



---

CARRERA DE ESPECIALIZACIÓN EN INTELIGENCIA ARTIFICIAL

---

## **Bases de Datos para Inteligencia Artificial**

16Co2024

### **Trabajo Práctico Final**

#### ***Análisis con la Base de Datos Relacional de LEGO***

DEGANO, MYRNA LORENA

N° SIU a1618

[myrna.l.degano@gmail.com](mailto:myrna.l.degano@gmail.com)

abril de 2025

Índice

Objetivo ..... 2

SetUp..... 2

    Modelo de datos ..... 2

    Conjunto de datos ..... 2

    Creación del entorno de trabajo ..... 6

Consultas..... 9

    1. Colores más utilizados en los 90 ..... 9

    2. Colores únicos ..... 11

    3. Tendencia de piezas por sets a lo largo de los años ..... 15

    4. Temáticas más populares de los 2000 ..... 18

Anexos..... 24

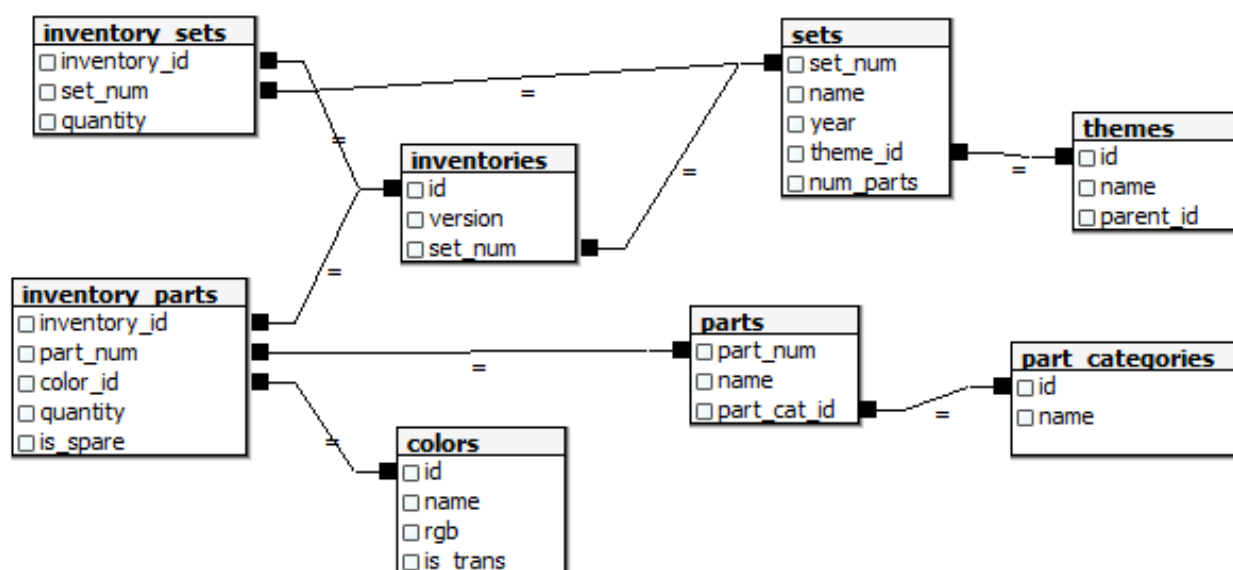
Referencias..... 24

## Objetivo

Trabajar con la base de datos de LEGO, colección de datos relacionada con los famosos bloques de construcción. La base de datos se encuentra en formato relacional.

## SetUp

### Modelo de datos











### Conjunto de datos

Provisto en archivos de texto separado por comas, uno por cada entidad del modelo relacional:

- **colors.csv**

Esta tabla contiene información sobre los colores LEGO.







- **id** (PK) - Identificador único del color.
- **name** - Nombre del color.
- **rgb** - Color RGB aproximado.
- **is\_trans** - Flag que indica si el color es transparente/traslúcido o no.

 id 	 name 	 rgb 	 is_trans 
-1	Unknown	0033B2	f
0	Black	05131D	f
1	Blue	0055BF	f
2	Green	237841	f

- **inventories.csv**

Esta tabla contiene información sobre los inventarios.

- **id** (PK) - Identificador único del inventario.
- **version** - Número de versión.
- **set\_num** (FK 'sets.csv') - Número de conjunto (set).

 id 	 version 	 set_num 
1	1	7922-1
3	1	3931-1
4	1	6942-1
15	1	5158-1

- **inventory\_parts.csv**

Esta tabla contiene información sobre las piezas de los inventarios.

- **inventory\_id** (PK) (FK 'inventories.csv') - Identificador único del inventario en el que aparece esta pieza.
- **part\_num** (PK) (FK 'parts.csv') - Identificador único de la pieza.
- **color\_id** (PK) (FK 'colors.csv') - Identificador único del color.
- **quantity** - El número de copias de la pieza incluidas en el set.
- **is\_spare** (PK) - Flag que indica si se trata de una pieza de repuesto o no. Las piezas de repuesto son piezas adicionales que no se necesitan para completar el set.

inventory_id	part_num	color_id	# quantity	is_spare
1	48379c01	72	1	f
1	48395	7	1	f
1	mcsport6	25	1	f
1	paddle	0	1	f

*\* Nota: La FK de part\_num no puede implementarse en el modelo físico porque los datos no son consistentes con esa definición. Varios de los registros en inventory\_parts.csv hacen referencia a números de piezas que no existen en parts.csv.*

- **inventory\_sets.csv**

Esta tabla contiene información sobre qué inventario está incluido en qué conjuntos.

