# Chapter 2

# SQL for Data Science

# เครื่องมือที่ใช้วันนี้

**1) โปรแกรมในการ run SQL วันนี้ :** sqlitebrowser

https://drive.google.com/file/d/1eFzjKq76irF0dQXxqO2wYnoTvxWSp8md/view?usp=sharing

**2) ไฟล์ Database ที่ใช้ในการเรียน :** chinook

https://drive.google.com/file/d/1Q2u4NZlbngAEkLeKaxasTY1Z4uJmuWS7/view?usp=sharing

# 1. ทำความเข้าใจธุรกิจ (Business Understanding )

- Chinook เป็นฐานข้อมูลของร้านดนตรี

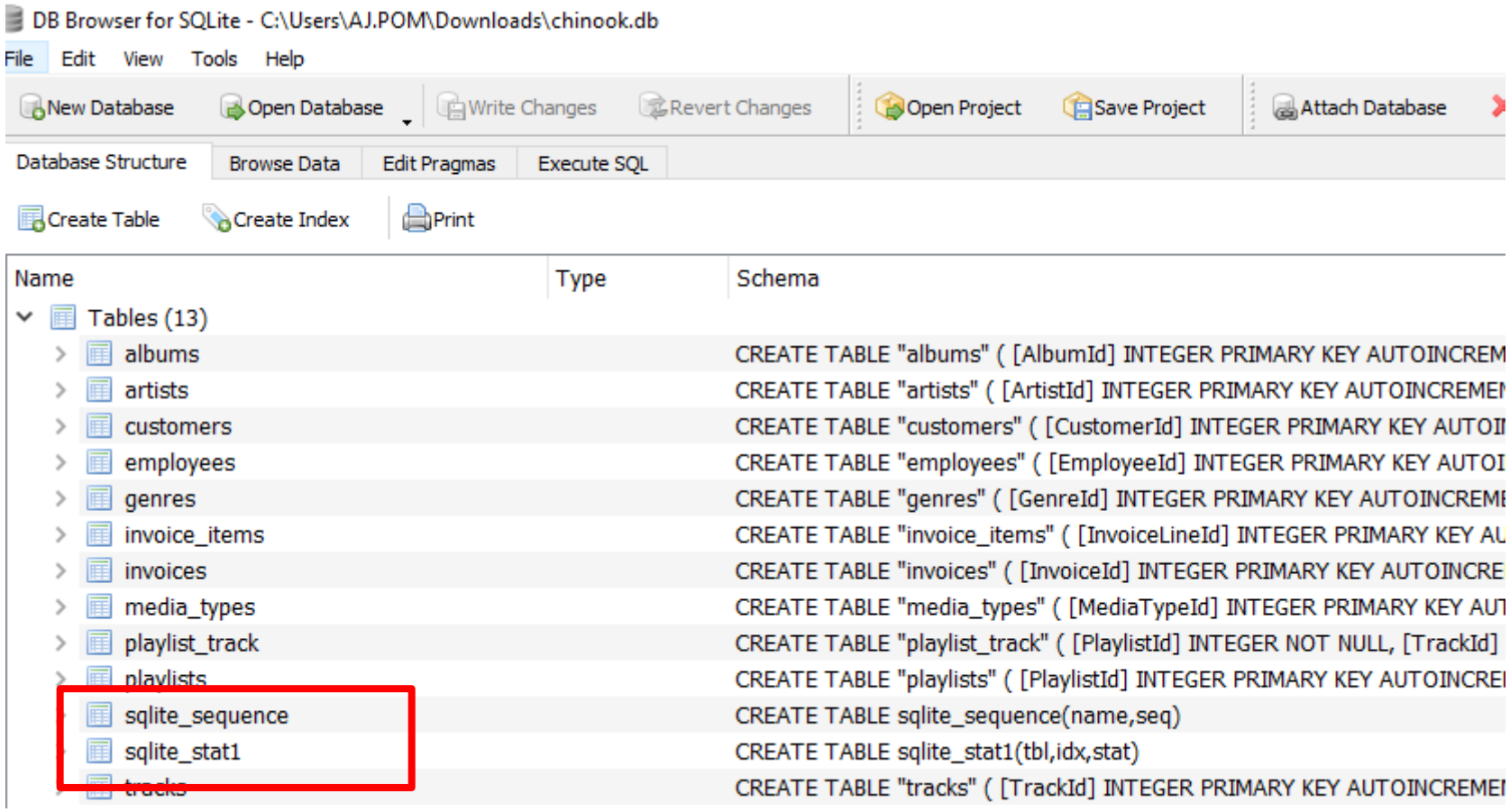
