



# SQL & Data Collection

<https://s3-us-west-2.amazonaws.com/secure.notion-static.com/755b853d-487f-4c10-bb09-194422d6e9c1/R2DE2PrecourseSQLV2.pdf>

<https://s3-us-west-2.amazonaws.com/secure.notion-static.com/c727e64f-55e7-43ca-ae2b-d55f1959732f/R2DE2CH1DataPipeline.pdf>

ໃຫ້ງານນີ້ web <http://sqlfiddle.com/>

Upload Twá start\_r2de.sql

start\_r2de.sql

intro\_to\_sql.sql

```

1 - INSERT TABLE
2 CREATE TABLE `students` (
3   `student_id` int(10) unsigned NOT NULL AUTO_INCREMENT,
4   `name` varchar(255) NOT NULL,
5   `birthday` date NOT NULL,
6   `weight` int(3) NOT NULL,
7   `height` int(3) NOT NULL,
8   PRIMARY KEY (`student_id`)
9 );
10
11
12 CREATE TABLE `scores` (
13   `score_id` int(10) unsigned NOT NULL AUTO_INCREMENT,
14   `student_id` int(10) unsigned NOT NULL,
15   `subject` varchar(255) NOT NULL,
16   `score` int(2) NOT NULL,
17   PRIMARY KEY (`score_id`)
18 );
19
20 - INSERT DATA FOR STUDENTS TABLE
21 INSERT INTO `students` (
22   `name`, `birthday`, `weight`, `height`
23 ) VALUES ('David', '1990-02-01', 75, 175);

```

Build Schema | Edit Fullscreen | Browser | 1:1 | Run SQL | Edit Fullscreen | 1:1 | Query Panel  
Use this panel to try to solve the problem with other SQL statements (SELECTs, etc.). Results will be displayed below. Share your queries by copying and pasting the URL that is generated after each run.

✓ Schema Ready

Table ประกอบไปด้วย column และ row

## Table ประกอบไปด้วย Column และ Row



Table = ตารางเก็บข้อมูล

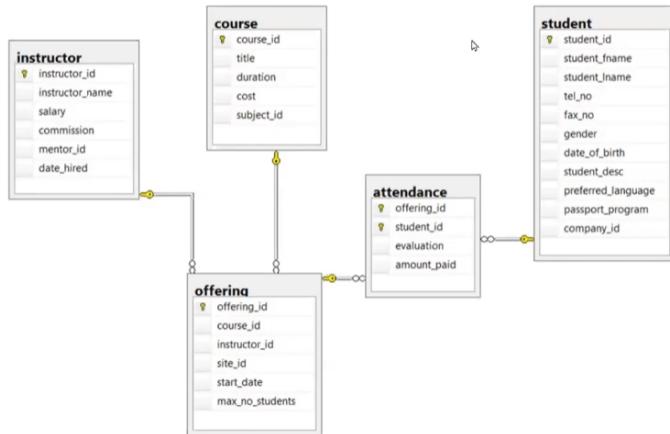
student_id	name	birthday	weight	height
1	David	2/1/1990	75	175
2	John	4/30/1989	67	169
3	Mary	6/22/1993	58	171
4	Jane	8/10/1990	60	153

ตาราง แบ่งออกเป็นหลาย คอลัมน์ (column) และ แถว (row)





# ใน 1 ฐานข้อมูล สามารถมีได้หลาย Table



ตัวอย่าง Database ข้อมูล  
คอร์สเรียน

ขอบคุณรูปจาก Database Design - 2nd Edition by Adrienne Watt



Table ที่จะเรียนกับเป็น student



## Table 1: students

student_id	name	birthday	weight	height
1	David	2/1/1990	75	175
2	John	4/30/1989	67	169
3	Mary	6/22/1993	58	171
4	Jane	8/10/1990	60	153

ตารางเก็บข้อมูลนักเรียนแต่ละคน

มี 5 คอลัมน์:

student_id	ID ของนักเรียน
name	ชื่อนักเรียน
birthday	วันเกิด (DD/MM/YYYY)
weight	น้ำหนัก (kg)
height	ส่วนสูง (cm)



ตัวอย่างที่ 1 ดึงข้อมูล birthday student

```
SELECT name, birthday
```

```
FROM students;
```

ตัวอย่างที่ 2 ดึงนักเรียนกั้งหนดที่ weight > 60

ใช้ WHERE

```
SELECT *
FROM students
WHERE weight > 60;
```

ตัวอย่างที่ 3 ดึงนักเรียนกั้งหนดที่ weight > 60 และสูงมากกว่า 170

ใช้ AND

```
SELECT *
FROM students
WHERE weight > 60 AND height > 170;
```

ตัวอย่างที่ 4 ต้องการทราบว่านักเรียนแต่ละคนได้กี่คะแนน

ใช้ inner join เข้าช่วย

student_id	name	subject	score
1	David	Maths	70
1	David	Computer	85
1	David	Science	74
2	John	Maths	68
2	John	Computer	72
2	John	Science	83
3	Mary	Maths	56
3	Mary	Computer	92
3	Mary	Science	47

```
SELECT a.student_id, name, subject, score
FROM students a INNER JOIN scores b
ON a.student_id = b.student_id;
```

