REPORT ON

**CASE STUDY**

**Hospital Emergency System**

**Submitted as part of Internal – 2 on (28-04-2025) for B.E IV – SEMESTER**

**In**

**COMPUTER SCIENCE AND ENGINEERING BY TEAM-10:**

**M DHEERAJ ADITHYA 160123733113**

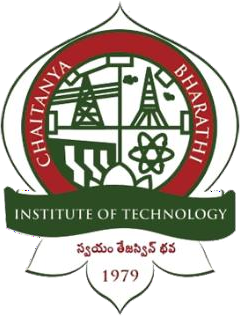
**POLA VENKATESHWARA REDDY** **160123733126**

**PULLAYI THRISHA 160123733309**

**Submitted to**

**Dr J Shiva Sai & Dr A Mohan Assistant Professor**

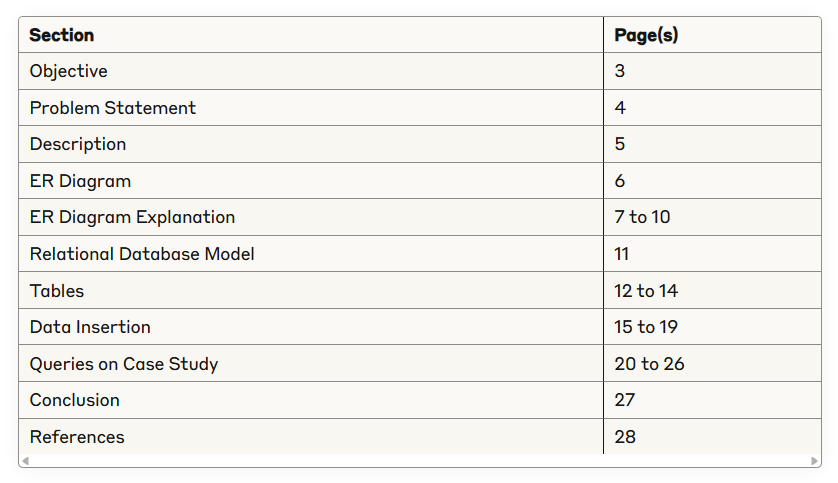
**Department of Computer Science & Engineering CBIT(A)**

****

**CHAITANYA BHARATHI INSTITUTE OF TECHNOLOGY (A)**

**(Affiliated to Osmania University) Gandipet,Hyderabad-500075**

**TABLE OF CONTENT**

****

**OBJECTIVE**

The main objective of this report is to design and implement a comprehensive Hospital Emergency Management Database System that effectively organizes and oversees critical hospital operations associated with emergency care. This involves the structured storage and management of patient details, doctor and staff information, room and ambulance availability, emergency case monitoring, treatments and medical tests, along with billing activities. The aim is to maintain data integrity, optimize emergency workflows, support timely decision-making, and improve the overall efficiency of hospital operations through a well-structured and relational database design. The system is intended to act as a strong backend for hospital information systems, enabling secure data storage, accurate data retrieval, and automated handling of emergency services.

**PROBLEM STATEMENT**

Suppose you are tasked with designing a hospital emergency management database system based on the following details:

* Each **patient** has a unique patient ID, name, age, gender, contact number, and address.
* Each **doctor** has a unique doctor ID, name, qualification, specialization, and phone number.
* Each **intern** has unique intern ID, name, phone number and doctor ID under whose guidance they work
* Each **room** has a unique room ID, type (e.g., ICU), and status (e.g., occupied, available).
* Each **ambulance** has a unique ambulance ID, driver’s name, and status.
* An **emergency case** is identified by a unique case ID and is associated with one patient, one doctor, one room, and one ambulance. It includes date and time, severity level, and a description.
* Each **treatment** is associated with one emergency case and one doctor, and includes treatment notes and medication given.
* Each **medical test** is linked to one emergency case and includes the test name and its result.
* Each **billing record** is linked to one emergency case and contains a unique bill ID, billed amount, and payment status.

**DESCRIPTION**

### Hospital Emergency Management Database System

The Hospital Emergency Management Database System is developed to efficiently handle emergency operations within a hospital by integrating crucial components, thereby improving patient care and administrative processes.

