



# Creating Databases and Tables

## Lecture

- 
- เราได้มุ่งเน้นไปที่การสืบค้นและการอ่านข้อมูลจากฐานข้อมูลและตารางที่มีอยู่
  - ตอนนี้มาเปลี่ยนโฟกัสไปที่การสร้างฐานข้อมูลและตารางของเราเอง
- 
- Section Overview
    - Data Types
    - Primary and Foreign Keys
    - Constraints
    - CREATE
    - INSERT
    - UPDATE
    - DELETE, ALTER, DROP
- 
- อันดับแรก เรามุ่งเน้นไปที่การเรียนรู้แนวคิดทางทฤษฎีบางอย่าง เช่น การเลือกประเภทข้อมูลที่ต้องใช้สำหรับค่าที่เก็บไว้ และการตั้งค่าข้อจำกัดที่เป็นไปได้
  - เราจะได้เรียนรู้เกี่ยวกับ primary and foreign keys
-

# Data Types

- Boolean
  - True or False
- Character
  - char, varchar, and text
- Numeric
  - integer and floating-point number
- Temporal
  - date, time, timestamp, and interval
- UUID
  - Universally Unique Identifiers
- Array
  - Stores an array of strings, numbers, etc.
- JSON
- Hstore key-value pair
- Special types such as network address and geometric data.

- เมื่อสร้างฐานข้อมูลและตาราง คุณควรพิจารณาอย่างรอบคอบว่าควรใช้ข้อมูลประเภทใดสำหรับข้อมูลที่จะจัดเก็บ
- ตรวจสอบเอกสารประกอบเพื่อดูข้อกำหนดของประเภทข้อมูล:

## Chapter 8. Data Types

Chapter 8. Data Types Table of Contents 8.1. Numeric Types 8.1.1. Integer Types 8.1.2. Arbitrary Precision Numbers 8.1.3. Floating-Point Types 8.1.4. Serial ...

 <https://www.postgresql.org/docs/current/datatype.html>



- ตัวอย่างเช่น
- ลองนึกภาพเราต้องการเก็บหมายเลขโทรศัพท์ควรเก็บเป็นตัวเลขหรือไม่?  
ถ้าใช่ เป็นตัวเลขประเภทไหน?
- เราสามารถดูเอกสารสำหรับตัวเลือก...

Name	Storage Size	Description	Range
<code>smallint</code>	2 bytes	small-range integer	-32768 to +32767
<code>integer</code>	4 bytes	typical choice for integer	-2147483648 to +2147483647
<code>bigint</code>	8 bytes	large-range integer	-9223372036854775808 to +9223372036854775807
<code>decimal</code>	variable	user-specified precision, exact	up to 131072 digits before the decimal point; up to 16383 digits after the decimal point
<code>numeric</code>	variable	user-specified precision, exact	up to 131072 digits before the decimal point; up to 16383 digits after the decimal point
<code>real</code>	4 bytes	variable-precision, inexact	6 decimal digits precision
<code>double precision</code>	8 bytes	variable-precision, inexact	15 decimal digits precision
<code>smallserial</code>	2 bytes	small autoincrementing integer	1 to 32767
<code>serial</code>	4 bytes	autoincrementing integer	1 to 2147483647
<code>bigserial</code>	8 bytes	large autoincrementing integer	1 to 9223372036854775807

- ตามข้อจำกัด คุณอาจคิดว่าเหมาะสมที่จะจัดเก็บเป็นประเภทข้อมูล BIGINT แต่เราควรคิดว่าอะไรดีที่สุดสำหรับสถานการณ์
- ทำไมต้องกังวลกับตัวเลขเลย? เราไม่ได้ใช้การคำนวณทางคณิตศาสตร์กับตัวเลข
- ดังนั้นการใช้ประเภทข้อมูล VARCHAR จึงน่าจะเหมาะสมกว่า
- ในความเป็นจริง การค้นหาแนวทางปฏิบัติที่ดีที่สุดทางออนไลน์ คุณจะพบว่าโดยปกติแล้วระบบจะแนะนำให้จัดเก็บเป็นประเภทข้อมูลตามข้อความเนื่องจากปัญหาต่างๆ
- ไม่มีการคำนวณทางคณิตศาสตร์
- เลขศูนย์นำหน้าอาจทำให้เกิดปัญหาได้ 7 และ 07 ถือว่าตัวเลขเหมือนกัน แต่ไม่ใช่หมายเลขโทรศัพท์เดียวกัน

