



SQL for Data Analyst 103

➤ Parent item	👤 <u>OnDemand Class</u>
⚙ Status	Success



Table of Contents



💡 Table of Contents 💡

JOIN using WHERE clause

Convert WHERE to INNER JOIN

Review JOIN Concepts

Review CREATE TABLE

INNER vs. LEFT JOIN

CROSS JOIN (aka. Cartesian)

SELF JOIN

Intersect and Except

UNION

Intro to Subqueries

JOIN using WHERE clause

การ JOIN TABLE ต้องใช้ Primary Key = Foreign Key

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


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

```
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%';
```

Convert WHERE to INNER JOIN

Both queries give the same result.



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

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

เปลี่ยนจาก WHERE เป็น JOIN 1 ทาร

```
#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%';

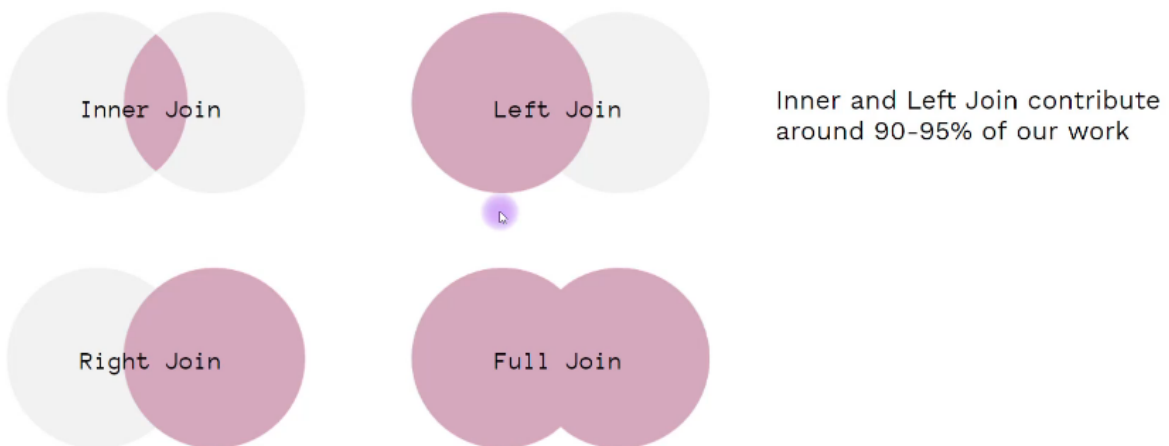
#INNER JOIN
SELECT
    A.ArtistId,
    A.Name Artist_Name,
    B.Title Album_Name
FROM artists A INNER JOIN albums B
ON A.ArtistId = B.ArtistId
WHERE A.Name LIKE 'C%';
```

JOIN ทาร

```
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%';
```

Review JOIN Concepts

Review Join Types



INNER JOIN และ **LEFT JOIN** ใช้เยอะและบ่อยมาก 90-95% ของงานเลย

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

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

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 แล้วเขียน **LEFT JOIN**

Review CREATE TABLE

```
#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);
```

