ORDER BY to Sort Data

by Sophia

ORDER BY to Sort Data

WHAT'S COVERED

This tutorial explores the ORDER BY clause within a SELECT statement to sort data based on columns in three parts:

1. Sorting By One Column
2. Ascending or Descending
3. Cascading Order Sequence

1. Sorting By One Column

When we simply query from a table, the result set is sorted by default based on the order of the inserted data. In most relational databases, a unique identifier is automatically generated through an auto-incremented value. In Postgres, this is called a serial. Since this value is automatically generated when data is inserted into a table, you will generally see the data sorted in this way. For example, if we query the invoice table, we will see it ordered automatically by the invoice_id.

```
SELECT *
FROM invoice;
```

invoice_id	customer_id	invoice_date	billing_address
1	2	2009-01-01T00:00:00.000Z	Theodor-Heuss-Straße 34
2	4	2009-01-02T00:00:00.000Z	Ullevålsveien 14
3	8	2009-01-03T00:00:00.000Z	Grétrystraat 63
4	14	2009-01-06T00:00:00.000Z	8210 111 ST NW
5	23	2009-01-11T00:00:00.000Z	69 Salem Street
6	37	2009-01-19T00:00:00.000Z	Berger Straße 10
7	38	2009-02-01T00:00:00.000Z	Barbarossastraße 19
8	40	2009-02-01T00:00:00.000Z	8, Rue Hanovre
9	42	2009-02-02T00:00:00.000Z	9, Place Louis Barthou
10	46	2009-02-03T00:00:00.000Z	3 Chatham Street

The ORDER BY clause is useful for when we want to list records in a specific order. For example, If we wanted to sort the result based on the billing_country, we can add an ORDER BY clause to the SELECT statement after the FROM clause:

```
SELECT *
FROM invoice
ORDER BY billing_country;
```

invoice_id	customer_id	invoice_date	billing_address	billing_city	billing_state	billing_country	billing_postal_code	total
403	56	2013-11-08T00:00:00.000Z	307 Macacha Güemes	Buenos Aires		Argentina	1106	9
348	56	2013-03-10T00:00:00.000Z	307 Macacha Güemes	Buenos Aires		Argentina	1106	14
119	56	2010-06-12T00:00:00.000Z	307 Macacha Güemes	Buenos Aires		Argentina	1106	2
142	56	2010-09-14T00:00:00.000Z	307 Macacha Güemes	Buenos Aires		Argentina	1106	4
164	56	2010-12-17T00:00:00.000Z	307 Macacha Güemes	Buenos Aires		Argentina	1106	6
216	56	2011-08-07T00:00:00.000Z	307 Macacha Güemes	Buenos Aires		Argentina	1106	1
337	56	2013-01-28T00:00:00.000Z	307 Macacha Güemes	Buenos Aires		Argentina	1106	2
118	55	2010-05-30T00:00:00.000Z	421 Bourke Street	Sidney	NSW	Australia	2010	1
239	55	2011-11-21T00:00:00.000Z	421 Bourke Street	Sidney	NSW	Australia	2010	2
66	55	2009-10-09T00:00:00.000Z	421 Bourke Street	Sidney	NSW	Australia	2010	6
250	55	2012-01-01T00:00:00.000Z	421 Bourke Street	Sidney	NSW	Australia	2010	14
44	55	2009-07-07T00:00:00.000Z	421 Bourke Street	Sidney	NSW	Australia	2010	4
21	55	2009-04-04T00:00:00.000Z	421 Bourke Street	Sidney	NSW	Australia	2010	2
305	55	2012-08-31T00:00:00.000Z	421 Bourke Street	Sidney	NSW	Australia	2010	9
89	7	2010-01-18T00:00:00.000Z	Rotenturmstraße 4, 1010 Innere Stadt	Vienne		Austria	1010	19

Notice that the data is now sorted based on the billing_country. It starts with Argentina for the first row, continues with all of the other rows that have the value of Argentina, followed by Australia, and so forth. Also notice that the invoice_id is no longer in order, as the results have been reordered based on the billing_country.