- **inventory\_id** (PK) (FK 'inventories.csv') - Identificador único del inventario.
- **set\_num** (PK)(FK 'sets.csv')- Identificador único del set.
- **quantity** - La cantidad del inventario incluido.

inventory_id	set_num	# quantity
35	75911-1	1
35	75912-1	1
39	75048-1	1
39	75053-1	1

- **part\_categories.csv**

Esta tabla sobre la categoría de la pieza (qué tipo de pieza es).

- **id** (PK) - Identificador único de la categoría de pieza.
- **name** - Nombre de la categoría de material a la que pertenece la pieza.

id	name
1	Baseplates
2	Bricks Printed
3	Bricks Sloped
4	Duplo, Quatro and Primo

- **parts.csv**

Esta tabla incluye información sobre piezas de Lego.

- **part\_num** (PK) - Identificador único de la pieza.
- **name** - Nombre de la pieza.
- **part\_cat\_id** (FK 'part\_categories.csv') - Identificador único de la categoría de la pieza.

▲ part_num	▲ name	↻ part_cat_id
0687b1	Set 0687 Activity Booklet 1	17
0901	Baseplate 16 x 30 with Set 080 Yellow House Print	1

- **sets.csv**

Esta tabla contiene información sobre los conjuntos (sets) LEGO.

- **set\_num** (PK) - Identificador único del set.
- **name** - Nombre del set.
- **year** - Año en que fue publicado.
- **theme\_id** (FK 'themes.csv') - Identificador único del tema usado para el set.
- **num\_parts** - Número de partes incluidas en el set.

▲ set_num	▲ name	# year	↻ theme_id	# num_parts
00-1	Weetabix Castle	1970	414	471
0011-2	Town Mini- Figures	1978	84	12
0011-3	Castle 2 for 1 Bonus Offer	1987	199	2
0012-1	Space Mini- Figures	1979	143	12

- **themes.csv**

Esta tabla incluye información sobre los temas de LEGO.

- **id** (PK) - Identificador único del tema.
- **name** - Nombre del tema.

- **parent\_id** (FK 'themes.csv') - Identificador único de un tema padre, si existe.

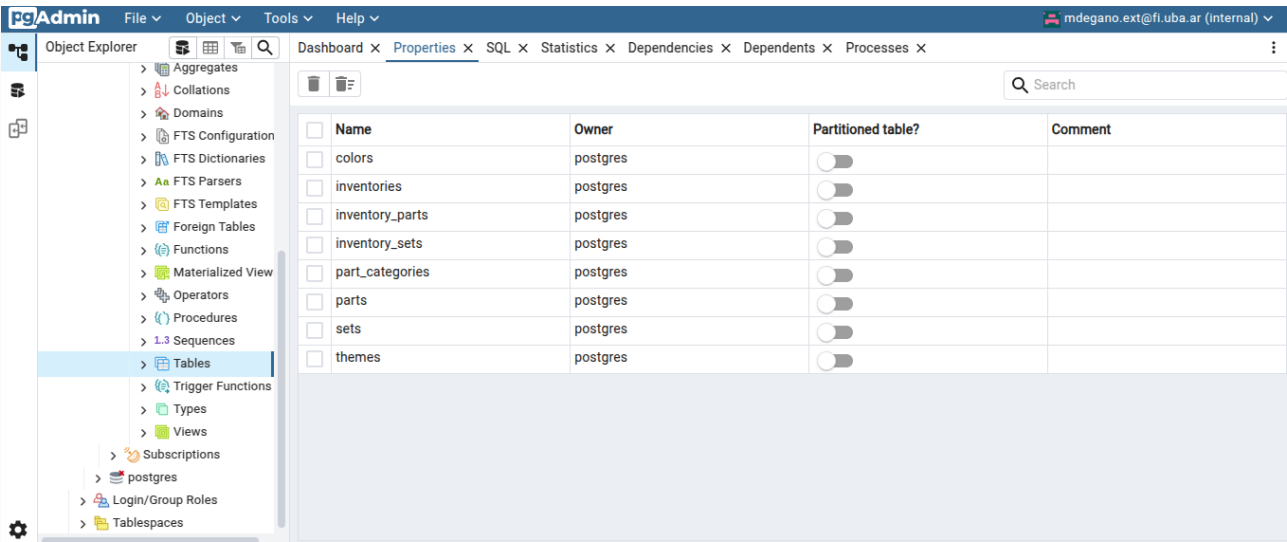
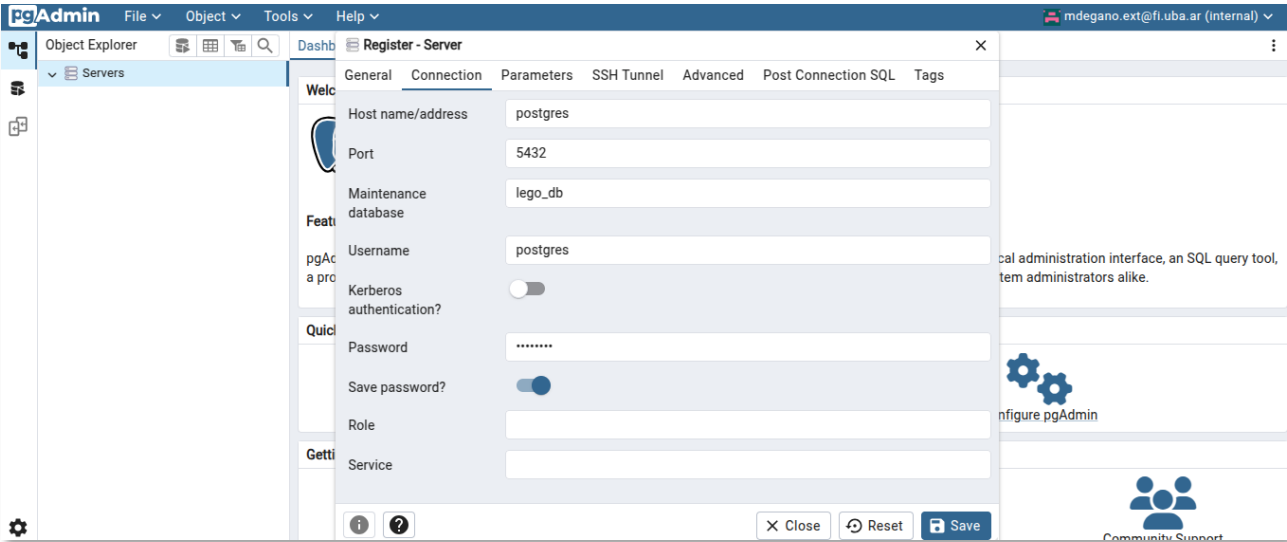
id	name	parent_id
1	Technic	
2	Arctic Technic	1
3	Competition	1
4	Expert Builder	1

