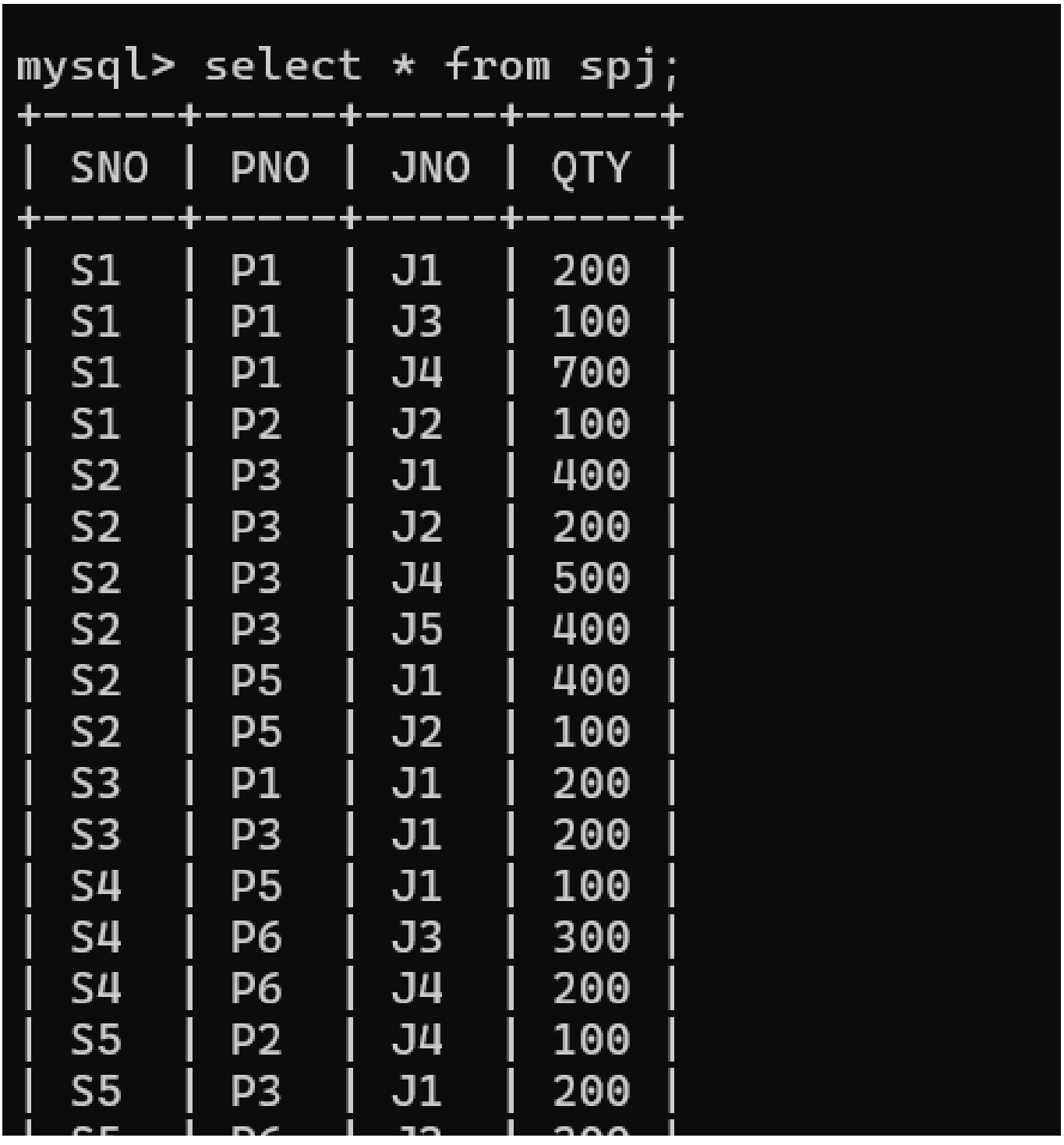
我们就以书上的**spj**表 或者 **sc**表 为例

DQl

- *消除重复元素(distinct)*

语法:

```SQL
SELECT DISTINCT 列名, ... FROM 表名;
```



```
mysql> select * from spj;
+------+------+------+------+
| SNO  | PNO  | JNO  | QTY  |
+------+------+------+------+
| S1   | P1   | J1   | 200  |
| S1   | P1   | J3   | 100  |
| S1   | P1   | J4   | 700  |
| S1   | P2   | J2   | 100  |
| S2   | P3   | J1   | 400  |
| S2   | P3   | J2   | 200  |
| S2   | P3   | J4   | 500  |
| S2   | P3   | J5   | 400  |
| S2   | P5   | J1   | 400  |
| S2   | P5   | J2   | 100  |
| S3   | P1   | J1   | 200  |
| S3   | P3   | J1   | 200  |
| S4   | P5   | J1   | 100  |
| S4   | P6   | J3   | 300  |
| S4   | P6   | J4   | 200  |
| S5   | P2   | J4   | 100  |
| S5   | P3   | J1   | 200  |
```

```
| S5   | P6   | J2   | 200  |
| S5   | P6   | J4   | 500  |
+------+------+------+------+
19 rows in set (0.00 sec)

mysql> select DISTINCT  SNO from spj;
+------+
| SNO  |
+------+
| S1   |
| S2   |
| S3   |
| S4   |
| S5   |
+------+
5 rows in set (0.00 sec)
```

注意：`distinct` 放在2个字段前，是*2个字段组合后出现重复*才去重，放在第一个后会报错

```
+------+------+
| SNO  | PNO  |
+------+------+
| S1   | P1   |
| S1   | P1   |
| S1   | P1   |
| S1   | P2   |
| S2   | P3   |
| S2   | P3   |
| S2   | P3   |
| S2   | P3   |
| S2   | P5   |
| S2   | P5   |
```

```
| S2   | P3   |
| S3   | P1   |
| S3   | P3   |
| S4   | P5   |
| S4   | P6   |
| S4   | P6   |
| S5   | P2   |
