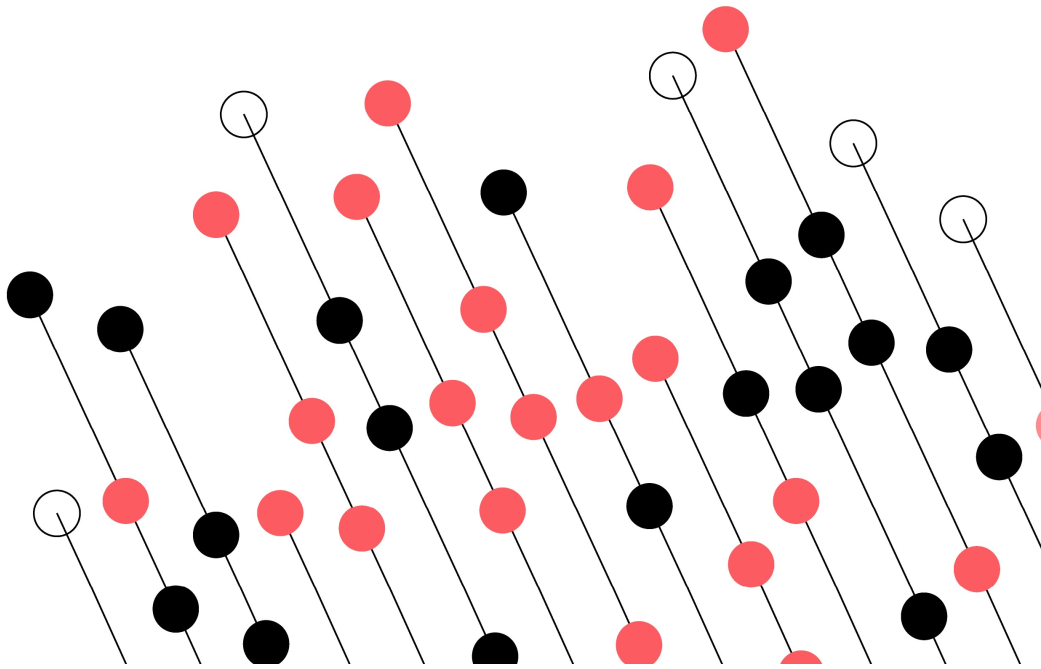
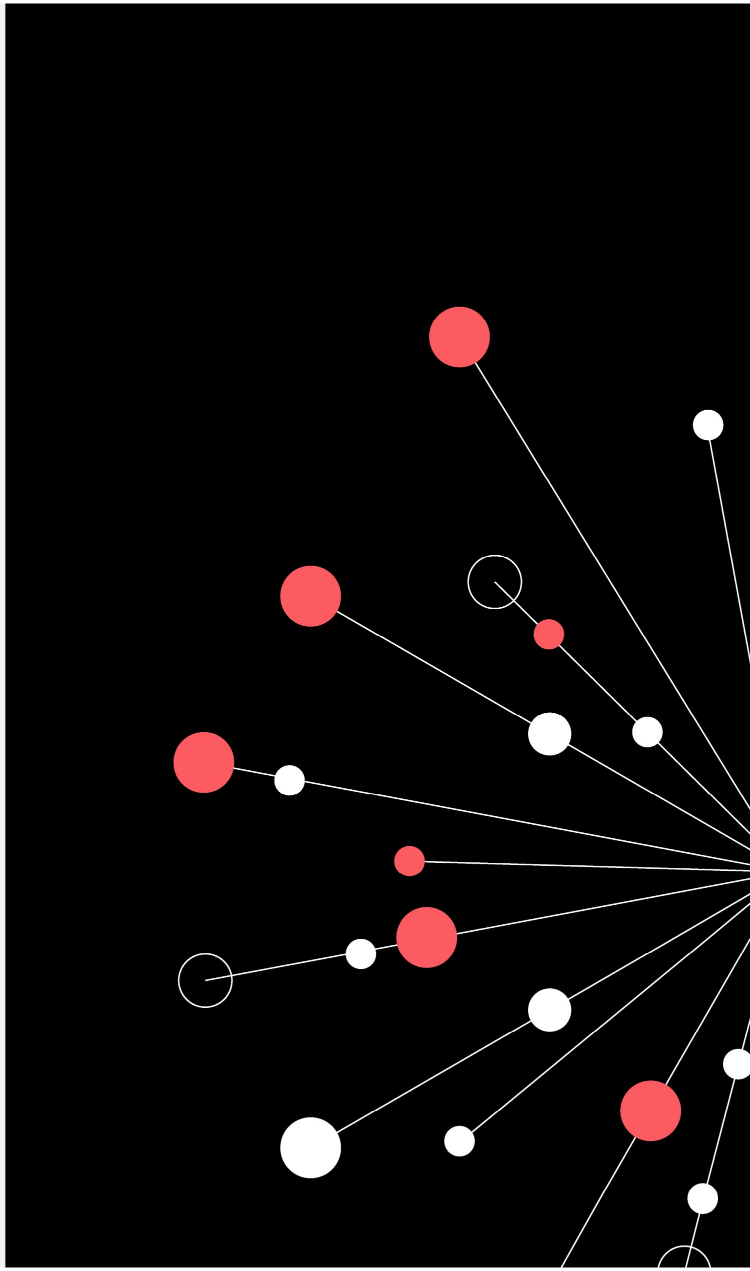
SQL: Joins



