Module 1

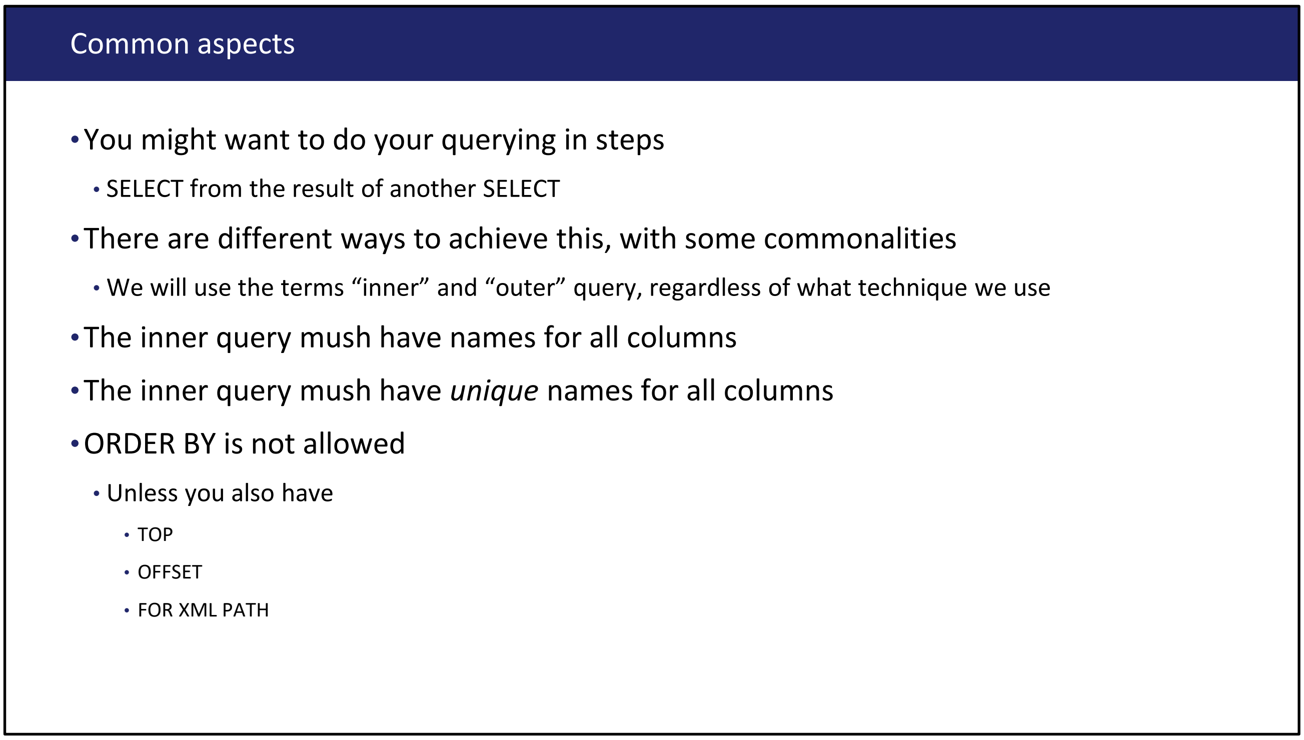
Table expressions

Module Overview

- Using the result of a query as a table
- Derived tables
- Common Table Expressions
- Views
- Using Inline Table-Valued Functions

Lesson: Using the result of a query as a table

- Common aspects



Common aspects

- You might want to do your querying in steps
  - SELECT from the result of another SELECT
- There are different ways to achieve this, with some commonalities
  - We will use the terms “inner” and “outer” query, regardless of what technique we use
- The inner query mush have names for all columns
- The inner query mush have *unique* names for all columns
- ORDER BY is not allowed
  - Unless you also have
    - TOP
    - OFFSET
    - FOR XML PATH

Fundamentals of table expressions, Part 1
https://sqlperformance.com/2020/04/t-sql-queries/table-expressions-part-1

Lesson: Using derived tables

- Writing Queries with Derived Tables
- Guidelines for Derived Tables
- Using Aliases for Column Names in Derived Tables
- Referring to variables and parameters
- Nesting and Reusing Derived Tables

Writing Queries with 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

```
SELECT <column_list>
FROM  (
        <derived_table_definition>
        ) AS <derived_table_alias>
```

