



# SQL

2110322 DATABASE SYSTEMS

CR: อาจารย์ดวงดาวและอาจารย์โซติรัตน์ สำหรับต้นแบบสไลด์

# การจัดการบรรยายเรื่อง SQL

- ต้องติดตั้ง RDBMS และเครื่องมือต่าง ๆ ให้เรียบร้อย
- วันนี้ ต้องรู้ **basic SQL commands**
- เราจะทดลองใช้ **SQL commands** ไปพร้อม ๆ กัน
- ใช้เวลาบรรยายทั้งสิ้น 3 ชม.
- ให้ทำ **Exercise 4 : SQL** (วันที่ 28 มกราคม 2568 บ่าย) หลังสอบ #1 เช้า
- จากนั้นรู้จักการเขียน **stored procedure** (วันที่ 30 มกราคม 2568 เช้า)
- ให้ทำ **Exercise 5 : Stored Proc** (วันที่ 30 มกราคม บ่าย)

## สิ่งที่นิสิตควรติดตั้งให้เรียบร้อยแล้ว

- Install Docker Desktop
- Install Postgres RDB – ใน docker
- Install PGAdmin – ใน docker ใช้งานผ่าน localhost:5050

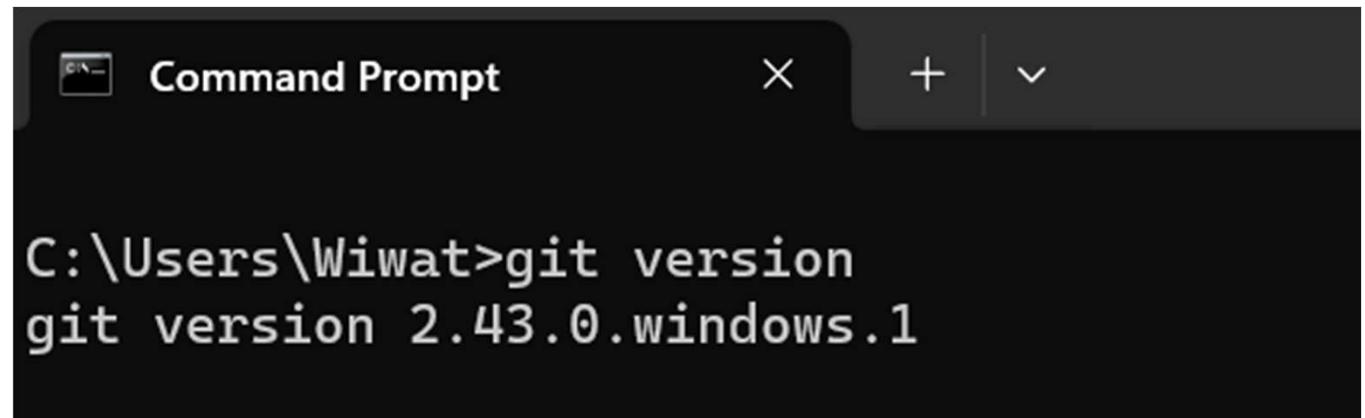


ใครยังทำไม่ครบ ยกมือขึ้น

ให้ทำตามดังนี้

## ทดสอบ Git

- C:\>git version



A screenshot of a Windows Command Prompt window titled "Command Prompt". The window shows the command "git version" being run and its output "git version 2.43.0.windows.1".

```
C:\Users\Wiwat>git version
git version 2.43.0.windows.1
```

# Initial Docker Desktop

A screenshot of the Docker Desktop interface. The left sidebar shows navigation options: Containers (selected), Images, Volumes, Builds (NEW), Dev Environments (BETA), Docker Scout, Extensions (with a plus icon for Add Extensions), and a three-dot menu. The main area is titled "Containers" with a "Give feedback" link. It displays container usage statistics: Container CPU usage (0.05% / 1000%) and Container memory usage (254.59MB / 7.44GB). Below this is a search bar and a filter option "Only show running containers". A table lists three running containers: 1) "postgresql" (Image: postgresql, Status: Running (2)), 2) "pgadmin" (Image: pgadmin4, Status: Running), and 3) "pg\_conta" (Image: postgres, Status: Running). Each row has an "Actions" column with icons for Stop, Open, and Delete. The bottom right of the table area says "Showing 3 items".

	Name	Image	Status	CPU (%)	Port(s)	Last started	Actions
<input type="checkbox"/>	postgresql	postgres/postgres	Running (2)	0.05%		13 minutes ago	<span>Stop</span> <span>⋮</span> <span>Delete</span>
<input type="checkbox"/>	pgadmin	ccf80fbf9c/dpage/pgadmin4	Running	0.05%	5050:80	13 minutes ago	<span>Stop</span> <span>⋮</span> <span>Delete</span>
<input type="checkbox"/>	pg_conta	b74c78802/postgres	Running	0%	5433:5432	13 minutes ago	<span>Stop</span> <span>⋮</span> <span>Delete</span>

# SQL (Structured Query Language)

- ภาษา SQL ถูกกำหนดโดยสถาบันมาตรฐานแห่งชาติของอเมริกา (American National Standards Institute: ANSI)
- ประกอบด้วย 2 ส่วนหลัก
  - **Data Definition Language (DDL)** เป็นชุดคำสั่งที่ใช้ในการกำหนดหรือปรับแต่งแก้ไข スキีมาของฐานข้อมูล
  - **Data Manipulation Language (DML)** เป็นชุดคำสั่งที่ใช้ในจัดการข้อมูล เช่น เพิ่ม ลด แก้ไข ข้อมูลใน ตารางต่างๆ ของฐานข้อมูล
  - **Data Control Language (DCL)** เป็นชุดคำสั่งที่ใช้ในจัดการสิทธิ์ต่างๆ ของผู้ใช้ฐานข้อมูล เช่น Grant หรือ Revoke สิทธิ์การ select, update, insert, etc. ให้กับ user ใดๆ

# Initiate PGAdmin

- เปิด web browser
- URL: localhost:5050

Login user:  
admin@admin.com

Password:  
root

pgAdmin

Login

admin@admin.com

....

[Forgotten your password?](#)

English

Login

The screenshot shows the pgAdmin 4 interface with two windows open. The top window displays the main menu with a red arrow pointing to the 'Server...' option under the 'Register' submenu. The bottom window shows the 'Register - Server' dialog, specifically the 'Connection' tab. The dialog form includes the following fields:

Field	Value
Host name/address	db
Port	5432
Maintenance database	postgres
Username	root
Kerberos authentication?	(Toggle off)
Password	...
Save password?	(Toggle off)
Role	(Empty)
Service	(Empty)

A yellow box highlights the 'Host Name: db' field. Another yellow box highlights the 'Username: root' field. A third yellow box highlights the 'Password: root' field. A red box highlights the warning message at the bottom left: 'Either Host name or Service must be specified.' At the bottom right, there are buttons for 'Close', 'Reset', 'Save', and 'Help'.

The screenshot shows the pgAdmin 4 interface. In the Object Explorer, under 'Servers (1) > DBServer > Databases (2)', there are two databases: 'postgres' and 'test\_db'. A yellow arrow points from the 'DBServer' node to the 'test\_db' node. A pink arrow points from the 'test\_db' node to a large purple cylinder labeled 'test\_db'. Another pink arrow points from the 'postgres' node to a larger purple cylinder labeled 'postgres'. A yellow box in the bottom-left corner contains the word 'Demo'.

postgres

test\_db

Demo

- DBServer มี 2 databases คือ
  - postgres
  - test\_db
- เราจะสร้าง database ใหม่กันนะครับ



# พร้อมแล้ว เข้าสู่เนื้อหาบรรยาย

---

## ประวัติความเป็นมา

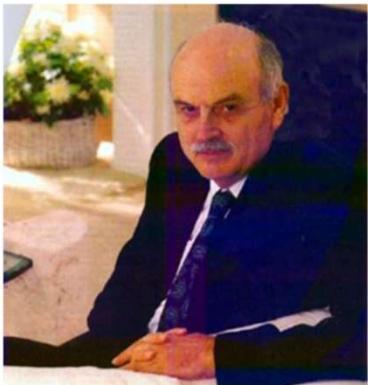
- IBM **Sequel** language developed as part of System R project at the IBM San Jose Research Laboratory
- Renamed Structured Query Language (SQL)
- ANSI and ISO standard SQL:
  - SQL-86
  - SQL-89
  - SQL-92
  - SQL:1999 (language name became Y2K compliant!)
  - SQL:2003
  - SQL:2016        (<https://www.iso.org/standard/63555.html>)

Lastest version SQL:2023

## ใครเป็นคนสร้างภาษา SQL

SQL vs RDBMS

- ถูกพัฒนาที่ IBM โดย Donald D. Chamberlin และ Raymond F. Boyce ในปี 1970
- เพื่อให้ใช้งานได้กับระบบจัดการฐานข้อมูล RDBMS ที่พัฒนามาจากงานตีพิมพ์ของ E.F.Codd ชื่อเรื่อง “A Relational Model of Data for Large Shared Data Banks”



Edgar Frank Codd



Donald D. Chamberlin



Raymond F. Boyce

**DDL**  
**(Data  
Definition  
Language  
commands)**

CREATE database      ใช้ในการสร้างฐานข้อมูล  
DROP database      ใช้ในการลบฐานข้อมูล

CREATE table      ใช้ในการสร้างตารางใหม่  
ALTER table      ใช้ในการเพิ่มลดคอลัมน์ในตาราง  
DROP table      ใช้ในการลบตารางออกจากฐานข้อมูล



