# NED University of Engineering & Technology

# 

# “ACADEMIC MANAGEMENT SYSTEM”

# GROUP MEMBERS:

# *MUHAIB SHAMSHER (CR-029)*

# *S. GHUFRAN RAZA (CR-028)*

# *HASNAIN RAZA KHAN (CR-034)*

**Instructor:** *Ms. Nudrat Naved*

Database Management System

(CT-261)

Department Of Computer Science & Information Technology

(Specialization In Cyber Security)

# DESCRIPTION:

# Project Description: Academic Management System

# This database project aims to create a comprehensive Student Enrollment and Course Management System for an educational institution. The system manages students, courses, instructors, departments, enrollments, grades, and course details. Here’s an overview of its components:

# Students Table: Stores student information including ID, name, date of birth, gender, enrollment date, email, and phone number.

# Courses Details Table: Contains details about courses such as course ID, name, code, description, and the department it belongs to.

# Departments Table: Lists different academic departments within the institution, each identified by a unique department ID and named accordingly.

# Instructors Table: Stores details of instructors who teach courses, including their ID, name, contact information (email and phone), hire date, and their associated department.

# Course Instructors Table: Links courses to instructors by storing pairs of course IDs and instructor IDs, establishing which instructors teach which courses.

# Enrollments Table: Manages student enrollments into courses by storing enrollment IDs along with student IDs, course IDs, and enrollment dates.

# Grades Table: Tracks grades awarded to students for enrolled courses, linked to enrollment IDs and storing the actual grade.

# The system supports complex queries and operations such as:

# Advanced Queries: Retrieving detailed information about students, courses, instructors, departments, and their relationships.

# Data Manipulation: Updating instructor details, modifying course descriptions, and handling changes in student enrollment statuses.

# Data Integrity: Ensuring data consistency through proper use of foreign key constraints and transaction management.

# Reporting: Generating reports on student performance, course popularity, instructor workloads, and departmental statistics.

# This Student Enrollment and Course Management System enhances administrative

# efficiency, improves data accessibility, and supports informed decision-making within the

# educational institution.

1. **Relational Schema:**

Students (**student\_id** , first\_name, last\_name, date\_of\_birth, gender, enrollment\_date, email, phone)

Courses\_details (**course\_id** , course\_name, course\_code, course\_description, department\_id )

Courses (**course\_code**, credits, semester\_offered )

Instructors (**instructor\_id**, first\_name, last\_name, email, phone, hire\_date, department\_id )

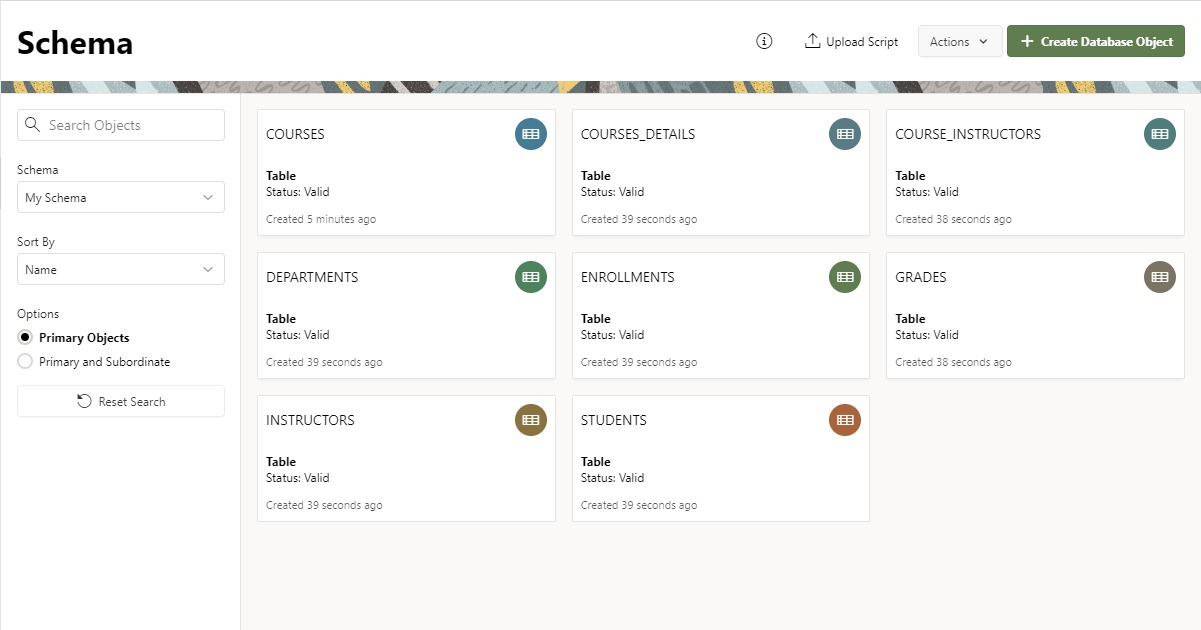
Departments (**department\_id** , department\_name)

Enrollments (**enrollment\_id** , student\_id , course\_id, enrollment\_date)

Grades (**grade\_id** , enrollment\_id, grade)

Course\_Instructors (**course\_id** , **instructor\_id**)

1. **Schema:**



**Queries:**

1. **To insert the student data in Students table**.

