person이란 table 데이터로 확인해보자.

mysql> desc person;

+-------+-------------+------+-----+---------+-------+

| Field | Type | Null | Key | Default | Extra |

+-------+-------------+------+-----+---------+-------+

| id | int | YES | | NULL | |

| name | varchar(16) | YES | | NULL | |

| age | int | YES | | NULL | |

| sex | char(1) | YES | | NULL | |

+-------+-------------+------+-----+---------+-------+

4 rows in set (0.00 sec)

위의 person table에는 2명의 데이터가 입력되어 있다.

**SELECT**

select 구문은 table 데이터를 조회하는 구문이다.  
기본 조회 방법은 아래와 같다.

select \* from table\_name;

'\*'은 모든 컬럼을 조회

mysql> select \* from person;

+------+---------+------+------+

| id | name | age | sex |

+------+---------+------+------+

| 1 | marry | 20 | F |

| 2 | charles | 30 | M |

+------+---------+------+------+

2 rows in set (0.00 sec)

select column1, column2.. from table\_name;

위와 같이 모든 데이터가 아닌, 이름과 나이만 조회해보자.

mysql> select name, age from person;

+---------+------+

| name | age |

+---------+------+

| marry | 20 |

| charles | 30 |

+---------+------+

2 rows in set (0.00 sec)

**WHERE**

where는 SQL문에 조건을 추가하는 구문이다.

select column1... from table\_name where condition;

person에서 조회할 때 '여성'인 조건을 만족하는 데이터만 불러와보자.

mysql> select \*

-> from person

-> where sex='F';

+------+-------+------+------+

| id | name | age | sex |

+------+-------+------+------+

| 1 | marry | 20 | F |

+------+-------+------+------+

1 row in set (0.00 sec)

**UPDATE**

update는 table 데이터를 '수정'할 때 사용이 된다.

update table\_name set column1=value1... where condition;

'marry'의 나이는 22세인데 20세로 잘못 입력이 되었다.  
20세에서 22세로 나이를 바꿔보자.

mysql> update person # 수정할 table 데이터 이름

-> set age=23 # set 다음 : 수정할 컬럼과 내용 입력

-> where name='marry'; # 'marry'라는 이름만(where 구문이 없으면 모든 나이가 23세로 바뀐다.)

Query OK, 1 row affected (0.00 sec)

Rows matched: 1 Changed: 1 Warnings: 0

수정이 잘 이루어졌는지 확인해보자.

mysql> select \* from person;

+------+---------+------+------+

| id | name | age | sex |

+------+---------+------+------+

| 1 | marry | 23 | F |

| 2 | charles | 30 | M |

+------+---------+------+------+

2 rows in set (0.00 sec)

**DELETE**

데이터를 삭제할 때 사용이 된다.

delete from table\_name where condition;

'marry'라는 데이터를 삭제해보자.

mysql> delete from person where name='marry';

Query OK, 1 row affected (0.00 sec)

삭제가 이루어졌는지 확인해보자.

mysql> select \* from person;

+------+---------+------+------+

| id | name | age | sex |

+------+---------+------+------+

| 2 | charles | 30 | M |

+------+---------+------+------+

1 row in set (0.00 sec)

**table 생성**

* not null : NaN값을 허용하지 않는 컬럼 생성 시 사용
* auto\_increment : 숫자형 데이터일 때, not null이라면 자동으로 오름차순으로 순서 부여.
* primary key : 데이터에는 하나의 'primary key'만 존재할 수 있으며, 해당 컬럼이 기본 키로 설정된다.  
  primary key(기본 키)는 다른 항목과 중복되어 나타날 수 없는 단일 값(unique)를 가진다. 그리고 null값을 가지지 못하기에 table를 생성 시, not null 설정해야 한다. 또한 하나 이상의 컬럼이 그룹화 되어 기본 키로 사용될 수 있다.  
  이러한 primary key는 다른 데이터들과 관계를 가지며 통합할 때 사용이 된다.

'celeb'이란 table을 생성하여 실습을 진행해보자.

mysql> create table celeb

-> (

-> id int not null auto\_increment primary key,

-> name varchar(32) not null default '',

-> birthday date,

-> age int,

-> sex char(1),

-> job\_title varchar(32),

-> agency varchar(32)

-> );

Query OK, 0 rows affected (0.02 sec)

생성된 table를 확인해보자.

mysql> desc celeb;

+-----------+-------------+------+-----+---------+----------------+

| Field | Type | Null | Key | Default | Extra |

+-----------+-------------+------+-----+---------+----------------+

| id | int | NO | PRI | NULL | auto\_increment |

| name | varchar(32) | NO | | | |

| birthday | date | YES | | NULL | |

| age | int | YES | | NULL | |

| sex | char(1) | YES | | NULL | |

| job\_title | varchar(32) | YES | | NULL | |

| agency | varchar(32) | YES | | NULL | |

+-----------+-------------+------+-----+---------+----------------+

7 rows in set (0.00 sec)

그리고 celeb에 임의로 셀럽들의 데이터를 입력하고 실습을 진행해보자.

mysql> select \* from celeb;

