Lab on DBMS answer pdf as per slip sheets list 1 to 26

1)

# Create Tables with Integrity Constraints

CREATE TABLE SUPPLIER (

Sno VARCHAR(5) PRIMARY KEY CHECK (Sno LIKE 'S%' AND CAST(SUBSTRING(Sno, 2) AS INT) BETWEEN 0 AND 9999),

Sname VARCHAR(50) NOT NULL, Address VARCHAR(100) NOT NULL,

City VARCHAR(20) NOT NULL CHECK (City IN ('London', 'Paris', 'Rome', 'New York', 'Amsterdam'))

);

CREATE TABLE PARTS (

Pno VARCHAR(5) PRIMARY KEY, Pname VARCHAR(50) NOT NULL, Color VARCHAR(20) NOT NULL, Weight DECIMAL(6,2) NOT NULL, Price DECIMAL(10,2) NOT NULL

);

CREATE TABLE PROJECT (

Jno VARCHAR(5) PRIMARY KEY, Jname VARCHAR(50) NOT NULL,

City VARCHAR(20) NOT NULL CHECK (City IN ('London', 'Paris', 'Rome', 'New York', 'Amsterdam'))

);

CREATE TABLE SPJ ( Sno VARCHAR(5), Pno VARCHAR(5), Jno VARCHAR(5), Qty INT NOT NULL,

PRIMARY KEY (Sno, Pno, Jno),

FOREIGN KEY (Sno) REFERENCES SUPPLIER(Sno), FOREIGN KEY (Pno) REFERENCES PARTS(Pno), FOREIGN KEY (Jno) REFERENCES PROJECT(Jno)

);

# ◻ 2. Insert Sample Data (10+ Records)

-- Suppliers

INSERT INTO SUPPLIER VALUES

('S001', 'Alpha Supplies', '123 Alpha St', 'London'), ('S002', 'Beta Traders', '456 Beta St', 'Paris'),

('S003', 'Gamma Goods', '789 Gamma Blvd', 'Rome'),

('S004', 'Delta Corp', '111 Delta Ln', 'New York'), ('S005', 'Epsilon LLC', '222 Epsilon Rd', 'Amsterdam'), ('S006', 'Zeta Ltd.', '333 Zeta Pl', 'London'),

('S007', 'Eta Supplies', '444 Eta St', 'Paris'),

('S008', 'Theta Inc.', '555 Theta Dr', 'Rome'),

('S009', 'Iota Partners', '666 Iota Blvd', 'New York'), ('S010', 'Kappa Co.', '777 Kappa Way', 'Amsterdam');

-- Parts

INSERT INTO PARTS VALUES

('P001', 'Bolt', 'Red', 1.25, 2.50),

('P002', 'Nut', 'Blue', 0.75, 1.25),

('P003', 'Screw', 'Green', 0.50, 1.00),

('P004', 'Washer', 'Red', 0.20, 0.80),

('P005', 'Pin', 'Yellow', 0.30, 0.60),

('P006', 'Bracket', 'Black', 2.00, 4.00),

('P007', 'Clamp', 'Silver', 3.00, 5.50),

('P008', 'Rod', 'Grey', 5.00, 6.25),

('P009', 'Pipe', 'White', 10.00, 8.00),

('P010', 'Valve', 'Orange', 1.10, 7.00);

-- Projects

INSERT INTO PROJECT VALUES

('J001', 'Bridge Build', 'London'),

('J002', 'Road Expansion', 'Paris'),

('J003', 'Metro Construction', 'Rome'), ('J004', 'Mall Development', 'New York'), ('J005', 'Tunnel Project', 'Amsterdam'), ('J006', 'Park Renovation', 'Paris'),

('J007', 'Hospital Build', 'London'),

('J008', 'Airport Extension', 'Rome'),

('J009', 'Harbor Setup', 'New York'),

('J010', 'Water Dam', 'Paris');

-- SPJ Relationships INSERT INTO SPJ VALUES

|  |  |  |  |
| --- | --- | --- | --- |
| ('S001', | 'P001', | 'J001', | 100), |
| ('S002', | 'P002', | 'J002', | 200), |
| ('S003', | 'P003', | 'J003', | 150), |
| ('S004', | 'P004', | 'J004', | 80), |
| ('S005', | 'P005', | 'J005', | 120), |
| ('S001', | 'P006', | 'J001', | 60), |
| ('S002', | 'P007', | 'J002', | 90), |
| ('S006', | 'P008', | 'J006', | 70), |
| ('S007', | 'P009', | 'J010', | 110), |
| ('S008', | 'P010', | 'J003', | 100), |
| ('S009', | 'P001', | 'J007', | 50), |
| ('S010', | 'P002', | 'J006', | 90); |

# ◻ 3. Queries

### Projects with 3 or More Parts

SELECT Jno, COUNT(DISTINCT Pno) AS PartCount FROM SPJ

GROUP BY Jno

HAVING COUNT(DISTINCT Pno) >= 3;

1. **Trigger: Jname Must Be Unique**

CREATE OR REPLACE FUNCTION check\_unique\_jname() RETURNS TRIGGER AS $$

BEGIN

IF EXISTS (SELECT 1 FROM PROJECT WHERE Jname = NEW.Jname AND Jno <> NEW.Jno) THEN

RAISE EXCEPTION 'Project name "%s" already exists.', NEW.Jname;

END IF;

RETURN NEW;

END;

$$ LANGUAGE plpgsql;

CREATE TRIGGER trg\_unique\_jname BEFORE INSERT OR UPDATE ON PROJECT FOR EACH ROW

EXECUTE FUNCTION check\_unique\_jname();

### Full Details of All Projects in London

SELECT \* FROM PROJECT WHERE City = 'London';

### Procedure to Calculate Total Sales of Parts to Paris Projects

CREATE OR REPLACE PROCEDURE calc\_total\_sales\_paris() LANGUAGE plpgsql

AS $$ DECLARE

total\_sales NUMERIC := 0; BEGIN

SELECT SUM(p.Price \* s.Qty) INTO total\_sales

FROM SPJ s

JOIN PARTS p ON s.Pno = p.Pno JOIN PROJECT j ON s.Jno = j.Jno

WHERE j.City = 'Paris';

RAISE NOTICE 'Total Sales for Paris Projects: %', total\_sales; END;

$$;

-- Call the procedure

CALL calc\_total\_sales\_paris();

2)

Here is the SQL code to create the database and apply the integrity constraints:

CREATE TABLE PRODUCT (

Maker VARCHAR(20) NOT NULL, Modelno VARCHAR(10) PRIMARY KEY,

Type VARCHAR(10) NOT NULL CHECK(Type IN ('PC', 'Laptop', 'Printer'))

);

CREATE TABLE PC (

Modelno VARCHAR(10) PRIMARY KEY, Speed DECIMAL(5, 2) NOT NULL,

RAM INTEGER NOT NULL,

HD DECIMAL(5, 2) NOT NULL, CD VARCHAR(10) NOT NULL, Price DECIMAL(10, 2) NOT NULL,

FOREIGN KEY (Modelno) REFERENCES PRODUCT(Modelno)

);

CREATE TABLE LAPTOP (

Modelno VARCHAR(10) PRIMARY KEY, Speed DECIMAL(5, 2) NOT NULL,

RAM INTEGER NOT NULL,

HD DECIMAL(5, 2) NOT NULL, Price DECIMAL(10, 2) NOT NULL,

FOREIGN KEY (Modelno) REFERENCES PRODUCT(Modelno)

);

CREATE TABLE PRINTER (

Modelno VARCHAR(10) PRIMARY KEY,

Color CHAR(1) NOT NULL CHECK(Color IN ('T', 'F')), Type VARCHAR(10) NOT NULL,

Price DECIMAL(10, 2) NOT NULL,

FOREIGN KEY (Modelno) REFERENCES PRODUCT(Modelno)

);

Here is the SQL code to insert records into each table:

INSERT INTO PRODUCT (Maker, Modelno, Type) VALUES

('IBM', 'M001', 'PC'),

('Compaq', 'M002', 'Laptop'),

('HP', 'M003', 'Printer'),

('Dell', 'M004', 'PC'),

('Lenovo', 'M005', 'Laptop'),

('Epson', 'M006', 'Printer'),

('IBM', 'M007', 'PC'),

('Compaq', 'M008', 'Laptop'),

('HP', 'M009', 'Printer'),

('Dell', 'M010', 'PC');

INSERT INTO PC (Modelno, Speed, RAM, HD, CD, Price) VALUES

('M001', 150.00, 256, 20.00, 'CD-ROM', 50000.00),

('M004', 200.00, 512, 40.00, 'CD-RW', 70000.00),

('M007', 250.00, 1024, 80, 60.00, 'CD-RW', 90000.00),

('M010', 300.00, 2048, 100.00, 'DVD-ROM', 120000.00);

INSERT INTO LAPTOP (Modelno, Speed, RAM, HD, Price) VALUES

('M002', 150.00, 256, 20.00, 40000.00),

('M005', 200.00, 512, 40.00, 60000.00),

('M008', 250.00, 1024, 80.00, 80000.00);

INSERT INTO PRINTER (Modelno, Color, Type, Price) VALUES

('M003', 'T', 'Ink-jet', 5000.00),

('M006', 'F', 'Laser', 10000.00),

('M009', 'T', 'Dot-matrix', 3000.00);

Here are the answers to the queries:

1. Find PC models having a speed of at least 150 MHz.

SELECT Modelno FROM PC

WHERE Speed >= 150;

1. Find those manufacturers that sell Laptops, but not PC's.

SELECT P.Maker FROM PRODUCT P

WHERE P.Type = 'Laptop'

AND P.Maker NOT IN (SELECT P.Maker FROM PRODUCT P WHERE P.Type = 'PC');

1. Write a trigger on LAPTOP table such that the price should not less than 30000.

CREATE TRIGGER TR\_LAPTOP\_PRICE BEFORE INSERT OR UPDATE ON LAPTOP FOR EACH ROW

BEGIN

IF NEW.Price < 30000 THEN

SIGNAL SQLSTATE '45000' SET MESSAGE\_TEXT = 'Price should not be less than 30000';

END IF;

END;

1. Write a procedure to find the manufacturer who has produced the most expensive laptop.

CREATE PROCEDURE SP\_MOST\_EXPENSIVE\_LAPTOP() BEGIN

SELECT P.Maker, L.Price FROM LAPTOP L

JOIN PRODUCT P ON L.Modelno = P.Modelno ORDER BY L.Price DESC

LIMIT 1; END;

3)

Same as it is slip number 22. 4)

# Create Tables with Constraints

CREATE TABLE DOCTOR ( Did INT PRIMARY KEY,

Dname VARCHAR(50) NOT NULL, Daddress VARCHAR(100) NOT NULL,

qualification VARCHAR(50) NOT NULL

);

CREATE TABLE PATIENTMASTER ( Pcode INT PRIMARY KEY, Pname VARCHAR(50) NOT NULL, Padd VARCHAR(100) NOT NULL, age INT NOT NULL,

gender CHAR(1) CHECK (gender IN ('M', 'F')) NOT NULL,

bloodgroup VARCHAR(3) NOT NULL, Pid INT NOT NULL,

FOREIGN KEY (Pid) REFERENCES DOCTOR(Did)

);

CREATE TABLE ADMITTEDPATIENT ( P\_code INT PRIMARY KEY,

Entry\_date DATE NOT NULL, Discharge\_date DATE NOT NULL,

wardno INT CHECK (wardno < 6) NOT NULL, disease VARCHAR(50) NOT NULL,

FOREIGN KEY (P\_code) REFERENCES PATIENTMASTER(Pcode)

);

# ◻ Step 2: Insert Sample Data (10 Records Each)

-- Insert into DOCTOR INSERT INTO DOCTOR VALUES

(1, 'Dr. Sharma', 'Delhi', 'MBBS'),

(2, 'Dr. Patel', 'Mumbai', 'MD'),

(3, 'Dr. Khan', 'Bangalore', 'MBBS'),

(4, 'Dr. Mehta', 'Chennai', 'MD'),

(5, 'Dr. Roy', 'Kolkata', 'MS'),

(6, 'Dr. Iyer', 'Hyderabad', 'MBBS'),

(7, 'Dr. Das', 'Pune', 'MD'),

(8, 'Dr. Gupta', 'Ahmedabad', 'MBBS'),

(9, 'Dr. Verma', 'Jaipur', 'MD'),

(10, 'Dr. Reddy', 'Lucknow', 'MS');

-- Insert into PATIENTMASTER INSERT INTO PATIENTMASTER VALUES

|  |  |
| --- | --- |
| (101, | 'Ravi', 'Delhi', 25, 'M', 'A', 1), |
| (102, | 'Sita', 'Mumbai', 30, 'F', 'B', 2), |
| (103, | 'Aman', 'Chennai', 28, 'M', 'O', 3), |
| (104, | 'Neha', 'Pune', 22, 'F', 'AB', 4), |
| (105, | 'Vijay', 'Hyderabad', 35, 'M', 'A', 5), |
| (106, | 'Priya', 'Bangalore', 40, 'F', 'O', 6), |
| (107, | 'Anil', 'Ahmedabad', 31, 'M', 'B', 7), |
| (108, | 'Kavita', 'Kolkata', 29, 'F', 'AB', 8), |
| (109, | 'Sunil', 'Delhi', 50, 'M', 'A', 9), |
| (110, | 'Divya', 'Lucknow', 33, 'F', 'O', 10); |

-- Insert into ADMITTEDPATIENT INSERT INTO ADMITTEDPATIENT VALUES

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| (101, | '2012-03-01', | '2012-03-05', | 3, | 'Flu'), |
| (102, | '2012-03-10', | '2012-03-15', | 2, | 'Malaria'), |
| (103, | '2012-03-12', | '2012-03-18', | 3, | 'Typhoid'), |
| (104, | '2012-02-20', | '2012-03-03', | 1, | 'COVID'), |
| (105, | '2012-03-04', | '2012-03-22', | 4, | 'Fracture'), |
| (106, | '2012-03-08', | '2012-03-24', | 3, | 'Asthma'), |
| (107, | '2012-03-10', | '2012-03-20', | 1, | 'Diabetes'), |
| (108, | '2012-03-02', | '2012-03-26', | 2, | 'Cancer'), |
| (109, | '2012-03-01', | '2012-03-07', | 3, | 'Flu'), |
| (110, | '2012-03-13', | '2012-03-25', | 5, | 'Cold'); |

# ◻ Step 3: Queries

### Find details of doctors treating patients in ward no 3

SELECT DISTINCT D.\* FROM DOCTOR D

JOIN PATIENTMASTER P ON D.Did = P.Pid

JOIN ADMITTEDPATIENT A ON P.Pcode = A.P\_code

WHERE A.wardno = 3;

### Trigger to ensure blood group is A, B, AB or O

CREATE OR REPLACE TRIGGER trg\_check\_bloodgroup BEFORE INSERT OR UPDATE ON PATIENTMASTER

FOR EACH ROW BEGIN

IF :NEW.bloodgroup NOT IN ('A', 'B', 'AB', 'O') THEN

RAISE\_APPLICATION\_ERROR(-20001, 'Invalid blood group!'); END IF;

END;

### Details of patients discharged between '03/03/12' and '25/03/12'

SELECT P.\*

FROM PATIENTMASTER P

JOIN ADMITTEDPATIENT A ON P.Pcode = A.P\_code

WHERE A.Discharge\_date BETWEEN TO\_DATE('03/03/2012', 'DD/MM/YYYY')

AND TO\_DATE('25/03/2012', 'DD/MM/YYYY');

### Procedure to calculate bill of all discharged patients

CREATE OR REPLACE PROCEDURE calculate\_bills AS CURSOR cur IS

SELECT P\_code, wardno, Entry\_date, Discharge\_date FROM ADMITTEDPATIENT;

days\_diff INT; amount INT;

BEGIN

FOR rec IN cur LOOP

days\_diff := rec.Discharge\_date - rec.Entry\_date; amount := days\_diff \* rec.wardno \* 100;

DBMS\_OUTPUT.PUT\_LINE('Patient Code: ' || rec.P\_code || ', Bill: Rs.' || amount); END LOOP;

END;

Use SET SERVEROUTPUT ON; before calling this procedure to see the output.

