# Mini Data Bootcamp
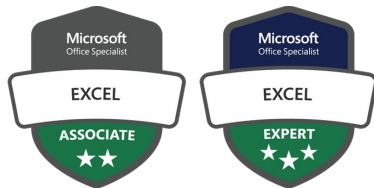## Day Two - SQL Database

# Mini Data Science Bootcamp

3 วิชา 6-7 ชั่วโมง
สำหรับผู้เริ่มต้น อยากเป็น Data Analyst

สอนโดยแอดทอย
คนดีคนเดิม

**Dashboard & Report** — Day 3

**SQL Database** — Day 2

**Microsoft Excel** — Day 1

เกณฑ์การสอบผ่านที่ 80%

Content designed and curated by DataRockie 2023

ทักษะสำคัญของ Data Analyst

**Data Analyst** = f( Spreadsheets, <mark>**Database**</mark>, Dashboard & Report )

**Senior Data Analyst**

KASIKORNBANK

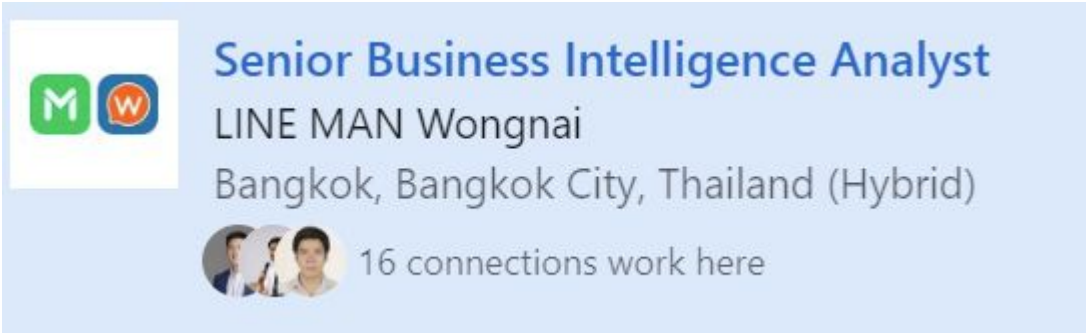Bangkok, Bangkok City, Thailand (Hybrid)

1 company alum works here

Promoted · in Easy Apply

## Qualifications

- Bachelor's degree in computer science, Information Systems, MIS, or a related field is required.
- A minimum of 5 years of experience in data analysis and database management.
- Intermediate proficiency in MS Excel.
- Strong skills in SQL.

งานสาย Data เกือบ 100% ตอนนี้ต้องใช้ **SQL** ในการทำงาน (required skill)

**Senior Business Intelligence Analyst**

LINE MAN Wongnai

Bangkok, Bangkok City, Thailand (Hybrid)

16 connections work here

**What you'll Need:**

- 5+ years of experience in Business Intelligence, Data analyst, or related fields.
- Knowledge of statistics and experience using data analytics tools (PowerBI, Tableau or similar BI tools.)
- Knowledge of and experience with reporting tools, database (SQL), programming (Python)
- Strong analytical and problem-solving skills.

SQL

## Senior Business Analyst

**Razer Inc.**

Bangkok, Bangkok City, Thailand

🟧 Premium tip: you'd be a top applicant

Promoted

## Pre-Requisites

- Bachelor's degree in a relevant field (e.g., Business, Finance, Economics, or Statistics)
- Able to work closely with internal teams and stakeholders to deliver tasks when required.
- Data driven, highly organized.
- Deadline orientated and self-motivated. Ability to think outside of the box.
- Understanding about payments for digital services and content is a plus.
- Ability to work independently and manage multiple priorities.
- Proficiency in SQL and data visualization tools (e.g., Tableau, Power BI)
- More than 2 years prior experience in SQL is preferred.

**SQL**

# Course Outline

# Content

- What is Database?
- What is SQL?
- ER Diagram
- Basic SQL Clauses
- JOIN Tables