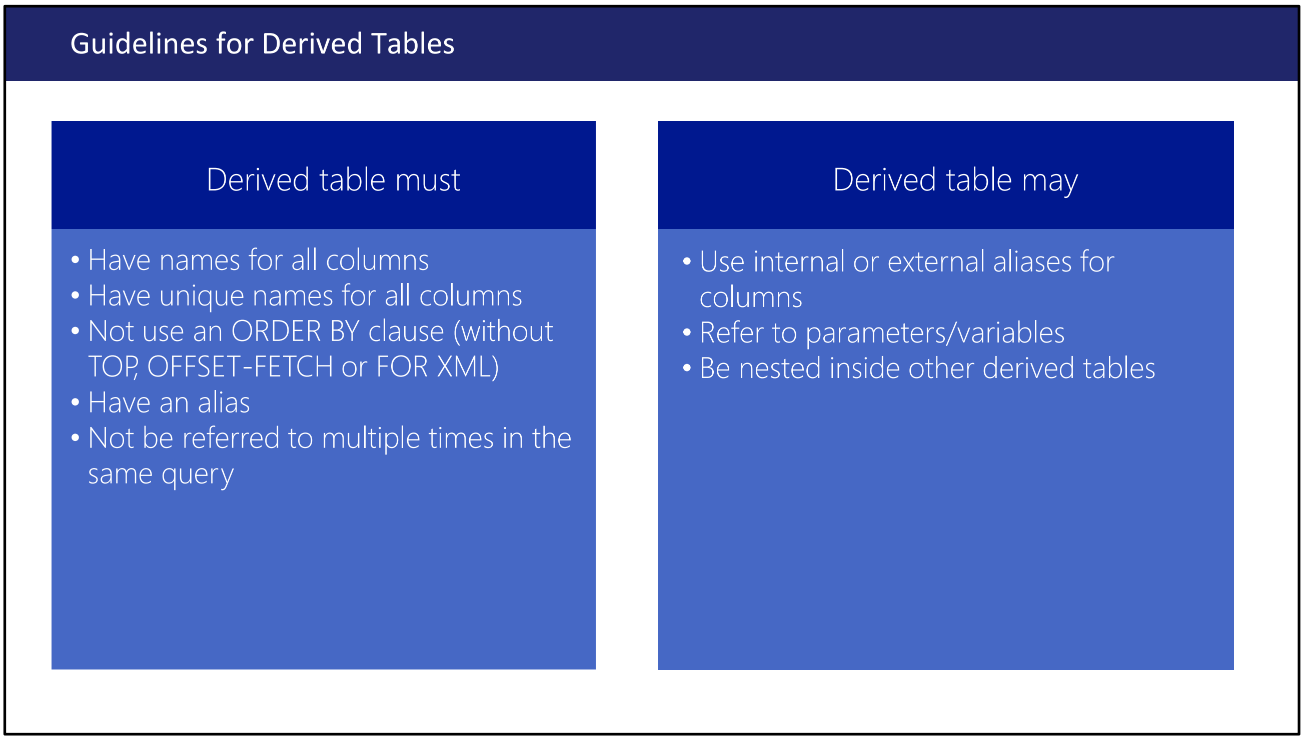
- Scope of a derived table is the query in which it is defined

SQL Server Derived Table Example
https://www.mssqltips.com/sqlservertip/6038/sql-server-derived-table-example/

Fundamentals of table expressions, Part 2 – Derived tables, logical considerations
https://sqlperformance.com/2020/05/t-sql-queries/table-expressions-part-2

Fundamentals of table expressions, Part 3 – Derived tables, optimization considerations
https://sqlperformance.com/2020/06/t-sql-queries/table-expressions-part-3

Fundamentals of table expressions, Part 4 – Derived tables, optimization considerations, continued
https://sqlperformance.com/2020/07/t-sql-queries/table-expressions-part-4



Guidelines for Derived Tables

Derived table must

- Have names for all columns
- Have unique names for all columns
- Not use an ORDER BY clause (without TOP, OFFSET-FETCH or FOR XML)
- Have an alias
- Not be referred to multiple times in the same query

Derived table may

- Use internal or external aliases for columns
- Refer to parameters/variables
- Be nested inside other derived tables

Using Aliases for Column Names in Derived Tables

- Column aliases may be defined inline:

```
SELECT orderyear, COUNT(DISTINCT custid) AS cust_count
FROM (
        SELECT YEAR(orderdate) AS orderyear, custid
        FROM Sales.Orders) AS derived_year
GROUP BY orderyear;
```

