CDAC MUMBAI

Batch August 2025

NAME: Priyanka Bhausaheb Thange

PRN: 250840320140

NAME: Suraj Rawat

PRN: 250840320210

SUBJECT: **Corporate Training Management System**

1) DRAW AN ER DIAGRAM IN *DRAW.IO* SHOWING ENTITIES, ATTRIBUTES, AND RELATIONSHIPS.

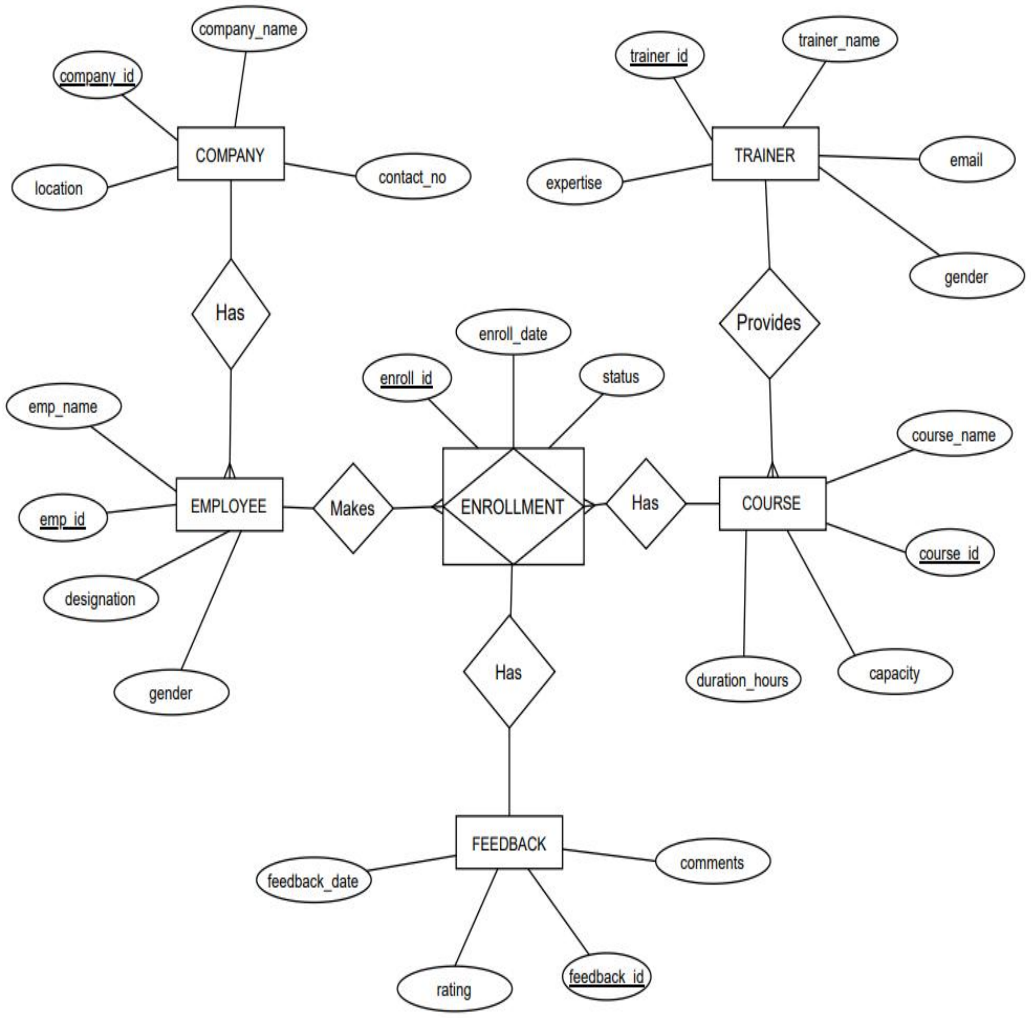


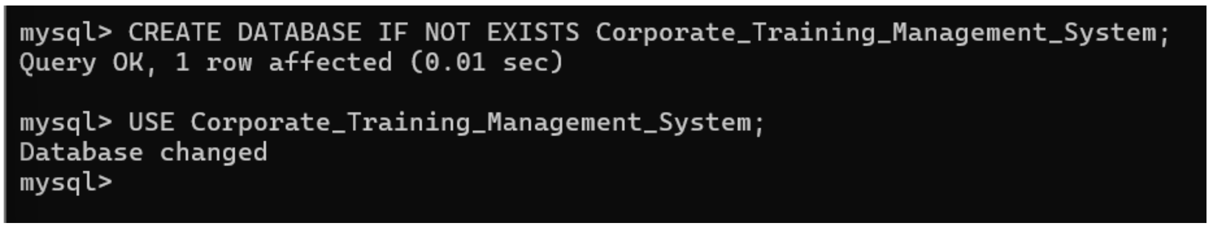
Fig. Corporate Training Management System

2) Create the database schema (DDL) with all required constraints and relationships.

■ Create Database

CREATE DATABASE IF NOT EXISTS Corporate_Training_Management_System;

USE Corporate_Training_Management_System;



```
mysql> CREATE DATABASE IF NOT EXISTS Corporate_Training_Management_System;
Query OK, 1 row affected (0.01 sec)

mysql> USE Corporate_Training_Management_System;
Database changed
mysql>
```

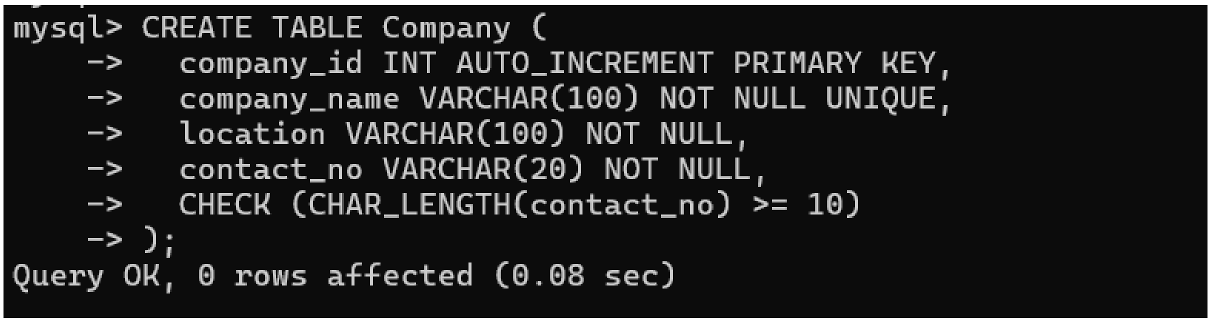
-- ==

■ COMPANY TABLE

-- Stores company information

-- ==

```
CREATE TABLE Company (
  company_id INT AUTO_INCREMENT PRIMARY KEY,
  company_name VARCHAR(100) NOT NULL UNIQUE,
  location VARCHAR(100) NOT NULL,
  contact_no VARCHAR(20) NOT NULL,
  CHECK (CHAR_LENGTH(contact_no) >= 10) ) ;
```



```
mysql> CREATE TABLE Company (
    ->   company_id INT AUTO_INCREMENT PRIMARY KEY,
    ->   company_name VARCHAR(100) NOT NULL UNIQUE,
    ->   location VARCHAR(100) NOT NULL,
    ->   contact_no VARCHAR(20) NOT NULL,
    ->   CHECK (CHAR_LENGTH(contact_no) >= 10)
    -> );
Query OK, 0 rows affected (0.08 sec)
```

-- ==

■ TRAINER TABLE

-- Stores trainer information

-- ==

```
CREATE TABLE Trainer (
  trainer_id INT AUTO_INCREMENT PRIMARY KEY,
```

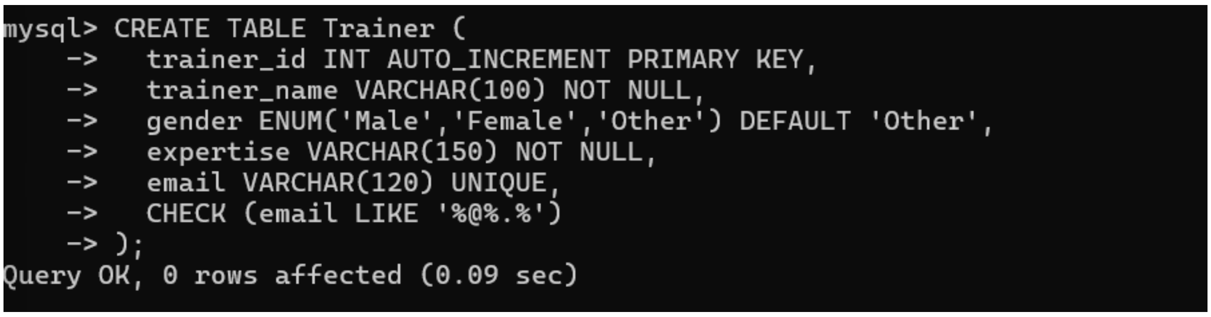
trainer_name VARCHAR(100) NOT NULL,

gender ENUM('Male','Female','Other') DEFAULT 'Other',

expertise VARCHAR(150) NOT NULL,

email VARCHAR(120) UNIQUE,

CHECK (email LIKE '%@%.%'));


```
mysql> CREATE TABLE Trainer (
    ->   trainer_id INT AUTO_INCREMENT PRIMARY KEY,
    ->   trainer_name VARCHAR(100) NOT NULL,
    ->   gender ENUM('Male','Female','Other') DEFAULT 'Other',
    ->   expertise VARCHAR(150) NOT NULL,
    ->   email VARCHAR(120) UNIQUE,
    ->   CHECK (email LIKE '%@%.%')
    -> );
Query OK, 0 rows affected (0.09 sec)
```

-- ==

■ COURSE TABLE

-- Stores course details handled by trainers

-- ==

CREATE TABLE Course (

course_id INT AUTO_INCREMENT PRIMARY KEY,

course_name VARCHAR(120) NOT NULL UNIQUE,

duration_hours INT NOT NULL CHECK (duration_hours > 0),

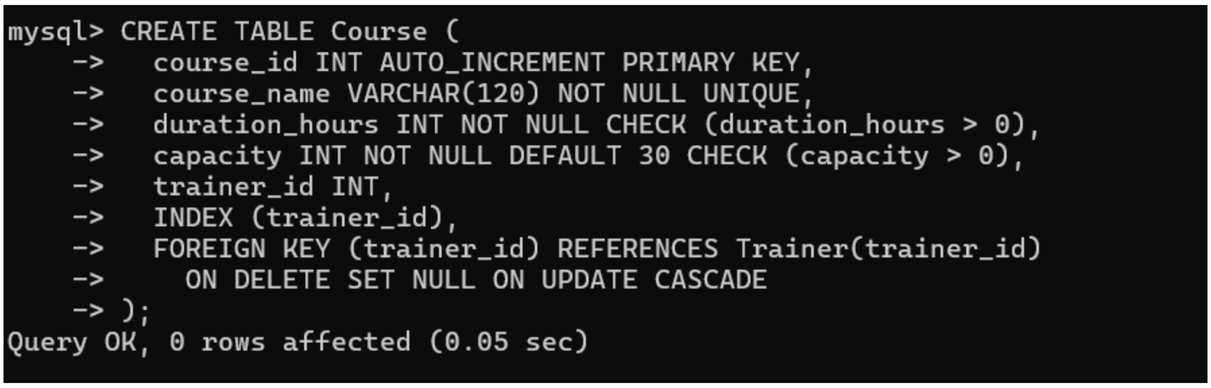
capacity INT NOT NULL DEFAULT 30 CHECK (capacity > 0),

trainer_id INT,

INDEX (trainer_id),

FOREIGN KEY (trainer_id) REFERENCES Trainer(trainer_id)

