02. Structured Query Language

▼ Index

01. Data type
02. Structure
03. SQL

▼ 01. Data type

정수형 소수점이 없는 숫자 데이터

※ 바이트 : 컴퓨터의 저장공간 단위 중 하나

데이터 타입	바이트 수		표현 가능한 숫자 범위
TINYINT	1		-128 ~ 127
SMALLINT	2		-32,768 ~ 32,767
MEDIUMINT	3		약 -838백만 ~ 838백만
INT	4		약 -21억~ +21억
BIGINT	8		약 -900경 ~ +900경

실수형 소수점이 있는 숫자 데이터

데이터 타입	바이트 수		표현 가능한 숫자 범위
FLOAT	4		소수점 아래 7자리까지 표현
DOUBLE	8		소수점 아래 15자리까지 표현

문자형

※ 바이트 : 컴퓨터의 저장공간 단위 중 하나

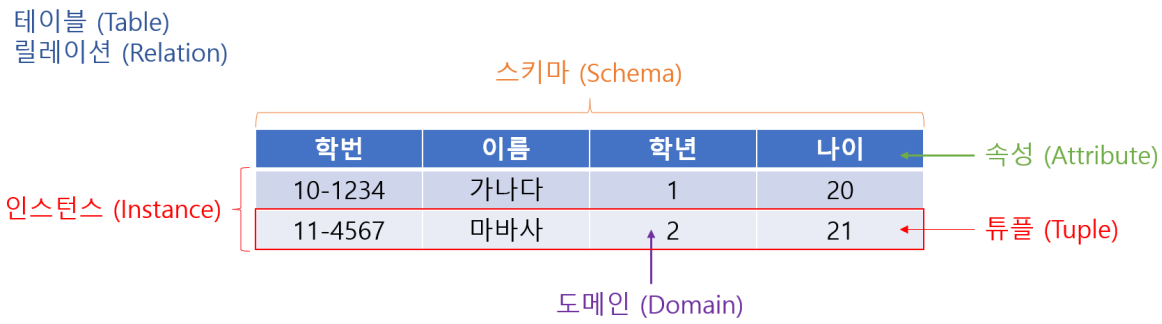
데이터 타입	최대 바이트 수	특징
CHAR(n)	255	n을 1부터 255까지 지정 가능, 지정 안 할 시 1 자동 입력 고정 길이로 문자열 저장.
VARCHAR(n)	65535	n을 1부터 65535까지 지정 가능, 지정 안 할 시 사용 불가 변동 길이로 문자열 저장.

데이터 타입	고정 바이트 수	특징
TINYTEXT	255	255 바이트의 문자열까지 표현 가능
TEXT	65535	65535 바이트의 문자열까지 표현 가능
MEDIUMTEXT	약 천 6백만	약 천 6백만 바이트의 문자열까지 표현 가능
LONGTEXT	약 42억	약 42억 바이트의 문자열까지 표현 가능

날짜형

데이터 타입	바이트 수	표현 가능 범위
DATE	3	0000-00-00 ~ 9999-12-31
DATETIME	3	0000-00-00 00:00:00 ~ 9999-12-31 23:59:59
TIME	4	-838:59:59 ~ 838:59:59
YEAR	1	1901 ~ 2155

▼ 02. Structure



- 속성 (Attribute) == 컬럼 (Column)

▼ 03. SQL

▼ DDL : Data Definition Language - 데이터 정의어

▼ CREATE : Database, Table 등 생성

▼ Database

- Query

```
CREATE DATABASE [데이터베이스명];
```

  - '데이터베이스명'으로 데이터베이스 생성

- Related features

```
SHOW DATABASES;
```

  - 현재 존재하는 모든 데이터베이스 목록

```
USE [데이터베이스명];
```

  - '데이터베이스명' 사용 시작

▼ Table

- Query

```
CREATE TABLE [테이블명] (
    [컬럼명] [데이터타입],
    [컬럼명] [데이터타입],
    ...
);
```

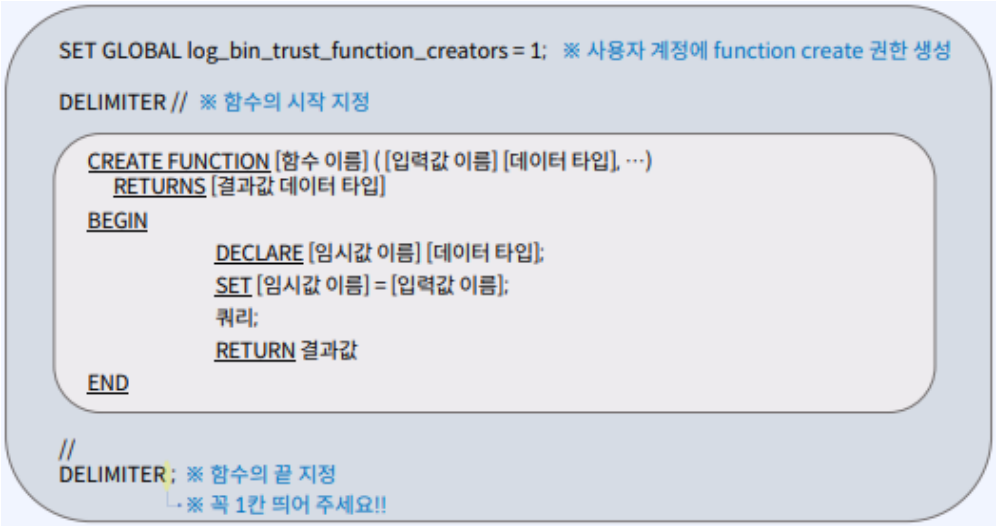
- Related features

```
SHOW TABLES;
```

  - 데이터베이스에 현재 존재하는 모든 테이블 목록

```
DESC [테이블명]
```

▼ Function




```
# 사용자 계정에 함수 생성 권한 설정
SET GLOBAL log_bin_trust_function_creators = 1;

# 함수 시작 지정
DELIMITER //
```

```
# 함수 생성
CREATE FUNCTION [함수명] ([입력값] [타입], ...)
        RETURNS [결과값 타입]
BEGIN
        DECLARE [변수명] [타입]; # 선언
        SET [변수명] = [입력값]; # 초기화
        Query...
        RETURN [결과값]
END

# 함수 종료 지정
//
DELIMITER ;


# Example
CREATE FUNCTION getAbility (attack INT, defense IN
        RETURNS INT
BEGIN
        DECLARE attVar INT;
        DECLARE defVar INT;
        DECLARE abilityVar INT;

        SET attVar = attack;
        SET defVar = defense;

        SELECT attVar + defVar INTO abilityVar;

        RETURN abilityVar;
END
```

▼ Constraints

제약 조건	의미
NOT NULL	이 컬럼에는 NULL 값을 저장할 수 없다.
UNIQUE	이 컬럼의 값들은 서로 다른 값을 가져야 한다.
DEFAULT	이 컬럼에 입력값이 없을 시 기본값이 설정된다.
PRIMARY KEY	이 컬럼은 테이블의 기본 키다. NOT NULL과 UNIQUE 특징을 모두 가진다.
FOREIGN KEY	이 컬럼은 테이블의 외래 키다. 이 컬럼은 다른 테이블의 특정 컬럼을 참조한다.

▼ Table

```
CREATE TABLE [테이블명] (
    [컬럼명] [데이터타입] [제약조건],
    [컬럼명] [데이터타입] [제약조건],
    ...
);
```

```
DROP DATABASE IF EXISTS pokemon;
CREATE DATABASE pokemon;
USE pokemon;
CREATE TABLE wildpokemon (
       number  INT PRIMARY KEY,
       name VARCHAR(20),
       type VARCHAR(10),
       height FLOAT,
       weight FLOAT,
       attack INT,
       defense INT,
       speed int
);
INSERT INTO wildpokemon
    (number, name, type, height, weight, attack
VALUES
    (10, 'caterpie'     , 'bug'     , 0.3, 2.9,
    (25, 'pikachu'      , 'electric', 0.4, 6, 5
    (26, 'raichu'       , 'electric', 0.8, 30,
    (133, 'eevee'       , 'normal'  , 0.3, 6.5,
    (152, 'chikoirita'  , 'grass'   , 0.9, 6.4,

CREATE TABLE mypokemon(
```

```
    number INT PRIMARY KEY,
    name VARCHAR(20) UNIQUE,
    type VARCHAR(10) NOT NULL,
    attack INT DEFAULT 0,
    defense INT DEFAULT 100,
    FOREIGN KEY (number) REFERENCES pokemon.wil
);
```

	Field	Type	Null	Key	Default	Extra
▶	number	int	NO	PRI	NULL	
	name	varchar(20)	YES	UNI	NULL	
	type	varchar(10)	NO		NULL	
	attack	int	YES		0	
	defense	int	YES		100	

