Intro. to DBS - Assignment 1

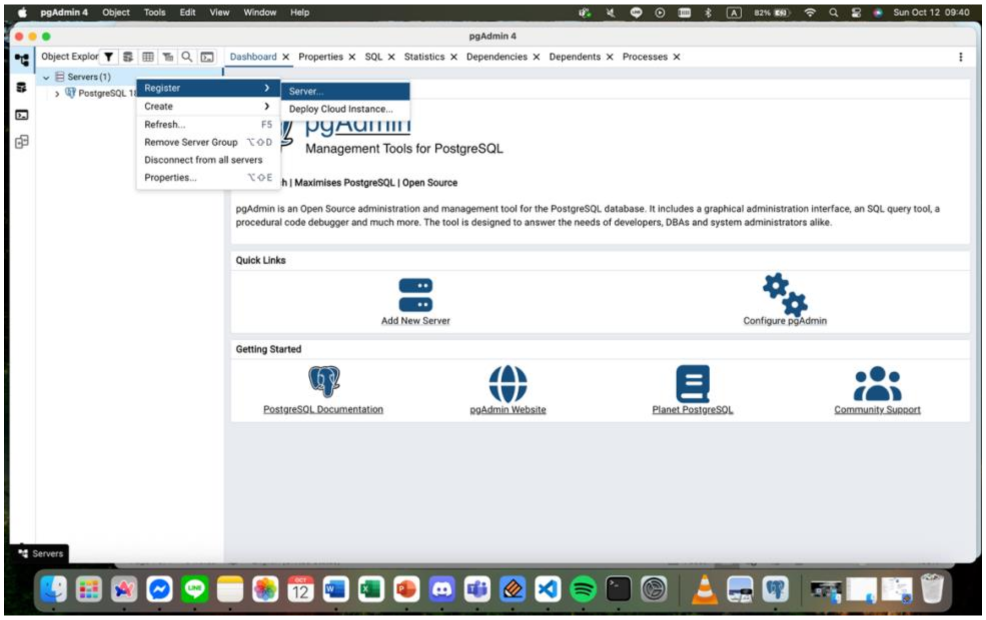
ID: 113550021　Name: 陳孟楷

Part1

1. Create Database

Step 1:

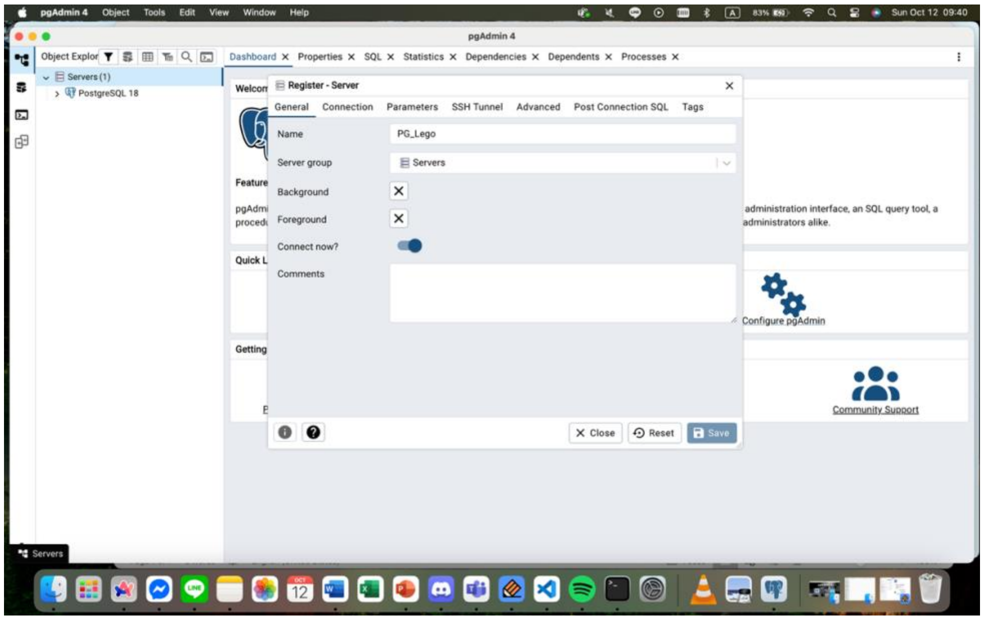
Go to **Servers** → **Register** → **Server...**



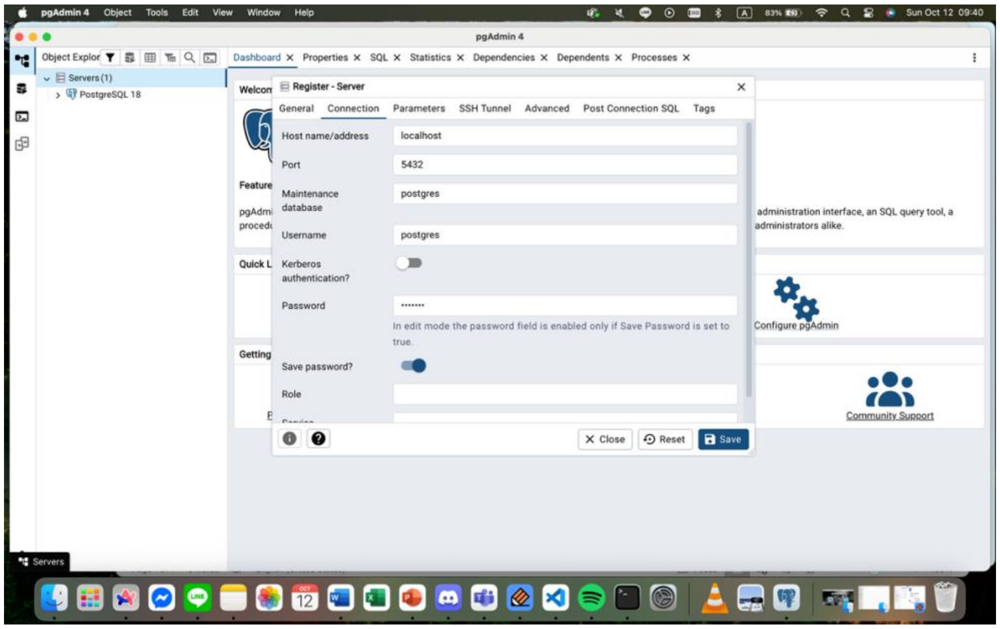
pgAdmin is an Open Source administration and management tool for the PostgreSQL database. It includes a graphical administration interface, an SQL query tool, a procedural code debugger and much more. The tool is designed to answer the needs of developers, DBAs and system administrators alike.

