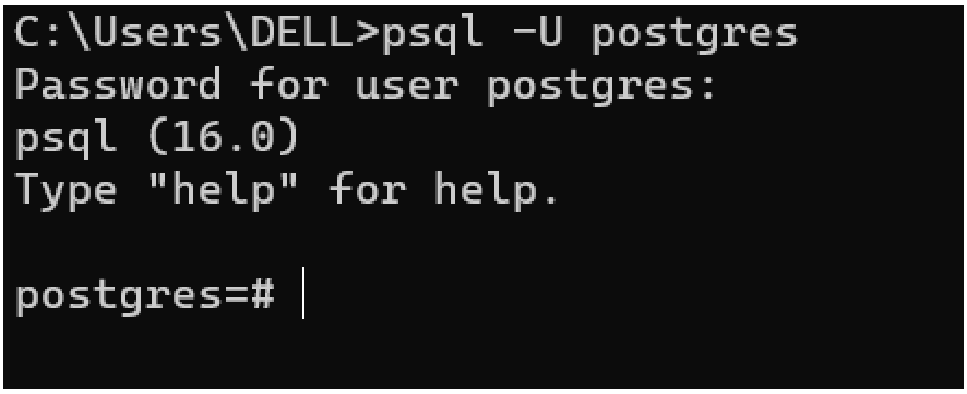
Database HW1

Created by	K Kent Lee
Created time	@October 18, 2023 11:49 PM
Tags	

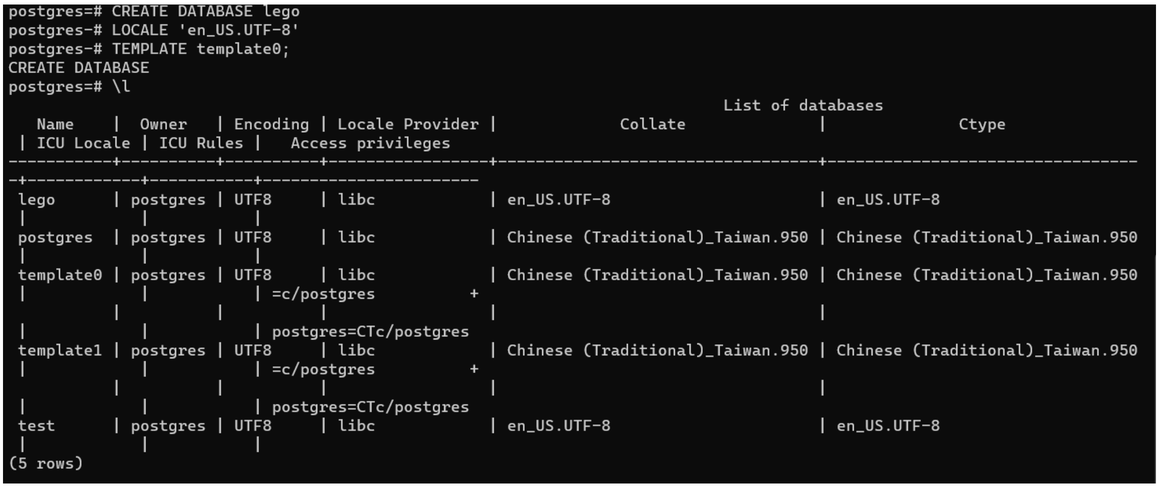
(1) The process of creating the “lego” database

- First Start up Postgresql



```
C:\Users\DELL>psql -U postgres
Password for user postgres:
psql (16.0)
Type "help" for help.

postgres=#
```

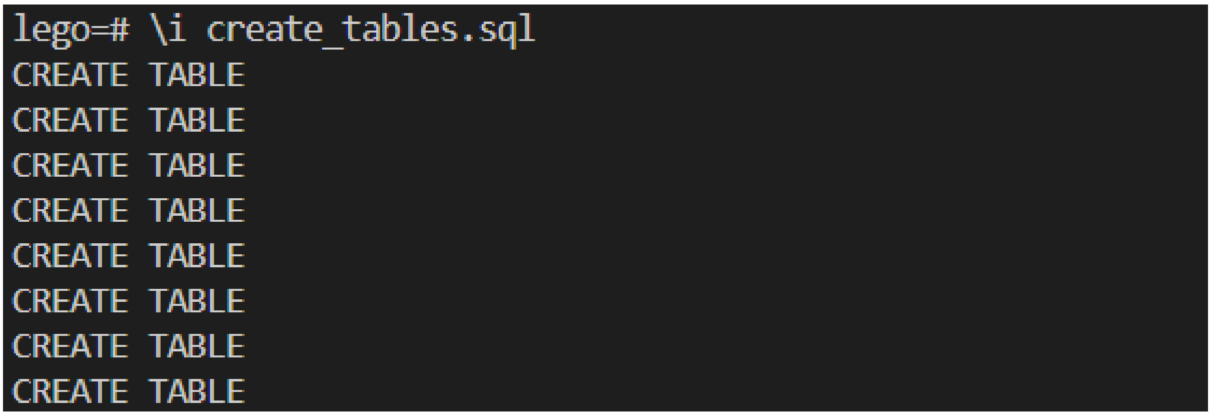
- create the database (change locale to English since we will be working with data in english, although maybe it doesn’t matter)



