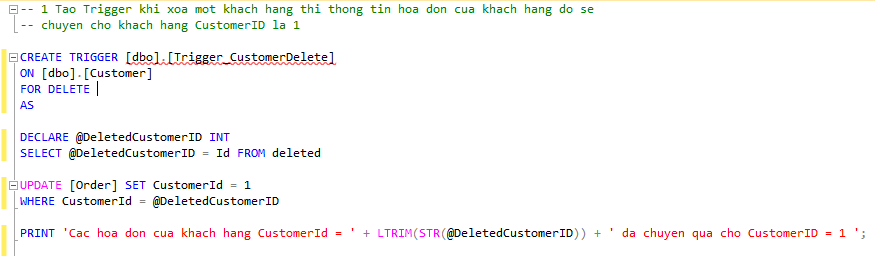
**LAB 7 – HQTCSDL – Trigger-Transaction-Cursor-Temp Table**

**1/ Sử dụng Trigger**

**Yêu cầu:** Viết ví dụ sử dụng trigger để kiểm soát dữ liệu

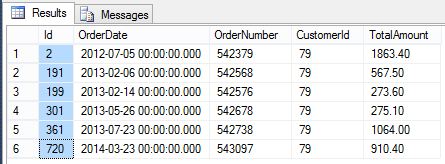
**Hướng dẫn:**

* Tạo một trigger để kiểm tra việc khi xóa một Customer thì thông tin Order của Customer đó sẽ chuyển về cho CustomerId là 1



* Kiểm tra xem khách hàng 79 có những hóa đơn nào





* Viết lại câu truy vấn khách hàng 79 thông qua OrderId



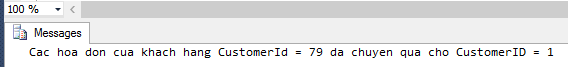
* Xóa bỏ Foreign Key Constraint để kiểm nghiệm trigger. Nếu không xóa thì constraint này sẽ thực thi trước trigger và không cho phép bạn xóa bất kỳ CustomerId nào



* Thực hiện lệnh xóa bỏ Customer thứ 79

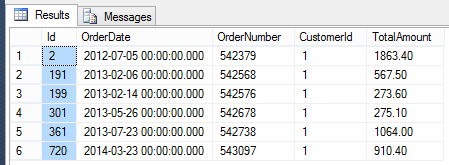


Ta có kết quả

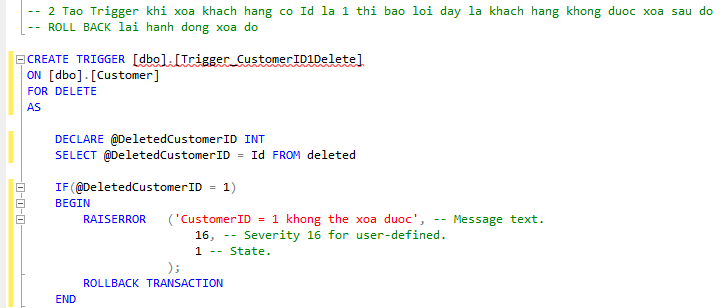


Truy vấn để kiểm tra các hóa đơn của Customer 79 ta sẽ thấy trigger đã tự động chuyển sang cho CustomerId là 1





* Tạo một trigger khi xóa CustomerId 1 thì sẽ không cho xóa và báo lỗi : “Đây là khách hàng không được xóa” sau đó ROLL BACK lại hành động xóa

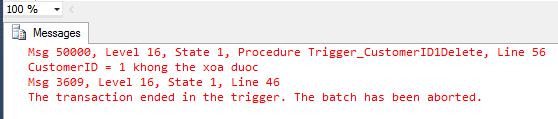


* Hiện bảng Customer có hai Trigger là “Trigger\_CustomerDelete” và Trigger “Trigger\_CustomerID1Delete”. Theo quy tắc trigger nào tạo trước sẽ thực hiện trước. Tuy nhiên ta sẽ ưu tiên Trigger “Trigger\_CustomerID1Delete” thực hiện trước bằng cách

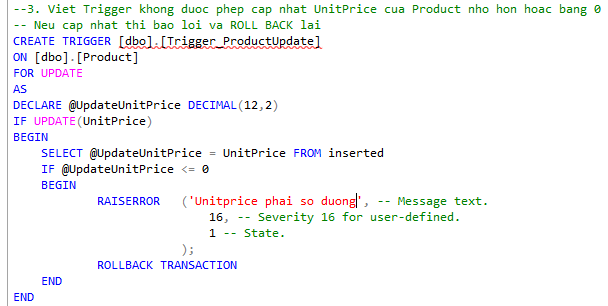


* Bây giờ ta thử xóa dữ liệu của CustomerId là 1 xem sao. Chương trình sẽ báo lỗi và không cho xóa. Do trigger mà ta tạo đảm bảo điều đó.



