=使用管理員身分去執行停止mysql

```
#停止
net stop mysql80
#啟動
net start mysql80
#退出

exit
crt+c
mysql -h localhost -P 3306 -u root -p
# mysql -h 主機名 -P 端口號 -u 用戶名 -p密碼

#windows查看mysql安裝路徑
show variables like "%char%";
#設置路徑跟環境變量
C:\Program Files\MySQL\MySQL Server 8.0\bin
mysql -h localhost -P 3306 -u root -p1234
-- 查詢資料庫使用者
SELECT User FROM mysql.user;
-- 查詢資料庫使用者（刪除重複的)
SELECT DISTINCT User FROM mysql.user;
-- 查詢資料庫使用者與來源主機
SELECT User, Host FROM mysql.user;
-- 查詢使用者的細部權限
SHOW GRANTS FOR 'officeguide'@'localhost';
#查看當前所有的數據庫
show databases;
#打開指定表
use bookstore;
#查看目前表格
show tables;
#查其他所有表單
show tables from mysql;

#確認自己在那個數據庫
select database();

#創建table
create stuinfo(
id int,
name varchar(20));

show tables;
#看表結構
desc stuinfo;

select * from stuinfo;
insert into stuinfo(id,name) values(1,'john');
insert into stuinfo(id,name) values(2,'rose');

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

```

```
delete from stuinfo where id=1;
#看表mysql版本
select version();
#dos下執行
Mysql --version
Mysql -V
#單行注釋
--單行注釋
/*
多行注釋
*/
```

#進階查詢

```
/*
語法
select 查詢列表 from 表名;

類似於 :System.out.println();

特點:
1.查表列表可以是:表中字段,常量,表達式,函數
2.查詢結果可以是一個虛擬的表格
*/
#1.查詢表中的單個字段
select last_name from employee;
#2.查詢表中的多個字段
select last_name,salary ,email =from employee;
#方式2
use coffeedb;
select * from answer;
select 'book_id','title','author' from book;
# 查詢常量值
select 60;
# 查詢表達式
select  'Espresso';
# 查詢函數
select version();

# 起別名(便於理解,有重名時,可以使用別名區分開來)
select  'Espresso' as result;
#方式1用 as 或空格
select name as  姓, salary as 薪水 from employee;
select name   姓, salary  薪水 from employee;
#案例:查詢salary,顯示結果為out put 有空格用''即可判別
select salary as 'out put' from employee;
#下列一行無法判別,無''
select salary as out put from employee;
#案例:查詢員工表中涉及到所有的部門編號
select department_id from employee;

INSERT INTO `coffeedb`.`employee` (`id`, `name`, `JOINING_DATE`, `salary`,
`ssn`) VALUES ('3', 'betty ', '2021-03-15', '62000.00', 'A1236547');
INSERT INTO `coffeedb`.`employee` (`id`, `name`, `JOINING_DATE`, `salary`,
`ssn`) VALUES ('4', 'John', '2021-03-14', '57000.00', 'B3255123');
```

```
#去除重複的
select distinct department_id from employee;
#去除重複的
select distinct Salary from employee;
#+功用
/*
java中的+號功用
1.運算符號:2個操作數據為數值型
2.連接符號:只要1個為字串就可以連接為字串

mysqL中的+號功用如下
select 100+90; 2個數都是數值做加法運算
select '123'+60; 其中之一是字串則將字串轉換成數值,轉換成功,則做加法運算
select 'john'+60; 轉換失敗,則將字串轉換成0
select null+10; 其中一方為null,結果肯定為null
*/

```

concat用法

```
select concat('a'+'b'+'c') as 結果;
select concat('a','b','c') as 結果;
select concat('last_name','first_name') as 姓名 from employee;
select concat('first_name','last_name')
```

ifnull 用法

```
select ifnull(commission_pct,0) as price,
commission_pct from employee;
```

concat用法+ifnull 用法

```
select
concat('last_name',',','first_name',',''job_id',',',ifnull(commission_pct,0)as 結果 from employee;
```

mysql條件查詢

語法

select 查詢表列

from 表名

where 篩選條件;

分類

1.按條件表達式篩選

條件運算符: < > = != <> >= <=

2.按邏輯表達式篩選

邏輯運算符: && || !

　　　　　and or not

3.模糊查詢

like

between and

in

is null/is not null

查詢工資>12000的員工訊息

```sql
select * from employee
where  salary>12000;
```

查詢部門編號不等於90號的員工名及部門編號

```sql
select last_name,deaprtment_id
from employee
where department_id<>90;
#---------------------------------
select last_name,deaprtment_id
from employee
where department_id!=90;
```

查詢工資在10000~20000的員工名及工資及獎金

```sql
select last_name,salary,commion_pct
from employee
where salary>=10000 and salary<=20000;
```

查詢部門編號不在90~110之間,或工資高於15000的員工訊息

```sql
select * from employee
wherer deaprtment_id<90 or department_id>110 or salary>15000;

select * from employee
wherer not(deaprtment_id>=90 or department_id<=110) or salary>15000;
```

```sql
#模糊查詢
/*
like   % 任意多個字包含0個字符
       _任意單個字符
between an
in
is null / is not null
*/
```

查詢員工名中包含字符a的員工訊息

```
select * from employee
wherer last_name like '%a%';
```

查詢員工名中第3個字符為e,第5個字符為a的員工名及工資

```
select last_name,salary from employee
where last_name like'__e_a%';
```

查詢員工名中第2個字符為_的員工名

```
select last_name from employee
where last_name like '_\_%';

select last_name from employee
where last_name like '_$_%' escape '$';

```

between and 用法

使用它可以提高語句的簡潔度,包含2邊的臨界值,2個臨界值不能調換位置

查詢員工編號在100~120之間的員工訊息

```
select * from employee
where employee_id >=100 and employee_id<=120;
#-----------------------------------
select * from employee
where employee_id between 100 and 120 ;
```

in用法

判斷某字段的值是否屬於in列表中的某一項

可提高語句簡潔度

in列表中的值類型必須一致或兼容

查詢員工編號 是 'IT_PROG' ,'AD_VP','FI_ACCOUNT'中的員工名跟編號

```
select last_name,job_id from employees
where  job_id='IT_PROG'  or job_id='AD_VP' or job_id='FI_ACCOUNT';
#-----------------------------------------------------------------
select last_name,job_id from employees
where  job_id in('IT_PROG','AD_VP','FI_ACCOUNT');

```

is null 用法

=不能判斷null

is null或 not is null 可以判斷 null值

查詢沒有獎金的員工名和獎金率

```
#錯誤示範
select last_name, commission_pct from employees
where commission_pct= null;
#正確示範
select last_name, commission_pct from employees
where commission_pct IS null;
#查詢有獎金的員工名和獎金率
select last_name, commission_pct from employees
where commission_pct is not null;
#錯誤示範
select last_name,commission_pct from employee
where salary is 12000;
```

安全等於<=>用法

```
#查詢沒有獎金的員工名和獎金率
select last_name, commission_pct from employees
where commission_pct <=> null;
```

```
#查詢工資=12000的員工訊息
select last_name, salary from employees
where salary <=> 12000;
```

is null pk <=>

is null :僅可以判斷null值,可讀性較高

<=>:既可以判斷null,又可以判斷普通數值,可讀性較低

```
#查詢員工姓名,部門編號及年薪
select last_name,
department_id,
salary*12*(1+ifnull(commission_pct,0) )as 年薪
from employees;
#查詢沒有獎金且工資小於18000的薪水及姓名
select salary,last_name
from employees
where commission_pct is null
and salary<18000;
#查詢employees表中,job_id不為 IT或工資為12000的員工訊息
select  * from employees
where job_id !='IT' or salary =12000;
#查詢employees表中,job_id不為 IT或工資為12000的員工訊息
select  * from employees
where job_id <>'IT' or salary =12000;
#查詢department的結構及全部數據
desc department;
#查詢部門表中涉及那些位置編號
select distinct location_id from departments;
#select * from employee;  select * from employees where commission_pct like
'%%' and last_name like'%%';是否依樣說明原因
不一樣  判斷內容有null

```

進階3:排序查詢

引入

```
select * from employees;
/*
select  查詢列表
from 表
where 篩選條件
order by 排序列表  desc/asc
asc 升序  desc降序
*/

select * from employees order by salary desc;
select * from employees order by salary asc;
```

查詢部門編號>=90的員工訊息,按入職時間先後排序

```
select * from employees
where  department_id>=90
order by hiredate asc;
```

按年薪的高低顯示員工的訊息和年薪排序(按別名排序)

```
select *,salary*12*(1+ifnull(commission_pct,0))年薪
from employees
order by salary*12*(1+ifnull(commission_pct,0)) desc;
```

按姓名的長度顯示員工的姓名跟工資(按函數排序)

```
select length(last_name) 字結長度,last_name,salary
from employees
order by length(last_name) desc;
```

查員工訊息,先按工資排升序,在按員工編號排降序(多字段排序)

```
select * from employees
order by salary ASC,employee_id desc;
```

查員工姓名,部門編號,年薪,先按姓名排升序,在按年薪降序(多字段排序)

```
select last_name,department_id,salary*12*(1+ifnull(commission_pct,0)) 年薪
from employees
order by 年薪 desc,last_name asc;
```

查詢工資不在8000-17000的員工的姓名跟工資,按工資降序

```
select last_name,salary
from employees
where salary not between 8000 and 17000
order by salary desc;
```

查詢信箱中包含e的員工訊息並按信箱的長度數座降序,在按部門編號作升序

```sql
select * ,length(email)
from employees
where email like '%e%'
order by length(email) desc,department_id asc;
```

進階 :常見函數

概念:類似java的方法,將一組邏輯語劇封裝在方法體中,對外爆露方法

好處:1.隱藏實現細節2.提高代碼重複使用性

調用:select 函數名(實參列表) from表;

特點: a.叫什嗎(函數名)

b.做甚麼(函數功能)

分類:1.單行函數

如 concat,length,ifnull等

2.分組函數

功能:作統計使用又稱統計函數,聚合函數,組函數

#1 字符函數

```sql
#1.length 獲取參數值的字節個數
select length('john');
select length('張三豐hahaha');
show variables like '%char%';
#2.拼接字符串
select concat(last_name,' ',first_name) 姓名 from employees;
#3.upper,lower
select  upper('john');
select lower('john');
#示範:將姓變大寫,將名改小寫,然後拼接起來
select concat(upper(last_name),lower(first_name)) 姓名 from employees;
#4.substr,substring
注意:索引從1開始
select substr('老鼠愛大米',4) out_put;
注意:截取索引從1開始處至字符長度的字元
select substr('大老鼠愛大米',1,3) out_put;
#示範:將姓名首字母變大寫,其他字母變小寫,然後拼接起來
select concat(upper(substr(last_name,1,1)),'_',lower(substr(last_name,2)))
out_put
from employees;
#5.instr 返回子串第一次出現的索引,如果找不到返回0
select instr('楊不悔愛上殷六俠','殷六俠') as out_put;
select instr('楊不悔愛殷八俠上殷六俠','殷八俠') as out_put;
select instr('楊不悔愛上殷六俠','殷八俠') as out_put;
#6.trim
select  trim('  張翠山   ') as out_put;
select length(trim('  張翠山   ')) as out_put;
select  trim('a' from 'aaaaaaaa張aaaaaa翠山aaaaaaaaa') as out_put;
#7.lpad 左填充 用*去填充指定的字符串到10
select lpad('殷素素',10,'*') as out_put;
#7.1 lpad 左填充 用*去填充指定的字符串到2
select lpad('殷素素',2,'*') as out_put;
#8.rpad 右填充 用*去填充指定的字符串到10
```

```
select rpad('殷素素',10,'*') as out_put;
#9 replace替換
select replace('張無忌愛上周芷若','周芷若','趙敏') as out_put;

```

#2.數學函數

```
#round 4捨5入
select round(-1.45);
select round(1.65);
select round(1.45);
#round 4捨5入 重載
select round(-1.45,3);
#向上取整數,返回>=該參數的最小整數
select ceil(1.01);
select ceil(1.00);
select ceil(-1.01);
#向下取整數,返回<=該參數的最大整數
select floor(1.01);
select floor(-9.99);
#truncate 截斷 從小數點後面1位截斷
select truncate(1.69999,1);
#mod取餘數
/*
mod(a,b):a-a/b*b
mod(-10,-3): -10-(-10)/(-3)*-3=-1
*/
select mod(10,3);
select 10%3;
select mod(-10,3);
```

#2.日期函數

```
#now 返回當前系統的日期+時間
select now();
#curdate 返回當前系統的日期不包含時間
select curdate();
#curtime 返回當前系統時間不包含日期
select curtime();
#可以獲取指定的部分,年,月,日,小時,分鐘,秒
select year(now()) 年;
select year('1998-1-1') 年;
select year(hiredate) 年 from employees;
select month(now()) 月 ;
select monthname(now()) 月 ;

```

#str_to_date:將日期格式的字符轉換成指定格式的日期

```
select str_to_date('9-13-1999','%m-%d-%Y') as out_put;
```

#查詢入值日期為1992-4-3的員工訊息

```
select * from employees where hiredate='1992-4-3
select * from employees where hiredate=str_to_date('4-3 1992','%c-%d %Y');
```

%Y　　4位月份

%y　　2位月份

%m　　月份(01,02,03....11,12)

%c　　月份(1,2,3....11,12)

%d　　日(01,02,03.....)

%H　　小時(24小時制)

%h　　小時(12小時制)

%i　　分鐘(00,,01,......59)

%s　　秒(00,01,.......59)

#date_format 將日期轉換成字符

```
select date_format(now(),'%y年%m月%d日') as out_put;
```

#查詢有獎金的員工名和入職日期(xx月/xx日/xx年)

```
select last_name,date_format(hiredate,'%m月/%d日 %y年') 入職日期
from employees;
```

#其他函數

```
select version();
select database();
select user();
```

#流程控制函數

#1.if函數：if else 效果

```
select if(10>5,'大','小');

select last_name,commission_pct,if(commission_pct is null,'沒獎金,呵呵','沒獎金,嘻嘻') 備註
from employees;
```

#2.case函數的使用一：類似於Java中 switch case的效果

```
/*
JAVA中(變量或表達式){
    case 常量1:語句1;break;
    ......
    default:語句n;break;

}
mysql中
case 要判斷的字段或表達式
when 常量1 then 要顯示的值1或語句1;
when 常量2 then 要顯示的值2或語句2;
........
else要顯示的值n或語句n;
```

```
end
*/

/* 案例:查詢員工的工資,要求
部門號=30,顯示工資為1.1倍
部門號=40,顯示工資為1.2倍
部門號=50,顯示工資為1.3倍
其他部門,顯示工資為1.0倍

/*

select salary 原始工資,department_id,
case department_id
when 30 then salary*1.1
when 40 then salary*1.2
when 50 then salary*1.3
else salary
end as 新工資
from employees;
```

#3.case函數使用二:類似于 多重if

```
/*
java中:
if(條件1){
       語句1:
}else if(條件2){
        語句2:
}
.....
else{
     語句n;
}

mysql中:
case
when  條件1  then 要顯示的值1或語句1
when  條件2  then 要顯示的值2或語句2
................
else要顯示的值n或語句n
end
*/
```

#案例:查詢員工工資的情況

工資>20000,級別A

工資>15000,級別B

工資>10000,級別C

否則顯示級別D

```sql
select salary,
case
when salary >20000 then 'A'
when salary >15000 then 'B'
when salary >10000 then 'C'
else 'D'
end as 工資級別
from employees;
```

常見函數

字符函數:length/concat/substr /instr /trim/upper/lower/lpad/rpad/replace

數學函數: round /ceil/floor/truncate/mod

日期函數
now/curdate/curtime/year/month/year/monthname/day/hour/minnte/second/str_to_date/date_fromat

其他函數:

version datebase user

控制函數: if /case

#1 顯示系統時間(日期+時間)

```sql
select now();
```

#2 查詢員工編號跟姓名與工資以及提高百分之20後的結果(new salary)

```sql
select employee_id,last_name,salary,salary*1.2 "new salary"
from employees;
```

#3 將員工姓名按首字母排序,並寫出姓名長度(length)

```sql
select length(last_name) 長度,substr(last_name,1,1) 首字母,last_name
from employees
order by 首字母;
```

#4 查詢產生以下結果

last_name earn salary monthly but wants salary*3

Dream Salary

King earns 24000 monthly but wants 72000

```sql
select concat(last_name,' earns ',salary,' monthly but wants ',salary*3)as "Dream Salary"
from employees
where salary=24000;
```

