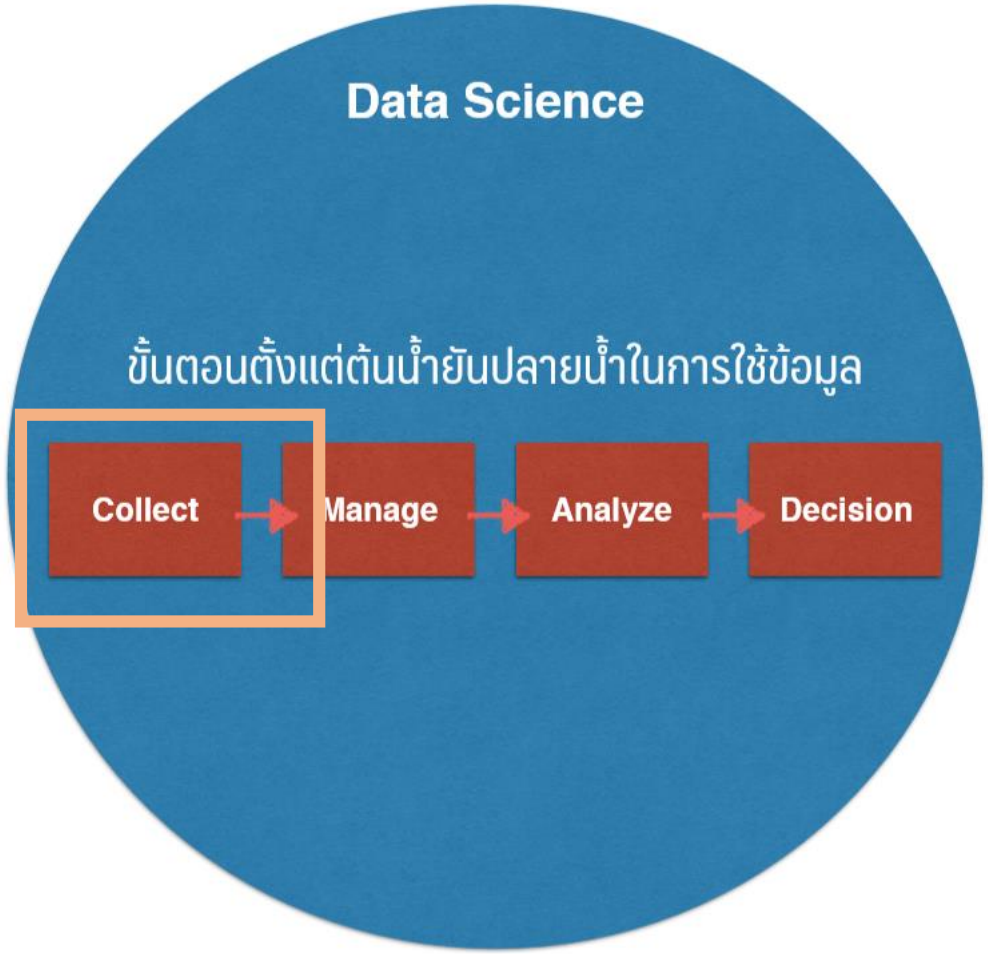
# Chapter 2

# Data Collection With SQL

# อยู่ขั้นตอนไหนของการทำ data science?



# เครื่องมือที่ใช้วันนี้

**1) โปรแกรมในการ run SQL**

https://sqlitebrowser.org/blog/version-3-12-2-released/ หรือ
https://sqliteonline.com/

**2) Database chinook**

chinook

https://drive.google.com/file/d/19AFIWZ5z8ymO4IvVOr8ltpWQiA8Q2TG9/view?usp=sharing

# SQL คืออะไร

SQL ย่อมาจาก **S**tructured **Q**uery **L**anguage เป็นภาษา ทางการที่ใช้งานกับ database มา มากกว่า 40 ปีและเป็น ทักษะสำคัญของ data analyst ที่ทุกคนควรมี Command สำคัญใน SQL ที่ต้องใช้ให้เป็น ประกอบด้วย

- ✓SELECT
- ✓WHERE
- ✓ORDER BY
- ✓GROUP BY
- ✓HAVING

# [1] SELECT ----------

เลือกทุกคอลัมน์ใน table นั้น ๆ ด้วย asterisk (*)

```
SELECT *

FROM table_name;
```

