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

-- T-SQL Fundamentals Fourth Edition

-- Chapter 05 - Table Expressions

-- Exercises

-- © Itzik Ben-Gan

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

-- 1

-- The following query attempts to filter orders that were not placed on the last day of the year.

USE TSQLV6;

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 17

Invalid column name 'endofyear'.

\*/

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

-- 2-1

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

-- Tables involved: TSQLV6 database, Sales.Orders table

--Desired output

empid maxorderdate

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

3 2022-04-30

6 2022-04-23

9 2022-04-29

7 2022-05-06

1 2022-05-06

4 2022-05-06

2 2022-05-05

5 2022-04-22

8 2022-05-06

(9 rows 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 2022-04-29 11058 6

8 2022-05-06 11075 68

7 2022-05-06 11074 73

6 2022-04-23 11045 10

5 2022-04-22 11043 74

4 2022-05-06 11076 9

3 2022-04-30 11063 37

2 2022-05-05 11073 58

2 2022-05-05 11070 44

1 2022-05-06 11077 65

(10 rows affected)

-- 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 2020-07-04 85 5 1

10249 2020-07-05 79 6 2

10250 2020-07-08 34 4 3

10251 2020-07-08 84 3 4

10252 2020-07-09 76 4 5

10253 2020-07-10 34 3 6

10254 2020-07-11 14 5 7

10255 2020-07-12 68 9 8

10256 2020-07-15 88 3 9

10257 2020-07-16 35 4 10

...

(830 rows 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 2020-07-17 20 1 11

10259 2020-07-18 13 4 12

10260 2020-07-19 56 4 13

10261 2020-07-19 61 4 14

10262 2020-07-22 65 8 15

10263 2020-07-23 20 9 16

10264 2020-07-24 24 6 17

10265 2020-07-25 7 2 18

10266 2020-07-26 87 3 19

10267 2020-07-29 25 4 20

(10 rows affected)

-- 4

-- 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 rows affected)

-- 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 2020 1620

1 2021 3877

1 2022 2315

2 2020 1085

2 2021 2604

2 2022 2366

3 2020 940

3 2021 4436

3 2022 2476

4 2020 2212

4 2021 5273

4 2022 2313

5 2020 778

5 2021 1471

5 2022 787

6 2020 963

6 2021 1738

6 2022 826

7 2020 485

7 2021 2292

7 2022 1877

8 2020 923

8 2021 2843

8 2022 2147

9 2020 575

9 2021 955

9 2022 1140

(27 rows affected)

-- 5-2

-- Write a query against Sales.VEmpOrders

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

-- Tables involved: TSQLV6 database, Sales.VEmpOrders view

-- Desired output:

empid orderyear qty runqty

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

1 2020 1620 1620

1 2021 3877 5497

1 2022 2315 7812

2 2020 1085 1085

2 2021 2604 3689

2 2022 2366 6055

3 2020 940 940

3 2021 4436 5376

3 2022 2476 7852

4 2020 2212 2212

4 2021 5273 7485

4 2022 2313 9798

5 2020 778 778

5 2021 1471 2249

5 2022 787 3036

6 2020 963 963

6 2021 1738 2701

6 2022 826 3527

7 2020 485 485

7 2021 2292 2777

7 2022 1877 4654

8 2020 923 923

8 2021 2843 3766

8 2022 2147 5913

9 2020 575 575

9 2021 955 1530

9 2022 1140 2670

(27 rows 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 rows affected)

-- 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 rows affected)

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

DROP VIEW IF EXISTS Sales.VEmpOrders;

DROP FUNCTION IF EXISTS Production.TopProducts;