ON DELETE SET NULL ON UPDATE CASCADE);


```
mysql> CREATE TABLE Course (
    ->   course_id INT AUTO_INCREMENT PRIMARY KEY,
    ->   course_name VARCHAR(120) NOT NULL UNIQUE,
    ->   duration_hours INT NOT NULL CHECK (duration_hours > 0),
    ->   capacity INT NOT NULL DEFAULT 30 CHECK (capacity > 0),
    ->   trainer_id INT,
    ->   INDEX (trainer_id),
    ->   FOREIGN KEY (trainer_id) REFERENCES Trainer(trainer_id)
    ->     ON DELETE SET NULL ON UPDATE CASCADE
    -> );
Query OK, 0 rows affected (0.05 sec)
```

-- ==

■ EMPLOYEE TABLE

-- Stores employees who belong to a company-
===

CREATE TABLE Employee (

emp_id INT AUTO_INCREMENT PRIMARY KEY,

emp_name VARCHAR(100) NOT NULL,

designation VARCHAR(100),

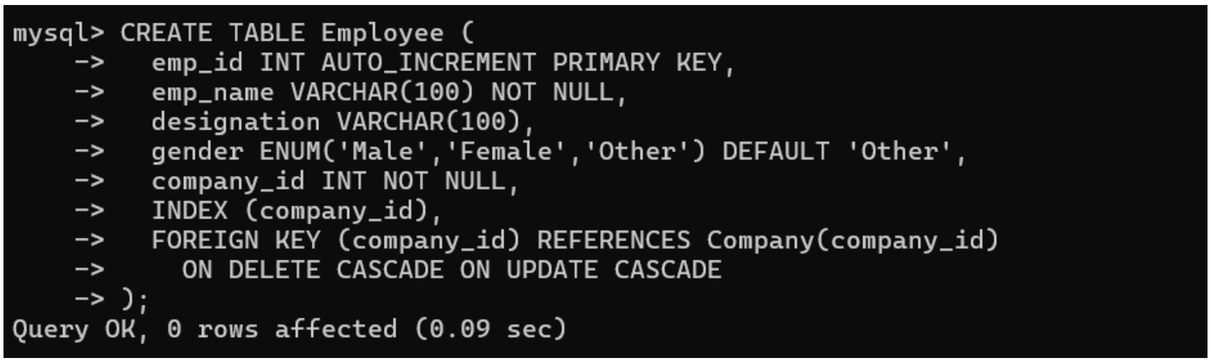
gender ENUM('Male','Female','Other') DEFAULT 'Other',

company_id INT NOT NULL,

INDEX (company_id),

FOREIGN KEY (company_id) REFERENCES Company(company_id)

ON DELETE CASCADE ON UPDATE CASCADE);



```
mysql> CREATE TABLE Employee (
    ->   emp_id INT AUTO_INCREMENT PRIMARY KEY,
    ->   emp_name VARCHAR(100) NOT NULL,
    ->   designation VARCHAR(100),
    ->   gender ENUM('Male','Female','Other') DEFAULT 'Other',
    ->   company_id INT NOT NULL,
    ->   INDEX (company_id),
    ->   FOREIGN KEY (company_id) REFERENCES Company(company_id)
    ->     ON DELETE CASCADE ON UPDATE CASCADE
    -> );
Query OK, 0 rows affected (0.09 sec)
```

-- ===

■ ENROLLMENT TABLE

-- Associates Employees with Courses (Many-to-Many relationship)

-- ===

CREATE TABLE Enrollment (

enroll_id INT AUTO_INCREMENT PRIMARY KEY,

emp_id INT NOT NULL,

course_id INT NOT NULL,

enroll_date DATE NOT NULL DEFAULT (CURRENT_DATE),

status ENUM('Registered','Completed','Cancelled') DEFAULT 'Registered',

UNIQUE (emp_id, course_id),

INDEX (emp_id),

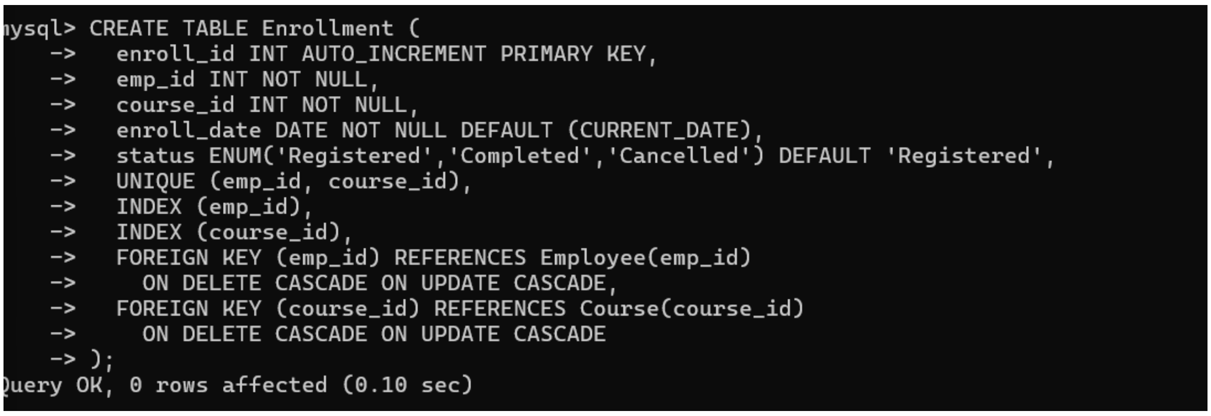
INDEX (course_id),

FOREIGN KEY (emp_id) REFERENCES Employee(emp_id)

  ON DELETE CASCADE ON UPDATE CASCADE,

FOREIGN KEY (course_id) REFERENCES Course(course_id)

  ON DELETE CASCADE ON UPDATE CASCADE);



```
ysql> CREATE TABLE Enrollment (
   ->   enroll_id INT AUTO_INCREMENT PRIMARY KEY,
   ->   emp_id INT NOT NULL,
   ->   course_id INT NOT NULL,
   ->   enroll_date DATE NOT NULL DEFAULT (CURRENT_DATE),
   ->   status ENUM('Registered','Completed','Cancelled') DEFAULT 'Registered',
   ->   UNIQUE (emp_id, course_id),
   ->   INDEX (emp_id),
   ->   INDEX (course_id),
   ->   FOREIGN KEY (emp_id) REFERENCES Employee(emp_id)
   ->     ON DELETE CASCADE ON UPDATE CASCADE,
   ->   FOREIGN KEY (course_id) REFERENCES Course(course_id)
   ->     ON DELETE CASCADE ON UPDATE CASCADE
   -> );
uery OK, 0 rows affected (0.10 sec)
```

-- ==

■ FEEDBACK TABLE

-- Stores feedback for each enrollment (one feedback per enrollment)

-- ==

CREATE TABLE Feedback (

feedback_id INT AUTO_INCREMENT PRIMARY KEY,

enroll_id INT NOT NULL UNIQUE,

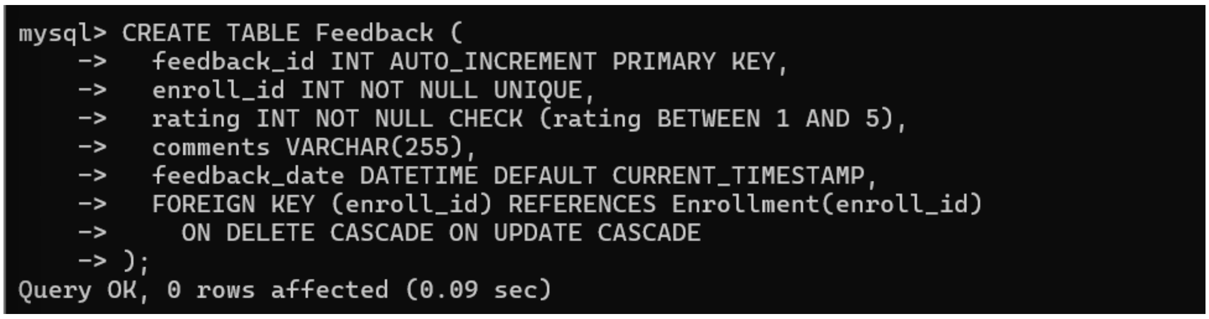
rating INT NOT NULL CHECK (rating BETWEEN 1 AND 5),

comments VARCHAR(255),

feedback_date DATETIME DEFAULT CURRENT_TIMESTAMP,

FOREIGN KEY (enroll_id) REFERENCES Enrollment(enroll_id)

  ON DELETE CASCADE ON UPDATE CASCADE);



```
mysql> CREATE TABLE Feedback (
    ->   feedback_id INT AUTO_INCREMENT PRIMARY KEY,
    ->   enroll_id INT NOT NULL UNIQUE,
    ->   rating INT NOT NULL CHECK (rating BETWEEN 1 AND 5),
    ->   comments VARCHAR(255),
    ->   feedback_date DATETIME DEFAULT CURRENT_TIMESTAMP,
    ->   FOREIGN KEY (enroll_id) REFERENCES Enrollment(enroll_id)
    ->     ON DELETE CASCADE ON UPDATE CASCADE
    -> );
Query OK, 0 rows affected (0.09 sec)
```