# ◻ Final Report: Doctors and Their Patients

SELECT D.Did, D.Dname, D.Daddress, D.qualification,

P.Pcode, P.Pname, P.age, P.gender, P.bloodgroup, A.disease, A.wardno FROM DOCTOR D

LEFT JOIN PATIENTMASTER P ON D.Did = P.Pid

LEFT JOIN ADMITTEDPATIENT A ON P.Pcode = A.P\_code ORDER BY D.Did;

5)

# Create Tables with Integrity Constraints

CREATE TABLE DOCTOR ( Did INT PRIMARY KEY,

Dname VARCHAR(50) NOT NULL, Daddress VARCHAR(100) NOT NULL,

qualification VARCHAR(50) NOT NULL

);

CREATE TABLE PATIENTMASTER ( Pcode INT PRIMARY KEY, Pname VARCHAR(50) NOT NULL, Padd VARCHAR(100) NOT NULL, age INT NOT NULL,

gender CHAR(1) CHECK (gender IN ('M', 'F')) NOT NULL,

bloodgroup VARCHAR(3) NOT NULL, Did INT NOT NULL,

FOREIGN KEY (Did) REFERENCES DOCTOR(Did)

);

CREATE TABLE ADMITTEDPATIENT ( Pcode INT PRIMARY KEY,

Entry\_date DATE NOT NULL, Discharge\_date DATE NOT NULL,

wardno INT NOT NULL CHECK (wardno BETWEEN 1 AND 5), disease VARCHAR(50) NOT NULL,

FOREIGN KEY (Pcode) REFERENCES PATIENTMASTER(Pcode)

);

# ◻ Step 2: Insert 10 Sample Records in Each Table

-- DOCTOR

INSERT INTO DOCTOR VALUES

|  |  |  |
| --- | --- | --- |
| (1, | 'Dr. | Verma', 'Delhi', 'MBBS'), |
| (2, | 'Dr. | Sharma', 'Mumbai', 'MD'), |
| (3, | 'Dr. | Reddy', 'Hyderabad', 'MS'), |
| (4, | 'Dr. | Gupta', 'Pune', 'MBBS'), |
| (5, | 'Dr. | Khan', 'Lucknow', 'MD'), |
| (6, | 'Dr. | Mehta', 'Chennai', 'MS'), |
| (7, | 'Dr. | Roy', 'Kolkata', 'MBBS'), |
| (8, | 'Dr. | Das', 'Jaipur', 'MD'), |
| (9, | 'Dr. | Iyer', 'Bangalore', 'MS'), |

(10, 'Dr. Singh', 'Ahmedabad', 'MBBS');

-- PATIENTMASTER

INSERT INTO PATIENTMASTER VALUES

|  |  |
| --- | --- |
| (101, | 'Ravi', 'Delhi', 25, 'M', 'A', 1), |
| (102, | 'Sita', 'Mumbai', 30, 'F', 'B', 2), |
| (103, | 'Aman', 'Chennai', 28, 'M', 'O', 3), |
| (104, | 'Neha', 'Pune', 22, 'F', 'AB', 4), |
| (105, | 'Vijay', 'Hyderabad', 35, 'M', 'A', 5), |
| (106, | 'Priya', 'Bangalore', 40, 'F', 'O', 6), |
| (107, | 'Anil', 'Ahmedabad', 31, 'M', 'B', 7), |
| (108, | 'Kavita', 'Kolkata', 29, 'F', 'AB', 8), |
| (109, | 'Sunil', 'Delhi', 50, 'M', 'A', 9), |
| (110, | 'Divya', 'Lucknow', 33, 'F', 'O', 10); |

-- ADMITTEDPATIENT

INSERT INTO ADMITTEDPATIENT VALUES

|  |  |  |
| --- | --- | --- |
| (101, | '2023-03-01', '2023-03-06', | 3, 'Flu'), |
| (102, | '2023-03-02', '2023-03-10', | 2, 'Malaria'), |
| (103, | '2023-03-03', '2023-03-08', | 3, 'Typhoid'), |
| (104, | '2023-03-01', '2023-03-05', | 4, 'COVID'), |
| (105, | '2023-03-06', '2023-03-12', | 1, 'Fracture'), |
| (106, | '2023-03-08', '2023-03-13', | 3, 'Asthma'), |
| (107, | '2023-03-04', '2023-03-07', | 5, 'Diabetes'), |
| (108, | '2023-03-05', '2023-03-10', | 2, 'Cancer'), |
| (109, | '2023-03-01', '2023-03-06', | 3, 'Flu'), |

(110, '2023-03-07', '2023-03-15', 5, 'Cold');

# ◻ Step 3: Queries

### Doctors treating patients of ward no. 3 with patient name and disease

SELECT D.Did, D.Dname, P.Pname, A.disease FROM DOCTOR D

JOIN PATIENTMASTER P ON D.Did = P.Did

JOIN ADMITTEDPATIENT A ON P.Pcode = A.Pcode

WHERE A.wardno = 3;

### Disease affecting maximum number of patients

SELECT disease, COUNT(\*) AS patient\_count FROM ADMITTEDPATIENT

GROUP BY disease

ORDER BY patient\_count DESC FETCH FIRST 1 ROWS ONLY;

*(If you're using MySQL, replace FETCH FIRST... with LIMIT 1)*

# Trigger to validate wardno in ADMITTEDPATIENT

CREATE OR REPLACE TRIGGER trg\_wardno\_check BEFORE INSERT OR UPDATE ON ADMITTEDPATIENT FOR EACH ROW

BEGIN

IF :NEW.wardno NOT BETWEEN 1 AND 5 THEN

RAISE\_APPLICATION\_ERROR(-20001, 'Ward number must be between 1 and 5.'); END IF;

END;

# D): Procedure to display patients admitted for more than 5 days

CREATE OR REPLACE PROCEDURE show\_long\_stay\_patients AS BEGIN

FOR rec IN (

SELECT P.Pcode, P.Pname, A.Entry\_date, A.Discharge\_date, (A.Discharge\_date - A.Entry\_date) AS stay\_duration

FROM PATIENTMASTER P

JOIN ADMITTEDPATIENT A ON P.Pcode = A.Pcode

WHERE (A.Discharge\_date - A.Entry\_date) > 5

) LOOP

DBMS\_OUTPUT.PUT\_LINE('Pcode: ' || rec.Pcode || ', Name: ' || rec.Pname || ', Stay (days): ' || rec.stay\_duration);

END LOOP;

END;

6)

## SQL Schema Creation with Constraints

-- DOCTOR Table CREATE TABLE DOCTOR (

Did INT PRIMARY KEY,

Dname VARCHAR(50) NOT NULL, Daddress VARCHAR(100) NOT NULL,

qualification VARCHAR(20) NOT NULL

);

-- PATIENTMASTER Table CREATE TABLE PATIENTMASTER (

Pcode INT PRIMARY KEY, Pname VARCHAR(50) NOT NULL, Padd VARCHAR(100) NOT NULL, age INT NOT NULL,

gender CHAR(1) CHECK (gender IN ('M', 'F')) NOT NULL,

bloodgroup VARCHAR(3) NOT NULL, aid INT NOT NULL,

FOREIGN KEY (aid) REFERENCES DOCTOR(Did)

);

-- ADMITTEDPATIENT Table CREATE TABLE ADMITTEDPATIENT (

Pcode INT PRIMARY KEY,

Entry\_date DATE NOT NULL, Discharge\_date DATE NOT NULL,

wardno INT NOT NULL CHECK (wardno < 6), disease VARCHAR(50) NOT NULL,

FOREIGN KEY (Pcode) REFERENCES PATIENTMASTER(Pcode)

);

## 🧾 2. Insert 10 Sample Records per Table

-- DOCTOR

INSERT INTO DOCTOR VALUES

|  |  |  |
| --- | --- | --- |
| (1, | 'Dr. | Verma', 'Delhi', 'MBBS'), |
| (2, | 'Dr. | Sharma', 'Mumbai', 'MD'), |
| (3, | 'Dr. | Reddy', 'Hyderabad', 'MBBS'), |
| (4, | 'Dr. | Gupta', 'Pune', 'MS'), |
| (5, | 'Dr. | Khan', 'Lucknow', 'MBBS'), |
| (6, | 'Dr. | Mehta', 'Chennai', 'MS'), |
| (7, | 'Dr. | Roy', 'Kolkata', 'MD'), |
| (8, | 'Dr. | Das', 'Jaipur', 'MBBS'), |
| (9, | 'Dr. | Iyer', 'Bangalore', 'MS'), |

(10, 'Dr. Singh', 'Ahmedabad', 'MBBS');

-- PATIENTMASTER

INSERT INTO PATIENTMASTER VALUES

|  |  |
| --- | --- |
| (101, | 'Ravi', 'Delhi', 25, 'M', 'A', 1), |
| (102, | 'Sita', 'Mumbai', 30, 'F', 'B', 2), |
| (103, | 'Aman', 'Chennai', 28, 'M', 'O', 3), |
| (104, | 'Neha', 'Pune', 22, 'F', 'AB', 4), |
| (105, | 'Vijay', 'Hyderabad', 35, 'M', 'A', 5), |
| (106, | 'Priya', 'Bangalore', 40, 'F', 'O', 6), |
| (107, | 'Anil', 'Ahmedabad', 31, 'M', 'B', 7), |
| (108, | 'Kavita', 'Kolkata', 29, 'F', 'AB', 8), |
| (109, | 'Sunil', 'Delhi', 49, 'M', 'A', 9), |
| (110, | 'Divya', 'Lucknow', 33, 'F', 'O', 10); |

-- ADMITTEDPATIENT

INSERT INTO ADMITTEDPATIENT VALUES

(101, '2024-03-01', '2024-03-06', 3, 'Flu'),

(102, '2024-03-02', '2024-03-10', 2, 'Malaria'),

(103, '2024-03-03', '2024-03-08', 3, 'Typhoid'),

(104, '2024-03-01', '2024-03-05', 4, 'COVID'),

(105, '2024-03-06', '2024-03-12', 1, 'Fracture'),

(106, '2024-03-08', '2024-03-13', 3, 'Asthma'),

(107, '2024-03-04', '2024-03-07', 5, 'Diabetes'),

(108, '2024-03-05', '2024-03-10', 2, 'Blood Cancer'),

(109, '2024-03-01', '2024-03-06', 3, 'Blood Cancer'),

(110, '2024-03-07', '2024-03-15', 5, 'Cold');

**🧾 3. SQL Queries**

# Patients treated by M.B.B.S. doctors

SELECT P.Pcode, P.Pname, D.Dname, D.qualification FROM PATIENTMASTER P

JOIN DOCTOR D ON P.aid = D.Did

WHERE D.qualification = 'MBBS';

# Patient with blood cancer, age < 50 and blood group 'A'

SELECT P.Pcode, P.Pname, P.age, P.bloodgroup, A.disease FROM PATIENTMASTER P

JOIN ADMITTEDPATIENT A ON P.Pcode = A.Pcode

WHERE A.disease = 'Blood Cancer' AND P.age < 50 AND P.bloodgroup = 'A';

## Procedure to calculate bills (no. of days \* ₹600)

CREATE OR REPLACE PROCEDURE calculate\_bill IS CURSOR bill\_cur IS

SELECT Pcode, Entry\_date, Discharge\_date FROM ADMITTEDPATIENT; days\_stayed INT;

total\_bill NUMBER;

BEGIN

FOR rec IN bill\_cur LOOP

days\_stayed := rec.Discharge\_date - rec.Entry\_date; total\_bill := days\_stayed \* 600;

DBMS\_OUTPUT.PUT\_LINE('Pcode: ' || rec.Pcode || ' | Bill: ₹' || total\_bill); END LOOP;

END;

Before calling: SET SERVEROUTPUT ON;

Then call: EXEC calculate\_bill;

1. **Cursor to fetch and display last record in PATIENTMASTER**

CREATE OR REPLACE PROCEDURE get\_last\_patient IS CURSOR cur IS

SELECT \* FROM PATIENTMASTER ORDER BY Pcode; last\_rec PATIENTMASTER%ROWTYPE;

BEGIN

FOR rec IN cur LOOP last\_rec := rec;

END LOOP;

