CS F212 - Database Systems
Project Number 7 - Assignment Management System

Submitted to:
Dr. Amit Dua

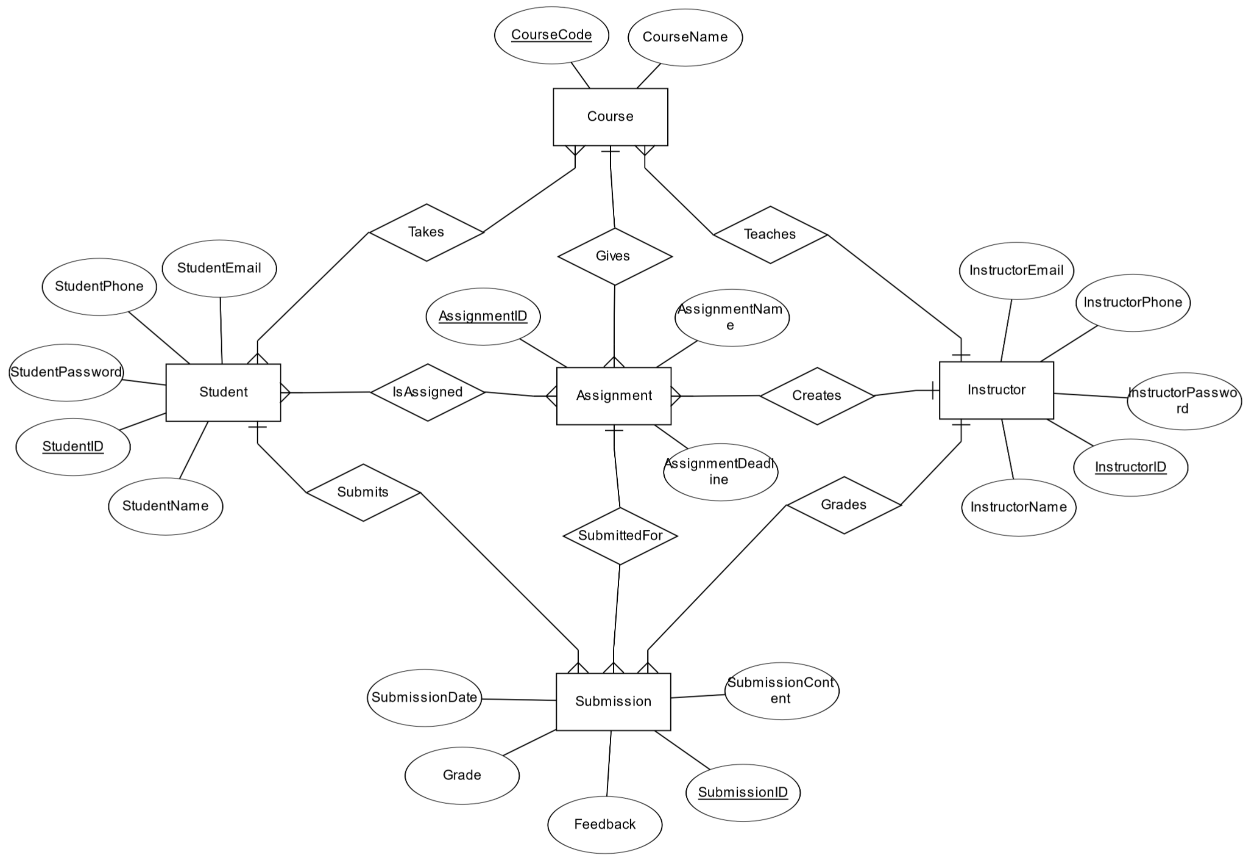
Submission Date:
11 April, 2023

Team Members:
Bhaskar Iyer - 2021A7PS2223P
Nishant Luthra - 2021A7PS2420P

BIRLA INSTITUTE OF TECHNOLOGY AND SCIENCE PILANI
PILANI, RAJASTHAN - 333031

Video Link
Nishant Luthra -
https://drive.google.com/file/d/1Y-BUI3gV1KihfX_aLS-_O98VIW-N1Gw0/view?usp=sharing

ER Diagram



Explanation

The ER Diagram consists of five entities -

1. Student - Each student has a StudentID as the primary key and various personal detail attributes such as name, email, password, etc.
2. Instructor - Each instructor (equivalent to faculty member) has an InstructorID as the primary key and various personal details similar to the Student entity.
3. Assignment - Each assignment consists of a primary AssignmentID, a name and a deadline attribute.
4. Course - Each course has a primary CourseCode and a name attribute.
5. Submission - Each submission has a primary SubmissionID and a grade, content, date and feedback attribute.

The Relationship Takes represents that a student can enroll in many courses and many students can be enrolled in a course.

