|  |  |
| --- | --- |
| Julie Sweany | |
|  | Your Answers |
| **1** | Before UPDATE trigger |
| Screen shot of the result grid from SELECT statement shown in the homework |
| SQL that creates your trigger, including the UPDATE statement  -- 14.1 - Julie Sweany  -- Write insert trigger to check that email for new entries into the faculty table is in standard form  DROP TRIGGER IF EXISTS Faculty\_Before\_Insert;  DELIMITER $$  CREATE TRIGGER Faculty\_Before\_Insert  BEFORE INSERT ON Faculty  FOR EACH ROW  BEGIN  SET NEW.Email =  CONCAT(NEW.FirstName, NEW.LastName, '@college.edu');  END $$  DELIMITER ;  -- Insert statement using trigger  USE College;  INSERT INTO Faculty (FirstName, LastName, EMail, Salary, HireDate, DepartmentID) values ('Julie', 'Sweany', 'bad@bad.bad', 90322, '2011-07-15', 1);  -- SELECT statement for new entry  SELECT  LastName, FirstName, Email  FROM  Faculty  WHERE  ID = LAST\_INSERT\_ID(); |
| **2** | Before UPDATE trigger |
| Screen shot of the Output Window showing the output both UPDATE statements |
| SQL that creates your trigger  -- 14.2 - Julie Sweany  -- Write trigger to prevent updating Student table with EnrolledDate set to a value in the future  DROP TRIGGER IF EXISTS Student\_Before\_Update;  DELIMITER $$  CREATE TRIGGER Student\_Before\_Update  BEFORE UPDATE ON Student  FOR EACH ROW  BEGIN  IF NEW.EnrolledDate > Current\_Date THEN  SIGNAL SQLSTATE 'HY000'  SET MESSAGE\_TEXT = 'Enrolled date may not be in the future';  END IF;  END $$  DELIMITER ; |

|  |  |
| --- | --- |
| **3** | Add another rule of your choosing |
| Screen shot of the Output Window showing the output with the error. |
| SQL that creates your trigger  -- 14.3 - Julie Sweany  -- Write trigger to prevent updating Student table with a negative value for Scholarship  DROP TRIGGER IF EXISTS Student\_Before\_Update;  DELIMITER $$  CREATE TRIGGER Student\_Before\_Update  BEFORE UPDATE ON Student  FOR EACH ROW  BEGIN  IF NEW.Scholarship < 0 THEN  SIGNAL SQLSTATE 'HY000'  SET MESSAGE\_TEXT = 'Scholarships cannot be a negative amount.';  END IF;  END $$  DELIMITER ; |

|  |  |
| --- | --- |
| **4** | After UPDATE trigger |
| Screen shot of the Output Window showing the output both UPDATE statements |
| SQL that creates your trigger  -- 14.4 - Julie Sweany  -- Write an after update trigger to create records for all changes to the sudent table  DROP TRIGGER IF EXISTS Student\_After\_Update;  DELIMITER $$  CREATE TRIGGER Student\_After\_Update  AFTER UPDATE ON Student  FOR EACH ROW  BEGIN  INSERT INTO SysLog  (TableName, Message, Created) VALUES  ( 'Student', CONCAT('Updated ID= ', OLD.ID ), NOW());  END $$  DELIMITER ; |

|  |  |
| --- | --- |
| **5** | Before Insert Trigger on Registration |
| Paste the text of your INSERT statement and a screen shot of the Output Window showing the output from two runs of the statement.  USE College;  INSERT INTO Registration  (StudentID, SectionID, Grade) VALUES (2, 11, 0); |
| SQL that creates your trigger  -- 14.5 - Julie Sweany  -- Write trigger to prevent duplication of student and section pairings in Registration table  DROP TRIGGER IF EXISTS Registration\_Before\_Insert;  DELIMITER $$  CREATE TRIGGER Registration\_Before\_Insert  BEFORE INSERT ON Registration  FOR EACH ROW  BEGIN  DECLARE StudentMatch INT;  DECLARE SectionMatch INT;  DECLARE Matches INT;  SELECT COUNT(StudentID) FROM Registration WHERE NEW.StudentID = StudentID INTO StudentMatch;  SELECT COUNT(SectionID) FROM Registration WHERE NEW.SectionID = SectionID INTO SectionMatch;  SELECT COUNT(New.StudentID) FROM Registration WHERE SectionID = NEW.SectionID INTO Matches;  IF Matches > 0 THEN  SIGNAL SQLSTATE 'HY000'  SET MESSAGE\_TEXT = 'Student is already registered for that section.';  END IF;  END $$  DELIMITER ; |