## การเพิ่มข้อมูล



## SQL: เพิ่มข้อมูลกีฬา

student_id	name	birthday	weight	height
1	David	2/1/1990	75	175
2	John	4/30/1989	67	169
3	Mary	6/22/1993	58	171
4	Jane	8/10/1990	60	153

+

5	Sarah	4/18/1991	70	170
---	-------	-----------	----	-----

```
INSERT INTO students  
VALUES (NULL, 'Sarah', '1991-04-18', 70, 170);
```

หากได้ข้อมูลมาเป็นไฟล์ CSV สามารถ Upload ข้อมูลโดยใช้ code



## SQL: เพิ่มข้อมูลจำนวนมาก

Database และ Data Warehouse สามารถโหลดข้อมูลจำนวนมากจากไฟล์ประเภทต่าง ๆ เช่น CSV, Parquet

ตัวอย่างคำสั่ง (แตกต่างกันไปตาม Database ที่ใช้):

```
LOAD DATA LOCAL INPATH './data.csv' OVERWRITE INTO TABLE students;
```

ไฟล์ CSV = ตาราง Excel (ในโปรแกรม Excel เราสามารถเชฟไฟล์ CSV ได้เลย)

Excel					CSV
1	A	B	C	D	E
2	student_id	name	birthday	weight	height
3	7	Peter	2010-06-30	50	160
4	8	Ali	2005-08-18	52	162
5	9	Sarah	2002-02-12	57	174
	10	Emma	2006-04-27	46	158

```
LOAD DATA LOCAL INPATH './data.csv' OVERWRITE INTO TABLE stud
```

# Update ข้อมูล



## SQL: UPDATE ข้อมูล

```
UPDATE students
SET name = "Tony"
WHERE student_id = 5;
```

สามารถอัพเดทที่ลับเฉพาะ หรือมากกว่านั้นก็ได้  
(เราไม่นิยม UPDATE ใน Data Warehouse เพราะใช้เวลาเยอะ)

```
UPDATE students
SET name = "Tony"
WHERE student_id = 5;
```

# DELETE ข้อมูล



## SQL: DELETE ข้อมูล

```
DELETE FROM students
WHERE student_id = 5;
```

สามารถลบที่ลับเฉพาะ หรือมากกว่านั้นก็ได้



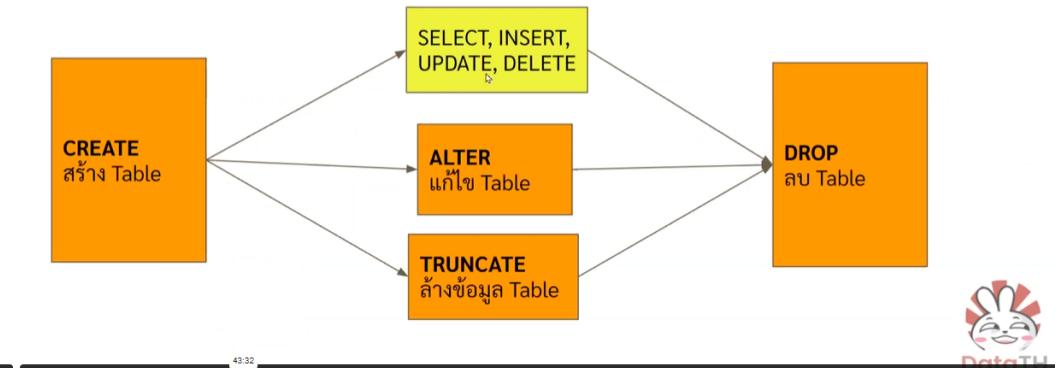
```
DELETE FROM students  
WHERE student_id = 5;
```

หรือ

```
DELETE FROM students  
WHERE student_id > 4;  
--ลบข้อมูลตั้งแต่ 4 ขึ้นไป--
```

## สร้าง แก้ไข ลบ ล้าง Table

Lifecycle (วงจรชีวิต) ของ Table



## Create Table

```
CREATE TABLE `students2` (  
    `student_id` int NOT NULL AUTO_INCREMENT,  
    `name` varchar(255) NOT NULL,  
    `birthday` date NOT NULL,  
    `weight` int NOT NULL,  
    `height` int NOT NULL,  
    PRIMARY KEY (`student_id`)  
) ;
```