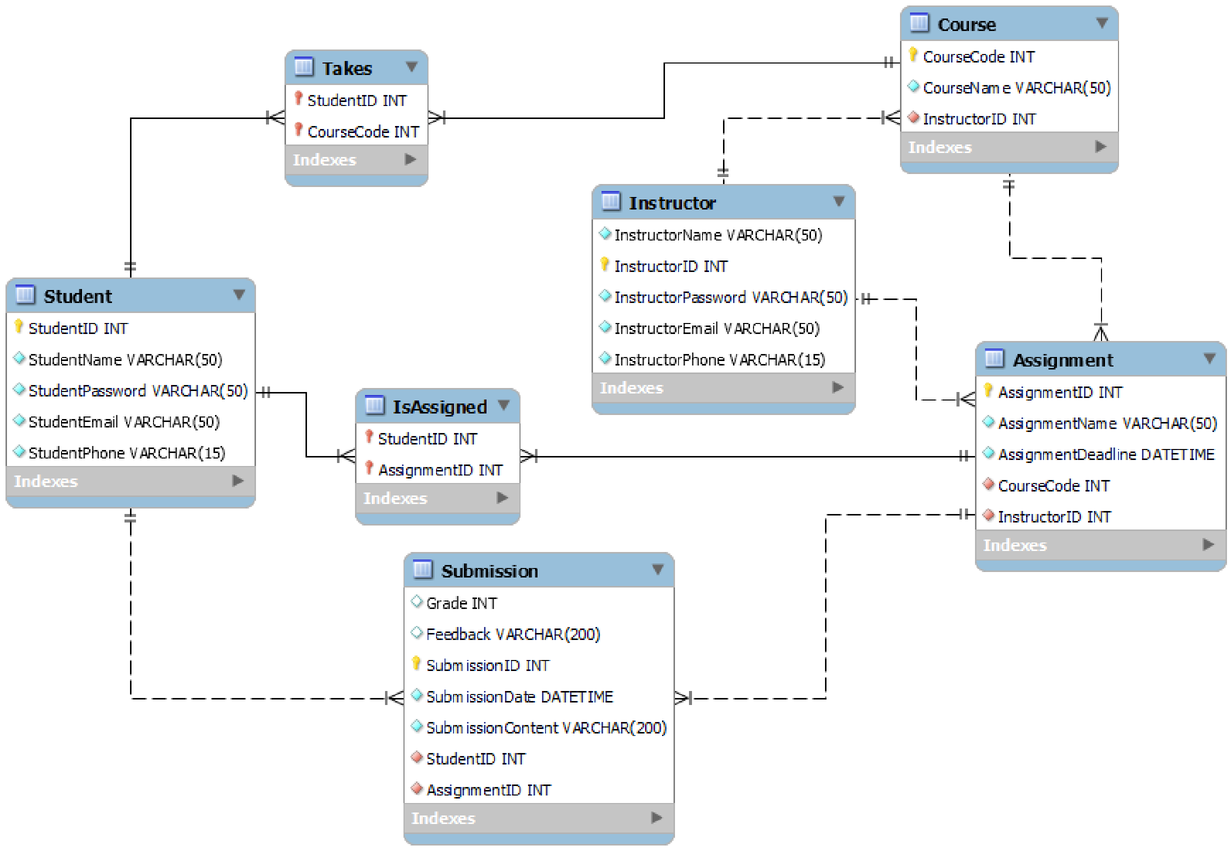
The Relationship Teaches represents that an instructor can teach many courses but a course can only be taught by a single instructor.

A Student can submit multiple submissions, indicated by the Submits Relationship and an instructor can grade multiple submissions, indicated by the Grades Relationship.

An instructor can create multiple assignments, indicated by the Creates Relationship and multiple students can be assigned multiple assignments, indicated by the IsAssigned Relationship.

A course can give out multiple assignments and each assignment can have multiple submissions, as shown by the Gives and SubmittedFor Relationships respectively.

Relational Schema



Explanation

The final relational schema consists of 7 tables, one for each entity and two additional tables - Takes and IsAssigned.
The table Takes stores records of which student is enrolled in which course in the form of (StudentID, CourseCode) tuples.
The table IsAssigned stores records of which student is assigned which assignments in the form of (StudentID, AssignmentID) tuples.

The schema consists of several foreign keys to ensure the integrity of data -

- Course has an InstructorID as a foreign key
- Assignment has CourseCode as a foreign key
- Submission has StudentID and AssignmentID as foreign keys
- Takes as well as IsAssigned have both of their attributes as foreign keys.

Normalization

In the relational diagram shown above, it is already in 1NF as none of the attributes are multivalued or composite.

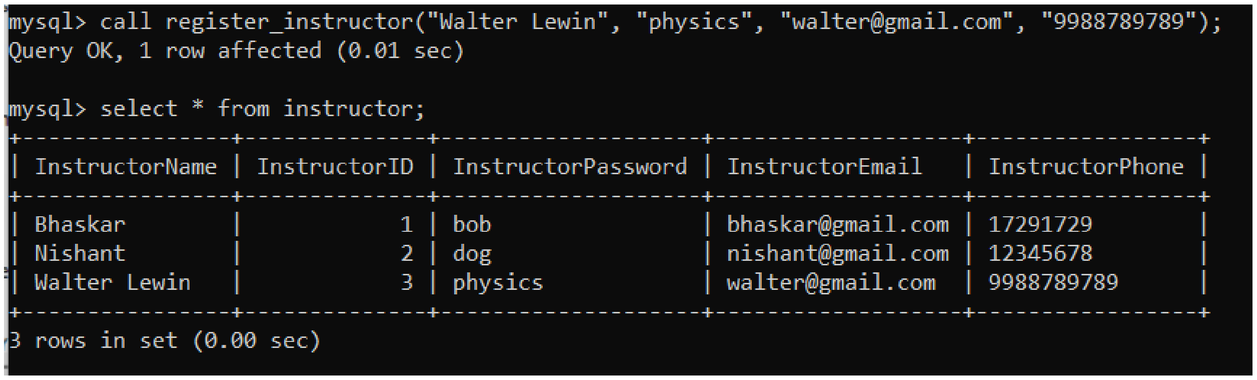
For 2NF, the requirement is that every non-prime attribute must be fully functionally dependent on the key and not be partially dependent on some of its attributes. The schema is also in 2NF as all of the primary keys consist of only one attribute, so partial functional dependencies cannot exist.

In order for the database to be in 3NF, transitive functional dependencies are forbidden - thus, functional dependencies of the form X->Y and Y->Z are forbidden, where Y is a non-prime attribute. In the given schema, the Assignment relation consists of InstructorID and CourseCode attributes. Since the CourseCode uniquely determines the instructorID of a course and thus for an assignment, it is a functional dependency. However, since the CourseCode is dependent on the AssignmentID primary key in the Assignment relation, this is a violation of the 3NF rule and thus, the InstructorID attribute must be removed from the Assignment relation to normalize the database into 3NF.

