MACQUARIE
University

FACULTY OF SCIENCE AND ENGINEERING

SCHOOL OF COMPUTING

COMP2350/COMP6350:

Database Systems

Session 2, 2024

Assignment Three: Procedural Programming

DUE DATE: 11:55 pm Sunday 20th October 2024 (Week 11)
Total Marks: 100 (10% of the final grade)
Objectives: ULO1, ULO2, ULO3 and ULO4 (refer to Unit Guide)

This is an individual assignment. Part A has 2 database programming and implementation tasks. Part B has a scenario-based task on database triggers and test of Part A (Task 2 and Task 3).

General requirements for all tasks *(when applicable):*

1. Complete the table below and include it at the beginning of each required document listed on page 4 in the Submission section:

Unit Code		Assignment#	3
Student ID Number		Student Name	
Tutor's Name		Workshop Date/Time	

2. Font Size must be 11 or 12 points.
3. Line Spacing must be single or 1.5 lines
4. Digitalized hand-written or hand-drawn contents will not be accepted for assessment.
5. Generative AI tools are forbidden in this unit.
6. Late submission penalty: see COMP2350/COMP6350 Unit Guide for 2024 S2.

Assignment Background

ABC education provider is planning to automate the manual process for fees management, unit enrolment, and student performance. As a student of the COMP2350, you are tasked to design a database solution for an organization that provides academic and certification courses. You are required to help them with database implementation by creating stored functions and stored procedures to handle scenarios involving *fees management*, *unit enrolment*, and *student performance* (Part A). In Part B, you are tasked to provide answers related to *trigger* and to provide test-cases for Part A. Business rules that are relevant to your tasks are provided below.

Business Rules:

BR1. A unit can only be registered in if seats are available, and the student is not already enrolled in the same unit in the same semester.
BR2. A student must have paid all outstanding fees[1] from previous semester[2] before they can register for a new unit.
BR3. Students can enrol in up to 6 units per semester, with no more than 2 advanced-level units.
BR4. If a student drops out after the no-penalty deadline, a dynamic penalty fee (i.e., $600 penalty for advanced units and $400 otherwise) based on the unit level is charged.
BR5. A student's overall performance, reflected by their average grade across all completed units, determines their eligibility for enrolling in advanced-level units.
BR6. If a student has an unpaid balance from previous semester or has received a score below 50 marks in more than 2 units, they will not be allowed to register for any advanced-level unit.

1. For students with **no outstanding fees**, the status will be marked as **"paid"**. If there are any outstanding fees (such as tuition fee, late unit enrolment fee, dropped unit penalty), the status will be marked as **"unpaid"**.

2. In the context of this assignment, **"previous semester"** does not necessary refer to the immediate previous semester.

Part A: Database Programming and Implementation

Use what has been covered during week 5 – week 11 (both lecture and workshop, including relevant textbook chapters) on SQL and Procedural Programming to complete all the tasks. If you use anything that was not covered during week 5 – week 11, your implementation for that task will not be marked.

TASK 1: (10 marks)

Use the provided **.sql** file to create and populate required tables.

> **Note:** For all tasks of this assignment, do not delete or modify any provided data (in the **.sql** file). If more data is required to complete the task, insert appropriate data into relevant table(s) when need.

For Tasks 2, 3, &4, assume each advanced unit has a flat rate of $4000 and $3000 for each non-advanced unit. Also, if a student earns a credit after dropping a unit, the credit will be banked into the student's banking account automatically handled by a subsystem that is out of the scope of this assignment (i.e., you do not need to worry about how to handle credits)

TASK 2: (20 marks)

Write a **stored function** that determines a student's total outstanding balance, including payment for late penalty and dropped units.

TASK 3: (40 marks)

Write a **stored procedure** that implements business rules BR6, BR2, BR4 such that the following requirements are met:

1. Prevent students with an outstanding balance or more than 2 failed units from enrolling in advanced-level units (BR6),
2. Enforce the unit registration **based on outstanding balance from previous semester** (BR2),
3. Automatically apply a penalty for dropped units, if dropped after the no-penalty deadline (BR4),
4. Error handler must be implemented to handle exceptions.