Step 2:

Set server name, host and password



Name	PG_Lego
Server group	Servers
Background	
Foreground	
Connect now?	●
Comments	

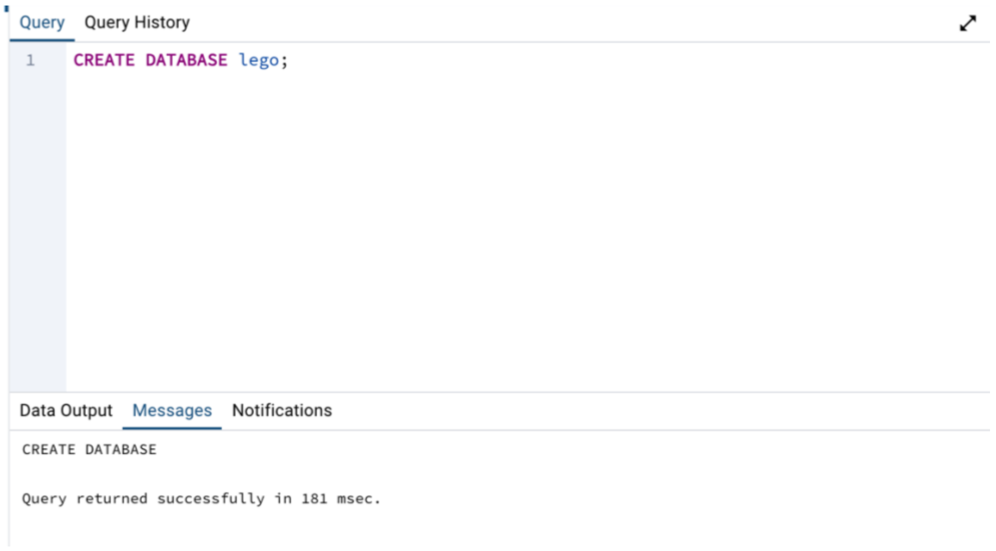


Host name/address	localhost
Port	5432
Maintenance database	postgres
Username	postgres
Kerberos authentication?	
Password	•••••••
Save password?	✓
Role	

In edit mode the password field is enabled only if Save Password is set to true.

Step 3:

Write codes and run



```sql
CREATE DATABASE lego;
```

```
CREATE DATABASE

Query returned successfully in 181 msec.
```

2. Import Data

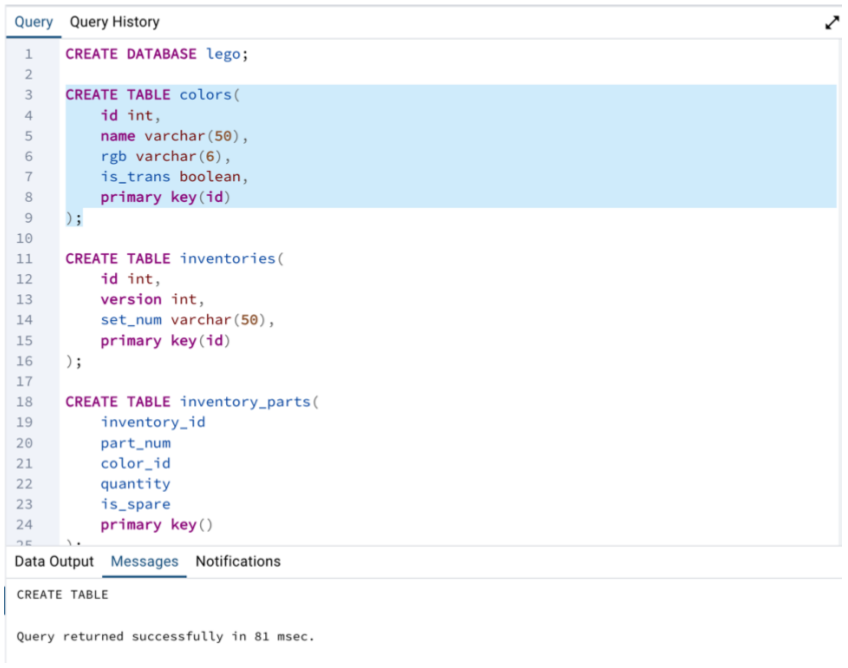
Step 1:

Download those 8 .csv files from Kaggle's website

Step 2:

Create tables for each csv file

- colors.csv
    - **id** is an integer and unique, so choose **id** as **primary key**
    - **name** is a string with different length, but all are less than 50
    - **rgb** is a string with 6-digits
    - **is_trans** is a boolean value



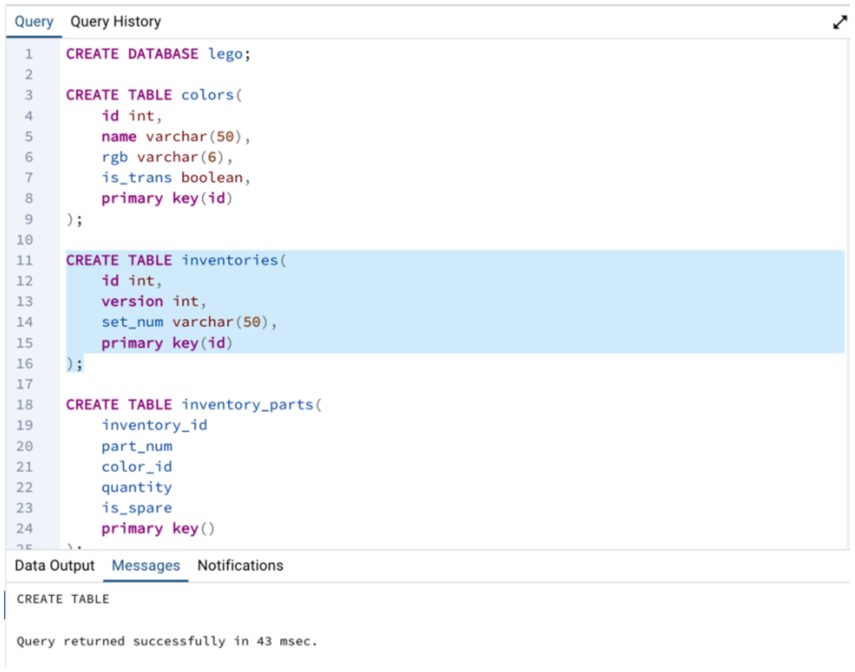
Query Query History

```
CREATE DATABASE lego;

CREATE TABLE colors(
    id int,
    name varchar(50),
    rgb varchar(6),
    is_trans boolean,
    primary key(id)
);

CREATE TABLE inventories(
    id int,
    version int,
    set_num varchar(50),
    primary key(id)
);

CREATE TABLE inventory_parts(
    inventory_id
    part_num
    color_id
    quantity
    is_spare
    primary key()
```

Data Output Messages Notifications

```
CREATE TABLE

Query returned successfully in 81 msec.
```

- inventories
    - **id** is an integer and unique, so choose **id** as **primary key**
    - **version** is an integer
    - **set_num** is a string with different length, but all are less than 50