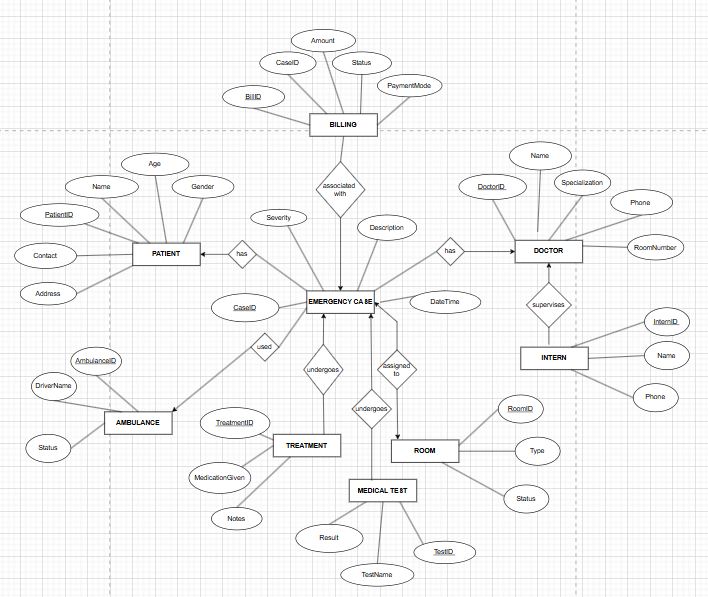
* **Patient Information**: maintains individual patient records such as ID, name, age, gender, contact, and address.
* **Doctor Details**: stores doctors profile including ID, name, specialization, and phone number.
* **Intern Details**: Maintain data on interns including ID, name , phone number and doctor
* **Room Management**: oversees room assignment with unique room IDs, types (ICU, general), and their status (available/occupied).
* **Ambulance Tracking**: Manages ambulance availability with unique IDs, driver names, and statuses.
* **Emergency Cases**: Associates each emergency case with corresponding patients, doctors, rooms, and ambulances, while recording date, time, severity level, and a brief case description.
* **Treatments & Medications**: Documents the treatments administered during emergencies, including doctors’ notes and prescribed medications.
* **Medical Tests**: Records details of medical tests conducted for each emergency case, including test names and results.
* **Billing**: Links billing records to emergency cases, tracking amounts and payment statuses.

This system facilitates quick access to critical data for better emergency response, patient care, and hospital management.

**CONTRIBUTION**

| **Name** | **Roll Number** | **Contribution** |
| --- | --- | --- |
| **M DHEERAJ ADITHYA** | **160123733113** | **Designed relational database model from ER diagram and inserted data for 5 tables.** |
| **POLA VENKATESHWARA REDDY** | **160123733126** | **Drew the ER diagram, created tables,inserted data into 4 tables.** |
| **PULLAYI THRISHA** | **160123733309** | **Created 11 queries, developed few tables and solved queries.** |

**ER DIAGRAM**

****

**ER DIAGRAM EXPLANATION**

This system is designed to manage emergency medical situations by recording information about patients, medical staff, facilities, and treatments. The following entities, attributes, and relationships form the basis of the database structure.

## PATIENT

### Attributes:

* + PatientID (PK): Unique identifier for each patient
  + Name: Full name of the patient
  + Age: Age of the patient
  + Gender: Gender (Male/Female/Other)
  + Phone number
  + Address: Residential address

### Relationships:

* + has → EMERGENCY CASE

One patient can be involved in multiple emergency cases (1:N)

## EMERGENCY CASE

### Attributes:

* + CaseID (PK): Unique identifier for each emergency case
  + Severity: Urgency level of the case (e.g., Critical, Moderate)
  + Description: Summary or notes regarding the case
  + Date&Time: Date and time when the emergency occurred

### Relationships:

* + has ← PATIENT: Belongs to a specific patient
  + used → AMBULANCE: One ambulance is used per emergency case
  + assigned to → DOCTOR, INTERN, ROOM: One doctor and interns, and one room, may be assigned
  + undergoes → TREATMENT, MEDICAL TEST: Each case may involve one or more treatments and tests
  + associated with → BILLING: Each emergency case results in multiple billing records

