Lecture 10: MS SQL Server

Other DDL/DML statements, functions, and examples

Examples and material from the book *Murach’s SQL Server 2005 for developers* written by Bryan Syverson and Joel Murach, Mike Murach & Associates, Inc.
**SELECT statements**

A SELECT statement that retrieves all invoices between given dates

```sql
SELECT InvoiceNumber, InvoiceDate, InvoiceTotal
FROM Invoices
WHERE InvoiceDate BETWEEN '2006-05-01' AND '2006-05-31'
ORDER BY InvoiceDate
```

A SELECT statement with renaming columns

```sql
SELECT InvoiceNumber AS [Invoice Number],
       InvoiceDate AS Date, InvoiceTotal AS Total
FROM Invoices
```

How to format strings and include apostrophes in literal values

```sql
SELECT VendorName + '''s Address: ' AS Name,
       VendorCity + ', ' + VendorState + ' ' + VendorZipCode
AS Address
FROM Vendors
```
A SELECT statement that uses the LEFT function

```sql
SELECT VendorContactFName, VendorContactLName,
    LEFT(VendorContactFName, 1) +
    LEFT(VendorContactLName, 1) AS Initials
FROM Vendors
```

**Other string functions:**

- `LEN(str)` returns the number of characters in str
- `LTRIM(str)` remove leading spaces in string
- `RTRIM(str)` remove trailing spaces in string
- `RIGHT(str, l)` returns l characters from start of string
- `SUBSTRING(str, st, l)` returns l characters from str at st position
- `REVERSE(str)` returns str in reverse order
- `LOWER(str)` convert str to lowercase letters
- `UPPER(str)` convert str to uppercase letters
SELECT with TOP and Percent

A SELECT statement with the TOP clause

SELECT TOP 5 VendorID, InvoiceTotal
FROM Invoices
ORDER BY InvoiceTotal DESC

The same statement with the PERCENT keyword

SELECT TOP 5 PERCENT VendorID, InvoiceTotal
FROM Invoices
ORDER BY InvoiceTotal DESC
Qualified table names

A join with fully-qualified table names
SELECT VendorName, CustLastName, CustFirstName, VendorState AS State, VendorCity AS City
FROM DBServer.AP.dbo.Vendors AS Vendors
JOIN DBServer.ProductOrders.dbo.Customers AS Customers
ON Vendors.VendorZipCode = Customers.CustZip
ORDER BY State, City

The same join with partially-qualified table names
SELECT VendorName, CustLastName, CustFirstName, VendorState AS State, VendorCity AS City
FROM Vendors
JOIN ProductOrders..Customers AS Customers
ON Vendors.VendorZipCode = Customers.CustZip
ORDER BY State, City
Queries with aggregate functions

A summary query that uses the COUNT(*), AVG, and SUM functions

```sql
SELECT 'After 1/1/2006' AS SelectionDate,
       COUNT(*) AS NumberOfInvoices,
       AVG(InvoiceTotal) AS AverageInvoiceAmount,
       SUM(InvoiceTotal) AS TotalInvoiceAmount
FROM Invoices
WHERE InvoiceDate > '2006-01-01'
```

A summary query that works on non-numeric columns

```sql
SELECT MIN(VendorName) AS FirstVendor,
       MAX(VendorName) AS LastVendor,
       COUNT(VendorName) AS NumberOfVendors
FROM Vendors
```
Use of **ALL**, **ANY** keywords

**A query that returns invoices that are larger than the largest invoice for vendor 34**

```sql
SELECT VendorName, InvoiceNumber, InvoiceTotal
FROM Invoices JOIN Vendors ON Invoices.VendorID = Vendors.VendorID
WHERE InvoiceTotal > ALL
  (SELECT InvoiceTotal FROM Invoices WHERE VendorID = 34)
ORDER BY VendorName
```

**A query that returns invoices smaller than the largest invoice for vendor 115**

```sql
SELECT VendorName, InvoiceNumber, InvoiceTotal
FROM Vendors JOIN Invoices ON Vendors.VendorID = Invoices.InvoiceID
WHERE InvoiceTotal < ANY
  (SELECT InvoiceTotal FROM Invoices WHERE VendorID = 115)
```
Use of CONVERT function

A SELECT statement that uses the CONVERT function

```sql
SELECT CONVERT(varchar, InvoiceDate) AS varcharDate,
       CONVERT(varchar, InvoiceDate, 1) AS varcharDate_1,
       CONVERT(varchar, InvoiceDate, 107) AS varcharDate_107,
       CONVERT(varchar, InvoiceTotal) AS varcharTotal,
       CONVERT(varchar, InvoiceTotal, 1) AS varcharTotal_1
FROM Invoices
```