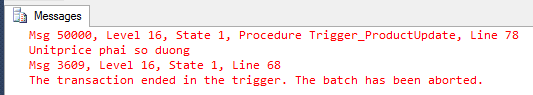


* Viết một trigger sau không cho phép cập nhật UnitPrice của Product nhỏ hơn hoặc bằng 0. Nếu cập nhật thì sẽ báo lỗi và ROLL BACK lại



* Bây giờ ta thử cập nhật UnitPrice của sản phẩm có Id là 1 về 0. Trigger sẽ báo lỗi và không cho cập nhật vậy.





**2/ Sử dụng CURSOR**

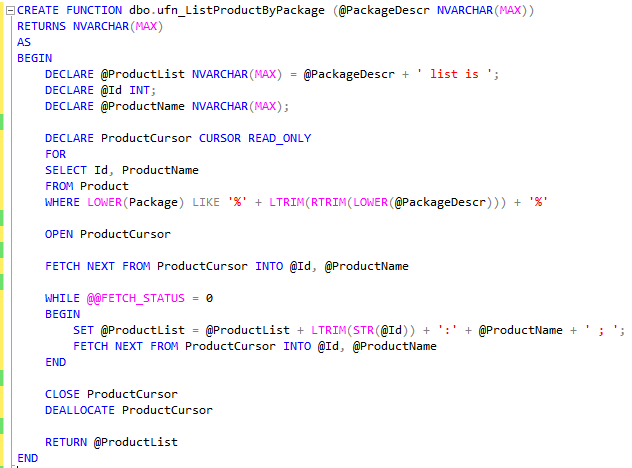
**Yêu cầu:** Viết ví dụ sử dụng CURSOR để duyệt dữ liệu

**Hướng dẫn:**

* Viết một Function với input là tiêu chuẩn dạng Package và sau đó dựa trên tiêu chuẩn này xuất ra danh sách các Id và ProductName như sau

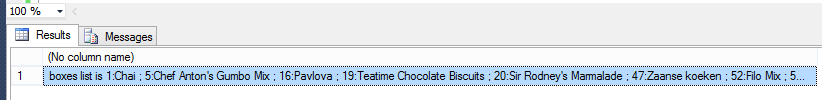
(INPUT: ‘boxes’

OUTPUT: boxes list is 1:Chai ; 5:Chef Anton's Gumbo Mix .... )



* Bây giờ ta thử gọi Function này với INPUT là boxes



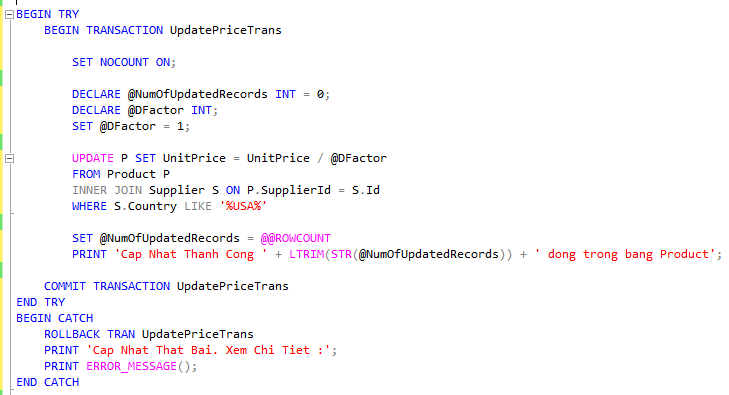


**3/ Sử dụng TRANSACTION**

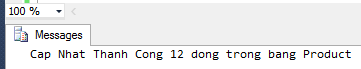
**Yêu cầu:** Viết ví dụ sử dụng TRANSACTION để kiểm soát lỗi và ROLLBACK nếu lỗi xảy ra

**Hướng dẫn:**

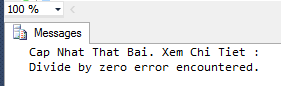
* Viết một giao dịch cập nhật UnitPrice của tất cả các sản phẩm có xuất sứ từ USA bằng cách input vào một cơ số @DFactor và tính UnitPrice mới theo công thức UnitPrice = UnitPrice / @DFactor. Sau đó cho biết có bao nhiêu sản phẩm đã được cập nhật UnitPrice. Dùng TRANSACTION trong trường hợp này để kiểm soát lỗi có thể xảy ra trong quá trình cập nhật và ROLL BACK khi cần.