Let's look at the customer table and see what the results would look like without ordering the data:

```
SELECT customer_id, first_name, last_name
FROM customer;
```

customer_id	first_name	last_name
1	Luís	Gonçalves
2	Leonie	Köhler
3	François	Tremblay
4	Bjørn	Hansen
5	František	Wichterlová
6	Helena	Holý
7	Astrid	Gruber
8	Daan	Peeters
9	Kara	Nielsen

If we wanted to order it by the first_name, we can simply add the ORDER BY clause:

```
SELECT customer_id, first_name, last_name
FROM customer
ORDER BY first_name;
```

The result set in the customer table would now be sorted by the first_name column:

customer_id	first_name	last_name
32	Aaron	Mitchell
11	Alexandre	Rocha
7	Astrid	Gruber
4	Bjørn	Hansen
39	Camille	Bernard
8	Daan	Peeters
20	Dan	Miller
56	Diego	Gutiérrez
40	Dominique	Lefebvre
10	Eduardo	Martins
30	Edward	Francis

2. Ascending or Descending

By default, when we use the ORDER BY clause in a SELECT statement, it sorts the data in ascending order. However, there are instances where we may want to sort the data in descending order. For example, if we wanted to query the invoice table and display the result set based on the total, we could sort it like this:

```
SELECT invoice_id, customer_id, total
FROM invoice
ORDER BY total;
```

Query Results

Row count: 412

invoice_id	customer_id	total
174	5	1
384	24	1
167	26	1
62	46	1
265	27	1
286	23	1
328	15	1
160	47	1

Notice that this command returns 412 rows, with a lot of the initial rows starting with 1. What if we wanted to find the highest total first? We can add the keyword DESC to the ORDER BY clause after the column to sort it accordingly.

```
SELECT invoice_id, customer_id, total
FROM invoice
ORDER BY total DESC;
```

This would result in the following:

Query Results

Row count: 412

invoice_id	customer_id	total
404	6	26
299	26	24
194	46	22
96	45	22
89	7	19
201	25	19
88	57	18
313	43	17
306	5	17
103	24	16

You can also use ASC in place of DESC to display the result in ascending order, although this won't be any different than if you just listed the column without ASC added.

These two commands would result in the same result set, in the same order:

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

And

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

---

3. Cascading Order Sequence

We may want to sort the data by multiple parts in the same query. For example, think about listing customers in a phone directory. In that case, we would want to sort based on the last name, first name, and the company. We would do this in three stages. First, ORDER BY last_name. Then, within the matching last_name values, we

would ORDER BY first_name. Next, within the matching last_name and first_name values, we would ORDER BY the company. This type of multilevel ordered sequence is called the cascading order sequence. It can easily be created by listing the columns, separated by commas, after the ORDER BY clause.

```
SELECT customer_id, last_name, first_name, company
FROM customer
ORDER BY last_name, first_name, company;
```

Query Results

Row count: 59

customer_id	last_name	first_name	company
12	Almeida	Roberto	Riotur
28	Barnett	Julia	
39	Bernard	Camille	
18	Brooks	Michelle	
29	Brown	Robert	
21	Chase	Kathy	
26	Cunningham	Richard	
41	Dubois	Marc	

This cascading order sequence may not appear particularly useful for our data set, since we only have 59 customers in the table. However, with more customers, you will have many more identical first and last names, and multilevel sorting becomes more important.

Video Transcription

[MUSIC PLAYING] Another clause can be quite useful when it comes to SELECT statements is the ORDER BY by clause. It allows us to be able to sort database on the column sets that are being returned. For example, if we want to take the customer data here and then sort it base on the first name, we can enter in ORDER BY and then include in first_name. Go ahead and can Run. This will now return the result set entirely ordered based on the first name.

Now, we can also take the data and then order it based on multiple different criteria. So for example, if you had two errands and then we wanted to ensure that we also sort it based on last name, we can add it utilizing a comma, and then including in the next column associated with it. And utilizing that, it was sorted based on the first name first and then the last name if the first name was matching up inside.

The other option too is that you can also sort the data in descending order. By default, it is sorting it based on ascending order. So if we change this and turn in DESC, which stands for a descending, it'll now display it in descending order based on that first name. So it starts at the Wyatt, and then Victor, then Tim, and so forth.

[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 ORDER BY clauses.

SUMMARY

We can sort the result set in ascending order using the ORDER BY clause, or use DESC to sort it in descending order. We can also sort using the cascading order sequence by listing multiple columns separated by commas.