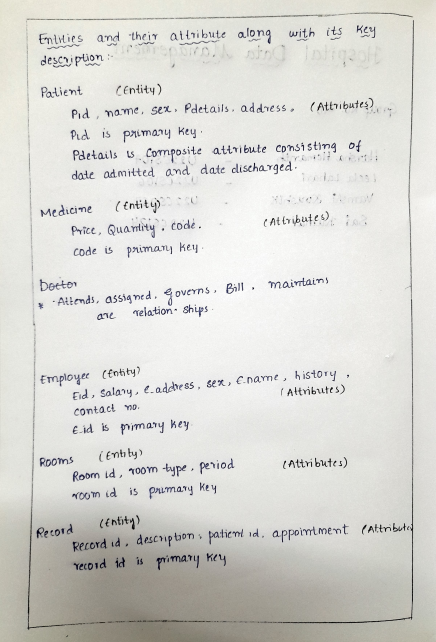
**MINI PROJECT**

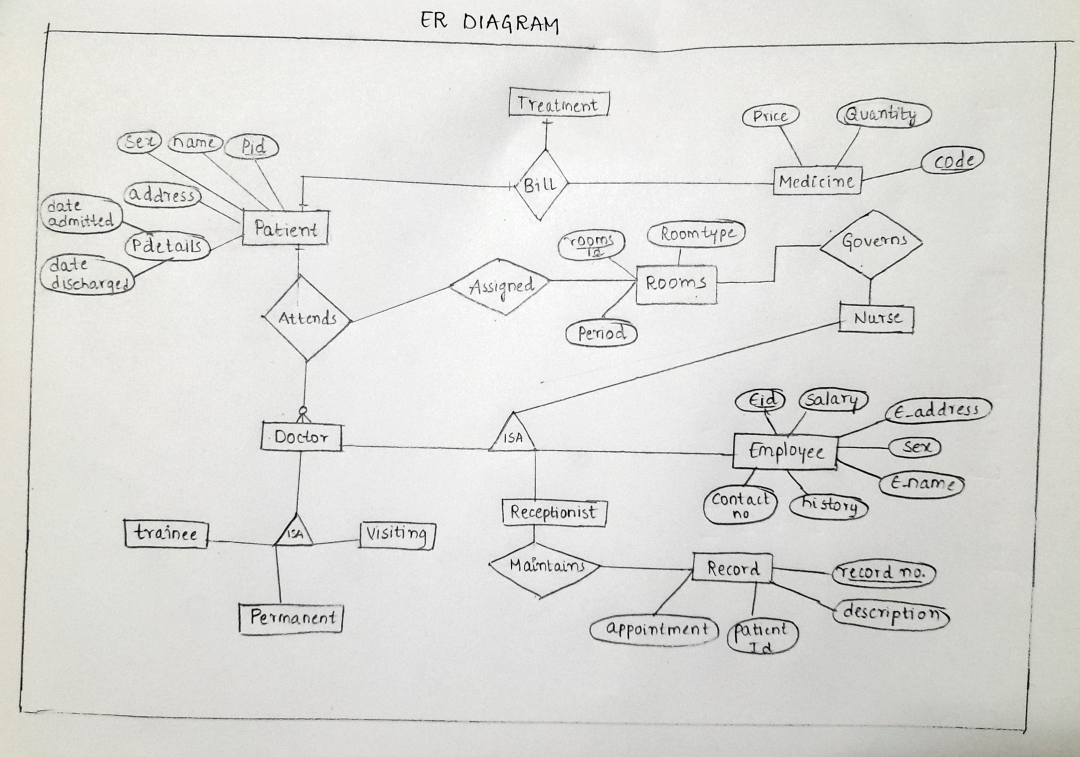
group no – 10

|  |  |  |
| --- | --- | --- |
| S.NO | NAME | ADM . NO |
| 1. | SAI KUMAR | U22CS124 |
| 2. | HARSHA HEMANTH | U22CS107 |
| 3. | LEELA LAHARI | U22CS106 |
| 4. | VAMSHI KOUSHIK | U22CS087 |

HOSPITAL DATA MANAGEMENT

E-R DIAGRAM





NORMALISATION :

normalization form for the tables :

1. Medicine Table:

- Code is the primary key, and it uniquely identifies each row.

