VIETNAM NATIONAL UNIVERSITY, HO CHI MINH CITY
UNIVERSITY OF TECHNOLOGY
FACULTY OF COMPUTER SCIENCE AND ENGINEERING

DATABASE SYSTEM (CO2014)

Lab Assignment

Lab 5

Instructor(s): Nguyễn Minh Tâm

Students: Phạm Trần Minh Trí - 2313622 *(Group TN01)*
Lê Nguyễn Kim Khôi - 2311671 *(Group TN01)*

HO CHI MINH CITY, DECEMBER 2025

```
CREATE DATABASE COMPANY;
GO
USE COMPANY;
```

1 Exercise 1

First, we use the script from database lab 4 to create and populate the database.

```
USE master;
CREATE DATABASE COMPANY;
GO
USE COMPANY;


CREATE TABLE EMPLOYEE (
    FNAME           VARCHAR(15)     NOT NULL,
    MINIT           CHAR,
    LNAME           VARCHAR(15)     NOT NULL,
    SSN             CHAR(9),
    BDATE           DATE,
    ADDRESS         VARCHAR(50),
    SEX             CHAR,
    SALARY          DECIMAL(10,2),
    SUPER_SSN       CHAR(9),
    DNO             INT,
    PRIMARY KEY (SSN),
    FOREIGN KEY (SUPER_SSN) REFERENCES EMPLOYEE (SSN)
);

CREATE TABLE DEPARTMENT (
    DNAME           VARCHAR(15)     NOT NULL UNIQUE,
    DNUMBER         INT,
    MGR_SSN         CHAR(9)         NOT NULL,
    MGR_START_DATE  DATE,
    PRIMARY KEY (DNUMBER)
);

CREATE TABLE DEPT_LOCATIONS (
    DNUMBER         INT,
    DLOCATION       VARCHAR(15),
    PRIMARY KEY (DNUMBER, DLOCATION),
    FOREIGN KEY (DNUMBER) REFERENCES DEPARTMENT (DNUMBER)
);

CREATE TABLE PROJECT (
    PNAME           VARCHAR(15)     NOT NULL,
    PNUMBER         INT,
    PLOCATION       VARCHAR(15)     NOT NULL,
    DNUM            INT,
    PRIMARY KEY (PNUMBER),
    FOREIGN KEY (DNUM) REFERENCES DEPARTMENT (DNUMBER)
);

CREATE TABLE WORKS_ON (
    ESSN            CHAR(9),
    PNO             INT,
    HOURS           DECIMAL(3,1),
    PRIMARY KEY (ESSN, PNO),
    FOREIGN KEY (ESSN) REFERENCES EMPLOYEE (SSN),
    FOREIGN KEY (PNO) REFERENCES PROJECT (PNUMBER)
```

```
);

CREATE TABLE DEPENDENT (
    ESSN            CHAR(9),
    DEPENDENT_NAME  VARCHAR(15),
    SEX             CHAR,
    BDATE           DATE,
    RELATIONSHIP    VARCHAR(15),
    PRIMARY KEY (ESSN, DEPENDENT_NAME),
    FOREIGN KEY (ESSN) REFERENCES EMPLOYEE (SSN)
);

ALTER TABLE EMPLOYEE
ADD FOREIGN KEY (DNO) REFERENCES DEPARTMENT (DNUMBER);

ALTER TABLE DEPARTMENT
ADD FOREIGN KEY (MGR_SSN) REFERENCES EMPLOYEE (SSN);
```

