ASSIGNMENT-4

NAME:A.RAKCHA

REGNO:192311421

1. Design a database to manage patients, Doctors and Appointment.

-model tables for patients, Doctors and Appointment.

Write stored procedures for list appointments for a doctor.

Implement triggers to set appointment status based on date.

Write SQL queries to analyze appointments for a doctor in a given data range.

**1. Database Model (Tables)**

We will create three main tables:

* **patients**: Stores information about patients.
* **doctors**: Stores information about doctors.
* **appointments**: Stores details about appointments made by patients with doctors.

**Table 1: patients**

sql

Copy code

CREATE TABLE patients (

patient\_id INT PRIMARY KEY AUTO\_INCREMENT,

patient\_name VARCHAR(255) NOT NULL,

date\_of\_birth DATE NOT NULL,

contact\_info VARCHAR(255)

);

**Table 2: doctors**

sql

Copy code

CREATE TABLE doctors (

doctor\_id INT PRIMARY KEY AUTO\_INCREMENT,

doctor\_name VARCHAR(255) NOT NULL,

specialization VARCHAR(255),

contact\_info VARCHAR(255)

);

**Table 3: appointments**

sql

Copy code

CREATE TABLE appointments (

appointment\_id INT PRIMARY KEY AUTO\_INCREMENT,

patient\_id INT,

doctor\_id INT,

appointment\_date DATETIME NOT NULL,

status VARCHAR(50) NOT NULL DEFAULT 'Scheduled', -- e.g., 'Scheduled', 'Completed', 'Cancelled'

notes TEXT,

CONSTRAINT fk\_patient FOREIGN KEY (patient\_id) REFERENCES patients(patient\_id),

CONSTRAINT fk\_doctor FOREIGN KEY (doctor\_id) REFERENCES doctors(doctor\_id)

);

**2. Stored Procedures**

**Stored Procedure to List Appointments for a Doctor**

This stored procedure allows listing all appointments scheduled with a specific doctor, either for a given date or a range of dates.

sql

Copy code

CREATE PROCEDURE ListAppointmentsForDoctor(IN doctor\_id INT, IN start\_date DATETIME, IN end\_date DATETIME)

BEGIN

SELECT a.appointment\_id, p.patient\_name, a.appointment\_date, a.status

FROM appointments a

JOIN patients p ON a.patient\_id = p.patient\_id

WHERE a.doctor\_id = doctor\_id

AND a.appointment\_date BETWEEN start\_date AND end\_date

ORDER BY a.appointment\_date;

This procedure can be called like this:

sql

Copy code

CALL ListAppointmentsForDoctor(1, '2024-11-01', '2024-11-30');

This will list all appointments for doctor with doctor\_id = 1 in the month of November 2024.

**3. Triggers**

**Trigger to Set Appointment Status Based on Date**

This trigger will automatically update the status of an appointment to 'Completed' if the appointment date is in the past and the status is still 'Scheduled'. It will also set appointments scheduled in the future to 'Scheduled' if they are not yet 'Cancelled' or 'Completed'.

sql

Copy code

CREATE TRIGGER UpdateAppointmentStatus

AFTER INSERT OR UPDATE ON appointments

FOR EACH ROW

BEGIN

-- If appointment date is in the past and status is still 'Scheduled', update to 'Completed'

IF NEW.appointment\_date < NOW() AND NEW.status = 'Scheduled' THEN

UPDATE appointments

SET status = 'Completed'

WHERE appointment\_id = NEW.appointment\_id;

END IF;

-- If appointment date is in the future and status is not 'Cancelled' or 'Completed', keep status 'Scheduled'

IF NEW.appointment\_date > NOW() AND NEW.status NOT IN ('Cancelled', 'Completed') THEN

UPDATE appointments

SET status = 'Scheduled'

WHERE appointment\_id = NEW.appointment\_id;

END IF;

This trigger will ensure that the status of appointments is updated automatically when they are inserted or updated based on the appointment date.

**4. SQL Queries for Analyzing Appointments**

**Query to Analyze Appointments for a Doctor in a Given Date Range**

This query returns the number of appointments a doctor has in a given date range along with the status of the appointments.

sql

Copy code

SELECT doctor\_name, COUNT(a.appointment\_id) AS total\_appointments,

SUM(CASE WHEN a.status = 'Completed' THEN 1 ELSE 0 END) AS completed\_appointments,

SUM(CASE WHEN a.status = 'Scheduled' THEN 1 ELSE 0 END) AS scheduled\_appointments,

SUM(CASE WHEN a.status = 'Cancelled' THEN 1 ELSE 0 END) AS cancelled\_appointments

FROM appointments a

JOIN doctors d ON a.doctor\_id = d.doctor\_id

WHERE a.appointment\_date BETWEEN '2024-11-01' AND '2024-11-30'

GROUP BY d.doctor\_id;

This query will show the total number of appointments, completed, scheduled, and cancelled appointments for a doctor in November 2024.

**Example Output (for one doctor)**

| **doctor\_name** | **total\_appointments** | **completed\_appointments** | **scheduled\_appointments** | **cancelled\_appointments** |
| --- | --- | --- | --- | --- |
| Dr. Smith | 50 | 40 | 5 | 5 |

**5. Conclusion**

The database system is designed to efficiently manage patient, doctor, and appointment data. The key features include:

1. **Tables**:
   * patients: Stores patient information.
   * doctors: Stores doctor information.
   * appointments: Tracks the details of appointments between patients and doctors.
2. **Stored Procedures**:
   * **ListAppointmentsForDoctor**: Allows querying of appointments for a doctor within a specific date range, helping doctors and administrators track their schedules.
3. **Triggers**:
   * **UpdateAppointmentStatus**: Automatically updates the status of appointments based on their date, ensuring that appointments in the past are marked as 'Completed' and future appointments remain 'Scheduled'.
4. **SQL Queries**:
   * The query to analyze appointments provides insights into the doctor’s workload, including the number of completed, scheduled, and cancelled appointments within a given date range. This helps in assessing the doctor’s performance and efficiency.

**Benefits:**

* **Automation**: The trigger ensures that appointment statuses are updated without manual intervention.
* **Efficiency**: The stored procedure allows for easy listing of appointments based on date ranges, reducing administrative workload.
* **Data-Driven Insights**: The SQL queries offer valuable insights into the doctor's schedule, helping with performance evaluation and future planning.

This database model can be easily expanded or modified to accommodate additional features like patient medical records, payment processing, or appointment reminders, making it a scalable solution for healthcare management.