- The table does not contain any repeating groups.

- All columns contain atomic (indivisible) values.

This table is in 1NF and is already in 2NF and 3NF since it doesn't have any partial or transitive dependencies.

2. Employee Table:

- Eid is the primary key, and it uniquely identifies each row.

- The table does not contain any repeating groups.

- All columns contain atomic values.

This table is in 1NF and is already in 2NF and 3NF since it doesn't have any partial or transitive dependencies.

3. Rooms Table:

- RoomID is the primary key, and it uniquely identifies each row.

- The table does not contain any repeating groups.

- All columns contain atomic values.

This table is in 1NF and is already in 2NF and 3NF since it doesn't have any partial or transitive dependencies.

4. Patients Table:

- Pid is the primary key, and it uniquely identifies each row.

- The table contains foreign keys RoomID and Eid, which reference the Rooms and Employee tables.

- All columns contain atomic values.

This table is in 1NF and is in 2NF and 3NF since the foreign keys create relationships, and there are no partial or transitive dependencies within the table itself.

5. Record Table:

- RecordNo is the primary key, and it uniquely identifies each row.

- The table contains a foreign key Pid, which references the Patients table.

- All columns contain atomic values.

This table is in 1NF and is in 2NF and 3NF since the foreign key creates a relationship, and there are no partial or transitive dependencies within the table itself.

6. Bill Table:

- BillID is the primary key, and it uniquely identifies each row.

- The table contains two foreign keys: Pid (referencing Patients) and Code (referencing Medicine).

- All columns contain atomic values.

This table is in 1NF and is in 2NF and 3NF since the foreign keys create relationships, and there are no partial or transitive dependencies within the table itself.

7. DoctorRelationship Table:

- DoctorID is the primary key, and it uniquely identifies each row.

- The table contains a foreign key Eid, which references the Employee table.

- All columns contain atomic values.

This table is in 1NF and is in 2NF and 3NF since the foreign key creates a relationship, and there are no partial or transitive dependencies within the table itself.

8. RoomGovernsNurse Table:

- The table contains two foreign keys: RoomID (referencing Rooms) and NurseID (referencing Employee).

- All columns contain atomic values.

This table is in 1NF and is in 2NF and 3NF since the foreign keys create relationships, and there are no partial or transitive dependencies within the table itself.

9. Attends Table:

- AppointmentID is the primary key, and it uniquely identifies each row.

- The table contains two foreign keys: Pid (referencing Patients) and DoctorID (referencing Employee).

- All columns contain atomic values.

This table is in 1NF and is in 2NF and 3NF since the foreign keys create relationships, and there are no partial or transitive dependencies within the table itself.

10. EmployeeTitle Table:

- TitleID is the primary key, and it uniquely identifies each row.

- The table does not contain any repeating groups.

- All columns contain atomic values.

This table is in 1NF and is already in 2NF and 3NF since it doesn't have any partial or transitive dependencies.

11. EmployeeIsA Table:

- The table contains two foreign keys: Eid (referencing Employee) and TitleID (referencing EmployeeTitle).

- All columns contain atomic values.

This table is in 1NF and is in 2NF and 3NF since the foreign keys create relationships, and there are no partial or transitive dependencies within the table itself.

12. ReceptionistMaintainsRecord Table:

- The table contains two foreign keys: Eid (referencing Employee) and RecordNo (referencing Record).

- All columns contain atomic values.

This table is in 1NF and is in 2NF and 3NF since the foreign keys create relationships, and there are no partial or transitive dependencies within the table itself.

In summary, all the provided tables appear to be in at least First Normal Form (1NF), and they are also in Second Normal Form (2NF) and Third Normal Form (3NF). There are no apparent violations of these normal forms in the provided table structures.