*** note: If a non-current student to register in a unit, the student will become a current student. However, for the purpose of this assignment, you are not required to implement this.**

Part B: Database Trigger and Test Cases Design

TASK 4: (20 marks)

It has been noticed that a large number of students drop out of units after the no-penalty deadline, causing problems in maintaining unit availability for other students. You are asked to design a solution to

- Automatically adjust student's outstanding balance, so that if a student drops out of a unit, any fees or penalties are automatically adjusted.

You will need to apply necessary penalties (as mentioned in BR4) for dropping out after the no-penalty deadline.

Note: This task does **not** require you to update the table structure and implement the trigger. Instead, you are asked to design and briefly explain the logic behind the trigger. This means you should focus on describing how the trigger would work, rather than writing SQL code. (**Max. length half A4 page**)

TASK 5: (10 marks)

Think about what data you would need to test different data scenarios for Task 2 and Task 3. Outline those scenarios in a section called "Test Plan" of your submission to help you verify the correctness and completeness of these tasks (i.e., Task 2 and Task 3). Add necessary data into relevant table(s) for testing all possible cases that you can think of.

Submission:

- A **.sql** file
  - Submit a sql text file consisting of all programming and implementation for Tasks 1, 2, 3 and 5.
- A **.pdf** file
  - Submit a pdf document for:
    - Task 1 new data (if inserted)
    - Task 4
    - Task 5 test cases, Including the screenshots as evidence and brief justification.

Marking Rubric

Tasks/Grades	HD	D	CR	P	F	Not Attempted
Task 1: Populate Tables (10 Marks)	**8.5-10 Marks:** Tables created with no errors, and additional data inserted logically. Constraints maintained.	**7.5-8.4 Marks:** Tables created correctly with minor issues in inserted data.	**6.5-7.4 Marks:** Tables created, but some data insertion or constraint issues.	**5-6.4 Marks:** Tables created but issues in data insertion.	**0-4.9 Marks:** Less than 50% of total marks (e.g., incomplete tables or significant issues).	**0 Mark**
Task 2: Stored Function (20 Marks)	**17-20 Marks:** Function calculates outstanding balance and penalties correctly, with efficient logic.	**15-16 Marks:** Function works with minor issues in penalty logic or performance.	**13-14 Marks:** Function works but with multiple issues in penalty application or performance.	**10-12 Marks:** Function incomplete or significant issues in logic.	**0-9 Marks:** Less than 50% of total marks (e.g., major issues or not functional).	**0 Mark**
Task 3: Stored Procedure (40 Marks)	**34-40 Marks:** Procedure correctly handles all rules with robust error handling.	**30-33 Marks:** Procedure works with minor issues, meets most requirements.	**26-29 Marks:** Procedure has some functionality but misses key rules or error handling.	**20-25 Marks:** Procedure incomplete, missing several business rules or error handling.	**0-19 Marks:** Less than 50% of total marks (e.g., incomplete procedures or major deficiencies).	**0 Mark**
Task 4: Trigger Design (20 Marks)	**17-20 Marks:** Clear, detailed explanation of trigger logic, covering all relevant rules and penalties.	**15-17 Marks:** Logical explanation, missing minor details or clarity.	**13-14 Marks:** Trigger logic mostly correct but missing key elements or details.	**10-12 Marks:** Trigger explanation incomplete or unclear.	**0-9 Marks:** Less than 50% of total marks (e.g., inadequate explanation or major issues).	**0 Mark**
Task 5: Test Cases (10 Marks)	**8.5-10 Marks:** Comprehensive test cases, covering all edge cases and appropriate data insertion.	**7.5-8.4 Marks:** Test cases cover most scenarios with minor gaps in coverage.	**6.5-7.4 Marks:** Test cases cover basic scenarios, missing some important cases.	**5-6 Marks:** Test cases incomplete, missing key scenarios or improper data insertion.	**0-4.9 Marks:** Less than 50% of total marks (e.g., significant gaps in testing).	**0 Mark**