INSERT INTO to Add Multiple Rows

by Sophia Tutorial

WHAT'S COVERED

This tutorial explores using the INSERT INTO command to add multiple rows into a table in two parts:

1. Getting Started
2. Using the RETURNING Statement

1. Getting Started

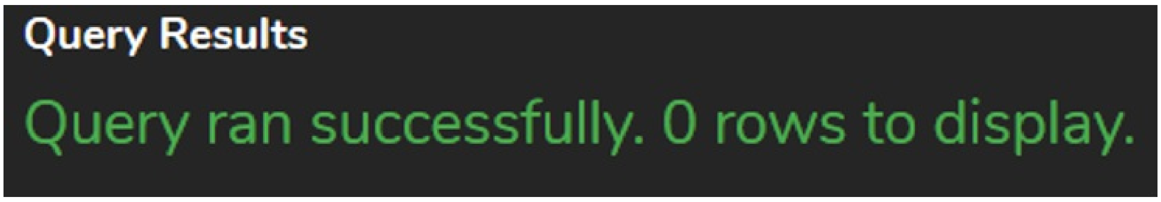
There are many instances when we may be loading multiple sets of data. It could be inefficient to constantly list out the column list each time, especially if it is the same with each INSERT INTO statement. Luckily, most databases will allow you to include multiple value lists together in a single INSERT statement. The syntax will look like the following:

```
INSERT INTO <tablename> ( <column1>, <column2>, …)
VALUES (<value1>,<value2>, …),
(<value1>,<value2>, …),
(<value1>,<value2>, …),
(<value1>,<value2>, …);
```

It's quite similar to a regular INSERT statement, but instead of having just one set of values, we have each value list separated by commas but enclosed by parentheses.

Let's create a new table to test this on:

```
CREATE TABLE referral( referral_id SERIAL, first_name VARCHAR(50), last_name VARCHAR(50), email VARCHAR(50) );
```



```
Query Results
Query ran successfully. 0 rows to display.
```

This statement will create a new table called referral to store the first name, last name, and email of individuals. A standard INSERT statement with one record would look like the following:

```
INSERT INTO referral (first_name,last_name,email)
VALUES ('Sandra','Boynton','s.boy@email.com');
```

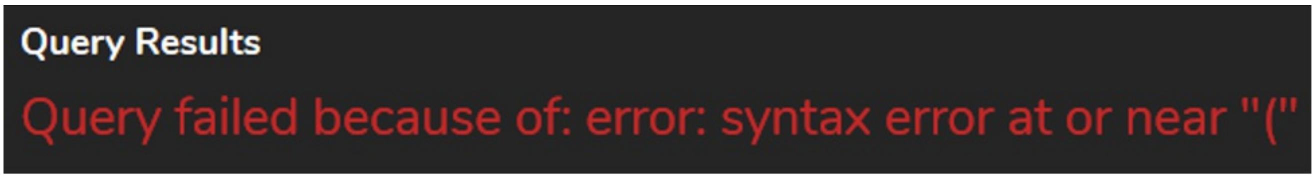
If we wanted to insert multiple rows at once, the INSERT statement would look like the following:

```
INSERT INTO referral (first_name,last_name,email)
VALUES ('Randall','Faustino','s.boy@email.com'),
('Park','Deanna','p.deanna@email.com'),
('Sunil','Carrie','s.carrie@email.com'),
('Jon','Brianna','j.brianna@email.com'),
('Lana','Jakoba','l.jakoba@email.com'),
('Tiffany','Walker','t.walk@email.com');
```

Note, though, if we forgot to include commas between each value list like this:

```
INSERT INTO referral (first_name,last_name,email)
VALUES ('Randall','Faustino','s.boy@email.com')
('Park','Deanna','p.deanna@email.com')
('Sunil','Carrie','s.carrie@email.com')
('Jon','Brianna','j.brianna@email.com')
('Lana','Jakoba','l.jakoba@email.com')
('Tiffany','Walker','t.walk@email.com');
```

