Database_HW1_111550076

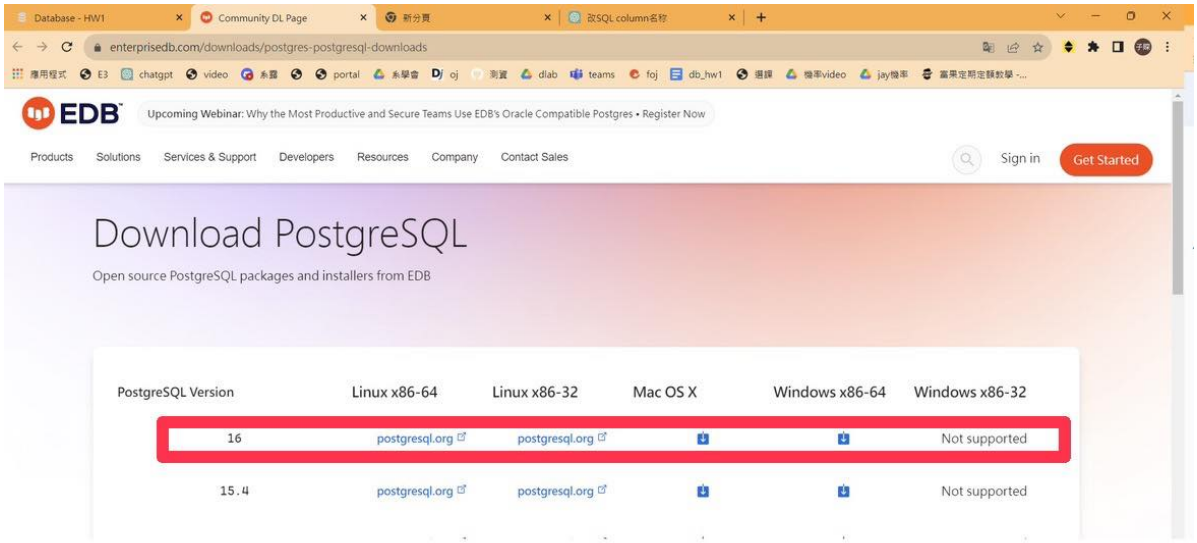
Link to my query code and result on github:

https://github.com/David810209/intro-to-database2023/tree/main/HW1

Q1. The process of creating the “lego” databases.

