

Session 4

10/28/2024

Module 4

Querying Multiple Tables

Module Overview

- Understanding Joins
- Querying with Inner Joins
- Querying with Outer Joins
- Querying with Cross Joins and Self Joins

Lesson 1: Understanding Joins

- The FROM Clause and Virtual Tables
- Join Terminology: Cartesian Product
- Overview of Join Types
- T-SQL Syntax Choices
- Demonstration: Understanding Joins

The FROM Clause and Virtual Tables

- FROM clause determines source tables to be used in SELECT statement
- FROM clause can contain tables and operators
- Result set of FROM clause is virtual table
 - Subsequent logical operations in SELECT statement consume this virtual table
- FROM clause can establish table aliases for use by subsequent phases of query

SQL JOINS

- A JOIN clause is used to combine rows from two or more tables, based on a related column between them.

	order_id	product_id	quantity	unit_price
▶	1	4	4	3.74
	2	1	2	9.10
	2	4	4	1.66
	2	6	2	2.94
	3	3	10	9.12
	4	3	7	6.99
	4	10	7	6.40
	5	2	3	9.89
	6	1	4	8.65
	6	2	4	3.28

	product_id	name	quantity_in_stock	unit_price
▶	1	Foam Dinner Plate	70	1.21
	2	Pork - Bacon,back Peameal	49	4.65
	3	Lettuce - Romaine, Heart	38	3.35
	4	Brocolinni - Gaylan, Chinese	90	4.53
	5	Sauce - Ranch Dressing	94	1.63
	6	Petit Baguette	14	2.39
	7	Sweet Pea Sprouts	98	3.29
	8	Island Oasis - Raspberry	26	0.74
	9	Longan	67	2.26
	10	Broom - Push	6	1.09

	order_id	name	quantity_in_stock
▶	1	Brocolinni - Gaylan, Chinese	90
	2	Foam Dinner Plate	70
	2	Brocolinni - Gaylan, Chinese	90
	2	Petit Baguette	14
	3	Lettuce - Romaine, Heart	38
	4	Lettuce - Romaine, Heart	38
	4	Broom - Push	6
	5	Pork - Bacon,back Peameal	49
	6	Foam Dinner Plate	70
	6	Pork - Bacon,back Peameal	49
	6	Lettuce - Romaine, Heart	38
	6	Sauce - Ranch Dressing	94

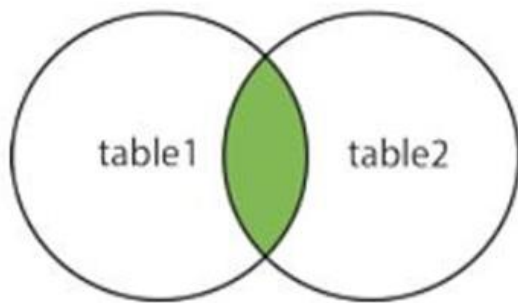
JOIN Types

Different Types of SQL JOINS

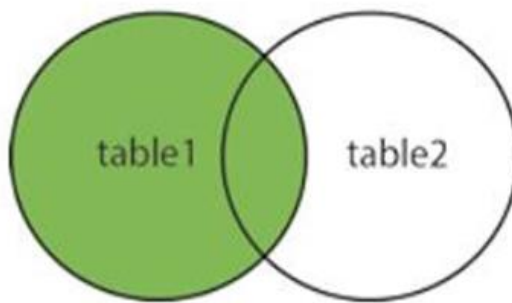
Here are the different types of the JOINS in SQL:

- **(INNER) JOIN** : Returns records that have matching values in both tables
- **LEFT (OUTER) JOIN** : Returns all records from the left table, and the matched records from the right table
- **RIGHT (OUTER) JOIN** : Returns all records from the right table, and the matched records from the left table
- **FULL (OUTER) JOIN** : Returns all records when there is a match in either left or right table