Listing 1: Create database

```
USE COMPANY;


INSERT INTO EMPLOYEE VALUES
    ('John', 'B', 'Smith', '123456789', '1965-01-09', '731 Fondren, Houston, TX',
    'M', 30000, NULL, NULL),
    ('Franklin', 'T', 'Wong', '333445555', '1955-12-08', '638 Voss, Houston, TX',
    'M', 40000, NULL, NULL),
    ('Alicia', 'J', 'Zelaya', '999887777', '1968-07-19', '3321 Castle, Spring, TX'
    , 'F', 25000, NULL, NULL),
    ('Jennifer', 'S', 'Wallace', '987654321', '1941-06-20', '291 Berry, Bellaire,
    TX', 'F', 43000, NULL, NULL),
    ('Ramesh', 'K', 'Narayan', '666884444', '1962-09-15', '975 Fire Oak, Humble,
    TX', 'M', 38000, NULL, NULL),
    ('Joyce', 'A', 'English', '453453453', '1972-07-31', '5631 Rice, Houston, TX',
     'F', 25000, NULL, NULL),
    ('Ahmad', 'V', 'Jabbar', '987987987', '1969-03-29', '980 Dallas, Houston, TX',
     'M', 25000, NULL, NULL),
    ('James', 'E', 'Borg', '888665555', '1937-11-10', '450 Stone, Houston, TX', 'M
    ', 55000, NULL, NULL);

INSERT INTO DEPARTMENT VALUES
    ('Research', 5, 333445555, '1988-05-22'),
    ('Administration', 4, 987654321, '1995-01-01'),
    ('Headquarters', 1, 888665555, '1981-06-19');


UPDATE EMPLOYEE SET SUPER_SSN = '333445555', DNO = 5 WHERE SSN = '123456789';
UPDATE EMPLOYEE SET SUPER_SSN = '888665555', DNO = 5 WHERE SSN = '333445555';
UPDATE EMPLOYEE SET SUPER_SSN = '987654321', DNO = 4 WHERE SSN = '999887777';
UPDATE EMPLOYEE SET SUPER_SSN = '888665555', DNO = 4 WHERE SSN = '987654321';
UPDATE EMPLOYEE SET SUPER_SSN = '333445555', DNO = 5 WHERE SSN = '666884444';
UPDATE EMPLOYEE SET SUPER_SSN = '333445555', DNO = 5 WHERE SSN = '453453453';
UPDATE EMPLOYEE SET SUPER_SSN = '987654321', DNO = 4 WHERE SSN = '987987987';
UPDATE EMPLOYEE SET SUPER_SSN = NULL, DNO = 1 WHERE SSN = '888665555';

INSERT INTO DEPT_LOCATIONS VALUES
    (1, 'Houston'),
    (4, 'Stafford'),
    (5, 'Bellaire'),
    (5, 'Sugarland'),
    (5, 'Houston');
```

```
INSERT INTO PROJECT VALUES
    ('ProductX', 1, 'Bellaire', 5),
    ('ProductY', 2, 'Sugarland', 5),
    ('ProductZ', 3, 'Houston', 5),
    ('Computerization', 10, 'Stafford', 4),
    ('Reorganization', 20, 'Houston', 1),
    ('Newbenefits', 30, 'Stafford', 4);

INSERT INTO WORKS_ON VALUES
    ('123456789', 1, 32.5),
    ('123456789', 2, 7.5),
    ('666884444', 3, 40.0),
    ('453453453', 1, 20.0),
    ('453453453', 2, 20.0),
    ('333445555', 2, 10.0),
    ('333445555', 3, 10.0),
    ('333445555', 10, 10.0),
    ('333445555', 20, 10.0),
    ('999887777', 30, 30.0),
    ('999887777', 10, 10.0),
    ('987987987', 10, 35.0),
    ('987987987', 30, 5.0),
    ('987654321', 30, 20.0),
    ('987654321', 20, 15.0),
    ('888665555', 20, NULL);

INSERT INTO DEPENDENT VALUES
    ('333445555', 'Alice', 'F', '1986-04-05', 'DAUGHTER'),
    ('333445555', 'Theodore', 'M', '1983-10-25', 'SON'),
    ('333445555', 'Joy', 'F', '1958-05-03', 'SPOUSE'),
    ('987654321', 'Abner', 'M', '1942-02-28', 'SPOUSE'),
    ('123456789', 'Michael', 'M', '1988-01-04', 'SON'),
    ('123456789', 'Alice', 'F', '1988-12-30', 'DAUGHTER'),
    ('123456789', 'Elizabeth', 'F', '1967-05-05', 'SPOUSE');
```

