LIMIT and OFFSET to Cap Results

by Sophia

WHAT'S COVERED

This tutorial explores using the LIMIT and OFFSET clauses to cap results in two parts:

1. LIMIT Clause
2. OFFSET Clause

1. LIMIT Clause

The LIMIT clause helps to constrain the number of rows that are returned by a query. It is an optional clause added to the end of the SELECT statement.

```
SELECT <columns>
FROM <tablename>
LIMIT <rowcount>;
```

For example, let's take a look at invoices by their total values in descending order:

```
SELECT *
FROM invoice
ORDER BY total DESC;
```

This ends up returning all of the rows (412 in this case), although in a larger database, this can take a long time to return all rows:

Query Results
Row count: 412

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

If we were only interested in looking at the top five, we could add the LIMIT clause so that the query only returns the top five rows:

```
SELECT *
FROM invoice
ORDER BY total DESC
LIMIT 5;
```

Query Results

Row count: 5

invoice_id	customer_id	invoice_date	billing_address	billing_city	billing_state	billing_country	billing_postal_code	total
404	6	2013-11-13T00:00:00.000Z	Rilská 3174/6	Prague		Czech Republic	14300	26
299	26	2012-08-05T00:00:00.000Z	2211 W Berry Street	Fort Worth	TX	USA	76110	24
96	45	2010-02-18T00:00:00.000Z	Erzsébet krt. 58.	Budapest		Hungary	H-1073	22
194	46	2011-04-28T00:00:00.000Z	3 Chatham Street	Dublin	Dublin	Ireland		22
89	7	2010-01-18T00:00:00.000Z	Rotenturmstraße 4, 1010 Innere Stadt	Vienne		Austria	1010	19

You will see the same top five rows as the prior query, but it returns much faster. Exploring data in small sets can be more efficient. Think about how you search results on a search engine. The results do not return every single result on the same page; it would simply be too slow to do so. Instead, they return only the top few results.

Let's look at another example of invoices that have the billing_country as Belgium in descending order:

```
SELECT *
FROM invoice
WHERE billing_country = 'Belgium'
ORDER BY total DESC;
```

This should return seven rows:

Query Results

Row count: 7

invoice_id	customer_id	invoice_date	billing_address	billing_city	billing_state	billing_country	billing_postal_code	total
187	8	2011-03-28T00:00:00.000Z	Grétrystraat 63	Brussels		Belgium	1000	14
242	8	2011-11-26T00:00:00.000Z	Grétrystraat 63	Brussels		Belgium	1000	9
3	8	2009-01-03T00:00:00.000Z	Grétrystraat 63	Brussels		Belgium	1000	6
394	8	2013-10-04T00:00:00.000Z	Grétrystraat 63	Brussels		Belgium	1000	4
176	8	2011-02-15T00:00:00.000Z	Grétrystraat 63	Brussels		Belgium	1000	2
371	8	2013-07-02T00:00:00.000Z	Grétrystraat 63	Brussels		Belgium	1000	2
55	8	2009-08-24T00:00:00.000Z	Grétrystraat 63	Brussels		Belgium	1000	1

If we had set the LIMIT to 10:

```
SELECT *
FROM invoice
WHERE billing_country = 'Belgium'
ORDER BY total DESC
LIMIT 10;
```

The result set would stay the same, with seven rows:

Query Results

Row count: 7

invoice_id	customer_id	invoice_date	billing_address	billing_city	billing_state	billing_country	billing_postal_code	total
187	8	2011-03-28T00:00:00.000Z	Grétrystraat 63	Brussels		Belgium	1000	14
242	8	2011-11-26T00:00:00.000Z	Grétrystraat 63	Brussels		Belgium	1000	9
3	8	2009-01-03T00:00:00.000Z	Grétrystraat 63	Brussels		Belgium	1000	6
394	8	2013-10-04T00:00:00.000Z	Grétrystraat 63	Brussels		Belgium	1000	4
176	8	2011-02-15T00:00:00.000Z	Grétrystraat 63	Brussels		Belgium	1000	2
371	8	2013-07-02T00:00:00.000Z	Grétrystraat 63	Brussels		Belgium	1000	2
55	8	2009-08-24T00:00:00.000Z	Grétrystraat 63	Brussels		Belgium	1000	1

2. OFFSET Clause

If we want to skip a number of rows when returning results, we can use the OFFSET clause. The OFFSET clause is placed after the LIMIT clause.

```
SELECT <columns>
FROM <tablename>
LIMIT <rowcount>
OFFSET <rowstoskip>;
```

By doing this, the statement will skip the first number of rows in the OFFSET and then return the number of rows based on the LIMIT. A search engine with multiple pages of content works in a very similar way.