Query Query History

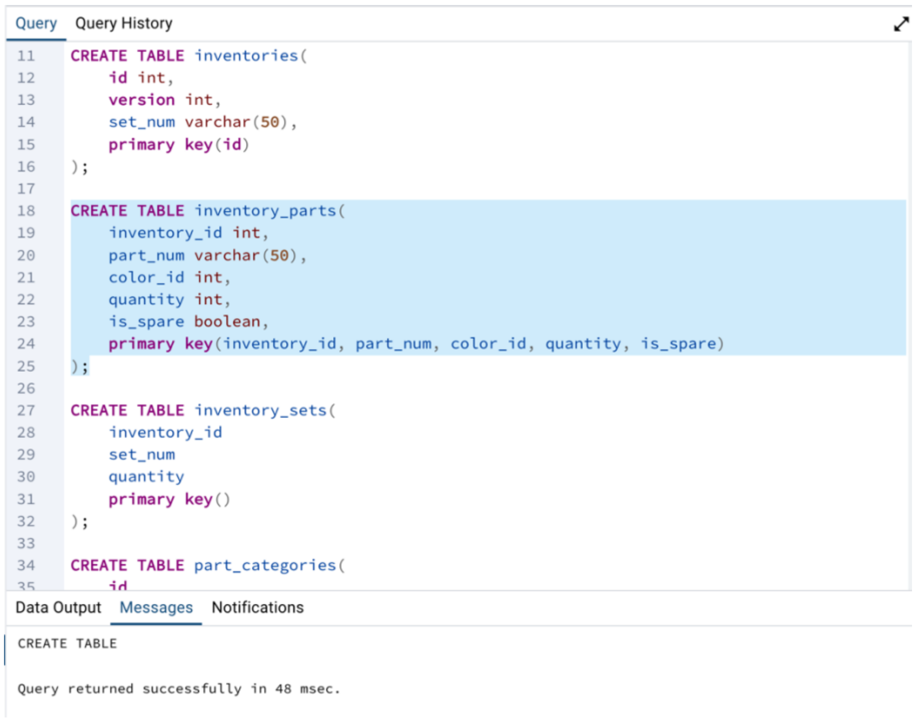
```
CREATE DATABASE lego;

CREATE TABLE colors(
    id int,
    name varchar(50),
    rgb varchar(6),
    is_trans boolean,
    primary key(id)
);

CREATE TABLE inventories(
    id int,
    version int,
    set_num varchar(50),
    primary key(id)
);

CREATE TABLE inventory_parts(
    inventory_id
    part_num
    color_id
    quantity
    is_spare
    primary key()
```

Data Output Messages Notifications

```
CREATE TABLE

Query returned successfully in 43 msec.
```

- inventory_parts
    - **inventory_id** is an integer
    - **part_num** is a string with different length, but all are less than 50
    - **color_id** is an interger
    - **quantity** is an integer
    - **is_spare** is a boolean value

    - **primary key** is the combination of above attributes since it is not unique if we only take some of them as the key.



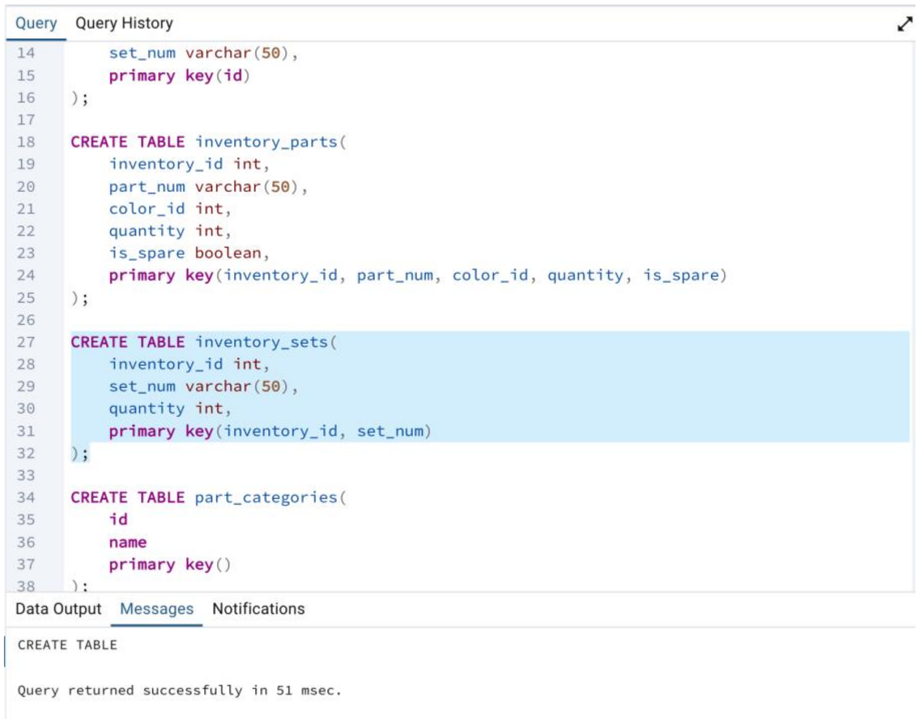
Query | Query History

```
CREATE TABLE inventories(
    id int,
    version int,
    set_num varchar(50),
    primary key(id)
);

CREATE TABLE inventory_parts(
    inventory_id int,
    part_num varchar(50),
    color_id int,
    quantity int,
    is_spare boolean,
    primary key(inventory_id, part_num, color_id, quantity, is_spare)
);

CREATE TABLE inventory_sets(
    inventory_id
    set_num
    quantity
    primary key()
);

CREATE TABLE part_categories(
    id
```

Data Output | Messages | Notifications

```
CREATE TABLE

Query returned successfully in 48 msec.
```

- inventory_sets
    - **inventory_id** is an integer
    - **set_num** is a string with different length, but all are less than 50
    - **quantity** is an integer
    - **primary key** is the combination of inventory_id and set_num, since it can identify a unique tuple



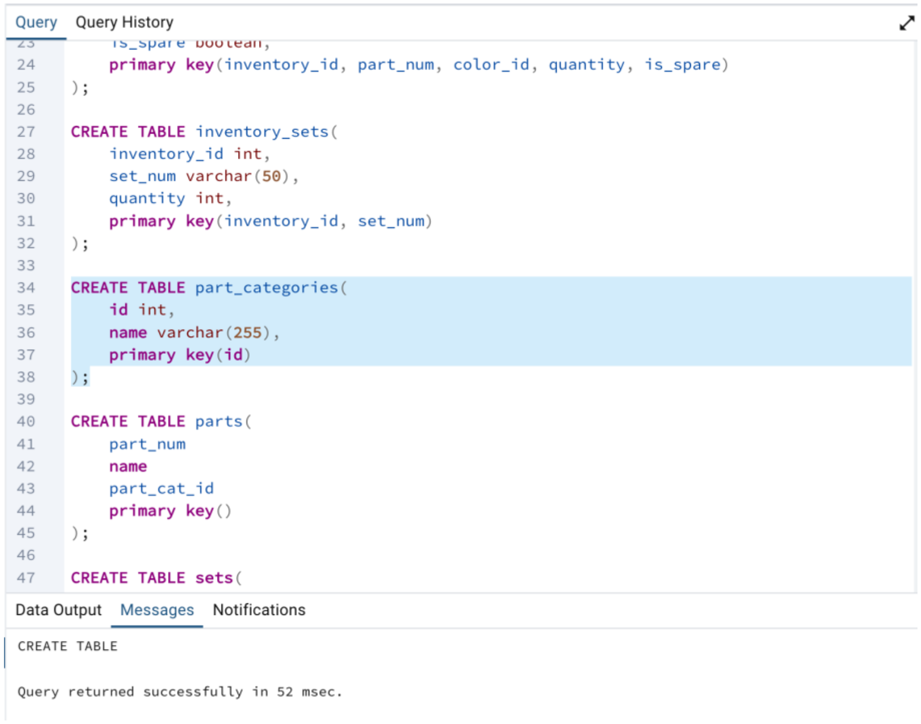
Query | Query History

```
    set_num varchar(50),
    primary key(id)
);

CREATE TABLE inventory_parts(
    inventory_id int,
    part_num varchar(50),
    color_id int,
    quantity int,
    is_spare boolean,
    primary key(inventory_id, part_num, color_id, quantity, is_spare)
);

CREATE TABLE inventory_sets(
    inventory_id int,
    set_num varchar(50),
    quantity int,
    primary key(inventory_id, set_num)
);

CREATE TABLE part_categories(
    id
    name
    primary key()
);
```