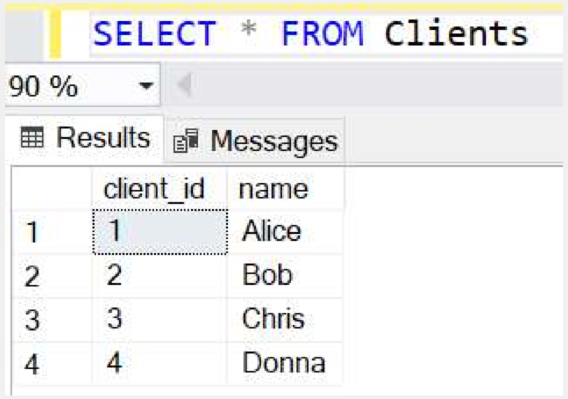
SQL

Contents



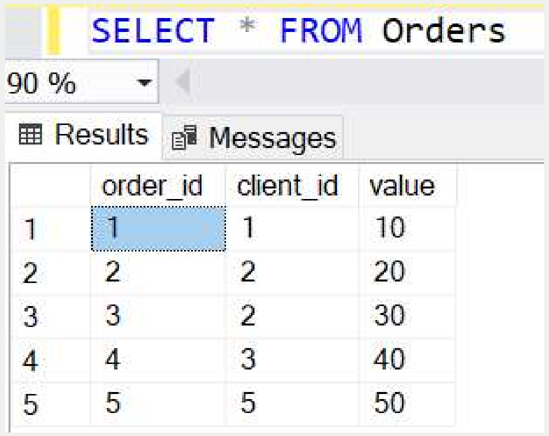
Using multiple tables: set the scene

- Create a database called **ClientsOrders.**
- Import the provided CSV files **Clients.csv** and **Orders.csv** into **ClientsOrders**.
- Check the contents of the two resulting database tables.



```
SELECT * FROM Clients
```

	client_id	name
1	1	Alice
2	2	Bob
3	3	Chris
4	4	Donna



```
SELECT * FROM Orders
```

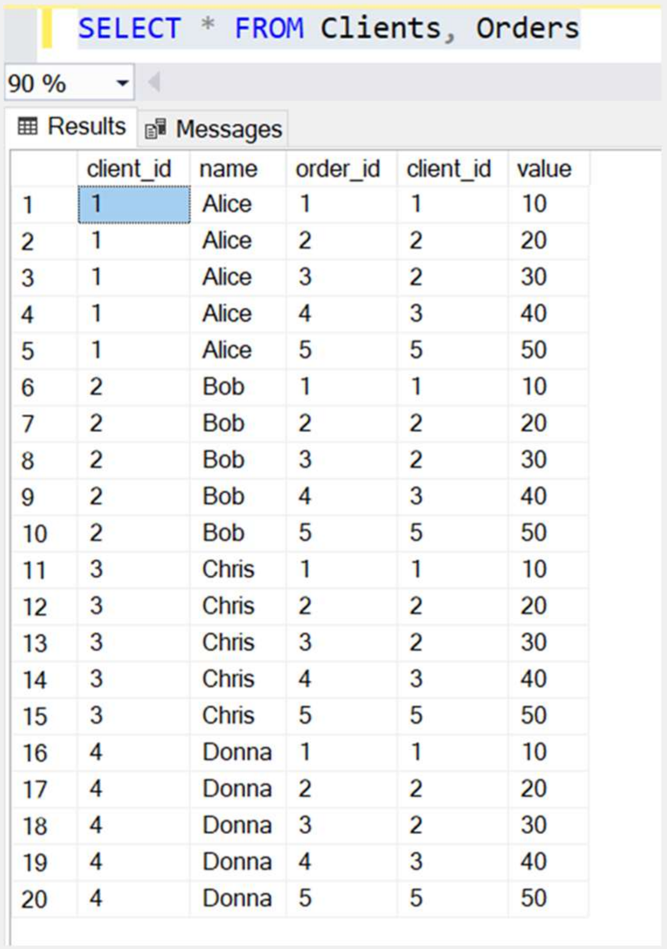
	order_id	client_id	value
1	1	1	10
2	2	2	20
3	3	2	30
4	4	3	40
5	5	5	50