+----+-----------+------------+------+------+-------------------------+--------------------------+

| ID | NAME | BIRTHDAY | AGE | SEX | JOB\_TITLE | AGENCY |

+----+-----------+------------+------+------+-------------------------+--------------------------+

| 1 | 아이유 | 1993-05-16 | 29 | F | 가수, 텔런트 | EDAM엔터테이먼트 |

| 2 | 이미주 | 1994-09-23 | 28 | F | 가수 | 울림엔터테이먼트 |

| 3 | 송강 | 1994-04-23 | 28 | M | 텔런트 | 나무엑터스 |

| 4 | 강동원 | 1981-01-18 | 41 | M | 영화배우, 텔런트 | YG엔터테이먼트 |

| 5 | 유재석 | 1972-08-14 | 50 | M | MC, 개그맨 | 안테나 |

| 6 | 차승원 | 1970-06-07 | 48 | M | 영화배우, 모델 | YG엔터테이먼트 |

| 7 | 이수현 | 1999-05-04 | 23 | F | 가수 | YG엔터테이먼트 |

+----+-----------+------------+------+------+-------------------------+--------------------------+

7 rows in set (0.00 sec)

# ORDER BY

select 구문에서 데이터를 특정 컬럼을 기준으로 오름차순 혹은 내림차순 정렬하여 조회할 때 사용된다.

* asc(ascending) : 오름차순 정렬
* desc(desending) : 내림차순 정렬

asc

celeb table에서 이름과 나이를 '나이'순으로 조회해보자.(default = asc)

mysql> select age, name

-> from celeb

-> order by age asc;

+------+-----------+

| age | name |

+------+-----------+

| 23 | 이수현 |

| 28 | 이미주 |

| 28 | 송강 |

| 29 | 아이유 |

| 41 | 강동원 |

| 48 | 차승원 |

| 50 | 유재석 |

+------+-----------+

7 rows in set (0.00 sec)

desc

celeb table에서 이름과 나이를 '나이' 역순으로 조회해보자.

mysql> select age, name from celeb

-> order by age desc;

+------+-----------+

| age | name |

+------+-----------+

| 50 | 유재석 |

| 48 | 차승원 |

| 41 | 강동원 |

| 29 | 아이유 |

| 28 | 이미주 |

| 28 | 송강 |

| 23 | 이수현 |

+------+-----------+

7 rows in set (0.00 sec)

asc, desc

celeb table에서 이름과 나이를 '나이' 역순으로 정렬한 뒤, 이름 순으로 조회해보자.

mysql> select age, name

-> from celeb

-> order by age desc, name asc;

+------+-----------+

| age | name |

+------+-----------+

| 50 | 유재석 |

| 48 | 차승원 |

| 41 | 강동원 |

| 29 | 아이유 |

| 28 | 송강 |

| 28 | 이미주 |

| 23 | 이수현 |

+------+-----------+

7 rows in set (0.00 sec)

아래의 가상의 celeb table로 실습을 진행해보자.

mysql> desc celeb;

+-----------+-------------+------+-----+---------+----------------+

| Field | Type | Null | Key | Default | Extra |

+-----------+-------------+------+-----+---------+----------------+

| ID | int | NO | PRI | NULL | auto\_increment |

| NAME | varchar(32) | NO | | | |

| BIRTHDAY | date | YES | | NULL | |

| AGE | int | YES | | NULL | |

| SEX | char(1) | YES | | NULL | |

| JOB\_TITLE | varchar(32) | YES | | NULL | |

| AGENCY | varchar(32) | YES | | NULL | |

+-----------+-------------+------+-----+---------+----------------+

7 rows in set (0.00 sec)

mysql> select \* from celeb;

+----+-----------+------------+------+------+-------------------------+--------------------------+

| ID | NAME | BIRTHDAY | AGE | SEX | JOB\_TITLE | AGENCY |

+----+-----------+------------+------+------+-------------------------+--------------------------+

| 1 | 아이유 | 1993-05-16 | 29 | F | 가수, 텔런트 | EDAM엔터테이먼트 |

| 2 | 이미주 | 1994-09-23 | 28 | F | 가수 | 울림엔터테이먼트 |

| 3 | 송강 | 1994-04-23 | 28 | M | 텔런트 | 나무엑터스 |

| 4 | 강동원 | 1981-01-18 | 41 | M | 영화배우, 텔런트 | YG엔터테이먼트 |

| 5 | 유재석 | 1972-08-14 | 50 | M | MC, 개그맨 | 안테나 |

| 6 | 차승원 | 1970-06-07 | 48 | M | 영화배우, 모델 | YG엔터테이먼트 |

| 7 | 이수현 | 1999-05-04 | 23 | F | 가수 | YG엔터테이먼트 |

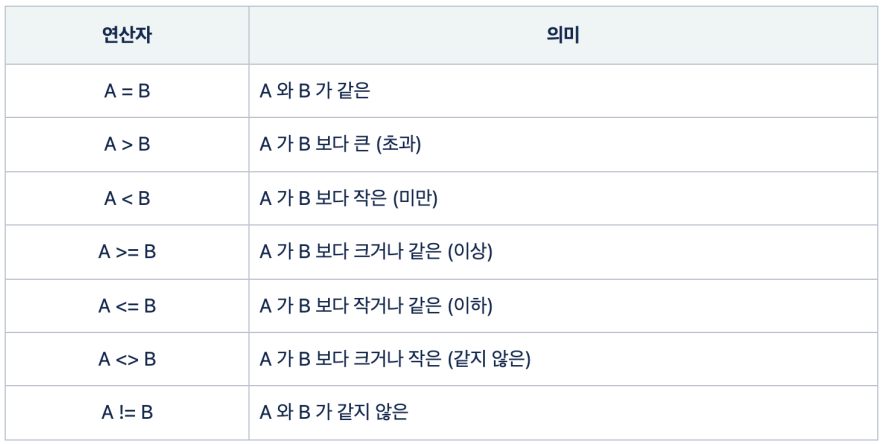
+----+-----------+------------+------+------+-------------------------+--------------------------+

7 rows in set (0.00 sec)

# Comparison Operators(비교 연산자)

아래와 같이 비교 연산자로 데이터를 조회할 수 있다.  
비교 연산자는 where 구문에서 사용된다.