Data Output | Messages | Notifications

```
CREATE TABLE

Query returned successfully in 51 msec.
```

- part_categories
  - **id** is an integer and unique, so choose **id** as **primary key**
  - **name** is a string with different length, but all are less than 255

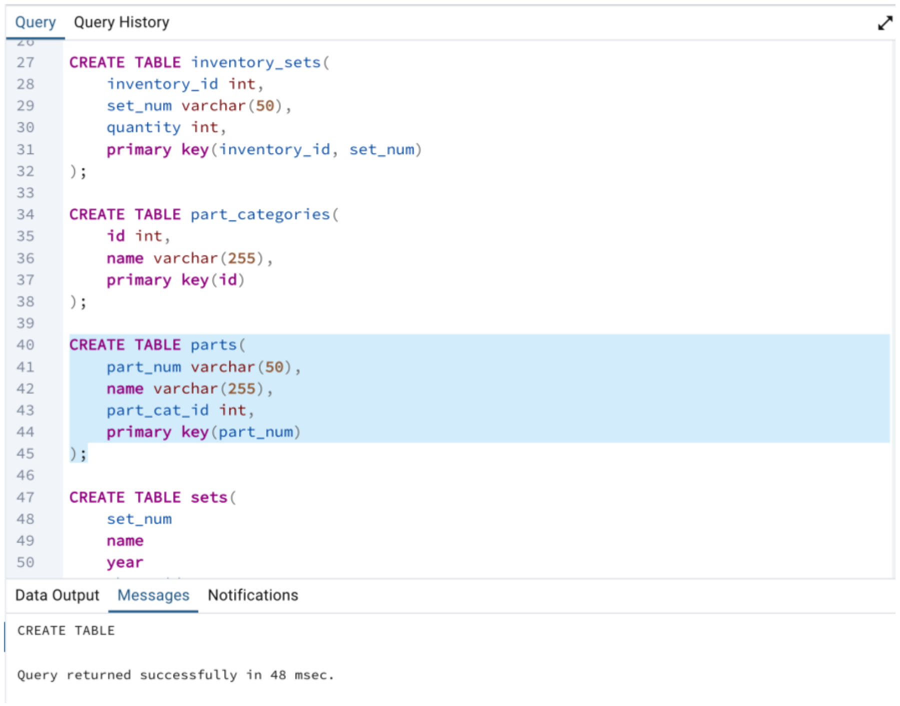

Query Query History

```
    is_spare boolean,
    primary key(inventory_id, part_num, color_id, quantity, is_spare)
);

CREATE TABLE inventory_sets(
    inventory_id int,
    set_num varchar(50),
    quantity int,
    primary key(inventory_id, set_num)
);

CREATE TABLE part_categories(
    id int,
    name varchar(255),
    primary key(id)
);

CREATE TABLE parts(
    part_num
    name
    part_cat_id
    primary key()
);

CREATE TABLE sets(
```

Data Output Messages Notifications

```
CREATE TABLE

Query returned successfully in 52 msec.
```

- parts
  - **part_num** is a string with different length, but all are less than 50. Also, it is unique, so it can be choosed as **primary key**
  - **name** is a string with different length, but all are less than 255
  - **part_cat_id** is an integer

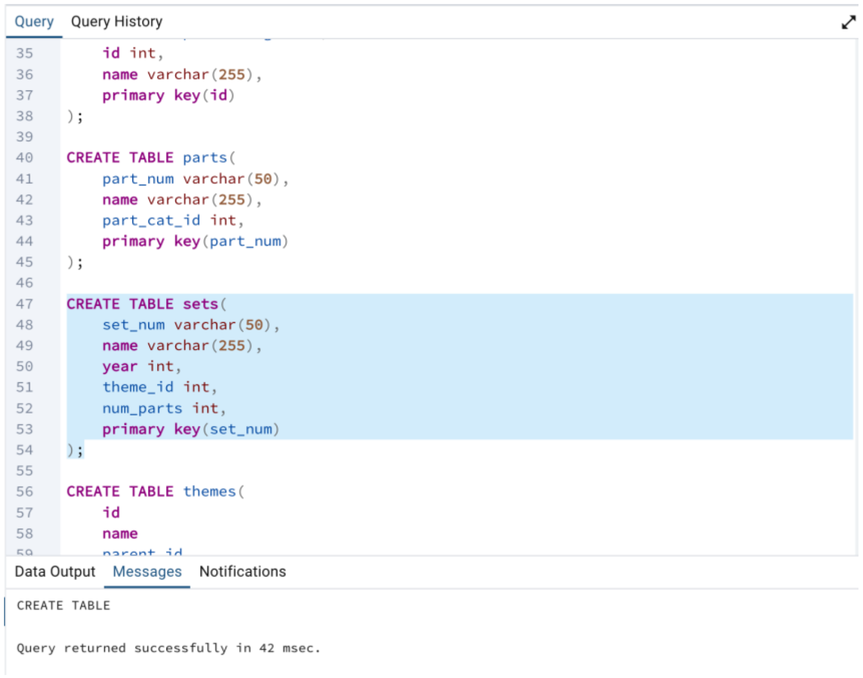

Query Query History

```
CREATE TABLE inventory_sets(
    inventory_id int,
    set_num varchar(50),
    quantity int,
    primary key(inventory_id, set_num)
);

CREATE TABLE part_categories(
    id int,
    name varchar(255),
    primary key(id)
);

CREATE TABLE parts(
    part_num varchar(50),
    name varchar(255),
    part_cat_id int,
    primary key(part_num)
);

CREATE TABLE sets(
    set_num
    name
    year
```

Data Output Messages Notifications

```
CREATE TABLE

Query returned successfully in 48 msec.
```

- sets
    - **set_num** is a string with different length, but all are less than 50. Also, it is unique, so it can be choosed as **primary key**
    - **name** is a string with different length, but all are less than 255
    - **year** is an integer
    - **theme_id** is an integer
    - **num_parts** is an integer

