Data Base HW1

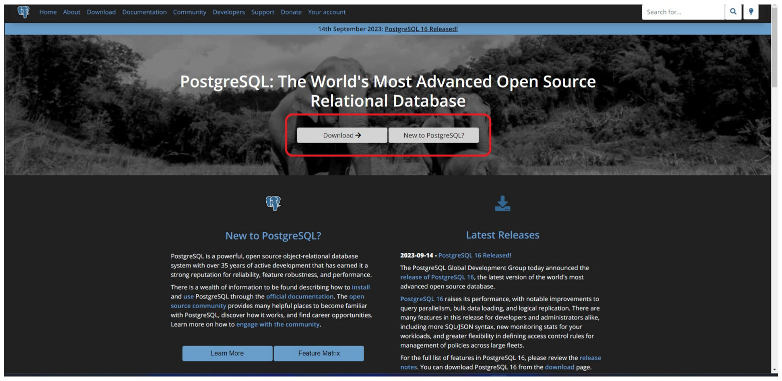
🔗: *Link to Github*

Q1. The process of creating the "lego" databases

Step 1

First of all, before creating a data base system, we should build the system first.

Go to https://www.postgresql.org/



Home About Download Documentation Community Developers Support Donate Your account
Search for...

14th September 2023: PostgreSQL 16 Released!

PostgreSQL: The World's Most Advanced Open Source Relational Database

Download →
New to PostgreSQL?

New to PostgreSQL?

PostgreSQL is a powerful, open source object-relational database system with over 35 years of active development that has earned it a strong reputation for reliability, feature robustness, and performance.

There is a wealth of information to be found describing how to install and use PostgreSQL through the official documentation. The open source community provides many helpful places to become familiar with PostgreSQL, discover how it works, and find career opportunities. Learn more on how to engage with the community.

Latest Releases

2023-09-14 - PostgreSQL 16 Released!

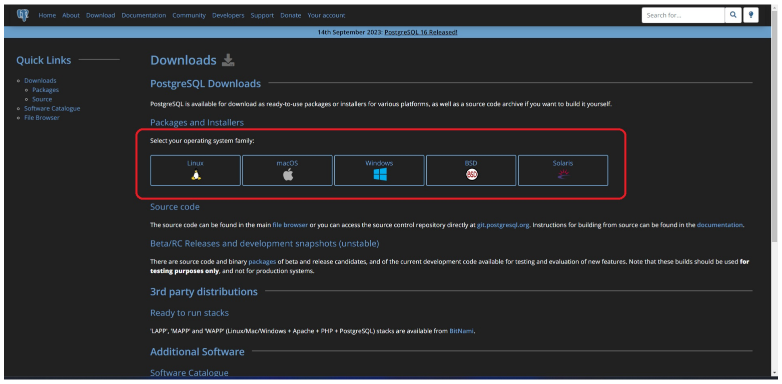
The PostgreSQL Global Development Group today announced the release of PostgreSQL 16, the latest version of the world's most advanced open source database.

PostgreSQL 16 raises its performance, with notable improvements to query parallelism, bulk data loading, and logical replication. There are many features in this release for developers and administrators alike, including more SQL/JSON syntax, new monitoring stats for your workloads, and greater flexibility in defining access control rules for management of policies across large fleets.

For the full list of features in PostgreSQL 16, please review the release notes. You can download PostgreSQL 16 from the download page.

and click the Download.

Subsequently, select the operating system platform that corresponds to your usage. In my case, I utilize the Windows environment; therefore, please proceed by clicking on the ***Windows icon***.



Home About Download Documentation Community Developers Support Donate Your account
Search for...

14th September 2023: PostgreSQL 16 Released!

Downloads

PostgreSQL Downloads

PostgreSQL is available for download as ready-to-use packages or installers for various platforms, as well as a source code archive if you want to build it yourself.

Packages and Installers

Select your operating system family:

Source code

The source code can be found in the main file browser or you can access the source control repository directly at git.postgresql.org. Instructions for building from source can be found in the documentation.

Beta/RC Releases and development snapshots (unstable)

There are source code and binary packages of beta and release candidates, and of the current development code available for testing and evaluation of new features. Note that these builds should be used **for testing purposes only**, and not for production systems.

