Piliton'

FOR UPDATE;

-- Try to update the row only if enough stock is available

UPDATE pharmacy_stock

SET pharmacy_Qty = pharmacy_Qty - 5

WHERE medicine_name = 'Piliton' AND pharmacy_Qty >= 5;

-- Check if the update happened, else raise error

SELECT

   CASE

      WHEN ROW_COUNT() = 0 THEN

         (SELECT RAISE_ERROR())

      ELSE

     'Stock updated successfully'

 END;

COMMIT;
```

Fig: 6.2

3.

```
USE pharmacy_v1;

SELECT * FROM pharmacy_stock WHERE medicine_name = 'Piliton';

START TRANSACTION;

UPDATE pharmacy_stock

SET pharmacy_Qty = pharmacy_Qty - 5

WHERE medicine_name = 'Piliton' AND pharmacy_Qty >= 5;

SELECT * FROM pharmacy_stock WHERE medicine_name = 'Piliton';

COMMIT;

SELECT * FROM pharmacy_stock WHERE medicine_name = 'Piliton';

ROLLBACK;

SELECT * FROM pharmacy_stock WHERE medicine_name = 'Piliton';
```

id	medicine_name	pharmacy_Qty	expiry_date	amount	stock_out	price	capacity	type	sale_date	dosage_sold	dosage	price_dosage	app	half_dosage_pri...
2	Piliton	1	2023-06-15 00:00:00			30	50mg	tablet						0
5	Piliton	34	2023-10-18 00:00:00				60ml	Tablet		Yes	10	100	100	50

Fig: 6.3

CHAPTER 7: RESULT

6A : FRONTEND & RESULT and DISCUSSIONS

(add patient)

```
<!DOCTYPE html>
<html dir="ltr" lang="en">

<?php
 session_start();
require "includes/head.php";
require "includes/auth.php";

?>

<body>
   <!-- Preloader - style you can find in spinners.css -->
   <div class="preloader">
      <div class="lds-ripple">
         <div class="lds-pos"></div>
         <div class="lds-pos"></div>
      </div>
   </div>
   <!-- ============================================================== -->
   <!-- Main wrapper - style you can find in pages.scss -->
   <!-- ============================================================== -->
   <div id="main-wrapper" data-layout="vertical" data-navbarbg="skin5" data-sidebartype="full"
      data-sidebar-position="absolute" data-header-position="absolute" data-boxed-layout="full">
      <!-- Topbar header - style you can find in pages.scss -->
      <?php require "includes/header.php"?>
      <!-- ============================================================== -->
      <!-- End Topbar header -->
      <!-- Left Sidebar - style you can find in sidebar.scss  -->
      <?php require "includes/aside.php";?>
      <!-- ============================================================== -->
      <!-- End Left Sidebar - style you can find in sidebar.scss  -->
      <!-- Page wrapper  -->
      <div class="page-wrapper">
         <!-- Bread crumb and right sidebar toggle -->
         <div class="page-breadcrumb">
            <div class="row align-items-center">
               <div class="col-5">
                  <h4 class="page-title">Add Patients Form</h4>
                  <div class="d-flex align-items-center">
                     <nav aria-label="breadcrumb">
                        <ol class="breadcrumb">
                           <li class="breadcrumb-item"><a href="index.php">Home</a></li>
                           <li class="breadcrumb-item active" aria-current="page">Add Patients Form </li>
                        </ol>
                     </nav>
                  </div>
               </div>

            </div>
         </div>
         <!-- ============================================================== -->
         <!-- End Bread crumb and right sidebar toggle -->
         <!-- ============================================================== -->
         <!-- ============================================================== -->
         <!-- Container fluid  -->
         <!-- ============================================================== -->
         <div class="container-fluid">
           <?php

                                    if(isset($_GET['id'])){
                                             $patient_no = $_GET['id'];
                                    }
                                    ?>
            <div class="col-12">
               <div class="card">
```

```
                <div class="card-body">
                    <h3 class="card-title"> <?php echo $patient_no;?></h3>

                </div>
                <div class="table-responsive">
                <div class="col-lg-12 ">
            <div class="card">
                <div class="card-body">
                    <form class="form-horizontal form-material"
                     action="includes/add_customers_inc.php"  method="post">
                        <div class="form-group">
                            <label for="example-email" class="col-md-12"> <b>Patient Name</b></label>
                            <div class="col-md-12">
                                <input type="text" name="patient_name" placeholder="Enter Patient Name"
                                    class="form-control form-control-line"
                                    id="example-email">
                                                                                                                                                                 <input
type="hidden" name="patient_no" placeholder="Enter Patient Name"
                                    class="form-control form-control-line"
                                    id="example-email"
                                                                                                                                                                 value  =  "<?php
echo $patient_no;?>"
                                                                                                                                                                 >
                                                                                                                                                                 <input

type="hidden" name="status" placeholder="Enter Patient Name"
                                    class="form-control form-control-line"
                                    id="example-email"

