Project 1

**My College Database**

In this project, you will use SQL Server Management Studio to create MyCollege database and to write and enter SQL statements and run them against this database.

**A. Database Setup [10 pts.]**

* Download sql files from this projcet tab and open CreatMyCollege.sql in SQL server management studio. Execute the entire script and show the message in the Message tab, indicating the script is executed successfully.
* **Navigate through the database objects and view the column definitions for each table. Open a new Query Editor window. Show details in Courses table and Instructors table using SELECT statement.**
* Open another Query Editor window and then enter and run this statement:

SELECT COUNT(\*) AS NumberOfInstructors FROM Instructors

* Open the script named InstructorDetails.sql. Note that this script contains just one SQL statement. Then, run the statement.
* Open the script named InstructorSummary.sql. Note that this opens another Query Editor window.
* Open the script named InstructorStatements.sql. Note that this script contains two SQL statements that end with semicolons.
* Press the F5 key or click the Execute button to run both statements in this script. Note that this displays the results in two Results tabs. Make sure to view the results of both SELECT statements.
* Exit from SQL Server Management Studio.

**B. Essential SQL Skills [60 pts.]**

* Write a SELECT statement that returns all of the columns from the Courses table. Then, run this statement to make sure it works correctly.
* Write a SELECT statement that returns one column from the Students table named FullName that joins the LastName and FirstName columns. Format this column with the last name, a comma, a space, and the first name. Sort the result set by last name in ascending sequence. Return only the students whose last name begins with a letter from A to M.
* Write a SELECT statement that returns these column names and data from the Instructors table:
  + LastName The LastName colum
  + FirstName The FirstName column
  + HireDate The HireDate column

Return only the rows with a hire date that’s in 2019. Sort the result set in ascending sequence by the HireDate column.

* Write a SELECT statement that returns these column names and data from the Students table:
  + FirstName The FirstName column
  + LastName The LastName column
  + EnrollmentDate The EnrollmentDate column
  + CurrentDate The current date
  + MonthsAttended A column that’s calculated by getting the difference between the enrollment date and the current date

Sort the result set in ascending sequence by the MonthsAttended column.

* Write a SELECT statement that returns these column names and data from the Instructors table:
  + FirstName The FirstName column
  + LastName The LastName column
  + AnnualSalary The AnnualSalary column

Return only the top 20 percent of instructors based on annual salary.

* Write a SELECT statement that returns these columns and data from the Tuition table, along with a constant value and two calculated values:
  + FullTimeCost The FullTimeCost column
  + PerUnitCost The PerUnitCost column
  + Units 12
  + TotalPerUnitCost A column that’s calculated by multiplying the per unit cost by the units
  + TotalTuition A column that’s calculated by adding the full time cost to the total per unit cost
* Write a SELECT statement that joins the Courses table to the Departments table and returns these columns: CourseNumber, CourseDescription, DepartmentName. Sort the result set by DepartmentName and then by CourseNumber in ascending order.
* Write a SELECT statement that joins the Instructors table to the Courses table and returns these columns: LastName, FirstName, CourseNumber, CourseDescription. Return all courses for each instructor with a status of “P” (part time). Sort the result set by LastName and then by FirstName in ascending order.
* Use the UNION operator to generate a result set consisting of five columns from the Students table:
  + Status A calculated column that contains a value of UNDERGRAD or GRADUATED
  + FirstName The FirstName column
  + LastName The LastName column
  + EnrollmentDate The EnrollmentDate column
  + GraduationDate The GraduationDate column

If the student doesn’t have a value in the GraduationDate column, the Status column should contain a value of UNDERGRAD. Otherwise, it should contain a value of GRADUATED.

Sort the final result set by EnrollmentDate.

