

SQL for Data Analyst 103

Tags	Foundation	SQL
Class		
Finished Yet?	<input checked="" type="checkbox"/>	
Knowledge	 The Second Sprint: SQL	

Lesson 1: JOIN using WHERE clause

- เรา JOIN TABLE ได้ด้วย Primary Key และ Foreign Key (Primary Key หรือ column ที่มีค่าไม่ซ้ำใน Table หนึ่ง สามารถเชื่อมกับอีก Table ที่มี column ซึ่งเดียวกันแต่อาจมีค่าซ้ำกันเรียกว่า Foreign Key) [artist.ArtistId = albums.ArtistId]

- ต้องรู้ก่อนว่าอย่างจะได้อะไร จากนั้นถึงค่อย JOIN ตามความเหมาะสม

- ใช้ WHERE ใน การ JOIN ได้ เช่น กัน เช่น:

```
SELECT * FROM artists AS A, albums AS B  
WHERE A.ArtistId = B.ArtistId;
```

- เราสามารถ AS ได้ ได้ผลลัพธ์เหมือนกัน เช่น:

```
SELECT
    A.ArtistId,
    A.Name Artist_Name,
    B.Title Album_Name
FROM artists A, albums B
WHERE A.ArtistId = B.ArtistId;
```

-เราสามารถเพิ่มเติมได้ด้วยการเพิ่มเงื่อนไขตรง WHERE clause เช่น

```
SELECT
    A.ArtistId,
    A.Name Artist_Name,
    B.Title Album_Name
FROM artists A, albums B
WHERE A.ArtistId = B.ArtistId AND A.Name LIKE 'C%';
```

Lesson 2: Convert WHERE to INNER JOIN

```
#WHERE
Select * FROM artists, albums
WHERE artists.ArtistId = albums.ArtistID

#INNER JOIN
Select * FROM artists
JOIN albums ON artists.ArtistId = albums.ArtistID
```

-ตัวอย่างเพิ่มเติม:

```
#WHERE
SELECT
    A.ArtistId,
    A.Name Artist_Name,
    B.Title Album_Name
FROM artists A, albums B
WHERE A.ArtistId = B.ArtistId AND A.Name LIKE 'C%';

#INNER JOIN
SELECT
    A.ArtistId,
    A.Name Artist_Name,
    B.Title Album_Name
FROM artists A INNER JOIN albums B
```

```

ON A.ArtistId = B.ArtistId
WHERE A.Name LIKE 'C%';

```

- เราสามารถ JOIN ได้มากกว่า 2 ตาราง เช่น:

```

#JOIN 3 Tables
SELECT
    A.ArtistId,
    A.Name Artist_Name,
    B.Title Album_Name,
    C.Name Track_Name
FROM artists A
INNER JOIN albums B ON A.ArtistId = B.ArtistId
INNER JOIN tracks C ON B.AlbumId = C.AlbumId
WHERE A.Name LIKE 'C%';

```

Lesson 3: Review JOIN Concepts

- การ JOIN มี 4 แบบ คือ INNER JOIN, LEFT JOIN (2 แบบแรกใช้เยอะกว่า 2 แบบหลังมาก), RIGHT JOIN, และ FULL JOIN [SQLite ไม่ Support FULL JOIN และ RIGHT JOIN]

- INNER JOIN:

Inner Join

Table 1

PK_ID	Name
1	David
2	John
3	Marry
4	Anna
5	Kevin

Table 2



FK_ID	Major
1	Econ
2	Econ
5	Data
12	Engineer
35	Mkt

Result Set

PK_ID	Name	Major
1	David	Econ
2	John	Econ
5	Kevin	Data

- LEFT JOIN:

Left Join

Table 1

PK_ID	Name
1	David
2	John
3	Marry
4	Anna
5	Kevin

Table 2

FK_ID	Major
1	Econ
2	Econ
5	Data
12	Engineer
35	Mkt

+

Result Set

PK_ID	Name	Major
1	David	Econ
2	John	Econ
3	Marry	NULL
4	Anna	NULL
5	Kevin	Data

=

-FULL JOIN:

Full Join

Table 1

PK_ID	Name
1	David
2	John
3	Marry
4	Anna
5	Kevin

Table 2

FK_ID	Major
1	Econ
2	Econ
5	Data
12	Engineer
35	Mkt

+

Result Set

PK_ID	Name	Major
1	David	Econ
2	John	Econ
3	Mary	NULL
4	Anna	NULL
5	Kevin	Data
12	NULL	Engineer
35	NULL	Mkt