Using our invoice example, perhaps we want to look through each set of invoices five lines at a time. To look at the next five rows, we would run the query using:

```
SELECT *
FROM invoice
ORDER BY total DESC
LIMIT 5
OFFSET 5;
```

Query Results

Row count: 5

invoice_id	customer_id	invoice_date	billing_address	billing_city	billing_state	billing_country	billing_postal_code	total
89	7	2010-01-18T00:00:00.000Z	Rotenturmstraße 4, 1010 Innere Stadt	Vienne		Austria	1010	19
88	57	2010-01-13T00:00:00.000Z	Calle Lira, 198	Santiago		Chile		18
313	43	2012-10-06T00:00:00.000Z	68, Rue Jouvence	Dijon		France	21000	17
306	5	2012-09-05T00:00:00.000Z	Klanova 9/506	Prague		Czech Republic	14700	17
208	4	2011-06-29T00:00:00.000Z	Ullevålsveien 14	Oslo		Norway	0171	16

If we wanted to look at the next five rows after that:

```
SELECT *
FROM invoice
ORDER BY total DESC
LIMIT 5
OFFSET 10;
```

Query Results

Row count: 5

invoice_id	customer_id	invoice_date	billing_address	billing_city	billing_state	billing_country	billing_postal_code	total
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
40	36	2009-06-15T00:00:00.000Z	Tauentzienstraße 8	Berlin		Germany	10789	14
82	28	2009-12-18T00:00:00.000Z	302 S 700 E	Salt Lake City	UT	USA	84102	14
68	11	2009-10-17T00:00:00.000Z	Av. Paulista, 2022	São Paulo	SP	Brazil	01310-200	14

Using the OFFSET and LIMIT clauses, we can jump to any subset of rows being returned within the SELECT statement.

Looking at our prior example of invoices in Belgium:

```
SELECT *
FROM invoice
WHERE billing_country = 'Belgium'
ORDER BY total DESC
LIMIT 10;
```

Query Results

Row count: 7

invoice_id	customer_id	invoice_date	billing_address	billing_city	billing_state	billing_country	billing_postal_code	total
187	8	2011-03-28T00:00:00.000Z	Grétrystraat 63	Brussels		Belgium	1000	14
242	8	2011-11-26T00:00:00.000Z	Grétrystraat 63	Brussels		Belgium	1000	9
3	8	2009-01-03T00:00:00.000Z	Grétrystraat 63	Brussels		Belgium	1000	6
394	8	2013-10-04T00:00:00.000Z	Grétrystraat 63	Brussels		Belgium	1000	4
176	8	2011-02-15T00:00:00.000Z	Grétrystraat 63	Brussels		Belgium	1000	2
371	8	2013-07-02T00:00:00.000Z	Grétrystraat 63	Brussels		Belgium	1000	2
55	8	2009-08-24T00:00:00.000Z	Grétrystraat 63	Brussels		Belgium	1000	1

Recall that the result set returned seven rows. But look at what happens if we add the OFFSET to 5.

```
SELECT *
FROM invoice
WHERE billing_country = 'Belgium'
ORDER BY total DESC
LIMIT 10
OFFSET 5;
```

Again, this would skip the first five rows and return the number of rows remaining, up to the LIMIT of 10:

Query Results

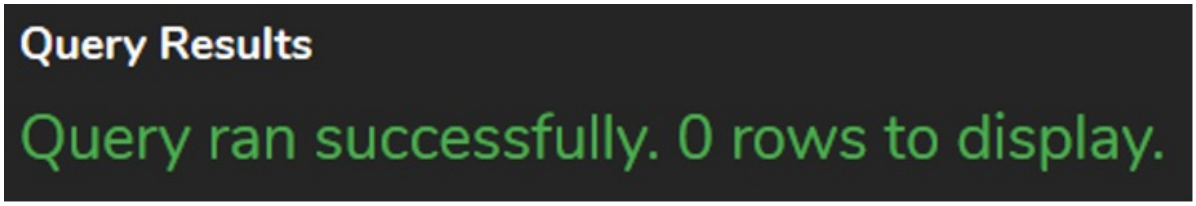
Row count: 2

invoice_id	customer_id	invoice_date	billing_address	billing_city	billing_state	billing_country	billing_postal_code	total
371	8	2013-07-02T00:00:00.000Z	Grétrystraat 63	Brussels		Belgium	1000	2
55	8	2009-08-24T00:00:00.000Z	Grétrystraat 63	Brussels		Belgium	1000	1

If we had added an OFFSET of 10, we would exceed the number of rows available to return:

```
SELECT *
FROM invoice
WHERE billing_country = 'Belgium'
ORDER BY total DESC
LIMIT 10
```

OFFSET 10;
So the query would simply return 0 rows:



Video Transcription

[MUSIC PLAYING] When we select from a table, we're returning all the rows with a table. As our tables get larger, this isn't always something that's feasible, being that the query would run for a long time. So what we can do is utilize limit and offset to be able to make some manipulations with the data so that we're not having to return all the data components all at once. So in this example here, if we're selecting from the customer, we're returning 59 rows. We can certainly scroll through and then take a look at every single row. However, this is certainly not viable as data sets get larger.

So what we can do is add a component here, which will be the limit keyword, and then the number of rows that we want to return. So in this case here, if we set it at five, we can go ahead and run this. And what this will do is just return the first five rows that meets whatever criteria there is. So in this case here, being that we're ordering and based on the customer ID, it's only going to return the first five.

We can make a change in this case here, utilizing the offset to be able to offset the results with the same limit parameter. So for example, in this case here, if we make an offset set to five, what this will do is that it'll skip the first five rows and then display the next five. So we're limiting it still to five. In this case here, it'll just make a manipulation to make that jump. If we change it to 10, it'll start at 11 in this case here for the customer ID and return the next five rows that would meet the criteria.

So one of the benefits in this case here with utilizing that option is that you can control all the different components of the offset and the limit without returning all the rows within a specific table. [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

The LIMIT and OFFSET clauses are used to cap records returned from SELECT statements.

Source: Authored by Vincent Tran