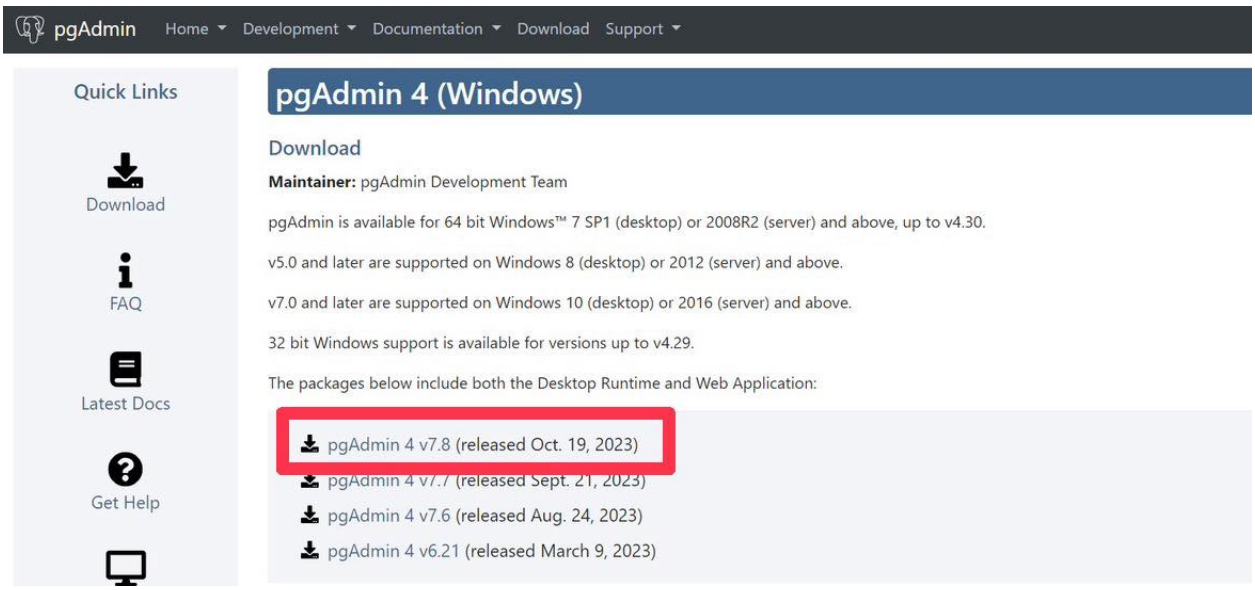
- **STEP 1.**

  Visited the PostgreSQL website to download the latest version of the package (Windows x86-64, version 16).



- **STEP 2.**

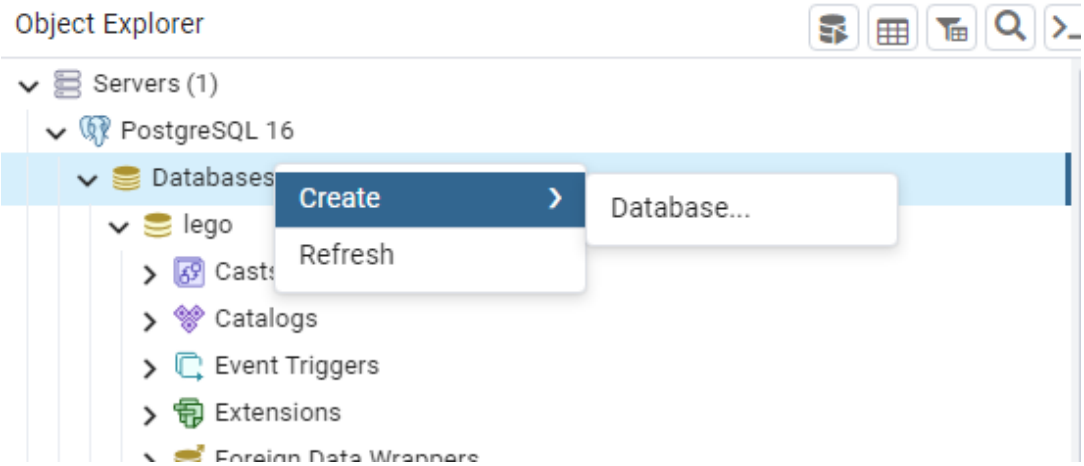
  Visited the pgAdmin website to download pgAdmin 4 version 7.8 in order to make it easier to work with PostgreSQL.



- **STEP 3.**

  Open pgAdmin → go to “Server” → right-click on “Database” → select “Create” → choose “Database” → Enter the name “lego” → click “Save” →

