Course: Database Systems and Software Engineering **Assignment Type:** Mixed Assignment (Theory + Practical + Problem-Solving) **Due Date:** March 30, 2026 **Total Points:** 100

Overview

This comprehensive assignment combines theoretical knowledge, practical database implementation, and problem-solving skills. You will design, implement, and document a database system for a university course registration system.

Part 1: Theoretical Questions (25 points)

Question 1: Database Normalization (10 points)

Consider the following unnormalized table for a university system:

COURSE_ENROLLMENT (StudentID, StudentName, StudentEmail, CourseID, CourseName, InstructorID, InstructorName, InstructorOffice, Grade, Semester, Department)

a) Identify all functional dependencies in this table. (4 points)

b) Explain why this table violates 2NF and 3NF. (3 points)

c) Normalize this table to 3NF, showing all intermediate steps (1NF $\rightarrow$ 2NF $\rightarrow$ 3NF). Draw the resulting schema. (3 points)

Question 2: ACID Properties (8 points)

Explain each of the ACID properties (Atomicity, Consistency, Isolation, Durability) in the context of database transactions. Provide a real-world scenario where violation of each property could cause problems in a banking system.

Question 3: Indexing (7 points)

a) What is the purpose of database indexing? (2 points)

b) Explain the difference between clustered and non-clustered indexes. (3 points)

c) When would creating an index be disadvantageous? (2 points)

Part 2: Database Design and Implementation (40 points)

Design and implement a **University Course Registration System** database.

Requirements:

Entity Requirements:

- **Students:** student_id, first_name, last_name, email, date_of_birth, major, enrollment_year
- **Courses:** course_id, course_code, course_name, credits, department, max_capacity
- **Instructors:** instructor_id, first_name, last_name, email, office, department
- **Enrollments:** enrollment_id, student_id, course_id, semester, year, grade
- **Prerequisites:** course_id, prerequisite_course_id

Task 2.1: ER Diagram (10 points)

Create an Entity-Relationship (ER) diagram that includes: - All entities with their attributes - Primary keys clearly marked - Relationships between entities with cardinality - Any necessary junction tables

Tools you can use: draw.io, Lucidchart, or hand-drawn (must be clear and legible)

Task 2.2: SQL Implementation (20 points)

Write SQL statements to:

a) Create Tables (10 points) - Create all five tables with appropriate: - Data types - Primary keys - Foreign keys with referential integrity - Constraints (NOT NULL, UNIQUE, CHECK where appropriate) - Default values where applicable

b) Insert Sample Data (5 points) - Insert at least: - 5 students - 6 courses - 3 instructors - 10 enrollments - 4 prerequisite relationships

c) Create Indexes (5 points) - Create appropriate indexes on: - Student email - Course code - Instructor department - Enrollment semester and year

---

Task 2.3: Complex SQL Queries (10 points)

Write SQL queries for the following scenarios:

Query 1 (3 points): Find all students who are enrolled in more than 3 courses in the current semester (Spring 2026).

Query 2 (3 points): List all courses with fewer than 5 enrolled students, including courses with no enrollments. Show course code, course name, and enrollment count.

Query 3 (4 points): For each instructor, display their name, department, and the average grade of all students in their courses. Only include instructors who teach at least 2 courses and have an average grade above 75.

Part 3: Stored Procedures and Triggers (20 points)

Task 3.1: Stored Procedure (10 points)

Create a stored procedure called `RegisterStudentForCourse` that: - Takes parameters: student_id, course_id, semester, year - Checks if the student has completed all prerequisites - Checks if the course has available capacity - Enrolls the student if both conditions are met - Returns appropriate error messages if enrollment fails

Include error handling and comments explaining your logic.

Task 3.2: Database Trigger (10 points)

Create a trigger that: - Fires when a grade is updated in the Enrollments table - Automatically updates a student's GPA in a separate STUDENT_GPA table - Calculate GPA using the formula: GPA = (sum of grade_points * credits) / total_credits - Grade points: A=4.0, B=3.0, C=2.0, D=1.0, F=0.0

You may need to create the STUDENT_GPA table first.

Part 4: Problem-Solving and Optimization (15 points)

Scenario:

The university database has grown to 50,000 students and 2,000 courses. Users are complaining that the following query is very slow:

```
SELECT s.first_name, s.last_name, c.course_name, e.grade
FROM Students s
JOIN Enrollments e ON s.student_id = e.student_id
JOIN Courses c ON e.course_id = c.course_id
WHERE e.semester = 'Spring'
  AND e.year = 2026
```

```
  AND s.major = 'Computer Science'
ORDER BY s.last_name, c.course_name;
```

Questions:

a) Query Analysis (5 points) - Use EXPLAIN or EXPLAIN ANALYZE to analyze this query - Identify potential performance bottlenecks

b) Optimization Strategy (7 points) - Propose at least 3 specific optimizations - Explain how each optimization would improve performance - Provide the modified query or index creation statements

c) Trade-offs (3 points) - Discuss any trade-offs of your optimization approach - What scenarios might make your optimizations less effective?

Part 5: Documentation and Report (Bonus +10 points)

Create a comprehensive report (3-5 pages) that includes:

1. **System Overview**
   - Purpose and scope of the database
   - Key design decisions and justifications
2. **Data Dictionary**
   - Complete list of all tables
   - Column descriptions and data types
   - Relationships and constraints
3. **Testing Plan**
   - Test cases for stored procedures
   - Expected vs actual results
   - Screenshots of successful executions
4. **Reflection**
   - Challenges encountered and solutions
   - What you learned
   - Potential improvements or extensions

Submission Guidelines

Deliverables:

1. **Theory_Answers.pdf** - Answers to Part 1 questions
2. **ER_Diagram.pdf** - Your ER diagram (Part 2.1)
3. **database_schema.sql** - All CREATE TABLE statements
4. **sample_data.sql** - All INSERT statements
5. **queries.sql** - All SELECT queries from Part 2.3
6. **procedures_triggers.sql** - Stored procedures and triggers from Part 3
7. **optimization_report.pdf** - Analysis and solutions from Part 4

8. **documentation.pdf** - Complete report (if attempting bonus)
9. **README.txt** - Instructions for running your SQL scripts

Package all files in: `StudentID_A6_DatabaseProject.zip`

Grading Rubric

Component	Points	Description
Theoretical Questions	25	Accuracy, completeness, clear explanations
Database Design	10	ER diagram completeness, proper notation
SQL Implementation	20	Correct syntax, proper constraints, sample data
Complex Queries	10	Correct logic, efficient queries, proper joins
Stored Procedures	10	Functionality, error handling, code quality
Triggers	10	Correct implementation, proper trigger logic
Optimization	15	Analysis depth, practical solutions, explanations
Total	**100**	
Bonus Documentation	**+10**	Professional quality, completeness

---

Technical Requirements

- **Database System:** MySQL 8.0+, PostgreSQL 12+, or SQL Server 2019+
- **SQL Dialect:** Specify which database system you're using
- **File Encoding:** UTF-8
- **Comments:** Include comments in all SQL files

Resources

- SQL Tutorial: https://www.w3schools.com/sql/
- Database Normalization: https://www.studytonight.com/dbms/database-normalization.php

- Query Optimization: https://use-the-index-luke.com/

Academic Integrity

This is an individual assignment. While you may discuss concepts, all SQL code and written answers must be your own original work. Plagiarism will result in zero points and potential disciplinary action.

Important Notes: - Test all SQL scripts before submission - Ensure your scripts run without errors on a fresh database - Include DROP TABLE IF EXISTS statements at the beginning of your schema file - Late submissions: -10% per day

Good luck with your database project!