# Dropbox Chapter Files

Chapter 4 HW assignment files: <https://www.dropbox.com/sh/yli6isdvjsyini9/AABQPCH1U00SaQwK3_uOLRlqa/Ch04?dl=0&lst=>

Chapter 5 HW assignment files: <https://www.dropbox.com/sh/yli6isdvjsyini9/AADOdPvtSkK3vPzXTNZ-Tfkba/Ch05?dl=0&lst=>

# Responsibilities:

* Team Lead (Aleks)
  + Record Presentation
    - Talk about use of the Gantt chart
    - Talk about use of the To-Do list
    - Talk about 5 Exercises including propositions
    - Mention NACE Competencies Used
    - Mention how ChatGPT was used
  + Upload MP4 presentation it to Google drive
  + Upload SQL Notebook (with output from running the cell) to [Google drive](https://drive.google.com/drive/u/0/folders/16NIrtoE6GcTgdmK1UYekNDf_SQengKDB)
  + Collect assignments & submit final package to blackboard
* Team members
  + Record Presentation
    - Talk about use of the To-Do list
    - Talk about 5 Exercises including propositions
    - Mention NACE Competencies Used
    - Mention how ChatGPT was used
  + Upload MP4 presentation it to Google drive
  + Upload SQL Notebook (with output from running the cell) to [Google drive](https://drive.google.com/drive/u/0/folders/16NIrtoE6GcTgdmK1UYekNDf_SQengKDB)

# Assignment Splits:

## Sigi

#### Chapter 04 - Subqueries - Exercises.sql

*-- 7 (Optional, Advanced)*

*-- Write a query that returns customers*

*-- who ordered product 12*

*-- Tables involved: TSQLV4 database,*

*-- Customers, Orders and OrderDetails tables*

*-- Desired output:*

custid companyname

*----------- ----------------------------------------*

48 Customer DVFMB

39 Customer GLLAG

71 Customer LCOUJ

65 Customer NYUHS

44 Customer OXFRU

51 Customer PVDZC

86 Customer SNXOJ

20 Customer THHDP

90 Customer XBBVR

46 Customer XPNIK

31 Customer YJCBX

87 Customer ZHYOS

(12 **row**(s) affected)

*-- 8 (Optional, Advanced)*

*-- Write a query that calculates a running total qty*

*-- for each customer and month using subqueries*

*-- Tables involved: TSQLV4 database, Sales.CustOrders view*

*-- Desired output:*

custid ordermonth qty runqty

*----------- ----------------------- ----------- -----------*

1 2015-08-01 00:00:00.000 38 38

1 2015-10-01 00:00:00.000 41 79

1 2016-01-01 00:00:00.000 17 96

1 2016-03-01 00:00:00.000 18 114

1 2016-04-01 00:00:00.000 60 174

2 2014-09-01 00:00:00.000 6 6

2 2015-08-01 00:00:00.000 18 24

2 2015-11-01 00:00:00.000 10 34

2 2016-03-01 00:00:00.000 29 63

3 2014-11-01 00:00:00.000 24 24

3 2015-04-01 00:00:00.000 30 54

3 2015-05-01 00:00:00.000 80 134

3 2015-06-01 00:00:00.000 83 217

3 2015-09-01 00:00:00.000 102 319

3 2016-01-01 00:00:00.000 40 359

...

(636 **row**(s) affected)

#### Chapter 04 - Subqueries.sql

*---------------------------------------------------------------------*

*-- Substitution Error in a Subquery Column Name*

*---------------------------------------------------------------------*

*-- Create and populate table Sales.MyShippers*

**DROP** **TABLE** **IF** **EXISTS** Sales.MyShippers;

**CREATE** **TABLE** Sales.MyShippers

(

shipper\_id **INT** **NOT** **NULL**,

companyname **NVARCHAR**(40) **NOT** **NULL**,

phone **NVARCHAR**(24) **NOT** **NULL**,

**CONSTRAINT** PK\_MyShippers **PRIMARY** **KEY**(shipper\_id)

);

**INSERT** **INTO** Sales.MyShippers(shipper\_id, companyname, phone)

**VALUES**(1, N'Shipper GVSUA', N'(503) 555-0137'),

(2, N'Shipper ETYNR', N'(425) 555-0136'),

(3, N'Shipper ZHISN', N'(415) 555-0138');