## Chinook Database

### Introduction

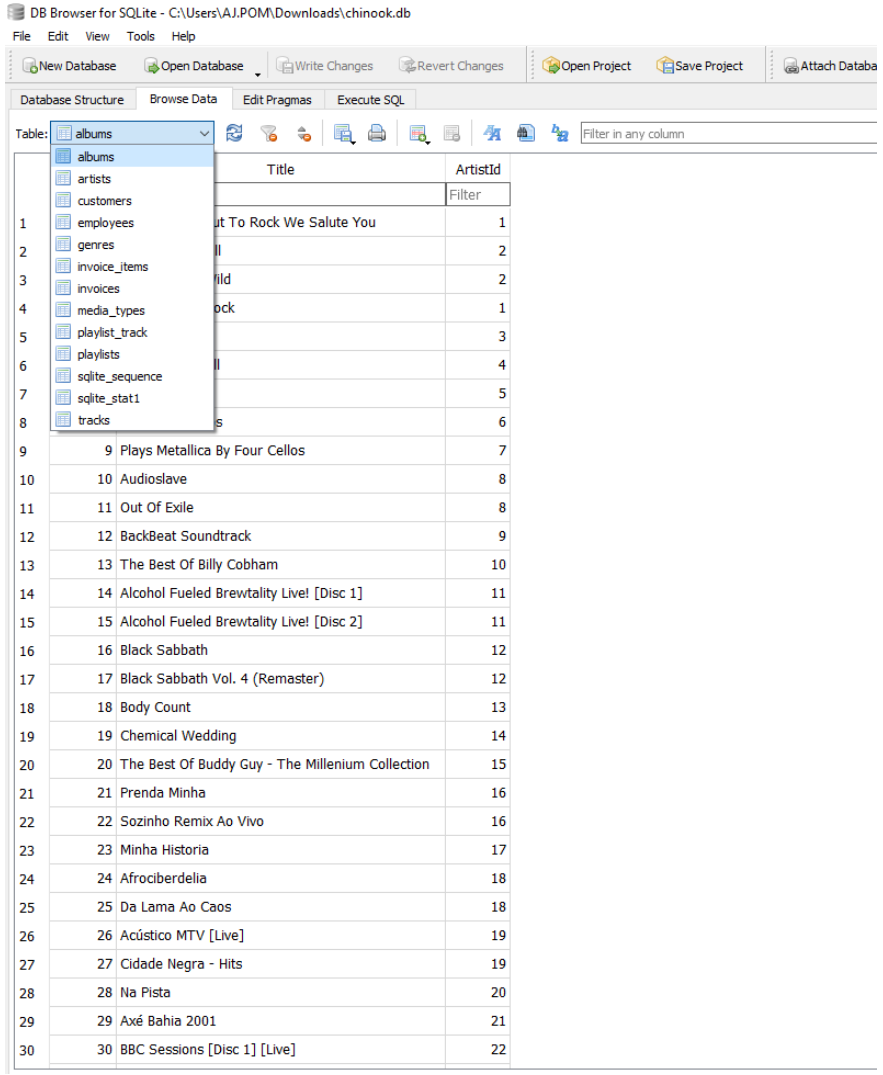
In this project, you will query the Chinook Database. The Chinook Database holds information about a music store. For this project, you will be assisting the Chinook team with understanding the media in their store, their customers and employees, and their invoice information. To assist you in the queries ahead, the schema for the Chinook Database is provided below. You can see the columns that link tables together via the arrows.

