# Funtion

# Function

- Function
  - ✓ 입력 데이터를 이용해서 연산을 수행한 후 출력 값을 만들어 내는 개체
  - ✓ 함수에 입력하는 데이터를 Argument(Parameter, 매개변수, 인수) 라 하고 출력을 리턴 값이라고도 함
  - ✓ 함수의 종류
    - Scala 함수: 값 하나를 계산하는 함수
    - Group 함수: 여러 개의 값으로부터 통계 값을 계산하는 함수
    - System 함수: NULL 관련 처리나 타입 변환 등을 수행하는 함수

# 숫자 관련 함수

▶ ABS(숫자) : 절대값 출력.

select abs(123);

▶ CEILING(숫자) : 값보다 큰 정수 중 가장 작은 수.

--양수일 경우는 소숫점 자리에서 무조건 반올림(4.0과 같은 0 값은 제외)

--음수일 경우는 소숫점 자리를 무조건 버림

select ceiling(4.0); select ceiling(4.1); select ceil(4.9);

▶ FLOOR(숫자) : 값보다 작은 정수 중 가장 큰 수[실수를 무조건 버림].

--음수일 경우는 [.0/.00/.000/...] 을 제외하고 무조건 소숫점을 버리고 반내림(?)

select floor(4.0); select floor(4.1); select floor(4.9); select floor(-4.6789);

▶ ROUND(숫자,자릿수) : 숫자를 소수점 이하 자릿수에서 반올림

--자릿수를 생략하면 소숫점이 5 이상일 때 반올림/자릿수를 지정하면 지정한 자리수에서 반올림

select round(4.5); select round(4.55);

select round(-4.5); select round(4.556);

select round(4.556,0); select round(4.556,1);

select round(4.556,2); select round(45.556,-1); select round(455.556,-2);

# 숫자 관련 함수

▶ TRUNCATE(숫자,자릿수) : 숫자를 소수점 이하 자릿수에서 버림.
➔ 소숫점 이전으로 정하면 소숫점이하는 버리고 나머지 값은 0 값으로 처리
/ 예) truncate(9999,-3) --> 9000
➔ 소숫점이하로 정하며, 해당숫자가 자릿수보다 소숫점이 모자랄경우 0 값 대치
/ 예) truncate(999,3) --> 999.000
음수일 경우는 해당자릿수에서 소숫점을 버리면서 무조건 반올림
==>(자릿수 숫자에서 이후 숫자가 0 일 경우는 제외 / 예)-4.0,0/-400,-2/-4.1230,4)
==>음수도 소숫점이하로 정하며, 자릿수보다 소숫점이 모자랄경우 0 값 대치
➔ 소숫점 이전으로 정하면 소숫점이하는 버리고 나머지 값은 역시 0 값으로 처리

▶ POW(X,Y) 또는 POWER(X,Y) : X의 Y승
--소숫점이 있는 경우도 실행, 단 음수는 양수로 승처리
select pow(-2.5,2);  select pow(1.5,2);

▶ MOD (분자, 분모) : 분자를 분모로 나눈 나머지를 구한다.(연산자 %와 같음)
select mod(12,5);  ==> 2  select 12%5;  ==> 2

▶ GREATEST(숫자1,숫자2,숫자3...) : 주어진 수 중 제일 큰 수 리턴.
select greatest(100,101,90);

▶ LEAST(숫자1,숫자2,숫자3...) : 주어진 수 중 제일 작은 수 리턴.
select least(100,101,90);

▶ INTERVAL(a,b,c,d.....) : a(숫자)의 위치 반환
--두 번째 이후는 오름차순 정렬이 되어야 함
INTERVAL(5,2,4,6,8) ==> 2(5는 4와 6사이에 존재, 4~6사이의 위치가 앞에서 2번째)
select interval(4,1,2,3,5,6);

# 문자 관련 함수

▶ ASCII(문자) : 문자의 아스키 코드값 리턴.