3rd party distributions

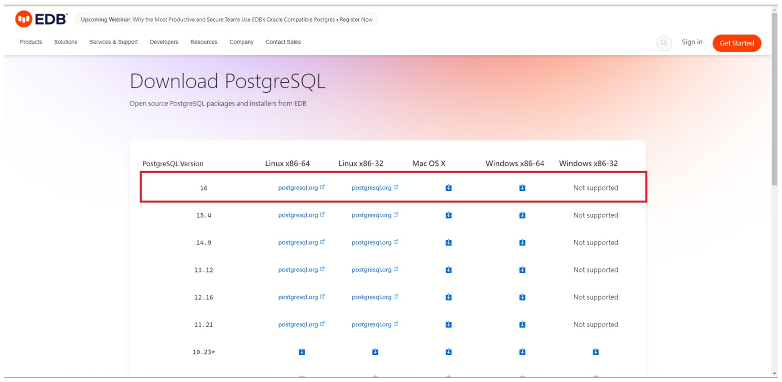
Ready to run stacks

'LAPP', 'MAPP' and 'WAPP' (Linux/Mac/Windows + Apache + PHP + PostgreSQL) stacks are available from BitNami.

Additional Software

Software Catalogue

Here, I have opted for the ***version 16***, as it potentially features certain specialized functionalities. While my current understanding of its utilization remains limited, I am open to the prospect of acquiring further knowledge later.


EDB
Upcoming Webinar: Why the Most Productive and Secure Teams Use EDB's Oracle Compatible Postgres • Register Now
Products
Solutions
Services & Support
Developers
Resources
Company
Contact Sales
Sign in
Get Started

Download PostgreSQL

Open source PostgreSQL packages and installers from EDB

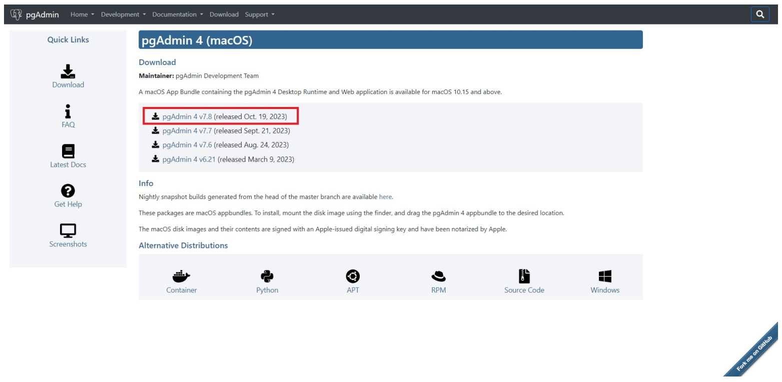
PostgreSQL Version	Linux x86-64	Linux x86-32	Mac OS X	Windows x86-64	Windows x86-32
16	postgresql.org	postgresql.org	⬇	⬇	Not supported
15.4	postgresql.org	postgresql.org	⬇	⬇	Not supported
14.9	postgresql.org	postgresql.org	⬇	⬇	Not supported
13.12	postgresql.org	postgresql.org	⬇	⬇	Not supported
12.16	postgresql.org	postgresql.org	⬇	⬇	Not supported
11.21	postgresql.org	postgresql.org	⬇	⬇	Not supported
10.23*	⬇	⬇	⬇	⬇	⬇

Step 2

After downloading the installer,I face with some problem, I could not open it. So i decide to downloading all the PostgreSQL 15.4 tools except for ***pgAdmin***(the original one), including ***SQL Shell(psql)*** and ***Application Stack Builder.***

I came to the official website of pgAdmin 4 and download it.

Here i still choose the latest verison.



pgAdmin 4 (macOS)

Download

Maintainer: pgAdmin Development Team

A macOS App Bundle containing the pgAdmin 4 Desktop Runtime and Web application is available for macOS 10.15 and above.

- 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)

Info

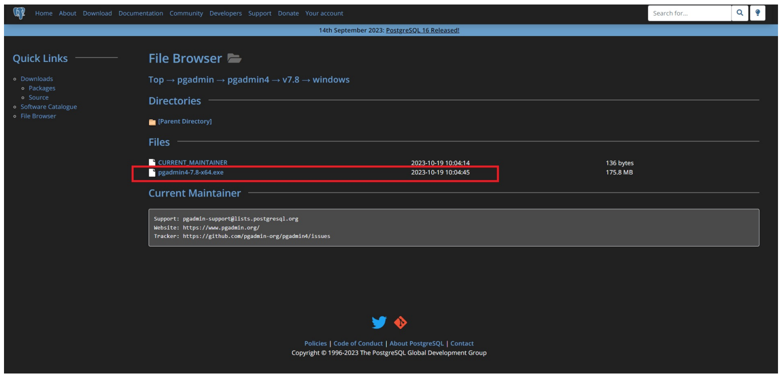
Nightly snapshot builds generated from the head of the master branch are available here.

These packages are macOS appbundles. To install, mount the disk image using the finder, and drag the pgAdmin 4 appbundle to the desired location.

The macOS disk images and their contents are signed with an Apple-issued digital signing key and have been notarized by Apple.

