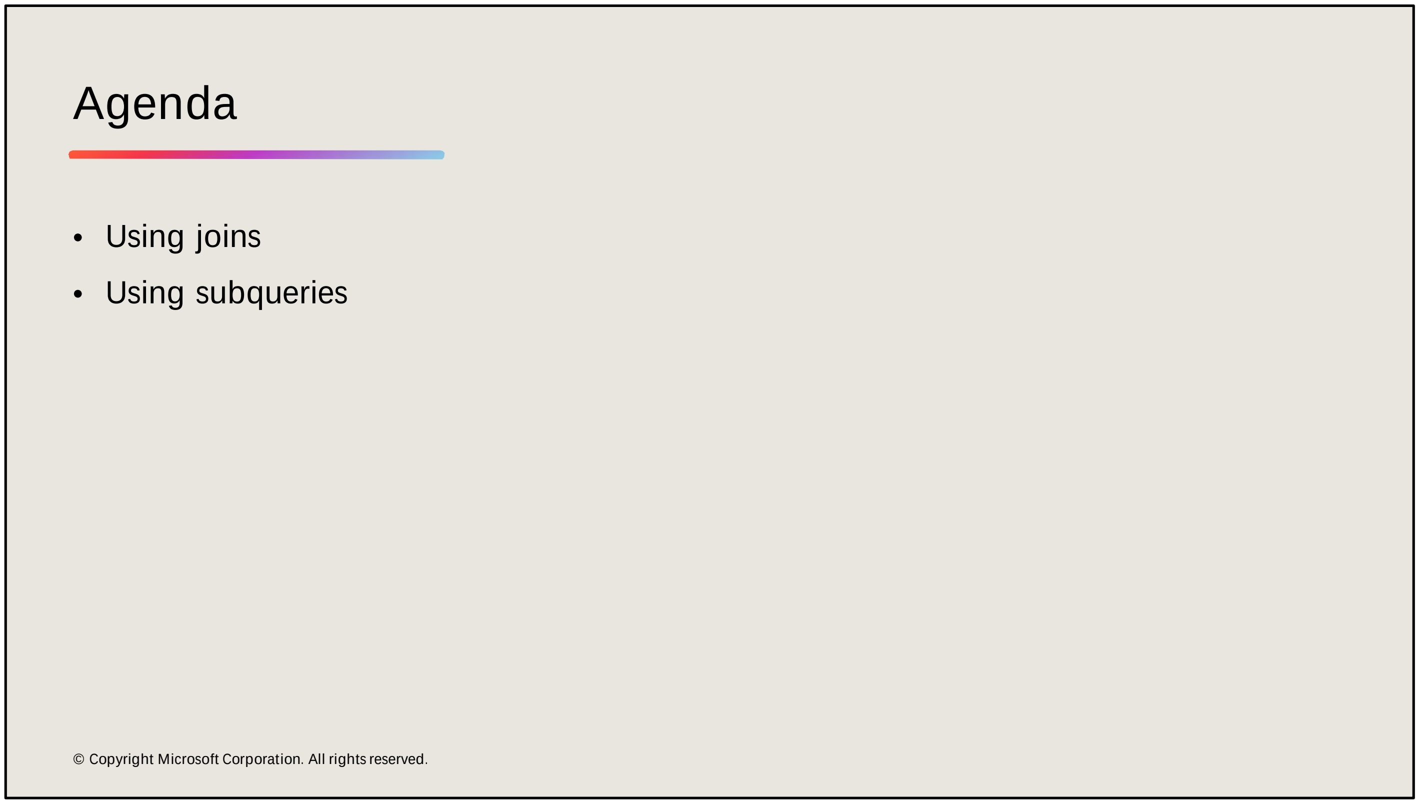
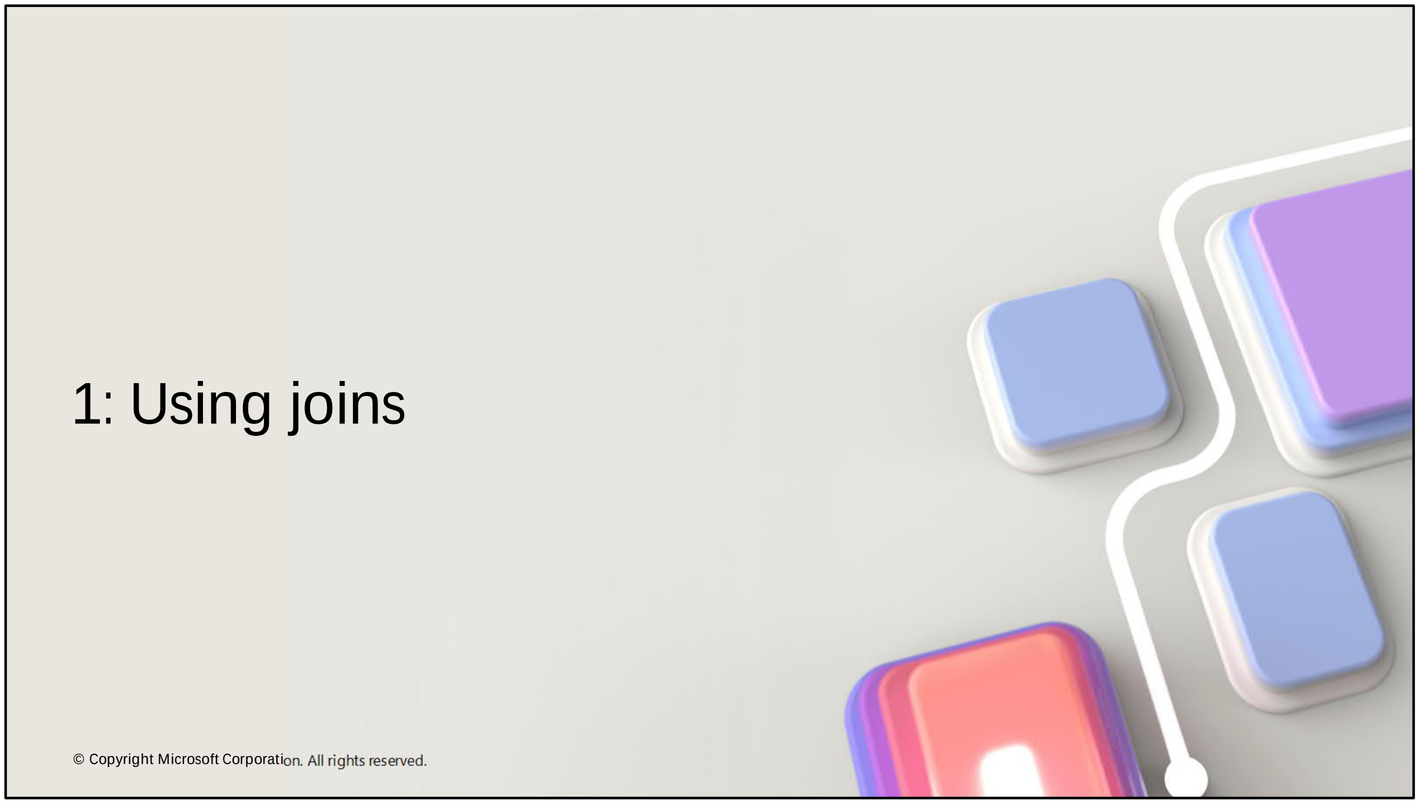
Microsoft

5: Using joins and subqueries



Agenda

- Using joins
- Using subqueries



1: Using joins

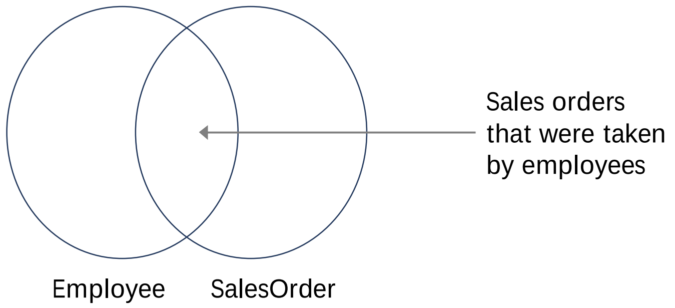
Relational databases usually contain multiple tables that are linked by common *key* fields. This *normalized* design minimizes duplication of data, but means that you'll often need to write queries to retrieve related data from two or more tables.

