

MySQL

D:\MYsql\data 里的 my.ini 是配置文件，可以改动，改动后要重新启动

MySQL 的启动和停止

MySQL的登录与退出

登录 mysql -h主机名 -P端口号 -u用户名 -p密码

退出 exit 或ctrl+z

MySQL的常见命令

DQL语言 (在SQLyog中操作)

进阶1：基础查询

1. 查询表中的单个字段

2. 查询表中的多个字段

3. 查询表中的所有字段

(1) * 只能按照表中的顺序

(2) 按照你的顺序打印出来

格式化 Ctrl+F12 或 F12

4. 查询常量值

字符型和日期型的常量值必须用单引号 " 引起来，数值型不需要

5. 查询表达式

6. 查询函数

7. 起别名

8. 去重

9. +号的作用

10. concat(str1,str2, ...) 拼接

11. IFNULL(expr1,expr2); expr1 为可能为null 的表的名字， expr2 为如果expr1 为 null ,则 返回的结果...

进阶2：条件查询

1. 分类

2. 按条件表达式筛选

案例一：查询工资>12000的员工信息

案例二：查询部门编号不等于90号的员工名字和部门编号

3. 按逻辑表达式筛选

案例一：查询工资在10000到20000之间的员工名、工资以及资金

案例二：查询部门编号不是在90到110之间，或者工资高于15000的员工信息

4. 模糊查询

(1) like (像什么)

特点：(1) 一般与通配符搭配使用

案例一：查询员工名中包含a的员工信息

案例二：查询员工名中第二个字符、第五个字符为a的员工名和工资

案例三：查询员工中第三个字符为_的员工名

5. between and (在什么与什么之间)

案例一：查询员工编号在100到120之间的员工信息

特点：

6. in

(1) 含义：判断某个字段的值是否属于 in 列表中的某一项

(2) 特点：in 列表的值类型必须一致或兼容 (in列表中不能出现通配符 % 和 _)

(3) 案例：查询员工的工种编号是 IT_PROG 、 AD_VP、 AD_PRES 中的一个员工名或工种编号

7. is null 和 is not null

(1) = 或 <> 不能用于判断null 值，

(2) is null 或 is not null 可以判断 null 值

(3) 案例一：查询没有奖金的员工名和奖金率

8. 安全等于 <=>

(1) 含义：判断是否等于，如果等于，返回true

(2) 案例一：查询没有奖金的员工名和奖金率

(3) 案例二：查询工资等于12000的员工信息

9. is null 与 安全等于<=> 的 对比

10. isnull(expr) 函数

功能：判断某字段或表达式是否为null，如果是为null，返回1；否则返回 0

11. 经典面试题

进阶3：排序查询

1. 语法

2. 特点：

案例1：查询员工信息，要求工资从高到低

案例2：查询部门编号 $>=90$ 的员工信息，按入职时间的先后顺序排列【按筛选添加筛选条件】

案例3：按年薪的高低显示员工的信息和年薪【按表达式排序】

案例4：按年薪的高低显示员工的信息和年薪【按别名排序】

案例5：按姓名的长度显示员工的姓名和工资【按函数排序】

LENGTH(str) 返回str的长度

案例6：查询员工信息，要求先按工资升序，再按员工编号降序【按多个字段排序】

3. 测试题

题1：查询员工的姓名和部门号和年薪，按年薪降序，按姓名升序

题2：选择工资不在8000到17000的员工的姓名和工资，按工资降序

题3：查询邮箱中包含e的员工信息，并按照邮箱的字节数降序、再按照部门号升序

进阶4：常见函数

一、概念：将一组逻辑语句封装在方法体中，对外暴露方法名

(1) 调用：select 函数名（实参列表）【from 表】

二、分类：

(1) 单行函数（传一个参数，返回一个值；给一组值，返回一组值），eg: concat , length , ifnull ...

(2) 分组函数（传一组参数，返回一个值）

功能：做统计使用，又称为统计函数，聚合函数，组函数

三、单行函数

(一)、字符函数

1. length(str); 获取参数值的字节个数

2. concat(str1,str2, ...); 拼接字符串

3. upper (大写) 、 lower (小写)

4. substr , substring

5. instr(str,substr); 返回子串第一次出现的索引，如果找不到则返回0

6. trim 去掉字符串前后的空格

7. lpad(str,len,padstr); 用指定的字符实现左填充指定长度

8. rpad(str,len,padstr); 用指定的字符实现右填充指定长度

9. replace(str,from_str,to_str); 用to_str替换str中的from_str

(二)、数学函数

1. round 四舍五入（取绝对值四舍五入后，加表示正负的符号）

2. ceil 向上取整，返回 $>=$ 该参数的最小整数

3. floor 向下取整，返回 $<=$ 该参数的最大整数

4. truncate(X,D) 截断，只取小数点后的D位小数

5. mod 取余，

(三)、日期函数

1. now 返回系统当前日期+时间

2. curdate 返回系统当前日期，不包含时间

3. curtime 返回当前时间，不包含日期

4. 可以获取指定的部分，年，月，日，小时，分钟，秒

5. str_to_date(str,format) 将字符通过指定格式转化为日期

6. date_format(date,format) 将日期转化为字符

(四)、其他函数

1. select version(); 查看MySQL的版本号

2. select database(); 查看当前所在的数据库

3. select user(); 查看当前的用户

4. md5('字符'); 返回该字符的md5加密形式

5. sha1('字符'); 返回该字符的密码形式

(五)、流程控制函数

1. if(expr1,expr2,expr3): 如果expr1为true，则返回expr2，否则返回expr3 (if ... else 的效果)

2. case函数

四、分组函数

1. 功能：用于统计使用，又称为聚合函数或统计函数或组函数

2. 分类：sum(求和), avg(平均值), max(最大值), min(最小值), count(计算个数)

3. 特点

4. count函数

5. datediff(expr,expr2) 日期差，expr 大日期，expr2 小日期

进阶5：分组查询

一、语法

二、简单的分组查询

(1) 查询每个工种的最高工资 (group by 后面的列表就是工种)

(2) 查询每个位置上的部门个数

三、添加分组前筛选条件 (筛选条件包含的字段来自于from 表里的，用 where 筛选条件，在from后，gr...

案例1：查询邮箱中包含a字符的，每个工资的平均工资

案例2：查询有奖金的每个领导手下的最高工资

四、添加分组后的筛选条件（筛选条件包含的字段来自于分组后 group by 的结果，用 having ,在group b...

案例1：查询哪个部门的员工个数>2

案例2：查询每个工种有奖金的员工的最高工资>12000的工种编号和最高工资

案例3：查询领导编号>102的每个领导手下的最低工资>5000的领导编号是哪个，以及最低工资

五、特点

六、按表达式或函数分组

1. 案例：按员工姓名的长度分组，查询每一组的员工个数，筛选员工个数>5的有哪些

2. group by 和 having 后面可以用 别名 (from 表 as 别名，那么其他地方就应该用 这个别名，因为fr...

七、按照多个字段分组

案例1：查询每个部门每个工种的员工的平均工资

八、添加排序

案例1：查询每个部门每个工种的员工的平均工资，并按平均工资的高低显示

案例2：查询每个部门编号不为null，每个工种的员工的平均工资，且平均工资>10000，按平均工资的...

九、案例

进阶6：连接查询（多表查询）

一、含义：当查询的字段来自于多个表时，就会用到连接查询

1. 笛卡尔乘积现象：表1 有m行，表2 有n行，结果= m*n 行

二、分类

三、SQL92标准

(一)、等值连接

1. 特点

2. 为表起别名

3. 两个表的顺序可以调换

4. 可以加上筛选

案例5：查询有奖金的员工名，部门号

5. 可以加分组

6. 可以加排序

7. 三表连接

(二)、非等值连接

案例1：查询员工的工资和工资级别

(三)、自连接（把原本的表当作二张表或更多张表使用）（自己连接自己）

案例1：查询员工名和上级名

(六) 测试题

案例1：显示所有员工的姓名，部门号和部门名称

案例2：查询90号部门员工的 job_id 和90号部门的 location_id

案例3：选择所有有奖金的员工的 last_name, department_name, location_id, city

案例4：选择city在Toronto工作的员工的 last_name, job_id, department_id, department_name

案例5：查询每个工种，每个部门的部门名，工种名和最低工资

案例6：查询每个国家下的部门个数大于2的国家编号

案例7：选择指定员工的姓名，员工名，以及他的管理者的姓名和员工号，结果类似于下面的格式

四、SQL99标准

(一)、语法

分类：

(二)、内连接

1. 语法

2. 等值连接

3. 非等值连接

4. 自连接（把原本的表当作二张表或更多张表使用）（自己连接自己）

(三)、外连接(外连接的结果=内连接的结果 + 主表有，从表没有（从表为null）)

1. 应用场景：用于查询一个表中有，另一个表没有的记录

2. 特点

3. 主表：你要查询的信息主要来自于哪个表，则就是主表

4. 案例

5. 全外连接 = 内连接的结果 + 表1中有但表2中没有的 + 表2中有但表1中没有

(四)、交叉连接(使用99语法标准实现笛卡尔乘积)

(五)、SQL92 和 SQL 99

(六) 测试题

进阶7：子查询

一、含义

1. 分类

(1) 按子查询出现的位置：

(2) 按结果集的行列数不同分类：

二、where 或 having 后面的子查询

(一)、分类：

(二)、标量子查询(单行子查询)

a. 案例

(三)、列子查询 (多行子查询)

a. 案例

(四)、行子查询 (结果集为一行多列或多行多列)

案例：查询员工编号最小并且工资最高的员工信息（可能会不存在）

三、select 后面 (仅仅支持标量子查询，即一行一列)

四、from 后面

1. 注意：将子查询结果充当一张表，要求必须起别名

2. 案例：查询每个部门的平均工资的工资等级

五、exists 后面 (相关子查询) (还没弄懂)

(一)、语法

(二)、案例

(三)、exists(子查询)，这里子查询一般为连接查询

六、测试题

七、测试题

进阶8：分页查询

一、语法

1. 应用场景：当要显示的数据，一页显示不全，需要分页提交sql请求

二、案例

(一)、查询前5条员工信息

(二)、查询有奖金的员工信息，并且工资较高的前10名显示出来

(三)、查询第11条-第25条的员工信息

三、特点

1. limit 语句放在查询语句的最后

2. 公式

D:\MYsql\data 里的 my.ini 是配置文件，可以改动，改动后要重新启动

MySQL 的启动和停止

dos窗口管理员打开,

net stop MySQL80 停止

net start MySQL80 启动 (MySQL80 是我的MySQL的用户名)

MySQL的登录与退出

登录 mysql -h主机名 -P端口号 -u用户名 -p密码

1. MySQL 8.0 Command Line Client 登陆, 直接输入密码, 缺点是只能是root用户
2. dos窗口, 管理员打开, (远程登录) **mysql -h localhost -P 3306 -u root -p** 后输入密码。

其中, -h 主机host , -P 端口号 -u 用户名 -p 密码

3. mysql -h localhost -P 3306 -u root -p密码

密码显示出, 要求-p 后无空格

4. mysql -u root -p (本机登录)

退出 exit 或ctrl+z

MySQL的常见命令

1. 查看当前所有的数据库 **show databases;**
2. 打开指定的库 **use 库名;**
3. 查看当前库的所有表 **show tables;**
4. 查看其它库的所有表 **show tables from 库名;**
5. 创建表 **create table 表名{**

列名 列类型,

列名 列类型,

...

};

6. 查看当前处于哪一个库 **select database();**
7. 查看表结构 **desc 表名;**
8. 查看表中的数据 **select * from 表名;**

9. 更新stuinfo表里的id==1的name为lilei (stuinfo表在world 库里面)

update stuinfo set name='lilei' where id=1;

10. 删除表中id==1的数据

delete from 表名 where id=1;