-ส่วน RIGHT JOIN สามารถเขียนได้ด้วยการสลับตำแหน่ง 2 Table และเขียน LEFT JOIN

Lesson 4: Review CREATE TABLE

```
###Create two tables
#Create book_shop table
CREATE TABLE book_shop (
```

```

    id INT, #id as integer
    name TEXT, #name as text
    release_year INT #release_year as integer
);

#Create favourite_book table
CREATE TABLE favourite_book (
    id INT, #id as integer
    author TEXT, #author as text
    reviews REAL #reviews as real number
);

#Insert data into book_shop table
INSERT INTO book_shop VALUES
    (1, 'Think Like A Freak', 2014), #(tuples)
    (2, 'Ultralearning', 2019),
    (3, 'Blue Ocean Strategy', 2015),
    (4, 'The Power of Habit', 2012),
    (5, 'Outliers', 2008);

#Insert data into favourite_book table
INSERT INTO favourite_book VALUES
    (1, 'Steven D. Levitt, Stephen J. Dubner', 1904),
    (4, 'Charles Duhigg', 12007),
    (5, 'Malcolm Gladwell', 12063);

```

Lesson 5: INNER vs. LEFT JOIN

```

#INNER JOIN
SELECT * FROM book_shop A #Select all columns from book_shop table as A
INNER JOIN favourite_book B #INNER JOIN favourite_book table as B
ON A.id = B.id; #A.id = Primary Key, B.id = Foreign Key

#LEFT JOIN
SELECT * FROM book_shop A #Select all columns from book_shop table as A
LEFT JOIN favourite_book B #LEFT JOIN favourite_book table as B
ON A.id = B.id; #A.id = Primary Key, B.id = Foreign Key

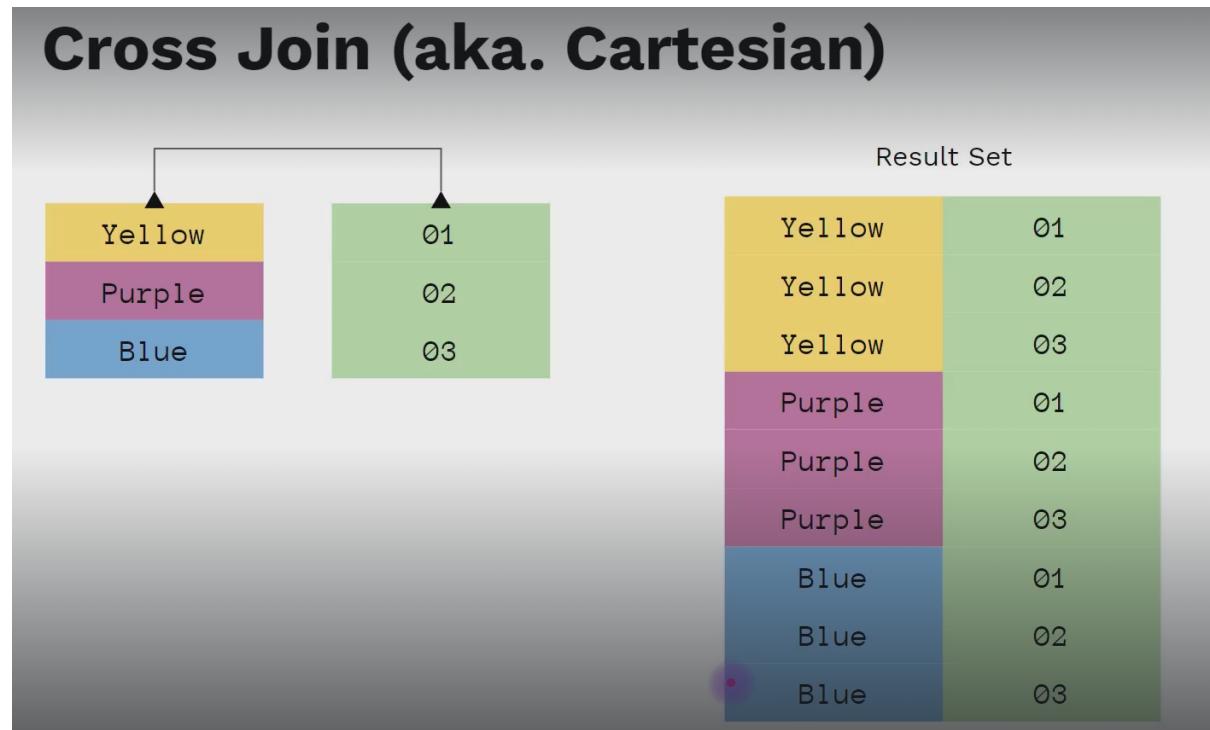
#Usage of INNER JOIN and LEFT JOIN depends on the situation.