SELECT ASCII('문자');  select ascii('A');

▶ CONCAT('문자열1','문자열2','문자열3'...) : 문자열들을 이어준다.

select concat('ASP,','PHP,','SQL',' WEB STUDY');

▶ INSERT('문자열','시작위치','길이','새로운문자열') : 문자열의 시작위치부터 길이만큼 새로운 문자열로 대치

'시작위치' 와 '길이'는 문자열이 아니므로 작은따옴표로 굳이 묶어주지 않아도 된다.

select insert('MySql web study','7','3','offline');

select insert('MySql web study',7,3,'offline');

▶ REPLACE('문자열','기존문자열','바뀔문자열') : 문자열 중 기존문자열을 바뀔 문자열로 바꾼다.

select replace('MySql web study','web','offline');

▶ INSTR('문자열','찾는문자열') : 문자열 중 찾는 문자열의 위치값을 출력 -> 값이 존재하지 않으면 0값 리턴

select instr('MySql web study','s');

select instr('MySql web study','S');

# 문자 관련 함수

▶LEFT('문자열',개수) : 문자열 중 왼쪽에서 개수만큼을 추출.
select left('MySql web study',5);  select left('MySql web study','5');

▶ RIGHT('문자열',개수) : 문자열 중 오른쪽에서 개수만큼을 추출.
select right('MySql web study',5);  select right('MySql web study','5');

▶ MID('문자열',시작위치,개수) : 문자열 중 시작위치부터 개수만큼 출력
select mid('MySql web study',7,3);  select mid('MySql web study','7','3');

▶ SUBSTRING('문자열',시작위치,개수) : 문자열 중 시작위치부터 개수만큼 출력
select substring('Mysql web study',11,5);
select substring('Mysql web study','11','5');

▶ LTRIM('문자열') : 문자열 중 왼쪽의 공백을 없앤다.
select ltrim('  web study');

▶ RTRIM('문자열') : 문자열 중 오른쪽의 공백을 없앤다.
select rtrim('web study  ');

▶ TRIM('문자열') : 양쪽 모두의 공백을 없앤다.
select trim('  web study  ');

▶ LCASE('문자열') 또는 LOWER('문자열') : 소문자로 바꾼다.
select lcase('MYSQL');  select lower('MySQL');

▶ UCASE('문자열') 또는 UPPER('문자열') : 대문자로 바꾼다.
select ucase('mySql');  select upper('mysql');

▶ REVERSE('문자열') : 문자열을 반대로 나열한다.
예) REVERSE('abcde') ==> edcba
select reverse('lqSyM');

# 날짜 관련 함수

▶ NOW() 또는 SYSDATE() 또는 CURRENT_TIMESTAMP() 현재 날짜와 시간 출력

※ 함수의 상황이 숫자인지 문자열인지에 따라
YYYYMMDDHHMMSS 또는 'YYYY-MM-DD HH:MM:SS' 형식으로 반환한다.

예) select now(); ==> '2001-05-07 09:10:10'
select now() + 0; ==> 20010507091010

▶ CURDATE() 또는 CURRENT_DATE() 현재 날짜 출력

※ 함수의 상황이 숫자인지 문자열인지에 따라
YYYYMMDD 또는 'YYYY-MM-DD 형식으로 반환한다.

예) select curdate(); ==> '2001-05-07'
select curdate() + 0; ==> 20010507

▶ CURTIME() 또는 CURRENT_TIME() 현재 시간 출력

※ 함수의 상황이 숫자인지 문자열인지에 따라 HHMMSS 또는 'HH:MM:SS' 형식

예) select curtime(); ==> '09:10:10'
select curtime() + 0; ==> 091010

▶ DATE_ADD(날짜,INTERVAL 기준값) 날짜에서 기준값 만큼 더한다.

※ 기준값 : YEAR, MONTH, DAY, HOUR, MINUTE, SECOND

예) select date_add(now(), interval 2 day); ==> 오늘보다 2일 후의 날짜와 시간
select date_add(curdate(), interval 2 day); ==> 오늘보다 2일 후의 날짜 출력.