DBMS\_OUTPUT.PUT\_LINE('Last Patient -> Code: ' || last\_rec.Pcode ||

', Name: ' || last\_rec.Pname || ', Age: ' || last\_rec.age ||

', Gender: ' || last\_rec.gender);

END;

7)

# Database Schema Creation with Constraints

-- DOCTOR table CREATE TABLE DOCTOR (

Did INT PRIMARY KEY,

Dname VARCHAR(50) NOT NULL, Daddress VARCHAR(100) NOT NULL,

qualification VARCHAR(20) NOT NULL

);

-- PATIENTMASTER table CREATE TABLE PATIENTMASTER (

Pcode INT PRIMARY KEY, Pname VARCHAR(50) NOT NULL, Padd VARCHAR(100) NOT NULL, age INT NOT NULL,

gender CHAR(1) CHECK (gender IN ('M', 'F')) NOT NULL,

bloodgroup VARCHAR(3) NOT NULL, Did INT NOT NULL,

FOREIGN KEY (Did) REFERENCES DOCTOR(Did)

);

-- ADMITTEDPATIENT table CREATE TABLE ADMITTEDPATIENT (

Pcode INT PRIMARY KEY,

Entry\_date DATE NOT NULL, Discharge\_date DATE NOT NULL,

wardno INT NOT NULL CHECK (wardno < 6), disease VARCHAR(50) NOT NULL,

FOREIGN KEY (Pcode) REFERENCES PATIENTMASTER(Pcode)

);

# ◻ 2. Sample Data Insertion (10 records each)

-- DOCTOR records

INSERT INTO DOCTOR VALUES

|  |  |  |
| --- | --- | --- |
| (1, | 'Dr. | Mehta', 'Delhi', 'MBBS'), |
| (2, | 'Dr. | Sharma', 'Mumbai', 'MD'), |
| (3, | 'Dr. | Reddy', 'Hyderabad', 'MS'), |
| (4, | 'Dr. | Gupta', 'Pune', 'MBBS'), |
| (5, | 'Dr. | Khan', 'Lucknow', 'MS'), |
| (6, | 'Dr. | Das', 'Chennai', 'MBBS'), |
| (7, | 'Dr. | Roy', 'Kolkata', 'MD'), |
| (8, | 'Dr. | Iyer', 'Bangalore', 'MS'), |
| (9, | 'Dr. | Patel', 'Jaipur', 'MBBS'), |

(10, 'Dr. Bansal', 'Ahmedabad', 'MD');

-- PATIENTMASTER records

INSERT INTO PATIENTMASTER VALUES

(101, 'Ravi', 'Delhi', 25, 'M', 'A', 3),

(102, 'Sita', 'Mumbai', 30, 'F', 'B', 5),

(103, 'Aman', 'Chennai', 28, 'M', 'O', 8),

(104, 'Neha', 'Pune', 22, 'F', 'AB', 4),

(105, 'Vijay', 'Hyderabad', 35, 'M', 'A', 3),

(106, 'Priya', 'Bangalore', 40, 'F', 'O', 5),

(107, 'Anil', 'Ahmedabad', 31, 'M', 'B', 6),

(108, 'Kavita', 'Kolkata', 29, 'F', 'AB', 8),

(109, 'Sunil', 'Delhi', 49, 'M', 'A', 3),

(110, 'Divya', 'Lucknow', 33, 'F', 'O', 1);

-- ADMITTEDPATIENT records

INSERT INTO ADMITTEDPATIENT VALUES

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| (101, | '2024-03-01', | '2024-03-18', | 3, | 'Flu'), |
| (102, | '2024-03-02', | '2024-03-05', | 2, | 'Malaria'), |
| (103, | '2024-03-03', | '2024-03-25', | 3, | 'Typhoid'), |
| (104, | '2024-03-01', | '2024-03-04', | 4, | 'COVID'), |
| (105, | '2024-03-06', | '2024-03-28', | 1, | 'Fracture'), |
| (106, | '2024-03-08', | '2024-03-10', | 3, | 'Asthma'), |
| (107, | '2024-03-04', | '2024-03-05', | 5, | 'Diabetes'), |
| (108, | '2024-03-05', | '2024-03-30', | 2, | 'Cancer'), |
| (109, | '2024-03-01', | '2024-03-20', | 3, | 'Flu'), |
| (110, | '2024-03-07', | '2024-03-10', | 5, | 'Cold'); |

# ◻ 3. SQL Queries

### Patients treated by M.S. doctors

SELECT P.\*

FROM PATIENTMASTER P

JOIN DOCTOR D ON P.Did = D.Did

WHERE D.qualification = 'MS';

### Name of doctor treating maximum number of patients

SELECT D.Dname, COUNT(\*) AS patient\_count FROM DOCTOR D

JOIN PATIENTMASTER P ON D.Did = P.Did GROUP BY D.Dname

ORDER BY patient\_count DESC FETCH FIRST 1 ROWS ONLY;

*In MySQL, replace FETCH FIRST 1 ROWS ONLY with LIMIT 1.*

# Procedure for patients admitted for more than 15 days

CREATE OR REPLACE PROCEDURE show\_long\_stay\_patients IS BEGIN

FOR rec IN (

SELECT P.Pcode, P.Pname, A.Entry\_date, A.Discharge\_date, (A.Discharge\_date - A.Entry\_date) AS stay\_duration

FROM PATIENTMASTER P

JOIN ADMITTEDPATIENT A ON P.Pcode = A.Pcode

WHERE (A.Discharge\_date - A.Entry\_date) > 15

) LOOP

DBMS\_OUTPUT.PUT\_LINE('Patient: ' || rec.Pname ||

', Days: ' || rec.stay\_duration);

END LOOP;

END;

Enable output with: SET SERVEROUTPUT ON;

# View combining DOCTOR & PATIENTMASTER

CREATE OR REPLACE VIEW doctor\_patient\_view AS

SELECT P.Pcode, P.Pname, P.age, P.gender, P.bloodgroup, D.Did, D.Dname, D.qualification FROM PATIENTMASTER P

JOIN DOCTOR D ON P.Did = D.Did;

# d). Update patients of 'B.A.-M.S.' doctors to MBBS

Assuming **"B.A.-M.S."** is a typo and should refer to doctors with **'MS' qualification**, this query will reassign patients to a doctor with **MBBS qualification**:

-- Step 1: Find a MBBS doctor (we'll pick Did = 1 for this example)

-- Step 2: Update patients linked to MS doctors to MBBS doctor

UPDATE PATIENTMASTER SET Did = 1

WHERE Did IN (

SELECT Did FROM DOCTOR WHERE qualification = 'MS'

);

8)

## CREATE TABLES with CONSTRAINTS (MSSQL)

-- ACCOUNT Table CREATE TABLE ACCOUNT (

accno INT PRIMARY KEY CHECK (accno < 1000),

open\_date DATE NOT NULL,

acctype CHAR(1) NOT NULL CHECK (acctype IN ('P', 'J')),

balance MONEY NOT NULL

);

-- CUSTOMER Table CREATE TABLE CUSTOMER (

cust\_id INT PRIMARY KEY, name VARCHAR(100) NOT NULL,

address VARCHAR(200) NOT NULL,

accno INT NOT NULL,

FOREIGN KEY (accno) REFERENCES ACCOUNT(accno)

);

-- TRANSACTION Table CREATE TABLE TRANSACTION (

trans\_id INT PRIMARY KEY, trans\_date DATE NOT NULL, accno INT NOT NULL,

trans\_type CHAR(1) NOT NULL CHECK (trans\_type IN ('C', 'D')),

amount MONEY NOT NULL,

FOREIGN KEY (accno) REFERENCES ACCOUNT(accno)

);

## 🧾 2. INSERT 10 RECORDS PER TABLE

-- ACCOUNT

INSERT INTO ACCOUNT VALUES

|  |  |  |  |
| --- | --- | --- | --- |
| (101, | '2024-01-01', | 'P', | 120000), |
| (102, | '2024-01-10', | 'J', | 85000), |
| (103, | '2024-02-05', | 'P', | 105000), |
| (104, | '2024-02-20', | 'P', | 60000), |
| (105, | '2024-03-01', | 'J', | 50000), |
| (106, | '2024-03-10', | 'P', | 30000), |
| (107, | '2024-03-15', | 'J', | 150000), |
| (108, | '2024-04-01', | 'P', | 95000), |
| (109, | '2024-04-05', | 'P', | 200000), |
| (110, | '2024-04-10', | 'J', | 40000); |

-- CUSTOMER

INSERT INTO CUSTOMER VALUES

(1, 'Ravi Kumar', 'Delhi', 101),

(2, 'Neha Sharma', 'Mumbai', 102),

(3, 'Arun Mehta', 'Chennai', 103),

(4, 'Sita Verma', 'Pune', 104),

(5, 'Anil Gupta', 'Hyderabad', 105),

(6, 'Priya Das', 'Kolkata', 106),

(7, 'Kiran Roy', 'Lucknow', 107),

(8, 'Nidhi Joshi', 'Bangalore', 108),

(9, 'Amit Patel', 'Ahmedabad', 109),

(10, 'Divya Iyer', 'Chandigarh', 110);

-- TRANSACTION

INSERT INTO TRANSACTION VALUES

|  |  |  |  |
| --- | --- | --- | --- |
| (1, | '2012-03-25', | 101, 'C', | 25000), |
| (2, | '2012-03-26', | 102, 'C', | 10000), |
| (3, | '2012-03-27', | 103, 'C', | 5000), |
| (4, | '2012-03-28', | 104, 'D', | 2000), |
| (5, | '2012-03-24', | 105, 'C', | 3000), |
| (6, | '2012-03-25', | 106, 'D', | 1500), |
| (7, | '2012-03-26', | 107, 'C', | 6000), |
| (8, | '2012-03-27', | 108, 'D', | 2500), |
| (9, | '2012-03-28', | 109, 'C', | 8000), |

(10, '2012-03-25', 110, 'C', 12000);

**🧾 3. QUERIES**

# Customers with minimum balance ≥ ₹1,00,000

SELECT C.\* FROM CUSTOMER C

JOIN ACCOUNT A ON C.accno = A.accno WHERE A.balance >= 100000;

# Amounts Credited Between 25-03-2012 and 28-03-2012

SELECT \*

FROM TRANSACTION

WHERE trans\_type = 'C'

AND trans\_date BETWEEN '2012-03-25' AND '2012-03-28';

## TRIGGER to UPDATE BALANCE After Transaction

CREATE TRIGGER trg\_update\_balance ON TRANSACTION

AFTER INSERT AS

BEGIN

UPDATE ACCOUNT

SET balance = CASE

WHEN I.trans\_type = 'C' THEN A.balance + I.amount WHEN I.trans\_type = 'D' THEN A.balance - I.amount

END

FROM ACCOUNT A

JOIN INSERTED I ON A.accno = I.accno; END;

## CURSOR to Check Loan Eligibility

DECLARE @accno INT, @balance MONEY;

DECLARE acc\_cursor CURSOR FOR SELECT accno, balance FROM ACCOUNT;

OPEN acc\_cursor;

FETCH NEXT FROM acc\_cursor INTO @accno, @balance;

WHILE @@FETCH\_STATUS = 0 BEGIN

PRINT 'Acc No: ' + CAST(@accno AS VARCHAR) + ' - ' + CASE

WHEN @balance < 10000 THEN 'Loan is not provided' ELSE 'Loan is provided'

END;

FETCH NEXT FROM acc\_cursor INTO @accno, @balance; END;

CLOSE acc\_cursor;

DEALLOCATE acc\_cursor;

9)

## Database Schema with Constraints

-- ACCOUNT Table CREATE TABLE ACCOUNT (

accno INT PRIMARY KEY CHECK (accno < 1000),

open\_date DATE NOT NULL,

acctype CHAR(1) NOT NULL CHECK (acctype IN ('P', 'J')),

balance MONEY NOT NULL

);

-- CUSTOMER Table CREATE TABLE CUSTOMER (

cust\_id INT PRIMARY KEY,

name VARCHAR(100) NOT NULL, address VARCHAR(200) NOT NULL,

accno INT NOT NULL,

FOREIGN KEY (accno) REFERENCES ACCOUNT(accno)

);

-- TRANSACTION Table CREATE TABLE TRANSACTION (

trans\_id INT PRIMARY KEY, trans\_date DATE NOT NULL, cno INT NOT NULL,

trans\_type CHAR(1) NOT NULL CHECK (trans\_type IN ('C', 'D')), amount MONEY NOT NULL,

FOREIGN KEY (cno) REFERENCES CUSTOMER(cust\_id)

);

## 🧾 2. Insert 10 Sample Records per Table

-- ACCOUNT records

INSERT INTO ACCOUNT VALUES

|  |  |  |  |
| --- | --- | --- | --- |
| (101, | '2023-01-01', | 'P', | 150000), |
| (102, | '2023-02-10', | 'J', | 250000), |
| (103, | '2023-03-15', | 'P', | 100000), |
| (104, | '2023-04-01', | 'P', | 190000), |
| (105, | '2023-04-20', | 'J', | 50000), |
| (106, | '2023-05-01', | 'J', | 210000), |
| (107, | '2023-05-15', | 'P', | 290000), |
| (108, | '2023-06-01', | 'P', | 120000), |
| (109, | '2023-06-10', | 'J', | 80000), |
| (110, | '2023-06-20', | 'P', | 60000); |

-- CUSTOMER records

INSERT INTO CUSTOMER VALUES

(1, 'Ravi Kumar', 'Delhi', 101),

(2, 'Neha Sharma', 'Mumbai', 102),

(3, 'Aman Verma', 'Chennai', 103),

(4, 'Sita Singh', 'Pune', 104),

(5, 'Anil Gupta', 'Hyderabad', 105),

(6, 'Priya Das', 'Kolkata', 106),

(7, 'Kiran Roy', 'Lucknow', 107),

(8, 'Nidhi Joshi', 'Bangalore', 108),

(9, 'Amit Patel', 'Ahmedabad', 109),

(10, 'Divya Iyer', 'Chandigarh', 110);