Cr. https://m-soro.github.io/Business-Analytics/SQL-for-Data-Analysis/L4-Project-Query-Music-Store/

# 2. ทำความเข้าใจข้อมูล (Data Understanding)



# ดูข้อมูลในฐานข้อมูล Chinook



# 3. เตรียมข้อมูล (Data Preparation)

>> เตรียมข้อมูลด้วย SQL

# 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 ----------

# เลือก ทุกคอลัมน์

เลือก *ทุกคอลัมน์* ใน table นั้น ๆ ด้วย asterisk (*) #ดูข้อมูลทั้งหมดของตารางนั่นเอง

```
SELECT *
FROM table_name;
```

เช่น `SELECT *  FROM customers;`

# ปฏิบัติ #1

- จงแสดงข้อมูลทั้งหมดของ table invoices

# เฉลยปฏิบัติ #1

- จงแสดงข้อมูลทั้งหมดของ table invoices

เฉลย

```
SELECT * FROM invoices;
```

# เลือกคอลัมน์ที่เราต้องการใน table นั้น ๆ

เลือกคอลัมน์ที่เราต้องการใน table นั้น ๆ สามารถพิมพ์ชื่อคอลัมน์ได้เลย แยกชื่อคอลัมน์ด้วย comma (,)

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

เช่น **SELECT** Firstname, Lastname **FROM** customers;

# ปฏิบัติ #2

- จงแสดงข้อมูลคอลัมน์ albumid, name จาก table tracks

# เฉลยปฏิบัติ #2

- จงแสดงข้อมูลคอลัมน์ albumid, name จาก table tracks

เฉลย

```
SELECT albumid, name FROM tracks;
```

# เปลี่ยนชื่อคอลัมน์ด้วย AS (alias)

เปลี่ยนชื่อคอลัมน์ด้วย AS (alias)

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

เช่น `SELECT Firstname as customer_name FROM customers;`

# ปฏิบัติ #3

- จงเขียนเปลี่ยนชื่อคอลัมภ์ Bytes เป็น track_size ใน table tracks

# เฉลยปฏิบัติ #3

- จงเขียนเปลี่ยนชื่อคอลัมภ์ Bytes เป็น track_size ใน table tracks

เฉลย

```
SELECT Bytes AS track_size FROM tracks;
```

# ปฏิบัติ #4

- ทดลองพิมพ์ แล้วดูผลลัพธ์

SELECT Bytes, Bytes/1000000 AS MB FROM tracks;

หมายเหตุ* 1 เมกะไบต์ = 1,000,000 ไบต์

# กำหนดจำนวนแถวที่อยากดึงข้อมูลออกมาด้วย LIMIT

กำหนดจำนวนแถวที่อยากดึงข้อมูลออกมาด้วย LIMIT

```
SELECT columnA, columnB
FROM table_name
LIMIT number of limit;
```

# ปฏิบัติ #5

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

จงกำหนดจำนวนผลลัพธ์ให้แสดงแค่ 10 แถว

# เฉลยปฏิบัติ #5

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

จงกำหนดจำนวนผลลัพธ์ให้แสดงแค่ 10 แถว

เฉลย

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

# นับจำนวนแถวทั้งหมดใน table นั้นด้วย COUNT

นับจำนวนแถวทั้งหมดใน table นั้นด้วย COUNT

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

เช่น SELECT COUNT(*) FROM customers;

# ปฏิบัติ #6

- จงเขียนภาษา SQL เพื่อหาว่า table invoices มีจำนวนแถวเท่าไหร่?

# เฉลยปฏิบัติ #6

- จงเขียนภาษา SQL เพื่อหาว่า table invoices มีจำนวนแถวเท่าไหร่?

เฉลย

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

# แสดงค่าที่ไม่ซ้ำกันเลยในคอลัมน์นั้น ๆ ด้วย DISTINCT

แสดงค่าที่ไม่ซ้ำกันเลยในคอลัมน์นั้น ๆ ด้วย DISTINCT

```
SELECT DISTINCT column _name
FROM table_name;
```

เช่น SELECT DISTINCT country FROM customers;