| S5   | P3   |
| S5   | P6   |
| S5   | P6   |
+------+------+
19 rows in set (0.00 sec)

mysql> select DISTINCT SNO,PNO from spj;
+------+------+
| SNO  | PNO  |
+------+------+
| S1   | P1   |
| S1   | P2   |
| S2   | P3   |
| S2   | P5   |
| S3   | P1   |
| S3   | P3   |
| S4   | P5   |
| S4   | P6   |
| S5   | P2   |
| S5   | P3   |
| S5   | P6   |
+------+------+
```

- *算术运算符（+，-，*，/）*

1.算术运算符的使用范围 ：

对 number 型数据可以使用算术运算符（+，-，* ，/）对数据进行操作；对date型数据可以使用部分算术运算符（+，-）对数据进行操作。

2.算术运算符的优先级 :与数学中运算相同

例子： 查询 PNO 以及 对应的WEIGHT * 10

```
mysql> select * from p;
+-----+------------+-------+--------+
| PNO | PNAME      | COLOR | WEIGHT |
+-----+------------+-------+--------+
| P1  | 螺母       | 红    |     12 |
| P2  | 螺栓       | 绿    |     17 |
| P3  | 螺丝刀     | 蓝    |     14 |
| P4  | 螺丝刀     | 红    |     14 |
| P5  | 凸轮       | 蓝    |     40 |
| P6  | 齿轮       | 红    |     30 |
+-----+------------+-------+--------+
6 rows in set (0.00 sec)

mysql> select PNO,WEIGHT * 10 from p;
+-----+--------------+
| PNO | WEIGHT * 10 |
+-----+--------------+
| P1  |          120 |
| P2  |          170 |
| P3  |          140 |
| P4  |          140 |
| P5  |          400 |
| P6  |          300 |
+-----+--------------+
6 rows in set (0.00 sec)
```

- *设置别名（AS）*

  作用
  1）改变列的标题头；
  2）作为计算结果的含义；
  3）作为列的别名；
  4）如果别名使用特殊字符（强烈不建议使用特殊字符），或是强制大小写或有空格时都需要加单引号。

语法：

```SQL
// 第一种
SELECT 列名 AS 别名 FROM 表名 [WHERE];
// 第二种
SELECT 列名 别名 FROM 表名 [WHERE];
```