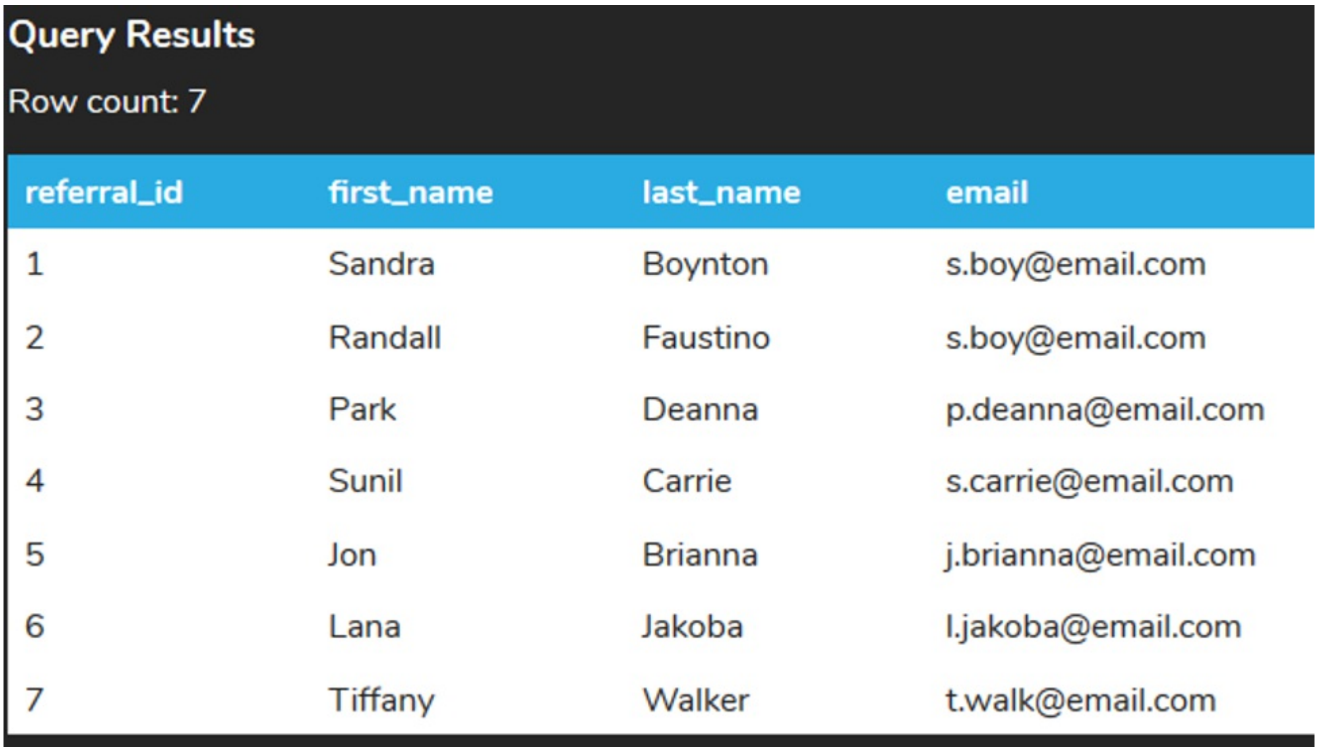
We would get this error:



```
Query Results
Query failed because of: error: syntax error at or near "("
```

Once inserted correctly, we should be able to take a look at what is in the table:

```
SELECT *
FROM referral;
```



Query Results

Row count: 7

referral_id	first_name	last_name	email
1	Sandra	Boynton	s.boy@email.com
2	Randall	Faustino	s.boy@email.com
3	Park	Deanna	p.deanna@email.com
4	Sunil	Carrie	s.carrie@email.com
5	Jon	Brianna	j.brianna@email.com
6	Lana	Jakoba	l.jakoba@email.com
7	Tiffany	Walker	t.walk@email.com

Notice that we have the referral_id incrementing automatically. If you were observant, you would also see an error in Randall Faustino's email. We will cover how to update this in an upcoming tutorial.