# What is Database?

# What is Database?



Database

Customer     Order     Invoice     Product

# Data Analyst Workflow

Database Server

Username Password

Query

Result

Data Analyst

Export

CSV

CSV Data
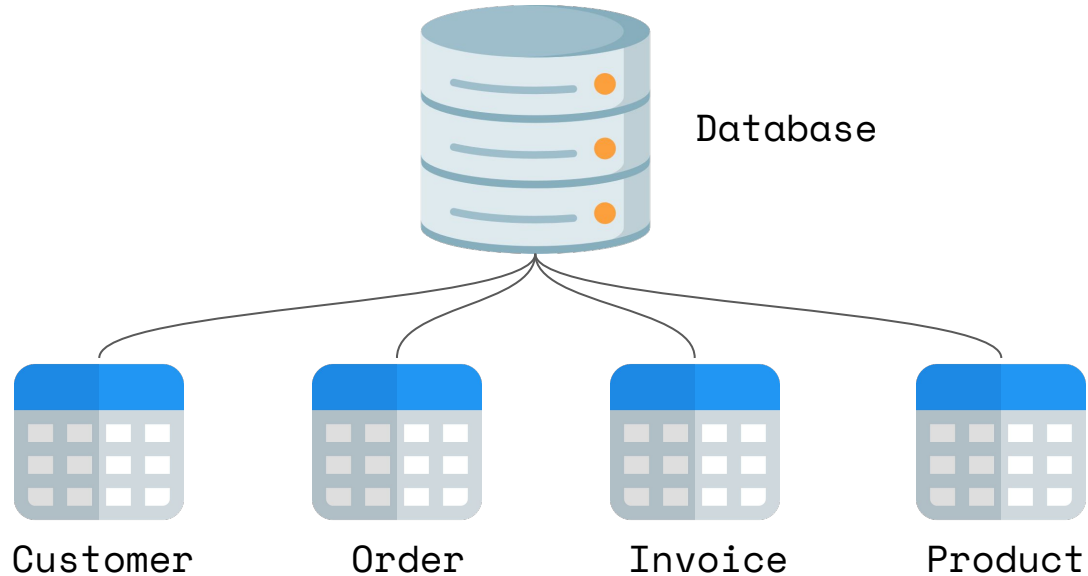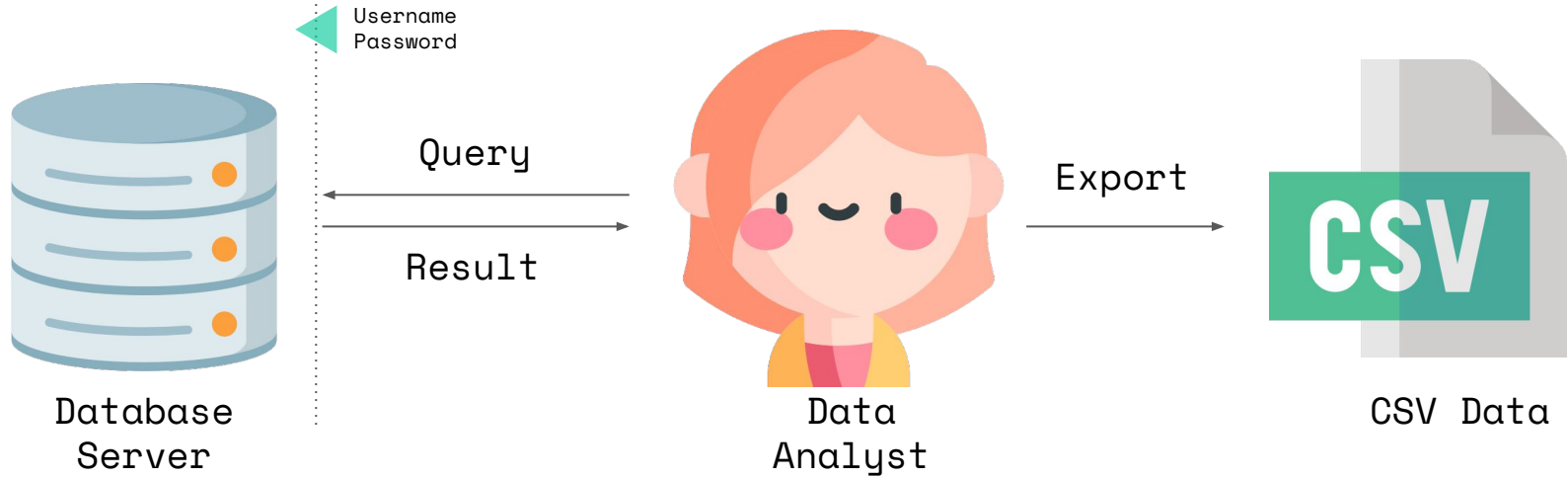
# What is SQL?

# What is SQL

## Structured Query Language

que·ry

/ˈkwirē/

*noun*

a question, especially one addressed to an official or organization.
"a spokeswoman said queries could not be answered until Monday"

Similar: question   inquiry   interrogation   examination   quiz   quizzing   ⌄
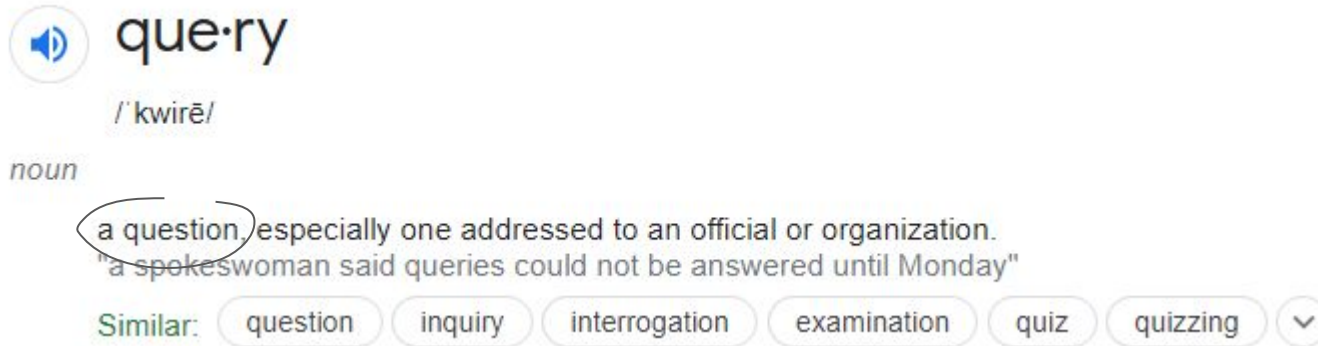
# Top programming languages for data scientists in 2022

Javier Canales Luna • January 21, 2022

in  f  y

In this article, we will look at some of the top data science programming languages for 2022, and present the strengths and capabilities of each of them.

