Database HW1

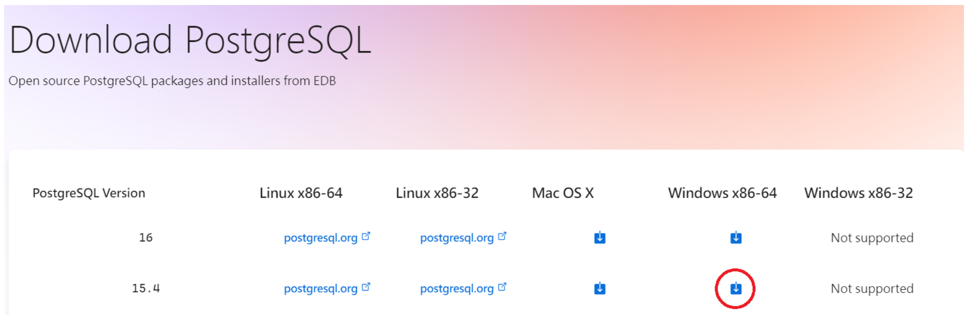
111550100 邱振源

🔗Link to this notion website, which is easier to read.

Q1: The process of creating the "lego" databases

Step 1

I visited the website of PostgreSQL, and downloaded the Windows x86-64 version of the interactive installer by EDB. Since I was concerned that the newest version will have some unexpected problem, I chose 15.4 version here.



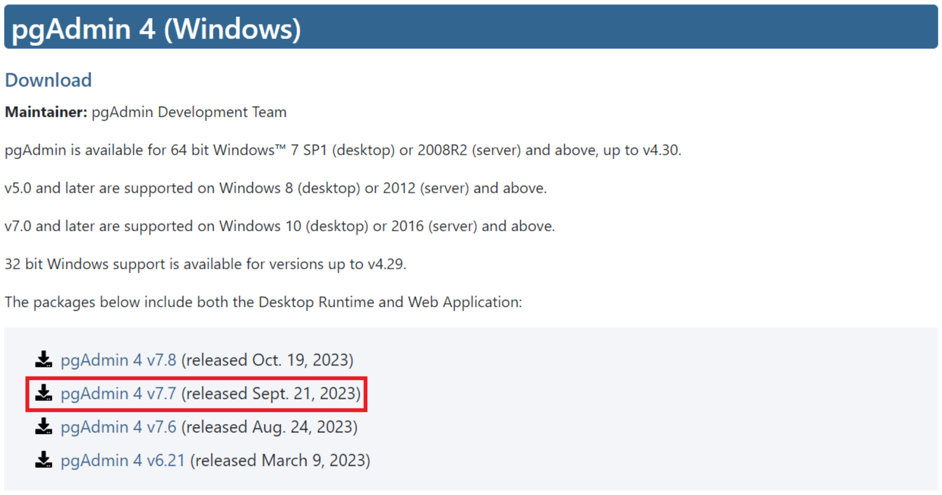
Download PostgreSQL

Open source PostgreSQL packages and installers from EDB

PostgreSQL Version	Linux x86-64	Linux x86-32	Mac OS X	Windows x86-64	Windows x86-32
16	postgresql.org	postgresql.org	⬇	⬇	Not supported
15.4	postgresql.org	postgresql.org	⬇	⬇	Not supported

Step 2

Since downloaded the pgAdmin with the above mentioned installer will cause the problem that I cannot successfully open it. As a result, I visited the official website of pgAdmin 4 to download the pgAdmin, which offers a more reliable and user-friendly interface.



pgAdmin 4 (Windows)

Download

Maintainer: pgAdmin Development Team

pgAdmin is available for 64 bit Windows™ 7 SP1 (desktop) or 2008R2 (server) and above, up to v4.30.

v5.0 and later are supported on Windows 8 (desktop) or 2012 (server) and above.

v7.0 and later are supported on Windows 10 (desktop) or 2016 (server) and above.