**GO**

*-- Shippers who shipped orders to customer 43*

*-- Bug*

**SELECT** shipper\_id, companyname

**FROM** Sales.MyShippers

**WHERE** shipper\_id **IN**

(**SELECT** shipper\_id

**FROM** Sales.Orders

**WHERE** custid = 43);

**GO**

*-- The safe way using aliases, bug identified*

**SELECT** shipper\_id, companyname

**FROM** Sales.MyShippers

**WHERE** shipper\_id **IN**

(**SELECT** O.shipper\_id

**FROM** Sales.Orders **AS** O

**WHERE** O.custid = 43);

**GO**

*-- Bug corrected*

**SELECT** shipper\_id, companyname

**FROM** Sales.MyShippers

**WHERE** shipper\_id **IN**

(**SELECT** O.shipperid

**FROM** Sales.Orders **AS** O

**WHERE** O.custid = 43);

*-- Cleanup*

**DROP** **TABLE** **IF** **EXISTS** Sales.MyShippers;

#### Chapter 05 - Table Expressions - Exercises.sql

*-- 1*

*-- The following query attempts to filter orders placed on the last day of the year.*

**USE** TSQLV4;

**GO**

**SELECT** orderid, orderdate, custid, empid,

DATEFROMPARTS(YEAR(orderdate), 12, 31) **AS** endofyear

**FROM** Sales.Orders

**WHERE** orderdate <> endofyear;

*-- When you try to run this query you get the following error.*

*/\**

*Msg 207, Level 16, State 1, Line 233*

*Invalid column name 'endofyear'.*

*\*/*

*-- Explain what the problem is and suggest a valid solution.*

*-- 6-2*

*-- Using the CROSS APPLY operator*

*-- and the function you created in exercise 6-1,*

*-- return, for each supplier, the two most expensive products*

*-- Desired output*

supplierid companyname productid productname unitprice

*----------- --------------- ----------- --------------- ----------*

8 Supplier BWGYE 20 Product QHFFP 81.00

8 Supplier BWGYE 68 Product TBTBL 12.50

20 Supplier CIYNM 43 Product ZZZHR 46.00

20 Supplier CIYNM 44 Product VJIEO 19.45

23 Supplier ELCRN 49 Product FPYPN 20.00

23 Supplier ELCRN 76 Product JYGFE 18.00

5 Supplier EQPNC 12 Product OSFNS 38.00

5 Supplier EQPNC 11 Product QMVUN 21.00

...

(55 **row**(s) affected)

*-- When you’re done, run the following code for cleanup:*

**DROP** **VIEW** **IF** **EXISTS** Sales.VEmpOrders;

**DROP** **FUNCTION** **IF** **EXISTS** Production.TopProducts;

#### 

#### 

#### Chapter 05 - Table Expressions.sql

*---------------------------------------------------------------------*

*-- 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;

*---------------------------------------------------------------------*

*-- Views Described*

*---------------------------------------------------------------------*

*-- Creating USACusts View*

**DROP** **VIEW** **IF** **EXISTS** Sales.USACusts;

**GO**

**CREATE** **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**

*---------------------------------------------------------------------*

*-- Inline User Defined Functions*

*---------------------------------------------------------------------*

*-- Creating GetCustOrders function*

**USE** TSQLV4;

**DROP** **FUNCTION** **IF** **EXISTS** dbo.GetCustOrders;

**GO**

**CREATE** **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**

## Nicholas

#### Chapter 04 - Subqueries - Exercises.sql

*-- 1*

*-- Write a query that returns all orders placed on the last day of*

*-- activity that can be found in the Orders table*

*-- Tables involved: TSQLV4 database, Orders table*

*--Desired output*

orderid orderdate custid empid

*----------- ----------- ----------- -----------*

11077 2016-05-06 65 1

11076 2016-05-06 9 4

11075 2016-05-06 68 8

11074 2016-05-06 73 7

(4 **row**(s) affected)