PL/SQL Scripting

**Code:**

BEGIN

    -- Create Medicine table

    EXECUTE IMMEDIATE 'CREATE TABLE Medicine (

        Code INT PRIMARY KEY,

        Name VARCHAR(8),

        Price DECIMAL(5, 2),

        Quantity INT

    )';

    -- Create Employee table

    EXECUTE IMMEDIATE 'CREATE TABLE Employee (

        Eid INT PRIMARY KEY,

        EName VARCHAR(8),

        Sex VARCHAR(5),

        EAddress VARCHAR(7),

        Salary DECIMAL(5, 2),

        History CLOB,

        ContactNo VARCHAR(3)

    )';

    -- Create Rooms table

    EXECUTE IMMEDIATE 'CREATE TABLE Rooms (

        RoomID INT PRIMARY KEY,

        RoomType VARCHAR(7),

        Period VARCHAR(7)

    )';

    -- Create Patients table

    EXECUTE IMMEDIATE 'CREATE TABLE Patients (

        Pid INT PRIMARY KEY,

        Name VARCHAR(8),

        Sex VARCHAR(5),

        Address VARCHAR(8),

        DateAdmitted DATE,

        DateDischarged DATE,

        RoomID INT,

        Eid INT,

        FOREIGN KEY (RoomID) REFERENCES Rooms(RoomID),

        FOREIGN KEY (Eid) REFERENCES Employee(Eid)

    )';

    -- Create Record table

    EXECUTE IMMEDIATE 'CREATE TABLE Record (

        RecordNo INT PRIMARY KEY,

        Description CLOB,

        Pid INT,

        Appointment DATE,

        FOREIGN KEY (Pid) REFERENCES Patients(Pid)

    )';

    -- Create Bill table

    EXECUTE IMMEDIATE 'CREATE TABLE Bill (

        BillID INT PRIMARY KEY,

        Pid INT,

        Code INT,

        Quantity INT,

        BillDate DATE,

        FOREIGN KEY (Pid) REFERENCES Patients(Pid),

        FOREIGN KEY (Code) REFERENCES Medicine(Code)

    )';

    -- Create DoctorRelationship table

    EXECUTE IMMEDIATE 'CREATE TABLE DoctorRelationship (

        DoctorID INT PRIMARY KEY,

        Type VARCHAR(6),

        Eid INT,

        FOREIGN KEY (Eid) REFERENCES Employee(Eid)

    )';

    -- Create RoomGovernsNurse table

    EXECUTE IMMEDIATE 'CREATE TABLE RoomGovernsNurse (

        RoomID INT,

        NurseID INT,

        FOREIGN KEY (RoomID) REFERENCES Rooms(RoomID),

        FOREIGN KEY (NurseID) REFERENCES Employee(Eid)

    )';

    -- Create Attends table

    EXECUTE IMMEDIATE 'CREATE TABLE Attends (

        AppointmentID INT PRIMARY KEY,

        Pid INT,

        DoctorID INT,

        AppointmentDate DATE,

        FOREIGN KEY (Pid) REFERENCES Patients(Pid),

        FOREIGN KEY (DoctorID) REFERENCES Employee(Eid)

    )';

    -- Create EmployeeTitle table

    EXECUTE IMMEDIATE 'CREATE TABLE EmployeeTitle (

        TitleID INT PRIMARY KEY,

        Title VARCHAR(7)

    )';

    -- Create EmployeeIsA table

    EXECUTE IMMEDIATE 'CREATE TABLE EmployeeIsA (

        Eid INT,

        TitleID INT,

        FOREIGN KEY (Eid) REFERENCES Employee(Eid),

        FOREIGN KEY (TitleID) REFERENCES EmployeeTitle(TitleID)

    )';

    -- Create ReceptionistMaintainsRecord table

    EXECUTE IMMEDIATE 'CREATE TABLE ReceptionistMaintainsRecord (

        Eid INT,

        RecordNo INT,

        FOREIGN KEY (Eid) REFERENCES Employee(Eid),

        FOREIGN KEY (RecordNo) REFERENCES Record(RecordNo)

    )';

    COMMIT;

END;

/

-- Create procedures using PL/SQL

CREATE OR REPLACE PROCEDURE InsertMedicine (

    p\_Code INT,

    p\_Name VARCHAR2,

    p\_Price DECIMAL,

    p\_Quantity INT

) AS

BEGIN

    INSERT INTO Medicine (Code, Name, Price, Quantity)

