DBMS Project



**Hospital Management System**

Submitted to: Dr.Sanjeev Rao

Gurjot Singh 102103173

Mayank Chopra 102103158

**INDEX**

1. Problem statement
2. Overview
3. Requirement Analysis
4. ER Diagram
5. ER to Table
6. Normalisation
7. Functional Dependencies
8. Code and Output

**Problem Statement**

Developing a hospital management system in order to effectively manage most aspects of hospitals such as booking appointments, managing patient records and keeping medical report.

**Overview**

The hospital is a multi-specialty facility that provides a wide range of healthcare services, including inpatient care, outpatient care, surgeries, diagnostics, and pharmacy services. The hospital has multiple departments, such as cardiology, oncology, orthopedics, pediatrics, etc., and each department has its own team of doctors, nurses, and support staff. Hence it is very important for a hospital to have a DBMS that easily allows patients to book appointments and allows doctors or administrators to manage patient data.

**Requirements Analysis**

1. Patient Management:

* Ability to add new patients with their details such as patient\_id, patient\_name, patient\_dob, patient\_gender, patient\_phone, and patient\_address.
* Ability to retrieve patient information based on patient\_id or patient\_name.
* Ability to update patient information such as patient\_name, patient\_dob, patient\_gender, patient\_phone, and patient\_address.

1. Doctor Management:

* Ability to add new doctors with their details such as doctor\_id, doctor\_name, doctor\_email, and doctor\_phone.
* Ability to retrieve doctor information based on doctor\_id or doctor\_name.
* Ability to update doctor information such as doctor\_name, doctor\_email, and doctor\_phone.

1. Appointment Management:

* Ability to add new appointments with their details such as appointment\_id, appointment\_date, patient\_id, doctor\_id, and appointment\_reason.
* Ability to retrieve appointment information based on appointment\_id, patient\_id, or doctor\_id.
* Ability to update appointment information such as appointment\_date and appointment\_reason.

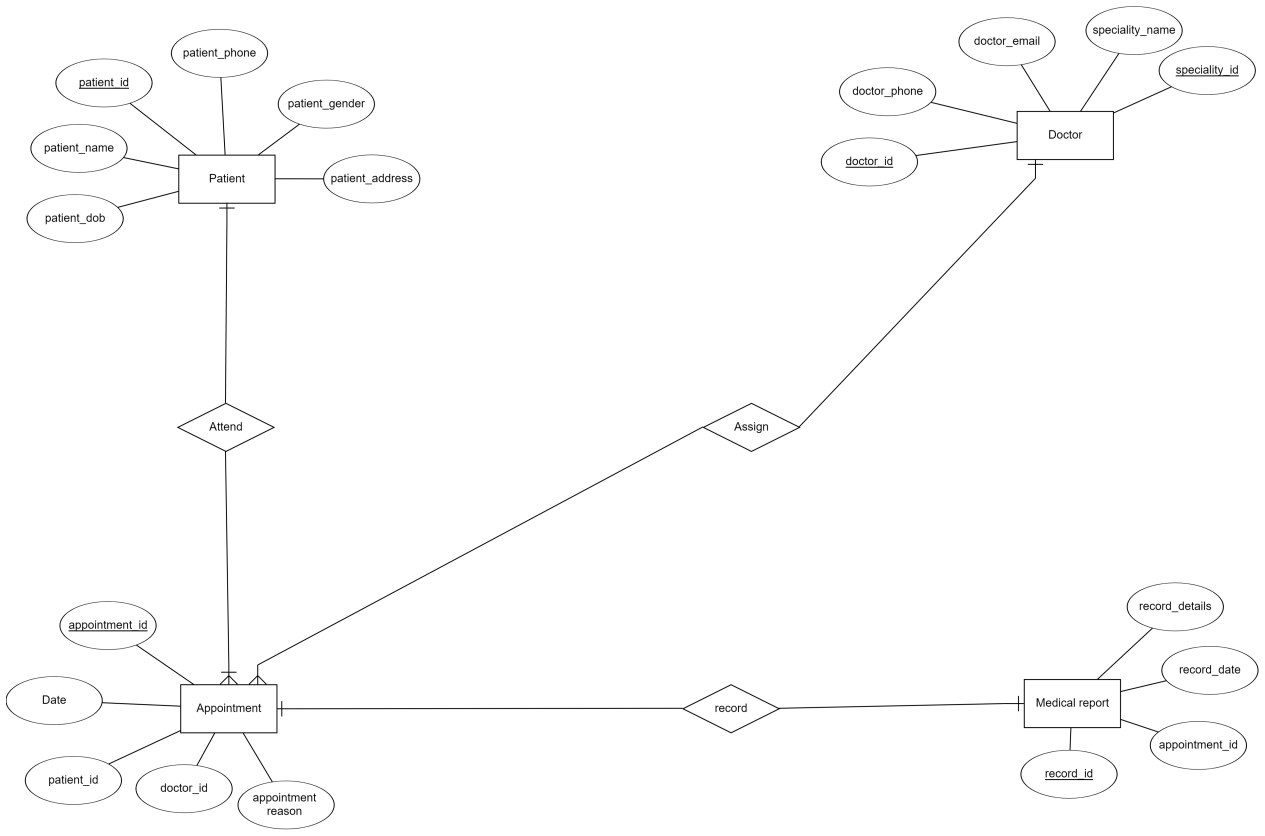
1. Medical Record Management:

* Ability to add new medical records with their details such as record\_id, appointment\_id, record\_date, and record\_details.
* Ability to retrieve medical record information based on record\_id or appointment\_id.
* Ability to update medical record information such as record\_date and record\_details.

1. Error Handling:

* Ability to handle errors that may occur during data insertion, retrieval, update, or deletion.
* Display appropriate error messages when errors occur, indicating the nature of the error and possible solutions.

**ER Diagram**

****

**ER Diagram To Table**

1 Patient

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Pid** | name | dob | gender | address | phone |

2 Doctor

|  |  |  |  |
| --- | --- | --- | --- |
| **Did** | doctor\_name | doctor\_email | doctor\_phone |
| **speciality\_id** | speciality\_name |

3 medical\_records

|  |  |  |  |
| --- | --- | --- | --- |
| **Rid** | Aid | record\_date | record\_details |

4 Appointment

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Aid** | appointment\_date | Pid | Did | appointment\_reason |

**Normalisation**

**Relation: Attend**

**UNF:** patient\_id, patient\_dob, patient\_name, patient\_gender, patient\_phone,patient\_address,appointment**\_**id, appointment\_date, patient\_id, doctor\_id, appointment\_reason

**2NF:**

1 **Patient** :

R = (**patient\_id**, patient\_dob, patient\_name, patient\_gender, patient\_phone,patient\_address)

**2 Appointment:**

R = (**appointment\_id**, appointment\_date, patient\_id, doctor\_id, appointment\_reason)

Already in 3nf

**Relation: Assign**

**UNF:**doctor\_id, doctor\_name, doctor\_email, doctor\_phone,speciality\_id**,** speciality\_name, appointment**\_**id, appointment\_date, patient\_id, doctor\_id, appointment\_reason

**2NF:**

**1 Doctor** :

R = (**doctor\_id**, doctor\_name, doctor\_email, doctor\_phone)

**2 Appointment:**

R = (**appointment\_id**, appointment\_date, patient\_id, doctor\_id, appointment\_reason)

**3 specialties:**

R = (**speciality\_id,** speciality\_name)

**3NF:**

**1 Doctor** :

R = (**doctor\_id**, doctor\_name, doctor\_email, doctor\_phone)

**2 Appointment:**

R = (**appointment\_id**, appointment\_date, patient\_id, doctor\_id, appointment\_reason)

