Lab Answer Key: Module 6: Working with SQL Server Data Types

Lab: Working with SQL Server 2016 Data Types

Exercise 1: Writing Queries That Return Date and Time Data

Task 1: Prepare the Lab Environment

1. Ensure that the **MT17B-WS2016-NAT**, **20761C-MIA-DC**, and **20761C-MIA-SQL** virtual machines are running, and then log on to **20761C-MIA-SQL** as **ADVENTUREWORKS\Student** with the password **Pa55w.rd**.
2. In the **D:\Labfiles\Lab06\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, and when prompted, press any key.

Task 2: Write a SELECT Statement to Retrieve Information About the Current Date

1. Start SQL Server Management Studio and connect to the **MIA-SQL** database engine using Windows authentication.
2. On the **File** menu, point to **Open**, and then click **Project/Solution**.
3. In the **Open Project** dialog box, navigate to the **D:\Labfiles\Lab06\Starter\Project** folder, and then double-click **Project.ssmssln**.
4. In Solution Explorer, expand **Queries**, and then double-click **51 - Lab Exercise 1.sql**.
5. In the query pane, highlight the statement **USE TSQL;**, and then click **Execute**.
6. In the query pane, after the **Task 1** description, type the following query:

```
SELECT
CURRENT_TIMESTAMP AS currentdatetime,
CAST(CURRENT_TIMESTAMP AS DATE) AS currentdate,
CAST(CURRENT_TIMESTAMP AS TIME) AS currenttime,
YEAR(CURRENT_TIMESTAMP) AS currentyear,
MONTH(CURRENT_TIMESTAMP) AS currentmonth,
DAY(CURRENT_TIMESTAMP) AS currentday,
DATEPART(week, CURRENT_TIMESTAMP) AS currentweeknumber,
DATENAME(month, CURRENT_TIMESTAMP) AS currentmonthname;
```

This query uses the CURRENT_TIMESTAMP function to return the current date and time. You can also use the SYSDATETIME function to get a more precise time element, compared to the CURRENT_TIMESTAMP function.

Note that you cannot use the alias currentdatetime as the source in the second column calculation because SQL Server supports a concept called all-at-once operations. This means that all expressions appearing in the same logical query processing phase are evaluated as if they occurred at the same point in time. This concept explains why, for example, you cannot refer to column aliases assigned in the SELECT clause within the same SELECT clause, even if it seems intuitive that you should be able to.

7. Highlight the written query, and click **Execute**.

Task 3: Write a SELECT Statement to Return the Date Data Type

1. In the query pane, after the **Task 2** description, type the following queries:

```
SELECT DATEFROMPARTS(2015, 12, 11) AS somedate;
SELECT CAST('20151211' AS DATE) AS somedate;
SELECT CONVERT(DATE, '12/11/2015', 101) AS somedate;
```

2. Highlight the written queries, and click **Execute**.

Task 4: Write a SELECT Statement That Uses Different Date and Time Functions

1. In the query pane, after the **Task 3** description, type the following query:

```
SELECT
DATEADD(month, 3, CURRENT_TIMESTAMP) AS threemonths,
DATEDIFF(day, CURRENT_TIMESTAMP, DATEADD(month, 3, CURRENT_TIMESTAMP)) AS diffdays,
DATEDIFF(week, '19920404', '20110916') AS diffweeks,
DATEADD(day, 1, EOMONTH(CURRENT_TIMESTAMP,-1)) AS firstday;
```

2. Highlight the written query, and click **Execute**.

Task 5: Write a SELECT Statement to Show Whether a Table of Strings Can Be Used as Dates

1. Under the **Task 4** description, highlight the written query, and click **Execute**.
2. In the query pane, type the following queries after the **Task 4** description:

```
SELECT
isitdate,
CASE WHEN ISDATE(isitdate) = 1 THEN CONVERT(DATE, isitdate) ELSE NULL END AS
converteddate
FROM Sales.Somedates;

--Uses the TRY_CONVERT function:
SELECT
isitdate,
TRY_CONVERT(DATE, isitdate) AS converteddate
FROM Sales.Somedates;
```

The second query uses the TRY_CONVERT function. This function returns a value cast to the specified data type if the casting succeeds; otherwise, it returns NULL. Don't worry if you do not recognize the type conversion functions, as they will be covered in the next module.

3. Highlight the written queries, and click **Execute**.

4. Observe the result and answer these questions:

   o What is the difference between the SYSDATETIME and CURRENT_TIMESTAMP functions?

     There are two main differences. First, the SYSDATETIME function provides a more precise time element compared to the CURRENT_TIMESTAMP function. Second, the SYSDATETIME function returns the data type **datetime2**(7), whereas the CURRENT_TIMESTAMP returns the data type **datetime**.

   o What is a language-neutral format for the data type date?

     You can use the formats 'YYYYMMDD' or 'YYYY-MM-DD'.

Result: After this exercise, you should be able to retrieve date and time data using T-SQL.

