POWERING
POTENTIAL

Module 1: Table Expressions

Version 1.6

Table Expressions

- Views
- Table-valued functions
- Derived tables
- Temporary tables
- Comparison

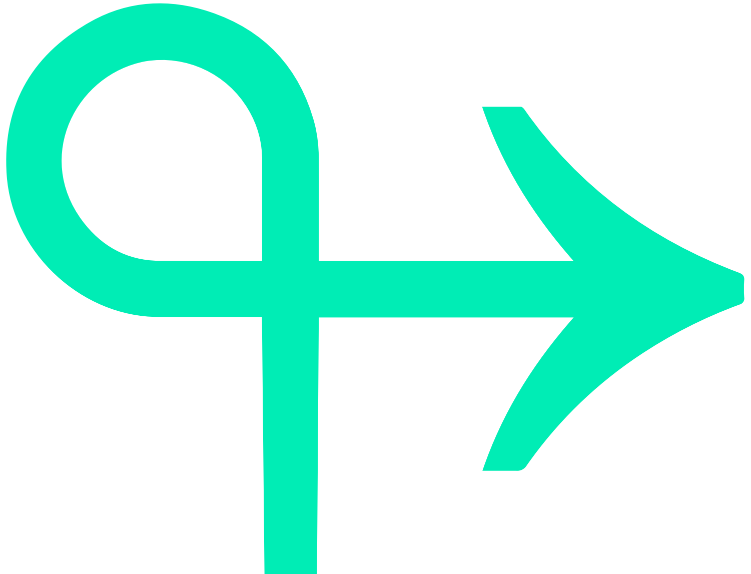
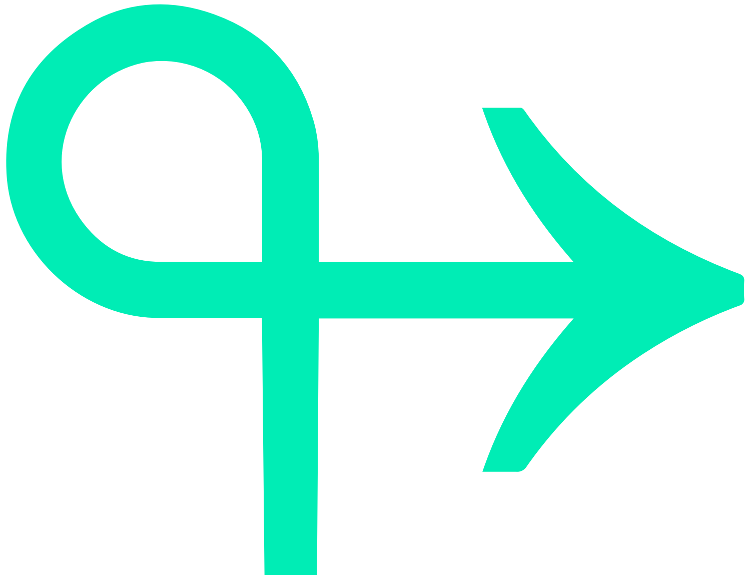


Table Expressions

- Can be used to simplify the TSQL code by dividing the query into more easily understood parts.

- Temporary tables and table variables are not table expressions but can be used to perform the same purpose.



Views

- Views are defined using a single SELECT statement.
- A view's definition is stored within the database for future use.
- A view can simplify the statements written by others where the same logic is reused.
- Administrators may use views to add security by not allowing access to the tables directly.
- ORDER BY is only permitted in a view if TOP, OFFSET/FETCH or FOR XML is used.

CREATING VIEWS

Command outline:

```
CREATE VIEW <viewname> AS
    SELECT <columns>
    FROM <table(s) including joins>
```

Demonstration:

```
CREATE VIEW dbo.SaleableProducts AS
    SELECT PSC.name AS Subcategory, Name, ListPrice, Color, Weight
    FROM Production.Product AS P
        INNER JOIN Production.ProductSubcategory AS PSC
            ON P.ProductSubcategoryID = PSC.ProductSubcategoryID
```

Use:

```
SELECT * FROM dbo.SaleableProducts
```

Module 1
Exercise 1 – Views

- Please complete exercise 1 of the Module 1 lab
- Stop at the end of task 2

Table-Valued Functions (TVF)

User defined table-valued functions

- The definition is stored within the database for others to use.
- Support input parameters.
- Used like a view.
- Use two-part naming convention when referring to objects such as tables within a function definition.