Thus, the normalized database remains the same except that the InstructorID foreign key is removed from the Assignment relation.

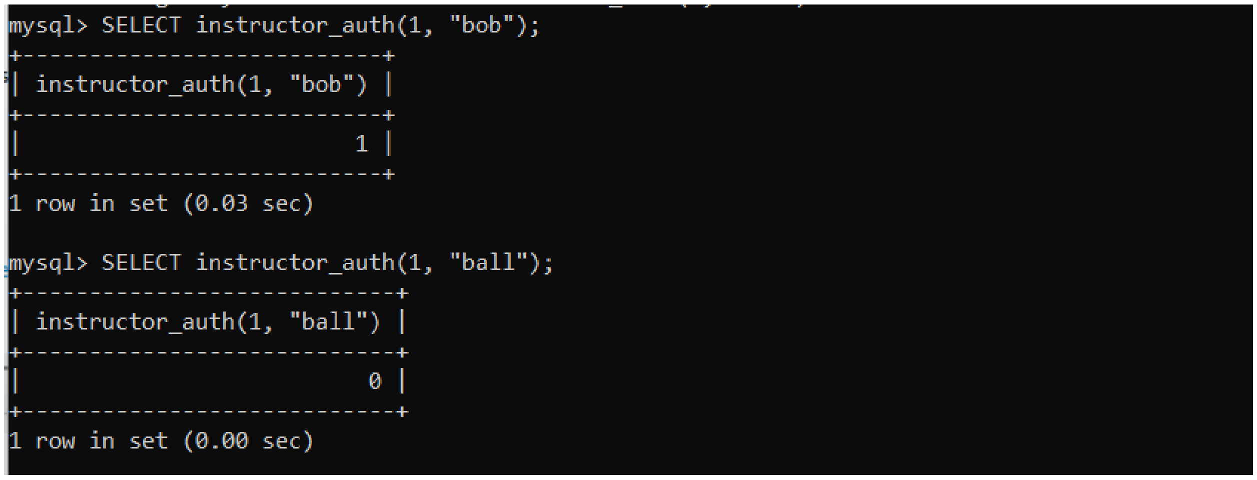
SQL Queries - Output Screenshots

1. **Insert records for teacher**



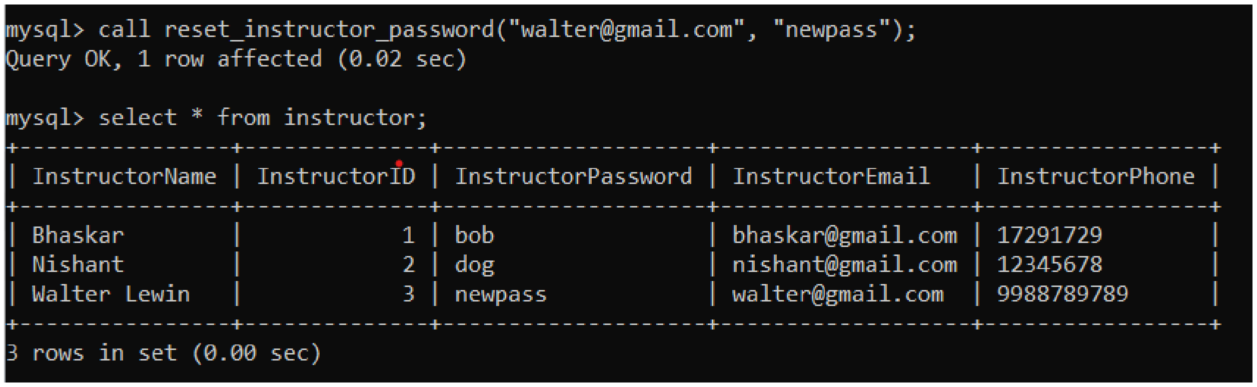
```
mysql> call register_instructor("Walter Lewin", "physics", "walter@gmail.com", "9988789789");
Query OK, 1 row affected (0.01 sec)

mysql> select * from instructor;
+----------------+--------------+--------------------+-------------------+-----------------+
| InstructorName | InstructorID | InstructorPassword | InstructorEmail   | InstructorPhone |
+----------------+--------------+--------------------+-------------------+-----------------+
| Bhaskar        |            1 | bob                | bhaskar@gmail.com | 17291729        |
| Nishant        |            2 | dog                | nishant@gmail.com | 12345678        |
| Walter Lewin   |            3 | physics            | walter@gmail.com  | 9988789789      |
+----------------+--------------+--------------------+-------------------+-----------------+
3 rows in set (0.00 sec)
```

2. **Authentication for teacher**



```
mysql> SELECT instructor_auth(1, "bob");
+--------------------------+
| instructor_auth(1, "bob") |
+--------------------------+
|                        1 |
+--------------------------+
1 row in set (0.03 sec)

mysql> SELECT instructor_auth(1, "ball");
+---------------------------+
| instructor_auth(1, "ball") |
+---------------------------+
|                         0 |
+---------------------------+
1 row in set (0.00 sec)
```