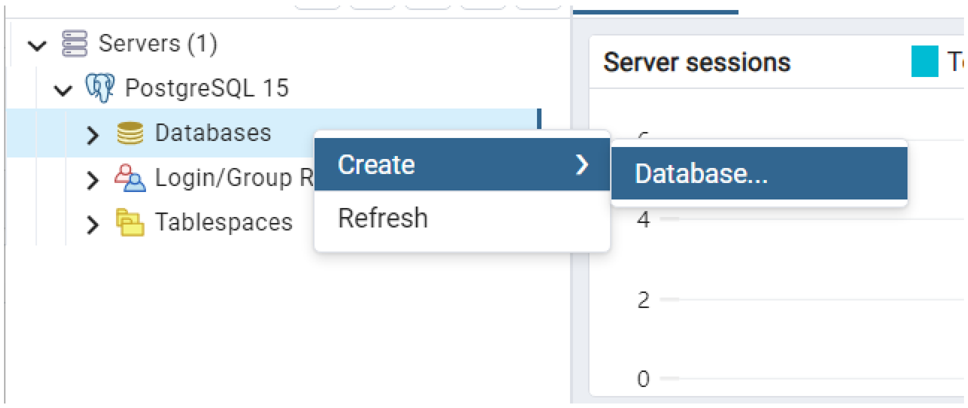
32 bit Windows support is available for versions up to v4.29.

The packages below include both the Desktop Runtime and Web Application:

- pgAdmin 4 v7.8 (released Oct. 19, 2023)
- pgAdmin 4 v7.7 (released Sept. 21, 2023)
- pgAdmin 4 v7.6 (released Aug. 24, 2023)
- pgAdmin 4 v6.21 (released March 9, 2023)

Step 3

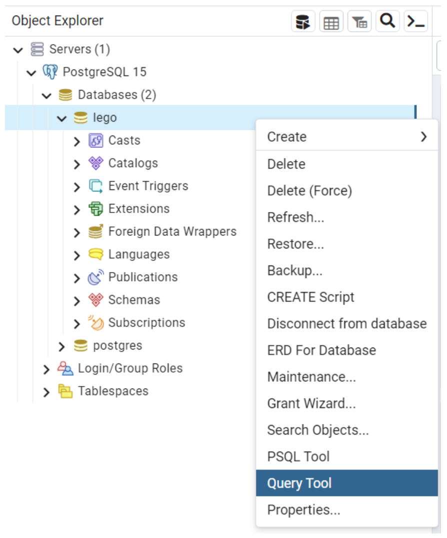
After downloading all the necessary applications, we can enter pgAdmin 4 and click the *"Server"* option on the left-hand side. By expanding the PostgreSQL 15 icon and clicking the *"Database"* button, we can proceed to *"Create"* and then *"Database"*. After naming the database and configuring its attributes, we successfully created the *"lego"* database.



Q2: The process of importing eight required .csv files into lego database.

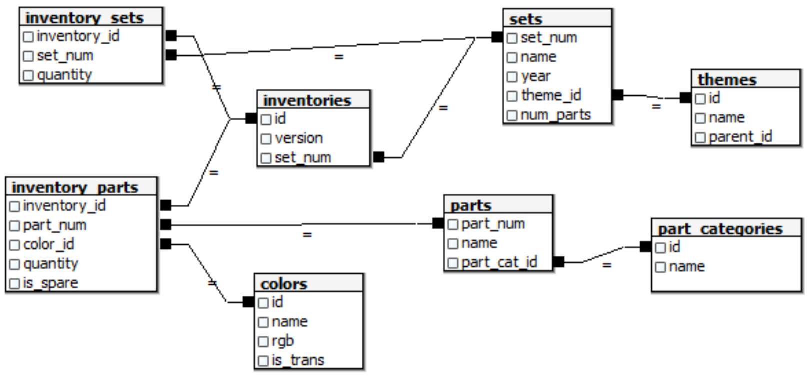
Step 1

Press the right click on the icon of *"lego"*, then click the *"Query Tool"*. By doing this, I could type the DDL on the user interface.



Step 2

However, to import the .csv files into lego database, generating table of each file is needed. So, I read the schema of each file and wrote the table query.



