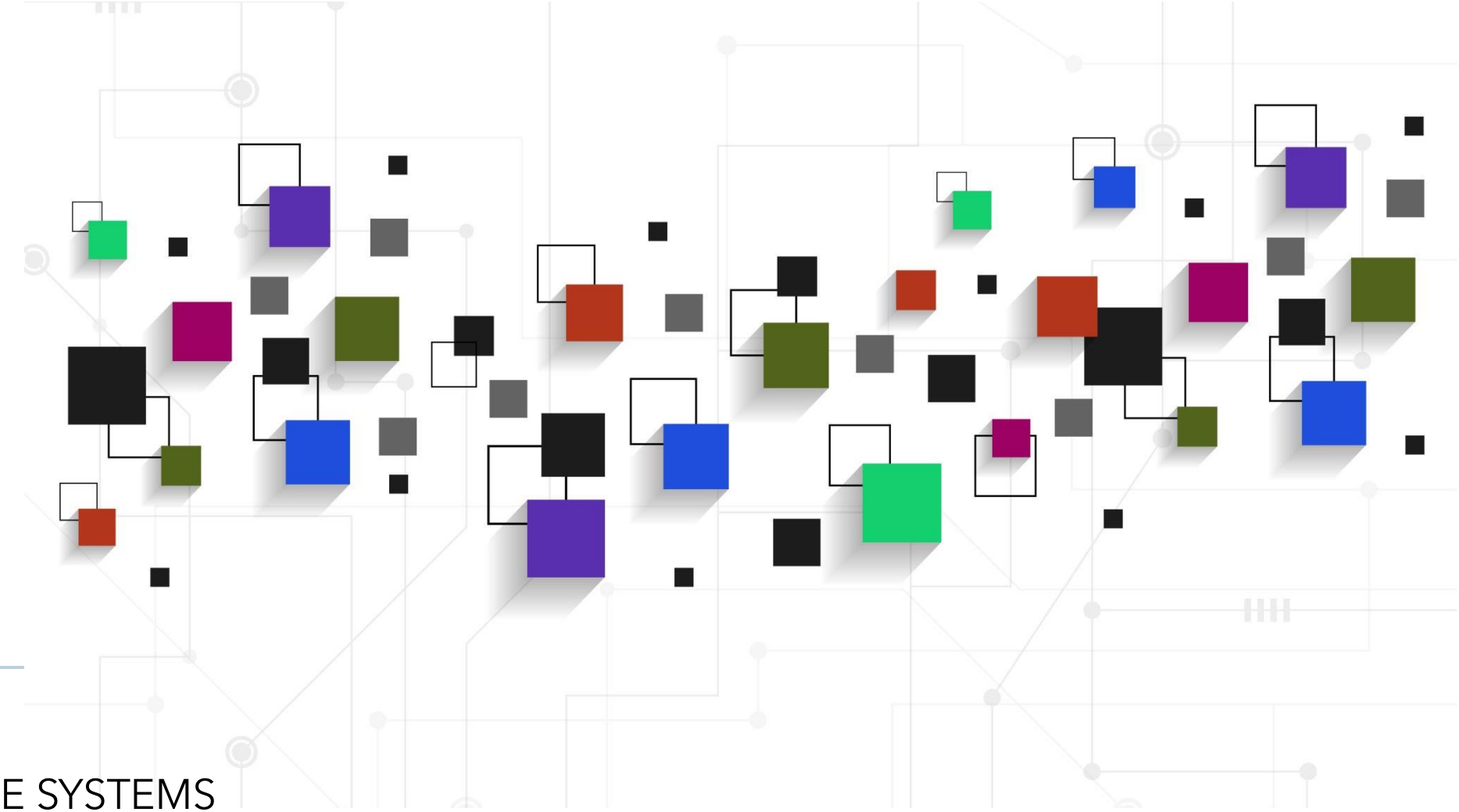


SQL

2110322 DATABASE SYSTEMS

ขอบพระคุณอาจารย์วิวัฒน์และอาจารย์โชติรัตน์

สำหรับต้นแบบสไลด์ในเทอมก่อนหน้า

- SQL introduction and history
- DDL
  - CREATE database and table
  - PostgreSQL data types
  - Constraints
  - ALTER TABLE
  - DROP database/ table
- DML (Data Manipulation Language) commands
  - Bongo ERD
  - INSERT
  - SELECT FROM WHERE DISTINCT
  - AS, LIKE
  - LIMIT, FETCH, OFFSET
  - UNION, INTERSECT, EXCEPT, IN and nested query
  - Aggregate operators: COUNT, SUM, AVG, MAX, MIN and also UPDATE
  - BETWEEN and AND

- Banking ERD
  - \i
  - \copy
  - EXISTS and NOT EXISTS
  - ALL and ANY/SOME
  - GROUP BY
  - HAVING
  - ORDER BY
  - JOIN, NATURAL JOIN, LEFT OUTER JOIN, RIGHT OUTER JOIN, FULL OUTER JOIN
- Relational algebra to SQL mapping
- Database backup and restore in PostgreSQL via pgAdmin
- Comparison between PostgreSQL and MySQL

SQL (Structured Query Language)

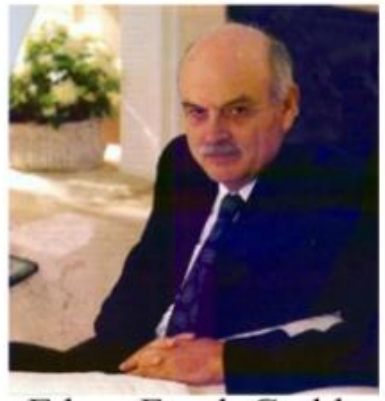
- ภาษา SQL ถูกกำหนดโดยสถาบันมาตรฐานแห่งชาติของอเมริกา (American National Standards Institute: ANSI)
- ประกอบด้วย 2 ส่วนหลัก
  - **Data Definition Language (DDL)** เป็นชุดคำสั่งที่ใช้ในการกำหนดหรือปรับแต่งแก้ไขสกีมาของฐานข้อมูล
  - **Data Manipulation Language (DML)** เป็นชุดคำสั่งที่ใช้ในจัดการข้อมูล เช่น เพิ่ม ลด แก้ไขข้อมูลในตารางต่างๆ ของฐานข้อมูล

ใครเป็นคนสร้างภาษา SQL

- เวอร์ชันแรกถูกพัฒนาที่ IBM โดย Donald D. Chamberlin และ Raymond F. Boyce
- ต่อมาได้ถูกพัฒนาเป็นภาษา SQL โดยอ้างอิงจากงานตีพิมพ์ของ E.F.Codd ชื่อเรื่อง “A Relational Model of Data for Large Shared Data Banks”

Donald D. Chamberlin

Raymond F. Boyce



Edgar Frank Codd

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 Latest (https://www.iso.org/standard/63555.html)

common

- Commercial systems offer most, if not all, SQL-92 features, plus varying feature sets from later standards and special proprietary features.

DDL

(**D**ata

Definition

Language

commands)

CREATE database/table

ใช้ในการสร้างฐานข้อมูลหรือตารางใหม่

ALTER table

ใช้ในการเพิ่ม ลด คอลัมน์ในตาราง

DROP database/table

ใช้ในการลบฐานข้อมูลหรือลบตารางออกจากฐานข้อมูล

Create database

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

เช่น

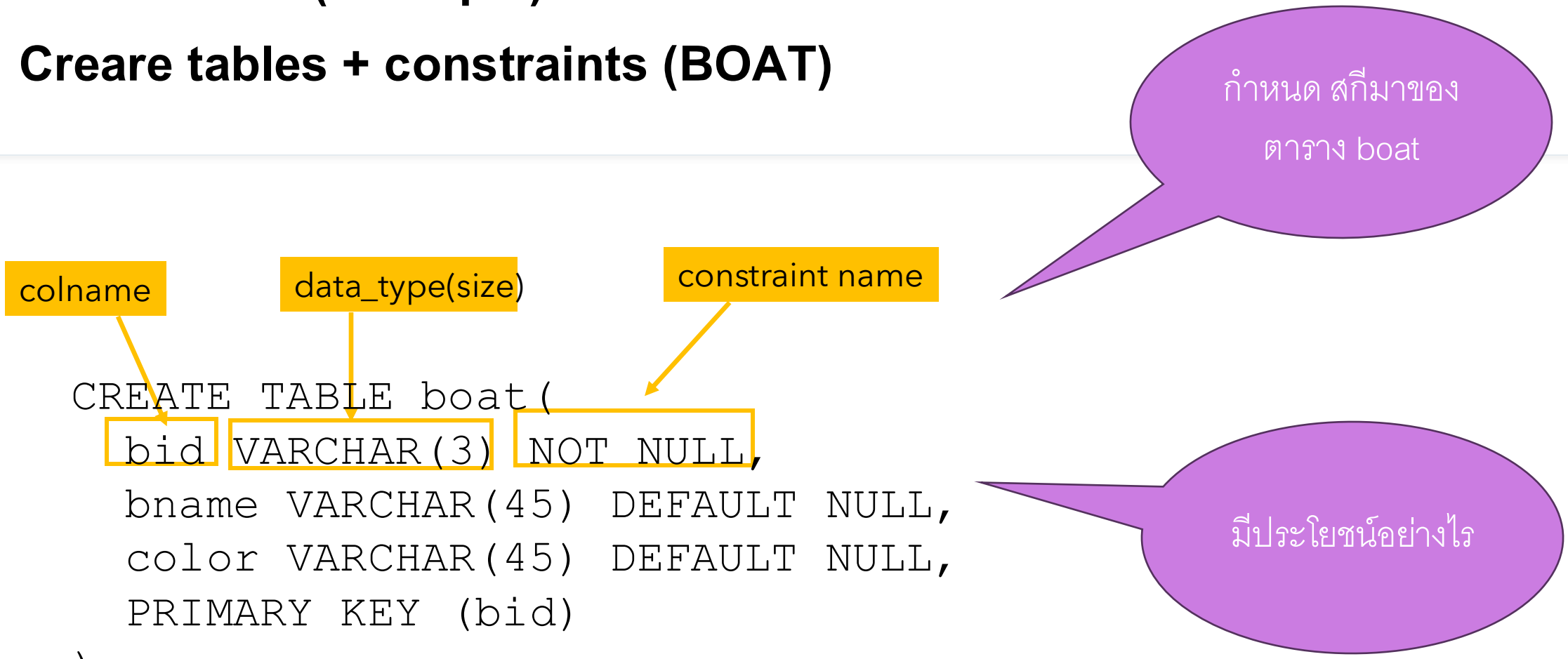
```
CREATE DATABASE bongo;
```

Create table

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

Create table (Example)

Creare tables + constraints (BOAT)



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 [***fields***] [(*p*)]		time span

การเลือกใช้ data type ที่เหมาะสมใน RDBMS หนึ่งๆ เป็นส่วนสำคัญในการออกแบบ database schema

https://www.postgresql.org/docs/current/datatype.html [online access on Feb 26. 2025]

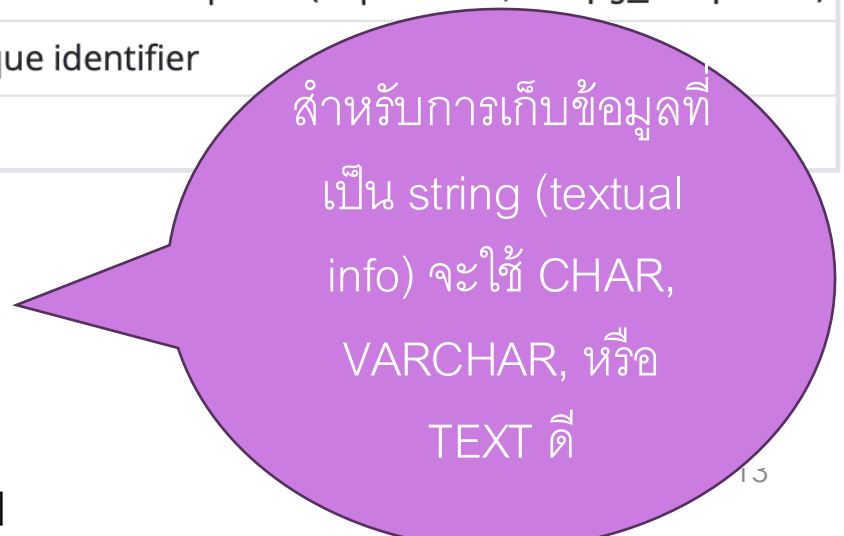
PostgreSQL data types (cont.)

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

https://www.postgresql.org/docs/current/datatype.html [online access on Feb 26. 2025]

PostgreSQL data types (cont.)

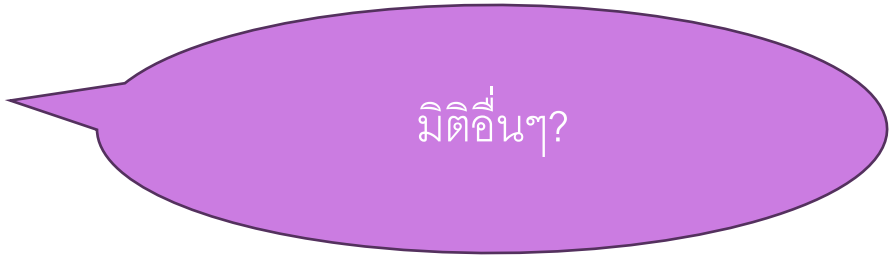
Name	Aliases	Description
`smallserial`	`serial2`	autoincrementing two-byte integer
`serial`	`serial4`	autoincrementing four-byte integer
`text`		variable-length character string
`time [ (p) ] [ without time zone ]`		time of day (no time zone)
`time [ (p) ] with time zone`	`timetz`	time of day, including 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



https://www.postgresql.org/docs/current/datatype.html [online access on Feb 26. 2025]

VARCHAR & TEXT use cases

- TEXT (max length unlimited)
  - เก็บ text ขนาดใหญ่ไม่จำกัดขนาด เช่น comments, งานตีพิมพ์, เอกสาร, blog posts ที่ไม่ทราบความยาวแน่นอน อาจยาวเกิน 65535 ไบต์



- VARCHAR(n) เก็บได้มากสุด 65535 ไบต์
  - มีการกำหนดความยาวสูงสุดไว้ แต่ข้อมูลมีความยาวต่างกันได้โดยยาวไม่เกินที่กำหนด เช่น ชื่อ นามสกุล อีเมล

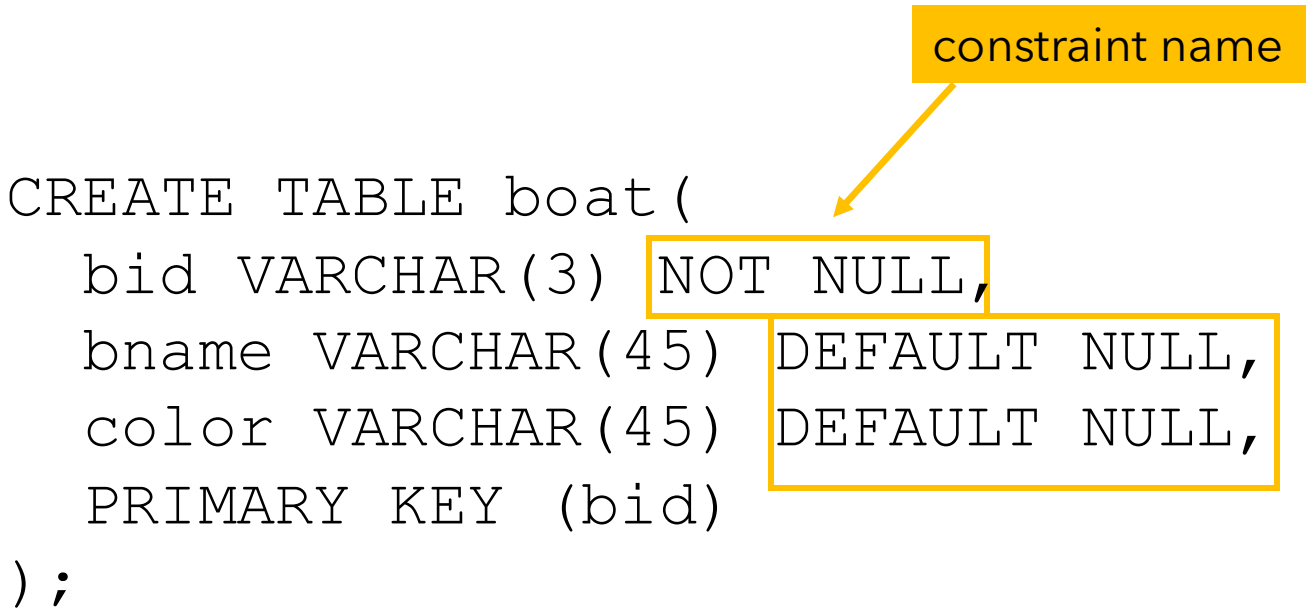
https://airbyte.com/data-engineering-resources/postgres-text-vs-varchar

Constraints (หัวใจของการใช้ RDBMS แทนไฟล์)

- **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

Create table (Example)

Creare tables + constraints (BOAT)



Not Null

Not Null and Unique constraint on P_Id

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

Unique constraint on multiple columns

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

PK Constraint

PK constraint on P_Id

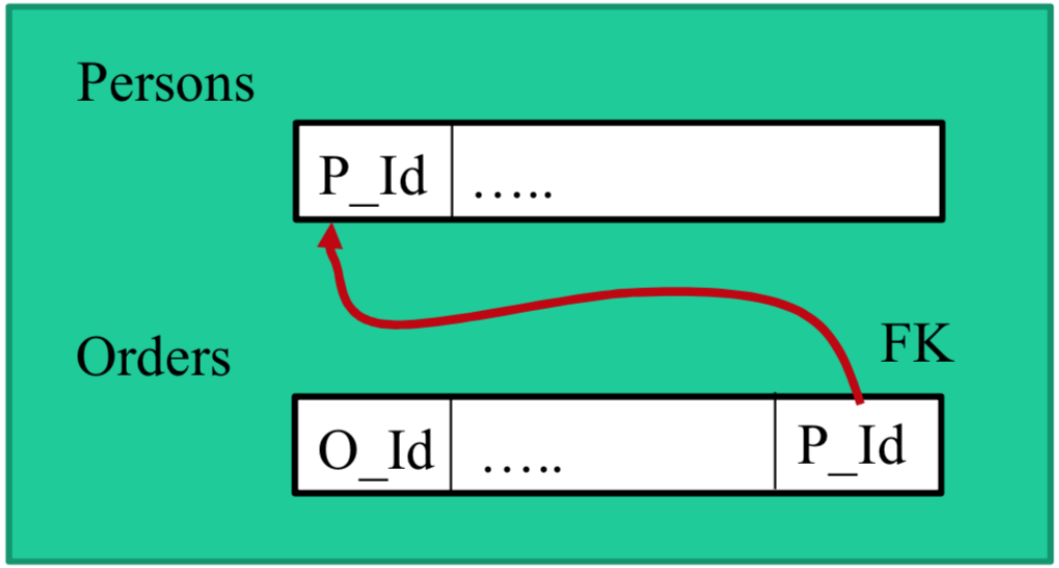
```
CREATE TABLE Persons
(
P_Id int NOT NULL,
LastName varchar(255) NOT NULL,
FirstName varchar(255),
Address varchar(255),
City varchar(255),
PRIMARY KEY (P_Id)
)
```

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

PK constraint on multiple columns

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

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)
)
```

FK constraint on multiple columns

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

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

Default

Default constraint on City

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

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

GETDATE() เป็น system function
Return ค่าวันที่ของระบบมาให้

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()
)
```

NOW()

SERIAL field for unique values in PostgreSQL

(Auto-Increment field in MySQL)

