

EBOOK

PRACTICAL SQL

Data Analytics Series



ตัวอย่างหนังสือ Practical SQL เวอร์ชัน 1.1

เขียนและเรียบเรียงโดย DataRockie

25 ธ.ค. 2562

July
17

Update History

Practical SQL version 1.0 จำนวน 8 บท (24 ธ.ค. 2562)

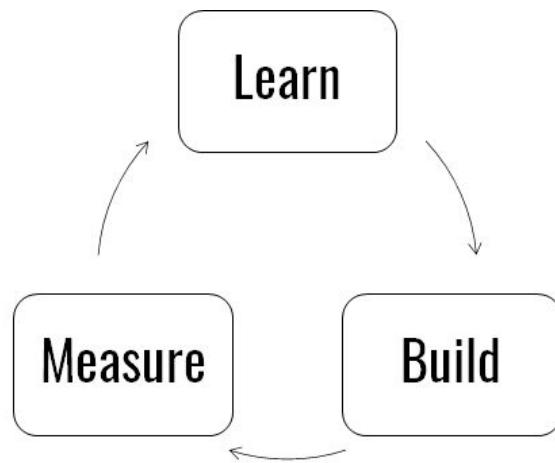
Practical SQL version 1.1 จำนวน 14 บท (25 ธ.ค. 2562)

- เพิ่มเติมเงื่อนไขในบท WHERE
- เพิ่มเติมเรื่อง Regular Expressions
- อัพเดท SQL Functions (Aggregate/ Value functions)
- อัพเดท GROUP BY
- อัพเดท HAVING
- อัพเดท ORDER BY
- อัพเดท JOIN
- อัพเดทการอัปโหลดไฟล์ CSV เข้าสู่ database

Important Note

หนังสือเล่มนี้ แอดตั้งใจเขียนขึ้นเพื่อเป็น Guide ให้กับทุกคนที่สนใจเรียนและพัฒนาทักษะ SQL สำหรับวิเคราะห์ข้อมูล โดยเฉพาะเพื่อนๆ ที่สนใจสมัครงานตำแหน่ง data analyst/ business & marketing analyst ของบริษัทต่างๆ

สไตล์การเขียน แอดใช้ภาษาที่เข้าใจง่าย กระชับ ตรงประเด็น และนำไปใช้ได้จริง



หลักการที่เราใช้สร้างหนังสือเล่มนี้คือ learn-build-measure หนังสือที่จะมีการอัพเดทจนกว่าตัวแอดคิดว่ามันจะดีที่สุดแล้วจาก feedback ของนักเรียนทุกคน เมื่อใน facebook app ที่มีการอัพเดทเวอร์ชัน 1.0 1.1 1.2 ... เพื่อสร้างประสบการณ์เรียนรู้ที่ดีที่สุดสำหรับนักเรียนของเราที่ลงทุนกับหนังสือเล่มนี้

นักเรียนที่ซื้อหนังสือเล่มนี้จะได้รับการอัพเดท/ ปรับปรุงเนื้อหาใหม่ที่ได้จาก feedback ของผู้อ่าน และจากประสบการณ์ของแอดที่เพิ่มขึ้นทุกวันเช่นกัน

หนังสือเวอร์ชันแรก v1.1 มีทั้งหมด 20 บท จะปล่อยให้นักเรียนทุกคนได้อ่านวันที่ 25 ธ.ค. 2562 นี้ ติดตามรายละเอียดได้ที่เพจบุ๊ค [DataRockie](https://datarockie.com) หรือทางเว็บไซต์ <https://datarockie.com>

ขอบคุณทุกคนที่เลือกและให้โอกาสหนังสือเล่มนี้ แอดหวังว่าหนังสือเล่มนี้จะช่วยให้ทุกคนเข้าใกล้ความฝันของตัวเองอีกกว่าหนึ่ง ไม่ว่าผู้อ่านจะเป็นอะไรก็ตาม

Live Your Dream

แอดทอย - Your Generalist Sensei

|f there's a book that you want to read, but it hasn't been written yet, then you must write it.”

ถ้ามีหนังสือที่เรารอ已久 อ่าน แต่ยังไม่มีใครเขียนมันขึ้นมา เราต้องเขียนมันขึ้นมาเอง

Toni Morisson

CONTENT

1. Why Learn SQL?
2. Download Software
 - a. DB Browser for SQLite
 - b. Sublime (Free text editor)
3. Understanding The ER Diagram
4. Connect to Database
5. Five Minutes Quick Start Guide
6. SELECT
 - a. SELECT *
 - b. SELECT columns
 - c. LIMIT
 - d. DISTINCT
 - e. Create new column
 - f. Rename column (AS)
7. WHERE
 - a. Boolean Operator
 - b. IN
 - c. BETWEEN AND
 - d. LIKE
 - e. IS NULL
 - f. NOT
 - g. Regular Expressions
8. When SQL meets Excel
 - a. Basic Pivot Table
9. Basic SQL Functions
 - a. Aggregate Function
 - b. Value Function
10. GROUP BY
 - a. One column
 - b. Two or more columns
11. HAVING
 - a. HAVING vs. WHERE
12. ORDER BY
 - a. Ascending/ Descending Order
 - b. One column
 - c. Two or more columns
13. JOIN
 - a. CROSS JOIN
 - b. INNER JOIN

- c. LEFT JOIN
- d. FULL JOIN
- e. SELF JOIN

14. Import CSV File to Database

สัญลักษณ์ที่เราใช้ในหนังสือเล่มนี้

-  ทิป/ เทคนิคสำคัญในการเขียน SQL queries
-  บทความหรือตัวอย่างเพิ่มเติมบนเว็บไซต์ที่เราคัดสรรมาให้แล้ว
-  แบบฝึกหัด SQL (พร้อมเฉลยท้ายเล่ม)
-  วีดีโอ YouTube อธิบาย concept สำคัญของพาร์ทนั้นๆ (เรื่องนี้)
-  อธิบายคอนเซ็ปต์ที่ค่อนข้าง advanced นักเรียนสามารถข้ามหัวข้อนั้นๆได้

SQL Clauses พื้นฐานที่ทุกคนต้องรู้

ชื่อ clause	ใช้ทำอะไร?
SELECT	ใช้เลือกคอลัมน์จาก table
WHERE	ใช้ฟิลเตอร์ records หรือแຄוที่ต้องการจาก table
COUNT	Aggregate function นับจำนวน records
AVG	Aggregate function หาค่าเฉลี่ย
SUM	Aggregate function หาผลรวม
MIN	Aggregate function หาค่าต่ำที่สุด
MAX	Aggregate function หาค่าสูงที่สุด
GROUP BY	จับกลุ่มข้อมูล ใช้ร่วมกับ Aggregate functions
HAVING	ฟิลเตอร์กลุ่มที่ต้องการ
ORDER BY	เรียงข้อมูลจากน้อยไปมาก (ascending) หรือมากไปน้อย (descending)
LIMIT	กำหนดจำนวนแຄวที่ต้องการ
JOIN	ใช้ร่วมกับ SELECT เพื่อดึงข้อมูลจากหลายๆ tables พร้อมกัน
CASE	ใช้เขียนเงื่อนไข (เหมือนฟังชั่น IF ของ Excel)
CAST	ใช้เปลี่ยนประเภทของข้อมูล i.e. data types

Chapter 1 - Why Learn SQL

บทที่หนึ่ง เราจะมาพูดกันเรื่อง motivation และเหตุผลที่ทุกคนควรเขียน SQL ให้เป็น



เหตุผลสั้นๆ คือ SQL เป็นทักษะที่ขาดไม่ได้เลยสำหรับงานด้าน data ในยุคปัจจุบัน

1.1 เมื่อ Excel ไม่เพียงพอในยุคปัจจุบัน

นักเรียนชอบถามว่า แอดเรียนจบด้าน data science มาตรงๆ หรือเปล่า?

จริงๆ แอดเรียนจบบริษัทตระดับเศรษฐศาสตร์ จากมหาวิทยาลัยเกษตรศาสตร์ตอนปี 2010 สมัยนั้นเวลาไปสมัครงานต้องมี Excel (และ Word/ PowerPoint) เป็นทักษะที่ขาดไม่ได้ เอาจริง มี skill แค่นั้นก็หางานได้ไม่ยากในยุคก่อน

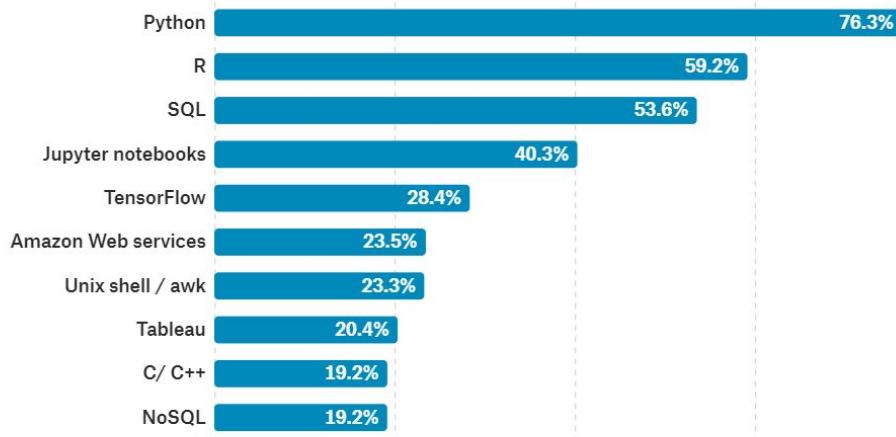
ปัจจุบันผ่านมาเกือบสิบปี โลกเรามีการเปลี่ยนแปลงแบบก้าวกระโดด แต่ไปสมัครงานของเด็ก พ.ศรี/ โท ส่วนใหญ่ยังใส่แค่ Excel อยู่เลย (บางคนยังใช้ Excel ไม่ได้เลย) ซึ่งนี่ถือเป็น Skill Gap ขนาดใหญ่ที่ทำให้หลาย คนเสียโอกาสในการได้งานที่ตัวเองต้องการ

I know, I know. Life is hard.

1.2 แล้วทักษะอะไรที่จะมาปิด Gap นี้ได้

คำตอบของคำถามข้อนี้ อยู่ในชื่อหนังสือเล่มนี้แล้ว (hint: SQL)

[Kaggle](#) ได้ทำแบบสำรวจ The State of ML and Data Science ในปี 2017 ได้รับการตอบรับจาก data professionals จำนวน 16,000 คนทั่วโลก ผลสำรวจพบว่าเครื่องมือที่คนใช้กันทุกวันสาม อันดับแรกคือ Python 76.3% R 59.2% และ SQL 53.6% ตามลำดับ



ถ้าม่วง Excel ยังติดอยู่ใน list นี้ใหม่ ติด! แต่ว่าตกไปอยู่อันดับที่ 15 มีคนใช้น้อยกว่า 15% ข้อมูลนี้ให้ insight เราสองอย่าง

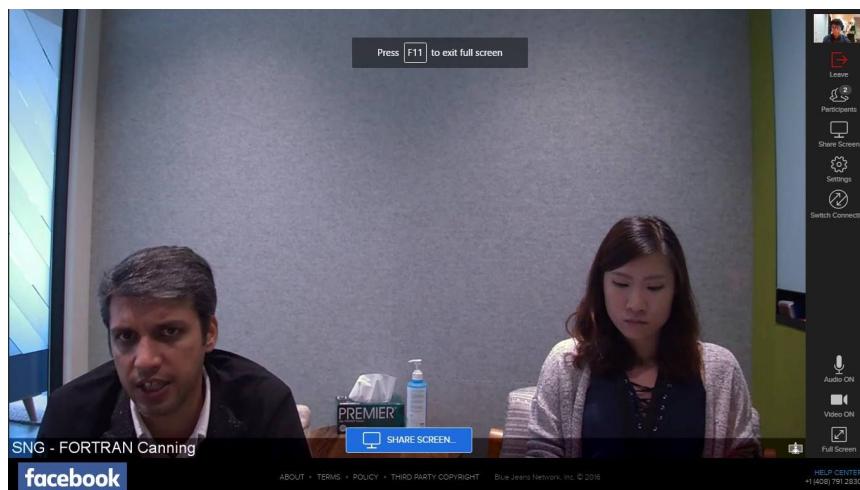
1. เครื่องมือที่คนส่วนใหญ่ (โดยเฉพาะด้าน data) ใช้กันในชีวิตประจำวันเป็นแบบ scripting/ programming language i.e. เขียนโค้ดเองเกือบหมดเลย เพราะการเขียนโค้ดเองมีความยืดหยุ่นกว่า และใช้ทดแทน Excel ได้หลายอย่าง
2. Excel/ Spreadsheet ถ้าม่วงยังจำเป็นไหม? จำเป็น แต่ความนิยมก็ลดลงไปพอสมควร สาเหตุหนึ่งอาจจะเป็นเพราะสื่อในปัจจุบันพวยยามขาย Python/ R กันเยอะเหลือเกิน ทีมที่ยังใช้ Spreadsheet เยอะๆอยู่คือทีม Business Intelligence และ Data Analyst

✓ เวอร์ชั่นใหม่ของ Excel 2019/ 2020 เรียกว่าโหมดมาก เริ่มมีการใช้พวก array อย่างเต็มรูปแบบ i.e. คอนเซปต์เดียวกับ vectorization ใน R/ Python ทำให้การวิเคราะห์ข้อมูลด้วย Excel มีประสิทธิภาพสูงขึ้นอย่างมาก ส่วนตัวแอดคิดว่า Excel น่าจะกลับมาทาง mind share ของเหล่า data analyst ได้ไม่ยาก

สำหรับนักเรียนที่อยากรเข้ามาเริ่มงานสาย data ภาษาแรกที่ควรจะเรียนคือ SQL เพราะว่าเรียนรู้ง่ายที่สุด (อันนี้พูดจริง ทำไม่มันง่ายอย่างนี้ 555+) และมีประโยชน์ในการทำงานจริง

1.3 ครั้งแรกที่แอดรู้จัก SQL

ปี 2016 แอดมีโอกาสสัมภาษณ์งานกับ Facebook Singapore ตำแหน่ง Measurement Lead (ของประเทศไทย แต่หัวหน้าเหมือนอยู่ที่สิงคโปร์) ในรูปด้านล่างเป็นตอนสัมภาษณ์แบบ con call โดยตำแหน่งงานนี้ require skills แค่สองอย่างเอง หลักๆคือความรู้ด้านการตลาด และ (guess what!) ต้องเขียน SQL ให้เป็น i.e. ใน job description เขียนว่า “proficient in SQL”



สัมภาษณ์งาน Facebook คือจุดเริ่มต้นให้แอดเริ่มศึกษา SQL

ก่อนสัมภาษณ์งาน บอกตรงๆ เลยว่า แอดเดย์นไม่เป็นเลย ตอนนี้รีบไปลงคอร์สออนไลน์หลายๆ ตัว อาย่าง [SQL for Data Science](#) ของ coursera และคอร์ส SQL ของ [Jose Portilla](#) บน Udemy

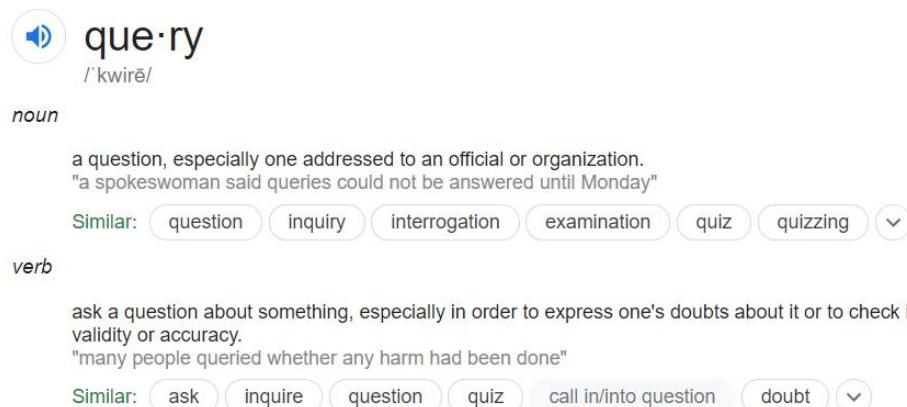
หลังจากเรียนจบ แอดก็พบว่า SQL ง่ายกว่าเขียน R/ Python เยอะเลย รู้สึกเรียนนานแล้ว 555+

ปล. ตอนนั้นแอดได้งานที่ดีแทค เลยตัดสินใจอยู่ไทยดีกว่า เพราะว่าขี้เกียจเดินทางไปมา และเราทำเพื่อยู่ด้วย อยู่ไทยสอนหนังสือ่ง่ายกว่า

I know, right? It's never too late to learn.

1.4 SQL คืออะไร

SQL ย่อมาจาก **Structured Query Language** พัฒนาโดย IBM ตั้งแต่ยุค 1970s (ใช้กันมาเกือบ 50 ปีแล้ว) เป็นภาษาทางการที่เราใช้ทำงานกับ Relational Databases ทั่วโลก



[Query](#) (verb) แปลว่าการตั้งคำถาม

SQL จริงๆคือการนำศัพท์ภาษาอังกฤษมาเขียนต่อ กันเป็น statement (หรือที่เราเรียกว่า query อ่านว่า “คิวรี่”) การเขียน query คือการเขียนคำสั่งไปถูก database ถ้า database เข้าใจคำสั่งของเรา มันก็จะส่งคำตอบกลับมาให้อย่างถูกต้อง ตัวอย่างคำสั่งที่ตอบได้ด้วยการเขียน SQL query

- จำนวนลูกค้าที่สมัครใช้บริการของเราตั้งแต่เดือนกรกฎาคมที่ผ่านมา (new customers)
- จำนวนลูกค้าที่เลิกใช้บริการของเราในเดือนธันวาคม (churn)
- รายได้ รายจ่าย และกำไร จากการดำเนินธุรกิจของบริษัท (revenue, expense, profit)
- ลูกค้า 100 คนแรกที่ซื้อสินค้าจากบริษัทของเรามากที่สุดในปี 2019 (top customers)
- และคำสั่งอื่นๆ ทาง business อีกมาก many

ถ้าเราเขียน SQL เป็น เราจะสามารถตอบคำถามทาง business ได้เกือบทั้งหมดเลย (ถ้าเรามีการเก็บข้อมูลเรื่องนั้นๆไว้แล้ว) และปกติ CEO บริษัทชอบคนที่ตอบคำถามแก่ได้ ว่ามั้ย? 😊

1.5 ตัวอย่างของ SQL query

หนังสือเล่มนี้ตั้งใจจะสอนทุกคนเขียน SQL ตั้งแต่ระดับเริ่มต้น (จับมือเขียน!) ไปจนถึงการเขียน Advanced Query เพื่อวิเคราะห์ข้อมูลอย่าง data analyst มืออาชีพ

```
SELECT firstname, lastname, email  
FROM customers  
WHERE country = 'USA' OR country = 'Belgium';
```

แค่อ่านจบบท SELECT และ WHERE (บทที่ 6-7) ทุกคนก็สามารถอ่านและเขียน query พื้นฐานเป็นแล้ว จริงๆยังไม่ต้องสอนเลย ทุกคนก็น่าจะเดาได้ว่า query ด้านบนกำลังทำอะไร?

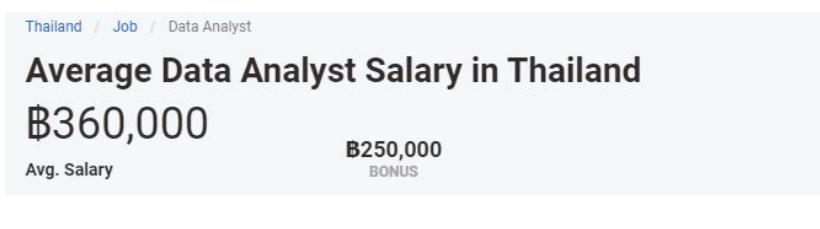
เดาแล้วใช่มั้ย? ไม่ต้องตอบก็ได้ แอดรูว่าทุกคนเดาถูก :P

1.6 เงินเดือนเฉลี่ยของ Data Analyst ในประเทศไทย

✓ ปกติแอ็อดไม่ค่อยชอบเอาเงินมาเป็น motivation เท่าไหร่ (เงินเป็นได้แค่ short term goal) แต่ถ้าเขียน SQL ได้โอกาสที่จะได้ปรับตำแหน่ง หรือย้ายสายงานเพื่อเพิ่มเงินเดือนก็จะมีสูงขึ้น

อ้างอิงจากเว็บไซต์ payscale (2019) เงินเดือนเฉลี่ย i.e. median salary ของ data analyst ในประเทศไทยอยู่ที่ 360,000 บาทต่อปี (เฉลี่ยเดือนละ 30,000 บาท) และสูงสุดประมาณ 2.0 ล้านบาทต่อปี (เฉลี่ยเดือนละ 160,000 บาท)

จากรูปด้านล่าง คนที่ได้เงินเดือนระดับสองล้านมีประมาณ 10% ของกลุ่ม data analyst ที่ payscale เก็บข้อมูลมาเท่านั้น (เงินเดือนยิ่งสูง ความรับผิดชอบยิ่งเยอะ)



ที่มา https://www.payscale.com/research/TH/Job=Data_Analyst/Salary

1.7 เทคนิคการเรียน SQL ให้เป็นเร็วๆ

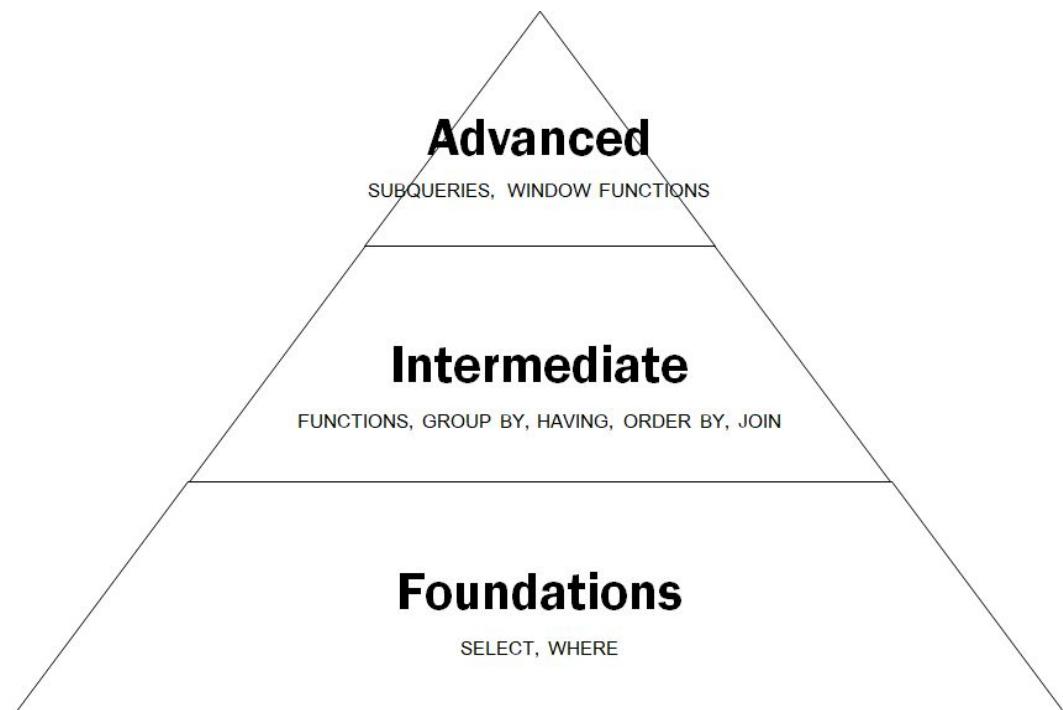
แออดก็อยากให้มันมีทางลัดเหมือนกัน แต่ความจริงคือไม่มีทางลัดในการเรียนห稻 กวิธีเดียวที่จะเขียน SQL ให้เป็นเร็วๆ คือลงมือเขียนมันด้วยตัวเอง แล้วอึกอย่างคือ อะไรที่มันได้มาง่ายๆ มันไม่ค่อยมีความหมายกับชีวิตเท่าไหร่ห稻 ว่ามั้ย?