Listing 2: Populate database

1.1 VIEW

```
USE COMPANY;
GO

-- a. A view that has the department name, manager name, and manager salary for
    every department.
CREATE VIEW VIEW_A
AS
    SELECT DNAME, FNAME, MINIT, LNAME, SALARY
    FROM DEPARTMENT JOIN EMPLOYEE ON MGR_SSN = SSN;
GO

-- b. A view that has the employee name, supervisor name, and employee salary for
    each employee who works in the 'Research' department.
CREATE VIEW VIEW_B
AS
    SELECT
        E1.FNAME AS EMPLOYEE_FNAME,
        E1.MINIT AS EMPLOYEE_MINIT,
        E1.LNAME AS EMPLOYEE_LNAME,
        E2.FNAME AS SUPER_FNAME,
```

```
        E2.MINIT AS SUPER_MINIT,
        E2.LNAME AS SUPER_LNAME,
        E1.SALARY
    FROM (EMPLOYEE E1 JOIN EMPLOYEE E2 ON E1.SUPER_SSN = E2.SSN) JOIN DEPARTMENT
    D1 ON E1.DNO = D1.DNUMBER
    WHERE D1.DNAME = 'Research';
GO

-- c. A view that has the project name, controlling department name, number of
    employees, and total hours worked per week on the project for each project.
CREATE VIEW VIEW_C
AS
    SELECT
        PNAME,
        DNAME, COUNT(ESSN) AS NUM_EMPLOYEE,
        SUM([HOURS]) AS TOTAL_HOUR
    FROM
        (PROJECT JOIN DEPARTMENT ON DNUM = DNUMBER) JOIN WORKS_ON ON PNUMBER = PNO
    GROUP BY PNAME, DNAME;
GO

-- d. A view that has the project name, controlling department name, number of
    employees, and total hours worked per week on the project for each project
    with more than two employees working on it.
CREATE VIEW VIEW_D
AS
    SELECT
        PNAME,
        DNAME, COUNT(ESSN) AS NUM_EMPLOYEE,
        SUM([HOURS]) AS TOTAL_HOUR
    FROM
        (PROJECT JOIN DEPARTMENT ON DNUM = DNUMBER) JOIN WORKS_ON ON PNUMBER = PNO
    GROUP BY PNAME, DNAME
    HAVING COUNT(ESSN) > 2;
GO

-- e. A view (SSN, Full Name of employee, Number of dependents) that includes
    information about employees who have the number of dependents greater than 2.
CREATE VIEW VIEW_E
AS
    SELECT
        SSN,
        CONCAT(FNAME, ' ', MINIT, ' ', LNAME) AS FULL_NAME,
        COUNT(DEPENDENT_NAME) AS NUM_DEPENDENT
    FROM EMPLOYEE JOIN DEPENDENT ON SSN = ESSN
    GROUP BY SSN, FNAME, MINIT, LNAME
    HAVING COUNT(DEPENDENT_NAME) > 2;
GO

-- f. A view (Full Name of employee, date of birth, gender) for those employees
    who have their birthdate in July.
CREATE VIEW VIEW_F
AS
    SELECT
        CONCAT(FNAME, ' ', MINIT, ' ', LNAME) AS FULL_NAME,
        BDATE,
        SEX
    FROM EMPLOYEE
    WHERE MONTH(BDATE) = 7;
GO

-- g. A view (Name of dependent, SSN of employee, date of birth of dependent) that
```

```
    includes information on all dependents who are less than 18 years old.
CREATE VIEW VIEW_G
AS
    SELECT DEPENDENT_NAME, ESSN, BDATE
    FROM DEPENDENT
    WHERE BDATE > DATEADD(YEAR, -18, GETDATE());
GO
```

Output results: See figure 1, 2, 3, 4, 5, 6, 7.

