page-1--

a) Find all the projects which are provided 3 or more parts .

b) Write a trigger on PROJECT table for update / insert such that the .jname value Should not be repeated.

c) Find full details of all projects in London.

d) Write a procedure for calculating the total sales of all the parts which are provided to projects in paris city.

```
CREATE DATABASE SupplierPartsDB;
USE SupplierPartsDB;
-- SUPPLIER Table
CREATE TABLE SUPPLIER (
Sno VARCHAR(5) PRIMARY KEY CHECK (Sno LIKE 'S[0-9][0-9][0-9][0-9]'),
Sname VARCHAR(50) NOT NULL,
address VARCHAR(100) NOT NULL,
City VARCHAR(50) NOT NULL CHECK (City IN ('London', 'Paris', 'Rome', 'New York', 'Amsterdam'))
);

-- PARTS Table
CREATE TABLE PARTS (
Pno INT PRIMARY KEY,
Pname VARCHAR(50) NOT NULL,
Color VARCHAR(20) NOT NULL,
Weight FLOAT NOT NULL,
Price FLOAT NOT NULL
);

-- PROJECT Table
CREATE TABLE PROJECT (
Jno INT PRIMARY KEY,
Jname VARCHAR(50) NOT NULL UNIQUE,
City VARCHAR(50) NOT NULL CHECK (City IN ('London', 'Paris', 'Rome', 'New York', 'Amsterdam'))
);

-- SPJ Table (Supply table)
CREATE TABLE SPJ (
Sno VARCHAR(5) FOREIGN KEY REFERENCES SUPPLIER(Sno),
Pno INT FOREIGN KEY REFERENCES PARTS(Pno),
Jno INT FOREIGN KEY REFERENCES PROJECT(Jno),
Qty INT NOT NULL,
PRIMARY KEY (Sno, Pno, Jno)
);

-- Suppliers
INSERT INTO SUPPLIER VALUES
('S0001', 'Alpha Co.', '123 Main St', 'London'),
('S0002', 'Beta Ltd.', '456 Park Ave', 'Paris'),
('S0003', 'Gamma Inc.', '789 Elm St', 'Rome'),
('S0004', 'Delta Corp.', '10 Downing St', 'New York'),
('S0005', 'Epsilon LLC', '22 Central Rd', 'Amsterdam'),
('S0006', 'Zeta Co.', '31 Oxford St', 'London'),
('S0007', 'Eta Corp.', '45 Maple St', 'Paris'),
```

('S0008', 'Theta Inc.', '67 Apple Rd', 'Rome'),
('S0009', 'Iota LLC', '98 Cherry Ln', 'New York'),
('S0010', 'Kappa Ltd.', '12 River St', 'Amsterdam');

-- Parts
INSERT INTO PARTS VALUES
(1, 'Bolt', 'Red', 0.5, 5),
(2, 'Nut', 'Blue', 0.3, 3),
(3, 'Screw', 'Black', 0.2, 2),
(4, 'Washer', 'White', 0.1, 1),
(5, 'Clamp', 'Silver', 1.0, 10),
(6, 'Spring', 'Yellow', 0.8, 8),
(7, 'Pin', 'Gold', 0.4, 4),
(8, 'Clip', 'Green', 0.6, 6),
(9, 'Rod', 'Brown', 1.2, 12),
(10, 'Nut-Bolt', 'Red', 1.1, 11);

-- Projects
INSERT INTO PROJECT VALUES
(101, 'Bridge Build', 'London'),
(102, 'Road Fix', 'Paris'),
(103, 'Tower Construction', 'Rome'),
(104, 'Mall Setup', 'New York'),
(105, 'Harbor Renovation', 'Amsterdam'),
(106, 'Airport Extension', 'London'),
(107, 'Metro Rail', 'Paris'),
(108, 'Skyscraper', 'Rome'),
(109, 'Tunnel Dig', 'New York'),
(110, 'Park Revamp', 'Amsterdam');

-- SPJ
INSERT INTO SPJ VALUES
('S0001', 1, 101, 10),
('S0002', 2, 102, 5),
('S0003', 3, 103, 15),
('S0004', 4, 104, 20),
('S0005', 5, 105, 8),
('S0001', 6, 101, 12),
('S0001', 2, 101, 6),
('S0002', 2, 107, 10),
('S0006', 3, 107, 5),
('S0007', 1, 107, 10);

--a) Find all the projects which are provided 3 or more parts .
--b) Write a trigger on PROJECT table for update / insert such that the .jname value Should not
--be repeated.
--c) Find full details of all projects in London.
--d) Write a procedure for calculating the total sales of all the parts which are provided to
--projects in paris city.

```
--a)
SELECT Jno
FROM SPJ
GROUP BY Jno
HAVING COUNT(DISTINCT Pno) >= 3;

--b)
CREATE TRIGGER trg_unique_jname
ON PROJECT
INSTEAD OF INSERT, UPDATE
AS
BEGIN
IF EXISTS (
SELECT Jname
FROM INSERTED
GROUP BY Jname
HAVING COUNT(*) > 1
UNION
SELECT I.Jname
FROM INSERTED I
JOIN PROJECT P ON I.Jname = P.Jname
)
BEGIN
RAISERROR('Duplicate project name not allowed.', 16, 1);
ROLLBACK TRANSACTION;
END
ELSE
BEGIN
-- Proceed with actual insert or update
IF EXISTS (SELECT * FROM INSERTED)
BEGIN
INSERT INTO PROJECT (Jno, Jname, City)
SELECT Jno, Jname, City FROM INSERTED;
END
END
END;
--insert unique project(executed)
INSERT INTO PROJECT (Jno, Jname, City)
VALUES ('J101', 'CleanWater', 'London');
--insert duplicate project(show error)
INSERT INTO PROJECT (Jno, Jname, City)
VALUES ('J102', 'CleanWater', 'Paris');
--show data from project
SELECT * FROM PROJECT;

--c)
SELECT * FROM PROJECT WHERE City = 'London';
```

```
--d)
CREATE PROCEDURE TotalSalesParis
AS
BEGIN
SELECT SUM(S.Qty * P.Price) AS TotalSales
FROM SPJ S
JOIN PARTS P ON S.Pno = P.Pno
JOIN PROJECT J ON S.Jno = J.Jno
WHERE J.City = 'Paris';
END;

-- To execute
EXEC TotalSalesParis;
```

Page 2—

```
--a) Find PC models having a speed of at least 150 MHz.
--b) Find those manufacturers that sell Laptops, but not PC's.
--c) Write a trigger on LAPTOP table such that the price should not less than 30000
--d) Write a procedure to find the manufacturer who has produced the most expensive laptop.

CREATE DATABASE ProductDB;
GO

USE ProductDB;
GO

CREATE TABLE PRODUCT (
Maker VARCHAR(50) NOT NULL,
ModelNo INT PRIMARY KEY NOT NULL,
Type VARCHAR(20) NOT NULL CHECK (Type IN ('PC', 'Laptop', 'Printer'))
);

CREATE TABLE PC (
ModelNo INT PRIMARY KEY NOT NULL FOREIGN KEY REFERENCES PRODUCT(ModelNo),
Speed INT NOT NULL,
RAM INT NOT NULL,
HD INT NOT NULL,
CD VARCHAR(20) NOT NULL,
Price INT NOT NULL
);

CREATE TABLE LAPTOP (
ModelNo INT PRIMARY KEY NOT NULL FOREIGN KEY REFERENCES PRODUCT(ModelNo),
Speed INT NOT NULL,
RAM INT NOT NULL,
HD INT NOT NULL,
Price INT NOT NULL
);

CREATE TABLE PRINTER (
```

```
ModelNo INT PRIMARY KEY NOT NULL FOREIGN KEY REFERENCES PRODUCT(ModelNo),
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
);

-- PRODUCT
INSERT INTO PRODUCT VALUES
('IBM', 101, 'PC'),
('Dell', 102, 'Laptop'),
('HP', 103, 'Printer'),
('Lenovo', 104, 'PC'),
('HP', 105, 'Laptop'),
('Dell', 106, 'Printer'),
('Apple', 107, 'Laptop'),
('Acer', 108, 'PC'),
('Canon', 109, 'Printer'),
('Asus', 110, 'Laptop');

-- PC
INSERT INTO PC VALUES
(101, 200, 8, 500, '52X', 40000),
(104, 180, 4, 250, '48X', 35000),
(108, 160, 8, 320, '40X', 37000);

-- LAPTOP
INSERT INTO LAPTOP VALUES
(102, 180, 8, 500, 45000),
(105, 200, 16, 512, 50000),
(107, 220, 16, 1000, 95000),
(110, 150, 8, 256, 32000);

-- PRINTER
INSERT INTO PRINTER VALUES
(103, 'T', 'ink-jet', 12000),
(106, 'F', 'laser', 15000),
(109, 'F', 'dot-matrix', 8000);

--a)
SELECT * FROM PC WHERE Speed >= 150;
--b)
SELECT DISTINCT P1.Maker
FROM PRODUCT P1
WHERE P1.Type = 'Laptop'
AND NOT EXISTS (
SELECT 1
FROM PRODUCT P2
WHERE P2.Type = 'PC' AND P2.Maker = P1.Maker
);
```

```
--c)
CREATE TRIGGER trg_LaptopPriceCheck
ON LAPTOP
AFTER INSERT, UPDATE
AS
BEGIN
IF EXISTS (
SELECT 1 FROM INSERTED WHERE Price < 30000
)
BEGIN
RAISERROR('Laptop price cannot be less than 30000.', 16, 1);
ROLLBACK TRANSACTION;
END
END;

--below 30000 (gives error)
INSERT INTO LAPTOP (Modelno, Speed, RAM, HD, Price)
VALUES (2001, 2.5, 8, 256, 25000);
--valid price above 30000(succeed)
-- Insert into PRODUCT first
INSERT INTO PRODUCT (Maker, Modelno, Type)
VALUES ('Dell', 2002, 'Laptop');

INSERT INTO LAPTOP (Modelno, Speed, RAM, HD, Price)
VALUES (2002, 3.0, 16, 512, 60000);

--d)
CREATE PROCEDURE MostExpensiveLaptopMaker
AS
BEGIN
SELECT TOP 1 PR.Maker, L.ModelNo, L.Price
FROM LAPTOP L
JOIN PRODUCT PR ON PR.ModelNo = L.ModelNo
ORDER BY L.Price DESC;
END;

-- Execute the procedure
EXEC MostExpensiveLaptopMaker;
```