ถ้าอยากรู้จะเก่งกว่านี้ ก็ต้องเริ่มทำอะไรที่อยู่นอก comfort zone ของตัวเองบ้าง หนังสือเล่มนี้จะพาทุกคนออกจาก comfort zone เอง 555+



อะไรที่ว่ายาก ถ้าแอดสอน ก็ง่ายหมดเลย 555+ Seriously, You have my words.

แออดแบ่งเนื้อหา SQL ที่ควรเรียนเป็นสามส่วน นักเรียนจะได้รู้ว่าตอนนี้ตัวเองอยู่ระดับไหนแล้ว



SQL สำหรับงาน data analyst จริงๆ แบ่งได้เป็นสามระดับเอง

- Foundations คือพื้นฐานสำคัญ และเป็นพาร์ที่ง่ายที่สุดเลย ใช้เวลาประมาณ 1-2 ชั่วโมง (ต้องเปิดคอมมาเขียนโค้ดด้วยนะ ไม่ใช่อ่านหนังสืออย่างเดียว)
- Intermediate เราจะเรียนวิธีการสรุปผลข้อมูลด้วยสถิติ จับกลุ่ม เรียงข้อมูล และดึงข้อมูลจากหลายๆ tables พร้อมกัน (ภาษา SQL เรียกว่า JOIN tables “จอย เทเบิล”)
- Advanced จะเน้นเรื่องเทคนิคที่ทำให้ SQL queries ของเรารีบซ่อนขึ้นนิดหน่อย แต่ก็จะทำอะไรกับข้อมูลได้มากขึ้น เช่น การเขียน query ซ้อน query หรือที่เราเรียกว่า subqueries เป็นต้น

Chapter 2 - Download Software

SQL เวอร์ชันที่แอดใช้สอนในหนังสือเล่มนี้คือ SQLite

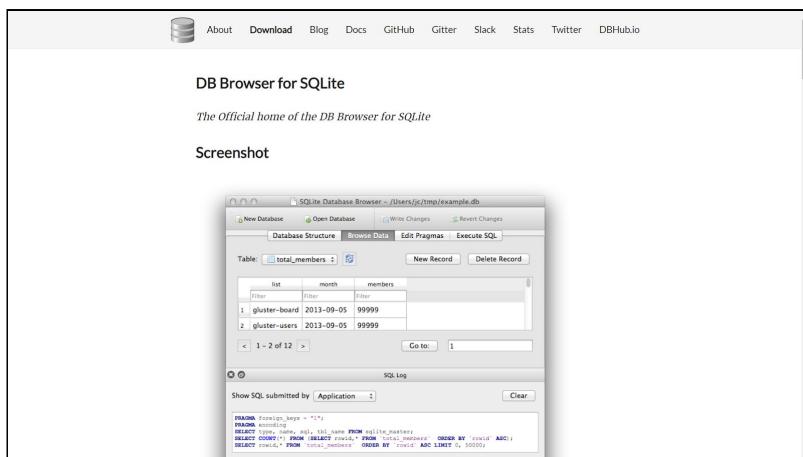
ทำไมถึงใช้ SQLite? เพราะว่ามันติดตั้งและใช้งานง่ายที่สุดเปรียบเทียบกับบรรดา SQL ทั้งหมดที่มีอยู่ในตลาดตอนนี้เลย โดยตัวอย่าง SQL queries ที่อยู่ในหนังสือเล่มนี้สามารถนำไปประยุกต์ใช้กับ database อื่นๆได้ประมาณ 90%

✓ SQL ของแต่ละเจ้า จะมีความแตกต่างกันอยู่นิดหน่อย แต่ละบริษัทพยายามจะเพิ่มฟีเจอร์และความสามารถใหม่ๆให้กับ SQL ที่เค้าพัฒนาขึ้นมา เจ้าดังๆในตลาด open source เช่น MySQL, PostgreSQL, MariaDB, SQLite และ Microsoft SQL Server Express Edition ถ้าเป็นบริษัทใหญ่ๆ อาจจะใช้ licensed software เช่น Teradata เป็นต้น (แต่ค่าใช้จ่ายสูงมาก)

นักเรียนสามารถดาวน์โหลด DB Browser (SQLite Client) สำหรับเขียน SQLite query ได้ฟรีจากเว็บไซต์ <https://sqlitebrowser.org/> ติดตั้งได้ทั้ง Windows และ macOS

2.1 ขั้นตอนการดาวน์โหลดและติดตั้งโปรแกรมสำหรับ Windows

1. เข้าไปที่เว็บไซต์ <https://sqlitebrowser.org/>
2. คลิกที่ลิงค์ Download



เข้าไปที่เว็บไซต์ sqlitebrowser เพื่อดาวน์โหลดโปรแกรมพรี

3. เลือกเวอร์ชันที่เหมาะสมกับ OS ของคอมพิวเตอร์ที่เราใช้ สำหรับ Windows 64bit ให้ดาวน์โหลดไฟล์ DB Browser for SQLite - Standard installer for 64-bit Windows ตามรูปด้านล่าง (ขนาดไฟล์ประมาณ 15 mb)

Downloads

Windows

Our latest release (3.11.2) for Windows:

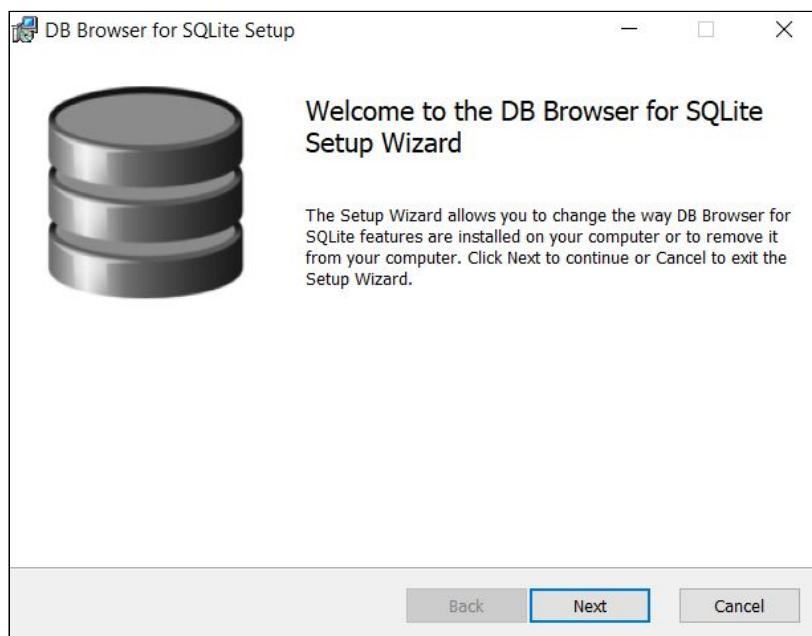
- DB Browser for SQLite – Standard installer for 32-bit Windows & Windows XP
- DB Browser for SQLite – .zip (no installer) for 32-bit Windows & Windows XP
- DB Browser for SQLite – Standard installer for 64-bit Windows
- DB Browser for SQLite – .zip (no installer) for 64-bit Windows
- DB Browser for SQLite – PortableApp

Note – If for any reason the standard Windows release does not work (e.g. gives an error), try a nightly build (below).

Nightly builds often fix bugs reported after the last release. 😊

ดาวน์โหลดตัว installer ที่เหมาะสมกับระบบปฏิบัติการของคอมพิวเตอร์ที่ใช้

4. เปิดไฟล์ที่ดาวน์โหลดมา และติดตั้งโปรแกรมตามปกติ (กด next next OK ได้เลยโดยไม่ต้องเปลี่ยน default setting)

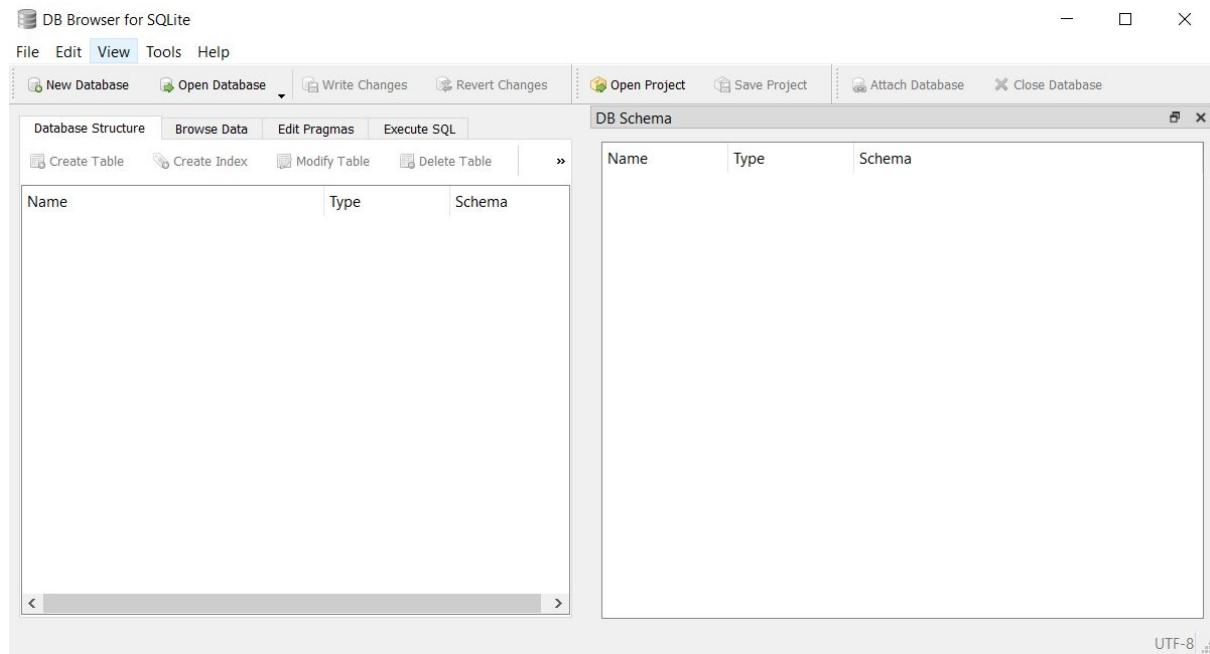


ติดตั้งโปรแกรมตามปกติ

5. เมื่อติดตั้งโปรแกรมเสร็จแล้ว ไปที่ Start พิมพ์ “DB Brower” เพื่อเปิดโปรแกรมขึ้นมา



รูปไอคอนของโปรแกรม DB Browser



หน้าตา interface ของ DB Browser สำหรับเขียน SQL

✓ วิธีการเช็คว่าคอมพิวเตอร์ของเราใช้ระบบ Windows 32 หรือ 64 bit? ให้เราคลิกขวาที่ My Computer (บนหน้าจอ Desktop) และเลือก Properties ให้ดูที่ System Profile ตามรูปด้านล่าง

System

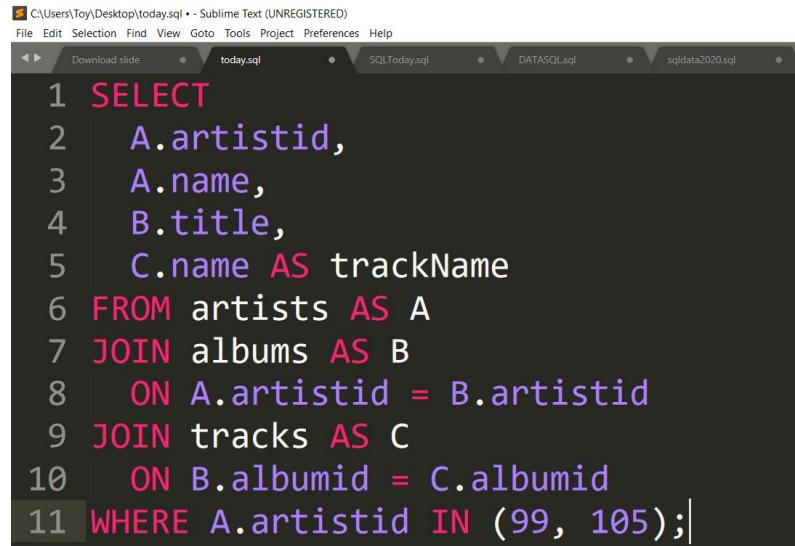
Processor:	Intel(R) Core(TM) i7-8565U CPU @ 1.80GHz 1.99 GHz
Installed memory (RAM):	20.0 GB (19.8 GB usable)
System type:	64-bit Operating System, x64-based processor
Pen and Touch:	No Pen or Touch Input is available for this Display

2.2 ขั้นตอนการดาวน์โหลดและติดตั้งโปรแกรมสำหรับ macOS

ไปที่หน้าดาวน์โหลด <https://sqlitebrowser.org/> และเลือกไฟล์ติดตั้งของ mac และติดตั้งโปรแกรมตามปกติ

2.3 ดาวน์โหลด Free Text Editor

อีกหนึ่งซอฟต์แวร์ที่อยากรู้ว่าทุกคนดาวน์โหลดด้วยคือ [Sublime Text](#) ไว้ใช้เปิดดูและเขียนโค้ด SQL queries ได้อย่างรวดเร็ว ใช้งานง่าย มี syntax highlight ให้ด้วย แฉมใช้งานฟรี



```
C:\Users\Toy\Desktop\today.sql - Sublime Text (UNREGISTERED)
File Edit Selection Find View Goto Tools Project Preferences Help
Download slide today.sql SQLToday.sql DATA_SQL.sql sql\data2020.sql
1 SELECT
2     A.artistid,
3     A.name,
4     B.title,
5     C.name AS trackName
6 FROM artists AS A
7 JOIN albums AS B
8     ON A.artistid = B.artistid
9 JOIN tracks AS C
10    ON B.albumid = C.albumid
11 WHERE A.artistid IN (99, 105);
```

ตัวอย่างการเขียน SQL query ในโปรแกรม Sublime

เข้าไปที่เว็บไซต์ <https://www.sublimetext.com/> และคลิกดาวน์โหลดได้เลย ติดตั้งได้ทั้ง Windows และ macOS เช่นเดียวกัน

✓ ส่วนตัวแอดชอบใช้ Sublime เวลาสอน ใช้งานง่ายและมีไฮไลท์ keyword ของหลายภาษาครบทั้ง R, Python, SQL และอีกมากmany อีกโปรแกรมนึงที่แอดมีติดเครื่องคือ [Visual Studio Code](#) ของ Microsoft อันนี้ก็เป็น free text editor ที่ดีมากๆ ฟีเจอร์จัดเต็ม

2.4 ดาวน์โหลดไฟล์ example database

นักเรียนสามารถเข้าไปที่ [Google Drive](#) ของหนังสือเล่มนี้ ดาวน์โหลดไฟล์ example database และ ER diagram มา save ไว้ในคอมพิวเตอร์ แอดแนะนำว่าเซฟไว้ที่หน้า Desktop เลย จะได้หาเจอง่ายๆ

บทต่อไป แอดจะอธิบายวิธีการอ่าน ER diagram (Entity-Relationship Diagram) นักเรียนจะได้เข้าใจข้อมูลและความสัมพันธ์ต่างๆที่มีอยู่ใน database ของเรา

Chapter 3 - Understanding The ER Diagram



ถ้าเป็นครั้งแรกที่เรียนเกี่ยวกับ database อย่าข้ามเนื้อหาในบทนี้ ทำความเข้าใจเนื้อหาทั้งหมดของบทสาม ก่อนจะเริ่มเขียน SQL query

3.1 ER Diagram คืออะไร?

ER Diagram ย่อมาจากคำว่า Entity-Relationship Diagram ใช้แสดงความสัมพันธ์ของตารางทั้งหมดในฐานข้อมูล รูปด้านล่างคือ ER Diagram ของ chinook database ที่แอดใช้สอนในหนังสือเล่มนี้ ดาวน์โหลด ER Diagram แบบ PDF ได้[ที่นี่](#)



สิ่งที่ทุกคนต้องรู้เกี่ยวกับ ER Diagram ของ chinook database

- ฐานข้อมูลนี้เกี่ยวกับธุรกิจเพลง มีทั้งหมด 11 ตาราง (table)
- ภาษาเฉพาะของ database เราเรียกตารางเหล่านี้ว่า “Entity”
- ในแต่ละตารางจะแสดงชื่อคอลัมน์ เช่น ตาราง customer มีทั้งหมด 13 คอลัมน์ เราเรียกคอลัมน์เหล่านี้ว่า “Attribute”
- แต่ละคอลัมน์จะมีประเภทข้อมูลกำกับไว้ด้วย (สีเทาอ่อน) ใน SQLite แบ่งข้อมูลออกเป็นสามประเภทหลักๆคือ integer ตัวเลขจำนวนเต็ม, numeric ตัวเลขที่มีเศษนิยม และ text ตัวอักษร/ข้อความ ส่วนคอลัมน์ datetime ใน SQLite จะเป็น text รูปแบบหนึ่ง

- “Relationship” หรือความสัมพันธ์ของแต่ละ entity ถูกแสดงด้วยเส้นสีดำ ที่ลากเข้ามายังตารางต่างๆเข้าด้วยกัน

ตอนนี้ทุกคนรู้แล้วว่าทำไม่เราถึงเรียกฐานข้อมูลประเภทนี้ว่า **Relational Database** เพราะตารางทั้งหมดมีความสัมพันธ์เชื่อมกันอยู่ i.e. แสดงความสัมพันธ์ของธุรกิจหนึ่งๆ เช่น ลูกค้า (customers) ซื้อสินค้าวันไหน (invoices) และซื้อเพลงของศิลปินคนไหนบ้าง (albums/ artists)

customers	
customerid:	integer
firstname:	text
lastname:	text
company:	text
address:	text
city:	text
state:	text
country:	text
postalcode:	text
phone:	text
fax:	text
email:	text
supportrepid:	integer

customers table มีคอลัมน์ customerid เป็น primary key

สังเกตในแต่ละตารางจะมีคอลัมน์หนึ่งที่มีไอคอนแม่กุญแจสีเหลืองอยู่ โดยคอลัมน์เหล่านี้ถูกเรียกว่า “**Primary Key**” ลักษณะสำคัญของ primary key คือ value ในคอลัมน์นี้จะไม่ซ้ำกันเลย เหมือนเลขบัตรประชาชน เช่น 001, 002, 003, 004, ...

มีข้อยกเว้นเดียวในตาราง playlist_track ที่มีสองคอลัมน์ {playlist_id, trackid} และทั้งสองคอลัมน์มีแม่กุญแจห้องคู่ กรณีนี้เรียกว่า “**Composite Key**” ถ้าเราเชื่อม (merge) สองคอลัมน์นี้เข้าด้วยกัน เราจะได้ primary key คอลัมน์ใหม่ที่มีค่าไม่ซ้ำกันเลย

ตอนนี้อยากรู้แล้วว่า Primary Key คือคืออะไรที่เราใช้ในการ JOIN หรือหรือดึงข้อมูลจากหลายตารางพร้อมกัน และจะอธิบายการเขียน JOIN อย่างละเอียดในหนังสือฉบับเต็มบทที่ 13

3.2 ตัวอย่างข้อมูลใน customers table

ตาราง customers เก็บข้อมูลลูกค้าแต่ละคนของบริษัทเรา มี customerid เป็น primary key

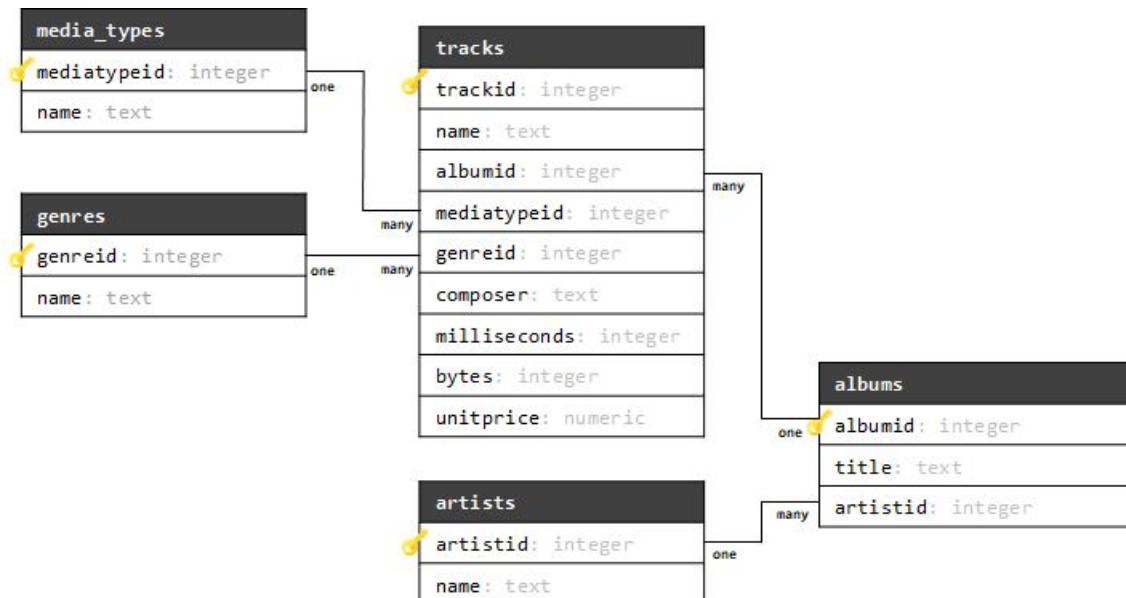
	CustomerId	FirstName	LastName	Company	Address	City	State	Country	PostalCode	Phone
	Filter	Filter	Filter	Filter	Filter	Filter	Filter	Filter	Filter	Filter
1	1	Luís	Gonçalves	Embraer - Empres...	Av. Brigadeiro Fari...	São José dos Cam...	SP	Brazil	12227-000	+55 (12) 3923-5555
2	2	Leоні	Köhler	NULL	Theodor-Heuss-Str...	Stuttgart	NULL	Germany	70174	+49 0711 2842222
3	3	François	Tremblay	NULL	1498 rue Bélanger	Montréal	QC	Canada	H2G 1A7	+1 (514) 721-4711
4	4	Bjørn	Hansen	NULL	Ullevålsveien 14	Oslo	NULL	Norway	0171	+47 22 44 22 22
5	5	František	Wichterlová	JetBrains s.r.o.	Klanova 9/506	Prague	NULL	Czech Republic	14700	+420 2 4172 5555
6	6	Helena	Holý	NULL	Rilská 3174/6	Prague	NULL	Czech Republic	14300	+420 2 4177 0449
7	7	Astrid	Gruber	NULL	Rotenturmstraße ...	Vienne	NULL	Austria	1010	+43 01 5134505
8	8	Daan	Peeters	NULL	Grétrystraat 63	Brussels	NULL	Belgium	1000	+32 02 219 03 03

3.3 รูปแบบความสัมพันธ์ใน relational database

ความสัมพันธ์ที่เรานิยมใช้กับ relational database จะมีอยู่สามแบบหลักๆ ประกอบด้วย

1. one to one
2. one to many
3. many to many

ลองศึกษารูปแบบความสัมพันธ์ของ diagram ด้านล่าง



รูปแบบความสัมพันธ์ที่ใช้เยอะที่สุดใน database คือ one to many

จากรูปจะเห็นว่าความสัมพันธ์ทั้งหมดเป็นแบบ one to many relationship ซึ่งเป็นรูปแบบความสัมพันธ์ที่นิยมใช้มากที่สุดในการสร้าง relational database วิธีการอ่าน ER Diagram แบบนี้ก็