NOT NULL AUTO\_INCREMENT = ไม่ให้มีค่าว่าง

AUTO\_INCREMENT = บรรทัดนี้สร้างคอลัมน์ซึ่ง

`student_id` ที่เป็นประเภท integer และไม่สามารถเป็นค่าว่างได้ ข้อบังคับ

`AUTO_INCREMENT` ช่วยให้มั่นใจได้ว่าคอลัมน์จะถูกเพิ่มค่าโดยอัตโนมัติเมื่อเพิ่มแถวใหม่ลงในตาราง

```
92 -- CREATE TABLE
93 CREATE TABLE `students2` (
94     `student_id` int NOT NULL AUTO_INCREMENT,
95     `name` varchar(255) NOT NULL,
96     `birthday` date NOT NULL,
97     `weight` int NOT NULL,
98     `height` int NOT NULL,
99     PRIMARY KEY (`student_id`)
00 );
01
02 -- INSERT TO CREATE SAMPLE DATA
03 INSERT INTO students2
04 VALUES (NULL, 'David', '1991-03-21', 60, 160);
05
06 INSERT INTO students2
07 VALUES (NULL, 'Sarah', '1991-04-18', 48, 152);
08
```

ยกตัวอย่าง Function `AUTO_INCREMENT`

สมมติว่าเราต้องการสร้างตารางเพื่อเก็บข้อมูลนักเรียน โดยกำหนดให้คอลัมน์ `student_id` เป็นคีย์หลักและมีค่าเริ่มต้นเป็นค่าที่เพิ่มขึ้นโดยอัตโนมัติเมื่อเพิ่มแถวใหม่ลงในตาราง เราสามารถเขียนคำสั่ง `CREATE TABLE` ดังนี้

## SQL

```
CREATE TABLE `students` (
    `student_id` int NOT NULL AUTO_INCREMENT,
    `name` varchar(255) NOT NULL,
    `birthday` date NOT NULL,
    `weight` int NOT NULL,
    `height` int NOT NULL,
```

```
    PRIMARY KEY (`student_id`)  
);
```

### Use code with caution. [Learn morecontent\\_copy](#)

เมื่อเราใช้คำสั่ง `INSERT INTO` เพื่อเพิ่มแถวใหม่ลงในตาราง คอลัมน์ `student_id` จะได้รับค่าเริ่มต้นเป็น 1 โดยอัตโนมัติ เช่น

#### SQL

```
INSERT INTO `students` (  
    `name`,  
    `birthday`,  
    `weight`,  
    `height`  
) VALUES (  
    'John Doe',  
    '2000-01-01',  
    50,  
    170  
)
```

### Use code with caution. [Learn morecontent\\_copy](#)

ในบรรทัด `INSERT INTO` ด้านบน คอลัมน์ `student_id` จะได้รับค่าเริ่มต้นเป็น 1 เนื่องจากเป็นคอลัมน์ที่มีข้อบังคับ `AUTO_INCREMENT`

หากเราต้องการเพิ่มนักเรียนอีกคน คอลัมน์ `student_id` จะได้รับค่าเริ่มต้นเป็น 2 โดยอัตโนมัติ เช่น

#### SQL

```
INSERT INTO `students` (  
    `name`,  
    `birthday`,  
    `weight`,  
    `height`  
) VALUES (  
    'Jane Doe',  
    '2001-02-02',  
    45,
```

```
160  
);
```

### Use code with caution. [Learn more](#)

ในบรรทัด `INSERT INTO` ด้านบน คอลัมน์ `student_id` จะได้รับค่าเริ่มต้นเป็น 2 เนื่องจากเป็นคอลัมน์ที่มีข้อบังคับ `AUTO_INCREMENT` และค่าก่อนหน้าคือ 1

เราสามารถกำหนดค่าเริ่มต้นสำหรับคอลัมน์ `AUTO_INCREMENT` ได้โดยใช้ตัวเลือก `START WITH` ตัวอย่างเช่น หากเราต้องการเริ่มต้นค่าเริ่มต้นของคอลัมน์ `student_id` เป็น 100 เราสามารถใช้คำสั่ง `CREATE TABLE` ดังนี้

### SQL

```
CREATE TABLE `students` (  
    `student_id` int NOT NULL AUTO_INCREMENT START WITH 10  
    0,  
    `name` varchar(255) NOT NULL,  
    `birthday` date NOT NULL,  
    `weight` int NOT NULL,  
    `height` int NOT NULL,  
    PRIMARY KEY (`student_id`)  
);
```

ในตัวอย่างนี้ คอลัมน์ `student_id` จะเริ่มต้นด้วยค่า 100 เมื่อเราเพิ่มแคลวใหม่ลงในตาราง