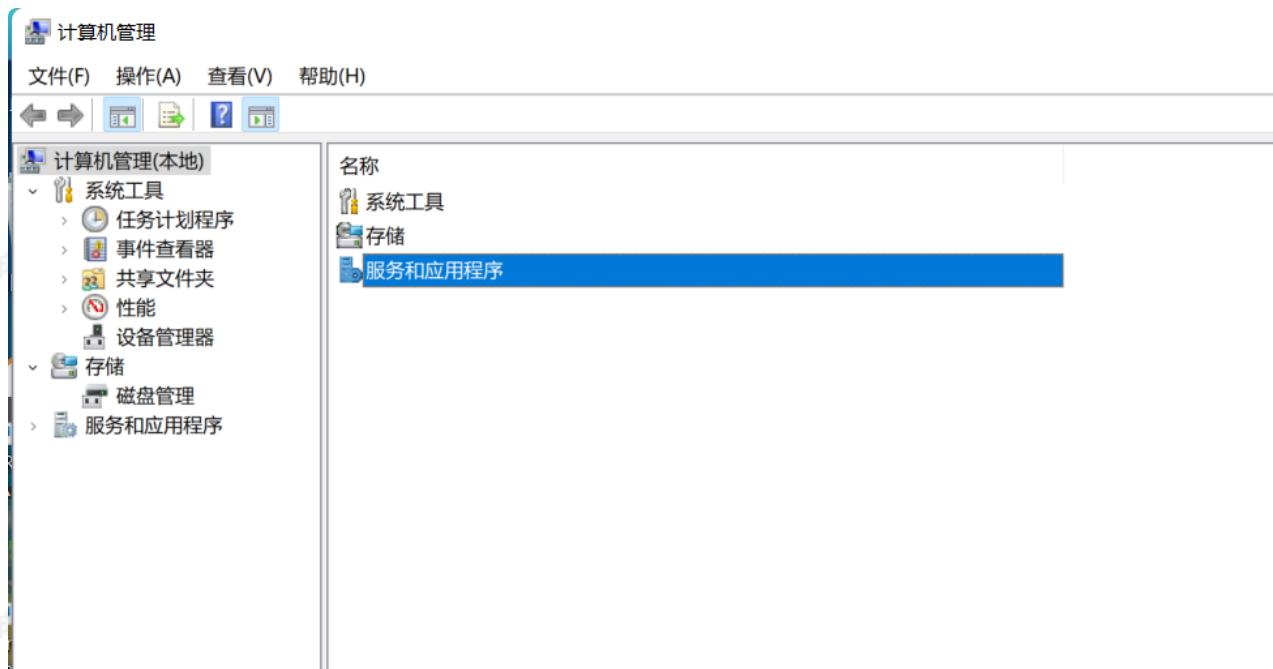
11. 查看服务器的版本

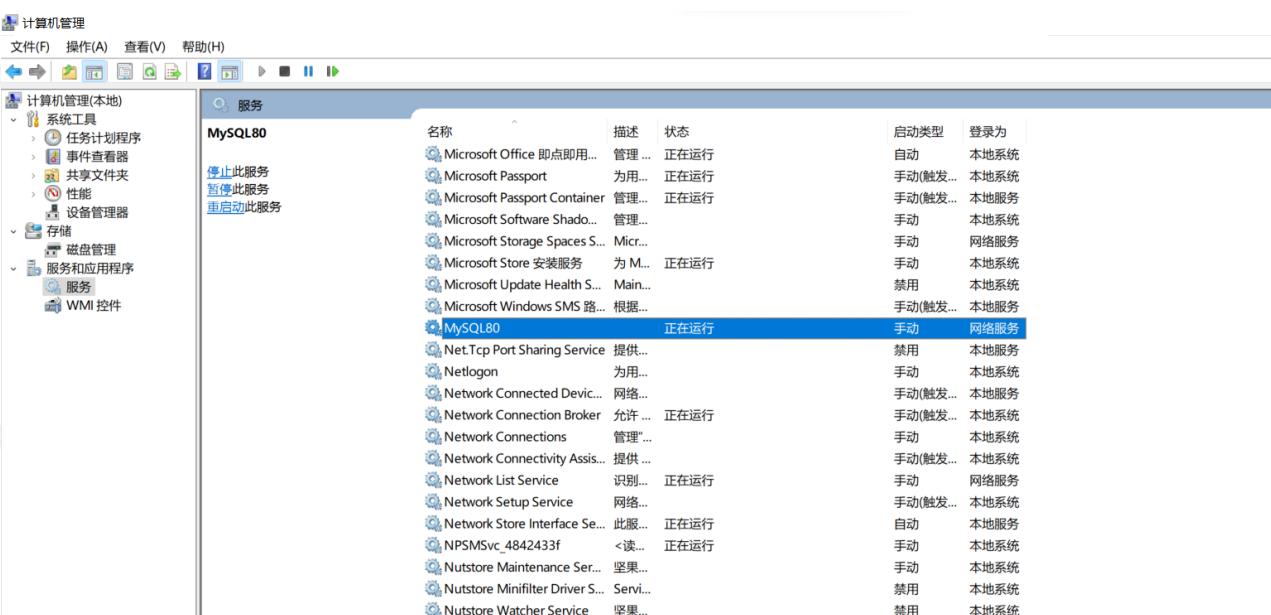
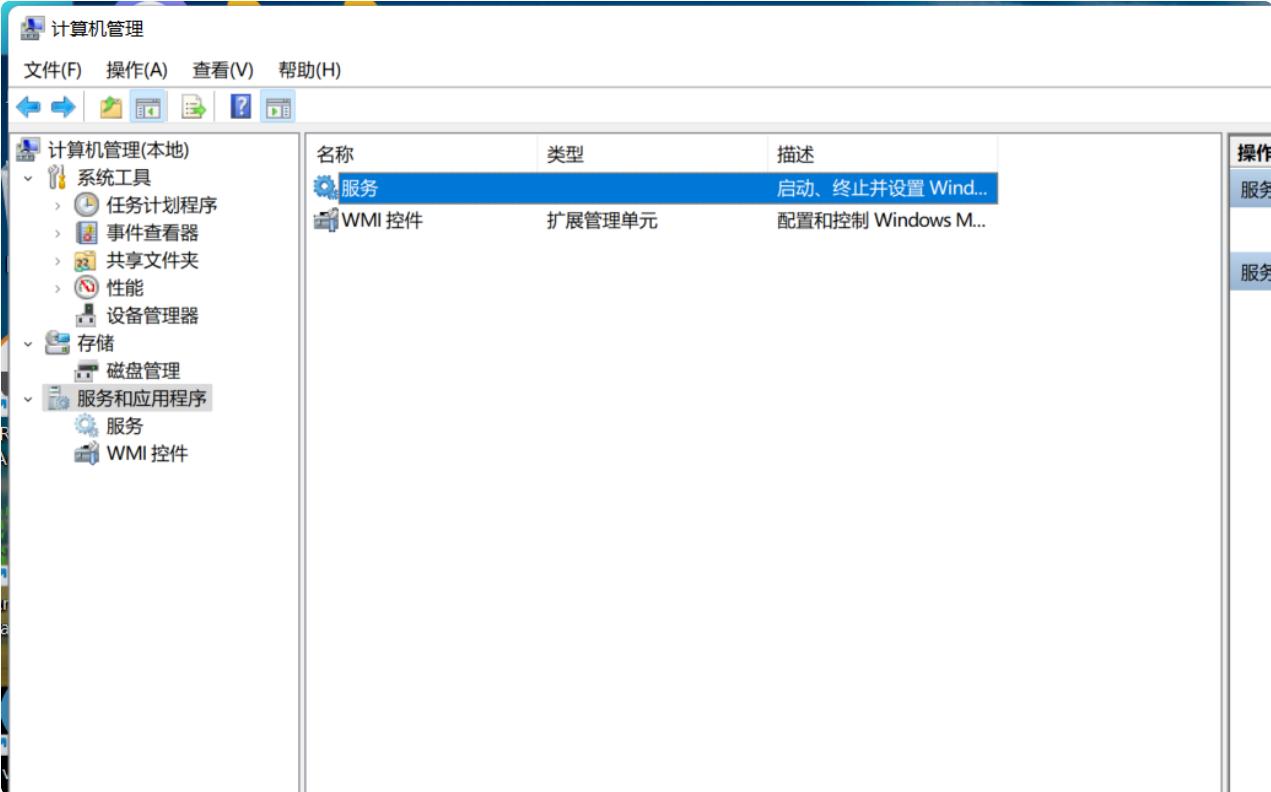
方式一：登录到mysql服务器 select version();

方式二：没有登录到mysql服务器 mysql --version 或 mysql -V

12. 图形化查看mysql是否登录

▼ 可修改是否开机自启动;





Microsoft Passport Container	管理...	正在运行	手动(触发...)	本地服务
Microsoft Software Shado...	管理...		手动	本地系统
Microsoft Storage Spaces S...	Micr...		手动	网络服务
Microsoft Store 安装服务	为 M...	正在运行	手动	本地系统
Microsoft Update Health S...	Main...		禁用	本地系统
Microsoft Windows SMS 路...	根据...		手动(触发...)	本地服务
MySQL80			启动(S)	手动
Net.Tcp Port Sharing Service	提供...		停止(O)	禁用
Netlogon	为用...		暂停(U)	手动
Network Connected Devic...	网络...		恢复(M)	手动(触发...)
Network Connection Broker	允许 ...		重新启动(E)	手动(触发...)
Network Connections	管理"		所有任务(K)	手动
Network Connectivity Assis...	提供 ...		刷新(F)	手动(触发...)
Network List Service	识别...		属性(R)	自动
Network Setup Service	网络...			手动
Network Store Interface Se...	此服...			网络服务
NPSMSvc_4842433f	<读...			手动(触发...)
Nutstore Maintenance Ser...	坚果...			本地系统
Nutstore Minifilter Driver S...	Servi...			手动
Nutstore Watcher Service	坚果...			禁用
NVIDIA Display Container LS	Cont...	正在运行		自动
NVIDIA LocalSystem Conta...	Cont...			手动
Office 64 Source Engine	Save			本地系统
Microsoft Store 安装服务	为 M...	正在运行		本地系统
Microsoft Update Health S...	Main...			禁用
Microsoft Windows SMS 路...	根据...			手动(触发...)
MySQL80		正在运行	启动(S)	手动
Net.Tcp Port Sharing Service	提供...		停止(O)	禁用
Netlogon	为用...		暂停(U)	手动
Network Connected Devic...	网络...		恢复(M)	手动(触发...)
Network Connection Broker	允许 ...	正在	重新启动(E)	手动(触发...)
Network Connections	管理..."		所有任务(K)	手动
Network Connectivity Assis...	提供 ...		刷新(F)	手动(触发...)
Network List Service	识别...	正在	属性(R)	自动
Network Setup Service	网络...			手动
Network Store Interface Se...	此服...	正在		网络服务
NPSMSvc_4842433f	<读...	正在		手动(触发...)
Nutstore Maintenance Ser...	坚果...			本地系统
Nutstore Minifilter Driver S...	Servi...			手动
Nutstore Watcher Service	坚果...			禁用
NVIDIA Display Container LS	Cont...	正在运行		自动

Microsoft Passport	为用...	正在运行
Microsoft Passport Container	管理...	正在运行
Microsoft Software Shado...	管理...	
Microsoft Storage Spaces S...	Micro...	
Microsoft Store 安装服务	为 M...	正在运行
Microsoft Update Health S...	Main...	
Microsoft Windows SMS 路...	根据...	
MySQL80	正在运行	
Net.Tcp Port Sharing Service	提供...	
Netlogon	为用...	
Network Connected Devic...	网络...	
Network Connection Broker	允许 ...	正在运行
Network Connections	管理...	
Network Connectivity Assis...	提供 ...	
Network List Service	识别...	正在运行
Network Setup Service	网络...	
Network Store Interface Se...	此服...	正在运行
NPSSvc_4842433f	<读...	正在运行
Nutstore Maintenance Ser...	坚果...	
Nutstore Minifilter Driver S...	Servi...	
Nutstore Watcher Service	坚果...	
NVIDIA Display Container LS	Cont...	正在运行
NVIDIA LocalSystem Conta...	Cont...	
Office 64 Source Engine	Save...	
OpenSSH Authentication ...	Age...	
Optimize drives	通过...	
P9RdrService_4842433f	启用...	
Peer Name Resolution Pro...	使用...	
Power Management Controller	使用...	



13. SQL的语法规范

1. 不去分大小写，但关键字大写，表名、列名小写
2. 每条命令最好用分号结尾
3. 每条命令根据需要，可以缩进蹲、或换行
4. 注释：

单行注释： #注释文字

单行注释： --空格注释文字

多行注释： /*注释文字 */

13. 运行 myemployees.sql 脚本文件，得到的数据库

▼ 数据库四张表里的一些信息

employees 员工表	
栏位	
employee_id, int(6)	员工编号
first_name, varchar(20), Nullable	名
last_name, varchar(25), Nullable	姓
email, varchar(25), Nullable	邮箱
phone_number, varchar(20), Nullable	电话号码
job_id, varchar(10), Nullable	工种编号
salary, double(10,2), Nullable	月薪
commission_pct, double(4,2), Nullable	奖金率
manager_id, int(6), Nullable	上级领导的员工编号
department_id, int(4), Nullable	部门编号
hiredate, datetime, Nullable	入职日期

employees 员工表	
栏位	
employee_id, int(6)	员工编号
first_name, varchar(20), Nullable	名
last_name, varchar(25), Nullable	姓
email, varchar(25), Nullable	邮箱
phone_number, varchar(20), Nullable	电话号码
job_id, varchar(10), Nullable	工种编号
salary, double(10,2), Nullable	月薪
commission_pct, double(4,2), Nullable	奖金率
manager_id, int(6), Nullable	上级领导的员工编号
department_id, int(4), Nullable	部门编号
hiredate, datetime, Nullable	入职日期

departments 部门表	
栏位	
department_id, int(4)	部门编号
department_name, varchar(3), Nullable	部门名称
manager_id, int(6), Nullable	部门领导的员工编号
location_id, int(4), Nullable	位置编号

locations 位置表	
栏位	
location_id, int(11)	位置编号
street_address, varchar(40), Nullable	街道
postal_code, varchar(12), Nullable	邮编
city, varchar(30), Nullable	城市
state_province, varchar(25), Nullable	州/省
country_id, varchar(2), Nullable	国家编号

jobs 工种	
栏位	
job_id, varchar(10)	工种编号
job_title, varchar(35), Nullable	工种名称
min_salary, int(6), Nullable	最低工资
max_salary, int(6), Nullable	最高工资

DQL语言（在SQLyog中操作）

进阶1：基础查询

1. 语法

select 查询列表 from 表名;

类似于：printf(打印的东西);

特点：

1. 查询列表可以是：表中的字段，常量值，表达式，函数
2. 查询的结果是一个虚拟的表格

2. 打开启用的数据库

```
USE myemployees;
```

1. 查询表中的单个字段

```
1 SELECT  
2 hiredate  
3 FROM  
4 employees ;
```

表中有一个或多个列，列又称为“字段”

2. 查询表中的多个字段

```
1 SELECT  
2 last_name,  
3 salary,  
4 email  
5 FROM  
6 employees ;
```

3. 查询表中的所有字段

(1) * 只能按照表中的顺序

```
1 SELECT  
2 *  
3 FROM  
4 employees ;
```

(2) 按照你的顺序打印出来

```
1 SELECT  
2 first_name, # `` 是着重号 , 代表是一个字段名, 区分字段与关键字, 可以去掉  
3 employee_id,  
4 last_name,  
5 phone_number,  
6 email,  
7 job_id,  
8 salary,  
9 commission_pct,  
10 manager_id,  
11 department_id,  
12 hiredate  
13 FROM  
14 employees ;
```