## AMBULANCE

### Attributes:

* + AmbulanceID (PK): Unique ID for each ambulance
  + DriverName: Name of the ambulance driver
  + Status: Indicates if the ambulance is "Available" or "Unavailable"

### Relationships:

* + used by ← EMERGENCY CASE: Each emergency case uses one ambulance and many emergency cases use one ambulance (M:1)

## DOCTOR

### Attributes:

* + DoctorID (PK): Unique ID for each doctor
  + Name: Full name of the doctor
  + Specialization: Field of medicine (e.g., Cardiology, Neurology)
  + Phone: Contact number
  + RoomNumber: Room assigned to the doctor

### Relationships:

* + has → EMERGENCY CASE: A doctor may handle multiple emergency cases
  + supervises → INTERN: A doctor can supervise multiple interns

## INTERN

### Attributes:

* + InternID (PK): Unique ID for each intern
  + Name: Full name
  + Phone: Contact number

### Relationships:

* + assigned to → EMERGENCY CASE: An intern may assist in multiple cases
  + supervised by ← DOCTOR: Each intern is supervised by one doctor

## ROOM

### Attributes:

* + RoomID (PK): Unique identifier for each room
  + Type: Type of room (e.g., ICU, General, Emergency)
  + Status: Indicates whether the room is Occupied or Available

### Relationships:

* + assigned to ← EMERGENCY CASE: Each case is assigned to one specific room (1:1)

## TREATMENT

### Attributes:

* + TreatmentID (PK): Unique ID for each treatment
  + MedicationGiven: Details of medication or procedures administered
  + Notes: Additional observations or remarks

### Relationships:

* + undergoes ← EMERGENCY CASE: A case may undergo one or more treatments

## MEDICAL TEST

### Attributes:

* + TestID (PK): Unique ID for each test
  + TestName: Name or type of medical test (e.g., Blood Test, X-Ray)
  + Result: Result of the medical test

### Relationships:

* + undergoes ← EMERGENCY CASE: Each case may include multiple medical tests

## BILLING

### Attributes:

* + BillID (PK): Unique billing ID
  + Amount: Total billed amount
  + Status: Indicates whether the bill is Paid or Unpaid
  + PaymentMode: Mode of payment (e.g., Cash, Card, Insurance)

### Relationships:

* + associated with ← EMERGENCY CASE: Each emergency case may associate with multiple billing record (M:1)

**Relationship Summary**

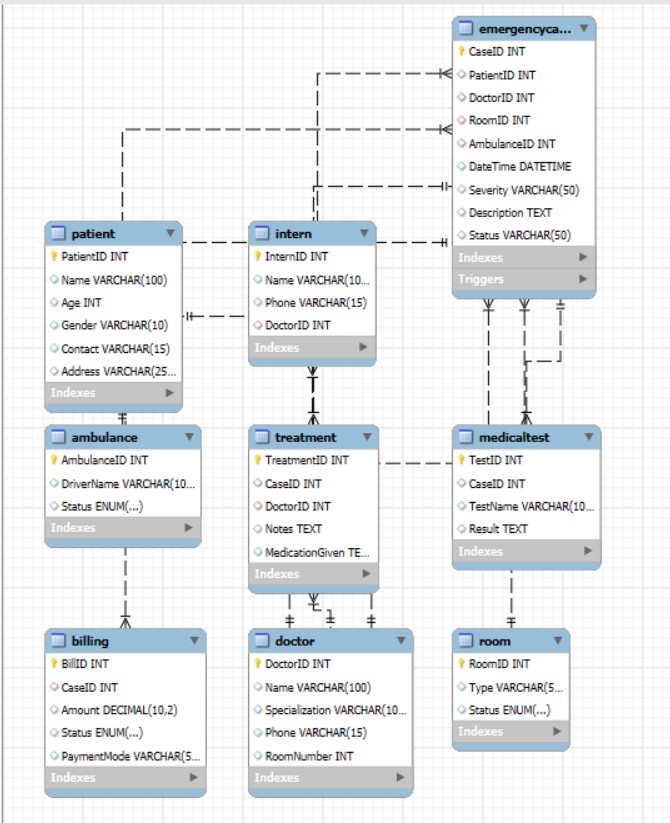
### One-to-Many (1:N):