▼ ALTER : Table 수정

▼ Name

```
ALTER TABLE [OLD_테이블명] RENAME [NEW_테이블명];
```

▼ Column

```
# 속성(Attribute) 추가
ALTER TABLE [테이블명] ADD COLUMN [속성명] [데이터타입]

# 속성 타입 변경
ALTER TABLE [테이블명] MODIFY COLUMN [속성명] [NEW_타입

# 속성명 & 타입 변경
ALTER TABLE [테이블명] CHANGE COLUMN [OLD_속성명] [NEW

# 속성 삭제
ALTER TABLE [테이블명] DROP COLUMN [속성명];
```

▼ DROP : Database, Table 등 삭제

▼ Database

```
DROP DATABASE [데이터베이스명];
```

```
# '존재한다면' 조건추가
DROP DATABASE IF EXISTS [데이터베이스명];
```

▼ Table

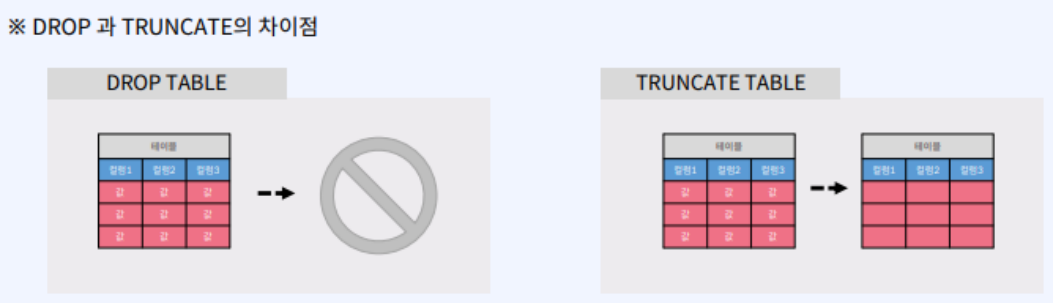
```
DROP TABLE [테이블명];

# '존재한다면' 조건추가
DROP TABLE IF EXISTS [테이블명];
```

▼ Function

```
DROP FUNCTION [함수명];
```

▼ TRUNCATE : Table 초기화

```
TRUNCATE TABLE [테이블명];
```



▼ DML : Data Manipulation Language - 데이터 조작어

▼ SELECT : 데이터 조회

```
SELECT * or (속성명, 속성명...) FROM ([데이터베이스명].)[테

# Case 01
SELECT * FROM 데이터베이스명.테이블명;

# Case 02
USING 데이터베이스명
SELECT * FROM 테이블명;
```

▼ Related features

▼ 출력 결과 형식 지정

```
# Alias : 출력 결과의 컬럼명을 새롭게 지정
SELECT 컬럼명 AS 별명 ...
FROM 데이터베이스명.테이블명;

# LIMIT : 가져올 데이터의 ROW 개수를 지정
SELECT 컬럼명 ...
FROM 데이터베이스명.테이블명
LIMIT 2; # 결과 인스턴스 출력 <= 2

# DISTINCT : 중복 데이터 제외
SELECT DISTINCT 컬럼명 ...
FROM 데이터베이스명.테이블명;
```

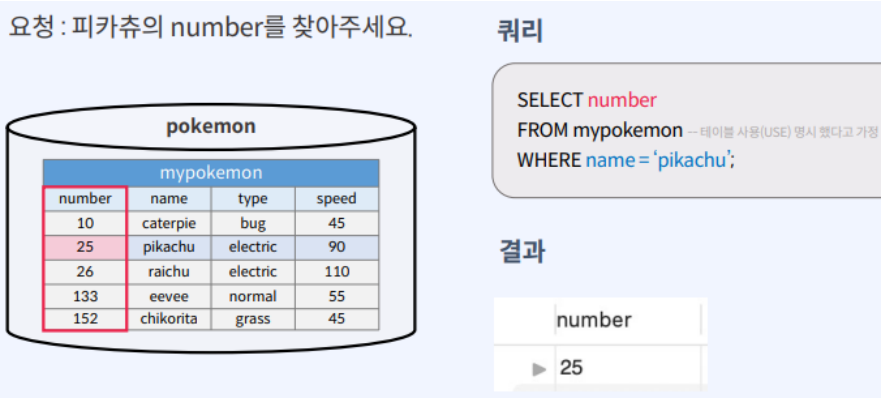
▼ WHERE

```
# 문법
SELECT [컬럼명]
FROM [테이블명]
WHERE 조건식;
```

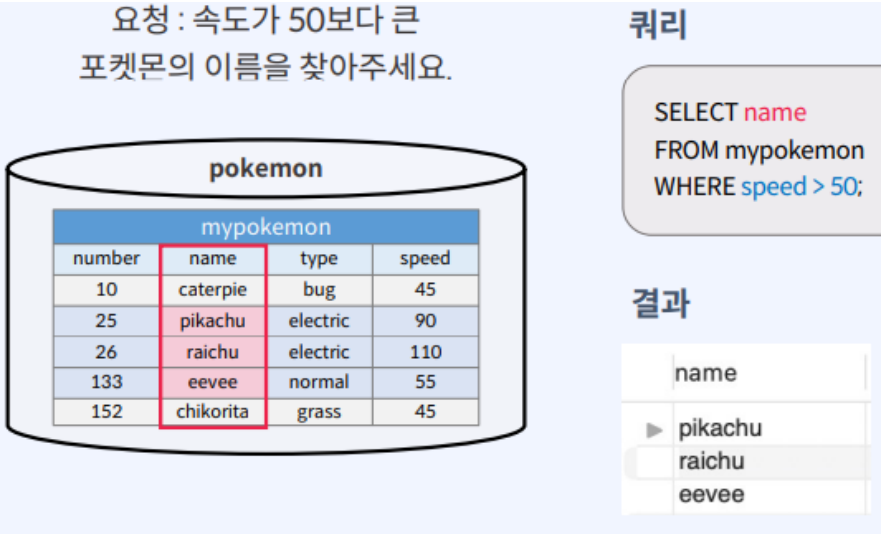
▼ 비교 연산자

연산자	활용	의미	예시
=	A = B	A와 B가 같다	1 = 1
!=	A != B	A와 B가 같지 않다	1 != 2
>	A > B	A가 B보다 크다	10 > 1
>=	A >= B	A가 B보다 크거나 같다	10 >= 10
<	A < B	A가 B보다 작다	10 < 100
<=	A <= B	A가 B보다 작거나 같다	10 <= 10

```
# =
SELECT number
FROM mypokemon
WHERE name='pikachu';
```




```
# >
SELECT name
FROM mypokemon
WHERE speed > 50;
```




```
# !=
SELECT name
FROM mypokemon
WHERE type != 'electric';
```

요청 : 전기 타입이 아닌
포켓몬의 이름을 찾아주세요.

pokemon

mypokemon

number	name	type	speed
10	caterpie	bug	45
25	pikachu	electric	90
26	raichu	electric	110
133	eevee	normal	55
152	chikorita	grass	45

쿼리

```
SELECT name
FROM mypokemon
WHERE type != 'electric';
```

결과

name
caterpie
eevee
chikoirita

▼ 논리 연산자

연산자	활용	의미
AND	A AND B	A와 B 모두 True이면 True
OR	A OR B	A와 B 둘 중 하나만 True이면 True
NOT	NOT A	A가 아니면 True

```
# AND, &
SELECT name
FROM mypokemon
WHERE speed <= 100 AND type = 'electric';
```

요청 : 속도가 100 이하인
전기 타입 포켓몬의 이름을 찾아주세요.

pokemon

mypokemon

number	name	type	speed
10	caterpie	bug	45
25	pikachu	electric	90
26	raichu	electric	110
133	eevee	normal	55
152	chikorita	grass	45

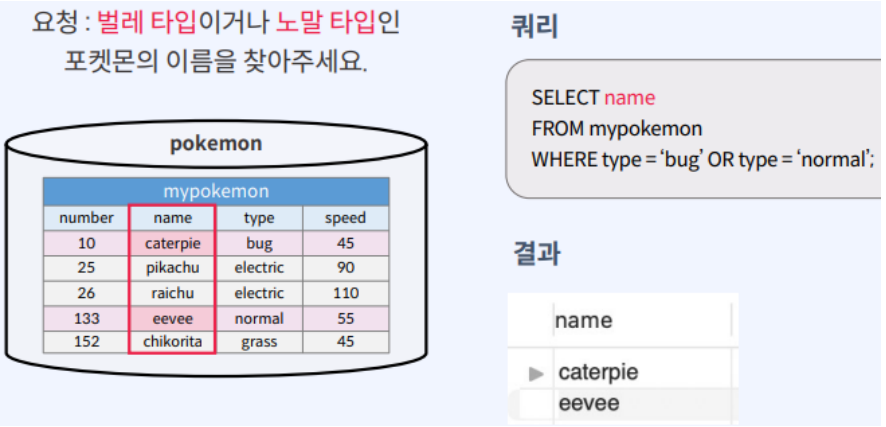
쿼리

```
SELECT name
FROM mypokemon
WHERE speed <= 100 AND type = 'electric';
```

