WHERE to Filter Data

by Sophia Tutorial

WHAT'S COVERED

This tutorial explores using the WHERE clause within a SELECT statement to filter data in the result set in three parts:

1. Getting Started
2. Filtering Strings
3. Comparison Operators

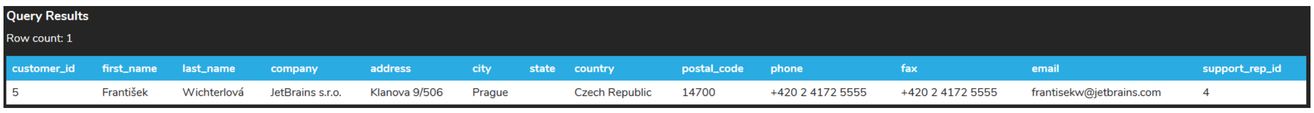
1. Getting Started

The WHERE clause is used to filter records in a SELECT statement. The WHERE clause is optional, and adds conditional restrictions to the SELECT statement that will help limit the result set. It only displays the records that fit the condition listed in the WHERE clause. By using the WHERE clause, you can easily answer questions like:

- Which invoices have a total greater than 14?
- Which customers live in Canada?
- Which employees report to the General Manager?

For example, if we wanted to find the customer information of the customer_id that was equal to 5, we would run it as:

```
SELECT *
FROM customer
WHERE customer_id = 5;
```



Query Results

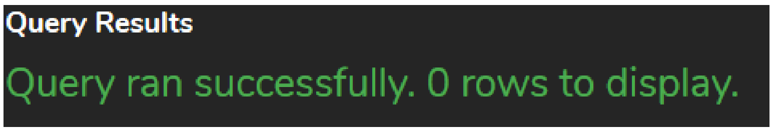
Row count: 1

customer_id	first_name	last_name	company	address	city	state	country	postal_code	phone	fax	email	support_rep_id
5	František	Wichterlová	JetBrains s.r.o.	Klanova 9/506	Prague		Czech Republic	14700	+420 2 4172 5555	+420 2 4172 5555	frantisekw@jetbrains.com	4

Notice that in the WHERE clause, we define the column (customer_id), the comparison operator (=), and the value that we wanted to compare it to (5).

If there are no rows that match the criteria in the WHERE clause, you should see a message similar to the following:

```
SELECT *
FROM customer
WHERE customer_id = 1000;
```


Query Results

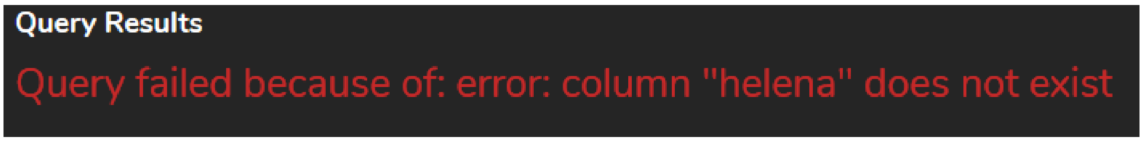
Query ran successfully. 0 rows to display.

2. Filtering Strings

Note that SQL requires single quotes around text values. Numeric values should not be enclosed in quotes. Here is an example of what would happen if we forgot to include quotes around the text value 'Helena':

```
SELECT *
FROM customer
WHERE first_name = Helena;
```

We would get an error message:

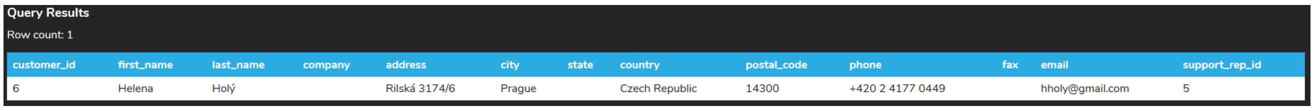

Query Results

Query failed because of: error: column "helena" does not exist

This is because the database thinks the text value is a column. This could also present a problem if the text value is also an actual column. You would not get an error message; however, the results would not be what wanted either.