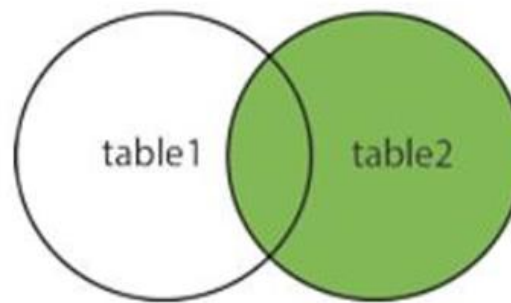
INNER JOIN



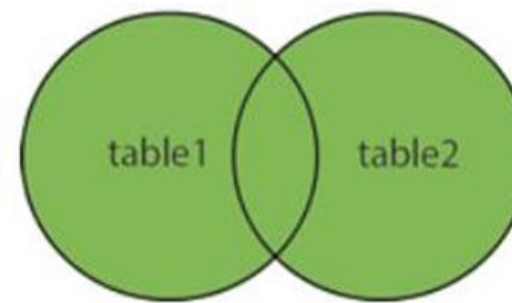
LEFT JOIN



RIGHT JOIN



FULL OUTER JOIN



Join Terminology: Cartesian Product

- Characteristics of a Cartesian product
 - Output or intermediate result of FROM clause
 - Combine all possible combinations of two sets
 - In T-SQL queries, usually not desired
 - Special case: table of numbers

Name
Davis
Funk
King



Product
Alice Mutton
Crab Meat
Ipoh Coffee



Name	Product
Davis	Alice Mutton
Davis	Crab Meat
Davis	Ipoh Coffee
Funk	Alice Mutton
Funk	Crab Meat
Funk	Ipoh Coffee
King	Alice Mutton
King	Crab Meat
King	Ipoh Coffee

Overview of Join Types

- Join types in FROM clauses specify the operations performed on the virtual table:

Join Type	Description
Cross	Combines all rows in both tables (creates Cartesian product)
Inner	Starts with Cartesian product; applies filter to match rows between tables based on predicate
Outer	Starts with Cartesian product; all rows from designated table preserved, matching rows from other table retrieved. Additional NULLs inserted as placeholders

T-SQL Syntax Choices

- ANSI SQL-92
 - Tables joined by JOIN operator in FROM Clause

```
SELECT ...  
FROM Table1 JOIN Table2  
ON <on_predicate>
```

- ANSI SQL-89
 - Tables joined by commas in FROM Clause
 - Not recommended: accidental Cartesian products!

```
SELECT ...  
FROM Table1, Table2  
WHERE <where_predicate>
```

Demonstration: Understanding Joins

In this demonstration, you will see how to:

- Use joins



Lesson 2: Querying with Inner Joins

- Understanding Inner Joins
- Inner Join Syntax
- Inner Join Examples
- Demonstration: Querying with Inner Joins

Understanding Inner Joins

- Returns only rows where a match is found in both input tables
- Matches rows based on attributes supplied in predicate
 - ON clause in SQL-92 syntax (preferred)
 - WHERE clause in SQL-89 syntax
- Why filter in ON clause?
 - Logical separation between filtering for purposes of join and filtering results in WHERE
 - Typically no difference to query optimizer
- If join predicate operator is =, also known as equi-join

Inner Join Syntax

- List tables in FROM Clause separated by JOIN operator
- Table aliases preferred
- Table order does not matter

```
FROM t1 JOIN t2  
ON t1.column = t2.column
```

```
SELECT o.orderid,  
       o.orderdate,  
       od.productid,  
       od.unitprice,  
       od.qty  
FROM Sales.Orders AS o  
JOIN Sales.OrderDetails AS od  
ON o.orderid = od.orderid;
```


Inner Join Examples

- Join based on single matching attribute

```
SELECT ...  
FROM Production.Categories AS C  
JOIN Production.Products AS P  
ON C.categoryid = P.categoryid;
```

- Join based on multiple matching attributes
(composite join)

```
-- List cities and countries where both --  
-- customers and employees live  
SELECT DISTINCT e.city, e.country  
FROM Sales.Customers AS c  
JOIN HR.Employees AS e  
ON c.city = e.city AND  
c.country = e.country;
```