**3 specialties:**

R = (**speciality\_id,** speciality\_name)

**4 doctors\_specialties:**

R = (doctor\_id(FK), speciality\_id(FK))

**Relation: Record**

**UNF:** appointment**\_**id, appointment\_date, patient\_id, doctor\_id, appointment\_reason,

record\_id, appointment\_id, record\_date, record\_details

**2 NF:**

**1 Appointment:**

R = (**appointment\_id**, appointment\_date, patient\_id, doctor\_id, appointment\_reason)

**2 Medical\_records** :

R = (**record\_id**, appointment\_id, record\_date, record\_details)

Already in 3nf

**Functional Dependencies:**

1. **Patient** :

R = (**patient\_id**, patient\_dob, patient\_name, patient\_gender, patient\_phone,patient\_address)

FDs:

1. patient\_id-> patient\_name
2. patient\_id-> patient\_dob
3. patient\_id-> patient\_gender
4. patient\_id-> patient\_phone
5. patient\_id->patient\_address
6. **medical\_records** :

R = (**record\_id**, appointment\_id, record\_date, record\_details)

FDs:

1. record\_id-> record\_date
2. record\_id-> record\_details
3. record\_id-> appointment\_id
4. **Doctor** :

R = (**doctor\_id**, doctor\_name, doctor\_email, doctor\_phone)

FDs:

1. doctor\_id-> doctor\_name
2. doctor\_id-> doctor\_email
3. doctor\_id-> doctor\_phone
4. **Appointment:**

R = (**appointment\_id**, appointment\_date, patient\_id, doctor\_id, appointment\_reason)

FDs:

1. appointment\_id-> appointment\_date
2. appointment\_id-> patient\_id
3. appointment\_id->doctor\_id
4. appointment\_id-> appointment\_reason
5. **doctors\_specialities:**

R = (**doctor\_id, speciality\_id**)

Since entire table is the key, it does not have partial and transitive dependencies. It also has atomic attributes.

1. **specialities:**

R = (**speciality\_id,** speciality\_name)

1. **speciality\_id**-> speciality\_name

**Final Database After Normalization**

1 Patient

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Pid** | name | dob | gender | address | phone |

2 Doctor

|  |  |  |  |
| --- | --- | --- | --- |
| **Did** | doctor\_name | doctor\_email | doctor\_phone |

3 medical\_records

|  |  |  |  |
| --- | --- | --- | --- |
| **Rid** | Aid | record\_date | record\_details |

4 Appointment

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Aid** | appointment\_date | Pid | Did | appointment\_reason |

5 doctors\_specialties

|  |  |
| --- | --- |
| Did | Pid |

6 specialties

|  |  |
| --- | --- |
| **speciality\_id** | speciality\_name |

**Code For Creation of Tables**

CREATE TABLE patients (

patient\_id INT PRIMARY KEY,

patient\_name VARCHAR2(50),

patient\_dob DATE,

patient\_gender CHAR(1),

patient\_phone VARCHAR2(20),

patient\_address VARCHAR2(100)

);

CREATE TABLE doctors (

doctor\_id INT PRIMARY KEY,

doctor\_name VARCHAR2(50),

doctor\_email VARCHAR2(50),

doctor\_phone VARCHAR2(20)

);

CREATE TABLE specialities (

speciality\_id INT PRIMARY KEY,

speciality\_name VARCHAR2(50)

);

CREATE TABLE doctors\_specialities (

doctor\_id INT,

speciality\_id INT,

FOREIGN KEY (doctor\_id) REFERENCES doctors(doctor\_id),

FOREIGN KEY (speciality\_id) REFERENCES specialities(speciality\_id)

);