- Python
- R
- SQL  **Top Three** Tools for Data Science
- Java
- Julia
- Scala
- C/C++
- JavaScript
- Swift
- Go
- MATLAB
- SAS

Most Popular Programming Languages for Data Science (KDnuggets Software Poll)

50%

# SQL Flavours

A lot of flavours

| | | |
|---|---|---|
| PostgreSQL | MySQL | Oracle Database |
| SQL Server | Azure | Amazon Redshift |
| SQLite | DB2 | H2 |
| Sybase | Exasol | Apache Derby |
| MariaDB | HyperSQL | Snowflake |
| Cassandra | ClickHouse | Greenplum |
| Apache Hive | Vertica | MongoDB |
| CockroachDB | Couchbase | BigQuery |

# ER Diagram

# ER Diagram

Entity Relationship Diagram

## playlists
🔑 PlaylistId: INTEGER
Name: NVARCHAR(120)

## playlist_track
🔑 PlaylistId: INTEGER
🔑 TrackId: INTEGER

## media_types
🔑 MediaTypeId: INTEGER
Name: NVARCHAR(120)

## genres
🔑 GenreId: INTEGER
Name: NVARCHAR(120)

## tracks
🔑 TrackId: INTEGER
Name: NVARCHAR(200)
AlbumId: INTEGER
MediaTypeId: INTEGER
GenreId: INTEGER
Composer: NVARCHAR(220)
Milliseconds: INTEGER
Bytes: INTEGER
UnitPrice: NUMERIC

## artists
🔑 ArtistId: INTEGER
Name: NVARCHAR(120)

## invoices
🔑 InvoiceId: INTEGER
CustomerId: INTEGER
InvoiceDate: DATETIME
BillingAddress: NVAR...
BillingCity: NVARCHA...
4 more columns...