# DDL สำหรับ DATABASE

---

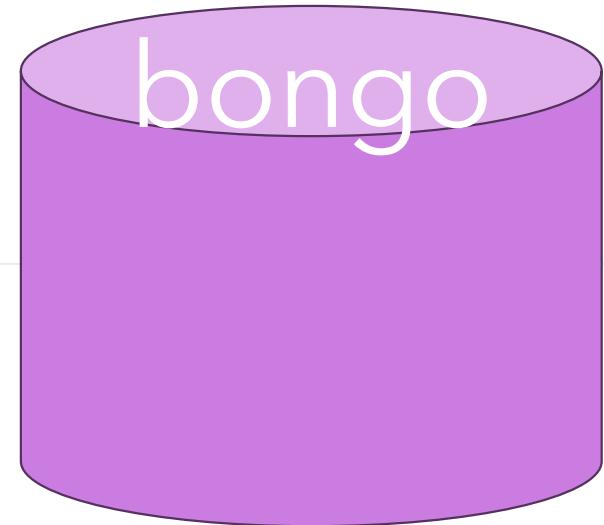
## Create database

Syntax

```
CREATE DATABASE [dbname];
```

ตัวอย่างคำสั่งจริง

```
CREATE DATABASE bongo;
```



## Drop database

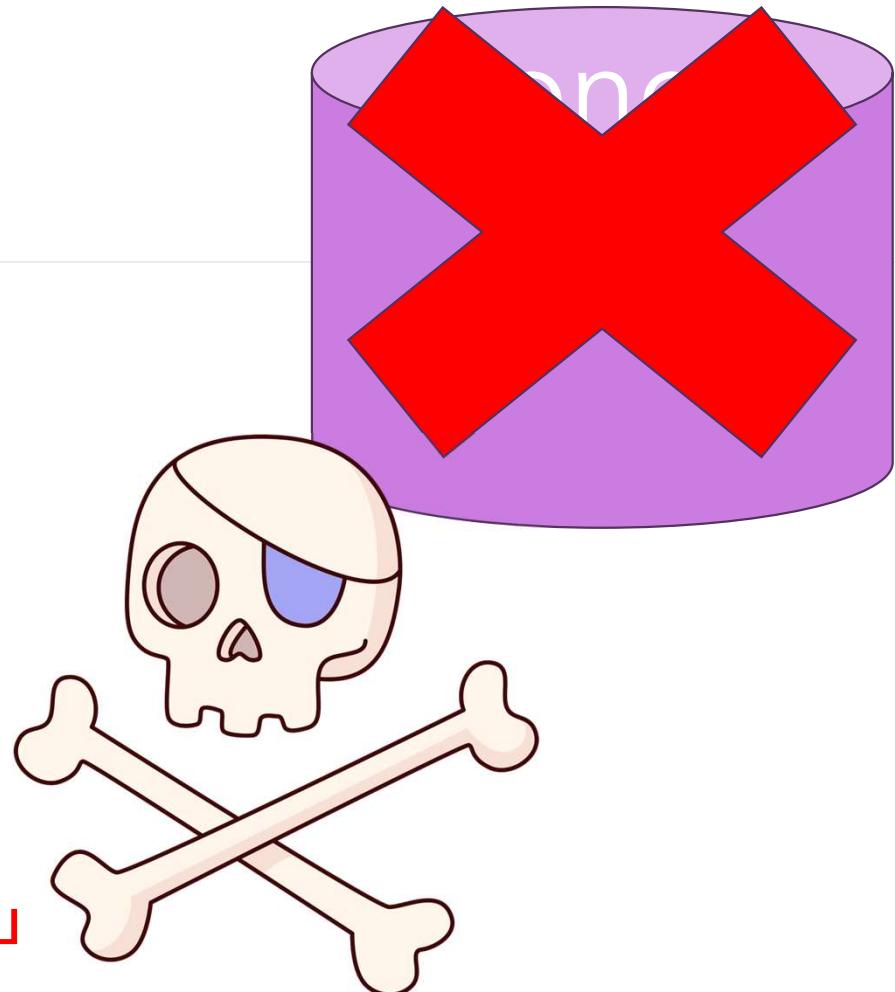
Syntax

```
DROP DATABASE [dbname];
```

ตัวอย่างคำสั่งจริง

```
DROP DATABASE bongo;
```

ก่อนลบฐานข้อมูล  
ต้องคิดแล้วคิดอีก นะครับ

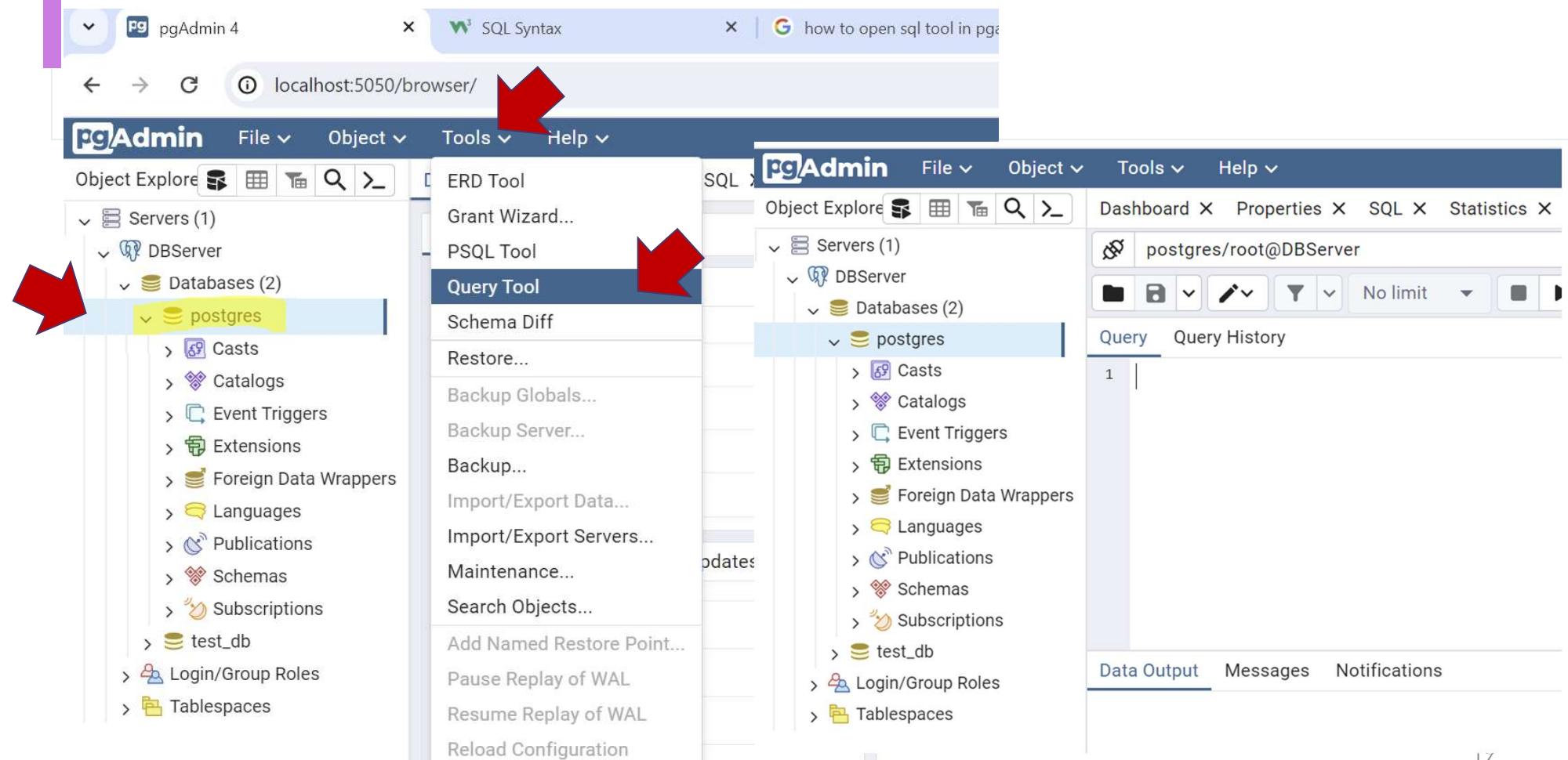




# ทดลอง DDL DATABASE

---

# ເປີດ Query Tool ອີ່ database 'postgres'



The screenshot shows the pgAdmin interface. In the Object Explorer, under 'Servers (1) > DBServer > Databases (2) > postgres', several database objects are listed: Casts, Catalogs, Event Triggers, Extensions, Foreign Data Wrappers, Languages, Publications, Schemas, Subscriptions, test\_db, Login/Group Roles, and Tablespaces. The 'Query' tab in the main window contains the following SQL code:

```
1 create database test1;
2
3 drop database test1;
4
5 drop database test1 with (force);
```

A yellow box highlights the first three lines of the query. A red box highlights the last two lines. A yellow box at the bottom left contains the word 'Demo'. To the right, a yellow box contains the following text:

1 ทดลองออกคำสั่งทีละคำสั่ง  
และดูผลลัพธ์  
2 อย่าลืม refresh database  
3 open query tool at new database

## เตรียมข้อมูล

- ให้ไป download SQL Query file ชื่อ ‘test1.txt’ จาก MCV
- copy commands ใน test1.txt ไปยัง query tool ที่ connect กับ database ‘test1’
- เวลาใช้คำสั่งให้ highlight เลพะส่วนคำสั่งที่ใช้และสั่ง execute



# DDL สำหรับ TABLE

---

## Create table (syntax)

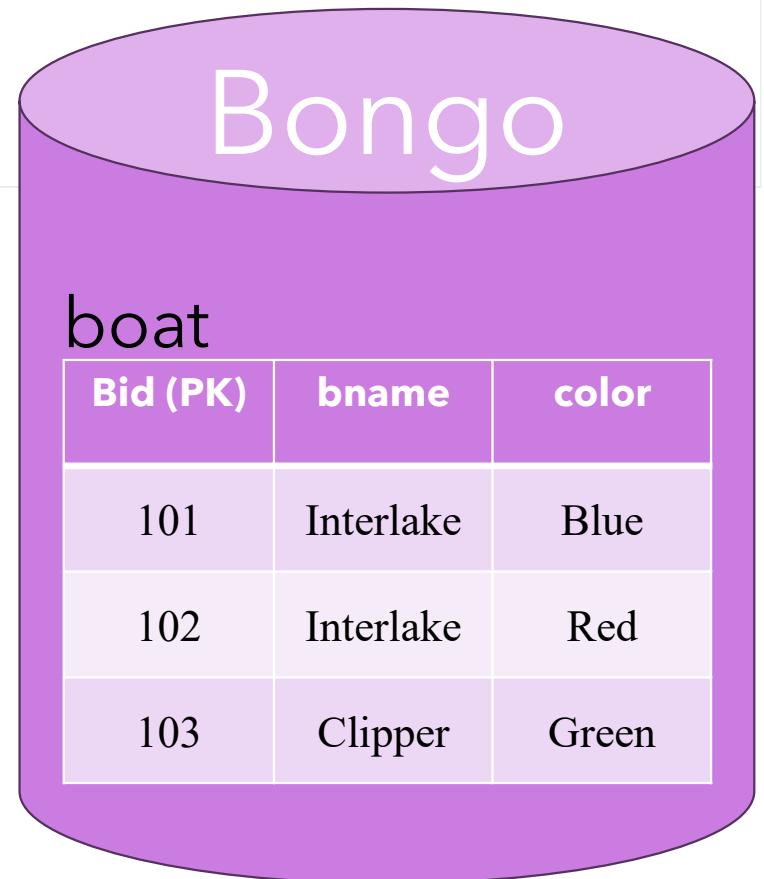
```
CREATE TABLE table_name  
(  
    colname1 data_type(size) constraint_name,  
    colname2 data_type(size) constraint_name,  
    colname3 data_type(size) constraint_name,  
    ...  
) ;
```

ระวังเรื่องสัญลักษณ์  
ต่าง ๆ ต้องครบถ้วน  
ด้วย เช่น  
( ), , ;

## Create table (Example)

## Create tables + constraints (BOAT)

```
colname      data_type(size)      constraint name  
CREATE TABLE boat (  
    bid VARCHAR(3) NOT NULL,  
    bname VARCHAR(45) DEFAULT NULL,  
    color VARCHAR(45) DEFAULT NULL,  
    PRIMARY KEY (bid)  
);
```



The cylinder is labeled "Bongo" at the top. Inside, there is a table named "boat" with three columns: "Bid (PK)", "bname", and "color". The data is as follows:

Bid (PK)	bname	color
101	Interlake	Blue
102	Interlake	Red
103	Clipper	Green



## PostgreSQL data types

Name	Aliases	Description
bigint	int8	signed eight-byte integer
bigserial	serial8	autoincrementing eight-byte integer
bit [ (n) ]		fixed-length bit string
bit varying [ (n) ]	varbit [ (n) ]	variable-length bit string
boolean	bool	logical Boolean (true/false)
box		rectangular box on a plane
bytea		binary data ("byte array")
character [ (n) ]	char [ (n) ]	fixed-length character string
character varying [ (n) ]	varchar [ (n) ]	variable-length character string
cidr		IPv4 or IPv6 network address
circle		circle on a plane
date		calendar date (year, month, day)
double precision	float8	double precision floating-point number (8 bytes)
inet		IPv4 or IPv6 host address
integer	int, int4	signed four-byte integer
interval [ <b>fields</b> ] [ (p) ]		time span

<https://www.postgresql.org/docs/current/datatype.html> [online access on Oct 7. 2023]

## PostgreSQL data types (cont.)

Name	Aliases	Description
json		textual JSON data
jsonb		binary JSON data, decomposed
line		infinite line on a plane
lseg		line segment on a plane
macaddr		MAC (Media Access Control) address
macaddr8		MAC (Media Access Control) address (EUI-64 format)
money		currency amount
numeric [ (p, s) ]	decimal [ (p, s) ]	exact numeric of selectable precision
path		geometric path on a plane
pg_lsn		PostgreSQL Log Sequence Number
pg_snapshot		user-level transaction ID snapshot
point		geometric point on a plane
polygon		closed geometric path on a plane
real	float4	single precision floating-point number (4 bytes)
smallint	int2	signed two-byte integer
smallserial	serial2	autoincrementing two-byte integer

<https://www.postgresql.org/docs/current/datatype.html> [online access on Oct 7, 2023]

## PostgreSQL data types (cont.)

Name	Aliases	Description
serial	serial4	autoincrementing four-byte integer
text		variable-length character string
time [ (p) ] [ without time zone ]		time of day (no time zone)
time [ (p) ] with time zone	timetz	time of day, including time zone
timestamp [ (p) ] [ without time zone ]		date and time (no time zone)
timestamp [ (p) ] with time zone	timestamptz	date and time, including time zone
tsquery		text search query
tsvector		text search document
txid_snapshot		user-level transaction ID snapshot (deprecated; see pg_snapshot)
uuid		universally unique identifier
xml		XML data

Data types ของแต่ละ RDBMS อาจจะไม่เหมือนกัน

## Constraints ที่ใช้บ่อย ๆ

- **NOT NULL** - Indicates that a column cannot store NULL value
- **UNIQUE** - Ensures that each row for a column must have a unique value
- **PRIMARY KEY** - A combination of a NOT NULL and UNIQUE. Ensures that a column (or combination of two or more columns) have a unique identity which helps to find a particular record in a table more easily and quickly
- **FOREIGN KEY** - Ensure the referential integrity of the data in one table to match values in another table
- **CHECK** - Ensures that the value in a column meets a specific condition
- **DEFAULT** - Specifies a default value for a column

## Not Null

Not Null and Unique constraint on P\_Id

```
CREATE TABLE Persons
(
    P_Id int NOT NULL,
    LastName varchar(255),
    FirstName varchar(255),
    Address varchar(255),
    City varchar(255),
    UNIQUE (P_Id)
)
```

Not Null and Unique constraint on multiple columns

```
CREATE TABLE Persons2
(
    P_Id int NOT NULL,
    FirstName varchar(255) NOT NULL,
    LastName varchar(255) NOT NULL,
    Address varchar(255),
    City varchar(255),
    CONSTRAINT uc_PersonID UNIQUE
    (P_Id,FirstName,LastName)
)
```

# PostgreSQL

## จัดการเรื่อง Null vs. Empty String



- Null คือไม่มีค่าใด ๆ ให้ใช้ null
- ส่วน Empty string คือ ตัวแปรชนิด string ที่มีค่า เป็น ""
- ให้ทดลองดูด้วย

# PK Constraint



PK constraint on P\_Id

```
CREATE TABLE Persons  
(  
P_Id int,  
LastName varchar(255) NOT NULL,  
FirstName varchar(255) NOT NULL,  
Address varchar(255),  
City varchar(255),  
PRIMARY KEY (P_Id)  
)
```

**primary key** declaration on a column  
automatically ensures **not null**  
ดังนั้น ไม่ต้องมี **NOT NULL** สำหรับ

PK constraint on multiple columns

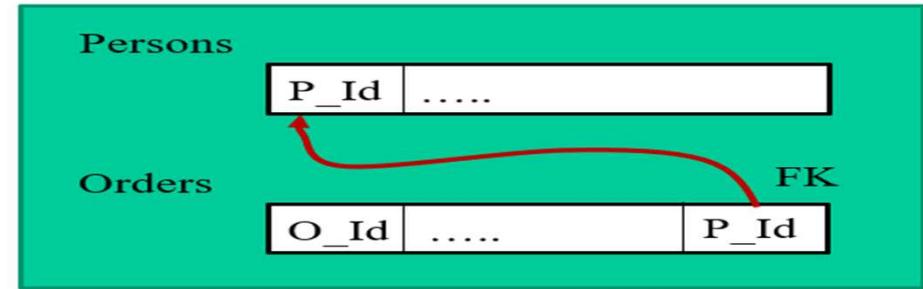
```
CREATE TABLE Persons2  
(  
P_Id int UNIQUE,  
FirstName varchar(255),  
LastName varchar(255),  
Address varchar(255),  
City varchar(255),  
CONSTRAINT pk_PersonID PRIMARY  
KEY (FirstName, LastName)  
)
```

# PK Constraint

PK constraint on P\_Id

```
CREATE TABLE Persons  
(  
P_Id int,  
LastName varchar(255) NOT NULL,  
FirstName varchar(255) NOT NULL,  
Address varchar(255),  
City varchar(255),  
PRIMARY KEY (P_Id)  
)
```

**primary key**  
declaration on a column  
automatically ensures **not null**  
ดังนั้น ไม่ต้องมี **NOT NULL** ก็ได้



# FK Constraint

FK constraint on P\_Id

```
CREATE TABLE Orders  
(  
O_Id int NOT NULL,  
OrderNo int NOT NULL,  
P_Id int,  
PRIMARY KEY (O_Id),  
FOREIGN KEY (P_Id) REFERENCES Persons(P_Id)  
)
```

## ทดลอง

```
create database test1;
```

```
create table professor  
( id varchar(10),  
  fname varchar(255),  
  lname varchar(255),  
  address varchar(255),  
  phone varchar(20),  
  PRIMARY KEY (id),  
  UNIQUE (phone)  
);
```

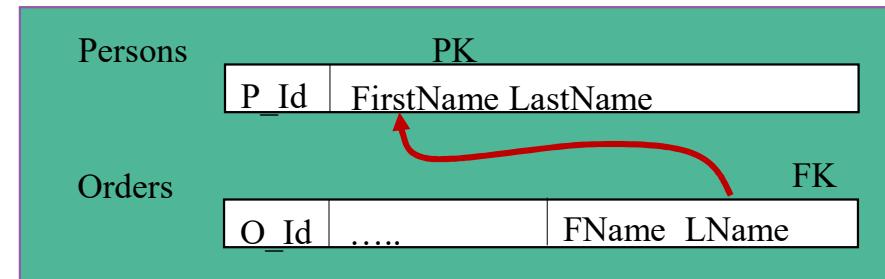
Demo

```
drop table professor;
```

# PK Constraint

PK constraint on multiple columns

```
CREATE TABLE Persons2
(
    P_Id int UNIQUE,
    FirstName varchar(255),
    LastName varchar(255),
    Address varchar(255),
    City varchar(255),
    CONSTRAINT pk_PersonID PRIMARY
    KEY (FirstName, LastName)
)
```



FK constraint on multiple columns

```
CREATE TABLE Orders2
(
    O_Id int NOT NULL,
    OrderNo int NOT NULL,
    FName varchar(255),
    LName varchar(255),
    PRIMARY KEY (O_Id),
    CONSTRAINT fk_PerOrders FOREIGN
    KEY (FName, LName) REFERENCES
    Persons2(FirstName, LastName)
)
```

## FK constraints on multiple columns

```
CREATE TABLE child_table
(
    column1 datatype [ NULL | NOT NULL ],
    column2 datatype [ NULL | NOT NULL ],
    ...
    CONSTRAINT fk_name
        FOREIGN KEY (child_col1, child_col2, ... child_col_n)
            REFERENCES parent_table (parent_col1, parent_col2, ... parent_col_n)
            [ ON DELETE { NO ACTION | CASCADE | SET NULL | SET DEFAULT } ]
            [ ON UPDATE { NO ACTION | CASCADE | SET NULL | SET DEFAULT } ]
);

```

# Check

Check constraint on P\_Id

```
CREATE TABLE Persons  
(  
P_Id int NOT NULL,  
LastName varchar(255),  
FirstName varchar(255),  
Address varchar(255),  
City varchar(255),  
CHECK (P_Id>0)  
)
```

Check constraint on multiple columns

```
CREATE TABLE Persons  
(  
P_Id int NOT NULL,  
LastName varchar(255),  
FirstName varchar(255),  
Address varchar(255),  
City varchar(255),  
CONSTRAINT chk_Person CHECK  
(P_Id>0 AND City='Sandnes')  
)
```

## Default

The default value will be added to all new records if no other value is specified.

