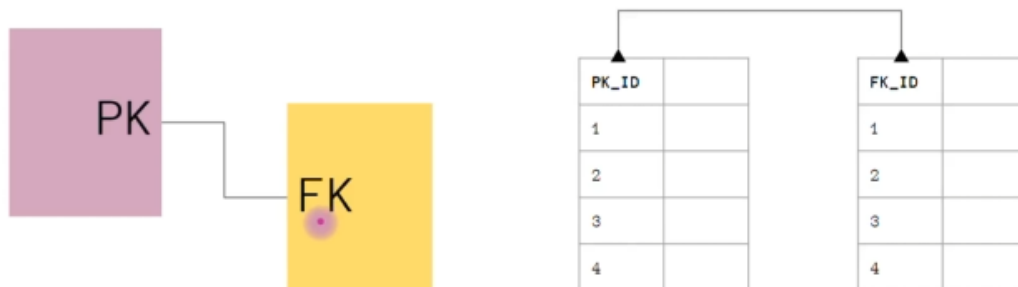


SQL for Data Analyst 103

Why we can join tables?

Table are joined using primary and foreign keys



สีเหลืองเป็น Primary key เป็นข้อมูล unique

หาก Primary key ไปโพล่ Table อื่นจะกลายเป็น Foreign key

ตัวอย่าง ดึงข้อมูลของ artists โดยให้ A = artistsid ,B = Title เวลาในการเขียน code

กำหนดเงื่อนไขสร้าง Column ใช้ชื่อใหม่ Name = artisName ,Title = albumName

และค้นหาชื่อศิลปินขึ้นต้นด้วยตัว C

chinook.db

chinook.db.9

```

1 SELECT
2   A.artistid,
3   A.Name artisName,
4   B.title albumName
5 FROM artists A, albums B
6 WHERE A.artistid = B.artistid AND A.name LIKE 'C%';

```

| i | Artistid | artisName | albumName |
|----|----------|------------------------------|--|
| 16 | | Caetano Veloso | Prenda Minha |
| 16 | | Caetano Veloso | Sozinho Remix Ao Vivo |
| 17 | | Chico Buarque | Minha Historia |
| 18 | | Chico Science & Nação Zumbi | Afrociberdella |
| 18 | | Chico Science & Nação Zumbi | Da Lama Ao Caos |
| 19 | | Cidade Negra | Acústico MTV [Live] |
| 19 | | Cidade Negra | Cidade Negra - Hits |
| 20 | | Cláudio Zoli | Na Pista |
| 76 | | Creedence Clearwater Revival | Chronicle, Vol. 1 |
| 76 | | Creedence Clearwater Revival | Chronicle, Vol. 2 |
| 77 | | Cássia Eller | Cássia Eller - Coleção Sem Limite [Disc 2] |

```

SELECT
    A.artistid,
    A.Name artisName,
    B.title albumName
FROM artists A, albums B
WHERE A.artistid = B.artistid AND A.name LIKE 'C%';

```

Convert WHERE to INNER JOIN

เขียนโดยใช้ INNER



chinook.db



chinook.db.9

```
1 SELECT
2   A.artistid,
3   A.Name artisName,
4   B.title albumName
5 FROM artists A INNER JOIN albums B
6 ON A.artistid = B.artistid
7 WHERE A.name LIKE 'C%';
```


| Artistid | artisName | albumName |
|----------|-----------------------------|-----------------------|
| 16 | Caetano Veloso | Prenda Minha |
| 16 | Caetano Veloso | Sozinho Remix Ao Vivo |
| 17 | Chico Buarque | Minha Historia |
| 18 | Chico Science & Nação Zumbi | Afrociberdelia |

```


SELECT
    A.artistid,
    A.Name artisName,
    B.title albumName
FROM artists A INNER JOIN albums B
ON A.artistid = B.artistid
WHERE A.name LIKE 'C%';

```


Filter Track





chinook.db



chinook.db.9







```
1 SELECT
2   A.artistid,
3   A.Name artisName,
4   B.title albumName,
5   C.Name trackName
6 FROM artists A
7 INNER JOIN albums B ON A.artistid = B.artistid
8 INNER JOIN tracks C ON B.albumid = C.albumid
9 WHERE A.name LIKE 'C%';
```

| i | Artistid | artisName | albumName | trackName |
|----|----------|----------------|--------------|--------------------------|
| 16 | | Caetano Veloso | Prenda Minha | Black Sabbath |
| 16 | | Caetano Veloso | Prenda Minha | The Wizard |
| 16 | | Caetano Veloso | Prenda Minha | Behind The Wall Of Sleep |
| 16 | | Caetano Veloso | Prenda Minha | N.I.B. |
| 16 | | Caetano Veloso | Prenda Minha | Evil Woman |
| 16 | | Caetano Veloso | Prenda Minha | Sleeping Village |

```

SELECT
    A.artistid,
    A.Name artisName,
    B.title albumName,
    C.Name trackName
FROM artists A
INNER JOIN albums B ON A.artistid = B.artistid
INNER JOIN tracks C ON B.artistid = C.albumid
WHERE A.name LIKE 'C%';