# 날짜 관련 함수

▶ DATE_SUB(날짜,INTERVAL 기준값) 날짜에서 기준값 만큼 뺀다.

※ 기준값 : YEAR, MONTH, DAY, HOUR, MINUTE, SECOND

```
select date_sub(now(),interval 2 day);   ==> 오늘보다 2일 전의 날짜와 시간 출력.
select date_sub(curdate(), interval 2 day);  ==> 오늘보다 2일 전의 날짜 출력.
```

▶ YEAR(날짜) : 날짜의 연도 출력.

```
select year('20000101');          select year(20000101);
select year('2000-01-01');        select year(now());
select year(curdate());           select year(date_add(now(),interval 2 year));
select year(date_sub(curdate(),interval 2 year));
```

▶ MONTH(날짜) : 날짜의 월 출력.

```
select month('20001231');             select month(20001231);
select month('2000-12-31');           select month(now());
select month(curdate());              select month(date_add(now(),interval 2 month));
select month(date_sub(curdate(),interval 2 month));
```

▶ MONTHNAME(날짜) : 날짜의 월을 영어로 출력.

```
select monthname(20021221);       select monthname('20000721');
select monthname('2000-08-10');   select monthname(now());
select monthname(curdate());
select monthname(date_add(now(),interval 17 month));
select monthname(date_sub(curdate(),interval 11 month));
```

# 날짜 관련 함수

▶ DAYNAME(날짜) : 날짜의 요일일 영어로 출력.

```
select dayname(20000121);          select dayname('20010123');
select dayname('2001-06-22');      select dayname(now());
select dayname(curdate());
select dayname(date_add(now(),interval 21 day));
select dayname(date_sub(curdate(),interval 333 day));
```

▶ DAYOFMONTH(날짜) : 날짜의 월별 일자 출력.

```
select dayofmonth(20030112);      select dayofmonth('20011231');
select dayofmonth('2001-12-23'); select dayofmonth(now());
select dayofmonth(curdate());
 select dayofmonth(date_add(now(),interval 56 day));
select dayofmonth(date_sub(curdate(),interval 33 day));
```

▶ DAYOFWEEK(날짜) : 날짜의 주별 일자 출력(월요일(2),화요일(3)...일요일(1))

```
select dayofweek(20011209);       select dayofweek('20001212');
select dayofweek('2003-03-21');  select dayofweek(now());
select dayofweek(curdate());
 select dayofweek(date_add(now(),interval 23 day));
select dayofweek(date_sub(curdate(),interval 31 day));
```

# 날짜 관련 함수

▶ WEEKDAY(날짜) : 날짜의 주별 일자 출력(월요일(0),화요일(1)...일요일(6))

```
select weekday(20000101);       select weekday('20030223');
select weekday('2002-10-26');   select weekday(now());
select weekday(curdate());      select weekday(date_add(now(),interval 23 day));
select weekday(date_sub(curdate(),interval 33 day));
```

▶ DAYOFYEAR(날짜) : 일년을 기준으로 한 날짜까지의 날 수.

```
select dayofyear(20020724);     select dayofyear('20001231');
select dayofyear('2002-01-01'); select dayofyear(now());
select dayofyear(curdate());    select dayofyear(date_add(curdate(),interval 44 year));
select dayofyear(date_sub(now(),interval 25 month));
select dayofyear(date_add(now(),interval 55 day));
select dayofyear(date_sub(curdate(),interval 777 hour));
select dayofyear(date_add(now(),interval 999999 minute));
```

▶ WEEK(날짜) : 일년 중 몇 번째 주.

```
select week(now());   select week(date_sub(curdate(),interval 12 month));
```

▶ FROM_DAYS(날 수)

```
--00년 00월 00일부터 날 수 만큼 경과한 날의 날짜 출력.
   ※ 날 수는 366 이상을 입력 그 이하는 무조건 '0000-00-00' 으로 출력.
--또한 9999-12-31 [from_days(3652424)] 까지의 날짜가 출력가능
--따라서 날 수는 366 이상 3652424[3652499] 이하가 되어야 한다.
select from_days(3652424);   select from_days('3652499');
```