INNER vs. LEFT JOIN

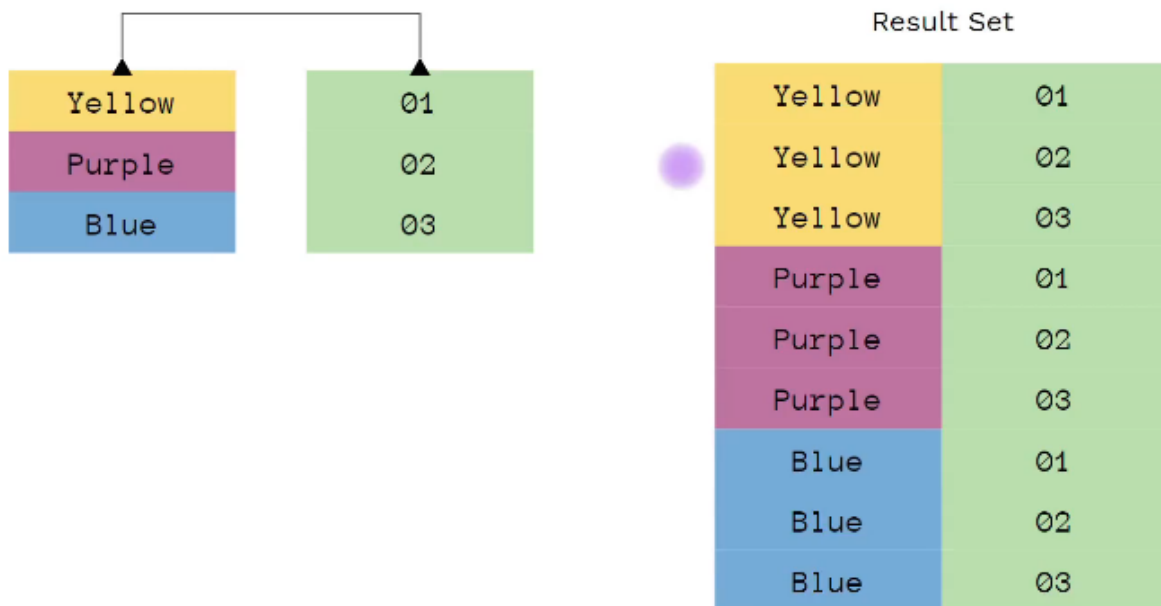
```
#INNER JOIN  
SELECT * FROM book_shop A  
INNER JOIN favourite_book B  
ON A.id = B.id;
```

```
#LEFT JOIN  
SELECT * FROM book_shop A  
LEFT JOIN favourite_book B  
ON A.id = B.id;
```

#INNER JOIN and LEFT JOIN จะใช้ตัวไหนขึ้นอยู่กับสถานการณ์และปัญหาที่ต้องการจะ

CROSS JOIN (aka. Cartesian)

Cross Join (aka. Cartesian)



CROSS JOIN (หรือ Cartesian Product) เหมือนการคูณกัน ตารางซ้ายมี 3 แถว ขวามี 3 แถว CROSS JOIN กันจะได้ $3 \times 3 = 9$ แถวนั่นเอง

ไม่จำเป็นต้องมี Primary Key หรือ Foreign Key

ตัวอย่างเช่น การทำสำรับไพ่

```
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'),  
  
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;
```

หรือ

```
SELECT * FROM ranks,suits ORDER BY suit;
```

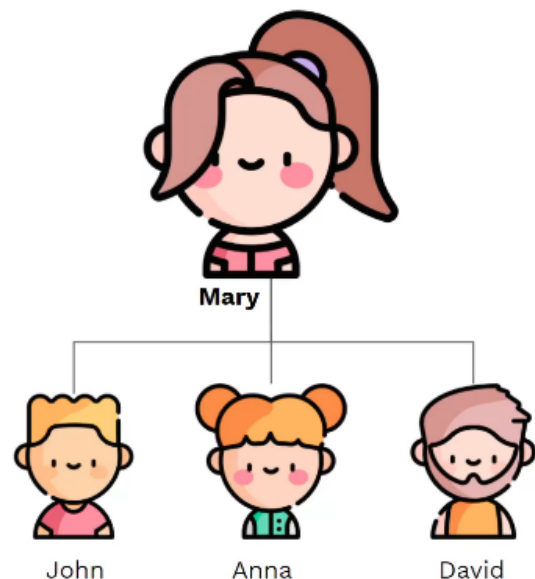
SELF JOIN

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



```
#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;

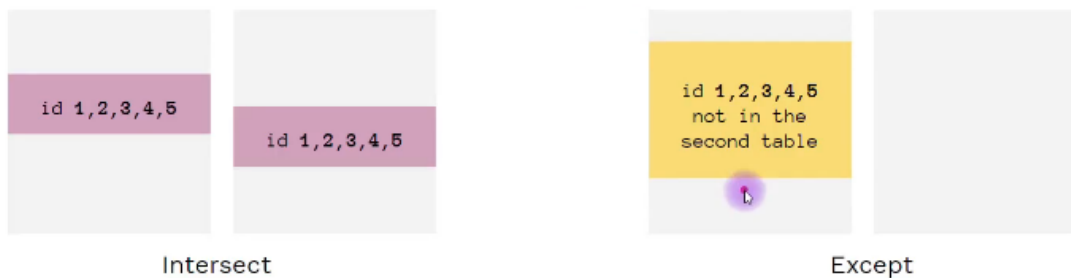
```

Intersect and Except

- Intersect = คำนวณค่าข้อมูลที่มีค่าซ้ำกันที่อยู่ในทั้งสองตาราง
- Except = คำนวณค่าข้อมูลที่มีแค่ในตารางแรก ที่ไม่ซ้ำกับข้อมูลในตารางที่สองและไม่อยู่ในตารางที่สอง