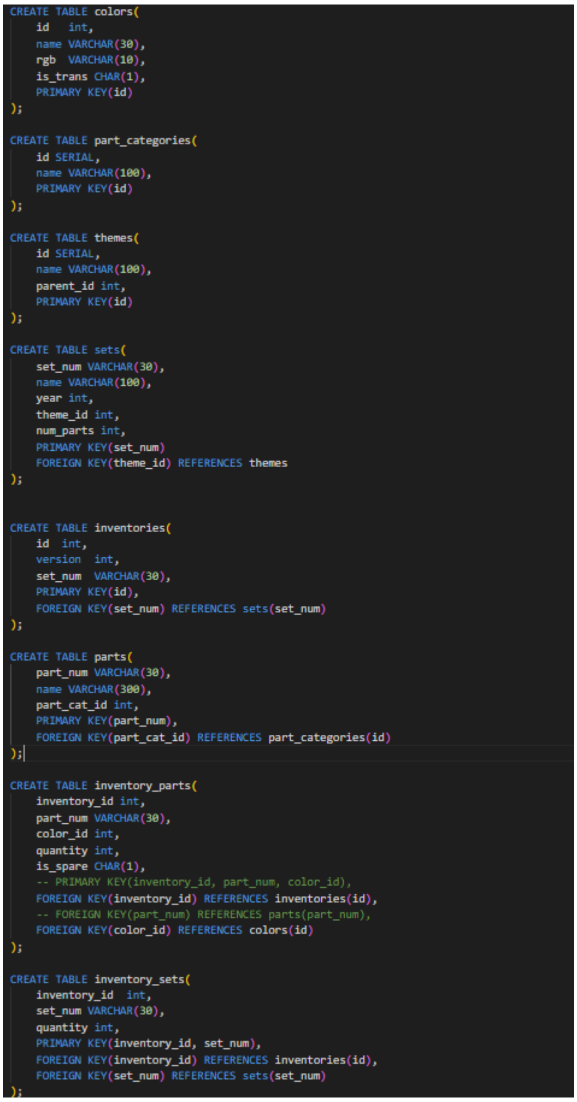
```
postgres=# CREATE DATABASE lego
postgres-# LOCALE 'en_US.UTF-8'
postgres-# TEMPLATE template0;
CREATE DATABASE
postgres=# \l
                                                                             List of databases
   Name    |  Owner   | Encoding | Locale Provider |             Collate              |              Ctype
 | ICU Locale | ICU Rules |   Access privileges
-----------+----------+----------+-----------------+----------------------------------+---------------------------------
-+------------+-----------+-----------------------
 lego      | postgres | UTF8     | libc            | en_US.UTF-8                      | en_US.UTF-8
 |            |           |
 postgres  | postgres | UTF8     | libc            | Chinese (Traditional)_Taiwan.950 | Chinese (Traditional)_Taiwan.950
 |            |           |
 template0 | postgres | UTF8     | libc            | Chinese (Traditional)_Taiwan.950 | Chinese (Traditional)_Taiwan.950
 |            |           | =c/postgres          +
           |          |          |                 |                                  |
 |            |           | postgres=CTc/postgres
 template1 | postgres | UTF8     | libc            | Chinese (Traditional)_Taiwan.950 | Chinese (Traditional)_Taiwan.950
 |            |           | =c/postgres          +
           |          |          |                 |                                  |
 |            |           | postgres=CTc/postgres
 test      | postgres | UTF8     | libc            | en_US.UTF-8                      | en_US.UTF-8
 |            |           |
(5 rows)
```

(2) The process of importing eight required .csv files into lego database

- create tables for each csv



```
lego=# \i create_tables.sql
CREATE TABLE
CREATE TABLE
CREATE TABLE
CREATE TABLE
CREATE TABLE
CREATE TABLE
CREATE TABLE
CREATE TABLE
```



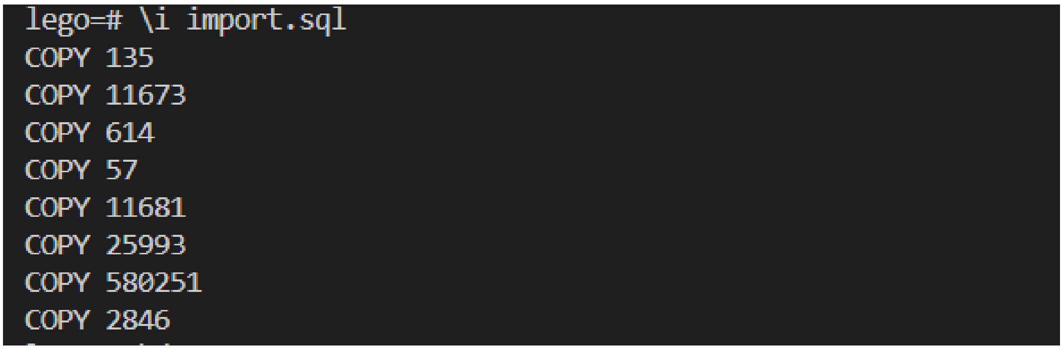
```
CREATE TABLE colors(
    id   int,
    name VARCHAR(30),
    rgb  VARCHAR(10),
    is_trans CHAR(1),
    PRIMARY KEY(id)
);

CREATE TABLE part_categories(
    id SERIAL,
    name VARCHAR(100),
    PRIMARY KEY(id)
);

CREATE TABLE themes(
    id SERIAL,
    name VARCHAR(100),
    parent_id int,
    PRIMARY KEY(id)
);

CREATE TABLE sets(
    set_num VARCHAR(30),
    name VARCHAR(100),
    year int,
    theme_id int,
    num_parts int,
    PRIMARY KEY(set_num)
    FOREIGN KEY(theme_id) REFERENCES themes
);


CREATE TABLE inventories(
    id  int,
    version  int,
    set_num  VARCHAR(30),
    PRIMARY KEY(id),
    FOREIGN KEY(set_num) REFERENCES sets(set_num)
);

CREATE TABLE parts(
    part_num VARCHAR(30),
    name VARCHAR(300),
    part_cat_id int,
    PRIMARY KEY(part_num),
    FOREIGN KEY(part_cat_id) REFERENCES part_categories(id)
);

CREATE TABLE inventory_parts(
    inventory_id int,
    part_num VARCHAR(30),
    color_id int,
    quantity int,
    is_spare CHAR(1),
    -- PRIMARY KEY(inventory_id, part_num, color_id),
    FOREIGN KEY(inventory_id) REFERENCES inventories(id),
    -- FOREIGN KEY(part_num) REFERENCES parts(part_num),
    FOREIGN KEY(color_id) REFERENCES colors(id)
);

CREATE TABLE inventory_sets(
    inventory_id  int,
    set_num VARCHAR(30),
    quantity int,
    PRIMARY KEY(inventory_id, set_num),
    FOREIGN KEY(inventory_id) REFERENCES inventories(id),
    FOREIGN KEY(set_num) REFERENCES sets(set_num)
);
```

Chose datatypes for each attribute of each relation by going through the .csv file, making sure all data of the dataset can fit into these tables. For example, parts(name) is of type VARCHAR(300) because some of its entries exceed even 200 characters.