# ปฏิบัติ #7

- จงเขียนภาษา SQL เพื่อแสดงค่าที่ไม่ซ้ำกันเลยของคอลัมน์ composer ในตาราง tracks

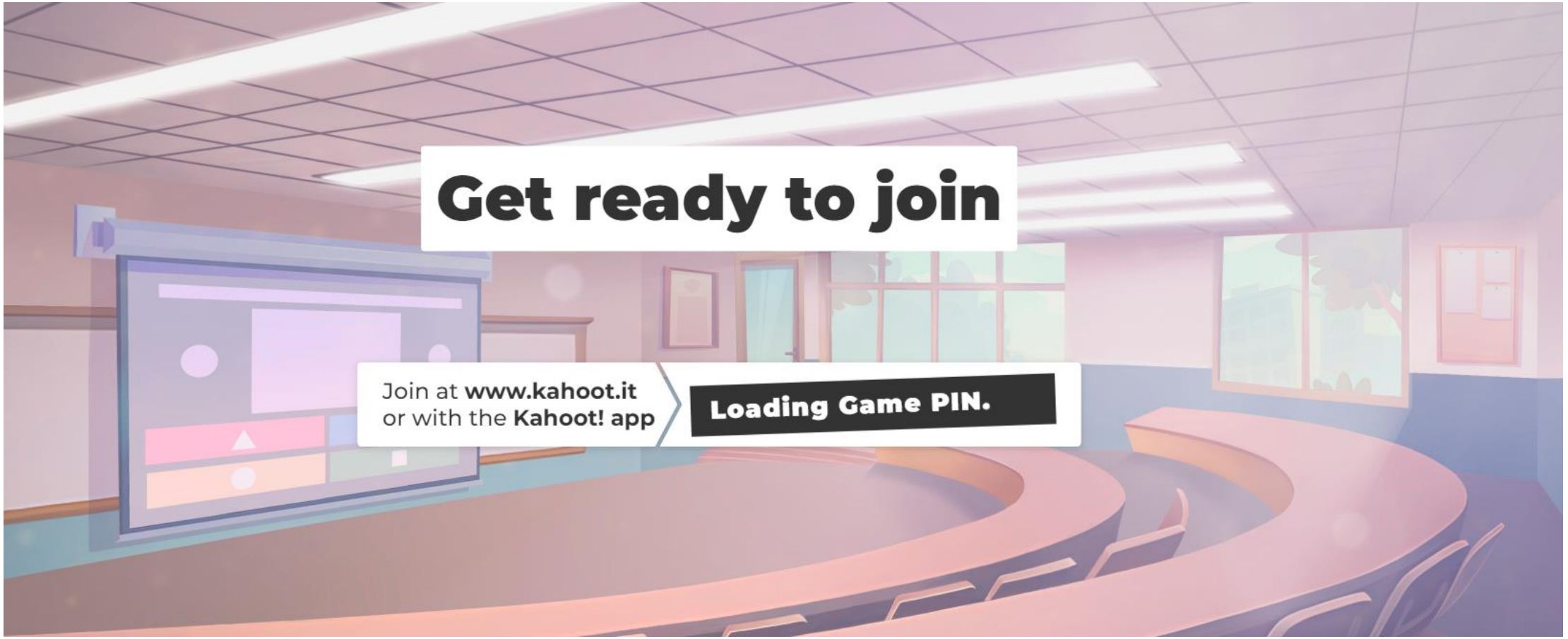
# เฉลยปฏิบัติ #7

- จงเขียนภาษา SQL เพื่อแสดงค่าที่ไม่ซ้ำกันเลยของคอลัมน์ composer ในตาราง tracks

เฉลย

```
SELECT DISTINCT composer
FROM tracks;
```

# Round #1 : Select (ข้อ 1-8)



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

# WHERE คืออะไร

WHERE คือการใส่เงื่อนไข (conditions) ในการกรองแถวที่เราต้องการ เช่น

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

ถ้าต้องการใส่ *มากกว่าหนึ่งเงื่อนไข* สามารถใช้ AND หรือ OR เข้ามาช่วยใน WHERE clause

# ปฏิบัติ #8

- เลือกทุกคอลัมน์จาก table invoices  กรองเฉพาะคอลัมภ์ total ที่ค่ามากกว่าหรือเท่ากับ 20

# เฉลยปฏิบัติ #8

- เลือกทุกคอลัมน์จาก table invoices  กรองเฉพาะคอลัมภ์ total ที่ค่ามากกว่าหรือเท่ากับ 20