## ALTER แก้ไขข้อมูล

```
ALTER TABLE students2  
ADD shirt_size varchar(1);
```

## TRUNCATE TABLE การล้างข้อมูล

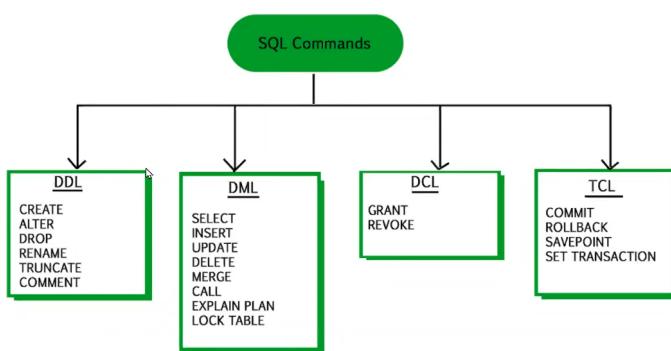
```
TRUNCATE TABLE students2;
```

## DROP TABLE การลบข้อมูลออก

```
DROP TABLE students2;
```

## Groups of SQL

### SQL: Groups of SQL commands



ลิ้งค์ที่มา: <https://www.geeksforgeeks.org/sql-ddl-dml-tcl-dcl/>

#### Data Engineer ใช้บ่อย

- **DDL** เช่น CREATE, ALTER, DROP
- **DML** เช่น SELECT, INSERT, UPDATE, DELETE

#### Data Engineer ใช้ไม่บ่อย

- **DCL** สำหรับจัดการ user
- **TCL** สำหรับจัดการ commit



## Optimisation : index

### SQL Optimisation: Index



Index เป็นการจำเลขอว่า ทำให้ Database ไม่ต้องอ่านข้อมูลตั้งแต่ แรก ทำให้ค้นหาข้อมูลได้เร็วขึ้น

score_id	student_id	subject	score
1	1	Maths	70
2	1	Computer	85
3	1	Science	74
4	2	Maths	68
5	2	Computer	72
6	2	Science	83
7	3	Maths	56
8	3	Computer	92
9	3	Science	47

Index on student\_id

แบ่ง Partition

Student\_id <= 2

Student\_id > 2

score_id	student_id	subject	score
1	1	Maths	70
2	1	Computer	85
3	1	Science	74
4	2	Maths	68
5	2	Computer	72
6	2	Science	83

score_id	student_id	subject	score
7	3	Maths	56
8	3	Computer	92
9	3	Science	47

วิธีการ Partition ขึ้นอยู่กับ Database Software ที่ใช้มีหลาย Algorithms เช่น B-Tree



ตัวอย่างการใช้งาน index

```
SELECT * FROM scores  
WHERE student_id = 3;
```

## โปรแกรม SQL Client ช่วยทำงานกับ SQL

### โปรแกรม SQL Client ช่วยให้ทำงานกับ SQL



The screenshot shows a PostgreSQL database interface. On the left, there's a tree view of databases, schemas, tables, and other objects. In the center, a query editor window displays a complex SQL query involving multiple joins between 'actor', 'film', and 'film\_category' tables. Below the query editor is a results grid showing 9 rows of actor data, including columns for actor\_id, first\_name, last\_name, and last\_update.

actor_id	first_name	last_name	last_update
56	DAN	HARRIS	2006-02-15 04:34:33.00
113	MORGAN	HOPKINS	2006-02-15 04:34:33.00
93	ELLEN	PRESLEY	2006-02-15 04:34:33.00
27	JULIA	MCQUEEN	2006-02-15 04:34:33.00
131	JANE	JACKMAN	2006-02-15 04:34:33.00
196	BELA	WALKEN	2006-02-15 04:34:33.00
68	RIP	WINSLET	2006-02-15 04:34:33.00
95	DARYL	WAHLBERG	2006-02-15 04:34:33.00
79	MAE	HOFFMAN	2006-02-15 04:34:33.00



Data Engineer ต้องเข้าใจ SQL ระดับไหน



## Data Engineer ต้องเข้าใจ SQL ระดับไหน

1. เข้าใจโค้ด SQL แบบอ่านออก เขียนได้ สำหรับทำ ETL  
<< พัฒนาโดยการฝึกเขียนบ่อย ๆ
2. เข้าใจเบื้องหลังของ Database และ Data Warehouse เช่น Table, Column, Partition, Modelling  
<< พัฒนาได้จากประสบการณ์ และขึ้นอยู่กับ Database Software ที่ใช้
3. BONUS: Optimise SQL ให้เร็วขึ้น / ใช้ทรัพยากรน้อยลง  
<< พัฒนาได้จากประสบการณ์ และขึ้นอยู่กับ Database Software ที่ใช้



การฝึก SQL



## ฝึก SQL อะไร ได้กี่ไหน

1. W3School <https://www.w3schools.com/sql/default.asp> - เว็บไซต์ฟรี มีให้ลองรันโค้ด
2. SQLZoo <https://sqlzoo.net/> - เว็บไซต์ฟรี มีให้ลองรันโค้ด มีโจทย์ให้ลองทำ
3. SQLBolt <https://sqlbolt.com/> - เว็บไซต์ฟรี มีให้ลองรันโค้ด มีโจทย์ให้ลองทำ
4. Hackerrank <https://www.hackerrank.com/> - เว็บไซต์ฟรี มีให้ลองรันโค้ด มีโจทย์ให้ลองทำ
5. DataRockie <https://datarockie.teachable.com/p/intro-sql-for-data-analysis>  
- วิดีโอภาษาไทย 2 ชั่วโมง โดยแอดထอยคนดีคนเดิม