Auto-Increment constraint on ID starting from 1

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

มี smallserial และ bigserial ด้วย

Not specify values for ID column on INSERT

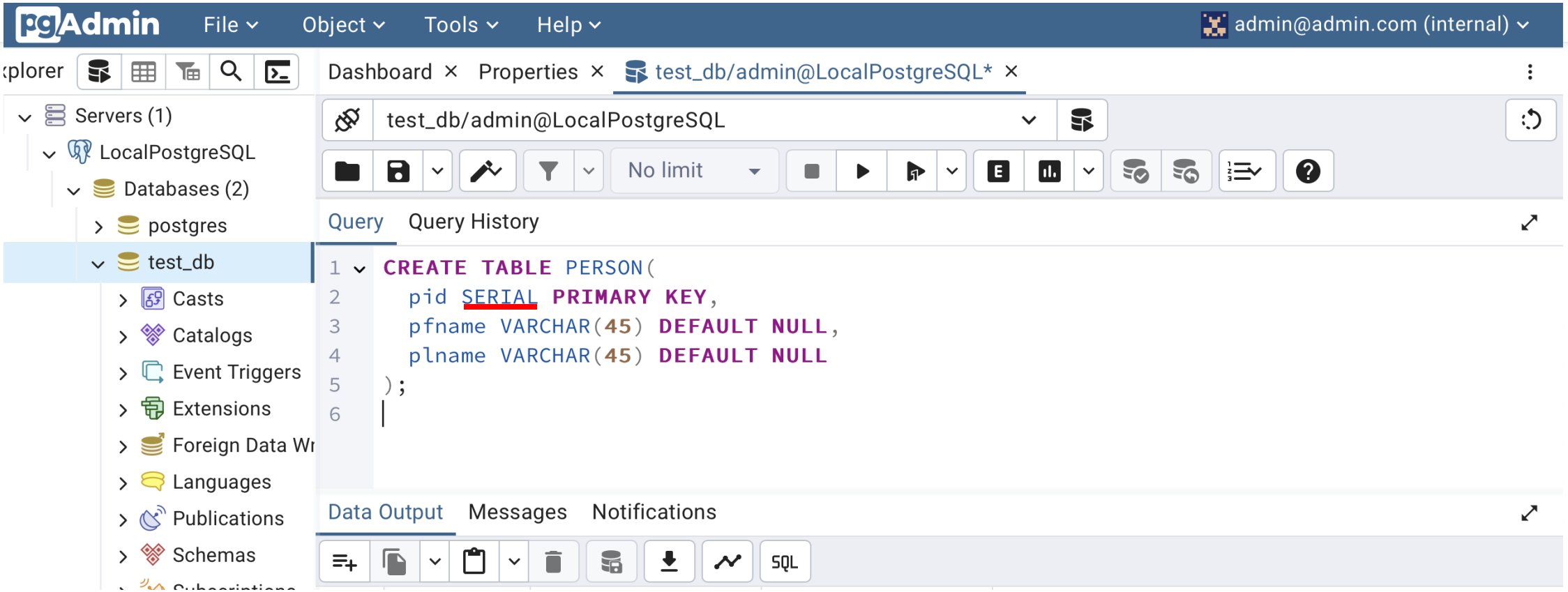
```
INSERT INTO Persons
(FirstName,LastName)
VALUES ('Lars','Monsen')
```

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

https://www.geeksforgeeks.org/postgresql-serial/ [online access on Feb 26, 2025]

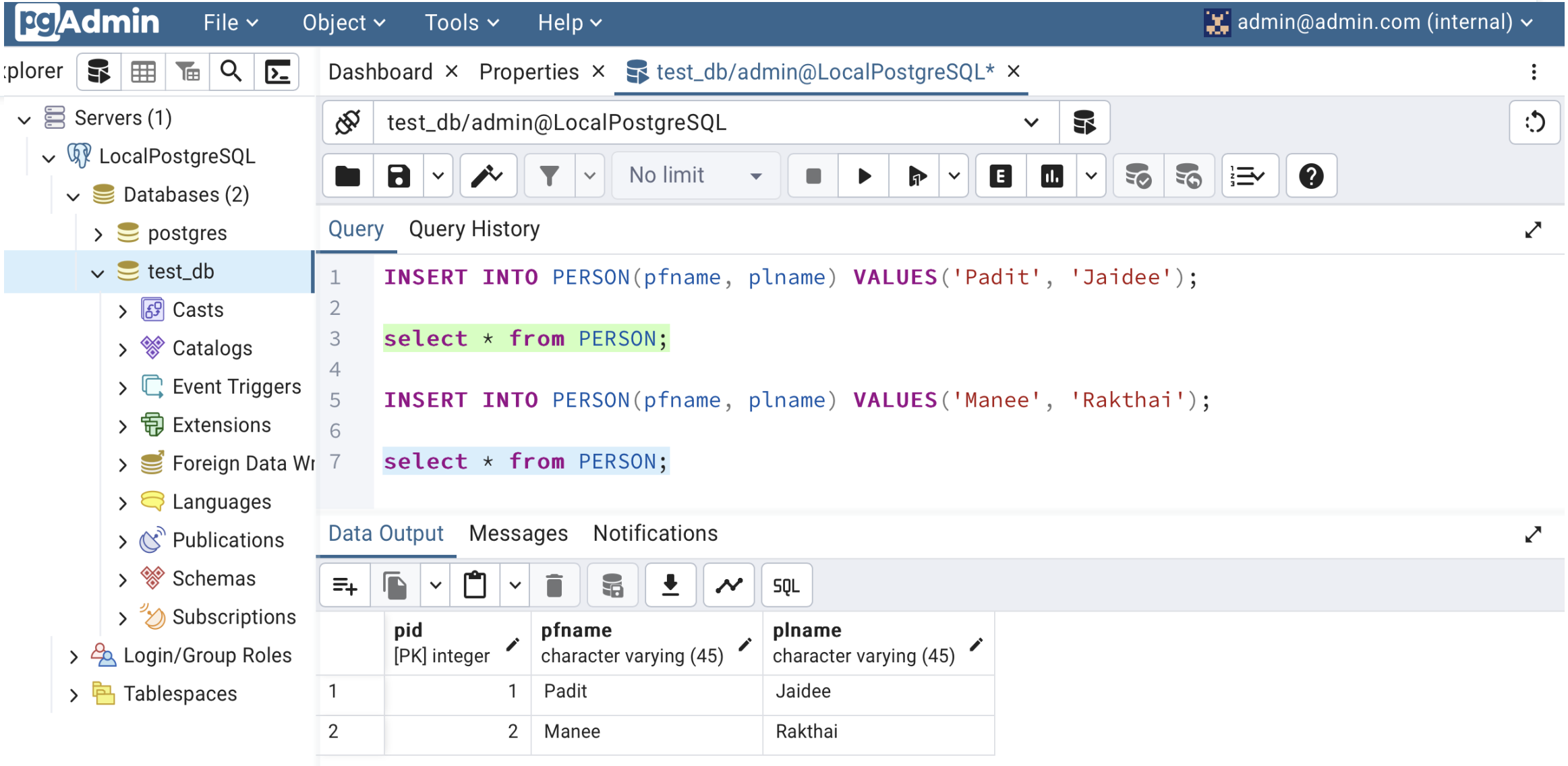
SERIAL field for unique values in PostgreSQL

(Auto-Increment field in PostgreSQL)

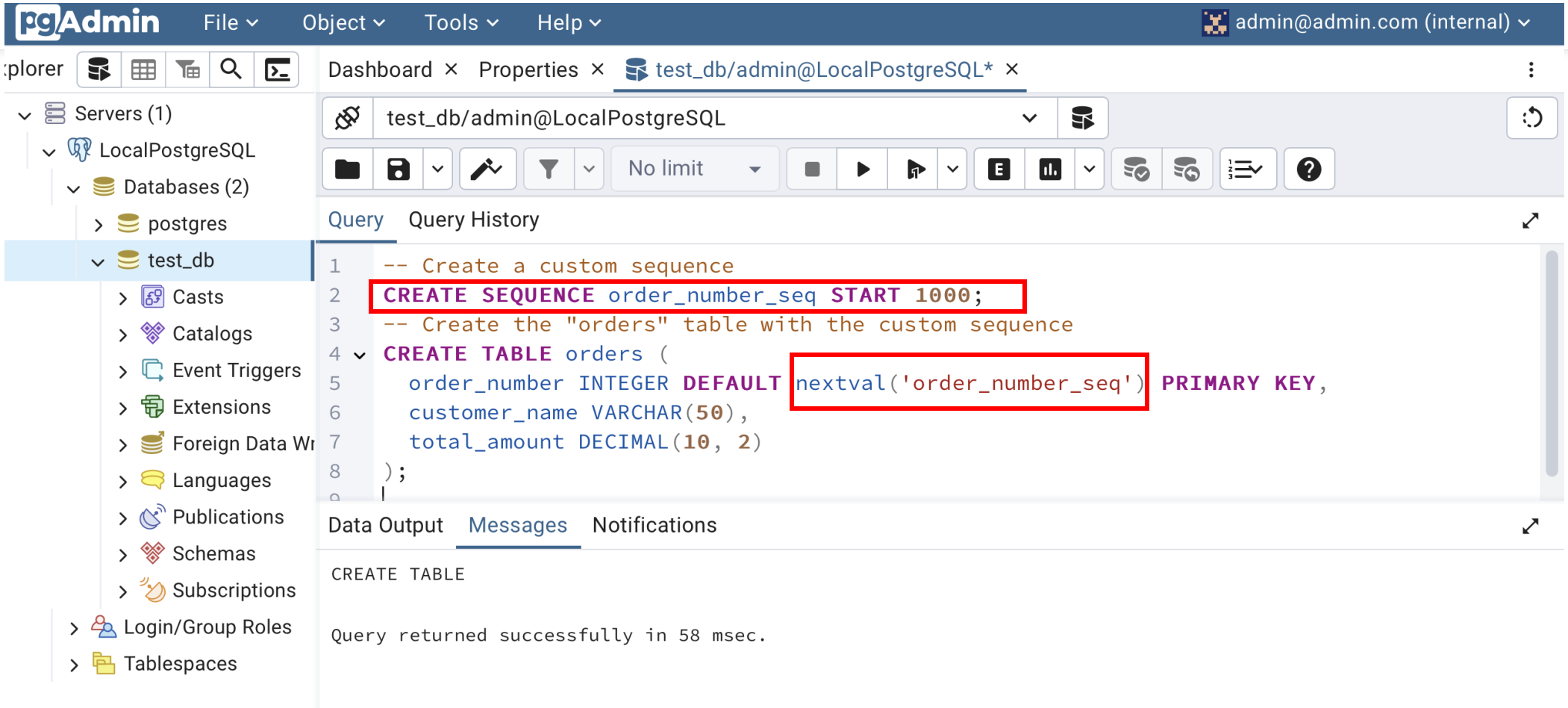


SERIAL field for unique values in PostgreSQL

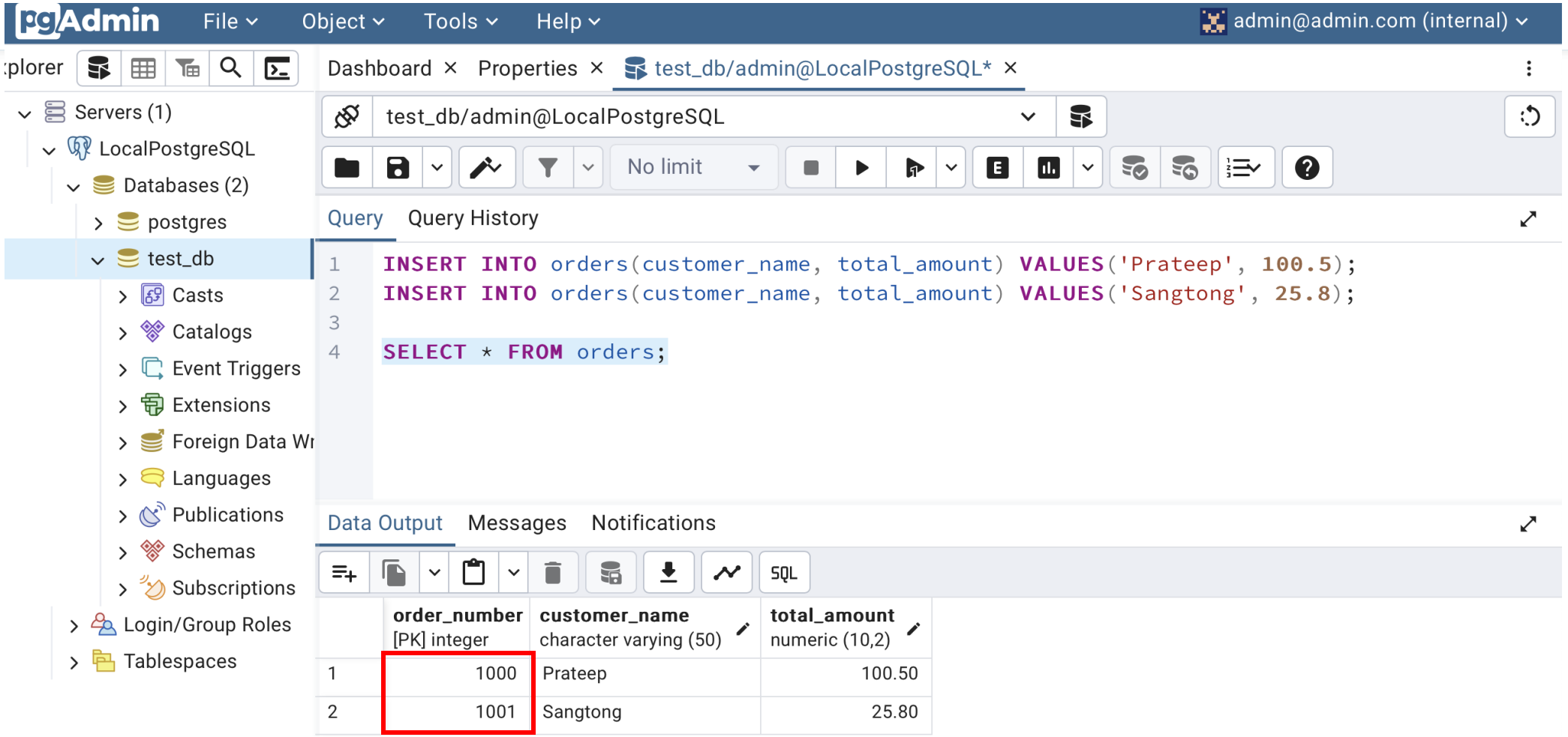
(Auto-Increment field in MySQL)



SERIAL field for unique values in PostgreSQL with starting value



SERIAL field for unique values in PostgreSQL with starting value



มาแทน SERIAL

GENERATE AS IDENTITY

เมื่อไหร่เราควรใช้ serial / auto-increment
เมื่อไหร่เราควรใช้ field ข้อมูลเป็น primary key
ไปเลย

ถ้าเป็น reg chula ที่มีตารางข้อมูลเก็บรหัสนิสิต
ชื่อ นามสกุล นิสิต คณะ ฯลฯ เราใช้รหัสนิสิตเป็น
primary key หรือทำ serial เพิ่มดี

ตลาดหลักทรัพย์มีตารางเก็บข้อมูลเกี่ยวกับหุ้น
โดยชื่อหุ้นห้ามซ้ำกันของแต่ละบริษัทที่จด
ทะเบียนกับตลาดหลักทรัพย์ เราใช้ชื่อหุ้นเป็น
primary key ดีไหม และหรือทำ serial เพิ่มมาดี

ความแตกต่างระหว่าง serial กับ
UUID?

https://pganalyze.com/blog/5mins-postgres-uuid-vs-serial-primary-keys

ลองสร้างฐานข้อมูลเล็ก ๆ เกี่ยวกับการจองเรือ

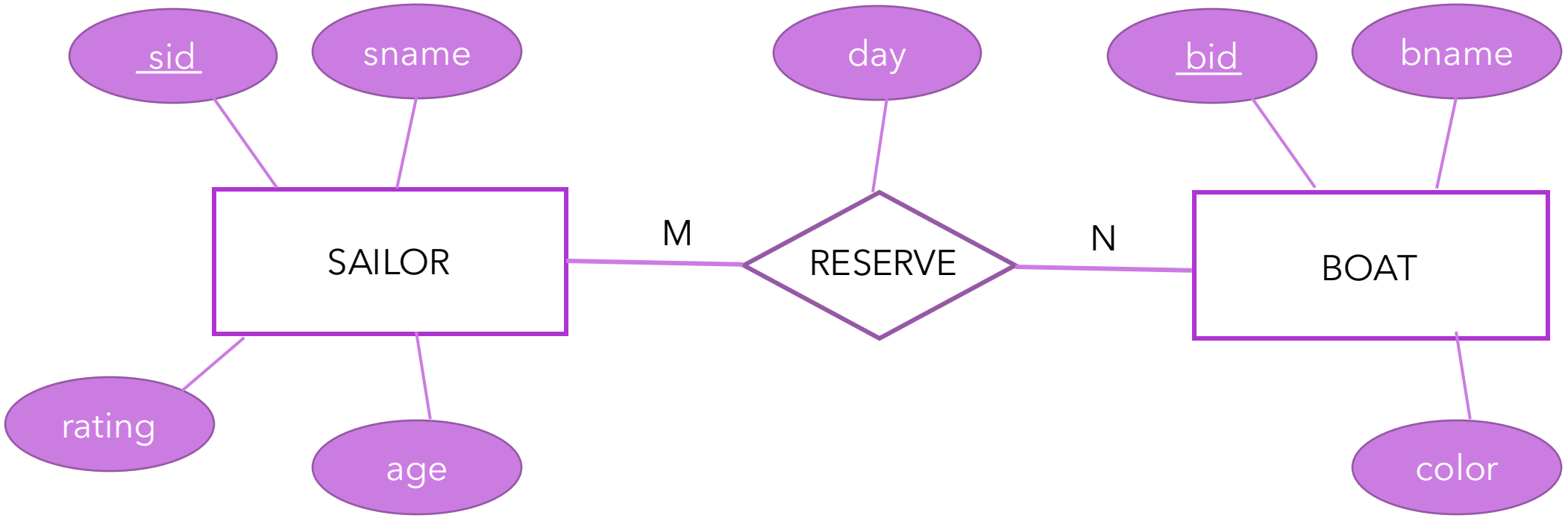
First toy database with DDL: CREATE DATABASE / CREATE TABLE

เตรียมข้อมูล

ให้ไป clone ชุดคำสั่งและไฟล์ข้อมูลจาก repository ต่อไปนี้

https://github.com/wichadak/2110322_DB_SYS

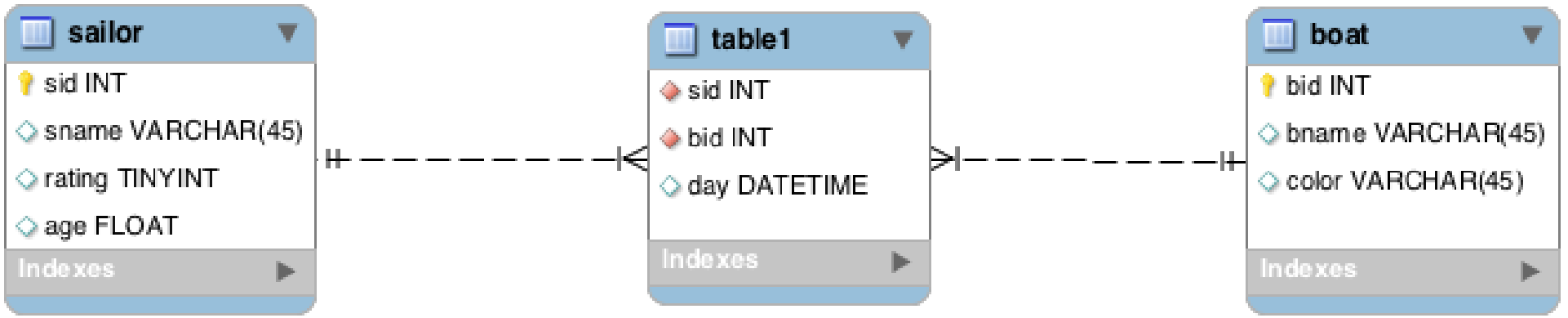
Bongo's ERR diagram (Chen's model)



ลูกค้าต้องการจัดทำฐานข้อมูล Bongo โดยต้องการเก็บข้อมูลต่อไปนี้

1. ข้อมูลกะลาสีเรือแต่ละคน ประกอบด้วย รหัสประจำตัว (sid) ชื่อ (sname) เรตติ้ง (rating) และอายุ (age)
2. ข้อมูลเรือแต่ละลำ ประกอบด้วย รหัสเรือ (bid) ชื่อเรือ (bname) และสีของเรือ (color)
3. ข้อมูลการจองเรือของกะลาสี (sid, bid, day)