***“colors”* table**

> Choose “id” as primary key since it’s unique.

```
create table public.colors
(
```

***“part_categories”* table**

> Choose “id” as primary key since it’s unique.

```
create table public.part_categories
(
    id varchar(20),
    name varchar(100),
    rgb char(6),
    is_trans boolean,
    primary key (id)
);
```

```
    id varchar(20),
    name varchar(100),
    primary key (id)
);
```

<u>*“theme”* table</u>

> Choose “id” as primary key since it’s unique.

```
create table themes
(
    id varchar(20),
    name varchar(100),
    parent_id varchar(15),
    primary key (id)
);
```

<u>*“parts”* table</u>

> Choose “part_num” as primary key since it’s unique.
>
> Choose “part_cat_id” as foreign key obtained from “part_categories”.

```
create table public.parts
(
    part_num varchar(20),
    name varchar(300),
    part_cat_id varchar(15),
    primary key (part_num),
    foreign key (part_cat_id)
      references part_categories(id)
);
```

<u>*“sets”* table</u>

> Choose “set_num” as primary key since it’s unique.
>
> Choose “theme_id” as foreign key obtained from “themes”.

```
create table sets
(
    set_num varchar(20),
    name varchar(100),
```

<u>*“inventories”* table</u>

> Choose “id” as primary key since it’s unique.
>
> choose “set_num” as foreign key obtained from “sets”.

```
create table public.inventories
(
    id varchar(20),
    version int,
    year int,
    theme_id varchar(15),
    num_parts int,
    primary key (set_num),
    foreign key (theme_id)
        references themes(id)
);
```

```
    set_num varchar(20),
    primary key (id),
    foreign key (set_num)
          references sets(set_num)
);
```

***“inventory_parts”* table**

> No primary key since there is no unique attribute in “inventory_parts.csv”.
>
> Choose “inventory_id” as foreign key obtained from “inventories”.
>
> Choose “color_id” as foreign key obtained from “colors”.

```
create table public.inventory_parts
(
    inventory_id varchar(15),
    part_num varchar(20),
    color_id varchar(15),
    quantity int,
    is_spare boolean,
    /*no primary key*/
    foreign key (inventory_id) references inventories(id),
    foreign key (color_id) references colors(id)
);
```

***“inventory_sets”* table**

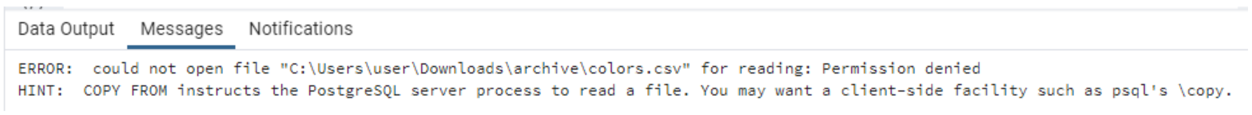
> Choose “inventory_id” and “set_num” as primary key since it’s unique.
>
> Choose “inventory_id” as foreign key obtained from “inventories”.
>
> Choose “set_num” as foreign key obtained from “sets”.

```
create table public.inventory_sets
(
    inventory_id varchar(15),
    set_num varchar(20),
    quantity int,
    primary key (inventory_id),
    primary key (set_num),
    foreign key (inventory_id) references inventories(id),
    foreign key (set_num) references sets(set_num)
);
```

📄Link to all the query

Step 3

After completing the table creation process, let's proceed to the file importation step. Download the file from the website and extract the contents from the 'archive' .zip file. Initially, I stored the files in *C:\Users\user\Downloads\archive,* but I encountered the error message shown below:

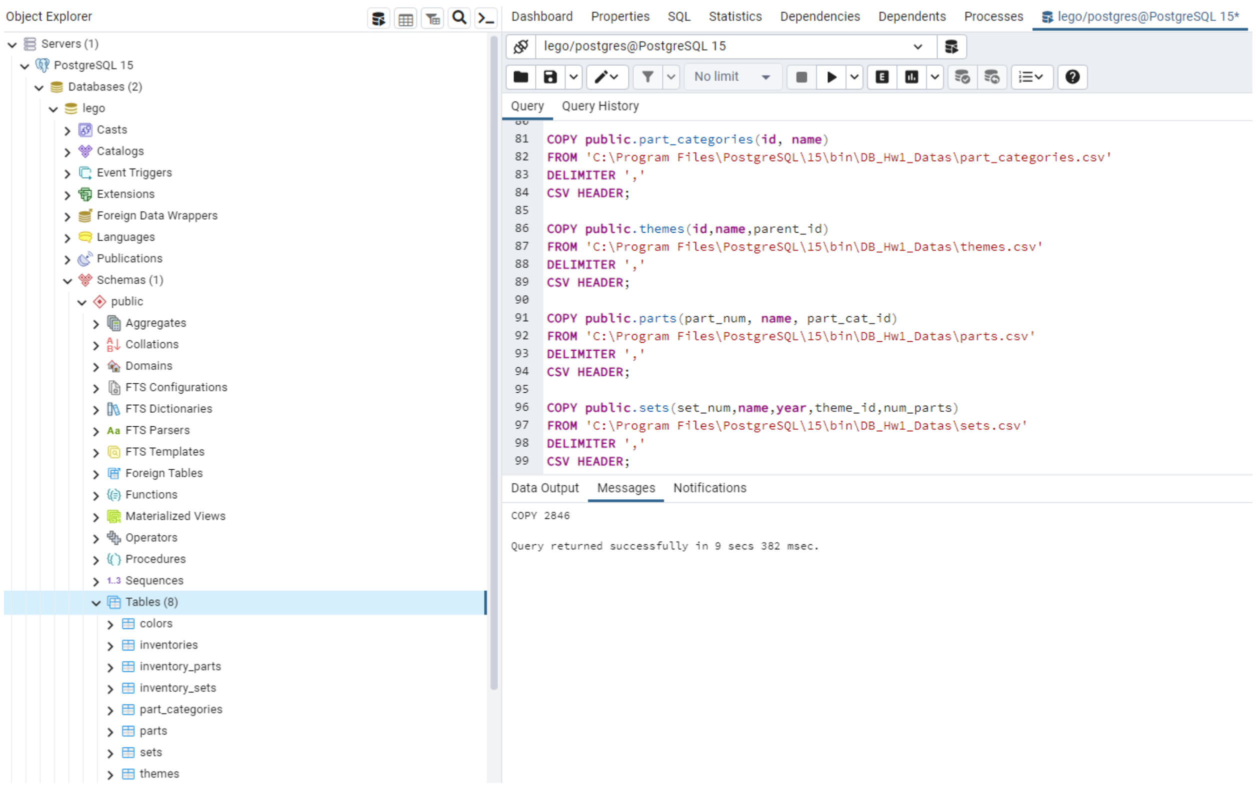

```
ERROR:  could not open file "C:\Users\user\Downloads\archive\colors.csv" for reading: Permission denied
HINT:  COPY FROM instructs the PostgreSQL server process to read a file. You may want a client-side facility such as psql's \copy.
```

So, I changed the location of the files to *C:\Program Files\PostgreSQL\15\bin\DB_Hw1_Datas* since PostgreSQL could directly access the files here without needing additional permissions. After that, I used the 'COPY' keyword to import the files.

```
COPY public.colors(id, name, rgb, is_trans)
FROM 'C:\Program Files\PostgreSQL\15\bin\DB_Hw1_Datas\colors.csv'
DELIMITER ','
CSV HEADER;
```

The code above specifies the columns (id, name, rgb, and is_trans), the file path, and the delimiter (",") to be used with the files. Additionally, it assumes that the first row of the .csv file contains column headers (HEADER).