เฉลย

```
SELECT *

FROM invoices

WHERE total >= 20;
```

# ปฏิบัติ #9

- เลือกทุกคอลัมน์จาก table invoices  กรองเฉพาะคอลัมภ์ total ที่ค่ามากกว่าหรือเท่ากับ 20 หรือ BillingCountry ที่เป็น USA

# เฉลยปฏิบัติ #9

- เลือกทุกคอลัมน์จาก table invoices  กรองเฉพาะคอลัมภ์ total ที่ค่ามากกว่าหรือเท่ากับ 20 หรือ BillingCountry ที่เป็น USA

เฉลย

```
SELECT *
FROM invoices
WHERE total >= 20 OR BillingCountry = 'USA';
```

# ปฏิบัติ #10

- จงเขียนคำสั่ง เลือกลูกค้าทุกคนที่อาศัยในประเทศ USA, Brazil, France จาก table customer

# เฉลยปฏิบัติ #10

- จงเขียนคำสั่ง เลือกลูกค้าทุกคนที่อาศัยในประเทศ USA, Brazil, France จาก table customer

เฉลย

```
SELECT * FROM customers WHERE Country='USA' or Country='Brazil' or Country='France';
```

สามารถเขียนในรูปแบบนี้ ด้วยการใช้ IN

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

# wildcard %

เราสามารถใช้ wildcard % เพื่อทำ pattern matching สำหรับคอลัมน์ที่เป็น text/string เช่น ชื่อลูกค้า ใน WHERE clause ได้