                                                                                                                                                                 value = "0"
                                                                                                                                                                 >
                            </div>
                        </div>
                        <div class="form-group">
                            <label for="example-email" class="col-md-12"> <b>Date of Birth</b></label>
                            <div class="col-md-12">
                                <input type="date" name="dob" placeholder="Enter the patient date of birth "
                                    class="form-control form-control-line"
                                    id="example-email">
                            </div>
                        </div><div class="form-group">
                            <label for="example-email" class="col-md-12"> <b>Location</b></label>
                            <div class="col-md-12">
                                <input type="text" name="location" placeholder="Enter the location"
                                    class="form-control form-control-line"
                                    id="example-email">
                            </div>
                        </div>
                                        <div class="form-group">
                                <label class="col-md-12"><b>This is an Emegency</b></label>
                                <div class="col-md-12">
                                    <input type="radio" name="patient_condation" value="Yes"> Yes
                                    <input type="radio" name="patient_condation" value="No"> No
                                </div>
                            </div>
                        <div class="form-group">
                            <div class="col-sm-12">
                                <button class="btn btn-success" type="submit" name="submit">Submit</button>
                            </div>
                        </div>
                    </form>
                </div>
            </div>
        </div>
                </div>
            </div>
        </div>
    </div>
    <!-- ============================================================== -->
    <!-- End Container fluid  -->
    <!-- ============================================================== -->
    <!-- ============================================================== -->
    <!-- footer -->
    <!-- ============================================================== -->
    <footer class="footer text-center">
        All Rights Reserved
    </footer>
```

```
            <!-- ============================================================== -->
            <!-- End footer -->
            <!-- ============================================================== -->
        </div>
        <!-- ============================================================== -->
        <!-- End Page wrapper  -->
        <!-- ============================================================== -->
    </div>
    <!-- ============================================================== -->
    <!-- End Wrapper -->
    <!-- ============================================================== -->
    <!-- ============================================================== -->
    <!-- All Jquery -->
    <!-- ============================================================== -->
    <script src="assets/libs/jquery/dist/jquery.min.js"></script>
    <!-- Bootstrap tether Core JavaScript -->
    <script src="assets/libs/popper.js/dist/umd/popper.min.js"></script>
    <script src="assets/libs/bootstrap/dist/js/bootstrap.min.js"></script>
    <script src="dist/js/app-style-switcher.js"></script>
    <!--Wave Effects -->
    <script src="dist/js/waves.js"></script>
    <!--Menu sidebar -->
    <script src="dist/js/sidebarmenu.js"></script>
    <!--Custom JavaScript -->
    <script src="dist/js/custom.js"></script>
    <!--This page JavaScript -->
    <!--chartis chart-->
    <script src="assets/libs/chartist/dist/chartist.min.js"></script>
    <script src="assets/libs/chartist-plugin-tooltips/dist/chartist-plugin-tooltip.min.js"></script>
    <script src="dist/js/pages/dashboards/dashboard1.js"></script>
</body>
</html>
<!DOCTYPE html>
<html dir="ltr" lang="en">
<?php
 session_start();
require "includes/head.php";
require "includes/auth.php";
?>
<body>
    <!-- ============================================================== -->
    <!-- Preloader - style you can find in spinners.css -->
    <!-- ============================================================== -->
    <div class="preloader">
        <div class="lds-ripple">
            <div class="lds-pos"></div>
            <div class="lds-pos"></div>
        </div>
    </div>
    <!-- ============================================================== -->
    <!-- Main wrapper - style you can find in pages.scss -->
    <!-- ============================================================== -->
    <div id="main-wrapper" data-layout="vertical" data-navbarbg="skin5" data-sidebartype="full"
        data-sidebar-position="absolute" data-header-position="absolute" data-boxed-layout="full">
        <!-- ============================================================== -->
        <!-- Topbar header - style you can find in pages.scss -->
        <!-- ============================================================== -->
        <?php require "includes/header.php"?>
        <!-- ============================================================== -->
        <!-- End Topbar header -->
        <!-- ============================================================== -->
        <!-- ============================================================== -->
        <!-- Left Sidebar - style you can find in sidebar.scss  -->
        <!-- ============================================================== -->
        <?php require "includes/aside.php";?>
        <!-- ============================================================== -->
        <!-- End Left Sidebar - style you can find in sidebar.scss  -->
        <!-- ============================================================== -->
        <!-- ============================================================== -->
        <!-- Page wrapper  -->
        <!-- ============================================================== -->
        <div class="page-wrapper">
            <!-- ============================================================== -->
            <!-- Bread crumb and right sidebar toggle -->
            <!-- ============================================================== -->
```