### Default constraint on City

```
CREATE TABLE Persons  
(  
P_Id int NOT NULL,  
LastName varchar(255),  
FirstName varchar(255),  
Address varchar(255),  
City varchar(255) DEFAULT  
'Sandnes'  
)
```

### Check constraint by System Values (Function)

```
CREATE TABLE Orders  
(  
O_Id int NOT NULL,  
OrderNo int NOT NULL,  
P_Id int,  
OrderDate date DEFAULT  
GETDATE()  
)
```

GETDATE() เป็น system function  
Return ค่าวันที่ของระบบมาให้

# Auto-Increment Field for Postgres

Auto-Increment constraint on ID

CREATE TABLE Persons

```
(  
ID SERIAL,  
LastName varchar(255) NOT NULL,  
FirstName varchar(255),  
Address varchar(255),  
City varchar(255),  
PRIMARY KEY (ID)  
)
```

Not specify values for ID column on INSERT

```
INSERT INTO Persons  
(FirstName,LastName)  
VALUES ('Lars','Monsen')
```

ມີ smallserial ແລະ  
bigserial ດ້ວຍ

Name	Storage Size	Range
SMALLSERIAL	2 bytes	1 to 32,767
SERIAL	4 bytes	1 to 2,147,483,647
BIGSERIAL	8 bytes	1 to 9,223,372,036,854,775,807

# Auto-Increment Field for Postgres (2)

Auto-Increment constraint on ID starting from 100

```
CREATE TABLE Persons
(
    ID SERIAL,
    LastName varchar(255) NOT NULL,
    FirstName varchar(255),
    Address varchar(255),
    City varchar(255),
    PRIMARY KEY (ID)
);
```

```
ALTER SEQUENCE Persons_id_seq RESTART WITH 100;
```

CREATE DATABASE  
DROP DATABASE

ต้องทำได้แล้ว

CREATE TABLE using Constraints  
(PRIMARY KEY, FOREIGN KEY, NOT NULL, CHECK)  
DROP TABLE

PGAdmin - QUERY TOOL

## CASE STUDY 1: BONGO (BOAT RESERVATION DATABASE)

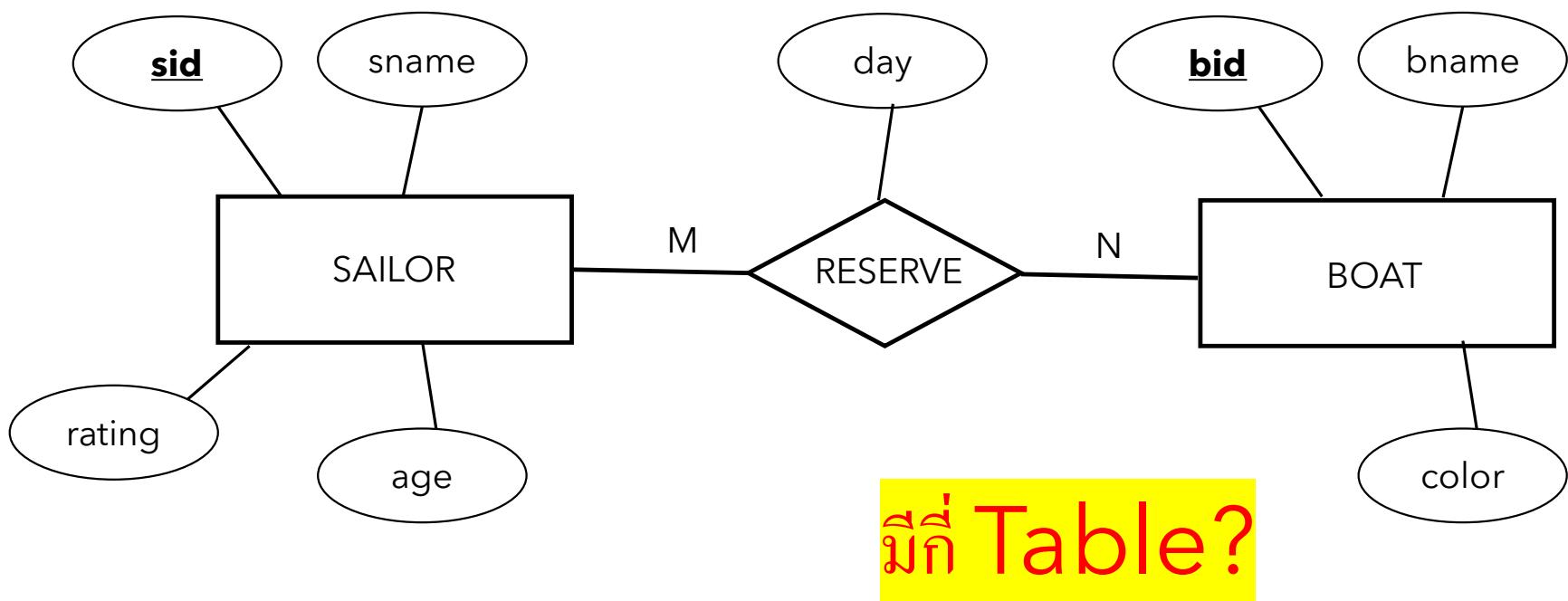
ลองสร้างฐานข้อมูลเล็ก ๆ เกี่ยวกับการจองเรือ

First toy database with DDL: CREATE DATABASE / CREATE TABLE

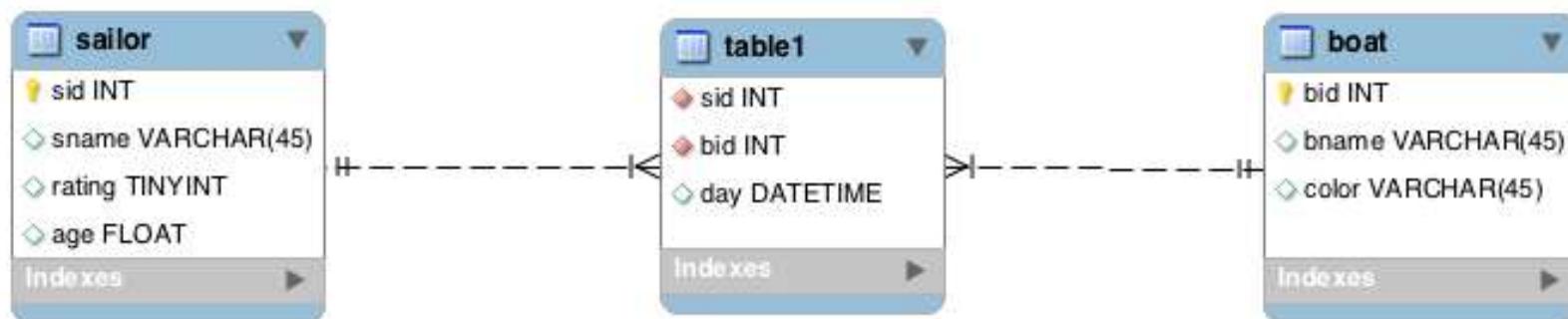
ลูกค้าต้องการจัดทำฐานข้อมูล Bongo โดยต้องการเก็บข้อมูลต่อไปนี้

1. ข้อมูลกำลังเลือกแต่ละคน ประกอบด้วย  
รหัสประจำตัว (sid) ชื่อ (rname) เรตติ้ง (rating) และอายุ (age)  
箭头指向 sid
2. ข้อมูลเรือแต่ละลำ ประกอบด้วย รหัสเรือ (bid) ชื่อเรือ (bname) และสีของเรือ (color)  
箭头指向 bid
3. ข้อมูลการจองเรือของกำลังเลือก (sid, bid, day)

## Bongo's ERR diagram (Chen's model)



## Bongo's EER diagram (Crow's foot model)



## Bongo's RDBMS schema design

- SAILOR (sid:integer, sname:string, rating:integer, age:real)
- BOAT (bid:integer, bname:string, color:string)
- RESERVE (sid:integer, bid:integer, day:date)

## เตรียมข้อมูล

- ให้ไป download SQL Query file ชื่อ ‘BONGO.txt’ จาก MCV
- create database bongo;
- และ copy commands จากแฟ้ม BONGO.txt ไปยัง query tool ที่ connect กับ database ‘bongo’
- เวลาใช้คำสั่งให้ highlight เลพะส่วนคำสั่งที่ใช้และสั่ง execute

## SQL – create tables + constraints

```
• CREATE TABLE table_name  
  (  
    colname1 data_type(size) constraint_name,  
    colname2 data_type(size) constraint_name,  
    colname3 data_type(size) constraint_name,  
    ...  
  ) ;
```

## Create tables + constraints (BOAT)

```
colname      data_type(size)      constraint name
CREATE TABLE boat (
    bid VARCHAR(3) NOT NULL,
    bname VARCHAR(45) DEFAULT NULL,
    color VARCHAR(45) DEFAULT NULL,
    PRIMARY KEY (bid)
);
```

## Create tables + constraints (SAILOR)

```
CREATE TABLE sailor (
    sid VARCHAR(2) NOT NULL,
    sname VARCHAR(45) DEFAULT NULL,
    rating INTEGER DEFAULT NULL,
    age NUMBERIC(10,2) DEFAULT NULL,
    PRIMARY KEY (sid)
);
```

## Create tables + constraints (RESERVE)

```
CREATE TABLE reserve (
    sid VARCHAR(2) NOT NULL,
    bid VARCHAR(3) NOT NULL,
    day DATE DEFAULT NULL,
    FOREIGN KEY (sid) REFERENCES sailor(sid),
    FOREIGN KEY (bid) REFERENCES boat(bid),
    PRIMARY KEY (sid, bid)
);
```

## Bongo's ER diagram (Crow's foot model)



## ทดลองสร้างฐานข้อมูล Bongo พร้อมกัน



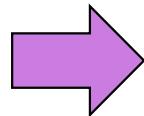


# คำสั่ง ALTER แก้ไข TABLE

---

Example:

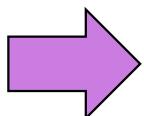
```
CREATE TABLE FoodCart (  
    date varchar(10),  
    food varchar(20),  
    profit float  
);
```



FoodCart

date	food	profit
------	------	--------

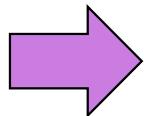
```
ALTER TABLE FoodCart (  
    ADD sold int  
);
```



FoodCart

date	food	profit	sold
------	------	--------	------

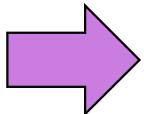
```
ALTER TABLE FoodCart(  
    DROP COLUMN profit  
);
```



FoodCart

date	food	sold
------	------	------

```
DROP TABLE FoodCart;
```



FoodCart

date	sold
------	------

A large purple 'X' is drawn over the entire table structure, indicating that the table no longer exists.

## Alter table : add column

- **ALTER TABLE** table\_name **ADD** column\_name datatype;

**ALTER TABLE** reserve **ADD** confirm boolean;

- **ALTER TABLE** table\_name **DROP** column\_name;

**ALTER TABLE** reserve **DROP** confirm;

# ALTER TABLE เหล่านี้ทำอะไร

```
ALTER TABLE reserve ADD CONSTRAINT fk_sid FOREIGN KEY (sid) REFERENCES sailor(sid);  
  
ALTER TABLE reserve ADD CONSTRAINT fk_bid FOREIGN KEY (bid) REFERENCES boat(bid);  
  
ALTER TABLE reserve ADD CONSTRAINT pk_reserve PRIMARY KEY (sid, bid);  
  
ALTER TABLE reserve ADD CONSTRAINT uc_combined UNIQUE (sid, bid);  
  
ALTER TABLE sailor ADD PRIMARY KEY (sid);
```

## Drop Table



```
DROP [TEMPORARY] TABLE [IF EXISTS] table_name [, table_name]  
[RESTRICT | CASCADE];
```

DROP TABLE boat; ได้รับการอนุญาต

DROP TABLE boat RESTRICT; ได้รับการอนุญาต

DROP TABLE boat CASCADE; ได้รับการอนุญาต

## Drop table w/ options

- RESTRICT : ถ้า primary key ของรายการในตารางนี้ไปเป็น foreign key ของตารางอื่น จะ drop ตารางนี้ไม่ได้
- CASCADE : ถ้า primary key ของรายการในตารางนี้ไปเป็น foreign key ของตารางอื่น Constraints ของข้อมูลเหล่านั้นต้องถูกลบออกด้วย

- SAILOR (sid:integer, sname:string, rating:integer, age:real)
- BOAT (bid:integer, bname:string, color:string)
- RESERVE (sid:integer, bid:integer, day:date)

DROP TABLE boat; ได้ใหม่

DROP TABLE reserve; ได้ใหม่

DROP TABLE boat RESTRICT; ได้ใหม่

DROP TABLE reserve; RESTRICT; ได้ใหม่

DROP TABLE boat CASCADE; ได้ใหม่

DROP TABLE sailor CASCADE; ได้ใหม่



# DML สำหรับ TABLE

---

# DML (Data Manipulation Language) commands (หลัก ๆ)

โปรแกรมเมอร์เรียกคำสั่ง  
เหล่านี้ว่าเป็น “CRUD”  
คำสามคือ CRUD คือ?

SELECT ใช้ในการ query หรือสืบค้นข้อมูล

INSERT ใช้ในการเพิ่มข้อมูลรายการใหม่เข้าตารางต่าง ๆ ในฐานข้อมูล

UPDATE ใช้ในการแก้ไขรายการข้อมูลต่าง ๆ ของตาราง ในฐานข้อมูล

DELETE ใช้ในการลบรายการข้อมูลต่าง ๆ ของตาราง ในฐานข้อมูล

## รูปแบบพื้นฐานในการทำ SQL queries

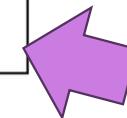
```
SELECT sid,sname FROM sailor  
WHERE age<40.0;
```

SELECT	<i>target-list</i>
FROM	<i>relation-list</i>
WHERE	<i>qualification</i>

- ***target-list*** A list of attributes of output relations in *relation-list*
- ***relation-list*** A list of relation names (possibly with a *range-variable* after each name)
  - e.g. Sailors S, Reserves R
- ***qualification*** Comparisons (Attr *op* const or Attr1 *op* Attr2, where *op* is one of <, >, ≤, ≥, =, ≠) combined using AND, OR and NOT.