INNER JOIN 3 tables example

order_id	customer_id	order_date	status	comments	shipped_date	shipper_id
1	6	2019-01-30	1	NULL	NULL	NULL
2	7	2018-08-02	2	NULL	2018-08-03	4
3	8	2017-12-01	1	NULL	NULL	NULL
4	2	2017-01-22	1	NULL	NULL	NULL
5	5	2017-08-25	2		2017-08-26	3
6	10	2018-11-18	1	Aliquam erat volutpat. In congue.	NULL	NULL
7	2	2018-09-22	2	NULL	2018-09-23	4
8	5	2018-06-08	1	Mauris enim leo, rhoncus sed, vestibulum sit am...	NULL	NULL
9	10	2017-07-05	2	Nulla mollis molestie lorem. Quisque ut erat.	2017-07-06	1
10	6	2018-04-22	2	NULL	2018-04-23	2

customer_id	first_name	last_name	birth_date	phone	address	city	state	points
1	Babara	MacCaffrey	1986-03-28	781-932-9754	0 Sage Terrace	Waltham	MA	2273
2	Ines	Brushfield	1986-04-13	804-427-9456	14187 Commercial Trail	Hampton	VA	947
3	Freddi	Boagey	1985-02-07	719-724-7869	251 Springs Junction	Colorado Springs	CO	2967
4	Ambur	Roseburgh	1974-04-14	407-231-8017	30 Arapahoe Terrace	Orlando	FL	457
5	Clemmie	Betchley	1973-11-07	NULL	5 Spohn Circle	Arlington	TX	3675
6	Elka	Twiddell	1991-09-04	312-480-8498	7 Manley Drive	Chicago	IL	3073
7	Ilene	Dowson	1964-08-30	615-641-4759	50 Lillian Crossing	Nashville	TN	1672
8	Thacher	Naseby	1993-07-17	941-527-3977	538 Mosinee Center	Sarasota	FL	205
9	Romola	Rumgay	1992-05-23	559-181-3744	3520 Ohio Trail	Visalia	CA	1486
10	Levy	Mynett	1969-10-13	404-246-3370	68 Lawn Avenue	Atlanta	GA	796

shipper_id	name
1	Hettinger LLC
2	Schinner-Predovic
3	Satterfield LLC
4	Mraz, Renner and Nolan
5	Waters, Mayert and Prohaska

```

SELECT o.order_id, CONCAT_WS(" ",c.first_name, c.last_name) AS customer_name, s.name AS shipper_name
FROM ((orders o
INNER JOIN customers c ON o.customer_id = c.customer_id)
INNER JOIN shippers s ON o.shipper_id = s.shipper_id)
ORDER BY order_id;

```

INNER JOIN 3 tables example

order_id	customer_id	order_date	status	comments	shipped_date	shipper_id
1	6	2019-01-30	1	NULL	NULL	NULL
2	7	2018-08-02	2	NULL	2018-08-03	4
3	8	2017-12-01	1	NULL	NULL	NULL
4	2	2017-01-22	1	NULL	NULL	NULL
5	5	2017-08-25	2		2017-08-26	3
6	10	2018-11-18	1	Aliquam erat volutpat. In congue.	NULL	NULL
7	2	2018-09-22	2	NULL	2018-09-23	4
8	5	2018-06-08	1	Mauris enim leo, rhoncus sed, vestibulum sit am...	NULL	NULL
9	10	2017-07-05	2	Nulla mollis molestie lorem. Quisque ut erat.	2017-07-06	1
10	6	2018-04-22	2	NULL	2018-04-23	2

customer_id	first_name	last_name	birth_date	phone	address	city	state	points
1	Babara	MacCaffrey	1986-03-28	781-932-9754	0 Sage Terrace	Waltham	MA	2273
2	Ines	Brushfield	1986-04-13	804-427-9456	14187 Commercial Trail	Hampton	VA	947
3	Freddi	Boagey	1985-02-07	719-724-7869	251 Springs Junction	Colorado Springs	CO	2967
4	Ambur	Roseburgh	1974-04-14	407-231-8017	30 Arapahoe Terrace	Orlando	FL	457
5	Clemmie	Betchley	1973-11-07	NULL	5 Spohn Circle	Arlington	TX	3675
6	Elka	Twiddell	1991-09-04	312-480-8498	7 Manley Drive	Chicago	IL	3073
7	Ilene	Dowson	1964-08-30	615-641-4759	50 Lillian Crossing	Nashville	TN	1672
8	Thacher	Naseby	1993-07-17	941-527-3977	538 Mosinee Center	Sarasota	FL	205
9	Romola	Rumgay	1992-05-23	559-181-3744	3520 Ohio Trail	Visalia	CA	1486
10	Levy	Mynett	1969-10-13	404-246-3370	68 Lawn Avenue	Atlanta	GA	796