실습을 통해 확인해보자.



## A = B

나이가 29세인 데이터를 이름과 나이로 조회.

mysql> select name, age

-> from celeb

-> where age=29

-> order by age;

+-----------+------+

| name | age |

+-----------+------+

| 아이유 | 29 |

+-----------+------+

1 row in set (0.00 sec)

## A != B

나이가 29세가 **아닌** 데이터를 이름과 나이로 조회.

mysql> select name, age

-> from celeb

-> where age!=29

-> order by age;

+-----------+------+

| name | age |

+-----------+------+

| 이수현 | 23 |

| 이미주 | 28 |

| 송강 | 28 |

| 강동원 | 41 |

| 차승원 | 48 |

| 유재석 | 50 |

+-----------+------+

6 rows in set (0.00 sec)

## A > B

나이가 29세 **보다 큰** 데이터를 이름과 나이로 조회.

mysql> select name, age

-> from celeb

-> where age>29

-> order by age;

+-----------+------+

| name | age |

+-----------+------+

| 강동원 | 41 |

| 차승원 | 48 |

| 유재석 | 50 |

+-----------+------+

3 rows in set (0.00 sec)

## A < B

나이가 29세 **보다 작은** 데이터를 이름과 나이로 조회.

mysql> select name, age

-> from celeb

-> where age<29

-> order by age;

+-----------+------+

| name | age |

+-----------+------+

| 이수현 | 23 |

| 이미주 | 28 |

| 송강 | 28 |

+-----------+------+

3 rows in set (0.00 sec)

## A >= B

나이가 29세 **이상인** 데이터를 이름과 나이로 조회.

mysql> select name, age

-> from celeb

-> where age>=29

-> order by age;

+-----------+------+

| name | age |

+-----------+------+

| 아이유 | 29 |

| 강동원 | 41 |

| 차승원 | 48 |

| 유재석 | 50 |

+-----------+------+

4 rows in set (0.00 sec)

## A <= B

나이가 29세 **이하인** 데이터를 이름과 나이로 조회.

mysql> select name, age

-> from celeb

-> where age<=29

-> order by age;

+-----------+------+

| name | age |

+-----------+------+

| 이수현 | 23 |

| 이미주 | 28 |

| 송강 | 28 |

| 아이유 | 29 |

+-----------+------+

4 rows in set (0.00 sec)

## A <> B

나이가 29세가**아닌**(보다 크거나 작은) 데이터를 이름과 나이로 조회.

mysql> select name, age

-> from celeb

-> where age<>29

-> order by age;

+-----------+------+

| name | age |

+-----------+------+

| 이수현 | 23 |

| 이미주 | 28 |

| 송강 | 28 |

| 강동원 | 41 |

| 차승원 | 48 |

| 유재석 | 50 |

+-----------+------+

6 rows in set (0.00 sec)

아래의 가상의 celeb table로 실습을 진행해보자.

mysql> desc celeb;

+-----------+-------------+------+-----+---------+----------------+

| Field | Type | Null | Key | Default | Extra |

+-----------+-------------+------+-----+---------+----------------+

| ID | int | NO | PRI | NULL | auto\_increment |

| NAME | varchar(32) | NO | | | |

| BIRTHDAY | date | YES | | NULL | |

| AGE | int | YES | | NULL | |

| SEX | char(1) | YES | | NULL | |

| JOB\_TITLE | varchar(32) | YES | | NULL | |

| AGENCY | varchar(32) | YES | | NULL | |

+-----------+-------------+------+-----+---------+----------------+

7 rows in set (0.00 sec)

mysql> select \* from celeb;

+----+-----------+------------+------+------+-------------------------+--------------------------+

| ID | NAME | BIRTHDAY | AGE | SEX | JOB\_TITLE | AGENCY |

+----+-----------+------------+------+------+-------------------------+--------------------------+

| 1 | 아이유 | 1993-05-16 | 29 | F | 가수, 텔런트 | EDAM엔터테이먼트 |

| 2 | 이미주 | 1994-09-23 | 28 | F | 가수 | 울림엔터테이먼트 |

| 3 | 송강 | 1994-04-23 | 28 | M | 텔런트 | 나무엑터스 |

| 4 | 강동원 | 1981-01-18 | 41 | M | 영화배우, 텔런트 | YG엔터테이먼트 |

| 5 | 유재석 | 1972-08-14 | 50 | M | MC, 개그맨 | 안테나 |

| 6 | 차승원 | 1970-06-07 | 48 | M | 영화배우, 모델 | YG엔터테이먼트 |

| 7 | 이수현 | 1999-05-04 | 23 | F | 가수 | YG엔터테이먼트 |

+----+-----------+------------+------+------+-------------------------+--------------------------+

7 rows in set (0.00 sec)

# Logical Operators

아래와 같이 논리 연산자로 데이터를 조회할 수 있다.  
비교 연산자는 where 구문에서 사용된다.

실습을 통해 확인해보자.



