Axiom: HealthCare Management

A DBMS PROJECT REPORT [DBMS REPORT]

Submitted by

Jayabrata Basu [RA2311003010505]
Rishabh Pradhan [RA2311003010524]
Akshaj Bansal [RA2311003010529]

Under the Guidance of

Dr. Snehasish Ghosh

(Assistant Professor, Department of Computing Technologies)

in partial fulfillment of the requirementsfor 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 be signed and dated along with your student registration number and included with all assignments you submit – work will not be marked unless this is done.

<u>To be completed by the student for all assessments</u>

Degree/ Course	:	**B.Tech CSE Core**
Student Name	:	**Jayabrata Basu, Rishabh Pradhan, Akshaj Bansal**
Registration Number	:	**RA2311003010505, RA2311003010524, RA2311003010529**
Title of Work	:	**Axiom: Healthcare Management**

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 for every student in your group.

SRM INSTITUTE OF SCIENCE AND TECHNOLOGY
KATTANKULATHUR – 603 203

BONAFIDE CERTIFICATE

Certified that 21CSC205P – Database Management System Project report titled "Axiom: **HealthCare Management**" is the bonafide work of "**Akshaj Bansal [RA2311003010529], Rishabh Pradhan [RA2311003010524], Jayabrata Basu [RA2311003010505],"** 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	**SIGNATURE**
Dr. Snehasish Ghosh	**Dr. G. NIRANJANA**
ASSISTANT PROFESSOR DEPARTMENT OF COMPUTING TECHNOLOGIES	**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. Snehasish Ghosh**, 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. Snehasish Ghosh**, 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.

Jayabrata Basu	Rishabh Pradhan	Akshaj Bansal
[RA2311003010505]	[RA2311003010524]	[RA231100301052

ABSTRACT

The Healthcare Management System is a database-driven project designed to manage and organize essential healthcare information for patients and medical professionals. The system stores user data such as medical records, prescriptions, doctor details, insurance information, and triage reports in a structured and relational format. It ensures that data is well-organized, easily accessible, and maintained with high integrity and security standards.

The database uses SQL and leverages advanced concepts including constraints, joins, views, triggers, and cursors. These components help enforce rules, connect multiple tables efficiently, automate routine tasks, and handle complex queries for real-time data retrieval and monitoring. Features such as active medication reminders, expiring insurance tracking, and severity-based triage logging provide users with a practical and intelligent way to manage their healthcare needs.

This project aims to improve the efficiency of healthcare data management while supporting better communication between patients and healthcare providers. By centralizing critical information and enabling automated decision support, the system contributes to faster response times, improved medical record accuracy, and enhanced patient care delivery.

TABLE OF CONTENTS

1.1 Problem Statement

In the current healthcare ecosystem, one of the most pressing challenges is the fragmentation of patient data. Patients often receive treatments from multiple providers, resulting in disjointed medical records spread across hospitals, clinics, and paper documents. This lack of centralized information can lead to delays in diagnosis, medication errors, redundant tests, and poor continuity of care. For patients with chronic conditions or allergies, the risk is even higher as missing or inaccessible information can directly affect treatment outcomes.

Additionally, patients often struggle to keep track of their active medications, insurance details, and upcoming renewals. Manual reminders and paper-based insurance policies can lead to lapses in coverage or missed doses, which could worsen health conditions. There's also a significant gap in how patients and healthcare professionals communicate and manage medical data. Without a user-friendly, digital platform, patients are left relying on memory or scattered documents to share critical health history, which can hinder accurate triage and emergency care.

On the administrative side, hospitals and clinics face difficulty maintaining updated doctor information, appointment schedules, and patient medical histories. The lack of real-time access to triage data and prescription history results in inefficiencies, especially during high-patient-load situations. Moreover, insurance providers and hospital billing systems often operate separately, leading to bottlenecks in claim processing. There is a clear need for an integrated healthcare management solution that securely stores, retrieves, and manages patient data, doctor information, medications, triage records, and insurance — all in one unified platform.

1.2 SDG Goals

Our project aligns with SDG Goal 3: Good Health and Well-being. The project aims to address accessibility in remote areas and hopes to provide a simple but effective solution that provides help in terms of accessing and managing information such as insurance and appointments along with providing triage for simple and common maladies to reduce the chance of discord and fear.

2.1 ER Diagram

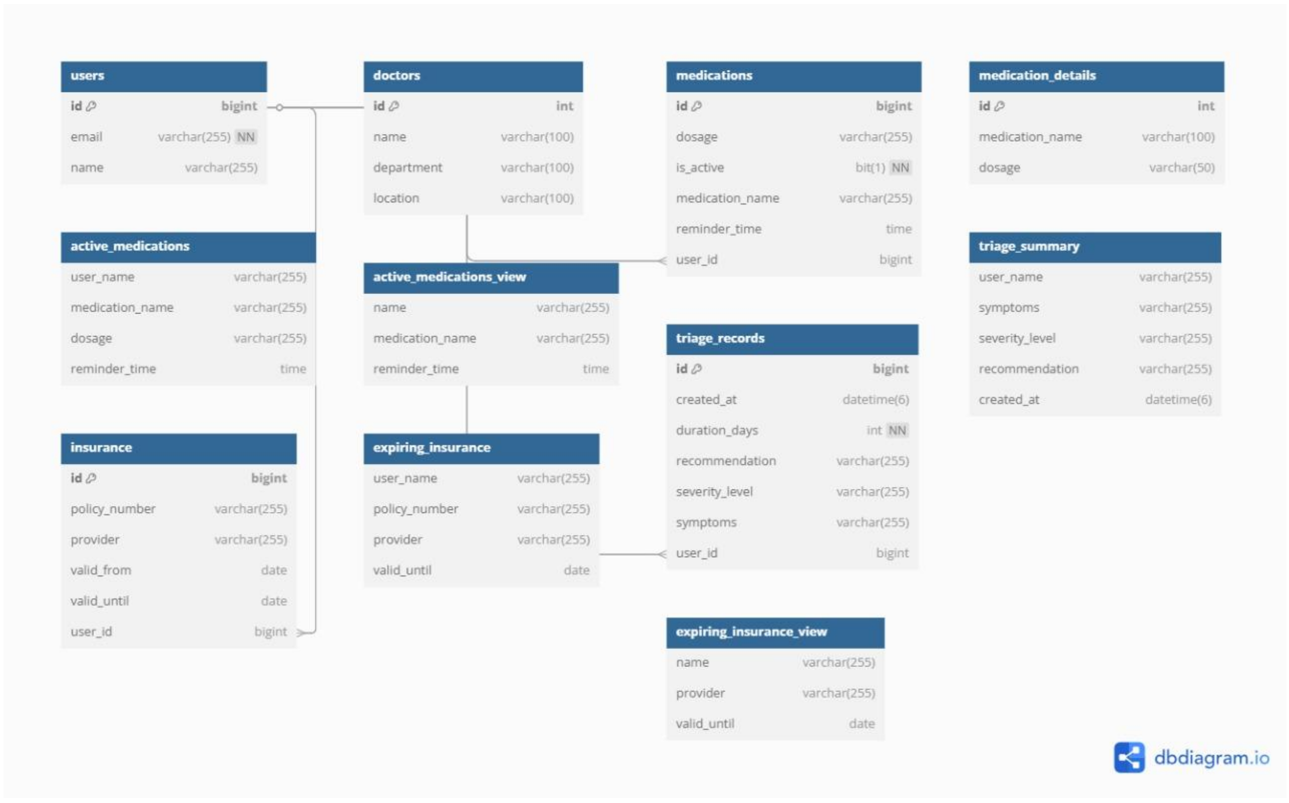


Fig 1. Entity Relationship Diagram (ER Diagram)

2.2 Attribute Description

<u>Entities</u>

User

Attributes: user_id (PK), email, name

Doctor

Attributes: doctor_id (PK), name, department, location

Medication

Attributes: medication_id (PK), medication_name, dosage, reminder_time, is_active, user_id (FK)

MedicationDetails

Attributes: detail_id (PK), medication_name, dosage

Insurance

Attributes: insurance_id (PK), policy_number, provider, valid_from, valid_until, user_id (FK)

TriageRecord

Attributes: triage_id (PK), created_at, duration_days, symptoms, severity_level, recommendation, user_id (FK)

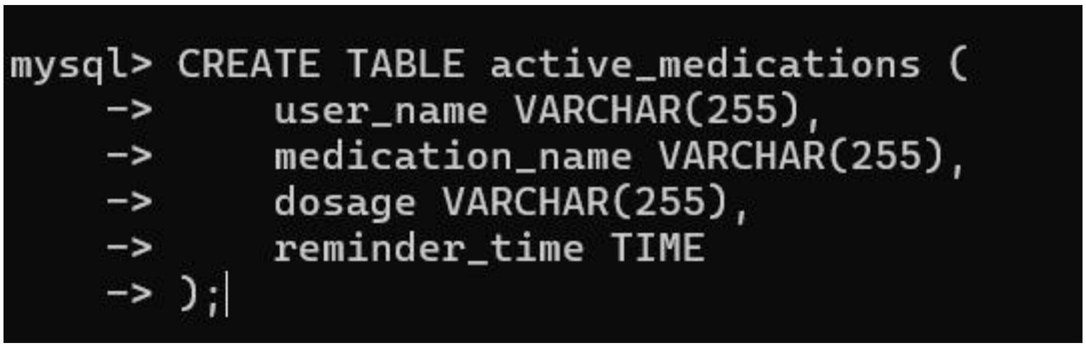
Relationships

- The **User–Medication** relationship is **one-to-many**: one User can have multiple Medications, but each Medication belongs to exactly one User.
- The **User–Insurance** relationship is **one-to-many**: one User can register multiple Insurance policies, but each policy is linked to only one User.

