

# SQL for Data Analysis 102

## SQL for Data Analysis 102

### ▼ How to use WHERE clause?

#### Common WHERE usage

ตัวอย่างด้านล่างคือตัวการเขียน `WHERE` ที่เราใช้กันบ่อยๆ เวลาเขียน `SQL` query

🌿 `LIKE` คือการเขียน pattern matching ปกติเราจะใช้ `LIKE` กับ wildcard คือ `%` หรือ `_`

- `%` ใช้ match any character ที่ตัวก็ได้
- `_` ใช้ match single character ตัวเดียว

ตัวอย่างเช่น `WHERE country LIKE 'U%'` จะฟิเตอร์เฉพาะประเทศที่ขึ้นต้นด้วยตัว U ทั้งหมด

หรือ `WHERE firstname LIKE 'J_hn'` จะฟิเตอร์ชื่อ firstname ลูกคำขึ้นต้นด้วยตัว J ตามด้วยตัวอักษรอะไรก็ได้หนึ่งตัวและปิดท้ายด้วย hn เช่น `John Jahn Jihn Jehn` เป็นต้น

🌿 ใน SQLite ตัว `LIKE` operator จะเป็นแบบ case insensitive ไม่สนตัวพิมพ์เล็กใหญ่ เวลาเขียน `LIKE 'J_hn'` สามารถ match ได้ทั้ง `JOHN john` หรือ `JohN` ไม่แตกต่างกัน

#### EP01 - Filter Data 1

- `AND` สองเงื่อนไขเป็นจริงทั้งคู่
- `OR` เงื่อนไขใดเงื่อนไขหนึ่งเป็นจริง
- `NOT` เปลี่ยนจาก `TRUE` เป็น `FALSE` or vice versa

```
-- filter data with WHERE clause
SELECT * FROM customers
WHERE country = 'USA';

SELECT * FROM customers
WHERE country = 'USA' AND state = 'CA';

SELECT * FROM customers
WHERE country = 'USA' OR country = 'Canada';

SELECT * FROM customers
WHERE LOWER(country) = 'united kingdom' AND state = 'CA';

SELECT * FROM customers
WHERE NOT (country = 'USA' OR country = 'Canada' OR country = 'France');
```

## EP02 - Filter Data 2

```
-- IN operator
SELECT * FROM customers
WHERE country IN ('USA', 'Canada', 'France');

-- NOT IN operator
SELECT * FROM customers
WHERE country NOT IN ('USA', 'Canada', 'France');
```

```
-- BETWEEN AND
SELECT * FROM customers
WHERE customerID BETWEEN 5 AND 10; -- inclusive

SELECT * FROM customers
WHERE customerid >= 5 AND customerid <= 10;
```

```
-- BETWEEN AND with DATE TIME column
SELECT invoicedata FROM invoices
WHERE invoicedate BETWEEN '2009-01-01 00:00:00' AND '2009-01-31 20:00:00';
```

Note - **SQLite** เก็บข้อมูล datetime เป็น text ปกติ โดยรูปแบบมาตรฐานของ datetime คือ **YYYY-MM-DD HH:MM:SS**

```
-- NULL
SELECT * FROM customers
WHERE company IS NULL;

SELECT * FROM customers
WHERE company IS NOT NULL;
```

## EP03 - Filter Data 3

### Wildcards

- **%** matches any number of characters (0 or more)
- **\_** matches single character
- ข้อจำกัดของ **LIKE** คือเป็นการ match แบบ case-insensitive แปลว่า **J\_HN** และ **j\_hn** จะได้ผลลัพธ์เหมือนกัน

```
-- Pattern Matching
SELECT * FROM customers
```

```
WHERE email LIKE '%@gmail.com'

SELECT * FROM customers
WHERE email NOT LIKE '%@gmail.com'
```

```
-- Find customers with phone number include 99
SELECT * FROM customers
WHERE phone LIKE '%99%';

-- Find customers firstname like 'John' etc.
# start with J end with hn find 1 character
SELECT * FROM customers
WHERE firstname LIKE 'J_hn';

# start with Leon find 2 character
SELECT firstname, lastname, country, email, phone FROM customers
WHERE firstnae LIKE 'Leon__';
```

## EP04 - COALESCE

```
-- replace NULL with desired values
SELECT
    company,
    COALESCE(company, 'End Customer') AS cleanCompany
FROM customers;
```

เราสามารถใช **CASE** เพื่อเขียนเงื่อนไข ทำความสะอาดข้อมูลคอลัมน์ที่มีค่า **NULL** ตามตัวอย่างด้านล่าง

```
-- replace NULL with desired values
SELECT
    company,
    CASE WHEN company IS NULL THEN 'End Customer'
        ELSE 'Corporate'
    END AS segment
FROM customers;
```

```
SELECT
    company,
    COALESCE(company, 'End Customer') AS 'Company clean',
    CASE WHEN company IS NULL THEN 'End Customer'
        ELSE 'Corporate'
    END AS 'segment'
FROM customers;
```

```

SELECT  firstname,
        company,
        COALESCE(company,"No Information") AS clean_company,
        CASE
            WHEN company IS NULL    THEN 'B2C'
            WHEN company IS NOT NULL THEN 'B2B'
        END
FROM customers

.....
SELECT  firstname,
        company,
        COALESCE(company,"No Information") AS clean_company,
        CASE
            WHEN company IS NULL    THEN 'B2C'
            ELSE 'B2B'
        END
FROM customers

=IF(ISBLANK(company),"B2C","B2B") ** in google sheet