	DNAME	FNAME	MINIT	LNAME	SALARY
1	Headquarters	James	E	Borg	55000.00
2	Administration	Jennifer	S	Wallace	43000.00
3	Research	Franklin	T	Wong	40000.00

Figure 1: Result of VIEW_A

	EMPLOY...	EMPLOY...	EMPLOY...	SUPER_...	SUPER_...	SUPER_...	SALARY
1	John	B	Smith	Franklin	T	Wong	30000.00
2	Franklin	T	Wong	James	E	Borg	40000.00
3	Joyce	A	English	Franklin	T	Wong	25000.00
4	Ramesh	K	Narayan	Franklin	T	Wong	38000.00

Figure 2: Result of VIEW_B

	PNAME	DNAME	NUM_E...	TOTAL_...
1	Computerization	Administration	3	55.0
2	Newbenefits	Administration	3	55.0
3	Reorganization	Headquarters	3	25.0
4	ProductX	Research	2	52.5
5	ProductY	Research	3	37.5
6	ProductZ	Research	2	50.0

Figure 3: Result of VIEW_C

	PNAME	DNAME	NUM_E...	TOTAL_...
1	Computerization	Administration	3	55.0
2	Newbenefits	Administration	3	55.0
3	Reorganization	Headquarters	3	25.0
4	ProductY	Research	3	37.5

Figure 4: Result of VIEW_D

	SSN	FULL_NA...	NUM_D...
1	123456789	John B Smith	3
2	333445555	Franklin T Wong	3

Figure 5: Result of VIEW_E

	FULL_N...	BDATE	SEX
1	Joyce A English	1972-07-31	F
2	Alicia J Zelaya	1968-07-19	F