## Creación del entorno de trabajo

El objetivo principal es montar esta base de datos en Postgres.

Por lo tanto, creo el archivo **init\_lego\_db.sql** con las sentencias DDL para crear las tablas según la estructura definida y cargar los datos a partir de los archivos \*.csv.

Incluyo este archivo de inicialización en el archivo **docker-compose.yaml** para crear el entorno **Postgres** con la base de datos inicializada por defecto y el cliente **pgAdmin** para realizar consultas sobre la base de datos. (ver [Anexos](#)).





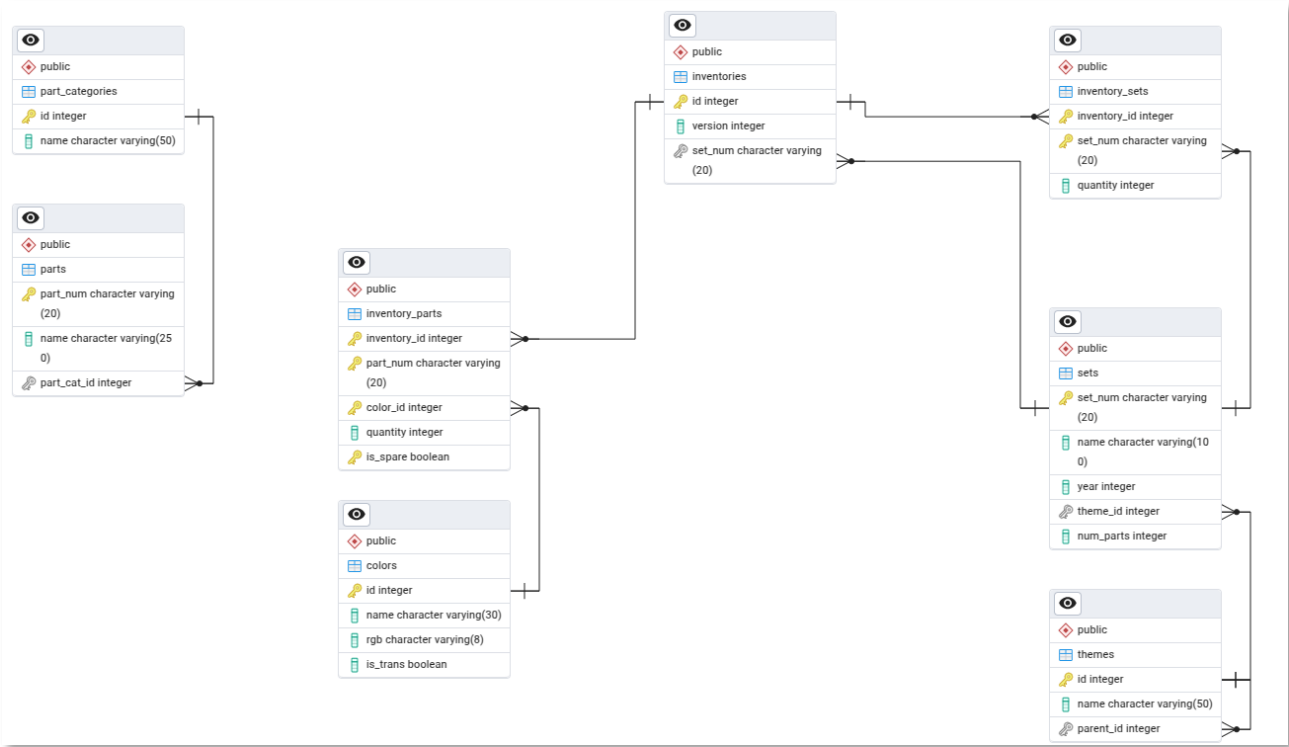


Diagrama de entidad-relación generado a partir de la DB implementada.