To import other files, I only needed to modify the table name, the attributes (columns), and the file name in the path. The results are displayed below:



```
COPY public.part_categories(id, name)
FROM 'C:\Program Files\PostgreSQL\15\bin\DB_Hw1_Datas\part_categories.csv'
DELIMITER ','
CSV HEADER;

COPY public.themes(id,name,parent_id)
FROM 'C:\Program Files\PostgreSQL\15\bin\DB_Hw1_Datas\themes.csv'
DELIMITER ','
CSV HEADER;

COPY public.parts(part_num, name, part_cat_id)
FROM 'C:\Program Files\PostgreSQL\15\bin\DB_Hw1_Datas\parts.csv'
DELIMITER ','
CSV HEADER;

COPY public.sets(set_num,name,year,theme_id,num_parts)
FROM 'C:\Program Files\PostgreSQL\15\bin\DB_Hw1_Datas\sets.csv'
DELIMITER ','
CSV HEADER;
```

```
COPY 2846

Query returned successfully in 9 secs 382 msec.
```

📄Link to all the query

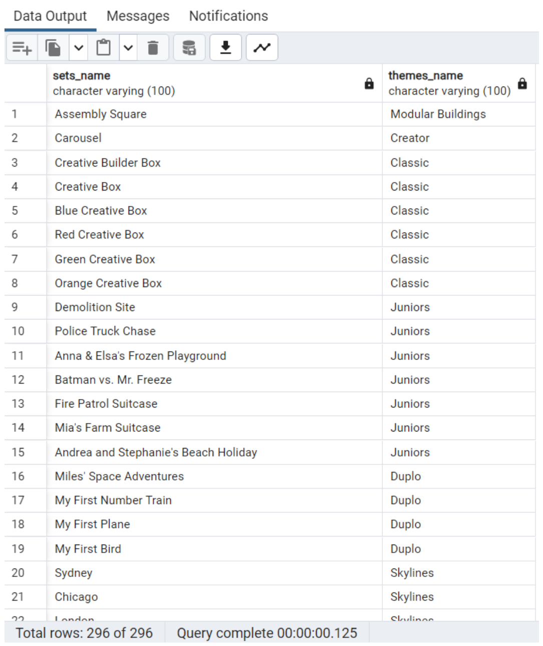
Q3: The SQL statements and output results of 4a.

- ### SQL statements

```
SELECT
    sets.name as sets_name,
    themes.name as themes_name
FROM
    sets,
    themes
WHERE
    themes.id = sets.theme_id AND
    sets.year = 2017
```

- ### Output results

> A part of table (total 296 rows)



Data Output Messages Notifications

	sets_name character varying (100)	themes_name character varying (100)
1	Assembly Square	Modular Buildings
2	Carousel	Creator
3	Creative Builder Box	Classic
4	Creative Box	Classic
5	Blue Creative Box	Classic
6	Red Creative Box	Classic
7	Green Creative Box	Classic
8	Orange Creative Box	Classic
9	Demolition Site	Juniors
10	Police Truck Chase	Juniors
11	Anna & Elsa's Frozen Playground	Juniors
12	Batman vs. Mr. Freeze	Juniors
13	Fire Patrol Suitcase	Juniors
14	Mia's Farm Suitcase	Juniors
15	Andrea and Stephanie's Beach Holiday	Juniors
16	Miles' Space Adventures	Duplo
17	My First Number Train	Duplo
18	My First Plane	Duplo
19	My First Bird	Duplo
20	Sydney	Skylines
21	Chicago	Skylines
22	London	Skylines

Total rows: 296 of 296 Query complete 00:00:00.125

📄Link to show all the output result

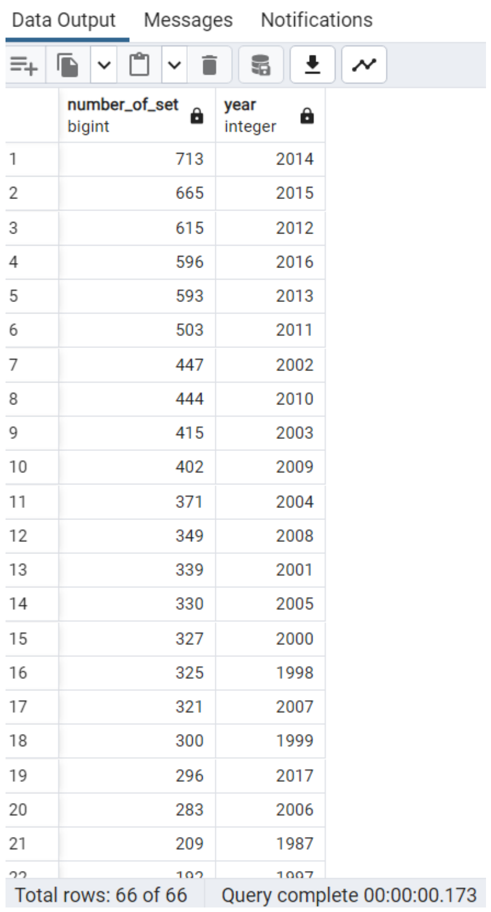
Q4: The SQL statements and output results of 4b.