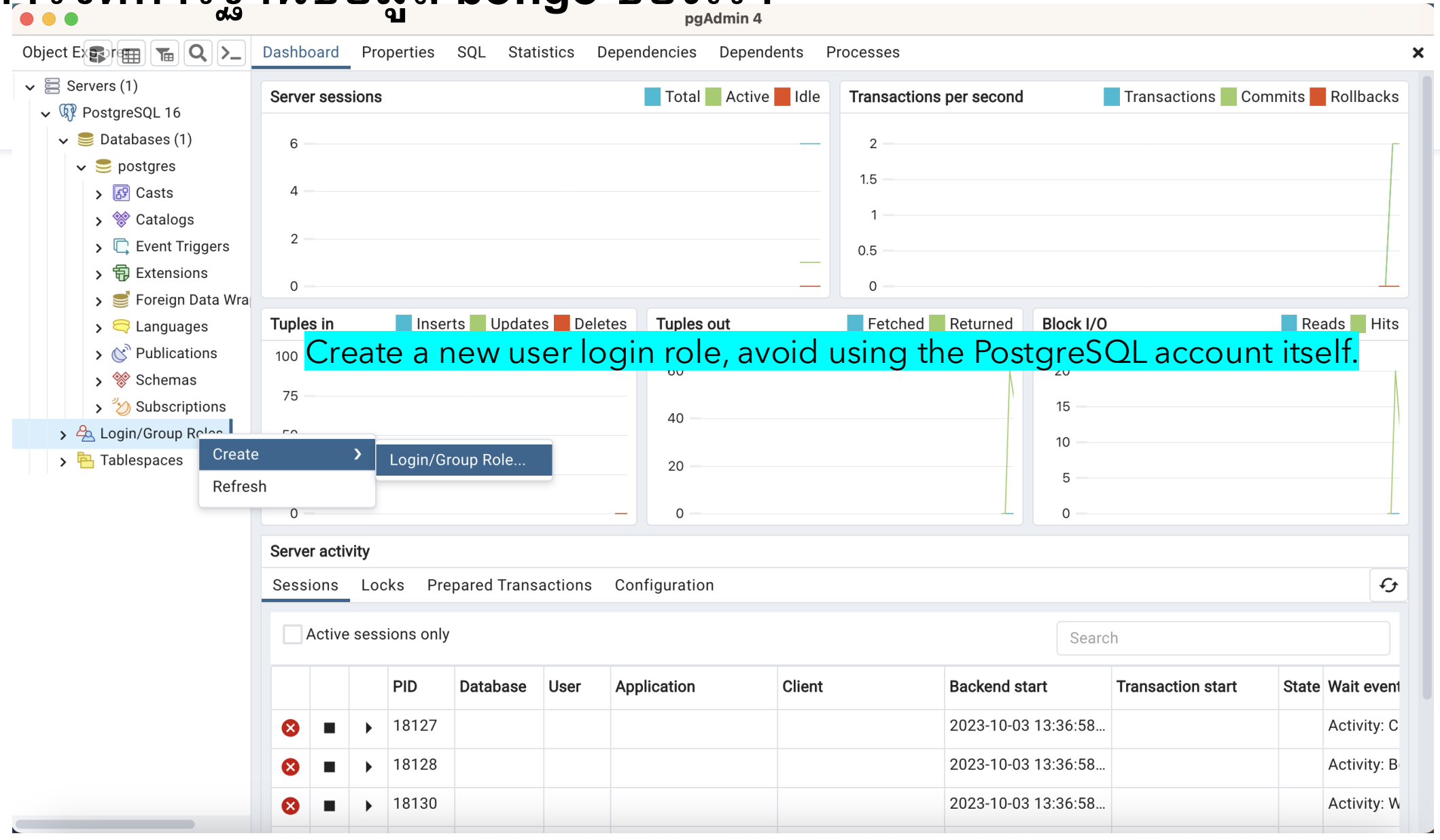
Bongo's ERR diagram (Crow's foot model)



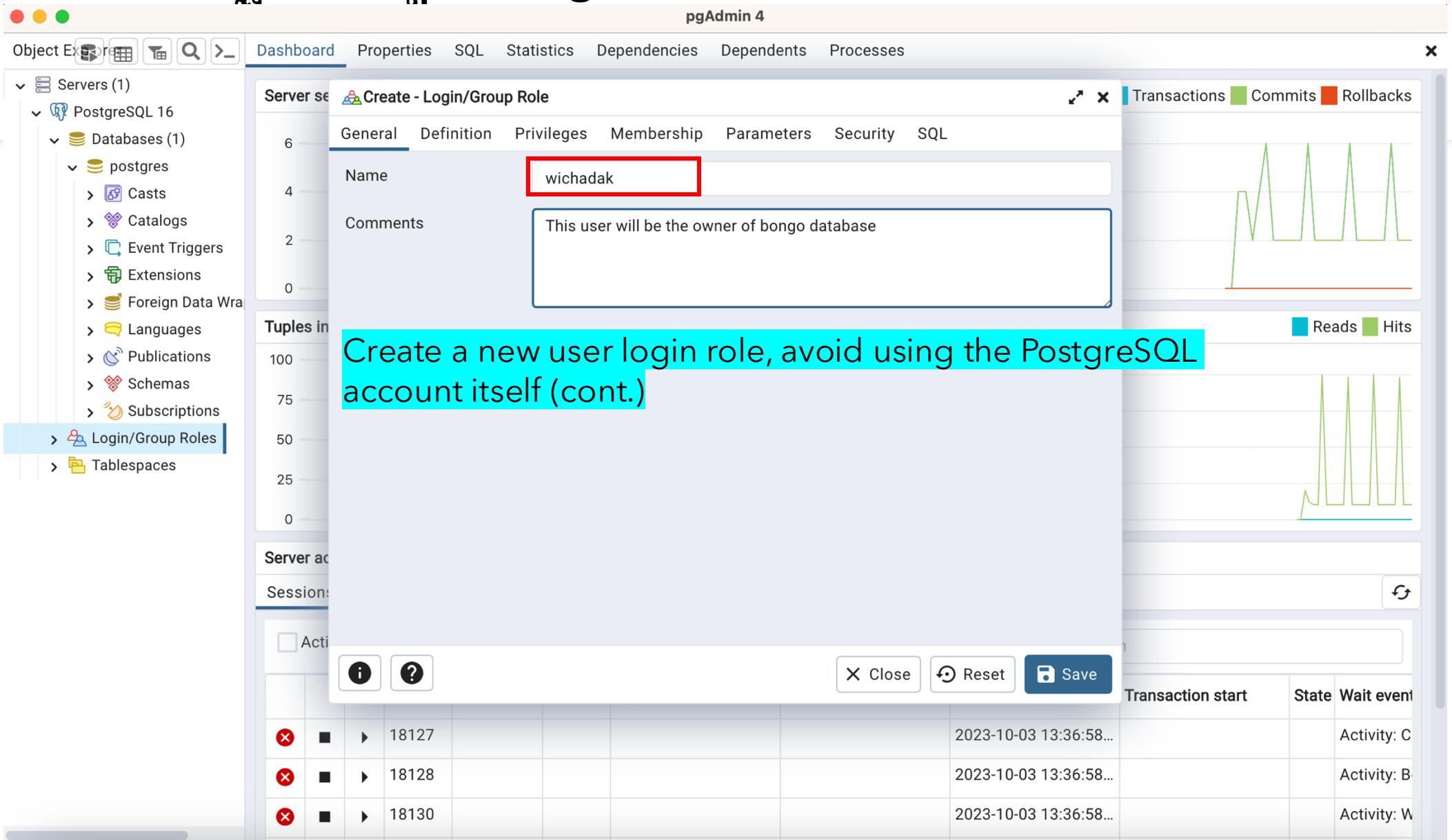
Bongo's RDBMS schema design

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

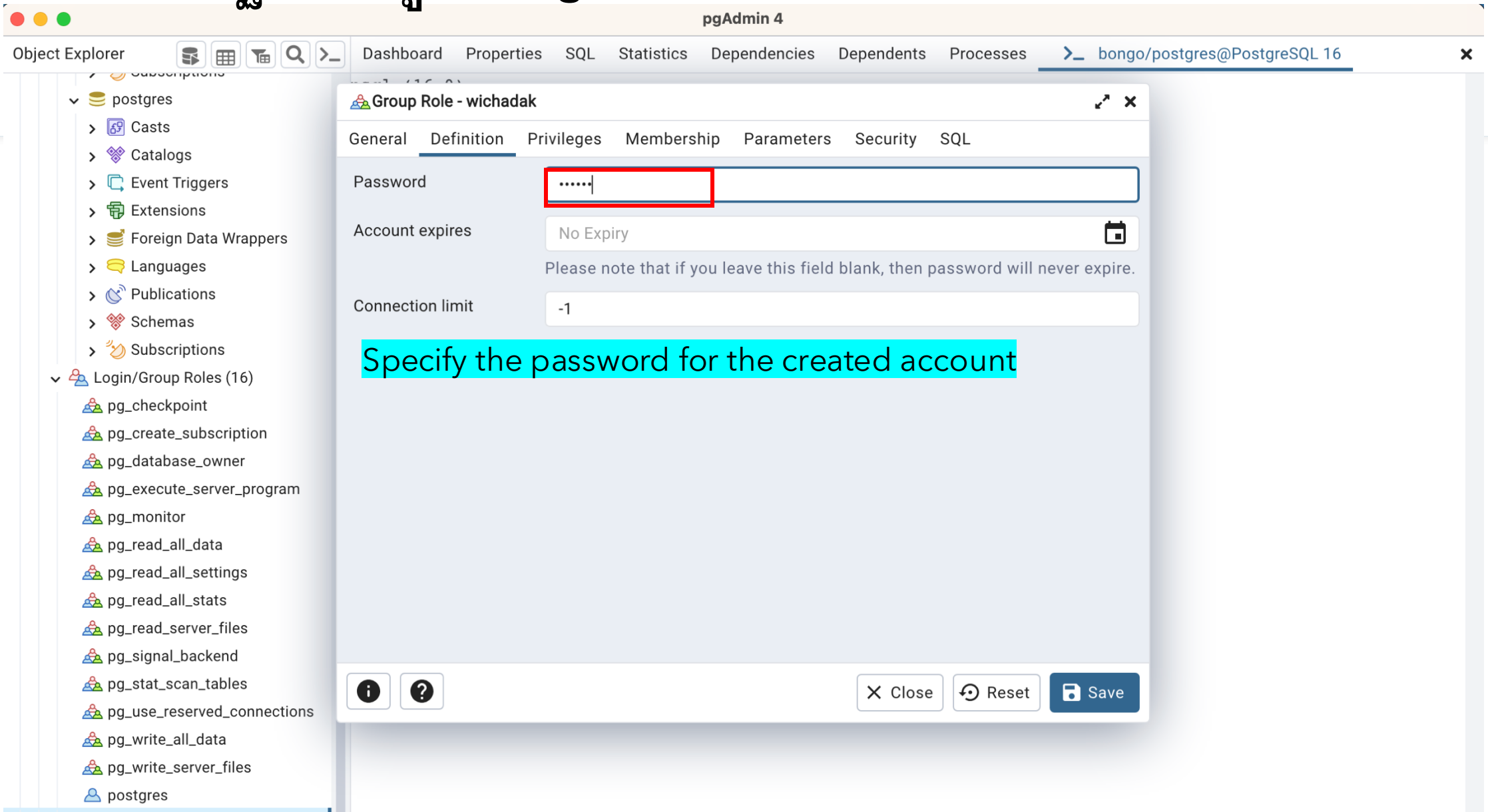
สร้าง account ใหม่ผ่าน pgAmin4 เพื่อให้ account ใหม่นี้ใน การจัดการฐานข้อมูล bongo ของเรา



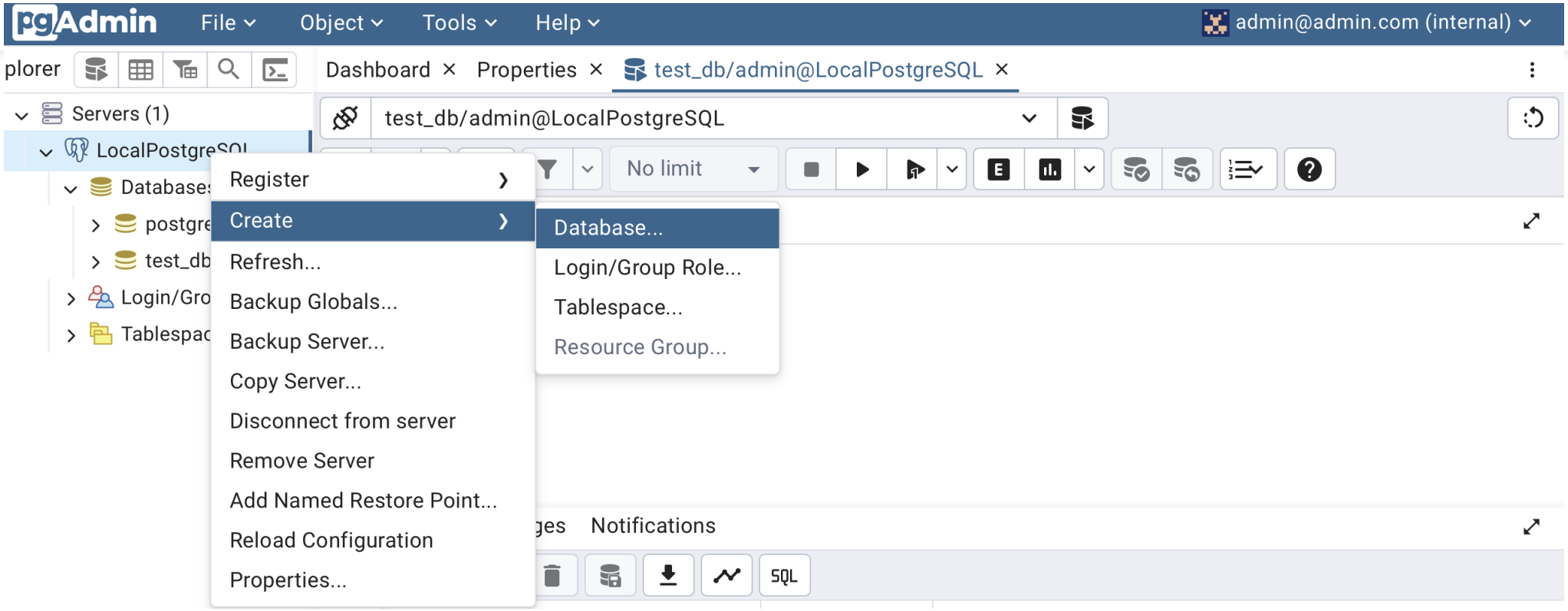
สร้าง account ใหม่ผ่าน pgAmin4 เพื่อให้ account ใหม่นี้ในการจัดการฐานข้อมูล bongo ของเรา



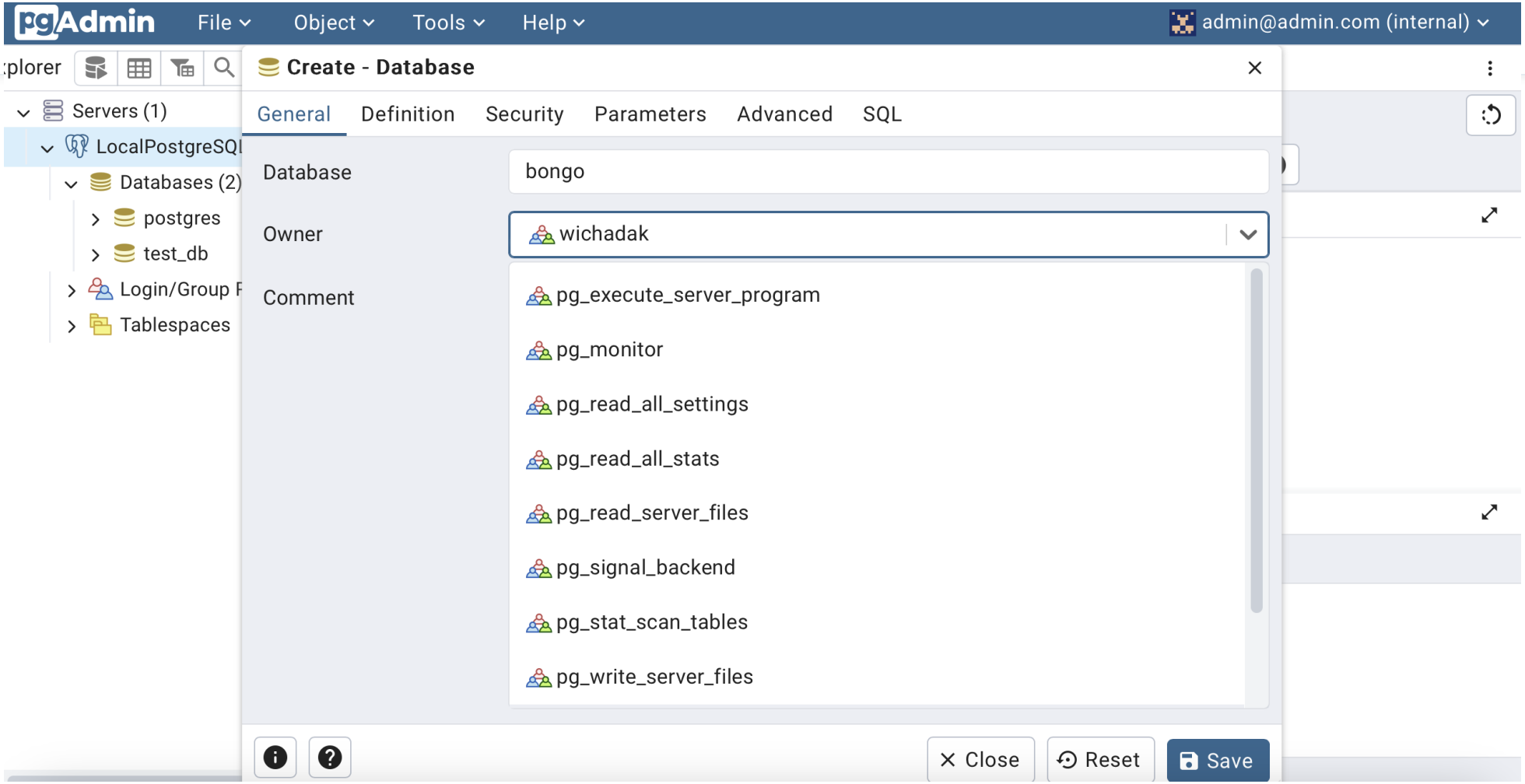
สร้าง account ใหม่ผ่าน pgAmin4 เพื่อให้ account ใหม่นี้ในการจัดการฐานข้อมูล bongo ของเรา



สร้างฐานข้อมูลโดยใช้ account ที่เราสร้างเมื่อกี๊



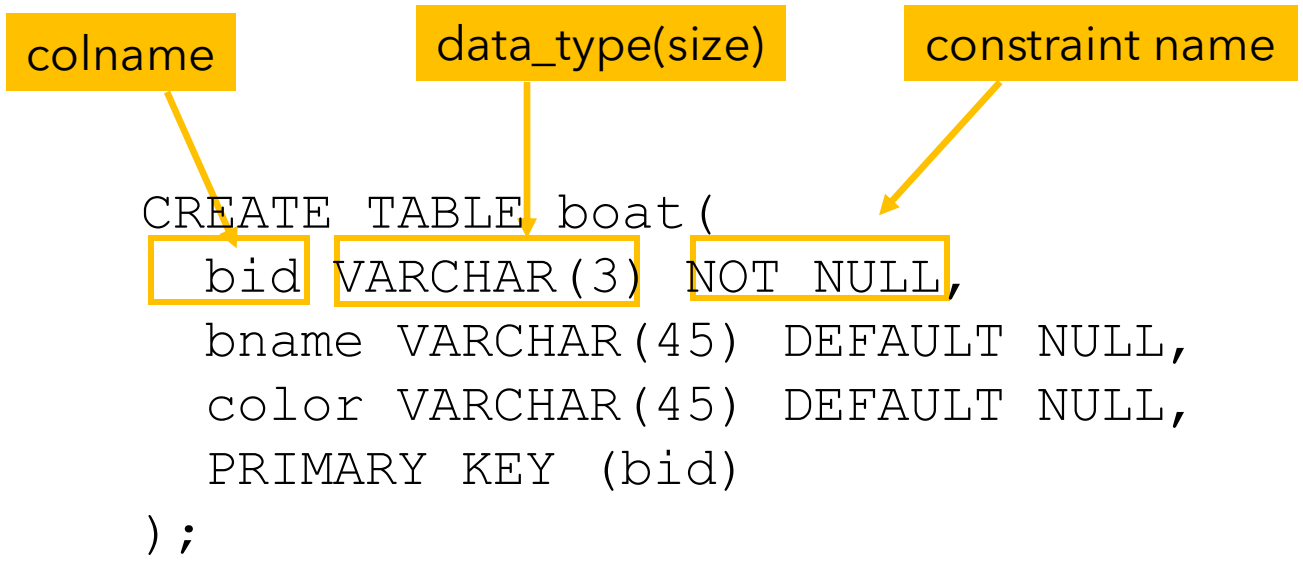
สร้างฐานข้อมูลโดยใช้ account ที่เราสร้างเมื่อกี๊