`` 是着重号 , 代表是一个字段名, 区分字段与关键字, 可以去掉

格式化 Ctrl+F12 或 F12

4. 查询常量值

SELECT 100;

SELECT 'john';

字符型和日期型的常量值必须用单引号 '' 引起来 , 数值型不需要

5. 查询表达式

SELECT 100%99;

6. 查询函数

SELECT VERSION(); (MySQL的版本号)

7. 起别名

SELECT 100%98 AS 结果;

方式一：使用 AS

ABAP |

```
1 SELECT
2     last_name AS 姓,
3     first_name AS 名
4 FROM
5     employees ;
6
```

方式二：使用 空格

ABAP |

```
1 SELECT
2     last_name 姓,
3     first_name 名
4 FROM
5     employees ;
6
```

SELECT salary AS "out put" FROM employees; (如果别名中有特殊符号，eg:空格，#，最好加上双引号)

8. 去重

SELECT DISTINCT department_id FROM employees; 在要查的表名前面加上 DISTINCT

9. +号的作用

运算符，两个操作数都为数值型

(1) **SELECT 100+90;** 两个数都为数值型，则作加法运算

(2) **SELECT '123'+10;** 其中一方为字符型，试图将字符型转化为数值型

如果转化成功，则继续做加法运算

如果转化失败，则将字符型数值转化为 0

(3) **SELECT null+10;** 只要其中一方为 null，则结果为 null

10. concat(str1,str2, ...) 拼接

```

1  SELECT
2      CONCAT(last_name, first_name) AS 姓名
3  FROM
4  employees ;

```

11. IFNULL(expr1,expr2); expr1 为可能为null 的表的名字, expr2 为如果expr1 为 null ,则 返回的结果为 expr2

进阶2： 条件查询

条件查询语法

1 SELECT	执行顺序
2 查询列表	(3)
3 FROM	
4 表名	(1)
5 WHERE 筛选条件 ;	(2)

1. 分类

(1) 按条件表达式筛选

条件运算符: > < = (等于) <>(不等于) != >= <=

(2) 按逻辑表达式筛选

逻辑运算符:

&& || !

and or not

(3) 模糊查询

like

between and

in

is null 或 is not null

2. 按条件表达式筛选

案例一：查询工资>12000的员工信息

```
ABAP |  
1 SELECT  
2   *  
3 FROM  
4   employees  
5 WHERE salary > 12000 ;
```

案例二：查询部门编号不等于90号的员工名字和部门编号

```
ABAP |  
1 SELECT  
2   CONCAT(last_name, ' ', first_name) AS 姓名,  
3   department_id AS 部门编号  
4 FROM  
5   employees  
6 WHERE department_id <> 90 ;
```

3. 按逻辑表达式筛选

案例一：查询工资在10000到20000之间的员工名、工资以及资金

```
ABAP |  
1 SELECT  
2   first_name AS 员工名,  
3   salary AS 工资,  
4   IFNULL(commission_pct, 0) AS 奖金  
5 FROM  
6   employees  
7 WHERE salary >= 10000 AND salary <= 20000 ;
```

案例二：查询部门编号不是在90到110之间，或者工资高于15000的员工信息

▼ (1)

ABAP |

```
1  SELECT
2    *
3  FROM
4    employees
5  WHERE NOT (
6    department_id <= 110
7    AND department_id >= 90
8  )
9  OR salary >= 15000 ;
10
```

▼ (2)

ABAP |

```
1  SELECT
2    *
3  FROM
4    employees
5  WHERE department_id > 110
6  OR department_id < 90
7  OR salary >= 15000 ;
8
```

4. 模糊查询

(1) like (像什么)

特点：(1) 一般与通配符搭配使用

通配符：

% 任意多个字符，包含0个字符

_ 任意单个字符

案例一：查询员工名中包含a的员工信息

```

1  SELECT
2    *
3  FROM
4    employees
5  WHERE last_name LIKE '%a%' ;
6

```

案例二：查询员工名中第二个字符、第五个字符为a的员工名和工资

```

1  SELECT
2    last_name,
3    salary
4  FROM
5    employees
6  WHERE last_name LIKE '_a_a%' ;
7

```

案例三：查询员工中第三个字符为 _ 的员工名

▼ (1) ESCAPE '@' --> 说明@是转义符， @可以是任意字符

```

1  SELECT
2    last_name
3  FROM
4    employees
5  WHERE last_name LIKE '_@_%' ESCAPE '@' ;

```

▼ (2) \ 是转义符

```

1  SELECT
2    last_name
3  FROM
4    employees
5  WHERE last_name LIKE '_\_%' ;
6

```

LIKE 值里有一些符号 (eg: _) 当做普通符号使用，则就可以使用 转义的方式， ESCAPE 'str'
--> 这里的str 就是转义符号

5. between and (在什么与什么之间)

案例一：查询员工编号在100到120之间的员工信息

```
1  SELECT
2    *
3  FROM
4    employees
5 WHERE employee_id BETWEEN 100
6   AND 120 ;
```

```
▼ (2) ABAP
1  SELECT
2    *
3  FROM
4    employees
5 WHERE employee_id >= 100
6   AND employee_id <= 120 ;
```

特点：

- (1) 包含两个临界值[a,b]
- (2) 两个临界值不要调换顺序，[a,b]，大于等于a ,小于等于b

6. in

(1) 含义：判断某个字段的值是否属于 in 列表中的某一项

(2) 特点：in 列表的值类型必须一致或兼容（in列表中不能出现通配符 % 和 _）

(3) 案例：查询员工的工种编号是 IT_PROG 、 AD_VP、 AD_PRES 中的一个员工名或工种编号

▼ (1)

ABAP |

```
1 SELECT
2   last_name,
3   job_id
4 FROM
5   employees
6 WHERE job_id IN ('AD_VP', 'IT_PROG', 'AD_PRES') ;
```

▼ (2)

ABAP |

```
1 SELECT
2   last_name,
3   job_id
4 FROM
5   employees
6 WHERE job_id = 'AD_VP'
7   OR job_id = 'IT_PROG'
8   OR job_id = 'AD_PRES' ;
```

7. is null 和 is not null

(1) = 或 <> 不能用于判断null 值 ,

(2) is null 或 is not null 可以判断 null 值

(3) 案例一：查询没有奖金的员工名和奖金率

```
1 SELECT
2   last_name,
3   commission_pct
4 FROM
5   employees
6 WHERE commission_pct IS NULL ;
```

8. 安全等于 <=>

(1) 含义：判断是否等于，如果等于，返回true

(2) 案例一：查询没有奖金的员工名和奖金率

```
1  SELECT
2      last_name,
3      commission_pct
4  FROM
5      employees
6  WHERE commission_pct <=> NULL ;
```

(3) 案例二：查询工资等于12000的员工信息

```
1  SELECT
2      *
3  FROM
4      employees
5  WHERE salary <=> 12000 ;
```

ABAP |

9. is null 与 安全等于<=> 的 对比

is null : 仅仅可以判断null值，可读性较高

<=> : 既可以判断null值，又可以判断普通的数值，可读性较低

10. isnul(expr) 函数

功能：判断某字段或表达式是否为null，如果是为null，返回1；否则返回 0

11. 经典面试题

试问，`SELECT * FROM employees;` (1) 和

`SELECT * FROM employees WHERE commission_pct LIKE '%%' AND last_name LIKE '%%';`
(2)

结果是否一样？说明原因。

答：不一样，commission_pct 和 last_name 中有null值，(2) 中判断的是没有null值得列表，(1) zh 中包含了null值得列表。

假设，`SELECT * FROM employees WHERE commission_pct LIKE '%%' OR last_name LIKE '%%';` (2)

(2) OR 了 employees 表中的所有列表（所有列表中总归有不为null的），那么 (1) (2) 结果一样。

进阶3： 排序查询

1. 语法

```
▼ 语法 ABAP |  
1 select 查询列表 (3)  
2 from 表 (1)  
3 【where 筛选条件】 (2)  
4 order by 排序列表【asc|desc】 (4)
```

这里的 (1) (2) (3) (4) 是执行的顺序

2. 特点：

- (1) asc 代表的是升序， desc 代表的是降序；如果不写， 默认为升序
- (2) order by 子句中可以支持单个字段、多个字段、表达式、函数、别名
- (3) order by 子句一般放在查询语句的最后面， limit 子句除外

案例1：查询员工信息，要求工资从高到低

```
▼ 从高到低，即降序 ABAP |  
1 SELECT  
2 *  
3 FROM  
4 employees  
5 ORDER BY salary DESC ;
```

案例2：查询部门编号>=90的员工信息，按入职时间的先后顺序排列 【按筛选添加筛选条件】

```
1 SELECT  
2 *  
3 FROM  
4 employees  
5 WHERE department_id >= 90  
6 ORDER BY hiredate ASC ;
```

案例3：按年薪的高低显示员工的信息和年薪 【按表达式排序】

```
1 SELECT
2   *,
3     salary * 12 * (1+ IFNULL(commission_pct, 0)) AS 年薪
4 FROM
5   employees
6 ORDER BY salary * 12 * (1+ IFNULL(commission_pct, 0)) DESC ;
```

案例4：按年薪的高低显示员工的信息和年薪【按别名排序】

```
1 SELECT
2   *,
3     salary * 12 * (1+ IFNULL(commission_pct, 0)) AS 年薪
4 FROM
5   employees
6 ORDER BY 年薪 DESC ;
```

案例5：按姓名的长度显示员工的姓名和工资【按函数排序】

LENGTH(str) 返回str的长度

```
1 SELECT
2   LENGTH(last_name) AS 字节长度,
3   last_name,
4   salary
5 FROM
6   employees
7 ORDER BY LENGTH(last_name) DESC ;
```

案例6：查询员工信息，要求先按工资升序，再按员工编号降序【按多个字段排序】

```
1 SELECT
2   *
3 FROM
4   employees
5 ORDER BY salary ASC,
6   employee_id DESC ;
```

整体上是按工资的升序，如果工资中有相同的几个，则这几个按员工编号排序

3. 测试题

题1：查询员工的姓名和部门号和年薪，按年薪降序，按姓名升序

```
1 SELECT
2     last_name,
3     department_id,
4     salary * 12 * (1+ IFNULL(commission_pct, 0)) AS 年薪
5 FROM
6     employees
7 ORDER BY 年薪 DESC,
8     last_name ASC ;
```

ABAP |

题2：选择工资不在8000到17000的员工的姓名和工资，按工资降序

```
1 SELECT
2     last_name,
3     salary
4 FROM
5     employees
6 WHERE NOT (salary BETWEEN 8000
7             AND 17000)
8 ORDER BY salary DESC ;
```

ABAP |

```
1 SELECT
2     last_name,
3     salary
4 FROM
5     employees
6 WHERE salary NOT BETWEEN 8000
7             AND 17000
8 ORDER BY salary DESC ;
9
```

ABAP |

题3：查询邮箱中包含e的员工信息，并按照邮箱的字节数降序、再按照部门号升序

```
1  SELECT
2    *
3  FROM
4    employees
5  WHERE email LIKE '%e%'
6  ORDER BY LENGTH(email) DESC,
7    department_id ASC ;
```

进阶4： 常见函数

一、 概念：将一组逻辑语句封装在方法体中，对外暴露方法名

(1) 调用：select 函数名（实参列表）【from 表】

二、 分类：

(1) 单行函数（传一个参数，返回一个值；给一组值，返回一组值），eg: concat , length , ifnull ...

(2) 分组函数（传一组参数，返回一个值）

功能：做统计使用，又称为统计函数，聚合函数，组函数

三、 单行函数

(一)、字符函数

1. length(str); 获取参数值的字节个数

在MySQL 8.0 客户端中 一个汉字 = 3 byte

```
1  SELECT LENGTH('john');
2  SELECT LENGTH('张三丰hahaha');
```

SHOW VARIABLES LIKE '%char%';

Variable_name	Value
character_set_client	utf8mb3
character_set_connection	utf8mb3

character_set_database	gb2312
character_set_filesystem	binary
character_set_results	utf8mb3
character_set_server	utf8mb4
character_set_system	utf8mb3
character_sets_dir	D:\\MYsql\\\\share\\\\charsets\\\\

2. concat(str1,str2, ...); 拼接字符串

```

1  SELECT
2      CONCAT(last_name, '_', first_name) AS 姓名
3  FROM
4      employees ;

```

3. upper (大写)、lower (小写)

```

1  SELECT UPPER('jilo');
2  SELECT LOWER('JYTREW');

```

(1) 示7例：将姓变大写，名变小写，然后拼接

```

1  SELECT
2      CONCAT(
3          UPPER(last_name),
4          '_',
5          LOWER(first_name)
6      )
7  FROM
8      employees ;

```

4. substr, substring

(1) 注意：索引从1开始

(2) substr(str,pos); 截取从指定索引后面所有字符

```
1 SELECT
2   SUBSTR(
3     '李莫愁爱上了陆展元',
4     7
5   ) AS out_put ;
```

(3) substr(str,pos,len); 截取从指定索引开始的指定长度len的字符串

```
1 SELECT
2   SUBSTR(
3     '李莫愁爱上了陆展元',
4     1,
5     3
6   ) AS out_put ;
```

(4) 案例：姓名中首字符大写，其他字符小写，然后用 _ 拼接，显示出来

```
1 SELECT
2   CONCAT(
3     UPPER(SUBSTR(last_name, 1, 1)),
4     '_',
5     LOWER(SUBSTR(last_name, 2))
6   ) AS 姓名
7 FROM
8   employees ;
```

5. instr(str,substr); 返回子串第一次出现的索引，如果找不到则返回0

▼ (1)

```
1 SELECT
2   INSTR(
3     '杨不悔爱上了殷六侠',
4     '殷六侠'
5   ) AS out_put ;
```

▼ (2)

```
1 SELECT
2   INSTR(
3     '杨不悔爱上了殷六侠',
4     '殷六侠'
5   ) AS out_put ;
```