shipper_id	name
1	Hettinger LLC
2	Schinner-Predovic
3	Satterfield LLC
4	Mraz, Renner and Nolan
5	Waters, Mayert and Prohaska
NULL	NULL



order_id	customer_name	shipper_name
2	Ilene Dowson	Mraz, Renner and Nolan
5	Clemmie Betchley	Satterfield LLC
7	Ines Brushfield	Mraz, Renner and Nolan
9	Levy Mynett	Hettinger LLC
10	Elka Twiddell	Schinner-Predovic

Demonstration: Querying with Inner Joins

In this demonstration, you will see how to:

- Use inner joins



Lesson 3: Querying with Outer Joins

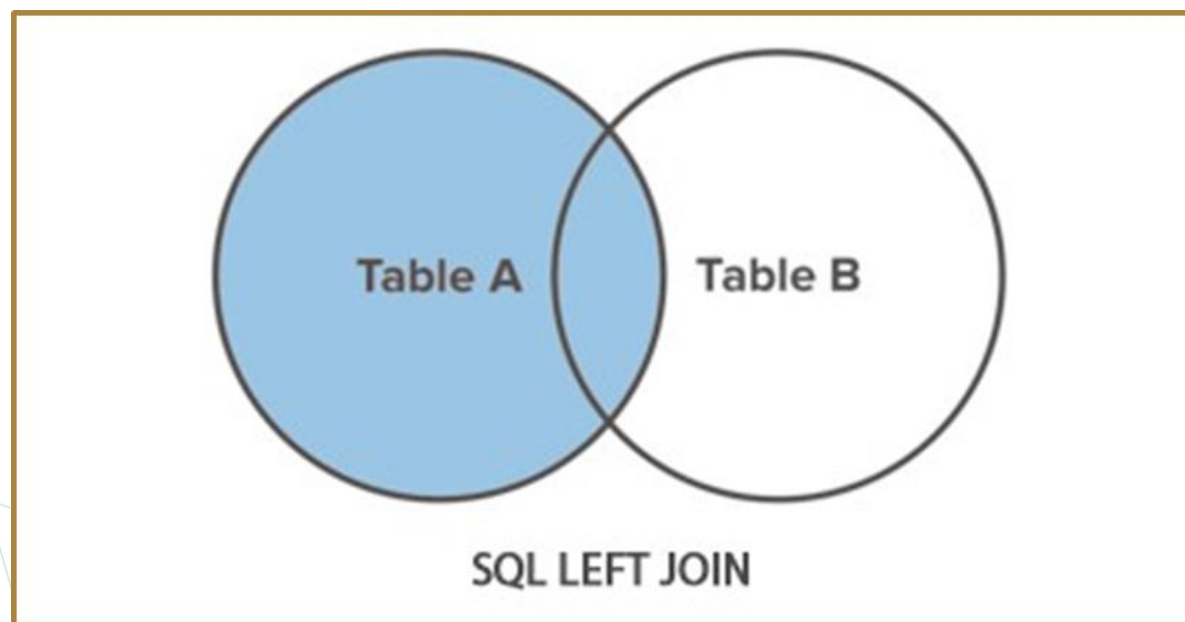
- Understanding Outer Joins
- Outer Join Syntax
- Outer Join Examples
- Demonstration: Querying with Outer Joins

Understanding Outer Joins

- Returns all rows from one table and any matching rows from second table
- One table's rows are "preserved"
 - Designated with LEFT, RIGHT, FULL keyword
 - All rows from preserved table output to result set
- Matches from other table retrieved
- Additional rows added to results for nonmatched rows
 - NULLs added in places where attributes do not match
- Example: return all customers and, for those who have placed orders, return order information; customers without matching orders will display NULL for order details

LEFT JOIN

- It returns all records from the left table (table1), and the matching records from the right table (table2).
- The result is 0 records from the right side, if there is no match.