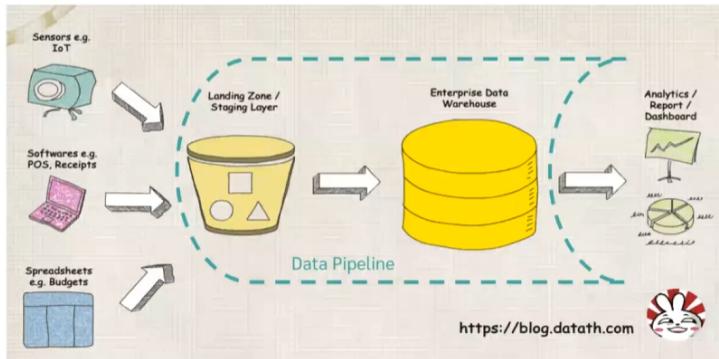


## Data Pipeline

ความหมายคือ ก่อลำเลียงข้อมูลจากต้นทาง ให้ไปถึงปลายทาง  
ยกตัวอย่างตามรูป data warehouse เป็นปลายทาง



## What is Data Pipeline?



ท่อในการลำเลียง  
ข้อมูล จาก  
แหล่งข้อมูล  
(Data Source)  
ไปยัง  
จุดหมาย  
(Destination)



## Why do we need Data Pipeline

เนื่องจากแต่ละ company มีการใช้ Tool ที่ต่างกัน จึงจำเป็นต้องมีตัวควบคุมเพื่อให้รวมกันเป็นหนึ่ง

อยู่ถูกที่ถูกเวลา สามารถใช้งานร่วมกันได้



## Why do we need Data Pipeline?



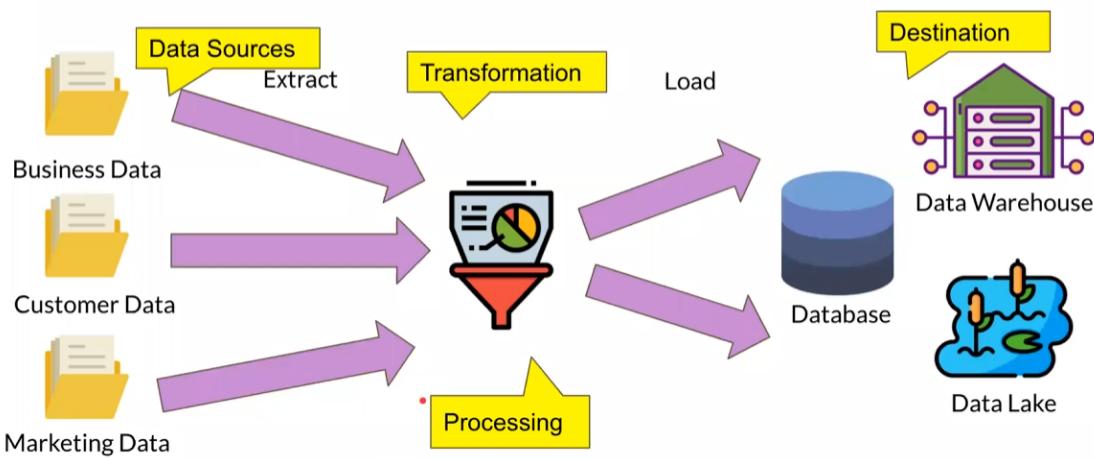
**Locality**  
รวมข้อมูลเป็นหนึ่งเดียว



**Decoupling**  
ไม่ต้องต่อท่อตรงจาก  
Source ไป Destination



# องค์ประกอบของ Data Pipeline



Data Engineer จะสร้างระบบให้เป็น Auto หลังจากเรียบร้อยแล้วจะ monitor อย่างเดียวเพื่อความสะดวกสบาย

E = Extract

- การดึงข้อมูลออกมาจากแหล่งข้อมูล(Data Source) ต่างๆ

## E = Extract



การดึงข้อมูลออกมาจากแหล่งข้อมูล (Data Source) ต่าง ๆ มีข้อที่ควรคำนึงถึงดังนี้:

- ประเภทข้อมูล**  
เช่น CSV, JSON, API, Database, Data Warehouse / Mart ฯลฯ
- หน้าตาข้อมูล**  
เช่น จำนวนคอลัมน์, ชื่อคอลัมน์, Format ของข้อมูลที่เก็บ (เช่น ชื่อ นามสกุล อาจเก็บรวมหรือแยกกัน)
- ความถี่ในการอัพเดท**  
เช่น อัพเดทข้อมูลทุกชั่วโมง หรืออาทิตย์ละครั้ง