#5 使用case_when 按下面的條件:

job   grade

AD_PRES   A

ST_MAN　B

IT_PROG　C

SA_REP　D

ST_CLERK　E　產生以下結果

last_name Job_id　Grade

king　AD_PRES A

```
select last_name, job_id as job,
case job_id
when 'AD_PRES' then 'A'
when 'ST_MAN ' then 'B'
when 'IT_PROG ' then 'C'
when 'SA_REP' then 'D'
when 'ST_CLERK' then 'E'
END  as Grade
from employees
where job_id='AD_PRES';
```

#分組函數

sum ,avg,max.min,count

特點:

1.sum ,avg 一般用於處理數值型,max min count可以處理任何類型

2.以上分組函數都忽略null

3.可以和distinct搭配去重的運算

4.count(*)用作統計行數

5.和分組函數一同查詢的字段要求是group by後的字段

#1簡單的使用distinct搭配實現去重

```
select sum(salary) from employees;
select avg(salary) from employees;
select max(salary) from employees;
select min(salary) from employees;
select count(salary) from employees;
select sum(salary)總和,avg(salary)平均,max(salary) 最高,min(salary)最低,count(salary)總數
from employees;

select sum(salary)總和,round(avg(salary),2)平均,max(salary) 最高,min(salary)最低,count(salary)總數
from employees;
```

#參數支持那些類型

#3.忽略null

```
select sum(commission_pct),avg(commission_pct) from employees;

select
sum(commission_pct),avg(commission_pct),sum(commission_pct)/35,sum(commission_pct)/107 from employees;
```

#4. 和distinct搭配

```
select sum(distinct salary),sum(salary) from employees;
select count(distinct salary),count(salary) from employees;
```

#5. count 函數詳細介紹

效率: MYISAM存儲引擎下.count(*)的效率高

INNODB存儲引擎下.count(*)與count(1)的效率差不多,比count(字段)要高一些

```
select count(salary)from employees;
select count(*)from employees;
select count(1)from employees;
```

#6.和分組函數一同查詢的字段有限制

```
select avg(salary),employee_id from employees;
```

#1.查詢員工工資的最大值,最小值,平均值,總和

```
select max(salary) mx_sal,min(salary) mi_sal,round(avg(salary),2)
ag_sal,sum(salary) sm_sal from employees;
```

#2.查詢員工表中最大入職時間與最晚入職時間相差幾天(difference)

```
max(hiredate) min(hiredate)
select datediff('2017-10-1','2017-9-29');
select datediff(now(),'1992-11-22');

select datediff(max(hiredate),min(hiredate)) difference
from employees;
```

#查詢部門編號為90的員工數

```
select count(*) 員工數量
from employees
where department_id=90;
```

#進階5:分組查詢

引入:查詢每個部門平均工資

```
select avg(salary) from employees;
```

select 分組函數,列

from 表

where 篩選條件

group by 分組列表

order by 子句

特點:

一.分組條件篩選分2類

	數據源不一樣	位置	關鍵字
分組前篩選	原始表	group by子句的前面	where
分組後篩選	分組後結果集	group by 子句的後面	having

分組函數作條件肯定放在having子句中

能分組前篩選就優先考慮用分組前篩選

二. group by 子句支持單個字段分組及多個字段分組,(多個字段之間用逗號間隔無順序要求),表達式或函數用的較少

三.也可以添加排序,此排序放在分組查詢的最後

#1.案例 查詢每個工種最高工資

```sql
select max(salary),job_id
from employees
group by job_id;
```

#2.案例 查詢每個位置上部門的個數

```sql
select count(*),location_id
from departments
group by location_id;
```

#3.添加條件篩選

案例: 查詢信箱中包含a字母的,每個部門平均工資

```sql
select avg(salary),department_id
from employees
where email like'%a%'
group by department_id;
```

案例: 查詢每個領導有獎金的手下員工中最高工資

```sql
select max(salary),manager_id
from employees
where commission_pct is not null
group by manager_id;
```

案例:查詢那個部門員工個數>2

#1.查詢每個員工部門的個數

```sql
select count(*),department_id
from employees
group by department_id;
```

根據1結果進行篩選查詢那個部門的員工個數>2

分組後的篩選

```sql
select count(*),department_id
from employees
group by department_id
having count(*)>2;
```

案例:查詢每個工種有獎金的員工最高工資>12000的工種編號和最高工資

查詢每個工種有獎金的員工的最高工資

```sql
select max(salary),job_id
from employees
group by job_id;
```

```sql
select max(salary),job_id
from employees
where commission_pct is not null
group by job_id;
```

2根據1結果繼續篩選最高工資>12000

```sql
select max(salary),job_id
from employees
where commission_pct is not null
group by job_id
having max(salary)>12000;
```

查詢領導編號>102的每個領導手下最低工資>50000是那位,以及其最低工資

1.查詢每位領導首下的最低工資

```sql
select min(salary),manager_id
from employees
group by manager_id
```

2.添加篩選條件: 編號>102

```sql
select min(salary),manager_id
from employees
where manager_id>102
group by manager_id
```

3.添加篩選條件,最低工資>5000

```sql
select min(salary),manager_id
from employees
where manager_id>102
group by manager_id
having min(salary)>5000;
```

按表達式或函數分組

案例:按員工姓名長度分組,查詢每一組員工個數,篩選員工個數>5有那些？

查詢每個長度的員工個數

```sql
select count(*),length(last_name) len_name
from employees
group by length(last_name);
```

添加篩選條件

```sql
select count(*),length(last_name) len_name
from employees
group by length(last_name)
having count(*)>5;
//給別名
select count(*) c,length(last_name) len_name
from employees
group by len_name
having c>5;
```

按多個字段分組

案例:查詢每個部門及每個工種員工的平均工資

```sql
select avg(salary),department_id,job_id
from employees
group by department_id,job_id;
```

案例:查詢每個部門每個工種員工平均工資,按其平均工資高低排序

```sql
select avg(salary),department_id,job_id
from employees
group by department_id,job_id
order by avg(salary) desc;
```

```sql
select avg(salary),department_id,job_id
from employees
where department_id is not null
group by department_id,job_id
having avg(salary)>10000
order by avg(salary) desc;
```

```sql
select avg(salary) a,department_id,job_id
from employees
where department_id is not null
group by department_id,job_id
having a>10000
order by a desc;
```

查詢各job_id的員工工資的最大值,最小值,平均值,總和,並按job_id作升序

```sql
select max(salary),min(salary),avg(salary),sum(salary),job_id
from employees
group by job_id
order by job_id;
```

查詢員工最高工資與最低工資的差距(difference)

```sql
select  max(salary)-min(salary)difference
from employees;
```

查詢各個管理主管手下員工中最低工資,其中最低工資不能低於6000,沒有管理主管員工不計入在內

```sql
select min(salary),manager_id
from employees
where manager_id is not null
group by manager_id
having min(salary)>=6000;
```

查詢所有部門的編號,員工數量跟平均工資,並按平均工資做降序

```sql
select department_id,count(*),avg(salary) a
from employees
group by department_id
order by avg(salary) desc;
```

選取具有job_id的員工人數

```sql
select count(*) 個數,job_id
from employees
group by job_id;
```

進階6:連接查詢

含義:多表查詢,當查詢的字段來自於多個表時,就會用到連接查詢

笛卡爾乘積現象 表1有m行,表2有n行,結果m*n行

發生原因:沒有有效的連接條件

分類:

按年代分類

SQL92標準:只有內連接 也支持一部分外連接(oracle ,sqlsever,mysql不支持)

SQL99標準:支持內連接+外連接(左外+右外)+交叉連接

按功能分類

內連接

等值連接

非等值連接

自連接

外連接

左外連接

右外連接

全外連接(mysql不支持)

交叉連接

Beauty ,boys表

```
select 字段1,字段2
from 表1,表2
```

```
select name ,boyname from beauty,boys
where beauty.boyfriend_id=boys.id;
```

一.SQL92標準

1.等值連接

語法:

```
select 查詢列表
from 表1 別名,表2 別名
where 表1.key=表2.key
and 篩選條件
group by 分組字段
having 分組後篩選
order by 排序字段
```

a.多表等值得多表連接,結果為多表的交集部分

b.n表連接需要n-1個連接條件

c.表連接順序沒有要求

d.一般需要為表取別名

e.可以搭配前面介紹所有子句使用,如排序,分組,篩選等

案例:查詢女神名與對對應的男神名

```
select name ,boyname
from beauty,boys
where beauty.boyfriend_id=boys.id;
```

Myemployees表

查詢員工名與對應的部門名

```
select last_name,department_name
from employees,departments
where employees.department_id=departments.department_id;
```

2.為表起別名

提高語句的簡潔度,區分多個重名的字段

注意:如果為表起別名,則查詢時不能使用原來的表名去限定

查詢員工名,工種號,工種名

```
select last_name,jobs.job_id,job_title
from employees,jobs
where employees.job_id=jobs.job_id;
```

```
#為表起別名
select e.last_name,j.job_id,j.job_title
from employees as e,jobs as j
where e.job_id=j.job_id;
```

3.2個表順序是否可以調換

查詢員工名,工種號,工種名

```
select last_name,j.job_id,job_title
from jobs as j,employees as e
where e.job_id=j.job_id;
```

4.可以加篩選

查詢有獎金的員工名,部門名

```
select e.last_name,d.department_name,e.commission_pct
from employees e,departments d
where e.department_id=d.department_id
and e.commission_pct is not null;
```

查詢城市名中第二個字母為o的對應到的部門名跟城市名

```sql
select department_name,city
from departments d,locations l
where d.location_id=l.location_id
and city like'_o%';
```

可以加分組?

查詢每個城市的部門個數

```sql
select count(*) 個數,city
from departments d,locations l
where d.location_id=l.location_id
group by city;

```

查詢有獎金的每個部門的部門名和其領導編號與該部門的最低工資

```sql
select department_name,d.manager_id,min(salary)
from departments d,employees e
where d.department_id=e.department_id
and commission_pct is not null
group by department_name,d.manager_id;
```

#6.可以加排序

案例:查詢每個工種的工種名和員工的個數,按員工的個數排降序

```sql
select job_title,count(*)
from employees e,jobs j
where e.job_id=j.job_id
group by job_title
order by count(*) desc;
```

#7.可以實現3表連接嗎？

案例: 查詢員工名及部門名與其所在的城市

```sql
select last_name,department_name,city
from employees e,departments d,locations L
where e.department_id=d.department_id
and d.location_id=l.location_id;
```

```sql
select last_name,department_name,city
from employees e,departments d,locations L
where e.department_id=d.department_id
and d.location_id=l.location_id
and city like 's%'
order by department_name desc;
```

非等值連接

語法:

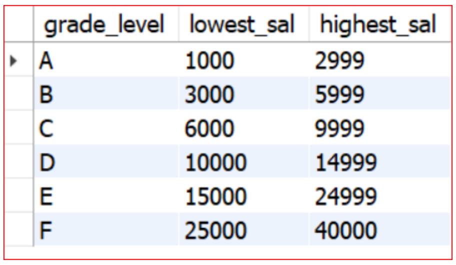
```
select 查詢列表
from 表1 別名,表2 別名
where 非等值連接條件
and 篩選條件
group by 分組字段
having 分組後篩選
order by 排序字段
```

案例:查詢員工工資與工資級別

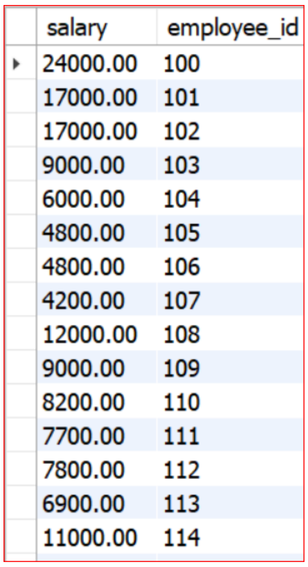
```

/*CREATE TABLE job_grades
(grade_level VARCHAR(3),
 lowest_sal  int,
 highest_sal int);

INSERT INTO job_grades
VALUES ('A', 1000, 2999);

INSERT INTO job_grades
VALUES ('B', 3000, 5999);

INSERT INTO job_grades
VALUES('C', 6000, 9999);

INSERT INTO job_grades
VALUES('D', 10000, 14999);

INSERT INTO job_grades
VALUES('E', 15000, 24999);

INSERT INTO job_grades
VALUES('F', 25000, 40000);*/
```

```
select salary,employee_id from employees;
```

```
SELECT * FROM myemployees.job_grades;
```

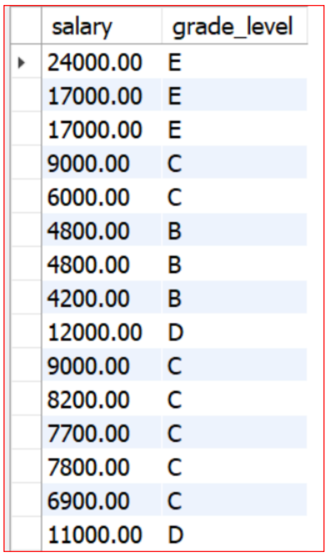


grade_level	lowest_sal	highest_sal
A	1000	2999
B	3000	5999
C	6000	9999
D	10000	14999
E	15000	24999
F	25000	40000



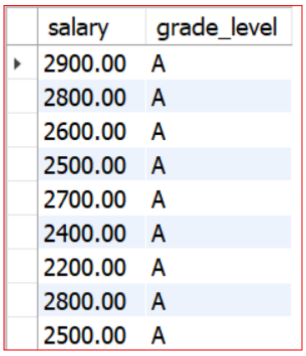
salary	employee_id
24000.00	100
17000.00	101
17000.00	102
9000.00	103
6000.00	104
4800.00	105
4800.00	106
4200.00	107
12000.00	108
9000.00	109
8200.00	110
7700.00	111
7800.00	112
6900.00	113
11000.00	114

```sql
select salary,grade_level
from employees e ,job_grades g
where salary  between g.lowest_sal and g.highest_sal;
```



salary	grade_level
24000.00	E
17000.00	E
17000.00	E
9000.00	C
6000.00	C
4800.00	B
4800.00	B
4200.00	B
12000.00	D
9000.00	C
8200.00	C
7700.00	C
7800.00	C
6900.00	C
11000.00	D

```sql
select salary,grade_level
from employees e ,job_grades g
where salary  between g.lowest_sal and g.highest_sal
and g.grade_level='A';
```



salary	grade_level
2900.00	A
2800.00	A
2600.00	A
2500.00	A
2700.00	A
2400.00	A
2200.00	A
2800.00	A
2500.00	A

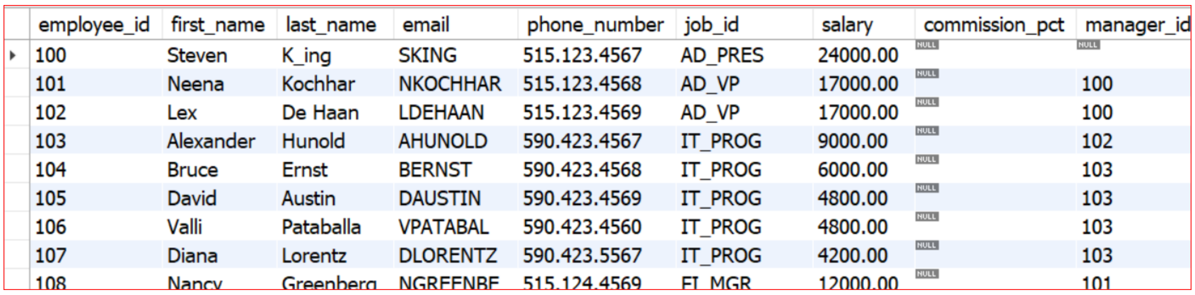
#3自連接

語法:

```
select 查詢列表
from 表 別名1,表 別名
where 表1.key=表2.key
and 篩選條件
group by 分組字段
having 分組後篩選
order by 排序字段
```