## AND

조건을 **모두** 만족하는 경우인 데이터를 조회.

ex) **나이**가 29세이고 **성별**이 여성인 데이터 조회.

mysql> select \*

-> from celeb

-> where age=29 and sex='F';

+----+-----------+------------+------+------+-------------------+------------------------+

| ID | NAME | BIRTHDAY | AGE | SEX | JOB\_TITLE | AGENCY |

+----+-----------+------------+------+------+-------------------+------------------------+

| 1 | 아이유 | 1993-05-16 | 29 | F | 가수, 텔런트 | EDAM엔터테이먼트 |

+----+-----------+------------+------+------+-------------------+------------------------+

1 row in set (0.00 sec)

ex) **나이**가 40세 보다 크고 **성별**이 남성인 데이터를 나이 역순으로 조회.

mysql> select \*

-> from celeb

-> where age>40 and sex='M'

-> order by age desc;

+----+-----------+------------+------+------+-------------------------+----------------------+

| ID | NAME | BIRTHDAY | AGE | SEX | JOB\_TITLE | AGENCY |

+----+-----------+------------+------+------+-------------------------+----------------------+

| 5 | 유재석 | 1972-08-14 | 50 | M | MC, 개그맨 | 안테나 |

| 6 | 차승원 | 1970-06-07 | 48 | M | 영화배우, 모델 | YG엔터테이먼트 |

| 4 | 강동원 | 1981-01-18 | 41 | M | 영화배우, 텔런트 | YG엔터테이먼트 |

+----+-----------+------------+------+------+-------------------------+----------------------+

3 rows in set (0.00 sec)

## OR

조건들 중 **하나**라도 만족하는 경우의 데이터 조회

ex) 나이가 25세보다 작거나 30보다 큰 데이터를 나이순으로 조회

mysql> select \*

-> from celeb

-> where age<25 or age>30

-> order by age;

+----+-----------+------------+------+------+-------------------------+----------------------+

| ID | NAME | BIRTHDAY | AGE | SEX | JOB\_TITLE | AGENCY |

+----+-----------+------------+------+------+-------------------------+----------------------+

| 7 | 이수현 | 1999-05-04 | 23 | F | 가수 | YG엔터테이먼트 |

| 4 | 강동원 | 1981-01-18 | 41 | M | 영화배우, 텔런트 | YG엔터테이먼트 |

| 6 | 차승원 | 1970-06-07 | 48 | M | 영화배우, 모델 | YG엔터테이먼트 |

| 5 | 유재석 | 1972-08-14 | 50 | M | MC, 개그맨 | 안테나 |

+----+-----------+------------+------+------+-------------------------+----------------------+

4 rows in set (0.00 sec)

## OR, AND

논리 연산자 2개 이상을 한 구문 안에서 섞어서 쓸 수 있다.

시스템은 AND, OR 연산자가 있을 때 AND 연산자부터 연산을 한다.  
하지만 가독성 및 쿼리가 길어질 시 혼동할 수가 있으니 괄호'()'를 사용하자.

연산자를 사용할 때, 산술 연산할 때와 같이 괄호'()'를 사용하면 괄호 안이 먼저 연산이 된다.  
'순서'를 고려하여 괄호를 잘 활용하도록 한다.

ex) 나이가 29세보다 작고 여자이거나, 나이가 30세 보다 크고 남자인 데이터를 나이와 성별 순으로 정렬하여 조회.  
(29 and 여자) or (30 and 남자)인 조건을 조회하면 된다.

mysql> select \*

-> from celeb

-> where (age<29 and sex='F') or (age>30 and sex='M')

-> order by age, sex;

+----+-----------+------------+------+------+-------------------------+--------------------------+

| ID | NAME | BIRTHDAY | AGE | SEX | JOB\_TITLE | AGENCY |

+----+-----------+------------+------+------+-------------------------+--------------------------+

| 7 | 이수현 | 1999-05-04 | 23 | F | 가수 | YG엔터테이먼트 |

| 2 | 이미주 | 1994-09-23 | 28 | F | 가수 | 울림엔터테이먼트 |

| 4 | 강동원 | 1981-01-18 | 41 | M | 영화배우, 텔런트 | YG엔터테이먼트 |

| 6 | 차승원 | 1970-06-07 | 48 | M | 영화배우, 모델 | YG엔터테이먼트 |

| 5 | 유재석 | 1972-08-14 | 50 | M | MC, 개그맨 | 안테나 |

+----+-----------+------------+------+------+-------------------------+--------------------------+

5 rows in set (0.00 sec)

## NOT

해당하는 조건이 **'아닌'**(만족하지 않는) 경우의 데이터를 조회.

ex) 성별이 여성이 '아닌' 데이터 조회.

mysql> select \*

-> from celeb

-> where not sex='F';

+----+-----------+------------+------+------+-------------------------+----------------------+

| ID | NAME | BIRTHDAY | AGE | SEX | JOB\_TITLE | AGENCY |

+----+-----------+------------+------+------+-------------------------+----------------------+

| 3 | 송강 | 1994-04-23 | 28 | M | 텔런트 | 나무엑터스 |

| 4 | 강동원 | 1981-01-18 | 41 | M | 영화배우, 텔런트 | YG엔터테이먼트 |

| 5 | 유재석 | 1972-08-14 | 50 | M | MC, 개그맨 | 안테나 |

| 6 | 차승원 | 1970-06-07 | 48 | M | 영화배우, 모델 | YG엔터테이먼트 |

+----+-----------+------------+------+------+-------------------------+----------------------+

4 rows in set (0.00 sec)

## BETWEEN

해당하는 조건 값이 범위 사이에 있는 경우의 데이터를 조회.  
and 혹은 or을 섞어쓰면 같은 조회가 가능하지만, 좀더 간편하게 쓸 수 있다는 장점이 있다.

ex) 생년월일이 1980년에서 1995년 사이가 아닌 데이터 조회.