- เมื่อสร้างฐานข้อมูลและตาราง ให้ใช้เวลาในการวางแผนสำหรับการจัดเก็บระยะยาว
- จำไว้ว่าคุณสามารถลบข้อมูลประวัติที่คุณตัดสินใจว่าจะไม่ใช้งานได้ตลอดเวลา แต่คุณไม่สามารถย้อนเวลากลับไปเพิ่มข้อมูลได้!

# Primary and Foreign Keys

- primary key คือคอลัมน์หรือกลุ่มของคอลัมน์ที่ใช้ระบุแถวที่ไม่ซ้ำกันในตาราง
- ตัวอย่างเช่น ในฐานข้อมูล dvdrental ของเรา เราพบว่าลูกค้ามีคอลัมน์ customer\_id ที่ไม่ซ้ำใครและเป็นคีย์หลักของพวกเขา
- primary key มีความสำคัญเนื่องจากช่วยให้เราแยกแยะได้ง่ายว่าควรใช้คอลัมน์ใดในการรวมตารางเข้าด้วยกัน

- Notice its integer based and unique

Query Editor

Query History

1

SELECT \* FROM customer

Data Output

Explain

Messages

Notifications

	customer_id [PK] integer	store_id smallint	first_name character varying (45)	last_name character varying (45)
1	524	1	Jared	Ely
2	1	1	Mary	Smith
3	2	1	Patricia	Johnson
4	3	1	Linda	Williams

- Foreign Key คือเขตข้อมูลหรือกลุ่มของเขตข้อมูลในตารางที่ระบุแถวในตารางอื่นโดยไม่ซ้ำกัน
- มีการกำหนด Foreign Key ในตารางที่อ้างอิงถึงคีย์หลักของตารางอื่น
- ตารางที่มี Foreign Key เรียกว่าตารางอ้างอิงหรือตารางลูก
- ตารางที่อ้างอิง Foreign Key เรียกว่าตารางอ้างอิงหรือ parent table

- ตารางสามารถมี Foreign Key ได้หลายคีย์ขึ้นอยู่กับ relationships with other tables.
- การเรียกคืนในตาราง payment ของฐานข้อมูล dvdrental แต่ละแถวการชำระเงินมี payment\_id (PK) ที่ไม่ซ้ำกัน และระบุลูกค้าที่ชำระเงินผ่าน customer\_id (FK เนื่องจากอ้างอิงถึง PK ของ customer table)

## • Primary Key for Payment Table

Query Editor   Query History

1 **SELECT** \* **FROM** payment

Data Output   Explain   Messages   Notifications

	payment_id [PK] integer	customer_id smallint	staff_id smallint	rental_id integer	amount numeric (5,2)
1	17503	341	2	1520	7.99
2	17504	341	1	1778	1.99
3	17505	341	1	1849	7.99
4	17506	341	2	2829	2.99

