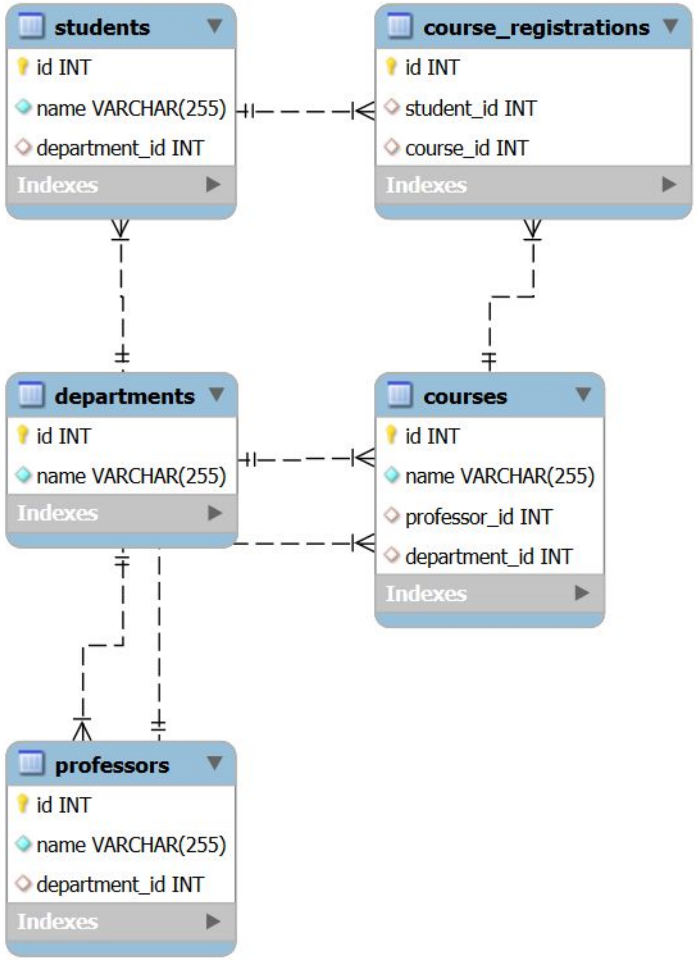
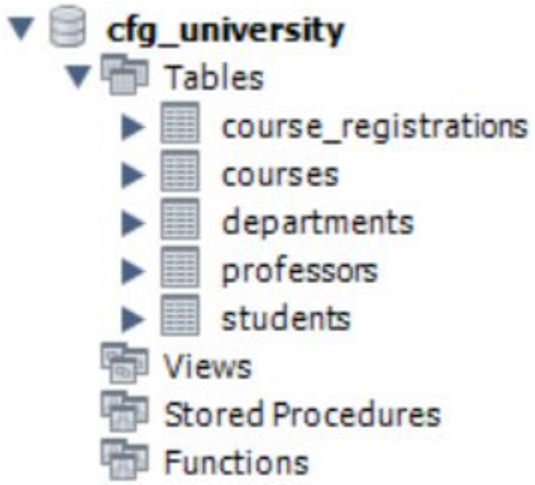
SQL Final Project

Sophia Clare Jenkinson



✔ Create DB diagram where all table relations are shown

- ✔ Create relational DB of your choice with minimum 5 tables
- ✔ Set Primary and Foreign Key constraints to create relations between the tables