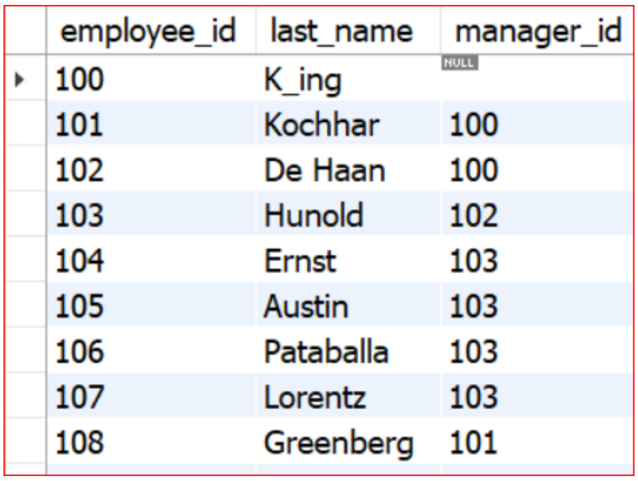
案例:查詢員工名與上級領導的名稱

```
SELECT * FROM myemployees.employees;
```



employee_id	first_name	last_name	email	phone_number	job_id	salary	commission_pct	manager_id
100	Steven	K_ing	SKING	515.123.4567	AD_PRES	24000.00	NULL	NULL
101	Neena	Kochhar	NKOCHHAR	515.123.4568	AD_VP	17000.00	NULL	100
102	Lex	De Haan	LDEHAAN	515.123.4569	AD_VP	17000.00	NULL	100
103	Alexander	Hunold	AHUNOLD	590.423.4567	IT_PROG	9000.00	NULL	102
104	Bruce	Ernst	BERNST	590.423.4568	IT_PROG	6000.00	NULL	103
105	David	Austin	DAUSTIN	590.423.4569	IT_PROG	4800.00	NULL	103
106	Valli	Pataballa	VPATABAL	590.423.4560	IT_PROG	4800.00	NULL	103
107	Diana	Lorentz	DLORENTZ	590.423.5567	IT_PROG	4200.00	NULL	103
108	Nancy	Greenberg	NGREENBE	515.124.4569	FI_MGR	12000.00	NULL	101

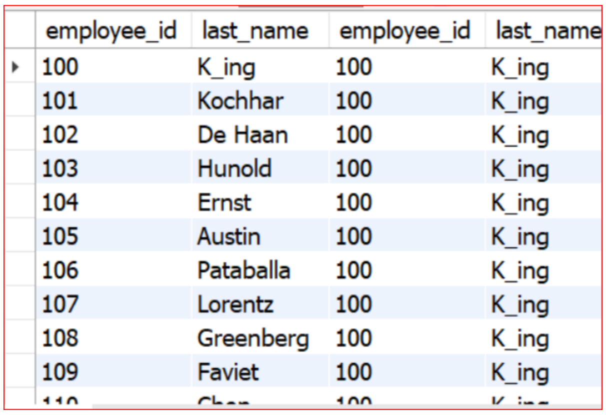
```
select employee_id,last_name,manager_id
from employees;
```



employee_id	last_name	manager_id
100	K_ing	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

案例:查詢員工名與上級領導的名稱

```
select e.employee_id,e.last_name,m.employee_id,m.last_name
from employees e,employees m;
where e.manager_id=m.employee_id;
```



	employee_id	last_name	employee_id	last_name
▸	100	K_ing	100	K_ing
	101	Kochhar	100	K_ing
	102	De Haan	100	K_ing
	103	Hunold	100	K_ing
	104	Ernst	100	K_ing
	105	Austin	100	K_ing
	106	Pataballa	100	K_ing
	107	Lorentz	100	K_ing
	108	Greenberg	100	K_ing
	109	Faviet	100	K_ing

b_id案例:顯示員工表最大工資與平均值

```sql
select max(salary),avg(salary)
from employees;
```

案例:查詢員工employee_id ,job_id,last_name按department_id做降序,按salary做升序

```sql
select employee_id,job_id,last_name
from employees
order by department_id desc, salary asc;
```

查詢員工表中job_id包含a和e的,且a在e前面

```sql
select job_id from employees where job_id like '%a%e%';
```

已知表student 有id(學號),name,gradeId(年級編號),已知表grade 有id(年級編號),name(年級名),已知表resule 有id,score,studentNo(學號),要查詢姓名,年級名,成績

```sql
select id,name,score
from student s,result t,grade g
where s.id=r.studentNo
and g.id=s.gradeId;
```

顯示當前日期,以及去前後空格,結曲子字串符函數

```sql
select now();
select trim(字符 drom '');
select substr(str,startIndex);
select substr(str,startIndex,length);
```

一.語法

select 查詢列表

from 表

where 篩選條件

order by 排序列表 (asc/desc)

二.特點

1.asc:升序(default),desc 降序

2.排序列表 支持 單字段,多字段,函數,表達式,別名

3.order by位置一般放在最後語句(除limit語句之外)

一.概述

功能:類似java中方法

好處:提高重用性與隱藏實現細節

調用:select 函數名(實參列表)

二.單行函數

1.字符函數

```
concat:連接

substr:截取字串

upper:變大寫

lower:變小寫

replace:替換

length:獲取字節長度

trim:去前後空白

lpad:左填充

rpad:右填充

instr:獲取子串第一次出現索引
```

2.數學函數

```
ceil:向上取整

round:四捨五入

mod:取模

floor:向下取整

truncate:截斷

rand:獲取隨機數,返回0~1的小數
```

3.日期函數

```
now:返回當前日期+時間

year:返回年

month:返回月份

day:返回日期

dete_format:將字期轉成字串

curdate:返回當前日期

str_to_date:返回當前日期

curtime:返回當前時間

hour:返回秒

minute:返回分鐘

second:返回秒

datediff:返回二個日期的相差天數

monthname:以英文形式返回月
```

4.其他函數

```
version: 當前數據庫服務器的版本

database:當前打開數據庫

user:當前用戶
```

password:返回當前字符密碼型式(加密)

```
select password('張小美');
select MD5('張小美');
```

md5('字符'):返回md5加密型式

5.流程控制函數

```
1.if(條件表達式1,表達式2,表達式3):如果條件表達式成立, 返回1,否則返回2

2.case情況1

case變量或表達式或字段

when 常量1  then  值1

when 常量2 then  值2

...

else  值n

end
```

```
case情況2

case

when 條件1  then  值1

when 條件2 then  值2

...

else  值n

end
```

三.分組函數

1.分類

max最大值,min最小值,sum總和,avg平均,count計算個數

2.特點

a.語法

select max(字段) from 表名:

b.支持類型

sum和avg一般用於數值型

max,min,count可以處理任何數據類型

c.以上分組函數都可以忽略null

d.以上都可以搭配distinct使用,實現去除重複的統計

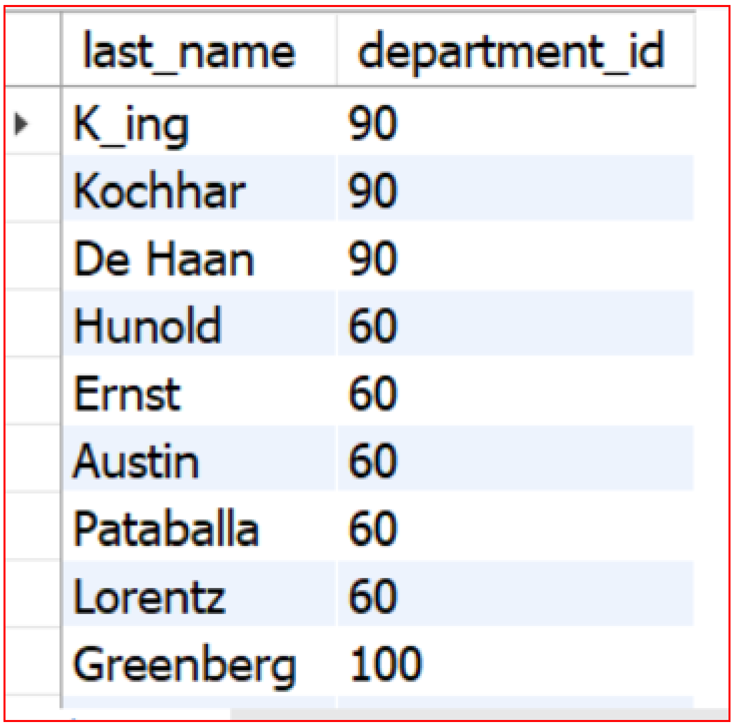
```
select sum(distinct 字段) from 表;
select count(distinct 字段) from 表;
select avg(distinct 字段) from 表;
```

e.count函數

count 字段:統計該字段非空值的個數

count(*):統計結果級的行數

案例:查詢每個部門的員工個數



last_name	department_id
K_ing	90
Kochhar	90
De Haan	90
Hunold	60
Ernst	60
Austin	60
Pataballa	60
Lorentz	60
Greenberg	100

count(1):統計結果集的行數

效率上:

MyISAM存儲引擎,count(*)最高

InnoDB存儲引擎,count(*)跟count(1)一樣>count(字段)

F.和分組函數一同查詢的字段,要求是group by後出現的字段

一語法

```
select  分組函數,分組後字段
from 表
where 篩選條件
group by 分組字段
having  分組後的篩選
order by  排序列表
```

二特點

```
             使用關鍵字          篩選的表          位置
分組前篩選     where              原始表           group by前面
分組後篩選     having             結果表           group by之後
```

案例查詢所有員工的姓名,部門編號,部門名稱

```
use myemployees;
select e.last_name,d.department_id,d.department_name
from employees e,departments d
where e.department_id=d.department_id;
```

查詢90號部門員工的job_id,location_id

```sql
select e.job_id,d.location_id
from employees e,departments d
where e.department_id=d.department_id
and e.department_id=90;
```

選擇有獎金的所有員工 last_name,department_id,location_id,city

```sql
select e.last_name,d.department_name,l.location_id,l.city
from employees e,departments d,locations l
where e.department_id=d.department_id
and d.location_id=l.location_id
and e.commission_pct is not null;
```

選擇city在toronto工作的員工的 last_name,department_id,department_name,job_id

```sql
select e.last_name,e.job_id,d.department_id,d.department_name,l.city
from employees e,departments d,locations l
where e.department_id=d.department_id
and d.location_id=d.location_id
and l.city='toronto';
```

查詢每個工種,每個部門的部門名,工種名和最低工資

```sql
select department_name,job_title,min(salary)最低工資
from employees e,departments d,jobs j
where e.department_id=d.department_id
and e.job_id=j.job_id
group by department_name,job_title;
```

要求查詢每個國家下的部門個數 > 2 的國家編號

```sql
select *
from departments d,locations l
where d.location_id=l.location_id;

select *
from departments d,locations l
where d.location_id=l.location_id
group by country_id;

select country_id,count(*) 部門個數
from departments d,locations l
where d.location_id=l.location_id
group by country_id;

select country_id,count(*) 部門個數
from departments d,locations l
where d.location_id=l.location_id
group by country_id
having count(*)>2;
```

選擇指定員工的姓名,員工號,以及他的主管的姓名與員工編號,結果類似於下格式

```
employees Emp#  manager  Mgr#
kochhar   101    king    100

select e.last_name employees,e.employee_id "Emp#", m.last_name
manager,m.employee_id "Mgr#"
from employees e,employees m
where e.manager_id=m.employee_id
and e.last_name="kochhar";
```

sql99語法

語法：

```
select  查詢列表
from 表1  別名  連接類型
join 表2  別名
on  連接條件內連接
where 篩選條件
group by 分組
having    篩選條件
order by   排序列表
```

內連接 :inner

外連接

   左外 :left outer

  右外 :right outer

  全外 :full outer

交叉連接:cross join

一.內連接

語法"

```
select  查詢列表
from 表1  別名  連接類型
join 表2  別名
on  連接條件內連接;
```

分類:

等值

非等值

自連接

特點:

a.添加排序,分組,篩選

b.inner 可以省略

c.篩選條件放在where後面,連接條件放在on後面,提高分離性,便於悅讀

d.inner join 連接和sql92語法中的等值效果是一樣的,都是查詢多表的交集

#1等值連接

案例1.查詢員工名,部門名(調換位置)

```
select last_name,department_name
from employees e
inner join departments d
on e.department_id=d.department_id;
```

查詢名字中包含e的員工名和工種名(添加 篩 選)

```
select last_name,job_title
from employees e
inner join jobs j
on e.job_id=j.job_id
where e.last_name like '%e%';
```

查詢部門個數>3的城市名和部門個數(分組+篩選)

```
select city,count(*) 部門個數
from departments d
inner join locations l
on d.location_id=l.location_id
group by city
having count(*)>3;
```

查詢那個部門的部門員工個數>3的部門名和員工個數並按個數降序(添加排序)

```
#a 查詢每個部門的員工個數
select count(*),department_name
from employees e
inner join departments d
on e.department_id=d.department_id
group by department_name;

#b在a結果上篩選員工數>3的紀錄,並排序
select count(*)個數,department_name
from employees e
inner join departments d
on e.department_id=d.department_id
group by department_name
having count(*)>3
order by count(*) desc;
```

查詢員工名,部門名,工種名,並按部門名降序

```
select last_name,department_name,job_title
from employees e
inner join departments d on e.department_id=d.department_id
inner join jobs j on  e.job_id=j.job_id
order by department_name desc;
```

#2非等值連接

1.查詢工資級別

```sql
select salary ,grade_level
from employees e
join job_grades j
on e.salary between j.lowest_sal and j.highest_sal;
```

2.查詢工資級別的個數>20並且按工資級別降序

```sql
select count(*) ,grade_level
from employees e
inner join job_grades j
on e.salary between j.lowest_sal and j.highest_sal
group by grade_level
having count(*)>20
order by grade_level desc;
```

#3.自連接

1.查詢員工名字,上級名字

```sql
select e.last_name, m.last_name
from employees e
inner join employees m
on e.manager_id=m.employee_id;
```

2.查詢員工名字中有包含k的,上級名字

```sql
select e.last_name, m.last_name
from employees e
inner join employees m
on e.manager_id=m.employee_id
where e.last_name like '%k%';
```

二.外連接

/*應用場景:用於查詢一表中有沒有另一個表所沒有的紀錄

特點:

a.外連接的查詢為主表中的所有記錄

　　如有匹配的則顯示其值

　　如無匹配的則顯示null

　　外連接查詢結果=內連接結果+主表中有而副表沒有的紀錄

b.左外連接 left join左邊是主表

　右外連接right join右邊的主表

c.左外和右外交換2各表順序,可以實現同樣效果

d.全外連接=內連接結果+表1有但表2沒有的+表2中有但表1沒有的

*/

#引入:查詢沒有男朋友的女神名

```
#左外連接
select b.name,bo.*
from beauty b
left outer join boys bo
on b.boyfriend_id=bo.id
where bo.id is null;
```

```
update boys set usercp=null where id=3;
```

```
#右外連接
select b.name,bo.*
from boys bo
right outer join beauty b
on b.boyfriend_id=bo.id
where bo.id is null;
```

```
#左外連接
select b.*,bo.*
from boys bo
left outer join beauty b
on b.boyfriend_id=bo.id
where b.id is null;
```

查詢那個部門沒有員工

```
#左外
select d.*,e.employee_id
from departments d
left outer join employees e
on d.department_id=e.department_id
where e.employee_id is null;

#右外
select d.*,e.employee_id
from employees e
right outer join departments d
on d.department_id=e.department_id
where e.employee_id is null;

```

#全外連接

```sql
use girls;
select b.*,bo.*
from beauty b
full outer join boys bo
on b.boyfriend_id=bo.id;
```

#交叉連接

```sql
select b.*,bo.*
from beauty b
cross join boys bo;

```

#sql92 pk sql99

功能:SQL 99支持的較多

可讀性:SQL99實現連接條件和篩選條件的分離,可讀性較高

#一.查詢編號>3的女神男朋友訊息,如果有則詳細列出,如果無,則用NULL填充

```sql
select b.id,b.name,bo.*
from beauty b
left outer join boys bo
on b.boyfriend_id=bo.id
where b.id>3;
```

#二.查詢那個城市沒有部門

```sql
select city,d.*
from departments d
right outer join locations l
on d.location_id=l.location_id
where d.department_id is null;
```

#三.查詢部門為SAL或IT的員工訊息

```sql
select e.*,d.department_name
from departments d
left join employees e
on d.department_id=e.department_id
where d.department_name in ('SAL','IT');


SELECT * FROM myemployees.departments d
where d.department_name in ('SAL','IT');
```

#進階7:子查詢

含意:出現在其他語句中的select語句,稱為子查詢或內查詢

外部的查詢語句稱為主查詢或外查詢

分類:

按子查詢出現的位置:

select 後面

僅僅支持標量子查詢

from 後面

支持 表子查詢

where 或having後面 **

標量子查詢 (單行)*

列子查詢 (多行) *

行子查詢

exist後面(相關子查詢)

表子查詢

按結果級的行列數不同

標量子查詢(結果集只有一行一列)

列子查詢(結果集只有一行多列)

行子查詢(結果集有行子查詢(結果集有一行多列)

表子查詢(結果集一般為多行多列)

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

一.where或having 後面

1.標量子查詢(結果集只有一行一列)

2.列子查詢(多行子查詢)