-- TRANSACTION records

INSERT INTO TRANSACTION VALUES

|  |  |  |
| --- | --- | --- |
| (1, | '2024-05-01', | 1, 'C', 20000), |
| (2, | '2024-05-02', | 2, 'D', 5000), |
| (3, | '2024-05-03', | 3, 'C', 10000), |
| (4, | '2024-05-04', | 4, 'D', 7000), |
| (5, | '2024-05-05', | 5, 'C', 8000), |
| (6, | '2024-05-06', | 6, 'D', 3000), |
| (7, | '2024-05-07', | 7, 'D', 4000), |
| (8, | '2024-05-08', | 8, 'C', 25000), |
| (9, | '2024-05-09', | 9, 'D', 6000), |

(10, '2024-05-10', 10, 'C', 15000);

**🧾 3. Queries**

# Customers with personal accounts and balance < ₹2,00,000

SELECT C.\* FROM CUSTOMER C

JOIN ACCOUNT A ON C.accno = A.accno

WHERE A.acctype = 'P' AND A.balance < 200000;

# Customers with joint accounts

SELECT C.\* FROM CUSTOMER C

JOIN ACCOUNT A ON C.accno = A.accno WHERE A.acctype = 'J';

## Trigger: Prevent withdrawal if balance < ₹300

CREATE TRIGGER trg\_check\_balance ON TRANSACTION

INSTEAD OF INSERT AS

BEGIN

DECLARE @accno INT, @amount MONEY, @trans\_type CHAR(1), @custid INT;

SELECT @custid = cno, @trans\_type = trans\_type, @amount = amount FROM inserted;

SELECT @accno = accno FROM CUSTOMER WHERE cust\_id = @custid;

IF @trans\_type = 'D' BEGIN

DECLARE @current\_balance MONEY;

SELECT @current\_balance = balance FROM ACCOUNT WHERE accno = @accno;

IF @current\_balance - @amount < 300 BEGIN

RAISERROR('Withdrawal denied: Balance would fall below ₹300.', 16, 1); RETURN;

END

END

INSERT INTO TRANSACTION

SELECT \* FROM inserted;

IF @trans\_type = 'C'

UPDATE ACCOUNT SET balance = balance + @amount WHERE accno = @accno; ELSE

UPDATE ACCOUNT SET balance = balance - @amount WHERE accno = @accno;

END;

## Procedure: Add transaction and update ACCOUNT balance

CREATE PROCEDURE sp\_add\_transaction @trans\_id INT,

@trans\_date DATE, @cno INT, @trans\_type CHAR(1), @amount MONEY

AS BEGIN

DECLARE @accno INT;

SELECT @accno = accno FROM CUSTOMER WHERE cust\_id = @cno;

IF @trans\_type = 'D' BEGIN

IF EXISTS (SELECT 1 FROM ACCOUNT WHERE accno = @accno AND balance - @amount <

300)

BEGIN

RAISERROR('Cannot withdraw: balance would drop below ₹300.', 16, 1); RETURN;

END

END

INSERT INTO TRANSACTION VALUES (@trans\_id, @trans\_date, @cno, @trans\_type, @amount);

IF @trans\_type = 'C'

UPDATE ACCOUNT SET balance = balance + @amount WHERE accno = @accno; ELSE

UPDATE ACCOUNT SET balance = balance - @amount WHERE accno = @accno;

END;

10)

## Create Tables with Constraints (MSSQL)

-- ACCOUNT table CREATE TABLE ACCOUNT (

accno INT PRIMARY KEY CHECK (accno < 1000),

open\_date DATE NOT NULL,

acctype CHAR(1) NOT NULL CHECK (acctype IN ('P', 'M')), -- 'M' assumed to be a typo for Joint, handled as per request

balance MONEY NOT NULL CHECK (balance IS NOT NULL)

);

-- CUSTOMER table CREATE TABLE CUSTOMER (

cust\_id INT PRIMARY KEY, name VARCHAR(100) NOT NULL,

address VARCHAR(200) NOT NULL,

accno INT NOT NULL,

FOREIGN KEY (accno) REFERENCES ACCOUNT(accno)

);

-- TRANSACTION table CREATE TABLE TRANSACTION (

trans\_id INT PRIMARY KEY, trans\_date DATE NOT NULL, accno INT NOT NULL,

trans\_type CHAR(1) NOT NULL CHECK (trans\_type IN ('C', 'D')), amount MONEY NOT NULL,

FOREIGN KEY (accno) REFERENCES ACCOUNT(accno)

);

## 🧾 2. Insert 10 Records into Each Table

-- ACCOUNT records

INSERT INTO ACCOUNT VALUES (101, '2012-03-01', 'P', 1500),

(102, '2012-03-05', 'M', 2500),

(103, '2012-03-10', 'P', 900),

(104, '2012-03-12', 'M', 4500),

(105, '2012-03-15', 'P', 600),

(106, '2012-03-16', 'M', 3000),

(107, '2012-03-17', 'P', 800),

(108, '2012-03-18', 'P', 1200),

(109, '2012-03-19', 'M', 2700),

(110, '2012-03-20', 'P', 1900);

-- CUSTOMER records

INSERT INTO CUSTOMER VALUES

(1, 'Ravi Kumar', 'Delhi', 101),

(2, 'Neha Sharma', 'Mumbai', 102),

(3, 'Anil Mehta', 'Chennai', 103),

(4, 'Sita Verma', 'Pune', 104),

(5, 'Kiran Roy', 'Hyderabad', 105),

(6, 'Priya Das', 'Kolkata', 106),

(7, 'Amit Patel', 'Lucknow', 107),

(8, 'Divya Iyer', 'Bangalore', 108),

(9, 'Ramesh Babu', 'Ahmedabad', 109),

(10, 'Nidhi Joshi', 'Chandigarh', 110);

-- TRANSACTION records

INSERT INTO TRANSACTION VALUES

|  |  |  |  |
| --- | --- | --- | --- |
| (1, | '2012-03-15', | 101, 'C', | 200), |
| (2, | '2012-03-16', | 101, 'D', | 100), |
| (3, | '2012-03-17', | 102, 'C', | 500), |
| (4, | '2012-03-18', | 103, 'D', | 50), |
| (5, | '2012-03-15', | 104, 'C', | 300), |
| (6, | '2012-03-17', | 105, 'D', | 200), |
| (7, | '2012-03-16', | 106, 'C', | 400), |
| (8, | '2012-03-18', | 107, 'C', | 250), |
| (9, | '2012-03-14', | 108, 'C', | 300), |

(10, '2012-03-15', 101, 'D', 50);

**🧾 3. SQL Queries**

# All transactions for account number 101 + customer(s) owning it:

SELECT T.\*, C.name FROM TRANSACTION T

JOIN CUSTOMER C ON T.accno = C.accno WHERE T.accno = 101;

# Details of amount credited between 15-03-2012 and 18-03-2012:

SELECT \*

FROM TRANSACTION

WHERE trans\_type = 'C'

AND trans\_date BETWEEN '2012-03-15' AND '2012-03-18';

## Trigger on ACCOUNT: Prevent debit if balance ≤ ₹500

Note: Since you want this on the ACCOUNT table for **debit**, but debits happen through TRANSACTION, the correct place for this logic is on the TRANSACTION table.

CREATE TRIGGER trg\_prevent\_low\_balance ON TRANSACTION

INSTEAD OF INSERT AS

BEGIN

DECLARE @accno INT, @amount MONEY, @type CHAR(1);

SELECT @accno = accno, @amount = amount, @type = trans\_type FROM inserted;

IF @type = 'D' BEGIN

DECLARE @current\_balance MONEY;

SELECT @current\_balance = balance FROM ACCOUNT WHERE accno = @accno;

IF @current\_balance - @amount <= 500 BEGIN

RAISERROR('Cannot debit. Balance would fall below or equal to ₹500.', 16, 1); RETURN;

END ELSE BEGIN

-- Proceed with insert and update

INSERT INTO TRANSACTION SELECT \* FROM inserted;

UPDATE ACCOUNT SET balance = balance - @amount WHERE accno = @accno;

END

END

ELSE BEGIN

INSERT INTO TRANSACTION SELECT \* FROM inserted;

UPDATE ACCOUNT SET balance = balance + @amount WHERE accno = @accno;

END END;

## Procedure to Calculate Interest on Balance

CREATE PROCEDURE sp\_calculate\_interest @accno INT,

@rate DECIMAL(5,2) -- rate as percent, e.g., 4.5

AS BEGIN

DECLARE @balance MONEY, @open\_date DATE, @days INT, @interest MONEY;

SELECT @balance = balance, @open\_date = open\_date FROM ACCOUNT WHERE accno = @accno; SET @days = DATEDIFF(DAY, @open\_date, GETDATE());

SET @interest = @balance \* (@rate / 100.0) \* (@days / 365.0);

PRINT 'Account Number: ' + CAST(@accno AS VARCHAR);

PRINT 'Interest from open date to today: ₹' + CAST(@interest AS VARCHAR(20)); END;

### Usage:

EXEC sp\_calculate\_interest @accno = 101, @rate = 5.0;

11)

## Create the Tables with Constraints

-- ACCOUNT table CREATE TABLE ACCOUNT (

accno INT PRIMARY KEY CHECK (accno < 1000), -- Account number must be less than 3 digits

open\_date DATE NOT NULL,

acctype CHAR(1) NOT NULL CHECK (acctype IN ('P', 'M')), -- 'P' = Personal, 'M' = Moira (Joint)

balance MONEY NOT NULL

);

-- CUSTOMER table CREATE TABLE CUSTOMER (

cust\_id INT PRIMARY KEY, name VARCHAR(100) NOT NULL,

address VARCHAR(200) NOT NULL,

accno INT NOT NULL,

FOREIGN KEY (accno) REFERENCES ACCOUNT(accno)

);

-- TRANSACTION table CREATE TABLE TRANSACTION (

trans\_id INT PRIMARY KEY, trans\_date DATE NOT NULL, accno INT NOT NULL,

trans\_type CHAR(1) NOT NULL CHECK (trans\_type IN ('C', 'D')), -- C = Credit, D =

Debit

amount MONEY NOT NULL,

FOREIGN KEY (accno) REFERENCES ACCOUNT(accno)

);

## ✅Step 2: Insert Sample Records (10 per table)

-- Insert into ACCOUNT INSERT INTO ACCOUNT VALUES

|  |  |  |  |
| --- | --- | --- | --- |
| (101, | '2012-03-25', | 'P', | 150000), |
| (102, | '2012-03-26', | 'M', | 190000), |
| (103, | '2012-03-27', | 'M', | 180000), |
| (104, | '2012-03-28', | 'P', | 120000), |
| (105, | '2012-03-20', | 'M', | 220000), |
| (106, | '2012-03-22', | 'P', | 100000), |
| (107, | '2012-03-29', | 'M', | 80000), |
| (108, | '2012-03-24', | 'P', | 50000), |
| (109, | '2012-03-23', | 'M', | 90000), |
| (110, | '2012-03-30', | 'P', | 75000); |

-- Insert into CUSTOMER INSERT INTO CUSTOMER VALUES

(1, 'Ravi Kumar', 'Delhi', 101),

(2, 'Neha Sharma', 'Mumbai', 102),

(3, 'Amit Roy', 'Chennai', 103),

(4, 'Priya Das', 'Kolkata', 104),

(5, 'Sita Verma', 'Pune', 105),

(6, 'Anil Mehta', 'Hyderabad', 106),

(7, 'Divya Iyer', 'Bangalore', 107),

(8, 'Kiran Rao', 'Lucknow', 108),

(9, 'Nidhi Joshi', 'Chandigarh', 109),

(10, 'Aman Singh', 'Ahmedabad', 110);

-- Insert into TRANSACTION INSERT INTO TRANSACTION VALUES

(1, '2022-04-01', 101, 'C', 10000),

(2, '2022-04-02', 102, 'D', 5000),

(3, '2022-04-03', 103, 'C', 8000),

(4, '2022-04-04', 104, 'D', 2000),

(5, '2022-04-05', 105, 'C', 3000),

(6, '2022-04-06', 106, 'D', 1000),

(7, '2022-04-07', 107, 'C', 4000),

(8, '2022-04-08', 108, 'D', 1500),

(9, '2022-04-09', 109, 'C', 2500),

(10, '2022-04-10', 110, 'D', 1200);

**🧾 Queries**

# Customers who opened accounts between 25-03-2012 and 28-03-2012

SELECT C.\* FROM CUSTOMER C

JOIN ACCOUNT A ON C.accno = A.accno

WHERE A.open\_date BETWEEN '2012-03-25' AND '2012-03-28';

# Customers with Moira (Joint) accounts and balance < ₹2,00,000

SELECT C.\* FROM CUSTOMER C

JOIN ACCOUNT A ON C.accno = A.accno

WHERE A.acctype = 'M' AND A.balance < 200000;

## Trigger: Update ACCOUNT balance on transaction

#### Updates balance **automatically** when a credit (C) or debit (D) is inserted.

CREATE TRIGGER trg\_update\_balance ON TRANSACTION

AFTER INSERT AS

BEGIN

DECLARE @accno INT, @amount MONEY, @type CHAR(1);

SELECT @accno = accno, @amount = amount, @type = trans\_type FROM inserted;

IF @type = 'C'

UPDATE ACCOUNT SET balance = balance + @amount WHERE accno = @accno; ELSE IF @type = 'D'

UPDATE ACCOUNT SET balance = balance - @amount WHERE accno = @accno;

END;

## Cursor: Fetch the last row of the CUSTOMER table

DECLARE @cust\_id INT, @name VARCHAR(100), @address VARCHAR(200), @accno INT;

DECLARE cust\_cursor CURSOR FOR SELECT cust\_id, name, address, accno FROM CUSTOMER

ORDER BY cust\_id;

OPEN cust\_cursor;

-- Go to first

FETCH NEXT FROM cust\_cursor INTO @cust\_id, @name, @address, @accno;

-- Loop through to last WHILE @@FETCH\_STATUS = 0

BEGIN

FETCH NEXT FROM cust\_cursor INTO @cust\_id, @name, @address, @accno;

END

-- Print the last row

PRINT 'Last Customer Row:';

PRINT 'ID: ' + CAST(@cust\_id AS VARCHAR);

PRINT 'Name: ' + @name; PRINT 'Address: ' + @address;

PRINT 'Account No: ' + CAST(@accno AS VARCHAR);