- **Are the records in each table uniquely identified? What are their primary keys?**
- **Are the two tables linked and if yes, how?**
- **Do you see any problems with the database?**

Accessing all data from both tables

What are we getting?

Which of the records make sense?



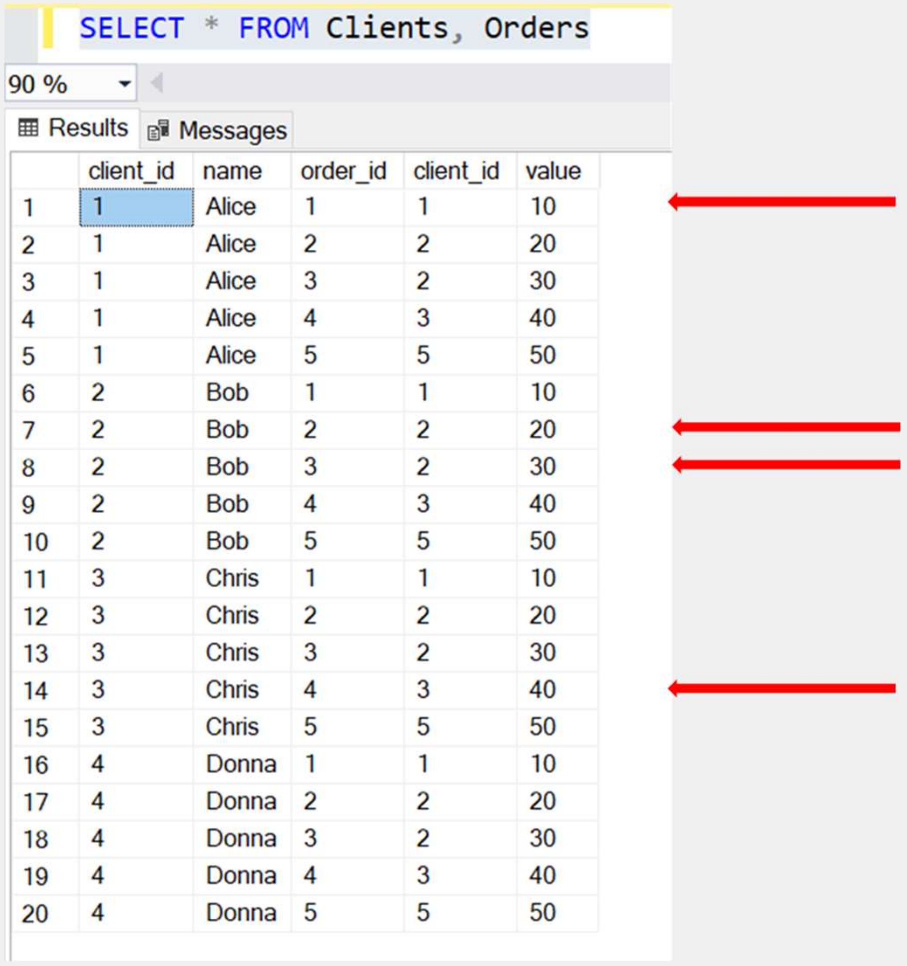
```
SELECT * FROM Clients, Orders
```

	client_id	name	order_id	client_id	value
1	1	Alice	1	1	10
2	1	Alice	2	2	20
3	1	Alice	3	2	30
4	1	Alice	4	3	40
5	1	Alice	5	5	50
6	2	Bob	1	1	10
7	2	Bob	2	2	20
8	2	Bob	3	2	30
9	2	Bob	4	3	40
10	2	Bob	5	5	50
11	3	Chris	1	1	10
12	3	Chris	2	2	20
13	3	Chris	3	2	30
14	3	Chris	4	3	40
15	3	Chris	5	5	50
16	4	Donna	1	1	10
17	4	Donna	2	2	20
18	4	Donna	3	2	30
19	4	Donna	4	3	40
20	4	Donna	5	5	50

Accessing all data from both tables

What are we getting?