## • Multiple Foreign Key References

Query Editor   Query History

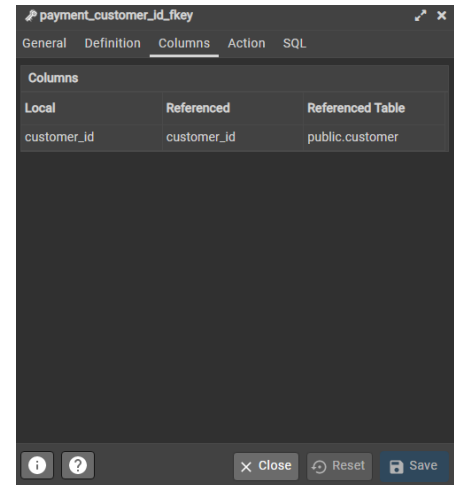
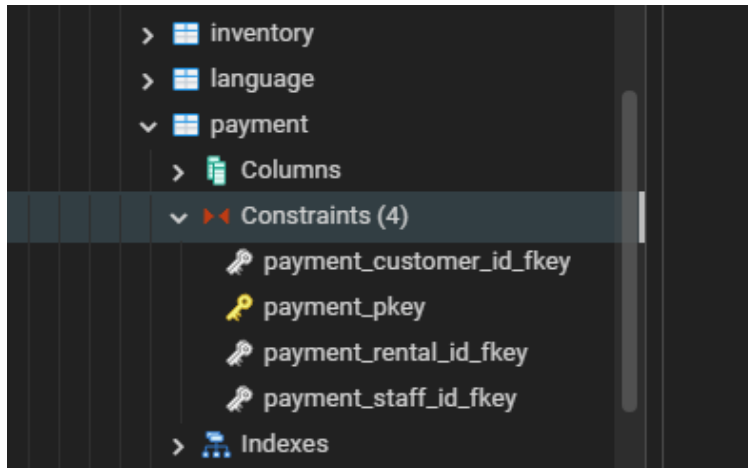
1 **SELECT** \* **FROM** payment

Data Output   Explain   Messages   Notifications

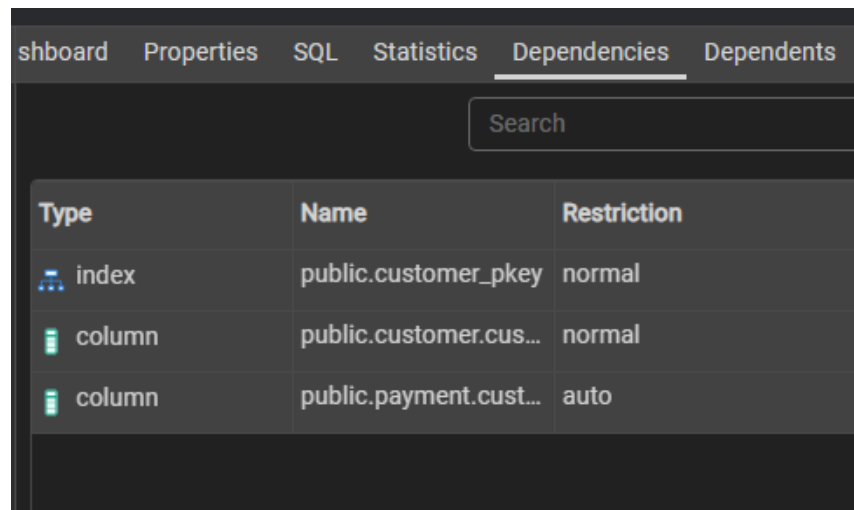
	payment_id [PK] integer	customer_id smallint	staff_id smallint	rental_id integer	amount numeric (5,2)
1	17503	341	2	1520	7.99
2	17504	341	1	1778	1.99
3	17505	341	1	1849	7.99
4	17506	341	2	2829	2.99

- คุณอาจเริ่มรู้แล้วว่า PK และ FK มักจะเป็นคอลัมน์สำหรับการรวมตารางสองเข้าด้วยกัน
- เมื่อสร้างตารางและกำหนดคอลัมน์ เราสามารถใช้ข้อจำกัดเพื่อกำหนดให้คอลัมน์เป็น PK หรือแบบความสัมพันธ์ของ FK กับตารางอื่น
- มาสำรวจคุณสมบัติของตารางอย่างรวดเร็วใน pgAdmin เพื่อดูวิธีรับข้อมูลเกี่ยวกับ PK และ FK!