例子：使用两种方式：查询出P中的PNO，并换一个别名为 PID

```
mysql> select * from p;
+-----+------------+-------+--------+
| PNO | PNAME      | COLOR | WEIGHT |
+-----+------------+-------+--------+
| P1  | 螺母       | 红    |     12 |
| P2  | 螺栓       | 绿    |     17 |
| P3  | 螺丝刀     | 蓝    |     14 |
| P4  | 螺丝刀     | 红    |     14 |
| P5  | 凸轮       | 蓝    |     40 |
| P6  | 齿轮       | 红    |     30 |
+-----+------------+-------+--------+
6 rows in set (0.00 sec)

mysql> select PNO PID from p;
+-----+
| PID |
+-----+
```

```
| P1   |
| P2   |
| P3   |
| P4   |
| P5   |
| P6   |
+------+
6 rows in set (0.00 sec)
```

```
mysql> select PNO AS PID from p;
+------+
| PID  |
+------+
| P1   |
| P2   |
| P3   |
| P4   |
| P5   |
| P6   |
+------+
6 rows in set (0.00 sec)
```

- *按格式输出（**CONCAT**）*

为了方便用户浏览查询结果数据，有时需要设置查询结果的显示格式，可以使用CONCAT函数来连接字符串。

语法：

```SQL
CONCAT(字符串1, 字符串2, ...)
```

例子:将PNO零件的名称 颜色格式打印出来

```
mysql> select CONCAT(PNO, "零件名为: ",PNAME, " 零件的颜色为: ", COLOR) from p;
+-----------------------------------------------------------------------+
| CONCAT(PNO, "零件名为: ",PNAME, " 零件的颜色为: ", COLOR)             |
+-----------------------------------------------------------------------+
| P1零件名为: 螺母 零件的颜色为: 红                                     |
| P2零件名为: 螺栓 零件的颜色为: 绿                                     |
| P3零件名为: 螺丝刀 零件的颜色为: 蓝                                   |
| P4零件名为: 螺丝刀 零件的颜色为: 红                                   |
| P5零件名为: 凸轮 零件的颜色为: 蓝                                     |
| P6零件名为: 齿轮 零件的颜色为: 红                                     |
+-----------------------------------------------------------------------+
6 rows in set (0.00 sec)
```

还支持转义符:

```
mysql> select CONCAT(PNO, "零件名为: \t",PNAME, " \t零件的颜色为: ", COLOR) from p;
+-----------------------------------------------------------------------+
| CONCAT(PNO, "零件名为: \t",PNAME, " \t零件的颜色为: ", COLOR)         |
+-----------------------------------------------------------------------+
| P1零件名为:    螺母    零件的颜色为: 红                                |
| P2零件名为:    螺栓    零件的颜色为: 绿                                |
| P3零件名为:    螺丝刀  零件的颜色为: 蓝                                |
| P4零件名为:    螺丝刀  零件的颜色为: 红                                |
| P5零件名为:    凸轮    零件的颜色为: 蓝                                |
| P6零件名为:    齿轮    零件的颜色为: 红                                |
+-----------------------------------------------------------------------+
6 rows in set (0.00 sec)
```

- *过滤查询（**WHERE**）*

可以支持嵌套查询
使用WHERE子句限定返回的记录。

语法:

```SQL
SELECT <selectList> FROM 表名 WHERE 条件;
```

注意：WHERE子句在FROM子句之后。

- *比较运算符（=, >, > =, <, < =, ! = ）*

  不等于： <> 等价 ! =;
  查询 WEIGHT > 20 的零件的全部信息

```
mysql> select * from p;
+-----+-----------+-------+--------+
```

```
| PNO | PNAME      | COLOR | WEIGHT |
+-----+------------+-------+--------+
| P1  | 螺母       | 红    |     12 |
| P2  | 螺栓       | 绿    |     17 |
| P3  | 螺丝刀     | 蓝    |     14 |
| P4  | 螺丝刀     | 红    |     14 |
| P5  | 凸轮       | 蓝    |     40 |
| P6  | 齿轮       | 红    |     30 |
+-----+------------+-------+--------+
6 rows in set (0.00 sec)

mysql> select * from p where WEIGHT > 20;
+-----+--------+-------+--------+
| PNO | PNAME  | COLOR | WEIGHT |
+-----+--------+-------+--------+
| P5  | 凸轮   | 蓝    |     40 |
| P6  | 齿轮   | 红    |     30 |
+-----+--------+-------+--------+
2 rows in set (0.00 sec)
```