# ปฏิบัติ #1

- จงเลือกทุกคอลัมน์ของ table invoices

# เลือกคอลัมน์ที่เราต้องการใน table นั้น ๆ

เลือกคอลัมน์ที่เราต้องการใน table นั้น ๆ สามารถพิมพ์ชื่อคอลัมน์ได้เลย แยกชื่อคอลัมน์ด้วย comma (,)

```
SELECT columnA, columnB
FROM table_name;
```

# ปฏิบัติ #2

- จงเลือกคอลัมน์ albumid, name, milliseconds, bytes จาก table tracks

# เปลี่ยนชื่อคอลัมน์ด้วย AS (alias)

เปลี่ยนชื่อคอลัมน์ด้วย AS (alias)

```
SELECT columnA AS new_column_name
FROM table_name;
```

# ปฏิบัติ #3

- SELECT Bytes, Bytes/1000000 AS MB FROM tracks;

# กำหนดจำนวนแถวที่อยากดึงข้อมูลออกมาด้วย LIMIT

กำหนดจำนวนแถวที่อยากดึงข้อมูลออกมาด้วย LIMIT

```
SELECT firstname, lastname
FROM table_name
LIMIT number of limit;
```

# ปฏิบัติ #4

- SELECT Bytes, Bytes/1000000 AS MB FROM tracks LIMIT 10;

# นับจำนวนแถวทั้งหมดใน table นั้นด้วย COUNT

นับจำนวนแถวทั้งหมดใน table นั้นด้วย COUNT

```
SELECT COUNT(*) FROM table_name;
```

# ปฏิบัติ #5

- table invoices มีกี่ rows?

# นับจำนวน value ที่ไม่ซ้ำกันเลยในคอลัมน์นั้น ๆ ด้วย DISTINCT

นับจำนวน value ที่ไม่ซ้ำกันเลยในคอลัมน์นั้น ๆ ด้วย DISTINCT

```
SELECT DISTINCT country
FROM customers;
```

# [2] WHERE ----------

WHERE คือการใส่เงื่อนไข (conditions) ในการ filter rows ที่เราต้องการ

```
SELECT *
FROM invoices
WHERE total  >= 20;
```

ถ้าต้องการใส่มากกว่าหนึ่งเงื่อนไข สามารถใช้ AND หรือ OR เข้ามาช่วยใน WHERE clause

# ปฏิบัติ #6

**ตัวอย่าง** เลือกทุกคอลัมน์จาก table invoices ฟิลเตอร์เฉพาะ total ที่มากกว่าหรือเท่ากับ 20 **หรือ** BillingCountry ที่เป็น USA

```
SELECT *

FROM invoices

WHERE total >= 20 OR BillingCountry = 'USA';
```

# ปฏิบัติ #7

```
SELECT * FROM customers WHERE country = 'USA' AND state = 'CA';
```

**เทียบความแตกต่าง**

```
SELECT * FROM customers WHERE country = 'USA' or state = 'CA';
```

# ปฏิบัติ #8

- จงเลือกลูกค้าทุกคนที่อาศัยในประเทศ USA, Brazil, France จาก table customer

สามารถเขียนในรูปแบบนี้ ด้วยการใช้ IN

```
SELECT *  FROM customers WHERE country IN ('USA', 'Brazil', 'France');
```

เราสามารถใช้ wildcard (%, _) เพื่อทำ pattern matching (สำหรับคอลัมน์ที่เป็น text/string เช่น ชื่อลูกค้า) ใน WHERE clause ได้เช่นกัน

ฟิลเตอร์เฉพาะลูกค้าที่ชื่อขึ้นต้นด้วยตัว D

```
SELECT *
FROM customers
WHERE firstname LIKE 'D%‘;
```

# ปฏิบัติ #9

ฟิลเตอร์เฉพาะลูกค้าที่ใช้อีเมลล์ @google.com

```
SELECT *
FROM customers
WHERE email LIKE '%@google.com';
```

# ปฏิบัติ #10

ฟิลเตอร์เฉพาะลูกค้าที่ใช้ชื่อจริงว่า J_hn โดยที่ _ จะเป็นตัวอักษรอะไรก็ character

```
SELECT *
FROM customers
WHERE firstname LIKE 'J_hn';
```

