# UNION to Combine Results

*by Sophia*

WHAT'S COVERED

This tutorial explores the UNION operator to combine result sets of various queries in four parts:

1. Introduction
2. Examples
3. Multiple UNION
4. Retaining Duplicates

## 1. Introduction

The UNION operator allows you to combine the result sets from two or more different SELECT statements into a single result set while excluding duplicate rows. The individual SELECT statements will not have any difference other than the first SELECT statement will not have a semi-colon at the end of it. For example:

```
SELECT <columnlist>
FROM <table1>
SELECT <columnlist>
FROM <table2>;
```

## 2. Examples

Let's take a look at an example in which our organization may want to send all users (customers and employees) an email to inform them of an upcoming sale. The same message would be sent to all individuals. Instead of querying the customer and employee table separately and having two result sets to merge together, we can pull their name and email into a single result set:

```
SELECT first_name, last_name, email
FROM employee
UNION
SELECT first_name, last_name, email
FROM customer;
```

**Query Results**

Row count: 67

| first_name | last_name | email |
|---|---|---|
| Manoj | Pareek | manoj.pareek@rediff.com |
| Alexandre | Rocha | alero@uol.com.br |
| Hannah | Schneider | hannah.schneider@yahoo.de |
| Wyatt | Girard | wyatt.girard@yahoo.fr |
| Helena | Holý | hholy@gmail.com |
| Julia | Barnett | jubarnett@gmail.com |

Note that this will not distinguish which individual is a customer or employee. We could add an extra string in the column list to distinguish the difference:

```
SELECT first_name, last_name, email, 'Employee'
FROM employee
UNION
SELECT first_name, last_name, email, 'Customer'
FROM customer;
```

**Query Results**

Row count: 67

| first_name | last_name | email | Employee |
|---|---|---|---|
| Steve | Johnson | steve@chinookcorp.com | Employee |
| Laura | Callahan | laura@chinookcorp.com | Employee |
| Margaret | Park | margaret@chinookcorp.com | Employee |
| Robert | King | robert@chinookcorp.com | Employee |
| Andrew | Adams | andrew@chinookcorp.com | Employee |
| Nancy | Edwards | nancy@chinookcorp.com | Employee |
| Michael | Mitchell | michael@chinookcorp.com | Employee |
| Jane | Peacock | jane@chinookcorp.com | Employee |
| Leonie | Köhler | leonekohler@surfeu.de | Customer |
| Hugh | O'Reilly | hughoreilly@apple.ie | Customer |

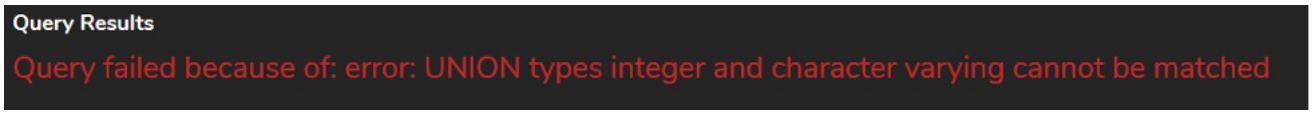
Note that your result set may look different in this case, as the 4th column was sorted by the type in descending order.

To use the UNION operator, the tables that we are querying from should have the same attribute characteristics, meaning that the number of columns and types of data between the two SELECT statements should match.

```
SELECT customer_id
FROM invoice
UNION
```

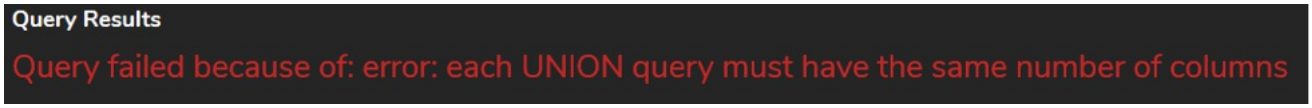
```
SELECT first_name
FROM customer;
```

In the example above, the customer_id is an integer, while the first_name is a character string. As such, the data types don't match and we get the following error:




If we have a different number of columns in each of the SELECT statements, we would run into another error:

```
SELECT customer_id
FROM invoice
UNION
SELECT customer_id, first_name
FROM customer;
```