3) DML – INSERT / UPDATE / DELETE

----------- INSERT sample data ---------

➢ **Company Table**

mysql> INSERT INTO Company (company_name, location, contact_no) VALUES

-> ('TechSoft Pvt Ltd','Pune','9876543210'),

-> ('NextGen Solutions','Mumbai','9123456789'),

-> ('AlphaTech','Bangalore','9988776655'),

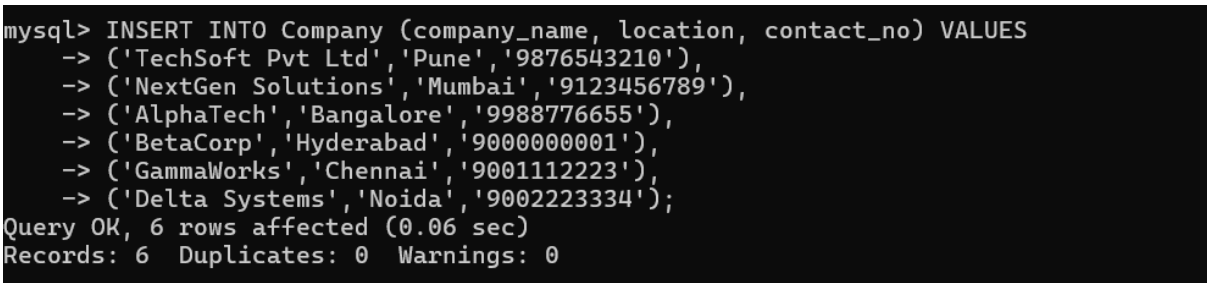
-> ('BetaCorp','Hyderabad','9000000001'),

-> ('GammaWorks','Chennai','9001112223'),

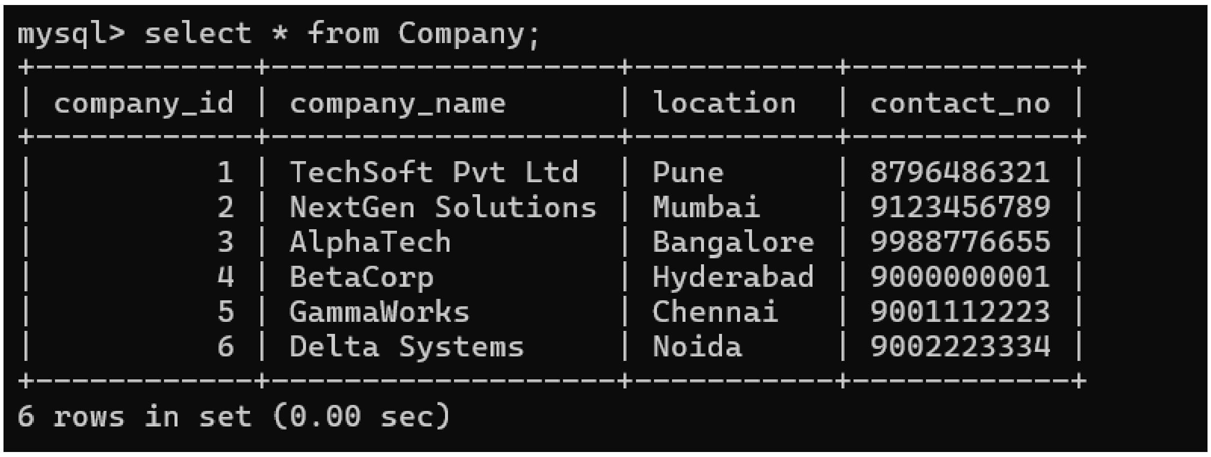
-> ('Delta Systems','Noida','9002223334');

Query OK, 6 rows affected (0.06 sec)

Records: 6 Duplicates: 0 Warnings: 0



```
mysql> INSERT INTO Company (company_name, location, contact_no) VALUES
    -> ('TechSoft Pvt Ltd','Pune','9876543210'),
    -> ('NextGen Solutions','Mumbai','9123456789'),
    -> ('AlphaTech','Bangalore','9988776655'),
    -> ('BetaCorp','Hyderabad','9000000001'),
    -> ('GammaWorks','Chennai','9001112223'),
    -> ('Delta Systems','Noida','9002223334');
Query OK, 6 rows affected (0.06 sec)
Records: 6  Duplicates: 0  Warnings: 0
```



```
mysql> select * from Company;
+------------+-------------------+-----------+------------+
| company_id | company_name      | location  | contact_no |
+------------+-------------------+-----------+------------+
|          1 | TechSoft Pvt Ltd  | Pune      | 8796486321 |
|          2 | NextGen Solutions | Mumbai    | 9123456789 |
|          3 | AlphaTech         | Bangalore | 9988776655 |
|          4 | BetaCorp          | Hyderabad | 9000000001 |
|          5 | GammaWorks        | Chennai   | 9001112223 |
|          6 | Delta Systems     | Noida     | 9002223334 |
+------------+-------------------+-----------+------------+
6 rows in set (0.00 sec)
```

==

➢ **Trainer Table**

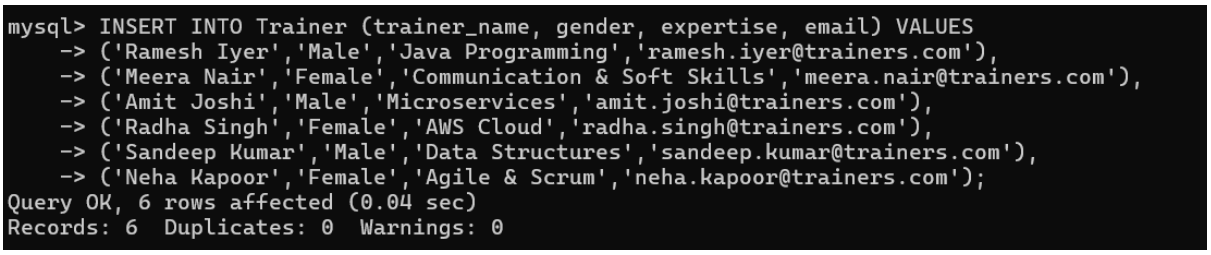
mysql> INSERT INTO Trainer

(trainer_name, gender, expertise, email) VALUES

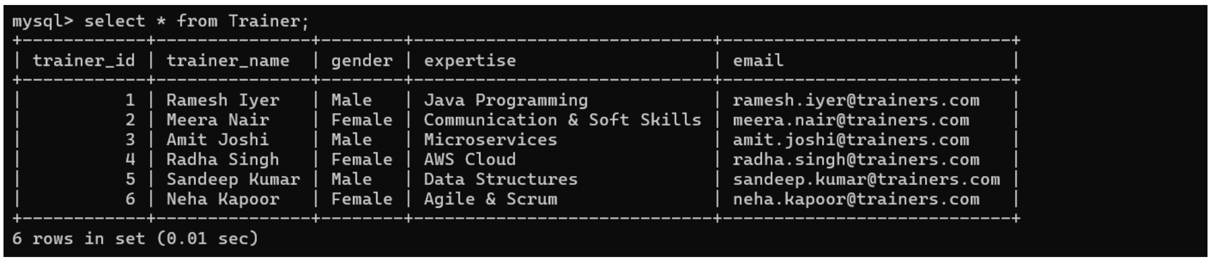
-> ('Ramesh Iyer','Male','Java Programming','ramesh.iyer@trainers.com'),

-> ('Meera Nair','Female','Communication & Soft Skills','meera.nair@trainers.com'),

```
    -> ('Amit Joshi','Male','Microservices','amit.joshi@trainers.com'),
    -> ('Radha Singh','Female','AWS Cloud','radha.singh@trainers.com'),
    -> ('Sandeep Kumar','Male','Data Structures','sandeep.kumar@trainers.com'),
    -> ('Neha Kapoor','Female','Agile & Scrum','neha.kapoor@trainers.com');
Query OK, 6 rows affected (0.04 sec)
Records: 6  Duplicates: 0  Warnings: 0
```



```
mysql> INSERT INTO Trainer (trainer_name, gender, expertise, email) VALUES
    -> ('Ramesh Iyer','Male','Java Programming','ramesh.iyer@trainers.com'),
    -> ('Meera Nair','Female','Communication & Soft Skills','meera.nair@trainers.com'),
    -> ('Amit Joshi','Male','Microservices','amit.joshi@trainers.com'),
    -> ('Radha Singh','Female','AWS Cloud','radha.singh@trainers.com'),
    -> ('Sandeep Kumar','Male','Data Structures','sandeep.kumar@trainers.com'),
    -> ('Neha Kapoor','Female','Agile & Scrum','neha.kapoor@trainers.com');
Query OK, 6 rows affected (0.04 sec)
Records: 6  Duplicates: 0  Warnings: 0
```

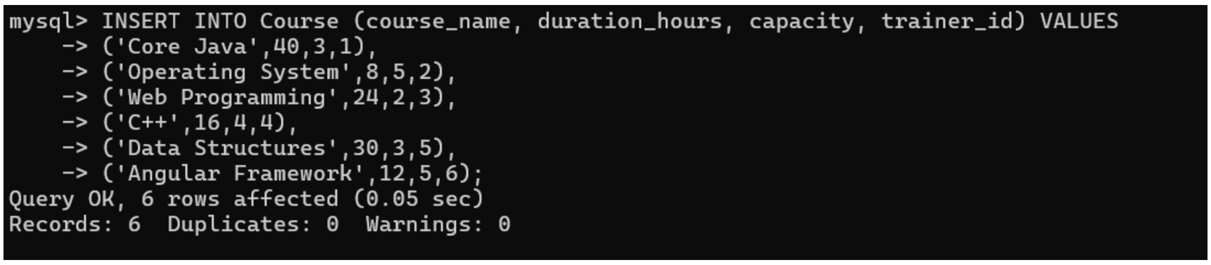


```
mysql> select * from Trainer;
+------------+---------------+--------+-----------------------------+----------------------------+
| trainer_id | trainer_name  | gender | expertise                   | email                      |
+------------+---------------+--------+-----------------------------+----------------------------+
|          1 | Ramesh Iyer   | Male   | Java Programming            | ramesh.iyer@trainers.com   |
|          2 | Meera Nair    | Female | Communication & Soft Skills | meera.nair@trainers.com    |
|          3 | Amit Joshi    | Male   | Microservices               | amit.joshi@trainers.com    |
|          4 | Radha Singh   | Female | AWS Cloud                   | radha.singh@trainers.com   |
|          5 | Sandeep Kumar | Male   | Data Structures             | sandeep.kumar@trainers.com |
|          6 | Neha Kapoor   | Female | Agile & Scrum               | neha.kapoor@trainers.com   |
+------------+---------------+--------+-----------------------------+----------------------------+
6 rows in set (0.01 sec)
```