- สามารถเข้ามาดูว่าคีย์มีความสัมพันธ์กับตารางใดได้ Properties... → Columns



- สามารถดูได้ใน Dependencies ว่าตารางใดขึ้นกับตารางใดบ้าง



# Constraints

- Constraints คือกฎที่บังคับใช้กับคอลัมน์ข้อมูลในตาราง
- ใช้เพื่อป้องกันการป้อนข้อมูลที่ไม่ถูกต้องลงในฐานข้อมูล
- สิ่งนี้ทำให้มั่นใจได้ถึงความถูกต้องและความน่าเชื่อถือของข้อมูลในฐานข้อมูล

ข้อจำกัดสามารถแบ่งออกเป็นสองประเภทหลัก:

- Column Constraints จำกัดข้อมูลในคอลัมน์ให้เป็นไปตาม certain conditions
- Table Constraints นำไปใช้กับทั้ง entire table จะใช้กับ individual column (นำไปใช้กับทั้งตารางแทนที่จะใช้กับแต่ละคอลัมน์)

#### **The most common constraints used:**

- NOT NULL Constraint ตรวจสอบให้แน่ใจว่าคอลัมน์ไม่มี NULL value
- UNIQUE Constraint ตรวจสอบให้แน่ใจว่าค่าทั้งหมดในคอลัมน์แตกต่างกัน
- PRIMARY Key ระบุแต่ละ row/record ในตารางฐานข้อมูลไม่ซ้ำกัน
- FOREIGN Key จำกัดข้อมูลตามคอลัมน์ในตารางอื่นๆ
- CHECK Constraint ตรวจสอบให้แน่ใจว่าค่าทั้งหมดในคอลัมน์เป็นไปตาม certain conditions
- EXCLUSION Constraint ตรวจสอบให้แน่ใจว่าหากมีการเปรียบเทียบสองแถวในคอลัมน์หรือนิพจน์ที่ระบุโดยใช้ตัวดำเนินการที่ระบุ การเปรียบเทียบเหล่านี้ทั้งหมดจะไม่ส่งคืนค่า TRUE
- (Ensures that if any two rows are compared on the specified column or expression using the specified operator, not all of these comparisons will return TRUE.)

#### **Table Constraints**

- CHECK (condition) เพื่อตรวจสอบเงื่อนไขเมื่อทำการแทรกหรือปรับปรุงข้อมูล
- REFERENCES เพื่อจำกัดค่าที่เก็บไว้ในคอลัมน์ที่ต้องมีอยู่ในคอลัมน์ในตารางอื่น
- UNIQUE (column\_list) บังคับให้ค่าที่จัดเก็บไว้ในคอลัมน์ที่อยู่ในวงเล็บไม่ซ้ำกัน
- PRIMARY KEY(column\_list) ให้คุณกำหนด PK ที่ประกอบด้วย multiple columns

ตอนนี้เราเข้าใจ data types, primary keys, foreign keys, and constraints เราพร้อมที่จะเริ่มใช้ SQL syntax เพื่อสร้างตารางแล้ว!

# CREATE

- syntax to create a table in SQL using the CREATE
- Full General Syntax
  - `CREATE TABLE table_name (  
    column_name TYPE column_constraint,  
    column_name TYPE column_constraint,  
    table_constraint table_constraint  
) INHERITS existing_table_name;`
- Common Simple Syntax
  - `CREATE TABLE table_name (  
    column_name TYPE column_constraint,  
    column_name TYPE column_constraint,  
);`

## SERIAL

- ใน PostgreSQL , sequence คือวัตถุฐานข้อมูลชนิดพิเศษที่สร้างลำดับของจำนวนเต็ม
- ลำดับมักใช้เป็นคอลัมน์ PK ในตาราง
- มันจะสร้าง sequence object และตั้งค่าถัดไปที่สร้างโดยลำดับเป็นค่าเริ่มต้นสำหรับคอลัมน์
- ศัพท์นี้เหมาะสำหรับ PK เพราะมันบันทึกการจํานวนเต็มที่ไม่ซ้ำกันให้คุณโดยอัตโนมัติเมื่อแทรก



- หากแถวถูกลบในภายหลัง คอลัมน์ที่มีชนิดข้อมูล SERIAL จะไม่ปรับเปลี่ยน (not adjust) โดยจะทำเครื่องหมายว่าแถวถูกลบออกจาก sequence
- 1,2,3,5,6,7 หาก 4 หายไป มันจะไม่มีใครมาแทน มันจะว่าง 4 ไว้