ตรงไปตรงมา เช่น ศิลปินหนึ่งคนหรือวง (one) สามารถมีได้มากกว่าหนึ่งอัลบัม (many) และหนึ่งอัลบัม (one) สามารถมีได้มากกว่าหนึ่งเพลง (many) เป็นต้น

รูปด้านล่างแสดงความสัมพันธ์ทั้งหมดของ chinook database



รูปแบบความสัมพันธ์ทั้งหมดของ chinook database

⚠️ ทฤษฎีการออกแบบฐานข้อมูล (database design) อยู่นอกเหนือจาก scope ของหนังสือเล่มนี้ แต่จะโฟกัสที่การเขียน SQL queries เพื่อดึงและวิเคราะห์ข้อมูลจาก database เท่านั้น

💡 อ่านเพิ่มเติมเรื่อง ER Diagram ของ chinook database เขียนโดย [SQLite Tutorial](#)

✓ chinook เป็นตัวอย่าง database ที่เรียบง่าย มีแค่ 11 ตาราง ในชีวิตจริงเราอาจต้องทำงานกับ database ที่มีขนาดใหญ่ อาจจะมากกว่า 100 ตาราง หรือมากกว่า 1,000 ตารางเลยก็ได้ (ยิ่งธุรกิจมีความซับซ้อนมากเท่าไหร่ database ก็จะยิ่งซับซ้อนมากขึ้นเท่านั้น)

✓ ความยากของการวิเคราะห์ข้อมูลด้วย SQL คือต้องเข้าใจการทำงานของธุรกิจ รู้ว่าข้อมูลถูกเก็บอยู่ที่ไหน และความสัมพันธ์ของแต่ละตารางเป็นอย่างไร ในชีวิตจริงบริษัทส่วนใหญ่ไม่ได้มีการพัฒนาหรืออัพเดท ER Diagram ไว้ ทำให้ data analyst ต้องอาศัยการลองผิดลองถูกดึงข้อมูล เก็บประสบการณ์ และเรียนรู้ระบบฐานข้อมูลด้วยตัวเอง

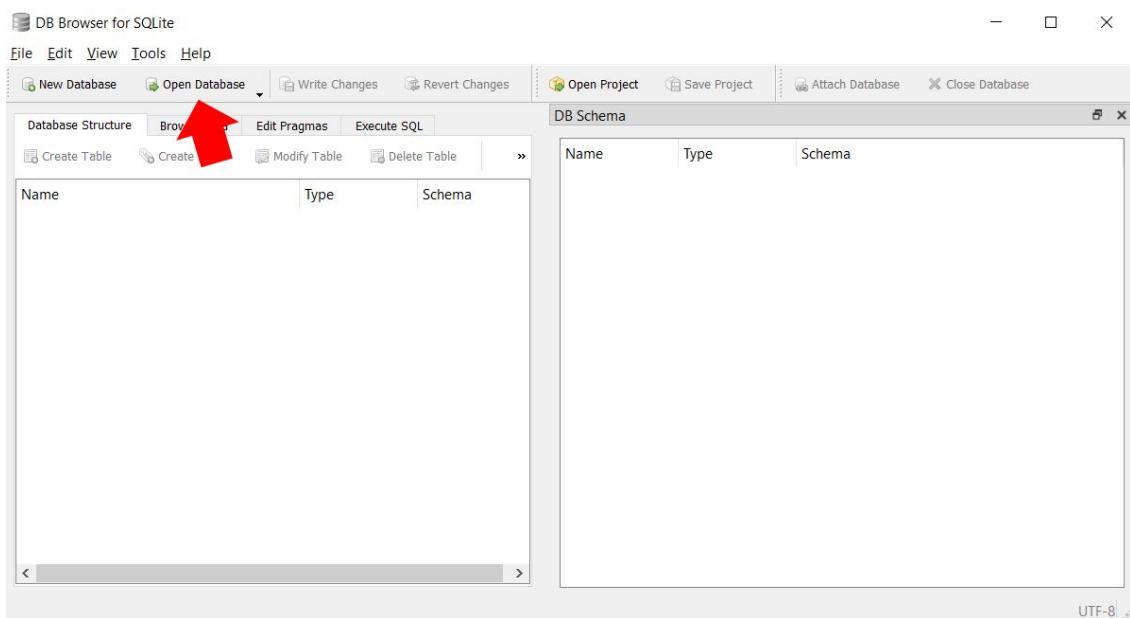
บทต่อไป เราจะเปิดโปรแกรม DB Browser เพื่อสร้างการเชื่อมต่อไปที่ไฟล์ database พร้อมกัน

Chapter 4 - Connect to Database

ปกติเวลาที่เราต้องการจะเข้าไปดึงข้อมูลจาก database เราจะเขียน query ผ่านซอฟต์แวร์ที่เรียกว่า “client” เสร็จแล้ว client จะส่ง query ของเราไปรันบนเซอร์เวอร์ และดึงผลลัพธ์กลับมาแสดงให้เราดู

โปรแกรม DB Browser เปรียบเสมือนตัว client และก่อนจะเขียน SQL queries ได้ เราต้องสร้างการเชื่อมต่อไปที่ database ก่อน (i.e. create connection) สามารถทำง่ายๆ ในสองขั้นตอน

1. เปิดโปรแกรมขึ้นมา และคลิกปุ่ม Open Database



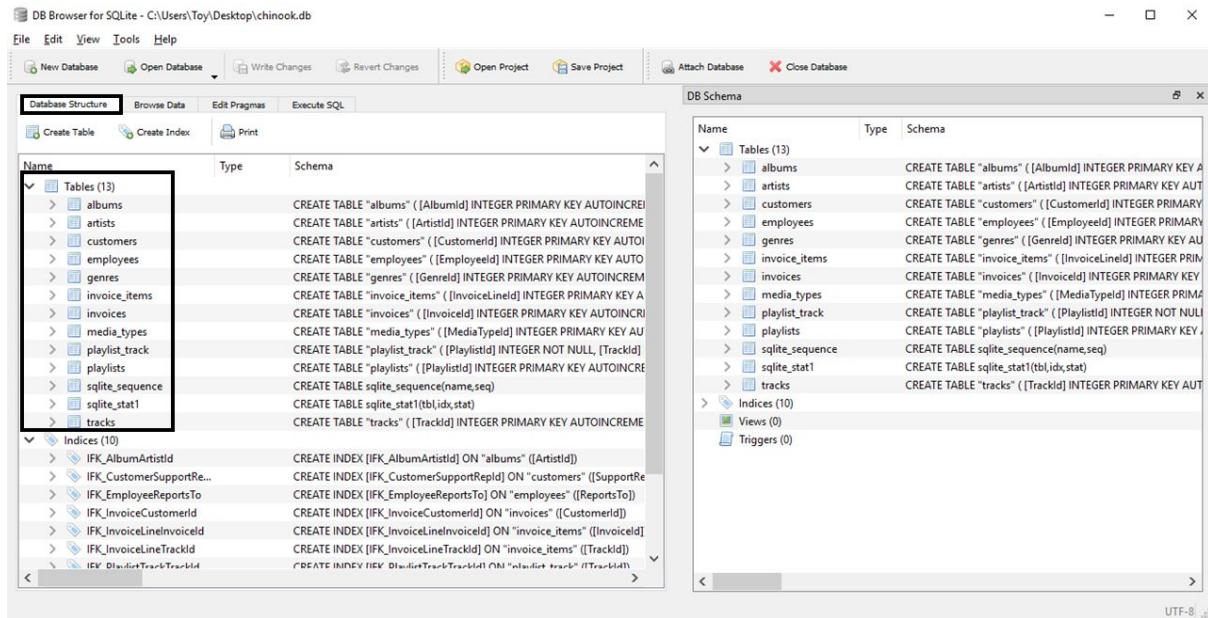
คลิก Open Database เพื่อ connect database

2. เลือกไฟล์ chinook.db และคลิก Open เท่านี้ก็เสร็จเรียบร้อย



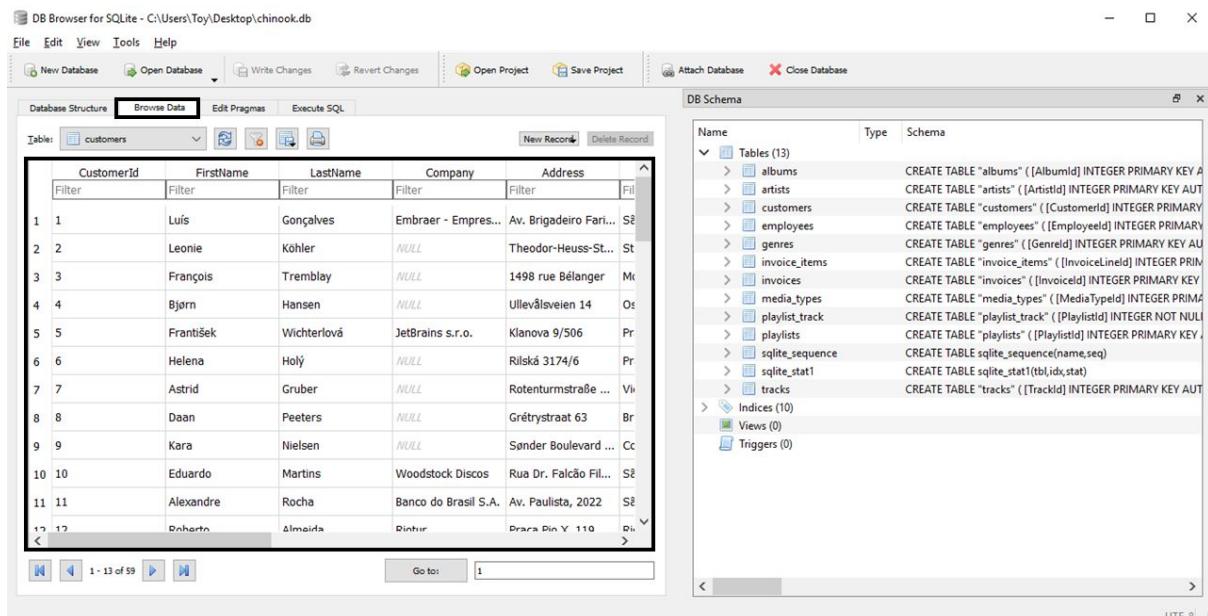
SQLite database จะมีนามสกุล .db ในบทต่อๆไปของหนังสือเล่มนี้ แออดจะสอนวิธีการสร้าง SQLite database แบบเริ่มจากศูนย์เลย Stay Tuned!

เมื่อเชื่อมต่อกับ database และ DB Browser จะแสดงรายชื่อตารางทั้งหมดในแท็บ **Database Structure** เราสามารถคลิกที่ไอคอน > เพื่อดูชื่อคอลัมน์ของแต่ละตารางได้



หน้าต่าง Database Structure

คลิกที่แท็บ Browse Data เพื่อพรีวิวข้อมูลในแต่ละตาราง ฟีเจอร์นี้สะดวกมากเพราะช่วยให้เราเห็นหน้าตาของข้อมูลเบื้องต้นในแต่ละตาราง



หน้าต่าง Browse Data

✓ เวลาทำงานจริง การ connect database ของบริษัทต้องใช้ **username** และ **password** ที่ database admin สร้างให้เราด้วย (เรารายกว่าการ grant สิทธิ์เข้าถึง database)

ถ้ามาคือแท็บที่เราจะทำงานด้วยເຍວະທີ່ສຸດໃນ DB Browser/ Client คือแท็บ Execute SQL ໄວ້ເຊີ້ນແລະຮັນ SQL queries ເພື່ອດຶງຂໍ້ມູນຈາກ database

The screenshot shows the DB Browser for SQLite interface. In the top menu bar, 'Execute SQL' is selected. In the main window, the SQL tab contains the query: 'SELECT * FROM customers LIMIT 10;'. The results pane displays 10 rows from the 'customers' table:

CustomerID	FirstName	LastName	Company	Address	City	State	Country	PostalCode	Phone
1	Luis	Gongalves	Embraer - Empresa Brasileira de Aeronáutica S.A.	Av. Brigadeiro Faria Lima, 2170	São José dos Campos	SP	Brazil	12227-000	+55 (12) 3923-5555
2	Leone	Köhler	NULL	Theodor-Heuss-Straße 34	Stuttgart	NULL	Germany	70174	+49 0711 2842222
3	François	Tremblay	NULL	1498 rue Bélganger	Montréal	QC	Canada	H2G 1A7	+1 (514) 721-4711
4	Bjørn	Hansen	NULL	Ullevålsveien 14	Oslo	NULL	Norway	0171	+47 22 44 22 22
5	František	Wichterlová	JetBrains s.r.o.	Klanova 9/506	Prague	NULL	Czech Republic	14700	+420 2 4172 5555
6	Helena	Högl	NULL	Rådhusgatan 3174/6	Prague	NULL	Czech Republic	14300	+420 2 4177 0449
7	Astrid	Gruber	NULL	Rotenturmstraße 4, 1010 Innere Stadt	Vienne	NULL	Austria	1010	+43 0 5134505
8	Daan	Peeters	NULL	Grétrystraat 63	Brussels	NULL	Belgium	1000	+32 02 219 03 03
9	Kara	Nielsen	NULL	Sender Boulevard 51	Copenhagen	NULL	Denmark	1720	+45 3331 9991
10	Eduardo	Martins	Woodstock Discos	Rua Dr. Felício Filho, 155	São Paulo	SP	Brazil	01007-010	+55 (11) 3033-5446

Below the results, it says 'Result: 10 rows returned in 25ms' and 'At time 1:'. The bottom right corner of the results pane has a small icon.

หน้าต่าง Execute SQL

ลอง copy query นี้ไปใส่ในแท็บ Execute SQL และกดปุ่ม **CTRL+Enter** สำหรับ Windows หรือ **Command+Enter** สำหรับ mac เพื่อ submit query (i.e. ภาษาบ้านๆเรียกว่าการรันคิวอาร์)

SELECT * FROM customers;

ถ้าเราเขียน query ถูกต้อง ข้อมูลจะถูกดึงขึ้นมาแสดงในหน้าต่าง trgk กลาง เราเรียกข้อมูลที่ได้กลับมาว่า “**Result Set**” ยินดีด้วย! เมื่อก็กุญแจได้ลองเขียน query และขอของตัวเองแล้ว

The screenshot shows the DB Browser for SQLite interface. In the top menu bar, 'Execute SQL' is selected. In the main window, the SQL tab contains the query: 'SELECT * FROM customers LIMIT 10;'. The results pane displays 10 rows from the 'customers' table:

CustomerID	FirstName	LastName	Company	Address	City	State	Country	PostalCode	Phone
1	Luis	Gongalves	Embraer - Empresa Brasileira de Aeronáutica S.A.	Av. Brigadeiro Faria Lima, 2170	São José dos Campos	SP	Brazil	12227-000	+55 (12) 3923-5555
2	Leone	Köhler	NULL	Theodor-Heuss-Straße 34	Stuttgart	NULL	Germany	70174	+49 0711 2842222
3	François	Tremblay	NULL	1498 rue Bélganger	Montréal	QC	Canada	H2G 1A7	+1 (514) 721-4711
4	Bjørn	Hansen	NULL	Ullevålsveien 14	Oslo	NULL	Norway	0171	+47 22 44 22 22
5	František	Wichterlová	JetBrains s.r.o.	Klanova 9/506	Prague	NULL	Czech Republic	14700	+420 2 4172 5555
6	Helena	Högl	NULL	Rådhusgatan 3174/6	Prague	NULL	Czech Republic	14300	+420 2 4177 0449
7	Astrid	Gruber	NULL	Rotenturmstraße 4, 1010 Innere Stadt	Vienne	NULL	Austria	1010	+43 0 5134505
8	Daan	Peeters	NULL	Grétrystraat 63	Brussels	NULL	Belgium	1000	+32 02 219 03 03
9	Kara	Nielsen	NULL	Sender Boulevard 51	Copenhagen	NULL	Denmark	1720	+45 3331 9991
10	Eduardo	Martins	Woodstock Discos	Rua Dr. Felício Filho, 155	São Paulo	SP	Brazil	01007-010	+55 (11) 3033-5446

Below the results, it says 'Result: 10 rows returned in 25ms' and 'At time 1:'. The bottom right corner of the results pane has a small icon.

ผลลัพธ์จะแสดงในหน้าต่างกลาง เราเรียกข้อมูลที่ถูกดึงขึ้นมาว่า Result Set

Chapter 5 - Five Minutes Quick Start Guide

ເອົາລ່ວ! ບໜີ້ແວດຈະອືບຍາກເຊື່ອ SQL queries ແບບເຮົວໆ 5 ນາທີຈົບ ເຊື່ອນເປັນເລີຍ ຕອນນີ້ ນັກເຮືອນທຸກຄົນຕ້ອງ connect DB Browser ກັບ chinook database ເຮືອນຮ້ອຍແລ້ວນະ

5.1 สิ่งที่ต้องรู้เกี่ยวกับการเขียน SQL

SQL เป็นภาษาแบบ case-insensitive แปลว่าเวลาเขียน query เราสามารถเขียนได้ทั้งตัวพิมพ์เล็กหรือตัวพิมพ์ใหญ่ ลองดูตัวอย่างสอง queries ด้านล่าง สังเกตว่าแต่ละ query จะปิดท้ายด้วย ;

```
-- lower case  
select firstname, lastname from customers;  
  
-- upper case  
SELECT FIRSTNAME, LASTNAME FROM CUSTOMERS;
```



 ในทางปฏิบัติ เรานิยมเขียน SQL clause เป็นตัวพิมพ์ใหญ่ (UPPERCASE) ส่วนชื่อคอลัมน์ และตารางเป็นตัวพิมพ์เล็ก (lowercase)

```
-- good practice  
SELECT firstname, lastname FROM customers;
```

SQL query ไม่สนใจ white space แปลว่าสองคิวอาร์ด้านล่างได้ผลเหมือนกัน เวลาเขียนเราว่าจัดวาง format การเขียนคิวอาร์ดของเรามาให้เหมาะสม อ่านง่าย ทั้งสำหรับตัวเองและเพื่อนร่วมงาน

```
-- a lot of white space
SELECT firstname, lastname FROM customers;

-- single white space
SELECT firstname, lastname FROM customers;
```

Query นี้อ่านว่า “จะเลือกคอลัมน์ firstname และ lastname ของลูกค้าจากตาราง customers”

	FirstName	LastName
1	Luís	Gonçalves
2	Leonie	Köhler
3	François	Tremblay
4	Bjørn	Hansen
5	František	Wichterlová
6	Helena	Holý
7	Astrid	Gruber
8	Daan	Peeters
9	Kara	Nielsen
10	Eduardo	Martins

Result set ที่ได้จากการ query ด้านบน

5.2 SQL clauses ที่เราใช้บ่อยที่สุด

SELECT และ WHERE คือ clauses ที่เราใช้บ่อยที่สุดในการเขียน queries โดย SELECT คือการดึงคอลัมน์ที่ต้องการ ส่วน WHERE คือการเขียนเงื่อนไขเพื่อฟิลเตอร์ข้อมูล i.e. records/ rows

Query ด้านล่างดึงคอลัมน์ชื่อจริง นามสกุลของลูกค้า และประเทศ (3 คอลัมน์) จากตาราง customers โดยฟิลเตอร์เฉพาะลูกค้าที่อยู่ในประเทศอเมริกาเท่านั้น สังเกตว่าคอลัมน์ใน SELECT clause จะถูกแยกด้วย comma ,

```
SELECT firstname, lastname, country  
FROM customers  
WHERE country = 'USA';
```

✓ เวลาเขียนชื่อคอลัมน์ใน SELECT clause เรา尼ยมเขียนชื่อคอลัมน์แต่ละชื่อบนบรรทัดใหม่ เพื่อให้ง่ายต่อการอ่าน query อย่างคิวอาร์ด้านล่างเห็นแล้วกรุ๊หันที่ว่าดึงข้อมูลอ กมา 3 คอลัมน์

```
SELECT  
firstname,  
lastname,  
country  
FROM customers  
WHERE country = 'USA';
```

เราสามารถกำหนดจำนวนแถวที่ต้องการใน result set ด้วยการใส่ LIMIT clause ต่อท้าย query ของเรา เช่น LIMIT 10 คือการดึงข้อมูลอ กมาแค่ 10 แถวแรกสุดเท่านั้น

```

SELECT
    firstname,
    lastname,
    country
FROM customers
WHERE country = 'USA'
LIMIT 10;

```

Query นี้อ่านว่า “จงเลือกคอลัมน์ firstname, lastname และ country จากตาราง customers ฟิลเตอร์เฉพาะลูกค้าที่อยู่ในประเทศอเมริกา และดึงออกมาแค่ 10 คนแรกเท่านั้น”

	FirstName	LastName	Country
1	Frank	Harris	USA
2	Jack	Smith	USA
3	Michelle	Brooks	USA
4	Tim	Goyer	USA
5	Dan	Miller	USA
6	Kathy	Chase	USA
7	Heather	Leacock	USA
8	John	Gordon	USA
9	Frank	Ralston	USA
10	Victor	Stevens	USA

ดึงรายชื่อลูกค้าจากประเทศอเมริกา 10 คนแรก (LIMIT 10)

5.3 การใส่คอมเม้นต์ใน SQL queries

เวลาเขียน SQL queries แอดแนวนำให้ใส่ comment เพื่ออธิบายโค้ดของเราด้วย (i.e. explainability) โดย comment จะเขียนได้สองแบบ: single-line และ multiple-line

Single-line comment เราใช้ syntax -- ข้อความ

```

-- select 10 customers from USA
SELECT
    firstname,
    lastname,
    country
FROM customers
WHERE country = 'USA'
LIMIT 10;

```

ข้อดีของ single-line comment คือเราสามารถ comment โค้ดบางบรรทัดของเราได้ เพื่อทดลองรัน query เร็วๆ ตัวอย่างด้านล่างสามารถรันได้ปกติ ดึงคอลัมน์ firstname, country จากตาราง customers ตั้งเฉพาะ 10 แถวแรกสุด

```
-- comment out some lines
SELECT
    firstname,
    -- lastname,
    country
FROM customers
-- WHERE country = 'USA'
LIMIT 10;
```

	FirstName	Country
1	Luís	Brazil
2	Leonie	Germany
3	François	Canada
4	Bjørn	Norway
5	František	Czech Republic
6	Helena	Czech Republic
7	Astrid	Austria
8	Daan	Belgium
9	Kara	Denmark
10	Eduardo	Brazil

Result set จาก query ด้านบน

Multiple-line comment เราใช้ syntax /* ข้อความ */

```
/*
SELECT
    firstname,
    lastname,
    country
FROM customers
WHERE country = 'USA'
LIMIT 10;
*/
```

ตื่นเต้นมั้ย? ตื่นเต้นกดแล้ว! เพราะบทต่อไป ทุกคนจะได้เรียนวิธีการเขียน SQL query ที่ใช้บ่อยที่สุดสำหรับงาน data analyst/ data science นั่นคือ SELECT clause แบบเจาะลึกทุก details

Chapter 6 - SELECT

การเขียน SQL query สำหรับดึงและวิเคราะห์ข้อมูลจาก database จะขึ้นต้นด้วย SELECT clause เมื่อเราใช้ SELECT เพื่อเลือกคอลัมน์ที่เราต้องการจากตาราง (table) ที่เราต้องการ บทนี้แสดงจะอธิบายการดึงข้อมูลจากหนึ่ง table ก่อน เมื่อเรียนไปถึงบท JOIN นักเรียนจะสามารถดึงข้อมูลจากหลายๆ tables ได้พร้อมกัน How Cool!