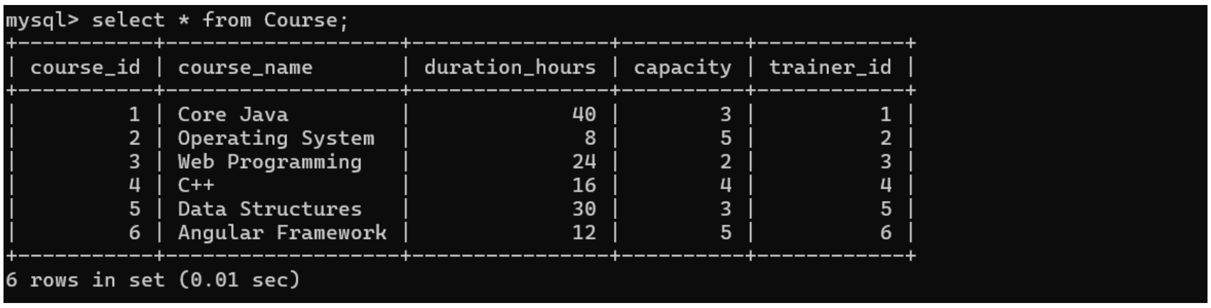
==

- **Course Table**

```
mysql> INSERT INTO Course (course_name, duration_hours, capacity, trainer_id) VALUES
    -> ('Core Java',40,3,1),
    -> ('Operating System',8,5,2),
    -> ('Web Programming',24,2,3),
    -> ('C++',16,4,4),
    -> ('Data Structures',30,3,5),
    -> ('Angular Framework',12,5,6);
Query OK, 6 rows affected (0.05 sec)
Records: 6  Duplicates: 0  Warnings: 0
```



```
mysql> INSERT INTO Course (course_name, duration_hours, capacity, trainer_id) VALUES
    -> ('Core Java',40,3,1),
    -> ('Operating System',8,5,2),
    -> ('Web Programming',24,2,3),
    -> ('C++',16,4,4),
    -> ('Data Structures',30,3,5),
    -> ('Angular Framework',12,5,6);
Query OK, 6 rows affected (0.05 sec)
Records: 6  Duplicates: 0  Warnings: 0
```



```
mysql> select * from Course;
+-----------+--------------------+----------------+----------+------------+
| course_id | course_name        | duration_hours | capacity | trainer_id |
+-----------+--------------------+----------------+----------+------------+
|         1 | Core Java          |             40 |        3 |          1 |
|         2 | Operating System   |              8 |        5 |          2 |
|         3 | Web Programming    |             24 |        2 |          3 |
|         4 | C++                |             16 |        4 |          4 |
|         5 | Data Structures    |             30 |        3 |          5 |
|         6 | Angular Framework  |             12 |        5 |          6 |
+-----------+--------------------+----------------+----------+------------+
6 rows in set (0.01 sec)
```

==

- **Employee Table**

mysql> INSERT INTO Employee (emp_name, designation, gender, company_id) VALUES

-> ('Pooja Patil','Software Engineer','Female',1),

-> ('Rohit Sharma','Team Lead','Male',1),

-> ('Ram Sharma','HR Executive','Female',2),

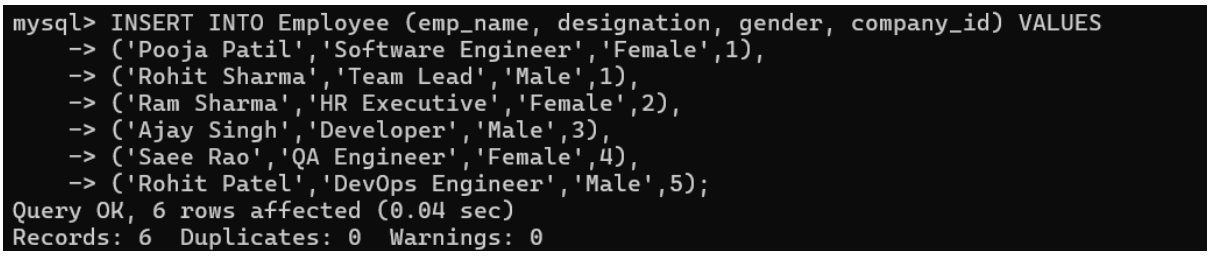
-> ('Ajay Singh','Developer','Male',3),

-> ('Saee Rao','QA Engineer','Female',4),

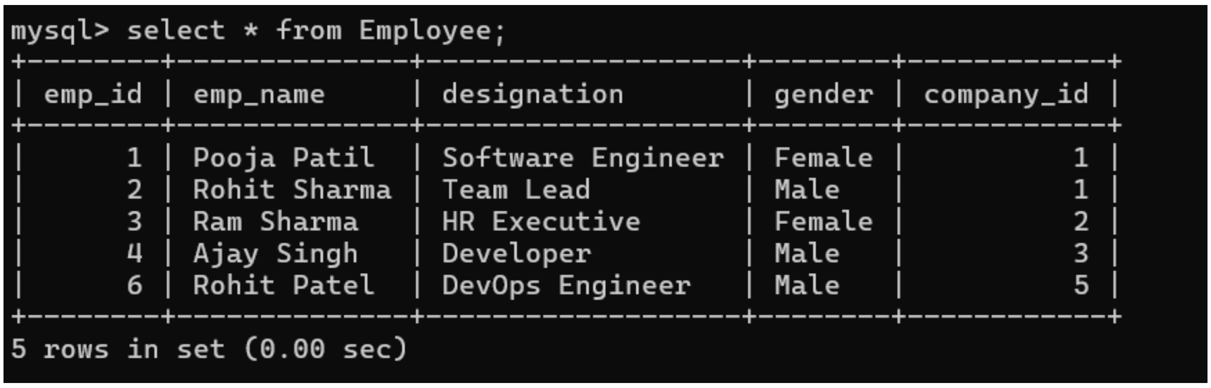
-> ('Rohit Patel','DevOps Engineer','Male',5);

Query OK, 6 rows affected (0.04 sec)

Records: 6 Duplicates: 0 Warnings: 0



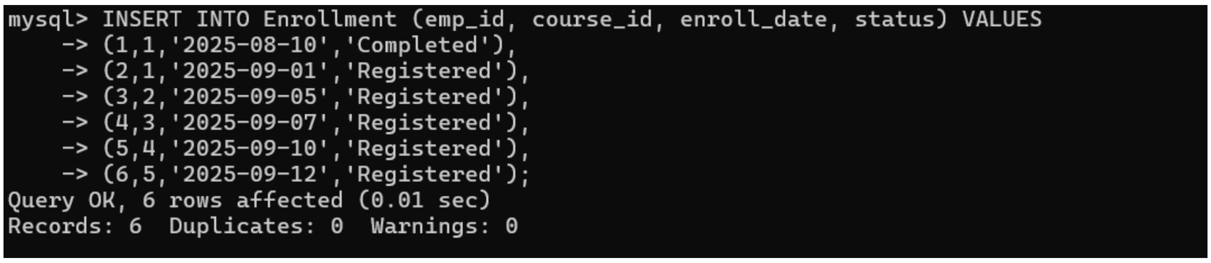
```
mysql> INSERT INTO Employee (emp_name, designation, gender, company_id) VALUES
    -> ('Pooja Patil','Software Engineer','Female',1),
    -> ('Rohit Sharma','Team Lead','Male',1),
    -> ('Ram Sharma','HR Executive','Female',2),
    -> ('Ajay Singh','Developer','Male',3),
    -> ('Saee Rao','QA Engineer','Female',4),
    -> ('Rohit Patel','DevOps Engineer','Male',5);
Query OK, 6 rows affected (0.04 sec)
Records: 6  Duplicates: 0  Warnings: 0
```



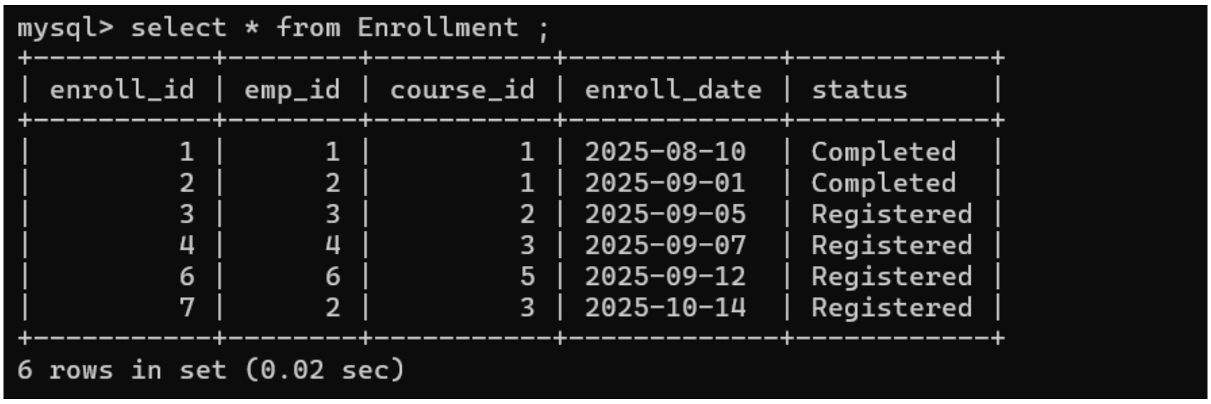
```
mysql> select * from Employee;
+--------+--------------+-------------------+--------+------------+
| emp_id | emp_name     | designation       | gender | company_id |
+--------+--------------+-------------------+--------+------------+
|      1 | Pooja Patil  | Software Engineer | Female |          1 |
|      2 | Rohit Sharma | Team Lead         | Male   |          1 |
|      3 | Ram Sharma   | HR Executive      | Female |          2 |
|      4 | Ajay Singh   | Developer         | Male   |          3 |
|      6 | Rohit Patel  | DevOps Engineer   | Male   |          5 |
+--------+--------------+-------------------+--------+------------+
5 rows in set (0.00 sec)
```

➢ Enrollment Table

```
mysql> INSERT INTO Enrollment (emp_id, course_id, enroll_date, status) VALUES
   -> (1,1,'2025-08-10','Completed'),
   -> (2,1,'2025-09-01','Registered'),
   -> (3,2,'2025-09-05','Registered'),
   -> (4,3,'2025-09-07','Registered'),
   -> (5,4,'2025-09-10','Registered'),
   -> (6,5,'2025-09-12','Registered');
Query OK, 6 rows affected (0.01 sec)
Records: 6  Duplicates: 0  Warnings: 0
```



```
mysql> INSERT INTO Enrollment (emp_id, course_id, enroll_date, status) VALUES
    -> (1,1,'2025-08-10','Completed'),
    -> (2,1,'2025-09-01','Registered'),
    -> (3,2,'2025-09-05','Registered'),
    -> (4,3,'2025-09-07','Registered'),
    -> (5,4,'2025-09-10','Registered'),
    -> (6,5,'2025-09-12','Registered');
Query OK, 6 rows affected (0.01 sec)
Records: 6  Duplicates: 0  Warnings: 0
```