Which of the records make sense? How can we select only them?



```
SELECT * FROM Clients, Orders
```

90 %

Results | Messages

	client_id	name	order_id	client_id	value
1	1	Alice	1	1	10
2	1	Alice	2	2	20
3	1	Alice	3	2	30
4	1	Alice	4	3	40
5	1	Alice	5	5	50
6	2	Bob	1	1	10
7	2	Bob	2	2	20
8	2	Bob	3	2	30
9	2	Bob	4	3	40
10	2	Bob	5	5	50
11	3	Chris	1	1	10
12	3	Chris	2	2	20
13	3	Chris	3	2	30
14	3	Chris	4	3	40
15	3	Chris	5	5	50
16	4	Donna	1	1	10
17	4	Donna	2	2	20
18	4	Donna	3	2	30
19	4	Donna	4	3	40
20	4	Donna	5	5	50

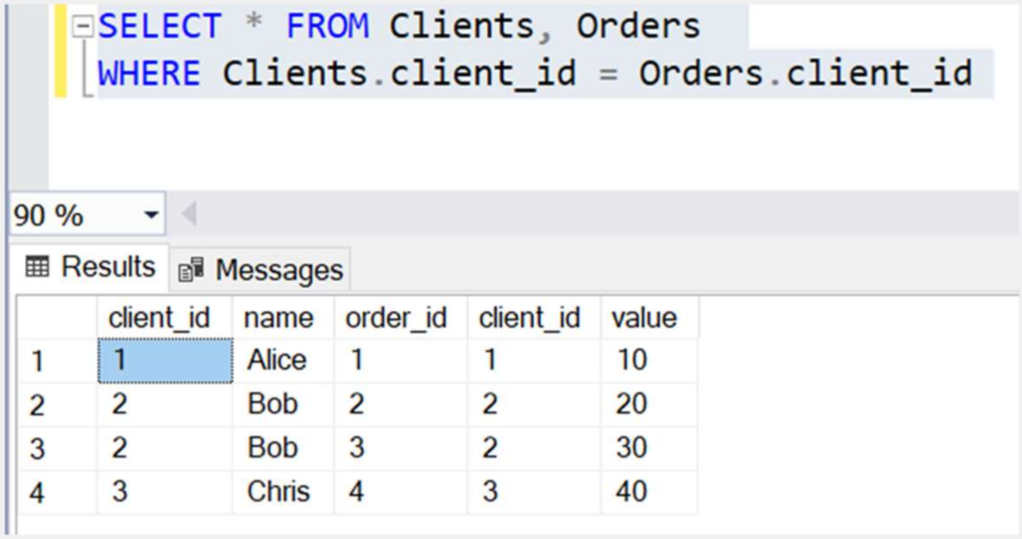
Inner join

How can we select only the records that make sense?

A better way to achieve the same result:

```
SELECT *
FROM Clients JOIN Orders
        ON Clients.client_id = Orders.client_id
```

This is **INNER JOIN**. It is the most fundamental and widely used. It is **strict**: it only contains **matching records from both tables**.



```
SELECT * FROM Clients, Orders
WHERE Clients.client_id = Orders.client_id
```

	client_id	name	order_id	client_id	value
1	1	Alice	1	1	10
2	2	Bob	2	2	20
3	2	Bob	3	2	30
4	3	Chris	4	3	40

Inner join

Returns only rows where the ON clause is true.

```
SELECT *
        FROM Table1 INNER JOIN Table2
                ON Table1.Col1 = Table2.Col2
```

