SQL

> SQL(StructuredQueryLanguage)는 관계형 데이터베이스 관리시스템(RDBMS)의데이터를 관리하기 위해 설계된 특수 목적의 프로그래밍 언어

Basic Syntax

SQL 문법의 세가지 종류

- DDL (Data Definition Languege)
  - CREATE
  - DROP
  - ALTER
- DML (Data Manipulation Languege)
  - INSERT
  - UPDATE
  - DELETE
  - SELECT
- DCL (Data Control Languege)
  - GRANT
  - REVOKE
  - COMMIT
  - ROLLBACK

https://programmers.co.kr/learn/challenges

문제를 푼 것을 함께 정리한 것

ALTER

- 테이블명 변경

```
ALTER TABLE 기존테이블명
RENAME TO 새로운테이블명;
```

- 새로운 컬럼 추가

```
ALTER TABLE 테이블명
ADD COLUMN 컬럼명 datatype;
```

- 컬럼명 변경

```
ALTER TABLE 테이블명
CHANGE 원래이름 바꿀이름 int;
```

SELECT

- 모든 레코드의 조회

```
SELECT * From ANIMAL_INS;
```

- 역순으로 조회하기

```
SELECT NAME, DATETIME
FROM ANIMAL_INS
ORDER BY ANIMAL_ID DESC;
```

- 조건에 따라 조회하기

```
SELECT ANIMAL_ID, NAME
FROM ANIMAL_INS
WHERE INTAKE_CONDITION = 'Sick';
```

- 조건에 따라 조회하기2

```
SELECT ANIMAL_ID, NAME
FROM ANIMAL_INS
WHERE INTAKE_CONDITION != 'Aged';
```

- 여러 기준으로 정렬하기

```
SELECT ANIMAL_ID, NAME, DATETIME
FROM ANIMAL_INS
ORDER BY NAME, DATETIME DESC;
```

- 이름을 사전순으로 정렬하고, 이 중에서 보호를 나중에 시작한 고양이를 먼저 조회

- 정렬된 데이터의 상위 n개 레코드

```
SELECT NAME
FROM ANIMAL_INS
ORDER BY DATETIME
LIMIT 1;
```

중첩 SELECT

- 중첩된 SELECT => 테이블을 복제해서 생각해보자 (Panel Data)

```
SELECT a.CART_ID
FROM CART_PRODUCTS AS a, CART_PRODUCTS AS b
WHERE a.CART_ID = b.CART_ID AND a.NAME = 'MILK' AND b.NAME = 'Yogurt';
```

- SELECT 된 그룹바이의 테이블 안에 아이디가 있는지 여부

```
SELECT ID, NAME, HOST_ID
FROM PLACES
WHERE HOST_ID IN (
    SELECT HOST_ID
    FROM PLACES
    GROUP BY HOST_ID HAVING COUNT(ID) > 1)
ORDER BY ID;
```

NULL 처리

- null 인 이름을 'No name'으로 바꾸기
    - ifnull 사용

```
SELECT ANIMAL_TYPE, IFNULL(NAME, "No name") AS NAME, SEX_UPON_INTAKE
FROM ANIMAL_INS;
```

- coalesce 사용
    - coalesce는 정의된 열 중 null이 아닌 첫번째 값을 화면에 출력
    - coalesce('a', 'b')이면 a 출력 / coalesce(null, 'a', 'b') 이면 a 출력

```
SELECT ANIMAL_TYPE, coalesce(NAME, "No name") AS NAME, SEX_UPON_INTAKE
FROM ANIMAL_INS;
```

LIMIT, OFFSET, DISTICT

```
-- LIMIT: 원하는 개수(num)만큼 가져오기 --
SELECT column1, column2
FROM table
LIMIT num;

-- OFFSET: 특정 위치에서부터 가져올 때 --
-- (맨 위부터 num만큼 떨어진 값부터 가져온다는 의미)
SELECT column1, column2
FROM table
LIMIT num OFFSET num;
```

```sql
-- WHERE: 조건을 통해 값 가져오기 --
SELECT column1, column2
FROM table
WHERE column=value;

-- DISTINCT: 중복없이 가져오기 --
SELECT DISTINCT column1 FROM table;
```