เช่น ฟิลเตอร์เฉพาะลูกค้าที่ชื่อขึ้นต้นด้วยตัว D

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

# ปฏิบัติ #11

- จงเขียนคำสั่ง SQL ฟิลเตอร์เฉพาะลูกค้าที่ใช้อีเมลล์ @gmail.com

# เฉลยปฏิบัติ #11

- จงเขียนคำสั่ง SQL ฟิลเตอร์เฉพาะลูกค้าที่ใช้อีเมลล์ @gmail.com

เฉลย

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

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

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

**เช่น** ฟิลเตอร์เฉพาะลูกค้าที่ไม่ระบุ company ใน table customers

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

# ปฏิบัติ #12

- จงเขียนคำสั่ง ฟิลเตอร์เฉพาะลูกค้าที่ระบุ Fax ใน table customers

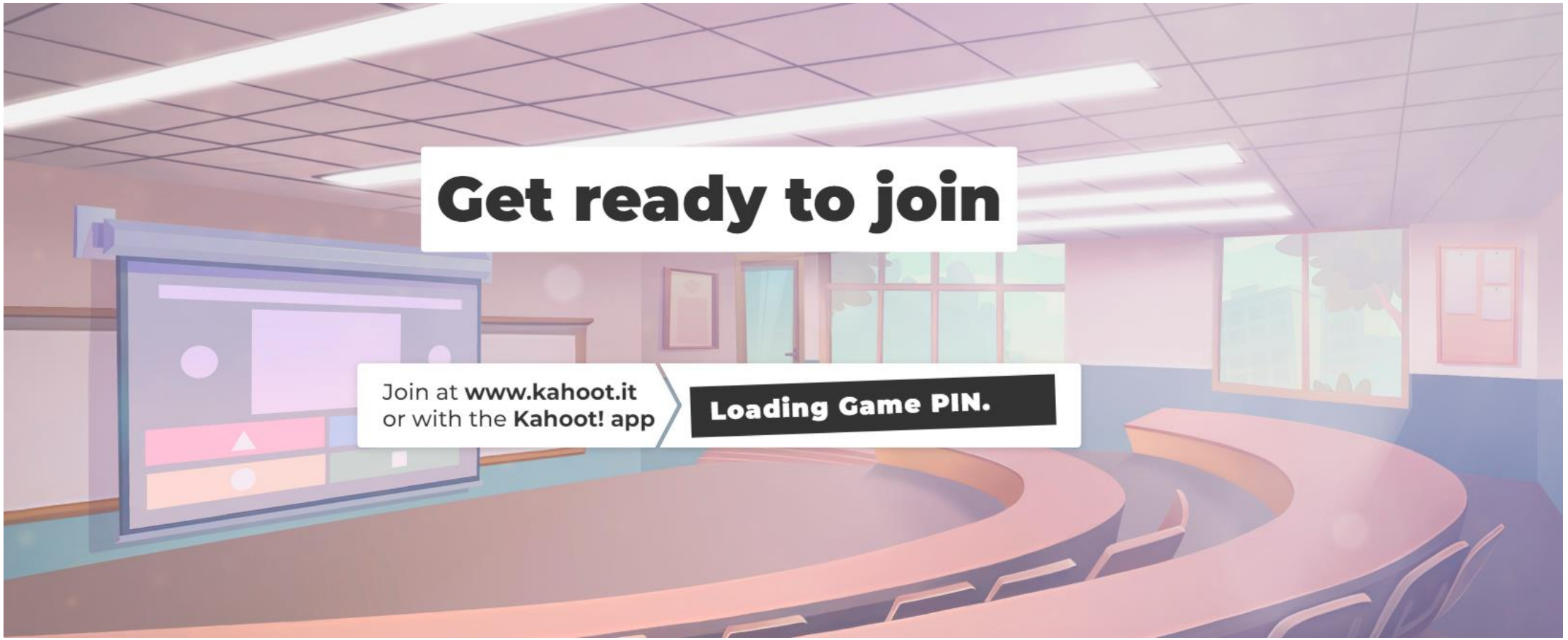
# เฉลยปฏิบัติ #12

- จงเขียนคำสั่ง ฟิลเตอร์เฉพาะลูกค้าที่ระบุ Fax ใน table customers

เฉลย

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

# Round #2 : WHERE (ข้อ 9-11)



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

# [3] ORDER BY

ตัวอย่าง เรียงข้อมูลแถว (rows) จากค่าน้อยไปมาก (Ascending)

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

ตัวอย่าง เรียงข้อมูลแถว (rows) จากค่ามากไปน้อย (Descending)

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

# หมายเหตุ*

เราสามารถใช้ ORDER BY กับคอลัมน์ที่เป็นตัวอักษร text/string

ได้โดย ascending order เรียงแถวจาก A-Z

และ descending order (DESC) เรียงแถวจาก Z-A

# ปฏิบัติ #13

- จงเขียนคำสั่งเรียงข้อมูลชื่อลูกค้า จาก A-Z

# เฉลยปฏิบัติ #13

- จงเขียนคำสั่งเรียงข้อมูลชื่อลูกค้า จาก A-Z

เฉลย

SELECT * FROM customers ORDER BY FirstName;

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

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

ใช้จับกลุ่มผล query ของเรา

เช่น

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

# ปฏิบัติ #14

- จงเขียนคำสั่งนับว่าเมืองที่ลูกค้าอาศัยมีเมืองใดบ้าง และแต่ละเมืองมีจำนวนลูกค้าเท่าใด

# เฉลยปฏิบัติ #14

- จงเขียนคำสั่งนับว่าเมืองที่ลูกค้าอาศัยมีเมืองใดบ้าง และแต่ละเมืองมีจำนวนลูกค้าเท่าใด

เฉลย

SELECT City, count(*) FROM customers GROUP BY City;

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

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

ใช้ฟิลเตอร์ผล query ที่ได้จากการ GROUP BY (ใช้เหมือนกับ WHERE clause)

ตัวอย่าง ฟิลเตอร์เอาเฉพาะประเทศที่มีลูกค้าตั้งแต่ 10 คนขึ้นไป

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

# ปฏิบัติ #15

- ฟิลเตอร์เอาเฉพาะเมืองที่มีลูกค้าตั้งแต่ 2 คนขึ้นไป

# เฉลยปฏิบัติ #15

- ฟิลเตอร์เอาเฉพาะเมืองที่มีลูกค้าตั้งแต่ 2 คนขึ้นไป

เฉลย

SELECT City, count(*) as n FROM customers GROUP BY City HAVING n >= 2;

# **เพิ่มเติม*** AGGREGATE functions ----------

# AGGREGATE functions

AGGREGATE functions คือ ฟังก์ชันที่ใช้คำนวณค่าสถิติเบื้องต้น ในคอลัมน์ที่เราเลือกมา เช่น