Page 3—

```
--a) Find the different types of printers produced by Epson.
--b) Find those hard disk sizes which occur in two or more PC's.
--c) Write a trigger on LAPTOP table such that the minimum speed should be 1201\4Hz.
--d) Demonstrate the use of cursor using PRODUCT table.

CREATE DATABASE ProductDB2;

USE ProductDB2;

-- Create PRODUCT Table
```

```
CREATE TABLE PRODUCT (
Maker VARCHAR(50) NOT NULL,
Modelno INT PRIMARY KEY,
Type VARCHAR(50) NOT NULL CHECK (Type IN ('PC', 'Laptop', 'Printer'))
);

-- Create PC Table
CREATE TABLE PC (
Modelno INT PRIMARY KEY,
Speed FLOAT NOT NULL,       -- Speed in MHz
RAM INT NOT NULL,           -- RAM in MB
HD INT NOT NULL,            -- HD size in GB
CD FLOAT NOT NULL,          -- Speed of CD reader
Price DECIMAL(10, 2) NOT NULL,  -- Price of the PC
FOREIGN KEY (Modelno) REFERENCES PRODUCT(Modelno)
);

-- Create LAPTOP Table
CREATE TABLE LAPTOP (
Modelno INT PRIMARY KEY,
Speed FLOAT NOT NULL,       -- Speed in MHz
RAM INT NOT NULL,           -- RAM in MB
HD INT NOT NULL,            -- HD size in GB
Price DECIMAL(10, 2) NOT NULL,  -- Price of the laptop
FOREIGN KEY (Modelno) REFERENCES PRODUCT(Modelno)
);

-- Create PRINTER Table
CREATE TABLE PRINTER (
Modelno INT PRIMARY KEY,
Color CHAR(1) NOT NULL CHECK (Color IN ('T', 'F')),  -- T for color, F for black and white
Type VARCHAR(50) NOT NULL CHECK (Type IN ('Laser', 'Ink-jet', 'Dot-matrix', 'Dry')),
Price DECIMAL(10, 2) NOT NULL,  -- Price of the printer
FOREIGN KEY (Modelno) REFERENCES PRODUCT(Modelno)
);

-- Insert records into PRODUCT table
INSERT INTO PRODUCT (Maker, Modelno, Type) VALUES
('IBM', 1001, 'PC'),
('Compaq', 1002, 'Laptop'),
('Dell', 1003, 'PC'),
('Epson', 1004, 'Printer'),
('HP', 1005, 'Laptop'),
('Acer', 1006, 'PC'),
('Canon', 1007, 'Printer'),
('Lenovo', 1008, 'Laptop'),
('Acer', 1009, 'PC'),
('Epson', 1010, 'Printer');
```

```
-- Insert records into PC table
INSERT INTO PC (Modelno, Speed, RAM, HD, CD, Price) VALUES
(1001, 2400, 8, 500, 48, 50000),
(1003, 3000, 16, 1000, 52, 70000),
(1006, 2200, 4, 250, 40, 45000),
(1009, 2800, 8, 750, 48, 65000);

-- Insert records into LAPTOP table
INSERT INTO LAPTOP (Modelno, Speed, RAM, HD, Price) VALUES
(1002, 2500, 8, 500, 60000),
(1005, 3200, 16, 1000, 85000),
(1008, 2800, 8, 750, 70000);

-- Insert records into PRINTER table
INSERT INTO PRINTER (Modelno, Color, Type, Price) VALUES
(1004, 'T', 'Laser', 20000),
(1010, 'F', 'Ink-jet', 15000),
(1007, 'F', 'Dot-matrix', 5000);



--a)
SELECT DISTINCT PR.Type
FROM PRINTER PR
JOIN PRODUCT P ON PR.Modelno = P.Modelno
WHERE P.Maker = 'Epson';

--b)
SELECT HD
FROM PC
GROUP BY HD
HAVING COUNT(HD) >= 2;

--c)
CREATE TRIGGER trg_LaptopSpeedCheck
ON LAPTOP
AFTER INSERT, UPDATE
AS
BEGIN
IF EXISTS (
SELECT 1 FROM INSERTED WHERE Speed < 1200
)
BEGIN
RAISERROR('Laptop speed cannot be less than 1200 MHz.', 16, 1);
ROLLBACK TRANSACTION;
END
END;
--This will trigger an error (speed < 1200)
INSERT INTO LAPTOP (Modelno, Speed, RAM, HD, Price)
VALUES (4001, 1100, 8, 512, 45000);
```

```
--This will succeed (speed >= 1200)
INSERT INTO LAPTOP (Modelno, Speed, RAM, HD, Price)
VALUES (4002, 1300, 16, 1024, 60000);

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

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;
```

Page 4

```
--a) Find the details of doctors who are treating the patient of ward no 3.
--b) Write a trigger on PATIENTMASTER table such that the blood group should be
--A,B,AB or 0.
--c) Find the details of patient who are discharge within the period 03/03/12 to 25/ 03/12
--d) Write a procedure on ADMIFTEDPATIENT table such as to calculate bill of all
--discharged patients. The charges per day for a ward is WardNo. * 100. e.g. For ward no 3
--charges/day are 300Rs.
----------------------------------------------------------------------------------------------------------
create database ass1
use ass1
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) NOT NULL CHECK (gender IN ('M', 'F')),
bloodgroup VARCHAR(3) NOT NULL,
Pid INT NOT NULL REFERENCES DOCTOR(Did)
);
CREATE TABLE ADMITTEDPATIENT (
P_code INT REFERENCES PATIENTMASTER(Pcode),
Entry_date DATE NOT NULL,
Discharge_date DATE NOT NULL,
wardno INT NOT NULL CHECK (wardno < 6),
disease VARCHAR(100) NOT NULL,
PRIMARY KEY(P_code, Entry_date)
);
-- Doctors
INSERT INTO DOCTOR VALUES
(1, 'Dr. Mehra', 'Delhi', 'MBBS'),
(2, 'Dr. Singh', 'Mumbai', 'MD'),
(3, 'Dr. Khan', 'Kolkata', 'MBBS'),
(4, 'Dr. Rao', 'Chennai', 'MD'),
(5, 'Dr. Patel', 'Ahmedabad', 'MBBS'),
(6, 'Dr. Das', 'Bangalore', 'MD'),
(7, 'Dr. Gupta', 'Jaipur', 'MBBS'),
(8, 'Dr. Sharma', 'Pune', 'MD'),
(9, 'Dr. Roy', 'Lucknow', 'MBBS'),
(10, 'Dr. Iyer', 'Hyderabad', 'MD');

-- Patients
INSERT INTO PATIENTMASTER VALUES
(101, 'Anita', 'Delhi', 25, 'F', 'A', 1),
(102, 'Ravi', 'Mumbai', 35, 'M', 'B', 2),
(103, 'Sita', 'Kolkata', 30, 'F', 'O', 3),
(104, 'Arjun', 'Chennai', 28, 'M', 'AB', 4),
(105, 'Reena', 'Ahmedabad', 40, 'F', 'A', 5),
(106, 'Ajay', 'Bangalore', 45, 'M', 'B', 6),
(107, 'Pooja', 'Jaipur', 22, 'F', 'O', 7),
(108, 'Karan', 'Pune', 33, 'M', 'AB', 8),
(109, 'Meena', 'Lucknow', 29, 'F', 'A', 9),
(110, 'Vijay', 'Hyderabad', 37, 'M', 'B', 10);

-- Admitted Patients
INSERT INTO ADMITTEDPATIENT VALUES
(101, '2012-03-01', '2012-03-05', 3, 'Fever'),
(102, '2012-03-10', '2012-03-15', 3, 'Cold'),
(103, '2012-03-20', '2012-03-25', 2, 'Malaria'),
(104, '2012-02-01', '2012-02-10', 1, 'Injury'),
(105, '2012-03-05', '2012-03-12', 3, 'Dengue'),
(106, '2012-04-01', '2012-04-05', 4, 'Flu'),
(107, '2012-03-08', '2012-03-22', 5, 'Covid'),
(108, '2012-03-03', '2012-03-24', 2, 'Typhoid'),
```

```
(109, '2012-03-12', '2012-03-19', 1, 'Allergy'),
(110, '2012-03-21', '2012-03-30', 3, 'Asthma');


--a)
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;

--b)
CREATE TRIGGER check_bloodgroup
ON PATIENTMASTER
AFTER INSERT, UPDATE
AS
BEGIN
IF EXISTS (
SELECT 1
FROM INSERTED
WHERE bloodgroup NOT IN ('A', 'B', 'AB', 'O')
)
BEGIN
RAISERROR ('Invalid blood group. Allowed values are A, B, AB, or O.', 16, 1);
ROLLBACK TRANSACTION;
END
END;


--show output of procedure if query is right otherwise it gives error
INSERT INTO PATIENTMASTER (Pcode, Pname, Padd, age, gender, bloodgroup, Pid)
VALUES (111, 'amol', 'shirpur', 20, 'M', 'abs', 1);

--c)
SELECT *
FROM PATIENTMASTER P
JOIN ADMITTEDPATIENT A ON P.Pcode = A.P_code
WHERE A.Discharge_date BETWEEN '2012-03-03' AND '2012-03-25';

--d)

CREATE PROCEDURE calculate_bill
AS
BEGIN
DECLARE @P_code INT,
@Entry_date DATE,
@Discharge_date DATE,
@wardno INT,
@days_stayed INT,
```

```
@bill_amount INT;

DECLARE patient_cursor CURSOR FOR
SELECT P_code, Entry_date, Discharge_date, wardno
FROM ADMITTEDPATIENT;

OPEN patient_cursor;
FETCH NEXT FROM patient_cursor INTO @P_code, @Entry_date, @Discharge_date, @wardno;

WHILE @@FETCH_STATUS = 0
BEGIN
SET @days_stayed = DATEDIFF(DAY, @Entry_date, @Discharge_date);
SET @bill_amount = @days_stayed * @wardno * 100;

PRINT 'Patient Code: ' + CAST(@P_code AS VARCHAR) +
', Bill: Rs. ' + CAST(@bill_amount AS VARCHAR);

FETCH NEXT FROM patient_cursor INTO @P_code, @Entry_date, @Discharge_date, @wardno;
END

CLOSE patient_cursor;
DEALLOCATE patient_cursor;
END;
--execute
EXEC calculate_bill;
```

Page 5

a) Find the details of the doctors who are treating the patients of ward no 3 & display the result along with patient name & disease. b) Find the name of the disease by which maximum patients are suffering. c) Write a trigger on ADMITTEDPATIENT table such that the wardno value should be between 1-5. d) Write a procedure to give the details of patients who are admitted in the hospital for more than 5 days.

a)

```
--a)
SELECT DISTINCT
D.Did, D.Dname, D.Daddress, D.qualification,
P.Pname AS PatientName,
A.disease
FROM DOCTOR D
JOIN PATIENTMASTER P ON D.Did = P.Pid
JOIN ADMITTEDPATIENT A ON P.Pcode = A.P_code
WHERE A.wardno = 3;

--b)
SELECT TOP 1
disease, COUNT(*) AS total_patients
FROM ADMITTEDPATIENT
GROUP BY disease
ORDER BY total_patients DESC;
```

```
--c)
CREATE TRIGGER trg_check_wardno
ON ADMITTEDPATIENT
AFTER INSERT, UPDATE
AS
BEGIN
IF EXISTS (
SELECT 1 FROM INSERTED WHERE wardno < 1 OR wardno > 5
)
BEGIN
RAISERROR('Invalid ward number. Must be between 1 and 5.', 16, 1);
ROLLBACK TRANSACTION;
END
END;

-- This query will succeed because wardno is between 1 and 5
INSERT INTO ADMITTEDPATIENT (P_code, Entry_date, Discharge_date, wardno, disease)
VALUES (111, '2025-04-21', '2025-04-25', 3, 'Fever');

-- This query will fail because wardno is outside the valid range (greater than 5)
INSERT INTO ADMITTEDPATIENT (P_code, Entry_date, Discharge_date, wardno, disease)
VALUES (112, '2025-04-21', '2025-04-25', 6, 'Cold');

--d)
CREATE PROCEDURE get_long_stay_patients
AS
BEGIN
SELECT
P.Pcode, P.Pname, P.age, P.gender, A.Entry_date, A.Discharge_date,
DATEDIFF(DAY, A.Entry_date, A.Discharge_date) AS Days_Admitted
FROM PATIENTMASTER P
JOIN ADMITTEDPATIENT A ON P.Pcode = A.P_code
WHERE DATEDIFF(DAY, A.Entry_date, A.Discharge_date) > 5;
END;

EXEC get_long_stay_patients;
```

Page 6--

a) Find details of the patients who are treated by M.B.B.S. doctors.

b) Find the details of patient who is suffered from blood cancer having age less than 50 years & blood group is A.

c) write a procedure on ADMITTEDPATIENT table such as to calculate the bill of all patients. (bill no of days * 600)

d) Write a cursor on PATIENTMASTER table to fetch the last record & display the rows in that table.

```
--a)
SELECT P.Pcode, P.Pname, P.Padd, P.age, P.gender, P.bloodgroup, P.Pid
FROM PATIENTMASTER P
JOIN DOCTOR D ON P.Pid = D.Did
WHERE D.qualification = 'MBBS';

--b)
SELECT P.Pcode, P.Pname, P.Padd, P.age, P.gender, P.bloodgroup, A.disease
```

```
FROM PATIENTMASTER P
JOIN ADMITTEDPATIENT A ON P.Pcode = A.P_code
WHERE A.disease LIKE '%blood cancer%'
AND P.age < 50
AND P.bloodgroup = 'A';

--c)
CREATE PROCEDURE Calculate_Bills
AS
BEGIN
DECLARE @P_code INT, @Entry_date DATE, @Discharge_date DATE, @wardno INT, @days_stayed INT, @bill_amount INT;

DECLARE patient_cursor CURSOR FOR
SELECT P_code, Entry_date, Discharge_date, wardno
FROM ADMITTEDPATIENT;

OPEN patient_cursor;
FETCH NEXT FROM patient_cursor INTO @P_code, @Entry_date, @Discharge_date, @wardno;

WHILE @@FETCH_STATUS = 0
BEGIN
-- Calculate number of days stayed
SET @days_stayed = DATEDIFF(DAY, @Entry_date, @Discharge_date);
-- Calculate the bill (days * 600)
SET @bill_amount = @days_stayed * 600;

PRINT 'Patient Code: ' + CAST(@P_code AS VARCHAR) +
', Bill: Rs. ' + CAST(@bill_amount AS VARCHAR);

FETCH NEXT FROM patient_cursor INTO @P_code, @Entry_date, @Discharge_date, @wardno;
END;

CLOSE patient_cursor;
DEALLOCATE patient_cursor;
END;
--run
EXEC Calculate_Bills;

--d)
DECLARE @Pcode INT, @Pname VARCHAR(50), @Padd VARCHAR(100), @age INT, @gender CHAR(1), @bloodgroup
VARCHAR(3), @Pid INT;

DECLARE patient_cursor CURSOR FOR
SELECT TOP 1 Pcode, Pname, Padd, age, gender, bloodgroup, Pid
FROM PATIENTMASTER
ORDER BY Pcode DESC;

OPEN patient_cursor;
FETCH NEXT FROM patient_cursor INTO @Pcode, @Pname, @Padd, @age, @gender, @bloodgroup, @Pid;
```

```
-- Display the last row data
PRINT 'Pcode: ' + CAST(@Pcode AS VARCHAR) + ', Pname: ' + @Pname + ', Address: ' + @Padd +
', Age: ' + CAST(@age AS VARCHAR) + ', Gender: ' + @gender + ', Bloodgroup: ' + @bloodgroup +
', Doctor ID: ' + CAST(@Pid AS VARCHAR);

CLOSE patient_cursor;
DEALLOCATE patient_cursor;
```