# 1

mysql> select \*

-> from celeb

-> where birthday not between 19800101 and 19951231;

+----+-----------+------------+------+------+----------------------+----------------------+

| ID | NAME | BIRTHDAY | AGE | SEX | JOB\_TITLE | AGENCY |

+----+-----------+------------+------+------+----------------------+----------------------+

| 5 | 유재석 | 1972-08-14 | 50 | M | MC, 개그맨 | 안테나 |

| 6 | 차승원 | 1970-06-07 | 48 | M | 영화배우, 모델 | YG엔터테이먼트 |

| 7 | 이수현 | 1999-05-04 | 23 | F | 가수 | YG엔터테이먼트 |

+----+-----------+------------+------+------+----------------------+----------------------+

3 rows in set (0.00 sec)

# 2

mysql> select \*

-> from celeb

-> where birthday<19800101 or birthday>19951231;

+----+-----------+------------+------+------+----------------------+----------------------+

| ID | NAME | BIRTHDAY | AGE | SEX | JOB\_TITLE | AGENCY |

+----+-----------+------------+------+------+----------------------+----------------------+

| 5 | 유재석 | 1972-08-14 | 50 | M | MC, 개그맨 | 안테나 |

| 6 | 차승원 | 1970-06-07 | 48 | M | 영화배우, 모델 | YG엔터테이먼트 |

| 7 | 이수현 | 1999-05-04 | 23 | F | 가수 | YG엔터테이먼트 |

+----+-----------+------------+------+------+----------------------+----------------------+

3 rows in set (0.00 sec)

## IN

해당 조건이 목록 안에 존재하는 경우의 데이터를 조회  
or 연산자를 사용하면 같은 조회가 가능하지만, 좀 더 간편하게 쓸 수 있다는 장점이 있다.

ex) 나이가 28세, 28세 중 하나인 데이터 조회

# 1

mysql> select \*

-> from celeb

-> where age in (28, 48);

+----+-----------+------------+------+------+----------------------+--------------------------+

| ID | NAME | BIRTHDAY | AGE | SEX | JOB\_TITLE | AGENCY |

+----+-----------+------------+------+------+----------------------+--------------------------+

| 2 | 이미주 | 1994-09-23 | 28 | F | 가수 | 울림엔터테이먼트 |

| 3 | 송강 | 1994-04-23 | 28 | M | 텔런트 | 나무엑터스 |

| 6 | 차승원 | 1970-06-07 | 48 | M | 영화배우, 모델 | YG엔터테이먼트 |

+----+-----------+------------+------+------+----------------------+--------------------------+

3 rows in set (0.00 sec)

# 2

mysql> select \*

-> from celeb

-> where age=28 or age=48;

+----+-----------+------------+------+------+----------------------+--------------------------+

| ID | NAME | BIRTHDAY | AGE | SEX | JOB\_TITLE | AGENCY |

+----+-----------+------------+------+------+----------------------+--------------------------+

| 2 | 이미주 | 1994-09-23 | 28 | F | 가수 | 울림엔터테이먼트 |

| 3 | 송강 | 1994-04-23 | 28 | M | 텔런트 | 나무엑터스 |

| 6 | 차승원 | 1970-06-07 | 48 | M | 영화배우, 모델 | YG엔터테이먼트 |

+----+-----------+------------+------+------+----------------------+--------------------------+

3 rows in set (0.00 sec)

## LIKE

해당 조건이 '패턴'에 맞는 데이터 조회

패턴값

* % : 0개 이상의 문자열 패턴
* \_(언더바) : 하나의 문자 패턴(2글자면 \_\_(언더바 2개))

ex) 두번째 글자가 'G'로 시작하는 소속사 이름을 가진 데이터 조회.

mysql> select \*

-> from celeb

-> where agency like '\_G%';

+----+-----------+------------+------+------+-------------------------+----------------------+

| ID | NAME | BIRTHDAY | AGE | SEX | JOB\_TITLE | AGENCY |

+----+-----------+------------+------+------+-------------------------+----------------------+

| 4 | 강동원 | 1981-01-18 | 41 | M | 영화배우, 텔런트 | YG엔터테이먼트 |

| 6 | 차승원 | 1970-06-07 | 48 | M | 영화배우, 모델 | YG엔터테이먼트 |

| 7 | 이수현 | 1999-05-04 | 23 | F | 가수 | YG엔터테이먼트 |

+----+-----------+------------+------+------+-------------------------+----------------------+

3 rows in set (0.00 sec)

ex) 직업을 2개 가지는 데이터 조회

mysql> select \*

-> from celeb

-> where job\_title like '%, %';

+----+-----------+------------+------+------+-------------------------+------------------------+

| ID | NAME | BIRTHDAY | AGE | SEX | JOB\_TITLE | AGENCY |

+----+-----------+------------+------+------+-------------------------+------------------------+

| 1 | 아이유 | 1993-05-16 | 29 | F | 가수, 텔런트 | EDAM엔터테이먼트 |

| 4 | 강동원 | 1981-01-18 | 41 | M | 영화배우, 텔런트 | YG엔터테이먼트 |

| 5 | 유재석 | 1972-08-14 | 50 | M | MC, 개그맨 | 안테나 |

| 6 | 차승원 | 1970-06-07 | 48 | M | 영화배우, 모델 | YG엔터테이먼트 |

+----+-----------+------------+------+------+-------------------------+------------------------+

4 rows in set (0.00 sec)

아래 2개의 가상 셀럽 table로 실습을 진행해보자.