```

## EP05 - Join Data with WHERE clause



วิธีการดึงข้อมูลจากหลายตารางด้วย **WHERE** clause ที่เราเรียนในบทเรียนนี้ เทียบเท่ากับการเขียน **INNER JOIN** หัวใจของการดึงข้อมูลแบบ **JOIN** คือ primary key = foreign key

```

SELECT * FROM artists, albums
WHERE artists.artistid = albums.artistid;

```

```

SELECT * FROM artists JOIN albums
ON artists.artistid = albums.artistid;

```

```

-- join table with WHERE clause, two tables
SELECT
    artists.artistid,
    artists.name AS artist_name,
    albums.title AS album_name
FROM artists, albums
WHERE artists.artistid = albums.artistid -- PK = FK
    AND artists.artistid IN (8, 100, 120);

```

```
-- join table with WHERE clause, three tables
SELECT
    artists.artistid,
    artists.name AS artist_name,
    albums.title AS album_name,
    tracks.name AS song_name
FROM artists, albums, tracks
WHERE artists.artistid = albums.artistid -- PK = FK
    AND albums.albumid = tracks.albumid
    AND artists.artistid IN (8, 100, 120);
```

## EP06 - Aggregate Functions

Aggregate functions คือฟังก์ชันสถิติเบื้องต้นไว้สรุปผลข้อมูล ฟังก์ชันที่เราเป็นประจําใน standard SQL จะมีอยู่ 5 functions คือ Count, Average, Sum, Min, Max



Aggregate Functions จะไม่สนใจ rows ที่เป็นค่า **NULL**



**AS** เป็น optional clause ไม่ต้องเขียนก็ได้

```
-- Aggregate Functions
SELECT
    ROUND(AVG(milliseconds), 2) AS avg_mill,
    SUM(milliseconds) AS sum_mill,
    MIN(milliseconds) AS min_mill,
    MAX(milliseconds) AS max_mill,
    COUNT(milliseconds) AS count_mill
FROM tracks;
```

## EP07 - Count Distinct

```
-- COUNT DISTINCT
SELECT COUNT(DISTINCT country), COUNT(*) FROM customers;
```

```
-- Intersect ข้อมูลที่ต้องการมีอยู่ในทั้ง 2 Tables
SELECT id FROM employee
```

```


INTERSECT
SELECT id FROM customers

-- EXCEPT ข้อมูลที่ต้องการอยู่ใน employee แต่ไม่อยู่ใน customers
SELECT id FROM employee
EXCEPT
SELECT id FROM customers

-- UNION 2 tables : union รวมข้อมูลแต่จะลบข้อมูลที่ซ้ำกัน , union all รวมข้อมูลแต่เอาข้อมูลที่ซ้ำมารวมกันด้วย
SELECT * FROM table1
UNION / UNION ALL
SELECT * FROM table2

```

## EP08 - Group By

 เราใช้ **GROUP BY** กับ **Aggregate Functions** เพื่อสรุปข้อมูลแบ่งตามกลุ่มที่เราต้องการ


```

-- Aggregate functions with GROUP BY clause
SELECT country, COUNT(*) AS count_country
FROM customers
GROUP BY country;

SELECT genres.name, COUNT(*) AS count_songs
FROM genres, tracks
WHERE genres.genreid = tracks.genreid
GROUP BY genres.name;

```

## EP09 - Having

 เราใช้ **HAVING** สำหรับ filter กลุ่ม (เขียนหลังจาก **GROUP BY** clause) ส่วน **WHERE** ใช้ filter ข้อมูลในตาราง (เขียนก่อน **GROUP BY** clause)

```

-- Filter groups
SELECT
    genres.name,
    COUNT(*) AS count_songs
FROM genres, tracks
WHERE genres.genreid = tracks.genreid AND genres.name <> 'Rock'
GROUP BY genres.name
HAVING COUNT(*) >= 100;

```

## EP10 - Order By & Limit

**ORDER BY** ใช้เพื่อ sort data เรียงข้อมูลจากน้อยไปมาก (default, ascending order) หรือถ้าอยากเรียงจากมากไปน้อยให้ใส่ **DESC** ต่อท้ายชื่อคอลัมน์

```
-- Order By + Limit
SELECT genres.name, COUNT(*)
FROM genres, tracks
WHERE genres.genreid = tracks.genreid
GROUP BY genres.name
ORDER BY COUNT(*) DESC      -- desc = descending order
LIMIT 5;

SELECT genres.name, COUNT(*)
FROM genres
JOIN tracks ON genres.genreid = tracks.genreid
GROUP BY genres.name
ORDER BY COUNT(*) DESC LIMIT 5;
```