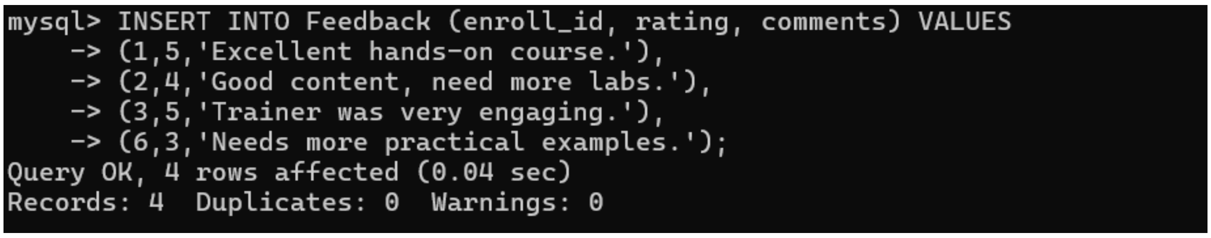


```
mysql> select * from Enrollment ;
+-----------+--------+-----------+-------------+------------+
| enroll_id | emp_id | course_id | enroll_date | status     |
+-----------+--------+-----------+-------------+------------+
|         1 |      1 |         1 | 2025-08-10  | Completed  |
|         2 |      2 |         1 | 2025-09-01  | Completed  |
|         3 |      3 |         2 | 2025-09-05  | Registered |
|         4 |      4 |         3 | 2025-09-07  | Registered |
|         6 |      6 |         5 | 2025-09-12  | Registered |
|         7 |      2 |         3 | 2025-10-14  | Registered |
+-----------+--------+-----------+-------------+------------+
6 rows in set (0.02 sec)
```

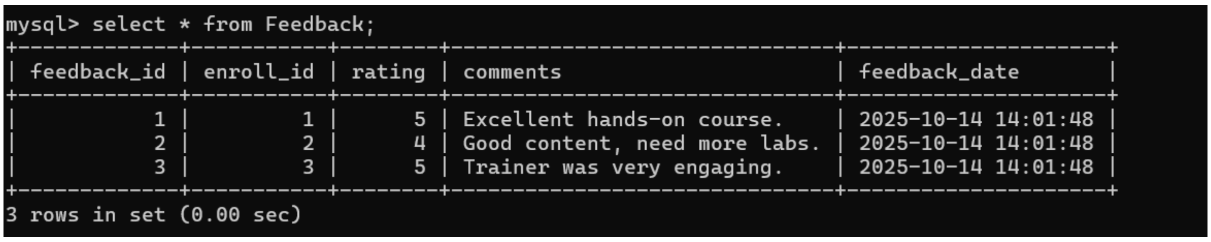
==

➢ Feedback Table

```
mysql> INSERT INTO Feedback (enroll_id, rating, comments) VALUES
   -> (1,5,'Excellent hands-on course.'),
   -> (2,4,'Good content, need more labs.'),
   -> (3,5,'Trainer was very engaging.'),
   -> (6,3,'Needs more practical examples.');
Query OK, 4 rows affected (0.04 sec)
Records: 4  Duplicates: 0  Warnings: 0
```



```
mysql> INSERT INTO Feedback (enroll_id, rating, comments) VALUES
    -> (1,5,'Excellent hands-on course.'),
    -> (2,4,'Good content, need more labs.'),
    -> (3,5,'Trainer was very engaging.'),
    -> (6,3,'Needs more practical examples.');
Query OK, 4 rows affected (0.04 sec)
Records: 4  Duplicates: 0  Warnings: 0
```



```
mysql> select * from Feedback;
+-------------+-----------+--------+-------------------------------+---------------------+
| feedback_id | enroll_id | rating | comments                      | feedback_date       |
+-------------+-----------+--------+-------------------------------+---------------------+
|           1 |         1 |      5 | Excellent hands-on course.    | 2025-10-14 14:01:48 |
|           2 |         2 |      4 | Good content, need more labs. | 2025-10-14 14:01:48 |
|           3 |         3 |      5 | Trainer was very engaging.    | 2025-10-14 14:01:48 |
+-------------+-----------+--------+-------------------------------+---------------------+
3 rows in set (0.00 sec)
```

===

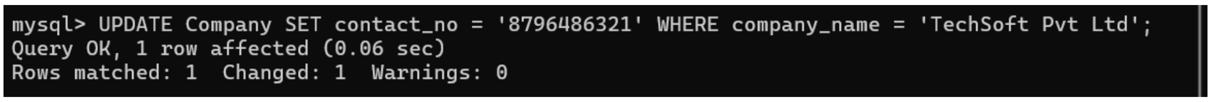
---------- UPDATE examples ---------

- **Change contact number for TechSoft**

mysql> UPDATE Company SET contact_no = '8796486321' WHERE company_name = 'TechSoft Pvt Ltd';

Query OK, 1 row affected (0.06 sec)

Rows matched: 1 Changed: 1 Warnings: 0



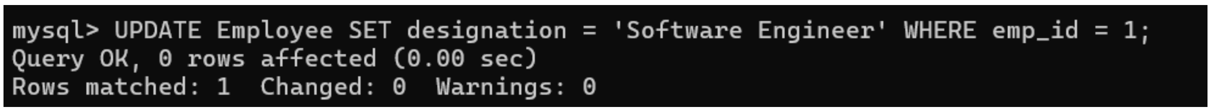
```
mysql> UPDATE Company SET contact_no = '8796486321' WHERE company_name = 'TechSoft Pvt Ltd';
Query OK, 1 row affected (0.06 sec)
Rows matched: 1  Changed: 1  Warnings: 0
```

- **Promote employee Pooja Patil**

mysql> UPDATE Employee SET designation = 'Software Engineer' WHERE emp_id = 1;

Query OK, 0 rows affected (0.00 sec)

Rows matched: 1 Changed: 0 Warnings: 0



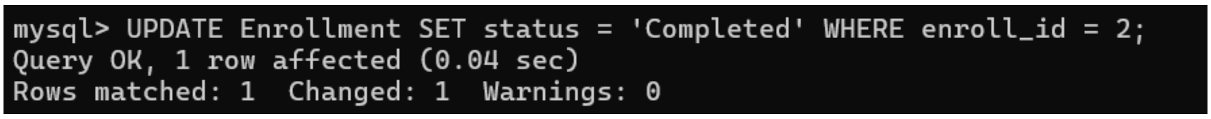
```
mysql> UPDATE Employee SET designation = 'Software Engineer' WHERE emp_id = 1;
Query OK, 0 rows affected (0.00 sec)
Rows matched: 1  Changed: 0  Warnings: 0
```

- **Mark enrollment 2 as Completed**

mysql> UPDATE Enrollment SET status = 'Completed' WHERE enroll_id = 2;

Query OK, 1 row affected (0.04 sec)

Rows matched: 1 Changed: 1 Warnings: 0



```
mysql> UPDATE Enrollment SET status = 'Completed' WHERE enroll_id = 2;
Query OK, 1 row affected (0.04 sec)
Rows matched: 1  Changed: 1  Warnings: 0
```

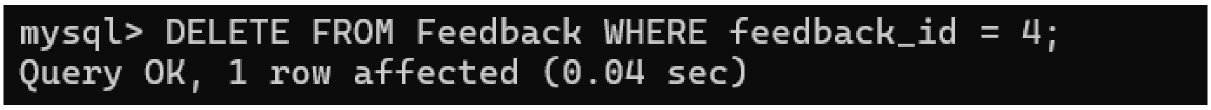
==

---------- DELETE examples ---------

- **Delete a specific feedback**

mysql> DELETE FROM Feedback WHERE feedback_id = 4;

Query OK, 1 row affected (0.04 sec)

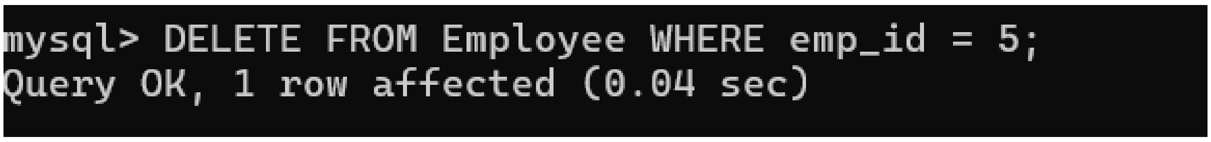


```
mysql> DELETE FROM Feedback WHERE feedback_id = 4;
Query OK, 1 row affected (0.04 sec)
```

- **Delete an employee (cascades to enrollments & feedback)**

mysql> DELETE FROM Employee WHERE emp_id = 5;

Query OK, 1 row affected (0.04 sec)

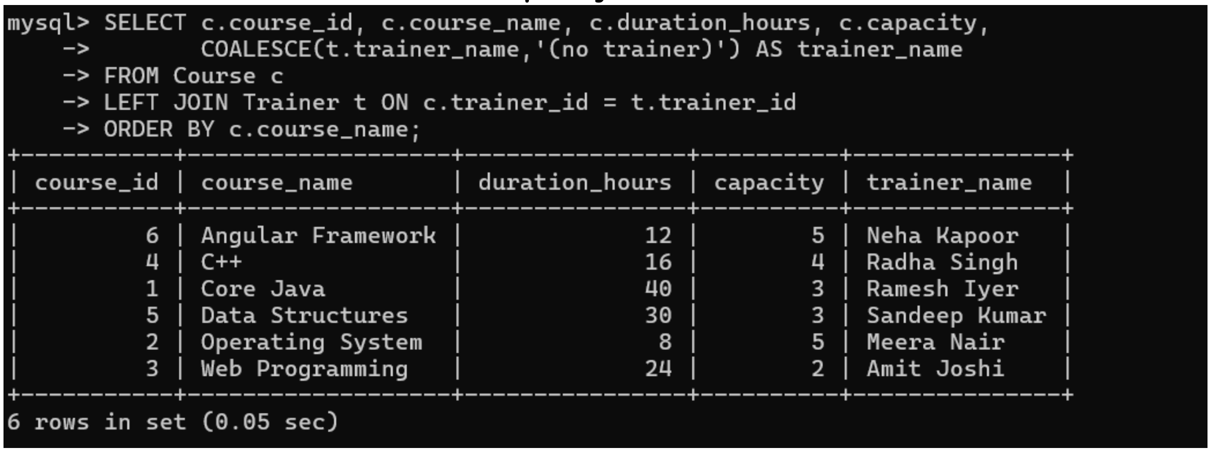


```
mysql> DELETE FROM Employee WHERE emp_id = 5;
Query OK, 1 row affected (0.04 sec)
```