ABAP |

ABAP |

6. trim 去掉字符串前后的空格

```
1  SELECT TRIM('    张翠山    ') AS out_put;
2  SELECT LENGTH(TRIM('    张翠山    ')) AS out_put;
```

(1) 去掉字符串前后的a

```
1  SELECT
2      TRIM(
3          'a' FROM 'aaaaaaaaaaa张aaaaaaaa翠山aaaaaaaa'
4      ) AS out_put ;
```

7. lpad(str,len,padstr); 用指定的字符实现左填充指定长度

```
1  SELECT
2      LPAD('殷素素', 10, '*') AS out_put ;
```

(1) 如果字符串超过指定长度，则从右边删去超过的字符，不填充

```
1  SELECT
2      LPAD('殷素素', 2, '*') AS out_put ;
```

8. rpad(str,len,padstr); 用指定的字符实现右填充指定长度

```
1  SELECT RPAD('殷素素',12,'ab') AS out_put;
```

(1) 如果字符串超过指定长度，则从右边删去超过的字符，不填充

```
1  SELECT RPAD('殷小素',2,'ab') AS out_put;
```

9. replace(str,from_str,to_str); 用to_str替换str中的from_str

▼ (1)

```
1  SELECT
2      REPLACE(
3          '张无忌爱上了周芷若',
4          '周芷若',
5          '赵敏'
6      ) AS out_put ;
```

ABAP |

```

1  SELECT
2    REPLACE(
3      '张无忌周芷若周芷若爱周芷若上了周芷若',
4      '周芷若',
5      '赵敏'
6    ) AS out_put ;
7

```

(二)、数学函数

1. round 四舍五入 (取绝对值四舍五入后，加表示正负的符号)

(1) round(X)

```
1  SELECT ROUND(-1.499);
```

(2) round(X,D) 四舍五入，小数点后保留D位小数

```
1  SELECT ROUND(1.567,2);
```

2. ceil 向上取整，返回 \geq 该参数的最小整数

```
1  SELECT CEIL(2.05);
```

3. floor 向下取整，返回 \leq 该参数的最大整数

```
1  SELECT FLOOR(1.76);
```

4. truncate(X,D) 截断，只取小数点后的D位小数

```
1  SELECT TRUNCATE(1.69,1);
```

5. mod 取余，

$$\text{mod}(a,b) = a - \frac{a}{b} * b$$

$$\text{eg: } \text{mod}(-10,-3) = -10 - (-10/-3)*-3 \\ = -1$$

```
1  SELECT MOD(8,6);
```

(三)、日期函数

1. now 返回系统当前日期+时间

```
1 SELECT NOW();
```

2. curdate 返回系统当前日期，不包含时间

```
1 SELECT CURDATE();
```

3. curtime 返回当前时间，不包含日期

```
1 SELECT CURTIME();
```

4. 可以获取指定的部分，年，月，日，小时，分钟，秒

(1) 获取当前系统的年份

```
1 SELECT YEAR(NOW());
```

(2) 获得给定日期的年份

```
1 SELECT YEAR('2002-3-7');
```

(3) 获得employees表中入职时间的年份

```
1 SELECT YEAR(hiredate) AS 年 FROM employees;
```

(4) 获取当前系统的月份

```
1 SELECT MONTH(NOW()) AS 月;
```

(5) 获得当前系统的月份，并按照英文写

```
1 SELECT MONTHNAME(NOW()) AS 月;
```

5. str_to_date(str,format) 将字符串通过指定格式转化为日期

```
1 SELECT STR_TO_DATE('2002-3-7','%Y-%c-%d') AS out_put;
```

前面字符中的年月日之间的间隔是怎样的，后面格式format中保持一致

(1) 查询入职日期为2000-9-9 的员工信息

```
1 SELECT * FROM employees WHERE hiredate = '2000-9-9';
```

```
1 SELECT * FROM employees WHERE hiredate = STR_TO_DATE('9-9 2000', '%c-%d %Y');
```

6. date_format(date,format) 将日期转化为字符

(1) 将当前系统的日期转化为 XX年XX月XX日

```
1 SELECT DATE_FORMAT(NOW(), '%y年%m月%d日') AS out_put;
```

(2) 查询有奖金的员工名和入职日期 (xx月/xx日 xx年)

```
1 SELECT
2     last_name,
3     DATE_FORMAT(hiredate, '%m月/%d日 %y年') AS 入职时期
4 FROM
5     employees
6 WHERE commission_pct IS NOT NULL ;
```

%Y	四位的年份
%y	2位的年份
%m	月份(01,02,...11,12)
%c	月份(1,2,...11,12)
%d	日(01,02,...)
%H	小时(24小时制)
%h	小时(12小时制)
%i	分钟(00,01,...59)
%s	秒(00,01,...59)

(四) 其他函数

1. select version(); 查看MySQL的版本号

2. select database(); 查看当前所在的数据库

3. select user(); 查看当前的用户

4. md5('字符'); 返回该字符的md5加密形式

5. sha1('字符'); 返回该字符的密码形式

(五)、流程控制函数

1. if(expr1,expr2,expr3): 如果expr1为true, 则返回expr2, 否则返回expr3 (if ... else 的效果)

(1) 如果 $10 > 5$, 则返回结果 大 ; 否则, 返回 小

```
1  SELECT IF(10>5, '大', '小');
```

(2) 查询员工姓名和奖金, 并标出又没有奖金

```
1  SELECT
2      last_name,
3      commission_pct,
4      IF(
5          commission_pct IS NULL,
6          '没奖金',
7          '有奖金'
8      ) AS out_put
9  FROM
10     employees ;
```

2. case函数

(1) case函数的使用一: switch case 的效果

C++中

ABAP |

```
1 ▼ switch (变量或表达式){
2     case 常量1: 语句1; break;
3     case 常量2: 语句2; break;
4     ...
5     default: 语句n; break;
6 }
```

MySQL中

ABAP |

```
1 case 要判断的字段或表达式,
2 when 常量1 then 要显示的 值1 或 语句1;
3 when 常量2 then 要显示的 值2 或 语句2;
4 ...
5 else 要显示的 值n 或 语句n;
6 end
```

语句后面要加分号；

(2) 案例1：查询员工的工资，要求

部门号=30，显示的工资为1.1倍

部门号=40，显示的工资为1.2倍

部门号=50，显示的工资为1.3倍

其他部门，显示的工资为原工资

```
1 SELECT salary AS 原始工资,department_id,
2 CASE department_id
3 WHEN 30 THEN salary*1.1
4 WHEN 40 THEN salary*1.2
5 WHEN 50 THEN salary*1.3
6 ELSE salary
7 END AS 新工资
8 FROM employees;
```

(3) case函数的使用二：类似于多重if

C++中

ABAP |

```
1 if(条件1){
2     语句1;
3 } else if(条件2){
4     语句2;
5 }
6 ...
7 else {
8     语句n;
9 }
```

```

1 case
2 when 条件1 then 要显示的 值1 或 语句1;
3 when 条件2 then 要显示的 值2 或 语句2;
4 ...
5 else 要显示的 值n 或 语句n;
6 end

```

(4) 案例2：查询员工的工资情况

如果工资>20000， 显示A级别

如果工资>15000， 显示B级别

如果工资>10000， 显示C级别

否则， 显示D级别

```

1 SELECT
2     salary AS 工资,
3     CASE
4         WHEN salary > 20000
5             THEN 'A'
6         WHEN salary > 15000
7             THEN 'B'
8         WHEN salary > 10000
9             THEN 'C'
10        ELSE 'D'
11    END AS 工资级别
12 FROM
13     employees ;

```

case函数块， if函数块， 可以看作一个字段

(5) 案例3：使用 case-when， 按照下面的条件：

job	Grade
AD_PRES	A
ST_MAN	B
IT_PROC	C
SA_REP	D

ST_CLERK

E

产生下面的结果

last_name	job_id	Grade
king	AD_PRES	A

```

1  SELECT
2      last_name, job_id,
3      CASE
4          job_id
5          WHEN 'AD_PRES' THEN 'A'
6          WHEN 'ST_MAN' THEN 'B'
7          WHEN 'IT_PROG' THEN 'C'
8          WHEN 'SA_REP' THEN 'D'
9          WHEN 'ST_CLERK' THEN 'E'
10     END AS Grade
11    FROM
12      employees
13    WHERE job_id = 'AD_PRES';
14

```

四、分组函数

1. 功能：用于统计使用，又称为聚合函数或统计函数或组函数

2. 分类：sum(求和), avg(平均值), max(最大值), min(最小值), count(计算个数)

```

1  SELECT SUM(salary) FROM employees;
2  SELECT MAX(salary) FROM employees;
3  SELECT MIN(salary) FROM employees;
4  SELECT AVG(salary) FROM employees;
5  SELECT COUNT(salary) FROM employees;

```

```
1 SELECT
2     SUM(salary) AS 和,
3     MAX(salary) AS 最大值,
4     MIN(salary) AS 最小值,
5     ROUND(AVG(salary),2) AS 平均值,
6     COUNT(salary) AS 个数
7 FROM
8     employees ;
```

3. 特点

- (1) sum, avg一般用于处理数值型; max, min, count 可以处理任何类型
- (2) 都忽略null值

```
1 SELECT
2     MAX(commission_pct),
3     MIN(commission_pct)
4 FROM
5     employees ;
6
7 SELECT
8     SUM(commission_pct),
9     AVG(commission_pct),
10    SUM(commission_pct) / 35,
11    SUM(commission_pct) / 107
12 FROM
13     employees ;
```

- (3) 可以和 distinct 搭配实现去重的运算

```
1 SELECT SUM(DISTINCT salary),SUM(salary) FROM employees;
2 SELECT AVG(DISTINCT salary),AVG(salary) FROM employees;
3 SELECT MAX(DISTINCT salary),MAX(salary) FROM employees;
4 SELECT MIN(DISTINCT salary),MIN(salary) FROM employees;
5 SELECT COUNT(DISTINCT salary),COUNT(salary) FROM employees;
```

4. count函数

- (1) 一般使用 count(*) 统计结果集的行数, (count(*),所有字段只要有一行不为null, 就+1)
- (2) **SELECT COUNT(1) FROM employees;**

count(常量值) 在所有字段旁边加了一列 常量 , 计算这个常量的数量, 即所有字段的总行数

效率: 在INNODB引擎下, count(*) 和 count(1) 的效率差不多, 比 count(字段名) 要高一些

5. datediff(expr,expr2) 日期差, expr 大日期 , expr2 小日期

```
SELECT DATEDIFF('2023-2-14','2022-10-3');
```

(1) 案例：查询员工表中的最大入职时间和最小入职时间的相差天数

```
1  SELECT DATEDIFF(MAX(hiredate),MIN(hiredate)) FROM employees;
```

进阶5： 分组查询

一、语法

```
1  select 分组函数,列(要求出现在 group by 的后面)          (5)
2  from 表                                         (1)
3  【where 筛选条件】                               (2)
4  group by 分组的列表                           (3)
5  【having 分组后的筛选】                         (4)
6  【order by 子句】                            (6)
```

注意：查询列表必须特殊，要求是 分组函数 和 group by 后出现的字段

二、简单的分组查询

(1) 查询每个工种的最高工资 (group by 后面的列表就是工种)

```
1  SELECT
2      MAX(salary),
3      job_id
4  FROM
5      employees
6  GROUP BY job_id ;
```

(2) 查询每个位置上的部门个数

```
1  SELECT
2      COUNT(*),
3      location_id
4  FROM
5      departments
6  GROUP BY location_id ;
```

三、添加分组前筛选条件（筛选条件包含的字段来自于from 表里的，用 where 筛选条件，在 from后， group by 前面）

案例1：查询邮箱中包含a字符的，每个工资的平均工资

```
1 SELECT
2     AVG(salary),
3     department_id
4 FROM
5     employees
6 WHERE email LIKE '%a%'
7 GROUP BY department_id ;
```

案例2：查询有奖金的每个领导手下的最高工资

```
1 SELECT
2     MAX(salary) AS 最高工资,
3     manager_id
4 FROM
5     employees
6 WHERE commission_pct IS NOT NULL
7 GROUP BY manager_id ;
```

四、添加分组后的筛选条件（筛选条件包含的字段来自于分组后 group by 的结果，用 having ,在 group by 后面）

案例1：查询哪个部门的员工个数>2

(1) 查询每个部门的员工数

```
1 SELECT
2     COUNT(*),
3     department_id
4 FROM
5     employees
6 GROUP BY department_id ;
```

(2) 根据 (1) 的结果进行筛选，查询哪个部门的员工个数>2

```
1 SELECT
2     COUNT(*) AS 数量,
3     department_id
4 FROM
5     employees
6 GROUP BY department_id
7 HAVING COUNT(*) > 2 ;
```

案例2：查询每个工种有奖金的员工的最高工资>12000的工种编号和最高工资

(1) 查询每个工种有奖金的员工的最高工资

```
1 SELECT
2     MAX(salary),
3     job_id
4 FROM
5     employees
6 WHERE commission_pct IS NOT NULL
7 GROUP BY job_id ;
```

(2) 根据 (1) 的结果继续筛选，最高工资>12000

```
1 SELECT
2     MAX(salary),
3     job_id
4 FROM
5     employees
6 WHERE commission_pct IS NOT NULL
7 GROUP BY job_id
8 HAVING MAX(salary) > 12000 ;
```

案例3：查询领导编号>102的每个领导手下的最低工资>5000的领导编号是哪个，以及最低工资

(1) 查询每个领导手下的员工最低工资

```
1 SELECT
2     MIN(salary),
3     manager_id
4 FROM
5     employees
6 GROUP BY manager_id ;
```

(2) 添加筛选条件： 编号 > 102

```

1  SELECT
2      MIN(salary),
3      manager_id
4  FROM
5      employees
6  WHERE manager_id > 102
7  GROUP BY manager_id

