

SQL Live Class

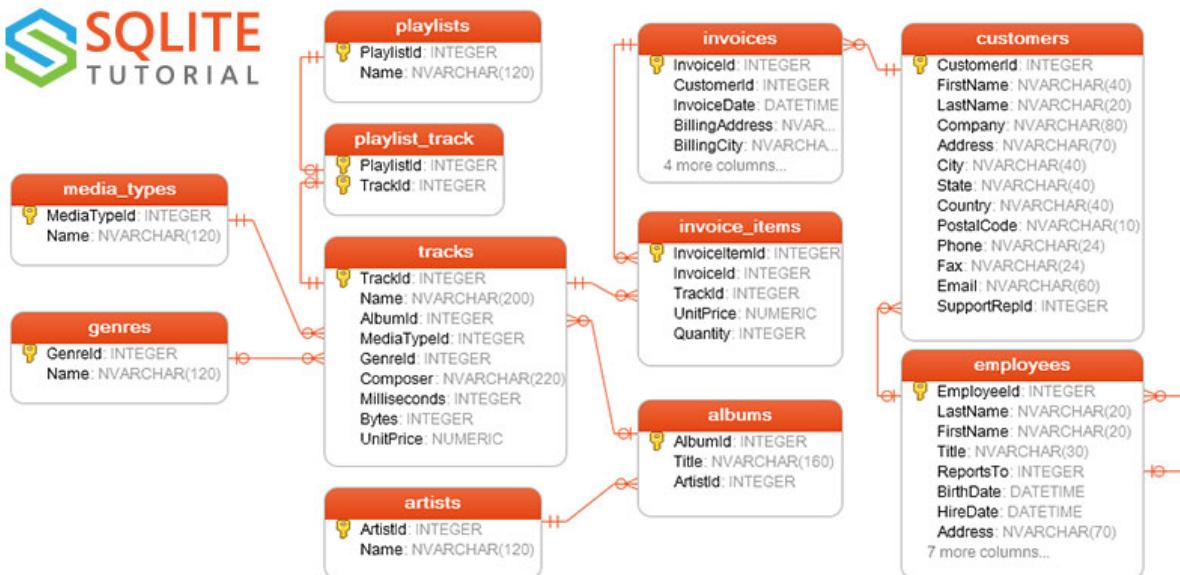
Tags	Live Class	SQL
Class		
Finished Yet?	<input checked="" type="checkbox"/>	
Knowledge	Live Classes	

-DBDiagram (ใช้สำหรับ ER Diagram): <https://dbdiagram.io/home>

-chinook.db (ไฟล์ Database):

<https://s3-us-west-2.amazonaws.com/secure.notion-static.com/de8b7695-3ae d-4e58-ab71-0e87fcf954e3/chinook.db>

-Chinook ER Diagram:



-SQL Live Class Slide:

https://s3-us-west-2.amazonaws.com/secure.notion-static.com/dfd920a2-e195-48f2-beba-7f6bcf0443fa>Hello_World_-_Databases_Bootcamp.pdf

- SQL เป็นภาษาที่สำคัญมากในการทำงานกับ Data เพราะ SQL ใช้ในการจัดการกับ Database
- ความยากของ SQL ไม่ได้อยู่ที่ SQL แต่เป็นเรื่องการนำไปใช้งาน เช่น เราไม่รู้ว่าควรจะ JOIN TABLE ไหนกับ TABLE ไหน และเราเข้าใจ Business มากแค่ไหน ถ้าเข้าใจ Business จะทำให้สร้าง Mental ER Diagram ในหัวเพื่อช่วยในการเชื่อม Table ที่เราต้องการได้
- การระบบเป็น SQL ตอบกลับมาเป็น Table
- แต่ละระบบก็จะมี Client แตกต่างกันไป เช่น MySQL จะใช้ Client เป็น MySQL Workbench เป็นต้น
- ในการเรียน จะใช้ SQLite ผ่าน Web Browser
- SQL = Structured Query Language
- Top 3 Programming Language for Data Science: Python, R, SQL
- SQL Flavours (SQL มีหลาย Dialect)
- DataGrip (ใช้ฟรี 30 วัน Connect กับ Database ได้หลาย Dialect เช่น Connect ไป MySQL, Oracle, SQL Server เป็นต้น)
- ER Diagram เปรียบเสมือนแบบที่ ที่แสดงความสัมพันธ์ระหว่าง Table ใน Database (แต่ในชีวิตจริง ไม่มี ER Diagram หรือถ้าจะมีก็น้อยมาก อย่างได้ต้องสร้างเอง)