```

เพิ่มเงื่อนไข ถึงข้อมูลเพราะของศิลปิน Aerosmith

```

SELECT
    A.artistid,
    A.Name artisName,
    B.title albumName,
    C.Name trackName
FROM artists A
INNER JOIN albums B ON A.artistid = B.artistid
INNER JOIN tracks C ON B.artistid = C.albumid
WHERE A.name = 'Aerosmith';

```

เขียนโดนใช้ Count ถึงข้อมูลว่า Aerosimth ออกมาแล้วกี่เพลง



```
SELECT
  Count(*) Aerosmith_Songs
FROM artists A
INNER JOIN albums B ON A.artistid = B.artistid
INNER JOIN tracks c on B.albumid = C.albumid
WHERE A.Name = 'Aerosmith';
```

Review JOIN Concepts

Review Join Types



Inner and Left Join contribute
around 90-95% of our work



การ Join มีรูปแบบการจอยทั้ง 4 แบบดังนี้

- Inner Join
- Left join
- Right Join
- Full join

Inner Join

Inner Join

| Table 1 | | Table 2 | | Result Set | | |
|---------|-------|---------|----------|------------|-------|-------|
| PK_ID | Name | FK_ID | Major | PK_ID | Name | Major |
| 1 | David | 1 | Econ | 1 | David | Econ |
| 2 | John | 2 | Econ | 2 | John | Econ |
| 3 | Marry | 5 | Data | 5 | Kevin | Data |
| 4 | Anna | 12 | Engineer | | | |
| 5 | Kevin | 35 | Mkt | | | |

Left Join

Left Join

| Table 1 | | Table 2 | | Result Set | | |
|---------|-------|---------|----------|------------|-------|-------|
| PK_ID | Name | FK_ID | Major | PK_ID | Name | Major |
| 1 | David | 1 | Econ | 1 | David | Econ |
| 2 | John | 2 | Econ | 2 | John | Econ |
| 3 | Marry | 5 | Data | 3 | Marry | NULL |
| 4 | Anna | 12 | Engineer | 4 | Anna | NULL |
| 5 | Kevin | 35 | Mkt | 5 | Kevin | Data |

Full Join (ไม่ค่อยได้ใช้)

Full Join

| Table 1 | | Table 2 | | Result Set | | |
|---------|-------|---------|----------|------------|-------|----------|
| PK_ID | Name | FK_ID | Major | PK_ID | Name | Major |
| 1 | David | 1 | Econ | 1 | David | Econ |
| 2 | John | 2 | Econ | 2 | John | Econ |
| 3 | Marry | 5 | Data | 3 | Mary | NULL |
| 4 | Anna | 12 | Engineer | 4 | Anna | NULL |
| 5 | Kevin | 35 | Mkt | 5 | Kevin | Data |
| | | | | 12 | NULL | Engineer |
| | | | | 35 | NULL | Mkt |

Right Join (ไม่ค่อยได้ใช้)

เขียนแบบ Left Join แล่สลับตำแหน่งกัน

Review CREATE TABLE

การสร้าง Table และ insert ข้อมูลเข้าไป

The screenshot shows a SQLite database interface. On the left, a sidebar lists databases: SQLite (0.0.4 beta), MariaDB, PostgreSQL, and MS SQL. Under the SQLite database, two tables are listed: 'book_shop' and 'favourite_book'. The 'book_shop' table has columns: id (INT), name (TEXT), and release_year (INT). The 'favourite_book' table has columns: id (INT), author (TEXT), and reviews (REAL). The main area displays the SQL code for creating these tables and inserting data. The code is as follows:

```
1 CREATE TABLE book_shop (  
2   id INT,  
3   name TEXT,  
4   release_year INT  
5 );  
6  
7 CREATE TABLE favourite_book (  
8   id INT,  
9   author TEXT,  
10  reviews REAL  
11 );  
12  
13 INSERT INTO book_shop VALUES  
14 (1, 'Think Like A Freak' , 2014),  
15 (2, 'Ultralearning' , 2019),  
16 (3, 'Blue Ocean Strategy', 2015),  
17 (4, 'The Power of Habit', 2012),  
18 (5, 'Outline', 2008);  
19  
20 INSERT INTO favourite_book VALUES  
21 (1, 'Steven D. Levitt, Stephen J. Dubner' , 1904),  
22 (4, 'Charles Duhigg' , 12007),  
23 (5, 'Malcolm Gladwell', 12063);
```