In this module, you'll learn how to:

- Understand join concepts and syntax
- Write queries that use inner joins
- Write queries that use outer joins
- Write queries that use cross joins
- Write queries that use self-joins

Join concepts

It can help to think of the tables as sets in a Venn diagram

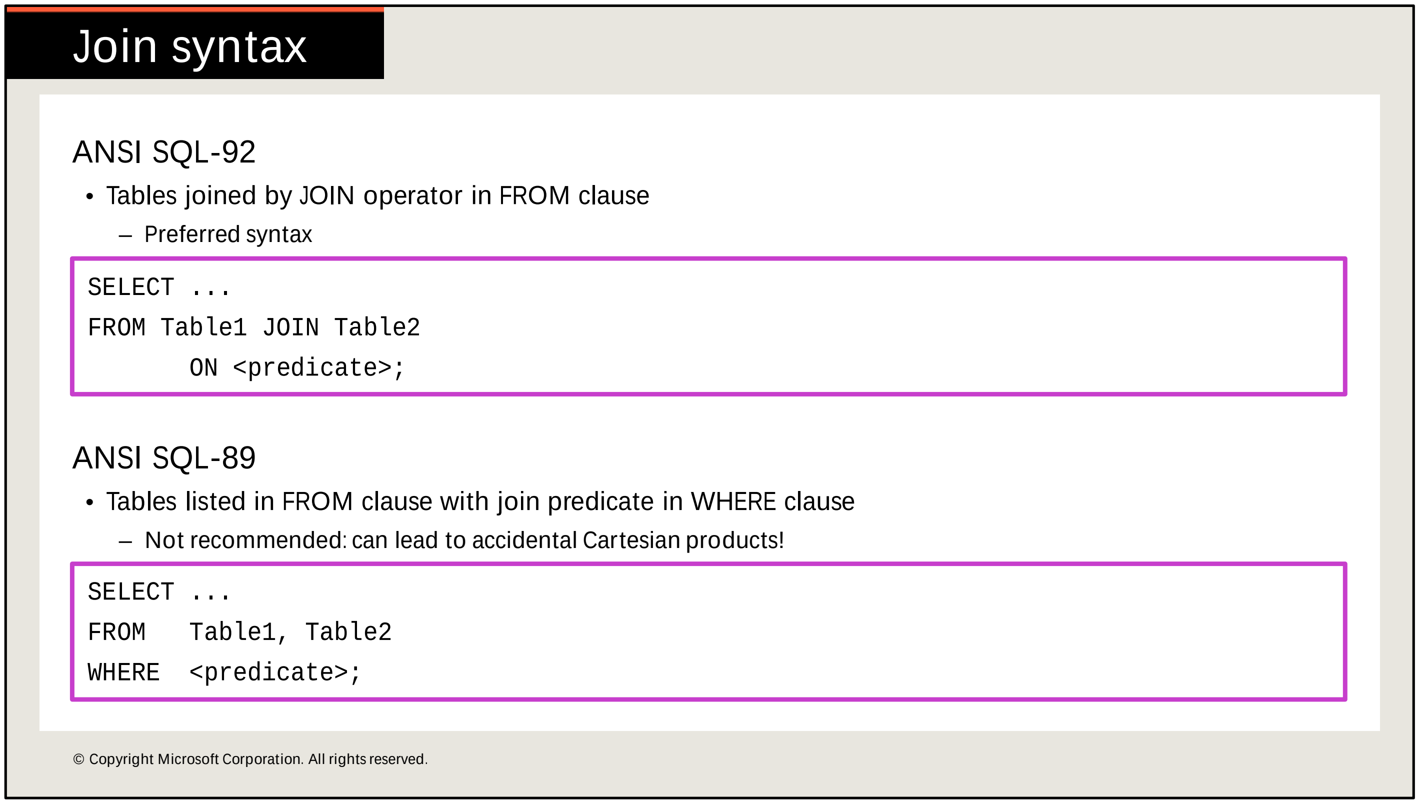


Combine rows from multiple tables by specifying matching criteria

Usually based on primary key – Foreign key relationships

For example, return rows that combine data from the Employee and SalesOrder tables by matching the Employee.EmployeeID primary key to the SalesOrder.EmployeeID foreign key

https://learn.microsoft.com/en-us/training/modules/query-multiple-tables-with-joins/2-join-concepts



Join syntax

ANSI SQL-92

- Tables joined by JOIN operator in FROM clause
  - Preferred syntax

```
SELECT ...
FROM Table1 JOIN Table2
       ON <predicate>;
```

ANSI SQL-89

- Tables listed in FROM clause with join predicate in WHERE clause
  - Not recommended: can lead to accidental Cartesian products!

```
SELECT ...
FROM   Table1, Table2
WHERE  <predicate>;
```

https://learn.microsoft.com/en-us/training/modules/query-multiple-tables-with-joins/2-join-concepts

