```
-- ========================================
-- 🔹 STEP 1: DROP & CREATE DATABASE
-- ========================================
DROP DATABASE IF EXISTS collabconnect;
CREATE DATABASE collabconnect;
USE collabconnect;

-- ========================================
-- 🔹 STEP 2: TABLE CREATION (Normalized)
-- ========================================

CREATE TABLE Users (
    user_id INT AUTO_INCREMENT PRIMARY KEY,
    first_name VARCHAR(50) NOT NULL,
    middle_name VARCHAR(50),
    last_name VARCHAR(50),
    email VARCHAR(100) UNIQUE NOT NULL,
    created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP
);

CREATE TABLE Projects (
    project_id INT AUTO_INCREMENT PRIMARY KEY,
    project_name VARCHAR(100) NOT NULL,
    user_id INT,
    FOREIGN KEY (user_id) REFERENCES Users(user_id)
    ON DELETE CASCADE
);

CREATE TABLE Skills (
    skill_id INT AUTO_INCREMENT PRIMARY KEY,
    skill_name VARCHAR(100) NOT NULL
);

CREATE TABLE UserSkills (
    user_id INT,
    skill_id INT,
    PRIMARY KEY (user_id, skill_id),
    FOREIGN KEY (user_id) REFERENCES Users(user_id)
        ON DELETE CASCADE,
    FOREIGN KEY (skill_id) REFERENCES Skills(skill_id)
        ON DELETE CASCADE
);

CREATE TABLE Teams (
    team_id INT AUTO_INCREMENT PRIMARY KEY,
    team_name VARCHAR(100) NOT NULL,
    project_id INT,
    FOREIGN KEY (project_id) REFERENCES Projects(project_id)
        ON DELETE CASCADE
);

CREATE TABLE TeamMembers (
    team_id INT,
    user_id INT,
    role VARCHAR(50),
    PRIMARY KEY (team_id, user_id),
    FOREIGN KEY (team_id) REFERENCES Teams(team_id)
        ON DELETE CASCADE,
    FOREIGN KEY (user_id) REFERENCES Users(user_id)
        ON DELETE CASCADE
);

CREATE TABLE Requests (
    request_id INT AUTO_INCREMENT PRIMARY KEY,
    sender_id INT,
    receiver_id INT,
    status VARCHAR(50),
    created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
    FOREIGN KEY (sender_id) REFERENCES Users(user_id)
        ON DELETE CASCADE,
    FOREIGN KEY (receiver_id) REFERENCES Users(user_id)
        ON DELETE CASCADE
);

-- ========================================
-- 🔹 STEP 3: INSERT SAMPLE DATA
-- ========================================

INSERT INTO Users (first_name, middle_name, last_name, email) VALUES
('Amit', 'Kumar', 'Sharma', 'amit@gmail.com'),
('Neha', NULL, 'Patel', 'neha@gmail.com'),
('Rohan', NULL, 'Mehta', 'rohan@gmail.com'),
('Sneha', NULL, 'Rai', 'sneha@gmail.com'),
('Anjali', NULL, 'Verma', 'anjali@gmail.com');

INSERT INTO Projects (project_name, user_id) VALUES
('AI Chatbot', 1),
('Web Portal', 2),
('Cyber Security Tool', 1);

INSERT INTO Skills (skill_name) VALUES
('Python'), ('Java'), ('MySQL'), ('Data Analysis');

INSERT INTO UserSkills VALUES
(1, 1), (1, 3), (2, 2), (3, 1), (4, 4);

INSERT INTO Teams (team_name, project_id) VALUES
('Team Alpha', 1),
('Team Beta', 2);

INSERT INTO TeamMembers VALUES
(1, 1, 'Leader'),
(1, 2, 'Developer'),
(2, 3, 'Tester'),
(2, 4, 'Designer');

INSERT INTO Requests (sender_id, receiver_id, status) VALUES
(1, 2, 'Pending'),
(2, 1, 'Accepted'),
(3, 4, 'Pending');

-- ========================================
-- 🔹 STEP 4: CRUD OPERATIONS
-- ========================================

-- CREATE
INSERT INTO Users (first_name, last_name, email)
VALUES ('Raj', 'Singh', 'raj@gmail.com');

-- READ
SELECT * FROM Users;

-- UPDATE
UPDATE Users SET email = 'neha123@gmail.com' WHERE user_id = 2;

-- DELETE
DELETE FROM Users WHERE user_id = 5;
```

```
-- ========================================
-- 🔷 STEP 5: DELETE CONSTRAINT DEMO
-- ========================================

-- Try deleting user_id=1 (will also delete related data)
DELETE FROM Users WHERE user_id = 1;
-- ✅ Works due to ON DELETE CASCADE

-- If we try to delete from a parent table without CASCADE, MySQL will stop it
-- Example (simulated):
-- DELETE FROM Projects WHERE project_id = 2;
-- ❌ Error if foreign key without cascade → “Cannot delete or update parent
row”
-- ✅ Solution: Either add ON DELETE CASCADE or delete child rows first.