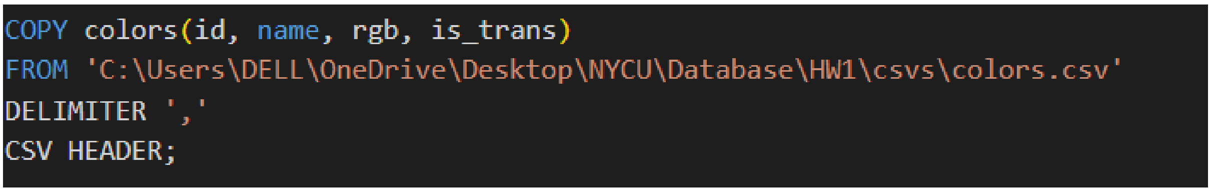
Next is to choose primary keys. For most tables it's quite simple to choose, however for *inventory_parts* there the only set of attributes that uniquely identifies all entries is choosing all attributes, so I chose to omit it initially.

For foreign keys I found that the schema diagram provided isn't completely aligned with the dataset, the diagram shows inventory_parts(part_num) references parts(part_num), however value 48002 exists only in *inventory_parts*, and is absent in *parts.*

- COPY data from .csv files to tables



```
lego=# \i import.sql
COPY 135
COPY 11673
COPY 614
COPY 57
COPY 11681
COPY 25993
COPY 580251
COPY 2846
```

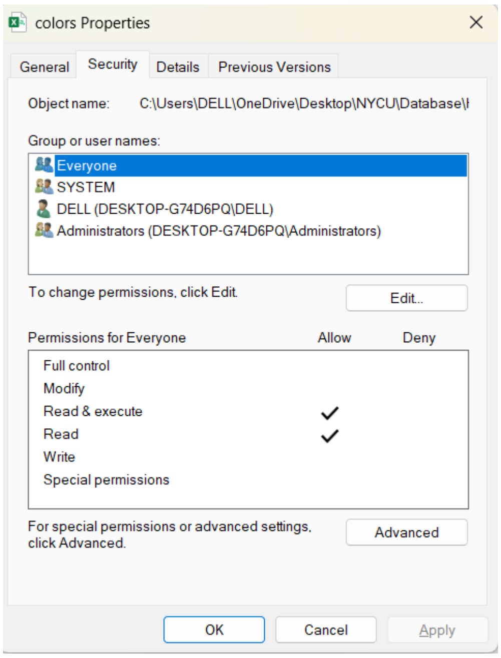


```
COPY colors(id, name, rgb, is_trans)
FROM 'C:\Users\DELL\OneDrive\Desktop\NYCU\Database\HW1\csvs\colors.csv'
DELIMITER ','
CSV HEADER;
```

example code for importing

Similar syntax is used for each .csv file, as shown in the example code above.

A issue I ran into is *permission denied*, solved it by going into the properties of each .csv file and enable its for everyone to read. as shown in the picture on the right.



Object name: C:\Users\DELL\OneDrive\Desktop\NYCU\Database\I

Group or user names:
Everyone
SYSTEM
DELL (DESKTOP-G74D6PQ\DELL)
Administrators (DESKTOP-G74D6PQ\Administrators)

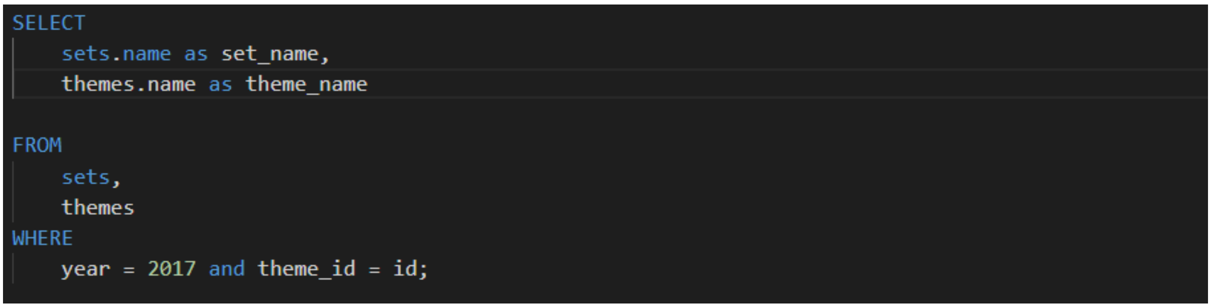
To change permissions, click Edit.

Permissions for Everyone	Allow	Deny
Full control		
Modify		
Read & execute	✓	
Read	✓	
Write		
Special permissions		

For special permissions or advanced settings, click Advanced.