- ### SQL statements

```
SELECT
    count (set_num)
            as number_of_set,
    year
FROM
    sets
WHERE
    year <= 2017 AND
    year >= 1950
GROUP BY
    year
ORDER BY
    number_of_set desc
```

- ### Output results

> A part of table (total 66 rows)


Data Output Messages Notifications

	number_of_set bigint	year integer
1	713	2014
2	665	2015
3	615	2012
4	596	2016
5	593	2013
6	503	2011
7	447	2002
8	444	2010
9	415	2003
10	402	2009
11	371	2004
12	349	2008
13	339	2001
14	330	2005
15	327	2000
16	325	1998
17	321	2007
18	300	1999
19	296	2017
20	283	2006
21	209	1987
22	192	1997

Total rows: 66 of 66 Query complete 00:00:00.173

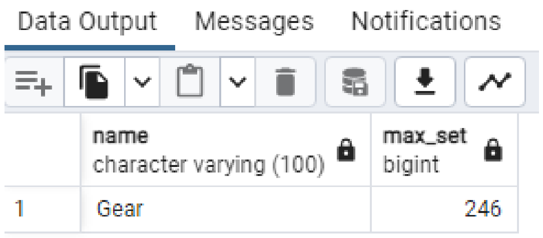
📄Link to show all the output result

Q5: The SQL statements and output results of 4c.

- **SQL statements**

```
WITH
    themes_count(name, num_set)as(
      SELECT
        themes.name,
        count(sets.name)
      FROM
        sets,
        themes
      WHERE
        themes.id = sets.theme_id
      GROUP BY
        themes.id)
SELECT
    name,
```

- **Output results**


Data Output Messages Notifications

	name character varying (100)	max_set bigint
1	Gear	246

```
    num_set as max_set
FROM
    themes_count
WHERE
    num_set = (
      SELECT
        MAX(num_set)
      FROM
        themes_count
    );
```

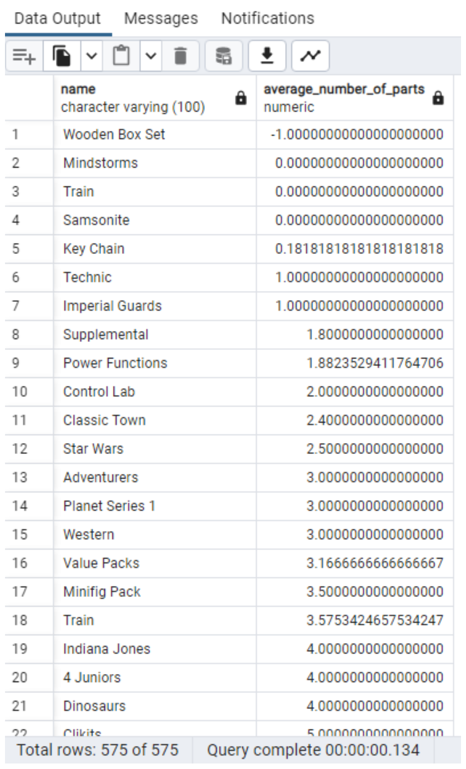
Q6: The SQL statements and output results of 4d.