## องค์ประกอบน้อยสุดของ 1 query

SELECT	<i>target-list</i>
FROM	<i>relation-list</i>
WHERE	<i>qualification</i>



ไม่ใส่ qualification  
WHERE True

*Every SQL Query must have:*

- *SELECT clause: specifies columns to be retained in result*
- *FROM clause: specifies a cross-product of tables*

*The WHERE clause (optional) specifies selection conditions on the tables mentioned in the FROM clause*

## Bongo's RDBMS schema design

- SAILOR (sid:integer, sname:string, rating:integer, age:real)
- BOAT (bid:integer, bname:rating, color:string)
- RESERVE (sid:integer, bid:integer, day:date)

เรา CREATE TABLE เหล่านี้ไว้เรียบร้อยแล้วใน Bongo database



*Sailors*

<i>sid</i>	<i>sname</i>	<i>rating</i>	<i>age</i>
22	Dustin	7	45.0
29	Brutus	1	33.0
31	Lubber	8	55.5
32	Andy	8	25.5
58	Rusty	10	35.0
64	Horatio	7	35.0
71	Zorba	10	16.0
74	Horatio	9	35.0
85	Art	3	25.5
95	Bob	3	63.5

*Boats*

<i>bid</i>	<i>bname</i>	<i>Color</i>
101	Interlake	Blue
102	Interlake	Red
103	Clipper	Green
104	Marine	red

<i>sid</i>	<i>bid</i>	<i>day</i>
22	101	10/10/14
22	102	10/10/14
22	103	10/08/14
22	104	10/07/14
31	102	11/10/14
31	103	11/06/14
31	104	11/12/14
64	101	09/05/14
64	102	09/08/14
74	103	09/08/14

*Reserves*

**Bongo's data**

จากนี่เราจะ INSERT ข้อมูลในตาราง  
sailor, boat, reserve  
(ดูคำสั่งในแฟ้ม bongo.txt)

## Insert data (sailor)

- **INSERT INTO** table\_name (colname, colname,...) **VALUES**  
(...), (...), (...), ...;

```
INSERT INTO sailor VALUES  
('22','Dustin',7,45),  
('29','Brutus',1,33),  
('31','Lubber',8,55.5),  
('32','Andy',8,25.5),  
('58','Rusty',10,35),  
('64','Horatio',7,35),  
('71','Zorba',10,16),  
('74','Horatio',9,35),  
('85','Art',3,25.5),  
('95','Bob',3,63.5);
```

optional

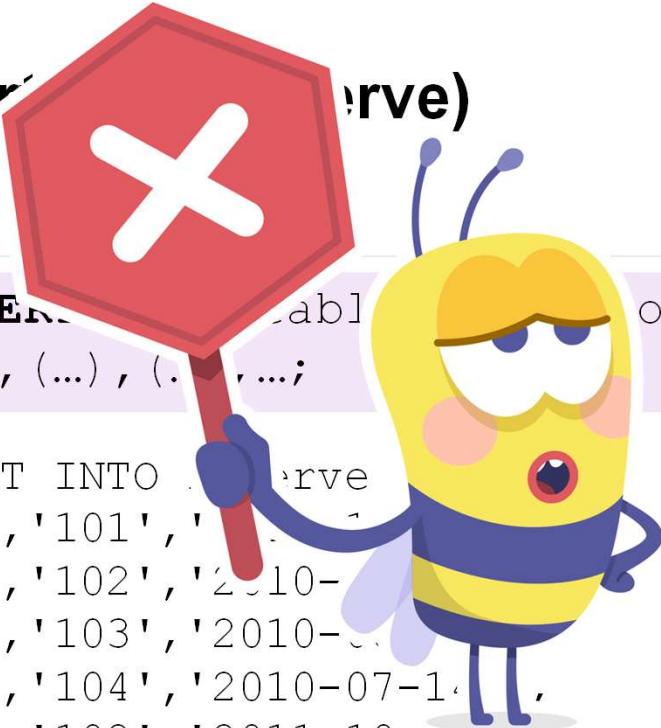


	sid [PK] character varying (2)	fname character varying (45)	rating integer	age numeric (10,2)
1	22	Dustin	7	45.00
2	29	Brutus	1	33.00
3	31	Lubber	8	55.50
4	32	Andy	8	25.50
5	58	Rusty	10	35.00
6	64	Horatio	7	35.00
7	71	Zorba	10	16.00
8	74	Horatio	9	35.00
9	85	Art	3	25.50
10	95	Bob	3	63.50

## Insert into reserve)

- **INSERT** [INTO table  
(...), (...), ...;]

```
INSERT INTO reserve
('22','101','2010-07-01',
('22','102','2010-07-01',
('22','103','2010-07-01',
('22','104','2010-07-01',
('31','102','2011-10-14',
('31', 'Data Output' Messages Notifications
('31', 'ERROR: Key (bid)=(101) is not present in table "boat".insert or update on table "reserve" violates foreign key constraint
"reserve_bid_fkey"
('64', 'ERROR: insert or update on table "reserve" violates foreign key constraint "reserve_bid_fkey"
SQL state: 23503
Detail: Key (bid)=(101) is not present in table "boat".
('74', '--- , --- -- + ,
```



optional

colname, colname,...) **VALUES**

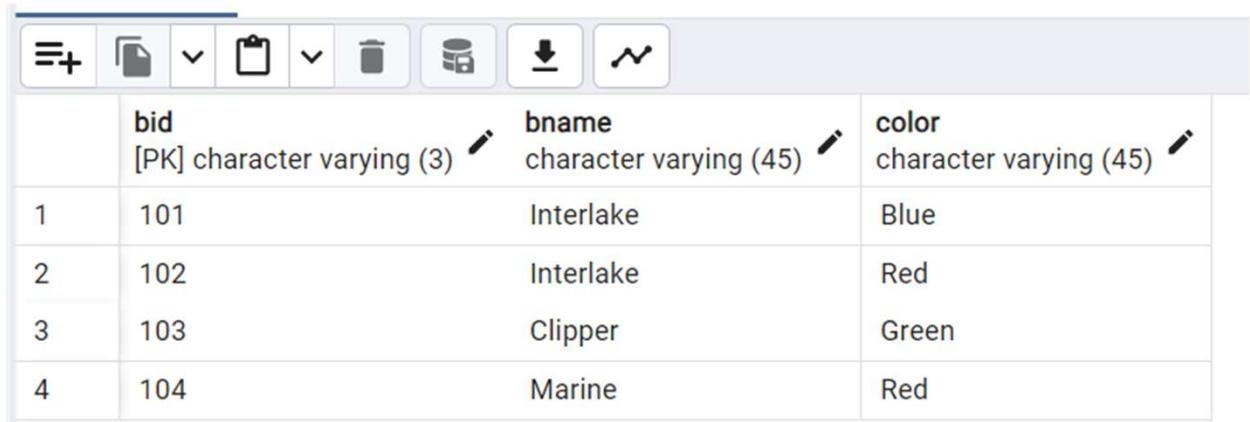
เรา **INSERT** ตาราง **reserve**  
ก่อน ไม่ได้ เพราะอะไร

## Insert data (boat)

- **INSERT INTO** table\_name (colname, colname, ...) **VALUES**  
(...), (...), (...), ...;

```
INSERT INTO boat VALUES
('101', 'Interlake', 'Blue'),
('102', 'Interlake', 'Red'),
('103', 'Clipper', 'Green'),
('104', 'Marine', 'Red');
```

optional



	bid [PK] character varying (3)	bname character varying (45)	color character varying (45)
1	101	Interlake	Blue
2	102	Interlake	Red
3	103	Clipper	Green
4	104	Marine	Red

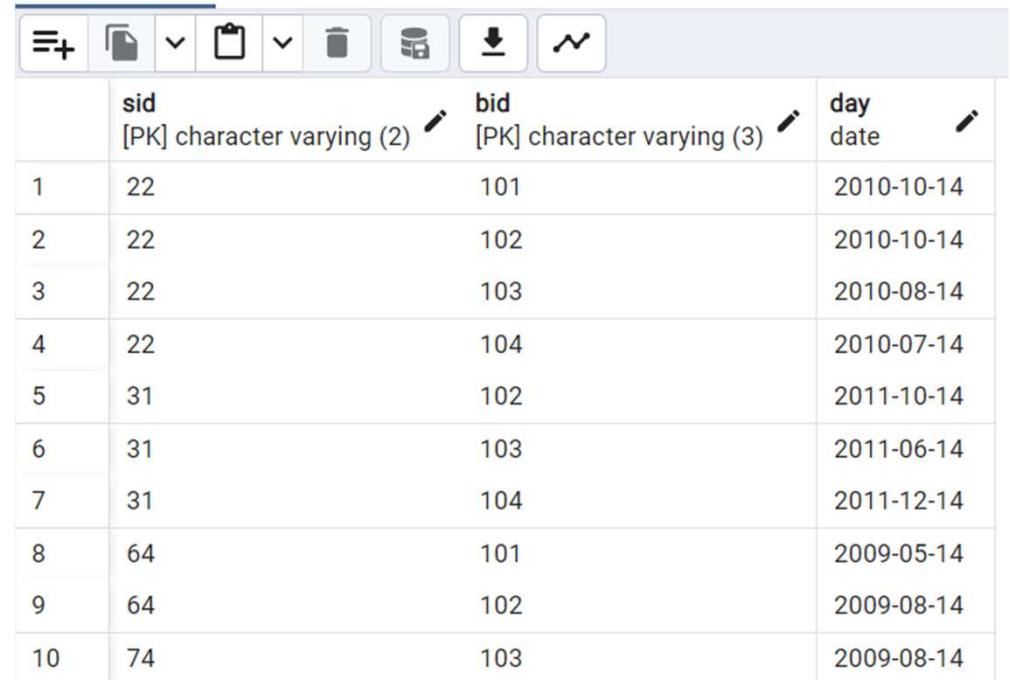
ให้ insert data เข้าตาราง reserve อีกที และลอง select ดู

# Insert data (RESERVE)

- **INSERT INTO** table\_name (colname, colname, ...) **VALUES**  
(...), (...), (...), ...;

```
INSERT INTO reserve VALUES  
('22','101','2010-10-14'),  
('22','102','2010-10-14'),  
('22','103','2010-08-14'),  
('22','104','2010-07-14'),  
('31','102','2011-10-14'),  
('31','103','2011-06-14'),  
('31','104','2011-12-14'),  
('64','101','2009-05-14'),  
('64','102','2009-08-14'),  
('74','103','2009-08-14');
```

optional



	sid [PK] character varying (2)	bid [PK] character varying (3)	day date
1	22	101	2010-10-14
2	22	102	2010-10-14
3	22	103	2010-08-14
4	22	104	2010-07-14
5	31	102	2011-10-14
6	31	103	2011-06-14
7	31	104	2011-12-14
8	64	101	2009-05-14
9	64	102	2009-08-14
10	74	103	2009-08-14

## Insert data (RESERVE)

optional

- **INSERT INTO** table\_name (colname, colname, ...) **VALUES**  
(...), (...), (...), ...;

```
INSERT INTO reserve VALUES  
('22','101','2010-10-14'),  
('22','102','2010-10-14'),  
('22','103','2010-08-14'),  
('22','104','2010-07-14'),  
('31','102','2011-10-14'),  
('31','103','2011-06-14'),  
('31','104','2011-12-14'),  
('64','101','2009-05-14'),  
('64','102','2009-08-14'), optional  
('74','103','2009-08-14');
```

ลอง **INSERT** รายการ  
reserve ต่อไปนี้แล้ว

ผลลัพธ์

```
INSERT INTO reserve (sid, bid, day) VALUES (22, 105, '2009-08-14');  
INSERT INTO reserve (sid, bid, day) VALUES (21, 101, '2009-08-14');
```

—

# ลงมือ INSERT กันได้

---

Demo



# ต่อไปนี้เป็นการใช้ SELECT (ใช้ทดลองไปพร้อมฟังบรรยาย)

---

# DML (Data Manipulation Language)

## commands

```
SELECT    target-list  
FROM      relation-list  
WHERE     qualification
```

- Target-list ระบุพิลต์หรือชุดของพิลด์ที่ต้องการแสดง
- Relation-list ระบุรีเลชันหรือชุดของรีเลชันที่ต้องการดึงข้อมูลออกมานะ
- Qualification ระบุเงื่อนไขที่ใช้ในการดึงข้อมูล

# SQL query: แสดงรายชื่อและอายุของกำล้าสีทั้งหมดที่อยู่ในตาราง sailor

sailor			
sid	sname	rating	age
22	Dustin	7	45.0
29	Brutus	1	33.0
31	Lubber	8	55.5
32	Andy	8	25.5
58	Rusty	10	35.0
64	Horatio	7	35.0
71	Zorba	10	16.0
74	Horatio	9	35.0
85	Art	3	25.5
95	Bob	3	63.5

sid	bid	day
22	101	10/10/14
22	102	10/10/14
22	103	10/08/14
22	104	10/07/14
31	102	11/10/14
31	103	11/06/14
31	104	11/12/14
64	101	09/05/14
64	102	09/08/14
74	103	09/08/14

reserve

```
SELECT sname, age
FROM sailor;
```

bid	bname	Color
101	Interlake	Blue
102	Interlake	Red
103	Clipper	Green
104	Marine	red

## SQL query: สืบค่ามและแสดงรายชื่อและ อายุของกะลาสีทั้งหมดในตาราง sailor

```
SELECT sname, age  
FROM   sailor;
```

sname	age
Dustin	45.00
Brutus	33.00
Lubber	55.50
Andy	25.50
Rusty	35.00
Horatio	35.00
Zorba	16.00
Horatio	35.00
Art	25.50
Bob	63.50
(10 rows)	

ระวังว่า  
ตารางผลลัพธ์  
มี row ที่ซ้ำกันได้  
 เพราะเราเลือก select บาง  
 column ที่ไม่ใช่ key

Demo

## SQL query: แสดงรายชื่อที่ไม่ซ้ำกันและอายุของกำล้าสีทึ่งหมวดในตาราง sailor

```
SELECT DISTINCT sname, age  
FROM sailor;
```

sname	age
Horatio	35.00
Art	25.50
Brutus	33.00
Dustin	45.00
Rusty	35.00
Bob	63.50
Zorba	16.00
Andy	25.50
Lubber	55.50
(9 rows)	

$\pi_{sname, age}(\text{sailor})$

การทำ Projection  $\pi$

กำหนดไว้ว่าตารางต้องมี Entity

Integrity คือทุก row ต้อง unique

แต่ต้องระวังว่า จำนวน sailor อาจจะนับ  
จำนวนไม่ถูก

Demo

# SQL query: แสดงรายชื่อ kabala สีทึ้งหมดที่เคย

จองเรือหมายเลข 103

sailor			
sid	sname	rating	age
22	Dustin	7	45.0
29	Brutus	1	33.0
31	Lubber	8	55.5
32	Andy	8	25.5
58	Rusty	10	35.0
64	Horatio	7	35.0
71	Zorba	10	16.0
74	Horatio	9	35.0
85	Art	3	25.5
95	Bob	3	63.5

sid	bid	day
22	101	10/10/14
22	102	10/10/14
22	103	10/08/14
22	104	10/07/14
31	102	11/10/14
31	103	11/06/14
31	104	11/12/14
64	101	09/05/14
64	102	09/08/14
74	103	09/08/14

boat		
bid	bname	Color
101	Interlake	Blue
102	Interlake	Red
103	Clipper	Green
104	Marine	red

```
SELECT S.sname  
FROM sailor S, reserve R  
WHERE S.sid = R.sid AND R.bid = 103;
```

```
SELECT S.sname  
FROM sailor S, reserve R  
WHERE S.sid = R.sid AND R.bid = 103;
```

Table's alias name

Cartesian product

Filter/join condition

## SQL query: แสดงรายชื่อ กัลลาสีทั้งหมดที่เคย จองเรือหมายเลข 103

```
SELECT S.sname  
FROM sailor S, reserve R  
WHERE S.sid = R.sid AND R.bid = 103;
```

sname
Dustin
Lubber
Horatio
(3 rows)

```
WHERE S.sid = R.sid AND R.bid = 103;  
ERROR: operator does not exist: character varying = integer  
LINE 3: WHERE S.sid = R.sid AND R.bid = 103;  
          ^
```

HINT: No operator matches the given name and argument types. You might need to add explicit type casts.

```
SELECT S.sname  
FROM sailor S, reserve R  
WHERE S.sid = R.sid AND R.bid = '103';
```