- The **User–TriageRecord** relationship is **one-to-many**: a User can have multiple Triage Records stored in the system, each capturing a separate self-assessment.
- The **Medication–MedicationDetails** relationship is **many-to-one**: each Medication
- refers to standard medication information from the MedicationDetails entity, matching via the medication_name.
- The **Doctor–TriageRecord** relationship can optionally be **many-to-one**: if implemented, each Triage Record may be reviewed or recommended by one Doctor, and a Doctor may review multiple Triage Records.
- The **ActiveMedication** and **ExpiringInsurance** are **view entities**, created to simplify querying and do not store data directly but reflect filtered information based on conditions from the main tables.

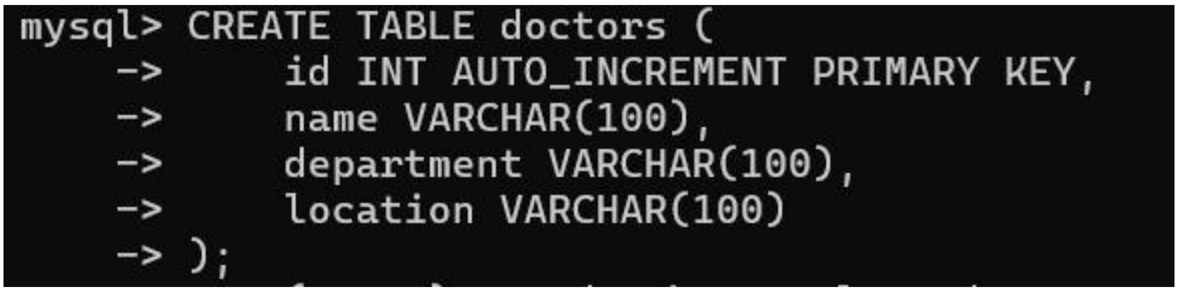
2.3 SQL Queries for table creation

Table: active_medications



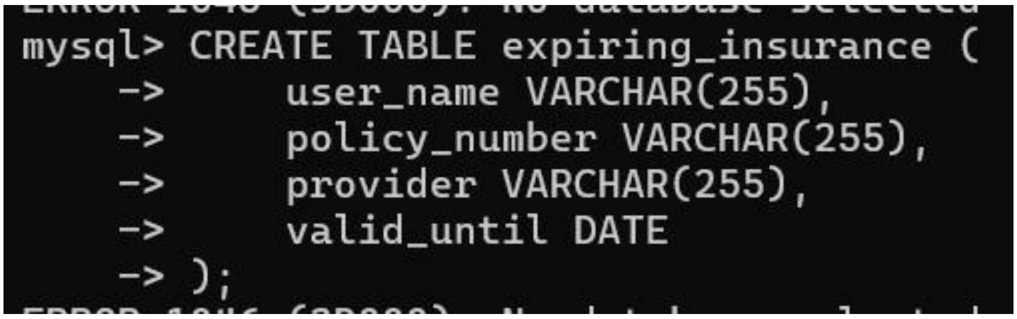
```
mysql> CREATE TABLE active_medications (
    ->     user_name VARCHAR(255),
    ->     medication_name VARCHAR(255),
    ->     dosage VARCHAR(255),
    ->     reminder_time TIME
    -> );
```

Table: doctors



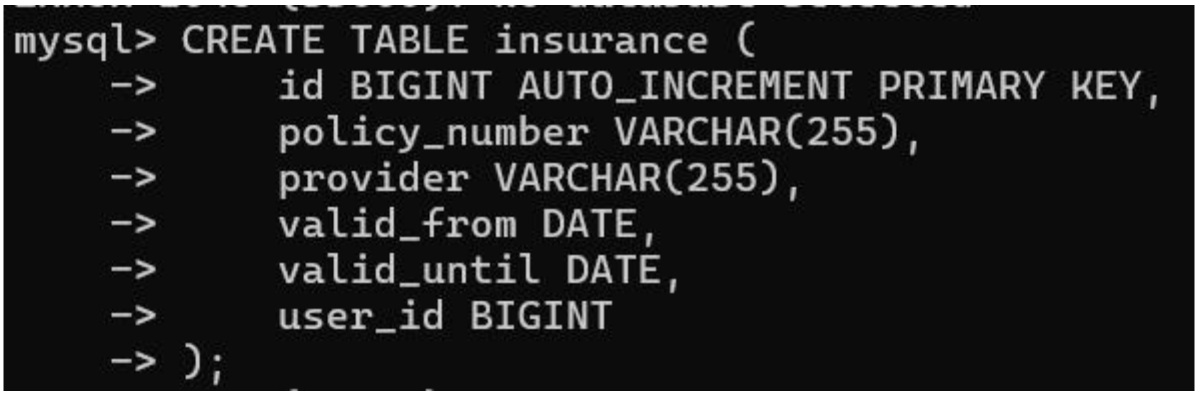
```
mysql> CREATE TABLE doctors (
    ->     id INT AUTO_INCREMENT PRIMARY KEY,
    ->     name VARCHAR(100),
    ->     department VARCHAR(100),
    ->     location VARCHAR(100)
    -> );
```

Table: expiring_insurance



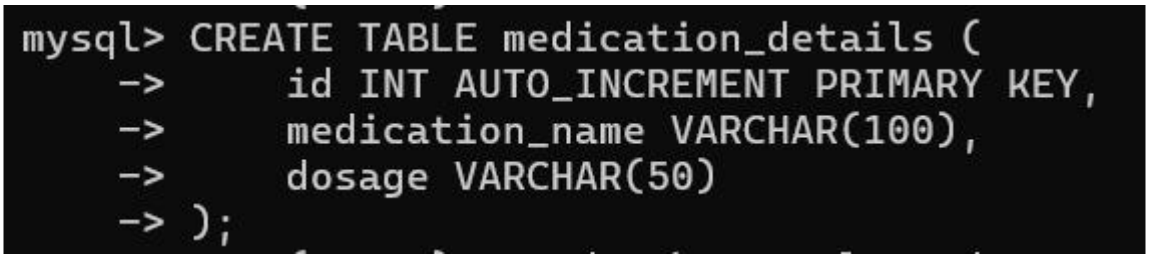
```
mysql> CREATE TABLE expiring_insurance (
    ->     user_name VARCHAR(255),
    ->     policy_number VARCHAR(255),
    ->     provider VARCHAR(255),
    ->     valid_until DATE
    -> );
```

Table: insurance



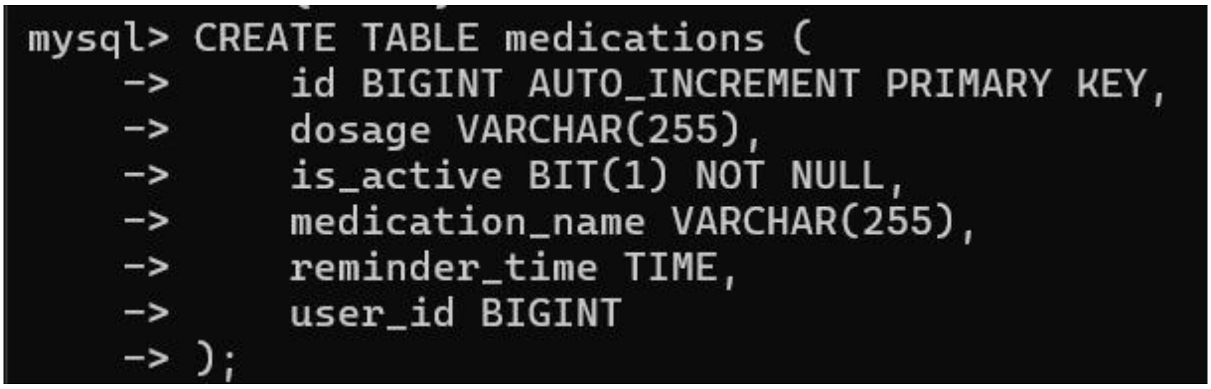
```
mysql> CREATE TABLE insurance (
    ->     id BIGINT AUTO_INCREMENT PRIMARY KEY,
    ->     policy_number VARCHAR(255),
    ->     provider VARCHAR(255),
    ->     valid_from DATE,
    ->     valid_until DATE,
    ->     user_id BIGINT
    -> );
```