Below the SQL code, a table view shows the data inserted into the 'book_shop' table:

| i | id | author | reviews |
|---|----|-------------------------------------|---------|
| 1 | | Steven D. Levitt, Stephen J. Dubner | 1904 |
| 4 | | Charles Duhigg | 12007 |
| 5 | | Malcolm Gladwell | 12063 |

```
CREATE TABLE book_shop (  
    id INT,  
    name TEXT,  
    release_year INT  
);  
  
CREATE TABLE favourite_book (  
    id INT,  
    author TEXT,  
    reviews REAL  
);  
  
INSERT INTO book_shop VALUES  
    (1, 'Think Like A Freak' , 2014),  
    (2, 'Ultralearning' , 2019),
```



```
(3, 'Blue Ocean Strategy', 2015),
(4, 'The Power of Habit', 2012),
(5, 'Outline', 2008);
```

```
INSERT INTO favourite_book VALUES
(1, 'Steven D. Levitt, Stephen J. Dubner' , 1904),
(4, 'Charles Duhigg' , 12007),
(3, 'Malcolm Gladwell', 12063);
```

Inner Join

SQLite

SQLite.1

SQLite.2

SQLite.3

```
1 -- inner join
2 SELECT *FROM book_shop A
3 INNER JOIN favourite_book B ON A.id = B.id;
4
```

| id | name | release_year | id | author | reviews |
|----|--------------------|--------------|----|------------------------------|---------|
| 1 | Think Like A Freak | 2014 | 1 | Steven D. Levitt, Stephen... | 1904 |
| 4 | The Power of Habit | 2012 | 4 | Charles Duhigg | 12007 |
| 5 | Outline | 2008 | 5 | Malcolm Gladwell | 12063 |

```
-- inner join
SELECT *FROM book_shop A
Inner JOIN favourite_book B ON A.id = B.id;
```

Left Join

| SQLite | | | | | | |
|---|----|--------------------|--------------|----|------------------------------|---------|
| SQLite.1 | | | | | | |
| SQLite.2 | | | | | | |
| SQLite.3 | | | | | | |
| <pre> 1 -- left join 2 SELECT *FROM book_shop A 3 LEFT JOIN favourite_book B ON A.id = B.id; </pre> | | | | | | |
| i | id | name | release_year | id | author | reviews |
| | 1 | Think Like A Freak | 2014 | 1 | Steven D. Levitt, Stephen... | 1904 |
| | 4 | The Power of Habit | 2012 | 4 | Charles Duhigg | 12007 |
| | 5 | Outline | 2008 | 5 | Malcolm Gladwell | 12063 |

```

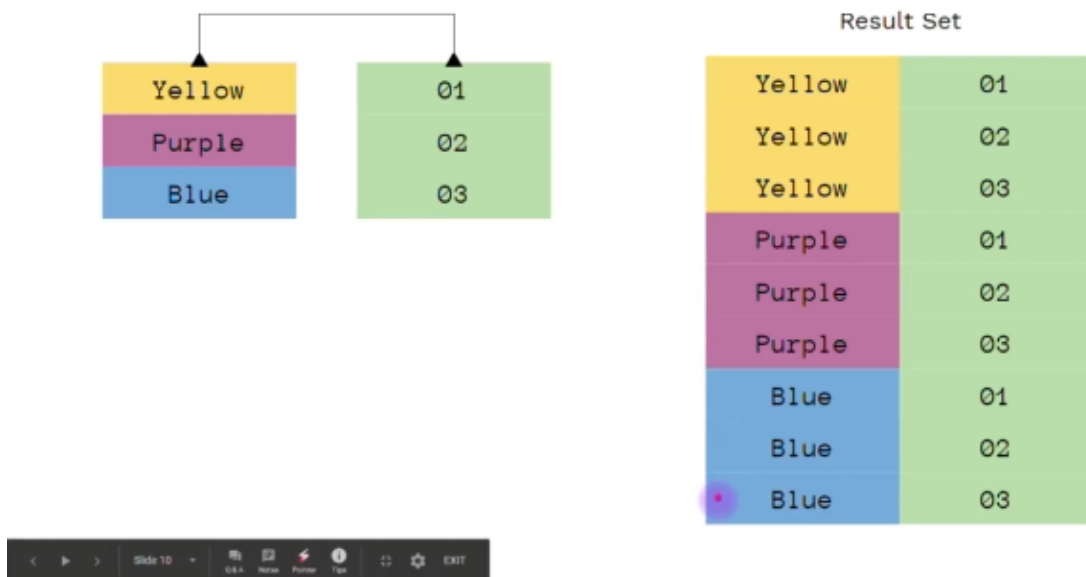
SELECT *FROM book_shop A
LEFT JOIN favourite_book B ON A.id = B.id;