T = Transform

- การเปลี่ยนแปลงข้อมูลสามารถทำได้หลากหลายรูปแบบ

## T = Transform



การเปลี่ยนแปลงข้อมูลสามารถทำได้หลากหลายรูปแบบ เช่น



- ปรับให้รูปแบบเหมาะสมกับระบบปลายทาง เช่น ต้นทางใช้วันที่แบบ DD/MM/YYYY ส่วนปลายทางใช้ YYYY-MM-DD, พศ vs คศ
- สรุปข้อมูล (Aggregation) เช่น จำนวนค่าเฉลี่ย, ผลรวม
- เพิ่มคุณค่าของข้อมูล (Enrichment) เช่น รวมข้อมูลยอดขายของแต่ละวัน กับข้อมูลสภาพอากาศในวันนั้น ๆ



## L = Load

- การนำข้อมูลเข้าไปในระบบปลายทาง

## L = Load



Data Warehouse



Data Lake

การนำข้อมูลเข้าไปในระบบปลายทาง

**Database, Data Warehouse, Data Lake** ที่สามารถเป็นระบบปลายทางได้ทั้งหมด

หากทำการ Transform มาถ่องหน้าแล้ว การ Load จะเป็นการส่งข้อมูลจากที่เก็บข้อมูลชั่วคราว (Staging Layer / Area) เข้าไปเก็บในระบบปลายทาง



## ETL vs ELT

คือลำดับขั้นตอนการทำงาน

ETL

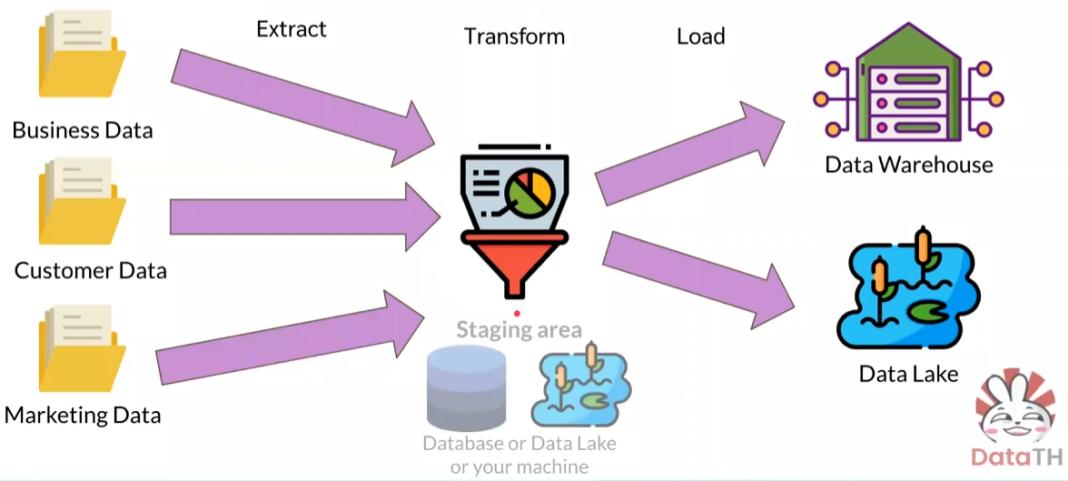
ถูกนิยามว่าใช้กันมาบาน

### ETL - Extract - Transform - Load

- วิธีปกติในการย้ายข้อมูล เป็นที่นิยมในการย้ายข้อมูลไปที่ต่าง ๆ
- ระบบปลายทางไม่จำเป็นต้องประมวลผล (Transform) ข้อมูลเยอะ
- ต้องมี Database หรือ Data Lake ก่อน Load (เรียกว่า Staging Area) เพื่อเก็บและประมวลผลข้อมูล
- Data Analyst ต้องรอ ETL เสร็จถึงจะได้ข้อมูล



## ETL - Extract, Transform, Load



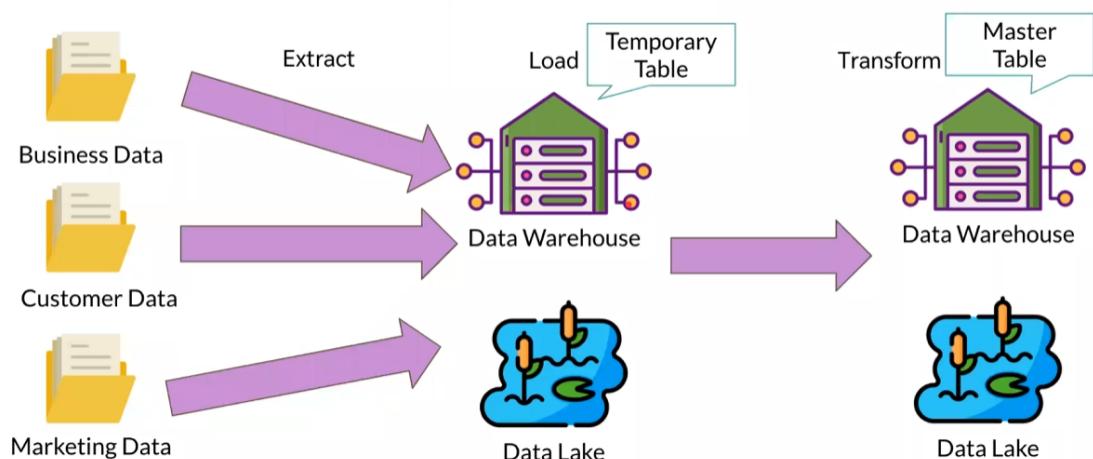
## ELT

เป็นวิธีสมัยใหม่ เนื่องจากยุคปัจจุบัน Hardware มีความรวดเร็ว จึงสามารถดำเนินการได้

