**Database Design Guide**

This guide will help the student to create a database on the Student Management System. It will help to manage the below functionalities.

* Student details
* Instructor details
* Course details
* Enrollment details
* Score
* Feedback

We will use MySQL as the DBMS to create the database and its related operations.

**1. Introduction to MySQL**

MySQL is an open-source relational database management system (RDBMS) that uses structured query language (SQL) to manage and manipulate data in a database. It is widely used for various applications, from small web applications to large enterprise systems.

MySQL's key features include:

* Scalability: Capable of handling large amounts of data and concurrent connections.
* Flexibility: Supports various data types and storage engines.
* Performance: Optimized for speed and efficiency.
* Reliability: Known for its stability and robustness.

**2. Installation of MySQL**

MySQL can be installed on various operating systems, including Windows, macOS, and Linux. Here are the general steps to install MySQL:

**Windows:**

* Download the MySQL installer from the official website.

<https://dev.mysql.com/downloads/installer/>

* Run the installer and follow the on-screen instructions.
* Choose the installation type (Typical, Complete, or Custom). Recommended Custom.
* Set a root password for the MySQL server.

**3. E-R Diagram (ERD)**

An Entity-Relationship Diagram (ERD) is a visual representation of the data model that shows the entities, attributes, relationships between entities, and cardinality. ERDs are commonly used in database design to help developers and stakeholders understand the structure and relationships within a database.

**Identify Entities**

* Start by identifying the main entities in your system. These are the objects or concepts about which you want to store data.
* Each entity should correspond to a table in your database.

**Define Attributes**

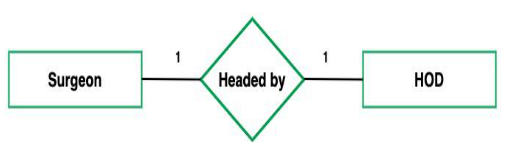
* For each entity, list the attributes (properties or fields) that describe it.
* These attributes will become columns in the corresponding database table.

**Identify Relationships**

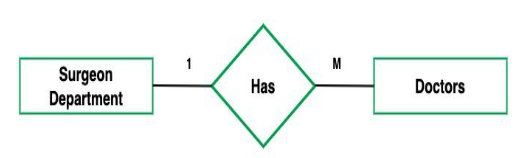
* Determine how entities are related to each other. There are three types of relationships: one-to-one (1:1), one-to-many (1:N), and many-to-many (N:M).
* Represent these relationships using lines connecting the entities.

Let’s see a few examples of relationships:

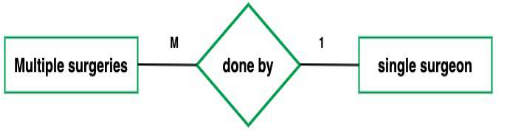
**One to One**



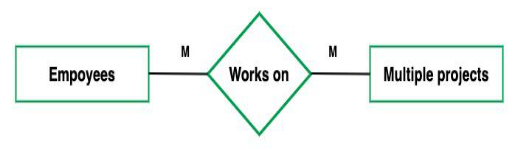
**One to Many**



**Many to One**



**Many to Many**



**Cardinality Notation**

Cardinality represents the number of times an entity of an entity set participates in a relationship set. Or we can say that the cardinality of a relationship is the number of tuples (rows) in a relationship.

* Use notation (such as Crow's Foot Notation or Chen Notation) to indicate the cardinality of each relationship.
* Cardinality describes how many instances of one entity are related to how many instances of another entity.
* Common notations include:
* One (1)
* Zero or one (0..1)
* Many (N)
* Zero or many (0..N)

**Optional:**

**Add Attributes and Constraints**

* Include additional information in your ERD, such as primary keys, foreign keys, and constraints (e.g., unique constraints).

**Create the Diagram**

* Use specialized diagramming software or tools (e.g., Lucidchart, draw.io, or even pen and paper) to create your ERD.

**Refine and Review:**

* Review your ERD with stakeholders and team members to ensure it accurately represents the data model and relationships. Make any necessary refinements.

Let’s identify the entities of the Student management system

1. Student
2. Course
3. Instructor
4. Enrollment
5. Score
6. Feedback

\*\*\* Now let’s identify the attributes and relationships of each entity for the Student Management System.

**Student**

* **Attributes:**

StudentID (Primary Key)

FirstName

LastName

DateOfBirth

Gender

Email

Phone

* **Relationships:**

One **Student** can enroll in more than one **Course** (One-to-Many)

**Course**

* **Attributes:**

CourseID (Primary Key)

CourseTitle

Credits

* **Relationships:**

Many **Course** is taught by one **Instructor** (**Many-to-One**)

**Instructor**

* **Attributes:**

InstructorID (Primary Key)

FirstName

LastName

Email

* **Relationships:**

One **Instructor** teaches many **Courses** (**One-to-Many**)

One **Instructor** has many **Students** (**One-to-Many**)

**Enrollment**

* **Attributes:**

EnrollmentID (Primary Key)

EnrollmentDate

StudentID(Foreign key)

CourseID(Foreign Key)

              InstructorID(Foreign key)