## Consultas

### 1. Colores más utilizados en los 90

Identificar cuáles son los 10 colores más frecuentemente usados en los sets de LEGO durante la década de los 90.

```
SELECT
c.name AS color_name, c.rgb,
SUM(i_p.quantity) AS total_quantity
FROM
sets AS s
JOIN inventories AS i ON (s.set_num = i.set_num)
JOIN inventory_parts AS i_p ON (i.id = i_p.inventory_id)
JOIN colors AS c ON (i_p.color_id = c.id)
WHERE
s.year BETWEEN 1990 AND 1999
GROUP BY
c.name, c.rgb
ORDER BY
total_quantity DESC
LIMIT 10;
```

Query Query History

```
1  --1. Colores más utilizados en los 90
2  SELECT
3  c.name AS color_name, c.rgb,
4  SUM(i_p.quantity) AS total_quantity
5  FROM
6  sets AS s
7  JOIN inventories AS i ON (s.set_num = i.set_num)
8  JOIN inventory_parts AS i_p ON (i.id = i_p.inventory_id)
9  JOIN colors AS c ON (i_p.color_id = c.id)
10 WHERE
11 s.year BETWEEN 1990 AND 1999
12 GROUP BY
13 c.name, c.rgb
14 ORDER BY
15 total_quantity DESC
16 LIMIT 10;
```

Considero la suma total de piezas involucradas en sets publicados entre los años 1990 y 1999.  
Los 10 colores que más ocurrencias tuvieron son:

Data Output Messages Notifications			
Showing rows: 1 to 10			
	color_name character varying (30)	rgb character varying (8)	total_quantity bigint
1	Black	05131D	62515
2	Light Gray	9BA19D	37905
3	White	FFFFFF	33168
4	Red	C91A09	30702
5	Yellow	F2CD37	25818
6	Blue	0055BF	19590
7	Dark Gray	6D6E5C	5478
8	Green	237841	5449
9	Brown	583927	3072
10	Trans-Neon Green	F8F184	1735

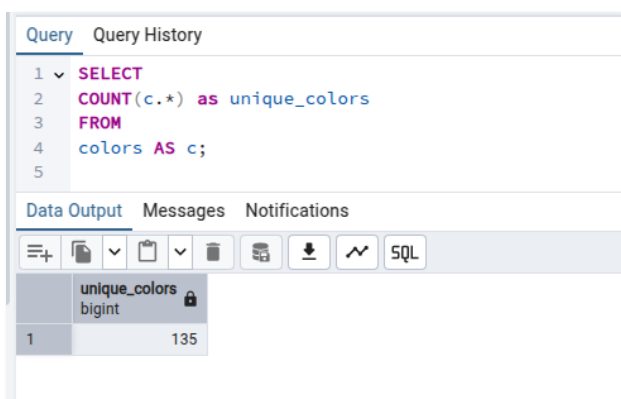
color_name	rgb	total_quantity	<color lego>
Black	05131D	62515	
Light Gray	9BA19D	37905	
White	FFFFFF	33168	
Red	C91A09	30702	
Yellow	F2CD37	25818	
Blue	0055BF	19590	
Dark Gray	6D6E5C	5478	
Green	237841	5449	
Brown	583927	3072	
Trans-Neon Green	F8F184	1735	

## 2. Colores únicos

Determinar la cantidad de colores que son únicos en toda la base de datos.

```
SELECT
COUNT(c.*) as unique_colors
FROM
colors AS c;
```

Los colores posibles están enumerados en la tabla **colors** identificados una clave unívoca. Son en total 135:



The screenshot shows a SQL query editor with a query window and a results window. The query window contains the following SQL code:

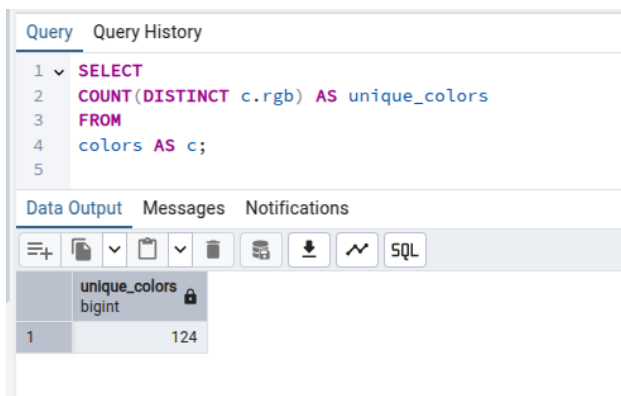
```
1 SELECT
2 COUNT(c.*) as unique_colors
3 FROM
4 colors AS c;
```

The results window shows the output of the query:

	unique_colors bigint
1	135

```
SELECT
COUNT(DISTINCT c.rgb) as unique_colors
FROM
colors AS c;
```

Cada color tiene asociado un código RGB aproximado y se observa que más de un color diferente tienen el mismo código RGB. Si considero los códigos de color únicos son 124:



The screenshot shows a SQL query editor with a query window and a results window. The query window contains the following SQL code:

```
1 SELECT
2 COUNT(DISTINCT c.rgb) AS unique_colors
3 FROM
4 colors AS c;
```

The results window shows the output of the query:

	unique_colors bigint
1	124

Considerando únicamente aquellos colores que en efecto se usaron en alguna pieza:

```
SELECT
COUNT(DISTINCT c.rgb) AS unique_used_colors
FROM
inventory_parts AS i_p
JOIN colors AS c ON i_p.color_id = c.id;

SELECT
COUNT(DISTINCT c.name) AS unique_used_colors
FROM
inventory_parts AS i_p
JOIN colors AS c ON i_p.color_id = c.id;
```

Son en realidad 131 colores diferentes y 121 códigos únicos RGB:

Query		Query History
1	SELECT	
2	COUNT(DISTINCT c.name) AS unique_used_colors	
3	FROM	
4	inventory_parts AS i_p	
5	JOIN colors AS c ON i_p.color_id = c.id;	