*-- 3*

*-- Write a query that returns employees*

*-- who did not place orders on or after May 1st, 2016*

*-- Tables involved: TSQLV4 database, Employees and Orders tables*

*-- Desired output:*

empid FirstName lastname

*----------- ---------- --------------------*

3 Judy Lew

5 Sven Mortensen

6 Paul Suurs

9 Patricia Doyle

(4 **row**(s) affected)

#### Chapter 04 - Subqueries.sql

*---------------------------------------------------------------------*

*-- Scalar Subqueries*

*---------------------------------------------------------------------*

*-- Order with the maximum order ID*

**USE** TSQLV4;

**DECLARE** @maxid **AS** **INT** = (**SELECT** MAX(orderid)

**FROM** Sales.Orders);

**SELECT** orderid, orderdate, empid, custid

**FROM** Sales.Orders

**WHERE** orderid = @maxid;

**GO**

**SELECT** orderid, orderdate, empid, custid

**FROM** Sales.Orders

**WHERE** orderid = (**SELECT** MAX(O.orderid)

**FROM** Sales.Orders **AS** O);

*-- Scalar subquery expected to return one value*

**SELECT** orderid

**FROM** Sales.Orders

**WHERE** empid =

(**SELECT** E.empid

**FROM** HR.Employees **AS** E

**WHERE** E.lastname **LIKE** N'C%');

**GO**

**SELECT** orderid

**FROM** Sales.Orders

**WHERE** empid =

(**SELECT** E.empid

**FROM** HR.Employees **AS** E

**WHERE** E.lastname **LIKE** N'D%');

**GO**

**SELECT** orderid

**FROM** Sales.Orders

**WHERE** empid =

(**SELECT** E.empid

**FROM** HR.Employees **AS** E

**WHERE** E.lastname **LIKE** N'A%');

*---------------------------------------------------------------------*

*-- Multi-Valued Subqueries*

*---------------------------------------------------------------------*

**SELECT** orderid

**FROM** Sales.Orders

**WHERE** empid **IN**

(**SELECT** E.empid

**FROM** HR.Employees **AS** E

**WHERE** E.lastname **LIKE** N'D%');

**SELECT** O.orderid

**FROM** HR.Employees **AS** E

**INNER** **JOIN** Sales.Orders **AS** O

**ON** E.empid = O.empid

**WHERE** E.lastname **LIKE** N'D%';

*-- Orders placed by US customers*

**SELECT** custid, orderid, orderdate, empid

**FROM** Sales.Orders

**WHERE** custid **IN**

(**SELECT** C.custid

**FROM** Sales.Customers **AS** C

**WHERE** C.country = N'USA');

*-- Customers who placed no orders*

**SELECT** custid, companyname

**FROM** Sales.Customers

**WHERE** custid **NOT** **IN**

(**SELECT** O.custid

**FROM** Sales.Orders **AS** O);

*-- Missing order IDs*

**USE** TSQLV4;

**DROP** **TABLE** **IF** **EXISTS** dbo.Orders;

**CREATE** **TABLE** dbo.Orders(orderid **INT** **NOT** **NULL** **CONSTRAINT** PK\_Orders **PRIMARY** **KEY**);

**INSERT** **INTO** dbo.Orders(orderid)

**SELECT** orderid

**FROM** Sales.Orders

**WHERE** orderid % 2 = 0;

**SELECT** n

**FROM** dbo.Nums

**WHERE** n **BETWEEN** (**SELECT** MIN(O.orderid) **FROM** dbo.Orders **AS** O)

**AND** (**SELECT** MAX(O.orderid) **FROM** dbo.Orders **AS** O)

**AND** n **NOT** **IN** (**SELECT** O.orderid **FROM** dbo.Orders **AS** O);

*-- CLeanup*

**DROP** **TABLE** **IF** **EXISTS** dbo.Orders;

#### Chapter 05 - Table Expressions - Exercises.sql

*-- 5-1*

*-- Create a view that returns the total qty*

*-- for each employee and year*

*-- Tables involved: Sales.Orders and Sales.OrderDetails*

*-- Desired output when running:*

*-- SELECT \* FROM Sales.VEmpOrders ORDER BY empid, orderyear*

empid orderyear qty

*----------- ----------- -----------*

1 2014 1620

1 2015 3877

1 2016 2315

2 2014 1085

2 2015 2604

2 2016 2366

3 2014 940

3 2015 4436

3 2016 2476

4 2014 2212

4 2015 5273

4 2016 2313

5 2014 778

5 2015 1471

5 2016 787

6 2014 963

6 2015 1738

6 2016 826

7 2014 485

7 2015 2292

7 2016 1877

8 2014 923

8 2015 2843

8 2016 2147

9 2014 575

9 2015 955

9 2016 1140

(27 **row**(s) affected)