Demo

ใน postgresql ใช้  
ผนทาง ห้ามใช้ พื้นหนู

# SQL query: แสดงรายชื่อ ก าล า สี ท ง ห မ ด ท ่ เ ค ย จ อง เ ร ื อ ห မ า ย เ ล ข 103

Relational Algebra:

$\pi_{\text{lname}} ((\sigma_{\text{bid}=103} \text{Reserves}) \bowtie_c \text{Sailors})$

<i>sid</i>	<i>bid</i>	<i>day</i>
22	103	10/08/14
31	103	11/06/14
74	103	09/08/14



<i>sid</i>	<i>sname</i>	<i>rating</i>	<i>age</i>
22	Dustin	7	45.0
29	Brutus	1	33.0
31	Lubber	8	55.5
32	Andy	8	25.5
58	Rusty	10	35.0
64	Horatio	7	35.0
71	Zorba	10	16.0
74	Horatio	9	35.0
85	Art	3	25.5
95	Bob	3	63.5

Result:

<i>sname</i>
Dustin
Lubber
Horatio

```
SELECT S.sname
FROM sailor S, reserve R
WHERE S.sid = R.sid AND
R.bid = '103';
```

Demo

# SQL query: แสดงรหัสของล่าสุดทั้งหมดที่

เคยจองเรือสีแดง

sailor

sid	sname	rating	age
22	Dustin	7	45.0
29	Brutus	1	33.0
31	Lubber	8	55.5
32	Andy	8	25.5
58	Rusty	10	35.0
64	Horatio	7	35.0
71	Zorba	10	16.0
74	Horatio	9	35.0
85	Art	3	25.5
95	Bob	3	63.5

Demo

boat

bid	bname	Color
101	Interlake	Blue
102	Interlake	Red
103	Clipper	Green
104	Marine	red

sid	bid	day
22	101	10/10/14
22	102	10/10/14
22	103	10/08/14
22	104	10/07/14
31	102	11/10/14
31	103	11/06/14
31	104	11/12/14
64	101	09/05/14
64	102	09/08/14
74	103	09/08/14

reserve

```
SELECT R.sid  
FROM boat B, reserve R  
WHERE R.bid = B.bid AND  
B.color = 'Red';
```

sid
-----
22
22
31
31
64
(5 rows)

ข้อมูลซ้ำอีกแล้ว  
แก้ยังไง

# SQL query: แสดงชื่อของกำล้าสีทึ้งหมดที่เคย

จองเรือสีแดง

sailor			
<i>sid</i>	<i>sname</i>	<i>rating</i>	<i>age</i>
22	Dustin	7	45.0
29	Brutus	1	33.0
31	Lubber	8	55.5
32	Andy	8	25.5
58	Rusty	10	35.0
64	Horatio	7	35.0
71	Zorba	10	16.0
74	Horatio	9	35.0
85	Art	3	25.5
95	Bob	3	63.5

<i>sid</i>	<i>bid</i>	<i>day</i>
22	101	10/10/14
22	102	10/10/14
22	103	10/08/14
22	104	10/07/14
31	102	11/10/14
31	103	11/06/14
31	104	11/12/14
64	101	09/05/14
64	102	09/08/14
74	103	09/08/14

boat		
<i>bid</i>	<i>bname</i>	<i>Color</i>
101	Interlake	Blue
102	Interlake	Red
103	Clipper	Green
104	Marine	red

# SQL query: แสดงชื่อของกัลารีทั้งหมดที่เคย จองเรือสีแดง

```
SELECT S.sname  
FROM sailor S, boat B, reserve R  
WHERE S.sid = R.sid AND R.bid = B.bid AND  
B.color = 'Red';
```

sname
Dustin
Dustin
Lubber
Lubber
Horatio
(5 rows)

ระวังมี row ซ้ำ  
ถ้าไม่ต้องการซ้ำ  
แก้ไขอย่างไร

Demo

# SQL query: แสดงสีของเรือทั้งหมดที่เคยถูกจองโดยกลาสี Lubber

sailor			
<i>sid</i>	<i>sname</i>	<i>rating</i>	<i>age</i>
22	Dustin	7	45.0
29	Brutus	1	33.0
31	Lubber	8	55.5
32	Andy	8	25.5
58	Rusty	10	35.0
64	Horatio	7	35.0
71	Zorba	10	16.0
74	Horatio	9	35.0
85	Art	3	25.5
95	Bob	3	63.5

<i>sid</i>	<i>bid</i>	<i>day</i>
22	101	10/10/14
22	102	10/10/14
22	103	10/08/14
22	104	10/07/14
31	102	11/10/14
31	103	11/06/14
31	104	11/12/14
64	101	09/05/14
64	102	09/08/14
74	103	09/08/14

boat

<i>bid</i>	<i>bname</i>	<i>Color</i>
101	Interlake	Blue
102	Interlake	Red
103	Clipper	Green
104	Marine	red

reserve

จาก Query ข้างต้น ต้องใช้ข้อมูลกี่ตาราง

# SQL query: แสดงสีของเรือทั้งหมดที่เคยถูกจองโดยกลาสี Lubber

```
SELECT B.color  
FROM sailor S, reserve R, boat B  
WHERE S.sname = 'Lubber' AND S.sid = R.sid  
AND R.bid = B.bid;
```

Demo

color
Red
Green
Red
(3 rows)

จำไว้ว่า  
ถ้าเราเลือก projection ของ attributes ที่  
ไม่ได้เป็น PK ของตารางผลลัพธ์  
โอกาสที่จะมี row ซ้ำแน่นอน  
ดังนั้นให้ใช้ SELECT DISTINCT เสมอ

# SQL query: แสดงชื่อของกำล้าสีทึ้งหมดที่เคยมีการจองเรืออย่างน้อย 1 ลำ

จาก Query ข้างต้น ถ้าเปลี่ยน sname เป็น sid จะได้ผลต่างกัน  
ใหม่อย่างไร

Demo

```
SELECT S.sname  
FROM sailor S, reserve R  
WHERE S.sid = R.sid;
```

sname
Dustin
Dustin
Dustin
Dustin
Lubber
Lubber
Lubber
Horatio
Horatio
Horatio
(10 rows)

```
SELECT DISTINCT S.sname  
FROM sailor S, reserve R  
WHERE S.sid = R.sid;
```

sname
Dustin
Horatio
Lubber

```
SELECT S.sid  
FROM sailor S, reserve R  
WHERE S.sid = R.sid;
```

sid
22
22
22
22
31
31
31
64
64
74

## SQL query: เปลี่ยนชื่อคอลัมน์ในการแสดงผลผ่าน AS และสืบค้นสตริงผ่าน LIKE

- AS ใช้เปลี่ยนชื่อคอลัมน์ที่ใช้แสดงผล
- LIKE ใช้ในการเทียบสตริงเพื่อดึงข้อมูล ทำ regular expression

```
SELECT S.sname, S.age, S.age-1 AS age1,  
       2*S.age AS age2  
  FROM sailor S  
 WHERE S.sname LIKE 'B%';
```

Col.  
ใหม่ ไม่มี  
ในตาราง  
ตั้งนับบัน

Demo

sname	age	age1	age2
Brutus	33.00	32.00	66.00
Bob	63.50	62.50	127.00
(2 rows)			

% อักขระใด ๆ จำนวน 0 อักขระหรือมากกว่า  
\_ อักขระใด ๆ จำนวน 1 ตัวเท่านั้น

## SQL query: เปลี่ยนชื่อคอลัมน์ในการแสดงผลผ่าน AS และสืบค้นสตริงผ่าน LIKE

```
SELECT S.sname, S.age, S.age-1 AS age1, 2*S.age AS age2  
FROM sailor S  
WHERE S.sname LIKE 'B__';
```

sname	age	age1	age2
Bob	63.50	62.50	127.00
(1 row)			

Brutus หายไปแล้ว

Demo

## SQL query: เปลี่ยนชื่อคอลัมน์ในการแสดงผลผ่าน AS และสีบคันสตริงผ่าน LIKE

```
SELECT S.sname, S.age, S.age-1 AS age1, 2*S.age AS age2  
FROM sailor S  
WHERE S.sname LIKE '%b';
```

sname	age	age1	age2
Bob	63.50	62.50	127.00
(1 row)			

Demo

จงแสดงชื่อเรื่องทั้งหมดที่มีคำ  
ว่า er ในชื่อ

จงแสดงชื่อภาษาสีทั้งหมดที่  
ลงท้ายด้วย y

```
select sname from sailor where sname like '%y';
```

```
sname
```

```
-----  
Andy
```

```
Rusty
```

```
(2 rows)
```

```
select bname from boat where bname like '%er%';
```

```
bname
```

```
-----  
Interlake
```

```
Interlake
```

```
Clipper
```

```
(3 rows)
```

Demo

## SQL query: จำกัดจำนวนรายการที่ต้องการ

```
SELECT S.sname, S.age  
FROM sailor S;
```

sname	age
Dustin	45.00
Lubber	55.50
Andy	25.50
Rusty	35.00
Horatio	35.00
Zorba	16.00
Horatio	35.00
Art	25.50
Bob	63.50
Brutus	33.00
(10 rows)	

```
SELECT S.sname, S.age  
FROM sailor S  
LIMIT 5;      LIMIT row_to_fetch
```

sname	age
Dustin	45.00
Lubber	55.50
Andy	25.50
Rusty	35.00
Horatio	35.00
(5 rows)	

Demo

## SQL query: จำกัดจำนวนรายการที่ต้องการ

```
SELECT S.sname, S.age  
FROM sailor S;
```

sname	age
Dustin	45.00
Lubber	55.50
Andy	25.50
Rusty	35.00
Horatio	35.00
Zorba	16.00
Horatio	35.00
Art	25.50
Bob	63.50
Brutus	33.00
(10 rows)	

```
SELECT S.sname, S.age  
FROM sailor S  
FETCH FIRST 5 ROW ONLY;
```

sname	age
Dustin	45.00
Lubber	55.50
Andy	25.50
Rusty	35.00
Horatio	35.00
(5 rows)	

Demo

Provide the same result as LIMIT but conforms to SQL standard

## SQL query: จำกัดจำนวนรายการที่ต้องการ

```
SELECT S.sname, S.age  
FROM sailor S;
```

sname	age
Dustin	45.00
Lubber	55.50
Andy	25.50
Rusty	35.00
Horatio	35.00
Zorba	16.00
Horatio	35.00
Art	25.50
Bob	63.50
Brutus	33.00

(10 rows)

```
SELECT S.sname, S.age  
FROM sailor S  
OFFSET 5;    OFFSET row_to_skip
```

sname	age
Zorba	16.00
Horatio	35.00
Art	25.50
Bob	63.50
Brutus	33.00

(5 rows)

Demo

## SQL query: จำกัดจำนวนรายการที่ต้องการ

```
SELECT S.sname, S.age  
FROM sailor S;
```

sname	age
Dustin	45.00
Lubber	55.50
Andy	25.50
Rusty	35.00
Horatio	35.00
Zorba	16.00
Horatio	35.00
Art	25.50
Bob	63.50
Brutus	33.00
(10 rows)	

```
SELECT S.sname, S.age  
FROM sailor S  
OFFSET 5 FETCH FIRST 3 ROW ONLY;
```

sname	age
Zorba	16.00
Horatio	35.00
Art	25.50
(3 rows)	

Demo



## SQL query: UNION, INTERSECT, EXCEPT

- เป็น operator ทำงานระหว่างตารางที่ compatible กัน
- ใน MySQL ไม่มี INTERSECT และ EXCEPT ให้ใช้
- แต่ PostgreSQL มี

# SQL query: แสดงชื่อของกลาสีทั้งหมดที่เคยจอง เรือสีเขียวหรือแดงก็ได้ (UNION)

sailor			
<i>sid</i>	<i>sname</i>	<i>rating</i>	<i>age</i>
22	Dustin	7	45.0
29	Brutus	1	33.0
31	Lubber	8	55.5
32	Andy	8	25.5
58	Rusty	10	35.0
64	Horatio	7	35.0
71	Zorba	10	16.0
74	Horatio	9	35.0
85	Art	3	25.5
95	Bob	3	63.5

<i>sid</i>	<i>bid</i>	<i>day</i>
22	101	10/10/14
22	102	10/10/14
22	103	10/08/14
22	104	10/07/14
31	102	11/10/14
31	103	11/06/14
31	104	11/12/14
64	101	09/05/14
64	102	09/08/14
74	103	09/08/14

reserve

<i>bid</i>	<i>bname</i>	<i>Color</i>
101	Interlake	Blue
102	Interlake	Red
103	Clipper	Green
104	Marine	red

boat

## SQL query: แสดงชื่อของล่าสุดทั้งหมดที่เคยจองเรือสีเขียวหรือแดงก็ได้

```
SELECT S.sid, S.sname  
FROM sailor S, boat B, reserve R  
WHERE S.sid = R.sid AND R.bid = B.bid AND  
(B.color = 'Red' OR B.color = 'Green');
```

sid	sname
22	Dustin
22	Dustin
22	Dustin
31	Lubber
31	Lubber
31	Lubber
64	Horatio
74	Horatio
(8 rows)	

ถ้าเขียนอย่างนี้ได้ผลอะไร

Demo

## SQL query: แสดงชื่อของล่าสุดทั้งหมดที่เคยจองเรือสีเขียวหรือแดงก็ได้โดยใช้ UNION

```
SELECT S.sid, S.sname  
FROM sailor S, boat B, reserve R  
WHERE S.sid = R.sid AND R.bid = B.bid AND  
B.color = 'Red'  
UNION  
SELECT S.sid, S.sname  
FROM sailor S, boat B, reserve R  
WHERE s.sid = R.sid AND R.bid = B.bid AND  
B.color = 'Green';
```

Set operator จะเอาสมาชิกที่  
ซ้ำออกโดยอัตโนมัติ

sid	sname
74	Horatio
31	Lubber
22	Dustin
64	Horatio
(4 rows)	

Demo

## SQL query: แสดงชื่อของกลาสีทั้งหมดที่เคยจอง เรือทั้งสีเขียวและสีแดง (INTERSECT??)

```
SELECT S.sid, S.sname
FROM sailor S, boat B, reserve R
WHERE S.sid = R.sid AND R.bid = B.bid AND
(B.color = 'Green' AND B.color = 'Red');
```

sid	sname
( 0 rows )	

ทำไมไม่มีผลเลย??