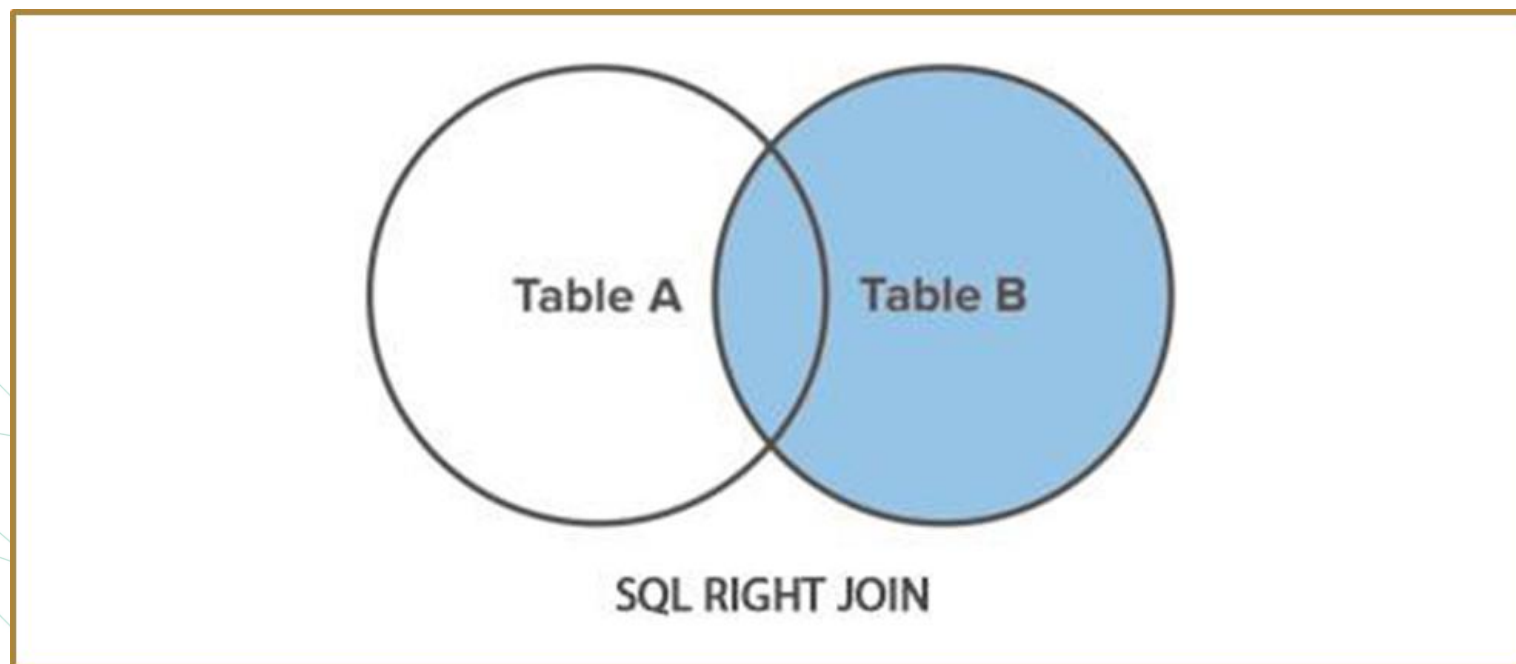
Left join Example

```
SELECT CONCAT(c.first_name, " ", c.last_name) AS customer_name, o.order_id, o.order_date
FROM customers c
LEFT JOIN orders o ON c.customer_id = o.customer_id
ORDER BY customer_name;
```

	customer_name	order_id	order_date
▶	Ambur Roseburgh	NULL	NULL
	Babara MacCaffrey	NULL	NULL
	Clemmie Betchley	5	2017-08-25
	Clemmie Betchley	8	2018-06-08
	Elka Twiddell	1	2019-01-30
	Elka Twiddell	10	2018-04-22
	Freddi Boagey	NULL	NULL
	Ilene Dowson	2	2018-08-02
	Ines Brushfield	4	2017-01-22
	Ines Brushfield	7	2018-09-22
	Levy Mynett	6	2018-11-18
	Levy Mynett	9	2017-07-05
	Romola Rungay	NULL	NULL
	Thacher Naseby	3	2017-12-01

RIGHT JOIN

- It returns all records from the right table (table2), and the matching records from the left table (table1).
- The result is 0 records from the left side, if there is no match.