## invoice_items
🔑 InvoiceItemId: INTEGER
InvoiceId: INTEGER
TrackId: INTEGER
UnitPrice: NUMERIC
Quantity: INTEGER

## albums
🔑 AlbumId: INTEGER
Title: NVARCHAR(160)
ArtistId: INTEGER

## customers
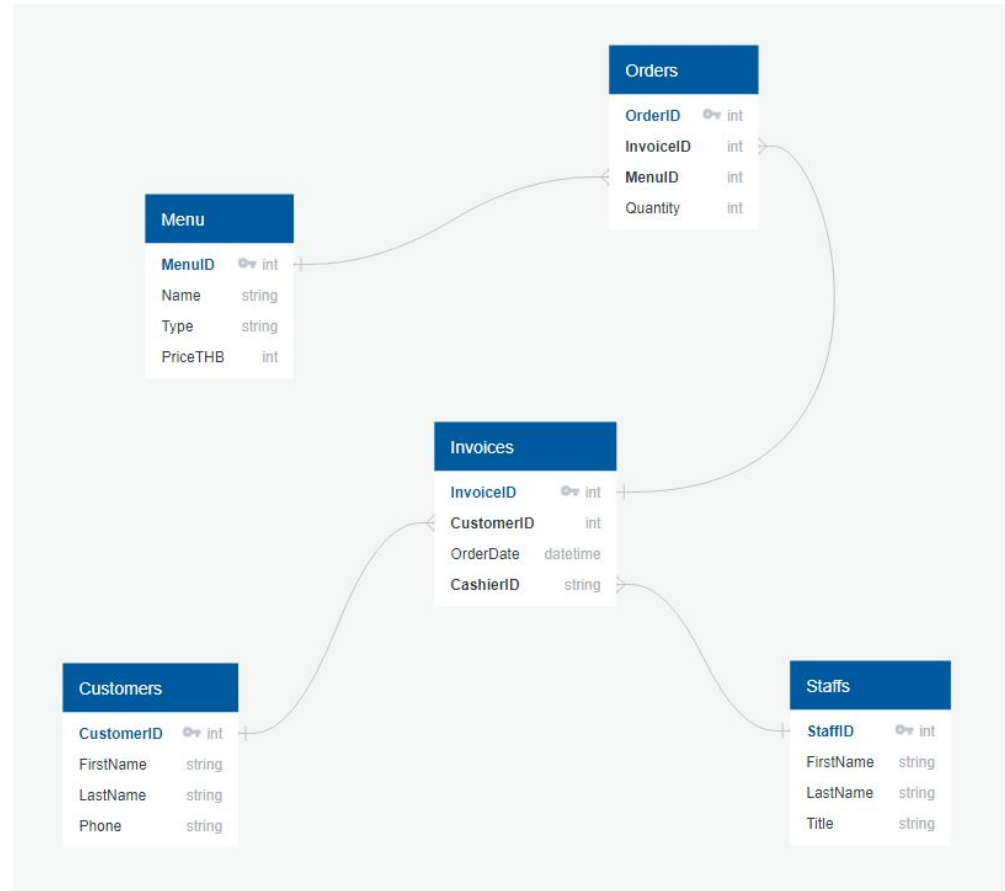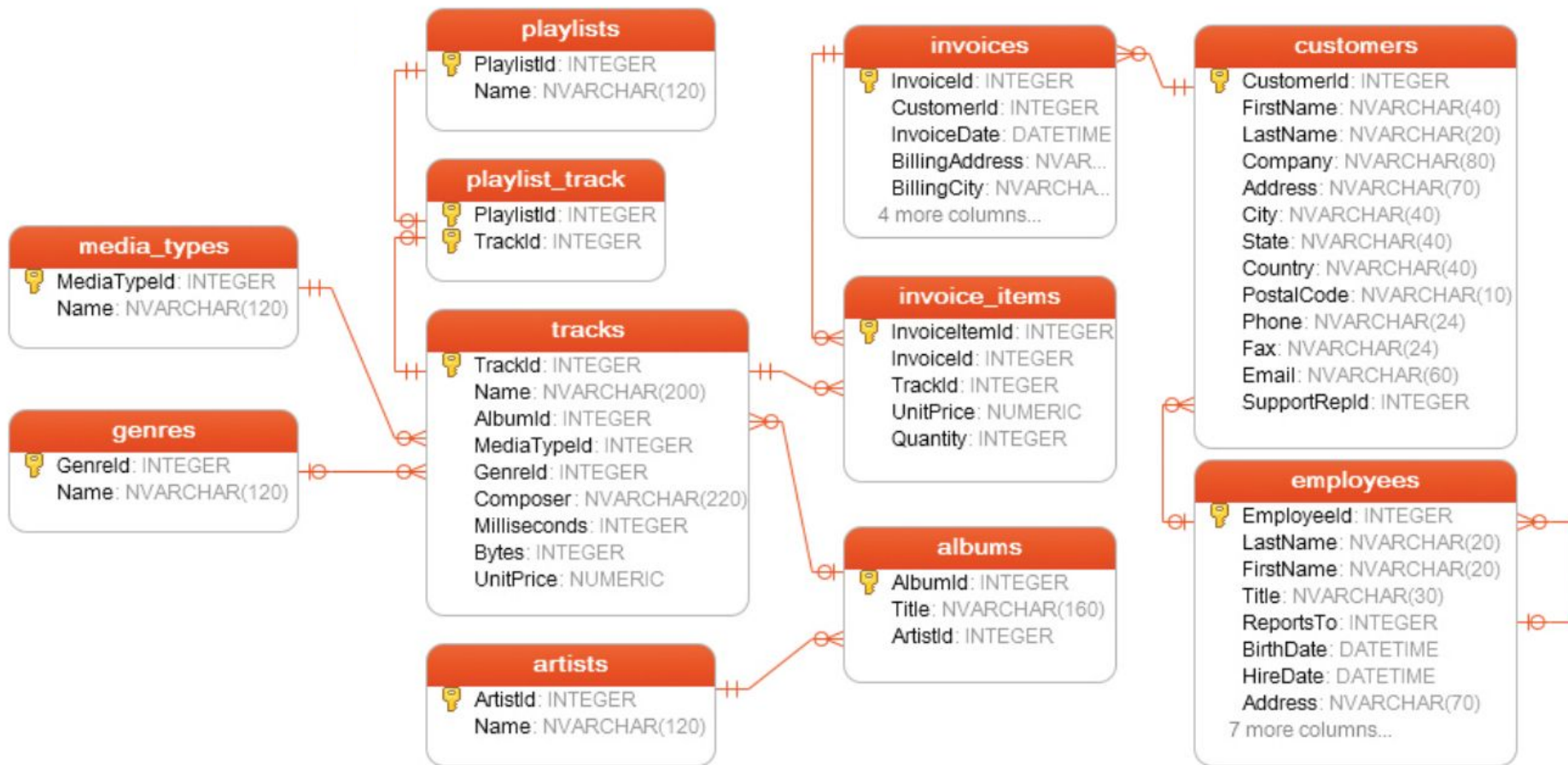🔑 CustomerId: INTEGER
FirstName: NVARCHAR(40)
LastName: NVARCHAR(20)
Company: NVARCHAR(80)
Address: NVARCHAR(70)
City: NVARCHAR(40)
State: NVARCHAR(40)
Country: NVARCHAR(40)
PostalCode: NVARCHAR(10)
Phone: NVARCHAR(24)
Fax: NVARCHAR(24)
Email: NVARCHAR(60)
SupportRepId: INTEGER

## employees
🔑 EmployeeId: INTEGER
LastName: NVARCHAR(20)
FirstName: NVARCHAR(20)
Title: NVARCHAR(30)
ReportsTo: INTEGER
BirthDate: DATETIME
HireDate: DATETIME
Address: NVARCHAR(70)
7 more columns...

**playlists**
🔑 PlaylistId: INTEGER
Name: NVARCHAR(120)

**playlist_track**
🔑 PlaylistId: INTEGER
🔑 TrackId: INTEGER

**media_types**
🔑 MediaTypeId: INTEGER
Name: NVARCHAR(120)

**genres**
🔑 GenreId: INTEGER
Name: NVARCHAR(120)

**tracks**
🔑 TrackId: INTEGER
Name: NVARCHAR(200)
AlbumId: INTEGER
MediaTypeId: INTEGER
GenreId: INTEGER
Composer: NVARCHAR(220)
Milliseconds: INTEGER
Bytes: INTEGER
UnitPrice: NUMERIC

**artists**
🔑 ArtistId: INTEGER
Name: NVARCHAR(120)

**invoices**
🔑 InvoiceId: INTEGER
CustomerId: INTEGER
InvoiceDate: DATETIME
BillingAddress: NVAR...
BillingCity: NVARCHA...
4 more columns...