Figure 6: Result of VIEW_F

	DEPENDENT_...	ESSN	BDATE

Figure 7: Result of VIEW_G

1.2 TRIGGER, FUNCTION, STORE PROCEDURE, CURSOR

(a) Create a database trigger

```
USE COMPANY;
GO

-- 1. The supervisor of an employee must be older than the employee
CREATE TRIGGER TR_EMPLOYEE_SUPERVISOR_AGE
ON EMPLOYEE
FOR INSERT, UPDATE
AS
BEGIN
    IF EXISTS (
        SELECT *
        FROM inserted I
        JOIN EMPLOYEE S ON I.SUPER_SSN = S.SSN
        WHERE I.BDATE <= S.BDATE
    )
    BEGIN
        RAISERROR('Supervisor must be older than the employee.', 16, 1);
        ROLLBACK TRANSACTION;
    END
END;
GO

-- 2. The salary of an employee cannot be greater than the salary of his/her
    supervisor
CREATE TRIGGER TR_EMPLOYEE_SUPERVISOR_SALARY
ON EMPLOYEE
FOR INSERT, UPDATE
AS
BEGIN
    IF EXISTS (
        SELECT *
        FROM inserted I
        JOIN EMPLOYEE S ON I.SUPER_SSN = S.SSN
        WHERE I.SALARY > S.SALARY
    )
    BEGIN
        RAISERROR('Employee salary cannot be greater than supervisor salary.'
    , 16, 1);
        ROLLBACK TRANSACTION;
    END
END;
GO

-- 3. The salary of an employee can only increase
-- 4. When increasing salary, the amount must not be more than 20% of current
     salary
CREATE TRIGGER TR_EMPLOYEE_SALARY_INCREASE
ON EMPLOYEE
FOR UPDATE
AS
BEGIN
    -- Check if salary decreased
    IF EXISTS (
        SELECT *
        FROM inserted I
        JOIN deleted D ON I.SSN = D.SSN
        WHERE I.SALARY < D.SALARY
    )
```

```
    BEGIN
        RAISERROR('Employee salary can only increase, not decrease.', 16, 1);
        ROLLBACK TRANSACTION;
        RETURN;
    END

    -- Check if salary increase exceeds 20%
    IF EXISTS (
        SELECT *
        FROM inserted I
        JOIN deleted D ON I.SSN = D.SSN
        WHERE I.SALARY > D.SALARY * 1.20
    )
    BEGIN
        RAISERROR('Salary increase cannot exceed 20%% of current salary.',
    16, 1);
        ROLLBACK TRANSACTION;
    END
END;
GO

-- 5. An employee works on at most 4 projects
CREATE TRIGGER TR_WORKS_ON_PROJECT_LIMIT
ON WORKS_ON
FOR INSERT
AS
BEGIN
    IF EXISTS (
        SELECT ESSN
        FROM WORKS_ON
        WHERE ESSN IN (SELECT ESSN FROM inserted)
        GROUP BY ESSN
        HAVING COUNT(*) > 4
    )
    BEGIN
        RAISERROR('An employee can work on at most 4 projects.', 16, 1);
        ROLLBACK TRANSACTION;
    END
END;
GO

-- 6. The maximum number of hours an employee can work on all projects per
    week is 56
CREATE TRIGGER TR_WORKS_ON_WEEKLY_HOURS
ON WORKS_ON
FOR INSERT, UPDATE
AS
BEGIN
    IF EXISTS (
        SELECT ESSN
        FROM WORKS_ON
        WHERE ESSN IN (SELECT ESSN FROM inserted)
        GROUP BY ESSN
        HAVING SUM(HOURS) > 56
    )
    BEGIN
        RAISERROR('Total weekly hours cannot exceed 56 hours per employee.',
    16, 1);
        ROLLBACK TRANSACTION;
    END
END;
GO
```

```
-- 7. The location of a project must be one of the locations of its
    department
CREATE TRIGGER TR_PROJECT_LOCATION_VALIDATION
ON PROJECT
FOR INSERT, UPDATE
AS
BEGIN
    IF EXISTS (
        SELECT *
        FROM inserted I
        WHERE NOT EXISTS (
            SELECT *
            FROM DEPT_LOCATIONS DL
            WHERE DL.DNUMBER = I.DNUM
            AND DL.DLOCATION = I.PLOCATION
        )
    )
    BEGIN
        RAISERROR('Project location must be one of the department locations.'
    , 16, 1);
        ROLLBACK TRANSACTION;
    END
END;
GO

-- 8. The salary of a department manager must be higher than other employees
    in that department
CREATE TRIGGER TR_EMPLOYEE_MANAGER_SALARY
ON EMPLOYEE
FOR INSERT, UPDATE
AS
BEGIN
    -- Check if any non-manager employee has salary >= manager's salary
    IF EXISTS (
        SELECT *
        FROM inserted I
        JOIN DEPARTMENT D ON I.DNO = D.DNUMBER
        WHERE I.SSN != D.MGR_SSN
        AND I.SALARY >= (
            SELECT SALARY
            FROM EMPLOYEE
            WHERE SSN = D.MGR_SSN
        )
    )
    BEGIN
        RAISERROR('Department manager must have the highest salary in the
    department.', 16, 1);
        ROLLBACK TRANSACTION;
    END

    -- Check if a manager's salary is being set lower than any employee in
    their department
    IF EXISTS (
        SELECT *
        FROM inserted I
        JOIN DEPARTMENT D ON I.SSN = D.MGR_SSN
        WHERE EXISTS (
            SELECT *
            FROM EMPLOYEE E
            WHERE E.DNO = D.DNUMBER
            AND E.SSN != D.MGR_SSN
```

```
            AND E.SALARY >= I.SALARY
        )
    )
    BEGIN
        RAISERROR('Department manager must have the highest salary in the
    department.', 16, 1);
        ROLLBACK TRANSACTION;
    END
END;
GO

-- 9. Only department managers can work less than 5 hours on a project
CREATE TRIGGER TR_WORKS_ON_MINIMUM_HOURS
ON WORKS_ON
FOR INSERT, UPDATE
AS
BEGIN
    IF EXISTS (
        SELECT *
        FROM inserted I
        WHERE I.HOURS < 5
        AND NOT EXISTS (
            SELECT *
            FROM DEPARTMENT D
            WHERE D.MGR_SSN = I.ESSN
        )
    )
    BEGIN
        RAISERROR('Only department managers can work less than 5 hours on a
    project.', 16, 1);
        ROLLBACK TRANSACTION;
    END
END;
GO
```