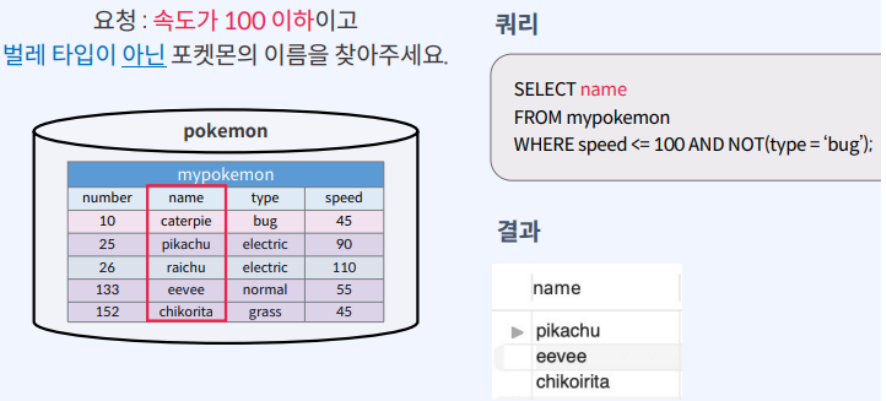
결과

name
pikachu

```
# OR, |
SELECT name
from mypokemon
WHERE type = 'bug' OR type = 'normal';
```




```
# NOT, !
SELECT name
FROM mypokemon;
WHERE speed <= 100
            AND NOT(type='bug');
            or
            AND type != 'bug';
```



▼ 범위

```
# BETWEEN : 범위
SELECT name
FROM mypokemon
WHERE speed BETWEEN 50 AND 100;
( WHERE 50 <= speed and speed <= 100; )
```

요청 : 속도가 50과 100 사이인
포켓몬의 이름을 찾아주세요.

pokemon

mypokemon

number	name	type	speed
10	caterpie	bug	45
25	pikachu	electric	90
26	raichu	electric	110
133	eevee	normal	55
152	chikorita	grass	45

쿼리

```
SELECT name
FROM mypokemon
WHERE speed BETWEEN 50 AND 100;
```

결과

name
▶ pikachu
eevee

```
# IN : 목록 내 포함
SELECT name
FROM mypokemon
WHERE type IN ('bug', 'normal');
( type = 'bug' or type = 'normal'; )
```

요청 : 벌레 타입이거나 노말 타입인
포켓몬의 이름을 찾아주세요.

pokemon

mypokemon

number	name	type	speed
10	caterpie	bug	45
25	pikachu	electric	90
26	raichu	electric	110
133	eevee	normal	55
152	chikorita	grass	45

쿼리

```
SELECT name
FROM mypokemon
WHERE type IN ('bug', 'normal');
```

결과

name
▶ caterpie
eevee

▼ 와일드카드

와일드카드	의미
%	0개 이상의 문자
_	1개의 문자

- '%e' e로 끝나는 문자열
e, ee, eevee, apple, pineapple

- 'e%' e로 시작하는 문자열
e, ee, eevee, eric

- '%e%' e가 포함된 문자열
e, ee, eevee, apple, pineapple, aespa

- '_e' e로 끝나고 e 앞에 1개 의 문자가 있는 문자열
ae, ee, ce

- '%_e' e로 끝나고 e 앞에 1개 이상의 문자가 있는 문자열
ee, eevee, apple, pineapple

- '%_e_%' e를 포함하고 e 앞 뒤로 각각 1개 이상의 문자가 있는 문자열
eevee, aespa

```
# LIKE
SELECT [컬럼명]
FROM [테이블명]
WHERE [컬럼명] LIKE [검색할 문자열];
```

요청 : 이름이 'chu'로 끝나는
포켓몬의 이름을 찾아주세요.

pokemon

mypokemon

number	name	type	speed
10	caterpie	bug	45
25	pikachu	electric	90
26	raichu	electric	110
133	eevee	normal	55
152	chikorita	grass	45

쿼리

```
SELECT name
FROM mypokemon
WHERE name LIKE '%chu';
```

결과

name
▶ pikachu
raichu

요청 : 이름에 'a'가 포함되는
포켓몬의 이름을 찾아주세요.

pokemon

mypokemon

number	name	type	speed
10	caterpie	bug	45
25	pikachu	electric	90
26	raichu	electric	110
133	eevee	normal	55
152	chikorita	grass	45

쿼리

```
SELECT name
FROM mypokemon
WHERE name LIKE '%a%';
```

결과

name
caterpie
pikachu
raichu
chikoirita

▼ NULL

쿼리

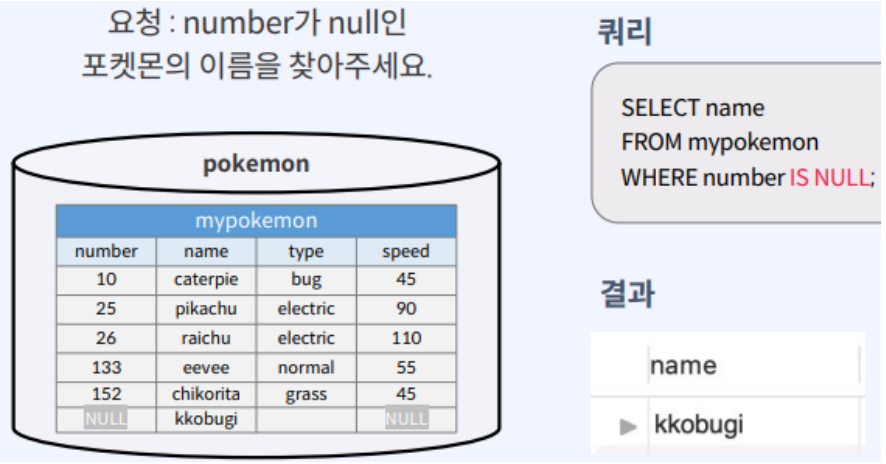
```
INSERT INTO mypokemon (name, type)
VALUES ('kkobugi', '');

SELECT *
FROM mypokemon;
```

결과

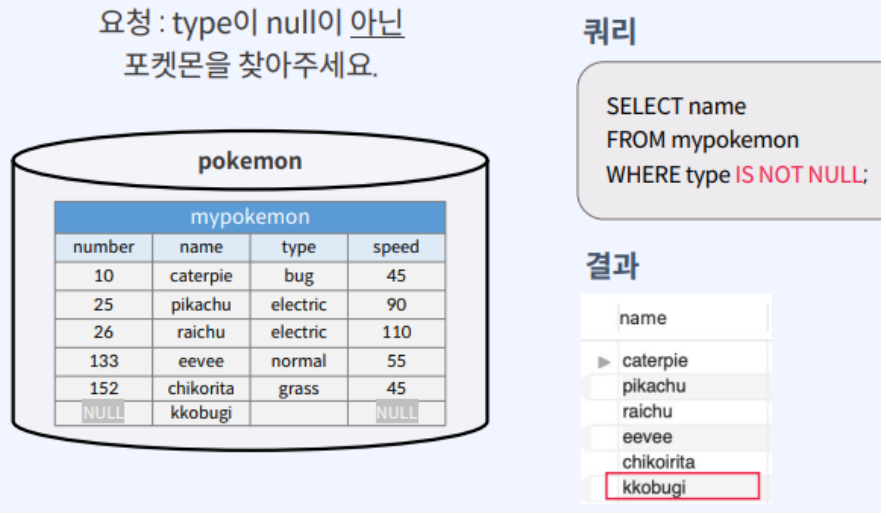
number	name	type	height	weight	attack	defense	speed
10	caterpie	bug	0.3	2.9	30	35	45
25	pikachu	electric	0.4	6	55	40	90
26	raichu	electric	0.8	30	90	55	110
133	eevee	normal	0.3	6.5	55	50	55
152	chikoirita	grass	0.9	6.4	49	65	45
NULL	kkobugi		NULL	NULL	NULL	NULL	NULL

공백

```
# IS NULL
SELECT [컬럼명]
FROM [테이블명]
WHERE [컬럼명] IS NULL;
```



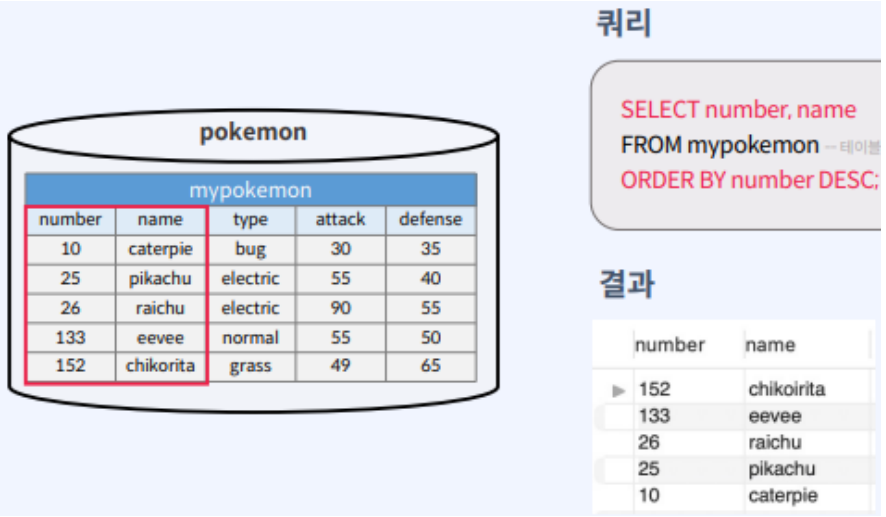

```
# IS NOT NULL
SELECT [컬럼명]
FROM [테이블명]
WHERE [컬럼명] IS NOT NULL;
```

