DML 语句：
插入来自其他表的记录：

```
select * from stu_i;
-- 插入来自其他表的数据
insert into stu_i(sid,sname) select sid,sname from stu_01; -- 要求：字段数据相同、字段顺序要一致、字段类型要相近
```

replace 关键字：类似 insert，也是插入数据的含义，但它有个特殊用法：
当插入主键字段重复数据的时候，replace 会先删除当前行，再插入新行

```
replace into stu_i values(7,'小黑','2003-09-11');

replace into stu_01 values('0010018','666','666');
```

复制表结构和数据到新表： == 对一个表进行复制操作

```
create table stu_02 select * from stu_01; -- 对表格进行复制操作 把stu_01表进行复制 名称为 stu_02 但是不复制表中的字段关系等
```

不复制关系 表示 不复制主外建、约束等

修改表中的数据：
Update

```
select * from stu_i;
update stu_i set birthday = '2002-10-01' where sname='李山'; -- 修改数据
```

删除数据：
Delete 是 DML 语句 ； truncate 是 DDL 语句
Truncate 清空表数据 - 在表格数据量较大的时候，删除效率会快很多

```
delete from stu_i where sid=6;  -- 删除数据
TRUNCATE TABLE stu_02; -- 删除表中数据，不能加where条件，一般适合删除整表数据
```

查询语句：
select 字段列表 from 表名 [where 条件表达式]
查询语句语序：
Select 要查询的字段
From 数据源（表、视图）
Where 查询条件
Group by 分类汇总分段
Having 分类汇总条件
Order by 排序字段
Limit 截取行

```sql
select * from stu_i; --  查询所有字段
select sname,birthday from stu_i; -- 查询指定字段
select sname '姓名',birthday '生日' from stu_i;  -- 查询的时候设置别名
select sname as '姓名',birthday as '生日' from stu_i; -- 查询的时候设置别名
select stu_i.sname from stu_i; -- 字段表示方法  表名.字段名
select stu_i.sname from java_demo.stu_i;  -- 表的表示方法  库名.表名
```

排序 及 limit 操作

```sql
select '姓名',sname from stu_i; -- 查询的时候使用常量列

select * from stu_i ORDER BY sid desc; -- 默认是升序排序   asc 升序排序  desc 降序排序
select * from stu_i ORDER BY sid asc,birthday desc; -- 支持多字段排序  先遵循第一个排序字段排序

select * from stu_i limit 5; -- 截取查询结果的前5行进行显示  limit 5 == limit 0,5
select * from stu_i limit 3,2; -- 截取查询结果,从第4行开始，取2行
```

数据库查询是统计化批量的按条件查询 where sid>5 ；
但如果用编程的思想，所有的数据挨个进行遍历比较

```sql
select * from stu_i where sid != 3; -- 不等于

select * from stu_i where sid BETWEEN 2 and 4 ; --  sid>=2 and sid<=4
select * from stu_i where sid>=2 and sid<=4 ;

select * from stu_i where sid not BETWEEN 2 and 4 ;
select * from stu_i where sid<2 or sid>4 ;  -- 转化语句

select * from stu_i where sid in (1,3,10012); --  in 包含
select * from stu_i where sid=1 or sid=3 or sid=10012;

select * from stu_i where sid not in (1,3,10012); --  not in 不包含


select * from stu_i where birthday is null ;  -- 查询字段为null的数据 ， 字段为空的数据不是为0 或者是空字符串 ， 而是 null
select * from stu_i where birthday is not null ;
```