CREATE TABLE appointments (

appointment\_id INT PRIMARY KEY,

appointment\_date DATE,

patient\_id INT,

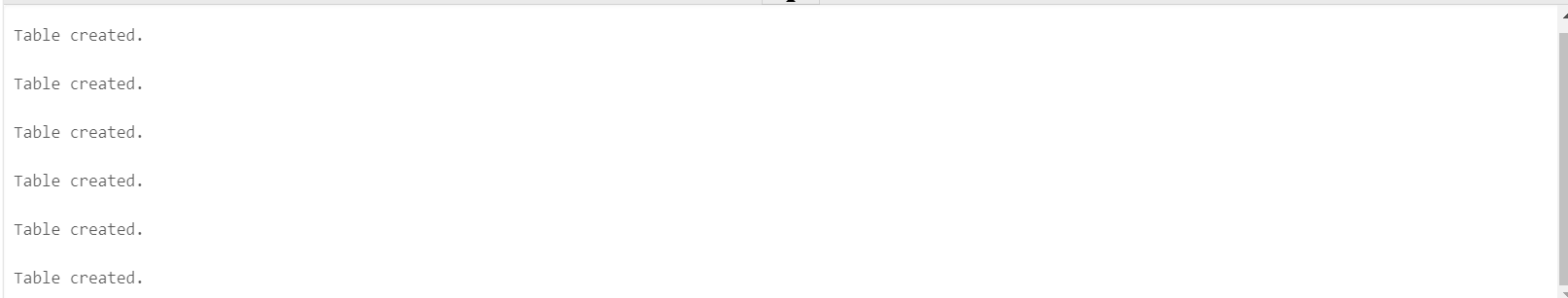
doctor\_id INT,

appointment\_reason VARCHAR2(100),

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

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

);



**Procedure To insert values into tables**

-- To add a patient

CREATE OR REPLACE PROCEDURE add\_patient (

p\_patient\_id IN patients.patient\_id%TYPE,

p\_patient\_name IN patients.patient\_name%TYPE,

p\_patient\_dob IN patients.patient\_dob%TYPE,

p\_patient\_gender IN patients.patient\_gender%TYPE,

p\_patient\_phone IN patients.patient\_phone%TYPE,

p\_patient\_address IN patients.patient\_address%TYPE

)

IS

BEGIN

INSERT INTO patients (

patient\_id,

patient\_name,

patient\_dob,

patient\_gender,

patient\_phone,

patient\_address

) VALUES (

p\_patient\_id,

p\_patient\_name,

p\_patient\_dob,

p\_patient\_gender,

p\_patient\_phone,

p\_patient\_address

);

COMMIT;

END;

/

-- TO add doctor

CREATE OR REPLACE PROCEDURE add\_doctor(

p\_doctor\_id IN doctors.doctor\_id%TYPE,

p\_doctor\_name IN doctors.doctor\_name%TYPE,

p\_doctor\_email IN doctors.doctor\_email%TYPE,

p\_doctor\_phone IN doctors.doctor\_phone%TYPE

)

IS

BEGIN

INSERT INTO doctors(doctor\_id, doctor\_name, doctor\_email, doctor\_phone)

VALUES(p\_doctor\_id, p\_doctor\_name, p\_doctor\_email, p\_doctor\_phone);

COMMIT;

dbms\_output.put\_line('Doctor added successfully');

EXCEPTION

WHEN OTHERS THEN

dbms\_output.put\_line('Error occurred while adding doctor: ' || SQLERRM);

END;

/

-- To add Specialities

CREATE OR REPLACE PROCEDURE add\_speciality(

p\_speciality\_id IN specialities.speciality\_id%TYPE,

p\_speciality\_name IN specialities.speciality\_name%TYPE

)

IS

BEGIN

-- insert the new speciality into the table

INSERT INTO specialities(speciality\_id, speciality\_name)

VALUES(p\_speciality\_id, p\_speciality\_name);

DBMS\_OUTPUT.PUT\_LINE('Speciality added successfully');

EXCEPTION

WHEN OTHERS THEN

DBMS\_OUTPUT.PUT\_LINE('Error adding speciality: ' || SQLERRM);

END;

