A. 1 professor and 3 students registered their accounts on this website
    - Write 4 INSERT statements for this scenario.
        - **For ALL INSERT statements** (in this scenario and the rest scenarios), include the **id column** (in normal practice, we don't insert the id column but let the database generate them automatically; here we're only doing this so that later on we can easily refer to these inserted records).
    - Use any dummy data (for example, use email1@example.com as email address)

Queries:

```
-- Professor's Account
INSERT INTO Account (id, account_type, email, password)
VALUES (1, 'professor', 'professor1@example.com', 'passwordP1');

-- Student Accounts
INSERT INTO Account (id, account_type, email, password)
VALUES (2, 'student', 'student1@example.com', 'passwordST1');

INSERT INTO Account (id, account_type, email, password)
VALUES (3, 'student', 'student2@example.com', 'passwordST2');

INSERT INTO Account (id, account_type, email, password)
VALUES (4, 'student', 'student3@example.com', 'passwordST3');
```

B. The professor created a course and the 3 students enrolled in his course.
    - For the student_id/professor_id column, use the ids you previously inserted in scenario A.
    - Again, for any INSERT statements, include the id column.

Queries:

```
-- Professor creates a course
INSERT INTO Course (id, professor_id, name, description, start_date, end_date)
VALUES (1, 1, 'Introduction to SQL', 'Learn the basics of SQL', '2023-01-01', '2023-04-30');

-- Students enroll in the course
INSERT INTO StudentCourse (id, student_id, course_id, enrollment_date)
VALUES (1, 2, 1, '2023-01-02');

INSERT INTO StudentCourse (id, student_id, course_id, enrollment_date)
VALUES (2, 3, 1, '2023-01-02');

INSERT INTO StudentCourse (id, student_id, course_id, enrollment_date)
VALUES (3, 4, 1, '2023-01-02');
```

C. The professor then created 2 assignments (titles are Homework1, Homework2) for this course.

Queries:

```
-- Professor creates assignments for the course
INSERT INTO Assignment (id, course_id, title, description, points, available_date, due_date)
VALUES (1, 1, 'Homework1', 'Complete the given SQL problems.', 100, '2023-01-03', '2023-01-10');

INSERT INTO Assignment (id, course_id, title, description, points, available_date, due_date)
VALUES (2, 1, 'Homework2', 'Advanced SQL queries.', 100, '2023-01-11', '2023-01-18');
```

D. One student (randomly choose any student) submitted twice for **Homework1** and left a comment.
   - Note that you also need to create the corresponding StudentAssignment record.
   - The StudentAssignment records are designed to be created on the fly. Meaning they're only created when a student submits/comments an assignment but there is no corresponding StudentAssignment record available.

Queries:

```
-- Creating StudentAssignment record
INSERT INTO StudentAssignment (id, student_id, assignment_id, grade, grade_time)
VALUES (1, 2, 1, NULL, NULL);

-- Student submissions for Homework1
INSERT INTO Submission (id, studentassignment_id, content, create_time)
VALUES (1, 1, 'Submission content 1', '2023-01-04');

INSERT INTO Submission (id, studentassignment_id, content, create_time)
VALUES (2, 1, 'Submission content 2', '2023-01-05');

-- Student comments on Homework1
INSERT INTO Comment (id, account_id, studentassignment_id, content, create_time)
VALUES (1, 2, 1, 'This was difficult assignment!', '2023-01-05');
```

E. The professor opened the website.
   - He first navigated to the course's assignment management page (write a SELECT statement listing the assignments in this course).
   - He then navigated to Homework1's StudentAssignment page (write a SELECT statement listing the StudentAssignments for Homework1; in this case, it's supposed to return only one result as only one of the student had ever submitted/commented).
   - He then navigated to the student's (who submitted twice for Homework1) StudentAssignment page for Homework1

- Write a SELECT statement listing the **submissions** for this StudentAssignment.
- Write another SELECT statement listing the **comments** for this StudentAssignment.

Queries:

```
-- Listing assignments in the course
SELECT * FROM Assignment WHERE course_id = 1;

-- Listing StudentAssignments for Homework1
SELECT * FROM StudentAssignment WHERE assignment_id = 1;

-- Listing the submissions for the student's StudentAssignment for Homework1
SELECT * FROM Submission WHERE studentassignment_id = 1;

-- Listing the comments for the student's StudentAssignment for Homework1
SELECT * FROM Comment WHERE studentassignment_id = 1;
```

F. The professor graded this student's Homework1 and also left a comment.

```
-- Professor grades the student's Homework1
UPDATE StudentAssignment SET grade = 100, grade_time = '2023-01-06' WHERE id = 1;

-- Professor leaves a comment on the student's Homework1
INSERT INTO Comment (id, account_id, studentassignment_id, content, create_time) VALUES (2,
1, 1, 'Excellent work! Keep it up!', '2023-01-06');
```