```

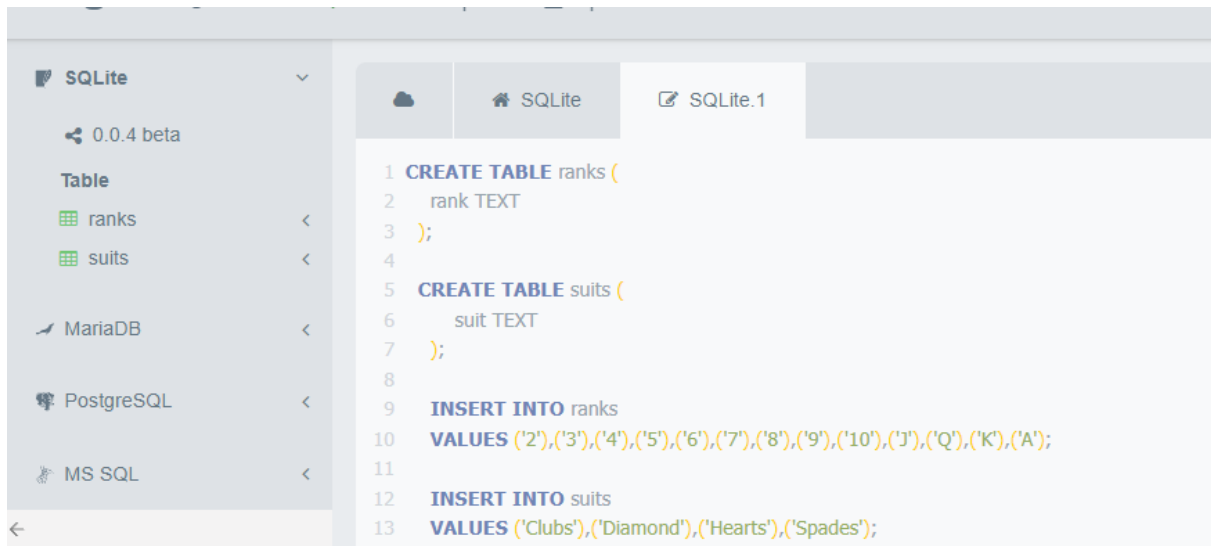
Cross Join

Cross Join ไม่จำเป็นต้องมี Primary Key หรือ Foreign key
เพราะจะนำทุก Record มา Join กัน

Cross Join (aka. Cartesian)



สร้าง Table



```
CREATE TABLE ranks (  
    rank TEXT  
);  
  
CREATE TABLE suits (  
    suit TEXT  
);  
  
INSERT INTO ranks  
VALUES ('2'),('3'),('4'),('5'),('6'),('7'),('8'),('9'),('10'),('J'),('Q'),('K'),('A');  
  
INSERT INTO suits  
VALUES ('Clubs'),('Diamond'),('Hearts'),('Spades');
```

ดึงข้อมูลขึ้นมา 2 Table

```
SELECT * FROM ranks, suits;
```

| SQLite | | SQLite.5 |
|--------------------------------------|---------|----------|
| 1 SELECT * FROM ranks, suits; | | |
| rank | suit | |
| 2 | Clubs | |
| 2 | Diamond | |
| 2 | Hearts | |
| 2 | Spades | |

ดึงข้อมูลแบบ Cross Join

| SQLite | | SQLite.5 |
|--|-------|----------|
| 1 SELECT * FROM ranks CROSS JOIN suits ORDER BY suit; | | |
| rank | suit | |
| 2 | Clubs | |
| 3 | Clubs | |

```
SELECT * FROM ranks CROSS JOIN suits ORDER BY suit;
```

สามารถย่อ Query ตัว Cross Join เป็น , และ Suit เป็น 2 ได้ (Suit อยู่ใน Column ที่2)
ได้ผลลัพธ์เหมือนกัน

```
SELECT * FROM ranks , suits ORDER BY 2;
```

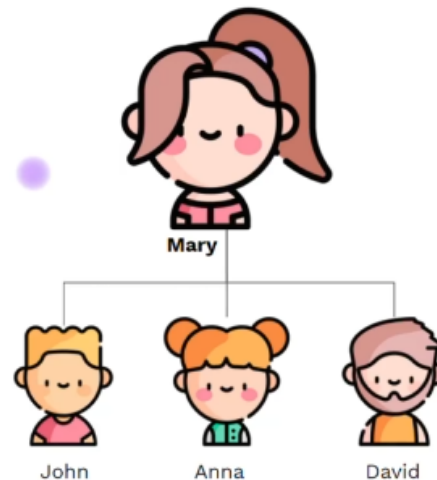
SELF JOIN

การ Join ใน Table มันเอง

Self Join

Table can join itself (self-join)

