




Intro to Database and CRUD

☰ Tags	Foundation SQL
➤ Class	
☑ Finished Yet?	☑
➤ Knowledge	 <u>The Second Sprint: SQL</u>

Lesson 1: SQLite Online Editor

- ภาษา SQL (Structured Query Language) เป็นภาษาที่ใช้ในการทำงานกับ Database
- Database ประกอบด้วยหลาย Tables ในฐานข้อมูล เรียกว่า Relational Database
- ทุกคนที่อยู่ในองค์กรที่ขับเคลื่อนด้วยข้อมูล (Data-Driven Company) ควรจะรู้จักพื้นฐาน SQL เพื่อให้สามารถวิเคราะห์ข้อมูลและหา Insight จากข้อมูลที่อยู่ใน Database ขององค์กรได้

Lesson 2: Create Table

- Data Types ที่ใช้ใน SQL มี 4 ประเภทหลัก ๆ คือ

1. INT (เลขจำนวนเต็ม)

2. REAL (เลขที่เป็นจำนวนจริง หรือ Float เลขทศนิยม)

3. TEXT (ข้อความ)

4. DATE (วันที่)

-SQLite จะมองวันที่เป็น TEXT ทำให้เหลือแค่ INT, REAL, และ TEXT เป็นหลัก

-เราไม่สามารถสร้างตารางที่มีชื่อซ้ำกับตารางที่มีอยู่เดิมได้

-เราสามารถ DROP TABLE ชื่อตาราง; เพื่อลบตารางทิ้งได้ แต่เราจะเรียกตารางกลับมาไม่ได้ อีก ต้องระวังก่อนใช้คำสั่งนี้เสมอ

-Statement ของภาษา SQL จะปิดท้ายด้วย ; เสมอ

-เราสามารถสร้างตารางได้ด้วย CREATE TABLE ตามด้วยชื่อของ Table พร้อมกับระบุชื่อและประเภทของข้อมูลในแต่ละ column เช่น

```
CREATE TABLE employee (  
    id INT UNIQUE,  
    name TEXT,  
    department TEXT,  
    position TEXT,  
    salary REAL  
);
```

[สร้างตารางชื่อ employee ประกอบด้วย column id เป็นเลขจำนวนเต็มที่มีค่าไม่ซ้ำกัน, name department position เป็นข้อความ, และ salary เป็นเลขจำนวนจริง]

Lesson 3: Insert Data

-ตอนสร้างตาราง ถ้าเรากำหนด column เป็น PRIMARY KEY หรือ UNIQUE จะไม่สามารถ INSERT ข้อมูลซ้ำได้

-เราสามารถตรวจสอบข้อมูลทั้งหมดของตารางได้ด้วย SELECT * FROM ชื่อตาราง เพื่อแสดงข้อมูลทั้งหมด และเช็คได้ว่าตารางเป็นแบบที่เราต้องการหรือไม่

-white space (ช่องว่างระหว่างข้อมูล) ไม่มีผลต่อโค้ด

-เราสามารถ INSERT ข้อมูลใส่ในตารางได้ ยกตัวอย่างเช่น:

```
INSERT INTO employee VALUES  
(1, 'Diluc', 'Marketing', 'CEO', 100000),  
(2, 'Kaeya', 'Marketing', 'VP', 85000),  
(3, 'Jean', 'Sales', 'Manager', 60000);
```

[ใส่ข้อมูลเข้าไปในตาราง employee ประกอบด้วยข้อมูลของ Diluc, Kaeya, และ Jean]

-เช็คด้วย `SELECT * FROM employee;`

id	name	department	position	salary
1	Diluc	Marketing	CEO	100000
2	Kaeya	Marketing	VP	85000
3	Jean	Sales	Manager	60000

Common SQL Errors

1. Table ชื่อตาราง already exists: มีชื่อตารางนั้นใน Database อยู่แล้ว สร้างตารางชื่อซ้ำไม่ได้
2. UNIQUE constraint failed: ชื่อตาราง.ชื่อ column: ถ้ากำหนด column ของตารางเป็น PRIMARY KEY หรือ UNIQUE จะ INSERT ข้อมูลซ้ำกับข้อมูลใน column นั้น ๆ ไม่ได้
3. No such table: ไม่มีตารางในฐานข้อมูล อาจเกิดจากการพิมพ์ชื่อตารางผิด ให้ดูชื่อตารางให้ถี่ถ้วนรับ Query ทุกครั้ง [เราจะนิยมพิมพ์ชื่อของตารางและ column เป็นตัวพิมพ์เล็ก ส่วน SQL Clause จะนิยมเขียนเป็นตัวพิมพ์ใหญ่]
4. near "FROM": syntax errors: พิมพ์ไวยากรณ์ผิด อย่าลืมตรวจสอบ comma ก่อนกดรัน Query ทุกครั้ง