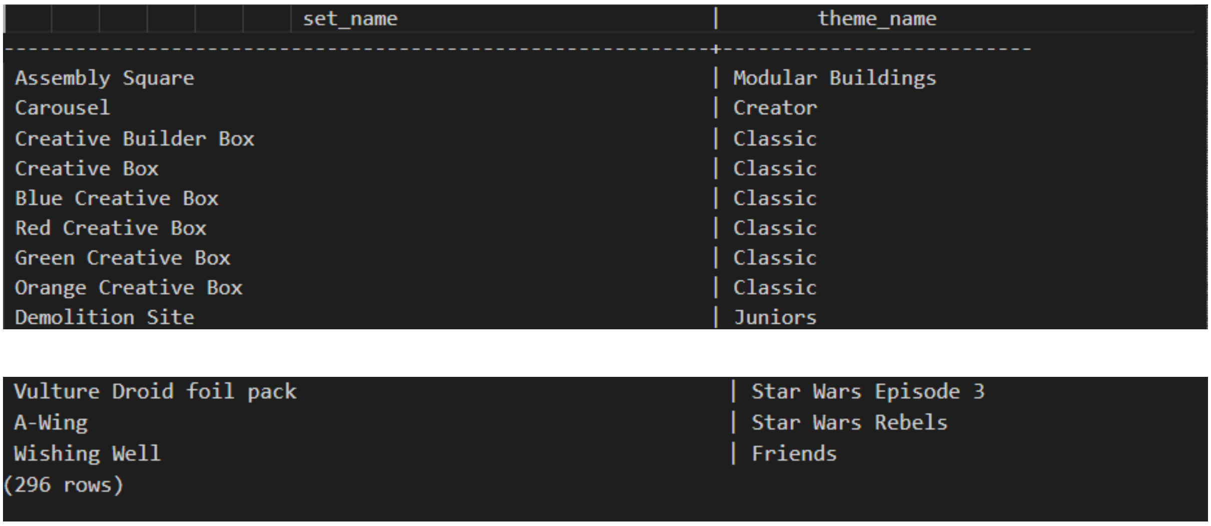
Another Issue I encountered is encoding. It keeps showing that certain character sequences are in BIG5 encoding, even though they all seem to be in UTF-8 when inspected with Notepad. Solved it by adding the command `\encoding UTF-8` in the beginning of import.sql.

(3) extract the name of the set and name of the theme of all the LEGO sets published in 2017



```
SELECT
    sets.name as set_name,
    themes.name as theme_name

FROM
    sets,
    themes
WHERE
    year = 2017 and theme_id = id;
```

SQL statement



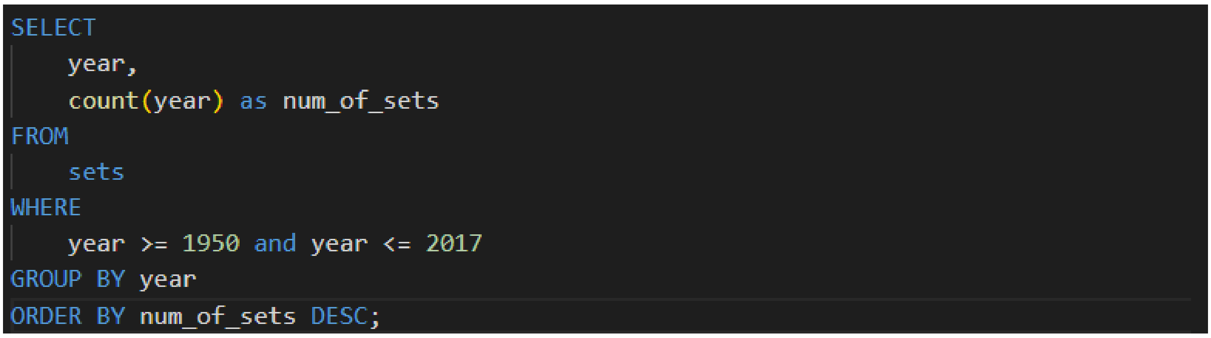
```
                         set_name                         |         theme_name
----------------------------------------------------------+-------------------------
 Assembly Square                                          | Modular Buildings
 Carousel                                                 | Creator
 Creative Builder Box                                     | Classic
 Creative Box                                             | Classic
 Blue Creative Box                                        | Classic
 Red Creative Box                                         | Classic
 Green Creative Box                                       | Classic
 Orange Creative Box                                      | Classic
 Demolition Site                                          | Juniors
```

```
 Vulture Droid foil pack                                  | Star Wars Episode 3
 A-Wing                                                   | Star Wars Rebels
 Wishing Well                                             | Friends
(296 rows)
```

partial query output for 4a

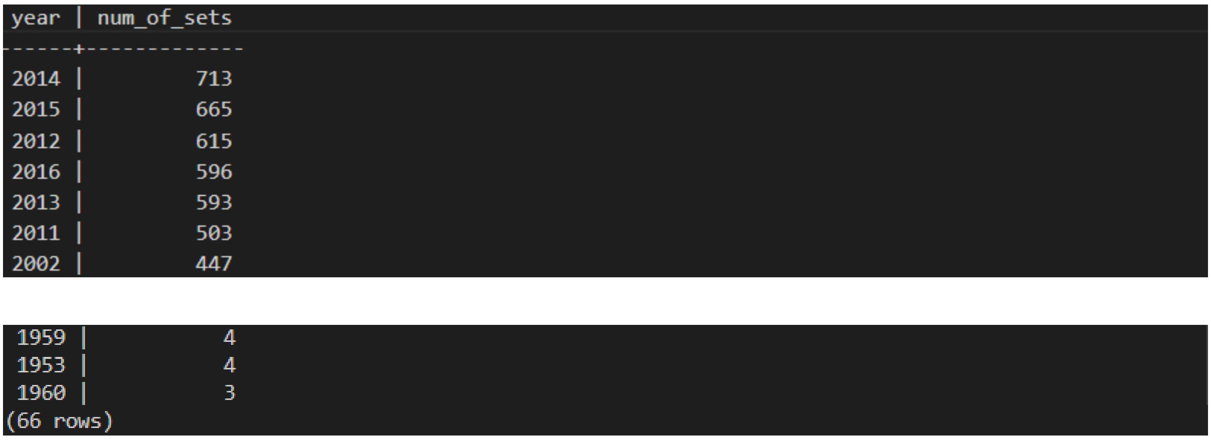
full query output: Database-HW1/results/query_result_4a.txt at master · Kent-mak/Database-HW1 (github.com)

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



```
SELECT
    year,
    count(year) as num_of_sets
FROM
    sets
WHERE
    year >= 1950 and year <= 2017
GROUP BY year
ORDER BY num_of_sets DESC;
```

