Lab Answer Key: Module 16: Programming with T-SQL

Lab: Programming with T-SQL

Exercise 1: Declaring Variables and Delimiting Batches

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\Lab16\Starter** folder, right-click **Setup.cmd**, and then click **Run as administrator**.
3. In the **User Account Control** dialog box, click **Yes**.
4. Wait for the script to finish then press any key to continue.

Task 2: Declare a Variable and Retrieve the Value

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\Lab16\Starter\Project\Project.ssmssln**.
4. In Solution Explorer, expand **Queries**, and then double-click the query **51 - Lab Exercise 1.sql**.
5. In the query window, highlight the statement **USE TSQL;** and click **Execute**.
6. In the query pane, type the following T-SQL code after the **Task 1** description:

```
DECLARE @num int = 5;

SELECT @num AS mynumber;
```

7. Highlight the written T-SQL code and click **Execute**.
8. In the query pane, type the following T-SQL code after the previous one:

```
DECLARE
@num1 int,
@num2 int;
```

```
SET @num1 = 4;
SET @num2 = 6;

SELECT @num1 + @num2 AS totalnum;
```

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

Task 3: Set the Variable Value Using a SELECT Statement

1. In the query pane, type the following T-SQL code after the **Task 2** description:

```
DECLARE @empname nvarchar(30);

SET @empname = (SELECT firstname + N' ' + lastname FROM HR.Employees WHERE empid = 1);

SELECT @empname AS employee;
```

2. Highlight the written T-SQL code and click **Execute**.
3. Observe the result. What would happen if the SELECT statement was returning more than one row? You would get an error because the SET statement requires you to use a scalar subquery to pull data from a table. Remember that a scalar subquery fails at runtime if it returns more than one value.

Task 4: Use a Variable in the WHERE Clause

1. In the query pane, type the following T-SQL code after the **Task 3** description:

```
DECLARE
@empname nvarchar(30),
@empid int;

SET @empid = 5;

SET @empname = (SELECT firstname + N' ' + lastname FROM HR.Employees WHERE empid =
@empid);

SELECT @empname AS employee;
```

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

3. Observe and compare the results that you achieved with the desired results shown in the file D:\Labfiles\Lab16\Solution\55 - Lab Exercise 1 - Task 3 Result.txt.

4. Change the @empid variable's value from 5 to 2 and execute the modified T-SQL code to observe the changes.

Task 5: Use Variables with Batches

1. Highlight the T-SQL code in **Task 3**. On the toolbar, click **Edit** and then **Copy**.

2. In the query window, click the line after the **Task 4** description. On the toolbar, click **Edit** and then **Paste**.

3. In the code you just copied, add the batch delimiter GO before this statement:

```
SELECT @empname AS employee;
```

4. Make sure your T-SQL code looks like this:

```
DECLARE
@empname nvarchar(30),
@empid int;

SET @empid = 5;

SET @empname = (SELECT firstname + N' ' + lastname FROM HR.Employees WHERE empid =
@empid)

GO
SELECT @empname AS employee;
```

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

6. Observe the error:

   Must declare the scalar variable "@empname".

Can you explain why the batch delimiter caused an error? Variables are local to the batch in which they are defined. If you try to refer to a variable that was defined in another batch, you get an error saying that the variable was not defined. Also, keep in mind that GO is a client command, not a server T-SQL command.

Result: After this exercise, you should know how to declare and use variables in T-SQL code.

Exercise 2: Using Control-of-Flow Elements

Task 1: Write Basic Conditional Logic

1. In Solution Explorer, double-click the query **61 - Lab Exercise 2.sql**.
2. In the query window, highlight the statement **USE TSQL;** and click **Execute**.
3. In the query pane, type the following T-SQL code after the **Task 1** description:

```
DECLARE
@i int = 8,
@result nvarchar(20);

IF @i < 5
SET @result = N'Less than 5'
ELSE IF @i <= 10
SET @result = N'Between 5 and 10'
ELSE if @i > 10
SET @result = N'More than 10'
ELSE
SET @result = N'Unknown';

SELECT @result AS result;
```

4. Highlight the written T-SQL code and click **Execute**.
5. In the query pane, type the following T-SQL code:

```
DECLARE
@i int = 8,
@result nvarchar(20);

SET @result =
CASE
WHEN @i < 5 THEN
N'Less than 5'
WHEN @i <= 10 THEN
N'Between 5 and 10'
WHEN @i > 10 THEN
N'More than 10'
ELSE
N'Unknown'
END;

SELECT @result AS result;
```

This code uses a CASE expression and only one SET expression to get the same result as the previous T-SQL code. Remember to use a CASE expression when it is a matter of returning an expression. However, if you need to execute multiple statements, you cannot replace IF with CASE.