**invoice_items**
🔑 InvoiceItemId: INTEGER
InvoiceId: INTEGER
Tra...
Un...
Quantity: INTEGER

**albums**
🔑 AlbumId: INTEGER
Title: NVARCHAR(160)
ArtistId: INTEGER

`Primary Key (PK)`

**customers**
🔑 CustomerId: INTEGER
FirstName: NVARCHAR(40)
LastName: NVARCHAR(20)
Company: NVARCHAR(80)
Address: NVARCHAR(70)
City: NVARCHAR(40)
State: NVARCHAR(40)
Country: NVARCHAR(40)
PostalCode: NVARCHAR(10)
Phone: NVARCHAR(24)
Fax: NVARCHAR(24)
Email: NVARCHAR(60)
RepId: INTEGER

**employees**
🔑 EmployeeId: INTEGER
LastName: NVARCHAR(20)
FirstName: NVARCHAR(20)
Title: NVARCHAR(30)
ReportsTo: INTEGER
BirthDate: DATETIME
HireDate: DATETIME
Address: NVARCHAR(70)
7 more columns...

**playlists**
- 🔑 PlaylistId: INTEGER
- Name: NVARCHAR(120)

**playlist_track**
- 🔑 PlaylistId: INTEGER
- 🔑 TrackId: INTEGER

**invoices**
- 🔑 InvoiceId: INTEGER
- CustomerId: INTEGER
- InvoiceDate: DATETIME
- BillingAddress: NVAR...
- BillingCity: NVARCHA...

**customers**
- 🔑 CustomerId: INTEGER
- FirstName: NVARCHAR(40)
- LastName: NVARCHAR(20)
- Company: NVARCHAR(80)
- Address: NVARCHAR(70)
- City: NVARCHAR(40)
- State: NVARCHAR(40)
- Country: NVARCHAR(40)
- PostalCode: NVARCHAR(10)
- Phone: NVARCHAR(24)
- Fax: NVARCHAR(24)
- Email: NVARCHAR(60)
- SupportRepId: INTEGER

**media_types**
- 🔑 MediaTypeId: INTEGER
- Name: NVARCHAR(120)

**Foreign Key (FK)**

**tracks**
- 🔑 TrackId: INTEGER
- Name: NVARCHAR(200)
- AlbumId: INTEGER
- MediaTypeId: INTEGER
- GenreId: INTEGER
- Composer: NVARCHAR(220)
- Milliseconds: INTEGER
- Bytes: INTEGER
- UnitPrice: NUMERIC

**invoice_items**
- 🔑 InvoiceItemId: INTEGER
- InvoiceId: INTEGER
- TrackId: INTEGER
- UnitPrice: NUMERIC
- Quantity: INTEGER

**genres**
- 🔑 GenreId: INTEGER
- Name: NVARCHAR(120)

**albums**
- 🔑 AlbumId: INTEGER
- Title: NVARCHAR(160)
- ArtistId: INTEGER

**employees**
- 🔑 EmployeeId: INTEGER
- LastName: NVARCHAR(20)
- FirstName: NVARCHAR(20)
- Title: NVARCHAR(30)
- ReportsTo: INTEGER
- BirthDate: DATETIME
- HireDate: DATETIME
- Address: NVARCHAR(70)
- 7 more columns...

**artists**
- 🔑 ArtistId: INTEGER
- Name: NVARCHAR(120)

# ERD Workshop

# DBML

```
1    // Use DBML to define your database structure
2    // Docs: https://dbml.dbdiagram.io/docs
3
4    Table food_orders {
5      order_id int
6      order_date date
7      rider_id int
8      shop_id int
9      value float
10   }
11
12   Table riders {
13     rider_id int
14     name text
15     age int
16     address text
17   }
18
19   Table shops {
20     shop_id int
21     name text
22   }
23
24   Ref: food_orders.rider_id > riders.rider_id
25
26   Ref: shops.shop_id < food_orders.shop_id
```

https://dbdiagram.io/d/6501902302bd1c4a5e7ccf4a
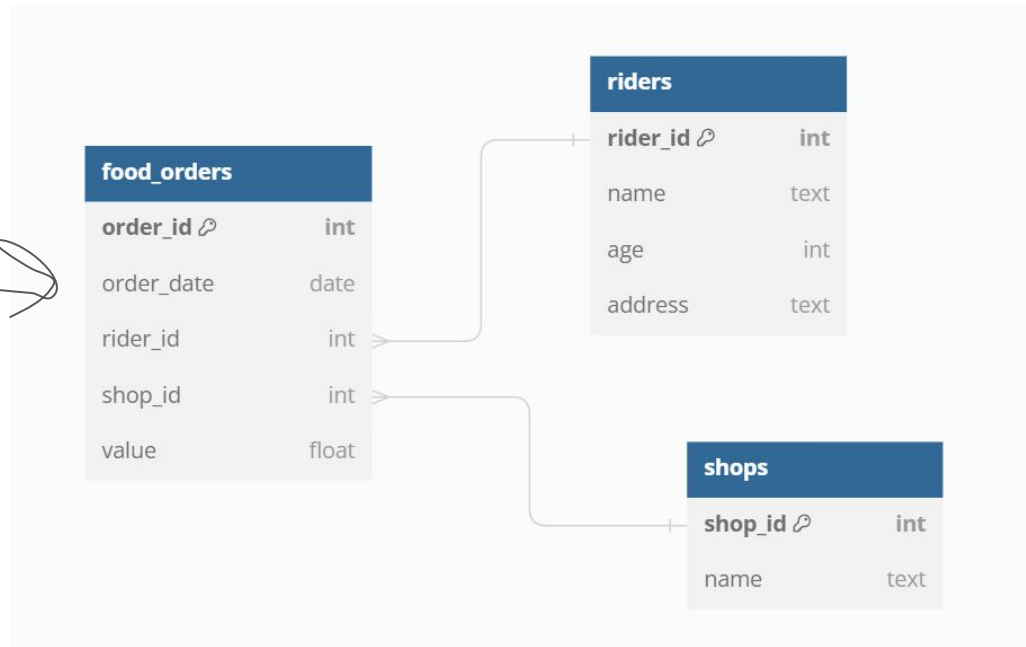
```
Table food_orders {
  order_id int
  order_date date
  rider_id int
  shop_id int
  value float
}
```