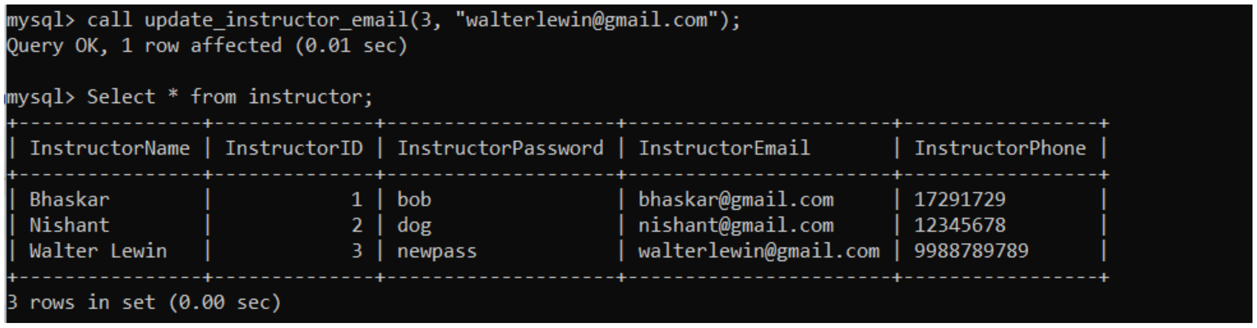
3. **Reset instructor password**



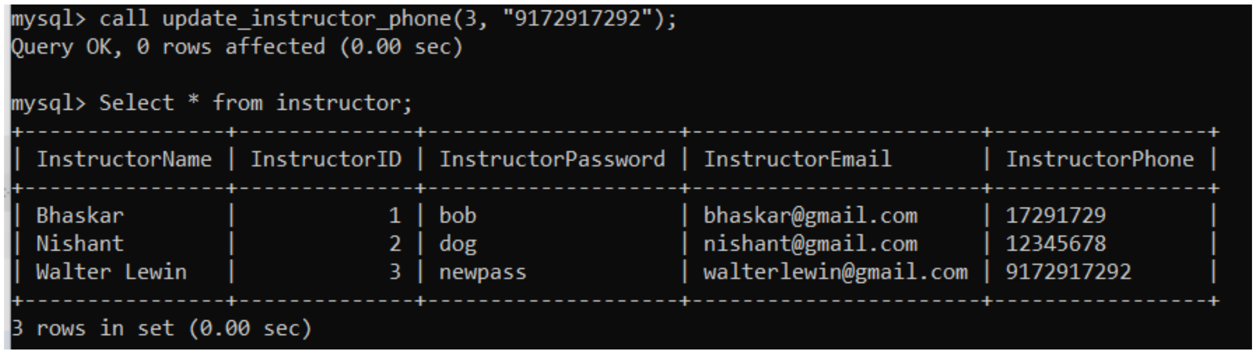
```
mysql> call reset_instructor_password("walter@gmail.com", "newpass");
Query OK, 1 row affected (0.02 sec)

mysql> select * from instructor;
+----------------+--------------+--------------------+-------------------+-----------------+
| InstructorName | InstructorID | InstructorPassword | InstructorEmail   | InstructorPhone |
+----------------+--------------+--------------------+-------------------+-----------------+
| Bhaskar        |            1 | bob                | bhaskar@gmail.com | 17291729        |
| Nishant        |            2 | dog                | nishant@gmail.com | 12345678        |
| Walter Lewin   |            3 | newpass            | walter@gmail.com  | 9988789789      |
+----------------+--------------+--------------------+-------------------+-----------------+
3 rows in set (0.00 sec)
```

4. **Update instructor details**

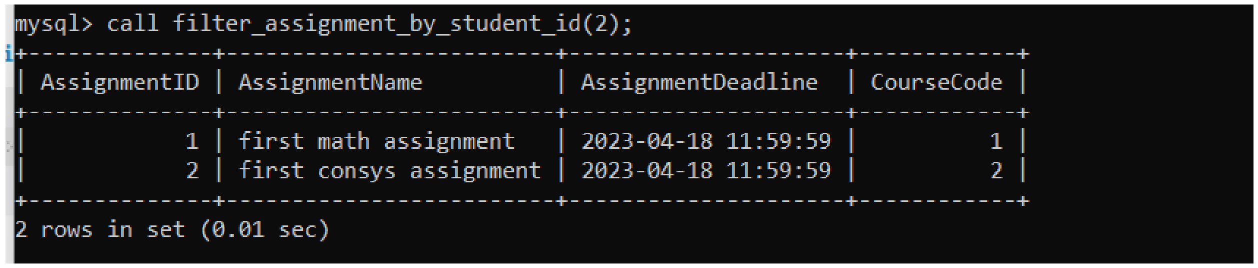
a) **Update Instructor Email -**


```
mysql> call update_instructor_email(3, "walterlewin@gmail.com");
Query OK, 1 row affected (0.01 sec)

mysql> Select * from instructor;
+----------------+--------------+--------------------+----------------------+------------------+
| InstructorName | InstructorID | InstructorPassword | InstructorEmail      | InstructorPhone |
+----------------+--------------+--------------------+----------------------+------------------+
| Bhaskar        |            1 | bob                | bhaskar@gmail.com    | 17291729         |
| Nishant        |            2 | dog                | nishant@gmail.com    | 12345678         |
| Walter Lewin   |            3 | newpass            | walterlewin@gmail.com | 9988789789      |
+----------------+--------------+--------------------+----------------------+------------------+
3 rows in set (0.00 sec)
```