*-- 5-2 (Optional, Advanced)*

*-- Write a query against Sales.VEmpOrders*

*-- that returns the running qty for each employee and year*

*-- Tables involved: TSQLV4 database, Sales.VEmpOrders view*

*-- Desired output:*

empid orderyear qty runqty

*----------- ----------- ----------- -----------*

1 2014 1620 1620

1 2015 3877 5497

1 2016 2315 7812

2 2014 1085 1085

2 2015 2604 3689

2 2016 2366 6055

3 2014 940 940

3 2015 4436 5376

3 2016 2476 7852

4 2014 2212 2212

4 2015 5273 7485

4 2016 2313 9798

5 2014 778 778

5 2015 1471 2249

5 2016 787 3036

6 2014 963 963

6 2015 1738 2701

6 2016 826 3527

7 2014 485 485

7 2015 2292 2777

7 2016 1877 4654

8 2014 923 923

8 2015 2843 3766

8 2016 2147 5913

9 2014 575 575

9 2015 955 1530

9 2016 1140 2670

(27 **row**(s) affected)

*-- When you’re done, run the following code for cleanup:*

**DROP** **VIEW** **IF** **EXISTS** Sales.VEmpOrders;

**DROP** **FUNCTION** **IF** **EXISTS** Production.TopProducts;

#### 

#### 

#### Chapter 05 - Table Expressions.sql

*---------------------------------------------------------------------*

*-- 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;

## 

*---------------------------------------------------------------------*

*-- SCHEMABINDING*

*---------------------------------------------------------------------*

**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*

**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**

## Akiva

#### Chapter 04 - Subqueries - Exercises.sql

*-- 2 (Optional, Advanced)*

*-- Write a query that returns all orders placed*

*-- by the customer(s) who placed the highest number of orders*

*-- \* Note: there may be more than one customer*

*-- with the same number of orders*

*-- Tables involved: TSQLV4 database, Orders table*

*-- Desired output:*

custid orderid orderdate empid

*----------- ----------- ---------- -----------*

71 10324 2014-10-08 9

71 10393 2014-12-25 1

71 10398 2014-12-30 2

71 10440 2015-02-10 4

71 10452 2015-02-20 8

71 10510 2015-04-18 6

71 10555 2015-06-02 6

71 10603 2015-07-18 8

71 10607 2015-07-22 5

71 10612 2015-07-28 1

71 10627 2015-08-11 8

71 10657 2015-09-04 2

71 10678 2015-09-23 7

71 10700 2015-10-10 3

71 10711 2015-10-21 5

71 10713 2015-10-22 1

71 10714 2015-10-22 5

71 10722 2015-10-29 8

71 10748 2015-11-20 3

71 10757 2015-11-27 6

71 10815 2016-01-05 2

71 10847 2016-01-22 4

71 10882 2016-02-11 4

71 10894 2016-02-18 1

71 10941 2016-03-11 7

71 10983 2016-03-27 2

71 10984 2016-03-30 1

71 11002 2016-04-06 4

71 11030 2016-04-17 7

71 11031 2016-04-17 6

71 11064 2016-05-01 1

(31 **row**(s) affected)

*-- 4*

*-- Write a query that returns*

*-- countries where there are customers but not employees*

*-- Tables involved: TSQLV4 database, Customers and Employees tables*

*-- Desired output:*

country

*---------------*

Argentina

Austria

Belgium

Brazil

Canada

Denmark

Finland

France

Germany

Ireland

Italy

Mexico

Norway

Poland

Portugal

Spain

Sweden

Switzerland

Venezuela

(19 **row**(s) affected)