Like / rlike 支持模糊匹配

```sql
select * from stu_i where sname like '小红' ; -- like 匹配字符串
select * from stu_i where sname like '小%' ;  -- 匹配 以 小开头的字符串  % 一个或多个字符
select * from stu_i where sname like '小_' ;  -- 匹配 以 小开头的字符串 _ 一个字符

select * from stu_i where sname rlike '^小'     ; -- ^ 以 小 开头
select * from stu_i where sname rlike '海$'     ; -- $ 以 海 结尾
select * from stu_i where sname rlike '小[红白]'  ; -- [] 中括号中任取一个
select * from stu_i where sname rlike '小[红白]*'  ; -- 小  小[红白]  小[红白][红白]  小[红白][红白][红白] ....
```

mysql 中系统自带函数：
数学函数：

```
select abs(-10);
select FLOOR(9.99); -- 9  返回当前数值的最小整数值  向下取整
select FLOOR(-9.99); -- -10
select CEILING(1.0000001); -- 向上取整，返回大于x的最小整数值
select mod(10,3) ; -- 取余
select TRUNCATE(1.6666,2) -- 截取  1.66
select ROUND(1.6666,2) -- 四舍五入保留两位小数  1.67
select ROUND(16.88888,-1)  -- 四舍五入到十位数  20
```

日期函数：

```
-- 日期函数
select now(); -- 返回当前的日期和时间
select CURRENT_DATE; -- 返回当前的日期
select CURRENT_TIME; -- 返回当前的时间
select Hour('1995-10-02 15:20:32'); -- 返回指定日期的小时
select MONTH('1995-10-02 15:20:32'); -- 返回指定日期的月份
select DAYNAME(now()); -- 返回指定日期是周几
select DAYOFWEEK(now()); -- 返回指定日期是一周的第几天

select DATE_ADD(now(),INTERVAL 2 YEAR); -- 日期时间增减函数  HOUR  MINUTE  SECOND
select DATE_ADD(now(),INTERVAL -2 MONTH);
select DATE_ADD(now(),INTERVAL 20 Day);

select DATEDIFF(now(),'1990-10-1 00:00:00') -- 日期相减函数
```

字符串处理函数：

```
-- 字符串处理函数
select CONCAT('hello',' ','world!');  -- 字符串拼接
select CONCAT_WS('$','中国','泰国','新加坡');  -- 字符串拼接时增加链接符
select LEFT("newdream",3); -- 左边截取
select RIGHT("newdream",5); -- 右边截取
SELECT LENGTH('1234new'); -- 求字符串长度
select LTRIM('  123new'); -- 去除字符串左边的所有空格
select RTRIM('123new   '); -- 去除字符串右边的所有空格
select TRIM('   123new   '); -- 去除字符串左右所有空格
select UPPER('hello') ; -- 把字符串转为大写
select LOWER('HELLO');-- 把大写转为小写
select SUBSTRING("123456new",4); -- 字符串截取函数  456new
select SUBSTRING("123456new",4,3); -- 字符串截取函数,从第4个字符开始截取3个  456
select SUBSTRING("123456new",-4,3); -- 字符串截取函数,从右开始计数，倒数第4个字符开始截取3个  6ne
select REPLACE("newdream",'e','i'); -- 字符串替换
```

其它函数：

```
-- 其它函数
select CAST('12abc' AS SIGNED);
select CONVERT('666.5',signed);  -- 类型转化函数  字符串转其它类型
select DATABASE(); -- 返回当前的数据库
SELECT CONNECTION_ID(); -- 返回连接数据库的id号
select USER(); -- 返回当前登录的用户名
SELECT VERSION(); -- 查看数据库版本

SELECT FOUND_ROWS() ; -- 返回最后一个select 查询的记录数
```

聚合函数：

```sql
-- 聚合函数  mysql提供的汇总统计函数  sum()  avg()  max()  min()  count():统计行数
select sum(score) from  score;  -- 求和
select avg(score) from  score;  -- 求平均数
select max(score) from  score;  -- 求最大值
select min(score) from  score; -- 求最小值
select count(*) from  score; -- 求总行数
```