| food_orders | |
|---|---|
| order_id 🔑 | int |
| order_date | date |
| rider_id | int |
| shop_id | int |
| value | float |

```
Table food_orders {
  order_id int [primary key]
  order_date date
  rider_id int
  shop_id int
  value float
}

Table riders {
  rider_id int [primary key]
  name text
  age int
  address text
}

Table shops {
  shop_id int [primary key]
  name text
}
```

**food_orders**

| order_id 🔑 | int |
|---|---|
| order_date | date |
| rider_id | int |
| shop_id | int |
| value | float |

**riders**

| rider_id 🔑 | int |
|---|---|
| name | text |
| age | int |
| address | text |

**shops**

| shop_id 🔑 | int |
|---|---|
| name | text |

< แปลว่า **one to many**

```
Ref: riders.rider_id < food_orders.rider_id

Ref: shops.shop_id < food_orders.shop_id
```

**Rider** หนึ่งคน รับได้
มากกว่าหนึ่ง **Order**

**riders**

| | |
|---|---|
| rider_id 🔑 | int |
| name | text |
| age | int |
| address | text |

**food_orders**

| | |
|---|---|
| order_id 🔑 | int |
| order_date | date |
| rider_id | int |
| shop_id | int |
| value | float |

**shops**

| | |
|---|---|
| shop_id 🔑 | int |
| name | text |
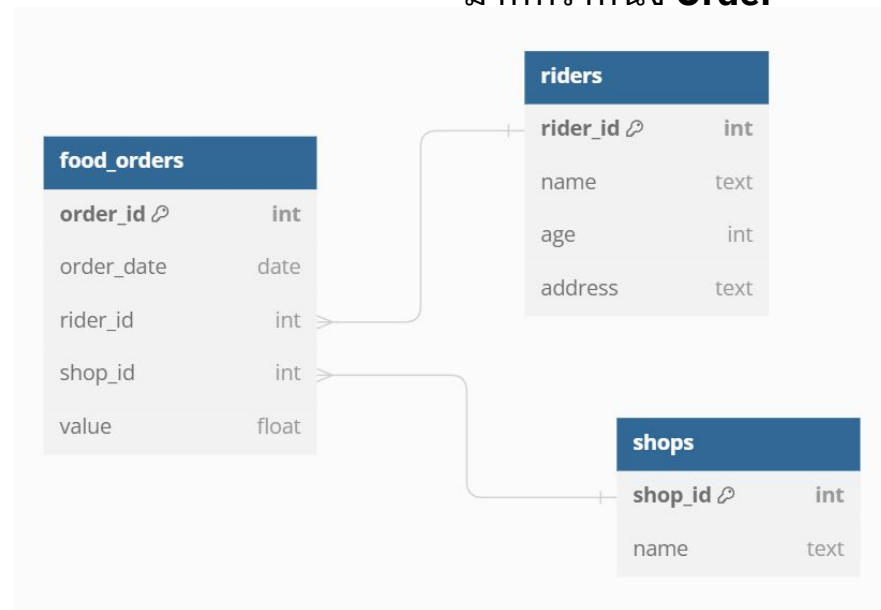
**Shop** ร้านอาหารหนึ่งร้าน รับ
ได้มากกว่าหนึ่ง **Order** เช่นกัน

# Basic SQL for Data Analysts

# Basic SQL Clauses

- ❏ SELECT
- ❏ FROM
- ❏ WHERE
- ❏ AGGREGATE FUNCTIONS()
- ❏ GROUP BY
- ❏ ORDER BY

# Simple Data Manipulation

Select Columns

| ID | Name | City | Email | Spending |
|---|---|---|---|---|
| 1001 | Toy | BKK | toy@mail.com | 1500 |
| 1002 | Anna | LON | anna@mail.com | 1200 |
| 1003 | Marry | LON | marry@mail.com | 900 |
| 1004 | Ken | JPN | ken@mail.com | 2500 |

# Simple Data Manipulation

Filter rows

| ID | Name | City | Email | Spending |
|---|---|---|---|---|
| 1001 | Toy | BKK | toy@mail.com | 1500 |
| 1002 | Anna | LON | anna@mail.com | 1200 |
| 1003 | Marry | LON | marry@mail.com | 900 |
| 1004 | Ken | JPN | ken@mail.com | 2500 |