**celeb table**

mysql> desc celeb;

+-----------+-------------+------+-----+---------+----------------+

| Field | Type | Null | Key | Default | Extra |

+-----------+-------------+------+-----+---------+----------------+

| ID | int | NO | PRI | NULL | auto\_increment |

| NAME | varchar(32) | NO | | | |

| BIRTHDAY | date | YES | | NULL | |

| AGE | int | YES | | NULL | |

| SEX | char(1) | YES | | NULL | |

| JOB\_TITLE | varchar(32) | YES | | NULL | |

| AGENCY | varchar(32) | YES | | NULL | |

+-----------+-------------+------+-----+---------+----------------+

7 rows in set (0.00 sec)

mysql> select \* from celeb;

+----+-----------+------------+------+------+-------------------------+--------------------------+

| ID | NAME | BIRTHDAY | AGE | SEX | JOB\_TITLE | AGENCY |

+----+-----------+------------+------+------+-------------------------+--------------------------+

| 1 | 아이유 | 1993-05-16 | 29 | F | 가수, 텔런트 | EDAM엔터테이먼트 |

| 2 | 이미주 | 1994-09-23 | 28 | F | 가수 | 울림엔터테이먼트 |

| 3 | 송강 | 1994-04-23 | 28 | M | 텔런트 | 나무엑터스 |

| 4 | 강동원 | 1981-01-18 | 41 | M | 영화배우, 텔런트 | YG엔터테이먼트 |

| 5 | 유재석 | 1972-08-14 | 50 | M | MC, 개그맨 | 안테나 |

| 6 | 차승원 | 1970-06-07 | 48 | M | 영화배우, 모델 | YG엔터테이먼트 |

| 7 | 이수현 | 1999-05-04 | 23 | F | 가수 | YG엔터테이먼트 |

+----+-----------+------------+------+------+-------------------------+--------------------------+

7 rows in set (0.00 sec)

**show table**

mysql> desc snl\_show;

+----------------+-------------+------+-----+---------+----------------+

| Field | Type | Null | Key | Default | Extra |

+----------------+-------------+------+-----+---------+----------------+

| ID | int | NO | PRI | NULL | auto\_increment |

| SEASON | int | NO | | NULL | |

| EPISODE | int | NO | | NULL | |

| BROADCAST\_DATE | date | YES | | NULL | |

| HOST | varchar(32) | NO | | NULL | |

+----------------+-------------+------+-----+---------+----------------+

5 rows in set (0.00 sec)

mysql> select \* from snl\_show;

+----+--------+---------+----------------+-----------+

| ID | SEASON | EPISODE | BROADCAST\_DATE | HOST |

+----+--------+---------+----------------+-----------+

| 1 | 8 | 7 | 2020-09-05 | 강동원 |

| 2 | 8 | 8 | 2020-09-12 | 유재석 |

| 3 | 8 | 9 | 2020-09-19 | 차승원 |

| 4 | 8 | 10 | 2020-09-26 | 이수현 |

| 5 | 9 | 1 | 2021-09-04 | 이병헌 |

| 6 | 9 | 2 | 2021-09-11 | 하지원 |

| 7 | 9 | 3 | 2021-09-18 | 제시 |

| 8 | 9 | 4 | 2021-09-25 | 조정석 |

| 9 | 9 | 5 | 2021-10-02 | 조여정 |

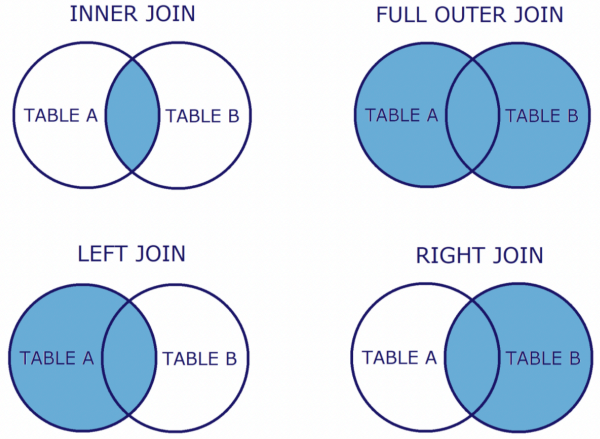
| 10 | 9 | 6 | 2021-10-09 | 옥주현 |

+----+--------+---------+----------------+-----------+

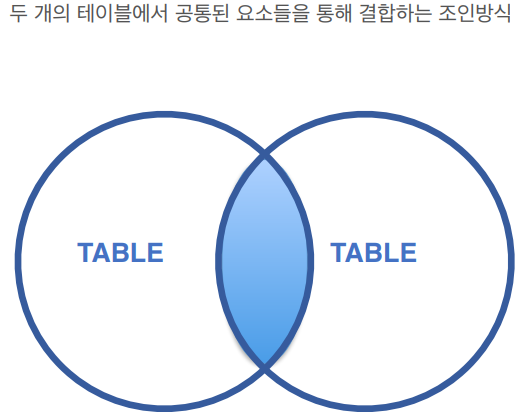
10 rows in set (0.01 sec)

# JOIN

두 개 이상의 table을 결합할 때 사용된다.  
쿼리를 작성할 시, column부분에서 닷('.')을 사용한다. **table.column**으로 지정하여 table에 해당하는 column인 것을 명시해주어야 한다.