Page 7--

```
--a) Find details of the patients who are treated by M.S. doctors.
--b) Find the name of doctor who is treating maximum number of patients.
--c) Write a procedure to give the details of patients who are admitted in the hospital for more
--than 15 days.
--d) Create a view on DOCTOR & PATIENTMASTER tables. Update details of the patients
--who are treated by B.A.-M.S. doctors to M.B.B.S doctor.

--a)
SELECT P.Pcode, P.Pname, P.Padd, P.age, P.gender, P.bloodgroup, D.Dname AS Doctor_Name, A.disease
FROM PATIENTMASTER P
JOIN DOCTOR D ON P.Pid = D.Did
JOIN ADMITTEDPATIENT A ON P.Pcode = A.P_code
WHERE D.qualification = 'MD';

--b)
SELECT TOP 1 D.Dname
FROM DOCTOR D
JOIN PATIENTMASTER P ON D.Did = P.Pid
GROUP BY D.Dname
ORDER BY COUNT(P.Pcode) DESC;

--c)
CREATE PROCEDURE GetLongTermPatients
AS
BEGIN
SELECT P.Pcode, P.Pname, P.Padd, P.age, P.gender, P.bloodgroup, A.Entry_date, A.Discharge_date, A.disease
FROM PATIENTMASTER P
JOIN ADMITTEDPATIENT A ON P.Pcode = A.P_code
WHERE DATEDIFF(DAY, A.Entry_date, A.Discharge_date) > 15;
END;
--run procedure
EXEC GetLongTermPatients;

--d) create view
CREATE VIEW DoctorPatientDetails AS
SELECT D.Did, D.Dname, D.qualification, P.Pcode, P.Pname, P.Padd, P.age, P.gender, P.bloodgroup
FROM DOCTOR D
JOIN PATIENTMASTER P ON D.Did = P.Pid;

select * from DoctorPatientDetails
```

```
--update
UPDATE DOCTOR
SET qualification = 'MS'
WHERE qualification = 'MD';

select * from DOCTOR
```

---

page 8--

--a) Find the details of customers whose minimum balance is 1 lakhs.
--b) Find the details of amount credited within the period 25-3-2012 to 28-3- 2012
--c) Write a trigger on TRANSACTION table to calculate current balance of account on which transaction is made.
--d) Write a cursor on ACCOUNT table of balance attribute such that if the balance is less than 10000 then print the 'loan is not provided'else 'loan is provided'.

```
CREATE DATABASE BankDB;
GO

USE BankDB;
GO

-- ACCOUNT Table
CREATE TABLE ACCOUNT (
accno INT PRIMARY KEY CHECK (accno < 1000),
opendate DATE NOT NULL,
acctype CHAR(1) NOT NULL CHECK (acctype IN ('P', 'J')),
balance MONEY NOT NULL
);

-- CUSTOMER Table
CREATE TABLE CUSTOMER (
custid INT PRIMARY KEY,
name VARCHAR(50) NOT NULL,
address VARCHAR(100) NOT NULL,
accno INT NOT NULL FOREIGN KEY REFERENCES ACCOUNT(accno)
);

-- TRANSACTION Table
CREATE TABLE TRANSACTION1 (
transid INT PRIMARY KEY,
transdate DATE NOT NULL,
accno INT NOT NULL FOREIGN KEY REFERENCES ACCOUNT(accno),
transtype CHAR(1) NOT NULL CHECK (transtype IN ('C', 'D')),
amount MONEY NOT NULL
);

-- ACCOUNT Records
INSERT INTO ACCOUNT VALUES
```

```
(101, '2012-01-10', 'P', 120000),
(102, '2012-02-15', 'J', 95000),
(103, '2012-01-20', 'P', 50000),
(104, '2012-03-10', 'P', 130000),
(105, '2012-03-20', 'J', 85000),
(106, '2012-03-25', 'P', 200000),
(107, '2012-03-30', 'P', 75000),
(108, '2012-04-05', 'J', 30000),
(109, '2012-04-10', 'P', 40000),
(110, '2012-04-15', 'P', 100000);

-- CUSTOMER Records
INSERT INTO CUSTOMER VALUES
(1, 'Amit', 'Delhi', 101),
(2, 'Neha', 'Mumbai', 102),
(3, 'Ravi', 'Kolkata', 103),
(4, 'Pooja', 'Chennai', 104),
(5, 'Karan', 'Ahmedabad', 105),
(6, 'Meena', 'Bangalore', 106),
(7, 'Vikram', 'Jaipur', 107),
(8, 'Riya', 'Pune', 108),
(9, 'Sunil', 'Lucknow', 109),
(10, 'Priya', 'Hyderabad', 110);

-- TRANSACTION Records
INSERT INTO TRANSACTION1 VALUES
(1, '2012-03-25', 101, 'C', 5000),
(2, '2012-03-26', 102, 'C', 3000),
(3, '2012-03-27', 103, 'D', 2000),
(4, '2012-03-28', 104, 'C', 7000),
(5, '2012-03-29', 105, 'D', 1000),
(6, '2012-03-30', 106, 'C', 8000),
(7, '2012-03-30', 107, 'D', 2000),
(8, '2012-03-31', 108, 'C', 6000),
(9, '2012-04-01', 109, 'C', 4000),
(10, '2012-04-02', 110, 'D', 3000);

select * from ACCOUNT
select * from CUSTOMER
select * from TRANSACTION1

--a)
SELECT C.*
FROM CUSTOMER C
JOIN ACCOUNT A ON C.accno = A.accno
WHERE A.balance >= 100000;
--b)
SELECT *
FROM TRANSACTION1
```

```
WHERE transtype = 'C' AND transdate BETWEEN '2012-03-25' AND '2012-03-28';
--c)
CREATE TRIGGER trg_UpdateBalance
ON TRANSACTION1
AFTER INSERT
AS
BEGIN
DECLARE @accno INT, @amount MONEY, @type CHAR(1)

SELECT @accno = accno, @amount = amount, @type = transtype 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;
-- Before inserting, check balance
SELECT accno, balance FROM ACCOUNT WHERE accno = 101;

-- Insert a credit transaction (e.g., Rs. 2000)
INSERT INTO TRANSACTION1 (transid, transdate, accno, transtype, amount)
VALUES (11, GETDATE(), 101, 'C', 2000);

-- Now check the balance again to see if it's updated
SELECT accno, balance FROM ACCOUNT WHERE accno = 101;

--d)
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
IF @balance < 10000
PRINT 'AccNo ' + CAST(@accno AS VARCHAR) + ': Loan is not provided'
ELSE
PRINT 'AccNo ' + CAST(@accno AS VARCHAR) + ': Loan is provided'

FETCH NEXT FROM acc_cursor INTO @accno, @balance;
END

CLOSE acc_cursor;
DEALLOCATE acc_cursor;
```


a) Find the details of customers who have personal accounts & balance is less than 2 lakhs.

b) Find the details of customers who have joint accounts. C
c) Write a trigger on ACCOUNT table such that if balance is less than 300 then customer should not withdraw the money.
d) Write a procedure on ACCOUNT & TRANSACTION table such that as user enters new transaction the balance is, updated in ACCOUNT table.

```
create database BankDB2
use BankDB2
CREATE TABLE ACCOUNT (
accno INT PRIMARY KEY CHECK (accno < 100),
opendate DATE NOT NULL,
acctype CHAR(1) NOT NULL CHECK (acctype IN ('P', 'J')),
balance MONEY NOT NULL CHECK (balance >= 0)
);

CREATE TABLE CUSTOMER (
cust_id INT PRIMARY KEY,
name VARCHAR(50) NOT NULL,
address VARCHAR(100) NOT NULL,
accno INT NOT NULL FOREIGN KEY REFERENCES ACCOUNT(accno)
);

CREATE TABLE TRANSACTION1 (
trans_id INT PRIMARY KEY,
trans_date DATE NOT NULL,
cno INT NOT NULL FOREIGN KEY REFERENCES CUSTOMER(cust_id),
trans_type CHAR(1) NOT NULL CHECK (trans_type IN ('C', 'D')),
amount MONEY NOT NULL CHECK (amount > 0)
);

INSERT INTO ACCOUNT (accno, opendate, acctype, balance) VALUES
(1, '2020-01-10', 'P', 50000),
(2, '2021-03-15', 'J', 250000),
(3, '2019-07-23', 'P', 180000),
(4, '2022-11-05', 'J', 300000),
(5, '2018-12-30', 'P', 220000),
(6, '2020-06-19', 'P', 40000),
(7, '2021-08-25', 'J', 350000),
(8, '2019-02-14', 'P', 120000),
(9, '2023-01-01', 'J', 99000),
(10, '2024-03-20', 'P', 150000);

INSERT INTO CUSTOMER (cust_id, name, address, accno) VALUES
(101, 'Ravi Kumar', 'Hyderabad', 1),
(102, 'Priya Sharma', 'Delhi', 2),
(103, 'Aman Verma', 'Mumbai', 3),
(104, 'Neha Gupta', 'Bangalore', 4),
(105, 'Vikram Singh', 'Chennai', 5),
(106, 'Meena Iyer', 'Pune', 6),
(107, 'Arjun Das', 'Kolkata', 7),
```

```
(108, 'Kiran Rao', 'Ahmedabad', 8),
(109, 'Nisha Patel', 'Surat', 9),
(110, 'Sahil Jain', 'Jaipur', 10);

INSERT INTO TRANSACTION1(trans_id, trans_date, cno, trans_type, amount) VALUES
(1001, '2024-04-01', 101, 'C', 5000),
(1002, '2024-04-02', 102, 'D', 10000),
(1003, '2024-04-03', 103, 'C', 15000),
(1004, '2024-04-04', 104, 'D', 20000),
(1005, '2024-04-05', 105, 'C', 25000),
(1006, '2024-04-06', 106, 'D', 5000),
(1007, '2024-04-07', 107, 'C', 10000),
(1008, '2024-04-08', 108, 'D', 8000),
(1009, '2024-04-09', 109, 'C', 12000),
(1010, '2024-04-10', 110, 'D', 4000);

--a)
SELECT C.*
FROM CUSTOMER C
JOIN ACCOUNT A ON C.accno = A.accno
WHERE A.acctype = 'P' AND A.balance < 200000;
--b)
SELECT C.*
FROM CUSTOMER C
JOIN ACCOUNT A ON C.accno = A.accno
WHERE A.acctype = 'J';

--c)
CREATE TRIGGER trg_PreventWithdrawalBelow300
ON TRANSACTION1
INSTEAD OF INSERT
AS
BEGIN
DECLARE @accno INT, @amount MONEY, @type CHAR(1);

SELECT TOP 1
@type = trans_type,
@amount = amount,
@accno = C.accno
FROM INSERTED I
JOIN CUSTOMER C ON I.cno = C.cust_id;

IF @type = 'D'
BEGIN
DECLARE @current MONEY;
SELECT @current = balance FROM ACCOUNT WHERE accno = @accno;

IF @current - @amount < 300
BEGIN
```

```
RAISERROR('Withdrawal not allowed. Balance would fall below 300.', 16, 1);
RETURN;
END
END

-- Proceed to insert and update balance
INSERT INTO TRANSACTION1 (trans_id, trans_date, cno, trans_type, amount)
SELECT trans_id, trans_date, cno, trans_type, amount 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;

--d)
CREATE PROCEDURE sp_AddTransaction
@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
DECLARE @currentBalance MONEY;
SELECT @currentBalance = balance FROM ACCOUNT WHERE accno = @accno;

IF @currentBalance - @amount < 300
BEGIN
RAISERROR('Withdrawal not allowed. Balance would fall below 300.', 16, 1);
RETURN;
END
END

INSERT INTO TRANSACTION1 (trans_id, trans_date, cno, trans_type, amount)
VALUES (@trans_id, @trans_date, @cno, @trans_type, @amount);

IF @trans_type = 'C'
UPDATE ACCOUNT SET balance = balance + @amount WHERE accno = @accno;
ELSE IF @trans_type = 'D'
UPDATE ACCOUNT SET balance = balance - @amount WHERE accno = @accno;
END;
```

```
EXEC sp_AddTransaction
@trans_id = 301,
@trans_date = '2025-04-21',
@cno = 1,
@trans_type = 'C',
@amount = 1500;

select * from ACCOUNT
```