Exercise 2: Writing Queries That Use Date and Time Functions

Task 1: Write a SELECT Statement to Retrieve Customers with Orders in a Given Month

1. In Solution Explorer, double-click **61 - Lab Exercise 2.sql**.

2. In the query pane, highlight the statement **USE TSQL;**, and then click **Execute**.

3. In the query pane, after the **Task 1** description, type the following query:

```
SELECT DISTINCT
custid
FROM Sales.Orders
```

```
WHERE
YEAR(orderdate) = 2008
AND MONTH(orderdate) = 2;
```

4. Highlight the written query, and click **Execute**.

   Note that, as a performance enhancement, you could also write a query that uses a range format that would utilize an index on Sales.Orders.orderdate. The query would then look like this:

```
SELECT DISTINCT
custid
FROM Sales.Orders
WHERE
orderdate >= '20080201'
AND orderdate < '20080301';
```

Task 2: Write a SELECT Statement to Calculate the First and Last Day of the Month

1. In the query pane, after the **Task 2** description, type the following query:

```
SELECT
CURRENT_TIMESTAMP AS currentdate,
DATEADD (day, 1, EOMONTH(CURRENT_TIMESTAMP, -1)) AS firstofmonth,
EOMONTH(CURRENT_TIMESTAMP) AS endofmonth;
```

2. Highlight the written query, and click **Execute**.

   This query uses the EOMONTH function, which was added in SQL Server 2012.

Task 3: Write a SELECT Statement to Retrieve the Orders Placed in the Last Five Days of the Ordered Month

1. In the query pane, after the **Task 3** description, type the following query:

```
SELECT
orderid, custid, orderdate
FROM Sales.Orders
WHERE
DATEDIFF(
day,
orderdate,
EOMONTH(orderdate)
) < 5;
```

2. Highlight the written query, and click **Execute**.

Task 4: Write a SELECT Statement to Retrieve All Distinct Products Sold in the First 10 Weeks of the Year 2007

1. In the query pane, after the **Task 4** description, type the following query:

```
SELECT DISTINCT
d.productid
FROM Sales.Orders AS o
INNER JOIN Sales.OrderDetails AS d ON d.orderid = o.orderid
WHERE
DATEPART(week, orderdate) <= 10
AND YEAR(orderdate) = 2007;
```

2. Highlight the written query, and click **Execute**.

Result: After this exercise, you should know how to use the date and time functions.

Exercise 3: Writing Queries That Return Character Data

Task 1: Write a SELECT Statement to Concatenate Two Columns

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

```
SELECT
CONCAT(contactname, N' (city: ', city, N')') AS contactwithcity
FROM Sales.Customers;
```

4. Highlight the written query, and click **Execute**.

   An alternate way to write this query would be to use the + (plus) operator:

```
SELECT
contactname + N' (city: ' + city + N')' AS contactwithcity
FROM Sales.Customers;
```

Task 2: Add an Additional Column to the Concatenated String Which Might Contain NULL

1. In the query pane, after the **Task 2** description, type the following query:

```
SELECT
CONCAT(contactname, N' (city: ', city,  N', region: ', region, N')') AS fullcontact
FROM Sales.Customers;
```

2. Highlight the written query, and click **Execute**.

   An alternative way to write this query would be to use the + (plus) operator, which requires the COALESCE function to replace a NULL with an empty string. Later modules will include more examples of how to handle NULL.

```
SELECT
contactname + N' (city: ' + city + N', region: ' + COALESCE(region, '') + N')' AS
fullcontact
FROM Sales.Customers;
```

Task 3: Write a SELECT Statement to Retrieve Customer Contacts Based on the First Character in the Contact Name

1. In the query pane, after the **Task 3** description, type the following query:

```
SELECT contactname, contacttitle
FROM Sales.Customers
WHERE contactname LIKE N'[A-G]%'
ORDER BY contactname;
```

2. Highlight the written query, and click **Execute**.

Result: After this exercise, you should have an understanding of how to concatenate character data.

Exercise 4: Writing Queries That Use Character Functions

Task 1: Write a SELECT Statement That Uses the SUBSTRING Function

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

```
SELECT
contactname,
SUBSTRING(contactname, 0, CHARINDEX(N',', contactname)) AS lastname
FROM Sales.Customers;
```

4. Highlight the written query, and click **Execute**.

Task 2: Write a Query to Retrieve the Contact's First Name Using SUBSTRING

1. In the query pane, after the **Task 2** description, type the following query:

```
SELECT
REPLACE(contactname, ',', '') AS newcontactname,
SUBSTRING(contactname, CHARINDEX(N',', contactname)+1, LEN(contactname)-CHARINDEX(N',',
contactname)+1) AS firstname
FROM Sales.Customers;
```

2. Highlight the written query, and click **Execute**.

Task 3: Write a SELECT Statement to Format the Customer ID

1. In the query pane, after the **Task 3** description, type the following query:

```
SELECT
custid,
N'C' + RIGHT(REPLICATE('0', 5) + CAST(custid AS VARCHAR(5)), 5) AS custnewid
FROM Sales.Customers;
```

2.

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

An alternative way to write this query would be to use the FORMAT function. The query would then look like this:

```
SELECT custid,
FORMAT(custid, N'\C00000') AS custnewid
FROM Sales.Customers;
```

Task 4: Challenge: Write a SELECT Statement to Return the Number of Character Occurrences

1. In the query pane, after the **Task 4** description, type the following query:

```
SELECT
contactname,
LEN(contactname) - LEN(REPLACE(contactname, 'a', '')) AS numberofa
FROM Sales.Customers
ORDER BY numberofa DESC;
```

This elegant solution first returns the number of characters in the contact name, and then subtracts the number of characters in the contact name without the character ‘a’. The result is stored in a new column named numberofa.

2. Highlight the written query, and click **Execute**.

3. Close SQL Server Management Studio without saving any files.

Result: After this exercise, you should have an understanding of how to use the character functions.