To properly use the WHERE clause, you would use the single quotes around the text values:

```
SELECT *
FROM customer
WHERE first_name = 'Helena';
```


Query Results

Row count: 1

customer_id	first_name	last_name	company	address	city	state	country	postal_code	phone	fax	email	support_rep_id
6	Helena	Holý		Rilská 3174/6	Prague		Czech Republic	14300	+420 2 4177 0449		hholy@gmail.com	5

3. Comparison Operators

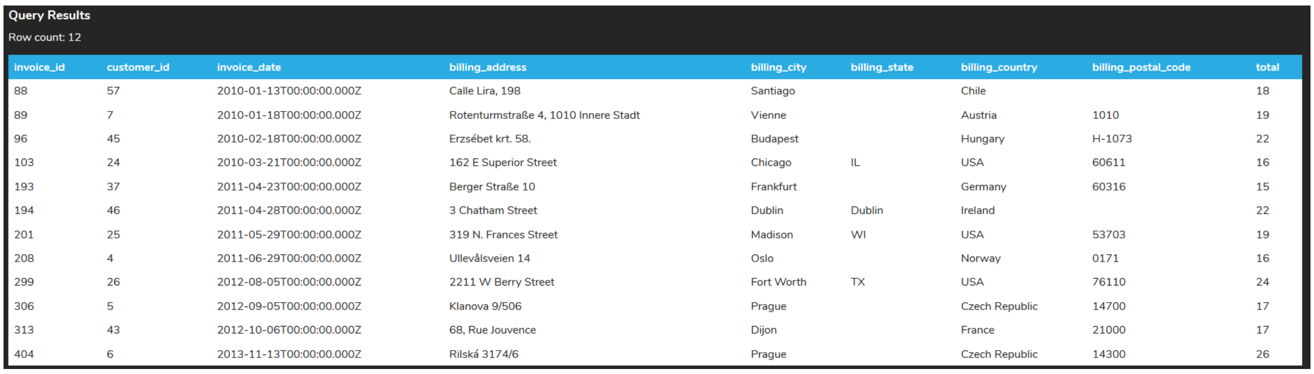
We looked at the = operator above, but there are many other operators that can be used in the WHERE clause. Other comparison operators include:

= means equal to
< means less than
<= means less than or equal to
> means greater than

>= means greater than or equal to
<> means not equal to

Let us find the invoices that have a total greater than 14.

```
SELECT *
FROM invoice
WHERE total > 14;
```