#### Chapter 04 - Subqueries.sql

*---------------------------------------------------------------------*

*-- Correlated Subqueries*

*---------------------------------------------------------------------*

*-- Orders with maximum order ID for each customer*

*-- Listing 4-1: Correlated Subquery*

**USE** TSQLV4;

**SELECT** custid, orderid, orderdate, empid

**FROM** Sales.Orders **AS** O1

**WHERE** orderid =

(**SELECT** MAX(O2.orderid)

**FROM** Sales.Orders **AS** O2

**WHERE** O2.custid = O1.custid);

**SELECT** MAX(O2.orderid)

**FROM** Sales.Orders **AS** O2

**WHERE** O2.custid = 85;

*-- Percentage of customer total*

**SELECT** orderid, custid, val,

CAST(100. \* val / (**SELECT** SUM(O2.val)

**FROM** Sales.OrderValues **AS** O2

**WHERE** O2.custid = O1.custid)

**AS** **NUMERIC**(5,2)) **AS** pct

**FROM** Sales.OrderValues **AS** O1

**ORDER** **BY** custid, orderid;

*---------------------------------------------------------------------*

*-- EXISTS*

*---------------------------------------------------------------------*

*-- Customers from Spain who placed orders*

**SELECT** custid, companyname

**FROM** Sales.Customers **AS** C

**WHERE** country = N'Spain'

**AND** **EXISTS**

(**SELECT** \* **FROM** Sales.Orders **AS** O

**WHERE** O.custid = C.custid);

*-- Customers from Spain who didn't place Orders*

**SELECT** custid, companyname

**FROM** Sales.Customers **AS** C

**WHERE** country = N'Spain'

**AND** **NOT** **EXISTS**

(**SELECT** \* **FROM** Sales.Orders **AS** O

**WHERE** O.custid = C.custid);

#### Chapter 05 - Table Expressions - Exercises.sql

*-- 4 (Optional, Advanced)*

*-- Write a solution using a recursive CTE that returns the*

*-- management chain leading to Patricia Doyle (employee ID 9)*

*-- Tables involved: HR.Employees*

*-- Desired output:*

empid mgrid firstname lastname

*----------- ----------- ---------- --------------------*

9 5 Patricia Doyle

5 2 Sven Mortensen

2 1 Don Funk

1 **NULL** Sara Davis

(4 **row**(s) affected)

*-- 6-1*

*-- Create an inline function that accepts as inputs*

*-- a supplier id (@supid AS INT),*

*-- and a requested number of products (@n AS INT)*

*-- The function should return @n products with the highest unit prices*

*-- that are supplied by the given supplier id*

*-- Tables involved: Production.Products*

*-- Desired output when issuing the following query:*

*-- SELECT \* FROM Production.TopProducts(5, 2)*

productid productname unitprice

*----------- ---------------------------------------- ---------------------*

12 Product OSFNS 38.00

11 Product QMVUN 21.00

(2 **row**(s) affected)

*-- When you’re done, run the following code for cleanup:*

**DROP** **VIEW** **IF** **EXISTS** Sales.VEmpOrders;

**DROP** **FUNCTION** **IF** **EXISTS** Production.TopProducts;

#### Chapter 05 - Table Expressions.sql

*---------------------------------------------------------------------*

*-- Derived Tables*

*---------------------------------------------------------------------*

**USE** TSQLV4;

**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;*

*\*/*

**GO**

*-- 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**

*---------------------------------------------------------------------*

*-- ENCRYPTION*

*---------------------------------------------------------------------*

**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**

**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**

## Edwin

#### Chapter 04 - Subqueries - Exercises.sql

*-- 5*

*-- Write a query that returns for each customer*

*-- all orders placed on the customer's last day of activity*

*-- Tables involved: TSQLV4 database, Orders table*

*-- Desired output:*

custid orderid orderdate empid

*----------- ----------- ----------- -----------*

1 11011 2016-04-09 3

2 10926 2016-03-04 4

3 10856 2016-01-28 3

4 11016 2016-04-10 9

5 10924 2016-03-04 3

...

