|  |  |
| --- | --- |
| Julie Sweany | |
|  | Your Answers |
| **1** | Students\_With\_Scholarships |
| Text of your CREATE PROCEDURE  -- 12.1 - Julie Sweany  -- Create a stored procedure to list last and first names of all students receiving a scholarship  USE College;  DROP PROCEDURE IF EXISTS Students\_With\_Scholarships;  DELIMITER $$  CREATE PROCEDURE Students\_With\_Scholarships()  BEGIN  SELECT  LastName AS 'Last Name',  FirstName AS 'First Name',  Scholarship  FROM  Student  WHERE  Scholarship > 0  ORDER BY LastName;  END  $$  DELIMITER ; |
| Screen shot of the result grid from CALL Students\_With\_Scholarships() ; |
| **2** | Students\_Major |
| Text of your CREATE PROCEDURE  -- 12.2 - Julie Sweany  -- Create a stored procedure to list last and first names of all students in a particular major  USE College;  DROP PROCEDURE IF EXISTS Students\_Major;  DELIMITER $$  CREATE PROCEDURE Students\_Major(Major INT)  BEGIN  SELECT  LastName AS 'Last Name',  FirstName AS 'First Name',  Major.Name AS Major  FROM  Student  INNER JOIN Major ON Major.ID = MajorID  WHERE Major.ID = Major;  END  $$  DELIMITER ; |
| Screen shot of the result grid from CALL Students\_Major(1) ; |
| **3** | Student\_Add |
| Text of your CREATE PROCEDURE  -- 12.3 - Julie Sweany  -- Create a stored procedure to add a record for a student, manually entering first and last name, birthdate, sex, major, and scholarship, and automatically including email address and enrolled date  USE College;  DROP PROCEDURE IF EXISTS Student\_Add;  DELIMITER $$  CREATE PROCEDURE Student\_Add(First varchar(45), Last varchar(45), Birth date, Gender char(1), MajorCode int(11), ScholarshipValue decimal(9,2))  BEGIN  INSERT INTO Student (FirstName, LastName, EMail, DateOfBirth, Sex, EnrolledDate, MajorID, Scholarship)  VALUES (First, Last, CONCAT(First, Last, '@college.edu' ), Birth, Gender, current\_date(), MajorCode, ScholarshipValue) ;  END  $$  DELIMITER ; |
| Text of your CALL Student\_Add(…  CALL Student\_Add('Julie', 'Sweany', '1970-03-19', 'F', 1, 0.00) ; |
| Screen shot of the result grid listing the new student |

|  |  |
| --- | --- |
| **4** | Faculty\_GPA |
| -- 12.4 - Julie Sweany  -- Create a stored procedure to return the average grades across all sections taught by a particular professor.  USE College;  DROP PROCEDURE IF EXISTS Faculty\_GPA;  DELIMITER $$  CREATE PROCEDURE Faculty\_GPA(IN FacultyNumber INT, OUT outGPA DECIMAL(3,2))  BEGIN  DECLARE averageGPA DECIMAL(3,2) ;  SET averageGPA =  (SELECT AVG(Registration.Grade)  FROM Faculty JOIN Section ON Faculty.ID = Section.TaughtByID  JOIN Registration ON Section.ID = Registration.SectionID  WHERE Faculty.ID = FacultyNumber AND Registration.Grade > 0) ;  SET outGPA = averageGPA ;  END  $$  DELIMITER ; |
| Screen shot of the result grid showing average GPA |

|  |  |
| --- | --- |
| **5** | FUNCTION Student\_Section\_Count |
| Text of your CREATE FUNCTION  -- 12.5 - Julie Sweany  -- Create and call a function that returns the number of sections students have registered for.alter  USE College;  DROP FUNCTION IF EXISTS Student\_Section\_Count;  DELIMITER $$  CREATE FUNCTION Student\_Section\_Count(StudentID INT)  RETURNS INT  BEGIN  DECLARE returnSections INT(11) ;  SET returnSections =  (SELECT COUNT(\*) FROM Section JOIN Registration ON Section.ID = Registration.SectionID JOIN Student ON Student.ID = Registration.StudentID  WHERE Student.ID = StudentID);  RETURN returnSections ;  END $$  DELIMITER ; |
| Screen shot of the result grid showing the call to the function |
| **6** | Do It Yourself Stored Procedure |
| Screen shot of the result grid showing the call to your stored procedure |
| Brief discussion of the purpose of your stored procedure. Tell me what it’s for, not how it works.  The procedure outputs the sections and courses offered during the Spring 2017 semester for a particular department. Also include last name of faculty. |
| Text of your CREATE PROCEDURE  -- 12.6 - Julie Sweany  -- Create a stored procedure to list sections and courses offered in Spring 2017 within a particular major. Also include last name of the faculty member teaching the section.  USE College;  DROP PROCEDURE IF EXISTS Offerings\_By\_Department;  DELIMITER $$  CREATE PROCEDURE Offerings\_By\_Department(DepartmentIn INT)  BEGIN  SELECT  Department.Name AS Department,  Course.Name AS Course,  Section.Name AS Section,  Faculty.LastName AS Faculty,  Semester.Name AS Semester  FROM  Semester  JOIN  Section ON Semester.ID = SemesterID AND Semester.ID = 1  JOIN  Course ON Course.ID = CourseID  JOIN  Department ON Course.DepartmentID = Department.ID  JOIN  Faculty ON Section.TaughtByID = Faculty.ID  WHERE Department.ID = DepartmentIn;  END  $$  DELIMITER ; |