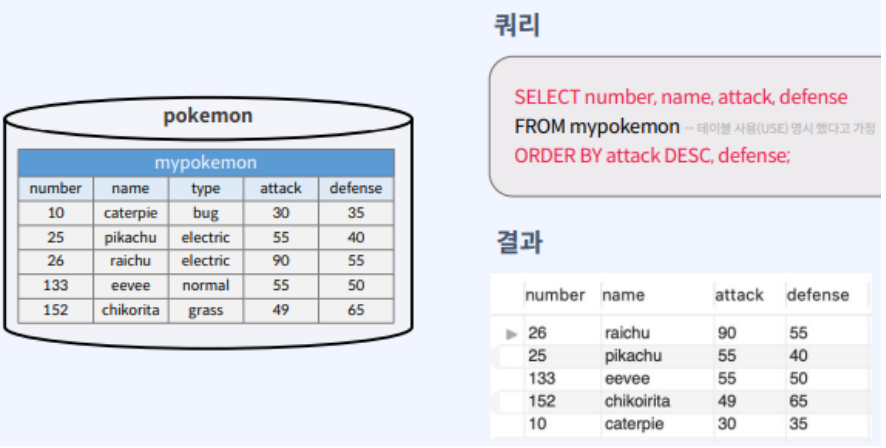


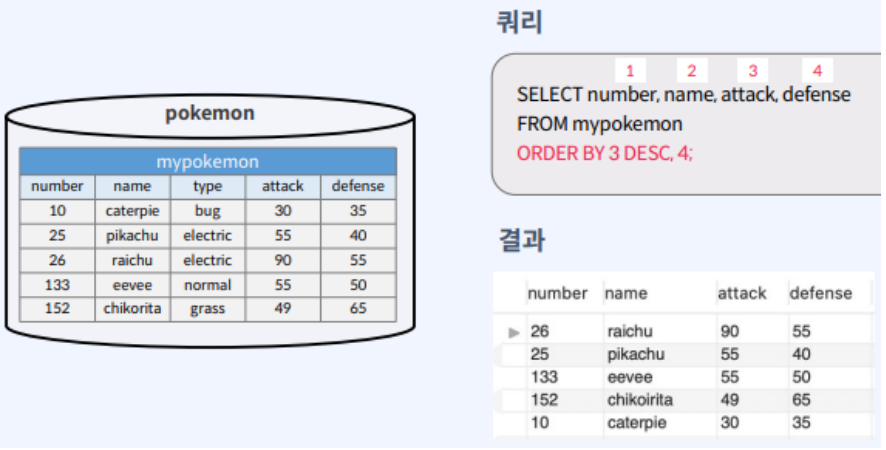
▼ ORDER BY

▼ 줄 세우기

```
SELECT [컬럼명]
FROM [테이블명]
WHERE [조건식]
ORDER BY [컬럼명] ASC or DESC;
```



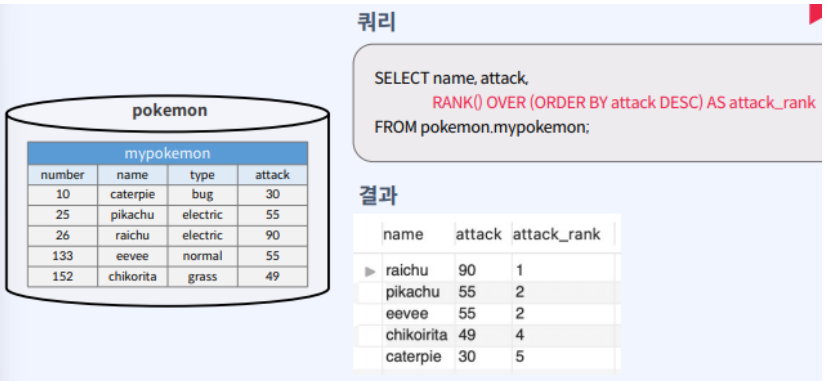




▼ 줄 세우고 순위 만들기

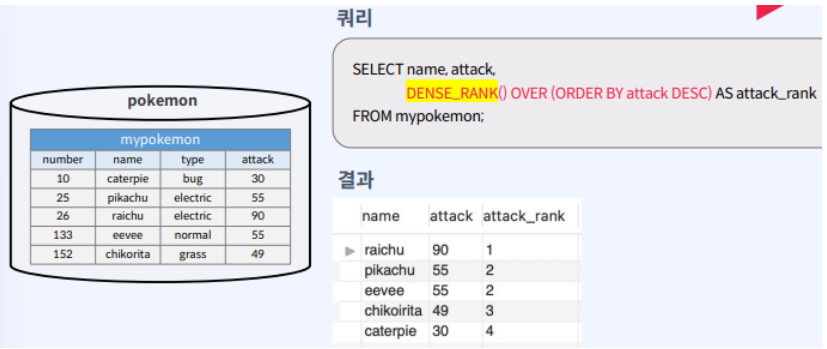
▼ RANK

```
SELECT [컬럼명], ..., RANK() OVER (ORDER BY
FROM [테이블명]
WHERE [조건식];
```



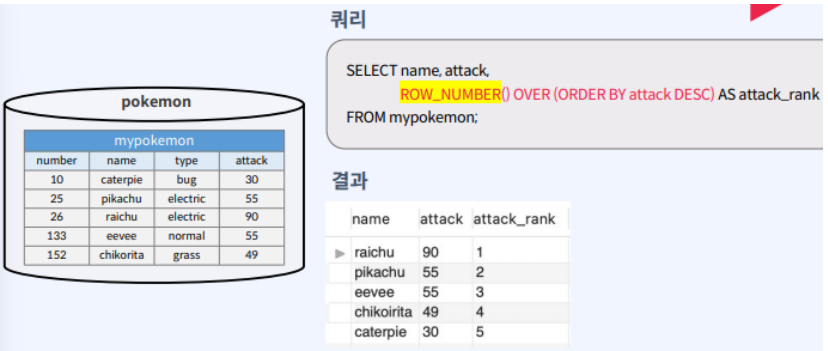
▼ DENSE_RANK

```
SELECT [컬럼명], ..., DENSE_RANK() OVER (OR
FROM [테이블명]
WHERE [조건식];
```



▼ ROW_NUMBER

```
SELECT [컬럼명], ..., ROW_NUMBER() OVER (OR
FROM [테이블명]
WHERE [조건식];
```



▼ 비교

```
SELECT name, attack,
        RANK() OVER (ORDER BY attack DESC) AS rank_rank,
        DENSE_RANK() OVER (ORDER BY attack DESC) AS rank_dense_rank,
        ROW_NUMBER() OVER (ORDER BY attack DESC) AS rank_row_number
FROM mypokemon;
```

결과

name	attack	rank_rank	rank_dense_rank	rank_row_number
raichu	90	1	1	1
pikachu	55	2	2	2
eevee	55	2	2	3
chikoirita	49	4	3	4
caterpie	30	5	4	5

RANK	공동 순위가 있으면 다음 순서로 건너 뜀
DENSE_RANK	공동 순위가 있어도 다음 순위를 뛰어 넘지 않음
ROW_NUMBER	공동 순위를 무시함

▼ Function

▼ 문자형 데이터 함수

함수	활용 예시	설명
LOCATE	LOCATE("A", "ABC")	"ABC"에서 "A"는 몇 번째에 위치해 있는지 검색해 위치 반환
SUBSTRING	SUBSTRING("ABC", 2)	"ABC"에서 2번째 문자부터 반환
RIGHT	RIGHT("ABC", 1)	"ABC"에서 오른쪽에서 1번째 문자까지 반환
LEFT	LEFT("ABC", 1)	"ABC"에서 왼쪽에서 1번째 문자까지 반환
UPPER	UPPER("abc")	"abc"를 대문자로 바꿔 반환
LOWER	LOWER("ABC")	"ABC"를 소문자로 바꿔 반환
LENGTH	LENGTH("ABC")	"ABC"의 글자 수를 반환
CONCAT	CONCAT("ABC", "DEF")	"ABC" 문자열과 "CDF" 문자열을 합쳐 반환
REPLACE	REPLACE("ABC", "A", "Z")	"ABC"의 "A"를 "Z"로 바꿔 반환