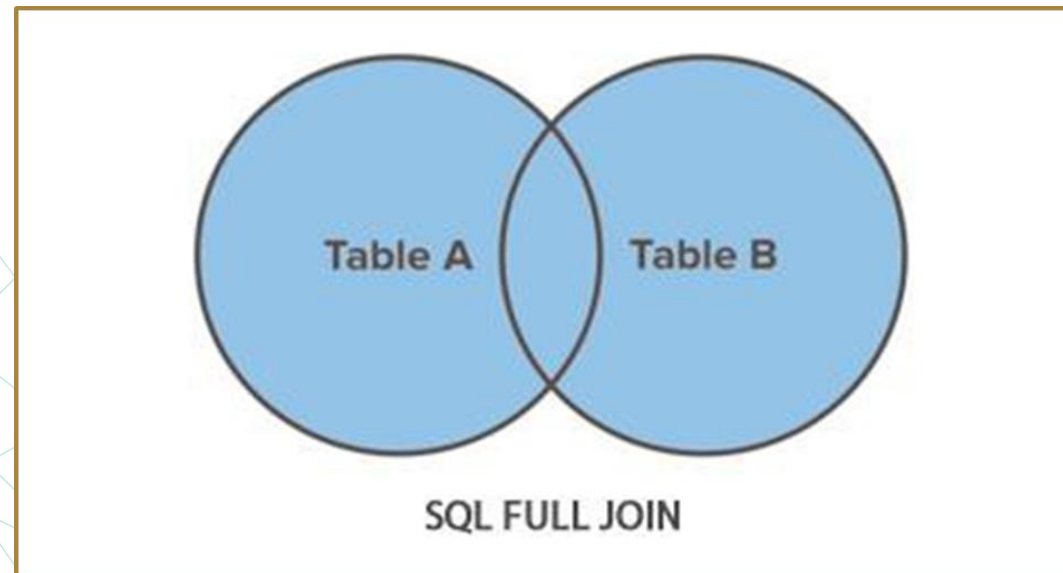
Right Join Example

```
SELECT o.order_id, o.order_date, sh.name
FROM orders o
RIGHT JOIN shippers sh ON o.shipper_id = sh.shipper_id
ORDER BY o.order_id;
```

	order_id	order_date	name
▶	NULL	NULL	Waters, Mayert and Prohaska
	2	2018-08-02	Mraz, Renner and Nolan
	5	2017-08-25	Satterfield LLC
	7	2018-09-22	Mraz, Renner and Nolan
	9	2017-07-05	Hettinger LLC
	10	2018-04-22	Schinner-Predovic

Full Join

- It returns all records when there is a match in left (table1) or right (table2) table records.
- It's not supported by MySQL but supported by SQL server.
- IN MySQL we can use UNIONs to emulate the functionality of FULL OUTER JOIN.
- We should use UNION ALL to avoid removing duplicates.



UNION

- It JOINS tables by matching every row from one table with every row from another table. If there are X rows in the first table, and Y rows in the second table, the result set will have exactly X times Y rows.

The **UNION** operator is used to combine the result-set of two or more **SELECT** statements.

- Every **SELECT** statement within **UNION** must have the same number of columns
- The columns must also have similar data types
- The columns in every **SELECT** statement must also be in the same order

The **UNION** operator selects only distinct values by default. To allow duplicate values, use **UNION ALL** :

Full Join Example

```
SELECT Customers.CustomerName, Orders.OrderID
FROM Customers
FULL OUTER JOIN Orders ON Customers.CustomerID=Orders.CustomerID
ORDER BY Customers.CustomerName;
```

CustomerName	OrderID
<i>Null</i>	10309
<i>Null</i>	10310
Alfreds Futterkiste	<i>Null</i>
Ana Trujillo Emparedados y helados	10308
Antonio Moreno Taquería	Null

Demonstration: Querying with Outer Joins

In this demonstration, you will see how to:

- Use outer joins

Lesson 4: Querying with Cross Joins and Self Joins

- Understanding Cross Joins
- Cross Join Syntax
- Cross Join Examples
- Understanding Self Joins
- Self Join Examples
- Demonstration: Querying with Cross Joins and Self Joins