6.1 ดึงข้อมูลทุกคอลัมน์

Query ที่เขียนบ่อยที่สุดคือ SELECT * FROM table_name; โดย * แปลว่า “all columns” query ด้านล่างเราดึงทุกคอลัมน์ของตาราง customers

```
SELECT * FROM customers;
```

เขียน query ในแท็บ Execute SQL เสร็จแล้วกดปุ่ม CTRL+Enter หรือปุ่ม F5 บนคีย์บอร์ดเพื่อรันคิวรี่ (สำหรับ mac กด Command+Enter) Result Set จะแสดงในหน้าต่างตรงกลาง

Result: 59 rows returned in 28ms At line 1 of 1 SELECT * FROM customers;										
	CustomerID	FirstName	LastName	Company	Address	City	State	Country	PostalCode	Phone
1	1	Luis	Gonçalves	Embraer - Empresa Brasileira de Aer...	Av. Brigadeiro Faria Lima, 2170	São José dos Campos	SP	Brazil	12227-000	+55 (12) 3923-5555
2	2	Leоні	Кöhler	NULL	Theodor-Heuss-Straße 34	Stuttgart	NULL	Germany	70174	+49 0711 2842222
3	3	François	Tremblay	NULL	1498 rue Bélanger	Montréal	QC	Canada	H2G 1A7	+1 (514) 721-4711
4	4	Bjørn	Hansen	NULL	Ullevålsveien 14	Oslo	NULL	Norway	0171	+47 22 44 22 22
5	5	František	Wichterlová	JetBrains s.r.o.	Klanova 9/506	Prague	NULL	Czech Republic	14700	+420 2 4172 5555
6	6	Helena	Holý	NULL	Rilská 3174/6	Prague	NULL	Czech Republic	14300	+420 2 4177 0449
7	7	Astrid	Gruber	NULL	Rofenturmstraße 4, 1010 Innere Stadt	Vienne	NULL	Austria	1010	+43 01 5134505
8	8	Daan	Peeters	NULL	Grétrystraat 63	Brussels	NULL	Belgium	1000	+32 02 219 03 03
9	9	Kara	Nielsen	NULL	Sønder Boulevard 51	Copenhagen	NULL	Denmark	1720	+45 3 3331 9991
10	10	Eduardo	Martins	Woodstock Discos	Rua Dr. Falcão Filho, 155	São Paulo	SP	Brazil	01007-010	+55 (11) 3033-5446
11	11	Alexandre	Rocha	Banco do Brasil S.A.	Av. Paulista, 2022	São Paulo	SP	Brazil	01310-200	+55 (11) 3055-3278
12	12	Roberto	Almeida	Riotur	Praça Pio X, 119	Rio de Janeiro	RJ	Brazil	20040-020	+55 (21) 2271-7000
13	13	Fernanda	Ramos	NULL	Qe 7 Bloco G	Brasília	DF	Brazil	71020-677	+55 (61) 3363-5547

ถ้า query รันสำเร็จ ผลลัพธ์จะแสดงในหน้าต่างตรงกลาง

สังเกตด้านล่างสุดคือ Log ของ query ที่เราเพิ่งเขียนไป SQLite รายงานกลับมาว่า “Result: 59 rows returned in 28ms” ผลลัพธ์ที่ดึงออกมาก้าว 59 แค่ ใช้เวลา 28 milliseconds

✓ ทิปสำหรับการเรียน SQL แออดแนะนำให้ฝึกเขียน query ด้วยตัวเอง พยายามไม่ copy & paste โค้ดลงไปใน DB Browser ฝึกเขียนบ่อยๆจะได้เป็นเร็วๆ

6.2 ดึงข้อมูลเฉพาะคอลัมน์ที่เราต้องการ

เราสามารถเขียนระบบชื่อคอลัมน์ที่เราต้องการได้โดยใน SELECT clause โดยชื่อคอลัมน์แต่ละชื่อจะถูก split ด้วย comma , และถูกเขียนบนบรรทัดใหม่ หลังจากขึ้นบรรทัดใหม่ให้เคาะ space bar สองครั้ง (ย่อหน้า) การเขียนคิวรีแบบนี้คือมาตรฐานที่ช่วยให้โค้ดของเราอ่านง่ายขึ้น

```
SELECT
    firstname,
    lastname,
    country,
    email
FROM customers;
```

✓ อย่าลืมตรวจสอบ query ก่อนรัน สาเหตุที่คิวรีรันไม่ออกส่วนใหญ่เกิดจาก 1. เขียนชื่อคอลัมน์หรือชื่อตารางผิด และ 2. ลืมใส่ comma หรือใส่เกินมาใน SELECT clause

6.3 กำหนดจำนวนแถวของ Result Set

เราใช้ LIMIT เพื่อกำหนดจำนวนแถวของ result set

```
SELECT
    firstname,
    lastname,
    country,
    email
FROM customers
LIMIT 10;
```

✓ เวลาทำงานจริง ข้อมูลใน database อาจจะมีถึงร้อยล้าน records (หรือพันล้านเลยก็ได้) ถ้าเราแค่ต้องการพรีวิวข้อมูลดูคร่าวๆ ควรใส่ LIMIT ต่อท้ายคิวรีของเรามเสมอ

6.4 คุณค่า Distinct Values ของคอลัมน์

เราใช้ SELECT DISTINCT เพื่อดู unique (หรือ distinct) values ในคอลัมน์ที่เราต้องการ ตัวอย่างด้านล่าง เราดึงรายชื่อประเทศที่ไม่ซ้ำกันออกจากคอลัมน์ country ในตาราง customers

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

Query นี้ตอบคำถามว่า “ลูกค้าของเรามาจากประเทศอะไรบ้าง?” คำตอบคือ 24 ประเทศ!

6.5 Merge คอลัมน์แบบ Text เข้าด้วยกัน

ถ้าเราต้องการรวมคอลัมน์ text สองคอลัมน์ เช่น firstname และ lastname รวมกันเป็น full name เราสามารถใช้เครื่องหมาย pipe หรือ || (อยู่หนีอปุ่ม Enter บนคีย์บอร์ด)

Query นี้เรา merge คอลัมน์ firstname และ lastname โดยเราเพิ่มช่องว่างหรือ white space ขั้นกลางระหว่างชื่อกับนามสกุล ใน SQL เราใช้ single quote ' ' เพื่อกำหนด white space ตามตัวอย่างด้านล่าง

```
SELECT
    firstname,
    lastname,
    firstname || ' ' || lastname
FROM customers;
```

	FirstName	LastName	firstname ' ' lastname
1	Luís	Gonçalves	Luís Gonçalves
2	Leonie	Köhler	Leonie Köhler
3	François	Tremblay	François Tremblay
4	Bjørn	Hansen	Bjørn Hansen
5	František	Wichterlová	František Wichterlová
6	Helena	Holý	Helena Holý
7	Astrid	Gruber	Astrid Gruber
8	Daan	Peeters	Daan Peeters
9	Kara	Nielsen	Kara Nielsen
10	Eduardo	Martins	Eduardo Martins

merge columns ด้วยสัญลักษณ์ pipe || ให้ได้ full name



สามารถใช้ pipe เพื่อ merge มากกว่าสองคอลัมน์ก็ได้ แค่เขียน || ต่อไปเรื่อยๆ

6.6 เปลี่ยนชื่อคอลัมน์ด้วย AS

สังเกตผลลัพธ์ของ query ที่เรารันเมื่อตั้งค่า คอลัมน์ที่สามชื่อว่า “firstname || ' ' || lastname” เราสามารถเปลี่ยนชื่อคอลัมน์ได้ง่ายๆด้วย AS (ย่อมาจาก Alias) แค่เขียน AS ต่อท้ายคอลัมน์ที่เราต้องการเปลี่ยนชื่อได้เลย query นี้ เราตั้งชื่อคอลัมน์ที่สามว่า “fullname”

```
SELECT
    firstname,
    lastname,
    firstname || ' ' || lastname AS fullname
FROM customers;
```

	FirstName	LastName	fullname
1	Luís	Gonçalves	Luís Gonçalves
2	Leonie	Köhler	Leonie Köhler
3	François	Tremblay	François Tremblay
4	Bjørn	Hansen	Bjørn Hansen
5	František	Wichterlová	František Wichterlová
6	Helena	Holý	Helena Holý
7	Astrid	Gruber	Astrid Gruber
8	Daan	Peeters	Daan Peeters
9	Kara	Nielsen	Kara Nielsen
10	Eduardo	Martins	Eduardo Martins

ตั้งชื่อคอลัมน์ใหม่ด้วย AS (Alias)

6.7 บวก ลบ คูณ หาร ง่ายๆด้วย SELECT

เราสามารถใช้ SQL เป็นเครื่องคิดเลขง่ายๆ ลองพิมพ์ statement ด้านล่างใน DB Browser และดูผลลัพธ์ ถ้าการคำนวณซับซ้อนขึ้น สามารถใช้ () ได้เช่นกัน

```
SELECT 8 + 2;
SELECT 8 - 2;
SELECT 8 * 2;
SELECT 8 / 2;
```

6.8 Transform คอลัมน์ด้วยสูตรคณิตศาสตร์

ตาราง Tracks มีคอลัมน์ที่ชื่อว่า bytes และ milliseconds ที่เก็บขนาดของไฟล์เพลงและความยาวของเพลงตามลำดับ เราสามารถเขียน SQL เพื่อเปลี่ยน (transform) bytes เป็น megabytes และ milliseconds เป็น minutes ลองดูตัวอย่างด้านล่าง

เราเปลี่ยน bytes เป็น megabytes ด้วยสูตร: bytes/ 1048576.0 และเปลี่ยน milliseconds เป็น minutes ด้วยสูตร: milliseconds/ 60000.0 ตามลำดับ

```
SELECT
    name,
    bytes/ (1024.0 * 1024.0) AS mb,
    milliseconds/ 60000.0 AS minutes
FROM tracks;
```

สังเกตว่าเราใช้ตัวเลขแบบ numeric ในคิวรีด้านบน เช่น milliseconds/ **60000.0** เพื่อให้ผลลัพธ์ (คอลัมน์ใหม่) ที่ได้ออกมาเป็นแบบ numeric คือมีจุดทศนิยม เช่น 3.5432 นาที เป็นต้น

	Name	mb	minutes
1	For Those About To Rock (We Salute...)	10.6528606414795	5.72865
2	Balls to the Wall	5.25514984130859	5.70936666666667
3	Fast As a Shark	3.80610847473145	3.84365
4	Restless and Wild	4.13110637664795	4.20085
5	Princess of the Dawn	5.99910831451416	6.25696666666667
6	Put The Finger On You	6.40244579315186	3.4277
7	Let's Get It Up	7.28279209136963	3.89876666666667
8	Inject The Venom	6.53539657592773	3.5139
9	Snowballed	6.293701171875	3.38503333333333
10	Evil Walks	8.21232318878174	4.39161666666667

transform สร้างสองคอลัมน์ใหม่คือ mb และ minutes ตามลำดับ

6.9 สรุปการใช้งานที่สำคัญของ SELECT

- เราใช้ SELECT เพื่อดึงคอลัมน์ที่ต้องการจาก table/ database
- เราใช้ SELECT DISTINCT เพื่อดึง unique values (ไม่ซ้ำกันเลย) ออกมายจากคอลัมน์ที่เราต้องการ เช่น ชื่อประเทศ ชื่อจังหวัด ชื่อตำบล เป็นต้น
- เราใช้ SELECT เป็นเครื่องคิดเลขได้
- เราใช้ SELECT เพื่อสร้างคอลัมน์ใหม่ได้
- เราใช้ || (อ่านว่า pipe) เพื่อ merge หลายคอลัมน์แบบ text เข้าด้วยกัน
- เราใช้ AS (ย่อมาจาก Alias) เพื่อเปลี่ยนชื่อคอลัมน์
- เราใช้ LIMIT ตามด้วยตัวเลขจำนวนเต็ม (integer) เพื่อกำหนดจำนวนแถวที่ต้องการดึง ออกมาใน Result Set
- อย่าลืมตรวจสอบ spelling ของชื่อคอลัมน์ และชื่อตารางก่อนรันคิวรี ตรวจสอบ comma , ด้วยว่าเขียนขาดหรือเกินหรือเปล่า เพื่อลดการเกิด error

Chapter 7 - WHERE

บทนี้เราจะเรียน SQL clause ที่ใช้เยอะที่สุดเป็นอันดับสองรองจาก SELECT นั่นคือ WHERE clause นั่นเอง เราใช้ WHERE เพื่อฟิลเตอร์ หรือภาษาไทยคือการกรองข้อมูล (rows/ records) ที่เราต้องการด้วยเงื่อนไขแบบต่างๆ บทนี้นำจะยกตัวอย่างที่สุดในหนังสือเล่มนี้เลย แอดจะอธิบายการใช้ WHERE ครบทุกแบบ และเทคนิคขั้นสูงสำหรับ pattern ด้วย Regular Expressions



บทนี้เราจะสอนการเขียน WHERE ครบทั้ง 8 แบบ มีอะไรบ้าง มาลองดูกันเลย

7.1 การฟิลเตอร์พื้นฐานด้วย logical operators

เราจะเขียน WHERE clause ต่อท้าย SELECT clause เช่นเดียวกับ query ด้านล่างฟิลเตอร์ลูกค้าที่อยู่ในประเทศอเมริกาเท่านั้น



ถ้าคอลัมน์ที่เราใช้ใน WHERE clause เป็น text เช่น คอลัมน์ country เราต้องใส่ชื่อประเทศในเครื่องหมาย single quote แบบนี้ 'USA'

```
SELECT * FROM customers  
WHERE country = 'USA';
```

logical operators ใน SQL จะเหมือนกับที่เราใช้ใน Excel เลย แอดสรุปไว้ให้ในตารางด้านล่าง

Logical Operators	ความหมาย (Eng)	ความหมาย (Thai)
=	equal to	เท่ากับ
>	greater than	มากกว่า
<	less than	น้อยกว่า
>=	greater than or equal to	มากกว่าหรือเท่ากับ
<=	less than or equal to	น้อยกว่าหรือเท่ากับ
<>	not equal to	ไม่เท่ากับ

Query ถัดไป เราดึงข้อมูลเฉพาะลูกค้าที่ ไม่ได้อาศัยอยู่ ในประเทศอเมริกา

```
SELECT * FROM customers  
WHERE country <> 'USA';
```

7.2 การเขียนมากกว่าหนึ่งเงื่อนไขด้วย AND/ OR

เราสามารถเขียนเงื่อนไขใน WHERE clause มากกว่าหนึ่งเงื่อนไข ด้วยการเขียน AND หรือ OR

- AND ทุกเงื่อนไขต้องเป็นจริงทั้งหมด (all TRUE) SQL ถึงจะดึงข้อมูลกลับมาให้เรา
- OR ถ้าเงื่อนไขใดเงื่อนไขหนึ่งเป็นจริง SQL ก็จะดึงข้อมูลกลับมาให้เราทันที

Query นี้ดึงข้อมูลเฉพาะลูกค้าที่อาศัยอยู่ในประเทศ USA และอยู่ในรัฐ CA สองเงื่อนไขต้องเป็นจริงทั้งคู่ เพราะเราเข้ม conditions ด้วย AND

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

	CustomerId	FirstName	LastName	Company	Address	City	State	Country
1	16	Frank	Harris	Google Inc.	1600 Amphitheatre Parkway	Mountain View	CA	USA
2	19	Tim	Goyer	Apple Inc.	1 Infinite Loop	Cupertino	CA	USA
3	20	Dan	Miller	NULL	541 Del Medio Avenue	Mountain View	CA	USA

ลูกค้าต้องอยู่ในประเทศอเมริกาและรัฐ CA เท่านั้น (สองเงื่อนไขเป็นจริงทั้งคู่)

Query ถัดไป เราเข้มเงื่อนไขทั้งหมดด้วย OR เพื่อดึงลูกค้าที่อาศัยอยู่ในหนึ่งในสามประเทศนี้ USA, France หรือ Belgium

```
SELECT * FROM customers  
WHERE country = 'USA' OR country = 'France' OR country = 'Belgium';
```

✓ เราสามารถใช้งานล็อก () เพื่อให้เงื่อนไขของเรามีความซัดเจนมากขึ้น เวลา Query เราเมื่อความซับซ้อน ตัวอย่างด้านล่าง เราดึงข้อมูลเฉพาะลูกค้าที่ (อาศัยอยู่ในประเทศ USA หรือ France) และ (ใช้อีเมล์ fharris@google.com หรือ marc.dubois@hotmail.com)

```
SELECT  
    firstname,  
    lastname,  
    email,  
    country  
FROM customers  
WHERE (country = 'USA' OR country = 'France')  
    AND (email = 'fharris@google.com' OR email =  
        'marc.dubois@hotmail.com');
```

	FirstName	LastName	Email	Country
1	Frank	Harris	farris@google.com	USA
2	Marc	Dubois	marc.dubois@hotmail.com	France

มีลูกค้าแค่สองคนที่ตรงกับเงื่อนไขที่เราเขียน

7.3 การเขียนเงื่อนไขด้วย IN operator

เราใช้ IN operator แทนที่การเขียน OR ซ้ำหลายครั้งได้เลย query ด้านล่างเราดึงข้อมูลเฉพาะลูกค้าที่อาศัยอยู่ใน (IN) ประเทศ USA, France, Belgium

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

✓ IN operator ใน query นี้เทียบเท่ากับการเขียน SELECT * FROM customers WHERE country = 'USA' OR country = 'FRANCE' OR country = 'Belgium';

7.4 การเขียนเงื่อนไขแบบช่วงด้วย BETWEEN AND

BETWEEN .. AND .. แปลว่า “ระหว่าง” สอง queries ด้านล่าง เราเปลี่ยนไปดึงข้อมูลจาก invoices table กันบ้าง query แรกเราดึงข้อมูล total invoices ระหว่าง 10 และ 20 (แบบป้ายปิด inclusive) ในทางคณิตศาสตร์เราเขียน inclusive ได้แบบนี้ [10, 20]

Inclusive คือ SQL จะดึงข้อมูลในคอลัมน์ total ที่มีค่าตั้งแต่ 10 ถึง 20 เลย

```
SELECT invoicedate, total FROM invoices
WHERE total BETWEEN 10 AND 20;
```

💡 แต่ถ้าเราใช้ BETWEEN AND กับคอลัมน์แบบ datetime หรือ text จะเป็นการดึงแบบป้ายเปิด Exclusive ทางด้านขวา ในทางคณิตศาสตร์เราเขียน exclusive ได้แบบนี้ ['2010-01-01', '2010-03-31')

Query ด้านล่างเราจะ filเตอร์ข้อมูลเฉพาะ invoicedate ตั้งแต่วันที่ 1 มกราคม 2010 จนถึง 30 มีนาคม 2010 (ในกรณีนี้ป้ายเปิดจะไม่รวมวันที่ 31 มีนาคม 2010)

```
SELECT invoicedate, total FROM invoices
WHERE invoicedate BETWEEN '2010-01-01' AND '2010-03-31';
```

Query สดท้ายของหัวข้อ BETWEEN AND แสดงเขียนตัวอย่างการฟิลเตอร์ข้อมูลในคอลัมน์แบบ text เช่น billingcountry โดย query นี้จะฟิลเตอร์ชื่อประเทศที่ขึ้นต้นด้วยอักษร A ไปจนถึง B (exclusive แบบปลายเปิดจะไม่รวม C)

```
SELECT invoicedate, billingcountry FROM invoices
WHERE billingcountry BETWEEN 'A' AND 'C';
```

	InvoiceDate	BillingCountry
1	2009-01-03 00:00:00	Belgium
2	2009-04-04 00:00:00	Australia
3	2009-04-09 00:00:00	Brazil
4	2009-05-23 00:00:00	Brazil
5	2009-06-05 00:00:00	Brazil
6	2009-07-07 00:00:00	Australia
7	2009-08-24 00:00:00	Belgium
8	2009-09-06 00:00:00	Brazil
9	2009-09-07 00:00:00	Brazil
10	2009-10-09 00:00:00	Australia

Result set จะดึงข้อมูลเฉพาะประเทศที่ชื่อขึ้นต้นด้วยตัว A, B

สรุปคือ BETWEEN AND จะ inclusive เฉพาะคอลัมน์ที่เป็นตัวเลขเท่านั้น ส่วนคอลัมน์แบบ datetime และ text จะเป็นการฟิลเตอร์แบบ exclusive ปลายเปิดทางด้านขวา

7.5 การเขียนเงื่อนไขเพื่อหาค่า NULL (missing value)

NULL ใน SQL database เปรียบเสมือนค่า missing value i.e. ข้อมูลไม่สมบูรณ์ วิธีการฟิลเตอร์ค่า NULL เราจะใช้ column_name IS NULL ตามตัวอย่างในคิวอาร์ด้านล่าง

```
SELECT firstname, lastname, company
FROM customers
WHERE company IS NULL;
```

	FirstName	LastName	Company
1	Leонie	Köhler	NULL
2	François	Tremblay	NULL
3	Bjørn	Hansen	NULL
4	Helena	Holý	NULL
5	Astrid	Gruber	NULL
6	Daan	Peeters	NULL
7	Kara	Nielsen	NULL
8	Fernanda	Ramos	NULL
9	Michelle	Brooks	NULL
10	Dan	Miller	NULL

ฟิลเตอร์เฉพาะ record ที่ company เป็นค่า NULL

7.6 การเขียนเงื่อนไขด้วย NOT

เราสามารถใช้ NOT operator เพื่อกลับค่า TRUE/ FALSE ของเงื่อนไขใน WHERE clause ได้ ง่ายๆ ตัวอย่างเช่น การเปลี่ยนเงื่อนไขจาก IS NULL เป็น IS NOT NULL

Query ด้านล่างอ่านว่า “เลือกข้อมูลทุก colum ของ customers table ฟิลเตอร์เฉพาะบริษัทที่ไม่มีค่า NULL” (หรืออ่านแบบบ้านๆว่า มีข้อมูลครบถ้วน) อย่าลืมว่าการอ่าน SQL query สามารถ อ่านเป็นภาษาอังกฤษตรงๆและแปลเป็นภาษาไทยได้เลย so simple!

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

	FirstName	LastName	Company
1	Luís	Gonçalves	Embraer - Empresa Brasileira de Aer...
2	František	Wichterlová	JetBrains s.r.o.
3	Eduardo	Martins	Woodstock Discos
4	Alexandre	Rocha	Banco do Brasil S.A.
5	Roberto	Almeida	Riotur
6	Mark	Philips	Telus
7	Jennifer	Peterson	Rogers Canada
8	Frank	Harris	Google Inc.
9	Jack	Smith	Microsoft Corporation
10	Tim	Goyer	Apple Inc.