▼ Example

bts_music

butter

part	lyric
1	Smooth like butter
2	Like a criminal undercover
3	Gon' pop like trouble
4	Breakin' into your heart like that
5	Cool shade stunner

▼ LOCATE

```
SELECT part, LOCATE('i', lyric)
FROM bts_music.butter;
```

butter	
part	lyric
1	Smooth like butter
2	Like a criminal undercover
3	Gon' pop like trouble
4	Breakin' into your heart like that
5	Cool shade stunner 없음

part	LOCATE('i', lyric)
1	9
2	2
3	11
4	6
5	0

▼ SUBSTRING

```
SELECT part, SUBSTRING(lyric, 3)
FROM bts_music.butter;
```

butter	
part	lyric
1	Smooth like butter
2	Like a criminal undercover
3	Gon' pop like trouble
4	Breakin' into your heart like that
5	Cool shade stunner

part	SUBSTRING(lyric, 3)
1	ooth like butter
2	ke a criminal undercover
3	n' pop like trouble
4	eakin' into your heart like that
5	ol shade stunner

▼ RIGHT, LEFT

```
SELECT part, RIGHT(lyric, 3), LEFT(lyr
FROM btn_music.butter;
```

butter	
part	lyric
1	Smooth like butter
2	Like a criminal undercover
3	Gon' pop like trouble
4	Breakin' into your heart like that
5	Cool shade stunner

part	RIGHT(lyric, 3)	LEFT(lyric, 3)
1	ter	Smo
2	ver	Lik
3	ble	Gon
4	hat	Bre
5	ner	Coo

▼ UPPER, LOWER

```
SELECT part, UPPER(lyric), LOWER(lyric
FROM btn_music.butter;
```

part	UPPER(lyric)	LOWER(lyric)
1	SMOOTH LIKE BUTTER	smooth like butter
2	LIKE A CRIMINAL UNDERCOVER	like a criminal undercover
3	GON' POP LIKE TROUBLE	gon' pop like trouble
4	BREAKIN' INTO YOUR HEART LIKE THAT	breakin' into your heart like that
5	COOL SHADE STUNNER	cool shade stunner

▼ LENGTH

```
SELECT part, LENGTH(lyric)
FROM btn_music.butter;
```

part	LENGTH(lyric)
1	18
2	26
3	21
4	34
5	18

▼ CONCAT

```
SELECT part,
        CONCAT(LEFT(lyric, 1), RIGHT(l
FROM btn_music.butter;
```

part	first_last
1	Sr
2	Lr
3	Ge
4	Bt
5	Cr

▼ REPLACE

```
SELECT part, REPLACE(lyric, '', '_')
FROM btn_music.butter;
```

part	REPLACE(lyric, ' ', '_')
1	Smooth_like_butter
2	Like_a_criminal_undercover
3	Gon'_pop_like_trouble
4	Breakin'_into_your_heart_like_that
5	Cool_shade_stunner

▼ 숫자형 데이터 함수

함수	활용	설명
ABS	ABS(숫자)	숫자의 절댓값 반환
CEILING	CEILING(숫자)	숫자를 정수로 올림해서 반환
FLOOR	FLOOR(숫자)	숫자를 정수로 내림해서 반환
ROUND	ROUND(숫자, 자릿수)	숫자를 소수점 자릿수까지 반올림해서 반환
TRUNCATE	TRUNCATE(숫자, 자릿수)	숫자를 소수점 자릿수까지 버림해서 반환
POWER	POWER(숫자A, 숫자B)	숫자A의 숫자B 제곱 반환
MOD	MOD(숫자A, 숫자B)	숫자A를 숫자B로 나눈 나머지 반환

▼ Example

pokemon

mypokemon

number	name	height	weight	friendship
10	caterpie	0.3	2.9	-1.455
25	pikachu	0.4	6	124.78
26	raichu	0.8	30	30.289
125	electabuzz	1.1	30	-10.67
133	eevee	0.3	6.5	15.988
137	porygon	0.8	36.5	-0.245
152	chikorita	0.9	6.4	67.164
153	bayleef	1.2	15.8	9.756
172	pichu	0.3	2	872.1
470	leafeon	1	25.5	3.42

▼ ABS

```
SELECT name, friendship, ABS(friendshi
FROM pokemon.mypokemon;
```

name	friendship	ABS(friendship)
caterpie	-1.455	1.4550000429153442
pikachu	124.78	124.77999877929688
raichu	30.289	30.288999557495117
electabuzz	-10.67	10.670000076293945
eevee	15.988	15.98799991607666
porygon	-0.245	0.24500000476837158
chikoirita	67.164	67.16400146484375
bayleef	9.756	9.755999565124512
pichu	872.1	872.0999755859375
leafeon	3.42	3.4200000762939453

▼ CEILING, FLOOR

```
SELECT name, friendship, CEILING(frien
FROM pokemon.mypokemon;
```

name	friendship	CEILING(friendship)	FLOOR(friendship)
caterpie	-1.455	-1	-2
pikachu	124.78	125	124
raichu	30.289	31	30
electabuzz	-10.67	-10	-11
eevee	15.988	16	15
porygon	-0.245	-0	-1
chikoirita	67.164	68	67
bayleef	9.756	10	9
pichu	872.1	873	872
leafeon	3.42	4	3

▼ ROUND, TRUNCATE

```
SELECT name, friendship, ROUND(friends
    TRUNCATE(friendship, 1)
FROM pokemon.mypokemon;
```

name	friendship	ROUND(friendship, 1)	TRUNCATE(friendship, 1)
caterpie	-1.455	-1.5	-1.4
pikachu	124.78	124.8	124.7
raichu	30.289	30.3	30.2
electabuzz	-10.67	-10.7	-10.6
eevee	15.988	16	15.9
porygon	-0.245	-0.2	-0.2
chikoirita	67.164	67.2	67.1
bayleef	9.756	9.8	9.7
pichu	872.1	872.1	872
leafeon	3.42	3.4	3.4

▼ POWER

```
SELECT name, number, POWER(number, 2)
FROM pokemon.mypokemon;
```

name	number	POWER(number, 2)
caterpie	10	100
pikachu	25	625
raichu	26	676
electabuzz	125	15625
eevee	133	17689
porygon	137	18769
chikoirita	152	23104
bayleef	153	23409
pichu	172	29584
leafeon	470	220900

▼ MOD

```
SELECT name, number, MOD(number, 2)
FROM pokemon.mypokemon;
```

name	number	MOD(number, 2)
caterpie	10	0
pikachu	25	1
raichu	26	0
electabuzz	125	1
eevee	133	1
porygon	137	1
chikoirita	152	0
bayleef	153	1
pichu	172	0
leafeon	470	0

▼ 날짜형 데이터 함수

함수	활용	설명
NOW	NOW()	현재 날짜와 시간 반환
CURRENT_DATE	CURRENT_DATE()	현재 날짜 반환
CURRENT_TIME	CURRENT_TIME()	현재 시간 반환
YEAR	YEAR(날짜)	날짜의 연도 반환
MONTH	MONTH(날짜)	날짜의 월 반환
MONTHNAME	MOTHNAME(날짜)	날짜의 월을 영어로 반환
DAYNAME	DAYNAME(날짜)	날짜의 요일을 영어로 반환
DAYOFMONTH	DAYOFMONTH(날짜)	날짜의 일 반환
DAYOFWEEK	DAYOFWEEK(날짜)	날짜의 요일을 숫자로 반환
WEEK	WEEK(날짜)	날짜가 해당 연도에 몇 번째 주인지 반환

함수	활용	설명
HOUR	HOUR(시간)	시간의 시 반환
MINUTE	MINUTE(시간)	시간의 분 반환
SECOND	SECOND(시간)	시간의 초 반환
DATE_FORMAT	DATEFORMAT(날짜/시간, 형식)	날짜/시간의 형식을 형식으로 바꿔 반환
DATEDIFF	DATEDIFF(날짜1, 날짜2)	날짜1과 날짜2의 차이 반환 (날짜1 – 날짜2)
TIMEDIFF	TIMEDIFF(시간,1 시간2)	시간1과 시간2의 차이 반환 (시간1 – 시간2)