CLOSE cust\_cursor;

DEALLOCATE cust\_cursor;

12)

## Create Tables with Integrity Constraints

-- Create EMPLOYEE table CREATE TABLE EMPLOYEE (

fname VARCHAR(50) NOT NULL, lname VARCHAR(50) NOT NULL,

ssn CHAR(9) PRIMARY KEY NOT NULL, -- Social Security Number sex CHAR(1) CHECK (sex IN ('M', 'F')) NOT NULL,

salary DECIMAL(10, 2) NOT NULL,

joindate DATE NOT NULL, superssn CHAR(9),

dno INT CHECK (dno < 1000) NOT NULL, -- Department number less than 4 digits FOREIGN KEY (superssn) REFERENCES EMPLOYEE(ssn),

FOREIGN KEY (dno) REFERENCES DEPT(dnum)

);

-- Create DEPT table CREATE TABLE DEPT (

dname VARCHAR(50) NOT NULL,

dnum INT PRIMARY KEY CHECK (dnum < 1000), -- Department number less than 4 digits mgrssn CHAR(9) NOT NULL, -- Manager's SSN

dlocation VARCHAR(100) NOT NULL,

FOREIGN KEY (mgrssn) REFERENCES EMPLOYEE(ssn)

);

-- Create PROJECT table CREATE TABLE PROJECT (

pname VARCHAR(50) NOT NULL,

pno INT PRIMARY KEY CHECK (pno < 1000), -- Project number less than 4 digits plocation VARCHAR(100) NOT NULL,

dnumber INT CHECK (dnumber < 1000) NOT NULL, -- Department number less than 4 digits FOREIGN KEY (dnumber) REFERENCES DEPT(dnum)

);

-- Create WORK\_ON table CREATE TABLE WORK\_ON (

ssn CHAR(9) NOT NULL, -- Employee's SSN pno INT NOT NULL, -- Project number hours DECIMAL(5, 2) NOT NULL,

PRIMARY KEY (ssn, pno),

FOREIGN KEY (ssn) REFERENCES EMPLOYEE(ssn), FOREIGN KEY (pno) REFERENCES PROJECT(pno)

);

## ✅Step 2: Insert Sample Records (10 per table)

-- Insert records into DEPT

INSERT INTO DEPT (dname, dnum, mgrssn, dlocation) VALUES ('HR', 1, '123456789', 'New York'),

('Finance', 2, '987654321', 'London'),

('IT', 3, '234567890', 'San Francisco'),

('Marketing', 4, '345678901', 'Tokyo'),

('Sales', 5, '456789012', 'Berlin'),

('Operations', 6, '567890123', 'Paris'),

('R&D', 7, '678901234', 'Sydney'),

('Legal', 8, '789012345', 'Toronto'),

('Customer Support', 9, '890123456', 'Dubai'),

('Logistics', 10, '901234567', 'Singapore');

-- Insert records into EMPLOYEE

INSERT INTO EMPLOYEE (fname, lname, ssn, sex, salary, joindate, superssn, dno) VALUES ('John', 'Doe', '123456789', 'M', 60000, '2010-01-15', NULL, 1),

('Jane', 'Smith', '987654321', 'F', 65000, '2012-03-22', '123456789', 2),

('Alice', 'Johnson', '234567890', 'F', 70000, '2015-06-10', '123456789', 3),

('Bob', 'Brown', '345678901', 'M', 55000, '2018-09-05', '987654321', 4),

('Charlie', 'Davis', '456789012', 'M', 48000, '2020-11-30', '234567890', 5),

('David', 'Martinez', '567890123', 'M', 53000, '2017-02-14', '345678901', 6),

('Eva', 'Garcia', '678901234', 'F', 75000, '2013-08-19', '234567890', 7),

('Frank', 'Wilson', '789012345', 'M', 80000, '2016-05-25', '345678901', 8),

('Grace', 'Moore', '890123456', 'F', 68000, '2019-01-10', '234567890', 9),

('Henry', 'Taylor', '901234567', 'M', 72000, '2021-07-04', '345678901', 10);

-- Insert records into PROJECT

INSERT INTO PROJECT (pname, pno, plocation, dnumber) VALUES ('Project Alpha', 101, 'New York', 1),

('Project Beta', 102, 'London', 2),

('Project Gamma', 103, 'San Francisco', 3),

('Project Delta', 104, 'Tokyo', 4),

('Project Epsilon', 105, 'Berlin', 5),

('Project Zeta', 106, 'Paris', 6),

('Project Eta', 107, 'Sydney', 7),

('Project Theta', 108, 'Toronto', 8),

('Project Iota', 109, 'Dubai', 9),

('Project Kappa', 110, 'Singapore', 10);

-- Insert records into WORK\_ON

INSERT INTO WORK\_ON (ssn, pno, hours) VALUES

|  |  |  |
| --- | --- | --- |
| ('123456789', | 101, | 40), |
| ('987654321', | 102, | 35), |
| ('234567890', | 103, | 45), |
| ('345678901', | 104, | 30), |
| ('456789012', | 105, | 25), |
| ('567890123', | 106, | 50), |
| ('678901234', | 107, | 38), |
| ('789012345', | 108, | 42), |
| ('890123456', | 109, | 33), |
| ('901234567', | 110, | 37); |
| **🧾 Queries** |  |  |

# Projects located in 'Jalgaon' with controlling department number and manager's last name

SELECT P.pno, P.dnumber AS controlling\_deptno, E.lname AS dept\_manager\_lastname FROM PROJECT P

JOIN DEPT D ON P.dnumber = D.dnum

JOIN EMPLOYEE E ON D.mgrssn = E.ssn WHERE P.plocation = 'Jalgaon';

# Projects with more than two employees working on them, showing project number, name, and number of employees

SELECT W.pno, P.pname, COUNT(DISTINCT W.ssn) AS num\_employees FROM WORK\_ON W

JOIN PROJECT P ON W.pno = P.pno

GROUP BY W.pno, P.pname

HAVING COUNT(DISTINCT W.ssn) > 2;

# Create a view showing department name, manager's name, and manager's salary

CREATE VIEW DeptManagerInfo AS

SELECT D.dname AS department\_name, E.fname + ' ' + E.lname AS manager\_name, E.salary AS manager\_salary

FROM DEPT D

JOIN EMPLOYEE E ON D.mgrssn = E.ssn;

# SQL assertion: Employee's salary must not exceed the salary of their department manager

-- SQL Server does not support direct assertions, so we use a trigger to enforce this constraint

CREATE TRIGGER trg\_check\_salary ON EMPLOYEE

FOR INSERT, UPDATE AS

BEGIN

DECLARE @emp\_ssn CHAR(9), @emp\_salary DECIMAL(10, 2), @mgr\_ssn CHAR(9), @mgr\_salary DECIMAL(10, 2);

SELECT @emp\_ssn = ssn, @emp\_salary = salary FROM inserted;

SELECT @mgr\_ssn = mgrssn FROM DEPT WHERE dnum = (SELECT dno FROM EMPLOYEE WHERE ssn = @emp\_ssn);

SELECT @mgr\_salary = salary FROM EMPLOYEE WHERE ssn = @mgr\_ssn;

IF @emp\_salary > @mgr\_salary BEGIN

RAISERROR ('Employee salary cannot exceed the department manager''s salary.', 16,

1);

ROLLBACK TRANSACTION;

END

END;

13)

## Create Tables

-- DEPT table CREATE TABLE DEPT (

dname VARCHAR(50) NOT NULL,

dnum INT PRIMARY KEY CHECK (dnum < 1000), -- 4-digit limit mgrssn CHAR(9) NOT NULL,

dlocation VARCHAR(50) NOT NULL

);

-- EMPLOYEE table CREATE TABLE EMPLOYEE (

fname VARCHAR(30) NOT NULL,

lname VARCHAR(30) NOT NULL,

ssn CHAR(9) PRIMARY KEY NOT NULL,

sex CHAR(1) CHECK (sex IN ('M', 'F')) NOT NULL,

salary MONEY NOT NULL, joindate DATE NOT NULL, superssn CHAR(9),

dno INT NOT NULL,

FOREIGN KEY (superssn) REFERENCES EMPLOYEE(ssn), FOREIGN KEY (dno) REFERENCES DEPT(dnum)

);

-- PROJECT table CREATE TABLE PROJECT (

pname VARCHAR(50) NOT NULL, pno INT PRIMARY KEY,

plocation VARCHAR(50) NOT NULL, dnumber INT NOT NULL,

FOREIGN KEY (dnumber) REFERENCES DEPT(dnum)

);

-- WORK\_ON table CREATE TABLE WORK\_ON (

ssn CHAR(9) NOT NULL, pno INT NOT NULL,

hours INT NOT NULL, PRIMARY KEY (ssn, pno),

FOREIGN KEY (ssn) REFERENCES EMPLOYEE(ssn), FOREIGN KEY (pno) REFERENCES PROJECT(pno)

);

## ✅Step 2: Insert Sample Data (10+ Records Each Table)

-- DEPT

INSERT INTO DEPT VALUES

('HR', 1, '111111111', 'Delhi'),

('Finance', 2, '222222222', 'Mumbai'),

('IT', 3, '333333333', 'Pune'),

('Sales', 4, '444444444', 'Chennai');

-- EMPLOYEE

INSERT INTO EMPLOYEE VALUES

('John', 'Smith', '111111111', 'M', 50000, '2015-01-01', NULL, 1),

('Sara', 'Khan', '222222222', 'F', 60000, '2016-05-20', '111111111', 2),

('Raj', 'Verma', '333333333', 'M', 55000, '2017-03-10', '222222222', 3),

('Anita', 'Sharma', '444444444', 'F', 52000, '2018-07-15', '333333333', 4),

('Dev', 'Patel', '555555555', 'M', 48000, '2019-08-01', '333333333', 3),

('Kiran', 'Bedi', '666666666', 'F', 47000, '2020-09-10', '111111111', 1),

('Ali', 'Shaikh', '777777777', 'M', 49000, '2017-11-11', '222222222', 2),

('Meera', 'Joshi', '888888888', 'F', 51000, '2021-12-12', '333333333', 3),

('Arjun', 'Kapoor', '999999999', 'M', 53000, '2022-02-22', '444444444', 4),

('Riya', 'Desai', '000000000', 'F', 50000, '2019-10-10', '111111111', 1);

-- PROJECT

INSERT INTO PROJECT VALUES

('Alpha', 101, 'Delhi', 1),

('Beta', 102, 'Mumbai', 2),

('Gamma', 103, 'Pune', 3),

('Delta', 104, 'Chennai', 4),

('Epsilon', 105, 'Bangalore', 3),

('Zeta', 106, 'Hyderabad', 1),

('Theta', 107, 'Ahmedabad', 2),

('Iota', 108, 'Jaipur', 3),

('Kappa', 109, 'Kolkata', 4),

('Lambda', 110, 'Surat', 3);

-- WORK\_ON

INSERT INTO WORK\_ON VALUES

|  |  |  |
| --- | --- | --- |
| ('111111111', | 101, | 10), |
| ('222222222', | 102, | 8), |
| ('333333333', | 103, | 12), |
| ('444444444', | 104, | 6), |
| ('555555555', | 103, | 5), |
| ('666666666', | 106, | 7), |
| ('777777777', | 102, | 6), |
| ('888888888', | 105, | 9), |
| ('999999999', | 109, | 4), |
| ('000000000', | 101, | 3), |
| ('333333333', | 105, | 7), |
| ('555555555', | 105, | 6), |
| ('333333333', | 110, | 8), |
| ('555555555', | 110, | 5), |
| ('888888888', | 110, | 4); |
| **🧾 Queries** |  |  |

# Employee and their Supervisor’s Names

SELECT

E.fname + ' ' + E.lname AS Employee\_Name, S.fname + ' ' + S.lname AS Supervisor\_Name

FROM EMPLOYEE E

LEFT JOIN EMPLOYEE S ON E.superssn = S.ssn;

# Department → Number of Employees and Average Salary

SELECT

dno AS DeptNo,

COUNT(\*) AS Num\_Employees, AVG(salary) AS Avg\_Salary

FROM EMPLOYEE GROUP BY dno;

# View with Project Details (Only Projects with > 1 Employee)

CREATE VIEW ProjectSummary AS SELECT

P.pname,

D.dname AS Dept\_Name, COUNT(W.ssn) AS Num\_Employees, SUM(W.hours) AS Total\_Hours

FROM PROJECT P

JOIN WORK\_ON W ON P.pno = W.pno

JOIN DEPT D ON P.dnumber = D.dnum GROUP BY P.pname, D.dname

HAVING COUNT(W.ssn) > 1;

# Procedure: Employees Eligible for Promotion (Worked on 5+ Projects)

CREATE PROCEDURE GetPromotableEmployees AS

BEGIN

SELECT

E.fname + ' ' + E.lname AS Employee\_Name, COUNT(W.pno) AS Projects\_Worked

FROM EMPLOYEE E

JOIN WORK\_ON W ON E.ssn = W.ssn

GROUP BY E.fname, E.lname, E.ssn HAVING COUNT(W.pno) >= 5;

END;

### To run the procedure:

EXEC GetPromotableEmployees;

14)

## Create Tables with Integrity Constraints

-- DEPT Table CREATE TABLE DEPT (

dname VARCHAR(50) NOT NULL,

dnum INT PRIMARY KEY CHECK (dnum < 1000), mgrssn CHAR(9) NOT NULL,

dlocation VARCHAR(50) NOT NULL

);

-- EMPLOYEE Table CREATE TABLE EMPLOYEE (

fname VARCHAR(30) NOT NULL, lname VARCHAR(30) NOT NULL,

ssn CHAR(9) PRIMARY KEY NOT NULL,

sex CHAR(1) CHECK (sex IN ('M', 'F')) NOT NULL,

salary MONEY NOT NULL, joindate DATE NOT NULL, superssn CHAR(9),

dno INT NOT NULL,

FOREIGN KEY (superssn) REFERENCES EMPLOYEE(ssn), FOREIGN KEY (dno) REFERENCES DEPT(dnum)

);

-- PROJECT Table CREATE TABLE PROJECT (

pname VARCHAR(50) NOT NULL, pno INT PRIMARY KEY,

plocation VARCHAR(50) NOT NULL, dnumber INT NOT NULL,

FOREIGN KEY (dnumber) REFERENCES DEPT(dnum)