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

# ตัวอย่างการใช้งาน AGGREGATE functions

ตัวอย่าง

```
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;
```

# ปฏิบัติ #16

- จงเขียนหายอดขายต่ำสุด และสูงสุดของตาราง invoices โดยเปลี่ยนชื่อคอลัมภ์ *ข้อมูลยอดขายต่ำสุด* เป็น min และ ชื่อคอลัมภ์ *ข้อมูลยอดขายสูงสุด* เป็น max

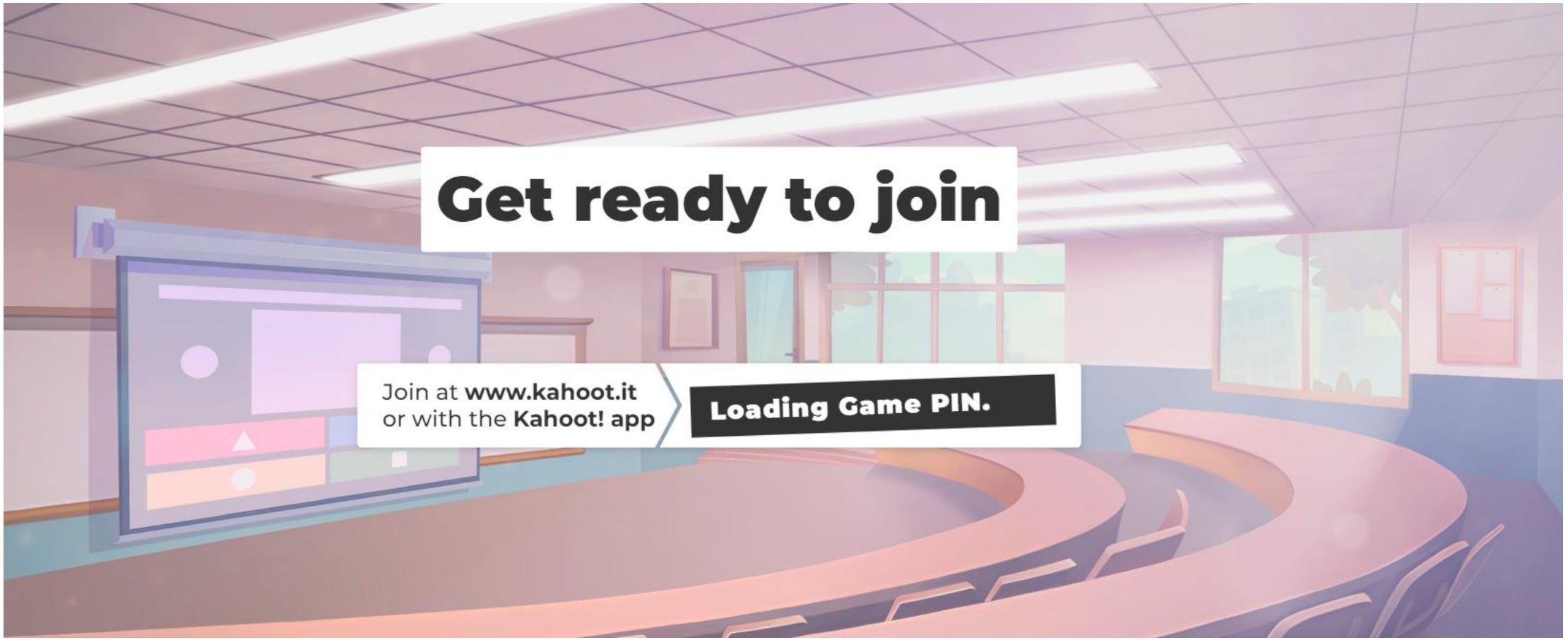
# เฉลยปฏิบัติ #16

- จงเขียนหายอดขายค่าต่ำสุด และสูงสุดของตาราง invoices โดยเปลี่ยนชื่อ *ยอดขายค่าต่ำสุด* เป็น min และ *ยอดขายค่าสูงสุด* เป็น max

เฉลย

```
SELECT
        MAX(total) AS max,
        MIN(total) AS min
 FROM invoices;
```

# Round #3 : GROUP BY, ORDER BY, HAVING (ข้อ 12-15)



# Reference

กษิดิศ สตางค์มงคล (2563). **SQL for data science.** กรุงเทพฯ: Datarockie