INSERT to Add Queried Data

by Sophia

WHAT'S COVERED

In this lesson, you will use the SELECT statement with the INSERT INTO statement to add data from other tables, in two parts. Specifically, this lesson will cover:

1. Using INSERT INTO With SELECT

Using INSERT INTO and SELECT in the same statement can come in handy when we need to load data into a table from another table. For example, suppose you have a table containing customer contact information and you want the data from some of its columns to appear in a new table you are creating, or to be imported into matching columns in an existing table. You can create an INSERT INTO statement that uses an embedded SELECT clause to specify what data to insert. Anything that can be queried via SELECT can be subsequently added to an INSERT INTO statement.

The structure of the statement would look like this:

```
INSERT INTO <tablename> ( <column1>, <column2>, ...)
<SELECT ...>;
```

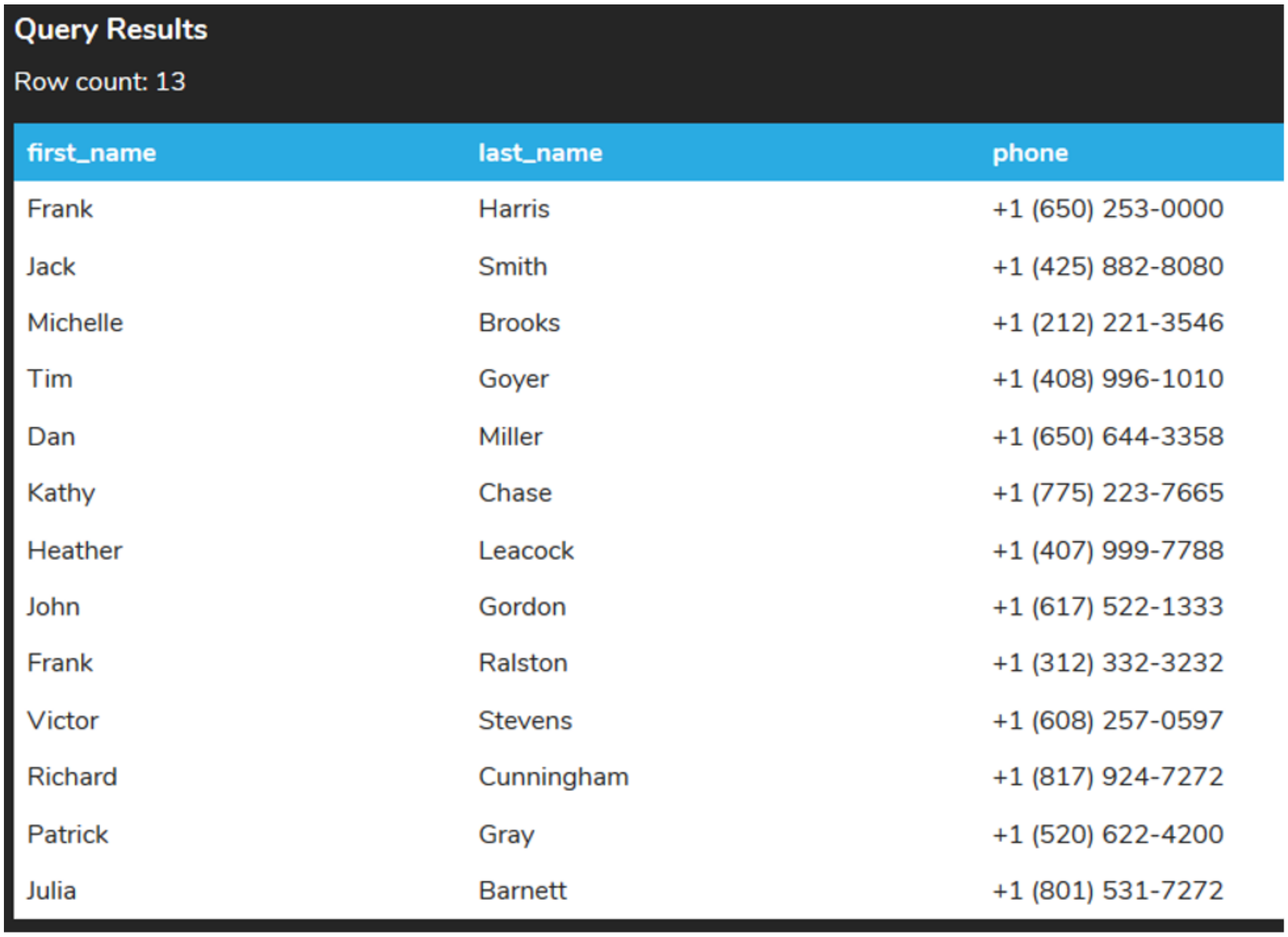
This is similar to the INSERT INTO statement you used in previous lessons: We want to include the columns and the order in which the query should return their values. In this structure, though, instead of the individual VALUE clauses, we provide a SELECT statement.