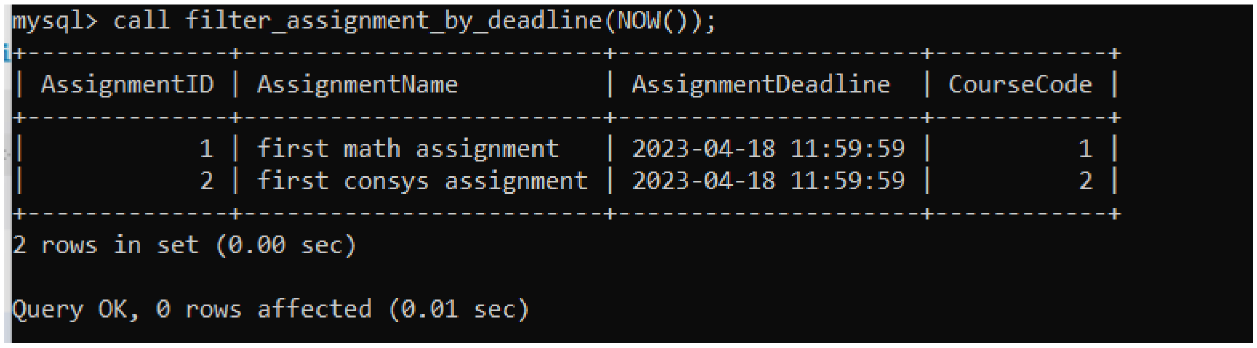
b) **Update Instructor Phone -**


```
mysql> call update_instructor_phone(3, "9172917292");
Query OK, 0 rows affected (0.00 sec)

mysql> Select * from instructor;
+----------------+--------------+--------------------+-----------------------+-----------------+
| InstructorName | InstructorID | InstructorPassword | InstructorEmail       | InstructorPhone |
+----------------+--------------+--------------------+-----------------------+-----------------+
| Bhaskar        |            1 | bob                | bhaskar@gmail.com     | 17291729        |
| Nishant        |            2 | dog                | nishant@gmail.com     | 12345678        |
| Walter Lewin   |            3 | newpass            | walterlewin@gmail.com | 9172917292      |
+----------------+--------------+--------------------+-----------------------+-----------------+
3 rows in set (0.00 sec)
```

5. **Filtering assignments by student ID**

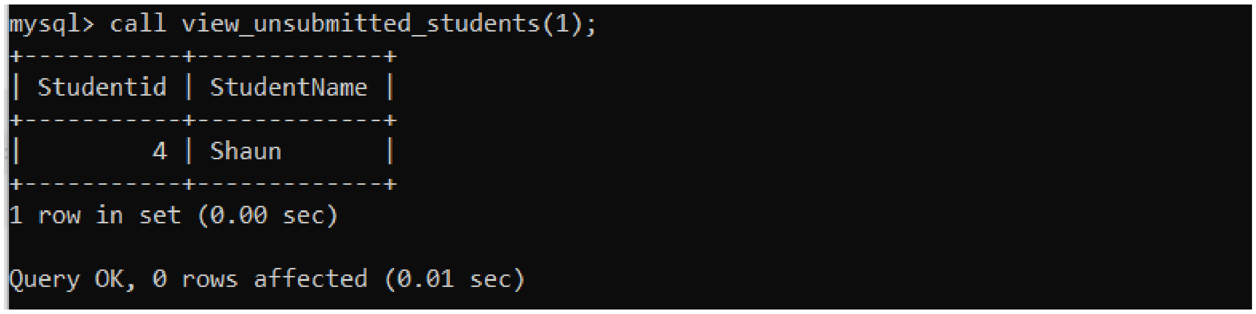

```
mysql> call filter_assignment_by_student_id(2);
+--------------+--------------------------+---------------------+------------+
| AssignmentID | AssignmentName           | AssignmentDeadline  | CourseCode |
+--------------+--------------------------+---------------------+------------+
|            1 | first math assignment    | 2023-04-18 11:59:59 |          1 |
|            2 | first consys assignment  | 2023-04-18 11:59:59 |          2 |
+--------------+--------------------------+---------------------+------------+
2 rows in set (0.01 sec)
```

6. **Filtering assignments by deadline**



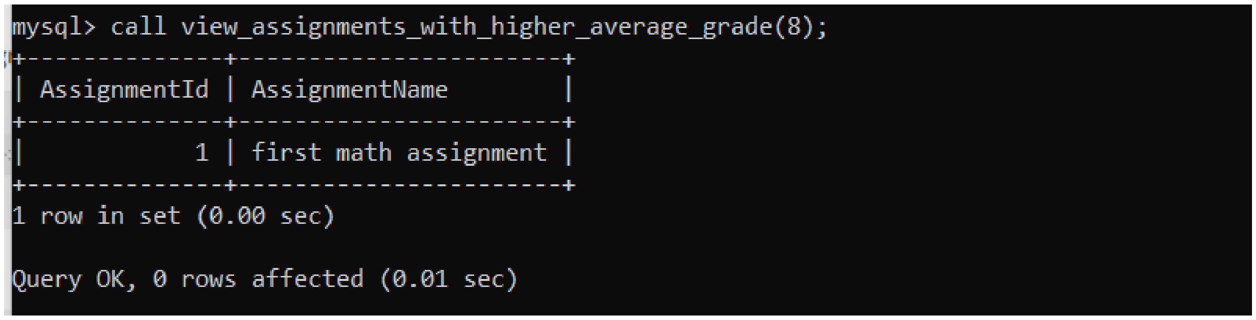
```
mysql> call filter_assignment_by_deadline(NOW());
+--------------+------------------------+---------------------+------------+
| AssignmentID | AssignmentName         | AssignmentDeadline  | CourseCode |
+--------------+------------------------+---------------------+------------+
|            1 | first math assignment  | 2023-04-18 11:59:59 |          1 |
|            2 | first consys assignment | 2023-04-18 11:59:59 |          2 |
+--------------+------------------------+---------------------+------------+
2 rows in set (0.00 sec)

Query OK, 0 rows affected (0.01 sec)
```

