-- UNIVERSITY COURSE REGISTRATION SYSTEM (MySQL)

-- File: university\_course\_reg.sql

-- Description: Complete SQL project for managing students, courses, faculty, and enrollment.

DROP DATABASE IF EXISTS university\_db;

CREATE DATABASE university\_db CHARACTER SET utf8mb4;

USE university\_db;

/\* ------------------------

1. Tables

------------------------ \*/

-- students table

CREATE TABLE students (

id BIGINT UNSIGNED AUTO\_INCREMENT PRIMARY KEY,

roll\_no VARCHAR(20) NOT NULL UNIQUE,

name VARCHAR(100) NOT NULL,

email VARCHAR(150) UNIQUE,

dob DATE,

department VARCHAR(100),

created\_at TIMESTAMP DEFAULT CURRENT\_TIMESTAMP

);

-- faculty table

CREATE TABLE faculty (

id BIGINT UNSIGNED AUTO\_INCREMENT PRIMARY KEY,

emp\_id VARCHAR(20) NOT NULL UNIQUE,

name VARCHAR(100) NOT NULL,

email VARCHAR(150) UNIQUE,

department VARCHAR(100),

created\_at TIMESTAMP DEFAULT CURRENT\_TIMESTAMP

);

-- courses table

CREATE TABLE courses (

id BIGINT UNSIGNED AUTO\_INCREMENT PRIMARY KEY,

code VARCHAR(10) NOT NULL UNIQUE,

title VARCHAR(150) NOT NULL,

description TEXT,

credit\_hours INT NOT NULL CHECK(credit\_hours > 0),

department VARCHAR(100),

faculty\_id BIGINT UNSIGNED,

max\_seats INT NOT NULL DEFAULT 40,

FOREIGN KEY (faculty\_id) REFERENCES faculty(id) ON DELETE SET NULL

);

-- enrollment table

CREATE TABLE enrollment (

id BIGINT UNSIGNED AUTO\_INCREMENT PRIMARY KEY,

student\_id BIGINT UNSIGNED NOT NULL,

course\_id BIGINT UNSIGNED NOT NULL,

semester VARCHAR(20) NOT NULL,

grade CHAR(2) DEFAULT NULL,

enrolled\_at TIMESTAMP DEFAULT CURRENT\_TIMESTAMP,

UNIQUE (student\_id, course\_id, semester),

FOREIGN KEY (student\_id) REFERENCES students(id) ON DELETE CASCADE,

FOREIGN KEY (course\_id) REFERENCES courses(id) ON DELETE CASCADE

);

-- audit\_logs table

CREATE TABLE audit\_logs (

id BIGINT UNSIGNED AUTO\_INCREMENT PRIMARY KEY,

entity VARCHAR(100) NOT NULL,

entity\_id BIGINT UNSIGNED,

action VARCHAR(50) NOT NULL,

details JSON,

created\_at TIMESTAMP DEFAULT CURRENT\_TIMESTAMP

);

/\* ------------------------

2. Sample Data

------------------------ \*/

INSERT INTO students (roll\_no, name, email, dob, department) VALUES

('STU001','Alice Johnson','alice@uni.edu','2003-05-14','Computer Science'),

('STU002','Bob Kumar','bob@uni.edu','2002-11-23','Electronics');

INSERT INTO faculty (emp\_id, name, email, department) VALUES

('FAC001','Dr. Smith','smith@uni.edu','Computer Science'),

('FAC002','Dr. Meera','meera@uni.edu','Electronics');

INSERT INTO courses (code, title, description, credit\_hours, department, faculty\_id, max\_seats) VALUES

('CS101','Intro to Programming','Basics of Python and programming logic',4,'Computer Science',1,50),

('EC201','Circuit Theory','Analysis of electric circuits',3,'Electronics',2,40);

/\* ------------------------

3. Views

------------------------ \*/

-- student course summary

CREATE VIEW vw\_student\_courses AS

SELECT s.name AS student\_name, s.roll\_no, c.title AS course\_title, e.semester, e.grade

FROM enrollment e

JOIN students s ON e.student\_id = s.id

JOIN courses c ON e.course\_id = c.id;

-- faculty course load

CREATE VIEW vw\_faculty\_load AS

SELECT f.name AS faculty\_name, f.department, COUNT(c.id) AS total\_courses

FROM faculty f

LEFT JOIN courses c ON f.id = c.faculty\_id

GROUP BY f.id;

/\* ------------------------

4. Stored Procedure

------------------------ \*/

DROP PROCEDURE IF EXISTS sp\_register\_course;

DELIMITER $$

CREATE PROCEDURE sp\_register\_course(

IN p\_student\_id BIGINT,

IN p\_course\_id BIGINT,

IN p\_semester VARCHAR(20),

OUT p\_status VARCHAR(100)

)

BEGIN

DECLARE available\_seats INT;

DECLARE enrolled\_count INT;

-- Check current enrollment

SELECT COUNT(\*) INTO enrolled\_count FROM enrollment

WHERE course\_id = p\_course\_id AND semester = p\_semester;

SELECT max\_seats INTO available\_seats FROM courses WHERE id = p\_course\_id;

IF enrolled\_count >= available\_seats THEN

SET p\_status = 'No seats available';

ELSE

INSERT INTO enrollment (student\_id, course\_id, semester)

VALUES (p\_student\_id, p\_course\_id, p\_semester);

SET p\_status = 'Enrolled successfully';

END IF;

-- Audit log

INSERT INTO audit\_logs(entity, entity\_id, action, details)

VALUES ('enrollment', p\_student\_id, 'register\_course', JSON\_OBJECT('course\_id',p\_course\_id,'semester',p\_semester));

END$$

DELIMITER ;

/\* ------------------------

5. Trigger

------------------------ \*/

DROP TRIGGER IF EXISTS trg\_enrollment\_after\_insert;

DELIMITER $$

CREATE TRIGGER trg\_enrollment\_after\_insert

AFTER INSERT ON enrollment

FOR EACH ROW

BEGIN

INSERT INTO audit\_logs(entity, entity\_id, action, details)

VALUES ('enrollment', NEW.id, 'inserted', JSON\_OBJECT('student\_id', NEW.student\_id,'course\_id', NEW.course\_id,'semester', NEW.semester));

END$$

DELIMITER ;

/\* ------------------------

6. Test Procedure Call

------------------------ \*/

SET @status = '';

CALL sp\_register\_course(1, 1, '2025-Spring', @status);

SELECT @status AS Registration\_Status;

SELECT \* FROM vw\_student\_courses;

SELECT \* FROM vw\_faculty\_load;