---------------------------------------------------------------------

-- T-SQL Fundamentals Fourth Edition

-- Chapter 05 - Table Expressions

-- © Itzik Ben-Gan

---------------------------------------------------------------------

---------------------------------------------------------------------

-- Derived Tables

---------------------------------------------------------------------

USE TSQLV6;

SELECT \*

FROM (SELECT custid, companyname

FROM Sales.Customers

WHERE country = N'USA') AS USACusts;

---------------------------------------------------------------------

-- Assigning Column Aliases

---------------------------------------------------------------------

-- Following fails

/\*

SELECT

YEAR(orderdate) AS orderyear,

COUNT(DISTINCT custid) AS numcusts

FROM Sales.Orders

GROUP BY orderyear;

\*/

-- Listing 5-1 Query with a Derived Table using Inline Aliasing Form

SELECT orderyear, COUNT(DISTINCT custid) AS numcusts

FROM (SELECT YEAR(orderdate) AS orderyear, custid

FROM Sales.Orders) AS D

GROUP BY orderyear;

SELECT YEAR(orderdate) AS orderyear, COUNT(DISTINCT custid) AS numcusts

FROM Sales.Orders

GROUP BY YEAR(orderdate);

-- External column aliasing

SELECT orderyear, COUNT(DISTINCT custid) AS numcusts

FROM (SELECT YEAR(orderdate), custid

FROM Sales.Orders) AS D(orderyear, custid)

GROUP BY orderyear;

GO

---------------------------------------------------------------------

-- Using Arguments

---------------------------------------------------------------------

-- Yearly Count of Customers handled by Employee 3

DECLARE @empid AS INT = 3;

SELECT orderyear, COUNT(DISTINCT custid) AS numcusts

FROM (SELECT YEAR(orderdate) AS orderyear, custid

FROM Sales.Orders

WHERE empid = @empid) AS D

GROUP BY orderyear;

GO

---------------------------------------------------------------------

-- Nesting

---------------------------------------------------------------------

-- Listing 5-2 Query with Nested Derived Tables

SELECT orderyear, numcusts

FROM (SELECT orderyear, COUNT(DISTINCT custid) AS numcusts

FROM (SELECT YEAR(orderdate) AS orderyear, custid

FROM Sales.Orders) AS D1

GROUP BY orderyear) AS D2

WHERE numcusts > 70;

SELECT YEAR(orderdate) AS orderyear, COUNT(DISTINCT custid) AS numcusts

FROM Sales.Orders

GROUP BY YEAR(orderdate)

HAVING COUNT(DISTINCT custid) > 70;

---------------------------------------------------------------------

-- Multiple References

---------------------------------------------------------------------

-- Listing 5-3 Multiple Derived Tables Based on the Same Query

SELECT Cur.orderyear,

Cur.numcusts AS curnumcusts, Prv.numcusts AS prvnumcusts,

Cur.numcusts - Prv.numcusts AS growth

FROM (SELECT YEAR(orderdate) AS orderyear,

COUNT(DISTINCT custid) AS numcusts

FROM Sales.Orders

GROUP BY YEAR(orderdate)) AS Cur

LEFT OUTER JOIN

(SELECT YEAR(orderdate) AS orderyear,

COUNT(DISTINCT custid) AS numcusts

FROM Sales.Orders

GROUP BY YEAR(orderdate)) AS Prv

ON Cur.orderyear = Prv.orderyear + 1;

---------------------------------------------------------------------

-- Common Table Expressions

---------------------------------------------------------------------

WITH USACusts AS

(

SELECT custid, companyname

FROM Sales.Customers

WHERE country = N'USA'

)

SELECT \* FROM USACusts;

---------------------------------------------------------------------

-- Assigning Column Aliases

---------------------------------------------------------------------

-- Inline column aliasing

WITH C AS

(

SELECT YEAR(orderdate) AS orderyear, custid

FROM Sales.Orders

)

SELECT orderyear, COUNT(DISTINCT custid) AS numcusts

FROM C

GROUP BY orderyear;

-- External column aliasing

WITH C(orderyear, custid) AS

(

SELECT YEAR(orderdate), custid

FROM Sales.Orders

)

SELECT orderyear, COUNT(DISTINCT custid) AS numcusts

FROM C

GROUP BY orderyear;

GO

---------------------------------------------------------------------

-- Using Arguments

---------------------------------------------------------------------

DECLARE @empid AS INT = 3;

WITH C AS

(

SELECT YEAR(orderdate) AS orderyear, custid

FROM Sales.Orders

WHERE empid = @empid

)

SELECT orderyear, COUNT(DISTINCT custid) AS numcusts

FROM C

GROUP BY orderyear;