3.行子查詢(多列多行)

a.子查詢放在()內

b.子查詢一般放在條件的右側

c.標量子查詢,一般搭配單行操作使用

<, > ,<=, >=,<>

列子查詢,一般搭配多行操作符使用

in,any,some,all

4.子查詢優先於主查詢的執行,主查詢用到子查詢的結果

1.標量子查詢

#誰的工資比Abel高

```sql
#查詢abel的工資
select salary
from employees
where last_name='abel'

#查詢員工訊息,滿足salary>abel薪水的結果
select *
from employees
where salary>(
    select salary
    from employees
    where last_name='abel'

);

```

#返回job_id與141號員工相同,salary比143號員工多的員工,其姓名,job_id和工資

```sql
#查詢141號員工的job_id
select job_id
from employees
where employee_id=141
#查詢143號員工的salary
select  salary
from employees
where employee_id=143
#查詢員工姓名,job_id與工資,要求job_id與141號相同,salary>143號員工
select last_name,job_id,salary
from employees
where job_id=(

    select job_id
    from employees
    where employee_id=141

) and salary>(

    select  salary
    from employees
    where employee_id=143

);
```

#返回公司工資最少的員工last_name,job_id與salary信息

```sql
#查詢公司的最低工資
select min(salary)
from employees
#查詢last_name,job_id,符合最低工資的人
select last_name,job_id,salary
from employees
where salary=(
    select min(salary)
    from employees
);
```

#查詢最低工資>50號部門的id與其最低工資

```sql
#A查詢50號部門最低工資
select min(salary)
from employees
where department_id=50
#B查詢每個部門的最低工資
select min(salary),department_id
from employees
group by department_id;
#C查詢其他部門的最低工資,要滿足B條件其部門id與最低工資
select min(salary),department_id
from employees
group by department_id
having min(salary)>(
    select min(salary)
    from employees
    where department_id=50

);
```

#列子查詢(多行子查詢)

IN/NOT IN :等於列表中的任何一個

ANY/SOME:和子查詢返回的某一個值比較

ALL:和子查詢返回的所有值比較

```sql
a in (10,20,30);
a not in(10,20,30);
a > any(10,20,30);=a>min(10);
a >all(10,20,30);=a>max(30);
```

#返回location_id是1400或1700的部門中所有員工姓名

```sql
#A查詢location_id是1400或1700的部門編號
select distinct department_id
from departments
where location_id in(1400,1700);
#查詢員工姓名,要求部門編號是A中列表的某一個
select last_name
from employees
where department_id in(
    select distinct department_id
    from departments
    where location_id in(1400,1700)
);
#或
select last_name
from employees
where department_id = any (
    select distinct department_id
    from departments
    where location_id in(1400,1700)
);

#查詢員工姓名,要求部門編號是A中列表的某一個 (相反)
select last_name
from employees
where department_id not in(
    select distinct department_id
    from departments
    where location_id in(1400,1700)
);
#或
select last_name
from employees
where department_id <>all(
    select distinct department_id
    from departments
    where location_id in(1400,1700)
);
```

#返回其他工種中比job_id為IT_PROG工種任一工資低的員工的員工編號,姓名,job_id以及salary

```
#A查詢job_id為IT_PROG部門任一工資
select distinct job_id,salary
from employees
where job_id ='IT_PROG';
#查詢,姓名,job_id以及salary,salary<any (A)的任意一個
select last_name,employee_id,job_id,salary
from employees
where salary<any(
    select distinct salary
    from employees
    where job_id ='IT_PROG'

)and job_id<> 'IT_PROG';
#或
select last_name,employee_id,job_id,salary
from employees
where salary<(
    select MAX( salary)
    from employees
    where job_id ='IT_PROG'

)and job_id<> 'IT_PROG';
```

#返回其他工種中比job_id為IT_PROG工種所有工資低的員工的員工編號,姓名,job_id以及salary

```sql
select last_name,employee_id,job_id,salary
from employees
where salary<all(
    select distinct salary
    from employees
    where job_id ='IT_PROG'

)and job_id<> 'IT_PROG';
#或
select last_name,employee_id,job_id,salary
from employees
where salary<(
    select min(salary)
    from employees
    where job_id ='IT_PROG'

)and job_id<> 'IT_PROG';
```

#3行子查詢(結果集一行多列或多行多列)

查詢員工編號最小且工資最高的員工資料

```sql
#查詢最小員工編號
select min(employee_id)
from employees;
#查詢最高工資
select max(salary)
from employees;
#查詢員工編號最小且工資最高的員工資料
select *
from employees
where employee_id=(
    #查詢最小員工編號
) and salary=(
    #查詢最高工資
);
#------------------------------
select *
from employees
where employee_id=(
   select min(employee_id)
   from employees
) and salary=(
   select max(salary)
   from employees
);

#新方法
select *
from employees
where(employee_id,salary)=(
    select min(employee_id),max(salary)
    from employees
);


```

select後面子查詢的使用

僅支持標量子查詢

#案例:查詢每個部門的員工個數

```sql
select d.*,(
    select count(*)
    from employees
)
from departments d;

//--------------------------
select d.*,(
    select count(*)
    from employees e
    where e.department_id=d.department_id
 ) 個數
from departments d;
```

#案例:查詢員工號=102的部門名

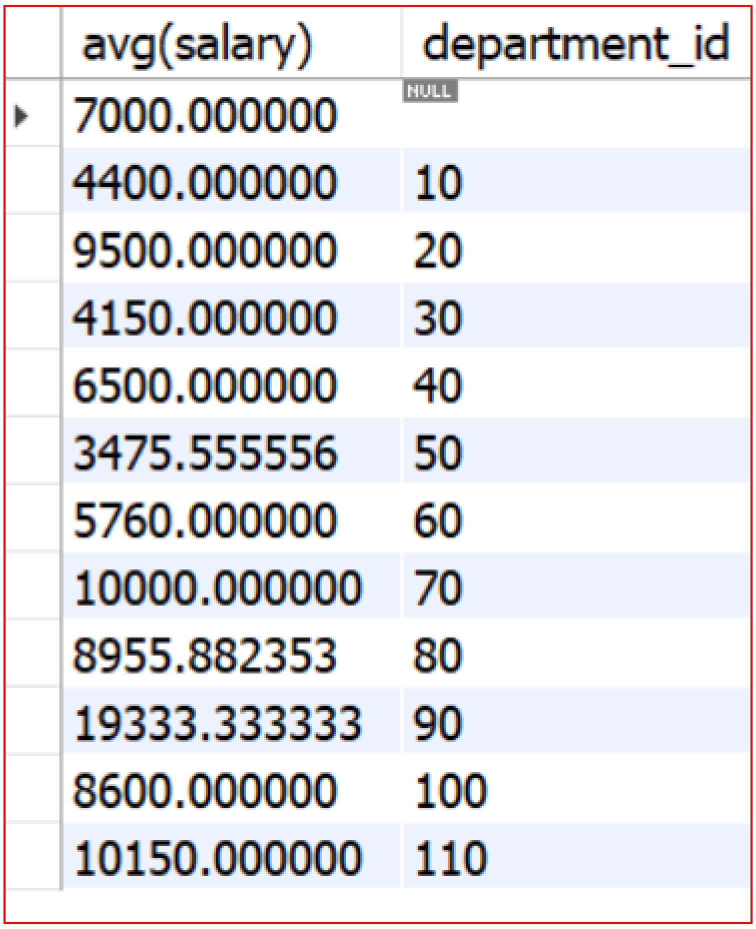
```sql
select (
    select department_name
    from departments d
    inner join employees e
    on d.department_id=e.department_id
    where e.employee_id=102
) 部門名;
```

三.from 後面

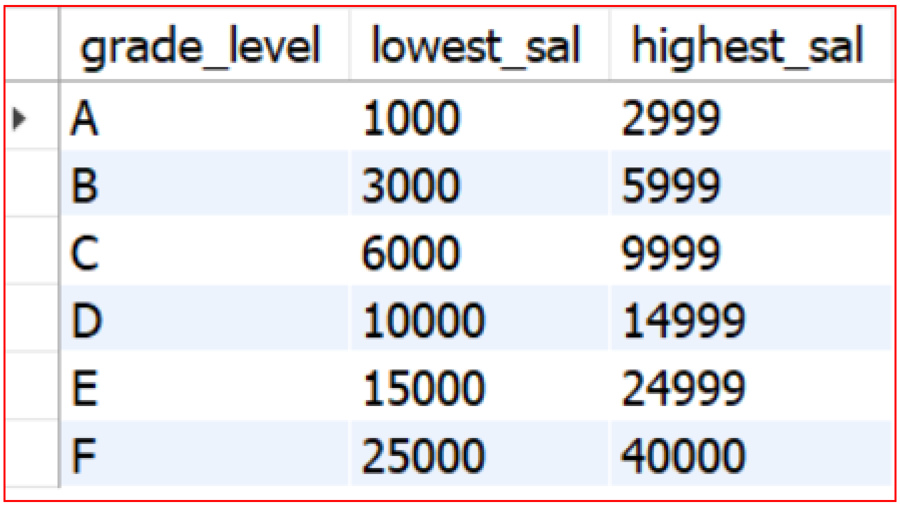
將"子查詢"結果當成一張表,要求必須起別名

案例:查詢每個部門平均的工資等級

```sql
查詢每個部門平均工資
select avg(salary),department_id
from employees
group by department_id
```



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

```sql
#2連接1的結果和job_grade表,篩選條件將平均工資用between lowest_sal and hightest_sal
select ag_dep.* ,g.grade_level
from (
    select avg(salary) ag ,department_id
    from employees
    group by department_id
) ag_dep
inner join job_grades g
on ag_dep.ag between lowest_sal and highest_sal;
```

4.exists後面(相關子查詢)

```
/*語法
exists(完整查詢語句)
結果為boolean 1 or 0
*/

select exists (select employee_id  from employees);
select exists (select employee_id  from employees where salary=30000);
```

查詢有員工名的部門名

```
select department_name
from departments
where exists(

);
//--------------------------------------
#exists作法
select department_name
from departments d
where exists(
    select *
    from employees e
    where d.department_id=e.department_id

);
//------------------------------------
#in作法
select department_name
from departments d
where d.department_id in(
    select department_id
    from employees
);
```

查詢沒有女朋友的男神訊息

```
#in 作法
select bo.*
from boys bo
where bo.id not in(
   select boyfriend_id
   from beauty
);
#exists作法
select bo.*
from boys bo
where not exists(
   select boyfriend_id
   from beauty b
   where bo.id=b.boyfriend_id
);
```

查詢與zlotkey相同部門員工姓名與工資

```sql
#step 1
select department_id
from employees
where last_name ='Zlotkey';
#2查詢部門編號=相同部門員工姓名與工資
select last_name,salary
from employees
where department_id =(
    select department_id
    from employees
    where last_name ='Zlotkey'
);
```

查詢工資比公司平均工資高的員工及員工編號,姓名與工資

```sql
#step1查詢平均工資
select avg(salary)
from employees;
#查詢工資>員工編號,姓名與工資
select employee_id,last_name,salary
from employees
where salary>(
    select avg(salary)
    from employees
);
```

查詢各部門中工資比該部門平均工資高的員工編號,姓名與工資

```sql
#step1 查詢個部門平均工資
select avg(salary),department_id
from employees
group by department_id;
#2連接1結果集與employess表進行篩選
select employee_id,last_name,salary,e.department_id
from employees e
inner join (
 select avg(salary) ag ,department_id
 from employees
 group by department_id
) ag_dep
on e.department_id=ag_dep.department_id
where salary>ag_dep.ag;

```

查詢和員工姓名中包含u的員工在相同部門的員工編號與姓名

```sql
#1.step1查詢姓名中包含u的部門
select distinct department_id
from employees
where last_name like '%u%';
#2查詢部門編號=step1中的任何一個員工編號跟姓名
select last_name,employee_id
from employees
where department_id in(
    select distinct department_id
    from employees
    where last_name like '%u%'
);
```

查詢在部門的location_id=1700同部門工作的員工編號

```sql
#1.step1查詢location_id=1700有那些?
select distinct department_id
from departments
where location_id=1700;
#查詢部門號為location_id=1700任意一個員工編號
select employee_id
from employees
where department_id=any(
    select distinct department_id
    from departments
    where location_id=1700
);

```

查詢直屬主管是k_ing的員工姓名與工資

```sql
#1.查詢姓名為king的員工編號
select employee_id
from employees
where last_name='k_ing';
#2查詢manager_id=(step1)中的員工姓名及工資
select last_name,salary
from employees
where manager_id in(
    select employee_id
    from employees
    where last_name='k_ing'
);


```

查詢工資最高的員工的姓名,並將first_name與last_name顯示唯一列,其列名為姓與名

```sql
#step 1 查詢最高工資
select max(salary)
from employees;
#查詢工資=step1姓與名
select concat(first_name,last_name)"姓.名"
from employees
where salary in(
    select max(salary)
    from employees
);
```

分頁查詢

```sql
#應用場景:當要顯示的數據,一頁顯示不全,需要分頁提交SQL請求
#語法
   select 查詢列表
   from 表
   (join type) join 表2
   on 連接條件
   where 篩選條件
   group by 分組字段
   having 分組後的篩選
   order by 排序的字段
   limit (offset,)size;
   offset要顯示條目的起始索引(起始索引從0開始)
   size 要顯示的條目個數
 #特點:limit語句放在查詢語句的最後
 #公式:要顯示頁數,每頁的條目數size
/* select 查詢列表
 from 表
 limit(page-1)*size,size;

 size=10
 page       index
 1          0
 2          10
 3          20
 */
```

#查詢前5條員工訊息

```sql
select * from employees limit 0,5;
select * from employees limit 5;
```

#查詢第11條到第25條

```sql
select * from employees limit 10,15;
```

#查詢有獎金的員工訊息,並且工資較高的前10名顯示出來

```sql
select * from employees
where commission_pct is not null
order by salary desc
limit 10;
```

測試題詳解

已知表 stuinfo

id學號

name姓名

email 郵箱 john@126.com

gradeId 年級編號

sex 性別 男女

age 年齡

已知表 grade

id 年級編號

gradeName 年級名稱

//-------------------------------

#查詢 所有學員的郵箱用戶名(注:郵箱中@前面的字符)

```sql
select substr(email,instr(email,'@')-1) 用戶名
from stuinfo;
```

#查詢男生女生的個數

```sql
select count(*) 個數,sex
from stuinfo
group by sex;
```

查詢年齡>18歲所有學生的姓名和年級名稱

```sql
select name,gradeName
from stuinfo s
inner join grade g  on s.gradeId=g.id
where age>18;
```

查詢那個年級的學生最小年齡>20歲

```sql
#1先查每個年級最小年齡
select min(age),gradeid
from stuinfo
group by gradeid;

#2在1的結果上篩選
select min(age),gradeid
from stuinfo
group by gradeid
having min(age)>20;
```

試說出查詢語句中涉及所有關鍵字以及其先後執行順序

```
#語法
   select 查詢列表
   from 表1 別名
   (join type) join 表2
   on 連接條件
   where 篩選條件
   group by 分組字段
   having 分組後的篩選
   order by 排序的字段
   limit (offset,)size;
   offset要顯示條目的起始索引(起始索引從0開始)
   size 要顯示的條目個數
```

sql99語法

1.內連接

```
#語法
select 查詢列表
from 表1 別名
inner join 表2 別名 on 連接條件
where 篩選條件
group by 分組列表
having 分組後的篩選
order by 排序列表
limit 子句;
/*特點:
  1.表的 順序可以調換
  2.內連接的結果=多表的交集
  3.n表連接,需n-1個連接條件
  分類:
  等值連接
  非等值連接
  自連接
*/

```

2.外連接

```
#語法
select 查詢列表
from 表1 別名
left/right/full outer join 表2 別名 on 連接條件
where 篩選條件
group by 分組列表
having 分組後的篩選
order by 排序列表
limit 子句;
/*特點:
  1.查詢結果=主表中所有的行,其中副表與其匹配的將顯示匹配行,如果沒有匹配的則顯示null
  2.left join 左邊救世主表,right join 右邊就是主表
    full join 二邊都可以是主表
  3.一般用於查詢除了交集部分的剩餘不匹配的行

*/
```

3.交叉連接

```
#語法
select 查詢列表
from 表1 別名
cross join 表2 別名;

/*特點:
  1.類似於笛卡爾乘績


*/
```

子查詢

1.含義:

嵌套在其他語句內部的select語句稱為子查詢或內查詢

外面語句可以是insert ,update,delet,select等,一般select作為外面的語句較多

外面如果為select語句,則此語句稱為主查詢或外查詢

2.分類:

1.按出現位置

select 後面:

    僅支持標量子查詢

from 後面:

    表子查詢

where 或having 後面

    標量子查詢,

    列子查詢

    行子查詢

exist後面

    標量子查詢(單行字查詢):結果集為一行一列

    列子查詢(多行子查詢):結果集為多行一列

    行子查詢:結果集為多行多列

    表子查詢:結果集為多行多列

2.按結果集的行列

a.標量子查詢(單行字查詢):結果集為一行一列

b.列子查詢(多行子查詢):結果集為多行一列

c.行子查詢:結果集為多行多列

d.表子查詢:結果集為多行多列

3.示例:

where 或having 後面

a.標量子查詢(單行字查詢):結果集為一行一列

```
#查詢最低工資的員工姓名跟工資
#1.最低工資
select  min(salary) from employees
#2.查詢員工的姓名跟工資,要求工資=條件1
select last_name,salary
from employees
where salary=(
    select  min(salary) from employees
);
```

b.列子查詢(多行子查詢):結果集為多行一列

```
#查詢所有是領導的員工姓名
#1.查詢所有員工的Manager_id
select manager_id
from employees;
#2.查詢姓名,employee_id屬於1列表的其中之一
select last_name
from employees
where employee_id in(
   select manager_id
   from employees
);

```

c.行子查詢:結果集為多行多列

d.表子查詢:結果集為多行多列

分頁查詢

1.應用場景

當要查詢的條目太多,一頁顯示不全

```
/*語法:

select 查詢列表
from
limit( offset,)size;
offset要顯示條目的起始索引(起始索引從0開始)
   size 要顯示的條目個數
 公式:
 假如要顯示的頁數為page,每一個條目數為size
 select 查詢列表
 from 表
 limit(page-1)*size,size;

```

```
size=10
page      index
1         0
2         10
3         20
*/

```

案例

1. 查詢工資最低的員工資訊: last_name, salary

```
#1查詢工資最低員工
select min(salary)
from employees;
#查詢 last_name, salary 符合條件1的員工
select last_name,salary
from employees
where salary=(
   select min(salary)
   from employees
);

```

2. 查詢平均工資最低的部門資訊

```
#方法一
#1查詢各部門平均工資
select avg(salary),department_id
from employees e
group by department_id;
#2查詢1結果集最低平均工資
select min(ag)
from (
   select avg(salary)ag,department_id
   from employees e
   group by department_id
) ag_dep;
#3查詢那個部門平均工資=2
select avg(salary),department_id
from employees e
group by department_id
having avg(salary)=(
      select min(ag)
      from (
       select avg(salary)ag,department_id
       from employees e
       group by department_id
) ag_dep
);
#4查詢部門訊息
select d.*
from departments d
where d.department_id=(

    select department_id
    from employees e
    group by department_id
    having avg(salary)=(
      select min(ag)
      from (
       select avg(salary)ag,department_id
       from employees e
       group by department_id
) ag_dep
)
);
```

```
#方法二
#1查詢各部門平均工資
select avg(salary),department_id
from employees e
group by department_id;
#2查詢1結果集最低平均工資的部門編號
select department_id
from employees
group by department_id
order by avg(salary)
limit 1;
#3查詢部門訊息
select *
from departments
where department_id=(
   select department_id
   from employees
   group by department_id
   order by avg(salary)
   limit 1
);
```

3. 查詢平均工資最低的部門資訊和該部門的平均工資

```
#1查詢各部門平均工資
select avg(salary),department_id
from employees e
group by department_id;
#2查詢1結果集最低平均工資的部門編號
select avg(salarydepartment_id
from employees
group by department_id
order by avg(salary)
limit 1;
#3查詢部門訊息
select d.*,ag
from departments d
join (
    select avg(salary)ag,department_id
    from employees
    group by department_id
    order by avg(salary)
```

```sql
    limit 1
) ag_dep
on d.department_id=ag_dep.department_id;

```

4. 查詢平均工資最高的 job 資訊

```sql
#1查詢最高job的平均工資
select avg(salary),job_id
from employees
group by job_id
order by avg(salary) desc
limit 1
#2查詢job信息
select *
from jobs
where job_id=(
    select job_id
    from employees
    group by job_id
    order by avg(salary) desc
    limit 1
);
```

5. 查詢平均工資高於公司平均工資的部門有哪些?

```sql
#1查詢公司的平均工資
select avg(salary)
from employees;
#2查詢各部門平均工資
select avg(salary),department_id
from employees e
group by department_id;
#3篩選2結果集,滿足平均工資>1
select avg(salary),department_id
from employees
group by department_id
having avg(salary)>(
    select avg(salary)
    from employees

);
```

6. 查詢出公司中所有 manager 的詳細資訊.

```
#1查詢所有manager員工編號
select distinct manager_id
from employees;
#2查詢詳細訊息,滿足employee_id=1
select *
from employees
where employee_id=any(
    select distinct manager_id
    from employees
);
```

7. 各個部門中 最高工資中最低的那個部門的 最低工資是多少

```
#1查詢各部門的最高工資中最低的部門
select department_id
from employees
group by department_id
order by max(salary) asc
limit 1;
#2查詢1條件部門的最低工資
select min(salary),department_id
from employees
where department_id=(
    select department_id
    from employees
    group by department_id
    order by max(salary) asc
    limit 1
);
```

8. 查詢平均工資最高的部門的 manager 的詳細資訊: last_name, department_id, email, salary

```
#1查詢公司的平均工資最高的其部門編號
select department_id
from employees
group by department_id
order by avg(salary) desc
limit 1;
#2連接employees與department_id二張表,篩選manager要符合條件1的主管訊息,last_name, department_id, email, salary
select last_name,d.department_id, email, salary
from employees e
inner join departments d on d.manager_id=e.employee_id
where d.department_id=(
    select department_id
    from employees
    group by department_id
    order by avg(salary) desc
    limit 1
);

```

查詢每個專業的學生人數

```
select majorid,count(*)
from student
group by majorid;
```

查詢參加考試的學生中,每個學生的平均分,最高分

```
select avg(score),max(score),studentno
from result
group by studentno;
```

查詢姓張的每個學生的最低分大於60的學號,姓名

```
select r.score, s.studentno,s.studentname
from result r
inner join student s
where score>60 and r.studentno=s.studentno and studentname like "張%";
//--------------------------------
select s.studentno,studentname,min(score)
from student s
inner join result r
on s.studentno=r.studentno
where s.studentname like "張%"
group by s.studentno
having min(score)>60;

```

查詢生日在"1998-1-1"後的學生姓名,專業名稱

```
select studentname,majorname
from student s
join major m
on s.majorid=m.majorid
where datediff(borndate,'1988-1-1')>0;
```

查詢每個專業的男生人數跟女生人數分別是多少

```
#方式1
select count(*) 個數,sex,majorid
from student
group by sex,majorid;
#方式2
select majorid,
(select count(*) from student where sex='男' and majorid=s.majorid) 男,
(select count(*) from student where sex='女' and majorid=s.majorid) 女
from student s
group by majorid;
```

查詢專業與張翠山一樣的最低分學生

```
#1 查詢張翠山的專業
select majorid
from student
where studentname ='張翠山';
#2 查詢編號=1的所有學生
```

```sql
select studentno,majorid
from student
where majorid=(
    select majorid
    from student
    where studentname ='張翠山'
);
#3 查詢專業與張翠山一樣的最低分學生
select min(score),studentno
from result
where studentno in(
select studentno
from student
where majorid=(
    select majorid
    from student
    where studentname ='張翠山'
)
);

```

查詢>60分的學生姓名,密碼,專業名

```sql
select studentname,loginpwd,majorname
from student s
join major m on s.majorid=m.majorid
join result r on s.studentno=r.studentno
where r.score>60
```

按郵箱位數分組,查詢每組的學生個數

```sql
select count(*),length(email)
from student
group by length(email);
```

查詢學生名,專業名,分數

```sql
select studentname,score,majorname
from student s
join major m on s.majorid=m.majorid
join result r on s.studentno=r.studentno
```

查詢那個專業沒有學生,分別用左連接和右連接實現

```sql
#左
select m.majorid,m.majorname,s.studentno
from major m
left join student s on m.majorid=s.majorid
where s.studentno is null;
#右
select m.majorid,m.majorname,s.studentno
from student s
right join major m on m.majorid=s.majorid
where s.studentno is null;
```

查詢沒有成績的學生人數

```
#1 step
select s.*,r.id
from student s
left join result r on s.studentno=r.studentno
where r.id is null;
#2 step
select count(*)
from student s
left join result r on s.studentno=r.studentno
where r.id is null;
```

聯合查詢

```
/*
union 聯合 合併:將多條查詢語句的結果合併成一個結果
適用於查詢多表的時候,查詢列基本是一致
語法
查詢語句1
union (all)
查詢語句2
union
...


應用場景:
要查詢的結果來自於多個表,且多個錶沒有直接的連接關係,但查詢信息一致時
特點:
1.要求多條查詢語句的查詢列數是一致的
2.多條查詢語句中每一列的類型與順序要一致
3.union去重,union all包含重複項

*/
```

```
#引入案例:查詢部門編號>90或郵箱包含a的員工信息
#方法1
select * from employees where email like '%a%' or department_id>90;
#方法2
select * from employees where email like '%a%'
union
select * from employees where department_id>90;
```

```
#1查詢中國用戶中男性的信息以及外國用戶中男性的用戶信息
select  id,cname,csex from t_ca where csex='男'
union
select t_id,tname,tGender from t_ua where tgender='male';
#查詢中國用戶中男性的信息以及外國用戶中男性的用戶信息,不想去重複需加all
select  id,cname,csex from t_ca where csex='男'
union all
select t_id,tname,tGender from t_ua where tgender='male';
```

DML語言

```
/*
數據操作語言
插入:insert
修改:update
刪除:delete

*/
#1插入語句
/*
語法:
insert into 表名(列名,...) values(值1,...);
表名
列名
新值

特點 :
1.要求值的類型與字段的類型要一致或兼容
2.字段的個數與順序可以和原始表中字段或個數不一致,但必須保證值與字段一一對應
3.假如表中有可以為null的字段,可以用以下2種方式插入null值
    a.字段與值都省略
    b.字段寫上,值使用null
4.字段與值個數需一致
5.字段名可以省略,默認所有列
*/


#A.插入的值類型要與列的類型一致或兼容
insert into beauty(id,name,sex,borndate,phone,photo,boyfriend_id)
values(13,'唐藝析','女','1990-4-23','1898888888',null,2);
#B.不可以為null的列必須插入值,可以為null的列如何插入值?
#方式1
insert into beauty(id,name,sex,borndate,phone,photo,boyfriend_id)
values(13,'唐藝析','女','1990-4-23','1898888888',null,2);
#方式2
insert into beauty(id,name,sex,borndate,phone,boyfriend_id)
values(13,'唐藝析','女','1990-4-23','1898888888',2);
#
insert into beauty(id,name,sex,borndate,phone,boyfriend_id)
values(14,'金星','女','1990-4-23','1388888888',9);
#
insert into beauty(id,name,sex,phone)
values(15,'娜扎','女','1388888888');
#列的順序可否調換
insert into beauty(name,sex,id,phone)
values('蔣欣','女',16,'110');
#列的個數與值的個數需一致
insert into beauty(name,sex,id,phone)
values('關曉彤','女',17,'110');
#可以省略列名,默認所有列,而且列的順序跟表中列的順序一致
insert into beauty
values(18,'張飛','男',NULL,'119',NULL,NULL);

#方式2
/*
語法:
```

```
insert into 表名
set 列名=值,列名=值,....
*/

insert into beauty
set id=19,name='劉濤',phone='999';

#2種方式大pk
#方式1支持插入多行,方式2不支持
insert into beauty
values(23,'唐藝析1','女','1990-4-23','1898888888',null,2)
,(24,'唐藝析2','女','1990-4-23','1898888888',null,2)
,(25,'唐藝析3','女','1990-4-23','1898888888',null,2);
#方式1支持子查詢,方式2不支持
insert into beauty(id,name,phone)
select 26,'宋茜','11809866';

insert into beauty(id,name,phone)
select id,boyname,'1234567'
from boys where id<3;
```

修改語句

```
/*
1.修改單表的紀錄
update 表名
set 列=新值,列=新值,....
where 篩選條件;

2.修改多表的紀錄
sql92語法
update 表1 別名,表2 別名
set 列=值,....
where 連接條件
and 篩選條件;

sql99語法
update 表1 別名
inner/left/right join 表2 別名
on 連接條件
set 列=值,....
where 篩選條件
*/
```

```
#1.修改單表的紀錄
#案例1.修改姓唐的女生電話為13899888899
SET SQL_SAFE_UPDATES=0;
update beauty set phone='1389999999'
where name like '唐%';
#案例2.修改boys表中id=2中名稱為張飛,魅力值 10
update boys set boyname='張飛',usercp=10
where id=2;
```

```
#2修改多表的記錄
#case1: 修改張無忌的女朋友的手機號碼為114
update boys bo
```

```sql
inner join beauty b on bo.id=b.boyfriend_id
set b.phone='114'
where bo.boyname='張無忌';

#case2:修改沒有男朋友的女神,將其男朋友都改為2號
update boys bo
right join beauty b on bo.id=b.boyfriend_id
set b.boyfriend_id=2
where bo.id is null;
select * from boys;

```

刪除語句

```sql
/*
方式1:delete
語法:
1.單表的刪除
delete from 表名 where 篩選條件

2.多表的刪除
sql92語法:
delete 表1別名,表2別名
from 表1 別名,表2 別名
where 連接條件
and 篩選條件;

sql99語法:
delete 表1別名,表2別名
from 表1 別名
inner/left/right join 表2別名 on連接條件
where 篩選條件;


方式2:Truncate
語法:
truncate 表名;

*/
```

```sql
#刪除手機號碼後碼為9的女神信息
SET SQL_SAFE_UPDATES=0;
delete from beauty where phone like '%9';

```

```sql
#多表刪除
#case1:刪除張無忌的女朋友信息
delete b
from beauty b
inner join boys bo on b.boyfriend_id=bo.id
where bo.boyname='張無忌';
```

```
#case2:刪除黃曉明以及他女朋友的信息
delete b,bo
from beauty b
inner join boys bo on b.boyfriend_id=bo.id
where bo.boyname='黃曉明';
```

```
#方式1 :truncate 語句
#case1
truncate table boys;
```

```
#delete pk truncate
/*
1.delete 可以加where,truncate 不能加
2.truncate刪除,效率高一些
3.假如要刪除表中AI
  用delete刪除後,再插入數據,AI值從斷點開始
  用truncate刪除後,再插入數據,AI值從1開始
4.truncate 刪除沒有返回值,delete有返回值
5.truncate刪除不可以回滾,delete刪除可以回滾
*/
```

```
delete  from boys
insert into boys(boyname,usercp)
values('張飛',100),('劉備',100),('觀雲長',100);

//-----------------------
truncate  from boys
insert into boys(boyname,usercp)
values('張飛',100),('劉備',100),('觀雲長',100);
```

```
#建立my_employees
use myemployees1;
create table myemployees1(
    id int(10),
    First_name varchar(10),
    Last_name varchar(10),
    Userid varchar(10),
    salary double(10,2)
);

create table users1(
    id int,
    userid varchar(10),
    department_id int

)
# 查看myemployees1 結構
desc  myemployees1;

#向myemployees1表中插入下列數據
```

```

id   first_name   last_name  userid   salary
1    patel        Ralph      Rpatal     895
2    Dancs        Betty      Bdancs     860
3    Biri         Ben        Bbiri      1100
4    Newman       Chad       Cnewman    750
5    Ropeburn     Audrey     Aropebur   1550

insert into myemployees1
values(1,'patel','Ralph','Rpatal',895),
(2,'Dancs','Betty','Bdancs',860),
(3,'Biri','Ben','Bbiri',1100),
(4,'Newman','Chad','Cnewman',750),
(5,'Ropeburn','Audrey','Aropebur',1550);

delete from  myemployees1;

insert into myemployees1
select 1,'patel','Ralph','Rpatal',895 union
select 2,'Dancs','Betty','Bdancs',860 union
select 3,'Biri','Ben','Bbiri',1100 union
select 4,'Newman','Chad','Cnewman',750 union
select 5,'Ropeburn','Audrey','Aropebur',1550 ;


#向user1表插入數據

1  Rpatal 10
2  Bdance 10
3  Bbiri  20
4  Cnewman 30
5  Aropebur 40

SELECT * FROM girls.users1;
insert into users1
values(1,'Rpatal', 10),
(2,'Bdance', 10),
(3,'Bbiri' ,20),
(4,'Cnewman',30),
(5,'Aropebur',40);