* + One Patient → Many Emergency Cases
  + One Doctor → Many Interns
  + One Emergency Case → Doctors, Interns, Treatments, and Medical Tests

### One-to-One (1:1):

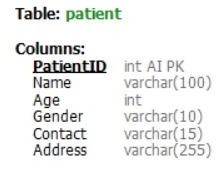
* + One Emergency Case → One Ambulance
  + One Emergency Case → One Room

**RELATIONAL DATABASE MODEL**

****

**TABLES**

# CODE: OUTPUT:

1. **Create Patient Table**

CREATE TABLE Patient (

PatientID INT AUTO\_INCREMENT PRIMARY KEY, Name VARCHAR(100),

Age INT,

Gender VARCHAR(10),

Contact VARCHAR(15), Address VARCHAR(255)

);

**2. Create Doctor Table (now Room exists)** CREATE TABLE Doctor (

DoctorID INT AUTO\_INCREMENT PRIMARY KEY, Name VARCHAR(100),

Specialization VARCHAR(100), Phone VARCHAR(15),

RoomNumber INT,

);

**3. Create Room Table (required by Doctor)** CREATE TABLE Room (

RoomID INT AUTO\_INCREMENT PRIMARY KEY, Type VARCHAR(50),

Status ENUM('Available', 'Occupied', 'Cleaning')

);

**4. Create Ambulance Table** CREATE TABLE Ambulance (

AmbulanceID INT AUTO\_INCREMENT PRIMARY KEY,

DriverName VARCHAR(100),

Status ENUM('Available', 'In Transit', 'Occupied', 'Under Maintenance'))

**5. Create Intern Table (after Doctor)** CREATE TABLE Intern (

InternID INT AUTO\_INCREMENT PRIMARY KEY, Name VARCHAR(100),

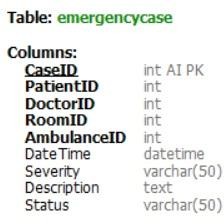
Phone VARCHAR(15),

DoctorID INT,

FOREIGN KEY (DoctorID) REFERENCES

Doctor(DoctorID)

);

**6. Create EmergencyCase Table (references Patient, Doctor, Ambulance, Room)**

CREATE TABLE EmergencyCase (

CaseID INT AUTO\_INCREMENT PRIMARY KEY,

PatientID INT, DoctorID INT, RoomID INT, AmbulanceID INT, DateTime DATETIME,

Severity VARCHAR(50), Description TEXT,

FOREIGN KEY (PatientID) REFERENCES Patient(PatientID), FOREIGN KEY (DoctorID) REFERENCES Doctor(DoctorID), FOREIGN KEY (RoomID) REFERENCES Room(RoomID),

FOREIGN KEY (AmbulanceID) REFERENCES Ambulance(AmbulanceID) );

**7. Create Treatment Table (references EmergencyCase and Doctor)** CREATE TABLE Treatment (

TreatmentID INT AUTO\_INCREMENT PRIMARY KEY, CaseID INT,

DoctorID INT, Notes TEXT,

MedicationGiven TEXT,

FOREIGN KEY (CaseID) REFERENCES EmergencyCase(CaseID), FOREIGN KEY (DoctorID) REFERENCES Doctor(DoctorID)

);

** 8. Create MedicalTest Table (reference EmergencyCase)**

CREATE TABLE MedicalTest (

TestID INT AUTO\_INCREMENT PRIMARY KEY, CaseID INT,

TestName VARCHAR(100),

Result TEXT,

FOREIGN KEY (CaseID) REFERENCES

EmergencyCase(CaseID)

);

**9. Create Billing Table (references EmergencyCase)** CREATE TABLE Billing (

BillID INT AUTO\_INCREMENT PRIMARY KEY, CaseID INT,

Amount DECIMAL(10, 2),

Status ENUM('Paid', 'Pending'),

FOREIGN KEY (CaseID) REFERENCES EmergencyCase(CaseID)

);