Table-valued Functions (TVF)

Two types of user defined function can return tables.

In-line:
- A single select statement.

Multi-line:
- Allows multiple lines for a more complex query.
- Table is pre-defined with the function and uses code to add rows.
- Use the create, alter and drop DDL statements.

CREATING IN-LINE TVF

Command outline:

```
CREATE FUNCTION <functionname> (<parameters>)
RETURNS TABLE AS
    (SELECT <columns>
    FROM <table(s) including joins>)
```

Demonstration:

```
CREATE FUNCTION dbo.GetProductsByColour(@Colour varchar(20))
    RETURNS TABLE AS
    RETURN(
        SELECT PSC.Name AS Subcategory, P.Name
            FROM Production.Product AS P
                INNER JOIN Production.ProductSubcategory AS PSC
                    ON P.ProductSubcategoryID = PSC.ProductSubcategoryID
            WHERE Color = @Colour
    )
```

Use:

```
SELECT * FROM dbo.GetProductsByColour(‘Red’)
```

CREATING IN-LINE TVF

Use:

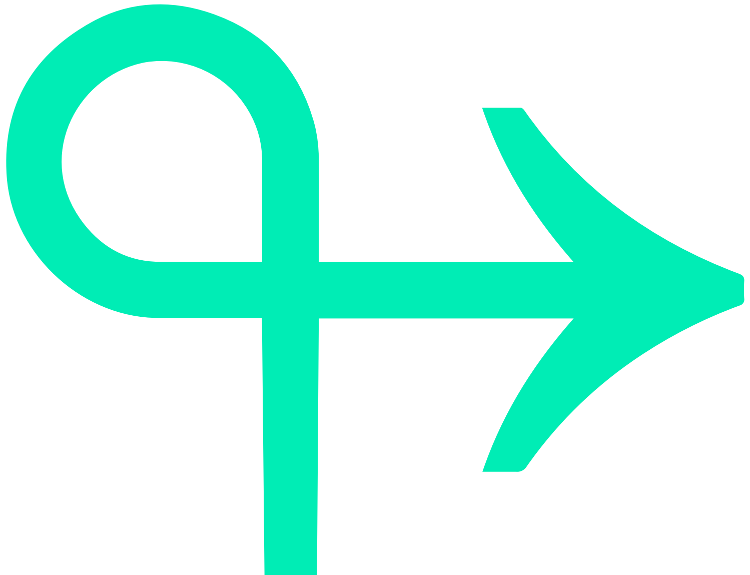
```
SELECT * FROM dbo.GetProductsByColour2(‘Blue’)  -- returns all blue products

SELECT * FROM dbo.GetProductsByColour2(‘Blue,Black’)  -- returns empty set
```

Module 1
exercise 2 – in-line table valued functions

- Please complete exercise 2 of the Module 1 lab
- Stop at the end of task 3

Derived tables

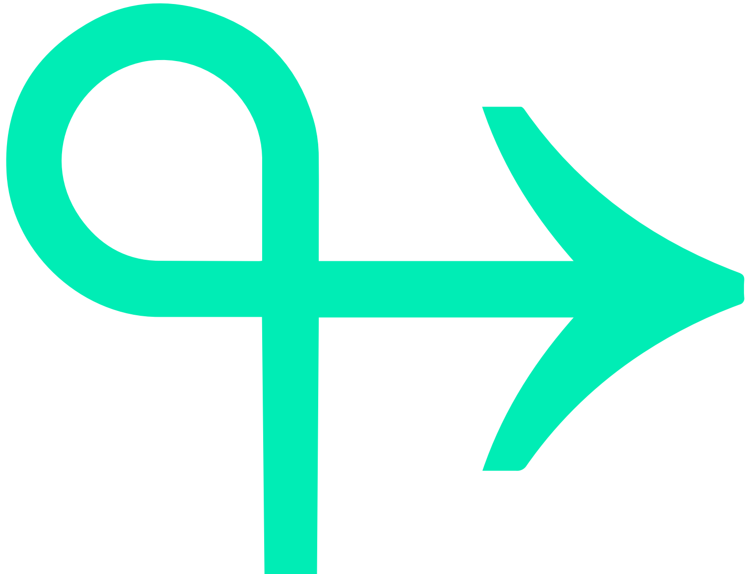
- Derived tables are named query expressions created within an outer SELECT statement.
- Not stored in database – represents a virtual relational table.
- When processed, unpacked into query against underlying referenced objects.
- Allow you to write more modular queries.
- Scope of a derived table is the query in which it is defined.



