SQL for Data Analysis 103

▼ Create View

```
CREATE VIEW usa_customers AS
  SELECT * FROM customers WHERE country = 'USA'

CREATE VIEW invoice_2009 AS
    SELECT * FROM invoices WHERE STRFTIME("%Y", invoicedate) = "2009"

SELECT * FROM usa_customer a
JOIN invoice_2009 b
on a.customerid = b.customerid
```

EP01. Select Data From Multiple Tables

🌷 `WHERE` ในวีดีโอนี้ได้ผลลัพธ์เหมือนการเขียน `INNER JOIN`

```
-- join tables using WHERE clause
SELECT * FROM artists, albums
WHERE artists.artistid = albums.artistid;

SELECT
    A.artistsid,
    A.name artistName
    B.title albumName
FROM artists A, albums B
WHERE A.artistsid = B.artistsid AND A.name LIKE 'C%'

-- using alias
SELECT * FROM artists A, albums B
WHERE A.artistid = B.artistid;

SELECT
    A.artistsid,
    A.name artistName
    B.title albumName
FROM artists A, albums B
WHERE A.artistsid = B.artistsid AND A.name LIKE 'C%'
```

EP02. Convert Where to Inner Join

```
-- change from WHERE to JOIN clause
SELECT * FROM artists JOIN albums
```

```
ON artists.artistid = albums.artistid;
```

```
-- find Aerosmith, note that AS is optional clause
SELECT
    A.artistid,
    A.Name  AS artistName,
    B.Title AS albumName,
    C.Name  AS trackName
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';

SELECT
    COUNT(*) Aerosmith_songs
FROM artists A
INNER JOIN albums B ON A.artistsid = B.artistsid
INNER JOIN tracks C ON B.albumid = C.albumid
WHERE A.name LIKE 'Aerosmith'
```

```
-- a little advanced query, try to read and interpret the result
SELECT
    customers.country,
    COUNT(DISTINCT customers.customerid) AS n_customers,
    COUNT(invoices.total) AS n_transactions,
    SUM(invoices.total)   AS total_revenue
FROM customers
JOIN invoices
ON customers.customerid = invoices.customerid
GROUP BY 1
ORDER BY 2 DESC;
```

EP03. Review Join Types

🪴 `JOIN` หรือ `INNER JOIN` คือ default join มาตรฐานของภาษา SQL เป็น the most popular join สำหรับงาน data analysis // SQLite ไม่รองรับ `RIGHT JOIN` และ `FULL OUTER JOIN`

```
-- join syntax is simple
SELECT *
FROM tableA
JOIN tableB ON tableA.PK = tableB.FK
JOIN tableC ON tableB.PK = tableC.FK
JOIN tableD ON tableC.PK = tableD.FK;
```

EP04. Create Table and Insert Data

```
-- create two tables
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, 'Outliers', 2008);

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

🌷 เราไม่สามารถสร้าง table ชื่อซ้ำกับ existing tables ใน database ได้ ต้อง `DROP TABLE` หรือ `ALTER TABLE` เปลี่ยนชื่อตารางไม่ให้ซ้ำกัน

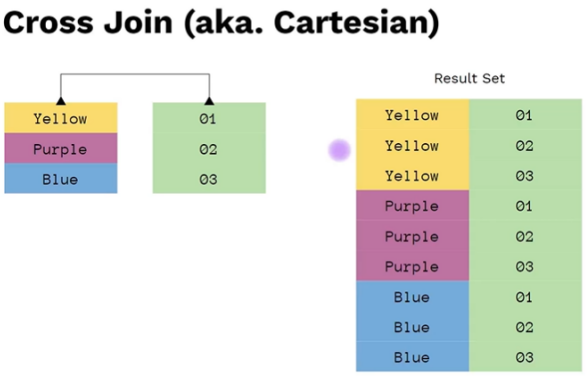
```
-- review our new tabl
SELECT * FROM book_shop;
SELECT * From favourite_book;
```

EP05. Left Join vs. Inner 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;
```

EP06. Cross Join




```
-- create two tables: ranks and suits
-- 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 table rank with table suits
SELECT * FROM ranks
CROSS JOIN suits
ORDER BY 2;
```

🪴 `CROSS JOIN` มีอีกชื่อว่า `Cartesian Product` ถ้าตารางซ้ายมี 7 แถว ตารางขวามี 5 แถว result set ของเราจะออกมาทั้งหมด 7 x 5 = 35 แถว (product)

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

EP07. Self Join

🪴 เราใช้ `SELF JOIN` เพื่อ join table กับตัวมันเอง นิยมใช้ในกรณีแบบ hierarchy เช่น manager - staff หรือ product category - items

```
-- 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
    t1.id,
    t1.name AS employeeName,
    t1.level AS employeeLevel,
    t2.name AS managerName,
    t2.level AS managerLevel
FROM employee t1, employee t2
WHERE t1.manager_id = t2.id;

SELECT
    e1.name staff,
    e1.level staff_level,
    e2.name manager,
    e2.level manager_level,
    e1.name || 'report to' || e2.name AS comment
FROM employee e1, employee e2
WHERE e1.manager_id = e2.id;
```

EP08. Intersect & Except

- `INTERSECT` return เฉพาะ distinct rows ที่มีในสองตาราง การใช้งานคล้ายๆ `INNER JOIN`
- `EXCEPT` return เฉพาะ distinct rows ในตารางด้านซ้ายที่**ไม่มี**ในตารางด้านขวา

```
-- intersect = which books are in both tables
SELECT id FROM book_shop
INTERSECT
SELECT id FROM favourite_book;

-- 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;
```

EP09. Union & Union All

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

SELECT * FROM book_shop
```

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

EP10. Intro to Subqueries

🪴 Subqueries คือเทคนิคการเขียน nested query หรือ `SELECT` ซ้อน `SELECT`

```
-- basic subqueries in WHERE clause
SELECT * FROM tracks
WHERE milliseconds = (SELECT max(milliseconds) FROM tracks);

SELECT COUNT(*) FROM tracks
WHERE milliseconds >= (SELECT AVG(milliseconds) FROM tracks);

SELECT * FROM tracks WHERE millisecond = 5286953

-- Subquery innner + outer : select two condition
-- How to write method I
SELECT sub1.firstname, sub2.total
FROM (SELECT * FROM customers WHERE country = 'USA') AS sub1
JOIN (SELECT * FROM invoices WHERE STRFTIME("%Y", invoicedate) = "2009") AS sub2
On sub1.customerid = sub2.customerid

-- How to write method II : same result
## common table expression => WITH
WITH sub1 AS (
          SELECT * FROM customers WHERE country = 'USA' ) ,
     sub2 AS (
          SELECT * FROM invoices WHERE STRFTIME("%Y", invoicedate) = "2009")
SELECT
  sub1.firstname, sub2.total
FROM sub1
JOIN sub2
ON sub1.customerid = sub2.customerid
```

```
-- another subqueries example in FROM clause
SELECT firstname, lastname, email
FROM ( SELECT * FROM customers WHERE country = 'USA');
```

🪴 Course Summary

- ตอนนี้ทุกคนสามารถบอกความแตกต่างระหว่าง `INNER` vs. `LEFT JOIN` ได้แล้ว
- เขียน `INNER JOIN` และ join ด้วย `WHERE` ได้อย่างชำนาญ
- ได้เห็นรูปแบบการ `JOIN` แบบอื่นๆทั้งแบบ `CROSS` และ `SELF` join รวมถึงการใช้งาน set operators

- คอร์สเรียนต่อไปเราจะสอนการเขียน subqueries และการใช้งาน window functions แล้วนะครับ :)