# 날짜 관련 함수

▶ TO_DAYS(날짜)

--00 년 00 월 00일 부터 날짜까지의 일자 수 출력. 정확한 날짜범위는 3652424 일 수

```
select to_days(now()) - to_days('1970-10-10');
```

응용 예제2) 살아 온 날수를 이용하여 자신의 나이를 만으로 구하기

```
select (to_days(now())-to_days('1970-10-10'))/365;
```

▶ DATE_FORMAT(날짜,'형식') : select date_format(now(),'%Y:%M:%p'); => 2001:May:PM

| 타입 | 기호 | 설명 | 기호 | 설명 |
|---|---|---|---|---|
| 년도 | %Y | 4자리 연도 | %y | 2자리 년도 |
| 월 | %M<br>%b | 긴 월 이름 (January, ...)<br>짧은 월 이름(Jan, ...) | %m<br>%c | 숫자의 월 (01...12)<br>숫자의 월 (1...12) |
| 요일 | %W | 긴 요일 이름 (Sunday, ...) | %a | 짧은 요일 이름 (Sun, ...) |
| 일 | %D<br>%w | 월 내에서 서수 형식의 일(1th, ...)<br>숫자의 요일 (0=Sunday, ...) | %d<br>%e<br>%j | 월 내의 일자 (01...31)<br>월 내의 일자 (1...31)<br>일년 중의 날수 (001...366) |
| 시 | %l<br>%h<br>%I | 12시간제의 시 (1...12)<br>12시간제의 시 (01...12)<br>12시간제의 시 (01...12) | %k<br>%H | 12시간제의 시 (0...23)<br>12시간제의 시 (00...23) |
| 분 | %i | 숫자의 분 (00...59) | | |
| 초 | %S | 숫자의 초 (00...59) | %s | 숫자의 초 (00...59) |
| 시간 | %r | 12시간제의 시간 (hh:mm:ss AM 또는 PM) | %T | 24시간제의 시간 (hh:mm:ss) |
| 주 | %U | 일요일을 기준으로 한 주 (0...52) | %u | 월요일을 기준으로 한 주 (0...52) |
| 기타 | %% | 문자 '%' | %p | AM 또는 PM |

# 논리관련함수

5) 논리관련함수

▶ IF(논리식,참일 때 값,거짓일 때 값)

Select pno,pname,pay,if(pay>=1500,'good','poor' as result from personal;

▶ IFNULL(값1,값2)

값1이 NULL 이면 값2로 대치하고 그렇지 않으면 값1을 출력

select pno, pname, pay, pay + ifnull(bonus,0) from personal;

6) 그밖의 함수

▶limit select pname from personal limit 5;

▶ select DATABASE() : 현재의 데이터베이스 이름을 출력한다.

▶ mysql5.* : select password('문자열') : 문자열을 암호화한다.

▶ mysql8.* : select sha('a');

▶ FORMAT(숫자,소수이하자리수) : 숫자를 #,###,###.## 형식으로 출력

--임의의 소수점자릿수를 생성한다./소숫점을 필요한 만큼 취한다.

--소숫점을 만들어 같은 길이로 불러와서 소숫점을 버리고 출력하는 등에 응용

select format(123,5); select format(123.12345600123,9); select format(123.123,-3);