DERIVED TABLE RULES

Must:

- have an alias. AS
- have unique names for all columns.
- not use the ORDER BY clause unless in conjunction with TOP, OFFSET / FETCH, or FOR XML.
- use **From** (Select Query)



DERIVED TABLE RULES

Column names

- Defined as part of the inner query

```
SELECT *
     FROM (
          SELECT ProductSubcategoryID, avg(ListPrice) AS
AvgPrice
               FROM Production.Product
               GROUP BY ProductSubCategoryID
     ) AS Derivedtable
```

- Defined as part of the alias definition

```
SELECT *
     FROM (
          SELECT ProductSubcategoryID, avg(ListPrice)
               FROM Production.Product
               GROUP BY ProductSubCategoryID
     ) AS Derivedtable (SubcategoryID, AvgPrice)
```

DERIVED TABLE EXAMPLE

- Show all products with above average list price for their subcategory

```
SELECT P.ProductID, P.Name, P.ListPrice, DT.AvgPrice
    FROM Production.Product AS P
    INNER JOIN
    (
        SELECT ProductSubcategoryID, avg(ListPrice) AS AvgPrice
            FROM Production.Product
            GROUP BY ProductSubcategoryID
    ) AS DT
    ON P.ProductSubcategoryID = DT.ProductSubcategoryID
    WHERE P.ListPrice >= DT.AvgPrice
```

Module 1
exercise 3 – derived tables

- Please complete exercise 3 of the Module 1 lab
- Stop at the end of task 2

TEMPORARY TABLES

- Temporary tables can be used to store the results of a query for use later.

- Stored within TempDB (dropped when the user connection is dropped)
- Use **WITH**

Can be:

- **local (#)** – only available to the creator, and is dropped automatically when the session ends.
- **global (##)** – available to all the users on the instance, and is dropped when all the users who have used the table finish their sessions.

Commands:

- **CREATE** – creates the temporary table.
- **ALTER** – updates the design of the table.
- **DROP** – deletes the table.

TEMPORARY TABLES

Command outline:

```
CREATE TABLE (# or ##)TableName(
      column definitions
)

INSERT INTO (# or ##)TableName
      SELECT / VALUES to insert

DROP TABLE (# or ##)TableName
```

TEMPORARY TABLES

Demonstration:

```
CREATE TABLE #Averages(
      SubcategoryID int,
      AverageListPrice money
)
GO

INSERT INTO #Averages
      SELECT ProductSubcategoryID, avg(ListPrice) AS AvgPrice
                  FROM Production.Product
                  GROUP BY ProductSubcategoryID
GO

SELECT *
      FROM #Averages
GO

DROP TABLE #Averages
```

Module 1
exercise 5 – temporary tables

- Please complete exercise 5 of the Module 1 lab
- Stop at the end of task 3

COMPARISON

	Views	Table-valued function	Derived tables	Temporary table
Main use	Storing the definition of a query for use later	Storing the definition of code for use later	Writing more complex queries than are possible normally	Storing a dataset for reuse later, by the session owner or other sessions
Definition stored inside database	●	●		
Data exists	Single execution of the view	Single execution of the TVF	Single execution of the query	Session
Design shared with others	●	●		Depends on type
Recursive				
Allow parameters to be passed		●		
Allows access to declared variables		Only declared variables within the function Declared variables cannot be used with in-line TVF	●	

POWERING
POTENTIAL

Review

- Views
- Table-valued functions
- Derived tables
- Temporary tables
- Comparison