OMIS 105: Final Group Project

By: Natalie Tun, Andrea Tovar, Jonah Engelmann, Andrew Wang, Kayla Huffman, Izzy Furuhashi

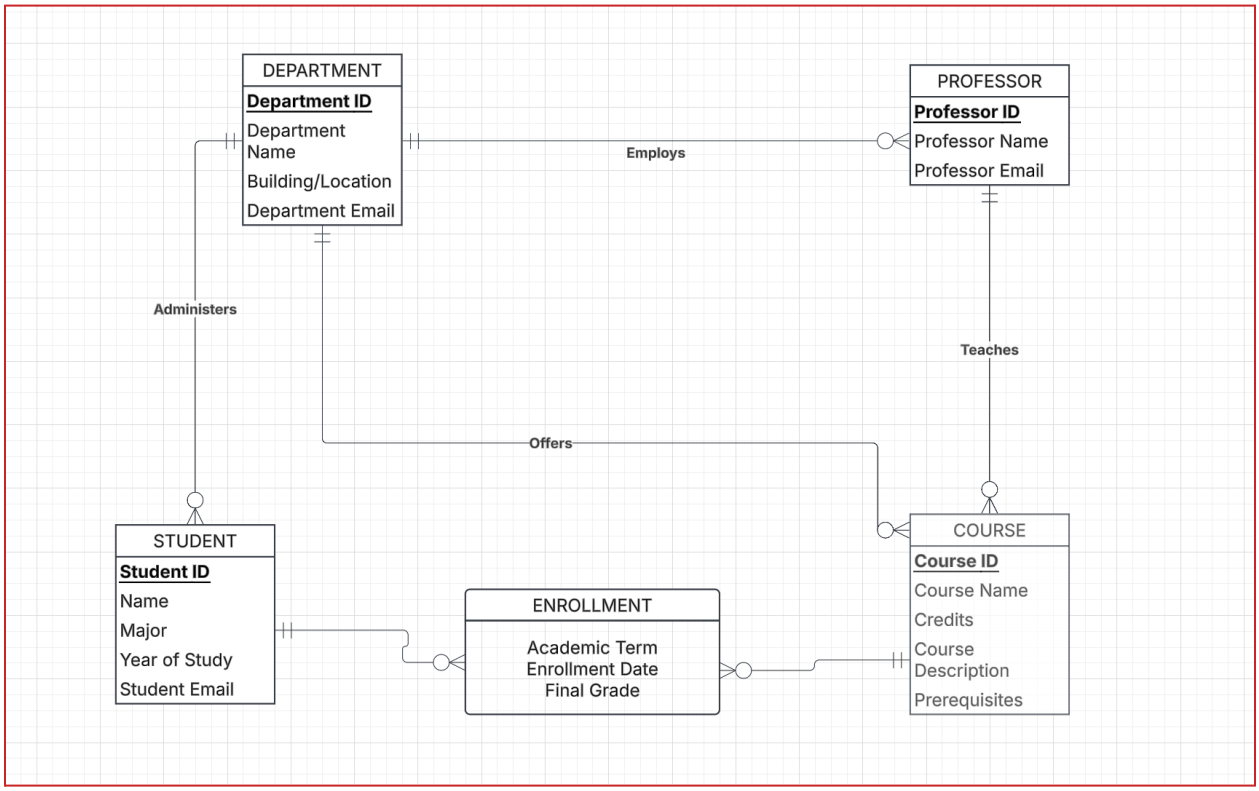
Part One: Background/Objective

Our SCU Leavey School of Business course management system is designed to efficiently organize and track essential academic information (ex: students, professors, courses, enrollments, departments). As the university has expanded, managing enrollments, course offerings, and student performance has become complex. With thousands of students, the administration has struggled to ensure smooth operations.

