窗口函数

1. 数据准备

```
-- 1. 建表语句
CREATE TABLE `order_tab` (
  `order_id` int(10) NOT NULL COMMENT '订单ID',
  `user_no` varchar(22) DEFAULT '' COMMENT '用户唯一标识',
  `amount` int(10) DEFAULT NULL COMMENT '金额',
  `create_date` timestamp NOT NULL DEFAULT CURRENT_TIMESTAMP ON UPDATE CURRENT_TIMESTAMP COMMENT '订单时间',
  PRIMARY KEY (`order_id`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4;
-- 2.插入数据语句
insert into `order_tab` (`order_id`, `user_no`, `amount`, `create_date`) values('0','001','100','2021-01-01 00:00:00');
insert into `order_tab` (`order_id`, `user_no`, `amount`, `create_date`) values('2','001','300','2021-01-02 00:00:00');
insert into `order_tab` (`order_id`, `user_no`, `amount`, `create_date`) values('3','001','500','2021-01-02 00:00:00');
insert into `order_tab` (`order_id`, `user_no`, `amount`, `create_date`) values('4','001','800','2021-01-03 00:00:00');
insert into `order_tab` (`order_id`, `user_no`, `amount`, `create_date`) values('5','001','900','2021-01-04 00:00:00');
insert into `order_tab` (`order_id`, `user_no`, `amount`, `create_date`) values('6','002','500','2021-01-03 00:00:00');
insert into `order_tab` (`order_id`, `user_no`, `amount`, `create_date`) values('7','002','600','2021-01-04 00:00:00');
insert into `order_tab` (`order_id`, `user_no`, `amount`, `create_date`) values('8','002','300','2021-01-10 00:00:00');
insert into `order_tab` (`order_id`, `user_no`, `amount`, `create_date`) values('9','002','800','2021-01-16 00:00:00');
insert into `order_tab` (`order_id`, `user_no`, `amount`, `create_date`) values('10','002','800','2021-01-22 00:00:00');
```

2.窗口函数的应用

1. 语法

```
<窗口函数> over(
                partition by <用于分组的列名>
                order by <用于排序的列名>
)
```

2. 举例：查询每个用户订单金额最高的前三个订单：

```
SELECT * FROM order_tab;

-- 1.把不同的用户根据金额（字段amount）降序排列得到一个新表格;
SELECT row_number() over(
              PARTITION by user_no
```

```
            ORDER BY amount DESC
) AS row_num,
    order_id,
    user_no,
    amount,
    create_date
    FROM order_tab
-- 注意：
/*row_number() over(
              PARTITION by user_no
            ORDER BY amount DESC
) AS row_num
以上是窗口函数，语法：
<窗口函数> over(
              partition by <用于分组的列名>
              order by <用于排序的列名>
)
row_number() ：给重新分组的新表重新排序；以上的意思是根据 user_no 分组之后每组再按照 amount 字段从高到低进行排序，把每个 user_no
对应的数据从1开始重新排序(这个功能特别好用)
*/
```

信息 结果1 概况 状态

row_num	order_id	user_no	amount	create_date
1	5	001	900	2021-01-04 00:00:00
2	4	001	800	2021-01-03 00:00:00
3	3	001	500	2021-01-02 00:00:00
4	2	001	300	2021-01-02 00:00:00
5	0	001	100	2021-01-01 00:00:00
1	9	002	800	2021-01-16 00:00:00
2	10	002	800	2021-01-22 00:00:00
3	7	002	600	2021-01-04 00:00:00
4	6	002	500	2021-01-03 00:00:00
5	8	002	300	2021-01-10 00:00:00

3. 代码

```
-- 2.因为每个用户的前三行，即为“每个用户订单金额最高的前三个订单”

SELECT *
FROM
    (  SELECT
            row_number() OVER (
                PARTITION BY user_no
                ORDER BY
                    amount DESC
            ) AS row_num,
            order_id,
            user_no,
            amount,
            create_date
```

```
        FROM
            order_tab
    ) t
WHERE
    row_num <= 3;
```