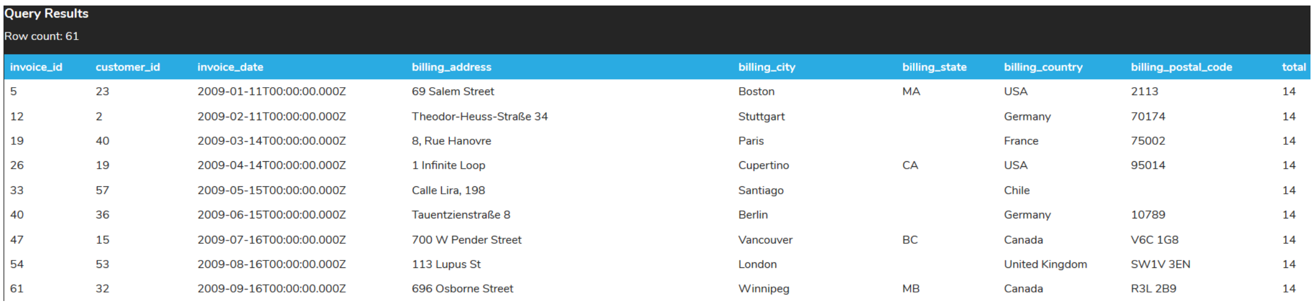


Query Results
Row count: 12

invoice_id	customer_id	invoice_date	billing_address	billing_city	billing_state	billing_country	billing_postal_code	total
88	57	2010-01-13T00:00:00.000Z	Calle Lira, 198	Santiago		Chile		18
89	7	2010-01-18T00:00:00.000Z	Rotenturmstraße 4, 1010 Innere Stadt	Vienne		Austria	1010	19
96	45	2010-02-18T00:00:00.000Z	Erzsébet krt. 58.	Budapest		Hungary	H-1073	22
103	24	2010-03-21T00:00:00.000Z	162 E Superior Street	Chicago	IL	USA	60611	16
193	37	2011-04-23T00:00:00.000Z	Berger Straße 10	Frankfurt		Germany	60316	15
194	46	2011-04-28T00:00:00.000Z	3 Chatham Street	Dublin	Dublin	Ireland		22
201	25	2011-05-29T00:00:00.000Z	319 N. Frances Street	Madison	WI	USA	53703	19
208	4	2011-06-29T00:00:00.000Z	Ullevålsveien 14	Oslo		Norway	0171	16
299	26	2012-08-05T00:00:00.000Z	2211 W Berry Street	Fort Worth	TX	USA	76110	24
306	5	2012-09-05T00:00:00.000Z	Klanova 9/506	Prague		Czech Republic	14700	17
313	43	2012-10-06T00:00:00.000Z	68, Rue Jouvence	Dijon		France	21000	17
404	6	2013-11-13T00:00:00.000Z	Rilská 3174/6	Prague		Czech Republic	14300	26

The result set includes 12 rows. If we change the WHERE clause to >= 14, and include all invoices with the value of 14, the result set goes from 12 rows to 61 rows returned.

```
SELECT *
FROM invoice
WHERE total >= 14;
```



Query Results
Row count: 61

invoice_id	customer_id	invoice_date	billing_address	billing_city	billing_state	billing_country	billing_postal_code	total
5	23	2009-01-11T00:00:00.000Z	69 Salem Street	Boston	MA	USA	2113	14
12	2	2009-02-11T00:00:00.000Z	Theodor-Heuss-Straße 34	Stuttgart		Germany	70174	14
19	40	2009-03-14T00:00:00.000Z	8, Rue Hanovre	Paris		France	75002	14
26	19	2009-04-14T00:00:00.000Z	1 Infinite Loop	Cupertino	CA	USA	95014	14
33	57	2009-05-15T00:00:00.000Z	Calle Lira, 198	Santiago		Chile		14
40	36	2009-06-15T00:00:00.000Z	Tauentzienstraße 8	Berlin		Germany	10789	14
47	15	2009-07-16T00:00:00.000Z	700 W Pender Street	Vancouver	BC	Canada	V6C 1G8	14
54	53	2009-08-16T00:00:00.000Z	113 Lupus St	London		United Kingdom	SW1V 3EN	14
61	32	2009-09-16T00:00:00.000Z	696 Osborne Street	Winnipeg	MB	Canada	R3L 2B9	14

When it comes to integer values being compared, there would be no difference between using these two statements:

```
SELECT *
FROM invoice
WHERE total >= 15;
```

or

```
SELECT *
FROM invoice
WHERE total > 14;
```

However, if the total contained decimal values, these two statements would not be equivalent. The second statement would return rows with a total value between 14 and 15, like 14.5, whereas the first one would not.

Video Transcription

[MUSIC PLAYING] Using the where clause in the select statement, you have the ability to be able to filter out data within the table itself. Utilizing the query select star from customer, we're returning every single row within the table itself. However, there's lots of instances in which we don't want to do so. We want to be able to filter out the details.

So for example, if we're looking for a specific customer ID, we can add in a where. We enter in the column name and equal to the specific value. In this case here, we'll enter in 5. So this should return just one row with the customer ID equal to 5, which we'll see here.

Another option as well, is that you can actually select based on different ranges. So for example, if you want to search for the customer ID with all the different values less than 5, it should return four rows with customer ID between 1 and 4.

We also have the option to be able to search on text. For example, if we're looking for those that live in Canada-- we're looking for the country, equal-- one thing to note is that when it comes to text within Postgres, you have to ensure that you are utilizing single quotes around the text.

If you don't enter in single quotes around the text, what you'll see is an error where it identifies that it's looking for a column, saying that the column Canada does not exist, being that this is a string literal. So because of that, we need to enter in single quotes to be able to have it work correctly. Now we have all eight rows returned. All of the individuals are from Canada.

[MUSIC PLAYING]

TRY IT

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 WHERE clauses.

SUMMARY

Filtering data using the WHERE clause in SELECT statements can help limit the amount of data being returned based on the conditions listed. There are multiple comparison operators that can be used to filter data.

Source: Authored by Vincent Tran