**DATA INSERTION**

### Table: Patient

### INSERT INTO Patient (Name, Age, Gender, Contact, Address) VALUES

### ('Rajesh Sharma', 45, 'Male', '9876543210', '123 MG Road, Mumbai'),

### ('Priya Patel', 32, 'Female', '8765432109', '456 Brigade Road, Bangalore'),

### ('Amit Singh', 60, 'Male', '7654321098', '789 Connaught Place, Delhi'),

### ('Ananya Gupta', 28, 'Female', '6543210987', '321 Park Street, Kolkata'),

### ('Vikram Joshi', 52, 'Male', '5432109876', '654 Jubilee Hills, Hyderabad'),

### ('Neha Reddy', 35, 'Female', '4321098765', '987 Koregaon Park, Pune'),

### ('Sanjay Kumar', 70, 'Male', '3210987654', '135 Boat Club Road, Chennai'),

### ('Divya Iyer', 40, 'Female', '2109876543', '246 Banjara Hills, Hyderabad');

### 

### Table: Doctor

### INSERT INTO Doctor (Name, Specialization, Phone, RoomNumber) VALUES

### ('Dr. Arun Khanna', 'Cardiology', '9876512345', 101),

### ('Dr. Deepika Sharma', 'Neurology', '8765423456', 102),

### ('Dr. Harish Patel', 'Orthopedics', '7654334567', 103),

### ('Dr. Meera Krishnan', 'Pediatrics', '6543245678', 104),

### ('Dr. Sameer Joshi', 'Emergency Medicine', '5432156789', 105),

### ('Dr. Nandini Reddy', 'General Surgery', '4321067890', 106),

### ('Dr. Vikas Malhotra', 'Oncology', '3210978901', 107),

### ('Dr. Ananya Iyer', 'Trauma Surgery', '2109889012', 108);

### 

### Table: Intern

### INSERT INTO Intern (Name, Phone, DoctorID) VALUES

### ('Rahul Verma', '9876501234', 1),

### ('Shreya Menon', '8765412345', 2),

### ('Aditya Banerjee', '7654323456', 3),

### ('Pooja Chatterjee', '6543234567', 4),

### ('Karan Oberoi', '5432145678', 5),

### ('Tanvi Kapoor', '4321056789', 6),

### ('Aryan Saxena', '3210967890', 7),

### ('Isha Nair', '2109878901', 8);

### 

### Table: EmergencyCase

### INSERT INTO EmergencyCase (PatientID, DoctorID, RoomID, AmbulanceID, DateTime, Severity, Description) VALUES

### (1, 1, 1, 1, '2023-10-15 08:30:00', 'Critical', 'Heart attack symptoms'),

### (2, 2, 2, 2, '2023-10-15 09:15:00', 'Moderate', 'Severe migraine'),

### (3, 3, 3, 3, '2023-10-15 10:00:00', 'Serious', 'Broken leg from fall'),

### (4, 4, 4, 4, '2023-10-15 11:45:00', 'Mild', 'Child with high fever'),

### (5, 5, 5, 5, '2023-10-15 13:20:00', 'Critical', 'Severe chest pain'),

### (6, 6, 6, 6, '2023-10-15 14:30:00', 'Serious', 'Appendicitis symptoms'),

### (7, 7, 7, 7, '2023-10-15 16:00:00', 'Moderate', 'Difficulty breathing'),

### (8, 8, 8, 8, '2023-10-15 17:45:00', 'Critical', 'Car accident injuries');

### 

### Table: Room

### INSERT INTO Room (Type, Status) VALUES

### ('General Ward', 'Available'),

### ('ICU', 'Occupied'),

### ('Operation Theater', 'Cleaning'),

### ('Emergency Room', 'Available'),

### ('Private Room', 'Occupied'),

### ('Recovery Room', 'Available'),

### ('Isolation Room', 'Available'),

### ('Pediatric Ward', 'Available');

### 

### Table: Treatment

INSERT INTO Treatment (CaseID, DoctorID, Notes, MedicationGiven) VALUES

(1, 1, 'Administered ECG and blood tests', 'Aspirin, Nitroglycerin'),

(2, 2, 'Performed neurological examination', 'Ibuprofen, Triptans'),

(3, 3, 'Set fracture and applied cast', 'Painkillers, Calcium supplements'),

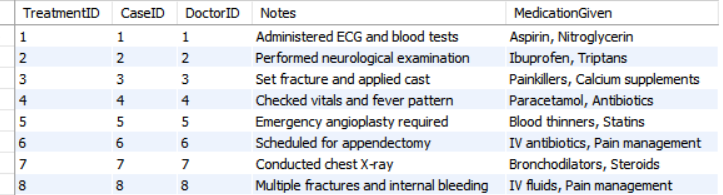
(4, 4, 'Checked vitals and fever pattern', 'Paracetamol, Antibiotics'),