);

-- WORK\_ON Table CREATE TABLE WORK\_ON (

ssn CHAR(9) NOT NULL, pno INT NOT NULL,

hours INT NOT NULL, PRIMARY KEY (ssn, pno),

FOREIGN KEY (ssn) REFERENCES EMPLOYEE(ssn), FOREIGN KEY (pno) REFERENCES PROJECT(pno)

);

## ✅Step 2: Insert Sample Data (10+ Rows Each Table)

-- DEPT

INSERT INTO DEPT VALUES

('HR', 1, '111111111', 'Delhi'),

('Finance', 2, '222222222', 'Mumbai'),

('IT', 3, '333333333', 'Pune'),

('Sales', 4, '444444444', 'Chennai');

-- EMPLOYEE

INSERT INTO EMPLOYEE VALUES

('John', 'Smith', '111111111', 'M', 50000, '2015-01-01', NULL, 1),

('Sara', 'Khan', '222222222', 'F', 60000, '2016-05-20', '111111111', 2),

('Raj', 'Verma', '333333333', 'M', 55000, '2017-03-10', '222222222', 3),

('Anita', 'Sharma', '444444444', 'F', 52000, '2018-07-15', '333333333', 4),

('Dev', 'Sonar', '555555555', 'M', 48000, '2019-08-01', '333333333', 3),

('Kiran', 'Bedi', '666666666', 'F', 47000, '2020-09-10', '111111111', 1),

('Ali', 'Shaikh', '777777777', 'M', 49000, '2017-11-11', '222222222', 2),

('Meera', 'Joshi', '888888888', 'F', 51000, '2021-12-12', '333333333', 3),

('Arjun', 'Kapoor', '999999999', 'M', 53000, '2022-02-22', '444444444', 4),

('Riya', 'Sonar', '000000000', 'F', 50000, '2019-10-10', '111111111', 1);

-- PROJECT

INSERT INTO PROJECT VALUES

('Alpha', 101, 'Delhi', 1),

('Beta', 102, 'Mumbai', 2),

('Gamma', 103, 'Pune', 3),

('Delta', 104, 'Chennai', 4),

('Epsilon', 105, 'Pune', 3),

('Zeta', 106, 'Delhi', 1),

('Theta', 107, 'Mumbai', 2),

('Iota', 108, 'Pune', 3),

('Kappa', 109, 'Chennai', 4),

('Lambda', 110, 'Surat', 3);

-- WORK\_ON

INSERT INTO WORK\_ON VALUES

|  |  |  |
| --- | --- | --- |
| ('555555555', | 101, | 5), |
| ('555555555', | 102, | 4), |
| ('555555555', | 103, | 4), |
| ('222222222', | 102, | 6), |
| ('333333333', | 103, | 8), |
| ('444444444', | 104, | 7), |
| ('000000000', | 101, | 2), |
| ('000000000', | 102, | 2), |
| ('000000000', | 103, | 2), |
| ('000000000', | 104, | 2), |
| ('000000000', | 105, | 1); |
| **🧾 Queries** |  |  |

# Find SSNs of employees who work on pno 101, 102, or 103

SELECT ssn FROM WORK\_ON

WHERE pno IN (101, 102, 103);

# List all pno for projects that involve an employee whose last name is 'Sonar', either as a worker or manager of the dept

-- Projects involving a 'Sonar' employee directly SELECT DISTINCT pno

FROM WORK\_ON W

JOIN EMPLOYEE E ON W.ssn = E.ssn

WHERE E.lname = 'Sonar' UNION

-- Projects managed by a 'Sonar' manager SELECT DISTINCT P.pno

FROM PROJECT P

JOIN DEPT D ON P.dnumber = D.dnum JOIN EMPLOYEE M ON D.mgrssn = M.ssn WHERE M.lname = 'Sonar';

# Trigger: Deduct salary if total work hours < 20

#### For simplicity, let's deduct 1000 from salary

CREATE TRIGGER trg\_DeductSalary ON WORK\_ON

AFTER INSERT AS

BEGIN

DECLARE @ssn CHAR(9);

SELECT TOP 1 @ssn = ssn FROM inserted;

DECLARE @total\_hours INT;

SELECT @total\_hours = SUM(hours) FROM WORK\_ON WHERE ssn = @ssn;

IF @total\_hours < 20 BEGIN

UPDATE EMPLOYEE

SET salary = salary - 1000 WHERE ssn = @ssn;

END END;

# Cursor: Fetch first row of PROJECT and count all rows

DECLARE @pname VARCHAR(50), @pno INT; DECLARE project\_cursor CURSOR FOR

SELECT pno, pname FROM PROJECT;

OPEN project\_cursor;

FETCH NEXT FROM project\_cursor INTO @pno, @pname;

PRINT 'First Project: ' + CAST(@pno AS VARCHAR) + ' - ' + @pname;

-- Count total rows DECLARE @rowCount INT;

SELECT @rowCount = COUNT(\*) FROM PROJECT;

PRINT 'Total Projects: ' + CAST(@rowCount AS VARCHAR);

CLOSE project\_cursor; DEALLOCATE project\_cursor;

15)

# Create Tables with Constraints

-- BOOKMASTER

CREATE TABLE BOOKMASTER ( bid INT PRIMARY KEY,

title VARCHAR(100) NOT NULL, author VARCHAR(100) NOT NULL, price MONEY NOT NULL

);

-- STUDENTMASTER

CREATE TABLE STUDENTMASTER (

stud\_enrollno INT PRIMARY KEY, sname VARCHAR(100) NOT NULL, class VARCHAR(20) NOT NULL, dept VARCHAR(50) NOT NULL

);

-- ACCESSIONTABLE

CREATE TABLE ACCESSIONTABLE ( bid INT NOT NULL,

accession\_no INT PRIMARY KEY,

avail CHAR(1) NOT NULL CHECK (avail IN ('T', 'F')), FOREIGN KEY (bid) REFERENCES BOOKMASTER(bid)

);

-- ISSUETABLE

CREATE TABLE ISSUETABLE (

issueid INT PRIMARY KEY, accession\_no INT NOT NULL, stud\_enrollno INT NOT NULL, issuedate DATE NOT NULL, duedate DATE NOT NULL, ret\_date DATE,

bid INT NOT NULL,

FOREIGN KEY (accession\_no) REFERENCES ACCESSIONTABLE(accession\_no), FOREIGN KEY (stud\_enrollno) REFERENCES STUDENTMASTER(stud\_enrollno), FOREIGN KEY (bid) REFERENCES BOOKMASTER(bid)

);

# ✅Step 2: Insert Sample Records (10+ per table)

-- BOOKMASTER

INSERT INTO BOOKMASTER VALUES

(1, 'Database Systems', 'Elmasri', 500),

(2, 'Operating Systems', 'Galvin', 600),

(3, 'Networking Basics', 'Tanenbaum', 450),

(4, 'C Programming', 'Dennis Ritchie', 300),

(5, 'Python Programming', 'Guido van Rossum', 550), (6, 'Java Complete Ref', 'Schildt', 700),

(7, 'Web Technologies', 'Achyut Godbole', 400),

(8, 'AI Basics', 'Stuart Russell', 800),

(9, 'ML Concepts', 'Andrew Ng', 900),

(10, 'Cloud Computing', 'Rajkumar Buyya', 750);

-- STUDENTMASTER

INSERT INTO STUDENTMASTER VALUES

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| (101, | 'Ravi Patel', 'SY', 'Computer'), | | | |
| (102, | 'Sneha Shah', 'FY', 'Electronics'), | | | |
| (103, | 'Arjun Mehta', 'TY', 'Computer'), | | | |
| (104, | 'Meera Joshi', 'SY', 'IT'), | | | |
| (105, | 'Kunal Shah', 'TY', 'Computer'), | | | |
| (106, | 'Nisha Verma', 'FY', 'Mechanical'), | | | |
| (107, | 'Pooja Singh', 'SY', 'Computer'), | | | |
| (108, | 'Anil Rao', 'TY', 'Civil'), | | | |
| (109, | 'Dinesh Jain', 'SY', 'Computer'), | | | |
| (110, | 'Seema Desai', 'FY', 'Computer'); | | | |
| -- ACCESSIONTABLE  INSERT INTO ACCESSIONTABLE VALUES  (1, 1001, 'F'), (1, 1002, 'T'), (2, 1003, 'F'), (3, 1004, 'F'), | | (4, | 1005, | 'T'), |
| (5, 1006, 'F'), (6, 1007, 'T'), (7, 1008, 'F'), (8, 1009, 'F'), | | (9, | 1010, | 'F'); |
| -- ISSUETABLE  INSERT INTO ISSUETABLE VALUES | |  |  |  |

(10, 1006, 109, '2024-05-01', '2024-05-08', NULL, 5);

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| (1, | 1001, | 101, '2024-05-01', | '2024-05-08', '2024-05-10', | 1), |
| (2, | 1003, | 103, '2024-05-03', | '2024-05-10', '2024-05-09', | 2), |
| (3, | 1004, | 105, '2024-05-04', | '2024-05-11', NULL, 3), |  |
| (4, | 1006, | 107, '2024-05-02', | '2024-05-09', '2024-05-10', | 5), |
| (5, | 1008, | 109, '2024-05-01', | '2024-05-08', NULL, 7), |  |
| (6, | 1009, | 110, '2024-05-03', | '2024-05-10', '2024-05-20', | 8), |
| (7, | 1010, | 101, '2024-05-05', | '2024-05-12', NULL, 9), |  |
| (8, | 1001, | 103, '2024-04-15', | '2024-04-22', '2024-04-20', | 1), |
| (9, | 1004, | 105, '2024-04-20', | '2024-04-27', '2024-05-01', | 3), |

**🧾 Queries**

# Find the name of the book issued maximum times

SELECT TOP 1 BM.title, COUNT(\*) AS issue\_count FROM ISSUETABLE IT

JOIN BOOKMASTER BM ON IT.bid = BM.bid

GROUP BY BM.title

ORDER BY issue\_count DESC;

# Detail of books issued by Computer Department students

SELECT IT.\*, BM.title, SM.sname, SM.dept FROM ISSUETABLE IT

JOIN STUDENTMASTER SM ON IT.stud\_enrollno = SM.stud\_enrollno JOIN BOOKMASTER BM ON IT.bid = BM.bid

WHERE SM.dept = 'Computer';

# Procedure to calculate fines for overdue books

CREATE PROCEDURE CalculateFines AS

BEGIN

SELECT

issueid, stud\_enrollno,

DATEDIFF(DAY, duedate, ret\_date) AS overdue\_days, CASE

WHEN ret\_date > duedate THEN DATEDIFF(DAY, duedate, ret\_date) \* 10 ELSE 0

END AS fine FROM ISSUETABLE

WHERE ret\_date IS NOT NULL; END;

### Run it using:

EXEC CalculateFines;

# Trigger to auto-set duedate = issuedate + 7 on insert

CREATE TRIGGER trg\_SetDueDate ON ISSUETABLE

INSTEAD OF INSERT AS

BEGIN

INSERT INTO ISSUETABLE (issueid, accession\_no, stud\_enrollno, issuedate, duedate, ret\_date, bid)

SELECT

issueid, accession\_no, stud\_enrollno, issuedate,

DATEADD(DAY, 7, issuedate), -- due date = issue date + 7 ret\_date,

bid

FROM INSERTED;

END;

16)

## Create Tables with Integrity Constraints

-- BOOKMASTER Table CREATE TABLE BOOKMASTER (

bid INT PRIMARY KEY,

title VARCHAR(100) NOT NULL, author VARCHAR(100) NOT NULL, price DECIMAL(10, 2) NOT NULL

);

-- STUDENTMASTER Table CREATE TABLE STUDENTMASTER (

stud\_enrollno INT PRIMARY KEY, sname VARCHAR(100) NOT NULL, class VARCHAR(20) NOT NULL, dept VARCHAR(50) NOT NULL

);

-- ACCESSIONTABLE Table CREATE TABLE ACCESSIONTABLE (

accession\_no INT PRIMARY KEY,

avail CHAR(1) NOT NULL CHECK (avail IN ('T', 'F'))

);

-- ISSUETABLE Table CREATE TABLE ISSUETABLE (

issueid INT PRIMARY KEY, accession\_no INT NOT NULL, stud\_enrollno INT NOT NULL, issuedate DATE NOT NULL, duedate DATE NOT NULL, ret\_date DATE,

bid INT NOT NULL,

FOREIGN KEY (accession\_no) REFERENCES ACCESSIONTABLE(accession\_no), FOREIGN KEY (stud\_enrollno) REFERENCES STUDENTMASTER(stud\_enrollno), FOREIGN KEY (bid) REFERENCES BOOKMASTER(bid)

);

## 🧾 Step 2: Insert Sample Records (10+ per table)

-- Insert records into BOOKMASTER INSERT INTO BOOKMASTER VALUES

(1, 'Database Systems', 'Elmasri', 500),

(2, 'Operating Systems', 'Galvin', 600),

(3, 'Networking Basics', 'Tanenbaum', 450),

(4, 'C Programming', 'Dennis Ritchie', 300),

(5, 'Python Programming', 'Guido van Rossum', 550), (6, 'Java Complete Ref', 'Schildt', 700),

(7, 'Web Technologies', 'Achyut Godbole', 400),

(8, 'AI Basics', 'Stuart Russell', 800),

(9, 'ML Concepts', 'Andrew Ng', 900),

(10, 'Cloud Computing', 'Rajkumar Buyya', 750);

-- Insert records into STUDENTMASTER INSERT INTO STUDENTMASTER VALUES

|  |  |
| --- | --- |
| (101, | 'Ravi Patel', 'SY', 'Computer'), |
| (102, | 'Sneha Shah', 'FY', 'Electronics'), |
| (103, | 'Arjun Mehta', 'TY', 'Computer'), |
| (104, | 'Meera Joshi', 'SY', 'IT'), |
| (105, | 'Kunal Shah', 'TY', 'Computer'), |
| (106, | 'Nisha Verma', 'FY', 'Mechanical'), |
| (107, | 'Pooja Singh', 'SY', 'Computer'), |
| (108, | 'Anil Rao', 'TY', 'Civil'), |
| (109, | 'Dinesh Jain', 'SY', 'Computer'), |
| (110, | 'Seema Desai', 'FY', 'Computer'); |