-- ========================================
-- 🔷 STEP 6: JOINS (ALL TYPES)
-- ========================================

-- INNER JOIN
SELECT U.first_name, P.project_name
FROM Users U
INNER JOIN Projects P ON U.user_id = P.user_id;

-- LEFT JOIN
SELECT U.first_name, P.project_name
FROM Users U
LEFT JOIN Projects P ON U.user_id = P.user_id;

-- RIGHT JOIN
SELECT U.first_name, P.project_name
FROM Users U
RIGHT JOIN Projects P ON U.user_id = P.user_id;

-- FULL JOIN (Using UNION)
SELECT U.first_name, P.project_name
FROM Users U
LEFT JOIN Projects P ON U.user_id = P.user_id
UNION
SELECT U.first_name, P.project_name
FROM Users U
RIGHT JOIN Projects P ON U.user_id = P.user_id;

-- CROSS JOIN
SELECT U.first_name, S.skill_name
FROM Users U
CROSS JOIN Skills S;

-- SELF JOIN (example if we had mentor-mentee)
SELECT A.first_name AS Sender, B.first_name AS Receiver
FROM Users A
JOIN Requests R ON A.user_id = R.sender_id
JOIN Users B ON R.receiver_id = B.user_id;

-- ========================================
-- 🔷 STEP 7: SQL FUNCTIONS
-- ========================================

-- DISTINCT
SELECT DISTINCT skill_name FROM Skills;

-- COUNT
SELECT COUNT(*) AS total_users FROM Users;
```

```
-- MIN, MAX, AVG
SELECT MIN(user_id) AS min_id, MAX(user_id) AS max_id, AVG(user_id) AS avg_id
FROM Users;

-- LENGTH (string length)
SELECT first_name, LENGTH(first_name) AS name_length FROM Users;

-- CONCAT (combine name)
SELECT CONCAT(first_name, ' ', last_name) AS Full_Name FROM Users;

-- UPPER / LOWER
SELECT UPPER(first_name) AS UpperName, LOWER(last_name) AS LowerName FROM Users;

-- BETWEEN (IDs between 2 and 4)
SELECT * FROM Users WHERE user_id BETWEEN 2 AND 4;

-- IN / NOT IN
SELECT * FROM Users WHERE user_id IN (2,3);
SELECT * FROM Users WHERE user_id NOT IN (2,3);

-- LIKE (pattern search)
SELECT * FROM Users WHERE first_name LIKE 'S%';

-- TRUNCATE (clear all data but keep table)
TRUNCATE TABLE Requests;

-- ========================================
-- 🔹 STEP 8: ALTER OPERATIONS
-- ========================================

ALTER TABLE Users ADD phone_no VARCHAR(15);
ALTER TABLE Users MODIFY phone_no BIGINT;
ALTER TABLE Users CHANGE phone_no mobile_no BIGINT;
ALTER TABLE Users DROP COLUMN mobile_no;

-- ========================================
-- 🔹 STEP 9: COMMENTS ON RELATIONSHIPS
-- ========================================
-- 1. Users ↔ Projects → 1-to-many (if user deleted, project deleted)
-- 2. Users ↔ Skills → many-to-many (UserSkills)
-- 3. Projects ↔ Teams → 1-to-many
-- 4. Teams ↔ TeamMembers → many-to-many
-- 5. Users ↔ Requests → sender and receiver both linked
-- 6. CASCADE ensures no orphan rows remain
-- 7. Without CASCADE, deletion will throw error “Cannot delete parent row”

SHOW TABLES;
```