# ตรวจสอบ missing value (NULL) ในตารางต่าง ๆ ด้วย IS NULL / IS NOT NULL

เราสามารถตรวจสอบ missing value (NULL) ในตารางต่าง ๆ ด้วย IS NULL / IS NOT NULL

ฟิลเตอร์เฉพาะลูกค้าที่ไม่ระบุ address ใน table customers

```
SELECT *
FROM customers
WHERE address IS NULL;
```

# ปฏิบัติ #11

ฟิลเตอร์เฉพาะลูกค้าที่ระบุ address ใน table customers

```
SELECT *
FROM customers
WHERE address IS NOT NULL;
```

# แทนที่ NULL ในคอลัมน์ ด้วยคำสั่ง COALESCE

ปฏิบัติ #12

แทนที่ NULL ในคอลัมน์ company ด้วยคำว่า 'freelance' ด้วยคำสั่ง COALESCE

```
SELECT COALESCE(company, 'freelance')
FROM customers;
```

# [3] ORDER BY ----------

เรียงข้อมูลแถว (rows) จากค่าน้อยไปมาก (Ascending)

ปฏิบัติ #13

```
SELECT * FROM invoices
WHERE total >= 5
ORDER BY total;
```

เรียงข้อมูลแถว (rows) จากค่ามากไปน้อย (Descending)

```
SELECT * FROM invoices
WHERE total >= 5
ORDER BY total DESC;
```

เราสามารถใช้ ORDER BY กับคอลัมน์ที่เป็นตัวอักษร text/string

ได้โดย ascending order จะเรียงแถวจาก A-Z และ desceding order (DESC) เรียงจาก Z-A

# [4] GROUP BY ----------

ใช้จับกลุ่มผล query ของเรา ปกติจะใช้คู่กับพวก AGGREGATE functions ปฏิบัติ #14

```
SELECT country, COUNT(*) AS n
FROM customers
GROUP BY country;
```

# AGGREGATE functions

AGGREGATE functions คืออะไร? คือ ฟังก์ชันที่ใช้คำนวณค่าสถิติเบื้องต้น ในคอลัมน์ที่เราเลือกมา เช่น

- COUNT – นับจำนวนแถว
- AVG - หาค่าเฉลี่ย
- SUM - หาผลรวม
- MAX - หาค่ามากที่สุด
- MIN – หาค่าน้อยที่สุด

# ตัวอย่างการใช้งาน AGGREGATE functions

ปฏิบัติ #15

ตัวอย่าง การใช้งาน AGGREGATE functions

```
SELECT
        COUNT(total) AS number_of_transaction,
        AVG(total) AS average_sales,
        SUM(total) AS total_sales,
        MAX(total) AS max_sales,
        MIN(total) AS min_sales
FROM invoices;
```

# [5] HAVING ----------

ฟิลเตอร์ผล query ที่ได้จากการ GROUP BY ใช้เหมือนกับ WHERE clause
ตัวอย่างด้านล่าง จะได้ออกมาสองคอลัมน์คือ country และ n โดยฟิลเตอร์เอาเฉพาะประเทศที่มีลูกค้าตั้งแต่ 10 คนขึ้นไป