(5, 5, 'Emergency angioplasty required', 'Blood thinners, Statins'),

(6, 6, 'Scheduled for appendectomy', 'IV antibiotics, Pain management'),

(7, 7, 'Conducted chest X-ray', 'Bronchodilators, Steroids'),

(8, 8, 'Multiple fractures and internal bleeding', 'IV fluids, Pain management');



### Table: MedicalTest

### INSERT INTO MedicalTest (CaseID, TestName, Result) VALUES

### (1, 'ECG', 'ST elevation detected'),

### (1, 'Troponin Test', 'Positive for heart attack'),

### (2, 'MRI Brain', 'No abnormalities detected'),

### (3, 'X-ray Leg', 'Fractured tibia confirmed'),

### (4, 'Blood Test', 'High WBC count'),

### (5, 'Angiogram', '90% blockage in LAD artery'),

### (6, 'Ultrasound', 'Inflamed appendix confirmed'),

### (7, 'CT Scan', 'Early stage pneumonia detected'),

### (8, 'Full Body CT', 'Multiple fractures and internal injuries');

### 

### Table: Ambulance

INSERT INTO Ambulance (DriverName, Status) VALUES

('Ramesh Nair', 'Available'),

('Sunita Desai', 'In Transit'),

('Arjun Mehta', 'Occupied'),

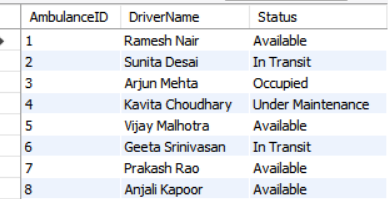
('Kavita Choudhary', 'Under Maintenance'),

('Vijay Malhotra', 'Available'),

('Geeta Srinivasan', 'In Transit'),

('Prakash Rao', 'Available'),

('Anjali Kapoor', 'Available');



### Table: Billing

INSERT INTO Billing (CaseID, Amount, Status) VALUES

(1, 25000.00, 'Pending'),

(2, 12000.00, 'Paid'),

(3, 18000.00, 'Pending'),

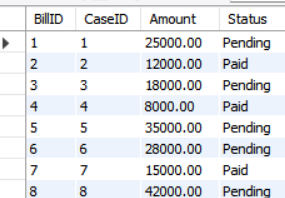
(4, 8000.00, 'Paid'),

(5, 35000.00, 'Pending'),

(6, 28000.00, 'Pending'),

(7, 15000.00, 'Paid'),

(8, 42000.00, 'Pending');



**QUERIES ON CASE STUDY**

**Query 1:** critical patientnames and doctors handling them.

**Description:** This query retrieves  the lists patients names and doctors of all critical patients.

# CODE:

SELECT p.Name, d.Name AS Doctor

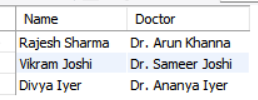
FROM EmergencyCase e

JOIN Patient p ON e.PatientID = p.PatientID

JOIN Doctor d ON e.DoctorID = d.DoctorID

WHERE e.Severity = 'Critical';

**OUTPUT:**

****

**Query 2:** Calculate Average Treatment Cost by Severity

**Description:** This query shows the average billing amount for cases grouped by severity level.

# CODE:

# SELECT e.Severity, AVG(b.Amount) AS AvgCost

# FROM EmergencyCase e

# JOIN Billing b ON e.CaseID = b.CaseID

# GROUP BY e.Severity;

# OUTPUT:

# 

**Query 3:** List Available Ambulances with Drivers

**Description:** This query displays all currently available ambulances with driver contact information.

# CODE:

# SELECT AmbulanceID, DriverName

# FROM Ambulance

# WHERE Status = 'Available';

# OUTPUT:

# 

**Query 4:** Show Patients with Outstanding Bills Over ₹20,000

**Description:** This query finds the lists patients with unpaid bills exceeding ₹20,000 for collection follow-up.

**CODE:**

SELECT p.Name, p.Contact, b.Amount

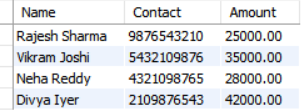
FROM Patient p

JOIN EmergencyCase e ON p.PatientID = e.PatientID

JOIN Billing b ON e.CaseID = b.CaseID

WHERE b.Status = 'Pending' AND b.Amount > 20000;