---

page 10—

--a) Find the details of all transactions performed on account number 101. Also specify the name/names of cutomers who owns that account.
--b) Find the details ol amount credited within the period 15 -3-2012 to 18 -3 -2012.
--c) Write a trigger on insert on ACCOUNT table such that the account which is having balance less than or equal to 500 should not be debited.
--d) Write a procedure on ACCOUNT table to calculate interest on current balance from open_date to today's date. (Take interest rate from user).

```
CREATE DATABASE BankDB10;
GO
USE BankDB10;
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
);

CREATE TABLE CUSTOMER (
cust_id INT PRIMARY KEY,
name VARCHAR(50) NOT NULL,
address VARCHAR(100) NOT NULL,
accno INT NOT NULL,
FOREIGN KEY (accno) REFERENCES ACCOUNT(accno)
);

CREATE TABLE TRANSACTION1 (
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)
);

-- ACCOUNT records
INSERT INTO ACCOUNT VALUES
(101, '2015-03-01', 'P', 50000),
(102, '2017-04-15', 'J', 150000),
(103, '2020-01-10', 'P', 20000),
```

```
(104, '2021-06-20', 'P', 1000),
(105, '2022-12-25', 'J', 30000),
(106, '2019-07-11', 'J', 500),
(107, '2023-01-01', 'P', 2500),
(108, '2024-05-05', 'P', 47000),
(109, '2018-09-30', 'J', 95000),
(110, '2020-10-10', 'P', 80000);

-- CUSTOMER records
INSERT INTO CUSTOMER VALUES
(1, 'Alice', 'Delhi', 101),
(2, 'Bob', 'Mumbai', 102),
(3, 'Charlie', 'Kolkata', 103),
(4, 'David', 'Chennai', 104),
(5, 'Eva', 'Bangalore', 105),
(6, 'Frank', 'Hyderabad', 106),
(7, 'Grace', 'Jaipur', 107),
(8, 'Hannah', 'Pune', 108),
(9, 'Ian', 'Ahmedabad', 109),
(10, 'Jack', 'Lucknow', 110);

-- TRANSACTION records
INSERT INTO TRANSACTION1 VALUES
(1, '2012-03-15', 101, 'C', 10000),
(2, '2012-03-16', 101, 'C', 15000),
(3, '2012-03-17', 102, 'C', 20000),
(4, '2012-03-18', 103, 'D', 5000),
(5, '2024-01-01', 104, 'C', 4000),
(6, '2024-03-10', 105, 'D', 2000),
(7, '2024-04-01', 106, 'D', 100),
(8, '2024-04-15', 107, 'C', 3000),
(9, '2024-04-20', 108, 'D', 7000),
(10, '2024-04-21', 109, 'C', 6000);

--a)
SELECT T.*, C.name
FROM TRANSACTION1 T
JOIN CUSTOMER C ON T.accno = C.accno
WHERE T.accno = 101;

--b)
SELECT *
FROM TRANSACTION1
WHERE trans_type = 'C'
AND trans_date BETWEEN '2012-03-15' AND '2012-03-18';

--c)
CREATE TRIGGER trg_PreventLowBalanceAccount
ON ACCOUNT
```

```
INSTEAD OF INSERT
AS
BEGIN
IF EXISTS (SELECT 1 FROM INSERTED WHERE balance <= 500)
BEGIN
RAISERROR('Account cannot be created with balance less than or equal to 500.', 16, 1);
RETURN;
END

INSERT INTO ACCOUNT
SELECT * FROM INSERTED;
END;

--Insert with balance <= 500 (Should Fail)
INSERT INTO ACCOUNT (accno, open_date, acctype, balance)
VALUES (111, '2025-04-21', 'P', 500);

--Insert with balance > 500 (Should Succeed)
INSERT INTO ACCOUNT (accno, open_date, acctype, balance)
VALUES (112, '2025-04-21', 'P', 1000);

--You can verify it with:
SELECT * FROM ACCOUNT WHERE accno = 112;

--d)
CREATE PROCEDURE sp_CalculateInterest
@interestRate FLOAT
AS
BEGIN
SELECT
accno,
balance,
open_date,
DATEDIFF(DAY, open_date, GETDATE()) AS Days,
ROUND(balance * POWER(1 + @interestRate / 100.0, DATEDIFF(DAY, open_date, GETDATE()) / 365.0), 2) AS
BalanceWithInterest
FROM ACCOUNT;
END;
--output
EXEC sp_CalculateInterest @interestRate = 5;
```

Page—11

```
--a) Find the details of customers who have opened the accounts within the period 25-3-2012 to 28-3-2012.
--b) Find the details of customers who have joint accounts & balance is less than 2 lakhs.
--c) Write a trigger TRANSACTION on table to calculate the current balance of the account on which transaction is made.
--( if trans_type = c then bal = bal amt else if trans_type = d then bal = bal amt)
--d) write a cursor on CUSTOMER table to fetch the last row.
CREATE DATABASE BankDB3;
USE BankDB3;
```

```
-- ACCOUNT Table
CREATE TABLE ACCOUNT (
accno INT CHECK (accno < 100) PRIMARY KEY,
open_date DATE NOT NULL,
acctype CHAR(1) CHECK (acctype IN ('P', 'M')) NOT NULL, -- P: Personal, M: Moira (assumed 'Moira' meant 'M')
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 REFERENCES ACCOUNT(accno)
);

-- TRANSACTION Table
CREATE TABLE TRANSACTION1 (
trans_id INT PRIMARY KEY,
trans_date DATE NOT NULL,
accno INT NOT NULL FOREIGN KEY REFERENCES ACCOUNT(accno),
trans_type CHAR(1) CHECK (trans_type IN ('C', 'D')) NOT NULL,
amount MONEY NOT NULL
);

-- ACCOUNT Records
INSERT INTO ACCOUNT VALUES
(1, '2012-03-25', 'P', 100000),
(2, '2012-03-26', 'M', 150000),
(3, '2012-03-27', 'P', 250000),
(4, '2012-03-28', 'M', 180000),
(5, '2012-04-01', 'P', 90000),
(6, '2012-04-02', 'M', 300000),
(7, '2012-03-20', 'P', 170000),
(8, '2012-03-21', 'M', 60000),
(9, '2012-03-29', 'P', 220000),
(10, '2012-04-05', 'M', 50000);

-- CUSTOMER Records
INSERT INTO CUSTOMER VALUES
(101, 'Alice', 'NY', 1),
(102, 'Bob', 'LA', 2),
(103, 'Charlie', 'NY', 3),
(104, 'David', 'Texas', 4),
(105, 'Eva', 'Florida', 5),
(106, 'Frank', 'Boston', 6),
(107, 'Grace', 'Chicago', 7),
(108, 'Helen', 'Seattle', 8),
(109, 'Ian', 'Austin', 9),
```

```
(110, 'Jane', 'Vegas', 10);

-- TRANSACTION Records
INSERT INTO TRANSACTION1 VALUES
(201, '2025-04-01', 1, 'C', 5000),
(202, '2025-04-02', 2, 'D', 20000),
(203, '2025-04-02', 3, 'C', 30000),
(204, '2025-04-03', 4, 'D', 10000),
(205, '2025-04-03', 5, 'C', 15000),
(206, '2025-04-04', 6, 'C', 20000),
(207, '2025-04-04', 7, 'D', 25000),
(208, '2025-04-05', 8, 'C', 10000),
(209, '2025-04-06', 9, 'D', 15000),
(210, '2025-04-06', 10, 'C', 5000);

--a)
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';

--b)
SELECT C.*
FROM CUSTOMER C
JOIN ACCOUNT A ON C.accno = A.accno
WHERE A.acctype = 'M' AND A.balance < 200000;

--c)
CREATE TRIGGER trg_update_balance
ON TRANSACTION1
AFTER INSERT
AS
BEGIN
SET NOCOUNT ON;

UPDATE ACCOUNT
SET balance =
CASE
WHEN i.trans_type = 'C' THEN balance + i.amount
WHEN i.trans_type = 'D' THEN balance - i.amount
END
FROM ACCOUNT A
INNER JOIN inserted i ON A.accno = i.accno;
END;

--Insert a new transaction (e.g., credit ₹5000 to accno 1):
INSERT INTO TRANSACTION1 VALUES (211, '2025-04-23', 1, 'C', 15000);
PRINT 'Trigger executed: Account balance updated.';
--View the updated balance in ACCOUNT:
```

```
SELECT * FROM ACCOUNT WHERE accno = 1;
--see the updated transactions
select * from TRANSACTION1

--d)
DECLARE @cust_id INT, @name VARCHAR(100), @address VARCHAR(200), @accno INT;

DECLARE customer_cursor CURSOR FOR
SELECT cust_id, name, address, accno
FROM CUSTOMER
ORDER BY cust_id;

OPEN customer_cursor;

FETCH NEXT FROM customer_cursor INTO @cust_id, @name, @address, @accno;

WHILE @@FETCH_STATUS = 0
BEGIN
FETCH NEXT FROM customer_cursor INTO @cust_id, @name, @address, @accno;
END

CLOSE customer_cursor;
DEALLOCATE customer_cursor;

-- Print last fetched values
PRINT 'Last Customer Row:';
PRINT 'ID: ' + CAST(@cust_id AS VARCHAR) + ', Name: ' + @name + ', Address: ' + @address + ', AccNo: ' + CAST(@accno AS
VARCHAR);
```