7. **Unsubmitted students**



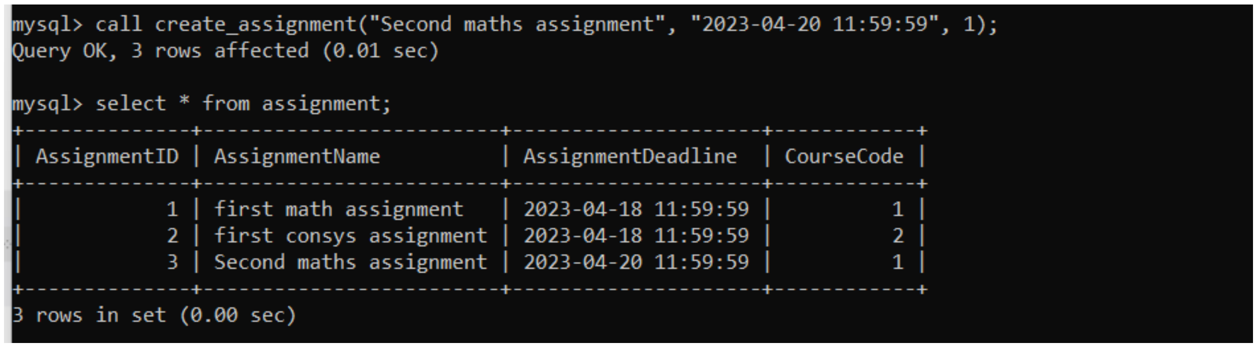
```
mysql> call view_unsubmitted_students(1);
+-----------+-------------+
| Studentid | StudentName |
+-----------+-------------+
|         4 | Shaun       |
+-----------+-------------+
1 row in set (0.00 sec)

Query OK, 0 rows affected (0.01 sec)
```

8. **Filter assignments by average grade**



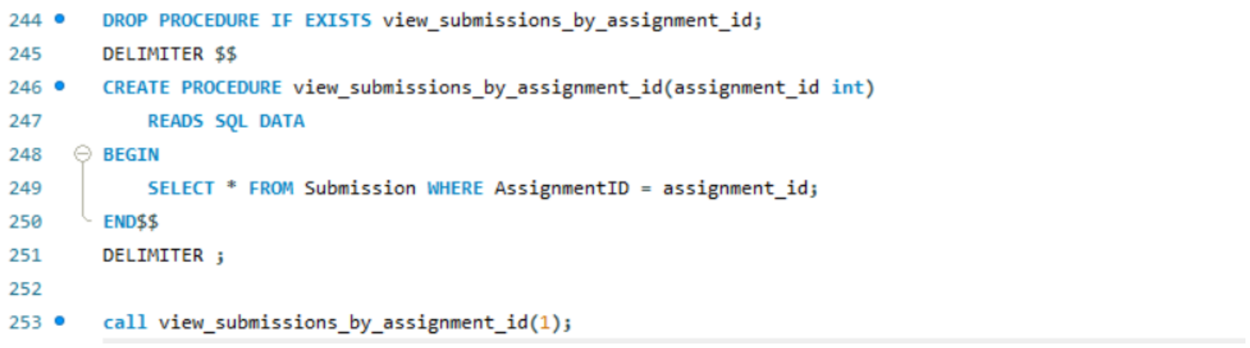
```
mysql> call view_assignments_with_higher_average_grade(8);
+--------------+-----------------------+
| AssignmentId | AssignmentName        |
+--------------+-----------------------+
|            1 | first math assignment |
+--------------+-----------------------+
1 row in set (0.00 sec)

Query OK, 0 rows affected (0.01 sec)
```

9. Create assignment



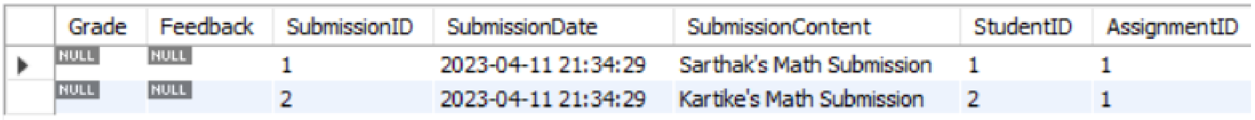
```
mysql> call create_assignment("Second maths assignment", "2023-04-20 11:59:59", 1);
Query OK, 3 rows affected (0.01 sec)

mysql> select * from assignment;
+--------------+-------------------------+---------------------+------------+
| AssignmentID | AssignmentName          | AssignmentDeadline  | CourseCode |
+--------------+-------------------------+---------------------+------------+
|            1 | first math assignment   | 2023-04-18 11:59:59 |          1 |
|            2 | first consys assignment | 2023-04-18 11:59:59 |          2 |
|            3 | Second maths assignment | 2023-04-20 11:59:59 |          1 |
+--------------+-------------------------+---------------------+------------+
3 rows in set (0.00 sec)
```

10. View Submissions by assignment ID - Procedure view_submissions_by_assignment_id



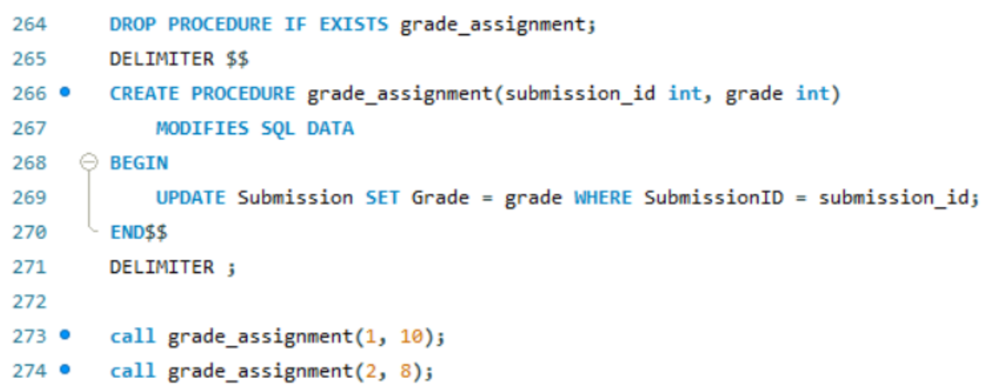
```
DROP PROCEDURE IF EXISTS view_submissions_by_assignment_id;
DELIMITER $$
CREATE PROCEDURE view_submissions_by_assignment_id(assignment_id int)
    READS SQL DATA
BEGIN
    SELECT * FROM Submission WHERE AssignmentID = assignment_id;
END$$
DELIMITER ;

call view_submissions_by_assignment_id(1);
```