==

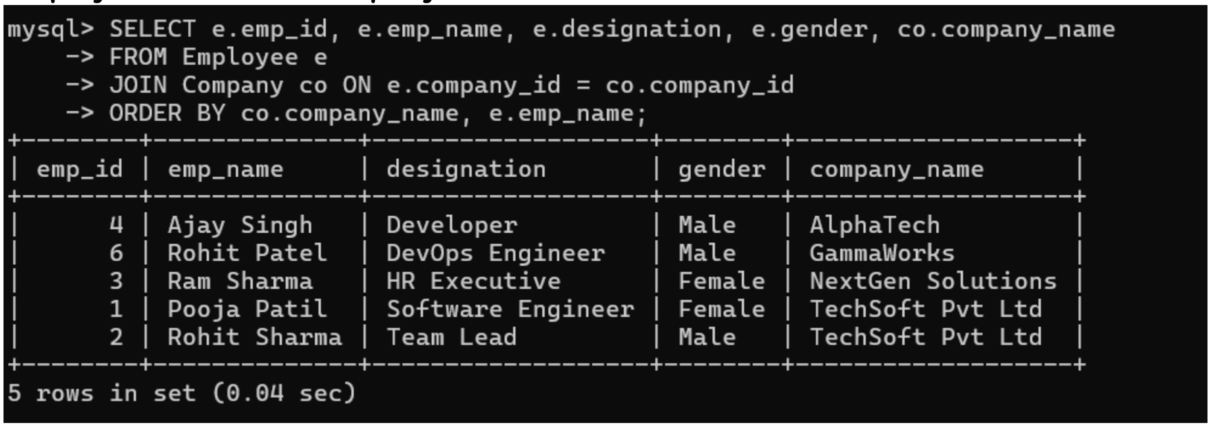
4) SQL Queries – Joins, Aggregates, Grouping, Subqueries

- **All courses with trainer name and capacity**



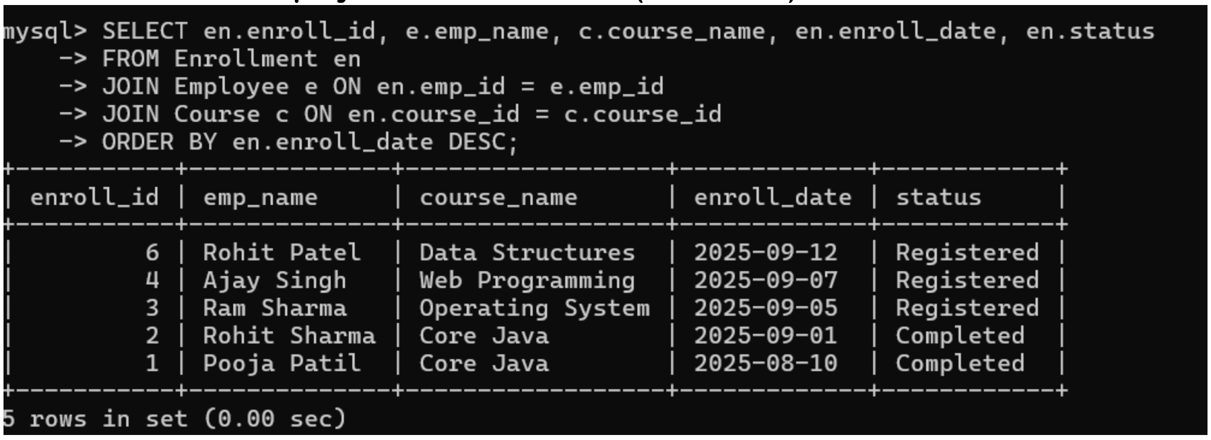
```
mysql> SELECT c.course_id, c.course_name, c.duration_hours, c.capacity,
    ->        COALESCE(t.trainer_name,'(no trainer)') AS trainer_name
    -> FROM Course c
    -> LEFT JOIN Trainer t ON c.trainer_id = t.trainer_id
    -> ORDER BY c.course_name;
+-----------+-------------------+----------------+----------+----------------+
| course_id | course_name       | duration_hours | capacity | trainer_name   |
+-----------+-------------------+----------------+----------+----------------+
|         6 | Angular Framework |             12 |        5 | Neha Kapoor    |
|         4 | C++               |             16 |        4 | Radha Singh    |
|         1 | Core Java         |             40 |        3 | Ramesh Iyer    |
|         5 | Data Structures   |             30 |        3 | Sandeep Kumar  |
|         2 | Operating System  |              8 |        5 | Meera Nair     |
|         3 | Web Programming   |             24 |        2 | Amit Joshi     |
+-----------+-------------------+----------------+----------+----------------+
6 rows in set (0.05 sec)
```

- **Employees with their company**



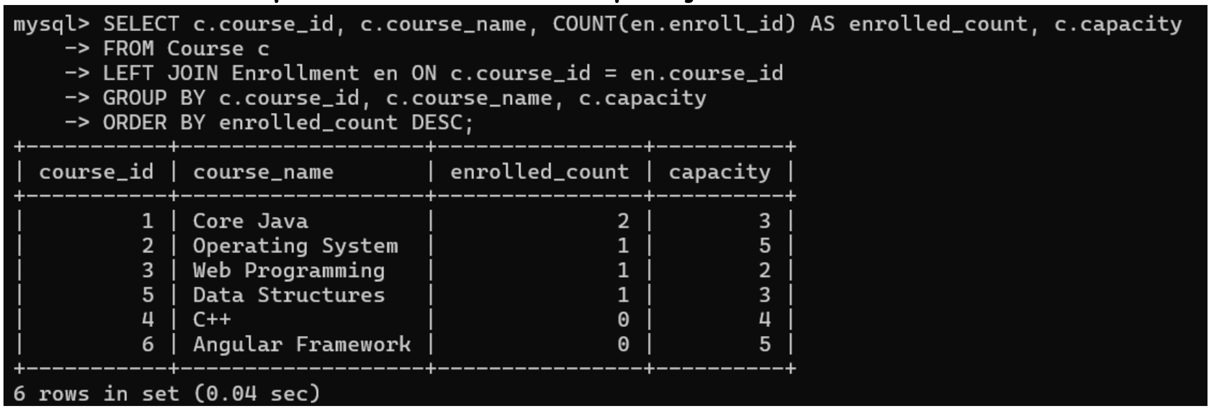
```
mysql> SELECT e.emp_id, e.emp_name, e.designation, e.gender, co.company_name
    -> FROM Employee e
    -> JOIN Company co ON e.company_id = co.company_id
    -> ORDER BY co.company_name, e.emp_name;
+--------+--------------+-------------------+--------+-------------------+
| emp_id | emp_name     | designation       | gender | company_name      |
+--------+--------------+-------------------+--------+-------------------+
|      4 | Ajay Singh   | Developer         | Male   | AlphaTech         |
|      6 | Rohit Patel  | DevOps Engineer   | Male   | GammaWorks        |
|      3 | Ram Sharma   | HR Executive      | Female | NextGen Solutions |
|      1 | Pooja Patil  | Software Engineer | Female | TechSoft Pvt Ltd  |
|      2 | Rohit Sharma | Team Lead         | Male   | TechSoft Pvt Ltd  |
+--------+--------------+-------------------+--------+-------------------+
5 rows in set (0.04 sec)
```

- **Enrollments with employee & course details (latest first)**



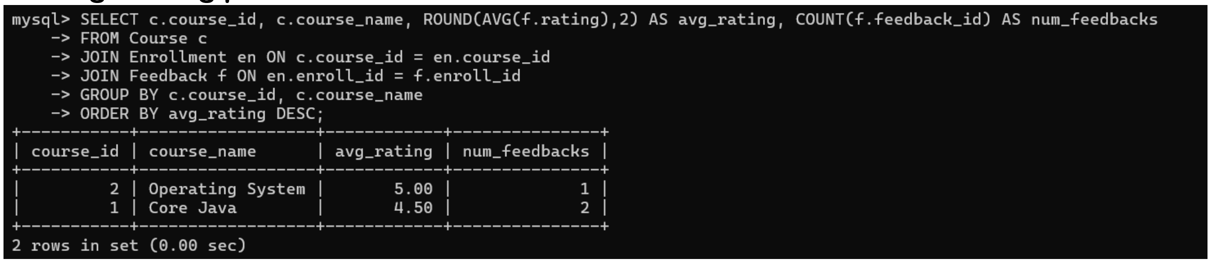
```
mysql> SELECT en.enroll_id, e.emp_name, c.course_name, en.enroll_date, en.status
    -> FROM Enrollment en
    -> JOIN Employee e ON en.emp_id = e.emp_id
    -> JOIN Course c ON en.course_id = c.course_id
    -> ORDER BY en.enroll_date DESC;
+-----------+--------------+------------------+-------------+------------+
| enroll_id | emp_name     | course_name      | enroll_date | status     |
+-----------+--------------+------------------+-------------+------------+
|         6 | Rohit Patel  | Data Structures  | 2025-09-12  | Registered |
|         4 | Ajay Singh   | Web Programming  | 2025-09-07  | Registered |
|         3 | Ram Sharma   | Operating System | 2025-09-05  | Registered |
|         2 | Rohit Sharma | Core Java        | 2025-09-01  | Completed  |
|         1 | Pooja Patil  | Core Java        | 2025-08-10  | Completed  |
+-----------+--------------+------------------+-------------+------------+
5 rows in set (0.00 sec)
```

- **Count enrollments per course and show capacity**



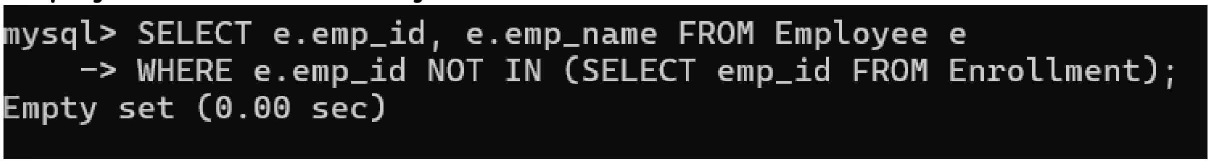
```
mysql> SELECT c.course_id, c.course_name, COUNT(en.enroll_id) AS enrolled_count, c.capacity
    -> FROM Course c
    -> LEFT JOIN Enrollment en ON c.course_id = en.course_id
    -> GROUP BY c.course_id, c.course_name, c.capacity
    -> ORDER BY enrolled_count DESC;
+-----------+--------------------+----------------+----------+
| course_id | course_name        | enrolled_count | capacity |
+-----------+--------------------+----------------+----------+
|         1 | Core Java          |              2 |        3 |
|         2 | Operating System   |              1 |        5 |
|         3 | Web Programming    |              1 |        2 |
|         5 | Data Structures    |              1 |        3 |
|         4 | C++                |              0 |        4 |
|         6 | Angular Framework  |              0 |        5 |
+-----------+--------------------+----------------+----------+
6 rows in set (0.04 sec)
```