    VALUES (p\_Code,p\_name,p\_price,p\_quantity);

END;

/

-- Procedure to insert data into the Employee table

CREATE OR REPLACE PROCEDURE InsertEmployee (

    p\_Eid INT,

    p\_EName VARCHAR2,

    p\_Sex VARCHAR2,

    p\_EAddress VARCHAR2,

    p\_Salary NUMBER,

    p\_History CLOB,

    p\_ContactNo VARCHAR2

) AS

BEGIN

    INSERT INTO Employee (Eid, EName, Sex, EAddress, Salary, History, ContactNo)

    VALUES (p\_Eid, p\_EName, p\_Sex, p\_EAddress, p\_Salary, p\_History, p\_ContactNo);

    COMMIT;

END;

/

-- Procedure to insert data into the Rooms table

CREATE OR REPLACE PROCEDURE InsertRoom (

    p\_RoomID INT,

    p\_RoomType VARCHAR2,

    p\_Period VARCHAR2

) AS

BEGIN

    INSERT INTO Rooms (RoomID, RoomType, Period)

    VALUES (p\_RoomID, p\_RoomType, p\_Period);

    COMMIT;

END;

/

-- Procedure to insert data into the Patients table

CREATE OR REPLACE PROCEDURE InsertPatient (

    p\_Pid INT,

    p\_Name VARCHAR2,

    p\_Sex VARCHAR2,

    p\_Address VARCHAR2,

    p\_DateAdmitted DATE,

    p\_DateDischarged DATE,

    p\_RoomID INT,

    p\_Eid INT

) AS

BEGIN

    INSERT INTO Patients (Pid, Name, Sex, Address, DateAdmitted, DateDischarged, RoomID, Eid)

    VALUES (p\_Pid, p\_Name, p\_Sex, p\_Address, p\_DateAdmitted, p\_DateDischarged, p\_RoomID, p\_Eid);

    COMMIT;

END;

/

-- Procedure to insert data into the Record table

CREATE OR REPLACE PROCEDURE InsertRecord (

    p\_RecordNo INT,

    p\_Description CLOB,

    p\_Pid INT,

    p\_Appointment DATE

) AS

BEGIN

    INSERT INTO Record (RecordNo, Description, Pid, Appointment)

    VALUES (p\_RecordNo, p\_Description, p\_Pid, p\_Appointment);

    COMMIT;

END;

/

-- Procedure to insert data into the Bill table

CREATE OR REPLACE PROCEDURE InsertBill (

    p\_BillID INT,

    p\_Pid INT,

    p\_Code INT,

    p\_Quantity INT,

    p\_BillDate DATE

) AS

BEGIN

    INSERT INTO Bill (BillID, Pid, Code, Quantity, BillDate)

    VALUES (p\_BillID, p\_Pid, p\_Code, p\_Quantity, p\_BillDate);

    COMMIT;

END;

/

-- Procedure to insert data into the DoctorRelationship table

CREATE OR REPLACE PROCEDURE InsertDoctorRelationship (

    p\_DoctorID INT,

    p\_Type VARCHAR2,

    p\_Eid INT

) AS

BEGIN

    INSERT INTO DoctorRelationship (DoctorID, Type, Eid)

    VALUES (p\_DoctorID, p\_Type, p\_Eid);

    COMMIT;

END;

/

-- Procedure to insert data into the RoomGovernsNurse table

CREATE OR REPLACE PROCEDURE InsertRoomGovernsNurse (

    p\_RoomID INT,

    p\_NurseID INT

) AS

BEGIN

    INSERT INTO RoomGovernsNurse (RoomID, NurseID)

    VALUES (p\_RoomID, p\_NurseID);

    COMMIT;

END;

/

-- Procedure to insert data into the Attends table

CREATE OR REPLACE PROCEDURE InsertAttends (

    p\_AppointmentID INT,

    p\_Pid INT,

    p\_DoctorID INT,

    p\_AppointmentDate DATE

) AS

BEGIN

    INSERT INTO Attends (AppointmentID, Pid, DoctorID, AppointmentDate)

    VALUES (p\_AppointmentID, p\_Pid, p\_DoctorID, p\_AppointmentDate);

