Lab Answer Key: Module 17: Implementing Error Handling

Lab: Implementing Error Handling

Exercise 1: Redirecting Errors with TRY/CATCH

Task 1: Prepare the Lab Environment

1. Ensure that the **20761C-MIA-DC** and **20761C-MIA-SQL** virtual machines are both running, and then log on to **20761C-MIA-SQL** as **ADVENTUREWORKS\Student** with the password **Pa55w.rd**.
2. In the **D:\Labfiles\Lab17\Starter** folder, right-click **Setup.cmd** and then click **Run as administrator**.
3. In the **User Account Control** dialog box, click **Yes**.
4. At the command prompt, type y, and then press **Enter**.
5. Press any key to continue.

Task 2: Write a Basic TRY/CATCH Construct

1. Start SQL Server Management Studio and connect to the **MIA-SQL** database engine using Windows authentication.
2. On the **File** menu, click **Open** and click **Project/Solution**.
3. In the **Open Project** window, open the project **D:\Labfiles\Lab17\Starter\Project\Project.ssmssln**.
4. In Solution Explorer, expand **Queries** and then double-click the **51 - Lab Exercise 1.sql**.
5. In the query window, highlight the statement **USE TSQL;** and click **Execute**.
6. Highlight the following SELECT statement under the **Task 1** description:

```
SELECT CAST(N'Some text' AS int);
```

7. Click **Execute**. Notice the conversion error.
8. Write a TRY/CATCH construct. Your T-SQL code should look like this:

```
BEGIN TRY
SELECT CAST(N'Some text' AS int);
END TRY
BEGIN CATCH
```

```
PRINT 'Error';
END CATCH;
```

9. Highlight the written T-SQL code and click **Execute**.

Task 3: Display an Error Number and an Error Message

1. Highlight the following T-SQL code under the **Task 2** description:

```
DECLARE @num varchar(20) = '0';

BEGIN TRY
PRINT 5. / CAST(@num AS numeric(10,4));
END TRY
BEGIN CATCH

END CATCH;
```

2. Click **Execute**. Notice that you did not get an error because you used the TRY/CATCH construct.
3. Modify the T-SQL code by adding two PRINT statements. The T-SQL code should look like this:

```
DECLARE @num varchar(20) = '0';

BEGIN TRY
PRINT 5. / CAST(@num AS numeric(10,4));
END TRY
BEGIN CATCH
PRINT 'Error Number: ' + CAST(ERROR_NUMBER() AS varchar(10));
PRINT 'Error Message: ' + ERROR_MESSAGE();
END CATCH;
```

4. Highlight the T-SQL code and click **Execute**.
5. Change the value of the @num variable to look like this:

```
DECLARE @num varchar(20) = 'A';
```

6. Highlight the T-SQL code and click **Execute**. Notice that you get a different error number and message.
7. Change the value of the @num variable to look like this:

```
DECLARE @num varchar(20) = ' 1000000000';
```

8. Highlight the T-SQL code and click **Execute**. Notice that you get a different error number and message.

Task 4: Add Conditional Logic to a CATCH Block

1. Modify the T-SQL code in **Task 3** to look like this:

```
DECLARE @num varchar(20) = 'A';

BEGIN TRY
PRINT 5. / CAST(@num AS numeric(10,4));
END TRY
BEGIN CATCH
IF ERROR_NUMBER() IN (245, 8114)
BEGIN
PRINT 'Handling conversion error...'
END
ELSE
BEGIN
PRINT 'Handling non-conversion error...';
END;

PRINT 'Error Number: ' + CAST(ERROR_NUMBER() AS varchar(10));
PRINT 'Error Message: ' + ERROR_MESSAGE();
END CATCH;
```

2. Highlight the written query and click **Execute**.
3. Change the value of the @num variable to look like this:


```
DECLARE @num varchar(20) = '0';
```

4. Highlight the T-SQL code and click **Execute**.

Task 5: Execute a Stored Procedure in the CATCH Block

1. Highlight the following T-SQL code under the **Task 4** description:

```
CREATE PROCEDURE dbo.GetErrorInfo AS
PRINT 'Error Number: ' + CAST(ERROR_NUMBER() AS varchar(10));
PRINT 'Error Message: ' + ERROR_MESSAGE();
PRINT 'Error Severity: ' + CAST(ERROR_SEVERITY() AS varchar(10));
PRINT 'Error State: ' + CAST(ERROR_STATE() AS varchar(10));
 PRINT 'Error Line: ' + CAST(ERROR_LINE() AS varchar(10));
 PRINT 'Error Proc: ' + COALESCE(ERROR_PROCEDURE(), 'Not within procedure');
```

2. Click **Execute**. You have created a stored procedure named dbo.GetErrorInfo.
3. Modify the T-SQL code under **TRY/CATCH** to look like this:

```
DECLARE @num varchar(20) = '0';

BEGIN TRY
PRINT 5. / CAST(@num AS numeric(10,4));
END TRY
BEGIN CATCH
EXECUTE dbo.GetErrorInfo;
END CATCH;
```

4. Highlight the written **TRY/CATCH** T-SQL code and click **Execute**.

Result: After this exercise, you should be able to capture and handle errors using a TRY/CATCH construct.

Exercise 2: Using THROW to Pass an Error Message Back to a Client

Task 1: Rethrow the Existing Error Back to a Client

1. In Solution Explorer, double-click the query **61 - Lab Exercise 2.sql**.
2. When the query window opens, highlight the statement **USE TSQL;** and click **Execute**.
3. Modify the T-SQL code under the **Task 1** description to look like this:

```
DECLARE @num varchar(20) = '0';

BEGIN TRY
PRINT 5. / CAST(@num AS numeric(10,4));
END TRY
BEGIN CATCH
EXECUTE dbo.GetErrorInfo; THROW;
END CATCH;
```

4. Highlight the written T-SQL code and click **Execute**.

Task 2: Add an Error Handling Routine

1. Modify the T-SQL code under the **Task 2** description to look like this:

```
DECLARE @num varchar(20) = 'A';

BEGIN TRY
PRINT 5. / CAST(@num AS numeric(10,4));
END TRY
BEGIN CATCH
EXECUTE dbo.GetErrorInfo;

IF ERROR_NUMBER() = 8134
BEGIN
PRINT 'Handling devision by zero...';
END
ELSE
BEGIN
PRINT 'Throwing original error';
THROW;
END;

END CATCH;
```

2. Highlight the written T-SQL code and click **Execute**.

Task 3: Add a Different Error Handling Routine

1. Find the following T-SQL code under the **Task 3** description:

```
DECLARE @msg AS varchar(2048);
SET @msg = 'You are doing the exercise for Module 17 on ' + FORMAT(CURRENT_TIMESTAMP,
'MMMM d, yyyy', 'en-US') + '. It''s not an error but it means that you are near the
final module!';
```

2. After the provided code, add a THROW statement. The completed T-SQL code should look like this:

```
DECLARE @msg AS varchar(2048);
SET @msg = 'You are doing the exercise for Module 17 on ' + FORMAT(CURRENT_TIMESTAMP,
'MMMM d, yyyy', 'en-US') + '. It''s not an error but it means that you are near the
final module!';

THROW 50001, @msg, 1;
```

3. Highlight the written T-SQL code and click **Execute**.

Task 4: Remove the Stored Procedure

- Highlight the provided T-SQL statement under the **Task 4** description and click **Execute**.

Result: After this exercise, you should know how to throw an error to pass messages back to a client.