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)



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)
);
```

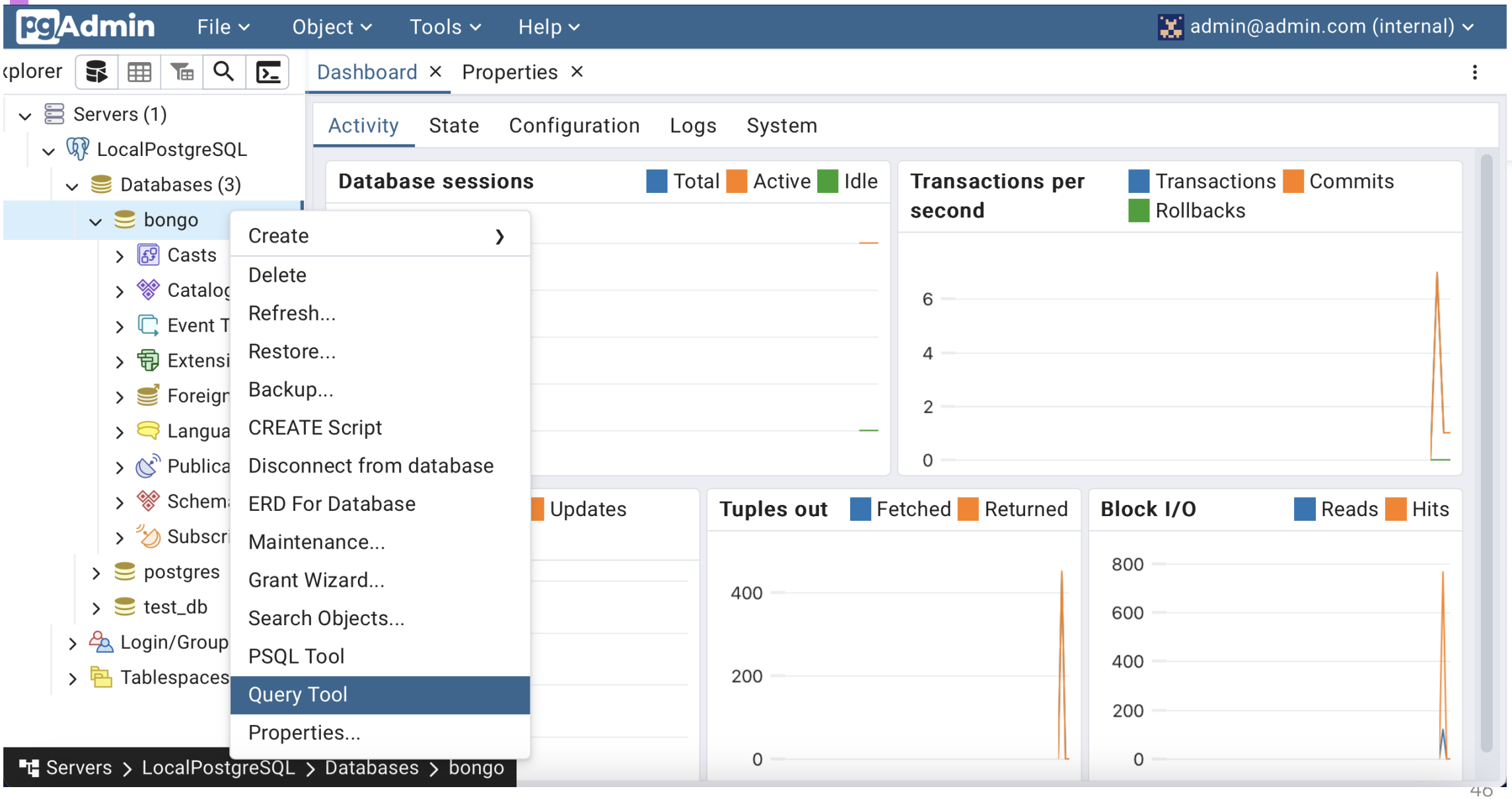
Create tables + constraints

(RESERVE)

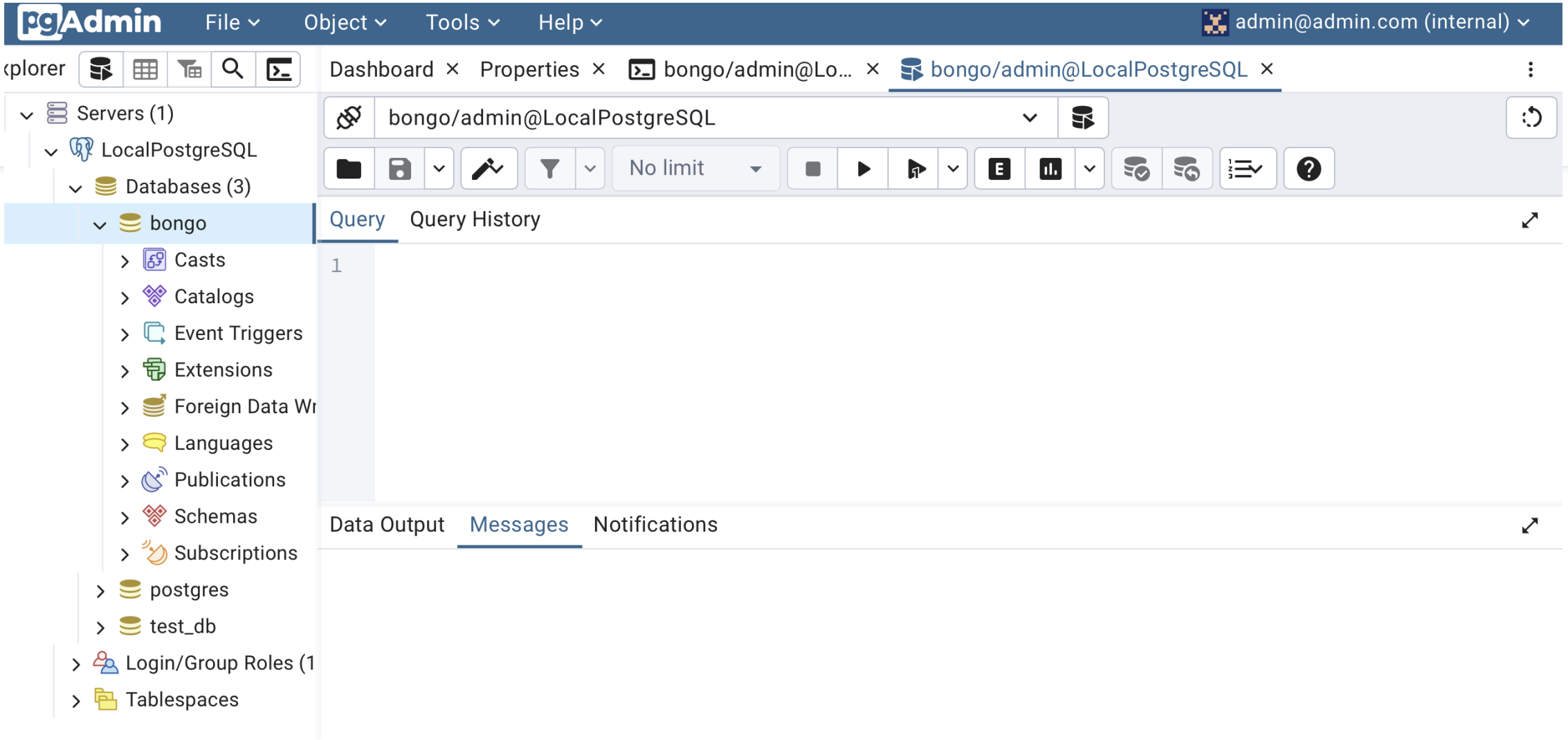
```
CREATE TABLE reserve (
   sid VARCHAR(2) NOT NULL,
   bid VARCHAR(3) NOT NULL,
   day DATE DEFAULT NULL
);
```

ลองสร้างฐานข้อมูลและทั้ง 3 ตารางดู โดยให้ copy คำสั่ง สร้างตารางจากไฟล์ `PostgreSQL_ToTry_commands_2024.txt` มาใส่เป็นคำสั่งใน Query Tool / PSQL tool ได้เลย ☺

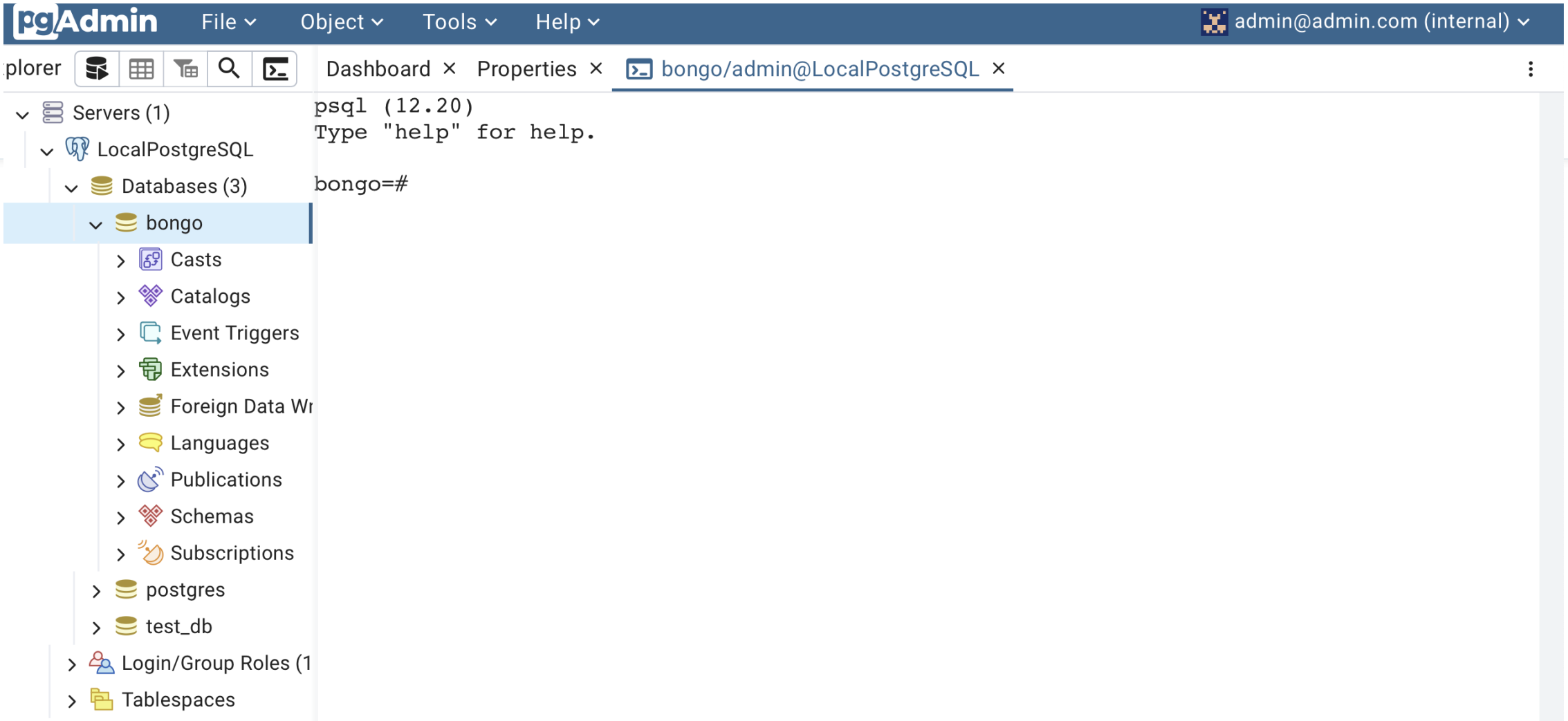
Open Query Tool



Query Tool GUI

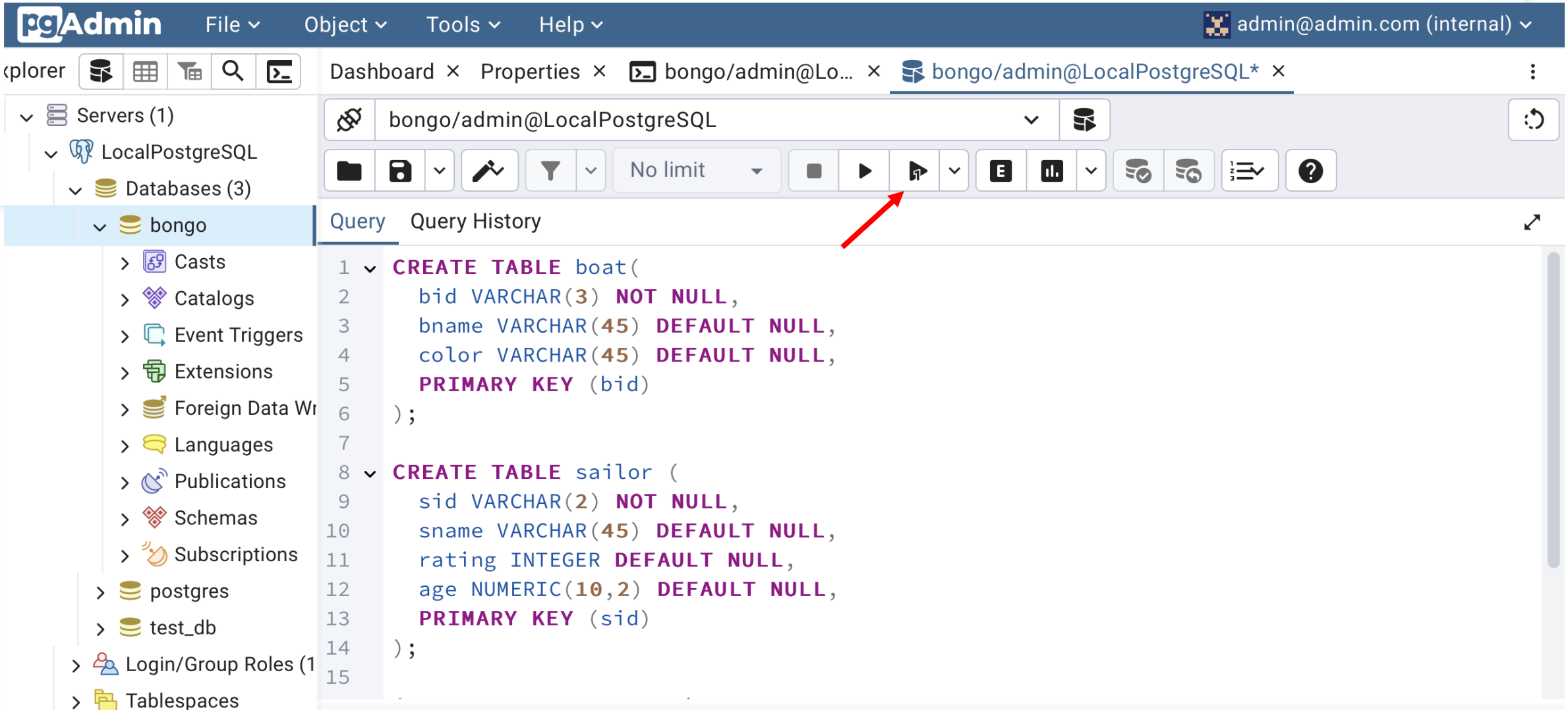


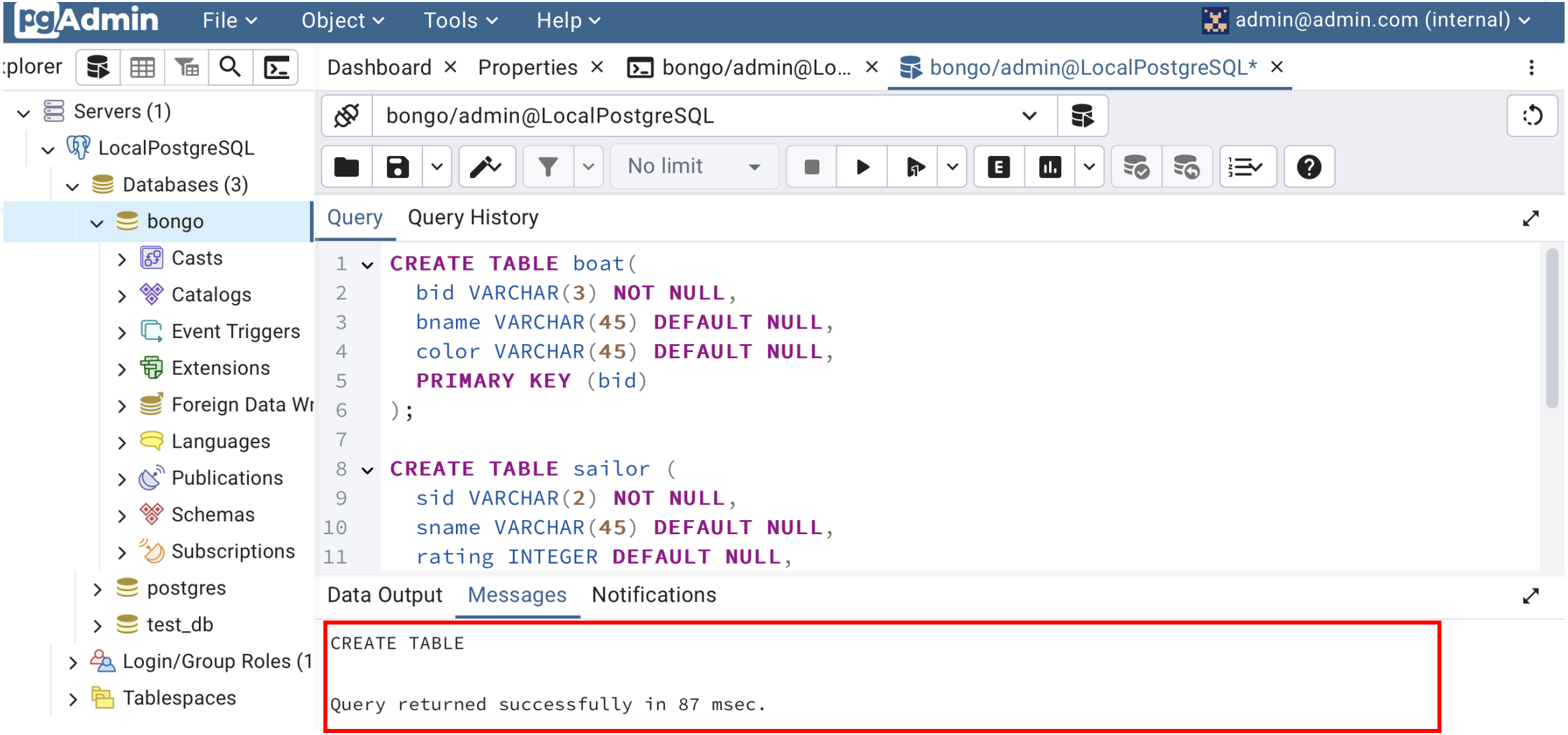
PSQL Tool GUI



ลองสร้างฐานข้อมูลเล็ก ๆ เกี่ยวกับการจองเรือ

First toy database with DDL : ALTER TABLE / DROP TABLE


pgAdmin
File
Object
Tools
Help
admin@admin.com (internal)
Dashboard
Properties
bongo/admin@Lo...
bongo/admin@LocalPostgreSQL*
Servers (1)
LocalPostgreSQL
Databases (3)
bongo
Casts
Catalogs
Event Triggers
Extensions
Foreign Data Wr
Languages
Publications
Schemas
Subscriptions
postgres
test_db
Login/Group Roles (1
Tablespaces
bongo/admin@LocalPostgreSQL
No limit
Query
Query History
1 CREATE TABLE boat(
2 bid VARCHAR(3) NOT NULL,
3 bname VARCHAR(45) DEFAULT NULL,
4 color VARCHAR(45) DEFAULT NULL,
5 PRIMARY KEY (bid)
6);
7
8 CREATE TABLE sailor (
9 sid VARCHAR(2) NOT NULL,
10 sname VARCHAR(45) DEFAULT NULL,
11 rating INTEGER DEFAULT NULL,
12 age NUMERIC(10,2) DEFAULT NULL,
13 PRIMARY KEY (sid)
14);
15


pgAdmin
File
Object
Tools
Help
admin@admin.com (internal)
Dashboard
Properties
bongo/admin@Lo...
bongo/admin@LocalPostgreSQL*
Servers (1)
LocalPostgreSQL
Databases (3)
bongo
Casts
Catalogs
Event Triggers
Extensions
Foreign Data Wr
Languages
Publications
Schemas
Subscriptions
postgres
test_db
Login/Group Roles (1
Tablespaces
bongo/admin@LocalPostgreSQL
No limit
Query
Query History
1 CREATE TABLE boat(
2 bid VARCHAR(3) NOT NULL,
3 bname VARCHAR(45) DEFAULT NULL,
4 color VARCHAR(45) DEFAULT NULL,
5 PRIMARY KEY (bid)
6);
7
8 CREATE TABLE sailor (
9 sid VARCHAR(2) NOT NULL,
10 sname VARCHAR(45) DEFAULT NULL,
11 rating INTEGER DEFAULT NULL,
Data Output
Messages
Notifications
CREATE TABLE
Query returned successfully in 87 msec.

ดูว่ามีตารางอะไรอยู่บ้าง และดูว่าตารางนั้น ๆ มี schema อย่างไร เช่น ตาราง reserve (ดูผ่าน PSQL Tool ใน pgAdmin)

MySQL