    COMMIT;

END;

/

-- Procedure to insert data into the EmployeeTitle table

CREATE OR REPLACE PROCEDURE InsertEmployeeTitle (

    p\_TitleID INT,

    p\_Title VARCHAR2

) AS

BEGIN

    INSERT INTO EmployeeTitle (TitleID, Title)

    VALUES (p\_TitleID, p\_Title);

    COMMIT;

END;

/

-- Procedure to insert data into the DoctorRelationship table

CREATE OR REPLACE PROCEDURE InsertDoctorRelationship (

    p\_DoctorID INT,

    p\_Type VARCHAR2,

    p\_Eid INT

) AS

BEGIN

    INSERT INTO DoctorRelationship (DoctorID, Type, Eid)

    VALUES (p\_DoctorID, p\_Type, p\_Eid);

    COMMIT;

END;

/

-- Procedure to insert data into the RoomGovernsNurse table

CREATE OR REPLACE PROCEDURE InsertRoomGovernsNurse (

    p\_RoomID INT,

    p\_NurseID INT

) AS

BEGIN

    INSERT INTO RoomGovernsNurse (RoomID, NurseID)

    VALUES (p\_RoomID, p\_NurseID);

    COMMIT;

END;

/

-- Procedure to insert data into the Attends table

CREATE OR REPLACE PROCEDURE InsertAttends (

    p\_AppointmentID INT,

    p\_Pid INT,

    p\_DoctorID INT,

    p\_AppointmentDate DATE

) AS

BEGIN

    INSERT INTO Attends (AppointmentID, Pid, DoctorID, AppointmentDate)

    VALUES (p\_AppointmentID, p\_Pid, p\_DoctorID, p\_AppointmentDate);

    COMMIT;

END;

/

-- Procedure to insert data into the EmployeeTitle table

CREATE OR REPLACE PROCEDURE InsertEmployeeTitle (

    p\_TitleID INT,

    p\_Title VARCHAR2

) AS

BEGIN

    INSERT INTO EmployeeTitle (TitleID, Title)

    VALUES (p\_TitleID, p\_Title);

    COMMIT;

END;

/

-- Procedure to insert data into the EmployeeIsA table

CREATE OR REPLACE PROCEDURE InsertEmployeeIsA (

    p\_Eid INT,

    p\_TitleID INT

) AS

BEGIN

    INSERT INTO EmployeeIsA (Eid, TitleID)

    VALUES (p\_Eid, p\_TitleID);

    COMMIT;

END;

/

-- Procedure to insert data into the ReceptionistMaintainsRecord table

CREATE OR REPLACE PROCEDURE InsertRMR (

    p\_Eid INT,

    p\_RecordNo INT

) AS

BEGIN

    INSERT INTO  ReceptionistMaintainsRecord(Eid, RecordNo)

    VALUES (p\_Eid, p\_RecordNo);

    COMMIT;

END;

/

-- Insert data into the Medicine table

BEGIN

    InsertMedicine(1, 'M1', 10.99, 100);

    InsertMedicine(2, 'M2', 15.99, 50);

    InsertMedicine(3, 'M3', 8.99, 75);

END;

/

-- Insert data into the Employee table

BEGIN

     InsertEmployee(1, 'John', 'M', 'Main', 50.0, 'y', '123');

    InsertEmployee(2, 'Jane', 'F', 'Elm', 87.0, 'r', '987');

    InsertEmployee(3, 'lah', 'M', 'Oak', 48.9, 'm', '555');

END;

/

-- Insert data into the Rooms table

BEGIN

    InsertRoom(1, 'Single', '24 Hr');

    InsertRoom(2, 'Double', '48 Hr');

    InsertRoom(3, 'Suite', '72 Hr');

END;

/

-- Insert data into the Patients table

BEGIN

    InsertPatient(1, 'hars', 'Male', 'YUI','02-JAN-23','02-JAN-23', 1, 1);

    InsertPatient(2, 'op', 'F', 'Oak St','17-FEB-23','17-JUN-24', 2, 2);

    InsertPatient(3, 'sai', 'M', 'Elm St', '12-MAR-23','15-JUL-23', 3, 3);

END;