ปฏิบัติ #16

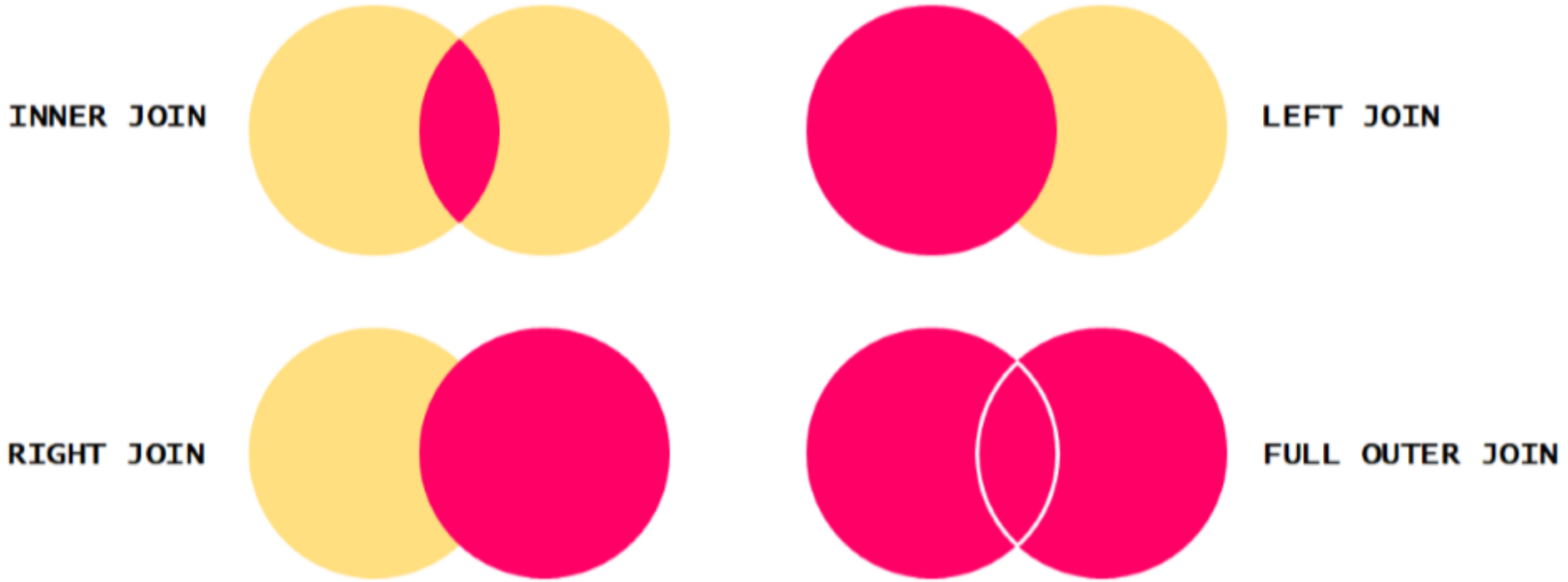
```
SELECT
        country,
        COUNT(*) AS n
FROM customers
GROUP BY country
HAVING n >= 10;
```

# Relational Database ----------

เราสามารถเขียน query JOIN หลายๆ tables เข้าด้วยกัน รูปแบบการ JOIN ที่ใช้กันบ่อย ๆ ได้แก่

- INNER JOIN
- LEFT JOIN
- RIGHT JOIN
- FULL OUTER JOIN

# Query JOIN



# Inner JOIN

## INNER JOIN

**Customer**

| ID | Name |
|---|---|
| 1 | Toy |
| 2 | Hello |
| 3 | World |
| 4 | SQL |
| 5 | Awesome |

**Age**

| ID | Age |
|---|---|
| 1 | 29 |
| 2 | 30 |
| 4 | 18 |
| 6 | 25 |
| 7 | 26 |

**Result**

| ID | Name | Age |
|---|---|---|
| 1 | Toy | 29 |
| 2 | Hello | 30 |
| 4 | SQL | 18 |

ผลลัพธ์ออกมาเฉพาะ ROW ที่ matched กันได้ของสองตารางเท่านั้น

# Left JOIN

LEFT JOIN

ตารางซ้ายมือยังอยู่เหมือนเดิม แต่จะเชื่อมตารางขวาใน row ที่ matched

**Customer**

| ID | Name |
|---|---|
| 1 | Toy |
| 2 | Hello |
| 3 | World |
| 4 | SQL |
| 5 | Awesome |

**Age**

| ID | Age |
|---|---|
| 1 | 29 |
| 2 | 30 |
| 4 | 18 |
| 6 | 25 |
| 7 | 26 |

**Result**

| ID | Name | Age |
|---|---|---|
| 1 | Toy | 29 |
| 2 | Hello | 30 |
| 3 | World | NULL |
| 4 | SQL | 18 |
| 5 | Awesome | NULL |


# Right JOIN

**RIGHT OUTER JOIN**

**Customers**

| CustomerId | Name |
|---|---|
| 1 | Robert |
| 2 | Peter |
| 3 | Smith |

**Orders**

| OrderId | CustomerId | OrderDate |
|---|---|---|
| 100 | 1 | 2016-10-19 15:21:27 |
| 200 | 4 | 2016-10-20 15:21:27 |
| 300 | 2 | 2016-10-21 15:21:27 |