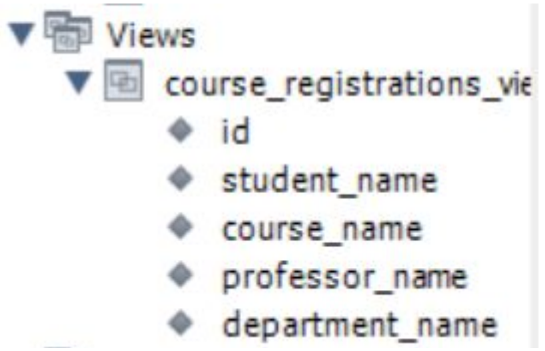


```
CREATE TABLE departments (
  id INT PRIMARY KEY AUTO_INCREMENT,
  name VARCHAR(255) NOT NULL
);

CREATE TABLE professors (
  id INT PRIMARY KEY AUTO_INCREMENT,
  name VARCHAR(255) NOT NULL,
  department_id INT,
  FOREIGN KEY (department_id) REFERENCES departments(id)
);

CREATE TABLE courses (
  id INT PRIMARY KEY AUTO_INCREMENT,
  name VARCHAR(255) NOT NULL,
  professor_id INT,
  department_id INT,
  FOREIGN KEY (professor_id) REFERENCES professors(id),
  FOREIGN KEY (department_id) REFERENCES departments(id)
);

CREATE TABLE students (
  id INT PRIMARY KEY AUTO_INCREMENT,
  name VARCHAR(255) NOT NULL,
  department_id INT,
  FOREIGN KEY (department_id) REFERENCES departments(id)
);

CREATE TABLE course_registrations (
  id INT PRIMARY KEY AUTO_INCREMENT,
  student_id INT,
  course_id INT,
  FOREIGN KEY (student_id) REFERENCES students(id),
  FOREIGN KEY (course_id) REFERENCES courses(id)
);
```

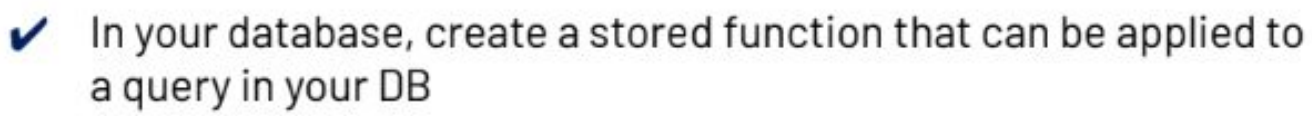
- ✓ Using any type of the joins create a view that combines multiple tables in a logical way

I will create a view that shows the course registrations along with student, course, and professor information:

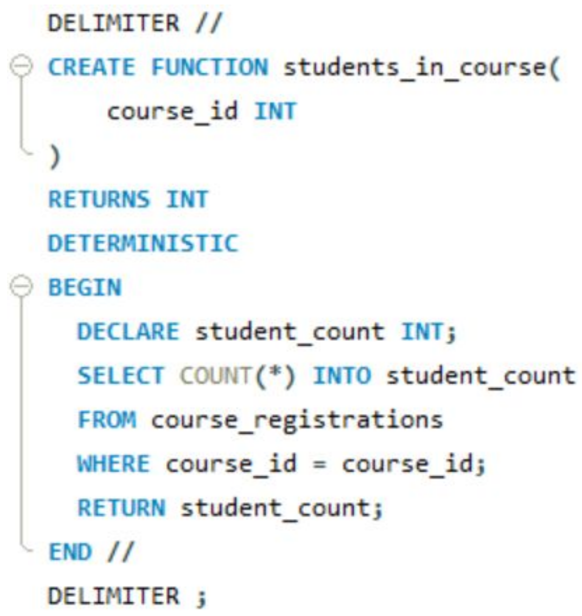
```
CREATE VIEW course_registrations_view AS
SELECT cr.id, s.name AS student_name, c.name AS course_name, p.name AS professor_name, d.name AS department_name
FROM course_registrations cr
JOIN students s ON cr.student_id = s.id
JOIN courses c ON cr.course_id = c.id
JOIN professors p ON c.professor_id = p.id
JOIN departments d ON c.department_id = d.id;
```



```
▼ Views
  ▼ course_registrations_vie
      ◆ id
      ◆ student_name
      ◆ course_name
      ◆ professor_name
      ◆ department_name
```


✔ In your database, create a stored function that can be applied to a query in your DB

I created a function that counts the number of students enrolled in a given course:



```
DELIMITER //
CREATE FUNCTION students_in_course(
    course_id INT
)
RETURNS INT
DETERMINISTIC
BEGIN
  DECLARE student_count INT;
  SELECT COUNT(*) INTO student_count
  FROM course_registrations
  WHERE course_id = course_id;
  RETURN student_count;
END //
DELIMITER ;
```