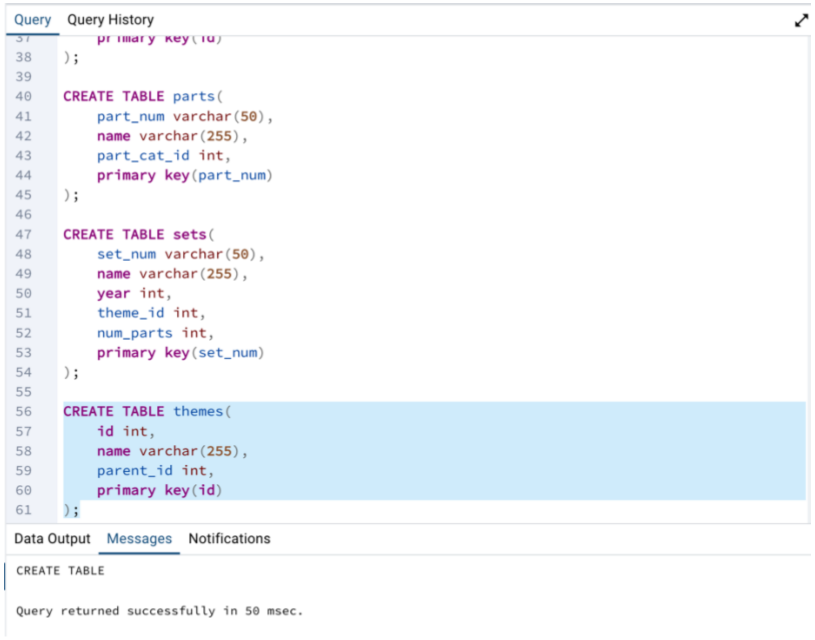

Query | Query History

```
    id int,
    name varchar(255),
    primary key(id)
);

CREATE TABLE parts(
    part_num varchar(50),
    name varchar(255),
    part_cat_id int,
    primary key(part_num)
);

CREATE TABLE sets(
    set_num varchar(50),
    name varchar(255),
    year int,
    theme_id int,
    num_parts int,
    primary key(set_num)
);

CREATE TABLE themes(
    id
    name
```

Data Output | Messages | Notifications

```
CREATE TABLE

Query returned successfully in 42 msec.
```

- themes
    - **id** is an integer and unique, so choose **id** as **primary key**
    - **name** is a string with different length, but all are less than 255
    - **parent_id** is an integer


Query | Query History

```
);

CREATE TABLE parts(
    part_num varchar(50),
    name varchar(255),
    part_cat_id int,
    primary key(part_num)
);

CREATE TABLE sets(
    set_num varchar(50),
    name varchar(255),
    year int,
    theme_id int,
    num_parts int,
    primary key(set_num)
);

CREATE TABLE themes(
    id int,
    name varchar(255),
    parent_id int,
    primary key(id)
);
```

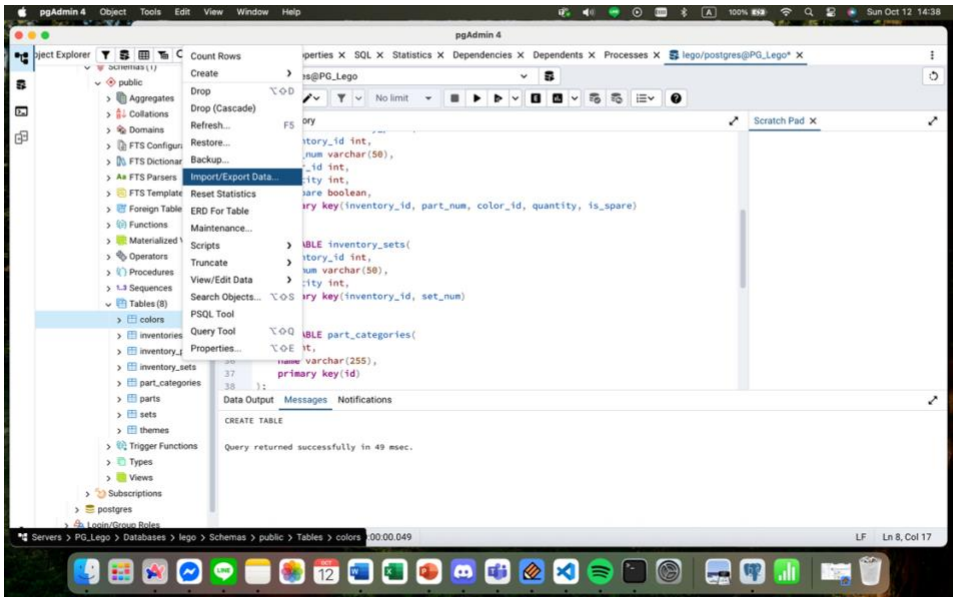
Data Output | Messages | Notifications

```
CREATE TABLE

Query returned successfully in 50 msec.
```

Step 3:

Import .csv files to each table, take colors.csv as example

Go to **`/PG_Lego/Database/lego/Schema/public/Table/colors`** → **Import/Export Data...**

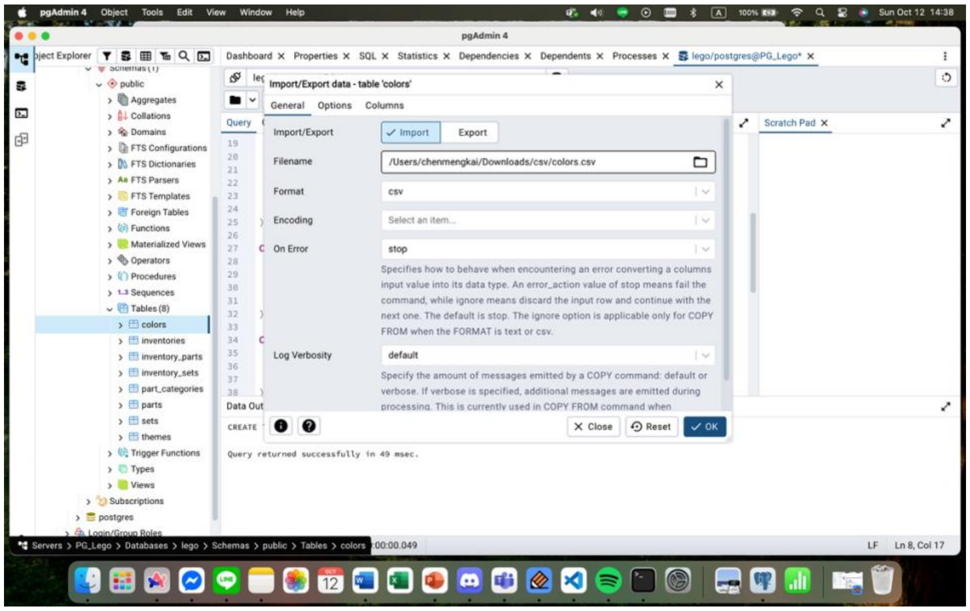


```
ntory_id int,
_num varchar(50),
_id int,
ity int,
pare boolean,
ary key(inventory_id, part_num, color_id, quantity, is_spare)

ABLE inventory_sets(
ntory_id int,
num varchar(50),
ity int,
ary key(inventory_id, set_num)

ABLE part_categories(
nt,
name varchar(255),
primary key(id)
);
```

```
CREATE TABLE

Query returned successfully in 49 msec.
```

Step 4:

Fill the path to the designated .csv file and swipe yes for header since it contains header in .csv file


pgAdmin 4
Import/Export data - table 'colors'
General
Options
Columns
Close
Reset
OK
Servers > PG_Lego > Databases > lego > Schemas > public > Tables > colors

Import/Export: [✓ Import] Export

Filename: /Users/chenmengkai/Downloads/csv/colors.csv

Format: csv

Encoding: Select an item...