Lesson 4: Select Data

-SELECT ใช้ในการดึงข้อมูลจาก Column ที่เราต้องการ เช่น:

```
SELECT * FROM employee;
```

[ดึงข้อมูลจาก column ทั้งหมดของตาราง employee]

-เราสามารถเลือกได้ว่าจะดึงข้อมูลที่ row ด้วย limit เช่น:

```
SELECT id, name, salary FROM employee LIMIT 2;
```

id	name	salary
1	Diluc	100000
2	Kaeya	85000

[ดึงข้อมูลจาก column id, name, และ salary จากตาราง employee โดยตั้งแต่ 2 row แรก เท่านั้น]

Lesson 5: Transform Columns

-เราสามารถสร้าง column ใหม่โดยอิงจาก column เดิมได้ เช่น:

```
SELECT
  name,
  salary,
  salary*1.5 AS new_salary,
  LOWER(name) || '@company.com' AS company_email
FROM employee;
```

[ดึงข้อมูลจาก column name, salary, new_salary ที่ได้ค่าจากค่าใน salary*1.5, และ company_email ที่ได้ค่าจาก column name ที่ถูกแปลงเป็นตัวพิมพ์เล็กทั้งหมด แล้วนำมา Concatenate (เชื่อม) กับ @company.com]

name	salary	new_salary	company_email
Diluc	100000	150000	diluc@company.com
Kaeya	85000	127500	kaeya@company.com
Jean	60000	90000	jean@company.com

Lesson 6: Filter Data

-วิธีการทำให้ Code ของเราเป็น Comment (ไม่ส่งผลกับ Code จริง) คือการใช้ /* Code */ เปลี่ยน Code ข้างในให้กลายเป็น Comment

-เราสามารถใช WHERE ในการสร้างเงื่อนไขในการดึงข้อมูล เช่น สมมติเราต้องการดึงข้อมูลจากเฉพาะแผนกเดียว ก็สามารถทำได้ เช่น:

```
Select * FROM employee WHERE department = 'IT';
```

[ดึงข้อมูลจาก column ทั้งหมดของตาราง employee โดยที่ column department เป็น IT]

id	name	department	position	salary
4	Albedo	IT	Manager	88000
5	Eula	IT	Manager	69000

-นอกจากนี้ เราสามารถเพิ่ม AND หลังเงื่อนไขแรกของ WHERE เพื่อเพิ่มเงื่อนไขในการกรองอื่น ๆ ได้ เช่น:

```
Select * FROM employee WHERE department = 'IT' AND salary < 80000;
```

[ดึงข้อมูลจาก column ทั้งหมดของตาราง employee โดยที่ column department เป็น IT และค่าใน column salary ต่ำกว่า 80000]

id	name	department	position	salary
5	Eula	IT	Manager	69000

-เราสามารถใช IN ในการกำหนดเงื่อนไขหลายเงื่อนไขได้เช่นกัน เช่น:

```
Select * FROM employee WHERE department IN ('Marketing', 'IT');
```

[ดึงข้อมูลจาก column ทั้งหมดของตาราง employee โดยที่ค่าของ column department เป็น Marketing หรือ IT]

id	name	department	position	salary
1	Diluc	Marketing	CEO	100000
2	Kaeya	Marketing	VP	85000
4	Albedo	IT	Manager	88000
5	Eula	IT	Manager	69000

Lesson 7: Update Data

-เราสามารถทำการอัปเดตข้อมูลในตารางได้ด้วย UPDATE เช่น:

```
UPDATE employee SET salary = 99000 WHERE id = 3;
```

[Update ข้อมูลในตาราง employee ด้วยการ Set ค่าให้ row ที่ id = 3 มีค่า salary = 99000]

id	name	department	position	salary
1	Diluc	Marketing	CEO	100000
2	Kaeya	Marketing	VP	85000
3	Jean	Sales	Manager	99000
4	Albedo	IT	Manager	88000
5	Eula	IT	Manager	69000

สังเกตว่าหลังจากการแสดงผลตารางใหม่อีกครั้ง ค่า salary ของ Jean ที่มี id = 3 เปลี่ยนจากเดิมคือ 60000 เป็น 99000