- **Average rating per course**



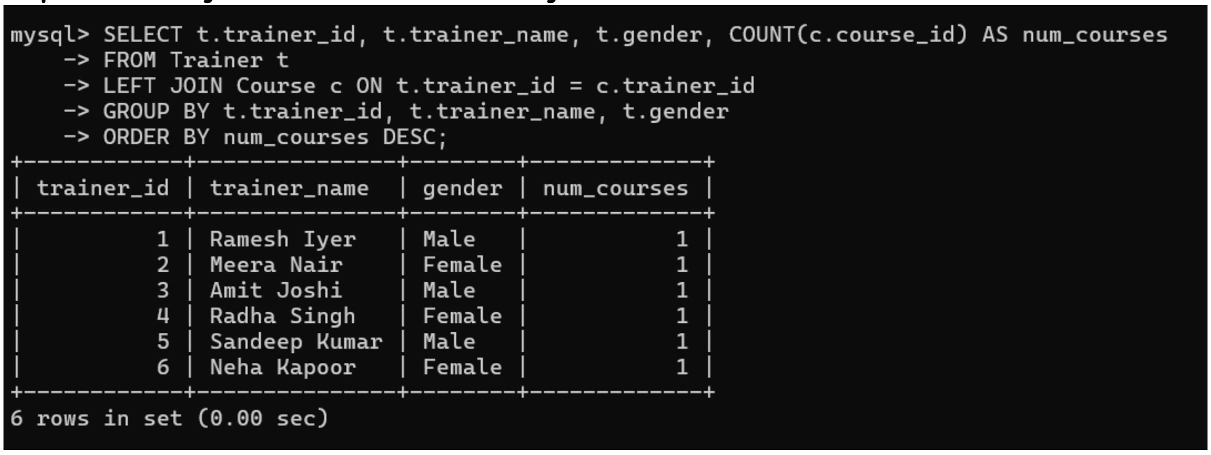
```
mysql> SELECT c.course_id, c.course_name, ROUND(AVG(f.rating),2) AS avg_rating, COUNT(f.feedback_id) AS num_feedbacks
    -> FROM Course c
    -> JOIN Enrollment en ON c.course_id = en.course_id
    -> JOIN Feedback f ON en.enroll_id = f.enroll_id
    -> GROUP BY c.course_id, c.course_name
    -> ORDER BY avg_rating DESC;
+-----------+------------------+------------+---------------+
| course_id | course_name      | avg_rating | num_feedbacks |
+-----------+------------------+------------+---------------+
|         2 | Operating System |       5.00 |             1 |
|         1 | Core Java        |       4.50 |             2 |
+-----------+------------------+------------+---------------+
2 rows in set (0.00 sec)
```

- **Employees not enrolled in any course**



```
mysql> SELECT e.emp_id, e.emp_name FROM Employee e
    -> WHERE e.emp_id NOT IN (SELECT emp_id FROM Enrollment);
Empty set (0.00 sec)
```

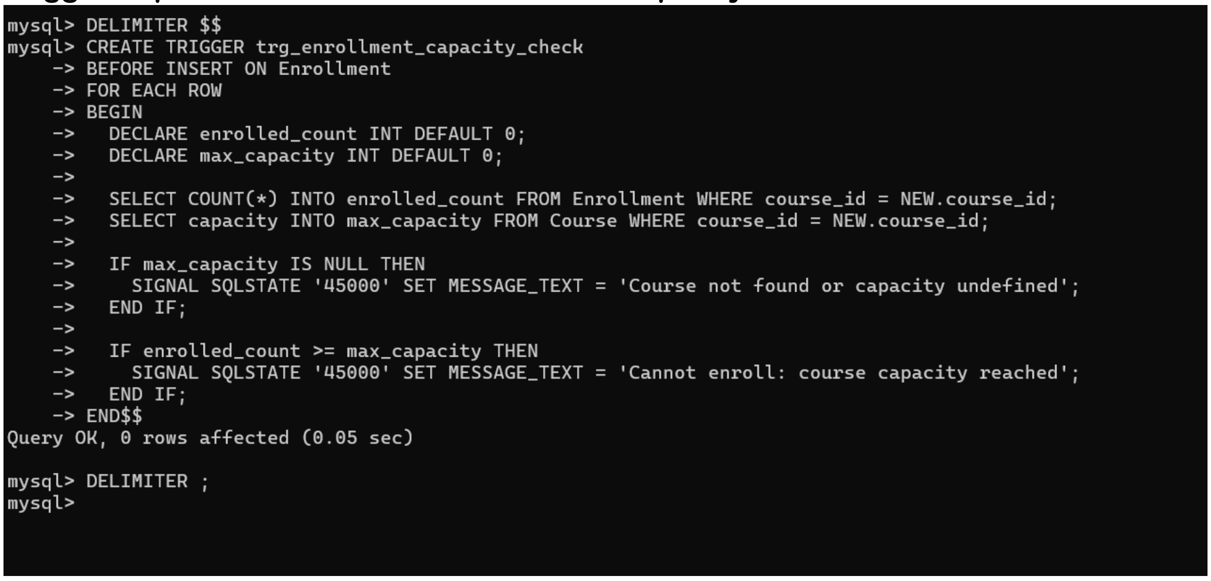
- **Top trainers by number of courses they handle**



```
mysql> SELECT t.trainer_id, t.trainer_name, t.gender, COUNT(c.course_id) AS num_courses
    -> FROM Trainer t
    -> LEFT JOIN Course c ON t.trainer_id = c.trainer_id
    -> GROUP BY t.trainer_id, t.trainer_name, t.gender
    -> ORDER BY num_courses DESC;
+------------+---------------+--------+-------------+
| trainer_id | trainer_name  | gender | num_courses |
+------------+---------------+--------+-------------+
|          1 | Ramesh Iyer   | Male   |           1 |
|          2 | Meera Nair    | Female |           1 |
|          3 | Amit Joshi    | Male   |           1 |
|          4 | Radha Singh   | Female |           1 |
|          5 | Sandeep Kumar | Male   |           1 |
|          6 | Neha Kapoor   | Female |           1 |
+------------+---------------+--------+-------------+
6 rows in set (0.00 sec)
```

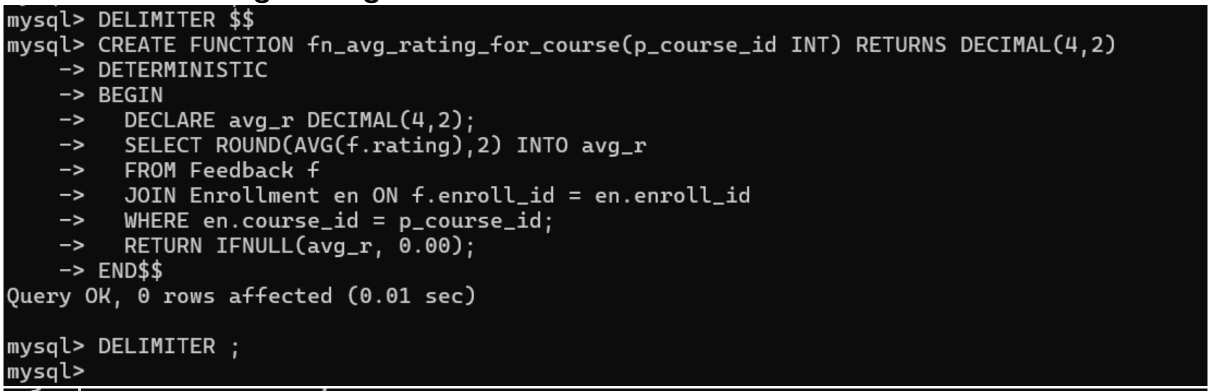
5) Trigger, Function, Stored Procedure

- **Trigger — prevent enrollment when course capacity reached**

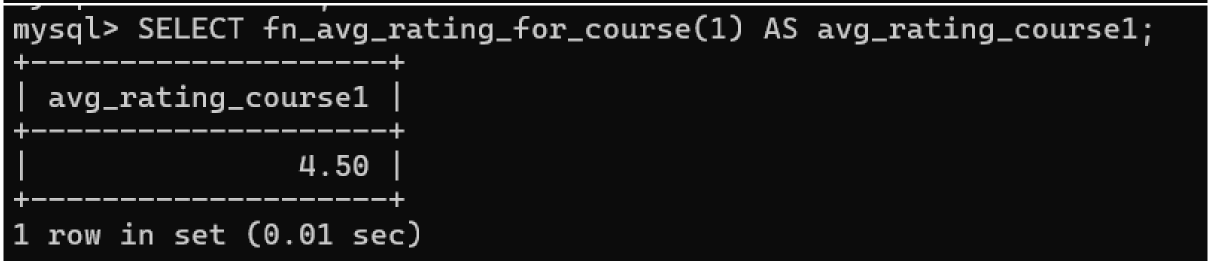


```
mysql> DELIMITER $$
mysql> CREATE TRIGGER trg_enrollment_capacity_check
    -> BEFORE INSERT ON Enrollment
    -> FOR EACH ROW
    -> BEGIN
    ->   DECLARE enrolled_count INT DEFAULT 0;
    ->   DECLARE max_capacity INT DEFAULT 0;
    ->
    ->   SELECT COUNT(*) INTO enrolled_count FROM Enrollment WHERE course_id = NEW.course_id;
    ->   SELECT capacity INTO max_capacity FROM Course WHERE course_id = NEW.course_id;
    ->
    ->   IF max_capacity IS NULL THEN
    ->     SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT = 'Course not found or capacity undefined';
    ->   END IF;
    ->
    ->   IF enrolled_count >= max_capacity THEN
    ->     SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT = 'Cannot enroll: course capacity reached';
    ->   END IF;
    -> END$$
Query OK, 0 rows affected (0.05 sec)

mysql> DELIMITER ;
mysql>
```