INSERT INTO Students (student\_id, first\_name, last\_name, date\_of\_birth, gender, enrollment\_date, email, phone)

VALUES

(1, 'Jane', 'Doe', '1995-03-15', 'F', '2023-09-01', 'jane.doe@example.com', '9876543210'),

(2, 'John', 'Brown', '1996-05-10', 'M', '2023-08-02', 'john.brown@example.com', '3456789012'),

(3, 'Hamid', 'Ali', '1997-11-12', 'M', '2023-05-28', 'hamid.ali@example.com', '5678901234'),

(5, 'Emily', 'Robinson', '1996-06-15', 'F', '2023-09-02', 'emily.robinson@example.com', '5678901234'),

(6, 'Frank', 'Patel', '1999-03-20', 'M', '2023-09-03', 'frank.patel@example.com', '6789012345'),

(7, 'Henry', 'Anderson', '1997-09-22', 'F', '2023-09-04', 'henry.anderson@example.com', '8901234567'),

(8, 'Isabella', 'Garcia', '1996-03-08', 'M', '2023-09-05', 'isabella.garcia@example.com', '9012345678'),

(9, 'Jack', 'Thompson', '1998-07-14', 'F', '2023-09-06', 'jack.thompson@example.com', '1230123012'),

(10, 'Katherine', 'Harris', '1995-01-25', 'M', '2023-09-07', 'katherine.harris@example.com', '2341234123'),

(11, 'Liam', 'Martin', '1997-12-03', 'F', '2023-09-08', 'liam.martin@example.com', '3452345234'),

(12, 'Mia', 'Rodriguez', '1996-05-20', 'M', '2023-09-09', 'mia.rodriguez@example.com', '4563456345'),

(13, 'Noah', 'Young', '1999-04-18', 'F', '2023-09-10', 'noah.young@example.com', '5674567456'),

(14, 'Olivia', 'Lee', '1994-08-07', 'M', '2023-09-11', 'olivia.lee@example.com', '6785678567');



1. **Delete query scenario involving multiple tables.**

DELETE FROM grades

WHERE enrollment\_id IN (

SELECT enrollment\_id

FROM enrollments

WHERE student\_id = 1

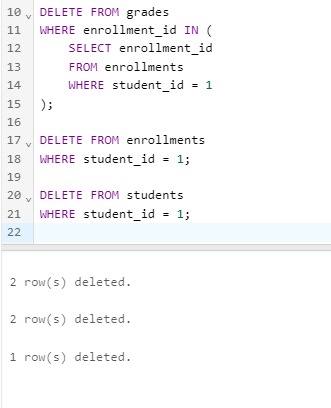
);

DELETE FROM enrollments

WHERE student\_id = 1;

DELETE FROM students

WHERE student\_id = 1;



1. **Update the department name and instructor details for a specific course.**

UPDATE Courses\_Details cd

SET cd.course\_description = 'Advanced Biology with Lab'

WHERE cd.course\_id = 101;

UPDATE Instructors i

SET i.email = 'updated.email@example.com'

WHERE i.instructor\_id = (

SELECT ci.instructor\_id

FROM Course\_Instructors ci

WHERE ci.course\_id = 101

AND ci.instructor\_id = 22101

);

UPDATE Departments d

SET d.department\_name = 'Biological Sciences'

WHERE d.department\_id = (

SELECT cd.department\_id

FROM Courses\_Details cd

WHERE cd.course\_id = 101

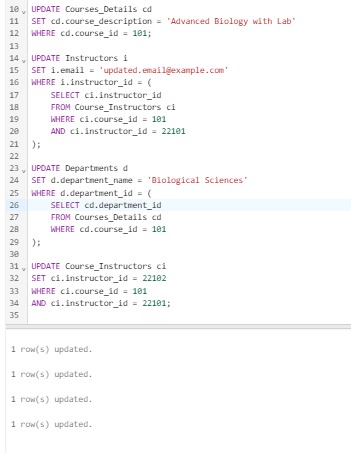
);

UPDATE Course\_Instructors ci

SET ci.instructor\_id = 22102

WHERE ci.course\_id = 101

AND ci.instructor\_id = 22101;



1. **Students with Multiple Enrollments. (Join & Aggregate function)**

SELECT s.student\_id, s.first\_name, s.last\_name, COUNT(e.enrollment\_id) AS enrollments\_count