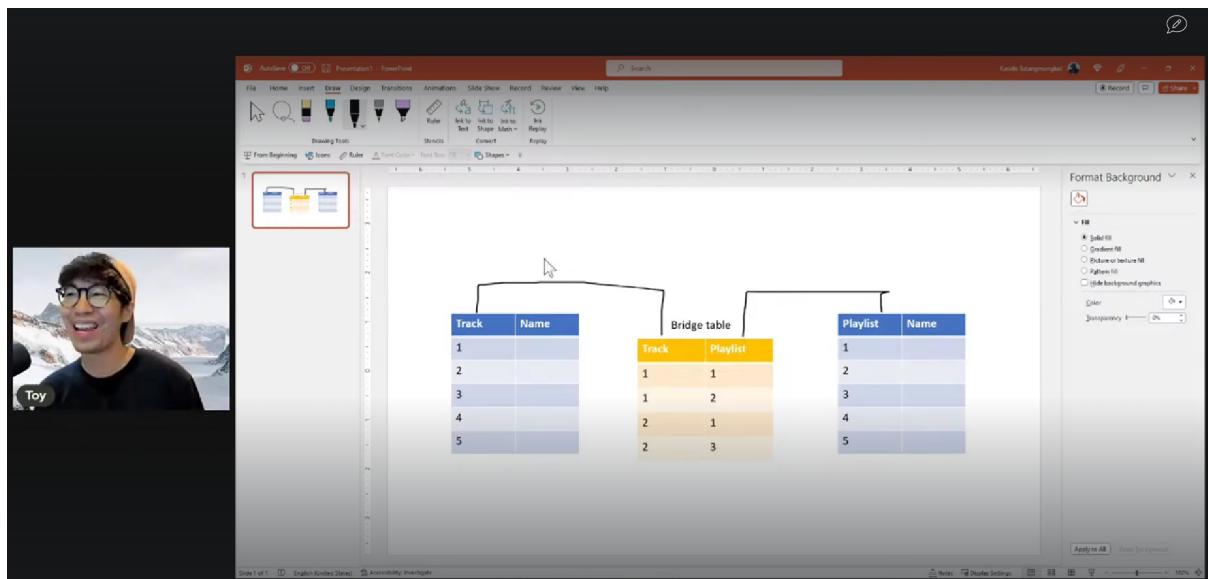
-คำถามที่ควรถามบริษัทก่อนทำงาน:

1. Database ของบริษัทใช้ตัวไหนอยู่?
2. มี ER Diagram หรือ Document ให้อ่านไหม?

-Primary Key ในแต่ละ Table คือ column ที่มีค่าไม่ซ้ำกัน (เช่น id) ใช้ในการ JOIN TABLE เข้าด้วยกัน

-Foreign Key คือ Primary Key ที่โผล่ใน Table อื่นที่ไม่ใช่ Table หลัก มีค่าซ้ำกันได้

-วิธีการจัดการกับ Many-to-Many Relationship คือการสร้าง Bridge Table ขึ้นมาคั่นกลาง



-เราสามารถ Model Relationship ตัว Data ของเราได้ (ส่วนมากคนทำจะเป็นกับวิศวกรหรือกับ Data) เรียกว่า Data Model ทำให้ทำงานได้ง่ายขึ้น

-CRUD Operations: Create, Retrieve (SELECT), Update, Delete

```
#Create
CREATE TABLE customer (
    id int,
    name text,
    city text,
    email text,
    avg_spending real
);

#Insert Into
INSERT INTO customer VALUES
    (1, "Toy", "BKK", "toy@mail.com", 500.25),
    (2, "Joe", "BKK", "joe@mail.com", 125.50),
    (3, "Ann", "LON", "ann@mail.com", 999.50),
    (4, "Ken", "LON", "ken@mail.com", 658.25);
```

```
#Retrieve (SELECT)
SELECT * FROM customer;

#DROP TABLE
DROP TABLE customer;
```

Basic SQL Clauses:

1. SELECT
2. FROM
3. WHERE
4. GROUP BY (+Aggregate Functions)
5. HAVING
6. ORDER BY

-Whitespace ไม่มีผลในการดึงข้อมูล

The screenshot shows a SQLite database named 'chinook.db'. In the main pane, the following SQL code is displayed:

```
1 SELECT
2   firstname || " " || lastname AS fullname
3 FROM customers
4 LIMIT 10;
```

In the results pane, the column header is 'fullname' and the data rows are:

fullname
Luís Gonçalves
Leonie Köhler
François Tremblay
Bjørn Hansen
František Wichterlová
Helena Holý
Astrid Gruber
Daan Peeters
Kara Nielsen
Eduardo Martins