```
SHOW TABLES;
```

PostgreSQL

```
\d
```

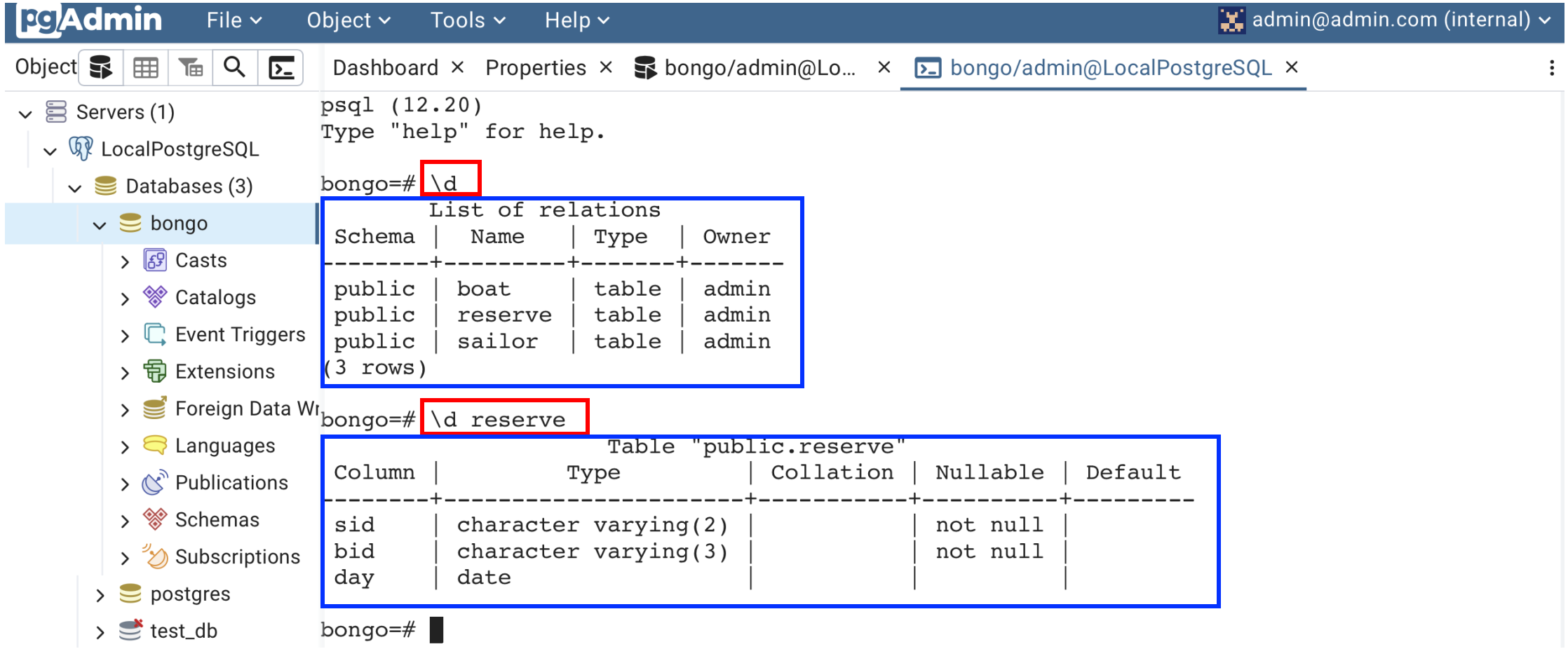
MySQL

```
DESCRIBE reserve;
```

PostgreSQL

```
\d reserve;
```

ดูว่ามีตารางอะไรอยู่บ้าง และดูว่าตารางนั้น ๆ มี schema อย่างไร เช่น ตาราง reserve (ดูผ่าน PSQL Tool ใน pgAdmin 4)



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;
```

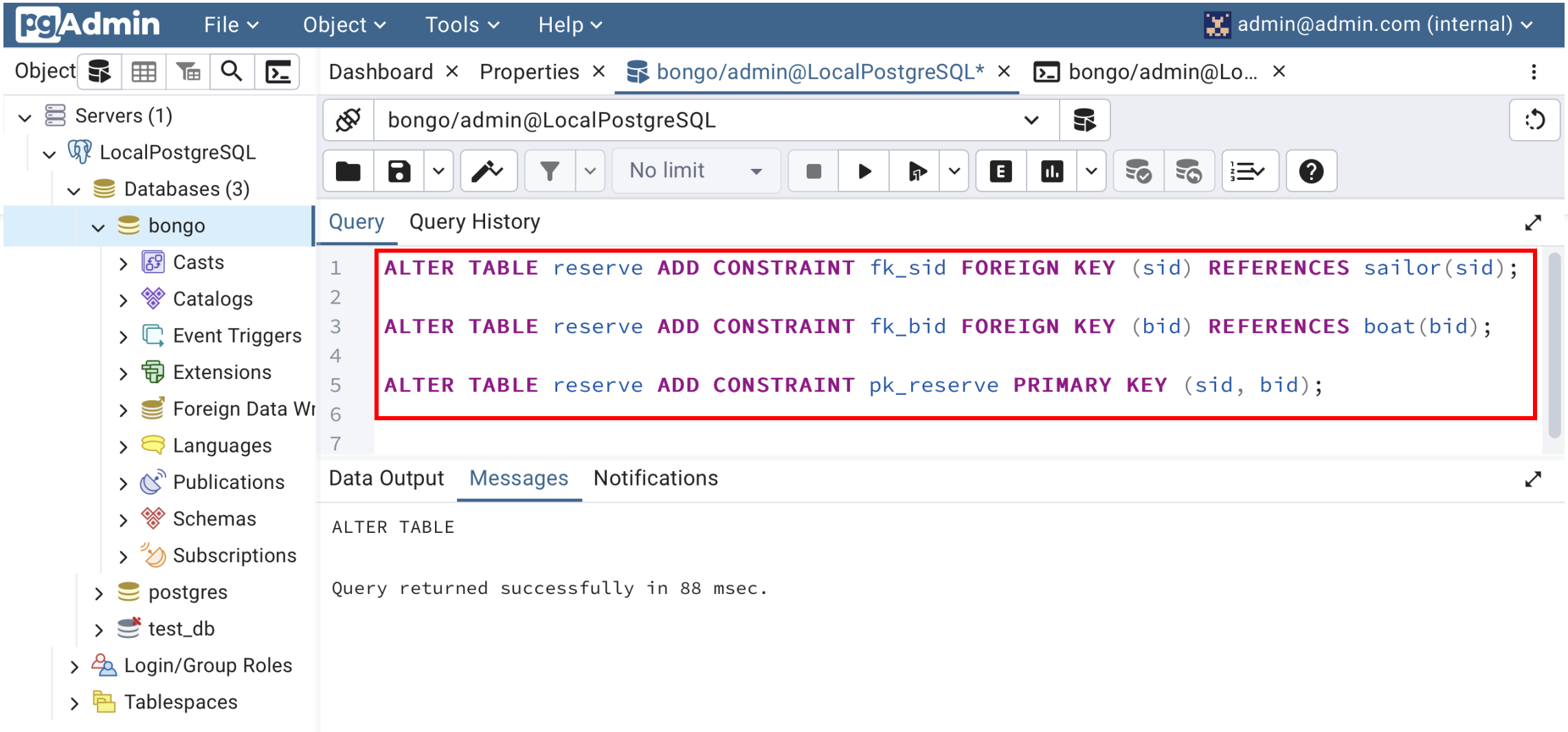
Within PSQL Tool

```
bongo=# \d reserve
                    Table "public.reserve"
 Column |          Type          | Collation | Nullable | Default
--------+------------------------+-----------+----------+---------
 sid    | character varying(2)   |           | not null |
 bid    | character varying(3)   |           | not null |
 day    | date                   |           |          |

bongo=# ALTER TABLE reserve ADD confirm boolean;
ALTER TABLE
bongo=# \d reserve
                    Table "public.reserve"
 Column  |          Type          | Collation | Nullable | Default
---------+------------------------+-----------+----------+---------
 sid     | character varying(2)   |           | not null |
 bid     | character varying(3)   |           | not null |
 day     | date                   |           |          |
 confirm | boolean                |           |          |
```

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);
```

ลองรันคำสั่งข้างต้นกับฐานข้อมูล bongo


pgAdmin
File
Object
Tools
Help
admin@admin.com (internal)
Object
Dashboard
Properties
bongo/admin@LocalPostgreSQL*
bongo/admin@Lo...
Servers (1)
LocalPostgreSQL
Databases (3)
bongo
Casts
Catalogs
Event Triggers
Extensions
Foreign Data Wr
Languages
Publications
Schemas
Subscriptions
postgres
test_db
Login/Group Roles
Tablespaces
bongo/admin@LocalPostgreSQL
No limit
Query
Query History
1 ALTER TABLE reserve ADD CONSTRAINT fk_sid FOREIGN KEY (sid) REFERENCES sailor(sid);
2
3 ALTER TABLE reserve ADD CONSTRAINT fk_bid FOREIGN KEY (bid) REFERENCES boat(bid);
4
5 ALTER TABLE reserve ADD CONSTRAINT pk_reserve PRIMARY KEY (sid, bid);
6
7
Data Output
Messages
Notifications
ALTER TABLE
Query returned successfully in 88 msec.

```
bongo=# \d reserve
                    Table "public.reserve"
 Column |          Type          | Collation | Nullable | Default
--------+------------------------+-----------+----------+---------
 sid    | character varying(2)   |           | not null |
 bid    | character varying(3)   |           | not null |
 day    | date                   |           |          |
Indexes:
    "pk_reserve" PRIMARY KEY, btree (sid, bid)
Foreign-key constraints:
    "fk_bid" FOREIGN KEY (bid) REFERENCES boat(bid)
    "fk_sid" FOREIGN KEY (sid) REFERENCES sailor(sid)
```

Drop database/table

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

```
bongo=# drop table sailor;
ERROR:  cannot drop table sailor because other objects depend on it
DETAIL:  constraint fk_sid on table reserve depends on table sailor
HINT:  Use DROP ... CASCADE to drop the dependent objects too.
```

```
bongo=# drop table reserve;
DROP TABLE
bongo=# \d
          List of relations
 Schema |  Name  | Type  |  Owner
--------+--------+-------+----------
 public | boat   | table | postgres
 public | sailor | table | postgres
(2 rows)
```

Drop database/table

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

Add Table reserve back to the bongo db and also alter table for the Referential integrity, composite key.

```
bongo=# drop table sailor cascade;
NOTICE:  drop cascades to constraint fk_sid on table reserve
DROP TABLE

bongo=# \d
           List of relations
 Schema |  Name   | Type  |  Owner
--------+---------+-------+----------
 public | boat    | table | postgres
 public | reserve | table | postgres
(2 rows)
```

Drop database/table

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

```
bongo=# \d reserve;
                  Table "public.reserve"
 Column |          Type          | Collation | Nullable | Default
--------+------------------------+-----------+----------+---------
 sid    | character varying(2)   |           | not null |
 bid    | character varying(3)   |           | not null |
 day    | date                   |           |          |
Indexes:
    "pk_reserve" PRIMARY KEY, btree (sid, bid)
Foreign-key constraints:
    "fk_bid" FOREIGN KEY (bid) REFERENCES boat(bid)
```

Referential integrity about sid (single reference) just gone

Drop database/table

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

Add table sailor back to the bongo db also need to add referential integrity in reserve too!

```
ALTER TABLE reserve ADD CONSTRAINT fk_sid FOREIGN KEY (sid) REFERENCES
sailor(sid);
```

Drop database/table

```
ALTER TABLE reserve ADD CONSTRAINT fk_sid FOREIGN KEY (sid) REFERENCES
sailor(sid);
```

```
bongo=# \d reserve
                    Table "public.reserve"
 Column |          Type          | Collation | Nullable | Default
--------+------------------------+-----------+----------+---------
 sid    | character varying(2)   |           | not null |
 bid    | character varying(3)   |           | not null |
 day    | date                   |           |          |
Indexes:
    "pk_reserve" PRIMARY KEY, btree (sid, bid)
    "uc_combined" UNIQUE CONSTRAINT, btree (sid, bid)
Foreign-key constraints:
    "fk_bid" FOREIGN KEY (bid) REFERENCES boat(bid)
    "fk_sid" FOREIGN KEY (sid) REFERENCES sailor(sid)
```

Drop database/table

```
bongo=# drop table sailor restrict;
ERROR:  cannot drop table sailor because other objects depend on it
DETAIL:  constraint fk_sid on table reserve depends on table sailor
HINT:  Use DROP ... CASCADE to drop the dependent objects too.
```

Drop table w/ options

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

Drop database/table

- **DROP DATABASE** dbname;

ทบทวน DDL และอื่น ๆ

- `CREATE DATABASE dbname;`
- `CREATE TABLE tablename(…);`
- `ALTER TABLE ADD colname data_type;`
- `ALTER TABLE DROP colname;`
- `DROP TABLE table_name;`
- `DROP DATABASE dbname;`

ทบทวน DDL และอื่น ๆ

MySQL	PostgreSQL
SHOW DATABASES;	\l
USE [dbname];	\c [dbname]
SHOW TABLES;	\d
DESCRIBE[table name]	\d [table name]

DML (Data Manipulation Language) commands (หลัก ๆ)

โปรแกรมเมอร์เรียกคำสั่งเหล่านี้ว่าเป็น “CRUD” คำถามคือ CRUD คือ?

SELECT ใช้ในการ query หรือสืบค้นข้อมูล

INSERT ใช้ในการเพิ่มข้อมูลรายการใหม่เข้าตารางต่าง ๆ ในฐานข้อมูล

UPDATE ใช้ในการแก้ไขรายการข้อมูลต่าง ๆ ของตาราง ในฐานข้อมูล

DELETE ใช้ในการลบรายการข้อมูลต่าง ๆ ของตาราง ในฐานข้อมูล

รูปแบบพื้นฐานในการทำ SQL queries

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

- *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 $<, >, \leq, \geq, =, \neq$) combined using AND, OR and NOT.

องค์ประกอบน้อยสุดของ 1 query

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)

Sailors

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

Reserves

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

Bongo's data

Boats

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

ลองคำสั่งต่อไปนี้

- **\c** bongo
- **\d**
- **\d** sailor;
- **SELECT** * **FROM** sailor;

Insert data (SAILOR)

optional

- **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);
```

```
bongo=# select * from sailor;
 sid |  sname   | rating |  age
-----+----------+--------+-------
 22  | Dustin   |      7 | 45.00
 29  | Brutus   |      1 | 33.00
 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
(10 rows)
```

ให้เวลาในการ insert data เข้าทั้ง 3 ตาราง 1 นาที 

ให้เพิ่มข้อมูลลงในทั้ง 3 ตาราง โดยใช้คำสั่ง INSERT INTO
โดย copy คำสั่ง จากไฟล์ PostgreSQL_ToTry_commands_2024.txt มาใส่เป็นคำสั่งใน shell ของ Query Tool / PSQL Tool ได้เลย ☺

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'),
```

นี่เป็นข้อดีของ RDBMS มีการเช็ค integrity

MySQL ก็พ่น error คล้ายกัน

```
ERROR:  Key (bid)=(101) is not present in table "boat".insert or update on table "reserve" violates foreign
key constraint "fk_bid"

ERROR:  insert or update on table "reserve" violates foreign key constraint "fk_bid"
SQL state: 23503
Detail: Key (bid)=(101) is not present in table "boat".
```

Insert data (BOAT)

optional

- **INSERT INTO** table_name (colname, colname,…) **VALUES** (…),(…),(…),…;

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

```
bongo=# select * from boat;
 bid |    bname    | color
-----+-------------+-------
 101 | Interlake   | Blue
 102 | Interlake   | Red
 103 | Clipper     | Green
 104 | Marine      | Red
(4 rows)
```

ให้ insert data เข้าตาราง reserve อีกที แล้วลอง select ดู

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'),
('74','103','2009-08-14');
```

optional

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

DML (Data Manipulation Language) commands

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

- `Target-list` ระบุฟิลด์หรือชุดของฟิลด์ที่ต้องการแสดง
- `Relation-list` ระบุรีเลชั่นหรือชุดของรีเลชั่นที่ต้องการดึงข้อมูลออกมา
- `Qualification` ระบุเงื่อนไขที่ใช้ในการดึงข้อมูล

(1) SQL query: แสดงรายชื่อและอายุของกะลาสีทั้งหมดที่อยู่ในตาราง 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

reserve

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

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

boat

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

(1) SQL query: สอบถามและแสดงรายชื่อและอายุของกะลาสีทั้งหมดในตาราง 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)
```

(2) SQL query: แสดงรายชื่อที่ไม่ซ้ำกันและอายุของกะลาสีทั้งหมดในตาราง 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)
```

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

(3) SQL query: แสดงรายชื่อกะลาสีทั้งหมดที่เคยจองเรือหมายเลข 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

reserve

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;
```

(3) SQL query: แสดงรายชื่อกะลาสีทั้งหมดที่เคยจองเรือหมายเลข 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';
```

อันนี้ต่างจาก MySQL เพราะว่า MySQL ยอม แต่ ProtgreSQL ไม่ยอม

(3) SQL query: แสดงรายชื่อกะลาสีทั้งหมดที่เคยจองเรือหมายเลข 103

Relational Algebra:

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

sid	*bid*	*day*
22	103	10/08/14
31	103	11/06/14
74	103	09/08/14

$\bowtie$

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

Result:

sname
Dustin
Lubber
Horatio

(4) SQL query: แสดงรหัสของกะลาสีทั้งหมดที่เคยจองเรือสีแดง

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

reserve

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 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

(4) SQL query: แสดงชื่อของกะลาสีทั้งหมดที่เคยจองเรือสีแดง

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

reserve

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

(4) SQL query: แสดงชื่อของกะลาสีทั้งหมดที่เคยจองเรือสีแดง

```
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)
```

(5) SQL query: แสดงสีของเรือทั้งหมดที่เคยถูกจองโดยกะลาสี Lubber

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

reserve

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

จาก Query ข้างต้น ต้องใช้ข้อมูลกี่ตาราง

boat

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

(5) SQL query: แสดงสีของเรือทั้งหมดที่เคยถูกจองโดยกะลาสี 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;
```

```
 color
-------
 Red
 Green
 Red
(3 rows)
```

(6) SQL query: แสดงชื่อของกะลาสีทั้งหมดที่เคยมีการจองเรืออย่างน้อย 1 ลำ

จาก Query ข้างต้น ถ้าเปลี่ยน sname เป็น sid จะได้ผลต่างกันไหม อย่างไร

```
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
(3 rows)
```

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

```
 sid
-----
 22
 22
 22
 22
 31
 31
 31
 64
 64
 74
(10 rows)
```

(7) SQL query: เปลี่ยนชื่อคอลัมน์ในการแสดงผล ผ่าน AS และสืบค้นสตริงผ่าน LIKE

- AS ใช้ เปลี่ยนชื่อคอลัมน์ที่ใช้แสดงผล
- LIKE ใช้ในการเทียบสตริงเพื่อดึงข้อมูล ทำ regular expression

คอลัมน์เหล่านี้เป็น derived attributes เกิดจากการคำนวณ

```
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
---------+--------+-------+--------
 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

(8) 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

(9) SQL query: เปลี่ยนชื่อคอลัมน์ในการแสดงผล ผ่าน 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)
```

จงแสดงชื่อกะลาสีทั้งหมดที่ลงท้ายด้วย y

```
bongo=# select sname from sailor where sname like '%y';
 sname
-------
 Andy
 Rusty
(2 rows)
```

