2021-12-09T18:00:00+02:00 | Tihomir Babic

How to Join Tables in SQL Without Using JOINs

Here's how you can combine tables without the JOIN keyword.

It seems like it shouldn't be possible: join tables in SQL without using the `JOIN` keyword. But using the `JOIN` keyword is not the only way to join tables in SQL.

This article will show you two additional methods for joining tables. Neither of them requires the `JOIN` keyword to work. They are:

> Using a comma between the table names in the `FROM` clause and specifying the joining condition in a `WHERE`
> Using `UNION`/`UNION ALL`.

I assume you know how `JOIN`s work. If you need it, here's a refresher on [how to join tables using JOINs](#).

Stop confusing INNER and OUTER JOINs with our interactive [SQL JOINs](#) course!

The First Method: FROM and WHERE

Syntax

The syntax for this method of joining tables without using `JOIN`s is:

```
SELECT *
FROM <table1>, <table2>
WHERE <condition>
</condition></table2></table1>
```

You can replace the `JOIN` keyword with a comma in the `FROM` clause. What do you do next? There's no `ON` keyword for you to state the joining condition as there would be when using `JOIN`, e.g., on which two columns you want to join the tables. In this method, you simply use a `WHERE` clause to do so.

This method does exactly the same as:

```
SELECT *
FROM <table1> JOIN <table2>
ON <condition>
```

Let's see how it works in practice.

Example

In this example, I need to find the artist's name, the albums he/she has recorded, and the years in which they were recorded, for every artist.

To show you how this method works, I'll use two tables. The first one is **`artist`**:

id	artist_name
1	Paul Simon
2	Marvin Gaye
3	Bettye LaVette
4	Joni Mitchell
5	Sly and the Family Stone

It's a simple table containing several of my favorite musicians.

The other table, **`album`**, looks like this:

id	album_name	year_recorded	artist_id
1	Dance to the Music	1968	5
2	Small Talk	1974	5
3	Stranger to Stranger	2016	1
4	I've Got My Own Hell to Raise	2005	3
5	Blues	1971	4
6	Court and Spark	1974	4
7	A Tribute to the Great Nat "King" Cole	1965	2

id	album_name	year_recorded	artist_id
8	Graceland	1986	1
9	Still Crazy After All These Years	1975	1
10	In the Blue Light	2018	1
11	There's a Riot Goin' On	1971	5
12	The Scene of the Crime	2007	3
13	Mingus	1979	4
14	What's Going On	1971	2
15	Here, My Dear	1978	2

These are the albums recorded by the musicians from the first table.

Query

The query for joining these tables without a `JOIN` keyword looks like this:

```
SELECT
    artist_name,
    album_name,
    year_recorded
FROM artist, album
WHERE artist.id = album.artist_id;
```

First, I specify the columns I want to see in my result: `artist_name`, `album_name`, and `year_recorded`. They come from either **`artist`** or **`album`**. I join these two tables simply by listing them in the `FROM` clause and separating them with a comma.

The `WHERE` clause is generally used to filter results. I can use it to filter this result so that it shows all data where the column `id` from the table **`artist`** is equal to the column `artist_id` from the table **`album`**. You can see the condition in WHERE in this query is the same as it would be after the ON keyword when using `JOIN`: `artist.id = album.artist_id`.

The equivalent query with a `JOIN` keyword is:

```
SELECT  artist_name,
        album_name,
        year_recorded
FROM artist
JOIN album
ON artist.id = album.artist_id;
```

Result

Both of these queries return the same result:

artist_name	album_name	year_recorded
Paul Simon	In the Blue Light	2018
Paul Simon	Still Crazy After All These Years	1975
Paul Simon	Graceland	1986
Paul Simon	Stranger to Stranger	2016
Marvin Gaye	Here, My Dear	1978
Marvin Gaye	What's Going On	1971
Marvin Gaye	A Tribute to the Great Nat "King" Cole	1965
Bettye LaVette	The Scene of the Crime	2007
Bettye LaVette	I've Got My Own Hell to Raise	2005
Joni Mitchell	Mingus	1979
Joni Mitchell	Court and Spark	1974
Joni Mitchell	Blues	1971
Sly and the Family Stone	There's a Riot Goin' On	1971
Sly and the Family Stone	Small Talk	1974
Sly and the Family Stone	Dance to the Music	1968

You see the result combines data from the two initial tables. I can now see, for example, Paul Simon recorded the album “In the Blue Light” in 2018. If you don’t trust my query, you can always check Wikipedia!

This method uses a common column to join tables. Recall that you can also join tables that do not have a common column.

Do you want to practice your SQL skills? That's great! Check out our SQL Practice track.

The Second Method: UNION/UNION ALL

Syntax

```
SELECT
    <column1>,
    <column2>,
    …
FROM <table1>


UNION / UNION ALL


SELECT
    <column1>,
    <column2>,
    …
FROM <table2>
</table2></column2></column1></table1></column2></column1>
```

Both `UNION` and `UNION ALL` are used to glue together the result of the first query with that of the second query. You simply write two `SELECT` statements and write `UNION` or `UNION ALL` between them. But what is the difference between those two SQL keywords? Recall:

> `UNION` removes duplicate rows from the query and shows them only once.
>
> `UNION ALL` shows all the rows from both tables, no matter how many duplicates there are.

For both `UNION` and `UNION ALL` to work, all columns in the two `SELECT` statements need to match. This means the number of columns selected must be the same, and the data types of those columns must be the same.

These two operators are called set operators. Read this article to find out more about these and other set operators.