Output -

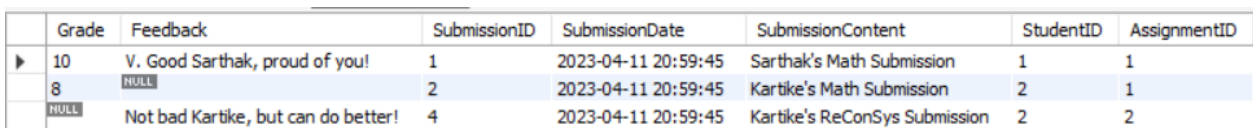


Grade	Feedback	SubmissionID	SubmissionDate	SubmissionContent	StudentID	AssignmentID
NULL	NULL	1	2023-04-11 21:34:29	Sarthak's Math Submission	1	1
NULL	NULL	2	2023-04-11 21:34:29	Kartike's Math Submission	2	1

11. Grading assignments by instructors - Procedure grade_assignment

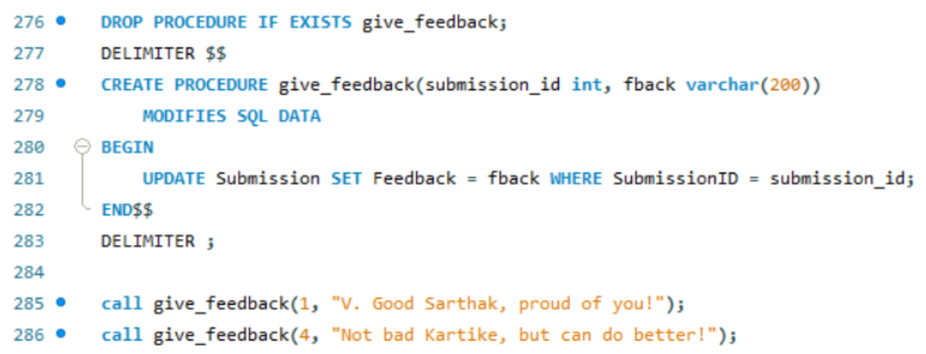

```sql
DROP PROCEDURE IF EXISTS grade_assignment;
DELIMITER $$
CREATE PROCEDURE grade_assignment(submission_id int, grade int)
    MODIFIES SQL DATA
BEGIN
    UPDATE Submission SET Grade = grade WHERE SubmissionID = submission_id;
END$$
DELIMITER ;

call grade_assignment(1, 10);
call grade_assignment(2, 8);
```

Output -

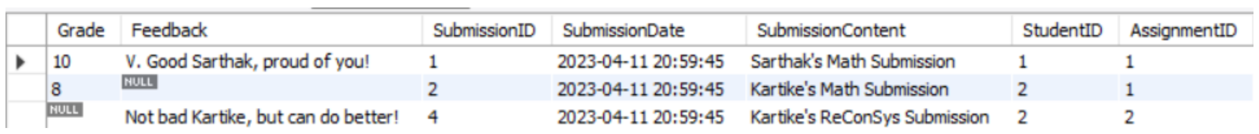


Grade	Feedback	SubmissionID	SubmissionDate	SubmissionContent	StudentID	AssignmentID
10	V. Good Sarthak, proud of you!	1	2023-04-11 20:59:45	Sarthak's Math Submission	1	1
8	NULL	2	2023-04-11 20:59:45	Kartike's Math Submission	2	1
NULL	Not bad Kartike, but can do better!	4	2023-04-11 20:59:45	Kartike's ReConSys Submission	2	2

12. Giving feedback for assignments by instructors - Procedure give_feedback

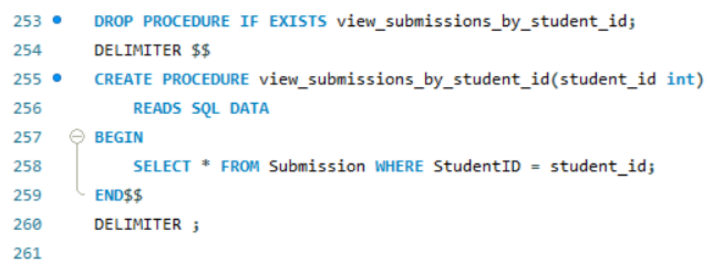

```sql
DROP PROCEDURE IF EXISTS give_feedback;
DELIMITER $$
CREATE PROCEDURE give_feedback(submission_id int, fback varchar(200))
    MODIFIES SQL DATA
BEGIN
    UPDATE Submission SET Feedback = fback WHERE SubmissionID = submission_id;
END$$
DELIMITER ;

call give_feedback(1, "V. Good Sarthak, proud of you!");
call give_feedback(4, "Not bad Kartike, but can do better!");
```