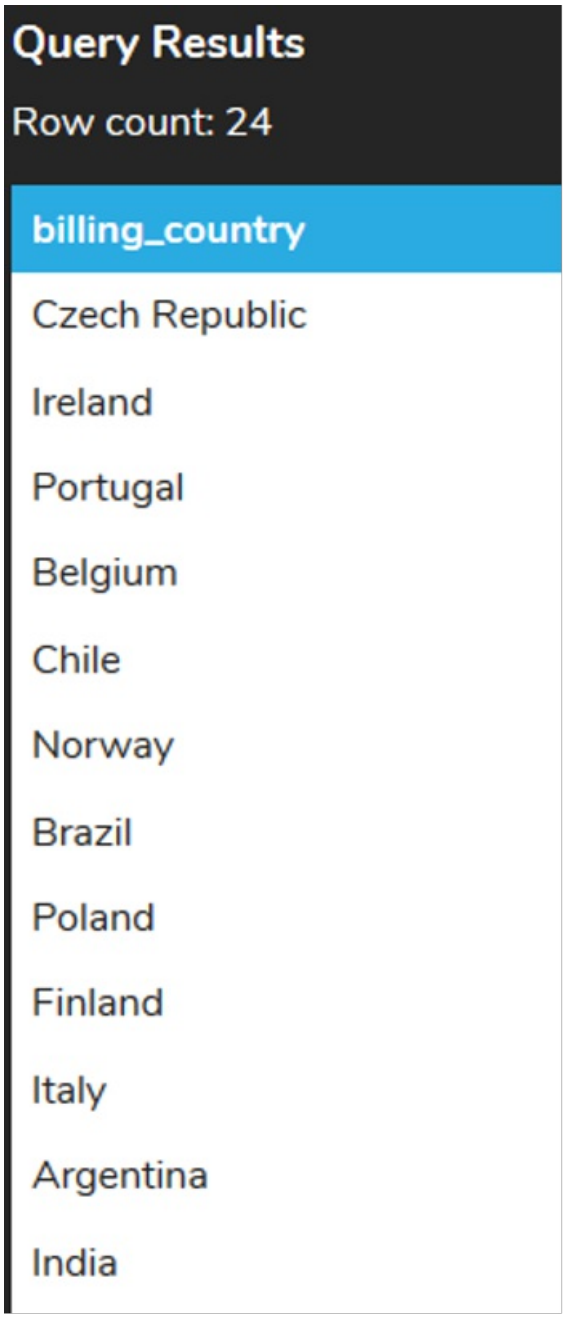



# 3. Multiple UNION

We could create a UNION for more than 2 queries as well. For example, we may want to look at all of the countries that we operate in. As such, we would need to look at the customer table, invoice, and employee table.

```
SELECT billing_country
FROM invoice
UNION
SELECT country
FROM customer
UNION
SELECT country
FROM employee;
```

Notice that the result set only has 24 rows, as it excludes all duplicate values:



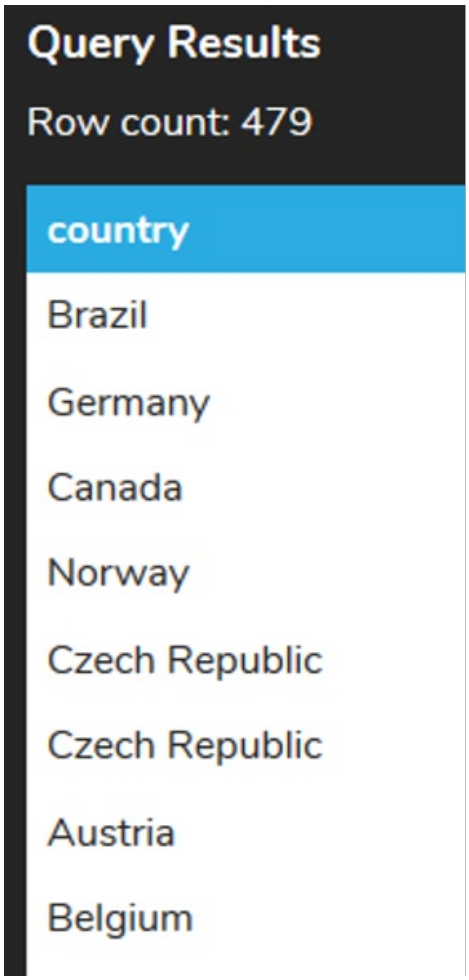

# 4. Retaining Duplicates

If we wanted to retain the duplicate values, we would use UNION ALL instead of UNION:

```
SELECT billing_country
FROM invoice
UNION ALL
SELECT country
FROM customer
UNION ALL
SELECT country
FROM employee;
```


You should see now that the duplicates are included, and we have 479 results returned. Note, as well, that the column name that is used as the output reflects the first SELECT statement in the list. If we swapped the first SELECT statement with the second, the country column displays as the output:

```
SELECT country
FROM customer
UNION ALL
SELECT billing_country
FROM invoice
UNION ALL
SELECT country
FROM employee;
```




As such, if we wanted to use aliases to rename a column, we would only need to do so for the first SELECT statement:

```
SELECT first_name as "First Name", last_name as "Last Name", email as "Email", 'Employee' as "Type"
FROM employee
UNION
SELECT first_name, last_name, email, 'Customer'
FROM customer;
```

Query Results

Row count: 67

| First Name | Last Name | Email | Type |
|---|---|---|---|
| Emma | Jones | emma_jones@hotmail.com | Customer |
| Stanisław | Wójcik | stanisław.wójcik@wp.pl | Customer |
| Victor | Stevens | vstevens@yahoo.com | Customer |
| Daan | Peeters | daan_peeters@apple.be | Customer |
| Alexandre | Rocha | alero@uol.com.br | Customer |

## Video Transcription

[MUSIC PLAYING] The union keyword allows us to be able to combine results from multiple different select statements. There are instances where we want to query different sets of data to be able to provide us a single result set. In essence, we're not really joining them based on anything in particular,

but rather we want to combine what the results are going to be. So in this case here, we're going to go ahead and get all the information from the employee, which could be the first name, last name, and email. And we want to combine that with the same results set to be able to send them and get the first name, last name, and email of the customer.

Rather than run each one of these individually, we're going to go ahead and select all this with a union between, and now we're going to have the same results set with 67 results that combine the two. In support a note with the union, it removes any duplicate information. So if you did have individuals or employees that were also customers, it would only list them once.

This is another example. We're going to take the country from the invoice, the customer, and the employee tables and combine them together with a union. Again, we're only going to list them individually. However, if we wanted to display duplicates, instead of union, we can use union all, and by doing so, it's going to combine all the results with the duplicates in place.

[MUSIC PLAYING]


Your turn! Open the SQL tool by clicking on the LAUNCH DATABASE button below. Then enter in one of the examples above and see how it works. Next, try your own choices for which columns you want the query to provide.


The UNION operator allows result sets to be combined together into a single result set.

Source: Authored by Vincent Tran