Page-12---

--a) For every project located in 'jalgaon". List the pno,the controlling detptno and dept manager last name.
--b) For each project on which more than two employee work, Find the pno, pname & no. of employees who work on the project.
--c)create a view that has the deptname,manager name & manager salary for every dept.
--d) Express the following constraint as SQL assertions - "salary of employee must not be greater than the salary of the manager of the dept".

```
CREATE DATABASE CompanyDB;
USE CompanyDB;

-- DEPT Table
CREATE TABLE DEPT (
dname VARCHAR(50) NOT NULL,
dnum INT CHECK (dnum < 10000) PRIMARY KEY,
mgrssn CHAR(9) NOT NULL,
dlocation VARCHAR(50) NOT NULL
);

-- EMPLOYEE Table
CREATE TABLE EMPLOYEE (
```

```
fname VARCHAR(50) NOT NULL,
mname VARCHAR(50) NOT NULL,
ssn CHAR(9) PRIMARY KEY,
sex CHAR(1) NOT NULL,
salary MONEY NOT NULL,
joindate DATE NOT NULL,
superssn CHAR(9) NOT NULL,
dno INT NOT NULL FOREIGN KEY 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 REFERENCES DEPT(dnum)
);

-- WORK_ON Table
CREATE TABLE WORK_ON (
ssn CHAR(9) NOT NULL FOREIGN KEY REFERENCES EMPLOYEE(ssn),
pno INT NOT NULL FOREIGN KEY REFERENCES PROJECT(pno),
hours FLOAT NOT NULL,
PRIMARY KEY (ssn, pno)
);
-- DEPT
INSERT INTO DEPT VALUES
('HR', 1, '123456789', 'Mumbai'),
('Finance', 2, '234567891', 'Pune'),
('IT', 3, '345678912', 'Jalgaon'),
('Sales', 4, '456789123', 'Delhi');

-- EMPLOYEE
INSERT INTO EMPLOYEE VALUES
('John', 'A', '123456789', 'M', 90000, '2020-01-10', '123456789', 1),
('Emma', 'B', '234567891', 'F', 85000, '2021-02-12', '123456789', 2),
('Raj', 'C', '345678912', 'M', 95000, '2019-03-15', '234567891', 3),
('Neha', 'D', '456789123', 'F', 80000, '2022-04-20', '345678912', 4),
('Amit', 'E', '567891234', 'M', 75000, '2022-05-01', '123456789', 3),
('Sara', 'F', '678912345', 'F', 70000, '2023-06-18', '345678912', 3),
('Ravi', 'G', '789123456', 'M', 85000, '2023-07-21', '234567891', 2),
('Priya', 'H', '891234567', 'F', 68000, '2024-08-22', '345678912', 4),
('Nina', 'I', '912345678', 'F', 73000, '2023-09-23', '456789123', 4),
('Kunal', 'J', '999999999', 'M', 72000, '2022-10-30', '123456789', 1);

-- PROJECT
INSERT INTO PROJECT VALUES
('Migration', 101, 'Jalgaon', 3),
('Upgrade', 102, 'Mumbai', 1),
```

```
('Automation', 103, 'Jalgaon', 3),
('Audit', 104, 'Delhi', 4);

-- WORK_ON
INSERT INTO WORK_ON VALUES
('345678912', 101, 10),
('567891234', 101, 5),
('678912345', 101, 7),
('123456789', 102, 6),
('234567891', 102, 5),
('789123456', 103, 8),
('891234567', 103, 9),
('912345678', 103, 7),
('456789123', 104, 6),
('999999999', 104, 5);

select * from PROJECT
--a)
SELECT pno, dnumber, pname
FROM PROJECT
JOIN DEPT D ON dnumber = D.dnum
JOIN EMPLOYEE E ON D.mgrssn = E.ssn
WHERE plocation = 'Jalgaon';

--b)
SELECT P.pno, P.pname, COUNT(W.ssn) AS num_employees
FROM PROJECT P
JOIN WORK_ON W ON P.pno = W.pno
GROUP BY P.pno, P.pname
HAVING COUNT(W.ssn) > 2;

--c)
CREATE VIEW DeptManagerDetails AS
SELECT D.dname, E.fname + ' ' + E.mname AS ManagerName, E.salary AS ManagerSalary
FROM DEPT D
JOIN EMPLOYEE E ON D.mgrssn = E.ssn;
--see view
select * from DeptManagerDetails

--d)
CREATE TRIGGER trg_CheckSalary
ON EMPLOYEE
AFTER INSERT, UPDATE
AS
BEGIN
IF EXISTS (
SELECT 1
FROM inserted i
```

```
JOIN EMPLOYEE mgr ON i.dno = mgr.dno AND i.salary > mgr.salary AND mgr.ssn = (SELECT mgrssn FROM DEPT WHERE
dnum = i.dno)
)
BEGIN
RAISERROR('Employee salary cannot be greater than department manager salary.', 16, 1);
ROLLBACK TRANSACTION;
END
END;

-- Insert that is valid:This will SUCCEED
INSERT INTO EMPLOYEE VALUES
('Test2', 'Y', '222222222', 'F', 85000, '2024-01-01', '123456789', 1);

SELECT * FROM EMPLOYEE WHERE ssn = '222222222';
```

```
--a) For each employee, Find the employee first & last name & the first & last name of his or her immediate supervisor.
--b) For each dept, Find the deptno, the no. of employees in the dept & their average salary.
--c) Create a view that has pname,controlling dept name, no of employees, & total hours worked per week on the project for
each project with more than one employee working on it.
--d) Create a procedure on EMPLOYEE table to determine the employees who will get promotion. (An employee will get
promotion after working on 5 projects.)

CREATE DATABASE CompanyDB2;
USE CompanyDB2;

-- DEPT Table
CREATE TABLE DEPT (
dname VARCHAR(50) NOT NULL,
dnum INT NOT NULL CHECK (dnum < 10000) PRIMARY KEY,
mgrssn CHAR(9) NOT NULL,
dlocation VARCHAR(50) NOT NULL
);

-- EMPLOYEE Table
CREATE TABLE EMPLOYEE (
fname VARCHAR(50) NOT NULL,
lname VARCHAR(50) NOT NULL,
ssn CHAR(9) NOT NULL PRIMARY KEY,
sex CHAR(1) NOT NULL,
salary MONEY NOT NULL,
joindate DATE NOT NULL,
superssn CHAR(9) NOT NULL,
dno INT NOT NULL FOREIGN KEY REFERENCES DEPT(dnum)
);

-- PROJECT Table
CREATE TABLE PROJECT (
pname VARCHAR(50) NOT NULL,
pno INT NOT NULL PRIMARY KEY,
plocation VARCHAR(50) NOT NULL,
```

```
dnumber INT NOT NULL FOREIGN KEY REFERENCES DEPT(dnum)
);

-- WORK_ON Table
CREATE TABLE WORK_ON (
ssn CHAR(9) NOT NULL FOREIGN KEY REFERENCES EMPLOYEE(ssn),
pno INT NOT NULL FOREIGN KEY REFERENCES PROJECT(pno),
hours FLOAT NOT NULL,
PRIMARY KEY (ssn, pno)
);

-- DEPT
INSERT INTO DEPT VALUES
('HR', 1001, '123456789', 'Mumbai'),
('Finance', 1002, '234567891', 'Delhi'),
('IT', 1003, '345678912', 'Pune');

-- EMPLOYEE
INSERT INTO EMPLOYEE VALUES
('John', 'Doe', '123456789', 'M', 90000, '2015-06-01', '123456789', 1001),
('Jane', 'Smith', '234567891', 'F', 85000, '2016-05-10', '123456789', 1002),
('Mike', 'Brown', '345678912', 'M', 95000, '2017-04-12', '234567891', 1003),
('Emma', 'White', '456789123', 'F', 78000, '2018-03-15', '345678912', 1003),
('Amit', 'Kumar', '567891234', 'M', 72000, '2019-02-20', '345678912', 1003),
('Neha', 'Sharma', '678912345', 'F', 70000, '2020-01-25', '345678912', 1002),
('Ravi', 'Verma', '789123456', 'M', 88000, '2020-12-05', '234567891', 1002),
('Priya', 'Singh', '891234567', 'F', 69000, '2021-11-11', '345678912', 1003),
('Kunal', 'Joshi', '912345678', 'M', 71000, '2022-10-10', '345678912', 1003),
('Anita', 'Mehta', '999999999', 'F', 74000, '2023-09-09', '345678912', 1003);

-- PROJECT
INSERT INTO PROJECT VALUES
('Payroll', 101, 'Mumbai', 1001),
('Audit', 102, 'Delhi', 1002),
('Migration', 103, 'Pune', 1003),
('ERP', 104, 'Pune', 1003),
('Website', 105, 'Pune', 1003);

-- WORK_ON
INSERT INTO WORK_ON VALUES
('345678912', 101, 5),
('456789123', 101, 6),
('567891234', 102, 5),
('678912345', 102, 6),
('789123456', 103, 7),
('891234567', 103, 8),
('912345678', 103, 4),
('999999999', 104, 5),
('345678912', 104, 6),
```

('567891234', 105, 4);

```
--a)
SELECT
E.fname AS EmpFname, E.lname AS EmpLname,
S.fname AS SupFname, S.lname AS SupLname
FROM EMPLOYEE E
JOIN EMPLOYEE S ON E.superssn = S.ssn;

--b)
SELECT
dno,
COUNT(*) AS EmployeeCount,
AVG(salary) AS AvgSalary
FROM EMPLOYEE
GROUP BY dno;

--c)
CREATE VIEW ProjectSummary AS
SELECT
P.pname,
D.dname AS DeptName,
COUNT(W.ssn) AS NumEmployees,
SUM(W.hours) AS TotalHours
FROM PROJECT P
JOIN DEPT D ON P.dnumber = D.dnum
JOIN WORK_ON W ON P.pno = W.pno
GROUP BY P.pname, D.dname
HAVING COUNT(W.ssn) > 1;
--see view
select * from ProjectSummary

--d)
CREATE PROCEDURE GetPromotedEmployees1
AS
BEGIN
SELECT E.fname, E.lname, COUNT(W.pno) AS ProjectCount
FROM EMPLOYEE E
JOIN WORK_ON W ON E.ssn = W.ssn
GROUP BY E.fname, E.lname
HAVING COUNT(W.pno) >= 4;
END;
--run
EXEC GetPromotedEmployees1;
```


--a) Find the ssn of all employees who work on pno 101, 102 or 103.

--b) Make a list of all pno for project that involve an employee whose last name is 'sonar' either as a worker or as a manager of the dept that control the project.

--c) Write a trigger on insert on WORK_ON table such that if total work hours of employee in company is less than 20 hours then his salary is deducted.
--d) Write a cursor on PROJECT table to fetch the first row from the table & display the total number of rows present in the table.

```
create database employee14
use employee14

CREATE TABLE DEPT (
dname VARCHAR(50) NOT NULL,
dnum INT NOT NULL CHECK (dnum < 10000),
mgrssn CHAR(9) NOT NULL,
dlocation VARCHAR(50) NOT NULL,
PRIMARY KEY (dnum)
);

CREATE TABLE EMPLOYEE (
fname VARCHAR(50) NOT NULL,
Mname VARCHAR(50) NOT NULL,
ssn CHAR(9) NOT NULL PRIMARY KEY,
sex CHAR(1) NOT NULL,
salary MONEY NOT NULL,
joindate DATE NOT NULL,
superssn CHAR(9) NOT NULL,
dno INT NOT NULL FOREIGN KEY REFERENCES DEPT(dnum)
);

CREATE TABLE PROJECT (
pname VARCHAR(50) NOT NULL,
pno INT NOT NULL PRIMARY KEY,
plocation VARCHAR(50) NOT NULL,
dnumber INT NOT NULL FOREIGN KEY REFERENCES DEPT(dnum)
);

CREATE TABLE WORK_ON (
ssn CHAR(9) NOT NULL FOREIGN KEY REFERENCES EMPLOYEE(ssn),
pno INT NOT NULL FOREIGN KEY REFERENCES PROJECT(pno),
hours INT NOT NULL,
PRIMARY KEY (ssn, pno)
);

-- DEPT
INSERT INTO DEPT VALUES
('HR', 1010, '123456789', 'Mumbai'),
('Finance', 1020, '234567891', 'Delhi'),
('IT', 1030, '345678912', 'Pune');

-- EMPLOYEE
INSERT INTO EMPLOYEE VALUES
```