* **Relationships:**

One **Student** maps to Many **Enrolment ids** (**One-to-Many**)

Many **Enrolment ids** map to one **Course** (**Many-to-One**)

**Score**

* **Attributes**:

ScoreID (Primary Key)

CourseID (Foreign key)

StudentID (Foreign Key)

DateOfExam

CreditObtained

* **Relationships:**

Many **ScoreIDs** will map to one **Student  (Many-to-one)**

Many **ScoresIDs** will map to one **Course (Many-to-one)**

**Feedback**

* **Attributes**:

FeedbackID (Primary Key)

StudentID (Foreign key)

Date

InstructorName

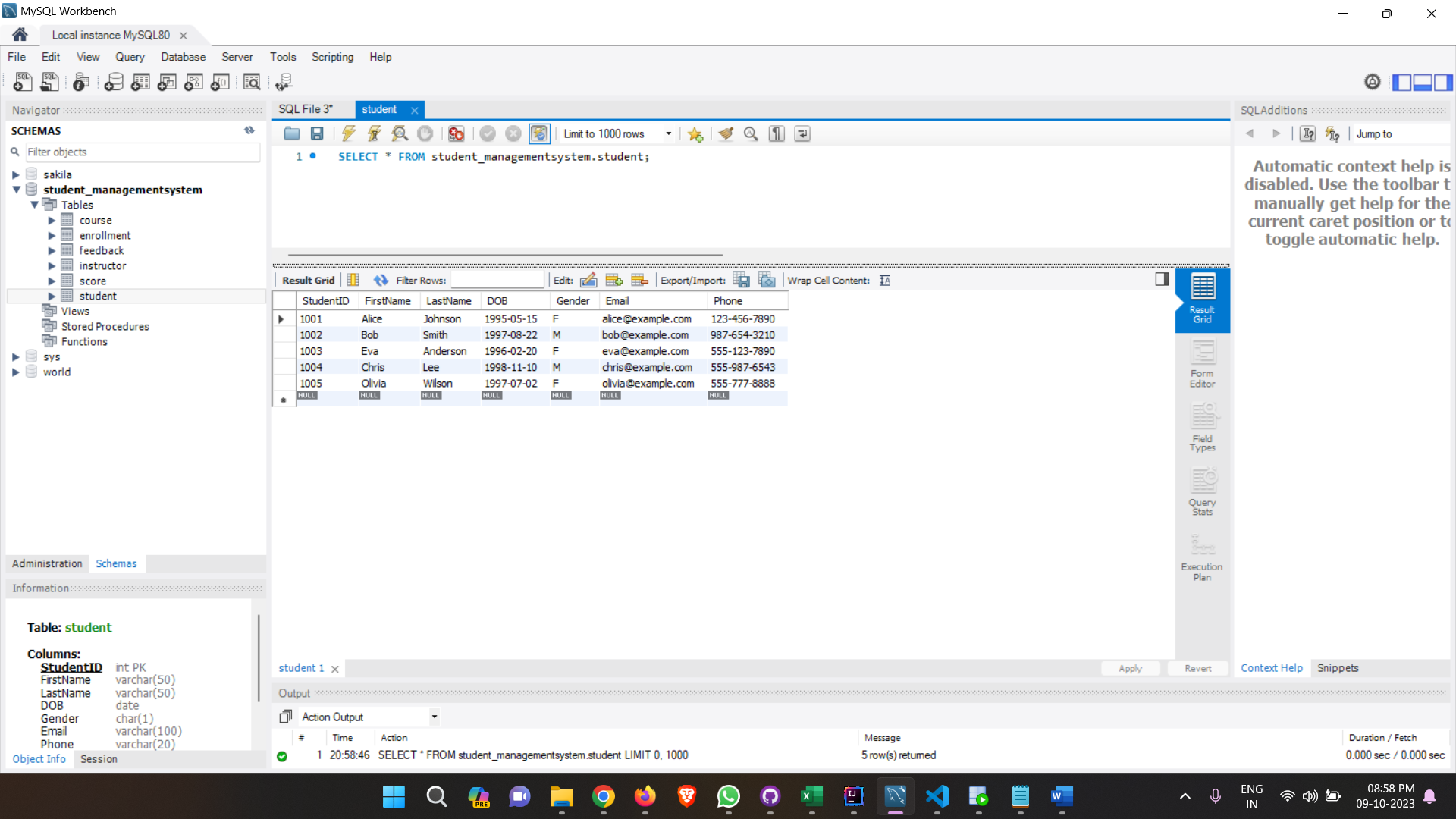
Feedback

* **Relationships**:

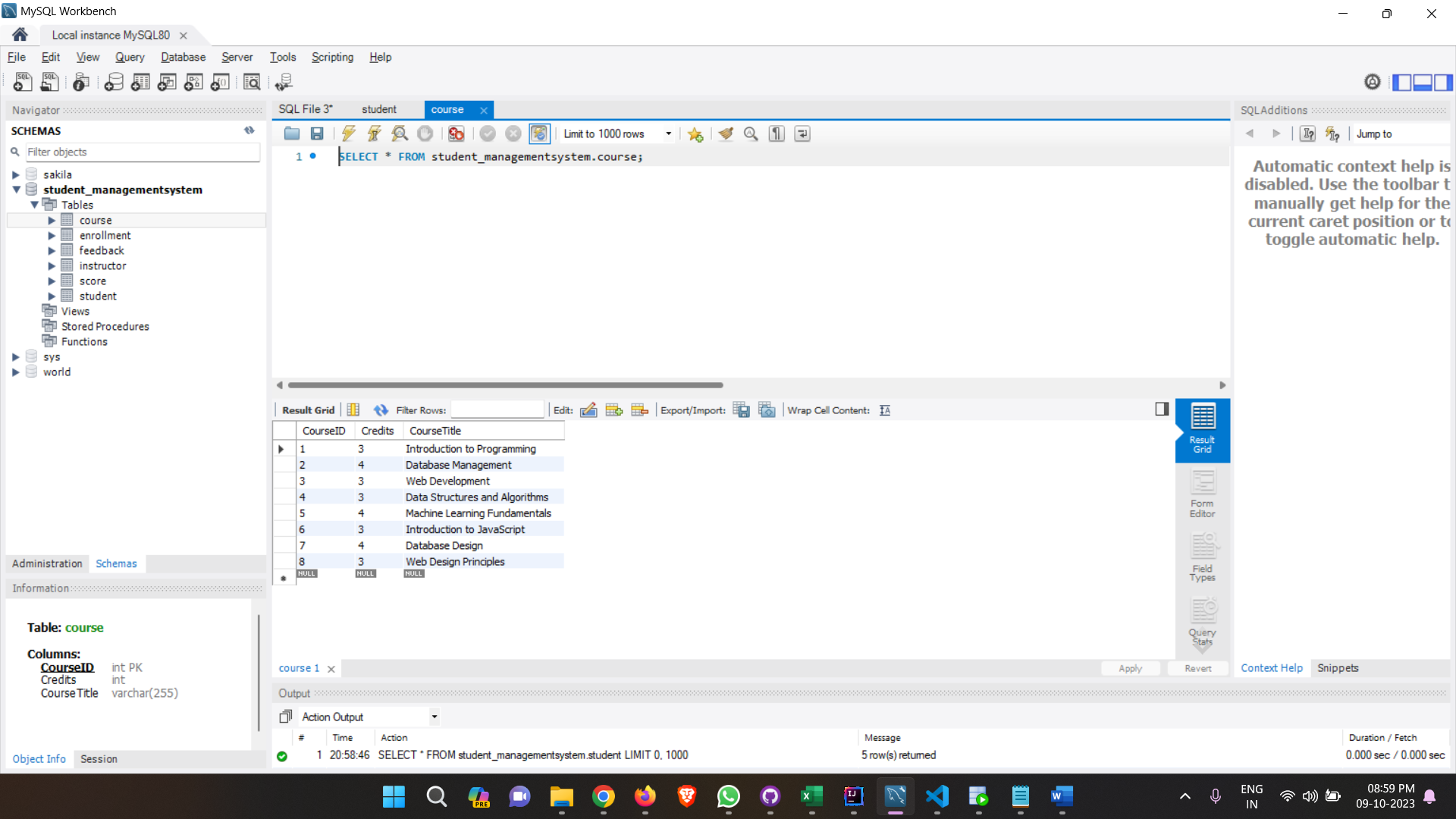
  One **Student** will maps to Many **Feedbacks** (**One-to-Many**)