# Simple Data Manipulation

Select columns + Filter rows

| ID | Name | City | Email | Spending |
|---|---|---|---|---|
| 1001 | Toy | BKK | toy@mail.com | 1500 |
| 1002 | Anna | LON | anna@mail.com | 1200 |
| 1003 | Marry | LON | marry@mail.com | 900 |
| 1004 | Ken | JPN | ken@mail.com | 2500 |

**Select columns:** id name and city
**Filter:** customers in London

# Your First Queries

# SELECT

```
SELECT * FROM customers;
```

# Excel vs. SQL



**=customers**

| | A |
|---|---|
| 1 | =cust |
| | customers |
| 2 | |

ถ้าเราตั้งชื่อไว้

```
SELECT *
FROM customers;
```

# SELECT

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

# Rename Column using AS

```
SELECT
    firstname AS fname,
    lastname AS lname,
    country,
    email
FROM customers;
```

# WHERE (filter rows)

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

เงื่อนไขที่ใช้ฟิลเตอร์ table

# Excel vs. SQL

**=FILTER**(
  customers,
  country = "USA"
)

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

# Aggregate Functions

```sql
SELECT
    AVG(total),
    SUM(total),
    MIN(total),
    MAX(total),
    COUNT(total)
FROM invoices;
```

# Excel vs. SQL

**=AVERAGE**(total)
**=SUM**(total)
**=MIN**(total)
**=MAX**(total)
**=COUNT**(total)

XLS

SQL

**SELECT**
    **AVG**(total),
    **SUM**(total),
    **MIN**(total),
    **MAX**(total),
    **COUNT**(total)
**FROM** invoices;

# AGG + GROUP BY

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

**Tips:** columns in group by should be presented in SELECT too

# ORDER BY

```
SELECT *
FROM customers
ORDER BY country DESC;
```

เรียงข้อมูลจาก**มากไปน้อย**
(descending)

# JOIN Multiple Tables

# What is JOIN?

## Getting data from multiple tables



JOIN
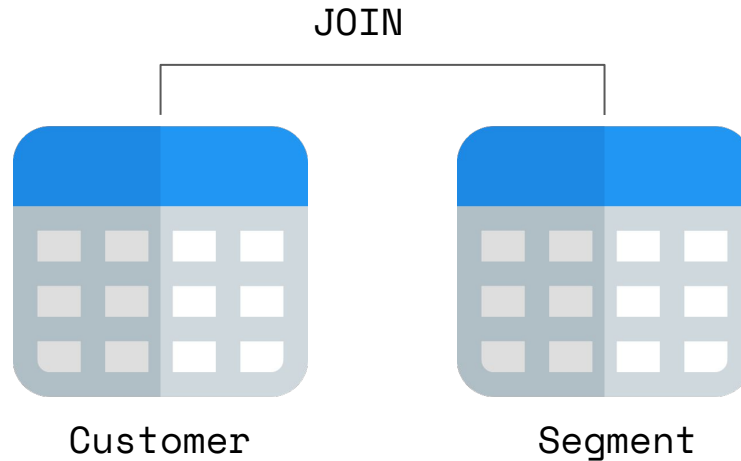
Customer          Segment

# SQL JOIN = VLOOKUP() 📊

Customer

| ID | Name | City |
|------|------|------|
| 1001 | Toy | BKK |
| 1002 | Anna | LON |
| 1003 | Marry | LON |

Segment

| ID | SegName | Cust_ID |
|----|---------------|---------|
| 1 | Deal Hunter | 1001 |
| 2 | Price Sensitive | 1002 |
| 3 | Premium | 1003 |

Join PK=FK

# Result Set

Cool!

| ID | Name | City | SegName |
|------|-------|------|-----------------|
| 1001 | Toy | BKK | Deal Hunter |
| 1002 | Anna | LON | Price Sensitive |
| 1003 | Marry | LON | Premium |

ง่ายจนงง!

# JOIN Example

```
SELECT * FROM table1
JOIN table2
ON table1.pk = table2.fk;
```

Primary Key

Foreign Key

# JOIN syntax

```
SELECT * FROM customer
JOIN segment
ON customer.id = segment.cust_id;
```

# Four Join Types

- ❏ INNER JOIN
- ❏ LEFT JOIN
- ❏ RIGHT JOIN
- ❏ FULL JOIN
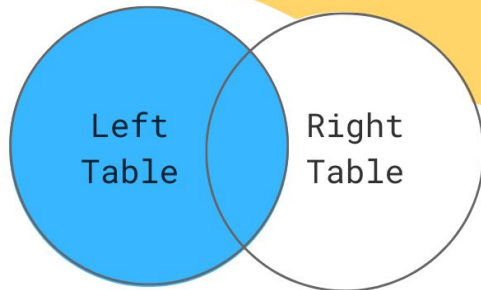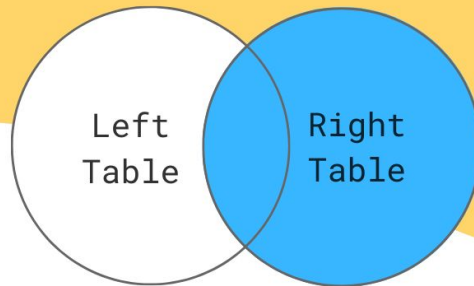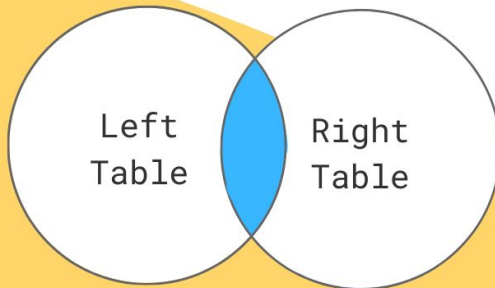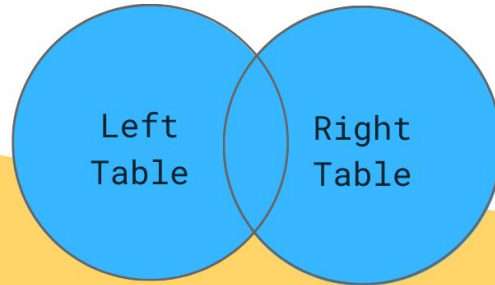
# INNER JOIN (default)
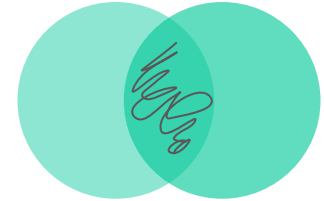
Customer