Table: medication_details



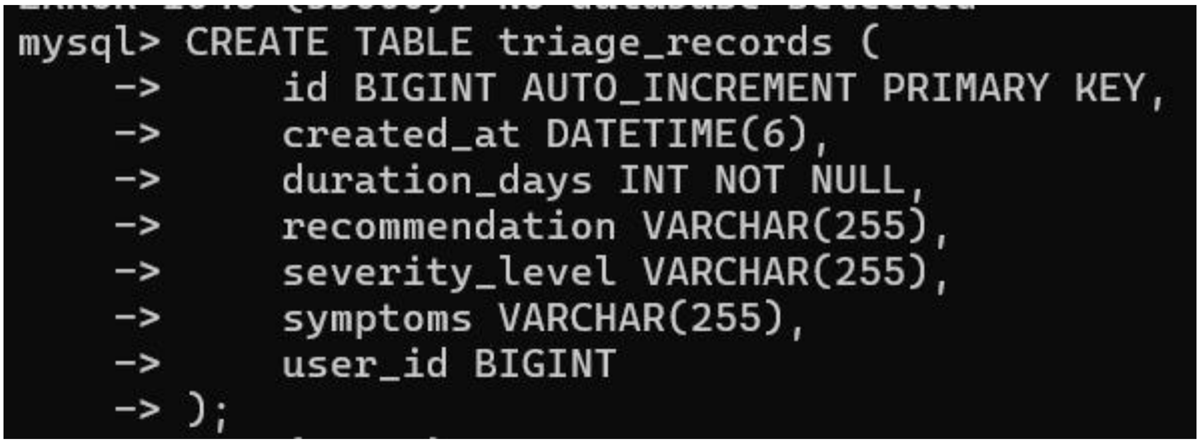
```
mysql> CREATE TABLE medication_details (
    ->     id INT AUTO_INCREMENT PRIMARY KEY,
    ->     medication_name VARCHAR(100),
    ->     dosage VARCHAR(50)
    -> );
```

Table: medications



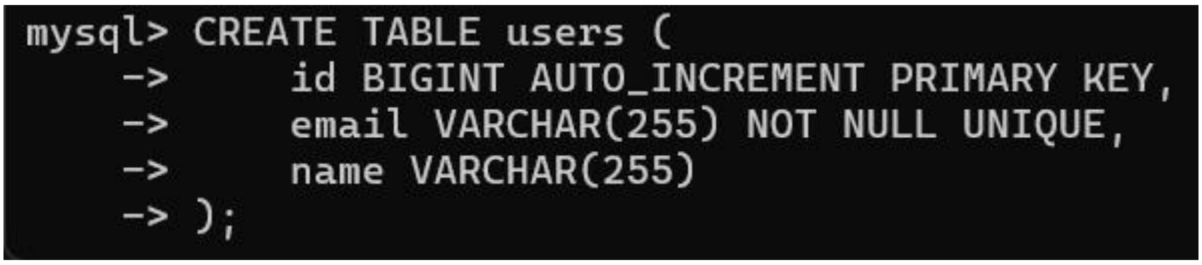
```
mysql> CREATE TABLE medications (
    ->     id BIGINT AUTO_INCREMENT PRIMARY KEY,
    ->     dosage VARCHAR(255),
    ->     is_active BIT(1) NOT NULL,
    ->     medication_name VARCHAR(255),
    ->     reminder_time TIME,
    ->     user_id BIGINT
    -> );
```

Table: triage_records



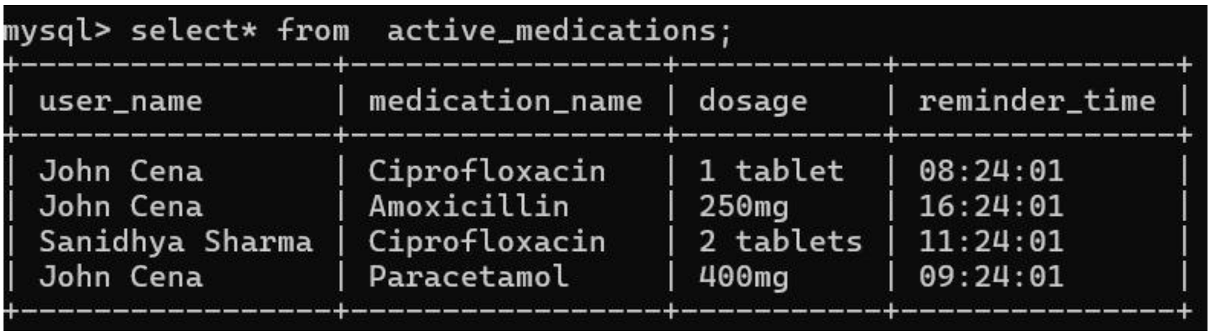
```
mysql> CREATE TABLE triage_records (
    ->     id BIGINT AUTO_INCREMENT PRIMARY KEY,
    ->     created_at DATETIME(6),
    ->     duration_days INT NOT NULL,
    ->     recommendation VARCHAR(255),
    ->     severity_level VARCHAR(255),
    ->     symptoms VARCHAR(255),
    ->     user_id BIGINT
    -> );
```

Table: users

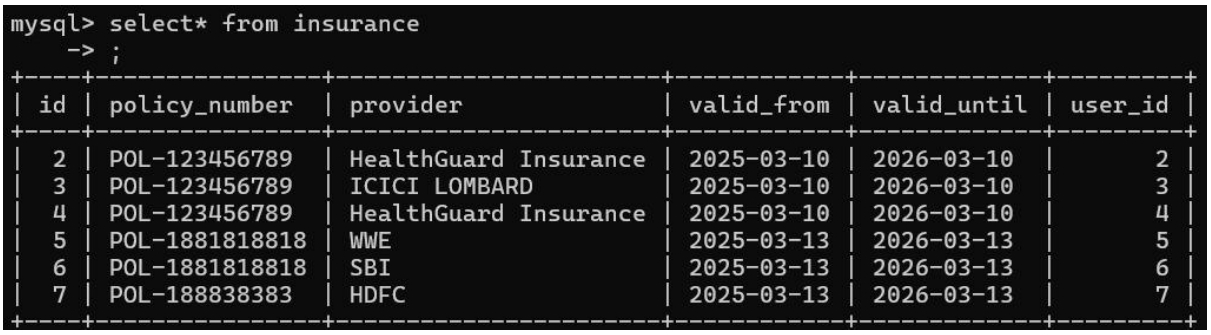


```
mysql> CREATE TABLE users (
    ->     id BIGINT AUTO_INCREMENT PRIMARY KEY,
    ->     email VARCHAR(255) NOT NULL UNIQUE,
    ->     name VARCHAR(255)
    -> );
```

2.4 Tables with Data



```
mysql> select* from  active_medications;
+-----------------+-----------------+-----------+---------------+
| user_name       | medication_name | dosage    | reminder_time |
+-----------------+-----------------+-----------+---------------+
| John Cena       | Ciprofloxacin   | 1 tablet  | 08:24:01      |
| John Cena       | Amoxicillin     | 250mg     | 16:24:01      |
| Sanidhya Sharma | Ciprofloxacin   | 2 tablets | 11:24:01      |
| John Cena       | Paracetamol     | 400mg     | 09:24:01      |
+-----------------+-----------------+-----------+---------------+
```



```
mysql> select* from insurance
    -> ;
+----+----------------+-----------------------+------------+-------------+---------+
| id | policy_number  | provider              | valid_from | valid_until | user_id |
+----+----------------+-----------------------+------------+-------------+---------+
|  2 | POL-123456789  | HealthGuard Insurance | 2025-03-10 | 2026-03-10  |       2 |
|  3 | POL-123456789  | ICICI LOMBARD         | 2025-03-10 | 2026-03-10  |       3 |
|  4 | POL-123456789  | HealthGuard Insurance | 2025-03-10 | 2026-03-10  |       4 |
|  5 | POL-1881818818 | WWE                   | 2025-03-13 | 2026-03-13  |       5 |
|  6 | POL-1881818818 | SBI                   | 2025-03-13 | 2026-03-13  |       6 |
|  7 | POL-188838383  | HDFC                  | 2025-03-13 | 2026-03-13  |       7 |
+----+----------------+-----------------------+------------+-------------+---------+
```

3.1 Constraints

Constraints were used to ensure the integrity of the data in the system. For example:

- PRIMARY KEY: Ensured that each record in the tables, such as the Users table and the Doctors table, has a unique identifier.
- FOREIGN KEY: Enforced relationships between tables, such as ensuring that each medical record is linked to a valid user and doctor.
- CHECK: Used to validate values, like ensuring a patient's age is above a certain threshold or that insurance details are correctly formatted.