* Write a SELECT statement that returns these columns:
  + InstructorDept The DepartmentName column from the Departments table for a related instructor
  + LastName The LastName column from the Instructors table
  + FirstName The FirstName column from the Instructors table
  + CourseDescription The CourseDescription column from the Courses table
  + CourseDept The DepartmentName column from the Departments table for a related instructor

Return one row for each course that’s in a different department than the department of the instructor assigned to teach that course.

(Hint: You will need to join the Departments table to both the Instructors table and the Courses table, which will require you to use table aliases to distinguish the two tables.)

* Write a SELECT statement that returns one row for each instructor that has courses with these columns:
  + The instructor first and last names from the Instructors table in this format: John Doe (Note: If the instructor first name has a null value, the concatenation of the first and last name will result in a null value.)
  + A count of the number of courses in the Courses table
  + The sum of the course units in the Courses table

(Hint: You will need to concatenate the instructor first and last names again in the GROUP BY clause.)

Sort the result set in descending sequence by the total course units for each instructor.

* Write a SELECT statement that answers this question: What is the total number of courses taught by parttime instructors? Return these columns:
  + The instructor last name and first name from the Instructors table in this format: Doe, John (Note: If the instructor first name has a null value, the concatenation of the first and last name will result in a null value.)
  + The total number of courses taught for each instructor in the Courses table

Use the ROLLUP operator to include a row that gives the grand total.

* Write a SELECT statement that returns the same result set as this SELECT statement, but don’t use a join. Instead, use a subquery in a WHERE clause that uses the IN keyword.

SELECT DISTINCT LastName, FirstName

FROM Instructors i JOIN Courses c

ON i.InstructorID = c.InstructorID

ORDER BY LastName, FirstName

* Write a SELECT statement that returns one row for each course with these columns:
  + The CourseID column from the Courses table
  + The most recent enrollment date for that course from the Students table

Change the SELECT statement to a CTE. Then, write a SELECT statement that returns one row per course that shows the CourseDescription for the course and the LastName, FirstName, and EnrollmentDate for the student with the most recent enrollment data.

* Write an INSERT statement that adds this row to the Departments table:
  + DepartmentName: History

Code the INSERT statement so SQL Server automatically generates the value for the DepartmentID column.

* Write a single INSERT statement that adds these rows to the Instructors table:
  + InstructorID: The next automatically generated ID  
    LastName: Benedict  
    FirstName: Susan  
    Status: P  
    DepartmentChairman: 0  
    HireDate: Today’s date  
    AnnualSalary: 34000.00  
    DepartmentID: 9
  + InstructorID: The next automatically generated ID  
    LastName: Adams  
    FirstName: null  
    Status: F  
    DepartmentChairman: 1  
    HireDate: Today’s date  
    AnnualSalary: 66000.00  
    DepartmentID: 9

Write this statement without using a column list.

* Write an UPDATE statement that modifies the first instructor you added in the above question. This statement should change the AnnualSalary column from 34,000 to 35,0000, and it should use the InstructorID column to identify the row.
* Write a DELETE statement that deletes the second instructor you added in question 21. This statement should use the InstructorID column to identify the row.
* Write a DELETE statement that deletes the row in the Departments table that has an ID of 9. When you execute this statement, it will produce an error since the department has related rows in the Instructors table. To fix that, precede the DELETE statement with another DELETE statement that deletes all instructors in this department.
* Write an UPDATE statement that increases the annual salary for all instructors in the Education department by 5%. To do that, join the Departments and Instructors tables and then filter the rows by the department name.
* Write a DELETE statement that deletes instructors that aren’t teaching any courses. To do that, use a subquery in the WHERE clause.
* Open the script named CreateGradStudents.sql that’s attached above. Run this file to create a table named GradStudents. This table has the same columns as the Students table, but the StudentID column isn’t defined as an identity column.
* Write an INSERT statement that inserts rows from the Students table into the GradStudents table. Include only the rows for students that have graduated, and don’t use a column list.
* Open the script named CreateMyCollege.sql. Then, run this script. That should restore the data that’s in the database. If an error message is displayed indicating that the database is in use, you’ll need to close and restart the Management Studio and then run the script again.
* Write a SELECT statement that returns these colums from the Students table:
  + A column that uses the CONVERT function to return the EnrollmentDate column in this format: MM/DD/YYYY. In other words, use 2-digit months and days and a 4-digit year, and separate each date component with slashes.
  + A column that uses the CONVERT function to return the EnrollmentDate column with the date, and the hours and minutes on a 12-hour clock with an am/pm indicator.
  + A column that uses the CONVERT function to return the EnrollmentDate column with just the time in a 24-hour format, including the milliseconds.
  + A column that uses the CONVERT function to return the EnrollmentDate column with just the month and day.