```
bongo=# select bname from boat where bname like '%er%';
   bname
-----------
 Interlake
 Interlake
 Clipper
(3 rows)
```

จงแสดงชื่อเรือทั้งหมดที่มีคำว่า er ในชื่อ

(10) 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)

(11) 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)

Provide the same result as LIMIT but conforms to SQL standard

(12) 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)

(13) 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)

SQL query: UNION, INTERSECT, EXCEPT

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

(14) SQL query: แสดงชื่อของกะลาสีทั้งหมดที่เคยจองเรือสีเขียวหรือแดงก็ได้ (UNION)

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

reserve

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

(14.1) SQL query: แสดงชื่อของกะลาสีทั้งหมดที่เคยจองเรือสีเขียวหรือแดงก็ได้

```
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');
```

ถ้าเขียนอย่างนี้ ได้ผลอะไร

```
 sid |  sname
-----+---------
 22  | Dustin
 22  | Dustin
 22  | Dustin
 31  | Lubber
 31  | Lubber
 31  | Lubber
 64  | Horatio
 74  | Horatio
(8 rows)
```

(14.2) SQL query: แสดงชื่อของกะลาสีทั้งหมดที่เคยจองเรือสีเขียวหรือแดงก็ได้โดยใช้ 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 จะเอาสมาชิกที่ซ้ำออกโดยอัตโนมัติ

```
 sid |   sname
-----+----------
 74  | Horatio
 31  | Lubber
 22  | Dustin
 64  | Horatio
(4 rows)
```

(15) SQL query: แสดงชื่อของกะลาสีทั้งหมดที่เคยจองเรือทั้งสีเขียวและสีแดง (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??

แล้วต้องเขียนยังไง?

(15.1) SQL query: แสดงชื่อของกะลาสีทั้งหมดที่เคยจองเรือทั้งสีเขียวและสีแดงโดยใช้ 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)
```

(15.2) SQL query: แสดงชื่อของกะลาสีทั้งหมดที่เคยจองเรือทั้งสีเขียวและสีแดง 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)
```

(16) SQL query: แสดงชื่อของกะลาสีทั้งหมดที่เคยจองเรือสีเขียวแต่ไม่เคยจองสีแดงเลย

ใครช่วยตอบที

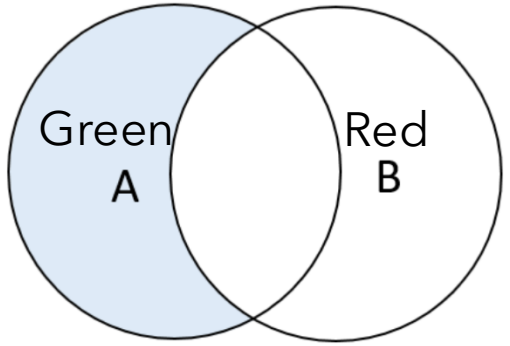
(16.1) SQL query: แสดงชื่อของกะลาสีทั้งหมดที่เคยจอง เรือสีเขียวแต่ไม่เคยจองสีแดงเลย โดยใช้ Nested Query และ NOT IN

```
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)
```

(16.2) SQL query: แสดงชื่อของกะลาสีทั้งหมดที่เคยจอง เรือสีเขียวแต่ไม่เคยจองสีแดงเลย 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)
```

Aggregate operators

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

ถ้าใช้ DISTINCT ประกอบจะนับแต่ค่าที่ไม่ซ้ำ

(17) SQL query: นับจำนวนกะลาสีทั้งหมดในตาราง 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

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

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

ถ้าใช้คำสั่ง

SELECT COUNT (DISTINCT S.sname) FROM sailor S;

ได้จำนวนกะลาสีกี่คน??

(18) SQL query: หาค่าอายุรวมของกะลาสีทุกคนในตาราง sailor ที่มีค่า rating เท่ากับ 10

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

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

```
-------
 51.00
(1 row)
```

(19) SQL query: หาค่าอายุเฉลี่ยของกะลาสีทุกคนในตาราง sailor ที่มีค่า rating เท่ากับ 10

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

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

```
---------------------
 25.5000000000000000
(1 row)
```

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

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

(20) 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

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

ใครช่วยตอบที

TikTok TikTok Tiktok

```
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;
```

(20) SQL query: แสดงชื่อและอายุของกะลาสีที่มีอายุมากที่สุด

```
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 คนได้ไหม

(21) SQL query: เปลี่ยนอายุของ Brutus เป็น 63.5

ตอบคำถามหน้าที่ผ่านมา มีผลลัพธ์มากกว่า 1 คนได้!!

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

```
UPDATE sailor
SET age = 63.5
WHERE sid = '29';
```

```
SELECT * FROM sailor;
```

```
 sid |  sname  | rating |  age
-----+---------+--------+-------
 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);
```

```
 sname  |  age
--------+-------
 Bob    | 63.50
 Brutus | 63.50
(2 rows)
```

จงแสดงชื่อกะลาสีที่มีอายุน้อยสุด

BETWEEN and AND

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

(21) SQL query: เปลี่ยนอายุของ Brutus เป็น 33.0 เหมือนเดิม

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

(22) SQL query: หาชื่อกะลาสีทั้งหมดที่มีอายุระหว่าง 25 ถึง 35 ปี

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 sid, sname
FROM sailor
WHERE age BETWEEN 25 AND 35;
```

ไม่ใช่ชื่อย่อ อย่าง S เพื่อแทน sailor ก็ได้ แต่ใช้ใน query ยาว ๆ ก็สะดวกขึ้น

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

BETWEEN and AND นี้เป็น inclusive หรือ exclusive??

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

ถ้าเป็น A ใหญ่หรือ a เล็กก็ได้ เขียนอย่างไร?

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

แล้ว Query นี้ทำอะไร

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

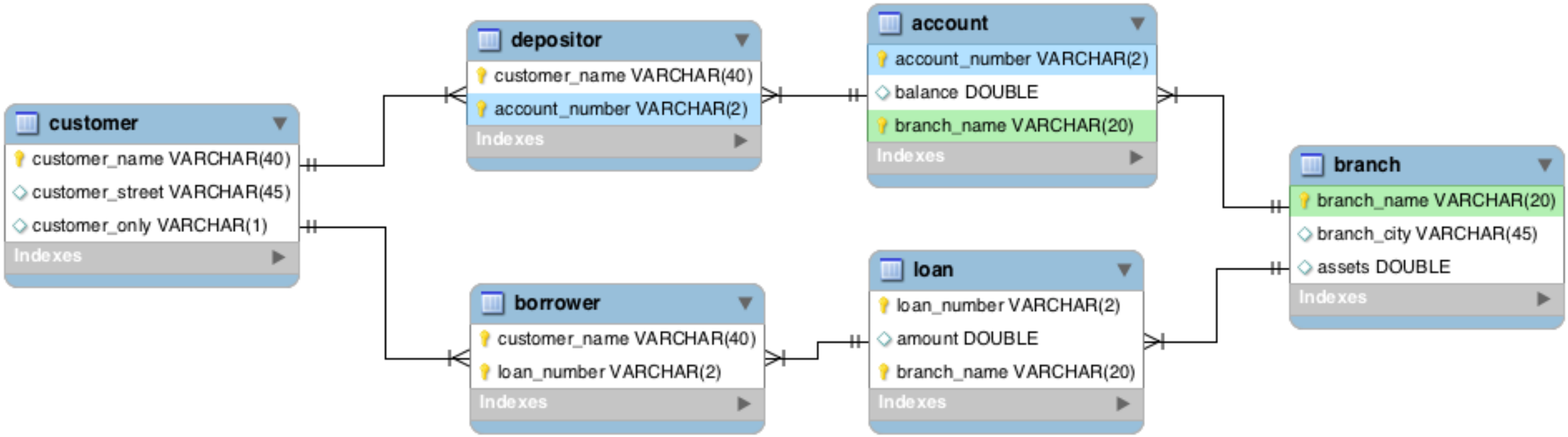
```
SELECT S.sname, MAX(S.age)
FROM sailor;
```

ถึงจุดนี้เราเรียนอะไรไปบ้าง

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

`GROUP BY`	ใช้รวมกลุ่มข้อมูลที่สืบค้นมาได้
`HAVING`	ใช้ระบุเงื่อนไขในการคัดกรองกลุ่มข้อมูลหลัง group มาแล้ว
`ORDER BY`	ใช้จัดลำดับของรายการในการแสดงผล

Banking ERR diagram

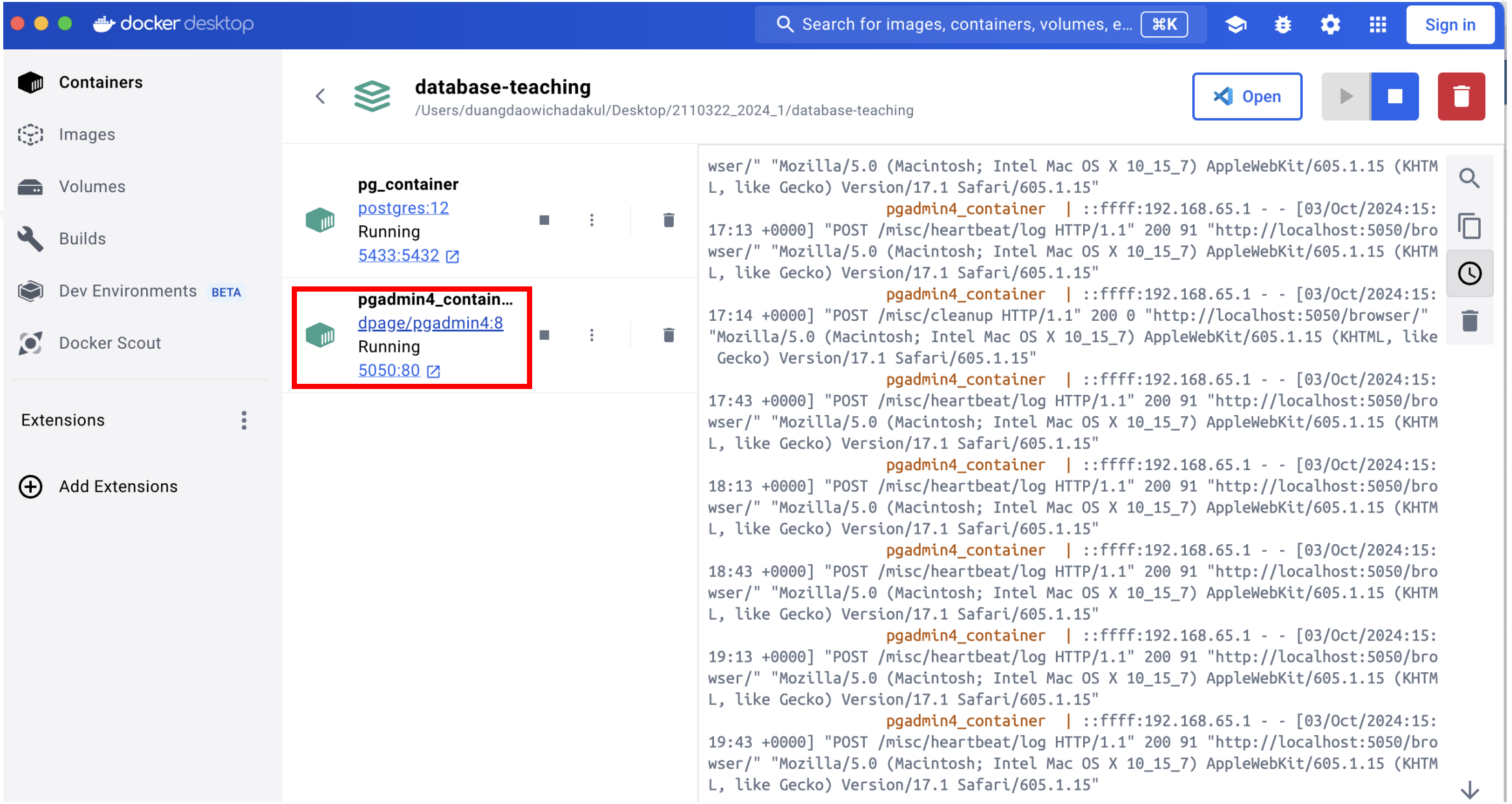


Banking's RDBMS schema design

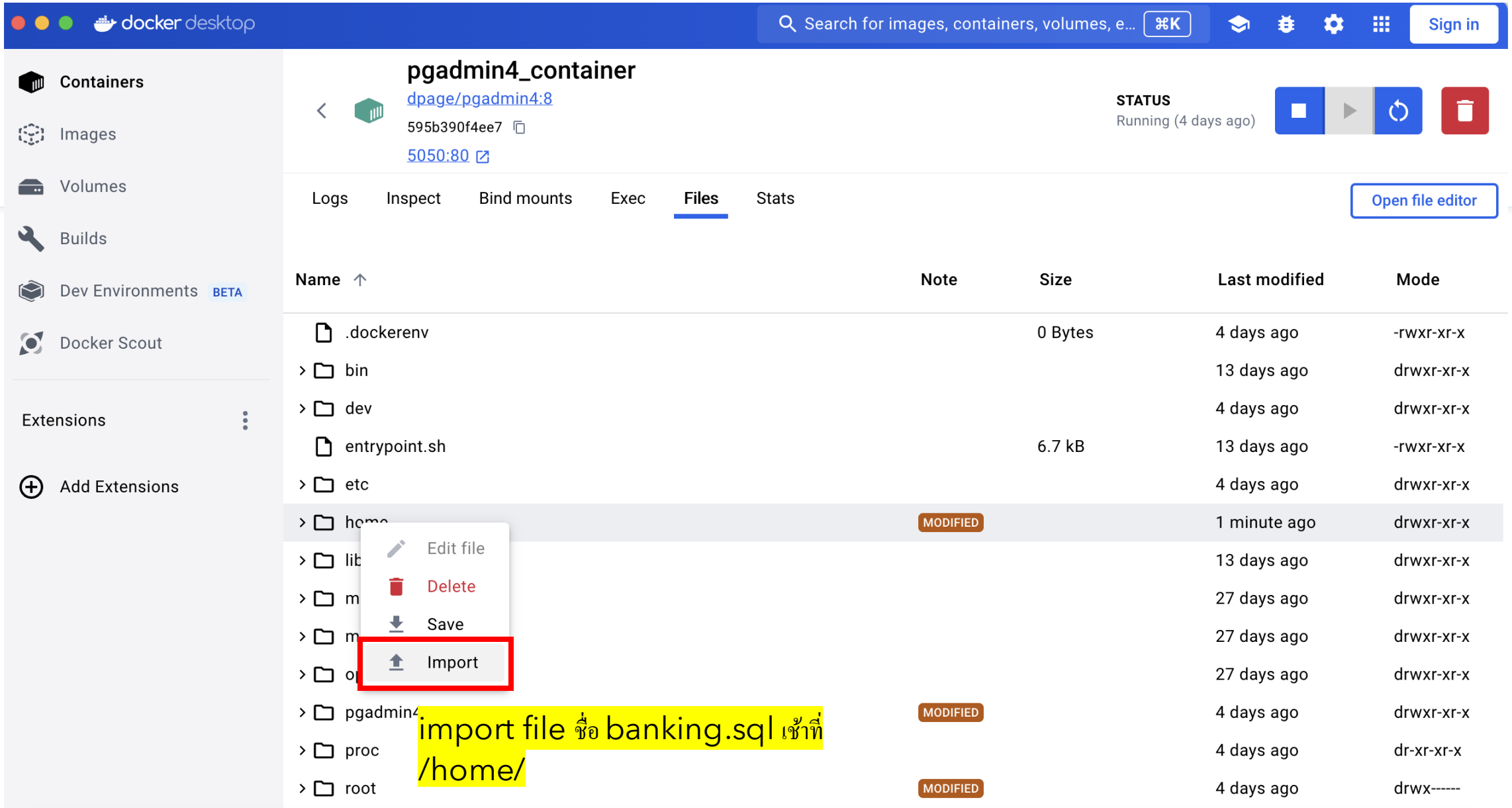
- branch(branch_name, branch_city, branch_assets)
- customer(customer_name, customer_street, customer_only)
- account(account_number, branch_name, balance)
- loan(loan_number, branch_name, amount)
- depositor(customer_name, account_number)
- borrower(customer_name, loan_number)

สร้างฐานข้อมูลใหม่ชื่อ banking ทำยังไงผ่าน pgAdmin4?

ใน docker desktop ไปที่ pgadmin4_container



นำข้อมูลเข้า pgadmin4_container



import file ชื่อ banking.sql เข้าที่ /home/

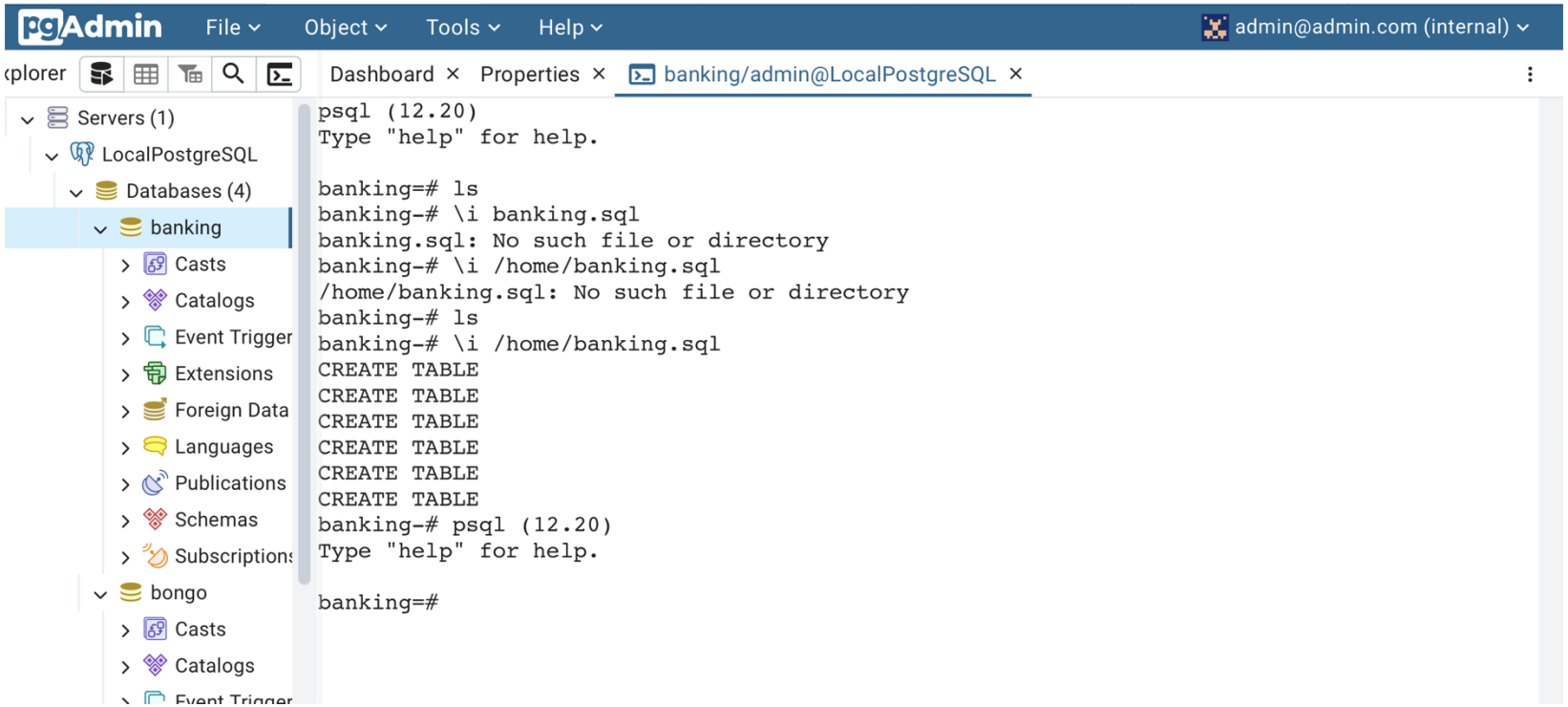
banking.sql

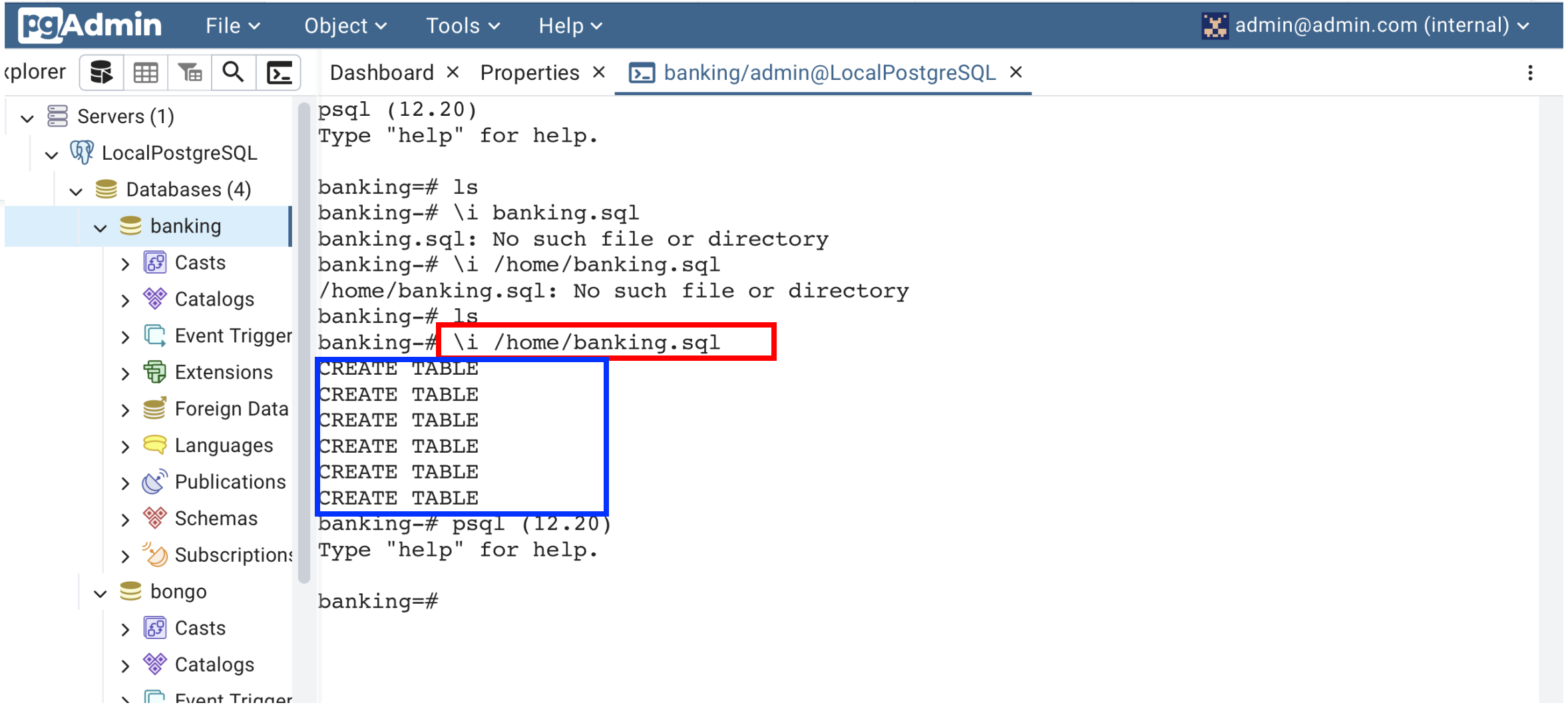
```sql
CREATE TABLE branch (
  branch_name varchar(20) NOT NULL,
  branch_city varchar(45) DEFAULT NULL,
  assets numeric(10,2) DEFAULT NULL,
  PRIMARY KEY (branch_name)
);

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)
);
```