(b) Alter table Department

```
USE COMPANY;
GO

-- Add Num_of_Emp column to Department table
ALTER TABLE DEPARTMENT
ADD Num_of_Emp INT;
GO

-- Create a trigger to automatically calculate Num_of_Emp when employees are
    inserted
CREATE TRIGGER trg_UpdateNumOfEmp_Insert
ON EMPLOYEE
AFTER INSERT
AS
BEGIN
    UPDATE DEPARTMENT
    SET Num_of_Emp = (
        SELECT COUNT(*)
        FROM EMPLOYEE
        WHERE DNO = DEPARTMENT.DNUMBER
    )
    WHERE DNUMBER IN (SELECT DISTINCT DNO FROM inserted);
END;
GO
```

```
-- Create a trigger to automatically calculate Num_of_Emp when employees are
    updated
CREATE TRIGGER trg_UpdateNumOfEmp_Update
ON EMPLOYEE
AFTER UPDATE
AS
BEGIN
    UPDATE DEPARTMENT
    SET Num_of_Emp = (
        SELECT COUNT(*)
        FROM EMPLOYEE
        WHERE DNO = DEPARTMENT.DNUMBER
    )
    WHERE DNUMBER IN (
        SELECT DISTINCT DNO FROM inserted
        UNION
        SELECT DISTINCT DNO FROM deleted
    );
END;
GO

-- Create a trigger to automatically calculate Num_of_Emp when employees are
    deleted
CREATE TRIGGER trg_UpdateNumOfEmp_Delete
ON EMPLOYEE
AFTER DELETE
AS
BEGIN
    UPDATE DEPARTMENT
    SET Num_of_Emp = (
        SELECT COUNT(*)
        FROM EMPLOYEE
        WHERE DNO = DEPARTMENT.DNUMBER
    )
    WHERE DNUMBER IN (SELECT DISTINCT DNO FROM deleted);
END;
GO

-- Initialize Num_of_Emp with current employee counts
UPDATE DEPARTMENT
SET Num_of_Emp = (
    SELECT COUNT(*)
    FROM EMPLOYEE
    WHERE DNO = DEPARTMENT.DNUMBER
);
GO
```

(c) Write a function

```
USE COMPANY;
GO

-- Function to return the total number of projects for a given employee ID
CREATE FUNCTION dbo.GetEmployeeProjectCount(@EmployeeSSN CHAR(9))
RETURNS INT
AS
BEGIN
    DECLARE @ProjectCount INT;

    SELECT @ProjectCount = COUNT(*)
    FROM WORKS_ON
    WHERE ESSN = @EmployeeSSN;
```

```
    RETURN @ProjectCount;
END;
GO

-- Example usage:
-- SELECT dbo.GetEmployeeProjectCount('123456789') AS TotalProjects;
```

(d) Create a stored procedure

```
USE COMPANY;
GO

-- Stored procedure to print SSN, Full name, Department name, and annual
    salary of all employees
CREATE PROCEDURE sp_GetEmployeeDetails
AS
BEGIN
    SELECT
        E.SSN,
        CONCAT(E.FNAME, ' ', E.MINIT, ' ', E.LNAME) AS FullName,
        D.DNAME AS DepartmentName,
        E.SALARY * 12 AS AnnualSalary
    FROM EMPLOYEE E
    INNER JOIN DEPARTMENT D ON E.DNO = D.DNUMBER
    ORDER BY E.SSN;
END;
GO
```