**C. Advanced SQL Skills (views/stores procedures/ functions / scripts) [20 pts.]**

1. Create a view named DepartmentInstructors that returns these columns: the DepartmentName column from the Departments table and the LastName, FirstName, Status, and AnnualSalary columns from the Instructors table.
2. Write a SELECT statement that returns all the columns from the DepartmentInstructors view that you created in exercise 1.
3. Return one row for each fulltime instructor in the English department.
4. Write an UPDATE statement that updates the DepartmentInstructors view you created in exercise 1 so it increases the annual salary for each fulltime instructor in the English department by 10%. Then, run the SELECT statement you wrote in exercise 2 to be sure this worked correctly.
5. Create a view named StudentCoursesMin that returns these columns: the FirstName and LastName from the Students table and the CourseNumber, CourseDescription, and CourseUnits from the Courses table.
6. Write a script that declares a variable and sets it to the count of all students in the Students table that haven’t graduated. If the count is greater than or equal to 100, the script should display a message that says, “The number of undergrad students is greater than or equal to 100”. Otherwise, it should say, “The number of undergrad students is less than 100”.
7. Write a script that uses two variables to store (1) the count of all of the instructors in the Instructors table and (2) the average annual salary for those instructors. If the instructor count is greater than or equal to 10, the script should print a message that displays the values of both variables. Otherwise, the script should print a message that says, “The number of fulltime instructors is less than 10”.
8. Write a script that attempts to delete the department with the name ‘Sociology’ from the Departments table. If the delete is successful, the script should display this message:

SUCCESS: Record was deleted.

If the delete is unsuccessful, the script should display a message something like this:

FAILURE: Record was not deleted.

Error 547: The DELETE statement conflicted with the REFERENCE constraint "FK\_\_Instructo\_\_Depar\_\_267ABA7A". The conflict occurred in database "MurachCollege", table "dbo.Instructors", column 'DepartmentID'.

1. Write a script that creates and calls a function named fnStudentUnits 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.
2. The SELECT statement that calls the function should return the StudentID from the StudentCourses table, the CourseNumber and CourseUnits from the Courses table, and the value return by the fnStudentUnits function for that student.

**D. Database Design [10 pts.]**

* Create a database diagram that shows the relationships between the six tables in the MyCollege database. You can use any digital tool to draw your diagrams. "Resources and Links" tab on this Blackboard page lists several tools for drawing E/R diagrams.
* Design a database diagram for a database that stores information about the downloads that users make from a book website.
  + Each user must have an email address, first name, and last name.
  + Each user can have one or more downloads.
  + Each download must have a filename and download date/time.
  + Each book can be related to one or more downloads.
  + Each book must have a name.

**Project Report**

Please use proper fonts (such as 12 pts) and pagination (page numbers, header/footer), and follow the same section (A, B, ...) and query numbering (1, 2, ...) as in this project description.

Please include the following in your typed report and submit it as a single PDF file:

* A cover page, a short abstract, and a table of contents.
* Screenshots for the actions taken for Part A.
* Complete SQL source codes with comments in typed format (no screenshots) and full screenshots of the requested actions and the corresponding results for Part B and C.
* Your diagrams and explanations for Part D.
* Your overall remarks about this project, such as experiences and insights gained.