/

--to add doctor\_specilisation

CREATE OR REPLACE PROCEDURE add\_doctor\_speciality(p\_doctor\_id IN INT, p\_speciality\_id IN INT)

IS

BEGIN

INSERT INTO doctors\_specialities(doctor\_id, speciality\_id)

VALUES (p\_doctor\_id, p\_speciality\_id);

COMMIT;

DBMS\_OUTPUT.PUT\_LINE('Doctor speciality added successfully.');

EXCEPTION

WHEN OTHERS THEN

ROLLBACK;

DBMS\_OUTPUT.PUT\_LINE('Error adding doctor speciality: ' || SQLERRM);

END;

/

-- Add an appointment

CREATE OR REPLACE PROCEDURE add\_appointment (

p\_appointment\_id IN appointments.appointment\_id%TYPE,

p\_appointment\_date IN appointments.appointment\_date%TYPE,

p\_patient\_id IN appointments.patient\_id%TYPE,

p\_doctor\_id IN appointments.doctor\_id%TYPE,

p\_appointment\_reason IN appointments.appointment\_reason%TYPE

)

IS

BEGIN

-- Insert appointment

INSERT INTO appointments (appointment\_id, appointment\_date, patient\_id, doctor\_id, appointment\_reason)

VALUES (p\_appointment\_id, p\_appointment\_date, p\_patient\_id, p\_doctor\_id, p\_appointment\_reason);

-- Commit the changes

COMMIT;

-- Display a message indicating successful insertion

DBMS\_OUTPUT.PUT\_LINE('Appointment added successfully!');

EXCEPTION

WHEN OTHERS THEN

-- Display an error message if an exception occurs

DBMS\_OUTPUT.PUT\_LINE('Error: ' || SQLERRM);

ROLLBACK;

END;

/

-- To add Medical\_record

CREATE OR REPLACE PROCEDURE add\_medical\_report (

p\_record\_id IN medical\_report.record\_id%TYPE,

p\_appointment\_id IN medical\_report.appointment\_id%TYPE,

p\_record\_date IN medical\_report.record\_date%TYPE,

p\_record\_details IN medical\_report.record\_details%TYPE

)

IS

BEGIN

-- Insert medical record

INSERT INTO medical\_report (record\_id, appointment\_id, record\_date, record\_details)

VALUES (p\_record\_id, p\_appointment\_id, p\_record\_date, p\_record\_details);

-- Commit the changes

COMMIT;

-- Display a message indicating successful insertion

DBMS\_OUTPUT.PUT\_LINE('Medical report added successfully!');

EXCEPTION

WHEN OTHERS THEN

-- Display an error message if an exception occurs

DBMS\_OUTPUT.PUT\_LINE('Error: ' || SQLERRM);

ROLLBACK;

END;

/

BEGIN

add\_patient(1,'Mahesh','11-FEB-2000','M','7589461236','Mohali');

add\_patient(2,'Suresh','10-JAN-1998','M','5896412368','Patiala');

add\_patient(3,'Sheela','01-NOV-2002','F','6258974131','Lucknow') ;

add\_doctor(1, 'Dr. John Doe', 'john.doe@example.com', '555-555-1234');

add\_doctor(2,'Himesh','himesh1@gmail.com','8974561256');

add\_doctor(3,'Sonia','soni5@gmail.com','9658741288') ;

add\_speciality(1,'Cardiology');

add\_speciality(2,'ENT');

add\_speciality(3,'Brain');

add\_doctor\_speciality(1, 1);

-- Adding doctor with doctor\_id = 1 to speciality with speciality\_id = 1

add\_doctor\_speciality(2, 3);

add\_doctor\_speciality(3, 2);

add\_appointment(1, '01-JAN-2023', 1, 1, 'Follow-up appointment');

add\_medical\_report( 1,1,TO\_DATE('2023-04-16', 'YYYY-MM-DD'),'Patient had a check-up, prescribed medication for common cold.');

select \* from patients;

select \* from doctors;

select \* from specialities;