Output -



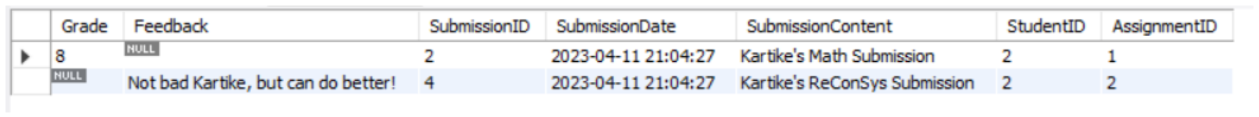
Grade	Feedback	SubmissionID	SubmissionDate	SubmissionContent	StudentID	AssignmentID
10	V. Good Sarthak, proud of you!	1	2023-04-11 20:59:45	Sarthak's Math Submission	1	1
8	NULL	2	2023-04-11 20:59:45	Kartike's Math Submission	2	1
NULL	Not bad Kartike, but can do better!	4	2023-04-11 20:59:45	Kartike's ReConSys Submission	2	2

13. Viewing the feedback and grades for a student's submissions - Procedure view_submissions_by_student_id



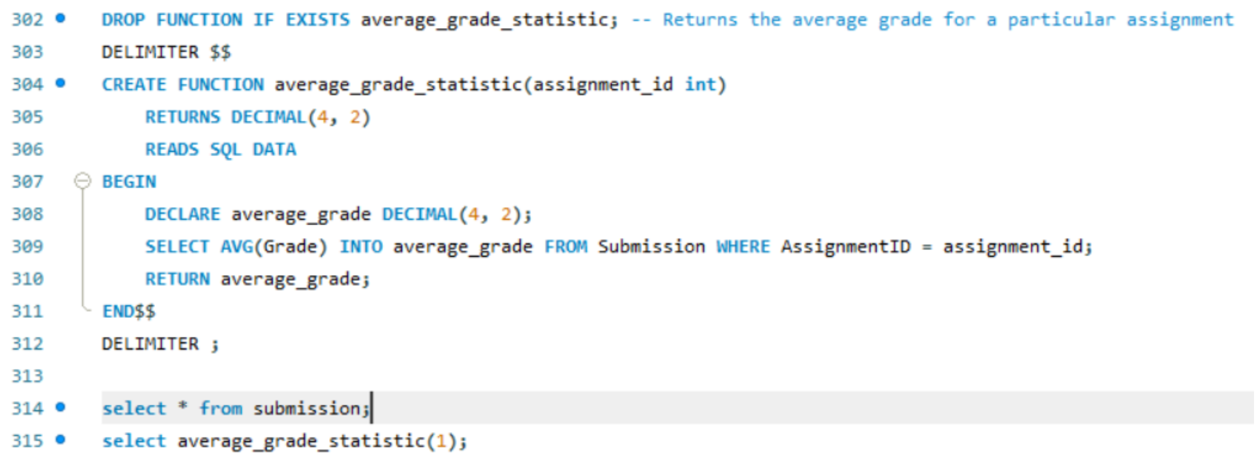
```sql
DROP PROCEDURE IF EXISTS view_submissions_by_student_id;
DELIMITER $$
CREATE PROCEDURE view_submissions_by_student_id(student_id int)
    READS SQL DATA
BEGIN
    SELECT * FROM Submission WHERE StudentID = student_id;
END$$
DELIMITER ;
```

Output -



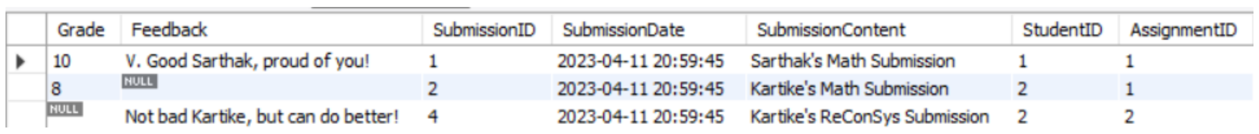
Grade	Feedback	SubmissionID	SubmissionDate	SubmissionContent	StudentID	AssignmentID
8	NULL	2	2023-04-11 21:04:27	Kartike's Math Submission	2	1
NULL	Not bad Kartike, but can do better!	4	2023-04-11 21:04:27	Kartike's ReConSys Submission	2	2

14. Finding the Average Grade of a particular course - Function average_grade_statistic



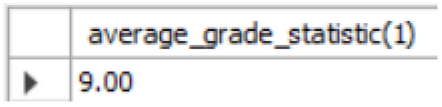
```sql
DROP FUNCTION IF EXISTS average_grade_statistic; -- Returns the average grade for a particular assignment
DELIMITER $$
CREATE FUNCTION average_grade_statistic(assignment_id int)
    RETURNS DECIMAL(4, 2)
    READS SQL DATA
BEGIN
    DECLARE average_grade DECIMAL(4, 2);
    SELECT AVG(Grade) INTO average_grade FROM Submission WHERE AssignmentID = assignment_id;
    RETURN average_grade;
END$$
DELIMITER ;

select * from submission;
select average_grade_statistic(1);
```

Output -



Grade	Feedback	SubmissionID	SubmissionDate	SubmissionContent	StudentID	AssignmentID
10	V. Good Sarthak, proud of you!	1	2023-04-11 20:59:45	Sarthak's Math Submission	1	1
8	NULL	2	2023-04-11 20:59:45	Kartike's Math Submission	2	1
NULL	Not bad Kartike, but can do better!	4	2023-04-11 20:59:45	Kartike's ReConSys Submission	2	2

The first two submissions are for the assignment with ID 1 and their grades are 10 and 8 respectively, thus the average grade should return 9.00.



average_grade_statistic(1)
9.00

The output is as expected.

Front-End Application

A GUI Application has been developed for this project in Python using the Tkinter library for GUI tools and the mysql-connector library for establishing a connection to the database.

Setup Instructions

1. Execute the SQL Script file submitted to generate the database for the project.
2. In the submitted Python file, change the value of the variable databaseRootPassword (line 9) to your system's MySQL root password in order for the application to connect to the database.
3. Run the python file submitted to view the GUI application.