```

(3) 添加筛选条件：最低工资 > 5000

```

1  SELECT
2      MIN(salary),
3      manager_id
4  FROM
5      employees
6  WHERE manager_id > 102
7  GROUP BY manager_id
8  HAVING MIN(salary) > 5000;

```

五、特点

	数据源	位置	关键字
分组前筛选	原始表	group by 子句的前面	where
分组后筛选	分组后的结果集	group by 子句的后面	having

(1) 分组函数做条件肯定放在having 子句中

(2) 能用分组前筛选的，就优先考虑使用前筛选

(3) group by 子句支持单个字段分组，多个字段分组（多个字段之间用逗号隔开，没有顺序要求），表达式或函数（用的较少）

(4) 也可以添加排序（排序放在整个分组查询的最后）

六、按表达式或函数分组

1. 案例：按员工姓名的长度分组，查询每一组的员工个数，筛选员工个数>5的有哪些

(1) 查询每个长度的员工个数

```
1 SELECT
2     LENGTH(last_name) AS len_name,
3     COUNT(*)
4 FROM
5     employees
6 GROUP BY LENGTH>Last_name;
```

(2) 添加筛选条件

```
1 SELECT
2     LENGTH(last_name) AS len_name,
3     COUNT(*)
4 FROM
5     employees
6 GROUP BY LENGTH>Last_name
7 HAVING COUNT(*)>5;
```

2. group by 和 having 后面可以用别名 (from 表 as 别名, 那么其他地方就应该用 这个别名, 因为from 表 是第一个执行)

```
1 SELECT
2     LENGTH(last_name) AS len_name,
3     COUNT(*) AS 个数
4 FROM
5     employees
6 GROUP BY len_name
7 HAVING 个数>5;
```

七、按照多个字段分组

案例1：查询每个部门每个工种的员工的平均工资

```
1 SELECT
2     AVG(salary),
3     department_id,
4     job_id
5 FROM
6     employees
7 GROUP BY department_id,
8     job_id ;
```

八、添加排序

案例1：查询每个部门每个工种的员工的平均工资，并按平均工资的高低显示

```
1 SELECT
2     AVG(salary),
3     department_id,
4     job_id
5 FROM
6     employees
7 GROUP BY department_id,
8     job_id
9 ORDER BY AVG(salary) DESC ;
```

案例2：查询每个部门编号不为null，每个工种的员工的平均工资，且平均工资>10000，按平均工资的高低显示

```
1 SELECT
2     AVG(salary),
3     department_id,
4     job_id
5 FROM
6     employees
7 WHERE department_id IS NOT NULL
8 GROUP BY department_id,
9     job_id
10 HAVING AVG(salary) > 10000
11 ORDER BY AVG(salary) DESC ;
```

九、案例

(1) 案例1：查询各job_id的员工工资的最大值，最小值，平均值，总和，并按job_id升序

```
1 SELECT
2     MAX(salary),
3     MIN(salary),
4     AVG(salary),
5     SUM(salary),
6     job_id
7 FROM
8     employees
9 GROUP BY job_id
10 ORDER BY job_id ASC ;
```

(2) 案例2：查询员工最高工资和最低工资的差距（DIFFERENCE）

```
1 SELECT
2   (MAX(salary) - MIN(salary)) AS DIFFERENCE
3 FROM
4   employees ;
```

(3) 案例3：查询各个管理者手下员工的最低工资，其中最低工资不能低于6000，没有管理者的员工不计算于内

```
1 SELECT
2   MIN(salary),
3   manager_id
4 FROM
5   employees
6 WHERE manager_id IS NOT NULL
7 GROUP BY manager_id
8 HAVING MIN(salary) >= 6000 ;
```

(4) 案例4：查询所有部门的编号，员工数量和工资平均值，并按平均工资降序

```
1 SELECT
2   department_id,
3   COUNT(*),
4   AVG(salary)
5 FROM
6   employees
7 GROUP BY department_id
8 ORDER BY AVG(salary) DESC ;
```

(5) 案例5：选择具有各个job_id的员工人数

```
1 SELECT
2   COUNT(*),
3   job_id
4 FROM
5   employees
6 GROUP BY job_id ;
```

进阶6：连接查询（多表查询）

一、含义：当查询的字段来自于多个表时，就会用到连接查询

1. 笛卡尔乘积现象：表1 有m行，表2 有n行，结果= m*n 行

发生条件：没有有效的连接条件

如何避免：添加有效的连接条件

用一个表去匹配另一个表，当连接条件的值相等时，才筛选出来

二、分类

按年代分类	按功能分类
SQL92 标准：仅仅支持内连接	内连接：等值连接+非等值连接+自连接
SQL99标准【推荐】：支持内连接+外连接（左外连接和右外连接）+交叉连接	外连接：左外连接+右外连接+全外连接
	交叉连接

三、SQL92标准

(一)、等值连接

1. 特点

- (1) 多表等值连接的结果是多表的交集的部分
- (2) n表连接，至少需要 n-1 个连接条件
- (3) 多表的顺序没有要求
- (4) 可以搭配前面介绍的所有子句使用，比如 排序， 分组， 筛选

案例1：查询女神名对应的男神名

```
1  SELECT
2      NAME,
3      boyName
4  FROM
5      boys,
6      beauty
7  WHERE beauty.boyfriend_id = boys.id ;
```

案例2：查询员工名和对应的部门名

```
1 SELECT
2   last_name,
3   department_name
4 FROM
5   employees,
6   departments
7 WHERE departments.`department_id` = employees.`department_id` ;
```

2. 为表起别名

案例3：查询员工名，工种号，工种名

未起别名

ABAP |

```
1 SELECT
2   last_name,
3   employees.job_id,
4   job_title
5 FROM
6   employees,
7   jobs
8 WHERE employees.`job_id` = jobs.`job_id` ;
```

```
1 SELECT
2   last_name,
3   e.job_id,
4   job_title
5 FROM
6   employees AS e,
7   jobs
8 WHERE e.`job_id` = jobs.`job_id` ;
```

特点：

- (1) 提高了语句的简洁度
- (2) 区分多个重名的字段

注意：如果为表起了别名，则查询的字段就不能使用原来的表名去限定

3. 两个表的顺序可以调换

案例4：查询员工名，工种号，工种名

```
1 SELECT
2     e.last_name,
3     e.job_id,
4     job_title
5 FROM
6     jobs,
7     employees AS e
8 WHERE e.`job_id` = jobs.`job_id` ;
```

4. 可以加上筛选

案例5：查询有奖金的员工名，部门号

```
1 SELECT
2     last_name,
3     department_name,
4     commission_pct
5 FROM
6     employees,
7     departments
8 WHERE departments.department_id = employees.`department_id`
9     AND employees.`commission_pct` IS NOT NULL ;
```

案例6：查询城市名中第二个字符为o的部门名和城市名

```
1 SELECT
2     department_name,
3     city
4 FROM
5     departments,
6     locations
7 WHERE locations.location_id = departments.`location_id`
8     AND city LIKE '_o%' ;
```

5. 可以加分组

案例7：查询每个城市的部门个数

```
1 SELECT
2   COUNT(*) 个数,
3   city
4 FROM
5   departments,
6   locations
7 WHERE departments.`location_id` = locations.`location_id`
8 GROUP BY city ;
```

案例8：查询有奖金的每个部门的部门名和部门的领导编号和该部分的最低工资

```
1 SELECT
2   department_name,
3   employees.manager_id,
4   MIN(salary)
5 FROM
6   departments,
7   employees
8 WHERE departments.`department_id` = employees.`department_id`
9   AND commission_pct IS NOT NULL
10 GROUP BY department_name,departments.`manager_id` ;
```

6. 可以加排序

案例9：查询每个工种的工种名和员工的个数，并且按照员工个数降序

▼ 不加别名

ABAP |

```
1 SELECT
2   job_title,
3   COUNT(*)
4 FROM
5   jobs,
6   employees
7 WHERE jobs.`job_id` = employees.`job_id`
8 GROUP BY job_title
9 ORDER BY COUNT(*) DESC ;
```

```
1 SELECT
2   job_title,
3   COUNT(*)
4 FROM
5   jobs AS j,
6   employees AS e
7 WHERE j.`job_id` = e.`job_id`
8 GROUP BY job_title
9 ORDER BY COUNT(*) DESC ;
```

7. 三表连接

案例10：查询员工名，部门名和所在的城市

```
1 SELECT
2   last_name,
3   department_name,
4   city
5 FROM
6   employees,
7   departments,
8   locations
9 WHERE departments.`department_id` = employees.`department_id`
10    AND departments.`location_id` = locations.`location_id` ;
11
```

案例11：查询员工名，部门名和所在的以s开头的城市，并将城市名按照降序排

```
1 SELECT
2   last_name,
3   department_name,
4   city
5 FROM
6   employees,
7   departments,
8   locations
9 WHERE departments.`department_id` = employees.`department_id`
10    AND departments.`location_id` = locations.`location_id`
11    AND city LIKE 's%'
12 ORDER BY department_name DESC ;
```

案例1：查询员工的工资和工资级别

```
1 SELECT
2     salary,
3     grade_level
4 FROM
5     employees,
6     job_grades
7 WHERE salary BETWEEN lowest_sal
8     AND highest_sal
```

(三)、自连接（把原本的表当作二张表或更多张表使用）（自己连接自己）

employee_id	last_name	manager_id
100	King	(NULL)
101	Kochhar	100
102	De Haan	100
103	Hunold	102
104	Ernst	103
105	Austin	103
106	Pataballa	103
107	Lorentz	103
108	Greenberg	101
109	Faviet	108
110	Chen	108
111	Sciarra	108
112	Urman	108
113	Popp	108
114	Raphaely	100
115	Khoo	114
116	Baida	114
117	Tobias	114
118	Himuro	114
119	Colmenares	114
120	Weiss	100
121	Fripp	100
122	Kaufling	100
123	Vollman	100

员工表

employee_id	last_name	manager_id
100	King	(NULL)
101	Kochhar	100
102	De Haan	100
103	Hunold	102
104	Ernst	103
105	Austin	103
106	Pataballa	103
107	Lorentz	103
108	Greenberg	101
109	Faviet	108
110	Chen	108
111	Sciarra	108
112	Urman	108
113	Popp	108
114	Raphaely	100
115	Khoo	114
116	Baida	114
117	Tobias	114
118	Himuro	114
119	Colmenares	114
120	Weiss	100
121	Fripp	100
122	Kaufling	100
123	Vollman	100

领导表

案例1：查询员工名和上级名

```
1 SELECT
2     e.employee_id,
3     e.last_name,
4     m.`employee_id`,
5     m.`last_name`
6 FROM
7     employees e,
8     employees m
9 WHERE e.`manager_id` = m.`employee_id` ;
```

(六)、测试题

案例1：显示所有员工的姓名，部门号和部门名称

```
1  SELECT
2      last_name,
3      d.department_id,
4      department_name
5  FROM
6      employees e,
7      departments d
8  WHERE e.`department_id` = d.`department_id` ;
```

案例2：查询90号部门员工的 job_id 和90号部门的 location_id

```
1  SELECT
2      d.department_id,
3      e.job_id,
4      d.location_id
5  FROM
6      departments d,
7      employees e
8  WHERE d.`department_id` = e.`department_id`
9      AND d.department_id = 90 ;
```

案例3：选择所有有奖金的员工的 last_name, department_name, location_id, city

```
1  SELECT
2      e.last_name,
3      d.department_name,
4      l.location_id,
5      l.city,
6      e.`commission_pct`
7  FROM
8      employees e,
9      departments d,
10     locations l
11 WHERE e.`department_id` = d.`department_id`
12     AND d.`location_id` = l.`location_id`
13     AND e.`commission_pct` IS NOT NULL ;
```

案例4：选择city在Toronto工作的员工的 last_name, job_id, department_id, department_name

```
1  SELECT
2      e.last_name,
3      e.job_id,
4      e.department_id,
5      d.department_name,
6      l.`city`
7  FROM
8      employees e,
9      departments d,
10     locations l
11 WHERE e.`department_id` = d.`department_id`
12     AND d.`location_id` = l.`location_id`
13     AND l.`city` = 'Toronto' ;
```

案例5：查询每个工种，每个部门的部门名，工种名和最低工资

```
1  SELECT
2      j.job_id,
3      e.department_id,
4      d.department_name,
5      j.job_title,
6      MIN(salary)
7  FROM
8      departments d,
9      employees e,
10     jobs j
11 WHERE d.`department_id` = e.`department_id`
12     AND j.`job_id` = e.`job_id`
13 GROUP BY job_id,
14     department_id ;
```

案例6：查询每个国家下的部门个数大于2的国家编号

```
1  SELECT
2      country_id,
3      COUNT(*) 部门数
4  FROM
5      locations l,
6      departments d
7  WHERE l.`location_id` = d.`location_id`
8  GROUP BY country_id
9  HAVING COUNT(*) > 2 ;
```

案例7：选择指定员工的姓名，员工名，以及他的管理者的姓名和员工号，结果类似于下面的格式

employees	Emp#	manager	Mgr#
kochhar	101	king	100

```

1  SELECT
2      e1.last_name AS "employees",
3      e1.employee_id AS "Emp#",
4      e2.last_name AS manager,
5      e2.employee_id AS "Mgr#"
6  FROM
7      employees e1,
8      employees e2
9  WHERE e1.`manager_id` = e2.`employee_id`
10 AND e1.`employee_id` = 101 ;

```

四、SQL99标准

(一)、语法

```

1  select   查询列表
2  from    表1 别名 【连接类型】
3  join    表2 别名
4  on     连接条件
5  【where 筛选条件】
6  【group by 分组】
7  【having 筛选条件】
8  【order by 排序列表】

```

分类:

内连接: inner

外连接:

左外连接: left 【outer】

右外连接: right 【outer】

全外连接: full 【outer】

交叉连接: cross

(二)、内连接

1. 语法

```
1 select 查询列表
2 from 表1 别名
3 inner join 表2 别名
4 on 连接条件
```

2. 等值连接

案例一：查询员工名，部门名

```
▼ (1)
          ABAP |  

1 SELECT
2   last_name,
3   department_name
4 FROM
5   employees e
6   INNER JOIN departments d
7     ON e.`department_id` = d.`department_id` ;
```

```
▼ (2) employees 与 departments调换位置，结果一样
          ABAP |  

1 ELECT
2   last_name,
3   department_name
4 FROM
5   departments d
6   INNER JOIN employees e
7     ON e.`department_id` = d.`department_id` ;
```

案例二：查询名字中包含e的员工名和工种名（筛选）

```
1 SELECT
2   last_name,
3   job_title
4 FROM
5   employees e
6   INNER JOIN jobs j
7     ON e.`job_id` = j.`job_id`
8 WHERE last_name LIKE '%e%' ;
9
```

案例三：查询部门个数>3的城市名和部门个数（分组+筛选）

- (1) 查询每个城市的部门个数
(2) 在(1)结果上筛选满足条件的

```
1  SELECT
2      city,
3      COUNT(*) 部门个数
4  FROM
5      departments d
6      INNER JOIN locations l
7          ON d.`location_id` = l.`location_id`
8  GROUP BY city
9  HAVING COUNT(*) > 3 ;
10
```

案例四：查询哪个部门的员工个数>3的部门名和员工个数，并按个数降序（添加排序）

- (1) 查询每个部门的员工个数

```
1  SELECT
2      COUNT(*) 员工个数,
3      department_name
4  FROM
5      employees e
6      INNER JOIN departments d
7          ON e.`department_id` = d.`department_id`
8  GROUP BY department_name ;
```

- (2) 在(1)的结果上筛选员工个数>3的记录，并排序

```
1  SELECT
2      COUNT(*) 员工个数,
3      department_name
4  FROM
5      employees e
6      INNER JOIN departments d
7          ON e.`department_id` = d.`department_id`
8  GROUP BY department_name
9  HAVING COUNT(*) > 3
10 ORDER BY COUNT(*) DESC ;
```

案例五：查询员工名，部门名，工种名，并按部门名降序（添加三表连接）

```
1 SELECT
2     last_name,
3     department_name,
4     job_title
5 FROM
6     employees e
7     INNER JOIN departments d
8         ON e.department_id = d.department_id
9     INNER JOIN jobs j
10        ON e.job_id = j.job_id
11 ORDER BY department_name DESC ;
```

(1) 多表(>2)连接的时候，前后表要有连接条件（表有顺序）

(2) from 表1 别名

inner join 表2 别名 on 连接条件1

inner join 表3 别名 on 连接条件2

...

等值连接的特点：

(1) 可以添加排序，分组，筛选

(2) inner 可以忽略

(3) 筛选条件放在where 后面，连接条件放在 on 后面，提高分离性，便于阅读

(4) inner join 连接 和 SQL92 语法中的等值连接效果是一样的，都是查询多表的交集

3. 非等值连接

案例1：查询员工的工资等级

```
1 SELECT
2     salary,
3     grade_level
4 FROM
5     employees e
6     INNER JOIN job_grades j
7         ON e.salary BETWEEN j.lowest_sal
8             AND j.highest_sal ;
```

案例2：查询工资级别的个数>20的个数，并且按工资级别降序

```

1  SELECT
2      COUNT(*),
3      grade_level
4  FROM
5      employees e
6      INNER JOIN job_grades j
7          ON e.`salary` BETWEEN j.`lowest_sal`
8          AND j.`highest_sal`
9  GROUP BY grade_level
10 HAVING COUNT(*) > 20
11 ORDER BY grade_level DESC ;

```

4. 自连接（把原本的表当作二张表或更多张表使用）（自己连接自己）

案例1：查询员工的名字，上级的名字

```

1  SELECT
2      e.last_name AS employees,
3      e.employee_id AS "Emp#",
4      m.last_name AS manager,
5      m.employee_id AS "Mge#"
6  FROM
7      employees e
8      INNER JOIN employees m
9          ON e.`manager_id` = m.`employee_id` ;

```

案例2：查询姓名中包含k的员工名字，上级的名字

```

1  SELECT
2      e.last_name AS employees,
3      e.employee_id AS "Emp#",
4      m.last_name AS manager,
5      m.employee_id AS "Mge#"
6  FROM
7      employees e
8      INNER JOIN employees m
9          ON e.`manager_id` = m.`employee_id` 
10     WHERE e.last_name LIKE '%k%' ;

```

(三)、外连接(外连接的结果=内连接的结果 + 主表有，从表没有 (从表为null))

1. 应用场景：用于查询一个表中有，另一个表没有的记录

主表

id	name	sex	boyfriend_id
1	柳岩	女	8
2	苍老师	女	9
3	Angelababy	女	3
4	热巴	女	2
5	周冬雨	女	9
6	周芷若	女	1
7	岳灵珊	女	9
8	小昭	女	1
9	双儿	女	9
10	王语嫣	女	4
11	夏雪	女	9
12	赵敏	女	1

boys表

从表

id	boyName	userCP
1	张无忌	100
2	鹿晗	800
3	黄晓明	50
4	段誉	300

id	name	boyname
6	周芷若	张无忌
8	小昭	张无忌
12	赵敏	张无忌
4	热巴	鹿晗
3	Angelababy	黄晓明
10	王语嫣	段誉
1	柳岩	(NULL)
2	苍老师	(NULL)
5	周冬雨	(NULL)
7	岳灵珊	(NULL)
9	双儿	(NULL)
11	夏雪	(NULL)

主表，从表的交集-->内连接

主表有，从表没有（从表为null）

2. 特点

- (1) 外连接的查询结果为主表中的所有记录

如果从表中有和主表匹配的，则显示匹配的值

如果从表中没有主表匹配的，则显示null值

- (2) 左外连接， left join 左边的是 主表

右外连接， right join 右边的是 主表

- (3) 左外和右外 交换两个表的顺序，（主表和从表不会改变），可以实现同样的效果

3. 主表：你要查询的信息主要来自于哪个表，则就是主表

4. 案例

- (1) 案例1：查询男朋友 不在男神表的女神名

▼ 左外连接 left outer

ABAP |

```
1 SELECT
2   b.name,
3   bo.*
4 FROM
5   beauty b
6   LEFT OUTER JOIN boys bo
7     ON b.`boyfriend_id` = bo.`id`
8 WHERE bo.id IS NULL ;
```

▼ 右外连接 right outer

ABAP |

```
1 SELECT
2   b.name,
3   bo.*
4 FROM
5   boys bo
6   RIGHT OUTER JOIN beauty b
7     ON b.`boyfriend_id` = bo.`id`
8 WHERE bo.id IS NULL ;
```

(2) 案例2：查询哪个部门没有员工

▼ 左外连接 left outer

ABAP |

```
1 SELECT
2   d.department_name,
3   d.department_id
4 FROM
5   departments d
6   LEFT OUTER JOIN employees e
7     ON d.`department_id` = e.`department_id`
8 WHERE e.`employee_id` IS NULL ;
```

▼ 右外连接 right outer

ABAP |

```
1 SELECT
2   d.department_name,
3   d.department_id
4 FROM
5   employees e
6   RIGHT OUTER JOIN departments d
7     ON d.`department_id` = e.`department_id`
8 WHERE e.`employee_id` IS NULL ;
```

5. 全外连接 = 内连接的结果 + 表1中有但表2中没有的 + 表2中有但表1中没有

(四)、交叉连接(使用99语法标准实现笛卡尔乘积)

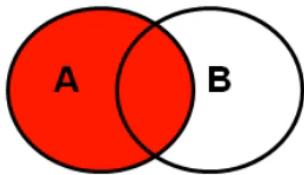
```
1  SELECT *
2  FROM beauty b
3  CROSS JOIN boys bo;
```

结果是48行， beauty 表匹配 boys 表的每一行

(五)、SQL92 和 SQL 99

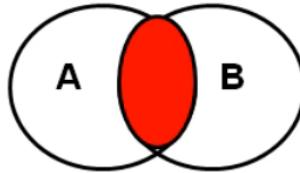
功能：sql 99 支持更多

可读性：sql99 实现连接条件和筛选条件的分离，可读性较高



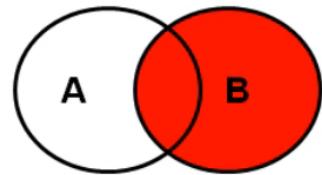
```
SELECT <select_list>
FROM A
LEFT JOIN B
ON A.key=B.key
```

左外连接



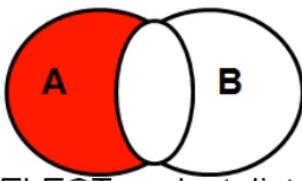
```
SELECT <select_list>
FROM A
INNER JOIN B
ON A.key=B.key
```

内连接

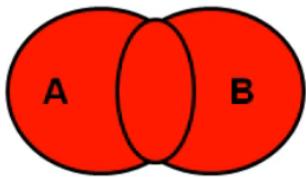


```
SELECT <select_list>
FROM A
RIGHT JOIN B
ON A.key=B.key
```

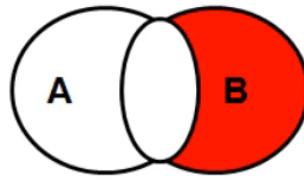
右外连接



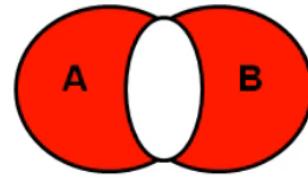
SELECT <select_list>
FROM A
LEFT JOIN B
ON A.key=B.key
WHERE B.key is null;



SELECT <select_list>
FROM A
FULL JOIN B
ON A.key=B.key



SELECT <select_list>
FROM A
RIGHT JOIN B
ON A.key=B.key
WHERE A.key is null;



SELECT <select_list>
FROM A
FULL JOIN B
ON A.key=B.key
WHERE A.key is null
OR B.key is null;

(六)、测试题

案例1：查询编号 >3 的女神的男朋友信息，如果有则列出详细，如果没有，用null填充（主表是beauty，从表是boys）

```
▼ 左外连接 ABAP |  
1 SELECT  
2   b.`name` ,  
3   bo.*  
4 FROM  
5   beauty b  
6   LEFT OUTER JOIN boys bo  
7     ON bo.`id` = b.`boyfriend_id`  
8 WHERE b.`id` > 3 ;
```

案例2：查询哪个城市没有部门(locations为主表，departments为从表，让从表的主键为null)

左外连接

ABAP |

```
1 SELECT
2   city,
3   d.`department_id`
4 FROM
5   locations l
6   LEFT OUTER JOIN departments d
7     ON l.`location_id` = d.`location_id`
8 WHERE d.department_id IS NULL ;
```

案例3：查询部门名为SAL或IT的员工信息(departments为主表， employees为从表， (因为有些部门没有员工))

右外连接

ABAP |

```
1 SELECT
2   e.*,
3   d.`department_name`
4 FROM
5   employees e
6   RIGHT OUTER JOIN departments d
7     ON e.`department_id` = d.`department_id`
8 WHERE d.`department_name` = 'SAL'
9   OR d.`department_name` = 'IT' ;
```

进阶7： 子查询

一、含义

出现在其他语句（删除，添加，查询）中的select语句，称为 内查询或子查询；外部的语句可以是

select , insert , update , delete 等语句，一般为查询语句，称为主查询 或 外查询

概念：出现在其他语句内部的select语句，称为子查询或内查询

内部嵌套其他select语句的查询，称为外查询或主查询

示例：

```
select first_name from employees where  
department_id in (  
    select department_id from departments  
    where location_id=1700  
)
```

↑
主查询或外查询

↓
内查询或子查询

1. 分类

(1) 按子查询出现的位置：

select 后面： 仅仅支持标量子查询

from 后面： 支持 表子查询

where 或 having 后面： ★

标量子查询（一行）

列子查询（多行）

行子查询（多列多行）

exists 后面（相关子查询）： 表子查询

(2) 按结果集的行列数不同分类：

标量子查询（结果集只有一行一列）

列子查询（结果集只有一列多行）

行子查询（结果集只有一行多列，<多列多行 也可以>）

二、where 或 having 后面的子查询

(一)、分类：

1. 标量子查询（单行子查询）
2. 列子查询（多行子查询）
3. 行子查询（多列多行子查询）

特点：

1. 子查询放在小括号内
2. 子查询一般放在条件的右侧
3. 标量子查询，一般搭配着单行操作符使用 > < >= <= = <>
4. 列子查询，一般搭配着多行操作符使用， in any/some all
5. 子查询的执行优先于主查询执行，主查询的条件用到了子查询的结果

(二)、标量子查询(单行子查询)

a. 案例

1. 谁的工资比 Abel 高？

- (1) 查询Abel的工资

```

1  SELECT
2      salary
3  FROM
4      employees
5  WHERE last_name = 'Abel';

```

- (2) 查询员工的信息，满足salary > (1)的结果

```
1  SELECT
2    *
3  FROM
4    employees
5  WHERE salary >
6    (SELECT
7      salary
8    FROM
9      employees
10     WHERE last_name = 'Abel') ;
```

2. 返回job_id与141号员工相同，salary比143号员工多的员工姓名，job_id和工资

(1) 查询141号员工的job_id

```
1  SELECT
2    job_id
3  FROM
4    employees
5  WHERE employee_id = 141
```

(2) 查询143号员工的salary

```
1  SELECT
2    salary
3  FROM
4    employees
5  WHERE employee_id = 143 ;
```