**Table Structure**

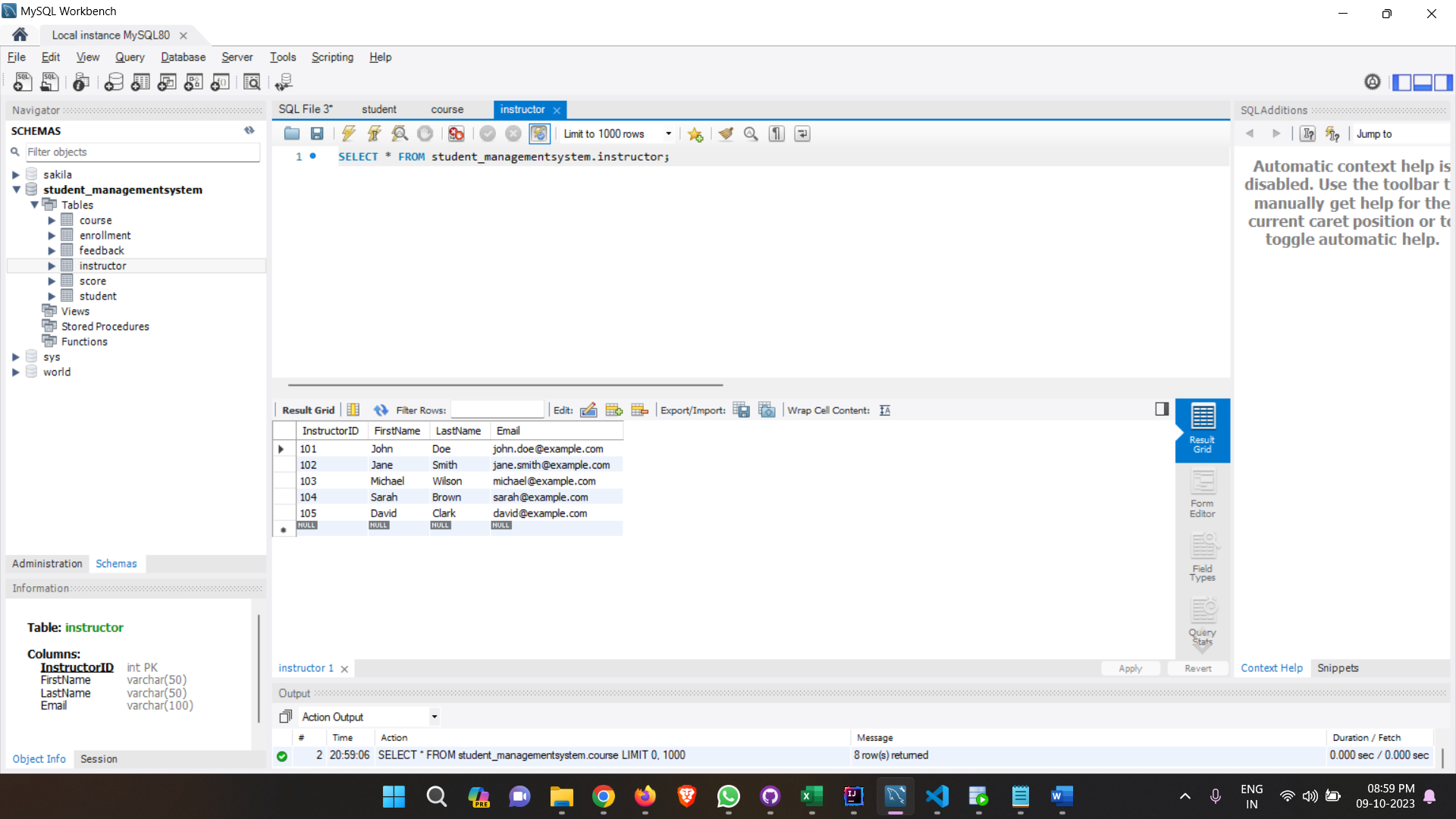
**1.** **Student**



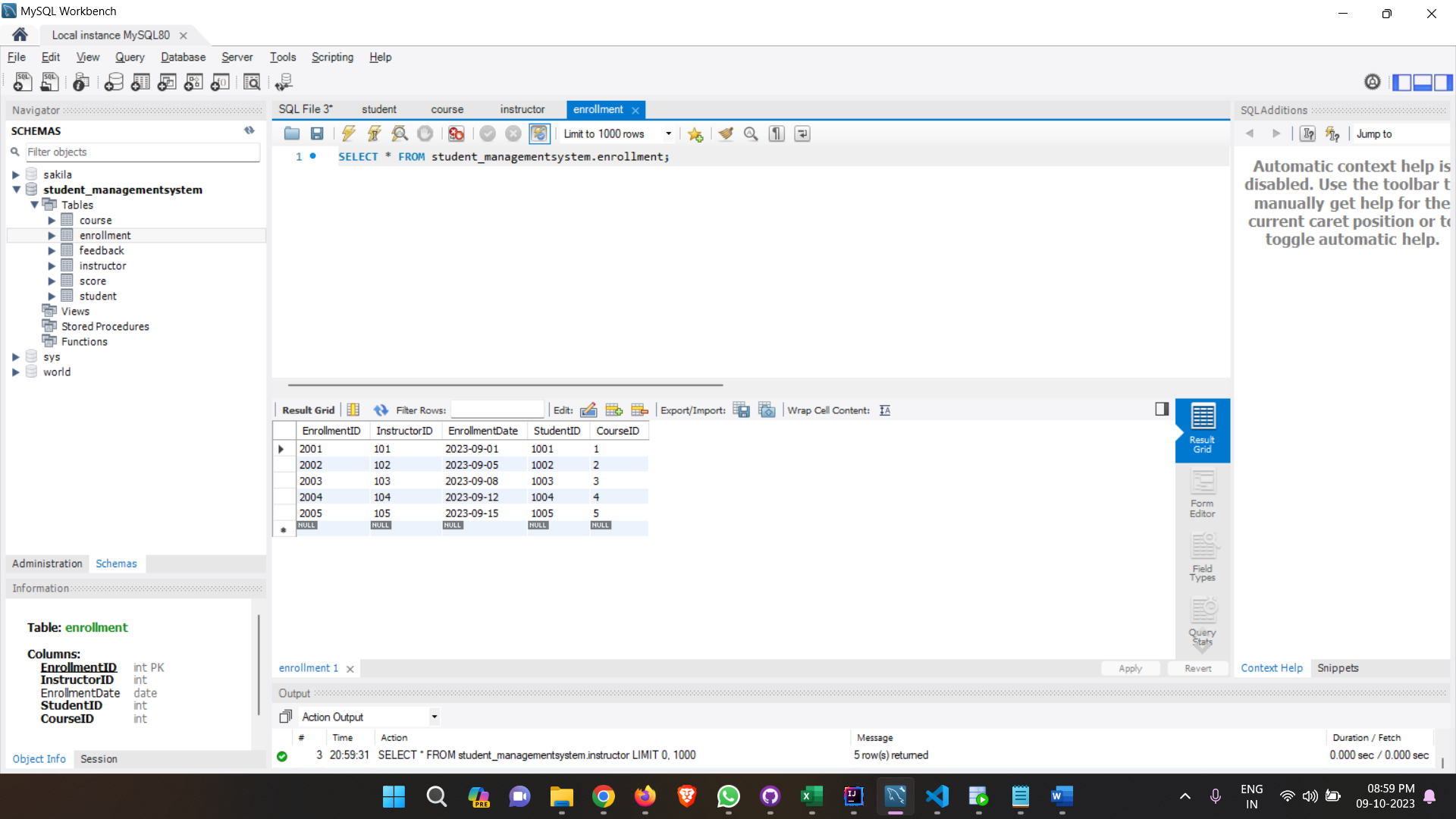
**2. Course**



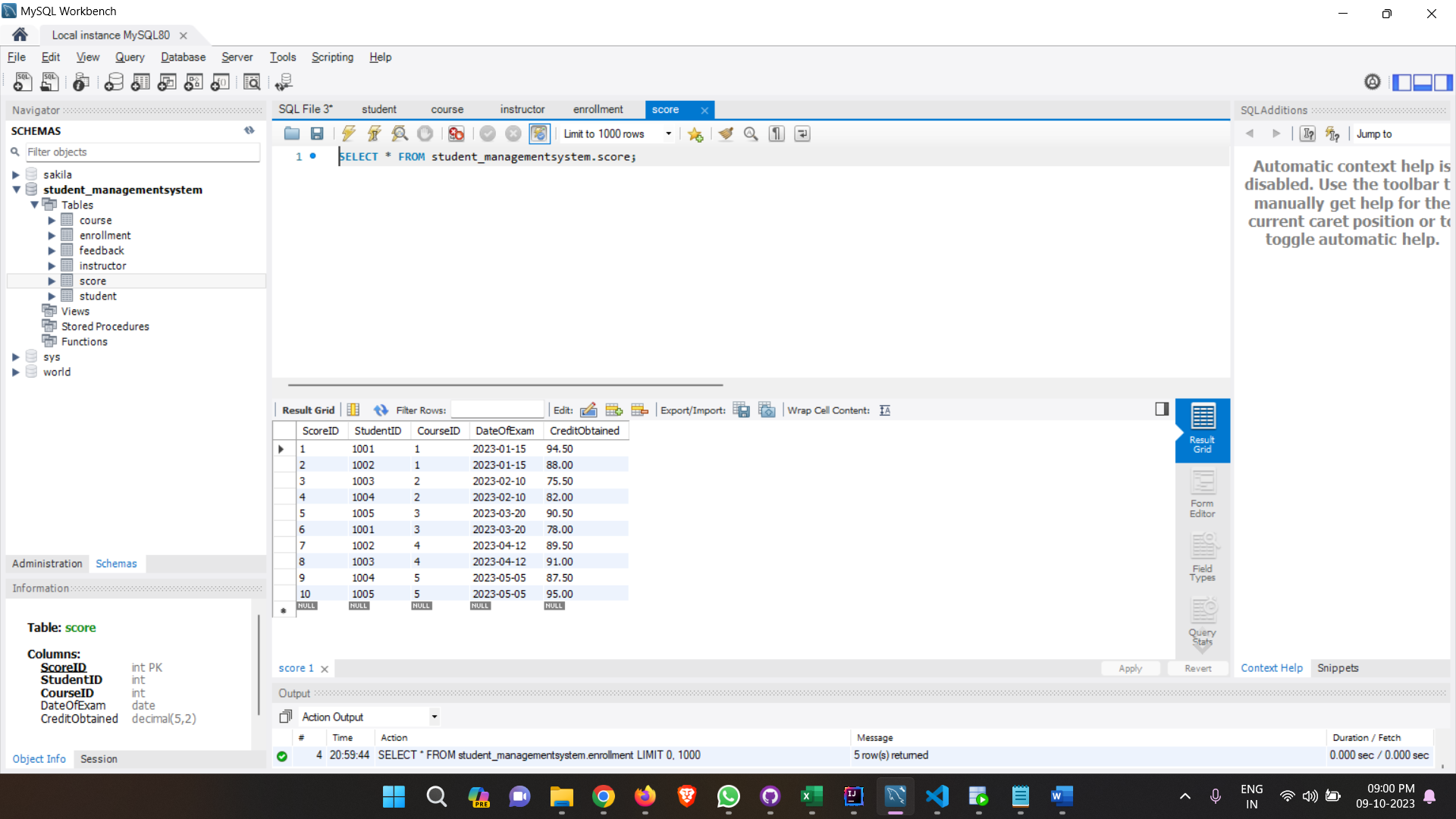
**3. Instructor**



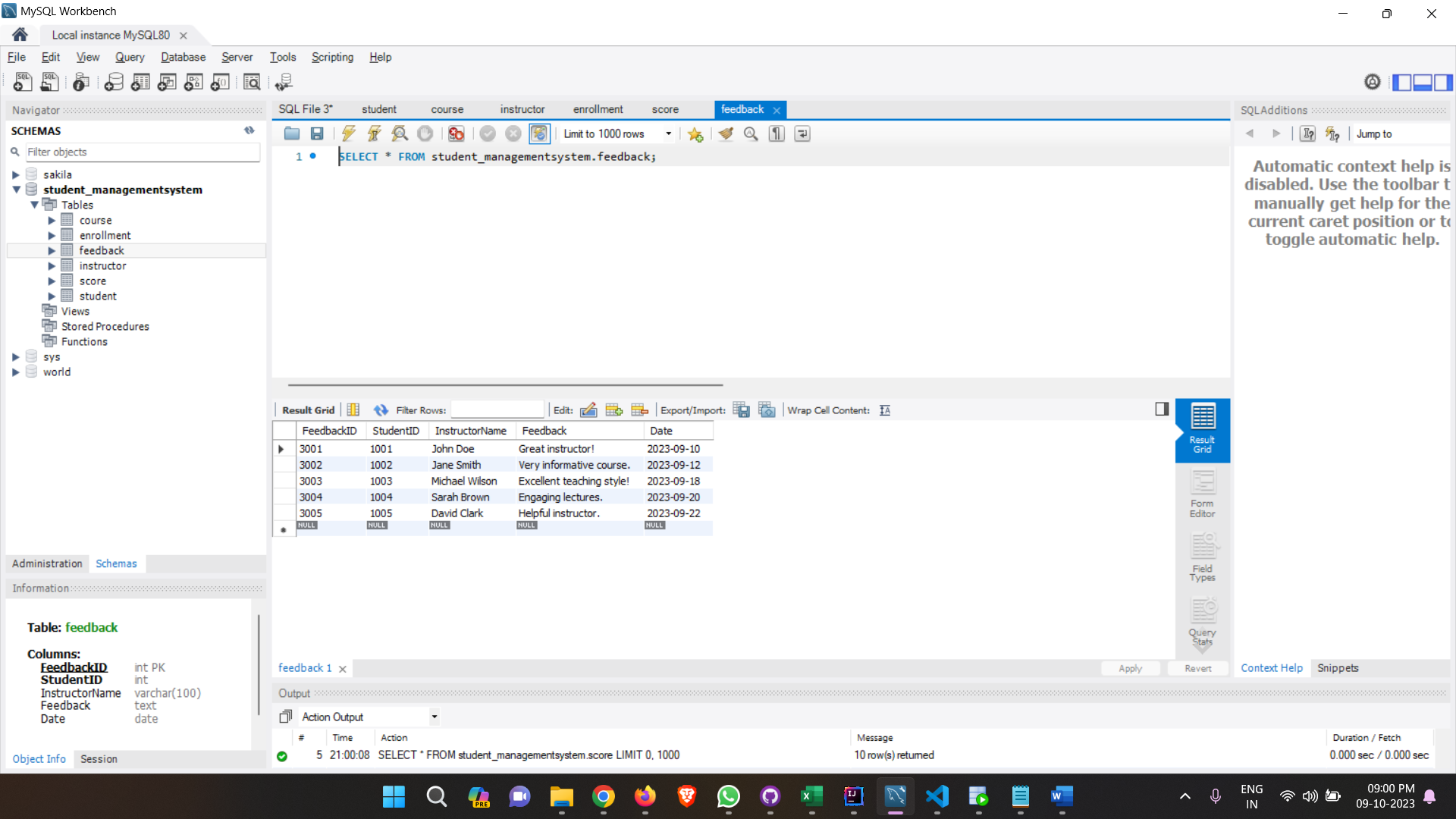
**4. Enrollment**



**5. Score**

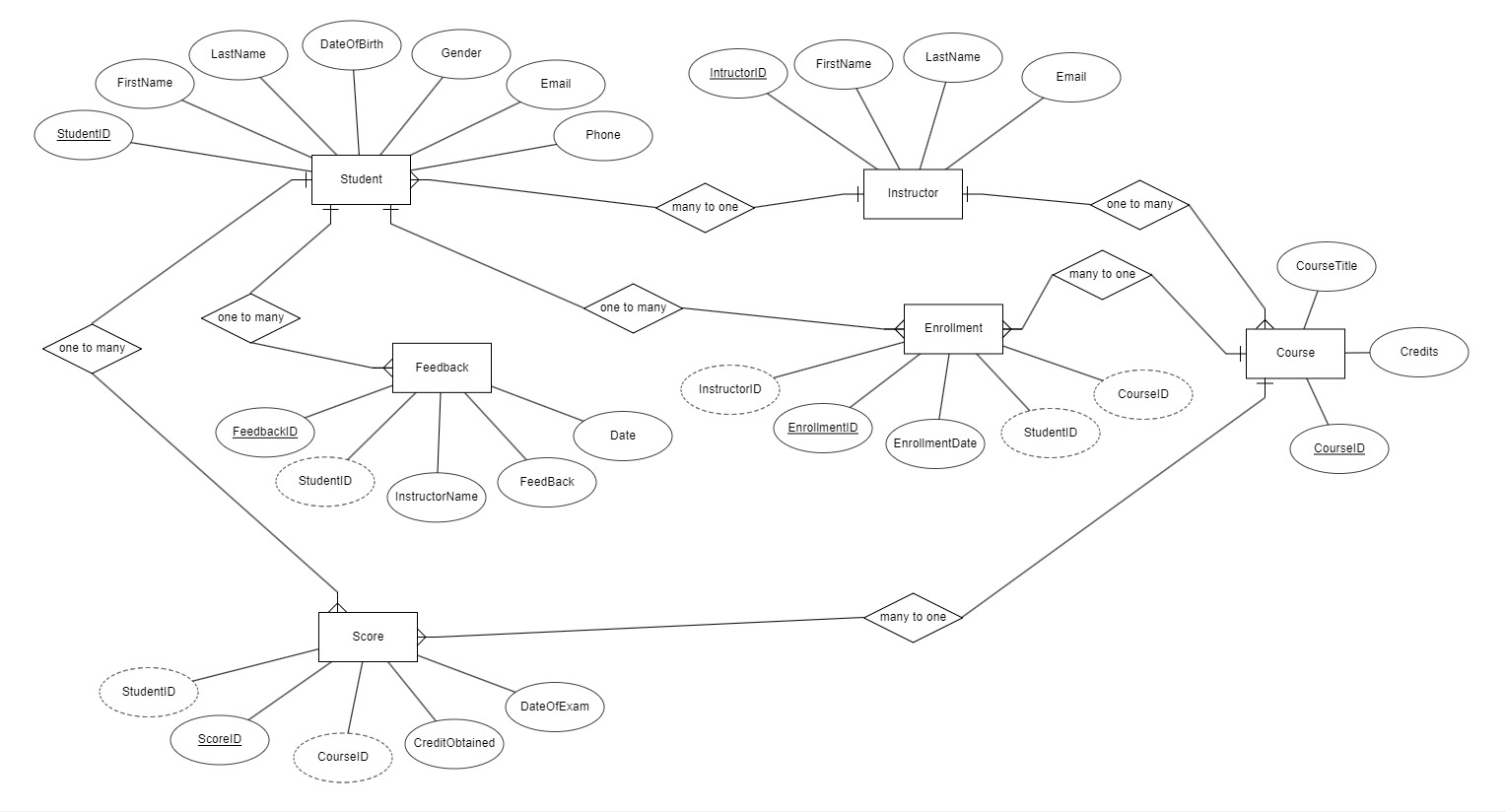


**6. Feedback**



Now, let’s create the ER diagram to visually represent the entities and relationships.

**ERD Diagram**



**In this ERD:**

* Students can enroll, and each course can have multiple students, creating a many-to-many relationship.
* The Enrollment entity serves as a bridge table between Student and Course entities to represent this relationship.