| ID | Name | City |
|------|-------|------|
| 1001 | Toy | BKK |
| 1002 | Anna | LON |
| 1003 | Marry | LON |
| 1004 | Ken | JPN |

Segment

| ID | SegName | Cust_ID |
|----|------------------|---------|
| 1 | Deal Hunter | 1001 |
| 2 | Price Sensitive | 1002 |
| 3 | Premium | 1003 |

Join PK=FK

# Result Set

Only Matched Rows Return

| ID | Name | City | ID | SegName | Cust_ID |
|----|------|------|-----|---------|---------|
| 1001 | Toy | BKK | 1 | Deal Hunter | 1001 |
| 1002 | Anna | LON | 2 | Price Sensitive | 1002 |
| 1003 | Marry | LON | 3 | Premium | 1003 |

# INNER JOIN

```
SELECT * FROM customer
INNER JOIN segment
ON customer.id = segment.cust_id;
```

# LEFT JOIN

ตารางซ้ายมือ
เป็นตัวตั้ง
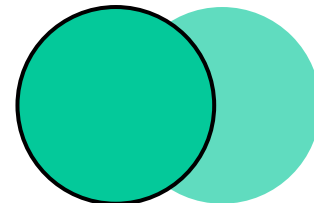
Customer

| ID | Name | City |
|------|-------|------|
| 1001 | Toy | BKK |
| 1002 | Anna | LON |
| 1003 | Marry | LON |
| 1004 | Ken | JPN |

Segment

| ID | SegName | Cust_ID |
|----|----------------|---------|
| 1 | Deal Hunter | 1001 |
| 2 | Price Sensitive | 1002 |
| 3 | Premium | 1003 |

Join PK=FK

# Result Set

All rows in left table will be in the result set

| ID | Name | City | ID | SegName | Cust_ID |
|------|------|------|------|-----------------|---------|
| 1001 | Toy | BKK | 1 | Deal Hunter | 1001 |
| 1002 | Anna | LON | 2 | Price Sensitive | 1002 |
| 1003 | Marry | LON | 3 | Premium | 1003 |
| 1004 | Ken | JPN | NULL | NULL | NULL |

# LEFT JOIN

```sql
SELECT * FROM customer
LEFT JOIN segment
ON customer.id = segment.cust_id;
```

# Key Takeaway

# สรุปการเรียนวันนี้ 🎉

- SQL เป็นภาษาทางการที่เราใช้ทำงานกับ database มา 50 ปีแล้ว
- เป็นภาษาอังกฤษ เขียนง่าย ใช้งานสะดวก
- SELECT, FROM, WHERE, JOIN
- ถ้าฝึกเขียน SQL เก่งๆ หางาน Data Analyst ได้สบายเลย เพราะทุกบริษัทต้องการคนที่มีทักษะนี้

พรุ่งนี้ Day 3 มาเรียน
**Dashboard & Reporting**
ด้วยกันนะคร้าบ

# Let's Connect🔗



**Kasidis Satangmongkol**
Digital Analytics Manager
Bangkok City, Thailand · Contact info
https://datarockie.com
9,055 followers · 500+ connections

adapter digital co., ltd.
The University of Reading

https://www.linkedin.com/in/kasidistoy/

## Your Feedback is A Gift 🎁

สำหรับคลาส Microsoft Excel วันแรก เรียนเป็นยังไงบ้าง แชร์ประสบการณ์เรียนให้แอดหน่อยนะครับทุกคน ชอบอะไรบ้าง เนื้อหาส่วนไหนที่มีประโยชน์ อยากรู้เรื่องอะไรเพิ่มเติมอีก บอกแอดได้เลยนะคร๊าบ

แอดอ่านเองทุก comment เลย และจะนำไปใช้ปรับปรุงเนื้อหา และการสอนครั้งต่อๆไปด้วยครับ (หรืออาจจะมีนัดสอนเพิ่ม ยิ้งงง 555+)

ขอบคุณทุกคนสำหรับ Day One พักผ่อนกันด้วยนะครับ

**Complete and Continue ›**

**Discussion**

Post a comment

*Instructor*

แชร์ **feedback** การเรียนวันแรกให้แอดหน่อยน๊า

**Post Comment**

# Mini Data Bootcamp
## Day Two - SQL Database