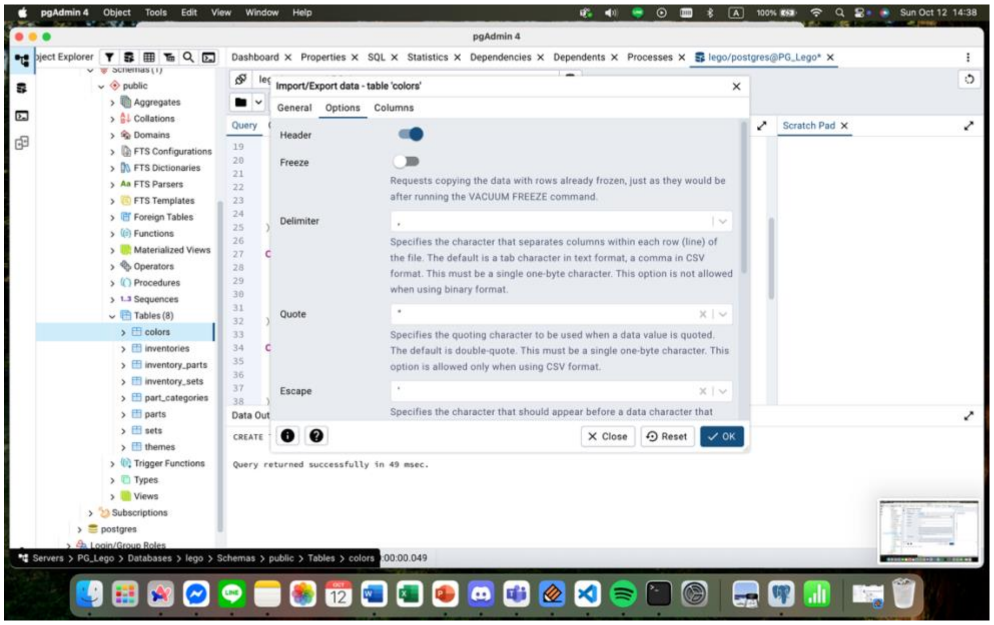
On Error: stop

Specifies how to behave when encountering an error converting a columns input value into its data type. An error_action value of stop means fail the command, while ignore means discard the input row and continue with the next one. The default is stop. The ignore option is applicable only for COPY FROM when the FORMAT is text or csv.

Log Verbosity: default

Specify the amount of messages emitted by a COPY command: default or verbose. If verbose is specified, additional messages are emitted during processing. This is currently used in COPY FROM command when

```
Query returned successfully in 49 msec.
```


Import/Export data - table 'colors'

General | Options | Columns

Header

Freeze

Requests copying the data with rows already frozen, just as they would be after running the VACUUM FREEZE command.

Delimiter: ,

Specifies the character that separates columns within each row (line) of the file. The default is a tab character in text format, a comma in CSV format. This must be a single one-byte character. This option is not allowed when using binary format.

Quote: "

Specifies the quoting character to be used when a data value is quoted. The default is double-quote. This must be a single one-byte character. This option is allowed only when using CSV format.

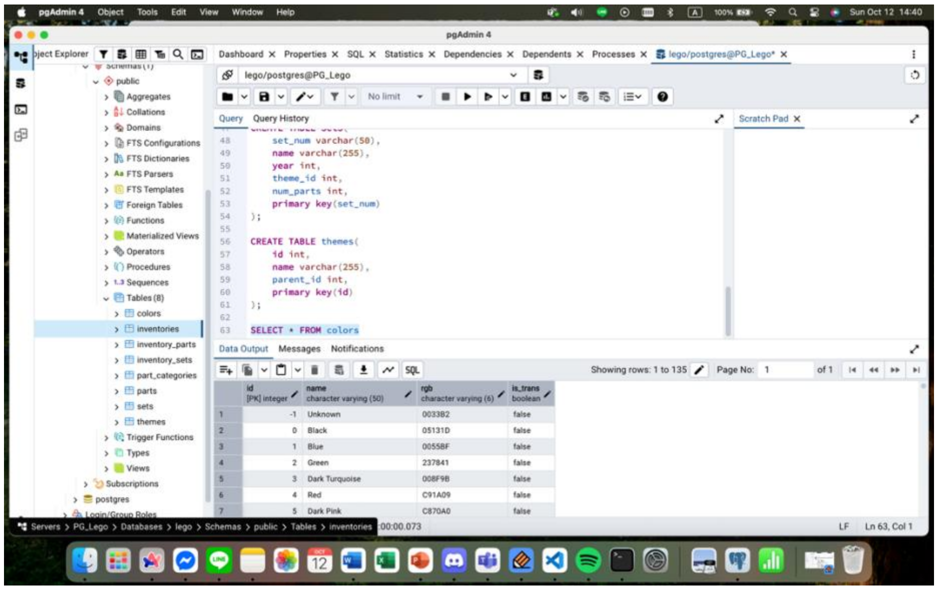
Escape: '

Specifies the character that should appear before a data character that

```
Query returned successfully in 49 msec.
```

Step 5:

Check if the .csv is really imported by **SELECT**



```
    set_num varchar(50),
    name varchar(255),
    year int,
    theme_id int,
    num_parts int,
    primary key(set_num)
);

CREATE TABLE themes(
    id int,
    name varchar(255),
    parent_id int,
    primary key(id)
);

SELECT * FROM colors
```

Showing rows: 1 to 135

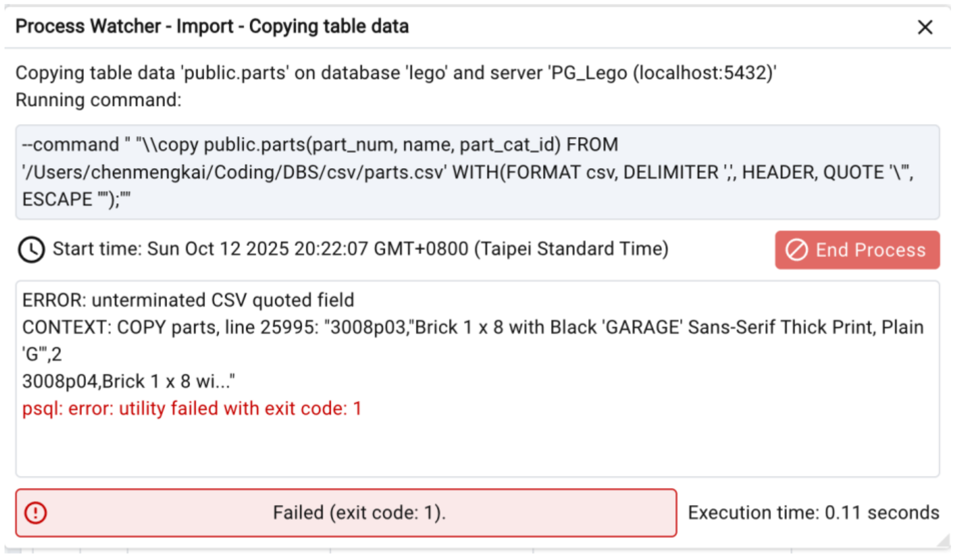
	id [PK] integer	name character varying (50)	rgb character varying (6)	is_trans boolean
1	-1	Unknown	0033B2	false
2	0	Black	05131D	false
3	1	Blue	0055BF	false
4	2	Green	237841	false
5	3	Dark Turquoise	008F9B	false
6	4	Red	C91A09	false
7	5	Dark Pink	C870A0	false