87 11025 2016-04-15 6

88 10935 2016-03-09 4

89 11066 2016-05-01 7

90 11005 2016-04-07 2

91 11044 2016-04-23 4

(90 **row**(s) affected)

*-- 10 (Optional, Advanced)*

*-- Write a query that returns for each order the number of days that past*

*-- since the same customer’s previous order. To determine recency among orders,*

*-- use orderdate as the primary sort element and orderid as the tiebreaker.*

*-- Tables involved: TSQLV4 database, Sales.Orders table*

*-- Desired output:*

custid orderdate orderid diff

*----------- ---------- ----------- -----------*

1 2015-08-25 10643 **NULL**

1 2015-10-03 10692 39

1 2015-10-13 10702 10

1 2016-01-15 10835 94

1 2016-03-16 10952 61

1 2016-04-09 11011 24

2 2014-09-18 10308 **NULL**

2 2015-08-08 10625 324

2 2015-11-28 10759 112

2 2016-03-04 10926 97

...

(830 **row**(s) affected)

#### Chapter 04 - Subqueries.sql

*--------------------------------------------------------------------*

*-- Returning "Previous" or "Next" Value*

*---------------------------------------------------------------------*

**SELECT** orderid, orderdate, empid, custid,

(**SELECT** MAX(O2.orderid)

**FROM** Sales.Orders **AS** O2

**WHERE** O2.orderid < O1.orderid) **AS** prevorderid

**FROM** Sales.Orders **AS** O1;

**SELECT** orderid, orderdate, empid, custid,

(**SELECT** MIN(O2.orderid)

**FROM** Sales.Orders **AS** O2

**WHERE** O2.orderid > O1.orderid) **AS** nextorderid

**FROM** Sales.Orders **AS** O1;

*---------------------------------------------------------------------*

*-- Running Aggregates*

*---------------------------------------------------------------------*

**SELECT** orderyear, qty

**FROM** Sales.OrderTotalsByYear;

**SELECT** orderyear, qty,

(**SELECT** SUM(O2.qty)

**FROM** Sales.OrderTotalsByYear **AS** O2

**WHERE** O2.orderyear <= O1.orderyear) **AS** runqty

**FROM** Sales.OrderTotalsByYear **AS** O1

**ORDER** **BY** orderyear;

#### Chapter 05 - Table Expressions - Exercises.sql

*-- 3-1*

*-- Write a query that calculates a row number for each order*

*-- based on orderdate, orderid ordering*

*-- Tables involved: Sales.Orders*

*-- Desired output:*

orderid orderdate custid empid rownum

*----------- ----------- ----------- ----------- -------*

10248 2014-07-04 85 5 1

10249 2014-07-05 79 6 2

10250 2014-07-08 34 4 3

10251 2014-07-08 84 3 4

10252 2014-07-09 76 4 5

10253 2014-07-10 34 3 6

10254 2014-07-11 14 5 7

10255 2014-07-12 68 9 8

10256 2014-07-15 88 3 9

10257 2014-07-16 35 4 10

...

(830 **row**(s) affected)

*-- 3-2*

*-- Write a query that returns rows with row numbers 11 through 20*

*-- based on the row number definition in exercise 3-1*

*-- Use a CTE to encapsulate the code from exercise 3-1*

*-- Tables involved: Sales.Orders*

*-- Desired output:*

orderid orderdate custid empid rownum

*----------- ----------- ----------- ----------- -------*

10258 2014-07-17 20 1 11

10259 2014-07-18 13 4 12

10260 2014-07-19 56 4 13

10261 2014-07-19 61 4 14

10262 2014-07-22 65 8 15

10263 2014-07-23 20 9 16

10264 2014-07-24 24 6 17

10265 2014-07-25 7 2 18

10266 2014-07-26 87 3 19

10267 2014-07-29 25 4 20

(10 **row**(s) affected)

*-- When you’re done, run the following code for cleanup:*

**DROP** **VIEW** **IF** **EXISTS** Sales.VEmpOrders;

**DROP** **FUNCTION** **IF** **EXISTS** Production.TopProducts;

#### 

#### Chapter 05 - Table Expressions.sql

*---------------------------------------------------------------------*

*-- 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**

*---------------------------------------------------------------------*

*-- Views and ORDER BY*

*---------------------------------------------------------------------*

*-- ORDER BY in a View is not Allowed*

*/\**

*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*

**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 rder*

**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**

## Aleks

#### Chapter 04 - Subqueries - Exercises.sql

*-- 6*

*-- Write a query that returns customers*

*-- who placed orders in 2015 but not in 2016*

*-- Tables involved: TSQLV4 database, Customers and Orders tables*

*-- Desired output:*

custid companyname

*----------- ----------------------------------------*

21 Customer KIDPX

23 Customer WVFAF

33 Customer FVXPQ

36 Customer LVJSO

43 Customer UISOJ

51 Customer PVDZC

85 Customer ENQZT

(7 **row**(s) affected)