- **Function — average rating for a course**

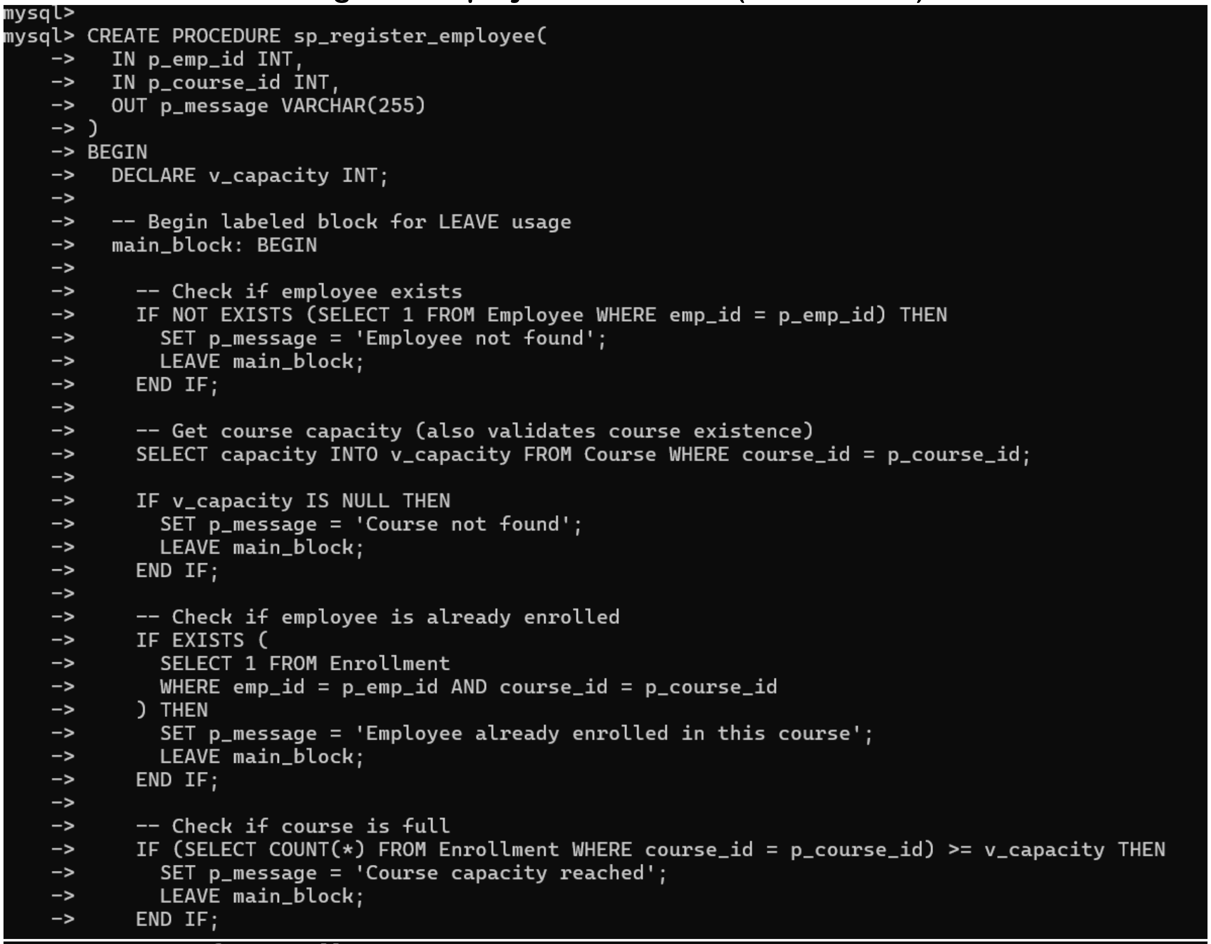


```
mysql> DELIMITER $$
mysql> CREATE FUNCTION fn_avg_rating_for_course(p_course_id INT) RETURNS DECIMAL(4,2)
    -> DETERMINISTIC
    -> BEGIN
    ->   DECLARE avg_r DECIMAL(4,2);
    ->   SELECT ROUND(AVG(f.rating),2) INTO avg_r
    ->   FROM Feedback f
    ->   JOIN Enrollment en ON f.enroll_id = en.enroll_id
    ->   WHERE en.course_id = p_course_id;
    ->   RETURN IFNULL(avg_r, 0.00);
    -> END$$
Query OK, 0 rows affected (0.01 sec)

mysql> DELIMITER ;
mysql>
```

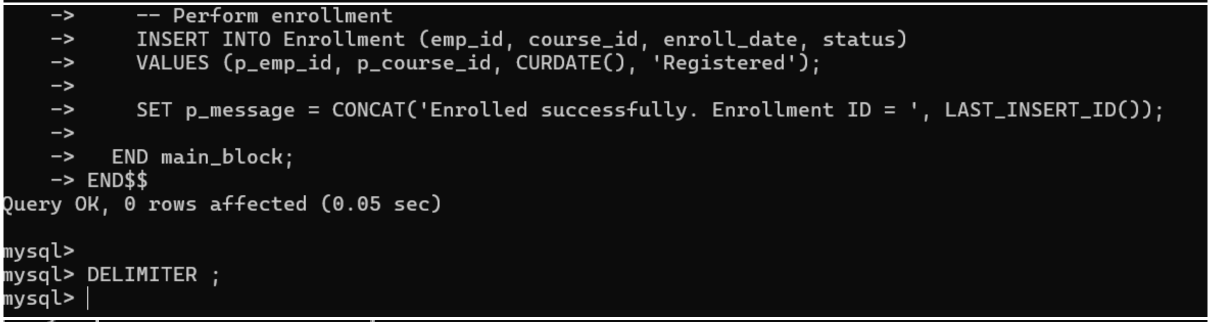


```
mysql> SELECT fn_avg_rating_for_course(1) AS avg_rating_course1;
+--------------------+
| avg_rating_course1 |
+--------------------+
|               4.50 |
+--------------------+
1 row in set (0.01 sec)
```

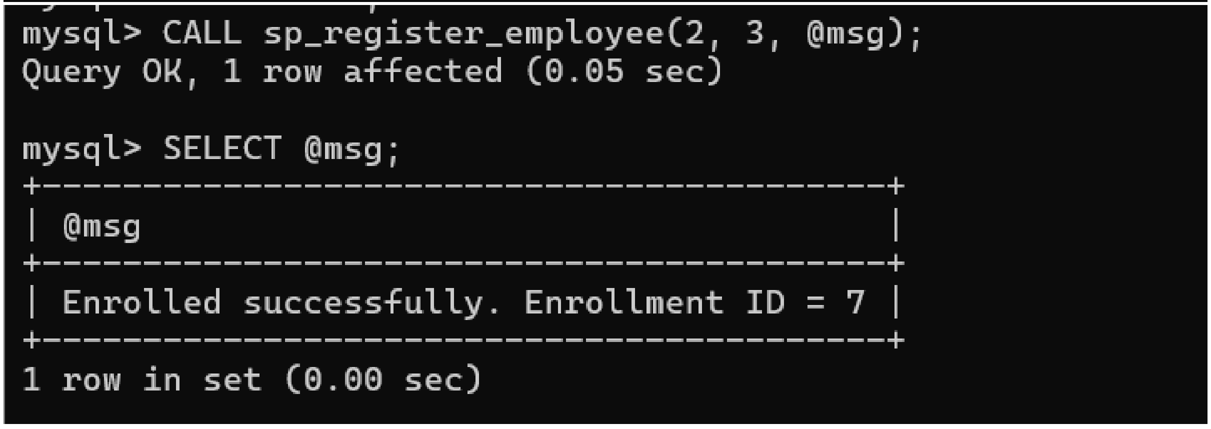
- **Stored Procedure — register employee to a course (with checks)**



```
mysql>
mysql> CREATE PROCEDURE sp_register_employee(
    ->   IN p_emp_id INT,
    ->   IN p_course_id INT,
    ->   OUT p_message VARCHAR(255)
    -> )
    -> BEGIN
    ->   DECLARE v_capacity INT;
    ->
    ->   -- Begin labeled block for LEAVE usage
    ->   main_block: BEGIN
    ->
    ->     -- Check if employee exists
    ->     IF NOT EXISTS (SELECT 1 FROM Employee WHERE emp_id = p_emp_id) THEN
    ->       SET p_message = 'Employee not found';
    ->       LEAVE main_block;
    ->     END IF;
    ->
    ->     -- Get course capacity (also validates course existence)
    ->     SELECT capacity INTO v_capacity FROM Course WHERE course_id = p_course_id;
    ->
    ->     IF v_capacity IS NULL THEN
    ->       SET p_message = 'Course not found';
    ->       LEAVE main_block;
    ->     END IF;
    ->
    ->     -- Check if employee is already enrolled
    ->     IF EXISTS (
    ->       SELECT 1 FROM Enrollment
    ->       WHERE emp_id = p_emp_id AND course_id = p_course_id
    ->     ) THEN
    ->       SET p_message = 'Employee already enrolled in this course';
    ->       LEAVE main_block;
    ->     END IF;
    ->
    ->     -- Check if course is full
    ->     IF (SELECT COUNT(*) FROM Enrollment WHERE course_id = p_course_id) >= v_capacity THEN
    ->       SET p_message = 'Course capacity reached';
    ->       LEAVE main_block;
    ->     END IF;
```



```
    ->     -- Perform enrollment
    ->     INSERT INTO Enrollment (emp_id, course_id, enroll_date, status)
    ->     VALUES (p_emp_id, p_course_id, CURDATE(), 'Registered');
    ->
    ->     SET p_message = CONCAT('Enrolled successfully. Enrollment ID = ', LAST_INSERT_ID());
    ->
    ->   END main_block;
    -> END$$
Query OK, 0 rows affected (0.05 sec)

mysql>
mysql> DELIMITER ;
mysql> |
```



```
mysql> CALL sp_register_employee(2, 3, @msg);
Query OK, 1 row affected (0.05 sec)

mysql> SELECT @msg;
+-----------------------------------------+
| @msg                                    |
+-----------------------------------------+
| Enrolled successfully. Enrollment ID = 7 |
+-----------------------------------------+
1 row in set (0.00 sec)
```

6) Normalize your database up to Third Normal Form (3NF) and provide a short explanation.

Normalization is the process of organizing data in a database to minimize redundancy and improve data integrity.
The **Corporate Training Management System** has been normalized step-by-step from an unstructured form to **Third Normal Form (3NF)** as described below.

- **1NF (atomicity & repeating groups)**
  In the unnormalized design, multiple courses could be stored in a single record for each employee, causing repeating groups.
  To satisfy 1NF, we ensure that each attribute holds only atomic (single) values, and repeating data is moved into separate rows.
  Example: multiple courses per employee become multiple rows in Enrollment.
  Create separate entity tables: Employee, Course, Trainer, Company.

- **2NF (remove partial dependencies)**
  In 2NF, every non-key attribute must depend on the entire primary key, not just a part of it.In a table like Enrollment(emp_id, course_id), attributes such as emp_name or company_name would depend only on part of the key (emp_id) — creating a partial dependency.
  To remove this:
  Employee details (emp_name, gender, designation) were moved to the Employee table.
  Company details were moved to a separate Company table.
  Each employee references a company using company_id.
  This ensures that all non-key attributes depend on the whole key and not on part of it.

- **3NF (remove transitive dependencies)**
  3NF removes attributes that depend on non-key attributes (transitive dependency).
  For example, trainer_name and trainer_email depend on trainer_id, not directly on course_id. Therefore, these attributes were moved to a separate Trainer table, and the Course table now stores only trainer_id as a foreign key.
  This step eliminates indirect dependencies and ensures that each non-key attribute depends only on the primary key of its own table.

- **Final normalized tables (already implemented)**

➢ Company(company_id, company_name, location, contact_no)
➢ Trainer(trainer_id, trainer_name, gender, expertise, email)
➢ Course(course_id, course_name, duration_hours, capacity, trainer_id)
➢ Employee(emp_id, emp_name, designation, gender, company_id)
➢ Enrollment(enroll_id, emp_id, course_id, enroll_date, status)
 — associative resolving M:N
➢ Feedback(feedback_id, enroll_id, rating, comments, feedback_date)
➢ Each table's non-key attributes depend only on that table's primary key
— satisfies 3NF.