*两种方式：查询**WEIGHT** 不等于 **14**的零件信息*

```
mysql> select * from p where WEIGHT <> 14;
+-----+--------+-------+--------+
| PNO | PNAME  | COLOR | WEIGHT |
+-----+--------+-------+--------+
| P1  | 螺母   | 红    |     12 |
| P2  | 螺栓   | 绿    |     17 |
| P5  | 凸轮   | 蓝    |     40 |
| P6  | 齿轮   | 红    |     30 |
+-----+--------+-------+--------+
4 rows in set (0.00 sec)
```

```
mysql> select * from p where WEIGHT != 14;
```

```
+-----+--------+-------+--------+
| PNO | PNAME  | COLOR | WEIGHT |
+-----+--------+-------+--------+
| P1  | 螺母   | 红    |     12 |
| P2  | 螺栓   | 绿    |     17 |
| P5  | 凸轮   | 蓝    |     40 |
| P6  | 齿轮   | 红    |     30 |
+-----+--------+-------+--------+
4 rows in set (0.00 sec)
```

- *逻辑运算符（AND、OR、NOT）*

AND：如果组合的田间都是 true，返回true；
OR：如果组合的条件之一是true，返回true；
NOT：如果给出的条件是false，返回true；如果给出的条件是true，则返回false。

1.查询零件WEIGHT > 20并且 WEIGHT < 40的所有零件信息

```
mysql> select * from p where WEIGHT > 20 and WEIGHT < 40;
+-----+--------+-------+--------+
| PNO | PNAME  | COLOR | WEIGHT |
+-----+--------+-------+--------+
| P6  | 齿轮   | 红    |     30 |
+-----+--------+-------+--------+
1 row in set (0.00 sec)
```

2.查询零件WEIGHT > 17 或者 WEIGHT < 14的所有零件

```
mysql> select * from p where WEIGHT > 17 OR WEIGHT < 14;
+-----+--------+-------+--------+
| PNO | PNAME  | COLOR | WEIGHT |
+-----+--------+-------+--------+
| P1  | 螺母   | 红    |     12 |
| P5  | 凸轮   | 蓝    |     40 |
| P6  | 齿轮   | 红    |     30 |
+-----+--------+-------+--------+
3 rows in set (0.00 sec)
```

3.查询零件WEIGHT 不等于 14的所有零件信息

```
mysql> select * from p where NOT WEIGHT = 14;
```

```
+-----+--------+-------+--------+
| PNO | PNAME  | COLOR | WEIGHT |
+-----+--------+-------+--------+
| P1  | 螺母   | 红    |     12 |
| P2  | 螺栓   | 绿    |     17 |
| P5  | 凸轮   | 蓝    |     40 |
| P6  | 齿轮   | 红    |     30 |
+-----+--------+-------+--------+
4 rows in set (0.00 sec)
```

- *范围和集合（BETWEEN AND）*

范围匹配：`BETWEEN AND` 运算符，一般使用在数字类型的范围上。但对于字符数据和日期类型同样可用。
注意：`BETWEE AND` 使用的是闭区间。

语法：

```SQL
// 使用的是闭区间，也就是包括minValue 和 maxValue
WHERE 列名 BETWEEN minValue AND maxValue;
```

例子：查询 `WEIGHT` 不在`[20,30]`之间的所有零件信息

```
mysql> select * from p where not WEIGHT BETWEEN 20 AND 30;
+-----+-----------+-------+--------+
| PNO | PNAME     | COLOR | WEIGHT |
+-----+-----------+-------+--------+
| P1  | 螺母      | 红    |     12 |
| P2  | 螺栓      | 绿    |     17 |
| P3  | 螺丝刀    | 蓝    |     14 |
| P4  | 螺丝刀    | 红    |     14 |
| P5  | 凸轮      | 蓝    |     40 |
+-----+-----------+-------+--------+
5 rows in set (0.00 sec)
```