分组查询：把数据根据特征分类后进行汇总统计分析：比如全班男女人数、科目平均分、科目总分等　但不根据数据的唯一标识列进行分类汇总，比如 学号

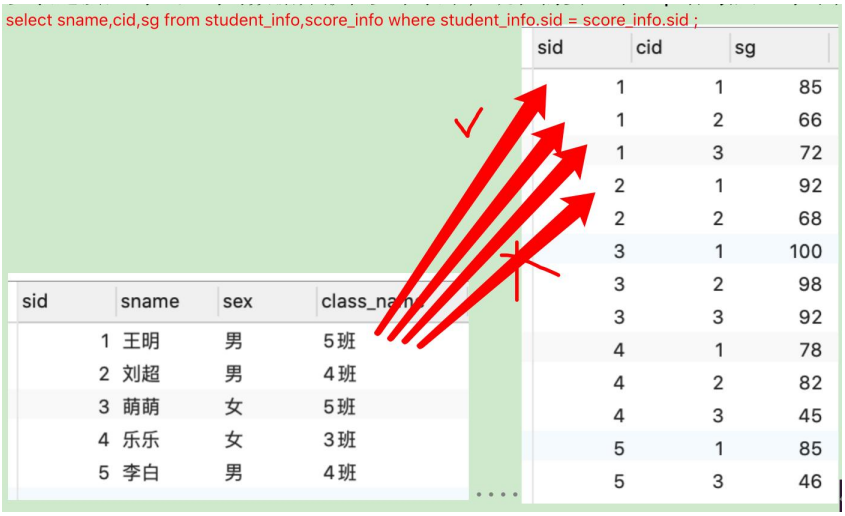
```sql
-- 分组查询:
select course,avg(score) from  score group by course; -- group by  分类汇总关键字
--  平均成绩大于80分才显示 where中不能使用聚合函数的结果作为判断条件
-- 如果想使用聚合函数的结果作为判断条件，则需使用 having 子句
select course,avg(score) from  score group by course having avg(score)>=80;
```

查询学生名单中，重名学生的姓名是什么？

```sql
select sname,count(*) from stu_i group by sname having count(*)>1;
```

多表连接查询：查询的数据分散在多个表中，现在需要一个 sql 语句能查询出来

```sql
-- 查询 每个学生 各科的成绩分别是多少？要求显示学生姓名 科目号 成绩:  学生信息 student_info  学生成绩 score_info
select sname,cid,sg from student_info,score_info;  --  因为该sql语句进行多表连接查询时，没有任何条件限制，导致产生了笛卡尔积
select sname,cid,sg from student_info,score_info where student_info.sid = score_info.sid ;
```



非等值链接：

```sql
-- 非等值链接  把score_info中的分数改为等级制中的等级
select score_info.*,s_level from score_info,score_level where sg between min_score and max_score ; -- 非等值链接
```

```sql
--三个表进行等值连接
select  si.sname,ci.cname,sc.sg
from student_info si,cource_info ci,score_info sc
where si.sid=sc.sid and ci.cid = sc.cid;
```

外连接：
左外连接：left join -- left outer join ：以左表的数据作为主表，右表有记录则匹配，无记录则显示为 null

```sql
-- 左连接  主表的数据都会显示，右边匹配上的能显示
select sname,cid,sg
from student_info
left JOIN score_info on student_info.sid = score_info.sid;
```

右外连接：right join -- right outer join :以右表的数据作为主表，左表有记录则匹配，无记录则显示为 null 左连接和右连接可以相互转换

```sql
-- 右连接  主表的数据都会显示，右边匹配上的能显示
select sname,cid,sg
from  score_info
right JOIN student_info on student_info.sid = score_info.sid;
```

内连接： inner join ：显示左右两个表共同的部分

```sql
-- 内连接 没有主从表关系，显示两个表的公共部分
select sname,cid,sg
from  score_info
inner JOIN student_info on student_info.sid = score_info.sid;
```

子查询：简单来说，就是在 where 条件中 包含 select 子句 理解为嵌套查询

```sql
-- 查询班上年龄最小学生的姓名
select min(age) from student_info;  -- 16
select sname from student_info where age = 16 ;

select sname from student_info where age = (select min(age) from student_info) ;  -- 子查询
```

在上面的子查询语句中，由于 select 子句返回的结果是一个数值 -- 标量子查询 =