## ELT - Extract - Load - Transform

- วิธีการย้ายข้อมูลสมัยใหม่ ระบบใหม่ ๆ จะสามารถทำได้ เช่น Redshift, Snowflake
- ระบบปลายทางจะต้องประมวลผลข้อมูลเยอะ
- ใช้ระบบปลายทางเป็น Staging Area
- Data Analyst เข้าถึงข้อมูลได้เร็วกว่า ไม่ต้องรอ ELT เสร็จ สามารถ Transform ข้อมูลดิบตอนเดียว ข้อมูลได้เลย

## ELT - Extract, Load, Transform



Key Considerations / Trade-offs



## Key Considerations / Trade-offs



**Accuracy**  
ความถูกต้องของข้อมูล



**Speed**  
ความเร็วในการย้ายข้อมูล



**Scalability**  
ความสามารถในการรับ  
ข้อมูลปริมาณมาก



**Security**  
ความปลอดภัยของข้อมูล  
ระหว่างส่ง

Tip: การเพิ่มเรื่องหนึ่ง อาจจะไปลดอีกเรื่อง เช่น เพิ่ม Speed แล้ว Accuracy ลดลง  
เราต้องหาบาลานซ์ให้เหมาะสมกับความต้องการของโปรเจค



## ประเภทของ Data Pipeline

### ประเภทของ Data Pipeline



**Initial Load / Historical load /  
Full load**

ดึงข้อมูลทั้งหมดจากแหล่งข้อมูล



**Incremental Load /  
Change Data Capture (CDC) Load**

ดึงข้อมูลใหม่ และข้อมูลที่เปลี่ยนแปลงจาก  
ครั้งล่าสุด



## ประเภทของการประมวลผลข้อมูล



## ประเภทของการประมวลผลข้อมูล (Processing)



Batch



Stream

**Scheduled:** ดึงตามช่วงเวลาที่กำหนด เช่น วันละครั้ง, ชั่วโมงละครั้ง

**Event-Driven:** ประมวลผลข้อมูลตอนที่ได้รับสัญญาณให้ประมวลผล

ข้อมูลจะถูกส่งเข้ามาทันที และเราประมวลผลทันที

(อภิปรีต์ที่นิยม คือ ทำเป็น Mini-batch ประมวลผลทุก 5 วินาที - 5 นาที)

ที่มา: <https://www.datanuts.io/batch-vs-stream-processing-explained/>



## Tool ในการทำ ETL



### เครื่องมือในการทำ ETL

นอกจากนี้ยังมีตัวควบคุม pipeline เรียกว่า pipeline orchestrator (CH 4)

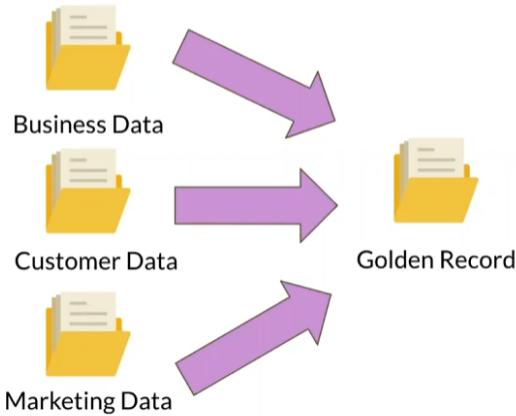
No Code	Code
<p>Pre-built connectors</p> <ul style="list-style-type: none"><li>• Fivetran</li><li>• Stitch</li></ul>  	<p>Drag and drop</p> <ul style="list-style-type: none"><li>• Talend</li><li>• Informatica</li><li>• Azure Data Factory</li></ul>  

## Data Integration

เป็นการนำข้อมูลจากหลากหลายแหล่งข้อมูล มารวมกันเป็นข้อมูลชุดเดียว



## Data integration คืออะไร



Data Integration เป็นการนำข้อมูลจากหลากหลายแหล่งแล้วจัดทำให้เป็นข้อมูลชุดเดียวที่มีประโยชน์กับองค์กร