※ 소숫점이하자리수가 0 이나 음수값은 해당이 안됨

연습

업무가 clerk이나 manager가 아닌 사원, 입사일이 1991년 이후이며 급여가 2000이사인 사원

업무가 manager이거나 sales, 사원의 이름 첫 글자가 A~S인 사원중에서 이른 순으로 정열

# 문자열 함수

- 종류
  - ✓ ASCII(문자), CHAR(숫자)
  - ✓ BIT_LENGTH(문자열), CHAR_LENGTH(문자열), LENGTH(문자열)
  - ✓ CONCAT(문자열1, 문자열2 …), CONCAT_WS(구분자, 문자열1, 문자열2,…): 문자열 결합 함수
  - ✓ ELT(위치, 문자열1, 문자열2), FIELD(찾을 문자열, 문자열1, 문자열2, ..), FIND_ IN_SET(찾을 문자열, 문자열 리스트), INSTR(기준 문자열, 부분 문자열), LOCATE(부 분 문자열, 기준 문자열)
    - ELT()는 위치 번째에 해당하는 문자열을 반환
    - FIELD()는 찾을 문자열의 위치를 찾아서 반환
    - FILED ()는 매치되는 문자열이 없으면 0을 반환
    - FIND_IN_SET()은 찾을 문자 열을 문자열 리스트에서 찾아서 위치를 반환하는데 문자열 리스트는 콤마(,)로 구분되어 있어야 하며 공백이 없어야 함
    - INSTR()은 기준 문자열에서 부분 문자열을 찾아서 그 시작 위치를 반환
    - LOCATE ()는 INSTR()과 동일하지만 파라미터의 순서가 반대로 되어 있음

# 문자열 함수

❖ 종류

- ✓ FORMAT(숫자, 소수점 자릿수): 숫자를 소수점 아래 자릿수까지 표현하는데 1000 단위마다 ,를 표시
- ✓ BIN(숫자), HEX(숫자), OCT(숫자): 2진수, 16진수, 8진수를 리턴
- ✓ INSERT(기준 문자열, 위치, 길이, 삽입할 문자열)
- ✓ LEFT(문자열, 길이), RIGHT(문자열, 길이)
- ✓ UPPER(문자열), LOWER(문자열)
- ✓ LPAD(문자열, 길이, 채울 문자열), RPAD(문자열, 길이, 채울 문자열)
- ✓ LTRIM(문자열), RTRIM(문자열)
- ✓ TRIM(문자열), TRIM(방향 자를 문자열 FROM 문자열): 방향 자를 문자열은 LEADING, BOTH, TRAILING
- ✓ REPEAT(문자열, 횟수)
- ✓ REPLACE(문자열, 원래 문자열, 바꿀 문자열)
- ✓ REVERSE(문자열)
- ✓ SPACE(길이): 길이 만큼의 공백 반환

# 날짜 함수

❖ 종류

- ✓ ADDDATE(날짜, 차이), SUBDATE(날짜, 차이): 날짜를 기준으로 차이를 더하거나 뺀 날짜를 구함
- ✓ ADDTIME(날짜/시간, 시간), SUBTIME(날짜/시간, 시간): 날짜/시간을 기준으로 시간을 더하거나 뺀 결과를 구함
- ✓ 현재 날짜 및 시간
  - CURRENT_DATE(), CURDATE() -> 현재 날짜
  - CURRENT_TIME(), CURTIME() -> 현재 시간
  - NOW(), LOCALTIME(), LOCALTIMESTAMP()
  - CURRENT_TIMESTAMP() -> 현재 날짜 및 시간
- ✓ YEAR(날짜), MONTH(날짜), DAY(날짜), HOUR(시간), MINUTE(시간), SECOND(시 간), MICROSECOND(시간)

# 날짜 함수

❖ 종류

✓ 특정 날짜 생성

- STR_TO_DATE(날짜 문자열, 서식 문자열)
- select STR_TO_DATE('1986-05-05 11:00:00', '%Y-%m-%d %H:%i:%S') ;

✓ 날짜와 기간형 데이터와 연산

- 날짜 데이터 와 INTERVAL 정수 단위 형태의 덧셈과 뺄셈 가능
- CURRENT_DATE() + INTERVAL 1 DAY : 현재 날짜에 하루를 추가

✓ 날짜 간의 뺄셈

- DATEDIFF(날짜 데이터, 날짜 데이터), TIMEDIFF(날짜 또는 시간, 날짜 또는 시간)
- select datediff(CURRENT_DATE(), STR_TO_DATE('1986-05-05', '%Y-%m-%d'));