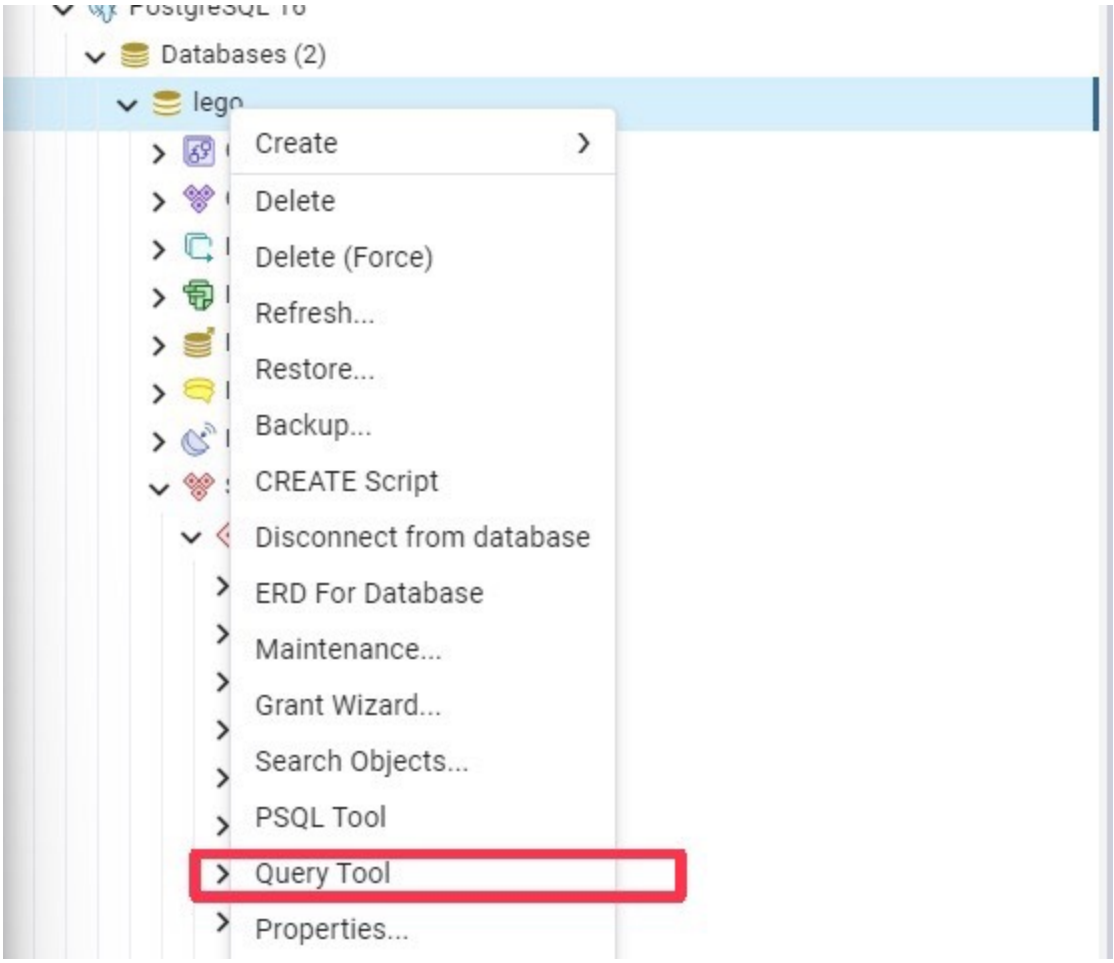
  I have successfully created the “lego” Database.



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

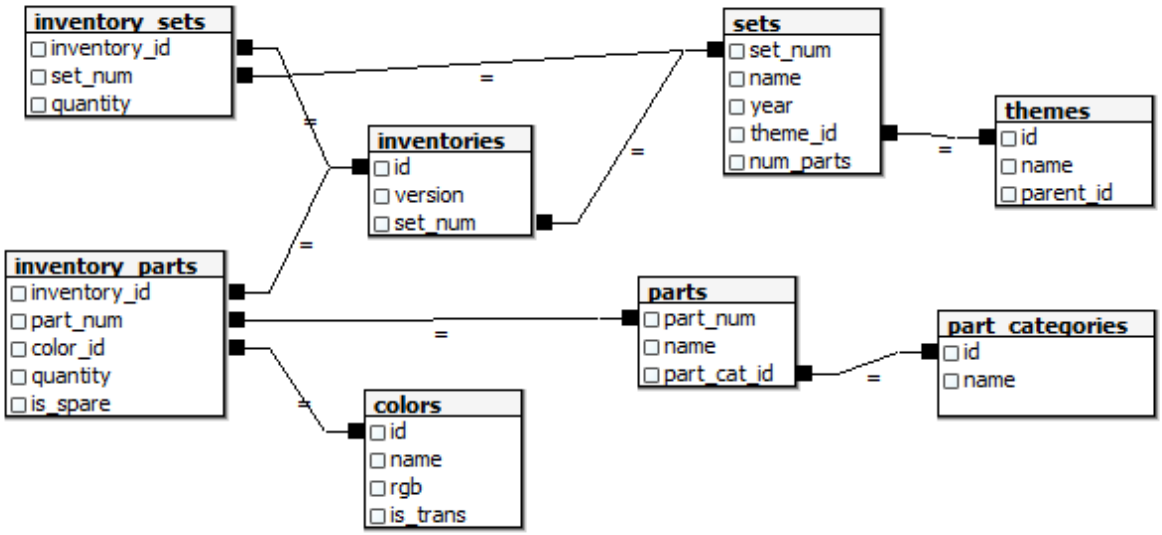
- **STEP 1.**

Utilizing the query tool in pgAdmin to type in the schema using Data Definition Language (DDL).