```

```
#將3號員工的last_name修改為'drelxer'
update myemployees1 set last_name='drelxer' where id=3;

#將所有的工資少於900的員工工資改為1000
update  myemployees1 set salary =1000 where salary<900;

#將userid為Bbiri的users1表和myemployees1表的紀錄全部刪除
delete u,e
from users1 u
join myemployees1 e on u.userid=e.userid
where u.userid='Bbiri';

#刪除所有數據
delete from myemployees1;
delete from users1;
```

```

#檢查所做的修正
SELECT * FROM girls.myemployees1;
SELECT * FROM girls.users1;

#清空myemployees1
truncate table myemployees1;
```

DDL語言

```
/*
數據定義語言
一.庫跟表的管理
1.庫的管理
創建,修改,刪除
2.表的管理
創建,修改,刪除

創建 CREATE
修改 ALTER
刪除 DROP

*/
```

```
#1.庫的創建
#語法
create database 庫名;
#case1 創建books

create database books;
#C:\ProgramData\MySQL\MySQL Server 8.0\Data

create database if not exists books;

#庫的修改
rename database books to books1;

#更改庫的字型
alter database books character set utf8;

#庫的刪除
drop database books;
drop database if exists books;

```

```
#2.表的管理
#1.表的創建
/*
語法:
 create table 表名(
     列名  列的類型  長度(約束),
     列名  列的類型  長度(約束),
     列名  列的類型  長度(約束),

     ......
     列名  列的類型  長度(約束)

)

*/

#case1

create table book(
     id int,#編號
     bName varchar(20),#書名
     price double,#價格
     authorId int ,#作者編號
     publishDate datetime #出版日期
    );


desc book;


#case2 創建author
create table author(
     id int,
     au_name varchar(20),
     nation varchar(10)

)

desc author;

```

```
#表的修改
/*
修改項目
1.列名
2.列的類型或約束
3.添加新列
4.刪除列
5.修改表名
*/
/*
 語法
 alter table 表名 add/drop/modify/change column 列名(類型 約束);
 1.添加新列
 alter table 表名 add column 列名 類型(first/after 字段名);
*/
```

```
#修改列名
alter table book change column publishdate pubdate datetime;

#修改列的類型或約束
alter table book modify column pubdate TIMESTAMP;

#添加新列
alter table author add column annual double;

#刪除列
```

```sql
alter table author drop column annual;

#修改表名
alter table author rename to book_author;
desc book_author;
```

```sql
#表的刪除
drop table book_author;
show tables;
drop table if exists book_author;
```

```sql
#通用寫法
drop database if exists 舊庫名;
create database 新庫名;
drop table if exists 舊表名;
create table 新表名;
```

```sql
#表的複製
insert into author values
(1,'村上春樹','日本'),
(2,'托爾斯泰','俄國'),
(3,'珍·奧斯汀','英國'),
(4,'陀思妥耶夫斯基','俄國');

#僅複製表的結構
create table copy like author;

#複製表的結構+數據
create table copy2
select * from author;

#僅複製部分數據
create table copy3
select id,au_name
from author
where nation='俄國';

#僅複製部分列的結構
create table copy4
select id,au_name
from author
where 1=2;

create table copy5
select id,au_name
from author
where 0;
```

1. 創建表dept1

NAME	NULL?	TYPE
id		INT(7)
NAME		VARCHAR(25)

```sql
use test;
create table dept1(
    id int(7),
    NAME varchar(25)

);
```

2.將表departments中的數據插入新表dept2中,跨庫別複製數據

```sql
create table dept2
select department_id,department_name
FROM myemployees.departments;
```

3.建立表emp5

NAME	NULL?	TYPE
id		int(7)
First_name		varchar(25)
Last_name		varchar(25)
Dept_id		int(7)

```sql
create table emp5(
    id  int(7),
    First_name varchar(25),
    Last_name varchar(25),
    Dept_id    int(7)
);

```

4.將列Last_name的長度增加到50

```sql
alter table emp5 modify column last_name varchar(50);
DESC EMP5;
```

5.根據表employees創建employees2

```sql
use myemployees ;
create table employees2 like myemployees.employees;
desc employees2;
```

6.刪除表emp5

```
use test;
drop table if exists emp5;
```

7.將表employees2重命名為emp5

```
alter table employees2 rename to emp5;
desc emp5;
```

8.在表dept和emp5中添加新列test_column，並檢查所作的操作

```
alter table emp5 add column test_column int;
alter table dept add column test_column int;
```

9.直接刪除表emp5中的列 dept_id

```
alter table emp5 drop column dept_id;
```

常見數據類型

```
/*
數值型:
   整數
   小數
       定點數
       浮點數
字元型:
       短文本 char  varchar
       長文本 text  blob(長的二進位數據)

日期型:
*/

/*
分類:
tinyint smallint mediumint int/Integer bigint
1         2         3         4          8      bytes
特點
a.如果不設置符號,默認是有符號,不須設置(unsigned)
b.插入內容數值超出整數範圍
c.如不設置長度,會有默認值 INT(11),INT(10)unsigned
d.長度代表顯示的最大寬度,如果不夠會在左邊用0填充,但必須搭配zerofill使用
*/
```

如何設置整數有符號與無符號

```
create table tab_int(
    t1 int

);

desc tab_int;

SELECT * FROM test.tab_int;

```

```
create  table tab_int (
    t1 int,
    t2 int unsigned

);

desc tab_int;

drop table if exists tab_int;

insert into tab_int values(-123456);
insert into tab_int values(123,123);
insert into tab_int values(-123456,-123456);
insert into tab_int values(2147483647,4294967296);


create  table tab_int (
    t1 int(7) ZEROFILL,
    t2 int(7) ZEROFILL

);
```

小數

```
/*分類:
1.浮點數 float  (M,D)  byte
        double (M,D) 8 byte
2.定點數  dec (M,D)
         decimal (M,D)
特點:
1 M,D
M:整數部位+小數整數部位+小數部位
D:小數部位
如果超過範圍,則顯示錯誤
2.M和D都可以省略
如果decimal,M默認為10,D默認為0
如果float double 則會根據插入的精度來決定精度
3.定點型精確度較高,如果插入的數值精度較高如貨幣運算則考慮使用
*/

#測試M,D

DROP TABLE tab_float;
create table tab_float(
    f1 float(5,2),
    f2 double(5,2),
    f3 decimal(5,2)
);

create table tab_float(
    f1 float,
    f2 double,
    f3 decimal
);
select * from tab_float;
insert into tab_float values(123.45,123.45,123.45);
insert into tab_float values(123.456,123.456,123.456);
```

```sql
insert into tab_float values(123.4,123.4,123.4);
insert into tab_float values(1523.4,1523.4,1523.4);
desc tab_float;
insert into tab_float values(123.4523,123.4523,123.4523);

#原則
/*
所選擇的類型越簡單越好,能保存的數值越小越好

*/
#三 字元型
/*
短文件
char
varchar
其他:
binary和varbinary用於保存較短的二進制
enum用於保存枚舉
set用於保存集合

長文件
text
blob(較大2進位)
特點
char    char(M)      M最大字元   固定長度字元    比較耗空間    效率高    可省略默認為1
VARCHAR  VARCHAR(M)  M最大字元   可變長度字元     比較不耗空間    效率低   不可省略默認為1
*/
#---------------------------------------------
SELECT * FROM test.tab_char;
desc test.tab_char;

create table tab_char(
       c1 enum('a','b','c')

);

insert  into tab_char values('a');
insert  into tab_char values('b');
insert  into tab_char values('c');
insert  into tab_char values('m');
insert  into tab_char values('A');

#----------------------------------------------------
SELECT * FROM test.tab_set;
desc tab_set;
create table tab_set(
    s1 set('a','b','c','d')
);

insert  into tab_set values('A');
insert  into tab_set values('a,b');
insert  into tab_set values('a,b,c');

#四.日期型
/*
分類:
date 只保存日期
```

```
time 只保存時間
year 只保存年

datetime 日期+時間
timestamp 日期+時間
特點:          空間大小bits        範圍        受時區影響
datetime       8               1000-9999         不受
timstamp       4               1970-2038          受
*/
select * from tab_date;
desc table tab_date;
create table tab_date (
      t1 datetime,
      t2 timestamp
);

insert into tab_date values(now(),now());

show variables like 'time_zone';

set time_zone='+9:00';


```

```
delete from beauty limit 1;
select * from beauty;

delete from beauty where  boyfriend_id=4 limit 1;
select * from beauty;

SELECT * FROM test.test_add_column;
create table test_add_column(
    t1 int,
    t2 int,
    t3 int

);

alter table test_add_column add column newT int first;
alter table test_add_column add column newT2 int after t2;
```

```
#常見約束
/*
含義:一種限制,用於限致表中的數據,為了保證表中的數據準確和可靠性
分類:6大約束
    not null;非空,用於保證該字段不能為空
    比如 姓名,學號
    defalt:默認,用於保證該字段有默認值
    比如性別
    primary key 主鍵 用於保證該字段的值具有唯一性,並且非空
    比如學號,員工編號等
    uniquue:唯一,用於保證該字段的值具有唯一性,可以為空
    比如座位號
    check 檢查約束,mysql不支持
    比如年齡,性別
    foreign key 外鍵,用於限制2個表的關係,用於保證該字段的值必須來自主表關聯列的值
    在副表添加外鍵約束,用於引用主表中的某列的值
    比如學生表的專業編號,員工表的部門編號,員工表的工種編號
 添加約束的時機
     1.創建表時
     2.修改表時
約束的添加分類
     1.列級約束
        6大約束大約束語法上都支持,但外鍵約束沒有效果

     2.表級約束
       除了非空,默認,其他都支持

 主鍵和唯一的大對比
                 保證唯一性      是否允許唯空        一個表中可以有多少個     是否允許組合
       主鍵            o             x                     至多有一個          o,但不推薦
       唯一            o             o                     可以有多個          o,但不推薦
 外鍵:
 1.要求在副表設置外鍵關係
 2.副表的外鍵列的類型語主表的關聯列要一致或兼容,名稱無要求
 3.主表的關聯列必須是一個key(主鍵,唯一)
 4.插入數據時,先插入主表,再插入從表
 5.刪除數據時,先刪從表再刪主表
       insert into major values(1,'java');
       insert into major values(2,'h5');
       insert into stuinfo values(1,'john','男',null,19,1);
       insert into stuinfo values(2,'lily','男',null,19,2);
*/
create table 表名(
    字段名  字段類型   列級約束
    字段名  字段類型
    表級約束
    );
```

```
#一.創建表時添加約束
#1.添加列級約束
/*
語法:
直接在字段和類型的後面追加約束類型即可
只支持:默認,非空,主鍵,唯一

*/

create database students;

use students;

create table stuinfo(
    id int primary key,#主鍵
    stuName varchar(20)  not null,#非空check
    gender char(1) check(gender='男' or gender='女'),#檢查
    seat int unique,#唯一
    age int default 18, #默認約束
    majorId int  references major(id) #外鍵

);
drop table if exists major;
create table major(
    id int primary key,
    majorName varchar(20)
);

desc  stuinfo;
#查看表中所有的索引,包括主鍵,外鍵,唯一
show index from stuinfo;

#1.添加表級約束
/*
語法:
在各個字段的最下面
(constraint 約束名)  約束類型(字段名)

*/
drop table if exists stuinfo;
create table stuinfo(
      id int unique (primary key),
      stuname varchar(20),
      gender char(1),
      seat int,
      age int,
      majorid int,

      constraint pk primary key(id),
      constraint ug unique(seat),
      constraint ck check(gender='男' or gender ='女'),
      constraint fk_stuinfo_major foreign key(majorid)  references
major(id)
);

show index from stuinfo;

#通用寫法

drop table if  exists stuinfo;
create table if not exists stuinfo(
      id int  primary key,
      stuname varchar(20) not null,
      gender char(1),
      age int default 18 ,
      seat int unique,
      majorid int,

      constraint fk_stuinfo_major foreign key(majorid)  references
major(id)
);


#---------------------------------------
#省略 constraint
drop table if  exists stuinfo;
create table if not exists stuinfo(
      id int ,
      stuname varchar(20),
      gender char(1),
      age int ,
      seat int ,
      majorid int,
      seat2 int ,

       primary key(id),
       unique(seat),
       check(gender='男' or gender ='女'),
       foreign key(majorid)  references major(id)

);

#不可多個primary key
drop table if  exists stuinfo;
create table if not exists stuinfo(
      id int ,
      stuname varchar(20),
      gender char(1),
      age int ,
      seat int ,
      majorid int,
      seat2 int ,

       primary key(id),
       unique(seat),
       primary key(stuname),
       check(gender='男' or gender ='女'),
       foreign key(majorid)  references major(id)

);

#二個列名合為一個主鍵或唯一鍵
drop table if  exists stuinfo;
create table if not exists stuinfo(
      id int ,
      stuname varchar(20),
      gender char(1),
      age int ,
      seat int ,
      majorid int,
      seat2 int ,

       primary key(id,stuname),
       unique(seat),

       check(gender='男' or gender ='女'),
       foreign key(majorid)  references major(id)

);
show index from stuinfo;
    insert into stuinfo values(1,'john','男',null,19,1,1);
    insert into stuinfo values(2,'john','男',null,18,2,2);

#---------------------------------------------------
drop table if  exists stuinfo;

create table if not exists stuinfo(
      id int ,
      stuname varchar(20),
      gender char(1),
```

```sql
        age int ,
        seat int ,
        majorid int,
        seat2 int ,

         primary key(id,stuname),
         unique(seat),

         check(gender='男' or gender ='女'),
         foreign key(majorid)  references major(id)

);
insert into stuinfo values(1,'john','男',null,19,1,1);
insert into stuinfo values(1,'lily','男',null,18,2,2);