Alternative Distributions

Certainly, initiate the process by obtaining the pgadmin4-7.8-x64.exe file and proceed with its execution.



14th September 2023: PostgreSQL 16 Released!

File Browser

Top → pgadmin → pgadmin4 → v7.8 → windows

Directories

[Parent Directory]

Files

CURRENT_MAINTAINER	2023-10-19 10:04:14	136 bytes
pgadmin4-7.8-x64.exe	2023-10-19 10:04:45	175.8 MB

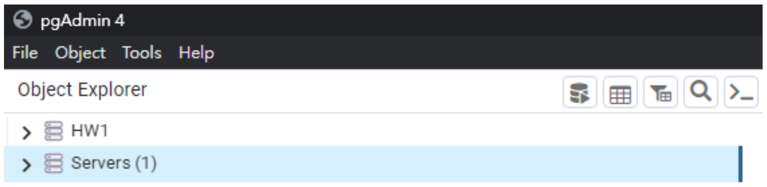
Current Maintainer

```
Support: pgadmin-support@lists.postgresql.org
Website: https://www.pgadmin.org/
Tracker: https://github.com/pgadmin-org/pgadmin4/issues
```

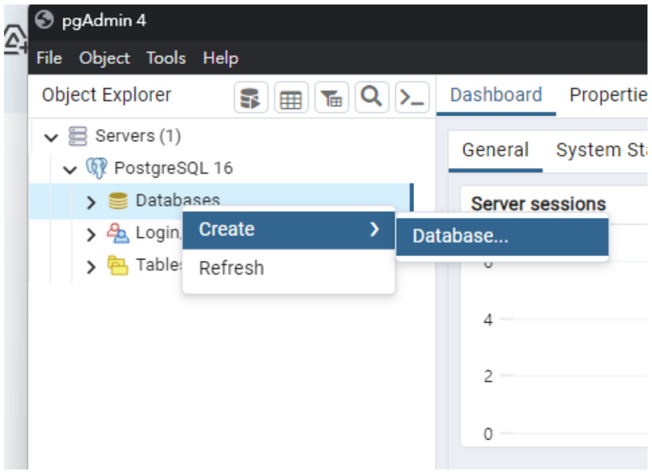
Copyright © 1996-2023 The PostgreSQL Global Development Group

Step 3

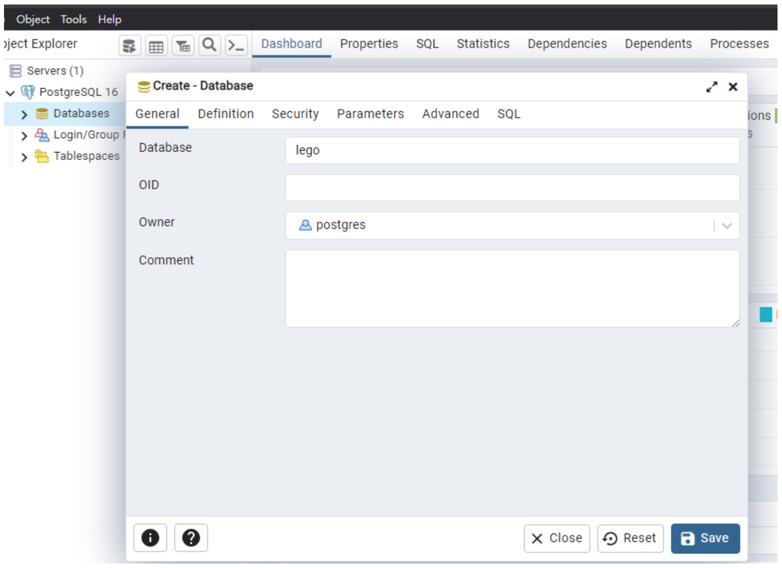
After downloading the ***pgAdmin 4*** and ***PostgreSQL 15.4***. Now entered the app(*pgAdmin 4*) and we could see the *“Server”* icon one left hand side. Click it and it will show *PostgreSQL 15* icon and *“Database”* button down below.



Right clicked the Database button and click *"Create→Database"* and then a window came up. Here you could type the name of this database and some other attributes.



According to the instruction TA provided, the designated database name is '***lego.***' Therefore, kindly input '*lego*' in the empty field situated to the right of the term '*Database*.'