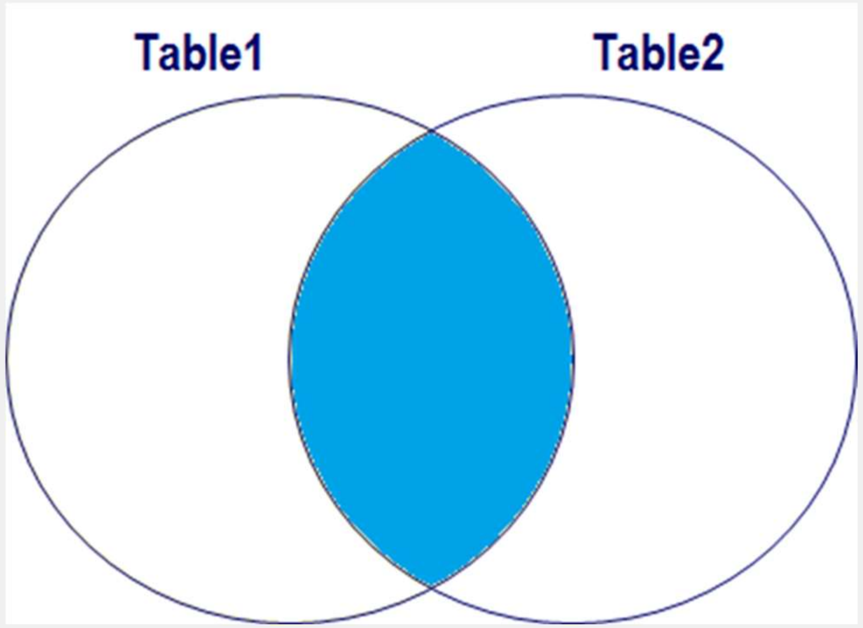


Table aliases

Concise representation is always better:

- Less to write
- Less risk of errors.

Instead of writing the names of the tables in the JOIN statement we can give them short names (aliases) and use them instead.

This is our query for the INNER JOIN:

```
SELECT *
FROM Clients JOIN Orders
       ON Clients.client_id =
Orders.client_id
```

Using aliases it becomes the following:

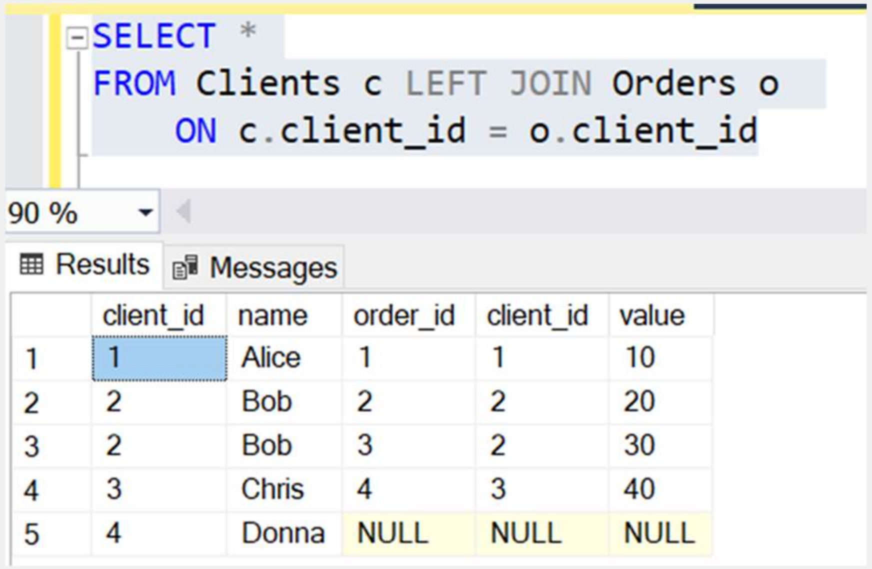
```
SELECT *
FROM Clients c JOIN Orders o
       ON c.client_id = o.client_id
```

Same result with less writing!

Left (outer) join

The **LEFT JOIN** returns the records returned by the INNER JOIN, plus "hanging" or "orphan" records from the **LEFT table**.