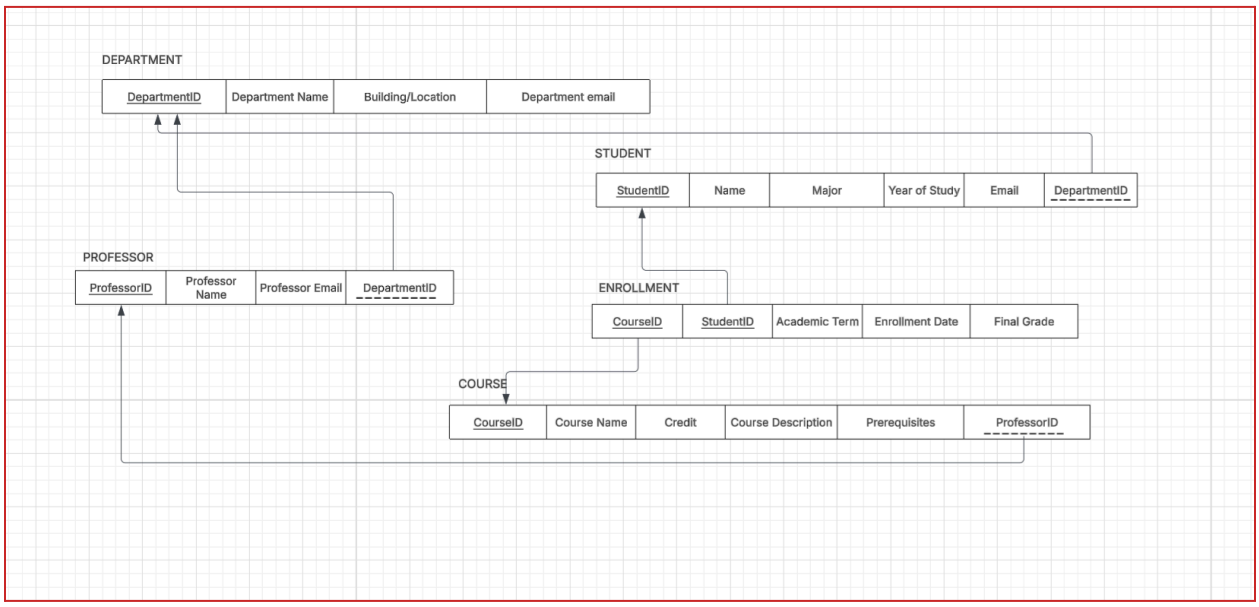
Thus, the administration has decided to hire our team to implement a database system that tracks all core elements of the university's courses and enrollments. With this system, the university will record and track student information, like student ID, name, major, year of study, and email. Due to Leavey School of Business policies, the database enforces a single major rule and therefore can only belong to one department. It will also manage professor details, such as professor ID, name, and their email. Each course will be recorded with a unique course ID, course name, number of credits, description, and prerequisites. Finally, our system tracks all departments to ensure each student, course, and professor is correctly associated with their respective academic unit.

When a student enrolls in a course, their enrollment is added to the system, linking the student to the specific course and professor for that academic term. By implementing this database, this system will improve in terms of organization, reducing errors, and supporting future academic planning.

Part Two: Conceptual Diagram



Part Three: Logical Design



** Based on feedback in class, we removed DepartmentID from COURSE to eliminate redundancy and the risk of mismatched department data.

Part 4: Data Dictionary

DEPARTMENT

Name	Data Type	Constraints	Key	Description
DepartmentID	numeric(11,0)	>0	PK	Unique identifier for a department
Department Name	varchar(25)			Name of department
Building/Location	varchar(25)			Building name of department
Department Email	varchar(20)			Email of department

PROFESSOR

Name	Data Type	Constraints	Key	Description
ProfessorID	numeric(11,0)	>0	PK	Unique identifier for a professor
Professor Name	varchar(20)			Full name of professor
Professor Email	varchar(30)			Email address of professor
DepartmentID	numeric(11,0)	>0	FK	Unique identifier for a department

STUDENT

Name	Data Type	Constraints	Key	Description
StudentID	numeric(11,0)	>0	PK	Unique identifier for a student
Name	varchar(25))			First and Last name of student
Major	varchar(25)			Major of a student
Year of Study	numeric(8,0)	>0		Number of years a student has been studying for
Email	varchar(20)			Email of a student
DepartmentID	numeric(11,0)	>0	FK	Unique identifier for a department

ENROLLMENT

Name	Data Type	Constraints	Key	Description
CourseID	numeric(11,0)	>0	PK	Unique identifier for a course
StudentID	numeric(11,0)	>0	PK	Unique identifier for a student
Academic Term	varchar(20)			Academic Term at time of enrollment
Enrollment Date	date			Date of student's enrollment
Final Grade	numeric(11,2)	>0		Final grade a student receives in the enrolled class