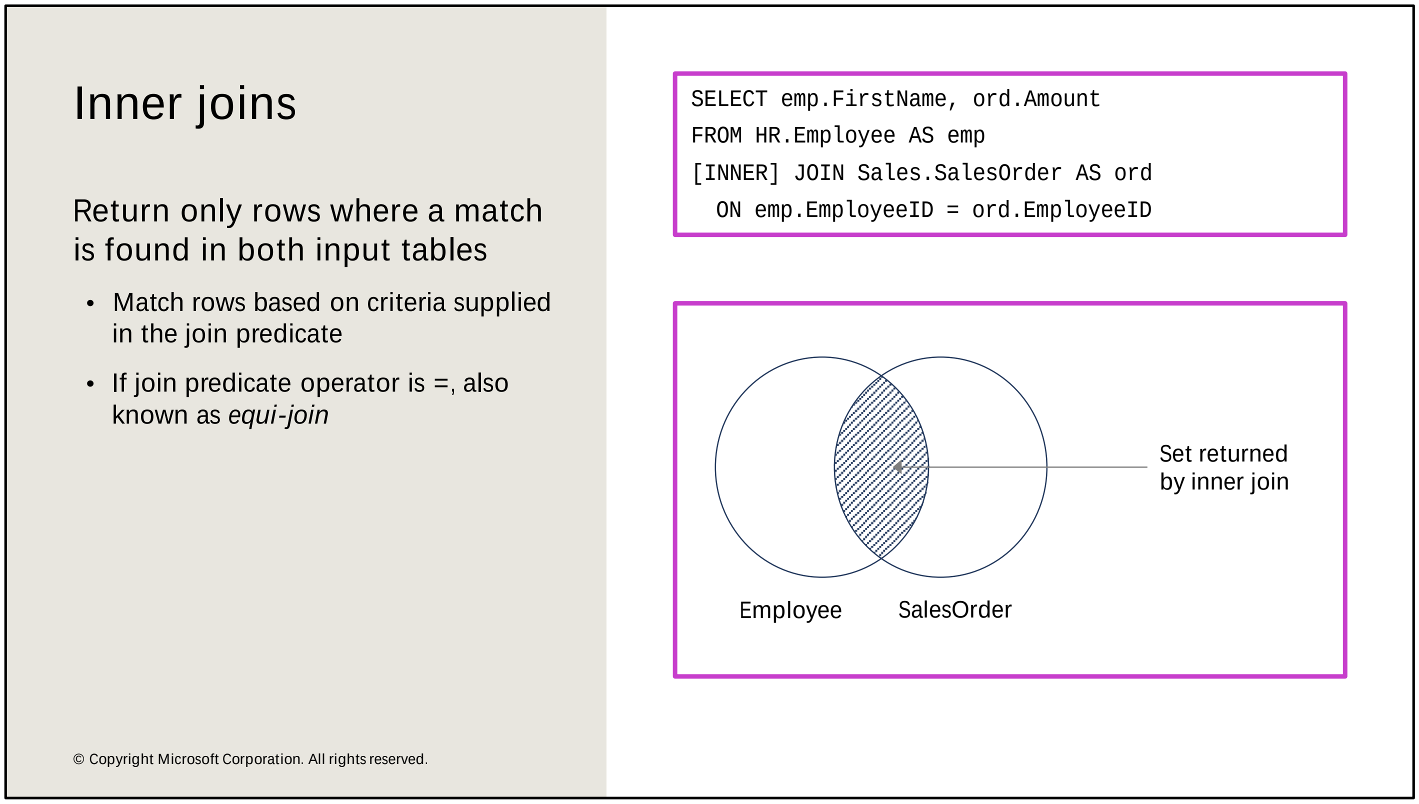
The FROM Clause and Virtual Tables

If you've learned about the logical order of operations that are performed when SQL Server processes a query, you've seen that the FROM clause of a SELECT statement is the first clause to be processed. This clause determines which table or tables will be the source of rows for the query.

The FROM can reference a single table or bring together multiple tables as the source of data for your query. You can think of the FROM clause as creating and populating a virtual table. This virtual table will hold the output of the FROM clause and be used by clauses of the SELECT statement that are applied later, such as the WHERE clause. As you add extra functionality, such as join operators, to a FROM clause, it will be helpful to think of the purpose of the FROM clause elements as either to add rows to, or remove rows from, the virtual table.

The virtual table created by a FROM clause is a logical entity only. In SQL Server, no physical table is created, whether persistent or temporary, to hold the results of the FROM clause, as it is passed to the WHERE clause or other parts of the query.

The virtual table created by the FROM clause contains data from all of the joined tables. It can be useful to think of the results as *sets*, and conceptualize the join results as a Venn diagram.

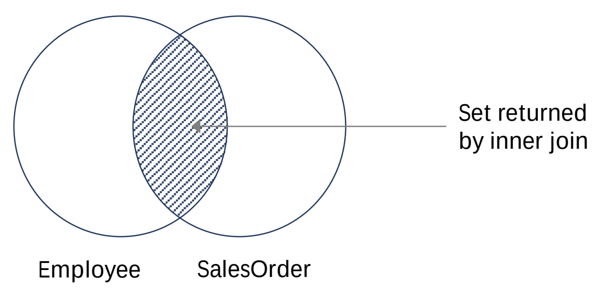


Inner joins

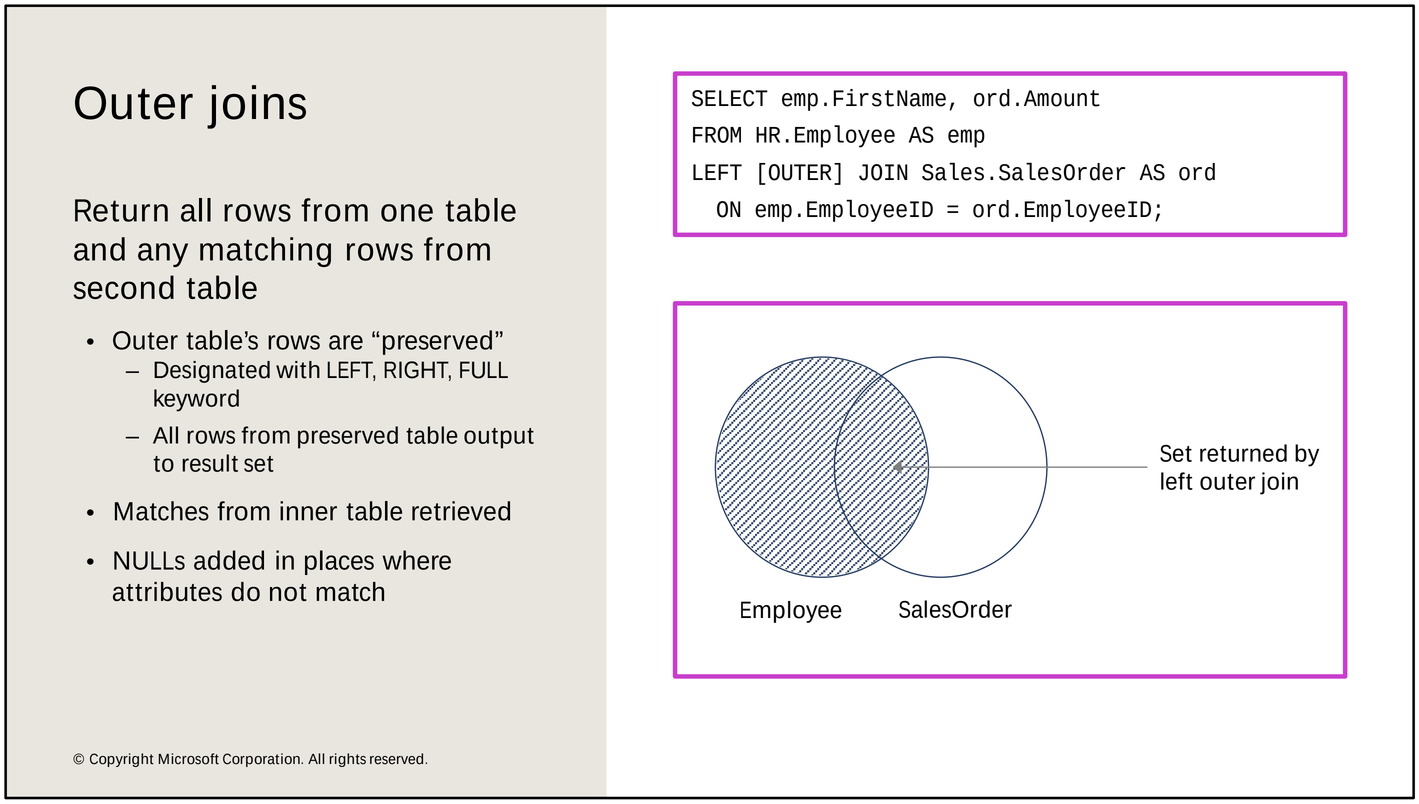
Return only rows where a match is found in both input tables

- Match rows based on criteria supplied in the join predicate
- If join predicate operator is =, also known as *equi-join*

```
SELECT emp.FirstName, ord.Amount
FROM HR.Employee AS emp
[INNER] JOIN Sales.SalesOrder AS ord
  ON emp.EmployeeID = ord.EmployeeID
```



https://learn.microsoft.com/en-us/training/modules/query-multiple-tables-with-joins/3a-inner-joins