Let's say that we've created a contact table and would like to add the customer's first name, last name, and phone number. We would like to have only the customers that live in the country USA, as the table will be used for customer service representatives to make the calls. The first step is to create the table that we'll be using. We will want to ensure that the data types are the same and the sizes will be the same or larger to ensure we do not have any issues with truncated data.

```
CREATE TABLE contactUSA( contact_id SERIAL, first_name VARCHAR(40), last_name VARCHAR(40), phone VARCHAR(24) );
```

The next step is to create and validate the SELECT statement, which will look like this:

```
SELECT first_name, last_name, phone
FROM customer
WHERE country = 'USA';
```



Query Results

Row count: 13

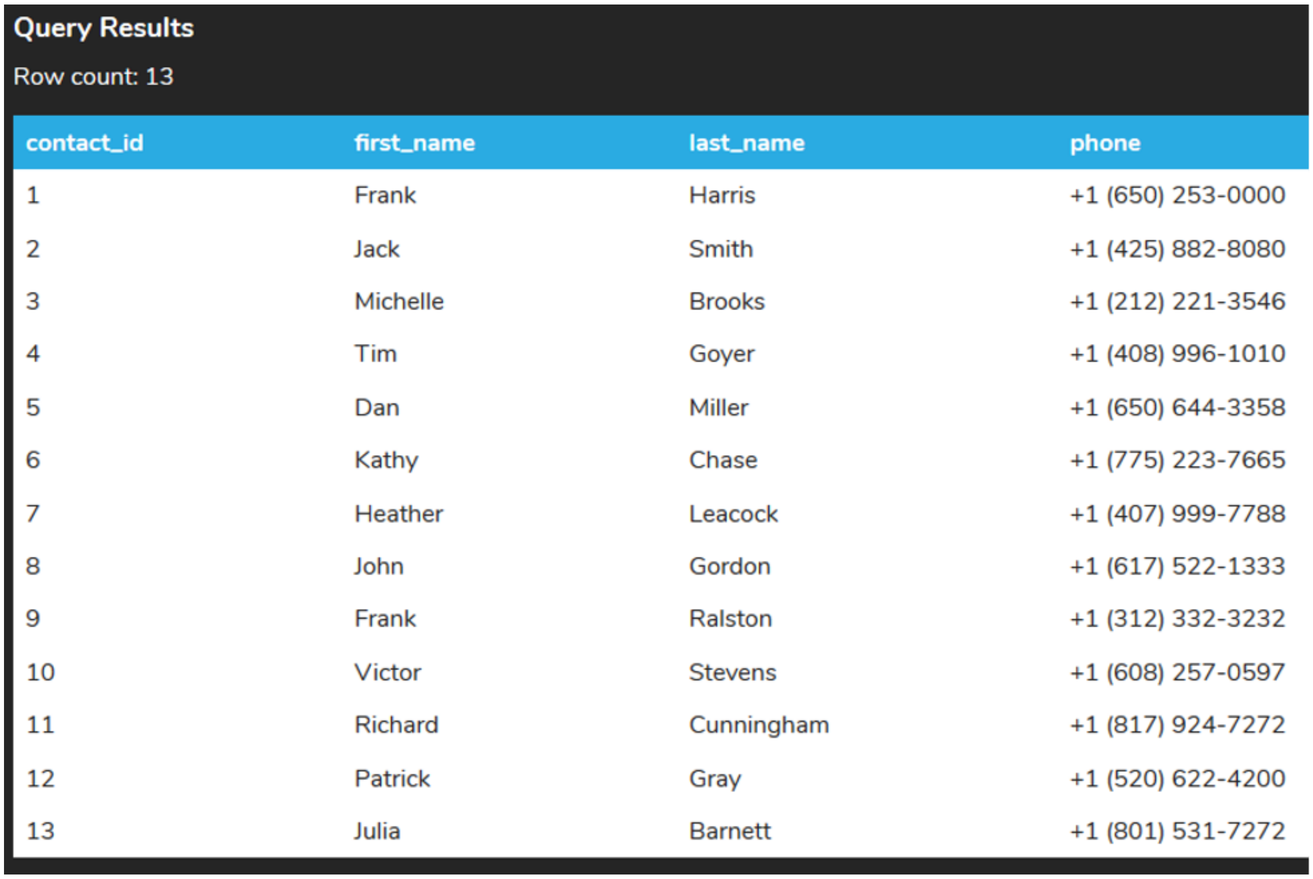
first_name	last_name	phone
Frank	Harris	+1 (650) 253-0000
Jack	Smith	+1 (425) 882-8080
Michelle	Brooks	+1 (212) 221-3546
Tim	Goyer	+1 (408) 996-1010
Dan	Miller	+1 (650) 644-3358
Kathy	Chase	+1 (775) 223-7665
Heather	Leacock	+1 (407) 999-7788
John	Gordon	+1 (617) 522-1333
Frank	Ralston	+1 (312) 332-3232
Victor	Stevens	+1 (608) 257-0597
Richard	Cunningham	+1 (817) 924-7272
Patrick	Gray	+1 (520) 622-4200
Julia	Barnett	+1 (801) 531-7272