Step 6:

Repeat step.3 ~ step.5 for the other 6 .csv files besides parts.csv (explained below). Since they are the same processes, let's just skip it here

For parts.csvs:

While doing the same process on part.csv as above, an error would occur



Copying table data 'public.parts' on database 'lego' and server 'PG_Lego (localhost:5432)'
Running command:

```
--command " "\\copy public.parts(part_num, name, part_cat_id) FROM
'/Users/chenmengkai/Coding/DBS/csv/parts.csv' WITH(FORMAT csv, DELIMITER ',', HEADER, QUOTE '\'',
ESCAPE '"');""
```

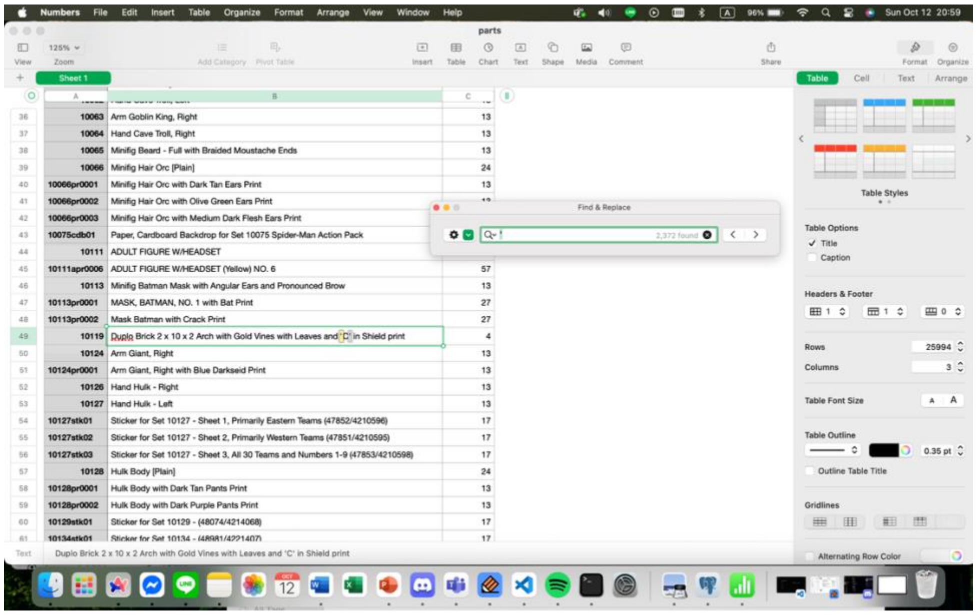
Start time: Sun Oct 12 2025 20:22:07 GMT+0800 (Taipei Standard Time)

```
ERROR: unterminated CSV quoted field
CONTEXT: COPY parts, line 25995: "3008p03,"Brick 1 x 8 with Black 'GARAGE' Sans-Serif Thick Print, Plain
'G'",2
3008p04,Brick 1 x 8 wi..."
psql: error: utility failed with exit code: 1
```

Failed (exit code: 1).

Execution time: 0.11 seconds

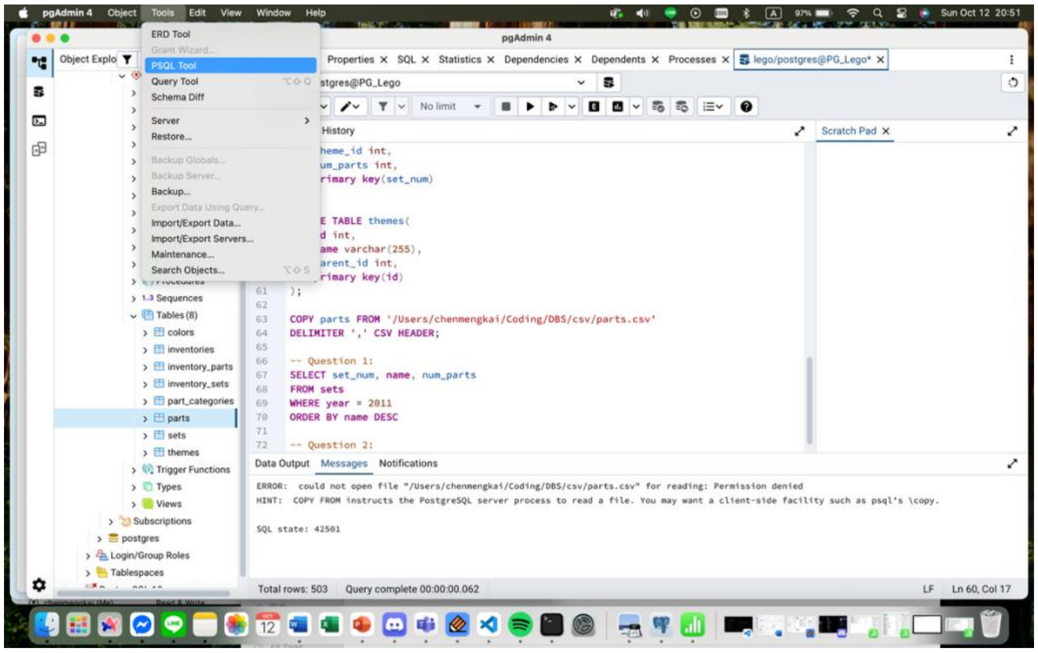
I guess it might because there are some names in the parts.csv contains apostraphe (')



	A	B	C
36	10063	Arm Goblin King, Right	13
37	10064	Hand Cave Troll, Right	13
38	10065	Minifig Beard - Full with Braided Moustache Ends	13
39	10066	Minifig Hair Orc [Plain]	24
40	10066pr0001	Minifig Hair Orc with Dark Tan Ears Print	13
41	10066pr0002	Minifig Hair Orc with Olive Green Ears Print	[illegible]
42	10066pr0003	Minifig Hair Orc with Medium Dark Flesh Ears Print	[illegible]
43	10075cdb01	Paper, Cardboard Backdrop for Set 10075 Spider-Man Action Pack	[illegible]
44	10111	ADULT FIGURE W/HEADSET	[illegible]
45	10111apr0006	ADULT FIGURE W/HEADSET (Yellow) NO. 6	57
46	10113	Minifig Batman Mask with Angular Ears and Pronounced Brow	13
47	10113pr0001	MASK, BATMAN, NO. 1 with Bat Print	27
48	10113pr0002	Mask Batman with Crack Print	27
49	10119	Duplo Brick 2 x 10 x 2 Arch with Gold Vines with Leaves and 'C' in Shield print	4
50	10124	Arm Giant, Right	13
51	10124pr0001	Arm Giant, Right with Blue Darkseid Print	13
52	10126	Hand Hulk - Right	13
53	10127	Hand Hulk - Left	13
54	10127stk01	Sticker for Set 10127 - Sheet 1, Primarily Eastern Teams (47852/4210596)	17
55	10127stk02	Sticker for Set 10127 - Sheet 2, Primarily Western Teams (47851/4210595)	17
56	10127stk03	Sticker for Set 10127 - Sheet 3, All 30 Teams and Numbers 1-9 (47853/4210598)	17
57	10128	Hulk Body [Plain]	24
58	10128pr0001	Hulk Body with Dark Tan Pants Print	13
59	10128pr0002	Hulk Body with Dark Purple Pants Print	13
60	10129stk01	Sticker for Set 10129 - (48074/4214068)	17
61	10134stk01	Sticker for Set 10134 - (48981/4221407)	17