- Column aliases may be defined externally:

```
SELECT orderyear, COUNT(DISTINCT custid) AS cust_count
FROM (
        SELECT YEAR(orderdate), custid
        FROM Sales.Orders) AS
        derived_year(orderyear, custid)
GROUP BY orderyear;
```

Referring to variables and parameters

- Derived tables may refer to variables and parameters
  - Variables declared in the same batch as the SELECT statement
  - Parameters passed into a table-valued function or stored procedure

```
DECLARE @emp_id INT = 9;
SELECT orderyear, COUNT(DISTINCT custid) AS cust_count
FROM (
          SELECT YEAR(orderdate) AS orderyear, custid
          FROM Sales.Orders
          WHERE empid=@emp_id
) AS derived_year
GROUP BY orderyear;
```

Nesting and reusing derived tables

- Derived tables may be nested, though not recommended:

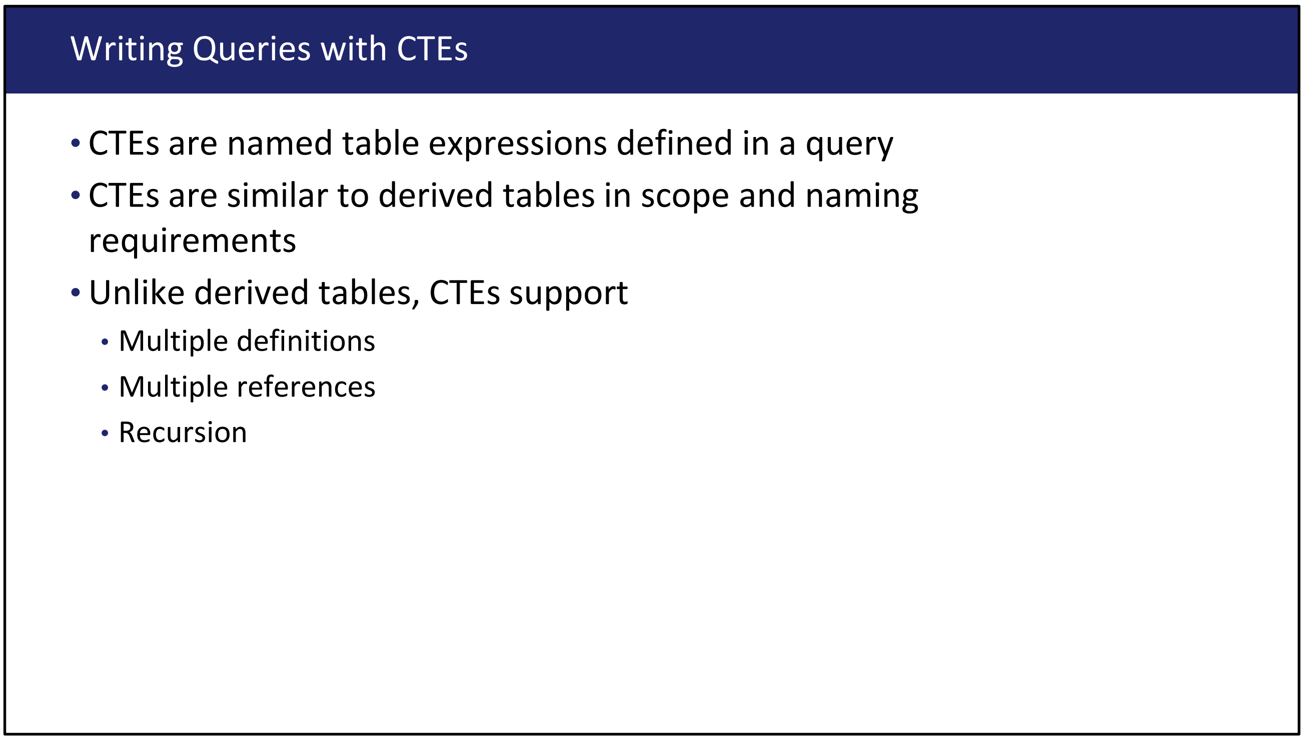
```
SELECT orderyear, cust_count
FROM (
          SELECT  orderyear,
          COUNT(DISTINCT custid) AS cust_count
          FROM (
                    SELECT YEAR(orderdate) AS orderyear ,custid
                    FROM Sales.Orders) AS derived_table_1
          GROUP BY orderyear) AS derived_table_2
WHERE cust_count > 80;
```

- Derived tables may not be referred to multiple times in the same query
  - Each reference must be separately defined