# 시스템 정보 함수

- 종류
  - ✓ USER(), DATABASE()
  - ✓ FOUND_ROWS()
  - ✓ ROW_COUNT()
  - ✓ VERSION()
  - ✓ SLEEP(초)

# 기타 함수

- 윈도우 함수
  - ✓ Window Function는 행 과 행 사이의 관계를 쉽게 정의하기 위해서 제공되는 함수
  - ✓ 윈도우 함수를 잘 활용한다면 복잡한 SQL을 쉽게 활용할 수 있음
  - ✓ 윈도우 함수는 OVER절이 들어간 함수라고 보면 되는데 윈도우 함수와 함께 사용되는 집계 함수로는 AVG(), COUNT0, MAX(), MIN(), STDDEV(), SUM(), VARIANCE() 등이 있음
  - ✓ 윈도우 함수와 함께 사용되는 비집계 함수에는 CUME_DIST(), DENSE_RANK(), FIRST_VALUE(), LAG(), LAST_VALUE(), LEAD(), NTH_VALUE(), NTILE(), PERCENT_RANK(), RANK(), ROW_NUMBER() 등이 있음

# 기타 함수

❖ 윈도우 함수

✓ 순위 함수

- RANK(), NTILE(), DENSE_RANK(), ROW_NUMBER()
- 기본 형식

  <순위함수이름>( ) OVER(

  [PARTITION BY <partition_by_list>]

  ORDER BY <order_by_list>

- 일련 번호 형태로 순위 설정

```
SELECT ROW_NUMBER( ) OVER(ORDER BY birthyear ASC) '나이가 많은 순서', name, birthyear  FROM usertbl;

SELECT ROW_NUMBER( ) OVER(ORDER BY birthyear ASC, name ASC) '나이가 많은 순서', name, birthyear  FROM usertbl;

SELECT ROW_NUMBER( ) OVER(PARTITION BY addr ORDER BY birthyear ASC) '나이가 많은 순서', name, birthyear FROM usertbl;
```

# 기타 함수

❖ 윈도우 함수

✓ 순위 함수

- 동일한 값은 동일한 순위 설정

```
SELECT DENSE_RANK( ) OVER(ORDER BY birthyear ASC) '나이가 많은 순서', name, birthyear  FROM usertbl;
```

- 동일한 값은 동일한 순위의 경우 다음 순위 건너뛰기

```
SELECT RANK( ) OVER(ORDER BY birthyear ASC) '나이가 많은 순서', name, birthyear  FROM usertbl;
```

- N등분 하기

```
SELECT NTILE(5) OVER(ORDER BY birthyear ASC) '나이가 많은 순서', name, birthyear
FROM usertbl;
```

# 기타 함수

❖ 분석 함수

✓ CUME_DIST(), LEAD(), FIRST_VALUE(), LAG(), LAST_VALUE(), PERCENT_RANK() 등

- 다음 행과의 차이

```
SELECT name, addr, birthyear AS "태어난 해", birthyear -
    (LEAD(birthyear, 1) OVER(ORDER BY birthyear desc)) as "나이 차이"
FROM usertbl;
```

- 이전 행 과의 차이

```
SELECT name, addr, birthyear AS "태어난 해", birthyear -
    (LAG(birthyear, 1) OVER(ORDER BY birthyear desc)) as "나이 차이"
FROM usertbl;
```

# 기타 함수

❖ 분석 함수

✓ CUME_DIST(), LEAD(), FIRST_VALUE(), LAG(), LAST_VALUE(), PERCENT_RANK() 등

- 첫번째 행 과의 차이

```
SELECT name, addr, birthyear AS "태어난 해", birthyear -
    (FIRST_VALUE(birthyear) OVER(PARTITION BY addr ORDER BY
    birthyear desc)) as "나이 차이"
FROM usertbl;
```

- 누적합

```
SELECT name, addr, birthyear AS "태어난 해", CUME_DIST()
    OVER(PARTITION BY addr ORDER BY birthyear desc) * 100 AS "
    누적인원 백분율"
FROM usertbl;
```