```

```sql
#修改表時的添加約束
/*
語法:
1.添加列級約束
alter table 表名 modify column 字段名 字段類型 新約束;
2.添加表級約束
alter table 表名 add (constraint 約束名) 約束類型(字段名) (外鍵的引用);
*/
drop table if  exists stuinfo;

create table if not exists stuinfo(
      id int ,
      stuname varchar(20),
      gender char(1),
      age int ,
      seat int ,
      majorid int
    );
#1.添加非空約束
alter table stuinfo modify column stuname varchar(20) not null;
#2.添加默認約束
alter table stuinfo modify column age int default 18;
#3.添加主鍵
#列級約束
alter table stuinfo modify column id int primary key;
#表級約束
alter table stuinfo add primary key(id);
desc stuinfo;
#4.添加唯一
#列級約束
alter table stuinfo modify column seat int unique;
#表級約束
alter table stuinfo add unique(seat);
#5.添加外鍵
alter table stuinfo add foreign key(majorid) references major(id);
alter table stuinfo add cnstraint fk_stuinfo_major foreign key(majorid)
references major(id);
#三.修改表時刪除約束
#1.刪除非空約束
alter table stuinfo modify column stuname varchar(20) null;
#2.刪除默認約束
```

```sql
alter table stuinfo modify column age int;
#3.刪除主鍵
alter table stuinfo drop primary key;
#4.刪除唯一
/*語法
alter table stuinfo drop index key_name ;
*/
alter table stuinfo drop index seat_unique ;
show index from stuinfo;

ALTER TABLE `students`.`stuinfo`
ADD UNIQUE INDEX `seat_UNIQUE` (`seat` ASC) VISIBLE,
DROP INDEX `gender_UNIQUE` ;
;
#5.刪除外鍵
alter table stuinfo drop foreign key stuinfo_ibfk_1;
```

```sql
#1.在表emp2的id列中添加primary key約束(my_emp_id_pk)
alter table emp2 modify column id int primary key;
alter table emp2 add constraint my_emp_id_pk primary key(id);
#2.在表emp2中添加dept_id,並在其中定義 foreign key約束,與其有關是dept2表中id
alter table emp2 add column dept_id int;
alter table emp2 add  comstraint fk_emp2_dept2 foreign key(dept_id)
references dept2(id);
```

```
/*
           位置              支持的約束類型                     可以起約束名

列級約束     列的後面             語法都支持,但外鍵沒效果                 不可
表級約束     所有列的下面          默認與非空不支持                        可(主鍵沒效果)


*/
```

```sql
#auto_increment標識列
/*
又稱為自增加列
含義:可以不用手動插入值,系統會提供默認的序列值,從1開始
特點
1.標識列必須和主鍵搭配嗎?不一定,但要求必須是一個key
2.一個表最多只能有一個auto_increment
3.標識列只能是數值型,int,float,double可以,varchar是不行的
4.標識列可以透過 auto_increment_increment=3;更改索引值變量
可以通過手動插入值,設置起始值
*/
#一.創建表時設置標識列
drop table if exists tab_identity;
create table tab_identity(
    id int primary key,
    name varchar(20)
);

insert into tab_identity values(1,'john');

//--------------------------------
```

```sql
SELECT * FROM students.tab_identity;
create table tab_identity(
    id int primary key auto_increment,
    name varchar(20)
);
truncate table tab_identity; #刪除整個資料表
insert into tab_identity values(1,'john'); #1可以更改起始值
insert into tab_identity(id,name) values(null,'mary');
insert into tab_identity(name) values('lucy');

show variables like '%auto_increment%';
set auto_increment_increment=3;#更改索引值變量

//----------------------------

drop table if exists tab_identity;
create table tab_identity(
     id int unique auto_increment,
    name varchar(20),
    seat int unique auto_increment

);
#二.修改表時設置標識列
alter table tab_identity modify column id int primary key auto_increment;

drop table if exists tab_identity;
create table tab_identity(
     id int,
    name float,
    seat int

);
#三.修改表時刪除標識列
alter table tab_identity modify column id int ;
```

```
#TCL語言
/*
Transaction CONTROL Language 事務控制語言
事務:
一個或一組sql語句組成一個執行單元,這個執行單元要麼全部執行,要不全部不執行


case1 轉帳
張三豐     1000;
郭襄       1000;

事物的特性:
1. 原子性（Atomicity）原子性是指事務是一個不可分割的工作單位，事務中的操作要麼都發生，要麼都不發生
2. 一致性（Consistency）事務必須使資料庫從一個一致性狀態變換到另外一個一致性狀態
3. 隔離性（Isolation）事務的隔離性是指一個事務的執行不能被其他事務干擾，即一個事務內部的操作及使用的資料對併發的其他事務是隔離的，併發執行的各個事務之間不能互相干擾
4. 持久性（Durability）持久性是指一個事務一旦被提交，它對資料庫中資料的改變就是永久性的，接下來的其他操作和資料庫故障不應該對其有任何影響

事務的創建
隱式事務:事務沒有明顯的開啟與結束的標記
```

```
如 insert ,update ,delete語句
顯式事務:事務有明顯的開啟與結束的標記
前提:必須先設置自動提交功能為禁用
   set autocommit=0;
   show variables like 'autocommit';

step-1 :開啟事務
set autocommit=0;
start transaction; 可省略

step-2 :編寫事務中的sql語句(select ,insert,update,delete)
語句1;
語句2;
........

setp-3 :結束事務
commit;提交事務
rollback:回滾事務

savepoint 節點名;設置保存點
 o:會發生   x:不會發生
事務的隔離級別:     贓讀        不可重複讀        幻讀
read uncommitted  o           o              o
read committed    x           o              o
repeatable read   x           x              o
serializable      x           x              x

mysql中默認是第3個 repeatable read級別
oracle中默認是第2個 read committed級別

查看隔離級別
#舊版本 5.1
select @@tx_isolation;
#新版本mysql8.0
select  @@transaction_isolation

設置隔離級別
set session transaction isolation level 設置隔離級別;

開起事務的語句
update 表 set 張三豐的餘額=500 where name='張三豐'
update 表 set 郭襄 的餘額=1500 where name='郭襄 '
結束事務的語句

*/

update 表 set 張三豐的餘額=500 where name='張三豐'
update 表 set 郭襄 的餘額=1500 where name='郭襄 '
show engines;#顯示存儲引擎

show variables like 'autocommit';

delete from 表 where id=1;

#演示事務的使用步驟
drop table if exists account;

create table account(
    id int primary key auto_increment,
    username varchar(20),
    balance double

);

insert into account(username,balance)
values('張無忌',1000),('趙敏',1000);

SET SQL_SAFE_UPDATES=0;
SET SQL_SAFE_UPDATES=1;//結束後再開啟

#開起事務的語句
set accountit=0;
start transaction;
#編寫一組事務的語句
update account set balance=500 where username='張無忌';
update account set balance=1500 where username='趙敏';
#結束事務的語句
rollback;
commit;
#2.delete 和truncate 在事務使用時的區別
#演示delete
set autocommit=0;
start transaction;
delete from account;
rollback;
#演示truncate
set autocommit=0;
start transaction;
truncate table account;
rollback;
```

```
#開2個mysql去操作資料庫及隔離級別運用

#舊版本 5.1
select @@tx_isolation;
#新版本mysql8.0
select  @@transaction_isolation
set session transaction isolation level read uncommitted;

use test;
select * from account;
set names utf8;
set autocommit=0;
update account set username='john' where id=1;
set session transaction isolation level read committed;
set session transaction isolation level repeatable read;
insert into account values(null,'關羽',2000);
set session transaction isolation level serializable;
update account set username='wwwwwww';
```

```sql
#3.演示savepoint的使用
set autocommit=0;
start transaction;
delete from account where id=1;
savepoint a; #設置保存點
delete from account where id=6;
rollback to a;#回滾到保存點
select * from account;
```

```sql
#視圖
/*
含義:虛擬表與普通表一樣使用
mysql5.版本出現的新特性,是通過表動態生成的數據

比如:舞蹈班和普通班的對比

•             創建與法的關鍵字      是否佔用物理空間               使用

視圖       create view             保存sql邏輯                  增刪改查,只是一般不能增刪改

表         create  table           保存數據                     增刪改查

*/
```

```sql
#case.1 查詢姓張的學生名與專業名

use students;
select stuname,majorName
from stuinfo s
inner join major m on s.majorid=m.id
where s.stuname like '張%';
//------------------------------
create view v1
as
select stuname,majorName
from stuinfo s
inner join major m on s.majorid=m.id

select * from v1 where stuname like'張%';
//-------------------------------

```

```sql
#一.創建視圖
/*
語法:
create view 視圖名
as
查詢語句;

*/
use myemployees;

#1.查詢姓名中包含a字母的員工名,部門名和工種信息

```

```sql
#a.創建
create view myv1
as
select last_name,department_name,job_title
from employees e
join departments d on e.department_id=d.department_id
join jobs j on j.job_id=e.job_id;

#b.使用
select * from myv1 where last_name like '%a%';

#2.查詢各部門的平均工資級別
#a.創建視圖,查看每個部門的平均工資
use myemployees;
create view myv2
as
select avg(salary) ag,department_id
from employees
group by department_id;

#b.使用
use myemployees;
SELECT myv2.ag,g.grade_level
 FROM myemployees.myv2
 join myemployees.job_grades  g
 on myv2.ag between g.lowest_sal and g.highest_sal;


#3.查詢平均工資最低的部門信息
select * from myv2 order by ag limit 1;

#4.查詢平均工資最低的部門名與工資
use myemployees;
#a.創建myv3,沿用myv2架構
create view myv3
as
select * from myv2 order by ag limit 1;
#b.連接myv3與departments表查詢最低工資
select d.*,m.ag
from myv3 m
join departments d
on m.department_id=d.department_id;
```

```sql
#二.視圖的修改
#方式1:
/*
語法
create or replace view  視圖名
as
查詢語句;

*/

create or replace view myv3
as
select avg(salary),job_id
from employees
```

```
group by job_id;

#方式2:
/*
語法:
alter view 視圖名
as
查詢語句;

*/

alter view myv3
as
select * from employees;

#三.刪除視圖
/*
語法:
 drop view 視圖名,視圖名,...;


*/

  drop view myv1,myv2,myv3;

#四.查看視圖
 desc myv3;
 show create view myv3;
#dos下執行
 show create view myv3;
 show create view myv3\G;

#五.視圖的更新
create or replace view myv1
as
select last_name,email,salary*12*(1+ifnull(commission_pct,0)) "annual salary"
from employees;

SELECT * FROM myemployees.myv1;
#1.插入
insert into myv1 values('fei','fei@gmail.com');
#2.修改
SET SQL_SAFE_UPDATES=0;
update myv1 set last_name='張無忌' where last_name='張飛';
#3.刪除
delete from myv1 where last_name='張無忌';
#具備以下特點的視圖是不允許修改的更新

#A.包含以下關鍵字的sql語句：分組函數、distinct、group by、having、union或者union all--------

create or replace view myv1
as
select max(salary) m,department_id
from employees
group by department_id;

```

```
select * from myv1;
#更新(結果無法更新)
update myv1 set m=9000 where department_id=10

#B.常量視圖------------------------------------------------------------------
----------
create or replace view myv2
as
select 'john' name;

select * from myv2;
#更新(結果無法更新)
update myv2 set name='mary';

#C.Select中包含子查詢-----------------------------------------------------------
-----
#@
create  or replace view myv3
as
select (select max(salary) from employees) 最高工資;

#@
create  or replace view myv3
as
select department_id,(select max(salary) from employees) 最高工資
from departments;

#更新

SELECT * FROM myemployees.myv3;
update myv3 set 最高工資=10000;#(結果無法更新)

#D.join------------------------------------------------------------------
create or replace view myv4
as
select last_name,department_name
from employees e
join departments d
on e.department_id=d.department_id;
#更新
SELECT * FROM myemployees.myv4;
update myv4 set last_name='張飛' where last_name='whalen';
insert into myv4 values('betty','new taipei city');#(結果無法更新)

#E.from一個不能更新的視圖--------------------------------------------------

create or replace view  myv5
as
select * from  myv3;

#更新
select * from myv5;

update myv5 set 最高工資=10000 where department_id=60;#(結果無法更新)



#F.where子句的子查詢引用了from子句中的表---------------------------------
```

```
create or replace view myv6
as
select last_name,email,salary
from employees
where employee_id in(
        select manager_id
        from employees
        where manager_id is not null
);
SELECT * FROM myemployees.myv6;
#更新
update myv6 set salary=10000 where last_name='k_ing';#(結果無法更新)

```

Case1.創建視圖emp_v1,要求查詢電話號'碼以'011'開頭的員工姓名與工資,郵箱

```
create or replace view emp_v1
as
select last_name,salary,email
from employees
where phone_number like '011%';

select * from emp_v1;
```

Case2.創建視圖emp_v2,要求查詢部門的最高工資高於12000的部門信息

```
#方法1
create or replace view emp_v2
as
select max(salary),department_id
from employees
group by department_id
having max(salary)>12000;

select * from emp_v2;

#方法2
select d.*,m.mx_dep
from departments d
join (
    select max(salary) mx_dep,department_id
    from employees
    group by department_id
    having max(salary)>12000

)  m
on m.department_id=d.department_id;

#方法3
create or replace view emp_v2
as
select max(salary),department_id
from employees
group by department_id
having max(salary)>12000;

```

```sql
select * from emp_v2;

select d.*,m.mx_dep
from departments d
join emp_v2 m
on m.department_id=d.department_id;

#更改table字型
ALTER TABLE `myemployees`.`departments`
CHARACTER SET = utf8 , COLLATE = utf8_general_ci ;


desc departments;
describe departments;

show create table departments;
```

```sql

/*
1.創建表Book,字段如下
bid 整數,要求主鍵
bname  字母型,要求設置唯一鍵,非空
price 浮點型,默認值10
btypeId 型別編號,要求引用bookType表的 id字段

已知bookType表(不用創建),字段如下:
id
name*/


create table Book
   bid int primary key,
   bname varchar(20) unique not null,
   price float default 10,
   btypeId int,
   foreign key(btypeid) references bookType(id)

);


#2.開啟事務向表中插入一行數據,結束

 set autocommit=0;
 insert into book(bid,bname,price,btypeId)
 values(1,'Java2',100,1)
 rollback;

#3.創建視圖,實現查詢價格>100的書名和類型名

create view myv1
as
select book b
join bookType t on b.typeid=t.id
where price>100;


#4.修改視圖,查詢價格在90~120之間的書名和價格
```

```
create or replace view myv1
as
select bname,price
from book
where price between 90 and 120;

5.刪除剛才建的視圖

drop view myv1;



```

```
show index from  major;

show index from stuinfo;

alter table stuinfo drop foreign key fk_stuinfo_major;

#傳統的方式添加外鍵alter
alter table stuinfo add constraint fk_stu_major foreign key (majorid)
references major(id);

select * from major;
insert into major
values(1,'java'),(2,'html5'),(3,'python');

select * from stuinfo;
insert into stuinfo
select 1,'john1','m',null,null,1  union all
select 1,'john2','m',null,null,1  union all
select 2,'john1','m',null,null,2  union all
select 3,'john3','m',null,null,2  union all
select 4,'john4','m',null,null,1  union all
select 5,'john5','m',null,null,1  union all
select 6,'john6','m',null,null,3  union all
select 7,'john7','m',null,null,3  union all
select 8,'john8','m',null,null,1
#刪除專業表上的3號專業
delete from major where id=3;#(結果無法刪除,因有foreign key,且刪除須從副表先刪)
alter table stuinfo drop foreign key fk_stu_major;

#方式一,級連刪除alter
alter table stuinfo add constraint fk_stu_major foreign key (majorid)
references major(id) on delete cascade;

delete from major where id=3;#(結果可刪除,因有上一句 on delete cascade)
select * from major;
select * from stuinfo;

#方式一,級連置空
alter table stuinfo add constraint fk_stu_major foreign key (majorid)
references major(id) on delete set null;
delete from major where id=2;
```

```
select * from major;
SELECT * FROM students.stuinfo;



```

```
#變量
/*
系統變量:
        全局變量
        會話變量
自定義變量:
        用戶變量
        局部變量
*/

#一 系統變量
說明:變量由系統提供,不是用戶定義,屬於服務器層面
使用的語法:
1.查看所有的系統變量
show session variables;
show global variables;

2.查看滿足條件的部分系統變量
show global varables like '%char%';
show session varables like '%char%';

3.查看指定的謀個系統變量的值
select @@global.系統變量名;
select @@session.系統變量名;

4.幫系統變量賦值
方式1
set gloabal 系統變量名=值;
set session 系統變量名=值;
方式2
set @@gloabal.系統變量名=值;
set @@session.系統變量名=值;

注意:
全局級別,需加global
會話級別,需加session(系統默認是session)



*/
```

```
#1.全局變量
作用域:服務器每次啟動將為所有的全局變量賦初始值,針對所有的會話連接有效,但不能跨重新啟動
mysql
#a.查看所有的全局變量
show global variables;
#b.查看部分的全局變量
show global variables like '%char%';
#c.查看指定的全局變量
select @@global.autocommit;
select @@transaction_isolation; #mysql8.1 version
select @@tx_isolation; #mysql5.1 version
#d.為某個指定的全局變量賦值
set @@global.autocommit=0;
select @@global.autocommit;
#2.會話變量
作用域:僅僅貞對於當前會話(連接)有效
#a.查看所有的會話變量
show session variables;
show  variables;
#b.查看部分的會話變量
show session variables like '%char%';
show  variables like '%char%';
#c.查看指定的某個會話變量
select @@transaction_isolation;
select @@session.transaction_isolation;
#d.為某個指定的會話變量賦值
方式1:
set @@seeeion.transaction_isolation='read_uncommitted';
方式2:
set seeeion.transaction_isolation='read_committed';

#二.自定義變量
/*
說明:變量是用戶自定義的,不是由系統給的
使用步驟:
聲明
賦值
使用(查看,比較,運算)

*/
#一.用戶變量
作用域:針對於當前會話(連接)有效,同於會話變量的作用域
應用在任何地方,也就是begin end裡面或begin end 外面
賦值的操作符號:  = or :=
#a.聲明並初始化
set @用戶變量名=值;
set @用戶變量名:=值;
select  @用戶變量名:=值;
#b.賦值(更新用戶變量的值)
方式1:通過set或select
    set @用戶變量名=值;
    set @用戶變量名:=值;
    select  @用戶變量名:=值;
方式2:通過select into
   select 字段 into @變量名
   from 表;
#c.使用(查看用戶變量的值)
   select @用戶變量名;

#二.局部變量
作用域:僅僅在定義他的begin end中有效
應用在begin end中第一句話
#a.聲明
declare 變量名  類型;
declare 變量名  類型 default 值;
#b.賦值
方式1:通過set或select
```

```
    set @局部變量名=值;
    set @局部變量名:=值;
    select  @局部變量名:=值;
方式2:通過select into
   select 字段 into @局部變量名
   from 表;
#c.使用(查看局部變量的值)
   select @局部變量名;


對比用戶變量與局部變量:
                  作用域            定義和使用的位置             語法
用戶變量        當前會話          會畫中的任何地方                  必須加@符號,不用限定類
型

局部變量     begin end       只能在begin end中,且為第一句話   一般不用加@,要限定類型


case 1 用戶變量
#聲明並初始化
set @name='john';
set @name=100;
set @count=1;
#賦值(更新用戶變量的值)
select count(*) into @count
from employees;
#查看
select @count;


case 2 聲明2個變量並賦初始值,求和,並印出資料
#a 用戶變量
set @m=1;
set @n=2;
set @sum=@m+@n;
select @sum;


#b 局部變量
declare m int default 1;
declare n int default 2;
declare sum int;
set sum=m+n;
select sum;