Data Output		Messages	Notifications
+	unique_used_colors		
	bigint		
1	131		


Query		Query History
1	SELECT	
2	COUNT(DISTINCT c.rgb) AS unique_used_colors	
3	FROM	
4	inventory_parts AS i_p	
5	JOIN colors AS c ON i_p.color_id = c.id;	

Data Output		Messages	Notifications
+	unique_used_colors		
	bigint		
1	121		

Considerando que la definición de "único" implica que ese color se usó una sola vez en una única pieza:

```
SELECT
c.name AS color_name, c.id AS color_id, c.rgb
FROM (
    SELECT color_id
    FROM inventory_parts
    GROUP BY color_id
    HAVING COUNT(*) = 1
) AS unique_colors
JOIN colors c ON unique_colors.color_id = c.id;
```

En ese caso, hay un único color usado en una sola pieza:



The screenshot shows a SQL query editor with a query window and a data output window. The query window displays the following SQL code:

```
1 SELECT
2 c.name AS color_name, c.id AS color_id, c.rgb
3 FROM (
4     SELECT color_id
5     FROM inventory_parts
6     GROUP BY color_id
7     HAVING COUNT(*) = 1
8 ) AS unique_colors
9 JOIN colors c ON unique_colors.color_id = c.id;
```

The data output window shows the results of the query. It has a table with the following columns: color\_name, color\_id, and rgb. The table contains one row of data:

	color_name character varying (30)	color_id integer	rgb character varying (8)
1	Trans Light Royal Blue	1006	B4D4F7

Según los resultados anteriores hay cuatro colores que están registrados en la tabla de colores, pero que, sin embargo, no fueron nunca utilizados en ninguna pieza.

```
SELECT
c.id, c.name, c.rgb, c.is_trans
FROM
colors AS c
WHERE
c.id NOT IN
(SELECT DISTINCT i_p.color_id FROM inventory_parts AS i_p);
```

Esos cuatro colores son:

Query

Query History

```

1 SELECT
2   c.id, c.name, c.rgb, c.is_trans
3 FROM
4   colors c
5 WHERE
6   c.id NOT IN
7     (SELECT DISTINCT i_p.color_id FROM inventory_parts AS i_p);

```

Data Output

Messages

Notifications

≡

📄

▼

📋

▼

🗑️

🗄️

⬇️

📈

SQL

Showing rows: 1 to 4

	Id [PK] integer	name character varying (30)	rgb character varying (8)	is_trans boolean
1	32	Trans-Black IR Lens	635F52	true
2	1004	Trans Flame Yellowish Orange	FCB76D	true
3	1005	Trans Fire Yellow	FBE890	true
4	1007	Reddish Lilac	8E5597	false

Trans-Black IR Lens	Trans Flame Yellowish Orange	Trans Fire Yellow	Reddish Lilac

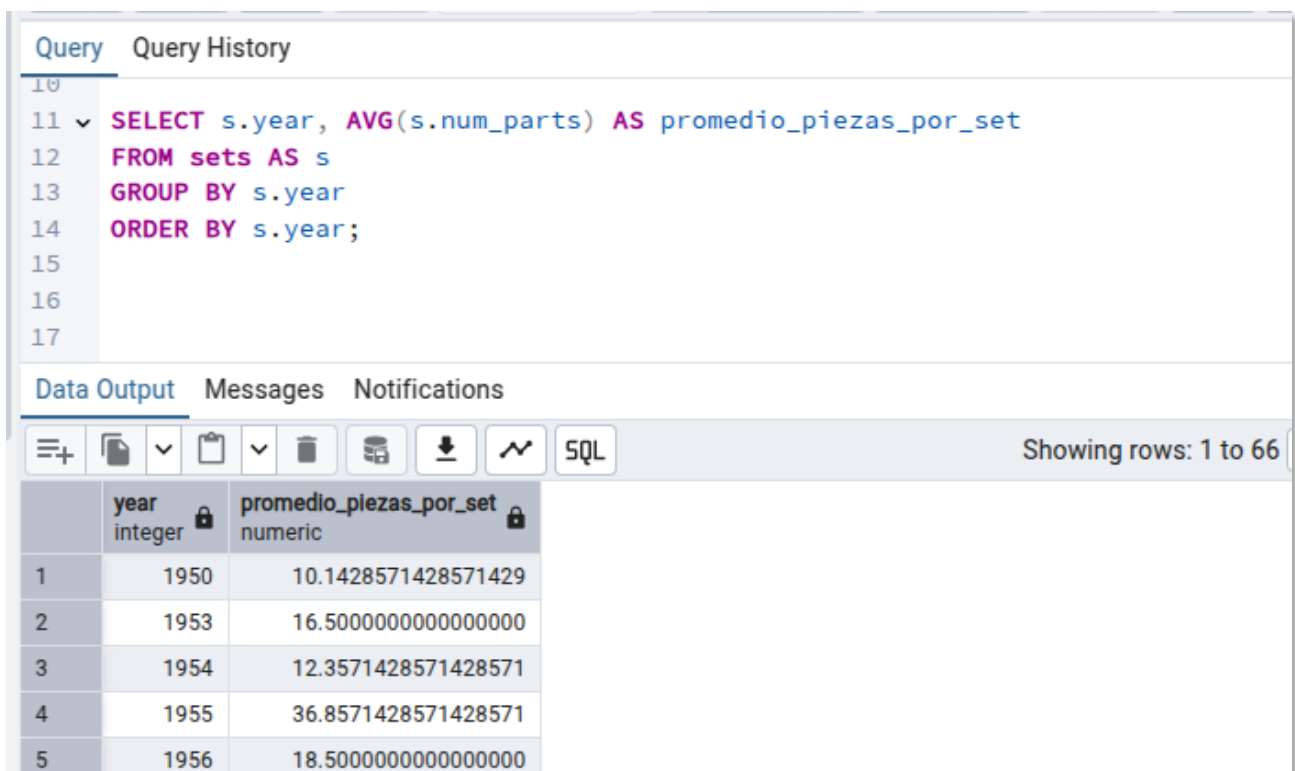
### 3. Tendencia de piezas por sets a lo largo de los años