Demo

## SQL query: แสดงชื่อของล่าสิทั้งหมดที่เคยจองเรือทั้งสีเขียวและสีแดงโดยใช้ INTERSECT

```
SELECT S.sid, S.sname  
FROM sailor S, boat B, reserve R  
WHERE S.sid = R.sid AND R.bid = B.bid AND  
B.color = 'Red'
```

INTERSECT

```
SELECT S.sid, S.sname  
FROM sailor S, boat B, reserve R  
WHERE s.sid = R.sid AND R.bid = B.bid AND  
B.color = 'Green';
```

sid	sname
31	Lubber
22	Dustin
(2 rows)	

Demo

**SQL query:** แสดงชื่อของกลาสีทั้งหมดที่เคยจองเรือทั้งสีเขียวและสีแดง โดยใช้ **Nested Query** และ **IN**

```
SELECT S.sid, S.sname  
FROM sailor S, boat B, reserve R  
WHERE S.sid = R.sid AND R.bid = B.bid AND  
B.color = 'Red'  
AND S.sid IN  
(SELECT S.sid  
FROM sailor S, boat B, reserve R  
WHERE s.sid = R.sid AND R.bid = B.bid AND  
B.color = 'Green');
```

sid	sname
22	Dustin
22	Dustin
31	Lubber
31	Lubber
(4 rows)	

**Demo**

เราเรียกว่า **sub query** - หาผลของ **sub query** ก่อน

**SQL query:** แสดงชื่อของglassticทั้งหมดที่เคยจอง  
เรือสีเขียวแต่ไม่เคยจองสีแดงเลย

ใครช่วยตอบที

## SQL query: แสดงชื่อของล่าสุดทั้งหมดที่เคยจองเรือสีเขียวแต่ไม่เคยจองสีแดงเลย โดยใช้ Nested Query และ NOT IN

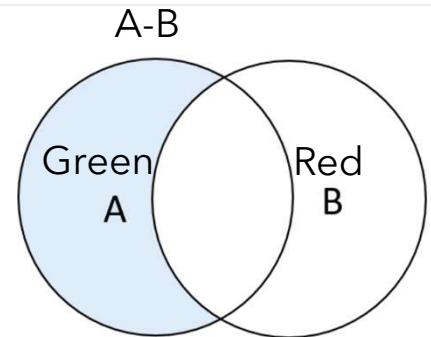
```
SELECT S.sid, S.sname  
FROM sailor S, boat B, reserve R  
WHERE S.sid = R.sid AND R.bid = B.bid AND  
B.color = 'Green'  
AND S.sid NOT IN  
(SELECT S.sid  
FROM sailor S, boat B, reserve R  
WHERE s.sid = R.sid AND R.bid = B.bid AND  
B.color = 'Red');
```

sid	sname
74	Horatio
(1 row)	

Demo

## SQL query: แสดงชื่อของล่าสุดทั้งหมดที่เคยจองเรือสีเขียวแต่ไม่เคยจองสีแดงเลย โดยใช้ EXCEPT

```
SELECT S.sid, S.sname  
FROM sailor S, boat B, reserve R  
WHERE S.sid = R.sid AND R.bid = B.bid AND  
B.color = 'Green'  
  
EXCEPT  
  
SELECT S.sid, S.sname  
FROM sailor S, boat B, reserve R  
WHERE s.sid = R.sid AND R.bid = B.bid AND  
B.color = 'Red';
```



sid	sname
74	Horatio
(1 row)	

Demo



## Aggregate operators

- COUNT(A) : นับจำนวนของค่าทั้งหมดในคอลัมน์ A
- SUM(A) : หาผลรวมของค่าทั้งหมดในคอลัมน์ A
- AVG(A) : หาค่าเฉลี่ยของค่าทั้งหมดในคอลัมน์ A
- MAX(A) : หาค่าที่มากที่สุดจากค่าทั้งหมดในคอลัมน์ A
- MIN(A) : หาค่าที่น้อยที่สุดจากค่าทั้งหมดในคอลัมน์ A

ถ้าใช้ DISTINCT ประกอบจะนับ  
แต่ค่าที่ไม่ซ้ำ

# SQL query: นับจำนวนกะลาสีทั้งหมดใน ตาราง sailor

sailor			
<i>sid</i>	<i>sname</i>	<i>rating</i>	<i>age</i>
22	Dustin	7	45.0
29	Brutus	1	33.0
31	Lubber	8	55.5
32	Andy	8	25.5
58	Rusty	10	35.0
64	Horatio	7	35.0
71	Zorba	10	16.0
74	Horatio	9	35.0
85	Art	3	25.5
95	Bob	3	63.5

```
SELECT COUNT(*)  
FROM sailor S;
```

```
count  
-----  
10  
(1 row)
```

ถ้าใช้คำสั่ง  
**SELECT COUNT(DISTINCT  
S.sname) FROM sailor S;**  
ได้จำนวนกะลาสีกี่คน??

Demo

## SQL query: หาค่าอายุรวมของล่าสีทุกคน ในตาราง sailor ที่มีค่า rating เท่ากับ 10

sailor			
<i>sid</i>	<i>sname</i>	<i>rating</i>	<i>age</i>
22	Dustin	7	45.0
29	Brutus	1	33.0
31	Lubber	8	55.5
32	Andy	8	25.5
58	Rusty	10	35.0
64	Horatio	7	35.0
71	Zorba	10	16.0
74	Horatio	9	35.0
85	Art	3	25.5
95	Bob	3	63.5

```
SELECT SUM(S.age)  
FROM sailor S  
WHERE S.rating = 10;
```

-----  
51.00  
(1 row)

Demo

# SQL query: หาค่าอายุเฉลี่ยของกะลาสีทุกคน ในตาราง sailor ที่มีค่า rating เท่ากับ 10

sailor			
<i>sid</i>	<i>sname</i>	<i>rating</i>	<i>age</i>
22	Dustin	7	45.0
29	Brutus	1	33.0
31	Lubber	8	55.5
32	Andy	8	25.5
58	Rusty	10	35.0
64	Horatio	7	35.0
71	Zorba	10	16.0
74	Horatio	9	35.0
85	Art	3	25.5
95	Bob	3	63.5

```
SELECT AVG(S.age)
FROM sailor S
WHERE S.rating = 10;
```

```
-----
25.500000000000000000
(1 row)
```

```
SELECT ROUND(AVG(S.age) , 2)
FROM sailor S
WHERE S.rating = 10;
```

```
-----
25.50
(1 row)
```

Demo

# SQL query: แสดงชื่อและอายุของกะลาสีที่มีอายุมากที่สุด

sid	sname	rating	age
22	Dustin	7	45.0
29	Brutus	1	33.0
31	Lubber	8	55.5
32	Andy	8	25.5
58	Rusty	10	35.0
64	Horatio	7	35.0
71	Zorba	10	16.0
74	Horatio	9	35.0
85	Art	3	25.5
95	Bob	3	63.5

```
SELECT S.sname, MAX(S.age)  
FROM sailor S;
```

```
mysql> SELECT S.sname, MAX(S.age)  
-> FROM sailor S;  
+-----+-----+  
| sname | MAX(S.age) |  
+-----+-----+  
| Dustin | 63.5 |  
+-----+-----+  
1 row in set (0.00 sec)
```

ใครช่วยตอบที่  
TikTok TikTok Tiktok

Demo

```
bongo=# select s.sname, max(s.age) from sailor S;  
ERROR: column "s.sname" must appear in the GROUP BY clause or be used in an aggregate function  
LINE 1: select s.sname, max(s.age) from sailor S;  
^
```

## SQL query: แสดงชื่อและอายุของกะลาสีที่มี อายุมากที่สุด

```
SELECT S.sname, S.age  
FROM sailor S  
WHERE S.age = (SELECT  
MAX(S2.age) FROM sailor S2);
```

sname	age
Bob	63.50
(1 row)	



จะมีผลลัพธ์มากกว่า 1 คนได้  
ไหม

Demo

# SQL query: เปลี่ยนอายุของ Brutus เป็น 63.5

<i>sid</i>	<i>sname</i>	<i>rating</i>	<i>age</i>
22	Dustin	7	45.0
29	Brutus	1	33.0
31	Lubber	8	55.5
32	Andy	8	25.5
58	Rusty	10	35.0
64	Horatio	7	35.0
71	Zorba	10	16.0
74	Horatio	9	35.0
85	Art	3	25.5
95	Bob	3	63.5

UPDATE *sailor*

SET *age* = 63.5

WHERE *sid* = '29';

SELECT \* FROM *sailor*;

<i>sid</i>	<i>sname</i>	<i>rating</i>	<i>age</i>
22	Dustin	7	45.00
31	Lubber	8	55.50
32	Andy	8	25.50
58	Rusty	10	35.00
64	Horatio	7	35.00
71	Zorba	10	16.00
74	Horatio	9	35.00
85	Art	3	25.50
95	Bob	3	63.50
29	Brutus	1	63.50

(10 rows)

SELECT *S.sname*, *S.age*  
FROM *sailor S*  
WHERE *S.age* = (SELECT  
MAX(*S2.age*) FROM  
*sailor S2*);

<i>sname</i>	<i>age</i>
Bob	63.50
Brutus	63.50
(2 rows)	

Demo

จงแสดงชื่อคลาสีที่มีอายุน้อยสุด

## BETWEEN and AND

- ทั้ง BETWEEN และ AND ใช้ในการระบุช่วงของค่า
- ช่วงของค่าอาจเป็น ตัวเลข (number) ตัวหนังสือ (text)  
หรือ วันที่ (Date) ก็ได้

## SQL query: เปลี่ยนอายุของ Brutus เป็น 33.0 เหมือนเดิม

```
UPDATE sailor  
SET age = 33.0  
WHERE sid = '29';
```

Demo

## SQL query: หาชื่อของล่าสีทั้งหมดที่มีอายุระหว่าง

25 ถึง 35 ปี

ไม่ใช้ชื่อย่ออย่าง S เพื่อแทน  
sailor ก็ได้ แต่ใช้ใน query ยาก ๆ  
ก็สะดวกขึ้น

sid	sname	rating	age
22	Dustin	7	45.0
29	Brutus	1	33.0
31	Lubber	8	55.5
32	Andy	8	25.5
58	Rusty	10	35.0
64	Horatio	7	35.0
71	Zorba	10	16.0
74	Horatio	9	35.0
85	Art	3	25.5
95	Bob	3	63.5

```
SELECT sname  
FROM sailor  
WHERE age BETWEEN 25 AND 35;
```

sid	sname
32	Andy
58	Rusty
64	Horatio
74	Horatio
85	Art
29	Brutus
(6 rows)	

BETWEEN and AND นี้เป็น<sup>๊</sup>  
inclusive หรือ exclusive??

Demo

```
SELECT sid, sname  
FROM sailor  
WHERE age=25.5;
```

แล้ว Query นี้ทำอะไร

```
SELECT COUNT(*)  
FROM sailor  
WHERE age=25.5;
```

```
SELECT *  
FROM sailor  
WHERE sname NOT BETWEEN 'Hansen'  
AND 'Pettersen';
```

SELECT 'Hansen' < 'Bob';  
SELECT 'Bob' < 'Hansen';  
SELECT 'Hansen' < 'Hansen1';

```
SELECT *  
FROM sailor  
WHERE sname LIKE 'A%';
```

เปรียบเทียบที่ละตัวอักษรก่อน  
จากนั้นดูความยาว

Demo

```
SELECT SUM(age)  
FROM sailor  
WHERE age>20;
```

แล้ว Query นี้ทำอะไร

```
SELECT MIN(age)  
FROM sailor  
WHERE age>20;
```

```
SELECT S.sname, MAX(S.age)  
FROM sailor;
```



Demo

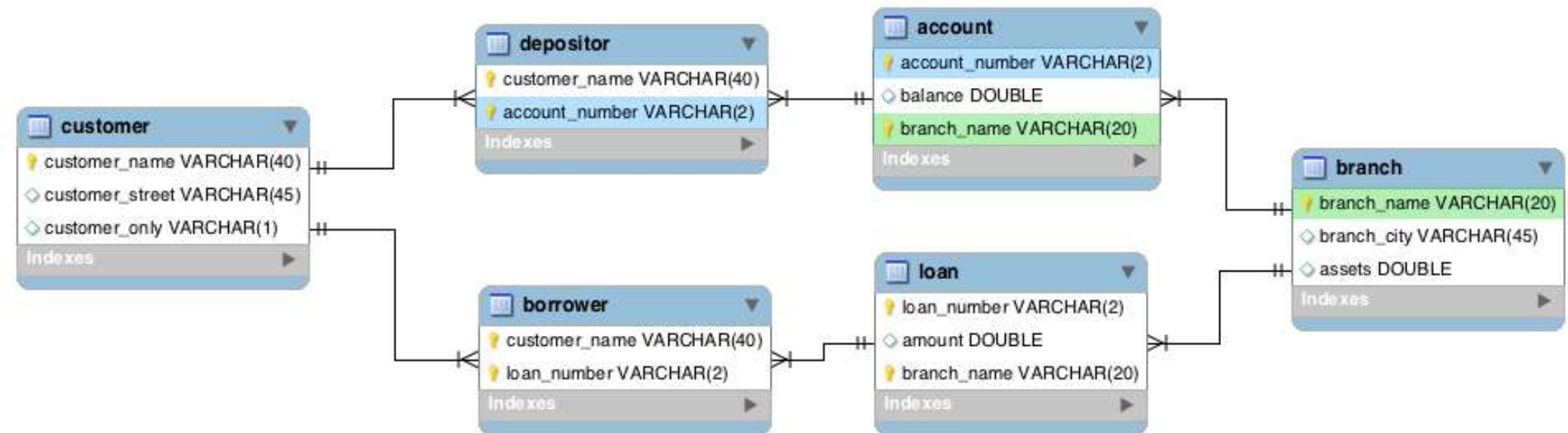


# ถึงจุดนี้เรารีบอนอะไรไปบ้าง

SELECT	target-list	ใช้เลือก attributes (+) aggregate functions เพื่อแสดงผล
FROM	relation-list	ใช้ระบุตารางที่เกี่ยวข้องกับการดึงข้อมูล
WHERE	qualification	ใช้ระบุเงื่อนไขต่างๆ ในการดึงข้อมูล

GROUP BY	ใช้รวมกลุ่มข้อมูลที่สืบคันมาได้
HAVING	ใช้ระบุเงื่อนไขในการคัดกรองกลุ่มข้อมูลหลัง group มาแล้ว
ORDER BY	ใช้จัดลำดับของรายการในการแสดงผล

# Banking EER diagram



## Banking's RDBMS schema design

- branch(branch name, branch\_city, branch\_assets)
- customer(customer name, customer\_street, customer\_only)
- account(account number, branch\_name, balance)
- loan(loan number, branch\_name, amount)
- depositor(customer name, account number)
- borrower(customer name, loan number)

จากนี้เราจะ CREATE TABLE  
branch, customer,  
account, loan, depositor,  
borrower  
(ดูคำสั่งในไฟล์ banking.txt)

```
CREATE TABLE branch (
    branch_name varchar(20) NOT NULL,
    branch_city varchar(45) DEFAULT NULL,
    assets numeric(10,2) DEFAULT NULL,
    PRIMARY KEY (branch_name)
);
```

```
CREATE TABLE customer (
    customer_name varchar(40) NOT NULL,
    customer_street varchar(45) DEFAULT NULL,
    customer_only varchar(1) DEFAULT NULL,
    PRIMARY KEY (customer_name)
);
```

```
CREATE TABLE account (
    account_number varchar(2) NOT NULL,
    branch_name varchar(45) DEFAULT NULL,
    balance numeric(10,2) DEFAULT NULL,
    PRIMARY KEY (account_number)
);
```

```
CREATE TABLE loan (
    loan_number varchar(2) NOT NULL,
    branch_name varchar(45) DEFAULT NULL,
    amount numeric(10,2) DEFAULT NULL,
    PRIMARY KEY (loan_number)
);
```

```
CREATE TABLE borrower (
    customer_name varchar(45) NOT NULL,
    loan_number varchar(2) NOT NULL,
    PRIMARY KEY (customer_name, loan_number)
);
```

```
CREATE TABLE depositor (
    customer_name varchar(45) NOT NULL,
    account_number varchar(2) NOT NULL,
    PRIMARY KEY (customer_name,account_number)
);
```

Demo

จากนี้เราจะ **INSERT** ข้อมูลในตาราง  
branch, customer,  
account, loan, depositor,  
borrower  
(ดูคำสั่งในไฟล์ banking.txt)

Demo

## INSERT

```
INSERT INTO table_name (column1, column2, column3, ...)  
VALUES (value1, value2, value3, ...);
```

```
INSERT INTO table_name  
VALUES (value1, value2, value3, ...);
```