Text: Duplo Brick 2 x 10 x 2 Arch with Gold Vines with Leaves and 'C' in Shield print

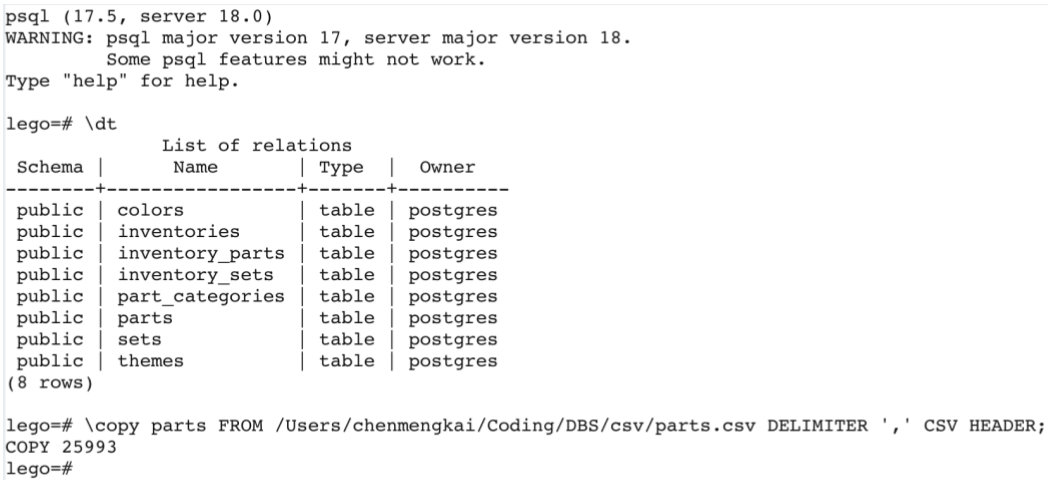
Solution: use the terminal in pgAdmin to copy the .csv file to the table in the database



```
heme_id int,
um_parts int,
rimary key(set_num)

E TABLE themes(
d int,
ame varchar(255),
arent_id int,
rimary key(id)
);

COPY parts FROM '/Users/chenmengkai/Coding/DBS/csv/parts.csv'
DELIMITER ',' CSV HEADER;

-- Question 1:
SELECT set_num, name, num_parts
FROM sets
WHERE year = 2011
ORDER BY name DESC

-- Question 2:
```

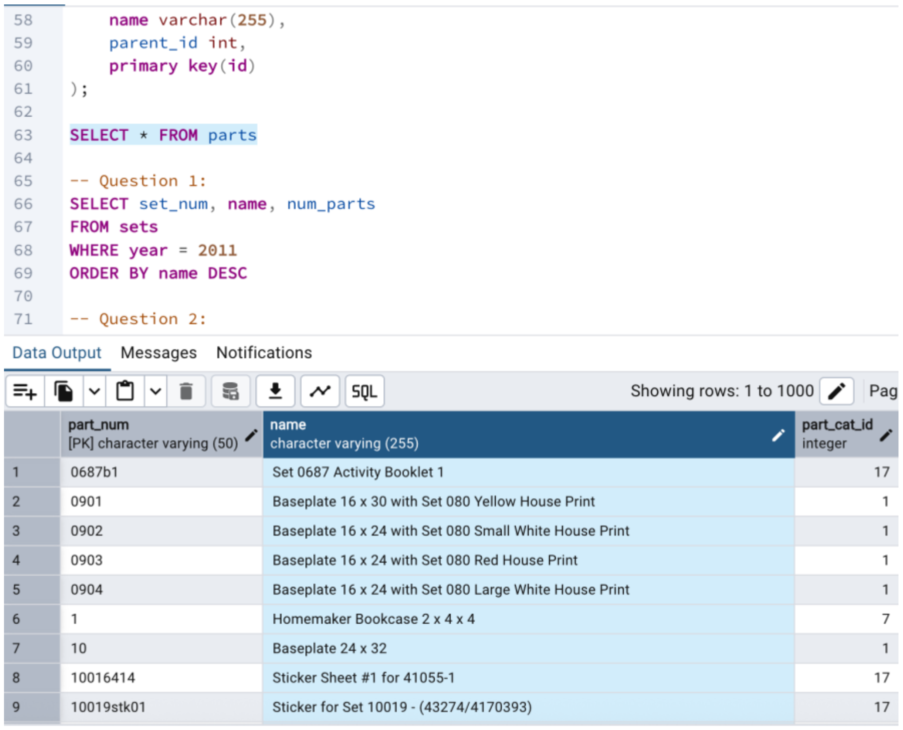
```
ERROR:  could not open file "/Users/chenmengkai/Coding/DBS/csv/parts.csv" for reading: Permission denied
HINT:  COPY FROM instructs the PostgreSQL server process to read a file. You may want a client-side facility such as psql's \copy.

SQL state: 42501
```

Total rows: 503 Query complete 00:00:00.062



```
psql (17.5, server 18.0)
WARNING: psql major version 17, server major version 18.
         Some psql features might not work.
Type "help" for help.

lego=# \dt
              List of relations
 Schema |       Name        | Type  |  Owner
--------+-------------------+-------+----------
 public | colors            | table | postgres
 public | inventories       | table | postgres
 public | inventory_parts   | table | postgres
 public | inventory_sets    | table | postgres
 public | part_categories   | table | postgres
 public | parts             | table | postgres
 public | sets              | table | postgres
 public | themes            | table | postgres
(8 rows)

lego=# \copy parts FROM /Users/chenmengkai/Coding/DBS/csv/parts.csv DELIMITER ',' CSV HEADER;
COPY 25993
lego=#
```

Check if the data is really imported



```
    name varchar(255),
    parent_id int,
    primary key(id)
);

SELECT * FROM parts

-- Question 1:
SELECT set_num, name, num_parts
FROM sets
WHERE year = 2011
ORDER BY name DESC

-- Question 2:
```

	part_num [PK] character varying (50)	name character varying (255)	part_cat_id integer
1	0687b1	Set 0687 Activity Booklet 1	17
2	0901	Baseplate 16 x 30 with Set 080 Yellow House Print	1
3	0902	Baseplate 16 x 24 with Set 080 Small White House Print	1
4	0903	Baseplate 16 x 24 with Set 080 Red House Print	1
5	0904	Baseplate 16 x 24 with Set 080 Large White House Print	1
6	1	Homemaker Bookcase 2 x 4 x 4	7
7	10	Baseplate 24 x 32	1
8	10016414	Sticker Sheet #1 for 41055-1	17
9	10019stk01	Sticker for Set 10019 - (43274/4170393)	17

Reference

1. pdAdmin Tutorial – How to Use pgAdmin
   https://www.youtube.com/watch?v=WFT5MaZN6g4&t=413s
2. Import CSV file into PostgreSQL with PgAdmin
   https://www.youtube.com/watch?v=WFT5MaZN6g4&t=413s