**Step 1: Conceptual Diagram**

We will use the same entities as before:

**Department**

dept\_id (Primary Key)

dept\_name

**Employee**

emp\_id (Primary Key)

emp\_name

dept\_id (Foreign Key)

role\_id (Foreign Key)

**Role**

role\_id (Primary Key)

role\_name

**Permission**

perm\_id (Primary Key)

perm\_name

Role\_Permission (Junction table)

role\_id (Foreign Key)

perm\_id (Foreign Key)

**Relationships:**

Department ➔ Employee (One-to-Many)

Role ➔ Employee (One-to-Many)

Role ➔ Permission (Many-to-Many via Role\_Permission)

**Conceptual Diagram**

Department (dept\_id, dept\_name) ────< Employee (emp\_id, emp\_name, dept\_id, role\_id)

Role (role\_id, role\_name) ────< Employee (emp\_id, emp\_name, dept\_id, role\_id)

Role (role\_id, role\_name) ────< Role\_Permission (role\_id, perm\_id) >──── Permission (perm\_id, perm\_name)

**Step 2: SQL Commands to Create Tables**

**-- Create Department table**

CREATE TABLE Department (

dept\_id NUMBER PRIMARY KEY,

dept\_name VARCHAR2(100) NOT NULL

);

**-- Create Role table**

CREATE TABLE Role (

role\_id NUMBER PRIMARY KEY,

role\_name VARCHAR2(100) NOT NULL

);

**-- Create Permission table**

CREATE TABLE Permission (

perm\_id NUMBER PRIMARY KEY,

perm\_name VARCHAR2(100) NOT NULL

)

**-- Create Employee table**

CREATE TABLE Employee (

emp\_id NUMBER PRIMARY KEY,

emp\_name VARCHAR2(100) NOT NULL,

dept\_id NUMBER,

role\_id NUMBER,

CONSTRAINT fk\_dept FOREIGN KEY (dept\_id) REFERENCES Department(dept\_id) ON DELETE CASCADE,

CONSTRAINT fk\_role FOREIGN KEY (role\_id) REFERENCES Role(role\_id) ON DELETE CASCADE

);

**-- Create Role\_Permission junction table**

CREATE TABLE Role\_Permission (

role\_id NUMBER,

perm\_id NUMBER,

CONSTRAINT fk\_role\_perm\_role FOREIGN KEY (role\_id) REFERENCES Role(role\_id) ON DELETE CASCADE,

CONSTRAINT fk\_role\_perm\_perm FOREIGN KEY (perm\_id) REFERENCES Permission(perm\_id) ON DELETE CASCADE,

PRIMARY KEY (role\_id, perm\_id)

);

**Step 3: SQL Commands for Data Manipulation**

**Insert Data**

**-- Insert into Department**

INSERT INTO Department (dept\_id, dept\_name) VALUES (1, 'Human Resources');

INSERT INTO Department (dept\_id, dept\_name) VALUES (2, 'Engineering');

**-- Insert into Role**

INSERT INTO Role (role\_id, role\_name) VALUES (1, 'Manager');

INSERT INTO Role (role\_id, role\_name) VALUES (2, 'Developer');

**-- Insert into Permission**

INSERT INTO Permission (perm\_id, perm\_name) VALUES (1, 'View Reports');

INSERT INTO Permission (perm\_id, perm\_name) VALUES (2, 'Edit Reports');

**-- Insert into Employee**

INSERT INTO Employee (emp\_id, emp\_name, dept\_id, role\_id) VALUES (1, 'Alice', 1, 1);

INSERT INTO Employee (emp\_id, emp\_name, dept\_id, role\_id) VALUES (2, 'Bob', 2, 2);

INSERT INTO Employee (emp\_id, emp\_name, dept\_id, role\_id) VALUES (3, 'Charlie', 1, 2);

**Update Data::**

**-- Update Employee name**

UPDATE Employee SET emp\_name = 'Alice Smith' WHERE emp\_id = 1;

**-- Update Role name**

UPDATE Role SET role\_name = 'Senior Developer' WHERE role\_id = 2;

Delete Data::

**-- Delete an employee**

DELETE FROM Employee WHERE emp\_id = 3;

**-- Delete a department (this will also delete employees due to ON DELETE CASCADE)**

DELETE FROM Department WHERE dept\_id = 1;

Step 4: SQL Queries (SELECT, JOINs, and Subqueries)

**Basic SELECT::**

**-- Select all employees**

SELECT \* FROM Employee;

**-- Select all departments**

SELECT \* FROM Department;

**Joins::**

**-- Inner Join: Get employees with their respective department names**

SELECT e.emp\_id, e.emp\_name, d.dept\_name FROM Employee e

INNER JOIN Department d ON e.dept\_id = d.dept\_id;

**-- Left Join: Get all employees and their roles (if any)**

SELECT e.emp\_id, e.emp\_name, r.role\_name

FROM Employee e

LEFT JOIN Role r ON e.role\_id = r.role\_id;

**Subqueries::**

**-- Find employees who work in departments with 'Engineering'**

SELECT emp\_name

FROM Employee

WHERE dept\_id IN (SELECT dept\_id FROM Department WHERE dept\_name = 'Engineering');

**-- Find roles that have more than one permission**

SELECT r.role\_name

FROM Role r

WHERE r.role\_id IN (SELECT rp.role\_id FROM Role\_Permission rp GROUP BY rp.role\_id HAVING COUNT(rp.perm\_id) > 1);

**Step 5: DDL, DML, DCL, and TCL Operations**

**DDL** (Data Definition Language): Used for creating tables (CREATE TABLE).

**DML** (Data Manipulation Language): Used for inserting (INSERT), updating (UPDATE), and deleting (DELETE) data.

**DCL** (Data Control Language): Used for granting/revoking permissions (e.g., GRANT, REVOKE).

**TCL** (Transaction Control Language): Used for controlling transactions (e.g., COMMIT, ROLLBACK).

**DCL Example:::**

**-- Grant select permission on Employee table to a user**

GRANT SELECT ON Employee TO some\_user;

**-- Revoke select permission from the user**

REVOKE SELECT ON Employee FROM some\_user;

**TCL Example::**

**-- Commit changes**

**COMMIT;**

**-- Rollback changes**

ROLLBACK;