- INSERT INTO account (account\_number, branch\_name, balance)  
VALUES  
('1', 'B', 100.00),  
('2', 'A', 50.00);
- INSERT INTO account  
VALUES  
('1', 'B', 100.00),  
('2', 'A', 50.00);

DELETE

DELETE FROM *table\_name* WHERE *condition*;

- DELETE FROM account WHERE account\_number='1';

UPDATE

```
UPDATE table_name
SET column1 = value1, column2 = value2, ...
WHERE condition;
```

- UPDATE account  
SET balance=500.00  
WHERE account\_number = '1';

branch

Branch_name	Branch_city	assets
A	Riverside	\$10,000
B	LA	\$20,000
C	Long Beach	\$15,000
D	Irvine	\$12,000
E	Pomona	\$7,000
F	San Jose	\$18,000

customer

Customer_name	Customer_street	Customer_only
Joe	Joe_street	Y
Alan	Mary_street	Y
Jason	Jason_street	N
Mary	Mary_street	N
Mike	Mary_street	Y
Keith	Keith_street	N

depositor

Customer_name	Account_number
Joe	1
Joe	2
Mary	2
Keith	4
Mike	5
Keith	6
Joe	3

account

Account_number	Branch_name	balance
1	B	\$100
2	A	\$50
3	A	\$30
4	F	\$120
5	A	\$500
6	B	\$324

borrower

Customer_name	Loan_number
Joe	1
Jason	2
Joe	3
Keith	4
Mary	5
Joe	6

loan

Loan_number	Branch_name	Amount
1	B	\$100
2	E	\$27
3	F	\$543
4	A	\$129
5	A	\$26
6	B	\$67

# SQL query: แสดงเลขที่บัญชีเงินฝากทั้งหมดที่เปิดในเมือง Riverside

```
SELECT A.account_number  
FROM account A, branch B  
WHERE A.branch_name = B.branch_name  
AND B.branch_city = 'Riverside';
```

account	Account_number	Branch_name	balance
	1	B	\$100
	2	A	\$50
	3	A	\$30
	4	F	\$120
	5	A	\$500
	6	B	\$324

account_number
-----
2
3
5
(3 rows)

branch	Branch_name	Branch_city	assets
	A	Riverside	\$10,000
	B	LA	\$20,000
	C	Long Beach	\$15,000
	D	Irvine	\$12,000
	E	Pomona	\$7,000
	F	San Jose	\$18,000

# SQL query: แสดงเลขที่บัญชีเงินฝากทั้งหมดที่เปิดในเมือง Riverside

```
SELECT account_number  
FROM account  
WHERE branch_name IN  
(SELECT branch_name  
FROM branch  
WHERE branch_city = 'Riverside');
```

account_number
2
3
5

(3 rows)

เขียน Query โดยใช้ IN ก็ได้ผล  
แบบเดียวกัน  
อาจจะใช้เวลาประมวลผลน้อย  
กว่าถ้า cartesian product ใหญ่  
มาก

IN ต่างจาก = อย่างไร

```
SELECT S.sname, S.age  
FROM sailor S  
WHERE S.age =  
(SELECT MAX(S2.age)  
FROM sailor S2);
```

กรณี = จากตาราง sub query  
แบบนี้ ต้องระวังว่าค่าในตารางต้องเป็น  
ค่าเดียวเท่านั้น

# SQL query: แสดงเลขที่บัญชีเงินฝากทั้งหมด ของสาขา 'A' และ 'B'

```
SELECT account_number  
FROM account  
WHERE branch_name IN ('A', 'B');
```

account_number
-----
1
2
3
5
6
(5 rows)

account	Account_number	Branch_name	balance
1	1	B	\$100
2	2	A	\$50
3	3	A	\$30
5	4	F	\$120
6	5	A	\$500
	6	B	\$324

## SQL query: EXISTS และ NOT EXISTS

- EXISTS predicate เป็น \*จริง\* ก็ต่อเมื่อ sub-query ไม่เป็นเท็จต่อว่าง
- NOT EXISTS predicate เป็น \*จริง\* ก็ต่อเมื่อ sub-query เป็นเท็จต่อว่าง

NOT EXISTS ใช้ทดแทน MINUS operator ใน relational algebra  
ได้อย่างไร

## SQL query: แสดงเลขที่บัญชีเงินฝากทั้งหมดที่เปิด ในเมือง Riverside อีกครั้งโดยใช้ EXISTS

```
SELECT A.account_number
FROM account A
WHERE EXISTS (SELECT *
               FROM branch B
              WHERE B.branch_city = 'Riverside'
                AND B.branch_name = A.branch_name);
```

account_number
2
3
5
(3 rows)

account		
Account_number	Branch_name	balance
1	B	\$100
2	A	\$50
3	A	\$30
4	F	\$120
5	A	\$500
6	B	\$324

branch		
Branch_name	Branch_city	assets
A	Riverside	\$10,000
B	LA	\$20,000
C	Long Beach	\$15,000
D	Irvine	\$12,000
E	Pomona	\$7,000
F	San Jose	\$18,000

# SQL query: แสดง account number ทั้งหมดที่ ไม่ได้อยู่ในสาขา Riverside โดยใช้ **NOT EXISTS**

```
SELECT A.account_number  
FROM account A  
WHERE NOT EXISTS (SELECT *  
                   FROM branch B  
                   WHERE B.branch_city = 'Riverside'  
                   AND B.branch_name = A.branch_name);
```

account_number
-----
1
4
6
(3 rows)

account		
Branch_name	balance	
B	\$100	
A	\$50	
A	\$30	
F	\$120	
A	\$500	
B	\$324	

branch		
Branch_name	Branch_city	assets
A	Riverside	\$10,000
B	LA	\$20,000
C	Long Beach	\$15,000
D	Irvine	\$12,000
E	Pomona	\$7,000
F	San Jose	\$18,000

## Subquery predicates

### Quantified comparison predicates

A *quantified predicate* compares a simple value of an expression with the result of a *Subquery*.

- Given a comparison operation  $\theta$ , representing some operator in the set  $\{<, \leq, =, >, \geq\}$ , the equivalent predicates '*expr*  $\theta$  *SOME (Subquery)*' and '*expr*  $\theta$  *ANY (Subquery)*' are TRUE if and only if, for at least one element *s* returned by the *Subquery*, it is true that '*expr*  $\theta$  *s*' is TRUE.
- the predicate '*expr*  $\theta$  *ALL (Subquery)*' is TRUE if and only if '*expr*  $\theta$  *s*' is TRUE for every one of the elements *s* of the *Subquery*;

## SQL query: หาเลขที่บัญชีเงินฝากที่มี balance น้อยสุด

```
SELECT account_number  
FROM account  
WHERE balance <= ALL (SELECT  
    balance from account);
```

account	Account_number	Branch_name	balance
	1	B	\$100
	2	A	\$50
	3	A	\$30
	4	F	\$120
	5	A	\$500
	6	B	\$324

account_number
-----
3
(1 row)

expr θ ALL (Subquery)

balance <= ALL (SELECT  
 balance from account)

- 1) หา subquery table ก่อน
- 2) หยิบ mainquery table มาแล้วเทียบ expr θ (<=)  
กับทุกตัวใน 1)

## Mainquery Table

	account_number [PK] character varying (2)	branch_name character varying (45)	balance numeric (10,2)
1	1	B	100.00
2	2	A	50.00
3	3	A	30.00
4	4	F	120.00
5	5	A	500.00
6	6	B	324.00

```

SELECT account_number
FROM account
WHERE balance <= ALL(SELECT
    balance from account);

```

Subquery Table

	balance numeric (10,2)
1	100.00
2	50.00
3	30.00
4	120.00
5	500.00
6	324.00

Subquery Table

	balance numeric (10,2)
1	100.00
2	50.00
3	30.00
4	120.00
5	500.00
6	324.00

## SQL query: หา balances ของบัญชี เงินฝากทั้งหมดในเมือง Riverside โดยใช้ ANY

```
SELECT balance
FROM account
WHERE branch_name = ANY (SELECT
    B.branch_name
    FROM branch B
    WHERE branch_city ='Riverside');
```

account	Account_number	Branch_name	balance
	1	B	\$100
	2	A	\$50
	3	A	\$30
	4	F	\$120
	5	A	\$500
	6	B	\$324

```
balance
-----
50.00
30.00
500.00
(3 rows)
```

## SQL query: หา balances ของบัญชี เงินฝากทั้งหมดในเมือง Riverside โดยใช้ SOME

```

SELECT balance
FROM account
WHERE branch_name = SOME (SELECT
    B.branch_name
    FROM branch B
    WHERE branch_city = 'Riverside');
  
```

Account_number	Branch_name	balance
1	B	\$100
2	A	\$50
3	A	\$30
4	F	\$120
5	A	\$500
6	B	\$324

balance
-----
50.00
30.00
500.00
(3 rows)

## SQL query: Aggregate functions (revisited)

หา balance รวมของทุกบัญชีเงินฝากในเมือง  
**Riverside**

```
SELECT SUM(A.balance) AS total_amount
FROM account A, branch B
WHERE B.branch_city = 'Riverside' AND
A.branch_name = B.branch_name;
```

total_amount
-----
580.00
(1 row)

account		
Account_number	Branch_name	balance
1	B	\$100
2	A	\$50
3	A	\$30
4	F	\$120
5	A	\$500
6	B	\$324

## SQL query: Aggregate functions (revisited)

หาจำนวนบัญชีเงินฝากทั้งหมด

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

หรือ

```
SELECT COUNT(account_number)  
      FROM account;
```

----  
6  
(1 row)

account

Account_number	Branch_name	balance
1	B	\$100
2	A	\$50
3	A	\$30
4	F	\$120
5	A	\$500
6	B	\$324



# การจัดกลุ่มข้อมูลด้วย Group by

---

## ตัวอย่างการจัดกลุ่มข้อมูล และหาค่าของกลุ่มด้วย Aggregation function

account

Account_number	Branch_name	balance
1	B	\$100
2	A	\$50
3	A	\$30
4	F	\$120
5	A	\$500
6	B	\$324

- จัดกลุ่มตามสาขา แล้วหา `Count()` ผลรวมจำนวนบัญชีของแต่ละกลุ่ม
- จัดกลุ่มตามสาขา แล้วหา `Sum()` ยอดเงินฝากของแต่ละกลุ่ม

`SELECT branch_name, count(*) FROM account GROUP BY branch_name`

`SELECT branch_name, sum(balance) FROM account GROUP BY branch_name`

Attribute ที่อยู่หลัง `GROUP BY` ต้องปรากฏใน `select` ด้วยนะครับ

## SQL query: รวมกลุ่มรายการที่ค้นหาได้โดยใช้ GROUP BY หา balance รวมของบัญชีเงินฝาก แยกตามสาขา

```
SELECT branch_name, SUM(balance)  
FROM account  
GROUP BY branch_name;
```

branch_name	sum
B	424.00
F	120.00
A	580.00
(3 rows)	

Account_number	Branch_name	balance
1	B	\$100
2	A	\$50
3	A	\$30
4	F	\$120
5	A	\$500
6	B	\$324

## SQL query: รวมกลุ่มรายการที่ค้นหาได้โดยใช้ GROUP BY หา balance รวมของบัญชีเงินฝาก แยกตามสาขา

```
SELECT account_number, branch_name, SUM(balance)  
FROM account  
GROUP BY branch_name;
```

```
ERROR: column "account.account_number" must appear in the GROUP BY clause or be used in an aggregate function  
LINE 1: SELECT account_number, branch_name, SUM(balance)
```

- ข้อแนะนำเบื้องต้น
- SELECT ต้องเลือกเฉพาะ column ที่ใช้จัดกลุ่มและ Aggregate function ของแต่ละกลุ่ม
- หลัง GROUP BY ต้องเป็น column ที่ใช้จัดกลุ่ม

# SQL query: รวมกลุ่มรายการที่ค้นหาได้โดยใช้ GROUP BY แสดงชื่อและ balance รวมของลูกค้าเงินฝากแต่ราย

**customer** Customer บังคนอาจะไม่ฝากเงินเดบ

Customer_name	Customer_street	Customer_only
Joe	Joe_street	Y
Alan	Mary_street	Y
Jason	Jason_street	N
Mary	Mary_street	N
Mike	Mary_street	Y
Keith	Keith_street	N

**depositor**

Customer_name	Account_number
Joe	1
Joe	2
Mary	2
Keith	4
Mike	5
Keith	6
Joe	3

**account**

Account_number	Branch_name	balance
1	B	\$100
2	A	\$50
3	A	\$30
4	F	\$120
5	A	\$500
6	B	\$324

```
SELECT C.customer_name, SUM(A.balance)
FROM customer C, account A, depositor D
WHERE C.customer_name = D.customer_name AND
D.account_number = A.account_number
GROUP BY C.customer_name;
```

customer_name	sum
Joe	180.00
Mary	50.00
Mike	500.00
Keith	444.00
(4 rows)	

## การกรองการจัดกลุ่ม

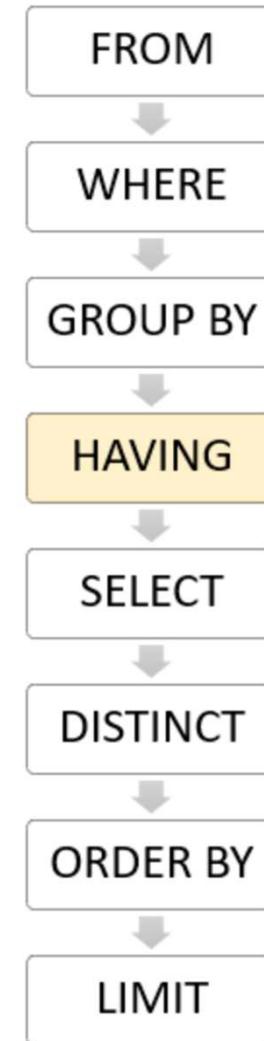
- To eliminate rows from the result of a select statement where a *GROUP BY* clause appears we use the *HAVING* clause, which is evaluated after the *GROUP BY*.
- For example, the query:

```
SELECT account_branch, SUM(balance) FROM account  
GROUP BY account_branch HAVING SUM(balance)>1000.
```

will print the account branches and total balances for every branch where the total account balance exceeds 1000.