2. Using the RETURNING Statement

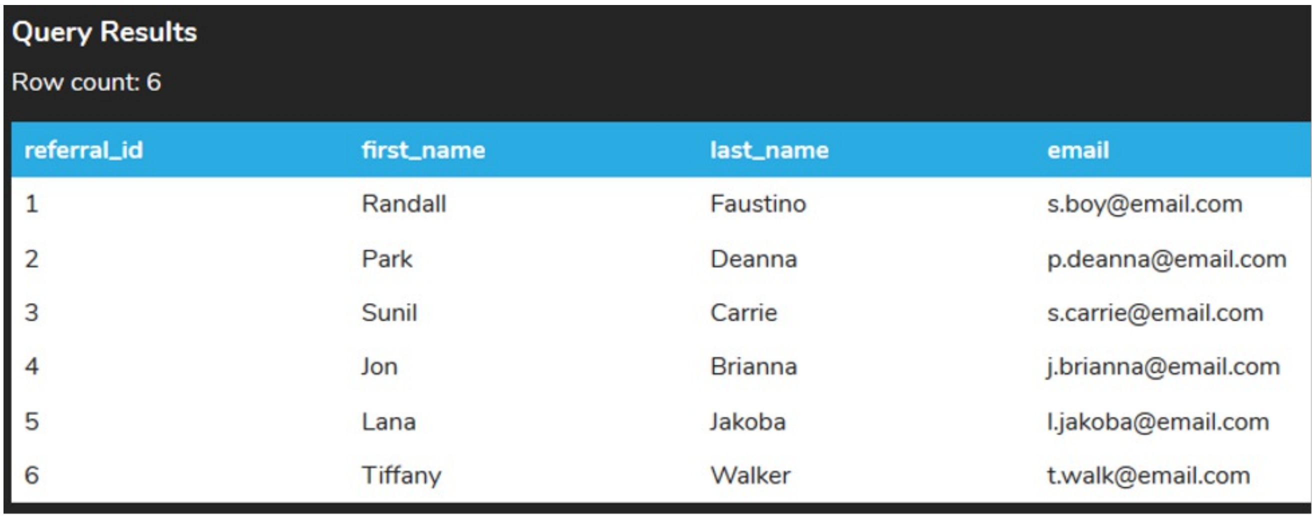
Let's recreate the table so that we can look at another example, although this is specific to PostgreSQL. Run the following statement to drop the table:

```
DROP TABLE referral;

CREATE TABLE referral( referral_id SERIAL, first_name VARCHAR(50), last_name VARCHAR(50), email VARCHAR(50) );
```

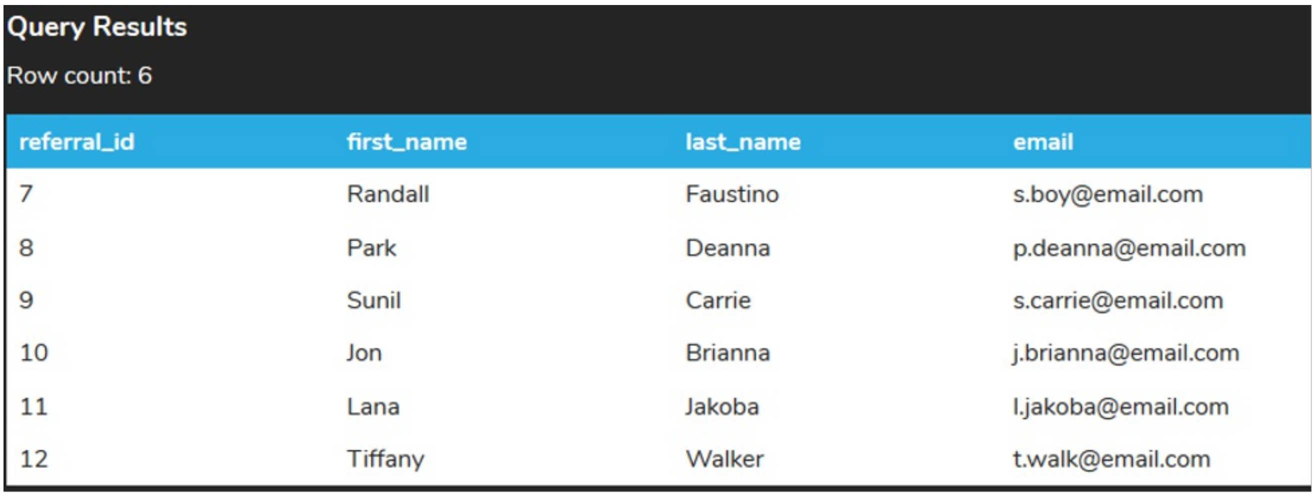
It can be useful to add RETURNING * at the end of the INSERT INTO statement. By doing so, we'll automatically query the results and output them to the user.

```
INSERT INTO referral (first_name,last_name,email)
VALUES ('Randall','Faustino','s.boy@email.com'),
('Park','Deanna','p.deanna@email.com'),
('Sunil','Carrie','s.carrie@email.com'),
('Jon','Brianna','j.brianna@email.com'),
('Lana','Jakoba','l.jakoba@email.com'),
('Tiffany','Walker','t.walk@email.com')
RETURNING *;
```