IN

- WHERE 절 내 여러 값을 설정할 때 사용
- 이름을 가진 동물 찾기

```sql
SELECT ANIMAL_ID, NAME, SEX_UPON_INTAKE
FROM ANIMAL_INS
WHERE NAME IN ('Lucy', 'Ella', 'Pickle', 'Rogan', 'Sabrina', 'Mitty');
```

IF

- SELECT, WHRER 절에서 사용 가능

```sql
SELECT IF(10>5, '크다', '작다') AS RE
```

- 중성화 여부를 O, X로 대치해서 확인

```sql
select animal_id, name, if(sex_upon_intake like 'Intact%', 'X', 'O')
from animal_ins
order by animal_id
```

BETWEEN

- WHERE 절 내 검색 조건으로 범위 설정

```sql
SELECT * FROM TABLE
WHERE (count between 10 and 20) and not id in (10, 20)
```

SUM, MAX, MIN

- AVG(), COUNT(), STDEV(), VARIANCE()
- 가장 최근의 데이터

```sql
SELECT MAX(DATETIME)
FROM ANIMAL_INS;
```

- 가장 먼저의 데이터

```
SELECT MIN(DATETIME)
FROM ANIMAL_INS;
```

- 데이터의 수 구하기

```
SELECT count(ANIMAL_ID)
FROM ANIMAL_INS;
```

- 중복 제거 & NULL값 제거

```
SELECT count(DISTINCT NAME)
FROM ANIMAL_INS
WHERE NAME is NOT NULL;
```

- NULL 값 찾을 땐 is NULL

- GROUP BY with AS

```
SELECT
    ANIMAL_TYPE,
    COUNT(*) AS count
FROM
    ANIMAL_INS
GROUP BY
    ANIMAL_TYPE
ORDER BY
    ANIMAL_TYPE;
```

- 동명 동물 수 찾기

```
SELECT * FROM (
    SELECT NAME, COUNT(*) AS COUNT
    FROM ANIMAL_INS
    WHERE NAME is not NULL
    GROUP BY NAME
    ORDER BY NAME
) t
WHERE t.COUNT >= 2;
```

- 찾아진 결과 값을 임시의 테이블로 보고, 그 안에서 정렬 가능

- DATETIME 시간에 따라 테이블 구하기 **순서는 WHERE, GROUP, ORDER임**

```
SELECT HOUR(DATETIME) as HOUR, count(*) AS COUNT
FROM ANIMAL_OUTS
WHERE HOUR(DATETIME) BETWEEN 9 AND 19
GROUP BY HOUR
ORDER BY HOUR
;
```

- 입양 시각 구하기

Recursive

1. 메모리 상에 가상의 테이블을 저장
2. 재귀 쿼리를 이용하여 실제로 테이블을 생성하거나 데이터 삽입을 하지 않고 가상의 테이블 활용

```
WITH RECURSIVE 테이블명 AS (
    SELECT 초기값 AS 컬럼별명1
    UNION ALL  # 중복값을 포함 (UNION은 중복값 제거)
    SELECT 컬럼별명1 계산식
    FROM 테이블명
    WHERE 제어문
)
```