- **SQL statements**

```
WITH
    themes_avg(name, parts) as(
      SELECT
        themes.name, avg(sets.num_parts)
      FROM
        sets, themes
      WHERE
        themes.id = sets.theme_id
      GROUP BY
        themes.id)
SELECT
    name,
    parts as average_number_of_parts
FROM
    themes_avg
ORDER BY
    average_number_of_parts asc
```

- **Output results**

> A part of table
>
> (total 575 rows)



	name character varying (100)	average_number_of_parts numeric
1	Wooden Box Set	-1.00000000000000000000
2	Mindstorms	0.00000000000000000000
3	Train	0.00000000000000000000
4	Samsonite	0.00000000000000000000
5	Key Chain	0.18181818181818181818
6	Technic	1.00000000000000000000
7	Imperial Guards	1.00000000000000000000
8	Supplemental	1.8000000000000000
9	Power Functions	1.8823529411764706
10	Control Lab	2.0000000000000000
11	Classic Town	2.4000000000000000
12	Star Wars	2.5000000000000000
13	Adventurers	3.0000000000000000
14	Planet Series 1	3.0000000000000000
15	Western	3.0000000000000000
16	Value Packs	3.1666666666666667
17	Minifig Pack	3.5000000000000000
18	Train	3.5753424657534247
19	Indiana Jones	4.0000000000000000
20	4 Juniors	4.0000000000000000
21	Dinosaurs	4.0000000000000000
22	Clikits	5.0000000000000000

Total rows: 575 of 575 Query complete 00:00:00.134

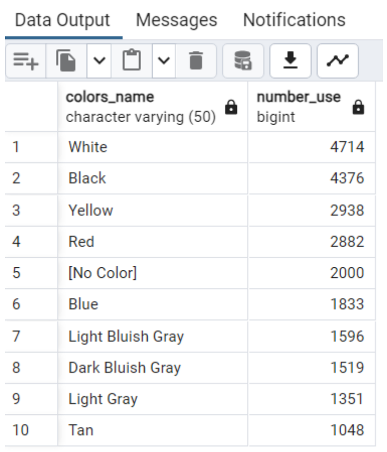
📄Link to show all the output result

Q7: The SQL statements and output results of 4e.

- **SQL statements**

```
WITH
  color_use(name, number_use) as(
  SELECT
    colors.name,
    count(distinct inventory_parts.part_num)
  FROM
    inventory_parts,
    colors
  WHERE
    colors.id = inventory_parts.color_id
  GROUP BY
    colors.id)
SELECT
    colors.name as colors_name,
    color_use.number_use
FROM
    colors,
    color_use
WHERE
  colors.name = color_use.name
ORDER BY
    number_use desc
LIMIT 10;
```

- **Output results**



	colors_name character varying (50)	number_use bigint
1	White	4714
2	Black	4376
3	Yellow	2938
4	Red	2882
5	[No Color]	2000
6	Blue	1833
7	Light Bluish Gray	1596
8	Dark Bluish Gray	1519
9	Light Gray	1351
10	Tan	1048