(e) Write a trigger

```
USE COMPANY;
GO

-- Create log table to track salary changes
CREATE TABLE SALARY_LOG (
    SSN CHAR(9),
    CONTENT VARCHAR(200),
    LOG_DATE DATE,
    FOREIGN KEY (SSN) REFERENCES EMPLOYEE(SSN)
);
GO

-- Create trigger to log salary changes when new salary > 50000
CREATE TRIGGER trg_LogSalaryChanges
ON EMPLOYEE
AFTER INSERT, UPDATE
AS
BEGIN
    -- Log for UPDATE operations
    IF EXISTS (SELECT * FROM inserted i INNER JOIN deleted d ON i.SSN = d.SSN
    )
    BEGIN
        INSERT INTO SALARY_LOG (SSN, CONTENT, LOG_DATE)
        SELECT
            i.SSN,
            'SALARY UPDATE FROM ' + CAST(d.SALARY AS VARCHAR(20)) + ' TO ' +
    CAST(i.SALARY AS VARCHAR(20)),
            CAST(GETDATE() AS DATE)
        FROM inserted i
        INNER JOIN deleted d ON i.SSN = d.SSN
```

```
        WHERE i.SALARY > 50000 AND i.SALARY != d.SALARY;
    END

    -- Log for INSERT operations
    ELSE
    BEGIN
        INSERT INTO SALARY_LOG (SSN, CONTENT, LOG_DATE)
        SELECT
            SSN,
            'SALARY INSERTED WITH VALUE ' + CAST(SALARY AS VARCHAR(20)),
            CAST(GETDATE() AS DATE)
        FROM inserted
        WHERE SALARY > 50000;
    END
END;
GO
```

(f) Write a stored procedure

```
USE COMPANY;
GO

-- Stored procedure to print salary levels for each employee
CREATE PROCEDURE PrintEmployeeSalaryLevels
AS
BEGIN
    DECLARE @SSN CHAR(9);
    DECLARE @FNAME VARCHAR(15);
    DECLARE @MINIT CHAR;
    DECLARE @LNAME VARCHAR(15);
    DECLARE @SALARY DECIMAL(10,2);
    DECLARE @LEVEL VARCHAR(10);

    -- Cursor to iterate through employees
    DECLARE employee_cursor CURSOR FOR
    SELECT SSN, FNAME, MINIT, LNAME, SALARY
    FROM EMPLOYEE
    ORDER BY SSN;

    OPEN employee_cursor;

    FETCH NEXT FROM employee_cursor INTO @SSN, @FNAME, @MINIT, @LNAME,
    @SALARY;

    WHILE @@FETCH_STATUS = 0
    BEGIN
        -- Determine salary level
        IF @SALARY < 20000
            SET @LEVEL = 'level C';
        ELSE IF @SALARY BETWEEN 20000 AND 50000
            SET @LEVEL = 'level B';
        ELSE
            SET @LEVEL = 'level A';

        -- Print the result
        PRINT @SSN + ', ' + @FNAME + ' ' + ISNULL(@MINIT, '') + ' ' + @LNAME
    + ', ' + @LEVEL;

        FETCH NEXT FROM employee_cursor INTO @SSN, @FNAME, @MINIT, @LNAME,
    @SALARY;
    END;
```

```
    CLOSE employee_cursor;
    DEALLOCATE employee_cursor;
END;
GO

-- Execute the stored procedure
EXEC PrintEmployeeSalaryLevels;
GO
```

2 Exercise 2

```
USE master;
GO

CREATE DATABASE HotelManagementDB;
GO


USE HotelManagementDB;
GO

CREATE TABLE Hotel(
    hotelNo VARCHAR(10) PRIMARY KEY,
    hotelName NVARCHAR(100),
    city NVARCHAR(50),
);
GO


CREATE TABLE Room(
    roomNo VARCHAR(10),
    hotelNo VARCHAR(10),
    type VARCHAR(20),
    price DECIMAL(18,2),
    NumAdultMax INT,
    PRIMARY KEY (roomNo, hotelNo),
    FOREIGN KEY(hotelNo) REFERENCES Hotel(hotelNo)
);
GO

CREATE TABLE Guest(
    guestNo VARCHAR(10) PRIMARY KEY,
    guestName NVARCHAR(100),
    guestAddress NVARCHAR(200),
    TotalAmount DECIMAL(18,2) DEFAULT 0
);
GO

CREATE TABLE Booking(
    hotelNo VARCHAR(10),
    dateFrom DATE,
    roomNo VARCHAR(10),
    guestNo VARCHAR(10),
    dateTo DATE,
    NumOfAdult INT,
    PRIMARY KEY (hotelNo, dateFrom, roomNo),
    FOREIGN KEY (roomNo, hotelNo) REFERENCES Room(roomNo, hotelNo),
    FOREIGN KEY (guestNo) REFERENCES Guest(guestNo)
);
GO
```