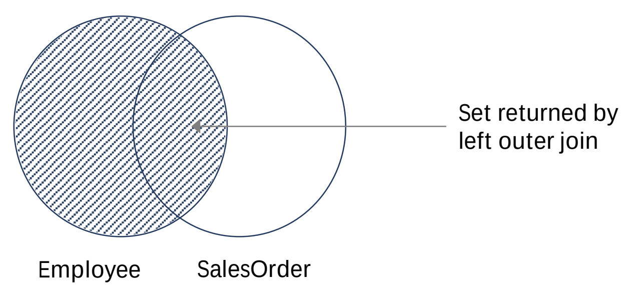


Outer joins

Return all rows from one table and any matching rows from second table

- Outer table's rows are "preserved"
  - Designated with LEFT, RIGHT, FULL keyword
  - All rows from preserved table output to result set
- Matches from inner table retrieved
- NULLs added in places where attributes do not match

```
SELECT emp.FirstName, ord.Amount
FROM HR.Employee AS emp
LEFT [OUTER] JOIN Sales.SalesOrder AS ord
  ON emp.EmployeeID = ord.EmployeeID;
```



https://learn.microsoft.com/en-us/training/modules/query-multiple-tables-with-joins/3b-outer-joins

Cross joins

Combine all rows from both tables

- All possible combinations output
- Logical foundation for inner and outer joins
  - Inner join starts with Cartesian product, adds filter
  - Outer join takes Cartesian output, filtered, adds back non-matching rows (with NULL placeholders)

Cartesian product output is typically undesired

- Some useful exceptions:
  - Table of numbers
  - Generating data for testing

Employee

EmployeeID	FirstName
1	Dan
2	Aisha

Product

ProductID	Name
1	Widget
2	Gizmo

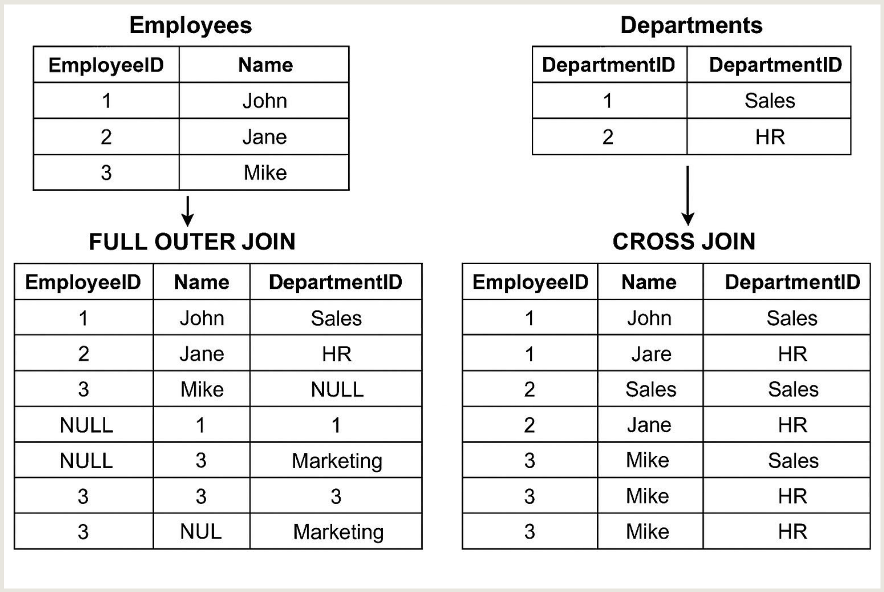
```
SELECT emp.FirstName, prd.Name
FROM HR.Employee AS emp
CROSS JOIN Production.Product AS prd;
```

Result

FirstName	Name
Dan	Widget
Dan	Gizmo
Aisha	Widget
Aisha	Gizmo

https://learn.microsoft.com/en-us/training/modules/query-multiple-tables-with-joins/4-explore-cross-joins

FULL OUTER JOIN vs. CROSS JOIN



Employees

EmployeeID	Name
1	John
2	Jane
3	Mike

Departments

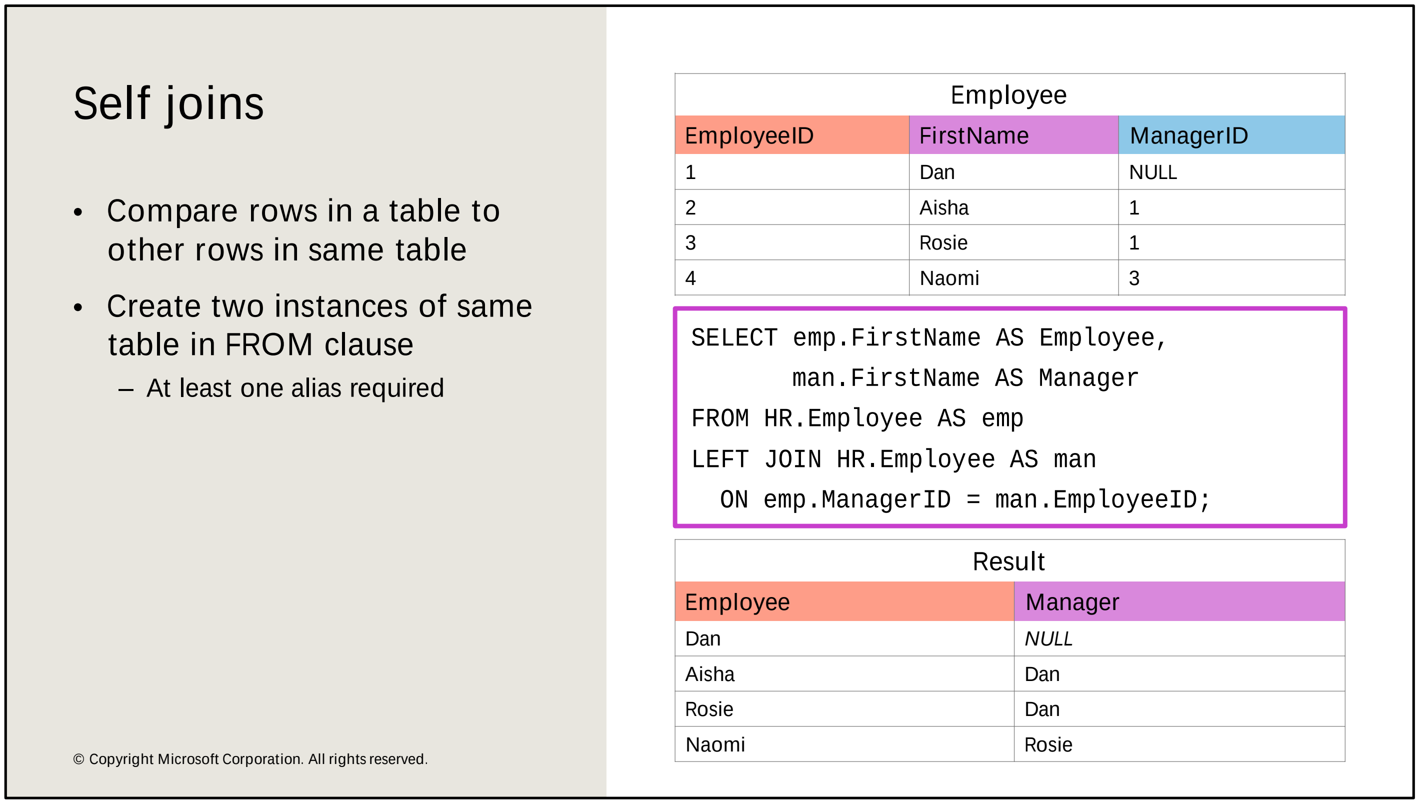
DepartmentID	DepartmentID
1	Sales
2	HR

FULL OUTER JOIN

EmployeeID	Name	DepartmentID
1	John	Sales
2	Jane	HR
3	Mike	NULL
NULL	1	1
NULL	3	Marketing
3	3	3
3	NUL	Marketing

