Data 331

Fall 2023

Team Project

**Team Final Programming Assignment**

Write the SQL commands required to accomplish the following in your team project database. Submit all of your commands in a SQL file (a file with a .sql extension) when complete. When submitting your file, label your scripts with the corresponding question number

1. Create foreign keys to establish the following relationships in your database:
   * Student to Faculty Advisor
   * Course to Course Offering
   * Student to Enrollment
   * Course Offering to Enrollment
   * Course Offering to Faculty Instructor
   * Student to Grades
   * Department to Faculty Chair
   * Department to Major
2. Add a column to the Grades table to store the equivalent Grade Points. Write update statements to populate these values using the following conversion table:

|  |  |  |
| --- | --- | --- |
| A+ | = | 4.33 |
| A | = | 4.00 |
| A- | = | 3.67 |
| B+ | = | 3.33 |
| B | = | 3.00 |
| B- | = | 2.67 |
| C+ | = | 2.33 |
| C | = | 2.00 |
| C- | = | 1.67 |
| D+ | = | 1.33 |
| D | = | 1.00 |
| D- | = | 0.67 |
| F | = | 0.00 |

1. Add a column to the Student table to store the student’s current, cumulative GPA as a decimal 3 digits in length with 2 digits of scale. Write scripts to calculate current GPA and update this column for all students. Current GPA can be calculated as follows:
2. Multiply each class grade by the number of credits for a course. For example: 3.0 (grade) x 4.0 (course credits) = 12. This is the number of grade points earned.
3. Divide this number by total credits attempted.

Hint: Use multiple aggregate functions in your query to sum the total number of credits attempted and total grade points earned. You can apply mathematical operators to aggregated values

1. Create a view to show average GPA per class level.
2. Create a view to show average GPA per major.
3. Create a view to show the total number of students enrolled in the courses for each department for each term.
4. Create a view to show the number of students per major. Order the view by the number of students in descending order.
5. Create a view to show enrollment by state for each term. Order the view by term, then by course enrollment in descending order, so the courses with the most enrolled students are first for each term.
6. Create a view to show the count, total salary, and average salary for active, full-time faculty for each department.
7. Create an index or constraint to ensure that all student e-mail addresses are unique.
8. Create an index or constraint to ensure that all grades are one of the valid values in the table above, plus I for incomplete courses.
9. Create an index or constraint to ensure that all students have a grade level of Freshman, Sophomore, Junior or Senior.
10. Create an index or constraint to ensure that all students have a status of Active, Inactive or Alumni.
11. Create an index or constraint to ensure that Course Name is unique for all courses.
12. Create an index or constraint to ensure that all faculty e-mail addresses are unique
13. Create an index or constraint to ensure that faculty status values are either Active, Terminated, or Emeritus.
14. Create a stored procedure to insert a single student record. The stored procedure should take as input all fields for a student record. Newly inserted student records should be in a status of Active, and if the procedure tries to insert a record with an invalid major or faculty advisor, the stored procedure should not perform the insert.
15. Create a stored procedure to update a single student record. The stored procedure should have all student fields as input parameters, and should perform the same validation on major and faculty advisor that your insert stored procedure does.
16. Create a stored procedure to insert a single faculty record. The stored procedure should take as input all fields for a faculty record. Newly inserted student records should be in a status of Active. The procedure should check the following requirement:

* Only Full Time faculty can be a Chair, Advisor or Tenured.

If the input parameter values do not meet this requirement, the procedure should not perform the insert.

1. Create a stored procedure to update a single faculty record. The stored procedure should take as input all fields for a faculty record. The procedure should check the following requirements:

* Only Full Time faculty can be a Chair, Advisor or Tenured.
* Faculty with an inactive status cannot be an advisor, chair, instructor, full time or tenured.

If the input parameter values do not meet these requirements, the procedure should not perform the update.

1. Create a stored procedure to insert a single course record. The stored procedure should take as input all fields for a course record. The procedure should check the following requirements:
   * There can only be one active course with a specific course number

If the input parameter values do not meet these requirements, the procedure should not perform the insert.

1. Create a stored procedure to update a single course record. The stored procedure should take as input all fields for a course record. The procedure should check the following requirements:
   * There can only be one active course with a specific course number

If the input parameter values do not meet these requirements, the procedure should not perform the update

1. Create a stored procedure to insert a course offering record. The stored procedure should take as input all fields for a course offering. The procedure should check the following requirements:
   * The combination of course, academic year, term and section is unique

If the input parameter values do not meet these requirements, the procedure should not perform the insert.

1. Create a stored procedure to update a course offering record. The stored procedure should take as input all fields for a course offering. The procedure should check the following requirements:
   * The combination of course, academic year, term and section is unique

If the input parameter values do not meet these requirements, the procedure should not perform the update.

1. Create a stored procedure to update all enrollment records for the current term that have a status of Enrolled to a status of Complete. This procedure will be used to mark enrollments in a course as complete at the end of an academic term. The stored procedure should take as input the value(s) needed to identify an academic term.
   * If the academic term is invalid, the stored procedure should not execute the update.
   * The stored procedure should only update records with a current status of Enrolled.
2. Create a stored procedure to update a grade status to Final. This procedure will be used to mark grades as final for students. The stored procedure should take as input the fields required to uniquely identify a grade record.
   * If the current grade is I (Incomplete) the procedure should not perform the update.