SQL statement



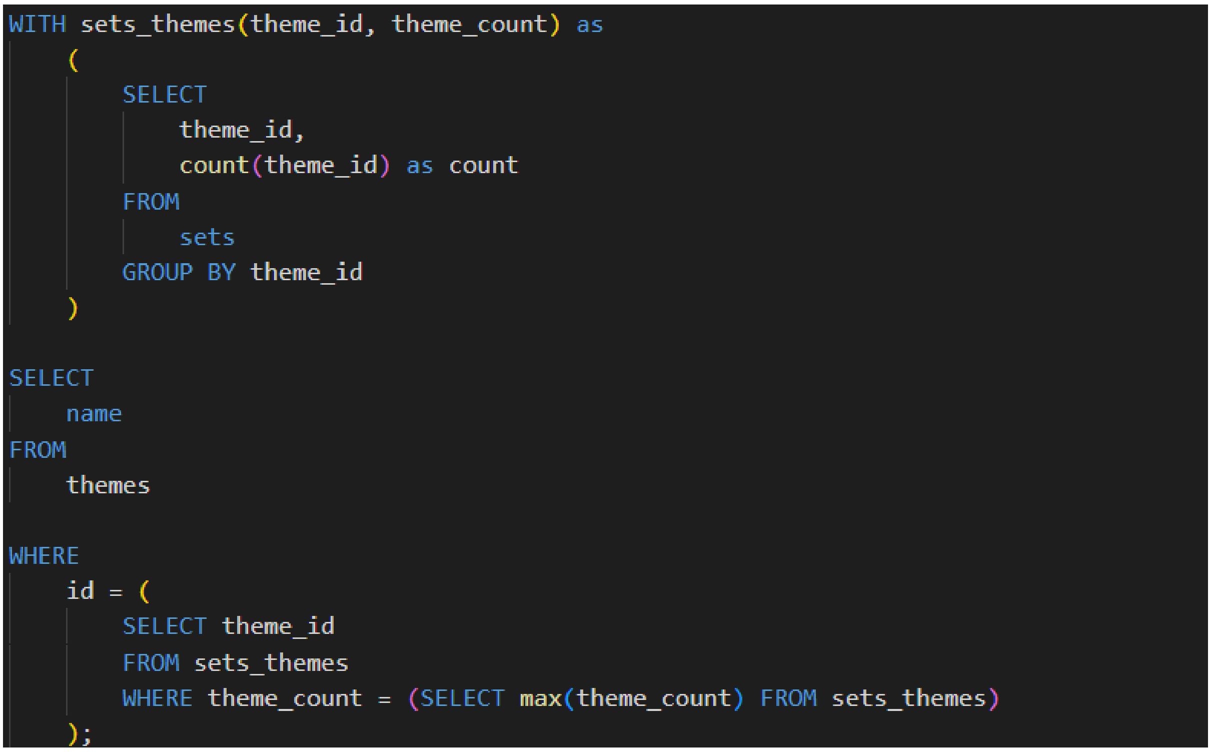
```
year | num_of_sets
-----+-------------
2014 |         713
2015 |         665
2012 |         615
2016 |         596
2013 |         593
2011 |         503
2002 |         447
```

```
 1959 |           4
 1953 |           4
 1960 |           3
(66 rows)
```

partial query output for 4b

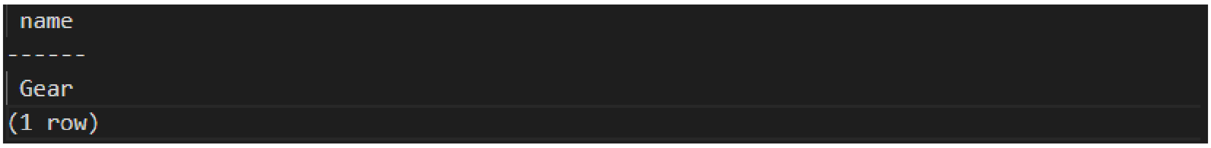
full query output: Database-HW1/results/query_result_4b.txt at master · Kent-mak/Database-HW1 (github.com)

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



```
WITH sets_themes(theme_id, theme_count) as
    (
        SELECT
            theme_id,
            count(theme_id) as count
        FROM
            sets
        GROUP BY theme_id
    )

SELECT
    name
FROM
    themes

WHERE
    id = (
        SELECT theme_id
        FROM sets_themes
        WHERE theme_count = (SELECT max(theme_count) FROM sets_themes)
    );
```