* Nếu ta cho SET @DFactor = 2 thì kết quả chạy là



* Tuy nhiên nếu ta cho SET @DFactor = 0 thì lỗi chia 0 sẽ xảy ra. Với TRANSACTION ta sẽ thực hiện ROLLBACK lại khi bất cứ lỗi nào xảy ra

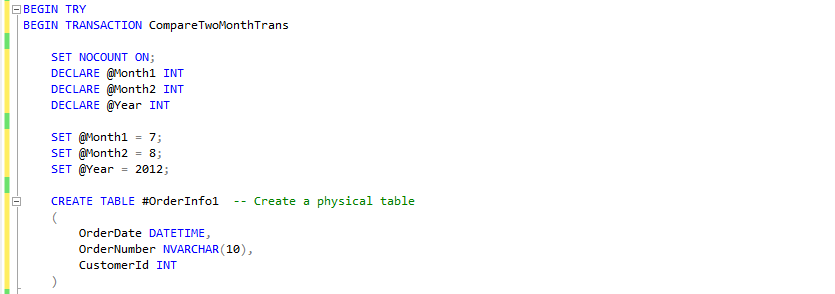


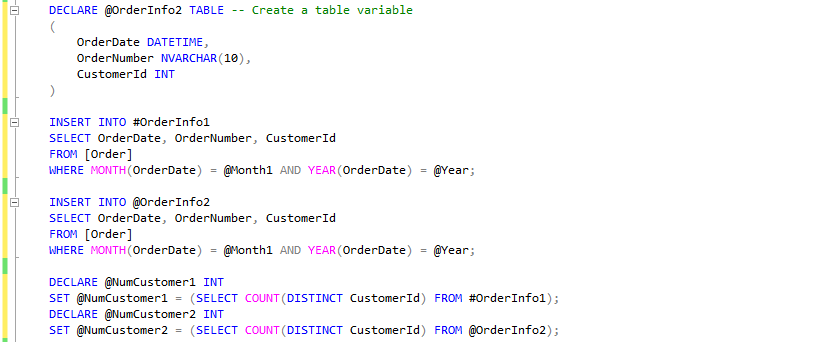
**4/ Sử dụng bảng tạm**

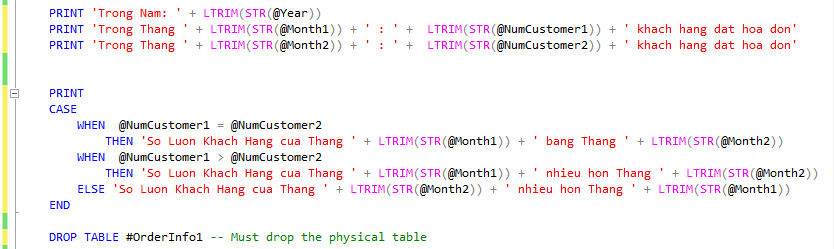
**Yêu cầu:** Viết ví dụ sử dụng hai dạng bảng tạm để lưu trữ dữ liệu

**Hướng dẫn:**

* Ta viết một ví dụ TRANSACTION với input vào là hai tháng bất kỳ của một năm sau đó cho biết tháng nào có nhiều khách hàng đặt hóa đơn hơn. Hai bảng tạm ta dùng là #OrderInfo1 và @OrderInfo2. #OrderInfo1 là bảng vật lý được tạo ra sẽ nằm trong Table của database TempDB. Do đó dùng xong ta phải DROP TABLE. Còn @OrderInfo2 là một biến dạng bảng. Biến này sẽ tự động xóa khi kết thúc thực thi script.









* Kết quả thực thi so sánh số khách hàng của tháng 7 và 8 trong năm 2012