```
        <div class="page-breadcrumb">
          <div class="row align-items-center">
            <div class="col-5">
              <h4 class="page-title">Add Patients Form</h4>
              <div class="d-flex align-items-center">
                <nav aria-label="breadcrumb">
                  <ol class="breadcrumb">
                    <li class="breadcrumb-item"><a href="index.php">Home</a></li>
                    <li class="breadcrumb-item active" aria-current="page">Add Patients Form </li>
                  </ol>
                </nav>
              </div>
            </div>
          </div>
        </div>
        <!-- ============================================================== -->
        <!-- End Bread crumb and right sidebar toggle -->
        <!-- ============================================================== -->
        <!-- ============================================================== -->
        <!-- Container fluid  -->
        <!-- ============================================================== -->
        <div class="container-fluid">
          <!-- ============================================================== -->
         <?php
                                    if(isset($_GET['id'])){
                                            $patient_no = $_GET['id'];
                                    }
                                    ?>
          <div class="col-12">
            <div class="card">
              <div class="card-body">
                <h3 class="card-title"> <?php echo $patient_no;?></h3>
              </div>
              <div class="table-responsive">
              <div class="col-lg-12 ">
            <div class="card">
              <div class="card-body">
                <form class="form-horizontal form-material"
                 action="includes/add_customers_inc.php"  method="post">
                  <div class="form-group">
                    <label for="example-email" class="col-md-12"> <b>Patient Name</b></label>
                    <div class="col-md-12">
                      <input type="text" name="patient_name" placeholder="Enter Patient Name"
                        class="form-control form-control-line"
                        id="example-email">
                                                                                                <input
type="hidden" name="patient_no" placeholder="Enter Patient Name"
                        class="form-control form-control-line"
                        id="example-email"
                                                                                                value = "<?php
echo $patient_no;?>"
                                                                                                >
                                                                                                <input
type="hidden" name="status" placeholder="Enter Patient Name"
                        class="form-control form-control-line"
                        id="example-email"
                                                                                                value = "0">
                    </div>
                  </div>
<div class="form-group">
                    <label for="example-email" class="col-md-12"> <b>Date of Birth</b></label>
                    <div class="col-md-12">
                      <input type="date" name="dob" placeholder="Enter the patient date of birth "
                        class="form-control form-control-line"
                        id="example-email">
                    </div>
                  </div>
<div class="form-group">
                    <label for="example-email" class="col-md-12"> <b>Location</b></label>
                    <div class="col-md-12">
                      <input type="text" name="location" placeholder="Enter the location"
                        class="form-control form-control-line"
                        id="example-email">
                    </div>
                  </div>
```