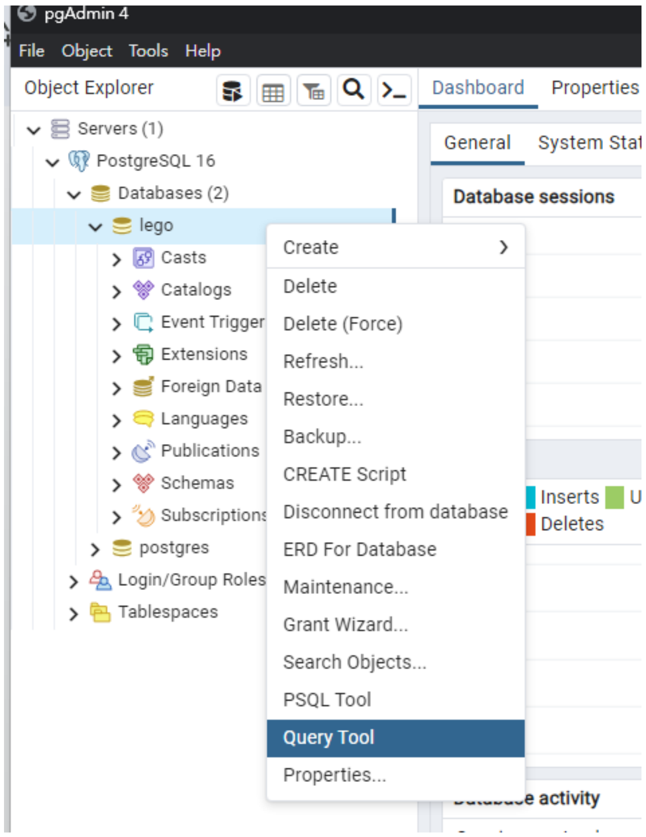

Object Tools Help
Object Explorer
Dashboard Properties SQL Statistics Dependencies Dependents Processes
Servers (1)
PostgreSQL 16
Databases
Login/Group
Tablespaces
Create - Database
General Definition Security Parameters Advanced SQL
Close
Reset
Save

Database	lego
OID	
Owner	postgres
Comment	

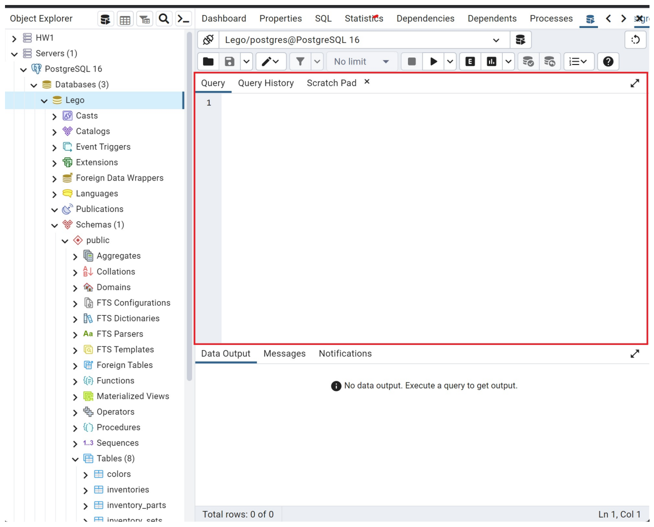
After setting all down, I clicked the save button and the "LEGO" database was created.

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

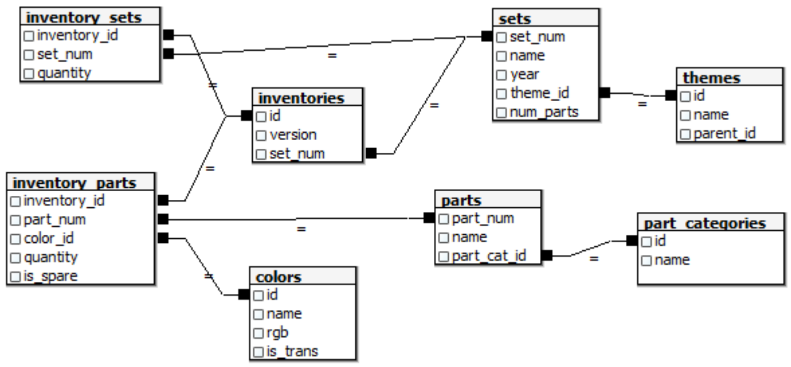
After "***LEGO***" database was created, now it is time to define the ***DDL(Data Definition Language).*** First of all,you can see the "lego" icon, the database we created before, on left hand side, right click it and you will find the keyword —- "Querry Tool" below. Click it and we can start to write DDL.



To import all the *.csv* files, I used the query tool in app's(pdAdmin) UI, the place that we can type the sql lanquage. Usually, most operation will occur here. It is shown below. And we could then type in the schema of database datas using *DDL.*



Before importing datas from 8 csv files, we need to create the tables for each file first. I read and post the relation between each table from the LEGO website. Below is the picture that demonstrate the relation of tables.