GO

---------------------------------------------------------------------

-- Defining Multiple CTEs

---------------------------------------------------------------------

WITH C1 AS

(

SELECT YEAR(orderdate) AS orderyear, custid

FROM Sales.Orders

),

C2 AS

(

SELECT orderyear, COUNT(DISTINCT custid) AS numcusts

FROM C1

GROUP BY orderyear

)

SELECT orderyear, numcusts

FROM C2

WHERE numcusts > 70;

---------------------------------------------------------------------

-- Multiple References

---------------------------------------------------------------------

WITH YearlyCount AS

(

SELECT YEAR(orderdate) AS orderyear,

COUNT(DISTINCT custid) AS numcusts

FROM Sales.Orders

GROUP BY YEAR(orderdate)

)

SELECT Cur.orderyear,

Cur.numcusts AS curnumcusts, Prv.numcusts AS prvnumcusts,

Cur.numcusts - Prv.numcusts AS growth

FROM YearlyCount AS Cur

LEFT OUTER JOIN YearlyCount AS Prv

ON Cur.orderyear = Prv.orderyear + 1;

---------------------------------------------------------------------

-- Recursive CTEs

---------------------------------------------------------------------

WITH EmpsCTE AS

(

SELECT empid, mgrid, firstname, lastname

FROM HR.Employees

WHERE empid = 2

UNION ALL

SELECT C.empid, C.mgrid, C.firstname, C.lastname

FROM EmpsCTE AS P

INNER JOIN HR.Employees AS C

ON C.mgrid = P.empid

)

SELECT empid, mgrid, firstname, lastname

FROM EmpsCTE;

GO

---------------------------------------------------------------------

-- Views

---------------------------------------------------------------------

---------------------------------------------------------------------

-- Views Described

---------------------------------------------------------------------

-- Creating USACusts View

CREATE OR ALTER VIEW Sales.USACusts

AS

SELECT

custid, companyname, contactname, contacttitle, address,

city, region, postalcode, country, phone, fax

FROM Sales.Customers

WHERE country = N'USA';

GO

SELECT custid, companyname

FROM Sales.USACusts;

GO

---------------------------------------------------------------------

-- Views and ORDER BY

---------------------------------------------------------------------

-- ORDER BY in a View is not Allowed

/\*

CREATE OR ALTER VIEW Sales.USACusts

AS

SELECT

custid, companyname, contactname, contacttitle, address,

city, region, postalcode, country, phone, fax

FROM Sales.Customers

WHERE country = N'USA'

ORDER BY region;

GO

\*/

-- Instead, use ORDER BY in Outer Query

SELECT custid, companyname, region

FROM Sales.USACusts

ORDER BY region;

GO

-- Do not Rely on TOP

CREATE OR ALTER VIEW Sales.USACusts

AS

SELECT TOP (100) PERCENT

custid, companyname, contactname, contacttitle, address,

city, region, postalcode, country, phone, fax

FROM Sales.Customers

WHERE country = N'USA'

ORDER BY region;

GO

-- Query USACusts

SELECT custid, companyname, region

FROM Sales.USACusts;

GO

-- DO NOT rely on OFFSET-FETCH, even if for now the engine does return rows in order

CREATE OR ALTER VIEW Sales.USACusts

AS

SELECT

custid, companyname, contactname, contacttitle, address,

city, region, postalcode, country, phone, fax

FROM Sales.Customers

WHERE country = N'USA'

ORDER BY region

OFFSET 0 ROWS;

GO

-- Query USACusts

SELECT custid, companyname, region

FROM Sales.USACusts;

GO

---------------------------------------------------------------------

-- View Options

---------------------------------------------------------------------

---------------------------------------------------------------------

-- ENCRYPTION

---------------------------------------------------------------------

CREATE OR ALTER VIEW Sales.USACusts

AS

SELECT

custid, companyname, contactname, contacttitle, address,

city, region, postalcode, country, phone, fax

FROM Sales.Customers

WHERE country = N'USA';

GO

SELECT OBJECT\_DEFINITION(OBJECT\_ID('Sales.USACusts'));

GO

CREATE OR ALTER VIEW Sales.USACusts WITH ENCRYPTION

AS

SELECT

custid, companyname, contactname, contacttitle, address,

city, region, postalcode, country, phone, fax

FROM Sales.Customers

WHERE country = N'USA';

GO

SELECT OBJECT\_DEFINITION(OBJECT\_ID('Sales.USACusts'));

EXEC sp\_helptext 'Sales.USACusts';

GO

---------------------------------------------------------------------

-- SCHEMABINDING

---------------------------------------------------------------------

CREATE OR ALTER VIEW Sales.USACusts WITH SCHEMABINDING

AS

SELECT

custid, companyname, contactname, contacttitle, address,

city, region, postalcode, country, phone, fax

FROM Sales.Customers

WHERE country = N'USA';

GO

-- Try a schema change

/\*

ALTER TABLE Sales.Customers DROP COLUMN address;