Then we combine the two statements into a single statement like this:

```
INSERT INTO contactUSA (first_name, last_name, phone)
SELECT first_name, last_name, phone
FROM customer
WHERE country = 'USA';
```

Note that the first_name, last_name, and phone in the first line represent the columns from the contact table rather than from the customer table, although they are named the same. Now that this is complete, we can run a query on the contactUSA table to verify that the rows have been inserted:

```
SELECT *
FROM contactUSA;
```



Query Results

Row count: 13

contact_id	first_name	last_name	phone
1	Frank	Harris	+1 (650) 253-0000
2	Jack	Smith	+1 (425) 882-8080
3	Michelle	Brooks	+1 (212) 221-3546
4	Tim	Goyer	+1 (408) 996-1010
5	Dan	Miller	+1 (650) 644-3358
6	Kathy	Chase	+1 (775) 223-7665
7	Heather	Leacock	+1 (407) 999-7788
8	John	Gordon	+1 (617) 522-1333
9	Frank	Ralston	+1 (312) 332-3232
10	Victor	Stevens	+1 (608) 257-0597
11	Richard	Cunningham	+1 (817) 924-7272
12	Patrick	Gray	+1 (520) 622-4200
13	Julia	Barnett	+1 (801) 531-7272

There are many instances when this combination of INSERT INTO and SELECT can be useful. For example, if we're loading data from another table/database/file but we need to have the data formatted in a specific way rather than ALTER an existing table's properties, we can build the new table as we would want it and then copy the data in.

2. Working With a Summary Table

Another use for this INSERT INTO statement combined with SELECT is to create a summary table to get point-in-time data recorded. For example, suppose you want a summary table that has the invoice total and the number of invoices as of the current date, and you want today's date to be inserted.

Since this is point-in-time data, capturing those changes without a more complex set of criteria can be harder. We'll start by creating this table:

```
CREATE TABLE invoice_summary( summary_date date, all_total numeric, num_of_invoice integer );
```

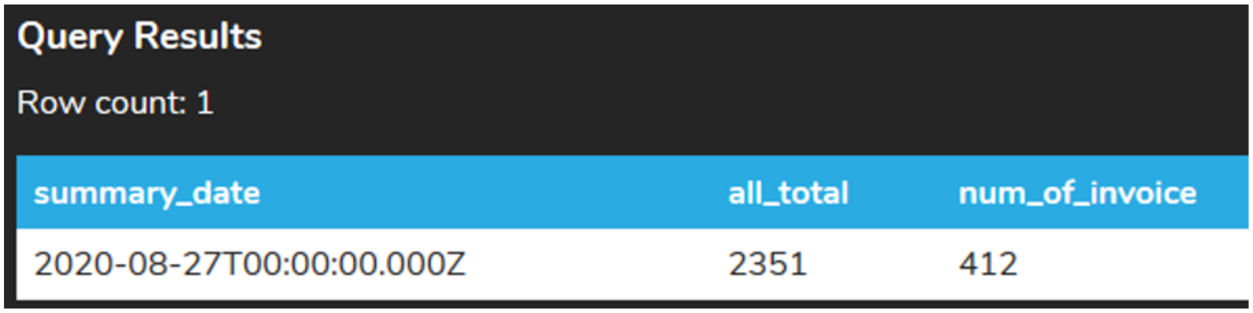
With the table created, we can now build the SELECT statement to generate the point-in-time data that includes the date:

```
SELECT now(), SUM(total), COUNT(invoice_id)
FROM invoice;
```

As a reminder, the now() function returns the current date/time. Putting this together with the SELECT statement, it would look like this:

```
INSERT INTO invoice_summary(summary_date,all_total,num_of_invoice)
SELECT now(), SUM(total), COUNT(invoice_id)
FROM invoice;
```

We can now query the invoice_summary table to see the new record that was added:


Query Results

Row count: 1

summary_date	all_total	num_of_invoice
2020-08-27T00:00:00.000Z	2351	412

WATCH

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

In this lesson, you learned how to insert new rows of data by querying them from another table. You do this by **using an INSERT INTO statement that uses a SELECT clause** to query the desired data rather than having individual VALUE clauses. Then, you saw an example of creating a **summary table** that gathers data from another table and adds a new column that contains the current date as gathered by the NOW() function.

Source: THIS TUTORIAL WAS AUTHORED BY DR. VINCENT TRAN, PHD (2020) AND Faithe Wempen (2024) FOR SOPHIA LEARNING. PLEASE SEE OUR TERMS OF USE.