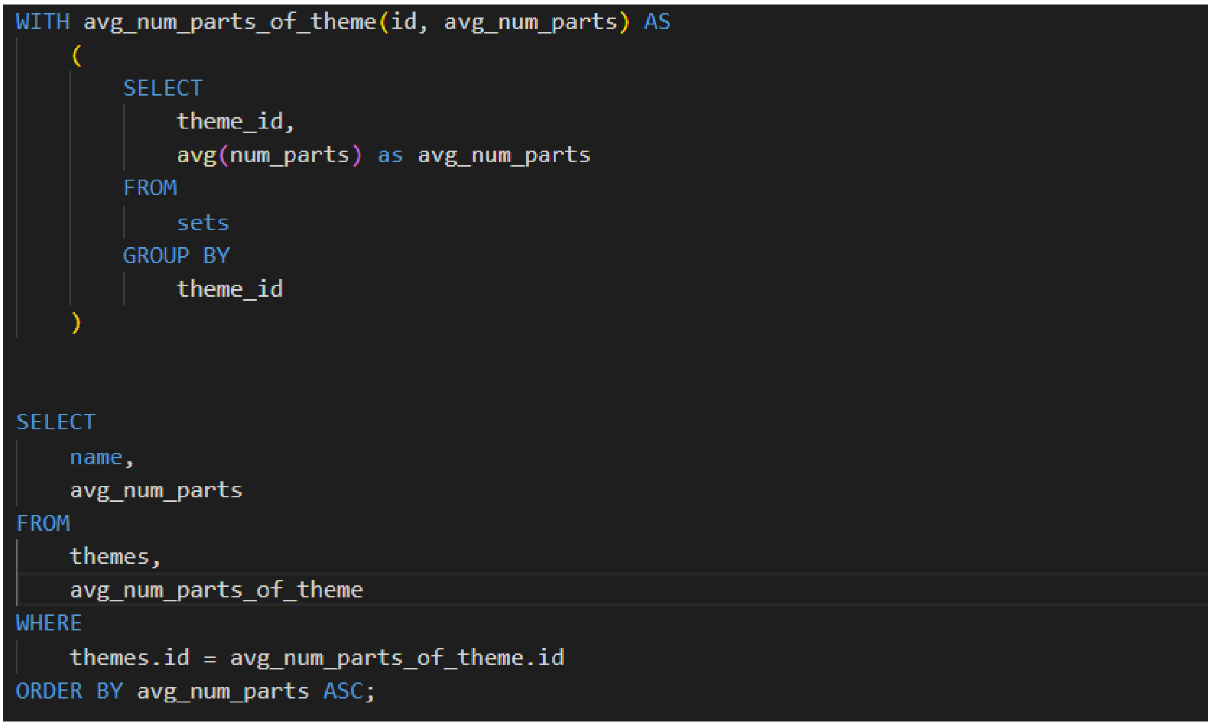
SQL statement



```
 name
------
 Gear
(1 row)
```

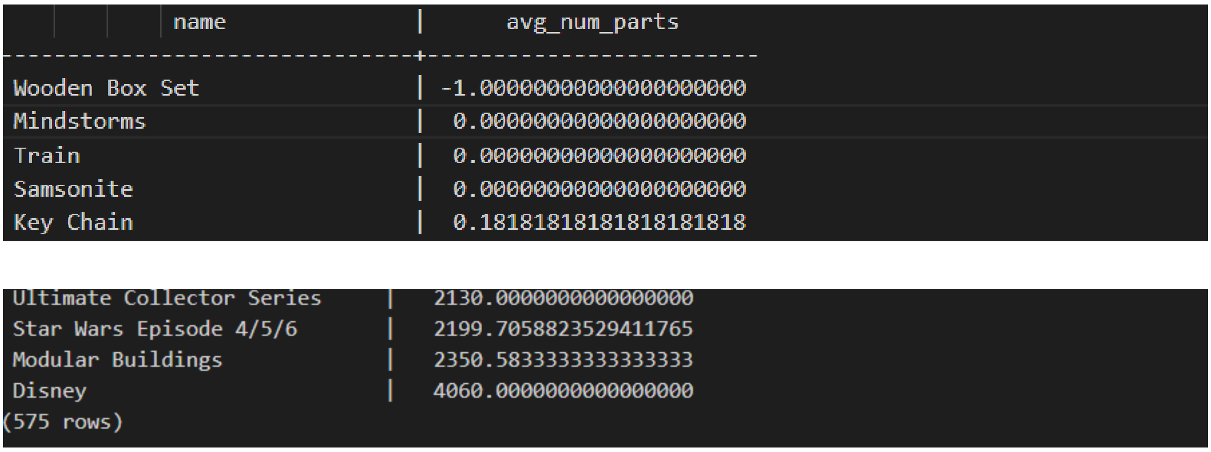
query output for 4c

(6) 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



```
WITH avg_num_parts_of_theme(id, avg_num_parts) AS
    (
        SELECT
            theme_id,
            avg(num_parts) as avg_num_parts
        FROM
            sets
        GROUP BY
            theme_id
    )


SELECT
    name,
    avg_num_parts
FROM
    themes,
    avg_num_parts_of_theme
WHERE
    themes.id = avg_num_parts_of_theme.id
ORDER BY avg_num_parts ASC;
```

SQL statement



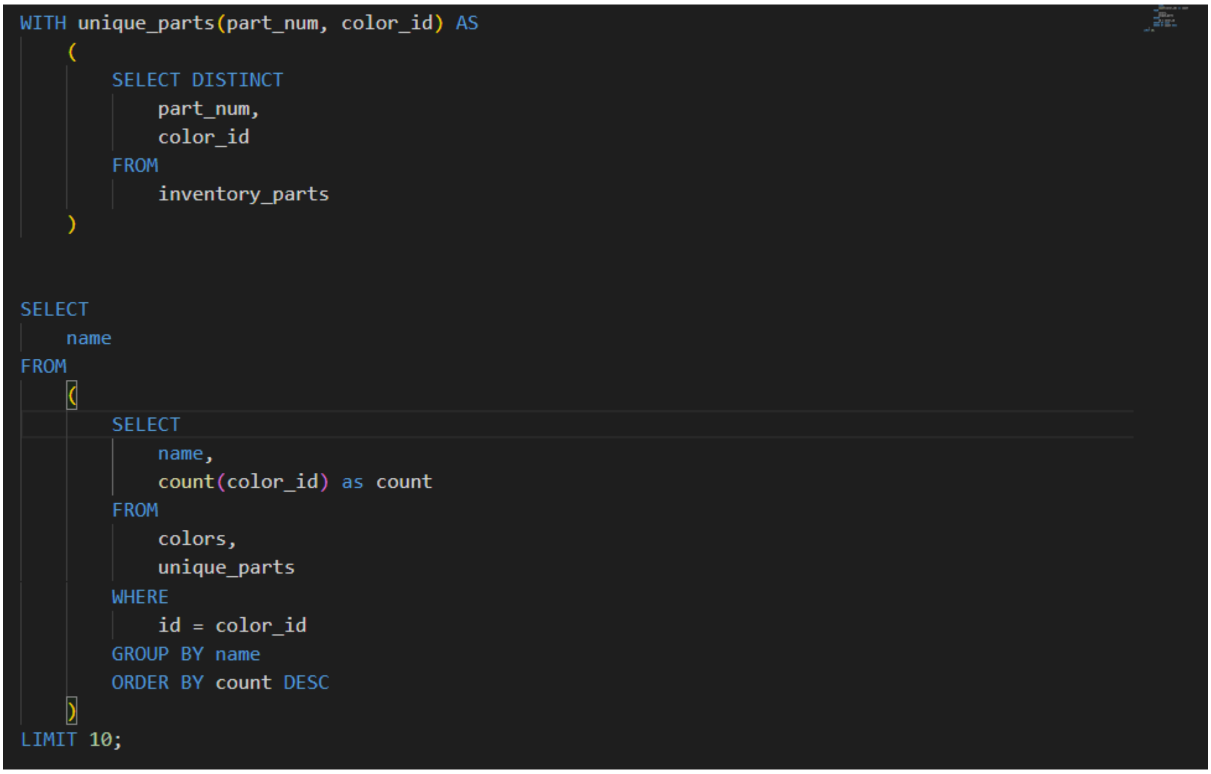
```
                name               |       avg_num_parts
-------------------------------+-------------------------
Wooden Box Set                 | -1.00000000000000000000
Mindstorms                     |  0.00000000000000000000
Train                          |  0.00000000000000000000
Samsonite                      |  0.00000000000000000000
Key Chain                      |  0.18181818181818181818
```

```
Ultimate Collector Series      |   2130.0000000000000000
Star Wars Episode 4/5/6        |   2199.7058823529411765
Modular Buildings              |   2350.5833333333333333
Disney                         |   4060.0000000000000000
(575 rows)
```