/

-- Insert data into the Record table

BEGIN

    InsertRecord(1, 'r1', 1, '02-JAN-23');

    InsertRecord(2, 'r2', 2, '17-FEB-23');

    InsertRecord(3, 'r3', 3, '12-MAR-23');

END;

/

-- Insert data into the Bill table

BEGIN

    InsertBill(1, 1, 1, 2,'02-JAN-23');

    InsertBill(2, 2, 2, 3,'17-JUN-24');

    InsertBill(3, 3, 3, 1, '23-JUL-23');

END;

/

-- Insert data into the EmployeeTitle table using the InsertEmployeeTitle procedure

BEGIN

    InsertEmployeeTitle(1, 'Title1');

    InsertEmployeeTitle(2, 'Title2');

    InsertEmployeeTitle(3, 'Title3');

END;

/

BEGIN

    -- Modify the values to match your data

    InsertDoctorRelationship(1, 'Visit', 1);

    InsertDoctorRelationship(2, 'Traine', 2);

    InsertDoctorRelationship(3, 'PMT', 3);

END;

/

-- Insert data into the RoomGovernsNurse table

BEGIN

    -- Modify the values to match your data

    InsertRoomGovernsNurse(1, 1);

    InsertRoomGovernsNurse(2, 2);

    InsertRoomGovernsNurse(3, 3);

END;

/

-- Insert data into the Attends table

BEGIN

    -- Modify the values to match your data

    InsertAttends(1, 1, 1,'01-JAN-23');

    InsertAttends(2, 2, 2,'09-JUN-23');

    InsertAttends(3, 3, 3, '22-JUN-22');

END;

/

-- Insert data into the EmployeeIsA table

BEGIN

    -- Modify the values to match your data

    InsertEmployeeIsA(1, 1);

    InsertEmployeeIsA(2, 2);

    InsertEmployeeIsA(3, 3);

END;

/

-- Insert data into the ReceptionistMaintainsRecord table

BEGIN

    -- Modify the values to match your data

    InsertRMR(1, 1);

    InsertRMR(2, 2);

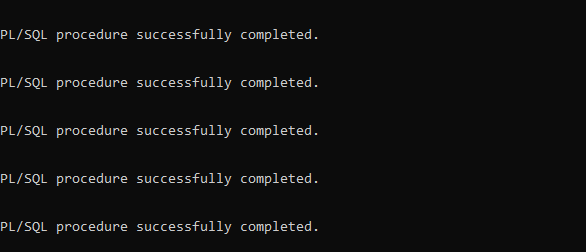
    InsertRMR(3, 3);

END;

