Information School
UNIVERSITY of WASHINGTON

INFO 330: DATABASE SYSTEMS AND DATA MODELING

Assignment #6: Working with CTEs and Database Objects

Due Date: Monday, December 4, 2023 (11:59pm) Marks: 70 (10@Question)

Learning Objectives

- Demonstrate ability to write SQL CTEs and code database objects (stored procedures, user-defined functions, and triggers) to effectively answer specific questions.
- Demonstrate ability to customize SQL queries to present results in a meaningful manner (e.g., use of aliases, descriptive names, etc.).

Part 1: Common Table Expressions (CTEs)

Please use the following relational schema to answer questions 1 to 3.

Students (**student_id**, first_name, last_name)
Courses (**course_code**, course_name, course_credits)
Exams (**exam_code**, **exam_date**, course_code, student_id, score)

The student_id and course_code are the primary keys in the Students and Courses tables, respectively, and foreign keys in the Exams table. The exam_code and exam_date are the composite primary key in the Exams table. The score attribute records the score a given student obtained in each exam.

Note: There is no implemented database; so, write your SQL queries using the schema above.

What you should do

Unless otherwise stated, please write SQL statements to answer the following questions. Please use Word or any text or SQL editor.

1. Using a CTE, write a query to display a student's last name and first name, total course credits the student has completed, and the ratio of the credits a student has completed to the average of total course credits completed by all the students. Note: a CTE should be followed by a main SELECT query to display the required information.

2. Using CTEs, write a query to display course name and the proportion of students whose average score for the given subject is above the overall average score for the course. Note: a CTE should be followed by a main SELECT query to display the required information.

3. Using CTEs, write a query to display course name, the minimum, maximum, and average score across all exams for the course, and the overall minimum, maximum, and average score across all exams and across all courses. Note: a CTE should be followed by a main SELECT query to display the required information.

Part 2: Coding Database Objects (Stored Procedures, User-Defined Functions, and Triggers)

For this lab exercise, please use the College_db_bk database on the class server. The relational schema of the College_db_bk database are given below.

Note: I got this database from some source. I do not necessarily agree with the design, but we are going to use it "as is" for the sake of our class work. Note that the tuition is more of a lookup table and is not linked (connected) to any table in the database. As far as tuition is concerned, in addition to either part-time or full-time tuition, it is assumed that students pay per unit (or per credit) cost.

Departments (**DepartmentID**, DepartmentName)
Instructors (**InstructorID**, LastName, FirstName, Status, DepartmentChairman, HireDate, AnnualSalary, DepartmentID)
Courses (**CourseID**, CourseNumber, CourseDescription, CourseUnits, DepartmentID, InstructorID)
Students (StudentID, Lastname, FirstName, EnrollmentDate, GraduationDate)
StudentCourses (**StudentID, CourseID**)
Tuition (PartTimeCost, FullTimeCost, PerUnitCost)

What you should do

- Unless otherwise stated, please write SQL statements to answer the following questions using the College_db_bk from the class server.
- Due to some restrictions about working with database objects, begin by creating a backup database for College_db_bk that you should use for this exercise. Creating a backup database is a 2-step procedure in both SQL Server Management Studio (SSMS) and Azure Data Studio (ADS):
    - **Step 1**: backup the database: in SSM right-click the database you want to backup (in this case College_db_bk) and then select Tasks and then Back Up. Leave the defaults shown and click OK. In ADS, right-click the name of the database you want to backup and choose Backup from the drop-down menu. Leave the defaults intact and click Backup.
    - **Step 2**: Restore the backup copy. Right-click on database name again, then Tasks, and then Restore, then Database. Leave the default entry for Source database as entered. However, change the name destination database to College_db_bk_init (where init will be your initials). Under "Backup sets to restore" make sure that the checkbox below Restore is CHECKED and then click OK. The steps are similar in Azure Data Studio. Right-click the database name and then select Restore.
    - **Please the backup copy of the database you created to answer the questions below**.

- Note: Views, User_defined Functions, Stored Procedures, and Triggers are database objects stored in the database. Therefore, their names must be unique in the database.
- To ensure that each database object you create is unique in the database, please replace the suffix in the object name (init) with your initials. For example, looking at 4 a), my object would be named DepartmentInstructors_so (obviously, it should be your initial and I hope there are no 2 students in the class with the same initials – please use the first letter of your first name, middle name, and last name as initials (if you don't have a middle name, use the first 2 letters of your first name and then the first letter of your last name. If you still get an error, add the last 3 digits of your student ID to your initial).
- Since your coding queries for an implemented database, please take screenshots of your query results/outputs and include them with the SQL code.

4. a). Create a view named DepartmentInstructors_init that returns these columns: the DepartmentName column from the Departments table and the LastName, FirstName, Status, and AnnualSalary columns from the Instructors table.

b). Write a SELECT statement that returns all the columns from the DepartmentInstructors_init view that you created in part a) above. Limit the output to fulltime instructor in the English department only.

c). Write an UPDATE statement that updates the DepartmentInstructors_init view you created in part a) above so it increases the annual salary for each fulltime instructor in the English department by 10%. Then, run the SELECT statement you wrote in part b) to be sure this worked correctly.

5. Write a script that creates a function named *fnStudentUnits_init* that calculates the total course units of a student in the **StudentCourses** table. To do that, this function should accept one parameter for the student ID, and it should return the sum of the course units for the student.

   Within your script, write a SELECT statement that calls the function and should return the StudentID from the StudentCourses table, the CourseNumber and CourseUnits from the Courses table, and the value returned by the fnStudentUnits_init function for that student.

6. Write a script that creates and calls a stored procedure named *spUpdateInstructor_init* that updates the **AnnualSalary** column in the **Instructors** table. This procedure should have one parameter for the instructor ID and another for the annual salary. If the value for the AnnualSalary column is a negative number, the stored procedure should raise an error that indicates that the value for this column must be a positive number.

   Code at least two EXEC statements that test this procedure.

7. Create a trigger named *Instructors_INSERT_init* that inserts the current date for the *HireDate* column of the **Instructors** table if the value for that column is null.

   Test this trigger with an appropriate INSERT statement.