```

```
#存儲過程和函數
/*
存儲過程和函數:類似於Java中方法
好處:
1.提高代碼的重用姓
2.簡化操作
3.減少了編譯次數與連接資料庫服務器的連接次數,增加效率
*/
#存儲過程
/*
```

```
含義:一組預先編譯好的SQL語句集合,理解成批次處裡
1.提高代碼的重用姓
2.簡化操作
3.減少了編譯次數與連接資料庫服務器的連接次數,增加效率
*/
1.創建語法
create  procedure 存儲過程名(參數存儲過程名(參數列表)
 begin
      存儲過程體(一組合法的SQL語句)
 end
注意:
a.參數列表包含3部分
參數模式   參數名   參數類型
舉例:
in stuname varchar(20)
參數模式:
in :該參數可以做為輸入,也就是該參數需要調用方傳入值
out :該參數可以做為輸出也可以作為返回值
inout :該參數可以做為輸入也可輸出,也就是該參數既須可當輸入值也可當返回值

b.如果存儲過程體僅僅一句話,begin end 可以省略
存儲過程體中每條SQL語句結尾必須加分號;
存儲過程的結尾可以使用 DELIMITER 重新設置

語法:
DELIMITER 結束標記
案例:
DELIMITER $

2.調用語法
call 存儲過程名(實參列表);
#a.空參列表
case 1.插入到admin表中5條紀錄
SELECT * FROM girls.admin;


--------------------------------------------------------------

dos下執行
mysql -h localhost -P 3306 -u root -p1234
use girls;
DELIMITER $

create procedure myp1()
begin
    insert into admin(username,password) values('john1','00000'),
('lily','00000'),('rose','00000'),('jack','00000'),('tom','00000');
    end $
#調用
call myp1()$    #mysql5.1版本
call myp1() #mysql 8.1版本
SELECT * FROM girls.admin
----------------------------------------------------------

#b.創建帶in模式的參數的存儲過程
case 1.創建過程實現,根據女神名,查詢對應的男神資料
use girls;
DELIMITER $
```

```

create procedure myp2(in beautyName varchar(20))
begin
    select bo.*
    from boys bo
    right join beauty b on bo.id=b.boyfriend_id
    where b.name=beautyName;
 end $

#調用
call myp2('Angelababy')$ #mysql5.1版本
call myp2('Angelababy') #mysql 8.1版本
call myp2('小昭')

----------------------------------------------------------------------------------------
case 2.創建存儲過程實現,用戶是否登錄成功
use girls;
DELIMITER $
create procedure  myp3(in username varchar(20),in password varchar(20))
begin
    declare result varchar(20) default '';  #聲明初始化
     select count(*) into result  #賦值
     from admin
     where admin.username=username
     and admin.password=password;
     select result;  #使用
end $

#調用
call myp3('張飛','8888')$ #mysql5.1版本
call myp3('張飛','8888') #mysql 8.1版本
call myp3('John','8888')


 ------------------------------------------------------------------
use girls;
DELIMITER $
create procedure  myp4(in username varchar(20),in password varchar(20))
begin
    declare result varchar(20) default 0;  #聲明初始化
     select count(*) into result  #賦值
     from admin
     where admin.username=username
     and admin.password=password;
     select if  (result>0,'成功','失敗');  #使用
end $


#調用
call myp4('張飛','8888')$ #mysql5.1版本
call myp4('張飛','8888') #mysql 8.1版本
call myp4('John','8888')
```

```
#c.創建帶out模式的參數的存儲過程

case 1.創建過程實現,根據女神名,返回對應的男神資料
```

```sql
use girls;
DELIMITER $
create procedure  myp5(in beautyname varchar(20),out boyname varchar(20))
begin
    select bo.boyName into boyName
    from boys bo
    right join beauty b on bo.id=b.boyfriend_id
    where b.name=beautyName;

end$
#調用
set @bName$
call myp5('小昭',@bName)$  #mysql5.1版本
call myp5('小昭',@bName) #mysql 8.1版本

#查看

select @bName$   #mysql5.1版本
select @bName    #mysql 8.1版本

-------------------------------------------------------------------
case 2.根據女神名,返回對應的男神名與其魅力值

use girls;
DELIMITER $
create procedure  myp6(in beautyname varchar(20),out boyname varchar(20),out 
userCP int)
begin
    select bo.boyName ,bo.userCP into boyName,userCP
    from boys bo
    right join beauty b on bo.id=b.boyfriend_id
    where b.name=beautyName;


end$

#調用
set @bName$ #前面有設,就不需再設
call myp6('小昭',@bName,@userCP)$  #mysql5.1版本
call myp6('小昭',@bName,@userCP) #mysql 8.1版本

#查看

select @bName ,@userCP$   #mysql5.1版本
select @bName,@userCP  #mysql 8.1版本
```

```sql
#d.創建帶inout模式的參數的存儲過程
#case 1:傳入a and b兩個值,最終a and b都翻倍返回
use girls;
DELIMITER $
create procedure  myp7(inout a int,inout b int)
begin
    set a=a*2;
    set b=b*2;

end$
#調用
```

```
set @m=10
set @n=20

call myp7(@m,@n)
#查看
select @m,@n
-------------------------------------------------------------------
#二.刪儲存儲過程
語法:

drop procedure 存儲過程名
drop procedure myp1;

#三.查看存儲過程的信息

show create procedure myp2;
```

```
1、創建存儲過程或函數實現傳入用戶名和密碼，插入到admin表中
use girls;
DELIMITER $
create procedure  test_pro1(in  username varchar(20),in loginPwd varchar(20))
begin
    insert into admin(admin.username,password)
    values(username,loginPwd);

end$
call test_pro1('admin','0000')

select * from admin;

```

```
2、創建存儲過程或函數實現傳入女神編號，返回女神名稱和女神電話
use girls;
DELIMITER $
create procedure  test_pro2(in id int,out name varchar(20),out phone varchar(20))
begin
    select b.name,b.phone into name,phone
    from beauty b
    where b.id=id;

end$

call test_pro2(1,@n,@p)
select @n,@p
```

```
3、創建存儲存儲過程或函數實現傳人兩個女神生日，返回大小
use girls;
DELIMITER $
create procedure  test_pro3(in birth1 datetime,in birth2 datetime,out result int)
begin
    select datediff(birth1,birth2) into result;

end$

call test_pro3('1998-1-1',now(),@result)

select @result
```

```
4、創建存儲過程或函數實現傳入一個日期，格式化成xx年xx月xx日並返回
use girls;
DELIMITER $

create procedure test_pro4(in mydate datetime,out strDate varchar(50))
begin
               select date_format(mydate ,'%y年%m月%d日') into strDate;

end$

call test_pro4(now(),@str)

select @str
```

```
5、創建存儲過程或函數實現傳入女神名稱，返回：女神 and 男神 格式的字串
如 傳入 ：小昭
返回： 小昭 and 張無忌

use girls;
DELIMITER $

create procedure test_pro5(in beautyName varchar(20),out str varchar(50))
begin
      select concat(beautyName,'and',boyName) into str
      from boys bo
      right join beauty b on b.boyfriend_id=bo.id
      where b.name=beautyName;

end$

call test_pro5('小昭',@str)

select @str
-----------------------------------------------------------------------------
use girls;
DELIMITER $

create procedure test_pro5(in beautyName varchar(20),out str varchar(50))
begin
      select concat(beautyName,'and',ifnull(boyName,'null')) into str
      from boys bo
      right join beauty b on b.boyfriend_id=bo.id
      where b.name=beautyName;

end$

call test_pro5('柳岩',@str)

select @str
```

```
6、創建存儲過程或函數，根據傳入的條目數和起始索引，查詢beauty表的記錄
drop procedure test_pro6
use girls;
DELIMITER $

create procedure test_pro6(in size int ,in startIndex int)
begin
         select * from beauty limit size,startIndex;


end$

call test_pro6(3,5)
```

```
#函數
/*
含義:一組預先編譯好的SQL語句集合,理解成批次處裡
1.提高代碼的重用姓
2.簡化操作
3.減少了編譯次數與連接資料庫服務器的連接次數,增加效率
區別:
存儲過程:可以有0個返回,也可以有多個返回,適合做批量插入,批量更新
函數:有返回值且僅有一個,適合做處理數據後,返回一個結果

*/

#一.創建語法
create function 函數名(參數列表) returns 返回類型
   begin
         函數體
   end

/*
注意:
1.參數列表 包含2部分
  參數名  參數類型
2.函數體:肯定會有return語句,如果沒有會報錯
如果return語句沒有放在函數體的最後也不會報錯,但不建議

return 值;

3.函數體中僅有一句話,則可以省略begin end
4.使用delimiter語句設置結束標記

delimiter $;
*/

```

```sql
#二.調用語句
select 函數名(參數列表)


--------------------------案例演示1------------------------------

#1.無參數有返回值
#返回公司員工的個數
use myemployees;
DELIMITER $         #讓mysql整批處理程式碼
create function myf1()  returns int
DETERMINISTIC
begin
      declare c int default 0 ;    #定義局部變量
      select count(*)  into  c    #賦值
      from employees;
      return c;
end $


#調用

select myf1()

--------------------------案例演示2------------------------------
#2.有參數有返回值
# 根據員工名,返回其工資
use myemployees;
DELIMITER $   #讓mysql整批處理程式碼
create function myf2(empName  varchar(20))  returns double
DETERMINISTIC
begin
        set @sal=0;   #定義用戶變量
        select salary into @sal  #賦值
        from employees
        where last_name=empName;
        return @sal;
end$

#調用

select myf2('khoo')


---------------------------------------------------------------
#2 根據部門名,返回該部門的平均工資
use myemployees;
DELIMITER $   #讓mysql整批處理程式碼
create function myf3(deptName  varchar(20))  returns double
DETERMINISTIC
begin
      declare sal double;
      select avg(salary) into sal
      from employees e
      join departments d on e.department_id=d.department_id
      where d.department_name=deptName ;
      return sal;

```

```
end$

#調用

select myf3('IT')

--------------------------------------------------------------------------------
--

#三.查看函數
show create function myf3;
#四.刪除函數
drop function myf3;


--------------------------------------------------------------------------------
-
#案例
#一.創建函數,傳入2個float,返回2個的和
use myemployees;
DELIMITER $    #讓mysql整批處理程式碼
create function myf4(num1 float,num2 float) returns float

DETERMINISTIC
begin
       declare sum float default 0;
       set sum=num1+num2;
       return sum;


end$

select myf4(1,5)
```

```
#流程控制結構
/*
順序結構:程式從上往下執行
分支結構:程式從2條或多條路徑中選擇一條去執行
循環結構:程序在滿足一定條件基礎上,重複執行一段代碼
*/

#一.分支結構
#a.if函數
語法:
if(表達式1,表達式2,表達式3)
執行順序
如果表達式1成立,則if函數返回表達式2的值,否則返回表達式3的值

應用:任何地方

#2.case結構
情況1:類似於Java中switch語句,一般用於實現的等值判斷
語法:
     case  變量|表達式|字段
     when 要判斷的值  then 返回的值1或語句1 ;
     when 要判斷的值  then 返回的值2或語句2 ;
     .....
     else 要返回的值n或語句n ;
     end case;

情況2:類似於Java中多重if語句,一般用於實現的區間判斷
語法:
     case
     when 要判斷的條件1  then 返回的值1或語句1;
     when 要判斷的條件2   then 返回的值2或語句2;
     .....
     else 要返回的值n或語句n;
     end  case;
特點:
a.可以做為表達式,嵌套在其他語句中使用,可以放在任何地方,begin end中或begin end外面
可以做為獨立的語句去使用,只能放在begin end中
b.如果when中的值或條件成立,則執行對應的then後面的語句,並結束case
如果都不滿足,則執行else中的語句或值
c.else可以省略,如果else省略了,並且所有when條件都不滿足,則返回null


---------------------------------------------------------
案例
創建存儲過程,根據傳入成績來顯示等級,比如傳入成績:100~90,顯示A, 90~80,顯示B,80~60,顯示C,否則顯示D
use myemployees;
DELIMITER $

create procedure test_case(in score int)
begin
     case
     when score<=100 AND score >=90 then select 'A';
     when score<90 and score >=80 then select 'B';
     when score<80 and score >=60 then select 'C';
     ELSE SELECT 'D';
     END CASE ;
end$
CALL TEST_CASE(95)


#3.if結構
/*
功能:實現多重分支
語法:
     if 條件1  then 語句1;
 elseif 條件2  then 語句2;
 ......
 (else 語句n;)

 end if;
 應用在begin end中
*/
-----------------------------------------------------------------------------
--------
案例
創建存儲過程,根據傳入成績來顯示等級,比如傳入成績:100~90,返回A, 90~80,返回B,80~60,返回C,否則返回D


 use myemployees;
```

```
DELIMITER $

 create function test_if(score int) returns char
 DETERMINISTIC
 begin
       if score>=90 and score<=100 then return 'A';
       elseif score >=80 then return 'B';
       elseif score >=60 then return 'C';
       else return 'D';
       end if;

 end$

 select test_if(86)

 #二.循環結構

 分類:
 while,loop,repeat
 循環控制:
 iterate類似於 continue 繼續,結束本次循環,繼續下一次
 leave 類似於 break,跳出,結束當前所在的循環

#1.while

語法:
    ( 標籤:)while循環條件  do
       循環體;
       end while(標籤);


聯想Java:
  while(循環條件) {
      循環體;

  }



#2.loop

語法:
    ( 標籤:)loop
       循環體;
       end loop(標籤);
可以用來摸擬簡單的死循環


#3.repeat
語法:
    ( 標籤:)repeat
       循環體;
       until 結束循環條件
       end repeat(標籤);

----------------------------------------------------------------------


#沒有添加循環控制語句
```

```
案例:批量插入,根據次數插入到admin表中多條紀錄
drop procedure pro_while

use girls;
DELIMITER $
create procedure pro_while(in insertCount int)

begin
      declare i int default 1;
      while i<=insertCount DO
          insert into admin(username,password)
values(concat('rose',i),'666');
           set i=i+1;
           end while;
end$

drop procedure pro_while

call pro_while(100)

select * from admin;


/*
int i=1;
while (i<=insertcount) {
      //插入
      i++;


}



*/


---------------------------------------------------------------------------

#2.添加leave語句
案例:批量插入,根據次數插入到admin表中多條紀錄,如果次數>20則停止

drop procedure test_while1
use girls;
DELIMITER $
create procedure test_while1(in insertCount int)

begin
    declare i int default 1;
    a:while i<=insertCount do
         insert into admin(username,password)
values(concat('xiaohua',i),'0000');

         if i>=20 then leave a;
         end if;
         set i=i+1;
         end while a;
end$
```

```

call test_while1(100)

select * from admin;
-----------------------------------------------------------------------

#3添加iterate語句
案例:批量插入,根據次數插入到admin表中多條紀錄,只插入偶數次
truncate table admin;
drop procedure test_while1
use girls;
DELIMITER $
create procedure test_while1(in insertCount int)

begin
    declare i int default 0;
    a:while i<=insertCount do
         set i=i+1;
         if mod(i,2)!=0 then iterate a;
         end if;
         insert into admin(username,password)
values(concat('xiaohua',i),'0000');



         end while a;
end$

call test_while1(100)

select * from admin;




/*
int i=1;
while(i<=insertCount) {
    if(i%2!=0)
           插入
      i++;
}
---------------------------
int i=0;
while(i<=insertCount) {
    i++;
    if(i%2==0)
          continue;
     }
     插入
}

*/
```

```
/*
一.已知表stringconyent
其中字段
```

```
id自動增長
content varchar(20)

向該表插入指定個數的,隨機的字符串

*/

drop table if exists stringcontent;
create table stringcontetent(
    id int primary key auto_increment,
    content varchar(20)

);

use girls;
DELIMITER $
create procedure test_randstr_insert(in insertCount int)

begin
    declare i int default 1; #定義循環變數,表示插入的次數
    declare str varchar(26) default 'abcdefghijklmnopqrstuvwxyz';
    declare startIndex int default 1;  #代表起始索引
    declare len int default 1; #代表擷取的字母長度
    while i<=insertCount do

      set len=floor(rand() *(20-startIndex+1)+1);
     #產生一個隨機的整數,代表擷取長度,1-(26-startIndex+1)
         set startIndex=floor(rand()*26+1);   #產生一個隨機的整數,代表起始索引1~26
         insert into stringcontetent(content) values(substr(str,startIndex,len));
    set i=i+1;   #循環變量更新
    end while;


end$

call test_randstr_insert(10)

SELECT * FROM girls.stringcontetent;

```