Step1. The creating table query for each .csv file :

Create the “*colors*” table

```
CREATE TABLE public.colors(
    id VARCHAR(15),
    name VARCHAR(50),
    rgb CHAR(6),
    is_trans BOOLEAN,

    primary key (id)
);
```

By the relation table, we can know that “id” is the primary key of colors table.

Create the “*themes*” table

```
CREATE TABLE themes(
    id VARCHAR(15),
    name VARCHAR(100),
    parent_id VARCHAR(15),

    primary key (id)
);
```

Create the “*sets*” table

```
CREATE TABLE sets(
  set_num VARCHAR(20),
    name VARCHAR(100),
    year INT,
    theme_id VARCHAR(15),
    num_parts INT,

    primary key (set_num),
    foreign key (theme_id) references themes(id)
);
```

Create the “*inventories*” table

```
CREATE TABLE public.inventories(
  id VARCHAR(15),
  version INT,
  set_num VARCHAR(20),

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

Create the “*inventory_sets*” table

```
CREATE TABLE public.inventory_sets(
  inventory_id VARCHAR(15),
    set_num VARCHAR(20),
    quantity INT,

    primary key (inventory_id, set_num),
    foreign key (inventory_id) references inventories(id),
    foreign key (set_num) references sets(set_num)
);
```

Create the “*colors*” table

```
CREATE TABLE public.part_categories(
    id VARCHAR(15),
    name VARCHAR(100),

    primary key (id)
);
```

Create the “*colors*” table

```
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)
);
```

Create the “*colors*” table

```
CREATE TABLE public.inventory_parts(
    inventory_id VARCHAR(15),
    part_num VARCHAR(20),
    color_id VARCHAR(15),
    quantity INT,
    is_spare BOOLEAN,

    foreign key (inventory_id) references inventories(id),
    foreign key (color_id) references colors(id)
);
```

Here is the *link* to my GitHub respository, which store all of the sql query that create the table.

Step2. Importing eight required .csv files into lego database

After creating the table, we could then start to insert file into these tables.The file download from net is a zip file called “archive”, decompose it and then you will get the .csv file in that.Then creat a copy in the bin of this app. For me, my file is stored in “*C:/Program Files/PostgreSQL/16/bin/DB_HW1_Datas*”. At the very begining, i storied directly in the desktop, but the pgAdmin can not find those files. So I then change the address of files to bin.

Then go back to the UI of pgAdmin, using the concept and keyword of “COPY ”.

Here is one of the example :

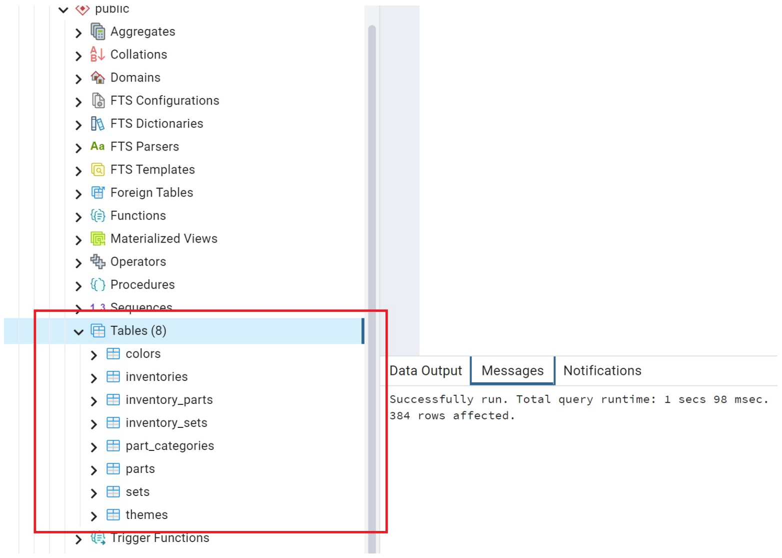
```
COPY public.colors(id,name,rgb,is_trans)
FROM 'C:/Program Files/PostgreSQL/16/bin/DB_HW1_Datas/colors.csv'
DELIMITER ','
CSV HEADER;
```

The SQL statement `COPY public.colors(id, name, rgb, is_trans) FROM 'C:/Program Files/PostgreSQL/16/bin/DB_HW1_Datas/colors.csv' DELIMITER ',' CSV HEADER;` is used to import data from a CSV file into the `public.colors` table in a PostgreSQL database. It specifies the columns (`id` , `name` , `rgb` , and `is_trans`) into which the data will be inserted, the source file path (`'C:/Program Files/PostgreSQL/16/bin/DB_HW1_Datas/colors.csv'`), and that the file uses a comma (`,`) as the delimiter between values. Additionally, it assumes that the first row of the CSV file contains column headers (`HEADER`), which PostgreSQL will use to match with the target table's columns. This command is commonly used for bulk data loading.

Oter tables with some common process , the change is of course the data type and keys and the file name are different. So to prevent duplicated steps, it put other in my GitHub. Here is my code:

🔗*(Link to all the query)*

Below you can check the result:



```
Successfully run. Total query runtime: 1 secs 98 msec.
384 rows affected.
```

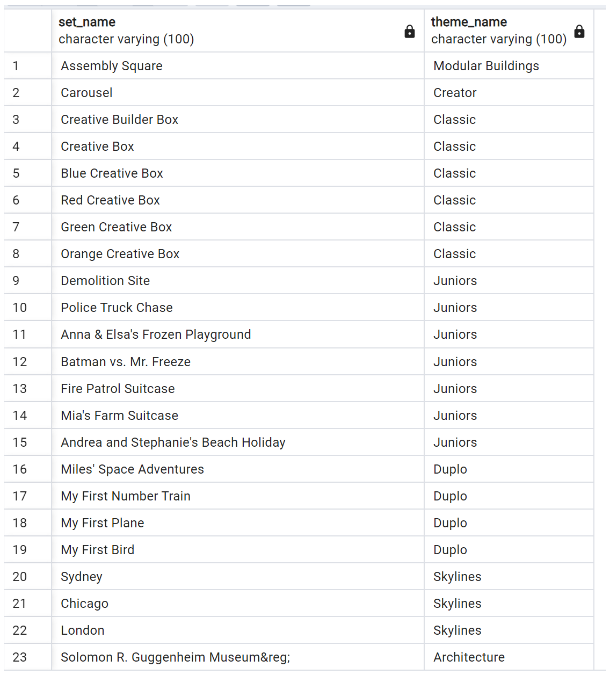
Q3. The SQL statements and output results of 4a

- The SQL

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

- The output result

total number of rows : 296



	set_name character varying (100)	theme_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
23	Solomon R. Guggenheim Museum&reg;	Architecture

🔗*: Link to Github*

Q3. The SQL statements and output results of 4b

- The SQL

```
SELECT
  count(sets.set_num) as total_set_num, sets.year
FROM
  sets
WHERE
  sets.year <= 2017 AND
  sets.year >= 1950
GROUP BY
  sets.year
ORDER BY
  total_set_nu DESC
```

- The output result

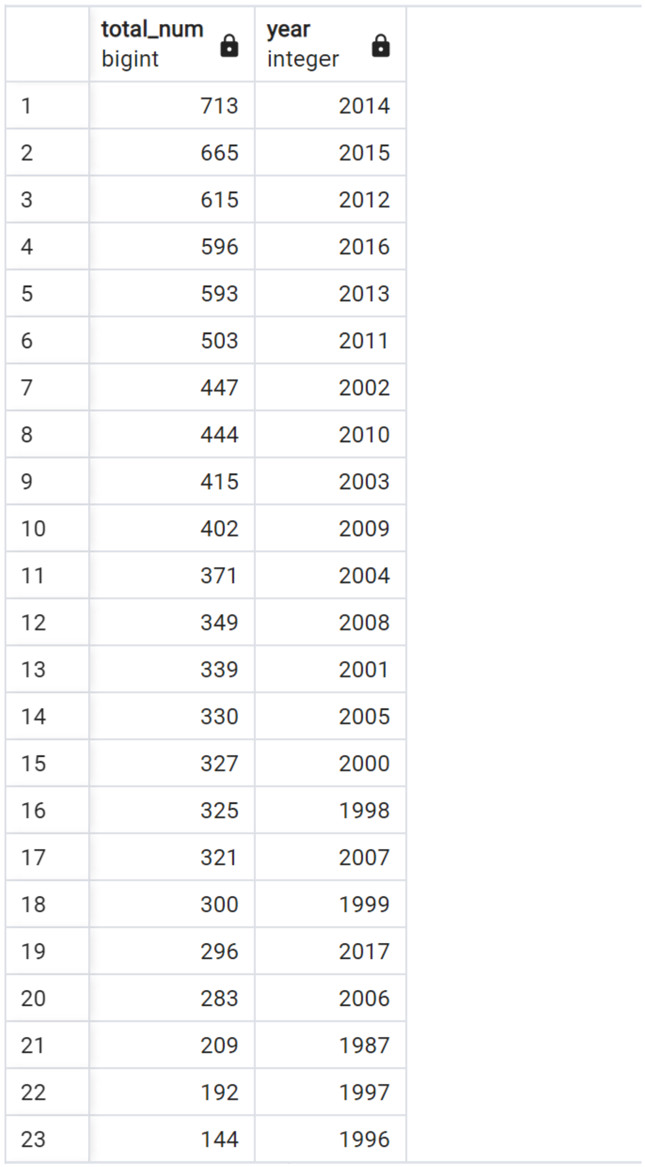
total number of rows : 66

- The SQL

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

- The output result

total number of rows : 296



	total_num 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
23	144	1996

: Link to Github

Q4. The SQL statements and output results of 4c

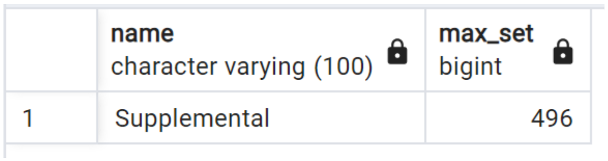
- The SQL statements