<table>
<thead>
<tr>
<th></th>
<th>varcharDate</th>
<th>varcharDate_1</th>
<th>varcharDate_107</th>
<th>varcharTotal</th>
<th>varcharTotal_1</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Feb 25 2006 12:00AM</td>
<td>02/25/06</td>
<td>Feb 25, 2006</td>
<td>116.54</td>
<td>116.54</td>
</tr>
<tr>
<td>2</td>
<td>Mar 14 2006 12:00AM</td>
<td>03/14/06</td>
<td>Mar 14, 2006</td>
<td>1083.58</td>
<td>1,083.58</td>
</tr>
<tr>
<td>3</td>
<td>Apr 11 2006 12:00AM</td>
<td>04/11/06</td>
<td>Apr 11, 2006</td>
<td>20551.18</td>
<td>20,551.18</td>
</tr>
<tr>
<td>4</td>
<td>Apr 16 2006 12:00AM</td>
<td>04/16/06</td>
<td>Apr 16, 2006</td>
<td>26881.40</td>
<td>26,881.40</td>
</tr>
</tbody>
</table>
Format real numbers

SELECT ID, R, CAST(R AS decimal(9,3)) AS R_decimal,
     CAST(CAST(R AS decimal(9,3)) AS varchar(9)) AS R_varchar,
     LEN(CAST(CAST(R AS decimal(9,3)) AS varchar(9))) AS R_LEN,
     SPACE(9 - LEN(CAST(CAST(R AS decimal(9,3)) AS varchar(9)))) +
     CAST(CAST(R AS decimal(9,3)) AS varchar(9)) AS R_Formatted
FROM RealSample

CAST(e, d) function converts an expression to a data type. SPACE(n) function returns n spaces.
**RealSample Table and output**

<table>
<thead>
<tr>
<th>ID</th>
<th>R</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1.0000000000011</td>
</tr>
<tr>
<td>2</td>
<td>1</td>
</tr>
<tr>
<td>3</td>
<td>0.99999999999999</td>
</tr>
<tr>
<td>4</td>
<td>1234.56789012345</td>
</tr>
<tr>
<td>5</td>
<td>999.04440209348</td>
</tr>
</tbody>
</table>

Output of last query (only last 4 columns):

<table>
<thead>
<tr>
<th>R_decimal</th>
<th>R_varchar</th>
<th>R_LEN</th>
<th>R_Formatted</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.000</td>
<td>1.000</td>
<td>5</td>
<td>1.000</td>
</tr>
<tr>
<td>1.000</td>
<td>1.000</td>
<td>5</td>
<td>1.000</td>
</tr>
<tr>
<td>1.000</td>
<td>1.000</td>
<td>5</td>
<td>1.000</td>
</tr>
<tr>
<td>1234.568</td>
<td>1234.568</td>
<td>8</td>
<td>1234.568</td>
</tr>
<tr>
<td>999.044</td>
<td>999.044</td>
<td>7</td>
<td>999.044</td>
</tr>
</tbody>
</table>

last column is right-formatted
# Date/time functions

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>GETDATE()</td>
<td>Return system’s date and time.</td>
</tr>
<tr>
<td>DAY(date)</td>
<td>Returns the day of the month as an integer.</td>
</tr>
<tr>
<td>MONTH(date)</td>
<td>Returns the month as an integer.</td>
</tr>
<tr>
<td>YEAR(date)</td>
<td>Returns the 4-digit yeas as an integer.</td>
</tr>
<tr>
<td>DATENAME(datepart, date)</td>
<td>Returns part of the date specified by datepart as string.</td>
</tr>
<tr>
<td>DATEADD(datepart, #, date)</td>
<td>Returns part of the date specified by datepart as int.</td>
</tr>
<tr>
<td>DATEDIFF(datepart, startdate, enddate)</td>
<td>Returns the number of datepart units between the specified start and end dates.</td>
</tr>
<tr>
<td>ISDATE(expression)</td>
<td>Returns 1 if the expression is a valid date/time; else 0.</td>
</tr>
<tr>
<td>DATEPART(datepart, date)</td>
<td>Returns part of a date as an integer.</td>
</tr>
</tbody>
</table>
## Example of date/time functions

<table>
<thead>
<tr>
<th>Function</th>
<th>Result</th>
</tr>
</thead>
<tbody>
<tr>
<td>MONTH(‘2007-10-22’)</td>
<td>10</td>
</tr>
<tr>
<td>DAY(‘2007-10-22’)</td>
<td>22</td>
</tr>
<tr>
<td>DATEPART(m, ‘2007-10-22’)</td>
<td>10</td>
</tr>
<tr>
<td>DATENAME(month, ‘2007-10-22’)</td>
<td>October</td>
</tr>
<tr>
<td>ISDATE(‘2007-10-22’)</td>
<td>1 (if today’s date is Oct. 22, 2007)</td>
</tr>
<tr>
<td>DATEPART(week, ‘2007-10-22’)</td>
<td>43</td>
</tr>
<tr>
<td>DATENAME(quarter, ‘2007-10-22’)</td>
<td>4</td>
</tr>
<tr>
<td>DATENAME(weekday, ‘2007-10-22’)</td>
<td>Monday</td>
</tr>
<tr>
<td>DATEADD(day, 1, ‘2007-10-20’)</td>
<td>‘2007-10-21 00:00:00.000’</td>
</tr>
<tr>
<td>DATEADD(year, 1.5, ‘2006-02-22’)</td>
<td>‘2007-02-22 00:00:00.000’</td>
</tr>
<tr>
<td>DATEDIFF(month, ‘2005-12-06’, ‘2006-09-30’)</td>
<td>9</td>
</tr>
</tbody>
</table>
ASCII code for control characters