Listing 3: Create database

2.1 2a

```
ALTER TABLE Room
ADD CONSTRAINT CHK_DoubleRoomPrice
CHECK ((type = 'Double' AND price > 100) OR type <> 'Double')
GO
```

2.2 2b

```
CREATE TRIGGER trg_CheckDoublePriceVsSingle
on Room
after INSERT, UPDATE
AS
BEGIN
    IF exists (
        SELECT 1
        FROM inserted i
        JOIN Room r ON i.hotelNo = r.hotelNo
        WHERE i.type = 'Double'
        AND r.type = 'Single'
        AND i.price <= r.price
    )
    BEGIN
        RAISERROR("Error: Double Price greater than Single Price", 16, 1);
        ROLLBACK TRANSACTION;
    END
END;
GO
```

2.3 2c

```
CREATE TRIGGER trg_CheckGuessOverlap
ON Booking
after UPDATE, INSERT
AS
BEGIN
    IF EXISTS(
        SELECT 1
        FROM inserted i
        JOIN Booking b ON i.guestNo = b.guestNo
        WHERE
            NOT (i.hotelNo = b.hotelNo AND i.roomNo = b.roomNo AND i.dateFrom = b.
    dateFrom)
            AND (i.dateFrom < b.dateTo AND i.dateTo > b.dateFrom)
    )
    BEGIN
        RAISERROR('Error: Overlap time customers', 16, 1);
        ROLLBACK TRANSACTION;
    END
END;
Go
```

2.4 2d

```
CREATE TRIGGER trg_CheckMaxAdults
ON Booking
AFTER INSERT, UPDATE
AS
BEGIN
    IF EXISTS(
        SELECT 1
        from inserted i
        JOIN Room r on i.hotelNo = r.hotelNo and i.roomNo = r.roomNo
        WHERE i.NumOfAdult > r.NumAdultMax
    )
```

```
    BEGIN
        RAISERROR('Error: Num of Adult greater than max capacity', 16, 1);
        ROLLBACK TRANSACTION;
    END
END;
GO
```

2.5 2e

```
CREATE TRIGGER trg_UpdateGuestTotalAmount
ON Booking
after INSERT, UPDATE, DELETE
AS
BEGIN
    SET NOCOUNT ON;

    DECLARE @AffectedGuests Table (guestNo VARCHAR(10));

    INSERT into @AffectedGuests
    SELECT guestNo from inserted
    UNION
    SELECT guestNo from deleted;

    UPDATE g
    SET TotalAmount = ISNULL((
        SElect sum(r.price * DATEDIFF(DAY, b.dateFrom, b.dateTo))
        From Booking b
        Join Room r on b.hotelNo = r.hotelNo and b.roomNo = r.roomNo
        where b.guestNo = g.guestNo
    ), 0)
    From Guest g
    WHERE g.guestNo IN (SELECT guestNo from @AffectedGuests);
END;
GO
```

2.6 2f

```
CREATE TRIGGER trg_InsertLondonHotelRoom
ON LondonHotelRoom
INSTEAD OF INSERT
AS
BEGIN
    set NOCOUNT ON;

    insert into Hotel(hotelNo, hotelName, city)
    SELECT distinct i.hotelNo, i.hotelName, i.city
    From inserted i
    WHERE not exists (select 1 from Hotel h where h.hotelNo = i.hotelNo);


    INSERT into Room(roomNo, hotelNo, type, price, NumAdultMax)
    SELECT i.roomNo, i.hotelNo, i.type, i.price, NULL
    FROM inserted i;
END;
GO
```