ฟิลเตอร์เฉพาะ record ที่ company ไม่มีค่า NULL (ลองเปรียบเทียบกับ query ก่อนหน้า)

นักเรียนสามารถใช้ NOT ได้กับทุก operators ที่แอดสอนมาในบทนี้ เช่น NOT IN, NOT LIKE, NOT BETWEEN AND รวมถึง NOT = หรือ NOT >= ก็ได้ ลองดูตัวอย่าง queries ด้านล่าง

```
-- not and logical operators  
SELECT * FROM customers  
WHERE NOT customerid >= 5;  
  
-- not and in  
SELECT * FROM customers  
WHERE country NOT IN ('USA', 'France', 'Belgium');  
  
-- not and between  
SELECT * FROM invoices  
WHERE invoicedate NOT BETWEEN '2010-01-01' AND '2010-01-31';
```



เว็บไซต์ที่สอน SQL พื้นฐานได้โดยเฉลยคือ [W3Schools](https://www.w3schools.com) และแนะนำลองเข้าไปศึกษาได้

7.7 การเขียนเงื่อนไขแบบ pattern matching

อีกหนึ่ง operator ที่แออดคิดว่ามีประโยชน์มากๆเวลาเขียนเงื่อนไขใน WHERE clause คือ LIKE สำหรับทำ pattern matching ภาษาไทยอธิบายง่ายๆคือการค้นหา pattern ที่เราต้องการ

เราใช้ LIKE คู่กับ wildcards หรืออักษรที่มีความหมายพิเศษ ตัวอย่างของ wildcards ที่เราใช้บ่อยๆใน SQL มีสองตัวคือ % และ _

- % ใช้ match ตัวอักษรหรือตัวเลข ≥ 1 character ขึ้นไป
- _ ใช้ match ตัวอักษรหรือตัวเลขเพียง 1 character

ลองวิเคราะห์ queries ต่อไปนี้ และอ่านคำอธิบายด้านล่าง

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

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

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

Query แรกฟิลเตอร์เฉพาะลูกค้าที่ชื่อขึ้นต้นด้วยตัว L และตามด้วยตัวอักษรอื่นๆที่ตัวก็ได้ Query ที่สองฟิลเตอร์ลูกค้าที่ใช้ gmail.com เท่านั้น และ query สุดท้ายฟิลเตอร์ลูกค้าที่ชื่อขึ้นต้นด้วยตัว J ตามด้วยตัวอักษร/ตัวเลขหนึ่งตัว ปิดท้ายด้วย hn ตามลำดับ (น่าจะชื่อ John)

ลองดูอีกตัวอย่าง ถ้าเราต้องการฟิลเตอร์ลูกค้าที่ใช้เบอร์โทรศัพท์ที่มีตัวเลข 555 อยู่ในเบอร์โดยไม่สนใจตัวอักษร (555 จะอยู่ข้างหน้า ตรงกลาง หรือข้างหลังก็ได้) สามารถเปลี่ยน query ได้แบบนี้

```
SELECT firstname, lastname, phone
FROM customers
WHERE phone LIKE '%555%';
```

	FirstName	LastName	Phone
1	Luís	Gonçalves	+55 (12) 3923-5555
2	František	Wichterlová	+420 2 4172 5555

ฟิลเตอร์เฉพาะเบอร์โทรศัพท์ที่มีเลข 555

✓ LIKE หา pattern แบบ case-insensitive แปลว่า 'L%' กับ '%L' จะหา pattern เดียวกันคือชื่อขึ้นต้นด้วยตัว L ไม่สนใจว่าเป็น uppercase หรือ lowercase

7.8 การเขียน Regular Expressions

ถ้าใครคิดว่า LIKE มันคูลมากแล้ว มาลองดูหัวข้อนี้ก่อน Super Cool!

Regular Expressions คือการเขียน sequence of string ที่ใช้ในการทำ pattern matching เหมือนกับ LIKE operator แต่มีความยืดหยุ่นกว่ามากในแง่ของการใช้งานจริง

Query แรก เราค้นหาชื่อลูกค้า firstname ที่มีตัวอักษร h อยู่ในชื่อ (ไม่ว่าจะอยู่ที่ตำแหน่งไหน)

```
SELECT firstname, postalcode FROM customers  
WHERE firstname REGEXP 'h';
```

	FirstName	PostalCode
1	Michelle	10012-2612
2	Kathy	89503
3	Heather	32801
4	John	2113
5	Richard	76110
6	Martha	B3S 1C5
7	Hannah	10789
8	Terhi	00530
9	Hugh	NULL
10	Johannes	1016
11	Phil	SW1V 3EN

ดึงชื่อ firstname ที่มีตัว h อยู่ด้วย

✓ ความแตกต่างอย่างแรงระหว่าง LIKE กับ REGEXP คือ REGEXP จะหา pattern แบบ case-sensitive และว่าตอนนี้ตัวพิมพ์เล็ก/ใหญ่มีผลกับการค้นหา

Query นี้เราค้นหาชื่อ firstname ที่มีตัวอักษร H ตัวใหญ่เท่านั้น จะได้ชื่อลูกค้ามาทั้งหมด 4 ชื่อ

```
SELECT firstname, postalcode FROM customers  
WHERE firstname REGEXP 'H';
```

	FirstName	PostalCode
1	Helena	14300
2	Heather	32801
3	Hannah	10789
4	Hugh	NULL

ดึงชื่อ firstname ที่มีตัว H อยู่ด้วย REGEXP ค้นหา pattern แบบ case-sensitive

แอ็ดสรุปแบบของ Regular Expressions ที่เราใช้บ่อยๆในตารางด้านล่าง

Regular Expression	ค้นหา pattern แบบใด
[a-z]	ตัวอักษร a-z แบบ lowercase
[A-Z]	ตัวอักษร A-Z แบบ uppercase
[0-9]	ตัวเลข 0-9
[a-zA-Z0-9]	ตัวอักษรทั้งตัวพิมพ์เล็ก/ใหญ่และตัวเลข (alphanumeric)
^A	ขีนต้นด้วยตัวอักษร A
A\$	ลงท้ายด้วยตัวอักษร A
\s	ค้นหา white space i.e. s ย่อมาจาก space
\d	ค้นหาตัวเลข i.e. d ย่อมาจาก digit
A a	ค้นหาตัวอักษร A หรือ a

ลองดูตัวอย่างการประยุกต์ใช้งาน Regular Expressions ใน queries ด้านล่าง

1. พิลเตอร์ชื่อลูกค้าที่ขีนต้นด้วยตัวอักษร L

```
SELECT firstname, postalcode FROM customers
WHERE firstname REGEXP '^L';
```

	FirstName	PostalCode
1	Luís	12227-000
2	Leonie	70174
3	Ladislav	H-1073
4	Lucas	00192
5	Luis	NULL

ลูกค้าที่ขีนต้นด้วยตัว L มีทั้งหมด 5 คน

2. พิลเตอร์ชื่อลูกค้าที่ขีนต้นด้วยตัวอักษร L หรือ A ได้

```
SELECT firstname, postalcode FROM customers
WHERE firstname REGEXP '^^(L|A)';
```

	FirstName	PostalCode
1	Luís	12227-000
2	Leonie	70174
3	Astrid	1010
4	Alexandre	01310-200
5	Aaron	R3L 2B9
6	Ladislav	H-1073
7	Lucas	00192
8	Luis	NULL

ลูกค้าที่ชื่อขึ้นต้นด้วยตัว L หรือ A มีทั้งหมด 8 คน

3. พิลเตอร์ชื่อลูกค้าที่ลงท้ายด้วยตัวอักษร a

```
SELECT firstname, postalcode FROM customers
WHERE firstname REGEXP 'a$';
```

	FirstName	PostalCode
1	Helena	14300
2	Kara	1720
3	Fernanda	71020-677
4	Julia	84102
5	Martha	B3S 1C5
6	Madalena	NULL
7	Emma	N1 5LH
8	Puja	560001

ลูกค้าที่ชื่อ firstname ลงท้ายด้วยตัว a มีทั้งหมด 8 คน

4. พิลเตอร์ postalcode ที่มีตัวเลขอย่างน้อยหนึ่งตัว

```
SELECT firstname, postalcode FROM customers
WHERE postalcode REGEXP '\d';
```

	FirstName	PostalCode
1	Luís	12227-000
2	Leonie	70174
3	François	H2G 1A7
4	Bjørn	0171
5	František	14700
6	Helena	14300
7	Astrid	1010
8	Daan	1000
9	Kara	1720
10	Eduardo	01007-010

ลูกค้าที่มี postalcode เป็นตัวเลขอย่างน้อยหนึ่งตัวมีทั้งหมด 55 คน

5. ฟิลเตอร์ postalcode ที่เป็นตัวเลข digit 5 ตัวติดกัน - pattern นี้จะค่อนข้างซ้อนนิดหน่อย เพราะเราใส่ {5} เข้ามาด้วย เพื่อกำหนดจำนวนตัวเลขที่เราต้องการ เช่น 5 หลัก

```
SELECT firstname, postalcode FROM customers  
WHERE postalcode REGEXP '^[0-9]{5}$';
```

	FirstName	PostalCode
1	Leонie	70174
2	František	14700
3	Helena	14300
4	Tim	95014
5	Kathy	89503
6	Heather	32801
7	Frank	60611
8	Victor	53703
9	Richard	76110
10	Patrick	85719

ลูกค้าที่มี postalcode เป็นตัวเลข 5 ตัวติดกันมีทั้งหมด 23 คน

 Regular Expressions ต้องใช้เวลาฝึกพอสมควร เพราะเราต้องจำพวก metacharacters ต่างๆให้ได้ก่อน เช่น ^ \$ | ฯลฯ แอดเดย์เขียนบทความไว้ [ที่นี่](#) ความรู้เรื่องนี้สามารถประยุกต์ใช้ได้กับภาษาโปรแกรมมิ่งอื่นๆ เช่น R และ Python มีประโยชน์!



อ่านเพิ่มเติมเรื่อง Regular Expressions ได้ [ที่นี่](#)

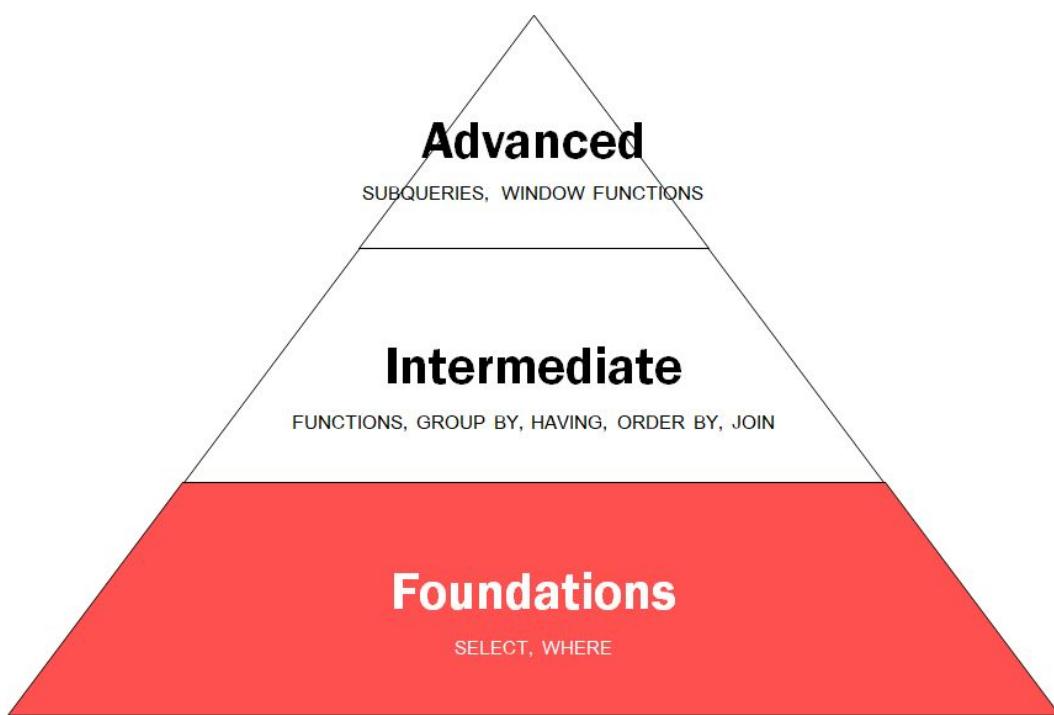
7.9 สรุปการใช้งานที่สำคัญของ WHERE

1. เราใช้ WHERE เพื่อฟิลเตอร์/กรองข้อมูลที่เราต้องการจาก table/ database
2. เวลาเขียน WHERE clause จริงๆคือการเขียนเงื่อนไขที่ใช้ในฟิลเตอร์
3. Operators ที่เราใช้ใน WHERE clause มีดังนี้: logical operators, AND, OR, NOT, IN, BETWEEN .. AND .., IS NULL, LIKE และ REGEXP
4. LIKE และ REGEXP ใช้ทำ pattern matching หา pattern ที่ต้องการในคอลัมน์ที่เราเลือก REGEXP จะมีความยืดหยุ่นและทรงพลังกว่า LIKE

บทถัดไป แอดจะอธิบาย workflow การทำงานจริงของตำแหน่ง data analyst ที่ผู้สมมติการใช้ SQL + Excel (Pivot Table) เพื่อหา insights ในข้อมูล

Congratulations!

Achievement I Unlocked



ยินดีด้วย! ตอนนี้ทุกคนจบ Foundations พื้นฐานการเขียน SQL แล้ว

ยินดีด้วย! ทุกคนเรียนจบ SQL Foundations แล้ว มาลองรีวิวเร็วๆ ว่าเราทำอะไรได้แล้วบ้าง

ตอนนี้ทุกคนสามารถเขียน SQL queries เพื่อดึงข้อมูลคอลัมน์ที่เราต้องการจาก table/database โดยใส่เงื่อนไขฟิลเตอร์ได้ด้วย

เราเรียนการดึงและสร้างคอลัมน์ (SELECT) และเรียนการเขียนฟิลเตอร์ (WHERE) ครบทุกแบบตามมาตรฐาน standard SQL

Are you getting more confidence?

ตอนนี้มั่นใจขึ้นหรือยัง? ไม่ได้ยินเลย .. ขออภัยที่ มั่นใจยัง? มั่นใจแล้ว!

ดีมาก เพราะความมั่นใจนี่แหล่ะ ที่จะทำให้เรารวยภาพผ่านตัวเองต่อไป แอดเข้าใจว่าความรู้สึกนี้ เป็นยังไง ตัวแอดเองก็เคยผ่านประสบการณ์นี้มาแล้ว 😊 ตื่นเต้นทุกครั้ง ที่ได้เรียนอะไรใหม่ๆ

Enough talk. Let's get moving.

Chapter 8 - When SQL meets Excel

แล้วเวลาที่ data analyst ดึงข้อมูลด้วย SQL เสร็จแล้ว เค้าเอาข้อมูลไปทำงานยังไงต่อ?

ปกติเราจะนำข้อมูลที่ได้ไปวิเคราะห์ต่อด้วยโปรแกรม Microsoft Excel เรียกว่า การหมุนข้อมูล โดยเครื่องมือที่เราใช้เป็นประจำคือ Pivot Table บทนี้แอดจะอธิบายวิธีการทำงานของ Pivot Table คร่าวๆ เพื่อเป็น guideline ในการทำงานกับข้อมูลที่ดึงออกมาจาก database ตัวอย่าง Excel ในหนังสือเล่มนี้ แอดใช้เวอร์ชั่น Office 365

8.1 ความแตกต่างของ Dimension และ Measurement

งานด้าน Data Analytics จะมีคำศัพท์สองคำที่เราใช้กันบ่อยมากคือ Dimension และ Measurement ก่อนจะเรียนการใช้ Pivot Table แอดขอรีวิวนิยามและความแตกต่างของสองคำนี้สั้นๆ (i.e. ความรู้เรื่องนี้เป็นพื้นฐานสำคัญเวลาใช้ Pivot Table หรือ Business Tools ตัวอื่นๆ)

- **Dimension** คือคอลัมน์ในตารางข้อมูลที่ไม่ใช่ตัวเลข ไม่สามารถคำนวณสถิติอะไรได้ นอกจากการนับความถี่ หรือ COUNT/ frequency เท่านั้น
- **Measurement** คือคอลัมน์ในตารางข้อมูลที่เป็นตัวเลข สามารถคำนวณค่าสถิติต่างๆได้ เช่น ค่าเฉลี่ย ผลรวม ค่าต่ำสุด ค่าสูงสุด ส่วนเบี่ยงเบนมาตรฐาน ความแปรปรวน ฯลฯ

โอเค! ตอนนี้ทุกคนน่าจะพอเข้าใจความแตกต่างของสองคำนี้แล้ว มาลองหมุนข้อมูลกัน

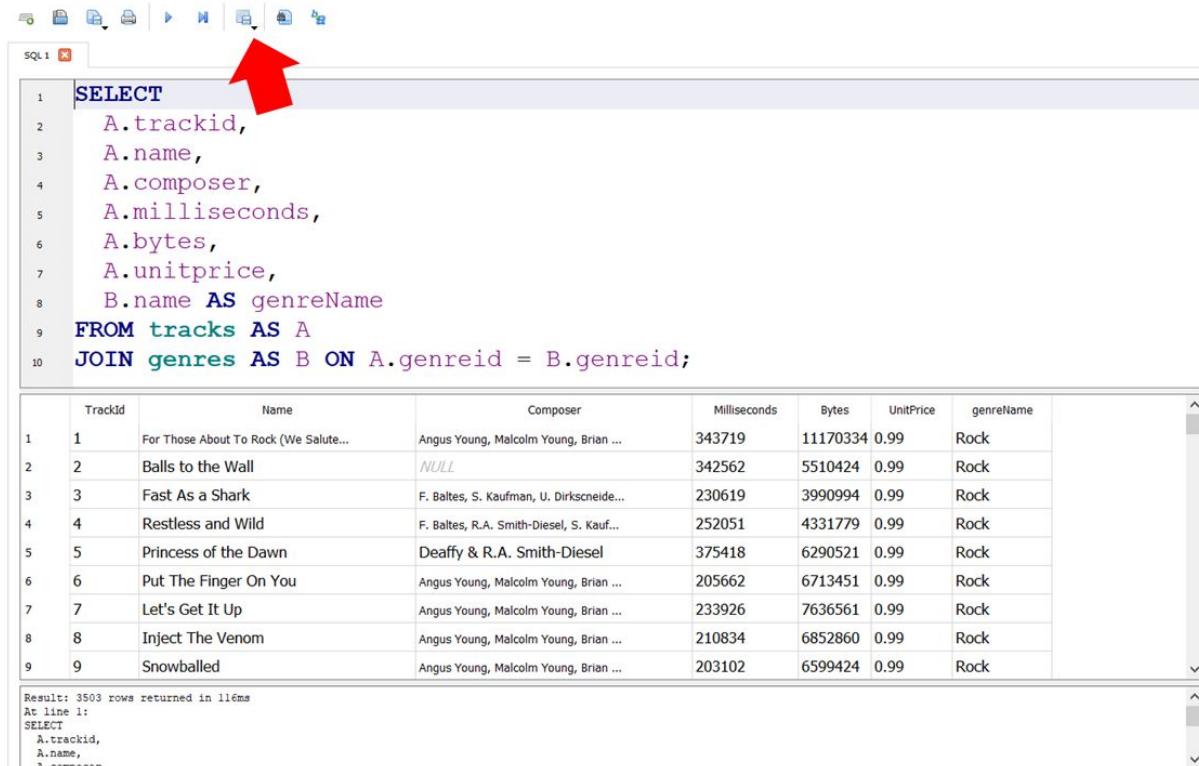
8.2 วิธี Export Result ออกจาก DB Browser

ลอง copy query นี้ไปลองรันใน DB Browser อันนี้คือตัวอย่าง query การดึงข้อมูลจากสองตาราง พร้อมกันด้วย JOIN clause

```
SELECT
    A.trackid,
    A.name,
    A.composer,
    A.milliseconds,
    A.bytes,
    A.unitprice,
    B.name AS genreName
FROM tracks AS A
JOIN genres AS B ON A.genreid = B.genreid;
```

Note - ตอนนี้ยังไม่จำเป็นต้องเข้าใจการทำงานของ query ด้านบน ให้ลองรันเลยๆก่อน

คลิกที่ไอคอน  เลือก Export to CSV และ Save File ไว้บนหน้าจอ Desktop ของนักเรียนได้เลย DB Browser จะ Export Result Set ออกมายเป็น CSV file (Comma-Separated Value) นักวิเคราะห์นิยมเปิดไฟล์ประเภทนี้ด้วยโปรแกรม Excel หรือ Text Editor ทั่วไป



```

SQL:1
SELECT
    A.trackid,
    A.name,
    A.composer,
    A.milliseconds,
    A.bytes,
    A.unitprice,
    B.name AS genreName
FROM tracks AS A
JOIN genres AS B ON A.genreid = B.genreid;

```

	TrackId	Name	Composer	Milliseconds	Bytes	UnitPrice	genreName
1	1	For Those About To Rock (We Salute...	Angus Young, Malcolm Young, Brian ...	343719	11170334	0.99	Rock
2	2	Balls to the Wall	NULL	342562	5510424	0.99	Rock
3	3	Fast As a Shark	F. Baltes, S. Kaufman, U. Dirkschneide...	230619	3990994	0.99	Rock
4	4	Restless and Wild	F. Baltes, R.A. Smith-Diesel, S. Kauf...	252051	4331779	0.99	Rock
5	5	Princess of the Dawn	Deaffy & R.A. Smith-Diesel	375418	6290521	0.99	Rock
6	6	Put The Finger On You	Angus Young, Malcolm Young, Brian ...	205662	6713451	0.99	Rock
7	7	Let's Get It Up	Angus Young, Malcolm Young, Brian ...	233926	7636561	0.99	Rock
8	8	Inject The Venom	Angus Young, Malcolm Young, Brian ...	210834	6852860	0.99	Rock
9	9	Snowballed	Angus Young, Malcolm Young, Brian ...	203102	6599424	0.99	Rock

```

Result: 3503 rows returned in 116ms
At line 1:
SELECT
    A.trackid,
    A.name,
    A.composer.

```

คลิกที่ไอコンด้านบนเพื่อ Export to CSV

8.3 หน้าตาของไฟล์ CSV (เหมือน Excel ทั่วไป)

	A	B	C	D	E	F	G
1	TrackId	Name	Composer	Milliseconds	Bytes	UnitPrice	genreName
2	1	For Those About To Rock (We Salute Yc	Angus Young, Malcolm Young, Brian Johnson	343719	11170334	0.99	Rock
3	2	Balls to the Wall		342562	5510424	0.99	Rock
4	3	Fast As a Shark	F. Baltes, S. Kaufman, U. Dirkschneide...	230619	3990994	0.99	Rock
5	4	Restless and Wild	F. Baltes, R.A. Smith-Diesel, S. Kaufman, U. Dirks	252051	4331779	0.99	Rock
6	5	Princess of the Dawn	Deaffy & R.A. Smith-Diesel	375418	6290521	0.99	Rock
7	6	Put The Finger On You	Angus Young, Malcolm Young, Brian Johnson	205662	6713451	0.99	Rock
8	7	Let's Get It Up	Angus Young, Malcolm Young, Brian Johnson	233926	7636561	0.99	Rock
9	8	Inject The Venom	Angus Young, Malcolm Young, Brian Johnson	210834	6852860	0.99	Rock
10	9	Snowballed	Angus Young, Malcolm Young, Brian Johnson	203102	6599424	0.99	Rock
11	10	Evil Walks	Angus Young, Malcolm Young, Brian Johnson	263497	8611245	0.99	Rock
12	11	C.O.D.	Angus Young, Malcolm Young, Brian Johnson	199836	6566314	0.99	Rock
13	12	Breaking The Rules	Angus Young, Malcolm Young, Brian Johnson	263288	8596840	0.99	Rock
14	13	Night Of The Long Knives	Angus Young, Malcolm Young, Brian Johnson	205688	6706347	0.99	Rock
15	14	Spellbound	Angus Young, Malcolm Young, Brian Johnson	270863	8817038	0.99	Rock
16	15	Go Down	AC/DC	331180	10847611	0.99	Rock
17	16	Dog Eat Dog	AC/DC	215196	7032162	0.99	Rock
18	17	Let There Be Rock	AC/DC	366654	12021261	0.99	Rock
19	18	Bad Boy Boogie	AC/DC	267728	8776140	0.99	Rock
20	19	Problem Child	AC/DC	325041	10617116	0.99	Rock
21	20	Overdose	AC/DC	369319	12066294	0.99	Rock

หน้าตาของ CSV file ที่เรา Export ออกมานะ (เหมือนกับไฟล์ Excel ทั่วไป)

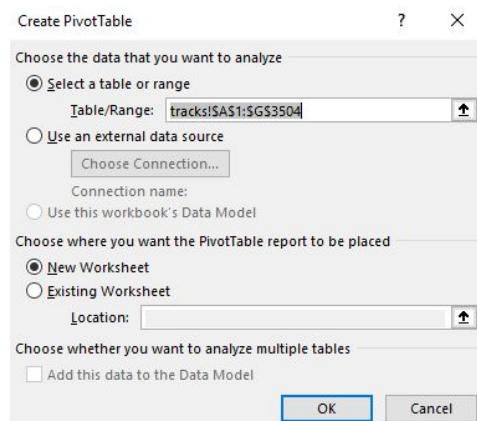
นักเรียนสามารถดาวน์โหลดไฟล์ CSV ของตัวอย่างนี้ได้ที่นี่ [ที่นี่](#)

8.4 วิธีการหมุนข้อมูลด้วย Pivot Table

1. เปิดไฟล์ CSV ที่เรา export ออกมายัง DB Browser ด้วยโปรแกรม Excel
2. คลิกที่ข้อมูลของเรา กดปุ่ม CTRL+A เพื่อเลือกข้อมูลทั้งหมด ไปที่แท็บ Insert > Pivot Table และคลิก OK ได้เลย (ไม่ต้องเปลี่ยน option ใดๆ)

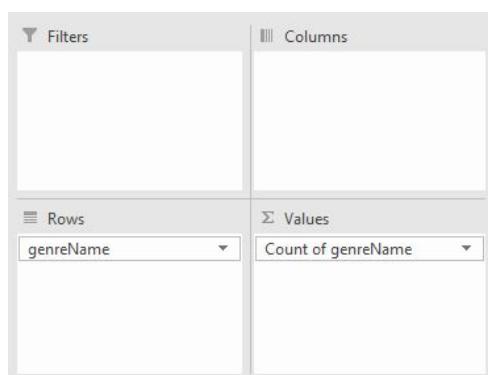
The screenshot shows a Microsoft Excel spreadsheet titled 'tracks.csv'. The 'Insert' tab is active. A tooltip 'PivotTable' is displayed over the table area. The table contains data from columns C through G, with headers 'Composer', 'Milliseconds', 'Bytes', 'UnitPrice', and 'genreName'. The 'genreName' column has values like 'Rock', 'Rock', 'Rock', etc.