CROSS JOIN

EmployeeID	Name	DepartmentID
1	John	Sales
1	Jare	HR
2	Sales	Sales
2	Jane	HR
3	Mike	Sales
3	Mike	HR
3	Mike	HR



Self joins

- Compare rows in a table to other rows in same table
- Create two instances of same table in FROM clause
  - At least one alias required

Employee

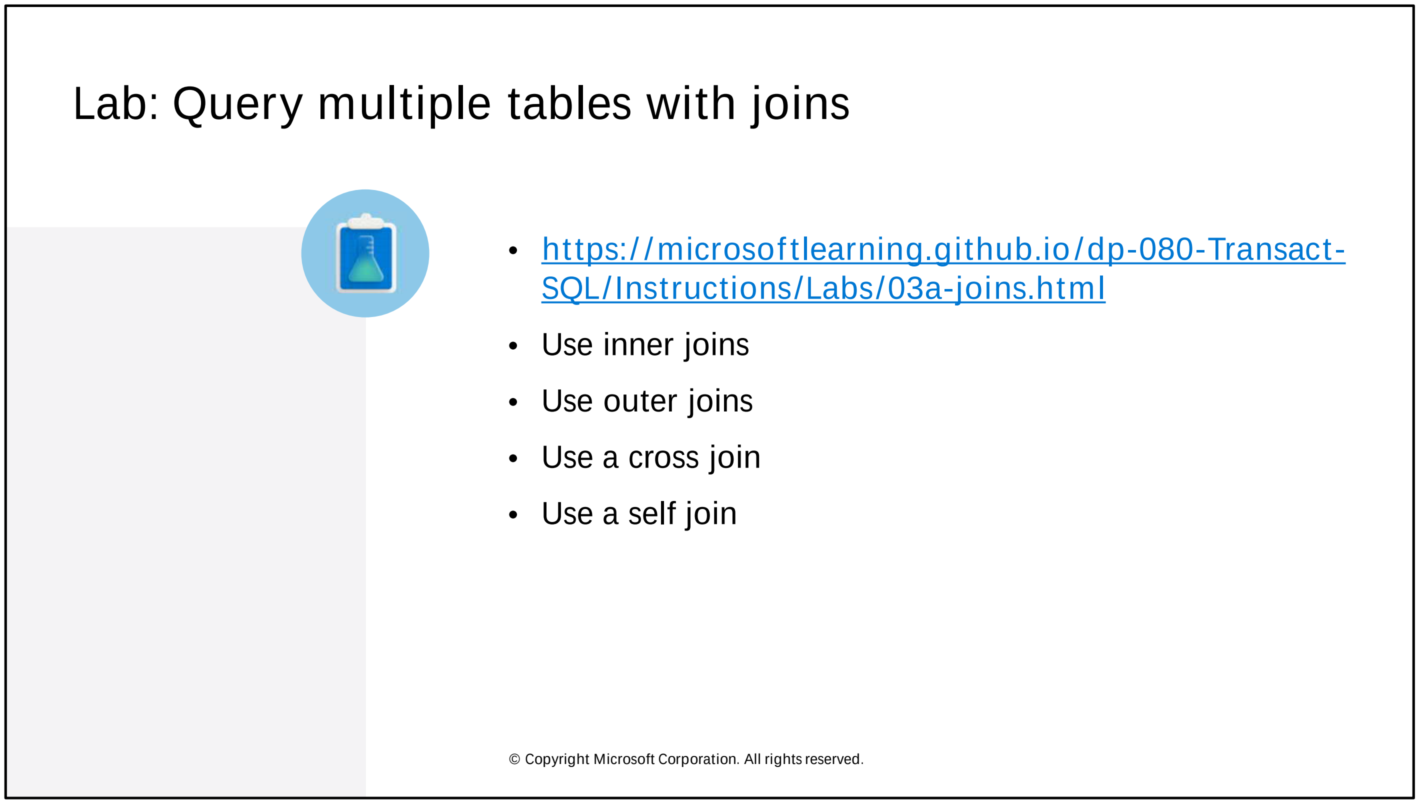
EmployeeID	FirstName	ManagerID
1	Dan	NULL
2	Aisha	1
3	Rosie	1
4	Naomi	3

```
SELECT emp.FirstName AS Employee,
       man.FirstName AS Manager
FROM HR.Employee AS emp
LEFT JOIN HR.Employee AS man
  ON emp.ManagerID = man.EmployeeID;
```

Result

Employee	Manager
Dan	*NULL*
Aisha	Dan
Rosie	Dan
Naomi	Rosie

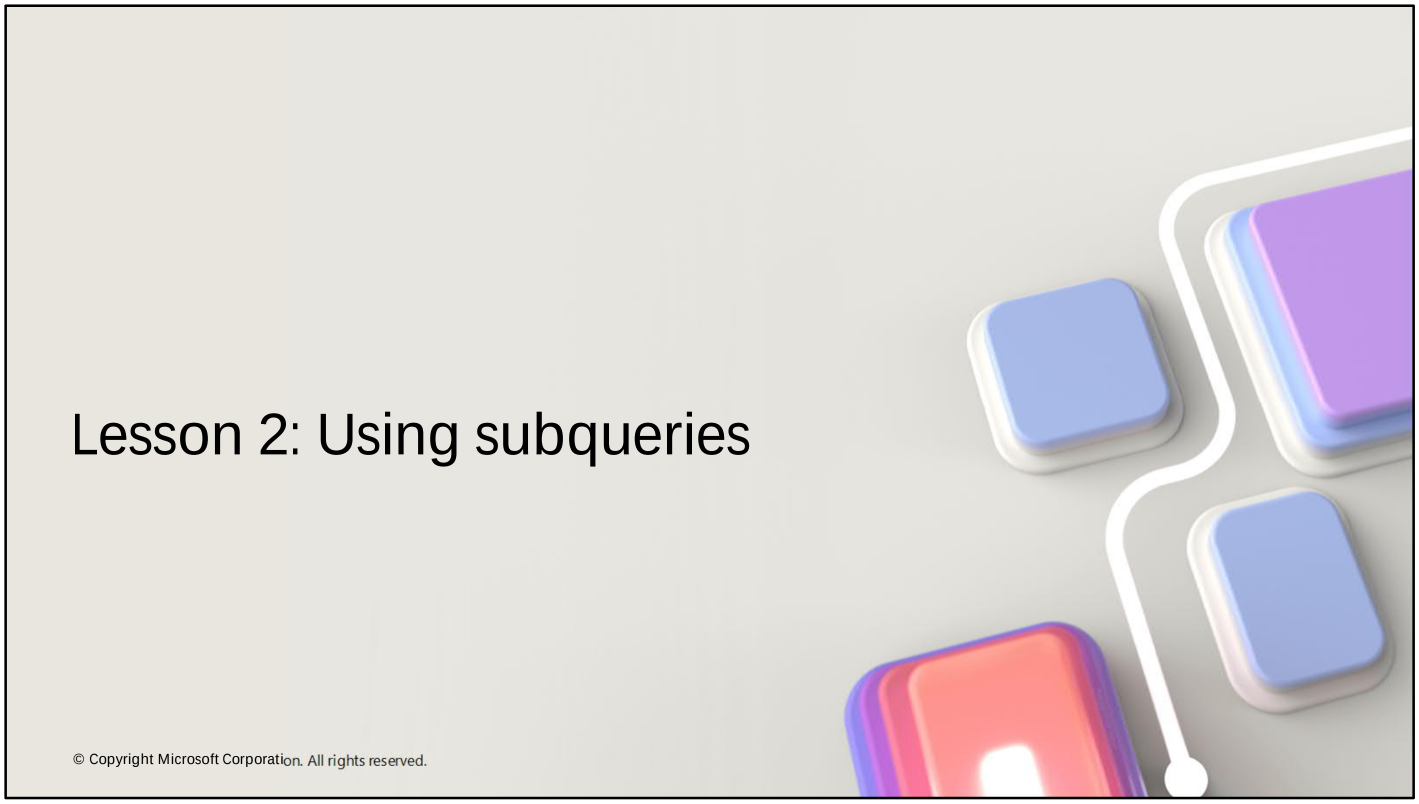
https://learn.microsoft.com/en-us/training/modules/query-multiple-tables-with-joins/5-self-joins