```
WITH
  themes_sets (name,num_set)
AS (
  select themes.name, count(sets.name)
  from themes,sets
  where themes.id = sets.theme_id
  group by themes.name
)

SELECT
   name,num_set as max_set
FROM
  themes_sets
WHERE
```

- The output result

total number of rows : 1



	name character varying (100)	max_set bigint
1	Supplemental	496

```
  num_set = (select max(num_set)
         from themes_sets);
```

Q5. The SQL statements and output results of 4d

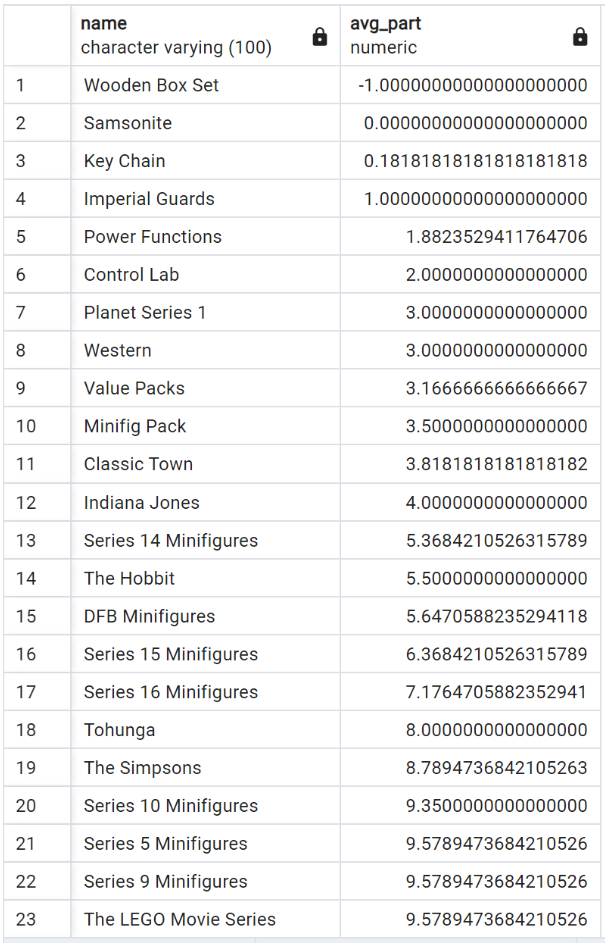
- The SQL statements

```
WITH
  themes_avg (name,part)
AS (
  select themes.name, avg(sets.num_parts)
  from themes,sets
  where themes.id = sets.theme_id
  group by themes.name
)

SELECT
   name,part as avg_part
FROM
  themes_avg
ORDER BY
  part ASC
```

- The output result

total number of rows : 386



	name character varying (100)	avg_part numeric
1	Wooden Box Set	-1.00000000000000000000
2	Samsonite	0.00000000000000000000
3	Key Chain	0.18181818181818181818
4	Imperial Guards	1.00000000000000000000
5	Power Functions	1.8823529411764706
6	Control Lab	2.0000000000000000
7	Planet Series 1	3.0000000000000000
8	Western	3.0000000000000000
9	Value Packs	3.1666666666666667
10	Minifig Pack	3.5000000000000000
11	Classic Town	3.8181818181818182
12	Indiana Jones	4.0000000000000000
13	Series 14 Minifigures	5.3684210526315789
14	The Hobbit	5.5000000000000000
15	DFB Minifigures	5.6470588235294118
16	Series 15 Minifigures	6.3684210526315789
17	Series 16 Minifigures	7.1764705882352941
18	Tohunga	8.0000000000000000
19	The Simpsons	8.7894736842105263
20	Series 10 Minifigures	9.3500000000000000
21	Series 5 Minifigures	9.5789473684210526
22	Series 9 Minifigures	9.5789473684210526
23	The LEGO Movie Series	9.5789473684210526

🔗 *: Link to Github*

Q6. The SQL statements and output results of 4e

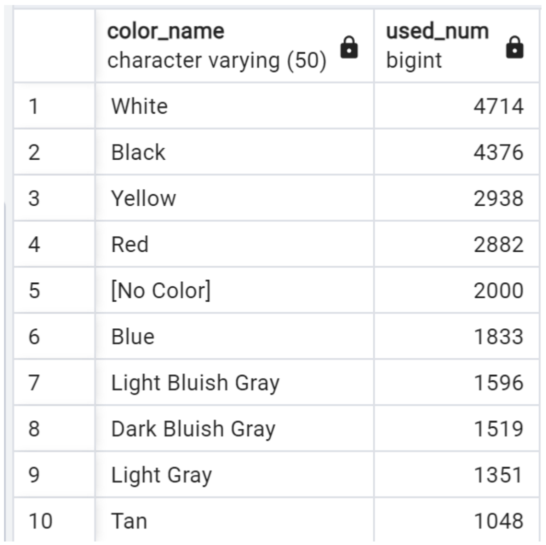
- The SQL statements

