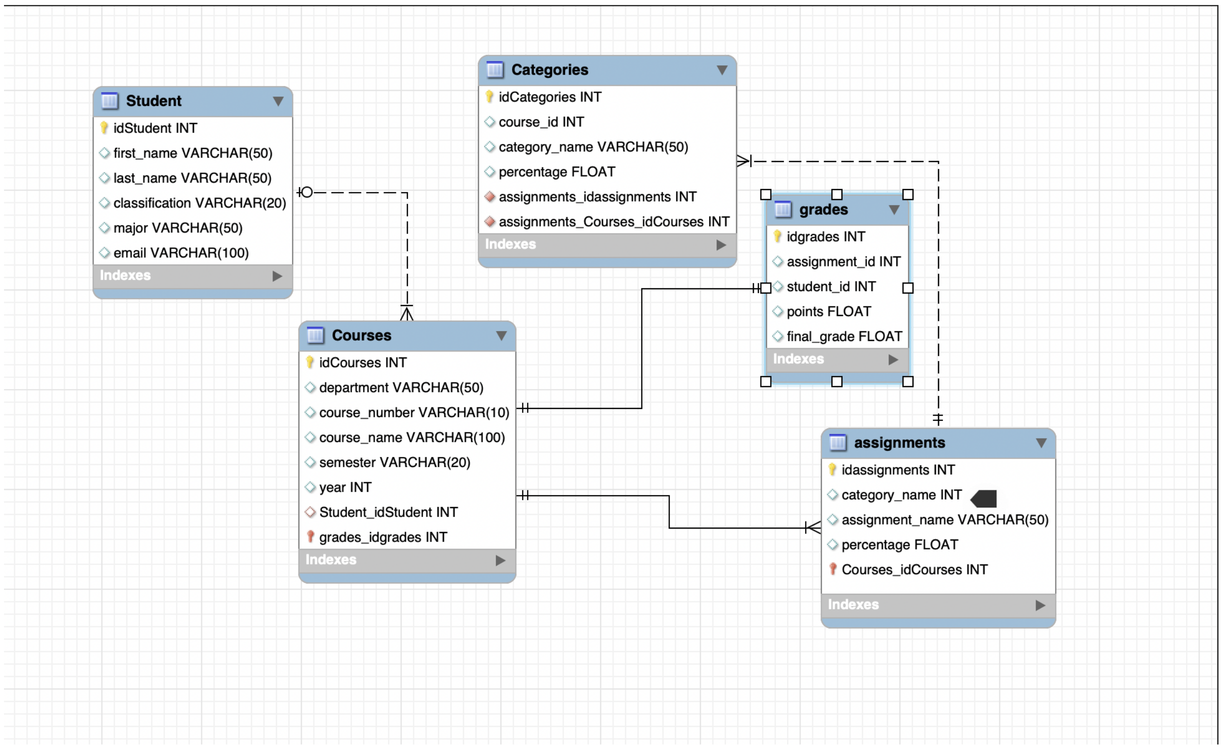
Project Submission

▼ ER- Diagram:



▼ Instructions:

This SQL script is optimized to run with SQLite. Either on Replit or through a CLI. To test any of the queries, copy them from here and pase them into the file.

▼ Main code:

```
CREATE TABLE students (
id INTEGER PRIMARY KEY AUTOINCREMENT,
first_name VARCHAR(50),
last_name VARCHAR(50),
classification VARCHAR(20),
major VARCHAR(50),
email VARCHAR(100)
);

CREATE TABLE courses (
id INTEGER PRIMARY KEY AUTOINCREMENT,
department VARCHAR(50),
course_number VARCHAR(10),
course_name VARCHAR(100),
semester VARCHAR(20),
year INTEGER
);

CREATE TABLE categories (
id INTEGER PRIMARY KEY AUTOINCREMENT,
course_id INTEGER REFERENCES courses(id),
category_name VARCHAR(50),
percentage FLOAT
);

CREATE TABLE assignments (
id INTEGER PRIMARY KEY AUTOINCREMENT,
category_id INTEGER REFERENCES categories(id),
assignment_name VARCHAR(50),
total_points FLOAT
);

CREATE TABLE grades (
id INTEGER PRIMARY KEY AUTOINCREMENT,
assignment_id INTEGER REFERENCES assignments(id),
student_id INTEGER REFERENCES students(id),
points FLOAT,
final_grade FLOAT
);

-- Insert students
INSERT INTO students (first_name, last_name, classification, major, email) VALUES
('John', 'Doe', 'Junior', 'Computer Science', 'john.doe@example.com'),
('Jane', 'Qoe', 'Senior', 'Biology', 'jane.doe@example.com'),
('Bob', 'Smith', 'Freshman', 'Psychology', 'bob.smith@example.com');

-- Insert courses
INSERT INTO courses (department, course_number, course_name, semester, year) VALUES
('Computer Science', 'CS101', 'Introduction to Programming', 'Fall', 2022),
('Biology', 'BIO201', 'Genetics', 'Spring', 2023),
('Psychology', 'PSY301', 'Abnormal Psychology', 'Fall', 2022),
('Mathematics', 'MAT202', 'Calculus II', 'Spring', 2023),
('History', 'HIS101', 'World History', 'Fall', 2022);

-- Insert categories
INSERT INTO categories (course_id, category_name, percentage) VALUES
(1, 'Participation', 10),
(1, 'Homework', 20),
(1, 'Tests', 50),
(1, 'Projects', 20);


-- Insert assignments
INSERT INTO assignments (category_id, assignment_name, total_points) VALUES
(1, 'Participation', 10),
(1, 'Homework', 20),
(1, 'Tests', 50),
(1, 'Projects', 20);

-- Insert grades for John Doe
INSERT INTO grades (assignment_id, student_id, points) VALUES
    (1, 1, 8), -- Introduction to Programming - Participation 1
    (2, 1, 88), -- Introduction to Programming - Homework 1
    (3, 1, 95), -- Introduction to Programming - Test 1
    (4, 1, 100); -- Introduction to Programming - Project 1

-- Insert grades for Jane Doe
INSERT INTO grades (assignment_id, student_id, points) VALUES
    (1, 2, 6), -- Introduction to Programming - Participation 1
    (2, 2, 90), -- Introduction to Programming - Homework 1
    (3, 2, 87), -- Introduction to Programming - Test 1
    (4, 2, 95); -- Introduction to Programming - Project 1

-- Insert grades for Bob Smith
INSERT INTO grades (assignment_id, student_id, points) VALUES
    (1, 3, 10), -- Introduction to Programming - Participation 1
    (2, 3, 75), -- Introduction to Programming - Homework 1
    (3, 3, 80), -- Introduction to Programming - Test 1
    (4, 3, 85); -- Introduction to Programming - Project 1




-- Compute the final grade for each student in every class. Reading the grade, catagories and assignemnt tables.
/*

SELECT students.first_name || ' ' || students.last_name AS student_name,
       courses.course_name,
       SUM(grades.points * assignments.total_points / 10 * categories.percentage / 100) AS final_grade
FROM students
JOIN grades ON students.id = grades.student_id
JOIN assignments ON grades.assignment_id = assignments.id
JOIN categories ON assignments.category_id = categories.id
JOIN courses ON categories.course_id = courses.id
GROUP BY students.id, courses.id;

*/

-- QUERIES:
```