✔ In your database, create a stored function that can be applied to a query in your DB

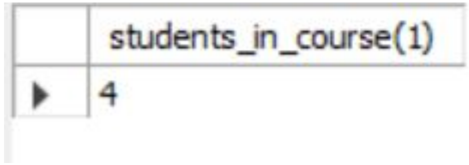
First I inserted some data into the tables so that I can test the function and queries.

```
INSERT INTO departments (name) VALUES ('Computer Science'), ('Mathematics'), ('Physics');
INSERT INTO professors (name, department_id) VALUES ('John Doe', 1), ('Jane Smith', 2), ('Sophia Jenkinson', 3);
INSERT INTO courses (name, professor_id, department_id) VALUES ('Programming 101', 1, 1), ('Algebra', 2, 2), ('Quantum Physics', 3, 3)
INSERT INTO students (name, department_id) VALUES ('Alice', 1), ('Lisa', 2), ('Lady Gaga', 3);
INSERT INTO course_registrations (student_id, course_id) VALUES (1, 1), (1, 2), (2, 2), (3, 3);
```

Testing the stored function:

```
SELECT students_in_course(1); -- Replace '1' with the desired course_id
```

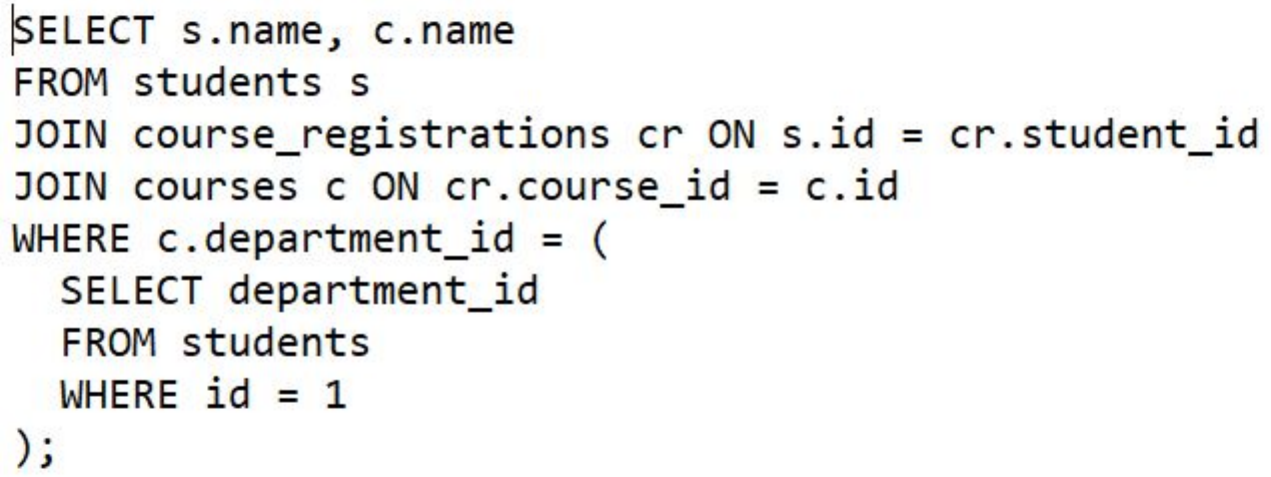
Result:



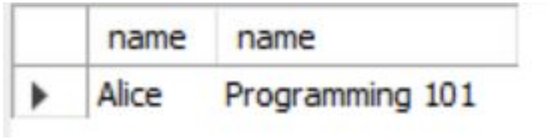
	students_in_course(1)
▶	4

- ✓ Prepare an example query with a subquery to demonstrate how to extract data from your DB for analysis

This query retrieves the names of courses that are registered by a specific student (identified by their "id") and belong to the same department as that student.



```
SELECT s.name, c.name
FROM students s
JOIN course_registrations cr ON s.id = cr.student_id
JOIN courses c ON cr.course_id = c.id
WHERE c.department_id = (
  SELECT department_id
  FROM students
  WHERE id = 1
);
```