표현	설명	표현	설명
%a	요일을 영문 약어으로 표현 (Sun..Sat)	%p	AM 또는 PM를 표현
%b	월을 영문 약어로 표현 (Jan..Dec)	%r	시간을 AM 또는 PM과 함께 표현 (hh:mm:ss AM or PM)
%c	월을 숫자로 표현(0..12)	%S	초를 표현 (00..59)
%D	일을 숫자와 영문으로 표현 (0th, 1st, 2nd, 3rd, ...)	%s	초를 표현 (00..59)
%d	일을 항상 숫자 두 글자로 표현 (00..31)	%T	시간을 24시간으로 표현 (hh:mm:ss)
%e	일을 표현 (0..31)	%U	1년 중 몇 번째 주인지를 표현 (한 주가 일요일부터 시작, 00..53)
%f	마이크로 초를 표현 (000000..999999)	%u	1년 중 몇 번째 주인지를 표현 (한 주가 월요일부터 시작, 00..53)
%H	시를 24시간으로 표현 (00..23)	%V	1년 중 몇 번째 주인지를 표현 (한 주가 일요일부터 시작, 00..53)
%h	시를 12시간으로 항상 숫자 두 글자로 표현 (01..12)	%v	1년 중 몇 번째 주인지를 표현 (한 주가 월요일부터 시작, 00..53)
%I	시를 12시간으로 표현 (1..12)	%W	요일을 영문으로 표현 (Sunday..Saturday)
%i	분을 표현 (00..59)	%w	요일을 숫자로 표현 (0=Sunday..6=Saturday)
%j	1년 중 몇 번 째 날인지 표현 (001..366)	%X	주가 속한 연도를 네글자로 표현 (한 주가 일요일부터 시작)
%k	시를 24시간으로 표현 (0..23)	%x	주가 속한 연도를 네글자로 표현 (한 주가 월요일부터 시작)
%l	시를 12시간으로 표현 (1..12)	%Y	연도를 네 글자로 표현 (0000.. 9999)
%M	월을 영문으로 표현 (January..December)	%y	연도를 두 글자로 표현 (00, 99)
%m	월을 표현 (00..12)	%%	글자 '%'.

▼ Example

▼ NOW, CURRENT_DATE, CURRENT_TIME

```
SELECT NOW(), CURRENT_DATE(), CURRENT_
```

	NOW()	CURRENT_DATE()	CURRENT_TIME()
▶	2023-12-15 17:42:19	2023-12-15	17:42:19

▼ YEAR, MONTH, MONTHNAME

```
SELECT NOW(), YEAR(NOW()), MONTH(NOW()
```

	NOW()	YEAR(NOW())	MONTH(NOW())	MONTHNAME(NOW())
▶	2023-12-15 17:43:59	2023	12	December

▼ DAYNAME, DAYOFMONTH, DAYOFWEEK, WEEK

```
SELECT NOW(), DAYNAME(NOW()), DAYOFMON
```

	NOW()	DAYNAME(NOW())	DAYOFMONTH(NOW())	DAYOFWEEK(NOW())	WEEK(NOW())
▶	2023-12-15 17:45:25	Friday	15	6	50

▼ HOUR, MINUT, SECOND

```
SELECT NOW(), HOUR(NOW()), MINUTE(NOW(
```

NOW()	HOUR(NOW())	MINUTE(NOW())	SECOND(NOW())
2023-12-15 17:46:36	17	46	36

▼ DATE_FORMAT

```
SELECT NOW(), DATE_FORMAT(NOW(), '%Y년
```

NOW()	formatted_date
2023-12-15 17:48:33	2023년 12월 15일 17시 48분 33초

▼ DATEDIFF, TIMEDIFF

```
SELECT DATEDIFF('2022-01-01 00:00:00',
        TIMEDIFF('2022-01-01 00:00:00'
```

	DATE_DIFF	TIME_DIFF
▶	7	156:00:00

▼ GROUP BY

```
SELECT [컬럼명], ..., [그룹함수]
FROM [테이블명]
WHERE 조건식
GROUP BY [컬럼명]
```

▼ Example

pokemon

mypokemon

number	name	type	height	weight
10	caterpie	bug	0.3	2.9
25	pikachu	electric	0.4	6
26	raichu	electric	0.8	30
125	electabuzz	electric	1.1	30
133	eevee	normal	0.3	6.5
137	porygon	normal	0.8	36.5
152	chikorita	grass	0.9	6.4
153	bayleef	grass	1.2	15.8
172	pichu	electric	0.3	2
470	leafeon	grass	1	25.5

```
SELECT type
FROM pokemon.mypokemon
GROUP BY type;
```

type
bug
electric
normal
grass

내부 구조

type	number	name	height	weight
bug	10	caterpie	0.3	2.9
electric	25	pikachu	0.4	6
	26	raichu	0.8	30
	125	electabuzz	1.1	30
	172	pichu	0.3	2
normal	133	eevee	0.3	6.5
	137	porygon	0.8	36.5
grass	152	chikorita	0.9	6.4
	153	bayleef	1.2	15.8
	470	leafeon	1	25.5

그룹

▼ HAVING

```
SELECT [컬럼명], ..., [그룹함수]
FROM [테이블명]
WHERE 조건식
GROUP BY [컬럼명]
HAVING 조건식;
```

▼ Group Function

▼ COUNT

```
SELECT [컬럼명], ..., COUNT([컬럼명])
FROM [테이블명]
WHERE 조건식
```

```
GROUP BY [컬럼명]
HAVING 조건식;
```

▼ SUM

```
SELECT [컬럼명], ..., SUM([컬럼명])
FROM [테이블명]
WHERE 조건식
GROUP BY [컬럼명]
HAVING 조건식;
```

▼ AVG

```
SELECT [컬럼명], ..., AVG([컬럼명])
FROM [테이블명]
WHERE 조건식
GROUP BY [컬럼명]
HAVING 조건식;
```

▼ MIN

```
SELECT [컬럼명], ..., MIN([컬럼명])
FROM [테이블명]
WHERE 조건식
GROUP BY [컬럼명]
HAVING 조건식;
```

▼ MAX

```
SELECT [컬럼명], ..., MAX([컬럼명])
FROM [테이블명]
WHERE 조건식
GROUP BY [컬럼명]
HAVING 조건식;
```

▼ Example

pokemon

mypokemon				
number	name	type	height	weight
10	caterpie	bug	0.3	2.9
25	pikachu	electric	0.4	6
26	raichu	electric	0.8	30
125	electabuzz	electric	1.1	30
133	eevee	normal	0.3	6.5
137	porygon	normal	0.8	36.5
152	chikorita	grass	0.9	6.4
153	bayleef	grass	1.2	15.8
172	pichu	electric	0.3	2
470	leafeon	grass	1	25.5

▼ SELECT 절

```
SELECT type, COUNT(*), COUNT(1), AVG(h
FROM pokemon.mypokemon
GROUP BY type;
```

결과

type	COUNT(*)	COUNT(1)	AVG(height)	MAX(weight)
bug	1	1	0.30000001192092896	2.9
electric	4	4	0.6500000134110451	30
normal	2	2	0.550000011920929	36.5
grass	3	3	1.0333333412806194	25.5

쿼리

```
SELECT type, COUNT(*), COUNT(1), AVG(height), MAX(weight)
FROM pokemon.mypokemon
GROUP BY type;
```

내부 구조

type	number	name	height	weight
bug	10	caterpie	0.3	2.9
electric	25	pikachu	0.4	6
	26	raichu	0.8	30
	125	electabuzz	1.1	30
	172	pichu	0.3	2
normal	133	eevee	0.3	6.5
	137	porygon	0.8	36.5
grass	152	chikorita	0.9	6.4
	153	bayleef	1.2	15.8
	470	leafeon	1	25.5

그룹

쿼리

```
SELECT type, COUNT(*), COUNT(1), AVG(height), MAX(weight)
FROM pokemon.mypokemon
GROUP BY type;
```

내부 구조

4개

type	number	name	height	weight
bug	10	caterpie	0.3	2.9
electric	25	pikachu	0.4	6
	26	raichu	0.8	30
	125	electabuzz	1.1	30
	172	pichu	0.3	2
normal	133	eevee	0.3	6.5
	137	porygon	0.8	36.5
grass	152	chikorita	0.9	6.4
	153	bayleef	1.2	15.8
	470	leafeon	1	25.5

쿼리

```
SELECT type, COUNT(*), COUNT(1), AVG(height), MAX(weight)
FROM pokemon.mypokemon
GROUP BY type;
```

내부 구조

type		number	name	height	weight
bug	1	10	caterpie	0.3	2.9
electric	1	25	pikachu	0.4	6
	1	26	raichu	0.8	30
	1	125	electabuzz	1.1	30
	1	172	pichu	0.3	2
normal	1	133	eevee	0.3	6.5
	1	137	porygon	0.8	36.5
grass	1	152	chikorita	0.9	6.4
	1	153	bayleef	1.2	15.8
	1	470	leafeon	1	25.5

쿼리

```
SELECT type, COUNT(*), COUNT(1), AVG(height), MAX(weight)
FROM pokemon.mypokemon
GROUP BY type;
```

내부 구조

type	number	name	height	weight
bug	10	caterpie	0.3	2.9
electric	25	pikachu	0.4	6
	26	raichu	0.8	30
	125	electabuzz	1.1	30
	172	pichu	0.3	2
normal	133	eevee	0.3	6.5
	137	porygon	0.8	36.5
grass	152	chikorita	0.9	6.4
	153	bayleef	1.2	15.8
	470	leafeon	1	25.5

쿼리

```
SELECT type, COUNT(*), COUNT(1), AVG(height), MAX(weight)
FROM pokemon.mypokemon
GROUP BY type;
```