The **LEFT table** is the one which is **FIRST under the FROM clause** (easy to remember: we read from left to right).



```
SELECT *
FROM Clients c LEFT JOIN Orders o
    ON c.client_id = o.client_id
```

90 %

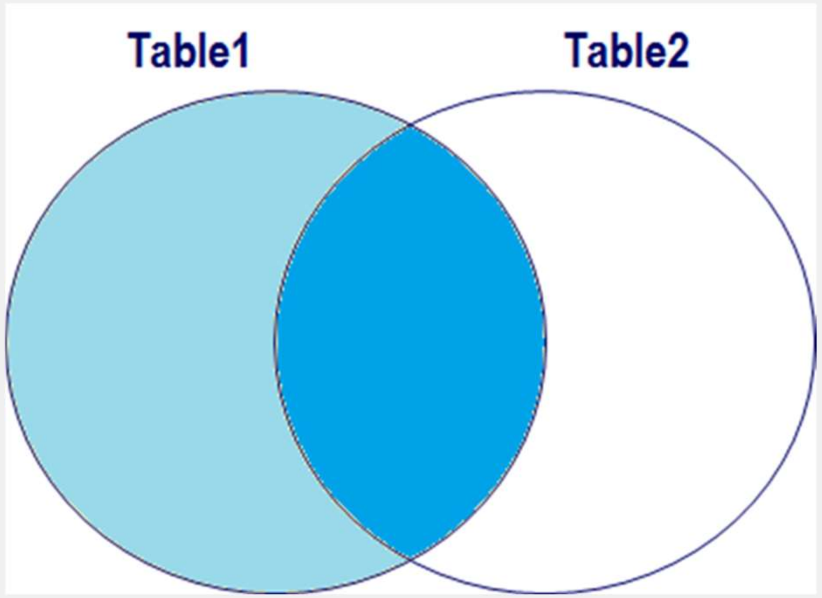
Results | Messages

	client_id	name	order_id	client_id	value
1	1	Alice	1	1	10
2	2	Bob	2	2	20
3	2	Bob	3	2	30
4	3	Chris	4	3	40
5	4	Donna	NULL	NULL	NULL

Left (outer) join

Returns all rows from Table1 and the rows from Table2 where the ON clause is true.

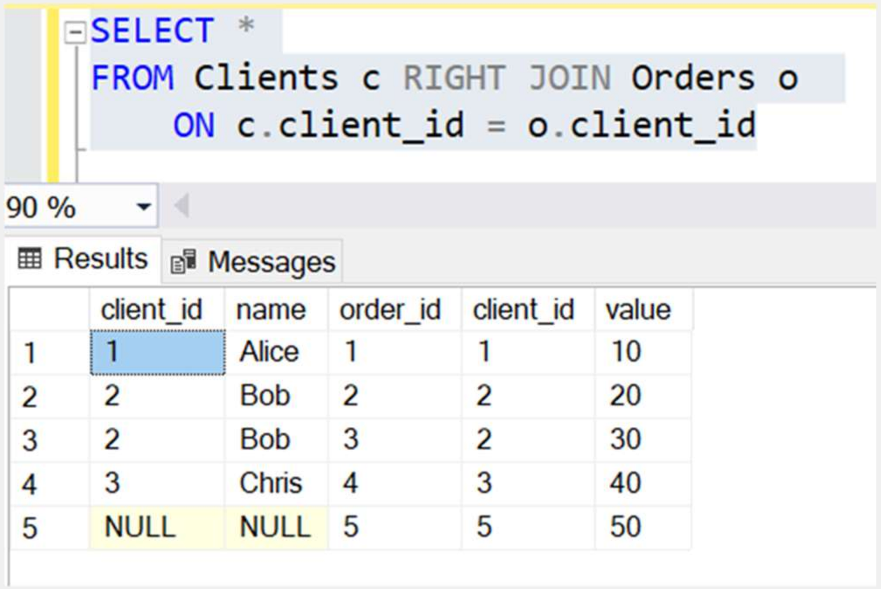
```
SELECT *
        FROM Table1 LEFT JOIN Table2
                ON Table1.Col1 = Table2.Col2
```



Right (outer) join

The **RIGHT JOIN** returns the records returned by the INNER JOIN, plus "hanging" or "orphan" records from the **RIGHT table**.

The **RIGHT table** is the one which is **SECOND under the FROM clause** (easy to remember: we read from left to right).