```
<div class="form-group">
                            <label class="col-md-12"><b>This is an Emegency</b></label>
                            <div class="col-md-12">
                               <input type="radio" name="patient_condation" value="Yes"> Yes
                               <input type="radio" name="patient_condation" value="No"> No
                            </div>
                         </div>
<div class="form-group">
                            <div class="col-sm-12">
                               <button class="btn btn-success" type="submit" name="submit">Submit</button>
                            </div>
                         </div>
                      </form>
                   </div>
                </div>
             </div>
                   </div>
                </div>
             </div>
</div>
         <!-- ============================================================== -->
         <!-- End Container fluid  -->
         <!-- ============================================================== -->
         <!-- ============================================================== -->
         <!-- footer -->
         <!-- ============================================================== -->
         <footer class="footer text-center">
            All Rights Reserved
         </footer>
         <!-- ============================================================== -->
         <!-- End footer -->
         <!-- ============================================================== -->
      </div>
      <!-- ============================================================== -->
      <!-- End Page wrapper  -->
      <!-- ============================================================== -->
   </div>
   <!-- ============================================================== -->
   <!-- End Wrapper -->
   <!-- ============================================================== -->
   <!-- ============================================================== -->
   <!-- All Jquery -->
   <!-- ============================================================== -->
   <script src="assets/libs/jquery/dist/jquery.min.js"></script>
   <!-- Bootstrap tether Core JavaScript -->
   <script src="assets/libs/popper.js/dist/umd/popper.min.js"></script>
   <script src="assets/libs/bootstrap/dist/js/bootstrap.min.js"></script>
   <script src="dist/js/app-style-switcher.js"></script>
   <!--Wave Effects -->
   <script src="dist/js/waves.js"></script>
   <!--Menu sidebar -->
   <script src="dist/js/sidebarmenu.js"></script>
   <!--Custom JavaScript -->
   <script src="dist/js/custom.js"></script>
   <!--This page JavaScript -->
   <!--chartis chart-->
   <script src="assets/libs/chartist/dist/chartist.min.js"></script>
   <script src="assets/libs/chartist-plugin-tooltips/dist/chartist-plugin-tooltip.min.js"></script>
   <script src="dist/js/pages/dashboards/dashboard1.js"></script>
</body>
</html>
```