(3) 查询员工的姓名，job_id和工资，要求job_id=(1)并且salary >(2)

```
1  SELECT
2      last_name,
3      job_id,
4      salary
5  FROM
6      employees
7  WHERE salary >
8      (SELECT
9          salary
10     FROM
11      employees
12  WHERE employee_id = 143)
13  AND job_id =
14  (SELECT
15      job_id
16  FROM
17      employees
18  WHERE employee_id = 141) ;
```

3. 返回公司工资最少的员工的last_name, job_id 和 salary

(1) 查询公司的最少工资

```
1  SELECT MIN(salary) FROM employees;
```

(2) 查询last_name, job_id 和 salary ,要求salary = (1)

```
1  SELECT
2      last_name,
3      job_id,
4      salary
5  FROM
6      employees
7  WHERE salary =
8      (SELECT
9          MIN(salary)
10     FROM
11      employees) ;
```

4. 查询最低工资大于50号部门最低工资的部门id和其最低工资

(1) 查询50号部门的最低工资

```
1 SELECT
2     MIN(salary)
3 FROM
4     employees
5 WHERE department_id = 50 ;
```

(2) 查询每个部门的最低工资

```
1 SELECT
2     MIN(salary)
3 FROM
4     employees
5 GROUP BY department_id ;
```

(3) 在 (2) 的基础上筛选，满足 $\min(\text{salary}) > (1)$

```
1 SELECT
2     department_id,
3     MIN(salary)
4 FROM
5     employees
6 GROUP BY department_id
7 HAVING MIN(salary) >
8     (SELECT
9         MIN(salary)
10        FROM
11        employees
12       WHERE department_id = 50);
```

(三)、列子查询 (多行子查询)

1. 返回多行
2. 使用多行操作符

操作符	含义
in / not in	等于列表中的任意一个
any /some	和子查询返回的某一个值比较
all	和子查询返回的所有值比较

a. 案例

1. 返回location_id 是1400或1700的部门中的所有员工姓名

(1) 查询location_id是1400或1700的部门编号

```
1  SELECT DISTINCT
2      department_id
3  FROM
4      departments
5  WHERE location_id IN (1400, 1700);
```

(2) 查询员工姓名，要求部门号是 (1) 列表中的某一个

▼ (1) in

```
1  SELECT
2      last_name
3  FROM
4      employees
5  WHERE department_id IN
6      (SELECT DISTINCT
7          department_id
8      FROM
9          departments
10     WHERE location_id IN (1400, 1700));
```

ABAP |

▼ (2) in 替换为 = any

```
1  SELECT
2      last_name
3  FROM
4      employees
5  WHERE department_id =ANY
6      (SELECT DISTINCT
7          department_id
8      FROM
9          departments
10     WHERE location_id IN (1400, 1700));
```

ABAP |

2. 返回其他工种中比job_id为 'IT_PRPC' 工种任一工资低的员工的员工号，姓名，job_id，以及salary

(1)查询job_id为 'IT_PRPC' 工种的任一工资

```
1  SELECT DISTINCT
2    salary
3  FROM
4    employees
5 WHERE job_id = 'IT_PROG' ;
6
```

(2)查询员工号, 姓名, job_id, 以及salary , 要求 salary < (1) 的任意一个



(1) some()

ABAP |

```
1  SELECT
2    employee_id,
3    last_name,
4    job_id,
5    salary
6  FROM
7    employees
8 WHERE salary < SOME
9   (SELECT
10    DISTINCT salary
11   FROM
12    employees
13 WHERE job_id = 'IT_PROG') AND job_id <> 'IT_PROG';
```



(2) <some() 替换为 < MAX()

ABAP |

```
1
2    employee_id,
3    last_name,
4    job_id,
5    salary
6  FROM
7    employees
8 WHERE salary <
9   (SELECT
10    MAX(salary)
11   FROM
12    employees
13 WHERE job_id = 'IT_PROG')
14 AND job_id <> 'IT_PROG' ;
```

3. 返回其他工种中比job_id为 'IT_PRPC' 工种所有工资低的员工的员工号, 姓名, job_id, 以及salary

→217)

→217)

(1) < ALL

ABAP |

```
1 SELECT
2   employee_id,
3   last_name,
4   job_id,
5   salary
6 FROM
7   employees
8 WHERE salary < ALL
9   (SELECT DISTINCT
10    salary
11   FROM
12    employees
13   WHERE job_id = 'IT_PROG')
14 AND job_id >> 'IT_PROG' ;
```

(2) < ALL 替换为 < MIN()

ABAP |

```
1 SELECT
2   employee_id,
3   last_name,
4   job_id,
5   salary
6 FROM
7   employees
8 WHERE salary <
9   (SELECT
10    MIN(salary)
11   FROM
12    employees
13   WHERE job_id = 'IT_PROG')
14 AND job_id >> 'IT_PROG' ;
```

(四) 行子查询 (结果集为一行多列或多行多列)

案例：查询员工编号最小并且工资最高的员工信息（可能会不存在）

(1) 查询最小的员工编号

```
1 SELECT MIN(employee_id) FROM employees;
2
```

(2) 查询最高的工资

```
1 SELECT MAX(salary) FROM employees;
```

(3) 查询员工信息

▼ (1)

ABAP |

```
1 SELECT
2   *
3   FROM
4     employees
5 WHERE employee_id =
6   (SELECT
7     MIN(employee_id)
8   FROM
9     employees)
10 AND salary =
11 (SELECT
12   MAX(salary)
13   FROM
14     employees) ;
```

▼ (2) 两个条件都用 = 连接，可以将两个条件 替换为一个大的条件

ABAP |

```
1 SELECT *
2   FROM employees
3 WHERE (employee_id,salary) = (
4   SELECT MIN(employee_id),MAX(salary)
5   FROM employees
6   );
```

三、select 后面 (仅仅支持标量子查询，即一行一列)

案例：查询每个部门的员工个数

```
1 SELECT *,(
2   SELECT COUNT(*)
3   FROM employees e
4   WHERE e.department_id=d.`department_id`
5   ) 个数
6   FROM departments d;
```

案例：查询员工号=102的部门名

```
1  SELECT (
2    SELECT d.department_name
3    FROM departments d
4    INNER JOIN employees e
5    ON e.department_id = d.department_id
6    WHERE e.employee_id = 102
7  ) 部门名;
```

▼ where 后面的标量子查询

ABAP |

```
1  SELECT
2    department_name 部门号
3  FROM
4    departments d
5  WHERE d.`department_id` =
6    (SELECT
7      department_id
8    FROM
9      employees
10   WHERE employee_id = 102) ;
```

四、from 后面

1. 注意：将子查询结果充当一张表，要求必须起别名

2. 案例：查询每个部门的平均工资的工资等级

(1) 查询每个部门的平均工资

```
1  SELECT AVG(salary),department_id
2    FROM employees
3    GROUP BY department_id;
```

(2) 连接 (1) 的结果集和job_grades 表，筛选条件平均工资 between lowest_sal and highest_sal

```
1  SELECT
2    ag_dep.*,
3    g.grade_level
4  FROM
5    (SELECT AVG(salary) ag,department_id
6    FROM employees
7    GROUP BY department_id) ag_dep
8    INNER JOIN job_grades g
9    ON ag_dep.ag BETWEEN g.lowest_sal AND g.highest_sal ;
```

between and

AVG(salary)	department_id
7000.000000	(NULL)
4400.000000	10
9500.000000	20
4150.000000	30
6500.000000	40
3475.555556	50
5760.000000	60
10000.000000	70
8955.882353	80
19333.333333	90
8600.000000	100
10150.000000	110

grade_level	lowest_sal	highest_sal
A	1000	2999
B	3000	5999
C	6000	9999
D	10000	14999
E	15000	24999
F	25000	40000
(NULL)	(NULL)	(NULL)

job_grades表

ag_dep表 (子查询的结果集)

五、exists 后面 (相关子查询) (还没弄懂)

(一)、语法

exists (完整的查询语句)

结果：1 或 0

(二)、案例

1. 查询有员工的部门名

```

▼ (1) 内连接 ABAP |
1 SELECT DISTINCT department_name
2 FROM employees e
3 INNER JOIN departments d
4 ON e.`department_id`=d.`department_id`;

```

▼ (2) exists

ABAP |

```

1 SELECT department_name
2 FROM departments d
3 WHERE EXISTS (
4     SELECT *
5         FROM employees e,departments d
6         WHERE d.`department_id`=e.`department_id`
7 );

```

employee_id	first_name	last_name	email	phone_number	job_id	salary	commission_pct	manager_id	department_id	hiredate	department_id	department_name
200	Jennifer	Whalen	JWHALEN	515.123.4444	AD_ASST	4400.00	(NULL)	101	10	2016-03-03 00:00:00	10	Adm
201	Michael	Hartstein	MHARTSTE	515.123.5555	MK_MAN	13000.00	(NULL)	100	20	2016-03-03 00:00:00	20	Mar
202	Pat	Fay	PFAY	603.123.6666	MK_REP	6000.00	(NULL)	201	20	2016-03-03 00:00:00	20	Mar
114	Den	Raphaely	DRAPHEAL	515.127.4561	FU_MAN	11000.00	(NULL)	100	30	2000-09-09 00:00:00	30	Pur
115	Alexander	Rhoo	AKRHO	515.127.4562	FU_CLERK	3100.00	(NULL)	114	30	2000-09-09 00:00:00	30	Pur
116	Shelli	Baida	SBaida	515.127.4563	FU_CLERK	2900.00	(NULL)	114	30	2000-09-09 00:00:00	30	Pur
117	Sigal	Tobias	STOBIA	515.127.4564	FU_CLERK	2800.00	(NULL)	114	30	2000-09-09 00:00:00	30	Pur
118	Guy	Himuro	GHIMURO	515.127.4565	FU_CLERK	2600.00	(NULL)	114	30	2000-09-09 00:00:00	30	Pur
119	Karen	Colmenares	KCOLMENNA	515.127.4566	FU_CLERK	2500.00	(NULL)	114	30	2000-09-09 00:00:00	30	Pur
203	Susan	Mavris	SMAVRIS	515.123.7777	HR REP	6500.00	(NULL)	101	40	2016-03-03 00:00:00	40	Hum
120	Matthew	Weiss	MWEISS	650.123.1234	ST_MAN	8000.00	(NULL)	100	50	2004-02-06 00:00:00	50	Shi
121	Adam	Fripp	AFRIPP	650.123.2234	ST_MAN	8200.00	(NULL)	100	50	2004-02-06 00:00:00	50	Shi
122	Payam	Kaufling	PKAUFLIN	650.123.3234	ST_MAN	7900.00	(NULL)	100	50	2004-02-06 00:00:00	50	Shi



▼ (3) in

ABAP |

```

1 SELECT department_name
2 FROM departments d
3 WHERE d.`department_id` IN (
4     SELECT department_id
5         FROM employees e
6 );

```

2. 查询没有女朋友的男神信息

▼ (1) not in

ABAP |

```

1 SELECT bo.*
2 FROM boys bo
3 WHERE bo.id NOT IN (
4     SELECT boyfriend_id
5         FROM beauty
6 );

```

```

1  SELECT bo.*  

2  FROM boys bo  

3  WHERE NOT EXISTS (  

4      SELECT boyfriend_id  

5      FROM beauty b  

6      WHERE bo.`id`=b.`boyfriend_id`  

7  );

```

(三)、exists(子查询) , 这里子查询一般为连接查询

exists(查询语句), 判断子查询的结果有没有值, 结果为0或1 (相当于 bool 类型)

六、测试题

1. 查询和 zlotkey 相同部门的员工姓名和工资

```

1  SELECT last_name,salary  

2  FROM employees e  

3  WHERE e.`department_id` = (  

4      SELECT department_id  

5      FROM employees  

6      WHERE last_name = 'Zlotkey'  

7  );

```

2. 查询工资比公司平均工资高的员工的员工号, 姓名和工资

(1) 公司的平均工资

```

1  SELECT AVG(salary)  

2  FROM employees;

```

(2) 查询比 (1) 高的员工号, 姓名, 工资

```

1  SELECT employee_id,last_name,salary  

2  FROM employees  

3  WHERE salary > (  

4      SELECT AVG(salary)  

5      FROM employees  

6  );

```

3. 查询各部门中工资比本部门平均工资高的员工的员工号, 姓名和工资

department_id	avg(salary)
(NULL)	7000.000000
10	4400.000000
20	9500.000000
30	4150.000000
40	6500.000000
50	3475.555556
60	5760.000000
70	10000.000000
80	8955.882353
90	19333.333333
100	8600.000000
110	10150.000000

每个部门的平均工资

avg_dep

employee_id	last_name	salary	department_id
100	King	24000.00	90
101	Kochhar	17000.00	90
102	De Haan	17000.00	90
103	Hunold	9000.00	60
104	Ernst	6000.00	60
105	Austin	4800.00	60
106	Pataballa	4800.00	60
107	Lorentz	4200.00	60
108	Greenberg	12000.00	100
109	Faviet	9000.00	100
110	Chen	8200.00	100
111	Sciarra	7700.00	100
112	Urman	7800.00	100
113	Popp	6900.00	100
114	Raphaely	11000.00	30
115	Khoo	3100.00	30
116	Baida	2900.00	30
117	Tobias	2800.00	30

每个人的工资

(1) 每个部门的平均工资

```

1  SELECT department_id,AVG(salary)
2  FROM employees
3  GROUP BY department_id;
```

(2) 每个部门比本部门平均工资高的员工号, 姓名, 工资

```

▼ (1) 连接条件用 in ABAP |
1  SELECT employee_id, last_name, salary
2  FROM employees e
3  INNER JOIN (
4    SELECT department_id, AVG(salary) avgg
5    FROM employees
6    GROUP BY department_id) AS avg_dep
7  ON e.department_id IN (avg_dep.department_id)
8  WHERE e.salary > avg_dep.avgg;
```