Demo derived tables

Lesson: Using CTEs

- Writing Queries with CTEs
- Creating Queries with Common Table Expressions



Writing Queries with CTEs

- CTEs are named table expressions defined in a query
- CTEs are similar to derived tables in scope and naming requirements
- Unlike derived tables, CTEs support
  - Multiple definitions
  - Multiple references
  - Recursion

WITH common_table_expression (Transact-SQL)
https://docs.microsoft.com/en-us/sql/t-sql/queries/with-common-table-expression-transact-sql

SQL Server Common Table Expressions (CTE)
https://www.sqlshack.com/sql-server-common-table-expressions-cte/

Fundamentals of table expressions, Part 5 – CTEs, logical considerations
https://sqlperformance.com/2020/08/t-sql-queries/table-expressions-part-5

Fundamentals of table expressions, Part 6 – Recursive CTEs
https://sqlperformance.com/2020/09/t-sql-queries/fundamentals-of-table-expressions-part-6-recursive-ctes

Fundamentals of table expressions, Part 7 – CTEs, optimization considerations
https://sqlperformance.com/2020/10/t-sql-queries/table-expressions-part-7

Fundamentals of table expressions, Part 8 – CTEs, optimization considerations continued
https://sqlperformance.com/2020/11/t-sql-queries/table-expressions-part-8

Creating Queries with Common Table Expressions

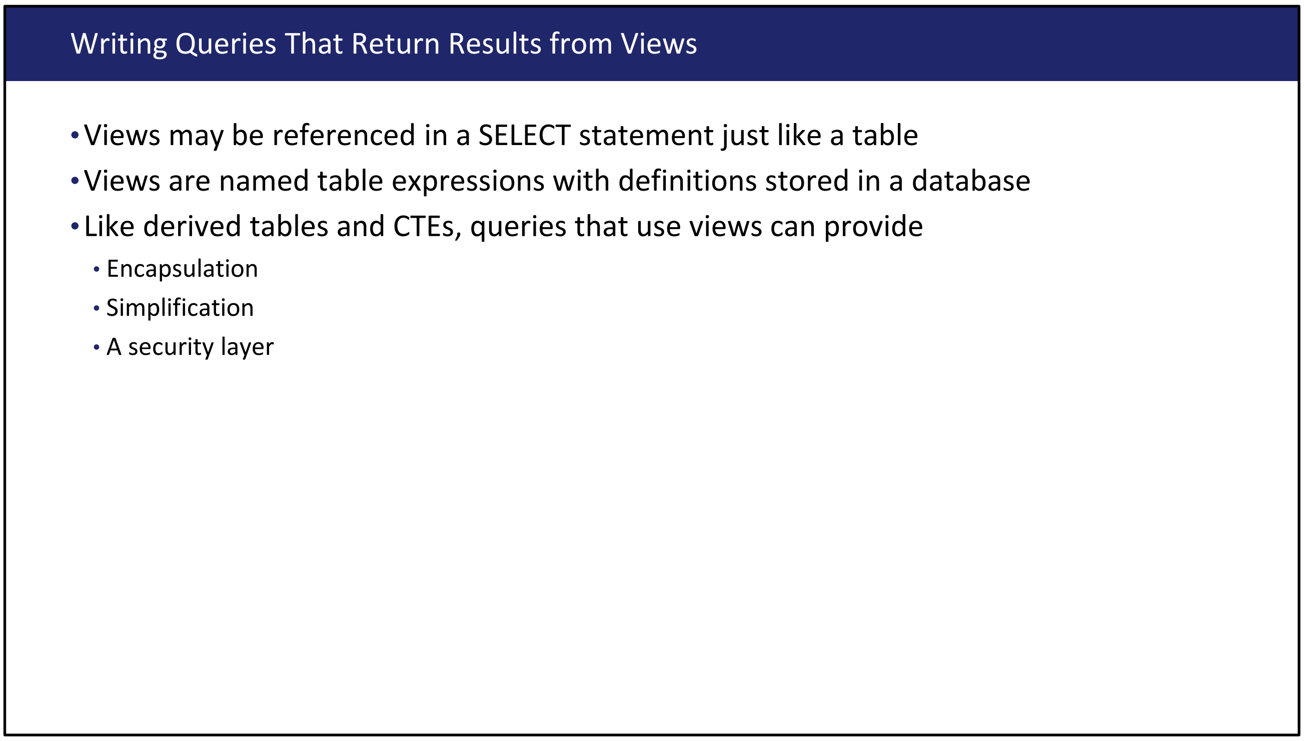
- To create a CTE:
  - Define the table expression in a WITH clause
  - Assign column aliases (inline or external)
  - Reference the CTE in the following/outer query

```
;WITH CTE_year AS
        (
        SELECT YEAR(orderdate) AS orderyear, custid
        FROM Sales.Orders
        )
SELECT orderyear, COUNT(DISTINCT custid) AS cust_count
FROM CTE_year
GROUP BY orderyear;
```