Sets:
The use of set operations like UNION and INTERSECT was essential for combining results from multiple tables. For example, the union of patients and doctors involved in a specific type of insurance policy could be achieved by combining data from the respective tables using the UNION operator.

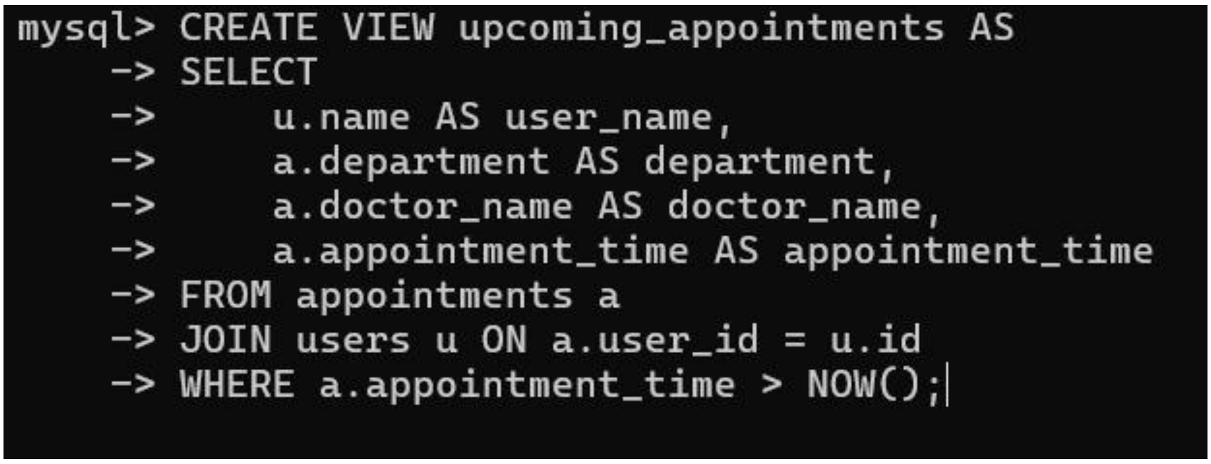
Joins:
Several types of joins were utilized to fetch data from multiple tables in a normalized database. These included:

- INNER JOIN: To fetch records where there is a matching record in both tables, such as retrieving medical records with corresponding doctor details.
- LEFT JOIN: Used to display all users, even if they do not have corresponding insurance records.
- RIGHT JOIN: For cases where doctors may not have scheduled appointments, but their information still needs to be displayed.
- SELF JOIN: Used to find relationships between records in the same table, such as querying patients and their referring doctors.

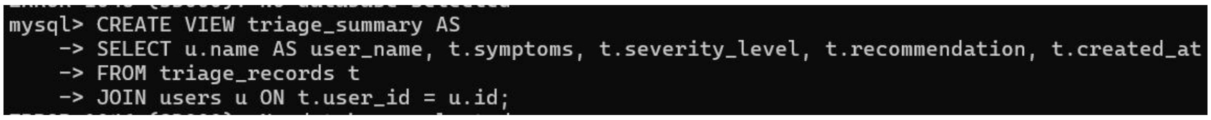
3.2 Views

Views were created to simplify complex queries and present data in a more user-friendly format. For example:

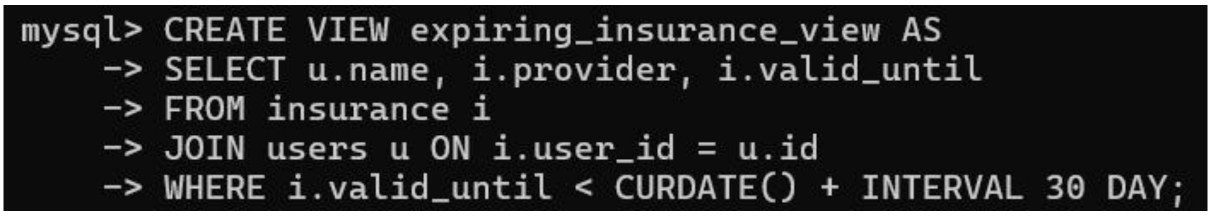
- A view combining patient details with their medical records and insurance status was created to provide a comprehensive view of a patient's history with just a single query, enhancing efficiency and readability for healthcare professionals.



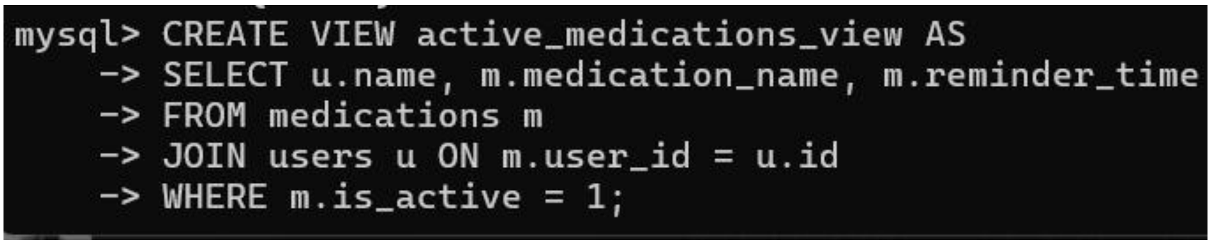
```
mysql> CREATE VIEW upcoming_appointments AS
    -> SELECT
    ->     u.name AS user_name,
    ->     a.department AS department,
    ->     a.doctor_name AS doctor_name,
    ->     a.appointment_time AS appointment_time
    -> FROM appointments a
    -> JOIN users u ON a.user_id = u.id
    -> WHERE a.appointment_time > NOW();
```



```
mysql> CREATE VIEW triage_summary AS
    -> SELECT u.name AS user_name, t.symptoms, t.severity_level, t.recommendation, t.created_at
    -> FROM triage_records t
    -> JOIN users u ON t.user_id = u.id;
```



```
mysql> CREATE VIEW expiring_insurance_view AS
    -> SELECT u.name, i.provider, i.valid_until
    -> FROM insurance i
    -> JOIN users u ON i.user_id = u.id
    -> WHERE i.valid_until < CURDATE() + INTERVAL 30 DAY;
```



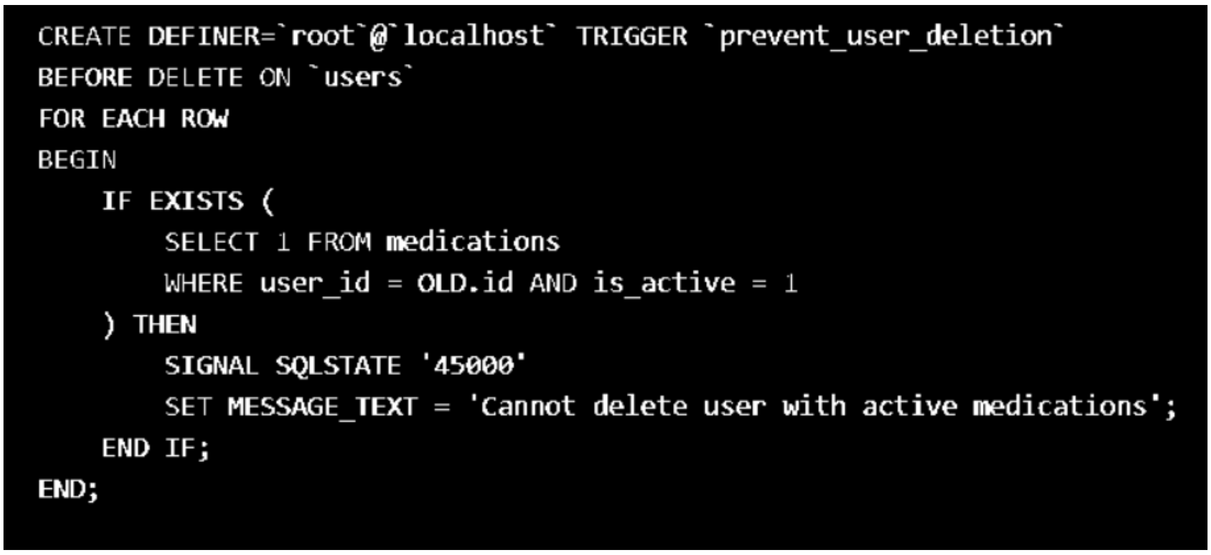
```
mysql> CREATE VIEW active_medications_view AS
    -> SELECT u.name, m.medication_name, m.reminder_time
    -> FROM medications m
    -> JOIN users u ON m.user_id = u.id
    -> WHERE m.is_active = 1;
```

3.3 Triggers

Triggers:

Triggers were implemented to automate certain actions based on specific events in the database. For example:

- A trigger was set to automatically update the Patient_Status field whenever a new triage record was inserted into the Triage_Records table.
- Another trigger was used to log any changes in insurance details, maintaining an audit trail for data integrity and security.