-- Insert records into ACCESSIONTABLE INSERT INTO ACCESSIONTABLE VALUES

(1001, 'F'), (1002, 'T'), (1003, 'F'), (1004, 'F'), (1005, 'T'),

(1006, 'F'), (1007, 'T'), (1008, 'F'), (1009, 'F'), (1010, 'F');

-- Insert records into ISSUETABLE INSERT INTO ISSUETABLE VALUES

|  |  |  |
| --- | --- | --- |
| (1, 1001, 101, '2024-05-01', | '2024-05-08', '2024-05-10', | 1), |
| (2, 1003, 103, '2024-05-03', | '2024-05-10', '2024-05-09', | 2), |
| (3, 1004, 105, '2024-05-04', | '2024-05-11', NULL, 3), |  |
| (4, 1006, 107, '2024-05-02', | '2024-05-09', '2024-05-10', | 5), |
| (5, 1008, 109, '2024-05-01', | '2024-05-08', NULL, 7), |  |
| (6, 1009, 110, '2024-05-03', | '2024-05-10', '2024-05-20', | 8), |
| (7, 1010, 101, '2024-05-05', | '2024-05-12', NULL, 9), |  |
| (8, 1001, 103, '2024-04-15', | '2024-04-22', '2024-04-20', | 1), |
| (9, 1004, 105, '2024-04-20', | '2024-04-27', '2024-05-01', | 3), |

(10, 1006, 109, '2024-05-01', '2024-05-08', NULL, 5);

**🧾 Queries**

# Find the detail information of the students who have issued books between two given dates.

SELECT SM.\*

FROM STUDENTMASTER SM

JOIN ISSUETABLE IT ON SM.stud\_enrollno = IT.stud\_enrollno WHERE IT.issuedate BETWEEN '2024-04-01' AND '2024-05-01';

# Create a view that displays all the accession information for a book having bid = 100.

CREATE VIEW BookAccessions AS SELECT AT.accession\_no, AT.avail FROM ACCESSIONTABLE AT

JOIN BOOKMASTER BM ON AT.bid = BM.bid WHERE BM.bid = 100;

# Write a cursor to fetch the last record from the view in (b).

DECLARE @accession\_no INT, @avail CHAR(1);

DECLARE book\_cursor CURSOR FOR SELECT accession\_no, avail FROM BookAccessions

ORDER BY accession\_no DESC; OPEN book\_cursor;

FETCH NEXT FROM book\_cursor INTO @accession\_no, @avail;

-- Fetch the last record

FETCH NEXT FROM book\_cursor INTO @accession\_no, @avail;

CLOSE book\_cursor;

DEALLOCATE book\_cursor;

# Find the information of books issued by MCA students.

SELECT BM.\*

FROM BOOKMASTER BM

JOIN ISSUETABLE IT ON BM.bid = IT.bid

JOIN STUDENTMASTER SM ON IT.stud\_enrollno = SM.stud\_enrollno WHERE SM.dept = 'MCA';

17)

## Create Tables With Integrity Constraints

-- BOOKMASTER Table CREATE TABLE BOOKMASTER (

bid INT PRIMARY KEY,

title VARCHAR(100) NOT NULL, author VARCHAR(100) NOT NULL,

price DECIMAL(10,2) NOT NULL

);

-- STUDENTMASTER Table CREATE TABLE STUDENTMASTER (

stud\_enrollno INT PRIMARY KEY, sname VARCHAR(100) NOT NULL, class VARCHAR(10) NOT NULL, dept VARCHAR(50) NOT NULL

);

-- ACCESSIONTABLE Table CREATE TABLE ACCESSIONTABLE (

bid INT NOT NULL,

accession\_no INT PRIMARY KEY,

avail CHAR(1) NOT NULL CHECK (avail IN ('T', 'F')), FOREIGN KEY (bid) REFERENCES BOOKMASTER(bid)

);

-- SUETABLE (ISSUETABLE) CREATE TABLE SUETABLE (

issueid INT PRIMARY KEY, accession\_no INT NOT NULL, stud\_enrollno INT NOT NULL, issuedate DATE NOT NULL, duedate DATE NOT NULL, ret\_date DATE NOT NULL, bid INT NOT NULL,

FOREIGN KEY (accession\_no) REFERENCES ACCESSIONTABLE(accession\_no), FOREIGN KEY (stud\_enrollno) REFERENCES STUDENTMASTER(stud\_enrollno), FOREIGN KEY (bid) REFERENCES BOOKMASTER(bid)

);

## 🧾 Step 2: Insert Sample Records (10+)

-- BOOKMASTER

INSERT INTO BOOKMASTER VALUES

(1, 'Database System Concepts', 'Henry Korth', 700), (2, 'Operating Systems', 'Silberschatz', 650),

(3, 'Networking', 'Tanenbaum', 550),

(4, 'Java Basics', 'Schildt', 600),

(5, 'Python 101', 'Guido Rossum', 500),

(6, 'Cloud Computing', 'Buyya', 750),

(7, 'AI Basics', 'Stuart Russell', 800),

(8, 'Software Engineering', 'Pressman', 650),

(9, 'C Programming', 'Dennis Ritchie', 450),

(10, 'DBMS Fundamentals', 'Henry Korth', 720);

-- STUDENTMASTER

INSERT INTO STUDENTMASTER VALUES

|  |  |
| --- | --- |
| (101, | 'Amit', 'FY', 'Computer'), |
| (102, | 'Riya', 'SY', 'IT'), |
| (103, | 'Neha', 'TY', 'Computer'), |
| (104, | 'Karan', 'FY', 'Computer'), |
| (105, | 'Priya', 'SY', 'Computer'), |
| (106, | 'Vikram', 'TY', 'IT'), |
| (107, | 'Sneha', 'FY', 'Computer'), |
| (108, | 'Ravi', 'SY', 'Computer'), |
| (109, | 'Pooja', 'TY', 'Computer'), |
| (110, | 'Nikhil', 'SY', 'IT'); |

-- ACCESSIONTABLE

INSERT INTO ACCESSIONTABLE VALUES

(1, 1001, 'F'), (2, 1002, 'F'), (3, 1003, 'T'), (4, 1004, 'F'), (5, 1005, 'F'),

(6, 1006, 'T'), (7, 1007, 'F'), (8, 1008, 'F'), (9, 1009, 'T'), (10, 1010, 'F');

-- SUETABLE

INSERT INTO SUETABLE VALUES

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| (1, | 1001, | 101, | '2024-05-01', | '2024-05-08', '2024-05-07', 1), |
| (2, | 1002, | 102, | '2024-05-02', | '2024-05-09', '2024-05-08', 2), |
| (3, | 1004, | 103, | '2024-05-01', | '2024-05-07', '2024-05-07', 4), |
| (4, | 1005, | 104, | '2024-05-03', | '2024-05-10', '2024-05-09', 5), |
| (5, | 1007, | 105, | '2024-05-02', | '2024-05-08', '2024-05-06', 7), |
| (6, | 1008, | 106, | '2024-05-04', | '2024-05-10', '2024-05-10', 8), |
| (7, | 1010, | 107, | '2024-05-05', | '2024-05-12', '2024-05-11', 10), |
| (8, | 1001, | 108, | '2024-05-01', | '2024-05-08', '2024-05-08', 1), |
| (9, | 1002, | 109, | '2024-05-01', | '2024-05-08', '2024-05-07', 2), |

(10, 1004, 110, '2024-05-01', '2024-05-08', '2024-05-08', 4);

✅**Queries and Procedures**

# Procedure: List available books in library

CREATE PROCEDURE AvailableBooks AS

BEGIN

SELECT BM.bid, BM.title, BM.author, BM.price, AT.accession\_no FROM BOOKMASTER BM

JOIN ACCESSIONTABLE AT ON BM.bid = AT.bid WHERE AT.avail = 'T';

END;

#### Run using:

EXEC AvailableBooks;

# Number of books issued by each student

SELECT SM.sname, COUNT(\*) AS books\_issued FROM SUETABLE ST

JOIN STUDENTMASTER SM ON ST.stud\_enrollno = SM.stud\_enrollno GROUP BY SM.sname;

# Trigger: Ensure return date does not exceed today

CREATE TRIGGER trg\_ReturnDateCheck ON SUETABLE

AFTER INSERT AS

BEGIN

IF EXISTS (

SELECT 1 FROM INSERTED WHERE ret\_date > CAST(GETDATE() AS DATE)

) BEGIN

RAISERROR ('Return date cannot be in the future.', 16, 1); ROLLBACK;

END

END;

# Count books available written by "Henry Korth"

SELECT COUNT(\*) AS Available\_By\_Korth FROM ACCESSIONTABLE AT

JOIN BOOKMASTER BM ON AT.bid = BM.bid

WHERE AT.avail = 'T' AND BM.author = 'Henry Korth';

18)

Same as it is 12 number slip. 19)

Same as it is 11 number slip. 20)

# Create Tables with Integrity Constraints

CREATE DATABASE ProductDB; GO

USE ProductDB;

GO

-- PRODUCT Table CREATE TABLE PRODUCT (

Maker VARCHAR(50) NOT NULL, Modelno INT PRIMARY KEY NOT NULL,

Type VARCHAR(10) NOT NULL CHECK (Type IN ('PC', 'Laptop', 'Printer'))

);

-- PC Table CREATE TABLE PC (

Modelno INT PRIMARY KEY NOT NULL, Speed FLOAT NOT NULL,

RAM INT NOT NULL, HD INT NOT NULL,

CD VARCHAR(10) NOT NULL, Price INT NOT NULL,

FOREIGN KEY (Modelno) REFERENCES PRODUCT(Modelno)

);

-- LAPTOP Table CREATE TABLE LAPTOP (

Modelno INT PRIMARY KEY NOT NULL, Speed FLOAT NOT NULL,

RAM INT NOT NULL, HD INT NOT NULL, Price INT NOT NULL,

FOREIGN KEY (Modelno) REFERENCES PRODUCT(Modelno)

);

-- PRINTER Table CREATE TABLE PRINTER (

Modelno INT PRIMARY KEY NOT NULL,

Color CHAR(1) NOT NULL CHECK (Color IN ('T', 'F')),

Type VARCHAR(20) NOT NULL CHECK (Type IN ('laser', 'ink-jet', 'dot-matrix', 'dry')), Price INT NOT NULL,

FOREIGN KEY (Modelno) REFERENCES PRODUCT(Modelno)

);

# ◻ 2. Insert Sample Records

-- PRODUCT Records

INSERT INTO PRODUCT VALUES

('IBM', 1001, 'PC'), ('Compaq', 1002, 'PC'), ('Dell', 1003, 'PC'),

('HP', 1004, 'PC'), ('Lenovo', 1005, 'PC'), ('IBM', 2001, 'Laptop'),

('Compaq', 2002, 'Laptop'), ('Dell', 2003, 'Laptop'), ('HP', 2004, 'Laptop'),

('Lenovo', 2005, 'Laptop'), ('IBM', 3001, 'Printer'), ('Compaq', 3002, 'Printer'),

('Dell', 3003, 'Printer'), ('HP', 3004, 'Printer'), ('Lenovo', 3005, 'Printer');

-- PC Records

INSERT INTO PC VALUES

(1001, 2.5, 4096, 250, '16x', 500), (1002, 3.0, 8192, 500, '24x', 700),

(1003, 2.2, 4096, 320, '8x', 600), (1004, 2.8, 8192, 500, '16x', 750),

(1005, 3.2, 16384, 1000, '24x', 900);

-- LAPTOP Records

INSERT INTO LAPTOP VALUES

(2001, 1.8, 2048, 128, 600), (2002, 2.2, 4096, 256, 800),

(2003, 2.0, 2048, 128, 650), (2004, 2.1, 4096, 256, 750),

(2005, 2.5, 8192, 512, 950);

-- PRINTER Records

INSERT INTO PRINTER VALUES

(3001, 'T', 'laser', 300), (3002, 'F', 'ink-jet', 200),

(3003, 'T', 'dot-matrix', 350), (3004, 'F', 'laser', 150),

(3005, 'T', 'dry', 400);

# ◻ 3. SQL Queries

### Manufacturers of color printers:

SELECT DISTINCT P.Maker FROM PRODUCT P

JOIN PRINTER R ON P.Modelno = R.Modelno WHERE R.Color = 'T';

### Laptops slower than any PC:

SELECT \* FROM LAPTOP L

WHERE L.Speed < ALL (SELECT Speed FROM PC);

### SQL Assertion: No black & white printer should be more expensive than any color printer

**Note**: SQL Server does not support CREATE ASSERTION. Instead, use a **trigger** or **check logic** within a stored procedure or constraint enforcement logic at the application level.