Understanding Cross Joins

- Combine each row from first table with each row from second table
- 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 nonmatching rows (with NULL placeholders)
- Due to Cartesian product output, not typically a desired form of join
- Some useful exceptions:
 - Table of numbers, generating data for testing

Cross Join Syntax

- No matching performed, no ON clause used
- Return all rows from left table combined with each row from right table (ANSI SQL-92 syntax):

```
SELECT ...  
FROM t1 CROSS JOIN t2
```

- Return all rows from left table combined with each row from right table (ANSI SQL-89 syntax):

```
SELECT ...  
FROM t1, t2
```

Cross Join Example

id	first_name
1	Ahmed
2	Moaz
3	Esraa

color_id	color_name
1	red
2	green
3	blue

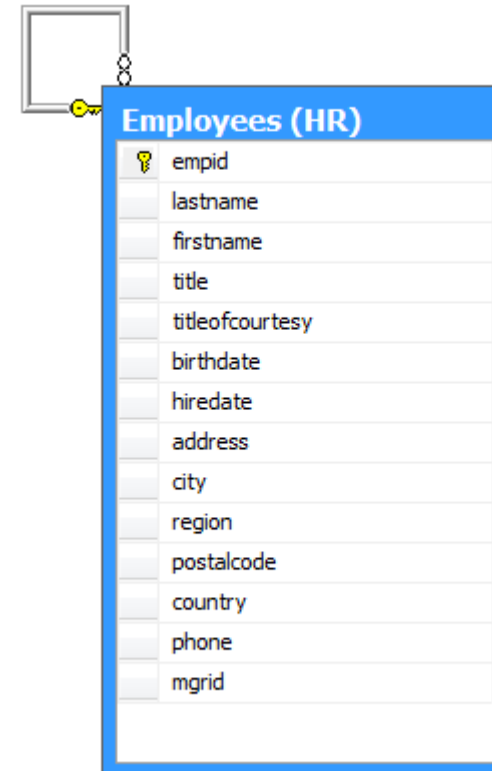
```
SELECT id,  
first_name,  
color_id,  
color_name  
FROM persons  
CROSS JOIN colors;
```



id	name	color_id	color_name
1	Ahmed	1	red
2	Moaz	1	red
3	Esraa	1	red
1	Ahmed	2	green
2	Moaz	2	green
3	Esraa	2	green
1	Ahmed	3	blue
2	Moaz	3	blue
3	Esraa	3	blue

Understanding Self Joins

- Why use self joins?
 - Compare rows in same table to each other
- Create two instances of same table in FROM clause
 - At least one alias required
- Example: Return all employees and the name of the employee's manager



empid
lastname
firstname
title
titleofcourtesy
birthdate
hiredate
address
city
region
postalcode
country
phone
mgrid

Self Join Examples

- Return all employees with ID of employee's manager when a manager exists (inner join):

```
SELECT e.empid, e.lastname,  
       e.title, e.mgrid, m.lastname  
FROM   HR.Employees AS e  
JOIN   HR.Employees AS m  
ON     e.mgrid=m.empid;
```

- Return all employees with ID of manager (outer join). This will return NULL for the CEO:

```
SELECT e. empid, e.lastname,  
       e.title, m.mgrid  
FROM   HR.Employees AS e  
LEFT OUTER JOIN HR.Employees AS m  
ON     e.mgrid=m.empid;
```

Demonstration: Querying with Cross Joins and Self Joins

In this demonstration, you will see how to:

- Use self joins and cross joins



Lab: Querying Multiple Tables

- Exercise 1: Writing Queries That Use Inner Joins
- Exercise 2: Writing Queries That Use Multiple-Table Inner Joins
- Exercise 3: Writing Queries That Use Self Joins
- Exercise 4: Writing Queries That Use Outer Joins
- Exercise 5: Writing Queries That Use Cross Joins

Lab Scenario

You are an Adventure Works business analyst who will be writing reports using corporate databases stored in SQL Server. You have been given a set of business requirements for data and you will write T-SQL queries to retrieve the specified data from the databases. You notice that the data is stored in separate tables, so you will need to write queries using various join operations.

Module Review and Takeaways

- Review Question(s)
- Best Practice