Lesson 8: Delete Data

-เราสามารถลบข้อมูลในตารางได้ด้วย DELETE เช่น:

```
DELETE FROM employee WHERE name = 'Albedo';  
SELECT * FROM employee;
```

[ลบ row ที่ค่าใน column name = Albedo และดึงข้อมูลจาก column ทั้งหมดของตาราง employee]

id	name	department	position	salary
1	Diluc	Marketing	CEO	100000
2	Kaeya	Marketing	VP	85000
3	Jean	Sales	Manager	99000
5	Eula	IT	Manager	69000

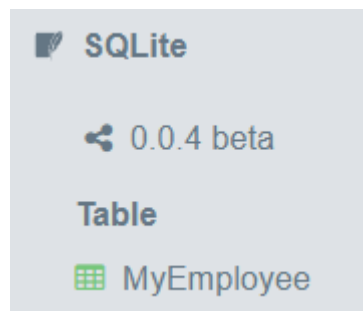
-เราสามารถลบข้อมูลได้หลาย row พร้อมกันด้วยการตั้งเงื่อนไขให้ลบหลาย row พร้อมกัน การไม่ใส่ WHERE จะทำให้ลบข้อมูลจากตารางทั้งทั้งหมด ดังนั้น ควรเช็คโค้ดให้ดีก่อนรัน Query เสมอ

Lesson 9: Alter Table

-เราสามารถเปลี่ยนแปลงตารางได้ด้วย ALTER TABLE เช่น:

```
ALTER TABLE employee RENAME TO MyEmployee;
SELECT * FROM MyEmployee;
```

[เปลี่ยนชื่อของตารางให้มีชื่อใหม่คือ MyEmployee แล้วดึงข้อมูลจาก column ทั้งหมดของตาราง MyEmployee]



-เราสามารถเพิ่ม column ด้วย ALTER TABLE ได้ เช่น:

```
ALTER TABLE MyEmployee ADD email TEXT;
SELECT * FROM MyEmployee;
```

[เพิ่ม column email ที่เก็บค่า TEXT แล้วดึงข้อมูลจาก column ทั้งหมดของตาราง MyEmployee]

id	name	department	position	salary	email
1	Diluc	Marketing	CEO	100000	NULL
2	Kaeya	Marketing	VP	85000	NULL
3	Jean	Sales	Manager	99000	NULL
5	Eula	IT	Manager	69000	NULL

-เราสามารถ Update ข้อมูลใน column ใหม่ได้ด้วย UPDATE เช่น:

```
UPDATE MyEmployee SET email = LOWER(name) || '@company.com';
SELECT * FROM MyEmployee;
```

[Update ข้อมูลในตาราง MyEmployee ด้วยการเซตให้อีเมลของทุกคนมีชื่อจากค่าของ column name ที่ถูกแปลงเป็นตัวพิมพ์เล็กทั้งหมด แล้วนำมา Concatenate (เชื่อม) กับ @company.com แล้วดึงข้อมูลจาก column ทั้งหมดของตาราง MyEmployee]



id	name	department	position	salary	email
1	Diluc	Marketing	CEO	100000	diluc@company.com
2	Kaeya	Marketing	VP	85000	kaeya@company.com
3	Jean	Sales	Manager	99000	jean@company.com
5	Eula	IT	Manager	69000	eula@company.com

Lesson 10: Copy and Drop Table

-เราสามารถสร้าง Backup Table เพื่อสำรองข้อมูลตารางของเราได้ด้วยการ CREATE TABLE ชื่อตารางใหม่ AS SELECT * FROM ชื่อตาราง ได้ เช่น:

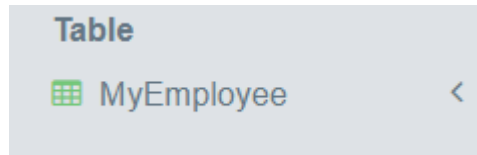
```
CREATE TABLE MyEmployee_backup AS SELECT * FROM MyEmployee;
```

[สร้างตารางชื่อ MyEmployee_backup ด้วยการดึงข้อมูลทั้งหมดจากตาราง MyEmployee]

Table	
 MyEmployee	<
 MyEmployee_backup	<

-เราสามารถลบตารางได้ด้วย DROP TABLE แต่ถ้าลบแล้วจะกู้คืนมาไม่ได้หากไม่ได้สำรองข้อมูลไว้ ดังนั้นคิดให้ดีก่อน DROP TABLE

```
DROP TABLE MyEmployee_backup;
```



-สังเกตว่าตาราง MyEmployee_Backup จะหายไปหลังจากการลบตารางทิ้ง