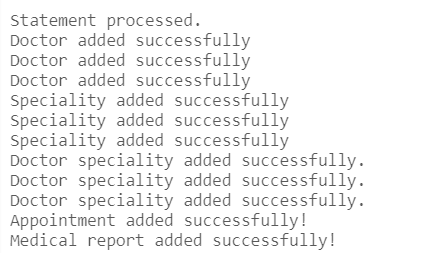
select \* from doctors\_specialities;

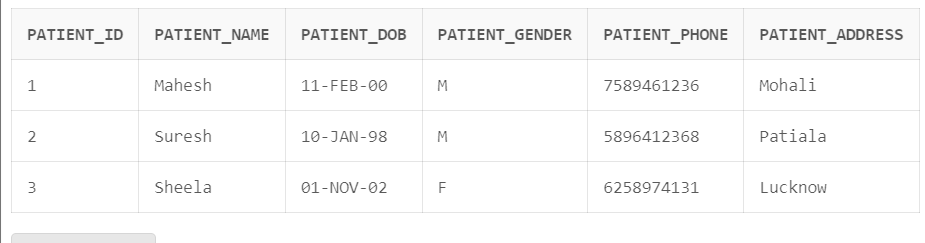
select \* from appointments;

select \* from medical\_report;

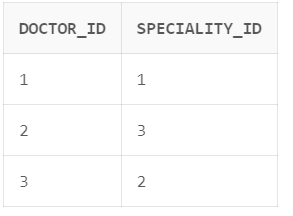
END;

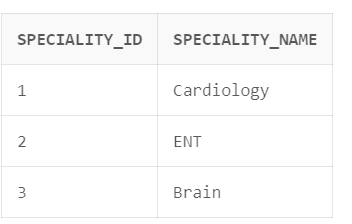
/

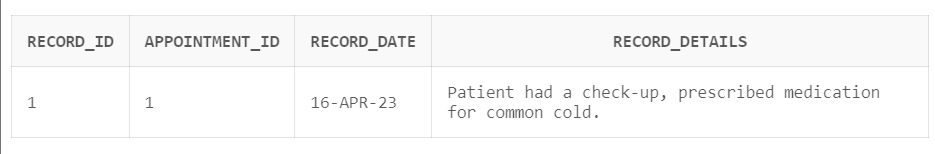
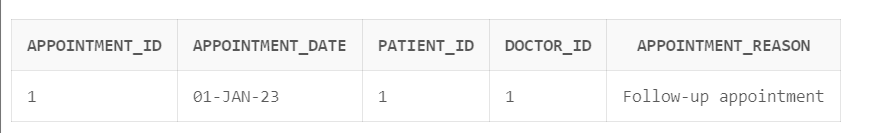












**Function to Get Doctor name From Specialization name**

CREATE OR REPLACE FUNCTION get\_doctor\_name\_from\_specialization(p\_specialization\_name IN VARCHAR2)

RETURN VARCHAR2

IS

v\_doctor\_name doctors.doctor\_name%TYPE;

BEGIN

SELECT d.doctor\_name

INTO v\_doctor\_name

FROM doctors d

INNER JOIN doctors\_specialities ds ON d.doctor\_id = ds.doctor\_id

INNER JOIN specialities s ON ds.speciality\_id = s.speciality\_id

WHERE s.speciality\_name = p\_specialization\_name;

RETURN v\_doctor\_name;

EXCEPTION

WHEN NO\_DATA\_FOUND THEN

RETURN NULL;

WHEN OTHERS THEN

RAISE;

END;

/

DECLARE

v\_doctor\_name doctors.doctor\_name%TYPE;

BEGIN

v\_doctor\_name := get\_doctor\_name\_from\_specialization('Cardiology');

IF v\_doctor\_name IS NOT NULL THEN

DBMS\_OUTPUT.PUT\_LINE('Doctor Name: ' || v\_doctor\_name);