*-- 9*

*-- Explain the difference between IN and EXISTS*

#### Chapter 04 - Subqueries.sql

*---------------------------------------------------------------------*

*-- NULL Trouble*

*---------------------------------------------------------------------*

*-- Customers who didn't place orders*

*-- Using NOT IN*

**SELECT** custid, companyname

**FROM** Sales.Customers

**WHERE** custid **NOT** **IN**(**SELECT** O.custid

**FROM** Sales.Orders **AS** O);

*-- Add a row to the Orders table with a NULL custid*

**INSERT** **INTO** Sales.Orders

(custid, empid, orderdate, requireddate, shippeddate, shipperid,

freight, shipname, shipaddress, shipcity, shipregion,

shippostalcode, shipcountry)

**VALUES**(**NULL**, 1, '20160212', '20160212',

'20160212', 1, 123.00, N'abc', N'abc', N'abc',

N'abc', N'abc', N'abc');

*-- Following returns an empty set*

**SELECT** custid, companyname

**FROM** Sales.Customers

**WHERE** custid **NOT** **IN**(**SELECT** O.custid

**FROM** Sales.Orders **AS** O);

*-- Exclude NULLs explicitly*

**SELECT** custid, companyname

**FROM** Sales.Customers

**WHERE** custid **NOT** **IN**(**SELECT** O.custid

**FROM** Sales.Orders **AS** O

**WHERE** O.custid **IS** **NOT** **NULL**);

*-- Using NOT EXISTS*

**SELECT** custid, companyname

**FROM** Sales.Customers **AS** C

**WHERE** **NOT** **EXISTS**

(**SELECT** \*

**FROM** Sales.Orders **AS** O

**WHERE** O.custid = C.custid);

*-- Cleanup*

**DELETE** **FROM** Sales.Orders **WHERE** custid **IS** **NULL**;

**GO**

#### Chapter 05 - Table Expressions - Exercises.sql

*-- 2-1*

*-- Write a query that returns the maximum order date for each employee*

*-- Tables involved: TSQLV4 database, Sales.Orders table*

*--Desired output*

empid maxorderdate

*----------- -------------*

3 2016-04-30

6 2016-04-23

9 2016-04-29

7 2016-05-06

1 2016-05-06

4 2016-05-06

2 2016-05-05

5 2016-04-22

8 2016-05-06

(9 **row**(s) affected)

*-- 2-2*

*-- Encapsulate the query from exercise 2-1 in a derived table*

*-- Write a join query between the derived table and the Sales.Orders*

*-- table to return the Sales.Orders with the maximum order date for*

*-- each employee*

*-- Tables involved: Sales.Orders*

*-- Desired output:*

empid orderdate orderid custid

*----------- ----------- ----------- -----------*

9 2016-04-29 11058 6

8 2016-05-06 11075 68

7 2016-05-06 11074 73

6 2016-04-23 11045 10

5 2016-04-22 11043 74

4 2016-05-06 11076 9

3 2016-04-30 11063 37

2 2016-05-05 11073 58

2 2016-05-05 11070 44

1 2016-05-06 11077 65

(10 **row**(s) affected)

*-- When you’re done, run the following code for cleanup:*

**DROP** **VIEW** **IF** **EXISTS** Sales.VEmpOrders;

**DROP** **FUNCTION** **IF** **EXISTS** Production.TopProducts;

#### 

#### Chapter 05 - Table Expressions.sql

*---------------------------------------------------------------------*

*-- 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 (Optional, Advanced)*

*---------------------------------------------------------------------*

**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;

*---------------------------------------------------------------------*

*-- 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 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;

*-- Creation Script for the Function TopOrders*

**DROP** **FUNCTION** **IF** **EXISTS** dbo.TopOrders;

**GO**

**CREATE** **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;