COURSE

Name	Data Type	Constraints	Key	Description
CourseID	numeric(11,0)		PK	Unique identifier for a course
Course Name	varchar(25)			Name of the course
Credits	numeric(5,0)	>=0		Number of credits assigned to the course
Course Descriptions	varchar(45)			Brief description of the course content
Prerequisites	varchar(20)			Courses required before taking this course
ProfessorID	numeric(11,0)	>0	FK	Unique identifier for a professor
DepartmentID	numeric(11,0)	>0	FK	Unique identifier for a department

Part 5: SQL Implementation - The Physical Design in My SQL

```sql
CREATE SCHEMA University;
USE University;

CREATE TABLE Department (
    DepartmentID NUMERIC(11,0) PRIMARY KEY CHECK (DepartmentID > 0),
    DepartmentName VARCHAR(25),
    BuildingLocation VARCHAR(25),
    DepartmentEmail VARCHAR(20)
);

CREATE TABLE Professor (
    ProfessorID NUMERIC(11,0) PRIMARY KEY CHECK (ProfessorID > 0),
    ProfessorName VARCHAR(20),
    ProfessorEmail VARCHAR(30),
    DepartmentID NUMERIC(11,0) CHECK (DepartmentID > 0),
    FOREIGN KEY (DepartmentID) REFERENCES Department(DepartmentID)
);

CREATE TABLE Student (
    StudentID NUMERIC(11,0) PRIMARY KEY CHECK (StudentID > 0),
    Name VARCHAR(25),
    Major VARCHAR(25),
    YearOfStudy NUMERIC(8,0) CHECK (YearOfStudy > 0),
    Email VARCHAR(20),
    DepartmentID NUMERIC(11,0) CHECK (DepartmentID > 0),
    FOREIGN KEY (DepartmentID) REFERENCES Department(DepartmentID)
);
```

```sql
CREATE TABLE Course (
    CourseID NUMERIC(11,0) PRIMARY KEY CHECK (CourseID > 0),
    CourseName VARCHAR(25),
    Credits NUMERIC(5,0) CHECK (Credits >= 0),
    CourseDescription VARCHAR(45),
    Prerequisites VARCHAR(20),
    ProfessorID NUMERIC(11,0) CHECK (ProfessorID > 0),
    DepartmentID NUMERIC(11,0) CHECK (DepartmentID > 0),
    FOREIGN KEY (ProfessorID) REFERENCES Professor(ProfessorID),
    FOREIGN KEY (DepartmentID) REFERENCES Department(DepartmentID)
);

CREATE TABLE Enrollment (
    CourseID NUMERIC(11,0) CHECK (CourseID > 0),
    StudentID NUMERIC(11,0) CHECK (StudentID > 0),
    AcademicTerm VARCHAR(20),
    EnrollmentDate DATE,
    FinalGrade NUMERIC(11,2) CHECK (FinalGrade >= 0),
    PRIMARY KEY (CourseID, StudentID),
    FOREIGN KEY (CourseID) REFERENCES Course(CourseID),
    FOREIGN KEY (StudentID) REFERENCES Student(StudentID)
);
```

```sql
INSERT INTO Department (DepartmentID, DepartmentName, BuildingLocation, DepartmentEmail) VALUES
(1, 'Finance', 'Lucas Hall', 'finance@scu.edu'),
(2, 'Marketing', 'Kenna Hall', 'marketing@scu.edu'),
(3, 'Accounting', 'Alumni Science', 'accounting@scu.edu'),
(4, 'Computer Science', 'SCDI', 'compsci@scu.edu'),
(5, 'Economics', 'Vari Hall', 'economics@scu.edu'),
(6, 'Business Analytics', 'Heafey Hall', 'analytics@scu.edu'),
(7, 'Entrepreneurship', 'Varsi Hall', 'entre@scu.edu');

INSERT INTO Professor (ProfessorID, ProfessorName, ProfessorEmail, DepartmentID) VALUES
(101, 'Alice Smith', 'asmith@scu.edu', 1),
(102, 'Michael Johnson', 'mjohnson@scu.edu', 2),
(103, 'David Lee', 'dlee@scu.edu', 3),
(104, 'Sarah Brown', 'sbrown@scu.edu', 4),
(105, 'Emily Davis', 'edavis@scu.edu', 5),
(106, 'James Wilson', 'jwilson@scu.edu', 6),
(107, 'Jessica Moore', 'jmoore@scu.edu', 7);

INSERT INTO Student (StudentID, Name, Major, YearOfStudy, Email, DepartmentID) VALUES
(201, 'Alice Johnson', 'Finance', 2, 'ajohnson@scu.edu', 1),
(202, 'Bob Smith', 'Marketing', 3, 'bsmith@scu.edu', 2),
(203, 'Charlie Brown', 'Accounting', 1, 'cbrown@scu.edu', 3),
(204, 'Daisy White', 'Management', 4, 'dwhite@scu.edu', 4),
(205, 'Evan Green', 'Economics', 2, 'egreen.edu', 5),
(206, 'Frank Adams', 'Business Analytics', 3, 'fadams@scu.edu', 6),
(207, 'Grace Hall', 'Entrepreneurship', 1, 'ghall@scu.edu', 7);
```

```sql
INSERT INTO Course (CourseID, CourseName, Credits, CourseDescription, Prerequisites, ProfessorID, DepartmentID) VALUES
(301, 'Financial Management', 4, 'Advanced financial concepts', NULL, '101', '1'),
(302, 'Intro to Marketing', 4, 'Marketing tactics and strategy', NULL, '102', '2'),
(303, 'Managerial Accounting', 5, 'Accounting Decision Making', 'Basic Accounting', '103', '3'),
(304, 'Business Law', 4, 'Basics of Business Law', NULL, '104','4'),
(305, 'Macroeconomics', 4, 'Study of large-scaled economic systems', NULL, '105', '5'),
(306, 'Data Analytics', 5, 'Business analytics tools and techniques', NULL,'106', '6'),
(307, 'Shark Tank', 2, 'Start-up Development Strategies', NULL,'107', '7');


INSERT INTO Enrollment (StudentID, CourseID, EnrollmentDate, AcademicTerm,FinalGrade) VALUES
(201, 301, '2024-08-15', 'Fall 2024', '98'),
(202, 302, '2024-08-20', 'Fall 2024', '85'),
(203, 303, '2025-01-10', 'Spring 2025', '86'),
(204, 304, '2025-01-12', 'Spring 2025', '76'),
(205, 305, '2024-08-18', 'Fall 2024', '94'),
(206, 306, '2024-12-05', 'Winter 2025', '90'),
(207, 307, '2025-01-14', 'Spring 2025', '67');
```