# 기타 함수

## ❖ 피벗

✓ 피벗은 한 열에 포함된 여러 값을 출력하고 이를 여러 열로 변환하여 테이블 반환 식을 회전하고 필요하면 집계까지 수행하는 것

```
CREATE TABLE pivotTest
  (  uName CHAR(20),
     season CHAR(20),
     amount INT );

INSERT  INTO  pivotTest VALUES
     ('aespa' , '겨울',  10) , ('블랙핑크' , '여름',  15) , ('aespa' , '가을',  25) ,
('aespa' , '봄',    3) ,
     ('aespa' , '봄',    37) , ('블랙핑크' , '겨울',  40) , ('aespa' , '여름',
14) ,('aespa' , '겨울',  22) ,
     ('블랙핑크' , '여름',  64) ;

SELECT * FROM pivotTest;
```

# 기타 함수

❖ 피벗

✓ 피벗은 한 열에 포함된 여러 값을 출력하고 이를 여러 열로 변환하여 테이블 반환 식을 회전하고 필요하면 집계까지 수행하는 것

```
SELECT uName,
  SUM(IF(season='봄', amount, 0)) AS '봄',
  SUM(IF(season='여름', amount, 0)) AS '여름',
  SUM(IF(season='가을', amount, 0)) AS '가을',
  SUM(IF(season='겨울', amount, 0)) AS '겨울',
  SUM(amount) AS '합계'
 FROM pivotTest
GROUP BY uName ;
```

# 기타 함수

❖ JSON 출력

```
SELECT JSON_OBJECT('name', name, 'birthyear', birthyear) AS 'JSON 값'
FROM usertbl
WHERE birthyear >= 1980;
```

# Data BACKUP및 복구(mysql8.*)

1. 데이터 위치 보기

```
show variables like "secure_file_priv";
```

2. 데이터를 해당 위치로 변경

C:₩ProgramData₩MySQL₩MySQL Server 8.0₩Uploads

3. Load명령(tab)

```
load data infile
'C:/ProgramData/MySQL/MySQL Server 8.0/Uploads/emp.txt'
into table emp character set euckr;
```

4. Load명령(csv)

```
load data infile
'C:/ProgramData/MySQL/MySQL Server 8.0/Uploads/emp2.csv'
into table emp2 character set euckr fields terminated by ','
lines terminated by '₩r₩n';
```

# Data BACKUP및 복구(mysql5.*)

set global local_infile=1;

1) DataBase

]# mysqldump -u 계정 -h 서버 -p DB명 > 파일명

]# mysql-u 계정 -h 서버 -p DB명 < 파일명

2) Table

mysqldump -u root -pmysql test personal > a.txt

mysql -u root -pmysql test < a.txt

3) load

mysql> load data infile '파일네임' replace into table personal;

load data infile '파일네임' into table personal;

load data infile 'c:/gov/mysql/sawon.csv' into table sawon
CHARACTER SET euckr fields terminated by ',' lines terminated by '₩r₩n';

SHOW VARIABLES LIKE 'char%';

3) DB내용

/usr/local/mysql/var : db가 디렉토리별로 되어 있음

4) SQL계정삭제

mysql> delete from user where user = '계정';

권한 재적용 mysql db에서 권한 재적용할 때에 사용하는 명령어

# Data BACKUP및 복구(mysql8.*)

1. 데이터 위치 보기

```
show variables like "secure_file_priv";
```

2. 데이터를 해당 위치로 변경

C:₩ProgramData₩MySQL₩MySQL Server 8.0₩Uploads

3. Load명령(tab)

```
load data [local] infile
'C:/ProgramData/MySQL/MySQL Server 8.0/Uploads/emp.txt'
into table emp character set euckr;
```

4. Load명령(csv)

```
load data [local] infile
'C:/ProgramData/MySQL/MySQL Server 8.0/Uploads/emp2.csv'
into table emp2 character set euckr fields terminated by ','
lines terminated by '₩r₩n';
```