Lab: Query multiple tables with joins

- https://microsoftlearning.github.io/dp-080-Transact-SQL/Instructions/Labs/03a-joins.html
- Use inner joins
- Use outer joins
- Use a cross join
- Use a self join

https://microsoftlearning.github.io/dp-080-Transact-SQL/Instructions/Labs/03a-joins.html



Lesson 2: Using subqueries

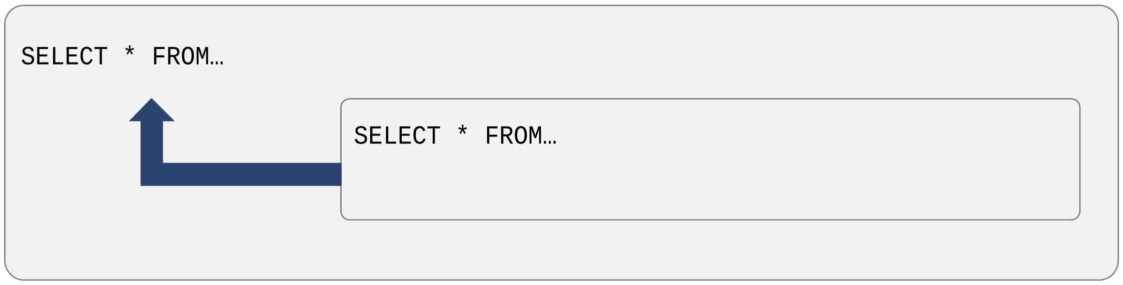
Sometimes, when using Transact-SQL to retrieve data from a database, it can be easier to simplify complex queries by breaking them down into multiple simpler queries that can be combined to achieve the desired results.

Transact-SQL supports the creation of *subqueries*, in which an inner query returns its result to an outer query.

In this module, you will learn how to:

- Understand what subqueries are
- Use scalar or multi-valued subqueries
- Use self-contained or correlated subqueries

Introduction to subqueries



```
SELECT * FROM…
```

```
SELECT * FROM…
```

Subqueries are nested queries: queries within queries

Results of inner query passed to outer query

- Inner query acts like an expression from perspective of the outer query

A subquery is a SELECT statement nested within another query. Being able to nest one query within another will enhance your ability to create effective queries in T-SQL. In general, subqueries are evaluated once, and provide their results to the outer query.

Working with subqueries

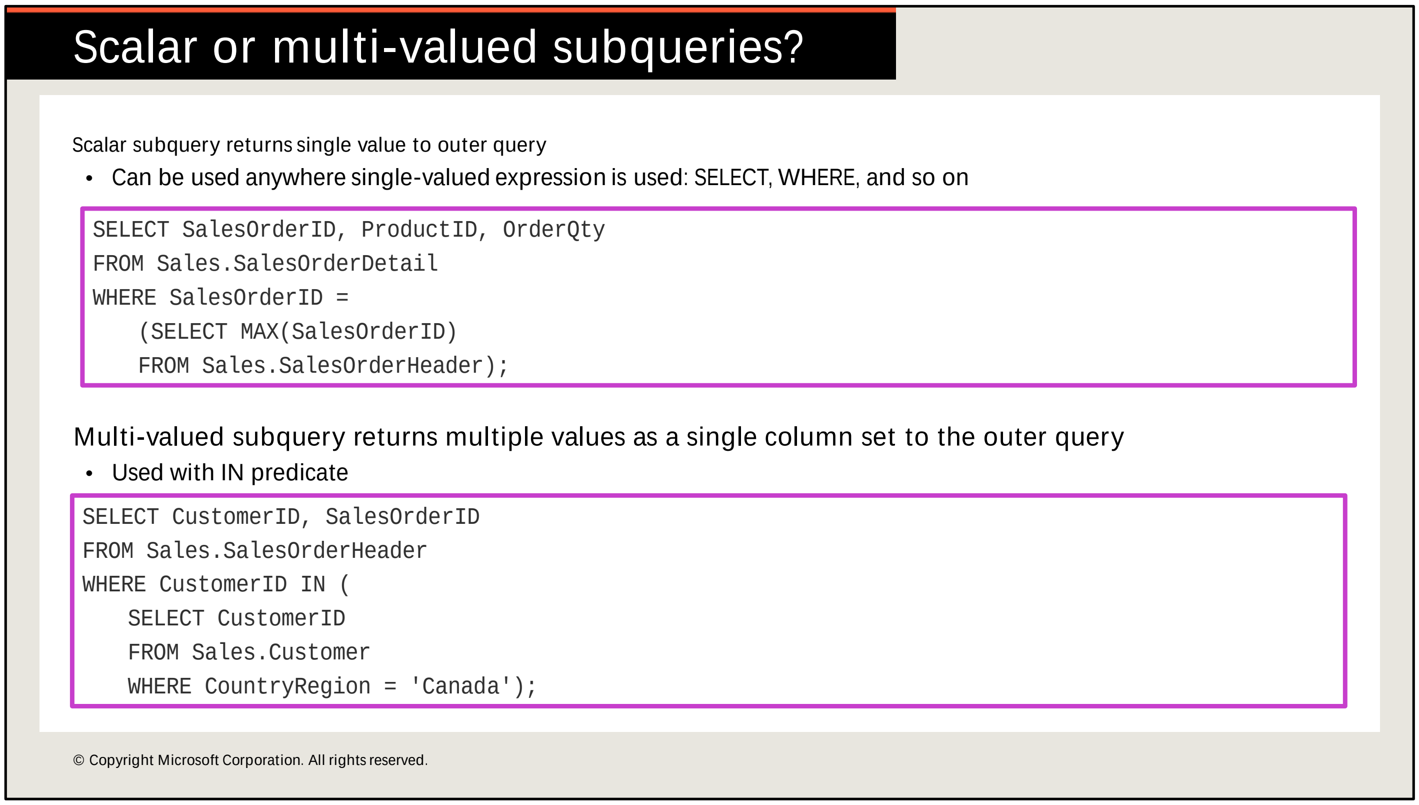
A subquery is a SELECT statement nested, or embedded, within another query. The nested query, which is the subquery, is referred to as the inner query. The query containing the nested query is the outer query.

The purpose of a subquery is to return results to the outer query. The form of the results will determine whether the subquery is a scalar or multi-valued subquery:

- Scalar subqueries return a single value. Outer queries must process a single result.
- Multi-valued subqueries return a result much like a single-column table. Outer queries must be able to process multiple values.

In addition to the choice between scalar and multi-valued subqueries, subqueries can either be self-contained subqueries or they can be correlated with the outer query:

- Self-contained subqueries can be written as stand-alone queries, with no dependencies on the outer query.
    - A self-contained subquery is processed once, when the inner query runs and passes its results to that outer query.
- Correlated subqueries reference one or more columns from the outer query and therefore depend on it.
    - Correlated subqueries cannot be run separately from the outer query.