Name	Storage Size	Description	Range
smallint	2 bytes	small-range integer	-32768 to +32767
integer	4 bytes	typical choice for integer	-2147483648 to +2147483647
bigint	8 bytes	large-range integer	-9223372036854775808 to +9223372036854775807
decimal	variable	user-specified precision, exact	up to 131072 digits before the decimal point; up to 16383 digits after the decimal point
numeric	variable	user-specified precision, exact	up to 131072 digits before the decimal point; up to 16383 digits after the decimal point
real	4 bytes	variable-precision, inexact	6 decimal digits precision
double precision	8 bytes	variable-precision, inexact	15 decimal digits precision
smallserial	2 bytes	small autoincrementing integer	1 to 32767
serial	4 bytes	autoincrementing integer	1 to 2147483647
bigserial	8 bytes	large autoincrementing integer	1 to 9223372036854775807

- Example Syntax
  - CREATE TABLE players(
   
player\_id SERIAL PRIMARY KEY,
   
age SMALLINT NOT NULL
   
);

```
#CREATE
CREATE TABLE account(
    user_id SERIAL PRIMARY KEY,
    username VARCHAR(50) UNIQUE NOT NULL,
    password VARCHAR(50) NOT NULL,
    email VARCHAR(250) UNIQUE NOT NULL,
    created_on TIMESTAMP NOT NULL,
    last_login TIMESTAMP
);
-- UNIQUE เพื่อจำกัดว่าชื่อต้องไม่ซ้ำกัน , NOT NULL กำหนดว่าชื่อต้องไม่ว่าง
-- TIMESTAMP เพื่อติดตามว่าตารางถูกเพิ่มหรือสร้างขึ้นมาเวลาใด
-- ติดตามว่ามีการเข้าสู่ระบบเมื่อใด

CREATE TABLE job(
```

```

    job_id SERIAL PRIMARY KEY,
    job_name VARCHAR(200) UNIQUE NOT NULL
);

CREATE TABLE account_job(
    user_id INTEGER REFERENCES account(user_id),
    job_id INTEGER REFERENCES job(job_id),
    hire_date TIMESTAMP
)
-- SERIAL ใช้กับ PK เท่านั้น ถ้าเป็น FK ให้ใช้ INTEGER
-- REFERENCES ใช้ระบุว่าคอลัมน์ไหนที่เป็นคอลัมน์อ้างอิงกับ FK

```

# INSERT

- INSERT ช่วยให้คุณสามารถเพิ่มแถวลงในตารางได้
- INSERT allows you to add in rows to a table.
- General Syntax
  - INSERT INTO table (column1, column2, ...) VALUES
    - (value1, value2, ...),
    - (value1, value2, ...) ,...;
- Syntax for Inserting Values from another table:
  - INSERT INTO table(column1,column2,...) SELECT column1,column2,... FROM another\_table WHERE condition;

- โปรดทราบว่าค่าแถวที่แทรกจะต้องตรงกับตาราง รวมถึงข้อจำกัดต่างๆ
- คอลัมน์ SERIAL ไม่จำเป็นต้องระบุค่า

```
#INSERT
INSERT INTO account(username, password,email,created_on)
VALUES
('Jose','password','jose@mail.com', CURRENT_TIMESTAMP)

INSERT INTO job(job_name)
VALUES
('Astronaut')

INSERT INTO job(job_name)
VALUES
('Preident')

INSERT INTO account_job(user_id, job_id, hire_date)
VALUES
(1,1, CURRENT_TIMESTAMP)

INSERT INTO account_job(user_id, job_id, hire_date)
VALUES
(10,10, CURRENT_TIMESTAMP)
/*
ERROR: insert or update on table "account_job" violates foreign key constraint "account_job_user_id_fkey"
DETAIL: Key (user_id)=(10) is not present in table "account".
SQL state: 23503
*/
-- เนื่องจากไม่ PK ที่ 10 ที่จะให้สามารถ FK อ้างอิงถึงได้
```