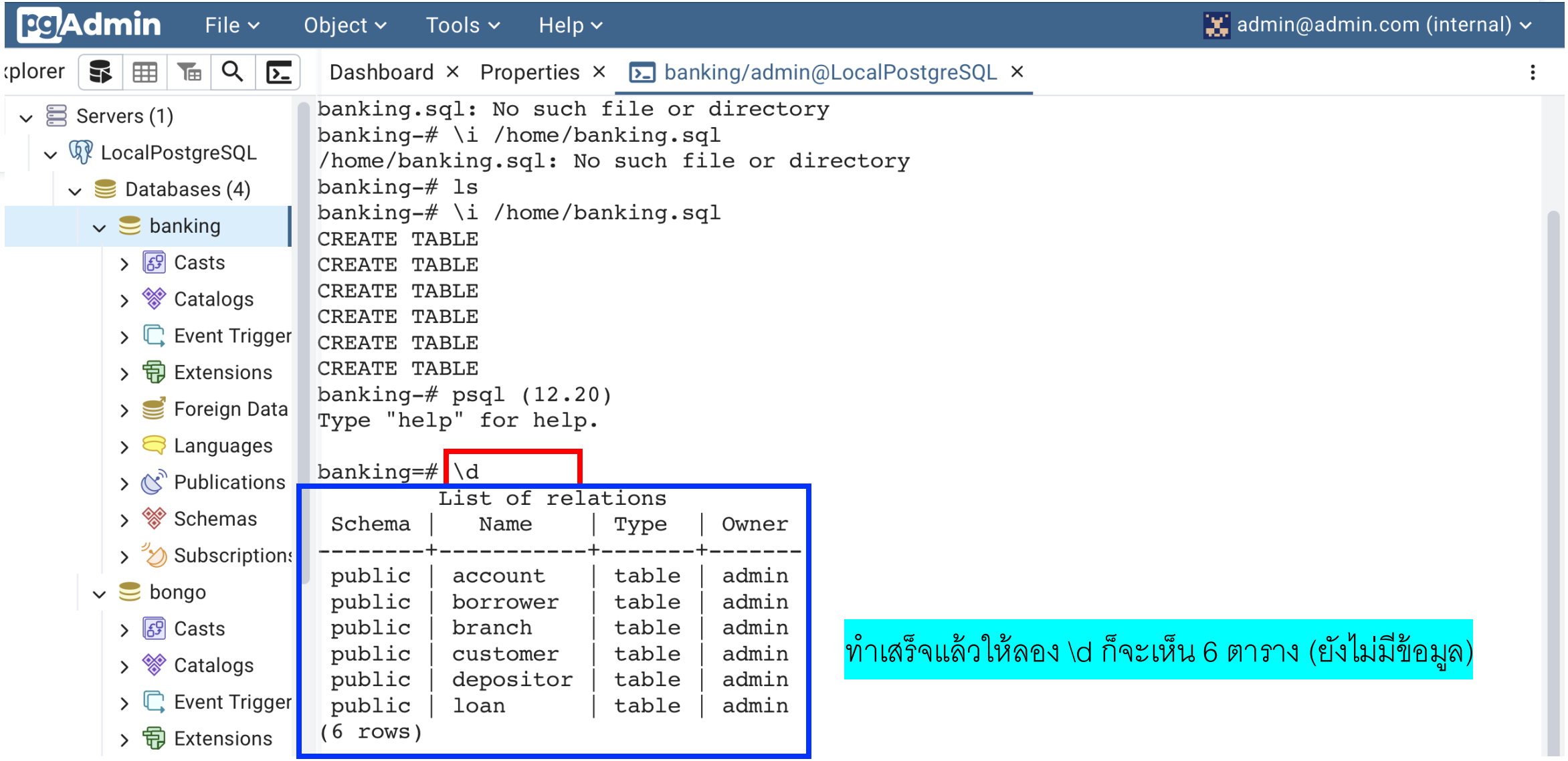
สร้าง 6 ตารางโดยเรียนคำสั่ง \i ผ่าน PSQL Tool

ที่ PSQL Tools prompt in pgAdmin4 GUI

\i [banking.sql with full path]




pgAdmin
File
Object
Tools
Help
admin@admin.com (internal)
Dashboard
Properties
banking/admin@LocalPostgreSQL
Servers (1)
LocalPostgreSQL
Databases (4)
banking
Casts
Catalogs
Event Trigger
Extensions
Foreign Data
Languages
Publications
Schemas
Subscriptions
bongo
Casts
Catalogs
psql (12.20)
Type "help" for help.
banking=# ls
banking-# \i banking.sql
banking.sql: No such file or directory
banking-# \i /home/banking.sql
/home/banking.sql: No such file or directory
banking-# ls
banking-# \i /home/banking.sql
CREATE TABLE
CREATE TABLE
CREATE TABLE
CREATE TABLE
CREATE TABLE
CREATE TABLE
banking-# psql (12.20)
Type "help" for help.
banking=#



ทำเสร็จแล้วให้ลอง \d ก็จะเห็น 6 ตาราง (ยังไม่มีข้อมูล)

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

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

borrower

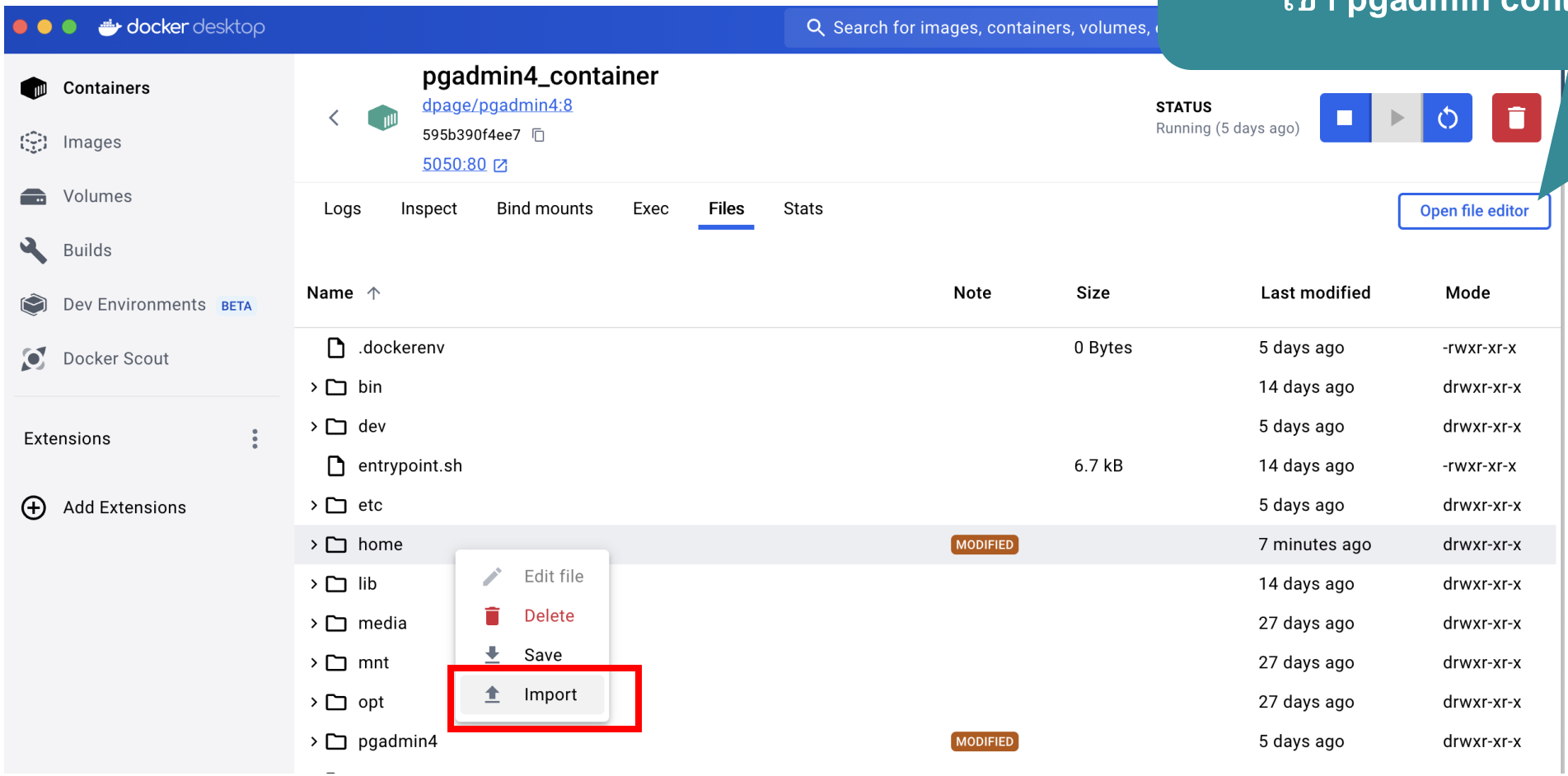
Customer_name	Loan_number
Joe	1
Jason	2
Joe	3
Keith	4
Mary	5
Joe	6

loan

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

เอาข้อมูลเข้าตาราง ไม่ใช้ INSERT แต่จะใช้ copy ข้อมูลจากไฟล์เข้าฐานข้อมูล ผ่าน PSQL Tool

ต้อง import files ข้อมูลชองแต่ละตาราง เข้า pgadmin container ก่อน



เอาข้อมูลเข้าตาราง ไม่ใช้ INSERT แต่จะใช้ copy ข้อมูลจากไฟล์เข้าฐานข้อมูล ผ่าน PSQL Tool

```
\copy branch from '[path to file]/branch.csv' DELIMITER ',' CSV;
\copy customer from '[path to file]/customer.csv' DELIMITER ',' CSV;
\copy account from '[path to file]/account.csv' DELIMITER ',' CSV;
\copy loan from '[path to file]/loan.csv' DELIMITER ',' CSV;
\copy depositor from '[path to file]/depositor.csv' DELIMITER ',' CSV;
\copy borrower from '[path to file]/borrower.csv' DELIMITER ',' CSV;
```

```
banking=# \copy customer FROM '/home/customer.csv' DELIMITER ',' CSV;
COPY 6
banking=# \copy account FROM '/home/account.csv' DELIMITER ',' CSV;
COPY 6
banking=# \copy borrower FROM '/home/borrower.csv' DELIMITER ',' CSV;
COPY 6
banking=# \copy branch FROM '/home/branch.csv' DELIMITER ',' CSV;
COPY 6
banking=# \copy depositor FROM '/home/depositor.csv' DELIMITER ',' CSV;
COPY 7
banking=# \copy loan FROM '/home/loan.csv' DELIMITER ',' CSV;
COPY 6
```

(24.1) 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_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

(24.2) SQL query: แสดงเลขที่บัญชีเงินฝากทั้งหมดที่เปิดในเมือง Riverside

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

เขียน Query โดยใช้ IN ก็ได้ผลแบบเดียวกัน

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

IN ต่างจาก = อย่างไร

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

SQL query: แสดงเลขที่บัญชีเงินฝากทั้งหมดของสาขา ‘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	B	$100
2	A	$50
3	A	$30
4	F	$120
5	A	$500
6	B	$324

SQL query: EXISTS และ NOT EXISTS

- EXISTS predicate เป็น *จริง* ก็ต่อเมื่อ sub-query ไม่เป็นเซ็ตว่าง
- NOT EXISTS predicate เป็น *จริง* ก็ต่อเมื่อ sub-query เป็นเซ็ตว่าง

NOT EXISTS ใช้ทดแทน MINUS operator ใน relational algebra ได้อย่างไร

(24.3) SQL query: แสดงเลขที่บัญชีเงินฝากทั้งหมดที่เปิดในเมือง 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

(24.4) SQL query: แสดง account number ทั้งหมดที่ไม่ได้อยู่ในสาขา 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

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

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 θ, representing some operator in the set {<, ≤, =, <>, >, ≥}, the equivalent predicates '*expr θ SOME (Subquery)*' *and* '*expr θ ANY (Subquery)*' are TRUE if and only if, for at least one element *s* returned by the *Subquery*, it is true that '*expr θ s*' is TRUE.
- the predicate '*expr θ ALL (Subquery)*' is TRUE if and only if '*expr θ s*' is TRUE for every one of the elements *s* of the *Subquery*;

(25) SQL query: หาเลขที่บัญชีเงินฝากที่มี balance น้อยสุด

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

```
 account_number
----------------
 3
(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

(26) SQL query: หา balances ของบัญชีเงินฝากทั้งหมดในเมือง Riverside โดยใช้ ANY

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

```
 balance
---------
   50.00
   30.00
  500.00
(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

(26) SQL query: หา balances ของบัญชีเงินฝากทั้งหมดในเมือง Riverside โดยใช้ SOME

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

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

(27) 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

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

(27) 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 func
tion
LINE 1: SELECT account_number, branch_name, SUM(balance)
               ^
```

(28) SQL query: รวมกลุ่มรายการที่ค้นหาได้โดยใช้ GROUP BY แสดงชื่อและ balance รวมของลูกค้าเงินฝากแต่ราย

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

```
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)
```

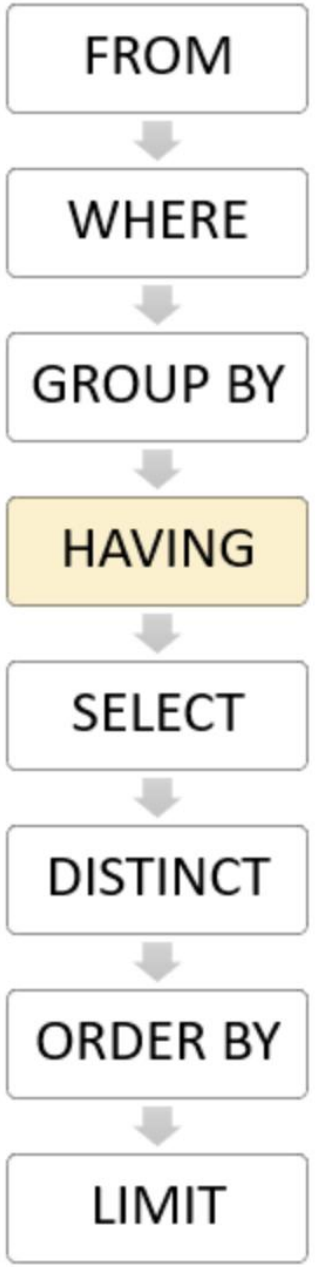
การกรองการจัดกลุ่ม

- 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:

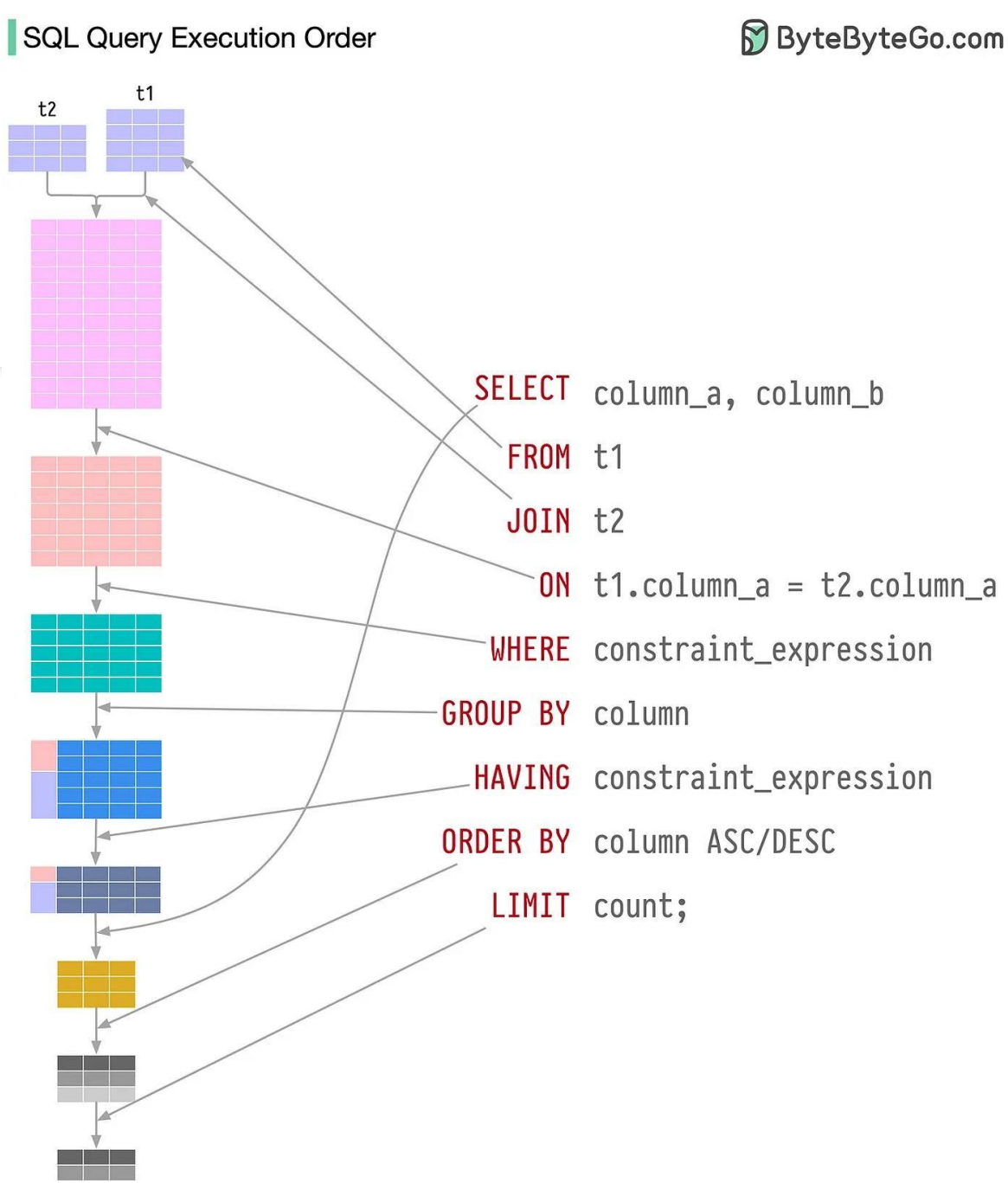
  SELECT account_branch, SUM(balance) FROM account
  GROUP BY account_branch HAVING SUM(balance)>1000.

  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



ลำดับของการ execute ส่วนของคำสั่งใน PostgreSQL



https://blog.bytebytego.com/p/ep50-visualizing-a-sql-query

(28) SQL query: คัดกรองผลของการรวมกลุ่มโดย GROUP BY โดยเอาเฉพาะที่ผ่านเงื่อนไขของ 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;
```