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 อิงจาก Lesson 4 ก่อน

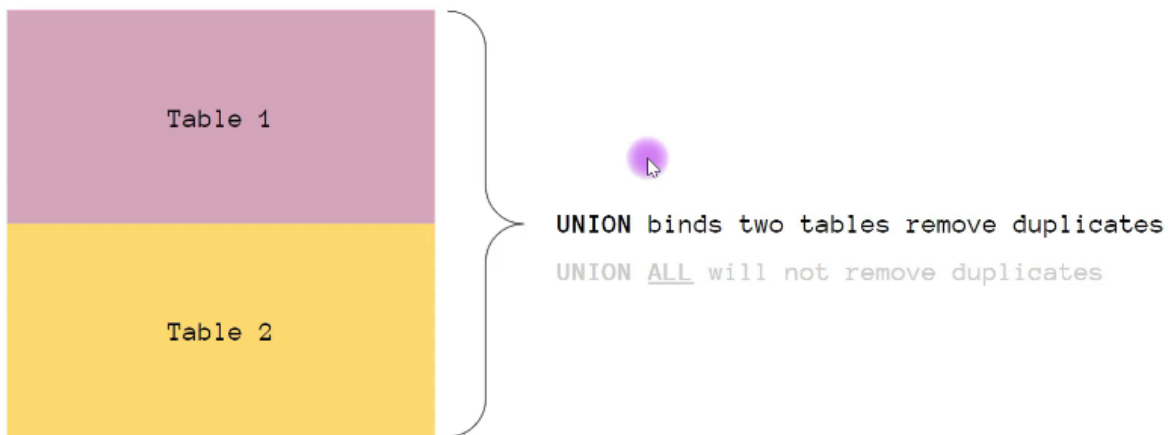
```
#Intersect = Which books are in both tables?
SELECT id FROM book_shop
INTERSECT
SELECT id FROM favourite_book; #id shown are id in both table

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

UNION

- **UNION** จะรวม 2 ตารางเข้าด้วยกัน และลบค่าซ้ำออก
- **UNION ALL** จะรวม 2 ตารางเข้าด้วยกัน แต่จะไม่ลบค่าซ้ำออก

Union & Union All



#ตัวอย่าง

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

ตัวอย่างเพิ่มเติม

#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

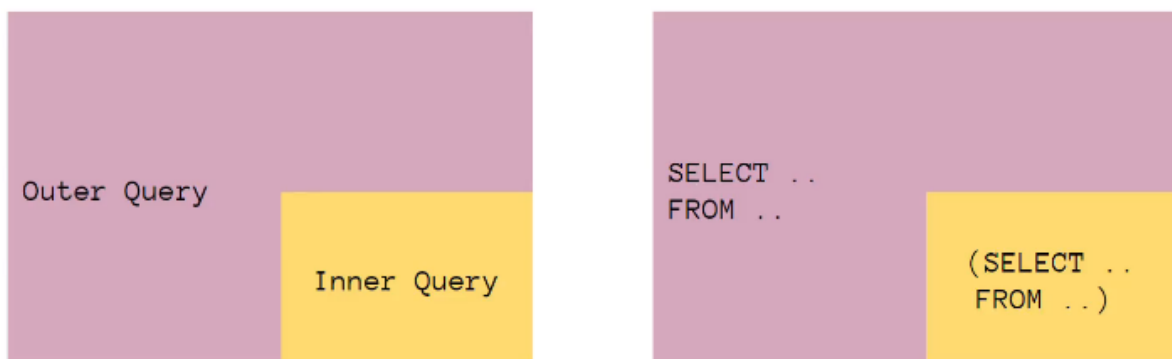
```
SELECT * FROM book_shop
UNION ALL #UNION with duplicates
SELECT * FROM book_shop_new
ORDER BY 3 DESC;
```

Intro to Subqueries

Subqueries คือการเขียน Nested Queries (การเขียน Query ซ้อน Query) โดยที่ SELECT ในวงเล็บจะเรียกว่า Inner Query และระบบจะรัน Inner Query ก่อนเสมอ

Intro to Subqueries

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



#Basic subqueries in WHERE clause

```
SELECT * FROM tracks
```

```
WHERE milliseconds = (SELECT max(milliseconds) FROM tracks);
```

#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