('Amit', 'Sonar', '123456789', 'M', 80000, '2020-01-01', '234567891', 1010),
('Sneha', 'P', '234567891', 'F', 85000, '2019-01-01', '345678912', 1020),
('Ravi', 'Q', '345678912', 'M', 75000, '2021-01-01', '123456789', 1030),
('Rohan', 'R', '456789123', 'M', 90000, '2022-01-01', '234567891', 1010),
('Pooja', 'S', '567891234', 'F', 82000, '2022-06-01', '345678912', 1020),
('Anil', 'Sonar', '678912345', 'M', 81000, '2023-01-01', '123456789', 1030),
('Sonal', 'U', '789123456', 'F', 86000, '2022-03-01', '345678912', 1010),
('Mayur', 'V', '891234567', 'M', 83000, '2021-07-01', '234567891', 1020),
('Kajal', 'W', '912345678', 'F', 79000, '2020-05-01', '123456789', 1030),
('Tina', 'X', '101234567', 'F', 76000, '2023-02-01', '345678912', 1010);

```
-- PROJECT
INSERT INTO PROJECT VALUES
('ERP', 101, 'Mumbai', 1010),
('Audit', 102, 'Delhi', 1020),
('Migration', 103, 'Pune', 1030),
('Web Dev', 104, 'Mumbai', 1010),
('Cloud', 105, 'Delhi', 1020),
('Support', 106, 'Pune', 1030),
('Infra', 107, 'Pune', 1030),
('Automation', 108, 'Mumbai', 1010),
('Testing', 109, 'Delhi', 1020),
('AI Dev', 110, 'Pune', 1030);

-- WORK_ON
INSERT INTO WORK_ON VALUES
('123456789', 101, 8),
('234567891', 102, 6),
('345678912', 103, 5),
('456789123', 104, 7),
('567891234', 105, 4),
('678912345', 106, 2),
('789123456', 107, 3),
('891234567', 108, 9),
('912345678', 109, 5),
('101234567', 110, 1);

--a)
SELECT ssn
FROM WORK_ON
WHERE pno IN (101, 102, 103);

--b)
-- As Worker
SELECT DISTINCT W.pno
FROM WORK_ON W
JOIN EMPLOYEE E ON W.ssn = E.ssn
WHERE E.Mname = 'Sonar'
```

```
UNION

-- As 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.Mname = 'Sonar';

--c)
CREATE TRIGGER trg_DeductSalary
ON WORK_ON
AFTER INSERT
AS
BEGIN
SET NOCOUNT ON;

UPDATE EMPLOYEE
SET salary = salary * 0.9
WHERE ssn IN (
SELECT i.ssn
FROM inserted i
GROUP BY i.ssn
HAVING (
SELECT SUM(hours)
FROM WORK_ON
WHERE ssn = i.ssn
) < 20
);
END;
--Step 1: Check salary before inserting into WORK_ON
SELECT ssn, salary FROM EMPLOYEE WHERE ssn = '678912345';
--Step 2: Insert new data into WORK_ON with total hours < 20
INSERT INTO WORK_ON VALUES ('678912345', 111, 3);
--Step 3: Check salary again
SELECT ssn, salary FROM EMPLOYEE WHERE ssn = '678912345';

--d)
DECLARE @pname VARCHAR(50), @pno INT, @total INT;

-- Get total number of rows
SELECT @total = COUNT(*) FROM PROJECT;

-- Declare cursor
DECLARE proj_cursor CURSOR FOR
SELECT pname, pno FROM PROJECT ORDER BY pno;

-- Open and fetch first
OPEN proj_cursor;
```

```
FETCH NEXT FROM proj_cursor INTO @pname, @pno;

-- Display results
PRINT 'First Project Name: ' + @pname;
PRINT 'First Project Number: ' + CAST(@pno AS VARCHAR);
PRINT 'Total Projects: ' + CAST(@total AS VARCHAR);

-- Close cursor
CLOSE proj_cursor;
DEALLOCATE proj_cursor;
```

Page 15—

```
--a) Find the name of books which is issued maximum times.
--b) Find the detail information of books that are issued by computer department students.
--c) Write a procedure to calculate the fines for the books which are not return on or before due date. no.of days = (ret_date - due_date) fine = no.of days (ret_date - due_date) * 10
--d)Write a trigger on insert of ISSUETABLE such that duedate = issuedate + 7



create database library
use library

CREATE TABLE BOOKMASTER (
bid INT NOT NULL PRIMARY KEY,
title VARCHAR(100) NOT NULL,
author VARCHAR(100) NOT NULL,
price DECIMAL(10, 2) NOT NULL
);

CREATE TABLE STUDENTMASTER (
enrollno INT NOT NULL PRIMARY KEY,
sname VARCHAR(100) NOT NULL,
class VARCHAR(50) NOT NULL,
dept VARCHAR(50) NOT NULL
);

CREATE TABLE ACCESSIONTABLE (
bid INT NOT NULL FOREIGN KEY REFERENCES BOOKMASTER(bid),
accession_no INT NOT NULL PRIMARY KEY,
avail CHAR(1) CHECK (avail IN ('T', 'F')) NOT NULL
);

CREATE TABLE ISSUETABLE (
issueid INT NOT NULL PRIMARY KEY,
accession_no INT NOT NULL FOREIGN KEY REFERENCES ACCESSIONTABLE(accession_no),
enrollno INT NOT NULL FOREIGN KEY REFERENCES STUDENTMASTER(enrollno),
issuedate DATE NOT NULL,
duedate DATE NOT NULL,
ret_date DATE,
bid INT NOT NULL FOREIGN KEY REFERENCES BOOKMASTER(bid)
```

```
);

-- BOOKMASTER
INSERT INTO BOOKMASTER VALUES
(101, 'Database Systems', 'Korth', 500),
(102, 'Operating Systems', 'Silberschatz', 600),
(103, 'Data Structures', 'Tanenbaum', 450),
(104, 'Algorithms', 'Cormen', 700),
(105, 'Computer Networks', 'Forouzan', 650),
(106, 'Software Engineering', 'Pressman', 550),
(107, 'Discrete Mathematics', 'Rosen', 400),
(108, 'Artificial Intelligence', 'Russell', 800),
(109, 'Computer Organization', 'Hennessey', 600),
(110, 'Introduction to C', 'Kernighan', 350);

-- STUDENTMASTER
INSERT INTO STUDENTMASTER VALUES
(201, 'Amit', 'BTech', 'Computer'),
(202, 'Sneha', 'MTech', 'Electrical'),
(203, 'Ravi', 'BTech', 'Computer'),
(204, 'Pooja', 'BTech', 'Mechanical'),
(205, 'Anil', 'MTech', 'Computer'),
(206, 'Tina', 'BTech', 'Civil'),
(207, 'Mayur', 'BTech', 'Computer'),
(208, 'Kajal', 'MTech', 'Electrical'),
(209, 'Rohan', 'BTech', 'Computer'),
(210, 'Sonal', 'BTech', 'Mechanical');

-- ACCESSIONTABLE
INSERT INTO ACCESSIONTABLE VALUES
(101, 1001, 'T'),
(102, 1002, 'F'),
(103, 1003, 'T'),
(104, 1004, 'F'),
(105, 1005, 'T'),
(106, 1006, 'T'),
(107, 1007, 'F'),
(108, 1008, 'T'),
(109, 1009, 'T'),
(110, 1010, 'T');

-- ISSUETABLE
INSERT INTO ISSUETABLE VALUES
(1, 1001, 201, '2025-01-01', '2025-01-08', '2025-01-09', 101),
(2, 1002, 203, '2025-01-03', '2025-01-10', '2025-01-12', 102),
(3, 1003, 204, '2025-01-04', '2025-01-11', '2025-01-14', 103),
(4, 1004, 205, '2025-01-06', '2025-01-13', '2025-01-14', 104),
(5, 1005, 207, '2025-01-07', '2025-01-14', '2025-01-16', 105),
(6, 1006, 208, '2025-01-08', '2025-01-15', '2025-01-18', 106),
```

```
(7, 1007, 209, '2025-01-09', '2025-01-16', '2025-01-19', 107),
(8, 1008, 202, '2025-01-10', '2025-01-17', '2025-01-18', 108),
(9, 1009, 206, '2025-01-12', '2025-01-19', '2025-01-21', 109),
(10, 1010, 210, '2025-01-13', '2025-01-20', '2025-01-22', 110);

--a)
SELECT TOP 1 b.title, COUNT(i.issueid) AS issue_count
FROM ISSUETABLE i
JOIN BOOKMASTER b ON i.bid = b.bid
GROUP BY b.title
ORDER BY issue_count DESC;

--b)
SELECT b.*
FROM BOOKMASTER b
JOIN ISSUETABLE i ON b.bid = i.bid
JOIN STUDENTMASTER s ON i.enrollno = s.enrollno
WHERE s.dept = 'Computer';

--c)
CREATE PROCEDURE CalculateFines
AS
BEGIN
DECLARE @enrollno INT, @ret_date DATE, @duedate DATE, @no_of_days INT, @fine DECIMAL(10,2);

DECLARE fine_cursor CURSOR FOR
SELECT enrollno, ret_date, duedate
FROM ISSUETABLE
WHERE ret_date > duedate;

OPEN fine_cursor;
FETCH NEXT FROM fine_cursor INTO @enrollno, @ret_date, @duedate;

WHILE @@FETCH_STATUS = 0
BEGIN
SET @no_of_days = DATEDIFF(DAY, @duedate, @ret_date);
SET @fine = @no_of_days * 10;

PRINT 'Student: ' + CAST(@enrollno AS VARCHAR) + ' Fine: ' + CAST(@fine AS VARCHAR);

FETCH NEXT FROM fine_cursor INTO @enrollno, @ret_date, @duedate;
END

CLOSE fine_cursor;
DEALLOCATE fine_cursor;
END;

EXEC CalculateFines;
```

```
--d)
CREATE TRIGGER trg_SetDueDate
ON ISSUETABLE
AFTER INSERT
AS
BEGIN
SET NOCOUNT ON;

UPDATE i
SET duedate = DATEADD(DAY, 7, i.issuedate)
FROM ISSUETABLE i
INNER JOIN inserted ins ON i.issueid = ins.issueid;
END;

--Inserting a record that fires the trigger
-- Insert sample data to test the trigger
INSERT INTO ISSUETABLE (issueid, accession_no, enrollno, issuedate, ret_date, bid)
VALUES (101, 201, '001', '2025-04-22', '2025-04-30', 1);
--Verifying the effect
SELECT * FROM ISSUETABLE WHERE issueid = 101;
```

Page 16—

```
--a) Find the detail information of the students who have issued books Between two given dates.
--b) Create a view that display all the accession infortnatiOn for a book having bid = 100
--c)Write a cursor to fetch last record from view in
--d) Find the information of books issued by MCA students.

create database library2
use library2

-- 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
CREATE TABLE ACCESSIONTABLE (
   accession_no INT PRIMARY KEY,
   avail CHAR(1) NOT NULL CHECK (avail IN ('T', 'F'))  -- T = not issued, F = issued
```

```
);

-- 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 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)
);

-- BOOKMASTER
INSERT INTO BOOKMASTER VALUES
(100, 'Database Systems', 'Elmasri', 500),
(101, 'Operating Systems', 'Silberschatz', 450),
(102, 'Data Structures', 'Lafore', 300),
(103, 'Java Programming', 'Deitel', 550),
(104, 'Computer Networks', 'Tanenbaum', 400),
(105, 'Python Basics', 'Zelle', 350),
(106, 'Machine Learning', 'Goodfellow', 650),
(107, 'Web Tech', 'Acharya', 250),
(108, 'AI', 'Russell', 700),
(109, 'Cloud Computing', 'Buyya', 600);

-- STUDENTMASTER
INSERT INTO STUDENTMASTER VALUES
(1, 'Alice', 'FY', 'MCA'),
(2, 'Bob', 'SY', 'MCA'),
(3, 'Charlie', 'TY', 'BCA'),
(4, 'David', 'FY', 'MCA'),
(5, 'Eva', 'SY', 'BSc'),
(6, 'Frank', 'TY', 'MCA'),
(7, 'Grace', 'FY', 'MCA'),
(8, 'Heidi', 'TY', 'BCA'),
(9, 'Ivan', 'SY', 'MSc'),
(10, 'Judy', 'TY', 'MCA');

-- ACCESSIONTABLE
INSERT INTO ACCESSIONTABLE VALUES
(1001, 'F'),
(1002, 'F'),
(1003, 'T'),
(1004, 'F'),
(1005, 'F'),
```

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

-- ISSUETABLE
INSERT INTO ISSUETABLE VALUES
(1, 1001, 1, '2024-04-01', '2024-04-15', '2024-04-10', 100),
(2, 1002, 2, '2024-04-02', '2024-04-16', '2024-04-11', 101),
(3, 1004, 3, '2024-04-03', '2024-04-17', '2024-04-13', 102),
(4, 1005, 4, '2024-04-04', '2024-04-18', '2024-04-14', 103),
(5, 1008, 5, '2024-04-05', '2024-04-19', '2024-04-15', 104),
(6, 1009, 6, '2024-04-06', '2024-04-20', '2024-04-16', 105),
(7, 1001, 7, '2024-04-07', '2024-04-21', '2024-04-17', 106),
(8, 1002, 8, '2024-04-08', '2024-04-22', '2024-04-18', 107),
(9, 1004, 9, '2024-04-09', '2024-04-23', '2024-04-19', 108),
(10, 1005, 10, '2024-04-10', '2024-04-24', '2024-04-20', 109);

--a)
-- Replace the dates as needed
SELECT DISTINCT S.*
FROM STUDENTMASTER S
JOIN ISSUETABLE I ON S.stud_enrollno = I.stud_enrollno
WHERE I.issuedate BETWEEN '2024-04-02' AND '2024-04-07';

--b) View for book with bid = 100

CREATE VIEW vw_book_accession AS
SELECT A.*
FROM ACCESSIONTABLE A
JOIN ISSUETABLE I ON A.accession_no = I.accession_no
WHERE I.bid = 100;

select * from vw_book_accession

--c) Cursor to fetch last record from view
DECLARE @accession_no INT, @avail CHAR(1);