partial query output for 4d

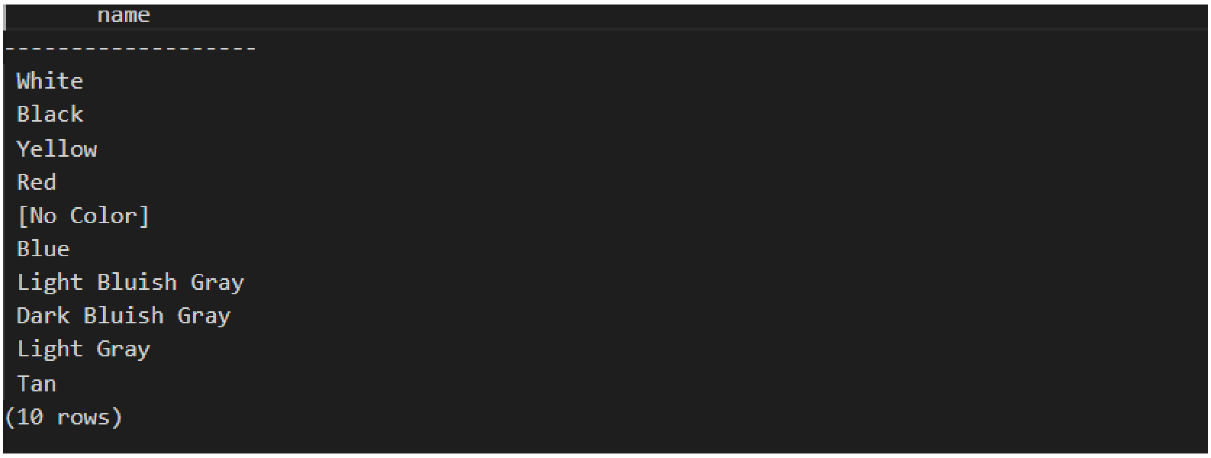
full query output: Database-HW1/results/query_result_4d.txt at master · Kent-mak/Database-HW1 (github.com)

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



```sql
WITH unique_parts(part_num, color_id) AS
    (
        SELECT DISTINCT
            part_num,
            color_id
        FROM
            inventory_parts
    )


SELECT
    name
FROM
    (
        SELECT
            name,
            count(color_id) as count
        FROM
            colors,
            unique_parts
        WHERE
            id = color_id
        GROUP BY name
        ORDER BY count DESC
    )
LIMIT 10;
```