## INNER JOIN



* 문법

select column1, column2, ...  
from tableA  
inner join tableB # join 방법  
on tableA.column = tableB.column # 합칠 때 기준 컬럼  
where condition; # 조건

ex) snl\_show에 호스트로 출연한 celeb을 기준으로, celeb table과 snl\_show table을 inner join하여 각 teble의 id와 이름 조회

mysql> select celeb.id, celeb.name, snl\_show.id, snl\_show.host

-> from celeb

-> inner join snl\_show

-> on celeb.name = snl\_show.host; # join 기준

+----+-----------+----+-----------+

| id | name | id | host |

+----+-----------+----+-----------+

| 4 | 강동원 | 1 | 강동원 |

| 5 | 유재석 | 2 | 유재석 |

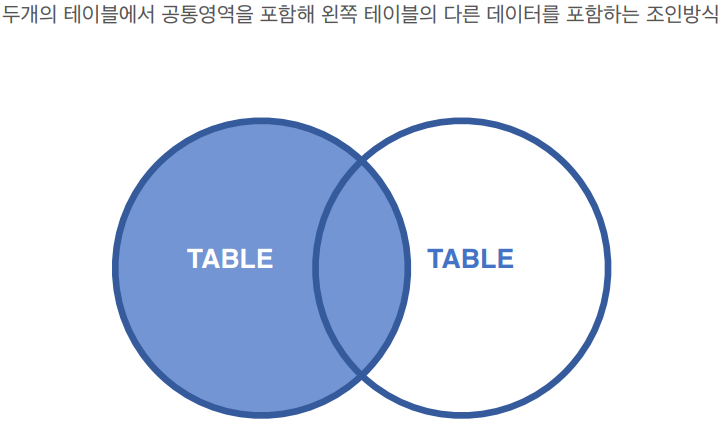
| 6 | 차승원 | 3 | 차승원 |

| 7 | 이수현 | 4 | 이수현 |

+----+-----------+----+-----------+

4 rows in set (0.00 sec)

## LEFT JOIN



* 문법

select column1, column2, ...  
from tableA  
left join tableB # join 방법  
on tableA.column = tableB.column # 합칠 때 기준 컬럼  
where condition; # 조건

ex) snl\_show에 호스트로 출연한 celeb을 기준으로, celeb table과 snl\_show table을 left join하여 각 teble의 id와 이름 조회  
아이유, 이미주, 송강은 snl\_show에 출연하지 않았기에 NaN값으로 조회가 되었다.

mysql> select celeb.id, celeb.name, snl\_show.id, snl\_show.host

-> from celeb

-> left join snl\_show

-> on celeb.name = snl\_show.host;

+----+-----------+------+-----------+

| id | name | id | host |

+----+-----------+------+-----------+

| 1 | 아이유 | NULL | NULL |

| 2 | 이미주 | NULL | NULL |

| 3 | 송강 | NULL | NULL |

| 4 | 강동원 | 1 | 강동원 |

| 5 | 유재석 | 2 | 유재석 |

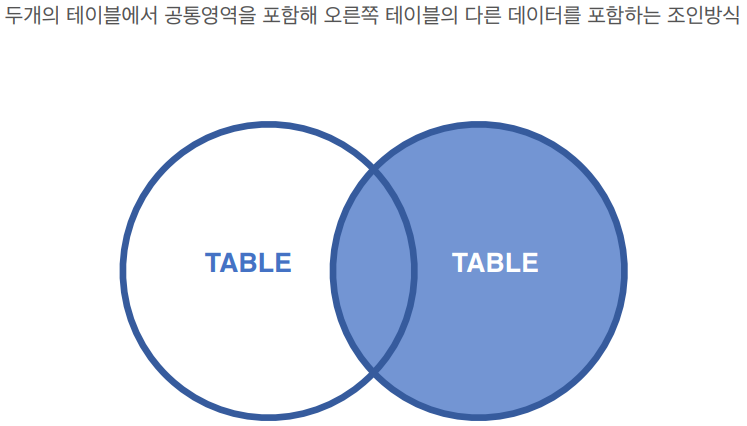
| 6 | 차승원 | 3 | 차승원 |

| 7 | 이수현 | 4 | 이수현 |

+----+-----------+------+-----------+

7 rows in set (0.00 sec)

## RIGHT JOIN



* 문법

select column1, column2, ...  
from tableA  
right join tableB # join 방법  
on tableA.column = tableB.column # 합칠 때 기준 컬럼  
where condition; # 조건

ex) snl\_show에 호스트로 출연한 celeb을 기준으로, celeb table과 snl\_show table을 right join하여 각 teble의 id와 이름 조회  
이병헌, 하지원, 제시, 조정석, 조여정, 옥주현은 celeb table에 없는 데이터이기에 NaN값으로 조회가 되었다.

mysql> select celeb.id, celeb.name, snl\_show.id, snl\_show.host

-> from celeb

-> right join snl\_show

-> on celeb.name = snl\_show.host;