DECLARE accession_cursor CURSOR FOR
  SELECT accession_no, avail
  FROM vw_book_accession
  ORDER BY accession_no DESC;  -- Ensures last record is first in reverse order

OPEN accession_cursor;

FETCH NEXT FROM accession_cursor INTO @accession_no, @avail;

PRINT 'Last Record (by accession_no):';
```

```
PRINT 'Accession No: ' + CAST(@accession_no AS VARCHAR) + ', Availability: ' + @avail;

CLOSE accession_cursor;
DEALLOCATE accession_cursor;

--d) Info of books issued by MCA students
SELECT B.*
FROM BOOKMASTER B
JOIN ISSUETABLE I ON B.bid = I.bid
JOIN STUDENTMASTER S ON I.stud_enrollno = S.stud_enrollno
WHERE S.dept = 'MCA';
```

Page 17—

```
--a) Write a procedure for giving the detail information of books available in the library.
--b) Find the number of books issued by each student.
--c) Write a trigger such that the return date should not exceed today's date.
--d) Find the number of books available in the library & written by "Henry Korth".
create database library3
use library3

-- BOOKMASTER
CREATE TABLE BOOKMASTER (
   bid INT PRIMARY KEY,
   title VARCHAR(100) NOT NULL,
   author VARCHAR(100) NOT NULL,
   price DECIMAL(10,2) NOT NULL
);

-- STUDENTMASTER
CREATE TABLE STUDENTMASTER (
   stud_enroll_no INT PRIMARY KEY,
   sname VARCHAR(100) NOT NULL,
   class VARCHAR(50) NOT NULL,
   dept VARCHAR(100) 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')), -- T = available, F = issued
   FOREIGN KEY (bid) REFERENCES BOOKMASTER(bid)
);

-- ISSUETABLE
CREATE TABLE ISSUETABLE (
   issueid INT PRIMARY KEY,
   accession_no INT NOT NULL,
   stud_enroll_no INT NOT NULL,
   issuedate DATE NOT NULL,
```

```
  duedate DATE NOT NULL,
  ret_date DATE NOT NULL,
  FOREIGN KEY (accession_no) REFERENCES ACCESSIONTABLE(accession_no),
  FOREIGN KEY (stud_enroll_no) REFERENCES STUDENTMASTER(stud_enroll_no)
);

-- BOOKMASTER
INSERT INTO BOOKMASTER VALUES
(1, 'DBMS Concepts', 'Henry Korth', 500),
(2, 'Operating Systems', 'Silberschatz', 450),
(3, 'Java Basics', 'Deitel', 400),
(4, 'Networking', 'Tanenbaum', 420),
(5, 'AI Intro', 'Russell', 520),
(6, 'Python Programming', 'Zelle', 350),
(7, 'Cloud Computing', 'Buyya', 480),
(8, 'Data Structures', 'Lafore', 390),
(9, 'Machine Learning', 'Goodfellow', 650),
(10, 'Web Development', 'Acharya', 370);

-- STUDENTMASTER
INSERT INTO STUDENTMASTER VALUES
(101, 'Alice', 'FY', 'MCA'),
(102, 'Bob', 'SY', 'MCA'),
(103, 'Charlie', 'TY', 'BCA'),
(104, 'David', 'FY', 'BSc'),
(105, 'Eva', 'SY', 'MSc'),
(106, 'Frank', 'TY', 'MCA'),
(107, 'Grace', 'FY', 'MCA'),
(108, 'Heidi', 'SY', 'BSc'),
(109, 'Ivan', 'TY', 'BCA'),
(110, 'Judy', 'FY', 'MCA');

-- ACCESSIONTABLE
INSERT INTO ACCESSIONTABLE VALUES
(1, 1001, 'T'),
(1, 1002, 'F'),
(2, 1003, 'F'),
(3, 1004, 'T'),
(4, 1005, 'F'),
(5, 1006, 'T'),
(6, 1007, 'T'),
(7, 1008, 'F'),
(1, 1009, 'T'),
(8, 1010, 'F');

-- ISSUETABLE
INSERT INTO ISSUETABLE VALUES
(1, 1002, 101, '2024-04-01', '2024-04-10', '2024-04-08'),
(2, 1003, 102, '2024-04-02', '2024-04-12', '2024-04-09'),
```

```
(3, 1005, 103, '2024-04-03', '2024-04-13', '2024-04-10'),
(4, 1008, 104, '2024-04-04', '2024-04-14', '2024-04-11'),
(5, 1010, 105, '2024-04-05', '2024-04-15', '2024-04-12');

--a) Procedure: Show available books in the library
CREATE PROCEDURE sp_AvailableBooks
AS
BEGIN
  SELECT B.bid, B.title, B.author, B.price, A.accession_no
  FROM BOOKMASTER B
  JOIN ACCESSIONTABLE A ON B.bid = A.bid
  WHERE A.avail = 'T';
END;
-- run
EXEC sp_AvailableBooks;

--b) Find number of books issued by each student
SELECT S.stud_enroll_no, S.sname, COUNT(I.issueid) AS issued_books
FROM STUDENTMASTER S
LEFT JOIN ISSUETABLE I ON S.stud_enroll_no = I.stud_enroll_no
GROUP BY S.stud_enroll_no, S.sname;

--c) Trigger: Ensure return date doesn't exceed today's date
CREATE TRIGGER trg_CheckReturnDate
ON ISSUETABLE
FOR INSERT, UPDATE
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 TRANSACTION;
  END
END;

--d) Number of available books written by "Henry Korth"
SELECT COUNT(*) AS available_books_by_korth
FROM ACCESSIONTABLE A
JOIN BOOKMASTER B ON A.bid = B.bid
WHERE A.avail = 'T' AND B.author = 'Henry Korth';
```

Page 18—

--a) For every project located in `jalgaon'. List the pno,the controlling detptno and dept manager last name.

--b) For each project on which more than two employee work, Find the pno, pnarne & no. of employees who work on the project.

--c) create a view that has the deptna.me,manager name 8z manager salary for every dept.

--d) Express the following constraint as SQL assertions - "salary of employee must not be greater than the salary of the manager of the dept".

```
create database company
use company

-- DEPT
CREATE TABLE DEPT (
   dname VARCHAR(100) NOT NULL,
   dnum INT PRIMARY KEY CHECK (dnum < 10000), -- less than 4 digits
   mgrssn CHAR(9) NOT NULL,
   dlocation VARCHAR(100) NOT NULL
);

-- EMPLOYEE
CREATE TABLE EMPLOYEE (
   fname VARCHAR(50) NOT NULL,
   lname VARCHAR(50) NOT NULL,
   ssn CHAR(9) PRIMARY KEY,
   sex CHAR(1) NOT NULL,
   salary DECIMAL(10,2) NOT NULL,
   joindate DATE NOT NULL,
   superssn CHAR(9),
   dno INT NOT NULL,
   FOREIGN KEY (dno) REFERENCES DEPT(dnum)
);

-- PROJECT
CREATE TABLE PROJECT (
   pname VARCHAR(100) NOT NULL,
   pno INT PRIMARY KEY,
   plocation VARCHAR(100) NOT NULL,
   dnum INT NOT NULL,
   FOREIGN KEY (dnum) REFERENCES DEPT(dnum)
);

-- WORK_ON
CREATE TABLE WORK_ON (
   ssn CHAR(9) NOT NULL,
   pno INT NOT NULL,
   hours DECIMAL(5,2) NOT NULL,
   FOREIGN KEY (ssn) REFERENCES EMPLOYEE(ssn),
   FOREIGN KEY (pno) REFERENCES PROJECT(pno)
);

-- DEPT
INSERT INTO DEPT VALUES
('HR', 101, '111223333', 'Mumbai'),
('IT', 102, '222334444', 'Pune'),
```

```
('Sales', 103, '333445555', 'Jalgaon'),
('Finance', 104, '444556666', 'Nagpur');

-- EMPLOYEE
INSERT INTO EMPLOYEE VALUES
('John', 'Doe', '111223333', 'M', 80000, '2020-01-10', NULL, 101),
('Jane', 'Smith', '222334444', 'F', 95000, '2019-03-15', '111223333', 102),
('Alan', 'Walker', '333445555', 'M', 87000, '2018-05-20', '222334444', 103),
('Nina', 'Brown', '444556666', 'F', 90000, '2017-07-25', '333445555', 104),
('Raj', 'Kumar', '555667777', 'M', 75000, '2021-06-01', '111223333', 101),
('Meera', 'Patel', '666778888', 'F', 72000, '2022-04-11', '222334444', 102),
('Vikas', 'Yadav', '777889999', 'M', 77000, '2020-10-30', '333445555', 103),
('Sneha', 'Sharma', '888990000', 'F', 76000, '2023-01-17', '444556666', 104),
('Arun', 'Mehta', '999001111', 'M', 73000, '2021-09-09', '222334444', 102),
('Priya', 'Joshi', '000112222', 'F', 70000, '2023-03-12', '111223333', 101);

-- PROJECT
INSERT INTO PROJECT VALUES
('Payroll System', 1, 'Pune', 102),
('Sales Tracking', 2, 'Jalgaon', 103),
('Inventory Management', 3, 'Nagpur', 104),
('Recruitment Portal', 4, 'Mumbai', 101),
('Customer Feedback', 5, 'Jalgaon', 103);

-- WORK_ON
INSERT INTO WORK_ON VALUES
('111223333', 1, 10),
('222334444', 1, 8),
('333445555', 2, 12),
('444556666', 3, 9),
('555667777', 4, 6),
('666778888', 5, 11),
('777889999', 2, 10),
('888990000', 5, 7),
('999001111', 1, 5),
('000112222', 5, 6);

--a) Projects in 'Jalgaon' with department and manager info
SELECT P.pno, P.dnum, E.lname AS manager_last_name
FROM PROJECT P
JOIN DEPT D ON P.dnum = D.dnum
JOIN EMPLOYEE E ON D.mgrssn = E.ssn
WHERE P.plocation = 'Jalgaon';

--b) Projects with more than 2 employees
SELECT P.pno, P.pname, COUNT(W.ssn) AS employee_count
FROM PROJECT P
JOIN WORK_ON W ON P.pno = W.pno
GROUP BY P.pno, P.pname
```

```
HAVING COUNT(W.ssn) > 2;

--c) View: Dept name, manager name & salary
CREATE VIEW vw_DeptManager AS
SELECT D.dname, E.fname + ' ' + E.lname AS manager_name, E.salary AS manager_salary
FROM DEPT D
JOIN EMPLOYEE E ON D.mgrssn = E.ssn;
select * from vw_DeptManager

--d) Create the Trigger
CREATE TRIGGER trg_ValidateSalary
ON EMPLOYEE
AFTER INSERT, UPDATE
AS
BEGIN
  IF EXISTS (
    SELECT 1
    FROM EMPLOYEE E
    JOIN DEPT D ON E.dno = D.dnum
    JOIN EMPLOYEE M ON D.mgrssn = M.ssn
    WHERE E.salary > M.salary
  )
  BEGIN
    RAISERROR ('Employee salary cannot exceed department manager salary.', 16, 1);
    ROLLBACK TRANSACTION;
  END
END;
--If the Salary is Valid (Lower than Manager's)

INSERT INTO EMPLOYEE VALUES ('Test2', 'LowSalary', '123456780', 'F', 30000, '2024-01-01', NULL, 101);

-- Get the trigger code
EXEC sp_helptext 'trg_ValidateSalary';
```

Page 19—

```
--a) Find the details of customers who have opened the accounts within the period 25-3-2006 to 28-3-2006.
--b) Find the details of customers who have joint accounts & balance is less than 2 lakhs.
--c) Write a trigger TRANSACTION on table to calculate the current balance of the account on which transaction is made. ( if
trans_type = c then bal = bal + amt else if trans_type = d then bal bal — amt)
--d) write a cursor on CUSTOMER table to fetch the last row.

create database bank
use bank

-- Create ACCOUNT table
CREATE TABLE ACCOUNT (
  accno INT PRIMARY KEY CHECK (accno < 100),
  open_date DATE NOT NULL,
  acctype CHAR(1) CHECK (acctype IN ('P', 'J')),
  balance DECIMAL(15,2) NOT NULL
```

```
);