내부 구조

type	number	name	height	weight
bug	10	caterpie	0.3	2.9
electric	25	pikachu	0.4	6
	26	raichu	0.8	30
	125	electabuzz	1.1	30
	172	pichu	0.3	2
normal	133	eevee	0.3	6.5
	137	porygon	0.8	36.5
grass	152	chikorita	0.9	6.4
	153	bayleef	1.2	15.8
	470	leafeon	1	25.5

▼ HAVING 절

```
SELECT type, COUNT(*), COUNT(1), AVG(h
FROM pokemon.mypokemon
GROUP BY type;
HAVING COUNT(1)=2;
```

결과

	type	COUNT(*)	COUNT(1)	AVG(height)	MAX(weight)
▶	normal	2	2	0.550000011920929	36.5

쿼리

```
SELECT type, COUNT(*), COUNT(1), AVG(height), MAX(weight)
FROM pokemon.mypokemon
GROUP BY type
HAVING COUNT(1) = 2;
```

COUNT(1) = 1
➔ 조건식 = False

내부 구조

type	number	name	height	weight
bug	10	caterpie	0.3	2.9
electric	25	pikachu	0.4	6
	26	raichu	0.8	30
	125	electabuzz	1.1	30
	172	pichu	0.3	2
normal	133	eevee	0.3	6.5
	137	porygon	0.8	36.5
grass	152	chikorita	0.9	6.4
	153	bayleef	1.2	15.8
	470	leafeon	1	25.5

쿼리

```
SELECT type, COUNT(*), COUNT(1), AVG(height), MAX(weight)
FROM pokemon.mypokemon
GROUP BY type
HAVING COUNT(1) = 2;
```

COUNT(1) = 4
➔ 조건식 = False

내부 구조

type	number	name	height	weight
bug	10	caterpie	0.3	2.9
electric	25	pikachu	0.4	6
	26	raichu	0.8	30
	125	electabuzz	1.1	30
	172	pichu	0.3	2
normal	133	eevee	0.3	6.5
	137	porygon	0.8	36.5
grass	152	chikorita	0.9	6.4
	153	bayleef	1.2	15.8
	470	leafeon	1	25.5

쿼리

```
SELECT type, COUNT(*), COUNT(1), AVG(height), MAX(weight)
FROM pokemon.mypokemon
GROUP BY type
HAVING COUNT(1) = 2;
```

COUNT(1) = 2
➔ 조건식 = TRUE

내부 구조

type	number	name	height	weight
bug	10	caterpie	0.3	2.9
electric	25	pikachu	0.4	6
	26	raichu	0.8	30
	125	electabuzz	1.1	30
	172	pichu	0.3	2
normal	133	eevee	0.3	6.5
	137	porygon	0.8	36.5
grass	152	chikorita	0.9	6.4
	153	bayleef	1.2	15.8
	470	leafeon	1	25.5

쿼리

```
SELECT type, COUNT(*), COUNT(1), AVG(height), MAX(weight)
FROM pokemon.mypokemon
GROUP BY type
HAVING COUNT(1) = 2;
```

COUNT(1) = 3
➔ 조건식 = False

내부 구조

type	number	name	height	weight
bug	10	caterpie	0.3	2.9
electric	25	pikachu	0.4	6
	26	raichu	0.8	30
	125	electabuzz	1.1	30
	172	pichu	0.3	2
normal	133	eevee	0.3	6.5
	137	porygon	0.8	36.5
grass	152	chikorita	0.9	6.4
	153	bayleef	1.2	15.8
	470	leafeon	1	25.5

▼ Conditional Expression

```
# IF
SELECT name, if(attack >= 60, 'strong', 'weak')
from pokemon.mypokemon;

# IS NULL
```

```
SELCT name, IFNULL(name, 'unknown') AS full_nam
FROM pokemon.mypokemon;
```

```
# SWITCH

-- Format 01
CASE
        WHEN 조건식1 THEN 결과값1
        WHEN 조건식2 THEN 결과값2
        ELSE 결과값3
END
-- Example
SELECT name,
CASE
        WHEN attack >= 100 THEN 'very strong'
        WHEN attack >= 60 THEN 'strong'
        ELSE 'week'
END AS attack_class
FROM pokemon.mypokemon;

-- Format 02
CASE [컬럼명]
        WHEN 조건식1 THEN 결과값1
        WHEN 조건식2 THEN 결과값2
        ELSE 결과값3
END
-- Example
SELECT name, type
CASE type
        WHEN 'bug' THEN 'grass'
        WHEN 'electric' THEN 'water'
        WHEN 'grass' THEN 'bug'
END AS rival_type
FROM pokemon.mypokemon;
```

▼ 6가지 핵심 Query 키워드

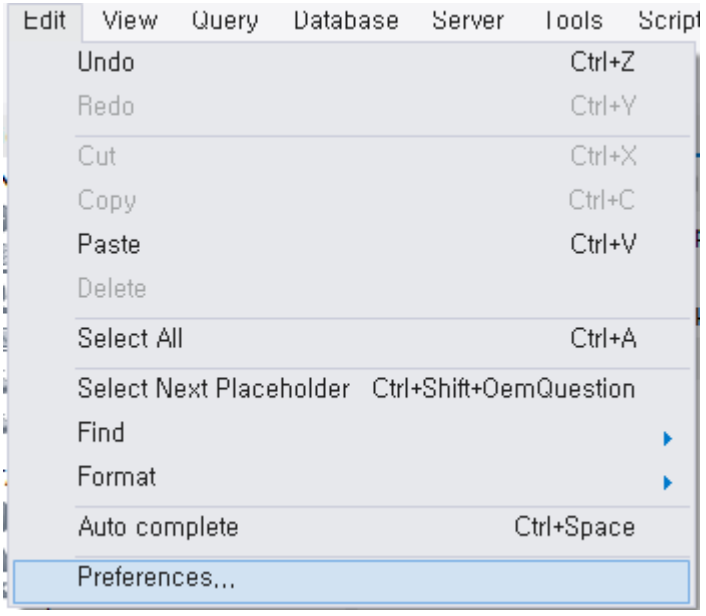
키워드	문법	작성 순서	실행 순서
SELECT	SELECT [컬럼 이름]	1	5
FROM	FROM [테이블 이름]	2	1
WHERE	WHERE 조건식	3	2
GROUP BY	GROUP BY [컬럼 이름]	4	3
HAVING	HAVING 조건식	5	4
ORDER BY	ORDER BY [컬럼 이름]	6	6

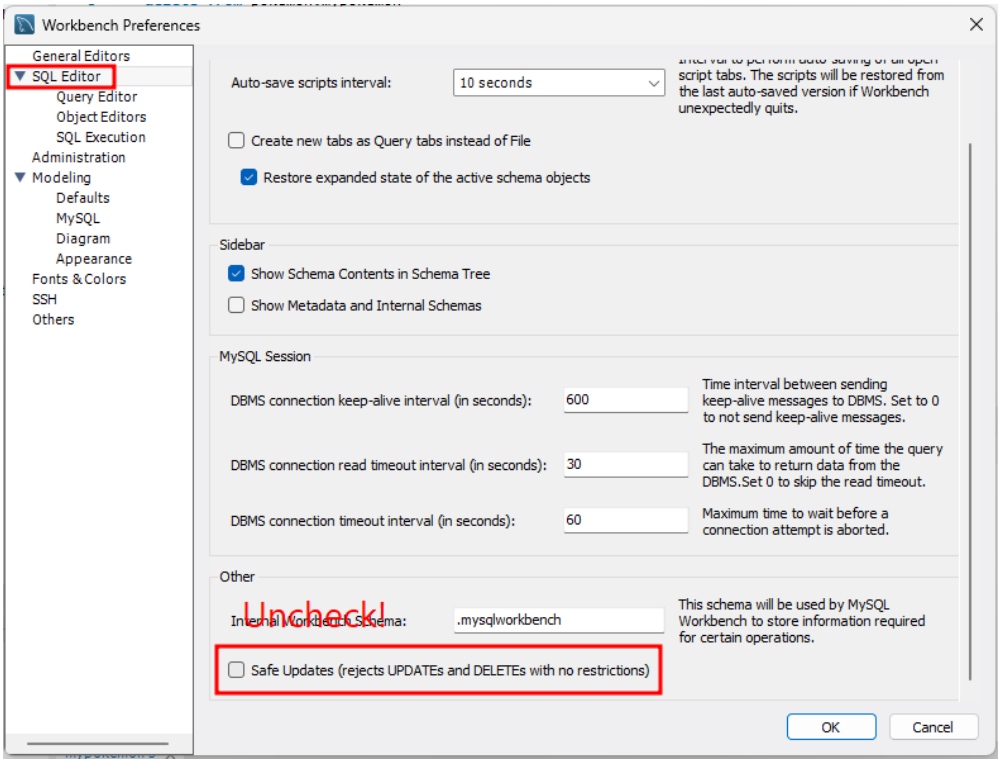
▼ Table Modify

* 테이블 수정 시 안전모드 변경 필요

```
Error Code: 1175. You are using safe update mode and
To disable safe mode, toggle the option in Preference
```