A SELECT statement that uses the CHAR function to format printer output

```
SELECT VendorName + CHAR(13) + CHAR(10)
    + VendorAddress1 + CHAR(13) + CHAR(10)
    + VendorCity + ', ' + VendorState + ' ' + VendorZipCode
FROM VENDORS
WHERE VendorID = 1
```

Output

```
US Postal Service
Attn: Supt. Window Services
Madison, WI 53707
```

Where

- CHAR(13) Carriage return
- CHAR(10) Line Feed
- CHAR(9) Tab
**ALTER statement**

```
ALTER TABLE table_name [WITH CHECK|WITH NOCHECK]
{ADD new_column_name data_type [column_attributes] |
 DROP COLUMN column_name |
 ALTER COLUMN column_name new_data_type [NULL|NOT NULL] |
 ADD [CONSTRAINT] new_constraint_definition |
 DROP [CONSTRAINT] constraint_name}
```

- **Example: Drop a column**
  ```
  ALTER TABLE Vendors
  DROP COLUMN LastTranDate
  ```

- **Example: Add a check constraint**
  ```
  ALTER TABLE Invoices WITH NOCHECK
  ADD CHECK (InvoiceTotal >= 1)
  ```

- **Example: Change the datatype of a column**
  ```
  ALTER TABLE Inv ALTER COLUMN ItemDesc VARCHAR(200)
  ```
CREATE VIEW

A CREATE VIEW statement for a view named VendX

CREATE VIEW VendX AS
    SELECT VendorName, VendorState, VendorPhone
    FROM Vendors

A SELECT statement that uses the VendX view

SELECT * FROM VendX
    WHERE VendorState = 'CA'
    ORDER BY VendorName

- A view can be based on another view.
- SELECT statement can include joins, sub-queries, unions.
More commands on VIEWs

A CREATE VIEW statement that summarizes invoices by vendor

```sql
CREATE VIEW InvoiceSummary
AS
SELECT VendorName, COUNT(*) AS InvoiceQty,
      SUM(InvoiceTotal) AS InvoiceSum
FROM Vendors JOIN Invoices ON Vendors.VendorID =
                      Invoices.VendorID
GROUP BY VendorName
```

You can ALTER and/or DROP VIEWs

```sql
DROP VIEW InvoiceSummary
ALTER VIEW InvoiceSummary
WITH ENCRYPTION ← Hide VIEW definition from others
AS SELECT ...
```
CREATE PROCEDURES

A procedure is program code stored in a DB

CREATE {PROC|PROCEDURE} procedure_name
[parameter_declarations]
[WITH [RECOMPILE] [, ENCRYPTION] [, EXECUTE_AS_clause]]
AS sql_statements

– Example:
CREATE PROC spInvRept
AS
SELECT InvoiceNumber, InvoiceDate, InvoiceTotal
FROM Invoices JOIN Vendors
    ON Invoices.VendorID = Vendors.VendorID
WHERE InvoiceTotal - CreditTotal - PaymentTotal > 0
ORDER BY VendorName

– Execute it with command: EXEC spInvRept
PROCEDURE with parameters

Procedure with input and output parameters

```
CREATE PROC spInvTotal1
    @DateVar smalldatetime = '2009-01-01', --optional
    @InvTotal money OUTPUT
AS
    SELECT @InvTotal = SUM(InvoiceTotal)
    FROM Invoices
    WHERE InvoiceDate >= @DateVar
```

Pass parameters to the procedure by position

```
DECLARE @MyInvTotal money
EXEC spInvTotal1 '2006-06-01', @MyInvTotal OUTPUT
```

Parameters can be passed by name

```
DECLARE @MyInvTotal money
EXEC spInvTotal1 @DateVar='2006-06-01',
    @InvTotal=@MyInvTotal OUTPUT
```
User-Defined Functions

・ User can create functions similar to procedures

```sql
CREATE FUNCTION fnVendorID
    (@VendorName varchar(50))
RETURNS int
BEGIN
    RETURN (SELECT VendorID FROM Vendors
            WHERE VendorName = @VendorName)
END
```

・ Invoking this function

```sql
SELECT InvoiceDate, InvoiceTotal
FROM Invoices
WHERE VendorID = dbo.fnVendorID('IBM')
```