**RIGHT OUTER JOIN on CustomerId Column**

RESULT

| CustomerId | Name | OrderId | CustomerId | OrderDate |
|---|---|---|---|---|
| 1 | Robert | 100 | 1 | 2016-10-19 15:21:27 |
| NULL | NULL | 200 | 4 | 2016-10-20 15:21:27 |
| 2 | Peter | 300 | 2 | 2016-10-21 15:21:27 |

# Full Outer JOIN

FULL OUTER JOIN

**Customer**

| ID | Name |
|---|---|
| 1 | Toy |
| 2 | Hello |
| 3 | World |
| 4 | SQL |
| 5 | Awesome |

**Age**

| ID | Age |
|---|---|
| 1 | 29 |
| 2 | 30 |
| 4 | 18 |
| 6 | 25 |
| 7 | 26 |

**Result**

| ID | Name | Age |
|---|---|---|
| 1 | Toy | 29 |
| 2 | Hello | 30 |
| 3 | World | NULL |
| 4 | SQL | 18 |
| 5 | Awesome | NULL |
| 6 | NULL | 25 |
| 7 | NULL | 26 |

# เทคนิคเวลาเขียน SQL JOIN

เทคนิคเวลาเขียน SQL JOIN

- ใช้ alias (**AS**) เข้ามาช่วยตั้งชื่อตาราง
- a.column_name

  โดยที่ a คือ alias ของตารางนั้น ๆ

  .column_name คือชื่อคอลัมน์ที่เราต้องการ
- ใช้ ON a.key = b.key เพื่อเชื่อมสองตารางเข้าด้วยกัน *(primary key = foreign key)*

# Template สำหรับเขียน INNER JOIN

**Template สำหรับเขียน INNER JOIN**

```
SELECT
        a.column_name,
        b.column_name
 FROM tableA AS a
 INNER JOIN tableB AS b
 ON a.key = b.key;
```

# ปฏิบัติ #17
# แก้ไขตัวอย่าง*

```
SELECT

a.FirstName,

b.BillingCity

FROM customers AS a

INNER JOIN invoices AS b

ON a.CustomerId = b.CustomerId;
```

# Template สำหรับเขียน LEFT JOIN

**Template สำหรับเขียน LEFT JOIN**

```
SELECT
        a.column_name,
        b.column_name
 FROM tableA AS a
 LEFT JOIN tableB AS b
 ON a.key = b.key;
```

# ปฏิบัติ #18
# แก้ไขตัวอย่าง*

```
SELECT
a.firstname,
b.BillingState
FROM customers AS a
Left JOIN invoices AS b
ON a.CustomerId = b.CustomerId;
```

# Template สำหรับเขียน RIGHT JOIN

**Template สำหรับเขียน RIGHT JOIN**

```
SELECT
        a.column_name,
        b.column_name
FROM tableA AS a
RIGHT JOIN tableB AS b
ON a.key = b.key;
```

# Template สำหรับเขียน FULL JOIN

**Template สำหรับเขียน FULL JOIN**

```
SELECT
        a.column_name,
        b.column_name
FROM tableA AS a
FULL OUTER JOIN tableB AS b
ON a.key = b.key;
```

# Reference

กษิดิศ สตางค์มงคล (2563). **Python for Non-Programmer.** กรุงเทพฯ: Datarockie

# ใบงาน

จับกลุ่มๆ ละ 4 คน แล้วทำดังนี้

- อธิบายข้อมูลในตาราง Database ใน Lab ว่าแต่ละตารางเก็บข้อมูลอะไร
- นำ SQL ที่เรียน ไปเขียนดึงข้อมูลจาก Database ใน Lab จำนวน 8 คำสั่ง โดยให้ได้ผลลัพธ์ออกมา ที่ผู้บริหาร/พนักงาน สามารถนำไปใช้ในการบริหารหรือช่วยในการทำงานได้
- ทำใส่ powerpoint นำเสนอครั้งหน้า ทั้งอธิบายตาราง, คำสั่ง SQL ที่เขียน, ผลลัพธ์ที่ได้จากการ run คำสั่ง SQL ที่เขียน และอธิบายว่า ผลลัพธ์ข้อมูลของแต่ละคำสั่ง เอาไปช่วยในการทำงานได้อย่างไร