

586. Customer Placing the Largest Number of Orders [Previous](#)

June 9, 2017 | 23.8K views

★★★★☆

Average Rating: 4.61 / 10 votes

Query the **customer_number** from the **orders** table for the customer who has placed the largest number of orders.

It is guaranteed that exactly one customer will have placed more orders than any other customer.

The **orders** table is defined as follows:

Column	Type
order_number (PK)	int
customer_number	int
order_date	date
required_date	date
shipped_date	date
status	char(15)
comment	char(200)

Sample Input

order_number	customer_number	order_date	required_date	shipped_date	status
1	1	2017-04-09	2017-04-13	2017-04-12	Closed
2	2	2017-04-15	2017-04-20	2017-04-18	Closed
3	3	2017-04-16	2017-04-25	2017-04-20	Closed
4	3	2017-04-18	2017-04-28	2017-04-25	Closed

Sample Output

```
| customer_number |
|-----|
| 3              |
```

Explanation

The customer with number '3' has two orders, which is greater than either customer '1'.
So the result is customer number '3'.

Follow up: What if more than one customer have the largest number of orders, can you find all the customer number in this case?

Solution

Approach: Using **LTMTT** [Accented]

Algorithm

First, we can select the **customer number** and the according count of orders using **GROUP BY**

```
SELECT
    customer_number, COUNT(*)
FROM
    orders
GROUP BY customer_number
```

customer_number	COUNT(*)
1	1
2	1
3	2

Then, the **customer number** of first record is the result after sorting them by order count descending.

customer_number	COUNT(*)
3	2

In MySQL, the **LIMIT** clause can be used to constrain the number of rows returned by the **SELECT** statement. It takes one or two nonnegative numeric arguments, the first of which specifies the offset of the first row to return, and the second specifies the maximum number of rows to return. The offset of the initial row is 0 (not 1).

It can be used with only one argument, which specifies the number of rows to return from the beginning of the result set. So **LIMIT 1** will return the first record.


MySQL

```
SELECT
    customer_number
FROM
    orders
GROUP BY customer_number
ORDER BY COUNT(*) DESC
LIMIT 1
;
```


Rate this article: ★★★★★

[Previous](#)
[Next](#)


Comments: 14 Sort By ▼



Type comment here... (Markdown is supported)

 Preview

Post

 **rabbitjii** ★ 102 · 🌐 March 2, 2018 3:02 AM

Use ALL:

```
select customer_number from orders
group by customer_number
having count(customer_number) >= all
```

25 🗑️ 🔄 | Share | Reply [Read More](#)

 **gavinguoja** ★ 5 · October 24, 2017 2:16 AM

Follow up: What if more than one customer have the largest number of orders, can you find all the customer_number in this case?

Do not use triangle join, because it is bad scalability and performance

5 ▲ ▼ |  Share |  Reply

[Read More](#)

[SHOW 1 REPLY](#)

[drfluid](#) ★ 24 🕒 October 26, 2018 2:50 PM


solution without using **LIMIT**

```
select customer_number from orders
group by customer_number
having count(order_number) >
```

[Read More](#)

4 📶 | [Share](#) | [Reply](#)





[SHOW 1 REPLY](#)

 **azimbabu** ★ 108 ○ October 15, 2017 1:06 PM


It can handle the case where more than one customer can have maximum number of orders.

```
SELECT customer_number
FROM orders
GROUP BY customer_number
```

[Read More](#)


4   |  Share |  Reply

[SHOW 1 REPLY](#)

 **tys9227** ★ 27 🕒 November 4, 2018 3:14 PM


I think ORDER BY COUNT(order_number) is more reasonable because it was not mentioned that order_number is primary key. so order_num column could have NULLs in it but COUNT(*) would include those NULLs.

2 🗑️ 🔄 📄 Share 🗨️ Reply

 **vemuri1989** ★ 2 · May 27, 2018 2:09 AM

```
select customer_number from(
select customer_number,DENSE_RANK() over (order by cnt desc) as a from
(
select customer_number, count(customer_number) as cnt from orders group by customer_number
order by cnt desc
```

2 · | Share | Reply [Read More](#)


 **pxj5333** ★ 9 🕒 July 9, 2017 8:17 AM

I think this can handle more than one numbers occurring the same number of max times:

```
SELECT customer_number FROM orders
GROUP BY customer_number
having count(distinct order_number) IN
(SELECT MAX(count) FROM (select count(distinct order number) as counts FROM orders GROUP BY
```





[Read More](#)


👍 🗑️ 📄 Share 📧 Reply

 **sweta31jan** ★ 0 · January 20, 2019 2:27 PM

```
select customer_number from
( select max(num),customer_number from
(SELECT count(distinct order_number) as num, customer_number FROM orders group by
customer_number)
)
```


[Read More](#)

0   |  Share |  Reply





 **Irongoddess** ★ 0 · January 15, 2019 8:36 AM

```
select customer_number
from orders
group by customer_number
order by count(order_number) desc
limit 1;
```

[Read More](#)

 [gadianov](#) ★ 78 · November 13, 2018 3:43 PM

```
select customer_number from
(select customer_number, count(customer_number) as cnt from orders
group by customer_number order by cnt desc) as t limit 1;
```

0    Share  Reply [Read More](#)