- **STEP 2.**

Reading the schema from the LEGO website and crafted a table creation query for each .csv file.




```
--"colors" table
-- primary key “id”.
CREATE TABLE public.colors
(
  id VARCHAR(15),
    name VARCHAR(50),
    rgb CHAR(6),
    is_trans BOOLEAN,
    primary key (id)
);

--“themes” table
--primary key “id”
CREATE TABLE public.themes
(
  id VARCHAR(15),
    name VARCHAR(100),
    parent_id VARCHAR(15),
    primary key (id)
);

--“sets” table
--primary key “set_num”
--choose theme_id as foreign key reference from “themes”
CREATE TABLE public.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)
);

--“inventories” table
--primary key “id”.
--choose set_num as foreign key reference from “sets”
CREATE TABLE public.inventories
(
  id VARCHAR(15),
    version INT,
    set_num VARCHAR(20),
    primary key (id),
    foreign key (set_num) references sets(set_num)
);

--“inventory_sets” table
--primary key “inventory_id” and “set_num”
--choose inventory_id as foreign key reference from “inventories”
--choose set_num as foreign key reference from “sets”
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)
);

--“part_categories” table
--primary key “id”
CREATE TABLE public.part_categories
(
    id VARCHAR(15),
    name VARCHAR(100),
    primary key (id)
);

--“parts” table
--primary key “part_num”
--choose part_cat_id as foreign key reference 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)
);

--“inventory_parts” table
```

```
--no primary key because there is no unique attribute in “inventory_parts.csv”
--choose inventory_id as foreign key reference from "inventories"
--choose color_id as foreign key reference from "colors"
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)
);
```

- **STEP 3.**

Importing .csv files using the SQL query tool and placing all the files under 'C:/Program Files/PostgreSQL/16/bin/' so that PostgreSQL can directly access the data from the 'bin' folder. This allows me to import the files in this manner.

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

COPY public.themes(id,name,parent_id)
FROM 'C:/Program Files/PostgreSQL/16/bin/themes.csv'
DELIMITER ','
CSV HEADER;

COPY public.sets(set_num,name,year,theme_id,num_parts)
FROM 'C:/Program Files/PostgreSQL/16/bin/sets.csv'
DELIMITER ','
CSV HEADER;

COPY public.inventories(id,version,set_num)
FROM 'C:/Program Files/PostgreSQL/16/bin/inventories.csv'
DELIMITER ','
CSV HEADER;

COPY public.inventory_sets(inventory_id,set_num,quantity)
FROM 'C:/Program Files/PostgreSQL/16/bin/inventory_sets.csv'
DELIMITER ','
CSV HEADER;

COPY public.part_categories(id,name)
FROM 'C:/Program Files/PostgreSQL/16/bin/part_categories.csv'
DELIMITER ','
```

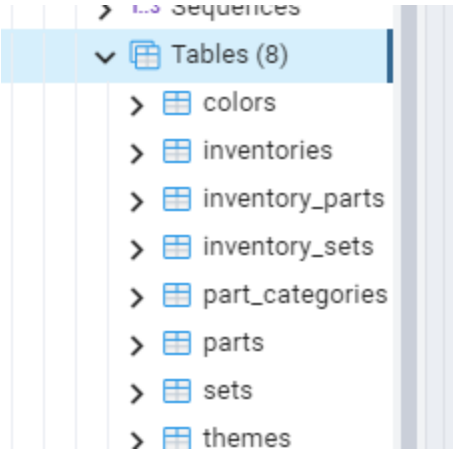
```
CSV HEADER;

COPY public.parts(part_num,name,part_cat_id)
FROM 'C:/Program Files/PostgreSQL/16/bin/parts.csv'
DELIMITER ','
CSV HEADER;

COPY public.inventory_parts(inventory_id,part_num,color_id,quantity,is_spare)
FROM 'C:/Program Files/PostgreSQL/16/bin/inventory_parts.csv'
DELIMITER ','
CSV HEADER;
```

🔗: **Link to all the query code on github**

In the end, eight tables were created in accordance with the schema rules provided by the LEGO website.



Q3. extract the name of the set and name of the theme of all the LEGO sets published in 2017.(4a.)

Query Query History

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

Messages Notifications Data Output

	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

Total rows: 296 of 296 Query complete 00:00:00.123 Ln 3,

part of the result. (total 296 datas)

🔗: pa result csv. file on github

Q4. extract the total number of LEGO sets in each year from 1950 to 2017, in descending order of total number of LEGO sets.(4b.)

Query Query History