▼ Task - Queries:

✓ ~~Show the tables with the contents you've inserted:~~

```
SELECT * FROM students;
SELECT * FROM courses;
SELECT * FROM categories;
SELECT * FROM assignments;
SELECT * FROM grades;
```

✓ ~~Compute the average/highest/lowest score of an assignment:~~

```
--Average Score of an assignemnt
SELECT AVG(points) AS average_score
FROM grades
WHERE assignment_id = 3;

-- Highest score of an assignment
SELECT MAX(points) AS highest_score
FROM grades
WHERE assignment_id = 3;

-- Lowest score of an assignment
SELECT MIN(points) AS lowest_score
FROM grades
WHERE assignment_id = 3;
```

- [x] ~~List all of the students in a given course:~~

```
SELECT students.first_name || ' ' || students.last_name AS student_name
FROM students
JOIN grades ON students.id = grades.student_id
JOIN assignments ON grades.assignment_id = assignments.id
JOIN categories ON assignments.category_id = categories.id
JOIN courses ON categories.course_id = courses.id
WHERE courses.course_name = 'Introduction to Programming';
```

- [x] ~~List all of the students in a course and all of their scores on every assignment:~~

```
SELECT students.first_name || ' ' || students.last_name AS student_name,
courses.course_name,
categories.category_name,
assignments.assignment_name,
grades.points
FROM students
JOIN grades ON students.id = grades.student_id
JOIN assignments ON grades.assignment_id = assignments.id
JOIN categories ON assignments.category_id = categories.id
JOIN courses ON categories.course_id = courses.id
WHERE courses.course_name = 'Introduction to Programming';
```

- [x] ~~Add an assignment to a course:~~

```
INSERT INTO assignments (category_id, assignment_name, total_points)
VALUES (1, 'Extra Participation', 5);
```

- [x] ~~Change the percentages of the categories for a course:~~

```
UPDATE categories
SET percentage = CASE category_name
WHEN 'Participation' THEN 15
WHEN 'Homework' THEN 15
WHEN 'Tests' THEN 50
WHEN 'Projects' THEN 20
END
WHERE course_id = 1;
```

- [x] ~~Add 2 points to the score of each student on an assignment:~~

```
UPDATE grades
SET points = points + 2
WHERE assignment_id = 2;
```

- [x] ~~Add 2 points just to those students whose last name contains a 'Q':~~

```
UPDATE grades
SET points = points + 2
WHERE assignment_id = 2
AND student_id IN (SELECT id FROM students WHERE last_name LIKE '%Q%');
```

- [x] ~~Compute the grade for a student:~~

```
SELECT students.first_name || ' ' || students.last_name AS student_name,
       courses.course_name,
       SUM(grades.points * assignments.total_points / 10 * categories.percentage / 100) AS final_grade
FROM students
JOIN grades ON students.id = grades.student_id
JOIN assignments ON grades.assignment_id = assignments.id
JOIN categories ON assignments.category_id = categories.id
JOIN courses ON categories.course_id = courses.id
WHERE students.first_name = 'John' AND students.last_name = 'Doe'
GROUP BY students.id, courses.id;
```

- ✓ ~~Compute the grade for a student, where the lowest score for a given category is dropped:~~

```
SELECT students.first_name || ' ' || students.last_name AS student_name,
       courses.course_name,
       SUM((grades.points - minGrades.min_points) * assignments.total_points / 10 * categories.percentage / 100) AS final_grade
FROM students
JOIN grades ON students.id = grades.student_id
JOIN assignments ON grades.assignment_id = assignments.id
JOIN categories ON assignments.category_id = categories.id
JOIN courses ON categories.course_id = courses.id
JOIN (
  SELECT grades.student_id, assignments.category_id, MIN(grades.points) AS min_points
  FROM grades
  JOIN assignments ON grades.assignment_id = assignments.id
  GROUP BY grades.student_id, assignments.category_id
) minGrades ON (
  minGrades.student_id = grades.student_id
  AND minGrades.category_id = categories.id
)
WHERE students.first_name = 'John' AND students.last_name = 'Doe'
GROUP BY students.id, courses.id;
```