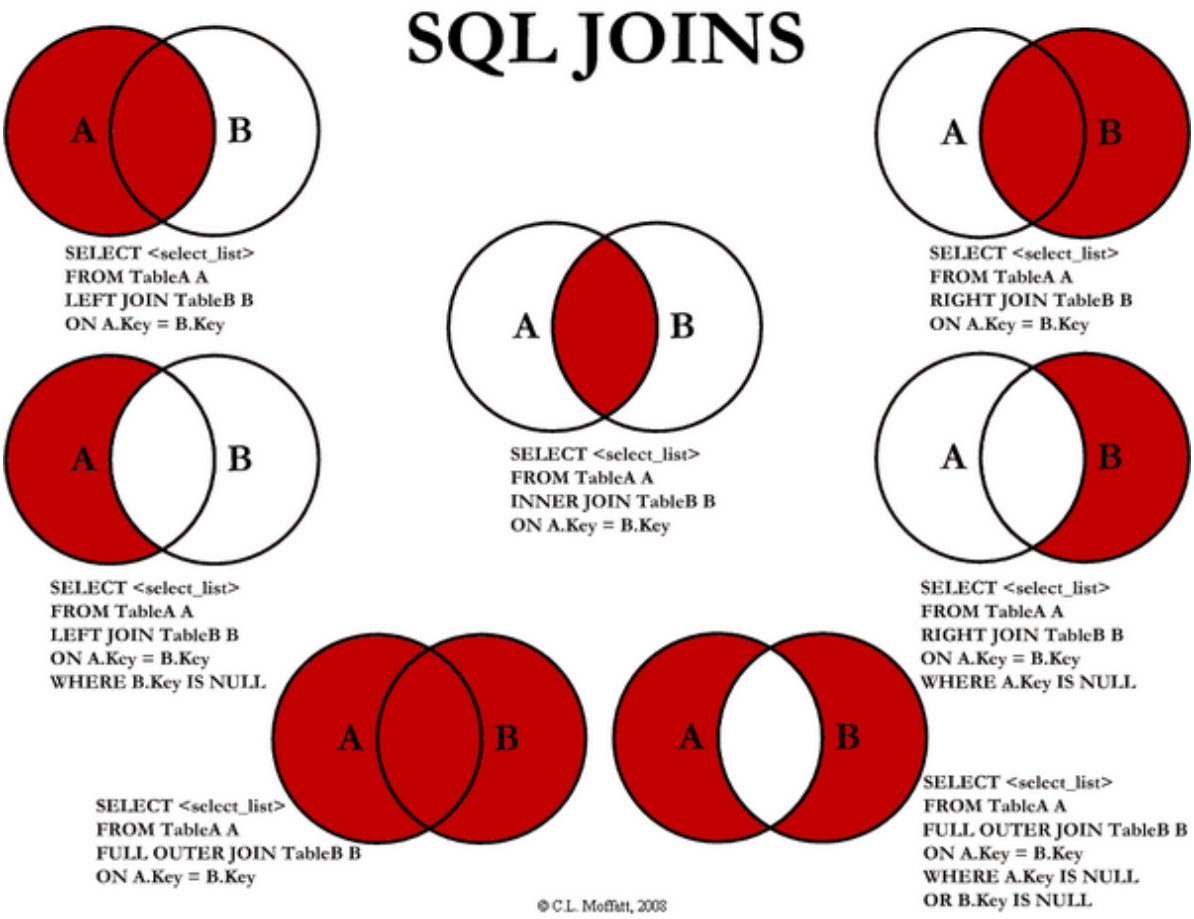
```
WITH RECURSIVE time AS (
    SELECT 0 AS HOUR
    UNION ALL
    SELECT HOUR+1 from time
    WHERE HOUR < 23
)

SELECT HOUR, COUNT(datetime) AS COUNT
FROM time
    LEFT JOIN ANIMAL_OUTS on HOUR = hour(datetime)
GROUP BY HOUR
ORDER BY HOUR;
```

```
# 구구단

WITH RECURSIVE gugu(n) AS (
    SELECT 1
    union all
    SELECT n+1 from gugu
    where n < 9
)
SELECT gugu.n * gugu2.n from gugu
cross join gugu as gugu2;
```