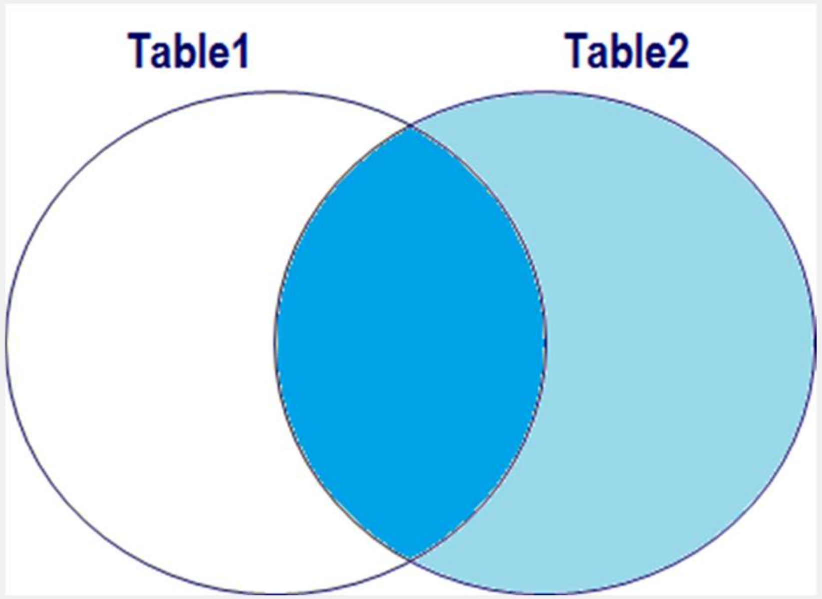
```
SELECT *
FROM Clients c RIGHT JOIN Orders o
    ON c.client_id = o.client_id
```

	client_id	name	order_id	client_id	value
1	1	Alice	1	1	10
2	2	Bob	2	2	20
3	2	Bob	3	2	30
4	3	Chris	4	3	40
5	NULL	NULL	5	5	50

Right (outer) join

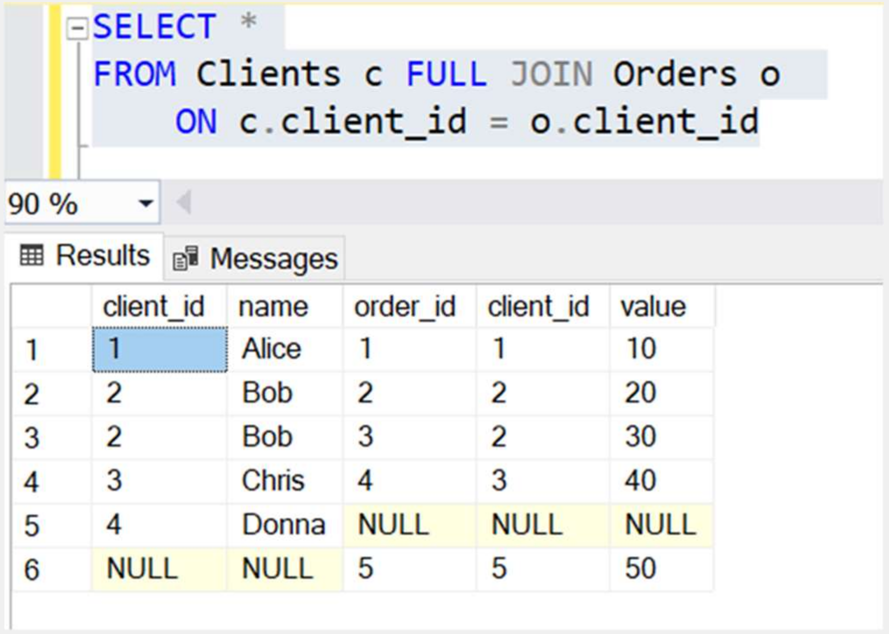
Returns all rows from Table2 and the rows from Table1 where the ON clause is true.

```
SELECT *
        FROM Table1 RIGHT JOIN Table2
                ON Table1.Col1 = Table2.Col2
```



Full (outer) join

The **FULL JOIN** is the combination of LEFT and RIGHT JOIN.