SQL SELECT Queries: 3 Different Datasets Queried in SQL

Multi-Table Query: This query is designed to provide the school the ability to see all students who have gotten above or below a certain grade in any course. For example, this could be used to find all students who have gotten an A in a course in order to give them a reward. In the image below, it is implemented to return all student names, course names and final grades where the student received a score over 90% (an A- or better) .

```sql
SELECT
    Student.Name, Course.CourseName, Enrollment.FinalGrade
FROM
    Student JOIN Enrollment ON Student.StudentID = Enrollment.StudentID
    JOIN Course on Enrollment.CourseID = Course.CourseID
WHERE
    Enrollment.FinalGrade >= 90.00
ORDER BY
    Enrollment.FinalGrade DESC
```

0% 28:9

Result Grid Filter Rows: Search Export:

Name	CourseName	FinalGrade
Alice Johnson	Financial Management	98.00
Evan Green	Macroeconomics	94.00
Frank Adams	Data Analytics	90.00

String Attributes Query: This query is designed to provide the school the ability to see all courses available in a full calendar year. For example, this could be used to track all the courses taken in 2024 to see what should and shouldn't be offered again in the future. In the image below, it is implemented to return the specific course name, and exact term of all courses offered in the year 2025.

```
1  SELECT
2      Course.CourseName, Enrollment.AcademicTerm
3  FROM
4      Enrollment JOIN Course on Enrollment.CourseID = Course.CourseID
5  WHERE
6      AcademicTerm LIKE "%2025%"
7  Order By
8      Enrollment.EnrollmentDate
```

0% 27:8

esult Grid Filter Rows: Search Export:

CourseName	AcademicTerm
Data Analytics	Winter 2025
Managerial Accounting	Spring 2025
Business Law	Spring 2025
Shark Tank	Spring 2025

Group By Query: This query is designed to help the school find classes where students are struggling or exceeding by looking at the average final grade. For example, this could be used to find the courses where the average grade is below a C or above a B+ in order for the school to make changed to the course. In the image below, it is implemented to find the Course ID and average grade for all courses where the average grade was below an 80 (worse than a B-)

```sql
SELECT CourseID, AVG(FinalGrade) AS AvgGrade
FROM Enrollment
GROUP BY CourseID
HAVING AVG(FinalGrade) < 80;
```

CourseID	AvgGrade
304	76.000000
307	67.000000