\*/

GO

---------------------------------------------------------------------

-- CHECK OPTION

---------------------------------------------------------------------

-- Notice that you can insert a row through the view

INSERT INTO Sales.USACusts(

companyname, contactname, contacttitle, address,

city, region, postalcode, country, phone, fax)

VALUES(

N'Customer ABCDE', N'Contact ABCDE', N'Title ABCDE', N'Address ABCDE',

N'London', NULL, N'12345', N'UK', N'012-3456789', N'012-3456789');

-- But when you query the view, you won't see it

SELECT custid, companyname, country

FROM Sales.USACusts

WHERE companyname = N'Customer ABCDE';

-- You can see it in the table, though

SELECT custid, companyname, country

FROM Sales.Customers

WHERE companyname = N'Customer ABCDE';

GO

-- Add CHECK OPTION to the View

CREATE OR ALTER VIEW Sales.USACusts WITH SCHEMABINDING

AS

SELECT

custid, companyname, contactname, contacttitle, address,

city, region, postalcode, country, phone, fax

FROM Sales.Customers

WHERE country = N'USA'

WITH CHECK OPTION;

GO

-- Notice that you can't insert a row through the view

/\*

INSERT INTO Sales.USACusts(

companyname, contactname, contacttitle, address,

city, region, postalcode, country, phone, fax)

VALUES(

N'Customer FGHIJ', N'Contact FGHIJ', N'Title FGHIJ', N'Address FGHIJ',

N'London', NULL, N'12345', N'UK', N'012-3456789', N'012-3456789');

\*/

GO

-- Cleanup

DELETE FROM Sales.Customers

WHERE custid > 91;

DROP VIEW IF EXISTS Sales.USACusts;

GO

---------------------------------------------------------------------

-- Inline User Defined Functions

---------------------------------------------------------------------

-- Creating GetCustOrders function

USE TSQLV6;

GO

CREATE OR ALTER FUNCTION dbo.GetCustOrders

(@cid AS INT) RETURNS TABLE

AS

RETURN

SELECT orderid, custid, empid, orderdate, requireddate,

shippeddate, shipperid, freight, shipname, shipaddress, shipcity,

shipregion, shippostalcode, shipcountry

FROM Sales.Orders

WHERE custid = @cid;

GO

-- Test Function

SELECT orderid, custid

FROM dbo.GetCustOrders(1) AS O;

SELECT O.orderid, O.custid, OD.productid, OD.qty

FROM dbo.GetCustOrders(1) AS O

INNER JOIN Sales.OrderDetails AS OD

ON O.orderid = OD.orderid;

GO

-- Cleanup

DROP FUNCTION IF EXISTS dbo.GetCustOrders;

GO

---------------------------------------------------------------------

-- APPLY

---------------------------------------------------------------------

SELECT S.shipperid, E.empid

FROM Sales.Shippers AS S

CROSS JOIN HR.Employees AS E;

SELECT S.shipperid, E.empid

FROM Sales.Shippers AS S

CROSS APPLY HR.Employees AS E;

-- 3 most recent orders for each customer

SELECT C.custid, A.orderid, A.orderdate

FROM Sales.Customers AS C

CROSS APPLY

(SELECT TOP (3) orderid, empid, orderdate, requireddate

FROM Sales.Orders AS O

WHERE O.custid = C.custid

ORDER BY orderdate DESC, orderid DESC) AS A;

-- With OFFSET-FETCH

SELECT C.custid, A.orderid, A.orderdate

FROM Sales.Customers AS C

CROSS APPLY

(SELECT orderid, empid, orderdate, requireddate

FROM Sales.Orders AS O

WHERE O.custid = C.custid

ORDER BY orderdate DESC, orderid DESC

OFFSET 0 ROWS FETCH NEXT 3 ROWS ONLY) AS A;

-- 3 most recent orders for each customer, preserve all customers

SELECT C.custid, A.orderid, A.orderdate

FROM Sales.Customers AS C

OUTER APPLY

(SELECT TOP (3) orderid, empid, orderdate, requireddate

FROM Sales.Orders AS O

WHERE O.custid = C.custid

ORDER BY orderdate DESC, orderid DESC) AS A;

GO

-- Creation Script for the Function TopOrders

CREATE OR ALTER FUNCTION dbo.TopOrders

(@custid AS INT, @n AS INT)

RETURNS TABLE

AS

RETURN

SELECT TOP (@n) orderid, empid, orderdate, requireddate

FROM Sales.Orders

WHERE custid = @custid

ORDER BY orderdate DESC, orderid DESC;

GO

SELECT

C.custid, C.companyname,

A.orderid, A.empid, A.orderdate, A.requireddate

FROM Sales.Customers AS C

CROSS APPLY dbo.TopOrders(C.custid, 3) AS A;