## 1、定义

窗口函数的应用。窗口可以理解为记录集合，窗口函数就是在满足某种条件的记录集合上执行的特殊函数。

## 2、语法格式

函数名(字段名) over(partition by <要分列的组> order by <要排序的列> rows between <数据范围>)

```
 rows between 2 preceding and current row # 取本行和前面两行

 rows between unbounded preceding and current row # 取本行和之前所有的行

 rows between current row and unbounded following # 取本行和之后所有的行

 rows between 3 preceding and 1 following # 从前面三行到下面一行，总共五行
```

## 3、分类

### 聚合类

**聚合窗口函数与普通聚合函数的区别**：

- 普通场景下的聚合函数是将多条记录聚合为一条**(多到一)**;
- 窗口函数是每条记录都会执行此函数，有几条记录执行完还是几条**(多到多)**。

**累计求和：** sum()over()

```sql
 -- 查询出2019年每月的支付总额和当年累积支付总额
 SELECT
     a.pay_amount,
     a.mon,
     sum( a.pay_amount ) over ( ORDER BY a.mon ) as
sum_amount
 FROM
     (
     SELECT
         SUM( u.pay_amount ) AS pay_amount,
         MONTH ( u.pay_time ) AS mon
     FROM
         user_trade u
     WHERE
         YEAR ( u.pay_time ) = '2019'
     GROUP BY
         mon
     ORDER BY
         mon
     ) a

 -- 查询出2018-2019年每月的支付总额和当年累积支付总额
 SELECT
     t.amount,
     t.ye,
     t.mon,
     sum( t.amount ) over ( PARTITION BY t.ye ORDER
BY t.mon )
 FROM
     (
     SELECT MONTH
         ( u.pay_time ) AS mon,
         YEAR ( u.pay_time ) AS ye,
         sum( u.pay_amount ) AS amount
     FROM
         user_trade u
     WHERE
```

```
        YEAR ( u.pay_time ) IN ( '2018', '2019' )
    GROUP BY
        ye,
        mon
    ORDER BY
        ye,
        mon ASC
    ) t
```

**移动平均：**avg() over()

```
 -- 需求3：查询出2019年每个月的近三月移动平均支付金额
 SELECT
     a.mon,
     a.pay_amount,
     avg( a.pay_amount ) over ( ORDER BY a.mon rows
BETWEEN 2 preceding AND current ROW ) AS avg_amount

 FROM
     (
     SELECT
         SUM( u.pay_amount ) AS pay_amount,
         MONTH ( u.pay_time ) AS mon
     FROM
         user_trade u
     WHERE
         YEAR ( u.pay_time ) = '2019'
     GROUP BY
         mon
     ORDER BY
     mon
     ) a
```

**最大/最小值：**max()/min() over()

```
 -- 需求4: 查询出每四个月的最大月总支付金额
 SELECT
     a.mon,
     a.pay_amount,
     max( a.pay_amount ) over ( ORDER BY a.mon rows
BETWEEN 3 preceding AND current ROW ) AS max_amount

 FROM
     (
     SELECT
         SUBSTRING( a.pay_time, 1, 7 ) AS mon,
         sum( a.pay_amount ) AS pay_amount
     FROM
         user_trade a
     GROUP BY
     SUBSTRING( a.pay_time, 1, 7 )
     ) a;
```

## 排序类

**row_number()、rank() 和dense_rank() 三种排序函数的区别：**

- row_number：每一行记录生成一个序号，依次排序且不会重复。1234...
- rank：跳跃排序，生成的序号有可能不连续。1134..
- dense_rank：在生成序号时是连续的。1123...

```
 -- 需求5: 2020年1月，购买商品品类数的用户排名
SELECT
  a.user_name,
  count( DISTINCT a.goods_category ) AS cat_num,
       ROW_NUMBER() over (

    ORDER BY
      count( DISTINCT a.goods_category )) AS
rank1,
      RANK() over (

    ORDER BY
      count( DISTINCT a.goods_category )) AS
rank2,

      DENSE_RANK() over (

    ORDER BY
      count( DISTINCT a.goods_category )) AS
rank3
FROM
      user_trade a
WHERE
      SUBSTRING( a.pay_time, 1, 7 ) = '2020-01'
GROUP BY
      a.user_name;
```