Scalar or multi-valued subqueries?

Scalar subquery returns single value to outer query

- Can be used anywhere single-valued expression is used: SELECT, WHERE, and so on

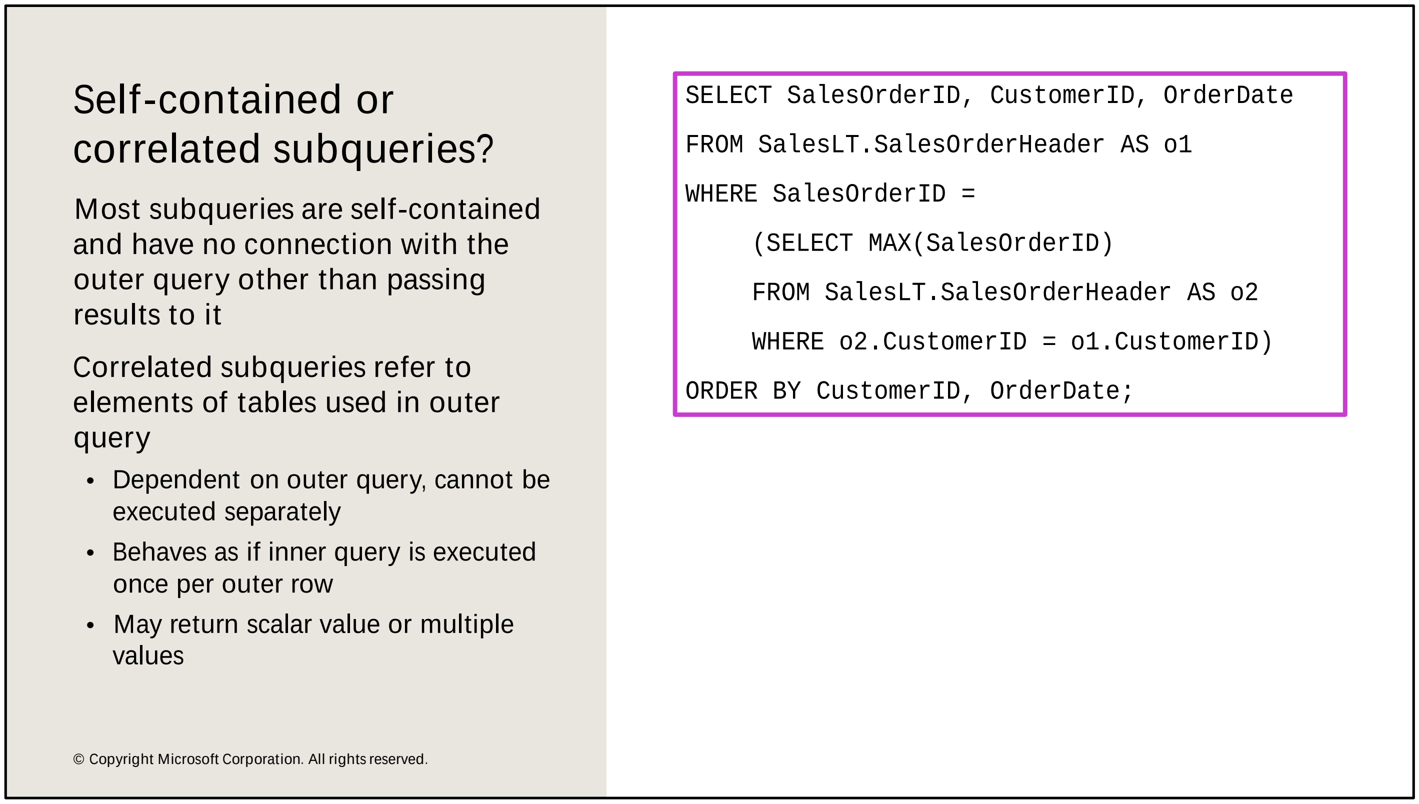
```
SELECT SalesOrderID, ProductID, OrderQty
FROM Sales.SalesOrderDetail
WHERE SalesOrderID =
   (SELECT MAX(SalesOrderID)
   FROM Sales.SalesOrderHeader);
```

Multi-valued subquery returns multiple values as a single column set to the outer query

- Used with IN predicate

```
SELECT CustomerID, SalesOrderID
FROM Sales.SalesOrderHeader
WHERE CustomerID IN (
   SELECT CustomerID
   FROM Sales.Customer
   WHERE CountryRegion = 'Canada');
```

https://learn.microsoft.com/en-us/training/modules/write-subqueries/3-scalar-multi-values-subqueries



Self-contained or correlated subqueries?

Most subqueries are self-contained and have no connection with the outer query other than passing results to it

Correlated subqueries refer to elements of tables used in outer query

- Dependent on outer query, cannot be executed separately
- Behaves as if inner query is executed once per outer row
- May return scalar value or multiple values

```
SELECT SalesOrderID, CustomerID, OrderDate
FROM SalesLT.SalesOrderHeader AS o1
WHERE SalesOrderID =
    (SELECT MAX(SalesOrderID)
    FROM SalesLT.SalesOrderHeader AS o2
    WHERE o2.CustomerID = o1.CustomerID)
ORDER BY CustomerID, OrderDate;
```

https://learn.microsoft.com/en-us/training/modules/write-subqueries/4-self-contained-correlated-subqueries

Writing correlated subqueries

To write correlated subqueries, consider the following guidelines:

- Write the outer query to accept the appropriate return result from the inner query. If the inner query is scalar, you can use equality and comparison operators, such as =, <, >, and <>, in the WHERE clause. If the inner query might return multiple values, use an IN predicate. Plan to handle NULL results.
- Identify the column from the outer query that will be referenced by the correlated subquery. Declare an alias for the table that is the source of the column in the outer query.
- Identify the column from the inner table that will be compared to the column from the outer table. Create an alias for the source table, as you did for the outer query.
- Write the inner query to retrieve values from its source, based on the input value from the outer query. For example, use the outer column in the WHERE clause of the inner query.