Demo Common table expressions

Lesson: Using Views

- Writing Queries That Return Results from Views
- Creating Simple Views



Writing Queries That Return Results from Views

- Views may be referenced in a SELECT statement just like a table
- Views are named table expressions with definitions stored in a database
- Like derived tables and CTEs, queries that use views can provide
  - Encapsulation
  - Simplification
  - A security layer

Views (Transact-SQL)
https://docs.microsoft.com/en-us/sql/relational-databases/views/views

SQL View – A complete introduction and walk-through
https://www.sqlshack.com/sql-view-a-complete-introduction-and-walk-through/

Fundamentals of table expressions, Part 9 – Views, compared with derived tables and CTEs
https://sqlperformance.com/2021/06/t-sql-queries/table-expressions-part-9

Fundamentals of table expressions, Part 10 – Views, SELECT *, and DDL changes
https://sqlperformance.com/2021/07/t-sql-queries/table-expressions-part-10

Fundamentals of Table Expressions, Part 11 – Views, Modification Considerations
https://sqlperformance.com/2021/09/t-sql-queries/table-expressions-part-11

Creating Simple Views

- Views are saved queries created in a database by administrators and developers
- Views are defined with a single SELECT statement
- ORDER BY is not permitted in a view definition without the use of
  - TOP
  - OFFSET/FETCH
  - FOR XML
- To sort the output, use ORDER BY in the outer query

```sql
CREATE VIEW HR.EmpPhoneList
AS
SELECT empid, lastname, firstname, phone
FROM HR.Employees;
```

Demo using views

Lesson: Using Inline Table Valued Functions TVFs

- Writing Queries That Use Inline TVFs
- Creating Simple Inline TVFs
- Retrieving from Inline TVFs

Writing Queries That Use Inline TVFs

- TVFs are named table expressions with definitions stored in a database
- TVFs return a virtual table to the calling query
- SQL Server provides two types of TVFs:
  - Inline, based on a single SELECT statement
  - Multi-statement, which creates and loads a table variable
- Unlike views, TVFs support input parameters
- Inline TVFs may be thought of as parameterized views

CREATE FUNCTION (Transact-SQL)
https://docs.microsoft.com/en-us/sql/t-sql/statements/create-function-transact-sql

SQL Server inline table-valued functions
https://www.sqlshack.com/sql-server-inline-table-valued-function/

Fundamentals of Table Expressions, Part 12 – Inline Table-Valued Functions
https://sqlperformance.com/2021/10/t-sql-queries/table-expressions-part-12

Fundamentals of Table Expressions, Part 13 – Inline Table-Valued Functions, Continued
https://sqlperformance.com/2021/11/t-sql-queries/table-expressions-part-13

Creating Simple Inline TVFs

- TVFs are created by administrators and developers
- Create and name function and optional parameters with CREATE FUNCTION
- Declare return type as TABLE
- Define inline SELECT statement following RETURN

```
CREATE FUNCTION Sales.fn_LineTotal (@orderid INT)
RETURNS TABLE
AS
RETURN
   SELECT orderid,
         CAST((qty * unitprice * (1 – discount)) AS
         DECIMAL(8, 2)) AS line_total
   FROM    Sales.OrderDetails
   WHERE   orderid = @orderid ;
```

Retrieving from Inline TVFs

- SELECT from function
- Use two-part name
- Pass in parameters

```
SELECT orderid, line_total
FROM Sales.fn_LineTotal(10252) AS LT;
```

orderid	line_total
10252	2462.40
10252	47.50
10252	1088.00

Demo Table valued functions

V1 Lab 1: Using Table Expressions

- ***Note carefully the folder name in the lab instructions!!!***
- Exercise 1: Writing Queries That Use Views
- Exercise 2: Writing Queries That Use Derived Tables
- Exercise 3: Writing Queries That Use CTEs
- Exercise 4: Writing Queries That Use Inline TVFs

Estimated Time: 60 minutes

V2 Lab 1: Table expressions

- Ex 1. Derived table
- Ex 2. Common Table Expression
- Ex 3. View

Estimated Time: 60 minutes