+------+-----------+----+-----------+

| id | name | id | host |

+------+-----------+----+-----------+

| 4 | 강동원 | 1 | 강동원 |

| 5 | 유재석 | 2 | 유재석 |

| 6 | 차승원 | 3 | 차승원 |

| 7 | 이수현 | 4 | 이수현 |

| NULL | NULL | 5 | 이병헌 |

| NULL | NULL | 6 | 하지원 |

| NULL | NULL | 7 | 제시 |

| NULL | NULL | 8 | 조정석 |

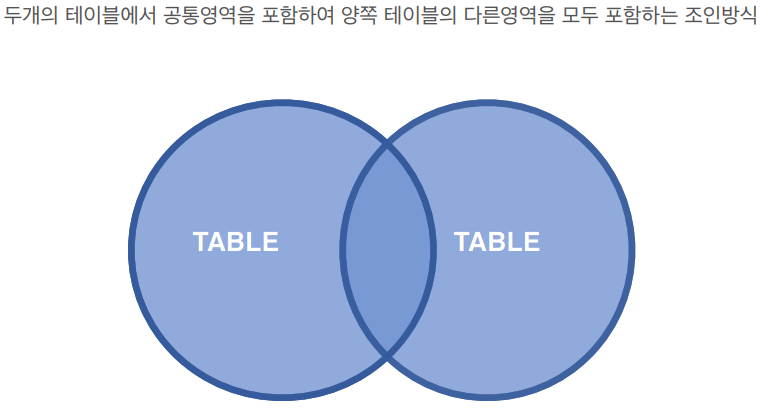
| NULL | NULL | 9 | 조여정 |

| NULL | NULL | 10 | 옥주현 |

+------+-----------+----+-----------+

10 rows in set (0.00 sec)

## FULL OUTER JOIN



* 문법

select column1, column2, ...  
from tableA  
right join tableB # join 방법  
on tableA.column = tableB.column # 합칠 때 기준 컬럼  
where condition; # 조건

ex) snl\_show에 호스트로 출연한 celeb을 기준으로, celeb table과 snl\_show table을 full outer join하여 각 teble의 id와 이름 조회  
에러 발생

# 지원하지 않는 형식

mysql> select celeb.id, celeb.name, snl\_show.id, snl\_show.host

-> from celeb

-> full outer join snl\_show

-> on celeb.name = snl\_show.host;

ERROR 1064 (42000): You have an error in your SQL syntax; check the manual that corresponds to your MySQL server version for the right syntax to use near 'outer join snl\_show

on celeb.name = snl\_show.host' at line 3

MySQL은 full outer join을 지원하지 않기에 다른 쿼리로 같은 결과를 만들어보자.  
left join과 right join을 union하여 만들 수 있다.

* 문법

select column1, column2, ...  
from tableA  
left join tableB # join 방법  
on tableA.column = tableB.column # 합칠 때 기준 컬럼  
union # 중복은 제외  
select column1, column2, ...  
from tableA  
right join tableB # join 방법  
on tableA.column = tableB.column # 합칠 때 기준 컬럼  
where condition;

ex) 다시 snl\_show에 호스트로 출연한 celeb을 기준으로, celeb table과 snl\_show table을 full outer join하여 각 teble의 id와 이름 조회  
위 3행은 celeb에만 있는 데이터이기에 snl\_show는 NaN값이고, 아래 6행은 snl\_show에만 있는 데이터이기에 celeb은 NaN값이 조회가 되었다.

mysql> select celeb.id, celeb.name, snl\_show.id, snl\_show.host

-> from celeb

-> left join snl\_show

-> on celeb.name = snl\_show.host

-> union

-> select celeb.id, celeb.name, snl\_show.id, snl\_show.host

-> from celeb

-> right join snl\_show

-> on celeb.name = snl\_show.host;

+------+-----------+------+-----------+

| id | name | id | host |

+------+-----------+------+-----------+

| 1 | 아이유 | NULL | NULL |

| 2 | 이미주 | NULL | NULL |

| 3 | 송강 | NULL | NULL |

| 4 | 강동원 | 1 | 강동원 |

| 5 | 유재석 | 2 | 유재석 |

| 6 | 차승원 | 3 | 차승원 |

| 7 | 이수현 | 4 | 이수현 |

| NULL | NULL | 5 | 이병헌 |

| NULL | NULL | 6 | 하지원 |

| NULL | NULL | 7 | 제시 |

| NULL | NULL | 8 | 조정석 |

| NULL | NULL | 9 | 조여정 |

| NULL | NULL | 10 | 옥주현 |

+------+-----------+------+-----------+

13 rows in set (0.00 sec)

## SELF JOIN

inner join과 유사하고 table 간의 공통된 데이터를 조회하며 가장 자주 쓰이는 join 방법이다.

* 문법

select column1, column2, ...  
from tableA, tableB, ... # 여러 개의 table  
where condition; # where에 join 기준을 정의

ex) snl\_show에 호스트로 출연한 celeb을 기준으로, celeb table과 snl\_show table을 self join하여 각 teble의 id와 이름 조회

mysql> select celeb.id, celeb.name, snl\_show.id, snl\_show.host

-> from celeb, snl\_show

-> where celeb.name = snl\_show.host;

+----+-----------+----+-----------+

| id | name | id | host |

+----+-----------+----+-----------+

| 4 | 강동원 | 1 | 강동원 |

| 5 | 유재석 | 2 | 유재석 |

| 6 | 차승원 | 3 | 차승원 |

| 7 | 이수현 | 4 | 이수현 |

+----+-----------+----+-----------+

4 rows in set (0.00 sec)

ex) celeb table의 셀럽 중, snl\_show에 host로 출연했고 소속사가 안테나인 셀럽의 이름과 직업을 조회

mysql> select celeb.name, celeb.job\_title

-> from celeb, snl\_show

-> where celeb.name = snl\_show.host and celeb.agency = '안테나';

+-----------+---------------+

| name | job\_title |

+-----------+---------------+

| 유재석 | MC, 개그맨 |

+-----------+---------------+

1 row in set (0.00 sec)