# UPDATE

- คีย์เวิร์ด UPDATE ช่วยให้สามารถเปลี่ยนค่าของคอลัมน์ในตารางได้

- General Syntax
  - UPDATE table  
 SET column1 = value1,  
     column2 = value2 ,...  
 WHERE  
     condition;
- Using another table's values (UPDATE join)
  - UPDATE TableA  
 SET original\_col = TableB.new\_col  
 FROM tableB  
 WHERE tableA.id = TableB.id
- Set based on another column
  - UPDATE account  
 SET last\_login = created\_on

- ส่งคืนแถวที่ได้รับผลกระทบ

- Return affected rows
  - UPDATE account  
SET last\_login = created\_on  
RETURNING account\_id, last\_login

```
#UPDATE
UPDATE account
SET last_login = CURRENT_TIMESTAMP

UPDATE account
SET last_login = created_on

#UPDATE AND JOIN
UPDATE account_job
SET hire_date = account.created_on
FROM account
WHERE account_job.user_id = account.user_id

#RETURNING ส่งคืนค่า
UPDATE account
SET last_login = CURRENT_TIMESTAMP
RETURNING email, created_on, last_login
```

# DELETE

```
เราสามารถใช้ DELETE clause เพื่อลบแถวออกจากตาราง
DELETE FROM table
WHERE row_id = 1

#เราสามารถลบแถวตามที่มีอยู่ในตารางอื่น
DELETE FROM tableA
USING tableB
WHERE tableA.id=TableB.id

#เราสามารถลบแถวทั้งหมดออกจากตารางได้
DELETE FROM table
```

- เช่นเดียวกับคำสั่ง UPDATE คุณยังสามารถเพิ่ม RETURNING เพื่อส่งคืนแถวที่ถูกลบออกไป

```
INSERT INTO job(job_name)
VALUES
('Cowboy')

#ส่งคืนแถวที่ถูกลบ
DELETE FROM job
WHERE job_name = 'Cowboy'
RETURNING job_id, job_name
```

# ALTER

- ส่วนคำสั่ง ALTER อนุญาตให้เปลี่ยนแปลงโครงสร้างตารางที่มีอยู่ เช่น:
- The ALTER clause allows for changes to an existing table structure, such as:
  - Adding, dropping, or renaming columns
  - Changing a column's data type
  - Set DEFAULT values for a column
  - Add CHECK constraints
  - Rename table
- General Syntax
  - ALTER TABLE table\_name action

```
#Adding Columns
ALTER TABLE table_name
```

```

ADD COLUMN new_col TYPE

#Removing Columns
ALTER TABLE table_name
DROP COLUMN col_name

#Alter constraints
ALTER TABLE table_name
ALTER COLUMN col_name
SET DEFAULT value

#Alter constraints
ALTER TABLE table_name
ALTER COLUMN col_name
DROP DEFAULT

#Alter constraints
ALTER TABLE table_name
ALTER COLUMN col_name
SET NOT NULL

ALTER TABLE table_name
ALTER COLUMN col_name
DROP NOT NULL

ALTER TABLE table_name
ALTER COLUMN col_name
ADD CONSTRAINT constraint_name

```

```

CREATE TABLE information(
    info_id SERIAL PRIMARY KEY,
    title VARCHAR(500) NOT NULL,
    PERSON VARCHAR(50) NOT NULL UNIQUE
)

#RENAME TABLE
ALTER TABLE information
RENAME TO new_info;

#RENAME COLUMN
ALTER TABLE new_info
RENAME COLUMN person TO people;

#
INSERT INTO new_info(title)
VALUES
('some new title')
/*
ERROR:  null value in column "people" of relation "new_info" violates not-null constraint
DETAIL:  Failing row contains (1, some new title, null).
SQL state: 23502
*/
--> เนื่องจากมีค่าเป็น null ในคอลัมน์ people ซึ่งละเมิดข้อจำกัด not-null constraint

# แก้โดย ใช้ DROP เพื่อลบเลิกข้อจำกัด
# เราใช้ SET เพื่อกำหนดข้อจำกัด
ALTER TABLE new_info
ALTER COLUMN people DROP NOT NULL