Query Results
Row count: 6

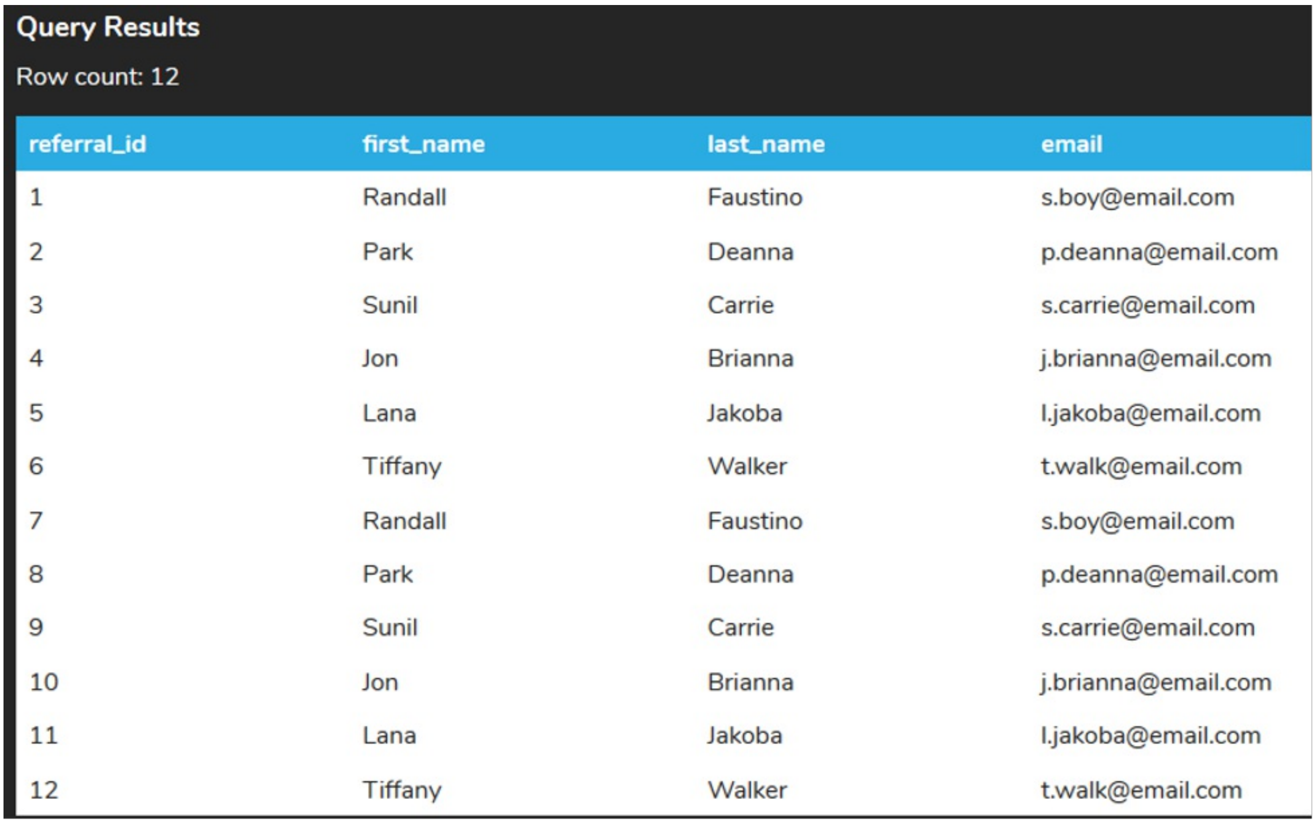
referral_id	first_name	last_name	email
1	Randall	Faustino	s.boy@email.com
2	Park	Deanna	p.deanna@email.com
3	Sunil	Carrie	s.carrie@email.com
4	Jon	Brianna	j.brianna@email.com
5	Lana	Jakoba	l.jakoba@email.com
6	Tiffany	Walker	t.walk@email.com