-- Create TRANSACTION table
CREATE TABLE TRANSACTION1 (
   trans_id INT PRIMARY KEY,
   trans_date DATE NOT NULL,
   accno INT,
   trans_type CHAR(1) CHECK (trans_type IN ('C', 'D')),
   amount DECIMAL(15,2) NOT NULL,
   FOREIGN KEY (accno) REFERENCES ACCOUNT(accno)
);

-- Create CUSTOMER table
CREATE TABLE CUSTOMER (
   cust_id INT PRIMARY KEY,
   name VARCHAR(100) NOT NULL,
   address VARCHAR(255) NOT NULL,
   accno INT,
   FOREIGN KEY (accno) REFERENCES ACCOUNT(accno)
);

-- Insert sample data into ACCOUNT
INSERT INTO ACCOUNT (accno, open_date, acctype, balance) VALUES
(1, '2006-03-26', 'P', 150000),
(2, '2006-03-27', 'J', 180000),
(3, '2006-03-28', 'P', 250000),
(4, '2006-03-29', 'J', 120000),
(5, '2006-03-30', 'P', 300000),
(6, '2006-03-31', 'J', 50000),
(7, '2006-04-01', 'P', 200000),
(8, '2006-04-02', 'J', 220000),
(9, '2006-04-03', 'P', 350000),
(10, '2006-04-04', 'J', 40000);

-- Insert sample data into CUSTOMER
INSERT INTO CUSTOMER (cust_id, name, address, accno) VALUES
(101, 'Alice', 'Goa', 1),
(102, 'Bob', 'Mumbai', 2),
(103, 'Charlie', 'Delhi', 3),
(104, 'David', 'Chennai', 4),
(105, 'Eva', 'Bangalore', 5),
(106, 'Frank', 'Pune', 6),
(107, 'Grace', 'Kolkata', 7),
(108, 'Hank', 'Hyderabad', 8),
(109, 'Ivy', 'Ahmedabad', 9),
(110, 'Jack', 'Surat', 10);

-- Insert sample data into TRANSACTION
INSERT INTO TRANSACTION1 (trans_id, trans_date, accno, trans_type, amount) VALUES
```

```
(1, '2025-04-20', 1, 'C', 5000),
(2, '2025-04-21', 2, 'D', 10000),
(3, '2025-04-22', 3, 'C', 15000),
(4, '2025-04-23', 4, 'D', 2000),
(5, '2025-04-24', 5, 'C', 25000),
(6, '2025-04-25', 6, 'D', 5000),
(7, '2025-04-26', 7, 'C', 8000),
(8, '2025-04-27', 8, 'D', 3000),
(9, '2025-04-28', 9, 'C', 12000),
(10, '2025-04-29', 10, 'D', 7000);

--a) Customers who opened accounts between 25-03-2006 and 28-03-2006
SELECT * FROM CUSTOMER
WHERE accno IN (
  SELECT accno FROM ACCOUNT
  WHERE open_date BETWEEN '2006-03-25' AND '2006-03-28'
);

--b) Customers with joint accounts and balance less than 2 lakhs
SELECT * FROM CUSTOMER
WHERE accno IN (
  SELECT accno FROM ACCOUNT
  WHERE acctype = 'J' AND balance < 200000
);

--c)Trigger to Update Account Balance
CREATE TRIGGER update_balance
ON TRANSACTION1
AFTER INSERT
AS
BEGIN
  SET NOCOUNT ON;

  DECLARE @accno INT, @trans_type CHAR(1), @amount DECIMAL(15,2);

  -- Loop through each inserted row
  DECLARE inserted_cursor CURSOR FOR
  SELECT accno, trans_type, amount FROM inserted;

  OPEN inserted_cursor;
  FETCH NEXT FROM inserted_cursor INTO @accno, @trans_type, @amount;

  WHILE @@FETCH_STATUS = 0
  BEGIN
    IF @trans_type = 'C'
    BEGIN
      -- Credit: Increase balance
      UPDATE ACCOUNT
      SET balance = balance + @amount
```

```
        WHERE accno = @accno;
    END
    ELSE IF @trans_type = 'D'
    BEGIN
      -- Debit: Decrease balance
      UPDATE ACCOUNT
      SET balance = balance - @amount
      WHERE accno = @accno;
    END
    FETCH NEXT FROM inserted_cursor INTO @accno, @trans_type, @amount;
  END
  CLOSE inserted_cursor;
  DEALLOCATE inserted_cursor;
END;

--d)Cursor to Fetch the Last Row from CUSTOMER Table
DECLARE @cust_id INT, @name NVARCHAR(100), @address NVARCHAR(255), @accno INT;

-- Declare the cursor to select all columns from the CUSTOMER table
DECLARE last_customer_cursor CURSOR FOR
  SELECT cust_id, name, address, accno
  FROM CUSTOMER
  ORDER BY cust_id DESC;  -- Assuming cust_id is unique and can be used to determine the last row

-- Open the cursor
OPEN last_customer_cursor;

-- Fetch the first (which is the last due to DESC) row
FETCH NEXT FROM last_customer_cursor INTO @cust_id, @name, @address, @accno;

-- Check if the fetch was successful
IF @@FETCH_STATUS = 0
BEGIN
  PRINT 'Last Customer ID: ' + CAST(@cust_id AS NVARCHAR);
  PRINT 'Last Customer Name: ' + @name;
  PRINT 'Last Customer Address: ' + @address;
  PRINT 'Last Customer Account Number: ' + CAST(@accno AS NVARCHAR);
END
ELSE
BEGIN
  PRINT 'No rows found in CUSTOMER table.';
END

-- Close and deallocate the cursor
CLOSE last_customer_cursor;
DEALLOCATE last_customer_cursor;
```


--a) Find the manufacturers of color printers.

--b) Find the laptops whose speed is slower than that of any PC.
--c) Express the following constraint as SQL assertions - "No black & white printer should have price greater than color printers."
--d) write a trigger on PC & LAPTOP table such that the hard disk size should be greater than 20 GB

```
create database sales
use sales

-- Create PRODUCT table
CREATE TABLE PRODUCT (
   Maker VARCHAR(50) NOT NULL,
   Modelno INT PRIMARY KEY,
   Type VARCHAR(10) CHECK (Type IN ('PC', 'Laptop', 'Printer')) NOT NULL
);

-- Create PC table
CREATE TABLE PC (
   Modelno INT PRIMARY KEY,
   Speed DECIMAL(5,2) NOT NULL,
   RAM INT CHECK (RAM > 0) NOT NULL,
   HD INT CHECK (HD > 0) NOT NULL,
   CD VARCHAR(10) NOT NULL,
   Price DECIMAL(10,2) CHECK (Price > 0) NOT NULL,
   FOREIGN KEY (Modelno) REFERENCES PRODUCT(Modelno)
);

-- Create LAPTOP table
CREATE TABLE LAPTOP (
   Modelno INT PRIMARY KEY,
   Speed DECIMAL(5,2) NOT NULL,
   RAM INT CHECK (RAM > 0) NOT NULL,
   HD INT CHECK (HD > 0) NOT NULL,
   Screen DECIMAL(4,1) CHECK (Screen > 0) NOT NULL,
   Price DECIMAL(10,2) CHECK (Price > 0) NOT NULL,
   FOREIGN KEY (Modelno) REFERENCES PRODUCT(Modelno)
);

-- Create PRINTER table
CREATE TABLE PRINTER (
   Modelno INT PRIMARY KEY,
   Color CHAR(1) CHECK (Color IN ('T', 'F')) NOT NULL,
   Type VARCHAR(20) CHECK (Type IN ('Laser', 'Ink-jet', 'Dot-matrix', 'Dry')) NOT NULL,
   Price DECIMAL(10,2) CHECK (Price > 0) NOT NULL,
   FOREIGN KEY (Modelno) REFERENCES PRODUCT(Modelno)
);

-- Insert into PRODUCT
INSERT INTO PRODUCT (Maker, Modelno, Type) VALUES
('IBM', 101, 'PC'),
```

```
('Dell', 102, 'Laptop'),
('HP', 103, 'Printer'),
('Lenovo', 104, 'PC'),
('Acer', 105, 'Laptop'),
('Canon', 106, 'Printer'),
('Asus', 107, 'PC'),
('Samsung', 108, 'Laptop'),
('Epson', 109, 'Printer'),
('MSI', 110, 'PC');

-- Insert into PC
INSERT INTO PC (Modelno, Speed, RAM, HD, CD, Price) VALUES
(101, 3.5, 8, 500, 'DVD', 1200),
(104, 3.2, 16, 1000, 'Blu-ray', 1500),
(107, 3.0, 8, 512, 'DVD', 1100),
(110, 3.6, 32, 2000, 'Blu-ray', 2000);

-- Insert into LAPTOP
INSERT INTO LAPTOP (Modelno, Speed, RAM, HD, Screen, Price) VALUES
(102, 2.5, 8, 512, 15.6, 800),
(105, 2.8, 16, 1024, 17.3, 1200),
(108, 3.0, 8, 512, 14.0, 900);

-- Insert into PRINTER
INSERT INTO PRINTER (Modelno, Color, Type, Price) VALUES
(103, 'F', 'Laser', 300),
(106, 'T', 'Ink-jet', 150),
(109, 'T', 'Dot-matrix', 100);

--a) Find the manufacturers of color printers.
SELECT DISTINCT p.Maker
FROM PRODUCT p
JOIN PRINTER pr ON p.Modelno = pr.Modelno
WHERE pr.Color = 'T';

--b) Find the laptops whose speed is slower than that of any PC.
SELECT l.Modelno, l.Speed
FROM LAPTOP l
WHERE l.Speed < ALL (SELECT p.Speed FROM PC p);

--c)
-- SQL Server does not support assertions directly. Use a trigger instead.
CREATE TRIGGER CheckPrinterPrice
ON PRINTER
FOR INSERT, UPDATE
AS
BEGIN
  IF EXISTS (
    SELECT 1
```

```
    FROM inserted i
    JOIN PRINTER pr ON i.Modelno = pr.Modelno
    WHERE pr.Color = 'F' AND pr.Price > ALL (SELECT Price FROM PRINTER WHERE Color = 'T')
  )
  BEGIN
    RAISERROR('Black & white printer price cannot be greater than color printers.', 16, 1);
    ROLLBACK TRANSACTION;
  END
END;

--d)Write a trigger on PC & LAPTOP tables such that the hard disk size should be greater than 20 GB.
-- Trigger for PC table
CREATE TRIGGER CheckPCDiskSize
ON PC
FOR INSERT, UPDATE
AS
BEGIN
  -- Declare a variable to hold the HD value
  DECLARE @HD INT;

  -- Cursor to iterate over each inserted or updated row
  DECLARE hd_cursor CURSOR FOR
    SELECT HD FROM inserted;

  OPEN hd_cursor;
  FETCH NEXT FROM hd_cursor INTO @HD;

  -- Loop through each row
  WHILE @@FETCH_STATUS = 0
  BEGIN
    -- Check if the HD value is less than or equal to 20
    IF @HD <= 20
    BEGIN
      RAISERROR('PC hard disk size must be greater than 20 GB.', 16, 1);
      ROLLBACK TRANSACTION;
      CLOSE hd_cursor;
      DEALLOCATE hd_cursor;
      RETURN;
    END

    FETCH NEXT FROM hd_cursor INTO @HD;
  END

  CLOSE hd_cursor;
  DEALLOCATE hd_cursor;
END;

-- Trigger for LAPTOP table
CREATE TRIGGER CheckLaptopDiskSize
```

```
ON LAPTOP
FOR INSERT, UPDATE
AS
BEGIN
  -- Declare a variable to hold the HD value
  DECLARE @HD INT;

  -- Cursor to iterate over each inserted or updated row
  DECLARE hd_cursor CURSOR FOR
    SELECT HD FROM inserted;

  OPEN hd_cursor;
  FETCH NEXT FROM hd_cursor INTO @HD;

  -- Loop through each row
  WHILE @@FETCH_STATUS = 0
  BEGIN
    -- Check if the HD value is less than or equal to 20
    IF @HD <= 20
    BEGIN
      RAISERROR('Laptop hard disk size must be greater than 20 GB.', 16, 1);
      ROLLBACK TRANSACTION;
      CLOSE hd_cursor;
      DEALLOCATE hd_cursor;
      RETURN;
    END

    FETCH NEXT FROM hd_cursor INTO @HD;
  END

  CLOSE hd_cursor;
  DEALLOCATE hd_cursor;
END;
```