Analizar cómo ha evolucionado la cantidad de piezas incluidas en los sets de LEGO a través del tiempo.

```
SELECT s.year, AVG(s.num_parts) AS promedio_piezas_por_set
FROM sets AS s
GROUP BY year
ORDER BY year;

SELECT s.year,
       COUNT(s.*) AS cantidad_sets,
       AVG(s.num_parts) AS promedio_piezas_por_set,
       SUM(s.num_parts) AS total_piezas
FROM sets AS s
GROUP BY s.year
ORDER BY s.year;
```

Agrupando los sets por año y calculando el promedio de piezas por set para cada año:



The screenshot shows a SQL query editor with a query window and a results window. The query window contains the following SQL code:

```
SELECT s.year, AVG(s.num_parts) AS promedio_piezas_por_set
FROM sets AS s
GROUP BY s.year
ORDER BY s.year;
```

The results window displays the output of the query, showing a table with two columns: 'year' (integer) and 'promedio\_piezas\_por\_set' (numeric). The table contains five rows of data, representing the years 1950, 1953, 1954, 1955, and 1956.

	year integer	promedio_piezas_por_set numeric
1	1950	10.1428571428571429
2	1953	16.5000000000000000
3	1954	12.3571428571428571
4	1955	36.8571428571428571
5	1956	18.5000000000000000



Obteniendo más detalles para tener una visión completa de cuántos sets se lanzaron, cuántas piezas en total, y el promedio de piezas por set cada año:

Query

Query History

1

2

3

4

5

6

7

8

SELECT

s.year,

COUNT(s.\*) AS cantidad\_sets,

AVG(s.num\_parts) AS promedio\_piezas\_por\_set,

SUM(s.num\_parts) AS total\_piezas

FROM sets AS s

GROUP BY s.year

ORDER BY s.year;