If we accidentally ran the exact statement again, it should run, as we did not have any constraints preventing our data from being inserted:


Query Results
Row count: 6

referral_id	first_name	last_name	email
7	Randall	Faustino	s.boy@email.com
8	Park	Deanna	p.deanna@email.com
9	Sunil	Carrie	s.carrie@email.com
10	Jon	Brianna	j.brianna@email.com
11	Lana	Jakoba	l.jakoba@email.com
12	Tiffany	Walker	t.walk@email.com

Notice that the referral_id moves up to 7 now. If we queried the entire table, we should see the records repeated:

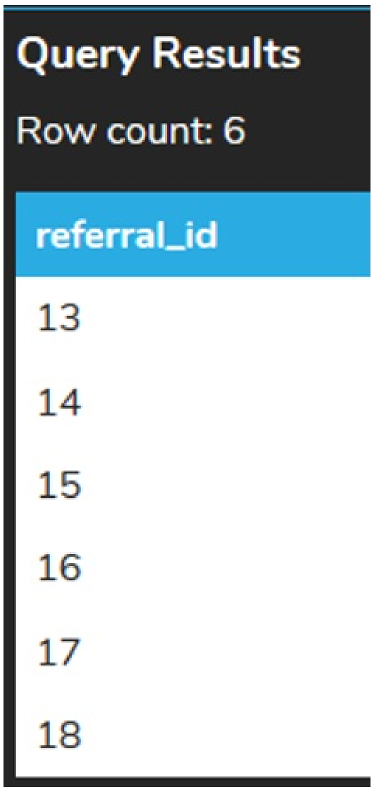

Query Results

Row count: 12

referral_id	first_name	last_name	email
1	Randall	Faustino	s.boy@email.com
2	Park	Deanna	p.deanna@email.com
3	Sunil	Carrie	s.carrie@email.com
4	Jon	Brianna	j.brianna@email.com
5	Lana	Jakoba	l.jakoba@email.com
6	Tiffany	Walker	t.walk@email.com
7	Randall	Faustino	s.boy@email.com
8	Park	Deanna	p.deanna@email.com
9	Sunil	Carrie	s.carrie@email.com
10	Jon	Brianna	j.brianna@email.com
11	Lana	Jakoba	l.jakoba@email.com
12	Tiffany	Walker	t.walk@email.com

Instead of having RETURNING *, we can also specify the columns to return. For example, we could use the referral_id as it is being auto-generated:

```
INSERT INTO referral (first_name,last_name,email)
VALUES ('Randall','Faustino','s.boy@email.com'),
('Park','Deanna','p.deanna@email.com'),
('Sunil','Carrie','s.carrie@email.com'),
('Jon','Brianna','j.brianna@email.com'),
('Lana','Jakoba','l.jakoba@email.com'),
('Tiffany','Walker','t.walk@email.com')
RETURNING referral_id;
```


Query Results

Row count: 6

referral_id
13
14
15
16
17
18

Video Transcription

[MUSIC PLAYING] Here we have a new table called referral in which has a referral ID, the first name, last name, and the email. If we trying to insert data into the table, we can certainly do so one room at a time. However, this is not always practical. There's lots of different ways to be able to simplify this process.

Let's take a look at the two different ways that we can actually approach this. The first way is inserting in each individual row, one at a time that you'll see here in this case of having the INSERT statement for each one of those, separated by a semicolon. The simpler approach to be able to kind of work through this is to be able to utilize one single statement. And what you'll do is actually separate the different values with a comma. And that just simplifies the entire statement significantly. That way you can be able to quickly scan through and view all the different items that you're going to be inserting in. And once you've actually inserted in, if you acquired a query from this table, you'll find that the same results will be in place as if you're just inserting one at a time.

[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 choices for which columns you want the query to provide.

SUMMARY

With the INSERT INTO statement, we can insert multiple rows into a table and use the RETURNING clause to return the inserted rows.

Source: Authored by Vincent Tran