คลิกที่แท็บ Insert เลือก Pivot Table



คลิก OK ได้เลย

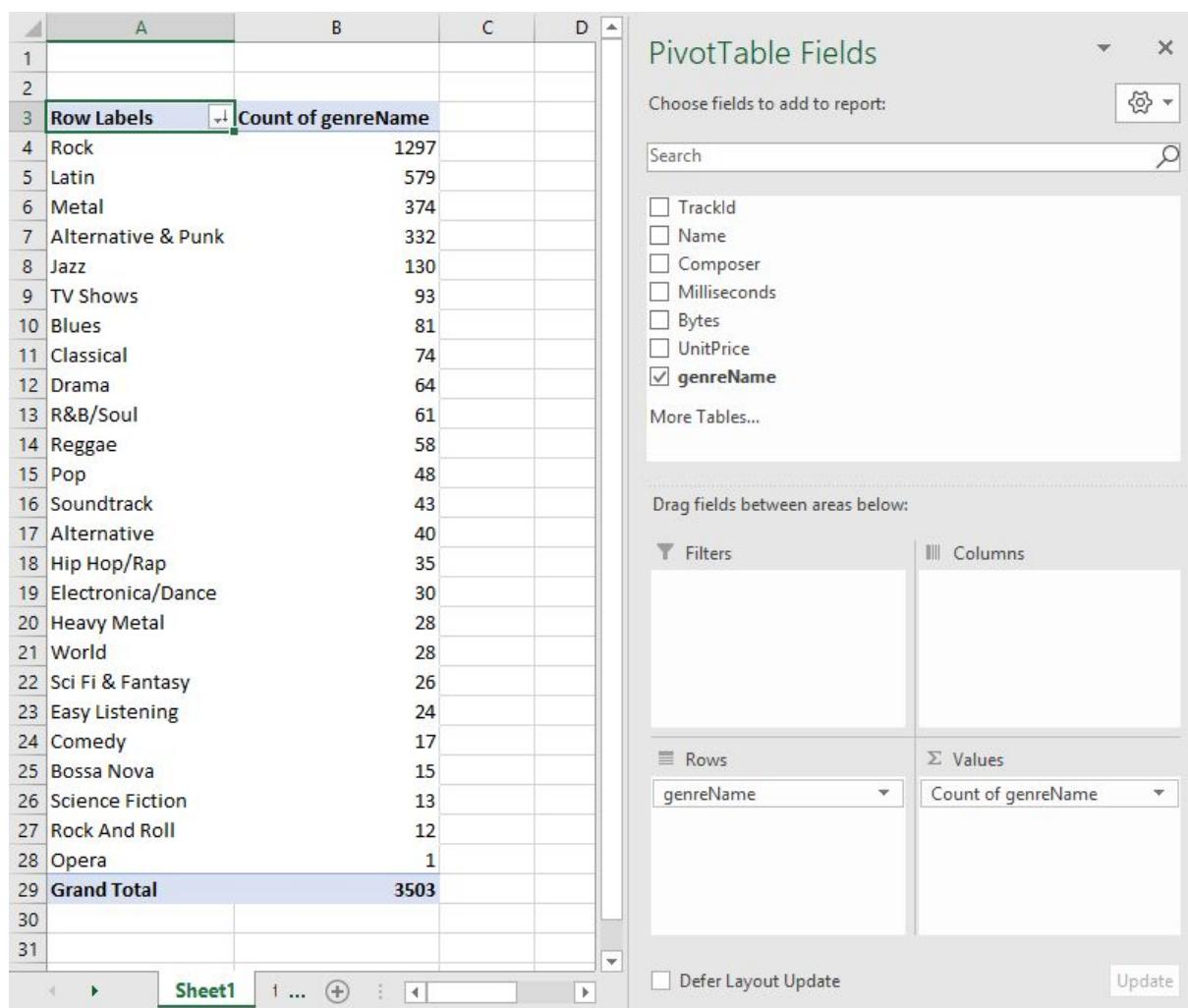
3. มาลองนับจำนวนเพลงในแต่ละ genre กันก่อน ให้เราดึงคอลัมน์ genreName ไปใส่ที่ Rows Field และนำคอลัมน์ genreName ตัวเดิมไปใส่ที่ Values Field ตามรูปด้านล่าง



ใช้เม้าส์ลากคอลัมน์ไปวางใน field ตามรูปนี้

Pivot Table จะสร้างรายงาน (report) ทางด้านซ้ายมือของหน้าจอ เป็นรายงานที่นับจำนวน เพลงในแต่ละ genre จากเราสามารถสรุป insight แบบเร็วๆ ได้ดังนี้

- จำนวนเพลงทั้งหมดคือ **3,503 เพลง** (cell B29)
- genre ที่มีจำนวนเพลงเยอะที่สุดสามอันดับแรกคือ Rock 1,297 เพลง Latin 579 เพลง และ Metal 374 เพลง ตามลำดับ
- genre ที่มีจำนวนเพลงน้อยที่สุดสามอันดับล่างคือ Opera 1 เพลง Rock And Roll 12 เพลง และ Science Fiction 13 เพลง ตามลำดับ



Pivot Table ช่วยให้เราวิเคราะห์ข้อมูลหา Insight ได้อย่างรวดเร็ว

✓ เราสามารถ Sort Rows ใน Pivot Table ได้ปกติ เหมือนกับใช้งาน Excel Table ทั่วไปเลย ไปที่แท็บ Data และคลิกไอคอน เพื่อเรียงข้อมูลจากน้อยไปมาก (ascending) หรือ เพื่อเรียงข้อมูลจากมากไปน้อย (descending)

Chapter 9 - SQL Functions

ฟังชั่นใน SQL ใช้งานเหมือนกับฟังชั่นใน Excel เลย โดยแอดแบ่งฟังชั่นของ SQL ออกเป็น 3 กลุ่ม

1. Aggregate Functions - ใช้สรุปผลสถิติ เช่น AVG SUM MIN MAX COUNT
2. Value Functions - ใช้ปรับ format ของ value ในคอลัมน์ที่เราต้องการ
3. Window Functions

บทนี้เราจะอธิบายวิธีใช้งาน Aggregate Functions และ Value Functions ก่อน ส่วน Window Functions (Analytics) จะอยู่ในบทที่ 19

9.1 สรุปผลสถิติด้วย Aggregate Functions

Aggregate Functions คือฟังชั่นสถิติเบื้องต้นของภาษา SQL โดยตัวหลักจะมีอยู่แค่ 5 ฟังชั่น ประกอบด้วย avg, sum, min, max และ count ตัวอย่างนี้เราสรุปผลสถิติของคอลัมน์ bytes ในตาราง tracks

```
SELECT
    AVG(bytes),
    SUM(bytes),
    MIN(bytes),
    MAX(bytes),
    COUNT(bytes)
FROM tracks;
```

	AVG(bytes)	SUM(bytes)	MIN(bytes)	MAX(bytes)	COUNT(bytes)
1	33510207.0653725	117386255350	38747	1059546140	3503

Result set ของ aggregate functions

แอดเฉียนสรุปวิธีการใช้งานของแต่ละฟังชั่นในตารางด้านล่าง

Aggregate Functions	ใช้ทำอะไร
AVG	หาค่าเฉลี่ย
SUM	หาผลรวม
MIN	หาค่าต่ำที่สุด
MAX	หาค่าสูงที่สุด
COUNT	นับจำนวน records/ rows

9.2 ตั้งชื่อผลลัพธ์ของ Aggregate Functions

เราสามารถตั้งชื่อแต่ละคอลัมน์ของ aggregated results ด้วย AS (AS ย่อมาจาก Alias)

```
SELECT
    AVG(bytes) AS avg_bytes,
    SUM(bytes) AS sum_bytes,
    MIN(bytes) AS min_bytes,
    MAX(bytes) AS max_bytes,
    COUNT(bytes) AS count_bytes
FROM tracks;
```

	avg_bytes	sum_bytes	min_bytes	max_bytes	count_bytes
1	33510207.0653725	117386255350	38747	1059546140	3503

เปลี่ยนชื่อคอลัมน์ใหม่

9.3 นับจำนวน records ทั้งหมดใน table

วิธีการนับจำนวน records/ rows ทั้งหมดในตาราง ให้เราเขียน select count(*) from table_name; ได้เลย

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

เราสามารถ count(column_name) เพื่อนับจำนวน completed records ในคอลัมน์ที่เราเลือกได้ ด้วย ที่แสดงใช้คำว่า completed เพราะว่าฟังชัน count จะนับเฉพาะ records ที่ไม่ใช่ค่า NULL

✓ Aggregate Functions ทั้งหมดที่เรารู้ในหนังสือเล่มนี้จะไม่สนใจ records ที่มีค่า NULL เลย หรือพูดง่ายๆคือ Aggregate Functions จะ ignore NULL records ในคอลัมน์ที่เราเลือก

ตัวอย่าง query ถัดไป แสดงนับจำนวน records ทั้งหมดใน tables และนับจำนวน records ใน คอลัมน์ company ลองเปรียบเทียบตัวเลขที่ได้จาก query นี้

```
SELECT
    COUNT(*),
    COUNT(company)
FROM customers;
```

	COUNT(*)	COUNT(company)
1	59	10

จำนวน completed records ในคอลัมน์ company มีแค่ 10 records

- 59 คือจำนวน records ทั้งหมดของ customers table
- 10 คือจำนวน completed records (NOT NULL) ของคอลัมน์ company

	Company
1	Embraer - Empresa Brasileira de Aer...
2	NULL
3	NULL
4	NULL
5	JetBrains s.r.o.
6	NULL
7	NULL
8	NULL
9	NULL
10	Woodstock Discos

ถ้าเราลอง query คอลัมน์ company ขึ้นมาดูจะเห็นค่า NULL หลายແກ່ໄລ

9.4 นับจำนวน Distinct Values ที่ไม่ซ้ำกันเลย

อีกหนึ่งเทคนิคที่แอดอยากให้ทุกคนใช้ให้เป็นเรียกว่าการ count distinct เป็นเทคนิคการนับจำนวน distinct/ unique values ที่อยู่ในคอลัมน์นั้นๆ ตัวอย่าง query ด้านล่างใช้นับจำนวนประเทศแบบ ไม่นับซ้ำ ของคอลัมน์ country ใน customers table

```
SELECT COUNT(DISTINCT country) FROM customers;
```

คำตอบของ query นี้เท่ากับ 24 ประเทศ

9.5 ปรับ Format ของข้อมูลด้วย Value Functions

Value functions จะมีรายละเอียดเดียวกับ aggregate functions พoSมคວR ถ้าใครเคยใช้ Excel มา ก่อน น่าจะเรียน value functions ได้เร็ว เพราะมันเขียนคล้ายๆกันเลย

เราใช้ value functions เพื่อปรับหน้าตาของคอลัมน์ที่เราสนใจ ตัวอย่างเช่น การปรับคอลัมน์ firstname ใน customers table ให้เป็น uppercase, lowercase หรือ ดึงตัวอักษรสามตัวแรกของชื่อ ลองรัน query ด้านล่าง และดูผลลัพธ์ใน result set

```
SELECT
    firstname,
    UPPER(firstname) AS upperName,
    LOWER(firstname) AS lowerName,
    SUBSTR(firstname, 1, 3) AS shortName
FROM customers;
```

	FirstName	upperName	lowerName	shortName
1	Luis	LUÍS	luís	Luí
2	Leonie	LEONIE	leonie	Leo
3	François	FRANÇOIS	françois	Fra
4	Bjørn	BJØRN	bjørn	Bjø
5	František	FRANTIŠEK	františek	Fra
6	Helena	HELENA	helena	Hel
7	Astrid	ASTRID	astrid	Ast
8	Daan	DAAN	daan	Daa
9	Kara	KARA	kara	Kar
10	Eduardo	EDUARDO	eduardo	Edu

query นี้เราใช้ value functions ปรับชื่อจริง ให้เป็นตัวพิมพ์ใหญ่ ตัวพิมพ์เล็ก และ substr() ตัวอักษรที่ 1-3 ตามลำดับ

✓ ลองสังเกตชื่อคอลัมน์ที่เราตั้งใน result set ด้านบน: upperName, lowerName, shortName โปรแกรมเมอร์เรียกวิธีการตั้งชื่อแบบนี้ว่า “**camel case**” หลังอูฐ! 🐂 ทุกครั้งที่ขึ้นคำใหม่ เราจะเขียนตัวพิมพ์ใหญ่ เช่น companyBudget, myHighSchool เป็นต้น

ฟังชั่น SUBSTR() ใช้ดึงจำนวน characters ออกมาจากคอลัมน์ SUBSTR(firstname, 1, 3) สั่งให้ SQL เริ่มดึง character ตั้งแต่ตัวที่ 1 และดึงออกมาทั้งหมด 3 characters

อีกหนึ่ง value function ที่ใช้บ่อยมากคือ ROUND() ไว้ใช้กำหนดจำนวนทศนิยมของคอลัมน์ที่เป็นตัวเลขแบบ numeric (i.e. ตัวเลขที่มีทศนิยม) ลองดูตัวอย่างการใช้ ROUND() ด้านล่าง

```
SELECT
AVG(bytes),
ROUND(AVG(bytes), 2) AS roundMeanByte
FROM tracks;
```

	AVG(bytes)	roundMeanByte
1	33510207.0653725	33510207.07

round() ใน query นี้ปรับทศนิยมให้เหลือสองตำแหน่งเท่านั้น

9.6 วิธีจัดการค่า NULL (missing values)

ค่า NULL เปรียบเสมือน missing values ใน SQL database ถ้าคอลัมน์ไหนมีค่า NULL เราสามารถใช้ฟังชั่น COALESCE() เพื่อแทนที่ค่า NULL ได้ง่ายๆ ตัวอย่าง query ด้านล่าง เราใช้ COALESCE() เพื่อแทนที่ค่า NULL ในคอลัมน์ company ด้วยคำว่า 'End Customer'

```
SELECT
    company,
    COALESCE(company, 'End Customer') AS cleanCompany
FROM customers;
```

	Company	cleanCompany
1	Embraer - Empresa Brasileira de Aer...	Embraer - Empresa Brasileira de Aer...
2	<i>NULL</i>	End Customer
3	<i>NULL</i>	End Customer
4	<i>NULL</i>	End Customer
5	JetBrains s.r.o.	JetBrains s.r.o.
6	<i>NULL</i>	End Customer
7	<i>NULL</i>	End Customer
8	<i>NULL</i>	End Customer
9	<i>NULL</i>	End Customer
10	Woodstock Discos	Woodstock Discos

ถ้าเราลอง query คอลัมน์ company ขึ้นมาดูจะเห็นค่า NULL หลายແລກ

9.7 Value Functions สำหรับปรับหน้าตา Date

ถ้าต้องการเรียกดูวันที่ปัจจุบัน เราสามารถใช้ฟังชั่น DATE() ตามตัวอย่างนี้

```
SELECT DATE('now');
```

✓ SQLite ไม่ได้รองรับคอลัมน์แบบ datetime ตัวอย่างเช่น เวลาเราเห็น '2019-01-01' ในคอลัมน์ (เช่น invoicedate) ให้รู้ว่าันคือการเก็บข้อมูล text ที่ format เมื่อนกับวันที่/เวลา

	InvoiceDate
1	2009-01-01 00:00:00
2	2009-01-02 00:00:00
3	2009-01-03 00:00:00
4	2009-01-06 00:00:00
5	2009-01-11 00:00:00
6	2009-01-19 00:00:00
7	2009-02-01 00:00:00
8	2009-02-01 00:00:00
9	2009-02-02 00:00:00
10	2009-02-03 00:00:00

พิจารณาคอลัมน์ invoicedate ใน invoices table

Data analyst นิยมใช้ฟังชั่น STRFTIME() เพื่อปรับ format ของคอลัมน์ datetime ใน SQL database ตัวอย่าง query ด้านล่าง เราปรับ format ของคอลัมน์ invoicedate ให้แสดงเฉพาะปี (%Y) เดือน (%m) และวันที่ (%d) ตั้งชื่อคอลัมน์ใหม่ว่า year, month, day ตามลำดับ

```
SELECT
    invoicedate,
    STRFTIME('%Y', invoicedate) AS year,
    STRFTIME('%m', invoicedate) AS month,
    STRFTIME('%d', invoicedate) AS day
FROM invoices;
```

	InvoiceDate	year	month	day
1	2009-01-01 00:00:00	2009	01	01
2	2009-01-02 00:00:00	2009	01	02
3	2009-01-03 00:00:00	2009	01	03
4	2009-01-06 00:00:00	2009	01	06
5	2009-01-11 00:00:00	2009	01	11
6	2009-01-19 00:00:00	2009	01	19
7	2009-02-01 00:00:00	2009	02	01
8	2009-02-01 00:00:00	2009	02	01
9	2009-02-02 00:00:00	2009	02	02
10	2009-02-03 00:00:00	2009	02	03

ถ้าเราลอง query คอลัมน์ company ขึ้นมาดูจะเห็นค่า NULL หลายແລກ



อ่านวิธีการจัดการ datetime ของ SQLite แบบละเอียดได้ที่เว็บ [SQLite official](#)

9.8 วิธีการใช้งาน value functions ใน WHERE clause

หัวข้อนี้มาลองดูวิธีการประยุกต์ใช้ value functions ใน WHERE clause กันบ้าง use-case ที่แอดใช้บ่อยๆคือเวลาที่เราต้องการเขียนพิลเตอร์คอลัมน์แบบ text เช่น ชื่อประเทศ แต่ชื่อประเทศที่อยู่ในคอลัมน์นี้ อาจมีความไม่ consistent กัน

สมมติชื่อประเทศอังกฤษ ในคอลัมน์ country มีอยู่หลายแบบคือ united kingdom, United Kingdom, UNITED KINGDOM หรือ UniTED KingDOM (i.e. ไม่ consistent)

เราสามารถแก้ปัญหานี้ได้ง่ายๆด้วยการใช้ UPPER() กับคอลัมน์ country ใน WHERE clause

```
SELECT firstname, lastname, country
FROM customers
WHERE UPPER(country) = 'UNITED KINGDOM';
```

	FirstName	LastName	Country
1	Emma	Jones	United Kingdom
2	Phil	Hughes	United Kingdom
3	Steve	Murray	United Kingdom

ถ้าเราใช้ `upper()` มาช่วยเขียนเงื่อนไข ก็จะสามารถถึง `United Kingdom` ออกมายังหมด

9.9 สรุปการใช้งานฟังชันเบื้องพื้นฐานของ SQL

- Standard SQL มีฟังชันทั้งหมดสามแบบคือ aggregate functions, value functions และ window functions (เราระอธิบาย window functions ในบทที่ 19)
- Aggregate functions ใช้สรุปผลสถิติเบื้องต้น ตัวหลักๆที่เราใช้มี `avg`, `sum`, `min`, `max` และ `count`
- Value functions ใช้ปรับ `format` ของคอลัมน์ที่เราต้องการ เช่น `lower`, `upper`, `substr`, `round`, `coalesce` และ `strftime` วิธีการเขียนจะคล้ายกับฟังชันของ Excel



แอดสรุป value functions ทั้งหมดที่เราระอธิบายในบทนี้ในตารางด้านล่าง พร้อม reference สำหรับนักเรียนที่อยากรู้เพิ่มเติม

Value Functions	ใช้ทำอะไร
<u>LOWER</u>	ปรับ text เป็นตัวพิมพ์เล็กทั้งหมด
<u>UPPER</u>	ปรับ text เป็นตัวพิมพ์ใหญ่ทั้งหมด
<u>SUBSTR</u>	ดึง characters ที่จุดเริ่มต้นและจำนวนที่เรากำหนด
<u>ROUND</u>	ปรับจำนวนทศนิยม
<u>COALESCE</u>	แทนค่า <code>NULL</code> ในคอลัมน์
<u>STRFTIME</u>	ปรับ <code>format</code> ของคอลัมน์ <code>datetime</code> (text)

แอดทอย: “มันยากขึ้นนิดหน่อยใช่มั้ย? อ่ายลืมว่าทำไมเราอยู่ตรงนี้ แอดรู้ว่าเนื้อหาเริ่มยากขึ้น ต้องใช้เวลาอ่านทำความเข้าใจนานขึ้น แต่นี่เราก็เดินทางมาจะครึ่งทางแล้ว ถ้าใครเห็นอยู่ก็พักก่อน แล้วพรุ่งนี้กลับมาอ่านต่อ ก็ได้ (แต่ถ้าใครยังไหว ลุยก่อนบทที่ 10 เลยพี)”

Chapter 10 - GROUP BY

มีกฎเหล็กสองข้อที่ทุกคนต้องรู้ ถ้าอยากจะใช้ GROUP BY กับ query ที่เราเขียน

1. เราใช้ GROUP BY คู่กับ Aggregate Functions เช่น (avg, sum, min, max, count)
2. ทุกคอลัมน์ที่อยู่ใน SELECT clause ที่ไม่ใช่ aggregate functions ต้องอยู่ใน GROUP BY clause เช่น หรืออธิบายอีกแบบคือ คอลัมน์ไหนที่อยู่ใน GROUP BY clause ต้องอยู่ใน SELECT clause เช่นกัน

วิธีการเรียน GROUP BY ที่ดีที่สุดคือดูตัวอย่าง แล้วลองเขียน query ตามเลย

 สรุปง่ายๆ GROUP BY + Aggregate Functions คือการสรุปผลสถิติแบ่งตามกลุ่ม เช่น หาค่าเฉลี่ยเงินเดือนพนักงานของแต่ละแผนก (การตลาด การเงิน การผลิต ฝ่ายขาย) หรือ นับจำนวนลูกค้าที่อยู่ในแต่ละประเทศ เป็นต้น

10.1 การจับกลุ่มแบบหนึ่งคอลัมน์

Query นี้เรานับจำนวนลูกค้าทั้งหมดใน customers table ได้คำตอบเท่ากับ 59 คน

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

ถ้าเราอยากรู้ว่า 59 คนนี้มาจากประเทศใดกี่คน จากประเทศอังกฤษกี่คน จากประเทศเบลเยียมกี่คน ฯลฯ เราแค่เขียน GROUP BY country ต่อท้าย query นี้ได้เลย

อย่าลืมกฎสองข้อที่เราเริ่วไปตอนต้นของบทนี้ คอลัมน์ไหนอยู่ใน GROUP BY คอลัมน์นั้นต้องอยู่ใน SELECT clause เช่น

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

 คำแนะนำ - ปกติแอ็อดจะเขียนชื่อคอลัมน์ใน GROUP BY ก่อน แล้วค่อยเขียนชื่อคอลัมน์นั้น อีกครั้งใน SELECT clause

	Country	n
1	Argentina	1
2	Australia	1
3	Austria	1
4	Belgium	1
5	Brazil	5
6	Canada	8
7	Chile	1
8	Czech Republic	2
9	Denmark	1
10	Finland	1

Result set ที่ได้จาก GROUP BY country

- ✓ ถ้าทุกคนยังจำกัดการใช้ Pivot Table ในบทที่ 8 ได้ เราสามารถสร้างตาราง result set แบบนี้ด้วย Pivot Table ได้โดยง่ายๆ แค่เรา export CSV file ข้อมูลลูกค้าและชื่อประเทศไปหมุนใน Excel ก็จบเลย (ได้คำตอบเหมือนกัน ไม่ต้องเขียน GROUP BY ให้มีอยู่มือ อ้าว) 555+

10.2 การจับกลุ่มแบบมากกว่าหนึ่งคอลัมน์

มาลองดูตัวอย่าง การเขียน GROUP BY แบบมากกว่าหนึ่งคอลัมน์บ้าง ตัวอย่างนี้ เราจับกลุ่ม COUNT(*) ด้วยคอลัมน์ country และ state ตามลำดับ กฎสองข้อที่เรารอธิบายไปตอนแรกก็ยังใช้กับการเขียน query นี้เหมือนเดิม (Golden Rules!)