-ROUND() ใช้ในการปัดเศษทศนิยมให้เป็นตัวเลขที่ต้องการ

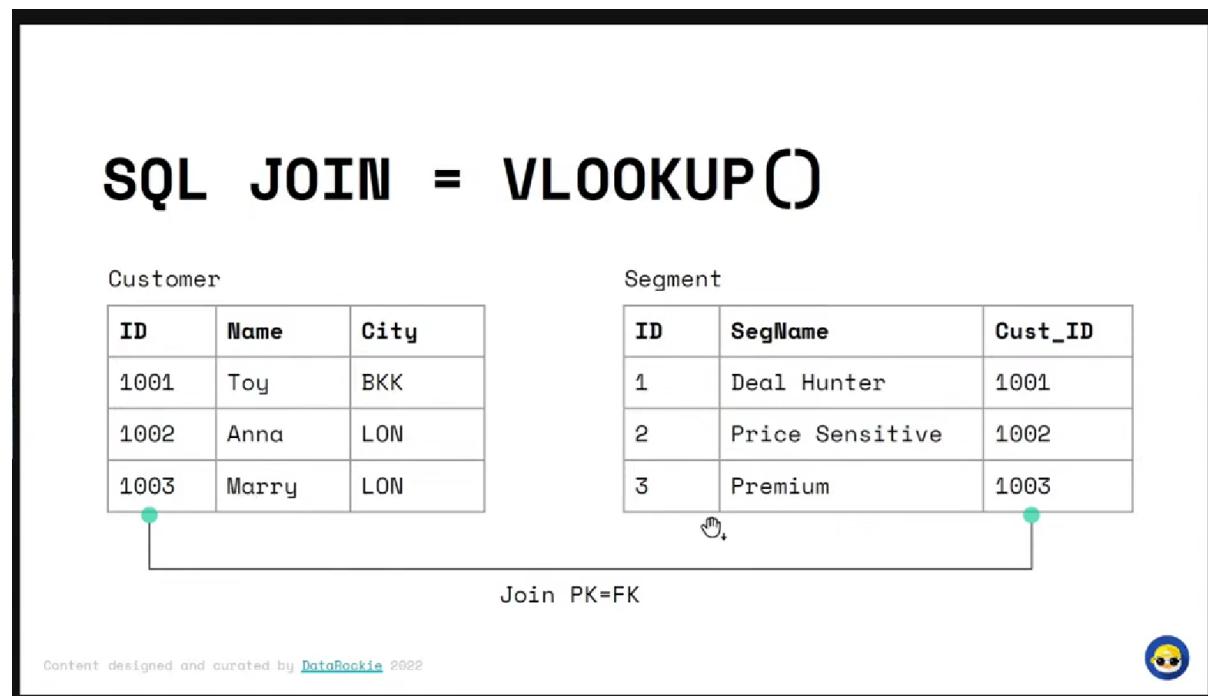
- เราสามารถ Export Table ที่เรา Query มาได้ ด้วยการไปที่ Export → CSV → เลือก Column Name เป็น First Line
- LIKE ใช้ทำ Pattern Matching
- การใส่ ; หลัง QUERY เมื่อตอนการเขียน Fullstop หลังจบประโยค
- FROM กับ WHERE รันก่อน SELECT ดังนั้น พิมพ์ชื่อ Table ให้ถูกก่อน จากนั้นถึงกำหนดเงื่อนไขด้วย WHERE และใช้ SELECT เลือก column ที่ต้องการดึงข้อมูล
- วิธีการเปลี่ยนประเภทของข้อมูลด้วยการใช้ CAST():

```
#Change Data Type
SELECT CAST(col AS data_type);
```

- Subquery (Query ซ้อน Query):

```
SELECT firstname, lastname
FROM (SELECT * FROM customers WHERE country = "USA") AS sub;
```

- การ JOIN TABLE จะใช้ Primary Key ทำการ Join กับ Foreign Key



```
#JOIN Syntax (Default Join = INNER JOIN)
SELECT * FROM table1
JOIN Table2
ON table1.pk = table2.fk;
```

*งาน JOIN 99% ใน SQL คือ INNER JOIN และ LEFT JOIN

```
SELECT *
FROM artists AS t1
JOIN albums AS t2 ON t1.artistid = t2.artistid
JOIN tracks AS t3 ON t2.albumid = t3.albumid
WHERE t2.title LIKE 'The %' AND t3.composer LIKE 'M%';
```