集合查询：使用 `IN` 运算符，判断列的值是否在指定的集合中。

语法：

```SQL
WHERE 列名 IN (值1, 值2, ...);
```

例子：查询零件，WEIGHT在集合(12,14,30)中有的零件全部信息

```
mysql> select * from p where WEIGHT IN (12,14,30);
+-----+-----------+-------+--------+
| PNO | PNAME     | COLOR | WEIGHT |
+-----+-----------+-------+--------+
| P1  | 螺母      | 红    |     12 |
| P3  | 螺丝刀    | 蓝    |     14 |
| P4  | 螺丝刀    | 红    |     14 |
| P6  | 齿轮      | 红    |     30 |
+-----+-----------+-------+--------+
4 rows in set (0.00 sec)
```

查询零件，WEIGHT不在集合(12,14,30)中有的零件全部信息

```
mysql> select * from p where NOT WEIGHT IN (12,14,30);
+-----+--------+-------+--------+
| PNO | PNAME  | COLOR | WEIGHT |
+-----+--------+-------+--------+
| P2  | 螺栓   | 绿    |     17 |
| P5  | 凸轮   | 蓝    |     40 |
+-----+--------+-------+--------+
2 rows in set (0.00 sec)
```

- *判空（IS NULL）*

IS NULL：判断列的值是否为空值，非空字符串，空字符串使用 == 判断；

语法：

```SQL
WHERE 列名 IS NULL;
```

查询Cpn为null的所有信息

```
mysql> select * from course;
+-----+--------------+------+---------+
| Cno | Cname        | Cpno | Ccredit |
+-----+--------------+------+---------+
|   1 | 数据库       |    5 |       4 |
|   2 | 数学         | NULL |       2 |
|   3 | 信息系统     |    1 |       4 |
|   4 | 操作系统     |    6 |       3 |
```

```
|   5 | 数据结构       |     7 |       4 |
|   6 | 数据处理       |  NULL |       2 |
|   7 | PASCAL语言     |     6 |       4 |
+-----+----------------+-------+---------+
7 rows in set (0.00 sec)

mysql> select * from course where Cpno IS NULL;
+-----+----------------+-------+---------+
| Cno | Cname          | Cpno  | Ccredit |
+-----+----------------+-------+---------+
|   2 | 数学           |  NULL |       2 |
|   6 | 数据处理       |  NULL |       2 |
+-----+----------------+-------+---------+
2 rows in set (0.00 sec)
```

查询Cpn为null的所有信息

```
mysql> select * from course where Cpno = "";
Empty set (0.00 sec)
```

结论：使用 = 来判断只能判断空字符串，不能判断 null；而使用 IS NULL 只能判断 null 值，不能判断空字符串。

- 模糊匹配查询（LIKE，%，_ ）

语法：

```SQL
WHERE 列名 LIKE "%M_";
```

1）LIKE：模糊查询数据使用 LIKE 运算执行通配符；
2）通配符：% 表示可有零个或多个任意字符；_ 便是需要一个任意字符；

```
一： % 包含零个或多个字符的任意字符串：

    1、LIKE 'Mc%'  将搜索以字母 Mc 开头的所有字符串（如 McBadden）。
    2、LIKE '%inger'  将搜索以字母 inger 结尾的所有字符串（如 Ringer、Stringer）。
    3、LIKE '%en%'  将搜索在任何位置包含字母 en 的所有字符串（如 Bennet、Green、McBadden）。


二：_（下划线） 任何单个字符：LIKE '_heryl' 将搜索以字母 heryl 结尾的所有六个字母的名称（如 Cheryl、
```

```
Sheryl）
```