**OUTPUT:**

****

**Query 5:** Count Cases by Hour of Day

**Description:** This query retrieves the emergency case distribution by hour to identify peak times.

**CODE:**

SELECT HOUR(DateTime) AS Hour, COUNT(\*) AS Cases

FROM EmergencyCase

GROUP BY Hour

ORDER BY Hour;

# OUTPUT:

# 

**Query 6:**  Find Long-Running Emergency Cases with doctors who have 3+ active cases.

**DESCRIPTION:**Identifies emergency cases that have been ongoing for more than 4 hours (based on admission time vs current time)

# CODE:

# SELECT e.CaseID, p.Name AS PatientName,TIMESTAMPDIFF(HOUR, e.DateTime, NOW()) HoursInTreatment

# FROM EmergencyCase e

# JOIN Patient p ON e.PatientID = p.PatientID

# WHERE TIMESTAMPDIFF(HOUR, e.DateTime, NOW()) > 4;

# OUTPUT:

# 

**Query 7:** List Critical Cases Without ICU Admission

**Description:** This query finds critical cases not assigned to ICU rooms for urgent review..

# CODE:

SELECT e.CaseID, p.Name AS Patient, r.Type AS CurrentRoom

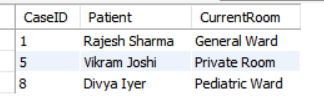
FROM EmergencyCase e

JOIN Patient p ON e.PatientID = p.PatientID

JOIN Room r ON e.RoomID = r.RoomID

WHERE e.Severity = 'Critical' AND r.Type != 'ICU';

**OUTPUT:**

****

**Query 8:**  Patients with Unpaid Bills

**Description:** This query Shows patients who have billing records with pending status.

**CODE:**

# SELECT DISTINCT

# p.Name AS PatientName,

# b.Amount AS DueAmount

# FROM Patient p

# JOIN EmergencyCase e ON p.PatientID = e.PatientID

# JOIN Billing b ON e.CaseID = b.CaseID

# WHERE b.Status = 'Pending';

# OUTPUT:

# 

**Query 9:** Patients with Concurrent Tests & Treatments

**Description:** Identifies patients undergoing both tests and treatments simultaneously, flagging cases needing coordination between lab and medical teams.

# CODE:

SELECT p.PatientID, p.Name AS PatientName,

COUNT(DISTINCT mt.TestID) AS TestCount,

COUNT(DISTINCT t.TreatmentID) AS TreatmentCount

FROM Patient p

JOIN EmergencyCase e ON p.PatientID = e.PatientID

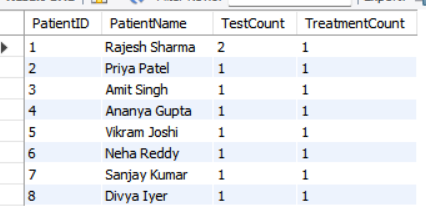
LEFT JOIN MedicalTest mt ON e.CaseID = mt.CaseID

LEFT JOIN Treatment t ON e.CaseID = t.CaseID

GROUP BY p.PatientID, p.Name

HAVING TestCount > 0 AND TreatmentCount > 0;

# OUTPUT:



**Query 10:** Create a procedure that get Latest Critical Case

**Description:**

DELIMITER //

CREATE PROCEDURE GetLatestCriticalCase()

BEGIN

SELECT

p.Name AS Patient,

d.Name AS Doctor,

e.DateTime AS AdmissionTime

FROM EmergencyCase e

JOIN Patient p ON e.PatientID = p.PatientID

JOIN Doctor d ON e.DoctorID = d.DoctorID

WHERE e.Severity = 'Critical'

ORDER BY e.DateTime DESC

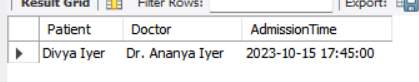
LIMIT 1;

END //

DELIMITER ;

-- CALL GetLatestCriticalCase();

**OUTPUT:**



**Query 11:** To automatically update the room status to "Occupied" when a new emergency case is inserted, but only if the room's status is "Available.