```

Lesson 6: CROSS JOIN

-CROSS JOIN (หรือ Cartesian Product) เหนือองการคูณกัน ตารางช้ายมี 3 แถว ขวามี 3 แถว CROSS JOIN กันจะได้ $3 \times 3 = 9$ แถวหนึ่ง

*ไม่จำเป็นต้องมี Primary Key หรือ Foreign Key



```
#Create two tables: ranks and suits to prepare for a CROSS JOIN
#Reference: https://www.sqlitetutorial.net/sqlite-cross-join/
CREATE TABLE ranks (
    rank TEXT NOT NULL
);

CREATE TABLE suits (
    suit TEXT NOT NULL
);

INSERT INTO ranks(rank)
VALUES('2'),('3'),('4'),('5'),('6'),('7'),('8'),('9'),('10'),('J'),('Q'),('K'),('A');

INSERT INTO suits(suit)
VALUES('Clubs'),('Diamonds'),('Hearts'),('Spades');
```

```
#CROSS JOIN both tables to create a full card deck
SELECT * FROM ranks
CROSS JOIN suits ORDER BY suit;
```

Lesson 7: SELF JOIN

- เราสามารถ JOIN TABLE กับตัวมันเองได้ เรียกว่า SELF JOIN ใช้ในกรณีการทำ Hierachy เช่นหัวหน้า-ลูกน้อง หรือประเภทสินค้า-สินค้า

Self Join

Table can join itself (self-join)

ID	Name	REPORT_TO
1	Mary	
2	John	1
3	Anna	1
4	David	1

```
graph TD; Mary[Mary] --- John[John]; Mary --- Anna[Anna]; Mary --- David[David]
```

The diagram illustrates a self-join relationship. A woman character labeled "Mary" is shown at the top. Three lines descend from her to three smaller characters below her, labeled "John", "Anna", and "David" respectively. This visualizes how the table "employee" is joined with itself where the "manager_id" column points back to the "id" column of the same table.

```
#Create a new employee table
CREATE TABLE employee (
    id INT,
    name TEXT,
    level TEXT,
    manager_id INT
);

INSERT INTO employee VALUES
    (1, 'David', 'CEO', NULL),
    (2, 'John', 'SVP', 1),
    (3, 'Mary', 'VP', 2),
    (4, 'Adam', 'VP', 2),
    (5, 'Scott', 'Manager', 3),
    (6, 'Louise', 'Manager', 3),
    (7, 'Kevin', 'Manager', 4),
    (8, 'Takeshi', 'Manager', 4),
    (9, 'Joe', 'AM', 6),
    (10, 'Anna', 'AM', 7);
```

```
##self join in action
SELECT
    e1.name Staff,
    e1.level Staff_Level,
    e2.name Manager,
    e2.level Manager_Level,
```

```
e1.name || 'reports to' || e2.name AS comment  
FROM employee e1, employee e2  
WHERE e1.manager_id = e2.id;
```

Lesson 8: Intersect and Except

-Intersect = Return *distinct* rows in both tables (คืนค่าแผลกที่มีค่าไม่ซ้ำกันที่อยู่ในทั้งสองตาราง)

-Except = Return *distinct* rows in the first table, not represented in the second table (คืนค่าแผลกที่มีค่าไม่ซ้ำกันที่อยู่ในตารางแรก และไม่อยู่ในตารางที่สอง)

Intersect & Except

- **Intersect** returns *distinct* rows in both tables
- **Except** returns *distinct* rows in the first table, not presented in the second table