JOIN



- 없어진 기록 찾기

```
SELECT ANIMAL_OUTS.ANIMAL_ID, ANIMAL_OUTS.NAME
FROM ANIMAL_OUTS
    LEFT JOIN ANIMAL_INS ON ANIMAL_OUTS.ANIMAL_ID = ANIMAL_INS.ANIMAL_ID
    WHERE ANIMAL_INS.ANIMAL_ID is NULL;
```

- LETFT JOIN 한 뒤 조건 걸기

```
SELECT ANIMAL_INS.ANIMAL_ID, ANIMAL_INS.NAME
FROM ANIMAL_INS
LEFT JOIN ANIMAL_OUTS
    ON ANIMAL_INS.ANIMAL_ID = ANIMAL_OUTS.ANIMAL_ID
    WHERE ANIMAL_INS.DATETIME > ANIMAL_OUTS.DATETIME
    ORDER BY ANIMAL_INS.DATETIME;
```

```
SELECT ANIMAL_INS.NAME, ANIMAL_INS.DATETIME
FROM ANIMAL_INS
    LEFT JOIN ANIMAL_OUTS
    ON ANIMAL_INS.ANIMAL_ID = ANIMAL_OUTS.ANIMAL_ID
    WHERE ANIMAL_OUTS.ANIMAL_ID is NULL
    ORDER BY ANIMAL_INS.DATETIME
    LIMIT 3;
```

LIKE

> LIKE는 두 가지 와일드 카드(언더스코어 그리고 퍼센트 기호)와 함께 동작한다.

- `_` (반드시 이 자리에 한 개의 문자가 존재해야 한다는 뜻)

```
-- 20대인 사람들만 가져올 때 --

SELECT *
FROM users
WHERE age LIKE '2_';
```

- `%` (이 자리에 문자열이 있을 수도, 없을 수도 있다. 0개 이상이라는 뜻)

```
-- 지역번호가 02인 사람만 가져올 때 --

SELECT *
FROM users
WHERE phone LIKE '02-%';
```

- 두 개를 조합해서 사용할 수도 있다.

```
-- 핸드폰 중간 번호가 반드시 4자리면서 511로 시작되는 사람들 --

SELECT * FROM users
WHERE phone LIKE '%-511_-%';
```

- LEFT JOIN with LIKE
  - 보호소에서 중성화한 동물

```
SELECT AO.ANIMAL_ID, AO.ANIMAL_TYPE, AO.NAME
FROM ANIMAL_OUTS AS AO
    LEFT JOIN ANIMAL_INS AS AI
    ON AI.ANIMAL_ID = AO.ANIMAL_ID
    WHERE (AO.SEX_UPON_OUTCOME LIKE 'spayed%' or AO.SEX_UPON_OUTCOME LIKE
'neutered%')
    AND AI.SEX_UPON_INTAKE LIKE 'intact%'
    ORDER BY AO.ANIMAL_ID;
```

STRING

- 대소문자 구별법
- MySQL은 대소문자 구별하지 않으므로 구별하려면 BINARY

```
SELECT * FROM table
WHERE BINARY id LIKE '%El%'
```

DATE

- 오랜 기간 보호한 동물

```
SELECT AO.ANIMAL_ID, AO.NAME
FROM ANIMAL_OUTS AS AO
    INNER JOIN ANIMAL_INS AS AI
    ON AI.ANIMAL_ID = AO.ANIMAL_ID
    ORDER BY AO.DATETIME - AI.DATETIME DESC
    LIMIT 2;
```

- DATETIME => DATE
    - `date_format`

```
SELECT ANIMAL_ID, NAME, date_format(DATETIME, '%Y-%m-%d')
FROM ANIMAL_INS;
```

CASE

- 조건 if... else같이
- CASE
  WHEN 조건 1 THEN 반환값
  WHEN 조건2 THEN 반환값
  ELSE 반환값
  END

```
SELECT score
    CASE
        WHEN (score BETWEEN 100 AND 90) THEN 'A+'
        WHEN (score BETWEEN 89 AND 80) THEN 'A-'
        ELSE 'B'
    END
FROM student
```