结果：

信息 结果1 概况 状态

row_num	order_id	user_no	amount	create_date
1	5	001	900	2021-01-04 00:00:00
2	4	001	800	2021-01-03 00:00:00
3	3	001	500	2021-01-02 00:00:00
1	9	002	800	2021-01-16 00:00:00
2	10	002	800	2021-01-22 00:00:00
3	7	002	600	2021-01-04 00:00:00

4. 表中的数据

```
SELECT * FROM order_tab;
```

信息 结果1 概况 状态

order_id	user_no	amount	create_date
0	001	100	2021-01-01 00:00:00
2	001	300	2021-01-02 00:00:00
3	001	500	2021-01-02 00:00:00
4	001	800	2021-01-03 00:00:00
5	001	900	2021-01-04 00:00:00
6	002	500	2021-01-03 00:00:00
7	002	600	2021-01-04 00:00:00
8	002	300	2021-01-10 00:00:00
9	002	800	2021-01-16 00:00:00
10	002	800	2021-01-22 00:00:00

窗口函数简单的总结：

根据以上例子，窗口函数比分组更加的强大，假如使用分组的话，根据user_no 分组，第二列只能写聚合函数，不能显示该用户所有的信息，后续加深学习后还会进行更新，感觉不错加个关注哈。

3. 注意

- 窗口函数发生在 ORDER BY, LIMIT, 和 SELECT DISTINCT 之前。

- 有点不同，它返回检索到的行数的计数，无论它们是否包含 `NULL` 值。

  COUNT(expr):

  返回由语句检索的行中非 `NULL` 值 的数量的计数。结果是一个 值。

- `IF(*expr1*,*expr2*,*expr3*)`

如果 `expr1` 是 `TRUE` (`*expr1* <> 0` 和 `*expr1* IS NOT NULL`)，则 `IF()` 返回 `expr2`。否则，它返回 `expr3`.

- 注意：使用窗口函数使用聚合函数 COUNT(*)时，不建议PARTITION BY 字段1 和 ORDER BY 字段2 使用不一样的字段，当 字段2 有重复时，则会计数就和 窗口函数 row_number() 不一致；因此建议使用 row_number() 对不同区行数从小到大计数。

```sql
SELECT *,
               COUNT(*) over(PARTITION BY user_no ORDER BY user_no) user_no_times,
               COUNT(user_no) over(PARTITION BY user_no ORDER BY amount) amount_no_times,
               row_number() over(PARTITION BY user_no ORDER BY amount) row_num
               FROM order_tab
```

查询创建工具 查询编辑器

```sql
SELECT *,
       COUNT(*) over(PARTITION BY user_no ORDER BY user_no) user_no_times,
       COUNT(user_no) over(PARTITION BY user_no ORDER BY amount) amount_no_times,
       row_number() over(PARTITION BY user_no ORDER BY amount) row_num
       FROM order_tab
```

信息 结果1 概况 状态

order_id	user_no	amount	create_date	user_no_times	amount_no_times	row_num
0	001	100	2021-01-01 00:00:00	5	1	1
2	001	300	2021-01-02 00:00:00	5	2	2
3	001	500	2021-01-02 00:00:00	5	3	3
4	001	800	2021-01-03 00:00:00	5	4	4
5	001	900	2021-01-04 00:00:00	5	5	5
8	002	300	2021-01-10 00:00:00	6	1	1
6	002	500	2021-01-03 00:00:00	6	2	2
7	002	600	2021-01-04 00:00:00	6	3	3
9	002	800	2021-01-15 00:00:00	6	6	4
10	002	800	2021-01-22 00:00:00	6	6	5
12	002	800	2022-03-01 00:00:00	6	6	6
11	003	1100	2022-03-01 00:00:00	1	1	1

amount一样的，计数就出现了问题