```
WITH color_used (name, num)
AS (
    SELECT colors.name, COUNT(DISTINCT inventory_parts.part_num) AS num
    FROM colors, inventory_parts
    WHERE colors.id = inventory_parts.color_id
    GROUP BY colors.name
```

- The output result

total number of rows : 10

```
)
SELECT
    name AS color_name, num AS used_num
FROM
    color_used
ORDER BY
    color_used.num DESC
LIMIT 10;
```



	color_name character varying (50)	used_num 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

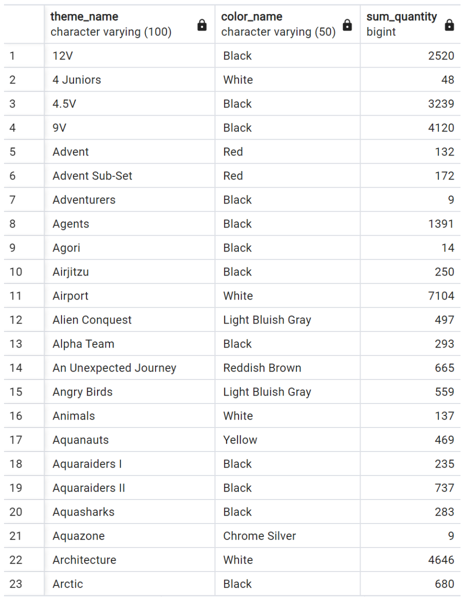
Q6. The SQL statements and output results of 4f

- The SQL statements

```
WITH color_theme_quantity AS (
    SELECT
        themes.name AS theme_name,
        colors.name AS color_name,
        quantity
    FROM
        colors, inventory_parts, inventories, sets, themes, parts
    WHERE
        parts.part_num = inventory_parts.part_num AND
        colors.id = inventory_parts.color_id AND
        inventory_parts.inventory_id = inventories.id AND
        sets.set_num = inventories.set_num AND
        sets.theme_id = themes.id
)
, color_theme_max_quantity AS (
    SELECT
        theme_name,
        color_name,
        SUM(quantity) AS sum_quantity
    FROM
        color_theme_quantity
    GROUP BY
        theme_name, color_name
)
, RankedColors AS (
    SELECT
        theme_name,
        color_name,
        sum_quantity,
        RANK() OVER (PARTITION BY theme_name ORDER BY sum_quantity DESC) AS rank
    FROM
        color_theme_max_quantity
)
SELECT
    theme_name,
    color_name,
    sum_quantity
FROM
    RankedColors
WHERE
    rank = 1;
```

- The output result

total number of rows : 384



	theme_name character varying (100)	color_name character varying (50)	sum_quantity bigint
1	12V	Black	2520
2	4 Juniors	White	48
3	4.5V	Black	3239
4	9V	Black	4120
5	Advent	Red	132
6	Advent Sub-Set	Red	172
7	Adventurers	Black	9
8	Agents	Black	1391
9	Agori	Black	14
10	Airjitzu	Black	250
11	Airport	White	7104
12	Alien Conquest	Light Bluish Gray	497
13	Alpha Team	Black	293
14	An Unexpected Journey	Reddish Brown	665
15	Angry Birds	Light Bluish Gray	559
16	Animals	White	137
17	Aquanauts	Yellow	469
18	Aquaraiders I	Black	235
19	Aquaraiders II	Black	737
20	Aquasharks	Black	283
21	Aquazone	Chrome Silver	9
22	Architecture	White	4646
23	Arctic	Black	680

🔗*: Link to Github*

🔗: Link to Github(the all SQL statement)