- The *HAVING* clause can only apply tests to values that are single-valued for groups in the SELECT statement.
- The *HAVING* clause can have a nested subquery, just like the *WHERE* clause

# ลำดับของการ execute ส่วนของคำสั่งใน PostgreSQL



## SQL query: คัดกรองผลของการรวมกลุ่มโดย GROUP BY โดยเฉพาะที่ผ่านเงื่อนไขของ HAVING

แสดงชื่อและ balance รวมของลูกค้าเงินฝากแต่รายโดยแสดงเฉพาะลูกค้าที่มี balance รวมมากกว่า 100

```
SELECT C.customer_name, SUM(A.balance) AS sum_balance  
FROM customer C, account A, depositor D  
WHERE C.customer_name = D.customer_name AND  
D.account_number = A.account_number  
GROUP BY C.customer_name  
HAVING SUM(A.balance) > 100;
```

ข้อมูล Mary หายไป เพราะ  
balance รวมของ  
Mary (50) < 100

customer_name	sum_balance
Joe	180.00
Mike	500.00
Keith	444.00
(3 rows)	

## SQL query: คัดกรองผลของการรวมกลุ่มโดย GROUP BY โดย เอาเฉพาะที่ผ่านเงื่อนไขของ HAVING

แสดงชื่อและ balance รวมของลูกค้าเงินฝากแต่รายโดยแสดงเฉพาะลูกค้าที่มีอย่างน้อย 2 accounts

```
SELECT C.customer_name, SUM(A.balance) AS sum_balance
FROM customer C, account A, depositor D
WHERE C.customer_name = D.customer_name AND
      D.account_number = A.account_number
GROUP BY C.customer_name
HAVING COUNT(C.customer_name) > 1;
```

How about count(\*) > 1?

customer_name	sum_balance
Joe	180.00
Keith	444.00
(2 rows)	

customer		
Customer_name	Customer_street	Customer_only
Joe	Joe_street	Y
Alan	Mary_street	Y
Jason	Jason_street	N
Mary	Mary_street	N
Mike	Mary_street	Y
Keith	Keith_street	N

depositor	
Customer_name	Account_number
Joe	1
Joe	2
Mary	2
Keith	4
Mike	5
Keith	6
Joe	3

account		
Account_number	Branch_name	balance
1	B	\$100
2	A	\$50
3	A	\$30
4	F	\$120
5	A	\$500
6	B	\$324

customer		
Customer_name	Customer_street	Customer_only
Joe	Joe_street	Y
Alan	Mary_street	Y
Jason	Jason_street	N
Mary	Mary_street	N
Mike	Mary_street	Y
Keith	Keith_street	N

depositor	
Customer_name	Account_number
Joe	1
Joe	2
Mary	2
Keith	4
Mike	5
Keith	6
Joe	3

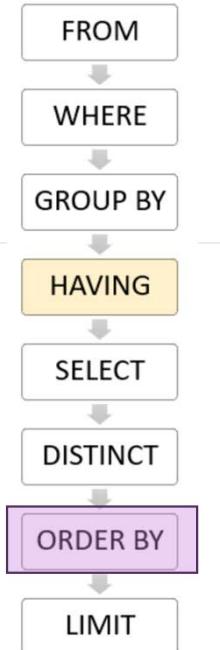
account		
Account_number	Branch_name	balance
1	B	\$100
2	A	\$50
3	A	\$30
4	F	\$120
5	A	\$500
6	B	\$324

## SQL query: จัดลำดับการแสดงผลโดยใช้ ORDER BY

แสดงชื่อและ balance รวมของลูกค้าเงินฝากแต่รายโดยแสดงเฉพาะลูกค้าที่มีอย่างน้อย 2 accounts และเรียงลำดับ balance รวมจากมากไปน้อย

```
SELECT C.customer_name, SUM(A.balance) AS sum_balance
FROM customer C, account A, depositor D
WHERE C.customer_name = D.customer_name AND
      D.account_number = A.account_number
GROUP BY C.customer_name
HAVING COUNT(C.customer_name) > 1
ORDER BY sum_balance DESC;
```

↑  
จากมากไปน้อย



customer_name	sum_balance
Keith	444.00
Joe	180.00
(2 rows)	

## เพิ่มข้อมูล 1 รายการลงในตาราง account

แกง่วง

SELECT \* FROM account; => ลองเรียกคำสั่งนี้ทั้งก่อนและหลัง INSERT

```
INSERT INTO account (account_number, branch_name, balance)  
VALUES ('7', 'X', 222);
```

ถ้าเรียกสั่งข้างต้นอีกครั้งโดยเปลี่ยนเฉพาะค่า balance จะเกิดอะไรขึ้น

```
INSERT INTO account (account_number, branch_name, balance)  
VALUES ('7', 'X', 225);
```

```
ERROR: duplicate key value violates unique constraint "account_pkey"  
DETAIL: Key (account_number)=(7) already exists.
```



# การ JOIN ตาราง

---

## SQL query: JOIN

account

Account_number	Branch_name	balance
1	B	\$100
2	A	\$50
3	A	\$30
4	F	\$120
5	A	\$500
6	B	\$324
7	X	\$222

branch

Branch_name	Branch_city	assets
A	Riverside	\$10,000
B	LA	\$20,000
C	Long Beach	\$15,000
D	Irvine	\$12,000
E	Pomona	\$7,000
F	San Jose	\$18,000

```
SELECT *
FROM account A JOIN branch B
ON A.branch_name = B.branch_name;
```

```
SELECT *
FROM account A, branch B
WHERE A.branch_name = B.branch_name;
```

มี join ประเภทใดใน relational algebra?  
e.g., Condition join,  
Equijoin  
Natural join

# SQL query: JOIN

account

Account_number	Branch_name	balance
1	B	\$100
2	A	\$50
3	A	\$30
4	F	\$120
5	A	\$500
6	B	\$324
7	X	\$222

branch

Branch_name	Branch_city	assets
A	Riverside	\$10,000
B	LA	\$20,000
C	Long Beach	\$15,000
D	Irvine	\$12,000
E	Pomona	\$7,000
F	San Jose	\$18,000

Duplicate columns

account_number	branch_name	balance	branch_name	branch_city	assets
1	B	100.00	B	LA	20000.00
2	A	50.00	A	Riverside	100000.00
3	A	30.00	A	Riverside	100000.00
4	F	120.00	F	San Jose	18000.00
5	A	500.00	A	Riverside	100000.00
6	B	324.00	B	LA	20000.00

(6 rows)

$\sigma_{Account.branch\_name=Branch.branch\_name} (Account \times Branch)$

$$r \bowtie_c s = \sigma_c(r \times s)$$

# SQL query: ลองใช้ NATURAL JOIN

```
SELECT *
FROM account NATURAL JOIN branch;
```

JOIN โดยใช้ key ร่วมคือ  
branch\_name และเอาคอลัมน์  
branch\_name ที่ซ้ำออกโดย  
อัตโนมัติ

branch_name	account_number	balance	branch_city	assets
B	1	100.00	LA	20000.00
A	2	50.00	Riverside	100000.00
A	3	30.00	Riverside	100000.00
F	4	120.00	San Jose	18000.00
A	5	500.00	Riverside	100000.00
B	6	324.00	LA	20000.00

(6 rows)

Account  Branch

# SQL query: ลองใช้ LEFT OUTER JOIN

```
SELECT *
FROM account A LEFT JOIN branch B
ON A.branch_name = B.branch_name;
```

LEFT JOIN / RIGHT JOIN ใช้  
ทำอะไรได้บ้าง?

account_number	branch_name	balance	branch_name	branch_city	assets
1	B	100.00	B	LA	20000.00
2	A	50.00	A	Riverside	100000.00
3	A	30.00	A	Riverside	100000.00
4	F	120.00	F	San Jose	18000.00
5	A	500.00	A	Riverside	100000.00
6	B	324.00	B	LA	20000.00
7	X	222.00			

(7 rows)

ใช้ LEFT JOIN หรือ LEFT OUTER JOIN ก็ได้

## SQL query: ลองใช้ LEFT OUTER JOIN

```
SELECT *
FROM account A LEFT JOIN branch B
ON A.branch_name = B.branch_name
WHERE B.branch_name is NULL;
```

account_number	branch_name	balance	branch_name	branch_city	assets
7	x	222.00			
(1 row)					

## SQL query: ลองใช้ RIGHT OUTER JOIN

```
SELECT *
FROM account A RIGHT JOIN branch B
ON A.branch_name = B.branch_name;
```

account_number	branch_name	balance	branch_name	branch_city	assets
1	B	100.00	B	LA	20000.00
2	A	50.00	A	Riverside	100000.00
3	A	30.00	A	Riverside	100000.00
4	F	120.00	F	San Jose	18000.00
5	A	500.00	A	Riverside	100000.00
6	B	324.00	B	LA	20000.00
			E	Pomona	7000.00
			C	Long Beach	15000.00
			D	Irvine	12000.00

(9 rows)

ใช้ RIGHT JOIN หรือ RIGHT OUTER JOIN ก็ได้

## SQL query: ลองใช้ FULL OUTER JOIN

```
SELECT *
FROM account A FULL OUTER JOIN branch B
ON A.branch_name = B.branch_name;
```

account_number	branch_name	balance	branch_name	branch_city	assets
1	B	100.00	B	LA	20000.00
2	A	50.00	A	Riverside	100000.00
3	A	30.00	A	Riverside	100000.00
4	F	120.00	F	San Jose	18000.00
5	A	500.00	A	Riverside	100000.00
6	B	324.00	B	LA	20000.00
7	X	222.00	E	Pomona	7000.00
			C	Long Beach	15000.00
			D	Irvine	12000.00

## SQL query: ลองใช้ FULL OUTER JOIN

```
SELECT *
FROM account A FULL OUTER JOIN branch B
ON A.branch_name = B.branch_name
ORDER BY A.account_number NULLS FIRST; Or NULLS LAST
```

	account_number	branch_name	balance	branch_name	branch_city	assets
				E	Pomona	7000.00
				D	Irvine	12000.00
				C	Long Beach	15000.00
1		B	100.00	B	LA	20000.00
2		A	50.00	A	Riverside	100000.00
3		A	30.00	A	Riverside	100000.00
4		F	120.00	F	San Jose	18000.00
5		A	500.00	A	Riverside	100000.00
6		B	324.00	B	LA	20000.00
7		X	222.00			

(10 rows)

## SQL query: ใน MySQL ไม่มี FULL OUTER JOIN แต่ใช้ UNION ทดแทนได้ ลองมาเขียนใน PostgreSQL

```
SELECT *
FROM account A LEFT JOIN branch B
ON A.branch_name = B.branch_name
UNION
SELECT *
FROM account A RIGHT JOIN branch B
ON A.branch_name = B.branch_name;
```

# SQL query: ใน MySQL ไม่มี FULL OUTER JOIN แต่ใช้ UNION ทดแทนได้ ลองมาเขียนใน PostgreSQL

account_number	branch_name	balance	branch_name	branch_city	assets
6	B	324.00	C	Long Beach	15000.00
4	F	120.00	B	LA	20000.00
			F	San Jose	18000.00
			D	Irvine	12000.00
1	B	100.00	B	LA	20000.00
3	A	30.00	A	Riverside	100000.00
			E	Pomona	7000.00
5	A	500.00	A	Riverside	100000.00
2	A	50.00	A	Riverside	100000.00
7	X	222.00			

1. ควรใช้วิธีนี้ก็ไม่ผิดอะไร แต่ตัวตรวจก็ต้องเอาผลมา sort ก่อน ยกเว้นใจที่กำหนด output ในลักษณะจำเพาะ หรือมีการวัดประสิทธิภาพของ query ว่าทำงานเร็วข้า กิน memory มากน้อย

## LEFT JOIN



Everything on the left  
+  
anything on the right that matches

```
SELECT *  
FROM TABLE_1  
LEFT JOIN TABLE_2  
ON TABLE_1.KEY = TABLE_2.KEY
```

## RIGHT JOIN



Everything on the right  
+  
anything on the left that matches

```
SELECT *  
FROM TABLE_1  
RIGHT JOIN TABLE_2  
ON TABLE_1.KEY = TABLE_2.KEY
```

## OUTER JOIN



Everything on the right  
+  
Everything on the left

```
SELECT *  
FROM TABLE_1  
OUTER JOIN TABLE_2  
ON TABLE_1.KEY = TABLE_2.KEY
```

## INNER JOIN



Only the things that match on the left AND the right

```
SELECT *  
FROM TABLE_1  
INNER JOIN TABLE_2  
ON TABLE_1.KEY = TABLE_2.KEY
```

## ANTI LEFT JOIN



Everything on the left that is NOT on the right

```
SELECT *  
FROM TABLE_1  
LEFT JOIN TABLE_2  
ON TABLE_1.KEY = TABLE_2.KEY  
WHERE TABLE_2.KEY IS NULL
```

## ANTI RIGHT JOIN



Everything on the right that is NOT on the left

```
SELECT *  
FROM TABLE_1  
RIGHT JOIN TABLE_2  
ON TABLE_1.KEY = TABLE_2.KEY  
WHERE TABLE_1.KEY IS NULL
```

## ANTI OUTER JOIN



Everything on the left and right that is unique to each side

```
SELECT *  
FROM TABLE_1  
OUTER JOIN TABLE_2  
ON TABLE_1.KEY = TABLE_2.KEY  
WHERE TABLE_1.KEY IS NULL  
OR TABLE_2.KEY IS NULL
```

## CROSS JOIN

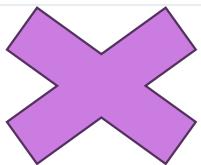


All combination of rows from the right and the left (cartesian product)

```
SELECT *  
FROM TABLE_1  
CROSS JOIN TABLE_2
```

## ข้อควรระวัง

- การ JOIN ต้องใช้ Foreign key เป็นหลัก มิเช่นนั้นผลลัพธ์จะคาดการณ์ยาก



```
SELECT *
FROM account A JOIN branch B
ON A.account_number = B.branch_name;
```

account		
Account_number	Branch_name	balance
1	B	\$100
2	A	\$50
3	A	\$30
4	F	\$120
5	A	\$500
6	B	\$324
7	X	\$222

branch		
Branch_name	Branch_city	assets
A	Riverside	\$10,000
B	LA	\$20,000
C	Long Beach	\$15,000
D	Irvine	\$12,000
E	Pomona	\$7,000
F	San Jose	\$18,000

## Relational algebra --> SQL command

Sailors			
<i>sid</i>	<i>sname</i>	<i>rating</i>	<i>age</i>

$\pi_{sname, age} (Sailors)$



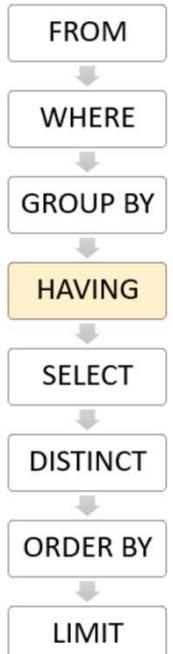
SELECT S.sname, S.age FROM Sailors S

$\pi_{sname, age} (\sigma_{age > 35 \wedge rating > 10} (Sailors))$

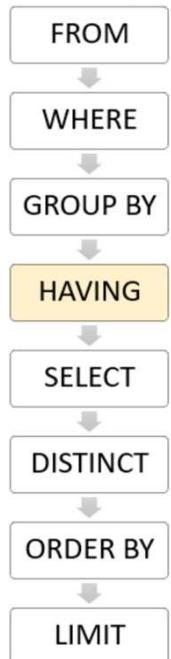


SELECT S.sname, S.age FROM Sailors S

WHERE S.age > 35 and S.rating > 10



## Relational algebra --> SQL command



$\pi_{\text{sname, age}} (\text{Sailors} \times \text{Reserve})$



SELECT S.sname, S.age FROM Sailors S, Reserve R

Cartesian Product

$\pi_{\text{sname, age}} (\sigma_{\text{age}>35 \wedge \text{rating}>10} (\text{Sailors} \times \text{Reserve}))$



SELECT S.sname, S.age FROM Sailors S, Reserve R

WHERE S.age > 35 and S.rating > 10

- “Compute available credit for every credit account.”
2. โจทย์อาจเขียนในรูปแบบ relational algebra แทนประโยคภาษาธรรมชาติ แล้วให้เขียน SQL

$$\Pi_{\text{cred\_id}, (\text{limit} - \text{balance}) \text{ as available\_credit}}(\text{credit\_acct})$$

cred_id	limit	balance
C-273	2500	150
C-291	750	600
C-304	15000	3500
C-313	300	25

*credit\_acct*



cred_id	available_credit
C-273	2350
C-291	150
C-304	11500
C-313	275

# Other useful functions

- `SELECT CONCAT(first_name, ' ', last_name) AS "Full name" FROM customer;`
- `SELECT first_name, LENGTH(first_name) AS "Length of a First Name" FROM employees WHERE length(first_name)>7;`