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

ข้อดีของ GROUP BY คือเราไม่จำเป็นต้องเขียนชื่อคอลัมน์แบบเต็มๆ ก็ได้ แต่เขียนเป็น column index เช่น 1, 2 แทน ตามตัวอย่างด้านล่าง โดยที่ 1 คือตำแหน่งของคอลัมน์ country และ 2 คือตำแหน่งของคอลัมน์ state ใน **Result Set** ลองวิเคราะห์ผลลัพธ์ที่ได้จาก query นี้ด้านล่าง

```
SELECT
    country,
    state,
    COUNT(*) AS n
FROM customers
GROUP BY 1, 2;
```

	Country	State	n
1	Argentina	NULL	1
2	Australia	NSW	1
3	Austria	NULL	1
4	Belgium	NULL	1
5	Brazil	DF	1
6	Brazil	RJ	1
7	Brazil	SP	3
8	Canada	AB	1
9	Canada	BC	1
10	Canada	MB	1

Result set ที่ได้จาก GROUP BY country, state

10.3 การเขียน GROUP BY ในชีวิตจริง

ในชีวิตจริง เราอาจใช้ GROUP BY กับหลายๆ คอลัมน์ รวมถึงหลายๆ Aggregate Functions พร้อมกันก็ได้ ลอง copy query นี้ไปรันใน DB Browser และลองวิเคราะห์ผลลัพธ์ด้วยตัวเอง ตัวอย่างนี้ใช้ JOIN ใน query อีกแล้ว เดียวแอดจะอธิบาย JOIN อย่างละเอียดในบทที่ 13 😊

```

SELECT
    B.genreid,
    B.name,
    COUNT(A.name) AS count_n,
    AVG(A.bytes/ (1024.0 * 1024.0)) AS avg_megabytes,
    SUM(A.milliseconds/ 60000.0/ 60.0) AS total_hours
FROM tracks AS A
JOIN genres AS B ON A.genreid = B.genreid
GROUP BY 1, 2
ORDER BY 3
LIMIT 10;

```

	GenreId	Name	count_n	avg_megabytes	total_hours
1	25	Opera	1	2.72890853881836	0.0485591666666667
2	5	Rock And Roll	12	2.02490067481995	0.4488116666666667
3	18	Science Fiction	13	483.588203576895	9.48114944444444
4	11	Bossa Nova	15	6.90370273590088	0.914958333333333
5	22	Comedy	17	302.223649754244	7.4859675
6	12	Easy Listening	24	5.87512763341268	1.26109472222222
7	20	Sci Fi & Fantasy	26	508.242059854361	21.0295441666667
8	13	Heavy Metal	28	9.03582777295794	2.31352277777778
9	16	World	28	6.79899842398507	1.7494075
10	15	Electronica/Dance	30	10.1966156641642	2.524881666666667

Result set ที่ได้จาก GROUP BY country, state

Chapter 11 - HAVING



กฎเหล็กข้อเดียว (Super Golden Rule) ของ HAVING คือต้อง ใช้หลัง GROUP BY เท่านั้น
บทนี้จะเป็นบทที่เนื้อหาน้อยที่สุดเลย เพราะแอดสอนไปหมดแล้ว! อ้าว 555+

- ทุก operators ที่เราเรียนในบท WHERE clause สามารถใช้กับ HAVING ได้หมดเลย
- ต่างกันนิดเดียวที่ WHERE ใช้ฟิลเตอร์ข้อมูลดิบใน table และ HAVING ใช้ฟิลเตอร์ groups ที่เราสร้างขึ้นมาด้วย GROUP BY

11.1 HAVING ใช้ฟิลเตอร์กลุ่ม (ที่ได้จาก GROUP BY)

Query นี้เราฟิลเตอร์เฉพาะประเทศที่มีจำนวนลูกค้ามากกว่าหรือเท่ากับ 5 คนขึ้นไป เวลาเขียน HAVING ให้เขียนต่อจาก GROUP BY clause ได้เลย

```
SELECT
    country,
    COUNT(*) AS n
FROM customers
GROUP BY country
HAVING n >= 5;
```

	Country	n
1	Brazil	5
2	Canada	8
3	France	5
4	USA	13

มีแค่ 4 ประเทศที่มีลูกค้า ≥ 5 คนขึ้นไป

มาลองดูอีกหนึ่ง query ต่อยอดจากเมื่อกี้ คราวนี้เราเพิ่มเงื่อนไขฟิลเตอร์คือ “ประเทศที่มีลูกค้ามากกว่าหรือเท่ากับ 5 คนขึ้นไป และต้องไม่ใช่ประเทศอเมริกา”

```
SELECT
    country,
    COUNT(*) AS n
FROM customers
GROUP BY country
HAVING n >= 5 AND country <> 'USA';
```

	Country	n
1	Brazil	5
2	Canada	8
3	France	5

ตอนนี้ result set เหลือแค่สามประเทศเท่านั้น

11.2 อธิบายความแตกต่างของ WHERE vs. HAVING

สอง queries ที่เราเขียนในหัวข้อ 11.1 เป็นการเขียนเงื่อนไขใน HAVING clause ทั้งหมดเลย ความแตกต่างที่สำคัญมากของ WHERE และ HAVING อยู่ที่ลำดับของการฟิลเตอร์

✓ ลำดับของการฟิลเตอร์ WHERE จะเกิดก่อน HAVING เสมอ เพราะเราเขียน WHERE clause ก่อน HAVING clause ลองดูตัวอย่าง query ต่อไปนี้

```
SELECT
    country,
    COUNT(*) AS n
FROM customers
WHERE country <> 'USA'
GROUP BY country
HAVING n >= 5;
```

Query นี้ได้ผลลัพธ์เหมือนกับ query ในหัวข้อที่แล้ว แต่การฟิลเตอร์จะเกิดขึ้นสองครั้ง

1. การฟิลเตอร์ครั้งแรกเกิดขึ้นที่ WHERE country <> 'USA'
2. การฟิลเตอร์ครั้งที่สองเกิดขึ้นที่ HAVING n >= 5

11.3 เมื่อไหร่จะใช้ WHERE เมื่อไหร่จะใช้ HAVING

ใช้ WHERE ถ้าเรารู้ตั้งแต่แรกว่าอยากรู้ข้อมูลแบบไหนสำหรับงานของเรา ถ้าเราไม่ต้องการประเทศอเมริกาและประเทศอังกฤษตั้งแต่แรก ก็เขียน WHERE country NOT IN ('USA', 'United Kingdom') ได้เลย

ใช้ HAVING ถ้าไม่ชัวร์ว่าอยากรู้อะไรตั้งแต่แรก ลองรัน GROUP BY ดูก่อน แล้วค่อยมาฟิลเตอร์ groups ที่เราไม่ต้องการออกทีหลัง

✓ ความแตกต่างเรื่องการใช้ WHERE กับ HAVING เป็นคำ功用ด้วยเวลาสัมภาษณ์งานเลย มีครั้งหนึ่งแอดเดย์ไปสัมภาษณ์ที่ Agoda ก็โดนคำถามนี้

Chapter 12 - ORDER BY

เราใช้ ORDER BY เพื่อเรียงข้อมูล (sorting) จากต่ำไปสูง (ascending order) หรือสูงไปต่ำ (descending order) สามารถ sort ได้ทั้งคอลัมน์ที่เป็นตัวเลขหรือตัวอักษรก็ได้



ตำแหน่งของ ORDER BY ต้องอยู่หลังจาก WHERE, GROUP BY, HAVING เสมอ

12.1 เรียงข้อมูลแบบ Ascending

Query แรกใช้เรียงข้อมูลด้วยคอลัมน์ firstname แบบต่ำไปสูง (หรือ A ไป Z) ลองวิเคราะห์ผลลัพธ์ที่ได้จาก query นี้ด้านล่าง

```
SELECT
    firstname,
    country
FROM customers
ORDER BY firstname;
```

	FirstName	Country
1	Aaron	Canada
2	Alexandre	Brazil
3	Astrid	Austria
4	Bjørn	Norway
5	Camille	France
6	Daan	Belgium
7	Dan	USA
8	Diego	Argentina
9	Dominique	France
10	Eduardo	Brazil

เรียงชื่อจริงลูกค้าจาก A ไป Z

12.2 เรียงข้อมูลแบบ Descending

ถ้าต้องการเรียงข้อมูลแบบสูงไปต่ำ (หรือ Z ไป A) แค่ใส่ DESC ต่อท้ายชื่อคอลัมน์นั้นใน ORDER BY clause กดรันคิวเพื่อดูผลลัพธ์

```
SELECT
    firstname,
    country
FROM customers
ORDER BY firstname DESC;
```

12.3 เรียงข้อมูลมากกว่าหนึ่งคอลัมน์

ถ้าต้องการเรียงข้อมูล (i.e. sort rows) มากกว่าหนึ่งคอลัมน์ ให้เราเขียนชื่อคอลัมน์ และใช้ comma , ใน ORDER BY clause ได้เลย

query ด้านล่างจะเรียงคอลัมน์ country ก่อน เสร็จแล้วค่อยเรียงคอลัมน์ firstname ตามลำดับ

```
SELECT
    firstname,
    country
FROM customers
ORDER BY country, firstname;
```

12.4 เรียงข้อมูลด้วย column index

เราสามารถเรียงข้อมูลใน ORDER BY clause ด้วย column index (ตัวเลขตำแหน่งคอลัมน์) แทน การเขียนชื่อคอลัมน์เต็มๆ query ด้านล่างจะได้ผลลัพธ์เหมือนกับ query ก่อนหน้านี้โดยเรียง country ก่อนเสร็จแล้วค่อยเรียง firstname ตามลำดับ แบบต่อไปสูง (ascending order)

```
SELECT
    firstname,
    country
FROM customers
ORDER BY 2, 1;
```

ถ้าต้องการเรียงข้อมูลแบบ descending order แค่ใส่ keyword DESC ต่อท้าย column index ได้เลย ทำไม้มันง่ายอย่างนี้ 😊

```
SELECT
    firstname,
    country
FROM customers
ORDER BY 2 DESC, 1 DESC;
```

✓ ทั้ง ORDER BY และ GROUP BY ที่เราเรียนมาทั้งหมด สามารถใช้ column index (i.e. ตัวเลขตำแหน่งคอลัมน์) แทนการเขียนชื่อคอลัมน์เต็มๆ ได้เลย Neat Trick!

Chapter 13 - JOIN

เราเดินทางมาถึงบทที่สำคัญที่สุดของหนังสือเล่มนี้แล้ว บทนี้เราจะเรียน syntax การเขียน JOIN clause เพื่อดึงข้อมูลจากหลายๆ tables พร้อมกัน

12 บทที่ผ่านมา เราเขียน query เพื่อดึงข้อมูลจากหนึ่ง table แต่หลังจากอ่านบทนี้จบทุกคนจะสามารถดึงข้อมูลจากทุก tables ใน chinook database ได้เลย และ JOIN คือบทสุดท้ายของระดับ Intermediate SQL .. พร้อมที่จะลุยกันยัง?



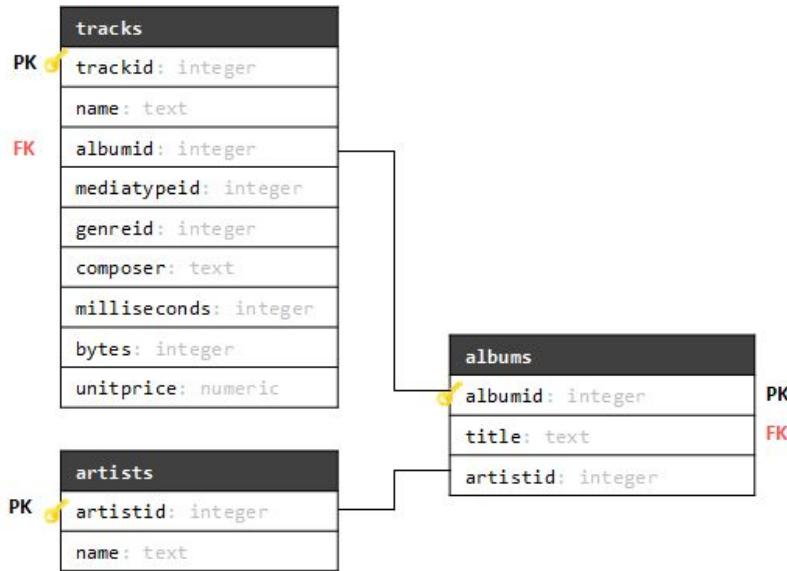
ก่อนที่เราจะเรียน JOIN syntax ตอนนี้แนะนำสมแก่เวลาที่แออดจะอธิบาย/ รีวิวตอนเช็ปต์เรื่อง ER Diagram, Primary Key และ Foreign Key อีกรอบหนึ่ง

13.1 รีวิว Primary Key และ Foreign Key

มาบททวนเนื้อหาในบทที่ 3 กันนิดนึง

- ER Diagram คือ ไดอะแกรมที่แสดงความสัมพันธ์ของ tables ทั้งหมดใน database
- Primary Key คือ คอลัมน์ที่มีแม่กุญแจสีเหลือง แต่ละ table จะมีหนึ่งคอลัมน์เท่านั้น
ยกเว้น `playlist_track` ที่ใช้ค่อนเช็ปต์ composite key (ถ้าใครจำไม่ได้ กลับไปอ่านบท 3)

แต่ถ้า Primary Key หรือตัวย่อ **PK** ของ `artists` table (คอลัมน์ `artistid`) ไปผลลัพธ์ใน tables อื่นๆ เช่น `albums` เราจะเรียกคอลัมน์นี้ว่า Foreign Key หรือตัวย่อ **FK** ลงดูตัวอย่างต่อไปนี้



ER diagram ของ tables: artists, albums และ tracks พิรุณ PK และ FK

จากรูป diagram เราสามารถสรุปได้ว่า

- PK ของ table artists คือ artistid ที่เป็น FK ของ table albums
- PK ของ table albums คือ albumid ที่เป็น FK ของ table tracks
- เส้นความสัมพันธ์สีดำ (relationship) ที่ลาก connect tables ทั้งหมดเข้าด้วยกัน เกิดจาก การเชื่อมกันของ PK และ FK



JOIN คือการเชื่อม tables ต่างๆเข้าด้วยกัน ด้วยเงื่อนไข primary key = foreign key

ลักษณะสำคัญของ FK ที่แตกต่างจาก PK คือ FK สามารถมีค่าซ้ำกันได้แล้ว ตัวอย่างเช่น ศิลปิน หนึ่งคน (unique ใน table artists) สามารถออกอัลบัมได้มากกว่าหนึ่งอัลบัม แปลว่าถ้าเราไปดึง ข้อมูลใน table albums เราจะพบว่า artistid สามารถซ้ำกันได้ (duplicate)

มาลองดูศิลปิน artistid = 50 ใน table artistid ว่าเค้าออกมาแล้วกี่อัลบัม

```

SELECT * FROM albums
WHERE artistid = 50;
    
```



วิธีง่ายที่สุดที่จะดูว่า tables สามารถ JOIN กันได้หรือเปล่า คือดูจาก ER Diagram แต่ในชีวิตจริง หลายบริษัทไม่ได้พัฒนาหรือทำเอกสารเกี่ยวกับ ER Diagram ไว้เลย นี่คือหนึ่งในความท้าทายของการเป็น SQL Data Analyst ที่ต้องเก็บเกี่ยวประสบการณ์ผ่านการทำงานจริง (เพราะไม่มี ER diagram ให้ดู)

	AlbumId	Title	ArtistId
1	35	Garage Inc. (Disc 1)	50
2	148	Black Album	50
3	149	Garage Inc. (Disc 2)	50
4	150	Kill 'Em All	50
5	151	Load	50
6	152	Master Of Puppets	50
7	153	ReLoad	50
8	154	Ride The Lightning	50
9	155	St. Anger	50
10	156	...And Justice For All	50

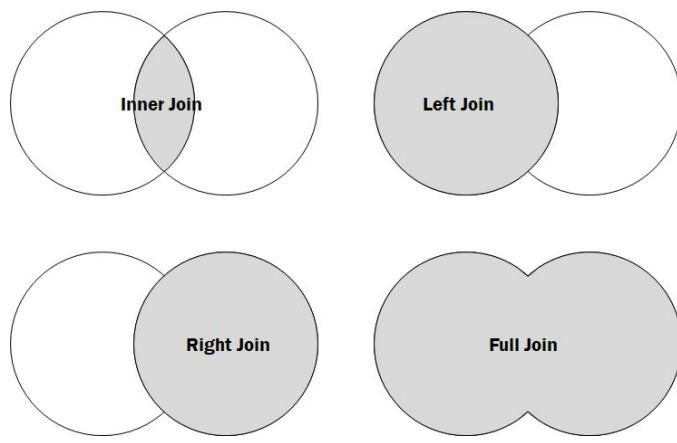
คอลัมน์ artistid สามารถมีค่าซ้ำกันได้ใน table albums

13.2 JOIN คืออะไร

ถ้าทุกคนเข้าใจตอนเข็ปต์เรื่อง PK และ FK ที่เหลือก็ง่ายแล้ว JOIN คือการดึงข้อมูลจากหลาย tables พร้อมกัน (PK=FK) โดยรูปแบบการ JOIN หลักๆที่เราใช้กันจะมีสี่แบบ เรียงตามความถี่/ ความนิยมในการใช้งาน

1. INNER JOIN (ใช้บ่อยที่สุด ประมาณ 80-90% ของการเขียน query เลย)
2. LEFT JOIN (ใช้บ่อยรองลงมาเป็นอันดับสอง จริงๆ % ไม่ต่างกับ INNER เท่าไหร่)
3. RIGHT JOIN (แทบไม่ค่อยได้ใช้ <10%)
4. FULL JOIN (แทบไม่ค่อยได้ใช้ <5%)

Diagram ด้านล่างแสดงตอนเข็ปต์ของ JOIN ทั้งสี่แบบ คำอธิบายอย่างละเอียดจะอยู่ในหัวข้อ JOIN แต่ละหัวข้อต่อจากนี้



รูปแบบของ common joins ทั้งสี่แบบ

13.3 วิธีการเขียน JOIN (syntax)

SQL มี syntax การเขียน JOIN tables ที่เรียบง่าย โดยทั่วไปเราสามารถเขียน JOIN จบได้ในหนึ่งบรรทัดเท่านั้น ทำไม้มันง่ายอย่างนี้! ลองดูตัวอย่าง query ด้านล่าง

```
SELECT  
    t1.column,  
    t2.column  
FROM t1 JOIN t2 ON t1.PK = t2.FK;
```

- t1 และ t2 คือชื่อ tables ที่เรานำมา JOIN กัน
- วิธีการเขียนดึงคอลัมน์ใน SELECT clause เราใช้ syntax: **table.column_name**
- **ON t1.PK = t2.FK** คือเงื่อนไขในการ JOIN สองตารางนี้เข้าด้วยกัน ซึ่งก็คือ PK=FK

ถ้าเขียน join แรกได้ join ต่อๆไปก็จะง่ายขึ้นอีกเยอะเลย เราจะเริ่มกันที่ INNER JOIN รูปแบบการ join ที่ใช้เยอะที่สุดในโลก SQL

 Syntax การเขียนดึงคอลัมน์ใน SELECT clause ใช้เป็น **table.column_name** เพราะว่าเวลาดึงคอลัมน์จากหลายๆ tables มีโอกาสที่ชื่อคอลัมน์จะซ้ำกัน เราจึงจำเป็นต้องเขียน **table.column_name** เพื่อเพิ่มความชัดเจนใน query ของเรา อันนี้คือ good practice ที่ทุกคนควรทำตาม

13.4 INNER JOIN

เราใช้ inner join เพื่อดึงส่วนที่ overlap กันของสอง tables อกกมา ตัวอย่างด้านล่างสมมติว่า คอลัมน์ ID ใน table A คือ primary key และคอลัมน์ ID ใน table B คือ foreign key

Result set จะมีแค่สาม records เท่านั้นที่คอลัมน์ ID ของทั้งสอง tables เชื่อมกันได้ (matched)

Table A		Table B		Result Set		
ID	Customer	ID	Location	ID	Customer	Location
1	John	1	USA	1	John	USA
2	Adam	2	USA	2	Adam	USA
3	Michael	3	United Kingdom	3	Michael	United Kingdom
4	Toshino	8	Japan			
5	Hayato	9	Japan			

ตัวอย่าง result set ของ inner join

มาลองดูตัวอย่าง query จริงๆ กันบ้าง เรากลับมาที่ chinook database ปกติแล้วจะเขียน JOIN clause กับ ON clause คนละบรรทัด จะได้อ่าน query ง่ายๆ สังเกตวิธีการเขียนขึ้นชื่อ table.column_name ใน SELECT clause

ON clause คือบรรทัดที่เราเขียนเงื่อนไข PK=FK ในตัวอย่างนี้คือ artists.artistid = albums.artistid ถ้าใครจำชื่อคอลัมน์ไม่ได้ ให้กลับไปดูหน้า ER diagram อีกครั้งหนึ่ง

```
SELECT
    artists.artistid,
    artists.name,
    albums.title
FROM artists
INNER JOIN albums
ON artists.artistid = albums.artistid;
```

✓ เทคนิคที่เราใช้เวลาเขียน join tables คือการใช้ AS (alias) เข้ามาช่วยตั้งชื่อ tables สนั่นๆ เช่น A B C หรือ t1 t2 t3 เป็นต้น เวลาเขียนดึงคอลัมน์ใน SELECT clause จะได้ไม่ต้องเขียนยาว ลองดูวิธีการใช้ AS ใน query ด้านล่าง ปกติแล้วชอบใช้ A B C เป็น alias ของชื่อตาราง เพราะว่า จำง่ายดี ลองเปรียบเทียบ query นี้กับอันก่อนหน้าที่ไม่ได้ใช้ alias แบบไหนเขียนได้สั้นกว่ากัน?

```
SELECT
    A.artistid,
    A.name AS artistName,
    B.title AS albumName
FROM artists AS A
INNER JOIN albums AS B
ON A.artistid = B.artistid;
```

	ArtistId	artistName	albumName
1	1	AC/DC	For Those About To Rock We Salute ...
2	2	Accept	Balls to the Wall
3	2	Accept	Restless and Wild
4	1	AC/DC	Let There Be Rock
5	3	Aerosmith	Big Ones
6	4	Alanis Morissette	Jagged Little Pill
7	5	Alice In Chains	Facelift
8	6	Antônio Carlos Jobim	Warner 25 Anos
9	7	Apocalyptica	Plays Metallica By Four Cellos
10	8	Audioslave	Audioslave

ผลลัพธ์ที่ได้จาก inner join query

13.5 LEFT JOIN

โอเครมั้ย? ถ้าต้องกี๊ทุกคนเขียน INNER JOIN ได้ เนื้อหาหลังจากนี้ไม่มีอะไรมากแล้ว เพราะว่า syntax ของ join ทุกแบบเขียนเหมือนกันหมดเลย ต่างกันที่ผลลัพธ์หรือ result set ที่เราได้กลับมาจากการ query นั่นๆ

LEFT JOIN จะแตกต่างจาก INNER JOIN นิดเดียว ตรงที่ table ทาง ด้านซ้ายมือจะเป็น table หลักที่เรานำไป join กับ table ทางด้านขวา มือ ตัวอย่างด้านล่างจะเห็นว่า result set ของ LEFT JOIN จะมีโอกาสเกิดค่า NULL ในแถวที่ ID ของทั้งสองตารางไม่สามารถ match กันได้

The diagram illustrates the LEFT JOIN operation. It shows two tables, Table A and Table B, followed by a plus sign (+) indicating addition, and then an arrow pointing to the Result Set. Table A has columns ID and Customer with rows 1 (John), 2 (Adam), 3 (Michael), 4 (Toshino), and 5 (Hayato). Table B has columns ID and Location with rows 1 (USA), 2 (USA), 3 (United Kingdom), 8 (Japan), and 9 (Japan). The Result Set table has columns ID, Customer, and Location, containing all rows from Table A (including rows 4 and 5 with NULL values for Location) and only the matching rows from Table B.