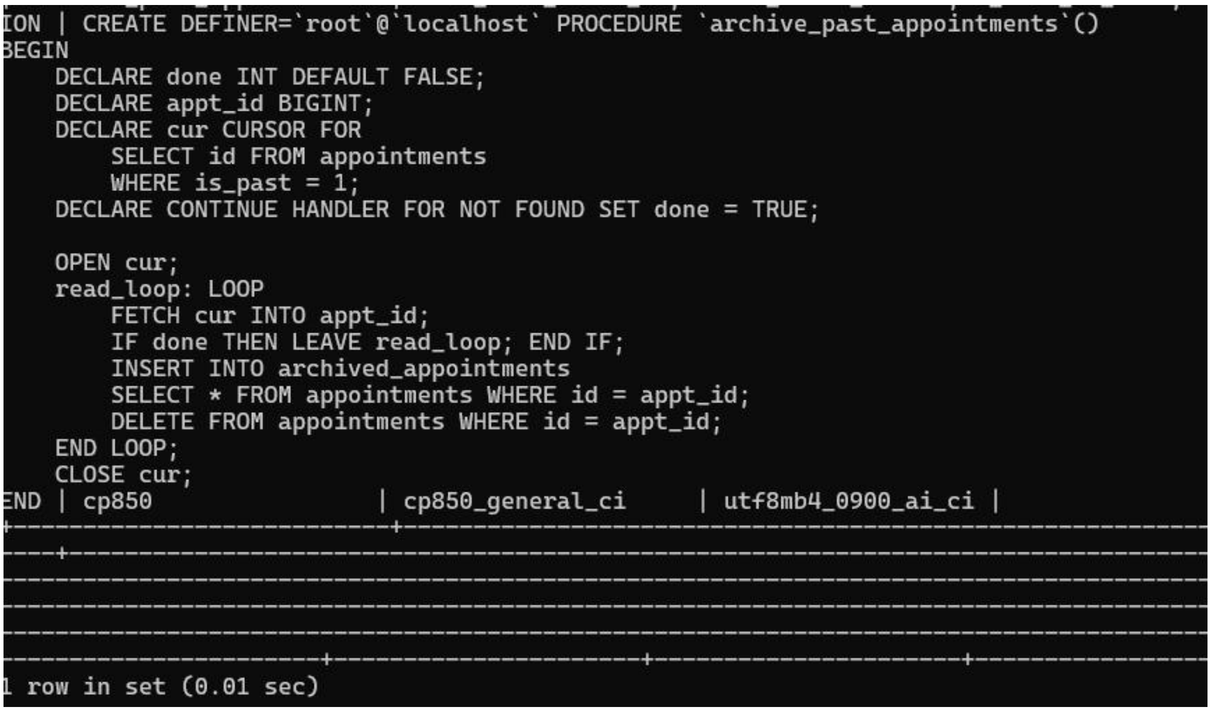


```
CREATE DEFINER=`root`@`localhost` TRIGGER `prevent_user_deletion`
BEFORE DELETE ON `users`
FOR EACH ROW
BEGIN
    IF EXISTS (
        SELECT 1 FROM medications
        WHERE user_id = OLD.id AND is_active = 1
    ) THEN
        SIGNAL SQLSTATE '45000'
        SET MESSAGE_TEXT = 'Cannot delete user with active medications';
    END IF;
END;
```

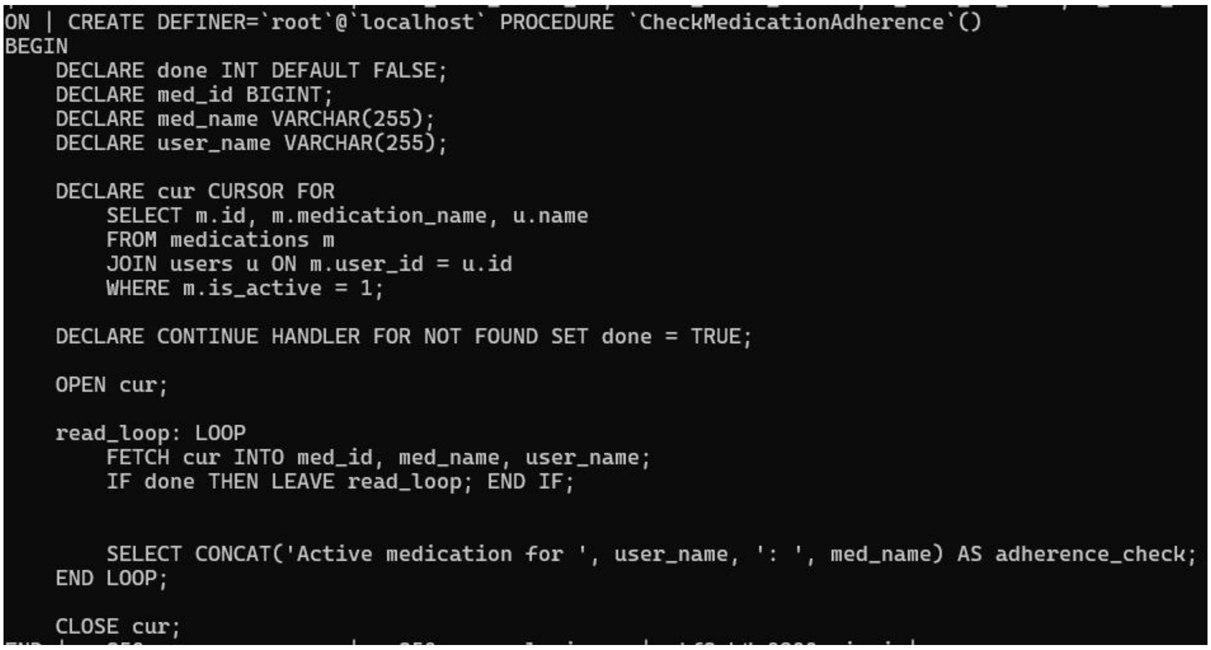
3.4 Cursors

Cursors:

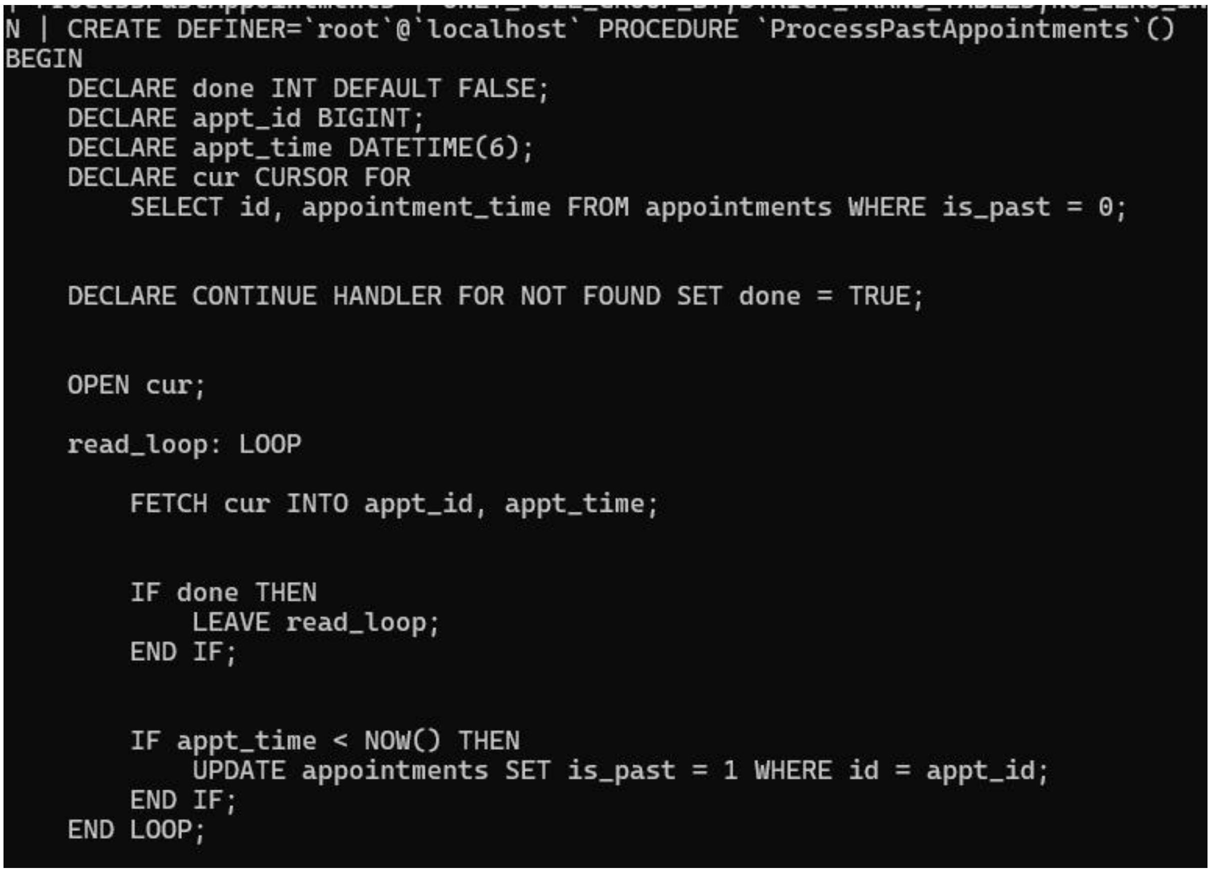
Cursors were employed to process multiple rows of data row by row. This was especially useful for operations that required sequential processing, such as updating a list of appointments or processing bulk patient data in a specific sequence for reports.



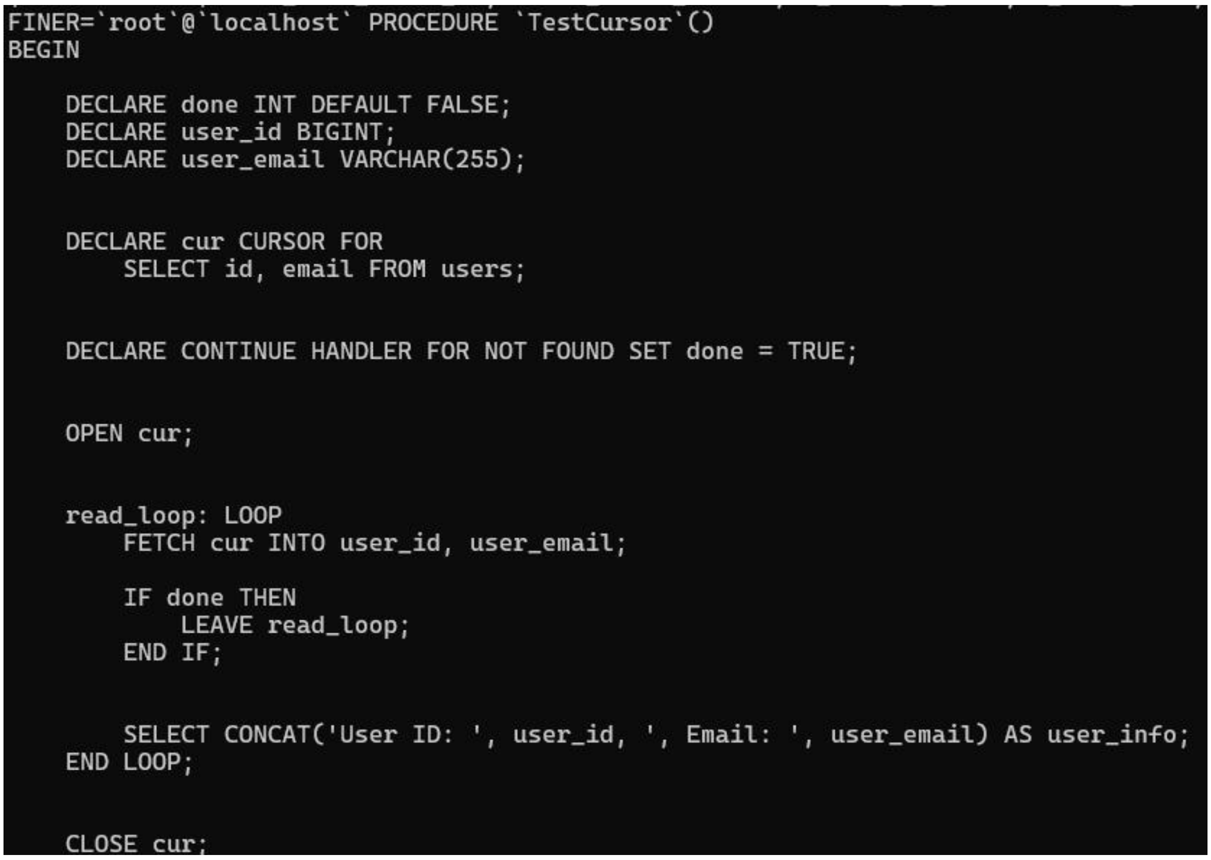
```
ION | CREATE DEFINER=`root`@`localhost` PROCEDURE `archive_past_appointments`()
BEGIN
    DECLARE done INT DEFAULT FALSE;
    DECLARE appt_id BIGINT;
    DECLARE cur CURSOR FOR
        SELECT id FROM appointments
        WHERE is_past = 1;
    DECLARE CONTINUE HANDLER FOR NOT FOUND SET done = TRUE;

    OPEN cur;
    read_loop: LOOP
        FETCH cur INTO appt_id;
        IF done THEN LEAVE read_loop; END IF;
        INSERT INTO archived_appointments
        SELECT * FROM appointments WHERE id = appt_id;
        DELETE FROM appointments WHERE id = appt_id;
    END LOOP;
    CLOSE cur;
END | cp850                 | cp850_general_ci     | utf8mb4_0900_ai_ci |
+---------------------------+------------------------------------------------
----+---------------------------------------------------------------------------
--------------------------------------------------------------------------------
--------------------------------------------------------------------------------
--------------------------------------------------------------------------------
-----------------------+----------------------+----------------------+----------
1 row in set (0.01 sec)
```



```
ON | CREATE DEFINER=`root`@`localhost` PROCEDURE `CheckMedicationAdherence`()
BEGIN
    DECLARE done INT DEFAULT FALSE;
    DECLARE med_id BIGINT;
    DECLARE med_name VARCHAR(255);
    DECLARE user_name VARCHAR(255);

    DECLARE cur CURSOR FOR
        SELECT m.id, m.medication_name, u.name
        FROM medications m
        JOIN users u ON m.user_id = u.id
        WHERE m.is_active = 1;

    DECLARE CONTINUE HANDLER FOR NOT FOUND SET done = TRUE;

    OPEN cur;

    read_loop: LOOP
        FETCH cur INTO med_id, med_name, user_name;
        IF done THEN LEAVE read_loop; END IF;


        SELECT CONCAT('Active medication for ', user_name, ': ', med_name) AS adherence_check;
    END LOOP;

    CLOSE cur;
```



```
N | CREATE DEFINER=`root`@`localhost` PROCEDURE `ProcessPastAppointments`()
BEGIN
    DECLARE done INT DEFAULT FALSE;
    DECLARE appt_id BIGINT;
    DECLARE appt_time DATETIME(6);
    DECLARE cur CURSOR FOR
        SELECT id, appointment_time FROM appointments WHERE is_past = 0;


    DECLARE CONTINUE HANDLER FOR NOT FOUND SET done = TRUE;


    OPEN cur;

    read_loop: LOOP

        FETCH cur INTO appt_id, appt_time;


        IF done THEN
            LEAVE read_loop;
        END IF;


        IF appt_time < NOW() THEN
            UPDATE appointments SET is_past = 1 WHERE id = appt_id;
        END IF;
    END LOOP;
```



```
FINER=`root`@`localhost` PROCEDURE `TestCursor`()
BEGIN

    DECLARE done INT DEFAULT FALSE;
    DECLARE user_id BIGINT;
    DECLARE user_email VARCHAR(255);


    DECLARE cur CURSOR FOR
        SELECT id, email FROM users;


    DECLARE CONTINUE HANDLER FOR NOT FOUND SET done = TRUE;


    OPEN cur;


    read_loop: LOOP
        FETCH cur INTO user_id, user_email;

        IF done THEN
            LEAVE read_loop;
        END IF;


        SELECT CONCAT('User ID: ', user_id, ', Email: ', user_email) AS user_info;
    END LOOP;


    CLOSE cur;
```

4.1 Pitfalls

Pitfalls Resolved

Unique Constraint on Email

The users table includes a unique constraint on the email column. This ensures that each user in the system has a distinct email address, thereby eliminating duplication and supporting secure authentication and user identification.