例子：

查询，所有Cname以"系统"为结尾的所有课程

```
mysql> select * from course;
+-----+--------------+------+---------+
| Cno | Cname        | Cpno | Ccredit |
+-----+--------------+------+---------+
|   1 | 数据库       |    5 |       4 |
|   2 | 数学         | NULL |       2 |
|   3 | 信息系统     |    1 |       4 |
|   4 | 操作系统     |    6 |       3 |
|   5 | 数据结构     |    7 |       4 |
|   6 | 数据处理     | NULL |       2 |
|   7 | PASCAL语言   |    6 |       4 |
+-----+--------------+------+---------+
7 rows in set (0.00 sec)

mysql> select * from course where Cname LIKE "%系统";
+-----+--------------+------+---------+
| Cno | Cname        | Cpno | Ccredit |
+-----+--------------+------+---------+
|   3 | 信息系统     |    1 |       4 |
|   4 | 操作系统     |    6 |       3 |
+-----+--------------+------+---------+
2 rows in set (0.00 sec)
```

- 结果排序（ORDER BY）

1）ORDER BY：使用 ORDER BY子句将查询结果进行排序，ORDER BY子句出现在 SELECT 语句的最后；
2）ASC：升序；DESC：降序；

语法：

```SQL
ORDER BY 列名 [排序规则]
```

例子： *查询 所有信息 按Cpno升序排序，如果Cpno相同再按Ccredit 降序排序*

```
mysql> select * from course ORDER BY Cpno ASC, Ccredit DESC;
+-----+--------------+------+---------+
```

```
| Cno | Cname          | Cpno | Ccredit |
+-----+----------------+------+---------+
|   2 | 数学           | NULL |       2 |
|   6 | 数据处理       | NULL |       2 |
|   3 | 信息系统       |    1 |       4 |
|   1 | 数据库         |    5 |       4 |
|   7 | PASCAL语言     |    6 |       4 |
|   4 | 操作系统       |    6 |       3 |
|   5 | 数据结构       |    7 |       4 |
+-----+----------------+------+---------+
7 rows in set (0.00 sec)
```

- *DQL字句的执行顺序*

1）FROM字句：从哪张表中去查数据；
2）WHERE字句：筛选需要的行数据；
3）SELECT字句：筛选需要显示的列数据；（*对字段 AS 起别名是在这时候，所以 WHERE 中不能用字段的别名，ORDER BY 中可以使用别名*（看执行顺序））
4）ORDER BY字句：排序操作。

- *统计函数*

常用关键字(COUNT、SUM、MAX、MIN、AVG)

概念
1）COUNT(*)：统计表中有多少条数据；
2）SUM(列)：汇总列的总和；
3）MAX(列)：获取某一列的最大值；
4）MIN(列)：获取某一列的最小值；
5）AVG(列)：获取某一列的平均值；

例子：
有多少行数据

```
mysql> select * from course;
+-----+----------------+------+---------+
| Cno | Cname          | Cpno | Ccredit |
+-----+----------------+------+---------+
|   1 | 数据库         |    5 |       4 |
|   2 | 数学           | NULL |       2 |
|   3 | 信息系统       |    1 |       4 |
|   4 | 操作系统       |    6 |       3 |
```

```
|    5 | 数据结构       |    7 |        4 |
|    6 | 数据处理       | NULL |        2 |
|    7 | PASCAL语言     |    6 |        4 |
+------+----------------+------+----------+
7 rows in set (0.00 sec)

mysql> select COUNT(*) from course;
+----------+
| COUNT(*) |
+----------+
|        7 |
+----------+
1 row in set (0.00 sec)
```

查一共有多少学分：

```
mysql> select SUM(Ccredit) from course;
+--------------+
| SUM(Ccredit) |
+--------------+
|           23 |
+--------------+
1 row in set (0.00 sec)
```

查询最高的学分

```
mysql> select MAX(Ccredit) from course;
+--------------+
| MAX(Ccredit) |
+--------------+
|            4 |
+--------------+
```

```
1 row in set (0.00 sec)
```

查找最低的学分

```
mysql> select MIN(Ccredit) from course;
+---------------+
| MIN(Ccredit) |
+---------------+
|             2 |
+---------------+
1 row in set (0.01 sec)
```

求学分平均值

```
mysql> select AVG(Ccredit) from course;
+---------------+
| AVG(Ccredit) |
+---------------+
|        3.2857 |
+---------------+
1 row in set (0.00 sec)
```