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



CREATE TABLE employee

```
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', 4),  
    (7, 'Kevin', 'Manager', 4),  
    (8, 'Takeshi', 'Manager', 4),
```

```
(9, 'Joe', 'AM', 6),
(10, 'Anna', 'AM', 7);
```

The screenshot shows a SQLite database interface with a tab labeled 'SQLite'. The SQL editor contains the following code:

```
1 CREATE TABLE employee (
2   id INT,
3   name TEXT,
4   level TEXT,
5   manager_id INT
6 );
7
8 INSERT INTO employee VALUES
9   (1, 'David', 'CEO', NULL),
10  (2, 'John', 'SVP', 1),
11  (3, 'Mary', 'VP', 2),
12  (4, 'Adam', 'VP', 2),
13  (5, 'Scott', 'Manager', 3),
14  (6, 'Louise', 'Manager', 4),
15  (7, 'Kevin', 'Manager', 4),
16  (8, 'Takeshi', 'Manager', 4),
17  (9, 'Joe', 'AM', 6),
18  (10, 'Anna', 'AM', 7);
```

Below the editor, a table view displays the data inserted into the 'employee' table:

| i | id | name | level | manager_id |
|---|----|-------|-------|------------|
| 1 | 1 | David | CEO | NULL |
| 2 | 2 | John | SVP | 1 |
| 3 | 3 | Mary | VP | 2 |
| 4 | 4 | Adam | VP | 2 |

ลองดึงข้อมูลขึ้นมา

The screenshot shows the same SQLite database interface, but the SQL editor now contains the query:

```
1 SELECT * FROM employee;
```

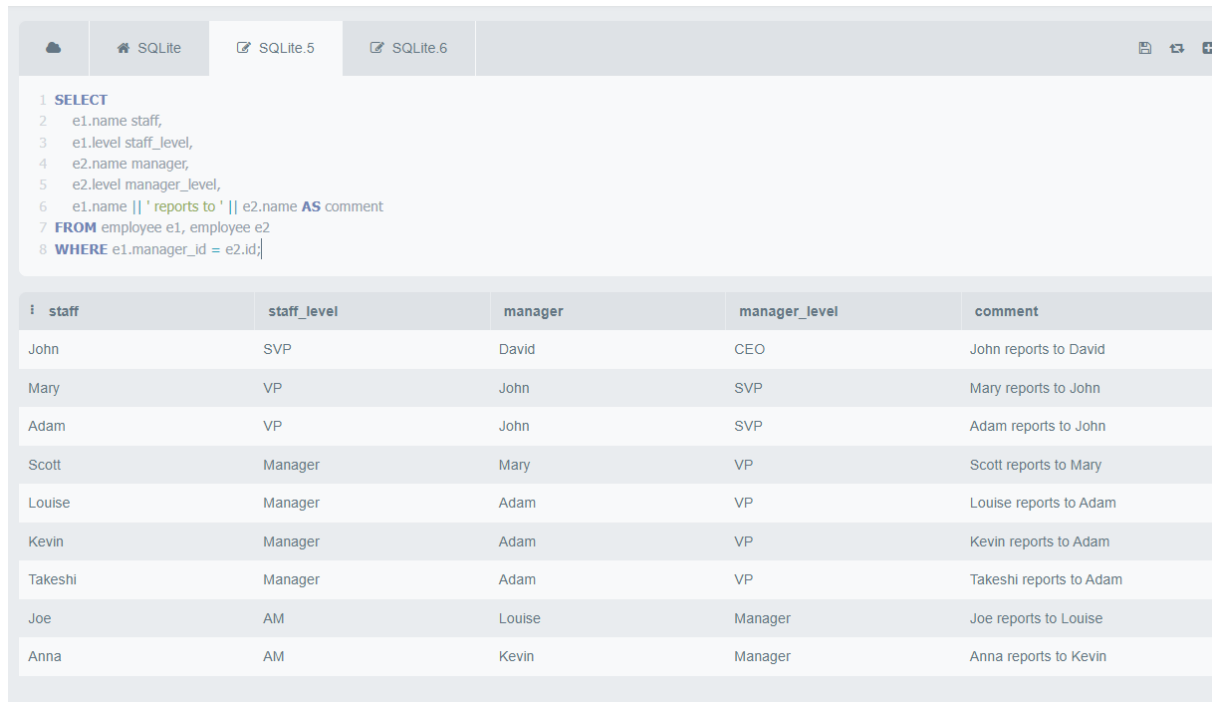
Below the editor, a table view displays the results of the query, showing all 10 rows of the 'employee' table:

| i | id | name | level | manager_id |
|----|----|---------|---------|------------|
| 1 | 1 | David | CEO | NULL |
| 2 | 2 | John | SVP | 1 |
| 3 | 3 | Mary | VP | 2 |
| 4 | 4 | Adam | VP | 2 |
| 5 | 5 | Scott | Manager | 3 |
| 6 | 6 | Louise | Manager | 4 |
| 7 | 7 | Kevin | Manager | 4 |
| 8 | 8 | Takeshi | Manager | 4 |
| 9 | 9 | Joe | AM | 6 |
| 10 | 10 | Anna | AM | 7 |

```
SELECT * FROM employee;
```