#### Here’s how it can be simulated using a **trigger**:

CREATE TRIGGER Check\_Printer\_Price ON PRINTER

AFTER INSERT, UPDATE AS

BEGIN

IF EXISTS (

SELECT 1

FROM PRINTER bw, PRINTER color

WHERE bw.Color = 'F' AND color.Color = 'T' AND bw.Price > color.Price

1);

) BEGIN

RAISERROR ('Black & white printer price cannot exceed color printer price.', 16,

ROLLBACK;

END END;

### Trigger on PC & LAPTOP to enforce HD > 20 GB

-- For PC

CREATE TRIGGER Check\_HD\_PC ON PC

AFTER INSERT, UPDATE AS

BEGIN

IF EXISTS (SELECT \* FROM inserted WHERE HD <= 20) BEGIN

RAISERROR ('PC HD size must be greater than 20 GB.', 16, 1); ROLLBACK;

END END;

-- For LAPTOP

CREATE TRIGGER Check\_HD\_LAPTOP ON LAPTOP

AFTER INSERT, UPDATE AS

BEGIN

IF EXISTS (SELECT \* FROM inserted WHERE HD <= 20) BEGIN

RAISERROR ('Laptop HD size must be greater than 20 GB.', 16, 1); ROLLBACK;

END END;

21)

Same as it is sleep number 20. 22)

## Create Tables with Integrity Constraints

CREATE DATABASE ProductDB2; GO

USE ProductDB2;

GO

-- PRODUCT table CREATE TABLE PRODUCT (

Maker VARCHAR(50) NOT NULL, Modelno INT PRIMARY KEY NOT NULL,

Type VARCHAR(10) NOT NULL CHECK (Type IN ('PC', 'Laptop', 'Printer'))

);

-- PC table CREATE TABLE PC (

Modelno INT PRIMARY KEY NOT NULL, Speed FLOAT NOT NULL,

RAM INT NOT NULL, HD INT NOT NULL,

CD VARCHAR(10) NOT NULL, Price INT NOT NULL,

FOREIGN KEY (Modelno) REFERENCES PRODUCT(Modelno)

);

-- LAPTOP table CREATE TABLE LAPTOP (

Modelno INT PRIMARY KEY NOT NULL,

Speed FLOAT NOT NULL CHECK (Speed >= 120), RAM INT NOT NULL,

HD INT NOT NULL, Price INT NOT NULL,

FOREIGN KEY (Modelno) REFERENCES PRODUCT(Modelno)

);

-- PRINTER table CREATE TABLE PRINTER (

Modelno INT PRIMARY KEY NOT NULL,

Color CHAR(1) NOT NULL CHECK (Color IN ('T', 'F')),

Type VARCHAR(20) NOT NULL CHECK (Type IN ('laser', 'ink-jet', 'dot-matrix', 'dry')), Price INT NOT NULL,

FOREIGN KEY (Modelno) REFERENCES PRODUCT(Modelno)

);

## ✅2. Insert Sample Data

-- PRODUCT

INSERT INTO PRODUCT VALUES

('IBM', 101, 'PC'), ('Compaq', 102, 'PC'), ('Dell', 103, 'PC'), ('HP', 104, 'PC'),

('Lenovo', 105, 'PC'),

('IBM', 201, 'Laptop'), ('Compaq', 202, 'Laptop'), ('Dell', 203, 'Laptop'), ('HP', 204,

'Laptop'), ('Lenovo', 205, 'Laptop'),

('Epson', 301, 'Printer'), ('Epson', 302, 'Printer'), ('Canon', 303, 'Printer'), ('HP', 304, 'Printer'), ('Brother', 305, 'Printer');

-- PC

INSERT INTO PC VALUES

(101, 220.5, 4096, 250, '16x', 550),

(102, 300.0, 8192, 500, '24x', 800),

(103, 250.0, 4096, 250, '16x', 600),

(104, 280.0, 8192, 500, '8x', 750),

(105, 320.0, 16384, 1000, '24x', 950);

-- LAPTOP

INSERT INTO LAPTOP VALUES (201, 180.0, 2048, 128, 600),

(202, 220.0, 4096, 256, 850),

(203, 150.0, 2048, 128, 650),

(204, 210.0, 4096, 512, 950),

(205, 240.0, 8192, 1024, 1200);

-- PRINTER

INSERT INTO PRINTER VALUES (301, 'T', 'laser', 300),

(302, 'F', 'ink-jet', 200),

(303, 'T', 'dot-matrix', 350),

(304, 'F', 'laser', 150),

(305, 'T', 'dry', 400);

**🧾 3. Queries**

# Find the different types of printers produced by Epson:

SELECT DISTINCT PR.Type FROM PRODUCT P

JOIN PRINTER PR ON P.Modelno = PR.Modelno WHERE P.Maker = 'Epson';

# Find those hard disk sizes which occur in two or more PCs:

SELECT HD FROM PC GROUP BY HD

HAVING COUNT(\*) >= 2;

# Trigger on LAPTOP: Speed must be ≥ 120 MHz

CREATE TRIGGER Check\_Laptop\_Speed ON LAPTOP

AFTER INSERT, UPDATE AS

BEGIN

IF EXISTS (SELECT \* FROM inserted WHERE Speed < 120) BEGIN

RAISERROR ('Laptop speed must be at least 120 MHz.', 16, 1); ROLLBACK;

END END;

Note: We also added a CHECK (Speed >= 120) constraint in table definition for extra safety.

# Cursor Example: Loop through PRODUCTs and print details

DECLARE @Maker VARCHAR(50), @Modelno INT, @Type VARCHAR(10);

DECLARE product\_cursor CURSOR FOR

SELECT Maker, Modelno, Type FROM PRODUCT; OPEN product\_cursor;

FETCH NEXT FROM product\_cursor INTO @Maker, @Modelno, @Type;

WHILE @@FETCH\_STATUS = 0 BEGIN

PRINT 'Maker: ' + @Maker + ', Modelno: ' + CAST(@Modelno AS VARCHAR) + ', Type: ' + @Type;

FETCH NEXT FROM product\_cursor INTO @Maker, @Modelno, @Type; END;

CLOSE product\_cursor; DEALLOCATE product\_cursor;

Or

# Trigger on LAPTOP Table – Minimum Speed 1200 MHz

CREATE OR REPLACE FUNCTION check\_laptop\_speed() RETURNS TRIGGER AS $$

BEGIN

IF NEW.Speed < 1200 THEN

RAISE EXCEPTION 'Laptop speed must be at least 1200 MHz'; END IF;

RETURN NEW; END;

$$ LANGUAGE plpgsql;

CREATE TRIGGER trg\_laptop\_min\_speed BEFORE INSERT OR UPDATE ON LAPTOP FOR EACH ROW

EXECUTE FUNCTION check\_laptop\_speed();

# Cursor on PRODUCT Table (e.g., Print all product details)

DO $$ DECLARE

rec RECORD;

cur CURSOR FOR SELECT \* FROM PRODUCT; BEGIN

OPEN cur;

LOOP

FETCH cur INTO rec;

EXIT WHEN NOT FOUND;

RAISE NOTICE 'Maker: %, Model: %, Type: %', rec.Maker, rec.Modelno, rec.Type; END LOOP;

CLOSE cur;

END;

$$;

23)

Same as it slip number 22. 24)

Same as it is 2 number slip.

25)

## Create Database and Tables with Constraints

CREATE DATABASE HospitalDB; GO

USE HospitalDB;

GO

-- DOCTOR Table CREATE TABLE DOCTOR (

Did INT PRIMARY KEY NOT NULL, Dname VARCHAR(100) NOT NULL, Daddress VARCHAR(200) NOT NULL,

qualification VARCHAR(20) NOT NULL CHECK (qualification IN ('M.B.B.S.', 'B.A.M.S', 'M.S.'))

);

-- PATIENTMASTER Table CREATE TABLE PATIENTMASTER (

Pcode INT PRIMARY KEY NOT NULL, Pname VARCHAR(100) NOT NULL, Padd VARCHAR(200) NOT NULL, age INT NOT NULL,

gender CHAR(1) NOT NULL CHECK (gender IN ('M', 'F')),

bloodgroup VARCHAR(5) NOT NULL, Did INT NOT NULL,

FOREIGN KEY (Did) REFERENCES DOCTOR(Did)

);

-- ADMITTEDPATIENT Table CREATE TABLE ADMITTEDPATIENT (

Pcode INT NOT NULL,

Entry\_date DATE NOT NULL, Discharge\_date DATE NULL,

wardno INT NOT NULL CHECK (wardno < 6), disease VARCHAR(100) NOT NULL,

FOREIGN KEY (Pcode) REFERENCES PATIENTMASTER(Pcode)

);

## ✅2. Insert Records

-- DOCTOR Records

INSERT INTO DOCTOR VALUES

|  |  |  |
| --- | --- | --- |
| (1, | 'Dr. | Sharma', 'Delhi', 'M.B.B.S.'), |
| (2, | 'Dr. | Kapoor', 'Mumbai', 'M.S.'), |
| (3, | 'Dr. | Mehta', 'Chennai', 'B.A.M.S'), |
| (4, | 'Dr. | Reddy', 'Hyderabad', 'M.S.'), |
| (5, | 'Dr. | Patel', 'Ahmedabad', 'M.B.B.S.'), |
| (6, | 'Dr. | Khan', 'Lucknow', 'M.B.B.S.'), |
| (7, | 'Dr. | Iyer', 'Pune', 'M.S.'), |
| (8, | 'Dr. | Das', 'Kolkata', 'B.A.M.S'), |
| (9, | 'Dr. | Sinha', 'Bhopal', 'M.B.B.S.'), |

(10, 'Dr. Rao', 'Vizag', 'B.A.M.S');

-- PATIENTMASTER Records

INSERT INTO PATIENTMASTER VALUES

(101, 'Amit', 'Delhi', 34, 'M', 'O+', 1),

(102, 'Reena', 'Mumbai', 29, 'F', 'A+', 2),

(103, 'Jalf\_zaon', 'Hyderabad', 45, 'M', 'B+', 3),

(104, 'Ravi', 'Chennai', 32, 'M', 'AB+', 4),

(105, 'Neha', 'Pune', 28, 'F', 'O-', 5),

(106, 'Sara', 'Bangalore', 37, 'F', 'B-', 6),

(107, 'Zoya', 'Delhi', 31, 'F', 'A-', 7),

(108, 'Imran', 'Kolkata', 40, 'M', 'AB-', 8),

(109, 'Rahul', 'Mumbai', 36, 'M', 'O+', 9),

(110, 'Priya', 'Ahmedabad', 30, 'F', 'B+', 10);

-- ADMITTEDPATIENT Records

INSERT INTO ADMITTEDPATIENT VALUES

|  |  |  |
| --- | --- | --- |
| (101, | '2008-03-02', | '2008-03-10', 1, 'Flu'), |
| (102, | '2008-03-05', | '2008-03-12', 2, 'Typhoid'), |
| (103, | '2008-03-10', | NULL, 3, 'Jalf\_zaon'), |
| (104, | '2008-02-20', | '2008-03-01', 1, 'Cold'), |
| (105, | '2008-03-18', | '2008-03-25', 2, 'Malaria'), |
| (106, | '2008-03-22', | NULL, 3, 'Covid-19'), |
| (107, | '2008-04-01', | NULL, 1, 'Asthma'), |
| (108, | '2008-03-15', | '2008-03-28', 4, 'Jalf\_zaon'), |
| (109, | '2008-03-07', | NULL, 5, 'Jalf\_zaon'), |
| (110, | '2008-03-09', | '2008-03-17', 2, 'Pneumonia'); |

**🧾 3. Queries**

# Find patients admitted between 03/03/08 and 25/03/08

SELECT \*

FROM ADMITTEDPATIENT

WHERE Entry\_date BETWEEN '2008-03-03' AND '2008-03-25';

# Find doctors treating Jalf\_zaon patients

SELECT DISTINCT D.Dname FROM DOCTOR D

JOIN PATIENTMASTER PM ON D.Did = PM.Did

JOIN ADMITTEDPATIENT AP ON PM.Pcode = AP.Pcode

WHERE AP.disease = 'Jalf\_zaon';

# Procedure to calculate bill (for currently admitted patients)

CREATE PROCEDURE CalculateBills AS

BEGIN

SELECT

AP.Pcode, PM.Pname,

DATEDIFF(DAY, AP.Entry\_date, ISNULL(AP.Discharge\_date, GETDATE())) AS No\_of\_days, DATEDIFF(DAY, AP.Entry\_date, ISNULL(AP.Discharge\_date, GETDATE())) \* 500 AS Bill

FROM ADMITTEDPATIENT AP

JOIN PATIENTMASTER PM ON AP.Pcode = PM.Pcode

WHERE AP.Discharge\_date IS NULL; END;

GO

-- Execute procedure EXEC CalculateBills;

# Trigger on DOCTOR to allow only specific qualifications

Note: Already applied CHECK constraint, but here’s the **trigger version**:

CREATE TRIGGER trg\_Doctor\_Qualification ON DOCTOR

AFTER INSERT, UPDATE AS

BEGIN

IF EXISTS (

SELECT \* FROM inserted

WHERE qualification NOT IN ('M.B.B.S.', 'B.A.M.S', 'M.S.')

) BEGIN

RAISERROR ('Invalid qualification. Must be M.B.B.S., B.A.M.S, or M.S.', 16, 1); ROLLBACK TRANSACTION;

END END;

26)

Table and record already in 25 slip.

# Find details of patients treated by M.B.B.S. doctors

SELECT PM.\*

FROM PATIENTMASTER PM

JOIN DOCTOR D ON PM.Did = D.Did

WHERE D.qualification = 'M.B.B.S.';

# Find name of the doctor treating male patients suffering from brain tumor & having age < 40

SELECT DISTINCT D.Dname FROM DOCTOR D

JOIN PATIENTMASTER PM ON D.Did = PM.Did

JOIN ADMITTEDPATIENT AP ON PM.Pcode = AP.Pcode

WHERE PM.gender = 'M' AND PM.age < 40

AND AP.disease = 'brain tumor';

◻ *Make sure you’ve inserted a patient record matching those conditions (age < 40, gender M, disease = 'brain tumor') for this to return results.*

# Procedure: Calculate bill for patients discharged on 2008-03-30

CREATE PROCEDURE CalculateDischargedBill AS

BEGIN

SELECT

AP.Pcode, PM.Pname,

DATEDIFF(DAY, AP.Entry\_date, AP.Discharge\_date) AS No\_of\_days,

DATEDIFF(DAY, AP.Entry\_date, AP.Discharge\_date) \* 500 AS Bill FROM ADMITTEDPATIENT AP

JOIN PATIENTMASTER PM ON AP.Pcode = PM.Pcode

WHERE AP.Discharge\_date = '2008-03-30'; END;

GO

-- Execute procedure

EXEC CalculateDischargedBill;

# Cursor on DOCTOR table to fetch the first row & count total rows

DECLARE @Dname VARCHAR(100), @RowCount INT;

-- Declare cursor

DECLARE doctor\_cursor CURSOR FOR SELECT Dname FROM DOCTOR;

-- Count rows

SELECT @RowCount = COUNT(\*) FROM DOCTOR;

-- Open cursor

OPEN doctor\_cursor;

-- Fetch first row

FETCH NEXT FROM doctor\_cursor INTO @Dname;

-- Print result

PRINT 'First Doctor Name: ' + @Dname;

PRINT 'Total Number of Doctors: ' + CAST(@RowCount AS VARCHAR);

-- Clean up

CLOSE doctor\_cursor; DEALLOCATE doctor\_cursor;