Foreign Key Integrity

Foreign key relationships—such as user_id in the medications, insurance, and triage_records tables—are properly enforced with referential integrity. Cascading actions (ON DELETE CASCADE, ON UPDATE CASCADE) are applied to maintain consistency across related tables. This ensures that when a user record is updated or deleted, the related data in associated tables is also appropriately handled, preventing orphaned records and maintaining data integrity.

Handling of NULLs

All critical fields such as email in the users table have been cleaned of NULL values before enforcing the NOT NULL constraint. This ensures that all records contain complete information and improves the robustness and reliability of operations like email notifications, password resets, and login processes.

Primary Keys and Indexing

Each table contains a well-defined primary key, such as id in the users, medications, and appointments tables. These primary keys ensure unique identification of records. Foreign key columns like user_id are also indexed to improve query performance, especially during join operations and lookups in a relational context.

4.2 Normalization

Normalization Status

First Normal Form (1NF)

Definition: A table is in 1NF if it contains only atomic values and each column contains values of a single type.

Status: All tables in the design satisfy 1NF. Each field holds atomic data (e.g., names, dosages, contact numbers), and there are no repeating groups or arrays in any column.

Second Normal Form (2NF)

Definition: A table is in 2NF if it is in 1NF and all non-key attributes are fully functionally dependent on the entire primary key.

Status: The schema adheres to 2NF. There are no composite keys currently, but in all tables, the non-key attributes are dependent solely on the primary key of their respective table. For example, in the appointments table, attributes like appointment_time, doctor_id, and user_id are all fully dependent on the appointment_id.

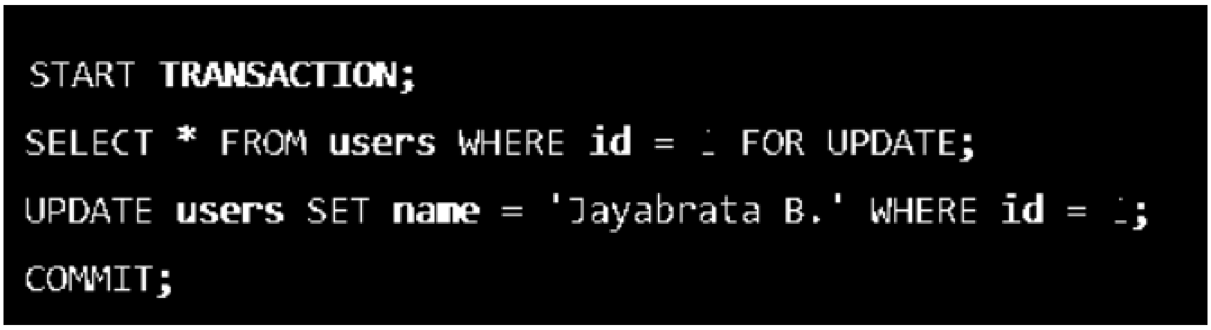
Third Normal Form (3NF)

Definition: A table is in 3NF if it is in 2NF and there are no transitive dependencies among non-key attributes.

Status: The schema conforms to 3NF. There are no attributes that depend on other non-key attributes. Foreign key relationships such as doctor_id in appointments and user_id in medications link only to their corresponding entity tables, and no attribute in a table is transitively dependent on another non-key field within the same table.

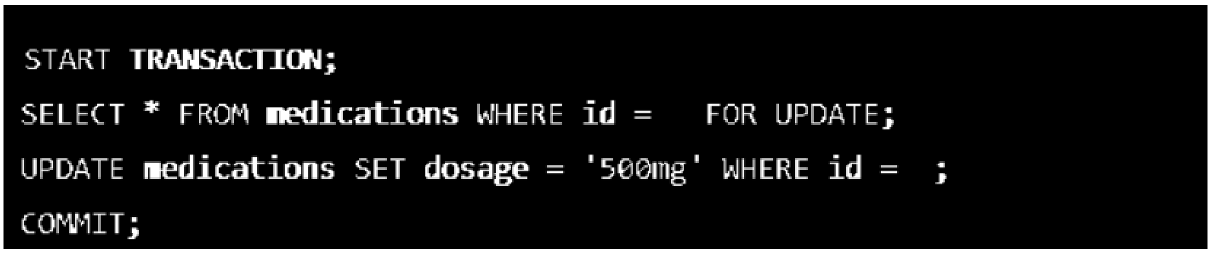
5.1 Concurrency Control & Recovery Mechanisms

Lock and Safely Update a User Row



```
START TRANSACTION;
SELECT * FROM users WHERE id = 1 FOR UPDATE;
UPDATE users SET name = 'Jayabrata B.' WHERE id = 1;
COMMIT;
```

Update Medication Dosage with Lock



```
START TRANSACTION;
SELECT * FROM medications WHERE id =   FOR UPDATE;
UPDATE medications SET dosage = '500mg' WHERE id =  ;
COMMIT;
```

Use Serializable Isolation to Prevent Anomalies

SET TRANSACTION ISOLATION LEVEL SERIALIZABLE;

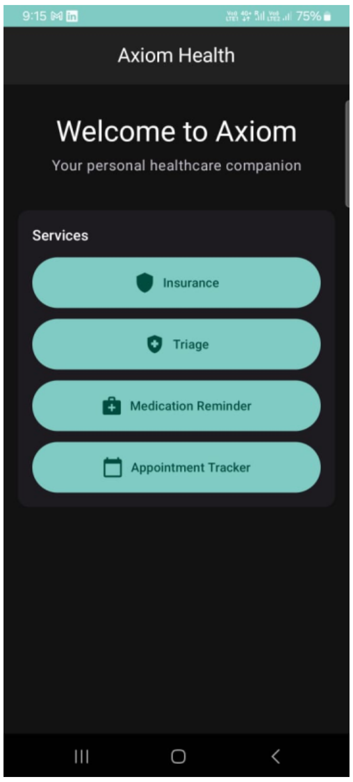
START TRANSACTION;

SELECT * FROM insurance WHERE user_id = 1 FOR UPDATE;

UPDATE insurance SET expiry_date = DATE_ADD(expiry_date, INTERVAL 1 YEAR) WHERE user_id = 1;

COMMIT;