* Multiple courses can be taught by one Instructor (many-to-one relationship).
* Each Instructor can teach multiple courses (one-to-many relationship).
* A student can give multiple feedbacks
* Student may have scores of multiple courses

**4. Creating a Database**

Using MySQL server, create a new database for your student management system. You can do this with SQL commands or through the graphical interface.

*CREATE DATABASE StudentManagementSystem;*

**5. Using a Database**

Before performing any operations on a database, you need to select it using the USE statement:

*USE StudentManagementSystem;*

**6. Creating the tables for each entity**

**------------------ For MYSQL: -----------------------**

CREATE DATABASE student\_managementsystem;

USE student\_managementsystem;

CREATE TABLE Course (

CourseID INT PRIMARY KEY,

Credits INT,

CourseTitle VARCHAR(255)

);

CREATE TABLE Instructor (

InstructorID INT PRIMARY KEY,

FirstName VARCHAR(50),

LastName VARCHAR(50),

Email VARCHAR(100)

);

CREATE TABLE Student (

StudentID INT PRIMARY KEY,

FirstName VARCHAR(50),

LastName VARCHAR(50),

DOB DATE,

Gender CHAR(1),

Email VARCHAR(100),

Phone VARCHAR(20)

);

CREATE TABLE Enrollment (

EnrollmentID INT PRIMARY KEY,

InstructorID INT,

EnrollmentDate DATE,

StudentID INT,

CourseID INT,

FOREIGN KEY (InstructorID) REFERENCES instructor (InstructorID),

FOREIGN KEY (StudentID) REFERENCES student (StudentID),

FOREIGN KEY (CourseID) REFERENCES course (CourseID)

);

CREATE TABLE Feedback (

FeedbackID INT PRIMARY KEY,

StudentID INT,

InstructorName VARCHAR(100),

Feedback TEXT,

Date DATE,

FOREIGN KEY (StudentID) REFERENCES student (StudentID)

);

CREATE TABLE Score (

ScoreID INT PRIMARY KEY,

StudentID INT,

CourseID INT,

DateOfExam DATE,

CreditObtained DECIMAL(5, 2), -- You can adjust the data type as needed for scores

FOREIGN KEY (StudentID) REFERENCES student (StudentID),

FOREIGN KEY (CourseID) REFERENCES course (CourseID)

);

**7. Insert records**

Add data to your tables to work with. This step helps you test your database.