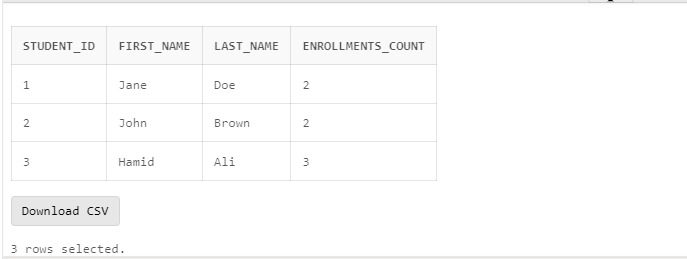
FROM students s

JOIN enrollments e

ON s.student\_id = e.student\_id

GROUP BY s.student\_id, s.first\_name, s.last\_name

HAVING COUNT(e.enrollment\_id) > 1;



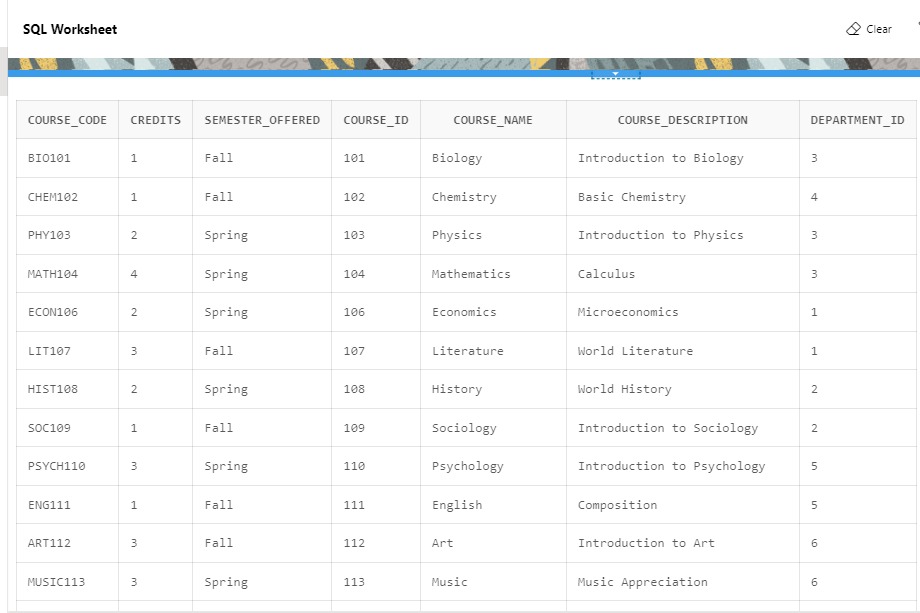
1. **To retrieve information from both courses and courses\_details tables using a join query. (Inner join)**

SELECT c.course\_code, c.credits, c.semester\_offered, cd.course\_id, cd.course\_name, cd.course\_description, cd.department\_id

FROM courses c

INNER JOIN courses\_details cd

ON c.course\_code = cd.course\_code;



1. **Retrieve Student Enrollment Details with Course Information**.

SELECT s.student\_id, s.first\_name, s.last\_name, s.enrollment\_date AS student\_enrollment\_date, e.enrollment\_id, e.enrollment\_date AS enrollment\_date, cd.course\_id, cd.course\_name, cd.course\_code, cd.department\_id

FROM students

JOIN enrollments e

ON s.student\_id = e.student\_id

JOIN courses\_details cd

ON e.course\_id = cd.course\_id;



1. **Filtering Courses by Semester Offered and Department Name.**

SELECT cd.course\_id, cd.course\_name, cd.course\_code, cd.course\_description, c.credits, c.semester\_offered,

d.department\_name

FROM courses\_details cd

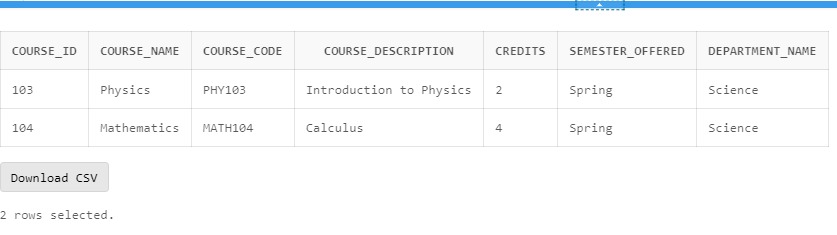
JOIN courses c

ON cd.course\_code = c.course\_code

JOIN departments d

ON cd.department\_id = d.department\_id

WHERE c.semester\_offered = 'Spring' AND d.department\_name = 'Science';



1. **Subquery to Find Courses Taught by Each Instructor.**

SELECT

i.instructor\_id,

i.first\_name || ' ' || i.last\_name AS instructor\_name,

(

SELECT LISTAGG(cd.course\_name, ', ')

WITHIN GROUP (ORDER BY cd.course\_name)

FROM courses\_details cd

JOIN course\_instructors ci

ON cd.course\_id = ci.course\_id

WHERE ci.instructor\_id = i.instructor\_id

) AS courses\_taught

FROM instructors i;



1. **Subquery to Count Courses per Department.**

SELECT

d.department\_id,

d.department\_name,

(

SELECT COUNT(\*)

FROM courses\_details cd

WHERE cd.department\_id = d.department\_id

) AS num\_courses

FROM departments d;