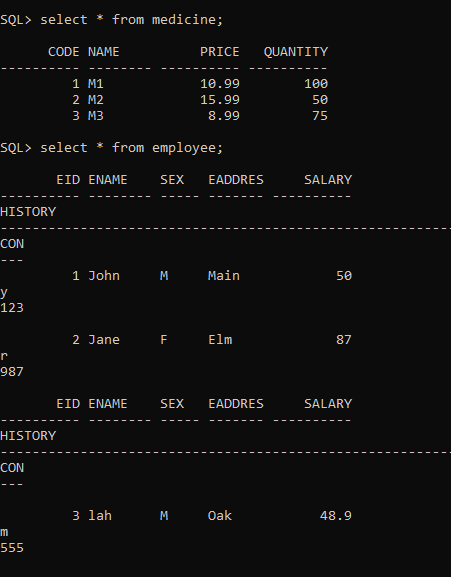
/

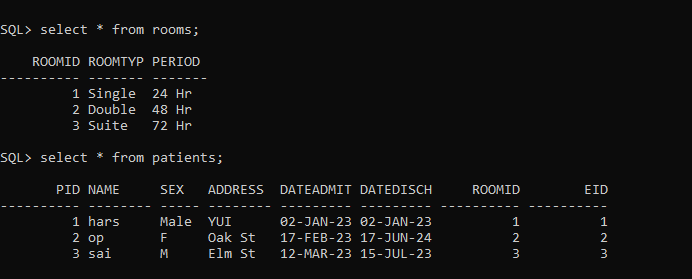
**OUTPUT:**

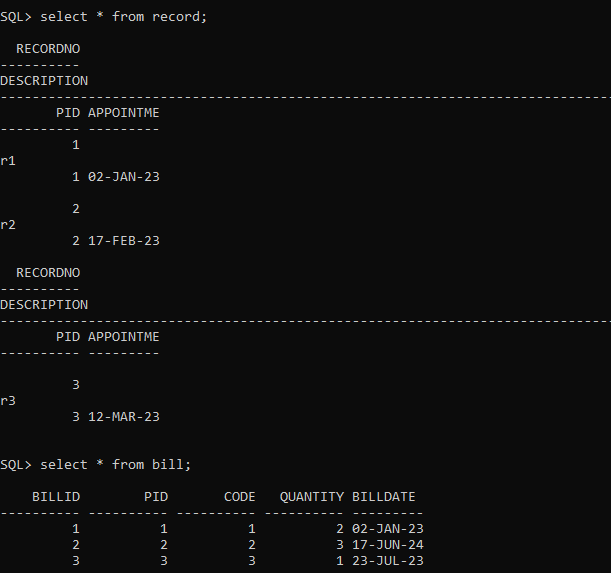
PROCEDURES

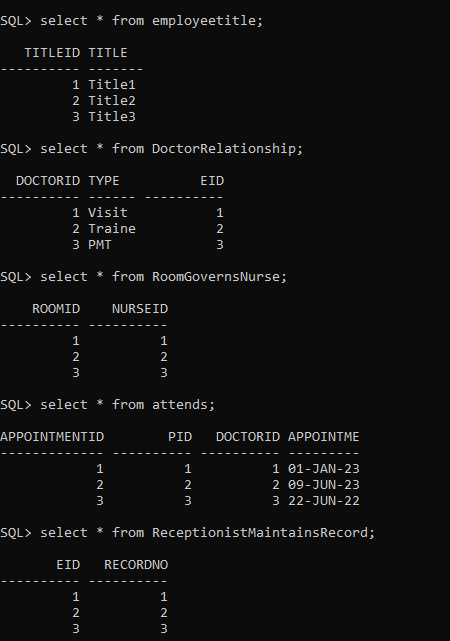


OUTPUTS OF ALL THE TABLES :









FOR CURSORS CODE + OUTPUT:

ON MEDICINE TABLE:

DECLARE

CURSOR medicine\_cursor IS

SELECT Code, Name, Price, Quantity

FROM Medicine;

-- Declare variables to hold column values

v\_Code Medicine.Code%TYPE;

v\_Name Medicine.Name%TYPE;

v\_Price Medicine.Price%TYPE;

v\_Quantity Medicine.Quantity%TYPE;

BEGIN

OPEN medicine\_cursor;

LOOP

FETCH medicine\_cursor INTO v\_Code, v\_Name, v\_Price, v\_Quantity;

EXIT WHEN medicine\_cursor%NOTFOUND;

-- Process or display the retrieved data as needed

DBMS\_OUTPUT.PUT\_LINE('Medicine Code: ' || v\_Code);

DBMS\_OUTPUT.PUT\_LINE('Medicine Name: ' || v\_Name);

DBMS\_OUTPUT.PUT\_LINE('Price: ' || v\_Price);

DBMS\_OUTPUT.PUT\_LINE('Quantity: ' || v\_Quantity);

DBMS\_OUTPUT.PUT\_LINE('--------------------------');

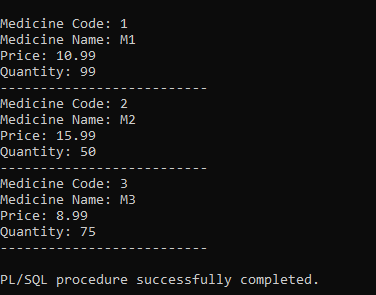
END LOOP;

CLOSE medicine\_cursor;

END;

/

OUTPUT:



ON EMPLOYEE TABLE :