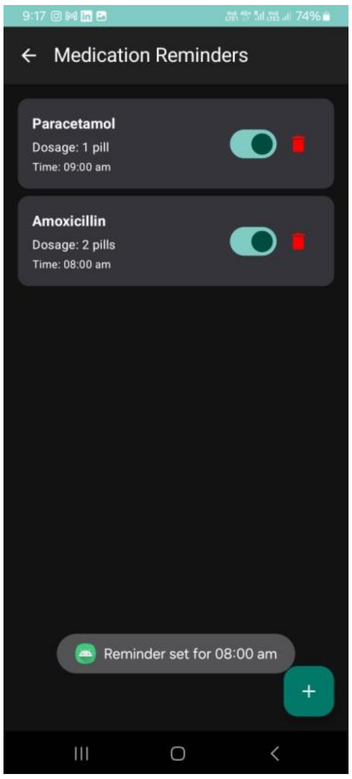
5.2 Frontend



Welcome to Axiom

Your personal healthcare companion

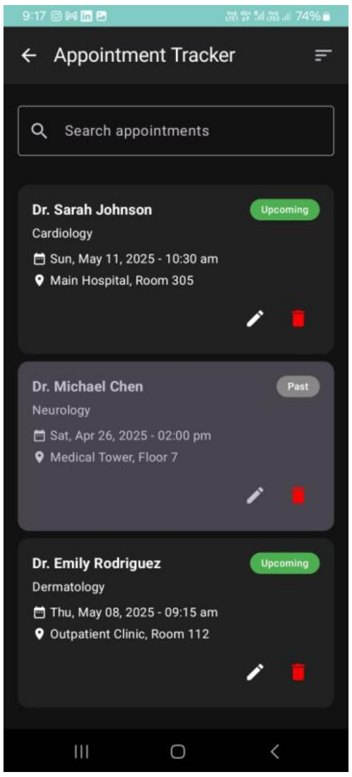
Services



Paracetamol
Dosage: 1 pill
Time: 09:00 am

Amoxicillin
Dosage: 2 pills
Time: 08:00 am

Reminder set for 08:00 am

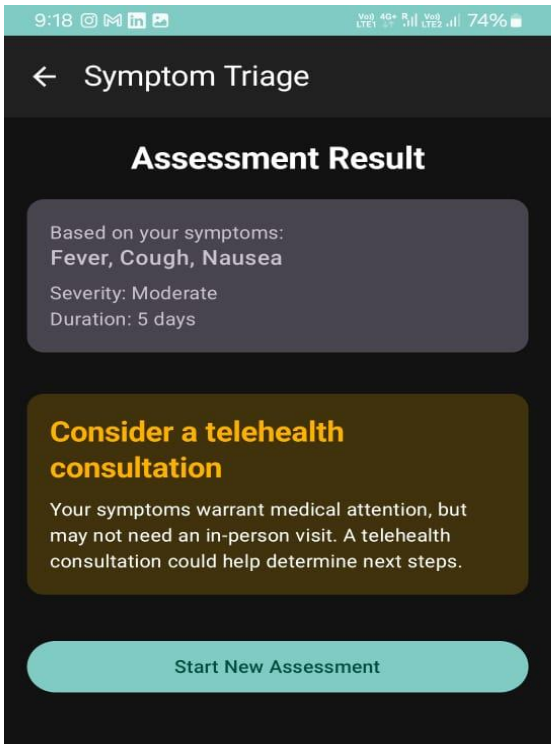


Dr. Sarah Johnson — Upcoming
Cardiology
Sun, May 11, 2025 - 10:30 am
Main Hospital, Room 305

Dr. Michael Chen — Past
Neurology
Sat, Apr 26, 2025 - 02:00 pm
Medical Tower, Floor 7

Dr. Emily Rodriguez — Upcoming
Dermatology
Thu, May 08, 2025 - 09:15 am
Outpatient Clinic, Room 112

Fig 2. Home Screen Fig 3. Medication Reminder Fig 4. Appointment Tracker



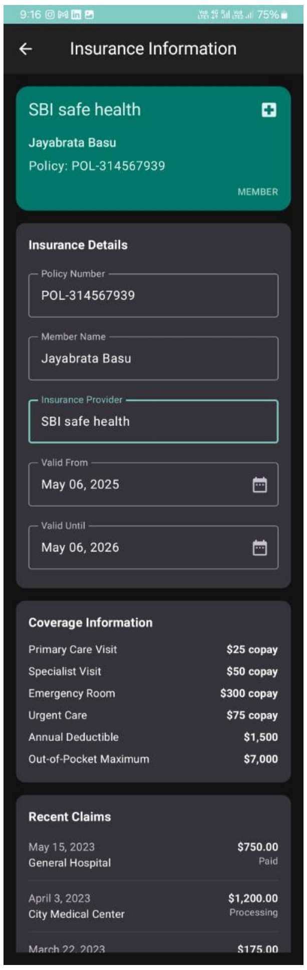
Assessment Result

Based on your symptoms:
Fever, Cough, Nausea
Severity: Moderate
Duration: 5 days

Consider a telehealth consultation

Your symptoms warrant medical attention, but may not need an in-person visit. A telehealth consultation could help determine next steps.

Fig 5. Triage Screen


9:16 75%

Insurance Information

SBI safe health

Jayabrata Basu

Policy: POL-314567939

MEMBER

Insurance Details

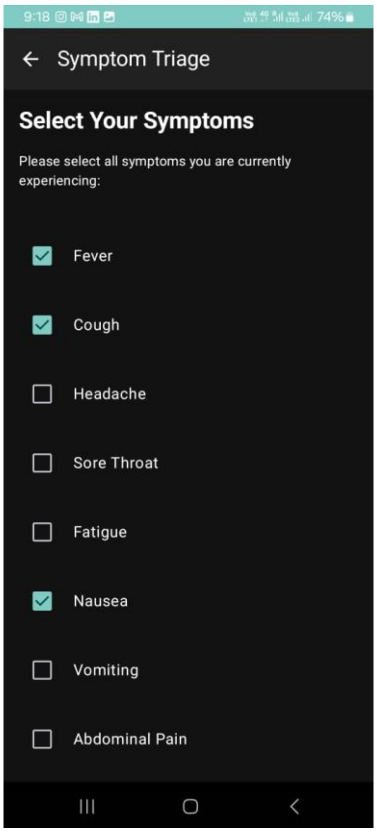
Policy Number	POL-314567939
Member Name	Jayabrata Basu
Insurance Provider	SBI safe health
Valid From	May 06, 2025
Valid Until	May 06, 2026

Coverage Information

Primary Care Visit	$25 copay
Specialist Visit	$50 copay
Emergency Room	$300 copay
Urgent Care	$75 copay
Annual Deductible	$1,500
Out-of-Pocket Maximum	$7,000

Recent Claims

May 15, 2023 General Hospital	$750.00 Paid
April 3, 2023 City Medical Center	$1,200.00 Processing
March 22, 2023	$175.00


9:18 74%

Symptom Triage

Select Your Symptoms

Please select all symptoms you are currently experiencing:

[✓ Fever]

[✓ Cough]

Headache

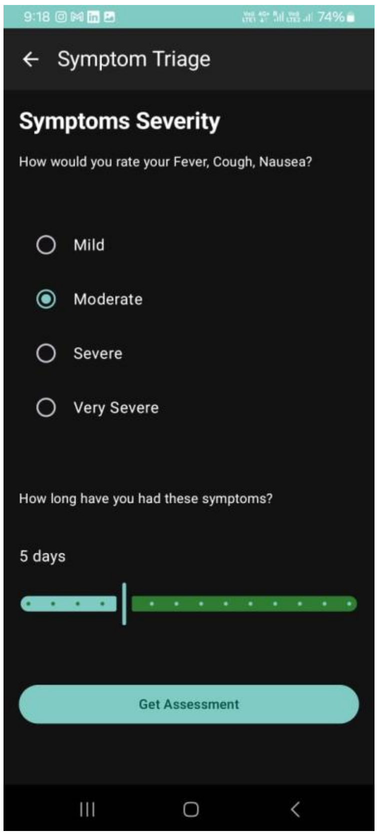
Sore Throat

Fatigue

[✓ Nausea]

Vomiting

Abdominal Pain


9:18 74%

Symptom Triage

Symptoms Severity

How would you rate your Fever, Cough, Nausea?

Mild [✓ Moderate] Severe Very Severe

How long have you had these symptoms?

5 days

Fig 5. Insurance Screen Fig 6. Symptom Selection Fig 7. Symptom Severity

The above images are the frontend of the project built in tandem with the backend. The project is deployed as an Android application.

6. Conclusion & Future Enhancements

The Healthcare Management System successfully addresses the need for structured and reliable handling of patient data, medications, insurance details, and medical consultations. By utilizing a relational database approach, it ensures consistency, accuracy, and efficiency in data storage and retrieval. The integration of SQL features like constraints, joins, views, triggers, and cursors not only enhances the performance but also automates key processes, reducing manual errors and improving the overall functionality of the system.

Looking ahead, the system can be enhanced by integrating a user-friendly front-end interface for real-time interaction with the database, enabling patients and doctors to access and update records more intuitively. Future enhancements could also include AI-based health risk prediction using patient history, integration with wearable health devices for real-time monitoring, and a notification system for appointments and medication reminders. Additionally, implementing role-based access control and encrypting sensitive data would further strengthen the system's security and scalability for deployment in real-world healthcare environments.

Furthermore, expanding the system to support multilingual access and cloud-based deployment can make it more inclusive and accessible to users across different regions. Incorporating analytics dashboards for doctors and healthcare administrators can provide valuable insights into patient trends and help in making informed decisions. These future improvements aim to transform the system into a more intelligent, secure, and scalable healthcare solution.

7. References

- https://infermedica.com/blog/articles/guide-to-virtual-triage-symptom-checkers?utm_source=perplexity
- https://www.nature.com/articles/s41746-022-00667-w?utm_source=perplexity
- https://www.bmj.com/content/351/bmj.h3480?utm_source=perplexity
- https://pmc.ncbi.nlm.nih.gov/articles/PMC10874001/?utm_source=perplexity