ต้องการดึงข้อมูลชื่อ Staff และ Manager ขึ้นมา

และสร้าง Column comment ขึ้นมาใหม่ว่า Staff ต้อง Report ให้ Manager คนนั้น



The screenshot shows a SQLite database interface with a query editor and a results table. The query is a JOIN between two employee tables, e1 and e2, where e1 is the staff and e2 is the manager. The results table has five columns: staff, staff_level, manager, manager_level, and comment. The comment column contains text indicating the reporting relationship, such as 'John reports to David'.

```
1 SELECT
2   e1.name staff,
3   e1.level staff_level,
4   e2.name manager,
5   e2.level manager_level,
6   e1.name || ' reports to ' || e2.name AS comment
7 FROM employee e1, employee e2
8 WHERE e1.manager_id = e2.id;
```

| i | staff | staff_level | manager | manager_level | comment |
|---|---------|-------------|---------|---------------|-------------------------|
| | John | SVP | David | CEO | John reports to David |
| | Mary | VP | John | SVP | Mary reports to John |
| | Adam | VP | John | SVP | Adam reports to John |
| | Scott | Manager | Mary | VP | Scott reports to Mary |
| | Louise | Manager | Adam | VP | Louise reports to Adam |
| | Kevin | Manager | Adam | VP | Kevin reports to Adam |
| | Takeshi | Manager | Adam | VP | Takeshi reports to Adam |
| | Joe | AM | Louise | Manager | Joe reports to Louise |
| | Anna | AM | Kevin | Manager | Anna reports to Kevin |

```
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 จะ return row อย่างเดียว

จะไม่สร้างเป็น column ใหม่

Intersect & Except

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



INTERSECT

ตัวอย่าง Query แล้วจะพบว่า id 1 3 4 จะอยู่ทั้ง 2 Table

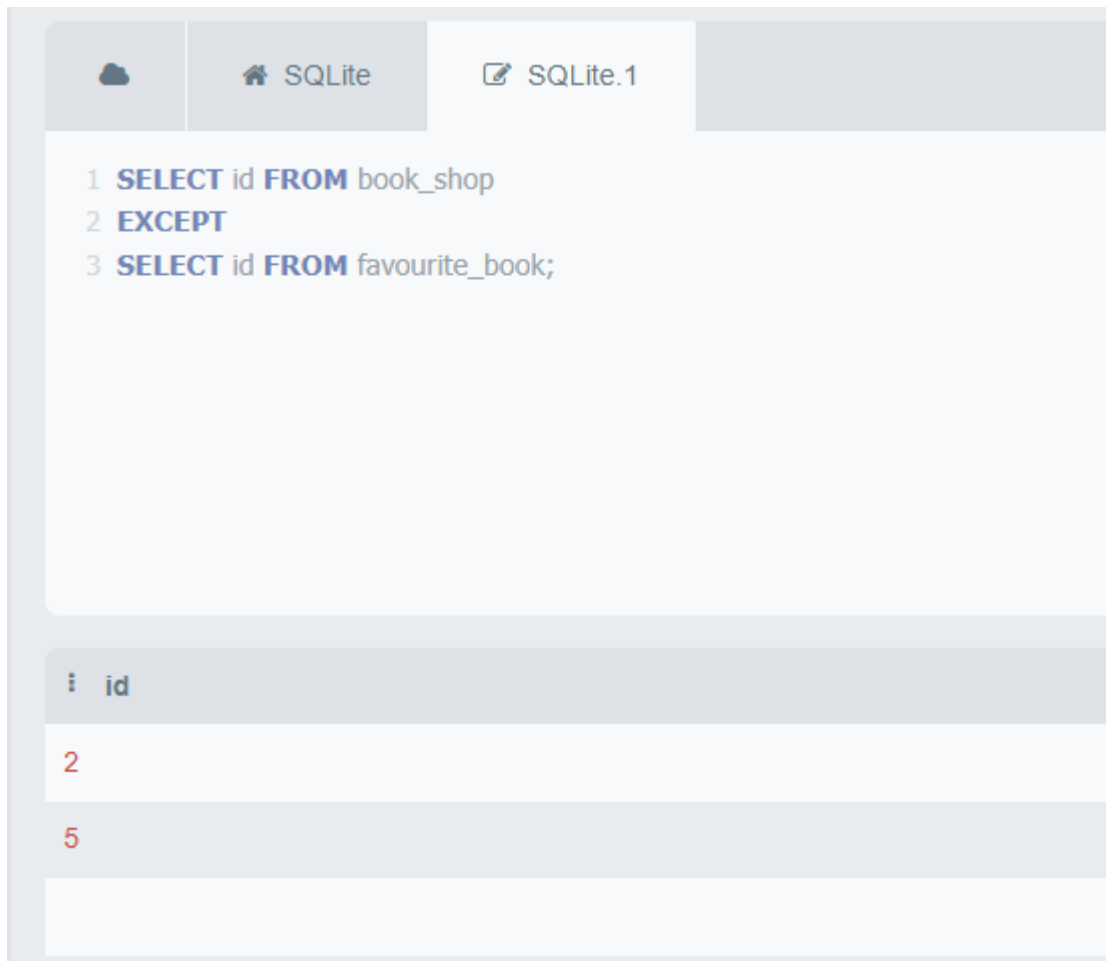
```
SQLite SQLite.1
1 SELECT id FROM book_shop
2 INTERSECT
3 SELECT id FROM favourite_book;
```

| id |
|----|
| 1 |
| 3 |
| 4 |