Example

We have two tables in this example, also. The first is named **`customer`**:

id	first_name	last_name
1	Jimmy	Hellas
2	Francois	Jambony
3	Renata	Traviata
4	Carmen	Naburana
5	Steve	Inicks
6	Elvira	Samson

id	first_name	last_name
7	Alpa	Cino
8	Rosa	Sparks
9	Bernardine	Sane
10	Bert	Randrussell

It contains a list of my customers.

The second table is `supplier`:

id	first_name	last_name
1	Arpad	Polanski
2	Billy	Van Persie
3	Steve	Inicks
4	Jean-Claude	Zaandam
5	Emina	Arkayeva
6	Bo	Milosz
7	Jan	Wrangler
8	Heinrich	Stroopwafel
9	Herringo	Starr
10	Josephine	Matijevic
11	Alpa	Cino
12	Flint	Westwood
13	Des	Paulmond
14	Wilma	Jackson
15	Katrin	Verson
16	Bert	Randrussell
17	Carmen	Naburana
18	Jimmy	Tulp
19	Luis	Wolf
20	Miranda	Fanucci

If you inspect these tables a little closer, you see some people are in both tables. For example, look at Steve Inicks – this means Steve is, along with some others, both my customer (he buys some services from my company) and my supplier (he provides some services to me).

Now, let's see what it looks like when I join these tables with `UNION` or `UNION ALL`.

Query: UNION

This method of combining tables without using the `JOIN` keyword translates to the following query:

```
SELECT
    first_name,
    last_name
FROM customer

UNION


SELECT
    first_name,
    last_name
FROM supplier;
```

The first `SELECT` statement selects the columns `first_name` and `last_name` from the table **`customer`**. The second `SELECT` statement does the same but from the table **`supplier`**. I use `UNION` to join the results of these two queries. It works: there are two columns in both `SELECT` statements, and they are of the same data type. They are even named the same, although that is not necessary for the query to work.

Result: UNION

Running the query gets you this:

first_name	last_name
Alpa	Arpad
Cino	Billy
Josephine	Matijevic
Luis	Wolf
Jean-Claude	Zaandam
Steve	Inicks
Billy	Van Persie
Emina	Arkayeva
Miranda	Fanucci
Bo	Milosz

first_name	last_name
Jimmy	Hellas
Francois	Jambony
Rosa	Sparks
Katrin	Verson
Jan	Wrangler
Bernardine	Sane
Heinrich	Stroopwafel
Elvira	Samson
Renata	Traviata
Jimmy	Tulp
Wilma	Jackson
Arpad	Polanski
Bert	Randrussell
Des	Paulmond
Flint	Westwood
Carmen	Naburana
Herringo	Starr

As I mentioned, it's a combination of all data from both tables.

Let's check for duplicates. I've mentioned Steve Inicks is in both `customer` and `supplier` tables. In the query result, you find him only once. I've marked the row where he is.

Let's see how this is different from `UNION ALL`.

Query: UNION ALL

The query in this case is:

```
SELECT
    first_name,
    last_name
FROM customer


UNION ALL


SELECT
    first_name,
    last_name
FROM supplier;
```

As you can see, this isn't very different. In fact, it is the same query, except it has `UNION ALL` instead of `UNION`.

However, the results are different. Let's take a look.

Result: UNION ALL

The result of the query is:

first_name	last_name
Jimmy	Hellas
Francois	Jambony
Renata	Traviata
Carmen	Naburana
Steve	Inicks
Elvira	Samson
Alpa	Cino
Rosa	Sparks
Bernardine	Sane
Bert	Randrussell
Arpad	Polanski
Billy	Van Persie
Steve	Inicks
Jean-Claude	Zaandam
Emina	Arkayeva
Bo	Milosz
Jan	Wrangler
Heinrich	Stroopwafel
Herringo	Starr
Josephine	Matijevic
Alpa	Cino
Flint	Westwood
Des	Paulmond
Wilma	Jackson
Katrin	Verson
Bert	Randrussell
Carmen	Naburana

first_name	last_name
Jimmy	Tulp
Luis	Wolf
Miranda	Fanucci

Knowing Steve Inicks appears as both a customer and a supplier, let's check how many times he appears in the result. He is there twice, both marked in blue. I've also marked three others who appear twice: Alpa Cino, Bert Randrussell, and Carmen Naburana.

Here's another example of using UNION and UNION ALL..

Yes, Tables Can Be Joined Without the JOIN Keyword

As you have just seen, it's not always necessary to use the JOIN keyword to combine two tables in SQL.

You can replace it with a comma in the `FROM` clause then state your joining condition in the `WHERE` clause.

The other method is to write two `SELECT` statements. The result of each is a table, so simply use `UNION` or `UNION ALL` to combine the two. Keep in mind this works only if your `SELECT` statements have the same number of columns and the same data types.

Use `UNION` when you don't want to show duplicates and `UNION ALL` when you want to see duplicates.

Complete learning path for modern SQL. Try out our SQL from A to Z track. 7 hands-on SQL courses with over 800 exercises!

Learn all the Possible Ways for Joining Tables in SQL

The more ways you know how to join tables in SQL, the better. With these two methods plus the JOIN, joining tables becomes easier. Your query results are closer to what you need, and you get more out of SQL.

Knowing a variety of ways to join tables also allows you to shake up your code a little. It can be boring to write everything the same way all the time. But most importantly, each of the three methods is useful in different situations.

You can learn all these methods of joining tables in our SQL Basics course. This is only a first step toward mastering SQL from A to Z. You also need a lot of practice to master joining tables. Maybe, the best way is to choose at least one online option from our list of ten.

Viewed using Just Read