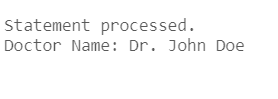
ELSE

DBMS\_OUTPUT.PUT\_LINE('No doctor found for the given specialization name.');

END IF;

END;

/



**Function to retrieve medical\_report from appointment\_Id**

CREATE OR REPLACE FUNCTION get\_medical\_record(p\_appointment\_id IN appointments.appointment\_id%TYPE)

RETURN medical\_report%ROWTYPE

IS

v\_medical\_record medical\_report%ROWTYPE;

BEGIN

-- Query the medical\_report table for the given appointment ID

SELECT \*

INTO v\_medical\_record

FROM medical\_report

WHERE appointment\_id = p\_appointment\_id;

RETURN v\_medical\_record;

EXCEPTION

WHEN NO\_DATA\_FOUND THEN

-- Handle the case when no medical record is found for the given appointment ID

RAISE\_APPLICATION\_ERROR(-20001, 'Medical record not found for the given appointment ID');

END;

/

DECLARE

v\_medical\_record medical\_report%ROWTYPE;

BEGIN

-- Call the function to get the medical record for appointment ID 12345

v\_medical\_record := get\_medical\_record(1);

-- Display the medical record details

DBMS\_OUTPUT.PUT\_LINE('Record ID: ' || v\_medical\_record.record\_id);

DBMS\_OUTPUT.PUT\_LINE('Appointment ID: ' || v\_medical\_record.appointment\_id);

DBMS\_OUTPUT.PUT\_LINE('Record Date: ' || v\_medical\_record.record\_date);

DBMS\_OUTPUT.PUT\_LINE('Record Details: ' || v\_medical\_record.record\_details);

EXCEPTION

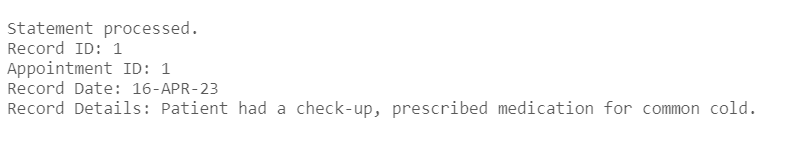
WHEN OTHERS THEN

-- Handle any exceptions that may occur

DBMS\_OUTPUT.PUT\_LINE('Error: ' || SQLERRM);

END;

/



**Trigger to check slot before booking appointment**

CREATE OR REPLACE TRIGGER check\_appointment

BEFORE INSERT ON appointments

FOR EACH ROW

DECLARE

count\_appt NUMBER;

BEGIN

SELECT COUNT(\*) INTO count\_appt

FROM appointments

WHERE patient\_id = :new.patient\_id

AND doctor\_id = :new.doctor\_id

AND TRUNC(appointment\_date) = TRUNC(:new.appointment\_date);

IF count\_appt > 0 THEN

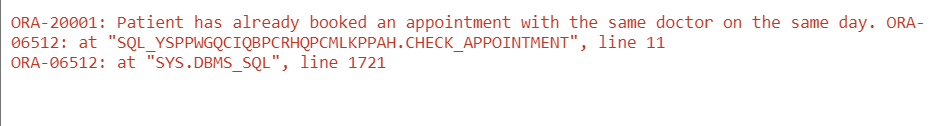
RAISE\_APPLICATION\_ERROR(-20001, 'Patient has already booked an appointment with the same doctor on the same day.');

END IF;

END;

/

insert into appointments values(1, '01-JAN-2023', 1, 1, 'Follow-up appointment');



**References**

www.youtube.com/parteekBhatia

**Conclusion**

In conclusion, the healthcare management system project requires the development of a relational database management system (DBMS) that can efficiently store and manage patient information, doctor information, appointments, and medical reports. The system should be able to enforce referential integrity, perform CRUD operations, and provide appropriate error handling, data validation, and data integrity checks.