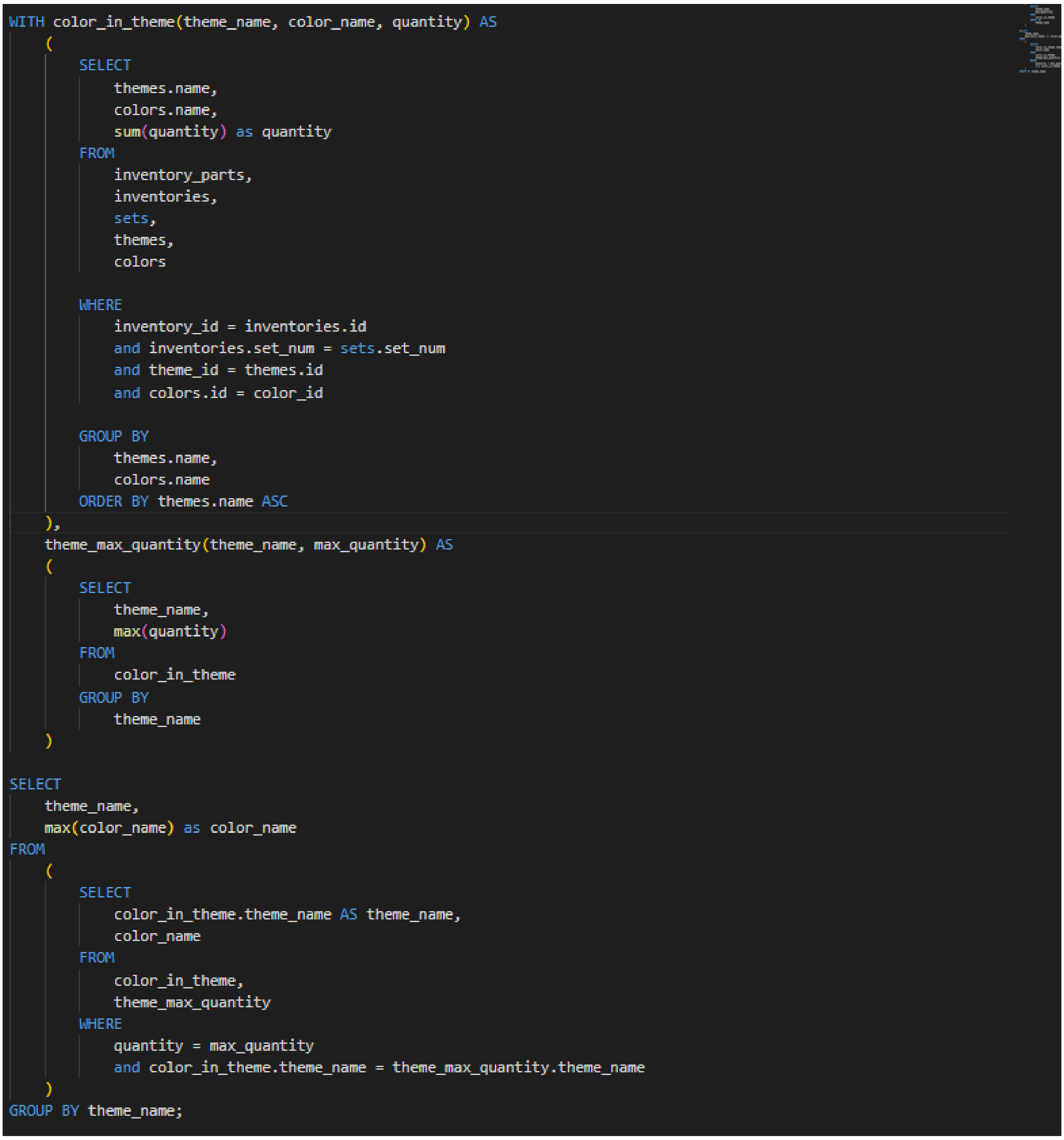
SQL statements



```
        name
-------------------
 White
 Black
 Yellow
 Red
 [No Color]
 Blue
 Light Bluish Gray
 Dark Bluish Gray
 Light Gray
 Tan
(10 rows)
```

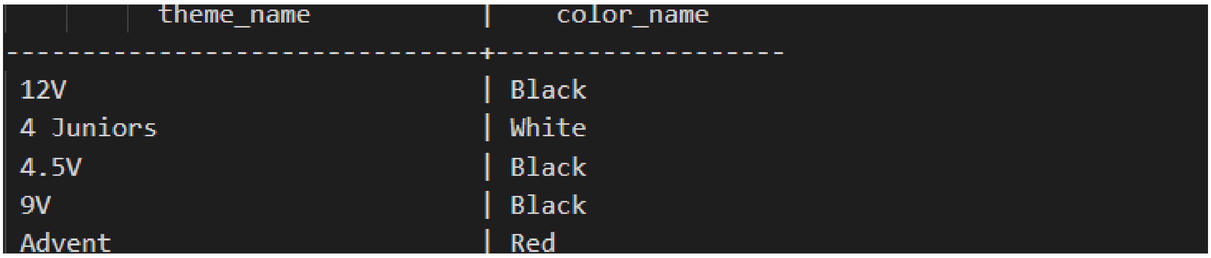
query output for 4e

(8) 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 (please provide the name of the theme, too).

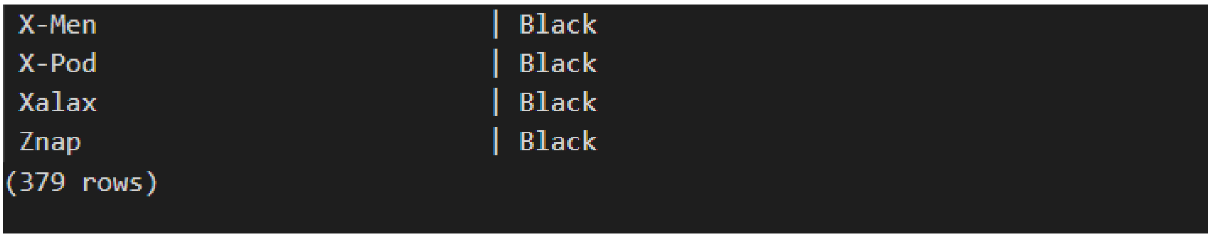


```
WITH color_in_theme(theme_name, color_name, quantity) AS
    (
        SELECT
            themes.name,
            colors.name,
            sum(quantity) as quantity
        FROM
            inventory_parts,
            inventories,
            sets,
            themes,
            colors

        WHERE
            inventory_id = inventories.id
            and inventories.set_num = sets.set_num
            and theme_id = themes.id
            and colors.id = color_id

        GROUP BY
            themes.name,
            colors.name
        ORDER BY themes.name ASC
    ),
    theme_max_quantity(theme_name, max_quantity) AS
    (
        SELECT
            theme_name,
            max(quantity)
        FROM
            color_in_theme
        GROUP BY
            theme_name
    )

SELECT
    theme_name,
    max(color_name) as color_name
FROM
    (
        SELECT
            color_in_theme.theme_name AS theme_name,
            color_name
        FROM
            color_in_theme,
            theme_max_quantity
        WHERE
            quantity = max_quantity
            and color_in_theme.theme_name = theme_max_quantity.theme_name
    )
GROUP BY theme_name;
```

SQL statement



```
          theme_name              |     color_name
-------------------------------+-------------------
 12V                           | Black
 4 Juniors                     | White
 4.5V                          | Black
 9V                            | Black
 Advent                        | Red
```



```
 X-Men                           | Black
 X-Pod                           | Black
 Xalax                           | Black
 Znap                            | Black
(379 rows)
```

partial query output for 4f

full query output: Database-HW1/results/query_result_4f.txt at master · Kent-mak/Database-HW1 (github.com)