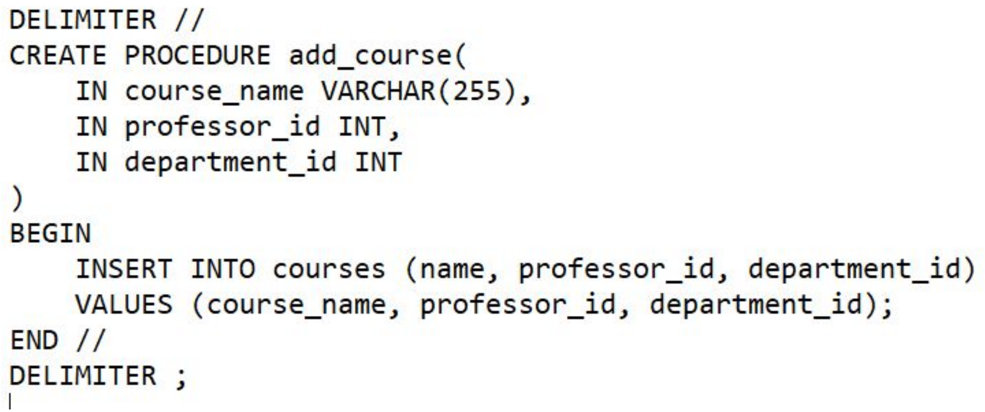


	name	name
▶	Alice	Programming 101

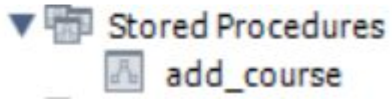
ADVANCED OPTIONS

✔ In your database, create a stored procedure and demonstrate how it runs

Stored procedure that adds a new course to the courses table.



```
DELIMITER //
CREATE PROCEDURE add_course(
    IN course_name VARCHAR(255),
    IN professor_id INT,
    IN department_id INT
)
BEGIN
    INSERT INTO courses (name, professor_id, department_id)
    VALUES (course_name, professor_id, department_id);
END //
DELIMITER ;
```

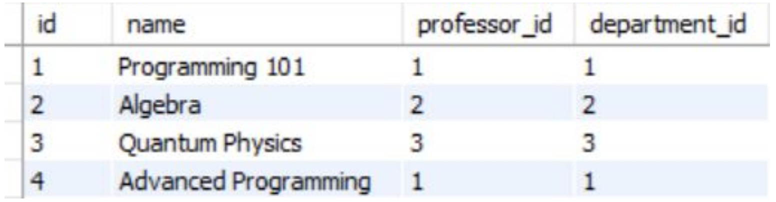


```
▼ Stored Procedures
    add_course
```

Run the procedure and verify:

```
CALL add_course('Advanced Programming', 1, 1);

SELECT * FROM courses
```

Result:



id	name	professor_id	department_id
1	Programming 101	1	1
2	Algebra	2	2
3	Quantum Physics	3	3
4	Advanced Programming	1	1

ADVANCED OPTIONS

✔ In your database, create a trigger and demonstrate how it runs

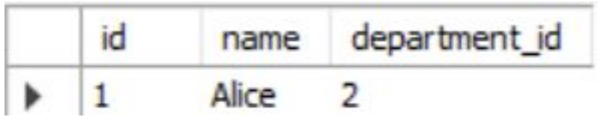
Trigger that automatically updates a student's department when they register for a course from a different department.

```
DELIMITER //
CREATE TRIGGER update_student_department
AFTER INSERT ON course_registrations
FOR EACH ROW
BEGIN
    UPDATE students
    SET department_id = (
        SELECT department_id
        FROM courses
        WHERE id = NEW.course_id
    )
    WHERE id = NEW.student_id;
END //
DELIMITER ;
```

Demonstration:

```
INSERT INTO course_registrations (student_id, course_id
VALUES (1, 2);


SELECT * FROM students WHERE id = 1;
```

Result:



	id	name	department_id
▶	1	Alice	2

Thank you!