**-- Insert students**

-- Dummy data for the student table

INSERT INTO Student (StudentID, FirstName, LastName, DOB, Gender, Email, Phone)

VALUES

(1001, 'Alice', 'Johnson', '1995-05-15', 'F', 'alice@example.com', '123-456-7890'),

(1002, 'Bob', 'Smith', '1997-08-22', 'M', 'bob@example.com', '987-654-3210'),

(1003, 'Eva', 'Anderson', '1996-02-20', 'F', 'eva@example.com', '555-123-7890'),

(1004, 'Chris', 'Lee', '1998-11-10', 'M', 'chris@example.com', '555-987-6543'),

(1005, 'Olivia', 'Wilson', '1997-07-02', 'F', 'olivia@example.com', '555-777-8888');

**-- Insert courses**

-- Dummy data for the course table

INSERT INTO Course (CourseID, Credits, CourseTitle)

VALUES

(1, 3, 'Introduction to Programming'),

(2, 4, 'Database Management'),

(3, 3, 'Web Development'),

(4, 3, 'Data Structures and Algorithms'),

(5, 4, 'Machine Learning Fundamentals'),

(6, 3, 'Introduction to JavaScript'),

(7, 4, 'Database Design'),

(8, 3, 'Web Design Principles');

**-- Insert instructor**

-- Dummy data for the instructor table

INSERT INTO Instructor (InstructorID, FirstName, LastName, Email)

VALUES

(101, 'John', 'Doe', 'john.doe@example.com'),

(102, 'Jane', 'Smith', 'jane.smith@example.com'),

(103, 'Michael', 'Wilson', 'michael@example.com'),

(104, 'Sarah', 'Brown', 'sarah@example.com'),

(105, 'David', 'Clark', 'david@example.com');

**-- Enroll students in courses**

-- Dummy data for the Enrollment table

INSERT INTO Enrollment (EnrollmentID, InstructorID, EnrollmentDate, StudentID, CourseID)

VALUES

(2001, 101, '2023-09-01', 1001, 1),

(2002, 102, '2023-09-05', 1002, 2),

(2003, 103, '2023-09-08', 1003, 3),

(2004, 104, '2023-09-12', 1004, 4),

(2005, 105, '2023-09-15', 1005, 5);

***—--Insert Score***

-- Dummy data for the Score table (additional records)

INSERT INTO Score (ScoreID, StudentID, CourseID, DateOfExam, CreditObtained)

VALUES

(1, 1001, 1, '2023-01-15', 94.5),

(2, 1002, 1, '2023-01-15', 88.0),

(3, 1003, 2, '2023-02-10', 75.5),

(4, 1004, 2, '2023-02-10', 82.0),

(5, 1005, 3, '2023-03-20', 90.5),

(6, 1001, 3, '2023-03-20', 78.0),

(7, 1002, 4, '2023-04-12', 89.5),

(8, 1003, 4, '2023-04-12', 91.0),

(9, 1004, 5, '2023-05-05', 87.5),

(10, 1005, 5, '2023-05-05', 95.0);

**— Example for Auto-increment primary key**

-- Dummy data for the Feedback table

INSERT INTO Feedback (FeedbackID, StudentID, InstructorName, Feedback, Date)

VALUES

(3001, 1001, 'John Doe', 'Great instructor!', '2023-09-10'),

(3002, 1002, 'Jane Smith', 'Very informative course.', '2023-09-12'),

(3003, 1003, 'Michael Wilson', 'Excellent teaching style!', '2023-09-18'),

(3004, 1004, 'Sarah Brown', 'Engaging lectures.', '2023-09-20'),

(3005, 1005, 'David Clark', 'Helpful instructor.', '2023-09-22');

**8. Select records**

Write SQL queries to retrieve and manage data.

For example:

**Retrieve all student:**

*Select \* FROM Student;*

**Retrieve a student's enrolled courses:**

*Select StudentID,FirstName from student where Gender='M';*

*Select Credits from Course where CourseTitle='History101' ;*

**\*Now try similar Select queries with other tables**

**9. Update records**

Write SQL statements to update record(s) when needed. For example:

**Update a student's email:**

**UPDATE Students**

*Update Student SET email = 'johndoe@example.com'*

*Where FirstName = 'John' and LastName='Doe';*

**10. Delete records**

Write SQL statements to delete record(s) when needed.

*Delete FROM Students*

*Where Where FirstName = 'John' and LastName='Doe';*

**PN:** Ideally no data should be deleted from any tables. You can use an additional column to set the status of that record to ‘Active/Inactive’, etc. Or you can use an Archive table to move the unnecessary records out of the main table.