▼ 영구 해제





▼ 일시 해제

```
set sql_safe_updates=0;
```

▼ INSERT : 데이터 삽입

```
INSERT INTO [테이블명] (속성1, 속성2, 속성3...)
VALUES  (값1, 값2, 값3...),
                (값1, 값2, 값3...);
```

▼ UPDATE : 데이터 수정

```
UPDATE [테이블명]
SET [속성명] = [NEW_값]
WHERE [조건식];
```

▼ Example

number	name	type	height	weight	attack	defense	speed
10	caterpie	bug	0.3	2.9	30	35	45
25	pikachu	electric	0.4	6	55	40	90
26	raichu	electric	0.8	30	90	55	110
133	eevee	normal	0.3	6.5	55	50	55
152	chikoirita	grass	0.9	6.4	49	65	45

```
--set sql_safe_updates=0; #수정 불능 모드 해제

update pokemon.mypokemon
set type = 'normal'
    , number = 153
where name = 'chikoirita';
```

	number	name	type	height	weight	attack	defense	speed
▶	10	caterpie	bug	0.3	2.9	30	35	45
	25	pikachu	electric	0.4	6	55	40	90
	26	raichu	electric	0.8	30	90	55	110
	133	eevee	normal	0.3	6.5	55	50	55
	153	chikoirita	normal	0.9	6.4	49	65	45

▼ DELETE : 데이터 삭제

```
DELETE FROM [테이블명]
WHERE [조건식];
```

▼ Example

	number	name	type	height	weight	attack	defense	speed
▶	10	caterpie	bug	0.3	2.9	30	35	45
	25	pikachu	electric	0.4	6	55	40	90
	26	raichu	electric	0.8	30	90	55	110
	133	eevee	normal	0.3	6.5	55	50	55
	152	chikoirita	grass	0.9	6.4	49	65	45

```
--set sql_safe_updates=0; #수정 불능 모드 해제

delete from pokemon.mypokemon
where attack > 50;
```

	number	name	type	height	weight	attack	defense	speed
▶	10	caterpie	bug	0.3	2.9	30	35	45
	152	chikoirita	grass	0.9	6.4	49	65	45

▼ DCL : Datas Control Language - 데이터 제어어

▼ User

```
# MySQL 기본 데이터베이스인 mysql 데이터베이스 선택
USE mysql

# 사용자 목록 조회
SELECT user, host from user;
```

user	host
mysql.infoschema	localhost
mysql.session	localhost
mysql.sys	localhost
root	localhost

```
# 사용자 생성
CREATE USER [사용자명]@[IP];

# 사용자 생성 및 패스워드 설정
CREATE USER [사용자명]@[IP] IDENTIFIED BY '[패스워드]';

# 사용자 삭제
DROP USER [사용자명];
```

```
# 사용자 생성
CREATE USER new@localhost IDENTIFIED BY 'abc123';
SELECT user, host from user;
DROP USER new@localhost;
```

user	host
mysql.infoschema	localhost
mysql.session	localhost
mysql.sys	localhost
new	localhost
root	localhost

▼ Authority

```
# 권한 부여
CRANT [권한] ON [데이터베이스명].[테이블명] TO [사용자명]@[IP
```

```
# 권한 확인
SHOW GRANTS FOR [사용자명]@[IP];

# 권한 삭제
REVOKE [권한] ON [데이터베이스명].[테이블명] FROM [사용자명]@

# 권한 적용
FLUSH PRIVILEGES;
```

```
SHOW GRANTS FOR root@localhost;
```

Grants for root@localhost
GRANT SELECT, INSERT, UPDATE, DELETE, CREATE, DROP, RELOAD, SHUTDOWN, PROCESS, FILE, REFERENCES, INDEX, ALTER, SHOW DAT...
GRANT APPLICATION_PASSWORD_ADMIN,AUDIT_ABORT_EXEMPT,AUDIT_ADMIN,AUTHENTICATION_POLICY_ADMIN,BACKUP_ADMIN,BINLO...
GRANT PROXY ON ``@`` TO `root`@`localhost` WITH GRANT OPTION

```
SHOW GRANTS FOR new@localhost;
```

Grants for new@localhost
GRANT USAGE ON *.* TO `new`@`localhost`

```
# User에게 해당 DB.TABLE의 대한 모든 권한 부여
GRANT ALL PRIVILEGES ON pokemon.wildpokemon TO new@lo
# REVOKE ALL PRIVILEGES ON pokemon.wildpokemon FROM n
```

Grants for new@localhost
GRANT USAGE ON *.* TO `new`@`localhost`
GRANT ALL PRIVILEGES ON `pokemon`.`wildpokemon` TO `new`@`localhost`

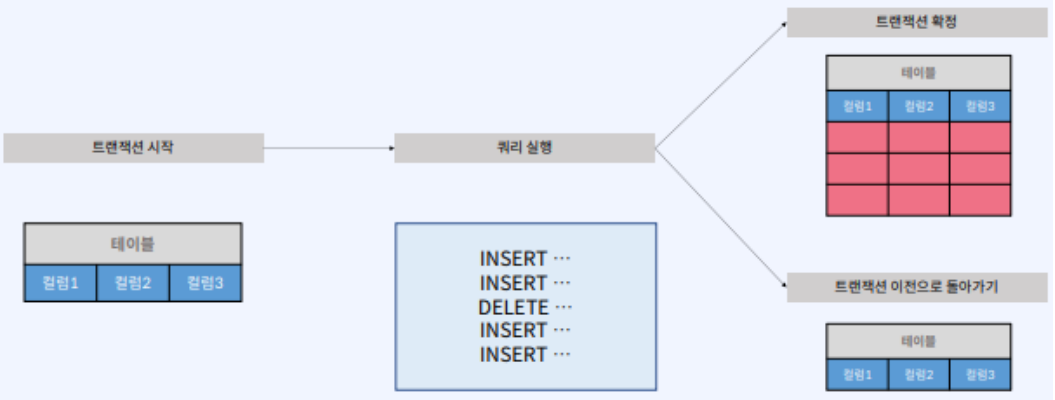
```
# User에게 모든 데이터베이스, 모든 테이블에 대한 권한 부여
GRANT ALL PRIVILEGES ON *.* TO new@localhost;
# REVOKE ALL PRIVILEGES ON *.* FROM new@localhost;
```

Grants for new@localhost
GRANT SELECT, INSERT, UPDATE, DELETE, CREATE, DROP, RELOAD, SHUTDOWN, PROCESS, FILE, REFERENCES, IN...
GRANT APPLICATION_PASSWORD_ADMIN,AUDIT_ABORT_EXEMPT,AUDIT_ADMIN,AUTHENTICATION_POLICY_ADMIN...

▼ TCL : Transaction Control Language - 트랜젝션 제어어

▼ Transaction

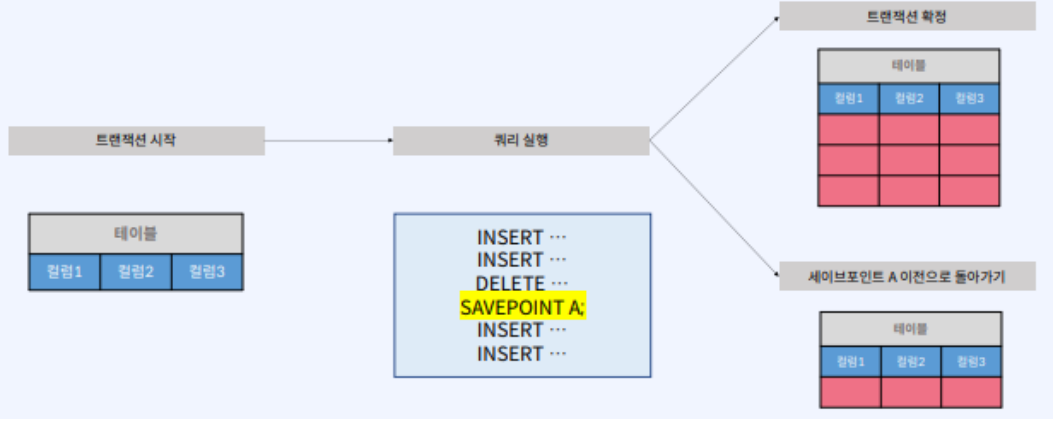
- 데이터베이스의 데이터 상태를 변경하는 작업 묶음




```
#트랜잭션 시작하기
START TRANSACTION;

#트랜잭션 확정하기
COMMIT;

#트랜잭션 이전으로 돌아가기
ROLLBACK
```

▼ SAVEPOINT

- 트랜잭션 내 특정한 저장 지점



```
#세이브포인트 만들기
SAVEPOINT [세이브포인트 이름];

#세이브포인트로 돌아가기
ROLLBACK TO [세이브포인트 이름]
```