Data Output

Messages

Notifications

≡+

▼

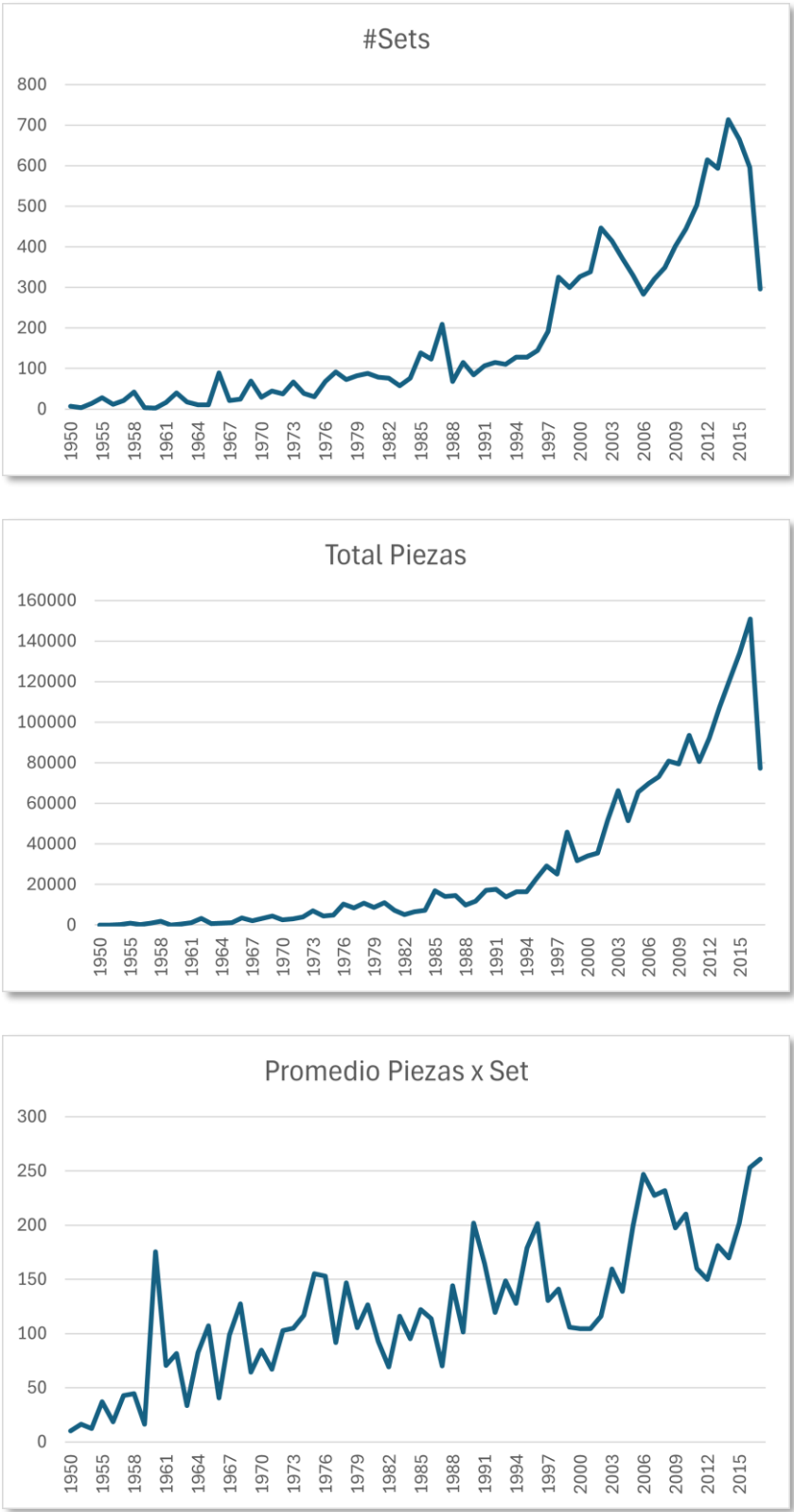
▼

SQL

Showing rows: 1 to 66

	year integer	cantidad_sets bigint	promedio_piezas_por_set numeric	total_piezas bigint
1	1950	7	10.1428571428571429	71
2	1953	4	16.5000000000000000	66
3	1954	14	12.3571428571428571	173
4	1955	28	36.8571428571428571	1032
5	1956	12	18.5000000000000000	222

Gráficas de evolución según los resultados totales del query:

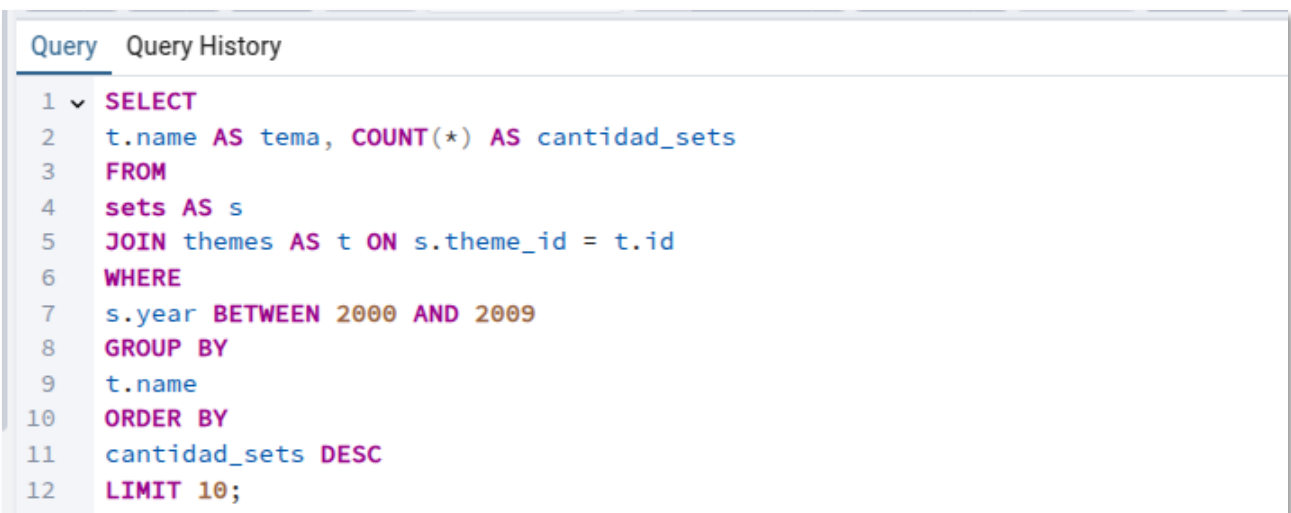


#### 4. Temáticas más populares de los 2000

Identificar cuáles fueron las temáticas de sets más populares durante la década de los 2000.

```
SELECT
t.name AS tema, COUNT(*) AS cantidad_sets
FROM
sets AS s
JOIN themes AS t ON s.theme_id = t.id
WHERE
s.year BETWEEN 2000 AND 2009
GROUP BY
t.name
ORDER BY
cantidad_sets DESC
LIMIT 10;
```

Considerando el top ten de temas usados en sets de Lego entre el 2000 y el 2009:



```
Query  Query History
1  SELECT
2  t.name AS tema, COUNT(*) AS cantidad_sets
3  FROM
4  sets AS s
5  JOIN themes AS t ON s.theme_id = t.id
6  WHERE
7  s.year BETWEEN 2000 AND 2009
8  GROUP BY
9  t.name
10 ORDER BY
11 cantidad_sets DESC
12 LIMIT 10;
```

Data Output Messages Notifications		
Showing rows: 1 to 10		
	tema character varying (50)	cantidad_sets bigint
1	City	142
2	Bulk Bricks	124
3	Basic Set	121
4	Creator	118
5	Clikits	105
6	Technic	104
7	Star Wars Episode 4/5/6	88
8	Soccer	80
9	Supplemental	73
10	Knights Kingdom II	69

Si medimos la popularidad no por cantidad de sets, sino por cantidad total de piezas en esos sets, como indicador de la relevancia resulta:

```
SELECT
t.name AS tema, SUM(s.num_parts) AS total_piezas
FROM
sets AS s
JOIN themes AS t ON s.theme_id = t.id
WHERE
s.year BETWEEN 2000 AND 2009
GROUP BY
t.name
ORDER BY
total_piezas DESC
LIMIT 10;
```

QueryQuery History

```
1 SELECT
2 t.name AS tema, SUM(s.num_parts) AS total_piezas
3 FROM
4 sets AS s
5 JOIN themes AS t ON s.theme_id = t.id
6 WHERE
7 s.year BETWEEN 2000 AND 2009
8 GROUP BY
9 t.name
10 ORDER BY
11 total_piezas DESC
12 LIMIT 10;
```

