Lab 11: Error Handling and Transactions

Logging Errors

You are implementing a Transact-SQL script to delete orders, and you want to handle any errors that occur during the deletion process.

The following code can be used to delete order data:

```
DECLARE @OrderID int = <the_order_ID_to_delete>;
DELETE FROM SalesLT.SalesOrderDetail WHERE SalesOrderID = @OrderID;
DELETE FROM SalesLT.SalesOrderHeader WHERE SalesOrderID = @OrderID;
```

This code always succeeds, even when the specified order does not exist. Your task is to modify the existing code.

Note: to assist error checking, please just modify the existing code in the editor.

Instructions

Modify the code to check for the existence of the specified order ID before attempting to delete it. If the order does not exist, your code should throw an error. Otherwise, it should go ahead and delete the order data.

```
DECLARE @OrderID int = 0


-- Declare a custom error if the specified order doesn't exist

DECLARE @error varchar(25) = 'Order #' + cast(@OrderID as varchar) + ' does not exist';


IF NOT EXISTS (SELECT * FROM SalesLT.SalesOrderHeader WHERE SalesOrderID = @OrderID)

BEGIN

        -- Throw the custom error

        THROW 50001, @error, 0;

END

ELSE

BEGIN

        DELETE FROM SalesLT.SalesOrderDetail WHERE SalesOrderID = @OrderID;
```

```
        DELETE FROM SalesLT.SalesOrderHeader WHERE SalesOrderID = @OrderID;
END
```

Logging Errors (2)

The solution to the previous exercise is shown in the editor. However, your code now throws an error if the specified order does not exist. Refine your code to catch this (or any other) error and print the error message to the user interface using the `PRINT`command. You can use `BEGIN TRY`, `END TRY`, `BEGIN CATCH` and `END CATCH`for this.
Note: to assist error checking, please just modify the existing code in the editor.

Instructions

- Add a `TRY`...`CATCH` to the code
  - Include the `IF`-`ELSE` block in the `TRY` part.
  - In the `CATCH` part, print the error with `ERROR_MESSAGE();`

```
DECLARE @OrderID int = 0
DECLARE @error varchar(25) = 'Order #' + cast(@OrderID as varchar) + ' does not exist';

BEGIN TRY
        IF NOT EXISTS (SELECT * FROM SalesLT.SalesOrderHeader WHERE SalesOrderID = @OrderID)
        BEGIN
                THROW 50001, @error, 0
        END
        ELSE
        BEGIN
                DELETE FROM SalesLT.SalesOrderDetail WHERE SalesOrderID = @OrderID;
   DELETE FROM SalesLT.SalesOrderHeader WHERE SalesOrderID = @OrderID;
        END
END TRY
BEGIN CATCH
        -- Catch and print the error
        PRINT ERROR_MESSAGE();
```

```
END CATCH
```

Ensuring Data Consistency

You have implemented error handling logic in some Transact-SQL code that deletes order details and order headers. However, you are concerned that a failure partway through the process will result in data inconsistency in the form of undeleted order headers for which the order details have been deleted.

Your task is to enhance the code you created in the previous challenge so that the two `DELETE` statements are treated as a single transactional unit of work.
Note: to assist error checking, please just modify the existing code in the editor.

Instructions

- Add `BEGIN TRANSACTION` and `COMMIT TRANSACTION` to treat the two `DELETE` statements as a single transactional unit of work. In the error handler, modify the code so that if a transaction is in process, it is rolled back. If no transaction is in process the error handler should continue to simply print the error message.

```
DECLARE @OrderID int = 0

DECLARE @error varchar(25) = 'Order #' + cast(@OrderID as varchar) + ' does not exist';


BEGIN TRY
        IF NOT EXISTS (SELECT * FROM SalesLT.SalesOrderHeader
                                    WHERE SalesOrderID = @OrderID)
        BEGIN
                THROW 50001, @error, 0
        END
        ELSE
        BEGIN
  -- Add code to treat as single transactional unit of work
          BEGIN TRANSACTION
                DELETE FROM SalesLT.SalesOrderDetail
                WHERE SalesOrderID = @OrderID;
                DELETE FROM SalesLT.SalesOrderHeader
                WHERE SalesOrderID = @OrderID;
```

```
        COMMIT TRANSACTION
    END
END TRY
BEGIN CATCH
    IF @@TRANCOUNT > 0
    BEGIN
        -- Rollback the transaction
        ROLLBACK TRANSACTION;
    END
    ELSE
    BEGIN
        -- Report the error
        PRINT ERROR_MESSAGE();
    END
END CATCH
```