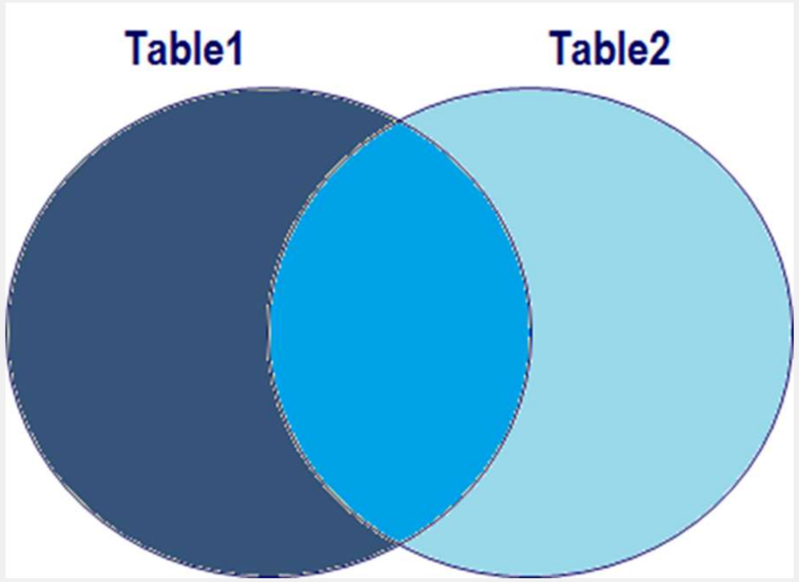
```
SELECT *
FROM Clients c FULL JOIN Orders o
    ON c.client_id = o.client_id
```

	client_id	name	order_id	client_id	value
1	1	Alice	1	1	10
2	2	Bob	2	2	20
3	2	Bob	3	2	30
4	3	Chris	4	3	40
5	4	Donna	NULL	NULL	NULL
6	NULL	NULL	5	5	50

Full (outer) join

Returns from both tables rows where the ON clause is true and disjointed rows where matches are not found.

```
SELECT *
        FROM Table1 FULL JOIN Table2
                ON Table1.Col1 = Table2.Col2
```



Joins in the select statement

SELECT <<field(s)>>

FROM <<table1>>

type-of **JOIN** *<<table2>>* **ON** *<<predicate>>*

WHERE <<condition(s)>>

GROUP BY <<field(s)>>

HAVING <<condition(s)>>

ORDER BY <<field(s)>>

Joins: usability

Imagine a real-life clients-and-orders database. Think of scenarios when each type of join would be used.

What type of joins would you use in your work?

Try joins

Using the Northwind database, write queries to display the following:

1. All categories with their IDs and names and the active (not discontinued) products in them with their IDs and names.
2. All customers with all their orders and order dates.
   Beware: some customers may not have placed any orders yet.
3. All customers that placed orders, with all their orders and order dates.

QA

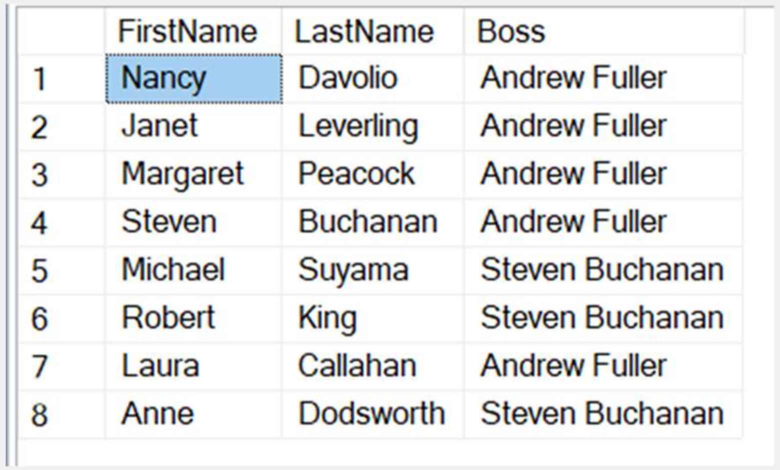
You can even join a table to itself

Using the Northwind database, let’s write a query to display a list of all employees and who they report to.

```
SELECT emp.FirstName,
emp.LastName,mgr.FirstName + ' ‘ +
       mgr.LastName AS Boss

FROME mployees emp JOIN Employees mgr

       ON emp.ReportsTo = mgr.EmployeeID
```



	FirstName	LastName	Boss
1	Nancy	Davolio	Andrew Fuller
2	Janet	Leverling	Andrew Fuller
3	Margaret	Peacock	Andrew Fuller
4	Steven	Buchanan	Andrew Fuller
5	Michael	Suyama	Steven Buchanan
6	Robert	King	Steven Buchanan
7	Laura	Callahan	Andrew Fuller
8	Anne	Dodsworth	Steven Buchanan