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

Task 2: Check the Employee Birthdate

1. In the query pane, type the following T-SQL code after the **Task 2** description:

```
DECLARE
@birthdate date,
@cmpdate date;

SET @birthdate = (SELECT birthdate FROM HR.Employees WHERE empid = 5);
SET @cmpdate = '19700101';

IF @birthdate < @cmpdate
PRINT 'The person selected was born before January 1, 1970'
ELSE
PRINT 'The person selected was born on or after January 1, 1970';
```

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

Task 3: Create and Execute a Stored Procedure

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

```
CREATE PROCEDURE Sales.CheckPersonBirthDate
@empid int,
@cmpdate date
AS

DECLARE
@birthdate date;

SET @birthdate = (SELECT birthdate FROM HR.Employees WHERE empid = @empid);
IF @birthdate < @cmpdate
PRINT 'The person selected was born before ' + FORMAT(@cmpdate, 'MMMM d, yyyy', 'en-
US');
ELSE
```

```
PRINT 'The person selected was born on or after ' + FORMAT(@cmpdate, 'MMMM d, yyyy',
'en-US');
```

2. Click **Execute**. You have created a stored procedure named Sales.CheckPersonBirthDate. It has two parameters: @empid, which you use to specify an employee ID, and @cmpdate, which you use as a comparison date.

3. In the query pane, type the following T-SQL code after the provided T-SQL code:

```
EXECUTE Sales.CheckPersonBirthDate @empid = 3, @cmpdate = '19900101';
```

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

Task 4: Execute a Loop Using the WHILE Statement

1. In the query pane, type the following T-SQL code after the **Task 4** description:

```
DECLARE @i int = 1;

WHILE @i <= 10
BEGIN
PRINT @i;
SET @i = @i + 1;
END;
```

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

Task 5: Remove the Stored Procedure

1. Highlight the following T-SQL code under the **Task 5** description:

```
DROP PROCEDURE Sales.CheckPersonBirthDate;
```

2. Click **Execute**.

Result: After this exercise, you should know how to control the flow of the elements inside the T-SQL code.

Exercise 3: Using Variables in a Dynamic SQL Statement

Task 1: Write a Dynamic SQL Statement That Does Not Use a Parameter

1. In Solution Explorer, double-click the query **71 - Lab Exercise 3.sql**.
2. In the query window, highlight the statement **USE TSQL;** and click **Execute**.
3. In the query pane, type the following T-SQL code after the **Task 1** description:

```
DECLARE @SQLstr nvarchar(200);

SET @SQLstr = N'SELECT empid, firstname, lastname FROM HR.Employees';

EXECUTE sys.sp_executesql @statement = @SQLstr;
```

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

Task 2: Write a Dynamic SQL Statement That Uses a Parameter

1. Highlight the T-SQL code in **Task 1**. On the toolbar, click **Edit** and then **Copy**.
2. In the query window, click the line after the **Task 2** description. On the toolbar, click **Edit** and then **Paste**.
3. Modify the T-SQL code to look like this:

```
DECLARE
@SQLstr nvarchar(200),
@SQLparam nvarchar(100);

SET @SQLstr = N'SELECT empid, firstname, lastname FROM HR.Employees WHERE empid =
@empid';
SET @SQLparam = N'@empid int';

EXECUTE sys.sp_executesql @statement = @SQLstr, @params = @SQLparam, @empid = 5;
```

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

Result: After this exercise, you should have a basic knowledge of generating and invoking dynamic SQL statements.

Exercise 4: Using Synonyms

Task 1: Create and Use a Synonym for a Table

1. In Solution Explorer, double-click the query **81 - Lab Exercise 4.sql**.
2. In the query window, highlight the statement **USE TSQL;** and click **Execute**.
3. In the query pane, type the following T-SQL code after the **Task 1** description:

```
CREATE SYNONYM dbo.Person
FOR AdventureWorks.Person.Person;
```

4. Highlight the written T-SQL code and click **Execute**. You have created a synonym named dbo.Person.
5. In the query pane, type the following SELECT statement after the previous T-SQL code:

```
SELECT FirstName, LastName
FROM dbo.Person;
```

6. Highlight the written query and click **Execute**.

Task 2: Drop the Synonym

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

```
DROP SYNONYM dbo.Person;
```

2. Click **Execute**.

Result: After this exercise, you should know how to create and use a synonym.