DECLARE

CURSOR employee\_cursor IS

SELECT Eid, EName, Sex, EAddress, Salary, History, ContactNo

FROM Employee;

v\_Eid Employee.Eid%TYPE;

v\_EName Employee.EName%TYPE;

v\_Sex Employee.Sex%TYPE;

v\_EAddress Employee.EAddress%TYPE;

v\_Salary Employee.Salary%TYPE;

v\_History Employee.History%TYPE;

v\_ContactNo Employee.ContactNo%TYPE;

BEGIN

OPEN employee\_cursor;

LOOP

FETCH employee\_cursor INTO v\_Eid, v\_EName, v\_Sex, v\_EAddress, v\_Salary, v\_History, v\_ContactNo;

EXIT WHEN employee\_cursor%NOTFOUND;

DBMS\_OUTPUT.PUT\_LINE('Employee ID: ' || v\_Eid);

DBMS\_OUTPUT.PUT\_LINE('Employee Name: ' || v\_EName);

DBMS\_OUTPUT.PUT\_LINE('Sex: ' || v\_Sex);

DBMS\_OUTPUT.PUT\_LINE('Address: ' || v\_EAddress);

DBMS\_OUTPUT.PUT\_LINE('Salary: ' || v\_Salary);

DBMS\_OUTPUT.PUT\_LINE('History: ' || v\_History);

DBMS\_OUTPUT.PUT\_LINE('Contact No: ' || v\_ContactNo);

DBMS\_OUTPUT.PUT\_LINE('--------------------------');

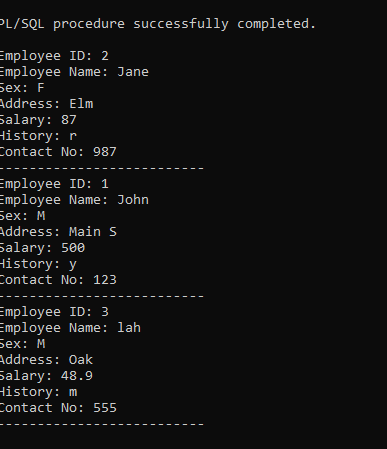
END LOOP;

CLOSE employee\_cursor;

END;

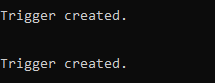
/

OUTPUT:



FOR TRIGGERS CODE + OUTPUT:

OUTPUTS FOR TRIGGERS :



CREATE OR REPLACE TRIGGER UpdatePatientDischargeDate

AFTER INSERT ON Record

FOR EACH ROW

BEGIN

UPDATE Patients

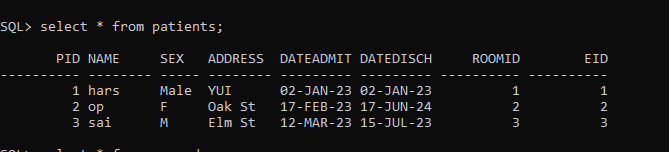
SET DateDischarged = :new.Appointment

WHERE Pid = :new.Pid;

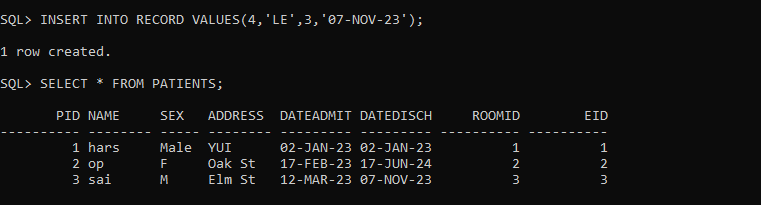
END;

/

**PATIENTS TABLE BEFORE INSERTING RECORD VALUES:**



**PATIENTS TABLE AFTER INSERTING RECORD VALUES:**



2.

CREATE OR REPLACE TRIGGER UpdateMedicineQuantity

AFTER INSERT ON Bill

FOR EACH ROW

BEGIN

UPDATE Medicine

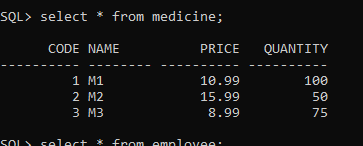
SET Quantity = Quantity - :new.Quantity

WHERE Code = :new.Code;

END;

/

**MEDICINE TABLE BEFORE INSERTING BILL VALUES:**



**MEDICINE TABLE AFTER INSERTING BILL VALUES:**