```

Data Output Messages Notifications			
	info_id [PK] integer	title character varying (500)	people character varying (50)
1	2	some new title	[null]

- อ่านเพิ่ม **ALTER TABLE** ได้ที่

#### ALTER TABLE

ALTER TABLE ALTER TABLE — change the definition of a table Synopsis ALTER TABLE [ IF EXISTS ] [ ONLY ...

 <https://www.postgresql.org/docs/current/sql-altertable.html>



# DROP

- DROP ช่วยให้สามารถลบคอลัมน์ในตารางได้อย่างสมบูรณ์
- ใน PostgreSQL สิ่งนี้จะลบดัชนีและข้อจำกัดทั้งหมดที่เกี่ยวข้องกับคอลัมน์โดยอัตโนมัติ
- อย่างไรก็ตาม DROP จะไม่ลบคอลัมน์ที่ใช้ใน views, triggers, or stored procedures ขึ้นตอนการจัดเก็บ โดยไม่มีคำสั่ง CASCADE เพิ่มเติม

- General Syntax

- ALTER TABLE table\_name

DROP COLUMN col\_name



```
#Remove all dependencies (ลบการอ้างอิงทั้งหมด)
ALTER TABLE table_name
DROP COLUMN col_name CASCADE

#Check for existence to avoid error (ตรวจสอบการมีอยู่เพื่อหลีกเลี่ยง Error)
ALTER TABLE table_name
DROP COLUMN IF EXISTS col_name

#Drop multiple columns
ALTER TABLE table_name
DROP COLUMN col_one,
DROP COLUMN col_two
```

```
#DROP COLUMN
ALTER TABLE new_info
DROP COLUMN people

ALTER TABLE new_info
DROP COLUMN IF EXISTS people
/*
NOTICE: column "people" of relation "new_info" does not exist, skipping
ALTER TABLE
*/
```

# CHECK

- CHECK constraint ช่วยให้เราสามารถสร้างข้อจำกัดที่กำหนดเองได้มากขึ้นซึ่งเป็นไปตาม certain condition (เงื่อนไขที่กำหนด)
- เช่น การตรวจสอบว่าค่าจำนวนเต็มทั้งหมดที่กรอกเข้ามาต่ำกว่าเกณฑ์ที่กำหนด
- ในตัวอย่างคือเรามีเงื่อนไขที่มา check ข้อจำกัด (age>21)

- General Syntax
  - CREATE TABLE example(  
ex\_id SERIAL PRIMARY KEY,  
age SMALLINT CHECK (age > 21),  
parent\_age SMALLINT CHECK (  
parent\_age > age)  
);

```
CREATE TABLE employees(  
    emp_id SERIAL PRIMARY KEY,  
    first_name VARCHAR(50) NOT NULL,  
    last_name VARCHAR(50) NOT NULL,  
    birthdate DATE CHECK (birthdate > '1900-01-01'),  
    hire_date DATE CHECK (hire_date > birthdate),  
    salary INTEGER CHECK (salary > 0)  
)  
  
-- (birthdate > '1900-01-01') เช็คว่าวันเกิดควรมากกว่า 1900-01-01  
-- (hire_date > birthdate) เช็คว่าวันที่ลงทะเบียนต้องมากกว่าวันเกิด  
-- (salary > 0) เช็คว่าเงินเดือนต้องมากกว่า 0  
-- ถ้าหากข้อมูลที่ insert มาไม่เป็นตามเงื่อนไขระบบจะแจ้ง Error  
-- หากมีการ insert ล้มเหลว SERIAL จะข้ามที่ล้มเหลวไป  
-- เช่น รอบแรก แทรก b เข้าไป ล้มเหลว แก้ error แล้วมาลองอีกครั้ง  
-- b จะไปอยู่ใน แถวที่ 2 แทน
```