▼ (2) 连接条件用 =

ABAP |

```
1 SELECT employee_id, last_name, salary, e.`department_id`  
2 FROM employees e  
3 INNER JOIN (  
4     SELECT department_id, AVG(salary) avgg  
5     FROM employees  
6     GROUP BY department_id) AS avg_dep  
7 ON e.department_id = avg_dep.department_id  
8 WHERE e.salary > avg_dep.avgg;
```

4. 查询和姓名中含字母u的员工在相同部门的员工的员工号和姓名

(1) 查询姓名中包含u的员工的部门

```
1 SELECT DISTINCT department_id  
2 FROM employees e  
3 WHERE e.last_name LIKE '%u%';
```

(2) 查询和 (1) 任一部门号相同的员工的员工号和姓名

```
1 SELECT e2.employee_id, e2.last_name  
2 FROM employees e2  
3 WHERE e2.department_id IN (  
4     SELECT DISTINCT department_id  
5     FROM employees e  
6     WHERE e.last_name LIKE '%u%'  
7 );
```

5. 查询在部门的location_id=1700的部门工作的员工的员工号

(1) 查询location_id = 1700 的部门号

```
1 SELECT department_id  
2 FROM departments d  
3 WHERE d.`location_id` = 1700;
```

(2) 查询部门号在 (1) 中的员工的员工号

▼ (1)筛选条件用 in

ABAP |

```
1 SELECT employee_id  
2 FROM employees e  
3 WHERE e.department_id IN (  
4     SELECT department_id  
5     FROM departments d  
6     WHERE d.`location_id` = 1700  
7 );
```

▼ (2) 筛选条件用 = ANY()

ABAP |

```
1 SELECT employee_id  
2 FROM employees e  
3 WHERE e.department_id = ANY (  
4     SELECT department_id  
5     FROM departments d  
6     WHERE d.`location_id` = 1700  
7 );
```

6. 查询管理者是k_ing 的员工姓名和工资

(1) 查询姓名为k_ing的员工号

```
1 SELECT employee_id  
2 FROM employees e  
3 WHERE last_name = 'k_ing'
```

(2) 查询manger_id = (1)的员工姓名和工资

```
1 SELECT e2.last_name,e2.salary  
2 FROM employees e2  
3 WHERE e2.manager_id IN (  
4     SELECT employee_id  
5     FROM employees e  
6     WHERE last_name = 'k_ing'  
7 );
```

7. 查询工资最高的员工的姓名，要求first_name和last_name显示为一列，列明为 姓名

(1) 查询最高的工资

```
1 SELECT MAX(salary)  
2 FROM employees;
```

(2)查询工资等于 (1) 的员工的姓名

```
1 SELECT CONCAT(first_name, last_name) AS "姓.名"
2 FROM employees e
3 WHERE salary = (
4     SELECT MAX(salary)
5     FROM employees
6 );
```

七、测试题

1. 查询工资最低的员工信息：last_name,salary

(1) 查询最低的工资

```
1 SELECT MIN(salary) FROM employees
```

(2) 查询工资= (1) 的last_name,salary

```
1 SELECT last_name,salary
2 FROM employees e
3 WHERE e.salary = (
4     SELECT MIN(salary) FROM employees
5 );
```

2. 查询平均工资最低的部门信息

方式一

(1) 查询每个部门的平均工资

```
1 SELECT AVG(salary),department_id
2 FROM employees
3 GROUP BY department_id
```

(2) 最低的平均工资

```
1 SELECT MIN(avgg)
2 FROM (
3     SELECT AVG(salary) avgg
4     FROM employees
5     GROUP BY department_id
6 ) avgg_dep
```

(3) 平均工资= (2) 的department_id

▼ <1> from后面上子查询, where后面上子查询

ABAP |

```
1  SELECT avg_dep.department_id
2  FROM (
3      SELECT AVG(salary) avgg,department_id
4      FROM employees
5      GROUP BY department_id
6  ) avg_dep
7  WHERE avg_dep.avgg = (
8      SELECT MIN(avgg)
9      FROM (
10         SELECT AVG(salary) avgg
11         FROM employees
12         GROUP BY department_id
13     ) avgg_dep
14 )
```

▼ <2> having后面上子查询

ABAP |

```
1  SELECT AVG(salary),department_id
2  FROM employees
3  GROUP BY department_id
4  HAVING AVG(salary) = (
5      SELECT MIN(avgg)
6      FROM (
7          SELECT AVG(salary) avgg
8          FROM employees
9          GROUP BY department_id
10     ) avgg_dep
11 )
12 )
```

(4) 部门编号= (3) 的信息

▼ <1>

ABAP |

```
1 SELECT *
2 FROM departments d
3 WHERE d.department_id = (
4     SELECT avg_dep.department_id
5     FROM (
6         SELECT AVG(salary) avgg,department_id
7         FROM employees
8         GROUP BY department_id
9     ) avg_dep
10    WHERE avg_dep.avgg = (
11        SELECT MIN(avgg)
12        FROM (
13            SELECT AVG(salary) avgg
14            FROM employees
15            GROUP BY department_id
16        ) avgg_dep
17    )
18 );
```

▼ <2>

ABAP |

```
1 SELECT *
2 FROM departments d
3 WHERE d.department_id = (
4     SELECT department_id
5     FROM employees
6     GROUP BY department_id
7     HAVING AVG(salary) = (
8         SELECT MIN(avgg)
9         FROM (
10            SELECT AVG(salary) avgg
11            FROM employees
12            GROUP BY department_id
13        ) avgg_dep
14    )
15 )
16 );
```

方式二

- (1) 查询各部门的平均工资，并按平均工资升序排列，使用分页查询，显示第一行（即最低平均工资对应的部门号）

```
1 SELECT department_id  
2 FROM employees  
3 GROUP BY department_id  
4 ORDER BY AVG(salary) ASC  
5 LIMIT 0,1
```

(2) 部门号= (1) 的部门信息

```
1 SELECT d.*  
2 FROM departments d  
3 WHERE d.department_id = (  
4     SELECT department_id  
5     FROM employees  
6     GROUP BY department_id  
7     ORDER BY AVG(salary) ASC  
8     LIMIT 0,1  
9 );
```

3. 查询平均工资最低的部门信息和该部门的平均工资

(1) 查询各部门的平均工资，并按平均工资升序排列，使用分页查询，显示第一行（即最低平均工资对应的部门号）

```
1 SELECT department_id ,AVG(salary)  
2 FROM employees  
3 GROUP BY department_id  
4 ORDER BY AVG(salary) ASC  
5 LIMIT 0,1
```

(2) 部门号= (1) 的部门号的部门信息

```
1 SELECT d.*  
2 FROM departments d  
3 WHERE d.department_id = (  
4     SELECT department_id  
5     FROM employees  
6     GROUP BY department_id  
7     ORDER BY AVG(salary) ASC  
8     LIMIT 0,1  
9 );
```

(3) 连接 (1) (2)

```
1  SELECT de_min.* ,min_sa.ag 平均工资
2  FROM (
3      SELECT department_id ,AVG(salary) ag
4      FROM employees
5      GROUP BY department_id
6      ORDER BY AVG(salary) ASC
7      LIMIT 0,1
8  ) min_sa
9  INNER JOIN (
10     SELECT d. *
11     FROM departments d
12     WHERE d.department_id = (
13         SELECT department_id
14         FROM employees
15         GROUP BY department_id
16         ORDER BY AVG(salary) ASC
17         LIMIT 0,1
18     )
19 ) de_min
20 ON de_min.department_id = min_sa.department_id;
```

4. 查询平均工资最高的job信息

(1) 查询平均工资最高的job_id

```
1  SELECT job_id
2  FROM employees
3  GROUP BY job_id
4  ORDER BY AVG(salary) DESC
5  LIMIT 0,1
```

(2) 工种号= (1) 的信息

```
1  SELECT j. *
2  FROM jobs j
3  WHERE j.job_id = (
4      SELECT job_id
5      FROM employees
6      GROUP BY job_id
7      ORDER BY AVG(salary) DESC
8      LIMIT 0,1
9  );
```

5. 查询平均工资高于公司的平均工资的部门有哪些?

(1) 查询公司的平均工资

```
1 SELECT AVG(salary) FROM employees
```

(2) 查询每个部门的平均工资

```
1 SELECT AVG(salary) ag,department_id d_id  
2 FROM employees  
3 GROUP BY department_id
```

(3) 在 (2) 的基础上, 查询部门平均工资>(1)的部门

▼ <1>from 子查询, where后面子查询

ABAP |

```
1 SELECT de_sal.d_id  
2 FROM (   
3     SELECT AVG(salary) ag,department_id d_id  
4     FROM employees  
5     GROUP BY department_id  
6 ) de_sal  
7 WHERE de_sal.ag >  
8     (SELECT AVG(salary) FROM employees  
9 );
```

▼ <2>having 子查询

ABAP |

```
1 SELECT department_id d_id  
2 FROM employees  
3 GROUP BY department_id  
4 HAVING AVG(salary) >  
5     (SELECT AVG(salary) FROM employees  
6 );
```

6. 查询出公司中所有的manager信息

(1)查询所有的manager_id(不为null)

```
1 SELECT DISTINCT manager_id FROM employees  
2 WHERE manager_id IS NOT NULL
```

(2) 查询员工号 = (1) 的信息

```
1 SELECT e.* FROM employees e  
2 WHERE e.employee_id IN (  
3     SELECT DISTINCT manager_id FROM employees  
4     WHERE manager_id IS NOT NULL  
5 );
```

7. 各个部门中 最高工资中最低的那个部门 的最低工资是多少

(1) 查询每个部门的最高工资

```
1  SELECT MAX(salary) max_sa,department_id FROM employees  
2  GROUP BY department_id
```

(2)查询 (1) 中的最小的工资对应的部门号

```
1  SELECT department_id  
2  FROM employees  
3  GROUP BY department_id  
4  ORDER BY MAX(salary) ASC  
5  LIMIT 0,1
```

(3)查询每个部门的最低工资

```
1  SELECT MIN(salary ),department_id  
2  FROM employees  
3  GROUP BY department_id
```

(4) 查询(3)中department_id= (2) 的信息

<1> from 后面的子查询, inner内连接

ABAP |

```
1  SELECT min_sal.*  
2  FROM (  
3      SELECT MIN(salary ) ag_min,department_id  
4      FROM employees  
5      GROUP BY department_id  
6  ) min_sal  
7  INNER JOIN (  
8      SELECT department_id  
9      FROM employees  
10     GROUP BY department_id  
11     ORDER BY MAX(salary) ASC  
12     LIMIT 0,1  
13  ) max_min_dep  
14  ON max_min_dep.department_id IN (min_sal.department_id )
```

```

1  SELECT MIN(salary)
2  FROM employees e
3  WHERE e.department_id = (
4      SELECT department_id
5      FROM employees
6      GROUP BY department_id
7      ORDER BY MAX(salary) ASC
8      LIMIT 0,1
9  )

```

8. 查询平均工资最高的部门的 manager 的详细信息：last_name, department_id, email ,salasry

(1) 查询每个部门的平均工资

```

1  SELECT AVG(salary),department_id
2  FROM employees
3  GROUP BY department_id

```

(2) 在 (1) 上最高的工资对应的部门号

```

1  SELECT department_id
2  FROM employees
3  GROUP BY department_id
4  ORDER BY AVG(salary) DESC
5  LIMIT 0,1

```

(3) 将employees 和 departments 连接查询，筛选条件是 (2)

部门的领导只有一个，员工的领导有多个

```

1  SELECT last_name,d.department_id,email,salary
2  FROM employees e
3  INNER JOIN departments d
4  ON d.manager_id = e.employee_id
5  WHERE d.department_id = (
6      SELECT department_id
7      FROM employees
8      GROUP BY department_id
9      ORDER BY AVG(salary) DESC
10     LIMIT 0,1
11 );

```

进阶8：分页查询

一、语法

1. 应用场景：当要显示的数据，一页显示不全，需要分页提交sql请求

	执行顺序
1 select 查询列表	(7)
2 from 表1	(1)
3 【【连接类型】 join 表2	(2)
4 on 连接条件	(3)
5 where 筛选条件	(4)
6 group by 分组字段	(5)
7 having 分组后的筛选	(6)
8 order by 排序的字段】	(8)
9 limit offset,size;	(9)

offset : 要显示条目的起始索引（起始索引从0开始）

size : 要显示的条目

offset 的值从0(第一条信息)开始,可以省略

limit 语法排在最后，执行在最后

(5) (6) 都为虚拟的表，(7) 将最后的表显示出来后，(8) 排序在显示的表的基础上进行

二、案例

(一)、查询前5条员工信息

```
1 SELECT * FROM employees LIMIT 0,5;
```

(二)、查询有奖金的员工信息，并且工资较高的前10名显示出来

```
1 SELECT
2   *
3   FROM
4     employees
5   WHERE commission_pct IS NOT NULL
6   ORDER BY salary DESC
7   LIMIT 0, 10 ;
```

2. 用子查询

(1) 有奖金的员工信息

```
1 SELECT * FROM employees WHERE commission_pct IS NOT NULL
```

(2) 在 (1) 基础上，按工资排名

```
1 SELECT *
2 FROM (SELECT * FROM employees WHERE commission_pct IS NOT NULL) sall
3 ORDER BY salary DESC
```

(3) 取前10名

```
1 SELECT *
2 FROM (
3   SELECT *
4     FROM (SELECT * FROM employees WHERE commission_pct IS NOT NULL) sall
5   ORDER BY salary DESC
6 ) high_sall
7 LIMIT 0,10;
```

(三)、查询第11条-第25条的员工信息

```
1 SELECT * FROM employees LIMIT 10,15;
```

三、特点

1. limit 语句放在查询语句的最后

2. 公式

要显示的页数 page , 每页的条目数 size

```
1 select 查询列表
2 from 表
3 limit (page-1)*size , size ;
```

size = 10

第1页	0
第2页	10
第3页	20