Table A		Table B		Result Set		
ID	Customer	ID	Location	ID	Customer	Location
1	John	1	USA	1	John	USA
2	Adam	2	USA	2	Adam	USA
3	Michael	3	United Kingdom	3	Michael	United Kingdom
4	Toshino	8	Japan	4	Toshino	NULL
5	Hayato	9	Japan	5	Hayato	NULL

ตัวอย่าง result set ของ left join ค่า NULL แสดงใน rows ID = 4 และ 5

Query ของ LEFT JOIN หน้าตาเหมือนกับ INNER JOIN เลย แค่เปลี่ยนบรรทัดที่ห้า จากคำว่า INNER เป็น LEFT เท่านี้ก็เสร็จเรียบร้อย

```
SELECT
  A.artistid,
  A.name AS artistName,
  B.title AS albumName
FROM artists AS A
LEFT JOIN albums AS B
ON A.artistid = B.artistid;
```

	ArtistId	artistName	albumName
49	23	Frank Zappa & Captain Beefheart	Bongo Fury
50	24	Marcos Valle	Chill: Brazil (Disc 1)
51	25	Milton Nascimento & Bebeto	NULL
52	26	Azymuth	NULL
53	27	Gilberto Gil	As Canções de Eu Tu Eles
54	27	Gilberto Gil	Quanta Gente Veio Ver (Live)
55	27	Gilberto Gil	Quanta Gente Veio ver--Bônus De Ca...
56	28	João Gilberto	NULL
57	29	Bebel Gilberto	NULL
58	30	Jorge Vercilio	NULL
59	31	Baby Consuelo	NULL

ผลลัพธ์ที่ได้จากการ left join query



สำหรับ RIGHT JOIN จริงๆไม่ต้องเรียนก็ได้ เพราะว่าแค่เราสลับตำแหน่ง tables ใน LEFT JOIN query (เปลี่ยนลำดับของ A กับ B ใน syntax) ก็เทียบเท่ากับการเขียน RIGHT JOIN แล้ว

13.6 FULL JOIN

FULL JOIN เป็นรูปแบบการ join tables ที่ใช้น้อยที่สุด เพราะว่า result set ของเราจะอ gồmมา ขนาดใหญ่มาก query ที่เขียนด้วย full join จะดึงมาหมดเลยทุก records ทั้ง table ซ้ายและขวา ที่อยู่ใน query ถึงแม้ว่า PK จะไม่สามารถ match กับ FK ได้ก็ตาม (แสดงเป็นค่า NULL)



SQLite ไม่รองรับ FULL JOIN เพราะว่าในชีวิตจริง เราไม่มีโอกาสได้ใช้ join แบบนี้เท่าไหร่ (มีโอกาสใช้น้อย อย่างที่เราอธิบายไป)

Table A

ID	Customer
1	John
2	Adam
3	Michael
4	Toshino
5	Hayato



Table B

ID	Location
1	USA
2	USA
3	United Kingdom
8	Japan
9	Japan

Result Set

ID	Customer	Location
1	John	USA
2	Adam	USA
3	Michael	United Kingdom
4	Toshino	NULL
5	Hayato	NULL
8	NULL	Japan
9	NULL	Japan

ตัวอย่าง result set ของ full join

Syntax ของ full join เขียนเหมือนกับ inner และ left join เลย ตัวอย่าง query นี้แอดเขียน A.* , B.* ใน SELECT clause เพื่อดึงทุก colum ของทั้งสองตารางอกรมาใน result set

```

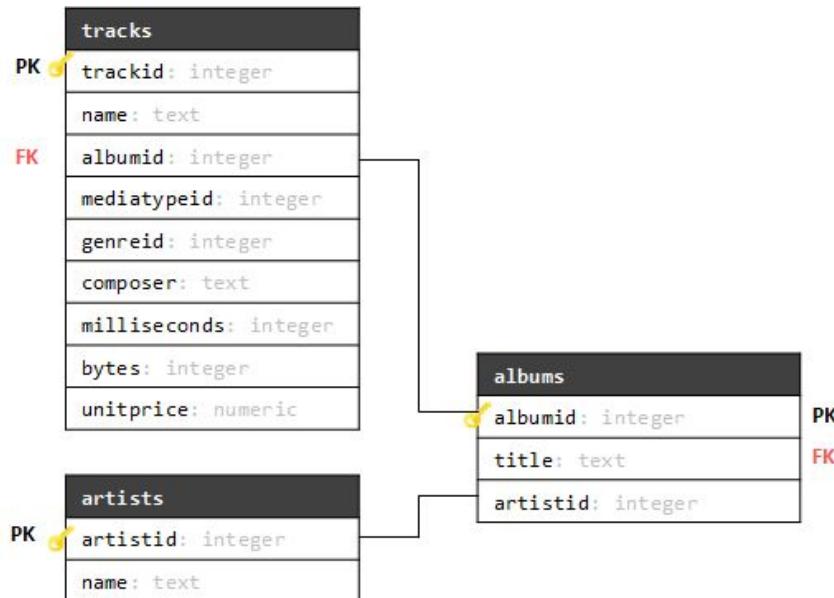
SELECT
  A.*,
  B.*
FROM artists AS A
FULL JOIN albums AS B
ON A.artistid = B.artistid;
  
```

13.7 JOIN more than two tables

อย่างที่เกริ่นไว้ในหัวข้อ 13.3 ถ้าเราเขียน JOIN แรกได้ JOIN ต่อๆไปก็จะง่ายเอง ด้านล่างคือ query ที่เขียน JOIN สามตารางเข้าด้วยกัน: artists, albums, tracks



ยิ่งเขียน join หลายๆ ตาราง เรายิ่งต้องระมัดระวังเรื่องการพิมพ์ชื่อคอลัมน์และชื่อตาราง ทั้งหมด เพราะจะมีโอกาสผิดง่าย วิธีแก้คือต้องฝึกเขียนบ่อยๆ ทำความเข้าใจ database ถ้าได้เขียนทุกวันแล้วมือของเราจะคล่องขึ้น



ER diagram ของ tables: artists, albums, tracks

Query นี้ยาวก็จริง (น่าจะยาวที่สุดตั้งแต่เราอ่านมาในหนังสือเล่มนี้) แต่泯ไม่ได้มีความซับซ้อน อะไรเลย ตอนนี้เขื่อว่าทุกคนสามารถอ่านทีละบรรทัด และเข้าใจ query นี้ทั้งหมดเลย ใช่มั้ย? 😊

```
SELECT
    A.artistid,
    A.name AS artistName,
    B.title AS albumName,
    C.name AS trackName,
    C.bytes,
    C.milliseconds,
    C.unitprice
FROM artists AS A
INNER JOIN albums AS B
ON A.artistid = B.artistid
INNER JOIN tracks AS C
ON B.albumid = C.albumid

WHERE A.artistid IN (99, 105, 220);
```



ข้ออีกครั้ง การเขียน JOIN ก็คือการเขียนเชื่อมตารางด้วย PK = FK เมื่อ join tables ที่ต้องการเสร็จแล้ว เราสามารถพิลเตอร์ result set ด้วย WHERE clause หรือจะใช้ GROUP BY, HAVING, ORDER BY หรือ LIMIT ทุกอย่างที่เราเรียนมาในบทก่อนหน้าได้หมดเลย

13.8 UNION

UNION คือการนำสอง queries มาเรียงต่อกัน (แบบ stacking) โดยทั้งสอง queries ต้องมีจำนวนคอลัมน์เท่ากัน และ data type ของคอลัมน์ต้องเหมือนกันด้วย

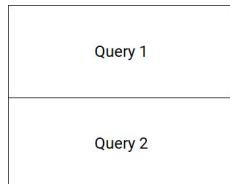


diagram แสดงการผลลัพธ์ของการ UNION

คราวนี้มาลองดูตัวอย่าง query ด้านล่าง แอดดิng ทุกคอลัมน์จาก customers table พิลเตอร์เฉพาะลูกค้าในประเทศ アメリカ และนำไป UNION กับลูกค้าในประเทศ อังกฤษ เสร็จแล้ว ORDER BY คอลัมน์ country ลองดูผลลัพธ์ของ query นี้ด้านล่าง

```
SELECT firstname, lastname, country FROM customers
WHERE country = 'USA'

UNION

SELECT firstname, lastname, country FROM customers
WHERE country = 'United Kingdom'

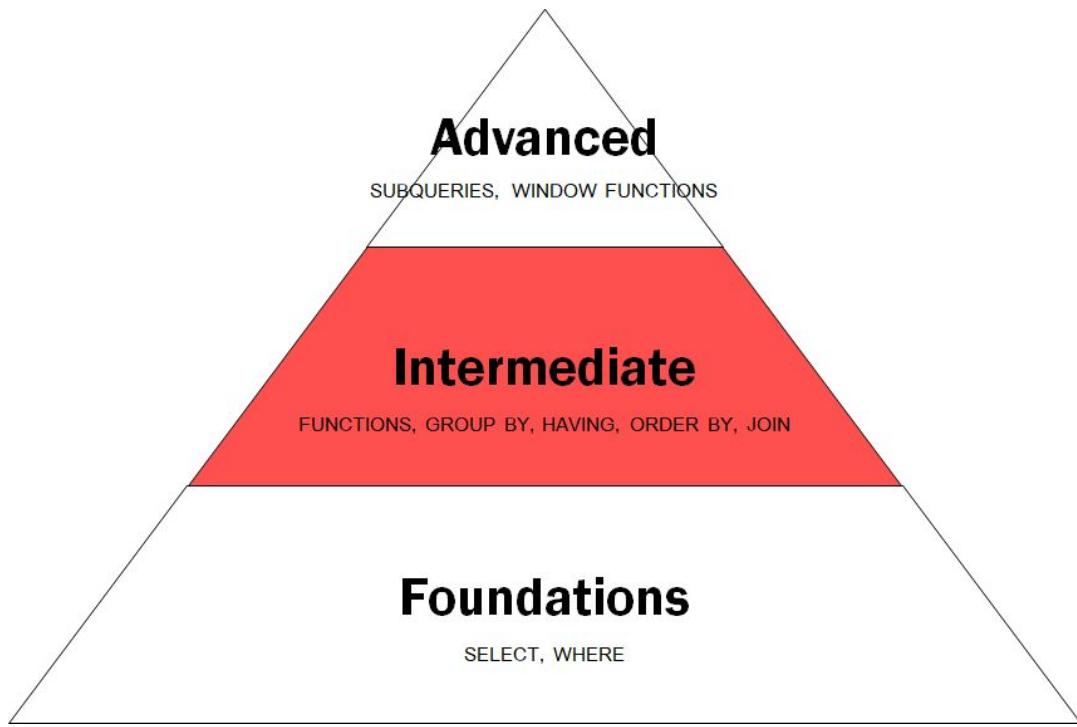
ORDER BY country;
```

	FirstName	LastName	Country
1	Dan	Miller	USA
2	Frank	Harris	USA
3	Frank	Ralston	USA
4	Heather	Leacock	USA
5	Jack	Smith	USA
6	John	Gordon	USA
7	Julia	Barnett	USA
8	Kathy	Chase	USA
9	Michelle	Brooks	USA
10	Patrick	Gray	USA
11	Richard	Cunningham	USA
12	Tim	Goyer	USA
13	Victor	Stevens	USA
14	Emma	Jones	United Kingdom
15	Phil	Hughes	United Kingdom
16	Steve	Murray	United Kingdom

Result set ที่ได้จากการ UNION สอง queries เข้าด้วยกัน

Congratulations!

Achievement II Unlocked



ยินดีด้วย! ตอนนี้ทุกคนเรียนจบ SQL ระดับ Intermediate มาลองรีวิวเร็ๆ ว่าในพาร์ทที่สอง เราสามารถเขียน query อะไรได้แล้วบ้าง

- สรุปผลสถิติ และปรับค่าในคอลัมน์ด้วย aggregate และ value function ตามลำดับ
- สรุปผลสถิติแบ่งตามกลุ่มด้วย GROUP BY พลเตอร์กลุ่มด้วย HAVING เรียงข้อมูลด้วย ORDER BY ทั้งแบบ ascending/ descending order
- สามารถอ่าน ER Diagram ได้อย่างชำนาญ รู้ความแตกต่างของ primary key (PK) และ foreign key (FK)
- ดึงข้อมูลจากหลายๆ tables ได้พร้อมกัน เช้าใจ syntax การเขียน INNER, LEFT และ FULL JOIN

ในพาร์ทที่สามของหนังสือเล่มนี้ เราจะเจาะลึกเทคนิคขั้นสูงอย่าง subqueries, window functions, create table (และ insert, update, delete), create view และอีกมาก!

Chapter 14 - Import CSV File to Database

บทนี้มาลงดูวิธีการ import CSV file เข้าสู่ฐานข้อมูลของเราบ้าง

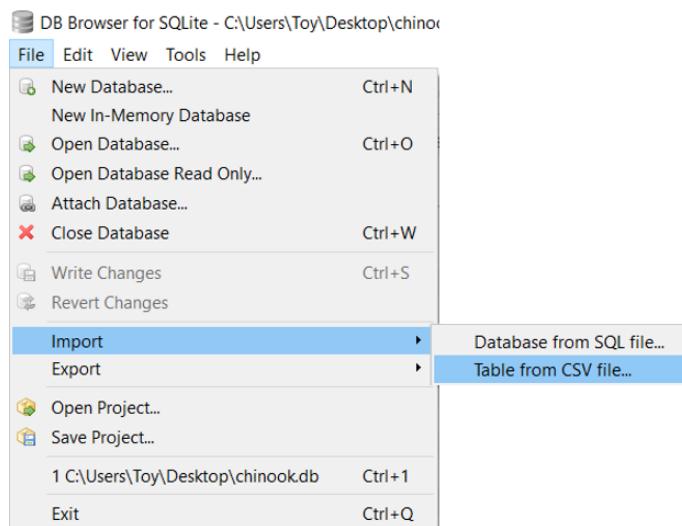
ดาวน์โหลดไฟล์ CSV สำหรับที่ได้[ที่นี่](#) ลองเปิดไฟล์ด้วยโปรแกรม Excel

	A	B	C	D
1	ID	Firstname	Lastname	Country
2	1	Thomas	Niel	USA
3	2	David	Eagle	United Kingdom
4	3	Joseph	Chan	Hong Kong
5	4	John	Kurt	USA
6	5	Anna	Mary	United Kingdom

ตัวอย่างไฟล์ CSV ที่เราจะ import เข้า chinook.db ด้วยโปรแกรม DB Browser

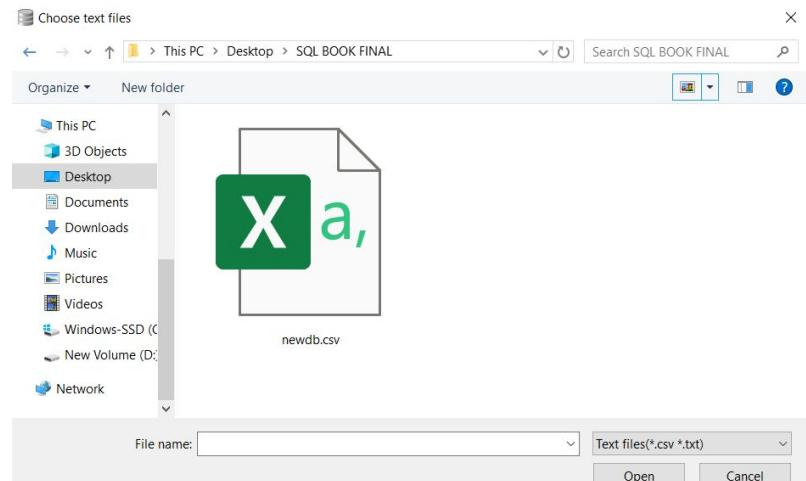
เราสามารถ import ไฟล์นี้เข้าสู่ chinook database ของเราด้วยโปรแกรม DB Browser ง่ายๆ ในสองขั้นตอน

1. ไปที่ File > Import > Table From CSV file



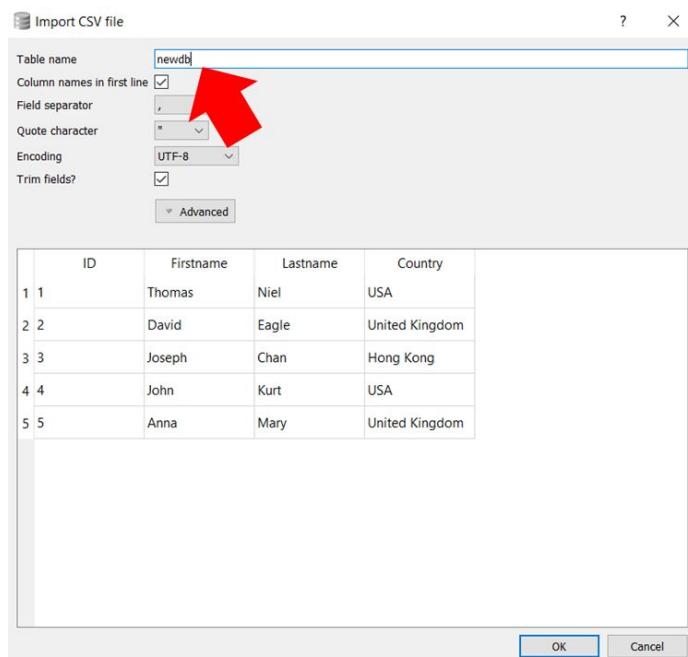
คลิกที่ File > Import > Table from CSV file

2. Browser หาไฟล์ CSV ที่ต้องการ import เข้าสู่ฐานข้อมูล และคลิก Open



เลือกไฟล์ csv ที่ต้องการ

3. DB Browser จะแสดงหน้าต่างใหม่ขึ้นมา เราสามารถตั้งชื่อ table ใหม่ได้ที่ช่อง Table Name เสร็จแล้วคลิก OK ถ้า import เสร็จแล้ว DB Browser จะขึ้นปี๊บอ้อพว่า “Import Completed”



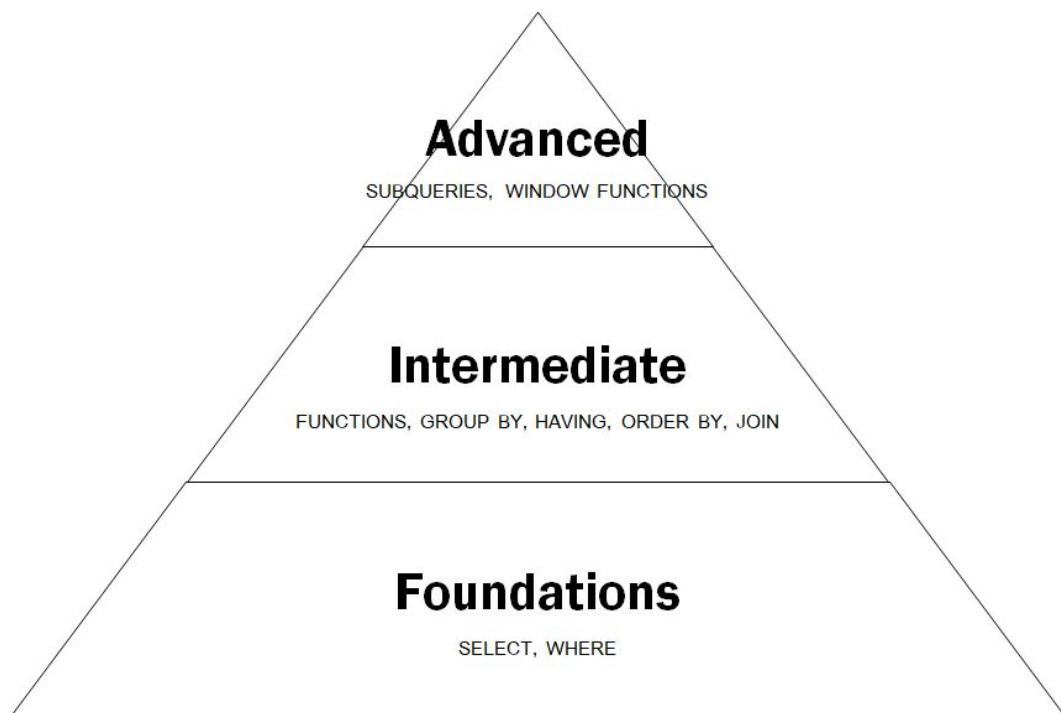
ตั้งชื่อ table ใหม่ที่ช่อง Table name เสร็จแล้วคลิก OK

เราสามารถ query ข้อมูลใน table ใหม่ได้ปกติเลย หรือถ้าไม่ต้องการ table นี้แล้ว พิมพ์ DROP TABLE ตามด้วยชื่อ table ที่ต้องการเอาออก (ลบ) จาก database

```
SELECT * FROM newdb;
DROP TABLE newdb;
```

Coming Real Soon!

หนังสือ **Version 1.2** (20 บท) จะปล่อยอัพเดทวันที่ 27 ธันวาคม 2562
ครอบคลุมเนื้อหาพาร์ท Advanced ทั้งหมด (subqueries/ window functions และอีกมาก!)



สำหรับนักเรียนที่สนใจ สั่งซื้อหนังสือ ebook ฉบับเต็มกับเรา พร้อมรับอัพเดทนิءองหาใหม่ (จนกว่า หนังสือจะสมบูรณ์ นักเรียนและปี คนเขียนและปี) สามารถสั่งจองได้ที่เว็บไซต์ [DataRockie](#)

😊 พิเศษ! หนังสือราคาเล่มละ 290 บาทเท่านั้น ถ้าสั่งจองภายในวันที่ 26 ธันวาคมนี้

👉 ถ้าอ่านแล้วชอบไม่ชอบยังไง มี feedback สามารถส่งมาให้แอดได้สองช่องทาง

- email: toy@datarockie.com
- facebook: <https://m.me/datarockie>

ขอบคุณนักเรียนทุกคนมากๆครับ
แอดทอย / Your Generalist Sensei