```
SELECT id FROM book_shop
INTERSECT
SELECT id FROM favourite_book;
```

EXCEPT

ดึงข้อมูลจาก Book_shop ที่ไม่มี id ใน Favourite_book



The screenshot shows a SQLite IDE interface with three tabs: a cloud icon, 'SQLite', and 'SQLite.1'. The 'SQLite.1' tab is active and contains the following SQL query:

```
1 SELECT id FROM book_shop
2 EXCEPT
3 SELECT id FROM favourite_book;
```

Below the query editor, the results are displayed in a table with one column labeled 'id'. The results are:

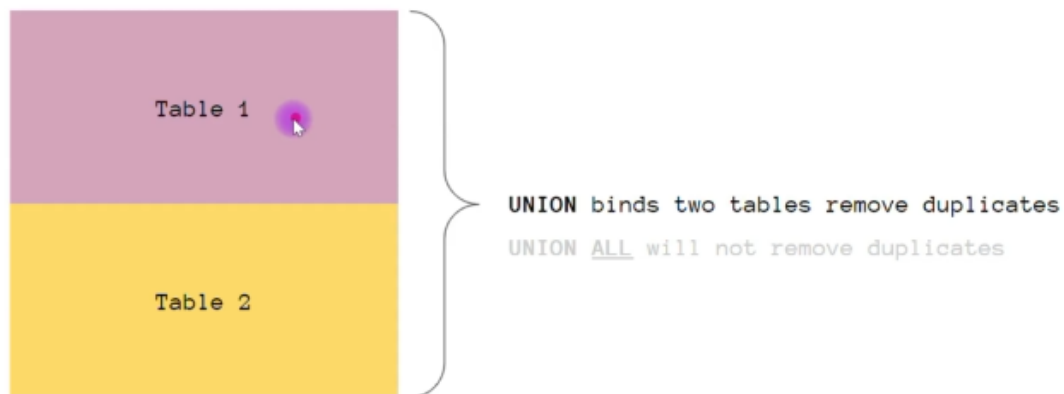
| id |
|----|
| 2 |
| 5 |

```
SELECT id FROM book_shop
EXCEPT
SELECT id FROM favourite_book;
```

Union

ดึงข้อมูลหากมี Table เหมือนกันจะดึงข้อมูลมาอันเดียว

Union & Union All



สร้าง Table ใหม่เป็น Book_shop_new

| | SQLite | SQLite.1 | SQLite.2 | SQLite.3 |
|--------------------------------|--------|----------------------------|--------------|----------|
| 1 SELECT * FROM book_shop_new; | | | | |
| i | id | name | release_year | |
| 1 | | Think Like A Freak | 2014 | |
| 6 | | Business Data Science | 2020 | |
| 7 | | Subliminal | 2018 | |
| 8 | | Good Strategy Bad Strategy | 2015 | |

```
CREATE TABLE book_shop_new (  
    id INT,  
    name TEXT,  
    release_year INT
```