**Description:** This trigger ensures that when a new emergency case is recorded, the associated room’s status is updated to "Occupied," provided the room was initially available.

# CODE:

# DELIMITER $$

# CREATE TRIGGER UpdateRoomStatusAfterEmergencyCaseInsert

# AFTER INSERT ON EmergencyCase

# FOR EACH ROW

# BEGIN

# -- Only update room status if it's available

# IF (SELECT Status FROM Room WHERE RoomID = NEW.RoomID) = 'Available' THEN

# UPDATE Room

# SET Status = 'Occupied'

# WHERE RoomID = NEW.RoomID;

# END IF;

# END $$

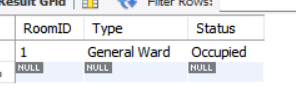
# DELIMITER ;

# INSERT INTO EmergencyCase (PatientID, DoctorID, RoomID, AmbulanceID, DateTime, Severity, Description)

# VALUES (1, 1, 1, 1, '2023-10-16 08:00:00', 'Critical', 'Chest pain');

# SELECT \* FROM Room WHERE RoomID = 1;

**OUTPUT:**

****

**CONCLUSION**

This report highlights various SQL operations and their application in efficiently managing a hospital database system. The operations covered demonstrate the practical use of SQL in real-world healthcare scenarios. Key areas of focus include:

* **Table Modifications and Updates**: Implementing ALTER TABLE and UPDATE statements to dynamically modify data, such as adding a PaymentMode column to billing records.
* **String Manipulations**: Utilizing functions like REPLACE and LIKE to correct and standardize patient contact numbers based on specific patterns.
* **Data Analysis with Joins and Aggregations**: Using JOIN, GROUP BY, and COUNT to analyze relationships within the database, such as categorizing emergency cases by severity or identifying patients with outstanding bills.
* **Advanced Data Retrieval Techniques**: Leveraging subqueries, EXISTS, and DISTINCT to filter and extract critical information, like doctor involvement in emergency cases and identifying unique room types.
* **Stored Procedures and Cursors**: Developing stored procedures that use cursors for complex calculations, such as computing the total billing for each emergency case, enabling server-side automation and processing.
* **Triggers for Data Integrity**: Creating triggers to automatically delete patient records after discharge, ensuring efficient record management and data consistency.

This report demonstrates a solid understanding of SQL and its role in building and maintaining an efficient hospital management system. By employing dynamic SQL features, procedural logic, and automation, the system ensures accurate record-keeping while enhancing responsiveness to real-time data updates.

**REFERENCES**

1. **GeeksforGeeks** – <https://www.geeksforgeeks.org>  
   – Used for tutorials and practical examples on relational databases, primary and foreign keys, database normalization, and query optimization.
2. **Elmasri, R., & Navathe, S. B.** (2017). *Fundamentals of Database Systems* (7th ed.). Pearson Education.
   * Used for understanding Entity-Relationship modelling, relational schema design, and normalization techniques.
3. **W3Schools SQL Tutorial** – <https://www.w3schools.com/sql/>  
   – A helpful reference for understanding SQL syntax, queries, constraints, and relationships. It provided foundational knowledge for writing efficient database queries.
4. **MySQL Documentation** – <https://dev.mysql.com/doc/>  
   – Essential for MySQL-specific syntax, constraints, and SQL functions, facilitating accurate relational schema design and query execution.
5. **Draw.io (diagrams.net)** – <https://app.diagrams.net/>  
   – Tool used for creating the ER diagram representing the hospital emergency management system, helping visualize the relationships between entities.
6. **Course Material & Lectures** – *Database Management System Course* (CBIT CSE Dept., 2nd Year)  
   – Theoretical foundation and academic guidelines followed during the project, providing essential knowledge on database management concepts.
7. **ChatGPT (OpenAI)** – <https://chat.openai.com>  
   – Assisted in ideation, clarifying DBMS concepts, troubleshooting, and refining SQL queries for the database system.