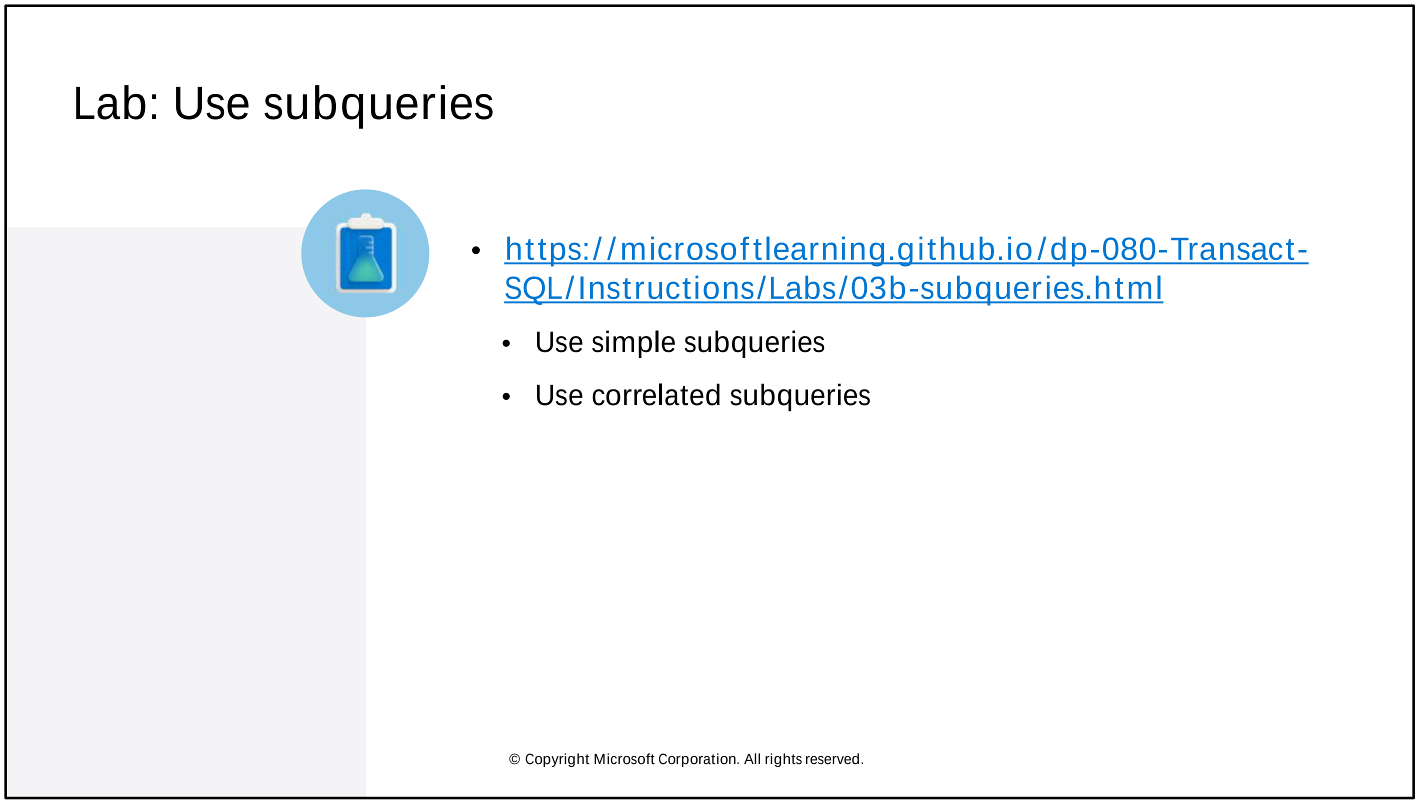
The correlation between the inner and outer queries occurs when the outer value is referenced by the inner query for comparison. It’s this correlation that gives the subquery its name.

Working with EXISTS

In addition to retrieving values from a subquery, T-SQL provides a mechanism for checking whether any results would be returned from a query. The EXISTS predicate determines whether any rows meeting a specified condition exist, but rather than return them, it returns TRUE or FALSE. This technique is useful for validating data without incurring the overhead of retrieving and processing the results.

When a subquery is related to the outer query using the EXISTS predicate, SQL Server handles the results of the subquery in a special way. Rather than retrieve a scalar value or a multi-valued list from the subquery, EXISTS simply checks to see if there are any rows in the result.

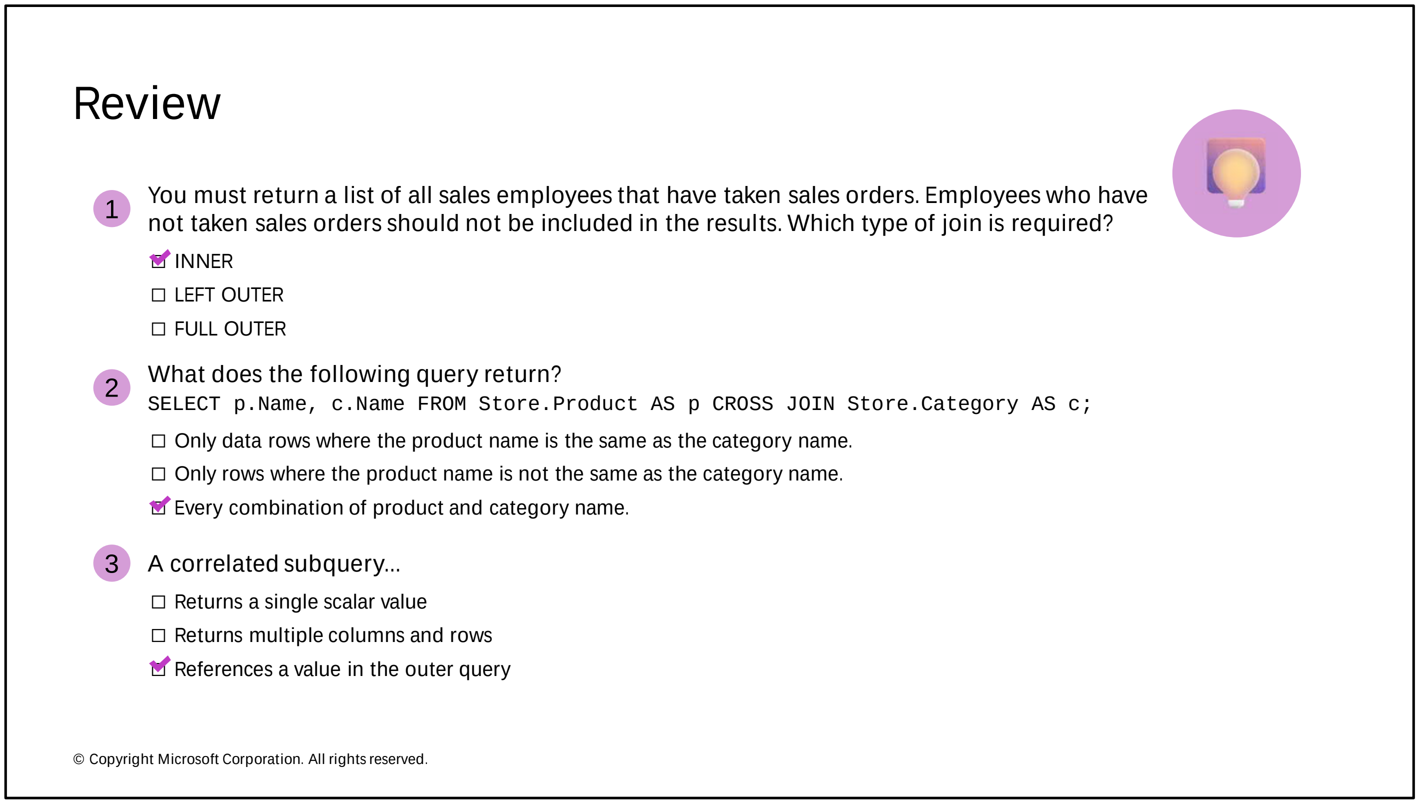
Conceptually, an EXISTS predicate is equivalent to retrieving the results, counting the rows returned, and comparing the count to zero. Compare the following queries, which will return details about customers who have placed orders:



Lab: Use subqueries

- https://microsoftlearning.github.io/dp-080-Transact-SQL/Instructions/Labs/03b-subqueries.html
  - Use simple subqueries
  - Use correlated subqueries

https://microsoftlearning.github.io/dp-080-Transact-SQL/Instructions/Labs/03b-subqueries.html



Review

1. You must return a list of all sales employees that have taken sales orders. Employees who have not taken sales orders should not be included in the results. Which type of join is required?
   - ☑ INNER
   - ☐ LEFT OUTER
   - ☐ FULL OUTER

2. What does the following query return?
   `SELECT p.Name, c.Name FROM Store.Product AS p CROSS JOIN Store.Category AS c;`
   - ☐ Only data rows where the product name is the same as the category name.
   - ☐ Only rows where the product name is not the same as the category name.
   - ☑ Every combination of product and category name.

3. A correlated subquery…
   - ☐ Returns a single scalar value
   - ☐ Returns multiple columns and rows
   - ☑ References a value in the outer query

Use the slide animation to reveal the correct answers.

Microsoft