## ประโยชน์ของ Data Integration

### ทำไม Data Integration ถึงมีประโยชน์



ช่วยให้เห็นภาพรวมของข้อมูลทั้งหมด

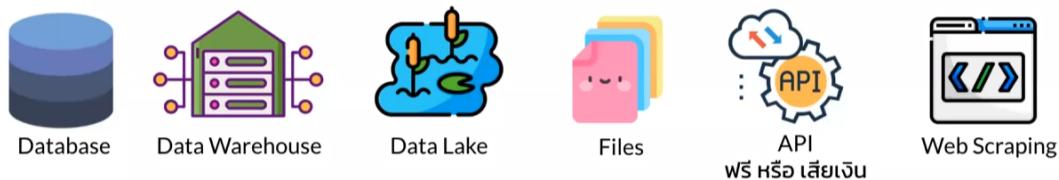
**Customer 360** ช่วยจากการรวมข้อมูลของทั้งบริษัท เราพบว่า คุณเพชรเป็นลูกค้าใช้บริการอินเตอร์เน็ต และได้ใช้บริการเบอร์โทรศัพท์ เรายังสามารถแนะนำบริการเบอร์โทรศัพท์ให้กับคุณ เพื่อเพิ่มรายได้ให้บริษัท

Informatica Customer 360

<https://www.informatica.com/products/master-data-management/customer-360.html>



# ข้อมูลมาจากไหนได้บ้าง



การทำ Data integration

## ประเภทหลักของการทำ Data integration



### 1) Schema integration

โครงสร้างข้อมูลแตกต่างกัน  
เช่น คอลัมน์ไม่เหมือนกัน, ใช้ชื่อเรียกต่างกัน, จัดกลุ่มข้อมูลไม่เหมือนกัน



### 2) Value integration

ข้อมูลเดียวกันแต่เก็บแตกต่างกัน  
เช่น ชื่อคน เก็บชื่อจริง กับชื่อเล่น (Jonathan กับ Jon)

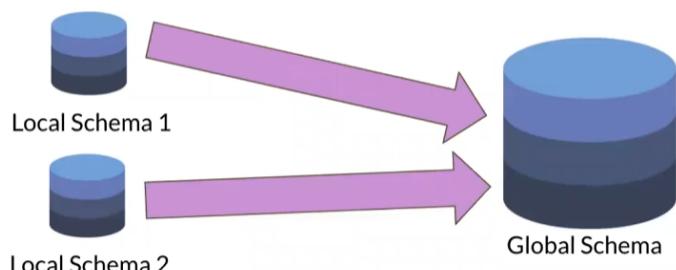


Schema integration

# สิ่งที่ต้องทำใน Schema Integration



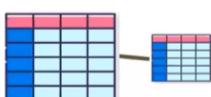
ต้องทำความเข้าใจโครงสร้างข้อมูลในแต่ละแหล่งข้อมูล (Local Schema) เพื่อนำมาสร้าง Global Schema



Tip: นัดคุยกับเจ้าของข้อมูลเพื่อสร้างความเข้าใจของแหล่งข้อมูลก่อน ถ้ามีคำศัพท์ เช่น มีคอลัมน์อะไรบ้าง อัพเดทบ่อยแค่ไหน หน้าตาข้อมูลเป็นยังไง



## ปัญหาที่พบบ่อยในการทำ Schema Integration



### 1. Structure Conflict

โครงสร้างข้อมูลไม่เหมือนกัน เช่น ระบบหนึ่งเก็บข้อมูลทุกอย่าง ใน 1 Table (Denormalised) อีกระบบเก็บข้อมูลแบบแยกเป็น 5 Table เล็ก ๆ (Normalised)



### 2. Naming Conflict

ข้อมูลเดียวกันแต่เรียกชื่อคอลัมน์ต่างกัน เช่น ระบบหนึ่งใช้ชื่อ คอลัมน์ Client ID อีกระบบใช้ชื่อ Customer ID

Tip: ขอข้อมูลเพิ่มจากเจ้าของแหล่งข้อมูลก่อนทำการแก้ไข



Value integration



## ปัญหาที่พบบ่อยในการทำ Value Integration

Mister Jonathan

=  
Mr. Jon

- ข้อมูลเดียวกัน แต่ค่าไม่เหมือนกัน ข้อมูลที่ค่าแตกต่างกันจากคละแหล่งข้อมูล  
อาจจะหมายถึงข้อมูลเดียวกัน

เช่น Mister Jonathan Holts กับ Mr. Jon อาจจะเป็นคนเดียวกัน ถ้ามีเบอร์โทรศัพท์เดียวกัน อีเมลเดียวกัน ที่อยู่เดียวกัน, ระยะทาง กิโลเมตร กับ ไมล์, ที่อยู่ระดับประเทศ กับ ระดับเมือง



- ข้อมูลไม่ตรงกัน เกิดขึ้นได้บ่อยเมื่อรวมข้อมูลจากหลายระบบที่เวลาอัพเดทต่างกัน เช่น ที่อยู่จากข้อมูลที่อัพเดทเป็นครั้ง กับที่อยู่จากข้อมูลที่อัพเดทเดือนละครั้ง จะไม่เหมือนกัน

Tip: แปลงข้อมูลให้อยู่ในหน่วยเดียวกันเสมอ และจับคู่ข้อมูลจากหลายแหล่งข้อมูลโดยใช้คอลัมน์ที่มีค่าร่วมกัน (Primary Key) เช่น customer ID