```
);
```

```
INSERT INTO book_shop_new VALUES
(1, 'Think Like A Freak', 2014),
(6, 'Business Data Science', 2020),
(7, 'Subliminal', 2018),
(8, 'Good Strategy Bad Strategy', 2015);
```

| | SQLite | SQLite.1 | SQLite.2 | SQLite.3 |
|---|--------|----------------------------|----------|--------------|
| <pre>1 SELECT * FROM book_shop 2 UNION 3 SELECT * FROM book_shop_new;</pre> | | | | |
| i | id | name | | release_year |
| 1 | | Think Like A Freak | | 2014 |
| 2 | | Ultralearning | | 2019 |
| 3 | | Blue Ocean Strategy | | 2015 |
| 4 | | The Power of Habit | | 2012 |
| 5 | | Outline | | 2008 |
| 6 | | Business Data Science | | 2020 |
| 7 | | Subliminal | | 2018 |
| 8 | | Good Strategy Bad Strategy | | 2015 |

```
SELECT * FROM book_shop
UNION
SELECT * FROM book_shop_new;
```

Union All

หากมี Table ใดข้อมูลเหมือนกันจะไม่ลบออก
ดึงข้อมูลมาอยู่ดี

| | | | SQLite | SQLite.1 | SQLite.2 | SQLite.3 |
|---------------------------------------|----|----------------------------|--------------|----------|----------|----------|
| 1 SELECT * FROM book_shop | | | | | | |
| 2 UNION ALL | | | | | | |
| 3 SELECT * FROM book_shop_new; | | | | | | |
| i | id | name | release_year | | | |
| | 1 | Think Like A Freak | 2014 | | | |
| | 2 | Ultralearning | 2019 | | | |
| | 3 | Blue Ocean Strategy | 2015 | | | |
| | 4 | The Power of Habit | 2012 | | | |
| | 5 | Outline | 2008 | | | |
| | 1 | Think Like A Freak | 2014 | | | |
| | 6 | Business Data Science | 2020 | | | |
| | 7 | Subliminal | 2018 | | | |
| | 8 | Good Strategy Bad Strategy | 2015 | | | |

```
SELECT * FROM book_shop
UNION ALL
SELECT * FROM book_shop_new;
```

Intro to Subqueries

Run สีเหลืองก่อน ค่อยขยายไปด้านนอก

Intro to Subqueries

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



การเขียนปกติ

The screenshot shows a SQL query editor with the following query:

```
1 SELECT * FROM tracks
2 WHERE milliseconds = 5286953;
```

The results are displayed in a table with the following columns: TrackId, Name, AlbumId, MediaTypeId, GenreId, Composer, Milliseconds, Bytes, and UnitPrice. The results show a single row with the following values:

| TrackId | Name | AlbumId | MediaTypeId | GenreId | Composer | Milliseconds | Bytes | UnitPrice |
|---------|--------------------|---------|-------------|---------|----------|--------------|------------|-----------|
| 2820 | Occupation / Pr... | 227 | 3 | 19 | NULL | 5286953 | 1054423946 | 1.99 |

```
SELECT * FROM tracks
WHERE milliseconds = 5286953;
```

เขียน Sub Query

จะเห็นได้ว่าได้ผลลัพธ์แบบเดียวกัน

| | | | | |
|--|------------|--------------|--------------|--|
| | chinook.db | chinook.db.1 | chinook.db.2 | |
|--|------------|--------------|--------------|--|

```

1 SELECT *
2 FROM tracks
3 WHERE milliseconds = (SELECT MAX(milliseconds)
4                       FROM tracks);

```

| i | TrackId | Name | AlbumId | MediaTypeId | GenreId | Composer | Milliseconds | Bytes | UnitPrice |
|---|---------|--------------------|---------|-------------|---------|----------|--------------|------------|-----------|
| | 2820 | Occupation / Pr... | 227 | 3 | 19 | NULL | 5286953 | 1054423946 | 1.99 |

```

SELECT *
FROM tracks
WHERE milliseconds = (SELECT MAX(milliseconds)
                      FROM tracks);

```

ตัวอย่างที่ 2

| | | | | |
|--|------------|--------------|--------------|--|
| | chinook.db | chinook.db.1 | chinook.db.2 | |
|--|------------|--------------|--------------|--|

```

7 */
8 SELECT firstname, lastname, country FROM
9 (SELECT *
10  FROM customers
11  WHERE country = 'USA')
12

```

| i | FirstName | LastName | Country |
|---|-----------|------------|---------|
| | Frank | Harris | USA |
| | Jack | Smith | USA |
| | Michelle | Brooks | USA |
| | Tim | Goyer | USA |
| | Dan | Miller | USA |
| | Kathy | Chase | USA |
| | Heather | Leacock | USA |
| | John | Gordon | USA |
| | Frank | Ralston | USA |
| | Victor | Stevens | USA |
| | Richard | Cunningham | USA |
| | Patrick | Gray | USA |
| | Julia | Barnett | USA |

```
SELECT firstname, lastname, country FROM  
(SELECT *  
FROM customers  
WHERE country = 'USA')
```