6B : Result and Screenshots

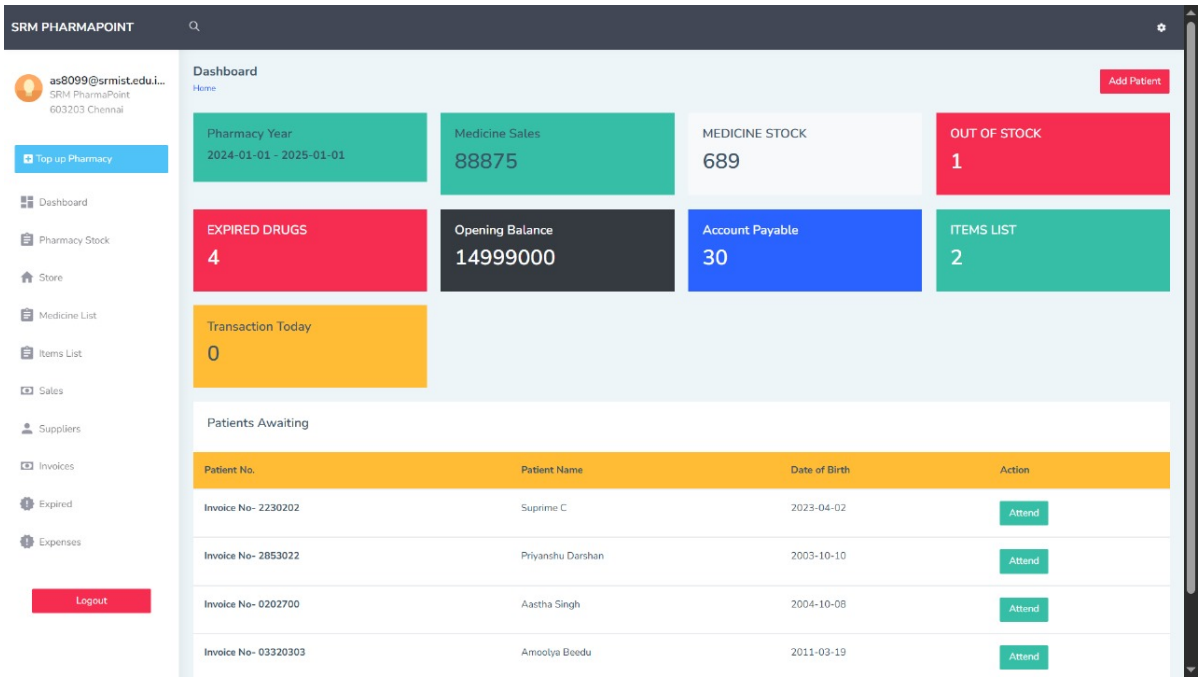


FIG: 7.1

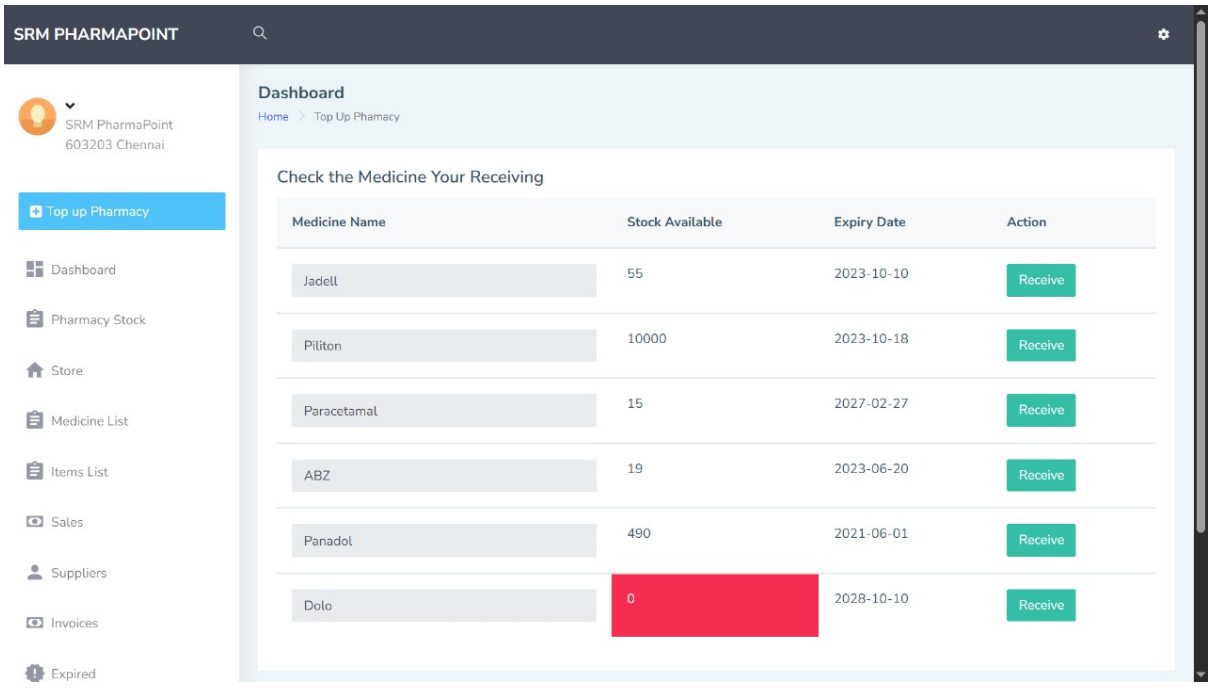


FIG: 7.2

CHAPTER 8: CONCLUSION & FUTURE ENHANCEMENTS

8.1 Conclusion:

The Pharmacy Management System developed provides an efficient and streamlined solution for managing key operations in a pharmacy, including inventory control, sales, purchases, employee login, and customer interactions. By automating these processes, the system reduces manual effort, minimizes errors, and enhances the overall workflow. It ensures accurate tracking of medicines, helps maintain stock levels, and generates essential sales and purchase reports. The role-based login system adds a layer of security, making sure that only authorized users can access sensitive data. Overall, the system contributes significantly to improving the operational efficiency and service quality of a pharmacy.

8.2 Future Enhancements:

1. **Online Order Management:** Allow customers to place medicine orders online and track their order status.
2. **Automatic Stock Alerts:** Implement low-stock and expiry-date notifications to avoid shortages and remove expired drugs.
3. **Mobile App Support:** Develop an Android/iOS version for both customers and employees for ease of access on the go.
4. **GST and Tax Calculation:** Add support for GST and other applicable taxes in billing to comply with regional regulations.
5. **Advanced Analytics:** Integrate dashboards to show trends, best-selling medicines, seasonal demand patterns, etc.
6. **E-Prescription Upload:** Let customers upload doctor prescriptions for validation and medicine dispatch.
7. **Inventory Forecasting:** Use predictive algorithms to suggest restocking quantities based on past sales data.
8. **User Feedback & Rating:** Allow customers to rate services and provide feedback for continuous improvement.
9. **Adding Payment Gateway:** To securely authorize and process online payments between a customer and a Company.

CHAPTER 9: REFERENCES

9.1 Websites and Documentations:

1. **PHP Manual**
   PHP: Hypertext Preprocessor. (n.d.). Official PHP Documentation.
   Retrieved from https://www.php.net/manual/en/
2. **MySQL Documentation**
   MySQL. (n.d.). MySQL 8.0 Reference Manual.
   Retrieved from https://dev.mysql.com/doc/
3. **W3Schools**
   W3Schools Online Web Tutorials. (n.d.). PHP, SQL, HTML, CSS, JavaScript Tutorials.
   Retrieved from https://www.w3schools.com/
4. **Stack Overflow**
   Stack Exchange Inc. (n.d.). Developer Community Forum.
   Retrieved from https://stackoverflow.com/
5. **GeeksforGeeks**
   GeeksforGeeks. (n.d.). PHP and SQL Programming Tutorials.
   Retrieved from https://www.geeksforgeeks.org/
6. **TutorialsPoint**
   TutorialsPoint. (n.d.). Learn Web Development (PHP, MySQL, JS, CSS).
   Retrieved from https://www.tutorialspoint.com/

9.2 Tools and Software Used:

- MySQL Workbench
- Apache Tomcat
- XAMPP
- Visual Studio Code
- MS Word