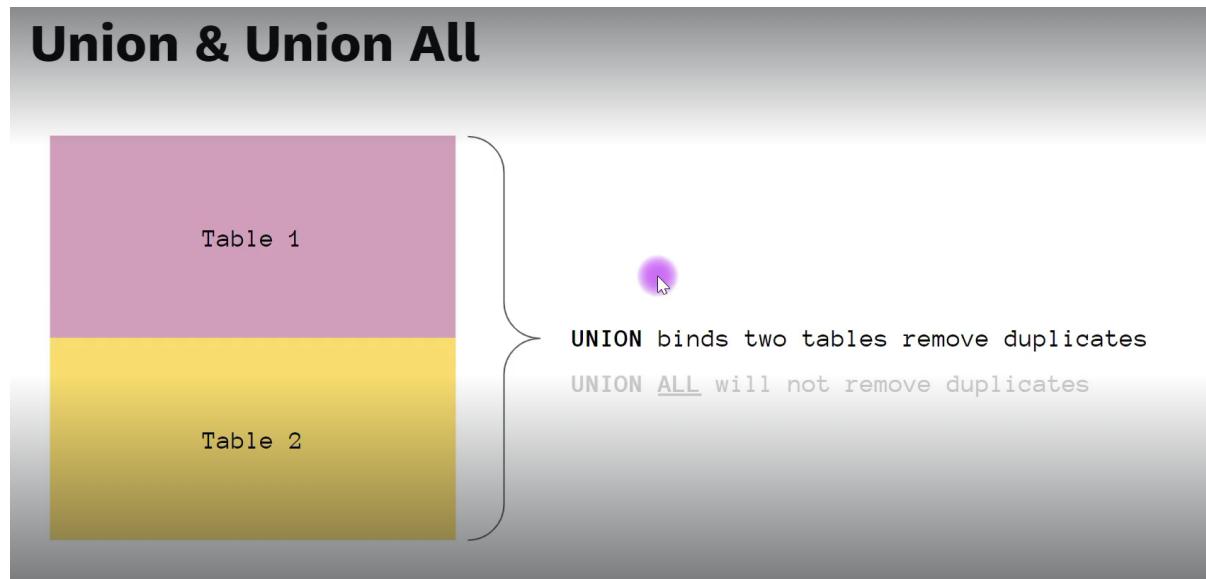
*สร้างตาราง book_shop องจาก Lesson 4 ก่อน

```
#Intersect = Which books are in both tables?  
SELECT id FROM book_shop  
INTERSECT  
SELECT id FROM favourite_book; #id shown are id in both tables  
  
#Except = Which books are in the left table, but not in the right tables?  
SELECT id FROM book_shop  
EXCEPT  
SELECT id FROM favourite_book; #id shown are id in only the left table
```

Lesson 9: UNION

-UNION จะรวม 2 ตารางเข้าด้วยกัน และลบค่าซ้ำออก

-UNION ALL จะรวม 2 ตารางเข้าด้วยกัน แต่จะไม่ลบค่าซ้ำออก



-ตัวอย่าง UNION:

```
SELECT * FROM Table1
UNION
SELECT * FROM Table2;
```

-ตัวอย่างเพิ่มเติมของการ UNION:

```
#Create a new book shop table
CREATE TABLE book_shop_new (
    id INT,
    name TEXT,
    release_year INT
);

INSERT INTO book_shop_new VALUES
(6, 'Business Data Science', 2020),
(7, 'Subliminal', 2018),
(8, 'Good Strategy Bad Strategy', 2015);
```

```
#UNION old and new book shop, then sorted by year
SELECT * FROM book_shop
UNION ALL #UNION with duplicates
```

```
SELECT * FROM book_shop_new  
ORDER BY 3 DESC;
```

Lesson 10: Intro to Subqueries

-Subqueries คือการเขียน Nested Queries (การเขียน Query ช้อน Query) โดยที่ SELECT ในวงเล็บจะเรียกว่า Inner Query และระบบจะรับ Inner Query ก่อนเสมอ

Intro to Subqueries

Subquery is a technique to write **nested** SELECT



-ตัวอย่างการเขียน Subqueries:

```
#Basic subqueries in WHERE clause  
SELECT * FROM tracks  
WHERE milliseconds = (SELECT max(milliseconds)  
                      FROM tracks); #INNER QUERY  
#Find the longest track in a table  
  
SELECT firstname, lastname, country FROM  
(SELECT * FROM customers WHERE country = 'USA'); #INNER QUERY  
#Find Name, Surname, and Country of the customers that lives in USA from customer table
```