Data OutputMessagesNotifications

Showing rows: 1 to 10

	tema character varying (50)	total_piezas bigint
1	Basic Set	60781
2	Star Wars Episode 4/5/6	54586
3	Sculptures	26575
4	Construction	21041
5	Ferrari	13170
6	Traffic	13153
7	Supplemental	13123
8	Star Wars Clone Wars	11614
9	FIRST LEGO League	10537
10	Mosaic	10198

Utilizando CTEs, analizo qué temas fueron populares al aplicar la intersección de ambos criterios:

```
WITH top_x_sets AS (  
    SELECT  
        t.name AS tema  
    FROM  
        sets AS s  
    JOIN themes AS t ON s.theme_id = t.id  
    WHERE s.year BETWEEN 2000 AND 2009  
    GROUP BY t.name  
    ORDER BY COUNT(*) DESC  
    LIMIT 10  
) ,  
top_x_parts AS (  
    SELECT  
        t.name AS tema  
    FROM  
        sets AS s  
    JOIN themes AS t ON s.theme_id = t.id  
    WHERE s.year BETWEEN 2000 AND 2009  
    GROUP BY t.name  
    ORDER BY SUM(s.num_parts) DESC  
    LIMIT 10  
)  
SELECT c.tema  
FROM top_x_sets c  
JOIN top_x_parts p ON c.tema = p.tema;
```

The screenshot shows a SQL IDE interface. The top panel displays a query with line numbers 19 to 26. The query uses two Common Table Expressions (CTEs) to find themes that are both popular (by count) and have many parts (by sum). The bottom panel shows the 'Data Output' tab with a table of results.

	tema character varying (50)
1	Basic Set
2	Star Wars Episode 4/5/6
3	Supplemental

Son tres temas. Análisis en este caso cómo evolucionó la popularidad de estos tres temas durante la década propuesta:

```
WITH top_x_sets AS (  
    SELECT  
        t.name AS tema  
    FROM  
        sets AS s  
    JOIN themes AS t ON s.theme_id = t.id  
    WHERE s.year BETWEEN 2000 AND 2009  
    GROUP BY t.name  
    ORDER BY COUNT(*) DESC  
    LIMIT 10  
) ,  
top_x_parts AS (  
    SELECT  
        t.name AS tema  
    FROM  
        sets AS s  
    JOIN themes AS t ON s.theme_id = t.id  
    WHERE s.year BETWEEN 2000 AND 2009  
    GROUP BY t.name  
    ORDER BY SUM(s.num_parts) DESC  
    LIMIT 10  
)  
SELECT  
    s.year,  
    c.tema,  
    COUNT(s.set_num) AS cantidad_sets,  
    SUM(s.num_parts) AS total_piezas  
FROM  
    sets AS s  
JOIN top_x_sets c ON s.theme_id = (SELECT id FROM themes WHERE name = c.tema  
LIMIT 1)  
JOIN top_x_parts p ON s.theme_id = (SELECT id FROM themes WHERE name = p.tema  
LIMIT 1)  
WHERE  
    s.year BETWEEN 2000 AND 2009  
    AND c.tema = p.tema  
GROUP BY  
    s.year, c.tema  
ORDER BY  
    s.year, c.tema;
```

Query

Query History

```

34      AND c.tema = p.tema
35  GROUP BY
36      s.year, c.tema
37  ORDER BY
38      s.year, c.tema;

```

Data Output

Messages

Notifications

≡

📄

▼

📋

▼

🗑️

🗄️

⬇️

📈

SQL

Showing rows: 1 to 14

✎

Pag

	year integer	tema character varying (50)	cantidad_sets bigint	total_piezas bigint
1	2000	Supplemental	7	617
2	2001	Basic Set	15	3535
3	2001	Supplemental	1	41
4	2002	Basic Set	12	4724
5	2003	Basic Set	12	6767
6	2004	Basic Set	13	8807
7	2005	Basic Set	14	9405

Popularidad durante la década propuesta según los resultados totales del query:

Año	Supplemental	Basic Set	Star Wars Episode 4/5/6
2000			
2001			
2002			
2003			
2004			
2005			
2006			
2007			
2008			
2009			



## Anexos

	<ul style="list-style-type: none"><li>• <a href="#">Archivo SQL</a> para crear la DB.</li></ul>
	<ul style="list-style-type: none"><li>• <a href="#">Archivo SQL</a> con las consultas.</li></ul>
	<ul style="list-style-type: none"><li>• <a href="#">Repositorio GitHub</a> con el archivo de configuración del entorno y los inputs.</li></ul>

## Referencias

- [https://github.com/FIUBA-Posgrado-Inteligencia-Artificial/BDIA/tree/main/clase\\_3](https://github.com/FIUBA-Posgrado-Inteligencia-Artificial/BDIA/tree/main/clase_3)
- <https://www.pgadmin.org/>
- <https://www.postgresql.org/docs/17/index.html>
- [https://github.com/FIUBA-Posgrado-Inteligencia-Artificial/BDIA/tree/main/TP\\_FINAL](https://github.com/FIUBA-Posgrado-Inteligencia-Artificial/BDIA/tree/main/TP_FINAL)
- [https://www.kaggle.com/datasets/rtatman/lego-database?select=downloads\\_schema.png](https://www.kaggle.com/datasets/rtatman/lego-database?select=downloads_schema.png)