อันนี้ต่างจาก MySQL เพราะใช้ alias คือ sum_balance ไม่ได้

ข้อมูล Mary หายไปเพราะ balance รวมของ Mary (50) < 100

```
 customer_name |  sum_balance
---------------+--------------
 Joe           |        180.00
 Mike          |        500.00
 Keith         |        444.00
(3 rows)
```

(29) SQL query: คัดกรองผลของการรวมกลุ่มโดย GROUP BY โดยเอาเฉพาะที่ผ่านเงื่อนไขของ 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

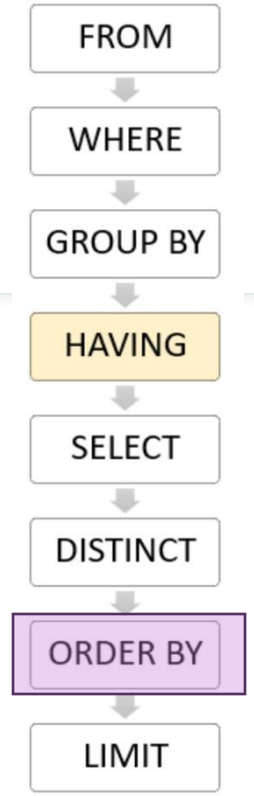
depositor

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

(30) 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


```
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.
```

(31) 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;
```

เป็น join ประเภทใดใน relational algebra?
e.g., Conditional/Theta 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)

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

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

(32) SQL query: ลองใช้ NATURAL JOIN

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

JOIN โดยใช้ key ร่วมคือ branch_name และเอาคอลัมน์ branch_name ที่ซ้ำออกโดยอัตโนมัติ

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*

(33) 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 ใช้ทำอะไรได้บ้าง?

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

(34) 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)
```

(35) 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

(36) 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

(10 rows)

(37) 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;
```

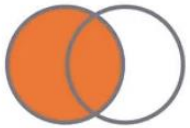
SQL query: ใน MySQL ไม่มี FULL OUTER JOIN แต่ใช้ UNION ทดแทนได้ ลองมาเขียนใน PostgreSQL

account_number	branch_name	balance	branch_name	branch_city	assets
			C	Long Beach	15000.00
6	B	324.00	B	LA	20000.00
4	F	120.00	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			

(10 rows)

1. ใครใช้วิธีนี้ก็ไม่ผิดอะไร แต่ตัวตรวจก็ต้องเอาผลมา sort ก่อน ยกเว้นโจทย์กำหนด output ในลักษณะจำเพาะ หรือมีการวัดประสิทธิภาพของ query ว่าทำงานเร็วช้า 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
```

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
```

RIGHT JOIN

Everything on the right
+
anything on the left that matches

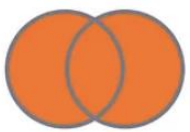
```
SELECT *
FROM TABLE_1
RIGHT JOIN TABLE_2
ON TABLE_1.KEY = TABLE_2.KEY
```

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
```

OUTER JOIN



Everything on the right
+
Everything on the left

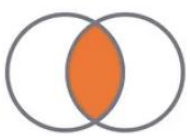
```
SELECT *
FROM TABLE_1
OUTER JOIN TABLE_2
ON TABLE_1.KEY = TABLE_2.KEY
```

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
```

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
```

CROSS JOIN

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

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

Taylor Brownlow, https://towardsdatascience.com/take-your-sql-from-good-to-great-part-3-687d797d1ede
[online accessed on Oct 5, 2024]

Relational algebra --> SQL command

Sailors

sid	*sname*	*rating*	*age*

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

SELECT S.sname, S.age FROM Sailors S

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

SELECT S.sname, S.age FROM Sailors S
WHERE S.age > 35 and S.rating >10

Relational algebra --> SQL command

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

Cartesian Product

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

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

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

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

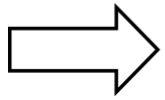
- “Compute available credit for every credit account.”

2. โจทย์อาจเขียนในรูปแบบ relational algebra แทนประโยคภาษาธรรมชาติ แล้วให้เขียน SQL

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

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

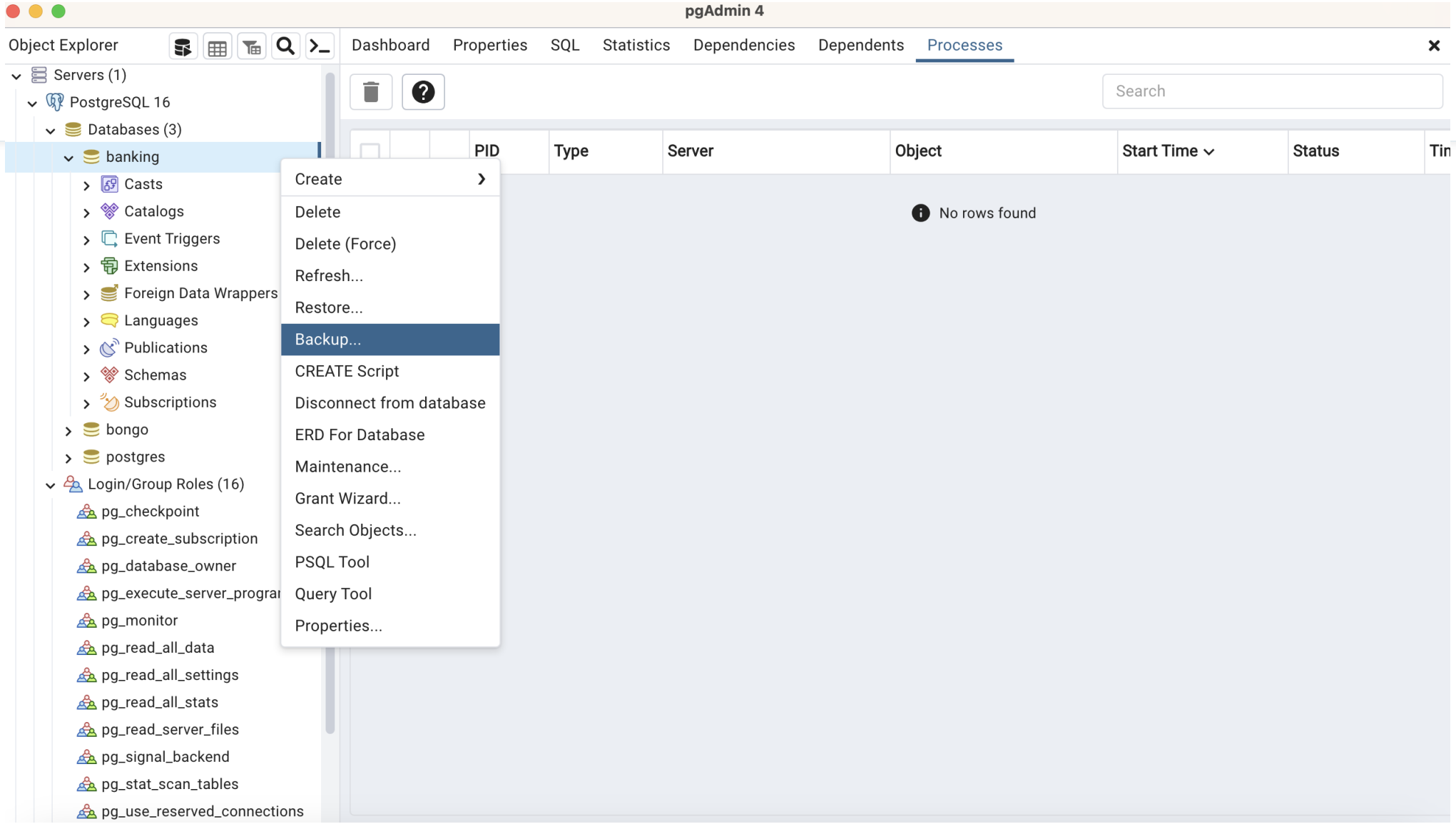
credit_acct



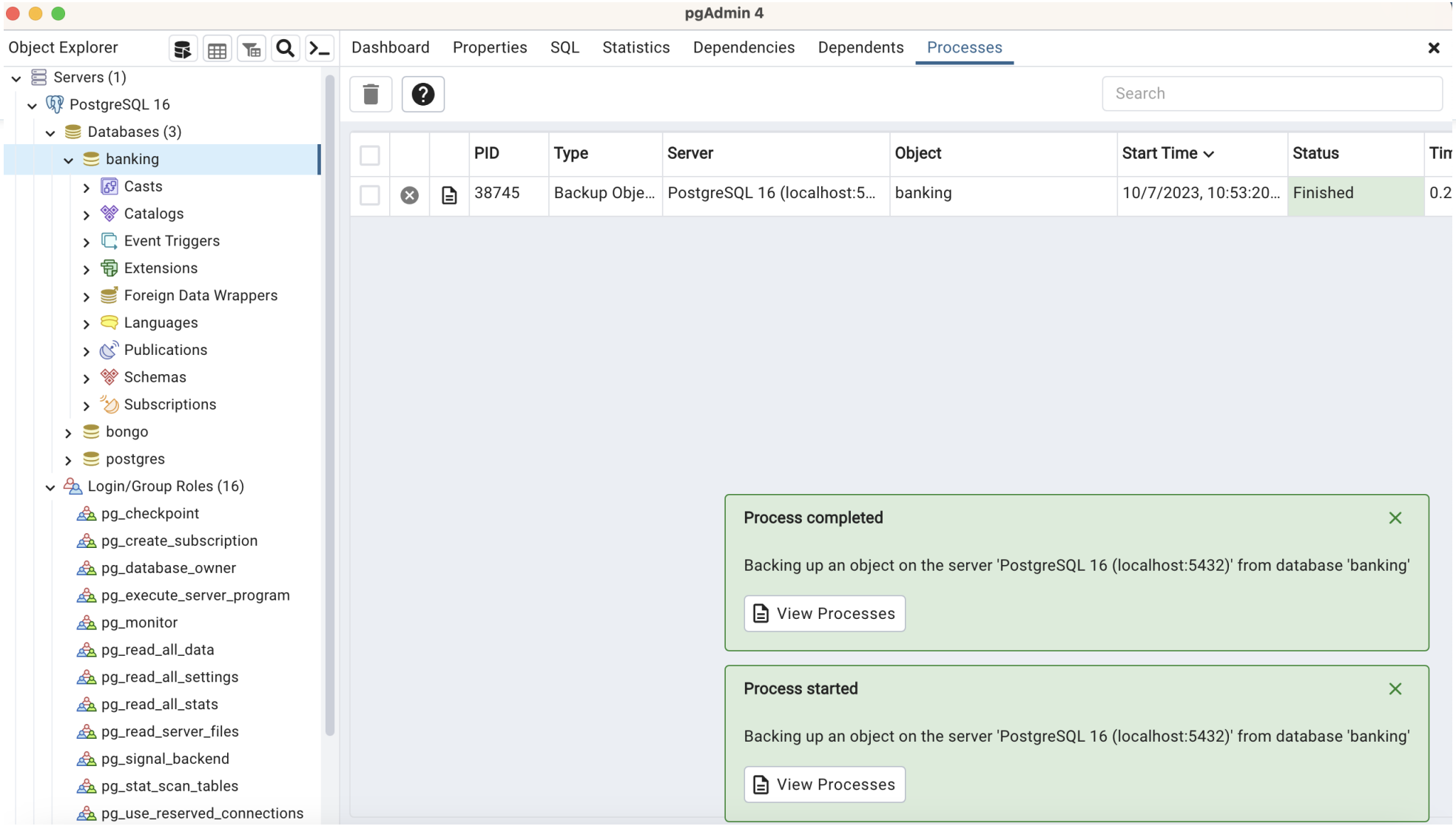
cred_id	available_credit
C-273	2350
C-291	150
C-304	11500
C-313	275

p. 6 http://users.cms.caltech.edu/~donnie/cs121/CS121Lec03.pdf / object not found anymore ☺ [online access on Oct 7, 2023]

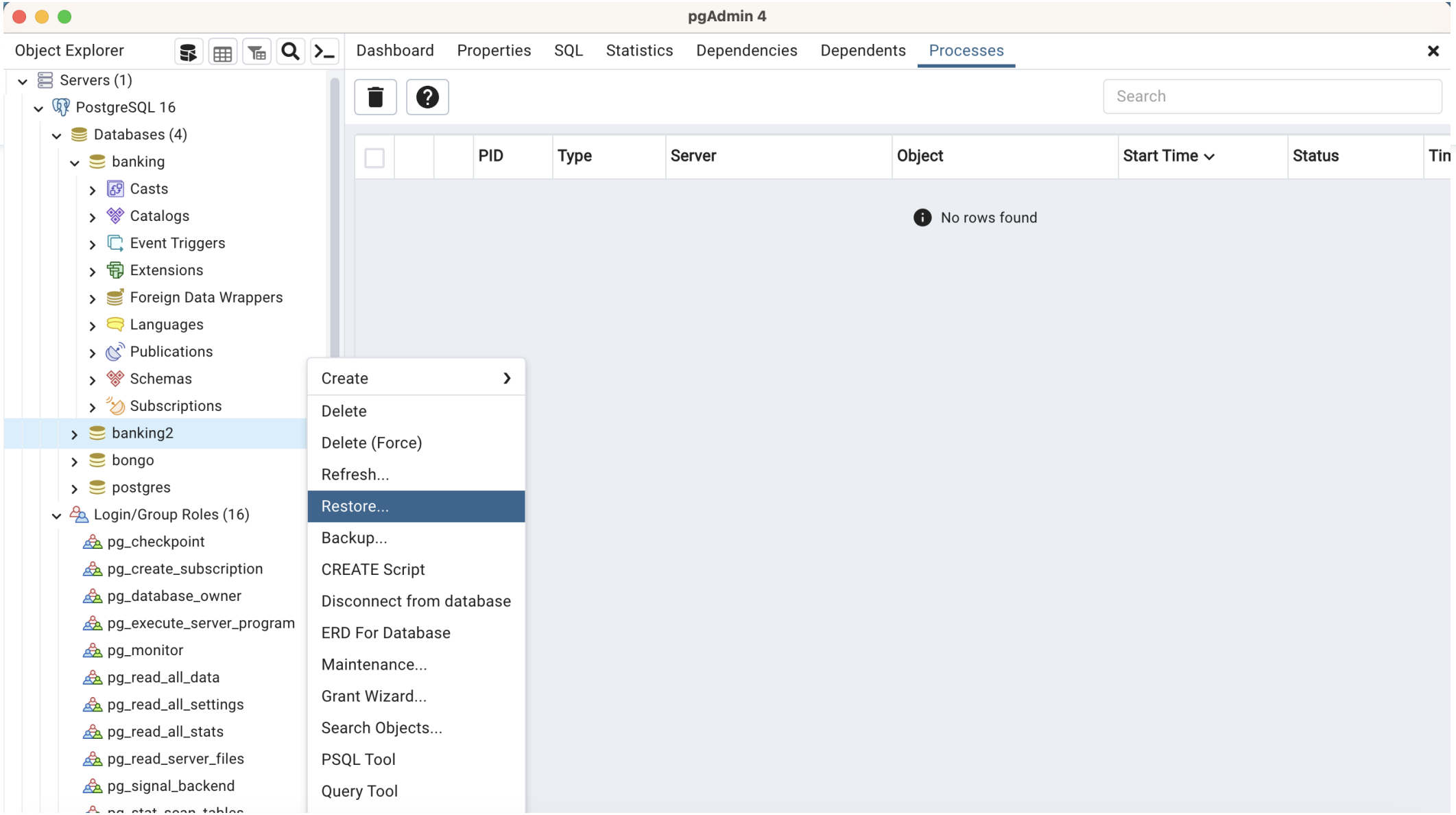
Create backup database โดยใช้ pgAdmin4



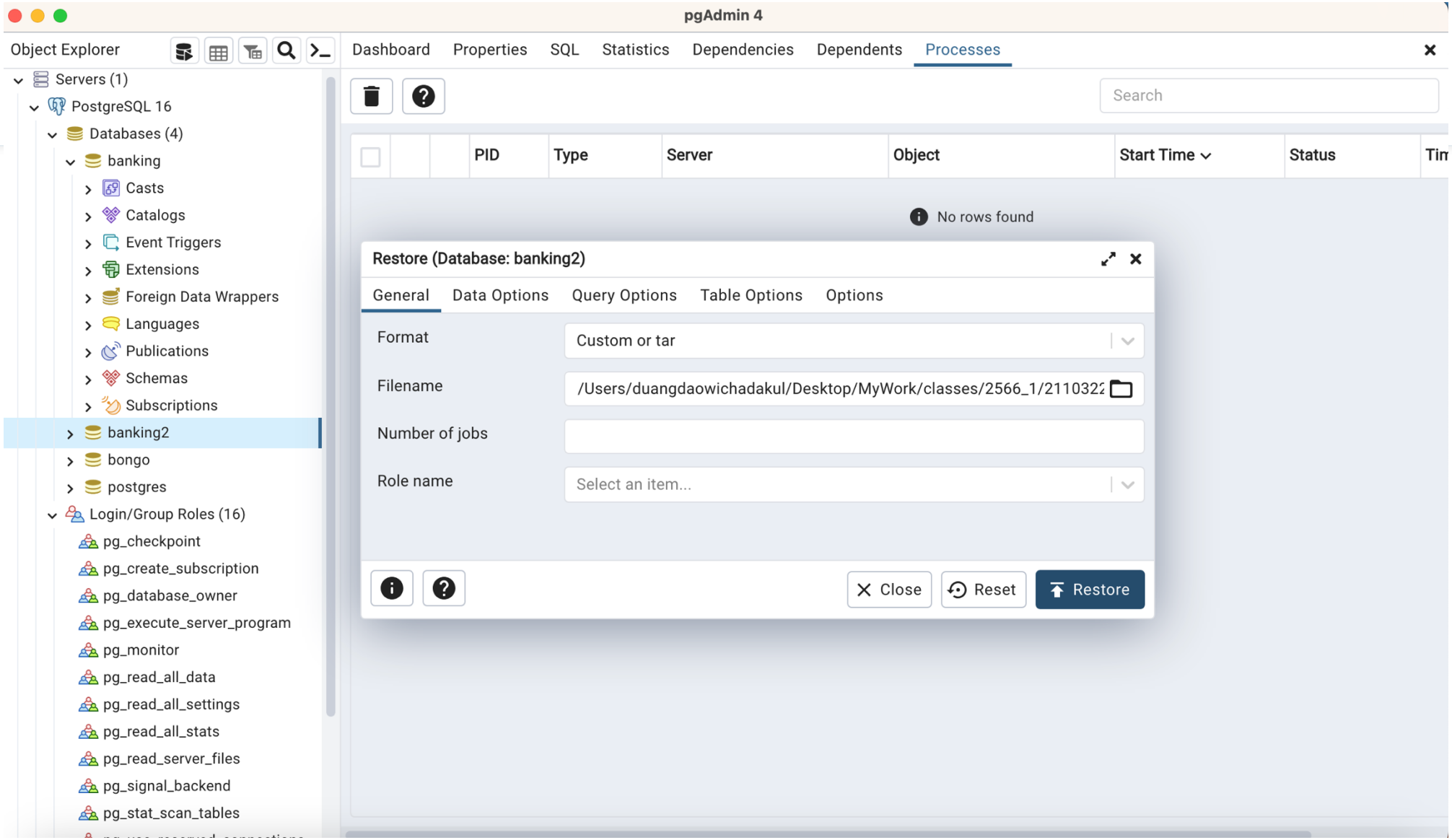
Create backup database โดยใช้ pgAdmin4



Restore database โดยใช้ pgAdmin4



Restore database โดยใช้ pgAdmin4



Comparison between PostgreSQL and MySQL

- https://www.postgresqltutorial.com/postgresql-tutorial/postgresql-vs-mysql/

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;`