*ยิ่ง JOIN TABLE เยอะจะยิ่งซ้ำ

```
#Aggregate Functions
SELECT
    ge.name,
    COUNT(*) AS no_tracks,
    AVG(tr.bytes) AS avg_bytes,
    SUM(tr.bytes) AS sum_bytes,
    MIN(tr.bytes) AS min_bytes,
    MAX(tr.bytes) AS max_bytes
FROM artists AS ar
JOIN albums AS al ON ar.artistid = al.artistid
JOIN tracks AS tr ON al.albumid = tr.albumid
JOIN genres AS ge ON tr.genreid = ge.genreid
WHERE al.title NOT LIKE "The %"
GROUP BY ge.name
HAVING ge.name LIKE "S%"
ORDER BY no_tracks DESC;
```

-老子อยู่ใน GROUP BY ต้องใส่ใน SELECT ด้วย

*เลือกใช้เครื่องมือให้เหมาะสมกับงาน

-WHERE ใช้ในการ Filter ข้อมูลจาก Table ส่วน HAVING ใช้ในการ Filter ข้อมูลจากกลุ่มที่เรา GROUP BY

```
#Subquery
SELECT * FROM (
    SELECT
        ar.name AS artist_name,
        al.title,
        tr.name AS track,
        tr.bytes,
        ge.name AS genre
    FROM artists AS ar
    JOIN albums AS al ON ar.artistid = al.artistid
    JOIN tracks AS tr ON al.albumid = tr.albumid
    JOIN genres AS ge ON tr.genreid = ge.genreid
) AS sub
WHERE genre = "Rock" and artist_name LIKE "D%";
```

-Common Table Expression (สร้างตัวแปรใน SQL ด้วย WITH):

```
#Using WITH
WITH sub AS (
    SELECT
        ar.name AS artist_name,
        al.title,
        tr.name AS track,
        tr.bytes,
        ge.name AS genre
    FROM artists AS ar
    JOIN albums AS al ON ar.artistid = al.artistid
    JOIN tracks AS tr ON al.albumid = tr.albumid
    JOIN genres AS ge ON tr.genreid = ge.genreid
)
SELECT * FROM sub
WHERE genre = "Rock" and artist_name LIKE "M%";
```

-Combine Query:

```
#The First Query
SELECT * FROM customers WHERE country = "USA";

#The Second Query
SELECT * FROM invoices WHERE STRFTIME("%Y", invoicedate) = "2010";

#The Combined Query
WITH usa_customers AS (
    SELECT * FROM customers WHERE country = "USA"
), invoice_2010 AS (
    SELECT * FROM invoices WHERE STRFTIME("%Y", invoicedate) = "2010"
)

SELECT c.firstname, SUM(i.total)
FROM usa_customers AS c
JOIN invoice_2010 AS i
ON c.customerid = i.customerid;

--Single Line Comment
/*Multiple
Line
Comment*/
```

-Windows Function

```
SELECT
    ROW_NUMBER() OVER() AS row_num, --Windows Function
    firstname,
    country
```

```

FROM customers
ORDER BY 1;

--With Subquery
SELECT * FROM (
    SELECT
        ROW_NUMBER() OVER(PARTITION BY country ORDER BY firstname) AS row_num,
        firstname,
        country
    FROM customers
)
WHERE row_num = 1;

--Segmentation (Rule-Based)
SELECT
    *,
    CASE WHEN segment IN (1,2) THEN "low"
    WHEN segment IN (4,5) THEN "high"
    ELSE "medium"
    END AS label_segment
FROM (
    SELECT
        name AS song_name,
        milliseconds,
        NTILE(5) OVER(Order BY milliseconds) AS segment
    FROM tracks ORDER BY milliseconds
)
GROUP BY segment;

```

-ໂຄດີ່ກອຍ: https://coda.io/d/Knowledge_dfWeZT_3wO9/SQL-Code_sure1

-read main.sql ຕ້ອກາຮັນ SQL ໃນ Replit

Homework:

-Restaurant Database with 5+ tables

-Write 5+ queries with:

1. 1x With
2. 1x Subquery
3. 1x Aggregate Function

My Homework Replit Link:

<https://replit.com/@phuurinthphitha/BootcampHomeworkSQL>

<https://replit.com/@phuurinthphitha/BootcampHomeworkSQL#main.sql>

https://s3-us-west-2.amazonaws.com/secure.notion-static.com/257908ca-f9c7-462c-8f6d-85ad6f8326b5/Poorin_Restaurant_ERdiagram.pdf

https://s3-us-west-2.amazonaws.com/secure.notion-static.com/cab1301f-3abe-476a-8d2c-e5411d436a0b/Poorin_Restaurant_ERdiagram.pdf