```
SELECT COUNT(s) num_of_sets,year
FROM sets s
WHERE year <= 2017 AND year >= 1950
GROUP BY year
ORDER BY num_of_sets DESC
```

Messages Notifications Data Output

	num_of_sets 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

Total rows: 66 of 66 Query complete 00:00:00.099

part of the result. (total 66 datas)

🔗: pb result csv. file on github

Q5. extract the name of the most popular theme, defined by the number of sets in the themes.(4c.)

Query Query History

```sql
WITH themes_set(name, total_set) AS(
        SELECT t.name , COUNT(s.name)
        FROM sets s, themes t
        WHERE t.id = s.theme_id
        GROUP BY t.id)
SELECT name, total_set as max_set
FROM themes_set
WHERE total_set = (
        SELECT MAX(total_set)
        FROM themes_set
    );
```

Messages Notifications Data Output

	name character varying (100)	max_set bigint
1	Gear	246

- result:

name: “Gear”,

max_set: 246

Q6. extract the average number of parts in a set for each theme, with the name of the theme and the average number of parts per set. In ascending order of average number of parts in a set.(4d.)

Query Query History

```sql
WITH themes_avg(name, part) AS(
    SELECT t.name,AVG(s.num_parts)
    FROM sets s, themes t
    WHERE t.id = s.theme_id
    GROUP BY t.id)

SELECT name theme_name, part avg_num
FROM themes_avg
ORDER BY avg_num
```

Messages Notifications Data Output

	theme_name character varying (100)	avg_num 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

Total rows: 575 of 575 Query complete 00:00:00.102

part of the result. (total 575 datas)

🔗: p4d.result on github

Q7. find out the name of the colors that are most used in the unique LEGO parts, and list the top 10.(4e.)

Query | Query History

```sql
WITH color_use(name,num_use) AS(
    SELECT c.name,
       COUNT(DISTINCT ip.part_num)
    FROM inventory_parts ip,colors c
    WHERE c.id = ip.color_id
    GROUP BY c.id
)



SELECT c.name colors_name,cu.num_use
FROM colors c,color_use cu
WHERE c.name = cu.name
ORDER BY num_use DESC
LIMIT 10
```

Messages | Notifications | Data Output

	colors_name character varying (50)	num_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

Total rows: 10 of 10 | Query complete 00:00:04.791

result

Q8. find out the name of the colors that are most used in the LEGO parts, for each theme, and list the top 1 for each theme.(4f.)

Query Query History

```sql
WITH quantity AS (
    SELECT c.name AS color_name, ip.inventory_id, SUM(ip.quantity) AS
    FROM inventory_parts ip
    JOIN colors c ON c.id = ip.color_id
    GROUP BY c.id, ip.inventory_id, ip.part_num
),total_quantity AS (
    SELECT
        t.name AS theme_name,
        q.color_name,
        SUM(q.quantity_sum) AS total_quantity,
        RANK() OVER (PARTITION BY t.id ORDER BY SUM(q.quantity_sum) DESC) AS rank
    FROM
        themes t
        JOIN sets s ON t.id = s.theme_id
        JOIN inventories i ON s.set_num = i.set_num
        JOIN quantity q ON i.id = q.inventory_id
    GROUP BY
        t.id,
        q.color_name
)

SELECT
    theme_name,
    color_name AS most_used_color
FROM
    total_quantity
WHERE
    rank = 1
ORDER BY total_quantity.theme_name
```

Messages Notificati

	theme_name character varying (
1	12V
2	12V
3	4 Juniors
4	4.5V
5	4.5V
6	9V
7	9V
8	Advent
9	Advent Sub-Set
10	Adventurers
11	Agents
12	Agori
13	Airjitzu
14	Airport
15	Airport
16	Airport
17	Airport
18	Airport
19	Airport
20	Airport
21	Airport
22	Airport
23	Airport
24	Airport

part of the result. (total 568 datas)

🔗: pf result on github