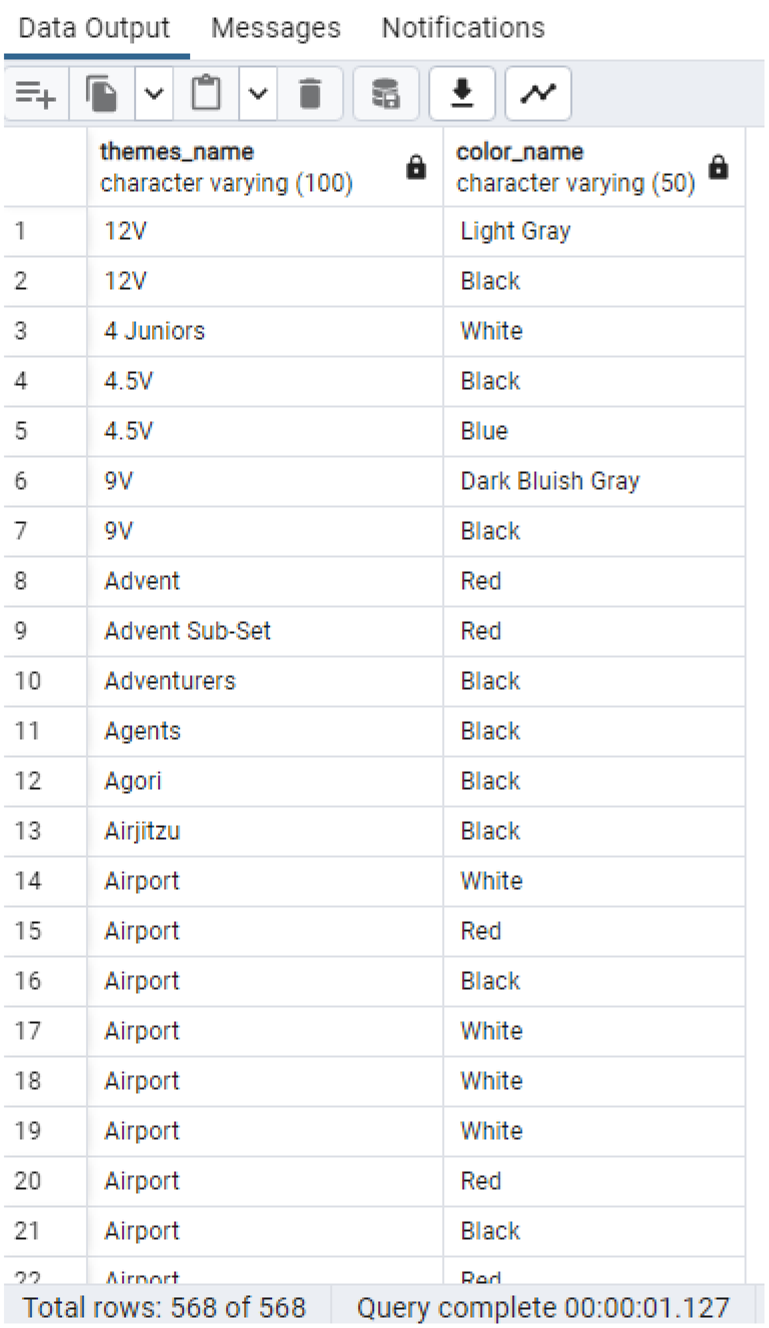
Q8: The SQL statements and output results of 4f.

- **SQL statements**

```
WITH
  quantity(color_name, inventory_id, quantity_sum) as(
  SELECT
    colors.name,
    inventory_parts.inventory_id,
    sum (inventory_parts.quantity)
  FROM
    inventory_parts join colors on colors.id = inventory_parts.color_id
  GROUP BY
    colors.name, inventory_parts.inventory_id, inventory_parts.part_num
  ),
  total_quantity(themes_name, color_name, total_quantity, rank) as(
  SELECT
    themes.name,
    quantity.color_name,
    sum(quantity.quantity_sum),
    rank() over (partition by themes.id
           order by sum(quantity.quantity_sum) desc) as rank
  FROM
    themes join sets on themes.id = sets.theme_id
    join inventories on sets.set_num = inventories.set_num
    join quantity on inventories.id = quantity.inventory_id
  GROUP BY
    themes.id,
    quantity.color_name
)
SELECT
  themes_name,
  color_name
FROM
  total_quantity
WHERE
  rank = 1
ORDER BY
  themes_name
```

- **Output results**

> A part of table (total 568 rows)



	themes_name character varying (100)	color_name character varying (50)
1	12V	Light Gray
2	12V	Black
3	4 Juniors	White
4	4.5V	Black
5	4.5V	Blue
6	9V	Dark Bluish Gray
7	9V	Black
8	Advent	Red
9	Advent Sub-Set	Red
10	Adventurers	Black
11	Agents	Black
12	Agori	Black
13	Airjitzu	Black
14	Airport	White
15	Airport	Red
16	Airport	Black
17	Airport	White
18	Airport	White
19	Airport	White
20	Airport	Red
21	Airport	Black
22	Airport	Red

Total rows: 568 of 568 Query complete 00:00:01.127

📄Link to show all the output result

🔗Link to show all the SQL query and the output result